LS-OPT® User's Manual

A DESIGN OPTIMIZATION AND PROBABILISTIC ANALYSIS TOOL FOR THE ENGINEERING ANALYST

NIELEN STANDER, Ph.D.
ANIRBAN BASUDHAR, Ph.D.
WILLEM ROUX, Ph.D.
KATHARINA LIEBOLD, Dipl.-Math.
TRENT EGGLESTON, Ph.D.
TUSHAR GOEL, Ph.D.
KEN CRAIG, Ph.D.

November 2020
Version 7.0

Corporate Address
Livermore Software Technology
P. O. Box 712
Livermore, California 94551-0712

Support Addresses
Livermore Software Technology
7374 Las Positas Road
Livermore, California 94551
Tel: 925-449-2500 ♦ Fax: 925-449-2507
Email: sales@lstc.com
Website: www.lstc.com
LS-OPT support site: www.lsoptsupport.com

Livermore Software Technology
1740 West Big Beaver Road
Suite 100
Troy, Michigan 48084
Tel: 248-649-4728 ♦ Fax: 248-649-6328

Disclaimer

LS-DYNA®, LS-OPT® and LS-PrePost® are registered trademarks of Livermore Software Technology Corporation in the United States. All other trademarks, product names and brand names belong to their respective owners.

LSTC reserves the right to modify the material contained within this manual without prior notice.

The information and examples included herein are for illustrative purposes only and are not intended to be exhaustive or all-inclusive. LSTC assumes no liability or responsibility whatsoever for any direct of indirect damages or inaccuracies of any type or nature that could be deemed to have resulted from the use of this manual.

Any reproduction, in whole or in part, of this manual is prohibited without the prior written approval of LSTC. All requests to reproduce the contents hereof should be sent to sales@lstc.com.

11/16/2020

PREFACE TO VERSION 1

LS-OPT originated in 1995 from research done within the Department of Mechanical Engineering, University of Pretoria, South Africa. The original development was done in collaboration with colleagues in the Department of Aerospace Engineering, Mechanics and Engineering Science at the University of Florida in Gainesville.

Much of the later development at LSTC was influenced by industrial partners, particularly in the automotive industry. Thanks are due to these partners for their cooperation and also for providing access to high-end computing hardware.

At LSTC, the author wishes to give special thanks to colleague and co-developer Dr. Trent Eggleston. Thanks are due to Mr. Mike Burger for setting up the examples.

Nielen Stander

Livermore, CA

August, 1999

PREFACE TO VERSION 2

Version 2 of LS-OPT evolved from Version 1 and differs in many significant regards. These can be summarized as follows:

1. The addition of a mathematical library of expressions for composite functions.
2. The addition of variable screening through the analysis of variance.
3. The expansion of the multidisciplinary design optimization capability of LS-OPT.
4. The expansion of the set of point selection schemes available to the user.
5. The interface to the LS-DYNA binary database.
6. Additional features to facilitate the distribution of simulation runs on a network.
7. The addition of Neural Nets and Kriging as metamodeling techniques.
8. Probabilistic modeling and Monte Carlo simulation. A sequential search method.

As in the past, these developments have been influenced by industrial partners, particularly in the automotive industry. Several developments were also contributed by Nely Fedorova and Serge Terekhoff of SFTI. Invaluable research contributions have been made by Professor Larsgunnar Nilsson and his group in the Mechanical Engineering Department at Linköping University, Sweden and by Professor Ken Craig’s group in the Department of Mechanical Engineering at the University of Pretoria, South Africa. The authors also wish to give special thanks to Mike Burger at LSTC for setting up further examples for Version 2.

Nielen Stander, Ken Craig, Trent Eggleston and Willem Roux

Livermore, CA

January, 2003

PREFACE TO VERSION 3

The development of LS-OPT has continued with an emphasis on the integration with LS-DYNA and LS-PREPOST and differs from the previous version in the following significant regards:

1. LS-OPT is now available for Microsoft Windows.
2. Commands have been added to simplify parameter identification using continuous curves of measured data.
3. Stochastic fields have been added to LS-DYNA (Version 971) to provide the capability of modeling geometric and shell thickness variability.
4. Extended visualization of statistical quantities based on multiple runs were implemented by further integrating LS-PREPOST.
5. An internal `d3plot` interface was developed.
6. Reliability-Based Design Optimization (RBDO) is now possible using the probability of failure in the design constraints.
7. Neural network committees were introduced as a means to quantify and generalize response variability.
8. Mixed discrete-continuous optimization is now possible.
9. Parameter identification is enhanced by providing the necessary graphical pre- and postprocessing features. Confidence intervals are introduced to quantify the uncertainty of the optimal parameters.
10. The importation of user-defined sampling schemes has been refined.
11. Matrix operations have been introduced.
12. Data extraction can be done by specifying a coordinate (as an alternative to a node, element or part) to identify the spatial location. The coordinate can be referred to a selected state.
13. A simple feature is provided to gather and compress the database for portability.
14. A utility is provide to both reduce the d3plot file sizes by deleting results and to transform the d3plot results to a moving coordinate system.
15. Checking of LS-DYNA keyword files is introduced as a means to avoid common output request problems.
16. Statistical distributions can be plotted in the distribution panel in the GUI.
17. A feature is introduced to retry aborted runs on queuing systems.
18. 3-Dimensional point plotting of results is introduced as an enhancement of metamodel plotting.
19. Radial basis function networks as surrogate models.
20. Multi-objective optimization for converging to the Pareto optimal front (direct & metamodel-based).
21. Robust parameter (Taguchi) design is supported. The variation of a response can be used as an objective or a constraint in the optimization process.
22. Mapping of results to the FE mesh of the base design: the results are considered at fixed coordinates. These capabilities allow the viewing of metalforming robustness measures in LS-PREPOST.
23. The ANSA morpher is supported as a preprocessor.
24. The truncated normal distribution is supported.
25. Extra input files can be provided for variable parsing.
26. A library-based user-defined metamodel is supported.
27. User-defined analysis results can be imported.
28. PRESS predictions can be plotted as a function of the computed values.
29. The DynaStats panel has been redesigned completely (Version 3.4)

30. Strategies for metamodel-based optimization are provided as GUI options
31. An algorithm panel has been added for setting optimization algorithm parameters.
32. User-defined sampling points can be evaluated using an existing metamodel.
33. The Adaptive Simulated Annealing algorithm has been added as a core optimization solver. Hybrid algorithms such as the Hybrid SA and Hybrid GA have also been added.
34. Kriging has been updated and accelerated.
35. Enhancements were made to the Accuracy selection in the viewer by allowing color-coded point attributes such as feasibility and iteration number.
36. The Tradeoff selection has also been enhanced by converting it to a 3-D application with color coding for the 4th dimension as well as color status of points for feasibility and iteration number.

As in the past, these developments were strongly influenced by industrial partners, particularly in the automotive industry. LS-OPT is also being applied, among others, in metal forming and the identification of system and material parameters.

In addition to long-time participants: Professor Larsgunnar Nilsson (Mechanical Engineering Department, Linköping University, Sweden), significant contributions have been made by Dr. Daniel Hilding, Mr. David Björkevik and Mr. Christoffer Belestam of Engineering Research AB (Linköping) as well as Dr.-Ing. Heiner Müllerschön, Dipl.-Ing. Marko Thiele and Dipl.-Math. Katharina Witowski of DYNA*more* GmbH, Stuttgart, Germany.

Nielen Stander, Willem Roux and Tushar Goel

Livermore, CA

January, 2009

PREFACE TO VERSION 4

The development of LS-OPT has continued with an emphasis on the integration with LS-DYNA and LS-PREPOST. The main focus of Version 4 has been the development of a new graphical postprocessor as well as the improvement of the job scheduling system, especially with regard to scheduling on computer clusters. The following features have been added:

Version 4.0:

1. The Viewer has been redesigned completely to accommodate a multi-window format using a split-window and detachable window feature.
2. The Correlation matrix for simulation variables and results has been added.
3. For visualizing the Pareto Optimal Frontier, Hyper-Radial Visualization and Parallel Coordinate plots have been added to the more traditional scatter plot. Multiple points can be selected to create a table of response values. Point highlighting is cross-connected between plot types.
4. An interface for the METAPost postprocessor has been added.
5. Topology optimization LS-OPT®/Topology has been added as a separate module. Please refer to the LS-OPT/Topology User's Manual.
6. Many of the features such as the Reliability-Based Design Optimization have been significantly accelerated.
7. The Blackbox queuing system has been streamlined in terms of providing better diagnostics and a special queuing system *Honda* has been added.
8. The NASTRAN® interface for frequency extraction and mode tracking has been added.

Version 4.1:

1. Discrete sampling can be done on a variable by variable basis for most sampling schemes including *D*-Optimality, Space Filling and Full Factorial.
2. The Space Filling algorithm has been improved for accuracy and speed.
3. Job scheduling has been significantly improved. Environment variables can be exported through queuing systems.
4. Job data is displayed on the run progress bars with a selection to view the solver log file at any stage of the run.
5. Three injury criteria: a3ms, Chest Compression and Viscous Criterion have been added.
6. SPH, DBBEMAC and NODFOR groups have been added to the LS-DYNA response interface.
7. GenEx, the LS-OPT Generic Extractor provides features for extracting entities from text files. This allows LS-OPT to be used with any solver code that produces a text database.
8. Responses can be linked to LS-DYNA cases (*CASE keyword).

9. In addition to polynomials, Radial Basis Functions can now be used for parameter identification.
10. The following features have been added to the Viewer: Self-Organizing Maps (for multi-objective optimization), two-dimensional interpolation matrix using metamodels, global sensitivities (Sobol), Computed (simulation) and Predicted (metamodel) histories, Parallel Coordinate plot for simulation results.
11. Experiments can be replicated for stochastic fields. Improvements have been made to Stochastic Fields (*PERTURBATION) in LS-DYNA. Special coordinate systems have been added. *PERTURBATION_MATERIAL has been added for MAT24.
12. To avoid synchronization errors, the Experiments and AnalysisResults databases have been converted to self-contained .csv files.
13. The Run page has been rationalized. Clean start options are now available for all tasks.
14. A selected subset of Pareto optimal points can be exported to a standard format. The file can be used to schedule the points as simulations.

Version 4.2:

1. The algorithm for constrained experimental design has been greatly improved. An optimization algorithm was introduced to locate design points within specified constraint bounds.
2. LSTCVM has been added as a Secure Proxy Server for distributing solver jobs across a computer cluster. Running LS-OPT on a Windows machine controlling solver jobs on a Linux cluster is now possible.
3. Individual jobs can be stopped using LSKILLJOB from the LS-OPT GUI. This feature has been implemented to kill lagging jobs which tend to hold up the entire optimization run. Accelerated job killing is provided as an option. A job can also be flagged for restart. LSTCVM and LSKILLJOB combined with LSCHEDULER and other auxiliary programs provide a sophisticated job distribution system.
4. More injury criteria are now available, namely MOC, NNIC, NIC, Nkm, LNLI, TTI and TI. A 3-node version of the injury criterion *Clip3m* has been added.
5. Kinematics for NODOUT-based responses and histories. Includes the calculation of deformation and distance in global, local and local-in-reference-frame coordinate systems.
6. DBFSI (fluid structure interaction) is available in the history and response interfaces.
7. Curve Mapping has been added to improve the curve matching metric for material identification, especially for hysteretic curves, curves with steep sections and cases where only partial test data is available. A newly developed Partial Curve Mapping algorithm is used.
8. Metamodel prediction accuracy based on PRESS error has been added as a stopping criterion for the Sequential Response Surface Method (SRSM).
9. Automatic internal constraint scaling based on the constraint bounds has been added to the GUI. This feature ensures that constraint violations are treated equally irrespective of their magnitudes.
10. The *Dominated Hypervolume* method as a stopping criterion for multi-objective optimization methods (GA). *Crowding Distance* and *Spread* of the Pareto Optimal Front can be monitored graphically.
11. Self-Organizing Maps is available to visualize simulation results.

12. Refinements have been made to the 2D Metamodel Cross-Section display by adding simulation points. The History display was improved by allowing the selection and display of multiple histories. There is stronger unification amongst the different types of displays.
13. LS-OPT database archiving has been expanded to include extra files such as solver input files.
14. Histories have been added to the GenEx (generic extraction) result extraction feature. In the past, only responses could be extracted.
15. The input file environment can be used to store include files. LS-OPT will in this case automatically be able to parse and transmit the files (e.g. to a cluster).
16. A derivative history function has been added to compute the derivative of a time history, e.g. acceleration from velocity.
17. A general filtering feature for time histories has been added. Filtering has been available for LS-DYNA-extracted data, but can now be applied to any time history, also those produced using expressions or generic extraction.

Version 4.3

1. The MAC criterion replaces the Generalized Mass criterion for mode tracking (merged to Version 4.2). An option to turn off mode tracking was added.
2. Mode tracking is supported for all versions of LS-DYNA, including LS-DYNA MPP (merged to Version 4.2).
3. Sampling of the Pareto Optimal Front as a sampling option. A Space Filling algorithm, to maximize the distance between any two points in the design space, is used.
4. Option for selecting the number of verification runs for the trade-off curve of multi-objective optimization. Space Filling sampling is done to obtain a well-distributed trade-off set.
5. Head injury criterion (HIC) using three nodes for the different coordinate directions.
6. Support Vector Regression introduced as a metamodeling type.
7. User-defined postprocessor option.

The automotive and other industries have again made significant contributions to the development of new features in LS-OPT. In addition to long-time participant Professor Larsgunnar Nilsson (Mechanical Engineering Department, Linköping University, Sweden), Dr. Daniel Hilding, Mr. David Björkevik and Mr. Christoffer Belestam of Engineering Research AB (Linköping) as well as Dr.-Ing. Heiner Müllerschön and Dipl.-Math. Katharina Witowski of DYNA*more* GmbH, Stuttgart, Germany have made major contributions as developers. Dr. Trent Eggleston has recently created LSTCVM and LSKILLJOB and, while working with customers, has made vast improvements to solver job scheduling via queuing systems.

Nielen Stander and Anirban Basudhar

Livermore, CA

August, 2012

PREFACE TO VERSION 5

The development of LS-OPT has continued with an emphasis on the integration with LS-DYNA. The main focus of Version 5 has been the development of a new graphical pre-processor to accommodate design processes, in which the design stages are dependent on one another, as well as the improvement of the job scheduling system to enable handling of job dependencies. Transparency of the job scheduling process has also been improved. The following features have been added:

Version 5.0:

1. A process consisting of a chain of dependent stages can be analyzed. The process can be defined in the form of a flow chart which can merge and branch. Solver stages have been added as a new concept and building block for defining a flow chart.
2. File operations such as deleting and copying between dependent stages are available.
3. GUI features have been added to easily identify sources of design parameters.
4. Job monitoring has been enhanced by allowing progress visualization on a stage-by-stage basis. Any run directory can be viewed.
5. Resource definitions have been added to enhance the concurrent job submission capability.
6. Variables can be de-activated arbitrarily using a table of checkboxes. This avoids the necessity for changing variables to constants.
7. New metal forming failure criteria.
8. String variables. These variables allow the definition of discrete variables sets with names as might be used for include file names. GUI support is provided.
9. The recovery of databases from remote servers has been added as a GUI feature.
10. A sorting feature has been added to the Correlation Matrix in the Viewer. The cross-correlations for any entity can be sorted.

Version 5.1:

1. Multilevel optimization. An LS-OPT solver type can be selected to allow the nesting of any LS-OPT task.
2. Parallel Feedforward Neural Networks. This feature allows the concurrent building of multiple networks and network ensemble components. FFNN building can also be done remotely, e.g. on a cluster. Job monitoring is provided in the GUI.
3. Significant enhancements have been made to histogram displays in the Viewer. Manual axis control is allowed while statistical quantities such as mean and standard deviation as well as constraints are depicted. Histogram types have been added.
4. Subregion-based sensitivity analysis is available using Sobol indices. Multiple subregions can be analyzed in the same run and stored for display. Global Sensitivity Analysis can now be activated from the GSA icon (as a post-processing function).

5. Design categories can be specified for user-selected simulation points. Name, color and type attributes can be chosen for each category display. Responses and histories are supported.
6. Excel is now supported as a solver type on Windows.
7. A new third party Finite Element solver is now supported. The support includes parameter (*PARAMETER) recognition using recursive include files during the problem setup phase.
8. De-activation of variables in iterative methods. The user can seamlessly deactivate variables at any stage of the iterative process. This is useful when performing other tasks such as optimization after variable screening.
9. Metamodel formulae for polynomials and Radial Basis Function Networks can be exported.
10. Multiple plots are allowed in optimization history displays. All the available entities such as variables, responses, etc. can be displayed on the same plot.
11. Differential Evolution was added as a global metamodel optimizer (unconstrained continuous problems only).
12. Responses and/or histories can be cloned (Stage dialog in GUI).

Version 5.2:

1. A new integrated progress window has been created to unify Windows and Linux progress monitoring. The window features separate tabs for text output and thermometer type progress monitoring. Warnings and errors are displayed in a separate tab window. Global progress is displayed. The window can be hidden while the older option via the stage dialog LED is still available.
2. Navigation tools are available in the GUI for navigating between levels when using multilevel optimization. E.g. the input setup and progress can be accessed with full functionality for lower levels by navigating from the start (top) level.
3. Response variables were created to allow the substitution of simulation results in input files of a child stage during a multi-stage process flow. Histories from simulation output can also be transferred to LS-DYNA input files as *DEFINE_CURVE data sets. Response and history expressions are fully supported.
4. The generation and display of comparison metamodels. A set of different metamodels based on the same set of analysis results can be selected by the user for display. Parallel Neural Networks are also available as comparison metamodels.
5. Histories can be displayed in three dimensions in which the third dimension is a variable.
6. Reliability statistics, e.g. as a result of direct or metamodel-based Monte Carlo analysis can be extracted in a multilevel setup. This allows the setup of, for instance, tolerance optimization or robust design problems using the direct Monte Carlo method. Mean values, standard deviation as well as the probability of failure are supported for individual constraints as well as globally.
7. Matlab is supported as a solver type on the Windows platform.
8. LS-OPT metamodels (DesignFunctions.*x* file format) can be imported. This is useful for problems in which a metamodel has already been constructed. Importing and optimization/Monte Carlo analysis

can be executed as a single step to allow for automatic importation preceding the inner level analysis of a multi-level optimization.

9. Parallelization is now automatic for extraction repair. The number of processors available on the local machine is automatically detected.
10. Mode tracking now runs in parallel.
11. Box plot options are available for histogram displays (reliability analysis). This includes whisker type options for min./max., interquartile range, standard deviation and 9%/91%.
12. The FE postprocessor can be customized.
13. Encryption features are available to encrypt the LS-OPT (.lsopt) input file.
14. The efficiency of the Curve Mapping algorithm has been improved.
15. Features have been added to the GenEx text extraction tool to simplify the selection and extraction of histories.
16. A response file option allows the specification of an output file with a single value that needs to be extracted (user-defined response). This feature solves a portability issue by obviating “type” (Windows) or “cat” (Linux) commands to write such a file to standard output as is required for user-defined responses.
17. Retry and timeout attributes required by the job scheduler to handle abnormal termination can now be specified in the GUI.
18. Special functions for differentiation have been improved. Irregular spacing of the history or crossplot curve is allowed.
19. An image of the flow chart can be saved as a picture file.

As in previous years, the automotive and other industries have made significant contributions to the development of new features in LS-OPT. In addition to long-time participant Professor Larsgunnar Nilsson (Mechanical Engineering Department, Linköping University, Sweden), Dr. Daniel Hilding, Mr. David Björkevik, Mr. Åke Svedin and Mr. Christoffer Belestam of DYNA*more* Nordic, Linköping as well as Dr.-Ing. Heiner Müllerschön and Dipl.-Math. Katharina Witowski of DYNA*more* GmbH, Stuttgart, Germany have made major contributions as developers. Special thanks go to Katharina for patiently editing and managing the manual, a major task in this version.

Dr. Trent Eggleston redesigned the job scheduler to accommodate the launching and load balancing of jobs with dependencies. Thanks also go to Prof. Satoshi Kitayama of Kanazawa University, Japan for providing the Differential Evolution algorithm.

Nielen Stander and Anirban Basudhar

Livermore, CA

July, 2015

PREFACE TO VERSION 6

The following features have been added:

Version 6.0:

1. *Classifiers* have been introduced to provide a new constraint handling approach for optimization or probabilistic analysis. The basic idea is to construct a decision boundary in the design space that predicts whether a given design is feasible or not. The support vector classification algorithm, a pattern recognition and machine learning method, is now available to approximate the boundaries. It focuses only on the decision boundary based on the pre-defined feasibility criteria and not on the prediction of response values. This approach is especially attractive for discontinuous or binary responses, and in the case of multi-disciplinary failure criteria.
2. *Multi-point histories* have been introduced to accommodate the full-field response of shell structures. The multi-point histories are defined at a user-specified coordinate set. Element or nodal values can be extracted from the LS-DYNA d3plot database. These can be used for parameter identification using Digital Image Correlation. A truncation function is available for filtering the multi-point histories.
3. *Similarity measures for curve comparison.* Based on multi-point histories, general expressions including Mean Squared Error, Discrete Fréchet and Dynamic Time Warping similarity measures as well as cross-plotting functions have been added. Multi-point cross-plots can be constructed to incorporate both histories and multi-point histories. The similarity measures can be used to compute the distance between any two curves or multi-point curves. The new expressions also apply to curves and multi-point curves imported from files. DTW is highly effective for addressing curves with a combination of noise and hysteresis.
4. *Digital Image Correlation*. Parameter identification can be done using full-field experimental data. Generic file-based as well as standard interfaces have been added for extracting full-field data. GenEx has been extended to accommodate multi-point histories. The gom/ARAMIS DIC system is supported.
5. *Binary databases* have been introduced as a replacement of critical older XML databases to speed up interactivity during post-processing and to accommodate the potential growth in data volume brought about by the introduction of full-field data. Temporary databases (such as the JobResults and multihistory databases) have been converted to binary, and are also deleted after use.
6. *Taguchi Method* is now available as a classical parametric study method. It uses orthogonal arrays to study the effects of variables on the responses. These arrays enable the calculation of main and some interaction effect using a relatively small number of samples, but do not calculate higher order effects.
7. *Efficient Global Optimization* (EGO) has been added as an optimization strategy.
8. *Stage library.* To allow standardization of a design problem setup, solver stages or processes (stage groups) can be imported and exported to and from user-specified locations.
9. *Interactive tables*. Previously available simple (dumb) tables for design data have been enhanced to assume a more spreadsheet-like behavior. Multiple tables can be generated. Tables now allow new design point generation in a selected region of interest as well as the simulation of newly generated points. Using an existing metamodel, responses can be interpolated using user-specified parameter values. The highlighting of infeasibility has been enhanced and table rows can be re-ordered according

to a selected column. As in the past, tables also interact with plots. The *point categories* feature has been enhanced. The current development progression is intended as a precursor to an independent graphical tool including many functions currently only available in the pre-processor GUI.

10. LS-TaSC has been added as a solver.
11. *Training examples* are packaged together with the LS-OPT installation and can be run from the starting menu.

The authors wish to thank Åke Svedin and Christoffer Belestam of DYNA*more* Nordic, Linköping as well as Katharina Witowski and Charlotte Keisser of DYNA*more* GmbH, Stuttgart who have made major contributions as developers. We also wish to thank interns Sophie Du Bois and Denis Kirpicev for their contributions to the development of similarity measures used in material identification applications.

As in the past, special thanks go to Katharina for patiently editing and managing the manual.

Nielen Stander and Anirban Basudhar

Livermore, CA

July, 2019

PREFACE TO VERSION 7

Version 7.0:

1. *Job scheduler.* This is a complete redesign. It includes SSH Proxy, new Blackbox features and new GUI dialogs. Supports user options for normal, error and abnormal termination when employing user-defined solvers.
2. *Principal Component Analysis.* Can be applied to histories and multi-point spatial responses.
3. *Process management:* Disable/enable samplings or process stages. Accompanied by automatic expression deactivation.
4. Sequential metamodel-based probabilistic analysis.
5. Classifier-based sampling constraints for adaptive sampling.
6. *Distribution fitting* (Normal and Weibull) using user-defined observation data.
7. *Multi-point responses* as a complement to the existing multi-point histories. Can be included in expressions and used in conjunction with PCA.
8. *Partial curve mapping* algorithm DTW-P for material identification. Applied to full-field calibration (multi-point histories).
9. *LS-OPT Command-line options.* General options such as cleaning a previous run, repair and archiving options can be specified on the command line.
10. Interface to Oasys PRIMER.
11. *Updated graphics library* (wxWidgets 3.0) used for the GUI.

The authors wish to thank Åke Svedin of DYNA*more* Nordic, Linköping as well as Katharina Witowski and Charlotte Keisser of DYNA*more* GmbH, Stuttgart and Versailles respectively, who have made major contributions as developers. We also wish to thank interns Sophie Du Bois and Denis Kirpicev for their contributions to the development of similarity measures used in material identification applications.

Nielen Stander and Anirban Basudhar

Livermore, CA

August, 2020

TABLE OF CONTENTS

1. Introduction

In the conventional design approach, a design is improved by evaluating its response and making design changes based on experience or intuition. This approach does not always lead to the desired result, that of a ‘best’ design, since design objectives are sometimes in conflict, and it is not always clear how to change the design to achieve the best compromise of these objectives. A more systematic approach can be obtained by using an inverse process of first specifying the criteria and then computing the ‘best’ design. The procedure by which design criteria are incorporated as objectives and constraints into an optimization problem that is then solved, is referred to as optimal design.

The state of computational methods and computer hardware has only recently advanced to the level where complex nonlinear problems can be analyzed routinely. Many examples can be found in the simulation of impact problems and manufacturing processes. The responses resulting from these time-dependent processes are, as a result of behavioral instability, often highly sensitive to design changes. Program logic, as for instance encountered in parallel programming or adaptivity, may cause spurious sensitivity. Roundoff error may further aggravate these effects, which, if not properly addressed in an optimization method, could obstruct the improvement of the design by corrupting the function gradients.

Among several methodologies available to address optimization in this design environment, *response surface methodology (RSM)*, a statistical method for constructing smooth approximations to functions in a multi-dimensional space, has achieved prominence in recent years. Rather than relying on local information such as a gradient only, RSM selects designs that are optimally distributed throughout the design space to construct approximate surfaces or ‘design formulae’. Thus, the local effect caused by ‘noise’ is alleviated and the method attempts to find a representation of the design response within a bounded design space or smaller region of interest. This extraction of global information allows the designer to explore the design space, using alternative design formulations. For instance, in vehicle design, the designer may decide to investigate the effect of varying a mass constraint, while monitoring the crashworthiness responses of a vehicle. The designer might also decide to constrain the crashworthiness response while minimizing or maximizing any other criteria such as mass, ride comfort criteria, etc. These criteria can be weighted differently according to importance and therefore the design space needs to be explored more widely.

Part of the challenge of developing a design program is that designers are not always able to clearly define their design problem. In some cases, design criteria may be regulated by safety or other considerations and therefore a response has to be constrained to a specific value. These can be easily defined as mathematical constraint equations. In other cases, fixed criteria are not available but the designer knows whether the responses must be minimized or maximized. In vehicle design, for instance, crashworthiness can be constrained because of regulation, while other parameters such as mass, cost and ride comfort can be treated as objectives to be incorporated in a multi-objective optimization problem. Because the relative importance of various criteria can be subjective, the ability to visualize the trade-off properties of one response vs. another becomes important.

Trade-off curves are visual tools used to depict compromise properties where several important response parameters are involved in the same design. They play an extremely important role in modern design where design adjustments must be made accurately and rapidly. Design trade-off curves are constructed using the principle of *Pareto* optimality. This implies that only those designs of which the improvement of one response will necessarily result in the deterioration of any other response are represented. In this sense no further improvement of a Pareto optimal design can be made: it is the best compromise. The designer still has a choice of designs but the factor remaining is the subjective choice of which feature or criterion is more important than another. Although this choice must ultimately be made by the designer, these curves can be helpful by limiting the number of possible solutions. An example in vehicle design is the trade-off between mass (or energy efficiency) and safety.

Adding to the complexity, is the fact that mechanical design is really an interdisciplinary process involving a variety of modeling and analysis tools. To facilitate this process, and allow the designer to focus on creativity and refinement, it is important to provide suitable interfacing utilities to integrate these design tools. Designs are bound to become more complex due to the legislation of safety and energy efficiency as well as commercial competition. It is therefore likely that in future an increasing number of disciplines will have to be integrated into a particular design. This approach of multidisciplinary design requires the designer to run more than one case, often using more than one type of solver. For example, the design of a vehicle may require the consideration of crashworthiness, ride comfort, noise level as well as durability. Moreover, the crashworthiness analysis may require more than one analysis case, e.g. frontal and side impact. It is therefore likely that as computers become more powerful, the integration of design tools will become more commonplace, requiring a multidisciplinary design interface as well as a process flow interface to allow for design stages.

Modern architectures feature remote clusters with multiple processors and all indications are that the demand for distributed computing will strengthen into the future. Optimization and RSM in particular, lend themselves very well to being applied in distributed computing environments because of the low level of message passing. Response surface methodology is efficiently handled, since each design can be analyzed independently during a particular iteration. Needless to say, sequential methods have a smaller advantage in distributed computing environments than global search methods such as RSM.

LS-OPT also features Monte Carlo based point selection schemes and optimization methods. The respective relevance of stochastic and response surface based methods may be of interest. In a pure response surface based method, the effect of the variables is distinguished from chance events while Monte Carlo simulation is used to investigate the effect of these chance events. The two methods should be used in a complimentary fashion rather than substituting the one for the other. In the case of events in which chance plays a significant role, responses of design interest are often of a global nature (being averaged or integrated over time). These responses are mainly deterministic in character. The full vehicle crash example in this manual can attest to the deterministic qualities of intrusion and acceleration pulses. These types of responses may be highly nonlinear and have random components due to uncontrollable noise variables, but they are not random.

Stochastic methods have an important purpose when conducted directly or on the surrogate (approximated) design response in reliability based design optimization and robustness improvement.

1.1. Overview of the manual

This LS-OPT® manual consists of four main parts.

I - User’s Manual

This part guides the user in the use of LS-OPT*ui*, the graphical user interface.

II - Examples

Examples are used to illustrate the application of LS-OPT to a variety of practical applications.

III - Theory

Fundamentals are provided for the various features in LS-OPT.

IV - Appendix

Appendices contain interface features, database file descriptions, a mathematical expression library, a Glossary, etc. Two appendices are dedicated to helping the user install LS-OPT. The second of these is more advanced and dedicated to remote job scheduling, e.g. using a queuing system.

1.2. How to read this manual

Most users will start learning LS-OPT by consulting the User’s Manual section beginning with Chapter 2 (Getting Started).

The Examples (Chapters 20 through 22) are included to demonstrate the features and capabilities and can be read together with Chapters 2 to 19 to help the user to set up a problem formulation.

The Theoretical Manual (Chapters 23 through 29) serves mainly as an in-depth reference section for the underlying methods.

The items in the Appendices are included for reference to detail, while the Appendix J: Document Type Definition (DTD) provides an overview of all the features.

The manual functions as a hypertext document such that links in the manual body can be used for cross-referencing and will take the reader to the relevant item such as Section 3.2.1, Reference [4] or Figure 24-5 (just click on any of the afore-mentioned references). **Alt+Left Arrow** returns to the original reference point.

Sections containing advanced topics are indicated with an asterisk (*).

I – User's Manual

2. Getting Started

2.1. Installation of LS-OPT

Refer to Appendix H: (Installing LS-OPT) for information on the installation of LS-OPT.

2.2. LS-OPT execution commands

Table 2-1 describes the LS-OPT execution commands.

Table 2-1: LS-OPT execution commands

Command	**Description**
lsoptui [*file_name.lsopt]*	Execute the graphical user interface
lsopt [options] *file_name.lsopt*	LS-OPT batch execution. For command line options see Section 2.2.1.
viewer [options] *file_name.lsopt*	Execute the graphical postprocessor (also accessible from main GUI). For command line options see Section 16.1.7.
com2lsopt *com.abcde abcde.lsopt*	Converts a legacy 'com' file (LS_OPT version < 5.0) to a .lsopt file in XML format

2.2.1. Command line options

Table 2-2, Table 2-3 and Table 2-4 describe the command line options for the executable *lsopt*.

Table 2-2: general lsopt command line options

Command	**Description**
-b, --baseline	Run only baseline run

--clean	Remove all results. The directory structure created by LS-OPT and all the files in this directory structure are deleted. Use –y option to omit asking for confirmation.
--clean-from=ITER	Removes all simulation data as well as optimization data from the specified iteration ITER onwards. Use –y option to omit asking for confirmation.
--clean-verif	Removes the simulation data as well as optimization data of the verification run. Use –y option to omit asking for confirmation.
-y, --yes	Run clean without asking for confirmation. Only valid with –clean, --clean-from or –clean-verif.
--env	Show the LS-OPT environment settings and exit. The LS-OPT environment is automatically set to the location of the lsopt executable.
--version	Show version information and exit
-h, --help	Show available command line options and exit

Table 2-3: Archive command line options, Section 3.6

Command	**Description**
-a, --archive	Archive ls-opt database
-H, --histresp	Include histories and responses. Only valid with –a option.
-I, --inputs	Include input deck and extra input files. Only valid with –a option.
-F, --file=FILE	Additional file to pack. Can be specified multiple times. Only valid with –a option.

Table 2-4: Repair command line options, Section 3.5

Command	**Description**
--add-points	Add points. Use –sampling to run operation locally for a specific sampling.
--read-points	Read points. Use –sampling to run operation locally for a specific sampling.

--import-points	Import results. Use –sampling to run operation locally for a specific sampling.
--run-jobs	Run jobs
--rerun-failed-jobs	Rerun failed jobs. User –stage to run operation locally for a specific stage.
--extract	Extract results. User –stage to run operation locally for a specific stage.
--rerun-verif	Rerun verification run
--build-mm	Build metamodels. Use –sampling to run operation locally for a specific sampling.
--eval-mm	Evaluate metamodels
--import-mm	Import metamodels. Use –sampling to run operation locally for a specific sampling.
--calc-globalsens	Calculate global sensitivites
--build-classifiers	Build classifiers
--optimize	Optimize
--iter=ITER	Repair location. Required for all repair options except --rerun-verif.
--sampling=NAME	Repair location. Only valid with sampling specific options.
--stage=NAME	Repair location. Only valid with stage specific options.

Examples

```
lsopt –b project.lsopt
```

runs only the baseline run in batch mode.

```
lsopt –a –H project.lsopt
```

archives the LS-OPT database including histories and responses.

```
lsopt --extract --iter=3
```

re-extracts the results for iteration 3 for all stages.

2.3. Name conventions in LS-OPT

2.3.1. Variable names

Variables as defined in this section are entities that can be used in mathematical expressions.

Variables are identified by their names. A name length is limited to *61* characters. In addition to numbers `0-9`, upper or lower case letters, a name can contain a period (`.`) and/or an underscore (**_**). Spaces are not allowed.

The leading character of a variable must be *alphabetical*. Variables must be given unique names, because mathematical expressions can be constructed using various entities in the same formula.

2.3.2. Stage and sampling names

For entities that cannot be used in mathematical expressions, i.e. `stage`, `sampling`, `distribution` and `resource`, the name can also include the characters **`-+%=`**. Spaces are not allowed.

Stage and Sampling names are limited by the software to 1023 characters (no spaces allowed). These names are used as sub-directory names, so stricter limits may apply depending on the operating system.

2.3.3. Environment variable names

`Envvar` names may also include **`-+%`**.

2.4. A modus operandi for design using response surfaces

2.4.1. Preparation for design

Since the design optimization process is expensive, the designer should avoid discovering major flaws in the model or process at an advanced stage of the design. Therefore the procedure must be carefully planned and the designer needs to be familiar with the model, procedure and design tools well in advance. The following points are considered important:

1. The user should be familiar with and have confidence in the accuracy of the model (e.g., finite element model) used for the design. Without a reliable model, the design would make little or no sense.
2. Select suitable criteria to formulate the design. The responses represented in the criteria must be produced by the analyses and be accessible to LS-OPT.
3. Request the necessary output from the analysis program and set appropriate time intervals for time-dependent output. Avoid unnecessary output as a high rate of output will rapidly deplete the available storage space.
4. Run at least one simulation using LS-OPT (baseline design). To save time, the termination time of the simulation can be reduced substantially. This exercise will test the response extraction commands and various other features. Automated response checking is available, but manual checking is still recommended.

5. Just as in the case of traditional simulation it is advisable to dump restart files for long simulations. LS-OPT will automatically restart a design simulation if a restart file is available. For this purpose, the `runrsf` file is required when using LS-DYNA as solver.
6. Determine suitable design parameters. In the beginning, it is important to select many rather than few design variables. If more than one discipline is involved in the design, some interdisciplinary discussion is required with regard to the choice of design variables.
7. Determine suitable starting values for the design parameters. The starting values are an estimate of the optimum design. These values can be acquired from a present design if it exists. The starting design will form the center point of the first region of interest.
8. Choose a design space. This is represented by absolute bounds on the variables that you have chosen. The responses may also be bounded if previous information of the functional responses is available. Even a simple approximation of the design response can be useful to determine approximate function bounds for conducting an analysis.
9. Choose a suitable starting design range for the design variables. The range should be neither too small, nor too large. A small design region is conservative but may require many iterations to converge or may not allow convergence of the design at all. It may be too small to capture the variability of the response because of the dominance of noise. It may also be too large, such that a large modeling error is introduced. This is usually less serious as the region of interest is gradually reduced during the optimization process.
10. If the user has trouble deciding the size of the starting range, it should be omitted. In this case the full design space is chosen.
11. Choose a suitable order for the design approximations when using polynomial response surfaces (the default). A good starting approximation is linear because it requires the least number of analyses to construct. However, it is also the least accurate. The choice therefore also depends on the available resources. However, linear experimental designs can be easily augmented to incorporate higher order terms.

Before choosing a metamodel, please also consult Sections 24.3 and 26.5.

After suitable preparation, the optimization process may now be commenced. At this point, the user has to decide whether to use an automated iterative procedure (Section 24.3) or whether to firstly perform variable screening (through ANOVA or Global Sensitivity Analysis) based on one or a few iterations. Variable screening is important for reducing the number of design variables, and therefore the overall computational time. Variable screening is illustrated in two examples (see Sections 20.5 and 20.6).

An automated iterative procedure can be conducted with any choice of approximating function. It automatically adjusts the size of the subregion and automatically terminates whenever the stopping criterion is satisfied. The feature that reduces the size of the subregion can also be overridden by the user so that points are sequentially added to the full design space. This becomes necessary if the user wants to explore the design space such as constructing a Pareto Optimal front. If a single optimal point is desired, it is probably the best to use a sequential linear approximation method with domain reduction, especially if there is a large number of design variables. See also Section 26.5.

A step-by-step semi-automated procedure can be just as useful, since it allows the designer to proceed more resourcefully. Computer time can be wasted with iterative methods, especially if handled carelessly. It mostly pays to pause after the first iteration to allow verification of the data and design formulation and inspection of the results, including ANOVA and GSA data. In many cases, it takes only 2 to 3 iterations to achieve a reasonably optimal design. An improvement of the design can usually be achieved within one iteration.

A suggested step-by-step semi-automated procedure is outlined as follows:

2.4.2. A step-by-step design optimization procedure

1. Evaluate as many points as required to construct a linear approximation. Assess the accuracy of the linear approximation using any of the error parameters. Inspect the main effects by looking at the ANOVA and GSA results. This will highlight insignificant variables that may be removed from the problem. An ANOVA/GSA is simply a single iteration run, typically using a linear response surface to investigate main and/or interaction effects. The ANOVA and GSA results can be viewed in the post-processor of LS-OPT (see Section 16.3.4).

2. If the linear approximation is not accurate enough, add enough points to enable the construction of a quadratic approximation. Assess the accuracy of the quadratic approximation. Intermediate steps can be added to assess the accuracy of the interaction and/or elliptic approximations. Radial Basis Functions (Section 24.1.3) can also be used as more flexible higher order functions (They do not require a minimum number of points).

3. If the higher order approximation is not accurate enough, the problem may be twofold:

   - There is significant noise in the design response.
   - There is a *modeling* error, i.e. the function is too nonlinear and the subregion is too large to enable an accurate quadratic approximation.

   In case (3a), different approaches can be taken. Firstly, the user should try to identify the source of the noise, e.g. when considering acceleration-related responses, was filtering performed? Are sufficient significant digits available for the response in the extraction database (not a problem when using LS-DYNA since data is extracted from a binary database)? Is mesh adaptivity used correctly? Secondly, if the noise cannot be attributed to a specific numerical source, the process being modeled may be chaotic or random, leading to a noisy response. In this case, the user could implement reliability-based design optimization techniques as described in Section 29.8. Thirdly, other less noisy, but still relevant, design responses could be considered as alternative objective or constraint functions in the formulation of the optimization problem.

   In case (3b), the subregion can be made smaller.

   In most cases the source of discrepancy cannot be identified, so in either case a further iteration would be required to determine whether the design can be improved.

4. Optimize the approximate subproblem. The solution will be either in the interior or on the boundary of the subregion.

   If the approximate solution is in the interior, the solution may be good enough, especially if it is close to the starting point. It is recommended to analyze the optimum design to verify its accuracy. If the accuracy of any of the functions in the current subproblem is poor, another iteration is required with a reduced subregion size.

   If the solution is on the boundary of the subregion the desired solution is probably beyond the region. Therefore, if the user wants to explore the design space more fully, a new approximation has to be built. The accuracy of the current response surfaces can be used as an indication of whether to reduce the size of the new region.

The whole procedure can then be repeated for the new subregion and is repeated automatically when selecting a larger number of iterations initially.

2.5. Recommended test procedure

A full optimization run can be very costly. It is therefore recommended to proceed with care. Check that the LS-OPT optimization run is set up correctly before commencing to the full run. By far the most of the time should be spent in checking that the optimization runs will yield useful results. A common problem is to not check the robustness of the design so that some of the solver runs are aborted due to unreasonable parameters which may cause distortion of the mesh, interference of parts or undefinable geometry.

The following general procedure is therefore recommended:

1. Test the robustness of the analysis model by running a few (perhaps two or three) designs in the extreme corners of the chosen design space. Run these designs to their full term (in the case of time-dependent analysis). Two important designs are those with all the design variables set at their minimum and maximum values. The starting design can be run by selecting *Baseline Run* from the control bar **Run** menu.
2. Modify the input to define the experimental design for a full analysis.
3. For a time dependent analysis or non-linear analysis, reduce the termination time or load significantly to test the logistics and features of the problem and solution procedure.
4. Execute LS-OPT with the full problem specified and monitor the process.

Also refer to Section 2.2.1.

2.6. Pitfalls in design optimization

A number of pitfalls or potential difficulties with optimization are highlighted here. The perils of using numerical sensitivity analysis have already been discussed and will not be repeated in detail.

2.6.1. Global optimality

The Karush-Kuhn-Tucker conditions govern the local optimality of a point. However, there may be more than one optimum in the design space. This is typical of most designs, and even the simplest design problem (such as the well known 10-bar truss sizing problem with 10 design variables), may have more than one optimum. The objective is, of course, to find the global optimum. Many gradient-based as well as discrete optimal design methods have been devised to address global optimality rigorously, but as there is no mathematical criterion available for global optimality, nothing short of an exhaustive search method can determine whether a design is optimal or not. Most global optimization methods require large numbers of function evaluations (simulations). In LS-OPT, global optimality is treated on the level of the approximate subproblem through a multi-start method originating at all the experimental design points. If the user can afford to run a direct optimization procedure, a Genetic Algorithm (Section 26.9) can be used.

2.6.2. Noise

Although noise may evince the same problems as global optimality, the term refers more to a high frequency, randomly jagged response than an undulating one. This may be largely due to numerical round-off and/or chaotic behavior. Even though the application of analytical or semi-analytical design sensitivities for 'noisy' problems is currently an active research subject, suitable gradient-based optimization methods which can be applied to impact and metal-forming problems are not likely to be forthcoming. This is largely because of the continuity requirements of optimization algorithms and the increased expense of the sensitivity analysis. Although fewer function evaluations are required, analytical sensitivity analysis is costly to implement and probably even more costly to parallelize.

2.6.3. Non-robust designs

Because RSM is a global approximation method, the experimental design may contain designs in the remote corners of the region of interest which are prone to failure during simulation (aside from the fact that the designer may not be remotely interested in these designs). An example is the identification of the parameters of a monotonic load curve which in some of the parameter sets proposed by the experimental design may be non-monotonic. This may cause unexpected behavior and possible failure of the simulation process. This is almost always an indication that the design formulation is non-robust. In most cases poor design formulations can be eliminated by providing suitable constraints to the problem and using these to limit future experimental designs to a 'reasonable' design space (see Section 23.2.9).

2.6.4. Impossible designs

The set of impossible designs represents a 'hole' in the design space. A simple example is a two-bar truss structure with each of the truss members being assigned a length parameter. An impossible design occurs when the design variables are such that the sum of the lengths becomes smaller than the base measurement, and the truss becomes unassemblable. It can also occur if the design space is violated resulting in unreasonable variables such as non-positive sizes of members or angles outside the range of operability. In complex structures it may be difficult to formulate explicit bounds of impossible regions or 'holes'.

2.6.5. Non-unique designs

In some cases multiple solutions will give the same or similar values for the objective function. The phenomenon often appears in under-defined parameter identification problems. The underlying problem is that of a singular system of equations having more than one solution. The symptoms of non-uniqueness are:

- Different solutions are found having the same objective function values
- The confidence interval for a non-linear regression problem is very large, signaling a singular system

For nonlinear regression problems, the user should ensure that the test/target results are sufficient. It could be that the data set is large but that some of the parameters are insensitive to the functions corresponding to the data. An example is the determination of the Young's modulus (E) of a material, but having test points only in the plastic range of deformation (see example Section 21.1). In this case the response functions are insensitive to E and will show a very high confidence interval for E (Section 21.1.4).

The difference between a non-robust design and an impossible one is that the non-robust design may show unexpected behavior, causing the run to be aborted, while the impossible design cannot be synthesized at all. Impossible designs are common in mechanism design.

2.7. Setup of a simple optimization problem

2.7.1. Working directory

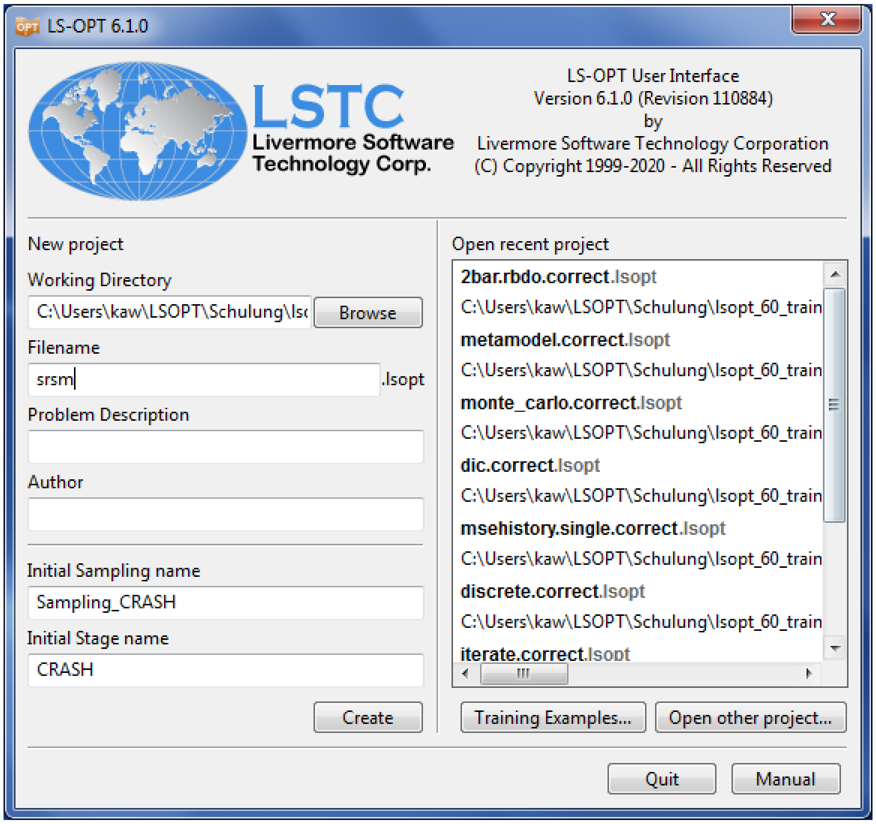
Create a working directory for keeping the main command file, input files and other command files as well as the LS-OPT program output. Make sure there are no blanks in the path names.

2.7.2. Startup

Open the graphical user interface of LS-OPT as described in Section 3.1 and enter the required specifications to generate an LS-OPT project file to start from, Figure 2-1. Selecting *Create* will open up the main LS-OPT GUI window, Figure 2-2.

Several training examples are accessible from the button *Training Examples …* and can be opened in the main GUI directly, Section 2.8.

This manual is accessible from the button *Manual*.



LS-OPT 6.1.0

LS-OPT User Interface
Version 6.1.0 (Revision 110884)
by
Livermore Software Technology Corporation
(C) Copyright 1999-2020 - All Rights Reserved

New project

Working Directory: C:\Users\kaw\LSOPT\Schulung\lso

Filename: srsm .lsopt

Problem Description:

Author:

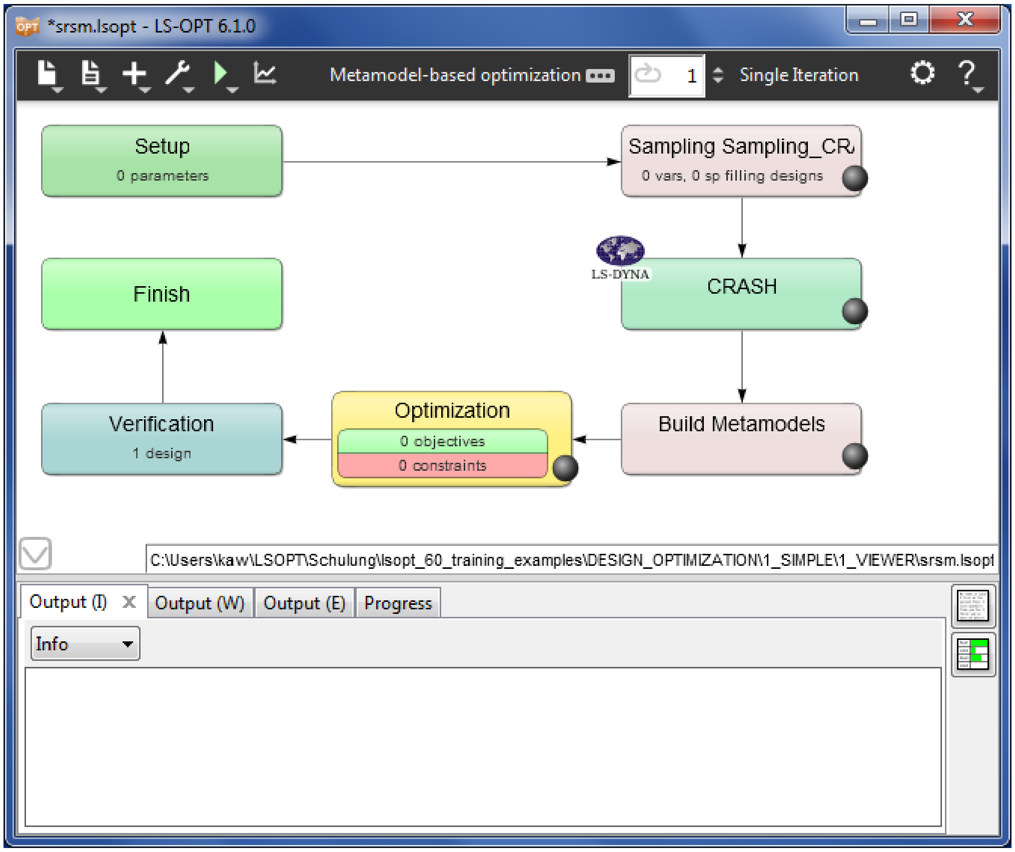
Initial Sampling name: Sampling_CRASH

Initial Stage name: CRASH

Open recent project

2bar.rbdo.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
metamodel.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
monte_carlo.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
dic.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
msehistory.single.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
discrete.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
iterate.correct.lsopt

Figure 2-1: LS-OPT Startup dialog. Select the working directory, enter a name for the LS-OPT project file and a name for the initial sampling and initial stage to generate a new project.

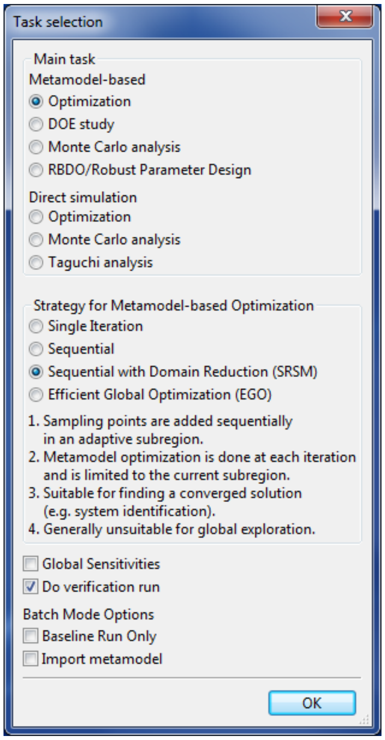


C:\Users\kaw\LSOPT\Schulung\lsopt_60_training_examples\DESIGN_OPTIMIZATION\1_SIMPLE\1_VIEWER\srsm.lsopt

Figure 2-2: The main LS-OPT GUI window visualizes the optimization process flow. Selecting a box opens the respective dialog. The stage box (CRASH) can be moved freely using the left mouse button.

2.7.3. Task

Open the **Task** dialog by selecting the corresponding icon from the control bar (▪▪▪). Select the task to run, Figure 2-3, e.g. *Metamodel-based Optimization* with Strategy: *Sequential with Domain Reduction*, Chapter 4. The main GUI displays the process flow of the selected task.



Task selection

Main task

Metamodel-based

● Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

○ Taguchi analysis

Strategy for Metamodel-based Optimization

○ Single Iteration

○ Sequential

● Sequential with Domain Reduction (SRSM)

○ Efficient Global Optimization (EGO)

1. Sampling points are added sequentially in an adaptive subregion.
2. Metamodel optimization is done at each iteration and is limited to the current subregion.
3. Suitable for finding a converged solution (e.g. system identification).
4. Generally unsuitable for global exploration.

□ Global Sensitivities

☑ Do verification run

Batch Mode Options

□ Baseline Run Only

□ Import metamodel

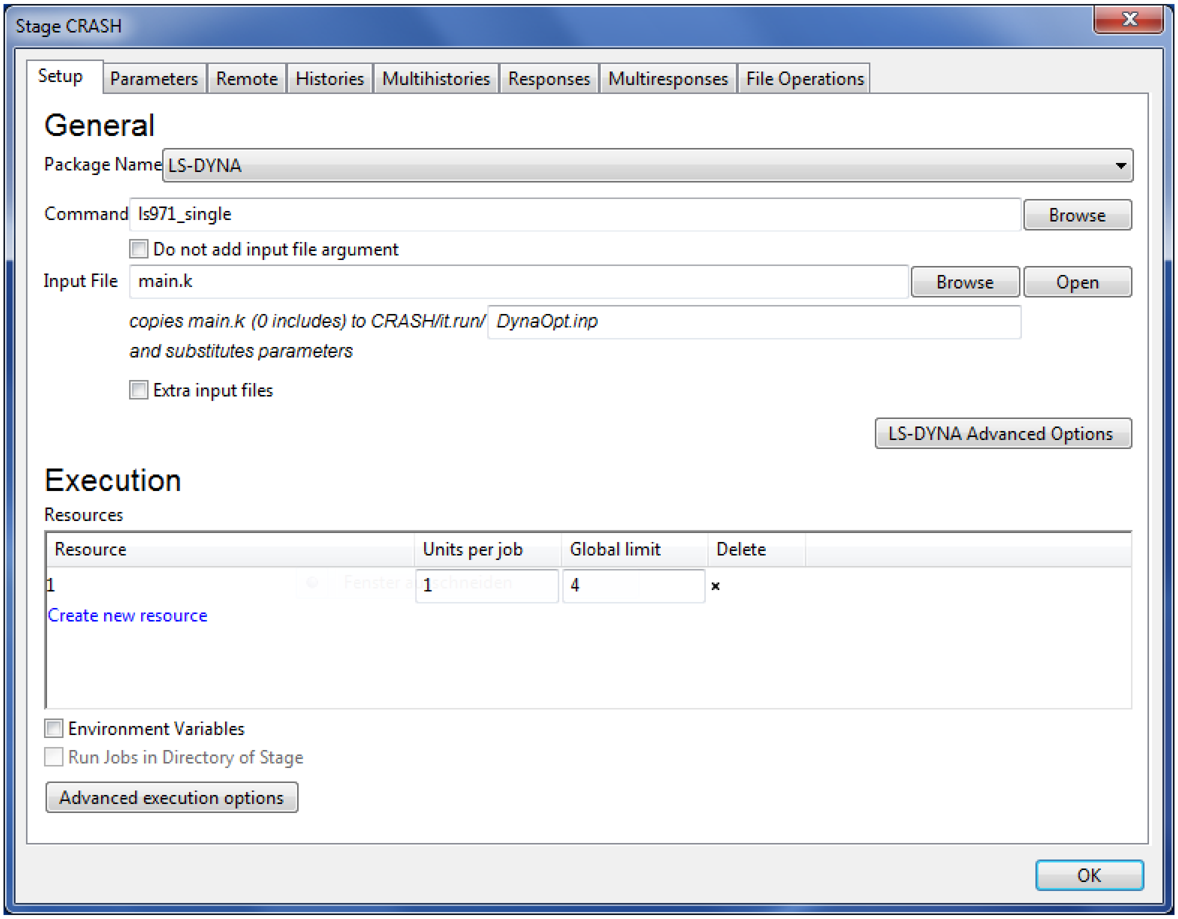
Figure 2-3: Task dialog. Select the main task and strategy

2.7.4. Stage

Set up the process chain. In the simplest case, a single **Stage** is required to interface with a solver, e.g. LS-DYNA. Select the already available **Stage** box, Figure 2-4. Select the solver *Package Name*, the solver *Command* and the parameterized *Input File*, Chapter 5. In more complex cases further stages can be added, e.g. for a pre-processor or post-processor.

Then switch to the **Parameters** tab to check the parameters found in the solver input file, Figure 2-5.

Next, switch to the **Responses** and **Histories** panel, Figure 2-6, to define results to be extracted from the solver output database (to be used as objectives or constraints in the optimization phase), Chapter 6.



General

Package Name LS-DYNA

Command ls971_single

□ Do not add input file argument

Input File main.k

copies main.k (0 includes) to CRASH/it.run/ *DynaOpt.inp*
and substitutes parameters

□ Extra input files

Execution

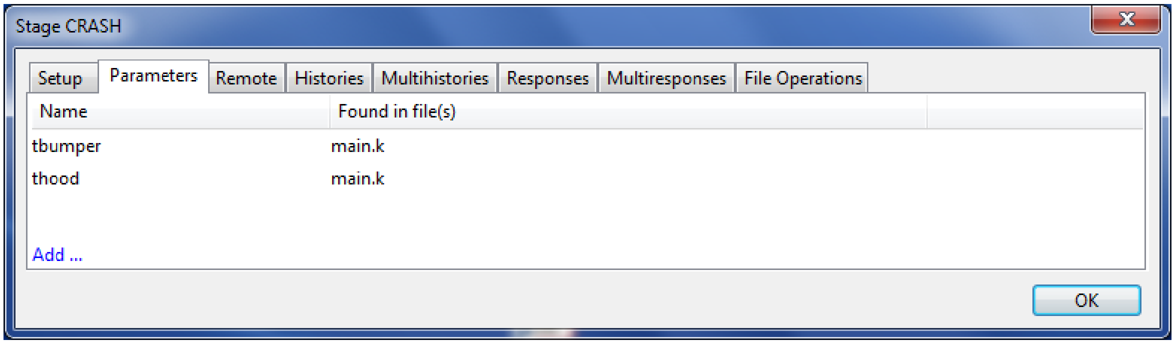
Resources

Resource	Units per job	Global limit	Delete
1	1	4	×

□ Environment Variables

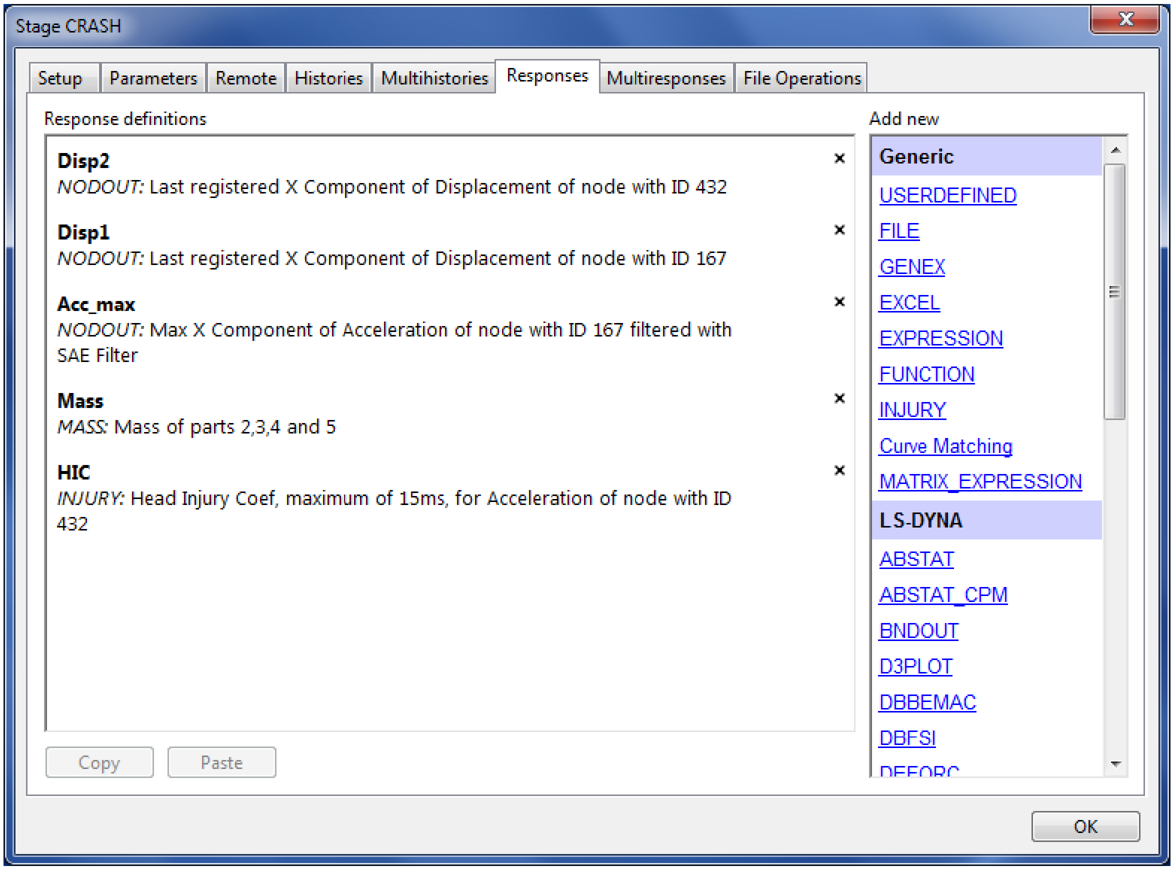
□ Run Jobs in Directory of Stage

Figure 2-4: Stage dialog - Setup. Select the solver package name, the command and the solver input file



Name	Found in file(s)
tbumper	main.k
thood	main.k

Figure 2-5: Stage dialog – Parameters. Displays the parameters found in the input file specified in Setup



Response definitions

Disp2
NODOUT: Last registered X Component of Displacement of node with ID 432

Disp1
NODOUT: Last registered X Component of Displacement of node with ID 167

Acc_max
NODOUT: Max X Component of Acceleration of node with ID 167 filtered with SAE Filter

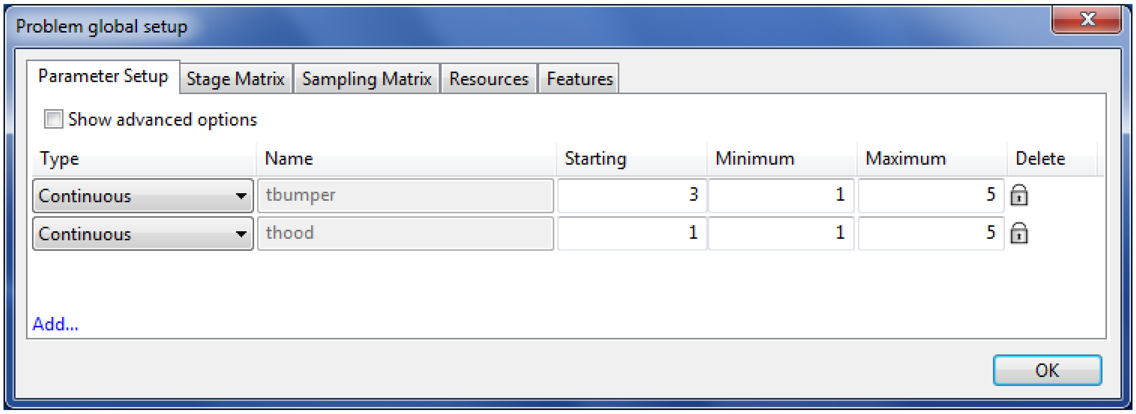
Mass
MASS: Mass of parts 2,3,4 and 5

HIC
INJURY: Head Injury Coef, maximum of 15ms, for Acceleration of node with ID 432

Figure 2-6: Stage dialog - Responses page. Select a response type from the list on the right to add a new response definition.

2.7.5. Setup

Select the **Setup** box at the top left of the main GUI, Chapter 8. All parameters that are defined in stage input files should automatically be available as constants, Figure 2-7.



□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	

Figure 2-7: Parameter Setup dialog. Define the parameter type and required values.

Select the desired variable **Types**. In most cases *Continuous* variables are used.

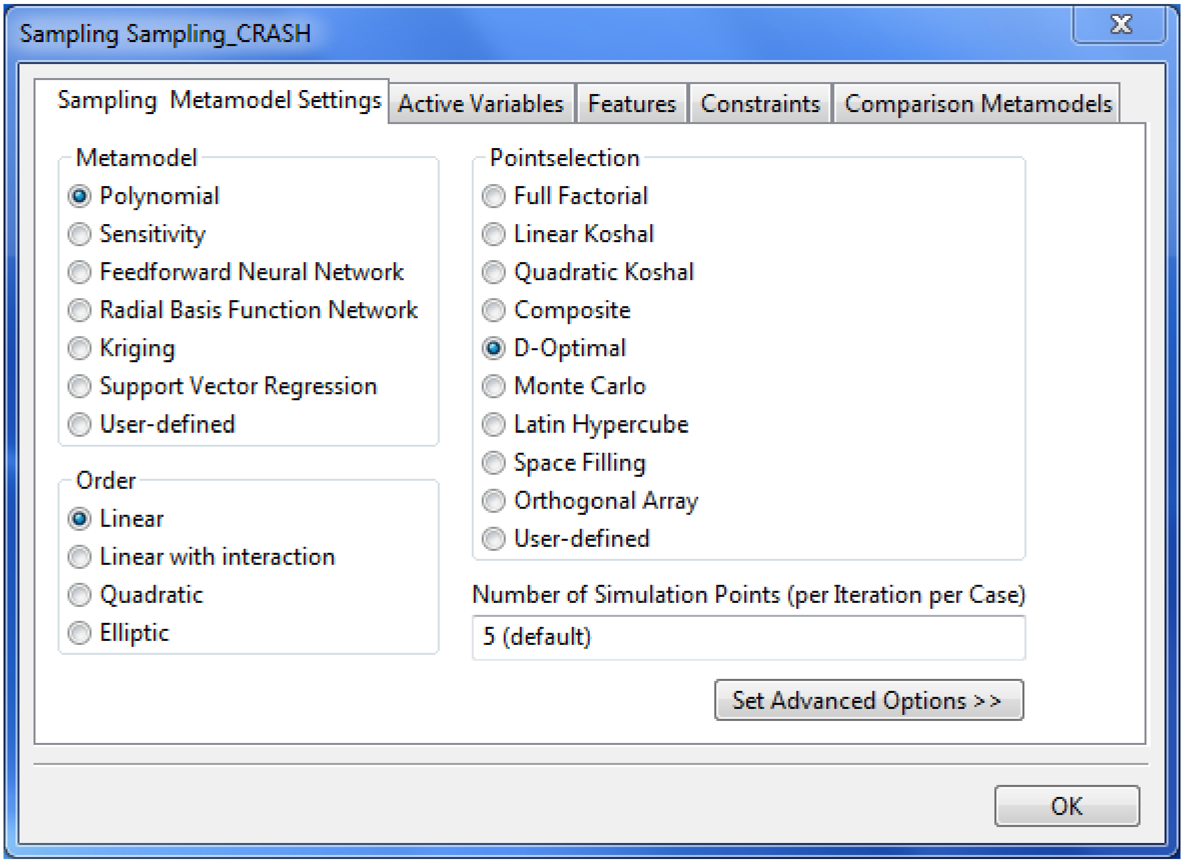
Then enter the requested values, e.g. the *Starting* value and *Minimum* and *Maximum* values to define the design space for a continuous variable.

Now follow the arrows to the next box in the optimization process flow to define the respective settings and options.

2.7.6. Sampling and Metamodels

Select the **Sampling** box, Chapter 9. Select the *Metamodel* and *Point Selection* types, or just use the default values, Figure 2-8.

The **Build Metamodels** box is coupled to the same dialog as the **Sampling** box. It is displayed at the end of the process to correctly represent the optimization process. Hence the Build Metamodels box can be skipped.



Sampling Sampling_CRASH

Sampling Metamodel Settings | Active Variables | Features | Constraints | Comparison Metamodels

Metamodel
- ● Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Order
- ● Linear
- ○ Linear with interaction
- ○ Quadratic
- ○ Elliptic

Pointselection
- ○ Full Factorial
- ○ Linear Koshal
- ○ Quadratic Koshal
- ○ Composite
- ● D-Optimal
- ○ Monte Carlo
- ○ Latin Hypercube
- ○ Space Filling
- ○ Orthogonal Array
- ○ User-defined

Number of Simulation Points (per Iteration per Case)

5 (default)

Set Advanced Options >>

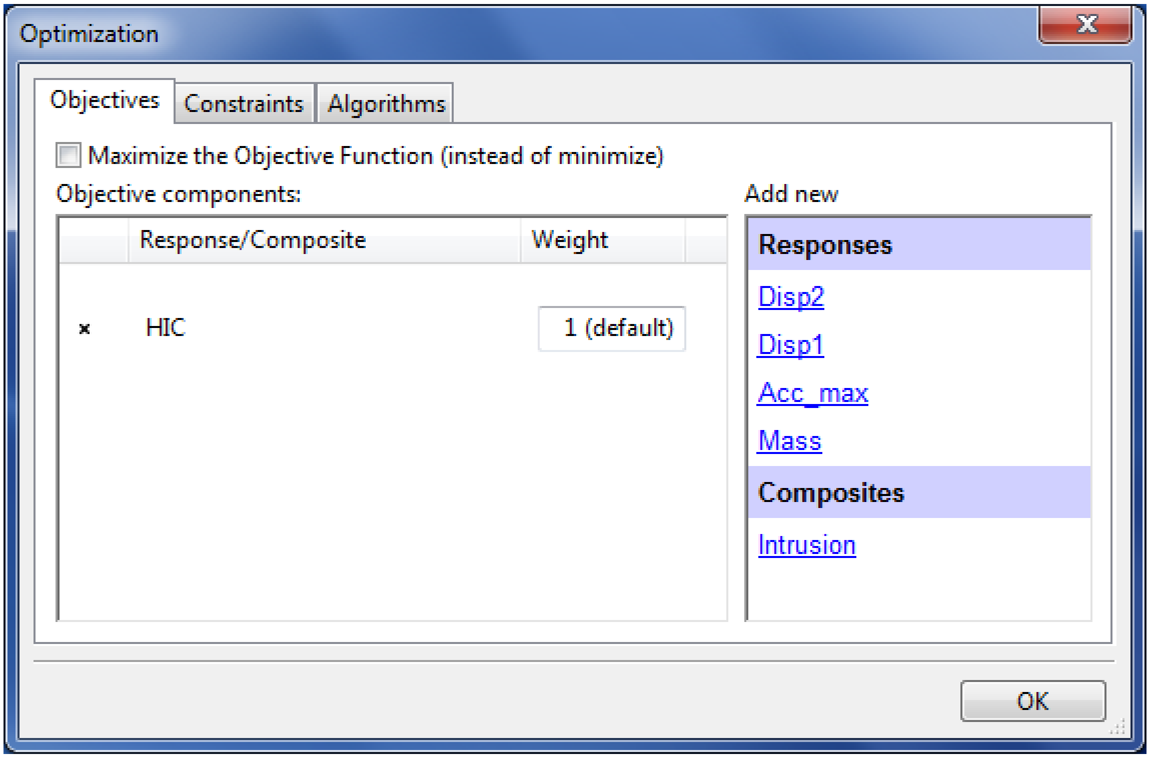
OK

Figure 2-8: Sampling dialog. Select the metamodel type and point selection scheme.

2.7.7. Optimization

Select the **Optimization** box, Chapter 12. From the previously defined *Responses*, select the objectives, Figure 2-9.

Switch to the **Constraints** tab. From the previously defined *Responses*, select the constraints and specify lower and upper bounds, respectively, Figure 2-10. Use the default setting for the algorithm.



□ Maximize the Objective Function (instead of minimize)

Objective components:

	Response/Composite	Weight
×	HIC	1 (default)

Add new

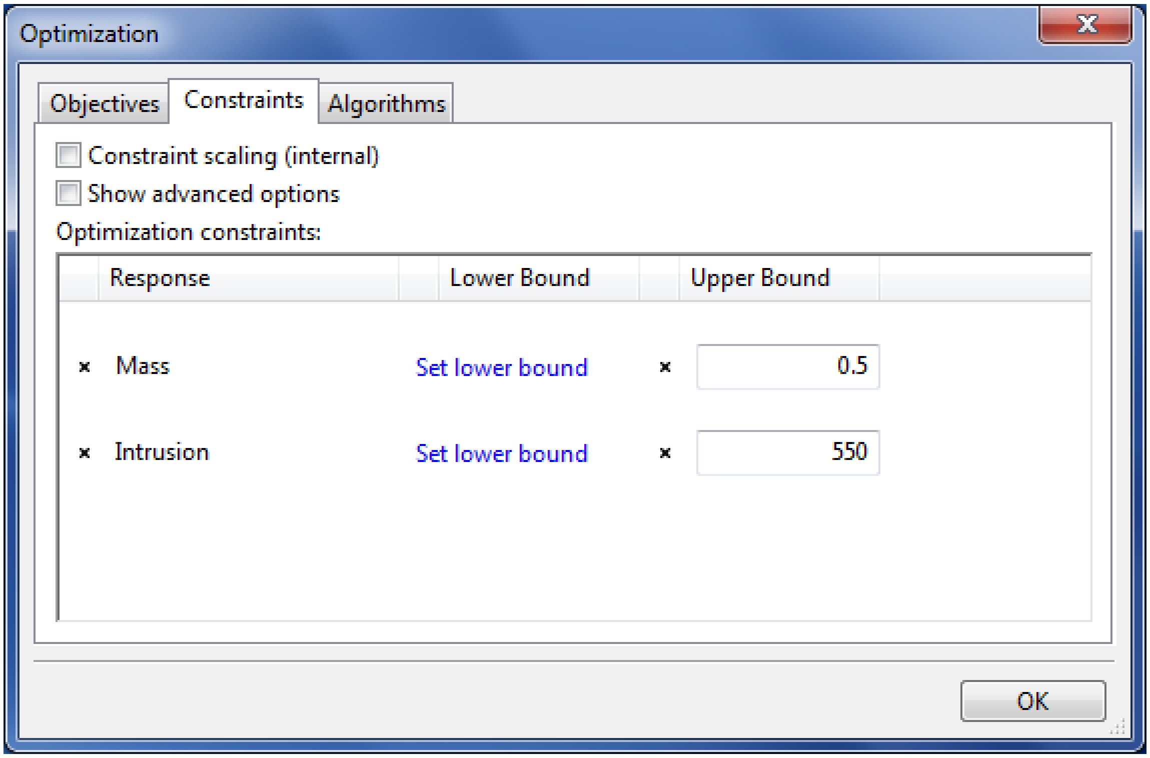
Responses

- Disp2
- Disp1
- Acc_max
- Mass

Composites

- Intrusion

Figure 2-9: Optimization - Objectives. Select the objective components from the list on the right.



□ Constraint scaling (internal)

□ Show advanced options

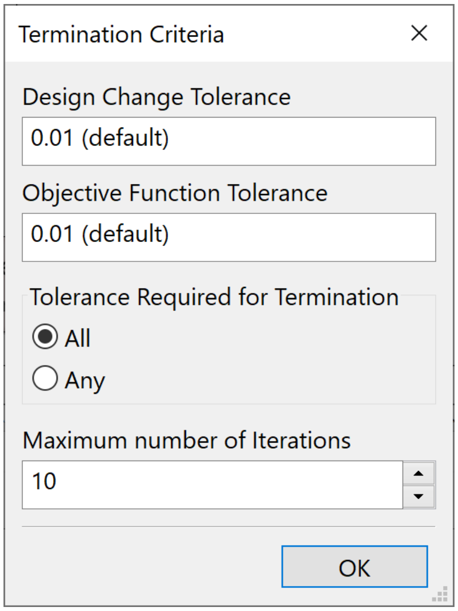
Optimization constraints:

	Response	Lower Bound		Upper Bound
×	Mass	Set lower bound	×	0.5
×	Intrusion	Set lower bound	×	550

Figure 2-10: Optimization - Constraints. Select constraints from the list on the right. Specify lower and upper bounds as required.

2.7.8. Termination criteria

Select the Termination criteria box, Chapter 13. Specify the *Maximum number of Iterations*, e.g. 10 iterations. Use the default values for the other options.



Termination Criteria

Design Change Tolerance: 0.01 (default)

Objective Function Tolerance: 0.01 (default)

Tolerance Required for Termination

● All
○ Any

Maximum number of Iterations: 10

Figure 2-11: Termination Criteria dialog. Specify the maximum number of iterations

2.7.9. Run

After setting up the optimization problem, run the task using the options from the control bar **Run** menu (▶), Section 3.3.

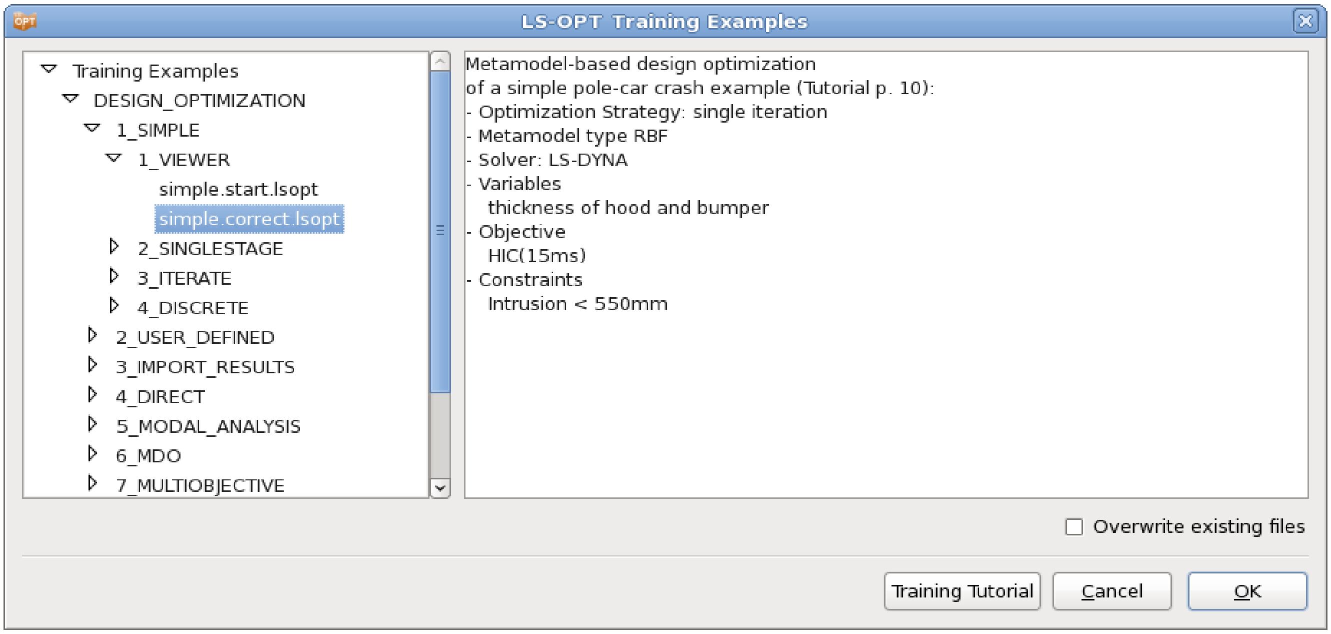
It is recommended to first run a *Baseline Run* to check if the stage process chain works correctly and the results are extracted as expected. Then run the full task using the *Normal Run* option.

2.7.10. Viewer

Use the Viewer (Chapter 16) to evaluate the results by selecting 📈 from the main GUI window control bar. The Viewer provides features to display metamodels and plot simulation results and optimization progress.

2.8. Training examples

Training examples are accessible from Startup dialog. Pressing the button *Training Examples* … opens the LS-OPT Training Examples dialog, Figure 2-12 with the list of available examples on the left, and a description of the selected example on the right.



Metamodel-based design optimization
of a simple pole-car crash example (Tutorial p. 10):
- Optimization Strategy: single iteration
- Metamodel type RBF
- Solver: LS-DYNA
- Variables
  thickness of hood and bumper
- Objective
  HIC(15ms)
- Constraints
  Intrusion < 550mm

Figure 2-12: Training Examples dialog

The tutorial accessible through the button *Training Tutorial* explains how to set up and evaluate the training examples.

3. Graphical User Interface

This chapter introduces the graphical user interface of LS-OPT. The LS-OPT GUI enables the user to construct a simulation process, using a flowchart to define the stage dependencies. The process can then be subjected to any of the available analysis tasks such as simulation, optimization, Monte Carlo analysis, etc. Using progress bars and LEDs, the GUI also provides a window on the progress of each of the optimization steps and simulation stages.

3.1. LS-OPT user interface (LS-OPTui)

On Linux, the user interface is launched with the command

```
lsoptui [command_file.lsopt]
```

On Windows, the user interface is launched using *lsoptui.exe*. A command file can be opened directly by drag and drop or by double-clicking on the `.lsopt` filename.

If the user interface is launched without a command file argument, the *Startup Dialog* opens up, where the user can either define a new LS-OPT project, or select an existing project to open, see Figure 3-1. The options are explained in Table 3-1. Otherwise the specified LS-OPT project is opened in the user interface, Figure 3-2.

Legacy `com.abcde` files generated with previous LS-OPT versions (4.x and older) can be opened with the command

```
lsoptui [com.abcde]
```

Saving the GUI contents produces a file `abcde.lsopt` in .xml format.

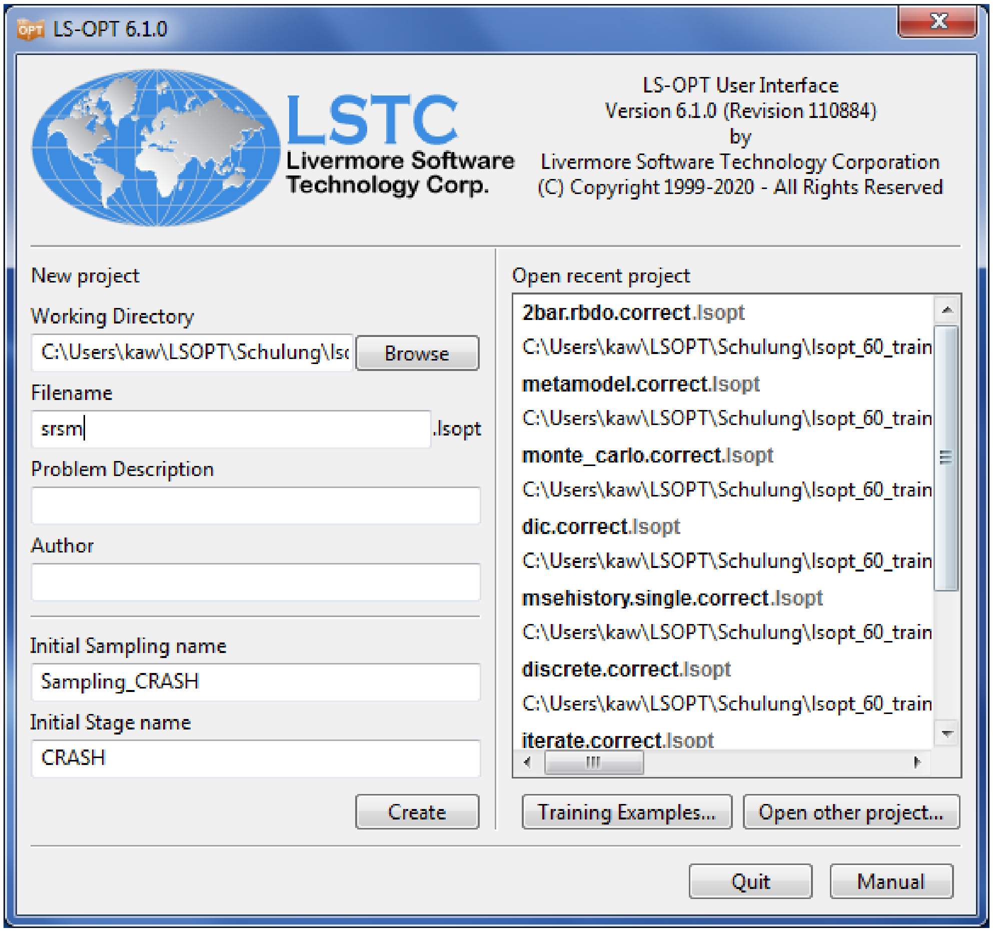
The file `abcde.lsopt` can also be generated by executing the following command in the command prompt:

```
com2xml com.abcde abcde.lsopt
```

Table 3-1: Startup Dialog options

Option	Description	Reference
Working Directory	Directory where the LS-OPT project input files and some of the results are stored.	

Filename	Name of the .xml file that stores the LS-OPT project. The extension `.lsopt` is automatically appended to the selected name.	
Problem Description	A description of the problem can be given. This description is echoed in the `lsopt_input` and `lsopt_output` files, in the plot file titles and in the GUI display (table at bottom right). (optional)	
Author	Author information (optional)	
Initial Sampling name	Each LS-OPT project requires at least one *Sampling* definition. The name of the first sampling has to be specified here. A default name is provided.	Chapter 9
Initial Stage name	Each LS-OPT project requires at least one *Stage* definition. The Stage definition includes the solver type and command as well as the main input file name. The name of the first stage has to be specified here. A default name is provided.	Chapter 5
Create	Creates a new LS-OPT project and opens it in the main GUI	Section 3.2
Open recent project	A project from the list of the last ten LS-OPT projects can be opened.	Section 3.2
Training Examples	Open list of training examples	Section 2.8
Open other project …	Option to open any existing LS-OPT project	Section 3.2
Manual	Open LS-OPT User’s Manual	
Quit	Quit `lsoptui`	



LS-OPT User Interface
Version 6.1.0 (Revision 110884)
by
Livermore Software Technology Corporation
(C) Copyright 1999-2020 - All Rights Reserved

New project

Working Directory: C:\Users\kaw\LSOPT\Schulung\lsc

Filename: srsm .lsopt

Problem Description:

Author:

Initial Sampling name: Sampling_CRASH

Initial Stage name: CRASH

Open recent project

2bar.rbdo.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
metamodel.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
monte_carlo.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
dic.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
msehistory.single.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
discrete.correct.lsopt
C:\Users\kaw\LSOPT\Schulung\lsopt_60_train
iterate.correct.lsopt

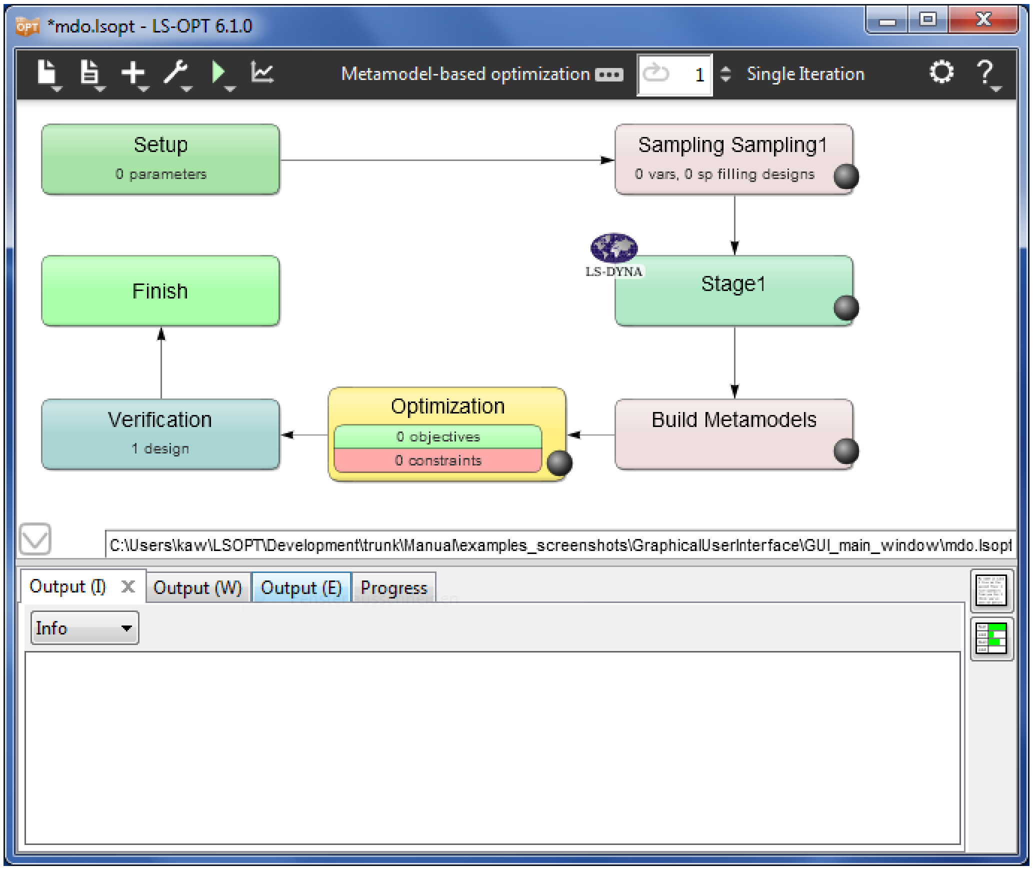
Figure 3-1: Startup Dialog of lsoptui

3.2. The GUI main window

The flowchart in the main GUI of LS-OPT (Figure 3-2) mimics the process of the selected task, e.g. starting from global parameters defined in *Setup*, through the sampling, the simulation process chain defined by the stages and dependencies, the building of metamodel, the metamodel optimization, checking of convergence, and domain reduction in one or more loops, and finally the verification run for a metamodel based, sequential optimization. Refer to Chapter4 for details on the available tasks.

Double clicking on any of the boxes opens the corresponding dialog, where settings can be viewed and adjusted. The dialogs and options are explained in the respective chapters, see Table 3-3.

The control bar menus are described in Table 3-1.



C:\Users\kaw\LSOPT\Development\trunk\Manual\examples_screenshots\GraphicalUserInterface\GUI_main_window\mdo.lsopt

Figure 3-2: Main LS-OPT GUI window for a setup of a Metamodel-based optimization

Table 3-2: Main GUI Control Bar options

Icon	Option	Description	Reference
🗋	New	Opens the Startup Dialog (Figure 3-1) to create a new optimization project.	Section 3.1
	Open	Option to open an existing LS-OPT project	
	Save	Save current project	
	Save as …	Save current project as …	
	Encrypt project	Encrypt the project file	Section 3.7
	Exit	Exit `lsoptui`	
🗎	Input	Open the `lsopt_input` file	
	Output	Open the `lsopt_output` file	

	Summary Report	Open the `lsopt_report` file	
	Warnings	Open the `WARNING_MESSAGE` file	
	Errors	Open the `EXIT_STATUS` file	
	Open project folder	Opens up the working directory	
	Open terminal	Opens up the working directory in a terminal	
	Other file…	Option to open any other text file	
+	Add Sampling	Add additional Sampling. The name of the sampling will be used as the name of a subdirectory used for sampling related databases such as `Experiments_n.csv` and `AnalysisResults_n.lsox`.	Chapter 9
	Add Stage in *Sampling*	Add additional Stage in selected sampling. The name of the stage will be used as the name of a sub-directory to the working directory. Stage-related databases are stored in this directory.	Chapter 5
		New stages may be generated by specifying a name, or previously exported stages can be imported.	Section 3.2.2
	Add Composite	Add Composite	Chapter 10
	Add Domain Reduction	Use Domain Reduction (same as *Sequential with Domain Reduction* option in Task dialog)	Section 4.9
	Add Termination Criteria	Switch to sequential Strategy	Chapter 13
	Add Verification Run	Run an additional simulation using the parameter values of the predicted optimum or Pareto optimal solutions at the end of the optimization run.	Section 4.12
	Add Global Sensitivities	Calculates Global Sensitivities on the meta-model.	Section 4.11
	Add Classifier	Add Classifier	Chapter 11
🔧	Re-layout stages	Layout the stage boxes according to the defined dependencies.	

	Show XML Tree	Show the XML Tree for the current settings.	
	Repair	Global repair or modification of an existing run. A local repair can be done by right-clicking on a Stage or Sampling.	Section 3.5
	Clean	Clean from current iteration [*iter*]: Removes all simulation data as well as optimization data from the specified iteration *iter* onwards. Clean Verification Run: Removes the simulation data as well as optimization data of the verification run. Clean All: The directory structure created by LS-OPT and all the files in this directory structure are deleted. Remote SSH Folders: This menu is only shown if the project contains at least one SSH proxy configuration. The menu is as above with “Clean from current iteration” and “Clean All” but checks and removes files at remote hosts only.	Section 3.4.1
	Archive LS-OPT Database	This option collects relevant files and creates a single tar-zipped (on *nix operating systems) file or zipped (on windows operating systems) file.	Section 3.6
	Save Flowchart image	Saves a PNG image of the LS-OPT main GUI Window	
	DynaStats	Opens DynaStats	Chapter 18
▶	Normal Run	Run task	Section 3.3.1
	Baseline Run	Run a single design, sampled at the initial values.	Section 3.3.2
🟥	Stop	Button is only available while LS-OPT is running. Stops the current optimization and all running jobs.	
📈	Viewer	Opens the viewer for post-processing.	Chapter 16
▪▪▪	Task	Opens Task Dialog.	Chapter 4
⟳ 1 ⇕	Iteration	While running LS-OPT, this visualizes the current running iteration. It is also used to select the current iteration for restarting or repair.	Section 3.4

⚙	Settings	Settings dialog.	Section 3.8
?	Manual	Opens the LS-OPT User’s Manual	
	About	Information about LS-OPT	

Table 3-3: Process Boxes

Box	**Description**	**Reference**
Setup	Parameters (global set), Global optimization settings, variable connectivity, resource data.	Chapter 8
Sampling	Point selection and metamodel settings	Chapter 9
Stage	Interface to solver such as solver command and input file.	Chapter 5
File Transfers	Transfer files to a downstream stage.	Section 3.2.2
Build Metamodels	Same as Sampling	Chapter 9
Composites	Define composites	Chapter 10
Classifiers	Define classifiers	Chapter 11
Global Sensitivities	Calculate global sensitivities	Section 4.11
Optimization	Definition of objectives, constraints and optimization algorithms	Chapter 12
Monte Carlo	Monte Carlo settings	Section 12.5
Termination Criteria	Termination criteria for sequential strategies	Chapter 13
Domain Reduction	Domain reduction settings for strategy sequential with domain reduction	Section 4.9
Verification Run	Perform (specified number of) verification run(s)	Section 4.12

3.2.1. Setting up a Process Flow

A process can be constructed for the purpose of running a sequence of dependent simulations. A typical simple process is a sequence: *pre-processor* → *solver* → *post-processor* which can be constructed by defining three sequential stages. However, a process of high complexity can also be created. For instance the flow of the process is allowed to merge and branch, Figure 3-3.

The process can be constructed in multiple steps by adding stages and connecting the stages using the mouse to create dependencies of one stage on another.

On creating a new optimization project, a first stage is generated. Additional stages can be added using the *Add stage* option of the + menu in the control bar. New stages can be generated or previously exported stages can be imported, Section 3.2.2. A sampling has to be selected to which the new stage is assigned. By default, the new stage is added in parallel to the already existing stages.

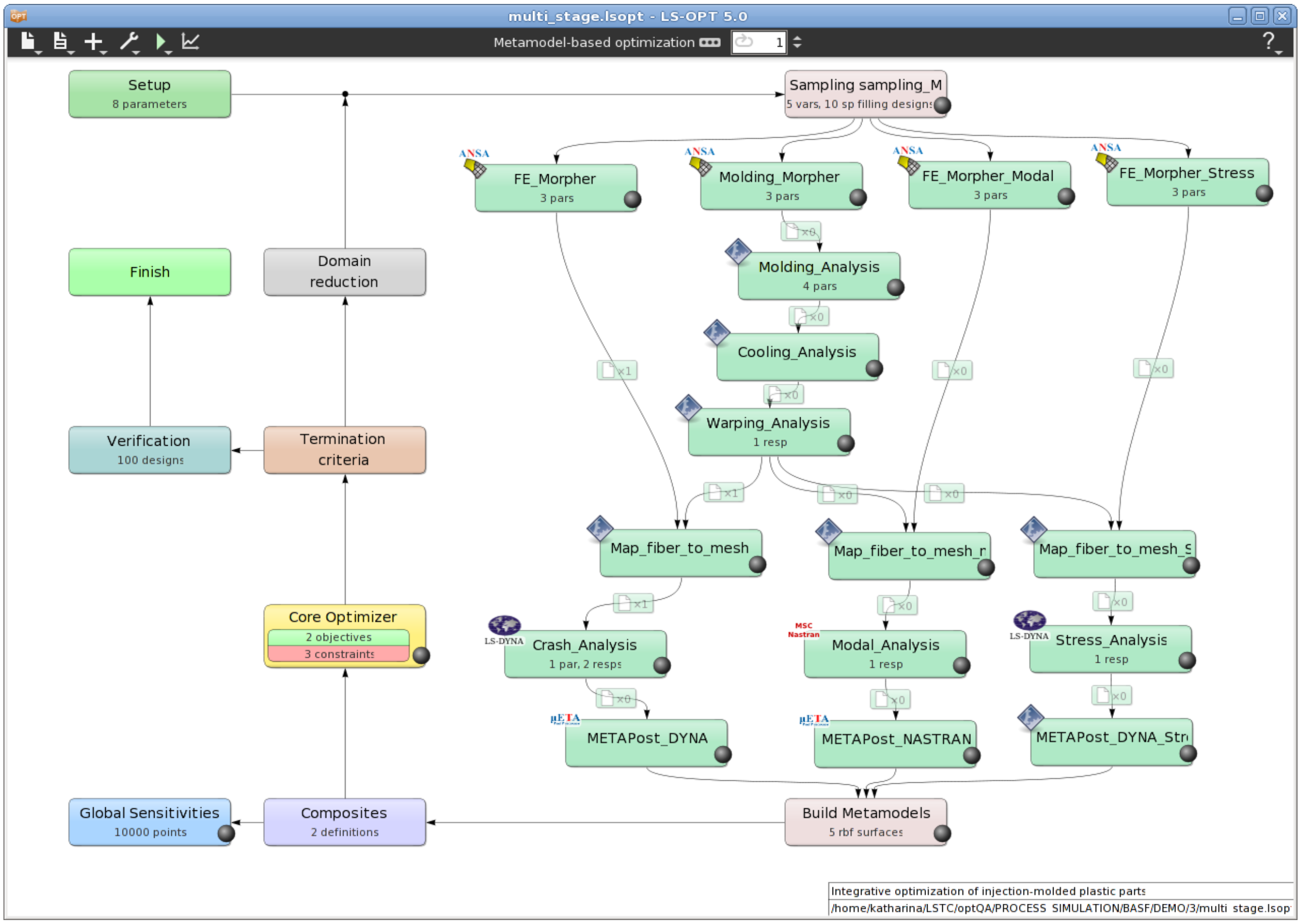


Figure 3-3: Setup of a complex optimization problem

If similar stages are needed for e.g. a multi-case optimization, a stage can be added by using the *Clone* option when right-clicking an already defined stage. This creates a new stage with the same definitions as the original stage. History and response names are updated to ensure uniqueness of names. If the name of the original stage is found in the original names, it is replaced, otherwise the name of the new stage is prepended.

Stages can also be copied using the *Copy* option when right-clicking an already defined stage and then pasted in another project. Uniqueness of names is ensured. The next free number is appended to the stage name if a stage with the same name already exists, and the response and history names are updated accordingly.

The desired dependencies are created as follows, see Figure 3-4:

1. Hover the mouse cursor over the Stage box. A circle appears at the lower edge of the box.

2. Move the mouse cursor to the circle (it should highlight in yellow) and drag the circle to the desired dependent stage box.
3. A connection will be created between the two boxes.

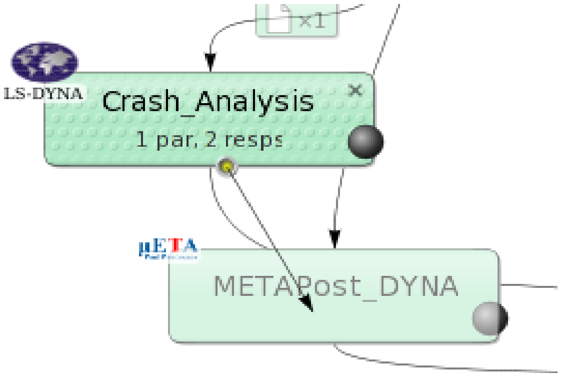


Figure 3-4: Creating stage dependencies

Connections can be deleted using the small icon located on the connection line. This icon also allows the definition of inter-stage file operations, Section 3.2.2.

Stages can be deleted by right-clicking on the stage and then selecting the delete function.

The layout of the stage boxes can be controlled by the user. Left-click and hold down on a stage box to move it freely. For complex process setups, it could be helpful to use the *Re-layout Stages* option from the *Tools* menu in the control bar.

If separate samplings are desired (as is often the case for MDO problems where different variables apply to different loadcases), new samplings can be added at the origin of each process sequence using the *Add sampling* option of the + menu in the control bar. Stages can then be assigned to the relevant samplings.

3.2.2. Export and Import Stages

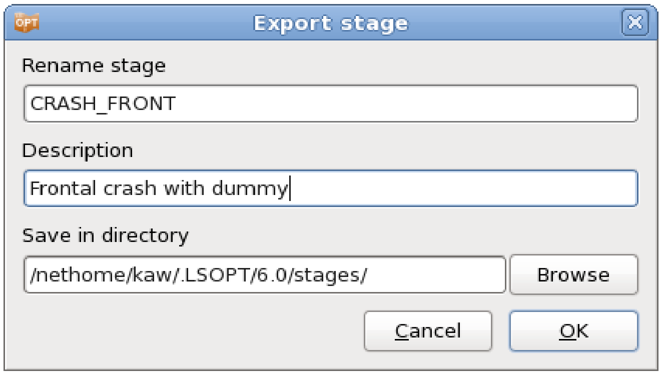
Stage definitions can be exported to files and later be imported into other projects.

To export a stage, right-click on the respective stage box to open the Export stage dialog, Figure 3-5. The stage name is used as filename to store the stage definition. The stage can be renamed in the Export dialog. The extension .lsstage is appended by LS-OPT. The description will be displayed in the Import dialog. The default directory to store the stage information is ~/.LSOPT/*version*/stages on Linux machines, and Application Data\LSOPT\version\stages in the user's home directory on Windows, respectively. The *Save in directory* textfield can be used to specify any other directory.

All stages of a sampling can be exported at once by selecting *Export Stages* from the right mouse menu of the respective Sampling dialog. The sampling name with the extension .lsstage is used as filename to store the stage definitions.

To import a previously exported stage, select *Add Stage in Sampling* from the control bar + menu and select *Import* and the desired directory to display the list of available stages, Figure 3-6. Multiple stages can be imported at a time by using the Ctrl key.

The imported stages are added in parallel, but the user can define dependencies to reorder the stages. If all stages of a sampling are imported, the dependencies are kept. Uniqueness of names is ensured. The next free number is appended to the stage name if a stage with the same name already exists, and the response and history names are updated accordingly.

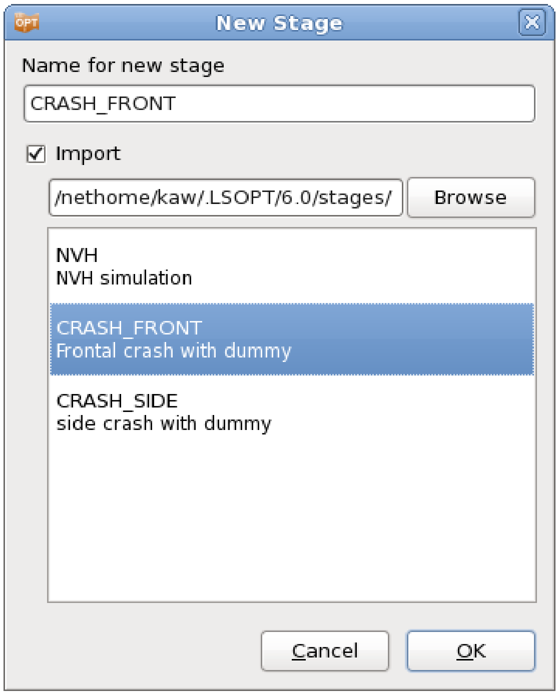


Rename stage: CRASH_FRONT

Description: Frontal crash with dummy

Save in directory: /nethome/kaw/.LSOPT/6.0/stages/

Figure 3-5: Export stage dialog accessible from right-mouse menu of Stage box.



Name for new stage: CRASH_FRONT

☑ Import

/nethome/kaw/.LSOPT/6.0/stages/

NVH
NVH simulation

CRASH_FRONT
Frontal crash with dummy

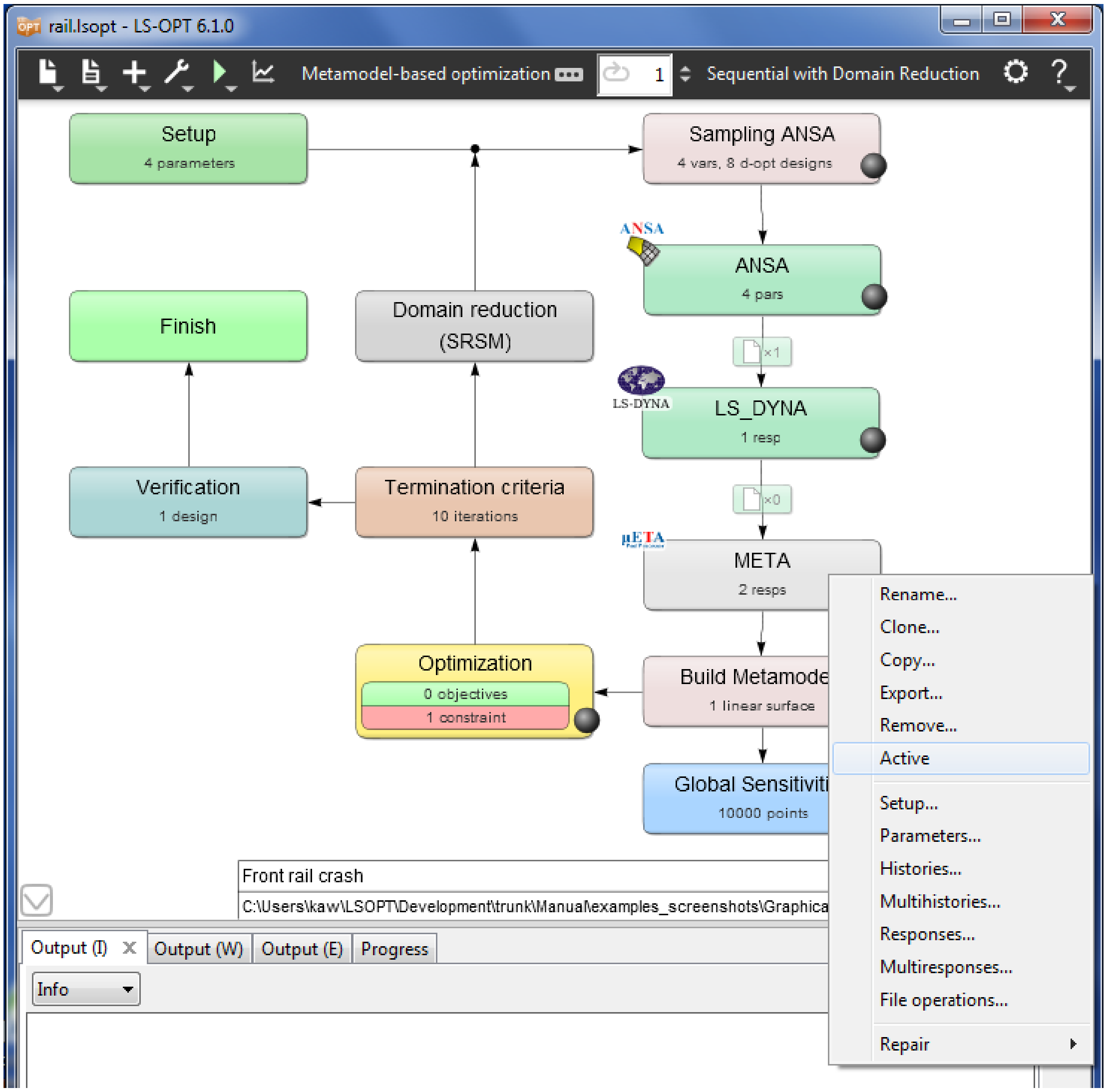
CRASH_SIDE
side crash with dummy

Figure 3-6: Dialog to add a new stage or import a previously exported stage.

3.2.3. Sampling and Stage Deactivation

Stages or samplings defined in the setup can be deactivated or reactivated using the *Active* option from the sampling and stage right mouse menu, Figure 3-7. A gray box indicates a deactivated sampling or stage. All entities such as composites, classifiers, objectives and constraints that depend on a deactivated sampling or stage are automatically deactivated or reactivated. To indicate the deactivation of an entity, the entity name is displayed in gray in the respective dialog, Figure 3-8.

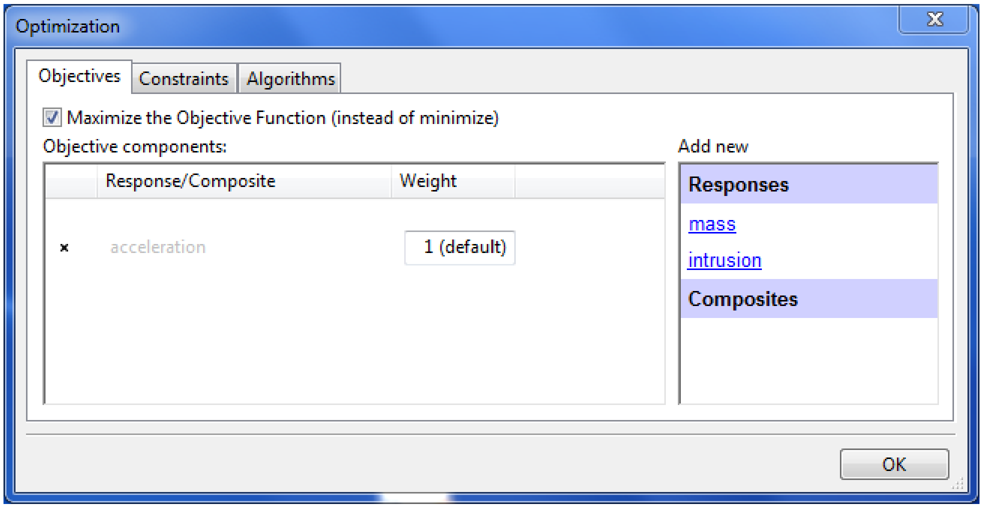
Deactivated samplings or stages will not be executed on running the setup, and deactivated entities will be ignored for the evaluation. If the LS-OPT output database already exists when a stage or sampling is (de)activated, LS-OPT offers options to regenerate the output files according to the new setup on rerunning.



Front rail crash

C:\Users\kaw\LSOPT\Development\trunk\Manual\examples_screenshots\Graphica

Figure 3-7: Setup with deactivated stage META. The Active options from the right mouse menu can be used to activate the stage again.



Optimization

Objectives | Constraints | Algorithms

☑ Maximize the Objective Function (instead of minimize)

Objective components:

	Response/Composite	Weight
×	acceleration	1 (default)

Add new

Responses

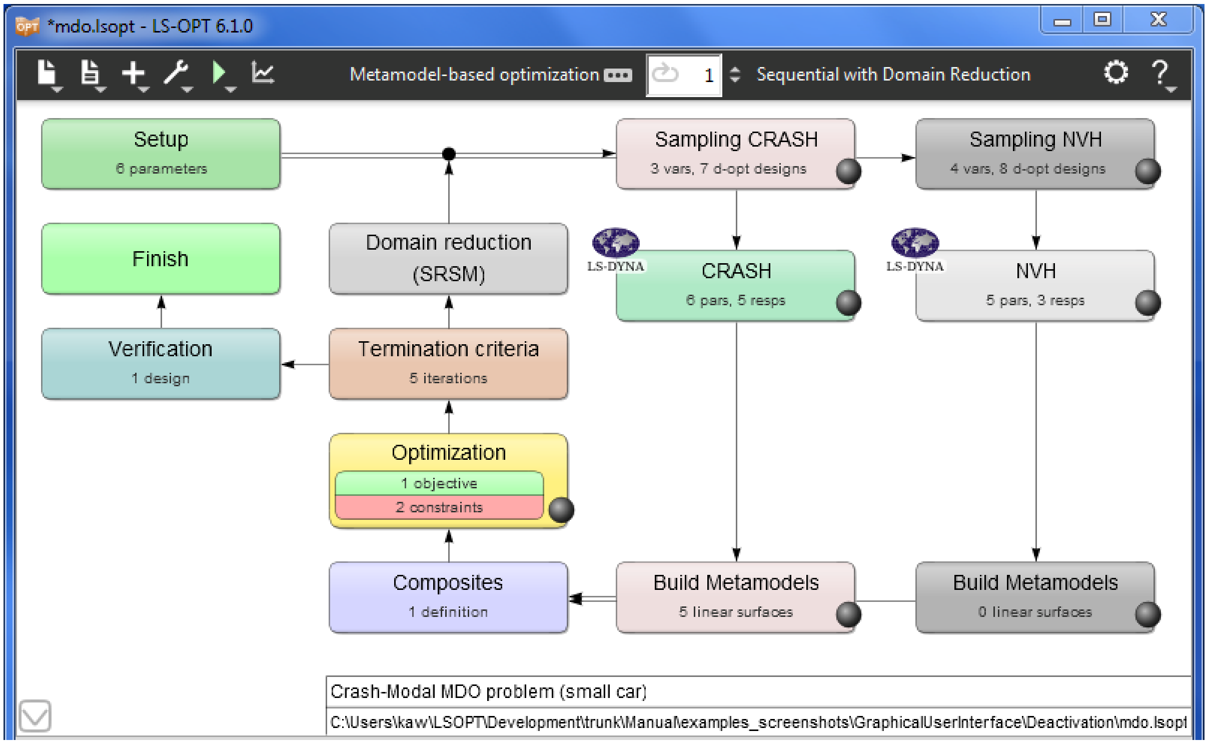
mass

intrusion

Composites

OK

Figure 3-8: Deactivated objective acceleration. The objective is deactivated because it is a response defined in a deactivated stage. Re-activation of the stage will also re-activate the objective.

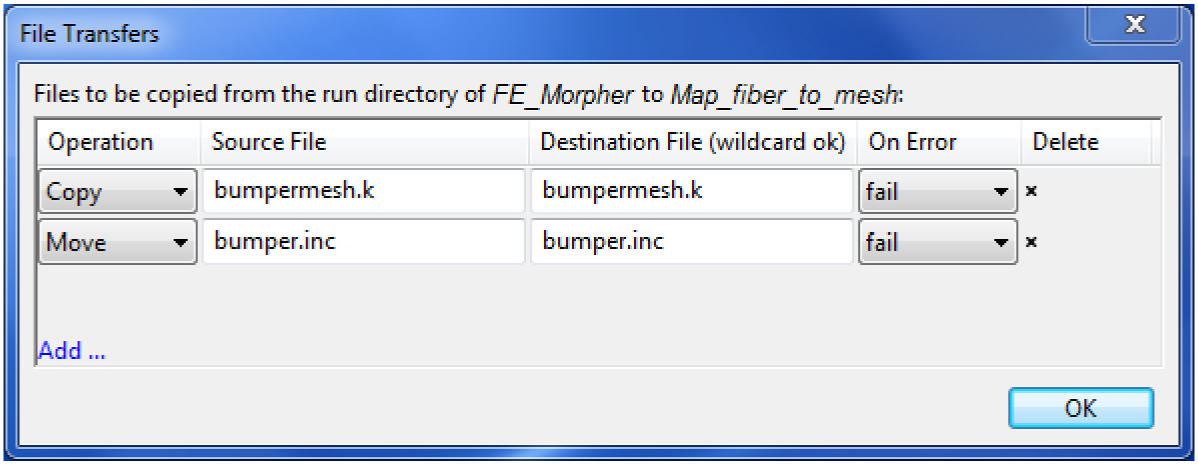


Crash-Modal MDO problem (small car)

C:\Users\kaw\LSOPT\Development\trunk\Manual\examples_screenshots\GraphicalUserInterface\Deactivation\mdo.lsopt

Figure 3-9: Deactivated Sampling NVH. All it’s stages and dependent entities are automatically deactivated as well.

3.2.4. File Transfers between Stages



Files to be copied from the run directory of *FE_Morpher* to *Map_fiber_to_mesh*:

Operation	Source File	Destination File (wildcard ok)	On Error	Delete
Copy	bumpermesh.k	bumpermesh.k	fail	×
Move	bumper.inc	bumper.inc	fail	×

Figure 3-10: File transfers between dependent stages

To use results of upstream stages, LS-OPT allows file transfers between dependent stages. The **File Transfer** dialog is accessible by selecting the dependency icon located on the arrow connecting the stages, see Figure 3-10 and Table 3-4. The requested file transfers are executed for all the run directories related to the Stages, e.g. if the dependency is between `CRASH` and `PRE_CRASH`, file transfer will be executed between `PRE_CRASH/1.1` and `CRASH/1.1`, `PRE_CRASH/1.2` and `CRASH/1.2`, etc.

Table 3-4: File transfer options between stages

Option	**Selections**	**Description**
Operation	Copy Move	Available operations
Source File		Name of source file, wildcards are supported
Destination File		Name of destination file
On Error	fail warn ignore	What to do if operation fails

3.3. Run LS-OPT

3.3.1. Normal Run

This option runs the selected task.

An incomplete run can be restarted using the current state of the optimization and solver databases. Completed simulation jobs are recognized by the presence of the *finished* file in each respective run directory and the termination status of its contents. The presence of the *finished* file allows LS-OPT to avoid a repeat of the simulation for either *error* or *normal* terminations. A clean start option is available, Section 3.4.

If a Normal Run is executed after sampling or stage (de)activation, Section 3.2.3, options are offered to regenerate the database using the current database according to the new activation.

3.3.2. Baseline Run

This feature provides the user with an option to run a single design, often referred to as the baseline design.

The design is sampled at the initial values specified in the **Parameter Setup** panel, Section 8.1. The simulations are executed in the Stage sub-directory 1.1 of the respective stage. This option facilitates a verification of the design, i.e. it allows checking

1. the correct solver command,
2. communication between LS-OPT and the queuing system, if any,
3. presence of all relevant control cards, database formats,
4. data extraction from simulation results, and
5. validity of responses and histories.

It is therefore recommended to use a single simulation using the "*Baseline Run*" option as a "dry" run before launching a full scale optimization run in LS-OPT. A successful baseline run will be recognized as a complete run, so will not have to be repeated in the full optimization run.

3.4. Restarting

3.4.1. Clean from Current Iteration

If the user wants to restart an existing optimization run from a specified iteration, the *Clean - Clean from Current Iteration* [iter] feature available from the control bar 🔧 menu can be used.

The current iteration is specified by the selection of the iteration number (using up/down arrows) in the iteration icon located in the control bar. It is important to note that the clean option removes all simulation data as well as optimization data from the specified iteration onwards.

The task is restarted by selecting *Normal Run* from the run menu.

3.4.2. Augmentation of an existing design

To retain existing (expensive) simulation data in the optimization process, it is advantageous to be able to augment an existing metamodel with additional sampling points and simulations. In this manner, new simulations can be added to old simulations to obtain a more accurate metamodel. This is performed by increasing the number of sampling points in the *Sampling* dialog and restarting e.g. the metamodel-based optimization.

When running the optimization, the experimental design table will be augmented, the additional simulations will be executed, a new metamodel will be constructed and a new predicted optimum will be computed. Note that if a verification run was previously calculated (e.g. Simulation 2.1), the *Clean* option *Clean Verification Run* should be used before restarting in order to replace the verification run in directory 2.1.

3.5. Repair or modification of an existing job

Several types of repairs and modifications are possible for an existing optimization iteration or a probabilistic analysis. The repair depends on the status of the LS-OPT database files as described in Appendix E: Database and Output Files.

Repair tasks can be executed globally or locally on individual Stages or Samplings.

- Global repair can be executed using the Repair option under Tools (available in the control bar).
- Local repair tasks are executed by right clicking on the relevant step (Stage or Sampling) in the main GUI window.

The available repair tasks are:

- *Add points*. Points are added to the existing sampling. This option is only available for the following sampling types: D-Optimal, space-filling, and Latin Hypercube. The D-Optimal and space-filling samplings will augment the previously computed points. The Latin Hypercube experimental design points will be computed using the number of previously computed points as a seed to the random number generator. If the database for the experimental design (`Experiments_`*`n`*`.csv` file for iteration *n*) does not exist, new points will be created.
- *Read points*. The `Experiments_`*`n`*`.csv` file is reconstructed from the data in the `XPoint` database files in the run directories.
- *Import results.* Import results from a .csv (comma separated variables) file (Section 9.5.3).
- *Run Jobs*. The stage jobs will be scheduled. Designs previously analyzed will not be analyzed again. Only available in the Tools menu.
- *Rerun jobs*. The jobs of the respective stage will be resubmitted. Only available in the right mouse menu.
- *Rerun failed jobs*. The jobs that failed to run will be resubmitted. The stage input files used will be regenerated from the files specified for the respective stage. If multiple stages are defined in the process chain, all stages will be rerun.
- *Extract Results*. The results will be extracted from the runs for all stages. This option also allows the user to change the responses for an existing iteration or Monte Carlo analysis.
- *Rerun Verification Run*. The verification run will be resubmitted.
- *Build Metamodels*. The metamodels will be built. This option also allows revision of the metamodels for an existing iteration or Monte Carlo analysis. The "ExtendedResults" file will be updated. Metamodels can for instance be built from imported user results (see section on *Import results* above).
- *Evaluate Metamodels*. Create a table with the error measures of a given set of points (Section 9.5.2) or create a table (.csv file) with response values interpolated from a metamodel (Section 8.5.1).

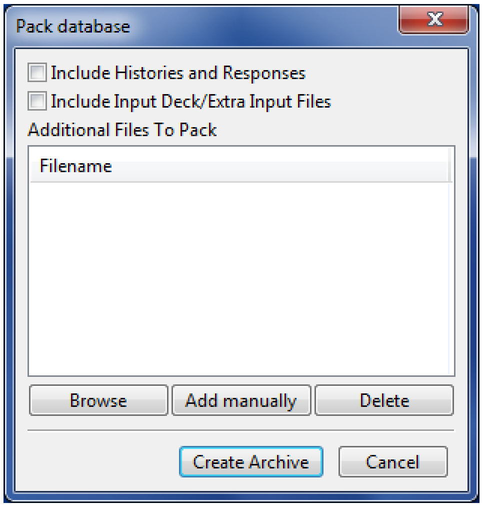
- *Import Metamodels*. Imports metamodels from an .xml file (Section 9.5.4)
- *Calculate Global Sensitivities*. Global Sensitivities are recalculated using the metamodels.
- *Optimize*. The metamodels are used for metamodel optimization. A new optimum results database is created. The "ExtendedResults" file will be updated. The optimization history database is deleted so the history will not be displayed in the Viewer.

Remarks

1. All the subsequent operations must be explicitly performed for the iteration. For example, augmenting an experimental design will not cause the jobs to be run, the results to be extracted, or the metamodels to be recomputed. Each of these tasks must be executed separately.
2. After repair of iteration *n*, and if the user is conducting an optimization task, verification runs of the optimized result must be done by switching back to the Metamodel-based optimization task and specifying the starting iteration (for a clean start) as *n*+1. If *n*+1 was a full iteration (not just a verification run), it also has to be repaired.

3.6. Archive LS-OPT Database

Using the *Archive LS-OPT Database* option in the *Tools* menu, the database can be gathered up and compressed in a file called `lsopack.tar.gz` (`lsopack.zip` on Windows) after completing the run. The packed database is suitable for post-processing on any computer platform.



Pack database

□ Include Histories and Responses
□ Include Input Deck/Extra Input Files

Additional Files To Pack

Filename

Figure 3-11: Dialog to specify options for archiving the LS-OPT database

By default, the files generated by LS-OPT in the working directory and the stage and sampling directories are gathered, the run directories are omitted.

More sophisticated options are available to also gather the history and response files residing in the run directories and all input files. The history/response files (e.g. `history.0`, etc.) are required to view history plots using the DynaStats tool. The inclusion of both histories and input decks results in `lsopack_h_i.tar.gz` (`lsopack_h_i.zip` in Windows).

The history/response files are not required for any of the Viewer functions since this data is available in the `AnalysisResults_`*`n`*`.lsox` file included in the basic archiving selection.

Table 3-5: Archive LS-OPT database options

Option	**Description**
Include Histories and Responses	Also gather the history and response files residing in the run directories. The file produced is `lsopack_h.tar.gz` (`lsopack_h.zip` in Windows). History and response files are only required for the use of DynaStats.
Include Input Deck/Extra Input Files	Various input files and other files required to run the LS-OPT job seamlessly are added to the packed database file. The file produced is `lsopack_i.tar.gz` (`lsopack_i.zip` in Windows).
Additional Files to Pack	List of additional files to pack. Files may be added by browsing or manually.

3.7. Encryption

The lsopt project file can be encrypted with a password by selecting “Encrypt project” from the file menu. Only the project file itself will be encrypted and not any input files defined in stages. The file is encrypted using 256-bit AES encryption.

When selecting “Encrypt project”, a dialog is shown where the encryption password should be entered. When clicking OK, the encryption mode is enabled. This can be confirmed by the checkmark next to the “Encrypt project” menu item. Note that the project itself is not encrypted on disk until you save it. This is to make it possible to save the file under a different filename. The temporary file “lsopt_db” which mirrors the lsopt project file but is used by the engine, will also be encrypted on the next run.

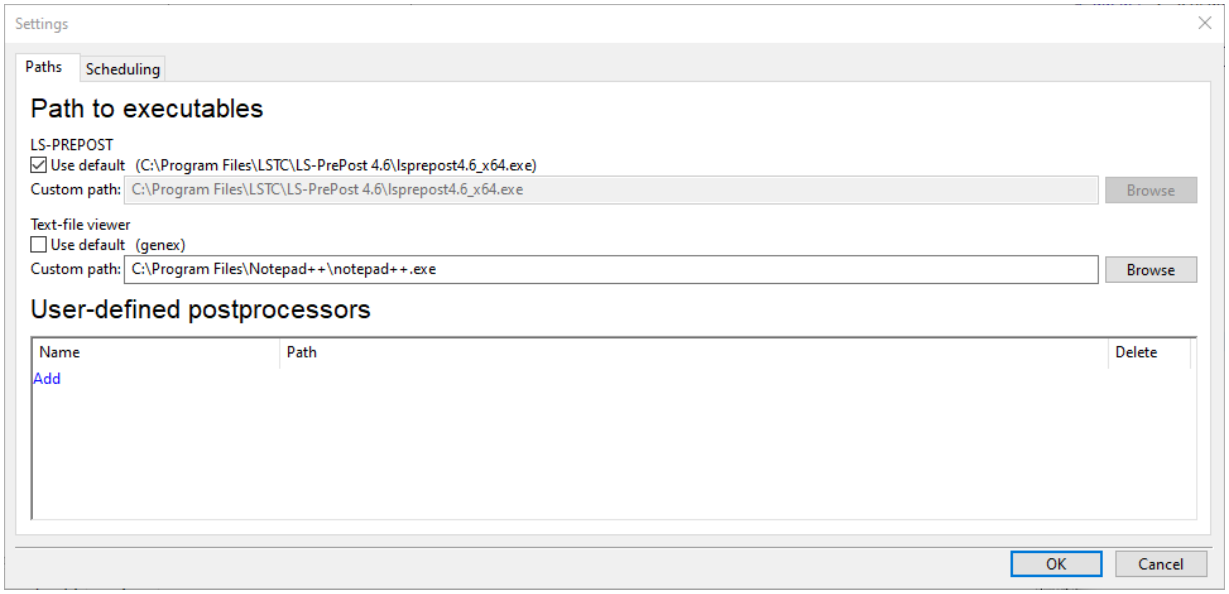
A password needs to be entered when opening an encrypted project file. There is no way to directly run the engine on encrypted projects. This needs to be done through the GUI.

The encryption can be removed by clicking the “Encrypt project” when it’s checked. Remember to save the project after this change.

3.8. Settings

3.8.1. Simulation postprocessor and text viewer

Both the simulation (e.g. Finite Element analysis) postprocessor and the text viewer used throughout the LS-OPT GUI and Viewer can be set, Figure 3-1. The default LS-PrePost executable may be resolved first by the environment variable LSTC_PREPOST, which is set by the LS-PrePost Windows installer, and alternately by an executable *lsprepost* in the LS-OPT installation directory. The default text-file viewer is GenEx which is included in the LS-OPT distribution



Path to executables

LS-PREPOST

☑ Use default (C:\Program Files\LSTC\LS-PrePost 4.6\lsprepost4.6_x64.exe)

Custom path: C:\Program Files\LSTC\LS-PrePost 4.6\lsprepost4.6_x64.exe

Text-file viewer

☐ Use default (genex)

Custom path: C:\Program Files\Notepad++\notepad++.exe

User-defined postprocessors

Name	Path	Delete
Add		

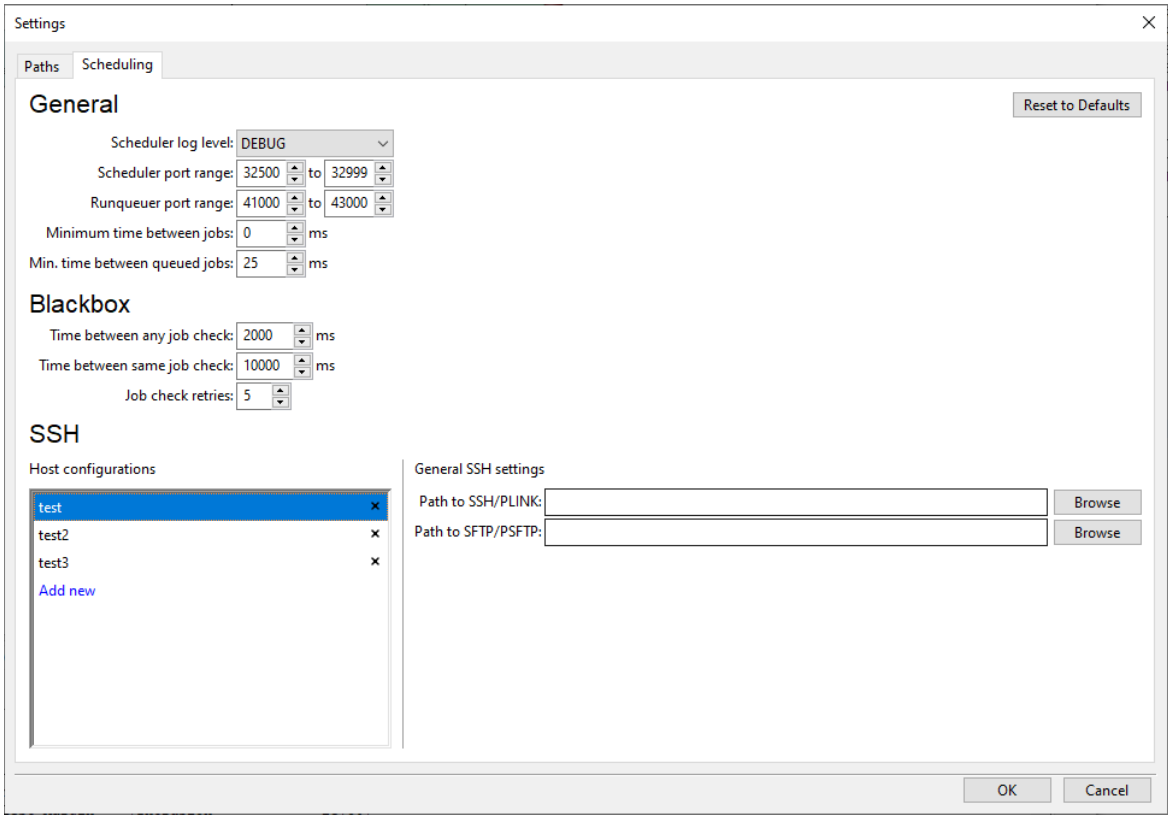
Figure 3-12: Dialog to specify FE postprocessor and text viewer settings

The specified postprocessors are available from the Progress dialog, Section 15.3, and the Viewer Tables, Chapter 17.

3.8.2. Scheduling

Typically, scheduling options are global and not tied to a specific project. Therefore, all options in the Scheduling tab are stored under the current user's data directory ($HOME on Linux or %APPDATA% on Windows) instead of in the current project file. This allows for entering the scheduling options once per host.

In the case of a project which uses SSH proxies is shared between multiple users, it is required that all users have matching SSH configurations, by name. The actual configuration may differ to suit each user.



General

Scheduler log level: DEBUG

Scheduler port range: 32500 to 32999

Runqueuer port range: 41000 to 43000

Minimum time between jobs: 0 ms

Min. time between queued jobs: 25 ms

Blackbox

Time between any job check: 2000 ms

Time between same job check: 10000 ms

Job check retries: 5

SSH

Host configurations

- test
- test2
- test3
- Add new

General SSH settings

Path to SSH/PLINK:

Path to SFTP/PSFTP:

Figure 3-13: Dialog to specify Scheduling settings

Table 3-6: General Scheduling options

Option	Description
Scheduling log level	The level of verbosity at which scheduler, runqueuer, wrapper and lrunremote writes messages to logs. INFO – Default scheduler logging. VERBOSE – More information printed DEBUG – Much more information printed. To be used in debugging purposes. Note that the LSOPT_DEBUG environment flag overrides this option.
Scheduler port range	Scheduler listens on a port in this range used by local subprocesses of scheduler. This may be changed if other services are in use in this range.

Runqueuer port range	Runqueuer listens on remote connections from wrapper in this range. This may be changed to conform with firewall restrictions.
Minimum time between jobs	Minimum time in milliseconds between starting any kind of job.
Min. time between queued jobs	Minimum time in milliseconds between starting jobs using a queuer.

Table 3-7: Blackbox options

Option	**Description**
Time between an job check	Minimum time between checking any blackbox job
Time between same job check	Minimum time between checking the same blackbox job
Job check retries	Maximum number of retries that job check may fail to report a status before setting the job as failed.

Table 3-8: General SSH settings

Option	**Description**
Path to SSH/PLINK	Path to ssh or plink binary. By default, the LS-OPT installation and PATH is checked for the presence of such binary
Path to SFTP/PSFTP	Path to sftp or psftp binary. By default, the LS-OPT installation and PATH is checked for the presence of such binary

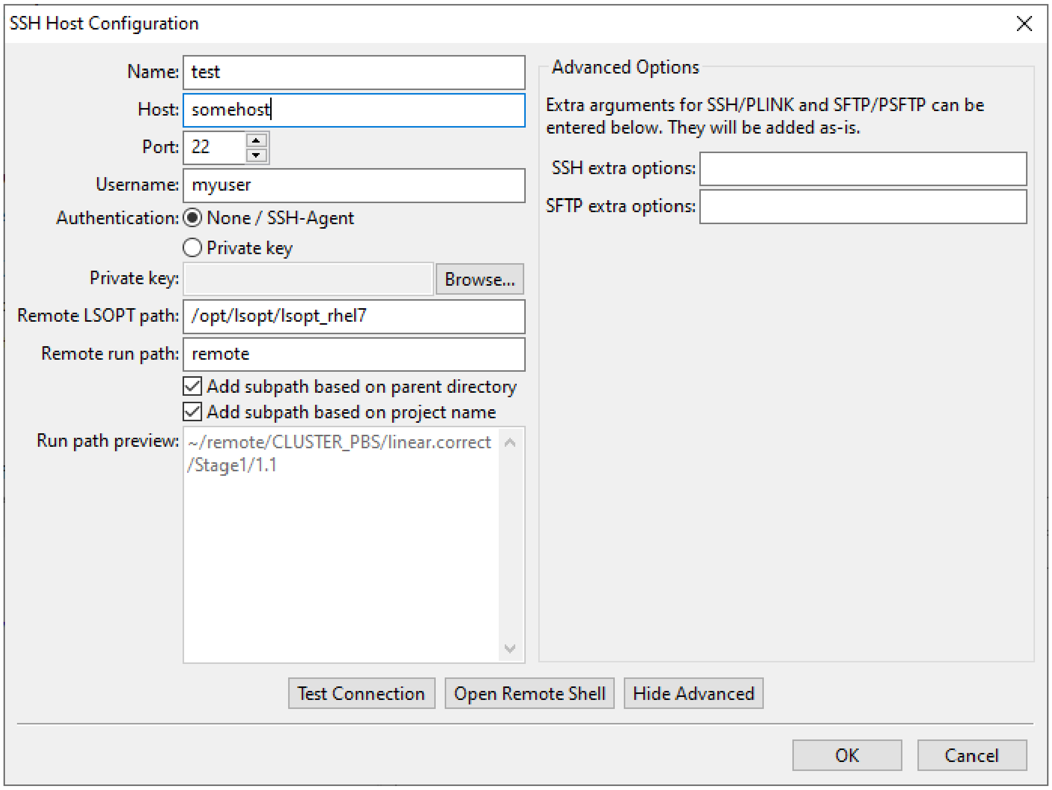
Table 3-9: Host configurations

Option	**Description**
Name	Unique name of configuration. A Stage using an SSH proxy refer to this entry.
Host	Hostname or IP of remote host
Port	SSH port to connect to (default: 22)
Username	Username to use (default: local username)

Authentication	None/SSH-Agent – Highly recommended option! Authentication is setup to require no further user action. E.g. host-based, default private key or SSH/Plink will connect using a key loaded in memory using an agent facility. Private key – A non-password protected private key.
Private key	Path to private key file
Remote LSOPT path	The path to a LSOPT installation on the remote host
Remote run path	The base run path on the remote host. If empty or a relative path it will be relative to the remote home directory (~).
Add subpath based on parent directory	Append the current projects directory last directory name on the remote run path.
Add subpath based on project name	Append the current projects name on the remote run path
Run path preview	A preview of the full remote run path, with example Stage “Stage1”, iteration 1 and experiment 1.
Test Connection	Run a remote test which will do basic checks and report the status.
Open Remote Shell	Spawn a shell of SSH or PLINK using the host configuration and the remote run path. Note that PLINK may not convert special terminal characters such as coloring information which will result in odd characters being showed.

Table 3-10: Advanced SSH Host Connection options

Option	**Description**
SSH extra options	Extra arguments for the SSH or PLINK binary
SFTP extra options	Extra arguments for the SFTP or PSFTP binary



SSH Host Configuration

Name: test

Host: somehost

Port: 22

Username: myuser

Authentication: ● None / SSH-Agent

○ Private key

Private key:

Remote LSOPT path: /opt/lsopt/lsopt_rhel7

Remote run path: remote

☑ Add subpath based on parent directory

☑ Add subpath based on project name

Run path preview: ~/remote/CLUSTER_PBS/linear.correct/Stage1/1.1

Advanced Options

Extra arguments for SSH/PLINK and SFTP/PSFTP can be entered below. They will be added as-is.

SSH extra options:

SFTP extra options:

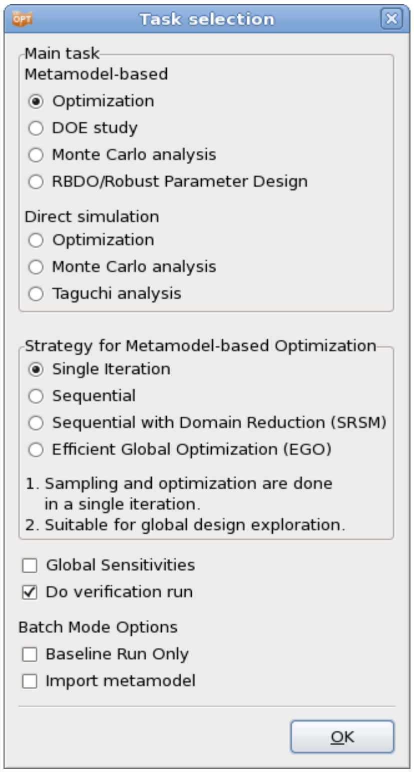
Figure 3-14: SSH Host Configuration

4. Task Dialog – Selecting a Task and Strategy

This chapter explains the available design tasks and strategies.

4.1. Task selection

The Task dialog allows the selection of a task. For a metamodel-based optimization or robustness task, the strategy can be selected. The two basic branches are Metamodel-based and Direct methods (Figure 4-1). The method selections can be made in the GUI using the *Show task settings* icon (▪▪▪) in the control bar in the top menu bar of the main GUI window. The available tasks and options are listed in Table 4-1.



Task selection

Main task

Metamodel-based

● Optimization
○ DOE study
○ Monte Carlo analysis
○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization
○ Monte Carlo analysis
○ Taguchi analysis

Strategy for Metamodel-based Optimization

● Single Iteration
○ Sequential
○ Sequential with Domain Reduction (SRSM)
○ Efficient Global Optimization (EGO)

1. Sampling and optimization are done in a single iteration.
2. Suitable for global design exploration.

☐ Global Sensitivities
☑ Do verification run

Batch Mode Options

☐ Baseline Run Only
☐ Import metamodel

Figure 4-1: Task and Strategy selection

Table 4-1: Task selection options

Option		Description	Reference
Metamodel based	Optimization	Optimization using meta-models	Section 4.2
	DOE-study	DOE study using meta-models	Section 4.3
	Monte Carlo analysis	Monte Carlo analysis using meta-models	Section 4.5.2
	RBDO	Reliability based design optimization using meta-models	Section 4.6
Direct simulation	Optimization	Direct optimization using the Genetic Algorithm	Section 4.4
	Monte Carlo analysis	Direct Monte Carlo analysis	Section 4.5.1
	Taguchi analysis	Direct optimization/parametric study using Orthogonal Array sampling	Section 4.7
Strategy for Metamodel based optimization (Available for Main Task Optimization, Monte Carlo Analysis and RBDO)	Single Iteration	Sampling and optimization are done in a single iteration. Suitable for global design exploration.	Section 4.8.1
	Sequential	Sampling points are added sequentially in the full design space. Suitable for global design exploration or Monte Carlo Analysis.	Section 4.8.2
	Sequential with Domain Reduction	Sampling points are added sequentially in an adaptive subregion. Metamodels are then constructed using the current iteration samples (in the subregion) or using all the samples. The optimum solution is located based on the metamodels. Suitable for finding a converged solution. Generally unsuitable for global exploration.	Section 4.8.3
	Efficient Global Optimization	Sampling points are added sequentially in the full design space. The optimum solution is located based on a Kriging metamodel and balances local and global search through compromise	Section 4.8.4

		between good objective function value and high prediction uncertainty. Suitable for global exploration.	
Available for metamodel-based tasks.	Global Sensitivities	Option to calculate Global Sensitivities on the metamodel.	Section 4.11
Available for Main Task Optimization and RBDO	Do verification run	Run an additional simulation using the parameter values of the predicted optimum. Multiple simulations can be run for Multi-Objective optimization problems.	Section 4.12
Available for global strategies with multiple objectives.	Create Pareto Optimal Front	Option, for Multi-Objective Optimization problems, to create Pareto optimal solutions instead of a single optimum.	Section 4.10
	Baseline Run Only	Batch Mode option to run only the Baseline Run	Section 3.3.2
	Import metamodel	Automated Import metamodel on run instead of manually importation,	Section 9.5.4

4.2. Metamodel based optimization

Metamodel-based optimization is used to create and optimize an approximate model of the design instead of optimizing the design through direct simulation. The metamodel is thus created as a simple and inexpensive surrogate of the actual design. Once the metamodel is created it can be used to find the optimum or, in the case of multiple objectives, the Pareto Optimal Front. The basic steps are as follows:

1. Point selection
2. Run the simulations
3. Build the metamodels
4. Execute the metamodel optimization

4.3. DOE study

A DOE study is also a metamodel-based method used to explore the design space or to calculate sensitivities. The DOE study has three steps:

1. Point selection
2. Run the simulations

3. Build the metamodels

Global sensitivities are evaluated on the metamodels if the option is selected, Section 4.11.

4.4. Direct optimization

Direct optimization uses only simulation results to find the optimal values using a Genetic Algorithm.

Note that the choice of the Direct Optimization (Direct Genetic Algorithm) may require a large number of simulations.

4.5. Probabilistic Analysis Tasks

This category of probabilistic tasks deals with the study of the effect of design parameter uncertainties on the responses. The goal is to obtain the statistics of response variations caused due to the uncertainties in a given design as well as the probability of failure for that design. Any probabilistic task requires the definition of random variables associated with distributions, Section 8.1.5. The point selection scheme for a probabilistic analysis depends on whether it is direct or metamodel-based, Section 14.5 and Section 14.6. More specific details about the available probabilistic analysis tasks are provided in Section 14.5 and Section 14.6. Two probabilistic analysis tasks are currently available in LS-OPT - Direct Monte Carlo Analysis and Metamodel-based Monte Carlo Analysis.

4.5.1. Direct Monte Carlo Analysis

Sampling is based on the distribution of random variables (Section 14.4). No metamodel is constructed to perform this task.

4.5.2. Metamodel-based Monte Carlo Analysis

Sampling is not based on the distribution of random variables (Section 14.4). Statistics are calculated based on metamodel approximations.

4.6. RBDO/Robust Parameter Design (Probabilistic Optimization Task)

This task allows one to perform an optimization under the effect of uncertainties. Considering the effect of uncertainties can be important to avoid unforeseen failure of the design due to variations of loading conditions, manufacturing process etc. In reliability-based design optimization (RBDO), a target probability of failure (typically small) is defined for the constraints to ensure that the optimal design cannot have a higher failure probability. In robust design, an optimal design is searched such that it is insensitive to uncertainties in certain design parameters. More specific details about the available probabilistic analysis tasks are provided in Section 14.7. The difference with deterministic optimization lies in the definition variables that are associated with probabilistic distributions, as well as in the definition of objectives (robust design) and constraints (RBDO).

4.7. Taguchi

The Taguchi method is a direct single-iteration optimization and parametric study technique. It involves a careful selection of process parameters using Orthogonal Arrays to analyze the variable effect on the results and, if required, obtain the optimum results based on a Signal-to-Noise (S/N) ratio.

4.8. Selecting strategies for metamodel-based optimization and Monte Carlo Analysis

In this section different strategies for building a metamodel are discussed. The strategies depend mostly on whether the user wants to build a metamodel that can be used for global exploration or whether he is only interested in finding an optimal set of parameters. An important criterion for choosing a strategy is also whether the user wants to build the metamodel and solve the problem iteratively or whether he has a "simulation budget" i.e. a certain number of simulations and just wants to use the budget as effectively as possible to build a metamodel for improving the design or evaluating statistics and obtaining as much information about the design as possible.

There are four available strategies for automating the metamodel-based optimization or robustness analysis procedure. These strategies only apply to the tasks *Metamodel-based Optimization*, *Metamodel-based Monte Carlo Analysis* and *RBDO/Robust Parameter Design*, Table 4-1. In the GUI, the strategies are selected in the "*Task selection*" dialog (Figure 4-1). The available optimization strategies are

1. Single iteration
2. Sequential
3. Sequential with Domain Reduction (SRSM)
4. Efficient Global Optimization (EGO)

The available Monte Carlo strategies are

1. Single iteration
2. Sequential

A strategy selection resets the *Sampling Dialog* (a warning is given!) with recommended selections for Metamodel type and Point selection scheme, Chapter 9 .

The strategies are discussed one by one in the following sections.

4.8.1. Single iteration

In this approach, the experimental design for choosing the sampling points is done only once. The metamodel selection defaults to Radial Basis Function Networks with Space Filling as the sampling scheme.

4.8.2. Sequential strategy

In this approach, sampling is done sequentially. A small number of points is typically chosen for each iteration and multiple iterations can be requested in the *Termination Criteria* dialog, Chapter 13. The approach has the advantage that the iterative process can be stopped as soon as the metamodels, optimum points or statistical entities have achieved sufficient accuracy.

The default settings for sampling follow below (see Chapter 9):

1. Radial Basis Function networks
2. Space Filling sampling.
3. The first iteration is Linear D-Optimal.
4. Choose the number of points per iteration to not be less than the default for a linear approximation $(1.5(n+1)+1)$ where n is the number of variables.

It was demonstrated in Reference [16] that, for Space Filling, the Sequential approach had similar accuracy compared to the Single Stage approach, i.e. 10 × 30 points added sequentially is almost as good as 300 points. Therefore both the Single Stage and Sequential methods are good for design exploration or Monte Carlo Analysis using a metamodel. Both these strategies are only suitable with metamodels other than polynomials because of the flexibility of metamodels such as RBF's and FFNN's to adjust to an arbitrary number of points.

4.8.3. Sequential strategy with domain reduction

This approach is the same as that in section 4.8.2, but in order to accelerate convergence, an adaptive domain reduction strategy is used to reduce the size of the subregion, Section 26.6. During a particular iteration, the new points are located within a subregion of the design space. This strategy is typically only used for optimization in which the user is only interested in the final optimal point and not in any global exploration of the design. For example, the method is often used in parameter identification, Section 27.3. This method cannot currently be used to construct a Pareto Optimal Front.

The default domain reduction approach is sequential response surface method (SRSM), which is the original LS-OPT design automation strategy. By default, a linear response surface is used and points belonging to previous iterations are ignored.

The default settings for sampling are listed below (see *Sampling* dialog, Chapter 9):

1. Linear polynomial
2. D-optimal sampling
3. Default number of sampling points based on the number of design variables.

4.8.4. Efficient Global Optimization

This approach is the same as that in section 4.8.2, but differs in the selection of the first sample of the successive iterations. Instead of minimizing the metamodel approximation of the objective function, this sample is located by maximizing an expected improvement (EI) function, Section 26.7. The primary idea is that the metamodel approximation can be locally inaccurate due to sparse sampling, and therefore, minimizing the current objective function approximation may not be the best or most efficient option. Instead, the EI function strikes a balance between objective function minimization and sampling of sparse regions. This method is currently limited to the Kriging metamodel and cannot be used to construct a Pareto Optimal Front.

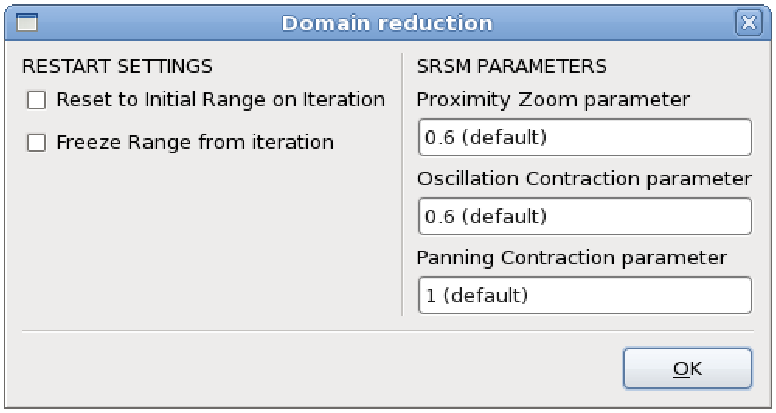
The default settings for sampling are listed below (see *Sampling and Metamodeling* dialog, Chapter 9):

1. Kriging metamodel

2. Linear trend
3. Space-filling sampling
4. Default number of sampling points based on the number of design variables.

4.9. Domain reduction in metamodel-based optimization

The Domain reduction dialog is displayed in Figure 4-2. Table 4-2 describes the options.



Domain reduction

RESTART SETTINGS

□ Reset to Initial Range on Iteration

□ Freeze Range from iteration

SRSM PARAMETERS

Proximity Zoom parameter

0.6 (default)

Oscillation Contraction parameter

0.6 (default)

Panning Contraction parameter

1 (default)

OK

Figure 4-2: Domain reduction dialog

Table 4-2: Restart Settings and Subdomain parameters

Option	**Description**	**Reference**
Reset to Initial Range on Iteration	Resetting the subdomain range to the initial range for a specified iteration.	Section 4.9.1
Freeze Range from Iteration	Freeze the subdomain range from a specified iteration	Section 4.9.1
Panning Contraction parameter	γ_{pan}	Section 4.9.2
Oscillation Contraction parameter	γ_{osc}	Section 4.9.2
Proximity Zoom parameter	Zoom parameter η	Section 4.9.2

4.9.1. Changing the behavior of the subdomain

Resetting the subdomain range

It is possible to reset the subregion range to the initial range, e.g. for adding points in the full design space (or any specified range around the optimum) after an optimization has been conducted. This feature is typically only used in a restart mode. The GUI option is "Reset to Initial Range on Iteration" (Figure 4-2).

The point selection of the specified iteration will be conducted in the initial range around the most recent optimum point. The subdomain will be adaptively updated again starting with the next iteration.

Freezing the subdomain range

This feature allows for points to be added without changing the size of the subregion. Adaptivity can be frozen at a specified iteration number. The GUI option is "Freeze Range from iteration" (Figure 4-2).

The subdomain range will be adaptively updated up to the previous iteration. Therefore the specified iteration and higher will have the same range (although the region of interest may be panning). The flag is useful for adding points to the full design space without any changes in the boundaries.

4.9.2. Setting the subdomain parameters*

To automate the successive subdomain scheme for SRSM, the size of the region of interest (as defined by the range of each variable) is adapted based on the accuracy of the previous optimum and also on the occurrence of oscillation (see theory in Section 26.6).

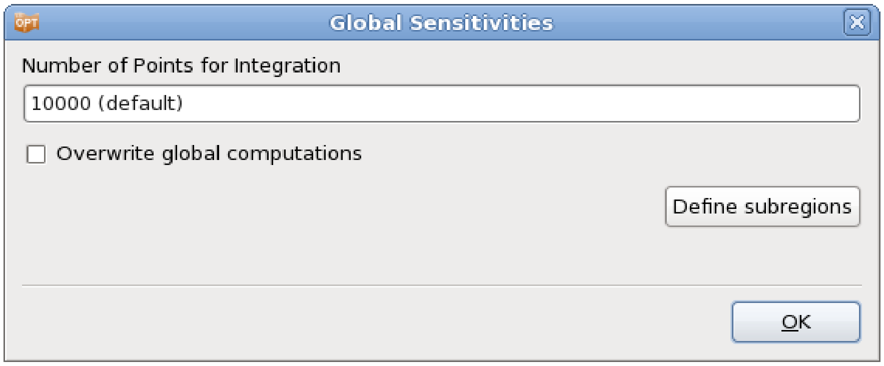
The following parameters can be adjusted in the GUI, Figure 4-2. The options are described in Table 4-2 (refer also to Section 26.6). A suitable default has been provided for each parameter and the user should not find it necessary to change any of these parameters.

4.10. Create Pareto Optimal Front

This option is only available if multiple objectives are defined. If *Create Pareto Optimal Front* is selected, multiple Pareto optimal solutions are calculated instead of a single optimum, see Section 26.10.2. If a metamodel-based method is used, available strategy options are limited to the *global* strategies *Single Stage* and *Sequential*, Section 4.8.2 and 4.8.3. Selection of the *Create Pareto Optimal Front* option resets the optimization algorithm used on the metamodel to Genetic Algorithm, Section 12.4.2, because this is the only algorithm that has the capability to calculate Pareto optimal solutions.

4.11. Global sensitivity analysis

While the ANOVA (Analysis of Variance, Section 23.4) is a very popular method to assess the contribution of different regression terms, Global Sensitivity Analysis (Sobol’s method, based on ANOVA) is widely used to study the importance of different variables for higher order models. In this method, a function is decomposed into sub-functions of different variables such that the mean of each sub-function is zero and each variable combination appears only once. Then, the variance of each sub-function represents the variance of the function with respect to that variable combination. The theory of Sobol’s global sensitivity analysis (GSA) method is described in Section 29.7.2. The GSA is carried out by selecting the appropriate flag (*Global Sensitivities*) in the Task dialog or by selecting Add Global Sensitivities from the *Add* (+) menu in the GUI. The GSA dialog is shown in Figure 4-3. The number of Monte-Carlo integration points used to compute sensitivities is 10000 by default, but this number can be changed by the user. Except for the linear case, the sensitivities depend on the region of design space under consideration. By default, the sensitivities are calculated for the region defined by the variable bounds specified in the *Global Setup*. These sensitivity indices are stored in the *Sobol_GSA.iteration* XML database files in the *work* directory. Existing GSA results can be repaired by checking on the *'Overwrite global computations'* box. This may be needed, for example, if the metamodel is changed after carrying out an earlier sensitivity analysis; the old *Sobol_GSA.iteration* files are then deleted and recreated based on the new analysis.



Number of Points for Integration: 10000 (default)

□ Overwrite global computations

Figure 4-3: Global Sensitivities Dialog

Table 4-3: Global Sensitivities options

Option	**Description**	**Reference**
Number of Points for Integration	Number of Monte-Carlo integration points required to compute sensitivities	
Overwrite global computations	GSA results overwritten for the global region defined by the variable bounds in *Setup* dialog.	
Define subregions	Define a subregion of the design space for GSA. It is possible to have the same bounds as the entire design space (e.g. same domain analyzed with different metamodels).	Section 4.11.1

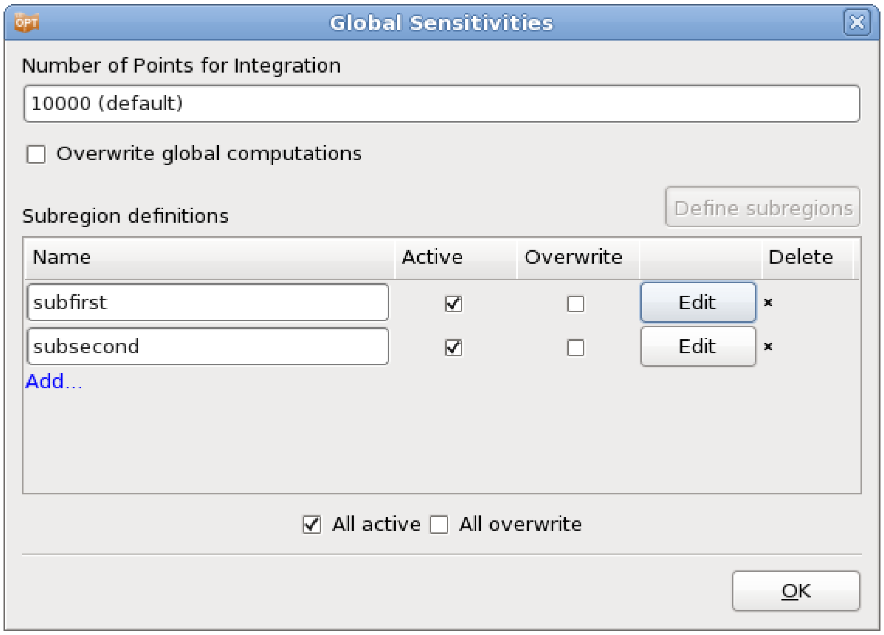
Remarks

1. In LS-OPT, global sensitivities are evaluated on the metamodels. Therefore, the accuracy depends on the quality of the metamodel.
2. Unless a subregion is considered (Section 4.11.1), the sensitivities are calculated for the global bounds of the variables. Sampling constraints are not considered while calculating the sensitivities.

3. Analytical equations are used to compute sensitivities for polynomials and Gaussian radial basis function metamodels.
4. The composite expressions and subregion sensitivities are always evaluated using the Monte-Carlo integration.
5. The default number of sampling points for Monte-Carlo integration is 10000. This number should be increased for better accuracy of sensitivity coefficients.

4.11.1. Sensitivity Analysis in Subregions

The Global Sensitivities dialog also provides the option to define subregions for GSA (Figure 4-4). The sensitivities can be calculated for different variable ranges using this feature, which can be different from the bounds specified in the *Global Setup*. By default, the subregions are created with the same ranges as the global design space. The subregion variable ranges can, however, be modified by clicking on *Edit*, which opens up another dialog. The dialog for variable bound definition or the *Subregion Dialog* is shown in Figure 4-5. The definition of a GSA subregion requires a name to be associated with it. The corresponding GSA results are stored in *Sobol_GSA.RegionName.iteration* files in the work directory.



Number of Points for Integration

10000 (default)

☐ Overwrite global computations

Subregion definitions

Name	Active	Overwrite		Delete
subfirst	☑	☐	Edit	×
subsecond	☑	☐	Edit	×

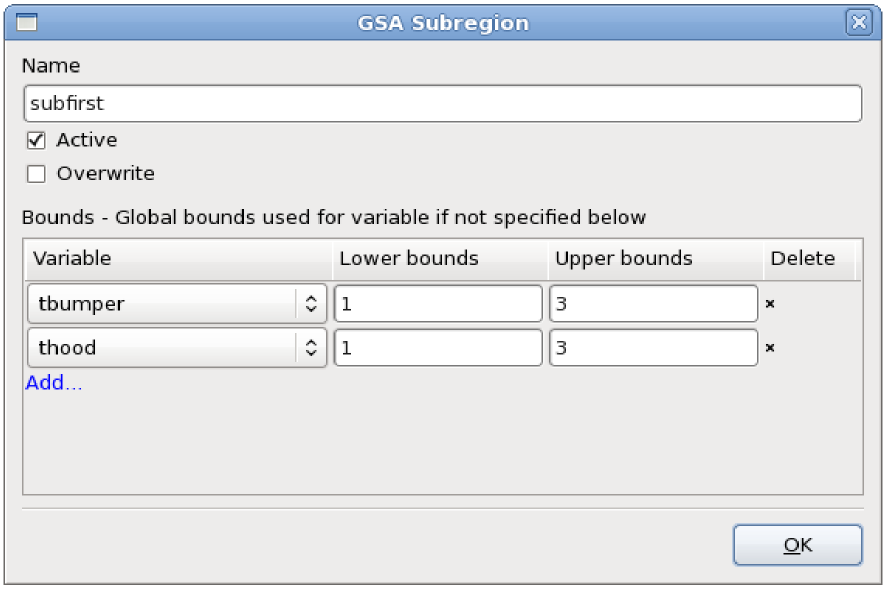
☑ All active ☐ All overwrite

Figure 4-4: Global Sensitivities dialog with subregion definitions

Table 4-4: GSA Subregion definition options

Option	Description
Name	Subregion name
Active	GSA is performed for the Subregion (default on)

Overwrite	Existing GSA results deleted (default off). GSA performed again if *Active* is on
Edit	Open GSA Subregion dialog (Figure 4-5) to define variable bounds of subregion
All active	All subregions active
All overwrite	Overwrite existing GSA results for all subregions


GSA Subregion

Name

subfirst

☑ Active

□ Overwrite

Bounds - Global bounds used for variable if not specified below

Variable	Lower bounds	Upper bounds	Delete
tbumper	1	3	×
thood	1	3	×

Figure 4-5: GSA Subregion dialog

Table 4-5: GSA Subregion dialog options

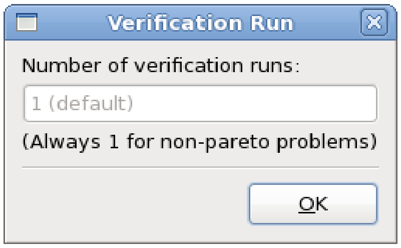
Option	**Description**
Name	Subregion name
Active	GSA is performed for the subregion (default on)
Overwrite	Existing GSA results deleted (default off). GSA performed again if *Active* is on
Bounds	Define subregion lower and upper bounds for variables. The global region bounds defined in *Setup* dialog are used for other variables

4.12. Verification runs

After the last full iteration, a verification run of the predicted optimal design is executed. This run can also be omitted if the user is only interested in the prediction of the optimum using the metamodel.

The verification run options can be edited in the GUI either in the Task dialog or using the “Add …” menu option in the control bar.

For multi-objective optimization problems, multiple verification runs can be done. A discrete Space Filling algorithm is used to select Pareto Optimal points which are evenly distributed in the design space. The number of verification runs can be set in the GUI using the Verification Run box, Figure 4-6.



Number of verification runs:

1 (default)

(Always 1 for non-pareto problems)

__Figure 4-6: Verification Run dialog__

5. Stage Dialog − Defining the Solver

This chapter describes how to interface LS-OPT with simulation packages, parametric preprocessors or postprocessors. Standard interfaces as well as interfaces for user-defined executables are discussed.

The main entity discussed here is the *Stage* dialog which allows the user to define a step in the simulation process.

5.1. Introduction

Since an executable program is considered to be a key part of the stage definition it is often simply referred to as the ***solver***. Therefore, in addition to its normal meaning as a program to, for instance, solve a physics problem, it can also refer to a pre- or postprocessor or any other executable program or script that is essential to the execution or management of a step within a simulation process.

5.2. General Setup

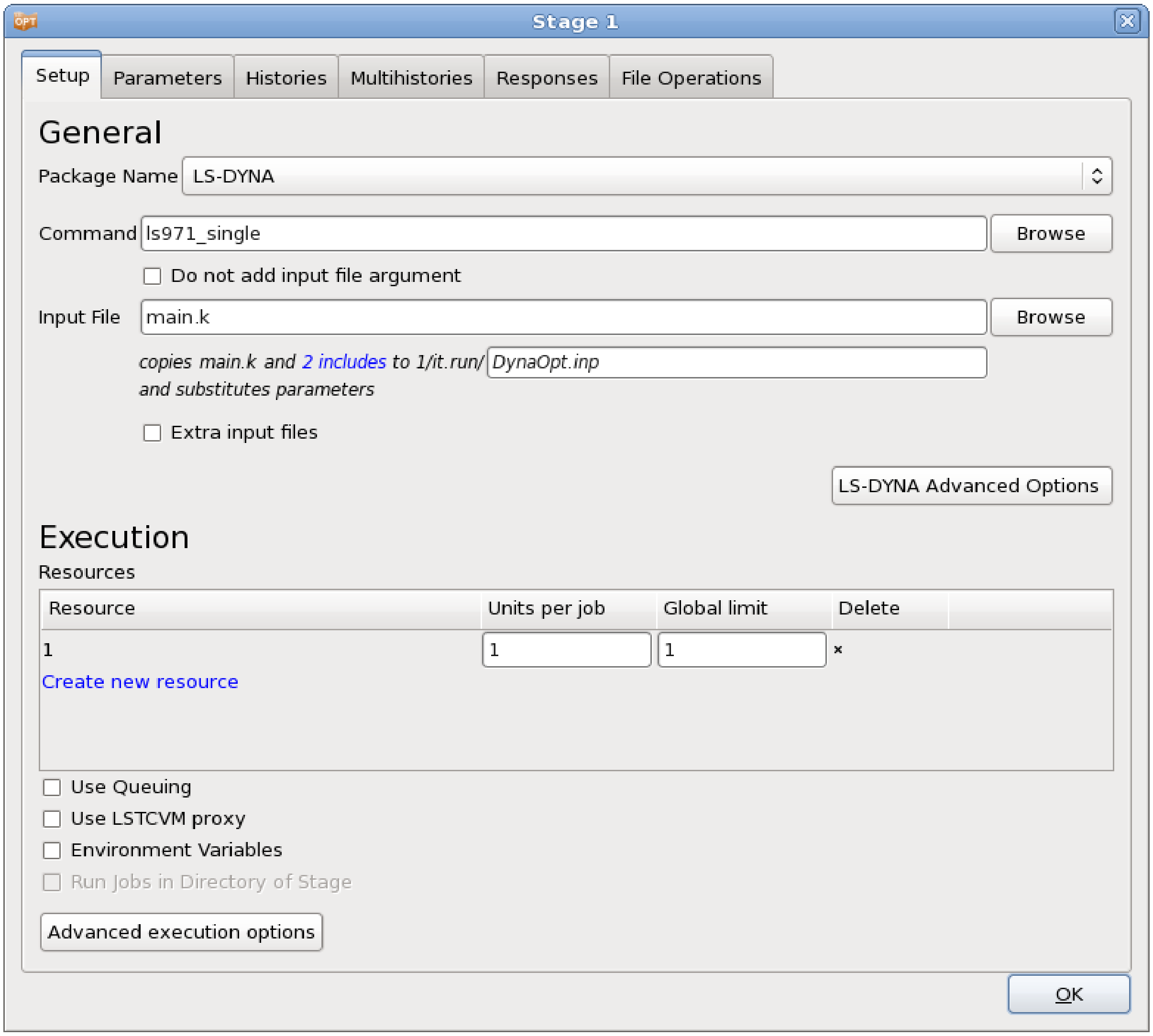
Figure 5-1 shows the general setup dialog for a Stage in the process. The options are described in Table 5-1.

Table 5-1: Stage dialog Setup options: General options

Option	Description	Reference
Package Name	The following software package identifiers are available:	
	LS-DYNA	Section 5.3.1
	LS-INGRID	Section 5.3.3
	LS-OPT	Section 5.3.9
	LS-PREPOST	Section 5.3.2
	LS-TaSC	Section 5.3.10
	ANSA	Section 5.3.5

	Excel	Section 5.3.10
	HyperMorph	Section 5.3.6
	Matlab	Section 5.3.13
	META Post	Section 5.3.7
	TrueGrid	Section 5.3.4
	User-Defined	Section 5.3.12
	User-Defined Postprocessor	Section 5.3.15
Command	Command to execute the solver.	Section 5.2.1
Do not add input file argument	Prevents LS-OPT from appending a standard input deck name to the execution command during run-time.	Section 5.2.1
Use default command	Path to the solver executable filled in automatically (only available for LS-OPT stage).	Section 5.3.9
Display graphics	Don’t run LS-PrePost in batch mode for debugging (only available for LS-PrePost stage).	Section 5.3.2
Input File	Parameterized input file for the preprocessor or solver. The specification of an input file is not required for a user-defined solver. The parameterization of the input file is explained in Section 5.2.4.	Section 5.2.2
(*n* includes)	LS-OPT displays the number of include files parsed for parameters and copied to the run directories. A list containing the include file names is accessible by clicking on the hyperlink.	Section 5.3.1
Name of standard input deck	Default standard input deck name depending on package. This can be edited in case another file name is required. Changes are only required in exceptional cases.	Section 5.2.1
Extra input files	A list of extra input files can be provided. The files are copied to the run directories from any user-defined source directory. Parameter values are substituted by default, but parsing can be omitted.	Section 5.2.2

	LS-DYNA Include files do not have to be specified as they are automatically and recursively searched by LS-OPT when given the name of the main input file. This feature is also supported for certain packages under the user-defined solver type (Section 5.3.12).	
Model Database (ANSA)	ANSA binary database file, typically with the extension *.ansa*	Section 5.3.5
Output File (HyperMorph, µETA, Matlab)	HyperMorph: nodal output file produced by Templex	Section 5.3.6
	µETA: output file used for parsing the history and response names	Section 5.3.7
	Matlab: output file containing response and history definitions	Section 5.3.13
Session file (µETA)	File containing information about which results to extract	Section 5.3.7
Excel File	Input File template for parameterizing and running Excel jobs.	Section 5.3.10
Do not copy Excel file to job folder	Avoid copying of potentially big Excel input file to each run directory and modify the original file instead. Option available only if one job is run at a time.	Section 5.3.10
Input definitions	Parameterization of the Excel input file	Section 5.3.10
LS-DYNA Advanced Options	Advanced interfacing options for LS-DYNA.	Section 5.3.1



General

Package Name: LS-DYNA

Command: ls971_single

□ Do not add input file argument

Input File: main.k

copies main.k and 2 includes to 1/it.run/ DynaOpt.inp and substitutes parameters

□ Extra input files

Execution

Resources

Resource	Units per job	Global limit	Delete
1	1	1	×

□ Use Queuing

□ Use LSTCVM proxy

□ Environment Variables

□ Run Jobs in Directory of Stage

Figure 5-1: Stage dialog Setup panel

5.2.1. Command

The command to execute the solver must be specified. The command depends on the solver type and can be an executable program or a script. Since a standard input deck name (also called the base file name) is automatically appended during run-time the solver input file name argument should be omitted by default. See respective package interface sections for details. In the case of the standard solvers, the appropriate syntax is automatically used (e.g. `i=DynaOpt.inp` for LS-DYNA). The execution command may include any number of additional arguments.

The base file name can be changed. This is useful when the output file of one stage becomes the input of the dependent stage (see Section 5.9).

Remarks

1. The command must be specified in one of the following formats:

- *Browse*. If browsing the project directory or a directory relative to the project directory, LS-OPT automatically prepends the project directory environment ${LSPROJHOME} to the execution command.
- *Absolute path,* e.g. "/origin/users/john/crash/runmpp" ("~/…" and "$HOME/…" are not supported)
- If the executable is located in a directory which is in the execution path, the command can be specified using only the name of the respective executable, e.g. "ls971_single"

2. Any path containing spaces must be surrounded with quotation marks. Quotation marks are automatically added if the file is added via browsing. E.g. “C:\Program Files\LSTC\LS-DYNA\ls-dyna.exe”
3. *Linux:* Do not specify the command `nohup` before the solver command and do not specify the UNIX background mode symbol `&`. These are automatically taken into account.
4. *Linux:* The command name must not be an alias.

5.2.2. Input Files

LS-OPT handles two main types of solver input files, namely

1. the main input file and
2. extra input files.

LS-OPT converts the input template to an input deck for the preprocessor or solver by replacing the original parameter values (or labels) with new values determined by the sampling procedure. The specification of an input file is not required for a user-defined solver.

For LS-DYNA and most of the preprocessor interfaces, LS-OPT automatically searches for include files specified in the main input file, see Table 5-2. Include files can be specified recursively, i.e. there can be include file specifications in include files. The user-defined stage type also supports these features, but only for certain solver types (see 5.3.12).

Input files are copied to the run directories, parsed to substitute parameter values and renamed. Each stage type has its own standard input file name, e.g. for LS-DYNA, the file is renamed to *DynaOpt.inp*. For remote runs, input files are automatically transmitted to a computer cluster.

A record of the specified input files and parameters is displayed in the GUI but can also be checked in the `lsopt_input` file.

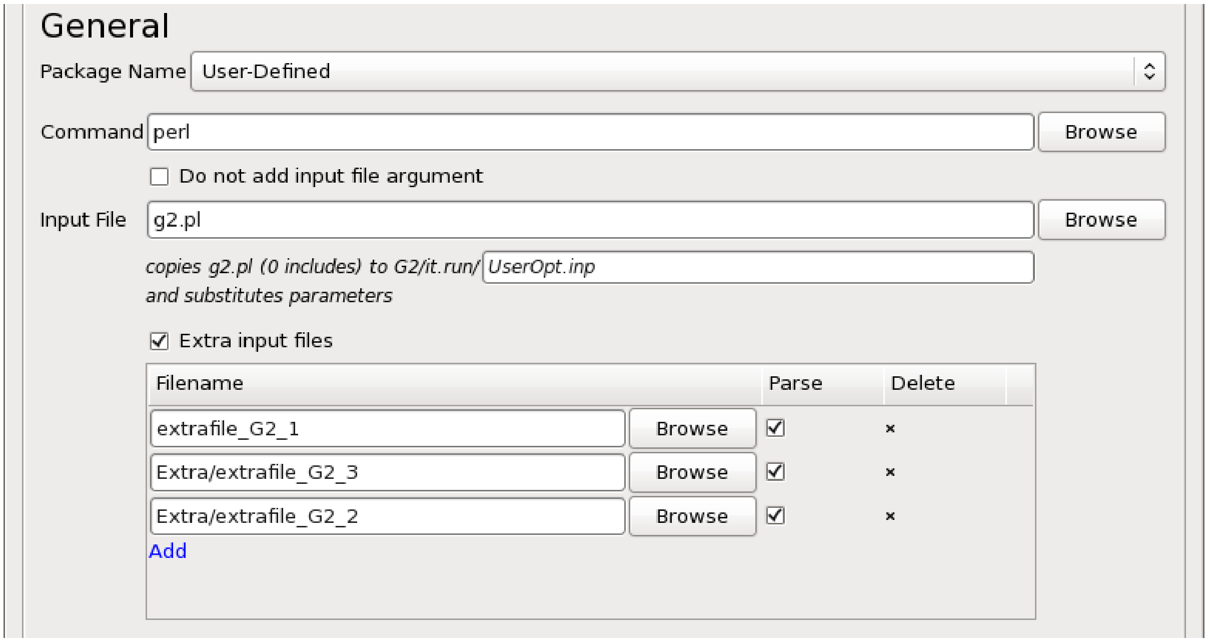
5.2.3. Extra input files

Extra files can be added for copying to run directories and substituting variables, Figure 5-2. For remote runs, extra input files are automatically transmitted to a computer cluster.

The files can be placed in any directory and are copied to the run directories during the setup phase. Parameters can be specified in the extra files using the native format (e.g. `*PARAMETER` for LS-DYNA) or the generic LS-OPT format (`<<parameter>>`), see Section 5.2.4. LS-OPT will parse the files for variable names if the **Parse** option is selected. In this case, parameters are listed on the **Parameters** page and imported to the **Setup** dialog as constants. The user can then change them to variables.

If the user wants a file to be copied to the run directories, but not parsed for parameters, parsing can be skipped by leaving **Parse** unchecked. This feature is typically used to move binary files to the run directories.

Extra files are also used in a multi-level setup (Section 5.3.9) to move input files to the lower level. In this case, **Parse** should also be left unchecked to avoid premature substitution at the upper level, before processing by the lower level LS-OPT run.



General

Package Name: User-Defined

Command: perl [Browse]

□ Do not add input file argument

Input File: g2.pl [Browse]

copies g2.pl (0 includes) to G2/it.run/ UserOpt.inp
and substitutes parameters

☑ Extra input files

Filename		Parse	Delete
extrafile_G2_1	Browse	☑	×
Extra/extrafile_G2_3	Browse	☑	×
Extra/extrafile_G2_2	Browse	☑	×

Add

Figure 5-2: Definition of Extra Input Files

Note that LS-DYNA include files do not have to be specified as extra files, since these are automatically processed. However, if the user has parameters in include files with a relative (e.g. `MyFiles/geometry.inc`) or absolute path (`/home/jo/LSOPT/MyFiles/Material59.inc`), these include files must be specified as extra input files in order to force copying to the run directory. The path option is mainly used to prevent the copying (and hence duplication) of very large files. Some user-defined solver types also support this feature (see 5.3.12).

`*INCLUDE` specifications pertaining to extra files should not include any path specifications since the files are automatically copied to the run directory and will reside together with the main input file.

5.2.4. Parameterization of Input Files

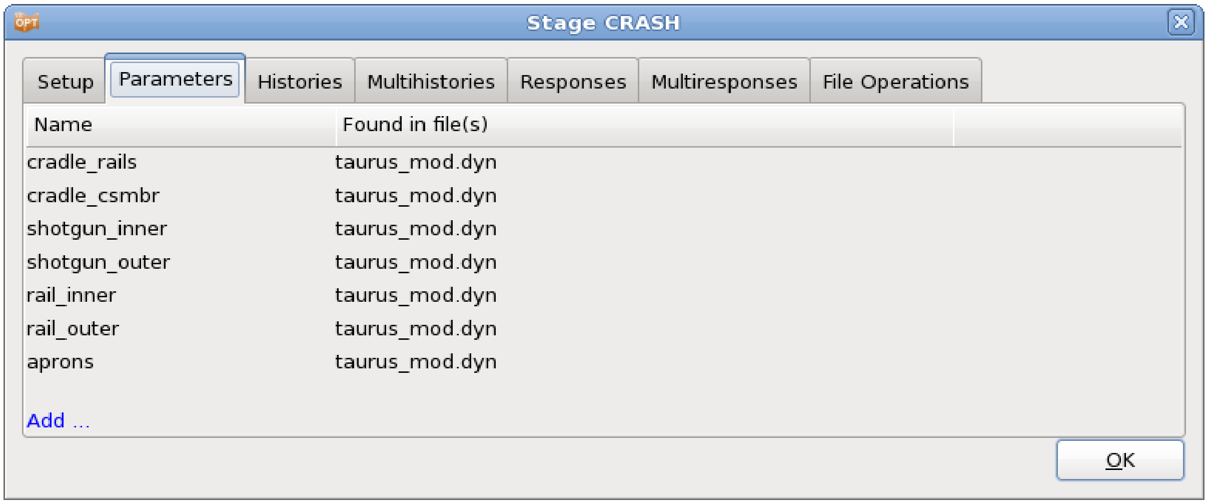
For all stage types, input files can be parameterized using the User-defined parameter format, Section 5.2.5. For the packages listed in Table 5-2, LS-OPT supports native parameters, see the respective package interface section for details. Native parameter types are also supported for certain solvers specified under user-defined solver types (see 5.3.12).

LS-OPTui will automatically recognize the native and User-defined parameters for the formats indicated in the table and list them on the **Parameters** panel, Figure 5-3. Parameters found in input files are also displayed as ‘Constants’ in the **Setup** dialog ‘Parameter Setup’ panel. The user can then change these constants to variables or dependents. The parameter names cannot be changed in the GUI so, if desired, must be changed in the original input file(s). A lock icon adjacent to the variable name indicates that the parameter names were imported from the input or include files.

Table 5-2: Parameters and include files

Package	**Native parameters recognized in input file**	**User-defined Parameter Format recognized (see Section 5.2.4)**	**Include files recognized in input file**	**Reference**
LS-DYNA®	Yes	Yes	Yes	Section 5.3.1
LS-PREPOST®	Yes	Yes	Yes	Section 5.3.2
ANSA[1]	Yes	Yes	Yes	Section 5.3.5
HyperMorph[2]	Yes	Yes	No	Section 5.3.6
PRIMER[3]	Yes	Yes	Yes	Section 5.3.7
Matlab	Yes	Yes	No	Section 5.3.13
TrueGrid[4]	No	Yes	Yes	Section 5.3.4
LS-INGRID	No	Yes	Yes	Section 5.3.3
LS-OPT	Yes	No	No	Section 5.3.9
LS-TaSC	N/A	Yes	No	Section 5.3.10
Excel	N/A	No	No	Section 5.3.11
User-defined	N/A	Yes	No	Section 5.3.12

[1] BETA CAE Systems S.A.
[2] Registered Trademark of Altair Engineering, Inc.
[3] Oasys Ltd.
[4] Registered Trademark of XYZ Scientific Applications, Inc.



Name	Found in file(s)
cradle_rails	taurus_mod.dyn
cradle_csmbr	taurus_mod.dyn
shotgun_inner	taurus_mod.dyn
shotgun_outer	taurus_mod.dyn
rail_inner	taurus_mod.dyn
rail_outer	taurus_mod.dyn
aprons	taurus_mod.dyn

Figure 5-3: Parameter panel: list of parameters found in stage input files

The ‘include’ files are also scanned wherever this feature is available making it nonessential to define extra files. Include files which are specified with a path, e.g. “`../../car5.k`” or “`/home/jim/ex4a/car6.k`” are not copied to the run directories and no parameter substitutions will be made in these files. This is solely to prevent unnecessary file proliferation. The user must however ensure that files, which are to be distributed to remote nodes through a queuing system (see Appendix H.3 , Remote job scheduling), do not contain any path specifications. These files are automatically transmitted to the relevant nodes where the solver will be executed. See also Section 5.3.1.

If parameters are specified in include files with path specifications, these files should be specified as extra files if the user wants them to be parsed and copied to the run directories, Section 5.2.2.

The User-defined parameter format described next is recognized in all types of input files.

5.2.5. The User-defined parameter format

LS-OPT provides a generic format that allows the user to substitute parameters in any type of input file, except the LS-OPT <name>.lsopt input file. The parameters or expressions containing parameters must be labeled using the double bracketed format **`<<`*`expression`*`:[i]`*`field-width`*`>>`** in the input file.

The *expression* field is for a FORTRAN or C type mathematical expression that can incorporate constants, design variables or dependents. The optional **`i`** character indicates the integer data type. The field width specification ensures that the number of significant digits is maximized within the field width limit. The default width is 10 (commonly used in e.g. LS-DYNA input files) for numeric fields. E.g. a number of 12.3456789123 will be represented as `12.3456789` and 12345678912345 will be represented as `1.23457e13` for a field width of 10.

A field width of zero implies that the number will be represented in the “%g” format for real numbers or “%ld” format for integers (*C* language). For real numbers, trailing zeros and a trailing decimal point will not be printed. This format is not suitable for LS-DYNA as the field width is always limited. Real numbers will be truncated if specified as integers, so if rounding is desired the “nearest integer” expression should be used, e.g. **`<<nint(`*`expression`*`)>>`**.

String parameters are indicated by the character **c**, ***<<expression*: c *field-width>>*.** For string parameters, the default width is the length of the replacement string up to a maximum of 64 characters. A field width of zero implies that the entire replacement string is printed (same as not specifying a width).

Examples

Inserting the relevant design variable or expression into the preprocessor command file requires that a preprocessor command such as

```
create fillet radius=5.0 line 77 line 89
```

be replaced with

```
create fillet radius=<<Radius*25.4:0>> line 77 line 89
```

where the design variable named `Radius` is the radius of the fillet and no trailing or leading spaces are desired. In this case, the radius multiplied by the constant 25.4 is replaced. Any expression can be specified.

An alternative option would be to specify:

```
create fillet radius=<<Radius_scaled:0>> line 77 line 89
```

while specifying the *dependent* Radius_scaled as a function of independent variable Radius, such that Radius_scaled = Radius * 25.4 . This specification is done in the ‘Setup’ dialog.

Similarly if the design variables are to be specified using a Finite Element (LS-DYNA) input deck then data lines such as

```
*SECTION_SHELL
1, 10, , 3.000
0.002, 0.002, 0.002, 0.002
```

can be replaced with

```
*SECTION_SHELL
1, 10, , 3.000
<<Thickness_3>>,<<Thickness_3>>,<<Thickness_3>>,<<Thickness_3>>
```

to make the shell thickness a design variable.

An example of an input line in a LS-DYNA structured input file is:

```
* shfact z-integr printout quadrule
.0 5.0 1.0 .0
* thickn1 thickn2 thickn3 thickn4 ref.surf
<<Thick_1:10>><<Thick_1:10>><<Thick_1:10>><<Thick_1:10>> 0.0
```

The field-width specification used above is not required since the default is 10. Consult the relevant User’s manual for rules regarding specific input field width limits.

5.2.6. System variables

System variables are internal LS-OPT variables. There are two system variables, namely `iterid` and `runid`. `iterid` represents the iteration number while `runid` represents the run number within an iteration. Hence the name of a run directory can be represented by: `iterid.runid`. System variables are useful for using files such as postprocessing files that were already created in an earlier stage, but which are re-used in the current stage. An LS-DYNA example of using system variables is as follows:

```
*INCLUDE
../../Case1/<<iterid:i0>>.<<runid:i0>>/frontrail.k
```

After substitution the second line might become:

```
../../Case1/1.13/frontrail.k
```

so that the current stage will always include the file in the corresponding directory in Case1.

The `i0` format forces an integer specification (see Section 5.2.5 for a more detailed description). Unfortunately the feature cannot be used with LS-DYNA `*PARAMETER` parameters.

In an alternative, simpler approach to achieve similar efficiency, LS-OPT also allows pre-processing as a first Stage of a process to generate a set of solver input files. This single Stage can be followed by multiple parallel simulation Stages using the same files. These files are copied from the preprocessing Stage to the simulation Stages. See Section 3.2.4.

5.2.7. How to avoid copying and parsing an include file

In some cases files might be very large, but they contain no parameters, so need not be parsed. For very large files, this can save a considerable amount of time. The steps are the following:

1. Unset “Do basic check for missing *DATABASE cards”.
2. Specify the name of the include file with an absolute path, e.g. “../../largeincludefile.k”.
3. Specify the exact full pathname of the include file as an *extra* input file. E.g. if the file was specified as “../../largeincludefile.k” in the keyword file, it should also be specified as extra file “../../largeincludefile.k”.
4. Do not select the “Parse” check box for this file.

It should be noted that if a file is not parsed, include files without paths specified in this file (for the purpose of copying to the run directory) cannot be detected.

5.3. Package Interfaces

5.3.1. LS-DYNA

The file `DynaOpt.inp` is created from the LS-DYNA input template file. By default, LS-OPT appends `i=DynaOpt.inp` to the solver command. Parameterization of the input file can be done using the User-defined parameter format or the *PARAMETER keyword. Include files in input files are recognized and parsed, see below for further information.

The LS-DYNA restart command will use the same command line arguments as the starting command line, replacing the `i=`*`input file`* with `r=runrsf`.

The *PARAMETER format

This is the recommended format. The parameters specified under the LS-DYNA `*PARAMETER` keyword are recognized by LS-OPT and will be substituted with a new value for each of the multiple runs. These parameters should automatically appear in the Parameter list of the GUI upon specification of the solver input file name. LS-OPT recognizes the "**`i`**", "**`r`**" and "**`c`**" formats for integers, real numbers and strings respectively and will replace the number or string in the appropriate format. Note that LS-OPT will ignore the `*PARAMETER_EXPRESSION` keyword so it may be used to change internal LS-DYNA parameters without interference by LS-OPT.

For details of the `*PARAMETER` format please refer to LS-DYNA User's Manual.

LS-DYNA include files

The handling (parsing, copying and transmitting) of include files by LS-OPT is automated. The following rules apply:

1. Include files may also contain parameters and are also parsed and copied (or transmitted) if the include file is specified in the keyword file *without a path*, for example:

   ```
   *INCLUDE

   input.k
   ```

2. If a path is specified for an include file, e.g.

   ```
   *INCLUDE

   C:\path\myinputfiles\input.k
   ```

   the file will not be copied, parsed or transmitted.

3. If the main input file is placed in a subdirectory of the main working directory and is specified with a relative path, e.g. `myinputfiles/input.k`, the directory (in this case `myinputfiles`) becomes a file environment for any include files which may also be placed in this directory. Therefore all include files specified without a path will automatically be copied (or transmitted) from this sub-directory (`myinputfiles`) to the run directories.

LS-DYNA/MPP

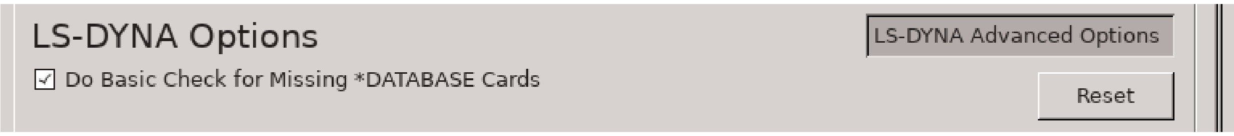
The LS-DYNA MPP (Message Passing Parallel) version can be run using the LS-DYNA option in the **Stage** dialog of LS-OPTui. The following run command is an example of how an MPP command can be specified:

```
mpirun -np 2 lsdynampp
```

where `lsdynampp` is the name of the MPP executable.

LS-DYNA Advanced Options

LS-DYNA advanced options are available in the Stage dialog by selecting the *LS-DYNA Advanced Options* button, Figure 5-4.



LS-DYNA Options

☑ Do Basic Check for Missing *DATABASE Cards

Figure 5-4: Stage Setup LS-DYNA advanced options

Table 5-3: LS-DYNA Advanced Options

Option	Description
Do Basic check for Missing *DATABASE Cards	Check if the required binout data types and the required nodes and/or elements are requested in the LS-DYNA input deck. For further details, see below.

Checking the *DATABASE cards

LS-OPT can perform some basic checks of the ***DATABASE** cards in the LS-DYNA input deck. The checks will be done using the input deck of the first run of the first iteration. The items checked are:

- Whether the required binout data types are requested in the LS-DYNA input deck. For example, if LS-OPT uses airbag data, then the LS-DYNA deck should contain a ***DATABASE_ABSTAT** card requesting binout output. Note that LS-DYNA smp does not write the results into the binout file when the default settings are used. Therefore, the BINARY option in the card must be set to either 2 or 3 for both LS-DYNA smp and mpp.
- Whether the required nodes and/or elements are requested in the LS-DYNA output. For example, if the LS-OPT output request refers to a specific beam, then a ***DATABASE_HISTORY_BEAM** or a ***DATABASE_HISTORY_BEAM_SET** card must exist and refer to the beam in question. Note that ***SET_*option*_GENERAL** or ***SET_option_COLUMN** card will not be interpreted and that an output entity specified using ***SET_*option*_GENERAL** or ***SET_option_COLUMN** may be flagged incorrectly as missing; switch off the checking in this case.

5.3.2. LS-PREPOST

The file `LsPrepostOpt.inp` is created from the LS-PREPOST input template file. LS-OPT automatically appends `runc=LsPrepostOpt.inp 2> /dev/null > /dev/null`” to the command.

If the option *Display graphics* is selected, LS-OPT appends “`c=LsPrepostOpt.inp 2> /dev/null > /dev/null`”.

LS-PREPOST input file example with include

test01.cfile:

```
$# LS-PrePost command file created by LS-PREPOST 3.0 - 1Mar2010(17:08)
$# Created on Apr-06-2010 (13:42:14)
cemptymodel
openc command "para01.cfile"
```

```
genselect target node
occfilter clear
genselect clear
genselect target node
occfilter clear
genselect clear
meshing boxshell create 0.000000 0.000000 0.000000 &size &size &size &num &num
&num
ac
meshing boxshell accept 1 1 1 boxshell
genselect target node
occfilter clear
refcheck modelclean 9
ac
mesh
save keyword "lsppout"
exit
```

para01.cfile

```
parameter size 1.0
parameter num 2
```

5.3.3. LS-INGRID

The file `ingridopt.inp` is created from the LS-INGRID input template file. LS-OPT appends automatically “`i=ingridopt.inp -d TTY`” to the command. Only the User-defined parameter format is supported.

5.3.4. TrueGrid

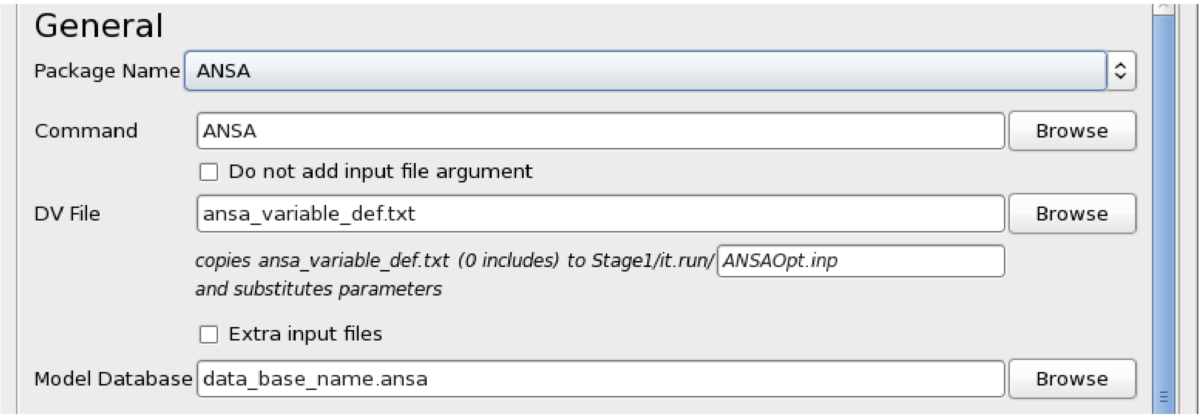
The file `TruOpt.inp` is created from the TrueGrid input template file. LS-OPT appends automatically `‶i=TruOpt.inp" to the command.` Only the User-defined parameter format is supported.

The TrueGrid input file requires the line:

```
write end
```

at the very end.

5.3.5. ANSA[5]



General

Package Name: ANSA

Command: ANSA

□ Do not add input file argument

DV File: ansa_variable_def.txt

copies ansa_variable_def.txt (0 includes) to Stage1/it.run/ ANSAOpt.inp
and substitutes parameters

□ Extra input files

Model Database: data_base_name.ansa

Figure 5-5: Stage Setup for ANSA

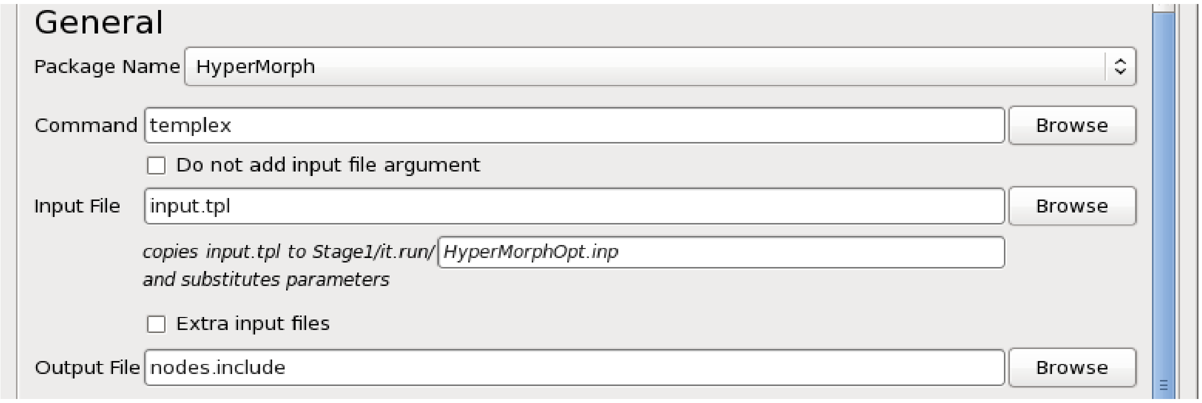
The ANSA preprocessor can be interfaced with LS-OPT allowing for shape changes to be specified. Several files must be specified:

1. Command: *ANSA executable*, typically named *ansa.sh*. Do not use an alias. Using the ANSA command line option *–lm_retry* is recommended.
2. DV File: *ANSA Design parameter file*, typically with the extension *.txt* or *.dat*. This file is generated using ANSA and LS-OPT will read the ANSA design parameter names, types and values from this file. If LS-OPT already has a design variable with the same name then this variable will be used to drive the value of the ANSA parameter.
3. Model Database: *ANSA binary database*, typically with the extension *.ansa*.

ANSA can produce multiple output files. These files can be used as LS-DYNA input files or include files (specified under `*INCLUDE`) in downstream stages. Make sure to specify the output files in the ANSA optimization task without a path to generate them in the respective run directory.

[5] BETA CAE Systems SA

5.3.6. HyperMorph



General

Package Name: HyperMorph

Command: templex

□ Do not add input file argument

Input File: input.tpl

copies input.tpl to Stage1/it.run/ HyperMorphOpt.inp
and substitutes parameters

□ Extra input files

Output File: nodes.include

Figure 5-6: Stage Setup for HyperMorph

To allow the specification of shape variables, the geometric preprocessor HyperMorph[6] has been interfaced with LS-OPT. Several files must be specified:

1. Command: templex command
2. Input file*:* At the top, the variables are defined as:

   ```
   {parameter(DVAR1,"Radius_1",1,0.5,3.0)}
   ```

3. Output File: Templex produces a nodal output file, this file can e.g. be used as an include file in a downstream stage.

The command will enable LS-OPT to execute the following command in the default case:

```
/origin 2/john/mytemplex/templex input.tpl > nodes.include
```

or if the input file is specified as in the example:

```
/origin 2/user/mytemplex/templex a.tpl > h.output
```

Remarks

LS-OPT uses the name of the variable on the DVARi line of the input file:

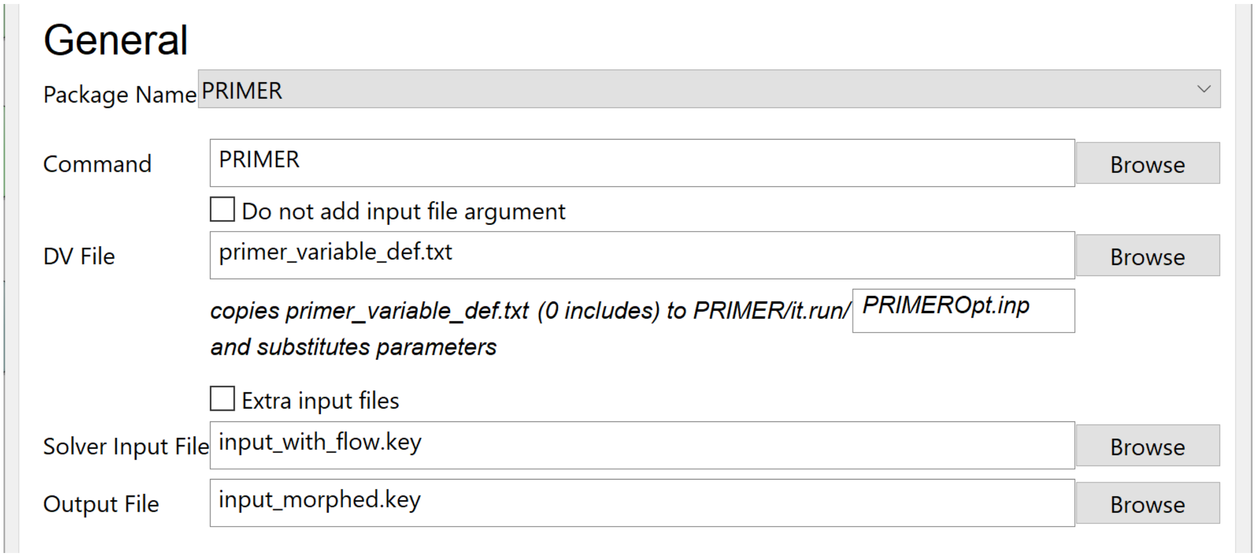
```
{parameter(DVAR1,"Radius_1",1,0.5,3.0)}

{parameter(DVAR2,"Radius_2",1,0.5,3.0)}
```

to replace the variables and bounds at the end of each line by the current values. This name, e.g. Radius_1 is recognized by LS-OPT and automatically displayed in the 'Setup' dialog. The lower and upper bounds (in this case: `[0.5,3.0]`) are also automatically displayed. The DVAR*i* designation is not changed in any way, so, in general there is no relationship between the number or rank of the variable specified in LS-OPT and the number or rank of the variable as represented by *i* in DVAR*i*.

[6] Registered Trademark of Altair Engineering, Inc.

5.3.7. Oasys PRIMER[7]

The PRIMER preprocessor can be interfaced with LS-OPT allowing for shape changes to be specified using the PRIMER morph tool. Several files must be specified.



General

Package Name PRIMER

Command PRIMER

□ Do not add input file argument

DV File primer_variable_def.txt

copies primer_variable_def.txt (0 includes) to PRIMER/it.run/ PRIMEROpt.inp
and substitutes parameters

□ Extra input files

Solver Input File input_with_flow.key

Output File input_morphed.key

Figure 5-7: PRIMER interface

1. Command: PRIMER command

   LS-OPT automatically appends the relevant arguments to apply morphing in batch mode.

2. DV File: Design variables file in JSON format generated in PRIMER:

```
{
  "name": "width",
  "type": "CONTINUOUS",
  "value": 0,
  "minimum": -20,
  "maximum": 20
 }
```

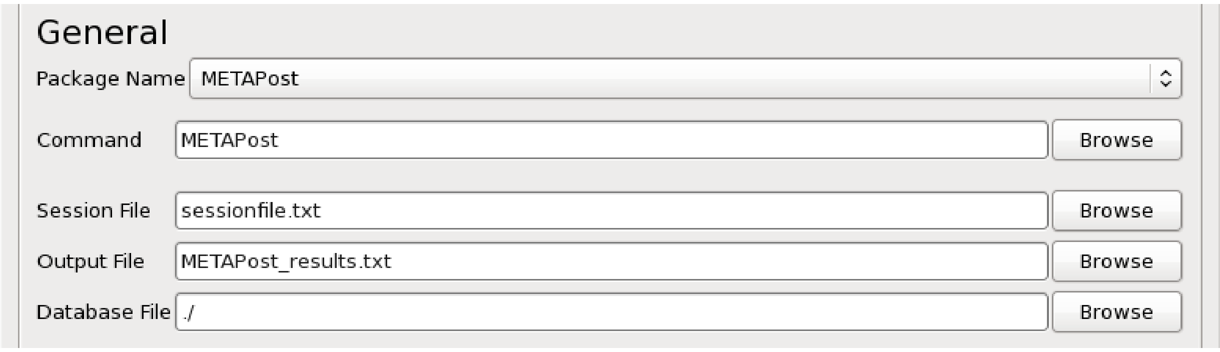
3. Solver Input File: LS-DYNA input file including morph flow defined in PRIMER. If desired, additional design variables may be defined in the input file using *PARAMETER.

4. Output File: Morphed LS-DYNA model. If no output file name is specified, the input file will be overwritten.

5.3.8. μETA[8]

The μETA interface allows extraction of data from any database it supports, so makes LS-OPT accessible to interface with any such supported solvers. This allows μETA to read results from the solver database and place them in a simple text file.

[7] Oasys Ltd.
[8] BETA CAE Systems SA



General

Package Name: METAPost

Command: METAPost

Session File: sessionfile.txt

Output File: METAPost_results.txt

Database File: ./

Figure 5-8: MetaPost interface

Several files must be specified:

1. Command: The μETA executable
2. Session File: The session file containing information about which results to extract. This can be created interactively using μETA.
3. Output File: This specification is only used for parsing the history and response names (to be automatically displayed in the GUI) during the LS-OPT setup phase (see below). The output file (result file) is the name of a file containing those results requested in the input (session) file. This is a text file so it can be easily parsed. This file has a predetermined format so that LS-OPT can automatically extract the individual results. The specified path + name is not used during the optimization run, but only during the setup phase while the user is preparing the LS-OPT input data. During this phase, the responses are parsed from a baseline result file and automatically displayed in the "Histories" and "Responses" pages of the GUI.
4. Database File: This is the path for finding the solver database. The default "./" means that μETA will look for the database locally. This specification has no effect during the optimization run as LS-OPT will always force μETA to look for the solver database locally, e.g. in the run directory *Stage_A/1.1*.

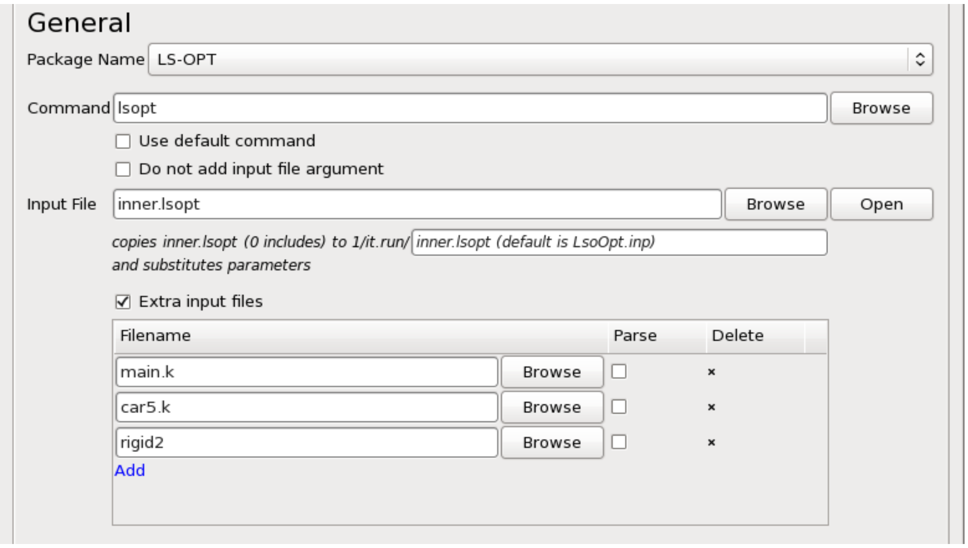
Setting up an LS-OPT problem:

1. Run μETA and use the session file thus created to create the result file. This is done manually, separately from the LS-OPT data preparation (an integrated feature might be provided in the future).
2. Open the LS-OPT GUI on the *Stage dialog* and select METAPost as the package name.
3. Specify the μETA settings in the LS-OPT GUI (see Figure 5-8). The user can browse for the μETA executable, session file and result file. The result file is the one that was created in the manual step (Step 1. above). The database path need not be changed.
4. The result file is parsed for history and response names to display in the relevant GUI pages. These can then be used to complete the optimization problem setup: define composites, objectives and constraints, etc.
5. After completion of the optimization setup, run LS-OPT.

5.3.9. LS-OPT

The LS-OPT stage allows one to extract optimized LS-OPT response values, which can then be used in another optimization with respect to a different set of variables. The LS-OPT stage can also be used to call a reliability task from an optimization task, e.g. for tolerance optimization or a robust design task using the direct Monte Carlo method at the lower level.

The LS-OPT stage simply executes another instance of the LS-OPT software in a nested optimization framework. Thus, it allows a user to set up a *Multilevel Optimization* problem, explained in Section 19.7. The LS-OPT stage setup dialog is shown in Figure 5-9.



General

Package Name: LS-OPT

Command: lsopt

□ Use default command

□ Do not add input file argument

Input File: inner.lsopt

copies inner.lsopt (0 includes) to 1/it.run/ *inner.lsopt (default is LsoOpt.inp)*
and substitutes parameters

☑ Extra input files

Filename	Parse	Delete
main.k	□	×
car5.k	□	×
rigid2	□	×

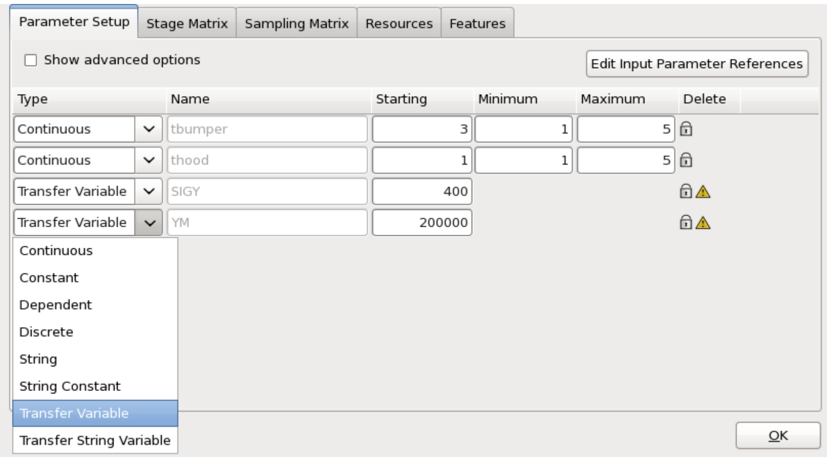
Figure 5-9: LS-OPT stage interface

The fields that need to be specified for an LS-OPT stage are as follows.

1. Command: Like all other solver interfaces, the user needs to provide the command to run LS-OPT. There is a *Use default command* option that automatically fills in the path to the LS-OPT executable being used for the setup.
2. Input file: The input file for an LS-OPT stage is a .lsopt file itself that contains the setup for an inner level LS-OPT sub-problem. The file `LsoOpt.inp` (or a user specified name) is created from the LS-OPT input template file. By default, LS-OPT appends `LsoOpt.inp` to the solver command. Parameterization of the input file is done using *Transfer Variables* (Figure 5-10).
3. Extra Files: An important aspect to note in the LS-OPT stage setup is the use of extra input files with the *Parse* option unchecked (Figure 5-16). This is important because the input files of the lower level(s) need to be passed down from the upper level while not considering the lower level variables in the upper level. The details of the directory structure for multilevel problems are presented in Appendix E.1 .

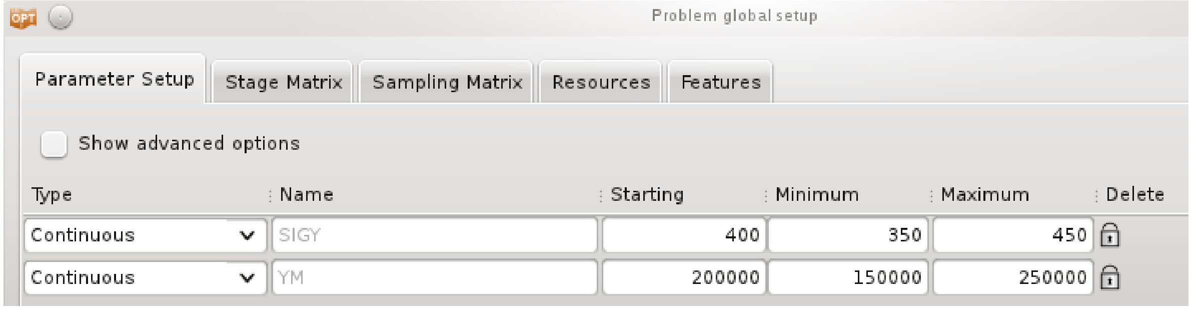
LS-OPT input file parameterization

The LS-OPT input file, i.e. the .lsopt file, is parameterized using *Transfer Variables*. The transfer variables are indicated using *type=”iconstant” in the* LS-OPT stage input file. *Continuous* and *Discrete* variables can be set as a *Transfer Variables* using the LS-OPT GUI (Figure 5-10); these are then considered as constants at that level, but can be set as variables in preceding levels. These variables are automatically detected as constants by LS-OPT and populate the outer level *Global Setup* (for which the parameterized *.lsopt* file is a stage input file). The user can either use them as constants in the outer level or set them as variables.



Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	
Transfer Variable	SIGY	400			⚠
Transfer Variable	YM	200000			⚠

Figure 5-10: Parameterization of inner level LS-OPT setup using Transfer Variables. The values of transfer variables are passed down from the upper level(s).



Type	Name	Starting	Minimum	Maximum	Delete
Continuous	SIGY	400	350	450	
Continuous	YM	200000	150000	250000	

Figure 5-11: Outer level global setup. `SIGY` and `YM` are automatically detected in the input file (i.e. inner level .lsopt file) and locked as they are Transfer Variables in the inner level.

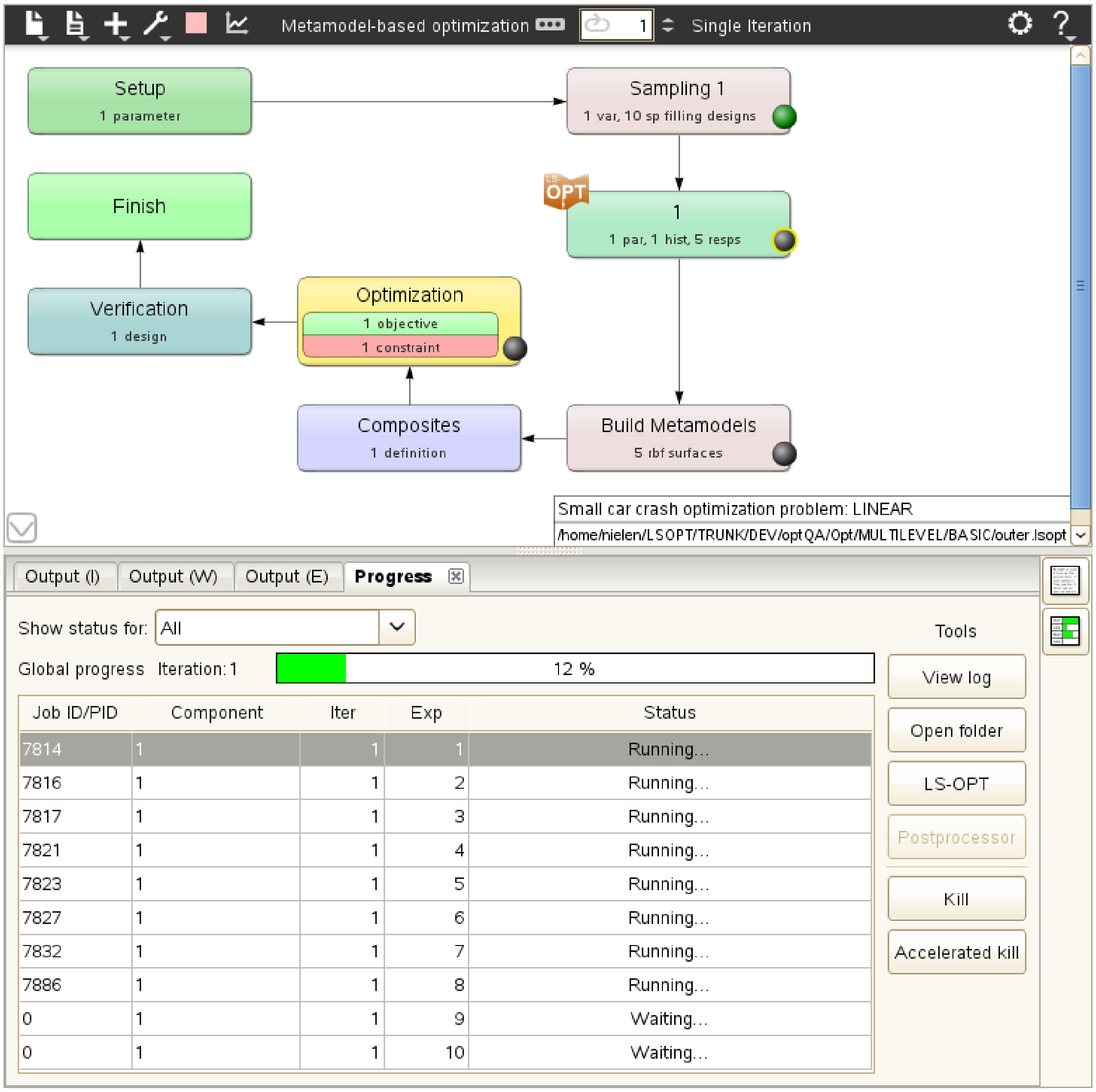
Remarks

1. The user-defined parameter format <<variable_name>> is not allowed for the LS-OPT stage.
2. LS-OPT stage responses are extracted using the *LSOPT* response type (Section 6.15).

Navigating to view lower level setups and progress

Because of the complex recursive nature of a multilevel setup, simple navigation options are provided so that lower level setups can be inspected or edited recursively starting at the main (upper level) setup. During runtime, job progress can also be viewed recursively starting at the main progress window.

1. The *Open* button opposite the *Input file* text box allows the user to navigate down to the next level and will display the GUI for inner.lsopt, see Figure 5-9.
2. While a multilevel run is in progress, the user can also navigate to display the progress of a selected lower level job by clicking on the *LS-OPT* button in the progress dialog. Lower level job progress can also be monitored using the *View log* button to display the text output, see Figure 5-12.



Small car crash optimization problem: LINEAR

/home/nielen/LSOPT/TRUNK/DEV/optQA/Opt/MULTILEVEL/BASIC/outer.lsopt

Show status for: All

Global progress Iteration: 1 12 %

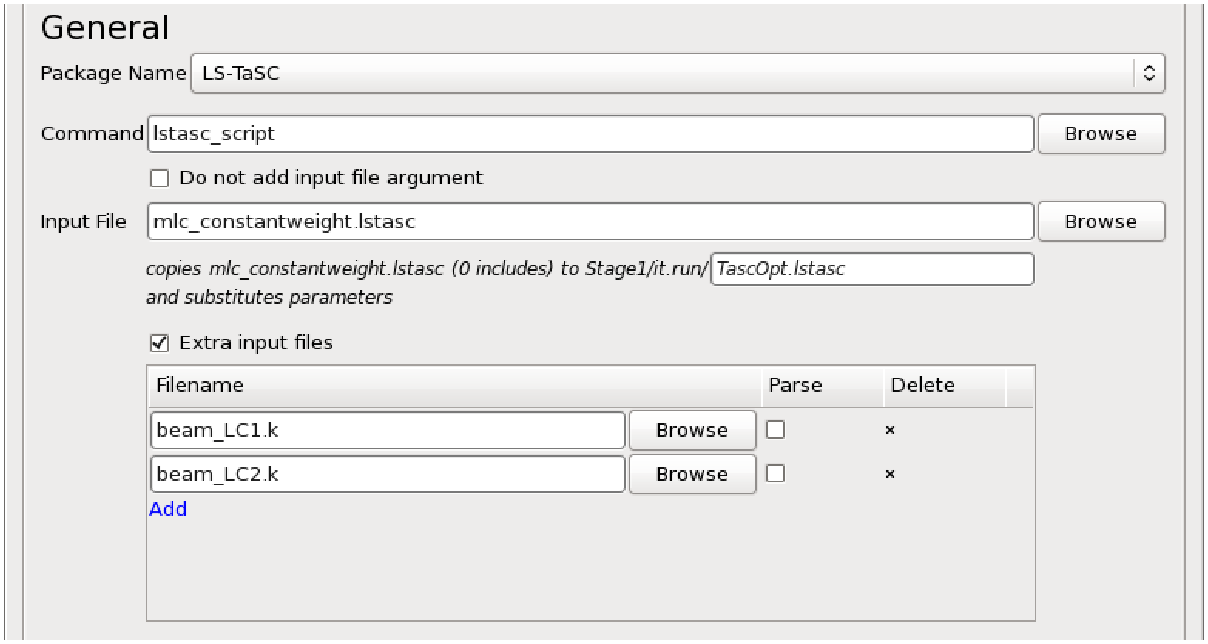
Job ID/PID	Component	Iter	Exp	Status
7814	1	1	1	Running...
7816	1	1	2	Running...
7817	1	1	3	Running...
7821	1	1	4	Running...
7823	1	1	5	Running...
7827	1	1	6	Running...
7832	1	1	7	Running...
7886	1	1	8	Running...
0	1	1	9	Waiting...
0	1	1	10	Waiting...

Figure 5-12: Progress window for the LS-OPT stage. Selecting the LS-OPT button for the selected job displays the LS-OPT GUI for that job which allows the user to monitor a lower level optimization run.

5.3.10. LS-TaSC

The LS-TaSC stage allows one to run an inner level topology optimization using LS-TaSC within an outer level LS-OPT task. This facilitates LS-TaSC to work with complex design schemes and constraints, for example, using advanced metamodels supported in LS-OPT.

The design constraints defined in the LS-TaSC project file (*.lstasc file) are automatically added as LS-OPT responses and the response values are extracted from final iteration of LS-TaSC run using lst.binout. Additional responses could be extracted from any LS-TaSC ASCII output using GenEx, Chapter 7, or the user-defined interface could be used, Section 6.13.


General

Package Name: LS-TaSC

Command: lstasc_script

☐ Do not add input file argument

Input File: mlc_constantweight.lstasc

copies mlc_constantweight.lstasc (0 includes) to Stage1/it.run/ TascOpt.lstasc
and substitutes parameters

☑ Extra input files

Filename	Parse	Delete
beam_LC1.k	☐	×
beam_LC2.k	☐	×

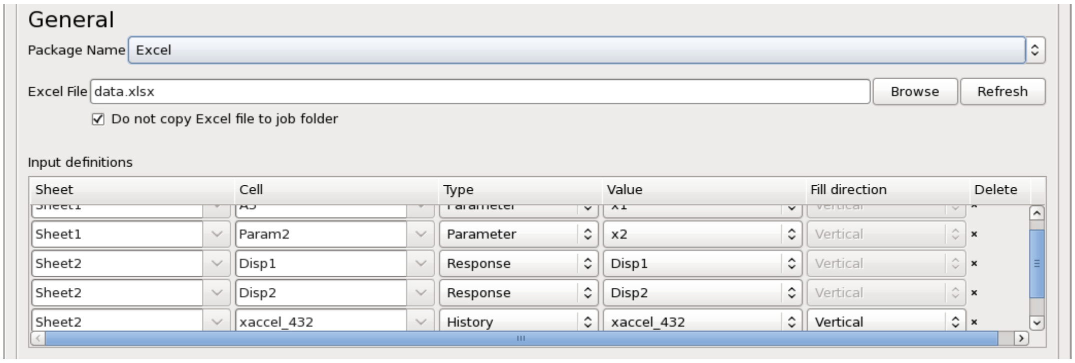
Figure 5-13: LS-TaSC stage interface

The fields that need to be specified for an LS-TaSC stage are as follows.

1. Command: The user needs to provide the command to run LS-TaSC in batch mode. Therefore, the executable to be used is lstasc_script available in the installation directory of LS-TaSC.
2. Input file: The input file for an LS-TaSC stage is the LS-TaSC project file (*.lstasc file) itself that contains the setup for an inner level LS-TaSC problem. The file `TascOpt.inp` (or a user specified name) is created from the LS-TaSC input template file. By default, LS-OPT appends `TascOpt.inp` to the solver command. Parameterization of the input file is done using the user-defined parameter format, Section 5.2.5.
3. Extra Files: An important aspect to note in the LS-TaSC stage setup is the use of extra input files with the *Parse* option unchecked (Figure 5-16). This is important because the LS-DYNA input files of the LS-TaSC run need to be passed down from the upper level.

5.3.11. Excel

An Excel stage can be used as a solver or a post-processor. It can be seen as being similar to any other solver, with the main differences lying in its parameterization and in the response and history definitions. Because the results need to be computed for several samples within an LS-OPT task, the Excel input file needs to be parameterized. This is achieved using *Input definitions* specified in the **Stage** dialog itself. These inputs may correspond to a single Excel cell or a group of cells in the input file, and are substituted for each sample (Figure 5-14).



General

Package Name: Excel

Excel File: data.xlsx

☑ Do not copy Excel file to job folder

Input definitions

Sheet	Cell	Type	Value	Fill direction	Delete
Sheet1	A3	Parameter	x1	Vertical	×
Sheet1	Param2	Parameter	x2	Vertical	×
Sheet2	Disp1	Response	Disp1	Vertical	×
Sheet2	Disp2	Response	Disp2	Vertical	×
Sheet2	xaccel_432	History	xaccel_432	Vertical	×

Figure 5-14: Excel stage interface

The attributes used for Input definitions are *Sheet*, *Cell*, *Type*, *Value* and *Fill Direction*. The details are given below.

1. The *Sheet* and *Cell* options direct LS-OPT to a unique location within the *Excel* document. A *Cell* can be assigned using the *Excel* row-column format (i.e. by typing A2, B4 etc). If cell names have been already defined in the parsed *Excel* document, LS-OPT displays all the existing names as a list under *Cell* option and the required cell can be selected directly. The displayed names under *Cell* option can also correspond to an array of *Excel* cells, used to assign LS-OPT histories.

2. *Type* and *Value* options are used to link LS-OPT design parameters, histories and responses with corresponding fields of the parsed *Excel* document. There are four different options within *Type* - *Parameter*, *Responses*, *History* and *User-Defined.*

   - *Parameter* is used to link the global LS-OPT parameters defined in *Setup* dialog to the specified cells of the *Excel* document. When *Parameter* is selected as type, all the global LS-OPT parameters defined in *Setup* dialog are listed under the *Value* option.

   - *Response* as a parameter type facilitates the use of LS-OPT responses defined in previous stages as input parameter for the current *Excel* stage. A list of responses defined in the previous stages is displayed under *Value* option and the user can select which response has to be written to the *Excel* document.

   - *History* as a type allows LS-OPT to input histories obtained from previous stages to the *Excel* document.

   - *User-defined* option as a type can be used to write histories and responses of previous stages to the *Excel* document using a command. For example the command '*type response.0'* will write the value present in file *response.0* of previous stage, to the *Excel* document; provided a file transfer operation is defined to transfer the file *response.0* from previous stage directories to the run directories of current *Excel* stage directory.

3. *Fill Direction* specifies how the history values are written to the *Excel* fields i.e. in *Vertical* or *Horizontal* directions.

If the *Global limit* for *Execution Resources* is set to 1, the option *'Do not copy Excel file to job folder'* is available in the Excel stage setup dialog. If the option is checked on then the original Excel input file template is modified for each sample analysis. This avoids copying of the (potentially large) input file to each run directory. All the possible combinations of *Input definitions* are illustrated in Figure 5-14. These are also listed below.

1. The first input definition in Figure 5-14 shows a parameter *x1* defined in *Setup* dialog of main GUI (also populated under *Value* option), has been assigned a cell *A3* in *Sheet1* of *Excel* document *data.xlsx*.
2. Similarly, if a user has assigned a name to the cell using *Name Manager* within *Excel*, all the *Sheet* specific cell names are populated as a list. In the second input definition, *Param2* is a name defined to a cell in *Sheet1* which is assigned to parameter *x2* using *Value* option.
3. The third input definition writes the response *Stage1_out_resp* obtained from previous stage to the cell *Stage2_in_resp1* of the *Excel* document *data.xlsx.*
4. The fourth input definition writes the history *Stage1_out_hist* obtained from previous stage to an array of *Excel* fields defined with name *Stage2_in_hist* in *Sheet2* of *data.xlsx* in *Vertical* direction.
5. The last input definition shows a response obtained from previous stage (where response.2 is the file with response value) is being written to a cell with name Stage2_in_resp2 of Sheet2 using User-defined option. This option allows writing values available in the output files of previous stages to the Excel document.

5.3.12. User-defined program

A user-defined solver or preprocessor can be specified by selecting User-defined in LS-OPT*ui*. The `command` can either execute a command, or a script. The substituted input file `UserOpt.inp` will automatically be appended to the command or script. Variable substitution will be performed in the input file (which will be renamed `UserOpt.inp)`. The specification of an input file is optional. In its simplest form, the user-defined program can be used in combination with the design point file: `XPoint` to read the design variables from the run directory.

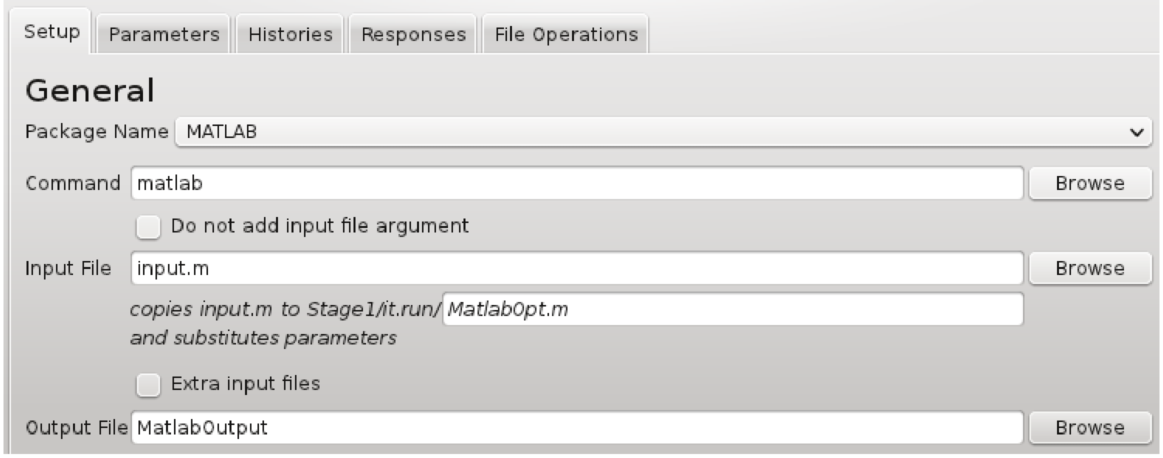
If the user-defined program does not generate a ‘Normal’ termination command to standard output, the solver command must execute a script that has as its last statement the command

```
echo ‘N o r m a l’
```

on successful execution of the program. The script could e.g. check for the existence of a file that is generated after program termination, or search for a specific string in an output file that gives information on the status of the program. If the program is not terminated successfully, the output of the script should be `E r r o r´.

5.3.13. Matlab

A Matlab stage can be specified by selecting MATLAB as the package name in the stage setup dialog of LS-OPT*ui* (Figure 5-15).



General

Package Name MATLAB

Command matlab

□ Do not add input file argument

Input File input.m

copies input.m to Stage1/it.run/ MatlabOpt.m
and substitutes parameters

□ Extra input files

Output File MatlabOutput

Figure 5-15: Matlab stage interface

The input file is a Matlab script consisting of the variable definitions using the `input` function in Matlab (e.g. `variable1 = input(‘description of the variable’);`). LS-OPT parses the input file and identifies the variable name. It then replaces the `input` function with a value during the run. Before replacement, the input file is copied to subdirectories under the stage directory. The default name for the copied file is *MatlabOpt.m*. It should be noted that this file must have the suffix *.m*. The Matlab input file must write the histories and responses in the METAPost format described in 5.3.15. In addition, it must write the termination status, as shown below, using try-catch and diary.

```
Try
  % Definition of variables x1 and x2
  x1 = input('x1-');
  x2 =input('x2:');

  % Computation of response(s) and histories
  s = x1+x2;
  h = [0 s;1 s+1;2 s+4;3 s+9];

  % Write responses and histories to MatlabOutput file
  fid = fopen('MatlabOutput','w');
  fprintf(fid,'#\n');
  fprintf(fid,'RESPONSES\n');
  % response 1
  fprintf(fid,'%d, %s, %f\n',0,'s',s);
  fprintf(fid,'END\n');
  fprintf(fid,'#\n');
  % history 1
  t=1:size(h,1);
  fprintf(fid,'HISTORY 99: h\n');
  for i=1:size(h,1)
    fprintf(fid,'%f, %f\n',t(i),h(i));
  end
  fprintf(fid,'END\n');fprintf(fid,'#\n');
  ChkClose=fclose(fid);
  % Write Normal termination status
  diary matstatus;
  disp('N o r m a l   t e r m i n a t i o n');
  diary off
catch
  % Write error termination status
  diary matstatus;
  disp('E r r o r   t e r m i n a t i o n');
  diary off;
end
exit
```

An output file also needs to be provided in the stage dialog that contains the response and history definitions. LS-OPT automatically populates the histories and responses to be extracted based on the definitions in this file. The output file must have the same format as METAPost or User-defined post-processor (See Section 5.3.15).

5.3.14. Third Party solvers

LS-OPT supports certain popular Finite Element Analysis solvers under the *User-defined* solver type. For these solver types all the syntax rules (e.g. recursive include files, parameter keywords, etc.) associated with the input file are obeyed so that parameters can automatically be imported to the LS-OPT setup dialog.

LS-OPT recognizes the solver type by initially parsing the first line of the main input file. This line should be a comment line which contains the name of the package it represents.

Special response interfaces are not available, but response and history extraction are supported using

- GenEx (Chapter 7)
- the user-defined post-processor (Section 5.3.15)
- commercially available post-processors supported by LS-OPT (see e.g. Section 5.3.8)
- user-defined history or response interface (Section 6.13).

5.3.15. User-defined post-processor

The postprocessor allows extraction of data from any database it supports, so makes LS-OPT accessible to interface with any such supported solvers. This allows the postprocessor to read results from the solver database and place them in a simple text file or files for individual extraction of results.

In the case of user-defined post-processor, the full command needs to be provided, because LS-OPT does not internally construct the command using the input, database and result files. The output file needs to be written in the same format as for the μETA package. The format is as follows:

```
#
RESPONSES
0, Weight, 0.591949043101576
1, StressL, 3.74281176328897
2, StressR, 1.99975762786926
END
#
HISTORY 99 : his1
```

```
0,0
0.0795849328001081,0.23516125192977
0.159169865600216,0.274354793918065
0.238754798400324,0.31354833590636
0.318339731200433,0.352741877894655
0.397924664000541,0.39193541988295
#
END
#
RESPONSES
END
#
HISTORY 100 : his2
0,0
0.0795849328001081,0.627096671812721
0.159169865600216,0.666290213801015
0.238754798400324,0.705483755789311
0.318339731200433,0.744677297777606
0.397924664000541,0.783870839765901
#
END
```

Setting up an LS-OPT problem is similar to μETA, except that **User-defined Postprocessor** is selected as the package, and the session file and database path need not be provided as the related information is available in the command.

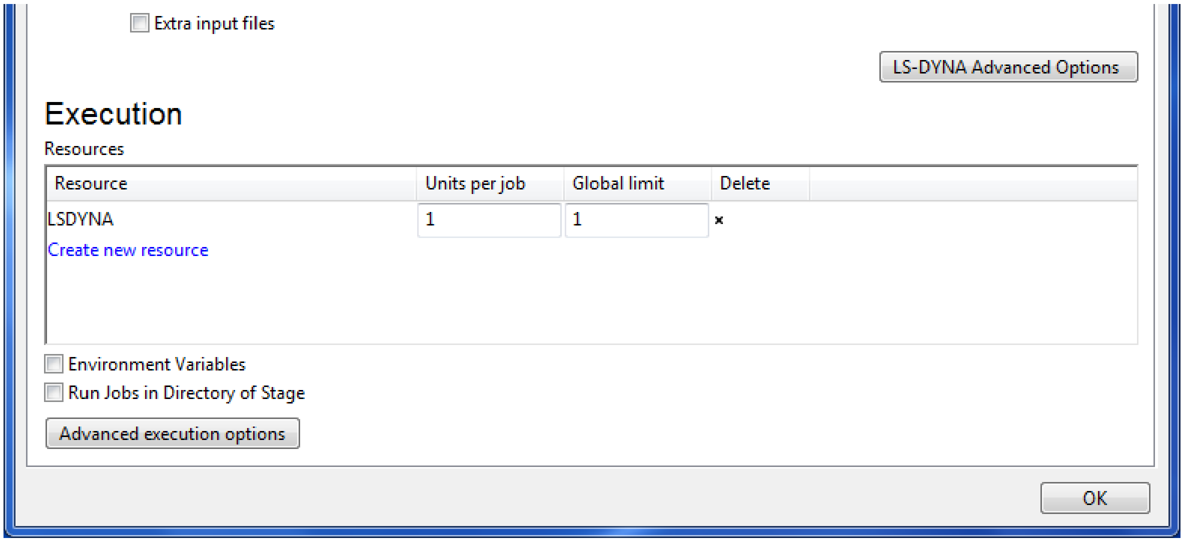
It is also possible to run μETA as a user-defined post-processor. In this case, the command provided in "`fullcommandscript`" is:

```
<metapost_executable> -b -s -foregr <path/sessionfile> "<database_path>"
"<path/result_file>"
```

Unlike in the case of μETA, the full command is not constructed internally by LS-OPT. Therefore, metapost_executable, path/sessionfile, database_path, and path/result_file need to be provided in `fullcommandscript.` Because all the information is available in the command, it is not necessary to provide the input and database files separately in this case.

The output file name must however be specified for the following reason. The output file is parsed for history and response names to import and display in the relevant GUI pages. These can then be used to complete the optimization problem setup: define composites, objectives and constraints, etc.

5.4. Solver Execution



□ Extra input files

Execution

Resources

Resource	Units per job	Global limit	Delete
LSDYNA	1	1	×
Create new resource			

□ Environment Variables

□ Run Jobs in Directory of Stage

Figure 5-16: Stage dialog Setup panel: Execution options

Table 5-4: Stage dialog Setup options: Execution options

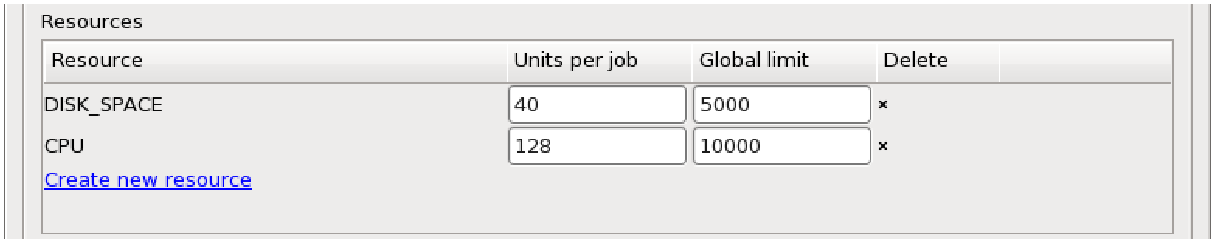
Option	Description	Reference
Resources	Settings for concurrent processing	Section 5.4.1
Environment Variables	Environment variables that will be set before executing a solver command.	Section 5.4.2
Run jobs in Directory of Stage	If multiple stages are defined, the command can be executed in the directory of another stage.	-
Advanced execution options	Options related to retry of job submissions for Abnormal Termination	Section 5.4.3

5.4.1. Specifying Computing Resources for Concurrent Processing

Resource definitions are used to calculate the number of jobs that are submitted concurrently. Multiple resource limits can be defined for each stage. The resource attributes consist of *Units per job* as well as the *Global limit* (see Figure 5-17). This feature is non-dimensional and therefore allows the user to specify limits on any type of computing resource such as number of processors, disk space, memory, available licenses, etc.

Example

A user has 10,000 processors available and wants to execute an optimization run using MPP simulations requiring 128 CPUs per job. She therefore specifies the units per job as 128 and the global limit as 10,000. For this same optimization run, the user has 5,000Gb disk space available while using 40 Gb of disk space per job (which is deleted after the completion of each job). A second resource therefore has to be specified with attribute values 40 units per job and a global limit of 5,000. The resource setup is shown in Figure 5-17. The job scheduler will launch jobs that will not exceed any of these two limits.



Resources

Resource	Units per job	Global limit	Delete
DISK_SPACE	40	5000	×
CPU	128	10000	×

Figure 5-17: Definition of Resources for a Stage

Resources must be defined at the Stage level, but can be viewed in the Resource tab of the Setup dialog (see Section 8.4). The limits can be changed in either the Stage or Setup dialogs.

Stages can share resources. For instance, as part of an MDO problem, the same resource can be defined for multiple stages.

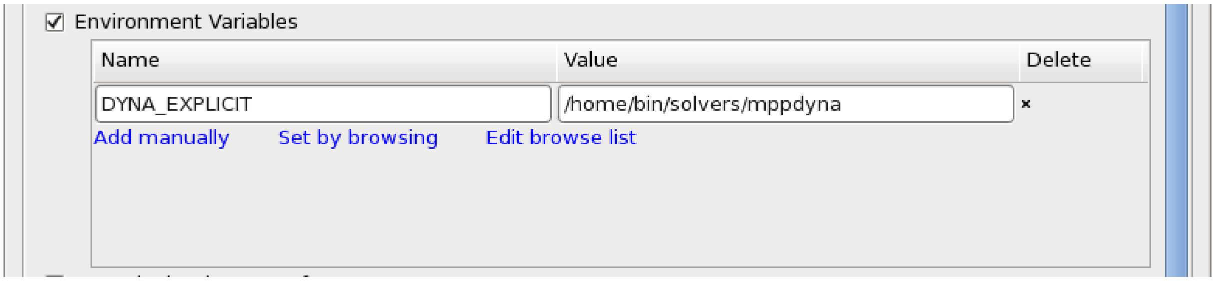
When using multiple computer clusters, independent resources are typically defined for each cluster. Jobs will then be run concurrently on all clusters within the limits defined for each cluster.

A single resource with a default of 1 Units per job and a Global limit of 1 is assumed for each stage at the beginning of the creation process. The default name is the solver type name. That also implies that if multiple stages use the same solver type, there will by default be only one resource definition. Resources can then be added or deleted as desired. To change a resource name, a new resource has to be added and the old resource deleted.

Remark

A resource definition related to e.g. the number of processors to be used for a simulation run does not replace the specification of the number of processors as a command line option or in the command script. The resource definitions are only used to calculate the number of jobs that are submitted concurrently.

5.4.2. Environment Variables



☑ Environment Variables

Name	Value	Delete
DYNA_EXPLICIT	/home/bin/solvers/mppdyna	×

Figure 5-18: Definition of Environment Variables

LS-OPT provides a way to define environment variables that will be set before executing a solver command. The desired environment variable settings can be specified in the **Stage** dialog if the *Environment Variables* checkbox is selected.

Passing environment variables to stage commands can be a convenient way to control the behavior of a command. For example, the command might be a script which queues a job on a remote machine; the environment variable settings might be used by the script to select various queuing options. Or, the environment variable settings might be passed along through the queuing system to set options for the remotely executed job, such as license server locations, input file names, whether to run the MPP version of LS-DYNA, whether to run a single or double precision solver, etc.

Select the button **Add manually** to define a single environment variable. After selecting this option, a new line will appear in the Environment Variables list where you can enter the variable name and an arbitrary value. We do not allow the names of variables to contain anything other than upper- or lower-case letters, numbers, and underscore (_) characters. This guarantees that all environment variable definitions can be used on all platforms. Variable values are not so limited.

The **Set by Browsing** option is used to set variables in bulk. This is done by running a user-supplied program or importing a user-supplied file (see Appendix H: Installing LS-OPT for further information). Activate the **Set by browsing** button in order to select from the available executables or files. A selection list containing all available files and programs will show up.

Selecting a file or executable will directly import all the specified variables into the Environment Variables list in bulk. In addition to these **Browse List** variables, a special browse variable is created that should not be edited. This variable records the program name used to create the Browse List.

NOTE: Strings in the Environment Variables list appearing above the *browse* line are all part of the Browse List. Strings that appear below *browse* are never part of the Browse List. User-defined environment variables will always follow after the browse variable definition.

Selecting the **Edit Browse list** button does nothing unless a Browse List has been previously created. If a valid Browse List is present in the Environment Variables list, then selecting this option will run the original program that created the Browse List, together with all of the current Browse List options passed as command line arguments, one per existing environment variable.

Executing the 'Edit Browse List' will cause the original file to be reread, which is convenient for testing purposes.

Note: The browse command can ABORT the replacement operation by printing a blank line to the standard output and immediately terminating. Otherwise the current Browse List may be deleted. If the browse command abnormally terminates, then an error box will appear with a title bar indicating that the command failed.

How the browse list is used by LS-OPT

The **Browse List** (indeed, the complete **Environment Variables List**) is used to set environment variables before running the solver command specified by LS-OPT. However, if the first variable returned by the browse command is **exe**, then a pre-processing command is run before running the actual solver command. The pre-processing command is the value of the **exe** variable. The pre-processing command has a command line

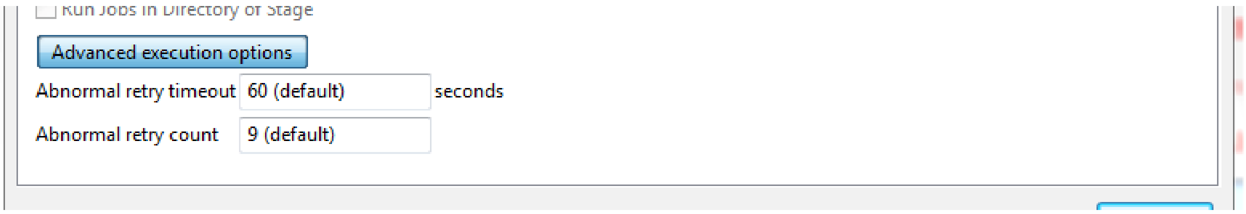
```
$exe var1=$var1, var2=$var2, ... varN=$varN
```

That is, the command executed is the value of the **exe** variable; additional command line arguments consist of all **Browse List** strings with a comma delimiter appended to each intermediate one. (The final argument is not followed by a comma.)

Note: Such a pre-processing command is always run from within the current **LS-OPT Job Directory**. Therefore, any file that the pre-processing command references must be specified by a fully-qualified path or must be interpreted relative to the current **LS-OPT Job Directory**. So, the **LS-OPT Stage Directory** will be **".."** and the **LS-OPT Project Directory** will be **"../.."**.

5.4.3. Advanced Execution Options

It may be prudent to retry job submissions for certain types of abnormal termination. For this purpose, the user can specify an Abnormal signal for terminations which are neither normal nor error termination. A job that has terminated in this way can then be retried by the LS-OPT job scheduler. The related options are described in Table 5-5. More information on abnormal termination is available in Appendix H.8 .


□ Run Jobs in Directory of Stage

Advanced execution options

Abnormal retry timeout: 60 (default) seconds

Abnormal retry count: 9 (default)

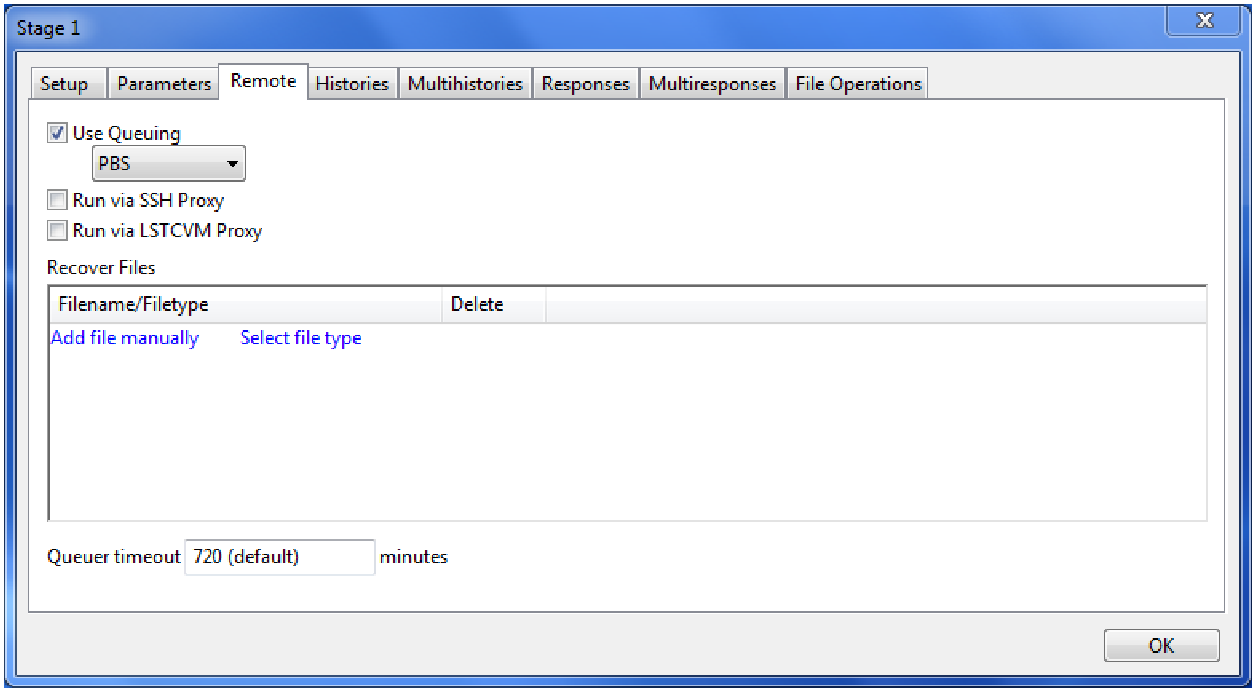
Figure 5-19: Advanced execution options

Table 5-5: Advanced execution options

Option	**Description**	**Reference**
Abnormal retry timeout	Submission script timeout (seconds)	Appendix H.8.1
Abnormal retry count	Number of retries if submission fails	Appendix H.8.1

5.5. Remote

The solver jobs do not have to be executed on the same machine as where LS-OPT is running. Settings for remote job scheduling can be defined in the **Remote** tab of the **Stage** dialog, Figure 5-20 and Table 5-6.



☑ Use Queuing

PBS

□ Run via SSH Proxy

□ Run via LSTCVM Proxy

Recover Files

Filename/Filetype	Delete
Add file manually Select file type	

Queuer timeout 720 (default) minutes

Figure 5-20: Stage dialog Remote options

Table 5-6: Stage dialog Remote options

Option	**Description**	**Reference**
Use Queuing	Interfacing with load sharing facilities to enable running simulation jobs across a network.	Section 5.5.1
Run via SSH Proxy	Run the job on a remote node using SSH.	Section 5.5.2
Run via LSTCVM Proxy	Enabling LSTCVM, Secure Proxy Server, for distributing solver jobs across a computer cluster.	Section 5.5.3
Recover Files	List of files to be recovered from remote machine, only available if a queuing system interface is used	Section 5.5.4
Queuer timeout	Time LS-OPT will wait for the wrapper to connect, otherwise it sets an abnormal termination status.	Section 5.4.3 Appendix H.8.2

5.5.1. Interfaces to Queuing Systems

The LS-OPT Queuing Interface interfaces with load sharing facilities (e.g. LSF[9] or LoadLeveler[10]) to enable running simulation jobs across a network. LS-OPT will automatically copy the simulation input files to each remote node, extract the results on the remote directory and transfer the extracted results to the local directory. The interface allows the progress of each simulation run to be monitored via LS-OPT*ui*. See Appendix H.5 for information on how to setup the interface.

Table 5-7: Queuing options

Option	**Description**	**Reference**
AQS	AQS	
LoadLeveler	LoadLeveler	
LSF	LSF	
NQE	NQE[11]	
PBS/TORQUE	PBS[12] and TORQUE	
PBSPRO	PBS PRO	
SLURM	SLURM	
SGE/UGE	SGE/Univa Grid Engine	
Black-Box	Black Box	Appendix **H.7**
Simple	Simple Queuing System	Appendix **H.7**
User-Defined	User Defined	Appendix **H.7**

5.5.2. Running via the SSH proxy server

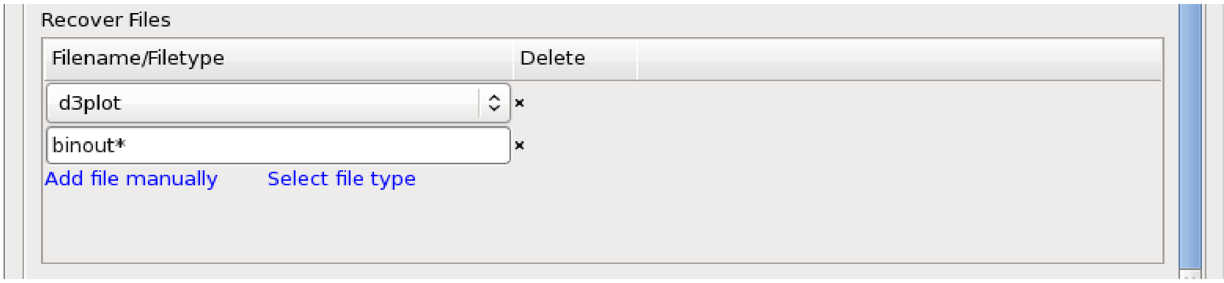
Selecting this option enabled the use of running the job on a remote node. This option can be used together with queuing options to for instance start jobs from a Windows machine to a Linux only cluster. See Appendix H.9.1 for more details.

[9] Registered Trademark of Platform Computing Inc.
[10] Registered Trademark of International Business Machines Corporation
[11] Network Queuing Environment. Registered Trademark of Cray Inc.
[12] Portable Batch System. Registered Trademark of Veridian Systems

5.5.3. Running via the LSTCVM secure proxy server

Selecting this option enables the interface to use LSTCVM. LSTCVM is a Secure Proxy Server for distributing solver jobs across a computer cluster, e.g. for running LS-OPT on a Windows machine controlling solver jobs on a Linux cluster. See Appendix H.10 for information on the installation of LSTCVM.

5.5.4. Recovering Output Files



Recover Files

Filename/Filetype	Delete
d3plot	×
binout*	×

Figure 5-21: Database recovery options

This option is only available if a queuing system interface is used, Section **Error! Reference source not found.**. When distributing the simulation runs, the information needed by LS-OPT is automatically extracted and transferred to the local node in the form of files `response.`*n* and/or `history.`*n*.

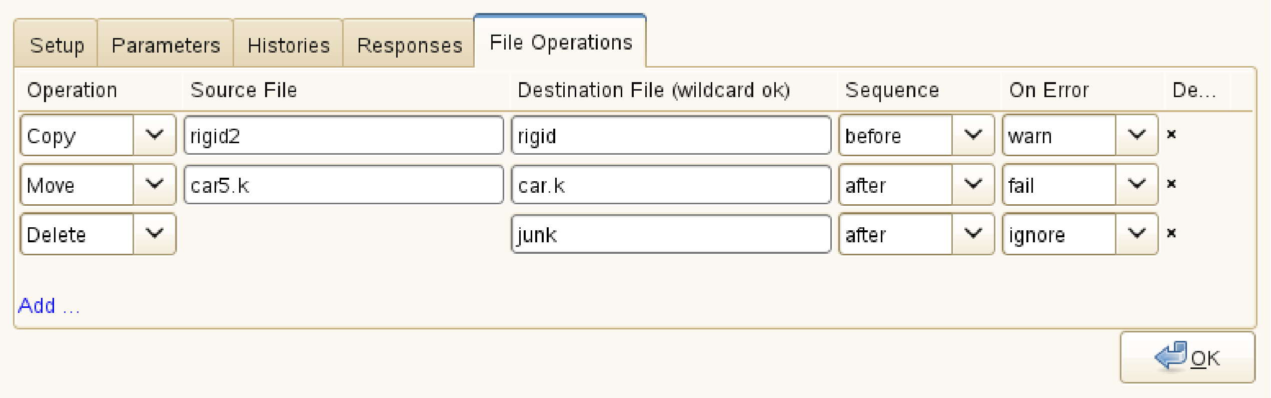
If the user wants to recover additional data to the local machine to do local post-processing (e.g. using LS-PREPOST), the *Recover Files* options can be used.

For LS-DYNA, the *Select file type* option can be used to recover *d3plot*, *d3hsp*, *binout*, *d3eigv* or *eigout* files. Each name is a prefix, so that e.g. `d3plot01, d3plot02, …` will be recovered when specifying *d3plot*.

Any database can be recovered by using the *Add file manually* option. Each name is a wildcard.

The requested database files will appear in the local run directory. The details of the recovery procedure is logged and available in the `job_log` file in the run directory on the local machine. Job logs can be viewed by double-clicking on the Stage LED during or after running. See Section 15.3.

5.6. File Operations



Operation	Source File	Destination File (wildcard ok)	Sequence	On Error	De...
Copy	rigid2	rigid	before	warn	×
Move	car5.k	car.k	after	fail	×
Delete		junk	after	ignore	×

Figure 5-22: File Operations within a Stage run directory

LS-OPT allows file operations between Stages or within a Stage.

The requested Stage file operations are executed for all the run directories related to the Stage, e.g. `CRASH/1.1`, `CRASH/1.2`, etc. Within a Stage run directory, several file operations can be executed on files previously copied to the run directories or generated by the stage command before or after executing the stage command. See Figure 5-22 and Table 3-4.

File operations between stages are discussed in Section 3.2.4.

Table 5-8: File Operations

Option	Selections	Description
Operation	Copy Move Delete	Available operations
Source File		Name of source file
Destination File		Name of destination file, wildcards are supported
Sequence	before after	Execute operation before or after executing the stage command
On Error	fail warn ignore	What to do if operation fails

5.7. The '`N o r m a l`' termination status

LS-OPT can only detect the solver termination status by reading the information that the solver prints to the screen (also called standard output or `stdout`). The LS-DYNA solver type automatically outputs the phrase '`N o r m a l`' which LS-OPT detects as a normal termination. For all other interfaces the starting command generated by LS-OPT handles the '`N o r m a l`' output. If '`N o r m a l`' is absent, LS-OPT assumes an error termination status and will not attempt to extract any results from the database. For user-defined stages, the user has the responsibility to write the status to standard output. This can be accomplished by inserting the solver command into a script or program in which the '`N o r m a l`' string is written at the end using a print statement. See also Appendix H.8 .

5.8. Managing disk space during run time

As multiple result output sets are generated during a parallel run, the user must be careful not to generate unnecessary output. The following rules should be considered:

- To save space, only those output files that are absolutely necessary should be requested.
- A significant amount of disk space can be saved by judiciously specifying the time interval between outputs (DT) e.g., in many cases, only the output at the final event time may be required. In this case the value of DT can be set slightly smaller than the termination time.
- For LS-DYNA stages the *DATABASE_EXTENT_BINARY option allows control over the size of the d3plot databases.
- The result extraction is done immediately after completion of each simulation run. Database files can be deleted immediately after extraction using the '*Delete*' file operation after the solver run (see Section 5.5).
- Database files can also be deleted by using the `clean` file (see Section 5.8.1).
- If the simulation runs are executed on remote nodes, the responses of each simulation are extracted on the remote node and transferred to the local run directory.

5.8.1. Using the clean file to delete solver output files

During a sequential approximation procedure, superfluous data can be erased after each run while keeping all the necessary data and status files (see Appendix E.3.4). For this purpose the user can provide a file named `clean` (`clean.bat` on Windows) containing the required erase statements such as

```
rm -rf d3*
rm -rf elout
rm -rf nodout
rm -rf rcforc
```

on Linux or

```
del d3*
del elout
del nodout
del rcforc
```

on Windows, respectively.

The `clean` file will be executed immediately after each simulation and will clean all the run directories except the baseline (first or 1.1) and the optimum (last) runs. Care should be taken not to delete the lowest level directories or the log files `started`, `finished`, `response.`*n* or `history.`*n* (which must remain in the lowest level directories). These directories and log files indicate different levels of completion status which are essential for effective restarting. Each file `response.`*response_number* contains the extracted value for the response: *response_number*. The essential data is thus preserved even if all solver data files are deleted. The *response_number* starts from 0.

Complete histories are similarly kept in `history.`*history_number*.

The minimal list to ensure proper restarting is:

```
XPoint
started
finished
response.0
response.1
.
.
history.0
history.1
.
.
```

Remarks

1. The `clean` file must be created in the work directory.
2. If the `clean` file is absent, all data will be kept for all the iterations.
3. For remote simulations, the `clean` file will be executed on the remote machine.

5.9. Alternative setups for running pre-processors

The easiest way of running a pre-processor is to define a separate stage for the pre-processor and solver and to make the solver stage dependent on the pre-processor stage. Because the output file of the pre-processor has to be used as input by the solver, the setup is important. There are at least three ways of setting up a pre-processor run:

1. Specify the output file of the pre-processor as an include file of the solver.
2. Copy the output file to the base file of the solver. E.g. if `lsppout` is the output file name of the pre-processor, copy `lsppout` to `DynaOpt.inp` which is the standard base file name for the LS-DYNA solver type. An inter- or intra-stage file operation is used for this purpose.
3. Rename the base file name of the solver to the output file name of the pre-processor (see Section 5.2.1). E.g. if the output file name of the pre-processor is `lsppout` rename the basefile of the solver (in this case the LS-DYNA type) from `DynaOpt.inp` to `lsppout`. LS-DYNA will then use `i=lsppout` as part of the solver command.

It should be noted that both the pre-processor and the solver can be run in the same directory by selecting the 'Run Job in Directory of Stage' option in the Setup tab of the Stage dialog. They can both be run in the directory of the pre-processor *or* the solver.

If they are both run in the **pre-processor** directory, a copy file operation (Section 5.5) should be specified in the 'File Operations' tab to copy the file *after* the **pre-processor** stage.

If they are both run in the **solver** directory, a copy file operation should be specified in the 'File Operations' tab to copy the file *before* the **solver** stage (Section 5.5).

If they are run in different directories (i.e. their own home directories), an inter-stage copy operation should be specified (Section 3.2.2).

6. History and Response Results

This chapter describes the specification of the history, multi-point history, response or multi-point response results to be extracted from the stage database. A history is a vector representing curve data, whereas a response is a scalar value. A multi-point history or response, respectively, additionally considers the spatial dimension and hence consists of multiple histories or responses. Responses can be used to define objectives or constraints (Chapter 12). Histories and multi-point histories are intermediate entities that can be used to calculate responses or composites (Chapter 10). Interfaces for result extraction from LS-DYNA output files are available, as well as mathematical expressions, file import, an interface for extracting values from ASCII databases, an Excel interface, and a user-defined interface where any program may be used for result extraction. The response dialogs are accessible from the **Stage** dialog: **Histories, Multihistories** and **Responses** tab, respectively.

6.1. Defining histories, multihistories and responses

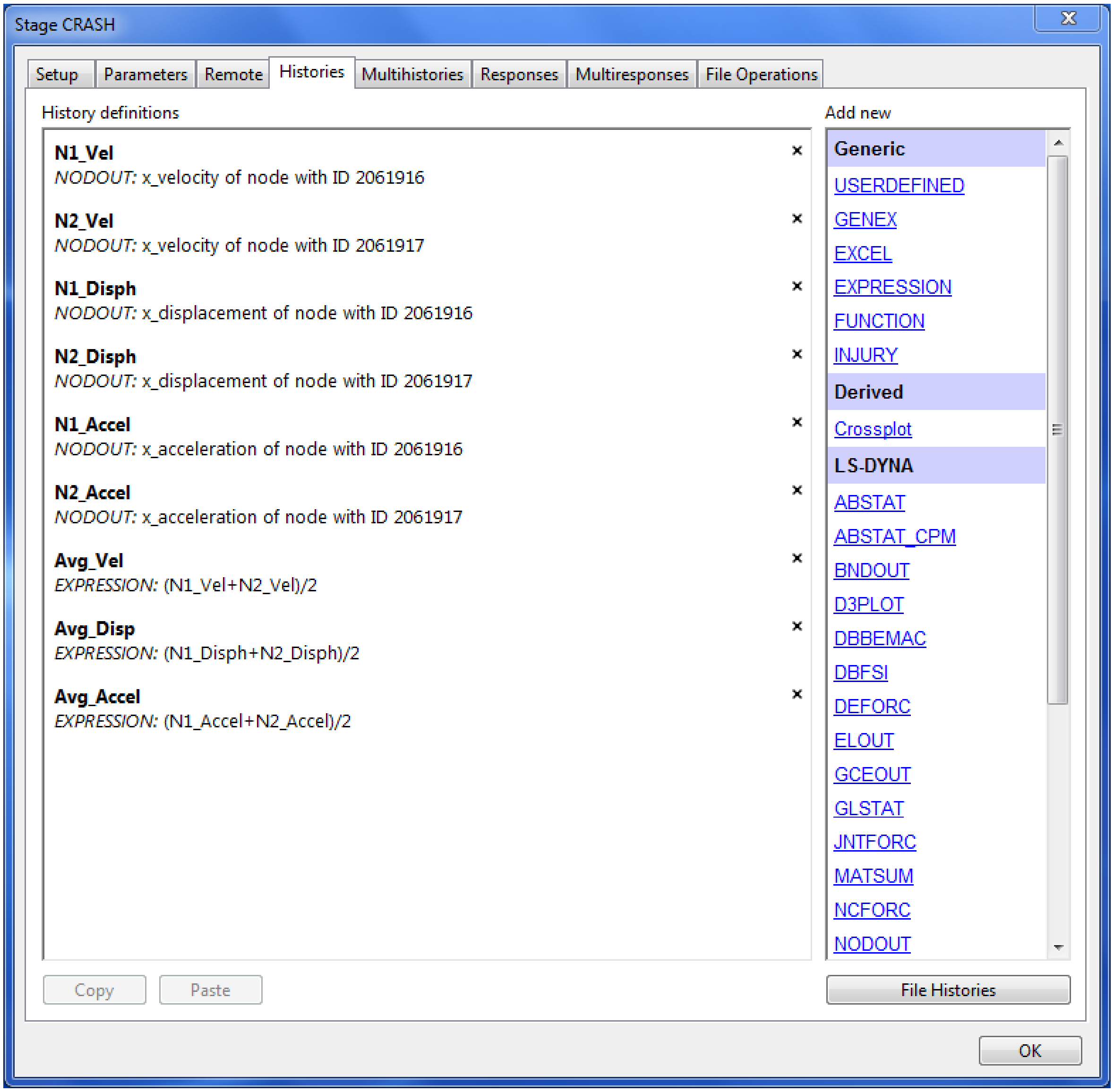
A history, multihistory, response or multiresponse can be defined by using the interfaces in the **Histories, Multihistories, Responses** and **Multiresponses** tab of the **Stage** dialog, respectively, Figure 6-1. To add a new definition, select the respective interface from the list on the right. The available interfaces are explained in Table 6-1. To edit an already defined (multi)history or (multi)response, double-click on the respective entry from the list on the left. Histories and responses may be deleted using the *delete* icon on the right of the respective definition.

There are five types of interfaces:

- Standard LS-DYNA or LS-OPT result interfaces. These interfaces provide access to the LS-DYNA binary databases (d3plot or binout, d3hsp or d3eigv) and the LS-OPT database, respectively. The interfaces are an integral part of LS-OPT.
- User specified interface programs. These can reside anywhere. The user specifies the full path.
- Mathematical expressions.
- GenEx. This interface allows the user to extract selected field values from a text file.
- Excel.

The extraction of responses consists of a definition for each response and a single extraction command or mathematical expression. A response is often the result of a mathematical operation of a response history, but can be extracted directly using the standard LS-DYNA interface (see Section 6.1.1) or a user-defined interface.

Each extracted (multi)response or (multi)history is identified by a name, Table 6-2, and the settings to be specified using the respective interface.



N1_Vel
NODOUT: x_velocity of node with ID 2061916

N2_Vel
NODOUT: x_velocity of node with ID 2061917

N1_Disph
NODOUT: x_displacement of node with ID 2061916

N2_Disph
NODOUT: x_displacement of node with ID 2061917

N1_Accel
NODOUT: x_acceleration of node with ID 2061916

N2_Accel
NODOUT: x_acceleration of node with ID 2061917

Avg_Vel
EXPRESSION: (N1_Vel+N2_Vel)/2

Avg_Disp
EXPRESSION: (N1_Disph+N2_Disph)/2

Avg_Accel
EXPRESSION: (N1_Accel+N2_Accel)/2

Figure 6-1: Histories definition in the GUI

Table 6-1: Interfaces for Response and History extraction

Option		**Description**	**Reference**
Generic	USERDEFINED	Result extraction using any script or program	Section 6.13
	FILE	Result extraction from a text file (Response only)	Section 6.14
	GENEX	Tool for extracting results from text files	Chapter 7

	EXCEL	Result extraction from an Excel document	Section 6.16
	EXPRESSION	Definition of mathematical expressions using previously defined entities	Section 6.4.1
	FUNCTION	Expressions using previously defined histories	Section 6.4.3
	INJURY	Injury criteria	Section 6.5
	Curve Matching	Metrics for comparison of two multi-point curves (Response only)	Section 0
	MATRIX_EXPRESSION	(Response only)	Section 0
Derived	Crossplot	Crossplot (History only)	Section 6.4.1
LS-DYNA	ABSTAT	Binout interface	Section 6.2.1
	ABSTAT_CPM	Binout interface	Section 6.2.1
	BNDOUT	Binout interface	Section 6.2.1
	D3PLOT	D3plot interface	Section 6.2.3
	DBBEMAC	Binout interface	Section 6.2.1
	DBFSI	Binout interface	Section 6.2.1
	DEFORC	Binout interface	Section 6.2.1
	ELOUT	Binout interface	Section 6.2.1
	FLD	Metal Forming results (Response only)	Section 6.3.2
	FREQUENCY	D3eigv interface (Response only)	Section 6.2.5
	GCEOUT	Binout interface	Section 6.2.1
	GLSTAT	Binout interface	Section 6.2.1
	JNTFORC	Binout interface	Section 6.2.1
	MASS	D3hsp interface (Response only)	Section 6.2.4
	MATSUM	Binout interface	Section 6.2.1

	NCFORC	Binout interface	Section 6.2.1
	NODOUT	Binout interface	Section 6.2.1,
	NODFOR	Binout interface	Section 6.2.1
	PSTRESS	Metal Forming results (Response only)	Section 6.3.3
	RBDOUT	Binout interface	Section 6.2.1
	RCFORC	Binout interface	Section 6.2.1
	RWFORC	Binout interface	Section 6.2.1
	SBTOUT	Binout interface	Section 6.2.1
	SECFORC	Binout interface	Section 6.2.1
	SPCFORC	Binout interface	Section 6.2.1
	SPHOUT	Binout interface	Section 6.2.1
	SWFORC	Binout interface	Section 6.2.1
	THICK	Metal Forming results (Response only)	Section 6.3.1
LS-OPT	LSOPT	Optimized inner level variables, responses, composites, objective functions, constraints, histories and reliability statistics	Section 6.15.2, Section 6.15.1
	LSOPT_STATISTICS	Statistical values produced by a Monte Carlo analysis (Response only)	Section 6.15.3
File Histories		Global file histories	Section 6.18
File Multihistories		Global file multihistories	Section 6.19
Copy		Copy the selected History/Response	
Paste		Paste a previously copied History/Response, also possible between stages. The next free number is automatically appended to the name.	

Table 6-2: General (Multi)History and Response options for all interfaces

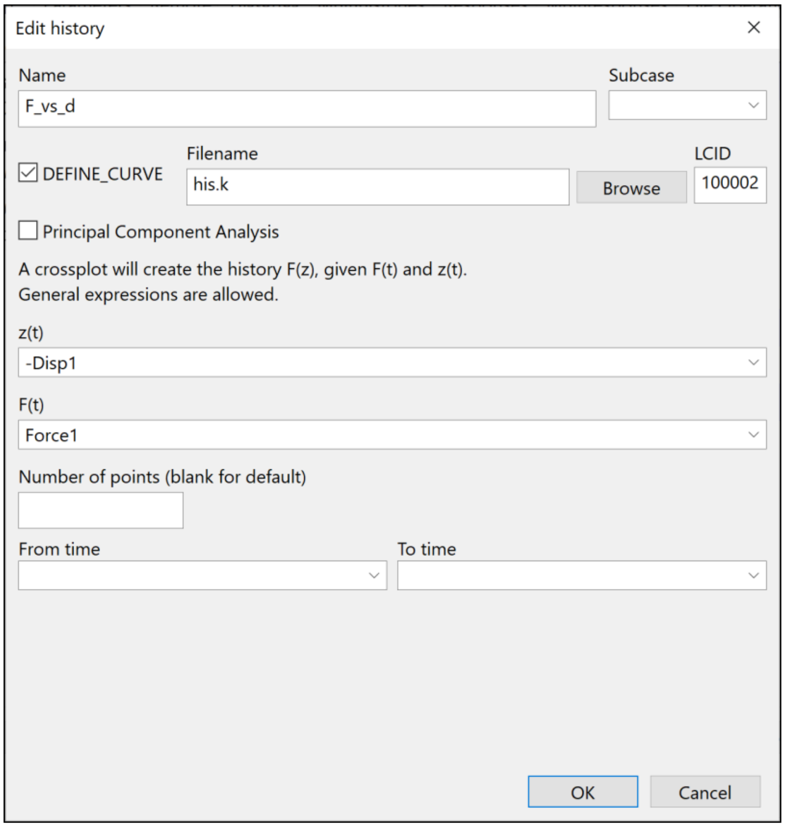
Option	Description
Name	History/Response name
Subcase	Integer CASE ID associated with the *CASE parameter in LS-DYNA. This option is mandatory for disciplines that use the *CASE parameter in LS-DYNA input files but is not required for other cases. For all other cases, first/last commands should be used.
Multiplier Offset	(Response only) If scaling and/or offsetting of the response is required, the final response is computed as (the extracted response × Multiplier) + Offset.
Not metamodel linked	(Response only) Sometimes it is beneficial to create intermediate responses without associated metamodels, although the task is metamodel-based. This promotes efficiency. Responses that are not metamodel linked cannot be included directly in composites, as composites rely on metamodel-based calculations.
Dump formula file	(Response only) Dump metamodel formula to file formula_dump_*responsename.iteration* in the working directory.
DEFINE_CURVE	(History only) *DEFINE_CURVE definition of history, Section 6.1.2
Principal Component Analysis	(History and Multiresponse only) Do Principal Component Analysis, Section 6.1.3

6.1.1. Result extraction

Each individual simulation run is immediately followed by a result extraction to create the `history.`*n*, `multihistory.`*n*, `response.`*n* and `multiresponse.`*n* files for that particular design point. For distributed simulation runs, this extraction process is executed on the remote machine. The `history.`*n*, `multihistory.`*n*, `response.`*n* and `multiresponse.`*n* files are subsequently transferred to the local run directory. If the extraction on the remote machine is not successful, and the solver database is locally available, it is repeated on the local machine. Hence programs and scripts needed for result extraction do not have to be accessible from the remote machine. These results are stored in the `AnalysisResults_n.csv`, `AnalysisResults_`*n*`.lsda` and `AnalysisResults_`*n*`.lsox` databases.

6.1.2. Creating a history file with an LS-DYNA *DEFINE_CURVE keyword

The DEFINE_CURVE selection allows the creation of an LS-DYNA include file (e.g. his.k) with the *DEFINE_CURVE keyword and history data. The LCID, which represents the load curve ID required by LS-DYNA, should be entered in the appropriate text box. See e.g. Figure 6-15.



Edit history

Name: F_vs_d

Subcase:

☑ DEFINE_CURVE

Filename: his.k

LCID: 100002

□ Principal Component Analysis

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t): -Disp1

F(t): Force1

Number of points (blank for default):

From time:

To time:

Figure 6-2: DEFINE_CURVE creates file `his.k` containing the `*DEFINE_CURVE` keyword with `LCID 100002` together with the generated history data.

6.1.3. Principal Component Analysis

The Principal Component Analysis (PCA) selection is available for histories and multi-responses as soon as a meta-model task is selected. It performs a Principal Component Analysis for the selected entity at every iteration (see Section 16.3.4).

The Principal Component Analysis is a multivariate statistical technique converting correlated observations into a set of linearly uncorrelated components called Principal Components. The Principal Components are sorted in decreasing order according to their variance, so that the first Principal Component has the largest variability in the data set, and so on (See Section 23.5).

In LS-OPT, PCA is used as a sensitivity analysis tool to rank variables according to their influence on the Principal Components.

The advantage of this analysis tool is that you can visualize the influence of parameters on histories and multi-responses.

6.2. Extracting history and response quantities: LS-DYNA

LS-OPT provides interfaces for history and response result extraction from binout, d3plot, d3hsp and d3eigv. Multihistories and multiresponses are extracted from the d3plot database. The user must ensure that the LS-

DYNA program will provide the output files required by LS-OPT, however, an option to check for missing *DATABASE cards is enabled by default, see Section 5.3.1.

The options for the extraction of LS-DYNA responses and histories are identical, except for the *Select* option.

Aside of the standard interfaces for extracting any data item from the LS-DYNA database, specialized responses for metal-forming are also available. The computation and extraction of these responses are discussed in Section 6.3.

6.2.1. LS-DYNA binout results

All LS-DYNA history and response result extraction options except for D3PLOT, MASS and FREQUENCY interface with the LS-DYNA binout output. The BINARY flag in the respective *DATABASE_OPTION card and the desired entity ID in the *DATABASE_HISTORY_OPTION card has to be set correctly in the LS-DYNA input file. Note that the LS-DYNA executable is interpreted as a single process (SMP) by LS-OPT, hence the default binary flag value 0 is not supported, Section 5.3.1.

Results can be extracted for the whole model or a finite element entity such as a node or element, depending on the selected interface. For shell and beam elements the through-thickness position can be specified as well.

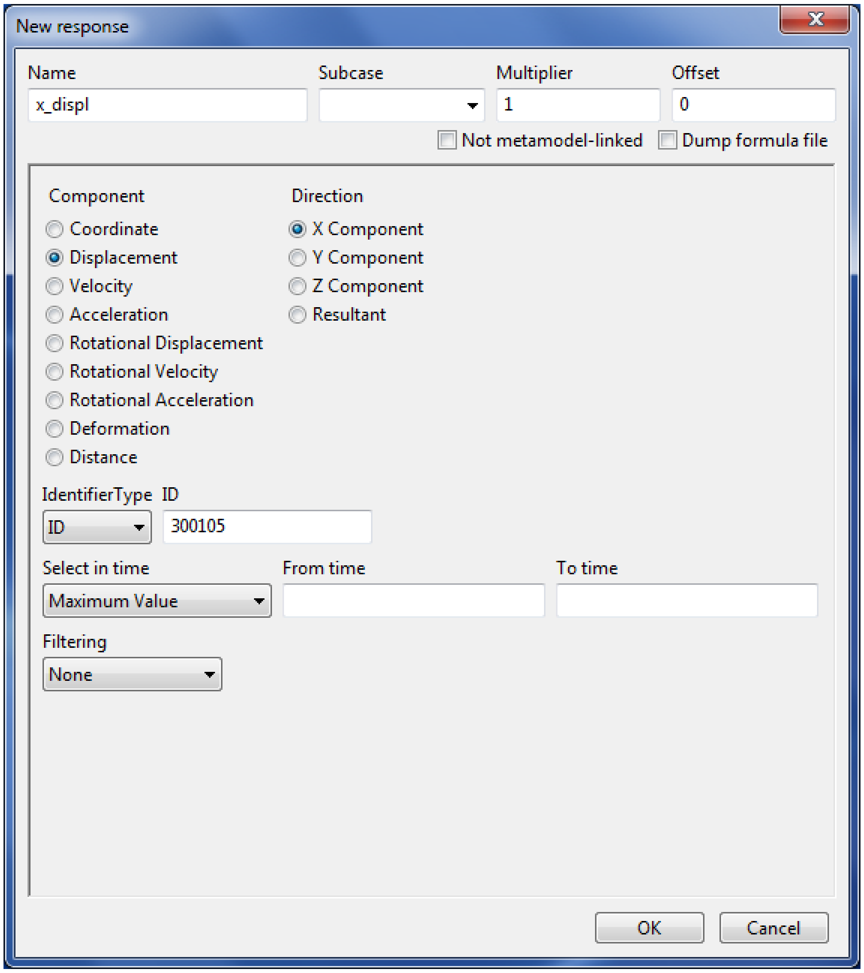
The response options are an extension of the history options – a history will be extracted as part of the response extraction. The *Select* option has to be specified to extract a scalar value from the curve. The optional attributes *From time* and *To time* can be specified to slice the curve before extracting the requested scalar value. The defaults are the begin and the end value of the history.

Filtering and averaging options are available for histories and responses.

These operations will be applied in the following order: averaging or filtering, and slicing.

The available results types and components are listed in Appendix A: LS-DYNA Binout Commands and Appendix B: LS-DYNA Binout Components.

The NODOUT components *Deformation* and *Distance* are described in detail in Section 6.2.2.



New response

Name: x_displ

Subcase:

Multiplier: 1

Offset: 0

□ Not metamodel-linked □ Dump formula file

Component
○ Coordinate
● Displacement
○ Velocity
○ Acceleration
○ Rotational Displacement
○ Rotational Velocity
○ Rotational Acceleration
○ Deformation
○ Distance

Direction
● X Component
○ Y Component
○ Z Component
○ Resultant

IdentifierType: ID

ID: 300105

Select in time: Maximum Value

From time:

To time:

Filtering: None

Figure 6-3: Response extraction: LS-DYNA NODOUT interface

6.2.2. Kinematics

Additional kinematics such as *distance*s and *deformations* can be computed directly using NODOUT results by defining two nodes on the finite element mesh. Kinematics consist of two main quantities:

- The distance vector $\boldsymbol{q}$ computed using the differences between the coordinates of the two nodes.
- The deformation derived using the difference between the distance vector computed at time t and the original distance vector (t = 0).

These quantities can be computed in

- the global coordinate system,
- a local coordinate system or
- local coordinates referred to the global reference frame (t = 0).

The local axes are computed using the convention defined in Section 6.4.5 to define the rotation matrix $\boldsymbol{A}$ where $\boldsymbol{A}$ is a function of time. The quantities are therefore defined as follows.

Table 6-3: Definitions of the kinematics of a displaced rigid body

Frame	**Distance**	**Deformation**
Global	$\boldsymbol{d} = \boldsymbol{q}$	$\boldsymbol{u} = \boldsymbol{q} - \boldsymbol{q}(0)$
Local	$\boldsymbol{d}' = \boldsymbol{A}(t)\boldsymbol{q}(t)$	$\boldsymbol{u}' = \boldsymbol{d}' - \boldsymbol{A}(0)\boldsymbol{q}(0)$
Local in reference	$\boldsymbol{d}'' = \boldsymbol{A}^T(0)\boldsymbol{A}(t)\boldsymbol{q}(t)$	$\boldsymbol{u}'' = \boldsymbol{d}'' - \boldsymbol{q}(0)$

The orthogonal matrix $\boldsymbol{A}(t)$ is defined by a local coordinate system (*x'y'z'* in Figure 6-4) which in turn is defined by three nodes on the finite element mesh as it displaces over time. Nodes **2** and **3** represent the local *x*-axis direction, Figure 6-4, while node **1** represents the third node. This is the same convention as defined in Section 6.4.5.

The second and third kinematic categories are both denoted “local” since deformation should be totally absent for pure rigid body systems. If the triangles 1-2-3 and 1′-2′-3′ are congruent, i.e. they represent a rigid body, the quantity defined as *Local in reference frame* is invariant with respect to the node numbering. E.g. the triplets (1, 2, 3), (2, 3, 1) or (1, 3, 2) should yield the same value.

To monitor congruence, A *Congruence ratio* for each history or response is displayed in the job_log (run directory) or lsopt_output files. The ratio for a node is defined as the ratio of the side length opposite the node *i* at time t_{final} divided by the same quantity applied to the undeformed structure, see equation below. Three values are therefore printed. The ideal ratio is unity, signifying a perfectly rigid body.

$$r_i = \frac{\left|\mathbf{x}_{i-1}(t) - \mathbf{x}_{i-2}(t)\right|}{\left|\mathbf{x}_{i-1}(0) - \mathbf{x}_{i-2}(0)\right|}, i = 1,2,3$$

Kinematic quantities are available as both histories and responses.

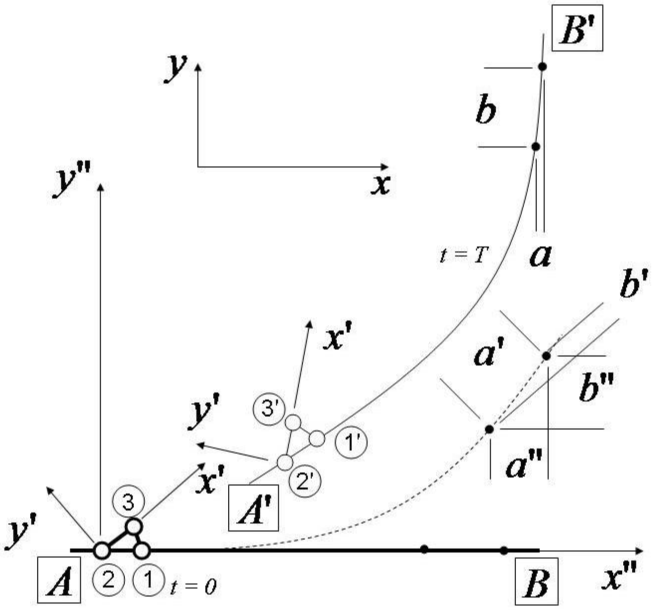


Figure 6-4: Local and global coordinate systems

6.2.3. LS-DYNA d3plot results

The D3PLOT interface is related to the Binout interface. The D3PLOT results differ from the Binout commands in that a response or history can be collected over a whole part. For example, the maximum stress can be evaluated in a part or over the whole model. Results can also be extracted for a finite element entity such as a node or element. For shell and beam elements the through-thickness position can be specified (currently not available in the GUI). Element results such as stresses are averaged in order to create nodal results.

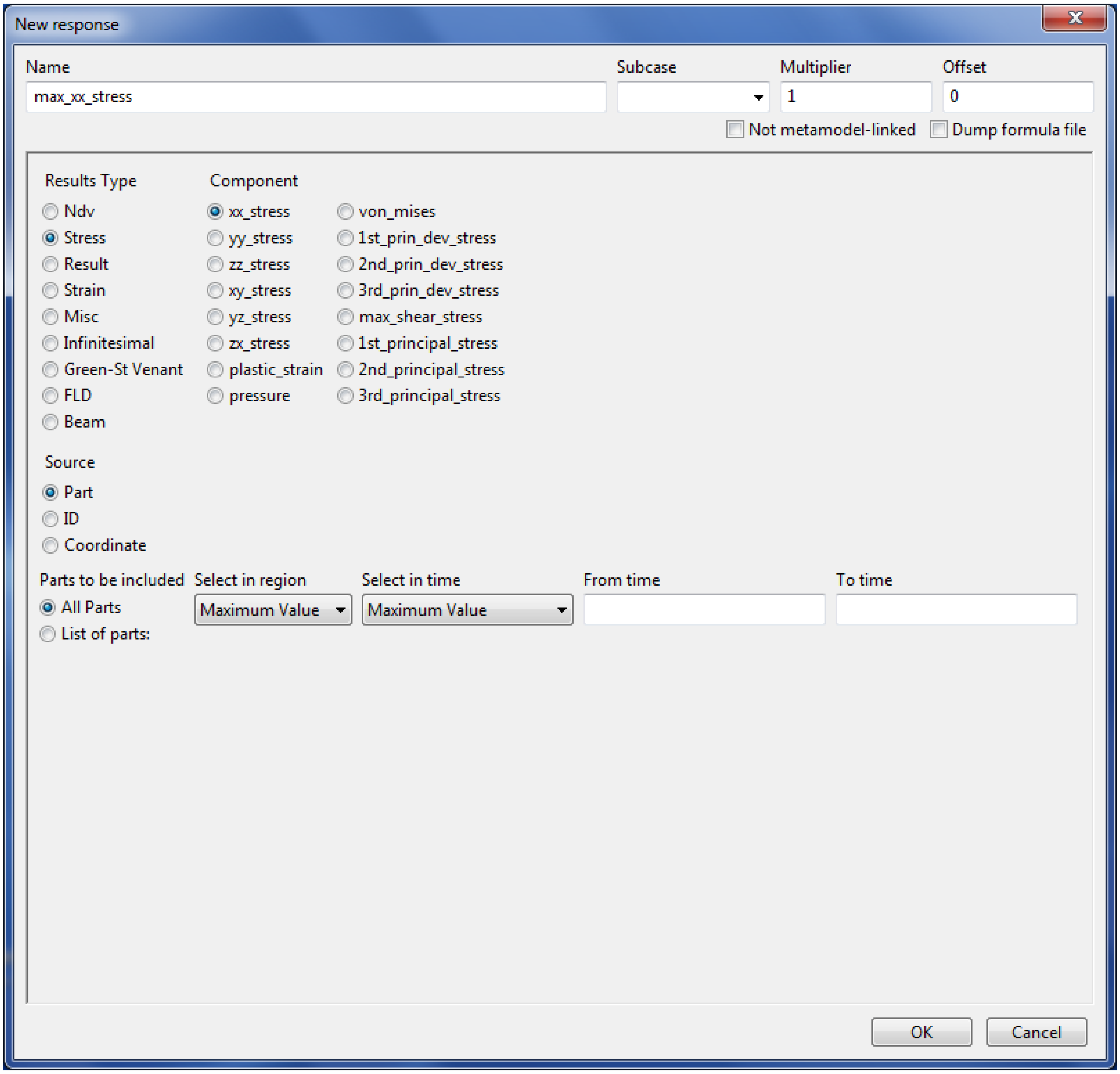
If the location of extraction is specified by *x,y,z* coordinates, the quantity will be extracted from the element nearest to *x,y,z* at the time of reference state. Only elements included in the `*SET_SOLID_GENERAL` element set are considered (only the `PART` and `ELEMENT` options).

The response options are an extension of the history options – a history will be extracted as part of the response extraction. For responses, the *Select in time* option has to be specified to extract a scalar value from the curve. The optional attributes *From time* and *To time* can be specified to slice the curve before extracting the requested scalar value. The defaults are the begin and the end value of the history.

If the selection must be done over parts as well, the maximum, minimum or average can be selected for the part (*Select in region*), followed by the selection of the maximum, minimum, or average over time.

The available results types and components are listed in Appendix C: LS-DYNA D3Plot Commands and Appendix D: LS-DYNA D3Plot Components.

The LS-PREPOST fringe plot capability can be used for the graphical exploration and troubleshooting of the data.



New response

Name: max_xx_stress

Subcase:

Multiplier: 1

Offset: 0

□ Not metamodel-linked □ Dump formula file

Results Type
○ Ndv
● Stress
○ Result
○ Strain
○ Misc
○ Infinitesimal
○ Green-St Venant
○ FLD
○ Beam

Component
● xx_stress
○ yy_stress
○ zz_stress
○ xy_stress
○ yz_stress
○ zx_stress
○ plastic_strain
○ pressure
○ von_mises
○ 1st_prin_dev_stress
○ 2nd_prin_dev_stress
○ 3rd_prin_dev_stress
○ max_shear_stress
○ 1st_principal_stress
○ 2nd_principal_stress
○ 3rd_principal_stress

Source
● Part
○ ID
○ Coordinate

Parts to be included
● All Parts
○ List of parts:

Select in region: Maximum Value

Select in time: Maximum Value

From time:

To time:

Figure 6-5: Response extraction from d3plot

D3Plot FLD results

If FLD results are requested then the FLD curve can be specified using (*i*) the *t* and *n* coefficients or (*ii*) a curve in the LS-DYNA input deck. The interpretation of the *t* and *n* coefficients is the same as in LS-PREPOST. Note that the *THICK*, *FLD* and *PSTRESS* interface options are an alternative, Section 6.3.

D3Plot Multihistories

Multihistories offer the possibility to extract histories at multiple points on the model, hence the spatial dimension is considered as well. The available options are described in Table 6-4.

Multihistories can be extracted for shells and solids. If the *Location* option is selected as *Coordinate File*, the coordinates of the extraction points must be specified in LS-DYNA format in a file using the ***NODE** keyword. The user should ensure that the IDs used are different from the IDs used in the LS-DYNA model and the data is terminated by **`*END`**.

Multihistories can also be interfaced with digital imaging systems. If the multihistory is defined for comparison to test results generated by the optical measurement system ARAMIS[13], the ARAMIS point locations can be used here (*Source* option *ARAMIS*). The respective ARAMIS File Multihistory, see Section 6.19, has to be selected from the ARAMIS multihistory menu. LS-OPT will automatically extract the coordinates of the points from the ARAMIS output defined by the File Multihistory.

Table 6-4: D3Plot Multihistory options

Option	**Description**
Source	ARAMIS Multihistory or Coordinate File. The file should be in the LS-DYNA *NODE keyword format to define extraction locations.
ARAMIS Multihistory (Source option *ARAMIS* only)	Name of File Multihistory which defines coordinates. Coordinates are extracted from ARAMIS output files.
File name (Source option *Coordinate File* only)	File in LS-DYNA format that contains the extraction point coordinates using the *NODE keyword format.
FE Interpolation (Element results only)	Nearest node: averaged element results Nearest element: exact coordinate, value interpolated using a bilinear interpolation
Cluster source points to nodes	Reduces the number of points to be similar to the number of nodes in the region of the source point set. The cluster represents those points that are closest to the FE nodes in the region of the source point set.
Distance Tolerance	Tolerance in distance units used to filtering source points. A point is discarded if the tolerance is exceeded.
Align points and simulation geometry	Transformation to align point set and simulation data
New alignment	Definition of transformation to align point set with simulation geometry
Open in LSPP	An LS-PrePost display is created showing the alignment selected for the point set and FE mesh.

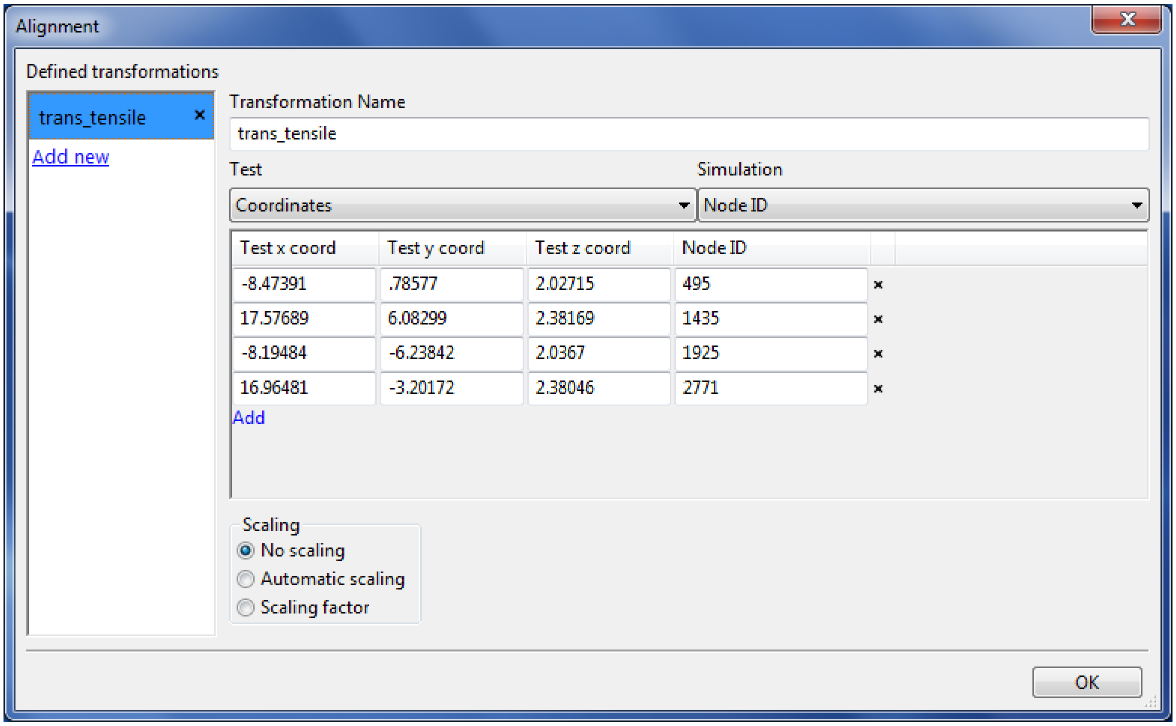
[13] GOM GmbH, Braunschweig

For element results such as stresses, the user can select whether to use the result of the nearest node or the nearest element. When selecting the element option, the quantity is extracted at the exact coordinate with the value interpolated within the element using a bilinear interpolation. In order to create the nodal results, element results will be averaged.

The extraction points can be filtered in two ways. The first is to select a cluster of points which are the nearest neighbors to the point set. The second is by specifying a tolerance on the distance between a point and a node. If no node is found within the distance tolerance, the point is discarded for extraction. The tolerance can be applied together with the cluster in which case the number of cluster points may be further reduced. Applying the cluster and/or tolerance improves the spatial matching of the point set and FE mesh, which in turn improves the accuracy and efficiency.

Since the position of test and simulation geometry might be different, a transformation can be defined to align test and simulation data in three dimensions. Select the **New Alignment** button to open the **Alignment** dialog, Figure 6-6.

At least three points need to be specified for test and simulation, respectively, to calculate the transformation. The test points can be defined by coordinates, or for ARAMIS the ARAMIS point IDs can be used. For the simulation, also coordinates can be used or nodal IDs. Additionally, if the dimensions of test and simulation geometry are not the same, a scaling factor can be specified to scale the test geometry to the simulation geometry. The alignment is done using a least squares distance.



trans_tensile ×

Transformation Name: trans_tensile

Test: Coordinates

Simulation: Node ID

Test x coord	Test y coord	Test z coord	Node ID	
-8.47391	.78577	2.02715	495	×
17.57689	6.08299	2.38169	1435	×
-8.19484	-6.23842	2.0367	1925	×
16.96481	-3.20172	2.38046	2771	×

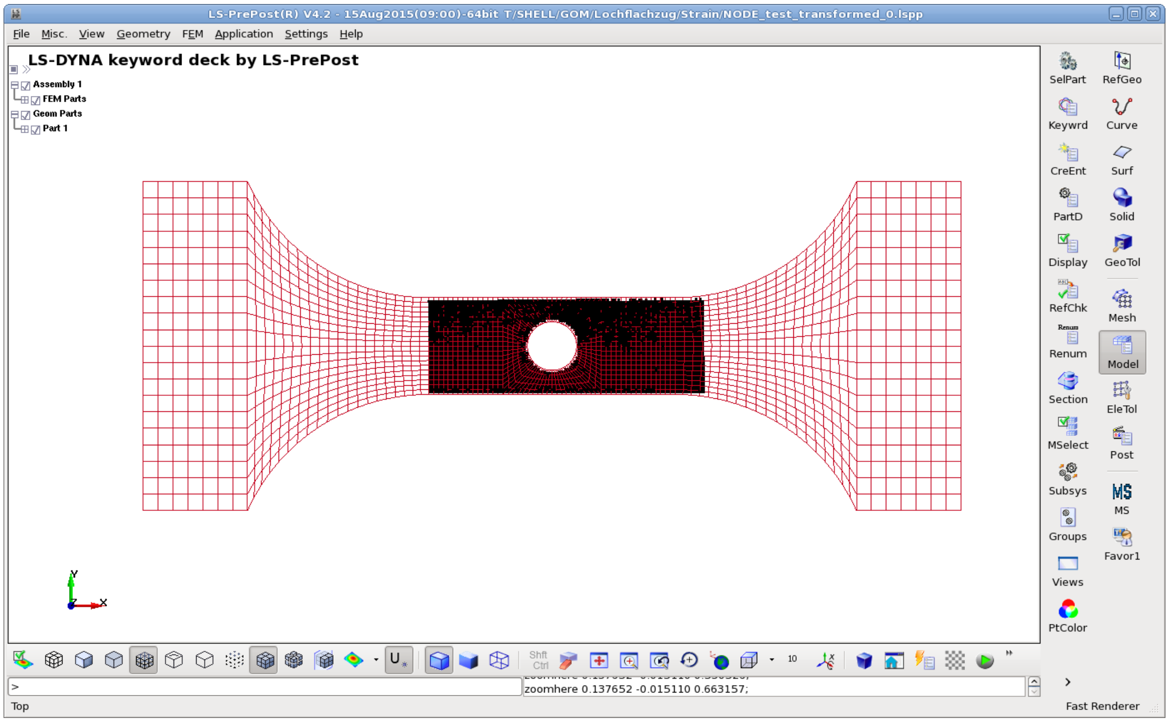
Scaling
- ● No scaling
- ○ Automatic scaling
- ○ Scaling factor

Figure 6-6: Alignment dialog

Table 6-5: Alignment options

Option	Description	
Transformation Name	Name of transformation	
Test	Format of alignment point specification for test data. The alignment points can be specified by coordinates, or by ARAMIS point ID.	
Simulation	Format of alignment point specification for simulation model. The alignment points can be specified by coordinates or by node ID.	
Scaling	No scaling	Only translation and rotation is calculated from alignment points
	Automatic scaling	Isotropic scaling is calculated from alignment points.
	Scaling factor	Use isotropic scale factor if dimensions of test and simulation geometry are different. The test coordinates are scaled.

Open in LSPP launches LS-PrePost with the LS-DYNA input file and the coordinate file that is specified for the multihistory locations, Figure 6-7. The alignment of test and simulation geometry is considered.

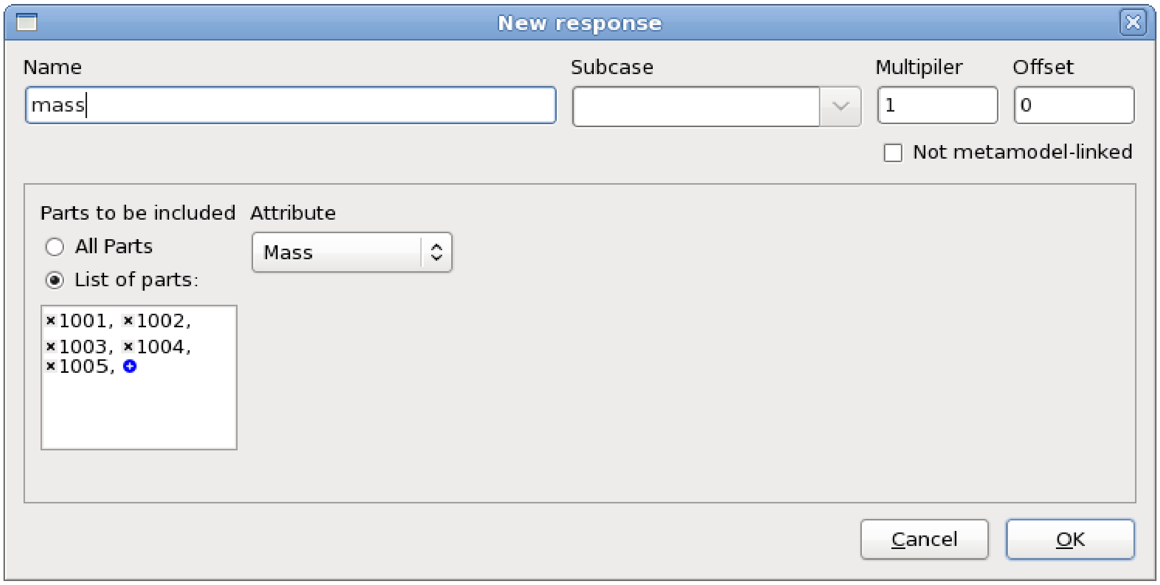

LS-PrePost(R) V4.2 - 15Aug2015(09:00)-64bit T/SHELL/GOM/Lochflachzug/Strain/NODE_test_transformed_0.lspp
File Misc. View Geometry FEM Application Settings Help
LS-DYNA keyword deck by LS-PrePost
Assembly 1
FEM Parts
Geom Parts
Part 1
SelPart
RefGeo
Keywrd
Curve
CreEnt
Surf
PartD
Solid
Display
GeoTol
RefChk
Mesh
Renum
Model
Section
EleTol
MSelect
Post
Subsys
MS
Groups
Favor1
Views
PtColor
Y
X
Top
Fast Renderer

zoomhere 0.137652 -0.015110 0.663157;

Figure 6-7: D3PLOT Multihistory: Test points superimposed on LS-DYNA model

6.2.4. Mass – Interfacing with d3hsp

The `MASS` response interfaces with the LS-DYNA output file **`d3hsp`**. The mass and related entities, Figure 6-8 and Table 6-6, can be extracted for the whole model or a list of parts.



New response

Name: mass

Subcase:

Multipiler: 1

Offset: 0

□ Not metamodel-linked

Parts to be included

○ All Parts

● List of parts:

×1001, ×1002, ×1003, ×1004, ×1005,

Attribute: Mass

Figure 6-8: Interface for extraction of Mass and related entities from LS-DYNA output d3hsp

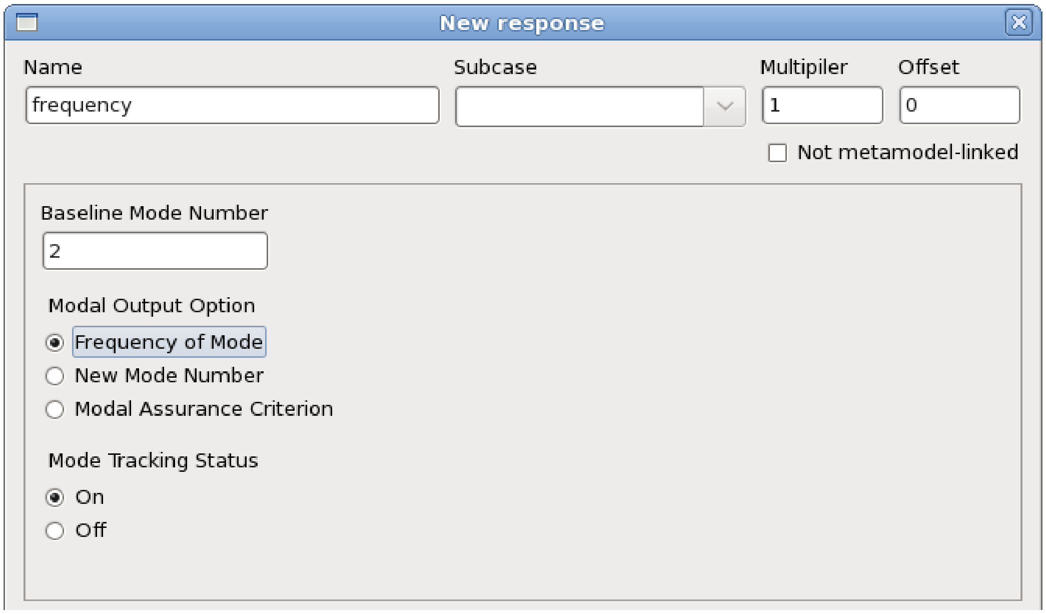
Values are summed if more than one part is specified (so only the mass value will be correct). However for the full model (part specification omitted) the correct values are given for all the quantities.

Table 6-6: Mass item description

Item	Description	
Parts to be included	Entity is extracted for the entire model or for the part IDs specified in the list.	
Attribute	Type of mass quantity:	
	Mass	Mass
	Principal Inertias	Component `I11, I22, I33`
	Inertia Tensor	Component `IXX, IXY, IXZ, IYX, IYY, IYZ, IZX, IZY, IZZ`
	Mass Center	Component *X*-Coordinate, *Y*-Coordinate or *Z*-Coordinate of mass center

6.2.5. Frequency – Interfacing with d3eigv

The `FREQUENCY` response interfaces with the LS-DYNA output file d3eigv, Figure 6-9. See Table 6-7 for a description of the available extraction options.



New response

Name: frequency

Subcase:

Multipiler: 1

Offset: 0

□ Not metamodel-linked

Baseline Mode Number: 2

Modal Output Option

● Frequency of Mode

○ New Mode Number

○ Modal Assurance Criterion

Mode Tracking Status

● On

○ Off

Figure 6-9: Interface for extraction of frequencies from LS-DYNA output d3eigv

Table 6-7: Frequency item description

Item	Description	
Baseline Mode Number	The number (sequence) of the baseline modal shape to be tracked. It cannot exceed 999. The user must identify which baseline mode is of interest by viewing the baseline `d3eigv` file in LS-PrePost.	
Modal Output Option	Type of modal quantity	
	Frequency of Mode	Frequency of current mode corresponding in modal shape to baseline mode specified.
	New Mode Number	Number of current mode corresponding in modal shape to baseline mode specified.
	Modal Assurance Criterion	Modal assurance criterion: $\max_j \frac{\{\varphi_0\}^H\{\varphi_j\}\{\varphi_j\}^H\{\varphi_0\}}{\{\varphi_0\}^H\{\varphi_0\}\{\varphi_j\}^H\{\varphi_j\}} = \max_j[MAC_j]$
Mode Tracking Status	Enable or disable mode tracking, see Theory section below	

Mode Tracking - Theory

Mode tracking is required during optimization using modal analyses as mode switching (a change in the sequence of modes) can occur as the optimizer modifies the design variables. In order to extract the frequency of a specified mode, LS-OPT calculates the modal assurance criterion (*MAC*). The scalar *MAC* value provides the degree of consistency between baseline modal shape and each mode shape of the current design. The maximum *MAC* value indicates the mode most similar in shape to the original mode selected. LS-OPT reads the eigenvectors from the d3eigv files, for calculating the *MAC* values. The *MAC* value for the reference modal vector φ_0 and the j^{th} modal vector of the current design φ_j is calculated as:

$$MAC_j = \frac{\{\varphi_0\}^H\{\varphi_j\}\{\varphi_j\}^H\{\varphi_0\}}{\{\varphi_0\}^H\{\varphi_0\}\{\varphi_j\}^H\{\varphi_j\}} \tag{6-1}$$

where *H* is the Hermitian operator. The MAC value corresponding to the most similar mode can be extracted using the respective Modal Output Option (see Table 6-7).

In certain cases, the user may be interested in the frequency corresponding to a specific mode number. To enable this option, the ability to turn mode tracking off is provide. By default this feature is on, but turning it off enables one to extract the responses corresponding to a specific mode number, irrespective of the mode shape.

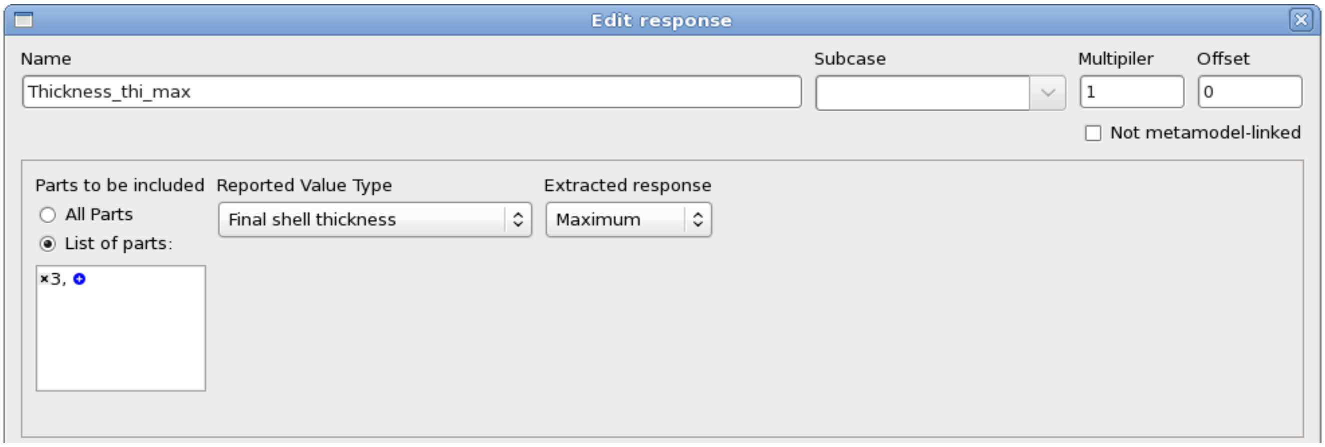
6.3. Extracting metal forming response quantities: LS-DYNA

Responses directly related to sheet-metal forming can be extracted, namely the final sheet thickness (or thickness reduction), Forming Limit criterion and principal stress. All the quantities can be specified on a part basis as defined in the input deck for LS-DYNA. Mesh adaptivity can be incorporated into the simulation run.

The user must ensure that the `d3plot` files are produced by the LS-DYNA simulation. Note that the *D3PLOT* interface options are an alternative.

6.3.1. Thickness and thickness reduction

Either thickness or thickness reduction can be specified using the `THICK` interface, Figure 6-10.



Edit response

Name: Thickness_thi_max

Subcase:

Multipiler: 1

Offset: 0

□ Not metamodel-linked

Parts to be included

○ All Parts

● List of parts:

×3,

Reported Value Type: Final shell thickness

Extracted response: Maximum

Figure 6-10: Thickness or Thickness reduction interface

Table 6-8: THICK options description

Item	**Description**
Parts to be included	Entity is extracted for the entire model or for the parts IDs specified in the list.
Reported Value Type	Final shell thickness Percentage thickness reduction
Extracted response	Minimum, maximum or average computed over all the elements of the selected parts

6.3.2. FLD constraint

The FLD constraint is shown in Figure 6-11. Two cases are distinguished for the FLD constraint:

- The values of some strain points are located above the FLD curve. In this case the constraint is computed as

$$g = d_{\max}$$

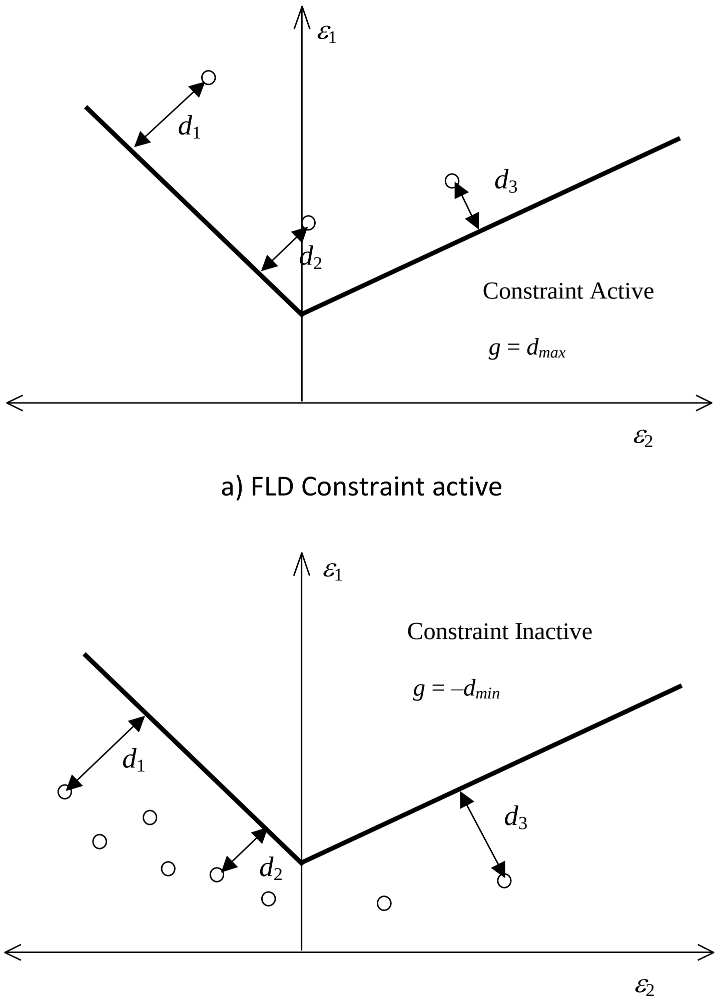
  with d_{max} the maximum smallest distance of any strain point above the FLD curve to the FLD curve.

- All the values of the strain points are located below the FLD curve. In this case the constraint is computed as

$$g = -d_{\min}$$

  with d_{min} the minimum smallest distance of any strain value to the FLD curve (Figure 6-11).

It follows that for a feasible design the constraint should be set so that $g(\mathbf{x}) < 0$.



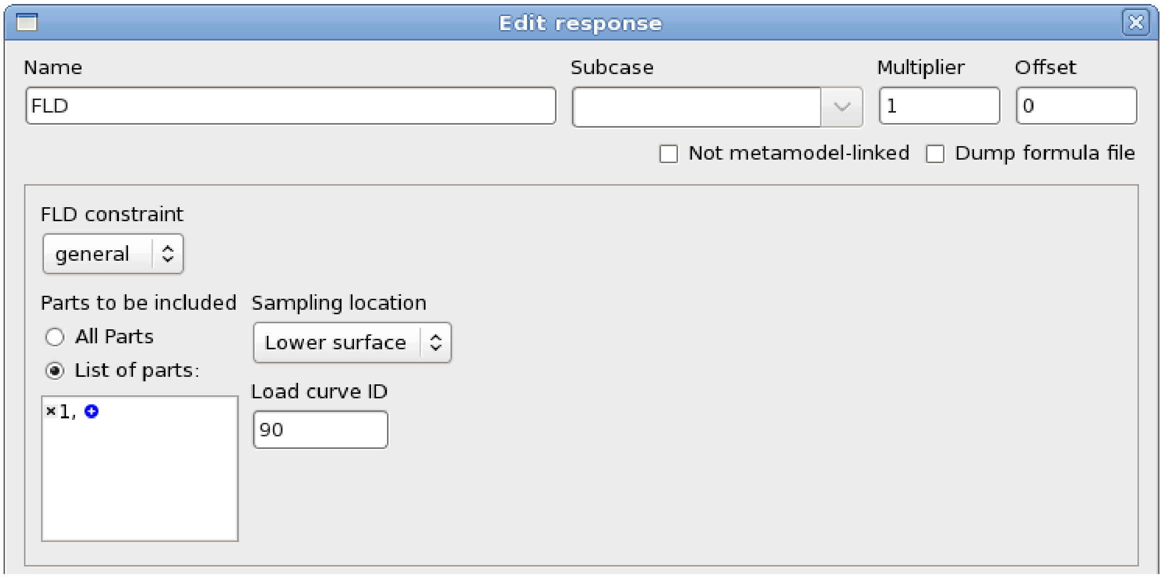
a) FLD Constraint active

b) FLD Constraint inactive

Figure 6-11: FLD curve – constraint definition

General FLD constraint

A more general FLD criterion is available if the forming limit is represented by a general curve. Any of the upper, lower or middle shell surfaces can be considered.



Edit response

Name: FLD

Subcase:

Multiplier: 1

Offset: 0

□ Not metamodel-linked □ Dump formula file

FLD constraint: general

Parts to be included

○ All Parts

● List of parts:

×1,

Sampling location: Lower surface

Load curve ID: 90

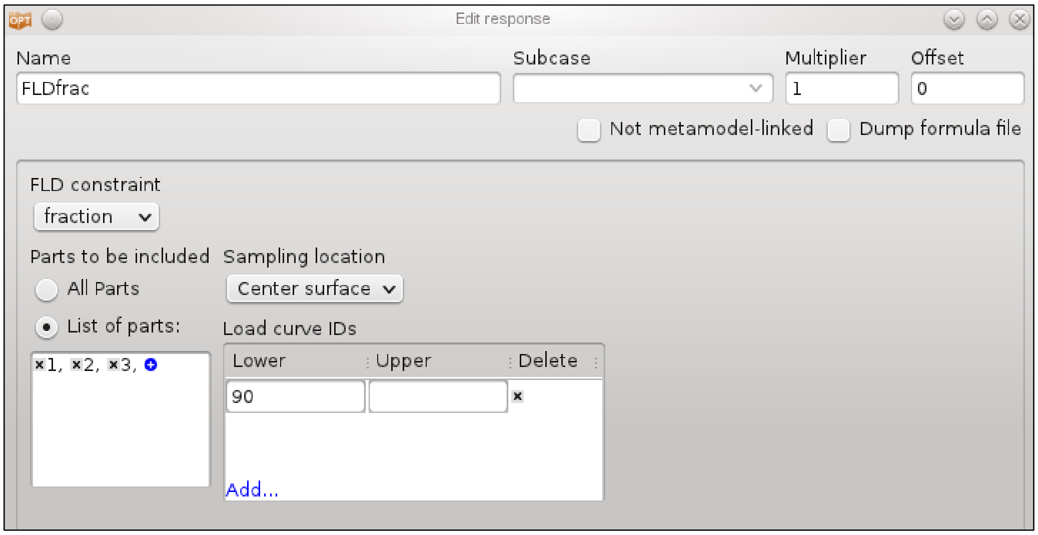
Figure 6-12: Definition of General FLD constraint

Remarks

1. A piece-wise linear curve is defined by specifying a list of interconnected points. The abscissae (ε_2) of consecutive points must increase (or an error termination will occur). Duplicated points are therefore not allowed.
2. The curve is extrapolated infinitely in both the negative and positive directions of (ε_2). The first and last segments are used for this purpose.
3. The computation of the constraint value is the same as shown in Figure 6-11.

FLD Fraction constraint

This option calculates the fraction of elements that violate the FLD constraint(s). Thus, it is useful in cases where a certain fraction of failure is tolerated. It also allows the definition of multiple zones, each defined by either one curve or two curves. The curves define the upper and lower bounds of the allowable principle strain states. The zones can overlap with each other, and the fraction calculated represents failure based on any of the defined criteria.



Edit response

Name: FLDfrac

Subcase:

Multiplier: 1

Offset: 0

□ Not metamodel-linked □ Dump formula file

FLD constraint: fraction

Parts to be included: ○ All Parts ● List of parts: 1, 2, 3,

Sampling location: Center surface

Load curve IDs

Lower	Upper	Delete
90		☒

Add...

Figure 6-13: Definition of FLD fraction constraint

The following must be defined for the model and FLD curve:

Table 6-9: LS-DYNA General FLD constraint options description

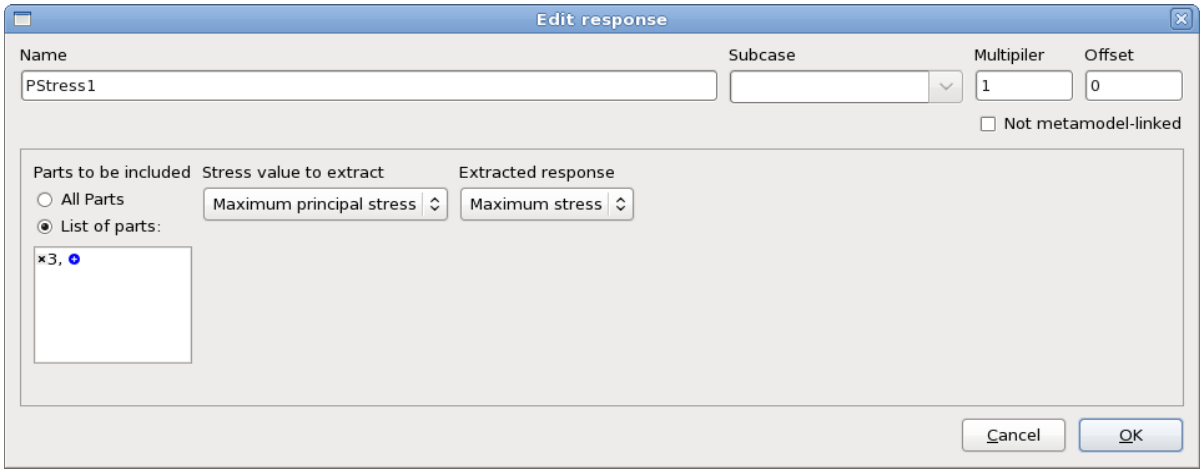
Option	Description
FLD constraint	General or fraction
Parts to be included	Entity is extracted for the entire model or for the parts IDs specified in the list.
Sampling location	Lower, middle or upper surface of the sheet
Load curve ID(s)	Identification number of a load curve in the LS-DYNA input file. The `*DEFINE_CURVE` keyword must be used. Refer to the LS-DYNA User’s Manual for an explanation of this keyword.

Remarks

1. The interface program produces an output file `FLD_curve` which contains the ε_1 and ε_2 values in the first and second columns respectively. Since the program first looks for this file, it can be specified in lieu of the keyword specification. The user should take care to remove an old version of the `FLD_curve` if the curve specification is changed in the keyword input file. If a structured input file is used for LS-DYNA input data, `FLD_curve` *must* be created by the user.
2. The scale factor and offset values feature of the `*DEFINE_CURVE` keyword are not utilized.

6.3.3. Principal stress

Any of the principal stresses or the mean can be computed using the `PSTRESS` interface. The values are nodal stresses.



Edit response

Name: PStress1

Subcase:

Multipiler: 1

Offset: 0

☐ Not metamodel-linked

Parts to be included

○ All Parts

● List of parts:

×3,

Stress value to extract: Maximum principal stress

Extracted response: Maximum stress

Figure 6-14: Principal Stress Interface

Table 6-10: Principal Stress options description

Item	Description	
Parts to be included	Entity is extracted for the entire model or for the parts IDs specified in the list.	
Stress value to extract	Maximum principal stress	σ_1
	Second principal stress	σ_2
	Minimum principal stress	σ_3
	Mean of principal stress	$(\sigma_1 + \sigma_2 + \sigma_3)/3$
Extracted response	Minimum, maximum or average computed over all the elements of the selected parts	

6.4. Generic Interfaces for History, Multihistory and Response extraction

6.4.1. Expressions

Mathematical expressions using previously defined entities can be defined here. The expression syntax and the available mathematical functions are described in Appendix F: .

Response Expressions

Histories may be evaluated at a specific time and multiresponse may be evaluated at a particular spatial point, respectively, to get a response.

Examples

```
history_name(0.05)
multiresponse_name(13)
```

Multiresponse Expressions

Multiresponses used in an expression must be written with the independent variable "pt" as a function: `my_multiresp( pt )`.

Example

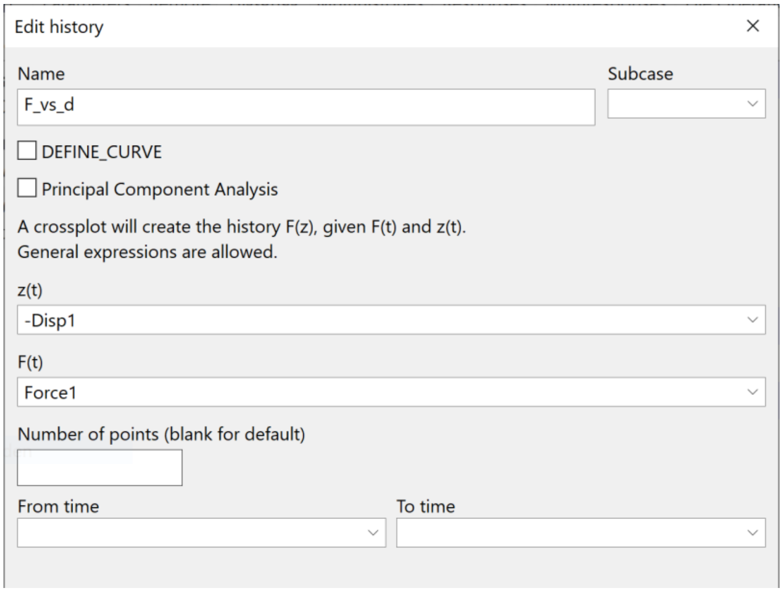
The expression

```
a( pt ) + b( pt )
```

defines the sum of two multiresponses a and b.

6.4.2. Crossplot history

A special history and multihistory function `Crossplot` is provided to construct a curve $F(z)$ given $F(t)$ and $z(t)$. File histories and file multihistories can also be used. For the Multihistory Crossplot, a history *or* multihistory can be selected to represent $z(t)$ or $F(t)$.



Edit history

Name: F_vs_d

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t): -Disp1

F(t): Force1

Number of points (blank for default):

From time:

To time:

Figure 6-15: Interface to define a crossplot history

The options are explained in Table 6-11.

Table 6-11: Description of `Crossplot` arguments

Option	Description	Default
$z(t)$	History or Multihistory of abscissa	-
$F(t)$	History or Multihistory of ordinate	-
Number of points	Number of points created in crossplot	Smallest of the numbers of points defining f and g
From time	Begin time	Largest t_0-value of F and z
To time	End time	Smallest t_p-value of F and z,

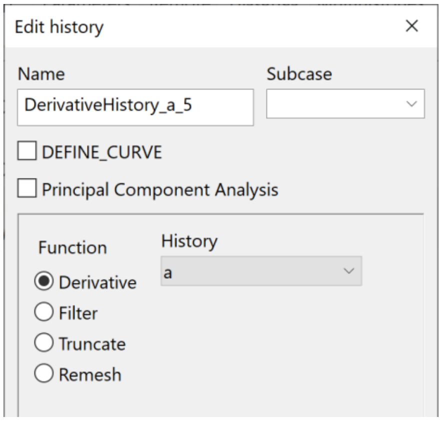
6.4.3. Function Interface

The functions available for the extraction of response values from previously defined histories are explained in Appendix F.3 .

The History and Multihistory functions are described below.

Derivative history

A special history function `Derivative` is provided to construct a curve $df(t)/dt$ given $f(t)$. Finite difference weights based on a 3-point template are used for the calculation [3]. The grid spacing of the reference history can be arbitrary.



Name: DerivativeHistory_a_5

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

Function: ● Derivative ○ Filter ○ Truncate ○ Remesh

History: a

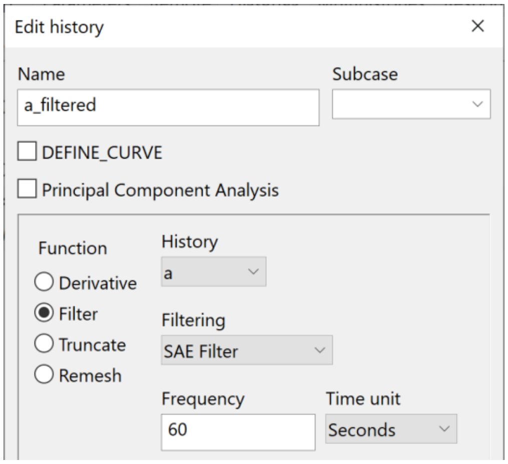
Figure 6-16: Interface to define derivative history

Remarks

1. Since the derivative approximation is based on a multipoint scheme, it is recommended to avoid having too few points in the history.
2. Irregular grid spacing is automatically supported.

Filtered history

A special history and multihistory function `Filter` is provided to construct a filtered curve.



Edit history

Name: a_filtered

Subcase

□ DEFINE_CURVE

□ Principal Component Analysis

Function

○ Derivative

● Filter

○ Truncate

○ Remesh

History: a

Filtering: SAE Filter

Frequency: 60

Time unit: Seconds

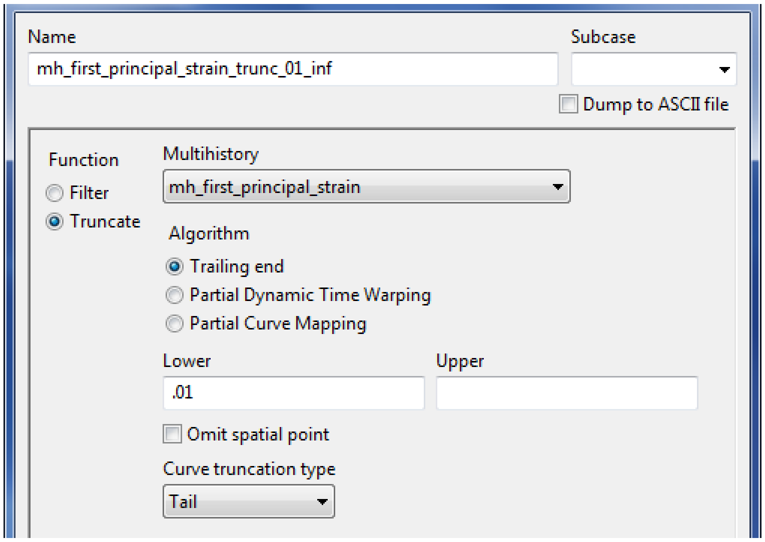
Figure 6-17: Interface to define a filtered history

Table 6-12: Description of ***`Filter`*** *arguments*

Argument name	Description
History	Pre-defined history
Filtering	Filtering type: SAE Filter, Butterworth Filter or Time Average
Frequency	Filtering frequency in Hz
Time unit	Units of time
Number of points	Number of averaging points

Curve truncation

Special history and multihistory functions are provided to truncate curves, Figure 6-18.



Name: mh_first_principal_strain_trunc_01_inf

Subcase:

□ Dump to ASCII file

Function

○ Filter

● Truncate

Multihistory: mh_first_principal_strain

Algorithm

● Trailing end

○ Partial Dynamic Time Warping

○ Partial Curve Mapping

Lower: .01

Upper:

□ Omit spatial point

Curve truncation type: Tail

Figure 6-18: Interface to define a truncated multihistory

Table 6-13: Description of Truncate arguments

Argument name	**Description**	
History	Pre-defined history	
Algorithm	Trailing end	If the tail of the function lies within the bounds, it is truncated, see below.
	Partial Dynamic Time Warping	Partial curve used in P-DTW calculation, Section 6.4.4
	Partial Curve Mapping	Partial curve used in PCM calculation, Section 6.4.4
Lower limit	Lower limit	
Upper limit	Upper limit. Tail end of curve truncated for trailing points between limits.	
Omit spatial point	If a trailing time state falls within the limits so that the constraint is active, omit the spatial point (e.g. DIC point). Only multihistories.	
Curve truncation type	Only multihistories. Tail All	

Target History Target Multihistory	Previously defined history, file history, multihistory or file multihistory containing target values, Section 6.4.4
Regression points	Regression points used to calculate response, Section 6.4.4: From target curve Fixed number (equidistant, interpolated)

Experimental curves may have trailing ends (tails) which do not represent actual data (dead signal). On the simulation side, the solver may produce curves which exhibit trailing ends representing the vibrational behavior of a coupon after fracture.

Several truncation options are available to generate partials of curves. Using the Trailing end option, bounds on the function value of the curve may be specified for truncating the trailing end of a curve. If the tail of the function lies within the bounds, it is truncated. The function loops over the curve points starting from the tail end and stops at the first point it finds outside the limits going backwards. To fill in the gap, an additional point is constructed on the bound using the intersected curve segment (edge).

Mathematically, the interval is closed: *[L,U]*, so that the limits are included. Thus if an experimental curve has a dead trailing signal (exact 0-flatlined tail), [0,0] will truncate the tail.

The curve can be bounded from one side ([L,∞] or [∞,U]) by omitting the other bound value, i.e. the curve is *preserved* below L or above U respectively. The transformation is ignored if both bounds are omitted.

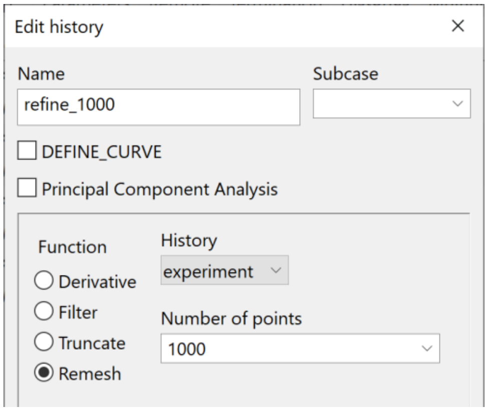
If the last curve point is outside the prescribed limits, the curve is fully preserved.

If the truncation type is “All”, the first point is always kept since this point often has coordinate (0,0).

For a detailed description of Partial Dynamic Time Warping and Partial Curve Mapping see Section 6.4.4.

Remesh

History applications may require remeshing of the curves. The number of points of the result curve may be specified. The new points are distributed uniformly along the normalized curve (i.e. scaled to [0,0];[1,1]).


Edit history

Name: refine_1000

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

Function
○ Derivative
○ Filter
○ Truncate
● Remesh

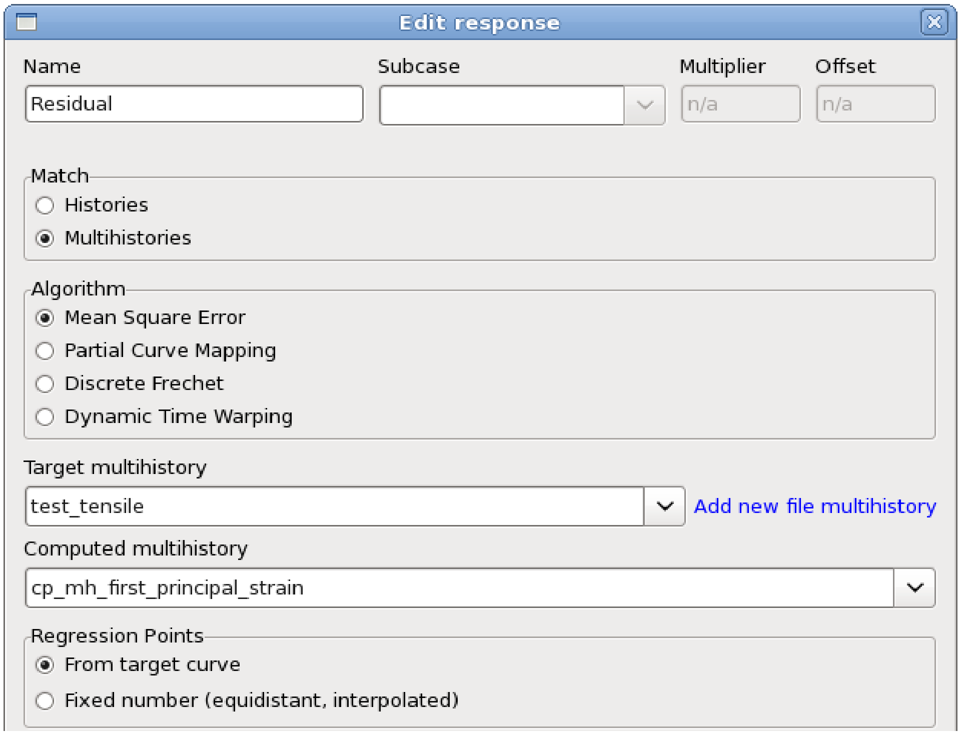
History: experiment

Number of points: 1000

Figure 6-19: Interface to remesh a history

6.4.4. Curve Matching Response

The Curve Matching interface provides metrics for comparison of target curves and curves extracted from simulation runs, Figure 6-20. Both histories and multipoint histories may be used. The options are explained in Table 6-14.



Edit response

Name: Residual

Subcase:

Multiplier: n/a

Offset: n/a

Match
- ○ Histories
- ● Multihistories

Algorithm
- ● Mean Square Error
- ○ Partial Curve Mapping
- ○ Discrete Frechet
- ○ Dynamic Time Warping

Target multihistory: test_tensile

Add new file multihistory

Computed multihistory: cp_mh_first_principal_strain

Regression Points
- ● From target curve
- ○ Fixed number (equidistant, interpolated)

Figure 6-20: Curve Matching Response dialog

Table 6-14: Curve Matching Response options

Option	Description	Reference
Match	Histories or Multihistories	
Algorithm	Curve matching metric to calculate "distance" between target and computed curve:	Section 28.2
	Mean Square Error Partial Curve Mapping Discrete Frechet Dynamic Time Warping Partial DTW	Section 28.2.1 Section 28.2.4 Section 28.2.2 Section 28.2.3 Section 28.2.3
Target history Target multihistory	Previously defined history, file history, multihistory or file multihistory containing target values.	

Add new file history Add new file multihistory	If the file history or file multihistory to be used as Target curve is not already defined, this can be done here.	Section 6.18 Section 6.19
Computed history Computed multihistory	Previously defined history, multihistory or Crossplot extracted from simulation results.	
Regression points	Regression points used to calculate response: From target curve Fixed number (equidistant, interpolated)	

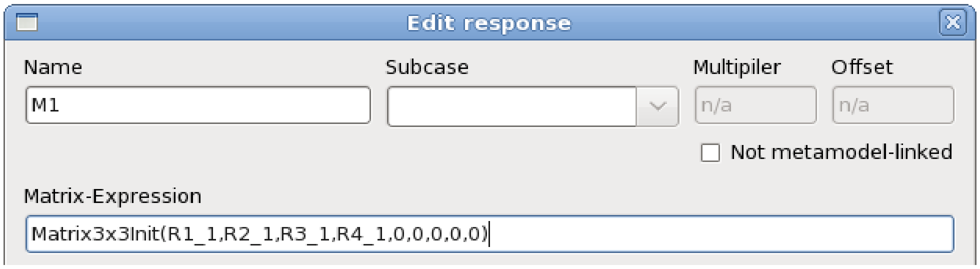
6.4.5. Matrix operations

Matrix operations can be performed by initializing a matrix, performing multiple matrix operations, and extracting components of the matrix as response functions or results. All these operations are defined using the `MATRIX_EXPRESSION` interface, Figure 6-21.

There are two functions available to initialize a matrix, namely `Matrix3x3Init` and `Rotate`. Both functions create 3×3 matrices.

The component of a matrix is extracted using the format `A.aij` (or the 0-based `A[i-1][j-1]`) e.g. `Strain.a23` (or `Strain[1][2]`) where *i* and *j* are limited to 1,2 or 3.

The matrix operation ***A*** – ***I*** (where ***I*** is the unit matrix) is coded as `A-1`.



Edit response

Name: M1

Subcase:

Multipiler: n/a

Offset: n/a

☐ Not metamodel-linked

Matrix-Expression: Matrix3x3Init(R1_1,R2_1,R3_1,R4_1,0,0,0,0,0)

Figure 6-21: Matrix Expression: Initialization of a matrix

Initializing a matrix

The command to initialize the matrix

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

is

Matrix3x3Init(a11,a12,a13, a21,a22,a23, a31,a32,a33) where a_{ij} is any previously defined variable (typically a response or result).

Creating a rotation matrix using 3 specified points

The expression is

Rotate(x1,y1,z1, x2,y2,z2, x3,y3,z3)

where the three triplets represent points 1, 2 and 3 in 3-dimensional space respectively.

- The vector v_{23} connecting points 2 and 3 forms the local ***X*** direction.
- $\boldsymbol{Z} = v_{23} \times v_{21}$
- $Y = Z \times X$

The vectors ***X, Y*** and ***Z*** are normalized to $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ which are used to form an orthogonal matrix:

$$T = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_3 & z_3 \end{bmatrix}$$

where $T^T = I$.

6.5. Injury criteria

All of the injury criteria were developed according to the specification in [1].

Injury criteria must be defined as responses, for some criteria, the intermediate histories are also available for extraction.

6.6. Head Injury Criteria

6.6.1. HIC

See Section 6.11.

6.7. Neck Criteria

6.7.1. MOC

MOC is the abbreviation for total Moment about Occipital Condyle. The criterion for the Total Moment calculates the total moment in relation to the moment measurement point.

The Total Moment MOC value for the Upper-Load-Cell is calculated as follows

$$MOC = M - (D \cdot F)$$

with	MOC	Total moment [Nm]
	F	Neck axial force resultant [N]
	M	Neck s-moment resultant [Nm]
	D	Distance between the force sensor axis and the Condyle axis,

depends on the dummy type, Table 6-16.

Table 6-15: Options for MOC

Option	**Description**	**Symbol**
Neck Force x	Neck axial force resultant	F
Neck Moment y	Neck s-moment resultant	M
Dummy_type	Dummy type	-
Length unit	Length units	-
Force unit	Force units	-

Table 6-16: Input constants for various dummy types

Dummy Type	**D[m]**
Hybrid III, male 95%	0.01778
Hybrid III, male 50%	0.01778
Hybrid III, female 5%	0.01778
Hybrid III, 10-year	0.01778
Hybrid III, 6-year	0.01778
Hybrid III, 3-year	0
Crabi 12, 18 month	0.00584
TNO P1,5	0.0247
Crabi 6 month	0.0102
TNO P 3/4, P3	0

ES-2	0
TNO Q series	0
SID-IIs	0.01778
BioRID	0.01778
WORLDSID	0.0195

6.7.2. NIC (rear impact)

NIC is the abbreviation for Neck Injury Criterion. LS-OPT calculates the NIC value specified for rear impact. The NIC value is calculated with the following formula:

$$NIC = a_{relative} \cdot 0.2 + v_{relative}^2$$

with $a_{relative} = a_x^{TI} - a_x^{Head}$ relative x-acceleration

$v_{relative} = \int a_{relative}$

Table 6-17: Options for NIC

Option	**Description**	**Symbol**
Acceleration 1. thorax spine	x-acceleration of first thorax spine	a_x^{TI}
Acceleration head	x-acceleration at the height of the c.o.g. of the head	a_x^{Head}
Time unit	Time units	-
Length unit	Length units	-

6.7.3. Nij (Nce, Ncf, Nte, Ntf)

Nij is the abbreviation for Normalized Neck Injury Criterion and is the four neck criterion Nte (tension-expression), Ntf (tension-flexion), Nce (compression-extension) and Ncf (compression-flexion).

The Nij value is the maximal value of Nte, Ntf, Nce, Ncf and is calculated with the following formula:

$$NIJ = \frac{F}{F_c} + \frac{MOC}{M_c}$$

with		
	F	Force at the point of transition from head to neck (t-shear resultant)
	F_c	Critical force (depending on dummy type)
	MOC	Total Moment (see *MOC*, section 6.7.1)
	M_c	Critical moment (depending on dummy type)

Table 6-18: Options for Nij arguments

Option	**Description**	**Symbol**
Neck Force x	Neck axial force resultant	See MOC
Neck Moment y	Neck s-moment resultant	See MOC
Neck Force z	Force at the point of transition from head to neck	*F*
Dummy type	Dummy type	-
Length unit	Length units	-
Force unit	Force units	-

Table 6-19: Input constants for various dummy types

Dummy type	**Test**	**F_C [N]**	**F_C [N]**	**M_C [Nm]**	**M_C [Nm]**
		Tension	**Compression**	**Flexion**	**Extension**
Hybrid III; male 50%	In position	6806	-6160	310	-135
Hybrid III; female 5%	In position	4287	-3880	155	-67
Hybrid III; female 5%	Out of position	3880	-3880	155	-61
Hybrid III; 6-year	Out of position	2800	-2800	93	-37
Hybrid III; 3-year	Out of position	2120	-2120	68	-27
Hybrid III; 12 month	Out of position	1460	-1460	43	-17

6.7.4. Nkm (Nfa, Nea, Nfp, Nep)

Nkm corresponds to the four neck criteria Nfa (flexion-anterior), Nea (extension-anterior), Nfp (flexion-posterior) and Nep (extension-posterior).

The Nkm value is calculated with the following formula, [2]:

$$Nkm(t) = \frac{F(t)}{F_{\text{int}}} + \frac{MOC(t)}{M_{\text{int}}}$$

with F Force at the point of transition from head to neck (axial force resultant)

F_{int} Critical force

MOC Total Moment (see MOC, section 6.7.1)

M_{int} Critical moment

Table 6-20: Options for Nkm arguments

Option	**Description**	**Symbol**
Neck Force x	Neck axial force resultant	*F*
Neck Moment y	Neck s-moment resultant	See MOC
Dummy type	Dummy type	-
Length unit	Length units	-
Force unit	Force units	-
Criterion	Nfa, Nea, Nfp, Nep	-

Table 6-21: Input constants

Criteria	**Description**	**Value**
*_anterior	Positive Shear F_{int}	845 N
*_posterior	Negative Shear F_{int}	-845 N
flexion_*	Flexion M_{int}	88.1 Nm
extension_*	Extension M_{int}	-47.5 Nm

6.7.5. LNL

LNL is the abbreviation for the Lower Neck Load Index. The LNL value is calculated with the following formula:

$$LNL = \frac{\sqrt{M_y^2 + M_x^2}}{C_{moment}} + \frac{\sqrt{F_y^2 + F_x^2}}{C_{shear}} + \left| \frac{F_z + off}{C_{tension}} \right|$$

with	M_y	s-Moment resultant
	M_x	Torsional resultant
	C_{moment}	Critical moment
	F_x	s-Shear resultant
	F_y	Axial force resultant
	C_{shear}	Critical force
	F_z	t-Shear resultant
	$C_{tension}$	Critical force
	off	offset to include pre-load, depends on dummy position

Table 6-22: Options for LNL arguments

Option	Description	Symbol
y Force	Axial force resultant	F_y
x Force	s-Shear resultant	F_x
z Force	t-Shear resultant	F_z
y Moment	s-Moment resultant	M_y
x Moment	Torsional resultant	M_x
Length unit	Length units	-
Force unit	Force units	-

Table 6-23: Input constants

Force/Moment	Description	Value
C_{moment}	Critical moment	15 [Nm]
C_{shear}	Critical force	250 [N]
$C_{tension}$	Critical force	900 [N]

6.8. Chest Criteria

6.8.1. Chest compression

Maximum relative rotation multiplied by a constant:

$$C_1 \max_t [\Theta(t)]$$

Table 6-24: Options for Chest Compression arguments

Option	Description	Symbol
History	relative rotation history	$\Theta(t)$
Dummy type	dummy type	−

Table 6-25: Input constants for various dummy types

Dummy Type	Scaling factor C_1
Hybrid III; male 95%	130.67
Hybrid III; male 50%	-139.0
Hybrid III; female 5%	-87.58

Remarks

The user is responsible for any required filters of the input history.

6.8.2. Viscous criterion (VC)

VC is an injury criterion for the chest area. The VC value [m/s] is the maximum crush of the momentary product of the thorax deformation speed and the thorax deformation. Both quantities are determined by measuring the rib deflection (side impact) or the chest deflection (frontal impact). The formula is:

$$-\min \frac{C_1}{C_2} Y(t) \frac{dY(t)}{dt}$$

Table 6-26: Options for Viscous Criterion arguments

Argument name	**Description**	**Symbol**
History	Thoracic deformation (m)	*Y(t)*
Dummy type	Dummy type	−
Time unit	Time units	−
Length unit	Length units	−

Table 6-27: Input constants for various dummy types

Dummy Type	**Scaling factor C_1**	**Deformation constant C_2 (m)**
Hybrid III; male 95%	1.3	0.254
Hybrid III; male 50%	1.3	0.229
Hybrid III; female 5%	1.3	0.187
BioSID	1.0	0.175
EuroSID-1	1.0	0.140
EuroSID-2	1.0	0.140
SID-IIs	1.0	0.138

Remarks

1. The derivative is computed using the 4th order (template size = 5) finite difference approximation:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h} + O(h^4)$$

where *h* is the time interval between the single measurements.

2. The user is responsible for any required filters of the input history.

6.8.3. Thoracic Trauma Index (TTI)

TTI is the abbreviation for Thoracic Trauma Index (Thorax Trauma Index).

The TTI value is calculated using the following formula:

$$TTI = \frac{A(\max .rib) + A(lwr.spine)}{2}$$

$$A(\max .rib) = \max\{ A(upr.rib), A(lwr.rib)\}$$

with $A(upr.rib)$ Maximum y-acceleration of the upper rib

$A(lwr.rib)$ Maximum y-acceleration of the lower rib

$A(lwr.spine)$ Maximum y-acceleration of the lower spine

The result is divided by the gravitational acceleration g (9810mm/s²).

Table 6-28: Options for TTI arguments

Option	**Description**	**Symbol**
Acceleration upper rib	y-acceleration of the upper rib	$A(upr.rib)$
Acceleration lower rib	y-acceleration of the lower rib	$A(lwr.rib)$
Acceleration lower spine	y-acceleration of the lower spine	$A(lwr.spine)$
Time unit	Time units	-
Length unit	Length units	-

6.9. Criteria for the Lower Extremities

6.9.1. Tibia Index (TI)

TI is the abbreviation for the Tibia Index.

The calculation of the TI value in based on the equation

$$TI = \left|\frac{M}{M_C}\right| + \left|\frac{F}{F_C}\right|$$

$$M = \sqrt{(M_x)^2 + (M_y)^2}$$

with $M_{x/y}$ Bending moments [Nm] (torsional resultant, s-moment resultant)

M_C Critical bending moment

F Axial compression [kN] (t-shear resultant)

F_C Critical compression force

Table 6-29: Options for TI arguments

Argument name	**Description**	**Symbol**
Bending moment x	Bending moment, torsional resultant	M_x
Bending moment y	Bending moment, s-moment resultant	M_y
Axial compression z	Axial compression, t-shear resultant	F
Dummy type	Dummy type	-
Length unit	Length units	-
Force unit	Force units	-

Table 6-30: Input constants for various dummy types

Dummy type	**Critical bending moment [Nm]**	**Critical compression force [kN]**
Hybrid III, male 95%	307.0	44.2
Hybrid III, male 50%	225.0	35.9
Hybrid III, female 5%	115.0	22.9

6.10. Additional Criteria

6.10.1. A3ms

The smallest resultant acceleration level maintained for 3ms. $r_{\Delta t}$ is computed as the level of $r = \sqrt{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2}$ exceeded for the specified time interval Δt (3ms). The resulting acceleration level is divided by the gravitational acceleration, *g = 9810mm/s²*.

Table 6-31: Options for a3ms arguments

Argument name	Description	Symbol
x History	*x*-acceleration history	$\ddot{x}$
y History	*y*-acceleration history	$\ddot{y}$
z History	*z*-acceleration history	$\ddot{z}$
Time unit	Time units	–
Length unit	Length units	–

Remarks

1. y History ($\ddot{y}$) and z History ($\ddot{z}$) are optional.
2. The user is responsible for any required filters of the input history.

6.11. LS-DYNA Binout injury criteria

The injury criteria HIC, HIC (3 nodes), Chest Severity Index, CLIP3m and CLIP3m (3 nodes) can only be compute for LS-DYNA. The acceleration components for the specified nodes will be extracted from binout, the magnitude computed, and the injury criteria computed from the acceleration magnitude history.

Note

- The length and time units are used to compute the gravity value based on 9.81 m/s^2

6.12. The GenEx tool for extracting responses and histories from a text file.

The GenEx tool is described in Chapter 12.

6.13. User-defined interface for extracting results

The user may provide an own extraction routine or any program, e.g. a postprocessor, to get response or history results. For responses, the command has to output a single floating-point number to standard output. For histories, the values have to be output to a file `LsoptHistory` in two columns. The command has to be specified in the *Definition* field in the *USERDEFINED* interface dialog, Figure 6-22.

Examples of the output statement in such a program for response extraction are:

- The C language:

```
printf ("%lf\n", output_value);
```

or

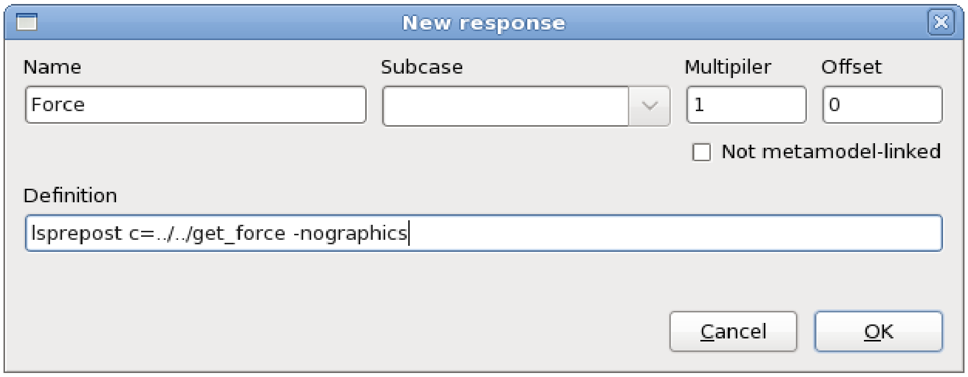
```
fprintf (stdout, "%lf\n", output_value);
```

- The FORTRAN language:

```
write (6,*) output_value
```

- The Perl script language:

```
print "$output_value\n";
```



Name: Force

Subcase:

Multipiler: 1

Offset: 0

☐ Not metamodel-linked

Definition: lsprepost c=../../get_force -nographics

Figure 6-22: Extracting a Response using a user-defined program

Examples

1. The user has an own executable program ”`ExtractForce`” which is kept in the directory `$HOME/own/bin`. The executable extracts a value from a result output file.
2. The relevant response definition command must therefore be as follows:

   ```
   $HOME/own/bin/ExtractForce
   ```

3. If *Perl* is to be used to execute the user script `DynaFLD2`, the command may be:

   ```
   $LSOPT/perl $LSOPT/DynaFLD2 0.5 0.25 1.833"
   ```

4. In this example the post-processor LS-PREPOST is used to produce a history file from the LS-DYNA database. The LS-PREPOST command file `get_force`:

   ```
   open d3plot d3plot
   ascii rcforc open rcforc 0
   ascii rcforc plot 4 Ma-1
   xyplot 1 savefile xypair LsoptHistory 1
   deletewin 1
   quit
   ```

   produces the `LsoptHistory` file. See Figure 6-22 for the LS-PREPOST command.

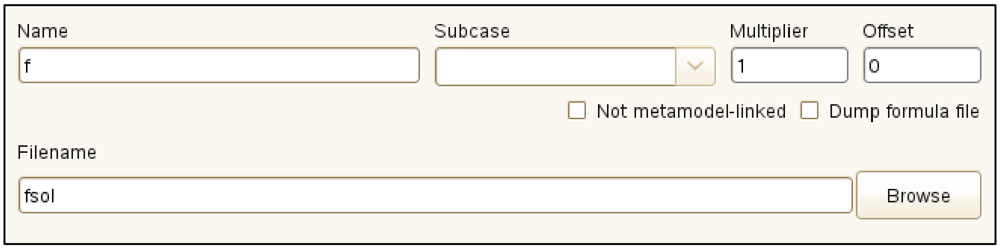
   Note : The *rcforc* history in this example can be obtained more easily by direct extraction, see Section 6.2.1 and Appendix A.1 .

Remarks

1. An alias must not be used for an interface program.
2. The program should be run in batch mode.
3. The program is called from the run directories. This has to be considered if relative paths are used.
4. Single quotes cannot be accommodated in the user-defined history and response command, respectively.

6.14. Response file

This is also a user-defined option, typically used in conjunction with a user-defined solver type. An output filename can be specified for extracting a single response output value. The user must write the calculated response value to the specified file during the simulation. The default for the filename is the name of the response. Figure 6-23 shows the dialog.



Name: f
Subcase:
Multiplier: 1
Offset: 0
□ Not metamodel-linked □ Dump formula file
Filename: fsol

Figure 6-23: Dialog for extracting a response value from a file

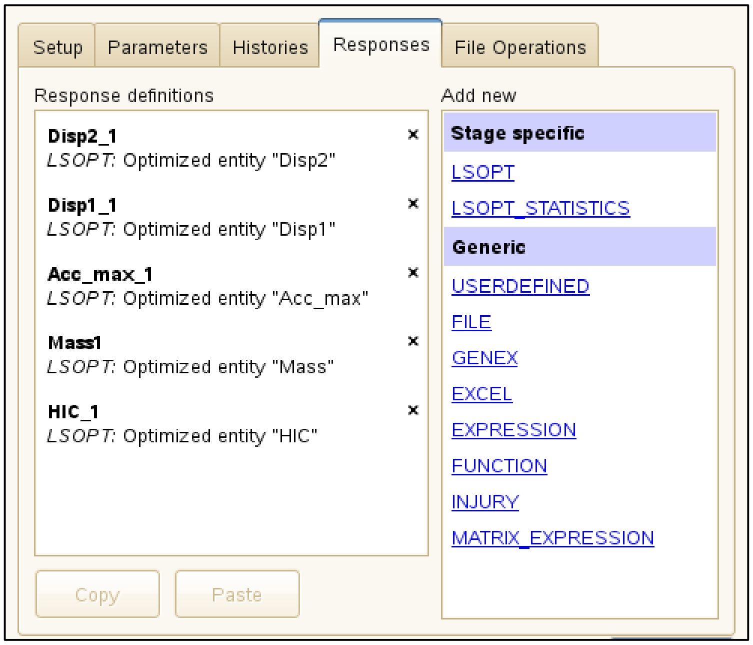
6.15. Extraction of LS-OPT entities

6.15.1. LS-OPT responses

The LS-OPT stage is used in the context of multilevel optimization, which involves running an inner level optimization within an outer level optimization. Each outer level sample evaluation, i.e. LS-OPT stage evaluation, involves an inner optimization. The results of these evaluations consist of entities that are optimized with respect to the inner level variables, which can be defined by the user as responses for the outer level LS-OPT setup.

The response dialog of the LS-OPT stage type provides the option to define an *LSOPT* response, which lists the available entities optimized in the inner level. These entities can be the optimized inner level variables or the corresponding optimized responses, composites, objective functions or constraints (Figure 6-25). It is also possible to extract responses at any specific inner level iteration by clicking the *'Iteration'* radio button and providing the required iteration number.

Since the inner level can also be a Monte Carlo analysis, statistical values such as standard deviation, mean and probability of failure are available in the *LSOPT_STATISTICS* interface, Section 6.15.3.



Disp2_1 ×
LSOPT: Optimized entity "Disp2"

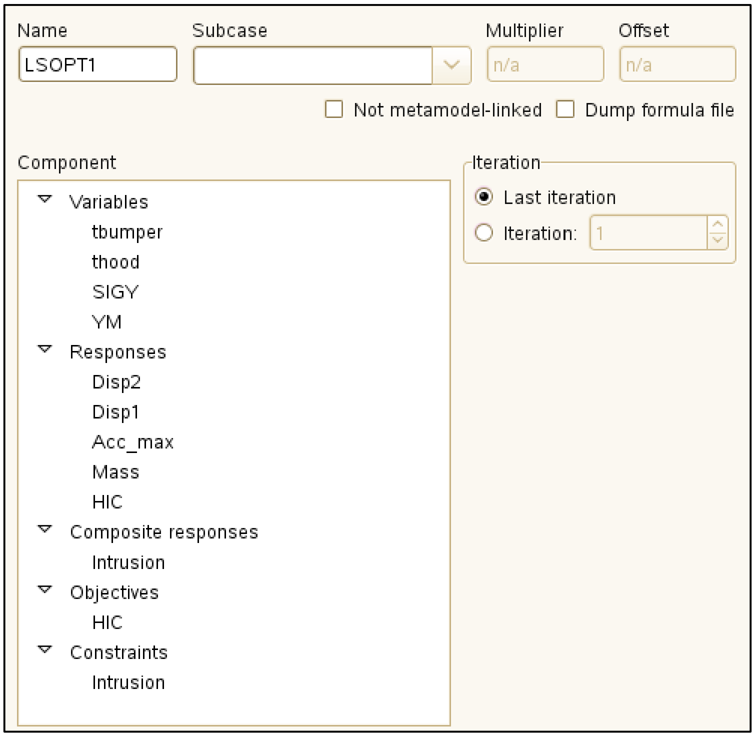
Disp1_1 ×
LSOPT: Optimized entity "Disp1"

Acc_max_1 ×
LSOPT: Optimized entity "Acc_max"

Mass1 ×
LSOPT: Optimized entity "Mass"

HIC_1 ×
LSOPT: Optimized entity "HIC"

Figure 6-24: Main dialog for the extraction of LS-OPT stage responses. A special category (LSOPT STATISTICS) is available for statistical results produced by a Monte Carlo analysis.



Name: LSOPT1

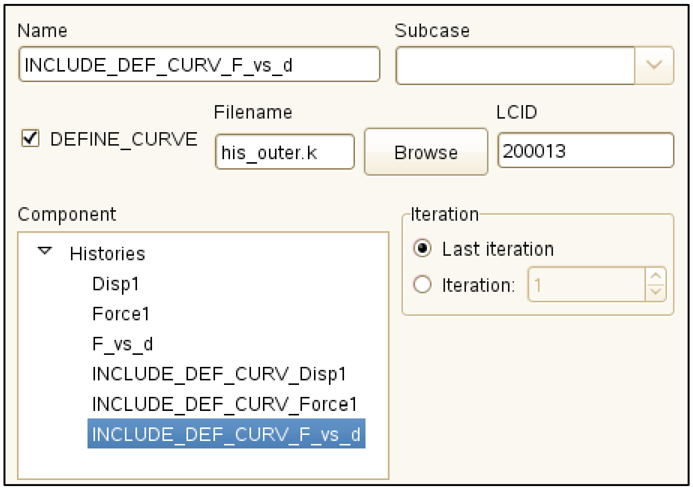
□ Not metamodel-linked □ Dump formula file

● Last iteration

Figure 6-25: Dialog for the extraction of LS-OPT optimal response results

6.15.2. LS-OPT histories

Figure 6-26 depicts the dialog for defining an LS-OPT history. Optimal histories produced by an optimization run can be extracted and converted to an LS-DYNA `*DEFINE_CURVE` keyword file, Section 6.1.2. This file can then be inserted into a subsequent stage analysis as an include file. Multiple `*DEFINE_CURVE` data sets can be dumped in the same file.



Name: INCLUDE_DEF_CURV_F_vs_d
Subcase:

☑ DEFINE_CURVE
Filename: his_outer.k
LCID: 200013

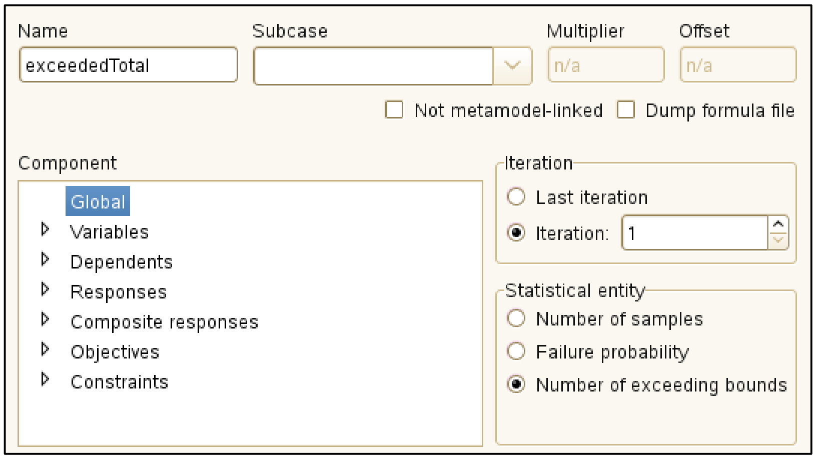
Component
- Histories
  - Disp1
  - Force1
  - F_vs_d
  - INCLUDE_DEF_CURV_Disp1
  - INCLUDE_DEF_CURV_Force1
  - INCLUDE_DEF_CURV_F_vs_d

Iteration
● Last iteration
○ Iteration: 1

Figure 6-26: Dialog for defining an LS-OPT history. The DEFINE_CURVE option has been selected to produce an LS-DYNA keyword file.

6.15.3. LS-OPT reliability statistics

The *LSOPT_STATISTICS* interface is a special category of the LS-OPT solver type responses which represent statistical values produced by a Monte Carlo analysis (direct or metamodel-based). Values can be extracted for global statistics (see Figure 6-27) or for individual entities such as constraints (see Figure 6-28), variables, dependents, responses and composites.



Name: exceededTotal
Subcase:
Multiplier: n/a
Offset: n/a

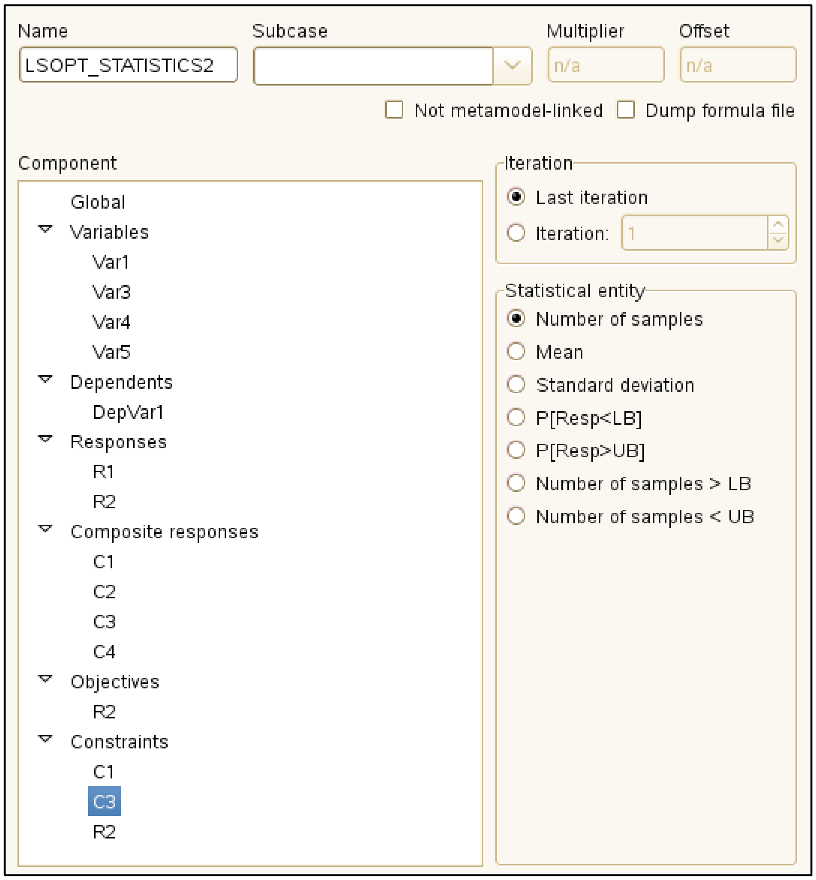
□ Not metamodel-linked □ Dump formula file

Component
- Global
- Variables
- Dependents
- Responses
- Composite responses
- Objectives
- Constraints

Iteration
○ Last iteration
● Iteration: 1

Statistical entity
○ Number of samples
○ Failure probability
● Number of exceeding bounds

Figure 6-27: Dialog for extraction of global statistics produced by a Monte Carlo analysis



Name: LSOPT_STATISTICS2

Subcase:

Multiplier: n/a

Offset: n/a

□ Not metamodel-linked □ Dump formula file

Component

- Global
- Variables
  - Var1
  - Var3
  - Var4
  - Var5
- Dependents
  - DepVar1
- Responses
  - R1
  - R2
- Composite responses
  - C1
  - C2
  - C3
  - C4
- Objectives
  - R2
- Constraints
  - C1
  - C3
  - R2

Iteration

● Last iteration

○ Iteration: 1

Statistical entity

● Number of samples

○ Mean

○ Standard deviation

○ P[Resp<LB]

○ P[Resp>UB]

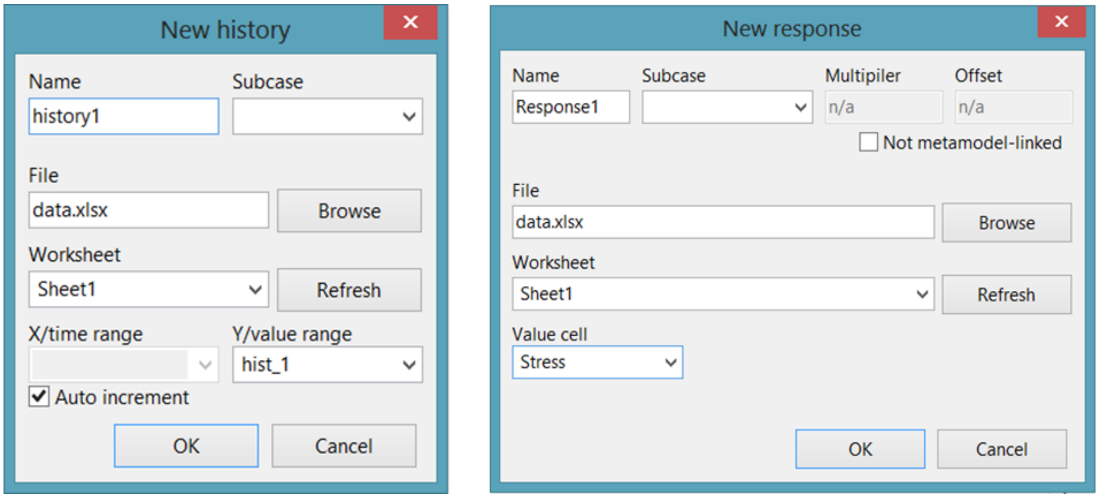
○ Number of samples > LB

○ Number of samples < UB

Figure 6-28: Dialog for extraction of constraint statistics produced by a Monte Carlo analysis

6.16. Excel

The histories and responses specific to *Microsoft Excel* can be defined using *EXCEL* option listed under *Generic* history and responses interfaces. The cells and/or array of cells of an *Excel* document can be defined as LS-OPT histories or responses and hence can also be utilized as design objective/constraints based on analysis *Task*.



New history

Name: history1

Subcase:

File: data.xlsx

Worksheet: Sheet1

X/time range:

Y/value range: hist_1

☑ Auto increment

New response

Name: Response1

Subcase:

Multipiler: n/a

Offset: n/a

□ Not metamodel-linked

File: data.xlsx

Worksheet: Sheet1

Value cell: Stress

Figure 6-29: Microsoft Excel History (left) and Response (right) interface

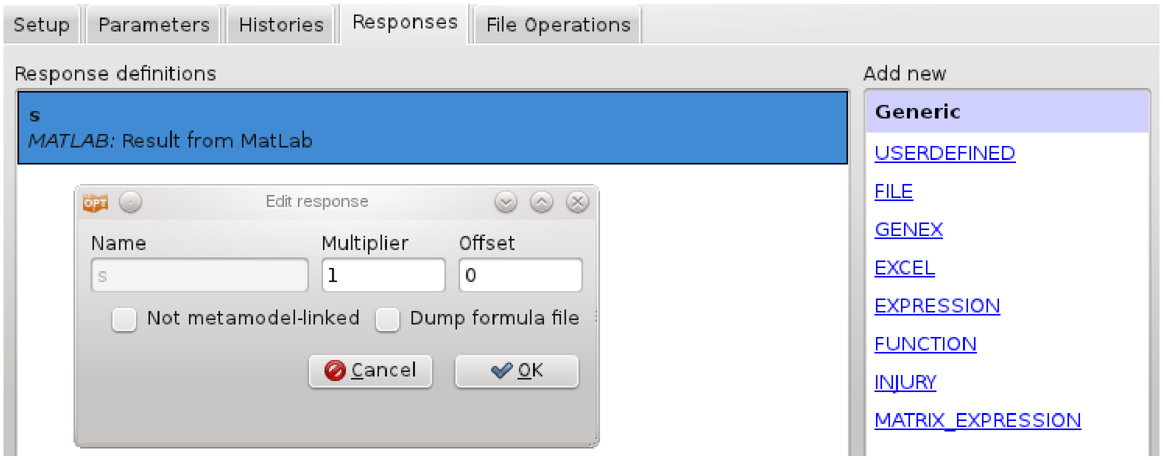
Figure 6-29 shows the interface for defining *Excel* histories and responses. The options are described in Table 6-32.

Table 6-32: Description of Excel History and Response options

Option	Description
File	*Excel* document for extraction
Worksheet	Worksheets of the *Excel* document are listed
X/time range	This field lists all the *Excel* names defined for cells and cell arrays. The name corresponding to the abscissa values of the history (typically time) should be selected. If auto increment is used, a positive integer sequence of length equal to the number of Y values is used starting from 1 (1, 2, 3…).
Y/value range	Lists all the cell names assigned to array of cells used for ordinate values of the histories.
Value cell	*Excel* cell assigned to response value.

6.17. Matlab

The histories and responses for a Matlab stage, Section 5.3.13, are defined in an output file specified in the stage setup dialog. The output file has the same format as for a METAPost or a User-defined stage, Section 5.3.15. Upon specifying an appropriate file, LS-OPT automatically populates the history and response dialogs. The response/history name cannot be edited manually, as only the responses and histories defined in the Matlab input files are allowed. Thus the effort associated with manual definition is avoided while avoiding errors at the same time.



Response definitions

s
MATLAB: Result from MatLab

Name: s
Multiplier: 1
Offset: 0
□ Not metamodel-linked
□ Dump formula file

Figure 6-30: Matlab response dialog. The response name cannot be edited manually.

6.18. File Histories

File Histories may be used to extract data from a text file containing a curve. File Histories are typically used to import test data for parameter identification problems.

Three file history types are available:

1. A history can be provided in a text file with arbitrary format. Coordinate pairs are found by assuming that any pair of numbers in a row is a legitimate coordinate pair. An example is given below.
2. GenEx can also be used to extract a curve from a text files (see Chapter 7).
3. An interface to extract data from files generated by the optical measurement system ARAMIS[14]. Supported versions are ARAMIS Professional v6.3 and ARAMIS Professional 2016- 2019.

File histories are global curves. They are neither sampling nor stage dependent; hence they are not listed in the *Stage* dialog history list.

[14] GOM GmbH

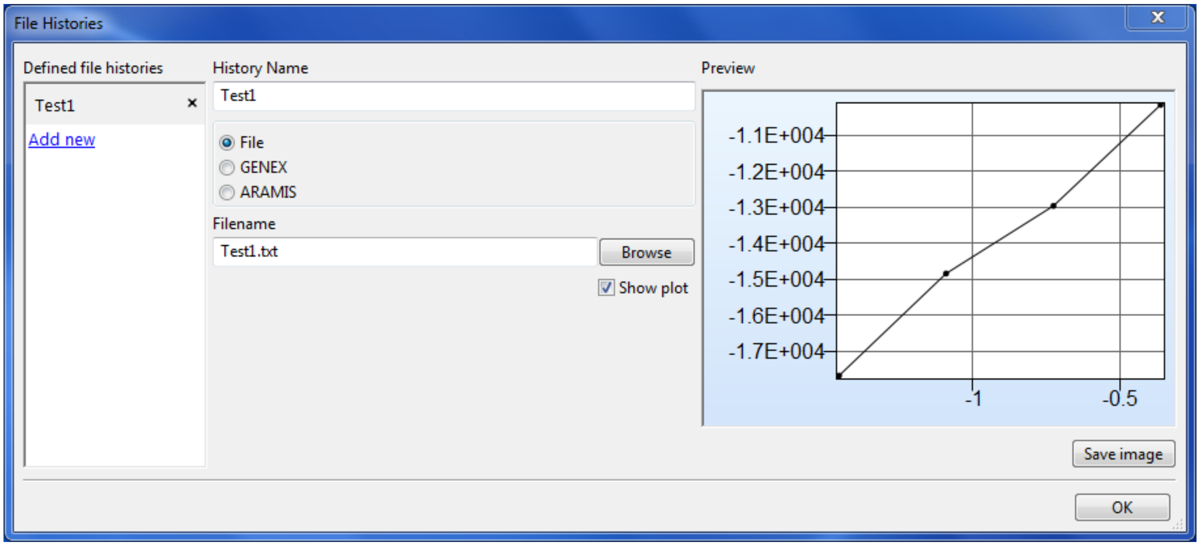


Figure 6-31: File Histories

File History Text File Example:

```
Time   Displacement
1.2,   143.97
1.4,   156.1
1.7,   923.77
```

6.19. File Multihistories

File Multihistories may be used to extract data from a set of text files containing multiple curves e.g. from full-field measurement. File Multihistories are typically used to import test data for parameter identification problems.

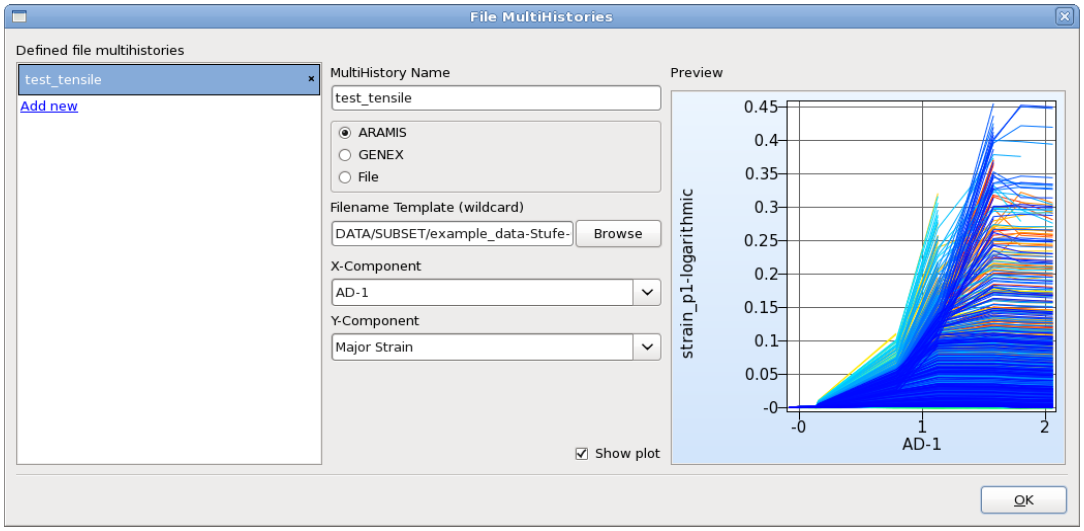
Three file multihistory types are available:

1. An interface to extract data from files generated by the optical measurement system ARAMIS[15]. Supported versions are ARAMIS Professional v6.3 and ARAMIS Professional 2016-2019.
2. GenEx can also be used to extract curves from multiple text files (see Chapter 7). The interface requires one file per time step.
3. A file option is also available for curves in a single text file using the LS-PrePost fixed format. An example is given below.

Remark

In options 1 and 2, intermediate files are created using the LS-PrePost format. This format is also used to create `history.n` and `multihistory`.n files for file- and other types of histories and multihistories.)

[15] GOM GmbH



test_tensile

MultiHistory Name: test_tensile

● ARAMIS
○ GENEX
○ File

Filename Template (wildcard): DATA/SUBSET/example_data-Stufe-

X-Component: AD-1

Y-Component: Major Strain

☑ Show plot

Figure 6-32: File Multihistory: ARAMIS interface showing preview

Example: Curves in LS-PrePost format (repeat for any number of curves)

```
header line 1 (any content or blank)
header line 2
header line 3
header line 4
header line 5
m #pts=n
x0 y0
x1 y1
.
.
.
xn yn
endcurve
```

Table 6-33: File Multihistory options

Option	Description
MultiHistory Name	Name of Multifilehistory
ARAMIS	Interface to data generated by optical measurement system ARAMIS
GENEX	GenEx interface, Chapter 7
File	Option to specify data in single file in LS-PrePost format
Filename Template (wildcard)	Name of files, in wildcard form, for extracting data. One file per time step.

X-Component (only ARAMIS)	X component (scalar or vector) of file multhistory. At least one of the components has to be a vector.
Y-Component (only ARAMIS)	Y component (scalar of vector) of file multhistory. At least one of the components has to be a vector.
Input GenEx	GenEx file (.g6)
Input data files (wildcard)	Data file template, assumes one file per stage/timestep. Wildcard is supported
Preview	Preview of multi-point data
X/time	X entity (scalar or vector) of file multihistory
Y/value vector	Y entity (scalar or vector) of file multihistory
Filename	Name of file containing curves in LS-PrePost format

File multihistories are global curves. They are neither sampling nor stage dependent; hence they are not listed in the *Stage* dialog multihistory list.

6.20. REFERENCES

[1] Data Processing Vehicle Safety Work Group - Crash Analysis Criteria Description. Version 2.1.1 Arbeitskreis Messdatenverarbeitung Fahrzeugsicherheit, May 2008.

[2] K.-U. Schmitt, M. Muser, How to calculate the N_{km} , Working Group on Accident Mechanics, Zürich, 2003

[3] Fornberg, B., Calculation of Weights in Finite Difference Formulas. SIAM Rev., Vol. 40, No. 3, 685-691, 1998.

7. GenEx: Extracting responses and histories from a text file

A user may choose to use a non-LS-DYNA solver for his application in which case the only elegant option, except for using commercial extraction tools (see e.g. Section 5.3.7), is to use a special graphical tool for identifying and extracting response values and history vectors from an output text file containing the analysis results. This chapter describes the use of the GenEx tool for extracting responses (scalars) and histories (vectors) from such a text file. GenEx is included in the LS-OPT distribution as the executable file `genex` and can be activated from the Responses, Histories or File Multihistories dialog.

7.1. The main window

GenEx can be started from the command line by typing `genex` <*filename*> or by selecting the ***Create/Edit*** button after selecting GenEx on the **Responses** or **Histories** page, or from the **File Histories** and **Multihistories** dialog, respectively.

When first starting GenEx, there will be two predefined anchors in the tree on the left, **Start of File** and **End of File**, Figure 7-1. It is not possible to change or remove these two anchors.

The middle part of the window displays the data file, with symbols for anchors and entities. The current entity/anchor will be highlighted or have a thin black border around it.

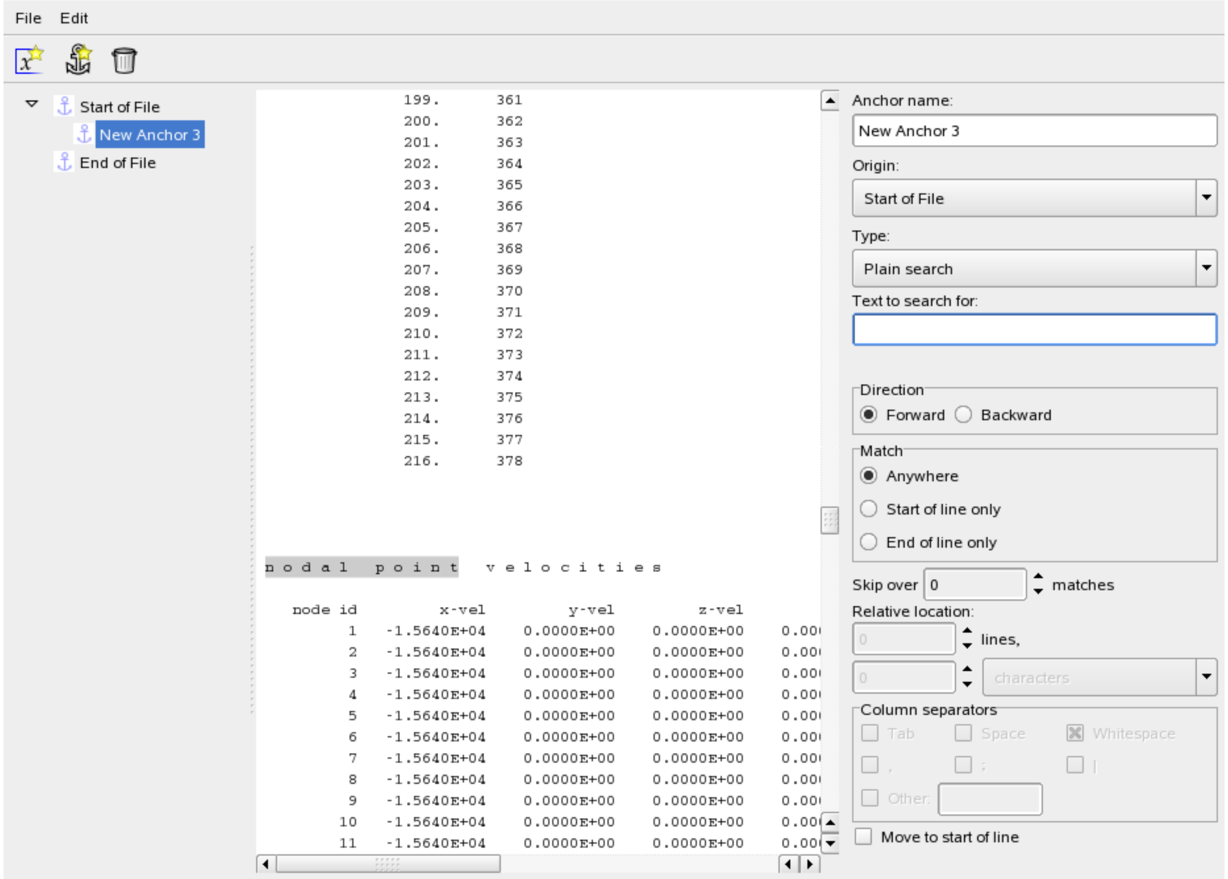
On the right is the dialog box for specifying options for the currently selected anchor/entity.

Anchors

Anchors describe how to find a certain position in the data file. This can be done with searching for keywords or with an absolute position.

Entities

An entity is a quantity we want to extract from LS-OPT. Entities describe both what the number should look like as well as where, relative to the parent, to find it. There are three types of entities, scalar, column and repeated anchor vectors (see Section **Options specific for entities** for the difference between them).



- Start of File
  - New Anchor 3
- End of File

```
                  199.      361
                  200.      362
                  201.      363
                  202.      364
                  203.      365
                  204.      366
                  205.      367
                  206.      368
                  207.      369
                  208.      370
                  209.      371
                  210.      372
                  211.      373
                  212.      374
                  213.      375
                  214.      376
                  215.      377
                  216.      378


 n o d a l   p o i n t   v e l o c i t i e s

    node id       x-vel         y-vel         z-vel
          1   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          2   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          3   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          4   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          5   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          6   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          7   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          8   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
          9   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
         10   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
         11   -1.5640E+04   0.0000E+00    0.0000E+00    0.00
```

Anchor name: New Anchor 3

Origin: Start of File

Type: Plain search

Text to search for:

Direction: ● Forward ○ Backward

Match: ● Anywhere ○ Start of line only ○ End of line only

Skip over 0 matches

Relative location: 0 lines, 0 characters

Column separators: □ Tab □ Space ☒ Whitespace □ , □ ; □ | □ Other:

□ Move to start of line

Figure 7-1: GenEx dialog

Options

When an anchor or an entity is selected, it is possible to change the options shown in the dialog box. A new search will be performed whenever an option is changed that requires it. The only exception is the **Text to search for**, this requires the user to hit **Enter** (on the keyboard) to start the new search.

Table 7-1: Options

Option	**Description**
Origin	This is the parent anchor of the anchor/entity.
Column separator	If columns are selected in **Relative positions** it is possible to change what separates the columns in the input file.

Options specific for anchors

Table 7-2: Options specific for anchors

Option	Description
Type	There are four types of searches. Three of them are keyword-based (search-phrase based). ***Plain text:*** This is the most basic search. The search looks for the given text in the file and positions the anchor in front of the match. ***Glob search:*** The main goal of the glob search is to be able to match the strings with the aid of the wild cards, '*' and '?'. The asterisk matches any character any number of times and the question mark matches any character one time. ***Regular expression search:*** The asterisk * matches the preceding element zero or more times and the dot **.** matches any character one time. If letters are put inside brackets this matches any single character inside the brackets. If a '^' is put inside the brackets this means that we should match any character not inside the brackets. ***Absolute search:*** In this search the user positions the anchor simply by specifying the row and the column at which the anchor should be positioned in the file.
Text to search for	This is the text/regular expression/glob to search for.
Direction	Starting from the origin, this is the direction to search in.
Match	This is where on the line the search text will have to match.
Skip over	Since the input file can contain several instances of the search term it is possible to skip some of them to find the desired position.
Relative Location	When **Absolute search** is selected, this section will be enabled. Here it is possible to enter the absolute position of the anchor if known.
Move to start of line	When this is checked the anchor will be positioned at the start of the line, even if it is found somewhere else.

Examples

Glob search

*abc

will match any word ending with abc (xxxabc, yyyabc, etc.) and the anchor will be placed where the match begins ((A)xxxabc, (A)yyyabc).

a?c

will match all three letter words starting with 'a' and ending with 'c' (axc, a5c, etc.) and the anchor will be placed before the match begins ((A)axc, (A)a5c).

Regular expression search

ab*c

matches "ac", "abc", "abbbc", etc.

a.c

matches all three letter strings starting with 'a' and ending with'c' (ahc, a8c, aHc, etc.)

[csad]bc

matches all strings starting with c, s, a or d followed by 'bc' (cbc, sbc, abc, and dbc).

[^csad]bc

matches all strings not starting with c, s, a or d followed by 'bc' (xbc, 5bc, kbc, etc.).

These can all be combined into a larger regular expression,"[skjfrdzh]*esp[ohjd]n.e" will match "response" (but also "espdn1e" for example).

Plain text, *glob* and *regular* expression search searches for a specific text string. The absolute search positions the anchor relative to the parent. The *glob* and *regular expression* searches are very similar to the search capabilities in the Perl language or the Unix/Linux scripting language.

Options specific for entities

Table 7-3: Options specific for entities

Option	Description
Type of entity	Here there are three options, scalar, column vector and repeated anchor vector
Scalar	The scalar entity is used for extracting responses and it extracts one result
Column vector	A column vector extracts a column of data
Repeated anchor vector	A repeated anchor vector repeats the search of the selected anchor to extract several entities found in different places in the input file
Number format	Here it is possible to specify what a number looks like

Relative Location	This is the position of the entity relative to the parent anchor
Maximum length	The default behavior is that an entity starts at the specified position and ends with a white space. Here it is possible to specify the length of the entity if this is not the case.
Number of entities	When using GenEx to extract histories the default behavior is to keep extracting until a match is not found, this option limits the number of extracted results
Stopping anchor	An anchor can be defined as a stopping criterion if the number of components of a column vector is unknown
Anchor to repeat	If the entity type is "repeated anchor vector" this will show a menu with valid anchors. Start of file and End of file will not be available since they cannot be repeated.

7.2. Creating a .g6 file for LS-OPT

First we have to select the input file in which to search. This is done from the **File** menu: **Select input file**. The file will be displayed in the middle window of the application.

Creating an anchor or entity

There are three ways to create anchors or entities. The first is to select the anchor used as parent and then click on the anchor or entity button in the menu depending on what is needed. This will create a new uninitiated child. By selecting the new anchor or entity in the tree view on the left side, the options will be visible on the right side panel.

The second way is to simply make a selection in the text file, right click and select **Create Anchor Here** or **Create Entity Here**. This will create a new child at that position with the currently selected anchor as the parent anchor. It is also possible to select a column of numbers from the text file to create a column vector. The column entity uses white space as the delimiter.

The third option is to make a selection in the text and drag that selection to the anchor we want to use as parent in the tree.

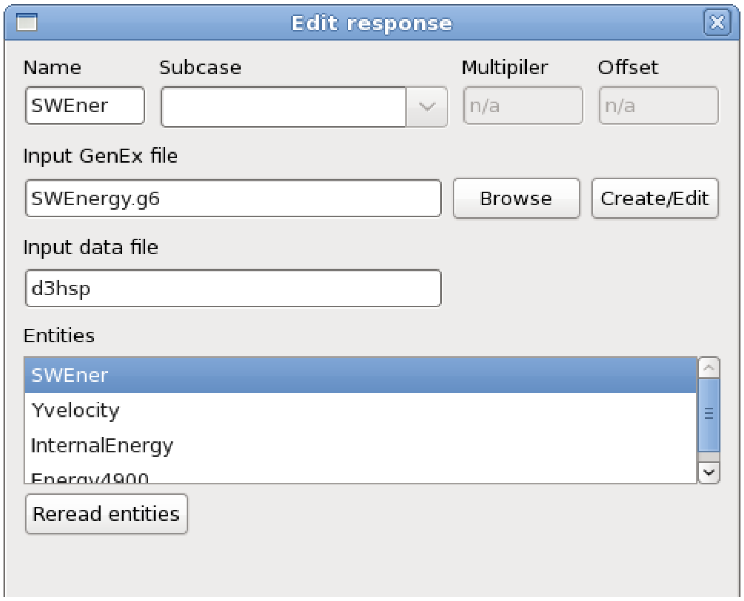
Creating a .g6 file without an input file

It is possible to create a .g6 file without access to the input file we want to extract from. However, this requires some knowledge of the file format and syntax.

Editing a .g6 file

From the "File" menu, select "Open GenEx file".

7.3. How to use GenEx from LS-OPT for extracting responses



Name: SWEner
Subcase:
Multipiler: n/a
Offset: n/a
Input GenEx file: SWEnergy.g6
Input data file: d3hsp
Entities:
SWEner
Yvelocity
InternalEnergy
Energy4900

Figure 7-2: Definition of a GenEx Response

From the **Responses** panel select GENEX as a response. This will open up a dialog showing a few options related to GenEx .

The first selection to be made is which `.g6` file to use. This option provides a list of available entities to choose from. The entities need to be of the "Scalar" type. It is also possible to edit a file by clicking the **Create/Edit** button. If no file name is given the default action is to create a new `.g6` file.

Secondly, enter the name of the input data file from which the responses are to be extracted. LS-OPT will look for this file in each of the run directories.

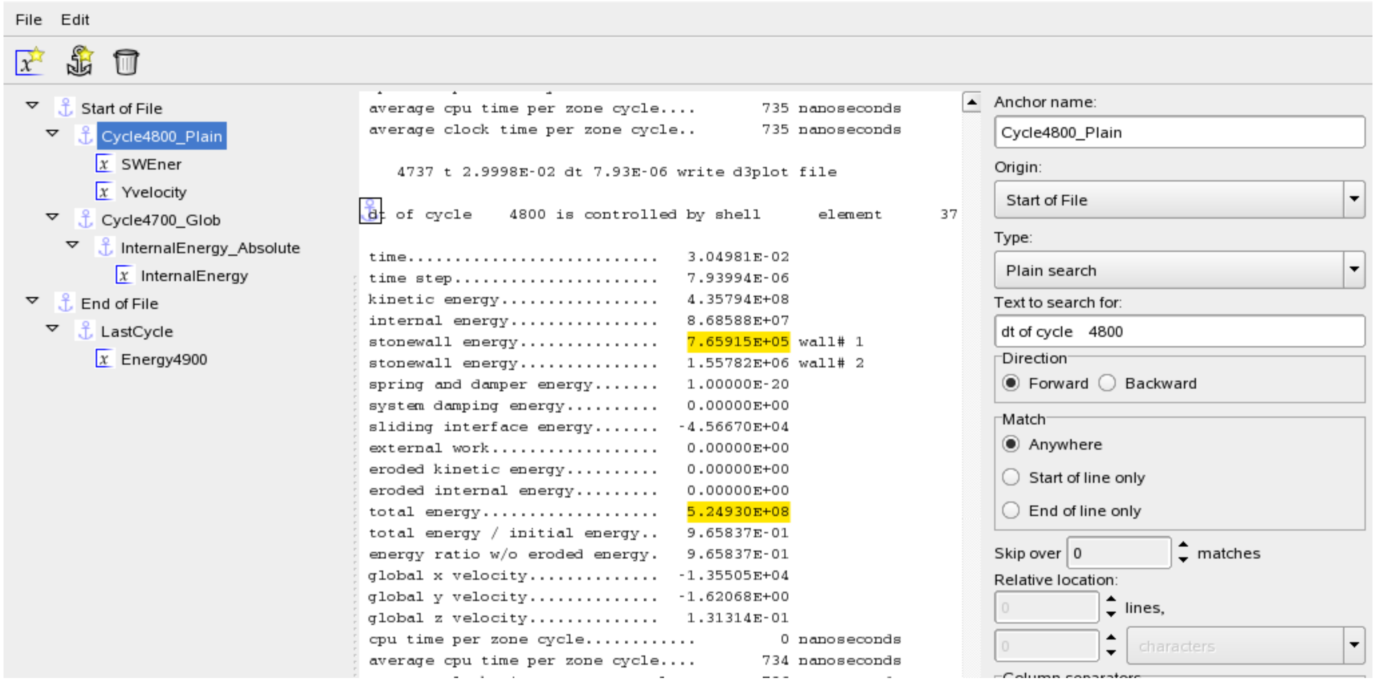
7.3.1. An example using GenEx to extract responses

This example explains how to extract a number of responses from the LS-DYNA `d3hsp` file. Different search options are employed to demonstrate the various options.

- Open the GenEx GUI by selecting **Create/Edit**. Then select d3hsp as the input file by using **File→Select input file**. The d3hsp file is displayed in the middle. We are interested in 3 responses at various cycles and a fourth response to be the last one in the file.

Defining an anchor:

- Define an anchor with the name `Cycle4800_Plain` by clicking on the anchor icon or using the **Edit** option.
- Use a plain search to search for the string "`dt of cycle      4800`". If you want to change the string in the text box, remember to hit the "Enter" key on the keyboard. The anchor is displayed as a small anchor icon in the leftmost column of the line that matches the search string. The next step would be to find the desired field relative to this anchor.



- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
average cpu time per zone cycle....       735 nanoseconds
average clock time per zone cycle..       735 nanoseconds

    4737 t 2.9998E-02 dt 7.93E-06 write d3plot file

dt of cycle    4800 is controlled by shell      element      37

time..........................    3.04981E-02
time step.....................    7.93994E-06
kinetic energy................    4.35794E+08
internal energy...............    8.68588E+07
stonewall energy..............    7.65915E+05 wall# 1
stonewall energy..............    1.55782E+06 wall# 2
spring and damper energy......    1.00000E-20
system damping energy.........    0.00000E+00
sliding interface energy......   -4.56670E+04
external work.................    0.00000E+00
eroded kinetic energy.........    0.00000E+00
eroded internal energy........    0.00000E+00
total energy..................    5.24930E+08
total energy / initial energy..   9.65837E-01
energy ratio w/o eroded energy.   9.65837E-01
global x velocity.............   -1.35505E+04
global y velocity.............   -1.62068E+00
global z velocity.............    1.31314E-01
cpu time per zone cycle...........          0 nanoseconds
average cpu time per zone cycle....       734 nanoseconds
```

Anchor name: Cycle4800_Plain

Origin: Start of File

Type: Plain search

Text to search for: dt of cycle   4800

Direction: ● Forward ○ Backward

Match: ● Anywhere ○ Start of line only ○ End of line only

Skip over 0 matches

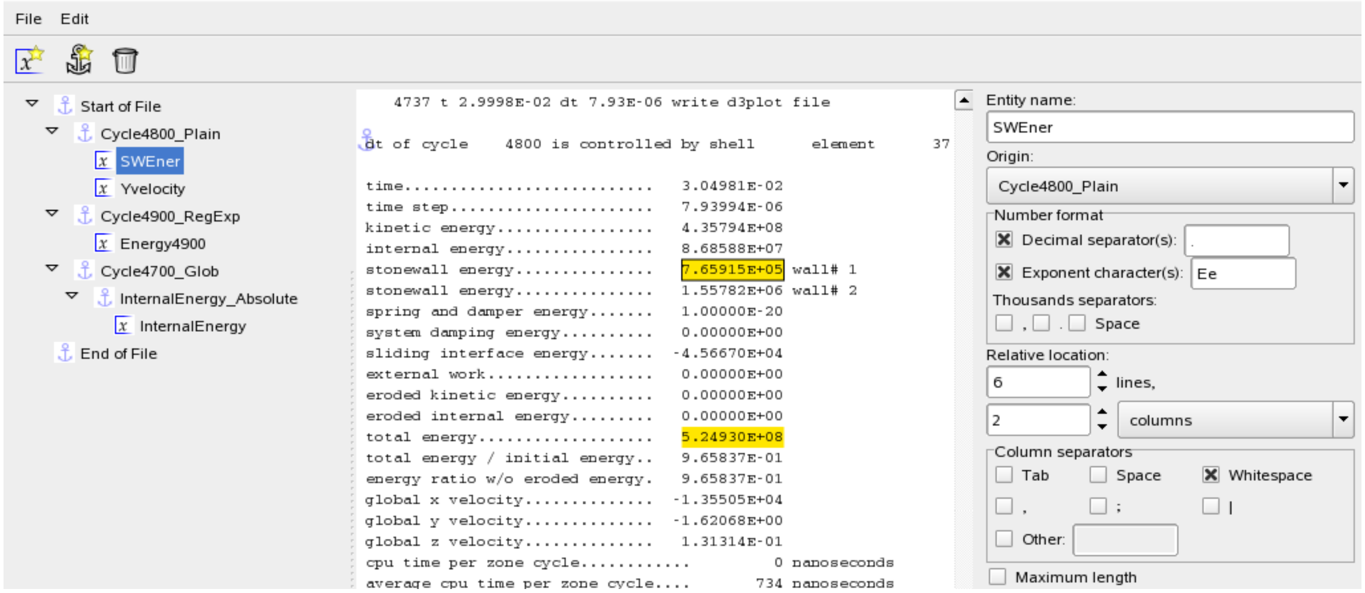
Relative location: 0 lines, 0 characters

Column separators

Figure 7-3: GenEx dialog; definition of an anchor

Defining an entity:

- Define a new entity `SWEner` by using the leftmost ***x***-icon or the **Edit** option.
- Choose the previously defined anchor as the Origin.
- Find the desired field by searching 6 lines below the anchor, 2 columns across. The desired field is displayed as highlighted in yellow with a black border. See figure below.



- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4900_RegExp
    - Energy4900
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File

```
    4737 t 2.9998E-02 dt 7.93E-06 write d3plot file

dt of cycle    4800 is controlled by shell      element      37

time..........................    3.04981E-02
time step.....................    7.93994E-06
kinetic energy................    4.35794E+08
internal energy...............    8.68588E+07
stonewall energy..............    7.65915E+05 wall# 1
stonewall energy..............    1.55782E+06 wall# 2
spring and damper energy......    1.00000E-20
system damping energy.........    0.00000E+00
sliding interface energy......   -4.56670E+04
external work.................    0.00000E+00
eroded kinetic energy.........    0.00000E+00
eroded internal energy........    0.00000E+00
total energy..................    5.24930E+08
total energy / initial energy..   9.65837E-01
energy ratio w/o eroded energy.   9.65837E-01
global x velocity.............   -1.35505E+04
global y velocity.............   -1.62068E+00
global z velocity.............    1.31314E-01
cpu time per zone cycle...........          0 nanoseconds
average cpu time per zone cycle....       734 nanoseconds
```

Entity name: SWEner

Origin: Cycle4800_Plain

Number format
☒ Decimal separator(s): .
☒ Exponent character(s): Ee
Thousands separators: ☐ , ☐ . ☐ Space

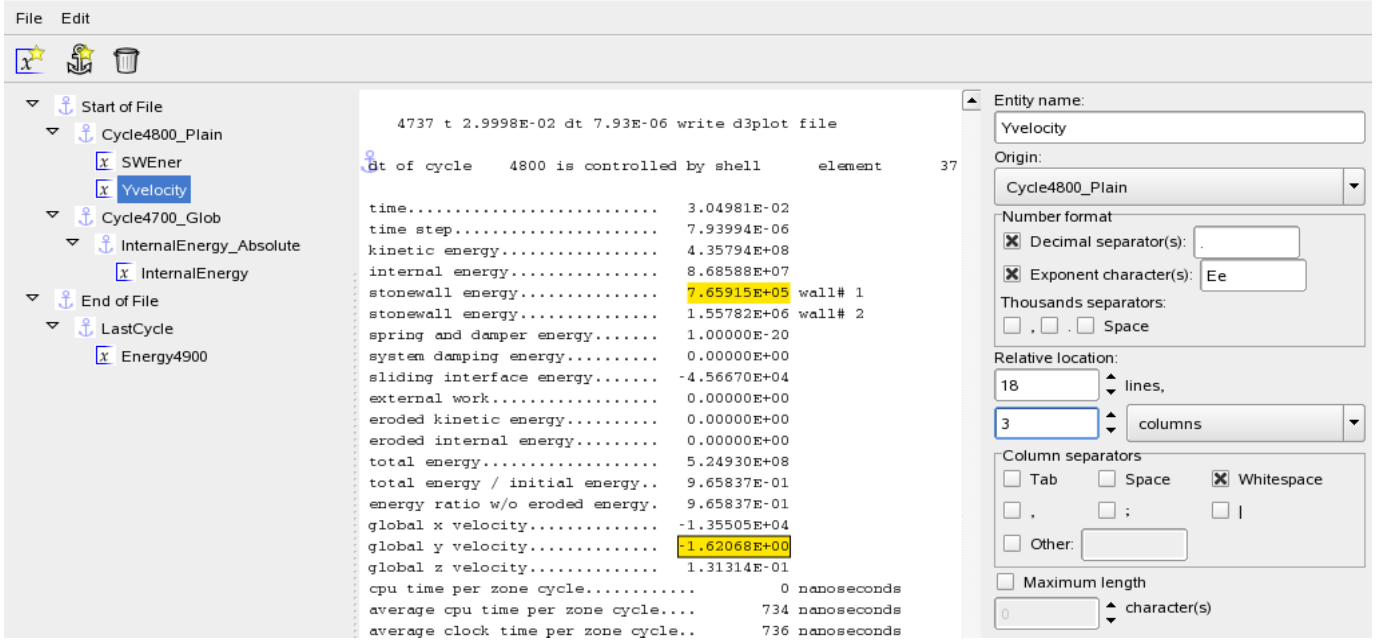
Relative location: 6 lines, 2 columns

Column separators
☐ Tab ☐ Space ☒ Whitespace
☐ , ☐ ; ☐ |
☐ Other:

☐ Maximum length

Figure 7-4: GenEx dialog; definition of an entity

- Now define a new entity referred to the same anchor `Cycle4800_Plain`. This entity is 18 lines below the anchor and 3 columns across as shown in the **Relative location** dialog below:



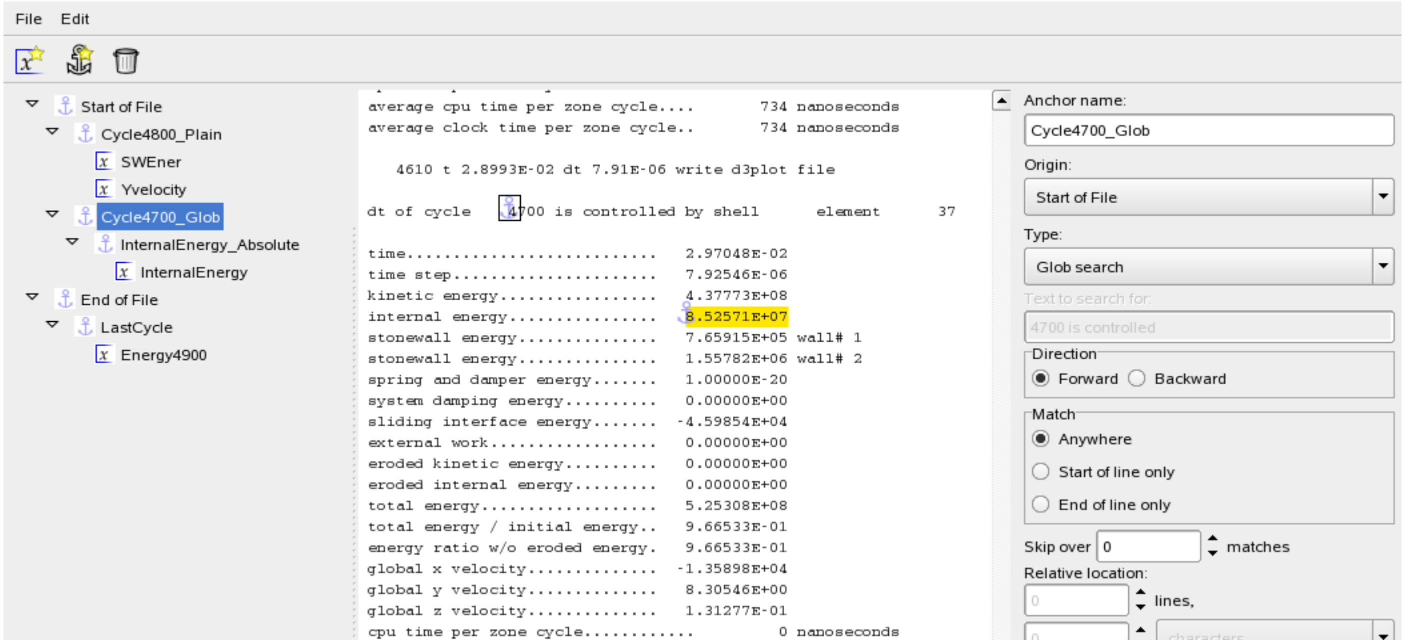
- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
   4737 t 2.9998E-02 dt 7.93E-06 write d3plot file

 dt of cycle    4800 is controlled by shell      element      37

 time..........................   3.04981E-02
 time step.....................   7.93994E-06
 kinetic energy................   4.35794E+08
 internal energy...............   8.68588E+07
 stonewall energy..............   7.65915E+05 wall# 1
 stonewall energy..............   1.55782E+06 wall# 2
 spring and damper energy.......  1.00000E-20
 system damping energy.........   0.00000E+00
 sliding interface energy....... -4.56670E+04
 external work.................   0.00000E+00
 eroded kinetic energy.........   0.00000E+00
 eroded internal energy........   0.00000E+00
 total energy..................   5.24930E+08
 total energy / initial energy..  9.65837E-01
 energy ratio w/o eroded energy.  9.65837E-01
 global x velocity............. -1.35505E+04
 global y velocity............. -1.62068E+00
 global z velocity.............   1.31314E-01
 cpu time per zone cycle...........          0 nanoseconds
 average cpu time per zone cycle....       734 nanoseconds
 average clock time per zone cycle..       736 nanoseconds
```

Entity name: Yvelocity
Origin: Cycle4800_Plain
Number format
☒ Decimal separator(s): .
☒ Exponent character(s): Ee
Thousands separators:
☐ , ☐ . ☐ Space
Relative location:
18 lines,
3 columns
Column separators
☐ Tab ☐ Space ☒ Whitespace
☐ , ☐ ; ☐ |
☐ Other:
☐ Maximum length
0 character(s)

Figure 7-5: GenEx dialog; definition of an entity

- Define a second anchor using a global search for the string "`4700 is controlled`". The origin of this anchor is also the start of the file and the search is forward from that point. Note the anchor placement on the figure below just before the string "`4700 is controlled`".



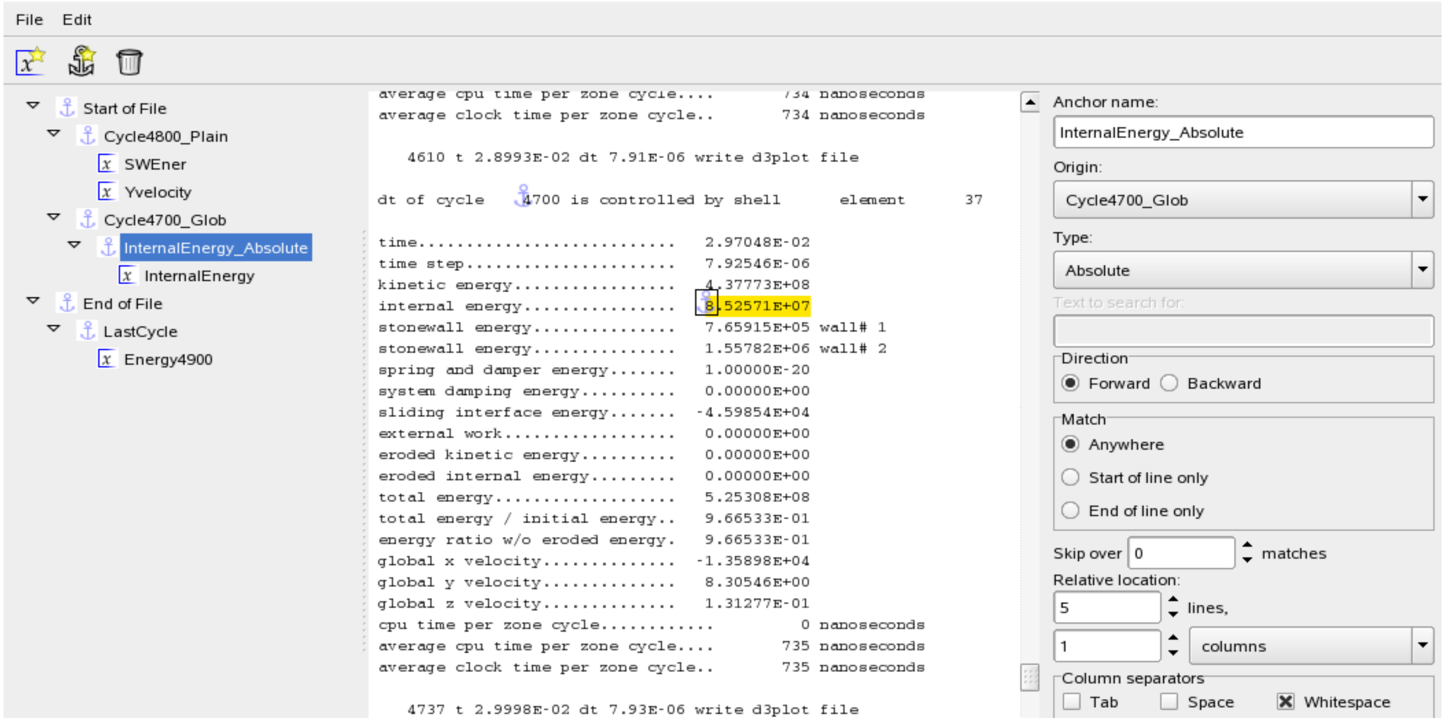
- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
 average cpu time per zone cycle....       734 nanoseconds
 average clock time per zone cycle..       734 nanoseconds

   4610 t 2.8993E-02 dt 7.91E-06 write d3plot file

 dt of cycle    4700 is controlled by shell      element      37

 time..........................   2.97048E-02
 time step.....................   7.92546E-06
 kinetic energy................   4.37773E+08
 internal energy...............   8.52571E+07
 stonewall energy..............   7.65915E+05 wall# 1
 stonewall energy..............   1.55782E+06 wall# 2
 spring and damper energy.......  1.00000E-20
 system damping energy.........   0.00000E+00
 sliding interface energy....... -4.59854E+04
 external work.................   0.00000E+00
 eroded kinetic energy.........   0.00000E+00
 eroded internal energy........   0.00000E+00
 total energy..................   5.25308E+08
 total energy / initial energy..  9.66533E-01
 energy ratio w/o eroded energy.  9.66533E-01
 global x velocity............. -1.35898E+04
 global y velocity.............   8.30546E+00
 global z velocity.............   1.31277E-01
 cpu time per zone cycle...........          0 nanoseconds
```

Anchor name: Cycle4700_Glob
Origin: Start of File
Type: Glob search
Text to search for: 4700 is controlled
Direction
● Forward ○ Backward
Match
● Anywhere
○ Start of line only
○ End of line only
Skip over 0 matches
Relative location:
0 lines,
0 characters

Figure 7-6: GenEx dialog; definition of an anchor

- Now define an anchor `InternalEnergy_Absolute` relative to the previous anchor by setting the origin as `Cycle4700_Glob`, then searching 5 lines down and one column across. Note the anchor icon just before the yellow-highlighted number in the figure below.



- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
average cpu time per zone cycle....        734 nanoseconds
average clock time per zone cycle..        734 nanoseconds

   4610 t 2.8993E-02 dt 7.91E-06 write d3plot file

dt of cycle     4700 is controlled by shell      element      37

time..........................   2.97048E-02
time step.....................   7.92546E-06
kinetic energy................   4.37773E+08
internal energy...............   8.52571E+07
stonewall energy..............   7.65915E+05 wall# 1
stonewall energy..............   1.55782E+06 wall# 2
spring and damper energy......   1.00000E-20
system damping energy.........   0.00000E+00
sliding interface energy......  -4.59854E+04
external work.................   0.00000E+00
eroded kinetic energy.........   0.00000E+00
eroded internal energy........   0.00000E+00
total energy..................   5.25308E+08
total energy / initial energy..  9.66533E-01
energy ratio w/o eroded energy.  9.66533E-01
global x velocity.............  -1.35898E+04
global y velocity.............   8.30546E+00
global z velocity.............   1.31277E-01
cpu time per zone cycle...........           0 nanoseconds
average cpu time per zone cycle....        735 nanoseconds
average clock time per zone cycle..        735 nanoseconds

   4737 t 2.9998E-02 dt 7.93E-06 write d3plot file
```

Anchor name: InternalEnergy_Absolute

Origin: Cycle4700_Glob

Type: Absolute

Text to search for:

Direction: ● Forward ○ Backward

Match: ● Anywhere ○ Start of line only ○ End of line only

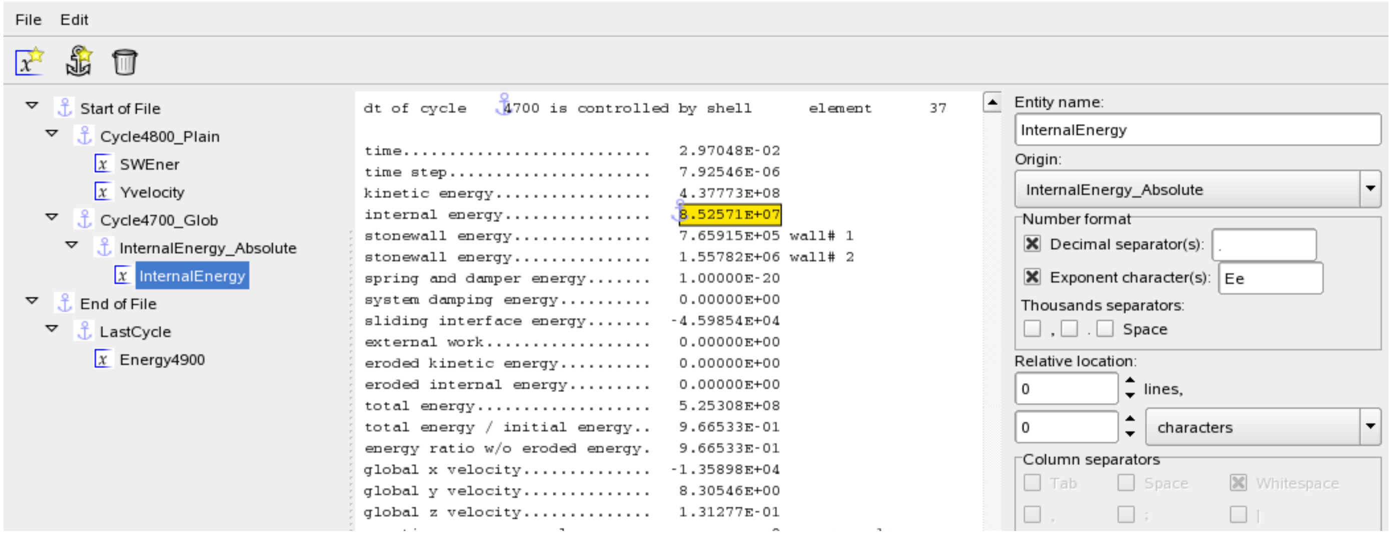
Skip over 0 matches

Relative location: 5 lines, 1 columns

Column separators: □ Tab □ Space ☒ Whitespace

Figure 7-7: GenEx dialog; definition of an anchor

- Define a new entity `InternalEnergy` using the `InternalEnergy_Absolute` anchor as reference point. The desired field is immediately found since the anchor is already at the desired location.



- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
dt of cycle     4700 is controlled by shell      element      37

time..........................   2.97048E-02
time step.....................   7.92546E-06
kinetic energy................   4.37773E+08
internal energy...............   8.52571E+07
stonewall energy..............   7.65915E+05 wall# 1
stonewall energy..............   1.55782E+06 wall# 2
spring and damper energy......   1.00000E-20
system damping energy.........   0.00000E+00
sliding interface energy......  -4.59854E+04
external work.................   0.00000E+00
eroded kinetic energy.........   0.00000E+00
eroded internal energy........   0.00000E+00
total energy..................   5.25308E+08
total energy / initial energy..  9.66533E-01
energy ratio w/o eroded energy.  9.66533E-01
global x velocity.............  -1.35898E+04
global y velocity.............   8.30546E+00
global z velocity.............   1.31277E-01
```

Entity name: InternalEnergy

Origin: InternalEnergy_Absolute

Number format: ☒ Decimal separator(s): . ☒ Exponent character(s): Ee

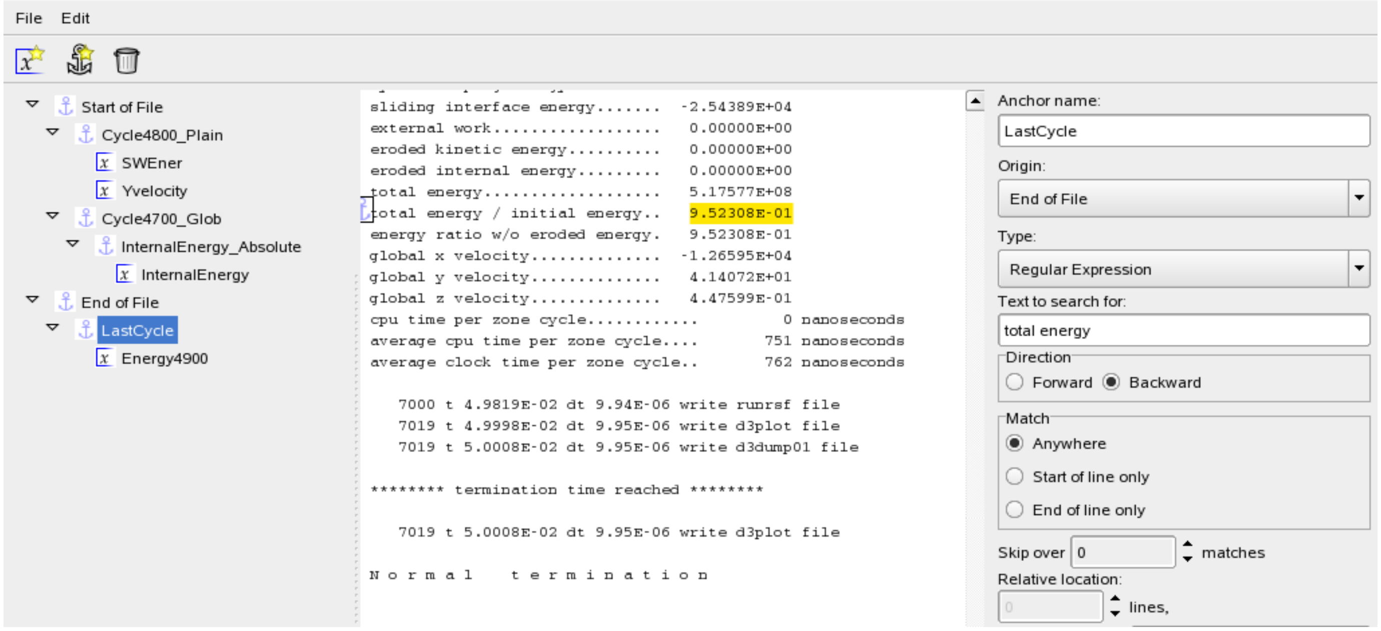
Thousands separators: □ , □ . □ Space

Relative location: 0 lines, 0 characters

Column separators: □ Tab □ Space ☒ Whitespace □ , □ ; □ |

Figure 7-8: GenEx dialog; definition of an entity

- The next desired entity is the final total energy ratio (i.e. the one in the last cycle in the file). In this case we will set the reference anchor called `LastCycle` to be the end of the file (Origin) and search backwards (Direction).
- The search string is "`total energy`" and the regular expression search type is used. The settings to find the anchor are shown below.



- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
sliding interface energy.......  -2.54389E+04
external work..................   0.00000E+00
eroded kinetic energy..........   0.00000E+00
eroded internal energy.........   0.00000E+00
total energy...................   5.17577E+08
total energy / initial energy..   9.52308E-01
energy ratio w/o eroded energy.   9.52308E-01
global x velocity..............  -1.26595E+04
global y velocity..............   4.14072E+01
global z velocity..............   4.47599E-01
cpu time per zone cycle............         0 nanoseconds
average cpu time per zone cycle....       751 nanoseconds
average clock time per zone cycle..       762 nanoseconds

    7000 t 4.9819E-02 dt 9.94E-06 write runrsf file
    7019 t 4.9998E-02 dt 9.95E-06 write d3plot file
    7019 t 5.0008E-02 dt 9.95E-06 write d3dump01 file

******** termination time reached ********

    7019 t 5.0008E-02 dt 9.95E-06 write d3plot file

Normal    termination
```

Anchor name: LastCycle

Origin: End of File

Type: Regular Expression

Text to search for: total energy

Direction: ○ Forward ● Backward

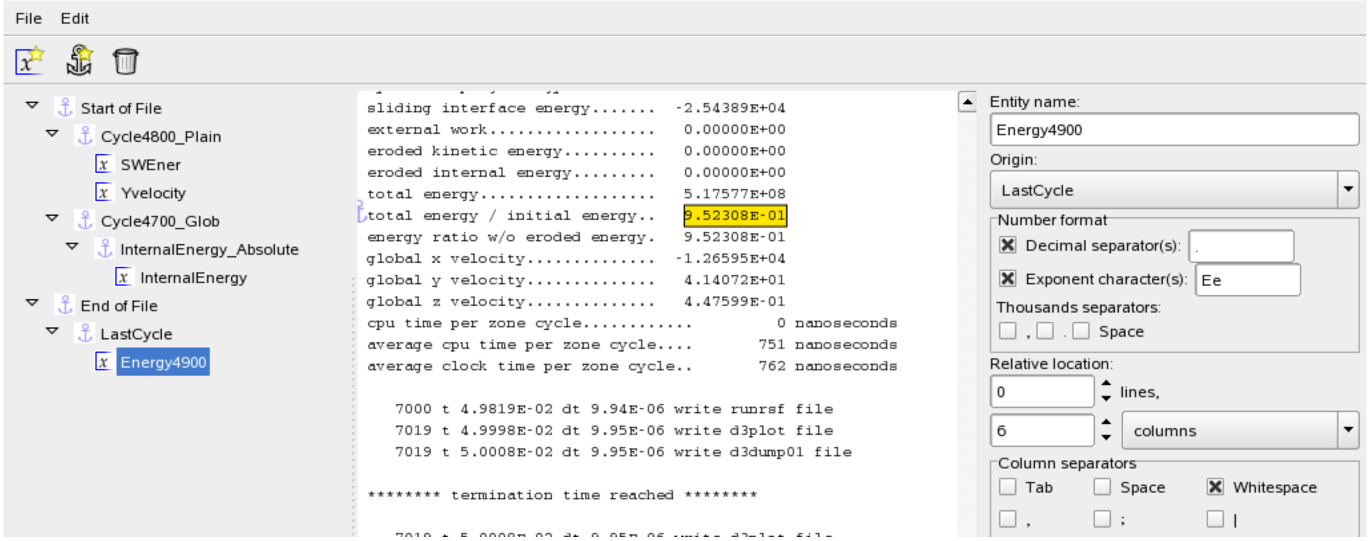
Match: ● Anywhere ○ Start of line only ○ End of line only

Skip over 0 matches

Relative location: 0 lines,

Figure 7-9: GenEx dialog; definition of an anchor

- The entity is found by using `LastCycle` as the anchor and searching in the sixth column. See **relative location** dialog box below.



- Start of File
  - Cycle4800_Plain
    - SWEner
    - Yvelocity
  - Cycle4700_Glob
    - InternalEnergy_Absolute
      - InternalEnergy
- End of File
  - LastCycle
    - Energy4900

```
sliding interface energy.......  -2.54389E+04
external work..................   0.00000E+00
eroded kinetic energy..........   0.00000E+00
eroded internal energy.........   0.00000E+00
total energy...................   5.17577E+08
total energy / initial energy..   9.52308E-01
energy ratio w/o eroded energy.   9.52308E-01
global x velocity..............  -1.26595E+04
global y velocity..............   4.14072E+01
global z velocity..............   4.47599E-01
cpu time per zone cycle............         0 nanoseconds
average cpu time per zone cycle....       751 nanoseconds
average clock time per zone cycle..       762 nanoseconds

    7000 t 4.9819E-02 dt 9.94E-06 write runrsf file
    7019 t 4.9998E-02 dt 9.95E-06 write d3plot file
    7019 t 5.0008E-02 dt 9.95E-06 write d3dump01 file

******** termination time reached ********
```

Entity name: Energy4900

Origin: LastCycle

Number format: ☒ Decimal separator(s): . ☒ Exponent character(s): Ee

Thousands separators: □ , □ . □ Space

Relative location: 0 lines, 6 columns

Column separators: □ Tab □ Space ☒ Whitespace □ , □ ; □ |

Figure 7-10: GenEx dialog; definition of an entity

- This completes the GenEx setup. Save the file.
- Now open the **Stage** dialog on the **Responses** page and select the **GENEX** response type on the right. Open the Input GenEx file. A browse option is available. Importing the file will display the selected entities in the **Entities** box.
- Select the input data file, namely `d3hsp`. This file must be available in the run directory during the LS-OPT run.
- Select an entity, define a response name at the top of the dialog and hit **Ok**. The response will appear in the list on the **Responses** page.
- Repeat the procedure for the remaining three response entities.

LS-OPT can now be run and the response entities will be extracted for each simulation run.

7.4. Extracting histories

7.4.1. An example using "Repeated anchor vector" to extract histories

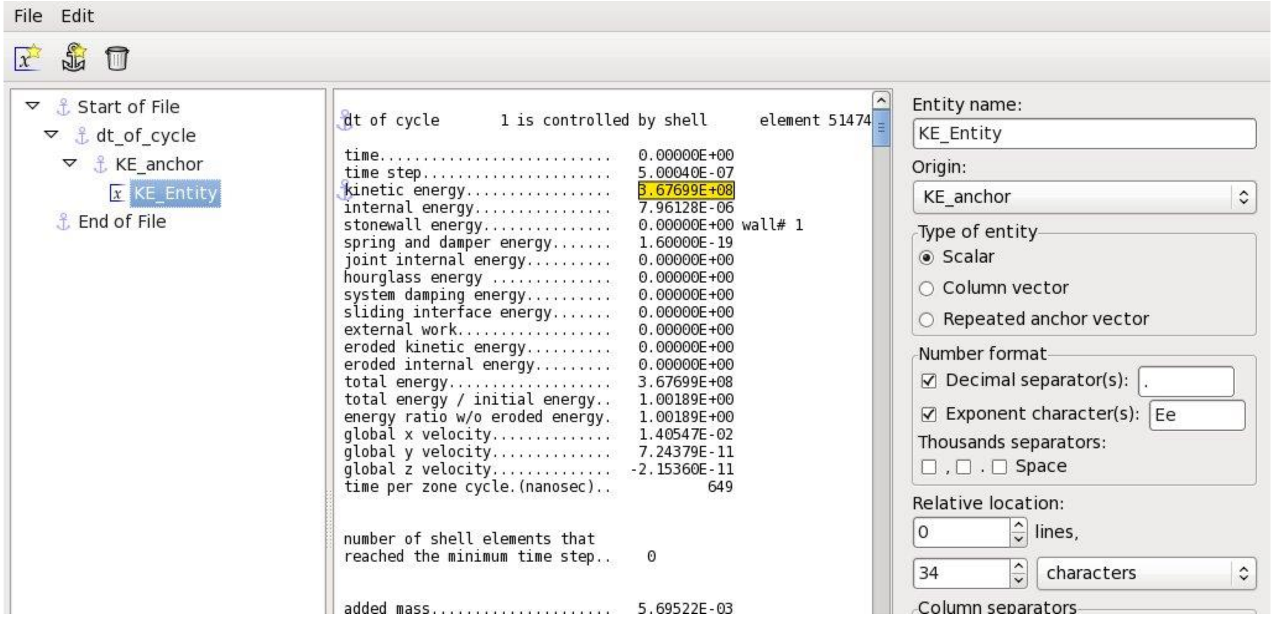
In this example we will use GenEx to extract histories of the value for "kinetic energy" in the "glstat" file created by LS-DYNA. We first start by creating the anchor `dt_of_cycles`. This anchor will be the base for further anchors. With this anchor as parent we now create the `KE_anchor` to search for the string we are looking for, in this case "`kinetic energy`".

As seen in Figure 7-11, this entity is of the Scalar type and needs to be changed to **Repeated anchor vector**. When creating a repeated anchor vector the default value for **Anchor to repeat** is the parent of the entity. Since "`kinetic energy`" appears twice between every `dt_of_cycle` the result is not what we want yet. In order to skip "eroded kinetic entity", we pick the grandparent `dt_of_cycle` anchor as the one to repeat.

The result of this setup will be that the extractor will find "`dt_of_cycle`", then search forward for "`kinetic energy`" and extract the first element of the vector. Then, it will find the next occurrence of "`dt_of_cycle`" and repeat, extracting the other elements of the vector.

After we have changed the **Anchor to repeat** to `dt_of_cycle`, we will have the correct result. The color of the other vector elements will be in light yellow with a dotted border, Figure 7-12.

We are now finished with the GenEx part and the file can be saved.



- Start of File
  - dt_of_cycle
    - KE_anchor
      - KE_Entity
- End of File

```
dt of cycle      1 is controlled by shell      element 51474

time..........................   0.00000E+00
time step.....................   5.00040E-07
kinetic energy................   3.67699E+08
internal energy...............   7.96128E-06
stonewall energy..............   0.00000E+00 wall# 1
spring and damper energy......   1.60000E-19
joint internal energy.........   0.00000E+00
hourglass energy .............   0.00000E+00
system damping energy.........   0.00000E+00
sliding interface energy......   0.00000E+00
external work.................   0.00000E+00
eroded kinetic energy.........   0.00000E+00
eroded internal energy........   0.00000E+00
total energy..................   3.67699E+08
total energy / initial energy..  1.00189E+00
energy ratio w/o eroded energy.  1.00189E+00
global x velocity.............   1.40547E-02
global y velocity.............   7.24379E-11
global z velocity.............  -2.15360E-11
time per zone cycle.(nanosec)..          649


number of shell elements that
reached the minimum time step..    0


added mass....................   5.69522E-03
```

Entity name: KE_Entity

Origin: KE_anchor

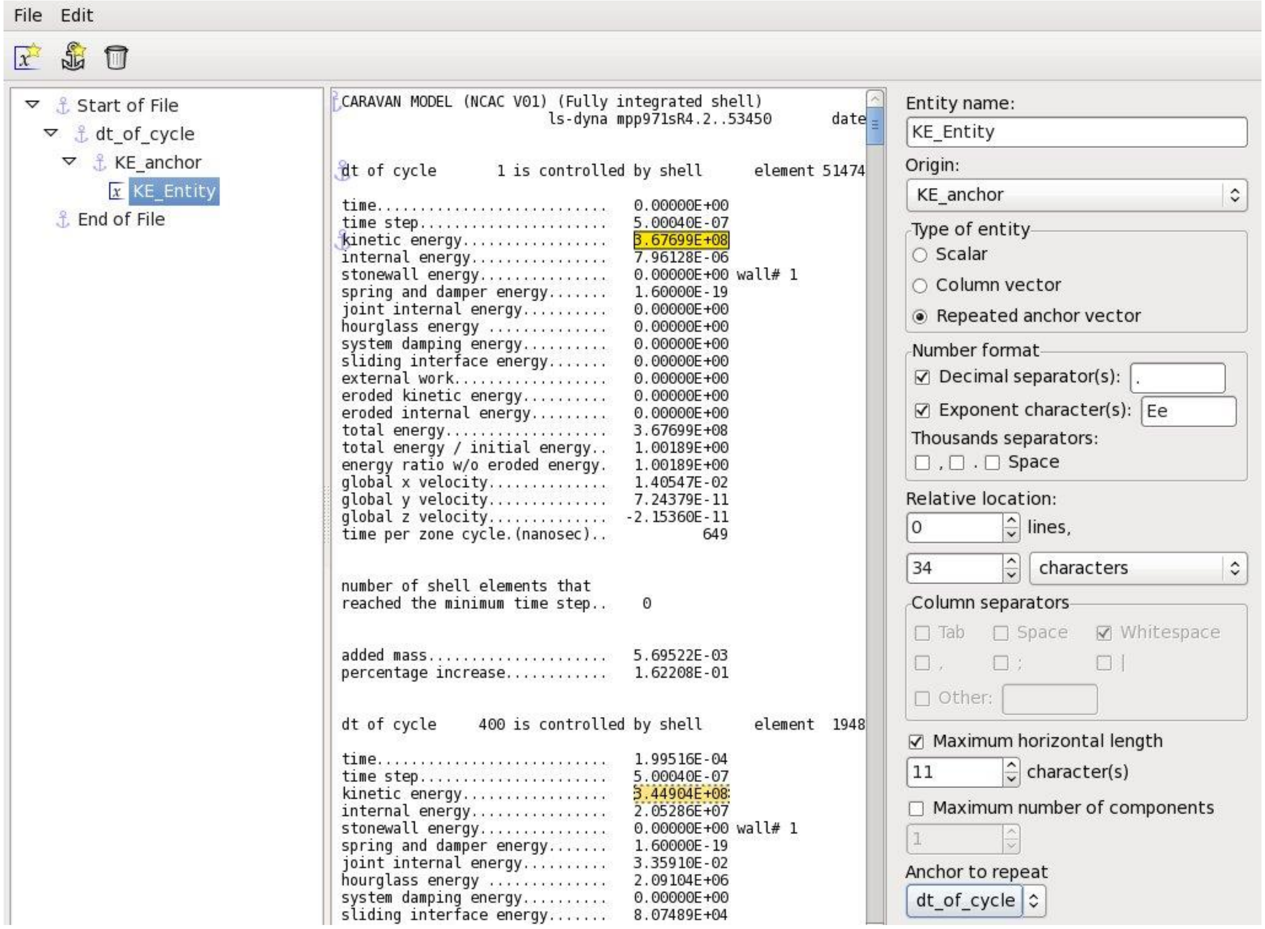
Type of entity
● Scalar
○ Column vector
○ Repeated anchor vector

Number format
☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

Relative location:
0 lines,
34 characters

Column separators

Figure 7-11: GenEx dialog; definition of an entity



- Start of File
  - dt_of_cycle
    - KE_anchor
      - KE_Entity
- End of File

```
CARAVAN MODEL (NCAC V01) (Fully integrated shell)
                        ls-dyna mpp971sR4.2..53450      date

dt of cycle      1 is controlled by shell      element 51474

time..........................   0.00000E+00
time step.....................   5.00040E-07
kinetic energy................   3.67699E+08
internal energy...............   7.96128E-06
stonewall energy..............   0.00000E+00 wall# 1
spring and damper energy......   1.60000E-19
joint internal energy.........   0.00000E+00
hourglass energy .............   0.00000E+00
system damping energy.........   0.00000E+00
sliding interface energy......   0.00000E+00
external work.................   0.00000E+00
eroded kinetic energy.........   0.00000E+00
eroded internal energy........   0.00000E+00
total energy..................   3.67699E+08
total energy / initial energy..  1.00189E+00
energy ratio w/o eroded energy.  1.00189E+00
global x velocity.............   1.40547E-02
global y velocity.............   7.24379E-11
global z velocity.............  -2.15360E-11
time per zone cycle.(nanosec)..          649


number of shell elements that
reached the minimum time step..    0


added mass....................   5.69522E-03
percentage increase...........   1.62208E-01


dt of cycle    400 is controlled by shell      element  1948

time..........................   1.99516E-04
time step.....................   5.00040E-07
kinetic energy................   3.44904E+08
internal energy...............   2.05286E+07
stonewall energy..............   0.00000E+00 wall# 1
spring and damper energy......   1.60000E-19
joint internal energy.........   3.35910E-02
hourglass energy .............   2.09104E+06
system damping energy.........   0.00000E+00
sliding interface energy......   8.07489E+04
```

Entity name: KE_Entity

Origin: KE_anchor

Type of entity
○ Scalar
○ Column vector
● Repeated anchor vector

Number format
☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

Relative location:
0 lines,
34 characters

Column separators
□ Tab □ Space ☑ Whitespace
□ , □ ; □ |
□ Other:

☑ Maximum horizontal length
11 character(s)

□ Maximum number of components
1

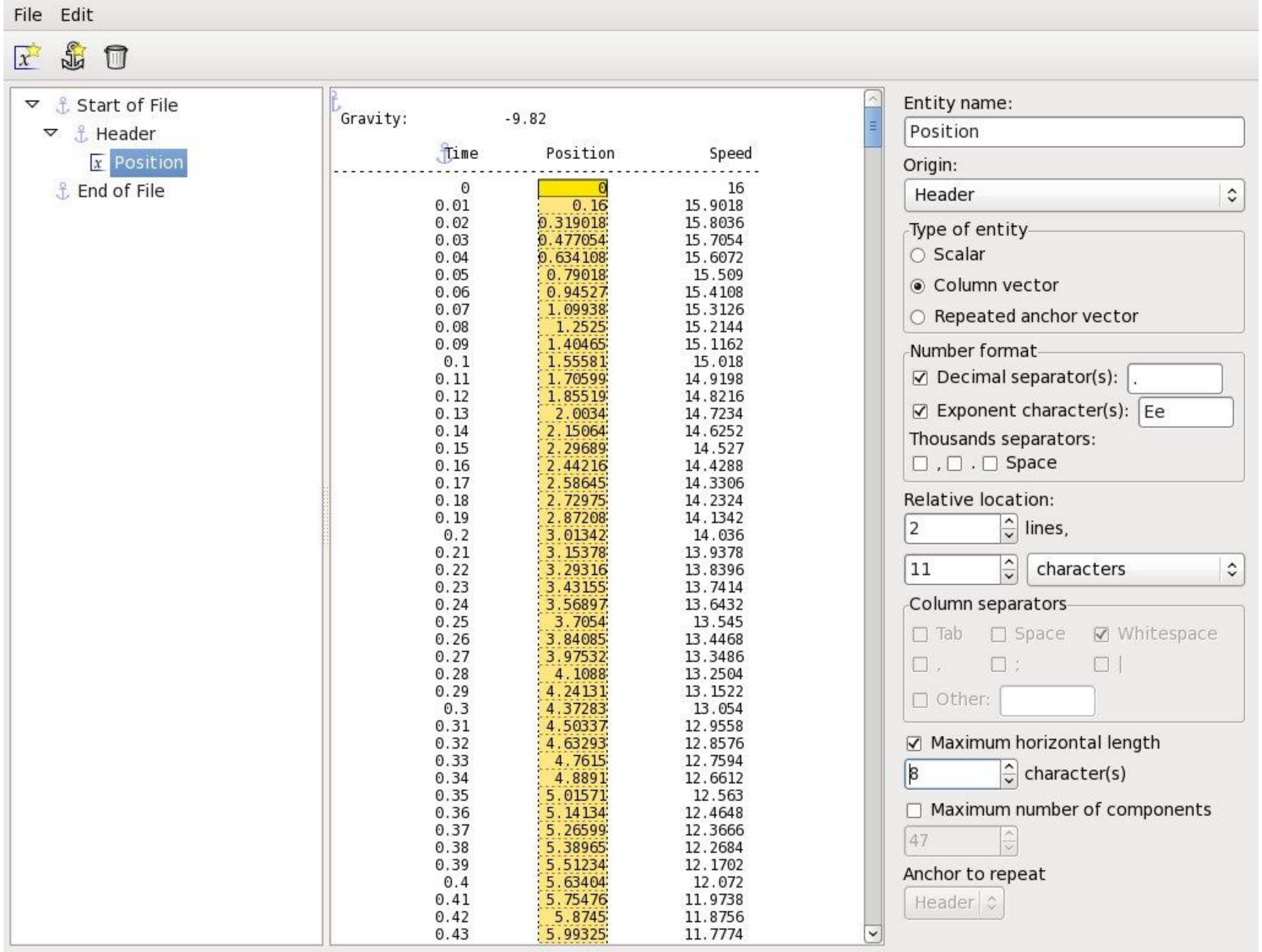
Anchor to repeat
dt_of_cycle

Figure 7-12: GenEx dialog; definition of a repeat anchor vector

7.4.2. An example using "Column vector" to extract histories

Column vectors are useful for extracting vectors in tables. In this example we extract a position vector generated by a fictitious solver. Just as in the previous example we start with the creation of the entity we want to be the first. We then change the type to **Column vector**.

It’s possible to create the vector by selecting a column in GenEx and right click to choose **New Entity**.



```
Gravity:          -9.82

          Time      Position          Speed
---------------------------------------------------
             0             0             16
          0.01          0.16        15.9018
          0.02      0.319018        15.8036
          0.03      0.477054        15.7054
          0.04      0.634108        15.6072
          0.05       0.79018         15.509
          0.06       0.94527        15.4108
          0.07       1.09938        15.3126
          0.08        1.2525        15.2144
          0.09       1.40465        15.1162
           0.1       1.55581         15.018
          0.11       1.70599        14.9198
          0.12       1.85519        14.8216
          0.13        2.0034        14.7234
          0.14       2.15064        14.6252
          0.15       2.29689         14.527
          0.16       2.44216        14.4288
          0.17       2.58645        14.3306
          0.18       2.72975        14.2324
          0.19       2.87208        14.1342
           0.2       3.01342         14.036
          0.21       3.15378        13.9378
          0.22       3.29316        13.8396
          0.23       3.43155        13.7414
          0.24       3.56897        13.6432
          0.25        3.7054         13.545
          0.26       3.84085        13.4468
          0.27       3.97532        13.3486
          0.28        4.1088        13.2504
          0.29       4.24131        13.1522
           0.3       4.37283         13.054
          0.31       4.50337        12.9558
          0.32       4.63293        12.8576
          0.33        4.7615        12.7594
          0.34        4.8891        12.6612
          0.35       5.01571         12.563
          0.36       5.14134        12.4648
          0.37       5.26599        12.3666
          0.38       5.38965        12.2684
          0.39       5.51234        12.1702
           0.4       5.63404         12.072
          0.41       5.75476        11.9738
          0.42        5.8745        11.8756
          0.43       5.99325        11.7774
```

Entity name: Position

Origin: Header

Type of entity
○ Scalar
● Column vector
○ Repeated anchor vector

Number format
☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

Relative location:
2 lines,
11 characters

Column separators
□ Tab □ Space ☑ Whitespace
□ , □ ; □ |
□ Other:

☑ Maximum horizontal length
8 character(s)

□ Maximum number of components
47

Anchor to repeat
Header

Figure 7-13: GenEx dialog; definition of a column vector entity

7.4.3. How to extract the histories from LS-OPT

Using GenEx for extracting histories is very similar to using it for responses. The main difference is that you have to select two entities to define the history, one for the x-axis and one for the y-axis. It’s possible to use "**Auto increment**" for the x-axis, in which case the x-axis values will simply be 0,1,2,3…

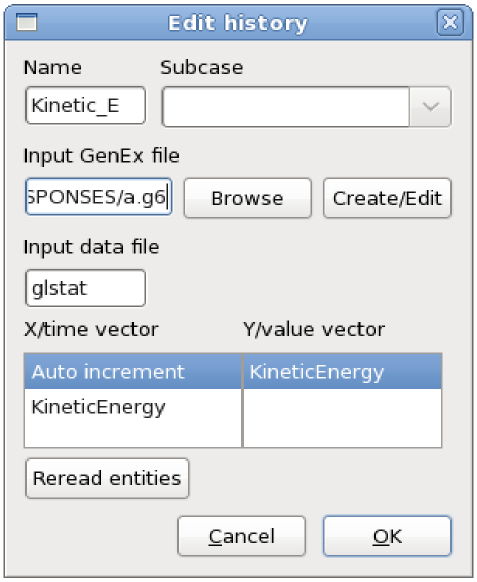


Figure 7-14: Interface to define a GenEx History

When creating the entities in GenEx they need to be either **Column vector** or **Repeated anchor vector** to be used for history extraction.

7.5. Command line options

GenEx can be started by selecting the ***Create/Edit*** button after selecting GenEx on the **Responses** or **Histories** page, or from the **File Multihistories** dialog, or the executable *genex* located in the LS-OPT installation directory may be called from the command line.

```
genex [-h] [-x] [-d] [-g <str>] [-i <str>] [-f <str>] [-v <str>] [-e1 <str>] [-
e2 <str>] [-c <str>] [file]
```

Table 16-3 explains the command line options.

Table 7-4: GenEx command line options

Option	Description
-h, --help	Show help message
-x, --extract	Extract
-d, --delete	Delete the file when program exits (be careful)
-g, --g6file=<str>	The g6-file used for entities
-i, --datafile=<str>	The data file containing the data
-f, --filename=<str>	Output to 'filename'

-v, --view=<str>	View a text file
-e1, --entity1=<str>	First entity in histories
-e2, --entity2=<str>	Second entity in histories
-c, --create=<str>	Create or Edit a GenEx file

7.6. Small car crashworthiness example using GenEx to extract histories/responses from data files

Refer to Section 20.9 for the GenEx example.

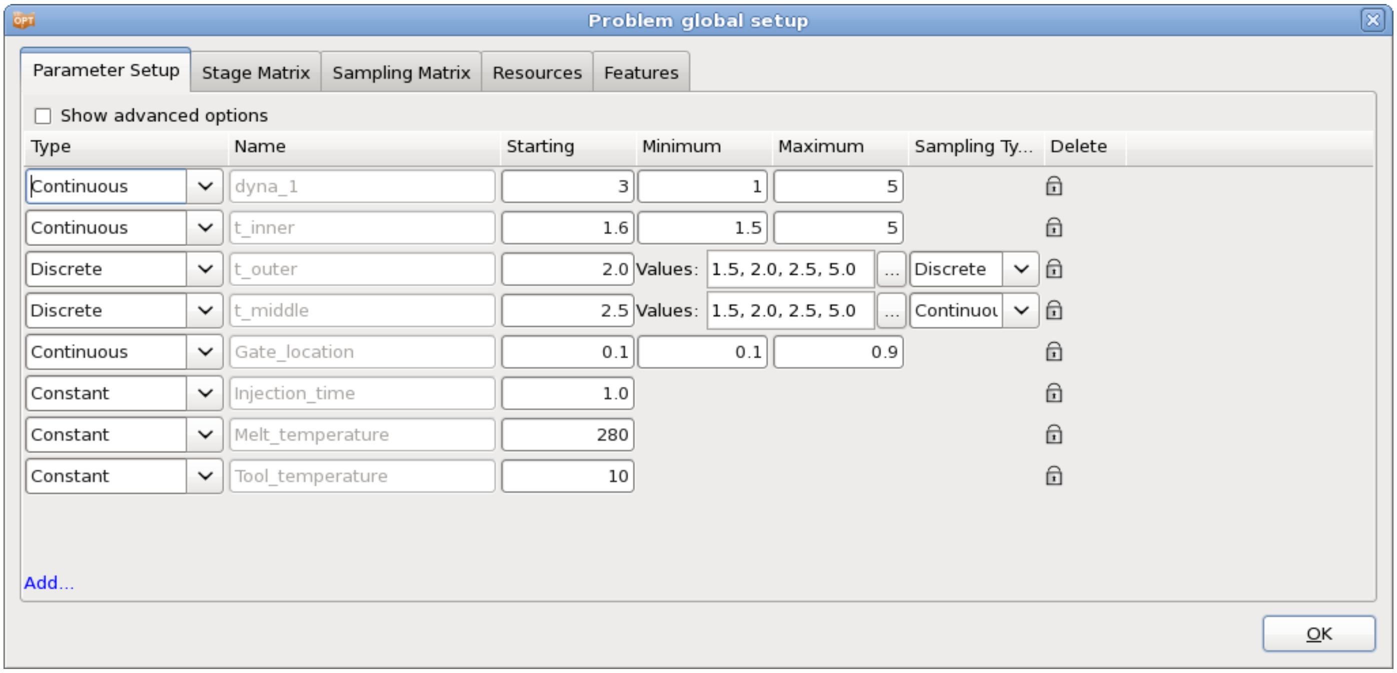
8. Setup Dialog – Defining the Variables

This chapter discusses the conversion of parameters defined in input files to design variables of different types. Graphical features allow the user to view file sources of parameters and the activation or de-activation of variables for selected samplings.

Resource definitions and other global features are also available in this dialog.

8.1. Parameter Setup

Parameters defined in the input files of the stages are automatically displayed in the **Parameter Setup** panel, Figure 8-1. The names of these parameters are not editable, and they cannot be deleted as indicated by the lock symbol displayed in the *Delete* column. If only a name and value are specified in the stage input file, the parameter type is set to *Constant* by default. The default starting value is 0.



□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Sampling Ty...	Delete
Continuous	dyna_1	3	1	5		🔒
Continuous	t_inner	1.6	1.5	5		🔒
Discrete	t_outer	2.0	Values:	1.5, 2.0, 2.5, 5.0 ...	Discrete	🔒
Discrete	t_middle	2.5	Values:	1.5, 2.0, 2.5, 5.0 ...	Continuou	🔒
Continuous	Gate_location	0.1	0.1	0.9		🔒
Constant	Injection_time	1.0				🔒
Constant	Melt_temperature	280				🔒
Constant	Tool_temperature	10				🔒

Figure 8-1: Setup Dialog – Parameter Setup panel in LS-OPTui

Other attributes such as parameter values or discrete sets defined in the input files are also displayed here, but can be overridden. The desired parameter type and other appropriate options can also be specified, Table 8-1.

Advanced options, such as initial range, that are not required can be specified by selecting the *Show advanced options* checkbox, Table 8-2.

Additional (non-file) parameters, although unusual, can be defined using the *Add* button at the bottom of the panel.

Table 8-1: Parameter Setup options to be specified for each parameter

Option	Description		Reference
Type	Parameter type:		
	Continuous	Continuous variable	-
	Constant	Constant value	Section 8.1.1
	Dependent	Parameter depending on other parameters	Section 8.1.2
	Discrete	Discrete variable	Section 8.1.3
	String	Discrete variable using string values	Section 8.1.3
	String Constant	Constant using string values	Section 8.1.1
	Transfer Variable	Parameter treated as variable at upper level and constant at lower level (multi-level optimization)	Section 8.1.4
	Transfer String Variable	Transfer Variable using string values	Section 8.1.4
	Response Variable	Variable which inherits the value of a response	Section 8.1.5
	Noise	Probabilistic variable described by a statistical distribution	Section 8.1.5
Name	Parameter name. If the parameter is imported from a stage input file, the name is not editable		-
Starting	Initial value of the variable, used in baseline run (1.1)		-
Minimum	Lower bound of the design space		-

Maximum	Upper bound of the design space	-
Values	List of allowable values for discrete and string variable	Section 8.1.3
Definition	Mathematical expression specifying a dependent parameter	Section 8.1.2
Distribution	Statistical distribution used to define a probabilistic variable	Section 8.1.7
Sampling Type	Sampling type for discrete variable: continuous or discrete	Section 8.1.3
Edit Input Parameter References	Set the relation of a transfer variable with another variable	Section 8.1.4
Variable Correlation	Dialog to define correlation between variables. Only available for probabilistic tasks.	Section 8.1.8

Table 8-2: Parameter Setup advanced options

Option	**Description**	**Reference**
Init. Range	Size of subregion of the design space used in the first iteration	Section 8.1.8
Saddle Direction	Saddle direction specification used for worst-case design	Section 8.1.10

Table 8-3: Parameter Setup options

Option	**Description**	**Reference**
Show advanced options	Shows Init. Range and Saddle Direction option for each parameter	Table 8-2
Noise Variable Subregion Size (in Standard Deviations)	Bounds are required for noise variables to construct the metamodels. The bounds are taken to a number of standard deviations away from the mean; the default being two standard deviations of the distribution. In general, a noise variable is bounded by the distribution specified and does not have upper and lower bounds similar to control variables.	-
Enforce Variable Bounds	Assigning a distribution to a control value may result in designs exceeding the bounds on the control variables. The default is not to enforce the bounds.	-

8.1.1. Constants

Each variable above can be modified to be a constant. A constant can be a number or a string. Constants are used:

1. to define constant values in the input file such as *π*, *e* or any other constant that may relate to the optimization problem, e.g. initial velocity, event time, integration limits, etc.
2. if native parameters defined in the input file are not to be used as optimization parameters.
3. to convert a variable to a constant. This requires only changing the designation variable to constant in the command file without having to modify the input template. The number of optimization variables is thus reduced without interfering with the template files. Variables can also be eliminated by unchecking them in the Sampling matrix (see Section 8.3)

8.1.2. Dependent variables

Dependent variables are functions of the basic variables and are required to define quantities that have to be replaced in the input template files, but which are dependent on the optimization variables. They do therefore not contribute to the size of the optimization problem. Dependents can be functions of dependents.

Dependent variables are specified using mathematical expressions (see Appendix F: Mathematical Expressions).

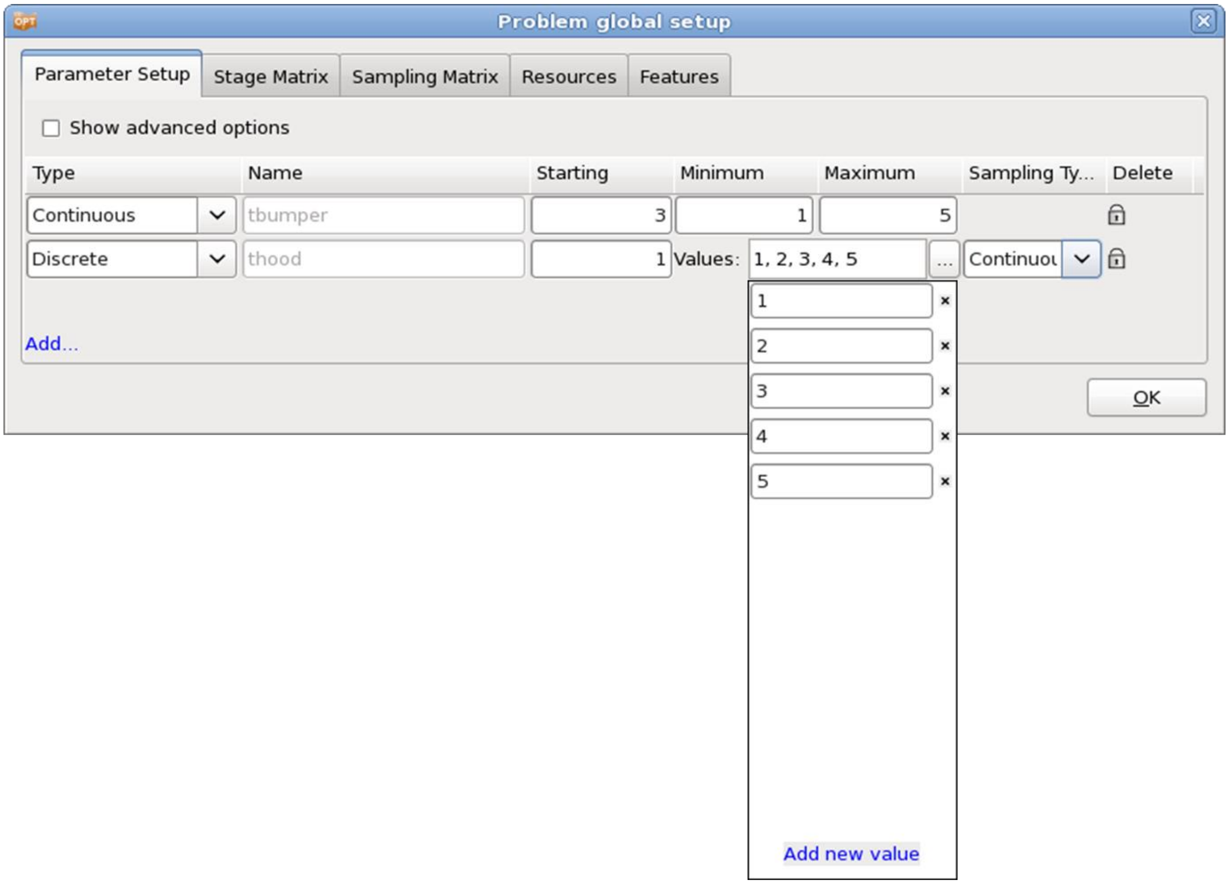
The dependent variables can be specified in an input template and will therefore be replaced by their actual values.

8.1.3. Discrete and String variables

For Discrete variables, a list of allowable values has to be specified. This can be done in the **Parameter Setup** dialog using the **…** button to the right of the *Values* textfield of the respective parameter, Figure 8-2. A list opens up showing the already defined values, a textfield to enter a new value appears by selecting the *Add new value* button or by using the return key.

For *String* variables, allowable string values are defined in the same way. The string values are internally treated as integers in LS-OPT. The mapping of these integer values and the actual strings is stored in the *StringVar.lsox* database in the work directory.

In addition to a list of values, the sampling type has to be specified for discrete variables. By default, the discrete variables are treated as continuous variables for generating experimental designs. The optimal values will assume an allowable value. If discrete sampling is selected, all experimental design points use allowable values. If possible, a continuous sampling is recommended, because it usually leads to a better distribution of the points within the design space and hence to a better metamodel quality.



Problem global setup

☐ Show advanced options

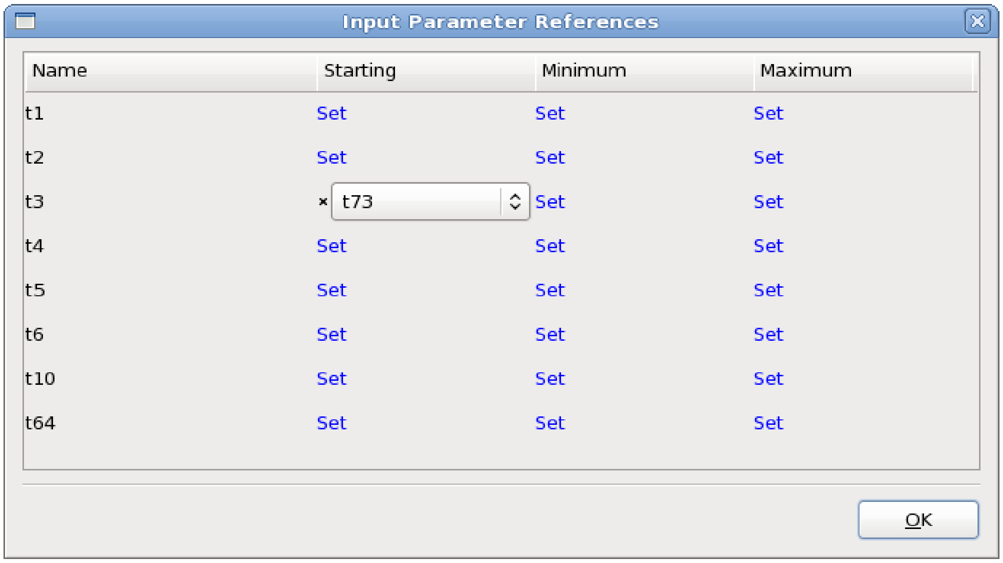
Type	Name	Starting	Minimum	Maximum	Sampling Ty...	Delete
Continuous	tbumper	3	1	5		🔒
Discrete	thood	1	Values: 1, 2, 3, 4, 5	...	Continuou	🔒

Values:

- 1 ×
- 2 ×
- 3 ×
- 4 ×
- 5 ×

Figure 8-2: Definition of discrete values

8.1.4. Transfer variables



Input Parameter References

Name	Starting	Minimum	Maximum
t1	Set	Set	Set
t2	Set	Set	Set
t3	× t73	Set	Set
t4	Set	Set	Set
t5	Set	Set	Set
t6	Set	Set	Set
t10	Set	Set	Set
t64	Set	Set	Set

Figure 8-3: Input Parameter References. Transfer Variable t73 is set as the starting value for t3.

Transfer variables are used in the context of multilevel optimization (see Section 5.3.9). These variables are sampled in one of the levels, but these sample values are passed down to the lower levels where these are treated as constants. Transfer variables can be referenced by preceding higher levels or by other variables in the same level. Within the same level, a transfer variable can be the starting value or the lower/upper bound for another variable (Figure 8-3). The dialog to define the parameter references is accessible from the Parameter Setup dialog.

8.1.5. Response variables

Response variables are used to define variables which inherit the values of responses. The main purpose is to allow substitution of response values in input files. The response must be calculated in an ancestor of the stage in which the substitution is done.

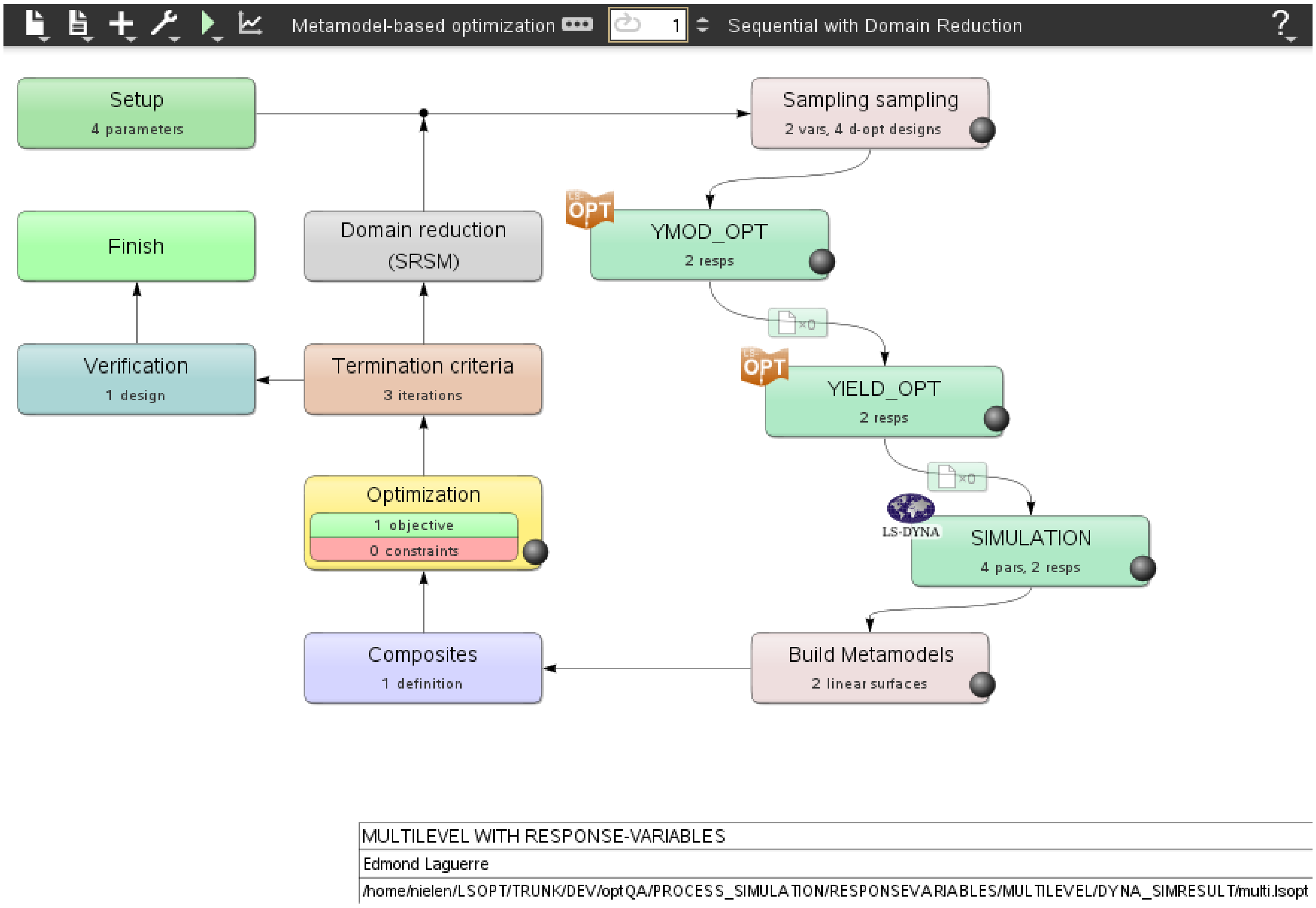
1. The main parameter setup allows the user to link a parameter to a response (See Figure 8-5). This selection causes the selected parameter value to be replaced by a response value defined in an ancestor stage. The transferred response value is substituted into the input file(s) of stages downstream where the parameters are defined.
2. The response value to be linked can be any response value which was directly extracted from the solver database or a mathematical expression involving any variables, dependents, histories or responses defined in any parent stages.
3. Response variables can be transferred between any two stages of a particular thread. They do not need to be consecutive as long as the response is defined in a stage which comes before the stage where the substitution is done.
4. A specific response can be linked to any number of parameters.
5. Response variables are not independent design variables, so have no effect on the sampling.

Example

The example is explained using the series of figures below. The optimization consists of an outer loop with three stages. The first stage is also an optimization loop which calibrates a parameter YMod to produce YMod_OPT. The second stage uses the optimized YMod_OPT as a constant parameter but optimizes a second variable Yield to produce Yield_OPT.

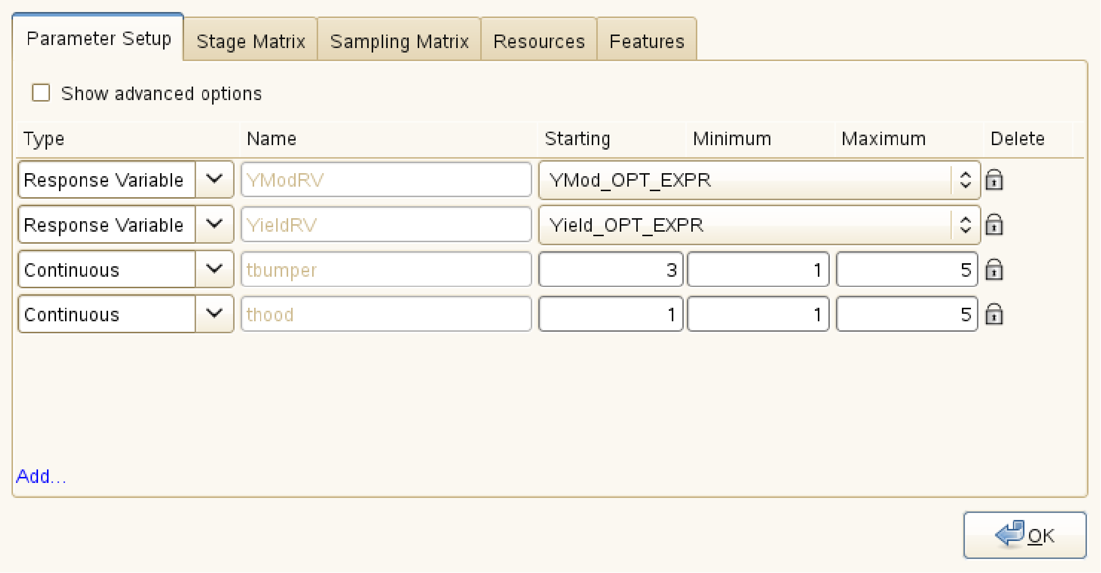
After the first two stages, YMod_OPT and Yield_OPT are converted using mathematical expressions and then transferred as material constants to a vehicle simulation stage. The outer loop optimizes the vehicle design variables tbumper and thood.

Figure 8-4 through Figure 8-10 show various parts of the problem setup.



MULTILEVEL WITH RESPONSE-VARIABLES
Edmond Laguerre
/home/nielen/LSOPT/TRUNK/DEV/optQA/PROCESS_SIMULATION/RESPONSEVARIABLES/MULTILEVEL/DYNA_SIMRESULT/multi.lsopt

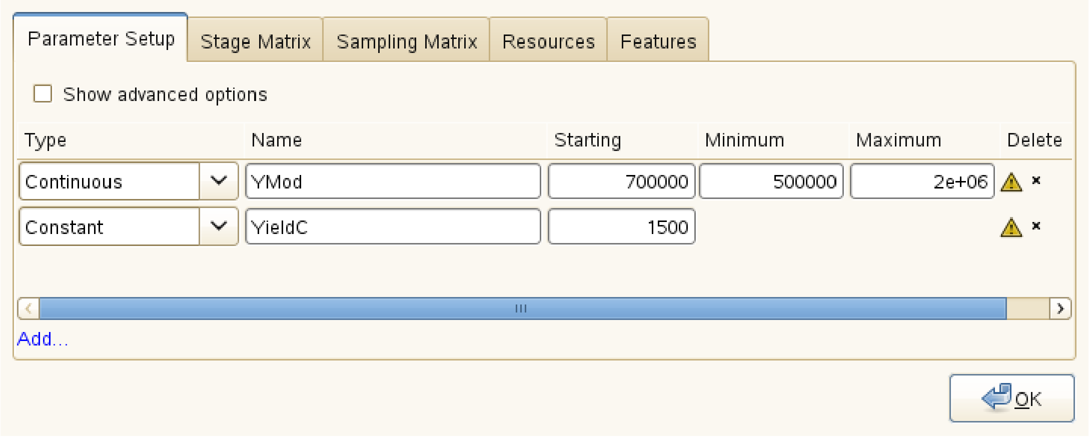
Figure 8-4: LS-OPT Problem multilevel setup. The first two stages (YMOD_OPT) and (YIELD_OPT) are sublevel optimization stages. YMOD_OPT produces an optimal material parameter YMod_OPT and converts it to YMod_OPT_EXPR using an expression. This value is transferred to the parameter YModRV defined as an input parameter to the YIELD_OPT stage. The YIELD_OPT stage therefore uses this value as a constant but optimizes a second variable Yield to produce Yield_OPT which is then converted to Yield_OPT_EXPR. Both YMod_OPT_EXPR and Yield_OPT_EXPR are then transferred to the SIMULATION stage as input parameters. The outer loop depicted here optimizes over design variables tbumper and thood to minimize vehicle intrusion.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Response Variable	YModRV	YMod_OPT_EXPR			🔒
Response Variable	YieldRV	Yield_OPT_EXPR			🔒
Continuous	tbumper	3	1	5	🔒
Continuous	thood	1	1	5	🔒

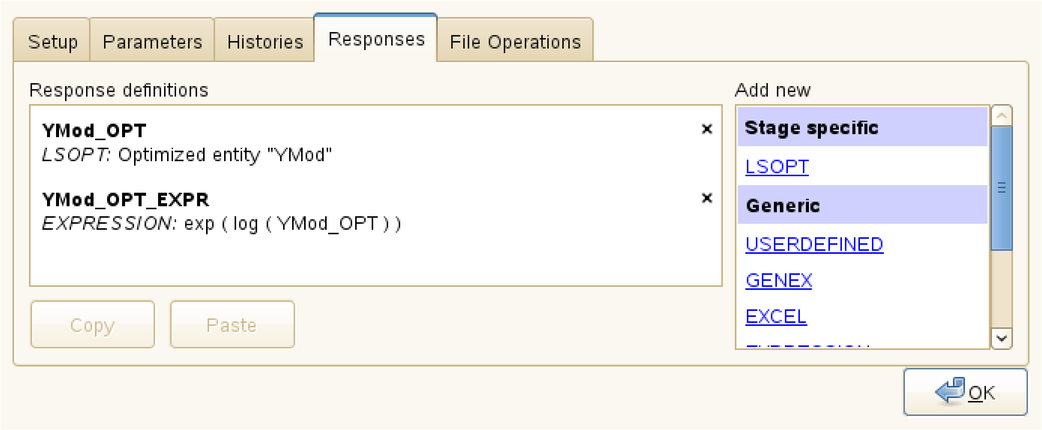
Figure 8-5: The main parameter setup (clicking green Setup box at top left of Figure 8-4) to define two response-variables YModRV and YieldRV. These respectively link to YMod_OPT_EXPR and Yield_OPT_EXPR produced by the parent optimization stages. The parameters tbumper and thood are optimization variables used in the outer loop.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	YMod	700000	500000	2e+06	⚠ ×
Constant	YieldC	1500			⚠ ×

Figure 8-6: Input parameters for the YMOD_OPT stage. YMod is an optimization variable defined in this stage while YieldC is a constant.

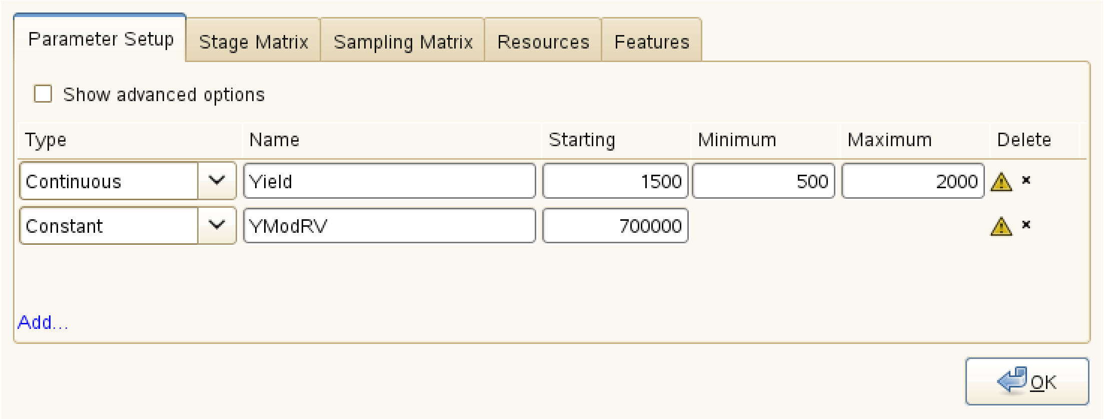


Response definitions

YMod_OPT ×
LSOPT: Optimized entity "YMod"

YMod_OPT_EXPR ×
EXPRESSION: exp (log (YMod_OPT))

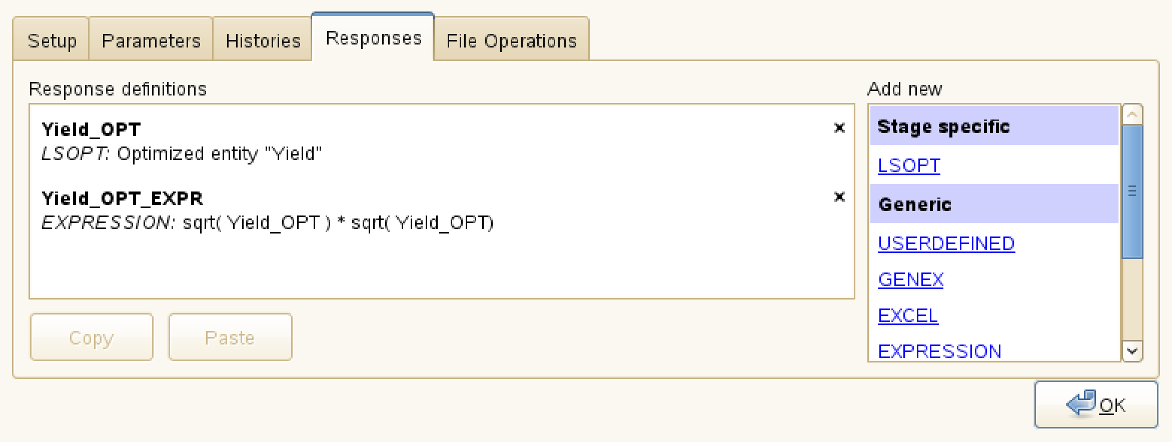
Figure 8-7: Response output definition for Stage YMOD_OPT.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	Yield	1500	500	2000	⚠ ×
Constant	YModRV	700000			⚠ ×

Figure 8-8: Input parameters for the YIELD_OPT stage. Yield is an optimization variable defined in this stage. YModRV is a response-variable replaced by YMod_OPT_EXPR (see Figure 8-5 for definition).

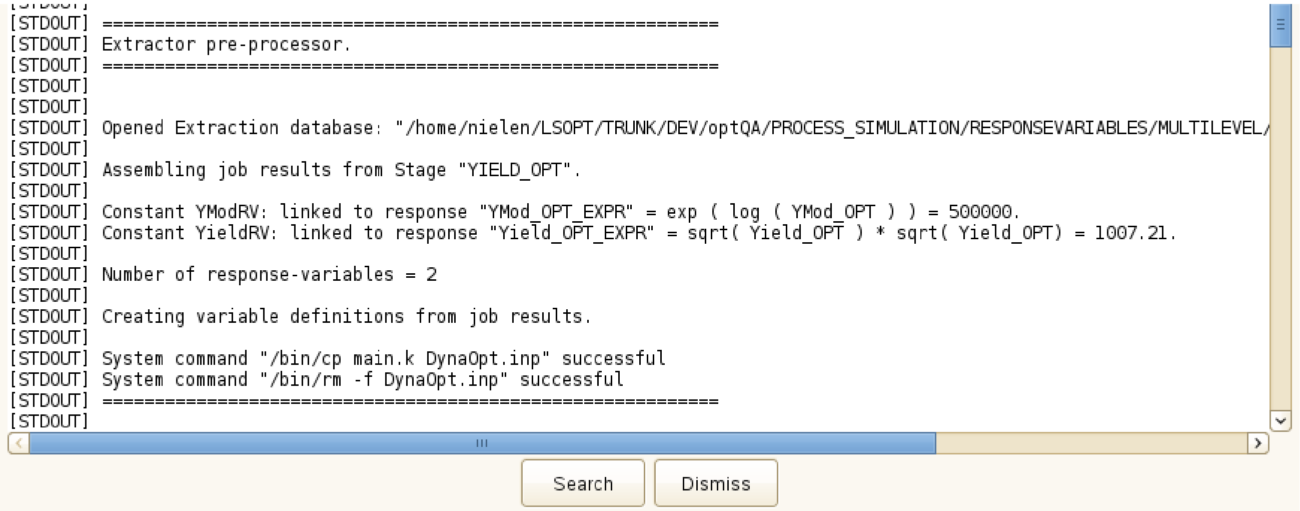


Response definitions

Yield_OPT ×
LSOPT: Optimized entity "Yield"

Yield_OPT_EXPR ×
EXPRESSION: sqrt(Yield_OPT) * sqrt(Yield_OPT)

Figure 8-9: Response output definition for Stage YIELD_OPT.



```
[STDOUT] ==========================================================
[STDOUT] Extractor pre-processor.
[STDOUT] ==========================================================
[STDOUT]
[STDOUT]
[STDOUT] Opened Extraction database: "/home/nielen/LSOPT/TRUNK/DEV/optQA/PROCESS_SIMULATION/RESPONSEVARIABLES/MULTILEVEL/
[STDOUT]
[STDOUT] Assembling job results from Stage "YIELD_OPT".
[STDOUT]
[STDOUT] Constant YModRV: linked to response "YMod_OPT_EXPR" = exp ( log ( YMod_OPT ) ) = 500000.
[STDOUT] Constant YieldRV: linked to response "Yield_OPT_EXPR" = sqrt( Yield_OPT ) * sqrt( Yield_OPT) = 1007.21.
[STDOUT]
[STDOUT] Number of response-variables = 2
[STDOUT]
[STDOUT] Creating variable definitions from job results.
[STDOUT]
[STDOUT] System command "/bin/cp main.k DynaOpt.inp" successful
[STDOUT] System command "/bin/rm -f DynaOpt.inp" successful
[STDOUT] ==========================================================
[STDOUT]
```

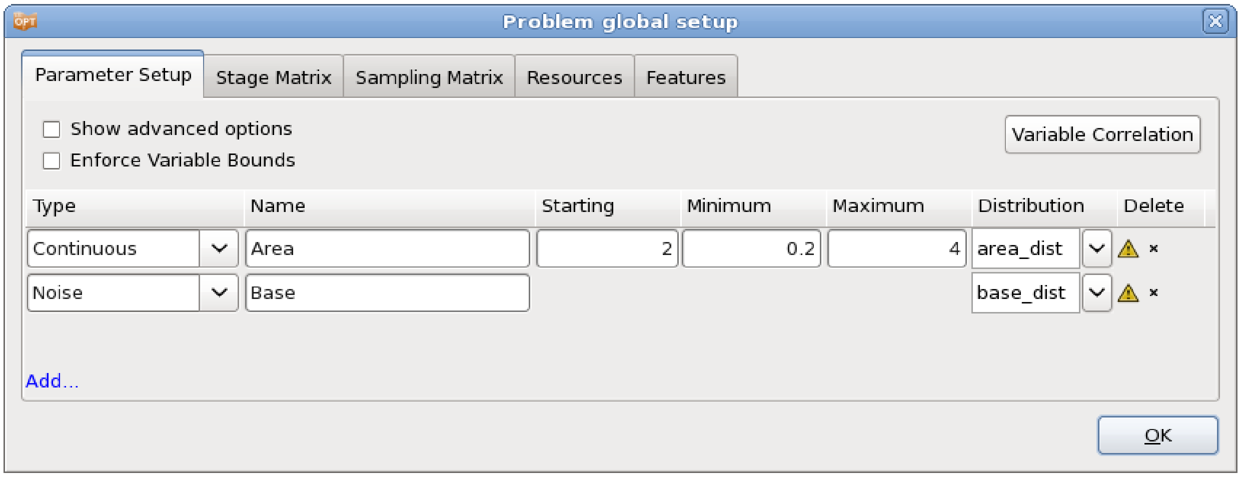
Figure 8-10: Job log of SIMULATION stage of the example (the display represents the pre-processor phase prior to simulation). Note the linking of the two parameters to responses.

8.1.6. Probabilistic Variables - Noise and Control Variables

Probabilistic variable values, unlike deterministic variables, cannot be stated with absolute confidence. In other words, there is uncertainty associated with these variables because of which we can only state that their value will lie within a certain interval with specific level of confidence. This difference makes probabilistic analysis and optimization much more involved than their deterministic counterparts. Therefore, a separate chapter (Chapter 14) is dedicated to probabilistic tasks and problem setup.

Probabilistic variables can either be control variables, whose nominal values are modified during optimization to get a more suitable design, or noise variables that are not controlled during optimization and only serve the purpose of introducing uncertainty in the problem. The variable type can be selected in the Parameter Setup panel (Figure 8-11).

8.1.7. Probabilistic distributions



☐ Show advanced options
☐ Enforce Variable Bounds

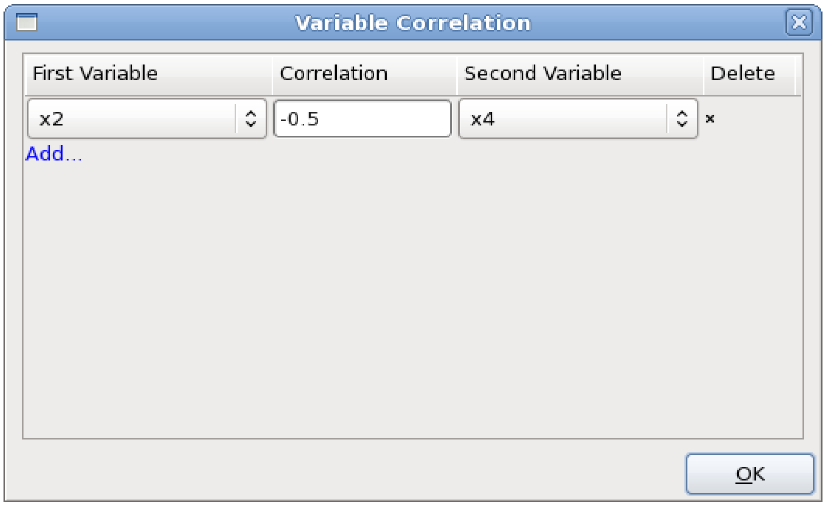
Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Continuous	Area	2	0.2	4	area_dist	×
Noise	Base				base_dist	×

Figure 8-11: Parameter setup panel for probabilistic tasks

In order to represent variable uncertainties, they are associated with probabilistic distributions, which are also part of the Parameter Setup panel when the selected task is probabilistic (Figure 8-11). Several types of distributions are available in LS-OPT. Further details of how to set up probabilistic variables and distributions are provided in Chapter 14.

8.1.8. Variable Correlation

For probabilistic tasks, the correlation between variables can be specified, Figure 8-12. The Variable Correlation dialog is accessible from the Parameter Setup panel. This correlation will be considered in Monte Carlo simulation (including metamodel based simulations) as well as in reliability based design optimization and robust parameter design, respectively. Only correlation between normally distributed variables is allowed.



First Variable	Correlation	Second Variable	Delete
x2	-0.5	x4	×

Figure 8-12: Variable Correlation dialog, accessible from Parameter Setup panel.

8.1.9. Size and location of initial region of interest (range)

If an initial range is specified, the initial subregion is defined as *[starting – range/2, starting + range/2]*.

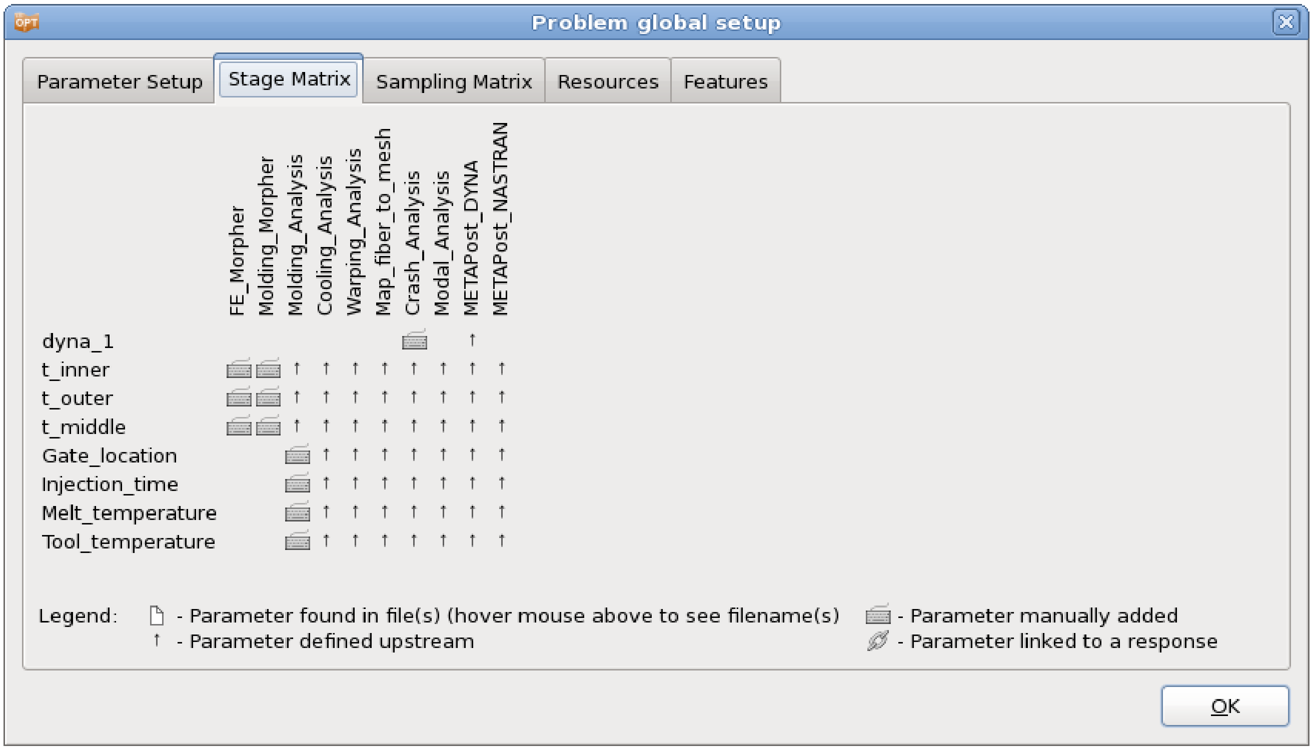
Remarks

1. The full design space is used if the range is omitted.
2. The region of interest is centered on a given design and is used as a sub-space of the design space to define the experimental design. If the region of interest protrudes beyond the design space, it is moved without contraction to a location flush with the design space boundary.

8.1.10. Saddle direction: Worst-case design

Worst-case or saddle-point design is defined as a method to minimize (or maximize) the objective function with respect to *some* variables, while maximizing (or minimizing) it with respect to the *remaining* variables in the variable set. The maximization variables are set using the Maximize option in the Saddle Direction field of the **Parameter Setup** panel. The default selection is Minimize.

8.2. Stage Matrix



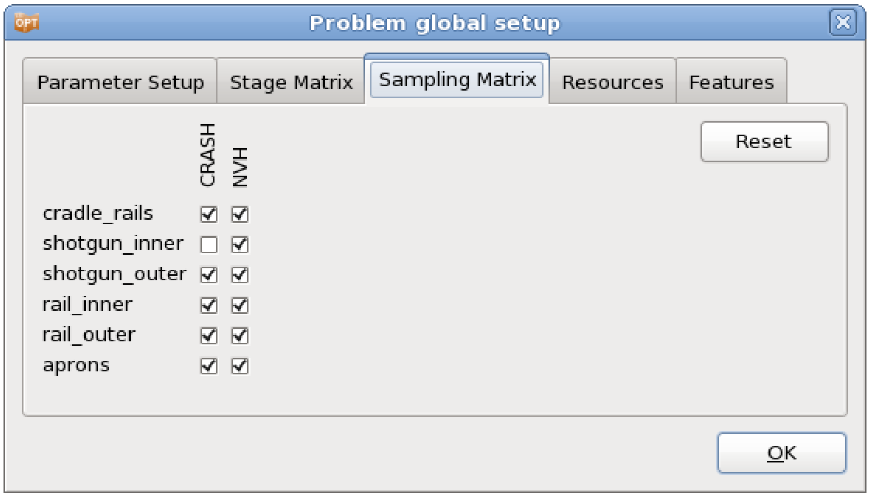
	FE_Morpher	Molding_Morpher	Molding_Analysis	Cooling_Analysis	Warping_Analysis	Map_fiber_to_mesh	Crash_Analysis	Modal_Analysis	METAPost_DYNA	METAPost_NASTRAN
dyna_1							▤		↑	
t_inner	▤	▤	↑	↑	↑	↑	↑	↑	↑	↑
t_outer	▤	▤	↑	↑	↑	↑	↑	↑	↑	↑
t_middle	▤	▤	↑	↑	↑	↑	↑	↑	↑	↑
Gate_location			▤	↑	↑	↑	↑	↑	↑	↑
Injection_time			▤	↑	↑	↑	↑	↑	↑	↑
Melt_temperature			▤	↑	↑	↑	↑	↑	↑	↑
Tool_temperature			▤	↑	↑	↑	↑	↑	↑	↑

Legend: 🗋 - Parameter found in file(s) (hover mouse above to see filename(s) ▤ - Parameter manually added
↑ - Parameter defined upstream 🔗 - Parameter linked to a response

Figure 8-13: Stage Matrix

The Stage Matrix provides an overview of the parameters defined in each stage. A parameter influences a stage if it is defined in a stage input file, manually added to a stage, or defined in an upstream stage. Hovering the mouse over a file icon shows a list of the files where the respective parameter is defined.

8.3. Sampling Matrix



	CRASH	NVH
cradle_rails	☑	☑
shotgun_inner	□	☑
shotgun_outer	☑	☑
rail_inner	☑	☑
rail_outer	☑	☑
aprons	☑	☑

Figure 8-14: Sampling Matrix

For multidisciplinary design optimization (MDO) certain variables could be relevant for some but not all disciplines. In such examples, several samplings (or cases) can be defined and the variables assigned to some but not all samplings. The assignment of a variable to a sampling can be selected in the Sampling Matrix. If a variable is absent in a particular sampling, it assumes the current global value as generated by the previous iteration for substitution in the input files of the next iteration. The number of variables selected for a sampling directly affects the number of sampling points (and hence the computational effort) required for that sampling. Each column is coupled to the **Active Variables** tab of the respective **Sampling Dialog**, Section 9.4.

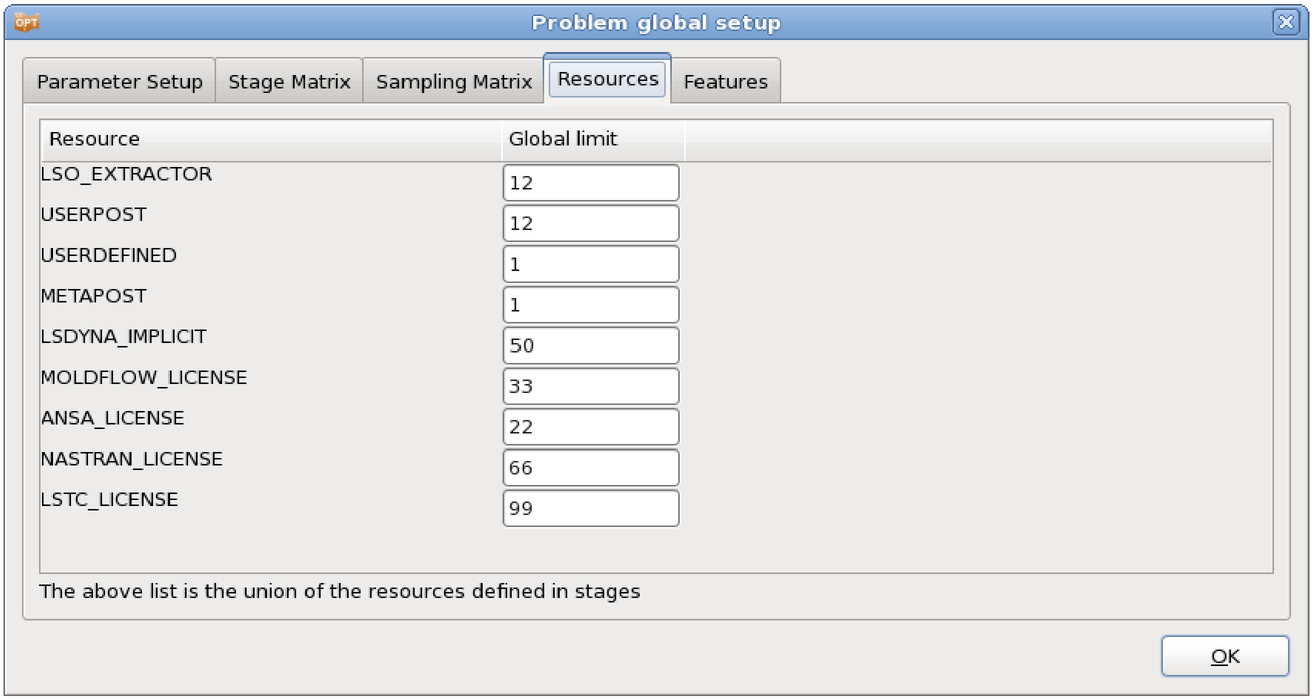
Clicking the *Reset* button reassigns the variables to the samplings as defined in the input files.

If a variable has been deselected for all the Samplings, it is treated as a constant value. Therefore the baseline value will be assumed throughout the optimization. This option can be selected in lieu of explicitly defining the parameter as a constant.

The sampling matrix can be changed between iterations. Variables detected as insensitive in the first or any other iteration could be switched off for the following iterations.

See Section 20.5 for an MDO example.

8.4. Resources



Resource	Global limit
LSO_EXTRACTOR	12
USERPOST	12
USERDEFINED	1
METAPOST	1
LSDYNA_IMPLICIT	50
MOLDFLOW_LICENSE	33
ANSA_LICENSE	22
NASTRAN_LICENSE	66
LSTC_LICENSE	99

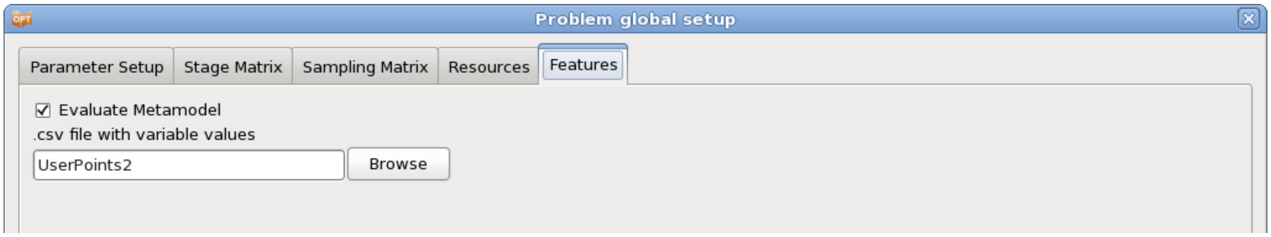
The above list is the union of the resources defined in stages

Figure 8-15: Setup – Resources

Resources are defined in the Stage dialogs, but, for convenience, allows editing of the global limits in the Setup dialog. The Resources tab shows a summary of all resources defined for all the stages, Section **5.4.1**.

The resource LSO_EXTRACTOR is generated automatically by LS-OPT and used for result extraction on the local machine. The default global limit is the number of logical CPUs available on the local machine.

8.5. Features



☑ Evaluate Metamodel

.csv file with variable values

UserPoints2

Figure 8-16: Setup – Features

Sampling independent features are available in the **Features** tab of the **Setup** dialog, Figure 8-16.

8.5.1. Evaluate Metamodel

The response values of any number of points can be computed using an existing metamodel and written to a .csv file (file with comma-separated variables that can be read by most spreadsheet programs). The input data is sampling independent.

There are two simple steps to obtain a table with response data.

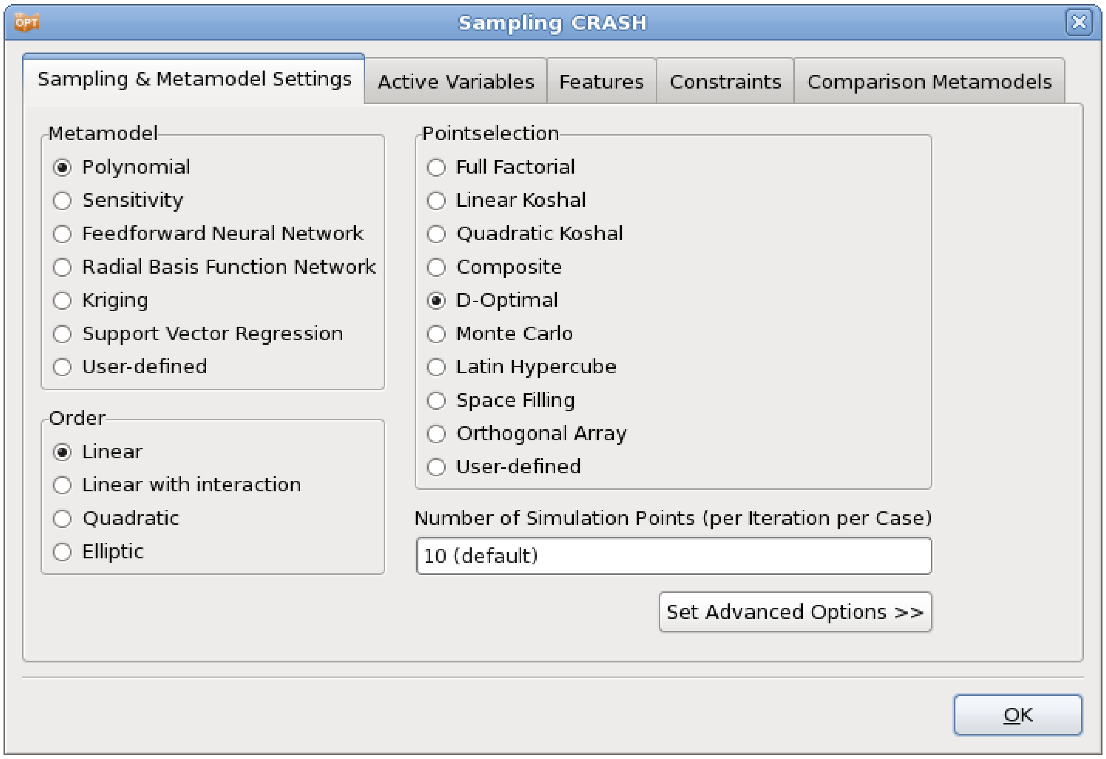
1. Browse for the file with the sampling point information using the *Evaluate Metamodel* option in the *Features* tab in the *Setup* dialog. The file must be in `.csv` format although spaces, commas or tabs are allowed as delimiters. Only files with ASCII characters are supported. The file must contain two header lines. The first header line contains the variable names. The second header line contains the variable types; in this case "`dv`" (design variable) suffices. The variable types “nv” (noise variable), “dc” (discrete variable) or “st” (string variable) can also be used and will achieve the same result (see also Appendix E.4.1). It should be noted that the entry for a string variable is the corresponding mapped integer value that can be found in the file *StringVar.lsox*. The variable coordinates are specified as one row for each design point. See example below.
2. Use the *Setup* dialog or command line *Repair* option *Evaluate Metamodels*, Section 2.2.1 and 3.5.
   - *Input:* Each sampling point file must represent all the variables. LS-OPT checks whether all the variables defined in the file are represented in the LS-OPT input. Variable order is not important.
   - *Output:* The `ExtendedResults` output can be found as a `META` file in the main working directory, e.g. `ExtendedResultsMETAMaster_3.csv`. The `ExtendedResults` file has variable, dependent, response, composite, objective, constraint, multi-objective and constraint violation values.
   - If sampling points are defined before the start of an optimization run, the META file will be automatically computed for each iteration.

Example .csv file:

```
x1 x2 x3
dv dv dv
1.0 2.0 3.0
2.0 3.0 4.0
4.1 6.2 3.3
```

9. Sampling & Metamodel Dialog

This chapter describes the specification of sampling settings, i.e. the metamodel types, point selection schemes (design of experiments or DOE), and related options available in the Sampling dialog, Figure 9-1. The terms *point selection* and *experimental design*, are used interchangeably.



Sampling CRASH

Sampling & Metamodel Settings | Active Variables | Features | Constraints | Comparison Metamodels

Metamodel
- ● Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Order
- ● Linear
- ○ Linear with interaction
- ○ Quadratic
- ○ Elliptic

Pointselection
- ○ Full Factorial
- ○ Linear Koshal
- ○ Quadratic Koshal
- ○ Composite
- ● D-Optimal
- ○ Monte Carlo
- ○ Latin Hypercube
- ○ Space Filling
- ○ Orthogonal Array
- ○ User-defined

Number of Simulation Points (per Iteration per Case): 10 (default)

Figure 9-1: Sampling dialog – metamodel and point selection settings

9.1. Metamodel types

The user can select one of the metamodel types shown in Figure 9-1 and Table 9-1, respectively. The default selection for the metamodel type and the point selection scheme depends on the choice of task and optimization strategy, Chapter 4. For the sequential response surface method (SRSM) strategy, the default choice is the polynomial response surface method (RSM) where response surfaces are fitted to results at data points using polynomials. For global approximations fitted in the single iteration and sequential strategies, the radial basis function networks are set as the default approximation models. For all strategies, the feed-forward neural network, Kriging, Support Vector Regression and user-defined approximation models are also available. Sensitivity data (analytical or numerical) can also be used for optimization. This method is more suitable for linear analysis solvers. For details see the sections referred to in Table 9-1.

Table 9-1: Sampling dialog options – Metamodel types

Metamodel Type	**Description**	**Reference**
Polynomial	Polynomial approximations up to quadratic order	Section 9.1.1
Sensitivity	Uses gradients to determine linear metamodels.	Section 9.1.2
Feedforward Neural Network	An artificial Neural network with sigmoid basis functions	Section 9.1.3
Radial Basis Function Network	A Neural Network with radial basis functions	Section 9.1.3
Kriging	A Gaussian process. Form of Bayesian inference.	Section 9.1.4
Support Vector Regression	Support Vector Regression	Section 9.1.5
User-defined	Interface for user-defined, dynamically linked metamodel.	Section 9.1.6

9.1.1. Polynomial

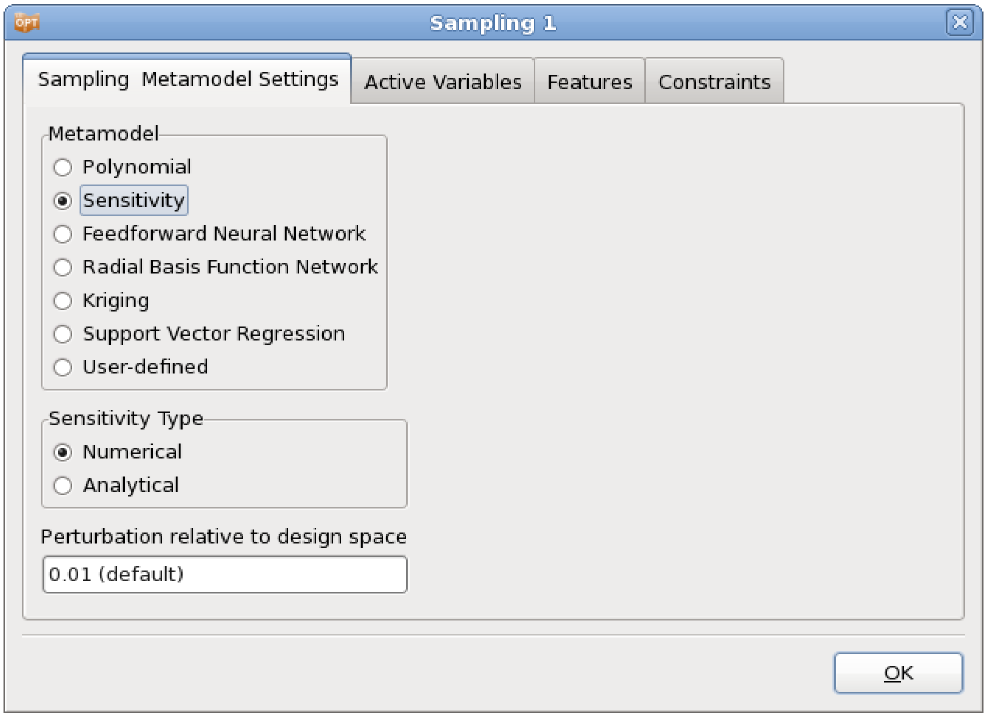
When polynomial response surfaces are constructed, the user can select from different approximation orders. The available options are linear, linear with interaction (linear and off-diagonal terms), elliptic (linear and diagonal terms) and quadratic, Section 23.1.1. In the *Sampling* dialog, the approximation order is set in the *Order* field, Figure 9-1. Increasing the order of the polynomial results in more terms in the polynomial, and therefore more coefficients that need to be determined, hence more simulation runs are needed. The default number of simulation runs is automatically updated for the polynomial type (Section 9.3.1).

The polynomial terms can be used during the variable screening process (Section 23.4) to determine the significance of certain variables (main effects) and the cross-influence (interaction effects) between variables when determining responses. These results can be viewed graphically (Section 16.3.4).

The recommended point selection scheme for polynomial response surfaces uses the *D*-optimality criterion (Section 9.3.2).

9.1.2. Sensitivity

In this approach, sensitivities are used to generate linear metamodels. Both analytical and numerical sensitivities can be used for optimization, Figure 9-2.



Metamodel
○ Polynomial
● Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression
○ User-defined

Sensitivity Type
● Numerical
○ Analytical

Perturbation relative to design space
0.01 (default)

Figure 9-2: Sampling Dialog: Sensitivity options

Analytical sensitivities

If analytical sensitivities are available, they must be provided for each response in its own file named `Gradient`. The values (one value for each variable) in `Gradient` should be placed on a single line, separated by spaces.

In the Sampling dialog, the Sensitivity Type must be set to *Analytical*.

A complete example is given in Section 20.7.

Numerical sensitivities

To use numerical sensitivities, select *Numerical* in the Sensitivity Type field in the Sampling dialog and assign the perturbation as a fraction of the design space, Figure 9-2.

Numerical sensitivities are computed by perturbing *n* points relative to the current design point $\mathbf{x}^0$, where the *j*-th perturbed point is:

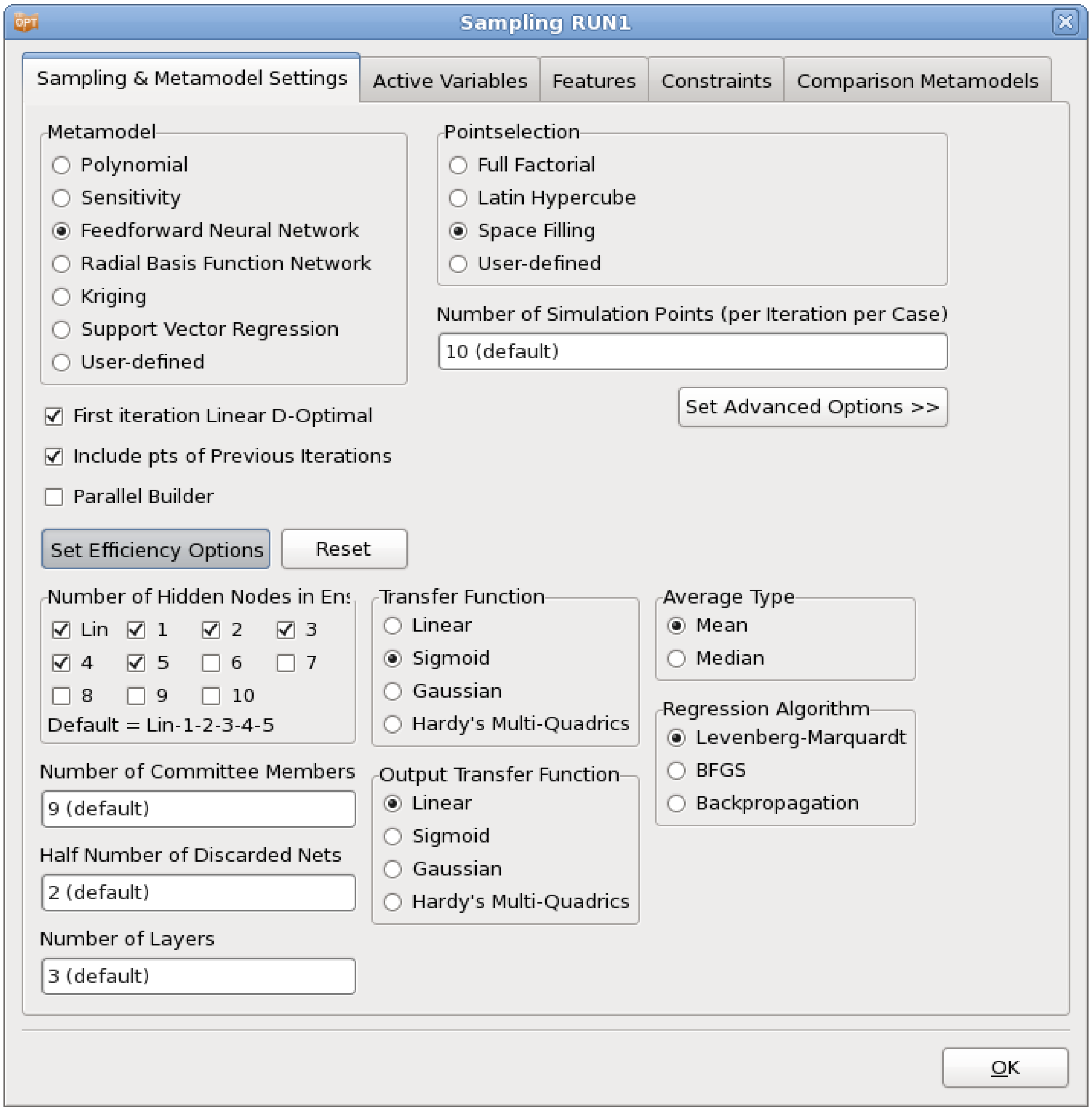
$$x_i^j = x_i^0 + \delta_{ij}\varepsilon(x_{iU} - x_{iL})$$

$\delta_{ij} = 0$ if $i \neq j$ and 1.0 if $i = j$. The perturbation constant ε is relative to the design space size. The same value applies to all the variables. The value of ε can be specified by the user, the default is 0.001.

9.1.3. Feedforward Neural networks and radial basis function networks

To apply feedforward neural network or radial basis function approximations, select the appropriate option in the *Metamodel* field in the *Sampling* dialog, see Figure 9-3 and Figure 9-6, respectively. The recommended Point Selection scheme for feedforward neural networks and radial basis functions is the space filling method (which is also the default), Section 9.3.4.

FFNN Efficiency Options*



Metamodel
- ○ Polynomial
- ○ Sensitivity
- ● Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Pointselection
- ○ Full Factorial
- ○ Latin Hypercube
- ● Space Filling
- ○ User-defined

Number of Simulation Points (per Iteration per Case): 10 (default)

- ☑ First iteration Linear D-Optimal
- ☑ Include pts of Previous Iterations
- □ Parallel Builder

Number of Hidden Nodes in Ens
- ☑ Lin ☑ 1 ☑ 2 ☑ 3
- ☑ 4 ☑ 5 □ 6 □ 7
- □ 8 □ 9 □ 10

Default = Lin-1-2-3-4-5

Transfer Function
- ○ Linear
- ● Sigmoid
- ○ Gaussian
- ○ Hardy's Multi-Quadrics

Average Type
- ● Mean
- ○ Median

Regression Algorithm
- ● Levenberg-Marquardt
- ○ BFGS
- ○ Backpropagation

Number of Committee Members: 9 (default)

Half Number of Discarded Nets: 2 (default)

Number of Layers: 3 (default)

Output Transfer Function
- ● Linear
- ○ Sigmoid
- ○ Gaussian
- ○ Hardy's Multi-Quadrics

Figure 9-3: Feedforward Neural Network Efficiency Options

Neural Network construction calculation may be time-consuming because of the following reasons:

1. The committee size is large
2. The ensemble size is large.

Committee size. The default committee size as specified above is largely required because the default number of points when conducting an iterative optimization process is quite small. Because of the tendency of NN's to have larger variability when supplied with fewer points, committees are relied on to stabilize the approximation through averaging. When a large number of points has been simulated however, the committee size can be reduced to a single neural net by setting *Number of Committee Members* to 1.

Ensemble size. The ensemble size can be reduced in two ways:

1. by exactly specifying the architecture of the ensemble and
2. by providing a threshold to the RMS training error.

The architecture is specified using the *Number of Hidden Nodes in Ensemble* options. Higher order neural nets are more expensive to compute.

FFNN efficiency options are available in the *Sampling* dialog if the *Set Efficiency Option* button is pressed, and may be reset to the default settings using the *Reset* button, Figure 9-3. The available options are explained in Table 9-2.

Please refer to Sections **24.3** and **26.5** for recommendations on how to use metamodels.

Table 9-2: Feedforward Neural Network Efficiency Options

Option	**Description**
Number of Hidden Nodes in Ensemble	Ensemble size from which one will be selected according to the minimum Generalized Cross Validation (GCV) value across the ensemble. The default is Lin-1-2-3-4-5.
Number of Committee Members	Because of the natural variability of neural networks (Section 24.1.2), the user is allowed to select the number of members in a neural net committee. To ensure distinct members, the regression procedure uses new randomly selected starting weights for generating each committee member.
Half Number of Discarded Nets	The discard option allows the user to discard committee members with the lowest mean squared fitting error and committee members with the highest MSE. This option is intended to exclude neural nets which are either under- or over-fitted. The total number of nets excluded is therefore 2 times the specified number. The discard feature is activated during the regression procedure.
Transfer Function	Activation function of intermediate layer (typically sigmoid)
Output Transfer Function	Activation function of output layer (typically linear)

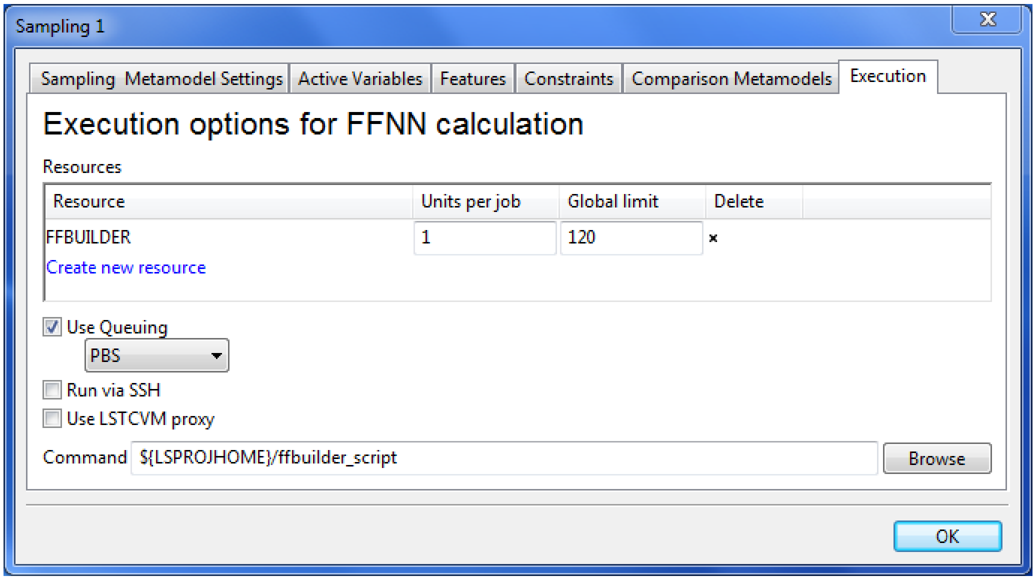
Average Type	Mean or Median
Regression Algorithm	Algorithm to solve non-linear regression problem. Levenberg-Marquardt, Broyden-Fletcher-Goldfarb-Shanno(BFGS) or Resilient Backpropagation (RPROP)

Execution options for FFNN calculation (Parallel Builder)

FFNNs can be solved concurrently. Select the *Parallel Builder* option in the **Settings** tab to enable the **Execution** tab.

The Parallel Builder involves the application of the job scheduler to treat each response and each member of a neural network ensemble as a job to be run in parallel. The committee (which constitutes a particular ensemble member and which is solved using a serial Monte Carlo analysis) is solved serially. Figure 9-4 shows the dialog. The main features are as follows:

1. Job monitoring is available by clicking on the LED on the Metamodel box of the main dialog, Figure 9-5. All the features that apply to the monitoring of simulations (except LS-PrePost) are also available for FFNN calculation.
2. Remote computation is supported, so if a cluster setup is available for e.g. LS-DYNA jobs, the FFNN solution setup may only involve a few special settings. See Section 5.5 for a description of the remote options.



Execution options for FFNN calculation

Resources

Resource	Units per job	Global limit	Delete
FFBUILDER	1	120	×

☑ Use Queuing

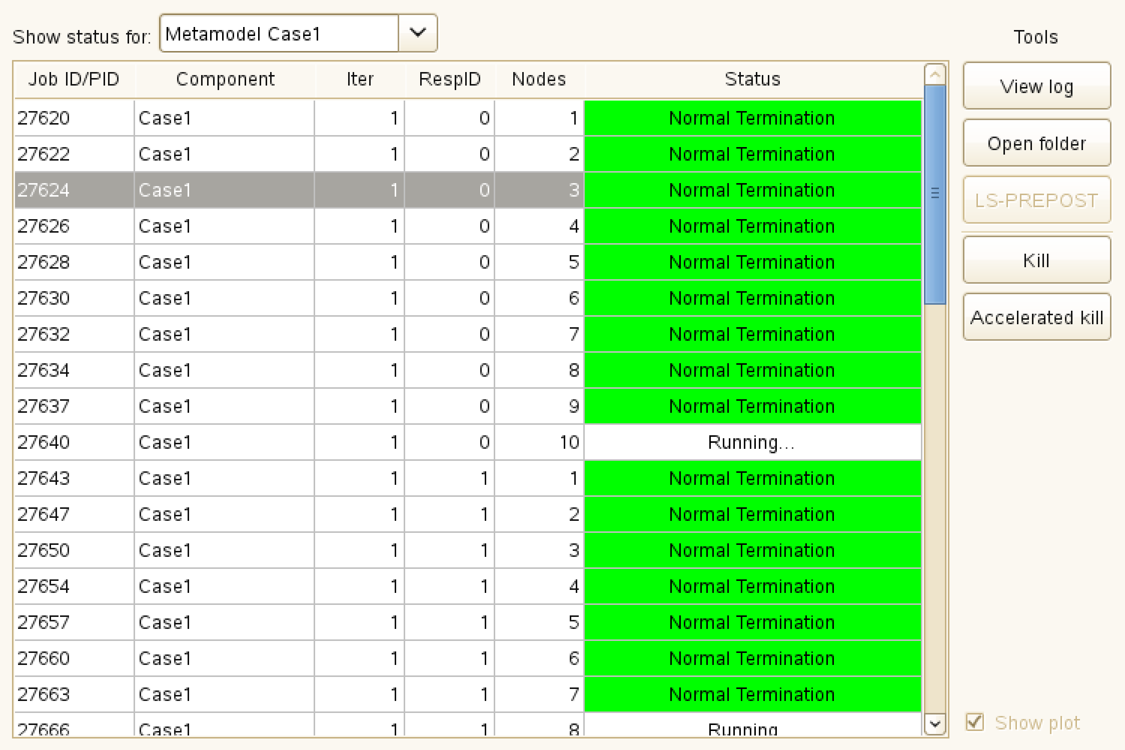
PBS

☐ Run via SSH

☐ Use LSTCVM proxy

Command ${LSPROJHOME}/ffbuilder_script

Figure 9-4: Dialog for Parallel FFNN builder



Show status for: Metamodel Case1

Job ID/PID	Component	Iter	RespID	Nodes	Status
27620	Case1	1	0	1	Normal Termination
27622	Case1	1	0	2	Normal Termination
27624	Case1	1	0	3	Normal Termination
27626	Case1	1	0	4	Normal Termination
27628	Case1	1	0	5	Normal Termination
27630	Case1	1	0	6	Normal Termination
27632	Case1	1	0	7	Normal Termination
27634	Case1	1	0	8	Normal Termination
27637	Case1	1	0	9	Normal Termination
27640	Case1	1	0	10	Running...
27643	Case1	1	1	1	Normal Termination
27647	Case1	1	1	2	Normal Termination
27650	Case1	1	1	3	Normal Termination
27654	Case1	1	1	4	Normal Termination
27657	Case1	1	1	5	Normal Termination
27660	Case1	1	1	6	Normal Termination
27663	Case1	1	1	7	Normal Termination
27666	Case1	1	1	8	Running

Figure 9-5: Job progress display for parallel FFNNs

Advanced RBF options: Basis functions and optimization criterion for RBF*

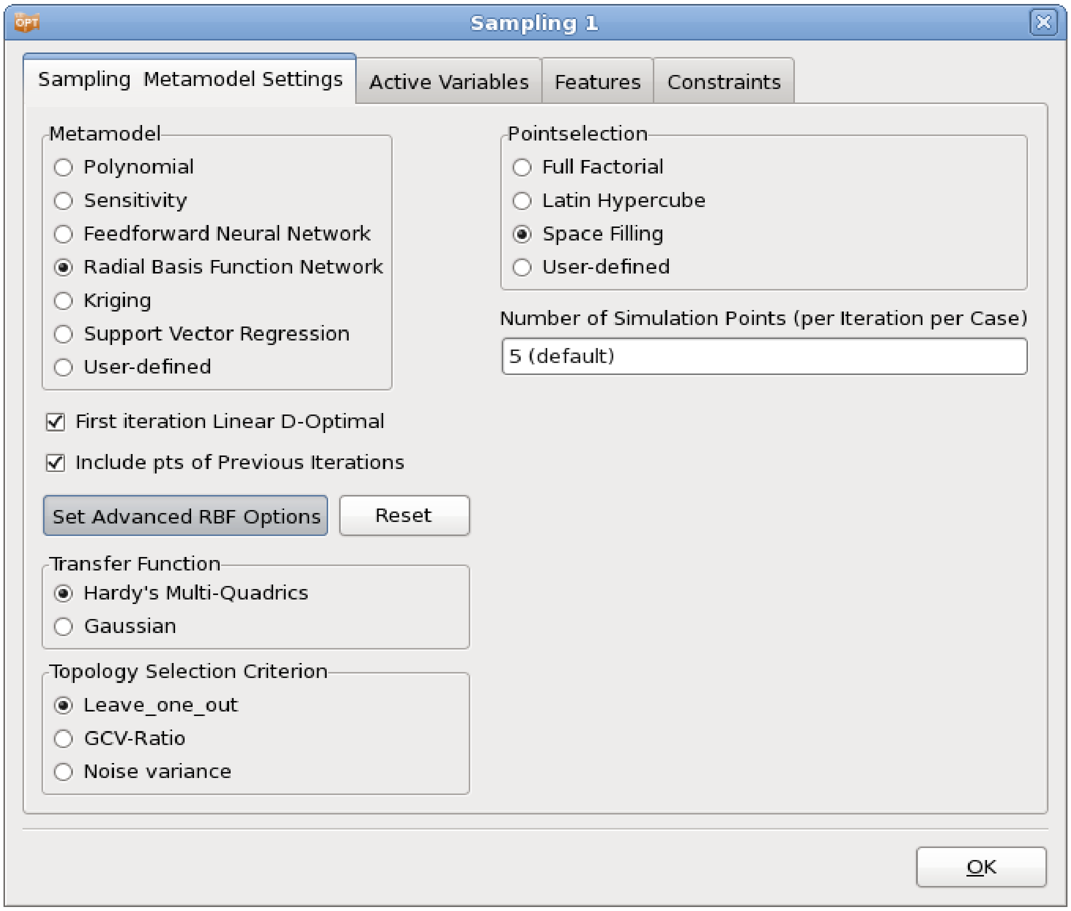
The performance of the RBFs can significantly vary with the choice of basis function and the optimization criterion. Two basis functions available for selection are Hardy's multi-quadrics (HMQ), and Gaussian RBF. HMQ is often preferred and has therefore been set as the default. The user is also allowed to select the optimization criterion to be generalized cross-validation error or the pointwise ratio of the generalized cross validation error, Figure 9-6.

The options are available in the *Sampling* dialog if the *Set Advanced RBF Options* button is pressed, and may be reset to the default settings using the *Reset* button, Figure 9-6. The available options are described in Table 9-3.

Table 9-3: RBF Advanced Options

Option	Description	Options	Description
Transfer Function	Basis function	Hardy's Multi-Quadrics	$g_h\,(x_1,\ldots,x_k) = \sqrt{1+(r^2/\sigma_h^2)}$
		Gaussian	$g_h\,(x_1,\ldots,x_k) = exp[-r^2/2\sigma_h^2]$
	Optimization criterion	Leave-one-out	Generalized cross-validation error (PRESS)

Topology Selection Criterion	GCV-Ratio	Pointwise ratio of the generalized cross validation error
	Noise variance	Variance of the fitting error


Sampling 1

Sampling Metamodel Settings | Active Variables | Features | Constraints

Metamodel
- ○ Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ● Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

☑ First iteration Linear D-Optimal

☑ Include pts of Previous Iterations

Set Advanced RBF Options | Reset

Transfer Function
- ● Hardy's Multi-Quadrics
- ○ Gaussian

Topology Selection Criterion
- ● Leave_one_out
- ○ GCV-Ratio
- ○ Noise variance

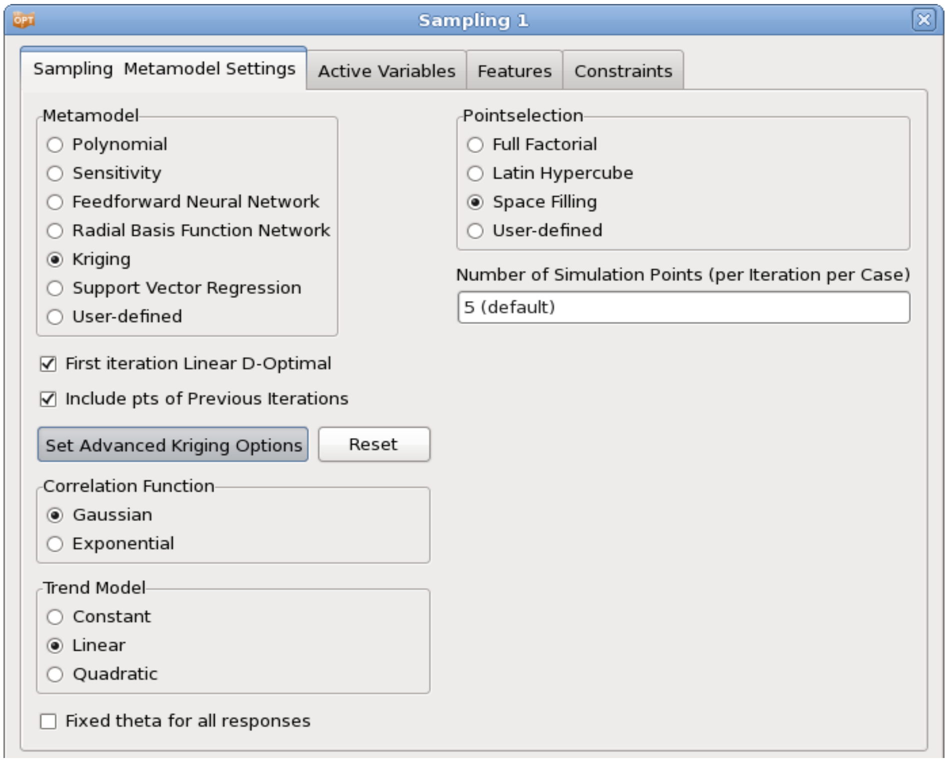
Pointselection
- ○ Full Factorial
- ○ Latin Hypercube
- ● Space Filling
- ○ User-defined

Number of Simulation Points (per Iteration per Case): 5 (default)

OK

Figure 9-6: Radial Basis Function Network Advanced Options

9.1.4. Kriging parameters



Metamodel
○ Polynomial
○ Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
● Kriging
○ Support Vector Regression
○ User-defined

☑ First iteration Linear D-Optimal
☑ Include pts of Previous Iterations

Set Advanced Kriging Options | Reset

Correlation Function
● Gaussian
○ Exponential

Trend Model
○ Constant
● Linear
○ Quadratic

□ Fixed theta for all responses

Pointselection
○ Full Factorial
○ Latin Hypercube
● Space Filling
○ User-defined

Number of Simulation Points (per Iteration per Case)
5 (default)

Figure 9-7: Kriging Advanced Options

The Kriging fit depends on the choice of appropriate correlation function and the trend model, Section 24.2. Two correlation functions available for selection are Gaussian and exponential. The user can also select either a constant, linear, or quadratic trend model. The available options are displayed in Table 9-4.

Table 9-4: Advanced Kriging Options

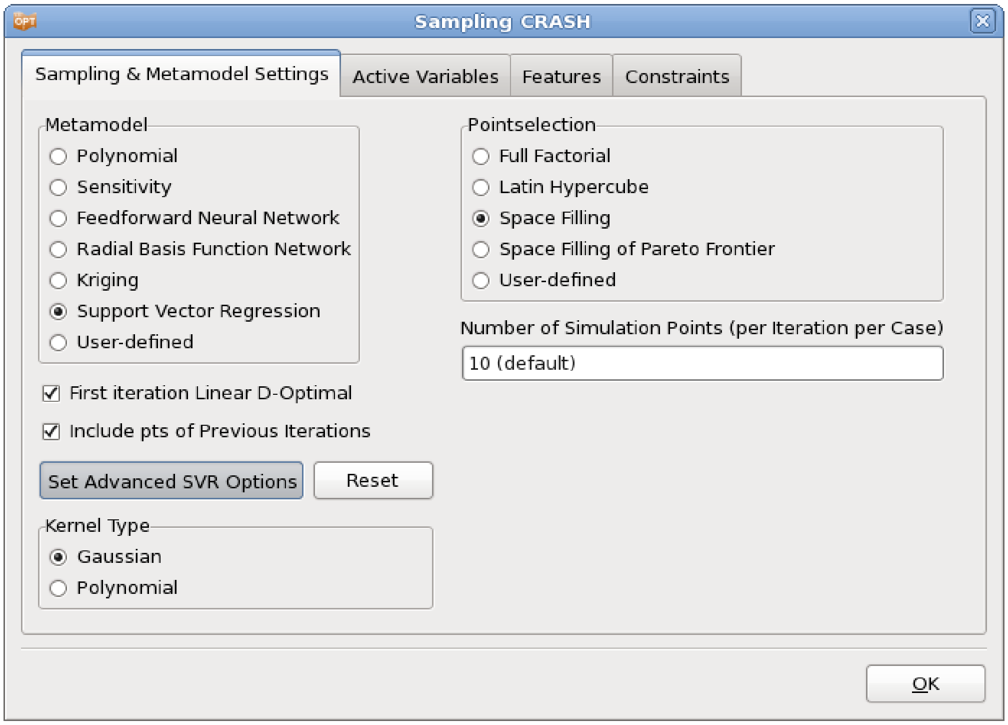
Option	Options	Description
Correlation Function	Gaussian, Exponential	Correlation function used in stochastic component of metamodel function, see Section 24.2.
Trend Model	Constant, Linear, Quadratic	Polynomial component of metamodel function. The linear trend model requires at least $(n + 2)$ design points, a quadratic trend model requires at least $\frac{(n+1)(n+2)}{2} + 1$ design points, where n is the number of variables.
Fixed theta for all responses		By default, a single set of theta values is fit to all responses, however the user can also fit individual set of correlation function parameters (theta) for each response by selecting this option.

9.1.5. Support Vector Regression

The support vector regression fit depends on the choice of appropriate kernel function (similar to correlation function), Section 24.3. Two kernel functions available for selection are Gaussian and polynomial. The available options are displayed in Table 9-5.

Table 9-5: Advanced Support Vector Regression Options

Option	Options	Description
Kernel Type	Gaussian, Polynomial	Basis function used in SVR expansion that maps the input variable space to a high dimensional feature space, see Section 24.3 .



Sampling CRASH

Sampling & Metamodel Settings | Active Variables | Features | Constraints

Metamodel
○ Polynomial
○ Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
● Support Vector Regression
○ User-defined

☑ First iteration Linear D-Optimal
☑ Include pts of Previous Iterations

Set Advanced SVR Options | Reset

Kernel Type
● Gaussian
○ Polynomial

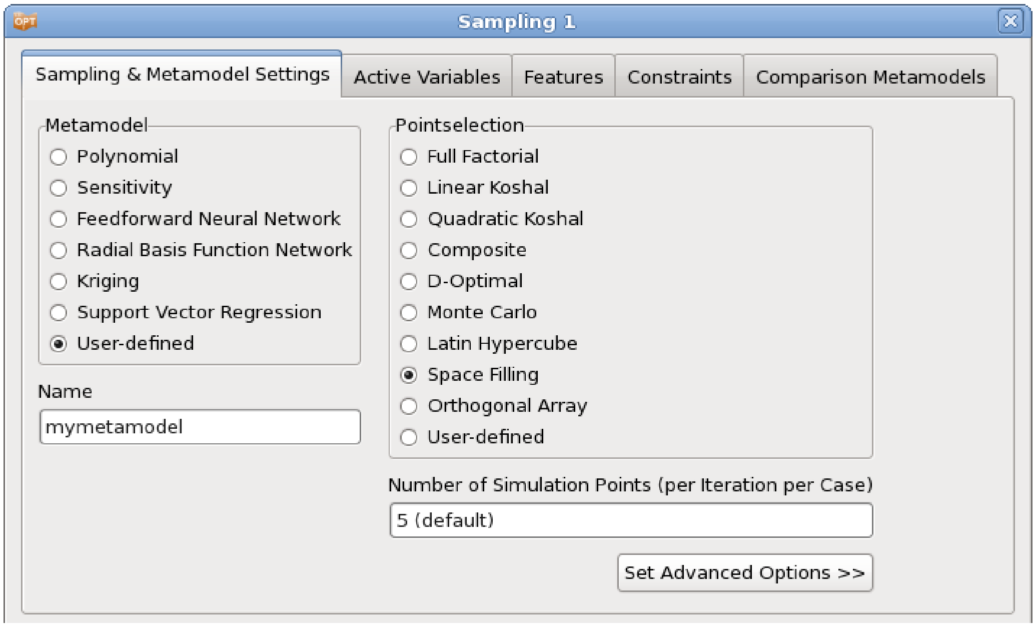
Pointselection
○ Full Factorial
○ Latin Hypercube
● Space Filling
○ Space Filling of Pareto Frontier
○ User-defined

Number of Simulation Points (per Iteration per Case)
10 (default)

OK

Figure 9-8: Metamodel selection Support Vector Regression

9.1.6. User-defined metamodel*



Metamodel

○ Polynomial
○ Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression
● User-defined

Name: mymetamodel

Pointselection

○ Full Factorial
○ Linear Koshal
○ Quadratic Koshal
○ Composite
○ D-Optimal
○ Monte Carlo
○ Latin Hypercube
● Space Filling
○ Orthogonal Array
○ User-defined

Number of Simulation Points (per Iteration per Case): 5 (default)

Figure 9-9: User Defined Metamodel Options

The user-defined metamodel distribution is available for download at http://ftp.lstc.com/user/ls-opt/Add_On_Libraries/.

Please ask LSTC or your local LS-DYNA distributor for the password.

Building the example

Under Linux, issue the command "make" while in this directory. Your resulting metamodel is called umm_avgdistance_linux_i386.so (or umm_avgdistance_linux_x86_64.so if running under 64-bit OS).

Under Windows, open usermetamodel.sln in Visual Studio. Open the Build menu, select "Build solution". Your resulting metamodel is called umm_avgdistance_win32.dll

Along with the metamodel binary you also get an executable called "testmodel". This program can be used for simple verification of your metamodel. Just give the name of your metamodel as a parameter, i.e.:

```
testmodel avgdistance
```

Note that you are not supposed to supply the full .dll/.so filename as a parameter.

Using the example as a template

If you wish to use the example as a template for your own metamodel, do the following steps (in this example, your metamodel is called mymetamodel):

Copy avgdistance.* to mymetamodel.*

Replace any occurrence of the string "`avgdistance`" with "`mymetamodel`" in the following files: Makefile, mymetamodel.def, mymetamodel.vcproj, Makefile, usermetamodel.sln

Distributable metamodel

When compiled, your metamodel binary will be called something like:

```
umm_mymetamodel_win32.dll
```

or

```
umm_mymetamodel_linux_i386.dll
```

This is the only file that is needed in order to use the metamodel from LS-OPT.

Referring to user-defined metamodels in the Sampling dialog

In order to use a user-defined metamodel for a certain sampling, select the User-defined option in the metamodel selection in the *Sampling* dialog and add the metamodel name to the *Name* textfield, (e.g. `umm_`**`mymetamodel`**`_linux_i386.so`), Figure 9-9.

Note that the name should not include the "`umm_`" prefix or the platform dependent suffix. LS-OPT will look for the correct file based upon the current platform. This allows for cross platform operation.

9.2. General Options for Non-Polynomial Metamodels

Additional options available for Feedforward Neural Networks, Radial Basis Functions, Kriging and Support Vector Regression are summarized in Table 9-6.

Table 9-6: FFNN, RBF, Kriging and SVR options

Option	Description	Reference
First iteration Linear D-Optimal	Use linear metamodels and the *D*-optimality point selection criterion for the first iteration instead of the selected types.	Section 9.2.1
Include pts of previous iterations	The new points for each iteration are selected within the new subregion while considering the locations of points from previous iterations. The metamodels are constructed using the new points as well as points from all previous iterations.	Section 9.2.2
Parallel Builder	Only FFNN, calculates metamodels in parallel	Section 9.1.3

9.2.1. First Iteration Linear D-Optimal

For Feedforward Neural Networks, Radial Basis Functions, Kriging and Support Vector Regression, the main scheme can be replaced in the first iteration by linear polynomials with D-optimal point selection, using the *First iteration Linear D-Optimal* option, because

1. D-optimality minimizes the size of the confidence intervals to ensure the most accurate variable screening, usually done in the first iteration.
2. It addresses the variability encountered with neural networks due to possible sparsity (or poor placement) of points early in the iterative process, especially in iteration 1, which has the lowest point density.

9.2.2. Include points of previous iterations

Updating the experimental design involves augmenting an existing design with new points. Updating only makes sense if the response surface can be successfully adapted to the augmented points such as for neural nets, Radial Basis Function networks or Kriging surfaces in combination with a space filling scheme.

The new points have the following properties:

1. They are located within the current region of interest.
2. The minimum distance between the new points and between the new and existing points, is maximized (space filling only).

9.3. Point selection schemes

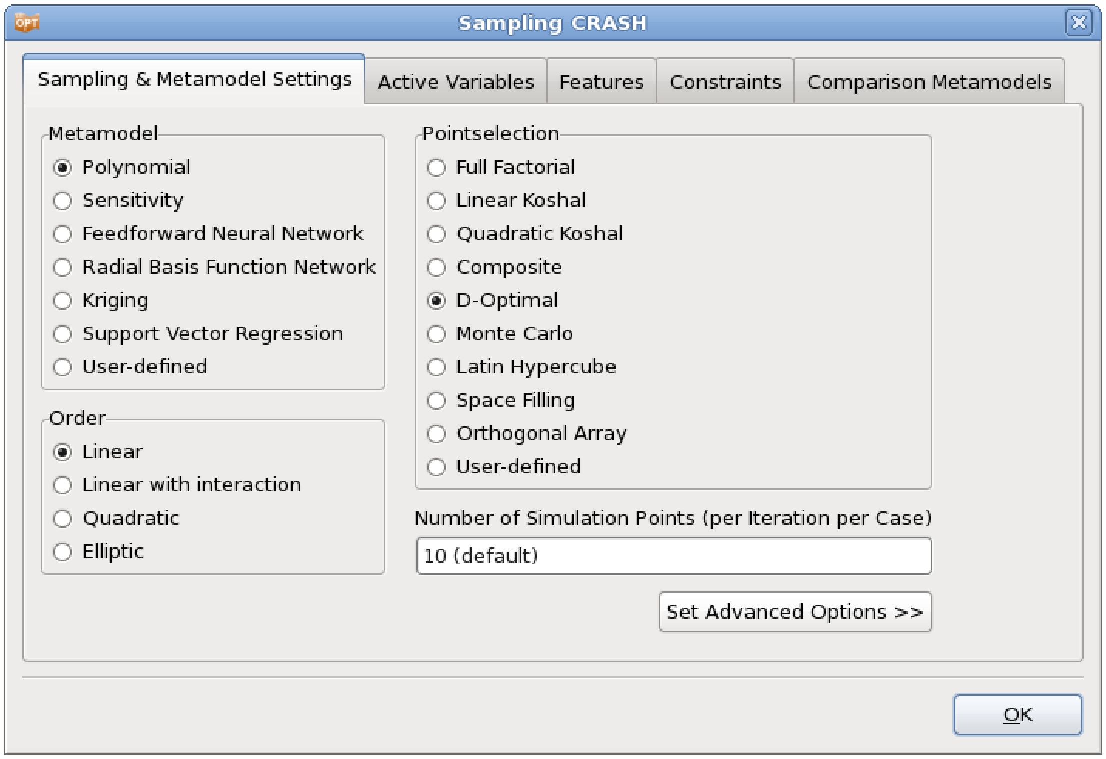
9.3.1. Overview

Table 9-7 shows the available point selection schemes (experimental design methods). The default point selection scheme depends on the selected metamodel type, e.g., the *D*-optimal point selection scheme (basis type: Full Factorial, 11 points per variable (for $n = 2$)) is the default for linear polynomials, and the space-filling scheme is the default for the Feedforward Neural Network, Radial Basis Function Network, Support Vector Regression and Kriging methods.

Table 9-7: Point selection schemes

Point Selection Scheme	Description	Reference
Full Factorial	-	Section 23.2.1
Linear Koshal	Saturated design for first order Polynomials	Section 23.2.2
Quadratic Koshal	Saturated design for quadratic Polynomials	Section 23.2.2
Composite	Central Composite design	Section 23.2.3

D-optimal	Design obtained by minimizing the determinant of the moment matrix	Section 9.3.2, Section 23.2.4
Latin Hypercube	Stratified random design	Section 9.3.3, Section 23.2.5
Monte Carlo	Random design	
Space Filling	Design obtained by maximizing the minimum distance between any two points.	Section 9.3.4, Section 23.2.6
Space Filling of Pareto Frontier	Design obtained by maximizing the minimum distance between any two points sampled from the Pareto Optimal Frontier.	Section 9.3.5
Orthogonal Array	Fractional Factorial design	Section 9.3.6
User-defined	-	Section 9.3.7



Metamodel
● Polynomial
○ Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression
○ User-defined

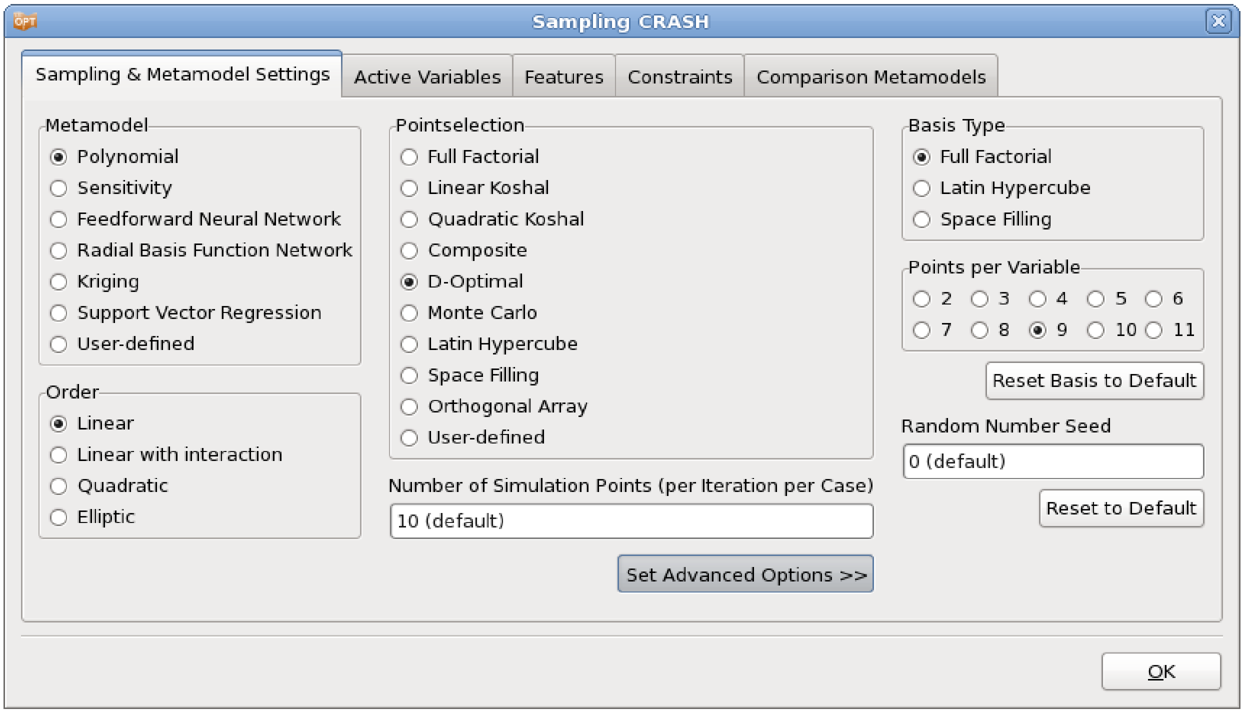
Order
● Linear
○ Linear with interaction
○ Quadratic
○ Elliptic

Pointselection
○ Full Factorial
○ Linear Koshal
○ Quadratic Koshal
○ Composite
● D-Optimal
○ Monte Carlo
○ Latin Hypercube
○ Space Filling
○ Orthogonal Array
○ User-defined

Number of Simulation Points (per Iteration per Case)
10 (default)

Figure 9-10: Point selection schemes

9.3.2. D-Optimal point selection



Metamodel
● Polynomial
○ Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression
○ User-defined

Order
● Linear
○ Linear with interaction
○ Quadratic
○ Elliptic

Pointselection
○ Full Factorial
○ Linear Koshal
○ Quadratic Koshal
○ Composite
● D-Optimal
○ Monte Carlo
○ Latin Hypercube
○ Space Filling
○ Orthogonal Array
○ User-defined

Number of Simulation Points (per Iteration per Case)
10 (default)

Basis Type
● Full Factorial
○ Latin Hypercube
○ Space Filling

Points per Variable
○ 2 ○ 3 ○ 4 ○ 5 ○ 6
○ 7 ○ 8 ● 9 ○ 10 ○ 11

Random Number Seed
0 (default)

Figure 9-11: D-optimal point selection: advanced options

The *D*-Optimality design criterion is available for Polynomial and User-defined metamodels and can be used to select the best (optimal) set of points for a response surface from a given set of points. The basis set can be determined using any of the other point selection schemes. The default basis experiment for the D-optimal design is based on the number of variables *n*. For small values of *n*, the Full Factorial design is used, whereas larger *n* employs a Space Filling method for the basis experiment. The Latin Hypercube design is also useful to construct a basis experimental design for the D-optimal design for a large number of variables where the cost of using a Full Factorial design is excessive. E.g. for 15 design variables, the number of basis points for a 3-level design is more than 14 million.

The basis experiment attributes can be overridden using the *Set Advanced Options* in the Sampling Dialog.

The type and order of the metamodel used has an influence on the distribution of the optimal experimental design. The default number of points selected for the *D*-optimal design is $\mathrm{int}(1.5(n+1))+1$ for linear, $\mathrm{int}(1.5(2n+1))+1$ for elliptic, $\mathrm{int}(0.75(n^2+n+2))+1$ for interaction, and $\mathrm{int}(0.75(n+1)(n+2))+1$ for quadratic. As a result, about 50% more points than the minimum required are generated. If the user wants to override this number of experiments, this can be done using the respective textfield in the Sampling dialog.

The *D*-optimal scheme is the recommended point selection scheme for polynomial response surfaces.

The D-optimal scheme is repeatable, but a random number seed can be provided to create different sets of random points, Section 9.3.8.

9.3.3. Latin Hypercube Sampling

The Latin Hypercube point selection scheme is typically used for probabilistic analysis. Like Monte Carlo and Space-Filling point selection schemes, it requires a user-specified number of experiments.

Latin Hypercube Sampling may be used to fit a response surface, but even if the Latin Hypercube design has enough points to fit a response surface, there is a likelihood of obtaining poor predictive qualities or near singularity (when fitting polynomials) during the regression procedure. It is therefore better to use the *D*–optimal experimental design for RSM.

For details on the default algorithm, see the description of Algorithm 2 in Section 23.2.6. Other Latin Hypercube algorithms may be selected using the advanced options, Table 9-8.

All Latin Hypercube algorithms are repeatable, but a random number seed can be provided to create different sets of random points, Section 9.3.8.

Table 9-8: Latin Hypercube advanced options

Option	Description	Reference
Generalized	'Generalized' LHS design with random pairing	Section 23.2.5
Central Point	'Central point' Latin Hypercube Sampling (LHS) design with random pairing	Section 23.2.5

9.3.4. Space Filling

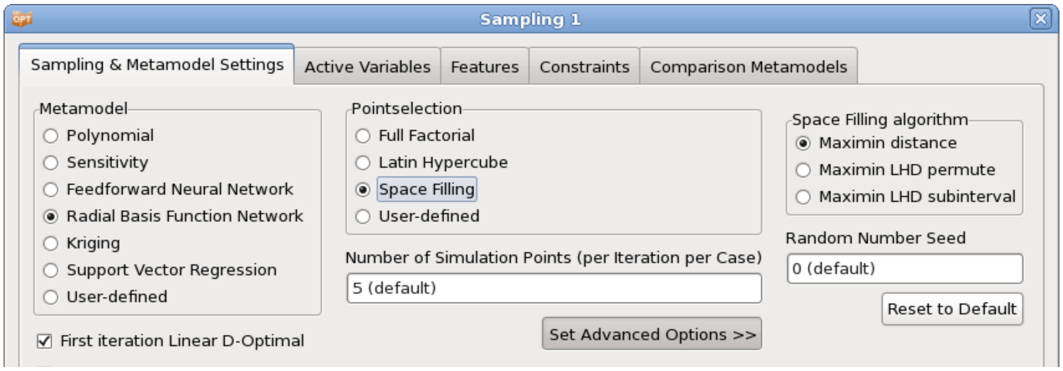
The default Space Filling algorithm maximizes the minimum distance between experimental design points for a given number of points. For details on the algorithm, see the description of Algorithm 5 in Section 23.2.6. Other space filling approaches may be selected using the advanced options, Table 9-9. The only data required is the number of sampling points that has to be specified in the Number of Simulation Points text field in the Sampling dialog. The default number of points depends on the number of variables, the metamodel type and also on the task and strategy. Space Filling is suitable for the Radial Basis Function, Neural Networks, Support Vector Regression as well as Kriging methods (see Section 9.1.3).

All space filling algorithms are repeatable, but a random number seed can be provided to create different sets of random points, Section 9.3.8.

Table 9-9: Space Filling advanced options

Option	Description	Reference
Maximin distance	Given an arbitrary design (and a set of fixed points), randomly moves the points so as to optimize the maximin distance criterion using simulated annealing.	Section 23.2.5

Maximin LHD permute	Given an LHS design, permutes the values in each column of the LHS matrix so as to optimize the maximin distance criterion taking into account a set of existing (fixed) design points. This is done using simulated annealing. Fixed points influence the maximin distance criterion, but are not allowed to be changed by Simulated Annealing moves.	Section 23.2.5
Maximin LHD subinterval	Given an LHS design, moves the points within each LHS subinterval preserving the starting LHS structure, optimizing the maximin distance criterion and taking into consideration a set of fixed points.	Section 23.2.5



Sampling 1

Sampling & Metamodel Settings | Active Variables | Features | Constraints | Comparison Metamodels

Metamodel
○ Polynomial
○ Sensitivity
○ Feedforward Neural Network
● Radial Basis Function Network
○ Kriging
○ Support Vector Regression
○ User-defined

☑ First iteration Linear D-Optimal

Pointselection
○ Full Factorial
○ Latin Hypercube
● Space Filling
○ User-defined

Number of Simulation Points (per Iteration per Case): 5 (default)

Set Advanced Options >>

Space Filling algorithm
● Maximin distance
○ Maximin LHD permute
○ Maximin LHD subinterval

Random Number Seed: 0 (default)

Reset to Default

Figure 9-12: Space Filling and Space Filling Advanced Options

9.3.5. Space Filling of Pareto Optimal Frontier

By selecting to create the Pareto Optimal Frontier (POF) as a strategy, a Space Filling algorithm which applies discrete Space Filling sampling of the POF is available. This sampling method uses the POF created in the previous iteration as a basis design point set. The distance between the points is maximized and can also be maximized with respect to previous simulation points by selecting to augment the design points. The user can specify the number of points required.

How to use the Pareto Optimal Frontier as a basis set for sampling

The following procedure can be followed to conduct simulations based on the POF. It is assumed that the user has conducted one or more metamodel-based iterations and that the POF has been created based on the metamodel.

1. *Task*: If not selected already, select any *Sequential* strategy in the *Task selection* dialog.
2. Sampling:
    a. Choose to conduct *Space Filling of Pareto Frontier* as a Sampling option.

b. Choose whether previous simulation points are to be considered in the Space Filling algorithm (check the box "*Include pts of Previous Iterations*").

c. Choose the number of simulation points required using the *Number of Simulation Points* textfield. The simulation will stop automatically if the POF basis set is too small.

d. If the number of simulations required differs from the current setting, choose "*Do not augment sampling before iteration*" in the Sampling dialog *Features* tab and set the iteration number at which you want to restart. For example, if one iteration is already available, set the starting iteration to 2, Section 9.5.4.

e. *Constraints:* The constraint values can be adjusted to filter POF points. Select those constraints which are to be applied as sampling filters as *Sampling Constraints* in the Sampling dialog Constraints tab, Section 9.6.

The constraints can be added or changed immediately before the final run, so do not have to be precise from the very beginning.

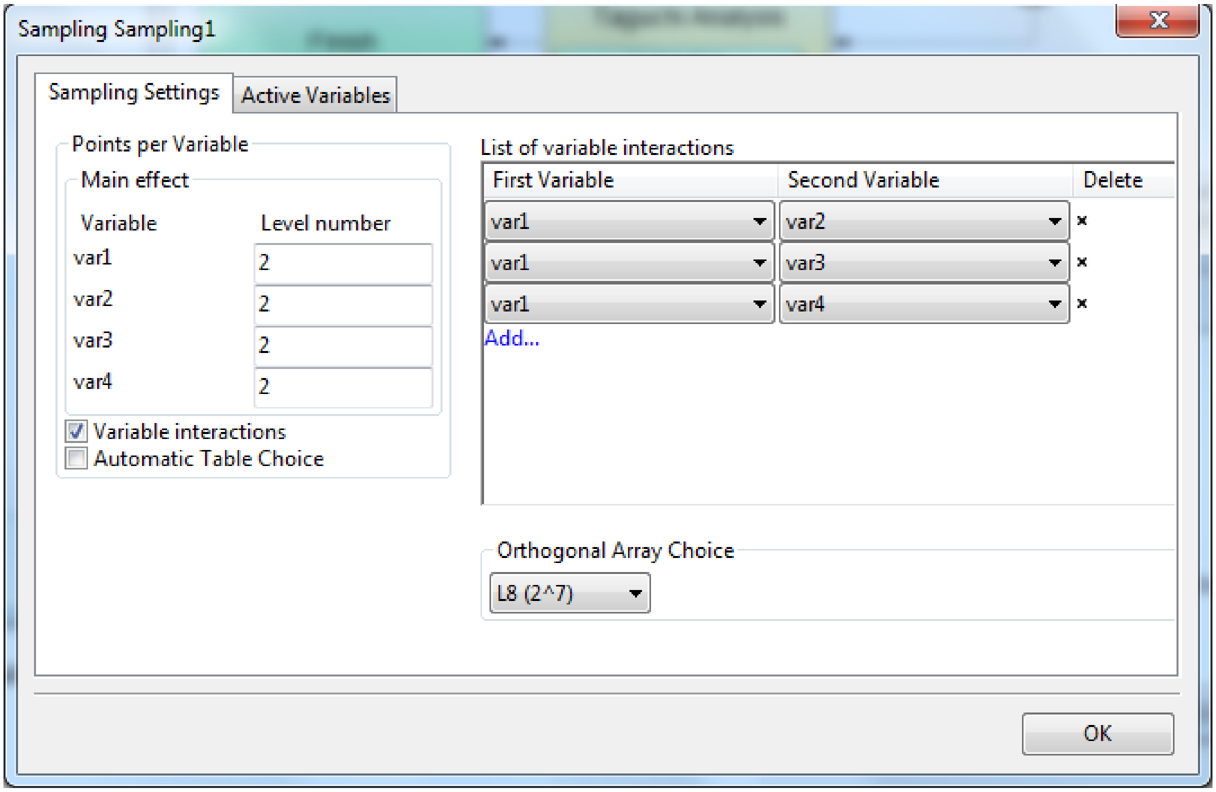
3. *Termination Criteria:* Increase the iteration limit by 1 assuming only 1 more iteration is to be done.
4. *Run:* To delete any existing runs which may exist in the current iteration (such as a previous verification run), choose *Clean from Current Iteration [it]* from the *Tools* menu and set the current iteration in the top menu bar.

9.3.6. Orthogonal Array

The Orthogonal Array sampling method is available in LS-OPT in case of a polynomial or user-defined metamodel-based optimization or DOE task, or in case of a direct Taguchi task.

It is possible to:

- Define the level number for each parameter (the level number is fixed for discrete variables and is equal to the number of discrete values defined);
- Add interactions between variables having only 2 levels. Two interaction tables are available: $L_8(2^7)$ and $L_{16}(2^{15})$;
- Select manually the Orthogonal Array. By default, LS-OPT automatically selects the best compatible Orthogonal Array.



Points per Variable

Main effect

Variable	Level number
var1	2
var2	2
var3	2
var4	2

☑ Variable interactions

☐ Automatic Table Choice

List of variable interactions

First Variable	Second Variable	Delete
var1	var2	×
var1	var3	×
var1	var4	×

Orthogonal Array Choice

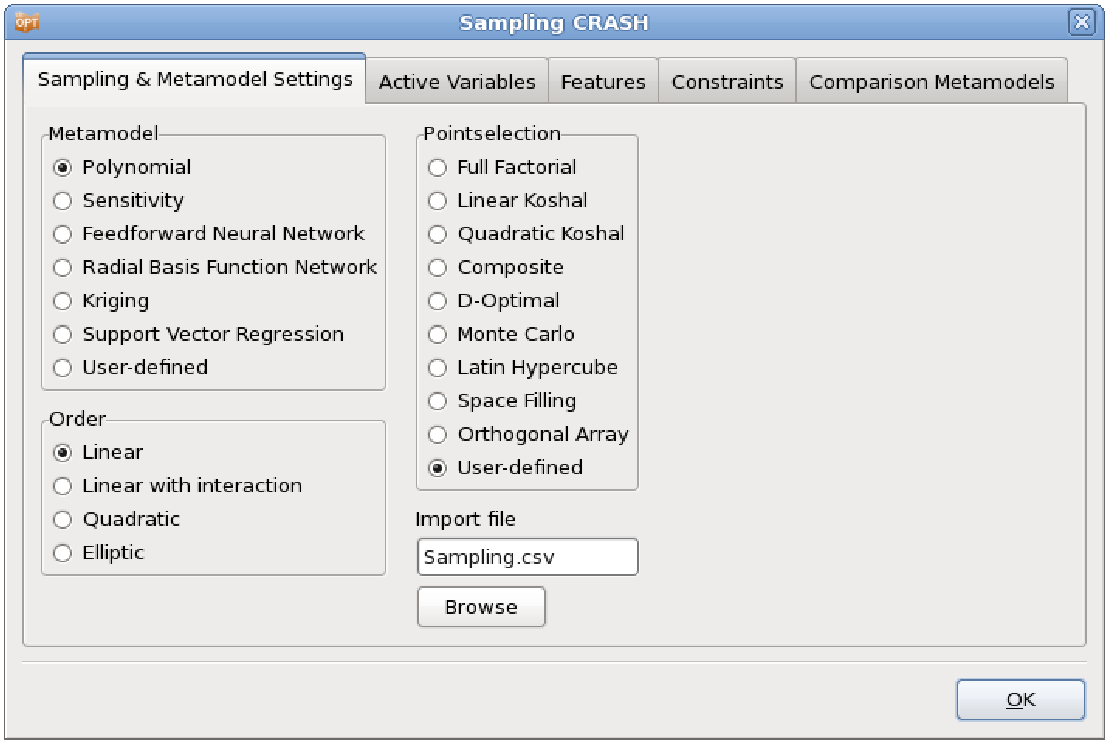
L8 (2^7)

Figure 9-13: Sampling Dialog for direct Taguchi task: Orthogonal Array point selection

Table 9-10: Orthogonal Array options

Option	**Description**	**Reference**
Variable interactions	Given an Orthogonal Array design, define interaction between 2-level variables. Interaction Orthogonal Array table will be used.	Section 23.2.7
Automatic Table Choice	Given an Orthogonal Array design, automatically choose the Orthogonal Array table. If not, a list of all possible Orthogonal Array tables is displayed.	Section 23.2.7

9.3.7. User-defined point selection



Sampling CRASH

Sampling & Metamodel Settings | Active Variables | Features | Constraints | Comparison Metamodels

Metamodel
- ● Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Order
- ● Linear
- ○ Linear with interaction
- ○ Quadratic
- ○ Elliptic

Pointselection
- ○ Full Factorial
- ○ Linear Koshal
- ○ Quadratic Koshal
- ○ Composite
- ○ D-Optimal
- ○ Monte Carlo
- ○ Latin Hypercube
- ○ Space Filling
- ○ Orthogonal Array
- ● User-defined

Import file: Sampling.csv

Browse

OK

Figure 9-14: Sampling Dialog: User-defined point selection

The User-defined point selection option allows the user to specify own sampling points. This may be useful if LS-OPT is used as a process manager. There are two formats supported to import the data, *csv* (comma separated variables) and a free format. Only files with ASCII characters are supported.

Comma separated variables

A user-defined experimental design can be specified in a text file using the .csv (comma separated variables) format. This allows the user to import a table from a text file with the following keyword-based format:

```
"Point","tbumper","thood",
"sk","dv","dv",
1,3.0000000000000000e+00,1.0000000000000000e+00,
2,5.0000000000000000e+00,1.0000000000000000e+00,
3,1.0000000000000000e+00,1.0000000000000000e+00,
4,1.0000000000000000e+00,5.0000000000000000e+00,
5,5.0000000000000000e+00,5.0000000000000000e+00,
```

The two header lines are required. The variable types are design variables (dv), noise variables (nv), discrete variables (dc) or string variables (st), respectively (see also Appendix E.4.1). It should be noted that the entry for a string variable is the corresponding mapped integer starting from 0. The variable names assure that each column is tied to a specific name and will be displayed as variables in the “Parameter Setup” panel in the Setup dialog. The variable types defined in the user file will take precedence over other type definitions of the same variable (e.g. from the input files).

The sk variable type can be used to screen out variables. Therefore variables of the sk type will not appear on the Parameter setup page when importing the file.

This format is convenient for use with Microsoft Excel which allows the export of a .csv text file. The browser for specifying an input file has a filter for .csv files. This feature is also ideal for setting up an LS-OPT run with using an exported file of Pareto Optimal points. Such a file can be produced using the Viewer.

Free format

A user-defined experimental design can also be specified in a text file using the following keyword-based free format:

```
lso_numvar 2
lso_numpoints 3
lso_varname                           t_bumper   t_hood
lso_vartype                           dv         nv
This is a comment     lso_point       1.0        2.0
                      lso_point       2.0        1.0
                      lso_point       1.0        1.0
```

The keywords (e.g. lso_numvar) except lso_vartype are required but can be preceded or followed by any other text or comments. The variable types are design variables (dv) or noise variables (nv) respectively. The variable names assure that each column is tied to a specific name and will be displayed as variables in the **Parameter setup** pane in the **Setup** dialog. The variable types defined in the user file will take precedence over other type definitions of the same variable (e.g. from the input files).

This format is convenient for use with Microsoft Excel which allows the export of a .txt text file. The browser for specifying an input file has a filter for .txt files.

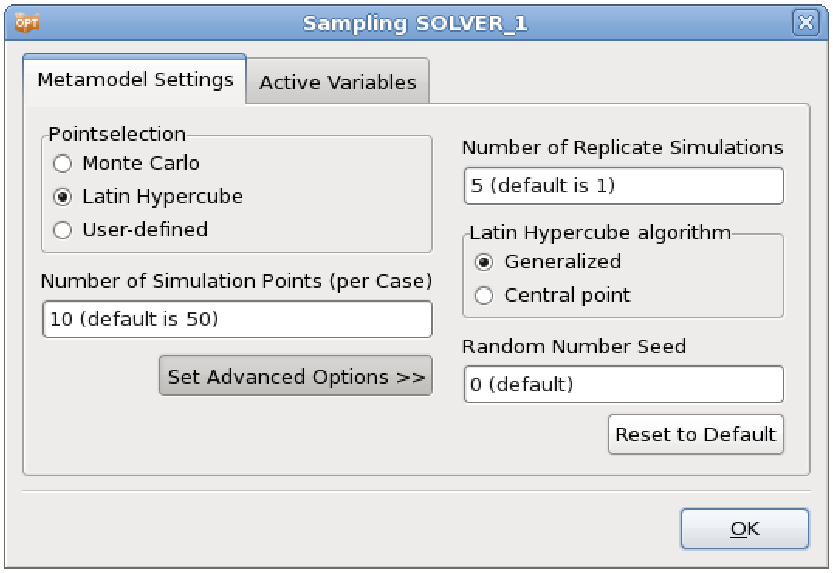
9.3.8. Advanced point selection options

Random number seed

All point selection schemes are repeatable, but a random number seed can be provided to create different sets of random points for methods that use randomness. The feature is particularly useful for Monte Carlo or Latin Hypercube point selections which both directly use random numbers. Because D-Optimal and Space Filling designs also use random numbers, albeit less directly, they may only show small differences due to the occurrence of local minima in the respective optimization procedures.

9.3.9. Replicate experimental points

For direct Monte Carlo analysis, when using stochastic fields, any particular design point can be (re-)analyzed using different stochastic fields. These are then replicate evaluations of the same design. The Number of Replicate Simulations can be specified in the Sampling dialog *Advanced Options*, Figure 9-15. The stochastic field is controlled using the LS-DYNA® *PERTURBATION and *PARAMETER cards. Note that the RND (random number seed) field of the card can be set to 0 to allow the field to vary freely, or set to a positive number to get a specific stochastic field.



Pointselection
○ Monte Carlo
● Latin Hypercube
○ User-defined

Number of Simulation Points (per Case): 10 (default is 50)

Number of Replicate Simulations: 5 (default is 1)

Latin Hypercube algorithm
● Generalized
○ Central point

Random Number Seed: 0 (default)

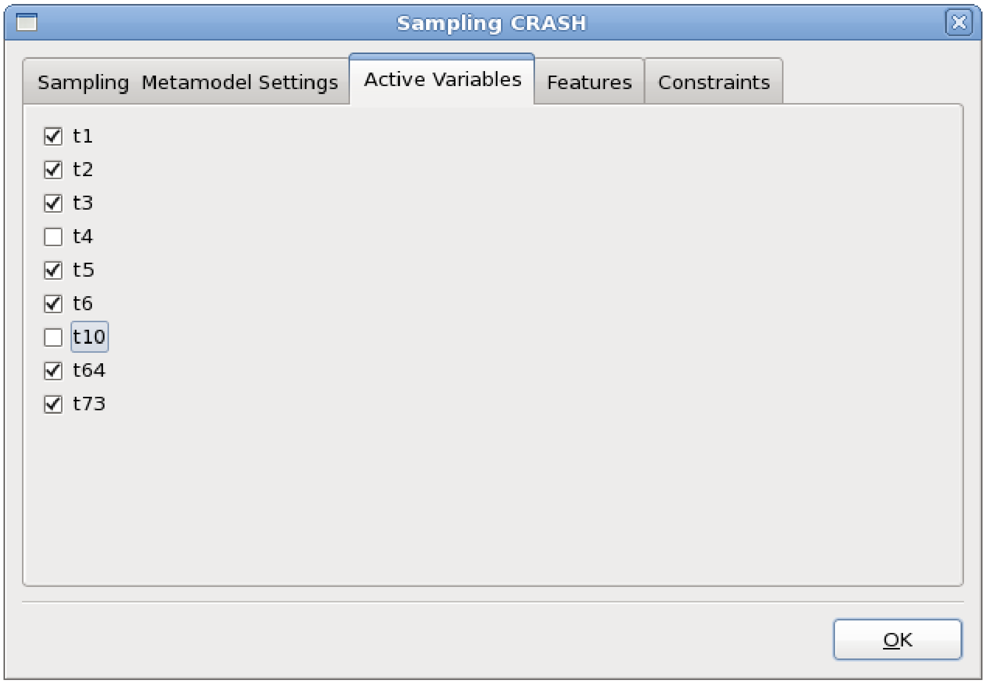
Figure 9-15: Sampling Dialog options for direct Monte Carlo Analysis

So, in the above, the original experimental design has 10 point, hence 50 FEA evaluations will be done. See also the example in Section 21.1.

9.3.10. Remarks: Point selection

1. The database files `Experiments_`*n*`.csv`, `AnalysisResults_`*n*`.lsda` and `AnalysisResults_`*n*`.csv` are synchronous, i.e. they will always have the same experiments after extraction of results. These files also mirror the result directories for a specific iteration.
2. Design points that replicate the starting point are omitted during the sampling phase.

9.4. Active Variables



☑ t1
☑ t2
☑ t3
□ t4
☑ t5
☑ t6
□ t10
☑ t64
☑ t73

Figure 9-16: Sampling Dialog: Active Variables panel

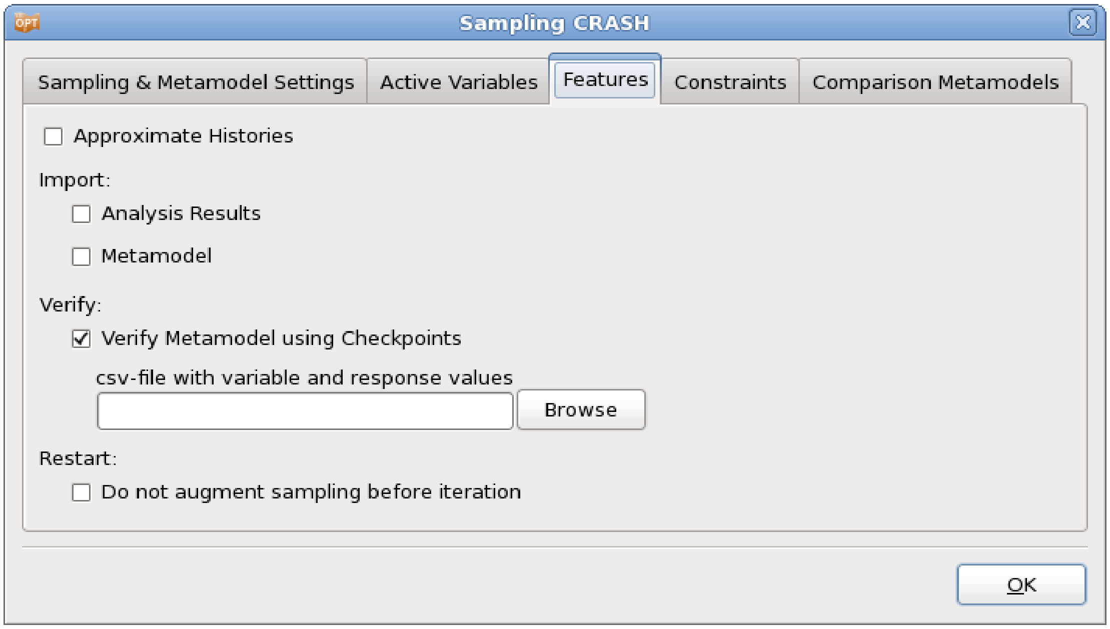
The *Active Variables* panel shows a list of all previously defined variables, Figure 9-16. Each variable has a checkbox that allows the user to select or deselect it for the respective sampling. Deselected variables are treated as constants using the optimal value of the previous iteration.

The selection in the *Active Variables* dialog is coupled to the respective column of the *Sampling Matrix* shown in the *Setup Dialog*, Section 8.3.

If a variable has been deselected across all the available samplings it will assume the baseline value over all iterations. It will therefore effectively be assumed to be a constant.

The active variable selection can also be changed between iterations.

9.5. Sampling Features



☐ Approximate Histories

Import:

☐ Analysis Results

☐ Metamodel

Verify:

☑ Verify Metamodel using Checkpoints

csv-file with variable and response values

Restart:

☐ Do not augment sampling before iteration

Figure 9-17: Sampling Features

Table 9-11: Sampling Features

Feature	Description	Reference
Approximate Histories	Extension of the metamodel concept to curves.	Section 9.5.1
Import Analysis Results	Import table of design points (variable and response values)	Section 9.5.3
Import Metamodel	Import previsouly generated metamodel	Section 9.5.4
Verify Metamodel using Checkpoints	Calculate error measures of the metamodel using a given metamodel and set of checkpoints (variables and response values)	Section 9.5.2
Restart: Do not augment sampling before iteration	Use larger number of sampling points from a specified iteration	Section 9.5.5

9.5.1. Approximate histories

Each history curve can be pointwise (at each sampled time-step) approximated using metamodels. These approximations of the entire history curves in time-domain are called predicted histories. These history approximations are used to study the influence of changes in the variables as well as for parameter identification problems. The approximation of histories is enabled by setting the *Approximate Histories* flag on the *Features* page of the *Sampling* dialog as shown in Figure 9-17. The user can approximate the data using either linear or quadratic polynomials or by radial basis functions. The approximations are carried out on the sampling points used for response approximations. While the approximation models for the histories and responses can be different, the number and location of sampling points remain the same such that all options for history approximation may not be suitable depending on the number of available data points, for example, if the response sampling is linear polynomial the number of points sampled would not be sufficient to approximate the histories using a quadratic polynomial and that option should be avoided. It is also important to note that approximation of histories may take significantly long as approximations at thousands of time-steps are carried out.

Remarks

1. It is assumed that the each history curve has the same number of time-steps for all points.
2. For sequential strategies, all points sampled so far would be used for creating RBF approximations, whereas only the points sampled in the current iteration are used for polynomial approximations.

9.5.2. Verify Metamodel using Checkpoints

The error measures of any number of designs (checkpoints) can be evaluated using an existing metamodel. There are two simple steps to obtaining a table with error data.

Browse for the file with the checkpoint information using the "*Verify Metamodel using Checkpoints*" option in the **Features** tab in the **Sampling** or the **Metamodeling** panel. The file must be in `.csv` format although spaces, commas or tabs are allowed as delimiters. Only files with ASCII characters are supported. The file must contain two header lines. The first header line contains the variable and response names. The second header line contains the variable and response types; in this case "`dv`", ”nv”, “dc” and “st” for variables and "rs" for responses (see also Appendix E.4.1). For string variables (“st”), the corresponding mapped integer values need to be provided. The mapping is stored in the file *StringVar.lsox*. The variable coordinates are then specified as one row for each design point. See example below.

Use the **Evaluate Metamodels** option from the **Tools** menu **Repair** option or the respective command line option to run (see Section 3.5 and Section 2.2.1).

Cases without checkpoint files will be ignored.

The results are available in lsopt_report.

Example of a checkpoints file:

```
x1, x2, x3, Disp, Acc
dv, dv, dv, rs, rs
1.0, 1.3, 1.2, 123.6, 1278654.7
2.1, 2.2, 639.2, 2444588.1
```

9.5.3. Importing user-defined analysis results

A table (in text form) of existing analysis results can be used for analysis.

Browse for the file with the analysis results to import using the *Import User Results* option in the **Features** tab in the **Sampling** panel. Only files with ASCII characters are supported.

Two header lines are required. The first header line contains the variable names. The second header line contains the variable types. The following lines contain the variable and response values for each design point, see example below. The types are defined as described in Table 9-12 (see also Appendix E.4.1). The parsing code looks for *double quotes*, *commas*, *spaces* and/or *tabs* as delimiters.

Table 9-12: Variable types

Symbol	**Explanation**
dv	Design variable
nv	Noise variable
rs	Response
dc	Discrete variable
sk	Ignore

Example

An example of a analysis results file (with 2 simulation points) is:

```
"var1","var2","var3","Displacement","Intrusion","Acceleration"
"dv",  "dv",  "nv",  "rs",          "rs",        "rs"
1.23   2.445  3.456  125.448        897.2        223.0
0.01,2.44,1.1,133.24,244,89,446.6
```

The steps for importing user-defined analysis result files are as follows:

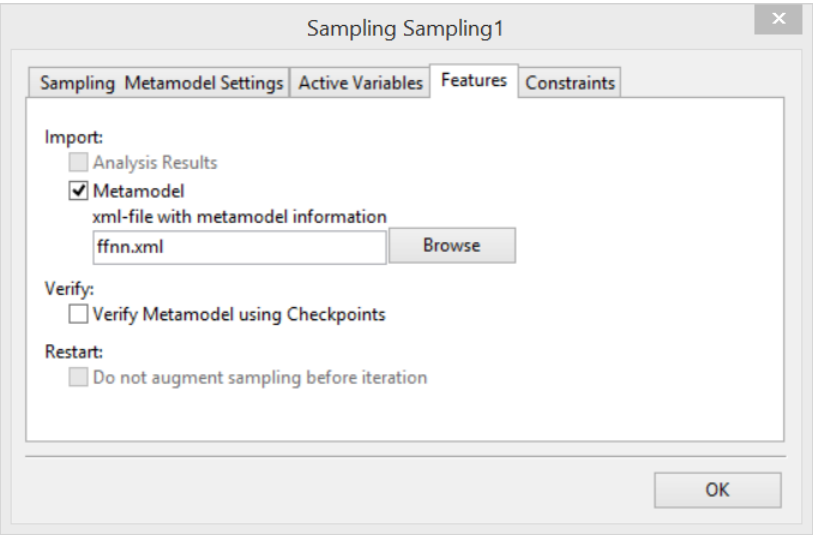
1. *Sampling panel, Features tab:* Browse for the text file in the *Import User Results* textfield. The browser prefers `.csv` and `.txt` files. Variables and responses are imported automatically into the GUI, the responses are added to the first stage of the respective sampling.
2. *Sampling panel.* Check that the number of points defined in the sampling panel is the same as the number of points in the user-provided file. If fewer points are available in the file, LS-OPT will augment the sampling points and try to run simulations.
3. *Sampling pane, right mouse menu.* Select "Repair", "Import results", Section 3.5, or use the respective command line option, Section 2.2.1. This is a critical step to convert the `.csv` format to the LS-OPT database format ready for analysis.
4. The user can now choose the type of analysis in the *Task* dialog.
    a. DOE Study: Change to the *Metamodel-based DOE Study* task and *Run*. Metamodels will be created and the Viewer can be used to study the metamodel results.
    b. Optimization: Define the Objectives and/or constraints. For RBDO, define the distributions for the input variables as well as the probability of failure.

Change to the *Metamodel-based Optimization* or *Metamodel-based RBDO* task, choose the *Single Stage* strategy and *Run*. An optimization history is created.

9.5.4. Importing metamodels

Metamodels can be imported for the purpose of performing metamodel-based tasks such as optimization or reliability analysis. Only files in the LS-OPT DesignFunctions.*x* format (xml format) can be imported. Figure 9-18 shows the feature for browsing an input file. To import the file, select the **Import Metamodels** option in the global **Repair** option found in the menu bar. The import repair feature can also be found by right-clicking on the **Metamodel** dialog box and selecting **Repair→Import**, Section 3.5. The respective command line option can also be used, Section 2.2.1.

An automatic import feature can be selected in the **Task** settings (**Import metamodel**). This feature can be used to automatically activate the metamodel import function as a pre-processor task before executing other tasks such as optimization or reliability analysis. This feature is useful when performing inner level tasks in a multi-level optimization based on existing metamodels. In such cases it is not possible to manually import a metamodel file for each inner level run.



Import:
□ Analysis Results
☑ Metamodel
xml-file with metamodel information
ffnn.xml

Verify:
□ Verify Metamodel using Checkpoints

Restart:
□ Do not augment sampling before iteration

Figure 9-18: Sampling Dialog: Selection of metamodel import feature

9.5.5. Changing the number of points on restart*

The number of points to be analyzed can be changed starting with any iteration. This feature is useful when the user wants to restart the process with a different (often larger) number of points. The option *Do not augment sampling before iteration* avoids adding points in iterations prior to the specified iteration. The feature is sampling-specific, so must be added to all the sampling definitions.

Example 1

In the first analysis, the following sampling scheme was specified: a single iteration with 5 D-optimal points was performed. By default, a single verification run is done in iteration 2.

After the first analysis, the user wants to restart, using 10 points per iteration and 3 iterations in total. *Do not augment sampling before iteration* is set to 2. Iterations 2 and 3 will then be conducted with 10 points each while iteration one will be left intact.

Example 2

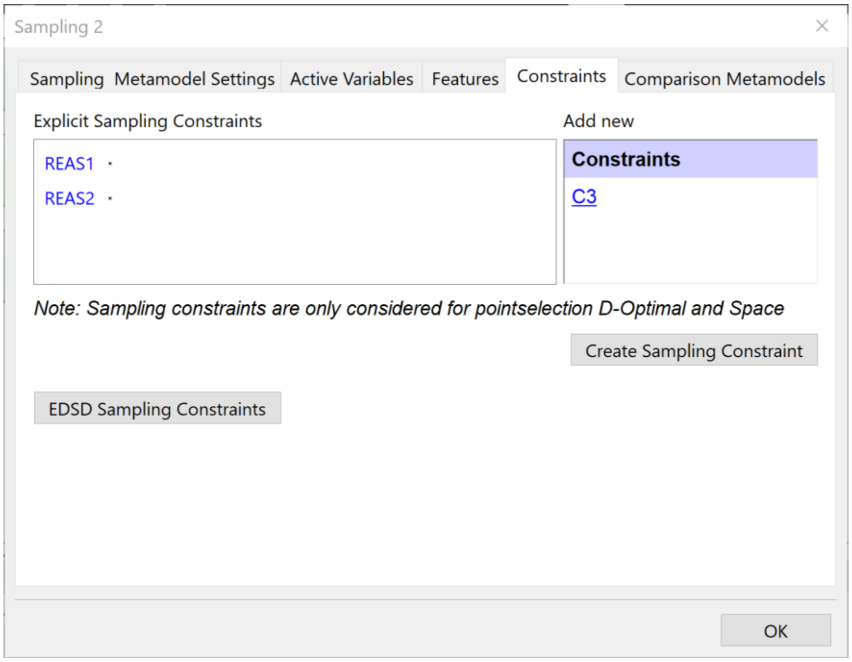
Starting with a single iteration with 5 *D*-optimal points and restarting with 10 *D*-optimal points, but now, *Do not augment sampling before iteration* is set to 1. Iteration 1 of the restart will be augmented with 5 points (to make a total of 10), before continuing with 10 points in further iterations.

Note: The user will have to delete the single verification point generated in the first analysis before restarting the run. This can be done by using the *Clean Verification Run* option. The restart will then generate a *new* starting point for iteration 2 and conduct 10 simulations altogether.

9.6. Sampling Constraints

Sampling constraints are used to specify an irregular design space. An irregular (reasonable) design space refers to a region of interest that, in addition to having specified bounds on the variables, is also bounded by arbitrary constraints. This may result in an irregular shape of the design space. This region of interest is thus defined by constraint bounds and by variable bounds. The purpose of an irregular design space is to avoid designs which may prove to be impossible to analyze or to select designs in specific regions of importance. Sampling constraints are defined in the **Constraints** tab of the **Sampling** dialog, Figure 9-19. Previously defined constraints are available for selection in the *Add new* list, new constraints may be defined using the Sampling constraint wizard, Figure 9-20, accessible by the *Create Sampling Constraint* button.

Only explicit constraints, i.e. constraints that do not require simulations, can be specified for the reasonable design space using the *Create Sampling Constraint* button; others are discussed in Section 9.7. A typical explicit constraint could be the difference of two design variables with a specified numeric upper or lower bound. That would represent a simple inequality relationship between the design variables.

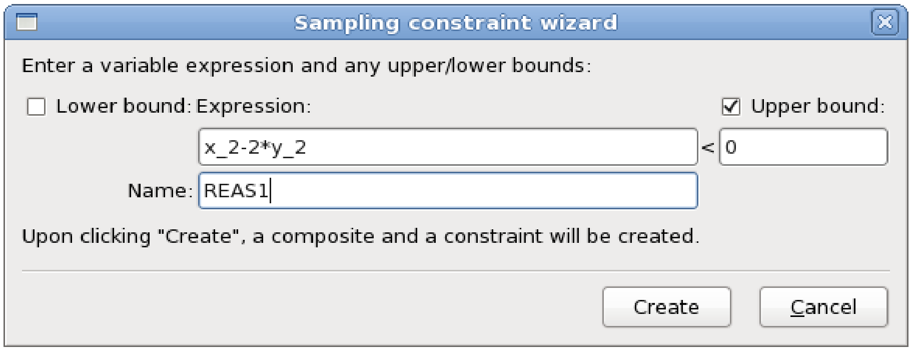


REAS1 ·
REAS2 ·

Constraints
C3

Note: Sampling constraints are only considered for pointselection D-Optimal and Space

Figure 9-19: Definition of Sampling Constraints by selection from list or new creation.



Sampling constraint wizard

Enter a variable expression and any upper/lower bounds:

□ Lower bound: Expression: ☑ Upper bound:

x_2-2*y_2 < 0

Name: REAS1

Upon clicking "Create", a composite and a constraint will be created.

Figure 9-20: Sampling constrain wizard: definition of an expression and bounds

This specification of the Sampling constraint ensures that the points are selected such that the bounds are not violated.

Remark

A reasonable design space can be created using the *D*-optimal experimental design as well as the Space Filling experimental design. These are the most used options that accompany the choice of polynomials, Radial Basis Function Networks, Neural Networks or Kriging as metamodels.

9.7. Adaptive Sampling - EDSD Sampling Constraints

As discussed in Section 9.6, sampling constraints can be defined explicitly in terms of the design variables if the reasonable design space is known a priori. In some scenarios, however, this knowledge is not available beforehand. For instance, some combinations of design variables can lead to failure without prior knowledge and the information about such combinations becomes available only through the evaluation of samples. The reasonable design space boundary can then be approximated using a machine learning algorithm based on the available information; as more samples are evaluated, the accuracy of the approximated reasonable design space increases. Thus, two types of sampling constraints are currently supported:

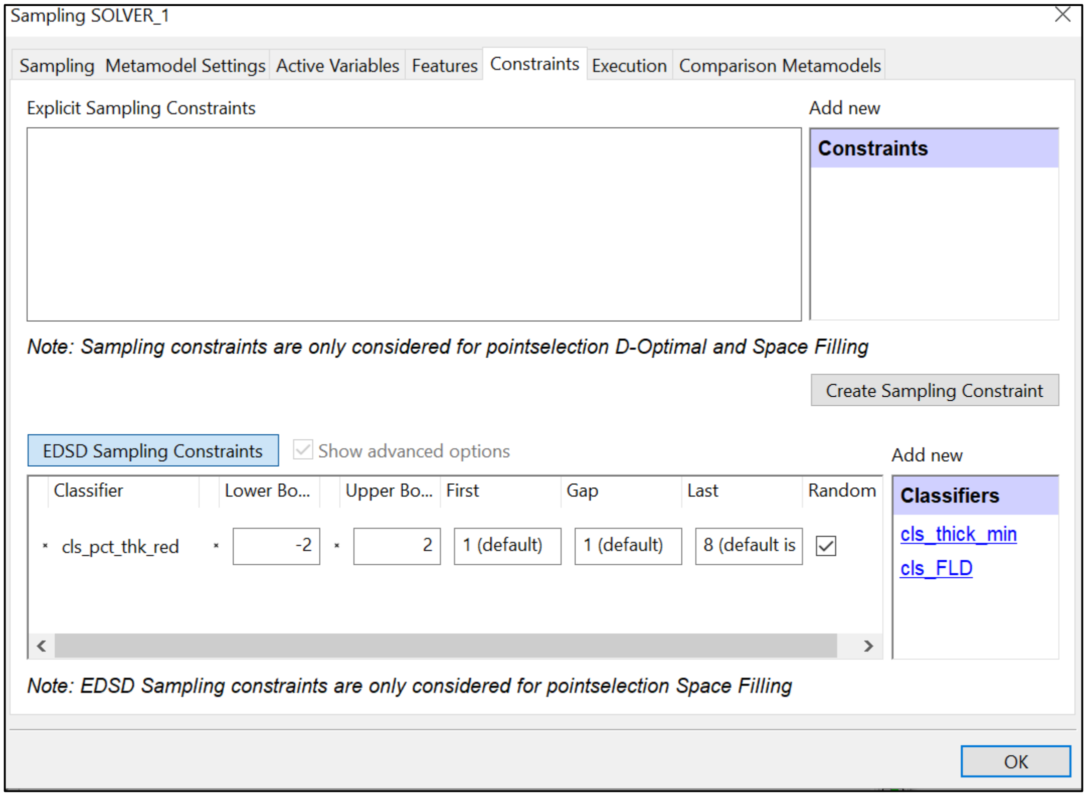
- *Explicit constraints defined as functions of design variables* ($g(\boldsymbol{x}) \leq 0$) for cases where the reasonable sampling domain is known a priori without the need of simulations (Section 9.6).
- *Classifier-based sampling constraints*, referred to as explicit design space decomposition (EDSD) sampling constraints, to approximate unknown sampling domain boundaries ($s(\boldsymbol{x}) \leq 0$). Here $s(\boldsymbol{x})$ is the predicted classifier value at design $\boldsymbol{x}$. The EDSD sampling constraints are activated only while selecting samples and are not applied as design constraints. The underlying classifier can be defined as a design constraint by the user, if needed.

EDSD sampling constraints are useful for adaptive sampling. A user can specify constraints depending on the type of problem (e.g. single-/multi-objective optimization/reliability assessment) to avoid selection of samples that are not expected to provide additional relevant information. For instance, the immediate vicinity of the limit state boundary is most important for reliability assessment, while the regions likely to be feasible are important for optimization. This topic is discussed further in 25.4.2.

By default, all the samples are subject to the sampling constraints, but these constraints can also be activated selectively for specific samples of an iteration using advanced options. Additionally, it is possible to randomize the lower and upper bounds. The advanced options are listed in Table 9-13. An example of classifier-based sampling constraints is shown in Figure 9-21.

Table 9-13: EDSD Sampling Constraints Advanced options

Option	**Description**
First	First sample of an iteration for which the constraint is applied
Gap	Constraint application frequency. E.g. First = 2, Last = 7, Gap = 2 implies that the constraint will be considered only for sample numbers 2, 4, 6.
Last	Last sample of an iteration for which the constraint is applied
Random	Bounds are randomized between the specified lower and upper bounds. Thus, different samples will have different bounds for the constraint.



Note: Sampling constraints are only considered for pointselection D-Optimal and Space Filling

☑ Show advanced options

Classifier	Lower Bo...	Upper Bo...	First	Gap	Last	Random
cls_pct_thk_red	-2	2	1 (default)	1 (default)	8 (default is	☑

Note: EDSD Sampling constraints are only considered for pointselection Space Filling

Figure 9-21: Definition of classifier-based EDSD sampling constraints

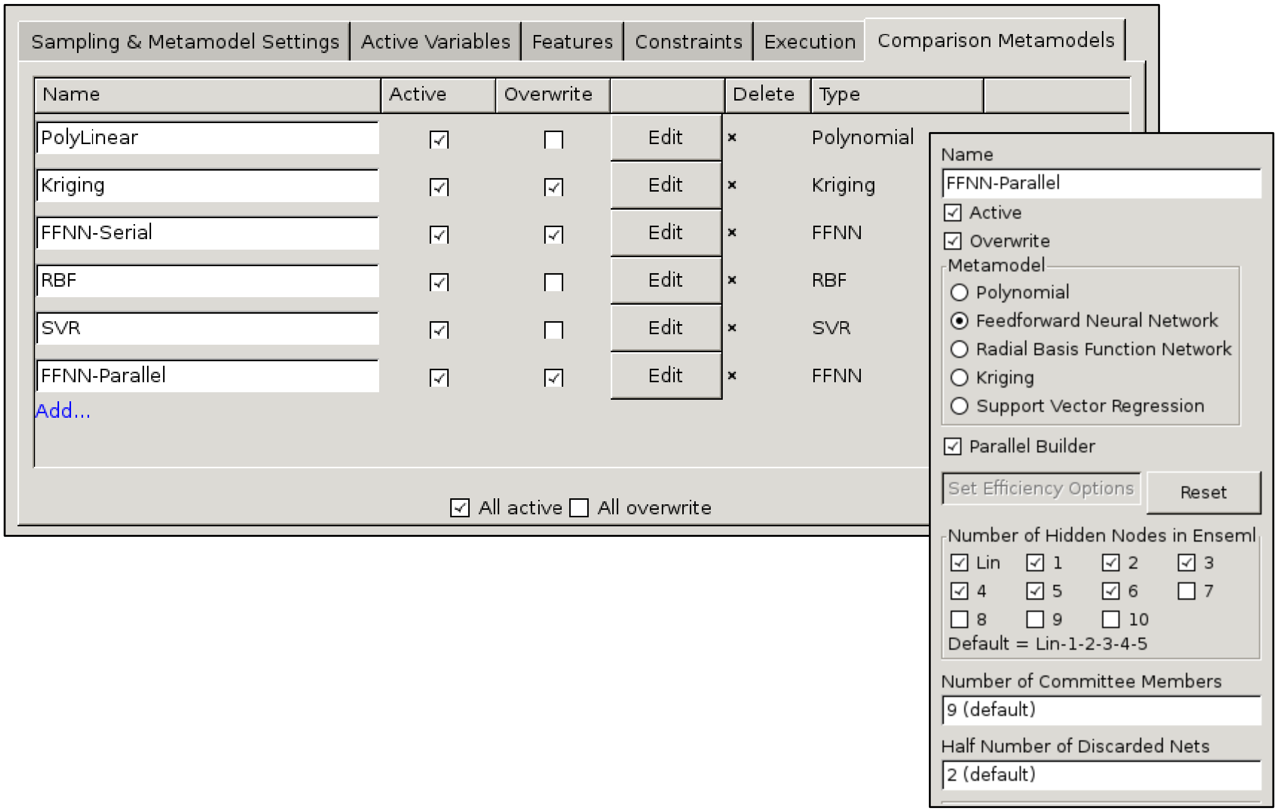
Remark

EDSD sampling constraints are applied only for space-filling sampling, which is commonly used for the construction of classifiers. These constraints will be ignored for other sampling types.

9.8. Comparison metamodels

Additional metamodels, not used in optimization or metamodel-based Monte Carlo analysis, can be created. These metamodels are based on the same analysis result set as the main metamodel and can be used for comparison, Figure 9-22. Comparison metamodels are calculated in all iterations.

A comparison metamodel is identified by a user-provided name and can be de-activated or overwritten. The attributes of a comparison metamodel can be edited as shown for e.g. Feedforward Neural Networks in Figure 9-22.



Name	Active	Overwrite		Delete	Type
PolyLinear	☑	□	Edit	×	Polynomial
Kriging	☑	☑	Edit	×	Kriging
FFNN-Serial	☑	☑	Edit	×	FFNN
RBF	☑	□	Edit	×	RBF
SVR	☑	□	Edit	×	SVR
FFNN-Parallel	☑	☑	Edit	×	FFNN

Add...

☑ All active □ All overwrite

Name
FFNN-Parallel
☑ Active
☑ Overwrite

Metamodel
○ Polynomial
● Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression

☑ Parallel Builder

Set Efficiency Options   Reset

Number of Hidden Nodes in Enseml
☑ Lin ☑ 1 ☑ 2 ☑ 3
☑ 4 ☑ 5 ☑ 6 □ 7
□ 8 □ 9 □ 10
Default = Lin-1-2-3-4-5

Number of Committee Members
9 (default)

Half Number of Discarded Nets
2 (default)

Figure 9-22: Definition of comparison metamodels and of the attributes of a selected comparison metamodel.

10. Composite Dialog

Composite functions can be used to combine response surfaces and variables as well as other composites. The response components can belong to any stage. The objectives and constraints can then be constructed using the composite functions.

10.1. Introduction

10.1.1. Composite vs. response expressions

There is an important distinction between response expressions and composites. This distinction can have a major impact on the accuracy of the result. Response expressions are converted to response surfaces after applying the expression to the results of each sampling point in the design space. Composites, on the other hand, are computed by combining response surface results. Therefore, the response expression will always be of the same order as the chosen response surface order while the composite can assume any complexity depending on the formula specified for the composite (which may be arbitrary).

Example

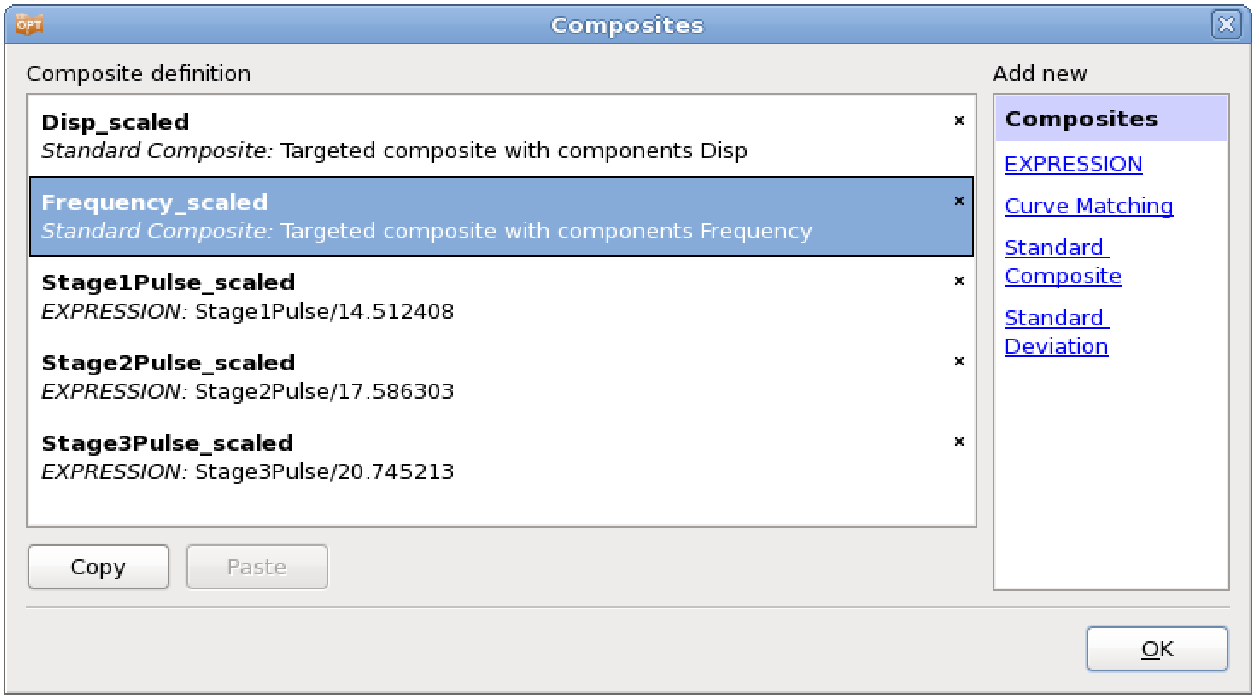
If a response function is defined as $f(x, y) = xy$ and linear response surfaces are used, the response expression will be a simple linear approximation $ax + by$ whereas a composite expression specified as xy will be exact.

10.2. Defining Composites

A composite can be defined by using the interfaces in the **Composites** dialog, Figure 10-1. To add the first Composite, select *Add Composite* from the main GUI control bar *Add* (**+**) menu. To add a new definition, select the respective interface from the list on the right. The available interfaces are explained in Table 10-1. To edit an already defined composite, double-click on the respective entry from the list on the left. Composites may be deleted using the *delete* icon on the right of the respective definition.

Remarks

1. An objective definition involving more than one response or variable requires the use of a composite function.
2. In addition to specifying more than one function per objective, multiple objectives can be defined, Section 12.2.



Disp_scaled
Standard Composite: Targeted composite with components Disp

Frequency_scaled
Standard Composite: Targeted composite with components Frequency

Stage1Pulse_scaled
EXPRESSION: Stage1Pulse/14.512408

Stage2Pulse_scaled
EXPRESSION: Stage2Pulse/17.586303

Stage3Pulse_scaled
EXPRESSION: Stage3Pulse/20.745213

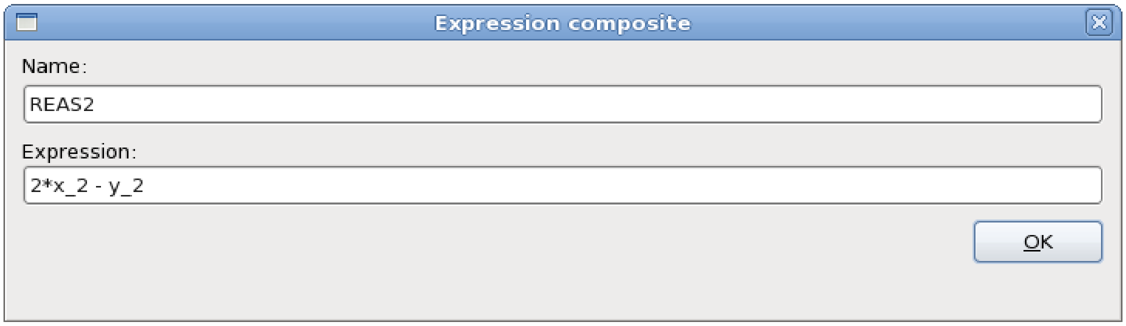
Figure 10-1: Composites Dialog

Table 10-1: Composite types

Composite type	**Description**	**Reference**
EXPRESSION	Mathematical expression using previously defined entities	Section 10.3
Curve Matching	Curve matching metrics	Section 10.5
Standard Composite	Weighted or targeted composites	Section 10.4
Standard Deviation	Standard deviation of another response or composite	Section 0
Copy	Copy the selected Composite	
Paste	Paste a previously copied Composite. The next free number is automatically appended to the name.	

10.3. Expression composite

A mathematical expression can be specified for a composite. The composite can therefore consist of previously defined constants, variables, dependent variables, responses and other composites (see Appendix F: Mathematical Expressions).



Name:

REAS2

Expression:

2*x_2 - y_2

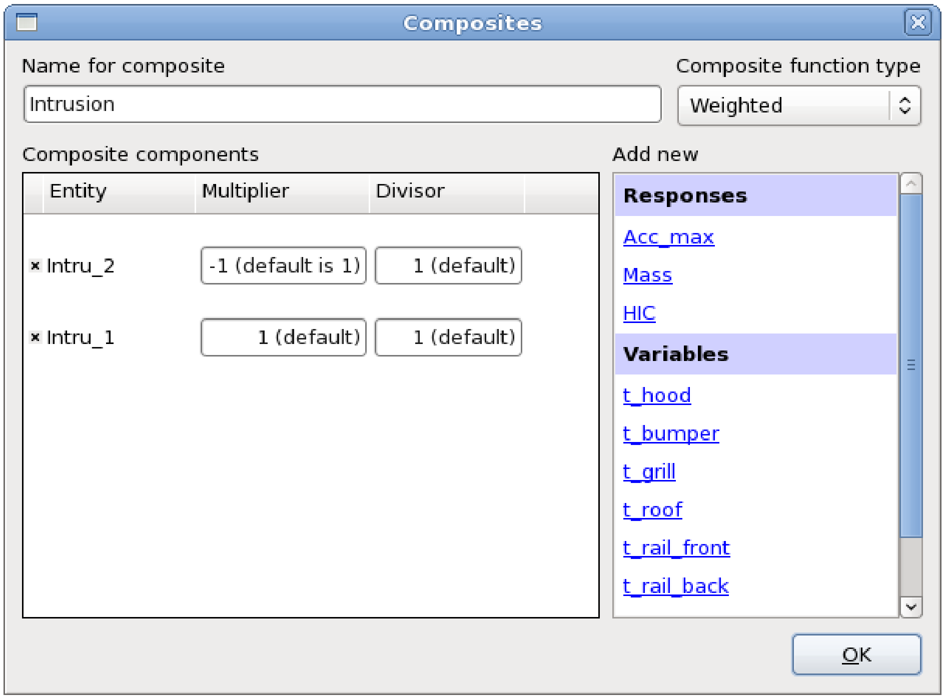
Figure 10-2: Definition of a Composite Expression

10.4. Standard composite

The Standard composite dialog is displayed in Figure 10-3. First the composite function type has to be selected, Table 10-2. Then select the Response or Variable components to be used to calculate the composite from the list on the right. The selected components appear in the list on the left with text fields to specify weighting and scaling factors and target values, respectively. Selected components can be deleted from the list by using the *delete* icon on the left of the entity name.

The composite function types are explained in detail in the following sections, Table 10-2.

Note that each formulation could alternatively be defined as a composite expression, examples are given in the following sections. Using the Standard Composite interface is convenient in many cases.



Name for composite: Intrusion

Composite function type: Weighted

Composite components

Entity	Multiplier	Divisor
× Intru_2	-1 (default is 1)	1 (default)
× Intru_1	1 (default)	1 (default)

Add new

Responses

Acc_max

Mass

HIC

Variables

t_hood

t_bumper

t_grill

t_roof

t_rail_front

t_rail_back

Figure 10-3: Standard Composite Interface

Table 10-2: Standard Composite function types

Composite function type	Reference
Weighted	Section 10.4.3
MSE	Section 10.4.2
Sqrt MSE	Section 10.4.1

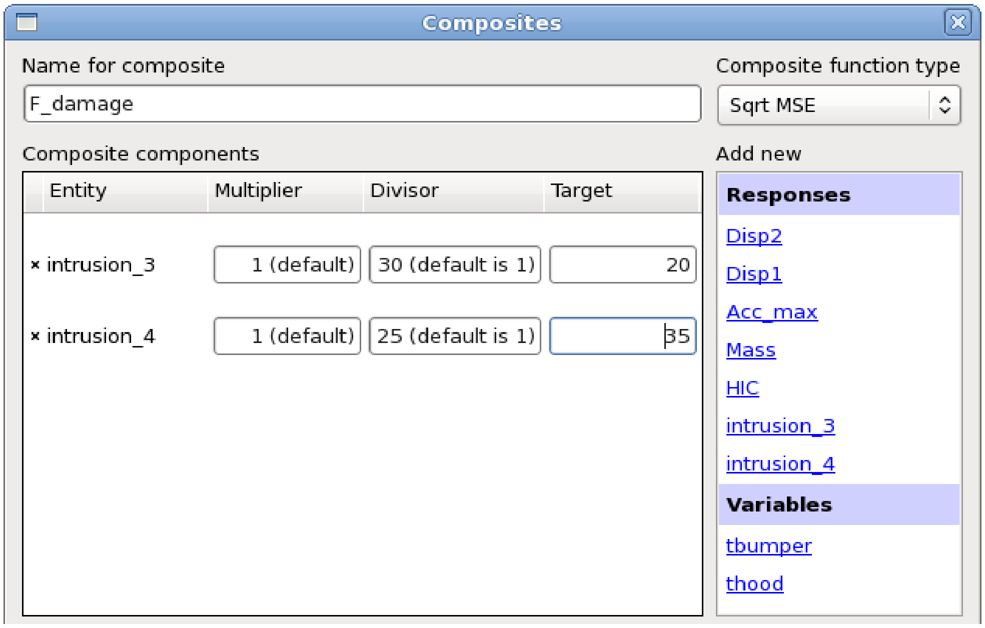
10.4.1. Targeted composite (square root of MSE)

This is a standard composite in which a target is specified for each response or variable in the *Target* text field. The composite is formulated as the 'distance' to the target using a Euclidean norm formulation. The components can be weighted and normalized.

$$F = \sqrt{\sum_{j=1}^{m} W_j \left[\frac{f_j(\boldsymbol{x}) - F_j}{\sigma_j}\right]^2 + \sum_{i=1}^{n} \omega_i \left[\frac{x_i - X_i}{\chi_i}\right]^2}$$

where σ and χ are scale factors (to be specified in the *Divisor* text fields) and W and ω are weight factors (to be specified in the *Multiplier* text fields). These are typically used to formulate a multi-objective optimization problem in which F is the distance to the target values of design and response variables.

In the GUI this type is selected as the *Sqrt MSE* composite function type.



Name for composite: F_damage

Composite function type: Sqrt MSE

Composite components

Entity	Multiplier	Divisor	Target
× intrusion_3	1 (default)	30 (default is 1)	20
× intrusion_4	1 (default)	25 (default is 1)	35

Add new

Responses
Disp2
Disp1
Acc_max
Mass
HIC
intrusion_3
intrusion_4

Variables
tbumper
thood

Figure 10-4: Definition of targeted (Root MSE) composite response in LS-OPTui

A suitable application is parameter identification. In this application, the target values F_j are the experimental results that have to be reproduced by a numerical model as accurately as possible. The scale factors σ_j and χ_i are used to normalize the responses. The second component, which uses the variables, can be used to regularize the parameter identification problem. Only independent variables can be included. See Figure 10-4 for an example of a targeted composite response definition. Here, F_damage will be calculated as

$$F_{damage} = \sqrt{\left[\frac{intrusion_3 - 20}{30}\right]^2 + \left[\frac{intrusion_4 - 35}{25}\right]^2}$$

The equivalent expression composite is:

```
sqrt(((intrusion_3 - 20)/30)**2 + ((intrusion_4 + 35)/25)**2)}
```

10.4.2. Mean squared error composite

This standard composite is the same as the targeted composite, except that the square root operation is omitted. This allows for composites to be added to make a larger composite (similar to the vector ordinate-based Mean squared error composite in Section 10.5.1).

10.4.3. Weighted composite

Weighted response functions and independent variables are summed in this standard composite. Each function component or variable is scaled (to be specified in the *Divisor* text fields) and weighted (to be specified in the *Multiplier* text fields).

$$F = \sum_{j=1}^{m} W_j \frac{f_j(\boldsymbol{x})}{\sigma_j} + \sum_{i=1}^{n} \omega_i \frac{x_i}{\chi_i}$$

These are typically used to construct objectives or constraints in which the responses and variables appear in linear combination. An example is given in Figure 10-3.

The equivalent expression composite is

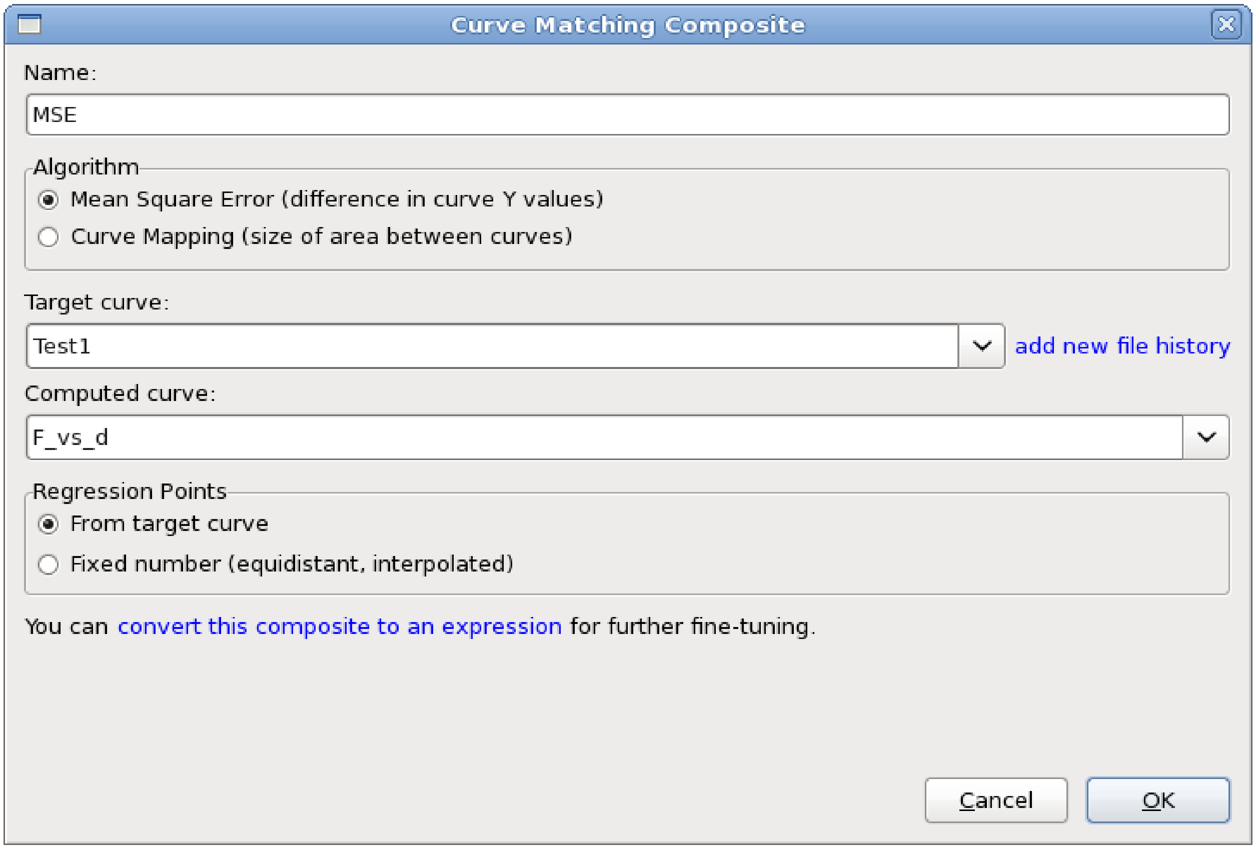
```
Intru_1 – Intru_2.
```

Needless to say, this is the preferable way to define this composite.

10.5. Curve Matching Composite

The Curve Matching interface provides two metrics for comparison of a target curve and curves extracted from simulation runs, Figure 10-5. The options are explained in Table 10-3.

To evaluate these composites, predicted histories (histories approximated by metamodels) are used, see Section 9.5.1 for details.



Curve Matching Composite

Name:

MSE

Algorithm

● Mean Square Error (difference in curve Y values)

○ Curve Mapping (size of area between curves)

Target curve:

Test1

add new file history

Computed curve:

F_vs_d

Regression Points

● From target curve

○ Fixed number (equidistant, interpolated)

You can convert this composite to an expression for further fine-tuning.

Figure 10-5: Curve Matching Composite Dialog

Table 10-3: Curve Matching Composite options

Option	**Description**	**Reference**
Algorithm	Curve matching metric to calculate “distance” between target and computed curve: ○ Mean Square Error (Ordinate-based) ○ Curve Mapping	 Section 10.5.1 Section 10.5.2
Target Curve	Previously defined File history containing target values.	
add new file history	If the file history to be used as Target curve is not already defined, this can be done here.	Section 6.16
Computed curve	Previously defined history or Crossplot extracted from simulation results	
Regression points	Regression points used to calculate composite:	

	From target curve Fixed number (equidistant, interpolated)	
convert this composite to an expression	Use a composite expression to define curve matching metric to be able to add further arguments	Appendix F:

10.5.1. Ordinate-based Curve Matching

A composite function is provided to compute the Mean Squared Error ε for the discrepancy between two curves

$$\varepsilon = \frac{1}{P}\sum_{p=1}^{P} W_p \left[\frac{f_p(\mathbf{x}) - G_p}{s_p} \right]^2 = \frac{1}{P}\sum_{p=1}^{P} W_p \left[\frac{e_p(\mathbf{x})}{s_p} \right]^2 \qquad (10\text{-}1)$$

It is constructed so that G_p , p=1, …, P are the values on the target curve G and $f_p(\mathbf{x})$ the corresponding components of the computed curve f. $f_p(\mathbf{x})$ are represented internally by response surface values. **x** is the design vector. See Section 28.2 for more detail.

Note

1. Only points within range of both curves are included in Equation (10-1), so P will be automatically reduced during the evaluation if there are missing points. A warning is issued in `WARNING_MESSAGE`.
2. The `Mean Square Error` composite makes use of response surfaces to avoid the nonlinearity (quadratic nature) of the squared error functional. Thus if the response curve f(**x**) is linear in terms of the design variables **x**, the composite function will be exactly represented.
3. `Mean Square Error` composites can be added together to make a larger MSE composite (e.g. for multiple test cases).
4. The simplest target curve that can be defined has only one point.
5. Ordinate-based Curve Matching should *not* be used for a non-monotonic abscissa (e.g. as found in hysteretic behavior) of the target curve. For this purpose, Curve Mapping (Section 10.5.2, Section 28.2.4) is available.

10.5.2. Curve Mapping

In contrast to the Mean Square Error curve-matching metric described in Section 10.5.1, Curve Mapping incorporates the ordinate and the abscissa into the curve-matching metric Points of the one curve are mapped onto the second curve and the volume (area) between the two curves is computed. It is therefore highly suited to matching hysteretic curves. Both curves are normalized internally to adjust the magnitude of ordinate and abscissa, respectively. Since the curves could be of significantly different length, partial mapping is done.

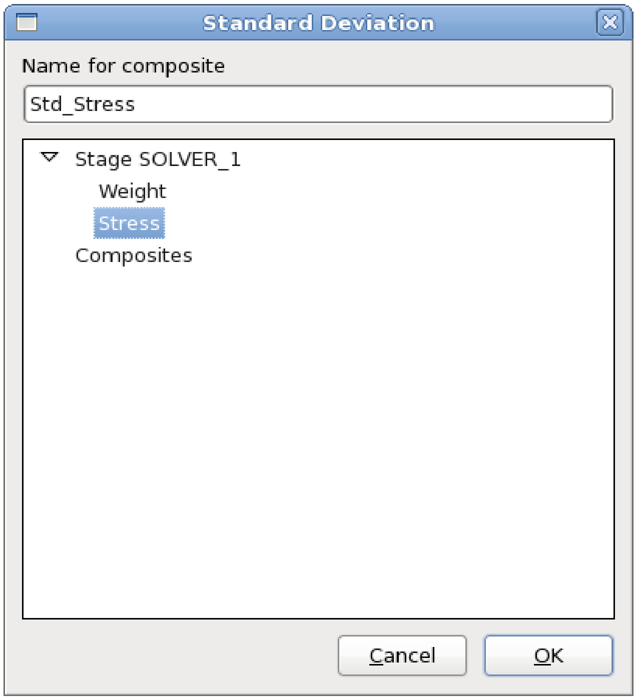
Please refer to Section 28.2.4 for the theory of Curve Mapping.

Note

It is recommended that both curves be filtered before matching to obtain curves which are as noise-free as possible. This avoids discrepancies in curve length which will affect the result. A general history filtering feature is available, Section 6.4.3. The user is also encouraged to study the Dynamic Time Warping measures (currently only available as responses).

10.6. Standard Deviation Composite

The standard deviation of another response or composite can be specified to be a composite, Figure 10-6. The dialog shows a list containing all previously defined responses and composites. The one to be used to calculate the standard deviation has to be selected.


Name for composite

Std_Stress

Stage SOLVER_1

Weight

Stress

Composites

Figure 10-6: Definition of a Standard Deviation composite

The variation of response approximated using response surfaces is computed analytically as documented for the LS-OPT stochastic contribution analysis, Section 29.7. For neural nets and composites a quadratic response surface approximation is created locally around the design, and this response surface is used to compute the robustness. Note that the recursion of composites (the standard deviation of a composite of a composite) may result in long computational times especially when combined with the use of neural networks. If the computational times are excessive, then the problem formulation must be changed to consider the standard deviations of response surfaces.

11. Classifier Dialog

Classifiers can be used to categorize design configurations into distinct classes, e.g. feasible vs infeasible. Thus, they can be used to define the constraints during optimization or reliability assessment. The definition of a classifier consists of two main parts – classifier components defining the criteria for categorization of the experimental design points, and a machine learning technique that can predict the expected category of any arbitrary design. The categorization criterion used to define a classifier can combine criteria based on responses, variables, dependents, composites as well as other classifiers. The components can belong to any stage with identical samplings.

11.1. Introduction

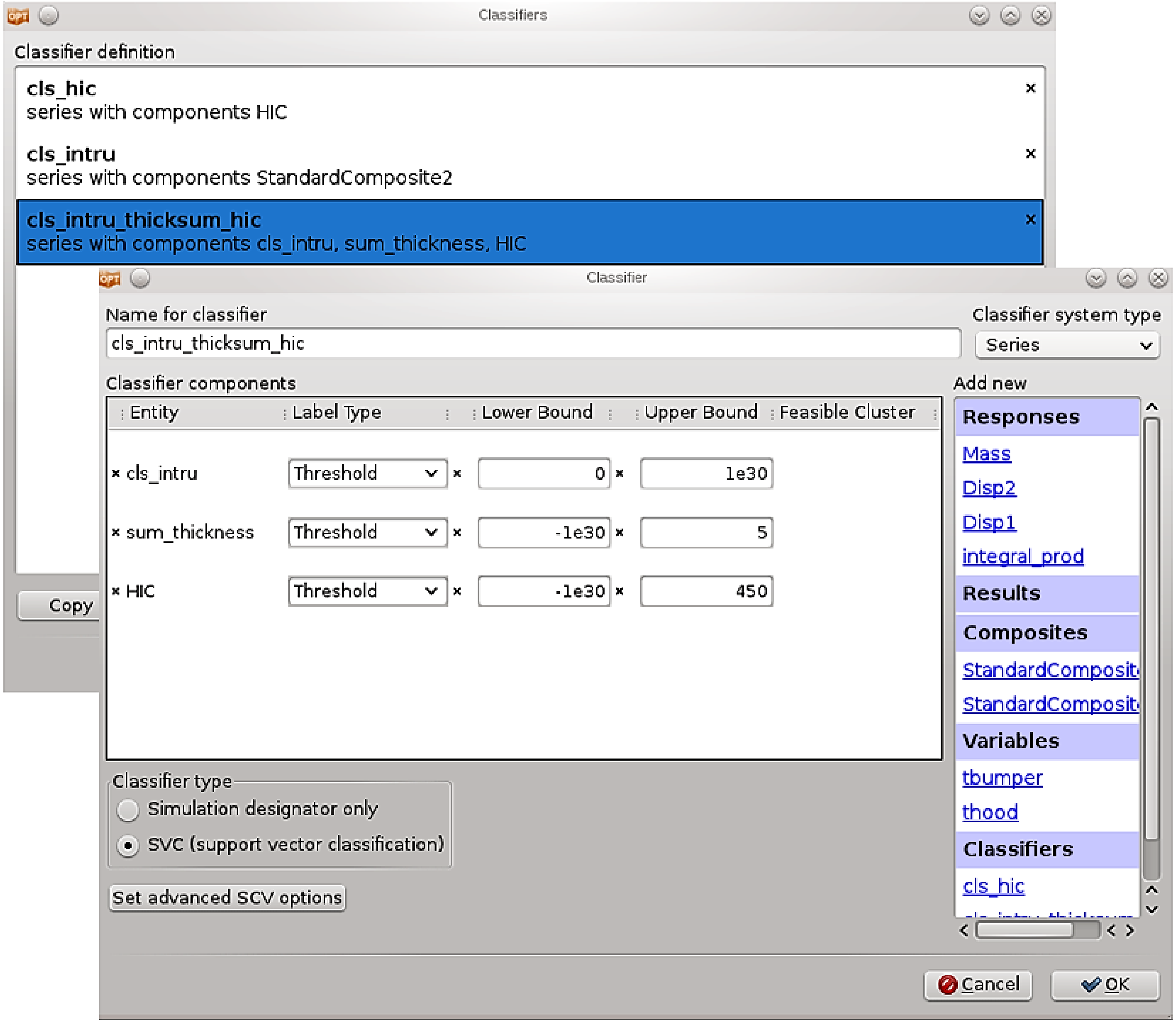
11.1.1. Classifier vs. metamodel

There is an important distinction between metamodels and classifiers. Metamodels predict the response values for a particular design, whereas a classifier predicts the class or category of the design. Thus, metamodels can be used to define objective functions or constraints, but classifiers are typically used to define constraints. During the training process, metamodels require the response values for a design, whereas a classifier only requires the category of the experimental design points. This gives the classification-based method a few advantages for certain problems and can also allow computation saving. Some of these advantages are listed in Remarks

.

11.2. Defining Classifiers

A classifier can be defined by using the interfaces in the **Classifiers** dialog, Figure 10-1. To add the first Classifier, select *Add Classifier* from the main GUI control bar *Add* (**+**) menu. To add a new definition, select the underlying entities from the list on the right and then specify their feasibility criteria. The Classifier system type specifies if the feasibility conditions on all the component entities need to be satisfied (series) or just one is enough (parallel). The Label type for each component specifies if its feasibility is based on a threshold value or on clustering. Apart from the definition of the system feasibility condition, the Classifier type is defined to specify the machine learning technique used to construct the decision boundary separating the distinct classes of designs. To edit an already defined classifier, double-click on the respective entry from the list on the left. Classifiers may be deleted using the *delete* icon on the right of the respective definition.



Classifiers

Classifier definition

cls_hic
series with components HIC

cls_intru
series with components StandardComposite2

cls_intru_thicksum_hic
series with components cls_intru, sum_thickness, HIC

Classifier

Name for classifier: cls_intru_thicksum_hic

Classifier system type: Series

Classifier components

Entity	Label Type	Lower Bound	Upper Bound	Feasible Cluster
× cls_intru	Threshold	0	1e30	
× sum_thickness	Threshold	-1e30	5	
× HIC	Threshold	-1e30	450	

Classifier type
○ Simulation designator only
● SVC (support vector classification)

Figure 11-1: Classifiers Dialog

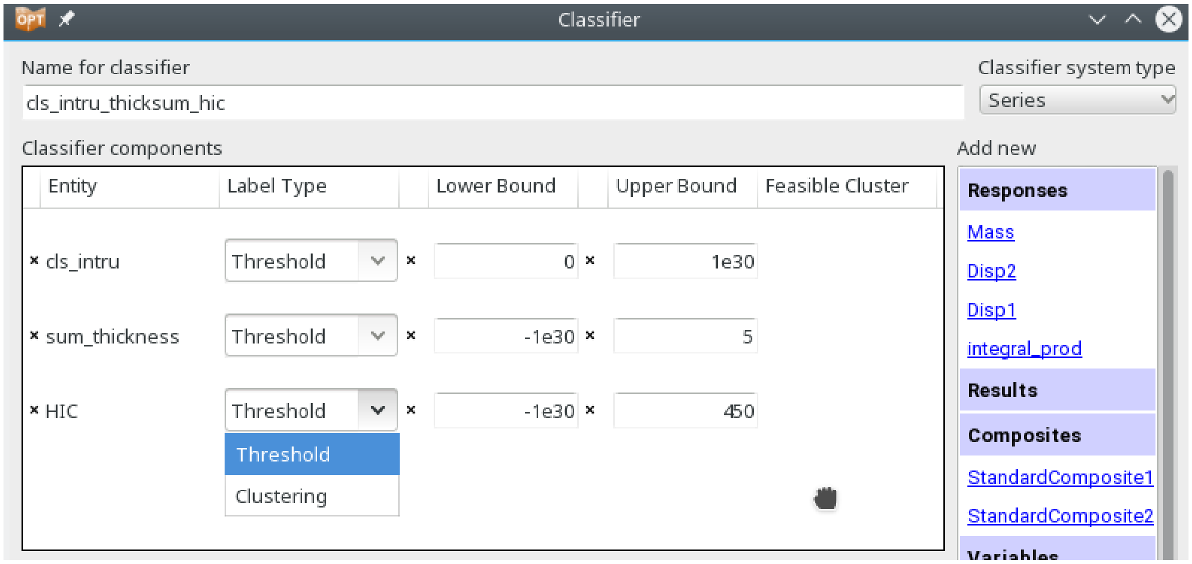
Table 11-1: Classifier options

Options	Description	Reference
Simulation designator only	Label the experimental design points into two classes (no class prediction at unknown points)	Section 11.4.1
SVC (Support vector classification)	Construct an optimal decision boundary using structural risk minimization that can be nonlinear	Section 11.4.2
Classifier system type (Series or Parallel)	Specify if the feasibility conditions need to be satisfied on all the component (series) or just on one of the component (parallel)	

Copy	Copy the selected Classifier
Paste	Paste a previously copied Classifier. The next free number is automatically appended to the name.

11.3. Class Label Assignment for Experimental Design Points

The classification algorithm in LS-OPT, i.e. SVM, is a semi-supervised machine learning algorithm. Thus, before SVM can be used to construct a decision boundary separating distinct design classes, the class labels need to be assigned for the training set (i.e. the experimental design points). This is done using a threshold value or using clustering in case the threshold is not known. Threshold limits can be applied on either the lower bound or the upper bound or both, just like in the case of constraints. The samples satisfying these bounds are labeled as +1 by LS-OPT and the ones violating either bound are labeled as -1. In the clustering-based approach, the user can specify whether the higher values are undesirable or the lower ones. LS-OPT performs a 2-class unsupervised clustering to assign the labels accordingly. If multiple classifier components are defined, both the feasibility of individual components and the system type (series or parallel) determine the class of a sample. It should be noted that mixed series-parallel systems can also be defined using nested classifier (i.e. when a classifier has another classifier as its component). Another point to note is that when clustering is used, the class label of a sample also depends on other samples. Therefore, the class labels of some samples may change between iterations.



Name for classifier: cls_intru_thicksum_hic

Classifier system type: Series

Classifier components

Entity	Label Type	Lower Bound	Upper Bound	Feasible Cluster
× cls_intru	Threshold	× 0	× 1e30	
× sum_thickness	Threshold	× -1e30	× 5	
× HIC	Threshold	× -1e30	× 450	

Figure 11-2: Class Label Definition

11.4. Classifier Type

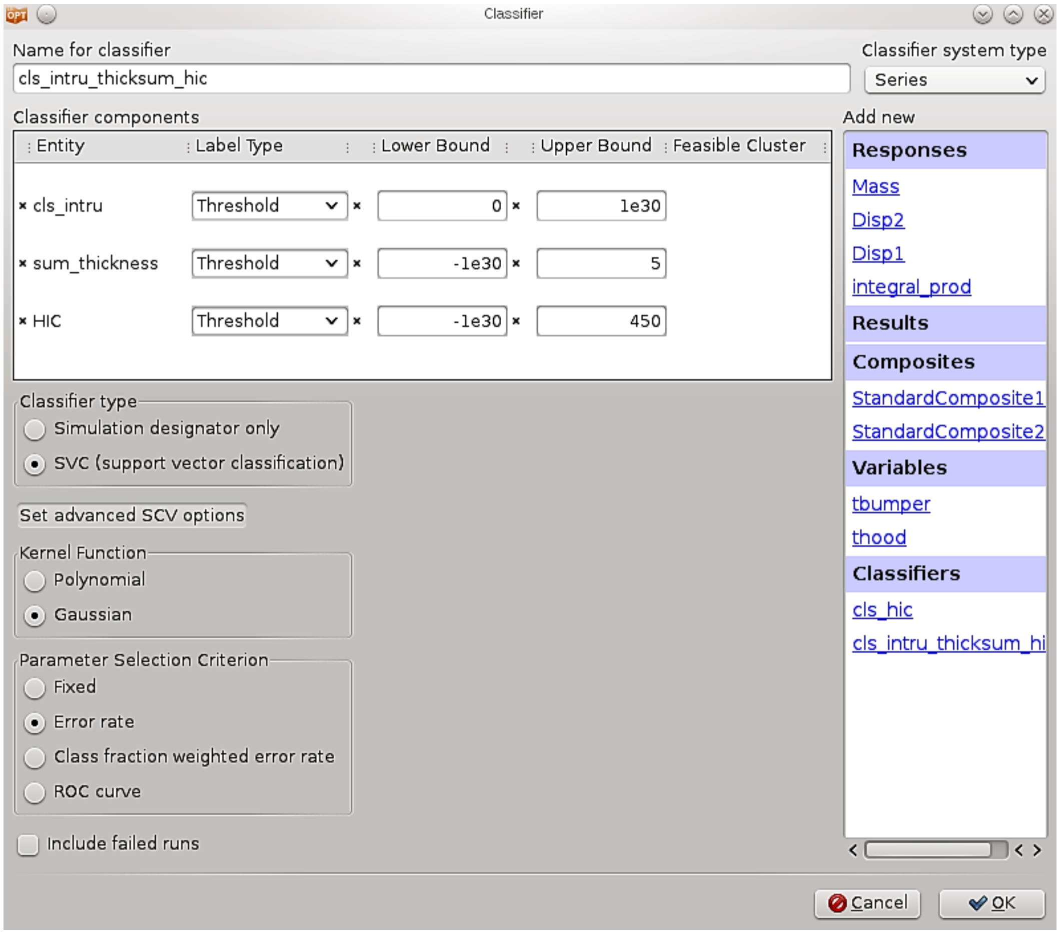
The type of classifier determines the algorithm used for class prediction at an arbitrary sample. SVC is the only supported method in LS-OPT. A classifier may also be defined as designator only, in which case there is no prediction.

11.4.1. Simulation Designator only

A “designator only” classifier cannot be used as a constraint as it does not predict the class. It only assigns class labels to the existing evaluated samples using the methods in section 11.3.

11.4.2. Support vector classification (SVC)

SVC is the default classification algorithm in LS-OPT. It belongs to a broader class of machine learning algorithms called support vector machines (SVM) that can be used for classification or regression. An SVC classifier, once trained using the evaluated samples, can predict the class of arbitrary designs. Therefore, it can easily be used as a constraint to predict feasibility. The attributes for an SVC classifier, available as advanced options, are displayed in Figure 11-3 and Table 11-2. A kernel is specified, similar to basis functions for metamodeling. The SVC decision boundary also depends on additional parameters C and ε that may be specified by the user (fixed) or internally optimized by LS-OPT based on a specified criterion (Section 25.2.2). There is also an option to include failed points, in which case such samples are included in the training set as part of the infeasible dataset for the classifier construction. The failed simulation points are assigned a separate -2 label in the database to distinguish them from the design criteria based infeasible samples. However, if the failed points are included then they are grouped together with the other infeasible designs (-1 label) while constructing the classifier. Please refer to Section 25.2 for the theory of SVC.



Name for classifier: cls_intru_thicksum_hic

Classifier system type: Series

Classifier components

Entity	Label Type	Lower Bound	Upper Bound	Feasible Cluster
× cls_intru	Threshold ×	0 ×	1e30	
× sum_thickness	Threshold ×	-1e30 ×	5	
× HIC	Threshold ×	-1e30 ×	450	

Add new
- Responses
  - Mass
  - Disp2
  - Disp1
  - integral_prod
- Results
- Composites
  - StandardComposite1
  - StandardComposite2
- Variables
  - tbumper
  - thood
- Classifiers
  - cls_hic
  - cls_intru_thicksum_hi

Classifier type
- ○ Simulation designator only
- ● SVC (support vector classification)

Set advanced SCV options

Kernel Function
- ○ Polynomial
- ● Gaussian

Parameter Selection Criterion
- ○ Fixed
- ● Error rate
- ○ Class fraction weighted error rate
- ○ ROC curve

□ Include failed runs

Figure 11-3: SVC attributes

Table 11-2: SVC advanced options

Options	Description	Reference
Kernel Function (Polynomial or Gaussian)	Define the basis functions type.	Section 25.2
Parameter Selection Criterion	Define the tolerance to quantify the strictness of the constraint satisfaction.	Section 25.2.2
Include failed runs	Include failed points and treat them as infeasible.	

Remarks

1. As classifiers only need classification information (vs response values for metamodeling) at the experimental design points, they can easily be applied to binary problems or to problems with discontinuous responses without additional complexity.
2. Multiple responses, including those coming from different stages, can be represented by a single classifier. As a result, all the stages need not always be calculated. For instance, if a relatively cheap NVH analysis fails for a vehicle design configuration then one doesn't have to run the more expensive crash simulations.
3. LS-OPT currently supports two-class or binary classification:

   Support vector classification (SVC) is supported currently. A classifier can also be specified to be an experimental design class designator only, but it cannot be used as an optimization or reliability constraint then.
4. The SVM decision between the two distinct classes is demarcated by the zero level. Therefore, when an SVM is used as a constraint defining the boundary between feasible and infeasible designs, one of the constraint bounds is set as zero (typically the lower bound, as the feasible class is +1).

12. Optimization Dialog – Objectives, Constraints and Algorithms

This chapter describes the specification of objectives and constraints for the design formulation and the optimization algorithms used for metamodel optimization.

12.1. Formulation of the optimization problem

Multi-criteria optimal design problems can be formulated. These typically consist of the following:

1. Multiple objectives (multi-objective formulation)
2. Multiple constraints.

Mathematically, the problem is defined as follows:

$$\text{Minimize} \qquad F\left(\Phi_1, \Phi_2, \ldots, \Phi_N\right)$$

subject to

$$\begin{aligned} &L_1 \le g_1 \le U_1 \\ &L_2 \le g_2 \le U_2 \\ &\vdots \\ &L_m \le g_m \le U_m \end{aligned}$$

where F represents the multi-objective function, $\Phi_i = \Phi_i\left(x_1, x_2, \ldots, x_n\right)$ represent the various objective functions and $g_j = g_j\left(x_1, x_2, \ldots, x_n\right)$ represent the constraint functions. The symbols x_i represents n design variables.

In order to generate a trade-off design curve involving objective functions, more than one objective Φ_i must be specified so that the multi-objective

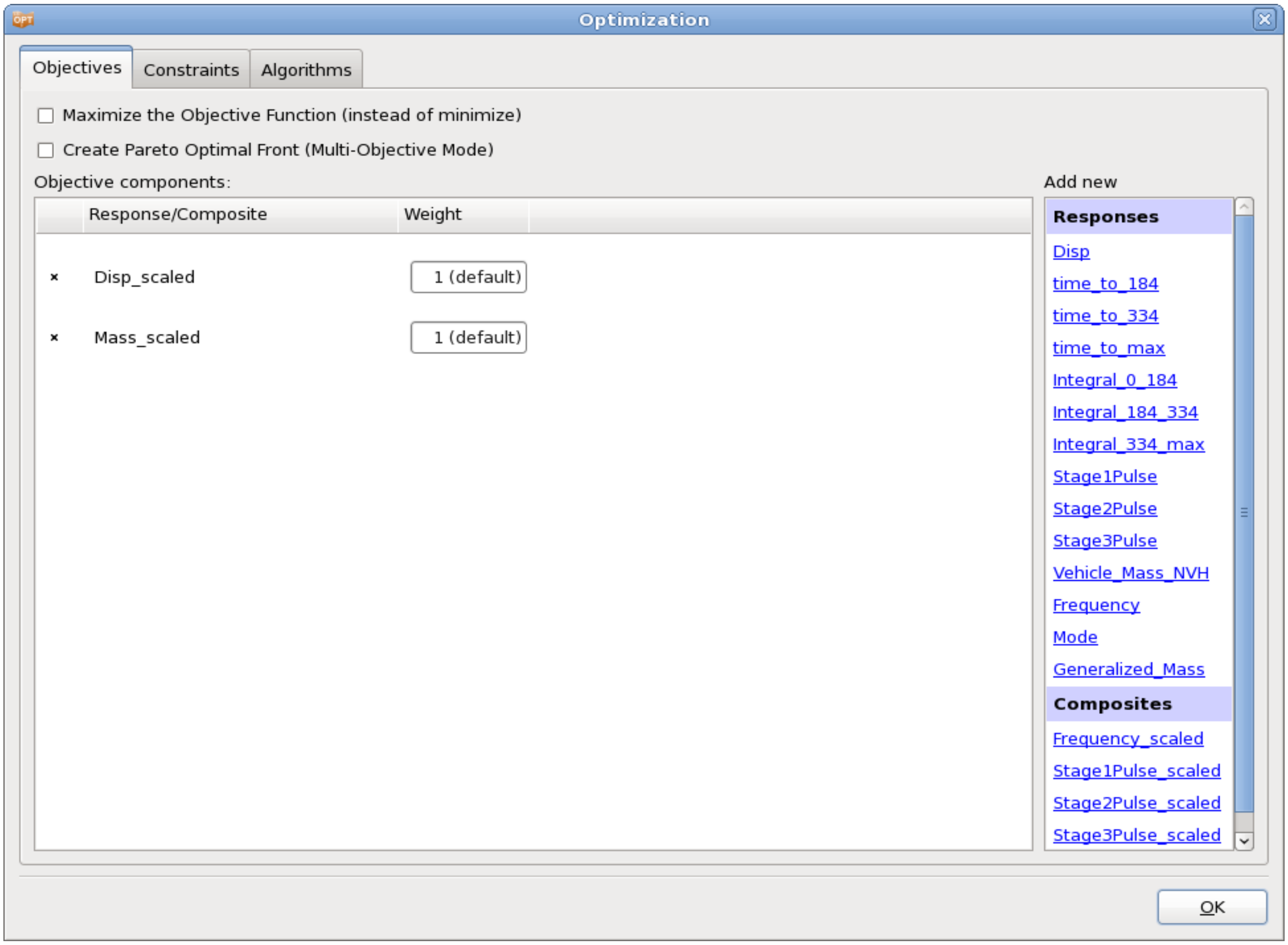
$$F = \sum_{k=1}^{N} \omega_k \Phi_k \,. \tag{12-1}$$

A component function must be assigned to each objective function where the component function can be defined as a *composite function F* (see Chapter 10) or a *response function f* (see Chapter 6).

12.2. Defining objective functions

Objectives are defined in the **Objectives** tab of the **Optimization** dialog, Figure 12-1. To define an objective, select a response or composite from the list on the right that contains all previously defined responses and composites. The entity will show up in the list on the left. For each objective, a weight has to be specified using the *Weight* text field. If multiple objectives are defined, LS-OPT uses the weights to build a multi-objective function as described in Section 12.1. The weight applies to each objective as represented by ω_k in Equation (12-1). Note that the optimization result depends in the specified weights.

The weights are not used in Multi-Objective Optimization, except to record the scalar multi-objective value. Additional options are described in Table 12-1.



☐ Maximize the Objective Function (instead of minimize)

☐ Create Pareto Optimal Front (Multi-Objective Mode)

Objective components:

	Response/Composite	Weight
×	Disp_scaled	1 (default)
×	Mass_scaled	1 (default)

Add new

Responses

Disp
time_to_184
time_to_334
time_to_max
Integral_0_184
Integral_184_334
Integral_334_max
Stage1Pulse
Stage2Pulse
Stage3Pulse
Vehicle_Mass_NVH
Frequency
Mode
Generalized_Mass

Composites

Frequency_scaled
Stage1Pulse_scaled
Stage2Pulse_scaled
Stage3Pulse_scaled

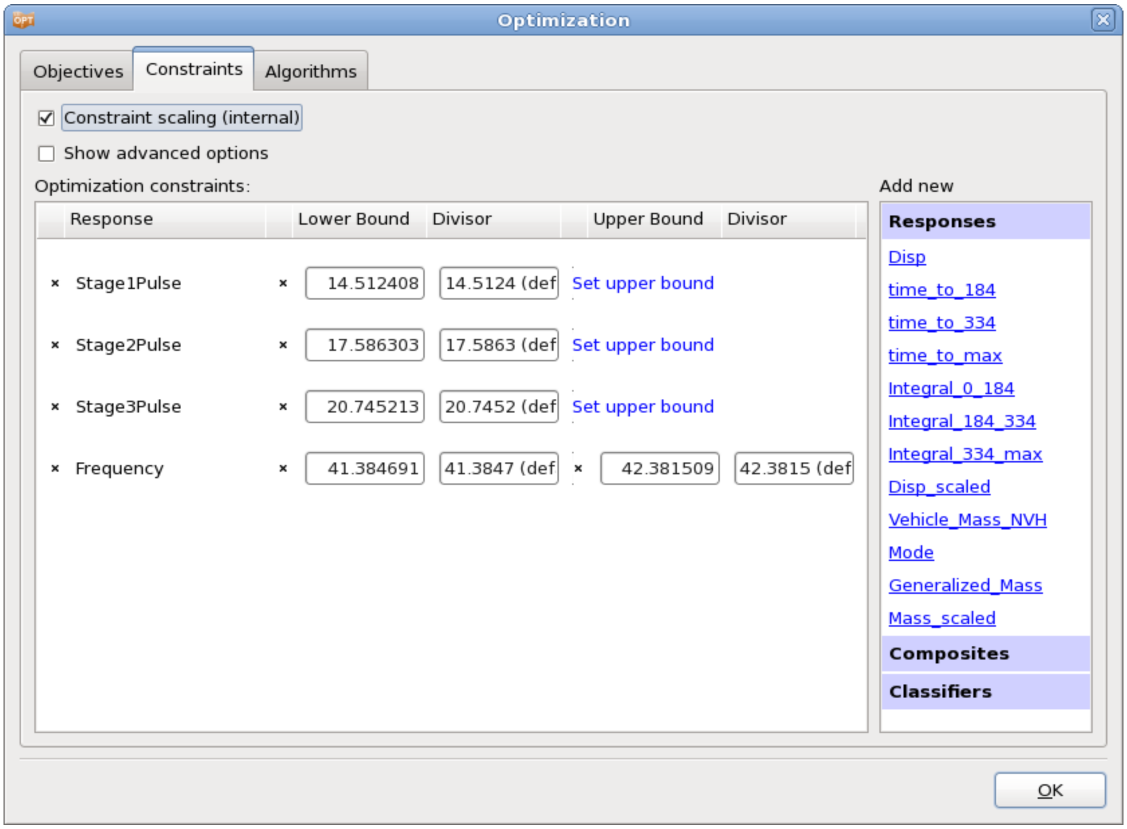
Figure 12-1: Objective panel in LS-OPTui Optimization dialog.

Table 12-1: Objective options

Option	Description
Maximize Objective Function (instead of minimize)	The default is to minimize the objective functions. The program can however be set to maximize the objective functions.
Create Pareto Optimal Front (Multi-Objective Mode)	Pareto optimal solutions are calculated instead of a single optimum. This option is only available if multiple objectives are defined, Section 4.10.

12.3. Defining a constraint

Constraints are defined in the **Constraints** tab of the **Optimization** dialog, Figure 12-2. To define a constraint, select a response or composite from the list on the right that contains all previously defined responses, composites and classifiers. The selected entity will show up in the list on the left. To specify a lower or an upper bound, select the respective hyperlink and enter the desired value in the text field. In the case of classifier-based constraints usually one of the bounds, typically the lower bound, is set as zero to define feasibility.



☑ Constraint scaling (internal)

□ Show advanced options

Optimization constraints:

	Response		Lower Bound	Divisor		Upper Bound	Divisor
×	Stage1Pulse	×	14.512408	14.5124 (def		Set upper bound	
×	Stage2Pulse	×	17.586303	17.5863 (def		Set upper bound	
×	Stage3Pulse	×	20.745213	20.7452 (def		Set upper bound	
×	Frequency	×	41.384691	41.3847 (def	×	42.381509	42.3815 (def

Add new

Responses
- Disp
- time_to_184
- time_to_334
- time_to_max
- Integral_0_184
- Integral_184_334
- Integral_334_max
- Disp_scaled
- Vehicle_Mass_NVH
- Mode
- Generalized_Mass
- Mass_scaled

Composites

Classifiers

Figure 12-2: Constraints panel in LS-OPTui

Additionally, for Reliability Based Design Optimization, the probability of exceeding a bound on a constraint can be set, Section 14.7.

Internal constraint scaling can be defined by selecting the *Constraint scaling* option and defining the respective scaling factors in the Divisor text field, Section 12.3.1.

To delete a constraint definition or a bound, use the respective *delete* icon.

If *Show advanced options* is selected, the *Strict* option is available. For details, see Section 12.3.2.

12.3.1. Internal scaling of constraints

Constraints can be scaled internally to ensure normalized constraint violations. This may be important when having several constraints and an infeasible solution so that when the maximum violation over the defined constraints is minimized, the comparison is independent of the choice of measuring units of the constraints. The scale factor s_j (to be specified in the respective *Divisor* test field) is applied internally to constraint j as follows:

$$\frac{-g_j(x)+L_j}{s_j^L} \le 0; \; \frac{g_j(x)-U_j}{s_j^U} \le 0 \; .$$

A logical choice for the selection of s is $s_j^L = L_j$ and $s_j^U = U_j$, so that the above inequalities become

$$\frac{-g_j(x)}{L_j} + 1 \le 0; \; \frac{g_j(x)}{U_j} - 1 \le 0 \; .$$

internally and in the infeasible phase:

$$\frac{-g_j(x)}{L_j} + 1 \le e; \; \frac{g_j(x)}{U_j} - 1 \le e; e \ge 0.$$

12.3.2. Minimizing the maximum response or violation*

If *Show advanced options* is selected, additional columns appear in the constraints list to select strict constraints. Refer to Section 27.1 for the theory regarding strict and slack constraints. To specify hard (strict) constraints, select the respective *Strict* checkboxes. Otherwise constraints are soft (slack) constraints.

The main purpose of a formulation using strict and slack constraints is to compromise only on the slack constraints if a feasible design cannot be found.

Remarks

1. The objective function is ignored if the problem is infeasible.
2. The variable bounds of both the region of interest and the design space are always hard.
3. Soft constraints will be strictly satisfied if a feasible design is possible.
4. If a feasible design is not possible, the most feasible design will be computed.

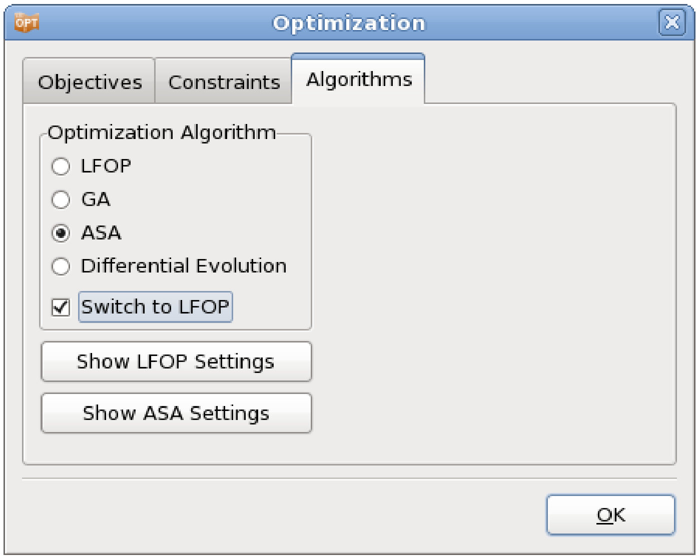
5. If feasibility must be compromised (there is no feasible design), the solver will automatically use the slackness of the soft constraints to try and achieve feasibility of the hard constraints. However, there is always a possibility that hard constraints must still be violated (even when allowing soft constraints). In this case, the variable bounds may be violated, which is highly undesirable as the solution will lie beyond the region of interest and perhaps beyond the design space. This could cause extrapolation of the response surface or worse, a future attempt to analyze a design which is not analyzable, e.g. a sizing variable might have become zero or negative.
6. Soft and strict constraints can also be specified for search methods. If there are feasible designs with respect to hard constraints, but none with respect to all the constraints, including soft constraints, the most feasible design will be selected. If there are no feasible designs with respect to hard constraints, the problem is 'hard-infeasible' and the optimization terminates with an error message.

12.4. Algorithms

Optimization algorithms for metamodel-based optimization can be selected in the Algorithms tab of the Optimization dialog, Figure 12-3.

The core solvers that can be used for metamodel optimization are LFOP, the Genetic Algorithm (GA), Adaptive Simulated Annealing (ASA) and Differential Evolution. Hybrid algorithms may also be selected by selecting *Switch to LFOP*, namely the Hybrid GA and Hybrid ASA. The hybrid algorithms start with the GA and ASA to find an approximate global optimum after which LFOP is used to sharpen the solution. The solution to a hybrid algorithm will be at least as good as the one provided by the global optimizer (GA and ASA). Hybrid Simulated Annealing is the default.

For each algorithm, advanced settings are available using the respective *Show *** Settings* button.



Optimization Algorithm

○ LFOP
○ GA
● ASA
○ Differential Evolution

☑ Switch to LFOP

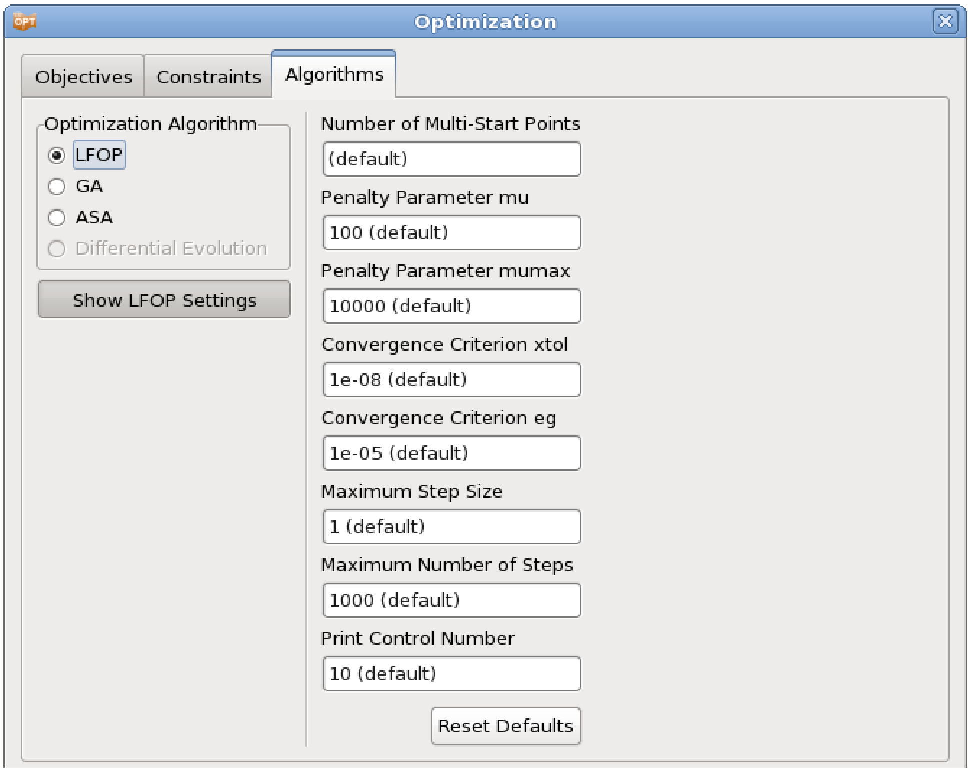
Figure 12-3: Selecting the optimization algorithm used for the optimization on the metamodel

Table 12-2: Algorithms options

Option	Description	Reference
LFOP	Leapfrog Optimizer	Section 12.4.1, Section 26.8
GA	Genetic Algorithm	Section 12.4.2, Section 26.9
ASA	Adaptive Simulated Annealing	Section 12.4.3, Section 26.11
Differential Evolution	Differential Evolution. Only available if discrete and string variables as well as constraints are absent.	Section 26.12
Switch to LFOP	Hybrid version	Section 26.13

12.4.1. Setting parameters in the LFOPC algorithm*

The values of the responses are scaled with the values at the initial design. The default parameters in LFOPC should therefore be adequate. Should the user have more stringent requirements, the following parameters may be set for LFOPC. These can be set in the GUI if *Show LFOP Settings* is selected. See Section 26.7 for the theory of LFOPC.



Optimization Algorithm
- ● LFOP
- ○ GA
- ○ ASA
- ○ Differential Evolution

Number of Multi-Start Points: (default)
Penalty Parameter mu: 100 (default)
Penalty Parameter mumax: 10000 (default)
Convergence Criterion xtol: 1e-08 (default)
Convergence Criterion eg: 1e-05 (default)
Maximum Step Size: 1 (default)
Maximum Number of Steps: 1000 (default)
Print Control Number: 10 (default)

Figure 12-4: LFOP settings

Table 12-3: LFOPC parameters and default values

Option	**Parameter**	**Remark**
Number of Multi-Start Points	Number of Multi-Start Points	
Penalty Parameter mu	Initial penalty value μ	
Penalty Parameter mumax	Maximum penalty value $\mu_{\max}$	1
Convergence Criterion xtol	Convergence tolerance $\varepsilon_x$on the step movement	2
Convergence Criterions eg	Convergence tolerance ε_f on the norm of the gradient	2
Maximum Step Size	Maximum step size δ	3
Maximum Number of Steps	Maximum number of steps per phase	1
Print Control Number	Printing interval	4

Remarks

1. For higher accuracy, at the expense of economy, the value of $\mu_{\max}$ can be increased. Since the optimization is done on approximate functions, economy is usually not important. The maximum number of steps must then be increased as well.
2. The optimization is terminated when either of the convergence criteria becomes active that is when

$$\|\Delta(x)\| < \varepsilon_x$$

or

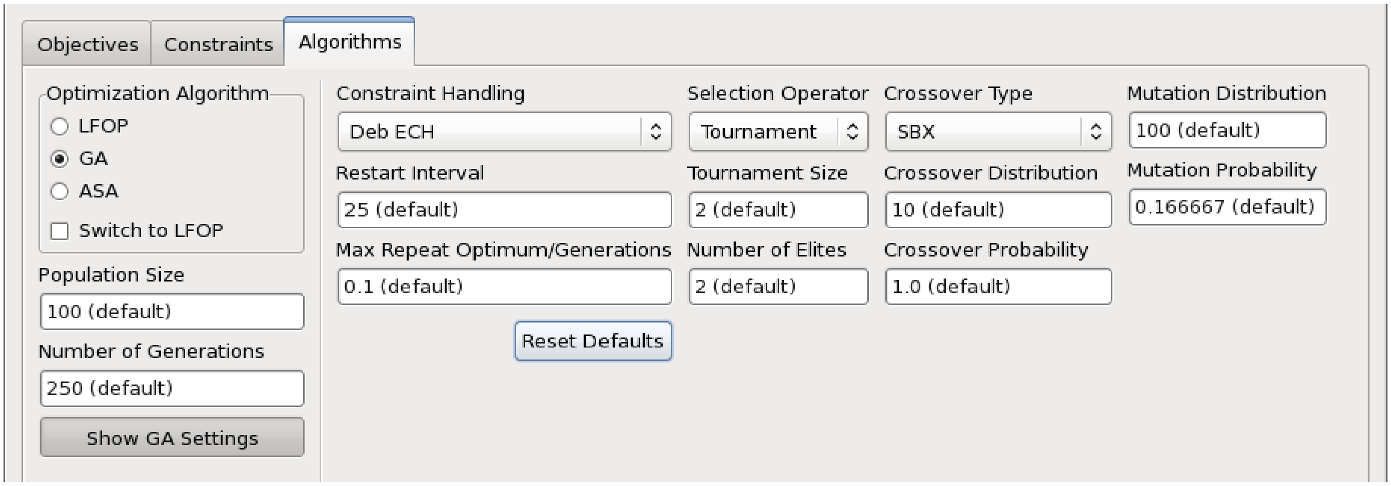
$$\|\nabla f(x)\| < \varepsilon_f$$

3. It is recommended that the maximum step size δ be of the same order of magnitude as the “diameter of the region of interest”. To enable a small step size for the sequential approximation scheme, the maximum step size has been defaulted to $= 0.05\sqrt{\sum_{i=1}^{n}(range)}$.
4. If the Print Control Number = Maximum number of steps + 1, then the printing is done on step 0 and exit only. The values of the design variables are suppressed on intermediate steps if the Print Control Number < 0.

In the case of an infeasible optimization problem, the solver will find the most feasible design within the given region of interest bounded by the simple upper and lower bounds. If LFOP is selected as a non-hybrid optimizer, a global solution is attempted by multiple starts from a set of random points.

12.4.2. Setting parameters in the genetic algorithm*

The default parameters in the GA should be adequate for most problems. However, if the user needs to explore different methods, the following parameters may be set in the GUI (see Figure 12-5). See Section 26.9 for the theory of the Genetic Algorithm.



Optimization Algorithm
○ LFOP
● GA
○ ASA
□ Switch to LFOP

Population Size: 100 (default)
Number of Generations: 250 (default)
Constraint Handling: Deb ECH
Restart Interval: 25 (default)
Max Repeat Optimum/Generations: 0.1 (default)
Selection Operator: Tournament
Tournament Size: 2 (default)
Number of Elites: 2 (default)
Crossover Type: SBX
Crossover Distribution: 10 (default)
Crossover Probability: 1.0 (default)
Mutation Distribution: 100 (default)
Mutation Probability: 0.166667 (default)

Figure 12-5: GA settings

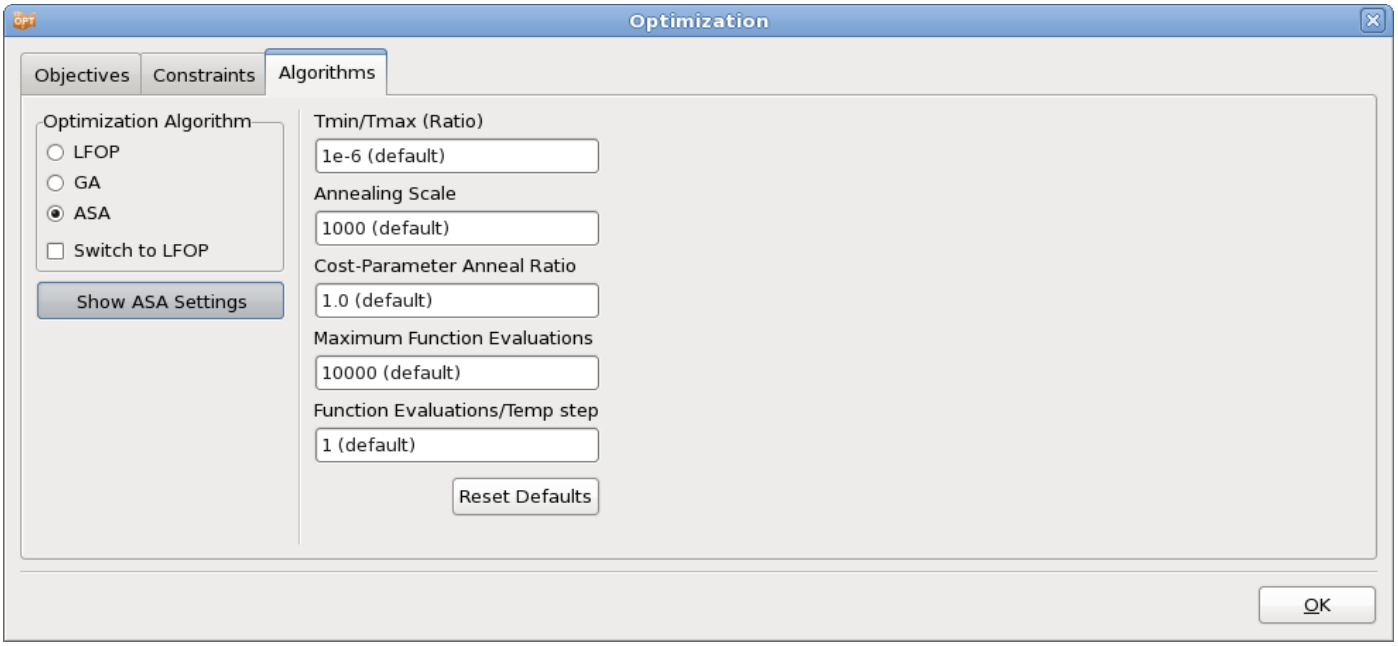
Table 12-4: GA parameters and default values

Option	Parameter	Remark
Population Size	Population size (always even)	
Number of Generations	Number of generations	
Selection Operator	Selection operator: Tournament, Roulette, SUS	
Tournament Size	Tournament size for tournament selection operator	
Elitism	Switch elitism for single objective GA: ON/OFF	
Number of Elites	Number of elites passed to next generation	
Encoding variable	Type of encoding for a variable: Binary=1, Real=2	
Numbits variable	Number of bits assigned to a binary variable	
Crossover type	Type of real crossover: SBX, BLX	
Crossover probability	Real crossover probability	

Alpha value for BLX	Value of α for BLX operator
Crossover distribution	Distribution index for SBX crossover operator
Mutation probability	Mutation probability in real-space
Mutation distribution	Distribution index for mutation operator
Algorithm Subtype	Multi-objective optimization algorithm: NSGA2, SPEA2
Restart Interval	Frequency of writing restart file. For multi-objective problems, this parameter governs the frequency of writing TradeOff files
Max Repeat Optimum/Generations	Maximum number of generations allowed to repeat as a fraction of the total number of generations allowed.
Constraint Handling	Constraint handling types: Deb Efficient Constraint Handling, Penalty

12.4.3. Setting parameters in the simulated annealing algorithm*

The adaptive simulated annealing parameters can be modified in the GUI, Figure 12-6. See Section 26.11 for the theory of Adaptive Simulated Annealing.



Optimization Algorithm

○ LFOP
○ GA
● ASA
□ Switch to LFOP

Tmin/Tmax (Ratio): 1e-6 (default)

Annealing Scale: 1000 (default)

Cost-Parameter Anneal Ratio: 1.0 (default)

Maximum Function Evaluations: 10000 (default)

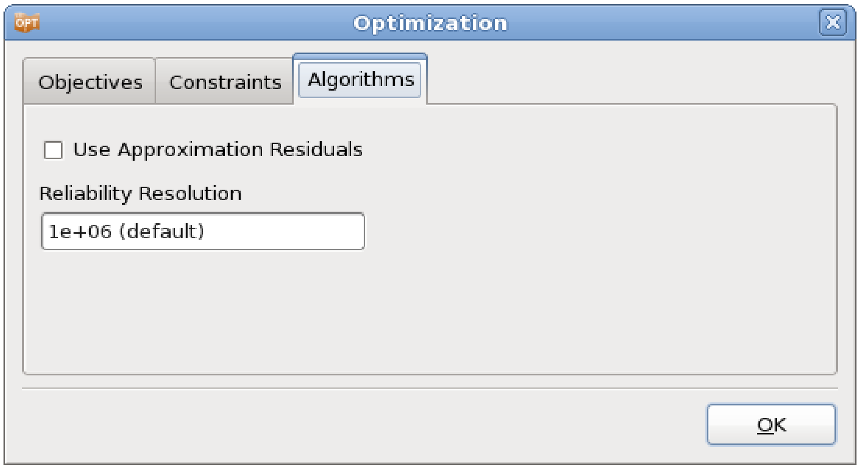
Function Evaluations/Temp step: 1 (default)

Figure 12-6: ASA settings

Table 12-5: ASA parameters and default values

Option	Parameter
Tmin/Tmax (Ratio)	Ratio of minimum and maximum temperature
Annealing Scale	Annealing scale
Cost-Parameter Anneal Ratio	Ratio of cost temperature ratio and parameter temperature ratio
Maximum Function Evaluations	Maximum number of function evaluations
Function Evaluations/Temp step	Number of function evaluations at some temperature

12.5. Algorithms for metamodel based Monte Carlo analysis



□ Use Approximation Residuals

Reliability Resolution

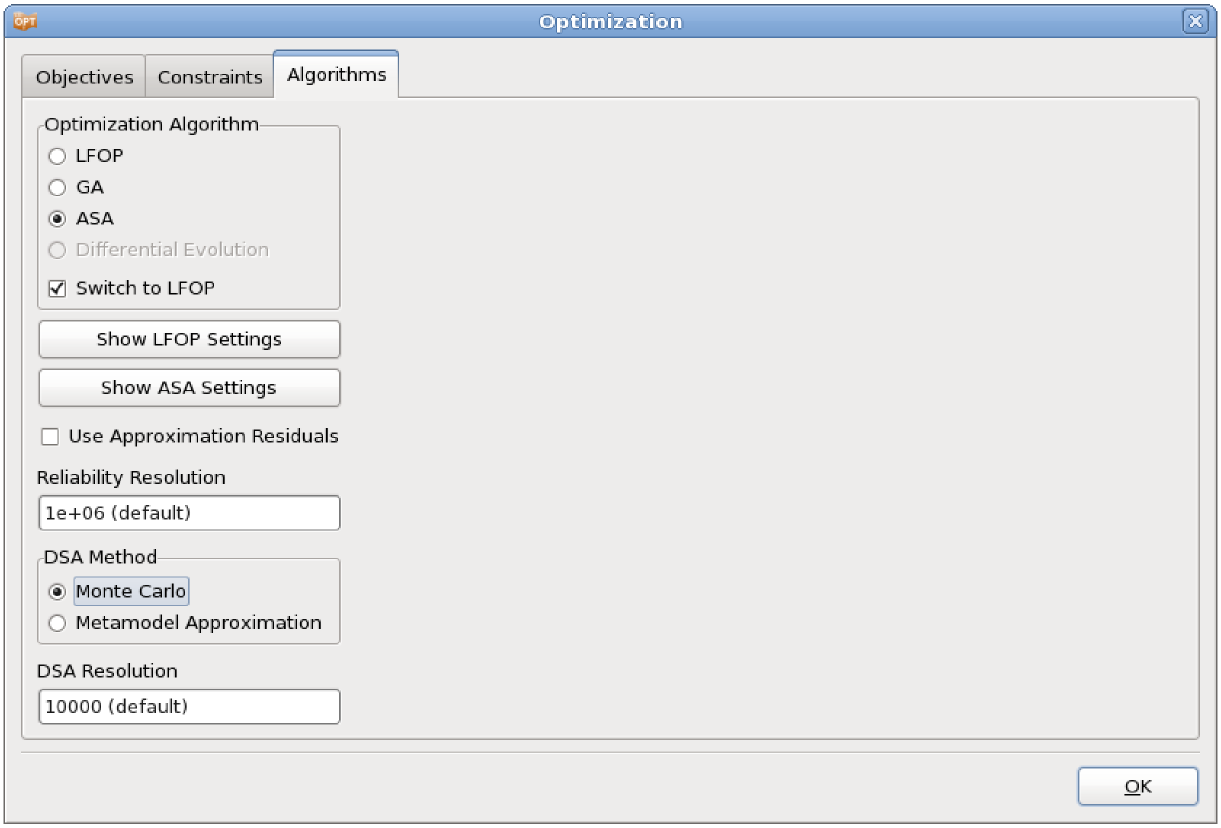
1e+06 (default)

Figure 12-7: Algorithm Options for Metamodel based Monte Carlo Analysis

Table 12-6: Algorithm Options for Metamodel based Monte Carlo Analysis

Option	Description
Use Approximation Residuals	If noise was found when the metamodel was created, then this noise may be reproduced whenever the metamodel is used for reliability computations. This is possible only for the response surfaces and neural nets. The noise is normally distributed with a zero mean and a standard deviation computed from the residuals of the least square fit.
Reliability Resolution	The number of Monte Carlo samples to be analyzed can be set by the user. These samples are evaluated based on the metamodels and not using the actual solver.

12.6. Algorithms for Reliability based Design Optimization (RBDO)



Optimization Algorithm

○ LFOP
○ GA
● ASA
○ Differential Evolution

☑ Switch to LFOP

□ Use Approximation Residuals

Reliability Resolution

1e+06 (default)

DSA Method

● Monte Carlo
○ Metamodel Approximation

DSA Resolution

10000 (default)

Figure 12-8: Algorithms options for RBDO

RBDO using a First Order Second Moment (FOSM), the current method used in LS-OPT, involves the computation of the stochastic contribution of the variables on the responses (Section 29.4.5). The stochastic contribution is closely related to the sensitivity of the design responses, which can be calculated in two ways based on the Design Sensitivity Analysis (DSA) options set by the user (Table 12-7). Higher design sensitivities and higher variable uncertainty, both lead to higher stochastic contribution.

Table 12-7: Algorithms options for RBDO

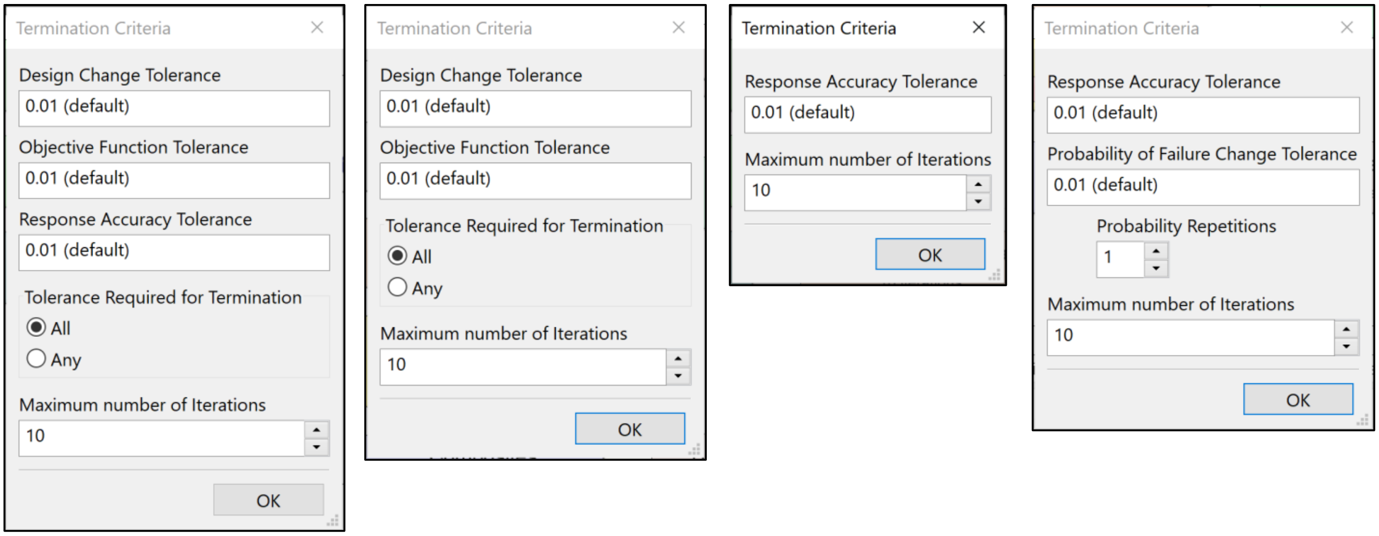
Option	Description	Reference
DSA Method	Monte Carlo: Stochastic contribution is calculated using MC sampling Metamodel Approximation: Stochastic contribution is calculated using quadratic approximations	Section 14.8
DSA Resolution	Number of sample values if DSA Method is Monte Carlo	

13. Termination Criteria

This chapter explains termination criteria for iterative tasks. The available criteria differ based on the optimization task, strategy and the number of objectives.

13.1. Metamodel based methods

Depending on the optimization task and strategy, the user can specify tolerances on the design change (Δx_i), the objective function change (Δf) or the accuracy of the metamodel. The user can also specify whether termination is reached if any one (*or* condition), or all (*and* condition) of these criteria are met, Figure 13-1. The options are described in Table 13-1. Response accuracy and the maximum number of iterations are the only available termination criteria for multi-objective optimization.



Termination Criteria

Design Change Tolerance: 0.01 (default)

Objective Function Tolerance: 0.01 (default)

Response Accuracy Tolerance: 0.01 (default)

Tolerance Required for Termination

● All

○ Any

Maximum number of Iterations: 10

Termination Criteria

Design Change Tolerance: 0.01 (default)

Objective Function Tolerance: 0.01 (default)

Tolerance Required for Termination

● All

○ Any

Maximum number of Iterations: 10

Termination Criteria

Response Accuracy Tolerance: 0.01 (default)

Maximum number of Iterations: 10

Termination Criteria

Response Accuracy Tolerance: 0.01 (default)

Probability of Failure Change Tolerance: 0.01 (default)

Probability Repetitions: 1

Maximum number of Iterations: 10

Figure 13-1: Termination Criteria dialog for metamodel based optimization or Monte Carlo analysis using different methods – single objective sequential (left), single objective sequential with domain reduction or efficient global optimization (middle left), multi-objective sequential (middle right), sequential Monte Carlo analysis (right).

Table 13-1: Termination Criteria options for metamodel based optimization

Option	Parameter	Reference
Tolerance Required for Termination	All: Design AND Objective AND Metamodel Accuracy Any: Design OR Objective OR Metamodel Accuracy	-
Design Change Tolerance	Tolerance on design accuracy ε_x	Section 13.1.1
Objective Function Tolerance	Tolerance on objective function accuracy ε_f	Section 13.1.1
Response Accuracy Tolerance	Tolerance on accuracy of response surface ε_r	Section 13.1.2
Probability of Failure	Tolerance on probability of failure	Section 13.1.3
Repeat	Only stop if probability of failure tolerance is met *Repeat* times	
Maximum number of Iterations	Maximum Number of Iterations	Section 13.1.4

13.1.1. Design Change Tolerance and Objective Function Tolerance

The design change termination criterion and the objective function termination criterion are available for the strategy sequential with domain reduction and the sequential strategy, if no Pareto optimal solutions are calculated.

The design change termination criterion becomes active if

$$\frac{\left\|x^{(k)} - x^{(k-1)}\right\|}{\|d\|} < \varepsilon_x ,$$

where ***x*** refers to the vector of design variables and ***d*** is the size of the design space.

The objective function termination criterion becomes active if

$$\left|\frac{f^{(k)} - f^{(k-1)}}{f^{(k-1)}}\right| < \varepsilon_f ,$$

where *f* denotes the value of the objective function, (*k*) and (*k* – 1) refer to two successive iteration numbers.

The use of these termination criteria is recommended for a metamodel based optimization with strategy sequential with domain reduction.

13.1.2. Response Accuracy Tolerance

The response accuracy tolerance criterion is available for sequential strategies. The tolerance on the metamodel accuracy is based on the *change* of the prediction accuracy measure (square root of the PRESS error, Section 23.3.5). The measure is divided by the mean of the simulated values used to construct the response surface unless this mean is zero. The value of the most critical response is used.

The response accuracy tolerance termination criteron becomes active if

$$\left|s_i^{(k)} - s_i^{(k-1)}\right| < \varepsilon_r ,$$

where s_i denotes the approximation error of i^{th} response characterized by the ratio of square root PRESS statistics and the mean value of response and, (k) and (k – 1) refer to two successive iteration numbers.

The use of this termination criterion is recommended for the sequential strategy, if the iterative process is used to improve the quality of the metamodel. Make sure to use the OR option and set the other tolerances to 0.

13.1.3. Probability of Failure

The probability of failure tolerance criterion is available for the sequential strategy of the metamodel-based Monte Carlo task. The tolerance on the failure probability is based on the change of the total probability of failure of all constraints. As the failure probability is between 0 and 1, scaling is not necessary and the tolerance is based on the absolute change in the probability value.

The failure probability tolerance termination criterion becomes active if

$$\left|P_f^{(k)} - P_f^{(k-1)}\right| < \varepsilon_p,$$

where $P_f^{(k)}$ denotes the total probability of failure for the k^{th} iteration and is the ε_p tolerance limit. If a stricter convergence criterion is desired then it is possible to either select a smaller value of ε_p or to require the above criterion to be met for multiple consecutive iterations. Thus, the tolerance criterion is

$$\left|P_f^{(k)} - P_f^{(k-1)}\right| < \varepsilon_p \qquad \forall k > k - n_r,$$

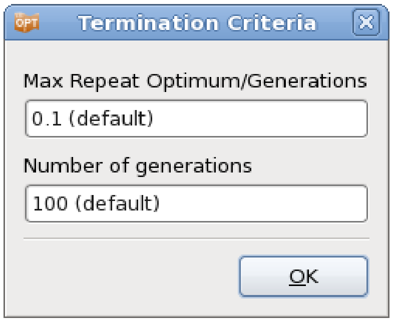
where n_r is the number of consecutive iterations for which the failure probability tolerance must be met for stopping.

13.1.4. Maximum Number of Iterations

The maximum number of optimization iterations is specified in the appropriate field in the **Termination Criteria** dialog. If previous results exist, LS-OPT will recognize this (through the presence of results files in the Run directories) and not rerun these simulations. If the termination criteria described above are reached first, LS-OPT will terminate and not perform the maximum number of iterations.

13.2. Direct Optimization

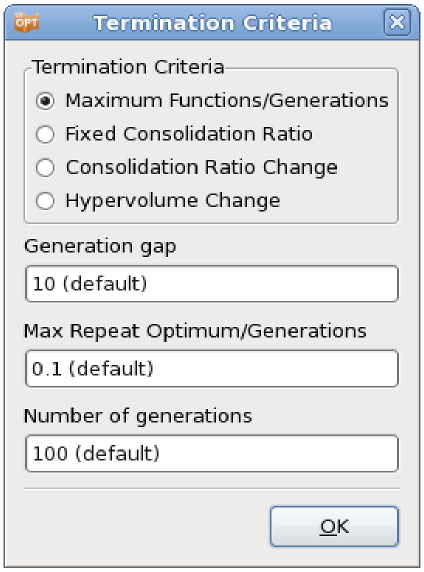
The termination criteria options for direct optimization are different compared to metamodel-based optimization as these criteria are defined for the core optimizer. The termination criteria are also different for single-objective (Figure 13-2) and multi-objective (Figure 13-3) optimization. While the default selection for multi-objective optimization is maximum number of function evaluations/generations, one can also use consolidation ratio or hypervolume based metrics to terminate the search. Table 13-2 lists the termination criteria for both single-objective and multi-objective optimization.


Termination Criteria

Max Repeat Optimum/Generations: 0.1 (default)

Number of generations: 100 (default)

Figure 13-2: Termination Criteria Dialog for single-objective direct optimization


Termination Criteria

Termination Criteria
- ● Maximum Functions/Generations
- ○ Fixed Consolidation Ratio
- ○ Consolidation Ratio Change
- ○ Hypervolume Change

Generation gap: 10 (default)

Max Repeat Optimum/Generations: 0.1 (default)

Number of generations: 100 (default)

Figure 13-3: Termination Criteria dialog for multi-objective direct optimization

Table 13-2: Termination Criteria options for direct optimization

Item	**Parameter**
Termination Criterion	MOO performance metric*: Consolidation Ratio | Variable Consolidation Ratio | Hypervolume *No information needed for maximum function criterion
Generation gap	Interval to calculate MOO performance metrics

Normalized hypervolume change threshold	Threshold value for the change in normalized hypervolume
Utility fraction cutoff	Parameter F defining bound (`CR`$_i$`/F`) on the variation in the consolidation ratio
Consolidation ratio threshold	Threshold value of the consolidation ratio
Max Repeat Optimum/Generations	Fraction of the limit on the total number of generations. This fraction acts as a limit on the number of repeated solutions.
Number of generations	Maximum number of generations. If the termination criteria described above are reached first, LS-OPT will terminate and not perform the maximum number of generations.

14. Probabilistic Modeling and Tasks

This chapter summarizes the specifications for probabilistic problems, such as tasks, variable setup, constraint definition etc. It also provides additional probabilistic task-specific details of these definitions. Probabilistic evaluations investigate the effects of uncertainties in the system parameters on the responses. Based on the uncertainty model and problem specification, the statistics of variation of the system responses, such as the nominal value of the response, reliability, and extreme values, can be computed.

The results can be viewed using the Viewer, Chapter 16. The simulation Statistical Tools, Scatter, Parallel Coordinate and Correlation Matrix plots as well as History plot statistics options are pertinent to a pure Monte Carlo (MC) analysis. For a metamodel-based Monte Carlo evaluation, the Accuracy, Sensitivities, and Stochastic Contribution plots are relevant in addition to the Statistical Tools, Scatter, History and Correlation Matrix plots. The LS-DYNA results can be investigated for possible bifurcations using DYNAStats described in Chapter 18. More background on the probabilistic methods is given in Chapter 29, while example problems can be found in Chapter 22.

14.1. Probabilistic problem modeling

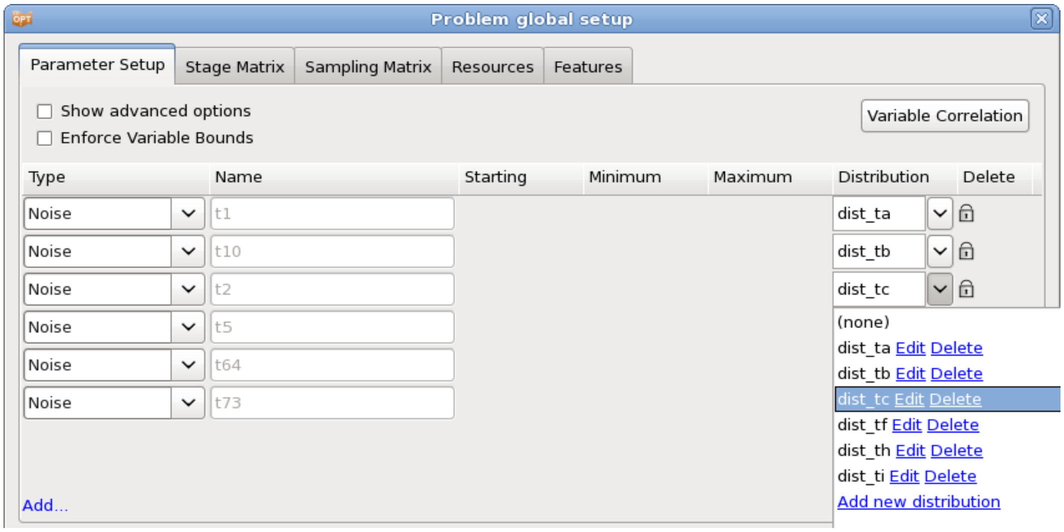
The definition of a probabilistic problem has several differences and additional features compared to a deterministic problem. The specifications for introducing probabilistic effects are:

1. Modeling of uncertainties: The source of the variation can be the variation of the design variables (control variables) as well as the variation of noise variables, whose value is not under the control of the analyst such as the variation in a load. The variation of the system parameters is described by:
   - Defining a statistical distribution , Section 14.2
   - Assigning the statistical distributions to design variables, Section 14.2
2. Definition of the probabilistic task: The available task options are Direct Monte Carlo Analysis (Section 14.5), Metamodel-based Monte Carlo Analysis (Section 14.6) and RBDO/Robust Parameter Design (Section 14.7).
3. Additional task-dependent problem specifications:
   - Experimental Design: For Monte Carlo analysis, a suitable sampling strategy based on the variable statistical distributions is needed. This is not the case in metamodel-based tasks.
   - Objective and constraint: Constraint bounds are used as failure limits for reliability computations. In the case of RBDO a target failure probability is also needed. In robust design, the objective is

to provide a design that is least sensitive to slight variations of the design. This can be achieved by minimizing the standard deviation of the response.

14.2. Probabilistic distributions

The most common way of describing the randomness or uncertainty of an input is through probabilistic distributions associated with random variables. The definition of a probabilistic distribution using the Distribution menu of the Parameter Setup panel in the LS-OPT GUI is presented in Figure 14-1. A distribution can be defined using “Add new distribution”. It is not required for a distribution to be associated with a variable. Many design variables can refer to a single distribution. New distribution definitions can be added and already defined distributions edited by using the **Statistical Distribution** dialog accessible from the **Distribution** menu in the **Parameter Setup** panel, Figure 14-1. The **Distribution** menu is also used to assign a distribution to a parameter. For each distribution, a name has to be specified, and the type selected. Additional parameters to be specified are described in the following sections for each distribution type.



Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	t1				dist_ta	
Noise	t10				dist_tb	
Noise	t2				dist_tc	
Noise	t5					
Noise	t64					
Noise	t73					

Figure 14-1: Setup Dialog, Parameter Setup: Definition of Probabilistic Distributions

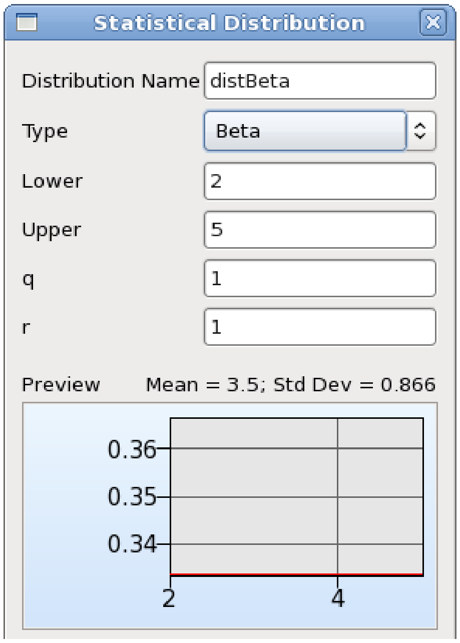
Beta distribution

The beta distribution is quite versatile as well as bounded by two limits. The shape of the distribution is described by two parameters *q* and *r*, Table 14-1. Swapping the values of q and r produces a mirror image of the distribution.

Table 14-1: Parameters defining a Beta distribution

Item	Description
Lower	Lower Bound
Upper	Upper Bound

Q	Shape parameter q
R	Shape parameter r


Statistical Distribution

Distribution Name: distBeta

Type: Beta

Lower: 2

Upper: 5

q: 1

r: 1

Preview Mean = 3.5; Std Dev = 0.866

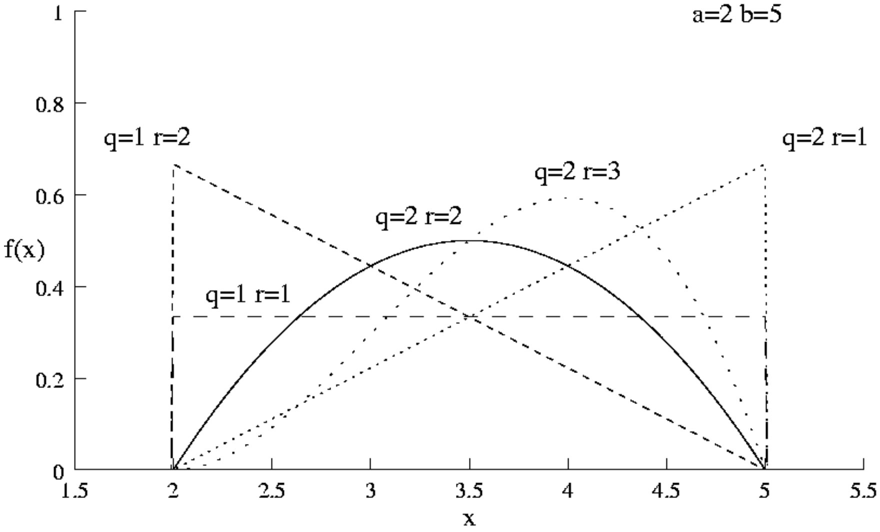
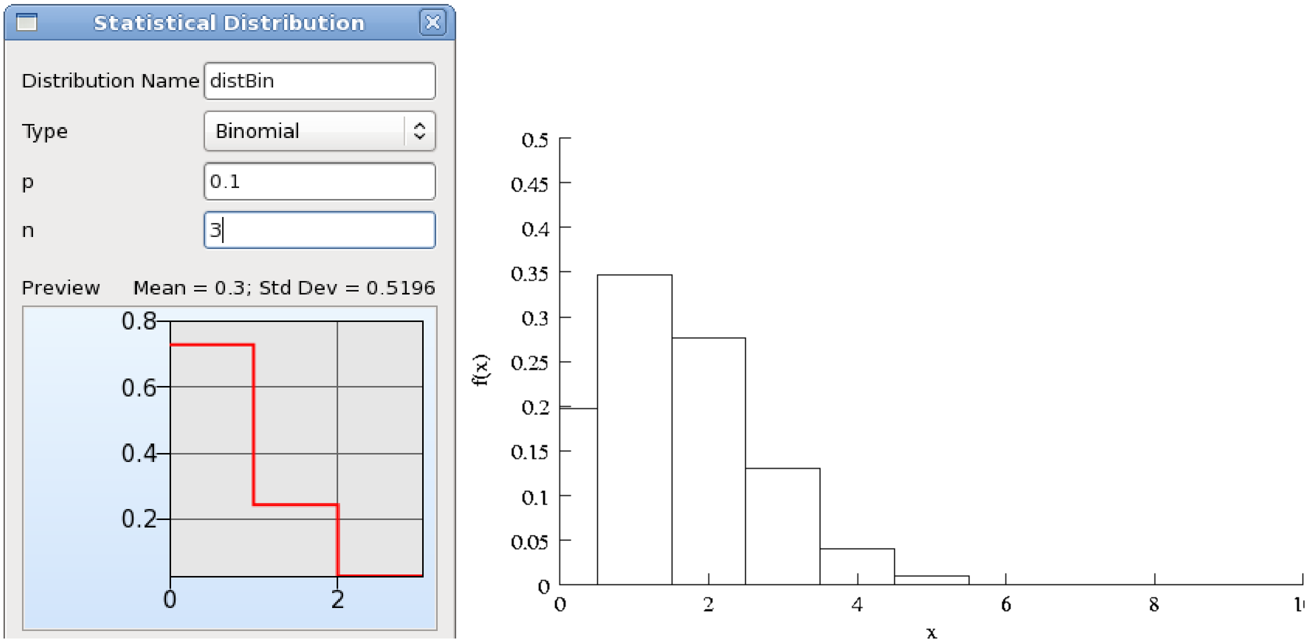


Figure 14-2: Beta distribution

Binomial distribution

The binomial distribution is a discrete distribution describing the expected number of events for an event with probability *p* evaluated over *n* trials, Table 14-2. For *n=1*, it is the Bernoulli distribution (experiments with two possible outcomes — success or failure) with probability of success *p*.



Statistical Distribution

Distribution Name: distBin

Type: Binomial

p: 0.1

n: 3

Preview Mean = 0.3; Std Dev = 0.5196

Figure 14-3: Binomial distribution

Table 14-2: Parameters defining a Binomial distribution

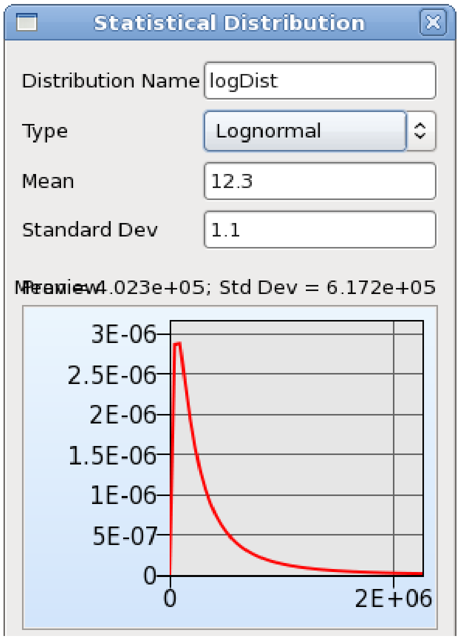
Item	Description
P	Probability of event (Success)
N	Number of trials

Lognormal distribution

If X is a lognormal random variable with parameters μ and σ, Table 14-3, the random variable $Y = \ln X$ has a normal distribution with mean μ and variance σ^2.

Table 14-3: Parameters defining a Lognormal distribution

Item	Description
Mean	Mean value in logarithmic domain
Standard Dev	Standard deviation in logarithmic domain



Statistical Distribution

Distribution Name logDist

Type Lognormal

Mean 12.3

Standard Dev 1.1

Preview Mean = 4.023e+05; Std Dev = 6.172e+05

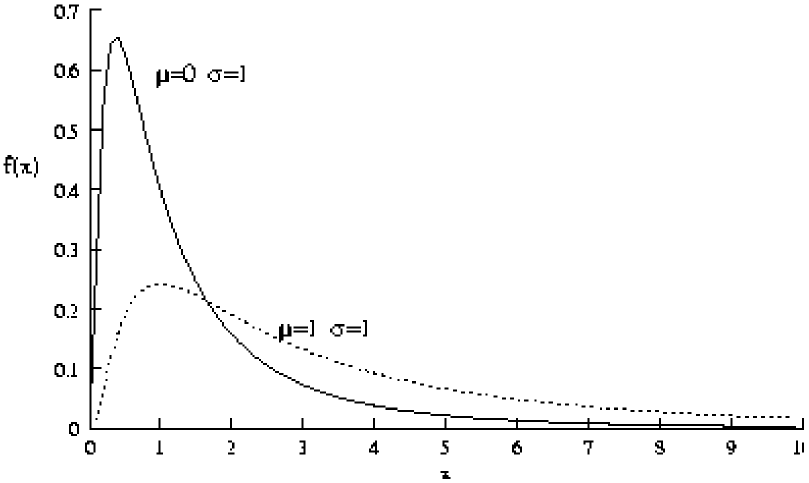
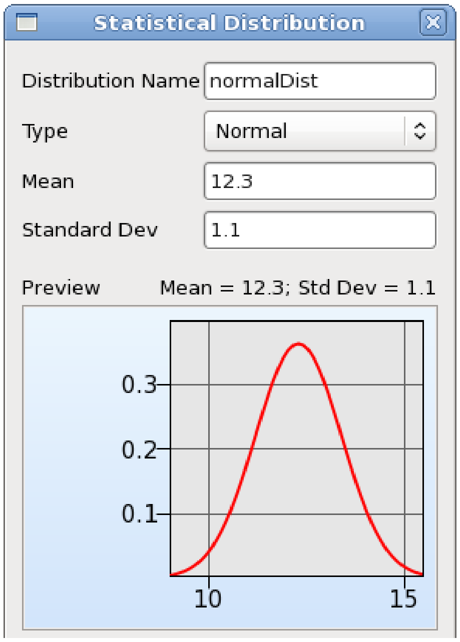


Figure 14-4: Lognormal distribution

Normal distribution

The normal distribution is symmetric and centered about the mean μ with a standard deviation of σ.



Statistical Distribution

Distribution Name normalDist

Type Normal

Mean 12.3

Standard Dev 1.1

Preview Mean = 12.3; Std Dev = 1.1

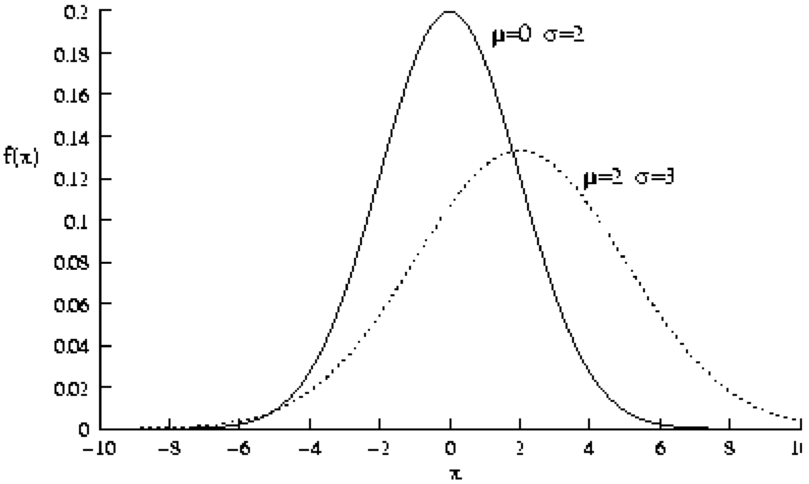


Figure 14-5: Normal Distribution

Table 14-4: Parameters defining a Normal distribution

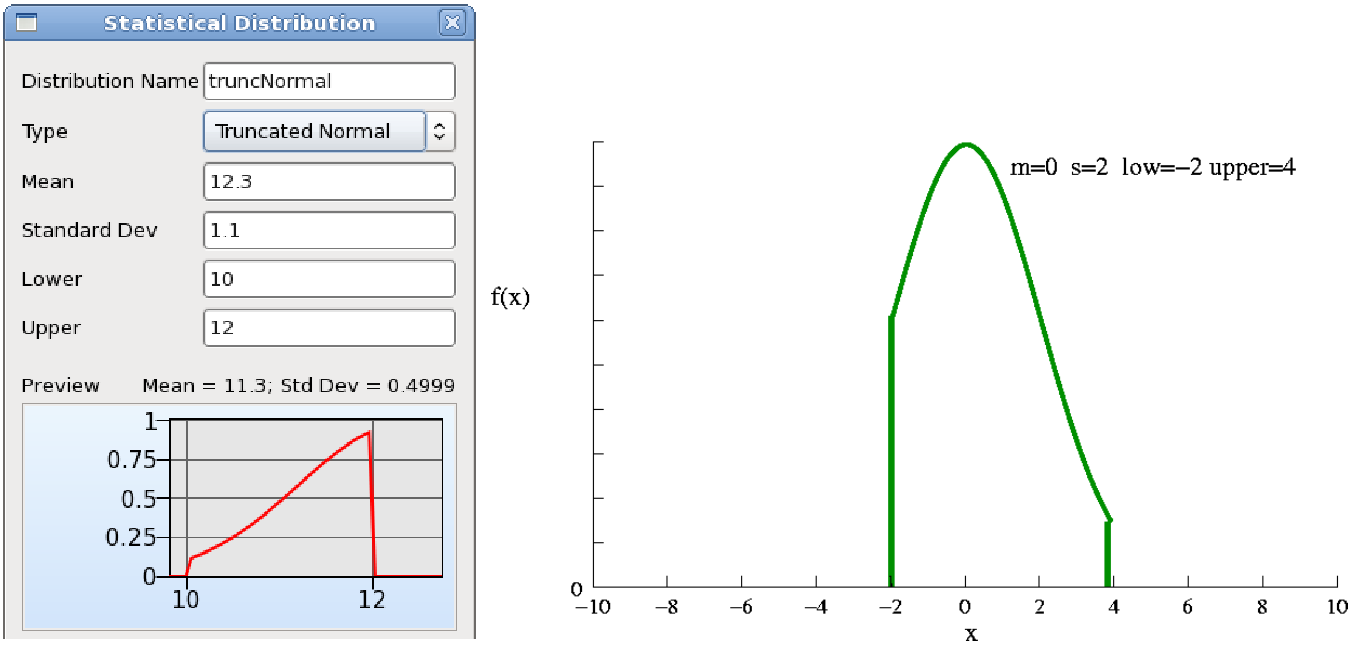
Item	Description
Mean	Mean value
Standard Dev	Standard deviation

Truncated normal distribution

The truncated normal distribution is a normal distribution with the values constrained to be within a lower and an upper bound. This distribution occurs when the tails of the distribution are censored through, for example, quality control.

Table 14-5: Parameters defining a truncated Normal distribution

Item	Description
Mean	Mean value
Standard Dev	Standard deviation
Lower	Lower bound on values
Upper	Upper bound on values



Statistical Distribution

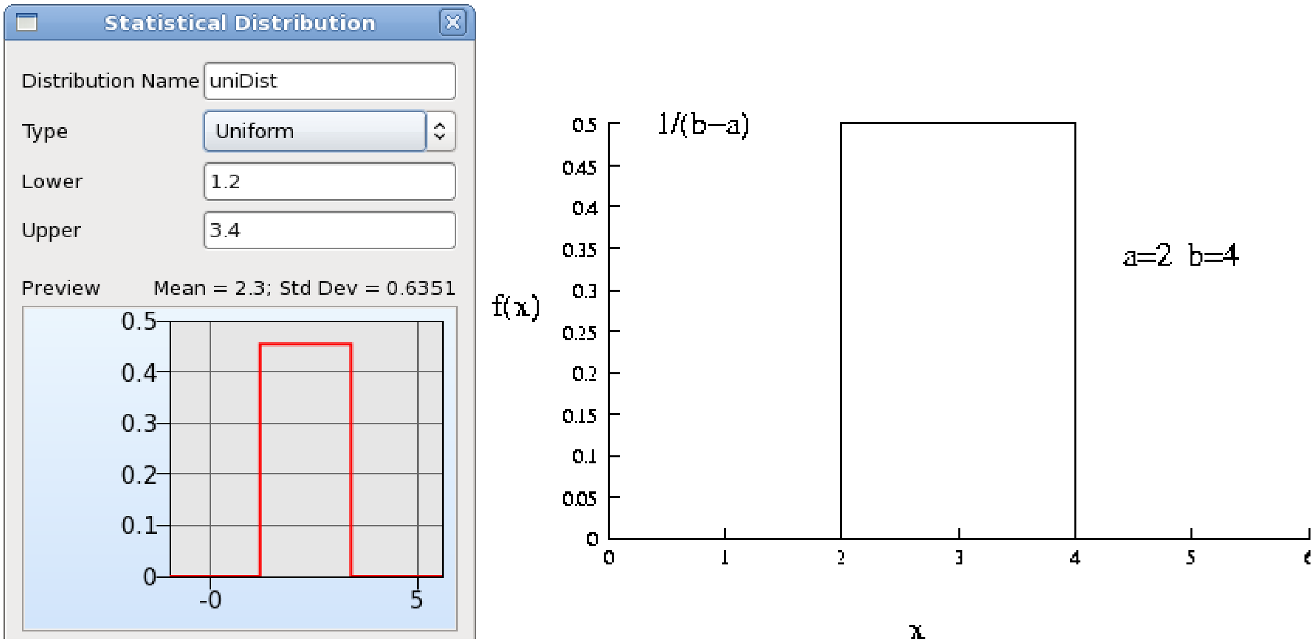
Distribution Name	truncNormal
Type	Truncated Normal
Mean	12.3
Standard Dev	1.1
Lower	10
Upper	12

Preview Mean = 11.3; Std Dev = 0.4999

Figure 14-6: Truncated Normal Distribution

Uniform distribution

The uniform distribution has a constant value over a given range.



Statistical Distribution

Distribution Name: uniDist

Type: Uniform

Lower: 1.2

Upper: 3.4

Preview: Mean = 2.3; Std Dev = 0.6351

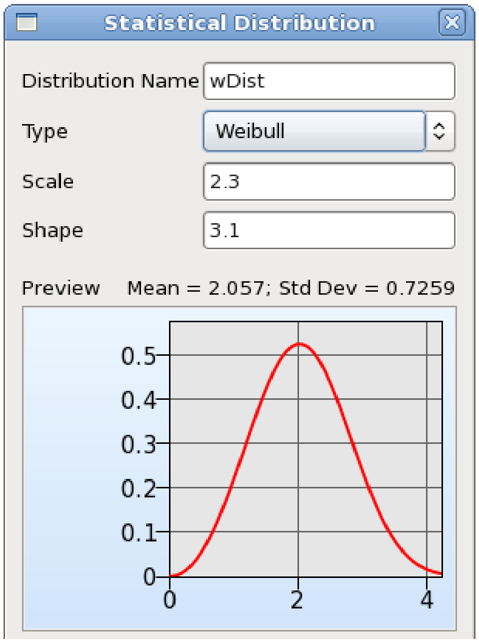
Figure 14-7: Uniform Distribution

Table 14-6: Parameters defining a Uniform distribution

Item	**Description**
Lower	Lower bound
Upper	Upper bound

Weibull distribution

The Weibull distribution is quite versatile – it has the ability to take on various shapes. The probability density function is skewed to the right, especially for low values of the shape parameter.



Statistical Distribution

Distribution Name	wDist
Type	Weibull
Scale	2.3
Shape	3.1

Preview Mean = 2.057; Std Dev = 0.7259

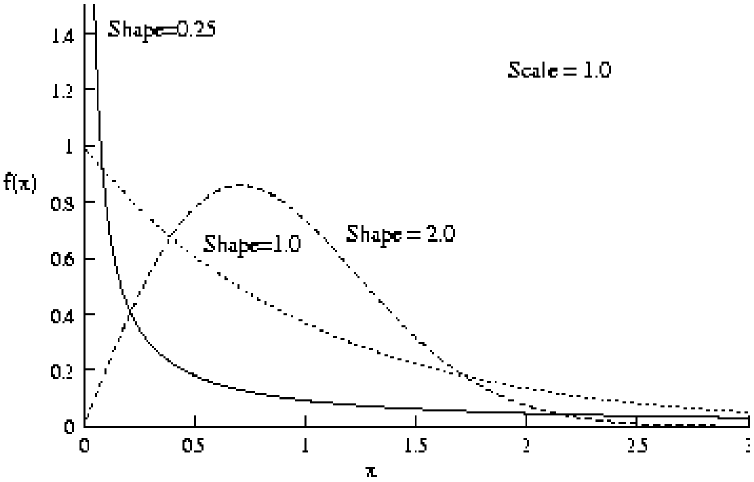


Figure 14-8: Weibull distribution

Table 14-7: Parameters defining a Weibull distribution

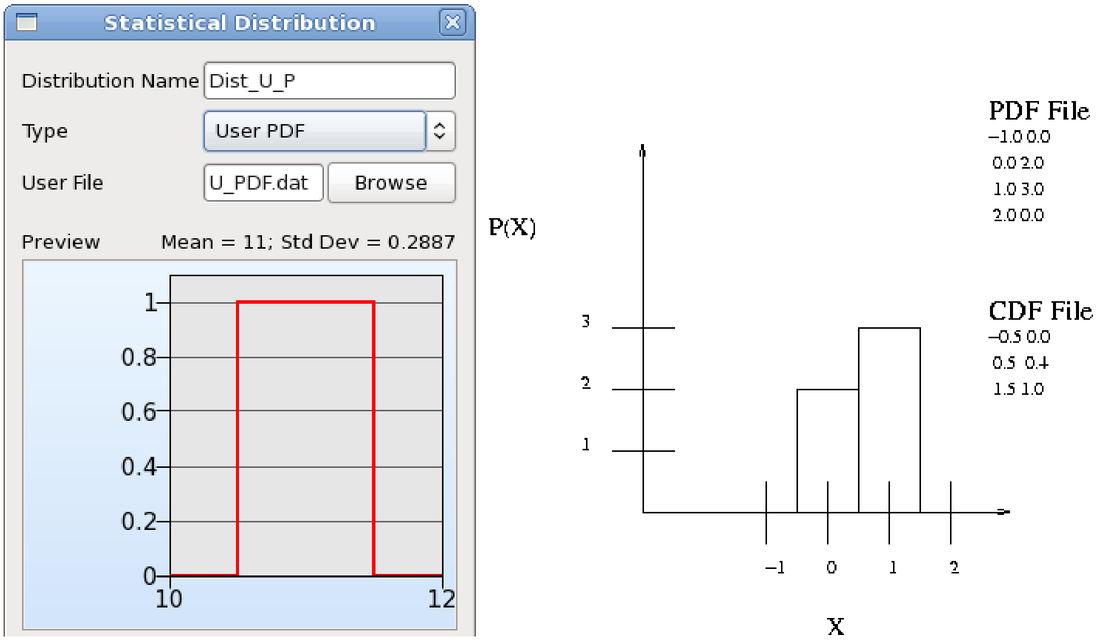
Item	Description
Scale	Scale parameter
Shape	Shape parameter

User defined distribution

A user-defined distribution is specified by referring to the file containing the distribution data. The probability density is to be assumed piecewise uniform and the cumulative distribution to be piecewise linear. Either the PDF or the CDF data can be given:

- **PDF distribution:** The value of the distribution and the probability at this value must be provided for a given number of points along the distribution. The probability density is assumed to be piecewise uniform at this value to halfway to the next value; both the first and last probability must be zero.
- **CDF distribution:** The value of the distribution and the cumulative probability at this value must be provided for a given number of points along the distribution. It is assumed to vary piecewise linearly. The first and last value in the file must be 0.0 and 1.0 respectively.

Lines in the data file starting with the character '$' will be ignored.



Statistical Distribution

Distribution Name: Dist_U_P

Type: User PDF

User File: U_PDF.dat

Preview: Mean = 11; Std Dev = 0.2887

Figure 14-9: User defined distribution

Table 14-8: Parameters defining a User defined distribution

Item	Description
User File	Name of file containing the distribution data

Example: User PDF file

```
$ Demonstration of user defined distribution with
$ piecewise uniform PDF values
$ x PDF
$ First PDF value must be 0
-5                0.00000
-2.5              0.11594
 0                0.14493
 2.5              0.11594
$ Last PDF value must be 0
 5                0.00000
```

Example: User CDF file

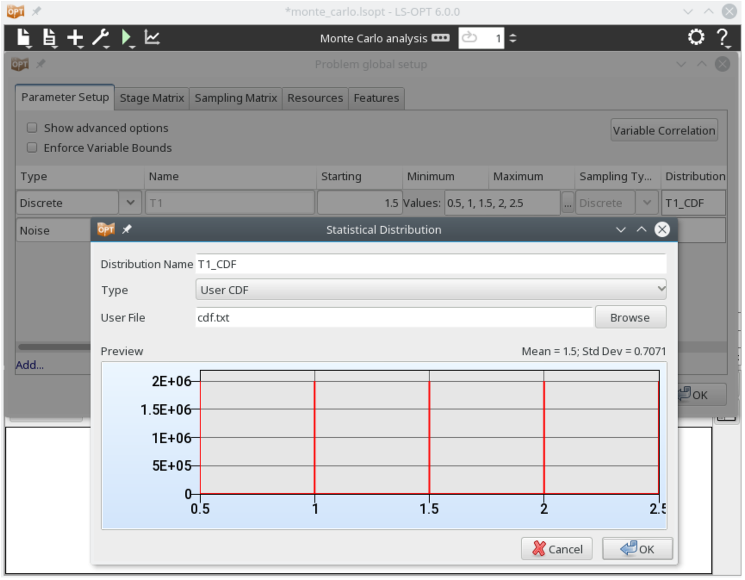
```
$ Demonstration of user defined distribution with
$ piecewise linear CDF values
$ x CDF
$ First CDF value must be 0
-5                 0.00000
-4.5              0.02174
-3.5              0.09420
-2.5              0.20290
```

```
-1.5            0.32609
-0.5            0.46377
0.5             0.60870
1.5             0.73913
2.5             0.85507
3.5             0.94928
$ Last CDF value must be 1
1.00000
```

User-defined discrete distribution

A discrete distribution can be defined using the user-defined CDF or PDF option with the distribution data specified in a text file. The noise variables are defined completely by the distributions given in the file, however in case of control variables (continuous or discrete), it is important to note that the associated distributions only provide variation around the nominal value. Therefore, for Monte Carlo analysis to follow the user-defined discrete distribution, a starting value at the mean of the data set should be used. Even if the mean value is not part of values used to define the distribution, it has to be added to the set of discrete values to be used as starting value of the control variable.

An example for discrete distribution is shown below. The thickness variable "T1" is defined as a control variable using a discrete uniform distribution, given in file cdf.txt, with variable values of 0.5, 1, 1.5, 2, and 2.5. The starting value was defined as 1.5, as it is the mean of the data set.



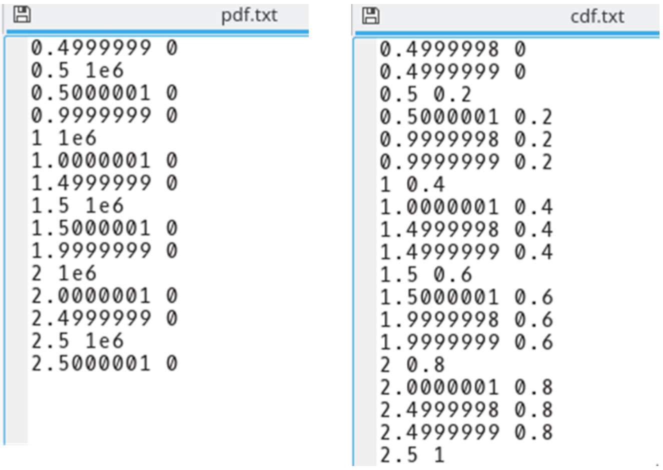
□ Show advanced options
□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Sampling Ty...	Distribution
Discrete	T1	1.5	Values: 0.5, 1, 1.5, 2, 2.5		Discrete	T1_CDF

Distribution Name T1_CDF
Type User CDF
User File cdf.txt
Mean = 1.5; Std Dev = 0.7071

Figure 14-10: Discrete distribution example setup

The user-defined PDF and CDF files for this example are shown below.



pdf.txt

```
0.4999999 0
0.5 1e6
0.5000001 0
0.9999999 0
1 1e6
1.0000001 0
1.4999999 0
1.5 1e6
1.5000001 0
1.9999999 0
2 1e6
2.0000001 0
2.4999999 0
2.5 1e6
2.5000001 0
```

cdf.txt

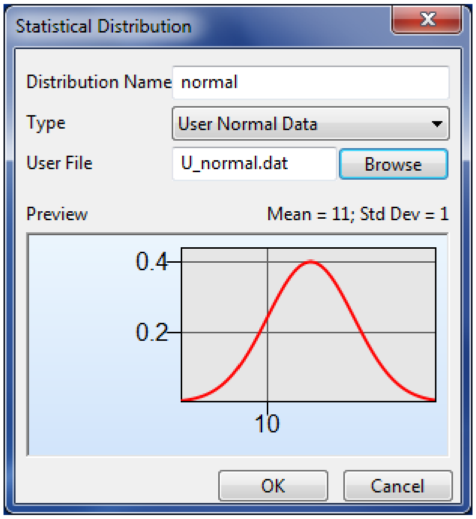
```
0.4999998 0
0.4999999 0
0.5 0.2
0.5000001 0.2
0.9999998 0.2
0.9999999 0.2
1 0.4
1.0000001 0.4
1.4999998 0.4
1.4999999 0.4
1.5 0.6
1.5000001 0.6
1.9999998 0.6
1.9999999 0.6
2 0.8
2.0000001 0.8
2.4999998 0.8
2.4999999 0.8
2.5 1
```

Figure 14-11: User-defined discrete PDF and CD

Fitting of user-defined distribution data

User-defined distribution data may be provided in a file in a single column of variable values that are to be fitted. Two distribution functions are available for fitting the data:

- User Normal Data: A normal distribution function is used to fit the user-defined data.
- User Weibull Data: A Weibull distribution function is used to fit the user-defined data.



Statistical Distribution

Distribution Name: normal

Type: User Normal Data

User File: U_normal.dat

Preview: Mean = 11; Std Dev = 1

Figure 14-12: Fitting of user-defined data

Table 14-9: Parameters defining a User defined distribution

Item	Description
User File	Name of file containing the distribution data to be fitted

14.3. Parametrization of distributions

Distributions can be parameterized by using the “&” operator. This allows the distribution parameters to vary during the solution, instead of being fixed numbers. A distribution parameter can be defined as a constant, a transfer variable or a dependent of transfer variables and constants. This has application in multilevel tolerance optimization, in which the distribution bounds may not be fixed a priori (Section 22.5).

14.4. Probabilistic variables

The uncertainty of a probabilistic variable is described by associating it with a statistical distribution. In the LS-OPT GUI, this is done in the Parameter Setup panel (Section 14.2). The statistical distribution defines the mean or nominal value and the variation around this nominal value. The nominal value, the probabilistic counterpart of a deterministic variable, may or may not change during the course of LS-OPT run. This depends on the task and variable type. The two main probabilistic variable types (Figure 14-13) are:

- Noise variables: These variables are completely described by the associated probabilistic distribution. These variables are not controlled at the design and production level, but only at the analysis level. A probabilistic variable can be defined as a noise variable either because the user chooses to study the effect of uncertainty around a fixed mean value or because it may not be possible to control the variable. An example of the later is wind velocity for which a statistical distribution can be defined from measurements, but one cannot design or control it. A noise variable will have the nominal value as specified by the distribution, i.e. it follows the distribution exactly.
- Control variables: Variables that can be controlled in the design, analysis, and production level; for example: a shell thickness. The nominal value can be adjusted during the design optimization phase in order to have a more suitable design. The associated distribution only provides the variation around this nominal value. A probabilistic control variable can be either continuous or discrete. A discrete variable is a special case of a control variable, in which the nominal value can only be among the specified list of values. However, due to uncertainty about the discrete nominal value, the variable can actually have a value that does not belong to the list. In other words the nominal value is discrete, but the variable value is continuous.

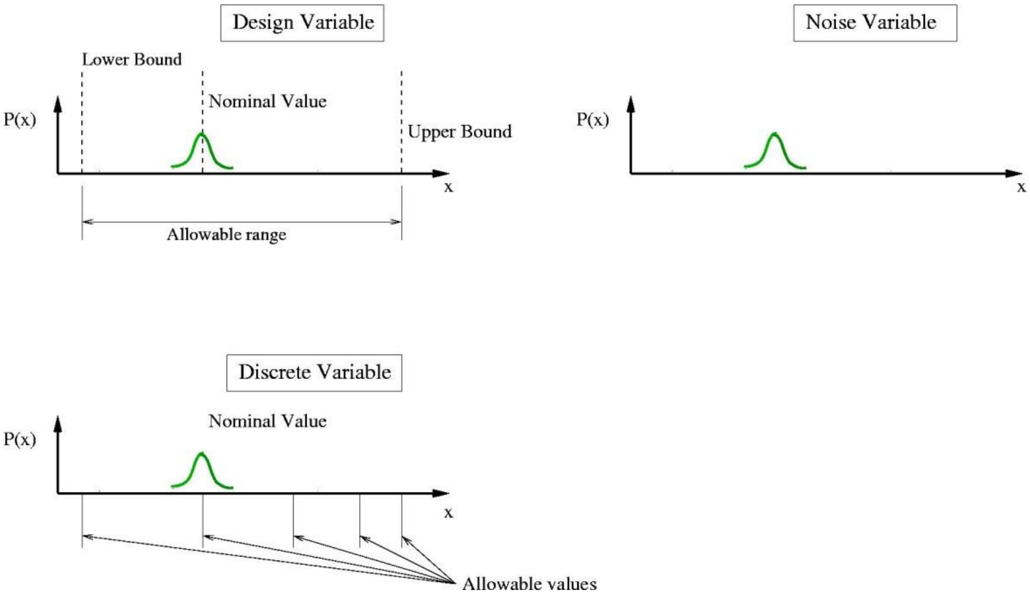


Figure 14-13: Probabilistic variables. The nominal value of a control variable can be adjusted by the optimization algorithm between the lower and upper bound; the probabilistic variation of a design variable is around this nominal value. A noise variable is described completely by the statistical distribution. A discrete variable, like design variable has a nominal value selected by the optimization algorithm; the probabilistic variation of the discrete variable is around this nominal value.

A variable is declared probabilistic by:

- Creating it as a noise variable or
- Assigning a distribution to a control variable.

Three associations between probabilistic variables are possible:

- Their nominal values are the same but their distributions differ
- Their nominal values and distributions are the same
- Their nominal values differ, but they refer to the same distribution.

14.4.1. Setting the nominal value of a probabilistic variable

The specified nominal value is used for a control variable; the associated distribution will be used to describe the variation around this nominal value. For example: a variable with a nominal value of 7 is assigned a normal distribution with μ=0 and σ=2; the values of the variable will be normally distributed around a nominal value of 7 with a standard deviation of 2.

This behavior is only applicable to control variables; noise variables will always follow the specified distribution exactly, i.e. they will have the same nominal value and variation as defined for the associated distribution.

14.4.2. Bounds of a probabilistic variable

The bounds of a control variable are defined by the user (minimum and maximum) like for deterministic variables. It should be noted, however, that if the nominal value of a variable is close to a bounding value, then the bound can be exceeded because of the uncertainty (Figure 14-14). This is the case by default, unless specified otherwise Using “Enforce Variable Bounds” in the Parameter Setup panel.

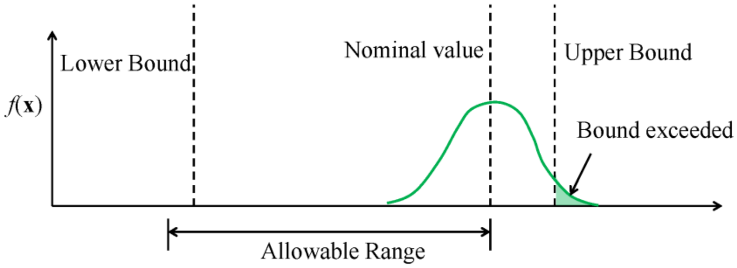


Figure 14-14: Bounds exceeded due to variable uncertainty

Noise variables are completely defined by their distributions; they are not bounded unless specified in the associated distribution. Thus no bounds are required in direct Monte Carlo Analysis. However, in a metamodel-based analysis or optimization, bounds are required even for noise variables to select the samples for metamodel construction. In such tasks, noise variable bounds are defined as multiples of the standard deviation (“Noise Variable Subregion Size”, Table 8-3). By default, two standard deviations are used on either side of the nominal value.

14.5. Direct Monte Carlo analysis

Monte Carlo analysis is used to simulate the uncertainty of variables using random samples based on the associated distribution. The Monte Carlo evaluation will

- Select the random sample points according to a user specified strategy and the statistical distributions assigned to the variables.
- Evaluate the structural behavior at each point.
- Collect the statistics of the responses.

The user must specify the experimental design strategy (sampling strategy) to be used in the Monte Carlo evaluation (Section 9.3). The Monte Carlo, Latin Hypercube and user-defined experimental designs are available. The experimental design will first be computed in a normalized, uniformly distributed design space and then transformed to the distributions specified for the design variables (except in the case of user-defined samples).

Only the variables with a statistical distribution will be perturbed; all other variables will be considered at their nominal value.

The following will be computed:

- Statistics such as the mean and standard deviation for all variables, responses and composites

- Reliability information regarding all constraints:
  - The number of times a specific constraint was violated during the simulation
  - The probability of violating the bounds and the confidence region of the probability
  - A reliability analysis for each constraint assuming a normal distribution of the response.
- The covariance of the responses with the variables can be investigated to get information on how the variation of each design variable contributes to the variation of a response..

The exact value at each point will be used. Defining multiple samplings is not allowed for Monte Carlo analysis; multiple disciplines must share the same samples.

14.6. Monte Carlo analysis using a metamodel

The Monte Carlo analysis will be done using metamodels – response surfaces, neural networks, Kriging or SVR – as prescribed by the user. The analysis can be done using a single iteration or sequentially by selecting the appropriate strategy. Unlike the direct Monte Carlo method, in which the Monte Carlo samples are evaluated using the actual stage solvers, this is a two step process:

1. First the metamodels are constructed based on a few samples evaluated using the actual stage solvers. These samples need not (typically do not) follow the variable statistical distributions. Singlestage and sequential strategy are available for metamodel-based Monte Carlo analysis.
2. Next, Monte Carlo Samples are randomly generated (typically a large number) based on the variable statistical distributions. These samples are evaluated using the metamodels. The number of Monte Carlo points can be set by the user using the *Reliability Resolution* option, Section 12.5. The default value is 10^6. A higher number of samples represents the underlying distribution more closely, and gives more accurate results provided the metamodel approximations are accurate.

Metamodel-based probabilistic analysis or optimization is accompanied by the calculation of stochastic contributions of the variables, Section 14.8. It can be useful to know how the variation of each design variable contributes to the variation of a response. These computations are also known as Stochastic Sensitivity Analysis or Sobol's analysis. The stochastic contribution will be printed for all the responses in a metamodel-based procedure. The stochastic contributions of the variables can also be examined in the Viewer component of the GUI (Section 16.6.2). The amount of variation due to noise or the residuals from the fitting procedure will be indicated. This term is taken as zero for composite functions, as they do not have associated metamodels and corresponding residuals.

The following data will be collected:

- Statistics such as the mean and standard deviation for all responses, composites, and variables
- The reliability information for each constraint:
  - The number of times a specific constraint was violated during the simulation
  - The probability of violating the bounds and the confidence region of the probability.
- Stochastic contributions of variables

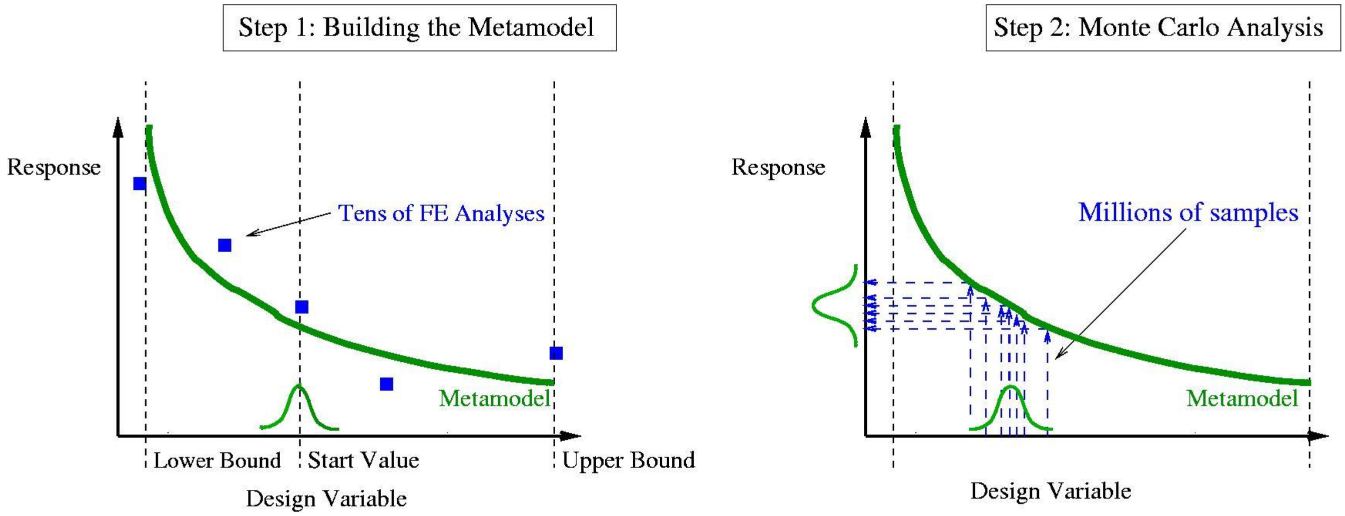


Figure 14-15: Metamodel-based Monte Carlo analysis. The method proceed in two steps: firstly a metamodel is created, and then the Monte Carlo simulation is done using the metamodel and the statistical distribution of the variable. Note that the metamodel for a design/control variable is constructed considering the upper and lower bound on the variable and not considering the statistical distribution. For a noise variable the upper and lower bounds for the creation of the metamodel are selected considering the statistical distribution.

14.7. RBDO/Robust parameter design

To find a robust parameter design, use the task *RBDO/Robust parameter design*, Section 29.8 and Section 29.9, and the strategy *Sequential with Domain Reduction*, Section 4.8.3.

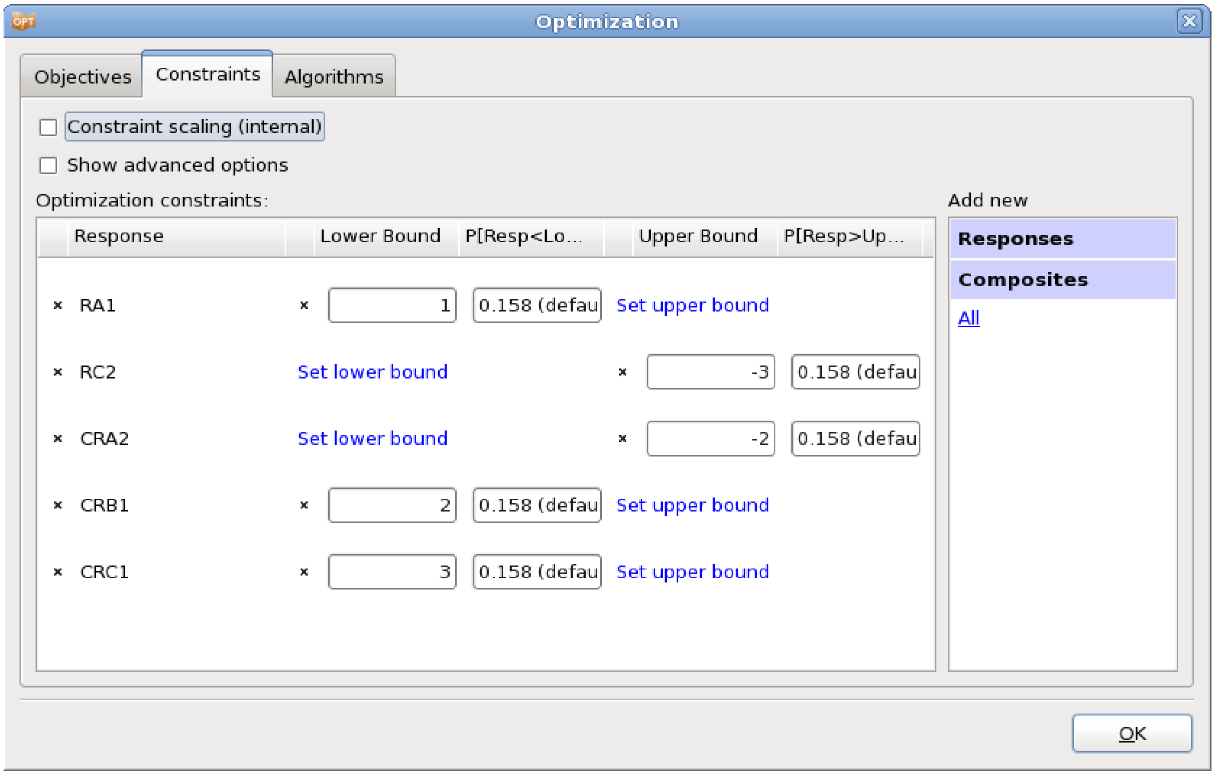
LS-OPT has a reliability/robustness-based design capability based on the computation of the standard deviation of any response. The standard deviation of a response is available as a composite, Section 0, and therefore available for use in a constraint or objective, or in another composite. The theoretical concerns are discussed in Section 29.8.

The method computes the standard deviation of the responses using the same metamodel as used for the deterministic optimization portion of the problem using the First Order Second Moment Method (FOSM), Section 29.4.5. No additional FE runs are therefore required for the probabilistic computations.

The method requires very little information in addition to what is required for deterministic optimization. The following need to be specified:

- Statistical distributions associated with the design variables
- Probabilistic bounds on the constraints.

The statistical distributions associated with the design variables are specified in the same manner as for a Monte Carlo analysis using a metamodel.



□ Constraint scaling (internal)

□ Show advanced options

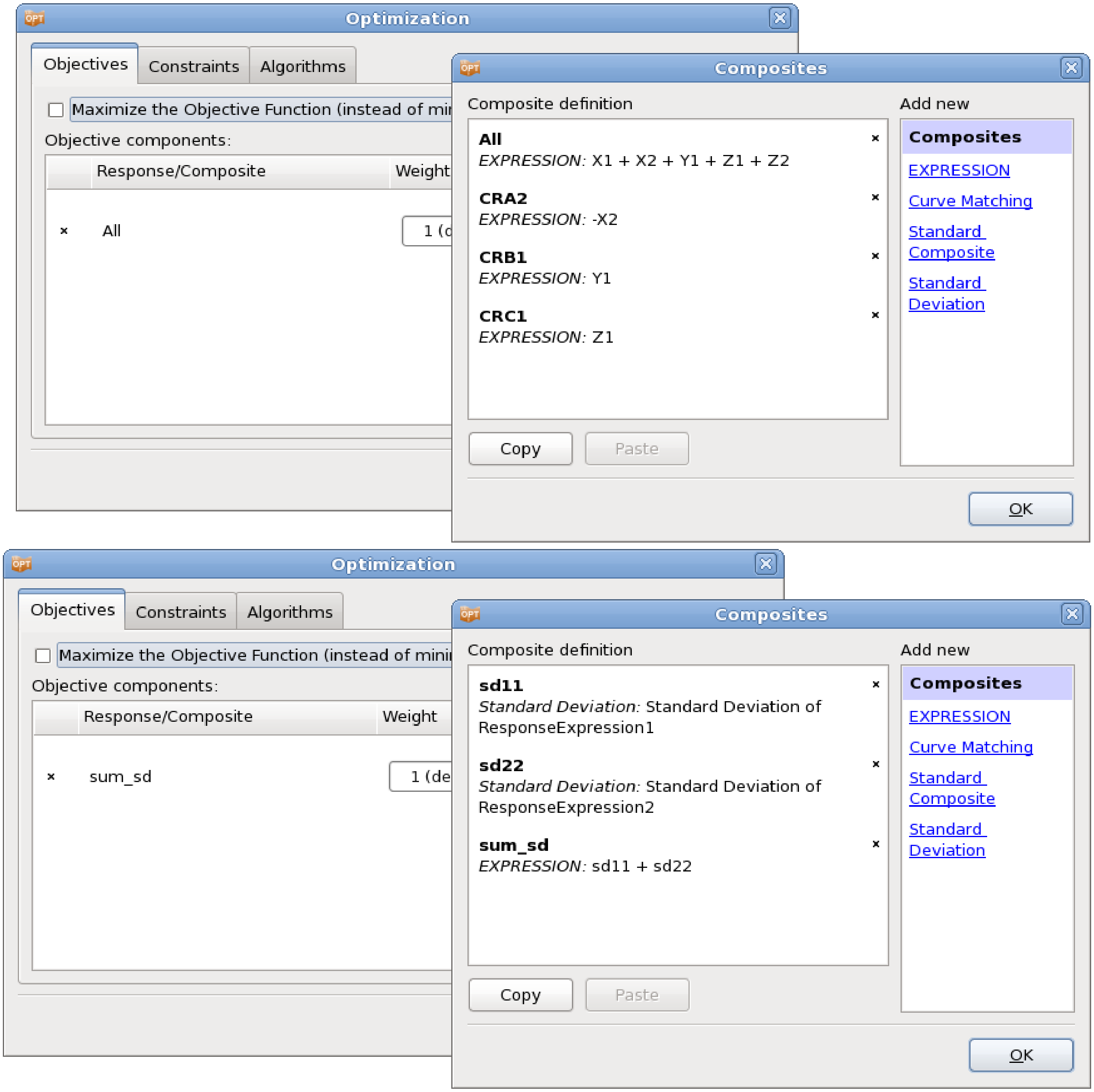
Optimization constraints:

Response	Lower Bound	P[Resp<Lo...	Upper Bound	P[Resp>Up...
× RA1	× 1	0.158 (defau	Set upper bound	
× RC2	Set lower bound		× -3	0.158 (defau
× CRA2	Set lower bound		× -2	0.158 (defau
× CRB1	× 2	0.158 (defau	Set upper bound	
× CRC1	× 3	0.158 (defau	Set upper bound	

Figure 14-16: Probabilistic constraint definition for RBDO. A target failure probability is defined.

The difference between RBDO and robust design lies in the optimization problem formulation; therefore, both capabilities are provided under the same task. In RBDO, "safety" of the design is ensured by the probabilistic bounds on the constraints (target failure probability) while the objective is defined such that it provides a better "deterministic" design goal (e.g. lowest cost or weight calculated at the variable means of the design). In robust design, the objective is to provide a design that is least sensitive to slight variations of the design. This can be achieved by minimizing the standard deviation of the response (Figure 14-17).

One extra consideration is required to select an experimental design for robust analysis: provision must be made to study the interaction between the noise and control variables. Finding a robust design requires that the experimental design considers the $x_i x_j$ cross-terms (considering variables x_i and x_j), and therefore, a linear metamodel should not be used. Thus, when using a polynomial approximation, the order should at least be linear with interaction terms. The x_i^2 and x_j^2 terms can be included for a more accurate variance computation. Non-polynomial metamodels such as RBF, FF, Kriging, SVR etc. can also be used. An example for robust design is presented in Section 22.4.



□ Maximize the Objective Function (instead of mi

Objective components:

	Response/Composite	Weight
×	All	1 (d

Composite definition

All ×
EXPRESSION: X1 + X2 + Y1 + Z1 + Z2

CRA2 ×
EXPRESSION: -X2

CRB1 ×
EXPRESSION: Y1

CRC1 ×
EXPRESSION: Z1

□ Maximize the Objective Function (instead of mini

Objective components:

	Response/Composite	Weight
×	sum_sd	1 (de

Composite definition

sd11 ×
Standard Deviation: Standard Deviation of ResponseExpression1

sd22 ×
Standard Deviation: Standard Deviation of ResponseExpression2

sum_sd ×
EXPRESSION: sd11 + sd22

Figure 14-17: An example of objective function for RBDO (top) and robust design (bottom). Standard deviation is defined as the objective in latter case.

14.8. Stochastic Contribution Analysis

It can be useful to know how the variation of each design variable contributes to the variation of a response. These computations are also known as Stochastic Sensitivity Analysis or Sobol's analysis. The stochastic contribution will be printed for all the responses in a metamodel-based procedure. If no metamodel is available the covariance of the responses with the variables can be investigated. The stochastic contributions of the variables can also be examined in the Viewer, Section 16.6.2.

The amount of variation due to noise or the residuals from the fitting procedure will be indicated. This term is taken as zero for composite functions. The stochastic contribution is computed analytically for polynomial response surfaces. For neural networks, Kriging models, and composite functions, two options are available, Section 12.6:

1. Approximate using second order response surface. The response surface is built using three times the number of terms in the response surface using a central point Latin hypercube experimental design over a range of plus/minus two standard deviations around the mean.
2. Use Monte Carlo. The number of points used will be the same as used for a metamodel based Monte Carlo analysis. A large number of points (10,000 or more) is required. The default of 10,000 points should give the 1 digit of accuracy needed to compare the effects of variables. This option, using 10,000 points, is the default method.

Theoretical concerns are discussed in Section 29.7.

Note that negative values of the variance can occur if computed using the Monte Carlo approach, especially if a small number of Monte Carlo Points is used. In general the analysis should compare the effects of the variables and not the variance. The default of 10,000 points should give the 1 digit of accuracy which means that if the maximum variance is 3×10^{12} then negative values of -3×10^{10} can be ignored as zero being two orders of magnitudes smaller. Inspecting the values printed for the effects of the variables should clarify the situation, because the effects are scaled values.

15. Running the Design Task

This chapter explains simulation job-related information and how to execute a design task from the graphical user interface as well as monitoring the status of the task and the simulation runs from the GUI.

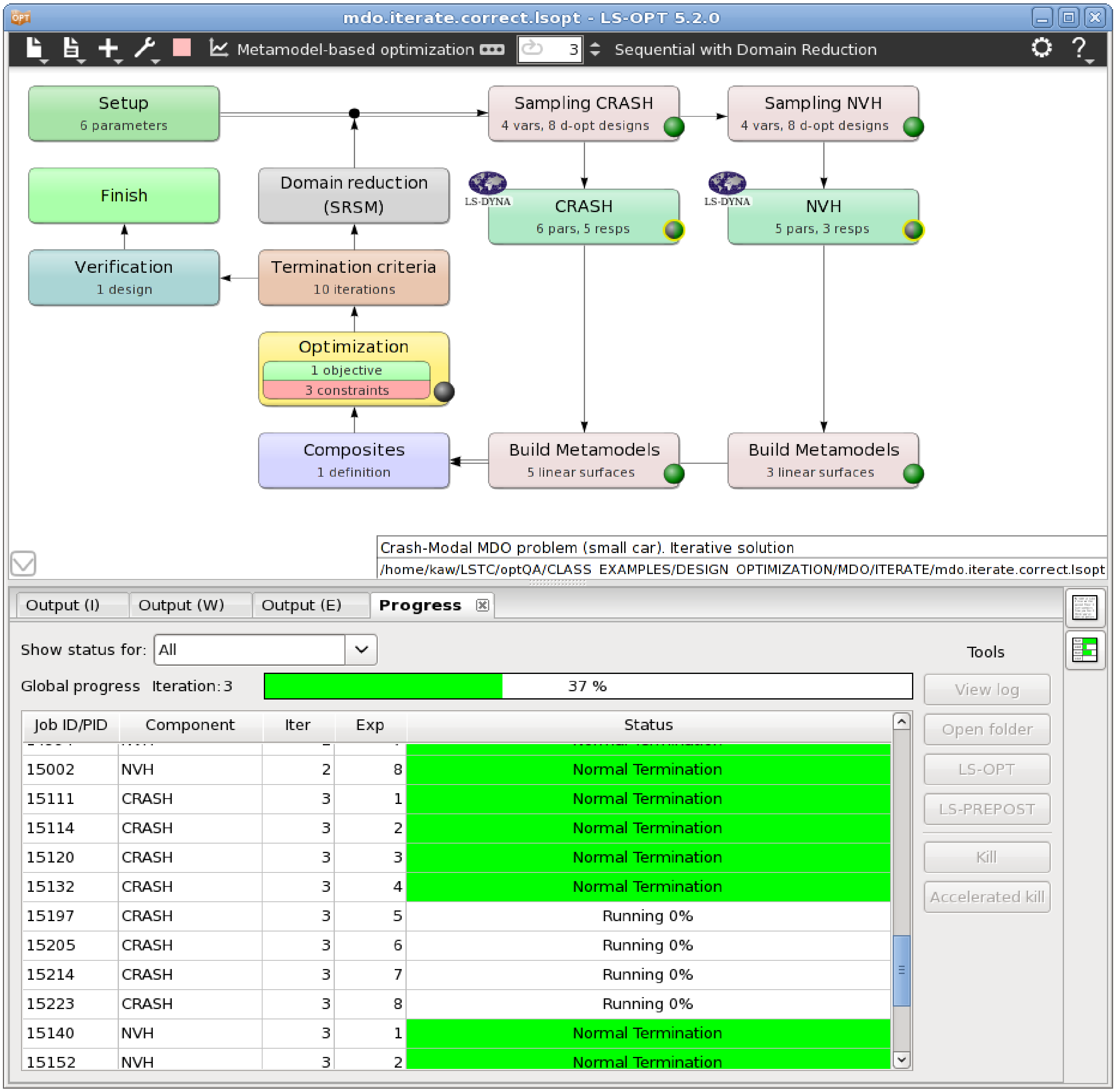
15.1. Running the design task

After setting up the task, run the design task using *Normal Run* or *Baseline Run* from the **Run** menu (▶) in the control bar of the main GUI as described in Section 3.3. If needed, previous results can be deleted using the **Clean** options in the Tools menu (🔧), Section 3.4.

15.2. Analysis monitoring

While running LS-OPT, the status and progress of the task can be visualized in the main GUI, Figure 15-1.

The currently running iteration number is displayed in the control bar at the top (1). The stage LED of the currently running task process is highlighted (glows) in yellow while the green “pie” fraction inside the LED visualizes the solver progress. For the stage LED’s, green and red is used for solver `Normal` and `Error` terminations, respectively. Double-clicking on a stage LED launches the **Progress** dialog described in Section 15.3. The status of individual jobs is also displayed in the *Progress* tab of the integrated output window, Section 15.3.2.



Metamodel-based optimization 3 Sequential with Domain Reduction

Crash-Modal MDO problem (small car). Iterative solution

/home/kaw/LSTC/optQA/CLASS EXAMPLES/DESIGN OPTIMIZATION/MDO/ITERATE/mdo.iterate.correct.lsopt

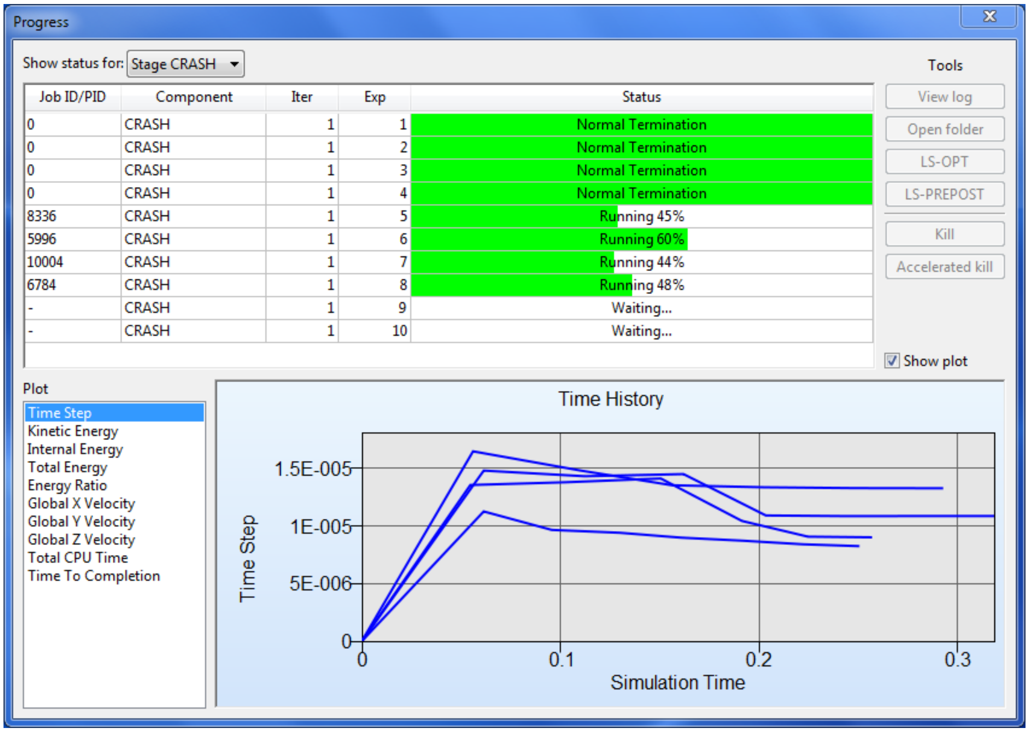
Show status for: All

Global progress Iteration: 3 37 %

Job ID/PID	Component	Iter	Exp	Status
15002	NVH	2	8	Normal Termination
15111	CRASH	3	1	Normal Termination
15114	CRASH	3	2	Normal Termination
15120	CRASH	3	3	Normal Termination
15132	CRASH	3	4	Normal Termination
15197	CRASH	3	5	Running 0%
15205	CRASH	3	6	Running 0%
15214	CRASH	3	7	Running 0%
15223	CRASH	3	8	Running 0%
15140	NVH	3	1	Normal Termination
15152	NVH	3	2	Normal Termination

Figure 15-1: Main GUI showing scheduled jobs in progress

15.3. Job monitoring – the Progress dialog



Show status for: Stage CRASH

Job ID/PID	Component	Iter	Exp	Status
0	CRASH	1	1	Normal Termination
0	CRASH	1	2	Normal Termination
0	CRASH	1	3	Normal Termination
0	CRASH	1	4	Normal Termination
8336	CRASH	1	5	Running 45%
5996	CRASH	1	6	Running 60%
10004	CRASH	1	7	Running 44%
6784	CRASH	1	8	Running 48%
-	CRASH	1	9	Waiting...
-	CRASH	1	10	Waiting...

Figure 15-2: Progress dialog displaying progress of stage runs

Table 15-1: Tools for selected run

Tool	Description	Reference
View log	Opens *job_log* file of selected run	Appendix E.3.4
Open folder	Opens run directory of selected job	-
LS-OPT	Opens LS-OPT GUI if solver type is LS-OPT	-
LS-PREPOST/ Postprocessor	Opens selected run in LS-PREPOST (LS-DYNA only) or selected user-defined postprocessor. A file selector dialog of the selected run directory opens up to let the user decide which file to open in the selected postprocessor.	Section 3.8.1
Kill	Kills selected job	Appendix I.2
Accelerated kill		Appendix I.2
Show plot	Show Time History plot	

The progress of the simulation jobs can be displayed for a selected stage or for all stages. If a job is selected from the list, the tools described in Table 15-1 are enabled.

When using LS-DYNA, the user can also view the progress (time history) of the analysis by selecting one of the available quantities from the *Plot* list (Time Step, Kinetic Energy, Internal Energy, etc.), Figure 15-2.

The **Progress** dialog allows a graphical indication of the job progress with the green horizontal bars linked to estimated completion time, Figure 15-2. This progress is only available for LS-DYNA explicit jobs. The job monitoring is also visible when running remotely through a supported job distribution (queuing) system. The job status is automatically reported at a regular interval.

The text screen output while running both the batch and the graphical version as well as the integrated output window, Section 15.3.2, also report the status as follows:

```
JobID Status                     PID        Remaining
----- ------                     -----      ---------
1 N o r m a l termination!
2 Running                  8427  00:01:38 (91% complete)
3 Running                  8428  00:01:16 (93% complete)
4 Running                  8429  00:00:21 (97% complete)
5 Running                  8430  00:01:13 (93% complete)
6 Running                  8452  00:21:59 (0% complete)
7 Waiting ...
8 Waiting ...
```

In the batch version, the user may also type control-C to get the following response:

```
Jobs started
Got control C. Trying to pause scheduler
Enter the type of sense switch:
sw1: Terminate all running jobs
sw2: Get a current job status report for all jobs
t: Set the report interval
v: Toggle the reporting status level to verbose
stop: Suspend all jobs
cont: Continue all jobs
c: Continue the program without taking any action
Program will resume in 15 seconds if you do not enter a choice switch:
```

If `v` is selected, more detailed information of the jobs is provided, namely event time, time step, internal energy, ratio of total to internal energy, kinetic energy and total velocity.

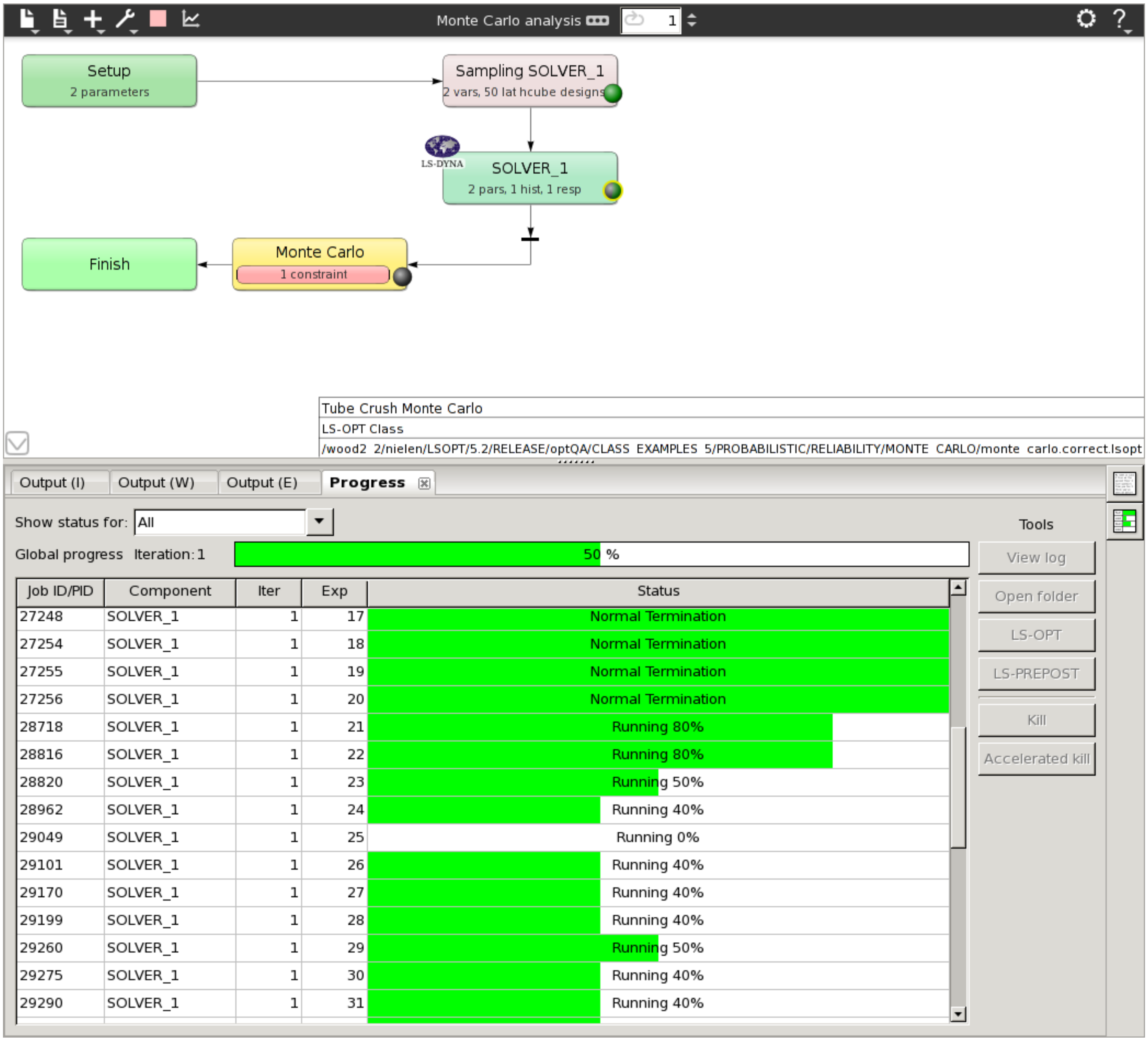
15.3.1. Error termination of a solver run

The job scheduler will mark an error-terminated job to avoid termination of LS-OPT. For error-terminated solver jobs, the progress bars in the GUI are colored in red. Results of error-terminated jobs are ignored, hence they are not used in the optimization, e.g. to construct metamodels. If there are not enough results to continue, e.g. to construct the approximate design surfaces, LS-OPT will terminate with an appropriate error message.

15.3.2. Integrated output and display window

An integrated window which shows job progress, Figure 15-3, as well as output (comprehensive [I], warnings [W] and errors [E] — Figure 15-4) is also available. The window size can be adjusted or hidden using the ∨

above the top left corner of the progress window. Global progress is shown at the top. The tool functionality (except for *Show plot*) is the same as for the stage-based progress window shown in Figure 15-2 (see also Table 15-1).



Tube Crush Monte Carlo

LS-OPT Class

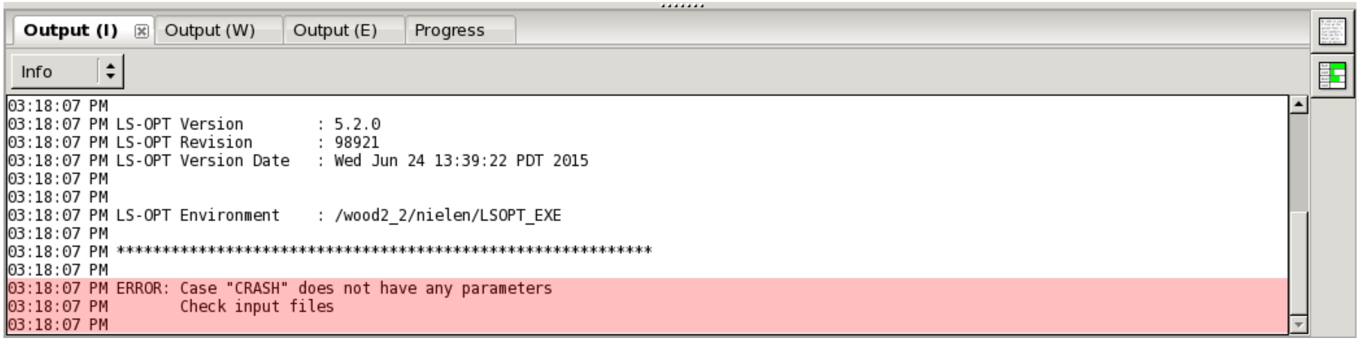
/wood2 2/nielen/LSOPT/5.2/RELEASE/optQA/CLASS EXAMPLES 5/PROBABILISTIC/RELIABILITY/MONTE CARLO/monte carlo.correct.lsopt

Show status for: All

Global progress Iteration: 1 50 %

Job ID/PID	Component	Iter	Exp	Status
27248	SOLVER_1	1	17	Normal Termination
27254	SOLVER_1	1	18	Normal Termination
27255	SOLVER_1	1	19	Normal Termination
27256	SOLVER_1	1	20	Normal Termination
28718	SOLVER_1	1	21	Running 80%
28816	SOLVER_1	1	22	Running 80%
28820	SOLVER_1	1	23	Running 50%
28962	SOLVER_1	1	24	Running 40%
29049	SOLVER_1	1	25	Running 0%
29101	SOLVER_1	1	26	Running 40%
29170	SOLVER_1	1	27	Running 40%
29199	SOLVER_1	1	28	Running 40%
29260	SOLVER_1	1	29	Running 50%
29275	SOLVER_1	1	30	Running 40%
29290	SOLVER_1	1	31	Running 40%

Figure 15-3: Progress dialog



```
03:18:07 PM
03:18:07 PM LS-OPT Version        : 5.2.0
03:18:07 PM LS-OPT Revision       : 98921
03:18:07 PM LS-OPT Version Date   : Wed Jun 24 13:39:22 PDT 2015
03:18:07 PM
03:18:07 PM
03:18:07 PM LS-OPT Environment    : /wood2_2/nielen/LSOPT_EXE
03:18:07 PM
03:18:07 PM ***********************************************************
03:18:07 PM
03:18:07 PM ERROR: Case "CRASH" does not have any parameters
03:18:07 PM        Check input files
03:18:07 PM
```

Figure 15-4: LS-OPT output showing error diagnostic

15.4. Restarting

Restarting is conducted by selecting the appropriate option from the **Run** menu (▶) in the control bar panel of LS-OPTui.

Completed simulation runs will be ignored, while half completed runs will be restarted automatically. However, the user must ensure that an appropriate restart file is dumped by the solver by specifying its name and dump frequency.

The following procedure must be followed when restarting a design run:

1. As a general rule, the run directory structure should not be erased. The reason is that on restart, LS-OPT will determine the status of progress made during a previous run from status and output files in the directories. Important data such as response values (`response.`*n* files), response histories (`history.`*n* files) are kept only in the run directories and may not be available elsewhere (with the exception of the `AnalysisResults_`*n*`.lsda` database in the sampling directory and in special cases, the `StageResults_n.lsda` database in the stage directory).
2. In most cases, after a failed run, the optimization run can be restarted as if starting from the beginning. There are a few notable exceptions:
   - A single iteration has been carried out but the design formulation is incorrect and must be changed. In this case the design formulation must be corrected before re-optimizing iteration 1 using the 'Optimize' repair function in the Tools (🔧) menu (see Section 3.5). If histories or responses are added, the 'Extract Results' repair function in the Tools menu must be used to re-extract the data.
   - Incorrect data was extracted, e.g., for the wrong node or in the wrong direction. In this case, the user must re-extract the results using the 'Extract Results' repair function in the Tools menu after correcting the response definitions, Section 3.5.
   - The user wants to change the response surface type, but keep the original experimental design. In this case the user must use the 'Build Metamodels' repair function in the Tools menu after correcting the metamodel type, Section 3.5.

   After completing the repair functions mentioned above, a normal restart can be executed (▶).

   Note: If the strategy Sequential with domain reduction is used, Section 4.8.3, a restart will only be able to retain the data of the first iteration if more than one iteration were completed. The directories of the other higher iterations must be deleted in their entirety. This can be accomplished by using the '*Clean from current iteration [iter]*' selection in the Tools menu. Unless the database was deleted (by, e.g., using the clean file or a 'Delete' file operation, see Section 5.7), no simulations will be unnecessarily repeated, and the optimization procedure will continue.

3. A restart can be made from any particular iteration by selecting the *Clean from current iteration [iter]* option from the Tools menu, see Section 3.4, and selecting the iteration number. The subdirectories representing this iteration and all higher-numbered iterations will be deleted after confirmation. Then select a Run option to restart.
4. The number of points can be changed for a restart (see Section 9.5.4).

15.5. Output and Result Files

Documentation of all output and result files is available in Appendix E: .

16. Viewing Results

This chapter describes the post-processing of LS-OPT result data using the Viewer.

16.1. Viewer overview

16.1.1. Plot Selector

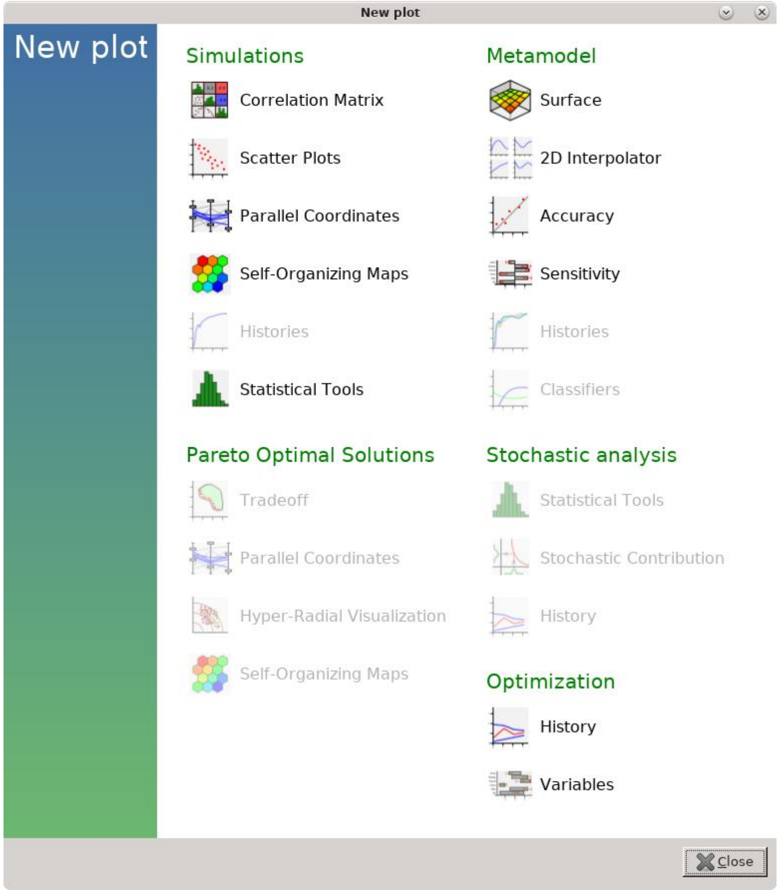


Figure 16-1: Plot Selector

To start the Viewer, select the respective icon (📈) from the control bar of the main GUI or start the executable *viewer* located in the LS-OPT installation directory (Section 16.1.7).

The plots are grouped into five categories (Figure 16-1):

- Simulations,
- Metamodel,
- Optimization,
- Pareto Optimal Solutions and
- Stochastic Analysis.

Depending on the optimization task, the selected options and the database availability, specific plot types are enabled or disabled. For example, in Figure 16-1 *History* plots are disabled due to the absence of history definitions. The *Pareto Optimal Solutions* and *Stochastic Contribution* plots are also disabled. Hovering of the mouse over a particular plot type gives additional information about that plot (Figure 16-2).

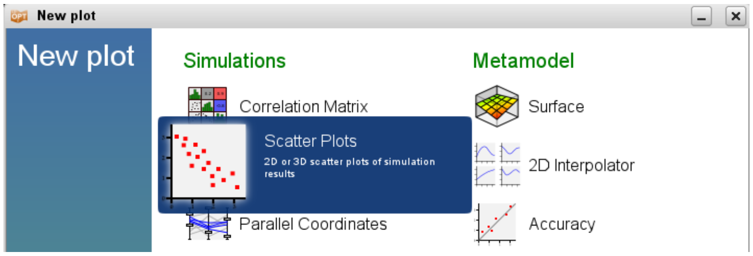


Figure 16-2: Plot Selector with additional information. Note the difference with Figure 16-1.

If plots already exist, the placement of the new plot may be specified in the plot selector, Figure 16-3. The default is to create a new plot. All available options are explained in Table 16-1. For details see Section 16.1.5.

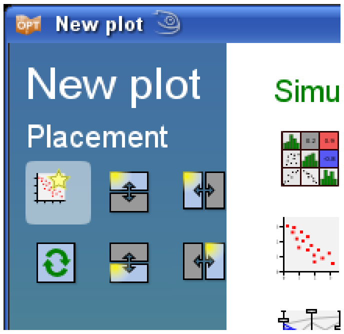


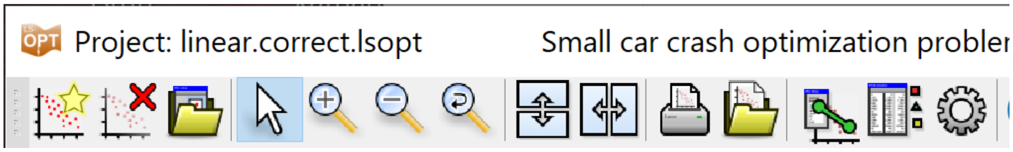
Figure 16-3: Selection for placement of new plot in the plot selector

Table 16-1: Plot placement options

Option	**Description**
☆	Create a new plot window
⟳	Replace current plot
⬍ ⬍ ⬌ ⬌	Split window and place new plot at the highlighted position

16.1.2. General Plot Options

General plot options are available on the toolbar at the top of the plot window, Figure 16-1. Table 16-2 explains the options.



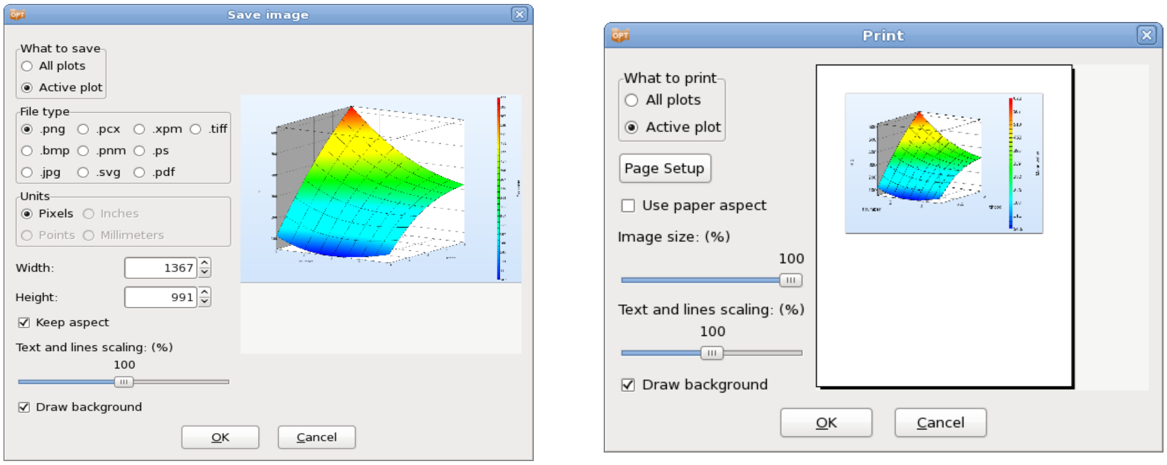
Project: linear.correct.lsopt Small car crash optimization probler

Figure 16-4: General options

Table 16-2: General plot options

Option		**Description**
	New plot	Opens Plot Selector with placement selection for the new plot, Section 16.1.1.
✕	Delete plot	Deletes active plot
	Save plot setup	Saves current plot setup to be reused later, Section 16.1.6.
	Pointer tool [F1]	Rectangular selection (rubber-banding) in plot or clicking marks points or curves and opens Point selection window, Section 16.2.
	Zoom in tool [F2]	Rectangular selection in plot specifies zoom region
	Zoom out [F3]	Clicking on plot zooms out
	Reset zoom	Resets plot to initial range
	Split vertical	Splits plot window vertical, Section 16.1.5.
	Split horizontal	Splits plot window horizontal, Section 16.1.5.
	Print	Prints the current plot, options see Figure 16-5.
	Save image	Saves the current plot, options see Figure 16-5.
	Visualize relations between controls and plots	If several plots are displayed in the same plot window, this option helps to find each plot’s control panel.

▦	Point selection table	Shows/hides a window showing the values of all entities for points in a table, Chapter 16. If point selection changes, this window shows up automatically.
⚙	Settings	Font size options for axes labels and tick marks
Output ⇕		Select table to be used for the plot, Section 16.1.3.
⌄		Control panel visibility and table options, Section 16.1.3. Only active if multiple plots are displayed in the same window.
	Switch to selected table	All plots will switch to show point data from active table in point selection window.



Save image

What to save
○ All plots
● Active plot

File type
● .png ○ .pcx ○ .xpm ○ .tiff
○ .bmp ○ .pnm ○ .ps
○ .jpg ○ .svg ○ .pdf

Units
● Pixels ○ Inches
○ Points ○ Millimeters

Width: 1367
Height: 991
☑ Keep aspect
Text and lines scaling: (%) 100
☑ Draw background

Print

What to print
○ All plots
● Active plot

☐ Use paper aspect
Image size: (%) 100
Text and lines scaling: (%) 100
☑ Draw background

Figure 16-5: Options for printing (right) and saving (left) images

16.1.3. Plot Panel Visibility and Table Options

Panels specific to the shown plot(s) and the backing data table can be controlled by the toolbar in Figure 16-6 and Figure 16-7.

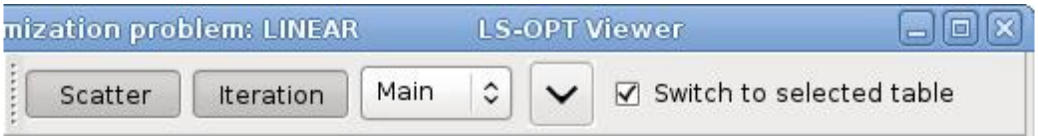
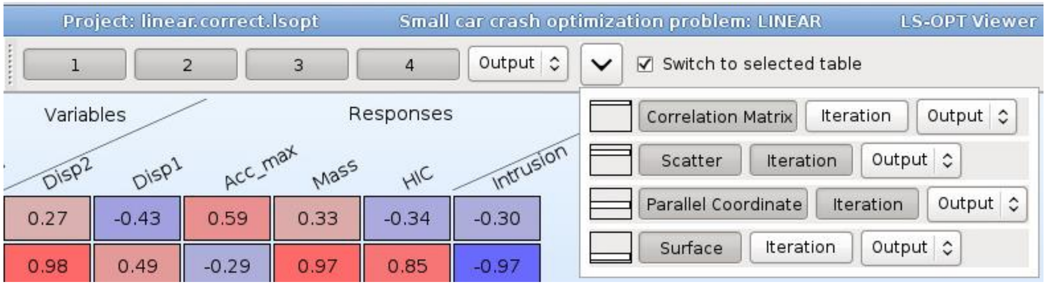


Figure 16-6: Options for plot panels and data table

The buttons in the figure above control the visibility of option panels for the current plot. The leftmost button label corresponds to the shown plot type. The two leftmost buttons are toggle buttons. The first toggles the visibility of panels tied to the specific plot type and the second button toggles the visibility of the iteration panel. The dropdown selects the data table (described further in 17.1.2) that the plot should use. When “Switch to selected table” is checked, the plots automatically switch to use the shown table in the point selection window (see 17.1.1). The checkbox becomes unchecked if the backing table is manually changed using the dropdown.

In the case where multiple plots are shown in the same plot window, the toolbar in Figure 16-6 changes into Figure 16-7. Now numbered buttons are shown where each button toggles the visibility of all panels tied to a certain plot. The table selection changes the data table for all plots. In the case plots display data from different tables the table selection will show “Mixed”. Each plot panel and data table can be controlled separately by using the dropdown button. The interface for each plot is similar to Figure 16-6 with the addition of a visual plot identifier showing where the plot is drawn on screen.



Disp2	Disp1	Acc_max	Mass	HIC	Intrusion
0.27	-0.43	0.59	0.33	-0.34	-0.30
0.98	0.49	-0.29	0.97	0.85	-0.97

Figure 16-7: Plot panel options when multiple plots are shown

16.1.4. Plot Rotation

For all 3D plots, image rotation is performed by holding down the Ctrl key while moving the mouse (same as LS-PREPOST).

16.1.5. Split Window

To display several plots side by side, there are two basic selections available to split the plot window - (i) options to split the window horizontally or vertically in the toolbar at the top of the plot window or (ii) select the new plot together with a placement option for the new plot in the Plot Selector.

If the split window options are used, the plot is repeated with the same settings, which is useful for e.g., displaying 3D surface plots for different responses side by side, Figure 16-8.

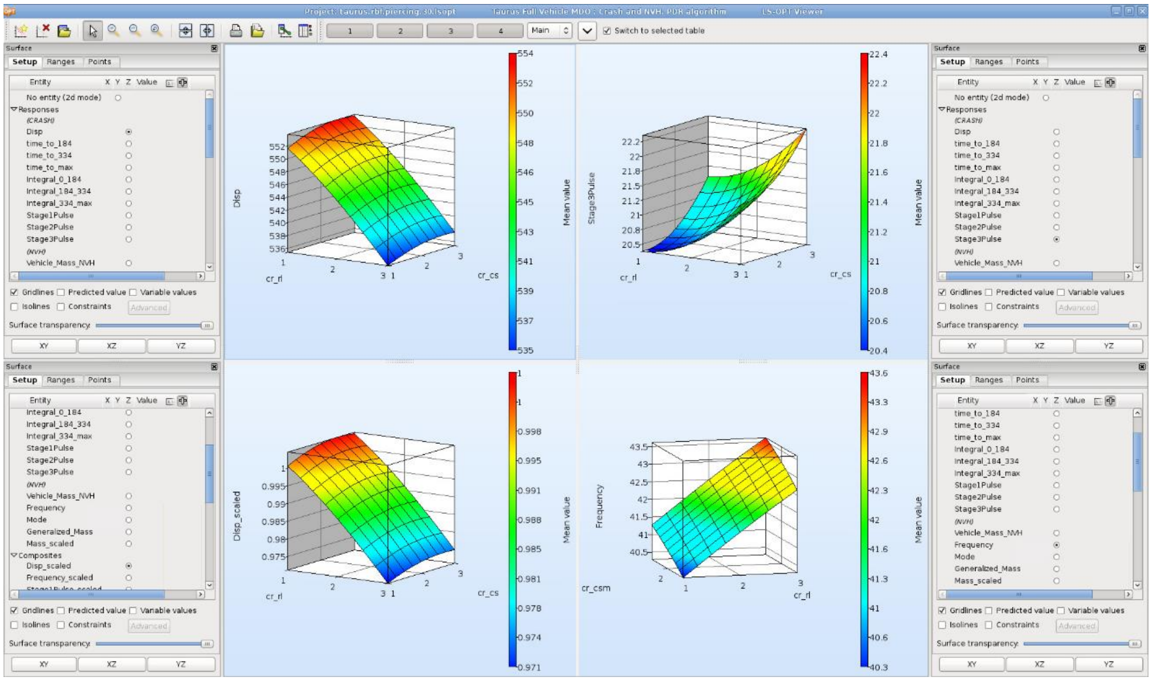


Figure 16-8: Example for split option

If split window options are used several times, the plots may become too small, and as much as possible space on the screen is needed to get a good view. Hence all control panels are detachable or may even be hidden by pushing the respective button in the toolbar at the top of the plot window, Figure 16-9.

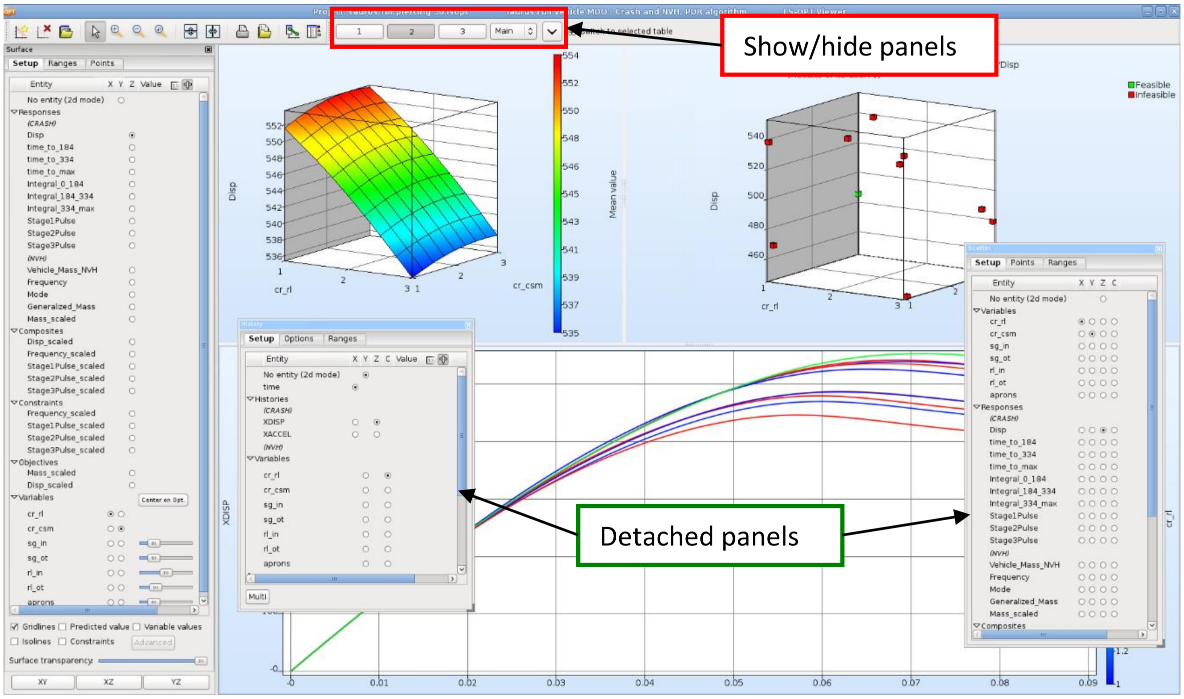


Figure 16-9: Detachable panels

16.1.6. Save Plot Setup

Window splitting and placement selection of new plots allows complex plot setups. To reuse a plot setup several times, even across problems, it may be saved. Later you can bring this plot state back by clicking on the preview in the plot selector, Figure 16-10.

The plot setup is stored in XML format in ~/.LSOPT/<version>/viewerstate/plotname.plot on Linux machines. On Windows, the plot setup is stored in %appdata%\LSOPT\<version>\viewerstate. The full path depends on the Windows version and setup, e.g. C:\Users\user\AppData\Roaming\LSOPT\6.0\viewerstate.

The command line option “-l” makes the viewer load a plot setup from a file immediately, without showing the plot selector. That makes it possible to write a script that generates the plot state XML file and then calls upon the viewer to display the plots. For more details on command line options, see Section 16.1.7.

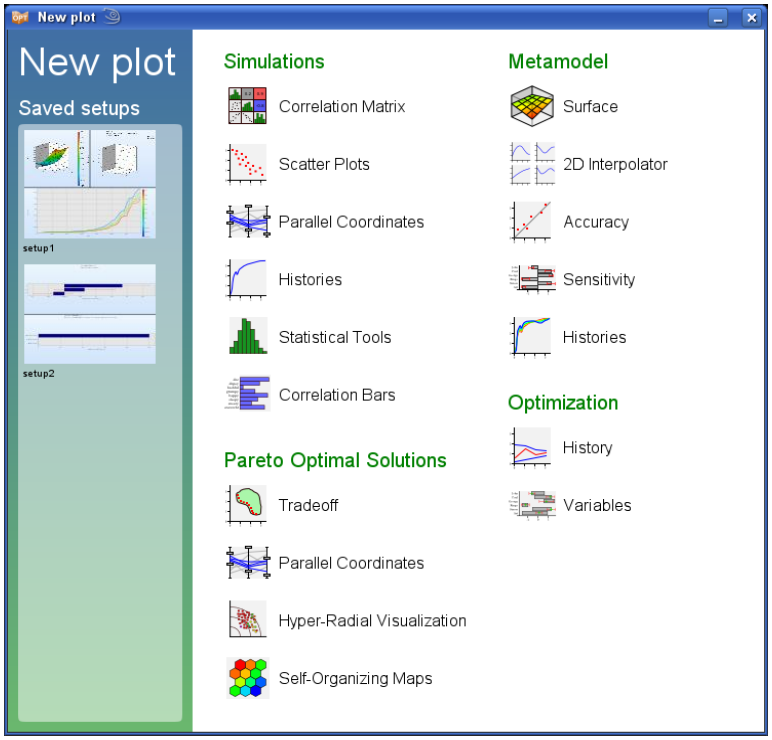


Figure 16-10: Plot Selector with previously saved setups

16.1.7. Command line options

The post-processing tool of LS-OPT may be started from the Viewer Panel in LS-OPTui, or the executable *viewer* located in the LS-OPT installation directory may be called from the command line:

```
viewer [-p <str>] [-l <str>] [-f <str>] [-n <str>] [-h] [--verbose] [.lsopt file]
```

Table 16-3 explains the command line options.

Table 16-3: Command line options

Option	Description	
-p <str>, --show-plot=<str>	Open the given plot, valid plot types are	
	accuracy	Accuracy (Section 16.3.3)
	classifier	Classifier (Section 16.3.6)
	correlation	Correlation Bars (Section 16.2.6)
	corrmatrix	Correlation Matrix (Section 16.2.1)
	history	Histories – Metamodel (Section 16.3.5)
	history_ar	Histories – Simulations (Section 16.2.4)
	hrv	Hyper-Radial Visualization (Section 16.5.3)
	interpol	2D Interpolator (Section 16.3.2)
	opthist	Optimization History (Section 16.4.1)
	parallelcoord	Parallel Coordinates – Pareto Optimal Solutions (Section 16.5.2)
	parallelcoord_ar	Parallel Coordinate – Simulations (Section 16.2.3)
	relhist	Statistics History (Section 16.6.3)
	scatter	Scatter Plots (Section 16.2.2)
	sensitivities	Sensitivity (Section 16.3.4)
	som	Self-Organizing Maps – Pareto Optimal Solutions (Section 16.5.4)
	som_ar	Self-Organizing Maps – Simulations (Section 16.2.4)
	statistics	Statistical Tools (Section 16.2.6)
	stoch	Stochastic Contribution (Section 16.6.2)
	surface	Surface (Section 16.3.1)
	tradeoff	Tradeoff (Section 16.5.1)

	variable Variables (Section 16.4.2)
-l <str>, --load-setup=<str>	Load plot Setup from file, see section **Save Plot Setup**
-f <str>, --format=<str>	Image format to export to (png, bmp, jpg, svg, tiff, pdf, ps)
-n <str>, --filename=<str>	Filename to export to
-h, --help	show help message for command line options
--verbose	generate verbose log messages
.lsopt file	LS-OPT command file. By default, the viewer loads the LS-OPT database called lsopt_db

16.1.8. Iteration Panel

Except for the Optimization and Reliability History plot which displays the iteration history, all plots allow specifications for the iteration data to be shown. The available options depend on the plot type (see Figure 16-11).

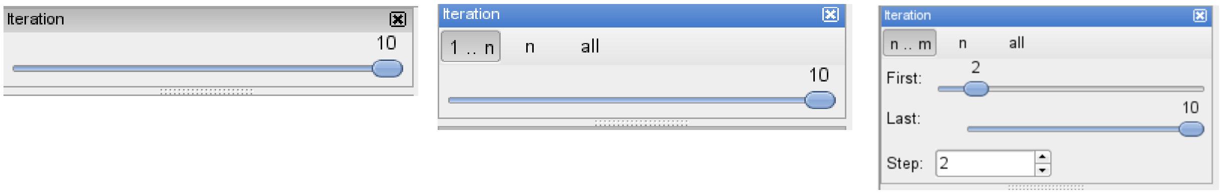


Figure 16-11: Iteration Panel- only current iteration (left), all previous/ all iterations (middle), iteration range and step size (right)

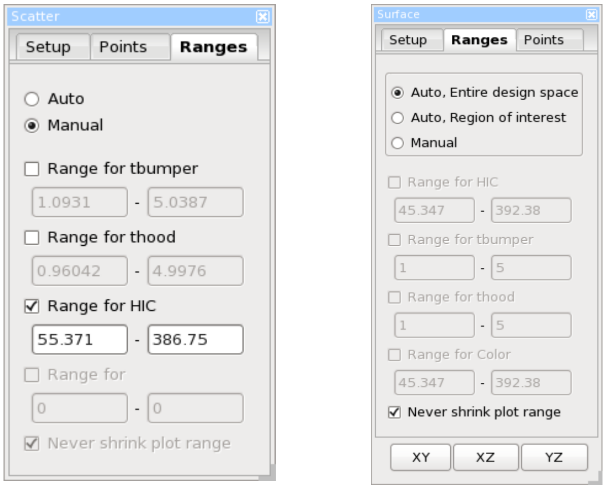
A slider is available to select the current iteration to be plotted. Some plots allow plotting all previous iterations or all iterations, and the Scatter- and Tradeoff plots also allow the specification of a range and a step size, e.g. the selection in the right iteration panel in Figure 16-11 plots iterations 2,4,6,8 and 10.

16.1.9. Ranges and Axes options

Most plots allow specification of the ranges for all plotted entities (Figure 16-12). The user can manually specify any desired plot range by providing the lower and upper bounds using the **Manual** option. Using the checkboxes, manual range specification can be switched on and off for each entity separately. However, there is also an option to let the **Viewer** automatically select the ranges based on the data. If the **Auto** option (default) is used, the range is set to include the minimal and maximal values based on the data.

For surface plots, there are three options for the plot range type when the **Auto** range is selected. The first option (**Auto, Entire design space**) plots the surface across the full design space. The second option (**Auto, region of interest**) uses only the sample selection subregion (Section 26.6) of the selected iteration. The third option (**Auto, GSA Region of interest**) uses only the selected GSA subregion (Section 4.11.1). For scatter plots, only the first and third options are available. For both surface and scatter plots, the option **GSA Region of interest** is available only if subregions for the calculation of global sensitivities are used. For surface plots, the selected GSA region of interest is displayed using a rectangle (two variable axes) or a line (one variable axis) on the xy plane (variable plane). For scatter plots with variable axes, the GSA subregion intervals are displayed using a different color for the relevant parts of the variable axes.

If **never shrink plot range** is selected (Figure 16-12), ranges of the new plot cannot be smaller than the previous plot. The previous ranges are used if all the values in the new plot lie within these ranges; the new plot may have some empty spaces in this case. If the new plot has values outside any of the previous ranges, those ranges are expanded to accommodate the new values. If the user selects a different quantity, the selections made for the shrinking and the plot range type options stay the same. However, the ranges for the new entity are unrelated to the previous entity. Therefore, the ranges of the new plot for a different entity will usually be different from the old plot ranges even if **never shrink plot range** is selected.



○ Auto
● Manual

□ Range for tbumper
1.0931 - 5.0387

□ Range for thood
0.96042 - 4.9976

☑ Range for HIC
55.371 - 386.75

□ Range for
0 - 0

☑ Never shrink plot range

● Auto, Entire design space
○ Auto, Region of interest
○ Manual

□ Range for HIC
45.347 - 392.38

□ Range for tbumper
1 - 5

□ Range for thood
1 - 5

□ Range for Color
45.347 - 392.38

☑ Never shrink plot range

Figure 16-12: Ranges selections

For Histogram plots (Statistical Tools), manual steps for the tick marks can be specified in addition to the manual range selection. The step length determines the number of grid lines and tick marks on the corresponding axis.

16.2. Visualization of Simulation Results

16.2.1. Correlation Matrix

The correlation matrix displays 2D scatter plots, histograms and the linear correlation coefficients calculated from the simulation results of the selected load case for the selected variables, dependents, responses and composites, Figure 16-13.

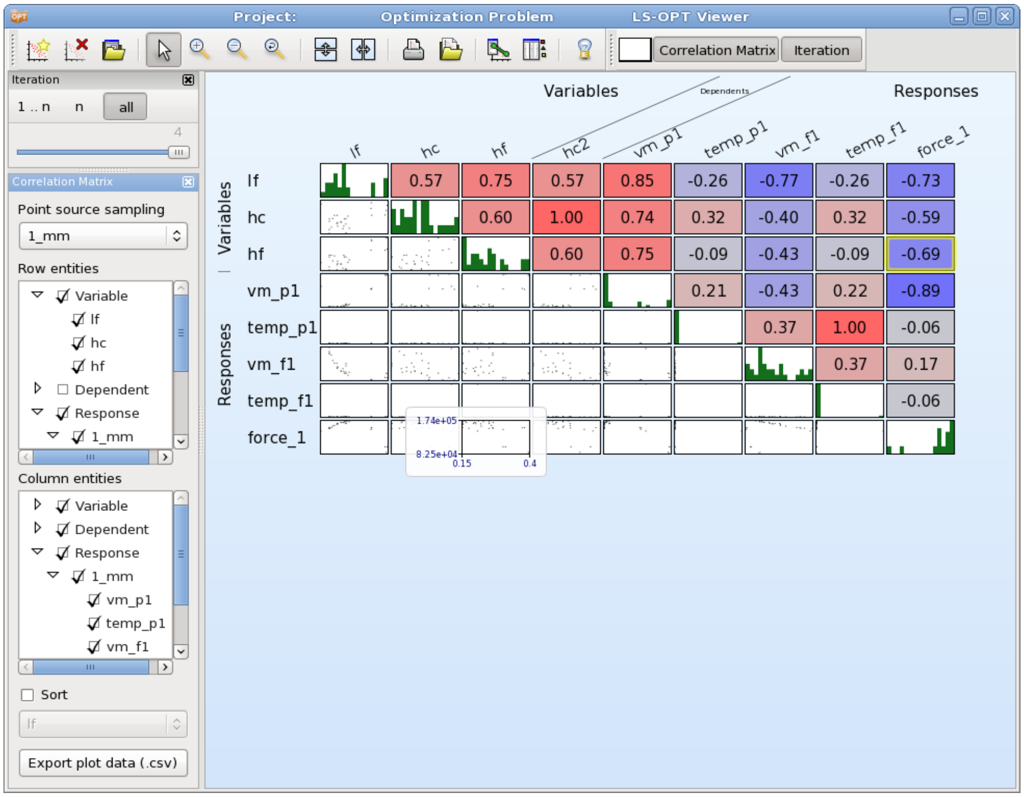
Moving the mouse on a scatter plot displays its ranges and marks the respective correlation coefficient with a yellow border, and vice versa. Row and column entities may be selected separately. Hence it is also possible to display, for example, only correlation coefficients (Figure 16-14).

By double-clicking on a scatter plot or histogram, the respective plot may be reached, see Section 16.2.2 or Section 16.2.6, respectively.

The correlation coefficients are color-coded from blue to red. Blue indicates a strong negative correlation, red a strong positive correlation, whereas grey indicates almost no correlation.

Using the *Sort* option, the order of the rows in the correlation matrix is sorted with respect to the selected column entity values. In the sorted correlation matrix, only correlation values are displayed.

The button *Export plot data* stores the correlation values in a .csv file in the working directory.



Correlation Matrix

Point source sampling: 1_mm

Row entities
- ☑ Variable
  - ☑ lf
  - ☑ hc
  - ☑ hf
- □ Dependent
- ☑ Response
  - ☑ 1_mm

Column entities
- ☑ Variable
- ☑ Dependent
- ☑ Response
  - ☑ 1_mm
    - ☑ vm_p1
    - ☑ temp_p1
    - ☑ vm_f1

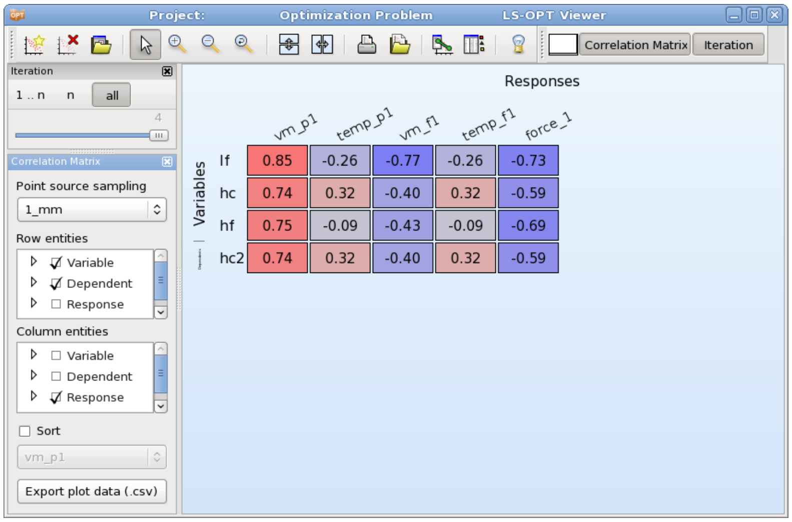
□ Sort

lf

Export plot data (.csv)

	lf	hc	hf	hc2	vm_p1	temp_p1	vm_f1	temp_f1	force_1
lf		0.57	0.75	0.57	0.85	-0.26	-0.77	-0.26	-0.73
hc			0.60	1.00	0.74	0.32	-0.40	0.32	-0.59
hf				0.60	0.75	-0.09	-0.43	-0.09	-0.69
vm_p1						0.21	-0.43	0.22	-0.89
temp_p1							0.37	1.00	-0.06
vm_f1								0.37	0.17
temp_f1									-0.06
force_1									

Figure 16-13: Correlation matrix with scatter plots, histograms and linear correlation coefficient



Point source sampling: 1_mm

Row entities: ☑ Variable ☑ Dependent ☐ Response

Column entities: ☐ Variable ☐ Dependent ☑ Response

☐ Sort

vm_p1

Export plot data (.csv)

Responses

Variables	vm_p1	temp_p1	vm_f1	temp_f1	force_1
lf	0.85	-0.26	-0.77	-0.26	-0.73
hc	0.74	0.32	-0.40	0.32	-0.59
hf	0.75	-0.09	-0.43	-0.09	-0.69
hc2	0.74	0.32	-0.40	0.32	-0.59

Figure 16-14: Correlation matrix, only correlation coefficients

16.2.2. Scatter Plot

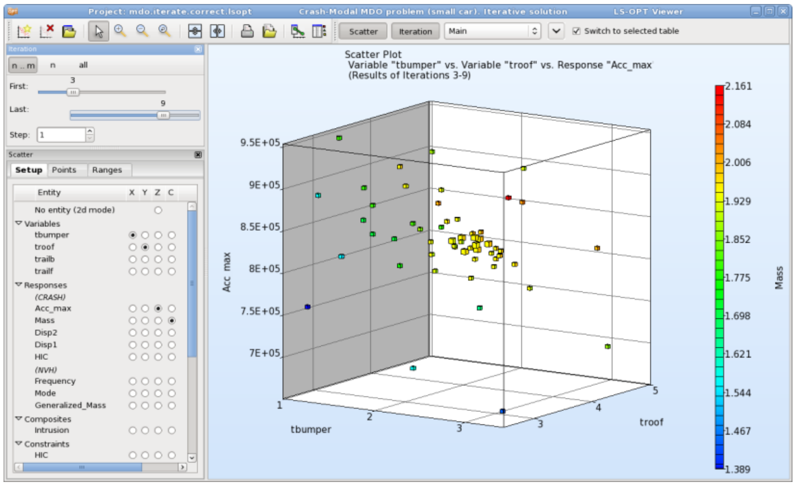
The results of all the simulated points for the selected iterations appear as dots on the scatter plots. This feature allows the three-dimensional plotting of any three entities. A fourth entity may be displayed using the color of the points. Other coloring options are explained below. 2D plots can be obtained by selecting **No entity** for the z axis. For 3D plots, the image rotation is performed by holding down the Ctrl key while moving the mouse (same as LS-PREPOST).

To be able to view the results of composite functions spanning two or more disciplines or stages, the same sampling (Section 27.2) must be selected before starting an analysis. This also implies that the number of variables must be the same for all the disciplines involved and yields coincident experimental designs.

Color Entities – 3D Plots

Table 16-4: Color entity options

Selection	Description
Feasibility	Feasible points are shown in green, infeasible points in red
(with previous in b/w	The points for the current iteration are shown in green (feasible) or red (infeasible). Previous points as light grey (feasible) or dark grey (infeasible)
Iterations	The iteration sequence is shown using a color progression from blue through red.
Neutral	All points are shown in blue
User-defined	User-defined colors and shapes, only available if Point Categories are defined, Section 17.1.5.



First: 3

Last: 9

Step: 1

Entity	X	Y	Z	C
No entity (2d mode)			○	
Variables				
tbumper	●	○	○	○
troof	○	●	○	○
trailb	○	○	○	○
trailf	○	○	○	○
Responses				
(CRASH)				
Acc_max	○	○	●	○
Mass	○	○	○	●
Disp2	○	○	○	○
Disp1	○	○	○	○
HIC	○	○	○	○
(NVH)				
Frequency	○	○	○	○
Mode	○	○	○	○
Generalized_Mass	○	○	○	○
Composites				
Intrusion	○	○	○	○
Constraints				
HIC	○	○	○	○

Figure 16-15: Scatter plot. The 4th dimension is represented by point color.

Points Options

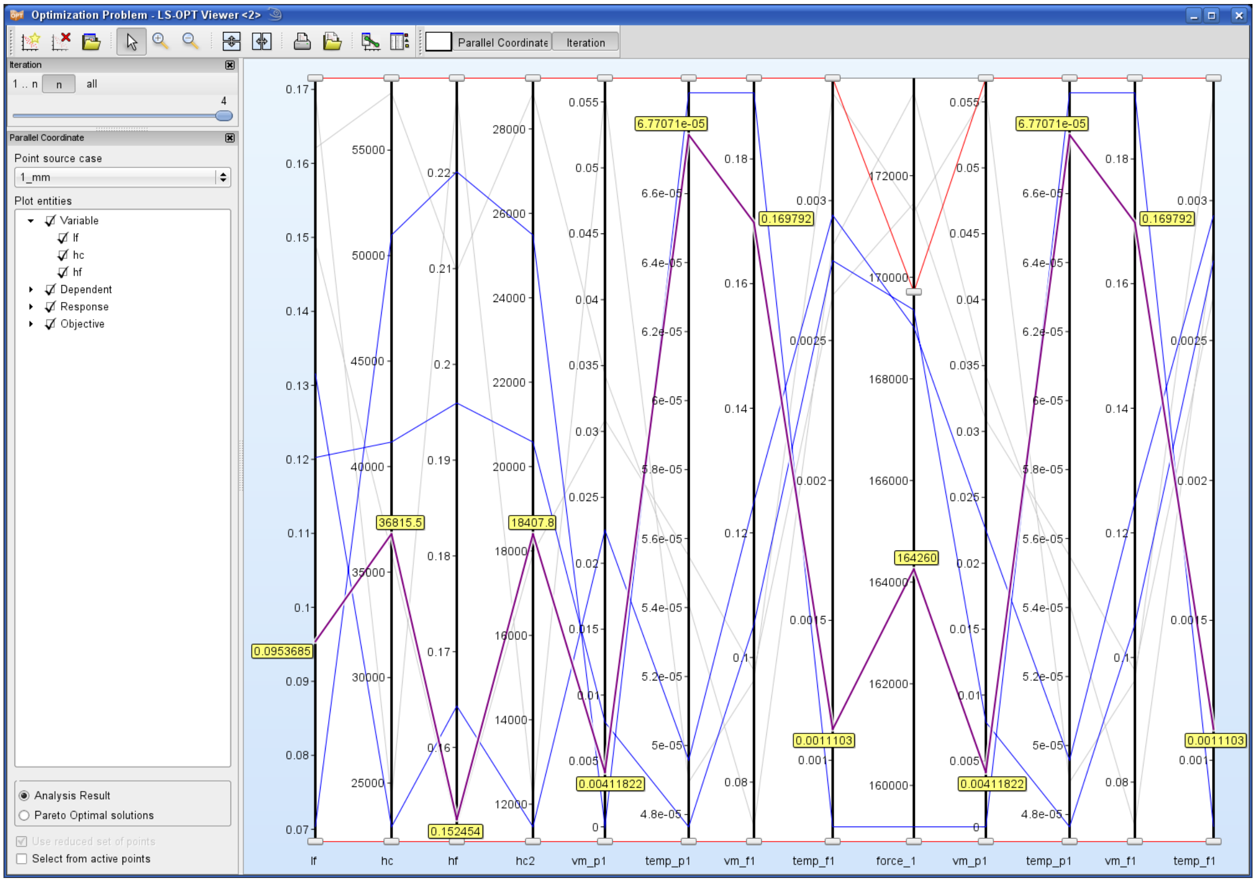
Table 16-5: Point options

Selection	**Description**
Experiments	Plot Experiments (Default if no simulation results are available)
Analysis Results	Plot simulation results (Default for the Scatter Plot)
Feasible	Plot Feasible simulation results (Default for the Scatter Plot)
Infeasible	Plot Infeasible simulation results (Default for the Scatter Plot)
Pareto Optimal Solutions	Plot Pareto optimal solutions (Default for Tradeoff Plot, Section 16.5.1)
Use reduced set of points	Only active for Pareto optimal solutions, plots 100 uniformly distributed points selected from the Pareto optimal solutions
Best computed	Plot the run with the smallest multi-objective value, or in the case where all the runs represent infeasible designs, the run with the smallest constraint violation.
Only show last optimum	Omit best computed points of previous iterations.
User-defined	Plot points that are assigned to selected categories and other selected point options, only available if Point Categories are defined, Section 17.1.5.

Only in GSA subregion	Plot points that satisfy other selected point options and are within the selected GSA subregion in the Ranges tab. Only available if subregions for GSA are defined (Sections 4.11.1 and 16.1.9).

16.2.3. Parallel Coordinate Plot

In contrast to the Scatter Plot, the number of dimensions that can be visualized using the Parallel Coordinate Plot is not restricted. Each dimension is visualized on a vertical axis and each data point is shown as a polyline connecting the respective values on the vertical axis, Figure 16-16. The ranges of the entities may be changed using the sliders at the ends of each vertical axis rendering the points outside the ranges unselectable. Points within the selected ranges are colored in blue, while the remaining points are colored in grey. Selected points are colored in purple, if only a single point is selected, the corresponding value for each entity is displayed in the plot.


Optimization Problem - LS-OPT Viewer <2>

Iteration: 1 .. n, n, all; 4

Parallel Coordinate

Point source case: 1_mm

Plot entities

- ☑ Variable
  - ☑ lf
  - ☑ hc
  - ☑ hf
- ☑ Dependent
- ☑ Response
- ☑ Objective

● Analysis Result
○ Pareto Optimal solutions

☑ Use reduced set of points
□ Select from active points

Figure 16-16: Parallel Coordinate Plot with selected point

Options

Table 16-6: Parallel Coordinate Plot options

Selection	Description
Analysis Results	Plots simulation results (Default)
Pareto Optimal Solutions	Plots Pareto optimal solutions (Section 16.5.2)
Use reduced set of points	Only active for Pareto optimal solutions; plots 100 points selected from the Pareto optimal solutions
Show virtual points	Plot virtual points generated in selected table, see Section 17.1.3.
Select from active points	Selects all points that are not outside the constraints set by the handles, see Section 16.2. Useful for visualizing this set of points in another plot.
Only selected	Plot only selected runs, see 16.2.
User-defined colors	User-defined colors, only available if Point Categories are defined, Section 17.1.5.
User-defined points	Plot points that are assigned to selected categories, only available if Point Categories are defined, Section 17.1.5.

16.2.4. Self-Organizing Maps

The Self-Organizing Maps plot in the Simulations category functions similar to the Self-Organizing Maps Plot described in Section 16.5.4, but here, the default setting is to visualize Analysis results.

16.2.5. History Plot

This plot visualizes history curves based on time data or crossplots obtained from simulations, Figure 16-17. If a variable is selected as y entity, a 3D plot will be displayed. The coloring options are the same as the point coloring options, see 16.2.2. If histories from files are defined in the optimization problem, they can be visualized in addition to the simulation curves, Figure 16-18. The **Multi** option enables plotting of multiple histories in the same plot.

Other History options are explained in Table 16-7.

History statistics may also be displayed, Table 16-8 and Table 16-9.

The **Predicted Histories** option and **Statistics of residuals** are explained in Section 16.3.5.

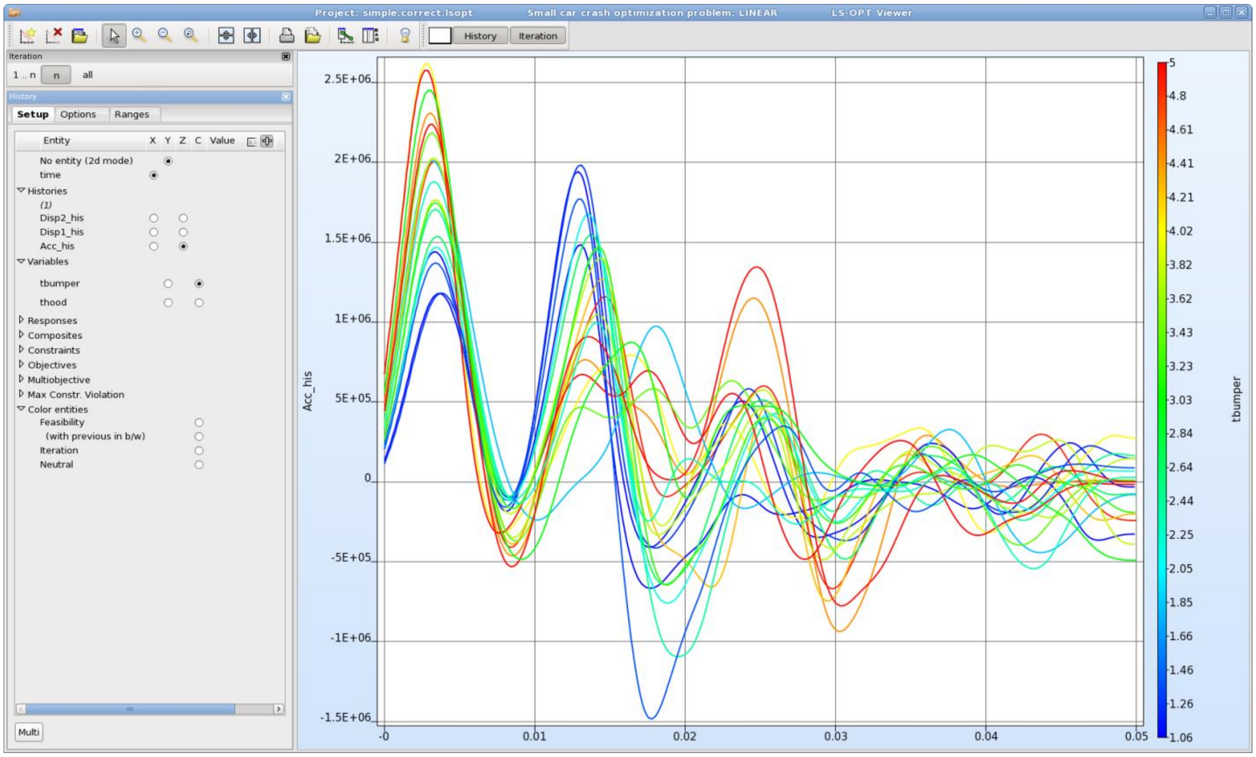


Figure 16-17: History Plot, curves colored by variable

Options

Table 16-7: History Plot options – Histories from Experiments

Selection	Description
Feasible	Plot feasible runs
Infeasible	Plot infeasible runs
Only optimal	Plot the optimal runs of the selected iterations
Only selected	Plot only selected runs, see Section 17.1.1. In this case, the selected curves are not highlighted
Only best computed	Plot the run with the smallest multiobjective and constraint violation, respectively
Points	Plot the discrete history points in addition to the interpolation line

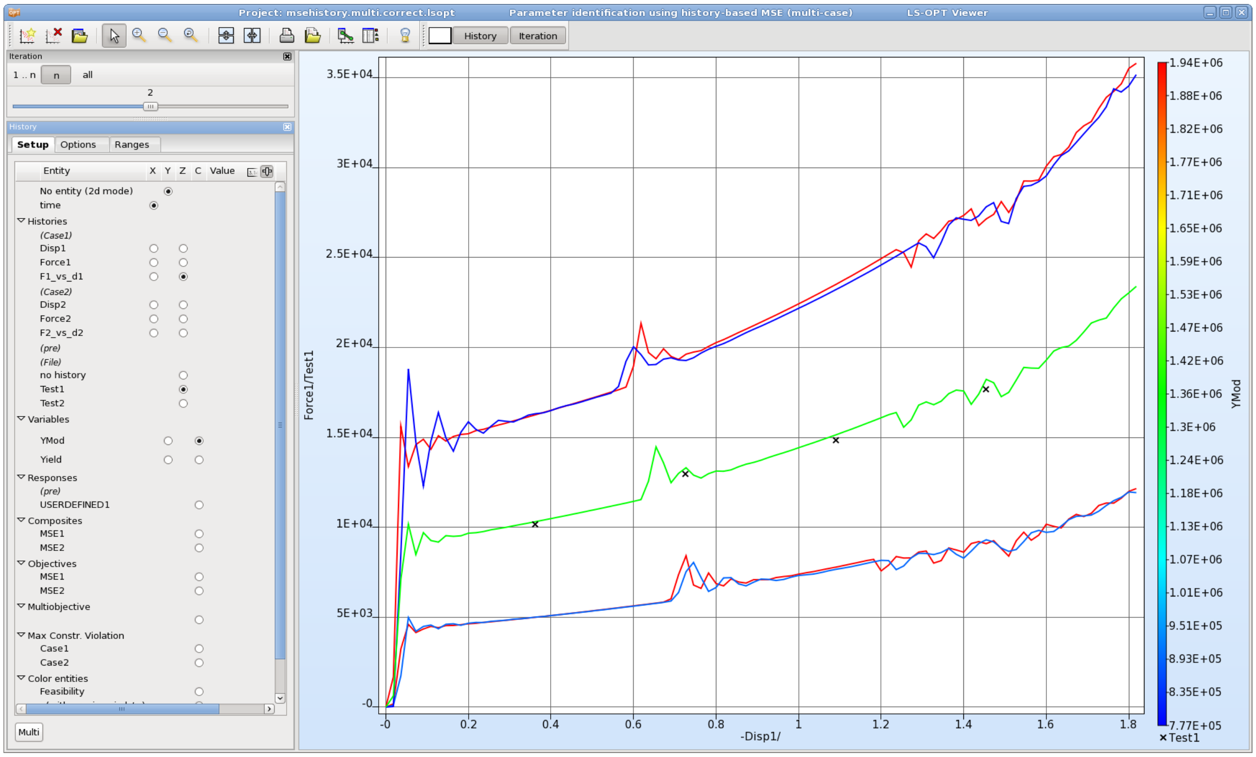


Figure 16-18: Histories from simulations colored by variable with target curve (File history)

Table 16-8: History Plot options - Statistics

Selection	**Description**
Use Metamodels and Distributions	Use metamodels and statistical distribution of the input variables to construct statistics
Mean	Mean values of history values
Standard deviation	Standard deviation of history values
Mean +- Standard deviation	(Mean value + Standard deviation) and (Mean value – Standard deviation) of history values
Max	Maximal values of history values (Mean + two standard deviations if metamodels and distributions are used)
Min	Minimal values of history values (Mean - two standard deviations if metamodels and distributions are used)

Table 16-9: History Plot Options - Advanced Statistics

Selection	**Description**
Range	Range of the history values (maximum values minus the minimum values; four standard deviations if metamodels and distributions are used)
Sample index of Min	ID of the simulation job where the maximum value occurred. This can be used to identify the jobs likely to contain a different bifurcation.
Sample index of Max	ID of the simulation job where the minimum value occurred. This can be used to identify the jobs likely to contain a different bifurcation.
Safety margin	The margin of safety (constraint margin) considering (i) a given bound on the response and (ii) the variation of the response as computed using the Monte Carlo analysis
Lower/Upper bound	Constraint bound
Value	Safety margin value
Scaled with standard deviation	Safety margin value scaled with standard deviation
Probability of failure	Probability of failure
N-th% confidence interval	N-th% confidence interval of probability of failure. Default is 95%.

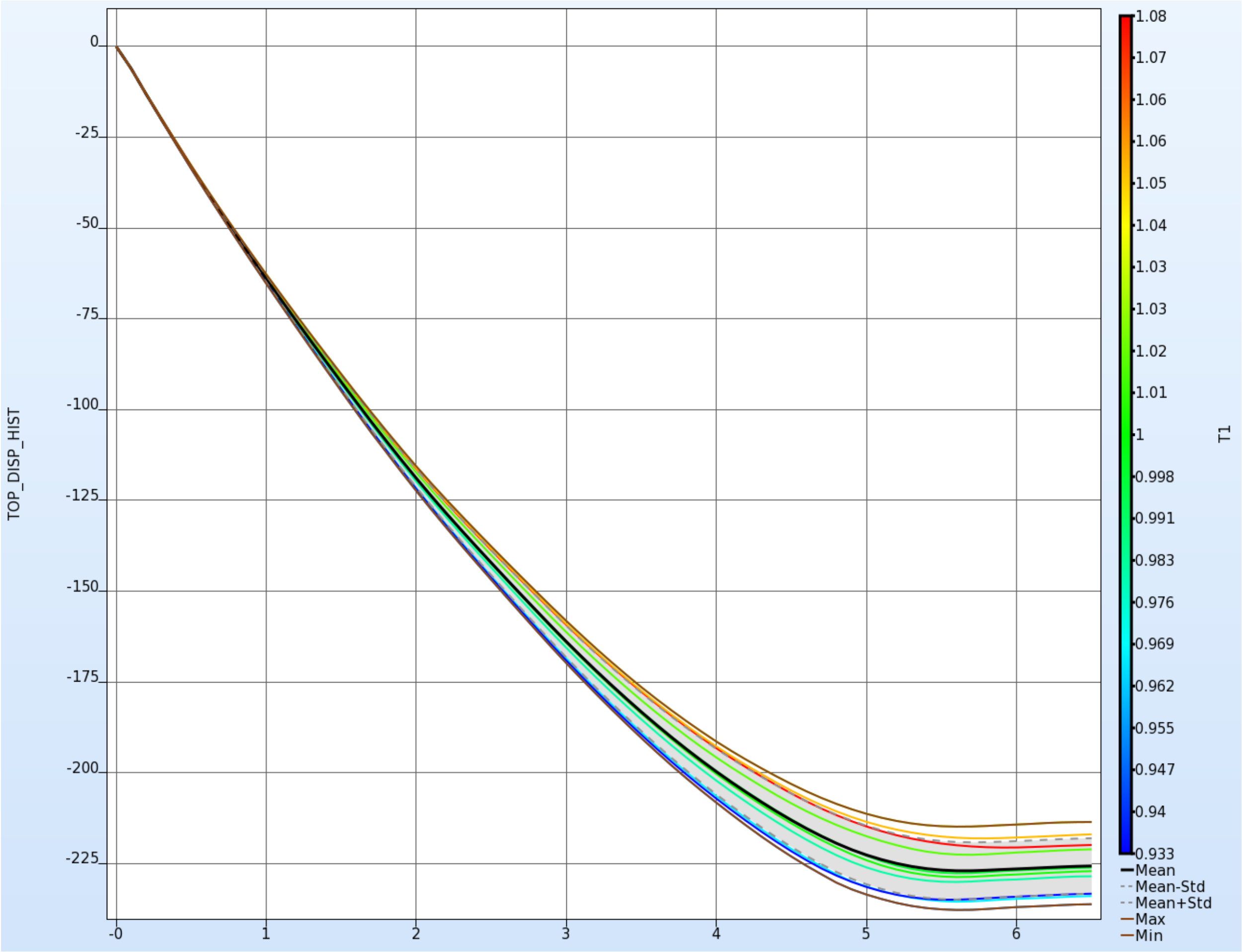


Figure 16-19: Histories from Experiments and selected Statistics

16.2.6. Statistical Tools

The Statistical Tools option offers multiple types of plots - Histogram, Statistics Summary, Probability of constraint violation, Correlation and Covariance. For Sampling type Orthogonal Array, main effect and interaction effect plots are available.

The feature enables display of statistical measures based on either the simulation results directly or based on metamodel approximations. The simulation results are read from the ExtendedResults file of the relevant sampling. If the use of the metamodels is selected, then a Monte Carlo simulation (MCS) is performed to calculate the statistics. The MCS points are generated using a Latin Hypercube experimental design, based on the statistical distributions of the variables. The user can control the number of points in this Monte Carlo simulation in the viewer, as mentioned among the available options in Table 16-10. A large number of MCS points can be used, as only inexpensive metamodel calculations are done at these. If desired, the residuals of the metamodel fit can be added to results of the Monte Carlo simulation as a normal distribution.

For optimization results, iteration can be selected, while for probabilistic evaluations the default iteration, iteration 1, will automatically be selected.

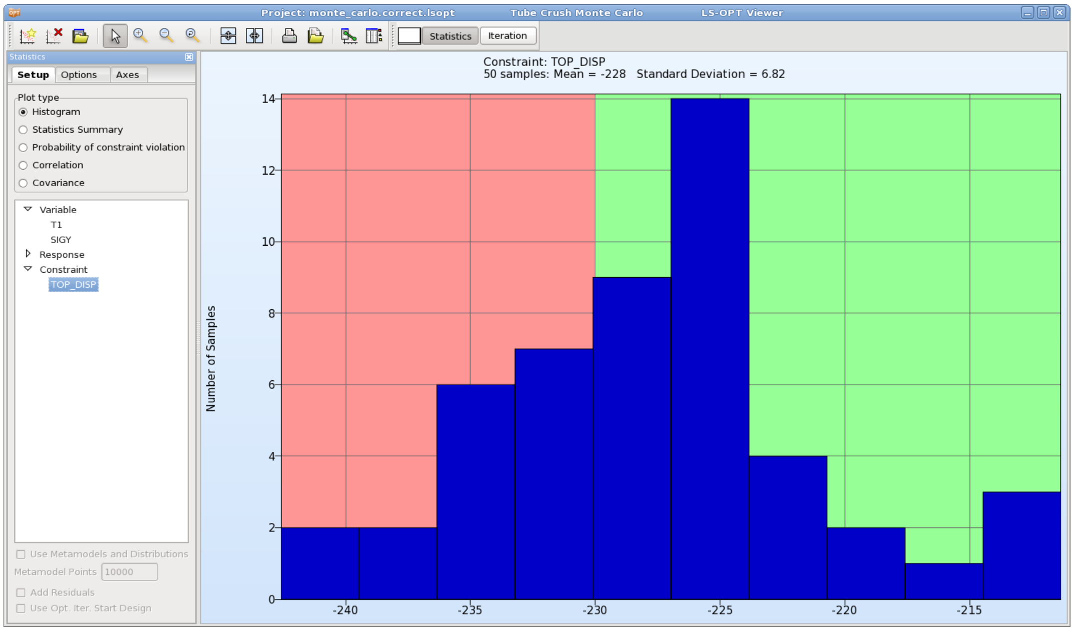
General Options

Table 16-10: General options

Selection	Description
Use Metamodels and Distributions	Use metamodels and statistical distribution of the input variables to construct statistics
Metamodel Points	Number of points used for Monte Carlo Simulation on the metamodel to construct statistics
Add Residuals	Add residuals of the metamodel fit (“noise”) to the results of the Monte Carlo simulation as a normal distribution
Use Opt. Iter. Start Design	Display statistics using the starting design for selected iteration as the mean. If not checked, by default, the optimum solution of the iteration, i.e., starting point of next iteration is used.

Histogram

Histograms of the variables, dependents, responses, composites, constraints and objectives are available. Three types of histograms are available – frequency (*Frequency*), relative frequency (*Probability*) and relative frequency per unit class width (*PDF*). Table 16-11 describes the available options. The histogram panel is shown in Figure 16-20.



Project: monte_carlo.correct.lsopt Tube Crush Monte Carlo LS-OPT Viewer

Plot type
● Histogram
○ Statistics Summary
○ Probability of constraint violation
○ Correlation
○ Covariance

□ Use Metamodels and Distributions
Metamodel Points 10000
□ Add Residuals
□ Use Opt. Iter. Start Design

Constraint: TOP_DISP
50 samples: Mean = -228 Standard Deviation = 6.82

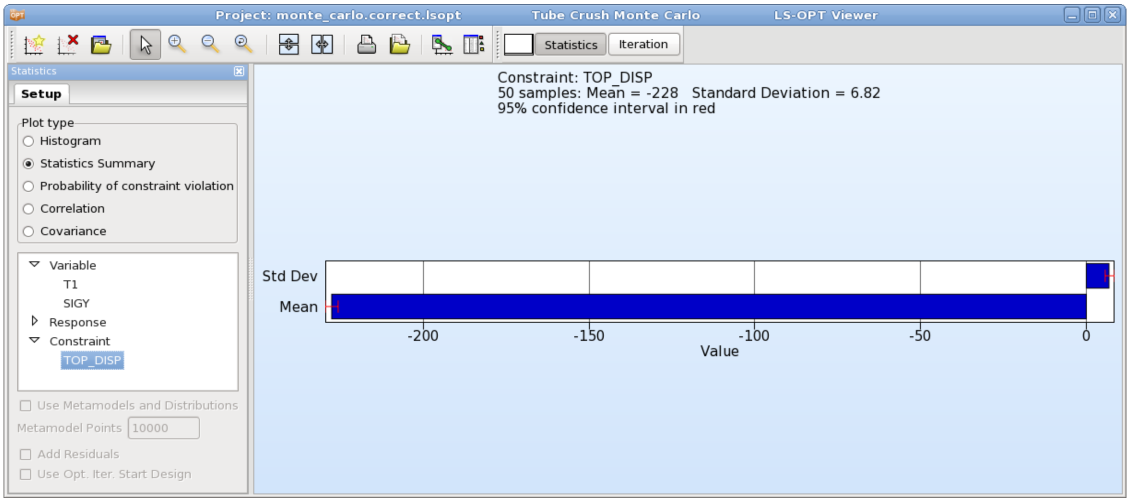
Figure 16-20: Histogram constructed from simulation results. Feasibility information is shown using different background colors (green for feasible and red for infeasible).

Table 16-11: Histogram options

Selection	Description	
Y-axis scaling	Frequency	Number of samples
	Probability	Relative frequency ((Number of samples)/(total number of samples))
	PDF	Relative frequency per unit class interval width ((Number of samples)/(total number of samples)/(class width))
Number of Bars	The number of histogram bars may be specified by the user.	
Mean value	The mean value is displayed as a thick line.	
Standard deviation	The standard deviation is displayed using two lines (mean ± std).	
Median	The median is displayed as a thick line.	
Kernel density estimation	Only Y-axis scaling PDF. An estimation of the probability density function of the plotted entity is displayed.	
Constraints	Feasibility	Color the background of the plot by feasibility
	Value	Display constraint bounds as lines
	Upper Bound	Define upper bound for entity
	Lower Bound	Define lower bound for entity
Box plot	Show box plot below histogram, left and right end of the box are first and third quartile. A tooltip or clicking on the plot visualizes the median and first and third quartile values. Several whisker types are available:	
	min/max	Minimum and maximum of all data
	Interquartile range	the lowest datum still within 1.5 IQR of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile
	Standard deviation	Mean ± standard deviation
	9%/91%	9th percentile and 91st percentile

Statistics Summary

Here, the standard deviation and the mean value for the selected variable, dependent, response or composite are visualized with bars. The confidence intervals are shown in red (Figure 16-21). The confidence level may be selected in the *Options* tab, default is 95%.



Plot type
○ Histogram
● Statistics Summary
○ Probability of constraint violation
○ Correlation
○ Covariance

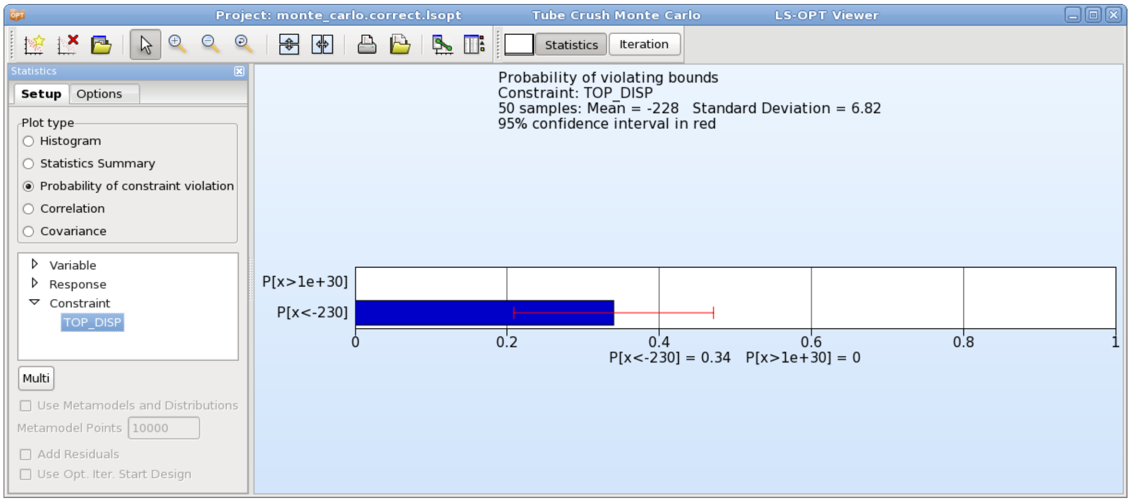
□ Use Metamodels and Distributions
□ Add Residuals
□ Use Opt. Iter. Start Design

Constraint: TOP_DISP
50 samples: Mean = -228 Standard Deviation = 6.82
95% confidence interval in red

Figure 16-21: Standard deviation and mean value of selected response constructed from simulation results

Probability of constraint violation

The user may specify lower and upper bounds, respectively, for the selected variable, dependent, response or composite in the *Options* tab. The probabilities that the entity violates the bounds are visualized using the bars. The confidence intervals are shown in red (Figure 16-22). The confidence level may be selected in the *Options* tab, default is 95%.



Plot type
○ Histogram
○ Statistics Summary
● Probability of constraint violation
○ Correlation
○ Covariance

□ Use Metamodels and Distributions
□ Add Residuals
□ Use Opt. Iter. Start Design

Probability of violating bounds
Constraint: TOP_DISP
50 samples: Mean = -228 Standard Deviation = 6.82
95% confidence interval in red

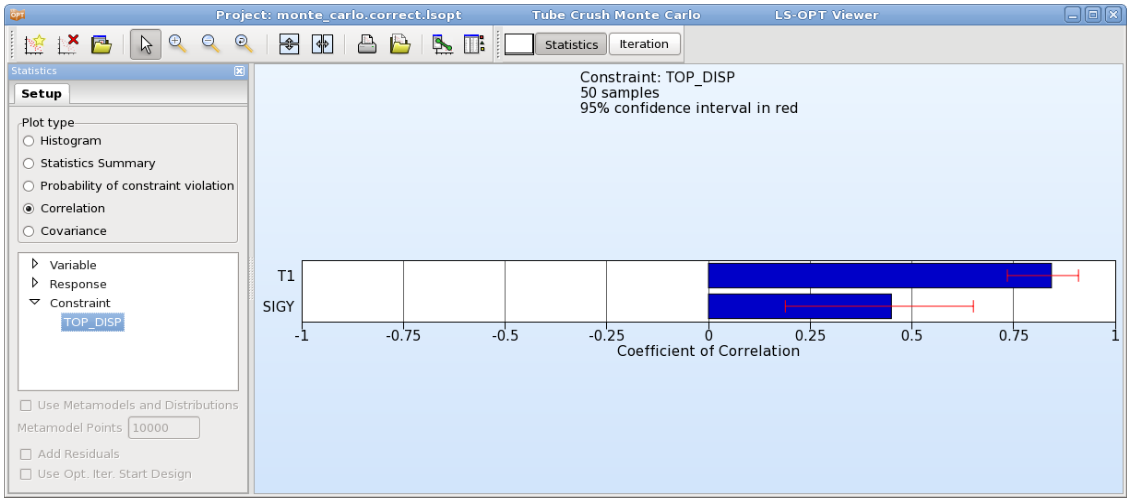
P[x<-230] = 0.34 P[x>1e+30] = 0

Figure 16-22: Probability of TOP_DISP < -230 (i.e. probability of violating the lower bound of -230) with 95% confidence interval shown in red, constructed from simulation results

By selecting the **Multi** button, values for lower and upper bounds can be specified directly in the entity selection list and multiple entities can be selected. The plot displays the probability of violating a lower bound and an upper bound, respectively, for all selected entities. Additionally, the combined effects of multiple constraints are displayed: the probability of violating any lower bound, any upper bound, and any constraint.

Correlation Bars

The coefficients of correlation of the responses and composites with respect to the design variables can be displayed, along with their confidence limits (Figure 16-23). The confidence level may be selected in the *Options* tab, default is 95%. Either the simulated points or the metamodels, together with the statistical distribution of the variables, can be used. If a metamodel is used then a Monte Carlo simulation using a Latin Hypercube experimental design and the statistical distributions of the variables will be conducted on the metamodel to obtain the desired results. The plot can be used to estimate the stochastic contribution of an analysis without a metamodel.



Plot type
○ Histogram
○ Statistics Summary
○ Probability of constraint violation
● Correlation
○ Covariance

▷ Variable
▷ Response
▽ Constraint
TOP_DISP

□ Use Metamodels and Distributions
Metamodel Points 10000
□ Add Residuals
□ Use Opt. Iter. Start Design

Constraint: TOP_DISP
50 samples
95% confidence interval in red

Figure 16-23: Coefficient of Correlation plot with 95% confidence interval in red

Covariance

The covariance of the responses and composites with respect to the design variables can be displayed (Figure 16-24). The plot is very similar to the correlation plot, and can be used to estimate the stochastic contribution of an analysis without a metamodel.

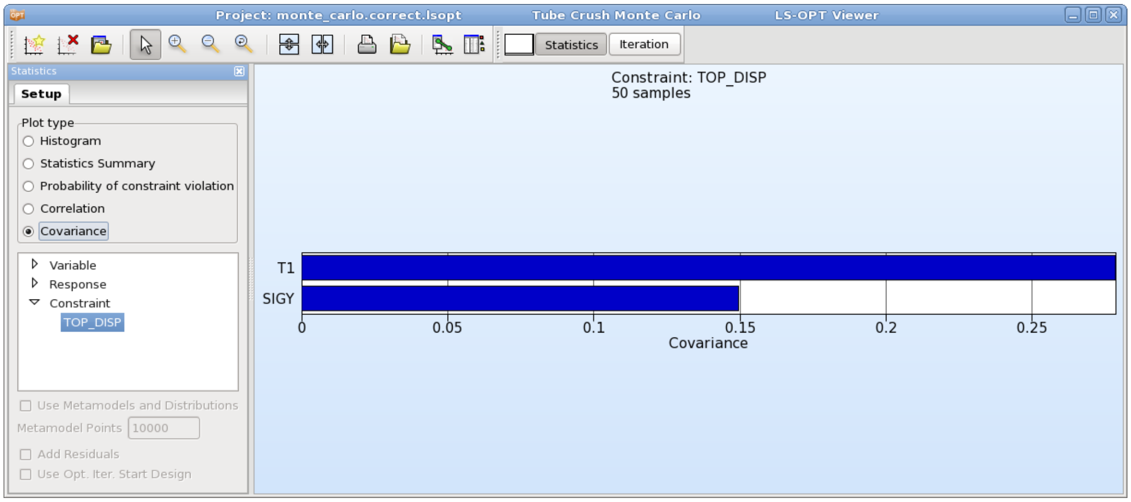


Figure 16-24: Covariance plot

Orthongonal Array: Main and Interaction effect

When an Orthogonal Array sampling is used, it is possible to observe the variable main and interaction effect on result. These plots are available in the Statistical Tools of the Simulation plots.

The main effect plot displays the effect analysis results for each variable value. It is possible to observe the effect of several variables and several responses.

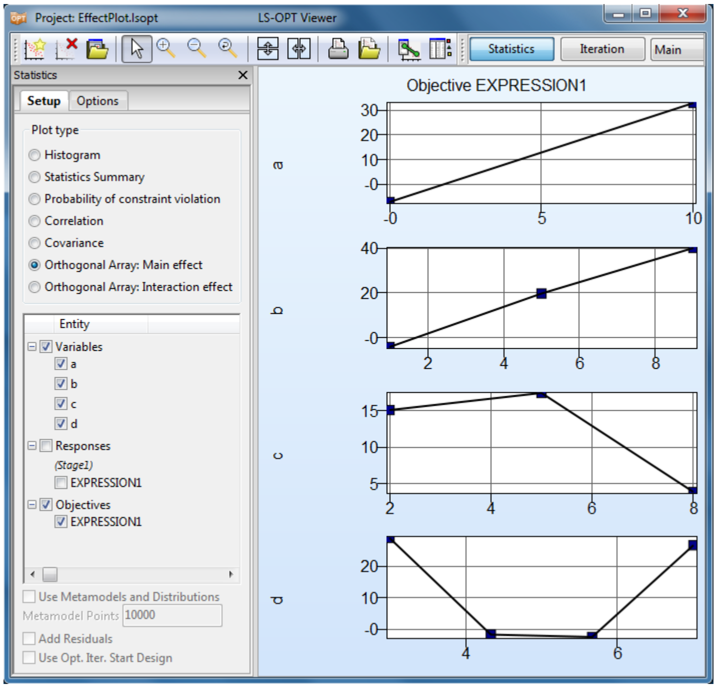


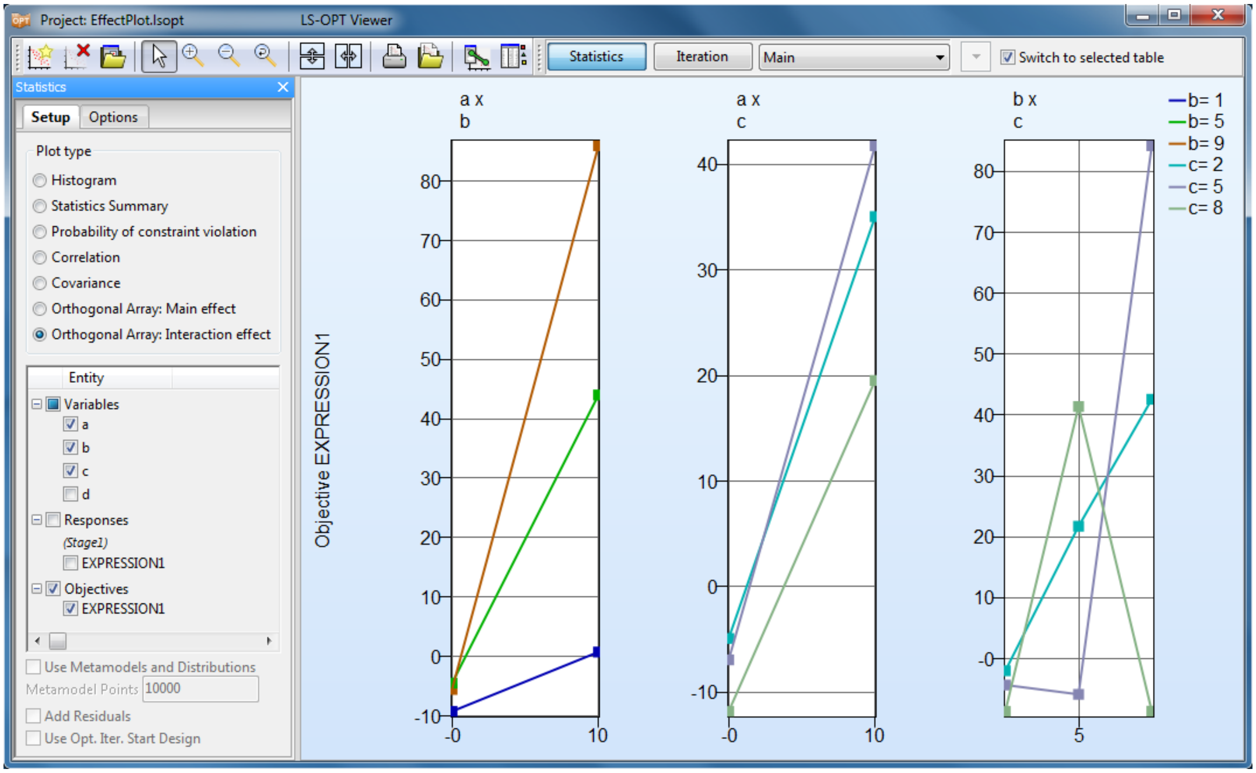
Figure 16-25: Main effect plot

For main or interaction effect plots, you can choose either to plot the means of the responses or the smaller-the-better S/N ratio or larger-the-better S/N ratio (see Section 23.2.7 for Results analysis description).

Table 16-12: Main and interaction effect Plot options

Selection	Description
Plot of Mean Responses	Plot the mean result for the different variable values.
Larger the better S/N ratio	Plot the log of squared result S/N ratio.
Smaller the better S/N ratio	Plot the log of inverted squared result S/N ratio.

The interaction effect plot displays the pairwise effect analysis results for each set of two variables.



Plot type

○ Histogram
○ Statistics Summary
○ Probability of constraint violation
○ Correlation
○ Covariance
○ Orthogonal Array: Main effect
● Orthogonal Array: Interaction effect

Entity

⊟ ■ Variables
- ☑ a
- ☑ b
- ☑ c
- □ d

⊟ □ Responses
- *(Stage1)*
- □ EXPRESSION1

⊟ ☑ Objectives
- ☑ EXPRESSION1

□ Use Metamodels and Distributions
Metamodel Points 10000
□ Add Residuals
□ Use Opt. Iter. Start Design

Figure 16-26: Interaction effect plot

16.3. Visualization of Metamodel Results

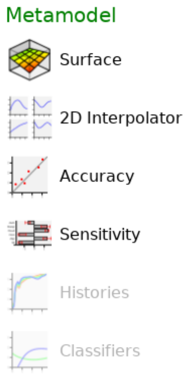


Figure 16-27: Metamodel options

16.3.1. Surface Plot

Two- or three-dimensional cross-sections of the metamodel surfaces and simulation points can be plotted and viewed from arbitrary angles. The image rotation is performed by holding down the Ctrl key while moving the mouse (same as LS-PREPOST). The XY, XZ and YZ buttons at the bottom of the panel rotate the plot to the respective coordinate plane.

The following options are available:

Setup

The selection of one or two variables and the response or composite function is done here. The sliders and textfields, respectively, allow changing the variable values for unselected variables (variables not plotted). Switching from sliders to textfields can be done using the icons in the column heading (3.14 and ⇕). The slider or textfield for the active variables can be activated by selecting the "Predicted Value" option.

Table 16-13: Surface Plot Setup options

Selection	Description
Gridlines	Gridlines are displayed on the surface, Figure 16-28
Isolines	Isolines are displayed on the surface, Figure 16-29
Predicted value	The predicted value for the selected variable values is displayed on the surface, the variable and response values are displayed in the top left corner, Figure 16-28
Variable values	Values of fixed variables and all variables, if *Predicted value* is selected, respectively, are displayed in the plot.

Constraints	Constraints are displayed on the surface, Figure 16-31. Feasible regions are in green, the shade of red shows the degree of infeasibility (number of violated constraints), the colored lines in 3D and the + marks in 2D, respectively show the location where the constraints are exactly met. Tooltips display constraint names and bounds. The *Advanced* button shows a checkbox for each constraint to select which constraints to display.
Classifiers	Classifier boundaries are displayed on the surface. Tooltips display the classifier names.
Surface transparency	Set transparency of surface
Center variable sliders on Optimum	Variable sliders are set to optimal values of selected iteration

Point plotting options

Table 16-14: Surface Plot point plotting options

Selection	**Description**
Feasible	Show feasible runs only
Infeasible	Show infeasible runs only
Predicted Optimum	Show predicted optimum
Computed Optimum	Show computed optimum
Best Computed	Plot the run with the smallest multiobjective and constraint violation, respectively
Only show last optimum	Omit best computed points and predicted and computed optimum of previous iterations.
Pareto Optimal Solutions	Show Pareto optimal solutions
Use reduced set of points	Only active for Pareto optimal solutions, plots 100 uniformly distributed points selected from the Pareto optimal solutions
Failed runs on surface	Failed runs such as error terminations are projected to the surface in grey
Points only	Show only points without surface

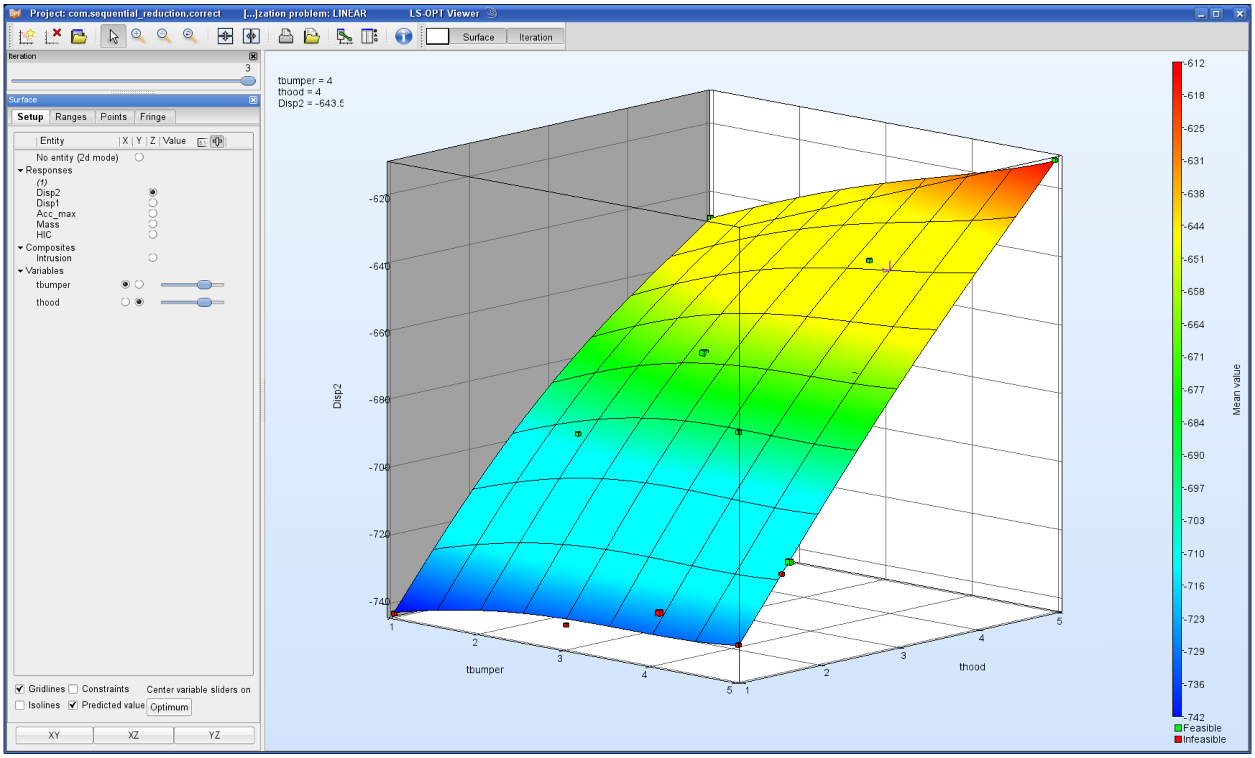
Show virtual points	Plot virtual points generated in the selected table, see Section 17.1.3
Only in GSA subregion	Plot points that satisfy other selected point options and are within the selected GSA subregion in the Ranges tab. Only available if subregions for GSA are defined (Section 4.11.1)
Project points to surface	The points are projected on the surface to improve visibility. Future versions will have a transparency option.
Show Residuals	Shows a black vertical line connecting the computed and predicted values.

Point status

The points are colored according to the selected *Status (colors)* menu option, Table 16-15.

Table 16-15: Surface Plot point status options

Selection	**Description**
Feasibility	Feasible points are shown in green, infeasible points in red
Previous b/w	The points for the current iteration are shown in green (feasible) or red (infeasible). Previous points as light grey (feasible) or dark grey (infeasible)
Iterations	The iteration sequence is shown using a color progression from blue through red. See Figure 16-29.
Optimum runs	Optimal points are shown in green/red and all other points in white.



Project: com.sequential_reduction.correct [...]zation problem: LINEAR LS-OPT Viewer

Iteration: 3

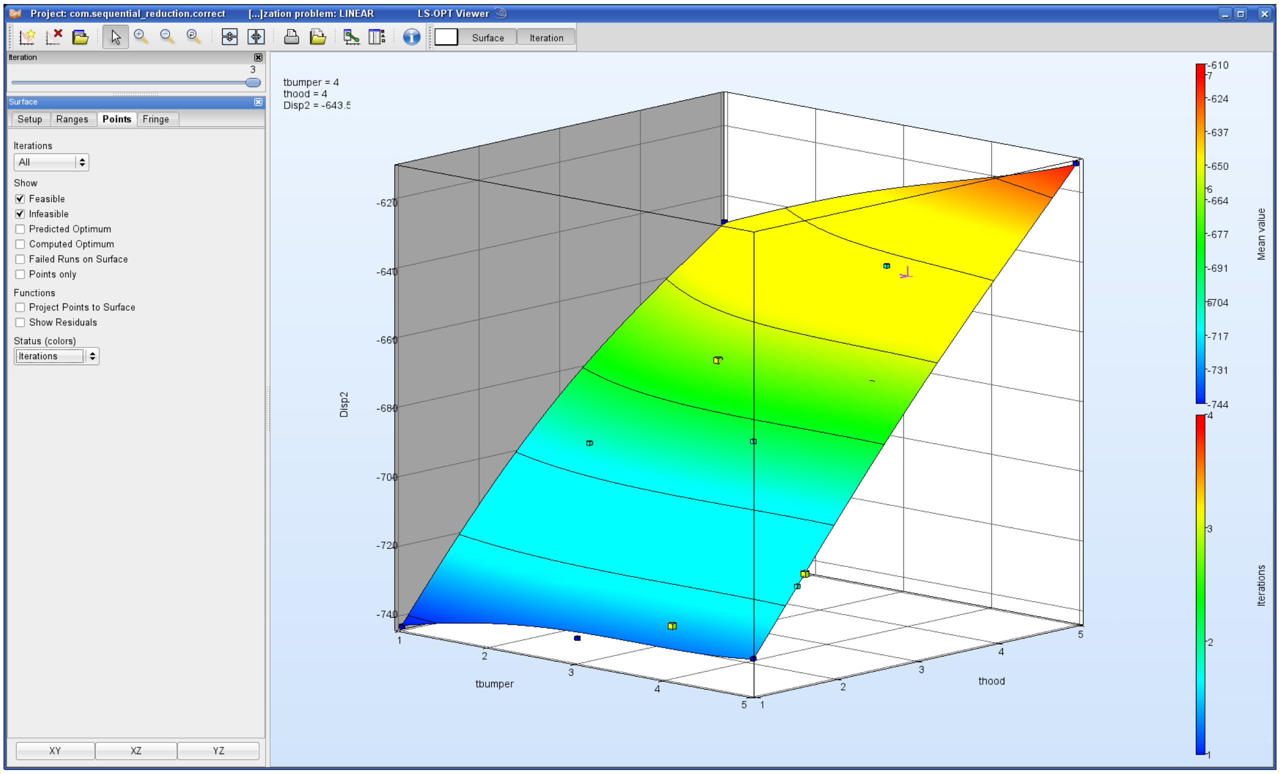
Surface — Setup | Ranges | Points | Fringe

Entity	X	Y	Z	Value
No entity (2d mode)	○			
▾ Responses				
(1)				
Disp2			●	
Disp1			○	
Acc_max			○	
Mass			○	
HIC			○	
▾ Composites				
Intrusion			○	
▾ Variables				
tbumper	●	○		
thood	○	●		

☑ Gridlines ☐ Constraints Center variable sliders on
☐ Isolines ☑ Predicted value Optimum

tbumper = 4
thood = 4
Disp2 = -643.5

Figure 16-28: Metamodel plot showing feasible (green) and infeasible (red) points. The predicted point is shown in violet (t_hood = 4, t_bumper = 4) with the values displayed at the top left.



Project: com.sequential_reduction.correct [...]zation problem: LINEAR LS-OPT Viewer

Iteration: 3

Surface — Setup | Ranges | Points | Fringe

Iterations: All

Show
☑ Feasible
☑ Infeasible
☐ Predicted Optimum
☐ Computed Optimum
☐ Failed Runs on Surface
☐ Points only

Functions
☐ Project Points to Surface
☐ Show Residuals

Status (colors): Iterations

tbumper = 4
thood = 4
Disp2 = -643.5

Figure 16-29: Metamodel plot showing point color coding for iteration numbers.

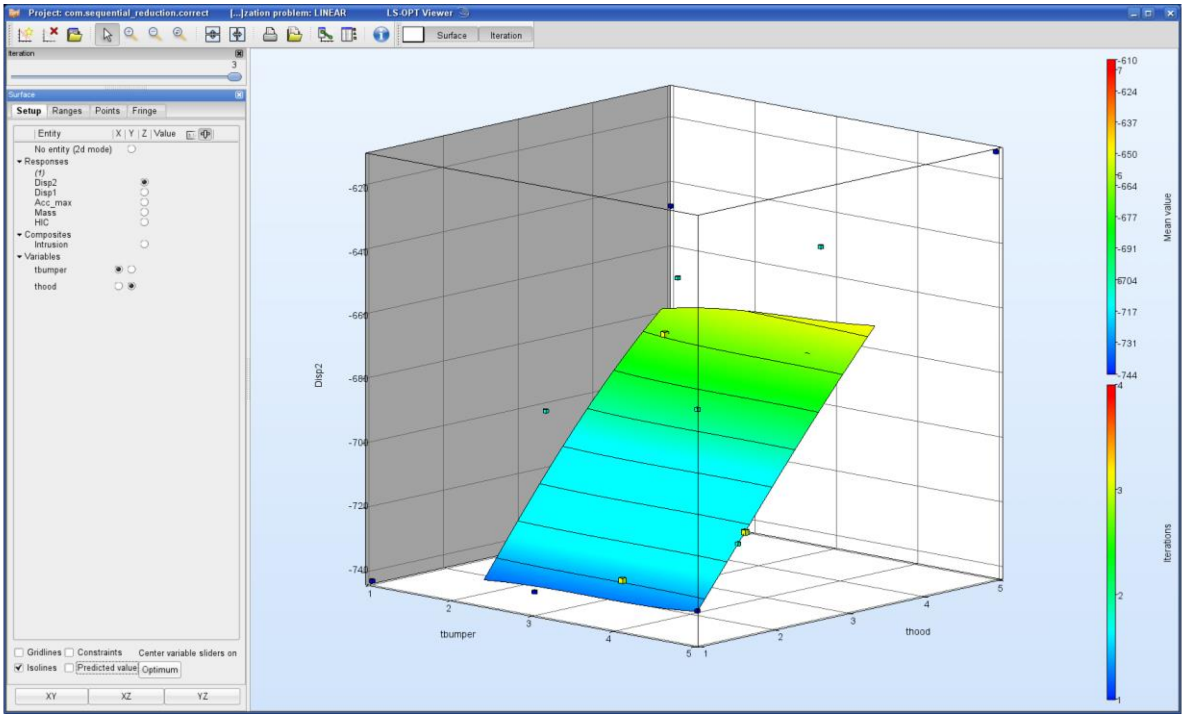


Figure 16-30: Surface plot representing only the region of interest of the fourth iteration.

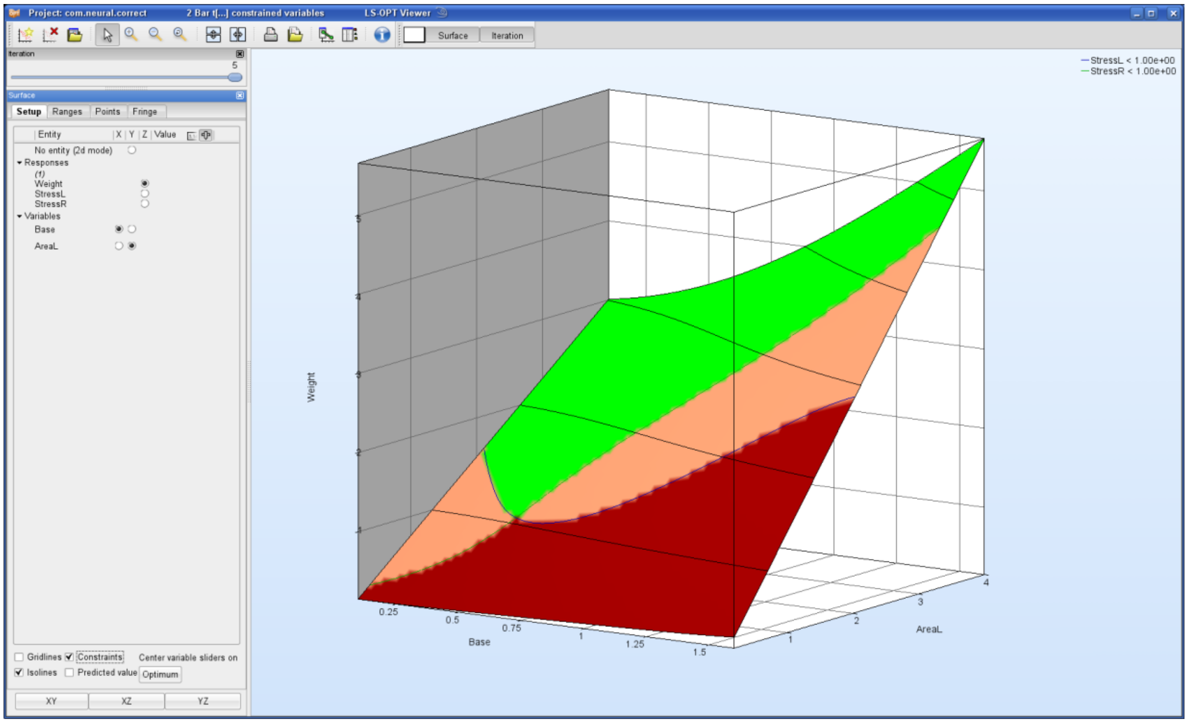
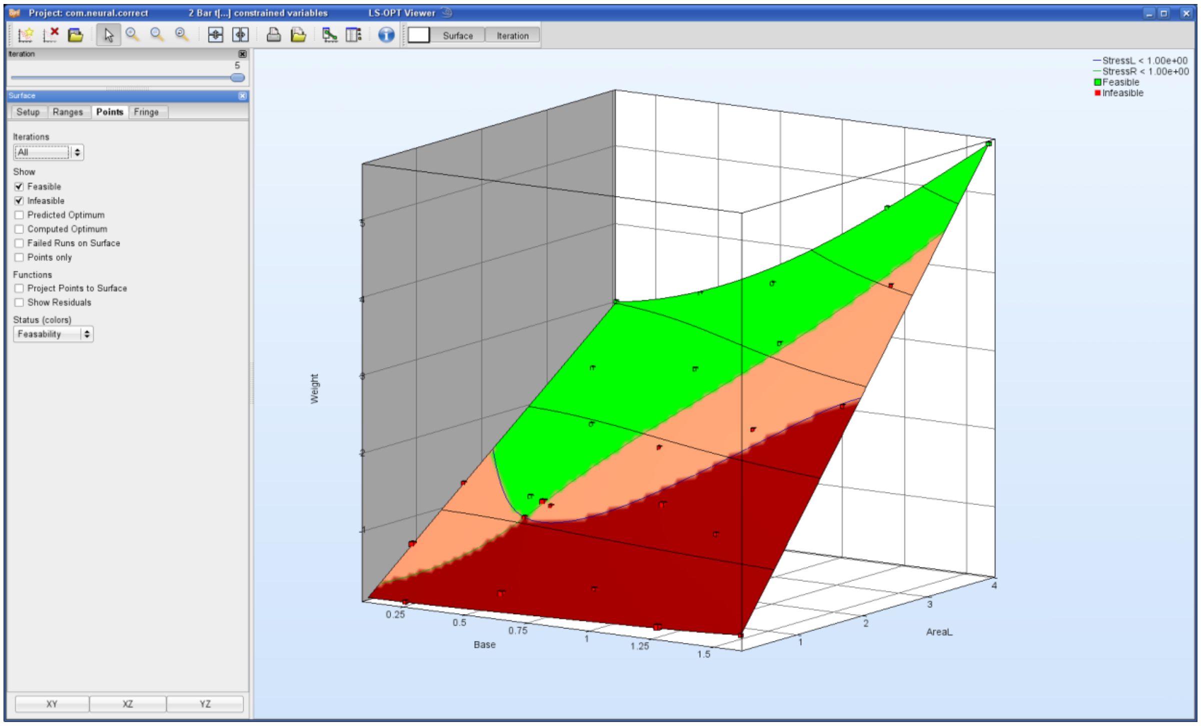


Figure 16-31: Plot showing isolines on the objective function as well as constraint contours and feasibility. Feasible regions are in green. Shade of red shows degree of infeasibility (number of violated constraints). Note the legend describing constraints at the top right. Tooltips displaying the constraint name and bounds are also available for the constraint lines.


Project: com.neural.correct 2 Bar t[...] constrained variables LS-OPT Viewer

Iteration: 5

Surface: Setup | Ranges | **Points** | Fringe

Iterations: All

Show
- ☑ Feasible
- ☑ Infeasible
- □ Predicted Optimum
- □ Computed Optimum
- □ Failed Runs on Surface
- □ Points only

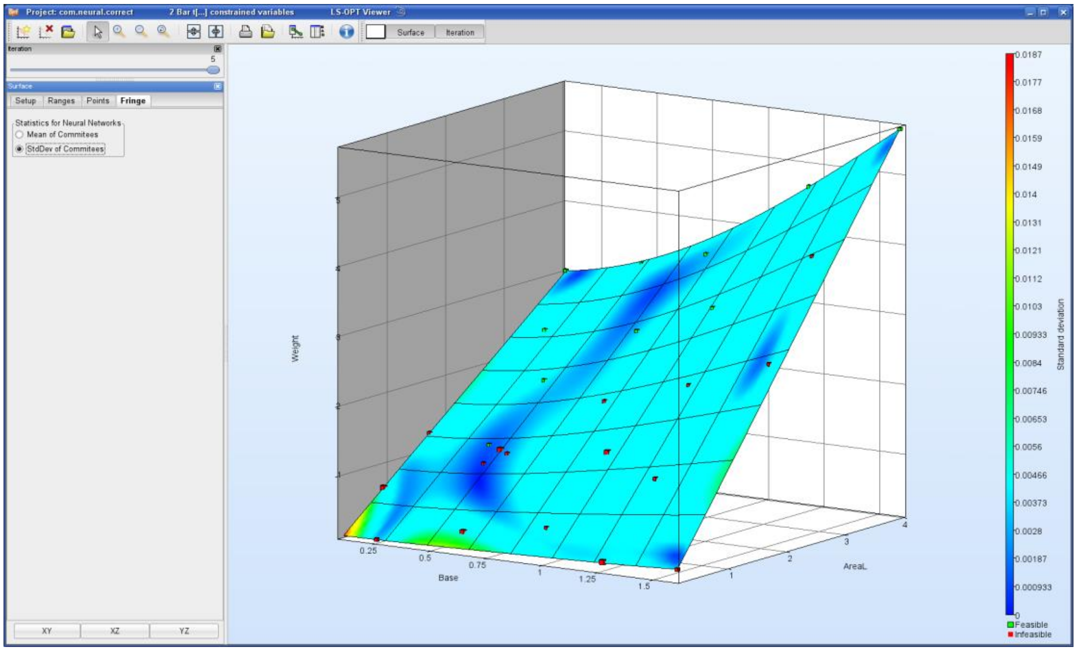
Functions
- □ Project Points to Surface
- □ Show Residuals

Status (colors): Feasability

Figure 16-32: Plot showing isolines and points opposite the "Points" tab.

Fringe plot options for neural nets

The options are function value or standard deviation of the Neural Net committee values and are only available if the metamodel type is Feedforward Neural Network. See Figure 16-33.


Project: com.neural.correct 2 Bar t[...] constrained variables LS-OPT Viewer

Iteration: 5

Surface: Setup | Ranges | Points | **Fringe**

Statistics for Neural Networks
- ○ Mean of Commitees
- ● StdDev of Commitees

Figure 16-33: Metamodel plot showing standard deviation of the Neural Net committee values.

Comparison metamodels

If comparison metamodels have been defined, they can be selected at the top of the surface display control panel. See Figure 16-34. The main metamodel selection is always available at the top of the list.

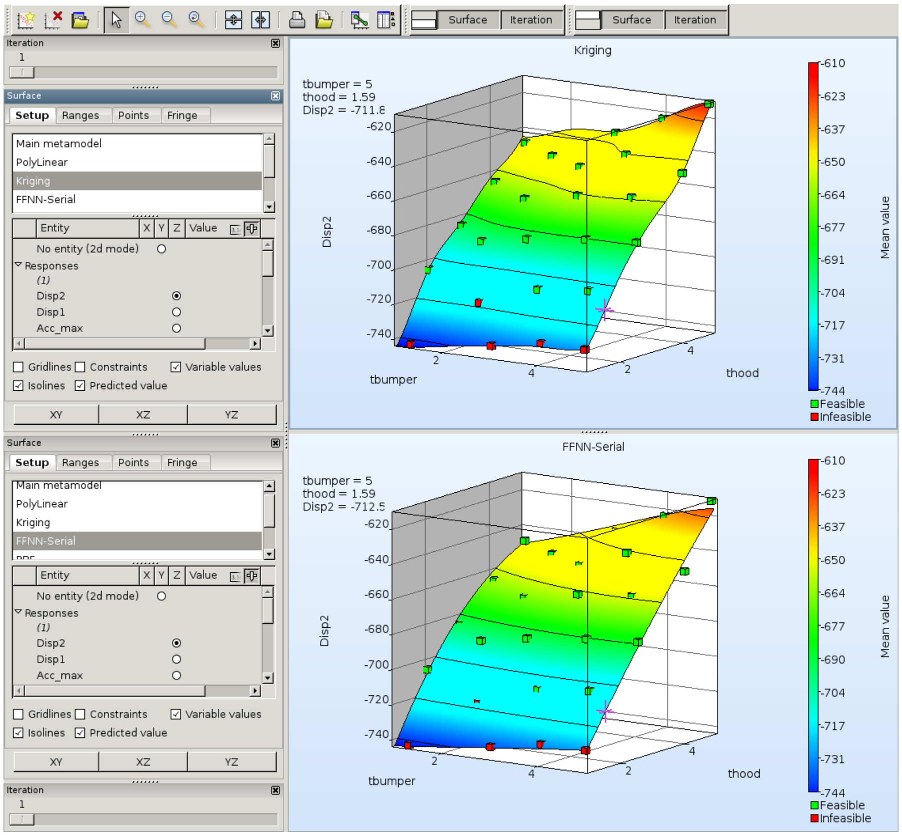


Figure 16-34: Metamodel plot showing two selected comparison metamodels based on the same simulation results. The individual metamodels are selected in the uppermost dialog of the Setup tab.

16.3.2. 2D Interpolator Plot

The Interpolator plot is a tool to display multiple two-dimensional surface plots. All selected responses and composites are plotted against all selected variables. The default is to display each response against all variables in a row.

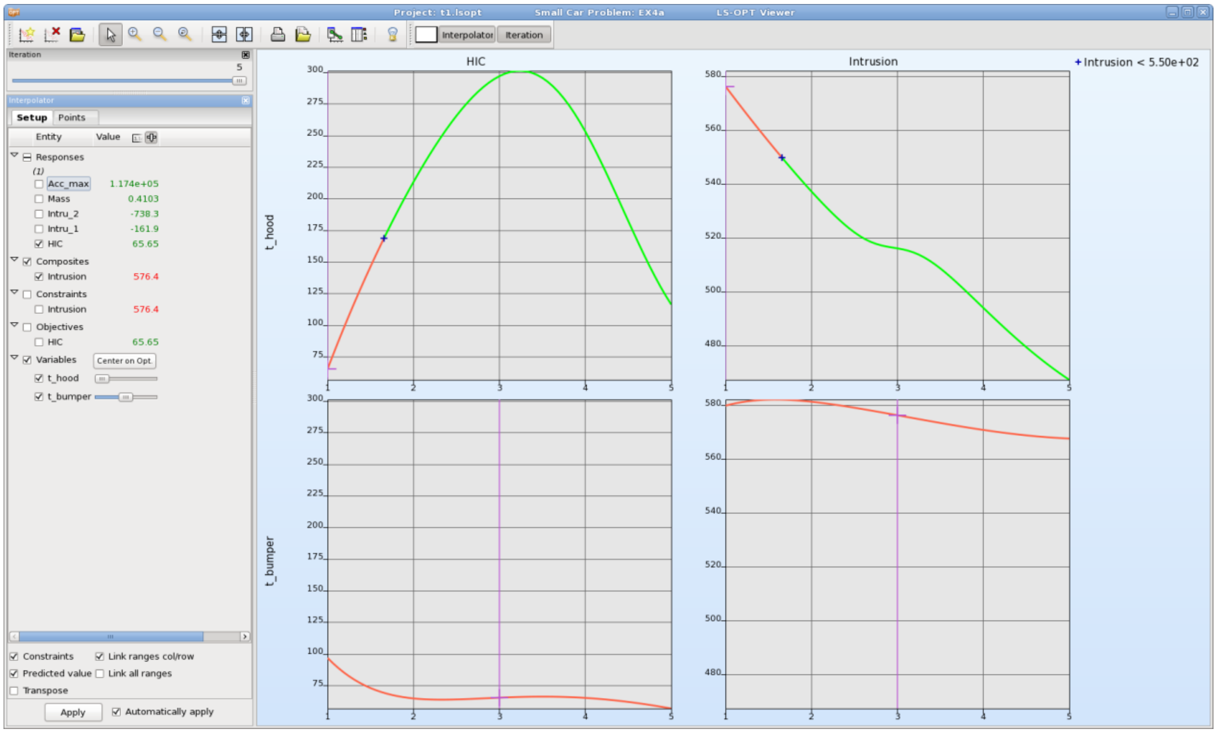
Options

Table 16-16: 2D Interpolator Plot options

Selection	**Description**
Constraints	Constraints are displayed on the surface. Feasible regions are in green, the shade of red shows the degree of infeasibility (number of violated constraints), the

	colored + marks in 2D show the location where the constraints are exactly met.
Predicted value	The predicted value for the selected variable values is displayed on the surface (purple line), the variable and response values are displayed in the panel
Transpose	Allows to display each response against all variables in a column.
Link ranges col/row	The same y range is used for all plots in a column and in a row, respectively, if Transpose is selected. This is the default.
Link all ranges	The same y range is used for all plots.
Center on Opt.	Variable sliders are set to optimal values of selected iteration
Automatically apply	By default, any new selection automatically regenerates the plot. Since this can take time, this can be switched off. Multiple option changes can be done in the panel, and the plot is only regenerated if the *Apply* button is pressed.

For a description of the **Points** options, see Table 16-14 and Table 16-15.



Project: t1.lsopt Small Car Problem: EX4a LS-OPT Viewer

Iteration 5

Interpolator

Setup | Points

Entity	Value
⊟ Responses	
(1)	
□ Acc_max	1.174e+05
□ Mass	0.4103
□ Intru_2	-738.3
□ Intru_1	-161.9
☑ HIC	65.65
☑ Composites	
☑ Intrusion	576.4
□ Constraints	
□ Intrusion	576.4
□ Objectives	
□ HIC	65.65
☑ Variables	Center on Opt.
☑ t_hood	
☑ t_bumper	

☑ Constraints ☑ Link ranges col/row
☑ Predicted value □ Link all ranges
□ Transpose
Apply ☑ Automatically apply

Figure 16-35: Interpolator Plot with constraints (Feasible regions are in green, shade of red shows degree of infeasibility (number of violated constraints)) and predicted value (purple line)

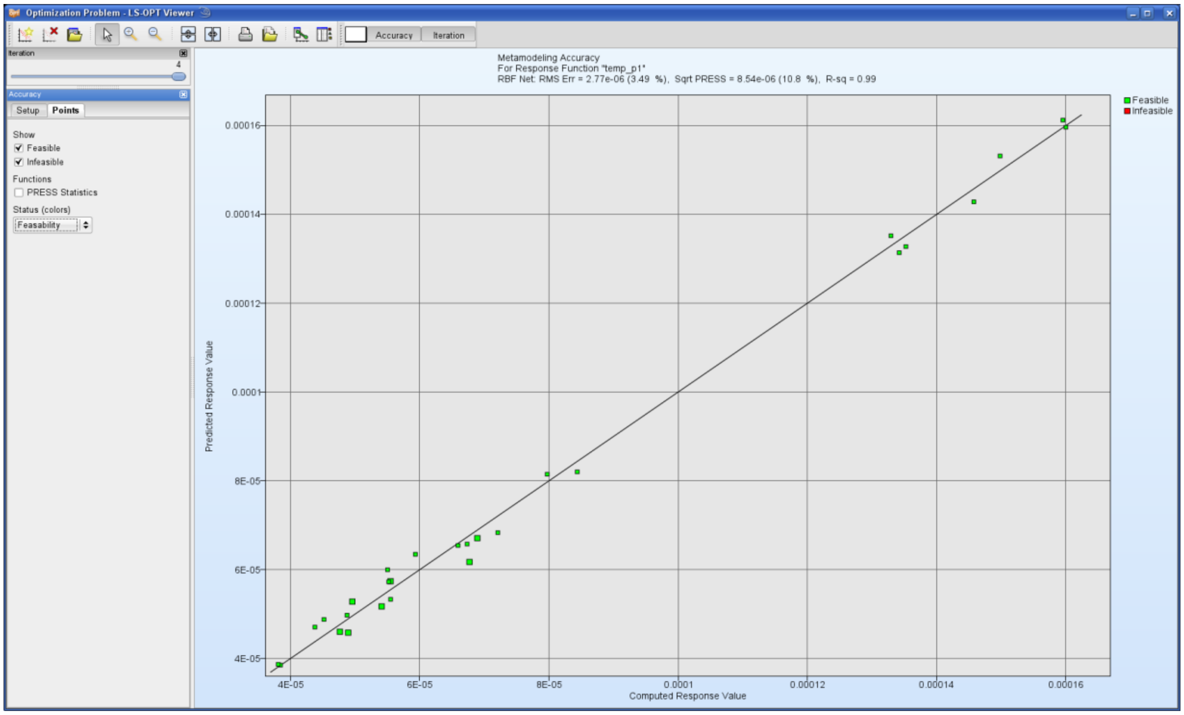
16.3.3. Accuracy Plot

The accuracy of the metamodel fit for the selected response or composite is illustrated in a Predicted vs. Computed plot, Figure 16-36. The results for the metamodel of each iteration are displayed separately using the slider bar. All points used to approximate the metamodel are displayed, i.e., for linear metamodels, the points of the current iteration are displayed, whereas for all other metamodels, the points of all previous iterations are also visualized, Figure 16-36. The error measures are displayed in the heading.

Options

Table 16-17: Accuracy Plot options

Selection	Description
Feasible	Plot feasible runs
Infeasible	Plot infeasible runs
PRESS statistics	PRESS residuals are plotted against computed values, add Reference to Section 23.3.5. The predicted value is evaluated on the metamodel fit to the remaining points.
Status (colors)	Coloring options for points, see 16.2.2



Metamodeling Accuracy
For Response Function "temp_p1"
RBF Net: RMS Err = 2.77e-06 (3.49 %), Sqrt PRESS = 8.54e-06 (10.8 %), R-sq = 0.99

Show
☑ Feasible
☑ Infeasible
Functions
□ PRESS Statistics
Status (colors)
Feasability

Figure 16-36: Computed vs. Predicted plot. The points are color-coded to represent the feasibility. The largest points represent the most recent iteration.

Comparison metamodels

If comparison metamodels have been defined, they can be selected at the top of the accuracy display control panel, see Figure 16-37. The main metamodel selection is always available at the top of the list.

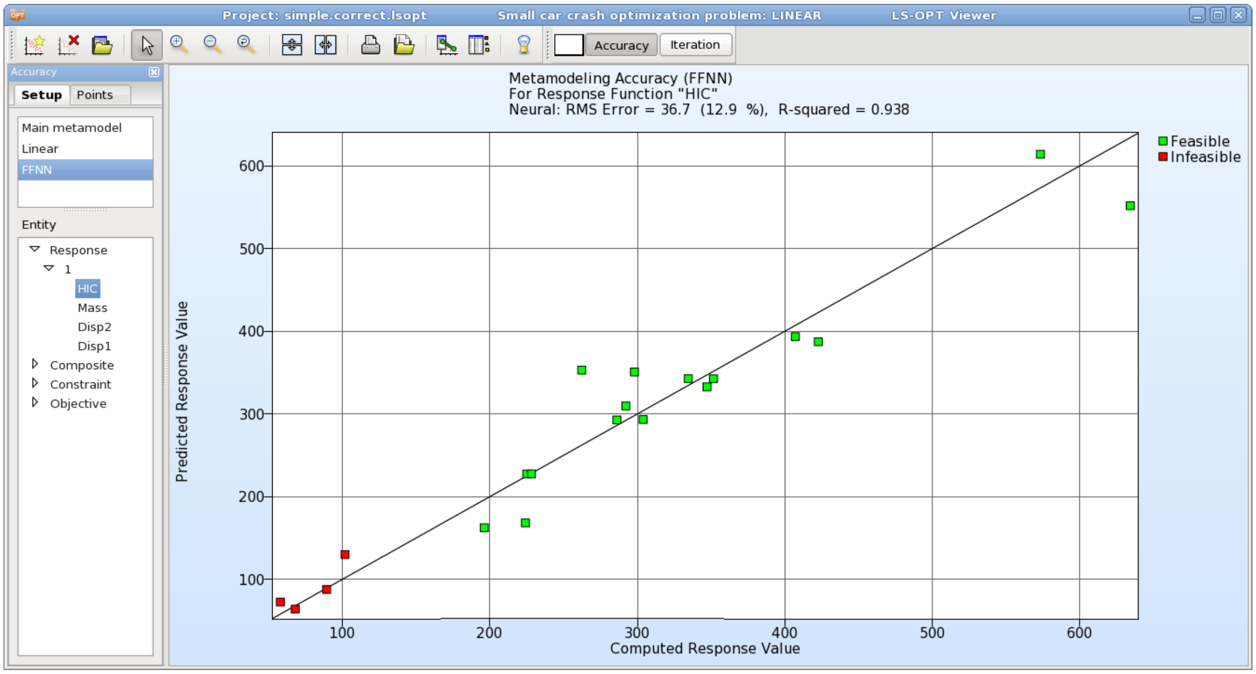


Figure 16-37: Computed vs. Predicted plot for Comparison Metamodel FFNN.

16.3.4. Sensitivities

The Sensitivities Plot provides visualization of the results of ANOVA and global sensitivity analysis (GSA) using Sobol's variance-based sensitivity indices.

Linear ANOVA

The Analysis of Variance (ANOVA) (refer to Section 23.4) of the approximation to the experimental design is automatically performed if a polynomial response surface method is selected. In the case of other approximation types, a linear approximation is also constructed to generate ANOVA information. The ANOVA information can be used to screen variables (remove insignificant variables) at the start of or during the optimization process. The ANOVA method, a more sophisticated version of what is sometimes termed 'Sensitivities' or 'DOE', determines the significance of main and interaction effects through a partial *F*-test (equivalent to Student's *t*-test) [1]. This screening is especially useful to reduce the number of design variables for different disciplines (see Sections 27.2 (theory) and 20.5 (example)).

If a probabilistic or an RBDO analysis is being done, then the Stochastic Contribution plots (see Section 16.6.2) are recommended.

The ANOVA results are viewed in bar/tornado chart format, Figure 16-38. The **Sort** option sorts the ANOVA values by relevance, the sorting doesn't consider the 95% confidence interval.

Using the *Export plot data (.csv)* option, the ANOVA results of the selected response can be stored in a .csv file.

Clicking on the chart displays the respective derivative values (scaled with variable range) and confidence interval bounds in the plot.

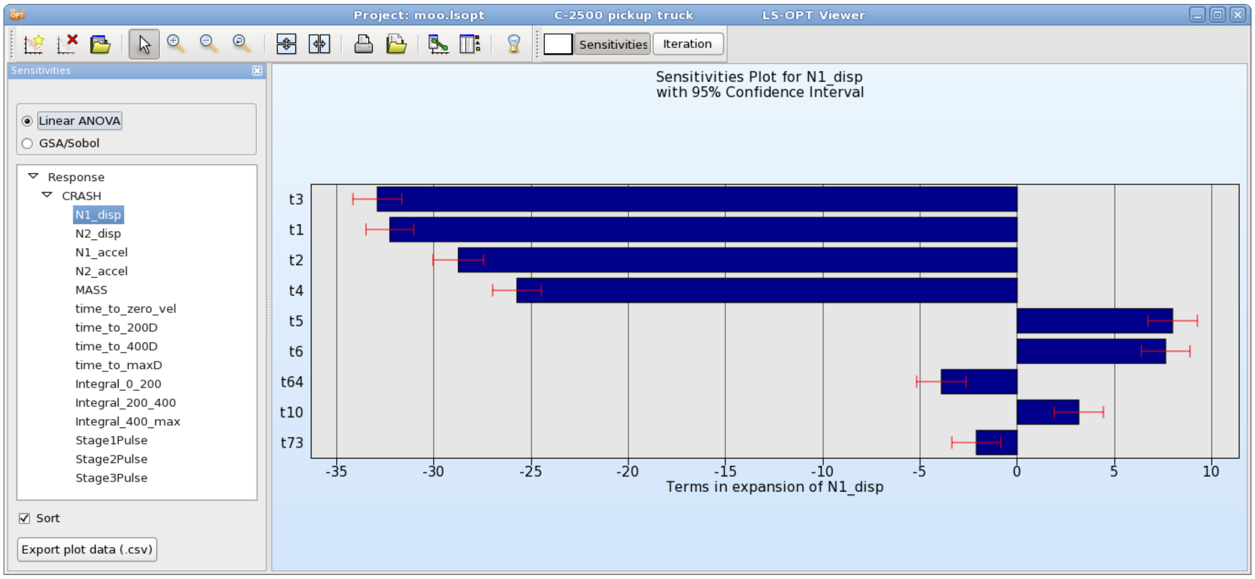


Figure 16-38: Linear ANOVA plot, sorted

GSA/Sobol

A global sensitivity analysis is only performed if **Global Sensitivities** is selected in the Task dialog or selected from the **Add** menu of the main GUI window, see Section 4.11.

Figure 16-39 displays an example of a global sensitivities plot. Each bar represents the contribution of a variable to the variance of the respective response (total sensitivity index). The values are normalized such that the sum of all displayed values is 100%. The values are displayed in the labels. For sorted plots, the cumulative sensitivity indices of all values in descending order are also displayed in the label.

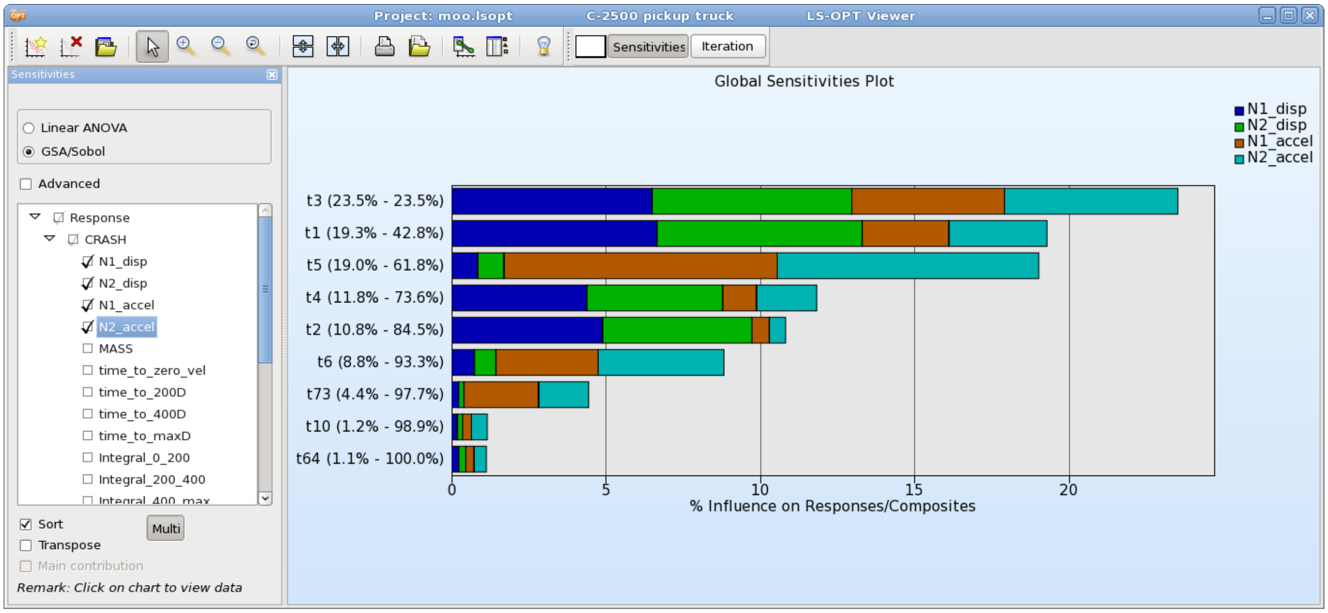
Clicking on the chart displays the respective sensitivity values and variances in the plot.

Options

Table 16-18: GSA/Sobol Plot options

Selection	Description
Sort	Sorts data by relevance
Transpose	Sensitivity values are grouped by response/composite
Main contribution	Main contribution is displayed in addition to total contribution

Multi	Allows selection of multiple responses/composites
Advanced	Weighted sums of the GSA values are displayed. A weight for each entity can be selected by using the slider right of the entity or by entering a value in the textfield (switch to textfield using the icon at the top of the entity list).
GSA Subregion	Sensitivities calculated in the selected subregion are displayed, Section 4.11.1. The range of each variable is displayed in the label.



○ Linear ANOVA
● GSA/Sobol
☐ Advanced

- Response
  - CRASH
    - ☑ N1_disp
    - ☑ N2_disp
    - ☑ N1_accel
    - ☑ N2_accel
    - ☐ MASS
    - ☐ time_to_zero_vel
    - ☐ time_to_200D
    - ☐ time_to_400D
    - ☐ time_to_maxD
    - ☐ Integral_0_200
    - ☐ Integral_200_400
    - ☐ Integral_400_max

☑ Sort
☐ Transpose
☐ Main contribution

Remark: Click on chart to view data

Figure 16-39: Sorted global sensitivities of all responses and composites

Principal Component Analysis

A principal component analysis is performed on a history or a multi-response if it has been selected in the entity definition, Section 6.1.3.

Figure 16-40 displays an example of a principal component analysis plot. Each blue bar represents the main effect (sensitivity indices) of a variable to the respective Principal Component and the selected history or multi-response. The grey bar represents the interaction effect (only 2-variables interactions are considered).

The main and interaction effects are represented by the main and interaction sensitivity indices which represents the contribution to the variance of the corresponding Principal Component. Sensitivity indices are always values between 0 and 1.

Principal Components are displayed in decreasing order according to their contribution to the total variance. This importance is represented by the inertia of the Principal Component. (Section 23.5). Only Principal Components with inertia > 0.9 are displayed until the cumulative variance reaches 95% of the total variance or 15 components.

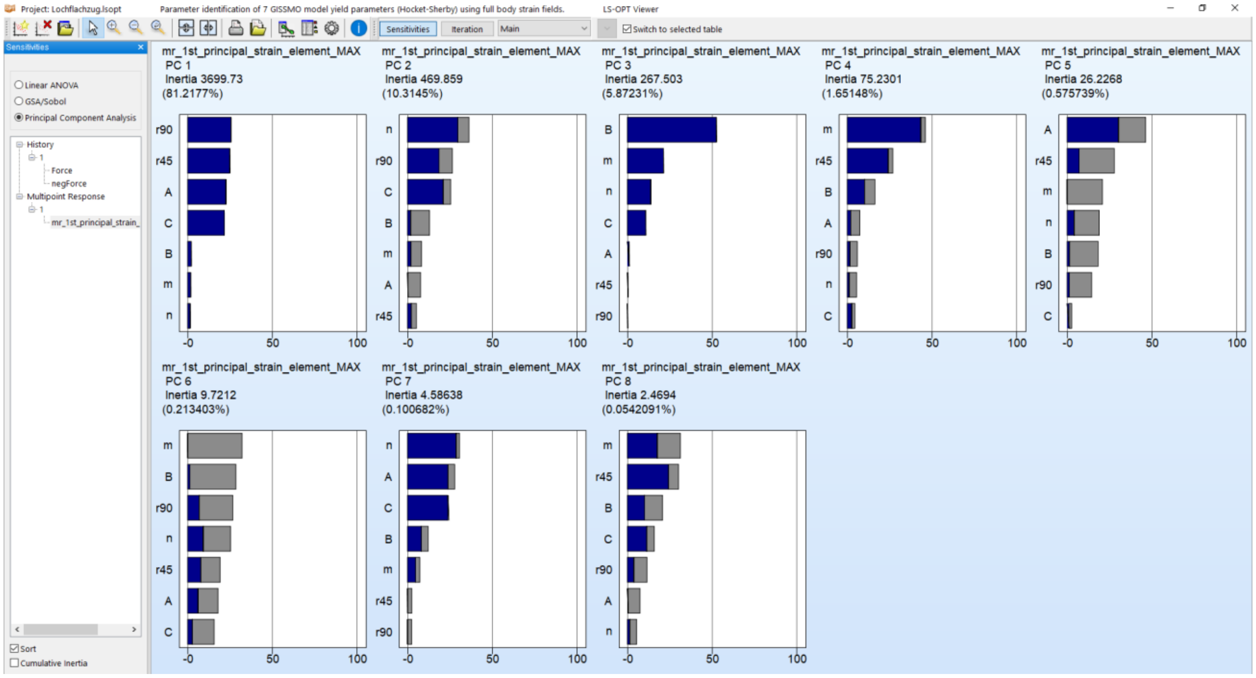

Project: Lochflachzug.lsopt
Parameter identification of 7 GISSMO model yield parameters (Hocket-Sherby) using full body strain fields.
LS-OPT Viewer
Sensitivities
Iteration
Main
☑ Switch to selected table
○ Linear ANOVA
○ GSA/Sobol
● Principal Component Analysis
History
1
Force
negForce
Multipoint Response
1
mr_1st_principal_strain_
☑ Sort
□ Cumulative Inertia
mr_1st_principal_strain_element_MAX PC 1 Inertia 3699.73 (81.2177%)
mr_1st_principal_strain_element_MAX PC 2 Inertia 469.859 (10.3145%)
mr_1st_principal_strain_element_MAX PC 3 Inertia 267.503 (5.87231%)
mr_1st_principal_strain_element_MAX PC 4 Inertia 75.2301 (1.65148%)
mr_1st_principal_strain_element_MAX PC 5 Inertia 26.2268 (0.575739%)
mr_1st_principal_strain_element_MAX PC 6 Inertia 9.7212 (0.213403%)
mr_1st_principal_strain_element_MAX PC 7 Inertia 4.58638 (0.100682%)
mr_1st_principal_strain_element_MAX PC 8 Inertia 2.4694 (0.0542091%)
-0
50
100

Figure 16-40: Principal Component Analysis plot, sorted

Options

Table 16-19: Principal Component Analysis Plot options

Selection	Description
Sort	Sorts variable by relevance
Cumulative Inertia	Plots the variable contribution to the total inertia (all components)

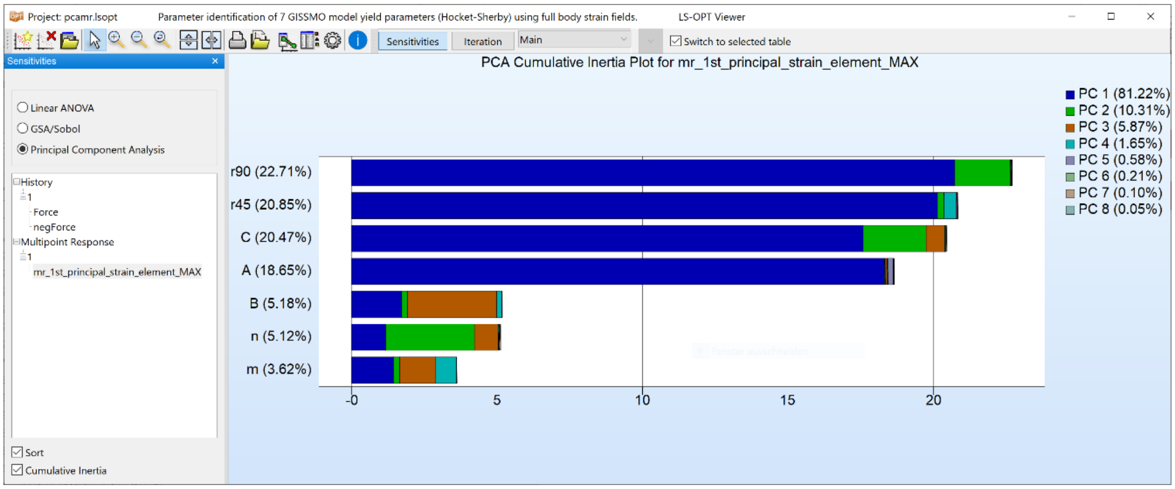


Figure 16-41: Principal Component Analysis, Cumulative Inertia plot, sorted

16.3.5. History Plot

If the **Approximate History** option is set in the Sampling dialog Features tab of LS-OPTui, a database that approximates the histories for any design point using metamodels is provided, see Section 9.5.1. If **Predicted Histories** is selected in the **Options** tab, the history evaluated on the metamodel for the selected design point is visualized, Figure 16-43. Error measures can be plotted to judge the quality of the metamodels using the **Value to plot** selection at the bottom of the **Setup** tab. If the predicted histories are colored by variable, multiple curves are plotted for equidistant values in the range of the selected variable. This visualizes the influence of the selected parameter on the history curve, Figure 16-18. A variable may also be selected as y entity to get a 3D history plot.

The **Center on …** options right of the Variables set the variable sliders to specific values that can be selected from the list that appears by clicking on the button, Table 16-20.

Table 16-20: History Plot Center on … - options for variable values

Selection – Center on …	**Description**
Optimum	Set variable sliders to optimum of current iteration
Nearest history	Set variable sliders to variable values of nearest history, this is the computed history with design point closest to selected design point for predicted history
Selected point	Set variable sliders to a selected point, e.g. a Pareto optimal solution Only active if there is only one selected point

Options

Table 16-21: Predicted History options

Selection	**Description**
Nearest	Show computed history with design point closest to selected design point
History ± Residual standard deviation	Show standard deviation of residuals of history prediction metamodels
History ± Max/Min Residual	Show maximal and minimal residual of history prediction metamodels
Number of predicted curves	Number of plotted curves if histories are colored by variable
Variable values	Values of fixed variables are displayed in the plot.

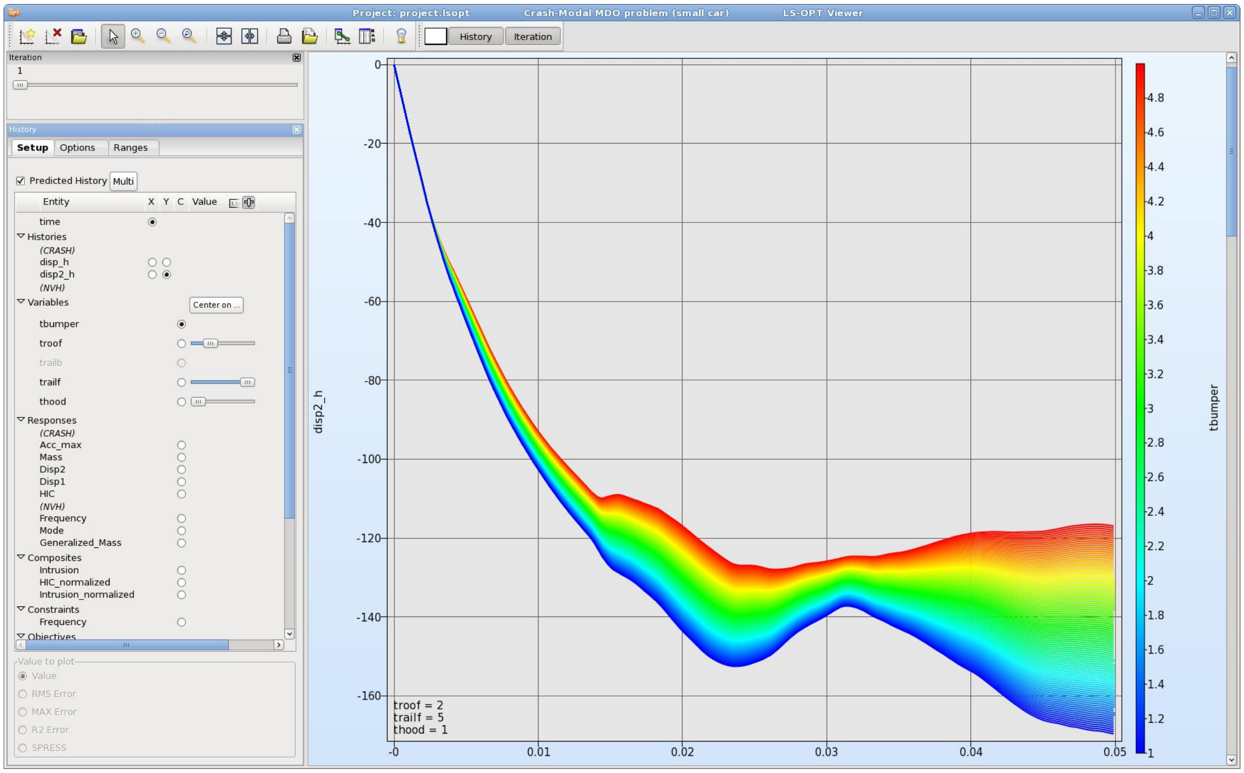


Figure 16-42: Predicted Histories colored by variable

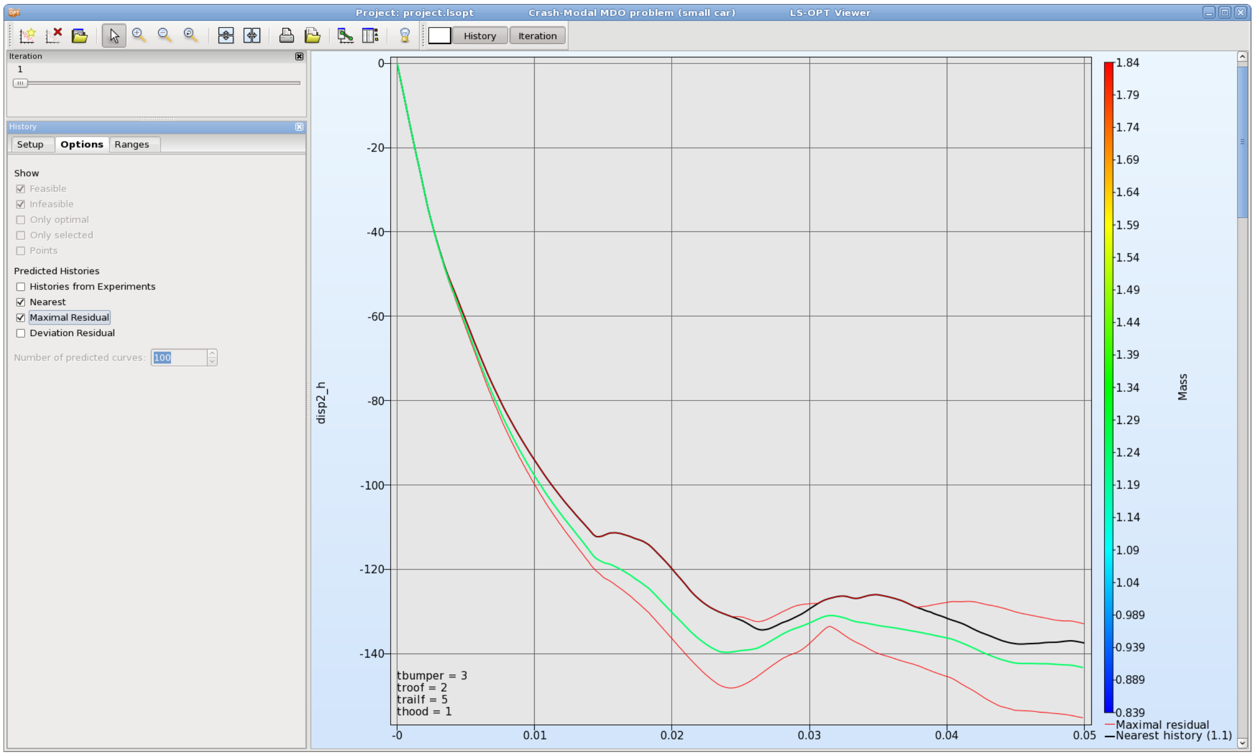


Figure 16-43: Predicted History with nearest history and maximal residual

Table 16-22: History Plot options – Statistics of residuals

Selection	Description
Mean	Mean values of residual values
Standard deviation	Standard deviation of residual values
Max	Maximal values of residual values
Min	Minimal values of residual values

Table 16-23: History Plot Options - Advanced Statistics

Selection	Description
Range	Range of the residual values (maximum values minus the minimum values)
Sample index of Min	ID of the simulation job where the maximum residual occurred. This can be used to identify the jobs likely to contain a different bifurcation.
Sample index of Max	ID of the simulation job where the minimum residual occurred. This can be used to identify the jobs likely to contain a different bifurcation.

16.3.6. Classifiers Plot

2D plots of classifier boundaries can be visualized, Figure 16-44. Variables can be selected to be displayed on the x and y axes. The checkboxes available in the **Cls** column allow the selection of the classifiers to be visualized.

For the description of the **Points** options see section 16.3.1.

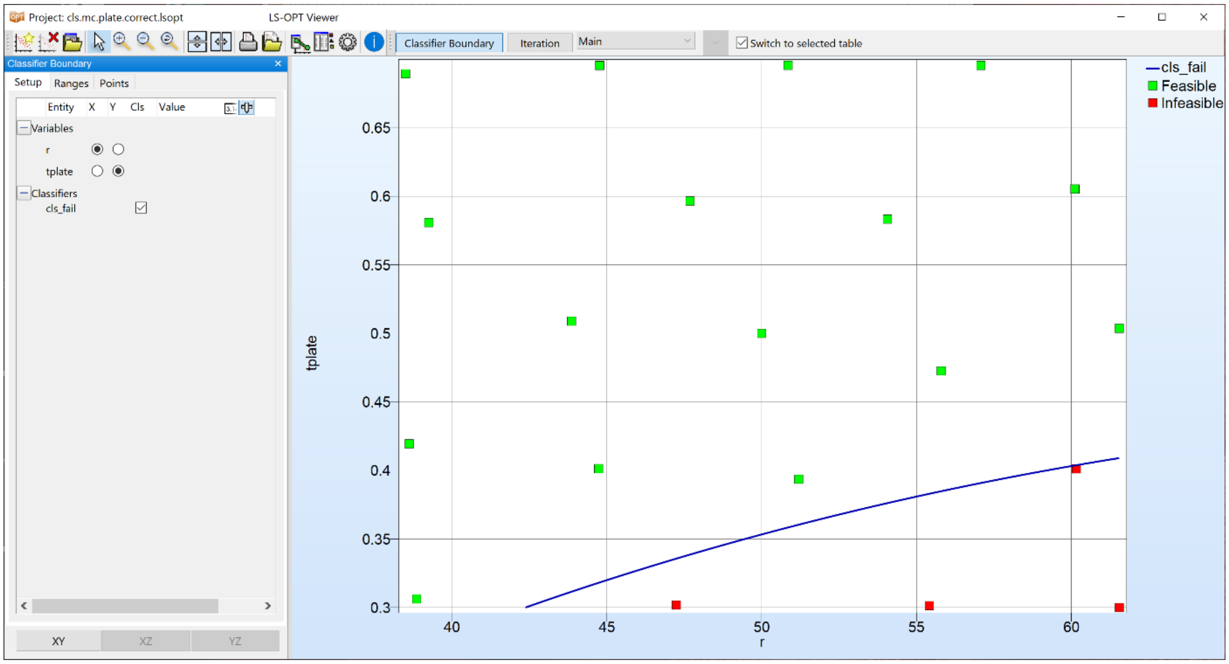


Figure 16-44: Classifier boundary plot

16.4. Visualization of Optimization Results

16.4.1. Optimization History

The optimization history of a variable, dependent, response, constraint, objective, multi-objective or the approximation error parameters of pure responses (not composites or expressions) shows the changes of the respective values of the optimum over the iterations. For the variables, the upper and lower bounds (subregion) are also displayed, Figure 16-45. For all the dependents, responses, objectives, constraints and maximum violation, a black solid line indicates the predicted values. The red squares represent the computed values at the starting point of each iteration (Figure 16-46). For constraints, the lower and upper bound are displayed with a blue and red line, respectively. For the error parameters, only one solid red line of the optimization history is plotted. RMS, Maximum and R^2 error indicators are available.

For multi-objective optimization, MOO performance metrics can be displayed, Section 26.15.

Additional options are explained in Table 16-24.

Table 16-24: Options for Optimization History plot

Option	Description
Multi	Plot optimization histories of all selected entities in the same plot.
Omit computed values	Only plot predicted values.

Omit predicted values	Only plot computed values.
Omit variable bounds	Don’t plot variable bounds for variables.
Omit constraint bounds	Don’t plot constraint bounds for constraints.
Scale variable values	Scale variable values to [0,1].
Core Solver Progress	Plot core solver progress.

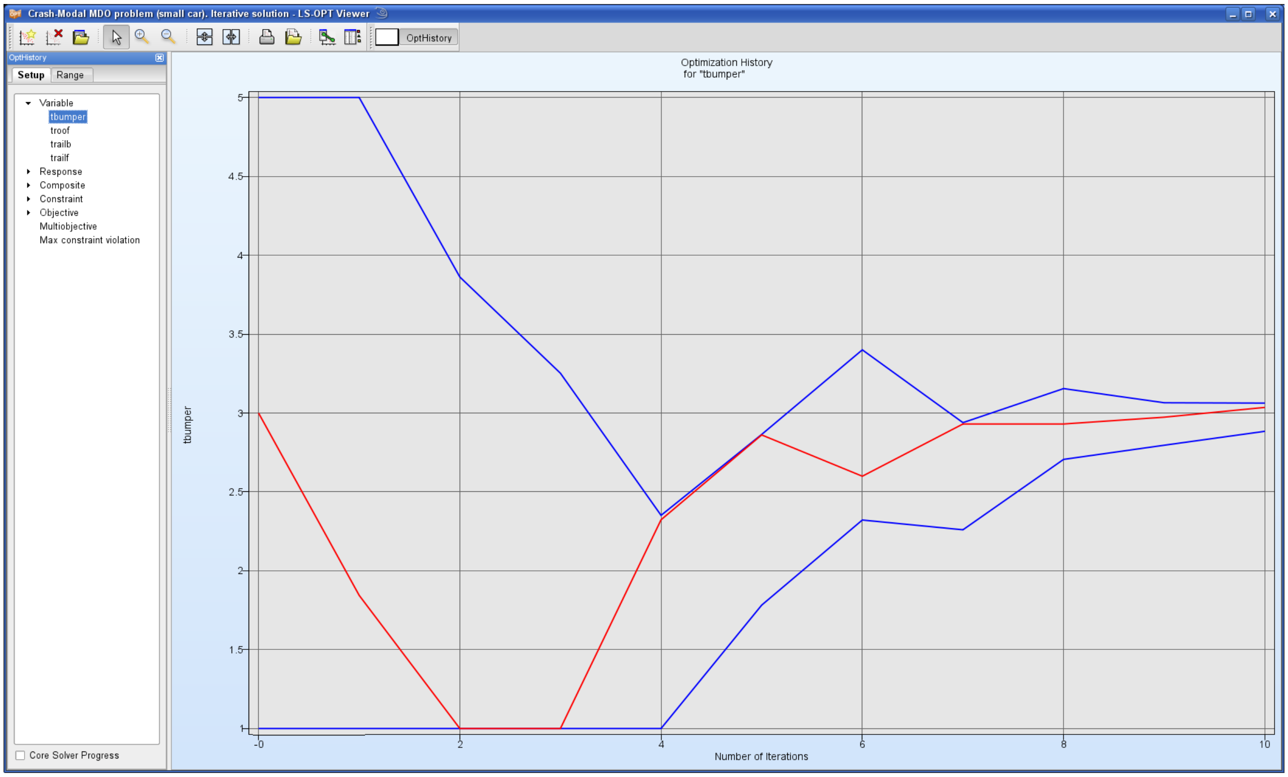


Figure 16-45: Optimization History plot of a variable – variable values (red) and subregions (blue)

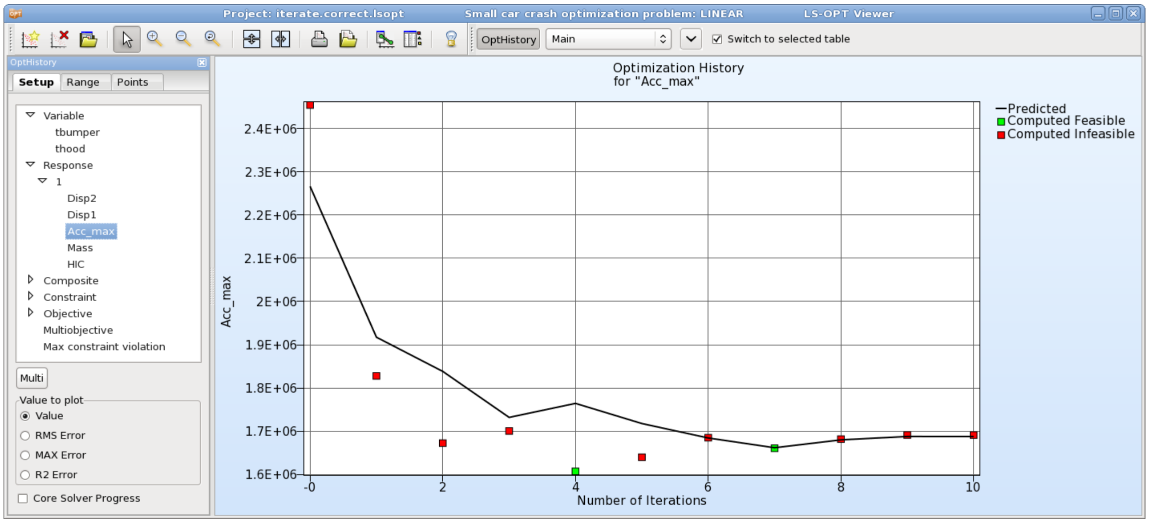


Figure 16-46: Optimization History of a response – computed (red and green points) and predicted (black) values

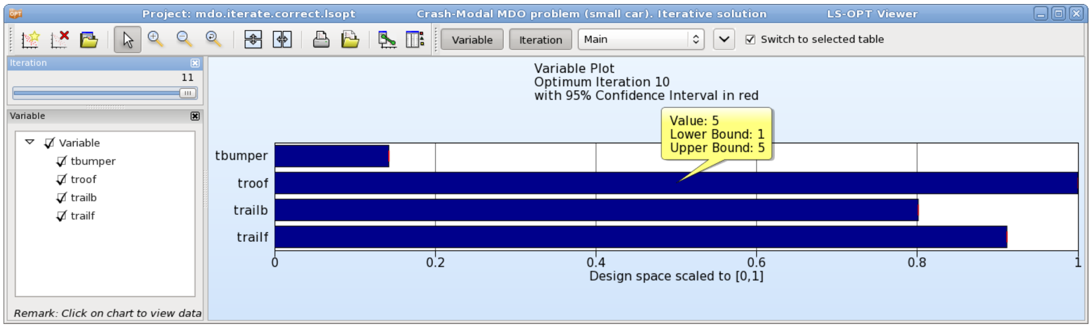
Points Options

Table 16-25: Optimization History Points options

Option	Description
Optimum	Plot computed and predicted optimum
Best Computed	Plot best computed point of each iteration. It is the run with the smallest multiobjective and constraint violation, respectively. In the case that all points are infeasible, the most feasible point is displayed.

16.4.2. Variables Plot

The variables plot visualizes variable values and confidence intervals for *.1 run of the selected iteration in a range scaled to [0,1], Figure 16-47. Clicking on the charts displays the actual value and the bounds on the plot.


Project: mdo.iterate.correct.lsopt
Crash-Modal MDO problem (small car). Iterative solution
LS-OPT Viewer
Variable
Iteration
Main
☑ Switch to selected table
Iteration
11
Variable
☑ Variable
☑ tbumper
☑ troof
☑ trailb
☑ trailf
Variable Plot
Optimum Iteration 10
with 95% Confidence Interval in red
tbumper
troof
trailb
trailf
0
0.2
0.4
0.6
0.8
1
Design space scaled to [0,1]

Value: 5
Lower Bound: 1
Upper Bound: 5

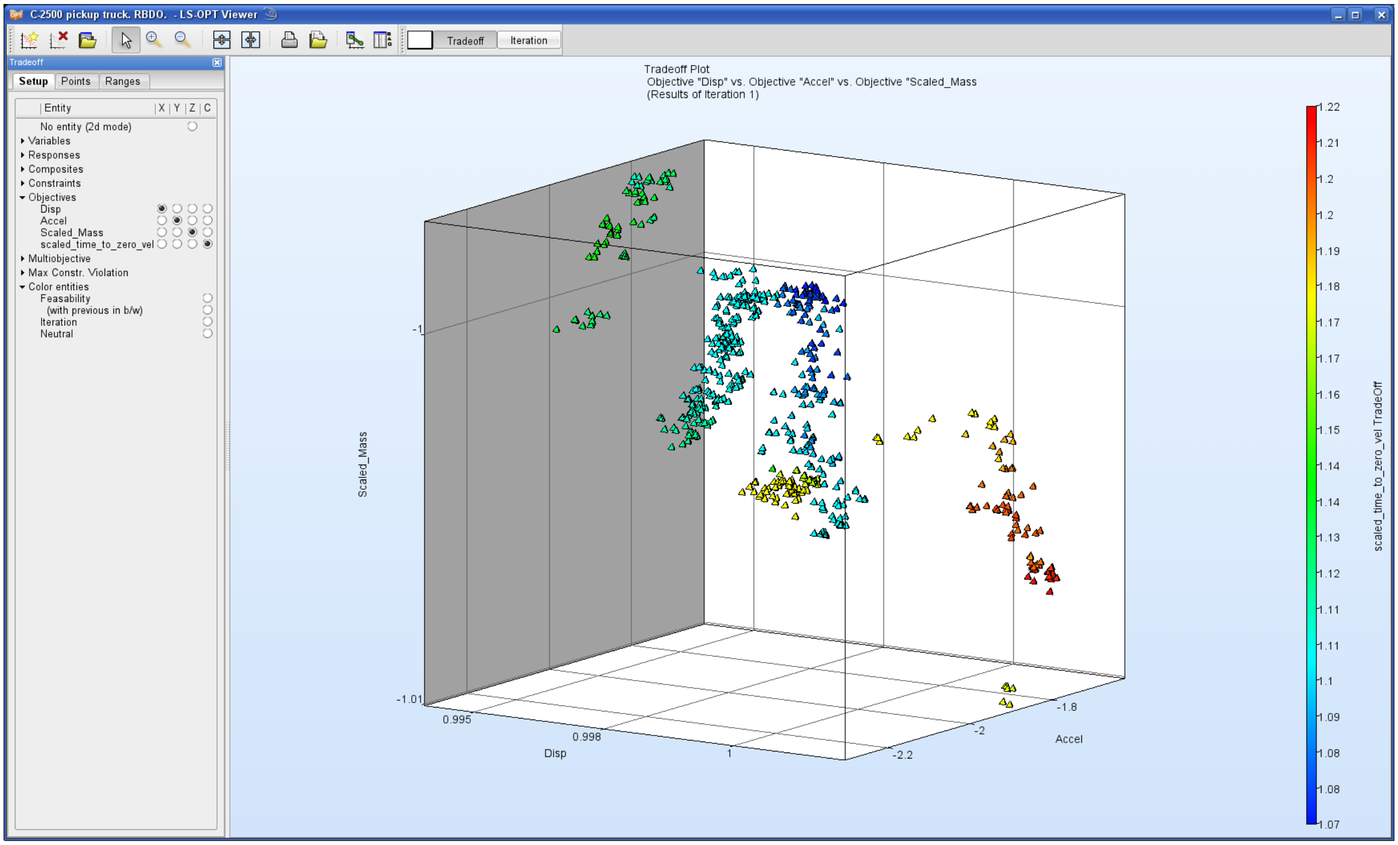
Remark: Click on chart to view data

Figure 16-47: Variable Plot

16.5. Visualization of Pareto Optimal Solutions

16.5.1. Tradeoff Plot

The Tradeoff plot (Section 26.14.1) functions similar to the Scatter plot, Section 16.2.2, but the default setting is here to plot Pareto optimal solution data instead of Analysis Result data.



C-2500 pickup truck. RBDO. - LS-OPT Viewer

Tradeoff

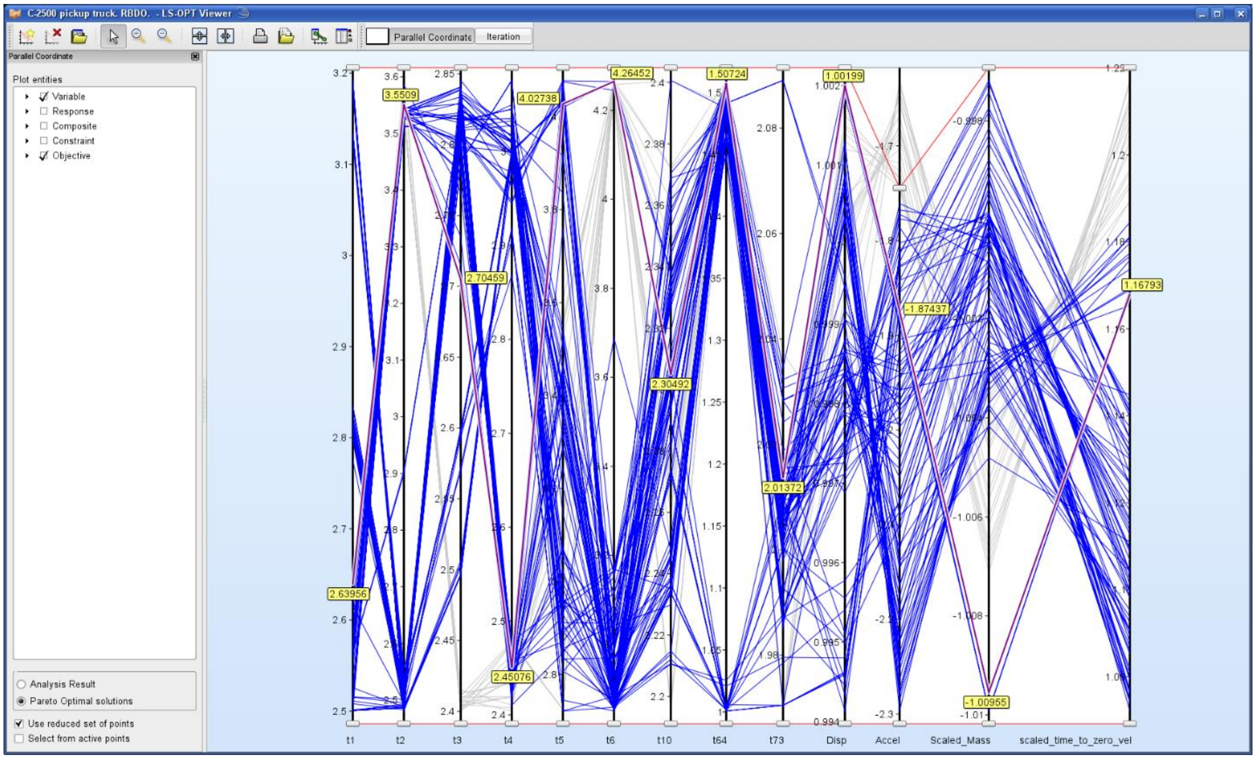
Entity	X	Y	Z	C
No entity (2d mode)				○
▸ Variables				
▸ Responses				
▸ Composites				
▸ Constraints				
▾ Objectives				
Disp	●	○	○	○
Accel	○	●	○	○
Scaled_Mass	○	○	●	○
scaled_time_to_zero_vel	○	○	○	●
▸ Multiobjective				
▸ Max Constr. Violation				
▾ Color entities				
Feasability				○
(with previous in b/w)				○
Iteration				○
Neutral				○

Tradeoff Plot
Objective "Disp" vs. Objective "Accel" vs. Objective "Scaled_Mass
(Results of Iteration 1)

Figure 16-48: Tradeoff plot

16.5.2. Parallel Coordinate Plot

The Parallel Coordinate Plot (Section 26.14.3) in the Pareto optimal solutions category functions similar to the Parallel Coordinate Plot described in Section 16.2.3, but here, the default setting is to visualize Pareto data.



Parallel Coordinate

Plot entities

- ☑ Variable
- □ Response
- □ Composite
- □ Constraint
- ☑ Objective

○ Analysis Result
● Pareto Optimal solutions

☑ Use reduced set of points
□ Select from active points

Figure 16-49: Parallel Coordinate Plot for Pareto optimal solutions with selected point (purple line)

16.5.3. Hyper-Radial Visualization

The hyper-radial visualization reduces multi-dimensional data to a two-dimensional graph by grouping the objectives and calculating a weighted sum for each group. These values are displayed in two dimensions. The designer may incorporate his preferences by selecting the weights. The best point with respect to the selected weights is colored purple in the plot (Figure 16-50). The theory of hyper-radial visualization is explained in Section 26.14.2.

Grouping

The objectives may be grouped using the 3-state buttons in the **Axis** column.

Table 16-26: Hyper-Radial Visualization Grouping options

Selection	Description
x	Add objective to the group displayed on the x axis
y	Add objective to the group displayed on the y axis
	Ignore objective

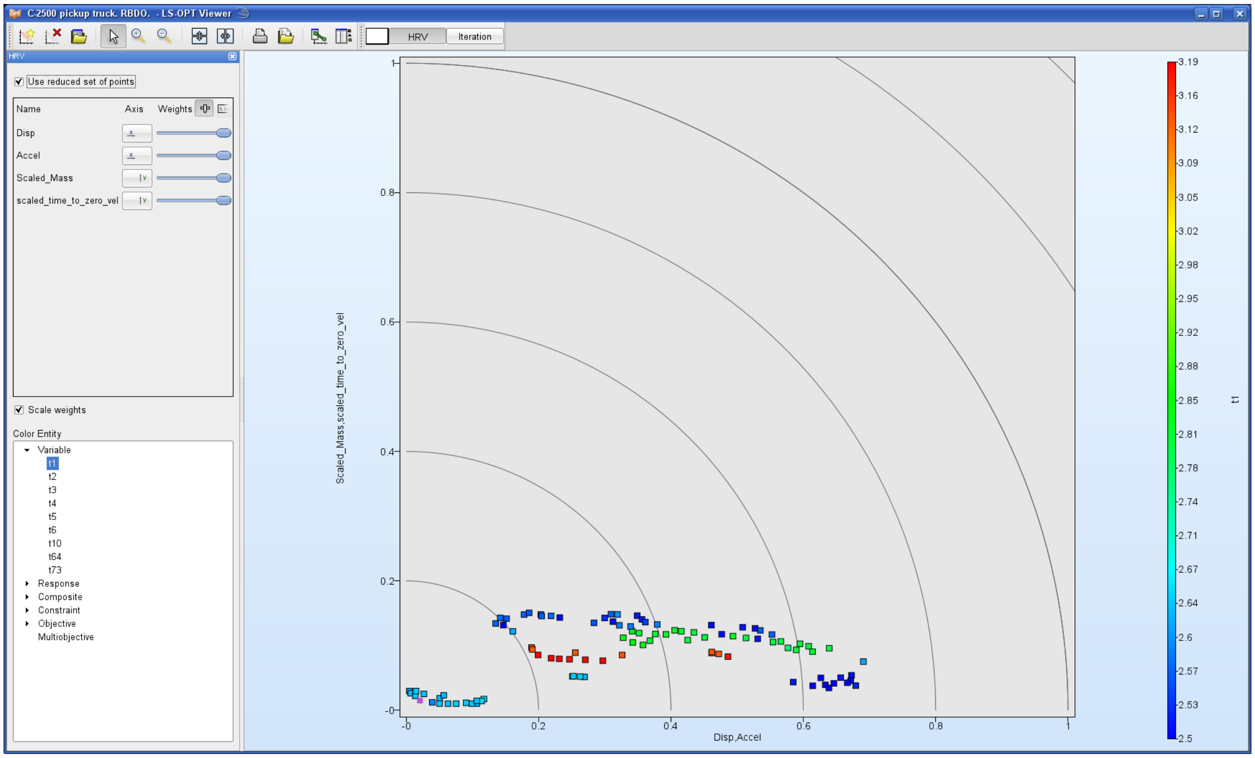
Selection of Weights

The weights may be selected using the sliders or the text fields in the **Weights** column. The selected values represent the ratio of the weights and are scaled internally such that the sum of the weights is 1.

Options

Table 16-27: Hyper-Radial Visualization options

Selection	Description
Use reduced set of points	Plot only reduced set of Pareto optimal solutions
Scale weights	Scale weights by range of objectives
Color Entity	Color entity for HRV points



C-2500 pickup truck. RBDO. - LS-OPT Viewer

☑ Use reduced set of points

Name	Axis	Weights
Disp	x	
Accel	x	
Scaled_Mass	y	
scaled_time_to_zero_vel	y	

☑ Scale weights

Color Entity: Variable t1

Figure 16-50: Hyper-radial visualization, equal weights, points colored by variable

16.5.4. Self-Organizing Maps

The theory of Self-Organizing Maps (SOM) is explained in Section 26.14.4. The default is to visualize Pareto optimal solutions.

Component Selection

By default, component maps of all objectives are displayed. To modify the plot, select the position in the dynamic grid, Figure 16-51, and the respective slot content. Refer to Section 26.14.4 for an explanation of the map types.

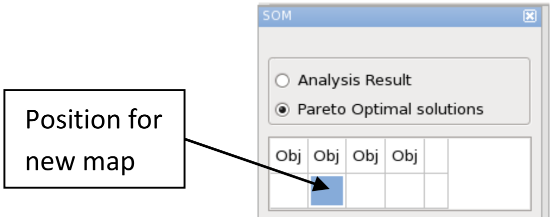


Figure 16-51: Selection of position for SOM

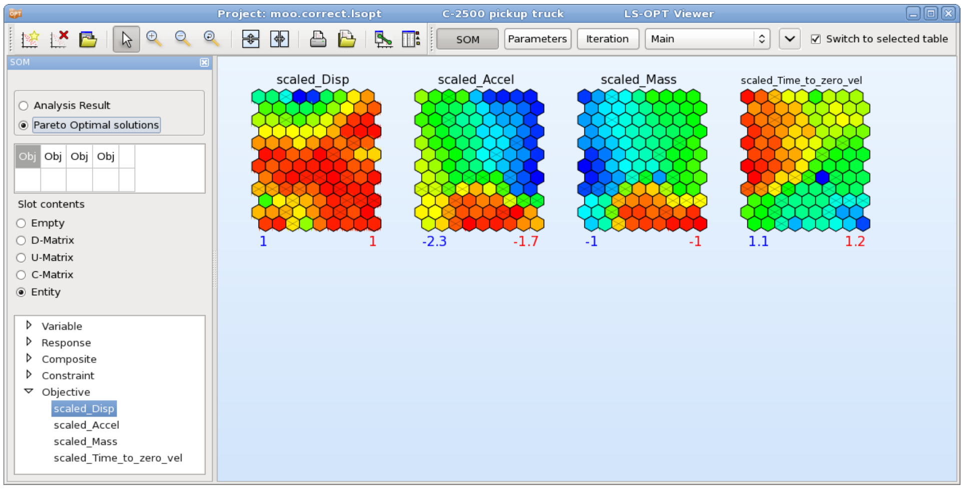


Figure 16-52: Self-Organizing Map, component plots of objectives

Parameter Panel

The advanced user may want to modify some parameters for the training of the SOM. These options are available in the **Parameters** panel. Modifications in the Parameter Panel effect retraining of the SOM.

Table 16-28: Self-Organizing Maps parameters

Selection	Description
Training Iterations	Number of iterations performed for training of SOM, default depends on honeycomb dimensions and number of data points
Initial Radius	Initial radius used for training of SOM, default depends on honeycomb dimensions
Honeycomb dimensions	Honeycomb dimensions, default 12x9

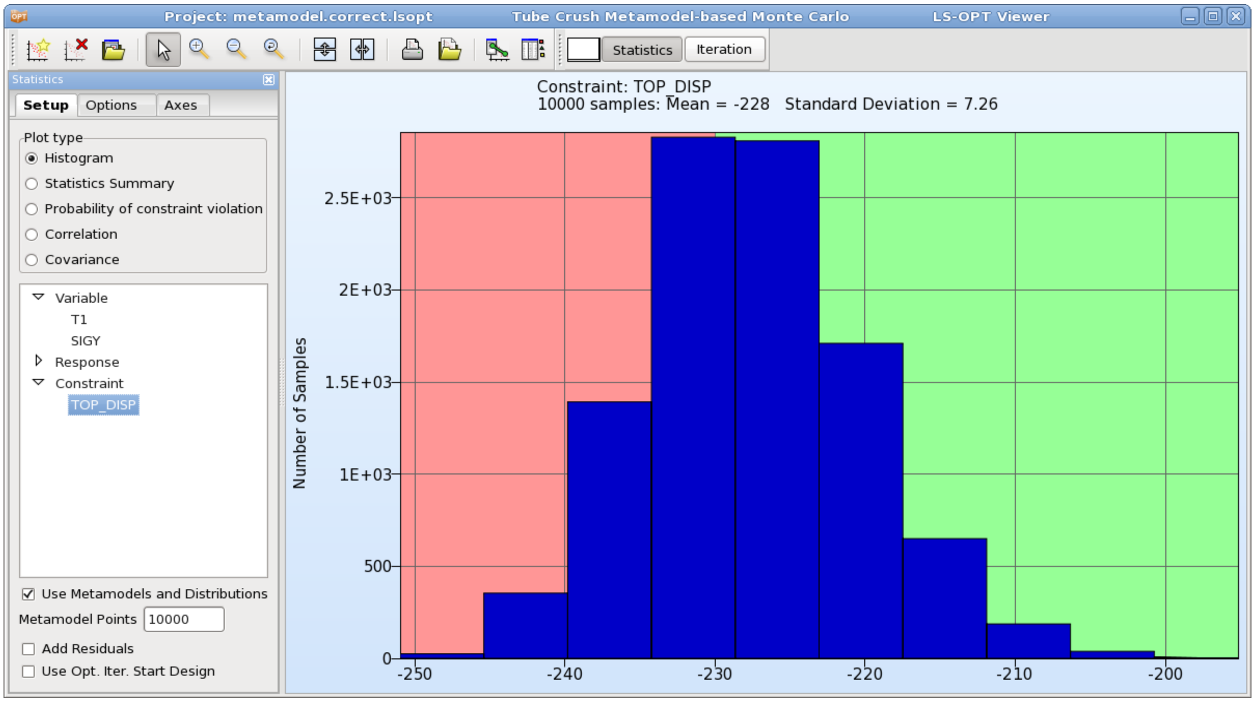
16.6. Stochastic Analysis

Two types of plots are available under stochastic analysis – *Statistical Tools* and *Stochastic Contribution*. These results are calculated based on metamodel approximations and variable distributions.

16.6.1. Statistical Tools

These plots are similar to simulation results-based plots (Section 16.2.6) except that metamodels are used for calculating the statistics. An LHS design of experiments is generated to perform metamodel-based Monte Carlo analysis for statistics calculation. The user can modify the number of Monte Carlo samples and can also use a large number, as the metamodel-based calculations are inexpensive. The available plot types under *Statistical Tools* are *Histogram*, *Statistics Summary*, *Probability of Constraint Violation*, *Correlation* and *Covariance*. It is possible to add effect of residuals while calculating statistics using metamodels. Figures for the various plot types are shown in the following sections. It is interesting to compare them to the respective plots in Section 16.2.6.

Histogram



Project: metamodel.correct.lsopt Tube Crush Metamodel-based Monte Carlo LS-OPT Viewer

Plot type

● Histogram
○ Statistics Summary
○ Probability of constraint violation
○ Correlation
○ Covariance

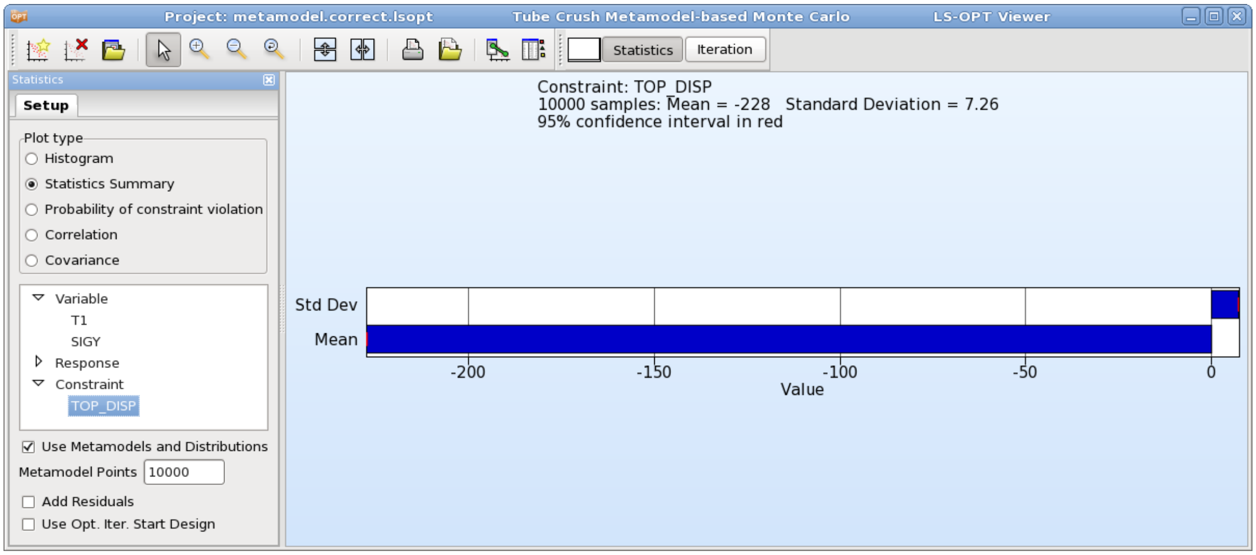
Variable
T1
SIGY
Response
Constraint
TOP_DISP

☑ Use Metamodels and Distributions
Metamodel Points 10000
□ Add Residuals
□ Use Opt. Iter. Start Design

Constraint: TOP_DISP
10000 samples: Mean = -228 Standard Deviation = 7.26

Figure 16-53: Histogram constructed using metamodel together with the statistical distribution of the variables. The background represents the feasibility status.

Summary



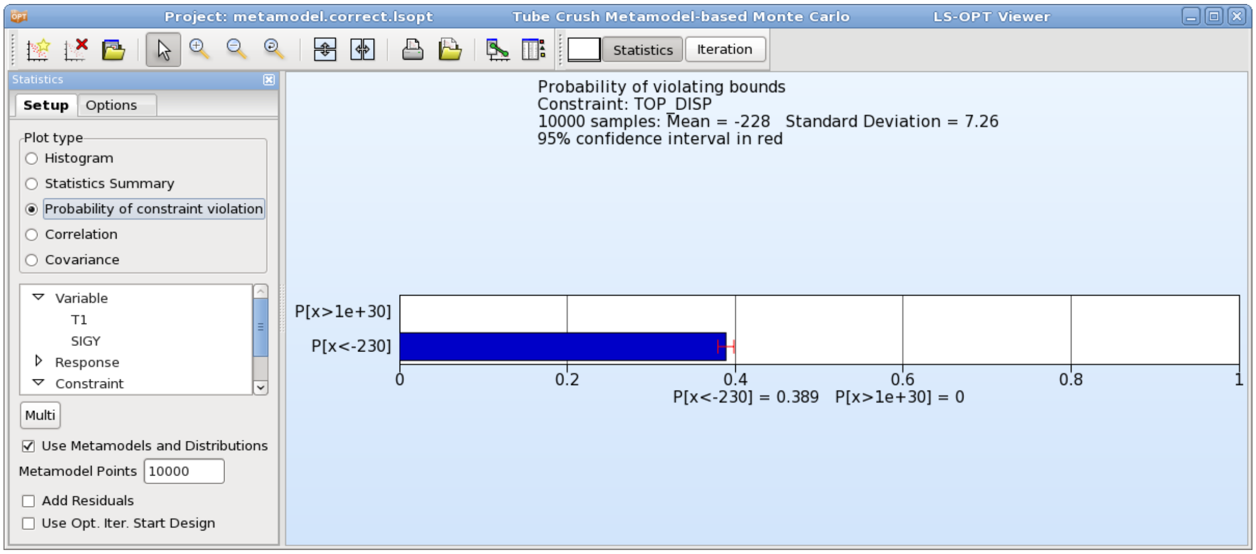
Constraint: TOP_DISP
10000 samples: Mean = -228   Standard Deviation = 7.26
95% confidence interval in red

Plot type
○ Histogram
● Statistics Summary
○ Probability of constraint violation
○ Correlation
○ Covariance

☑ Use Metamodels and Distributions
Metamodel Points 10000
☐ Add Residuals
☐ Use Opt. Iter. Start Design

Figure 16-54: Mean value and standard deviation constructed using metamodel together with the statistical distribution of the variables

Probability of violating constraint bounds



Probability of violating bounds
Constraint: TOP_DISP
10000 samples: Mean = -228   Standard Deviation = 7.26
95% confidence interval in red

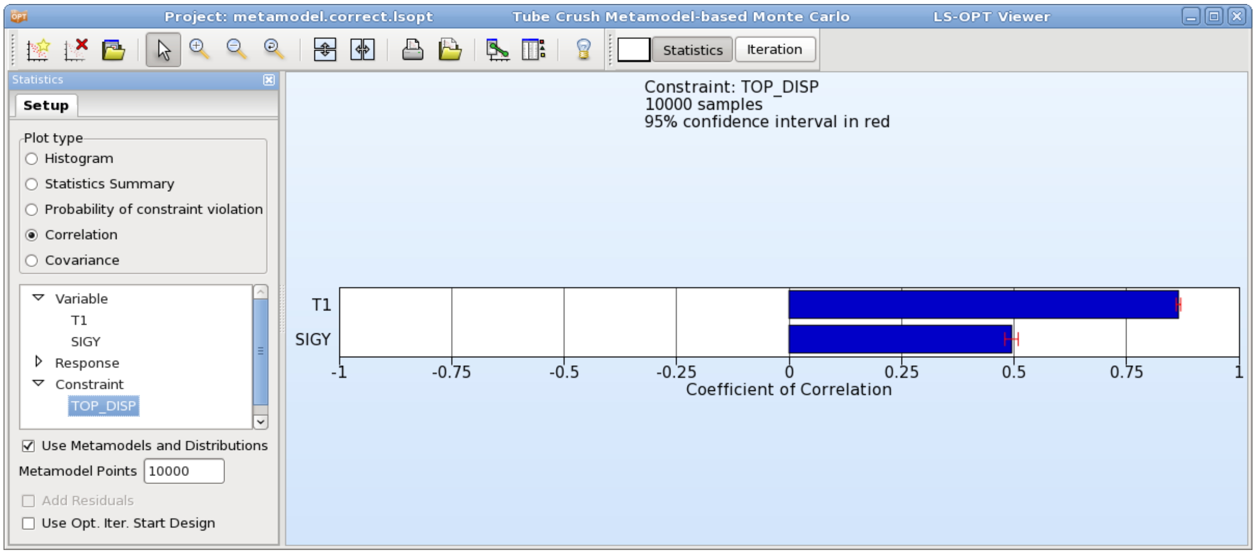
Plot type
○ Histogram
○ Statistics Summary
● Probability of constraint violation
○ Correlation
○ Covariance

☑ Use Metamodels and Distributions
Metamodel Points 10000
☐ Add Residuals
☐ Use Opt. Iter. Start Design

P[x<-230] = 0.389   P[x>1e+30] = 0

Figure 16-55: Probability of Response TOP_DISP < -230 with 95% confidence interval in red constructed using metamodel together with the statistical distribution of the variables

Correlation

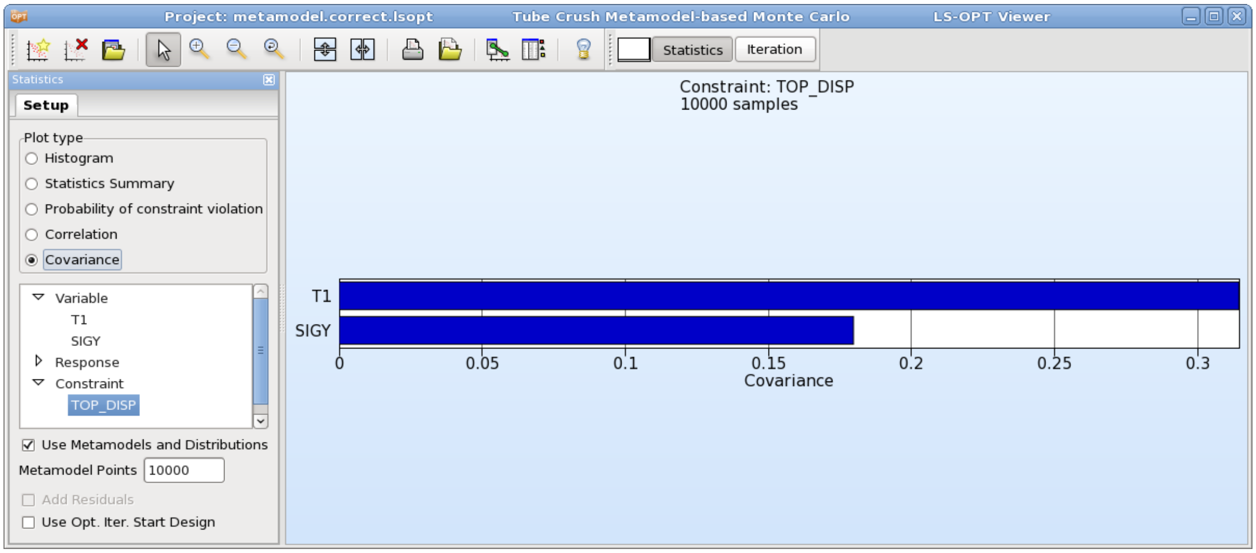


Plot type
○ Histogram
○ Statistics Summary
○ Probability of constraint violation
● Correlation
○ Covariance

☑ Use Metamodels and Distributions
Metamodel Points 10000
□ Add Residuals
□ Use Opt. Iter. Start Design

Figure 16-56: Correlation Bars evaluated on metamodel

Covariance



Plot type
○ Histogram
○ Statistics Summary
○ Probability of constraint violation
○ Correlation
● Covariance

☑ Use Metamodels and Distributions
Metamodel Points 10000
□ Add Residuals
□ Use Opt. Iter. Start Design

Figure 16-57: Covariance evaluated using metamodel

16.6.2. Stochastic Contribution

The stochastic contribution of the variables to the variance of the responses and composites (Section 29.7) can be displayed as a bar chart.

Optionally, the user can select to display the influence of the residuals from the metamodel fit and the effect of all the variables summed together. Contrasting these two values indicates how well the cause-effect relationship for the specific response is resolved. If both the residuals and the sum of the contributions are requested, then a total is displayed that is the sum of the contributions of all the variables as well as the residuals. The influence of residuals is calculated as the standard deviation of the residuals at the samples. The square root of the sum of residual and response variances gives the total contribution.

The computations are done using the metamodels and stored in databases for visualization. Higher order effects, if any, are included in the results plotted. In the Sobol terminology, the total effect as opposed to the main effect is therefore plotted. See Section 29.7 for the details.

For optimization, the stochastic contribution is computed using the optimal design. The stochastic contribution panel is shown in Figure 16-58.

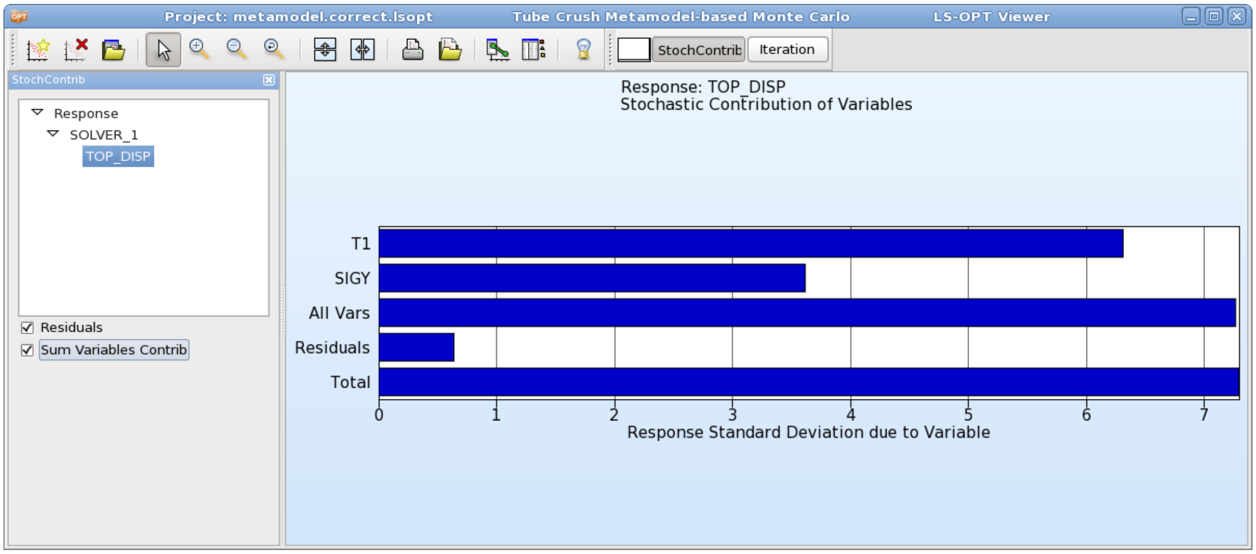


Figure 16-58: Stochastic Contribution plot

16.6.3. Statistics History

If a metamodel based Monte Carlo analysis is performed, the Statistics History plot can be used to visualize statistical values for all entities over the iterations evaluated on the metamodels.

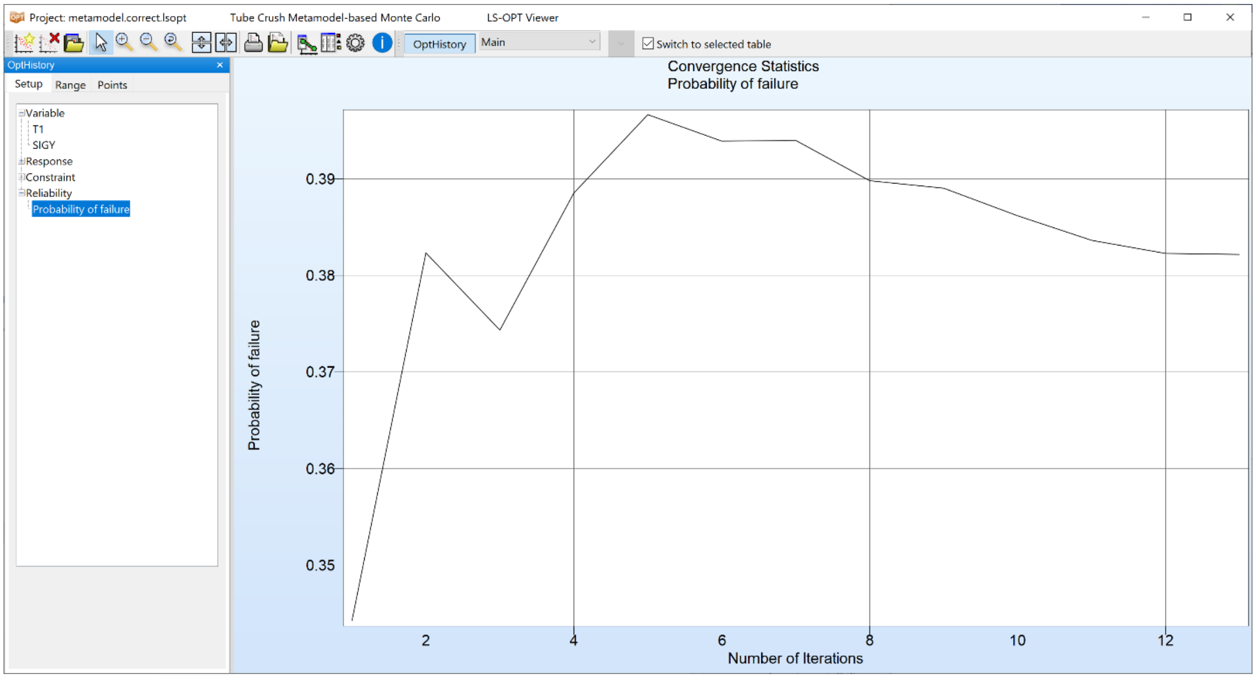


Figure 16-59: Statistics History plot visualizing the probability of failure considering all constraints over the iterations

16.7. REFERENCES

[1] Myers, R.H. and Montgomery, D.C. Response Surface Methodology. Process and Product Optimization using Designed Experiments. Wiley, 1995

17. Interactive Tables

This chapter describes the interactive tables displayed in the Point Selection Window accessible from the Viewer, Chapter 16. Interactive tables contain the point data generated by LS-OPT during the run, and the user has the possibility to operate on the data as described in the following sections. The idea of interactive tables is to provide a limited spreadsheet-like behavior to tables representing design data. This includes automatic interaction with Viewer plots such as Scatter plots, Surface plots and Parallel Coordinate plots. Interactive tables can be used as tools to interpolate values from existing metamodels.

17.1.1. Point Selection Window

The point selection window is accessible from the *Point selection window* icon in the control bar of a Viewer plot window, Table 16-2, and by marking of points in plots, see below. It shows the data of the design points displayed in plots. It is always available even when the displayed plot does not support showing any points, e.g. the bar chart plots. The interactive tables can operate in different modes; All points, Marked, Plotted and Filtered, Table 17-1.

All points mode shows all the points regardless of it being marked in the plot or whether any filtering is applied, Figure 17-2. This is the default when opening the point selection window from the toolbar.

Marked mode of the point selection window only shows points that are marked in the plot either by clicking on a single design point, Figure 17-1, or by selecting multiple points using rectangular selection (rubber-banding). This is the default mode when opening the point selection as a result of marking the points.

Plotted mode shows all points visible in a plot. It is filtered by the plot specific options such as iteration.

Filtered mode only shows points that are satisfied by any filters set in the same table.

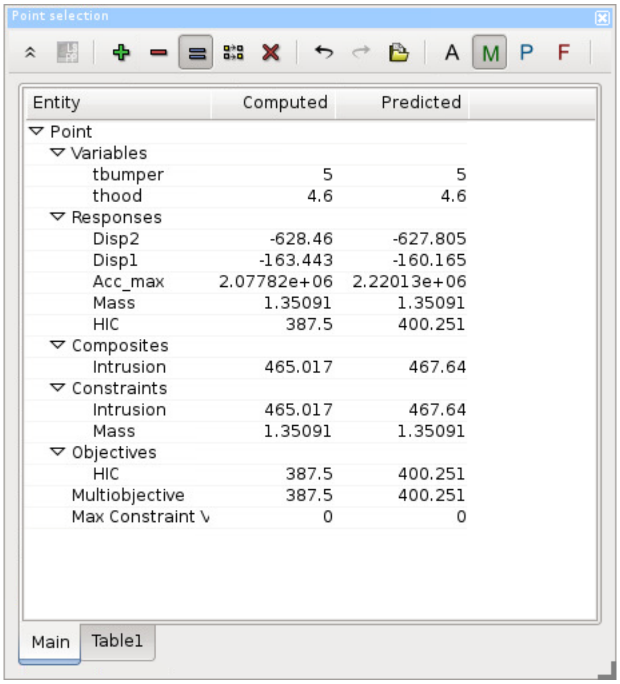
The Point selection window can be seen in Marked mode with a single point marked and in ‘All Points’ mode as shown in Figure 17-1 and Figure 17-2 respectively.

Operating on points in the point selection window is done in two ways; marking and selecting.

The points on Scatter, Tradeoff, Surface, Accuracy, Optimization History and HRV plots, and lines on Parallel Coordinate plots (PCP) and History plots allows marking points from the plot. Marking is done by clicking a point or curve in the plot or by holding the mouse down to mark several points using a rectangular box (rubber-banding). The SOM plot (Section 16.5.4) also supports marking. If a cell is selected, all points that are mapped to the selected cell are displayed in the point selection window. Marked points are highlighted in the plot and a check mark is shown in the Marked column of the point selection window, as shown in Figure 17-2.

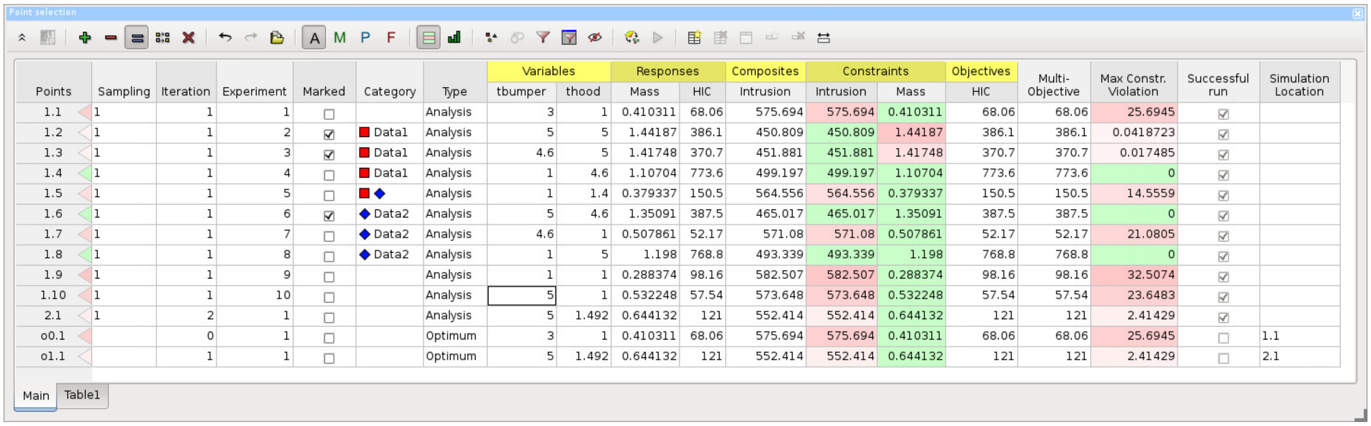
Selection is made by clicking the spreadsheet in the Point selection window, i.e. click the row label to the left to select an entire row, click the column label to select an entire column and click cells to select cells. Several cells and rows can be selected using *Ctrl* and *Shift* keys. Operations on the selected points in the point selection window can be made using either the toolbar or the right-click menus. It is sufficient to select a single cell when an operation on a row should be made.

Point selection is integrated, hence points are highlighted in all plots that use the same table, Figure 17-3.



Entity	Computed	Predicted
Point		
Variables		
tbumper	5	5
thood	4.6	4.6
Responses		
Disp2	-628.46	-627.805
Disp1	-163.443	-160.165
Acc_max	2.07782e+06	2.22013e+06
Mass	1.35091	1.35091
HIC	387.5	400.251
Composites		
Intrusion	465.017	467.64
Constraints		
Intrusion	465.017	467.64
Mass	1.35091	1.35091
Objectives		
HIC	387.5	400.251
Multiobjective	387.5	400.251
Max Constraint V	0	0

Figure 17-1: Point selection window in Marked mode for single point selection



							Variables		Responses		Composites	Constraints		Objectives				
Points	Sampling	Iteration	Experiment	Marked	Category	Type	tbumper	thood	Mass	HIC	Intrusion	Intrusion	Mass	HIC	Multi-Objective	Max Constr. Violation	Successful run	Simulation Location
1.1	1	1	1	□		Analysis	3	1	0.410311	68.06	575.694	575.694	0.410311	68.06	68.06	25.6945	☑	
1.2	1	1	2	☑	🟥 Data1	Analysis	5	5	1.44187	386.1	450.809	450.809	1.44187	386.1	386.1	0.0418723	☑	
1.3	1	1	3	☑	🟥 Data1	Analysis	4.6	5	1.41748	370.7	451.881	451.881	1.41748	370.7	370.7	0.017485	☑	
1.4	1	1	4	□	🟥 Data1	Analysis	1	4.6	1.10704	773.6	499.197	499.197	1.10704	773.6	773.6	0	☑	
1.5	1	1	5	□	🟥 ◆	Analysis	1	1.4	0.379337	150.5	564.556	564.556	0.379337	150.5	150.5	14.5559	☑	
1.6	1	1	6	☑	◆ Data2	Analysis	5	4.6	1.35091	387.5	465.017	465.017	1.35091	387.5	387.5	0	☑	
1.7	1	1	7	□	◆ Data2	Analysis	4.6	1	0.507861	52.17	571.08	571.08	0.507861	52.17	52.17	21.0805	☑	
1.8	1	1	8	□	◆ Data2	Analysis	1	5	1.198	768.8	493.339	493.339	1.198	768.8	768.8	0	☑	
1.9	1	1	9	□		Analysis	1	1	0.288374	98.16	582.507	582.507	0.288374	98.16	98.16	32.5074	☑	
1.10	1	1	10	□		Analysis	5	1	0.532248	57.54	573.648	573.648	0.532248	57.54	57.54	23.6483	☑	
2.1	1	2	1	□		Analysis	5	1.492	0.644132	121	552.414	552.414	0.644132	121	121	2.41429	☑	
o0.1		0	1	□		Optimum	3	1	0.410311	68.06	575.694	575.694	0.410311	68.06	68.06	25.6945	□	1.1
o1.1		1	1	□		Optimum	5	1.492	0.644132	121	552.414	552.414	0.644132	121	121	2.41429	□	2.1

Figure 17-2: Point selection window in All points mode for multi-point selection in Table1. Infeasible designs are highlighted in red. Points highlighted in the plot are checked in the Marked column. Highlighted points in plots can be seen in Figure 17-3 below.

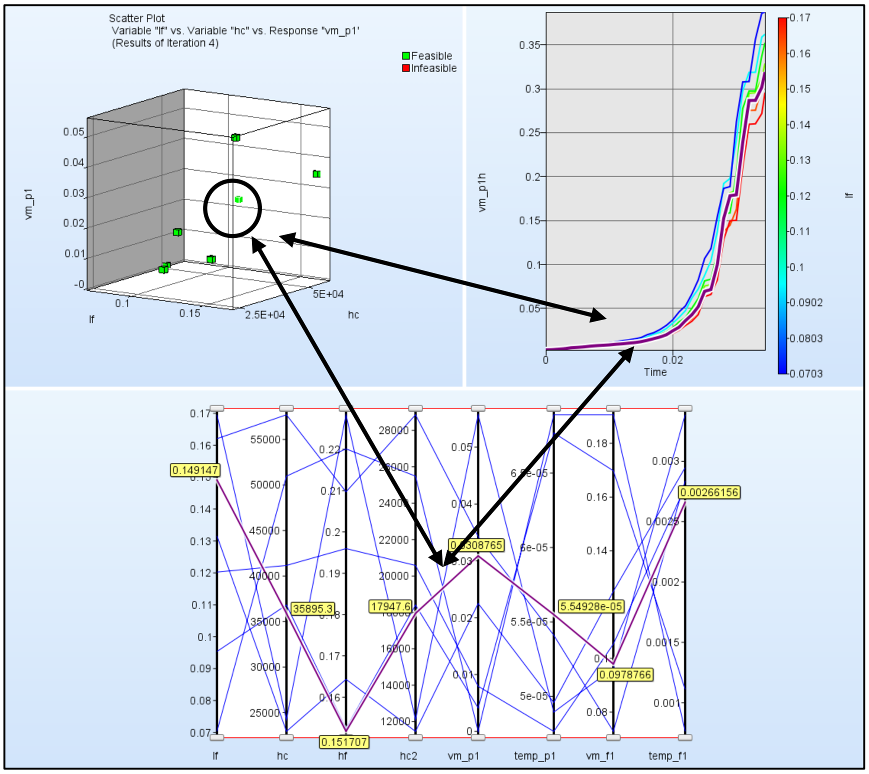


Figure 17-3: Cross-display of highlighted points in plots

Table 17-1: Point selection and table options

Option		Description
PP	Open selected run in Postprocessor	Opens selected run in LS-PrePost or any other user-defined postprocessor defined in Settings, Section 3.8.1. The job log of the marked run can also be viewed. LS-PrePost fringe plots are available for CurveMatching responses, Section 17.1.10. The folder of the marked run can be opened. Only enabled if only one point is marked.
➕	Add to set of selected points	Options for new point selection in plot. Points can be accumulated or subtracted.
➖	Subtract from set of selected points	
=	Replace set of selected points	
⧉	Toggle set of selected points (within rectangle)	

✖	Deselect all points	
↶	Undo	Undo last action
↷	Redo	Redo last action undone
📂	Export table data	Export the table data in different formats such as comma-delimited file, tab-delimited text-file etc., Section 17.1.11
A	All points mode	Show all points in the table
M	Marked mode	Show only plotted marked points
P	Plotted points mode	Show all plotted points
F	Filtered points mode	Show only points that satisfy current filters, Section 17.1.7
▤	Show colors	Show feasibility coloring when toggled
📊	Show statistics	Show statistical properties on the table when toggled. Currently shown points are used to calculate statistics.
▪▴	Manage categories	Define and manage point categories for user defined coloring or plotting of points, Section 17.1.5.
∞	Manage constraints	Set custom constraints for post-processing only, Section 17.1.6. Only enabled on User table.
▽	Quick filters	Show quick filtering menu, Section 17.1.7
▽	Manage filters	Show advanced filters dialog
⊘	Show or hide columns	Section 17.1.8
✦	Generate virtual points	Generate new (virtual) points using a selected sampling method. The response data is interpolated from the metamodel.
▶	Run virtual points	Launch new LS-OPT simulation on the marked virtual points
▦	Create new tab	Creates a new tab by copying the current one
▦	Remove current tab	
□	Clear table from virtual rows	

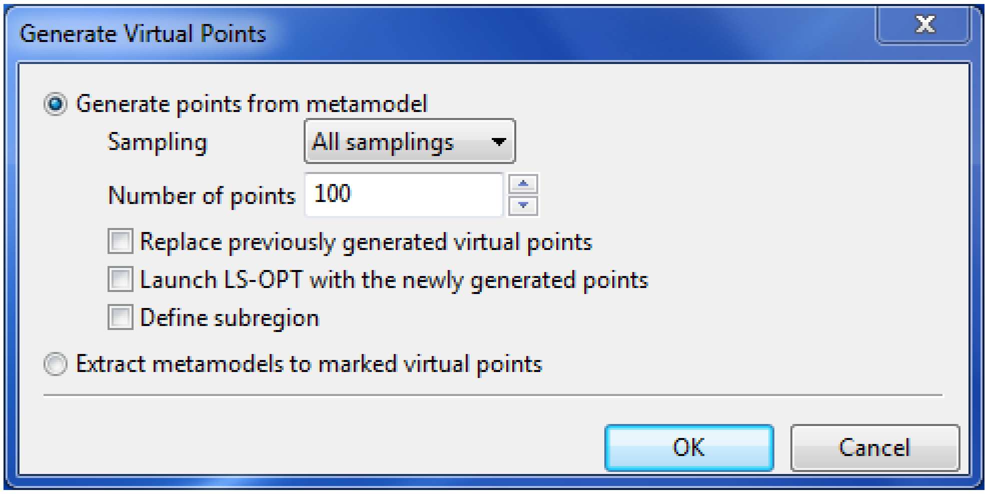
☆	Add new row	Add new virtual row
✖	Remove selected row(s)	Remove virtual row. Only enabled if all selected rows are virtual
⇆	Adjust column width to fit contents	
🔒	Unlock the table	The table is protected against editing values and rows operations. Click to unlock.
🔓	Lock the table	The table is fully editable. Click to lock.

17.1.2. Data Tables

A data table holds the point data for use when plotting points or to extract metamodel information in point data format. There are two types of tables; the *Main* table and *User* tables. The *Main* table is always available and it consists of the point data from the solver analysis runs. *User* tables can be added as user-defined tables from the toolbar button ‘Create new tab’ (⊞) and removed with the button ‘Remove current tab’ (⊞). The current set of tables is shown as tabs at the bottom of the point selection window as shown in Figure 17-2. Any additional table can be renamed by double-clicking the corresponding tab. The current table is selected by clicking the corresponding tab. If the option “Switch to selected table” is checked in the main toolbar (Figure 16-6) of a plot, all the plots will automatically switch to show point data from that table. A table maintains its own set of categorized points as explained in Section 17.1.5 and own set of filters. Category definitions however are shared across all tables.

17.1.3. Virtual Points and Point Generation

A virtual point is a user customizable point entry which can hold any set of values for custom post-processing scenarios. Virtual points can be added in an additional table using the *Add new row* icon (☆) from the control bar. To add a numerical value to a cell, the User table must be *unlocked* using the button (🔒) in the menu bar. Only User tables can be unlocked; the Main table always remains locked so that cells cannot be modified. Double-clicking the cells displays textfields and menus, respectively, to modify point data. If variable values are available, the metamodels can be evaluated for virtual points using the *Extract metamodels to marked virtual points* option from the Generate Virtual Points dialog, Figure 17-4.



Generate Virtual Points

● Generate points from metamodel

Sampling: All samplings

Number of points: 100

□ Replace previously generated virtual points

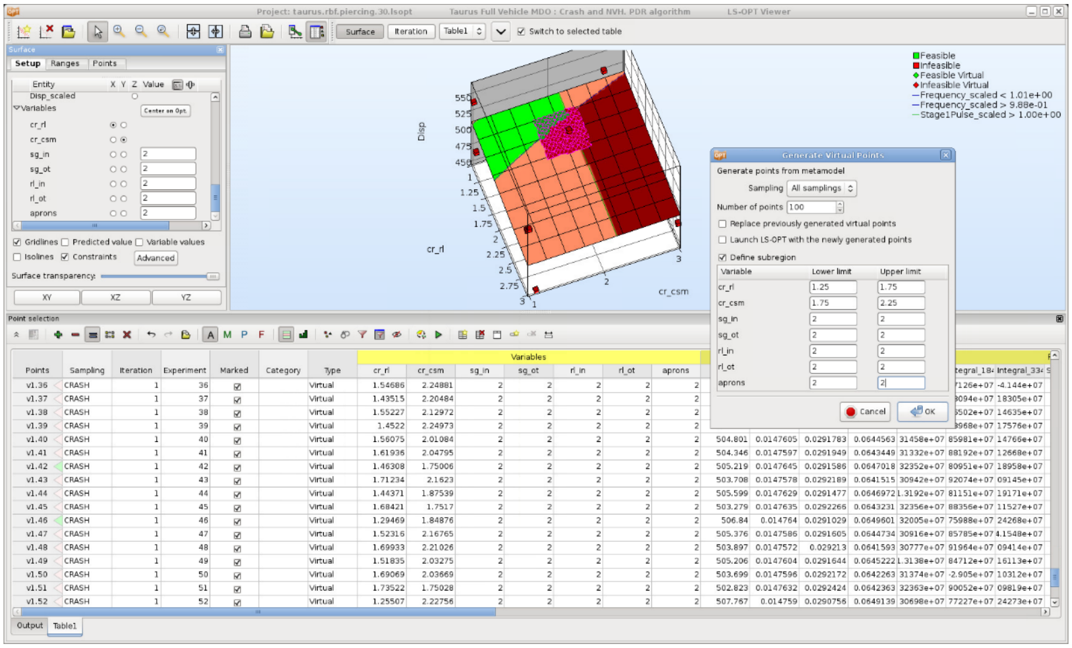
□ Launch LS-OPT with the newly generated points

□ Define subregion

○ Extract metamodels to marked virtual points

Figure 17-4: Generate virtual points dialog accessible from the toolbar button 'Generate virtual points' ( )

Virtual points can also be generated from a metamodel by clicking the toolbar button 'Generate virtual points' ( ), Figure 17-4.



Project: taurus.rbf.piercing.30.lsopt Taurus Full Vehicle MDO : Crash and NVH. PDR algorithm LS-OPT Viewer

☑ Switch to selected table

Surface: Setup | Ranges | Points

Entity	X	Y	Z	Value
Disp_scaled			○	
Variables				
cr_rl	●	○		
cr_csm	○	●		
sg_in	○	○		2
sg_ot	○	○		2
rl_in	○	○		2
rl_ot	○	○		2
aprons	○	○		2

☑ Gridlines □ Predicted value □ Variable values

□ Isolines ☑ Constraints

Surface transparency:

Generate Virtual Points

Generate points from metamodel

Sampling: All samplings

Number of points: 100

□ Replace previously generated virtual points

□ Launch LS-OPT with the newly generated points

☑ Define subregion

Variable	Lower limit	Upper limit
cr_rl	1.25	1.75
cr_csm	1.75	2.25
sg_in	2	2
sg_ot	2	2
rl_in	2	2
rl_ot	2	2
aprons	2	2

							Variables													
Points	Sampling	Iteration	Experiment	Marked	Category	Type	cr_rl	cr_csm	sg_in	sg_ot	rl_in	rl_ot	aprons						tegral_18	Integral_33
v1.36	CRASH	1	36	☑		Virtual	1.54686	2.24881	2	2	2	2	2						7126e+07	-4.144e+07
v1.37	CRASH	1	37	☑		Virtual	1.43515	2.20484	2	2	2	2	2						3094e+07	18305e+07
v1.38	CRASH	1	38	☑		Virtual	1.55227	2.12972	2	2	2	2	2						5502e+07	14635e+07
v1.39	CRASH	1	39	☑		Virtual	1.4522	2.24973	2	2	2	2	2						3968e+07	17576e+07
v1.40	CRASH	1	40	☑		Virtual	1.56075	2.01084	2	2	2	2	2	504.801	0.0147605	0.0291783	0.0644563	31458e+07	85981e+07	14766e+07
v1.41	CRASH	1	41	☑		Virtual	1.61936	2.04795	2	2	2	2	2	504.346	0.0147597	0.0291949	0.0643449	31332e+07	88192e+07	12668e+07
v1.42	CRASH	1	42	☑		Virtual	1.46308	1.75006	2	2	2	2	2	505.219	0.0147645	0.0291586	0.0647018	32352e+07	80951e+07	18958e+07
v1.43	CRASH	1	43	☑		Virtual	1.71234	2.1623	2	2	2	2	2	503.708	0.0147578	0.0292189	0.0641515	30942e+07	92074e+07	09145e+07
v1.44	CRASH	1	44	☑		Virtual	1.44371	1.87539	2	2	2	2	2	505.599	0.0147629	0.0291477	0.0646972	1.3192e+07	81151e+07	19171e+07
v1.45	CRASH	1	45	☑		Virtual	1.68421	1.7517	2	2	2	2	2	503.279	0.0147635	0.0292266	0.0643231	32356e+07	88356e+07	11527e+07
v1.46	CRASH	1	46	☑		Virtual	1.29469	1.84876	2	2	2	2	2	506.84	0.014764	0.0291029	0.0649601	32005e+07	75988e+07	24268e+07
v1.47	CRASH	1	47	☑		Virtual	1.52316	2.16765	2	2	2	2	2	505.376	0.0147586	0.0291605	0.0644734	30916e+07	85785e+07	4.1548e+07
v1.48	CRASH	1	48	☑		Virtual	1.69933	2.21026	2	2	2	2	2	503.897	0.0147572	0.029213	0.0641593	30777e+07	91964e+07	09414e+07
v1.49	CRASH	1	49	☑		Virtual	1.51835	2.03275	2	2	2	2	2	505.206	0.0147604	0.0291644	0.0645222	1.3138e+07	84712e+07	16113e+07
v1.50	CRASH	1	50	☑		Virtual	1.69069	2.03669	2	2	2	2	2	503.699	0.0147596	0.0292172	0.0642263	31374e+07	-2.905e+07	10312e+07
v1.51	CRASH	1	51	☑		Virtual	1.73522	1.75028	2	2	2	2	2	502.823	0.0147632	0.0292424	0.0642363	32363e+07	90052e+07	09819e+07
v1.52	CRASH	1	52	☑		Virtual	1.25507	2.22756	2	2	2	2	2	507.767	0.014759	0.0290756	0.0649139	30698e+07	77227e+07	24273e+07

Figure 17-5: Generate virtual points: the dialog is shown for generating the points and the figure shows generated points on a metamodel surface.

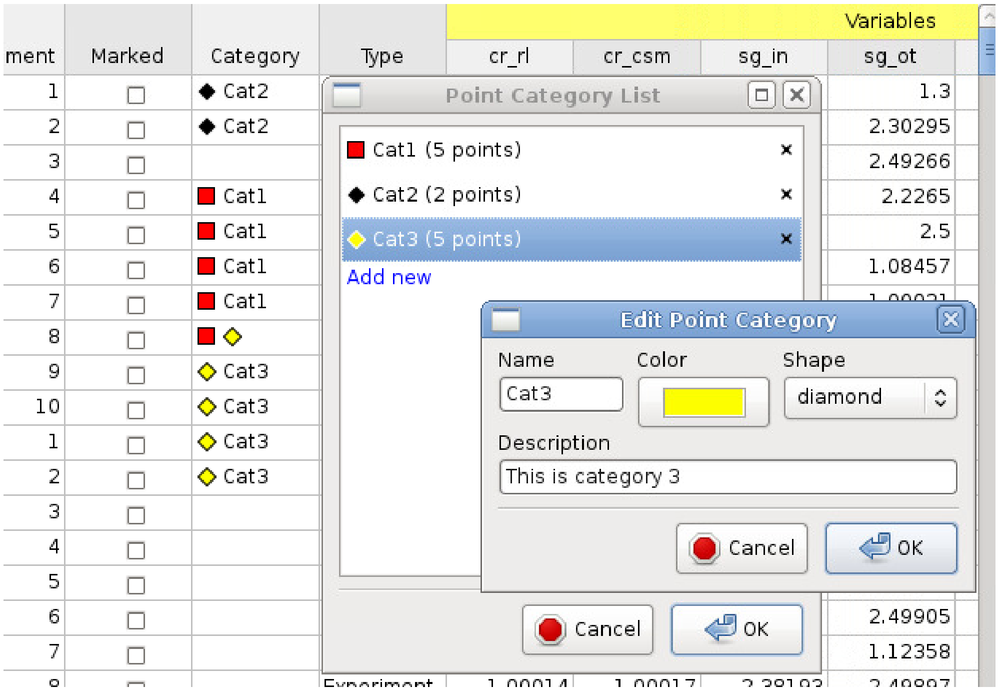
To generate the points using the metamodel, the sampling and the number of required virtual points should be defined. As a default, the points are generated across the original design bounds of the variables; however, the points could also be generated for a sub region using 'Define sub region' checkbox. The Generate virtual points dialog is shown in Figure 17-5. An example is given in Section 17.1.11. New points are always exclusively marked to allow operations on these points.

17.1.4. Run Virtual Points

Virtual points can be used to extend the current LS-OPT run. When the points in the virtual rows are marked, the toolbar button ‘Run virtual points (▶) is enabled. Clicking this button will launch the main GUI where a confirmation of the action is presented after which the current project is extended so that the number of points per iteration is incremented by the number of virtual points. The LS-OPT engine is run to execute solver jobs for the new points and extract corresponding histories, response values, etc. The full cycle of the LS-OPT task is executed by augmenting the design with the newly generated virtual points. The viewer will automatically replace the virtual points with Analysis points when they are detected in LS-OPT output files. It is also possible to run virtual points directly from ‘Generate virtual points’ dialog by checking the ‘*Launch LS-OPT with the newly generated points’* option.

17.1.5. User-defined Categorization of Points

Points can be categorized in the Point Categories dialog, Figure 17-6. Point Categories can be used for **User-defined** coloring and plotting of points in the Scatter Plot (Section 16.2.2) and the Parallel Coordinate Plot (Section 16.2.3). The dialog is accessible from the Point Selection window, Section 16.2.



							Variables
ment	Marked	Category	Type	cr_rl	cr_csm	sg_in	sg_ot
1	□	◆ Cat2					1.3
2	□	◆ Cat2					2.30295
3	□						2.49266
4	□	🟥 Cat1					2.2265
5	□	🟥 Cat1					2.5
6	□	🟥 Cat1					1.08457
7	□	🟥 Cat1					1.00021
8	□	🟥 ◇					
9	□	◇ Cat3					
10	□	◇ Cat3					
1	□	◇ Cat3					
2	□	◇ Cat3					
3	□						
4	□						
5	□						
6	□						2.49905
7	□						1.12358
8	□		Experiment	1.00014	1.00017	2.38193	2.49897

Point Category List

- 🟥 Cat1 (5 points) ×
- ◆ Cat2 (2 points) ×
- ◇ Cat3 (5 points) ×

Add new

Edit Point Category

Name: Cat3
Color: 🟨
Shape: diamond
Description: This is category 3

Figure 17-6: Point Categories dialog to define and manage Point Categories for user-defined coloring and plotting of points. Note that a design point can belong to multiple categories.

From the toolbar button ***Manage Categories*** ( ), the Point Category List is displayed. A new category can be added by clicking the *Add new* option and the existing categories can be modified by double clicking a category. Clicking on the ✕ removes a category.

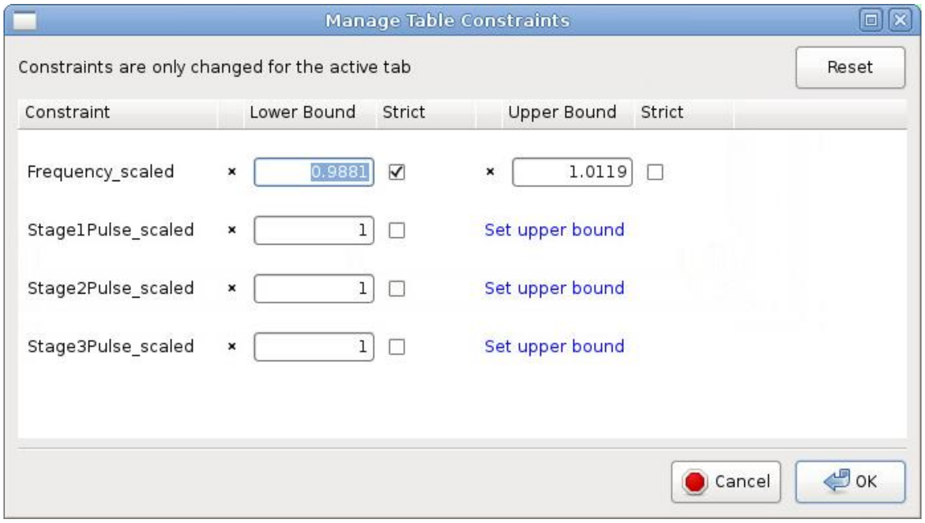
In the ‘Edit Point Category’ dialog the attributes *Name*, *Color*, *Shape* and *Description* can be modified. Spaces are allowed in the *Name*. Press *OK* to save changes to a category and category list or *Cancel* to abort any changes. Points can be assigned to a category by either double clicking a category cell in the table, where a drop-down selection is made, or by selecting multiple rows, right click in the table and select *Assign category* in the menu presented.

If the categories are not predefined, it is also possible to assign points directly using the ‘new category’ option from the right click menu, where the ‘Edit Point Category’ dialog is presented. Any point can be assigned multiple categories and the category column of the table shows corresponding markers used for the multiple categories. The category definitions are stored in XML format in the file *CategorizedPoints.lsox* and the categorized points are stored in the LSDA binary formatted file viewerstate.bin in the project folder.

Categories of points can be displayed using the *User-defined* selection (see Table 16-5, Table 16-6) and/or color-coded using the *User-defined* selection (see Table 16-4, Table 16-6) in Scatter plot and Parallel Coordinate Plot, respectively.

17.1.6. Customizable Constraints

From an additional table, the ‘Manage Constraints’ ( ) toolbar button is enabled which opens the ‘Manage Table Constraints’ dialog shown below. This dialog allows redefinition of constraints in a similar way as in the project setup in the main GUI. Currently, only responses and/or composites set as constraints in the project setup are allowed to be manipulated.



Manage Table Constraints

Constraints are only changed for the active tab

Constraint		Lower Bound	Strict		Upper Bound	Strict
Frequency_scaled	×	0.9881	☑	×	1.0119	□
Stage1Pulse_scaled	×	1	□		Set upper bound	
Stage2Pulse_scaled	×	1	□		Set upper bound	
Stage3Pulse_scaled	×	1	□		Set upper bound	

Figure 17-7: Customizable constraints for post-processing scenarios

Since the modified constraints affect the maximum constraint violation values, the feasibility calculations of the current table are updated accordingly. Any plot that shows data from the same data table will also show constraints and feasibility according to these custom values. The ‘Reset’ option can be used to switch the constraint bounds back to the original bounds used when the project was run.

17.1.7. Point Filtering

The Point Selection window has three toolbar buttons for filtering. The first is the ‘Show filtered rows’ mode (F) used to switch to a mode where only rows that satisfy current filters are shown, or all rows if there are no predefined filters. The toolbar buttons ‘Quick filters ( ) and ‘Manage filters’ ( ) manipulate the current set of filters. ‘Quick filters’ show a menu to quickly toggle between common single filters such as feasible and infeasible points, points from a certain sampling, iteration or category, etc. The currently selected filter will be shown with a checkmark.

More advanced filters can be defined using ‘Manage filters’ toolbar button or using ‘Advanced…’ option under ‘Quick filters’ menu. The options shown under ‘Quick filters’ menu can be used to filter points using one specific criterion, whereas advanced filters facilitate the use of multiple criteria. Using ‘Add new’ option in ‘Table filters’ dialog, multiple filters can be applied to the data set to filter points based on sampling, category, specific column values, etc., as shown in the figure 17-7. If multiple filters are specified in the Table Filters dialog, all of them are used to filter the rows. For example, to filter feasible points within a specific range of response value, a feasibility filter and a ‘column value’ filter can be defined by selecting the response and specifying the lower and upper bounds of the range. However, please note that a selection from the ‘Quick filters’ menu replaces all definitions in Table Filters dialog.

The available types of filters are described in Table 17-2.

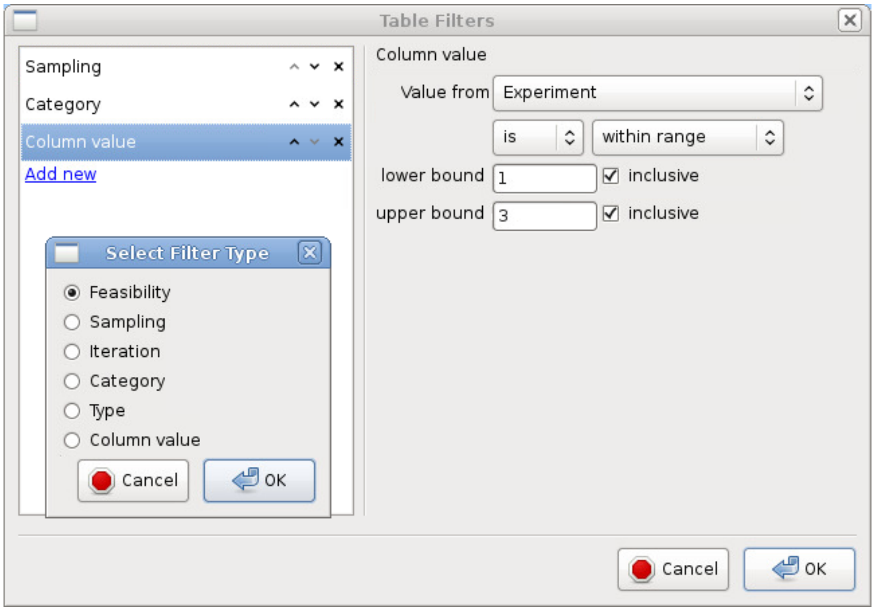


Table Filters

Sampling
Category
Column value
Add new

Column value
Value from Experiment
is within range
lower bound 1 ☑ inclusive
upper bound 3 ☑ inclusive

Select Filter Type
● Feasibility
○ Sampling
○ Iteration
○ Category
○ Type
○ Column value

Figure 17-8: Manage filters dialog with 3 filters defined and a selection of new filter options.

Table 17-2: Point filters allowed in a table

Name	Description
Feasibility	Show feasible or infeasible points
Sampling	Show points with no sampling or in a selected set of samplings
Iteration	Show points within the selected iteration(s).
Type	Show points in a selected set of possible types (Analysis, Tradeoff etc…)
Category	Show points based on selected point categories

Column value	Show points where a certain row value satisfies some selected criteria. For instance, if the value is or is not equal to, larger than, within range etc.

17.1.8. Show or Hide Columns

Clicking the toolbar button ‘Show or hide columns’ (👁) shows the dialog shown in Figure 17-9. This dialog can be used to manipulate the columns displayed in the point selection window. The columns already displayed in the point selection window are shown in the left list and hidden columns are shown in the right list. Columns that are grouped in a certain type, for instance Variables, are shown under the group name.

Columns or whole groups can be switched from either visible or hidden list to the other list by double clicking the column name or the group name. There are also ‘Quick action’ options to quickly show or hide all PRESS data, predicted, computed and responses for certain stages to the far right.

The visible rows can be reset using the ‘Default’ option shown in the dialog box.

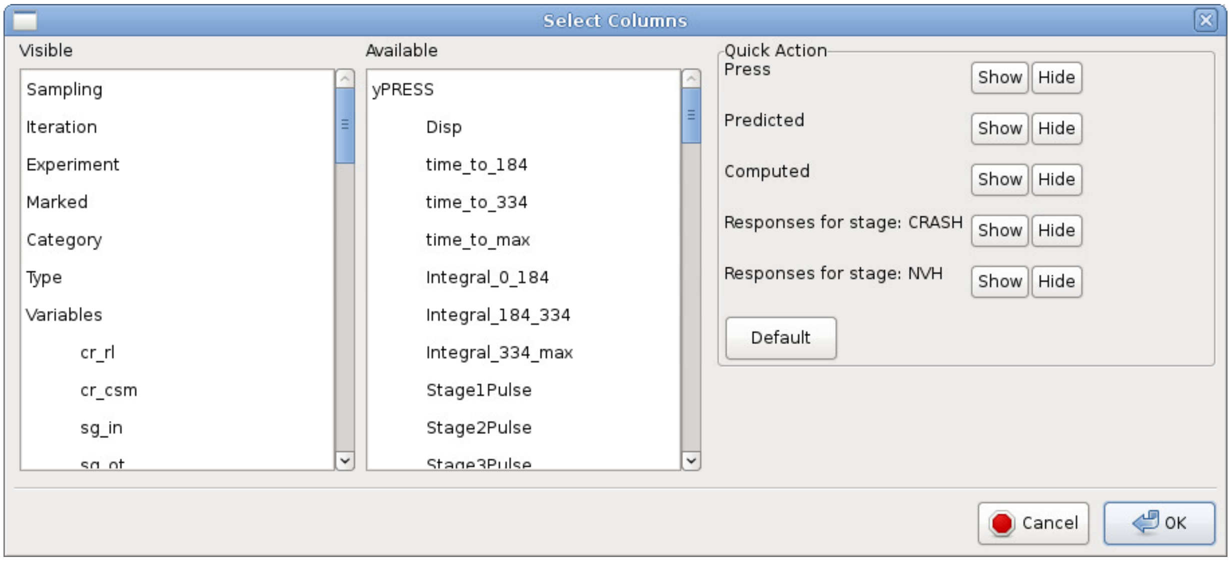


Figure 17-9: Show or hide columns dialog

17.1.9. Copy and Paste

Additionally to copying and pasting rows or ranges of cells within a table or from one table to another, interaction with programs like Excel or Libreoffice is possible.

A single row or a range of cells can be selected in any table, copied using the right mouse menu *copy* option or ctrl-c, and pasted in Excel/Libreoffice.

The same can be done from Excel/Libreoffice to a table. To paste a range into a table, create a new tab or use an already existing *User* table and unlock it. Create as many virtual rows as you want to paste data into. Copy your data from Excel/Libreoffice, select the top-left corner where data should be pasted and press ctrl-v or select the *paste* option from the right mouse menu.

Remarks

1. The range pasted into tables must be simple, without holes and the data type must match.
2. The row type is ignored when pasting back to tables, it is always added as a virtual row.
3. Sampling and row type is copied as text.
4. Boolean data is copied as values "true" or "false".
5. Category is copied as integer.
6. Everything else is copied as-is (double or int).

17.1.10. LS-PrePost Fringe Plots

For calibration problems using full-filed data, Section 28.4, LS-PrePost contour plots are available to visualize the difference between the simulation and target values superimposed on the Finite Element model. Both x- and y- components are available for the contour displays.

Simulation curves are typically constructed using a crossplot of $x(t)$ and $y(t)$ components to obtain $y(x)$. This value of t is used as *time* for mapping a simulation curve to test curve. Since the target curve typically does not have a time component (or the time is on an independent scale), the similarity measure is used to obtain a corresponding test point for each simulation point. The mapping rule depends on the *path* of the similarity measure. The path is defined as the point pairs which define the mapping (displayed as the red *connectors*, see Figure 17-10, for a DTW mapping). Note that for DTW, the mapping is not necessarily one-to-one, but can be one-to-many or many-to-one. The Mean Squared Error Mapping is one-to-one.

The contour plotting options consist of the x- and y-values for both curves as well as the differences Δx and Δy between the curves. These options are respectively labeled `Target x`, `Computed x`, `Difference x`, and similar for y.

In order to obtain all the contour values, a temporal mapping (using t as in Figure 17-10) is conducted at each node, using the nearest test point. Since the contour only has non-zero plot values in the region of the test query points, the remaining part of the FE mesh should have a uniform color.

(See also Section 28.2.3 for the theory of Dynamic Time Warping (DTW)).

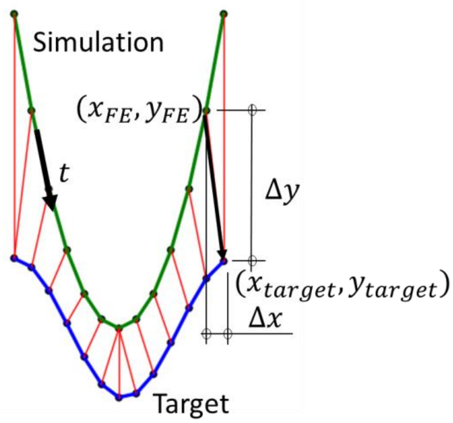


Figure 17-10: Contour mapping (Dynamic Time Warping example)

17.1.11. Export table data

The table data can be exported in different formats such as comma-delimited file, tab-delimited text-file etc., Table 17-3. If For use as user-defined sampling or import analysis results with Export Format Comma-separated values is selected, the output file has the format of ExtendedResults and can be reused in LS-OPT. The columns that are not needed by the import feature are ignored on import.

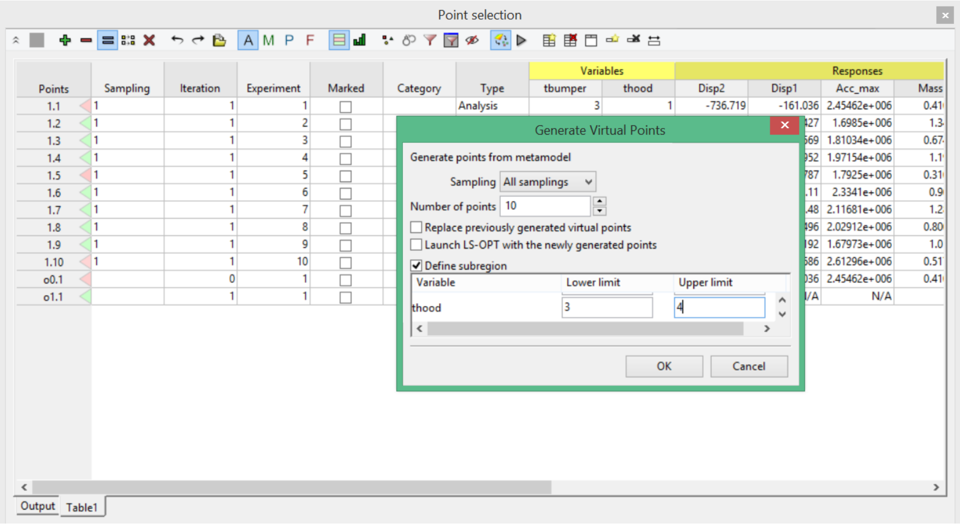
Table 17-3: Export options

Option	**Description**
Export usage	For use as user-defined sampling or import analysis results: ExtendedResults csv format Headers and values as shown in table: output contains all values and headers in selected export format
Export Format	Comma-separated values (CSV) Tab-separated values OpenDocument Spreadsheet (ODS)

17.1.12. Example: Generating virtual points to augment the design

The following example outlines one of the post-processing features available in the data table of the point selection window. The example setup consists of a simple car crash optimization problem with two thickness variables and five design responses with head injury criteria (HIC) as an objective and intrusion distance as a design constraint. A single iteration, metamodel-based optimization is run with 10 sampling points, selected using the Space Filling sampling technique. Steps:

1. The point selection window can be accessed by clicking on a design point from scatter, surface or parallel coordinate plots, etc. Before generating the virtual points, a new user table should be created using the ‘create new tab’ () button. In this example, for demonstration purpose, 10 new points are generated in a sub region of the original design space. The ‘Generate Virtual Points’ dialog, accessed through the button (), is shown in the figure below.



Point selection

Points	Sampling	Iteration	Experiment	Marked	Category	Type	tbumper	thood	Disp2	Disp1	Acc_max	Mass
							Variables		Responses			
1.1	1	1	1	☐		Analysis	3	1	-736.719	-161.036	2.45462e+006	0.41
1.2	1	1	2	☐						427	1.6985e+006	1.3
1.3	1	1	3	☐						569	1.81034e+006	0.67
1.4	1	1	4	☐						952	1.97154e+006	1.1
1.5	1	1	5	☐						787	1.7925e+006	0.31
1.6	1	1	6	☐						.11	2.3341e+006	0.9
1.7	1	1	7	☐						.48	2.11681e+006	1.2
1.8	1	1	8	☐						496	2.02912e+006	0.80
1.9	1	1	9	☐						192	1.67973e+006	1.0
1.10	1	1	10	☐						586	2.61296e+006	0.51
o0.1		0	1	☐						036	2.45462e+006	0.41
o1.1		1	1	☐						/A	N/A	

Generate Virtual Points

Generate points from metamodel

Sampling: All samplings

Number of points: 10

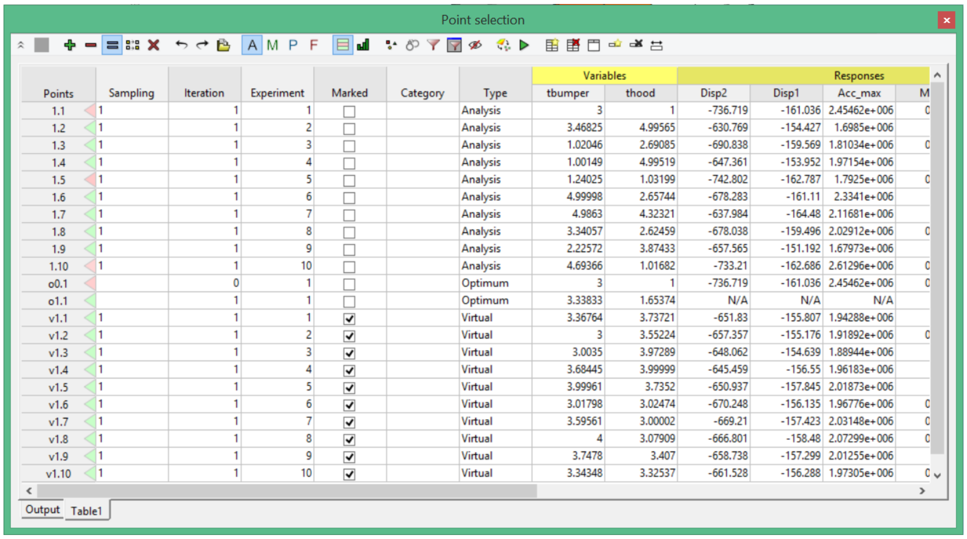
☐ Replace previously generated virtual points

☐ Launch LS-OPT with the newly generated points

☑ Define subregion

Variable	Lower limit	Upper limit
thood	3	4

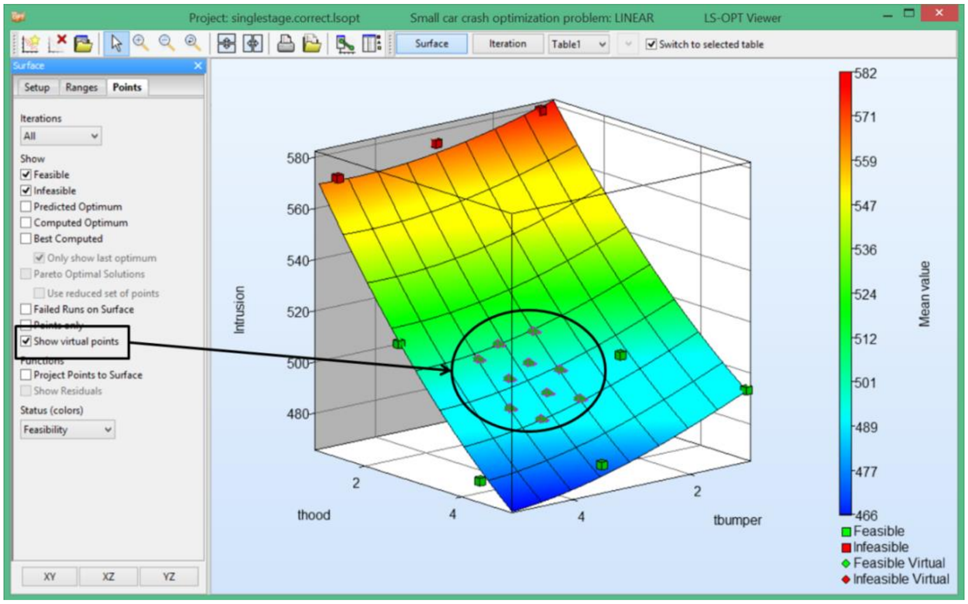
2. The point selection window with the new user table (Table1) and the newly generated virtual points is shown in the figure below. The virtual points are sampled using space filling sampling and the responses and composite values are evaluated using the metamodel generated in the original run.



Point selection

Points	Sampling	Iteration	Experiment	Marked	Category	Type	tbumper	thood	Disp2	Disp1	Acc_max	M
							Variables		Responses			
1.1	1	1	1	☐		Analysis	3	1	-736.719	-161.036	2.45462e+006	0
1.2	1	1	2	☐		Analysis	3.46825	4.99565	-630.769	-154.427	1.6985e+006	
1.3	1	1	3	☐		Analysis	1.02046	2.69085	-690.838	-159.569	1.81034e+006	0
1.4	1	1	4	☐		Analysis	1.00149	4.99519	-647.361	-153.952	1.97154e+006	
1.5	1	1	5	☐		Analysis	1.24025	1.03199	-742.802	-162.787	1.7925e+006	0
1.6	1	1	6	☐		Analysis	4.99998	2.65744	-678.283	-161.11	2.3341e+006	
1.7	1	1	7	☐		Analysis	4.9863	4.32321	-637.984	-164.48	2.11681e+006	
1.8	1	1	8	☐		Analysis	3.34057	2.62459	-678.038	-159.496	2.02912e+006	0
1.9	1	1	9	☐		Analysis	2.22572	3.87433	-657.565	-151.192	1.67973e+006	
1.10	1	1	10	☐		Analysis	4.69366	1.01682	-733.21	-162.686	2.61296e+006	0
o0.1		0	1	☐		Optimum	3	1	-736.719	-161.036	2.45462e+006	0
o1.1		1	1	☐		Optimum	3.33833	1.65374	N/A	N/A	N/A	
v1.1	1	1	1	☑		Virtual	3.36764	3.73721	-651.83	-155.807	1.94288e+006	
v1.2	1	1	2	☑		Virtual	3	3.55224	-657.357	-155.176	1.91892e+006	0
v1.3	1	1	3	☑		Virtual	3.0035	3.97289	-648.062	-154.639	1.88944e+006	
v1.4	1	1	4	☑		Virtual	3.68445	3.99999	-645.459	-156.55	1.96183e+006	
v1.5	1	1	5	☑		Virtual	3.99961	3.7352	-650.937	-157.845	2.01873e+006	
v1.6	1	1	6	☑		Virtual	3.01798	3.02474	-670.248	-156.135	1.96776e+006	0
v1.7	1	1	7	☑		Virtual	3.59561	3.00002	-669.21	-157.423	2.03148e+006	0
v1.8	1	1	8	☑		Virtual	4	3.07909	-666.801	-158.48	2.07299e+006	0
v1.9	1	1	9	☑		Virtual	3.7478	3.407	-658.738	-157.299	2.01255e+006	
v1.10	1	1	10	☑		Virtual	3.34348	3.32537	-661.528	-156.288	1.97305e+006	0

3. The generated virtual points can be plotted on the surface plots, as shown in the figure below. Since the virtual points were evaluated using the metamodel, the points lie on the surface of the plot. The sampling points for which LS-DYNA analyses were performed are shown using square markers and the virtual points are represented by diamond markers.



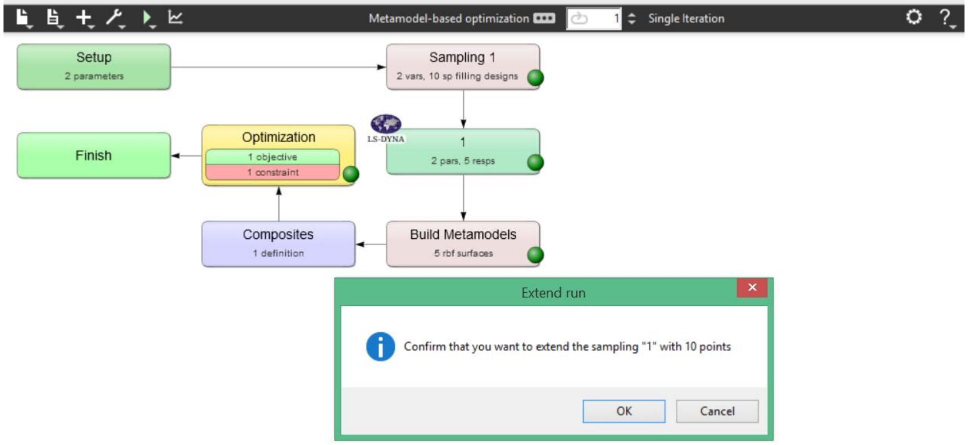
Iterations
All

Show
☑ Feasible
☑ Infeasible
□ Predicted Optimum
□ Computed Optimum
□ Best Computed
☑ Only show last optimum
□ Pareto Optimal Solutions
□ Use reduced set of points
□ Failed Runs on Surface
□ Points only
☑ Show virtual points

Functions
□ Project Points to Surface
□ Show Residuals

Status (colors)
Feasibility

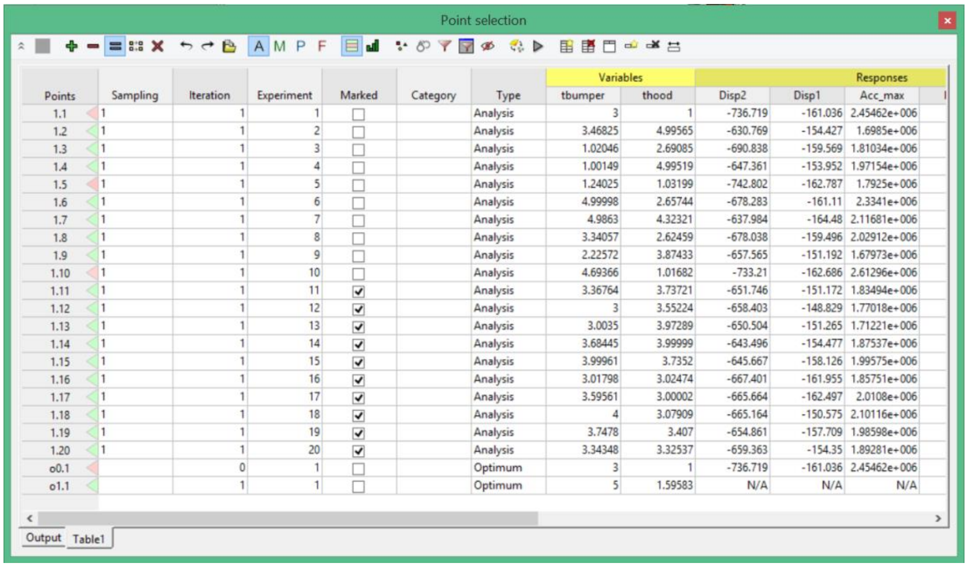
4. To run the solver analysis for the virtual points, ‘mark’ all the virtual points. This can be done by selecting all the virtual points (using shift key), then select ‘Mark -> add’ from the right click menu. Please note that the points can also be marked directly from the scatter plot, surface plot or parallel coordinate plot, etc. either by clicking on a point or using rectangular selection (rubber-banding) for marking multiple points. Once the points are marked, use the run virtual points button (▶) to confirm and run the solver jobs for the virtual points. The original sampling will be extended using the virtual points, as shown in the figure below.



Confirm that you want to extend the sampling "1" with 10 points

5. If the above action is confirmed, LS-OPT will augment the design with the newly generated virtual points and submit the solver jobs for the new points. Once all the new jobs are completed, LS-OPT will go through the rest of the task cycle automatically i.e. extracting the results for the new jobs, building the metamodels using the updated 20 design points and running optimization using the new metamodel.

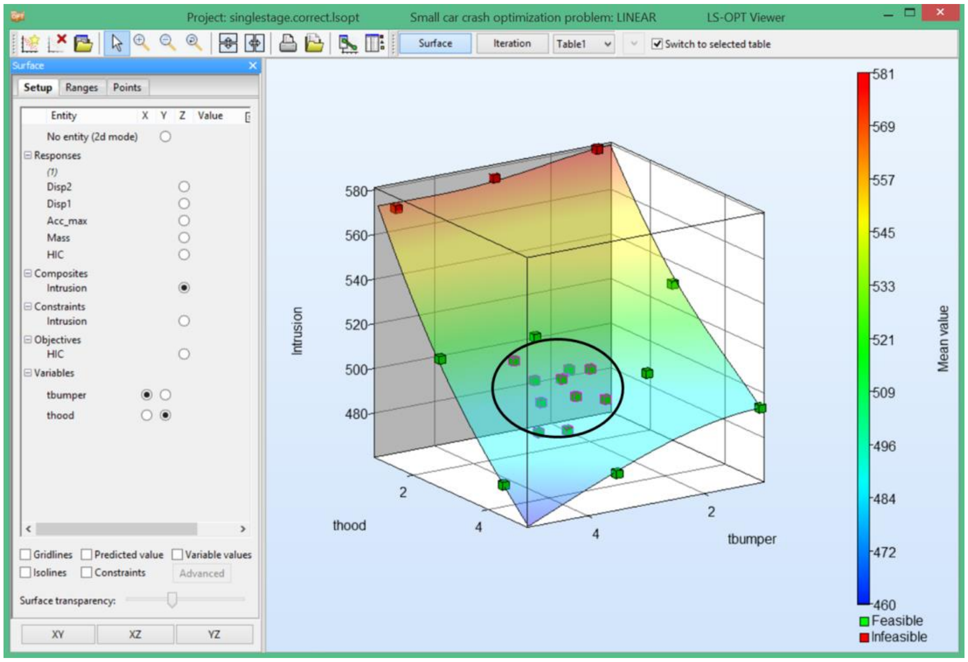
When the point selection window is opened again, the ‘virtual points’ of the user table are automatically replaced by ‘analysis points’, as shown in the figure below.



Point selection

Points	Sampling	Iteration	Experiment	Marked	Category	Type	Variables		Responses		
							tbumper	thood	Disp2	Disp1	Acc_max
1.1	1	1	1	□		Analysis	3	1	-736.719	-161.036	2.45462e+006
1.2	1	1	2	□		Analysis	3.46825	4.99565	-630.769	-154.427	1.6985e+006
1.3	1	1	3	□		Analysis	1.02046	2.69085	-690.838	-159.569	1.81034e+006
1.4	1	1	4	□		Analysis	1.00149	4.99519	-647.361	-153.952	1.97154e+006
1.5	1	1	5	□		Analysis	1.24025	1.03199	-742.802	-162.787	1.7925e+006
1.6	1	1	6	□		Analysis	4.99998	2.65744	-678.283	-161.11	2.3341e+006
1.7	1	1	7	□		Analysis	4.9863	4.32321	-637.984	-164.48	2.11681e+006
1.8	1	1	8	□		Analysis	3.34057	2.62459	-678.038	-159.496	2.02912e+006
1.9	1	1	9	□		Analysis	2.22572	3.87433	-657.565	-151.192	1.67973e+006
1.10	1	1	10	□		Analysis	4.69366	1.01682	-733.21	-162.686	2.61296e+006
1.11	1	1	11	☑		Analysis	3.36764	3.73721	-651.746	-151.172	1.83494e+006
1.12	1	1	12	☑		Analysis	3	3.55224	-658.403	-148.829	1.77018e+006
1.13	1	1	13	☑		Analysis	3.0035	3.97289	-650.504	-151.265	1.71221e+006
1.14	1	1	14	☑		Analysis	3.68445	3.99999	-643.496	-154.477	1.87537e+006
1.15	1	1	15	☑		Analysis	3.99961	3.7352	-645.667	-158.126	1.99575e+006
1.16	1	1	16	☑		Analysis	3.01798	3.02474	-667.401	-161.955	1.85751e+006
1.17	1	1	17	☑		Analysis	3.59561	3.00002	-665.664	-162.497	2.0108e+006
1.18	1	1	18	☑		Analysis	4	3.07909	-665.164	-150.575	2.10116e+006
1.19	1	1	19	☑		Analysis	3.7478	3.407	-654.861	-157.709	1.98598e+006
1.20	1	1	20	☑		Analysis	3.34348	3.32537	-659.363	-154.35	1.89281e+006
o0.1		0	1	□		Optimum	3	1	-736.719	-161.036	2.45462e+006
o1.1		1	1	□		Optimum	5	1.59583	N/A	N/A	N/A

6. The following figure highlights the new analysis points on the metamodel surface generated using the augmented design. The virtual points have been replaced by the response values of analysis points, obtained through LS-DYNA runs.



18. LS-DYNA Results Statistics

The statistics of the LS-DYNA results can be displayed on the FE model using *DynaStats*. The statistics of the LS-DYNA d3plot (or d3eigv) results and LS-OPT history data are computed by LS-OPT for viewing in LS-PREPOST. These statistics shows:

- The variation of the LS-DYNA results due to the variation of the design parameters.
- The variation of the LS-DYNA results due to bifurcations and other stochastic process events.

The d3plot results are computed and displayed for every node or element for every state in the d3plot database, while the history results are likewise computed and displayed for every time state in the history (in history.x file).

A more complete list of the statistics that can be computed and visualized is:

1. Statistics of the **Monte Carlo** data from the LS-DYNA jobs. These are the data from the experimental designs used. If the experimental design was for a Monte Carlo analysis then the experimental design reflects the variation of the design variables, but if the experimental design was for creating a metamodel then the experimental design does not reflect the statistical variation of the design variables.
2. Statistics of the results considering the variation of the design variables using the approximations (**metamodels**) created from the LS-DYNA jobs. It should be noted that these approximations differ from the ones defined for the responses under “Metamodeling” dialog of LS-OPT. In order to display statistics over the entire LS-DYNA model, several metamodels need to be fitted (for every element/node). Therefore, only linear and quadratic metamodeling options are available under *DynaStats* to make the computation fast. The distributions of the design variables and the metamodels are used to compute the variation of the responses. If distributions were not assigned to the design variables, the resulting variation will be zero. The metamodels allow the computations of the following:
   - The deterministic or parametric variation of the responses caused by the variation of the design variables.
   - Statistics of the residuals from the metamodels created from the LS-DYNA jobs. These residuals are used to find bifurcations in the structural behavior – the outliers comprise the displacement changes not associated with a design variable change. See Section 29.6 regarding the computation of outliers. This is the process variation associated with structural effects such as bifurcations and not with changes in the design variable values.
   - The stochastic contribution of a variable can be investigated.
   - A probabilistic safety margin with respect to a bound on the LS-DYNA response can be plotted.

- The LS-OPT histories of all the LS-DYNA runs as well as history statistics can be plotted.

3. The correlation of d3plot results or histories with an LS-OPT response can be displayed. This can be used, for example, to identify the changes in displacements associated with noise in an LS-OPT response.

18.1. Working with the plots

Select the *DynaStats* option from the *Tools* menu of the control bar of the main GUI. The dialog shown in Figure 18-1 opens up to work with the plots. Utilize the following actions:

- *Create* This creates a new plot. Note that this only creates the definition of the plot. The data for the plot must be generated before it can be displayed. The options are described in Section 18.2.
- *Generate* The data for a plot is generated. This is done only once per plot. More than one plot can be selected to be generated – there is no need to generate plots one-by-one.
- *Display* Plot previously created and generated can be displayed.
- *Edit* A plot can be edited or copied. This may require that the data be re-generated.
- *Bifurcation* A study can be investigated for bifurcations, and the bifurcation can be plotted.
- *Delete* A plot can be deleted.

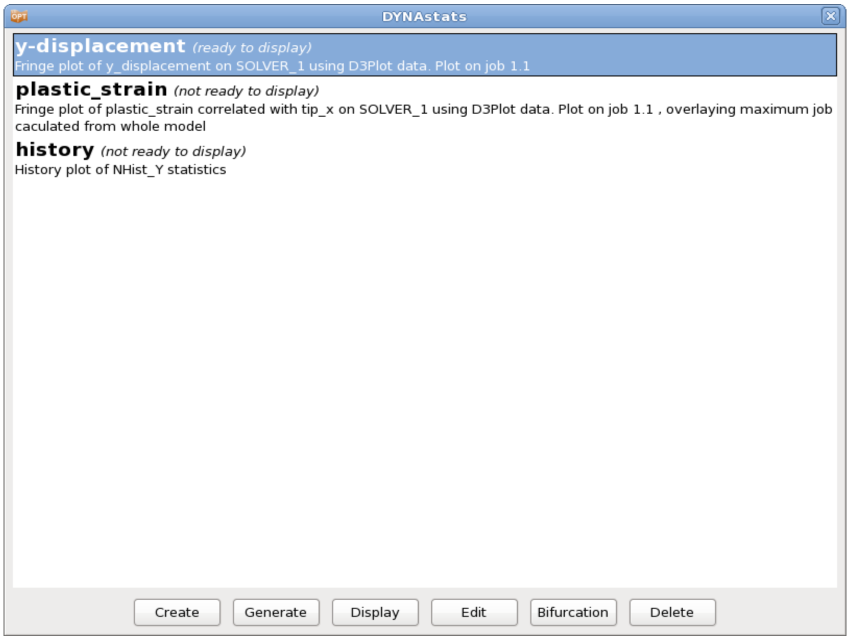
The plot definitions are stored in a file named *dynastatplots.xml* which allows re-use of a methodology in different studies (see Section 18.12).

18.2. Creation of a plot

A plot is created in four steps.

18.2.1. Step 1 – Fringe plot or History plot

In the first step, the user has to select whether to create a fringe plot or a history plot, Figure 18-2. Select the respective image to go to the next step.

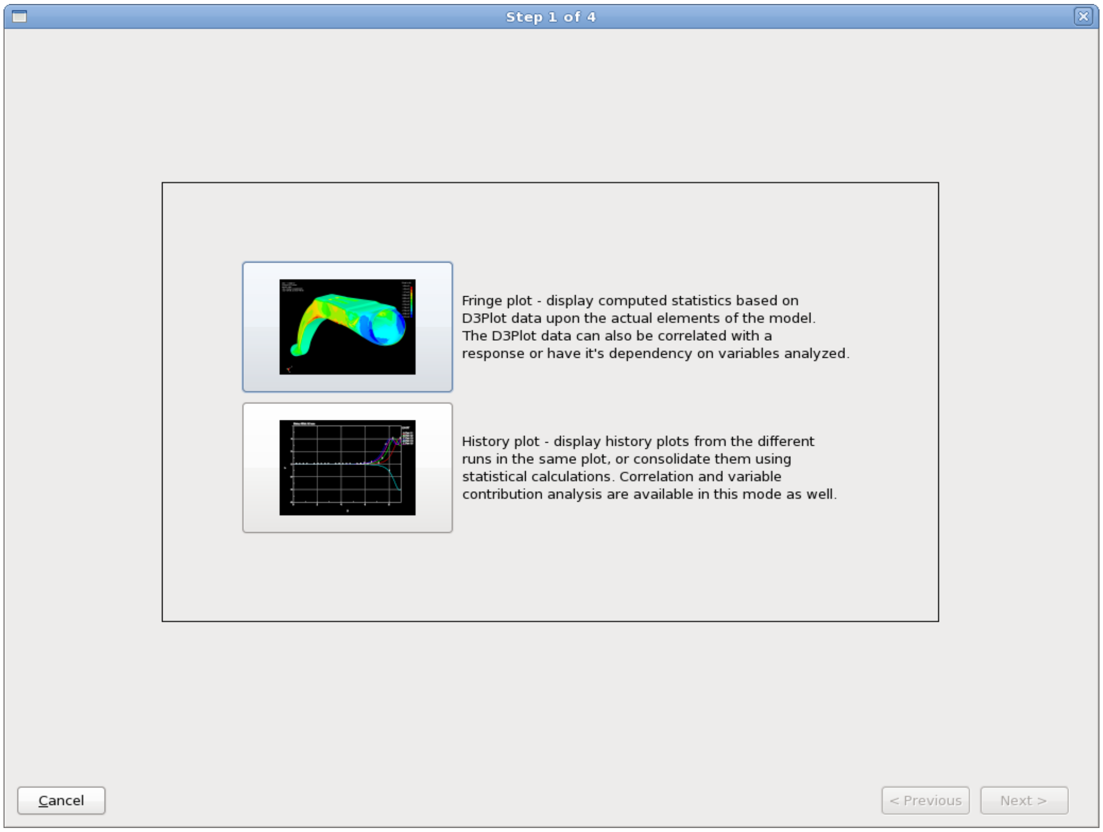


y-displacement *(ready to display)*
Fringe plot of y_displacement on SOLVER_1 using D3Plot data. Plot on job 1.1

plastic_strain *(not ready to display)*
Fringe plot of plastic_strain correlated with tip_x on SOLVER_1 using D3Plot data. Plot on job 1.1 , overlaying maximum job caculated from whole model

history *(not ready to display)*
History plot of NHist_Y statistics

Figure 18-1: Visualization of LS-DYNA results statistics. After plot creation using the wizard, the plot data must be generated. The plot can then be displayed in LS-PREPOST. Existing plots can be edited, deleted or investigated for a bifurcation.



Fringe plot - display computed statistics based on D3Plot data upon the actual elements of the model. The D3Plot data can also be correlated with a response or have it's dependency on variables analyzed.

History plot - display history plots from the different runs in the same plot, or consolidate them using statistical calculations. Correlation and variable contribution analysis are available in this mode as well.

Figure 18-2: First step of DynaStats plot definition creation; selection of plot type

18.2.2. Step 2 – D3Plot component or History

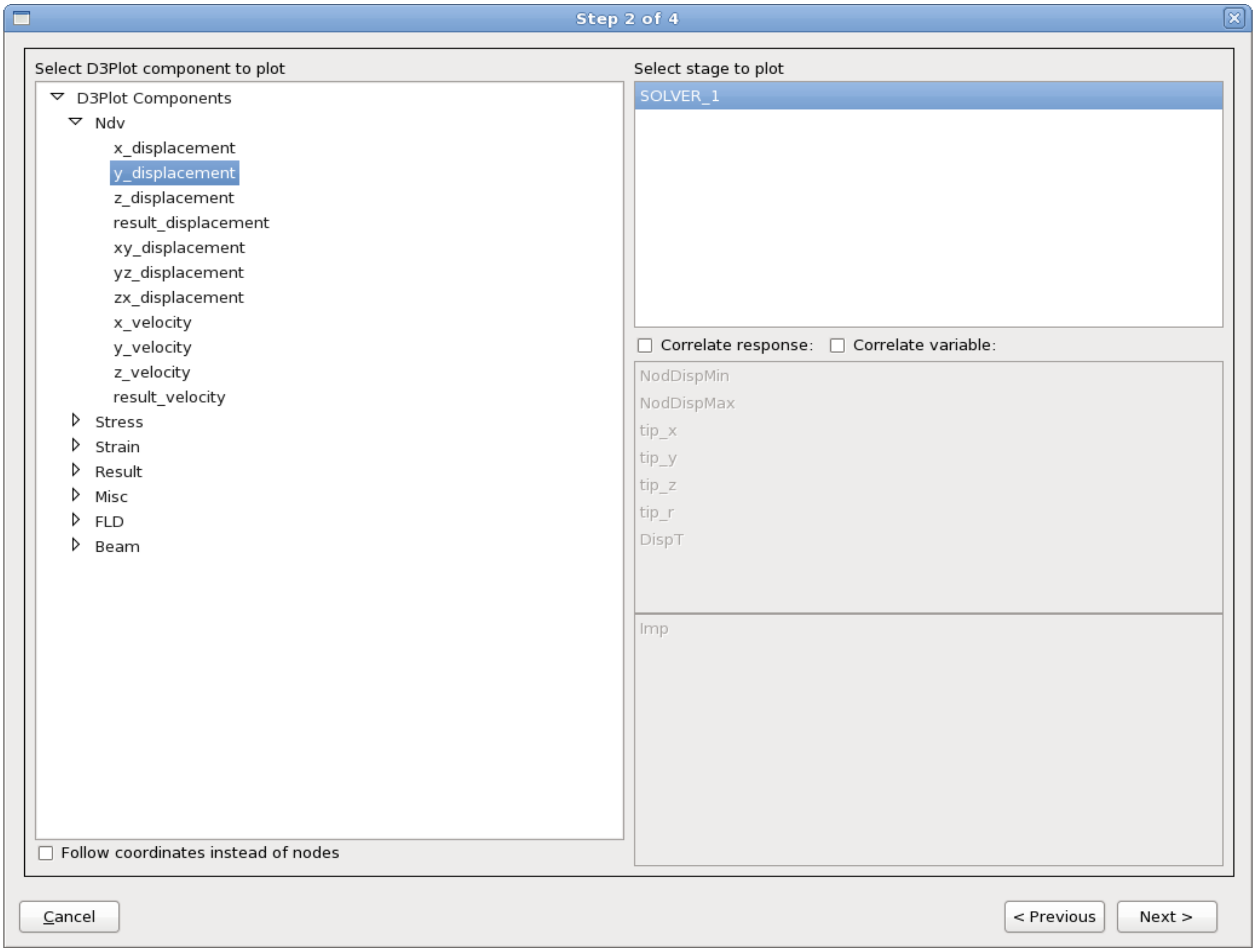


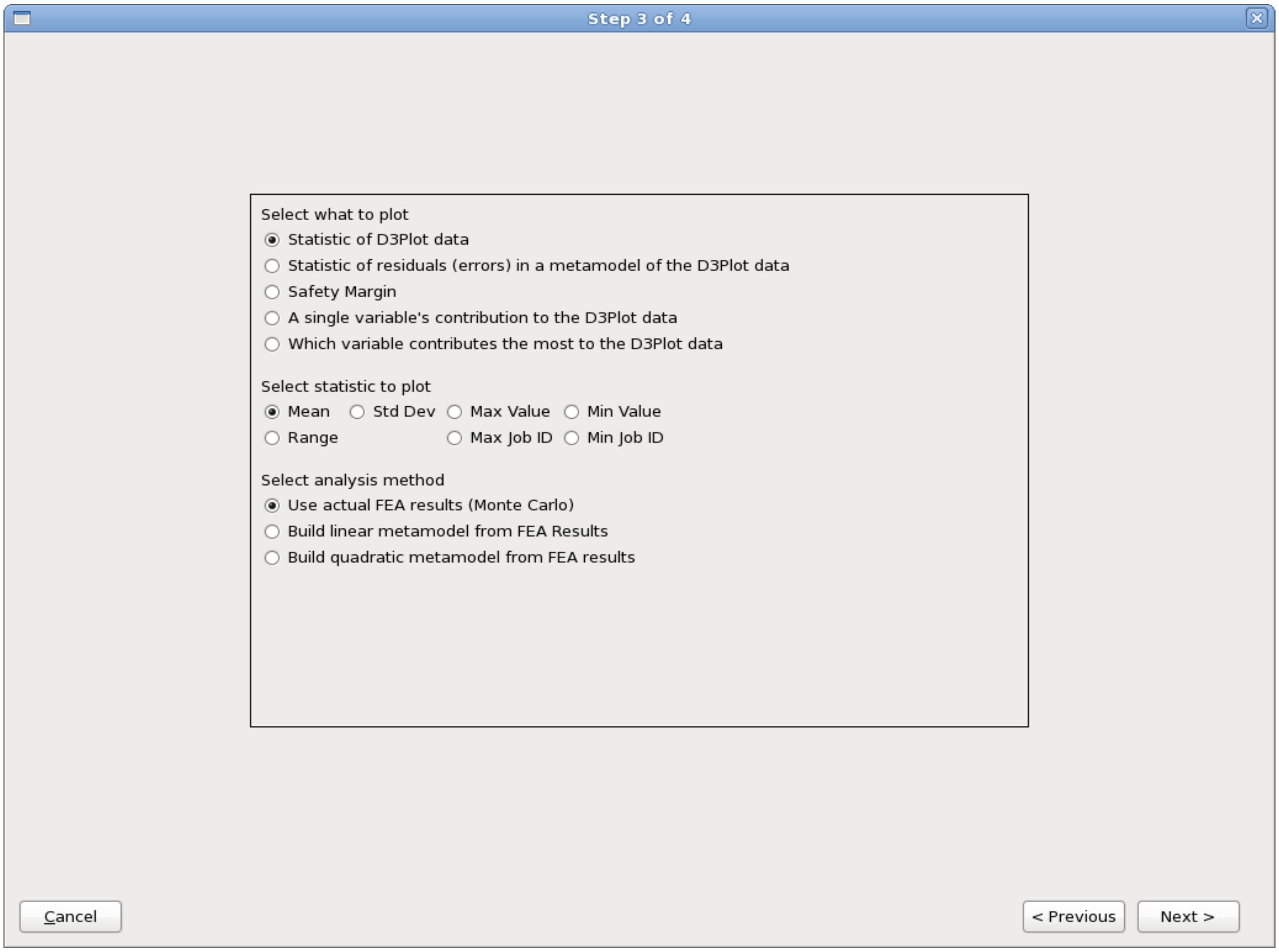
Figure 18-3: Second step of DynaStats plot creation; selection of d3plot component or history

Table 18-1: DynaStats Second step options

Option	**Description**	**Reference**
Select D3Plot component to plot	Statistics are calculated using values of selected component	
Select history to plot	Statistics are calculated using values of selected history	Section 18.7
Select stage to plot	Name of stage	
Subcase	Select subcase to use, only available if *CASE is used in the LS-DYNA input file.	

Follow coordinates instead of nodes	The ID of the part to be mapped has to be specified	Section 18.10
FLC curve	FLC curve specification (for FLD components, metal forming): Parametric FLD curve *t* and *n* coefficients Provided curve Curve ID in the LS-DYNA file of the FLD curve to be used	Section 18.10
Correlate response	Correlation between an LS-OPT response and a D3Plot component at all states	Section 18.4.1
Correlate variable	Correlation between an LS-OPT variable and a D3Plot component at all states	Section 18.4.2

18.2.3. Step 3 - Statistics



Select what to plot

- ● Statistic of D3Plot data
- ○ Statistic of residuals (errors) in a metamodel of the D3Plot data
- ○ Safety Margin
- ○ A single variable's contribution to the D3Plot data
- ○ Which variable contributes the most to the D3Plot data

Select statistic to plot

● Mean ○ Std Dev ○ Max Value ○ Min Value
○ Range ○ Max Job ID ○ Min Job ID

Select analysis method

- ● Use actual FEA results (Monte Carlo)
- ○ Build linear metamodel from FEA Results
- ○ Build quadratic metamodel from FEA results

Figure 18-4: Third step of DynaStats plot definition creation

Table 18-2: DynaStats Third step options

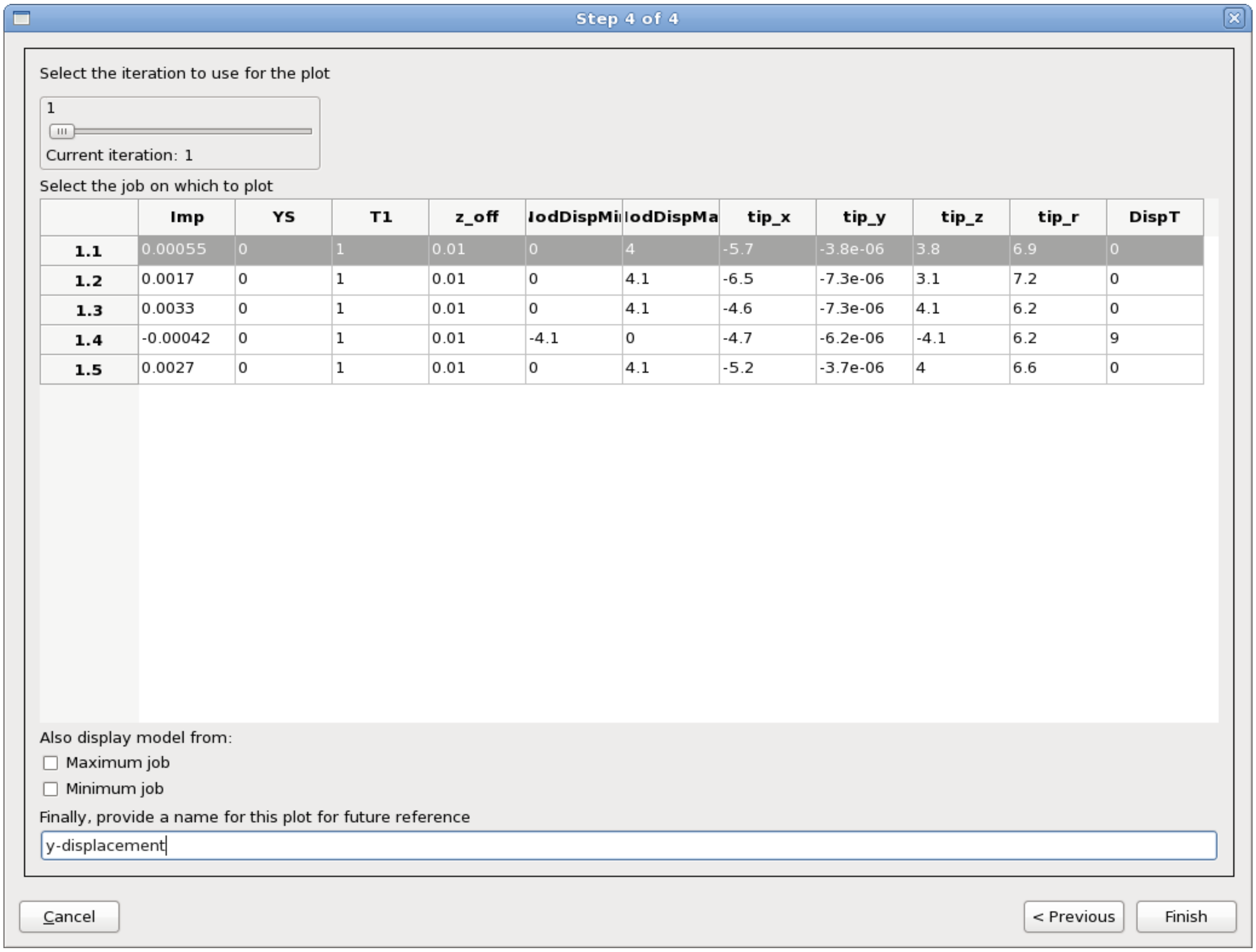
Option	Description	Reference
Select what to plot	Statistics of D3Plot data	
	Statistics of residuals (errors) in a metamodel of the D3Plot data	
	Safety Margin	Section 18.6
	A single variable's contribution to the D3Plot data	Section 18.5
	Which variable contributes the most to the D3Plot data	
Select statistics to plot		Section 18.3
Select analysis method	Use actual FEA results (Monte Carlo)	Section 18.3.1
	Build linear metamodel from FEA Results	Section 18.3.2
	Build quadratic metamodel from FEA results	Section 18.3.2

18.2.4. Step 4 – Visualization in LS-PREPOST

The user can select the LS-PREPOST plot details in LS-OPT (Figure 18-5). The GUI options will reflect whether fringe component response or history data is being investigated.

Table 18-3: DynaStats Visualization in LS-PREPOST options

Option	Description	Reference
Select the iteration to use for the plot	Iteration number	
Select the job on which to plot	Specifies the FE model used for the display of the results D3plot data from selected plot will be displayed in LS-PREPOST	
Also display model from	Bifurcation investigations: Additionally, to the model selected for the plot, the FE models containing the maximum and the minimum results can be overlayed in order to spot bifurcations.	Section 18.8
Name for this plot	Name for this plot used in the plots list in the DYNAStats main GUI.	



Select the iteration to use for the plot

1

Current iteration: 1

Select the job on which to plot

	Imp	YS	T1	z_off	lodDispMi	lodDispMa	tip_x	tip_y	tip_z	tip_r	DispT
1.1	0.00055	0	1	0.01	0	4	-5.7	-3.8e-06	3.8	6.9	0
1.2	0.0017	0	1	0.01	0	4.1	-6.5	-7.3e-06	3.1	7.2	0
1.3	0.0033	0	1	0.01	0	4.1	-4.6	-7.3e-06	4.1	6.2	0
1.4	-0.00042	0	1	0.01	-4.1	0	-4.7	-6.2e-06	-4.1	6.2	9
1.5	0.0027	0	1	0.01	0	4.1	-5.2	-3.7e-06	4	6.6	0

Also display model from:

☐ Maximum job

☐ Minimum job

Finally, provide a name for this plot for future reference

y-displacement

Figure 18-5: The statistics viewing options. The statistics will be shown in LS-PREPOST using the FE model from the LS-DYNA job specified using the Job field. The FE models of the jobs containing the maximum and minimum values can be overlayed in order to identify bifurcations as described in Section 18.8.

18.3. Monte Carlo and metamodel analysis

This section gives the options required for the computation of the statistics from a Monte Carlo or a metamodel based set of LS-DYNA results.

Either the LS-DYNA d3plot results or LS-OPT history results can be analyzed. The resulting output can be viewed in LS-PREPOST. The results will be stored in the stage directory with extensions of *.statdb* and *.history*.

The statistics are computed for a single stage and a single iteration.

18.3.1. Monte Carlo

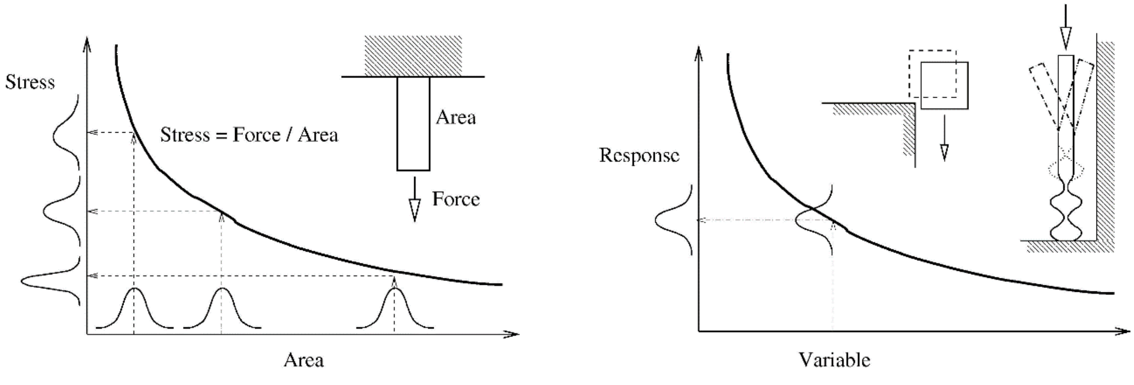
The statistics of the responses from a Monte Carlo procedure can be computed. The task will calculate:

1. Statistics of the response
    - Mean value of the response
    - Standard deviation of the response
    - Range of the response (maximum minus the minimum value)
    - Maximum value of the response
    - Minimum value of the response
    - ID of the LS-DYNA job where the maximum value occurred. This can be used to identify the jobs likely to contain a different bifurcation.
    - ID of the LS-DYNA job where the minimum value occurred. This can be used to identify the jobs likely to contain a different bifurcation.
2. The margin of safety (constraint margin) considering (*i*) a given bound on the response and (*ii*) the variation of the response as computed using the Monte Carlo analysis (see also Section 18.6).

18.3.2. Metamodels and residuals

Metamodels (approximations) can be used to predict the statistics of the responses. These metamodels will be computed for all results for all nodes and elements, respectively, for all time steps.

The metamodels are also useful for separating deterministic variation, caused by the variation of the design variables, from the process variation. The two types of variation are as shown in Figure 18-6.



Deterministic Variation

Process Variation

Figure 18-6: Different types of variation that can occur in a structure. The deterministic variation, predicted using the metamodel, is due to changes in the design variable values. The process variation, not associated with change in the design variable values, shows up in the residuals of the metamodel fit.

Metamodels are able to distinguish the process variation because, as shown in Figure 18-7, a metamodel can only predict the effect of the design variables. Process variation, not predictable by the design variables, becomes residuals.

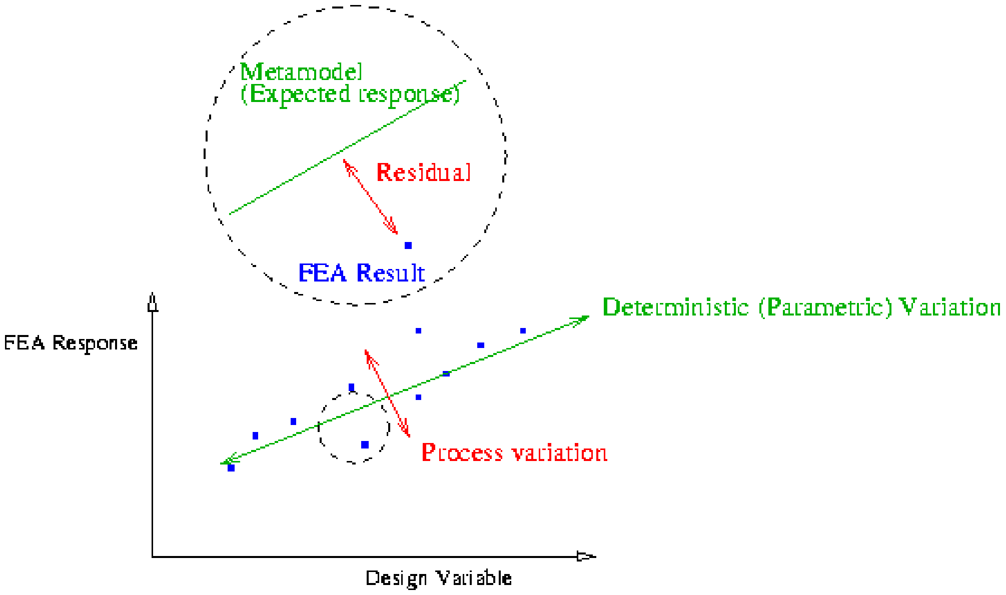


Figure 18-7: Metamodels can be used to distinguish between changes in the results due to the design variable changes and changes due to bifurcations.

The metamodel task will calculate:

1. Statistics of the response due to all the variables using the metamodel
   - Mean value of the response
   - Standard deviation of the response
   - Range (four standard deviations)
   - Maximum value (mean plus two standard deviations)
   - Minimum value (mean minus two standard deviations)
2. Statistics of the residuals
   - Mean value of the residuals (always zero)
   - Standard deviation of the residuals
   - Range of the residuals (maximum minus the minimum value)
   - Maximum value of the residuals
   - Minimum value of the residuals
   - ID of the LS-DYNA job where the maximum residual occurred. This can be used to identify the jobs likely to contain a different bifurcation.
   - ID of the LS-DYNA job where the minimum residual occurred. This can be used to identify the jobs likely to contain a different bifurcation.
3. Stochastic contribution of each individual variable and the variable contributing the most to the variation of the data, respectively.

4. The margin of safety (constraint margin) considering (i) a given bound on the response and (ii) the variation of the response as computed using the metamodel (see also Section 18.6).
5. All the computations as specified for the Monte Carlo procedure. The data required for this computation is read in for the metamodel computations, so very little time is expended computed these results as well.

The standard deviation of the variation caused by the design variables are computed using the metamodel as described in Section 29.7. The maximum, minimum, and range are computed using the mean value plus/minus two standard deviations. The *Max Job ID* and *Min Job ID* are not meaningful for the metamodel results.

The residuals are computed as the difference between the values computed using FEA and the values predicted using the metamodel (see Section 29.6 for more details).

A linear or a quadratic response surface can be used. The metamodel processing speed is approximately 10^5 - 10^6 finite element nodes a second, where the total nodes to be processed is the number of nodes in the model times the number of states times the number of jobs. FLD computations, which require the computation of the principle strains, can be a factor of five slower than computations using the nodal displacements. The overall speed is dominated by the time required to read the d3plot files from disk; accessing files over a network will be slow.

18.4. Correlation

18.4.1. Correlation of fringe plots or histories with responses

The correlation of the LS-DYNA results or LS-OPT histories with a response can be computed. This quantity indicates whether the changes in the responses are associated with the changes in the fringe or history. Figure 18-8 shows examples of a positive, a negative, and zero correlation. If not enough FE evaluations were conducted, the resulting fringe plot can be visually noisy. Thirty or more FE evaluations may be required. Note that the correlation of history is with respect to a response at a single time instance.

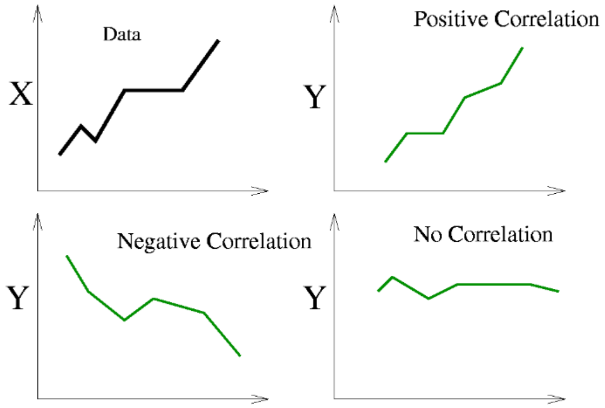
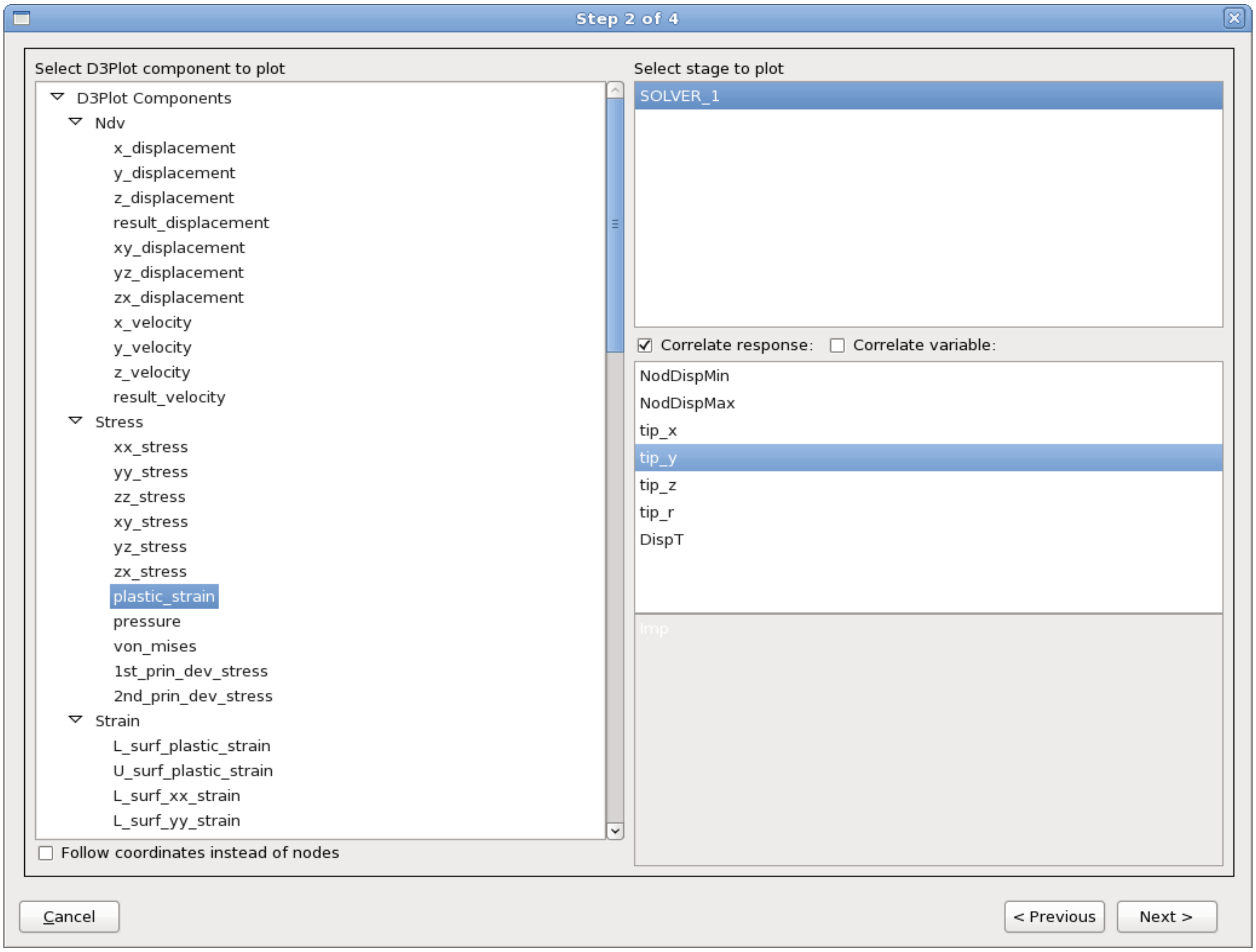


Figure 18-8: Correlation between X, shown in the upper left corner, and different responses Y. Different responses Y with a positive, a negative, and no correlation are shown.

18.4.2. Correlation of fringe plots or histories with variables

The correlation of the LS-DYNA results or LS-OPT histories with a variable can also be computed. This quantity indicates for all the time states whether the changes in a particular variable are associated with the changes in the D3Plot component or history. The correlation does not necessarily represent uncertainty or randomness of the variable. For example, even for a deterministic problem, such as a simple parametric or DOE study without random variables, there can be a non-zero correlation between a variable and a LS-DYNA response component.



☑ Correlate response: □ Correlate variable:

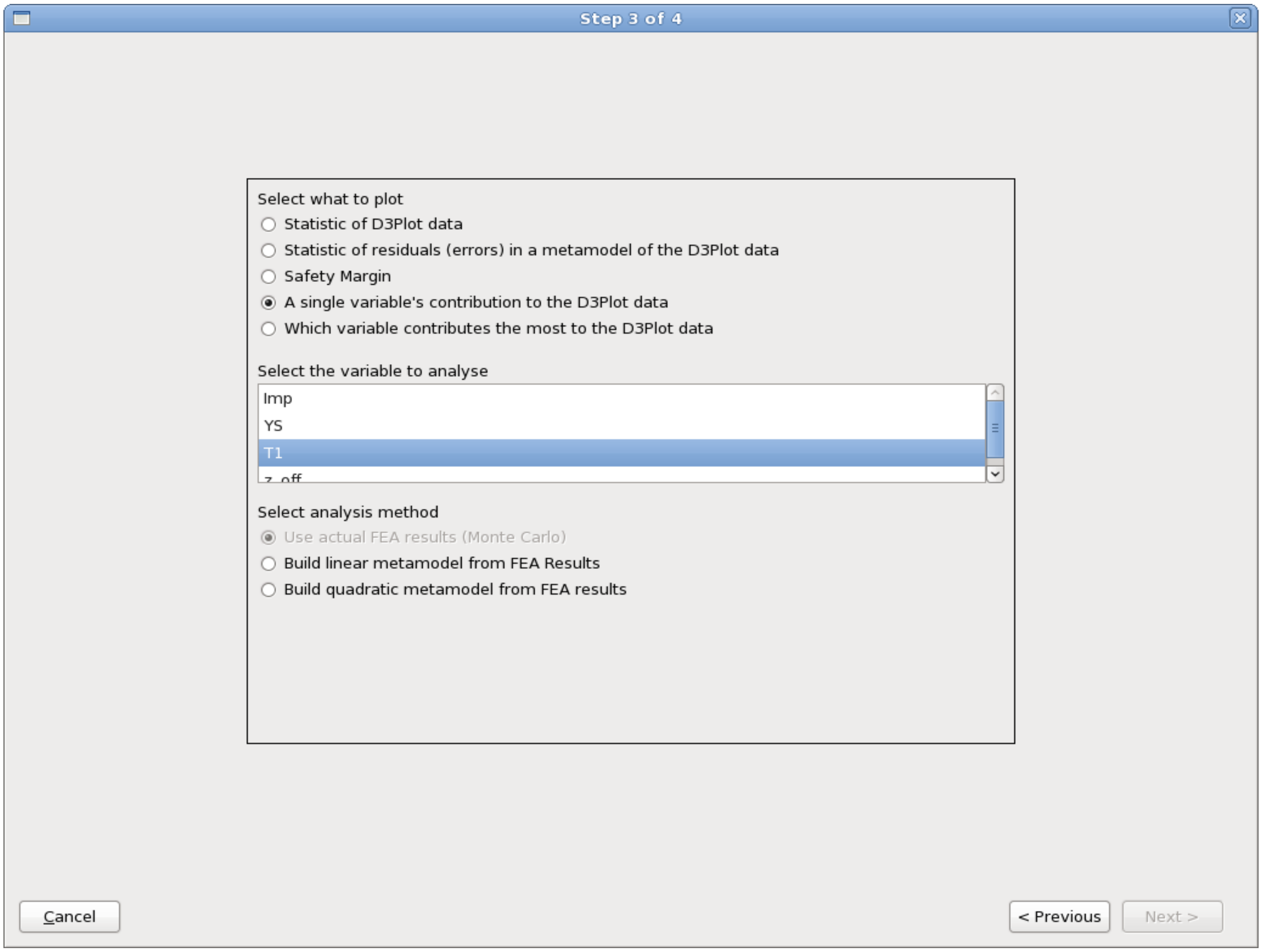
□ Follow coordinates instead of nodes

Figure 18-9: Viewing the correlation between an LS-DYNA response and an LS-OPT response. Additionally, the correlation between an LS-OPT history and an LS-OPT response or variable can also be viewed.

18.5. Stochastic contribution of a variable

The stochastic contribution of each design variable to the variation of the nodal response can also be plotted on the model by selecting *A single variable's contribution to the D3Plot data* and a variable form the list. These results are computed as described in Section 29.7. It is important to note that stochastic contribution, though closely related, is not the same as sensitivity or correlation. While sensitivity and correlation can be non-zero for both stochastic and deterministic problems, stochastic contribution of a deterministic variable is always zero. Stochastic contribution provides the variation of a response due to randomness of a variable. Thus it depends not only on the relation between the response and the variable (also studied using sensitivity or correlation), but also the degree of uncertainty in the variable. Higher randomness of a variable would lead to greater stochastic contribution (assuming non-zero sensitivity).

The most important variable based on stochastic contribution, or rather the variable responsible for the most variation of the response, can be plotted on the model by selecting *Which variable contributes the most to the D3Plot data*. Actually, only the index of the variable is displayed on the model. This index is the same as in the list of variables as shown in the LS-DYNA results statistics GUI. The importance of stochastic contribution analysis is more significant from the perspective of uncertainty or probabilistic analysis. The most important variable based on stochastic contribution may not necessarily be the most important based on sensitivity analysis, as the latter does not consider the actual probabilistic distributions of variables.



Select what to plot

○ Statistic of D3Plot data

○ Statistic of residuals (errors) in a metamodel of the D3Plot data

○ Safety Margin

● A single variable's contribution to the D3Plot data

○ Which variable contributes the most to the D3Plot data

Select the variable to analyse

Imp
YS
T1
z_off

Select analysis method

● Use actual FEA results (Monte Carlo)

○ Build linear metamodel from FEA Results

○ Build quadratic metamodel from FEA results

Figure 18-10: Viewing the stochastic contribution of a single variable.

18.6. Safety margin

The safety margin as shown in Figure 18-11 can be displayed in three ways:

1. The safety margin — the difference between the bound and mean,
2. The safety margin measured (scaled) in standard deviations (sigmas), and
3. The probability of exceeding the bound (probability of failure).

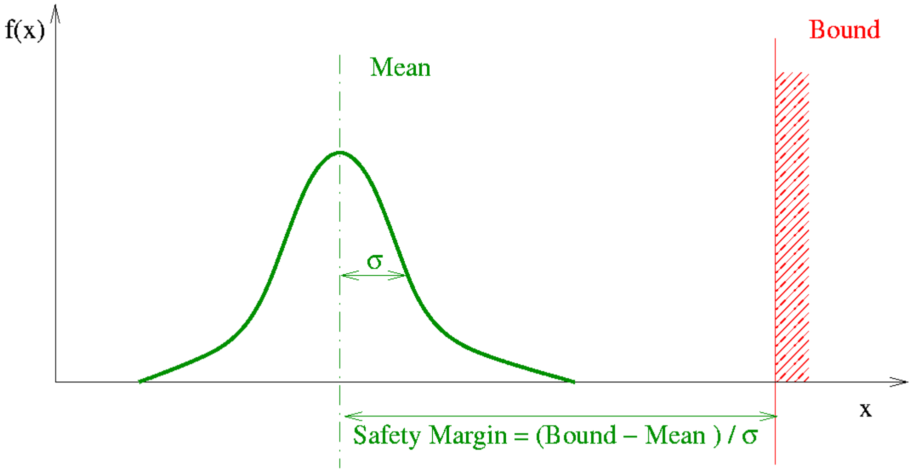
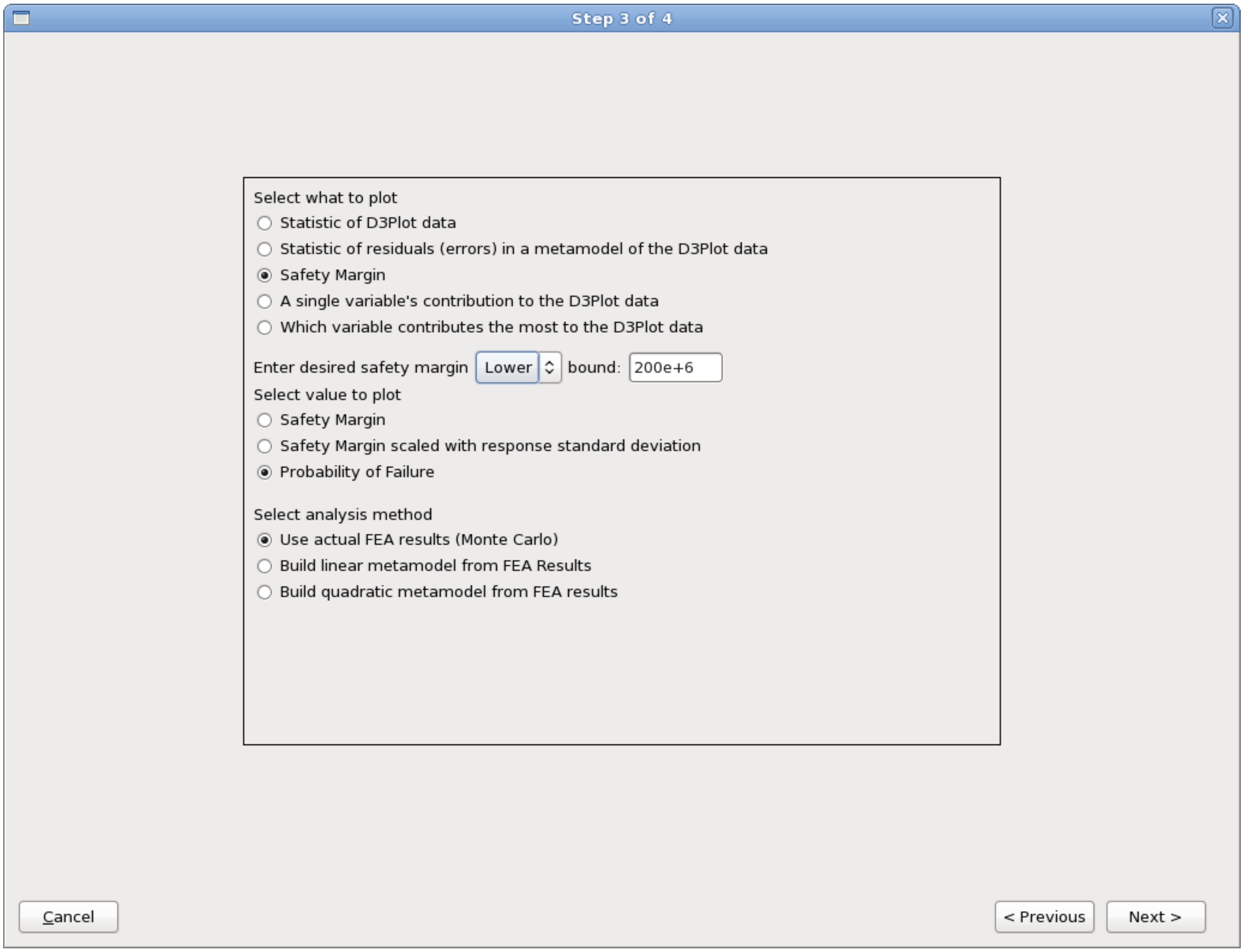


Figure 18-11: The safety margin is the difference, measured in standard deviations, between the mean response and the constraint bound on the response.

The bound must therefore be specified when the statistics are computed as shown in Figure 18-12. Obtaining the safety margin for a different bound requires the generation of a new plot.

The probability of exceeding the bound is computed using the FOSM method (see Section 29.4.5) using the normal distribution together with the safety margin measured in standard deviations (sigmas). The computation is therefore done in the six-sigma sense — the number of sigmas (standard deviations) is the unit of measure. If a Monte Carlo computation of the probability of failure is desired, then it must be computed using a response in the Statistical Tools plot, Section 16.2.6; if this response was not defined originally then it must be extracted from the binout or d3plot database: first defining a binout or d3plot response, do a Repair/Extract Results, Section 3.5, and use Statistical Tools plot with Plot type Probability of constraint violation.



Step 3 of 4

Select what to plot

○ Statistic of D3Plot data

○ Statistic of residuals (errors) in a metamodel of the D3Plot data

● Safety Margin

○ A single variable's contribution to the D3Plot data

○ Which variable contributes the most to the D3Plot data

Enter desired safety margin Lower bound: 200e+6

Select value to plot

○ Safety Margin

○ Safety Margin scaled with response standard deviation

● Probability of Failure

Select analysis method

● Use actual FEA results (Monte Carlo)

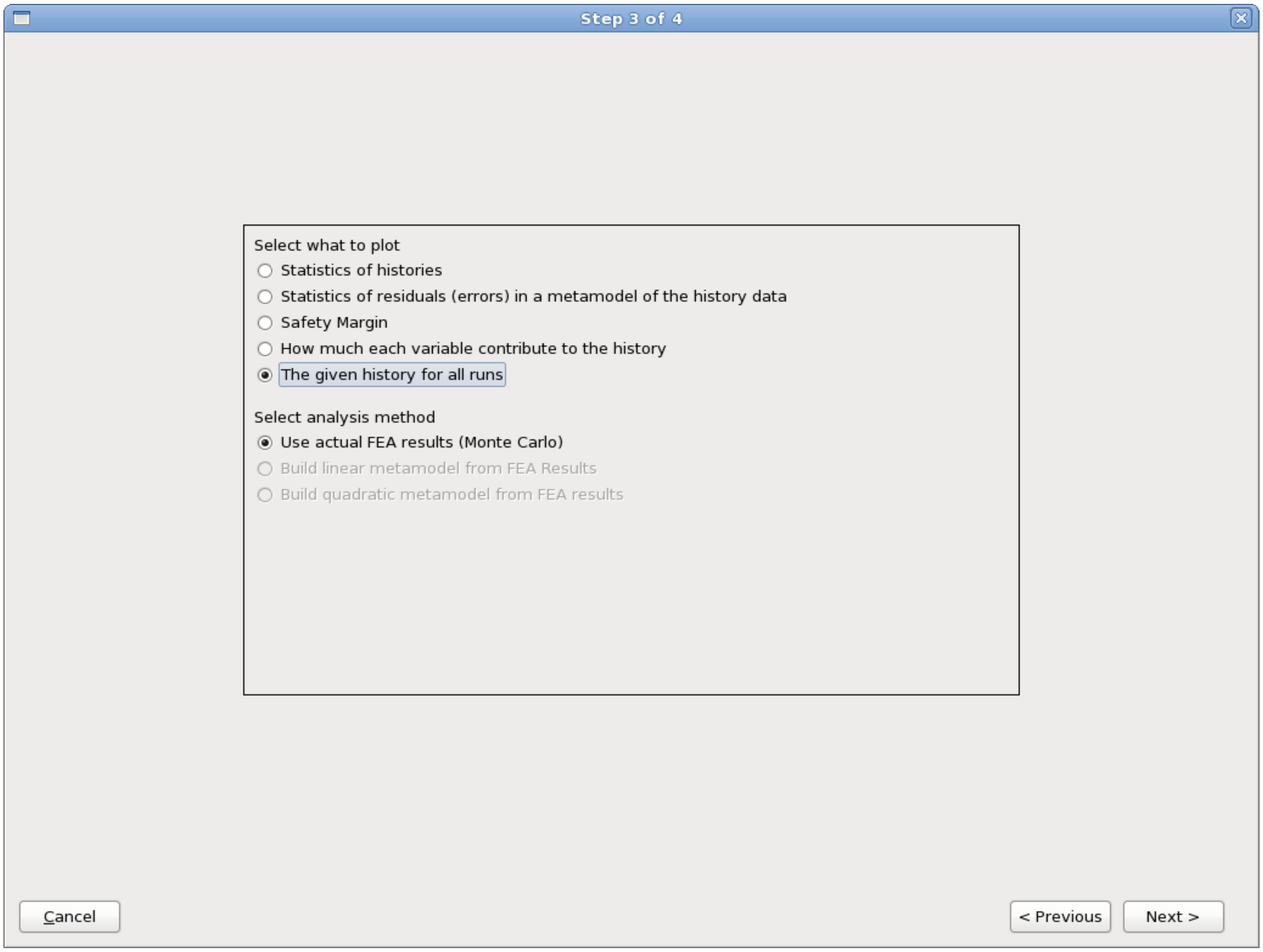
○ Build linear metamodel from FEA Results

○ Build quadratic metamodel from FEA results

Figure 18-12: Plotting a safety margin or the probability of failure requires that the bound must be specified.

18.7. Viewing LS-OPT histories

The LS-OPT histories for all the LS-DYNA runs can be viewed simultaneously. See Figure 18-15 for an example. In addition, various statistics of LS-OPT histories at all time states can also be viewed. The safety margin or probability of failure can also be viewed for all time states.



Step 3 of 4

Select what to plot

○ Statistics of histories
○ Statistics of residuals (errors) in a metamodel of the history data
○ Safety Margin
○ How much each variable contribute to the history
● The given history for all runs

Select analysis method

● Use actual FEA results (Monte Carlo)
○ Build linear metamodel from FEA Results
○ Build quadratic metamodel from FEA results

Figure 18-13: Viewing all the LS-OPT histories.

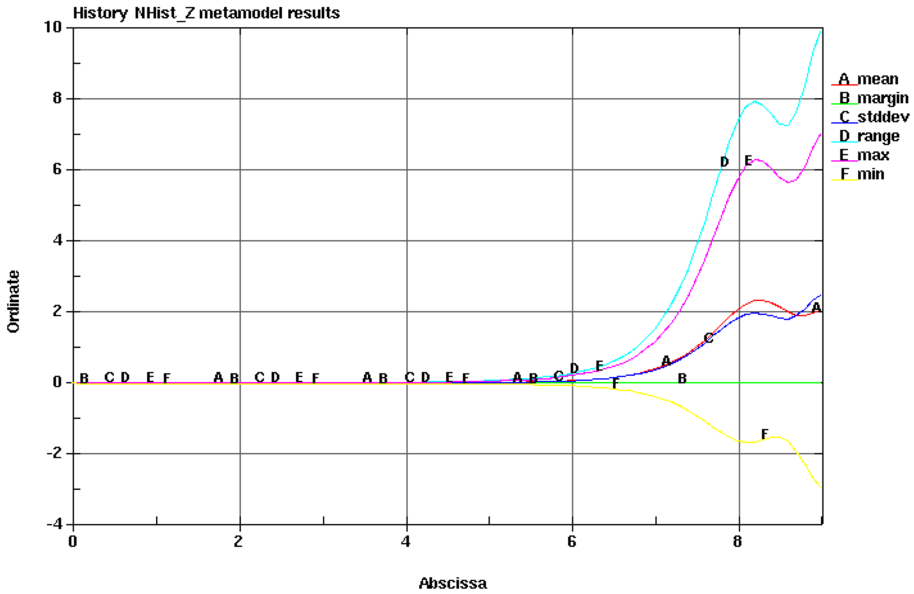


Figure 18-14: Statistics of an LS-OPT history.

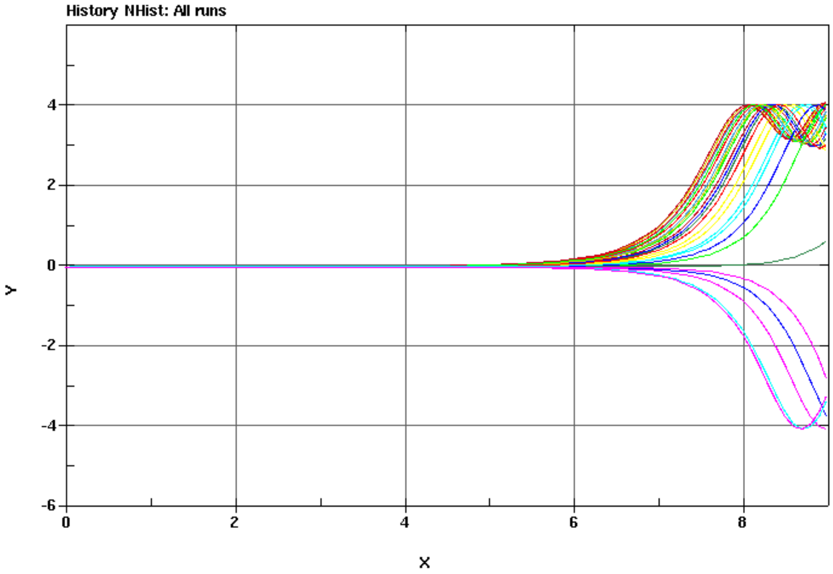
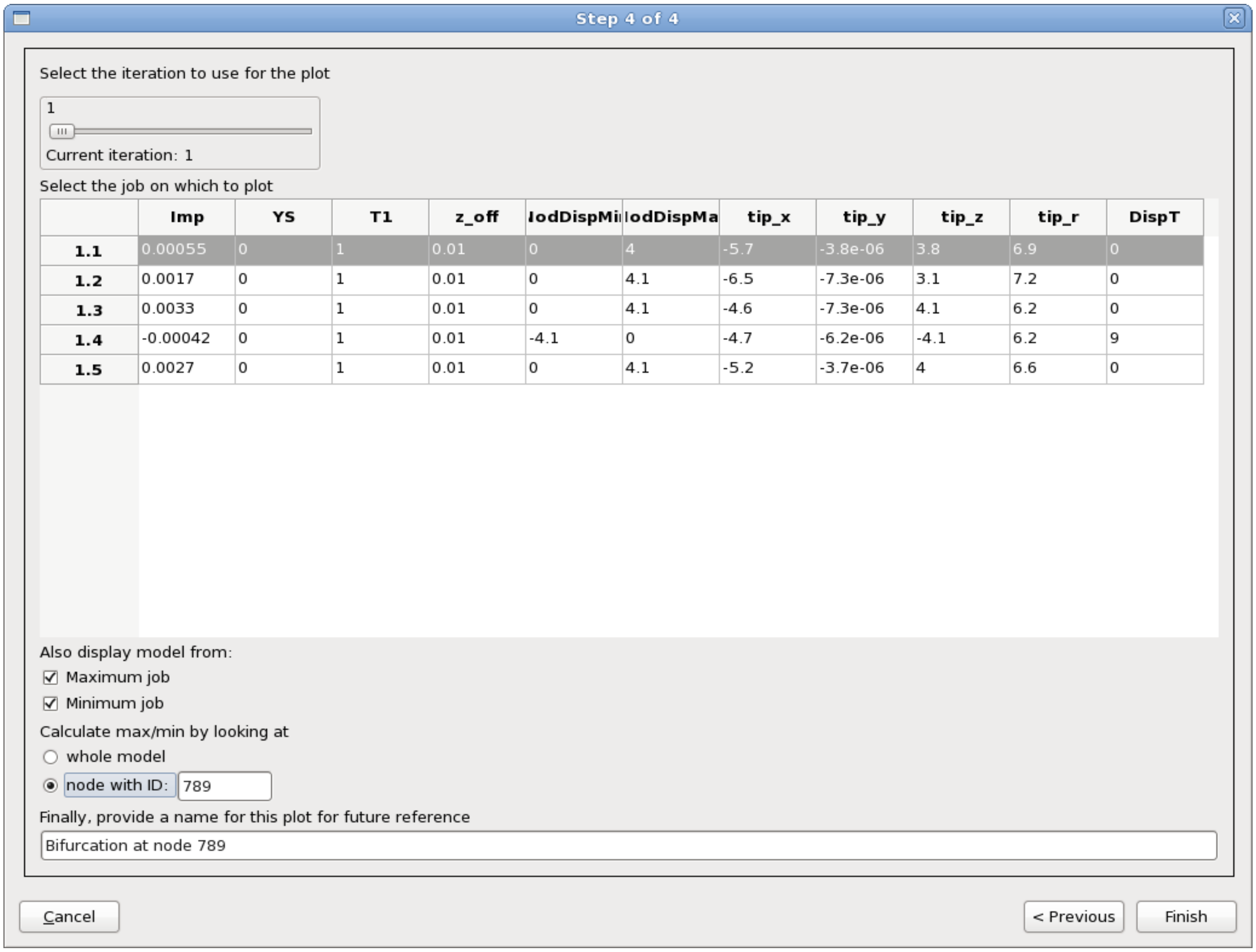


Figure 18-15: The LS-OPT histories of all the LS-DYNA runs can be viewed simultaneously.

18.8. Bifurcation investigations

The residuals plots are useful for finding bifurcations. The standard deviation (or range) of the residuals indicate regions where the changes in displacements are not explained by changes in the design variable values — it is therefore a plot of the unexpected displacements or 'surprise factor'. The plots from a Monte Carlo analysis can also be used to find bifurcations similarly to the residuals from a metamodel-based Monte Carlo analysis.



Select the iteration to use for the plot

1

Current iteration: 1

Select the job on which to plot

	Imp	YS	T1	z_off	lodDispMi	lodDispMa	tip_x	tip_y	tip_z	tip_r	DispT
1.1	0.00055	0	1	0.01	0	4	-5.7	-3.8e-06	3.8	6.9	0
1.2	0.0017	0	1	0.01	0	4.1	-6.5	-7.3e-06	3.1	7.2	0
1.3	0.0033	0	1	0.01	0	4.1	-4.6	-7.3e-06	4.1	6.2	0
1.4	-0.00042	0	1	0.01	-4.1	0	-4.7	-6.2e-06	-4.1	6.2	9
1.5	0.0027	0	1	0.01	0	4.1	-5.2	-3.7e-06	4	6.6	0

Also display model from:

☑ Maximum job

☑ Minimum job

Calculate max/min by looking at

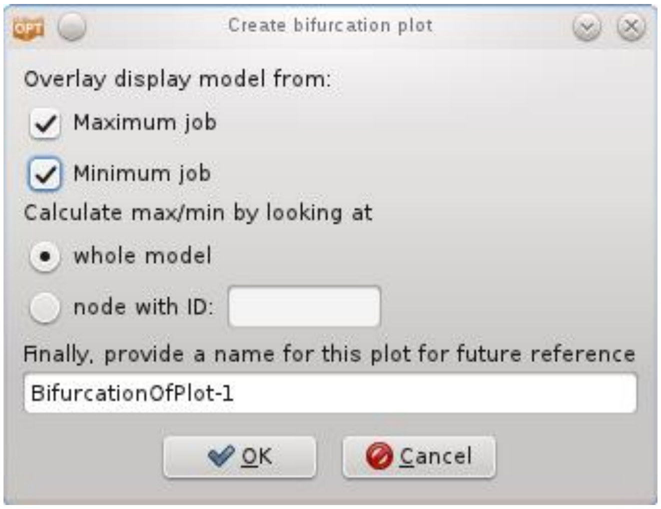
○ whole model

● node with ID: 789

Finally, provide a name for this plot for future reference

Bifurcation at node 789

Figure 18-16: Bifurcation options. The bifurcation is found by superimposing the FE models containing the maximum and minimum results. A node ID associated with the bifurcation may need to be specified if the extreme values in the model are not caused by the bifurcation.



Overlay display model from:

☑ Maximum job

☑ Minimum job

Calculate max/min by looking at

● whole model

○ node with ID:

Finally, provide a name for this plot for future reference

BifurcationOfPlot-1

Figure 18-17: Options to create Bifurcation Plot for an existing plot.

18.8.1. Automatic detection

Automatic detection of the LS-DYNA jobs containing the minimum and maximum outlier can be done as shown in Figure 18-16 and Figure 18-17. In the GUI the user must select (i) overlay of the FE models containing the maximum and minimum results and (ii) whether the global minimum or the minimum at specific node must be used. Viewing the maximum and minimum job simultaneously allows the bifurcation to be identified. See Figure 18-18 for an example of the resulting LS-PREPOST plot.

18.8.2. Manual detection

The steps for manual detection are:

1. Plot displacement magnitude outlier *Range* to identify location in FE model where the bifurcation occurred.
2. Identify job in which maximum value occurred using a *Max Job ID* plot
3. Identify job in which minimum value occurred using a *Min Job ID* plot
4. View the location in model for the jobs having the minimum and maximum value.

Recommendations:

- Engineering knowledge of the structure is important.
- Look at the *x*, *y*, and *z* components in addition to the displacement magnitude to understand in which direction the bifurcation occurred; most bifurcations are actually best identified considering a displacement component.
- The history results may be useful to find the time at which a bifurcation occurred.
- The correlation between a response and displacements (or histories) indicates if variation of the displacement is linked to variation of the response.
- Look at all of the states in the d3plot database; the bifurcation may be clearer at an earlier analysis time.

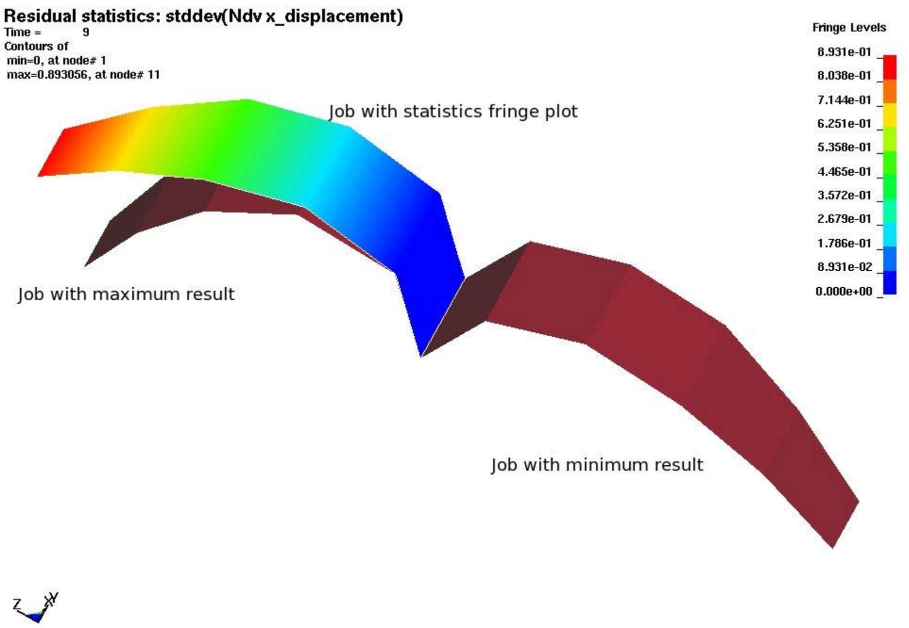


Figure 18-18: Viewing a bifurcation. Plate structure that can buckle either left or right. Three FE models are shown, and the two distinctly different solution modes are clearly visible. The creation and display of the plot containing all three models are automated in LS-OPT.

18.9. Displacement magnitude issues*

Approximation of the displacement magnitudes (resultants) introduces some special cases. The magnitude is defined as the square root of a sum of squares, which is difficult to approximate around the origin, especially using linear approximations, Figure 18-19. The x, y, and z displacement components do not suffer from this problem.

Unexpected results may occur even if the displacement magnitude is approximated correctly. The displacement magnitude is always a positive quantity, which, in addition to the fitting problems, can also cause problems when computing the coefficient of correlation with a response quantity. Figure 18-20 illustrates two buckling modes of a flange evaluated at two locations in space. The displacement magnitude variance differs for the two locations though the buckling modes are similar. The variance of the displacement magnitude will therefore be smaller than what would be found considering the components. Considering a displacement component will cure this problem, but a displacement component aligned with the required direction may not always exist.

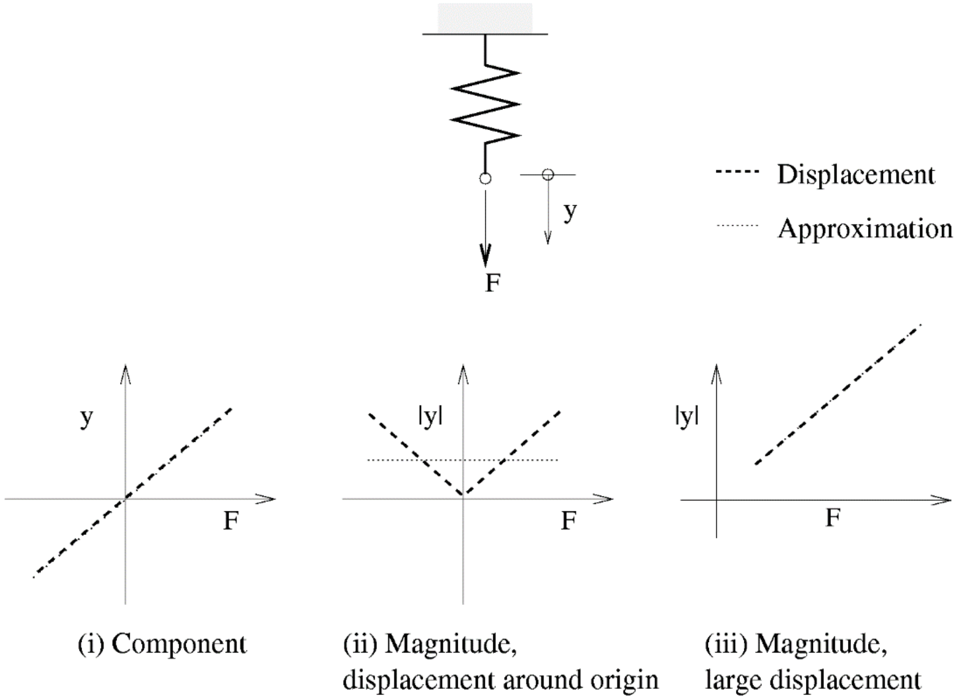


Figure 18-19: Displacement approximation scenarios. The displacement magnitude, being always larger than zero, cannot be approximated accurately around the origin if some of the displacement components can have a negative value.

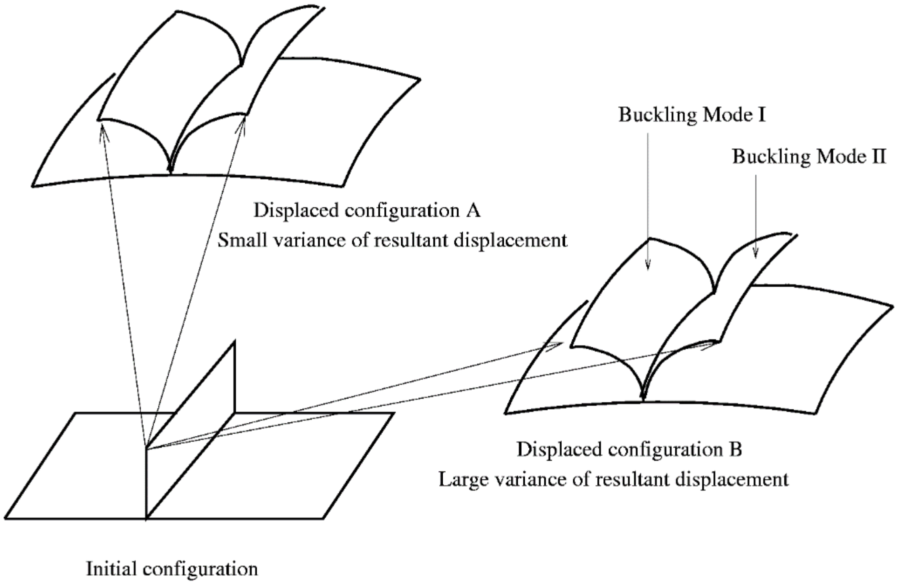

Buckling Mode I
Buckling Mode II
Displaced configuration A
Small variance of resultant displacement
Displaced configuration B
Large variance of resultant displacement
Initial configuration

Figure 18-20: The displacement magnitude can depend on the aligment of the flange with the axis. The buckling will be difficult to spot if it is aligned with the position of the axis. For configuration A, the two vectors have nearly the same length, while for configuration B, they clearly have different lengths.

Recommendations:

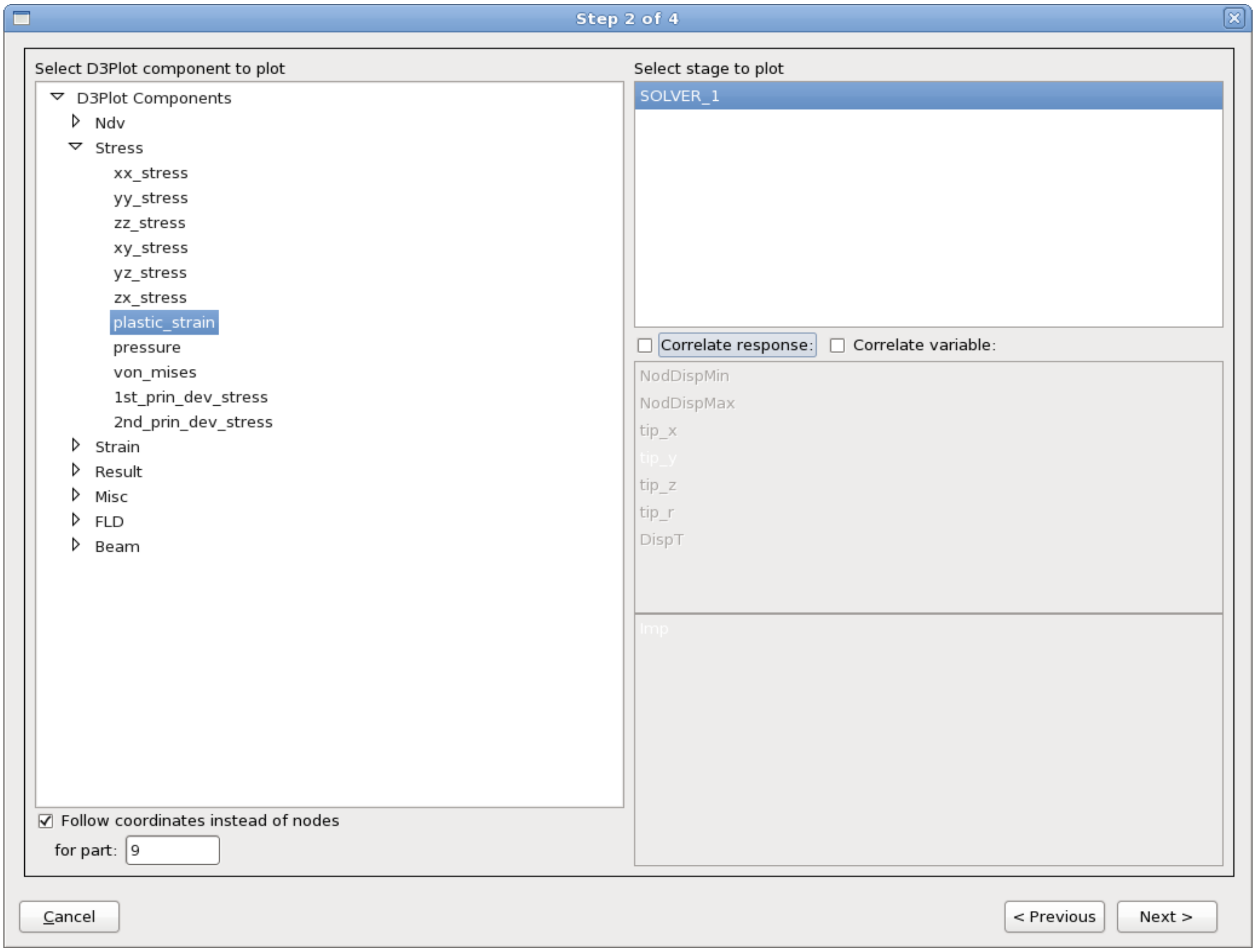
- Use the x, y, and z displacement components.

18.10. Metalforming options

Metalforming has some special requirements. It is possible to:

1. Map the results for each sample to the mesh of the base design. The results will be computed at a specific spatial location instead of a node (Eulerian system). This is required in metalforming because:
    - The adaptivity will result in the different iterations having different meshes.
    - It is more natural in metalforming to consider the results at a specific geometric location than at a specific node.

    This is done only for the work piece. This part must therefore be specified in the LS-OPT input. More detail is shown in Figure 18-21, Figure 18-22 and Figure 18-23.
2. Specify the FLC curve to be used in the computation of the FLD responses. This can be done by either specifying the ID of a curve defined in the LS-DYNA input deck (option *Provided curve*) or using two parameters similar to that being used in LS-PREPOST (option *Parametric*).



☑ Follow coordinates instead of nodes

for part: 9

Figure 18-21: For metal forming specify that the coordinates instead of the nodes must be followed and specify the part (blank) for which the results must be mapped.

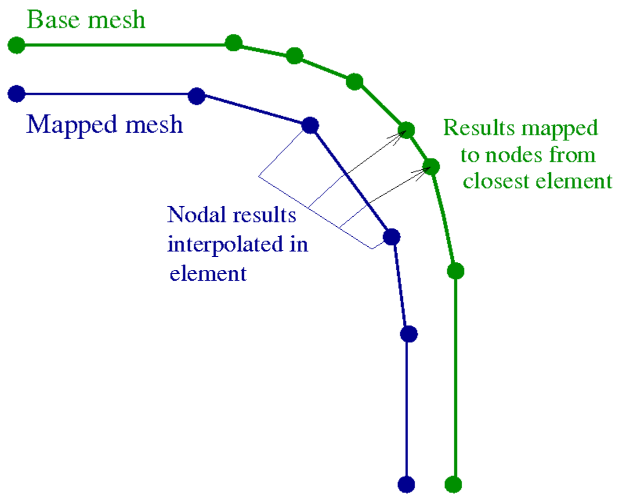


Figure 18-22: Interpolation of metal forming results.

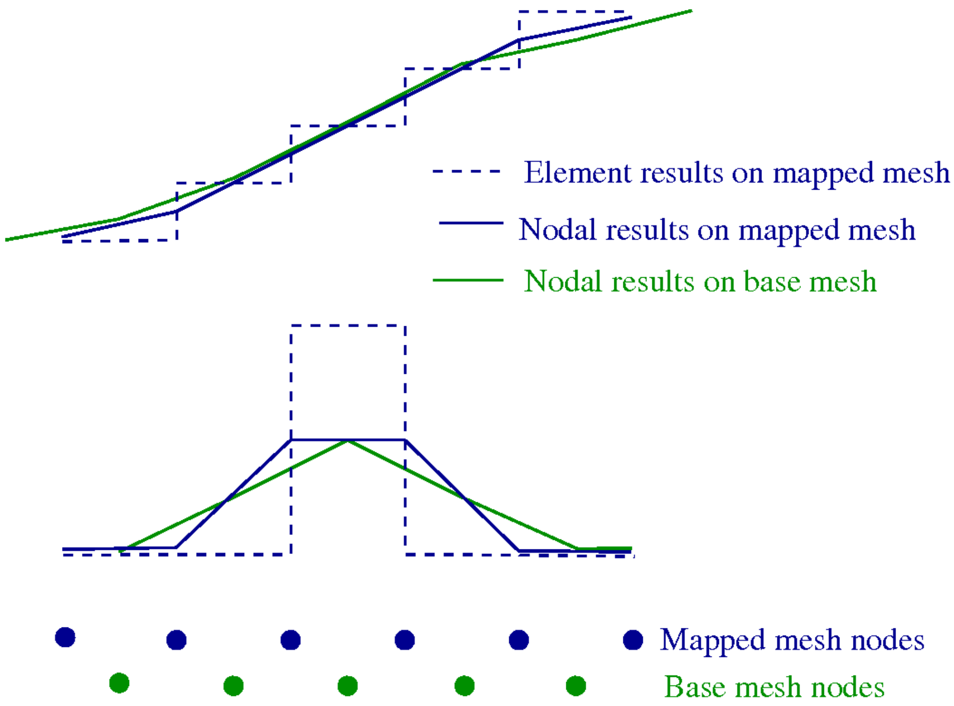


Figure 18-23: Acuracy of of the mapping operation for element results is shown for two cases. For each case the results are shown as the element centroid results for the original mapped mesh, the element results averaged at the nodes for the original mapped mesh, and the results mapped to the nodes of the base mesh. For the first case it can be seen that the mapping accuracy is good if the mesh is sufficiently fine to consider smoothly varying results. The second case, which occurs when yielding occurs in a single element, indicates a loss of information. But for this second case, the exact numerical value of the original results is not considered very accurate, so we can consider the mapped results as sufficient as long as they conserve the prediction of failure. For the second case the numerical values are mesh-dependent, so the prediction of failure is the quantity that should be mapped to another mesh.

18.11. User-defined statistics*

Although DynaStats provides an interface only for LS-DYNA response components, it also provides a way to visualize statistics of user-defined results. This requires a script from the user that is run by LS-OPT in each run directory to calculate the user-defined results for that subdirectory, and eventually the statistics of all the runs. The steps involved are listed below:

1. Select “Misc, user” as the D3Plot Component in DynaStats Creation Wizard, Section 18.2.2, to define the required statistic.
2. A script named “dstats_user” needs to be provided by the user. In each subdirectory LS-OPT will run the program “dstats_user -state n” for n ranging from 1 to the total number of states. The program “dstats_user” must dump a file called “dstats.lspp” for the particular state being run.
3. LS-OPT will open the file “dstats.lspp” for every state. The file must be in the same format as dumped by LS-PrePost output command. The data must be written in “%10d%10f” format as nodal results. If the results are available as element results, they must first be converted to nodal results using nodal averaging in LS-PrePost.

A sample dstats_user python program to dump nodal results from LS-PrePost is given below. In general, the user can dump results from any program into the file dstats.lspp.

```
import sys, os
cmp  = 9 # Von Mises
state = 1
print "state", state
print "argv", sys.argv
if len(sys.argv) > 2 : state = eval( sys.argv[2] )
print "state", state
ff = open( "lspp.cmd", 'w' )
ff.write( "openc d3plot \"d3plot\"\n" )
ff.write( "state %d;\n"%state )
ff.write( "fringe %d\n"%cmp )
ff.write( "pfringe\n" )
ff.write( "output dstats.lspp %d 1 0 1 0 0 0 0 0 1 0 0 0 0 0\n"%state )
ff.write( "exit\n" )
ff.close( )
os.system( "lsprepost c=lspp.cmd" )
```

If element results are available, they must first be dumped by dstats_user before running additional LS-PrePost commands to read those results, convert them into nodal outputs, and dump the new results. If the element results are written to dstats_e.lspp then the following dstats_user should be modified as follows.

```
import sys, os
cmp  = 9 # von mises
state = 1
print "state", state
print "argv", sys.argv
if len(sys.argv) > 2 : state = eval( sys.argv[2] )
print "state", state
ff = open( "lspp.cmd", 'w' )
ff.write( "openc d3plot \"d3plot\"\n" )
ff.write( "state %d;\n"%state )
ff.write( "fringe %d\n"%cmp )
ff.write( "pfringe\n" )
ff.write( "range avgfrng none\n" )
ff.write( "output dstats_e.lspp %d 1 0 1 0 0 0 0 1 0 0 0 0 0 0\n"%state )
# read the file with element data
ff.write( "open userfringe dstats_e.lspp 1\n")
ff.write( "fringe 5001\n")
ff.write( "pfringe\n" )
# write the corresponding file with nodal data
ff.write( "range avgfrng node\n" )
ff.write( "output dstats.lspp %d 1 0 1 0 0 0 0 0 1 0 0 0 0 0\n"%state )
ff.write( "exit\n" )
ff.close( )
os.system( "lsprepost c=lspp.cmd" )
```

18.12. Re-use and persistence of an evaluation methodology*

The definitions of the plots are saved in a file named *dynastatplots.xml*. Copy this file to the directory where you want to re-use the definitions. The plots will be available when you restart the LS-OPT GUI. The plots will have to be re-generated though; note that you can select all of the plots when you generate plots – there is no need to generate plots one-by-one.

Using the File menu Export and Import features, all defined plots may be exported to an .xml file and selected plots can be imported.

19. Applications of Optimization

This chapter provides a brief description of some of the applications of optimization that can be performed using LS-OPT. It should be read in conjunction with the examples chapters, where the applications are illustrated with practical examples.

19.1. Parameter Identification

Parameter identification problems are non-linear inverse problems which can be solved using mathematical optimization. System parameter identification is a commonly used feature of LS-OPT, especially for the purpose of calibrating material models.

The procedure consists of minimizing the mismatch between target values and corresponding solver output values, or between two curves. In the latter case, the two curves typically consist of a two-dimensional experimental target curve and a computed curve. The computed curve is a variable response, being dependent on the system parameters, e.g. material constants. It can also be a *crossplot*, constructed by combining two time histories such as *strain* and *stress* (Section 0).

The two main essential components of an algorithm designed for system identification are

- optimization algorithm and
- curve matching metric.

19.1.1. Optimization algorithm

The recommended optimization algorithm to be used to solve a parameter identification problem is the *Metamodel-based Optimization* with the strategy *Sequential with Domain Reduction* (SRSM), Section 4.8.3. Use linear polynomial metamodels and *D-optimal* point selection which is the default for the selected task and strategy, Section 9.3.2.

SRSM is the most efficient optimization method, but the result could be a local optimum. If global optimality is desired and simulation time is short enough, the Direct Genetic Algorithm, Section 4.4, is recommended. It might lead to better optimization results, but at roughly 20 times the cost.

19.1.2. Matching scalar values

To match scalar values, extract the respective responses from the solver output. Specify a *Standard Composite* of type *MSE* or *Sqrt MSE* using these responses as components associated with the respective target value, Section 10.4. Define this composite as an objective function.

19.1.3. Curve matching measures

To calculate the mismatch between the target and the computed curve, define a *Curve Matching* composite or response, Section 6.4.4 and Section 10.5. There are four curve matching measures available, namely the *Euclidean (formerly known as Mean Square Error), Curve Mapping, Dynamic Time Warping* and *Discrete Fréchet* measures.

- The *Euclidean* measure is an ordinate-based curve matching metric. Hence if the curve has steep parts or if the ordinate values are not unique, (e.g. the curve is a hysteretic curve), this should not be the measure of choice.
- *Curve Mapping* uses the length of the curve to calculate the mismatch. In the case of noisy curves, this results in an artificial mismatch of the curves because of possible difference in length. In this case filtering of the component history curves is recommended. Since filtering might not be robust, *Dynamic Time Warping* is the metric of choice for noisy curves.
- *Dynamic Time Warping* calculates the distance between two data sets (curves) via its corresponding warping path. This path is the result of the minimum accumulated distance necessary to traverse all points in the curves. The matching is end-to-end. If the curves are not of comparable length, e.g. if the simulation curves have trailing ends, using the option *Truncate computed using target* is recommended. While the Euclidean distance measure is a strict one-to-one mapping, DTW also allows one-to-many mappings
- The *Discrete Fréchet* distance measures the similarity between curves by taking the minimum of the maximum of all possible edge lengths along a path which connects all given data points, while taking into account the location and ordering of the points. The matching is end-to-end. If the curves are not of comparable length, *Dynamic Time Warping* with the *Truncate computed using target* option or *Curve Mapping* is recommended. If points are not uniformly distributed, particularly in the case of a big gap between two points on a chain, DFD will focus on the gap.

19.1.4. Parameter Identification using full-field measurements

If full-field test results generated by an optical measurement system are provided, `d3plot` multihistories (Section 6.2.3) and file multihistories (Section 6.19) should be used to perform the full-field calibration. The curve matching measures are applied to multihistories in the same manner as explained in Section 19.1.3, See also Section 28.3.

19.1.5. Sampling constraints

For parameter identification problems, there are often more restrictions on design variables than just a lower and an upper bound for each parameter, e.g. there may be a requirement to obtain monotonically increasing solver input curves. Such constraints can be defined as *Sampling Constraints* in LS-OPT, Section 9.6.

19.1.6. Parameterization of solver input curves

A common way to parameterize a solver input curve is to use a parameterized analytical function that represents the characteristic of the curve. Use a program or script as a solver of a preprocessor stage to calculate the solver input curve depending on parameters.

19.1.7. Viewer

This section describes some postprocessing options commonly used for parameter identification problems. Further options are described in Chapter 16.

Optimization History

The optimization history plot can be used to check the convergence of the variable values as well as the decrease of the objective over the iterations. The response optimization history displays computed and predicted values; hence it can be used to check the quality of the predictions. See Section 16.4.1 for further information on the optimization history plot.

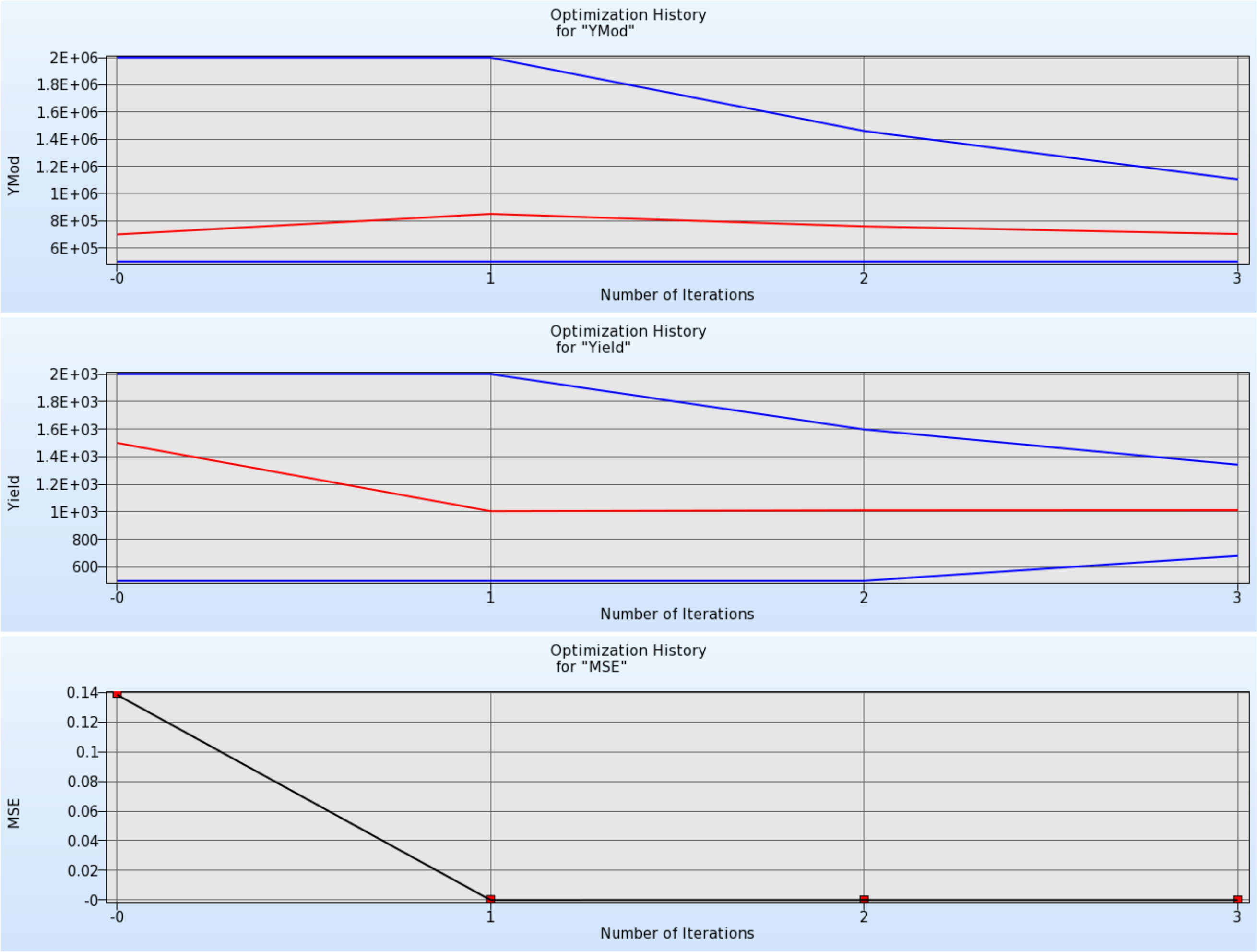


Figure 19-1: Optimization history for variables and objective

Sensitivities

If there are parameters that do not converge, the sensitivities plot can be used to see if those parameters are insensitive. See Section 16.3.4.

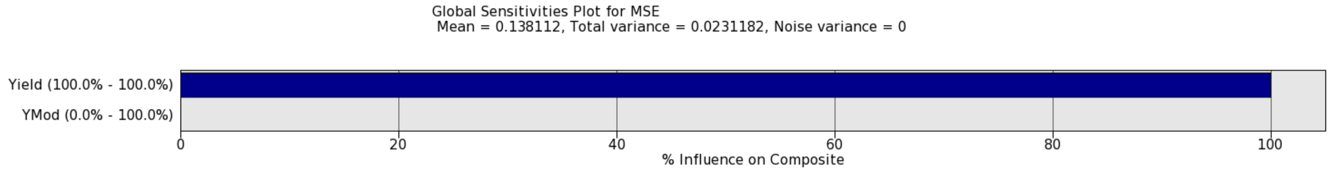
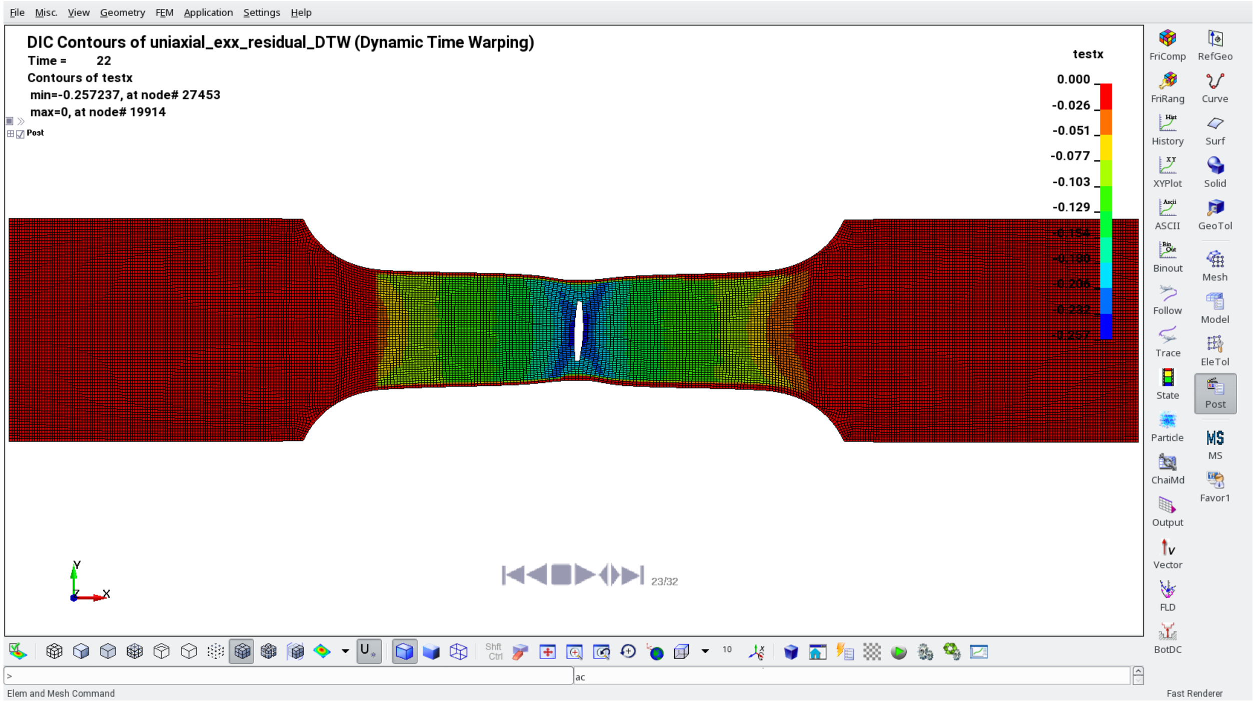


Figure 19-2: Global Sensitivities

LS-PrePost Fringe Plots

For full-field calibration, several fringe plot options are available from the Viewer table or plots, Section 17.1.9. These can be used to visualize the simulation or experimental contours as well as the difference contours, superimposed on the Finite Element model. The similarity measure selected by the user for the analysis is used in the mapping. An example is displayed in Figure 19-3.



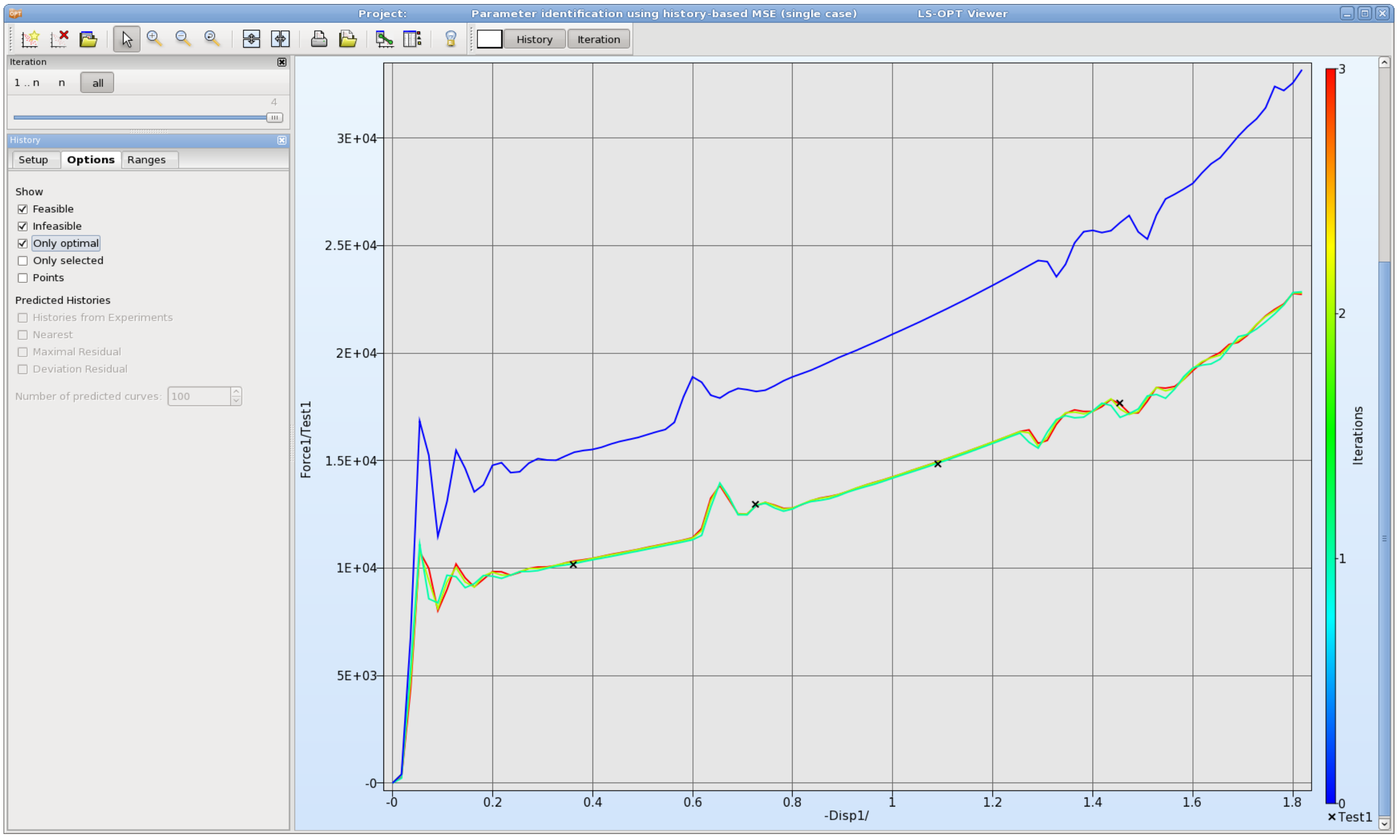
DIC Contours of uniaxial_exx_residual_DTW (Dynamic Time Warping)
Time = 22
Contours of testx
min=-0.257237, at node# 27453
max=0, at node# 19914

Figure 19-3: ε_{xx}-component of source (test) contour mapped using Dynamic Time Warping.

History

The history plot (Section 16.2.4) can be used to display the computed curves and the test curve in the same plot. There are several coloring options for the computed curves, e.g. the curves could be colored by the objective values (curve matching metric) to see if the curve matching metric works as expected.

Displaying the curves for all iterations, selecting the option *Only optimal* and coloring the curves by iteration visualizes the improvement of the optimal curves over the iterations.



Project: Parameter identification using history-based MSE (single case) LS-OPT Viewer

Iteration

1 .. n n all

4

History

Setup **Options** Ranges

Show

☑ Feasible

☑ Infeasible

☑ Only optimal

☐ Only selected

☐ Points

Predicted Histories

☐ Histories from Experiments

☐ Nearest

☐ Maximal Residual

☐ Deviation Residual

Number of predicted curves: 100

Figure 19-4: Target and computed curves, only optimal curves are displayed for all iterations

19.2. Sensitivity analysis

Responses can depend on many variables, and the computational effort of an optimization strongly depends on the number of variables. In most cases, only a few variables are significant.

Sensitivity analysis allows the user to determine the significance of design variables when computing a selected response. This helps to understand the simulation model and to reduce the design variables used in an optimization. The least significant ones can be de-selected to reduce the computational effort.

Metamodel-based tasks as well as direct sensitivity analysis is available in LS-OPT.

19.2.1. Metamodel-based Sensitivity Analysis

Two sensitivity measures are implemented in LS-OPT: Linear ANOVA and GSA/Sobol.

Both sensitivity measures are global in nature and are evaluated using the metamodel; hence the metamodel quality is essential to achieve reasonable sensitivity results.

ANOVA is a linear sensitivity measure, whereas GSA/Sobol is non-linear. Therefore, the results are comparable for linear metamodels. ANOVA depicts positive or negative influence, whereas GSA/Sobol just shows the absolute value. An advantage of GSA/Sobol is, that the values are normalized. Hence they can be summed up to determine the influence of a parameter on multiple responses, on a full load case, or on the entire optimization problem.

ANOVA is evaluated automatically if metamodels are available, to get GSA/Sobol values, select the *Global Sensitivities* option in the **Task** dialog (Section 4.11) or from the **Add** menu (Section 3.2).

To perform a sensitivity analysis, a global metamodel approximation should be used. Two approaches are described in the following sections.

DOE task

A global approximation can be achieved by selecting task *DOE*, Section 4.3. To get reasonable results, increase the *Number of Simulation Points (per Iteration per Case)* to at least 2*(*n*+1), where *n* is the number of variables. The greater the non-linearity of the response functions, the more points are needed to represent the nonlinearities. Hence the number of points is always a compromise between accuracy and computational effort.

Sequential

An approach for generating a metamodel to a specified prediction accuracy (using the PRESS metric, see Section 23.3.5) is to use an iterative method.

Select *Metamodel based Optimization* for the main task, and the *Sequential* strategy, Section 4.8.2. Here, the default *Number of Points per Iteration* can be used, because points are added sequentially. A nonlinear metamodel is recommended, e.g. Radial Basis Functions together with the Space Filling point selection scheme, Section 9.3.4.

An appropriate termination criterion for a sequential approach is *Response Accuracy Tolerance*, Section 13.1.2. Make sure to use the OR option and set the non-accuracy tolerances to 0. The number of iterations to be performed is again a compromise between accuracy and computational effort.

Viewer

This section describes some postprocessing options commonly used for a sensitivity analysis. Further options are described in Chapter 16.

Accuracy

Use the accuracy plot (Section 16.3.3) and the error measures displayed in the title to judge the quality of the metamodels.

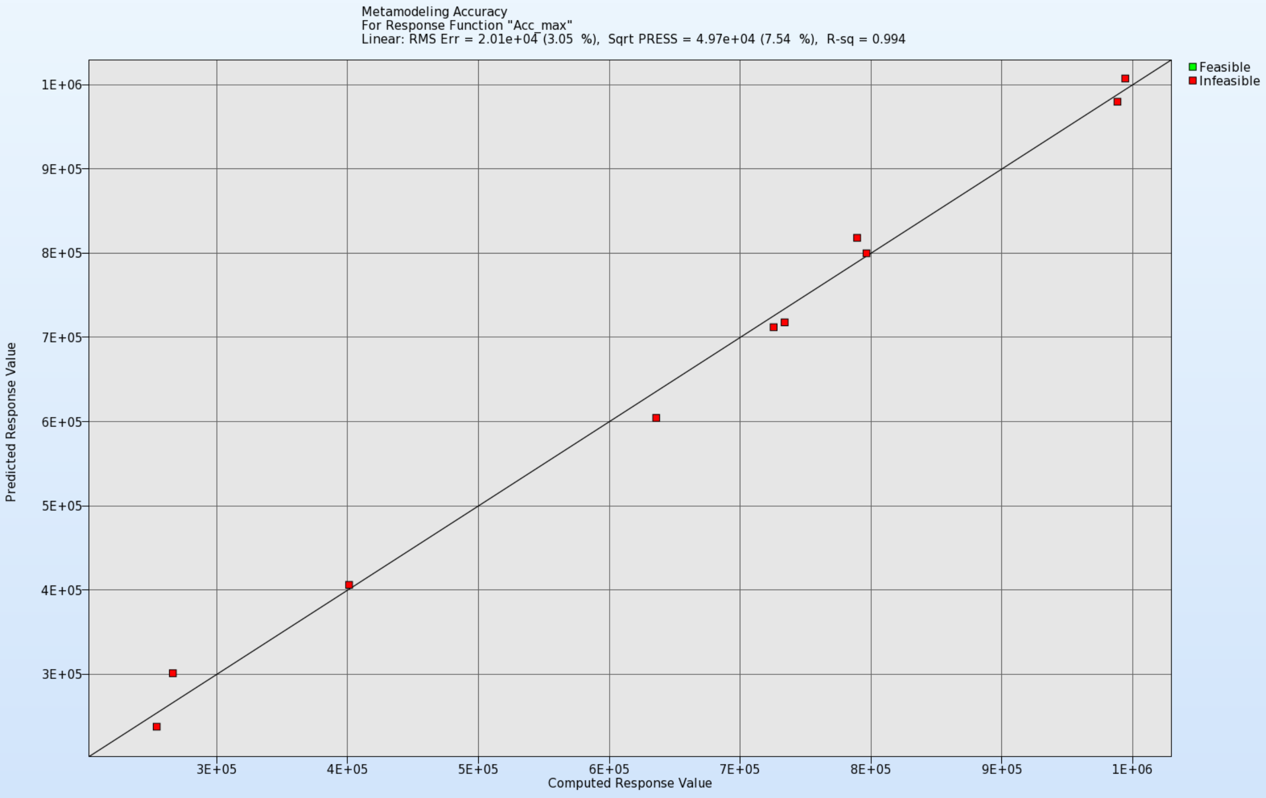


Figure 19-5: Accuracy plot; computed vs. predicted values; error measures are displayed in the title

Sensitivities

The sensitivity measures calculated by LS-OPT, Linear ANOVA and GSA/Sobol can be visualized in the Sensitivities plot (Section 16.3.4). By default, the values are sorted by significance, hence the ranking of the parameters can be directly taken from the order in the plots.

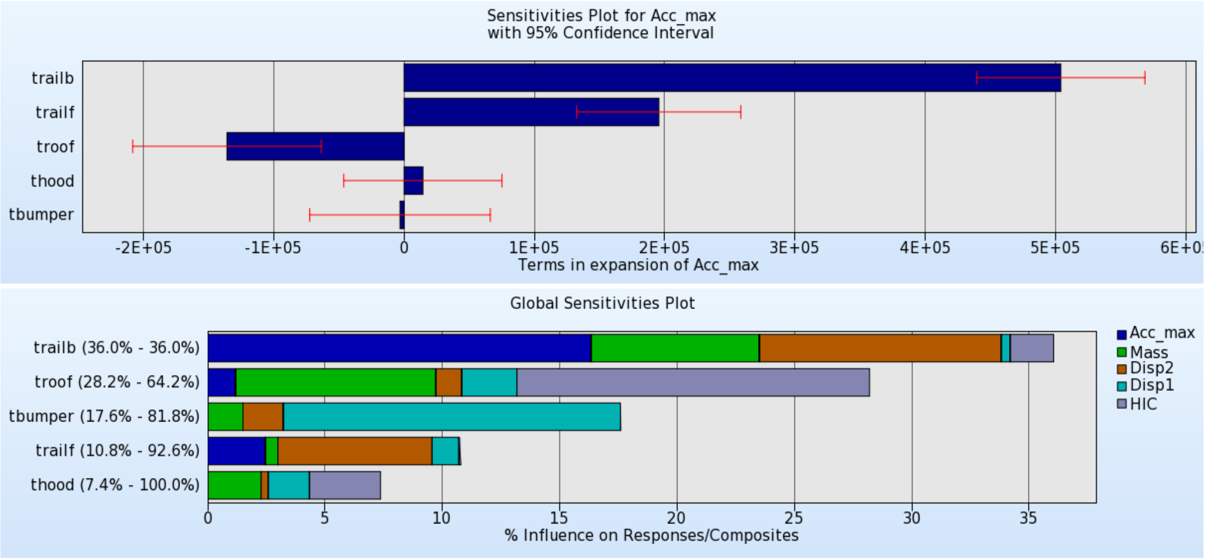
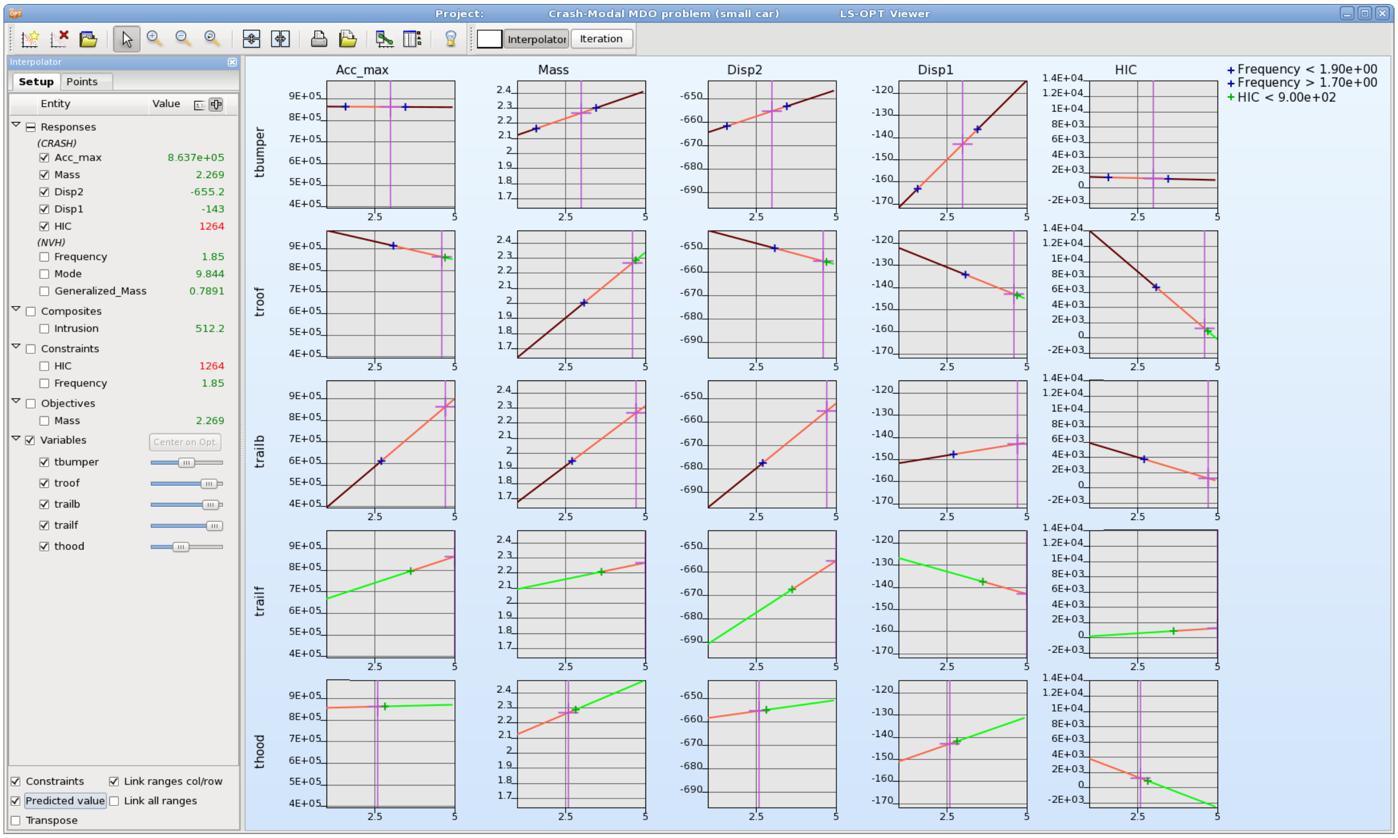


Figure 19-6: ANOVA values for a single response; Sobol values for multiple responses

Interpolator

The Interpolator plot (Section 16.3.2) displays 2D cross-sections of the metamodels in a matrix for selected responses and variables. Constraints and predicted values for a selected parameter combination can be visualized on the metamodel.



Interpolator

Setup | Points

Entity	Value
▽ ⊟ Responses	
(CRASH)	
☑ Acc_max	8.637e+05
☑ Mass	2.269
☑ Disp2	-655.2
☑ Disp1	-143
☑ HIC	1264
(NVH)	
□ Frequency	1.85
□ Mode	9.844
□ Generalized_Mass	0.7891
▽ □ Composites	
□ Intrusion	512.2
▽ □ Constraints	
□ HIC	1264
□ Frequency	1.85
▽ □ Objectives	
□ Mass	2.269
▽ ☑ Variables	Center on Opt.
☑ tbumper	
☑ troof	
☑ trailb	
☑ trailf	
☑ thood	

☑ Constraints ☑ Link ranges col/row
☑ Predicted value □ Link all ranges
□ Transpose

Figure 19-7: Interpolator plot: 2D surface plots for variables vs. responses; constraints on the metamodel and the predicted value for the selected parameter combination are displayed.

19.2.2. Taguchi analysis

Taguchi method is a direct single iteration optimization method that uses an Orthogonal Array experimental design, Section 9.3.6.

Taguchi method involves a careful selection of process parameters using Orthogonal Arrays to obtain the optimum results of the process. The idea of such a method is to analyze every variable effect on the result in order to choose the optimal design or to understand the impact of noise parameters on the system's robustness. However, the variable correlations cannot be separated from the main effect analyzes, unless the nature of interaction is explicitly specified.

The main steps involved in Taguchi method are:

1. Identify your problem objectives, variables and possible values.
2. Select the suitable Orthogonal Array.
3. Run the experiments.
4. Analyze the variable effect on the result and, if required, predict the optimum.

The Main and Interaction effect plots are used to visualize the variable effect, Section 16.2.6.

19.3. Multidisciplinary Design Optimization (MDO)

MDO is often used because in industry each design group typically has its own simulation tools, design criteria (constraints) and load cases. A different set of variables, constraints and objectives therefore needs to be used for each discipline.

The MDO capability in LS-OPT implies that the user has the option of assigning different variables, sampling types and job specification information to different cases or disciplines. Each case has to be defined with a unique Sampling (see Section 3.2.1).

Variables can be de-activated Sampling-wise in the **Sampling Matrix** tab (**Setup** dialog, Section 8.3). After each iteration, variables omitted from specific samplings will assume the global value. It is permissible to eliminate a set of variables across all Samplings, in which event they will remain constant during the optimization process.

Examples are available in Section 20.5.

19.4. Multi-objective optimization (MOO)

Design objectives are often in conflict. This implies that objectives cannot all be minimized to their single-objective minima (the so called Utopian solution) at the same time. In the mathematical sense multi-objective problems therefore have multiple solutions, typically defining a line or a surface in the space defined by the objectives (i.e. two-dimensional space for two objectives, etc.). In design optimization terminology such a solution is referred to as a Pareto Optimal Frontier (POF), or trade-off curve or surface. The POF curve can then be used by designers to choose a unique design which satisfies the needs of all the disciplines, although it is likely to be a compromise solution.

POF surfaces can be discontinuous.

To activate the POF feature, the option *Create Pareto Optimal Front* can be selected in the **Task** or **Optimization** dialog, Section 4.10. The option is only available if at least two objectives are defined.

The recommended optimization task and strategy for MOO is *Metamodel-based Optimization* using the **Single Iteration** or **Sequential** strategies, see Section 4.8.

19.4.1. Direct Genetic Algorithm

To calculate Pareto optimal solutions using the Direct Genetic Algorithm, select *Direct simulation Optimization* as main task, Section 4.4.

The advantage of using a direct task is, that it uses only simulation results to find the optimal value, hence there is no approximation error. The disadvantage is that the number of simulation runs needed to find an optimal value can be high. Therefore this task can only be used for small models or if sufficient computational resources are available.

19.4.2. Metamodel-based Genetic Algorithm

To calculate Pareto optimal solutions using a metamodel-based Genetic Algorithm, a global approximation is recommended. Select *Metamodel-based Optimization* as the main task, and use the strategy *Single iteration* or *Sequential* together with a nonlinear metamodel, e.g. Radial Basis Functions or FeedForward Neural Nets.

Because Pareto solutions are often global in nature (spans a significant part of the design space), global metamodel accuracy is typically required. This may be difficult to achieve with a large number of design variables. In this case the Direct GA (which will also be expensive) is the only remaining option.

19.4.3. Viewer

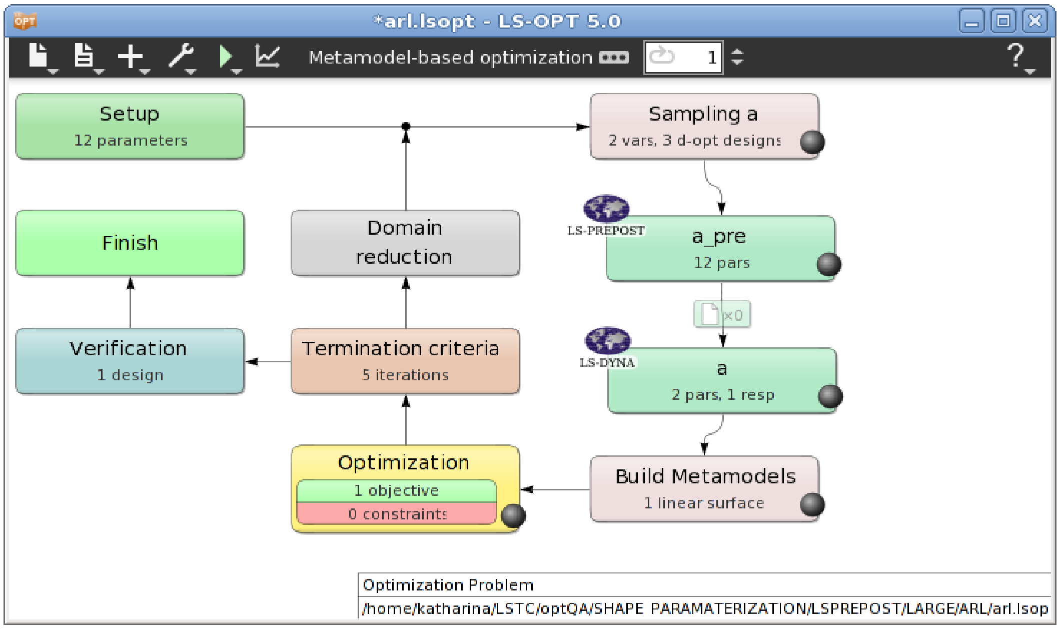
Various plot types that are available for the visualization of Pareto optimal solutions described in Section 16.5 can be used to explore those solutions and select the appropriate optimal solution that fits best to the application.

19.5. Shape Optimization

To implement geometrical parameters in LS-OPT, an interface to a preprocessor has to be used. The available interfaces, which include a user-defined option, are described in Chapter 5. The process chain to be optimized is at least a two-stage process including a preprocessor and a solver, Figure 19-8. Additional parameters can be defined in the solver input file. The preprocessor output is used as solver input. For LS-DYNA, the output can be used as an include file, specified in the main input file.

Some preprocessors allow the user to generate multiple output files which can be used in multiple parallel simulation stages by using a file operation function between the stages (see Section 3.2.2) to copy the selected preprocessor output files.

The recommended task and strategy for single objective optimization is *Metamodel based Optimization* and *Sequential with Domain Reduction*, Section 4.8.3.

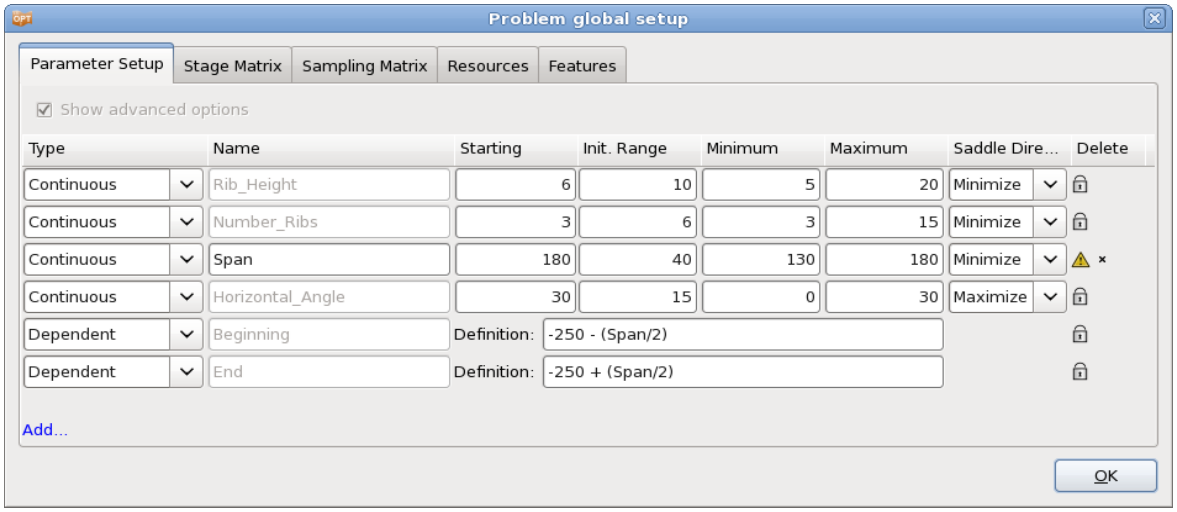


Optimization Problem
/home/katharina/LSTC/optQA/SHAPE PARAMATERIZATION/LSPREPOST/LARGE/ARL/arl.lsop

Figure 19-8: Possible setup for a shape optimization. a_pre interfaces with a preprocessor that generates the geometry of the model depending on parameters.

19.6. Worst-case design

The default setting in LS-OPT is that all design variables are treated as minimization variables. This means that the objective function is minimized (or maximized) with respect to all the variables. Maximization variables are selected in the **Setup** dialogs **Parameter Setup** panel (see Figure 19-9) by toggling the required variables from ‘Minimize’ to ‘Maximize’ in the *Saddle Direction* menu. This option is only available if *Show advanced options* is selected (Section 8.1.10).



☑ Show advanced options

Type	Name	Starting	Init. Range	Minimum	Maximum	Saddle Dire...	Delete
Continuous	Rib_Height	6	10	5	20	Minimize	🔒
Continuous	Number_Ribs	3	6	3	15	Minimize	🔒
Continuous	Span	180	40	130	180	Minimize	⚠ ×
Continuous	Horizontal_Angle	30	15	0	30	Maximize	🔒
Dependent	Beginning	Definition:	-250 - (Span/2)				🔒
Dependent	End	Definition:	-250 + (Span/2)				🔒

Figure 19-9: Parameter definition for a worst-case design optimization

19.7. Multilevel Optimization

In multilevel optimization, the optimization problem is solved in parts at two (usually) or more levels. Each sublevel optimizes a subset of the variable set while maintaining constant values for the variables belonging to preceding levels. Multilevel optimization can be used to group variables into the different levels to make the problem easier to solve. For example, a gradient based optimizer may be used for some of the variables while a zero order method is used for the others. Similarly, metamodels may be constructed with some of the variables while the rest are optimized using a direct method. In LS-OPT, this is performed using the LS-OPT stage and by specifying some of the inner level variables as *Transfer Variables* (Section 5.3.9).

The multilevel optimization process in LS-OPT can be briefly summarized as follows. For the sake of simplicity, the summary is provided for the case consisting of two levels.

1. *Input File preparation for LS-OPT stage of outer level setup*: The input file for the LS-OPT stage is an .lsopt file itself. Therefore, preparing this file involves exactly the same steps as any single level problem setup. While this file is an input file for the outer level, it is also the LS-OPT setup file for solving the inner level problem. As already mentioned, the inner level optimization is performed with respect to a subset of the variables while the rest are optimized in the outer level. Therefore, these other parameters are constants for the inner level. The LS-OPT GUI is used to prepare the .lsopt file; the inner level free variables are set as Continuous or Discrete Variables, but the rest are set as Transfer Variables and are treated as constants at this level.

2. *Stage setup for outer level*: See Section 5.3.9.

3. *Response definitions for outer level*: See Section 6.15.1.
4. *Global Setup for outer level*: Once a .lsopt file parameterized with *Transfer Variables* is specified as the LS-OPT stage input file in the outer level, the outer level LS-OPT stage automatically detects these parameters and they are added to the *Global Setup* as constants. These can then be set as Continuous or Discrete Variables by the user and thus, they become outer level variables (Figure 5-10 Section 5.3.9).
5. *Running the optimization*: The outer level optimization is started by pressing the run button in the GUI or from command line, which leads to the creation of a design of experiments for the outer level variables. A run directory is created for each outer level sample. The LS-OPT stage input file (i.e. the inner level .lsopt setup) is copied to each of these directories and named as LsoOpt.inp by default. The Transfer Variable values in a particular run directory are set as the corresponding outer level sample's variable values. Once the Transfer Variable values are set, they are treated as constants within a run directory and the inner optimization is carried out with respect to the free inner level variables. The optimized inner level entities are then extracted as sample responses at the outer level, thus providing the response values at each outer level sample. The outer level optimization is then carried out with respect to the remaining variables.

II – Examples

20. Examples − Optimization

20.1. Two-bar truss (3 variables)

This example has the following features:

- A user-defined solver is used.
- Extraction is performed using user-defined scripts.
- First- and second-order response surface approximations are compared.
- The effect of subregion size is investigated.
- The design optimization process is automated.

20.1.1. Description of problem

This example problem as shown in Figure 20-1 has one geometric and two element sizing variables.

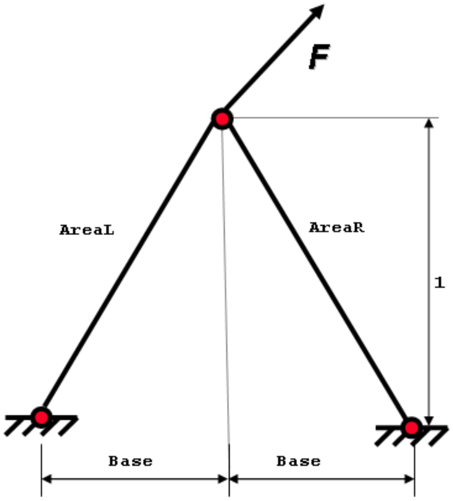


Figure 20-1: The two-bar truss example

The problem is statically determinate. The forces on the members depend only on the geometric variable. Only one load case is considered: $\boldsymbol{F} = (F_x,F_y) = (24.8kN,\ 198.4kN)$.

There are three design variables: *AreaL* and *AreaR*, the cross-sectional areas of the bars, and *Base*, half of the distance (*m*) between the supported nodes. The lower bounds on the variables are $0.2cm^2$ and $0.1m$, respectively. The upper bounds on the variables are 4.0cm^2 and 1.6m, respectively. The objective function is the weight of the structure

$$f(x) = \frac{1}{2}(AreaL + AreaR)\sqrt{1 + Base^2}.$$

The absolute values of the stresses in the members are constrained to be less than 100 MPa,

$$-1 \leq \sigma_1(x) = 0.124 \cdot \sqrt{1 + Base^2\left(\frac{8}{AreaL} + \frac{1}{Base \cdot AreaL}\right)} \leq 1,$$

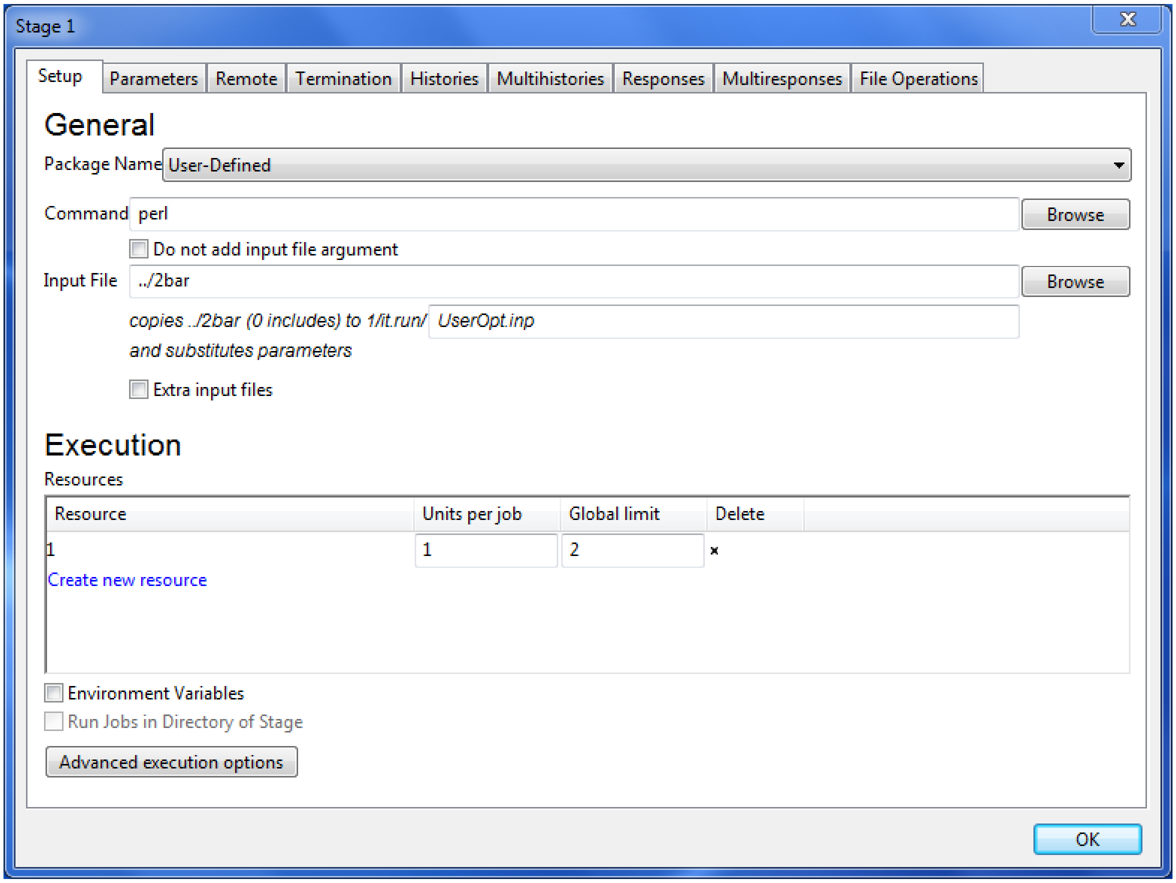
$$-1 \leq \sigma_2(x) = 0.124 \cdot \sqrt{1 + Base^2\left(\frac{8}{AreaR} - \frac{1}{Base \cdot AreaR}\right)} \leq 1.$$

The Perl program `2bar` printed below simulating the weight response and stress response respectively is used as solver. Note the output of the string `"N o r m a l"` so that the completion status may be recognized.

2bar:

```
#!/usr/bin/perl
#
#  2BAR truss
#
#  Open output files (database)
#     Each response is placed in its own file
#
   open(WEIGHT,">Weight");
   open(STRESSL,">StressL");
   open(STRESSR,">StressR");
#
#--Compute the responses
#
   $length = sqrt(1 + <<Base>>*<<Base>>);
   $cos  = <<Base>>/$length;
   $sin  = 1/$length;
   $Weight = (<<AreaL>> + <<AreaR>>) * sqrt(1 + <<Base>>*<<Base>>) /2;
   $StressL = ( 24.8/$cos + 198.4/$sin)/<<AreaL>>/200;
   $StressR = (-24.8/$cos + 198.4/$sin)/<<AreaR>>/200;
#
#--Write results to database
#
   print WEIGHT $Weight,"\n";
   print STRESSL $StressL,"\n";
   print STRESSR $StressR,"\n";
#*****************************************
#--Signal normal termination
#*****************************************
   print "N o r m a l\n";
```

Since the parameters are defined in `2bar` using the LS-OPT parameter format <<>>, the script is defined as the solver input file, while the solver command is `perl`, Figure 20-2. The response values are written to files that are used to define the FILE responses in LS-OPT, Figure 20-3.



General

Package Name: User-Defined

Command: perl

□ Do not add input file argument

Input File: ../2bar

copies ../2bar (0 includes) to 1/it.run/ UserOpt.inp
and substitutes parameters

□ Extra input files

Execution

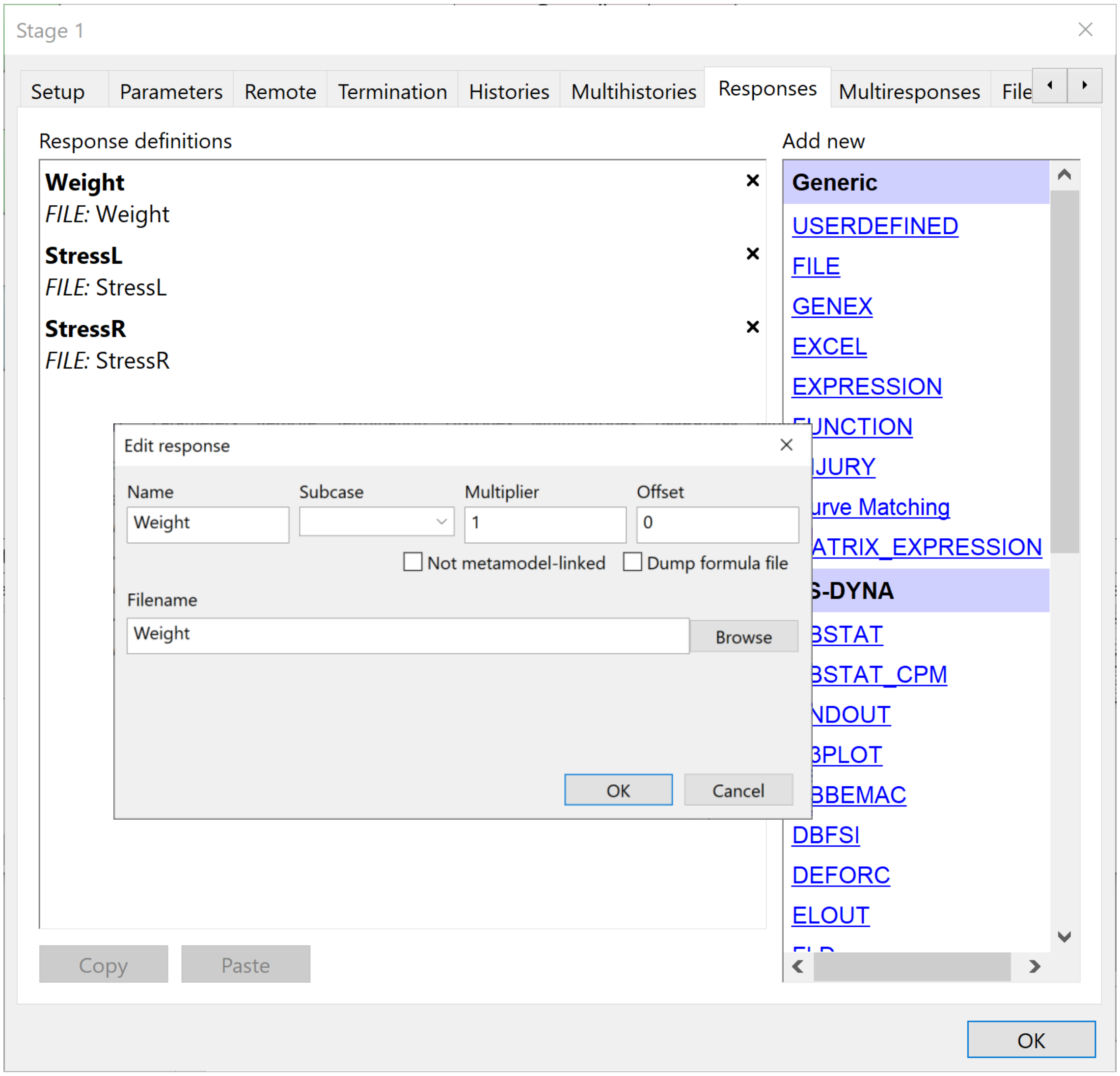
Resources

Resource	Units per job	Global limit	Delete
1	1	2	×

□ Environment Variables

□ Run Jobs in Directory of Stage

Figure 20-2: Stage dialog Setup for a user-defined solver. Parameters are specified in the input file using the LS-OPT parameter format <<>>.



Response definitions

Weight
FILE: Weight

StressL
FILE: StressL

StressR
FILE: StressR

Name	Subcase	Multiplier	Offset
Weight		1	0

□ Not metamodel-linked □ Dump formula file

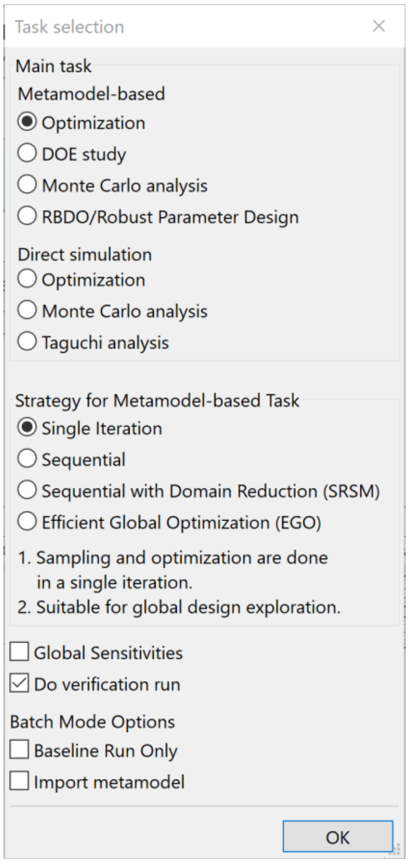
Filename: Weight

Figure 20-3: User-defined response definitions

The problem is solved using metamodel based optimization, Figure 20-4. In Sections 20.1.2 to 20.1.4, a typical semi-automated optimization procedure is illustrated. The last subsection 20.1.5 shows how an automated procedure can be specified for this example problem.

20.1.2. A first approximation using linear response surfaces

To get a first rough approximation of the problem, a single iteration is run, Figure 20-4.



Task selection

Main task

Metamodel-based

● Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

○ Taguchi analysis

Strategy for Metamodel-based Task

● Single Iteration

○ Sequential

○ Sequential with Domain Reduction (SRSM)

○ Efficient Global Optimization (EGO)

1. Sampling and optimization are done in a single iteration.
2. Suitable for global design exploration.

□ Global Sensitivities

☑ Do verification run

Batch Mode Options

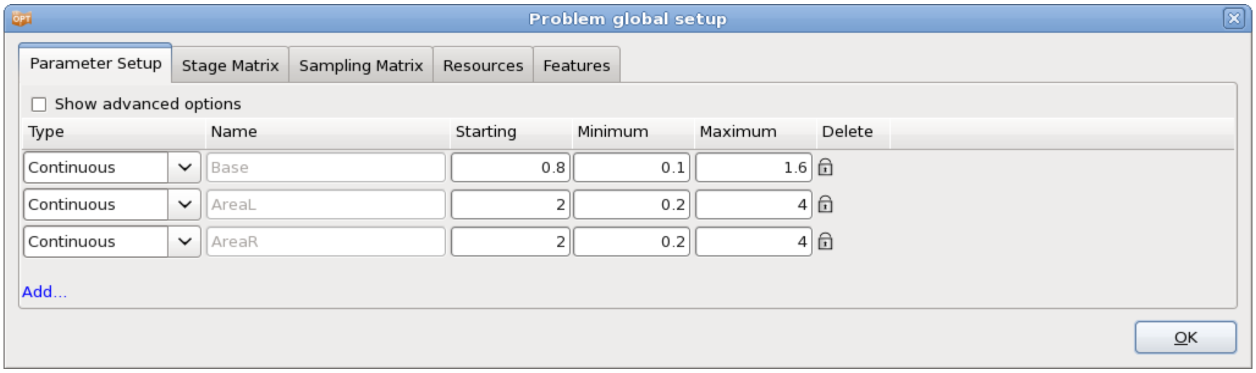
□ Baseline Run Only

□ Import metamodel

Figure 20-4: Task dialog; Selection for a metamodel base optimization using a single iteration.

The parameter setup is defined in the **Setup** dialog. The type of each parameter is set to continuous. A design space defined by minimum and maximum and a starting value is then specified for each parameter, Figure 20-5. The starting values are used for the initial design.

The **Sampling** dialog, allows for setting the metamodel and point selection, Figure 20-6. To get a first rough approximation of the problem, the metamodel type is chosen to be a linear polynomial. The default number of points is automatically adapted to the number of variables and the metamodel type.

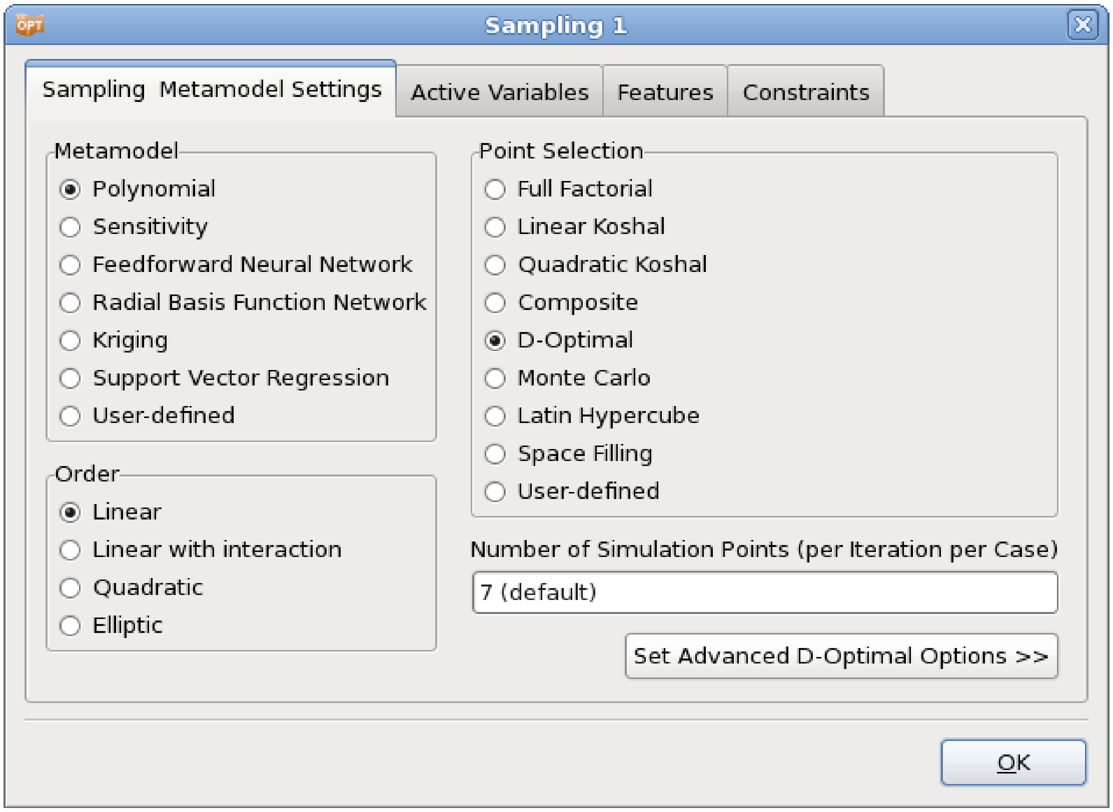


Problem global setup

□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	Base	0.8	0.1	1.6	🔒
Continuous	AreaL	2	0.2	4	🔒
Continuous	AreaR	2	0.2	4	🔒

Figure 20-5: Parameter Setup; specification of design space and starting values or all parameters



Sampling 1

Sampling Metamodel Settings | Active Variables | Features | Constraints

Metamodel
- ● Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Order
- ● Linear
- ○ Linear with interaction
- ○ Quadratic
- ○ Elliptic

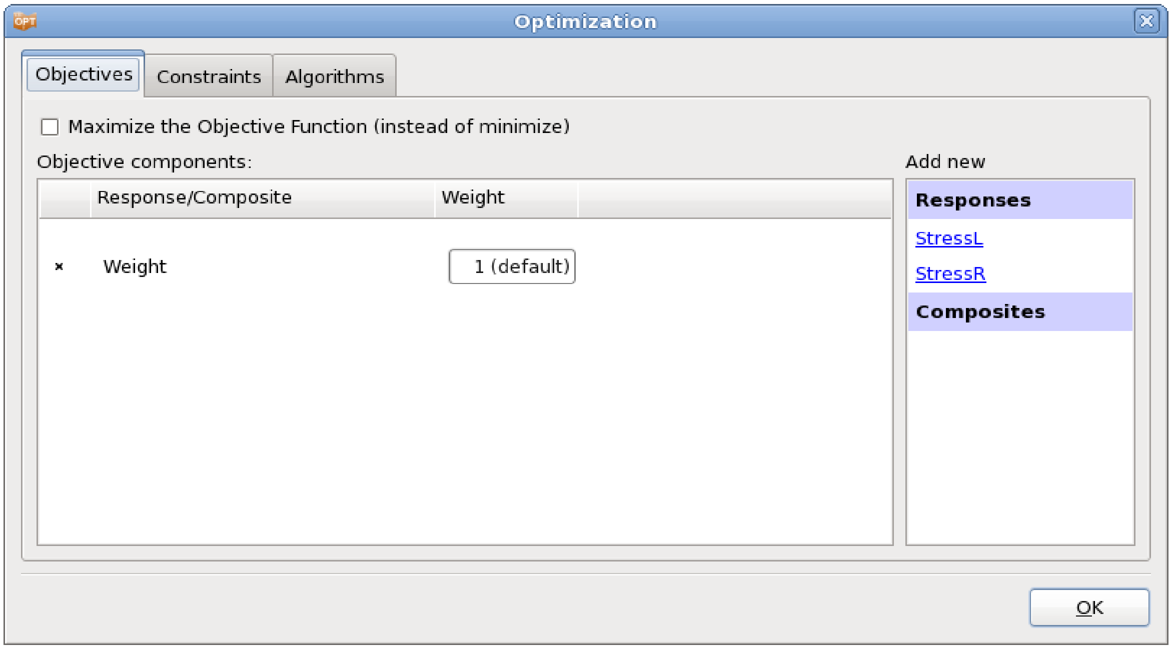
Point Selection
- ○ Full Factorial
- ○ Linear Koshal
- ○ Quadratic Koshal
- ○ Composite
- ● D-Optimal
- ○ Monte Carlo
- ○ Latin Hypercube
- ○ Space Filling
- ○ User-defined

Number of Simulation Points (per Iteration per Case): 7 (default)

Set Advanced D-Optimal Options >>

OK

Figure 20-6: Sampling and Metamodel; Select metamodel type Polynomial with order Linear; use the defaults for Point Selection and number of points



Optimization

Objectives | Constraints | Algorithms

□ Maximize the Objective Function (instead of minimize)

Objective components:

	Response/Composite	Weight
×	Weight	1 (default)

Add new

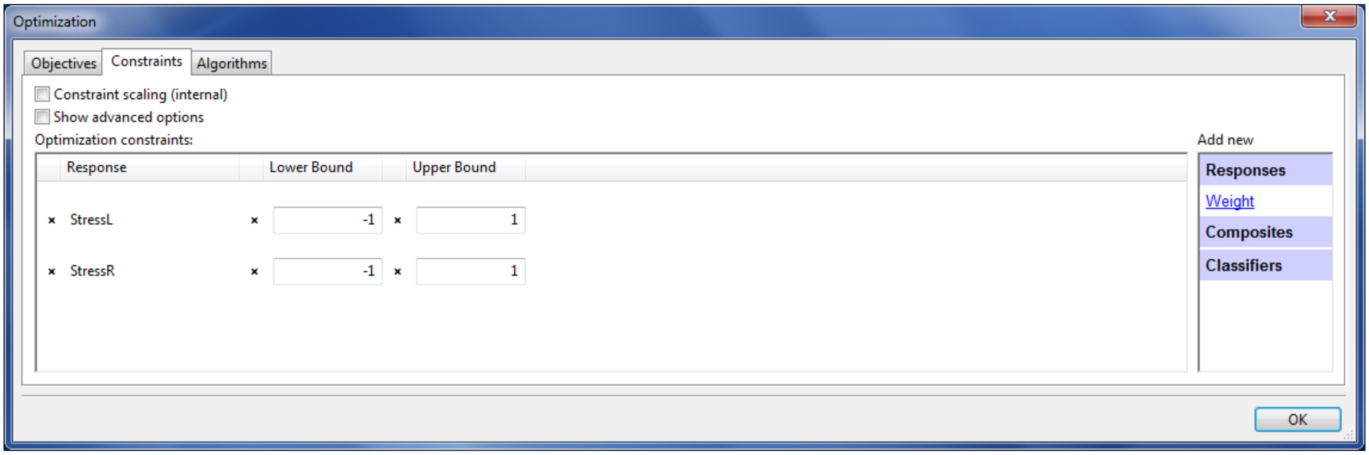
Responses
- StressL
- StressR

Composites

OK

Figure 20-7: Objectives; select the previously defined response Weight from the list on the right.

Open the **Optimization** dialog to define the optimization problem. To specify the objective function, select the previously defined *Weight* response from the list on the right in the **Objectives** tab, Figure 20-7. To define constraints, switch to the **Constraints** tab and select the previously defined responses *StressL* and *StressR* from the list on the right and enter the respective lower and upper bounds, Figure 20-8.



□ Constraint scaling (internal)
□ Show advanced options

Optimization constraints:

	Response		Lower Bound		Upper Bound
×	StressL	×	-1	×	1
×	StressR	×	-1	×	1

Figure 20-8: Constraints; select the respective responses from the list on the right and specify lower and upper bounds.

Results

The accuracy of the response surfaces can be illustrated by plotting the predicted results vs. the computed results using the **Accuracy** plot (Figure 20-9 and Figure 20-10). The error measures RMS, SPRESS and R^2 are displayed in the title of the plot.

The R^2 values are large. However the prediction accuracy (Sqrt PRESS), especially for the stresses, seems to be poor, so that either a higher order approximation or a smaller region of interest will be required.

Nevertheless an improved design is predicted with the constraint values (stress) changing from severely violated approximate values to active constraint, Table 20-1. Due to inaccuracy, the actual constraint values of the optimum differ, but also the computed constraints are not violated. The weight values have improved for both computed and predicted. Feasible and infeasible regions in the design space as well as the computed and predicted optimum are displayed in Figure 20-11.

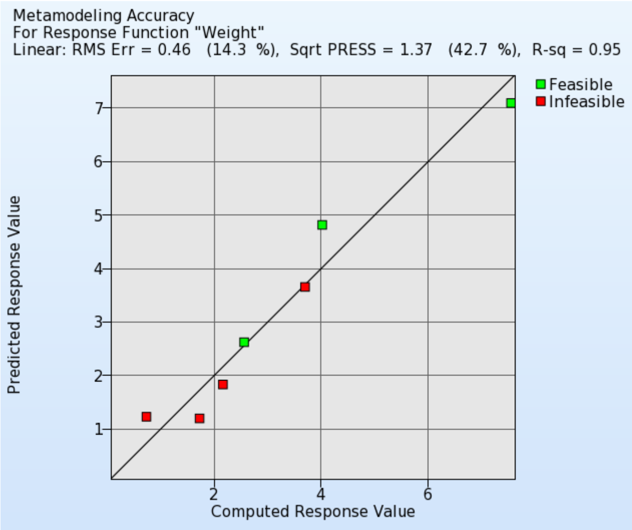


Figure 20-9: Accuracy of linear metamodel for response "Weight"

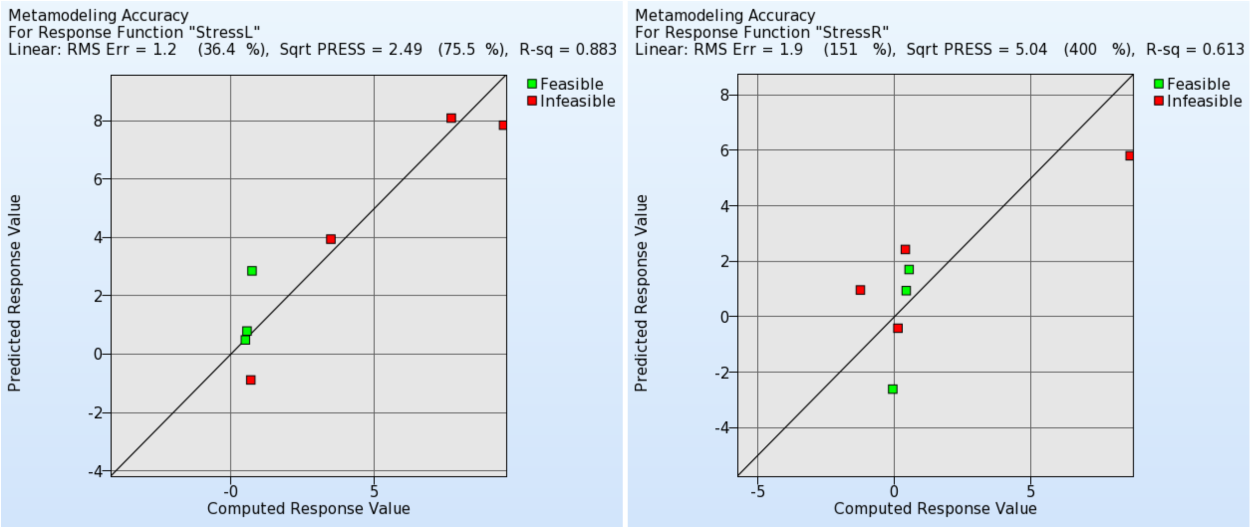


Figure 20-10: Accuracy of linear metamodel of responses "StressL" and "StressR"

Table 20-1: Comparison of baseline run and optimum (single iteration, linear metamodel)

	Baseline (Computed)	**Baseline (Predicted)**	**1. Opt (Computed)**	**1. Opt (Predicted)**
Weight	2.56	2.62	1.53	0.85
StressL	0.73	2.85	0.92	0.99
StressR	0.53	1.70	-0.41	1.00

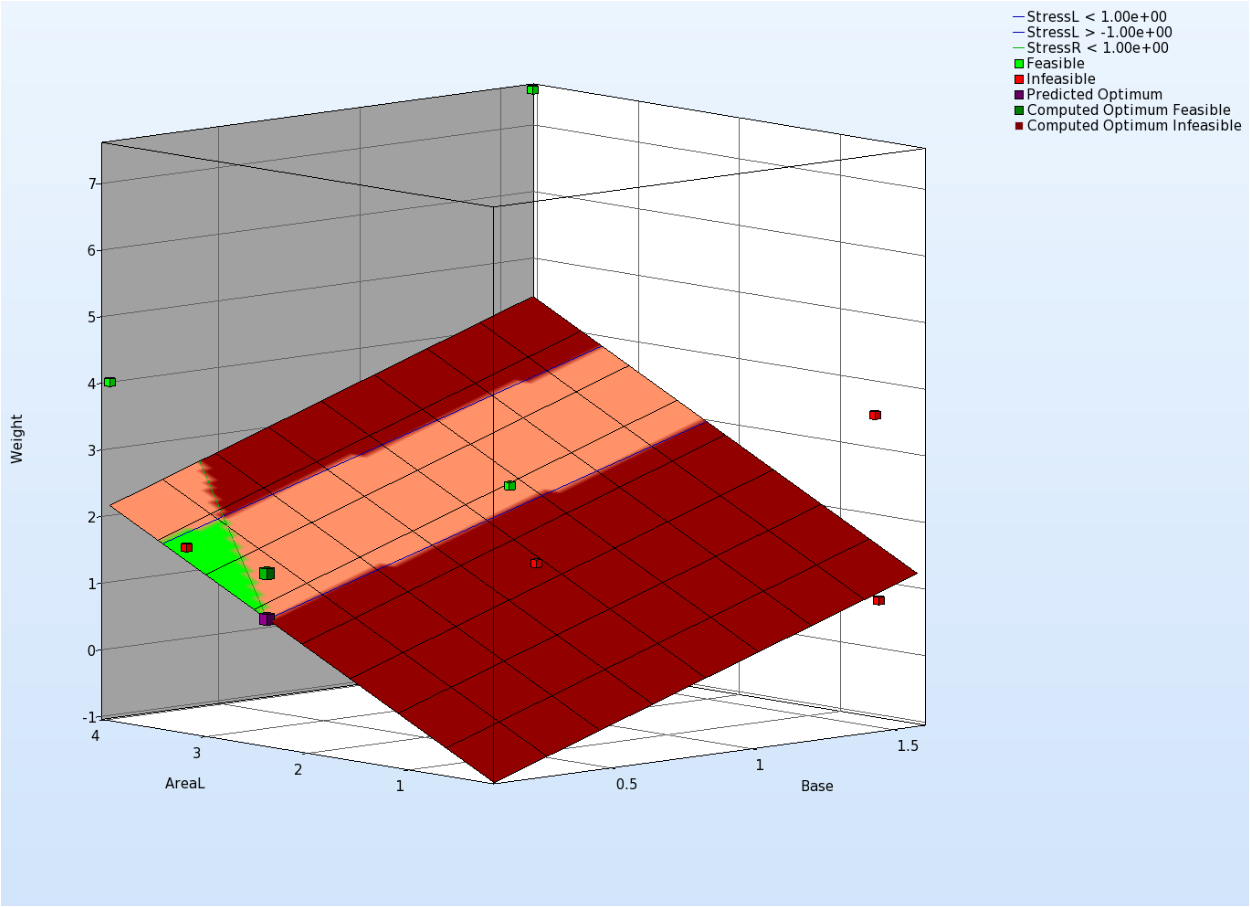
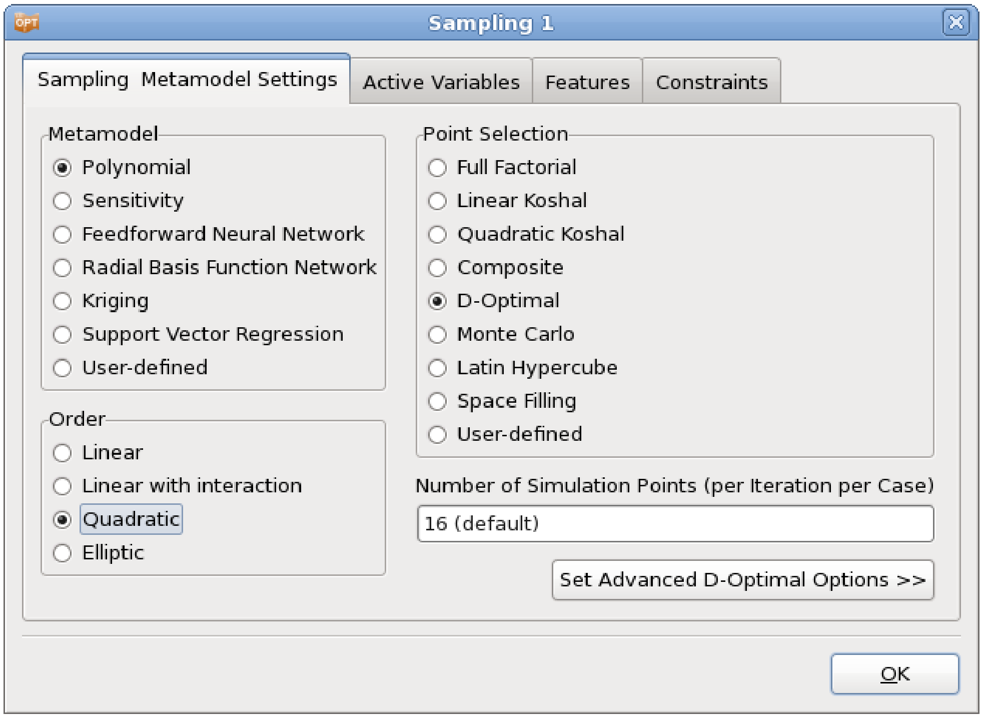


Figure 20-11: Surface plot for objective function Weight; constraints are displayed on the metamodel.

20.1.3. Updating the approximation to second order

To improve the accuracy of the metamodels, a second run is conducted using a quadratic approximation. Switch the metamodel order in the **Sampling** dialog to quadratic, Figure 20-12. The number of points will automatically update.


Sampling 1
Sampling Metamodel Settings
Active Variables
Features
Constraints
Set Advanced D-Optimal Options >>
OK

Metamodel

- ● Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Order

- ○ Linear
- ○ Linear with interaction
- ● Quadratic
- ○ Elliptic

Point Selection

- ○ Full Factorial
- ○ Linear Koshal
- ○ Quadratic Koshal
- ○ Composite
- ● D-Optimal
- ○ Monte Carlo
- ○ Latin Hypercube
- ○ Space Filling
- ○ User-defined

Number of Simulation Points (per Iteration per Case)

16 (default)

Figure 20-12: Sampling dialog settings for a quadratic approximation

Results

The approximation results have improved considerably, but the stress approximation is still poor. The fit is illustrated below in Figure 20-13 and Figure 20-14.

An improved design is predicted with the approximate constraint values (stress) becoming active, Table 20-2. Due to inaccuracy, the actual *StressR* value of the optimum is infeasible. Feasible and infeasible regions in the design space as well as the computed and predicted optimum are displayed in Figure 20-15.

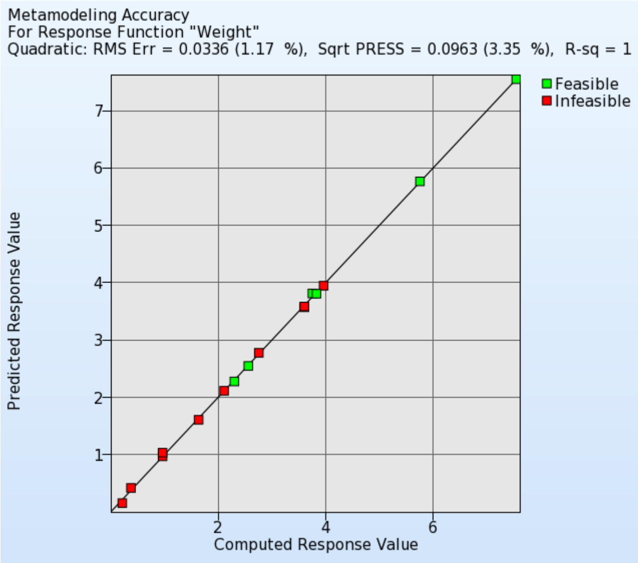


Figure 20-13: Accuracy of quadratic metamodel for response "Weight"

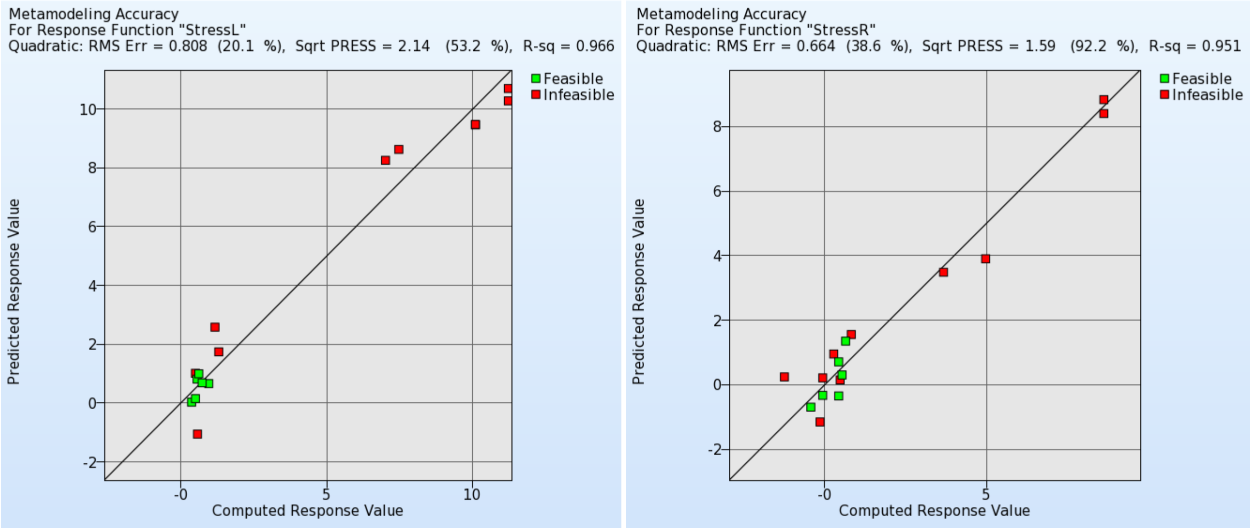


Figure 20-14: Accuracy of quadratic metamodel of responses "StressL" and "StressR"

Table 20-2: Comparison of baseline run and optimum (single iteration, quadratic metamodel)

	Baseline (Computed)	**Baseline (Predicted)**	**1. Opt (Computed)**	**1. Opt (Predicted)**
Weight	2.56	2.54	1.05	1.09
StressL	0.73	0.69	0.86	1.00
StressR	0.53	0.30	2.12	1.00

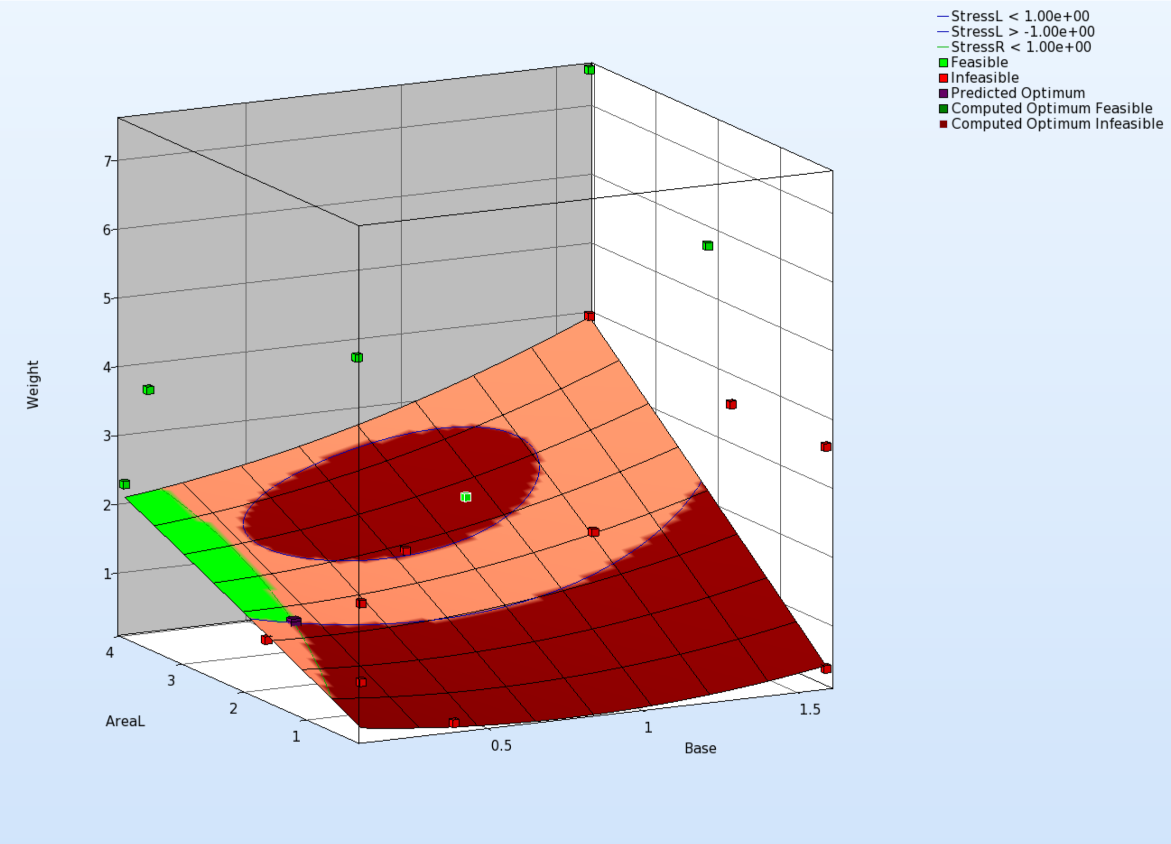
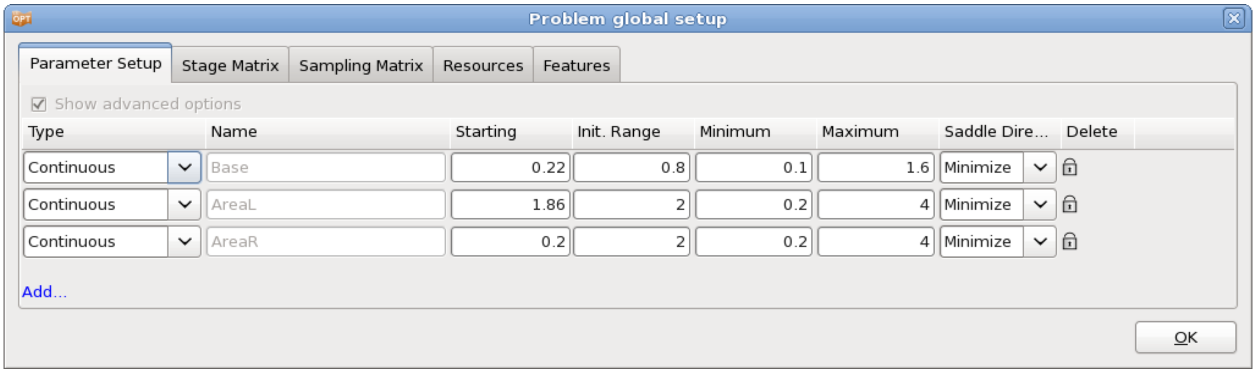


Figure 20-15: Surface plot for objective function weight; constraints are displayed on the metamodel.

20.1.4. Reducing the region of interest for further refinement

It seems that further accuracy can only be obtained by reducing the size of the subregion. In the following analysis, the current optimum (0.22, 1.86, 0.2) was used as a starting point while the region of interest was cut in half. The order of the approximation is quadratic. The required modifications are illustrated in Figure 20-16.



Type	Name	Starting	Init. Range	Minimum	Maximum	Saddle Dire...	Delete
Continuous	Base	0.22	0.8	0.1	1.6	Minimize	🔒
Continuous	AreaL	1.86	2	0.2	4	Minimize	🔒
Continuous	AreaR	0.2	2	0.2	4	Minimize	🔒

Figure 20-16: Reducing the design space by specifying an initial range; the starting values are the optimal values found in the previous approach.

Results

The approximations are significantly improved, Figure 20-17 and Figure 20-18.

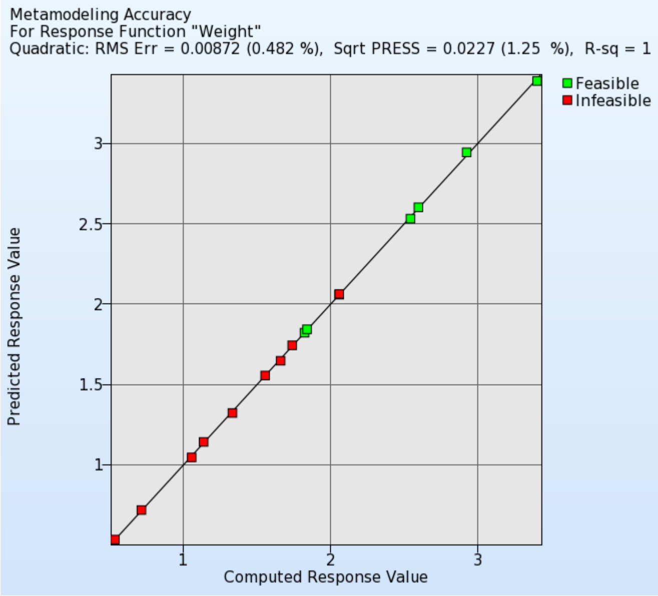


Figure 20-17: Accuracy of quadratic metamodel in reduced design space for response "Weight"

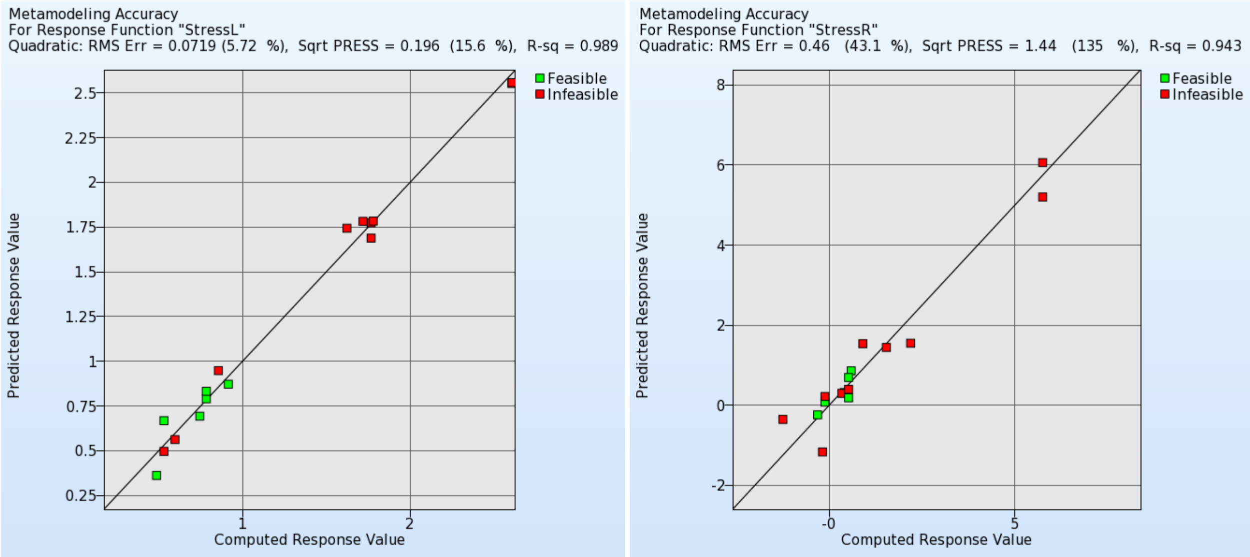


Figure 20-18: Accuracy of quadratic metamodel in reduced design space of responses "StressL" and "StressR"

The results are displayed in Table 20-3. An improved design is predicted with the approximate constraint values (stress) becoming active. Due to inaccuracy, the actual constraint values of the optimum are feasible. This value is now much closer to the value of the simulation result. For the optimal weight value, computed and predicted is the same.

Feasible and infeasible regions in the design space as well as the computed and predicted optimum are displayed in Figure 20-19.

Table 20-3: Comparison of baseline run and optimum (single iteration, quadratic metamodel, reduced design space)

	Baseline (Computed)	**Baseline (Predicted)**	**1. Opt (Computed)**	**1. Opt (Predicted)**
Weight	1.05	1.04	1.12	1.12
StressL	0.86	0.95	0.96	1.00
StressR	2.19	1.55	0.38	1.00

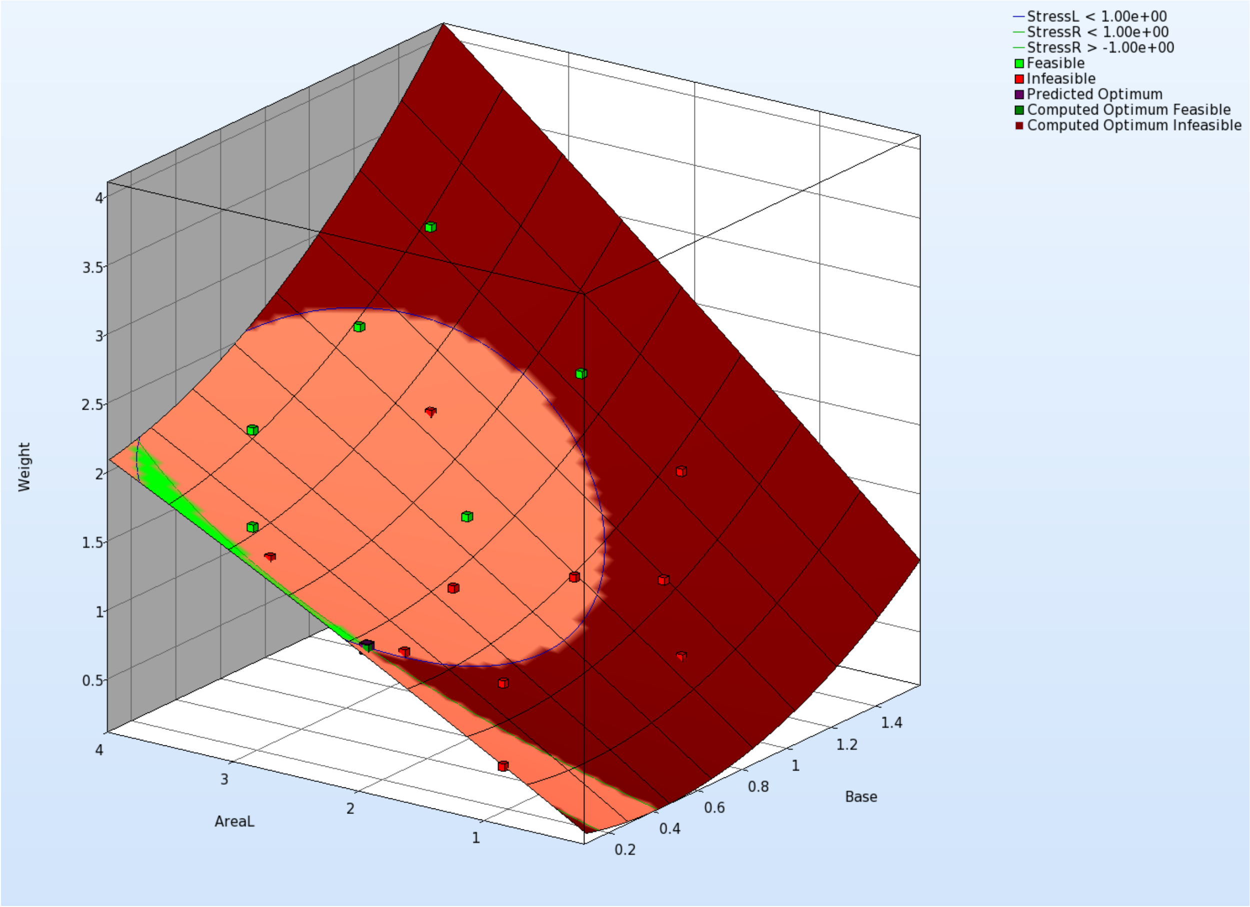
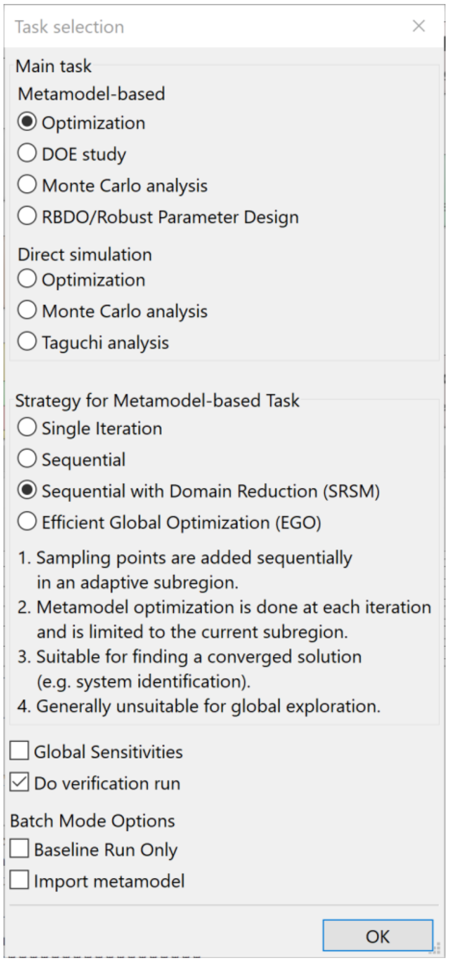


Figure 20-19: Surface plot for objective function weight; constraints are displayed on the metamodel.

20.1.5. Automating the design process

This section illustrates the automation of the design process of improving the accuracy of the metamodels by reducing the design space for both a linear and a quadratic response surface approximation order by using the strategy: sequential with domain reduction, Figure 20-20. 10 iterations are performed for the linear approximation, Figure 20-21, with only 5 iterations performed for the more expensive quadratic approximation.



Task selection

Main task

Metamodel-based

● Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

○ Taguchi analysis

Strategy for Metamodel-based Task

○ Single Iteration

○ Sequential

● Sequential with Domain Reduction (SRSM)

○ Efficient Global Optimization (EGO)

1. Sampling points are added sequentially in an adaptive subregion.
2. Metamodel optimization is done at each iteration and is limited to the current subregion.
3. Suitable for finding a converged solution (e.g. system identification).
4. Generally unsuitable for global exploration.

□ Global Sensitivities

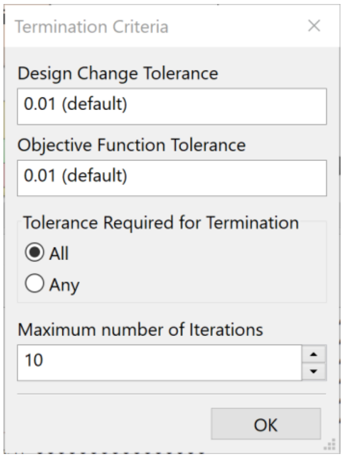
☑ Do verification run

Batch Mode Options

□ Baseline Run Only

□ Import metamodel

Figure 20-20: Task dialog; select strategy SRSM to automate the process.



Termination Criteria

Design Change Tolerance

0.01 (default)

Objective Function Tolerance

0.01 (default)

Tolerance Required for Termination

● All

○ Any

Maximum number of Iterations

10

Figure 20-21: Termination criteria; select 10 iterations for linear, 5 for quadratic approach

Results

The final results of the two types of approximations are displayed in Table 20-4. The optimization histories have been plotted to illustrate convergence in Figure 20-22 and Figure 20-23. Note that the more accurate but more expensive quadratic approximation converges in about 3 design iterations (48 simulations), while it takes about 7 iterations (49 simulations) for the objective of the linear case to converge. In general, the lower the order of the approximation, the more iterations are required to refine the optimum.

Table 20-4: Summary of final computed results (2-bar truss)

	Linear	**Quadratic**
Number of iterations	10	5
Number of simulations	71	81
AreaL	1.719	1.788
AreaR	0.304	0.200
Base	0.177	0.173
Weight	1.027	1.008
StressL	1.000	0.971
StressR	0.976	1.386

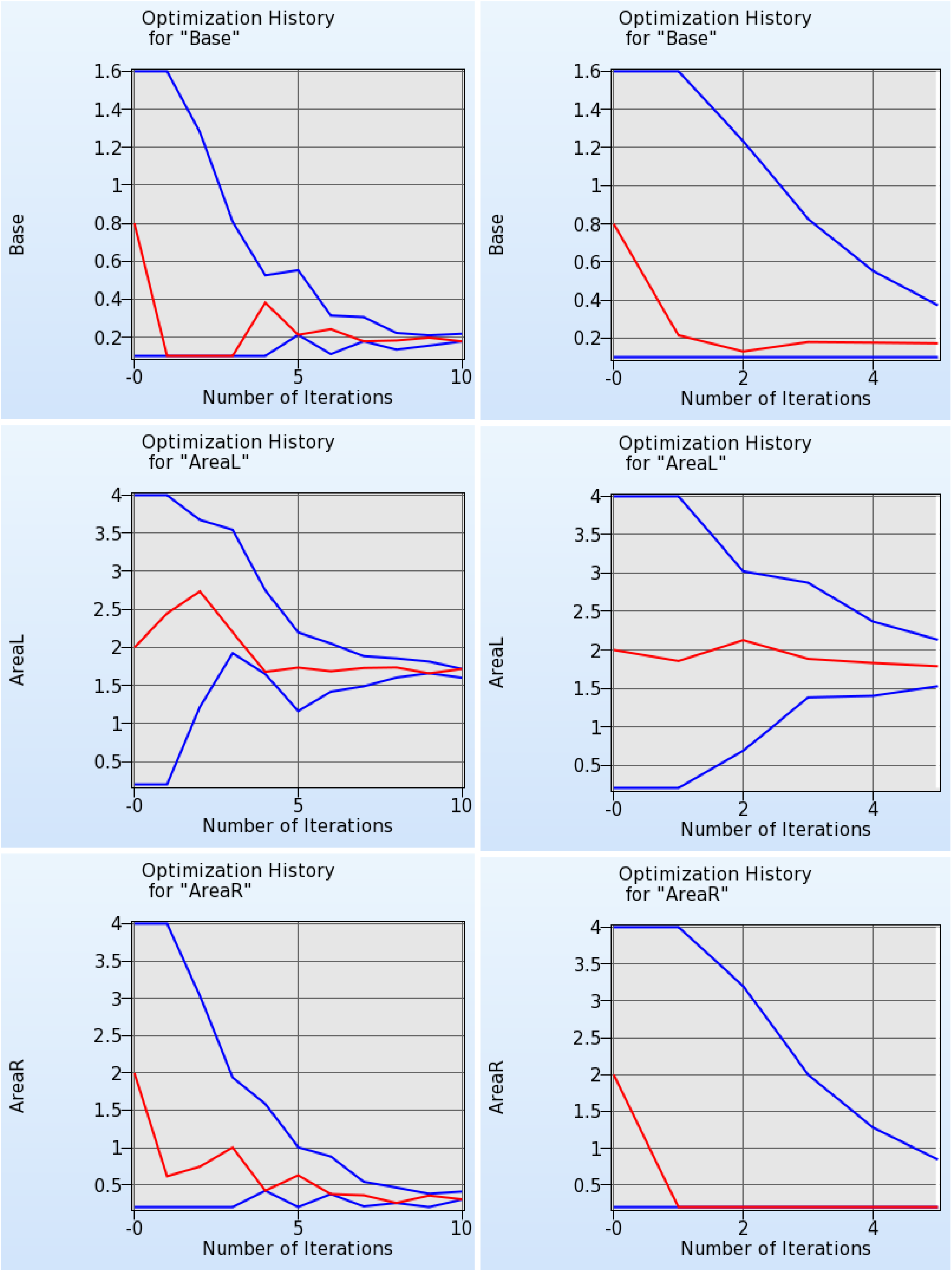


Figure 20-22: Optimization history of design variables; linear (left) and quadratic (right)

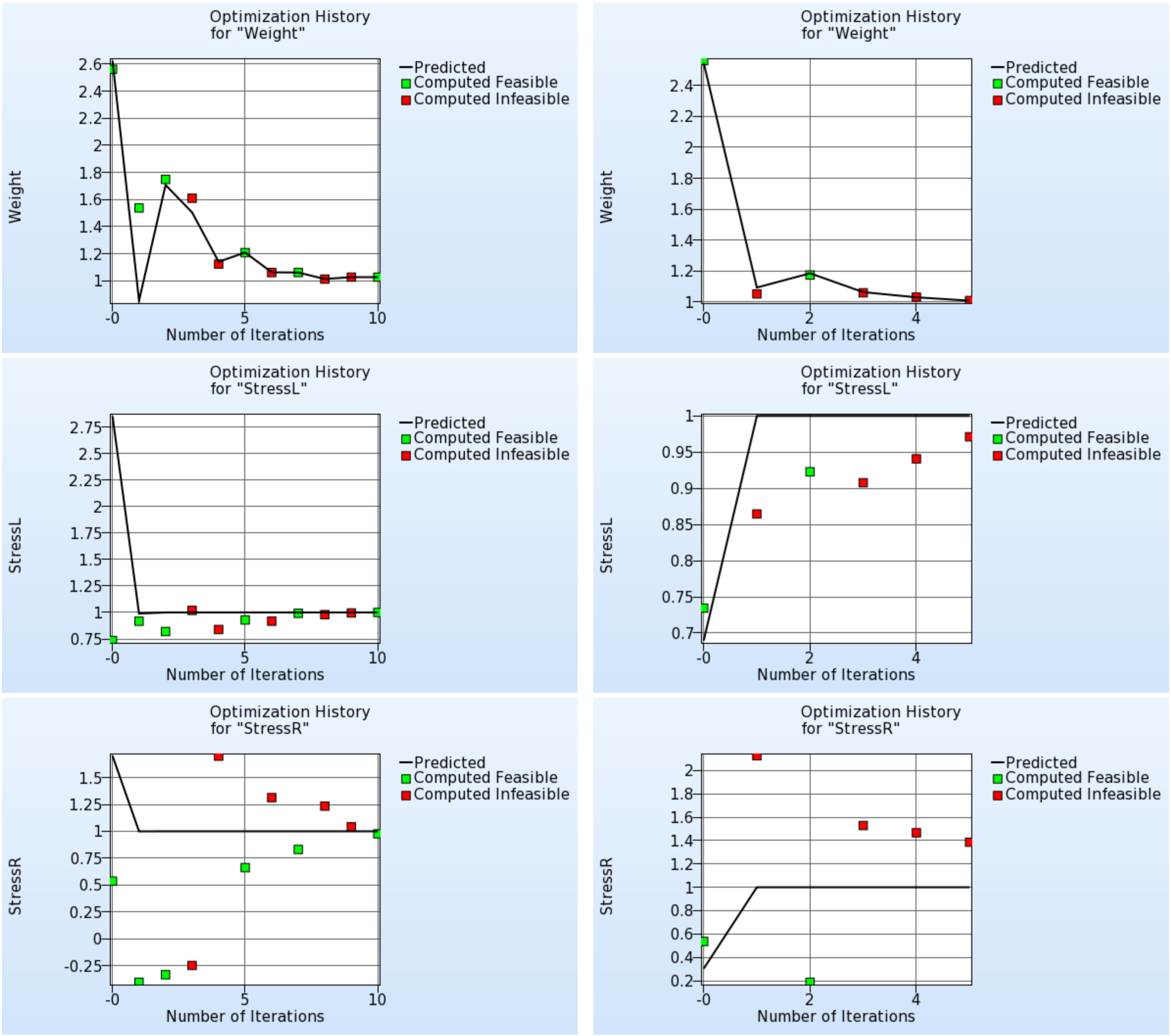


Figure 20-23: Optimization history of responses; linear (left) and quadratic (right)

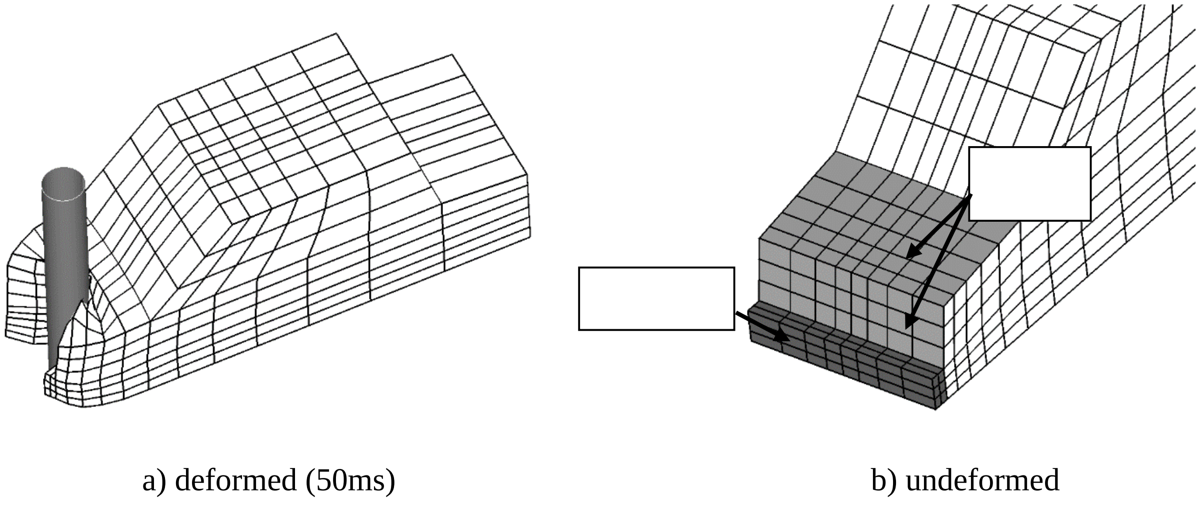
20.2. Small car crash (2 variables)

This example has the following features:

- An LS-DYNA explicit crash simulation is performed.
- Extraction is performed using standard LS-DYNA interfaces.
- A single iteration optimization using Radial Basis Function networks is performed.
- The design optimization process is automated.
- A mixed-discrete optimization is performed
- An optimization using the direct genetic algorithm is performed.

20.2.1. Introduction

This example considers the crashworthiness of a simplified small car model. A simplified vehicle moving at a constant velocity of 15.64m.s^{-1} (35mph) impacts a rigid pole. See Figure 20-24. The thickness of the front nose above the bumper is specified as part of the hood. LS-DYNA is used to perform a simulation of the crash for an event duration of 50ms.



a) deformed (50ms)

b) undeformed

Figure 20-24: Small car impacting a pole

20.2.2. Design criteria and design variables

The objective is to minimize the Head Injury Criterion (HIC) over a 15ms interval of a selected point subject to an intrusion constraint of 550mm of the pole into the vehicle at 50ms. The HIC is based on the linear head acceleration and is widely used in occupant safety regulations in the automotive industry as a brain injury criterion. In summary, the criteria of interest are the following:

- Head injury criterion (HIC) of a selected point (15ms)
- Peak acceleration of a chosen point filtered at 60Hz (SAE).
- Component Mass of the structural components (bumper, front, hood and underside)
- Intrusion computed using the relative motion of two points
- Units are in *mm* and *sec*

The design variables are the shell thickness of the car front (`thood`) and the shell thickness of the bumper (`tbumper`) (see Figure 20-24).

20.2.3. Design formulation

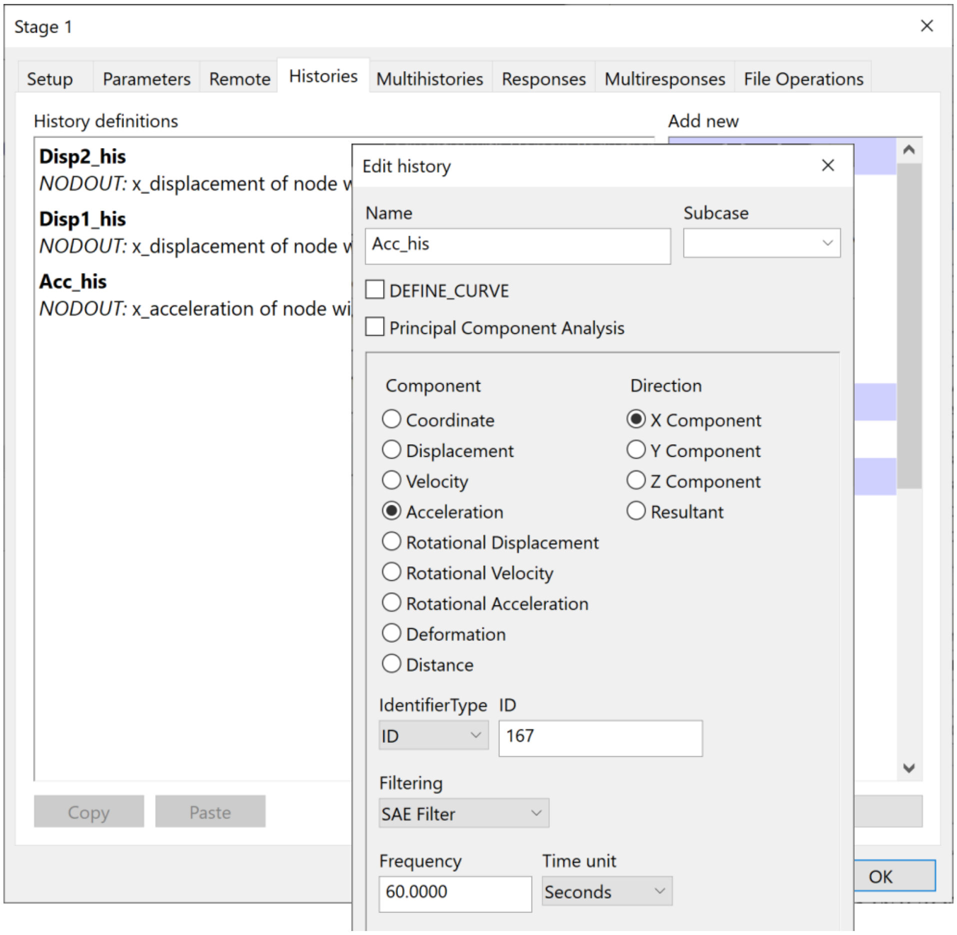
The design formulation is as follows:

Minimize

$$\text{HIC (15ms)} \qquad (20\text{-}1)$$

subject to

Intrusion (50ms) < 550mm



History definitions

Disp2_his
NODOUT: x_displacement of node w

Disp1_his
NODOUT: x_displacement of node w

Acc_his
NODOUT: x_acceleration of node wi

Edit history

Name: Acc_his

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

Component
○ Coordinate
○ Displacement
○ Velocity
● Acceleration
○ Rotational Displacement
○ Rotational Velocity
○ Rotational Acceleration
○ Deformation
○ Distance

Direction
● X Component
○ Y Component
○ Z Component
○ Resultant

IdentifierType: ID
ID: 167

Filtering: SAE Filter

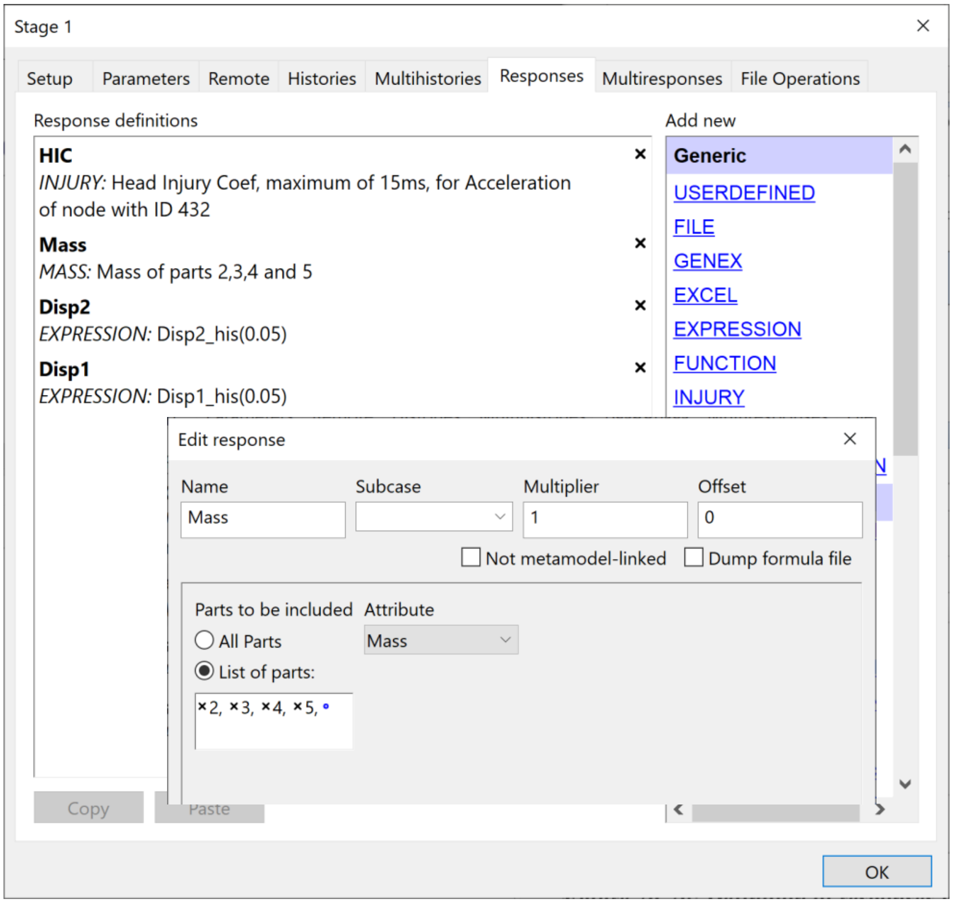
Frequency: 60.0000
Time unit: Seconds

Figure 20-25: Definition of response histories using standard LS-DYNA interfaces.

The HIC value is defined using the INJURY interface.

The intrusion is measured as the difference between the displacement of nodes **167** and **432**. The displacement curves are extracted using the LS-DYNA NODOUT interface, Figure 20-25. These curves are evaluated at time t=50ms using response expressions. The intrusion is defined using a composite expression, Figure 20-27.

The mass is computed using the LS-DYNA MASS interface, Figure 20-26, but not constrained. This is useful for monitoring the mass changes.



Response definitions

HIC
INJURY: Head Injury Coef, maximum of 15ms, for Acceleration of node with ID 432

Mass
MASS: Mass of parts 2,3,4 and 5

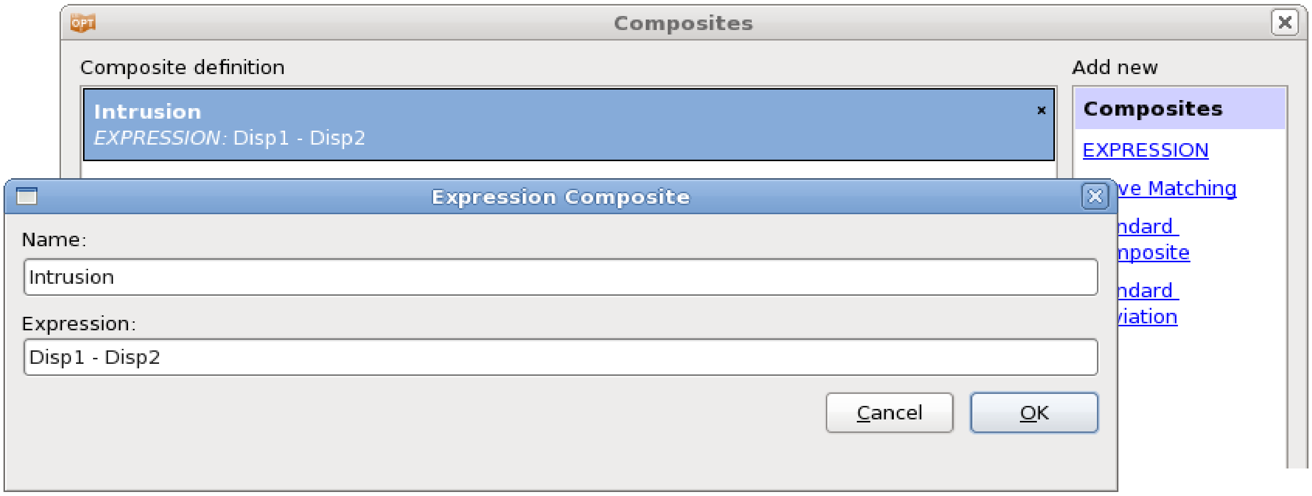
Disp2
EXPRESSION: Disp2_his(0.05)

Disp1
EXPRESSION: Disp1_his(0.05)

Name: Mass
Subcase:
Multiplier: 1
Offset: 0
□ Not metamodel-linked □ Dump formula file

Parts to be included
○ All Parts
● List of parts:
×2, ×3, ×4, ×5,
Attribute: Mass

Figure 20-26: *Definition of responses using standard LS-DYNA interfaces and expressions.*



Composite definition

Intrusion
EXPRESSION: Disp1 - Disp2

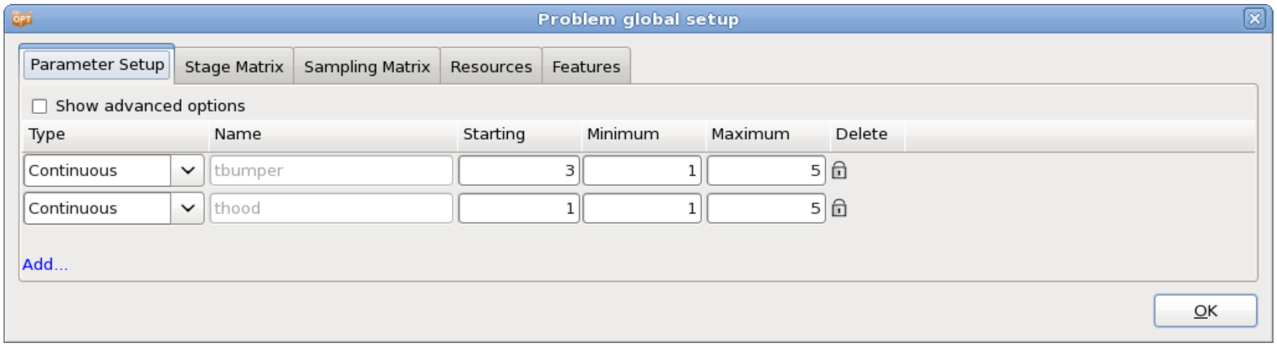
Name: Intrusion
Expression: Disp1 - Disp2

Figure 20-27: *Definition of composite expression using previously defined responses.*

20.2.4. Modeling

The simulation is performed using LS-DYNA. An extract from the parameterized input deck is shown below. The parameterization of the model is done using the `*PARAMETER` keyword. The cylinder for impact is modeled as a rigid wall.

```
*KEYWORD
*PARAMETER
rtbumper,3.0,rthood,1.0
```



☐ Show advanced options

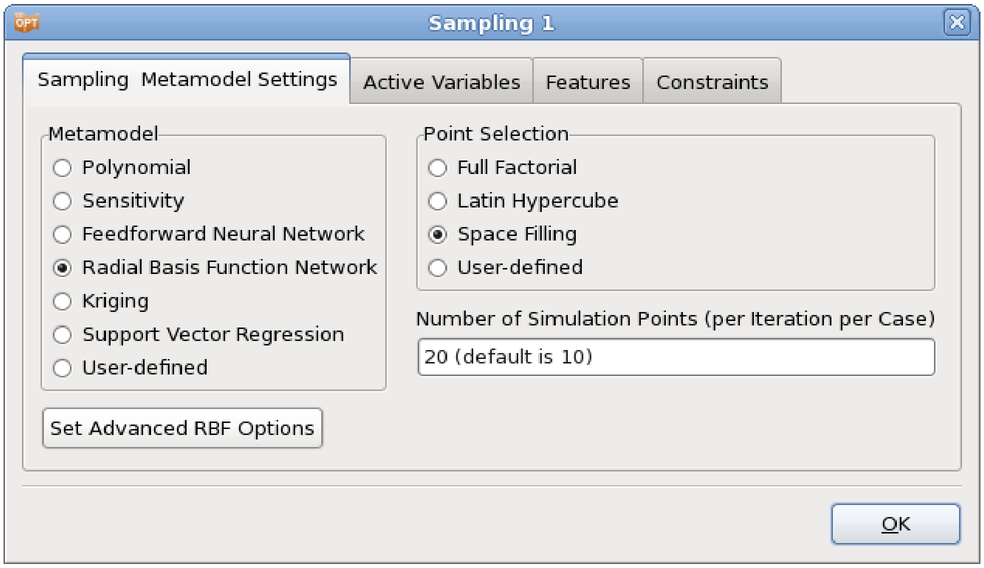
Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	🔒
Continuous	thood	1	1	5	🔒

Figure 20-28: Parameter Setup;

A design space of [1; 5] is used for both design variables, Figure 20-28.

20.2.5. Single iteration run using Radial Basis Functions

As a first step, a single iteration is run using Radial Basis Function networks (RBF). In this manner a non-linear approximation is created across the whole design space. The approximation can then be used for sensitivity analysis or optimization.



Metamodel
- ○ Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ● Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Point Selection
- ○ Full Factorial
- ○ Latin Hypercube
- ● Space Filling
- ○ User-defined

Number of Simulation Points (per Iteration per Case)
20 (default is 10)

Figure 20-29: Sampling dialog; Select metamodel RBF, increase the number of points to 20.

Results

The computed vs. predicted HIC and `Disp2` responses are given in Figure 20-30. The corresponding R^2 value for HIC is 0.998, while the RMS error is 4.61%. For Disp2, the R^2 value is 0.994, while the RMS error is 0.353%.

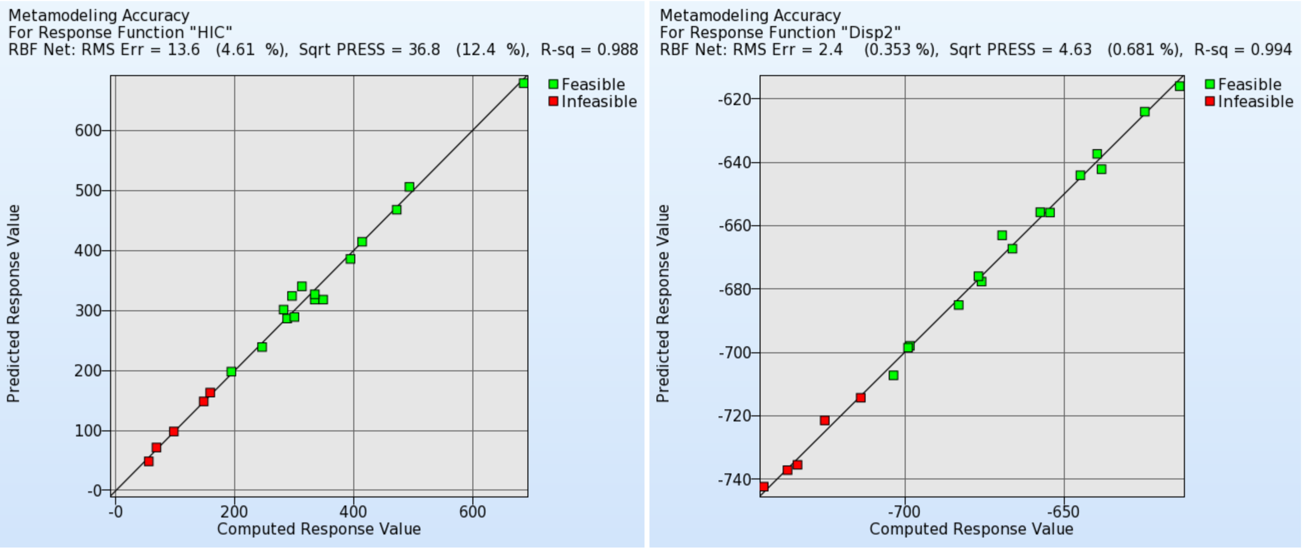


Figure 20-30: Computed vs. predicted responses – RBF approximation

Table 20-5: Comparison of baseline run and optimum (single iteration, RFB metamodel)

	Baseline (Computed)	**Baseline (Predicted)**	**1. Opt (Computed)**	**1. Opt (Predicted)**
t_hood	1	-	1.60	-
t_bumper	3	-	5	-
HIC	68.03	71.51	130.2	134.08
Intrusion	575.68	573.90	548.67	550
Mass	0.41	0.41	0.67	0.67

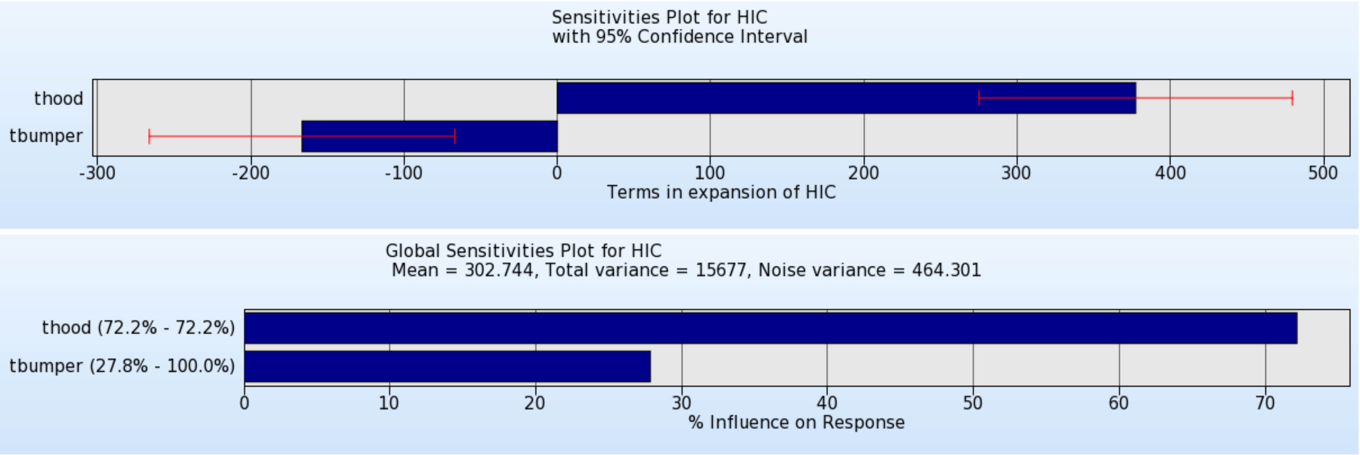


Figure 20-31: Sensitivities plots; ANOVA with 95% confidence interval (top) and GSA (bottom)

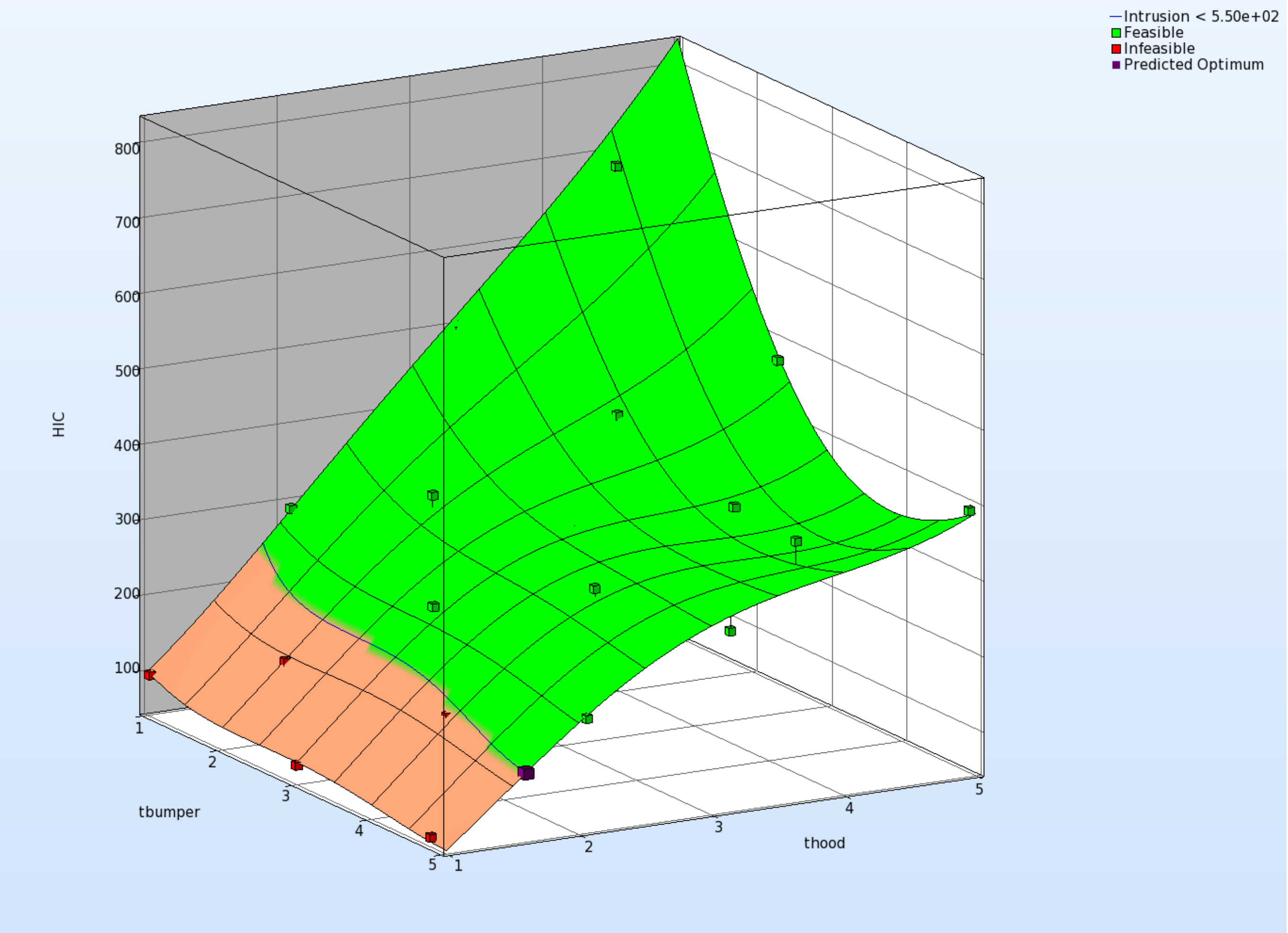


Figure 20-32: Surface plot for objective function HIC with predicted and computed optimum, simulation points and residuals; constraints are displayed on the surface.

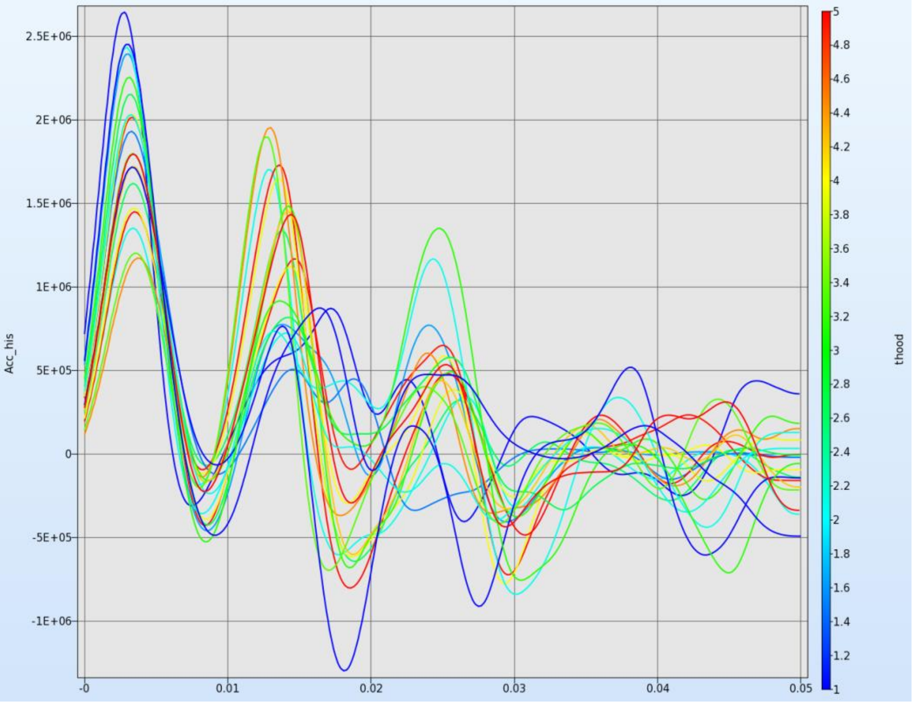
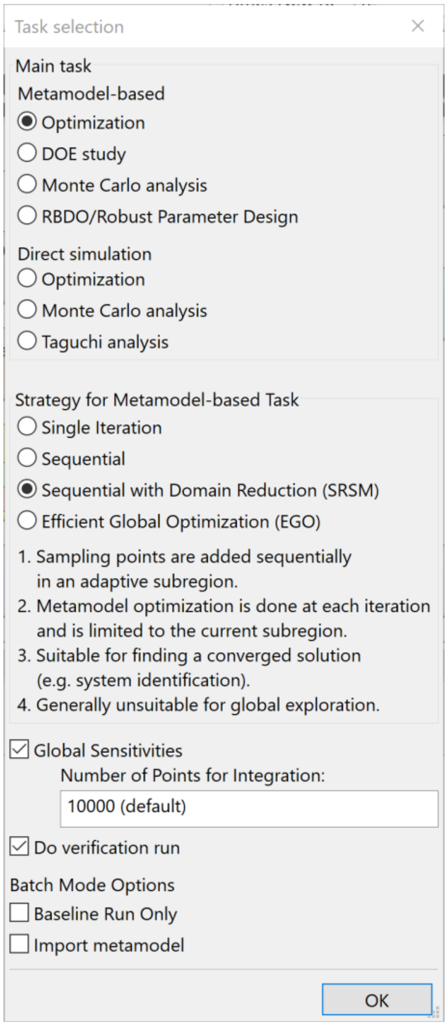


Figure 20-33: History plot for Acceleration; the curves are color-coded using the value of the variable thood.

20.2.6. Automated run using linear metamodels

An automated optimization is performed with a linear approximation. Select the strategy *Sequential with domain reduction*, Figure 20-34, and switch to the metamodel type *Polynomial linear*, Figure 20-35. Use the default number of points per iteration per case.

In the **Termination Criteria** dialog, set the maximum number of iterations to 8, Figure 20-36.



Task selection

Main task

Metamodel-based

● Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

○ Taguchi analysis

Strategy for Metamodel-based Task

○ Single Iteration

○ Sequential

● Sequential with Domain Reduction (SRSM)

○ Efficient Global Optimization (EGO)

1. Sampling points are added sequentially in an adaptive subregion.
2. Metamodel optimization is done at each iteration and is limited to the current subregion.
3. Suitable for finding a converged solution (e.g. system identification).
4. Generally unsuitable for global exploration.

☑ Global Sensitivities

Number of Points for Integration: 10000 (default)

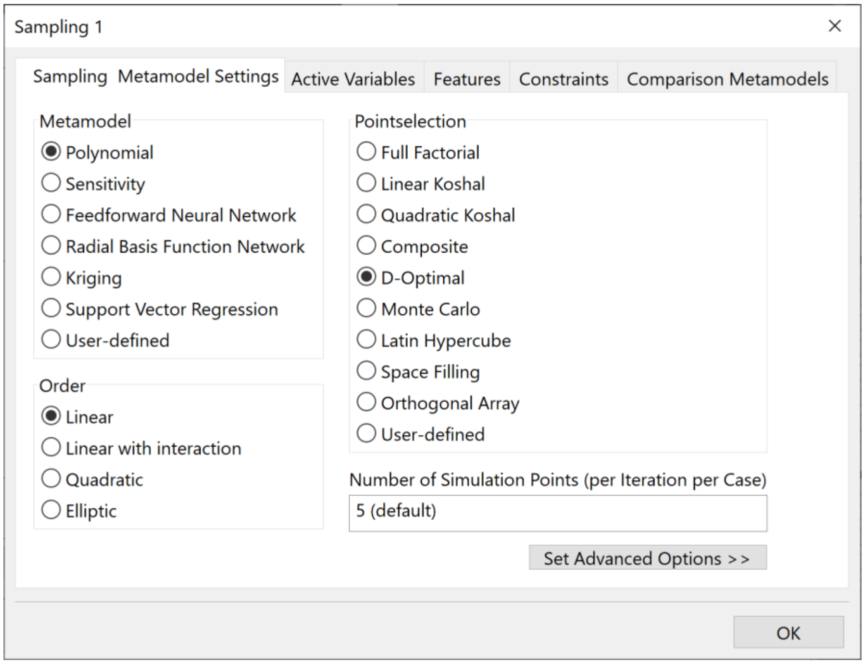
☑ Do verification run

Batch Mode Options

□ Baseline Run Only

□ Import metamodel

Figure 20-34: Task dialog; select Strategy Sequential with Domain Reduction



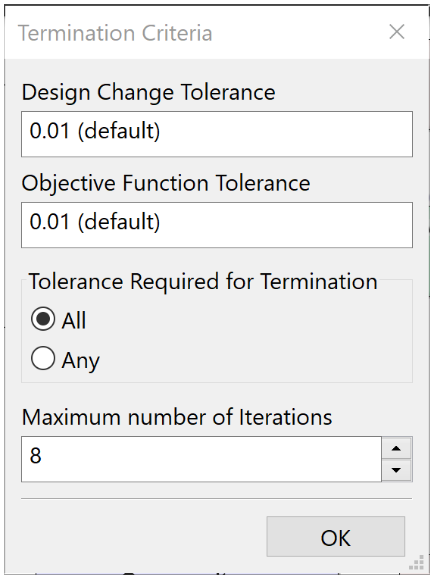
Metamodel
- ● Polynomial
- ○ Sensitivity
- ○ Feedforward Neural Network
- ○ Radial Basis Function Network
- ○ Kriging
- ○ Support Vector Regression
- ○ User-defined

Order
- ● Linear
- ○ Linear with interaction
- ○ Quadratic
- ○ Elliptic

Pointselection
- ○ Full Factorial
- ○ Linear Koshal
- ○ Quadratic Koshal
- ○ Composite
- ● D-Optimal
- ○ Monte Carlo
- ○ Latin Hypercube
- ○ Space Filling
- ○ Orthogonal Array
- ○ User-defined

Number of Simulation Points (per Iteration per Case): 5 (default)

Figure 20-35: Sampling Dialog; use the default settings for SRSM for metamodel type and order, point selection scheme and number of points



Design Change Tolerance: 0.01 (default)

Objective Function Tolerance: 0.01 (default)

Tolerance Required for Termination
- ● All
- ○ Any

Maximum number of Iterations: 8

Figure 20-36: Termination Criteria dialog; select the maximum number of iterations

Results

It can be seen in Figure 20-37 that the objective function (HIC) and intrusion constraint are approximately optimized at the 7th iteration. It takes about 8 iterations for the approximated (solid line) and computed (square symbols) HIC to correspond. The approximation improves through the contraction of the subregion. As the variable `thood` never moves to the edge of the subregion during the optimization process, the heuristic in

LS-OPT enforces pure zooming (see Figure 20-38). For `tbumper`, panning occurs as well due to the fact that the linear approximation predicts a variable on the edge of the subregion.

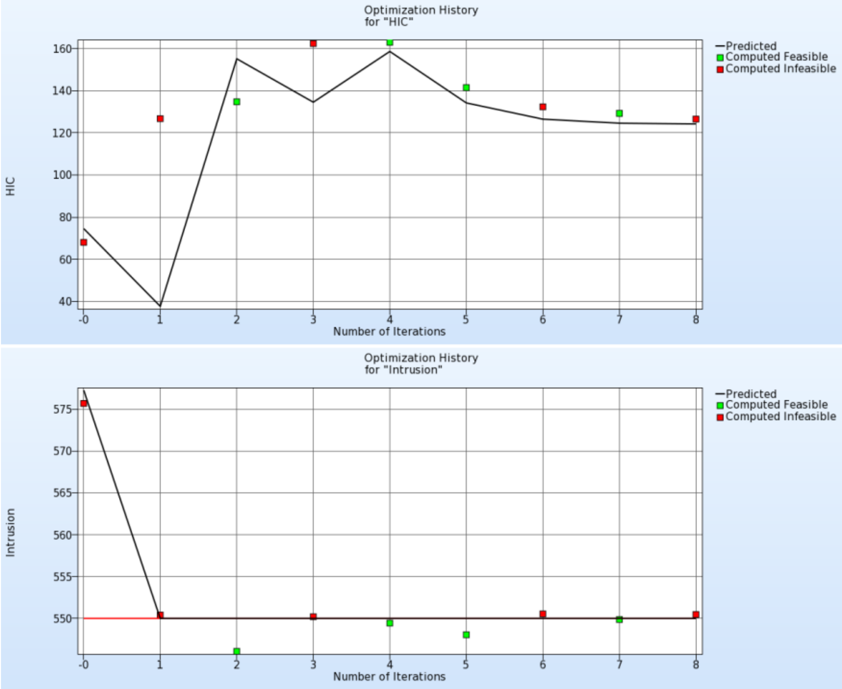


Figure 20-37: Optimization history of HIC and Intrusion

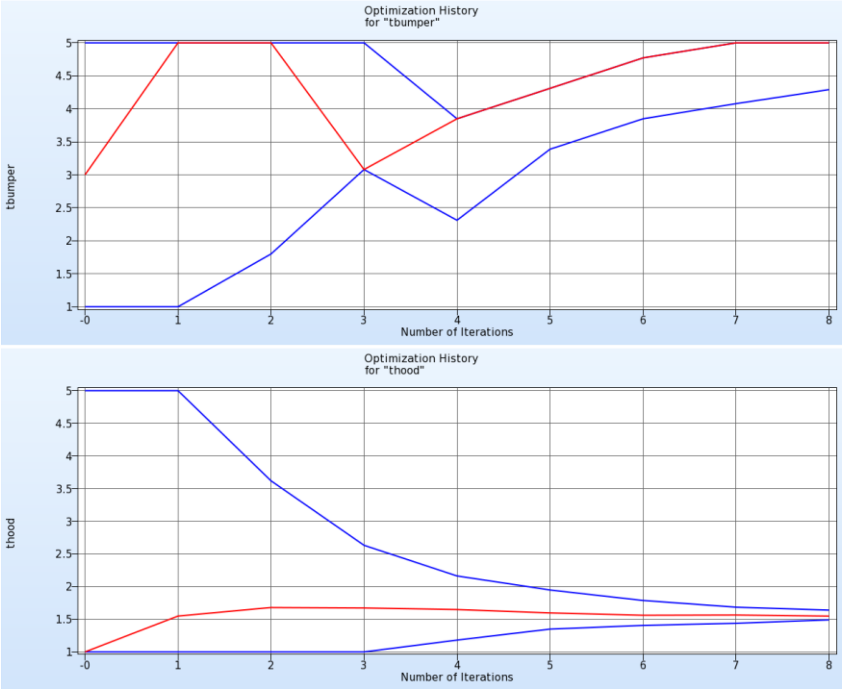
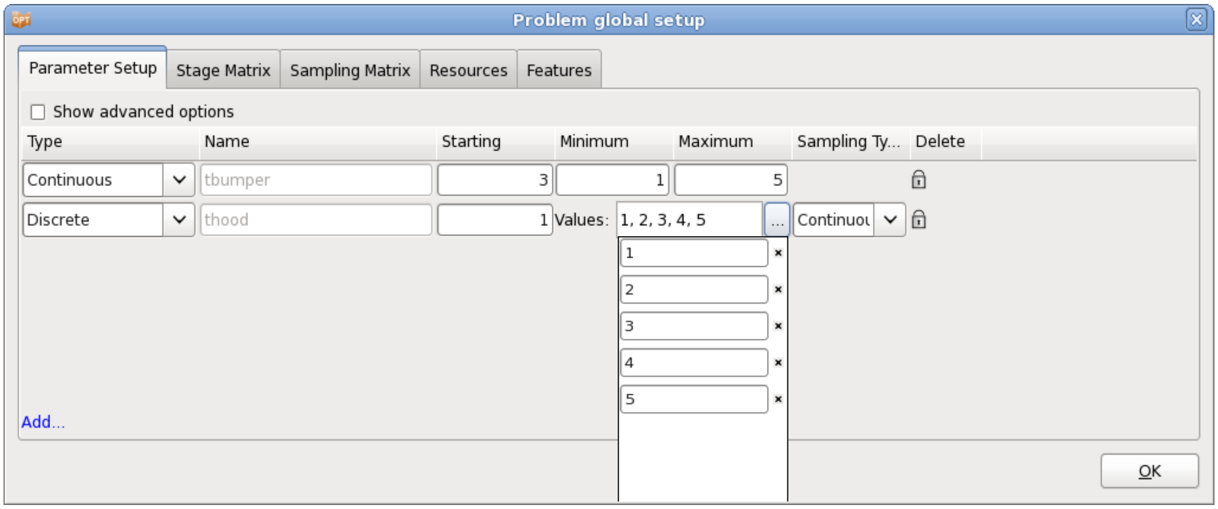


Figure 20-38: Optimization history of design variables

20.2.7. Mixed-discrete optimization

Mixed discrete optimization is achieved simply by setting the `thood` variable to be discrete with possible values of 1.0, 2.0, 3.0, 4.0, and 5.0. The definition of a discrete variable is displayed in Figure 20-39.



□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Sampling Ty...	Delete
Continuous	tbumper	3	1	5		🔒
Discrete	thood	1	Values: 1, 2, 3, 4, 5		Continuou	🔒

1 ×
2 ×
3 ×
4 ×
5 ×

Figure 20-39: Parameter Setup dialog; Definition of a discrete variable.

Results

The design variables histories are shown in Figure 20-41, the optimization histories for the objective `HIC` and the constraint `Intrusion` in Figure 20-40.

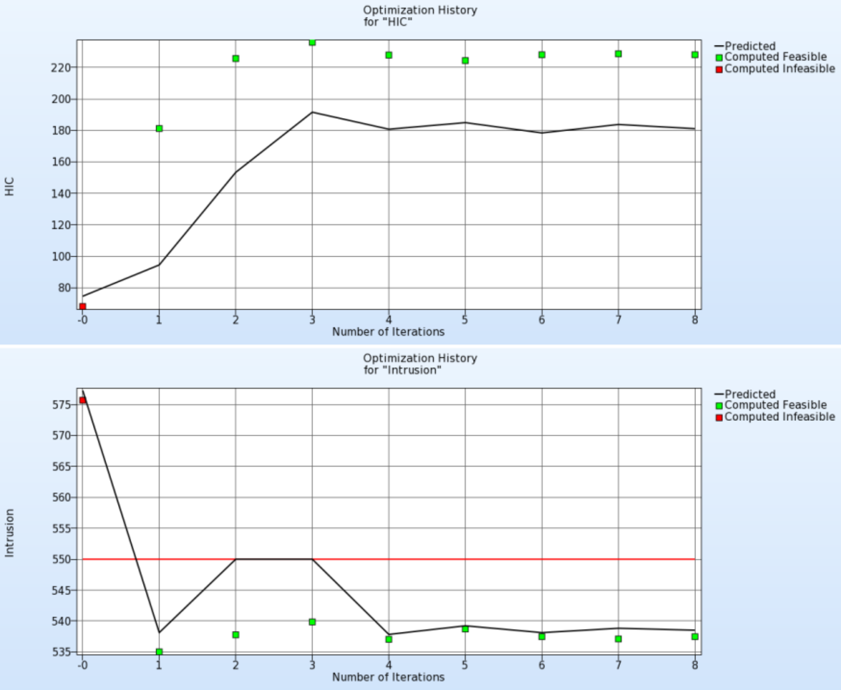


Figure 20-40: Optimization history of HIC and Intrusion for mixed-discrete optimization

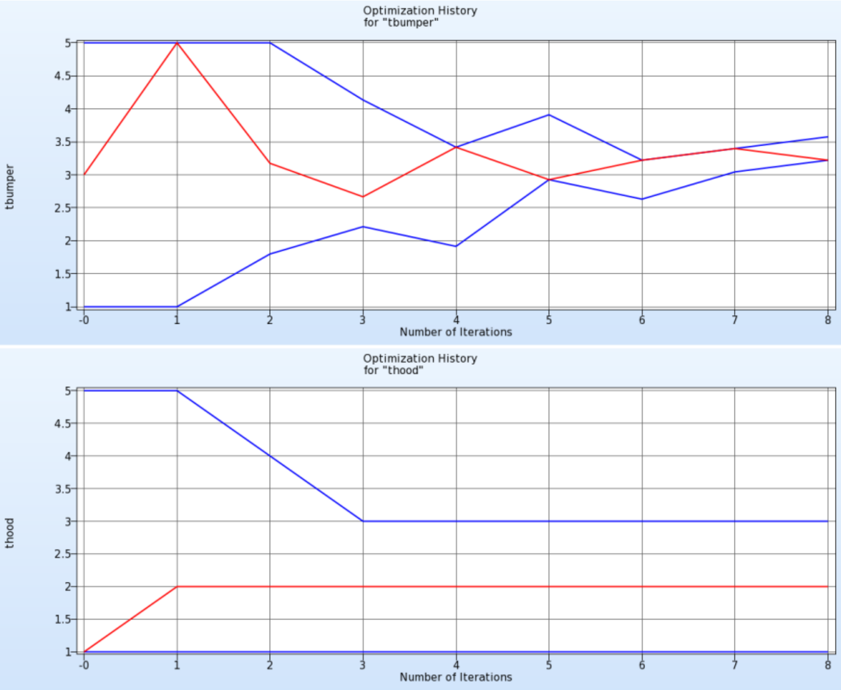
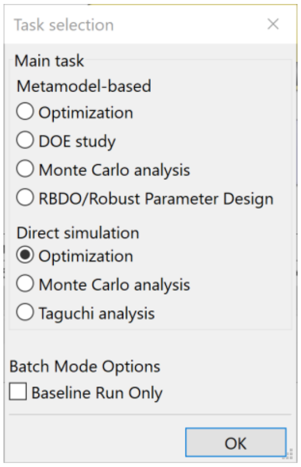


Figure 20-41: Mixed-discrete variable histories.

20.2.8. Optimization using Direct GA simulation

The same problem is solved using a direct GA simulation, Figure 20-42. GA specific settings and advanced options may be selected in the Optimization dialog, Figure 20-43. For illustration, the population size is taken as 10 and number of generations is limited to 15. The Stochastic Universal Sampling method is used as selection operator. Two elite members (*Number of Elites*) are used in each generation. For real crossover, SBX operator is used *(Crossover Type*) with a distribution index of 5 (*Crossover Distribution*) and crossover probability of 0.99 (*Crossover Probability*). The real mutation probability (*Mutation Probability*) is 1.0.


Task selection

Main task

Metamodel-based

○ Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

● Optimization

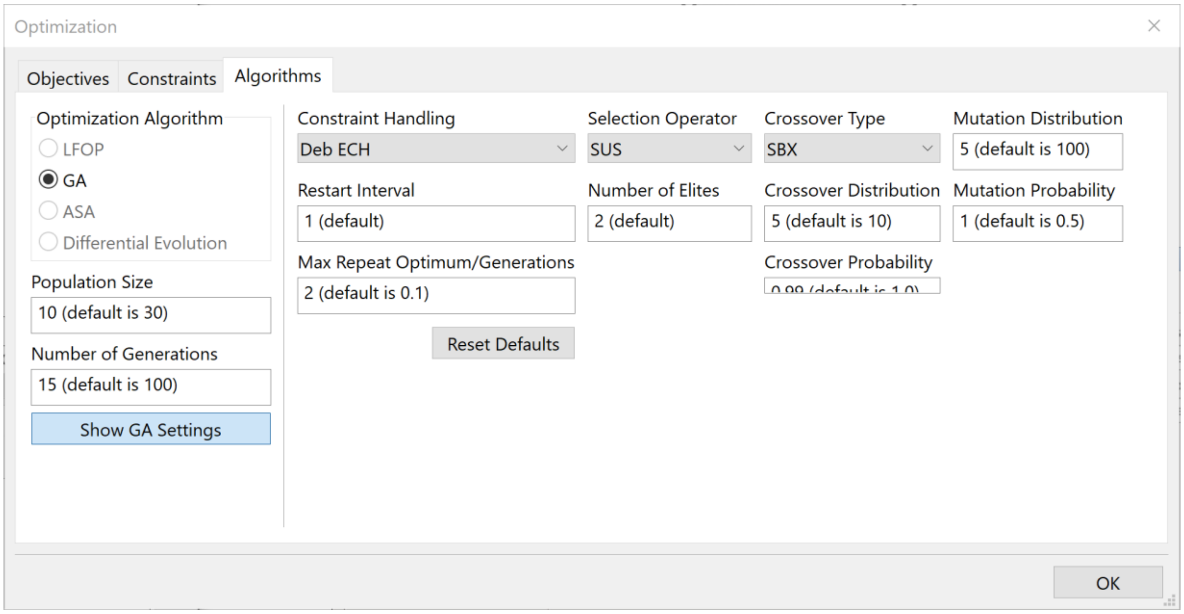
○ Monte Carlo analysis

○ Taguchi analysis

Batch Mode Options

□ Baseline Run Only

Figure 20-42: Task dialog; Direct Genetic algorithm


Optimization

Optimization Algorithm

○ LFOP

● GA

○ ASA

○ Differential Evolution

Population Size: 10 (default is 30)

Number of Generations: 15 (default is 100)

Constraint Handling: Deb ECH

Restart Interval: 1 (default)

Max Repeat Optimum/Generations: 2 (default is 0.1)

Selection Operator: SUS

Number of Elites: 2 (default)

Crossover Type: SBX

Crossover Distribution: 5 (default is 10)

Crossover Probability: 0.99 (default is 1.0)

Mutation Distribution: 5 (default is 100)

Mutation Probability: 1 (default is 0.5)

Figure 20-43: Optimization dialog; Specification of advanced GA options

Results

The outcome of the optimization is shown in Figure 20-44 and Figure 20-45. The discrete variable was fixed at 2 units. The direct GA does not terminate if the optimal result does not change from one iteration to the next, since the values may still improve. Note that the optimization history treats ‘generation’ as ‘iteration’ to display results.

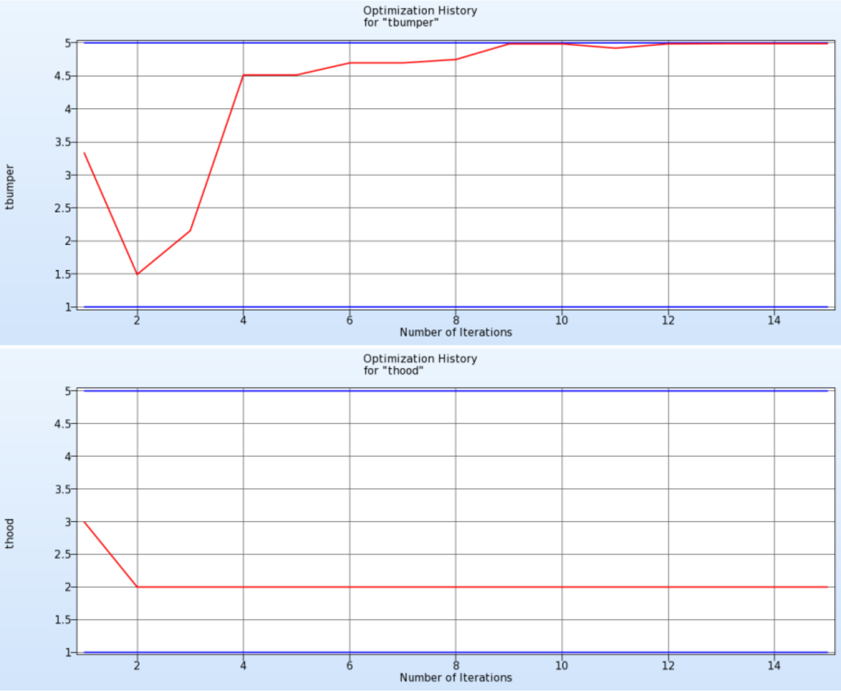


Figure 20-44: Optimization history of mixed-discrete variable optimization using direct GA.

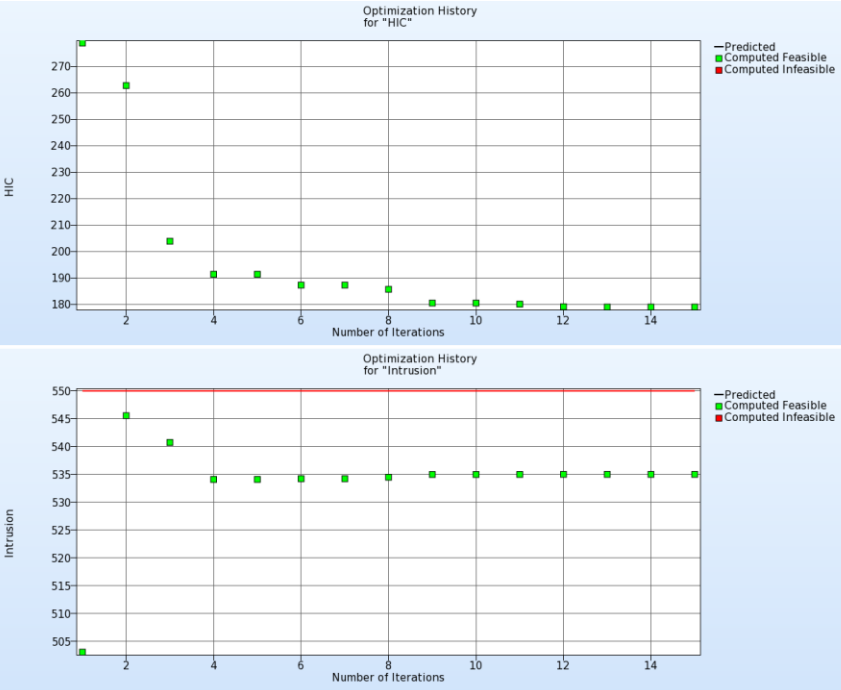


Figure 20-45: Optimization history of HIC and Intrusion

20.2.9. Multilevel Optimization using both Direct method and Metamodel

This example uses the same finite element model, but the optimization problem is modified to include two more variables. These variables are the material Young's modulus *YM* and the yield stress *SIGY*. The optimization problem is given in Equation (20-2). However, the optimization is solved in two levels – the outer level optimizes `SIGY` and `YM` using a single iteration metamodel-based method (Equation (20-3)) and the inner level optimizes the thickness values `thood and tbumper` using direct GA (Equation (20-4)).

$$\min_{thood,tbumper,SIGY,YM} HIC\ (15ms) \qquad (20\text{-}2)$$

$$s.t. Intrusion\ (50ms) < 550mm$$

The outer level optimization problem is:

$$\min_{SIGY,YM} HIC_{opt\_thood\_tbumper}\ (15ms) \qquad (20\text{-}3)$$

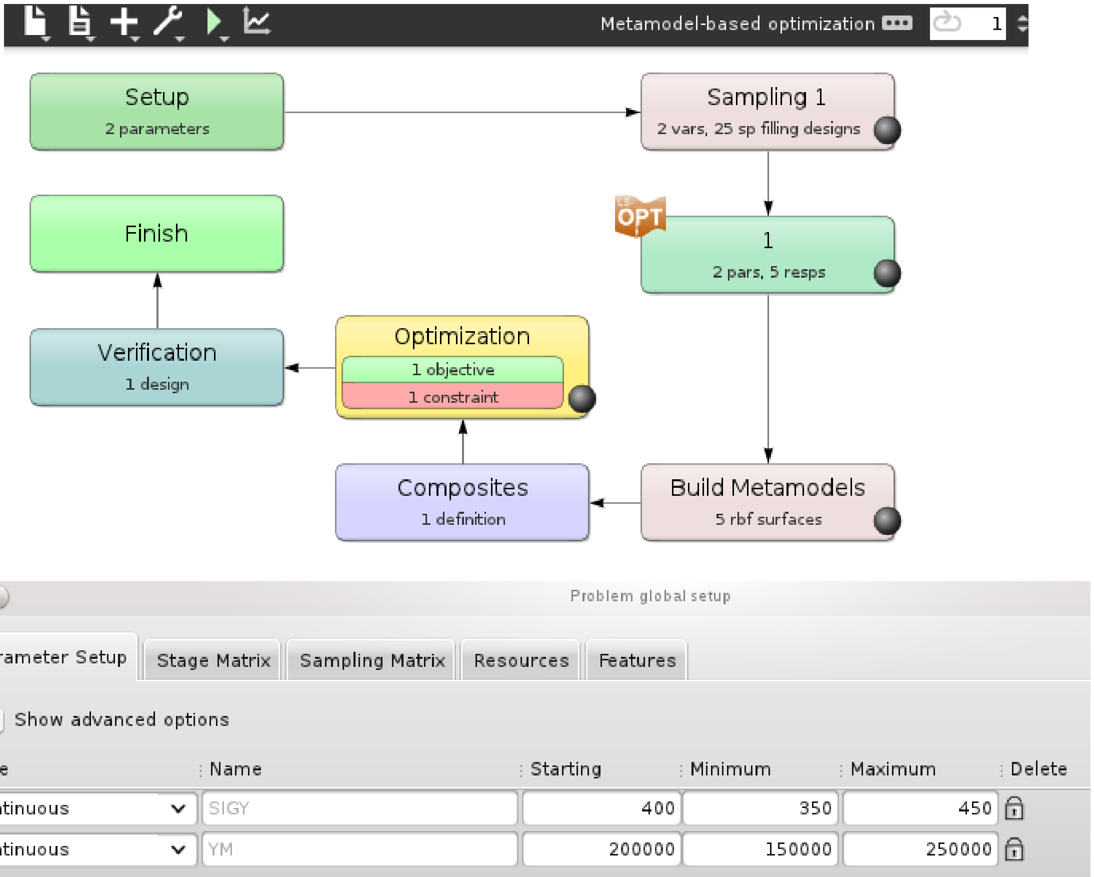
$$s.t. Intrusion_{opt\_thood\_tbumper}\ (50ms) < 550mm$$

where $\mathrm{HIC}_{\mathrm{opt\_thood\_tbumper}}$ and $\mathrm{Intrusion}_{\mathrm{opt\_thood\_tbumper}}$ are the HIC and intrusion values obtained as the results of the inner level optimization problem with respect to variables `thood and tbumper` given by Equation (20-4). $\mathrm{HIC}_{\mathrm{opt\_thood\_tbumper}}$ and $\mathrm{Intrusion}_{\mathrm{opt\_thood\_tbumper}}$ are obtained for every outer level sample (`SIGY-YM` pair) by running an inner level optimization for each sample. The inner level optimization problem for the *j*th outer level sample is:

$$\min_{thood,tbumper} HIC(\mathrm{thood}, \mathrm{tbumper}|\mathrm{YMj}, \mathrm{SIGYj})\ (15ms) \qquad (20\text{-}4)$$

$$s.t.\ Intrusion(\mathrm{thood}, \mathrm{tbumper}|\mathrm{YMj}, \mathrm{SIGYj})(50ms) < 550mm$$

The LS-OPT GUI for outer level problem setup is shown in Figure 20-46. The optimization problem setup is shown in Figure 20-47; HIC_1 and Intru are optimized responses calculated in the inner level.

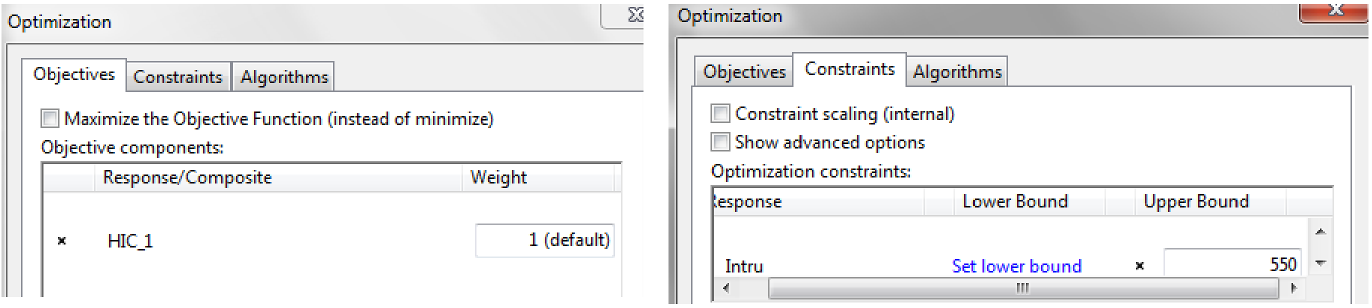


Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	SIGY	400	350	450	
Continuous	YM	200000	150000	250000	

Figure 20-46: Multilevel Optimization outer level setup


Optimization

Objectives | Constraints | Algorithms

□ Maximize the Objective Function (instead of minimize)

Objective components:

	Response/Composite	Weight
×	HIC_1	1 (default)

Optimization

Objectives | Constraints | Algorithms

□ Constraint scaling (internal)

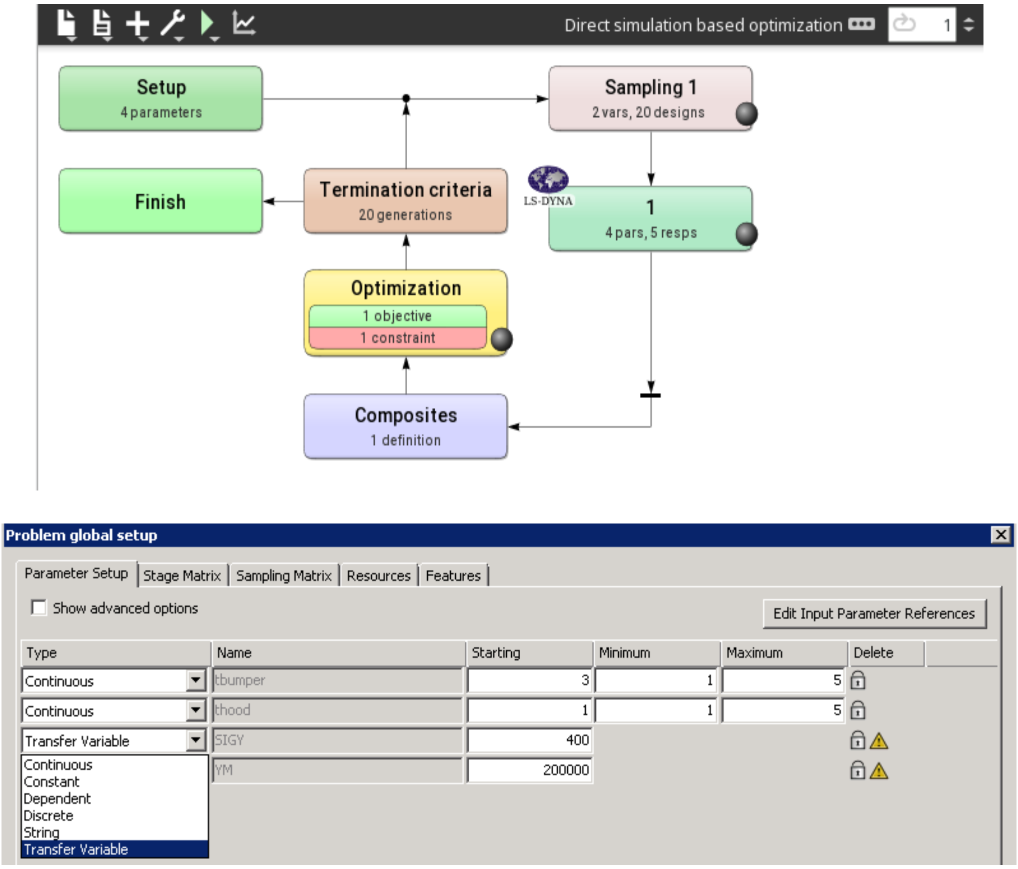
□ Show advanced options

Optimization constraints:

esponse	Lower Bound		Upper Bound
Intru	Set lower bound	×	550

Figure 20-47: Multilevel Optimization outer level optimization problem

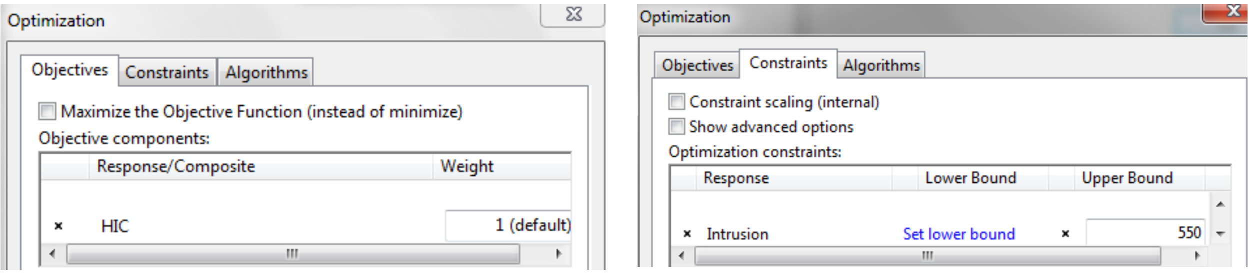
The LS-OPT GUI for inner level problem setup is shown in Figure 20-48. The optimization problem setup is shown in Figure 20-49. It should be noted that the outer level variables are *Transfer Variables* in the inner level and are treated as constants for the optimization.



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	
Transfer Variable	SIGY	400			
	YM	200000			

Figure 20-48: Multilevel Optimization inner level setup



☐ Maximize the Objective Function (instead of minimize)

Objective components:

Response/Composite	Weight
× HIC	1 (default)

☐ Constraint scaling (internal)

☐ Show advanced options

Optimization constraints:

Response	Lower Bound	Upper Bound
× Intrusion	Set lower bound	× 550

Figure 20-49: Multilevel Optimization inner level optimization problem

Results

The optimum solution is obtained at `SIGY` = 353.2, `YM` = 2.4E5, `tbumper` = 4.91, `thood` = 1.72. The corresponding HIC value is 80.47 and there is no constraint violation at the solution. It should be noted that this solution has a lower HIC value than in Section 20.2.5. This is because additional variables were introduced, leading to increased design options.

The metamodel for HIC, with respect to outer level variables `YM` and `SIGY,` is shown in Figure 20-50. The optimum is also plotted on the figure (purple cube). The inner level optimization history is depicted in Figure 20-51 for the outer level sample 2.1 (i.e. the sample with optimized `YM` and `SIGY`).

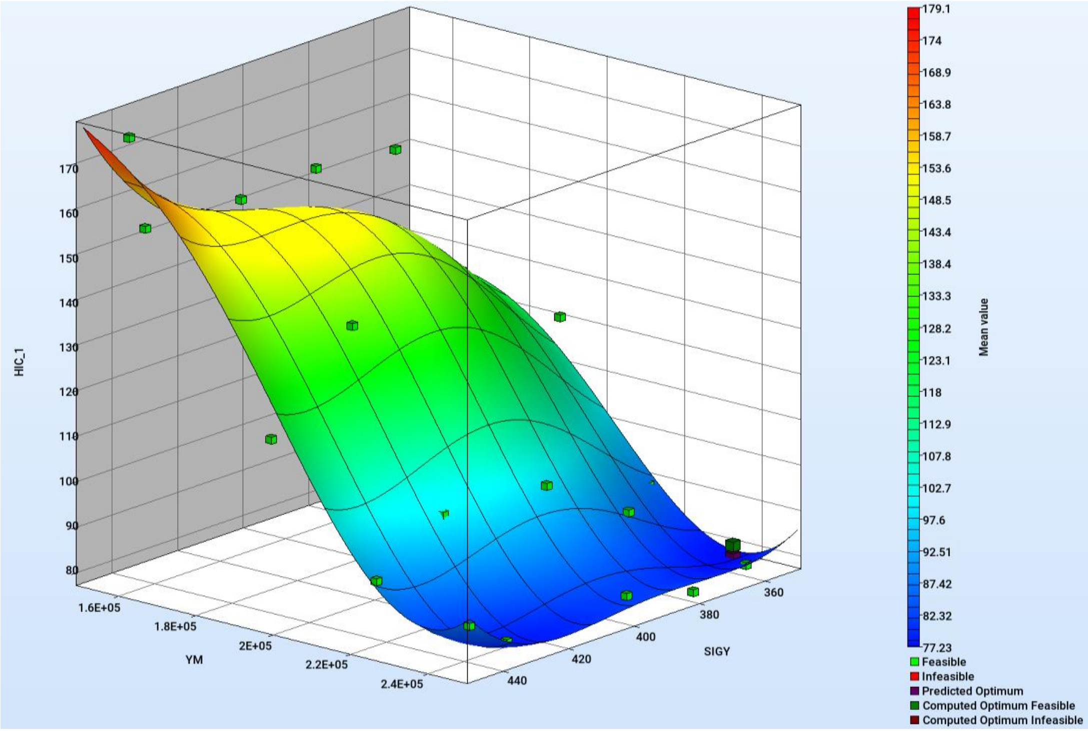


Figure 20-50: Multilevel Optimization. Metamodel for objective function (HIC)

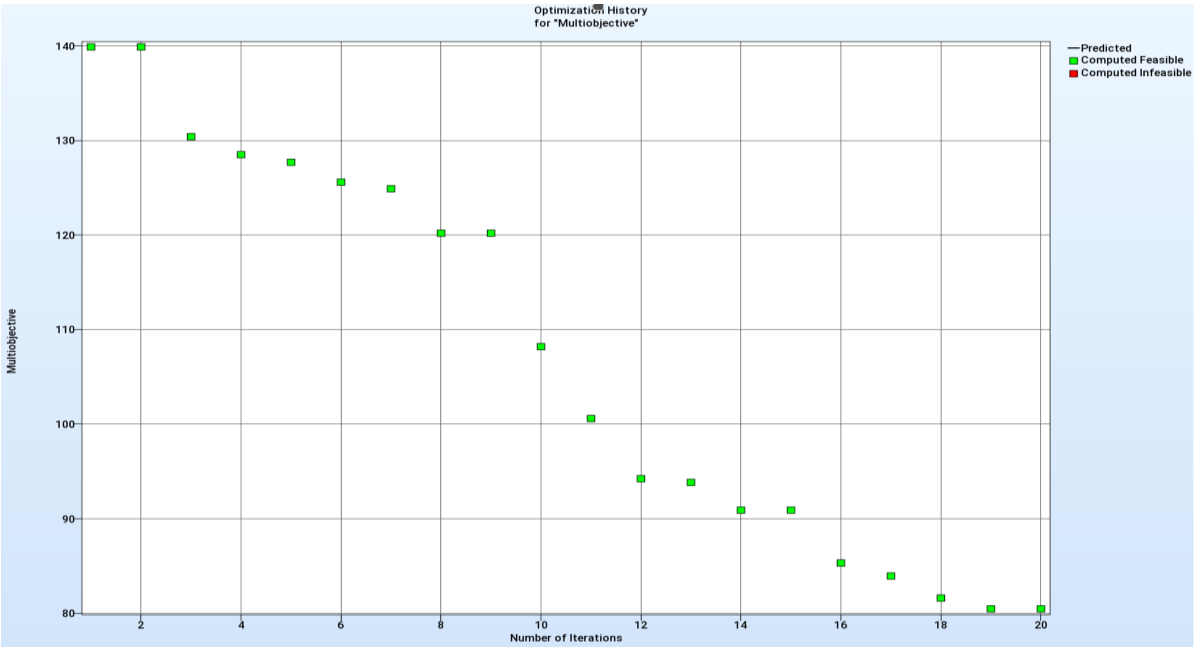


Figure 20-51: Inner level optimization history for the last (optimal) outer level sample.

20.2.10. Multilevel Optimization using continuous and string variables

Multilevel optimization can be used to optimize different sets of variables using different methods. For example, direct optimization is often preferred for string or categorical variables while metamodel-based methods are often used for other variables. In this example, two of the variables are continuous while two other variables are strings. The continuous variables represent component thicknesses `thood` and `tbumper` and the string variables `mat_b` and `mat_hood` are the names of include files with different material properties. Two string constants `m1` and `material3` are also used in the example. Different methods of parameterizing string variables and constants (native LS-DYNA parameterization and user-defined) are demonstrated through this example.

The optimization problem is given in Equation (20-5). However, the optimization is solved in two levels – the outer level optimizes `thood and tbumper` using a domain reduction metamodel-based method (Equation (20-6)) and the inner level optimizes the thickness values `mat_hood and mat_b` using direct GA (Equation (20-7)).

$$\underset{\text{thood,tbumper,mat\_b,mat\_hood}}{\text{Minimize}} \quad \text{Mass} \tag{20-5}$$

subject to

$$\text{Intrusion (50ms)} < 550\text{mm}$$

The outer level optimization problem is:

$$\underset{\text{tbumper,thood}}{\text{Minimize}}\, \text{Mass}_{\text{opt\_mat\_b\_mat\_hood}} \tag{20-6}$$

subject to

$$\text{Intrusion}_{\text{opt\_mat\_b\_mat\_hood}}\ \text{(50ms)} < 550\text{mm}$$

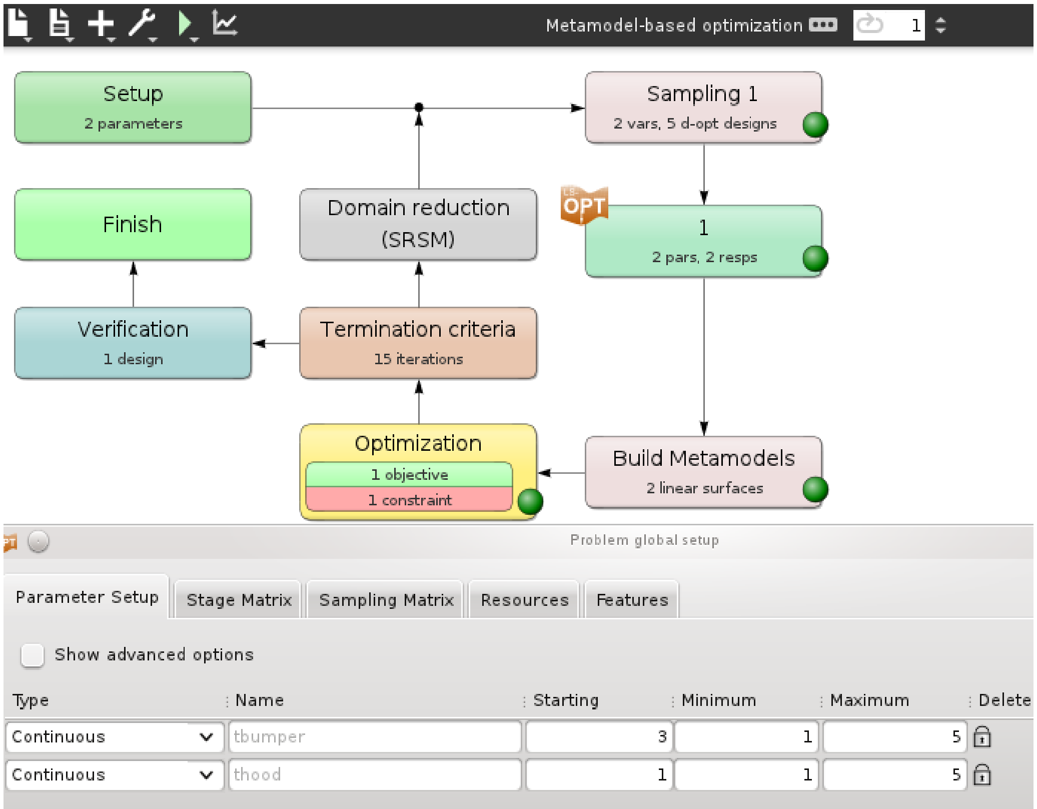
where $\text{Mass}_{\text{opt\_mat\_b\_mat\_hood}}$ and $\text{Intrusion}_{\text{opt\_mat\_b\_mat\_hood}}$ are the mass and intrusion values obtained as the results of the inner level optimization problem (Equation (20-7)) with respect to variables `mat_hood and mat_b.` $\text{Mass}_{\text{opt\_mat\_b\_mat\_hood}}$ and $\text{Intrusion}_{\text{opt\_mat\_b\_mat\_hood}}$ are obtained for every outer level sample (`tbumper-thood` pair) by running an inner level optimization for each sample. The inner level optimization problem for the *jth* outer level sample is:

$$\underset{\text{mat\_b,mat\_hood}}{\text{Minimize}}\, \text{Mass}\ (\text{mat\_b}, \text{nat\_hooh} \mid \text{tbumper}_j, \text{thood}_j) \tag{20-7}$$

subject to

$$\text{Intrusion (mat\_b,nat\_hooh}|\text{tbumper}_j\text{,thood}_j\text{) (50ms)} < 550\text{mm}$$

The outer level LS-OPT setup consists of an LS-OPT stage parameterized using two transfer variables. These variables, `tbumper` and `thood`, are continuous variables in the outer level (Figure 20-52).



☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	tbumper	3	1	5	
Continuous	thood	1	1	5	

Figure 20-52: Outer level optimization setup

The inner level consists of two string variables and two string constants, in addition to the two transfer variables whose values are passed down from the outer level. The LS-DYNA input deck is parameterized as follows. tbumper, thood, m1 and mat_b are parameterized using the *PARAMETER card. The string parameters are indicated using “c” before the variable names.

*PARAMETER

rtbumper,3.0,rthood,1.0,cm1,mat1,cmat_b,mat_b_o

Two other string parameters are defined using the user-defined format. The parameter thood appears at two places in the LS-DYNA deck:

*include

<<mat_hood:0>>

*include

<<mat_hood:30>>

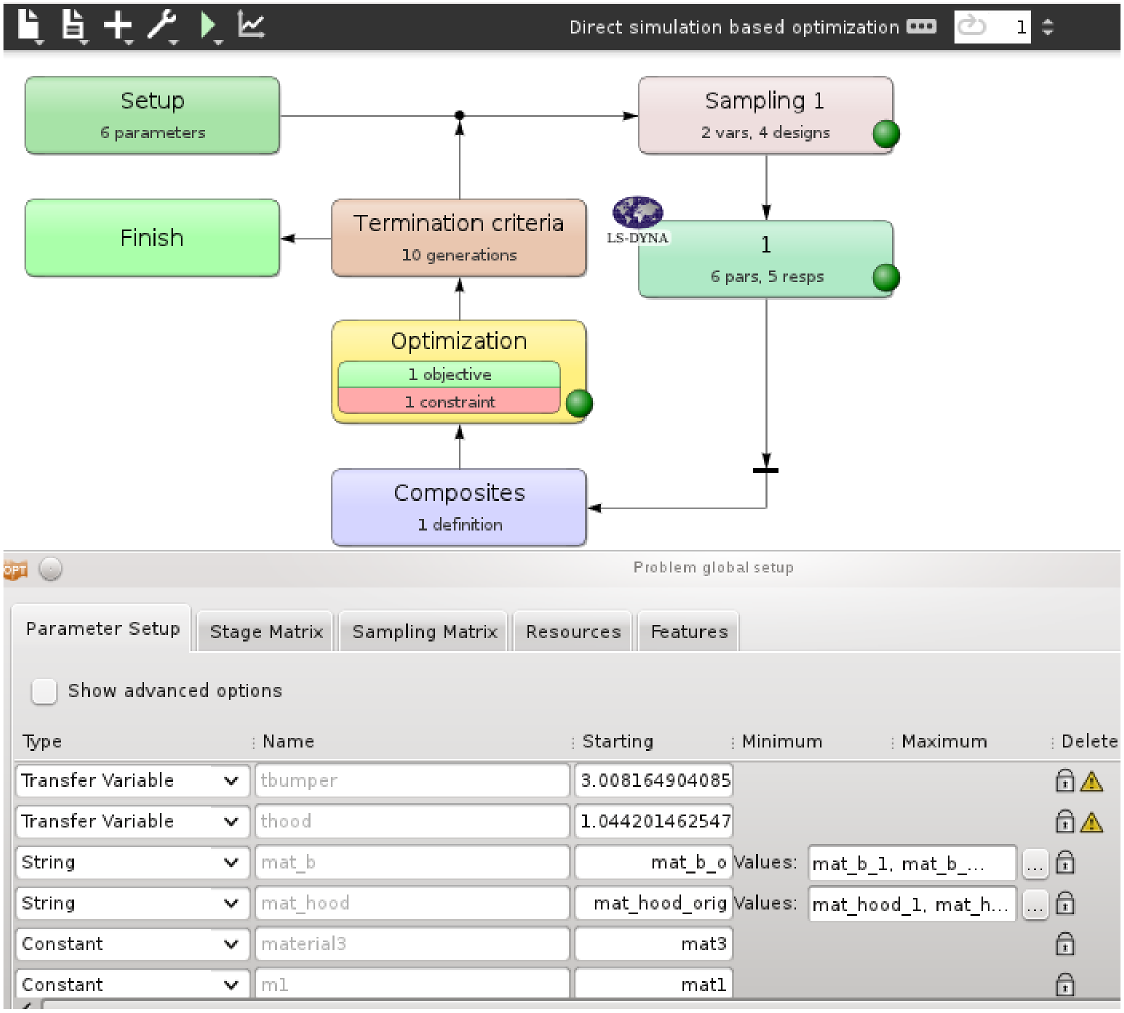
<<:0>> indicates that the entire replacement string will be printed without any additional spaces. <<:30>> indicates that if the length of the replacement string for mat_hood is longer than 30 then it will be truncated. Also, if the replacement string for mat_hood is shorter than 30 then it would be padded with spaces while printing.

The parameter material3 is defined without a colon and has the same meaning as <<:0>>.

*include

<<material3>>

The inner level LS-OPT GUI setup is shown in (Figure 20-53).



□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Transfer Variable	tbumper	3.008164904085			
Transfer Variable	thood	1.044201462547			
String	mat_b	mat_b_o	Values:	mat_b_1, mat_b_...	
String	mat_hood	mat_hood_orig	Values:	mat_hood_1, mat_h...	
Constant	material3	mat3			
Constant	m1	mat1			

Figure 20-53: Inner level optimization setup with string and transfer variables

Results

The optimum solution is obtained at `tbumper` = 3.01, `thood` = 1.04 `mat_b` = “mat_b_3”, `mat_hood` = “mat_hood_3”. The corresponding Mass value is 0.42 and there is no constraint violation at the solution. The outer level optimization history for the SRSM method is depicted in Figure 20-54.

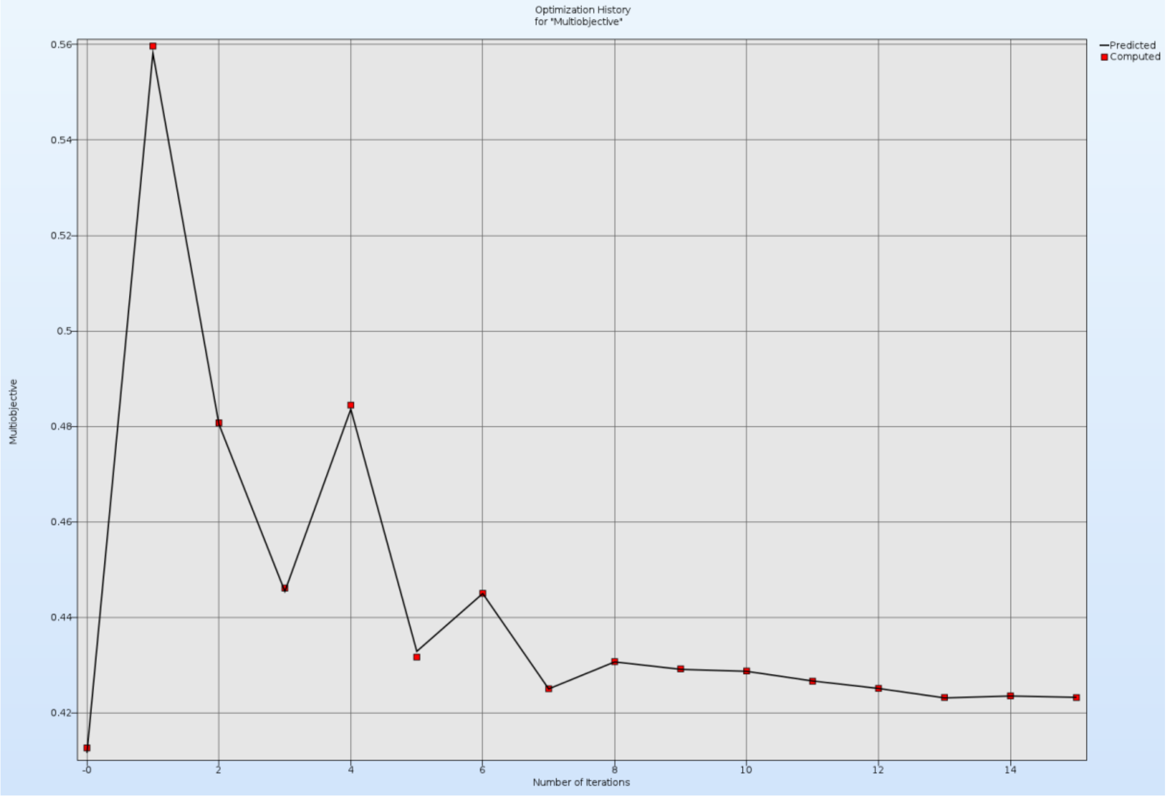


Figure 20-54: Outer level optimization history.

20.3. Impact of a cylinder (2 variables)

This example has the following features:

- LS-PREPOST is used to incorporate shape optimization.
- The LS-DYNA keyword *PERTURBATION is used to incorporate a geometric imperfection.
- An LS-DYNA explicit impact simulation is performed.
- Result extraction is performed using standard LS-DYNA interfaces.

The example in this chapter is modeled on one by Yamazaki [1].

20.3.1. Problem statement

The problem consists of a tube impacting a rigid wall as shown in Figure 20-55. The energy absorbed is maximized subject to a constraint on the rigid wall impact force. The cylinder has a constant mass of 0.52 kg with the design variables being the mean radius and thickness. The length of the cylinder is thus dependent on the design variables because of the mass constraint. A concentrated mass of 500 times the cylinder weight is attached to the end of the cylinder not impacting the rigid wall. The deformed shape at 20ms is shown in Figure 20-56 for a typical design.

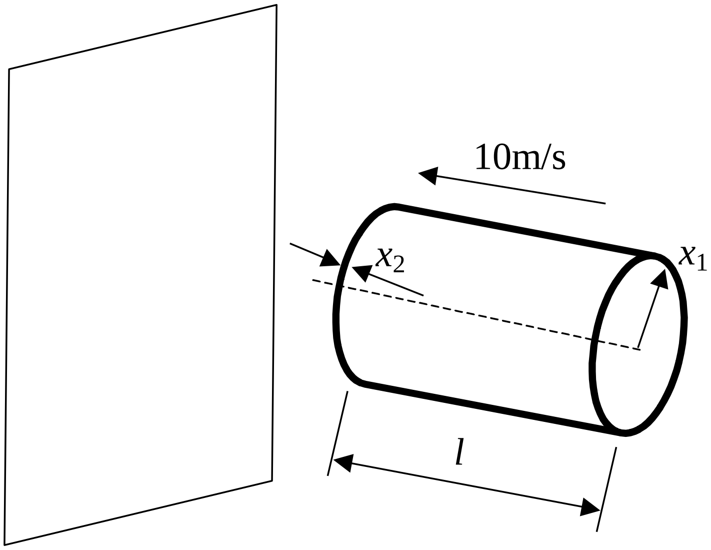


Figure 20-55: Impacting cylinder

The optimization problem is stated as

$$\text{Maximize } E_{internal}(x_1, x_2)|_{t=0.02}$$

subject to

$$\max\left(F_{normal}^{wall}(x_1, x_2)\right) \le 80,$$

$$l(x_1, x_2) = \frac{0.52}{2\pi\rho x_1 x_2}$$

where the design variables x_1 and x_2 are the radius and the thickness of the cylinder respectively. The internal energy $E_{\text{internal}}(x)\big|_{t=0.02}$ is the objective function and constraint functions $\max\left(F_{normal}^{wall}(x_1, x_2)\right)$ and $l(\boldsymbol{x})$ are the maximal normal force on the rigid wall and the length of the cylinder, respectively. The rigid wall force is filtered, frequencies exceeding 300Hz are excluded.

The problem is simulated using LS-DYNA. LS-PREPOST is used as a preprocessor to incorporate the geometrical parameters.

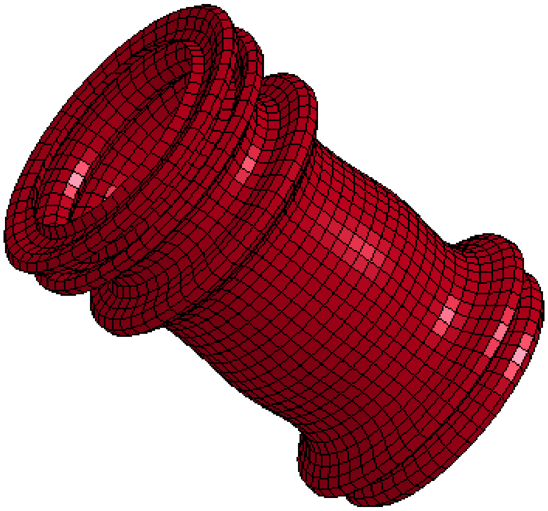


Figure 20-56: Deformed finite element model (time = 20ms)

20.3.2. Solution

The metamodel-based optimization method with linear metamodels, d-optimal point selection and strategy sequential with domain reduction is used.

The main LS-OPT GUI windows showing the process is displayed in Figure 20-57. LS-PREPOST is used to generate the finite element model of the cylinder depending on the parameter values, Figure 20-58. Since the LS-PREPOST output is used as include file in the LS-DYNA input, the file needs to be copied to the LS-DYNA run directories, Figure 20-59. Another option to make the file available for LS-DYNA is to run LS-PREPOST in the directories of stage RUN_DYNA.

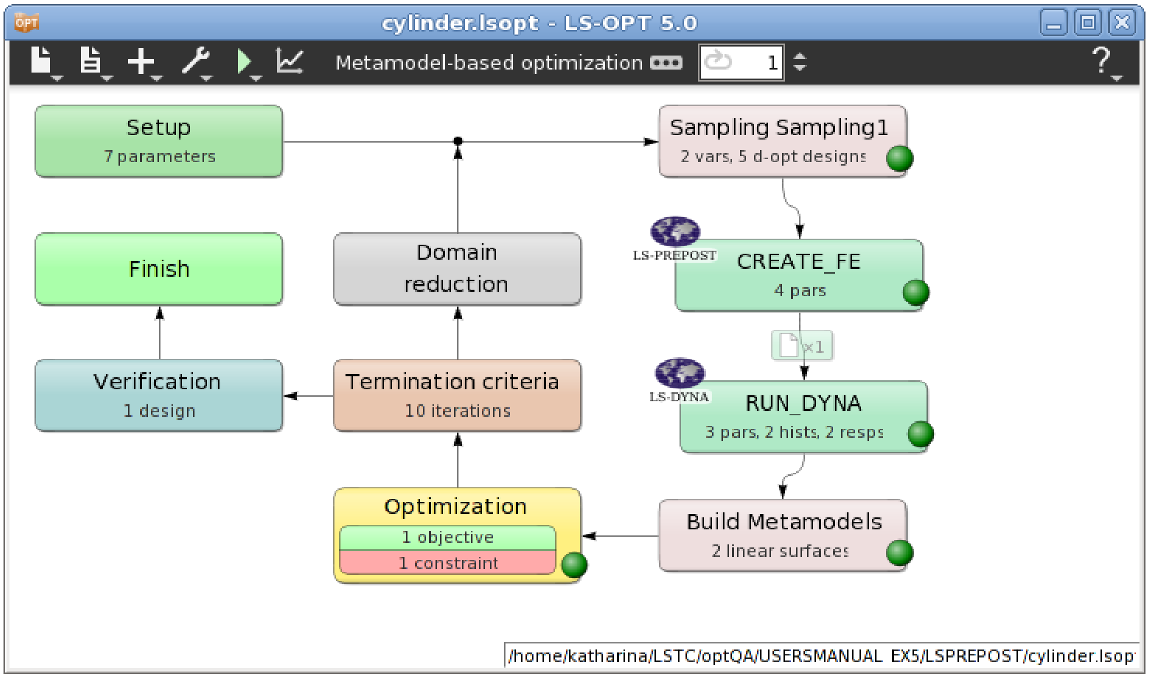
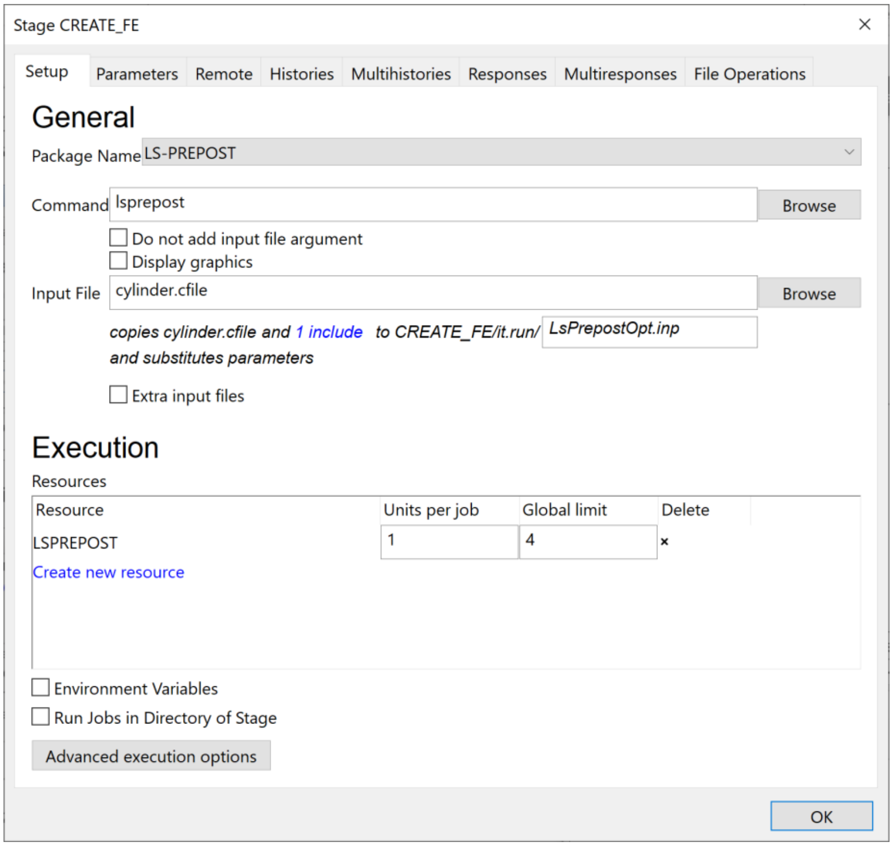


Figure 20-57: Main LS-OPT GUI window, LS-PREPOST is used as a preprocessor to incorporate geometrical parameters.



General

Package Name LS-PREPOST

Command lsprepost

□ Do not add input file argument
□ Display graphics

Input File cylinder.cfile

copies cylinder.cfile and 1 include to CREATE_FE/it.run/ LsPrepostOpt.inp
and substitutes parameters

□ Extra input files

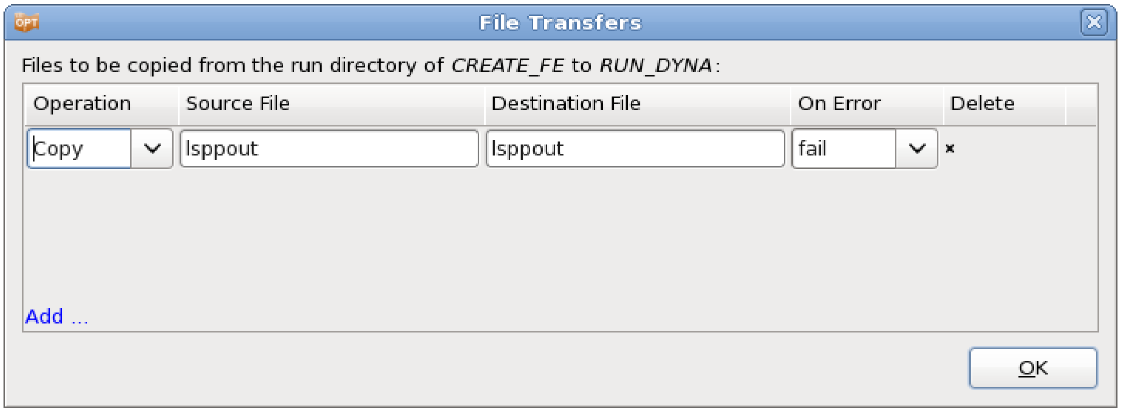
Execution

Resources

Resource	Units per job	Global limit	Delete
LSPREPOST	1	4	×

□ Environment Variables
□ Run Jobs in Directory of Stage

Figure 20-58: Stage dialog interfacing with LS-PREPOST



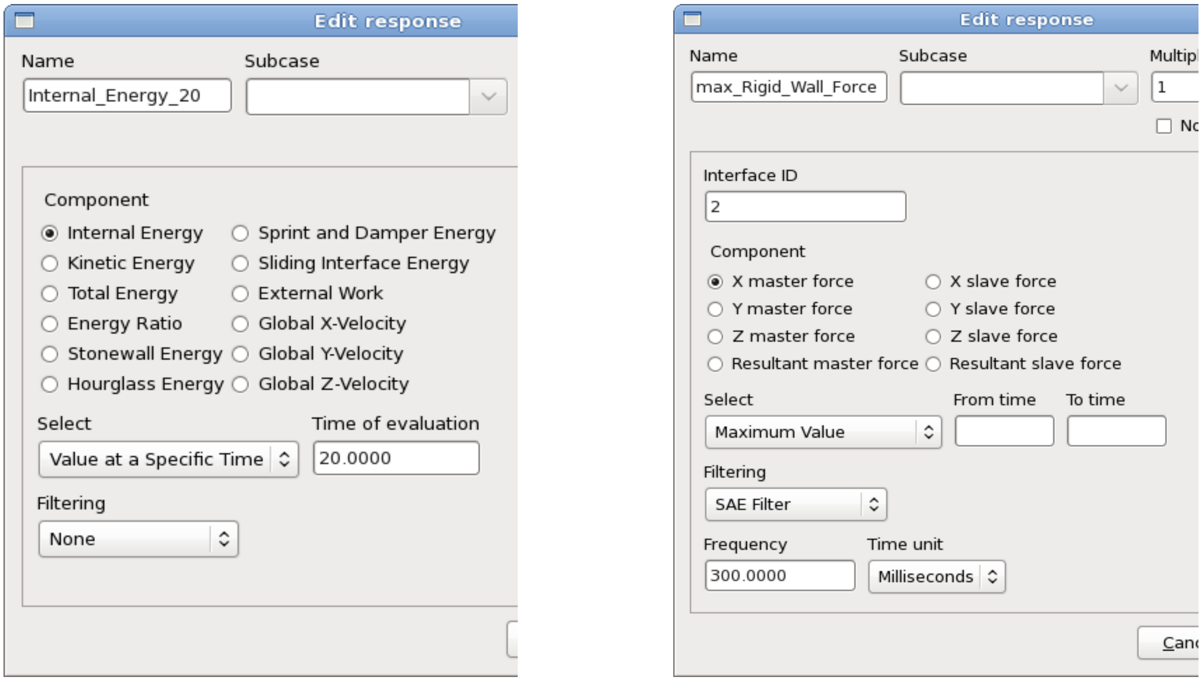
File Transfers

Files to be copied from the run directory of *CREATE_FE* to *RUN_DYNA*:

Operation	Source File	Destination File	On Error	Delete
Copy	lsppout	lsppout	fail	×

Figure 20-59: File transfer dialog: the output file of LS-PREPOST is copied to the LS-DYNA run directories.

To extract the rigid wall force and the internal energy, the LS-DYNA standard interfaces RCFORC and GLSTAT are used, respectively, Figure 20-60. The length of the cylinder is defined as a dependent of the radius and the thickness, also parameters concerning the element size and a value used for *PERTURBATION are defined as dependents, Figure 20-61.



Edit response

Name: Internal_Energy_20
Subcase:

Component
- ● Internal Energy
- ○ Kinetic Energy
- ○ Total Energy
- ○ Energy Ratio
- ○ Stonewall Energy
- ○ Hourglass Energy
- ○ Sprint and Damper Energy
- ○ Sliding Interface Energy
- ○ External Work
- ○ Global X-Velocity
- ○ Global Y-Velocity
- ○ Global Z-Velocity

Select: Value at a Specific Time
Time of evaluation: 20.0000
Filtering: None

Edit response

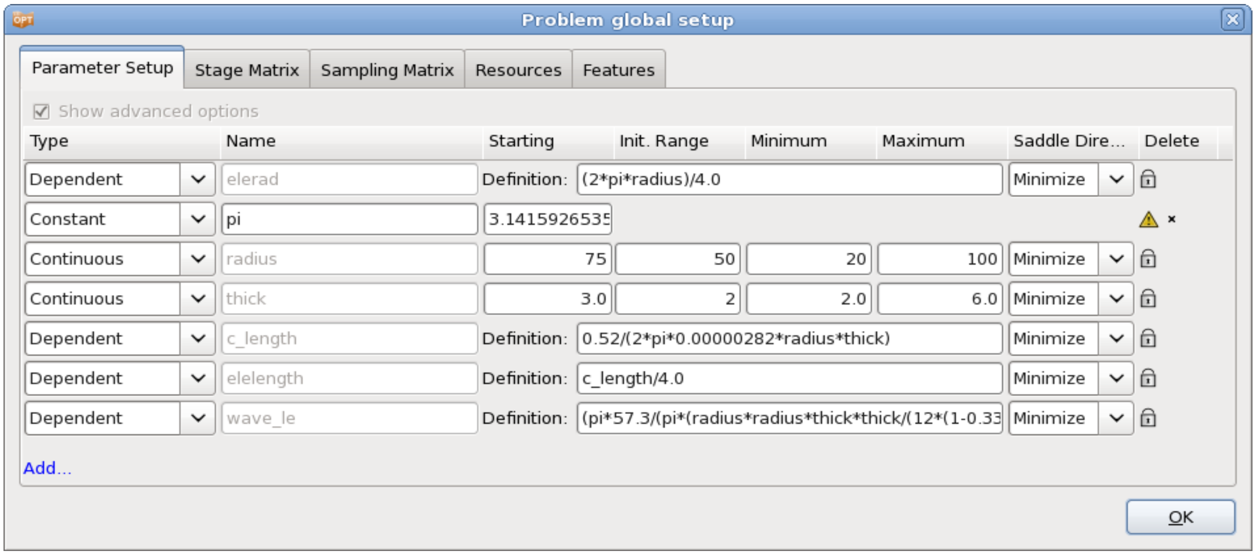
Name: max_Rigid_Wall_Force
Subcase:
Multipl: 1
☐ Nc

Interface ID: 2

Component
- ● X master force
- ○ Y master force
- ○ Z master force
- ○ Resultant master force
- ○ X slave force
- ○ Y slave force
- ○ Z slave force
- ○ Resultant slave force

Select: Maximum Value
From time:
To time:
Filtering: SAE Filter
Frequency: 300.0000
Time unit: Milliseconds

Figure 20-60: Result extraction using LS-DYNA interfaces GLSTAT and RCFORC



☑ Show advanced options

Type	Name	Starting	Init. Range	Minimum	Maximum	Saddle Dire…	Delete
Dependent	elerad	Definition:	(2*pi*radius)/4.0			Minimize	🔒
Constant	pi	3.1415926535					⚠ ×
Continuous	radius	75	50	20	100	Minimize	🔒
Continuous	thick	3.0	2	2.0	6.0	Minimize	🔒
Dependent	c_length	Definition:	0.52/(2*pi*0.00000282*radius*thick)			Minimize	🔒
Dependent	elelength	Definition:	c_length/4.0			Minimize	🔒
Dependent	wave_le	Definition:	(pi*57.3/(pi*(radius*radius*thick*thick/(12*(1-0.33			Minimize	🔒

Figure 20-61: Parameter Setup: the length of the cylinder depending on the radius and the thickness is defined as a dependent to satisfy the mass constraint.

20.3.3. Results

Figure 20-62 displays the deployment of the optimal values over the iterations for the variables, the constraint and the objective function, respectively. The initial design below shows that the constraint is severely exceeded. The optimization process steadily reduces the infeasibility. The final internal energy is significantly lower than the initial value to satisfy the constraint, but improved with respect to the value of iteration 4, where feasibility is reached the first time.

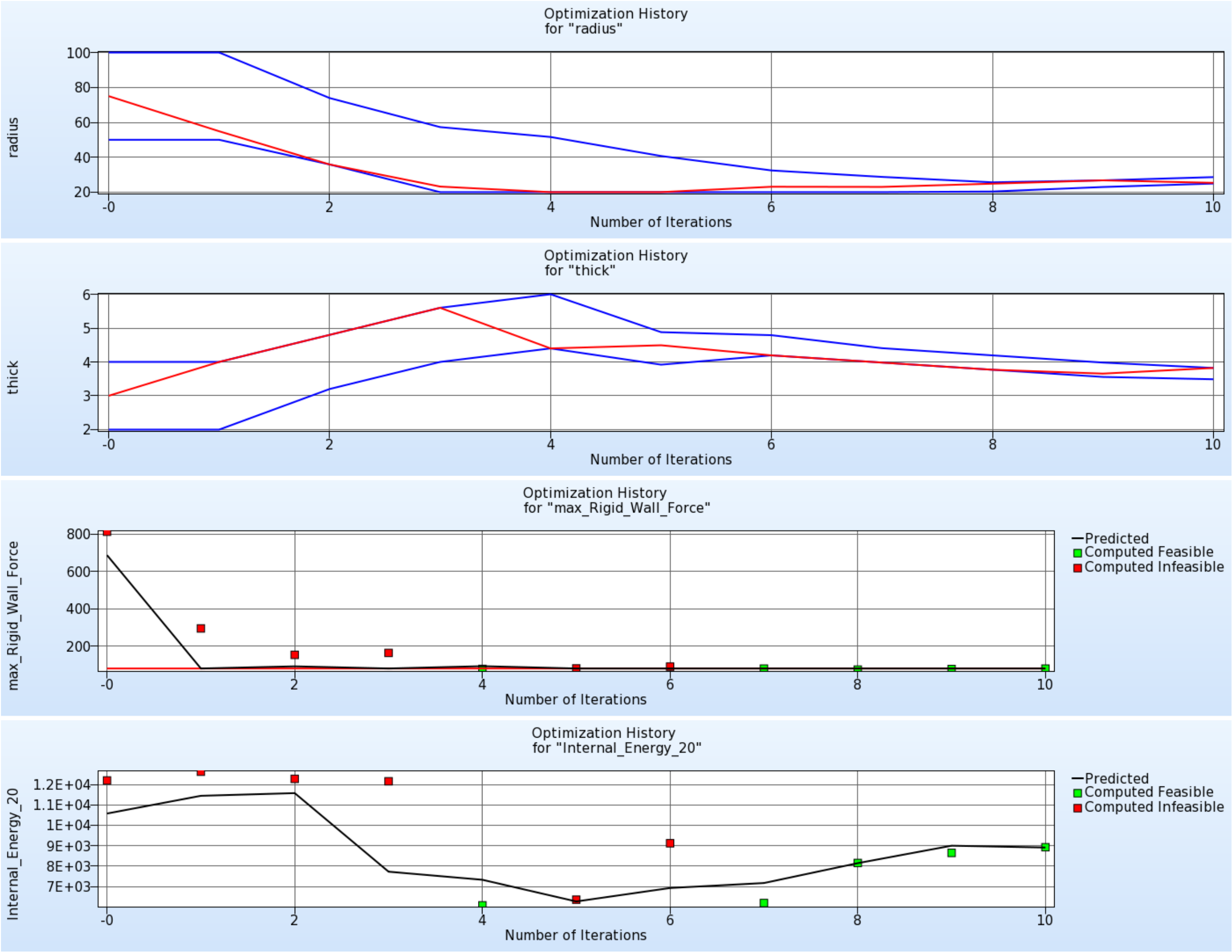


Figure 20-62: Optimization history of parameters, the constraint and objective function.

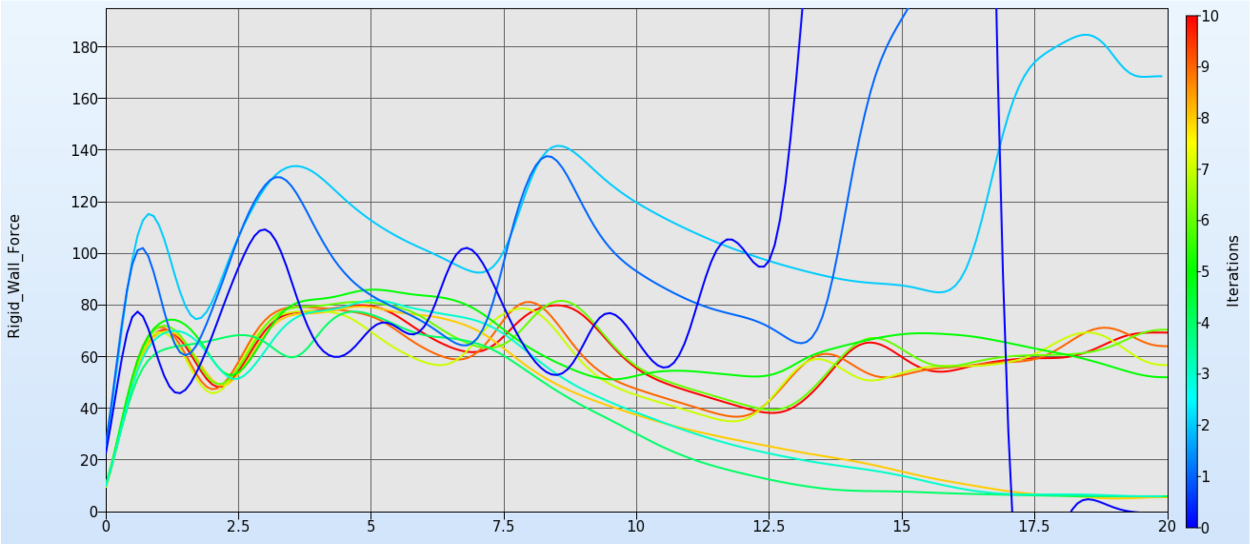


Figure 20-63: Cylinder: Constrained rigid wall force: F(t) < 80 (SAE 300Hz filtered); optimal curves of all iterations. The red curve is the final design.

20.4. Sheet-metal forming (3 variables)

A sheet-metal forming example in which the design involves thinning and FLD criteria is demonstrated in this chapter. The example has the following features:

- The example utilizes LS-PREPOST as preprocessor.
- *DEFINE_CURVE_TRIM is used to define the radius of the work piece.
- Adaptive meshing is used in the finite element analysis.
- The example employs the sheet metal forming interface utilities for result extraction.

20.4.1. Problem statement

The design variables are the radii of the work tool and the radius of the work piece as indicated in Figure 20-64. The design problem is formulated to minimize both tool radii while also specifying FLD constraints and a maximum thickness reduction of 25%. Hence the radii are variables and objectives at the same time. Adaptive meshing is chosen as an analysis feature for the simulation. The FE model is shown in Figure 20-65.

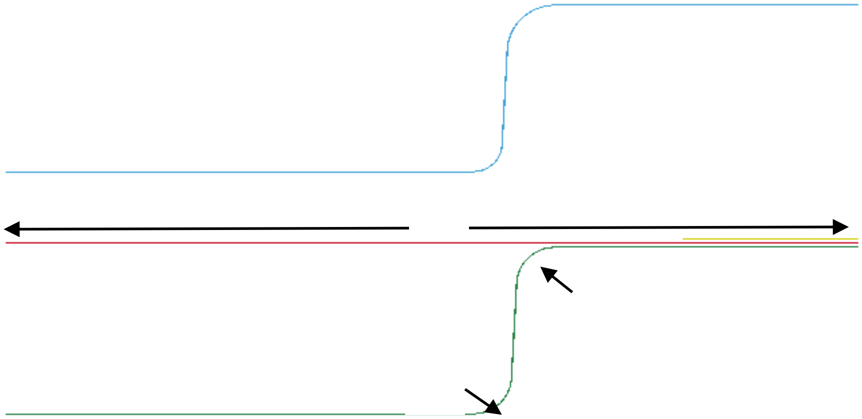


Figure 20-64: Parameterization of cross-section

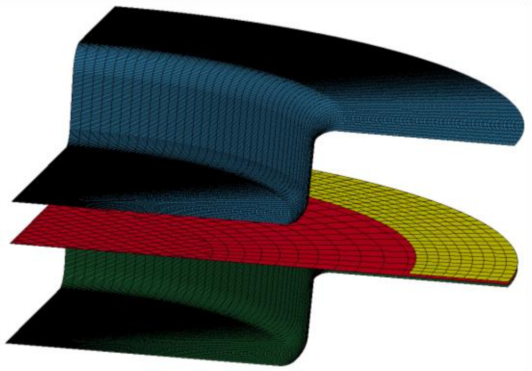


Figure 20-65: Quarter segment of FE model: tools and blank

20.4.2. Solution

LS-PREPOST is used as a preprocessor to incorporate the geometrical parameters, Figure 20-66. In a second stage, a trimming simulation is preformed using LS-DYNA. The thickness reduction and the FLD constraints are extracted from the forming simulation results using the FLD and THICK response interfaces, Figure 20-67. For each radius, a composite function is generated to be used as an objective function, Figure 20-68 and Figure 20-69.

The definition of the FLD and thickness reduction constraints is displayed in Figure 20-70.

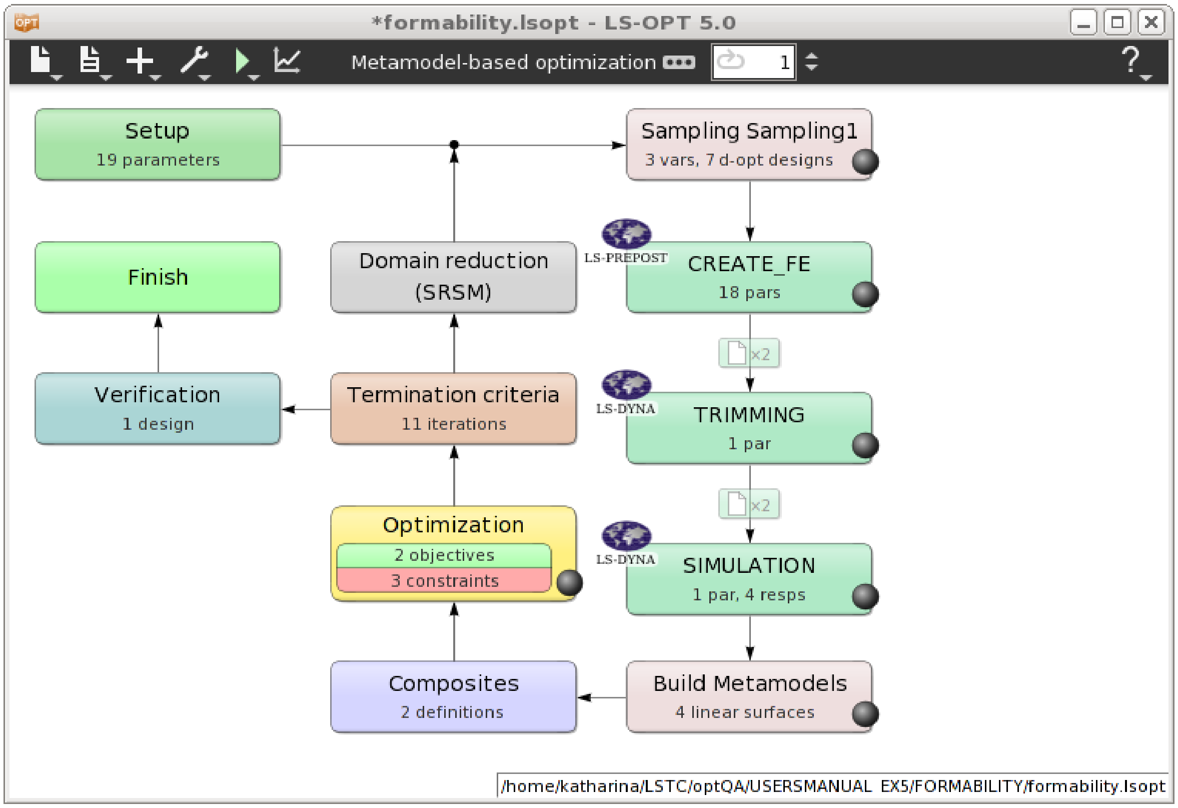
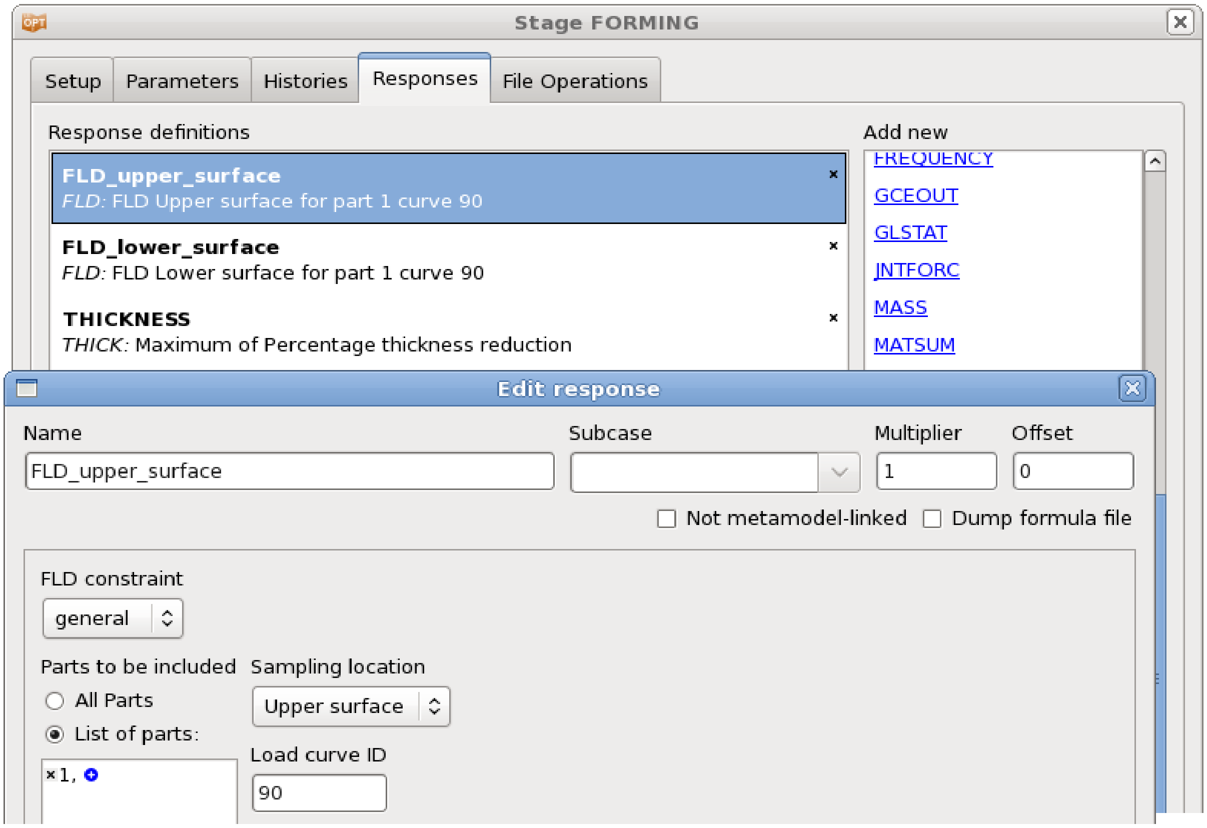


Figure 20-66: Main LS-OPT GUI window



Response definitions

FLD_upper_surface
FLD: FLD Upper surface for part 1 curve 90

FLD_lower_surface
FLD: FLD Lower surface for part 1 curve 90

THICKNESS
THICK: Maximum of Percentage thickness reduction

Name	Subcase	Multiplier	Offset
FLD_upper_surface		1	0

□ Not metamodel-linked □ Dump formula file

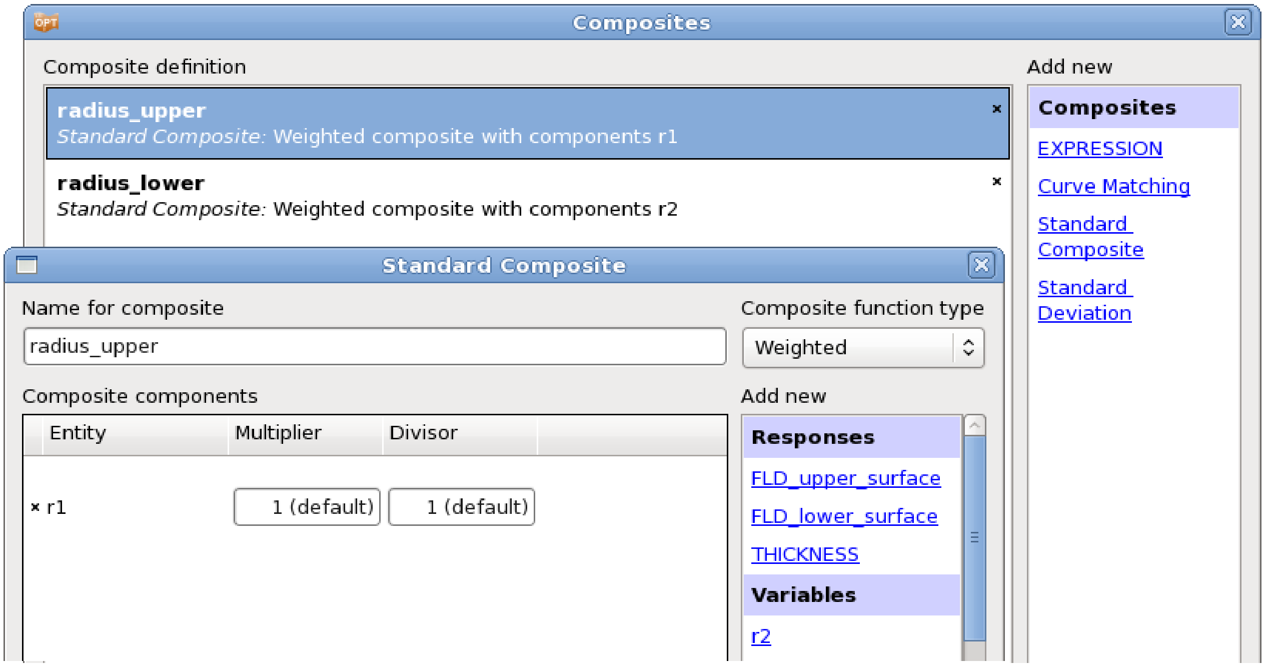
FLD constraint: general

Parts to be included: ○ All Parts ● List of parts: 1,

Sampling location: Upper surface

Load curve ID: 90

Figure 20-67: Interface for FLD response extraction



Composite definition

radius_upper
Standard Composite: Weighted composite with components r1

radius_lower
Standard Composite: Weighted composite with components r2

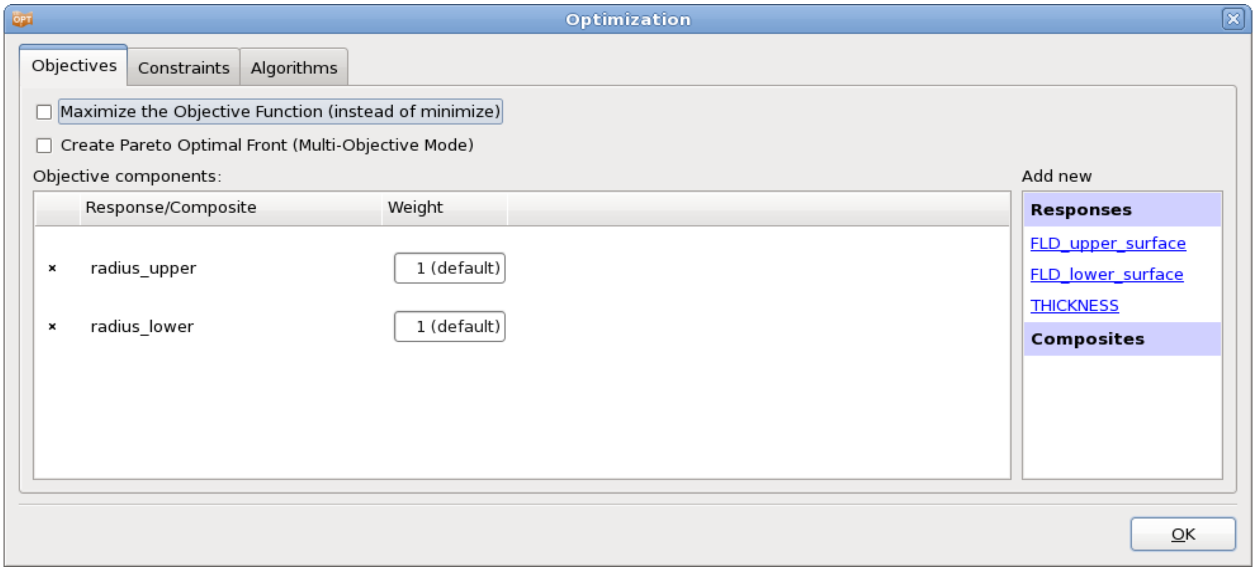
Name for composite: radius_upper

Composite function type: Weighted

Composite components

Entity	Multiplier	Divisor
r1	1 (default)	1 (default)

Figure 20-68: Definition of composite functions



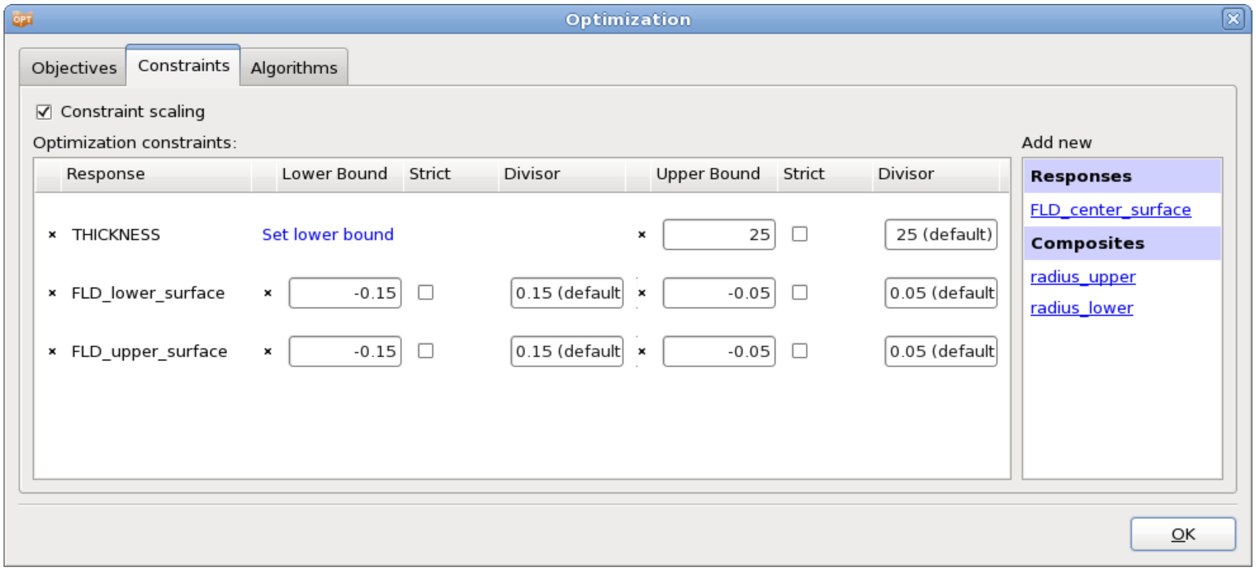
□ Maximize the Objective Function (instead of minimize)

□ Create Pareto Optimal Front (Multi-Objective Mode)

Objective components:

	Response/Composite	Weight
×	radius_upper	1 (default)
×	radius_lower	1 (default)

Figure 20-69: Definition of objective functions



☑ Constraint scaling

Optimization constraints:

	Response		Lower Bound	Strict	Divisor		Upper Bound	Strict	Divisor
×	THICKNESS	Set lower bound				×	25	□	25 (default)
×	FLD_lower_surface	×	-0.15	□	0.15 (default	×	-0.05	□	0.05 (default
×	FLD_upper_surface	×	-0.15	□	0.15 (default	×	-0.05	□	0.05 (default

Figure 20-70: Definition of constraint functions

20.4.3. Results

The optimization history for the objectives and constraints is shown in Figure 20-71 and Figure 20-72. A comparison between the starting and the final values is tabulated below, Table 20-6. The FLD diagrams (Figure 20-73) for the baseline design and the optimum illustrate the improvement of the FLD feasibility. A typical deformed state is depicted in Figure 20-75.

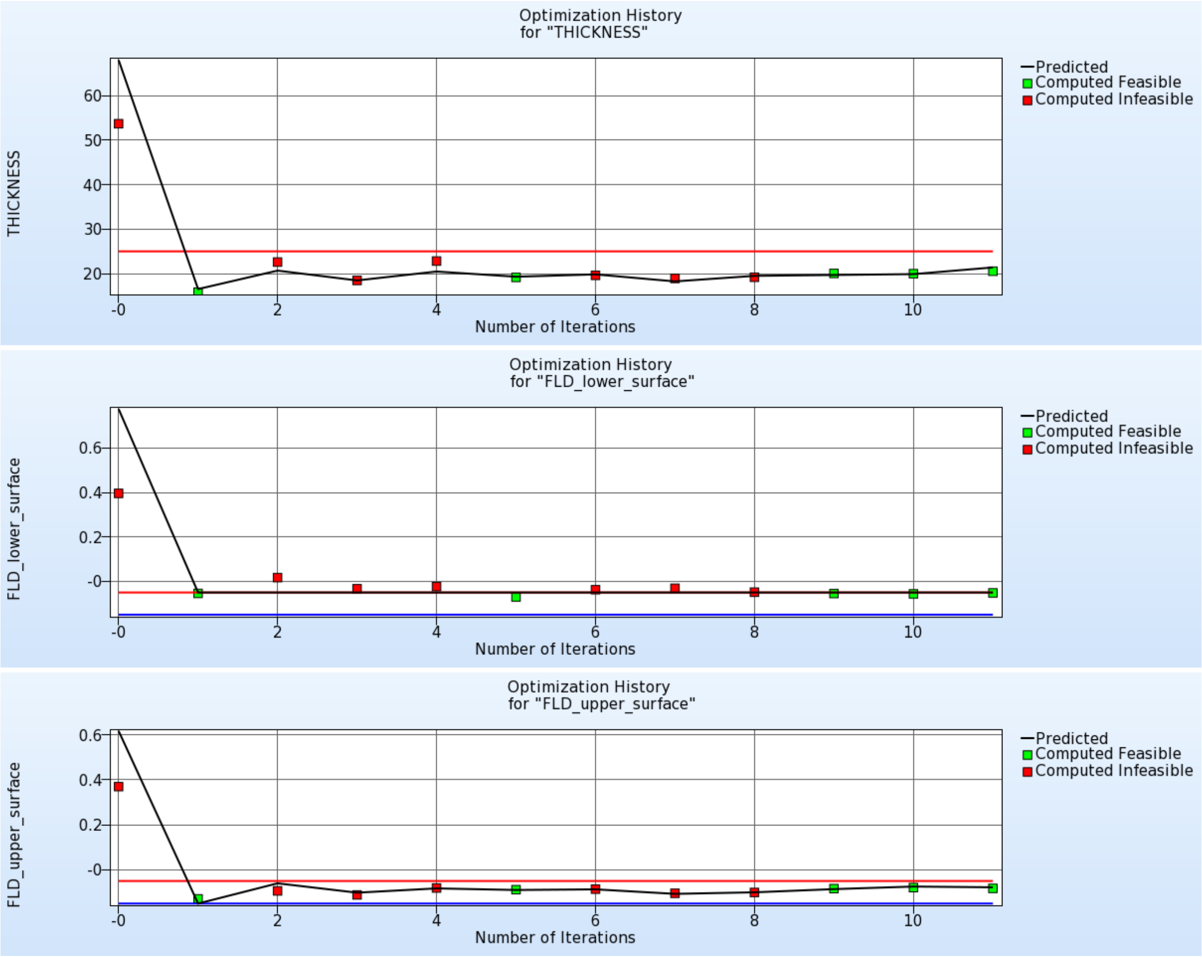


Figure 20-71: Optimization history of FLD constraints and thickness reduction

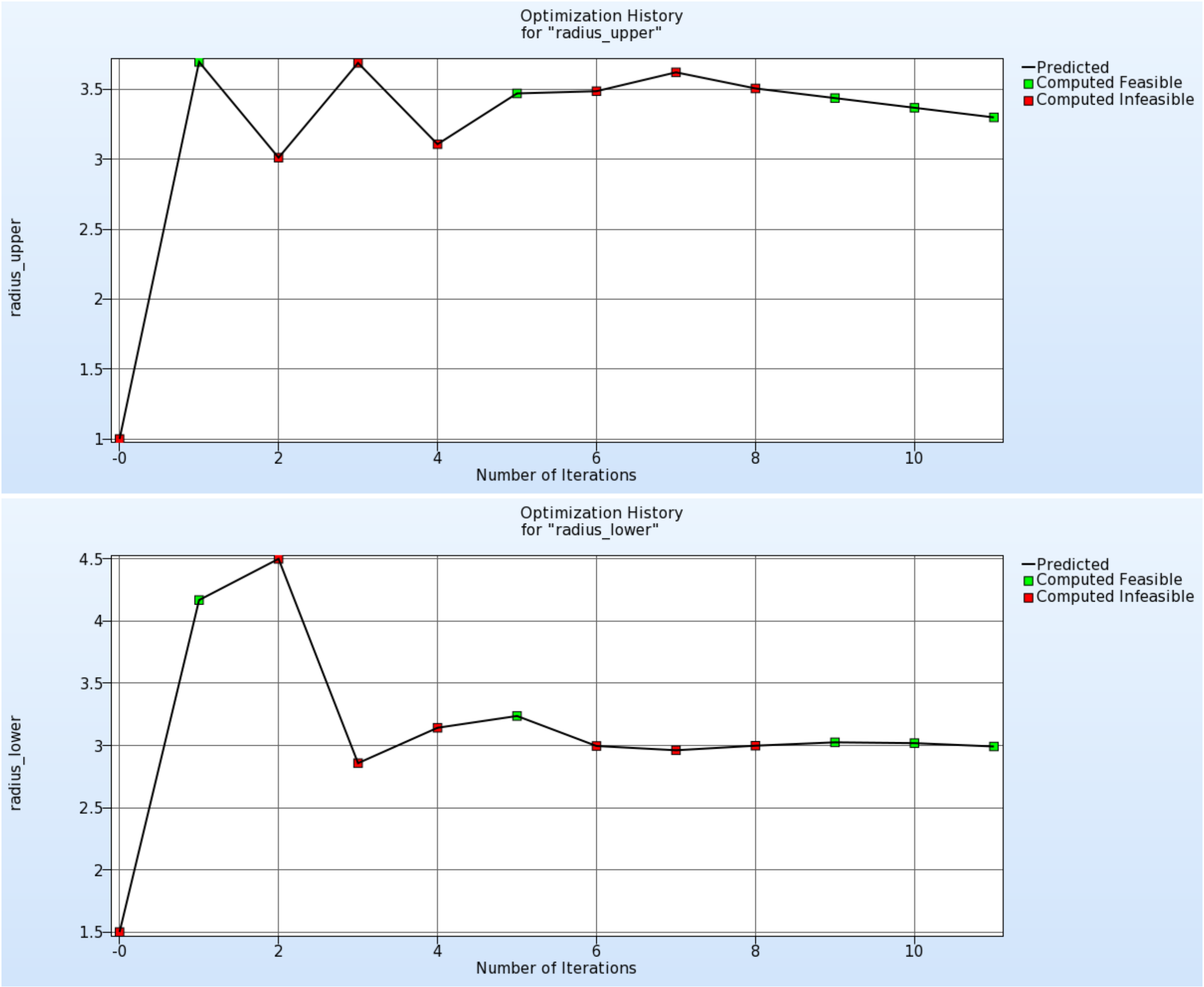
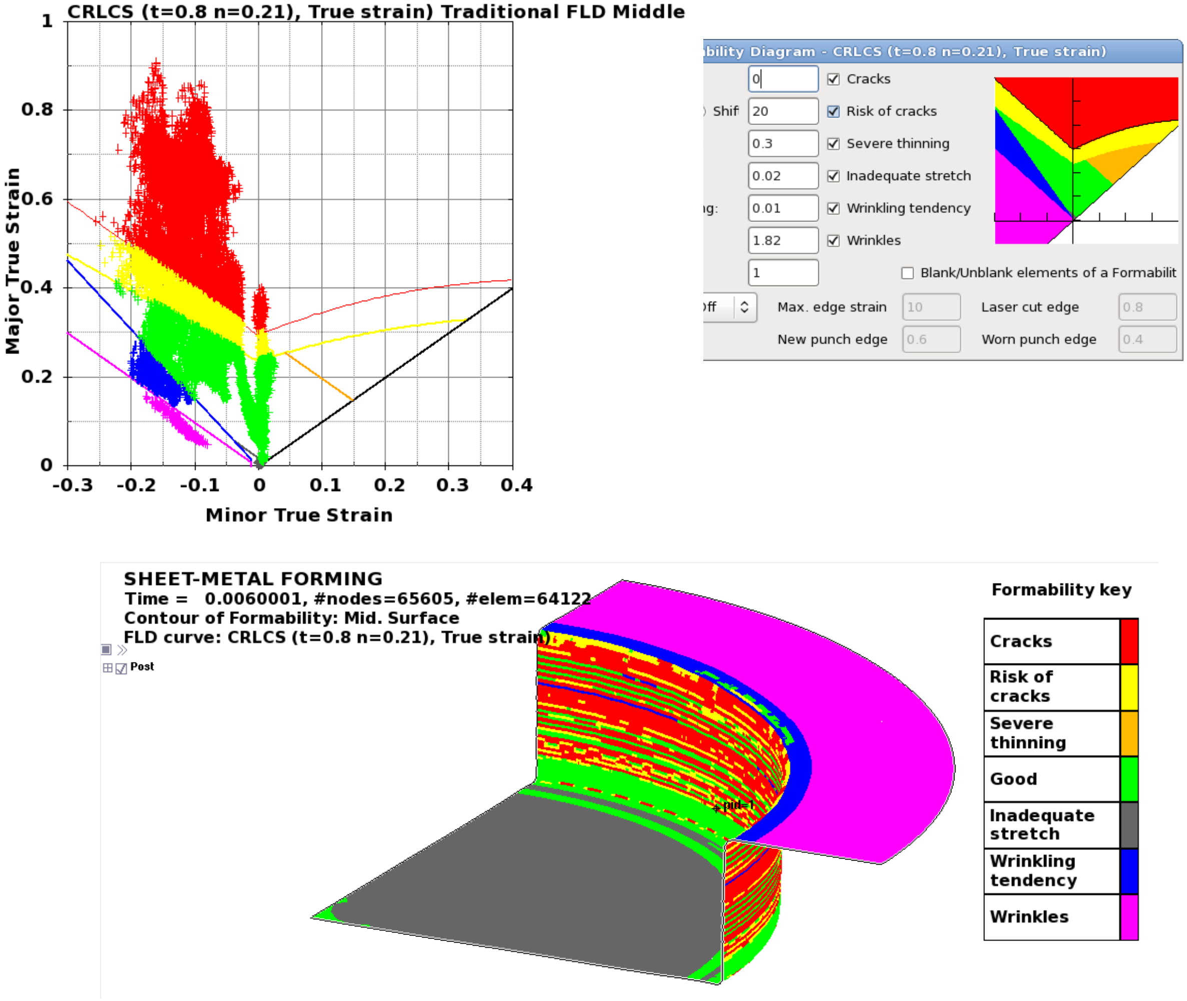


Figure 20-72: Optimization History of objectives radius_upper and radius_lower

Table 20-6: Comparison of results (Sheet-metal forming)

Variable	Start (Computed)	Optimal (Predicted)	Optimal (Computed)
THICKNESS	53.67	21.38	20.54
FLD_upper_surface	0.369	-0.078	-0.083
FLD_lower_surface	0.396	.-0.050	-0.052
radius_upper	1	3.30	-
radius_lower	1.5	2.99	-

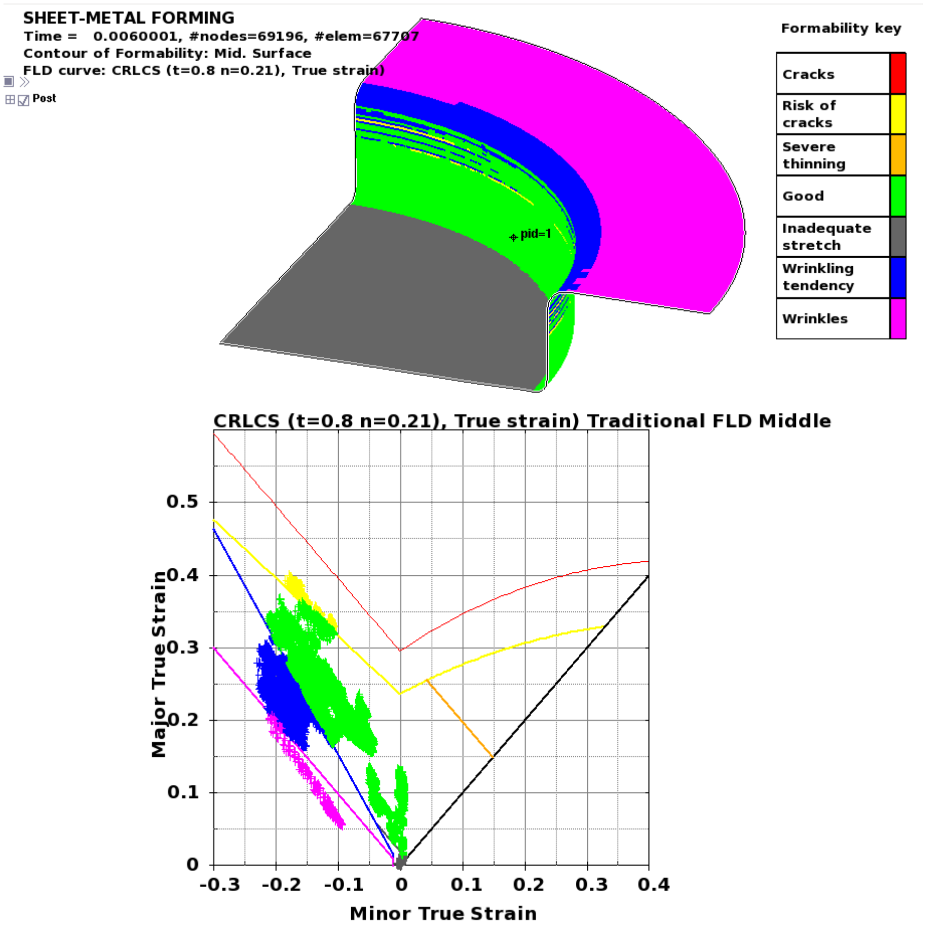


bility Diagram - CRLCS (t=0.8 n=0.21), True strain)

0 ☑ Cracks
) Shif 20 ☑ Risk of cracks
0.3 ☑ Severe thinning
0.02 ☑ Inadequate stretch
ıg: 0.01 ☑ Wrinkling tendency
1.82 ☑ Wrinkles
1 ☐ Blank/Unblank elements of a Formabilit
ff Max. edge strain 10 Laser cut edge 0.8
New punch edge 0.6 Worn punch edge 0.4

SHEET-METAL FORMING
Time = 0.0060001, #nodes=65605, #elem=64122
Contour of Formability: Mid. Surface
FLD curve: CRLCS (t=0.8 n=0.21), True strain)

Figure 20-73: FLD diagrams of baseline run



SHEET-METAL FORMING
Time = 0.0060001, #nodes=69196, #elem=67707
Contour of Formability: Mid. Surface
FLD curve: CRLCS (t=0.8 n=0.21), True strain)

Figure 20-74: FLD diagram of optimal run

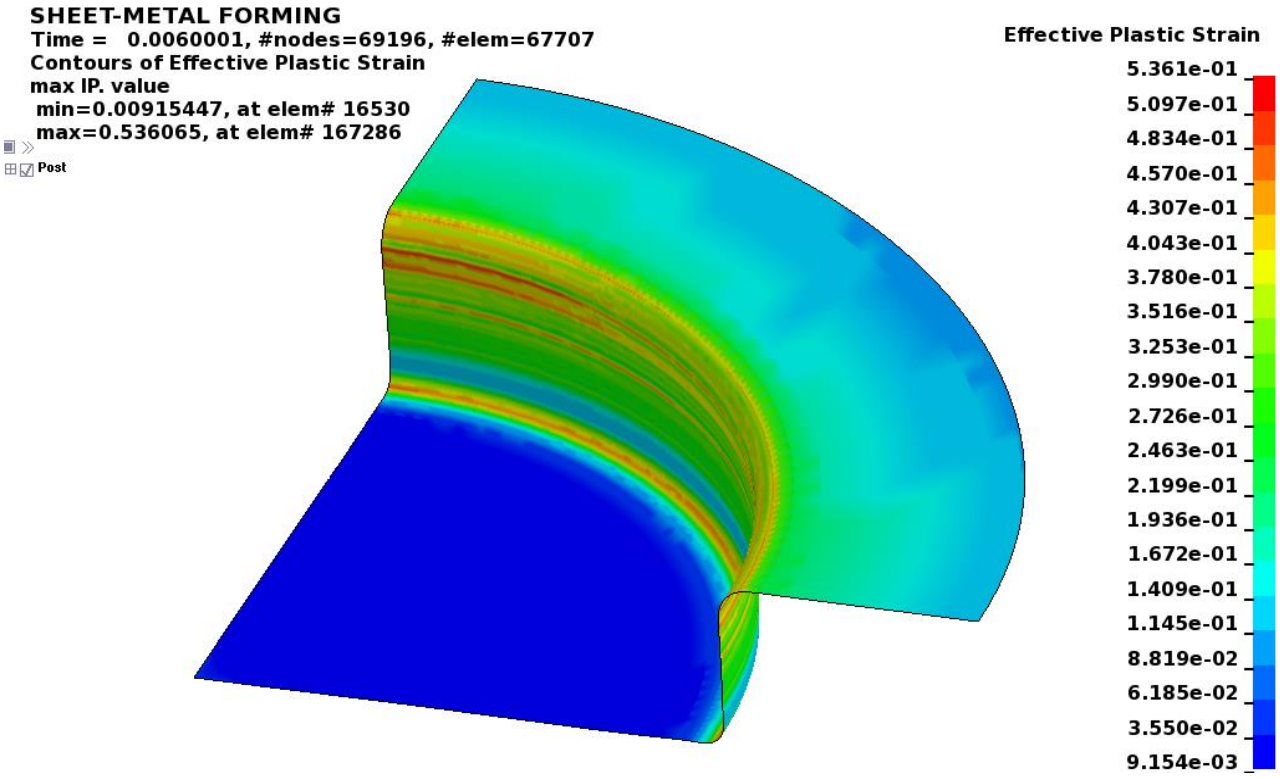


Figure 20-75: Deformed state (optimal run), Fringe component plastic strain

20.5. Large vehicle crash and vibration (MDO/MOO) (7 variables)

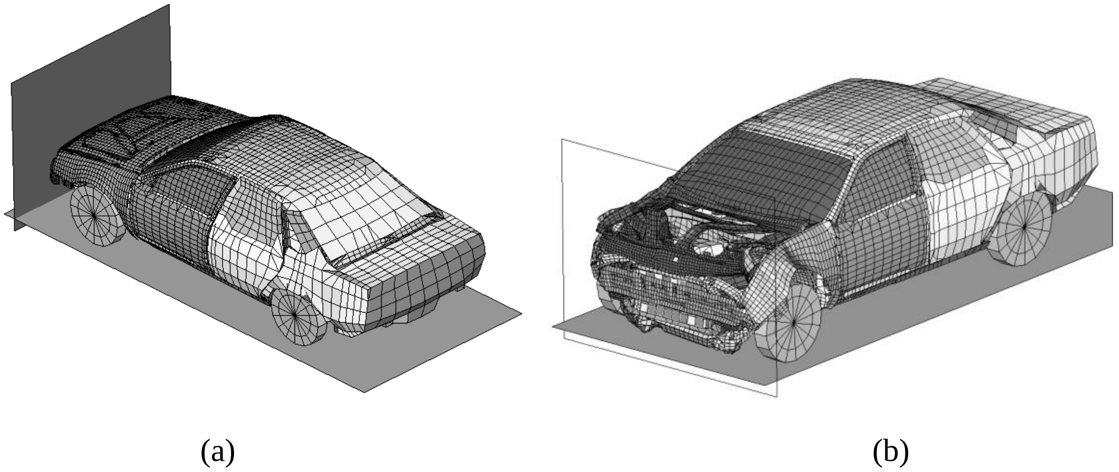
This example has the following features:

- LS-DYNA is used for both explicit full frontal crash and implicit NVH simulations.
- Multidisciplinary design optimization (MDO) and Multi-objective optimization (MOO) are illustrated with a realistic full vehicle example.
- Extraction is performed using standard LS-DYNA interfaces.
- Complex mathematical response expressions are used.

This example illustrates a realistic application of Multidisciplinary Design Optimization (MDO) and concerns the coupling of the crash performance of a large vehicle with one of its Noise Vibration and Harshness (NVH) criteria, namely the torsional mode frequency [2].

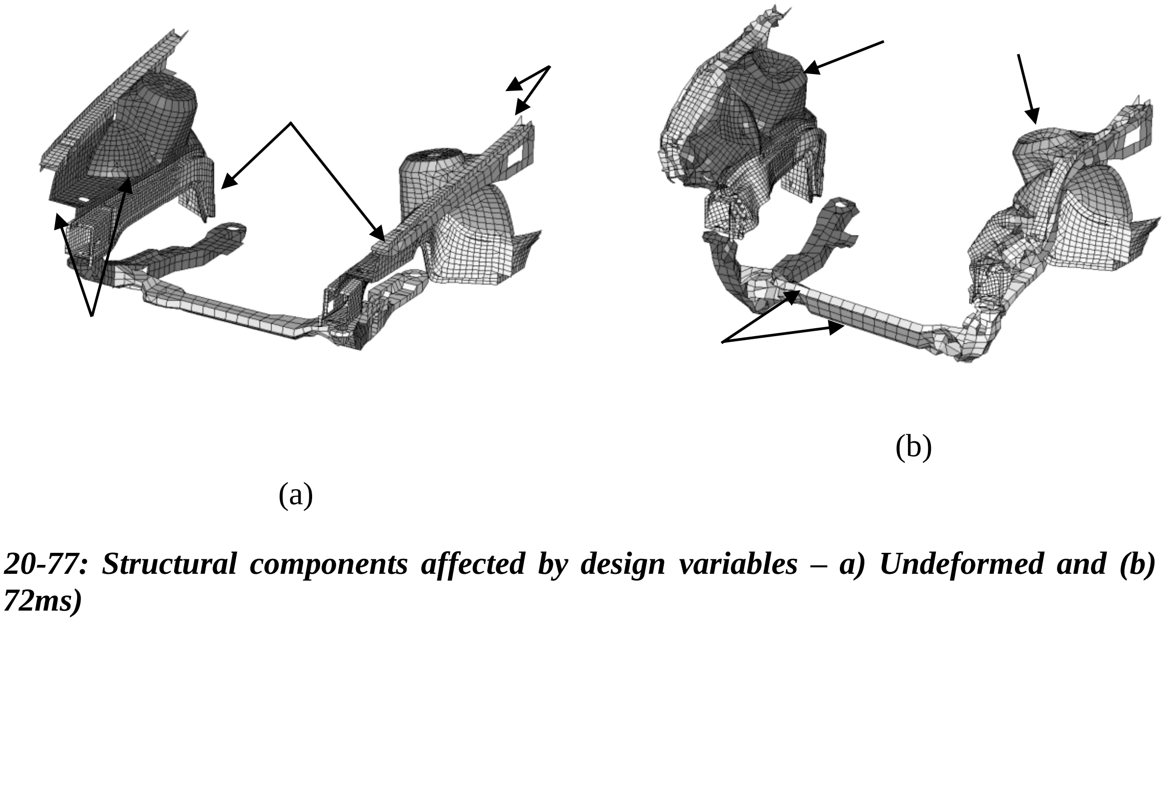
20.5.1. FE Modeling

The crashworthiness simulation considers a model containing approximately 30,000 elements of a National Highway Transportation and Safety Association (NHTSA) vehicle [3] undergoing a full frontal impact. A modal analysis is performed on a so-called ‘body-in-white’ model containing approximately 18,000 elements. The crash model for the full vehicle is shown in Figure 20-76 for the undeformed and deformed (time = 78ms) states, and with only the structural components affected by the design variables, both in the undeformed and deformed (time = 72ms) states, in Figure 20-77. The NVH model is depicted in Figure 20-78 in the first torsion vibrational mode. Only body parts that are crucial to the vibrational mode shapes are retained in this model. The design variables are all thicknesses or gages of structural components in the engine compartment of the vehicle (Figure 20-77), parameterized directly in the LS-DYNA input file. Twelve parts are affected, comprising aprons, rails, shotguns, cradle rails and the cradle cross member (Figure 20-77). LS-DYNA v.971 is used for both the crash and NVH simulations, in explicit and implicit modes respectively.



(a) (b)

Figure 20-76: Crash model of vehicle showing road and wall a) Undeformed b) Deformed (78ms)



(b)

(a)

Figure 20-77: Structural components affected by design variables – a) Undeformed and (b) deformed (time = 72ms)

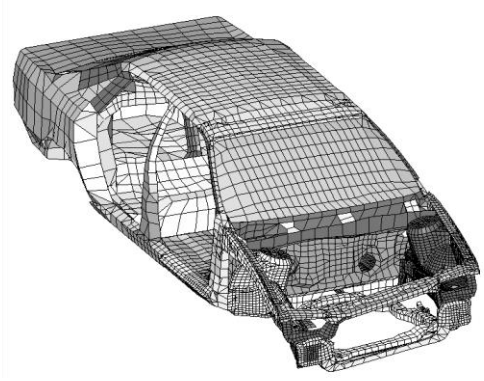


Figure 20-78: Body-in-white model of vehicle in torsional vibration mode (38.7Hz)

20.5.2. Design formulation

The formulation is as follows:

Minimize Mass

Minimize Maximum intrusion

subject to

Stage 1 pulse($\boldsymbol{x}_{\text{crash}}$) > 14.51g

Stage 2 pulse($\boldsymbol{x}_{\text{crash}}$) > 17.59g

Stage 3 pulse($\boldsymbol{x}_{\text{crash}}$) > 20.75g

41.38Hz < Torsional mode frequency($\boldsymbol{x}_{\text{NVH}}$) < 42.38Hz

The three stage pulses are calculated from the SAE filtered (60Hz) acceleration and displacement of a left rear sill node in the following fashion:

$$\text{Stage } i \text{ pulse} = \frac{-k}{d_2 - d_1} \int_{d_1}^{d_2} a\mathrm{d}x \ ;$$

$k = 2.0$ for $i = 1$, 1.0 otherwise;

with the limits $[d_1;d_2] = [0;184]$; $[184;334]$; $[334;\text{Max(displacement)}]$ for $i = 1,2,3$ respectively, all displacement units in mm and the minus sign to convert acceleration to deceleration. The Stage 1 pulse is represented by a triangle with the peak value being the value used.

20.5.3. Multi-objective optimization using metamodel-based optimization

The MDO and MOO features are specified as follows:

- **MDO.** The two disciplines (crash and NVH) are treated separately.

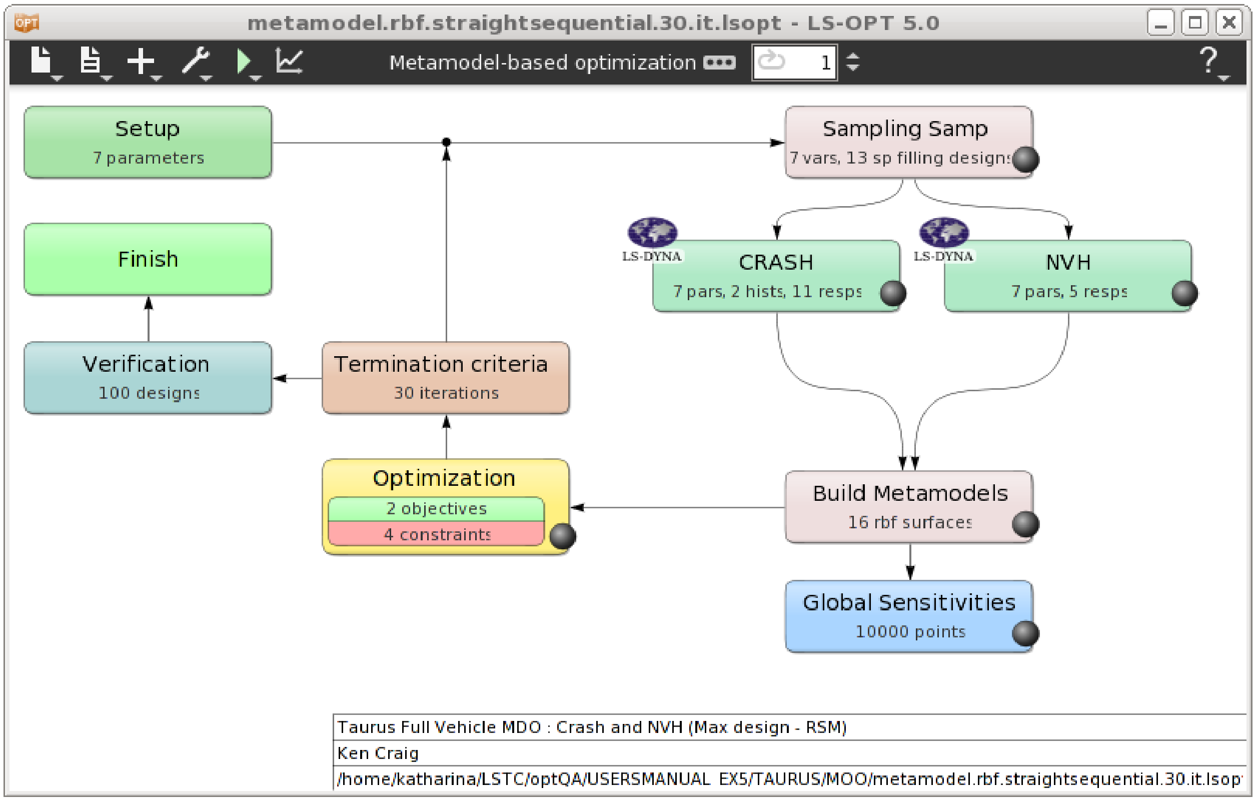
- **MOO.** Two design objectives (Intrusion and mass) are stated. The GA must be selected (in the Algorithms panel of the Optimization dialog or in the Task dialog) as metamodel optimizer to obtain the Pareto optimal front.

Figure 20-79 shows the LS-OPT main GUI window for a multi-disciplinary optimization using metamodels.

For the main task, select a metamodel-based optimization, Figure 20-80. Since Pareto Optimal solutions are generated, make sure to use a global strategy. To get a good approximation of the whole design space, choose a non-linear metamodel type, e.g. Radial Basis Functions, Figure 20-81. Since we use the sequential strategy, the default number of points per iteration per case is appropriate.

The displacements and the acceleration for the crash load case may be evaluated using the standard LS-DYNA interfaces, whereas more complex expressions are needed to calculate the stage pulses. The Lookup function may be used to get the value of t for a specified value of the selected history function, Figure 20-82. Then the stage pulses may be calculated using the Integral function, Figure 20-83.

For the NVH load case, the FREQUENCY interface may be used to extract the frequency and related responses, Figure 20-84. Make sure that mode tracking is used.

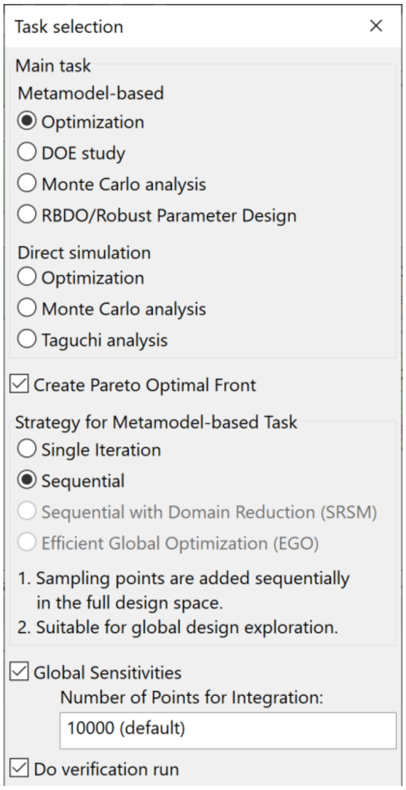


Taurus Full Vehicle MDO : Crash and NVH (Max design - RSM)

Ken Craig

/home/katharina/LSTC/optQA/USERSMANUAL_EX5/TAURUS/MOO/metamodel.rbf.straightsequential.30.it.lsop

Figure 20-79: Main LS-OPT GUI; Metamodel based optimization; two disciplines.



Task selection

Main task

Metamodel-based

● Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

○ Taguchi analysis

☑ Create Pareto Optimal Front

Strategy for Metamodel-based Task

○ Single Iteration

● Sequential

○ Sequential with Domain Reduction (SRSM)

○ Efficient Global Optimization (EGO)

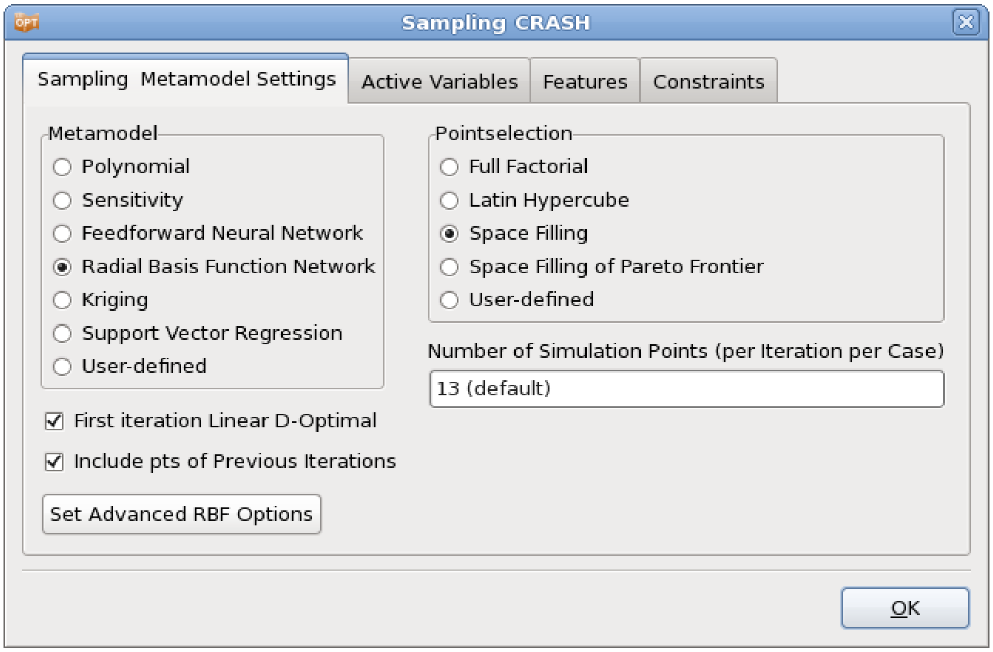
1. Sampling points are added sequentially in the full design space.
2. Suitable for global design exploration.

☑ Global Sensitivities

Number of Points for Integration: 10000 (default)

☑ Do verification run

Figure 20-80: Task dialog; Calculating Pareto Optimal solutions using a metamodel-base method using sequential strategy.



Sampling CRASH

Metamodel

○ Polynomial

○ Sensitivity

○ Feedforward Neural Network

● Radial Basis Function Network

○ Kriging

○ Support Vector Regression

○ User-defined

Pointselection

○ Full Factorial

○ Latin Hypercube

● Space Filling

○ Space Filling of Pareto Frontier

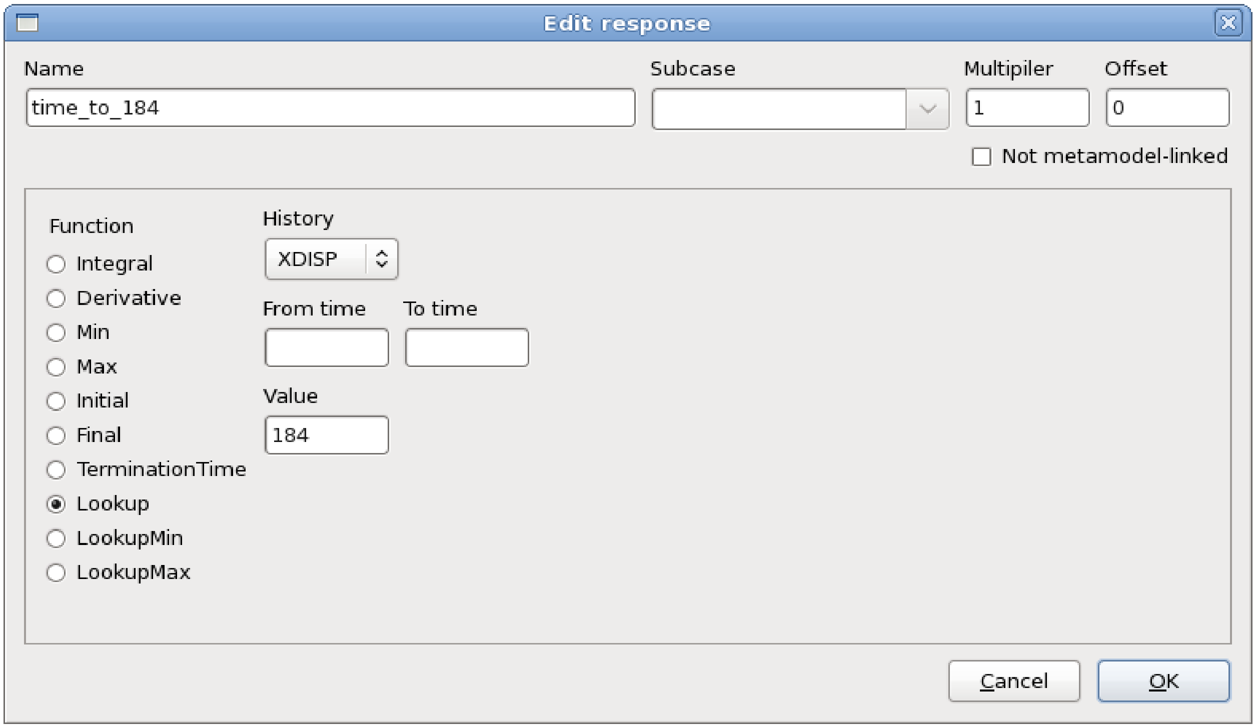
○ User-defined

Number of Simulation Points (per Iteration per Case): 13 (default)

☑ First iteration Linear D-Optimal

☑ Include pts of Previous Iterations

Figure 20-81: Sampling dialog; use Radial Basis Functions to get a global approximation.



Name: time_to_184

Subcase:

Multipiler: 1

Offset: 0

□ Not metamodel-linked

Function
- ○ Integral
- ○ Derivative
- ○ Min
- ○ Max
- ○ Initial
- ○ Final
- ○ TerminationTime
- ● Lookup
- ○ LookupMin
- ○ LookupMax

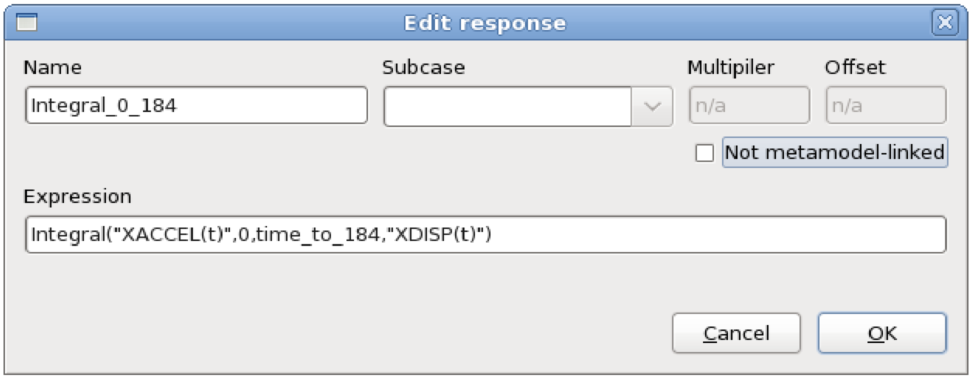
History: XDISP

From time:

To time:

Value: 184

Figure 20-82: Lookup function; evaluate the value of t for a specified value of the history XDISP



Name: Integral_0_184

Subcase:

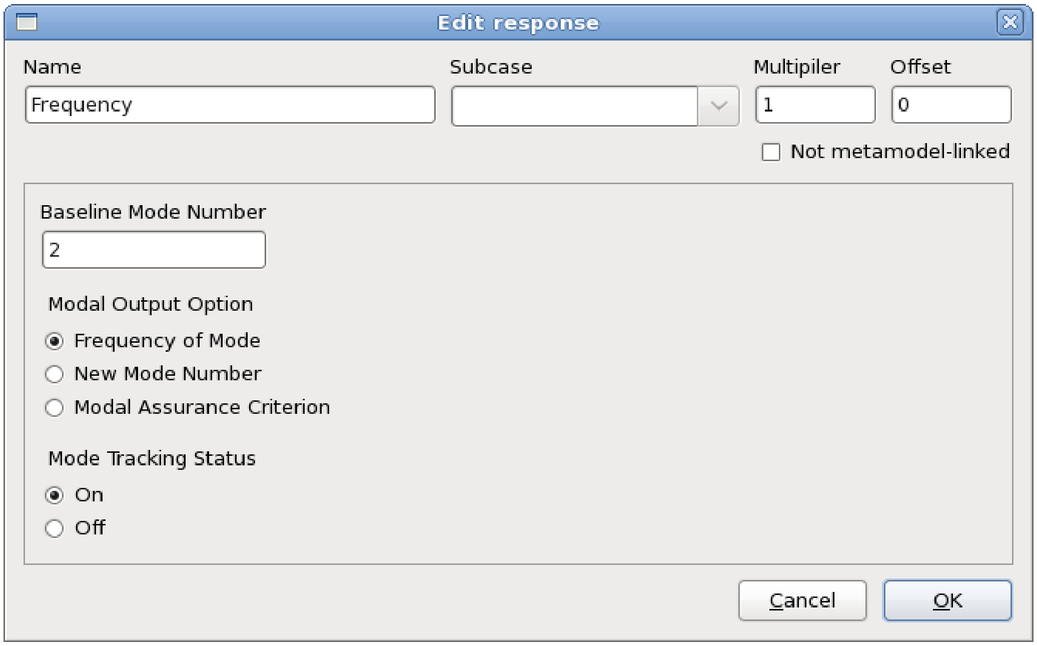
Multipiler: n/a

Offset: n/a

□ Not metamodel-linked

Expression: Integral("XACCEL(t)",0,time_to_184,"XDISP(t)")

Figure 20-83: Response Expression; The stage pulses are calculated using the Integral function.



Name: Frequency

Subcase:

Multipiler: 1

Offset: 0

☐ Not metamodel-linked

Baseline Mode Number: 2

Modal Output Option

● Frequency of Mode

○ New Mode Number

○ Modal Assurance Criterion

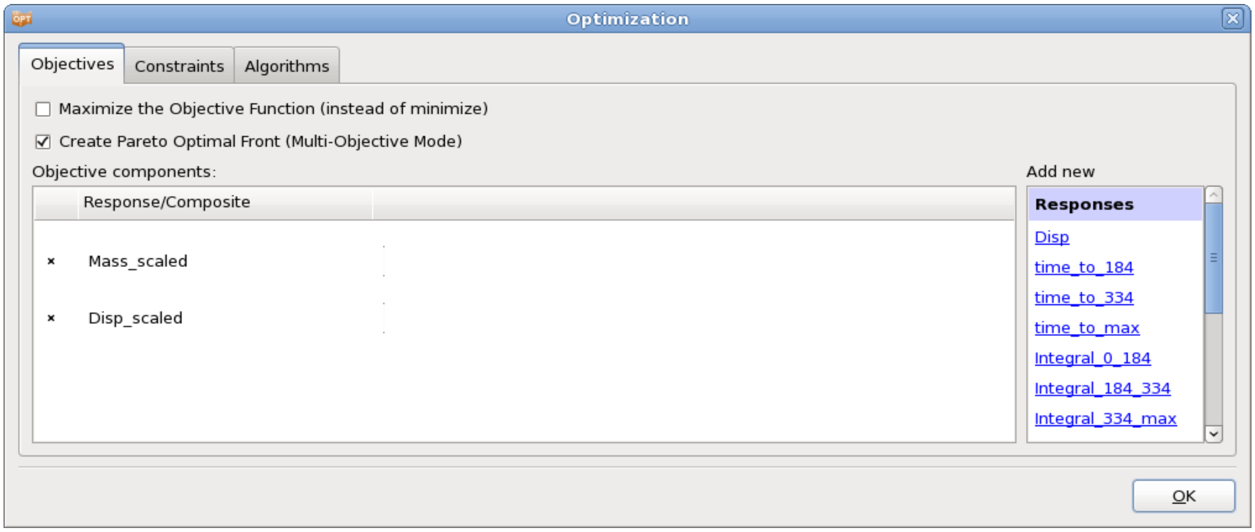
Mode Tracking Status

● On

○ Off

Figure 20-84: Frequency extraction with Mode Tracking

Define the objective and constraint functions in the Optimization dialog. For the objectives, make sure that the multi-objective mode is selected, Figure 20-85.



☐ Maximize the Objective Function (instead of minimize)

☑ Create Pareto Optimal Front (Multi-Objective Mode)

Objective components:

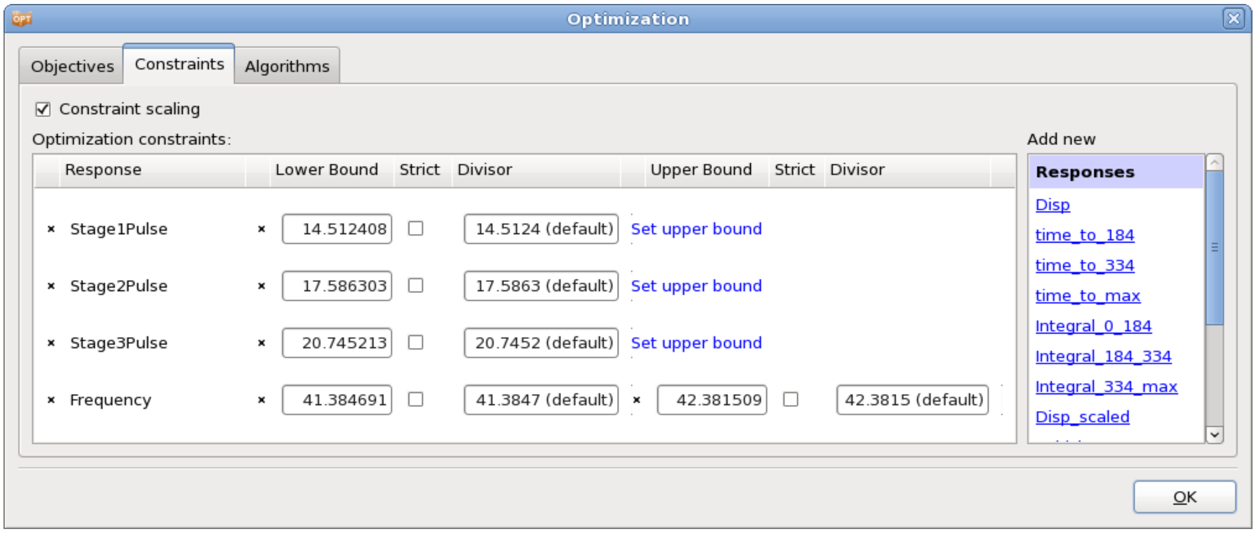
	Response/Composite
×	Mass_scaled
×	Disp_scaled

Add new

Responses
Disp
time_to_184
time_to_334
time_to_max
Integral_0_184
Integral_184_334
Integral_334_max

Figure 20-85: Objectives panel; Select Multi-Objective Mode to create Pareto Optimal Front

The constraints are scaled using the target values to balance the violations of the different constraints, Figure 20-86. This scaling is activated using a single check box and is only important in cases where multiple constraints are violated as in the current problem. However, it is a good idea to apply scaling of constraints as a rule.



☑ Constraint scaling

Optimization constraints:

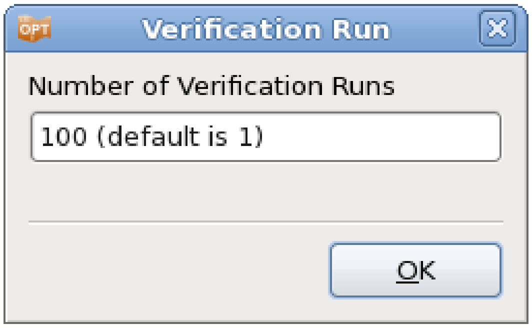
	Response		Lower Bound	Strict	Divisor		Upper Bound	Strict	Divisor
×	Stage1Pulse	×	14.512408	□	14.5124 (default)		Set upper bound		
×	Stage2Pulse	×	17.586303	□	17.5863 (default)		Set upper bound		
×	Stage3Pulse	×	20.745213	□	20.7452 (default)		Set upper bound		
×	Frequency	×	41.384691	□	41.3847 (default)	×	42.381509	□	42.3815 (default)

Add new

Responses
- Disp
- time_to_184
- time_to_334
- time_to_max
- Integral_0_184
- Integral_184_334
- Integral_334_max
- Disp_scaled

Figure 20-86: Constraints panel; Constraints are scaled using the target values. This is the default.

Since the Pareto Optimal solutions are calculated on the metamodel, 100 verification runs are executed after the last iteration to check the quality of the results, Figure 20-87.



Number of Verification Runs

100 (default is 1)

Figure 20-87: Verification Run; 100 verification runs are performed using results of the Pareto Optimal Front

Results

The LS-OPT viewer provides several tools to visualize Pareto Optimal solutions. Since this example has two objective functions, the Pareto optimal front obtained for the two cases can be displayed using the tradeoff plot, Figure 20-88. On the left, the Pareto Optimal solutions obtained from the metamodel are displayed while the plot on the right visualizes the verification runs. Some of the verification runs are infeasible due to the approximation error of the metamodel. Figure 20-89 shows the verification runs color-coded by the maximal constraint violation. For most of the simulations, the violation is almost 0, the highest constraint violation is 0.03, which is fairly small.

Figure 20-90 show the Self-Organizing maps plot (predicted) for the objective functions, the constraints and the variables. The conflict of the objectives is clearly visible (a blue cell in ‘Mass_scaled” corresponds to a red cell in “Disp_scaled” and *vice versa*). The corresponding ranges and influences of the variables can also be examined.

Figure 20-91 displays the Parallel Coordinate Plot of the predicted Pareto optimal solutions and the verification runs. This plot is useful to select a run from the various Pareto optimal solutions that best fits the requirements of the application. Using sliders located at the top and bottom of each vertical axis, the bounds of the constraints and the ranges of all entities can be interactively modified to narrow down the set of suitable solutions.

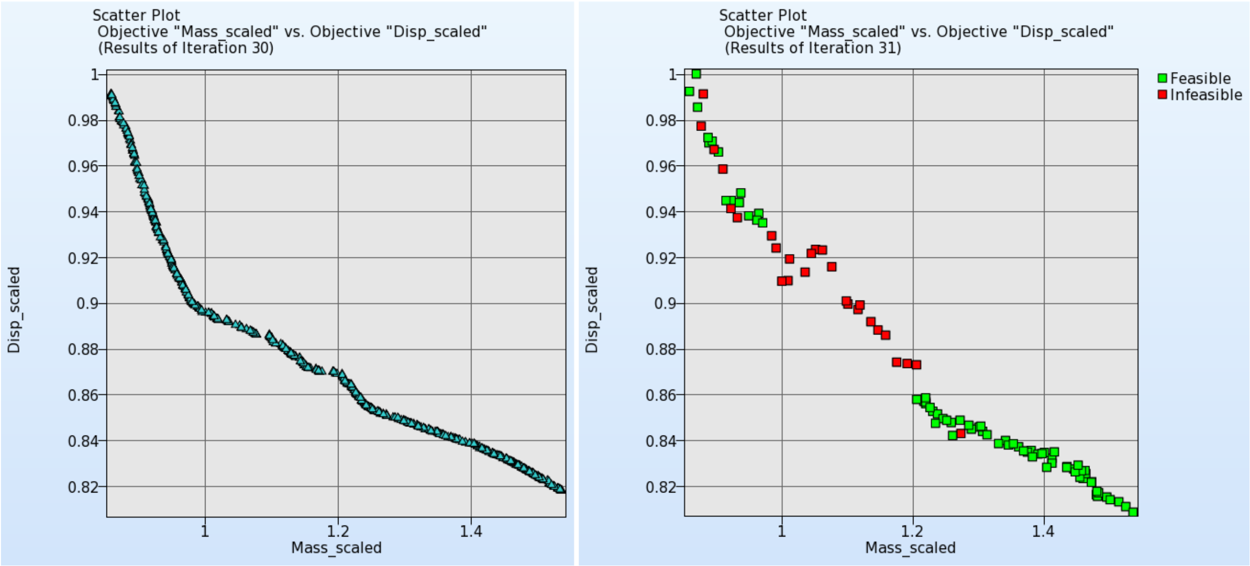


Figure 20-88: Pareto optimal front. Comparison of predicted results (left) and verification runs (right)

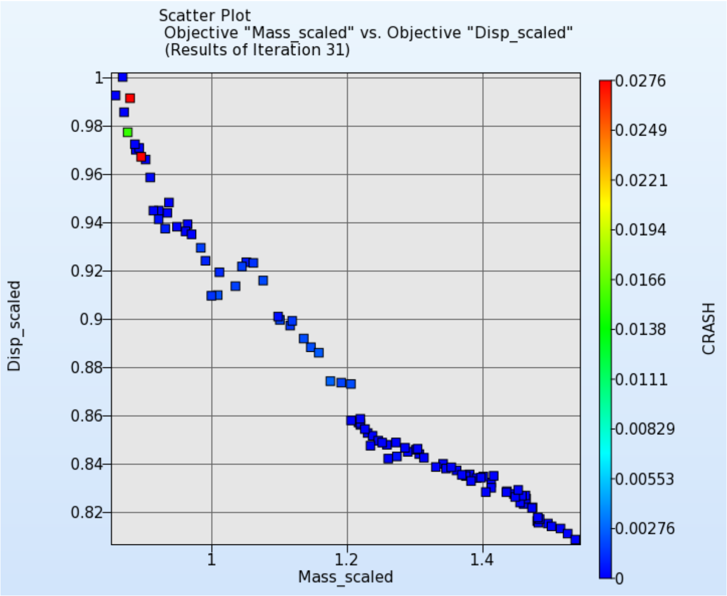


Figure 20-89: Verification runs color-coded by maximal constraint violation.

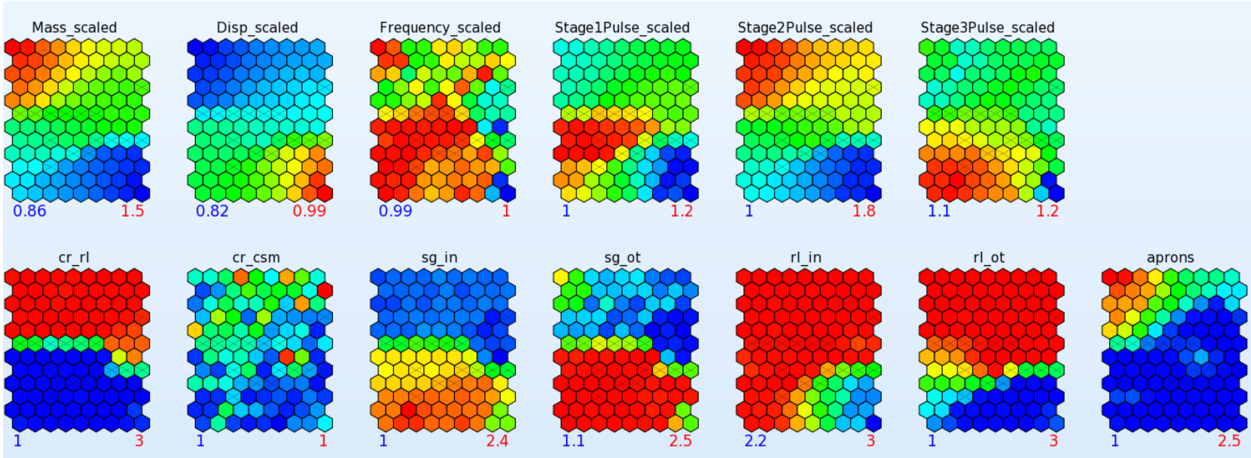


Figure 20-90: Self Organizing Maps plot of predicted Pareto optimal solutions

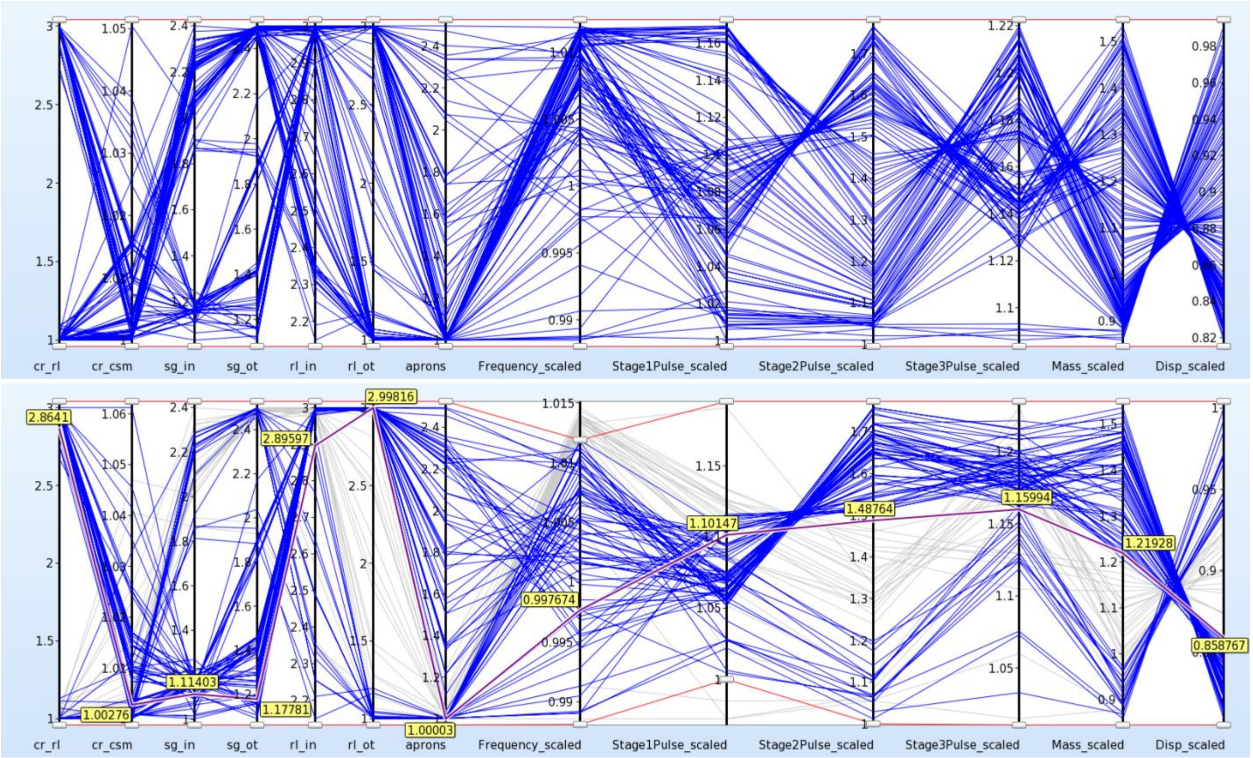
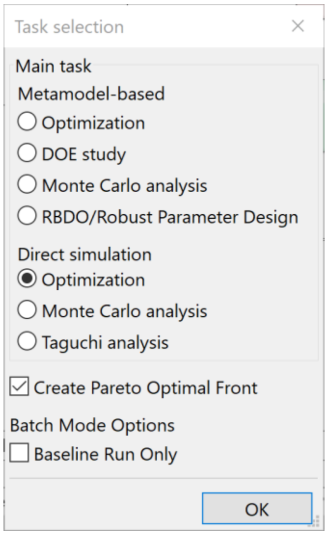


Figure 20-91: Parallel Coordinate Plot; Predicted Pareto Optimal solutions (top) and verification runs (bottom)

20.5.4. Multi-objective optimization using Direct GA simulation

Next, the problem is solved using direct GA simulations, Figure 20-92. The GA options used are displayed in Figure 20-93. The NSGA-II algorithm (*MOEA*) was used. Tournament selection operator (*Selection Operator*), with a tournament size of four (*Tournament Size*), was used to remove individuals with low fitness values. The simulated binary crossover (*Crossover Type*) and mutation operators were used to create child populations. The trade-off files were generated at each generation (*Restart Interval*).



Task selection

Main task

Metamodel-based

○ Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

● Optimization

○ Monte Carlo analysis

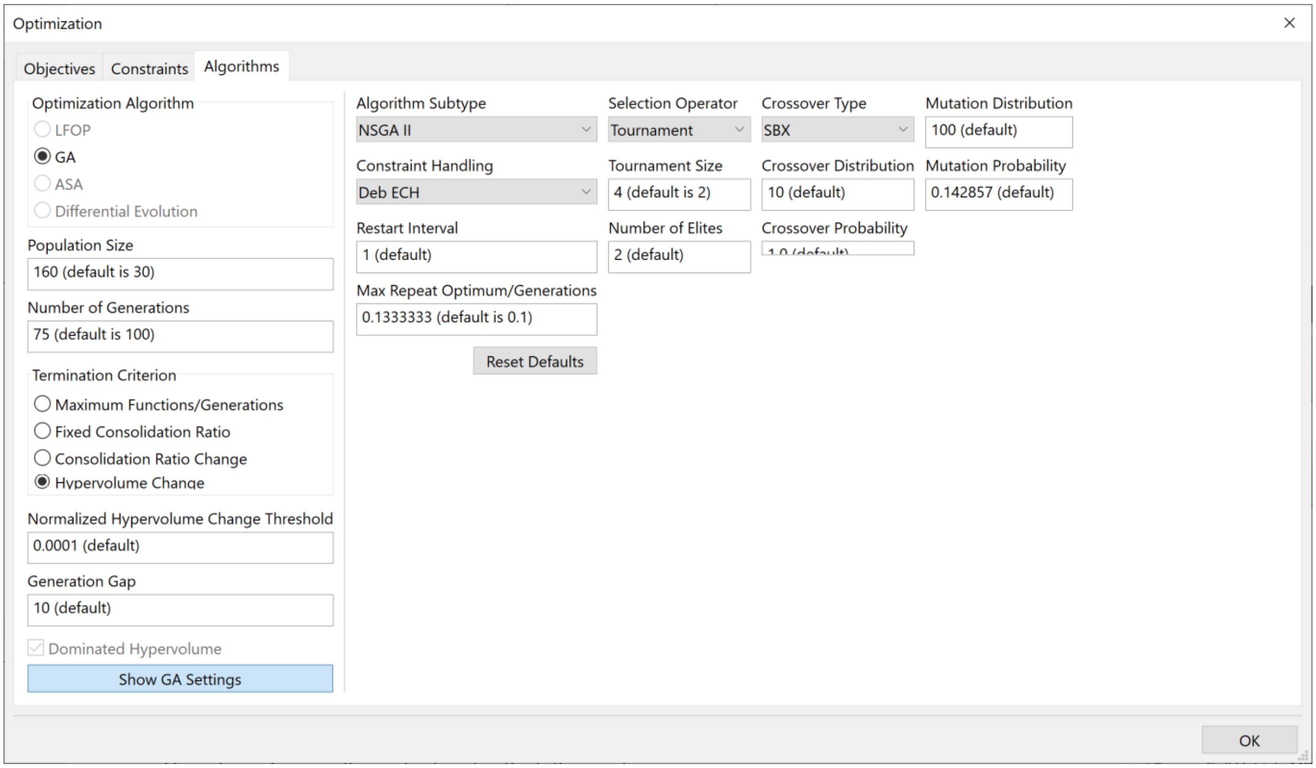
○ Taguchi analysis

☑ Create Pareto Optimal Front

Batch Mode Options

□ Baseline Run Only

Figure 20-92: Task dialog; Direct genetic algorithm



Optimization

Optimization Algorithm

○ LFOP

● GA

○ ASA

○ Differential Evolution

Population Size: 160 (default is 30)

Number of Generations: 75 (default is 100)

Termination Criterion

○ Maximum Functions/Generations

○ Fixed Consolidation Ratio

○ Consolidation Ratio Change

● Hypervolume Change

Normalized Hypervolume Change Threshold: 0.0001 (default)

Generation Gap: 10 (default)

☑ Dominated Hypervolume

Algorithm Subtype: NSGA II

Constraint Handling: Deb ECH

Restart Interval: 1 (default)

Max Repeat Optimum/Generations: 0.1333333 (default is 0.1)

Selection Operator: Tournament

Tournament Size: 4 (default is 2)

Number of Elites: 2 (default)

Crossover Type: SBX

Crossover Distribution: 10 (default)

Crossover Probability: 1.0 (default)

Mutation Distribution: 100 (default)

Mutation Probability: 0.142857 (default)

Figure 20-93: Options for Genetic Algorithm

Results

The optimization results are displayed in the following figures.

Since this example has two objective functions, the Pareto optimal front obtained for the two cases can be displayed using the tradeoff plot, Figure 20-94.

Figure 20-95 shows the Self-Organizing maps plot for the objective functions, the constraints and the variables. As in the metamodel-based optimization, the conflict of the objectives is again clearly visible while the ranges and influences of the variables can be examined.

Figure 20-96 displays the Parallel Coordinate Plot of the Pareto optimal solutions. This plot is useful to select a run out of the various Pareto optimal solutions that best fits the requirements of the application. As in the metamodel-based optimization, sliders located at the top and bottom of each vertical axis can be interactively adjusted to modify the bounds of the constraints and the ranges of all entities. This allows the user to narrow down the set of suitable solutions.

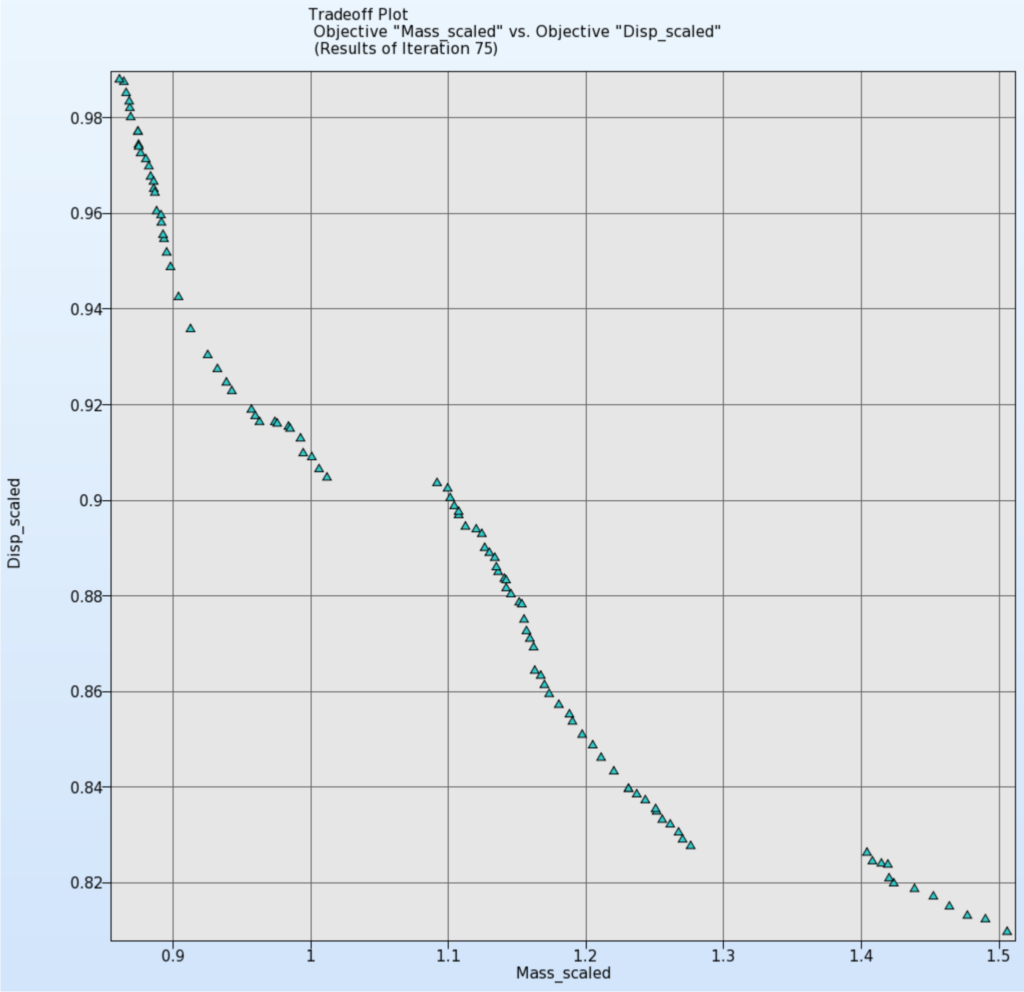


Figure 20-94: Tradeoffs between scaled mass and intrusion (displacement).

Trade-off between the two objectives shows that intrusion can be reduced by increasing the mass. The trade-off curve clearly illustrates that reduction in intrusion (from 0.81 to 0.988) will require a corresponding increase in mass (from 0.861 to 1.506). The ranges of the optimal design variables corresponding to the candidate Pareto optimal front are given in Table 20-7.

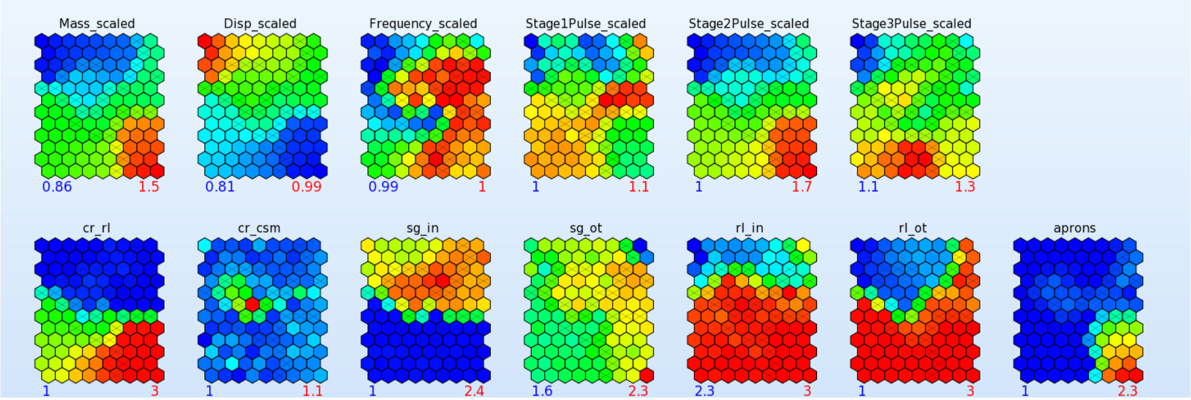


Figure 20-95: Self-organizing maps plot of Pareto optimal solutions; results of last generation

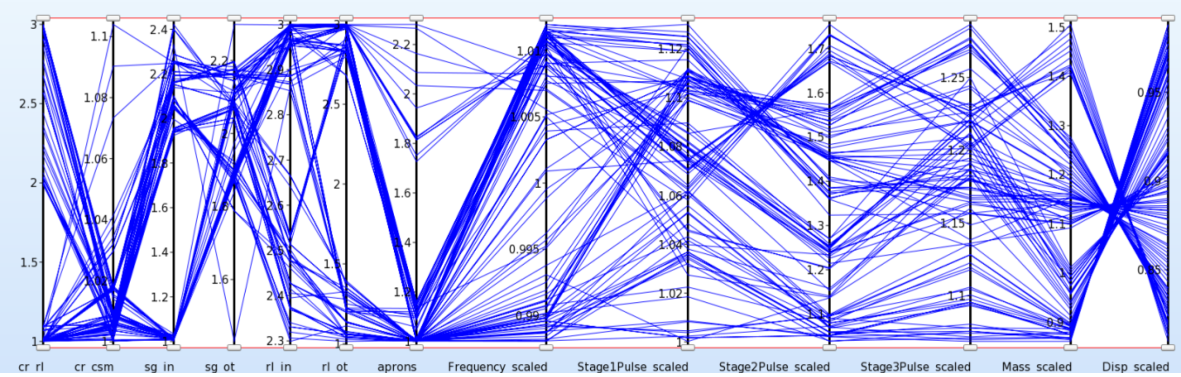


Figure 20-96: Parallel coordinate plot of Pareto optimal solutions; results of last generation

Table 20-7: Ranges of design variables in the final optimal solution set.

Variable	Lower	Upper
Rail inner	2.27	3.01
Rail outer	0.97	3.04
Aprons	0.97	2.32
Shotgun inner	0.97	2.47
Shotgun outer	1.44	2.40
Cradle cross member	1.00	1.09
Cradle rails	0.96	3.04

20.6. Knee impact with variable screening (11 variables)

(Example by courtesy of Visteon and Ford Motor Company)

This example has the following features:

- ANSA[16] is used as independent parametric preprocessor for shape parameterization
- A sensitivity analysis is performed to obtain a reduced set of variables for the optimization
- The minimum of two maxima is obtained in the objective (multi-criteria or multi-objective problem).

20.6.1. FE modeling

Figure 20-97 shows the finite element model of a typical automotive instrument panel (IP) [4]. For model simplification and reduced per-iteration computational times, only the driver's side of the IP is used in the analysis, and consists of around 25,000 shell elements. Symmetry boundary conditions are assumed at the centerline, and to simulate a bench component "Bendix" test, body attachments are assumed fixed in all 6 directions. Also shown in Figure 20-97 are simplified knee forms which move in a direction as determined from prior physical tests. As shown in the figure, this system is composed of a knee bolster (steel, plastic or both) that also serves as a steering column cover with a styled surface, and two energy absorption (EA) brackets (usually steel) attached to the cross vehicle IP structure. The brackets absorb a significant portion of the lower torso energy of the occupant by deforming appropriately. Sometimes, a steering column isolator (also known as a yoke) may be used as part of the knee bolster system to delay the wrap-around of the knees around the steering column. The last three components are non-visible and hence their shape can be optimized. The 11 design variables are shown in Figure 20-98.

The simulation is carried out for a 40 ms duration by which time the knees have been brought to rest. It may be mentioned here that the Bendix component test is used mainly for knee bolster system development; for certification purposes, a different physical test representative of the full vehicle is performed. Since the simulation used herein is at a subsystem level, the results reported here may be used mainly for illustration purposes.

[16] BETA CAE Systems S.A.

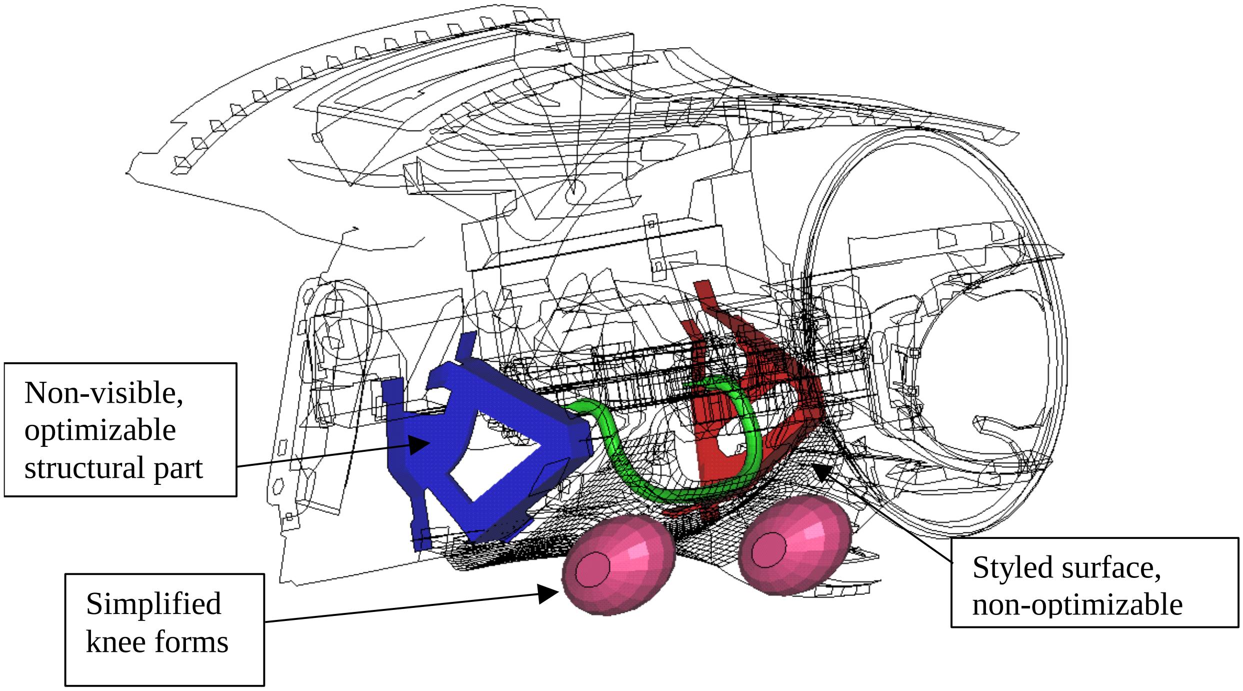


Figure 20-97: Typical instrument panel prepared for a "Bendix" component test

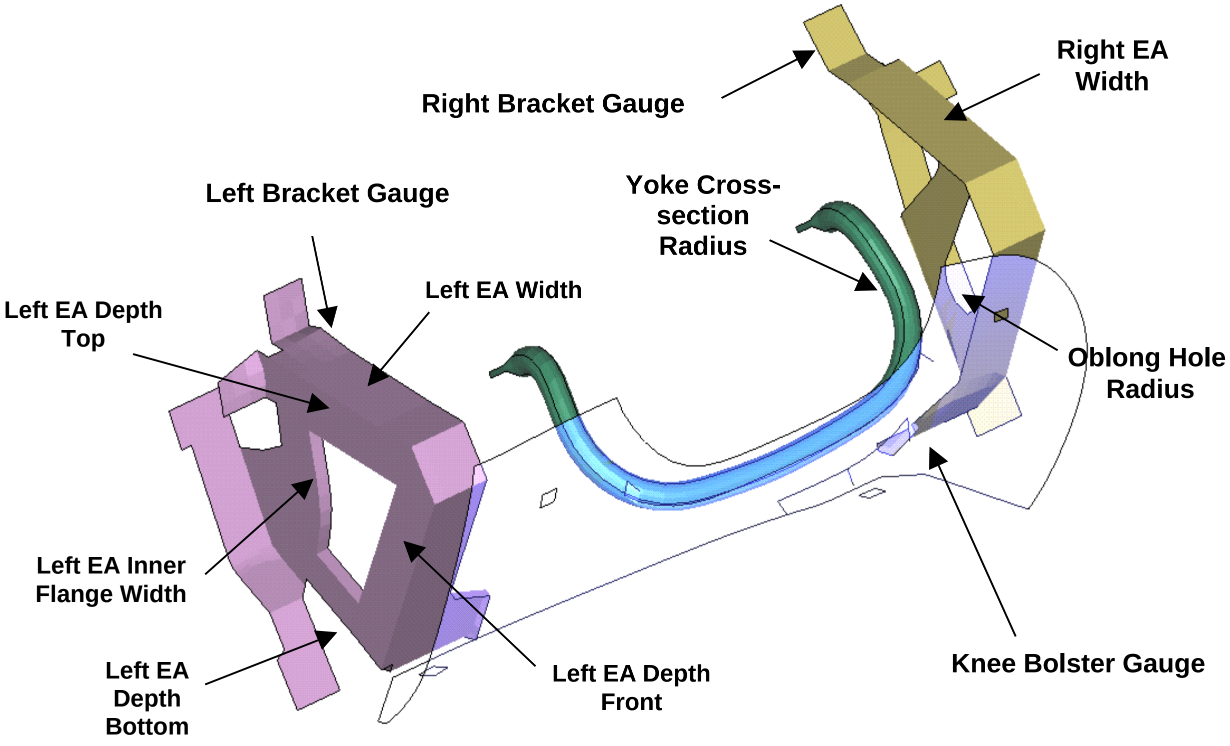


Figure 20-98: Typical major components of a knee bolster system and definition of design variables.

20.6.2. Design formulation

The optimization problem is defined as follows:

Minimize (max (Knee_Force_Left, Knee_Force_Right))

subject to

Left Knee intrusion	<	115mm
Right Knee intrusion	<	115mm
Yoke displacement	<	85mm
Kinetic Energy	<	1.54e5

The knee forces have been filtered, SAE 60 Hz, to improve the approximation accuracy.

20.6.3. Solution

ANSA is used to parameterize the geometry, Figure 20-99 and Figure 20-100. Since the ANSA output file is used as in include file in the LS-DYNA input, a file transfer has to be defined to copy the file to the respective LS-DYNA run directory, Figure 20-101. Alternatively, the option “*Run jobs in directory of stage*“ could be used. The maximal knee force to be used as objective function is defined as a composite expression, Figure 20-102. The definition of the constraints is displayed in Figure 20-103.

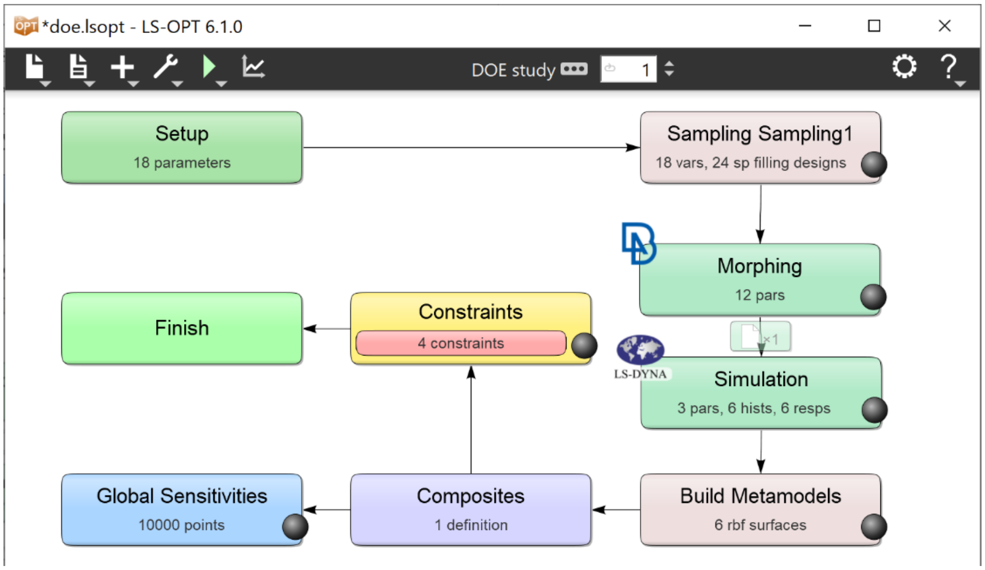
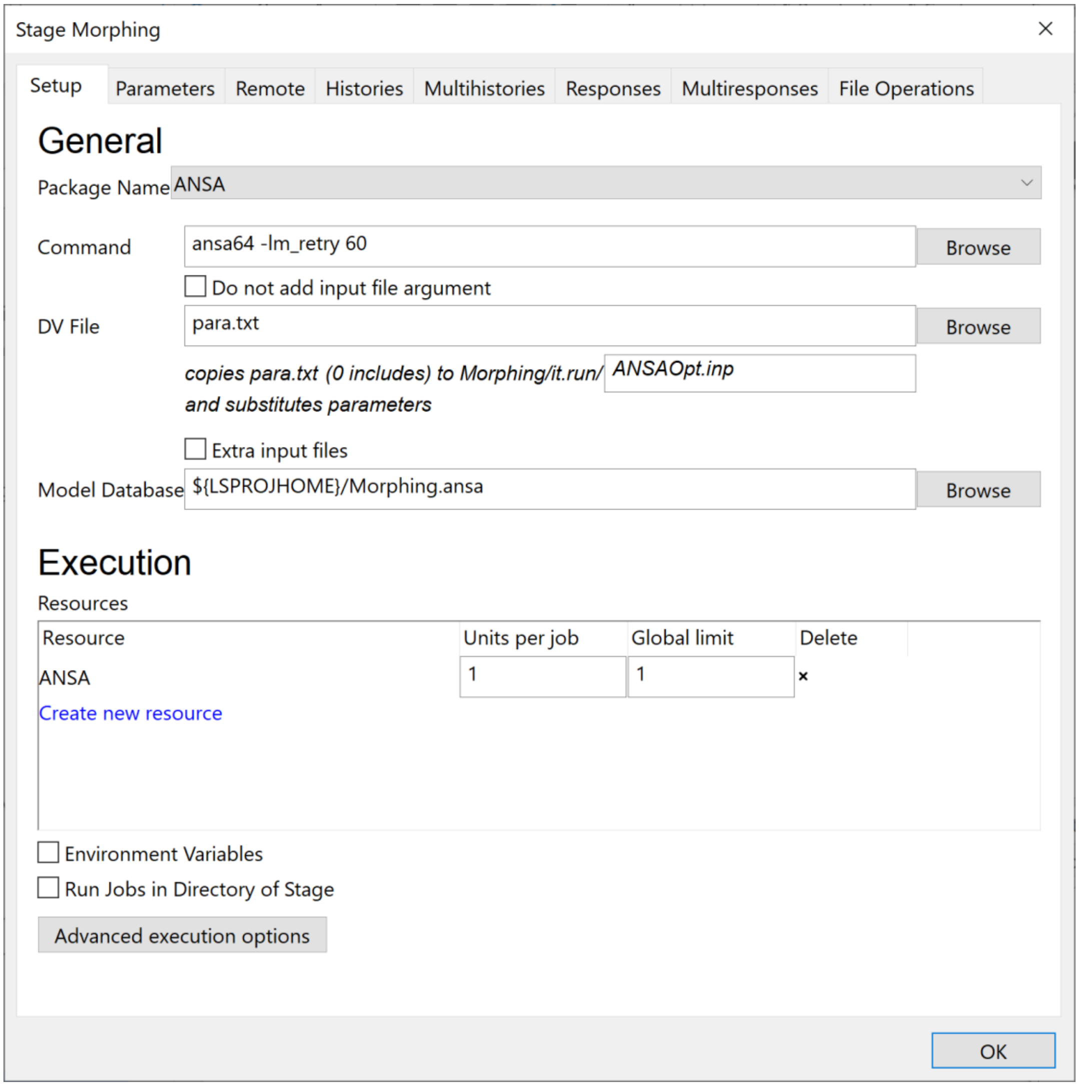


Figure 20-99: Main LS-OPT GUI window. ANSA is used as a preprocessor in LS-OPT to incorporate shape optimization



General

Package Name ANSA

Command ansa64 -lm_retry 60

□ Do not add input file argument

DV File para.txt

copies para.txt (0 includes) to Morphing/it.run/ *ANSAOpt.inp*
and substitutes parameters

□ Extra input files

Model Database ${LSPROJHOME}/Morphing.ansa

Execution

Resources

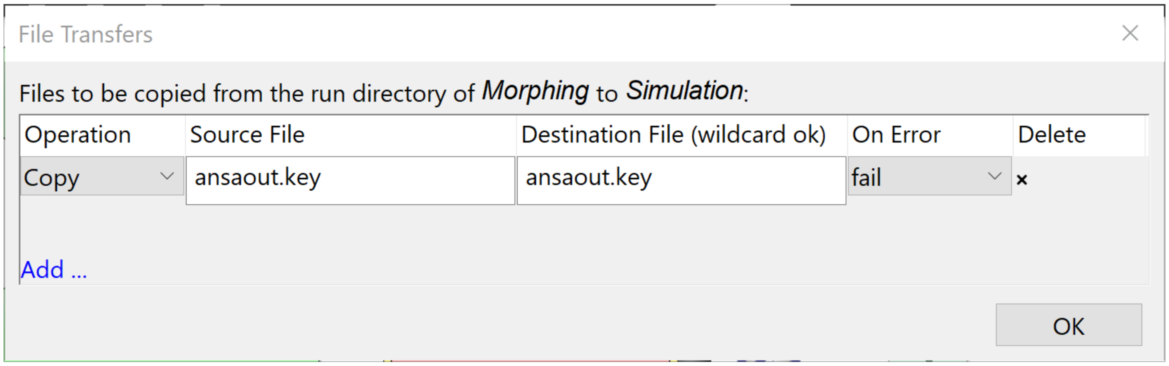
Resource	Units per job	Global limit	Delete
ANSA	1	1	×
Create new resource			

□ Environment Variables

□ Run Jobs in Directory of Stage

Advanced execution options

Figure 20-100: ANSA interface; definition of ANSA command, design variables file and ANSA database.

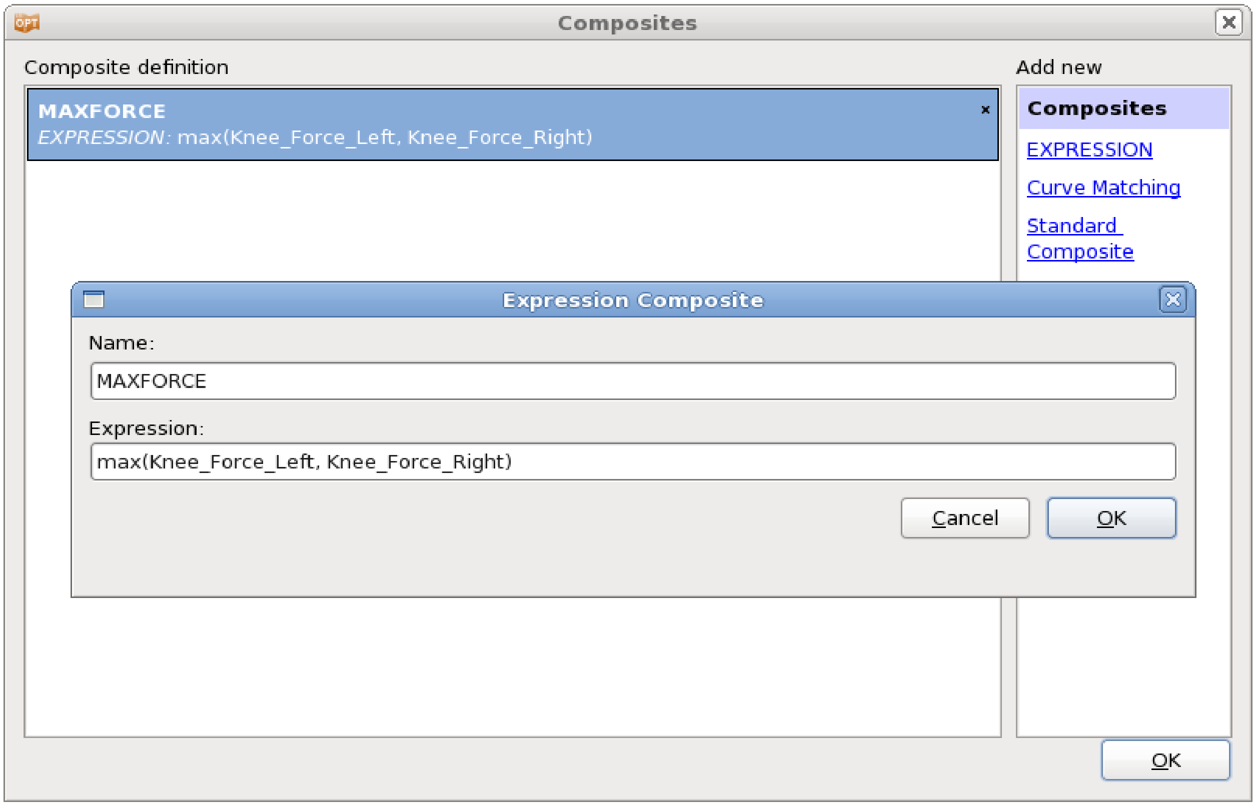


Files to be copied from the run directory of *Morphing* to *Simulation*:

Operation	Source File	Destination File (wildcard ok)	On Error	Delete
Copy	ansaout.key	ansaout.key	fail	×

Add ...

Figure 20-101: File transfer; the ANSA output is used as include file in the LS-DYNA input file.



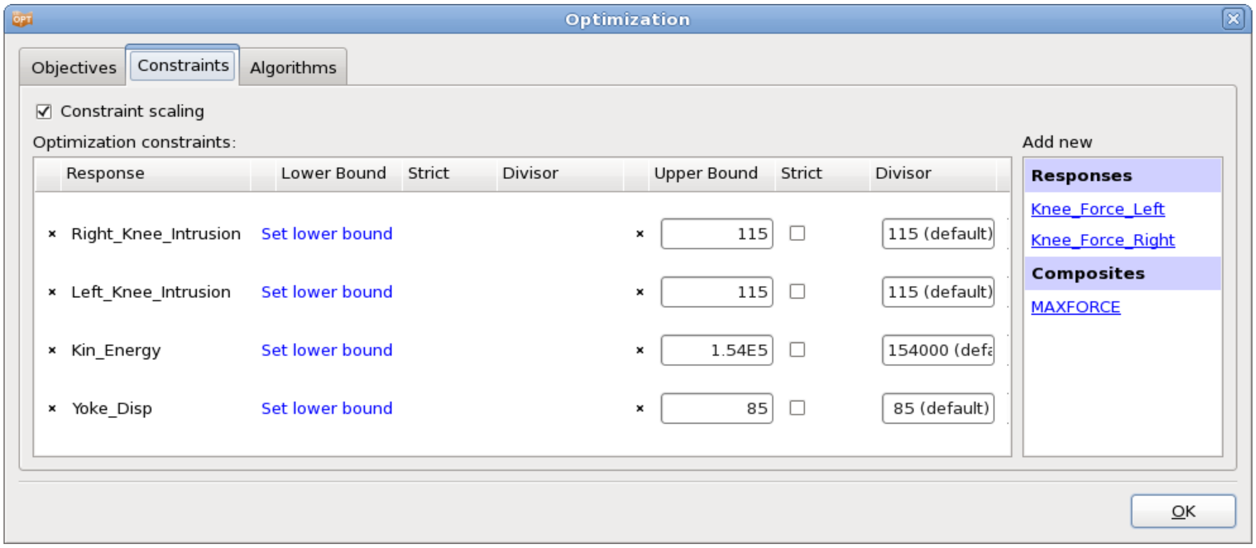
Composite definition

MAXFORCE
EXPRESSION: max(Knee_Force_Left, Knee_Force_Right)

Name:
MAXFORCE

Expression:
max(Knee_Force_Left, Knee_Force_Right)

Figure 20-102: Definition of maximal knee force as composite expression.



☑ Constraint scaling

Optimization constraints:

Response	Lower Bound	Strict	Divisor	Upper Bound	Strict	Divisor
× Right_Knee_Intrusion	Set lower bound			× 115	□	115 (default)
× Left_Knee_Intrusion	Set lower bound			× 115	□	115 (default)
× Kin_Energy	Set lower bound			× 1.54E5	□	154000 (defa
× Yoke_Disp	Set lower bound			× 85	□	85 (default)

Figure 20-103: Constraints for the knee bolster design problem.

20.6.4. Variable screening

First a DOE is done with a linear approximation to find the most sensitive parameters. The ANOVA and Sobol's global sensitivity analysis charts may be used to evaluate the results, Figure 20-104 and Figure

20-105. Note the large confidence intervals (low confidence levels) on some of the responses, especially the Kin_Energy, the Knee_Force_Left and Yoke_Disp.

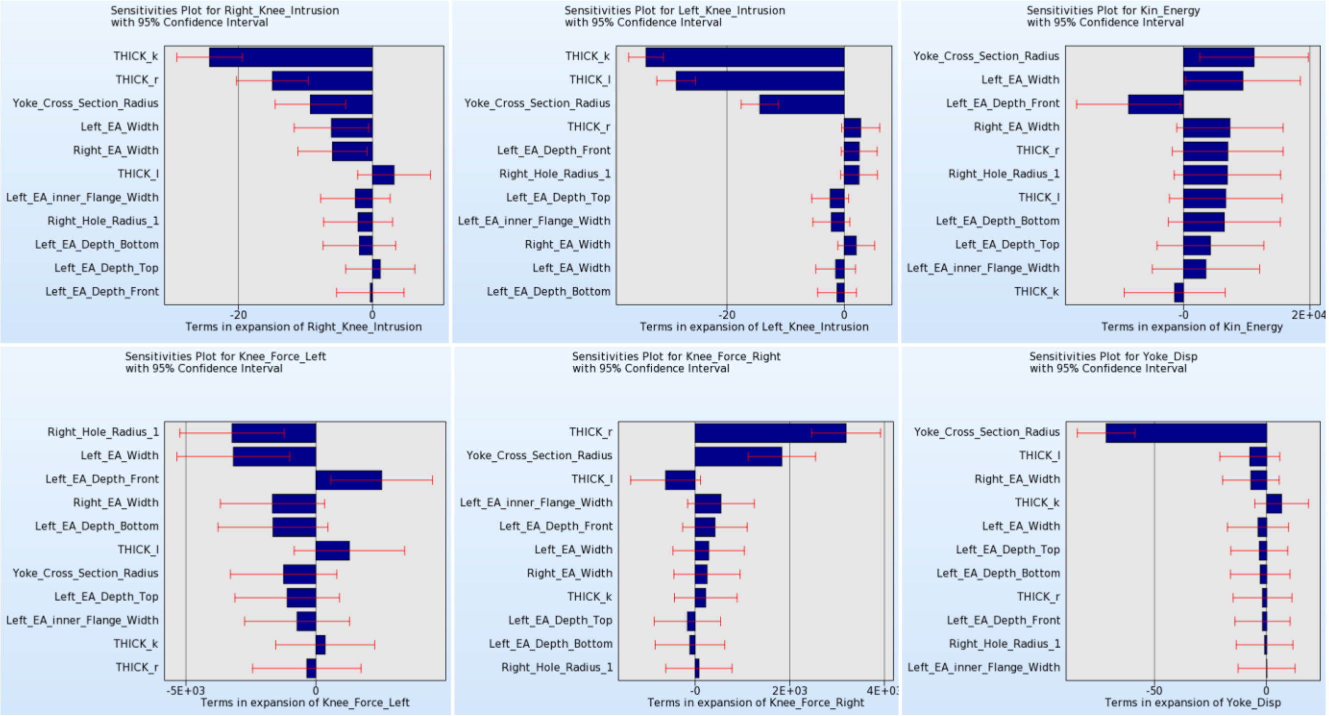


Figure 20-104: ANOVA plots for objectives and constraints of knee-bolster design problem.

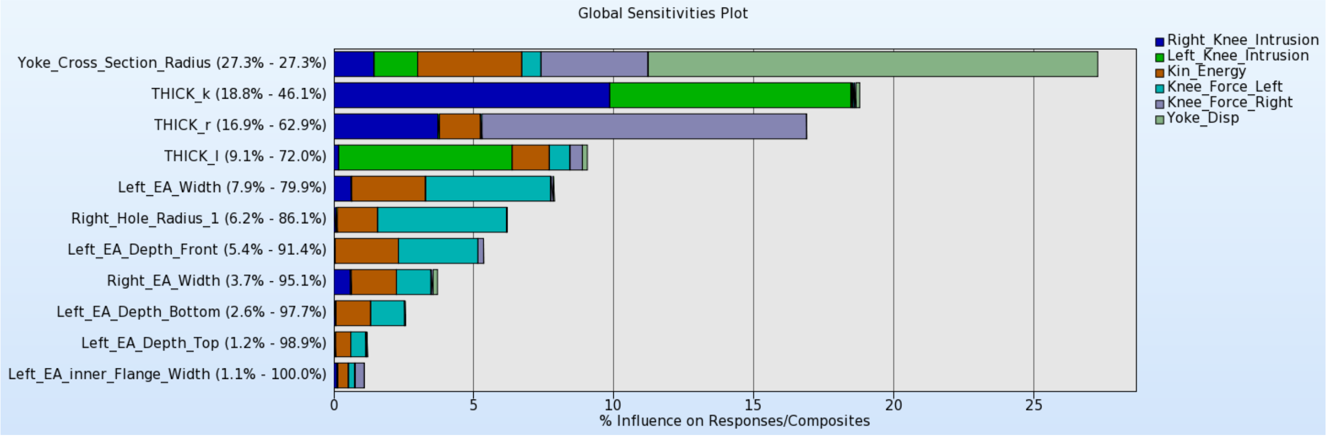


Figure 20-105: Global sensitivity analysis of objectives and constraints.

The six most sensitive variables chosen from the charts are:

x=[*Yoke_Cross_Scetion_Radiu, THICK_k, THICK_r, THICK_l, Left_EA_Width, Right_Hole_Radius_1*]^T;

Those variables are used to perform an optimization.

20.6.5. Optimization

After reducing the set of parameters considering the results of the previously performed DOE, a metamodel-based optimization is performed using the strategy *sequential with domain reduction* and linear metamodels. The reader is also referred to [5] for a discussion of the accuracy and purpose of the various sequential sampling strategies available in LS-OPT.

The plots below (Figure 20-106) show the optimization histories, i.e. the deployment of the optimal values over the iterations, of the objectives and the maximal constraint violation. While the baseline design resulted in a maximum force of 16551.7 and a maximal constraint violation of 41.7, the optimum design resulted in a maximum force of only 6720.7. Though intermediate computed results were infeasible because of the approximation error of the metamodels, the final design was feasible.

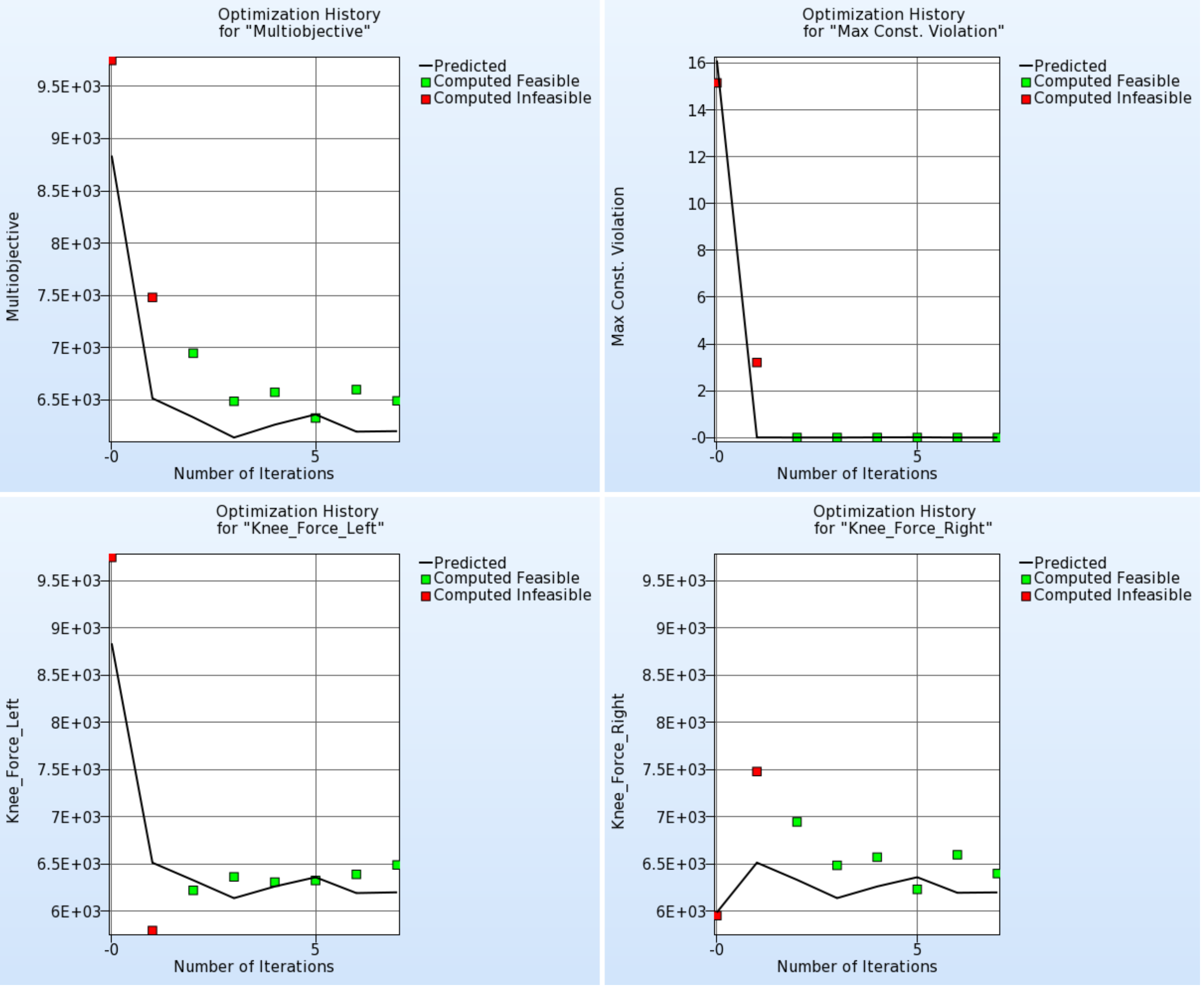


Figure 20-106: Optimization history of objectives and maximum constraint violations.

20.7. Shape optimization of a front rail using ANSA[17] and μETA[18]

This example has the following features:

- Optimization of a 3-stage process chain using ANSA, LS-DYNA and μETA
- Shape optimization using the ANSA morphing tool
- Discrete variables
- Result extraction using μETA

20.7.1. Problem Statement

The problem is of a front rail crash simulation. Embosses are to be used to

Minimize acceleration

subject to

mass < 1.8

intrusion < 300.

The design variables are the depth and width of the embosses, the distance between the embosses, and the thickness of the rail, Figure 20-107. Thickness and width are defined as discrete parameters.

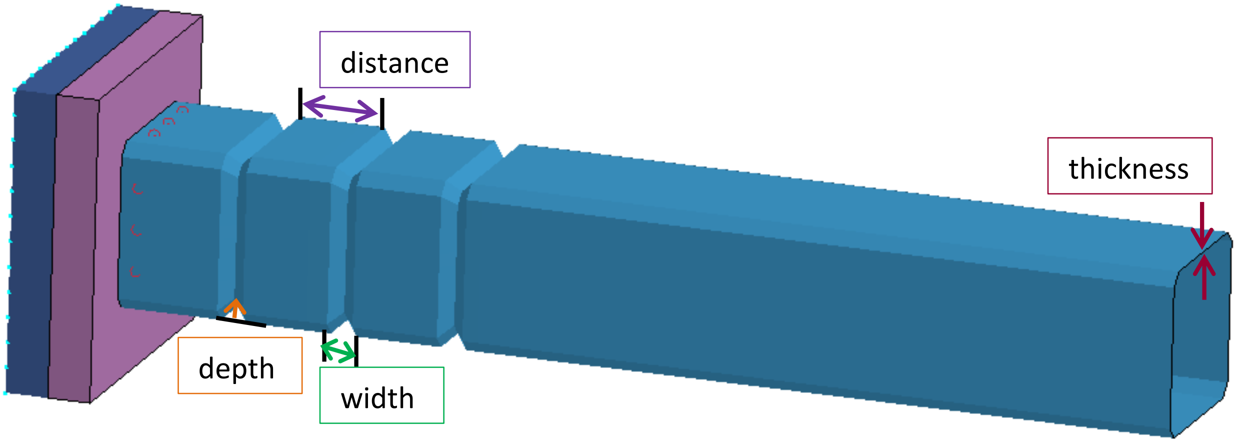


Figure 20-107: Rail with embosses.

20.7.2. Solution

The morphing tool of the preprocessor ANSA is used to incorporate the geometrical parameters. The thickness parameter is also defined in ANSA. The ANSA database including morphing boxes, morphing parameters,

[17] BETA CAE Systems S.A.
[18] BETA CAE Systems S.A.

and the optimization task is provided in `rail_task.ansa`. See [7] for the setup of the morphing boxes and the optimization task. Make sure that the `Optimization task` is set to `Execution` mode in ANSA before running the optimization.

A metamodel based optimization with strategy sequential with domain reduction is used. Figure 20-108 displays the main LS-OPT GUI window visualizing the optimization process and the ANSA – LS-DYNA – μETA process chain.

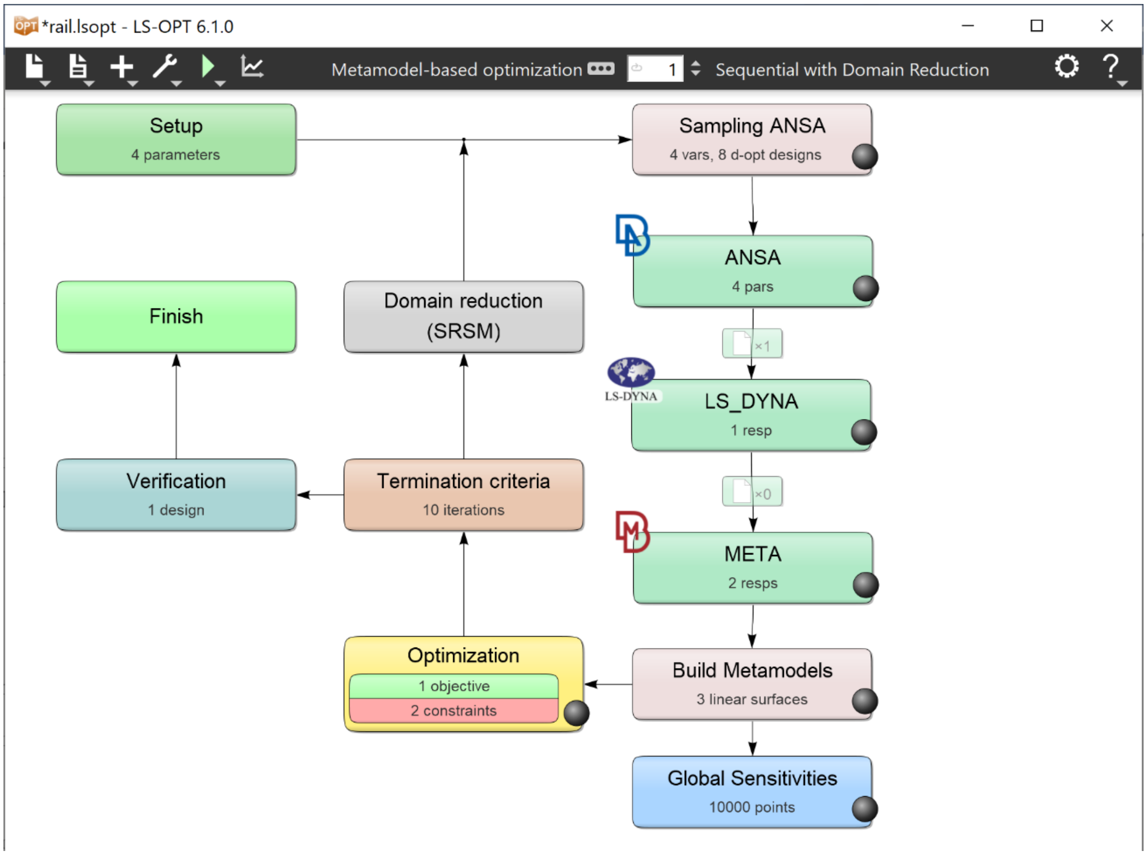
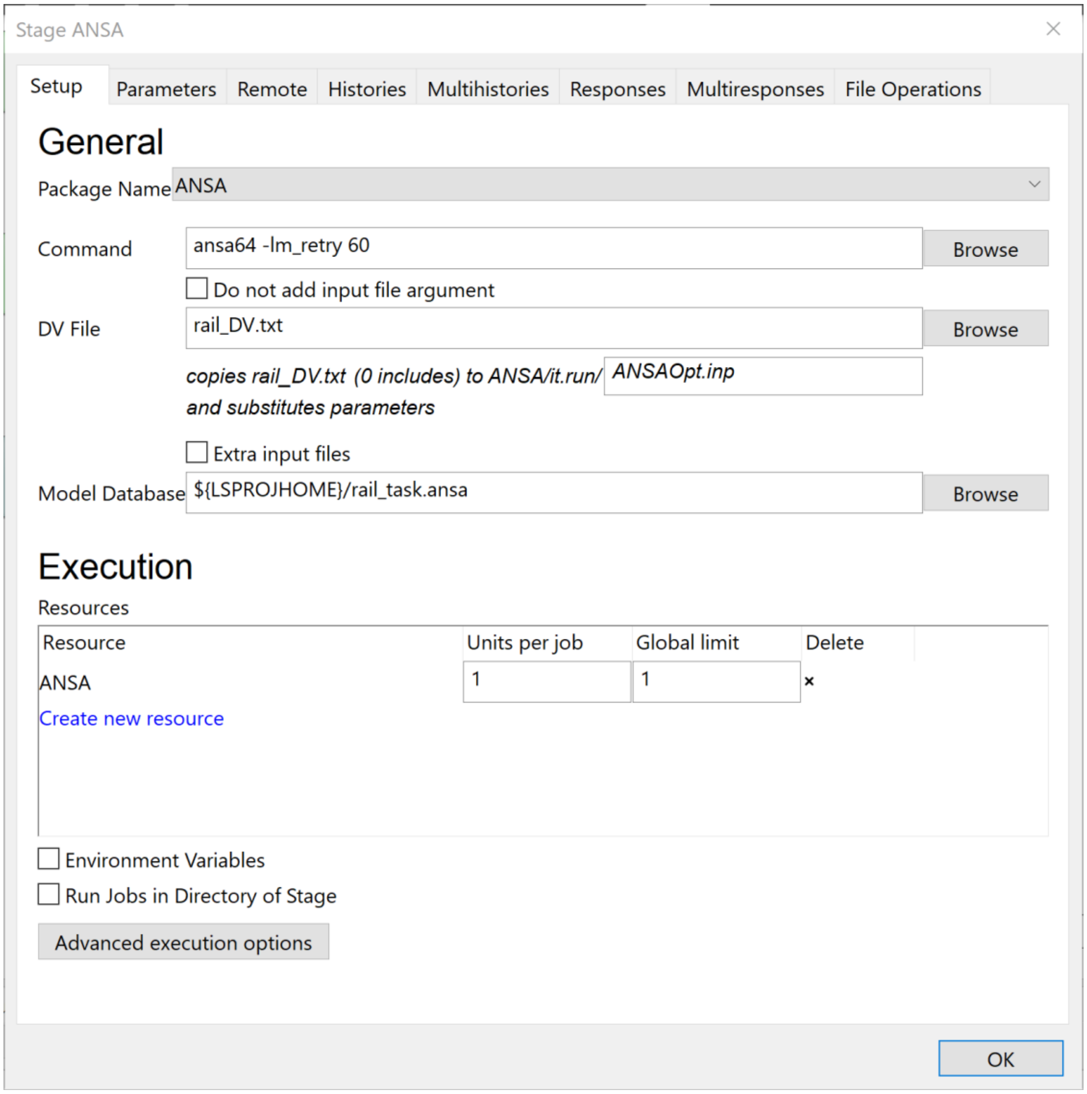


Figure 20-108: Main LS-OPT GUI window; Metamodel based design optimization, Strategy SRSM; Optimization of a process chain with 3 stages.

Figure 20-109 shows the Setup of the ANSA stage. This stage has no responses or histories.

All parameters defined in `rail_DV.txt` are imported to LS-OPT, including type, values and ranges, Figure 20-110.

Since the ANSA output is used for the LS-DYNA run, the specification of a file transfer is needed to copy the output to the respective LS-DYNA run directory, Figure 20-111. Another possibility to make output files available for other stages is to run the jobs in the respective stage directory. This is done here for the META stage, Figure 20-112.



General

Package Name: ANSA

Command: ansa64 -lm_retry 60

□ Do not add input file argument

DV File: rail_DV.txt

copies rail_DV.txt (0 includes) to ANSA/it.run/ *ANSAOpt.inp*
and substitutes parameters

□ Extra input files

Model Database: ${LSPROJHOME}/rail_task.ansa

Execution

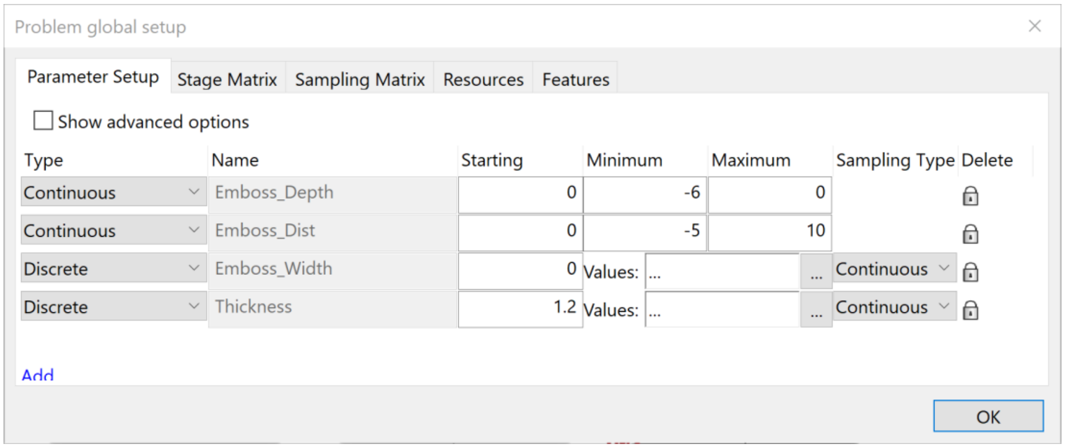
Resources

Resource	Units per job	Global limit	Delete
ANSA	1	1	×

□ Environment Variables

□ Run Jobs in Directory of Stage

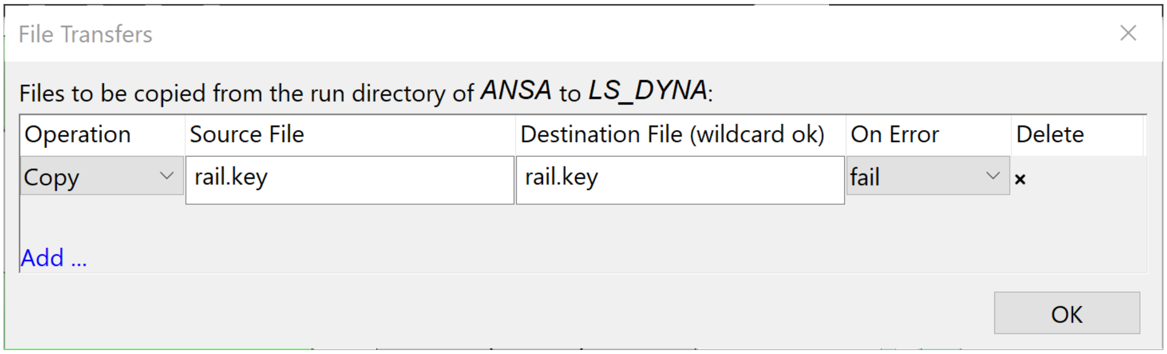
Figure 20-109: Stage dialog interfacing with ANSA



□ Show advanced options

Type	Name	Starting	Minimum	Maximum	Sampling Type	Delete
Continuous	Emboss_Depth	0	-6	0		🔒
Continuous	Emboss_Dist	0	-5	10		🔒
Discrete	Emboss_Width	0	Values: ...	...	Continuous	🔒
Discrete	Thickness	1.2	Values: ...	...	Continuous	🔒

Figure 20-110: Parameter Setup; variables, values and ranges are imported from the ANSA DV file

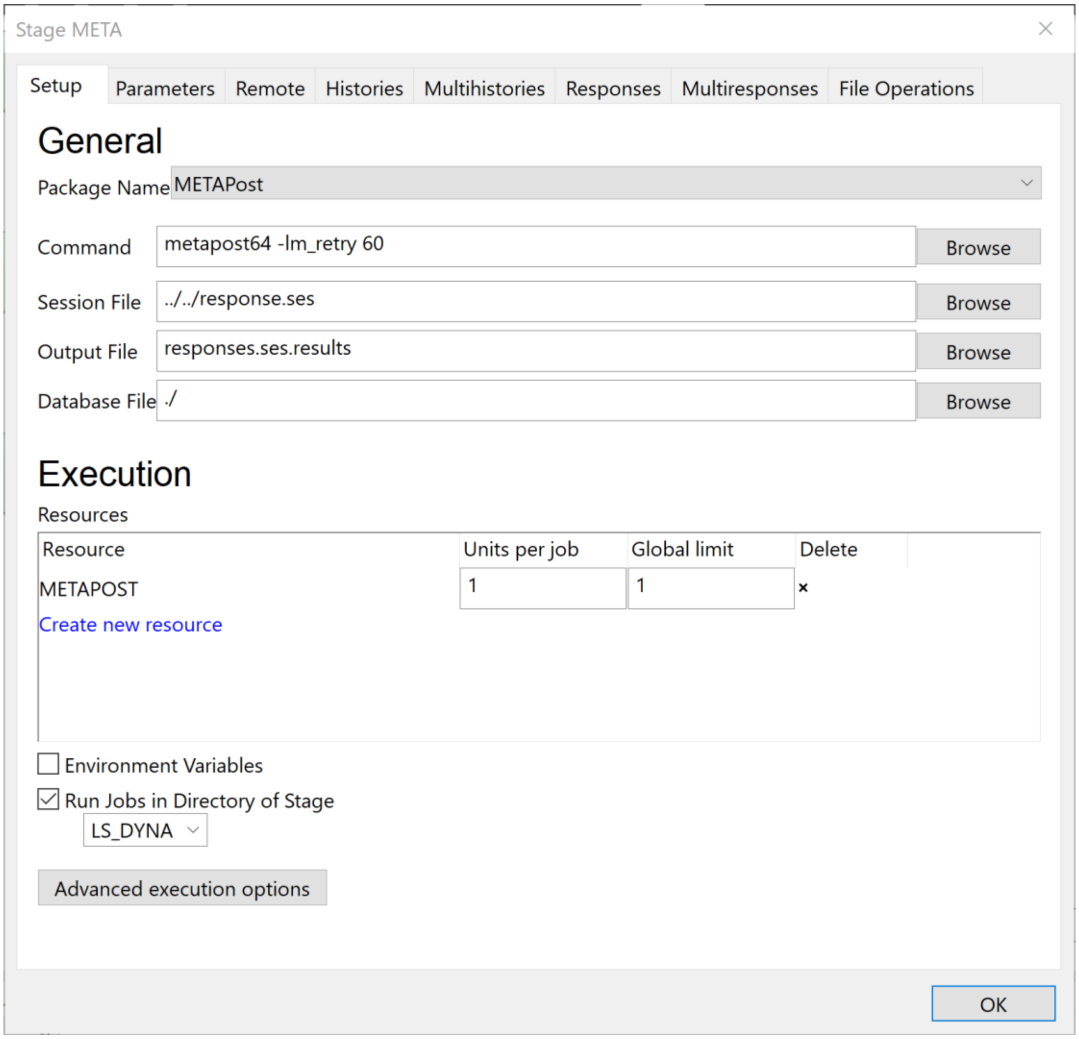


File Transfers

Files to be copied from the run directory of *ANSA* to *LS_DYNA*:

Operation	Source File	Destination File (wildcard ok)	On Error	Delete
Copy	rail.key	rail.key	fail	×

Figure 20-111: File Transfer dialog; the ANSA output file rail.key is copied to the respective LS-DYNA run directory.



Stage META

General

Package Name: METAPost

Command: metapost64 -lm_retry 60

Session File: ../../response.ses

Output File: responses.ses.results

Database File: ./

Execution

Resources

Resource	Units per job	Global limit	Delete
METAPOST	1	1	×

☐ Environment Variables

☑ Run Jobs in Directory of Stage

LS_DYNA

Figure 20-112: Stage dialog interfacing with µETApost. µETApost is run in the respective LS-DYNA run directory, since the results are extracted from the LS-DYNA output.

20.7.3. Results

Figure 20-113 and Figure 20-114 display the optimization history of the variables and the responses, respectively. There is no convergence for the variable `Emboss_Dist`, but as the Sensitivies Plot in Figure 20-115 shows, this variable is not sensitive. Due to metamodel inaccuracy, the final design is infeasible displayed in Figure 20-116, but e.g. the optimal value of iteration 8 is feasible, and the acceleration value is similar.

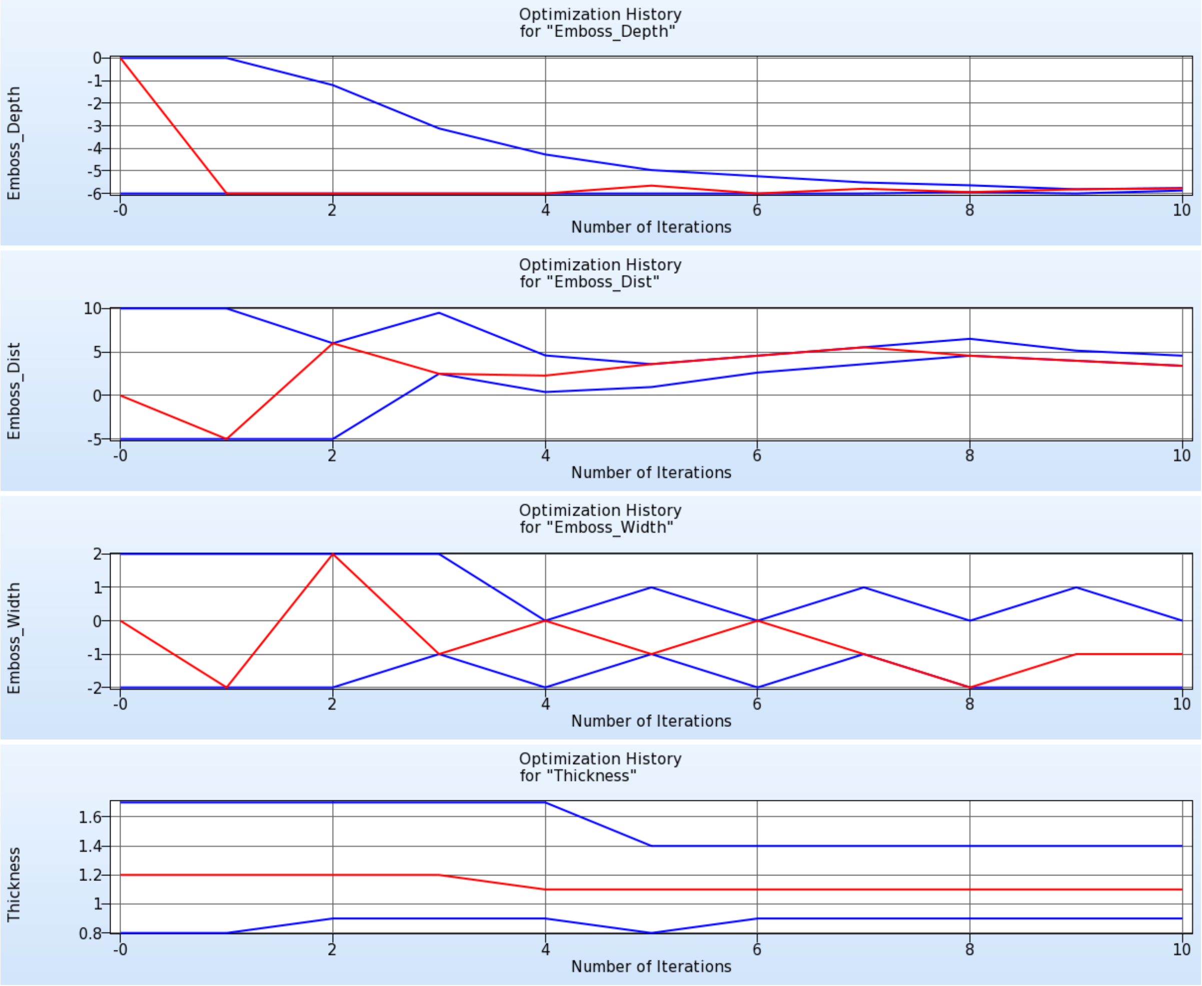


Figure 20-113: Optimization History of variables.

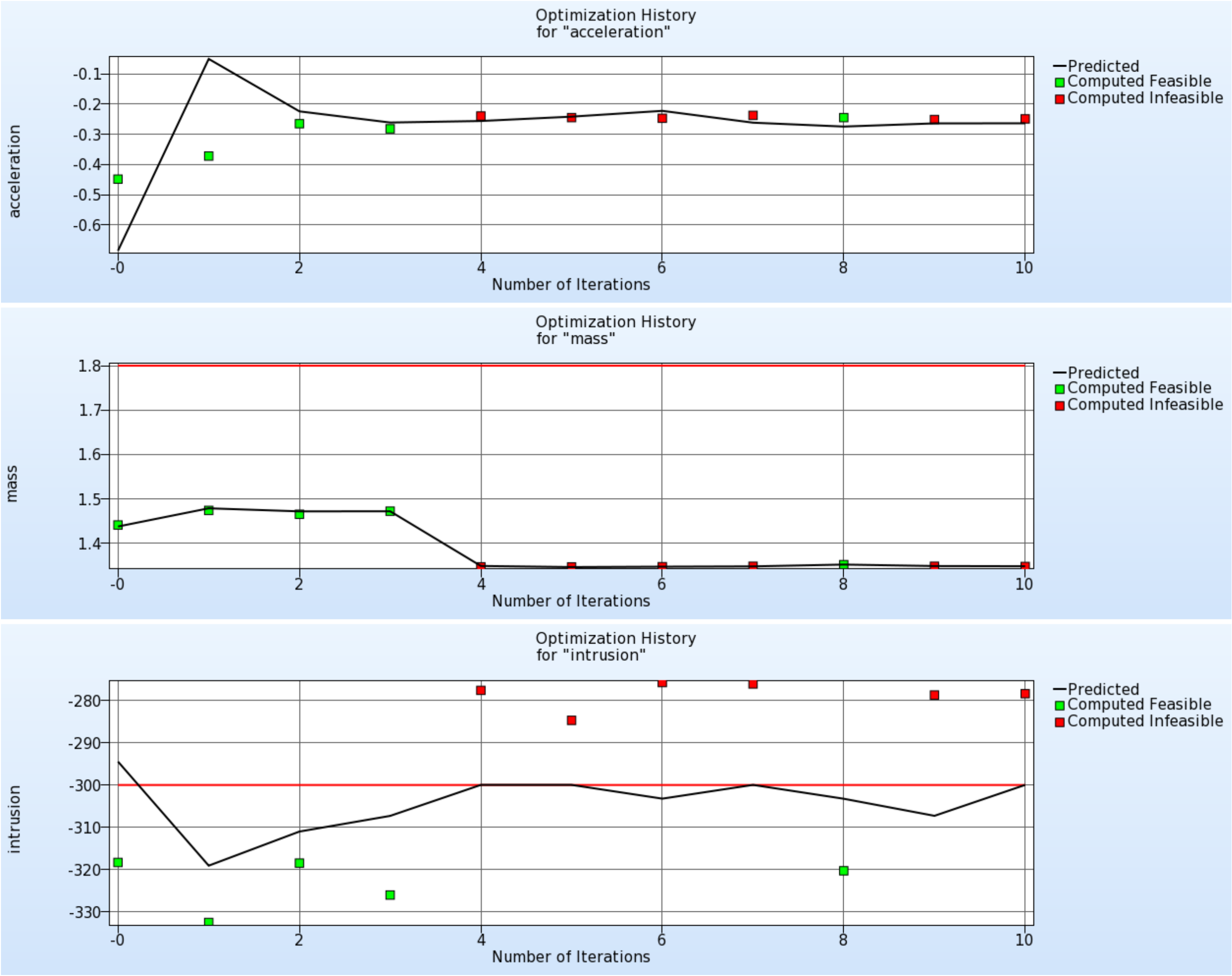


Figure 20-114: Optimization History of objective and constraints.

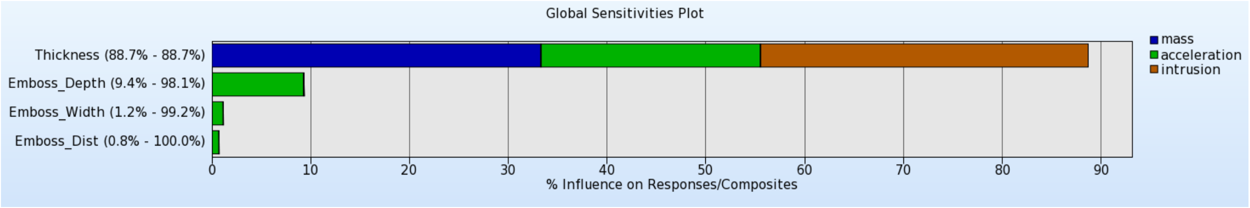


Figure 20-115: Global Sensitivities.

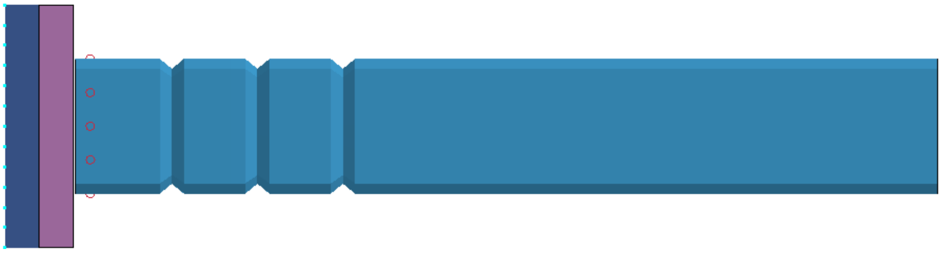


Figure 20-116: Final design. Optimum of iteration 10.

20.8. Optimization with analytical design sensitivities

This example has the following features:

- Using analytical design sensitivities for optimization
- Defining a user-defined solver

20.8.1. Problem Statement

The optimization problem to be solved is

$$\max(x_1^2 + 4(x_2 - 0.5)^2)$$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ -2x_1 + x_2 &\leq 2 \\ x_2 &\geq 0. \end{aligned}$$

Figure 20-117 displays the objective and constraint functions.

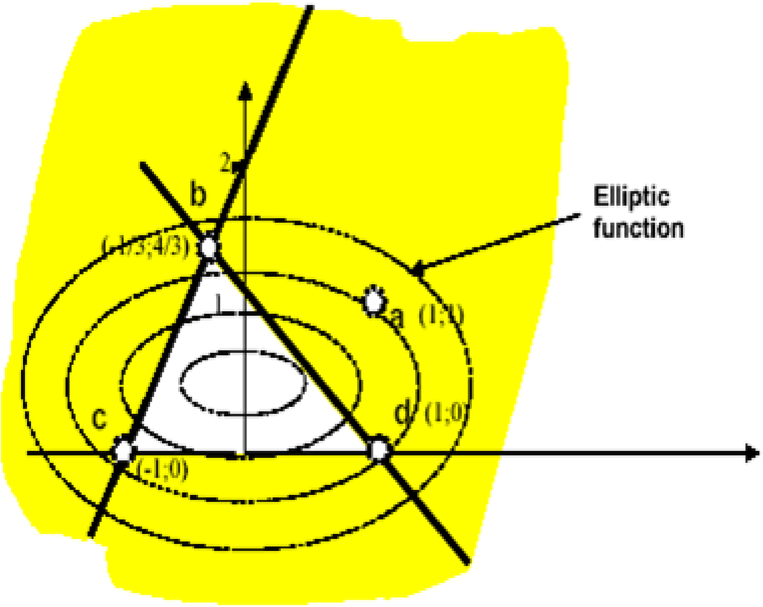


Figure 20-117: Objective and constraint functions.

This example demonstrates how analytical gradients (Section 9.1.2) provided by a solver can be used for optimization using the SLP algorithm and the domain reduction scheme [5] (Section **26.6**). The solver, a Perl program, is shown below. It calculates the analytical functions as well as the gradients at the respective simulation points.

In this example the input variables are read from the file: `XPoint` placed in the run directory by LS-OPT. The input variables can also be read by defining this file as an input file and using the <<*variable_name*>> format to label the variable locations for substitution. Note that each response requires a unique `Gradient` file.

Solver program:

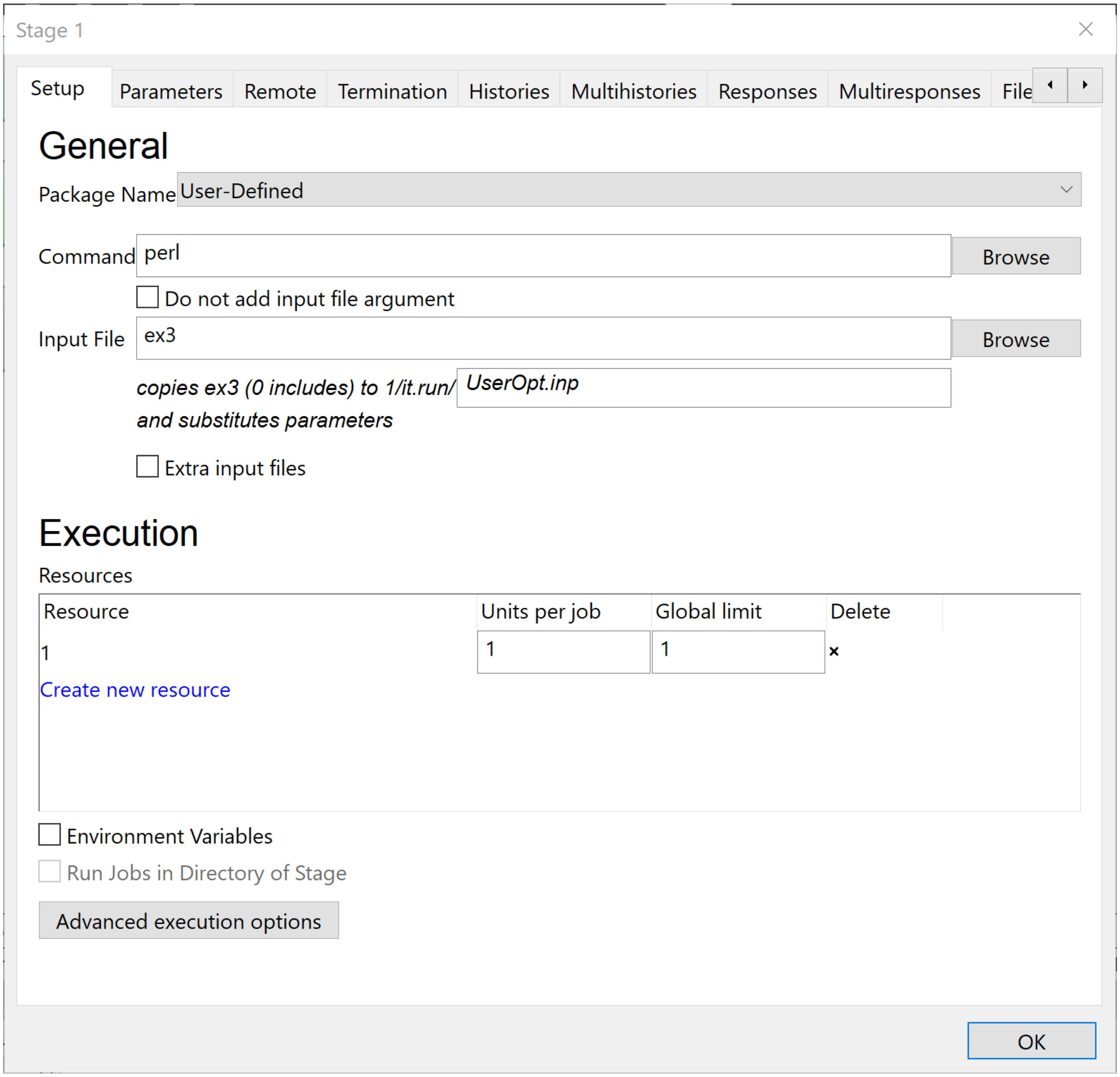
```
# Open output files for response results
#
open(FOUT,">fsol");
open(G1OUT,">g1sol");
open(G2OUT,">g2sol");
#
# Output files for gradients
#
open(DF,">Gradf");
open(DG1,">Gradg1");
open(DG2,">Gradg2");
#
# Open the input file "XPoint" (automatically
# placed by LS-OPT in the run directory)
#
open(X,"<XPoint");
#
# Compute results and write to the files
# (i.e. conduct the simulation)
#
while (<X>) {
   ($x1,$x2) = split;
}
#
print FOUT  ($x1*$x1) + (4*($x2-0.5)*($x2-0.5)),"\n";
# Derivative of f(x1,x2)
#-----------------------
print DF    (2*$x1)," ";          # df/dx1
print DF    (8*($x2-0.5)),"\n";   # df/dx2
#
print G1OUT $x1 + $x2,"\n";
# Derivative of g1(x1,x2)
#------------------------
print DG1 1," ";
print DG1 1,"\n";
#
print G2OUT (-2*$x1) + $x2,"\n";
# Derivative of g2(x1,x2)
#------------------------
print DG2 -2," ";
print DG2 1,"\n";
#
# Signal normal termination
print "N o r m a l\n";
```

20.8.2. Solution

Figure 20-118 shows the stage dialog defining the user-defined solver.

Figure 20-119 displays the response definitions. The gradient files generated by the Perl program need to be copied to a file called `Gradient`, the calculated response values need to be output to standard output.

Use the metamodel type *Sensitivity* to use analytical gradients for optimization, Figure 20-120.



Stage 1

General

Package Name User-Defined

Command perl

□ Do not add input file argument

Input File ex3

copies ex3 (0 includes) to 1/it.run/ *UserOpt.inp*
and substitutes parameters

□ Extra input files

Execution

Resources

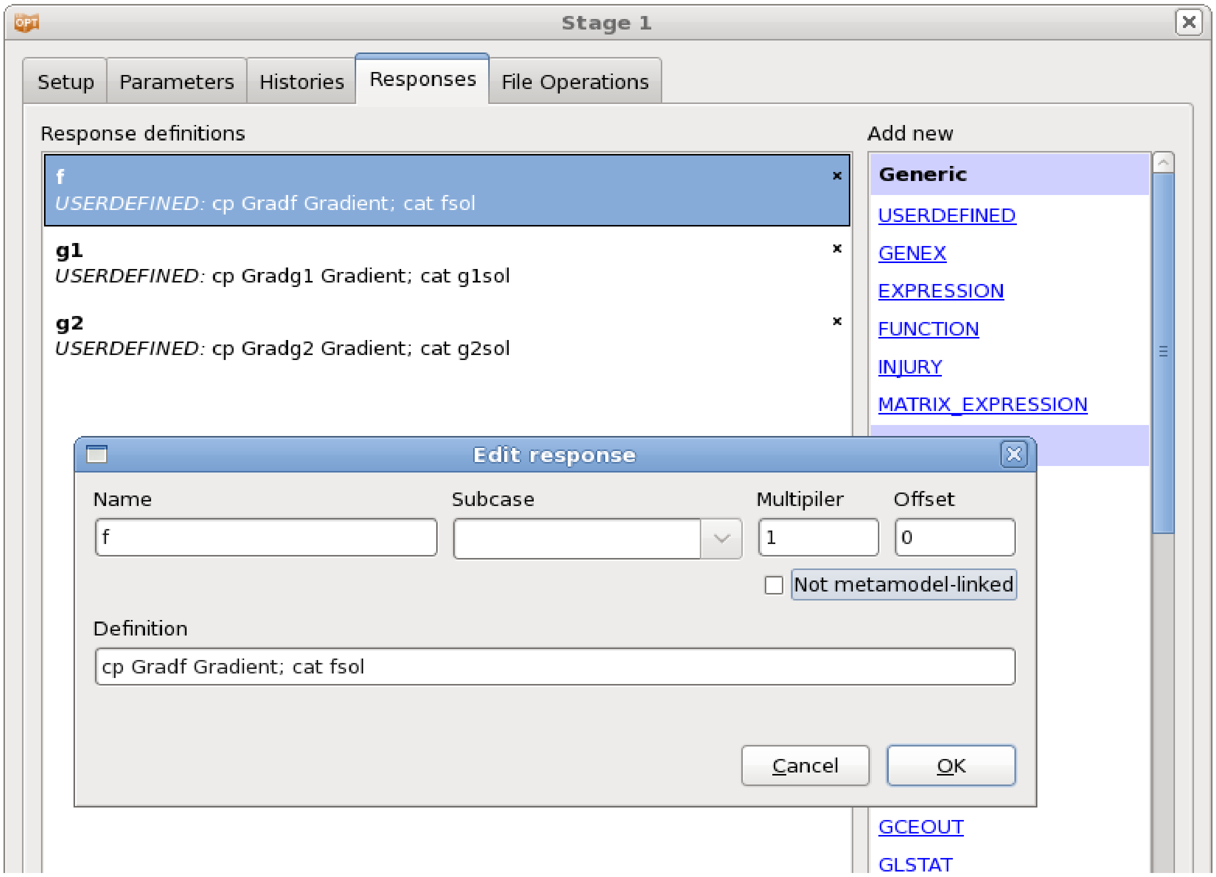
Resource	Units per job	Global limit	Delete
1	1	1	×

Create new resource

□ Environment Variables

□ Run Jobs in Directory of Stage

Figure 20-118: Defining a user-defined solver



Response definitions

f
USERDEFINED: cp Gradf Gradient; cat fsol

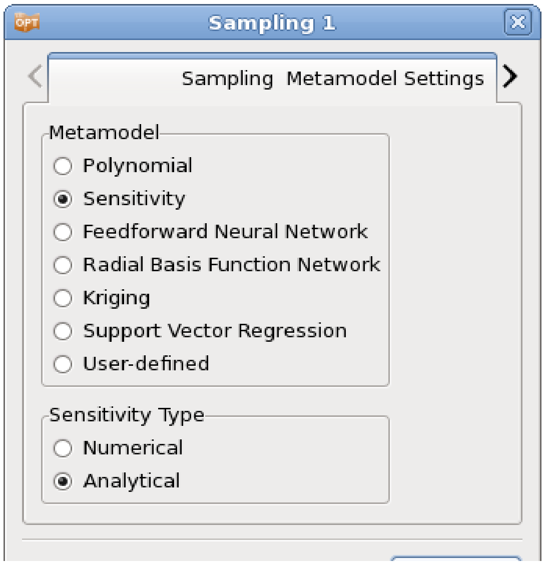
g1
USERDEFINED: cp Gradg1 Gradient; cat g1sol

g2
USERDEFINED: cp Gradg2 Gradient; cat g2sol

Name: f
Subcase:
Multipiler: 1
Offset: 0
□ Not metamodel-linked

Definition: cp Gradf Gradient; cat fsol

Figure 20-119: User-defined results extraction.



Metamodel
○ Polynomial
● Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression
○ User-defined

Sensitivity Type
○ Numerical
● Analytical

Figure 20-120: Sampling definition for optimization using analytical sensitivities

Typical "Gradient" file (e.g. for f):

```
1.8000000000 -3.20000000000
```

20.8.3. Results

The optimization results are shown in the plots below. An iteration represents a single simulation. The red dots represent the computed results while the solid line represents a linear approximation constructed from the gradient information of the previous point.

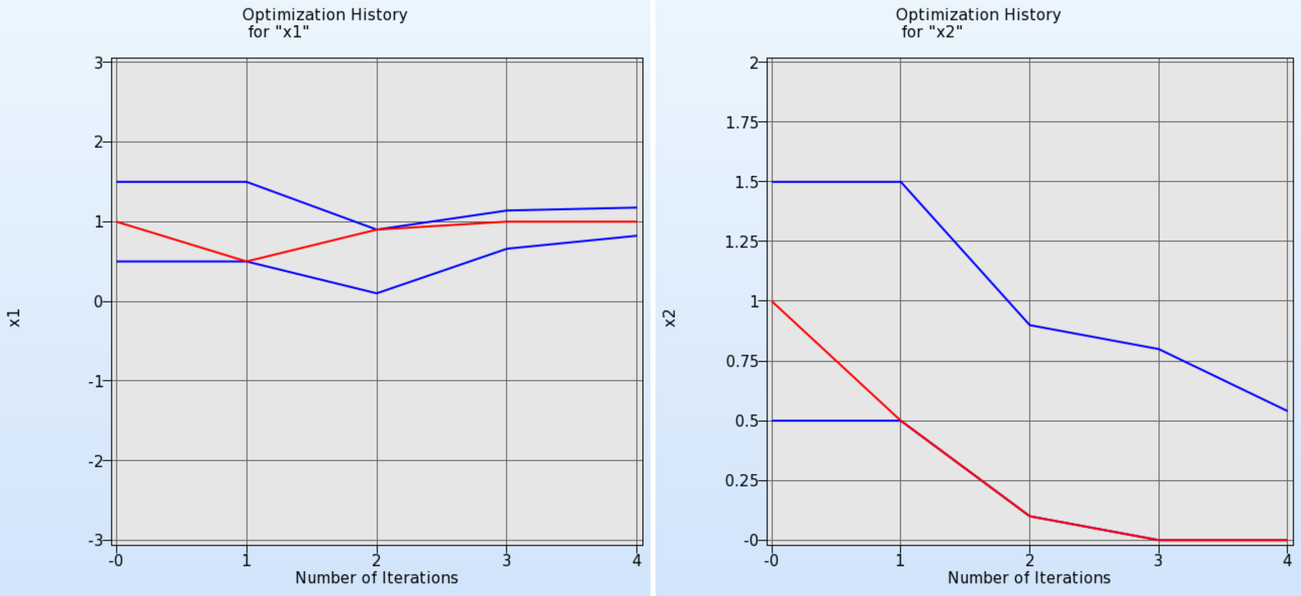


Figure 20-121: Optimization history for variables

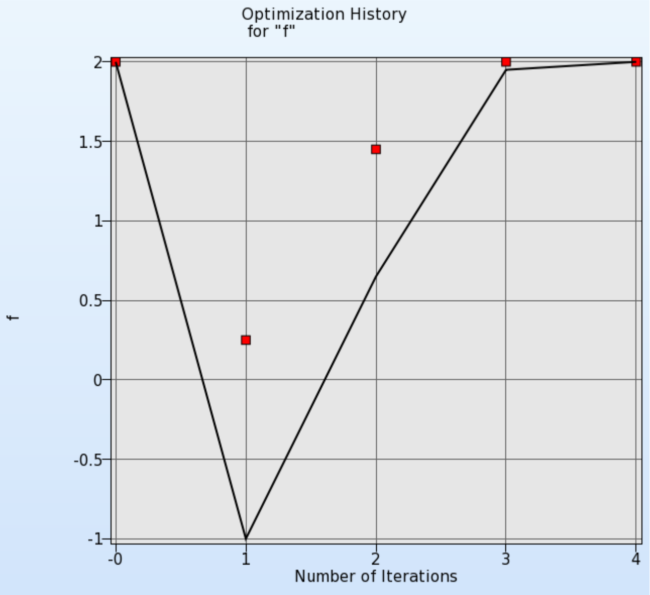


Figure 20-122: Optimization history for objective

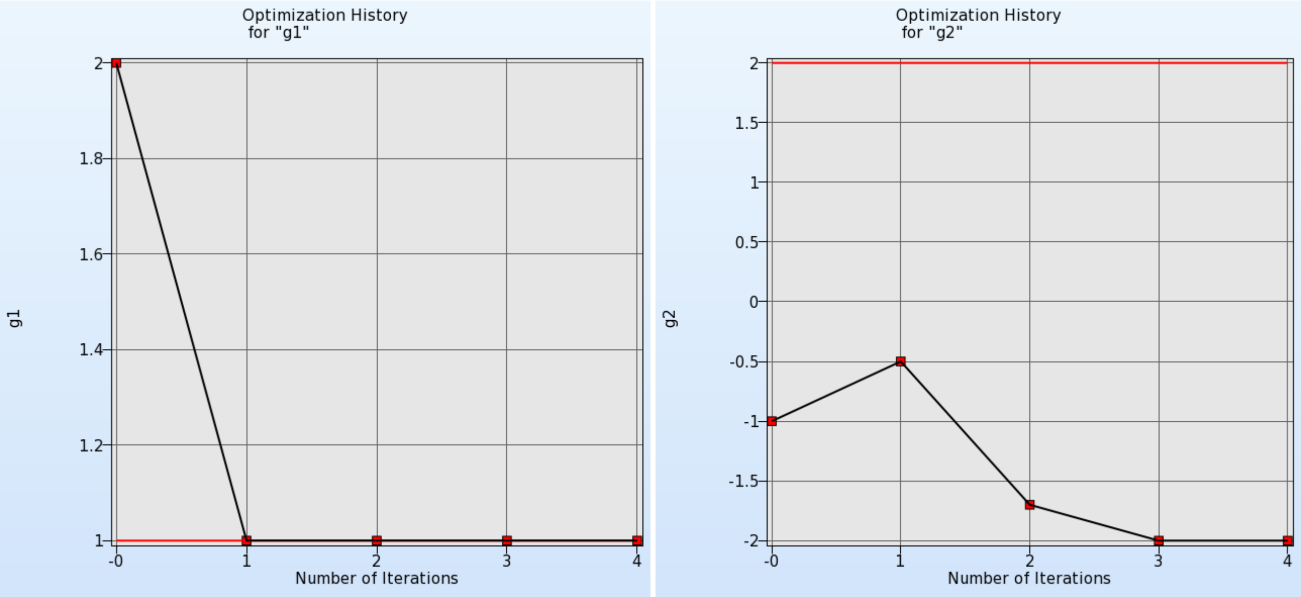


Figure 20-123: Optimization history for constraints

20.9. Small car crashworthiness example using GenEx to extract histories/responses from data files

20.9.1. Problem Description

- This example demonstrates the use of GenEx for extracting histories and responses from LS-DYNA data files.
- The small car crash design optimization example (see Section 17.2) has been modified by defining LS-DYNA histories and responses using GenEx. Even though GenEx is not required for this example, it has been used only to demonstrate its use.
- This example is a minimization problem with total mass of four parts (part no.2, 3, 4 and 5) as the objective and the intrusion distance calculated as the difference between displacements of two nodes (432 and 167) as the design constraint.
- The steps to define the task type and design parameters are similar to other simple LS-OPT examples. In this example, internal energy, nodal acceleration and rigid wall force are defined as histories with part masses and nodal displacements defined as responses using various GenEx features.

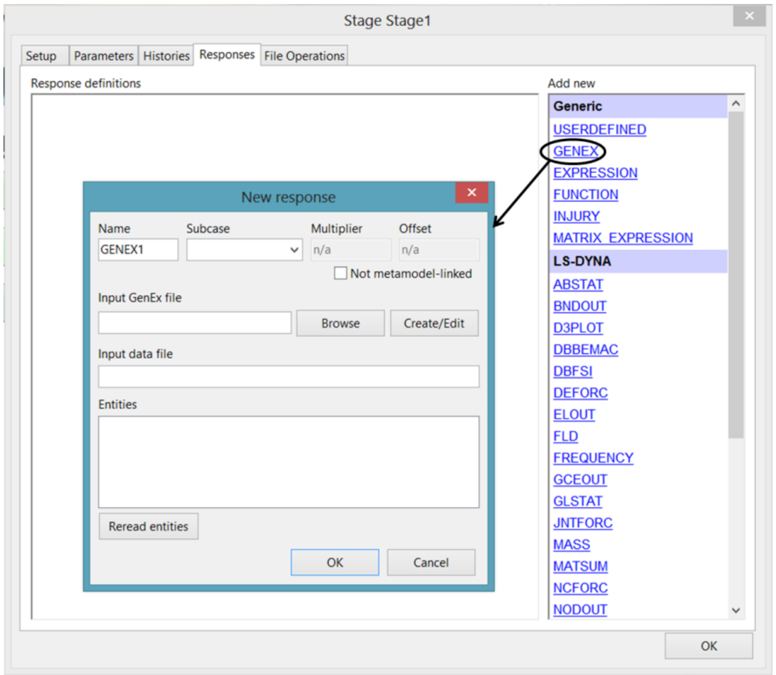
The following files are used in this example:

main.k	Main (root) file with LS-OPT design parameters
car5.k	Include file specified in main.k
rigid2	Include file specified in main.k
sample_nodout	Input data file for nodout histories/responses
sample_glstat	Input data file for glstat histories

sample_rwforc	Input data file for rwforc histories
sample_d3hsp	Input data file for d3hsp responses
genex_nodout.g6	GenEx input file for nodout
genex_glstat.g6	GenEx input file for glstat
genex_rwforc .g6	GenEx input file for rwforc
genex_d3hsp.g6	GenEx input file for d3hsp

20.9.2. Defining Responses in GenEx

1. Open the file `genex.start.lsopt` using the LS-OPT GUI.
2. The task, parameters, sampling and solver settings have already been defined in the project file.
3. The next step is to define responses using GenEx so that these responses can later be assigned as optimization objectives or constraints.
4. Instead of choosing LS-DYNA responses, the responses are defined using GenEx i.e. using the ‘GENEX’ option available under generic list of options within the ‘Histories’ and ‘Responses’ tabs of the *Setup* dialog box. Defining histories/responses using GenEx requires an input data file and a .g6 GenEx file as shown in Figure 20-124..

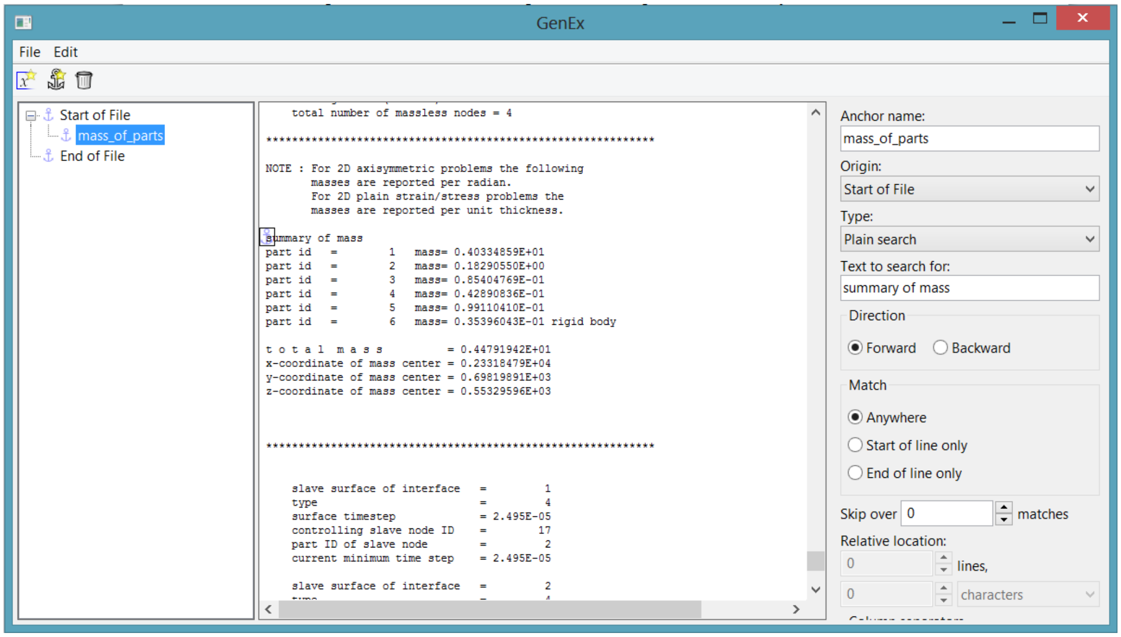


Name: GENEX1

Figure 20-124: GenEx response dialog box

5. To define the sum of part masses as an objective, the mass of each part should be defined separately as a response. Since the values of the part masses are extracted from the same data file (d3hsp), they can have the same GenEx input file (.g6 file). The GenEx input file stores the locations of response values to be extracted from the data file.

6. To define the mass of part 2 as a response, click on GENEX within the Responses tab. Assign a name to the response and click on Create/Edit to open the GenEx window for creating a .g6 file (GenEx input file).
7. The mass is extracted from the d3hsp file of LS-DYNA, therefore open sample_d3hsp file in GenEx (File → Select input file). These sample files are output files of the baseline analysis.
8. The mass of part 2 in d3hsp file is identified using anchors and entities. Anchors facilitate searching for a field from the data file and an Entity is the actual value field to be extracted.
9. The part mass information is printed in d3hsp under 'summary of mass' section. To define an anchor, click on New Anchor, assign a name (e.g. mass_of_parts) and enter "summary of mass" in 'Text to search for' field and hit enter. This creates an anchor at the start of the first occurrence of text "summary of mass" throughout d3hsp file as shown in Figure 20-125.

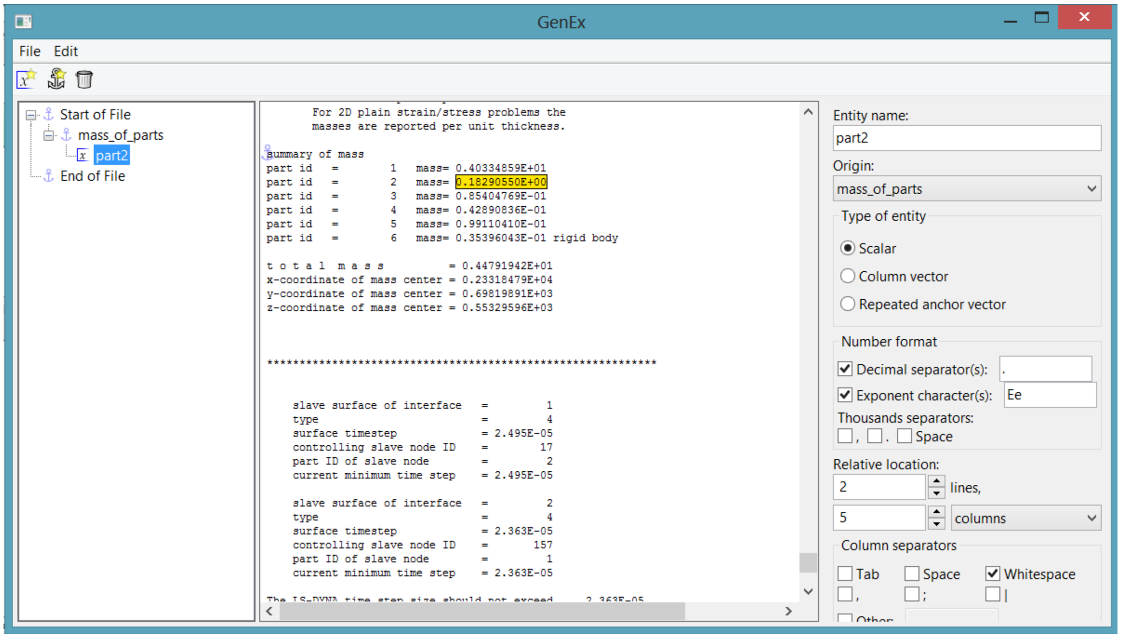


```
    total number of massless nodes = 4

 ***********************************************************

 NOTE : For 2D axisymmetric problems the following
        masses are reported per radian.
        For 2D plain strain/stress problems the
        masses are reported per unit thickness.

 summary of mass
 part id   =        1   mass= 0.40334859E+01
 part id   =        2   mass= 0.18290550E+00
 part id   =        3   mass= 0.85404769E-01
 part id   =        4   mass= 0.42890836E-01
 part id   =        5   mass= 0.99110410E-01
 part id   =        6   mass= 0.35396043E-01 rigid body

 t o t a l  m a s s          = 0.44791942E+01
 x-coordinate of mass center = 0.23318479E+04
 y-coordinate of mass center = 0.69819891E+03
 z-coordinate of mass center = 0.55329596E+03



 ***********************************************************


    slave surface of interface   =         1
    type                         =         4
    surface timestep             = 2.495E-05
    controlling slave node ID    =        17
    part ID of slave node        =         2
    current minimum time step    = 2.495E-05

    slave surface of interface   =         2
```

Anchor name: mass_of_parts
Origin: Start of File
Type: Plain search
Text to search for: summary of mass
Direction: ● Forward ○ Backward
Match: ● Anywhere ○ Start of line only ○ End of line only
Skip over 0 matches
Relative location: 0 lines, 0 characters

Figure 20-125: Creating Anchors in GenEx

10. Now an entity can be defined under this anchor using the *New Entry* option. Since responses are scalar values, the '*Type of Entity'* is selected as '*Scalar*'. The relative location of this entity is adjusted to obtain the value of mass of Part 2. For example, in the *sample_d3hsp* file, the relative location of mass of Part 2 with respect to the defined anchor is specified as line 2, column 5. The final value to be extracted will be highlighted on the data file as shown in Figure 20-126.



```
       For 2D plain strain/stress problems the
       masses are reported per unit thickness.

summary of mass
part id   =       1   mass= 0.40334859E+01
part id   =       2   mass= 0.18290550E+00
part id   =       3   mass= 0.85404769E-01
part id   =       4   mass= 0.42890836E-01
part id   =       5   mass= 0.99110410E-01
part id   =       6   mass= 0.35396043E-01 rigid body

t o t a l  m a s s          = 0.44791942E+01
x-coordinate of mass center = 0.23318479E+04
y-coordinate of mass center = 0.69819891E+03
z-coordinate of mass center = 0.55329596E+03



************************************************************


    slave surface of interface   =         1
    type                         =         4
    surface timestep             = 2.495E-05
    controlling slave node ID    =        17
    part ID of slave node        =         2
    current minimum time step    = 2.495E-05

    slave surface of interface   =         2
    type                         =         4
    surface timestep             = 2.363E-05
    controlling slave node ID    =       157
    part ID of slave node        =         1
    current minimum time step    = 2.363E-05

The LS-DYNA time step size should not exceed     2.363E-05
```

Entity name: part2

Origin: mass_of_parts

Type of entity

● Scalar
○ Column vector
○ Repeated anchor vector

Number format

☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

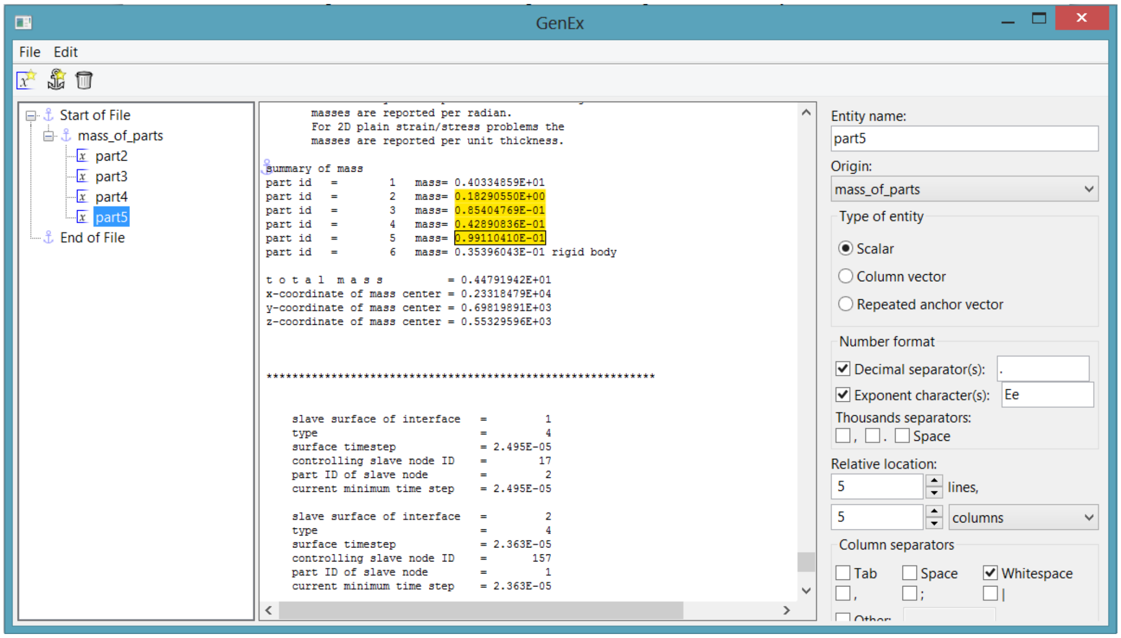
Relative location:
2 lines,
5 columns

Column separators

□ Tab □ Space ☑ Whitespace
□ , □ ; □ |
□ Other:

Figure 20-126: Entity selection in GenEx

11. Similarly, more entities can be created for other parts under the same anchor with the only difference being their respective relative locations, Figure 20-127.



```
       masses are reported per radian.
       For 2D plain strain/stress problems the
       masses are reported per unit thickness.

summary of mass
part id   =       1   mass= 0.40334859E+01
part id   =       2   mass= 0.18290550E+00
part id   =       3   mass= 0.85404769E-01
part id   =       4   mass= 0.42890836E-01
part id   =       5   mass= 0.99110410E-01
part id   =       6   mass= 0.35396043E-01 rigid body

t o t a l  m a s s          = 0.44791942E+01
x-coordinate of mass center = 0.23318479E+04
y-coordinate of mass center = 0.69819891E+03
z-coordinate of mass center = 0.55329596E+03



************************************************************


    slave surface of interface   =         1
    type                         =         4
    surface timestep             = 2.495E-05
    controlling slave node ID    =        17
    part ID of slave node        =         2
    current minimum time step    = 2.495E-05

    slave surface of interface   =         2
    type                         =         4
    surface timestep             = 2.363E-05
    controlling slave node ID    =       157
    part ID of slave node        =         1
    current minimum time step    = 2.363E-05
```

Entity name: part5

Origin: mass_of_parts

Type of entity

● Scalar
○ Column vector
○ Repeated anchor vector

Number format

☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

Relative location:
5 lines,
5 columns

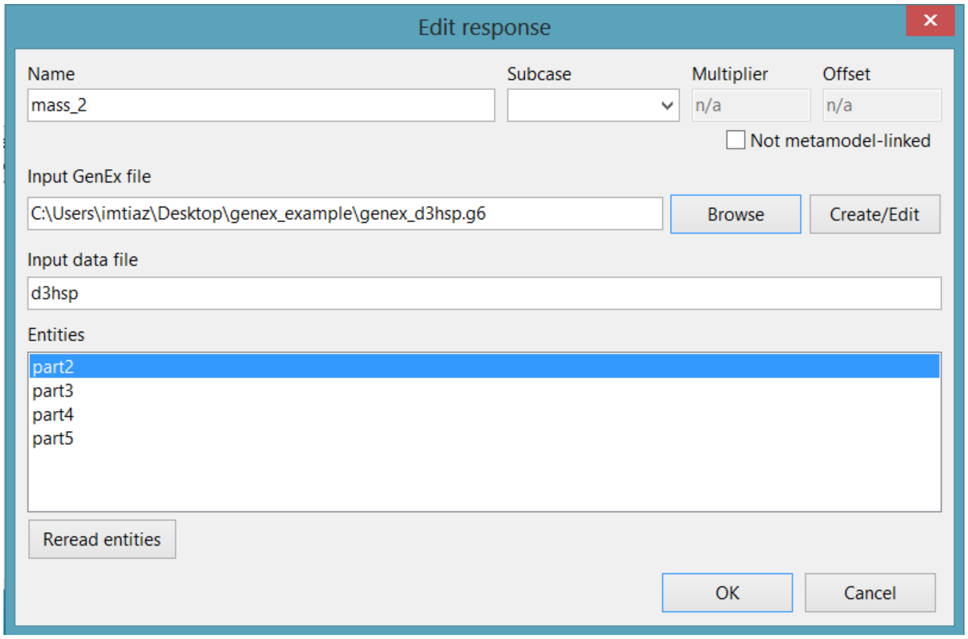
Column separators

□ Tab □ Space ☑ Whitespace
□ , □ ; □ |
□ Other:

Figure 20-127: Multiple entities of an anchor in GenEx

12. Once all the entities are defined, save the extraction setup to create a *.g6* file (*sample_d3hsp.g6*). Now the LS-OPT responses can be defined using this *.g6* file and corresponding data file. In the *New Response* dialog box, select this *.g6* file as the GenEx input file. Once this file is selected, LS-OPT lists all the entities defined in the file. Select the entity (part_2) to define it as an LS-OPT response

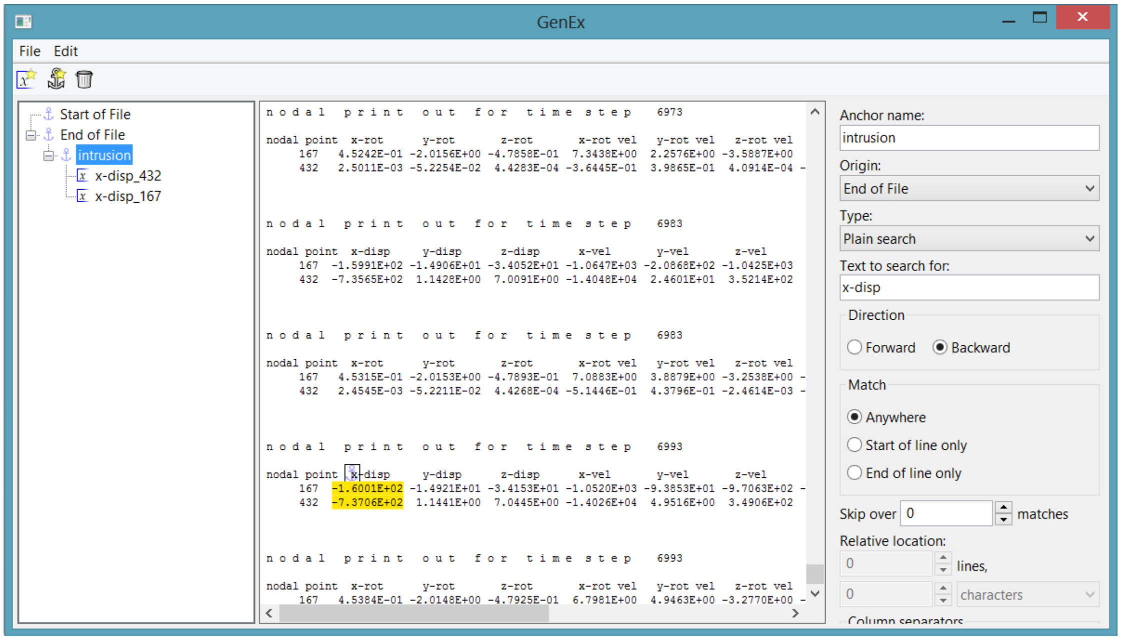
with *d3hsp* being the input data file, as shown in Figure 20-128. Repeat this process to define the LS-OPT mass responses of all the parts specified.



Name: mass_2
Subcase:
Multiplier: n/a
Offset: n/a
☐ Not metamodel-linked
Input GenEx file: C:\Users\imtiaz\Desktop\genex_example\genex_d3hsp.g6
Input data file: d3hsp
Entities:
part2
part3
part4
part5

Figure 20-128: Defining GenEx response in LS-OPT

13. To create GenEx responses for extracting nodal displacements, similar steps can be followed with *nodout* being the input data file. The sample *nodout* file (*sample_nodout*) provided with the example can be used to create the required GenEx input file. An anchor can be created to search for text 'x-disp'. By default, this anchor is created under *Start of file* anchor with *forward* search direction and hence the search results in locating the first occurrence of 'x-disp' within the *nodout* data file. Since the last reported displacement values are required, the *anchor origin* can be changed to '*End of file*' with a *backward* search direction as shown in Figure 20-129. Entities for the x-displacement of both nodes (432 and 167) can be defined under this anchor.



```
n o d a l   p r i n t   o u t   f o r   t i m e  s t e p    6973

nodal point  x-rot      y-rot       z-rot       x-rot vel   y-rot vel   z-rot vel
     167   4.5242E-01 -2.0156E+00 -4.7858E-01  7.3438E+00  2.2576E+00 -3.5887E+00
     432   2.5011E-03 -5.2254E-02  4.4283E-04 -3.6445E-01  3.9865E-01  4.0914E-04 -



n o d a l   p r i n t   o u t   f o r   t i m e  s t e p    6983

nodal point  x-disp     y-disp      z-disp      x-vel       y-vel       z-vel
     167  -1.5991E+02 -1.4906E+01 -3.4052E+01 -1.0647E+03 -2.0868E+02 -1.0425E+03
     432  -7.3565E+02  1.1428E+00  7.0091E+00 -1.4048E+04  2.4601E+01  3.5214E+02



n o d a l   p r i n t   o u t   f o r   t i m e  s t e p    6983

nodal point  x-rot      y-rot       z-rot       x-rot vel   y-rot vel   z-rot vel
     167   4.5315E-01 -2.0153E+00 -4.7893E-01  7.0883E+00  3.8879E+00 -3.2538E+00 -
     432   2.4545E-03 -5.2211E-02  4.4268E-04 -5.1446E-01  4.3796E-01 -2.4614E-03 -



n o d a l   p r i n t   o u t   f o r   t i m e  s t e p    6993

nodal point  x-disp     y-disp      z-disp      x-vel       y-vel       z-vel
     167  -1.6001E+02 -1.4921E+01 -3.4153E+01 -1.0520E+03 -9.3853E+01 -9.7063E+02 -
     432  -7.3706E+02  1.1441E+00  7.0445E+00 -1.4026E+04  4.9516E+00  3.4906E+02



n o d a l   p r i n t   o u t   f o r   t i m e  s t e p    6993

nodal point  x-rot      y-rot       z-rot       x-rot vel   y-rot vel   z-rot vel
     167   4.5384E-01 -2.0148E+00 -4.7925E-01  6.7981E+00  4.9463E+00 -3.2770E+00 -
```

Anchor name: intrusion

Origin: End of File

Type: Plain search

Text to search for: x-disp

Direction

○ Forward ● Backward

Match

● Anywhere

○ Start of line only

○ End of line only

Skip over 0 matches

Relative location:

0 lines,

0 characters

Column separators

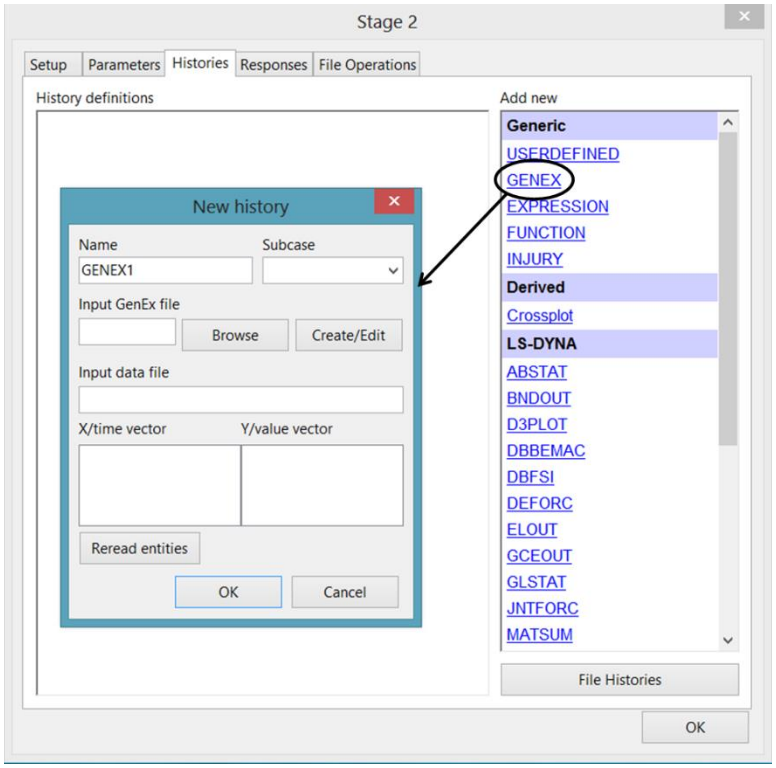
Figure 20-129: Changing anchor origin and search direction for backward search in GenEx

14. Once all the required entities are defined, the process is saved as genex_nodout.g6 and Step 12 can be repeated to define the LS-OPT responses using this GenEx file with nodout being the input data file.
15. As with any other LS-OPT responses, these GenEx responses can be assigned as optimization objectives/constraints.

20.9.3. Defining Histories in GenEx

Even though histories are not utilized in this example problem, internal energy, nodal acceleration and rigid wall force histories have been defined to demonstrate the use of GenEx for extracting histories from an input data file.

Similar to responses, GenEx histories require an input data file and its corresponding .g6 GenEx file, as shown in Figure 20-130.



Name: GENEX1

Figure 20-130: GenEx history dialog box

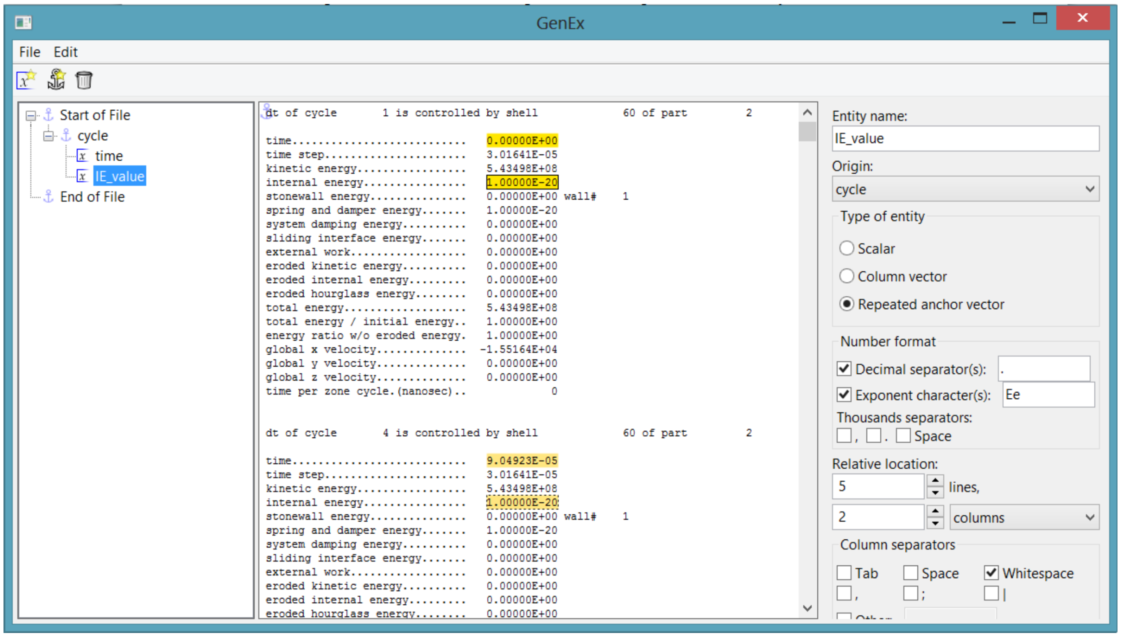
To create internal energy history, click on GENEX and assign a name to the history. The LS-DYNA ASCII file *glstat* is used as input data file, i.e. internal energy history is extracted from *glstat*. To create a .g6 file (genex input file), click on *Create/Edit* to open the GenEx window and select *sample_glstat* as the input file from the GenEx window (File → Select input file).

Now Anchors and Entities should be defined to locate internal energy values at each time interval.

To define an anchor, click on *New Anchor*, assign a name (e.g. cycle) and enter “dt of cycle” in the ‘*Text to search for*’ field and hit enter. This creates an anchor at the start of the first occurrence of text “dt of cycle”.

Now create entities for time (*x*-vector) and internal energy values (*y*-vector) using this anchor. Click on *‘New Entry’* and find the relative location of internal energy values with the defined anchor. The relative location can be determined using *lines*, *characters* and *columns* options. In this example, the entity *IE_value* is located at line 5, column 2 relative to the anchor and the entity *time* is located at line 2, column 1 (with whitespace as the column separator).

Since this is a history, the entities *time* and *IE_value* at each cycle should be extracted. This can be done by selecting *Repeated Anchor Vector* as entity type. Selecting *Repeated Anchor Vector* as anchor type highlights all the entity fields with locations relative to text “dt of cycle” throughout the data file (shown in Figure 20-131). Once all the anchors and entities are defined, save the GenEx file and close the GenEx window.



```
dt of cycle      1 is controlled by shell          60 of part       2

time..........................   0.00000E+00
time step.....................   3.01641E-05
kinetic energy................   5.43498E+08
internal energy...............   1.00000E-20
stonewall energy..............   0.00000E+00 wall#    1
spring and damper energy......   1.00000E-20
system damping energy.........   0.00000E+00
sliding interface energy......   0.00000E+00
external work.................   0.00000E+00
eroded kinetic energy.........   0.00000E+00
eroded internal energy........   0.00000E+00
eroded hourglass energy.......   0.00000E+00
total energy..................   5.43498E+08
total energy / initial energy..  1.00000E+00
energy ratio w/o eroded energy.  1.00000E+00
global x velocity............. -1.55164E+04
global y velocity.............   0.00000E+00
global z velocity.............   0.00000E+00
time per zone cycle.(nanosec)..            0


dt of cycle      4 is controlled by shell          60 of part       2

time..........................   9.04923E-05
time step.....................   3.01641E-05
kinetic energy................   5.43498E+08
internal energy...............   1.00000E-20
stonewall energy..............   0.00000E+00 wall#    1
spring and damper energy......   1.00000E-20
system damping energy.........   0.00000E+00
sliding interface energy......   0.00000E+00
external work.................   0.00000E+00
eroded kinetic energy.........   0.00000E+00
eroded internal energy........   0.00000E+00
eroded hourglass energy.......   0.00000E+00
```

Entity name: IE_value

Origin: cycle

Type of entity

○ Scalar
○ Column vector
● Repeated anchor vector

Number format

☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

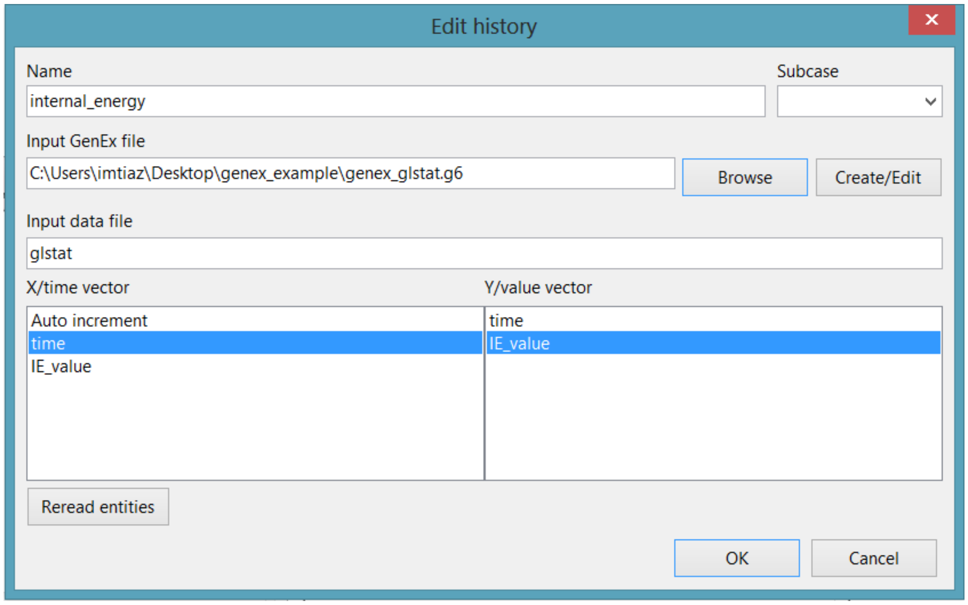
Relative location:
5 lines,
2 columns

Column separators

□ Tab □ Space ☑ Whitespace
□ , □ ; □ |

Figure 20-131: History definition using Repeated Anchor Vector as entity type

Now select the GenEx file created in the previous steps as the 'input GenEx file' of history and *glstat* as 'input data file'. Once the GenEx file is selected, the entities defined are listed under *X/time vector* and *Y/value vector*. Select the *time* entity as *X vector* and *IE_value* as *Y vector* and click *OK*, Figure 20-132. The internal energy history using GenEx has now been defined. When LS-OPT is executed, the defined entity fields are extracted as histories from *glstat* ASCII files generated as a result of the LS-DYNA analysis in each run directory.



Name: internal_energy

Subcase:

Input GenEx file: C:\Users\imtiaz\Desktop\genex_example\genex_glstat.g6

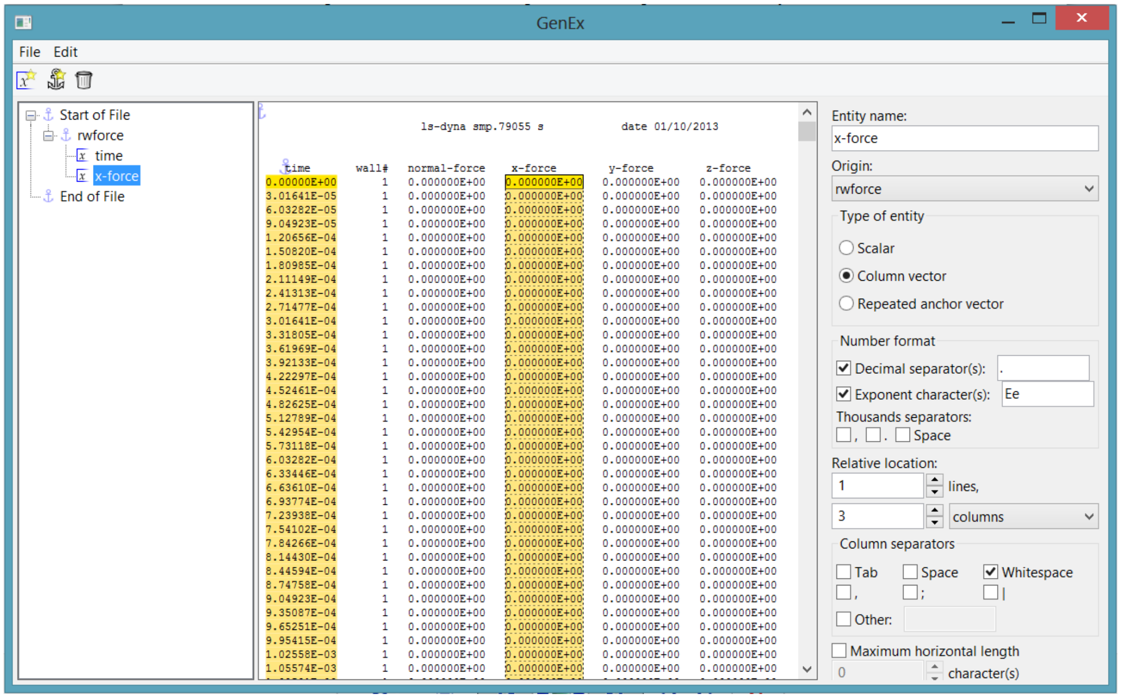
Input data file: glstat

X/time vector	Y/value vector
Auto increment	time
time	IE_value
IE_value	

Figure 20-132: Defining GenEx history in LS-OPT

Similarly, the nodal acceleration history can be formulated using Repeated Anchor Vector for the nodout input data file. To create the .g6 file, the same input GenEx file (genex_nodout) used for nodal displacement responses can be modified to include history entities.

The rigid wall force history can be extracted from rwforc data file. In this example, since only one rigid wall has been defined, the rwforc history (time vs. force) values are printed as a list. Therefore, after defining the anchor, the type of entity can be selected as Column Vector instead of using Repeated Anchor Vector. When Column Vector is selected all the components below the selected entity until the end of the file are highlighted (see Figure 20-133).



```
                  ls-dyna smp.79055 s          date 01/10/2013

   time      wall#  normal-force    x-force        y-force        z-force
0.00000E+00    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
3.01641E-05    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
6.03282E-05    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
9.04923E-05    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
1.20656E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
1.50820E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
1.80985E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
2.11149E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
2.41313E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
2.71477E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
3.01641E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
3.31805E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
3.61969E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
3.92133E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
4.22297E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
4.52461E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
4.82625E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
5.12789E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
5.42954E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
5.73118E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
6.03282E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
6.33446E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
6.63610E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
6.93774E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
7.23938E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
7.54102E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
7.84266E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
8.14430E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
8.44594E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
8.74758E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
9.04923E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
9.35087E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
9.65251E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
9.95415E-04    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
1.02558E-03    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
1.05574E-03    1    0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
```

Entity name: x-force

Origin: rwforce

Type of entity
○ Scalar
● Column vector
○ Repeated anchor vector

Number format
☑ Decimal separator(s): .
☑ Exponent character(s): Ee
Thousands separators:
□ , □ . □ Space

Relative location:
1 lines,
3 columns

Column separators
□ Tab □ Space ☑ Whitespace
□ , □ ; □ |
□ Other:

□ Maximum horizontal length
0 character(s)

Figure 20-133: History definition using Column Vector as entity type

If a user requires a limited number of components of the column, the *Maximum Number of Components* check box can be used to define the required number.

Once the GenEx file is created, the LS-OPT histories can be defined using this file as explained in *Step 8*.

20.9.4. Optimization Results

- The optimization problem was solved using metamodel-based sequential optimization with domain reduction technique.
- The process took seven iterations (with five design points in each iteration) to converge.
- At the optimum design, the total mass of selected parts was 0.465kg and the computed intrusion distance was 549.29mm vs. 550mm predicted by the metamodel.

20.10. Optimization using classifier-based constraint: Small car NVH (2 variables)

This example has the following features:

- An LS-DYNA implicit modal analysis is performed using the double precision solver.
- Mass of the car is minimized while constraining the first torsional mode frequency.
- Mode switching can occur from one design to another; therefore the torsional mode is tracked.
- The frequency response is discontinuous due to the mode switching.
- An SVM classifier is used to approximate the constraint boundary separating the feasible and infeasible designs.
- Optimization is performed using a hybrid SA algorithm using a neural network-based objective function approximation and the SVM classifier-based constraint boundary.

20.10.1. Introduction

This example consists of an LS-DYNA simplified car model (Figure 20-134) for which the mass is minimized while constraining the first torsional mode to be greater than a certain limit. The thickness values of the bumper (tbumper) and the rail (trailb) are the optimization variables. Thus the optimization problem is:

$$\min_{tbumper,thood} \quad Mass$$

$$s.t.\, \nu_{torsion} > 2.2$$

where $\nu_{torsion}$ is the first torsional mode frequency.

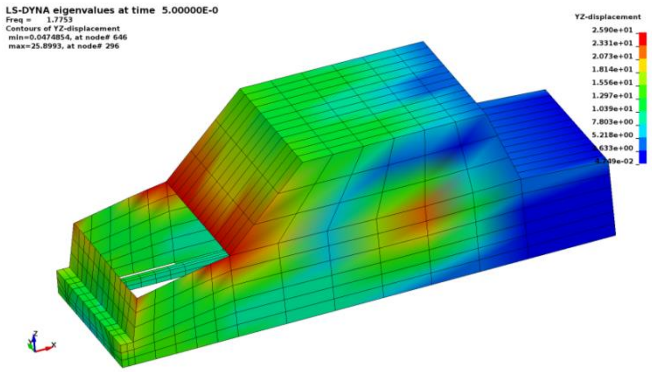


Figure 20-134: Simplified car LS-DYNA model for modal analysis

20.10.2. Solution and Results

The first torsional mode for the baseline design was identified before tracking the closest mode shape for the other samples, based on the MAC criterion (same as Figure 20-84). The mode number can switch from one design to another, resulting in a discontinuity (Figure 20-139).

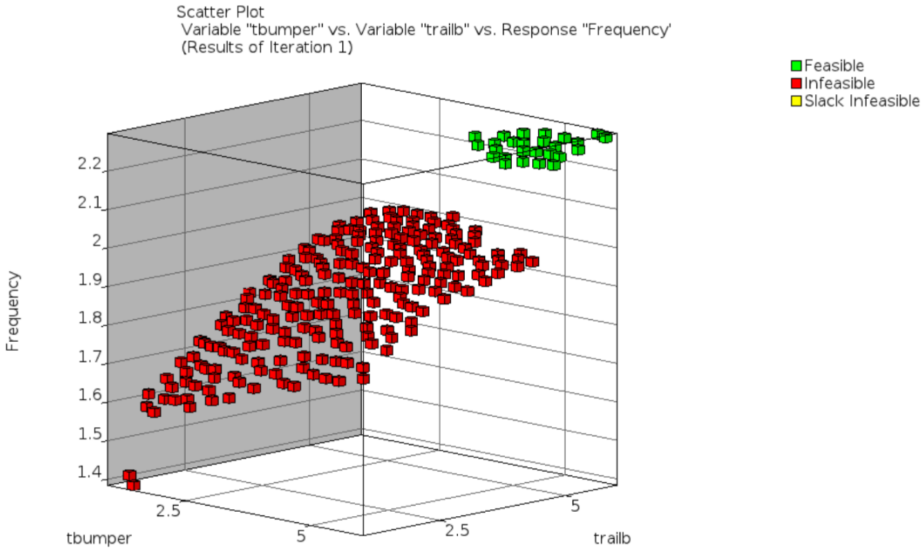


Figure 20-135: Discontinuous frequency response due to mode switching.

The problem is solved using a feedforward neural net (FNN) approximation of the frequency as well as using an SVM classifier in order to demonstrate the difference between the two approaches. Both the setups consist of a single iteration with 250 samples. However, the additional “Classifier” box in the GUI flowchart should be noted for classifier-based approach.

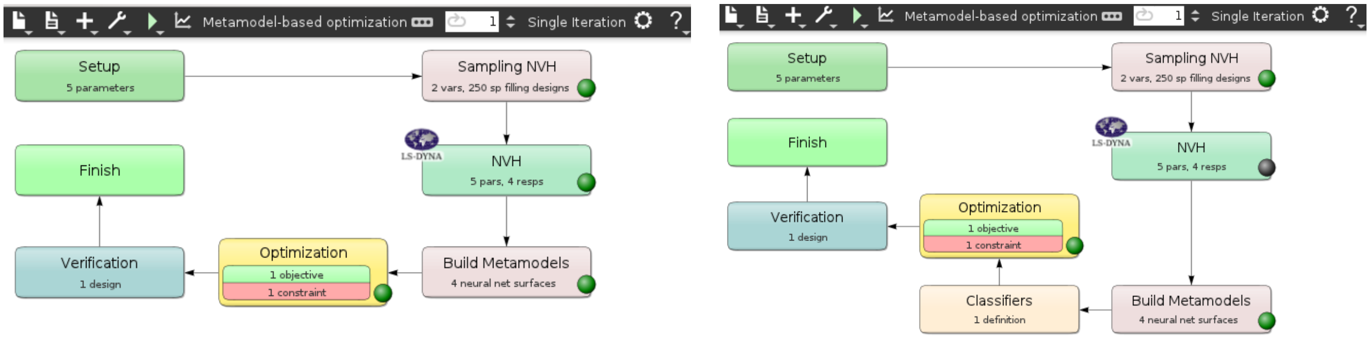
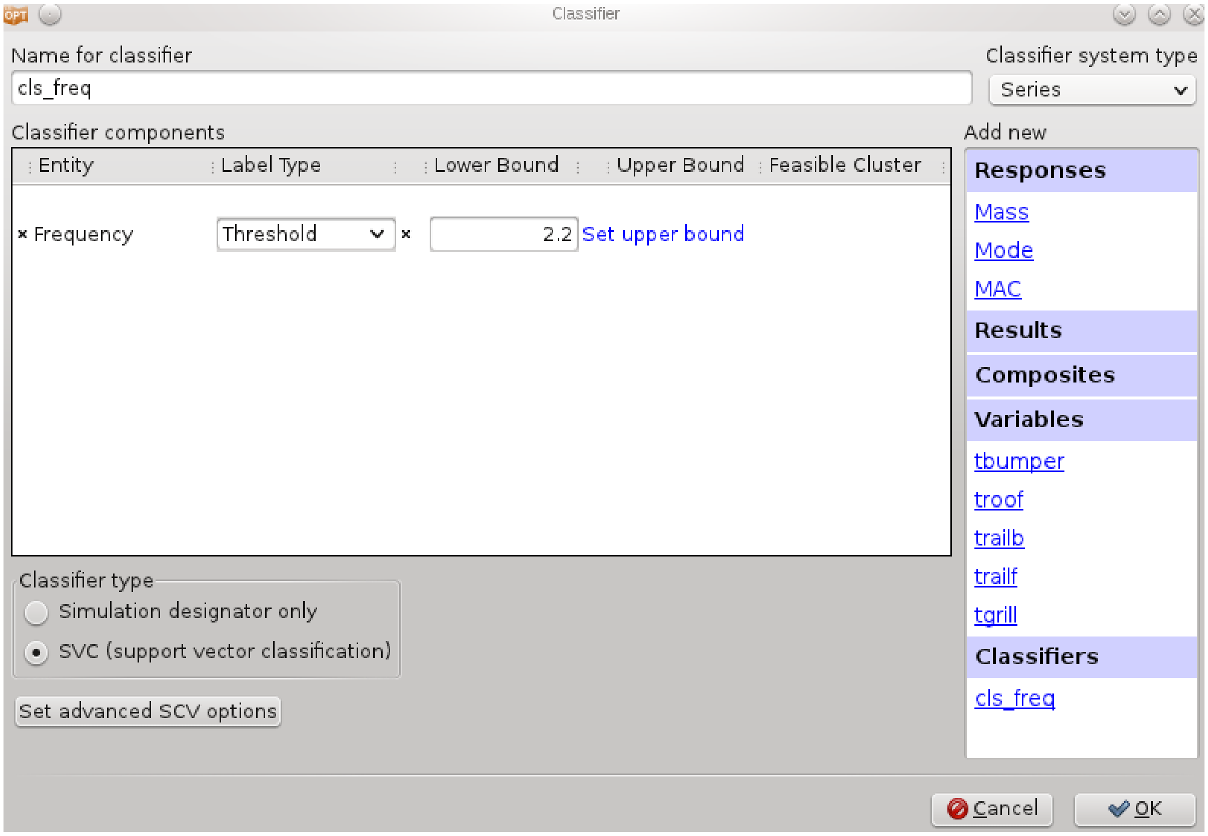


Figure 20-136: Modal Analysis Example. LS-OPT setups using metamodel-based (left) and classifier-based (right) methods.

The classifier box (Figure 20-136) contains the definition of a classifier based on the frequency response. The definition consists of a feasibility criterion with a frequency upper bound of 2.2 as well as the classifier type (SVC) (Figure 20-137). The classifier system type is set as default “series”, but is irrelevant here as there is only one feasibility criterion or classifier component. A comparison of the constraint definitions in the two approaches is provided in Figure 20-138.



Name for classifier: cls_freq

Classifier system type: Series

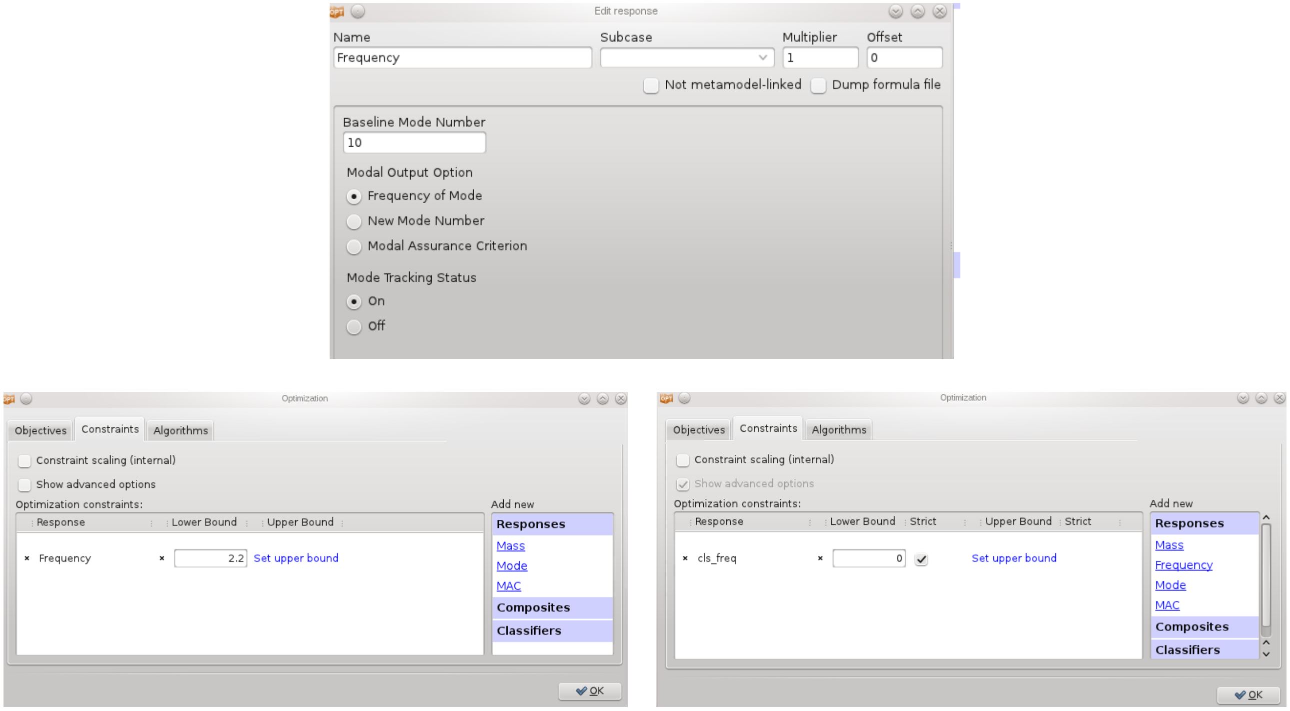
Classifier components

Entity	Label Type	Lower Bound	Upper Bound	Feasible Cluster
× Frequency	Threshold	×	2.2 Set upper bound	

Classifier type
○ Simulation designator only
● SVC (support vector classification)

Figure 20-137: Classifier definition dialog.

The principal difference between the metamodel and classifier based methods is that in the former the constraint function values, i.e. frequency values, are approximated for any arbitrary design while in the later approach only the feasibility of the design is predicted. Using a metamodel the feasibility of a design is determined by comparing its predicted frequency constraint value to its upper bound (2.2). Thus, the feasibility information is mapped back to the design space following the metamodel prediction and threshold comparison. Figure 20-139 shows the neural network approximations of the mass and frequency responses and the corresponding feasibility. Figure 20-140 (left) shows a projection of the metamodel-based predicted feasibility information on the design space along with constraint boundary approximation. On the contrary, the feasibility criterion is a part of the classifier definition (Figure 20-137); the classifier directly constructs the constraint decision boundary in the design space (Figure 20-140 right).



Edit response

Name	Subcase	Multiplier	Offset
Frequency		1	0

□ Not metamodel-linked □ Dump formula file

Baseline Mode Number
10

Modal Output Option
● Frequency of Mode
○ New Mode Number
○ Modal Assurance Criterion

Mode Tracking Status
● On
○ Off

Optimization

□ Constraint scaling (internal)
□ Show advanced options

Optimization constraints:

Response	Lower Bound	Upper Bound
× Frequency	× 2.2	Set upper bound

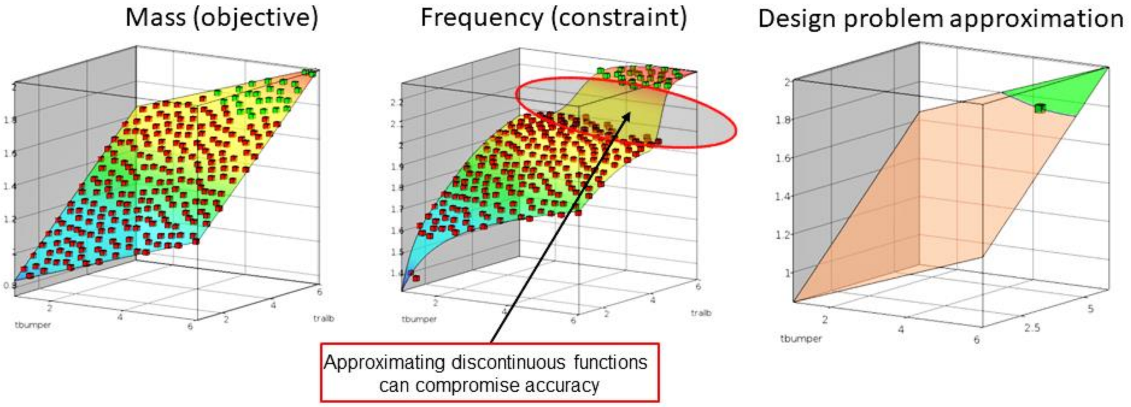
Optimization

□ Constraint scaling (internal)
☑ Show advanced options

Optimization constraints:

Response	Lower Bound	Strict	Upper Bound	Strict
× cls_freq	× 0	✓	Set upper bound	

Figure 20-138: Constraint definition in metamodel-based and classification-based approaches. On the top the underlying frequency response definition is shown. In the metamodel-based method the constraint is defined with respect to the approximation of the frequency response (bottom left). In the classification-based method the classifier defined in Figure 20-137 is constraint to be greater than zero (positive is feasible).



Mass (objective)

Frequency (constraint)

Design problem approximation

Approximating discontinuous functions can compromise accuracy

Figure 20-139: FNN approximation for mass minimization with torsional frequency constraint. The frequency response is discontinuous due to mode switching.

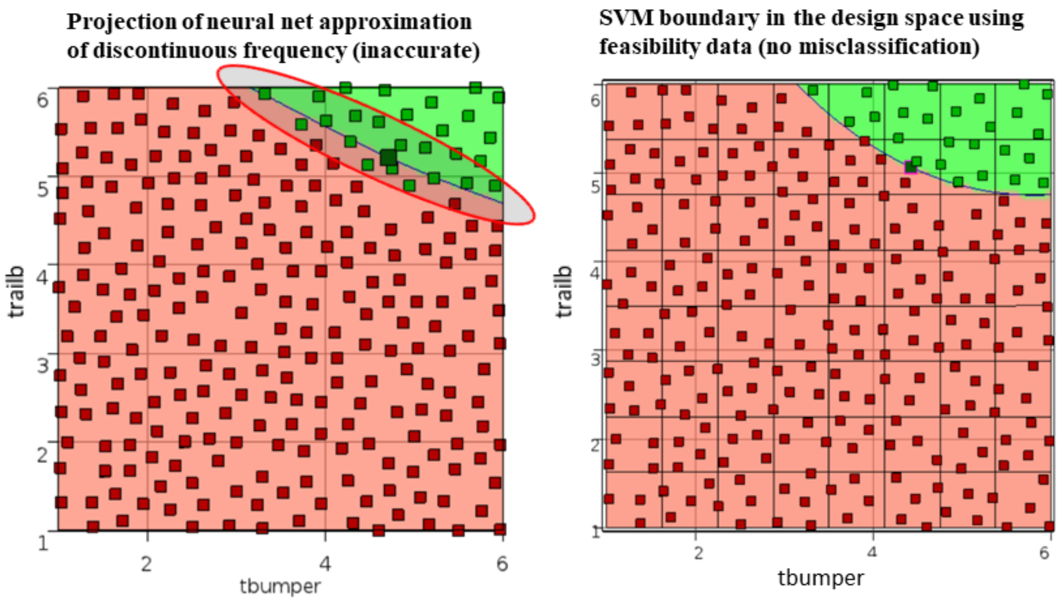


Figure 20-140: Modal analysis example with discontinuous frequency constraint. Comparison of FNN and SVM constraint boundaries.

The FNN-based approximation clearly misclassifies some of the feasible training samples and predicts them as infeasible. This leads to a conservative design with tbumper = 4.71496 and trailb 5.20927. The corresponding scaled mass objective is 1.78755. Using an SVM-based constraint, the optimal design is located at (tbumper = 4.42104, trailb 5.06372) with a lower mass value of 1.74285.

20.11. Multi-disciplinary optimization using classifier constraints

A multi-disciplinary optimization problem is solved using classifier-based constraints to demonstrate the computational savings associated with classifier-based constraints as compared to conventional metamodel-based constraint approximation approach.

The example has the following features:

- An LS-DYNA implicit modal analysis is performed initially using the double precision solver.
- A user-defined perl script is used to check feasibility of NVH constraint before running explicit crash simulations.
- The crash analysis termination time of infeasible NVH designs are set to approximately zero to save computational cost.
- Mass of the car is minimized while constraining the first torsional mode and three stage pulses from the crash analysis.
- An SVM classifier is used to approximate the constraint boundary separating the feasible and infeasible designs.
- Optimization is performed using sequential with domain reduction strategy, neural network-based objective function approximation, and an SVM classifier-based constraint boundary.

20.11.1. Problem Statement

The thicknesses of 10 parts in the front end of the vehicle were selected as design variables. Due to symmetry in the design, the number of design variables was reduced to seven. The total mass of selected parts was minimized while constraining the torsional frequency and three crash stage pulses to certain limits. The NVH "body in white" design and the finite model of the crash analysis along with the selected design parts are shown in Figure 20-141.

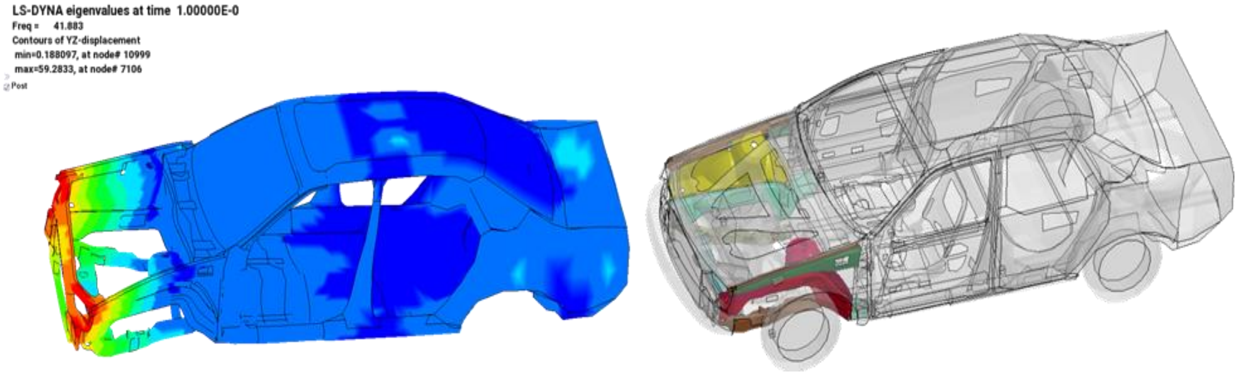

LS-DYNA eigenvalues at time 1.00000E-0
Freq = 41.883
Contours of YZ-displacement
min=0.188097, at node# 10999
max=59.2833, at node# 7106
Post

Figure 20-141: NVH FE model at torsional mode (left) and crash FE model with design parts (right).

The optimization problem formulation is shown below.

$$min \qquad Mass\left(x_{1,2..7}\right)$$

$$s.t \quad 41.38Hz < b_{freq} < 42.38Hz$$

$$Stage\ 1\ pulse > 13.94g$$

$$Stage\ 2\ pulse > 19.17g$$

$$Stage\ 3\ pulse > 21.30g$$

where, $x_{(1,2,..7)}$ are seven thickness design variables and $b_{freq}$is the baseline torsional frequency.

20.11.2. Solution

The LS-OPT setup consists of three analysis stages in series to evaluate NVH design, check feasibility of the NVH design, and run crash analysis of feasible NVH designs. Sequential optimization with domain reduction is selected as the design strategy; however, it is important to note that all points of previous iterations are included in the setup. The main LS-OPT setup of this example is shown in Figure 20-142.

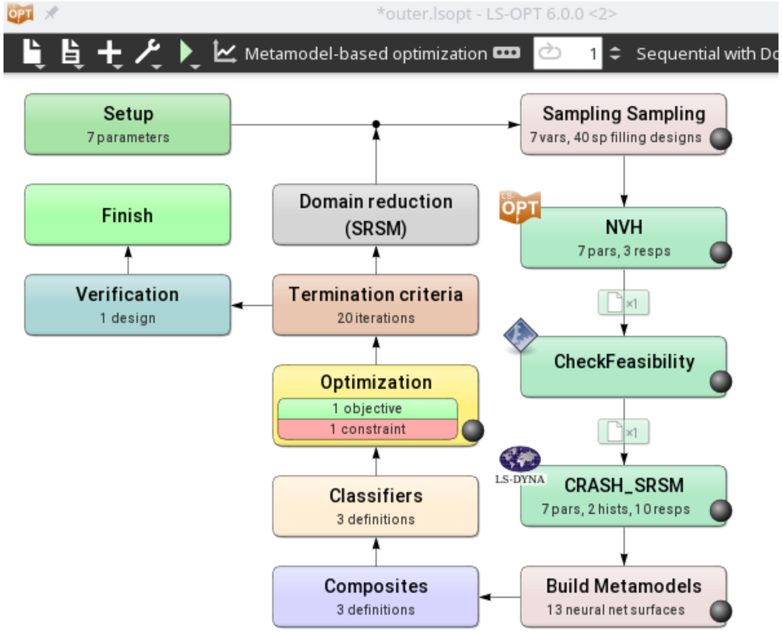
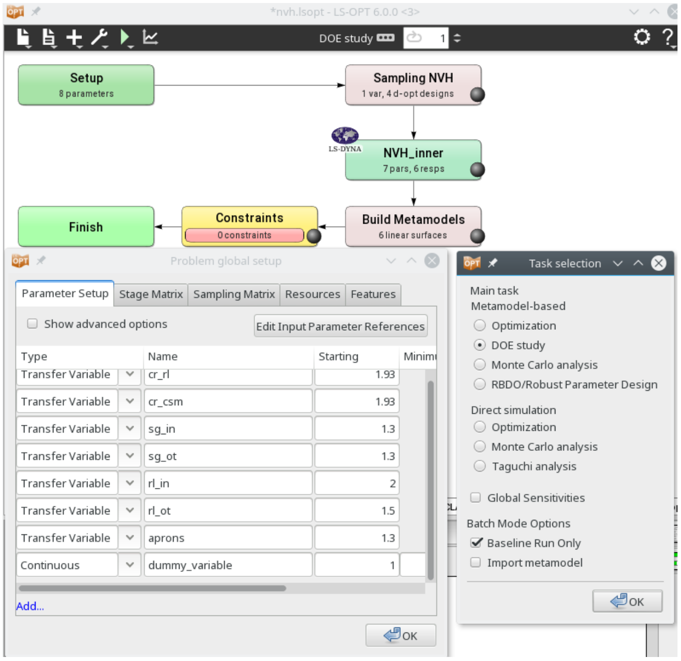


Figure 20-142: LS-OPT setup

Since mode switching can occur from one design to another, the baseline torsional mode shape was tracked using the MAC criterion. The NVH stage shown in the above figure corresponds to LS-DYNA analysis of the body in white model. However, since this setup requires extraction of the frequency response before analyzing the crash model, a multi-level setup was used for the NVH stage. The inner level NVH setup is shown in Figure 20-143.



□ Show advanced options

Type	Name	Starting	Minimu
Transfer Variable	cr_rl	1.93	
Transfer Variable	cr_csm	1.93	
Transfer Variable	sg_in	1.3	
Transfer Variable	sg_ot	1.3	
Transfer Variable	rl_in	2	
Transfer Variable	rl_ot	1.5	
Transfer Variable	aprons	1.3	
Continuous	dummy_variable	1	

Main task
Metamodel-based
○ Optimization
● DOE study
○ Monte Carlo analysis
○ RBDO/Robust Parameter Design

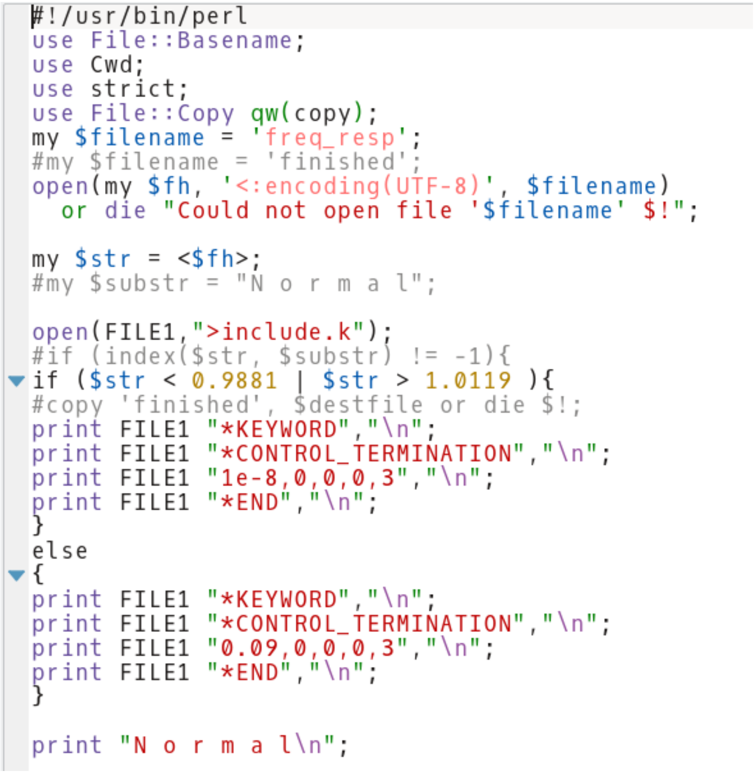
Direct simulation
○ Optimization
○ Monte Carlo analysis
○ Taguchi analysis

□ Global Sensitivities

Batch Mode Options
☑ Baseline Run Only
□ Import metamodel

Figure 20-143: Inner level setup for NVH analysis

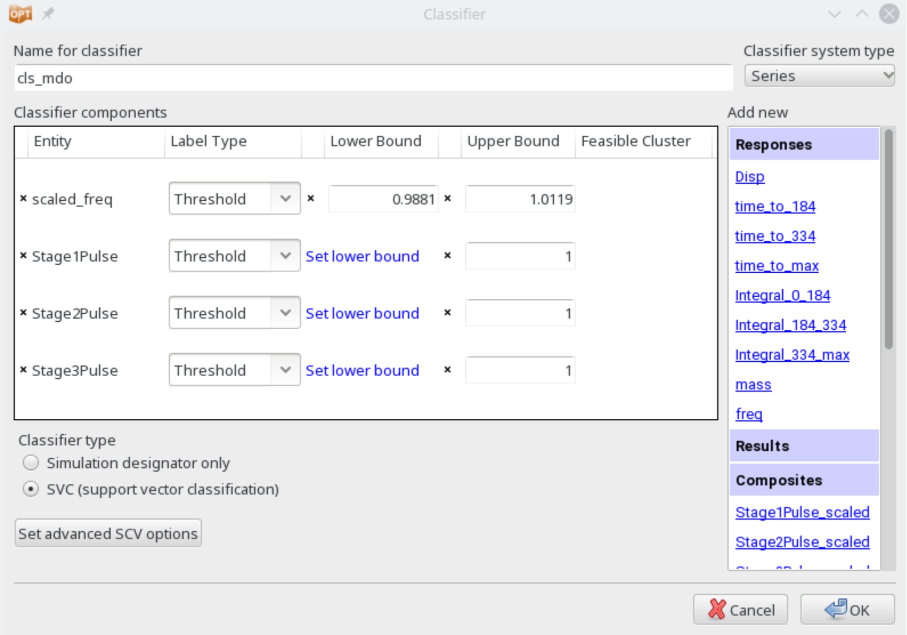
The inner level of the NVH stage is simply a baseline analysis of the design variables values passed as transfer variables from the outer level. The frequency response extracted from the outer level NVH stage is passed to the lower stage to check feasibility. The second stage in the main setup consists of a user-defined perl script used to modify the termination time of the crash analysis based on the feasibility of the NVH response. The termination time of the design points with infeasible NVH response is reduced to save computation cost. The perl script used in this example is shown in Figure 20-144.



```
#!/usr/bin/perl
use File::Basename;
use Cwd;
use strict;
use File::Copy qw(copy);
my $filename = 'freq_resp';
#my $filename = 'finished';
open(my $fh, '<:encoding(UTF-8)', $filename)
  or die "Could not open file '$filename' $!";

my $str = <$fh>;
#my $substr = "N o r m a l";

open(FILE1,">include.k");
#if (index($str, $substr) != -1){
if ($str < 0.9881 | $str > 1.0119 ){
#copy 'finished', $destfile or die $!;
print FILE1 "*KEYWORD","\n";
print FILE1 "*CONTROL_TERMINATION","\n";
print FILE1 "1e-8,0,0,0,3","\n";
print FILE1 "*END","\n";
}
else
{
print FILE1 "*KEYWORD","\n";
print FILE1 "*CONTROL_TERMINATION","\n";
print FILE1 "0.09,0,0,0,3","\n";
print FILE1 "*END","\n";
}

print "N o r m a l\n";
```

Figure 20-144: Perl script used to modify crash analysis termination time based on NVH response feasibility.

The third stage in the main setup consists of LS-DYNA analysis of the crash model. The frequency and stage pulse responses are scaled using the baseline values. The classifier box consists of classifier definition with feasibility criteria for all four responses. The bounds of the responses are scaled with respect to the baseline values, as shown in Figure 20-145. This classifier is selected as the constraint (lower bound zero) for the optimization with mass selected as the objective for minimization.



Classifier

Name for classifier: cls_mdo

Classifier system type: Series

Classifier components

Entity	Label Type	Lower Bound	Upper Bound	Feasible Cluster
× scaled_freq	Threshold	× 0.9881	× 1.0119	
× Stage1Pulse	Threshold	Set lower bound	× 1	
× Stage2Pulse	Threshold	Set lower bound	× 1	
× Stage3Pulse	Threshold	Set lower bound	× 1	

Classifier type

○ Simulation designator only

● SVC (support vector classification)

Figure 20-145: Classifier definitions for NVH and crash responses

20.11.3. Results

Figure 20-146 shows the optimization history of the best computed designs. The design variable values of the best computed design are cr_rl = 1.823, cr_csm = 1.191, sg_in = 1.218, sg_out = 1.901, rl_in = 2.226, rl_ot = 1.189, aprons = 1.027. The mass of the best computed design is 42.73 kg, a reduction of 3.27 kg from a baseline mass of 46 kg. The corresponding values of frequency constraint and three stage pulses are 41.63 Hz, 13.97g, 19.283g, and 22.033g, respectively.

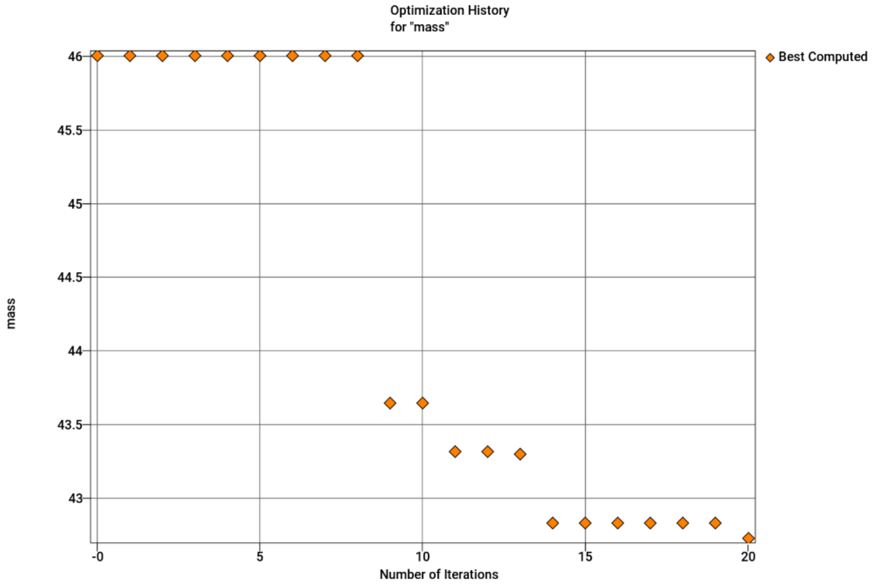


Figure 20-146: Mass response optimization history of the best computed designs

The computational cost savings in each iteration is shown inFigure 20-147.

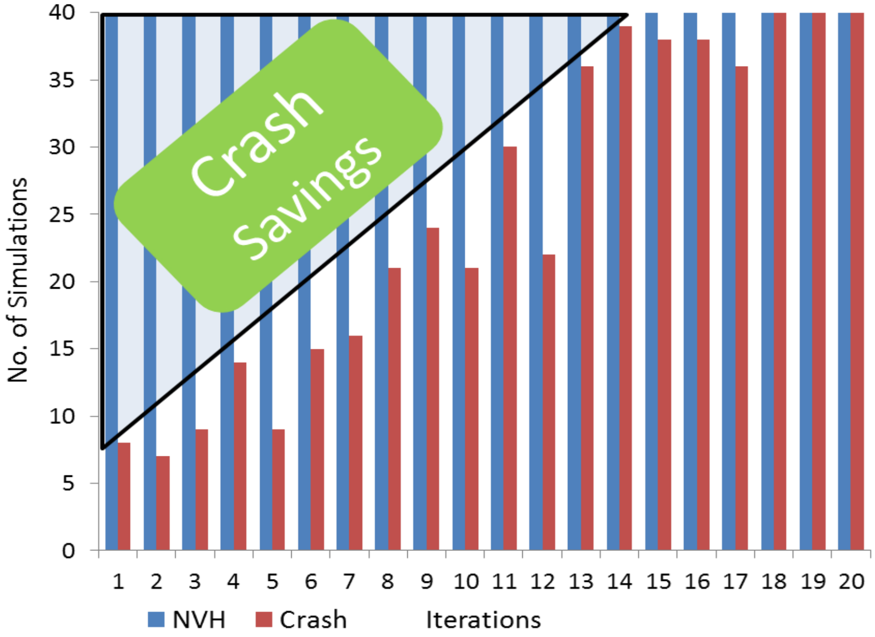


Figure 20-147: Computational cost savings with classifier-based constraints

Though the optimization setup consisted of 40 samples for NVH and crash disciplines for 20 iterations, only 503 crash simulations were analysed due to infeasible NVH design, resulting in savings of 297 simulations using the classfier-based approach.

Remarks

1. Note that a multilevel setup (LS-OPT stage type) is needed for this example only because a frequency response is used in the first stage. Frequency responses do not work as response variables unless they are extracted in an inner level.
2. As more stages are added in series, the number of samples evaluated in the latter stages is expected to decrease.
3. If the objective function is obtained as a simulation result then the corresponding solver must be run to obtain its value. E.g. if a certain injury criterion is defined as the objective function then the corresponding crash analysis will performed even if the design is known to be infeasible based on the NVH case. Otherwise the data point will not be available to improve the objective function metamodel.
4. This approach is suitable when all the disciplines use the same sampling.

20.12. REFERENCES

[1] Yamazaki, K., Han, J., Ishikawa, H., Kuroiwa, Y. Maximation of crushing energy absorption of cylindrical shells – simulation and experiment, Proceedings of the OPTI-97 Conference, Rome, Italy, September 1997.

[2] Craig K.J., Stander, N., Dooge, D., Varadappa, S. MDO of automotive vehicle for crashworthiness and NVH using response surface methods. Paper AIAA2002_5607, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 4-6 Sept 2002, Atlanta, GA.

[3] National Crash Analysis Center (NCAC). Public Finite Element Model Archive, www.ncac.gwu.edu/archives/model/index.html 2001.

[4] Akkerman, A., Thyagarajan, R., Stander, N., Burger, M., Kuhn, R., Rajic, H. Shape optimization for crashworthiness design using response surfaces. Proceedings of the 1st International Workshop on Multidisciplinary Design Optimization, Pretoria, South Africa, 8-10 August 2000, pp. 270-279.

[5] Stander, N. Goel, T. Metamodel sensitivity to sequential sampling strategies in crashworthiness design. *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada,* Sep 10-12, 2008.

[6] Stander, N., Craig, K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization, Engineering Computations, 19(4), pp. 431-450, 2002.

[7] Beta CAE Systems S.A., Tutorial Optimization - A Front Rail Crash Simulation using LS-OPT, ANSA v.14.x Tutorials, Tutorial 9.

21. Examples – Parameter Identification

21.1. Material identification (elastoplastic material) (2 variables)

A methodology for deriving system or material parameters from experimental results, known as system or parameter identification, is applied here using optimization. The example has the following features:

- The `Mean Square Error` composite is used as curve matching metric.
- The `Crossplot` history is used.
- The Min-Max formulation is demonstrated.
- Multiple test cases are employed.
- The confidence intervals of the optimal parameters are reported.

21.1.1. Problem statement

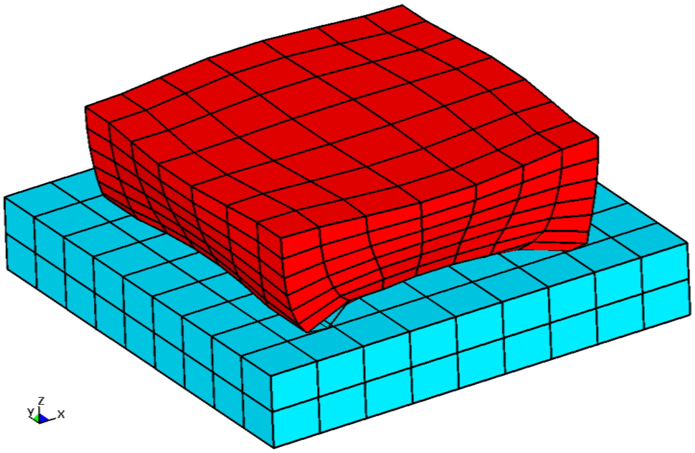


Figure 21-1: Sample of elastoplastic material subjected to a controlled vertical displacement

The material parameters of a foam material must be determined from experimental results, namely the resultant reaction force vs. displacement history of a cubic sample on a rigid base (see Figure 21-1). The problem is solved by minimizing the mean squared residual force (**`rcforc`** binary database) with the material parameters Young's modulus *E* and Yield stress *Y* as the unknown optimization variables.

The "experimental" resultant forces vs. displacements are shown below. The results were generated from an LS-DYNA run with the parameters ($E = 10^6$, $Y = 10^3$). Samples are taken at times 2, 4, 6 and 8 ms for the first load case, the test points for the second load case are taken within the linear range of force vs. deformation:

Test1.txt

```
0.36168 10162
0.72562 12964
1.0903  14840
1.4538  17672
```

Test2.txt

```
0.02272 2047
0.03671 6997
0.04653 12215
0.05779 17010
```

The finite element models for the two cases are represented in the keyword files *foam1.k* and *foam2.k* respectively.

21.1.2. Ordinate-based Curve Matching

The LS-OPT main GUI window is displayed in Figure 21-2.

The displacement and force histories are extracted from the simulation output using the `NODOUT` and `RCFORC` interfaces, respectively. Those histories are used to construct a force vs. displacement `Crossplot` for the two cases, Figure 21-3. The experimental curves used as target curves are read into LS-OPT as `File Histories`, Figure 21-4. The mean squared residual error (MSE) between each `Crossplot` and the corresponding test data is then computed. The two MSE values are simply added to find the objective value. Although only four test points are given for each case, 10 points at constant intervals are interpolated for use in the `Mean Square Error` composite, Figure 21-5 (Section **10.5.1**):

$$\varepsilon = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\boldsymbol{x}) - G_p}{s_p} \right)^2 = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{e_p(\boldsymbol{x})}{s_p} \right)^2$$

where $P = 10$, $s_p = 17672$ (Case 1), 17010 (Case 2) and $W_p = 1, p = 1, \ldots, 10$.

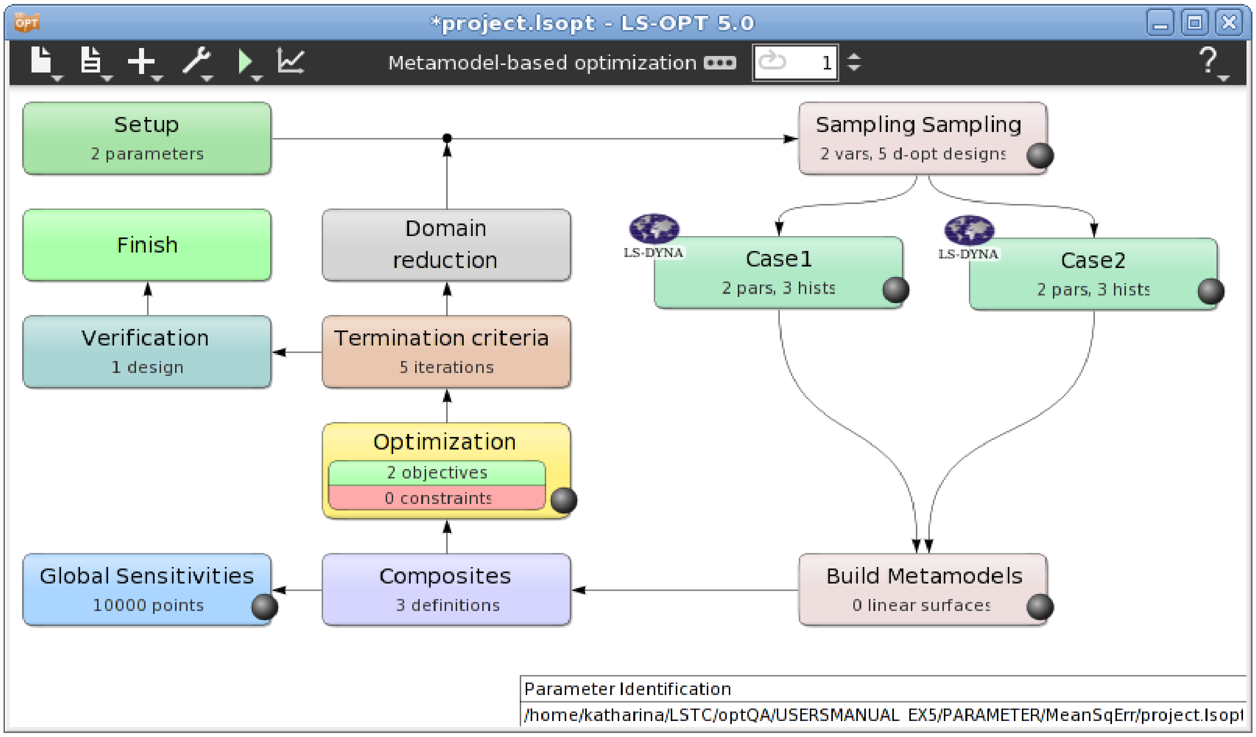
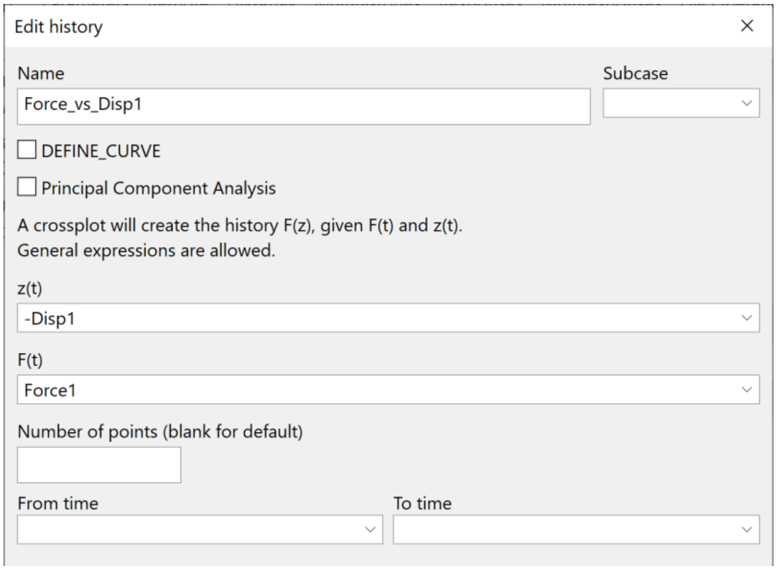


Figure 21-2: LS-OPT main GUI window



Edit history ✕

Name: Force_vs_Disp1

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t): -Disp1

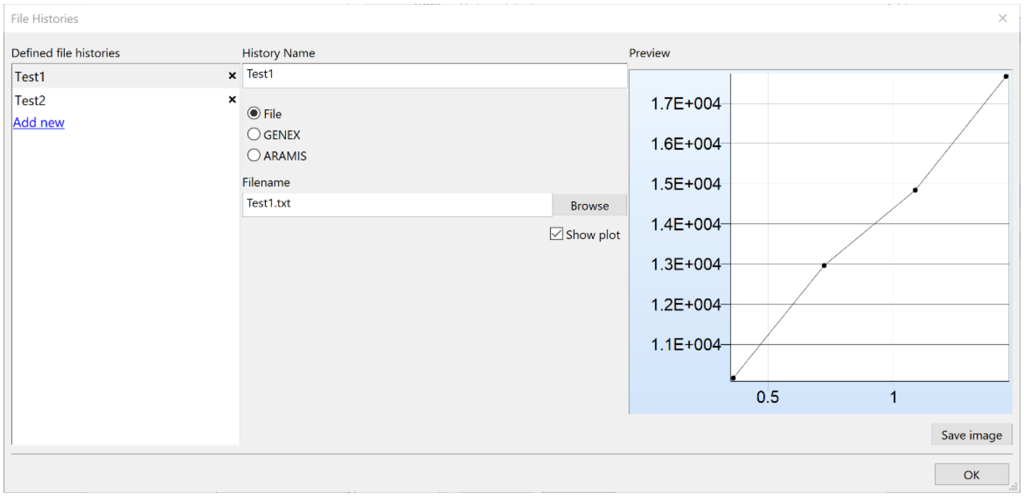
F(t): Force1

Number of points (blank for default):

From time:

To time:

Figure 21-3: Crossplot definition of force vs. displacement



File Histories

Defined file histories

Test1 ×

Test2 ×

Add new

History Name: Test1

● File

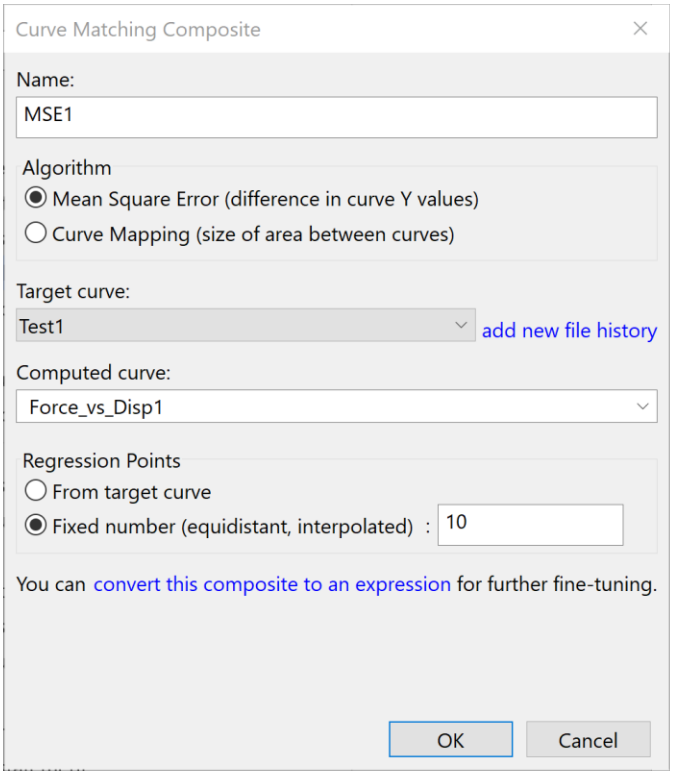
○ GENEX

○ ARAMIS

Filename: Test1.txt

☑ Show plot

Figure 21-4: File History definition. This dialog is accessible from the Histories tab of the Stage dialog or the Curve Matching composite dialog.



Curve Matching Composite

Name: MSE1

Algorithm

● Mean Square Error (difference in curve Y values)

○ Curve Mapping (size of area between curves)

Target curve: Test1

Computed curve: Force_vs_Disp1

Regression Points

○ From target curve

● Fixed number (equidistant, interpolated) : 10

You can convert this composite to an expression for further fine-tuning.

Figure 21-5: Definition of Mean Square Error composite

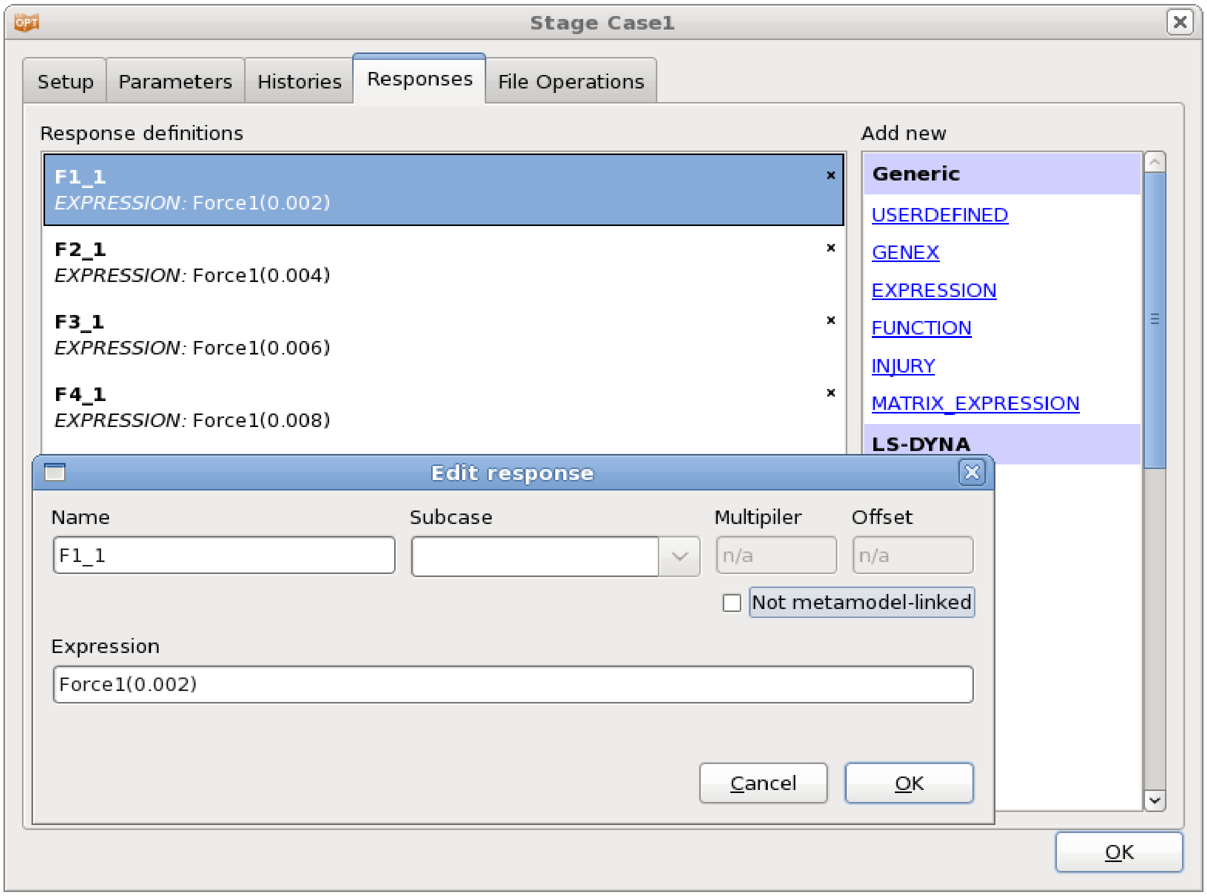
21.1.3. Targeted composite formulation

In this formulation, the force history is evaluated at specific times. The deviations from the respective target values calculated using the targeted composite formulation, so that the optimization problem for parameter identification becomes:

$$\text{Minimize } \sum_{j=1}^{7}\left[f_j(\mathbf{x}) - F_j\right],$$

where f_j are the force values evaluated from the simulation runs and F_j the respective target values.

As a method of second choice, this method presently requires a more laborious input preparation than the MSE approach. The force is evaluated using the `RCFORC` interface. This history is evaluated at the points where target values are available using the `EXPRESSION` interface, Figure 21-6. The definition of the targeted composite is displayed in Figure 21-7.



F1_1
EXPRESSION: Force1(0.002)

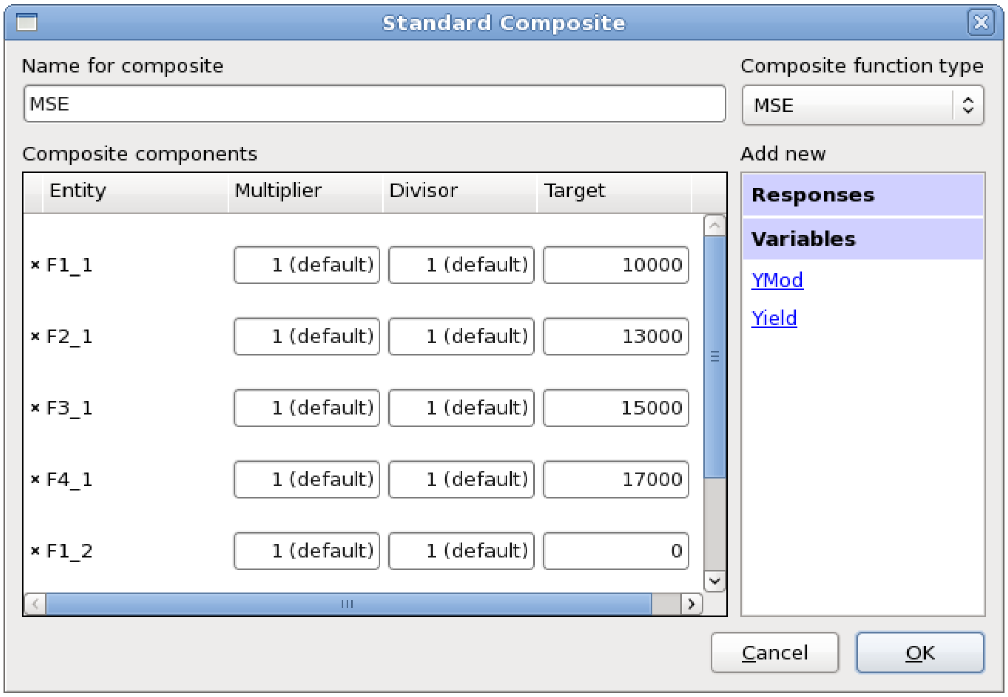
F2_1
EXPRESSION: Force1(0.004)

F3_1
EXPRESSION: Force1(0.006)

F4_1
EXPRESSION: Force1(0.008)

Name: F1_1
Subcase:
Multipiler: n/a
Offset: n/a
☐ Not metamodel-linked
Expression: Force1(0.002)

Figure 21-6: Evaluation of simulation curves at target t values



Name for composite: MSE

Composite function type: MSE

Composite components

Entity	Multiplier	Divisor	Target
× F1_1	1 (default)	1 (default)	10000
× F2_1	1 (default)	1 (default)	13000
× F3_1	1 (default)	1 (default)	15000
× F4_1	1 (default)	1 (default)	17000
× F1_2	1 (default)	1 (default)	0

Figure 21-7: Definition of equality constraints using the Standard Composite type MSE.

21.1.4. Results

The results for both methods are compared below.

21.1.5. Mean Squared Error (MSE) formulation

Figure 21-8 visualizes the optimal parameter values (red line) and the respective subregion (blue lines) over the iterations. Figure 21-9 displays the final optimal parameter values with respect to a normalized design space with 95% confidence interval. The larger confidence interval as well as the slower convergence of YMod can be explained by the insignificance of that parameter on the objective function, Figure 21-10.

Figure 21-11 displays the computed (red square) and predicted (black line) objective values over the iterations. Both objectives decrease, the quality of the predictions improves and we get convergence for both objectives.

Figure 21-12 visualizes the optimal force vs. displacement curves together with the target curves. The simulation curves are colored by iteration. There is already a good fit after the second iteration.

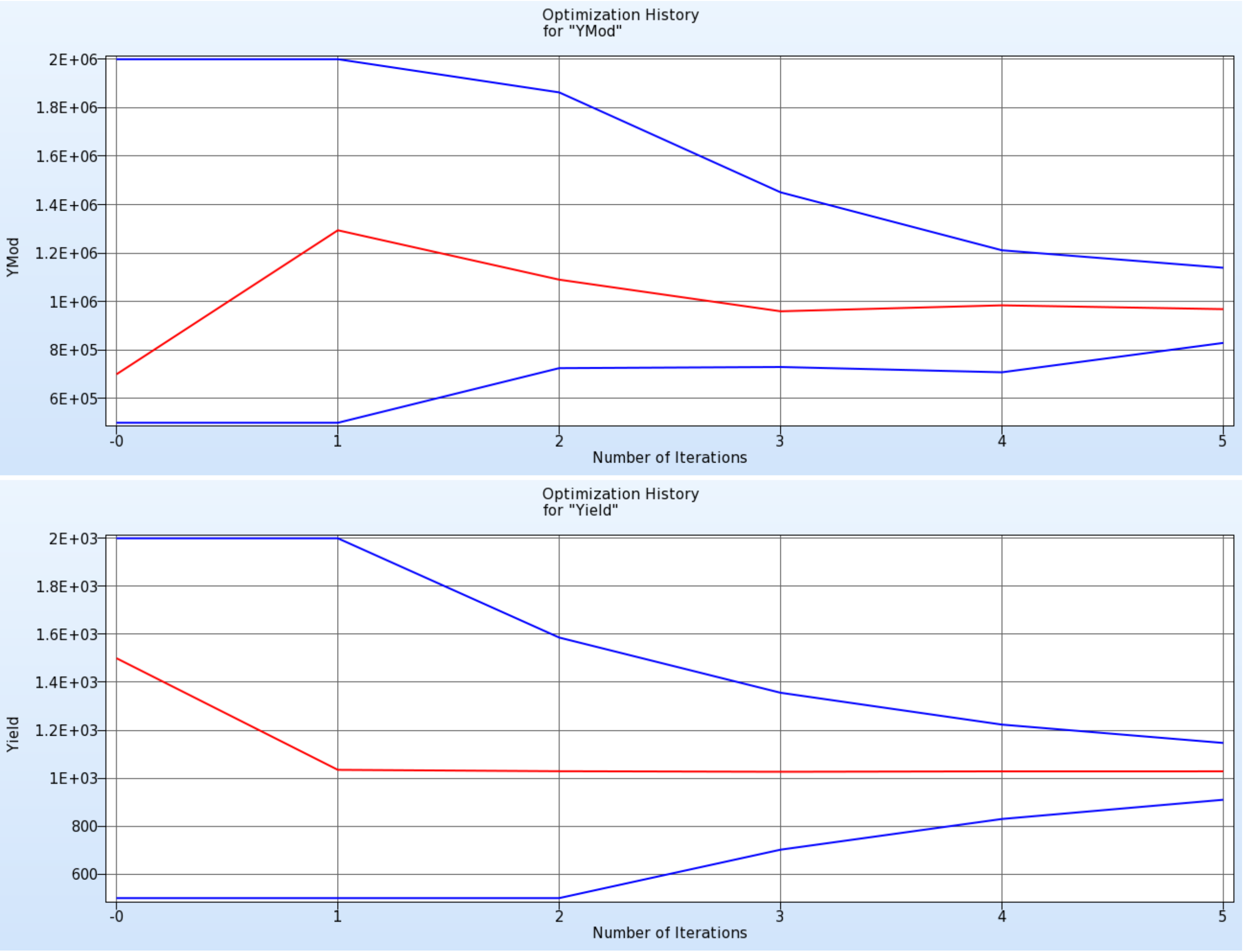


Figure 21-8: Optimization History for YMod and Yield

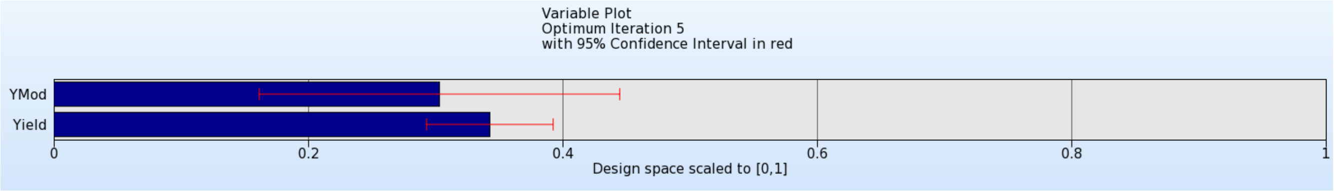


Figure 21-9: Parameter values of optimal point in normalized design space with 95% confidence interval.

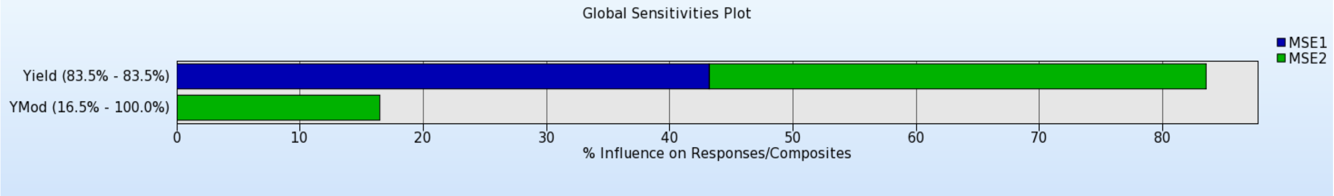


Figure 21-10: Global Sensitivities for MSE1 and MSE2

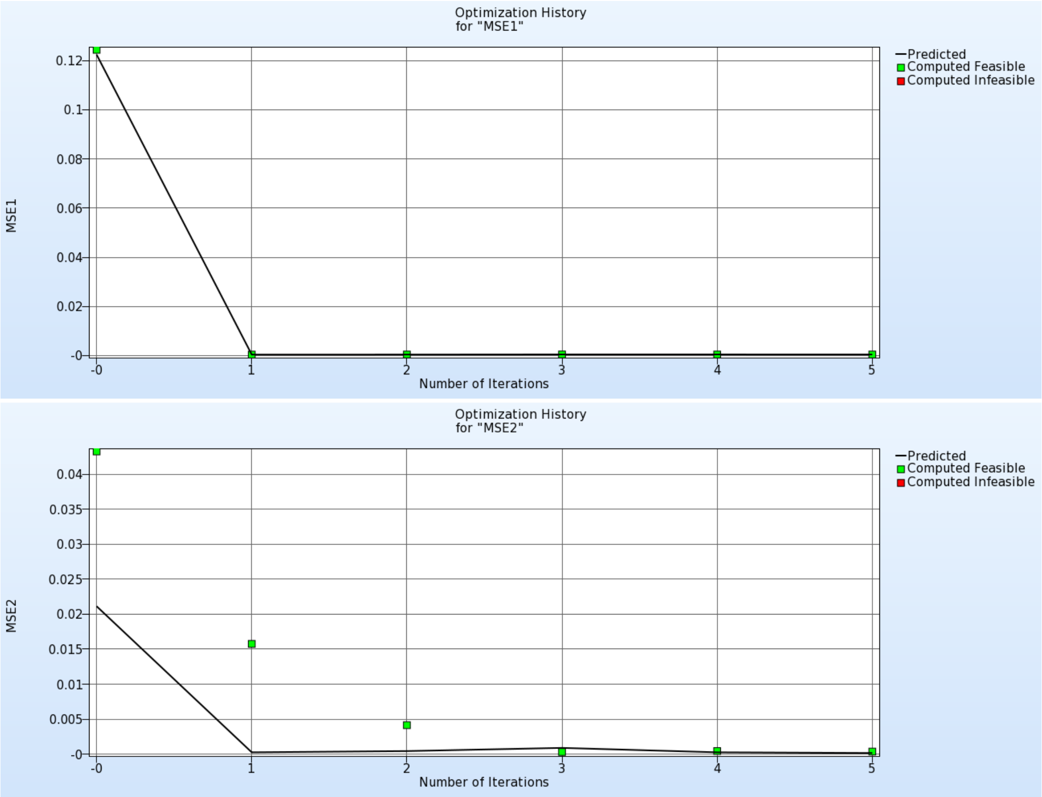


Figure 21-11: Optimization history of MSE 1 and MSE2. Both objectives decrease, and the accuracy of the metamodel improves over the iterations.

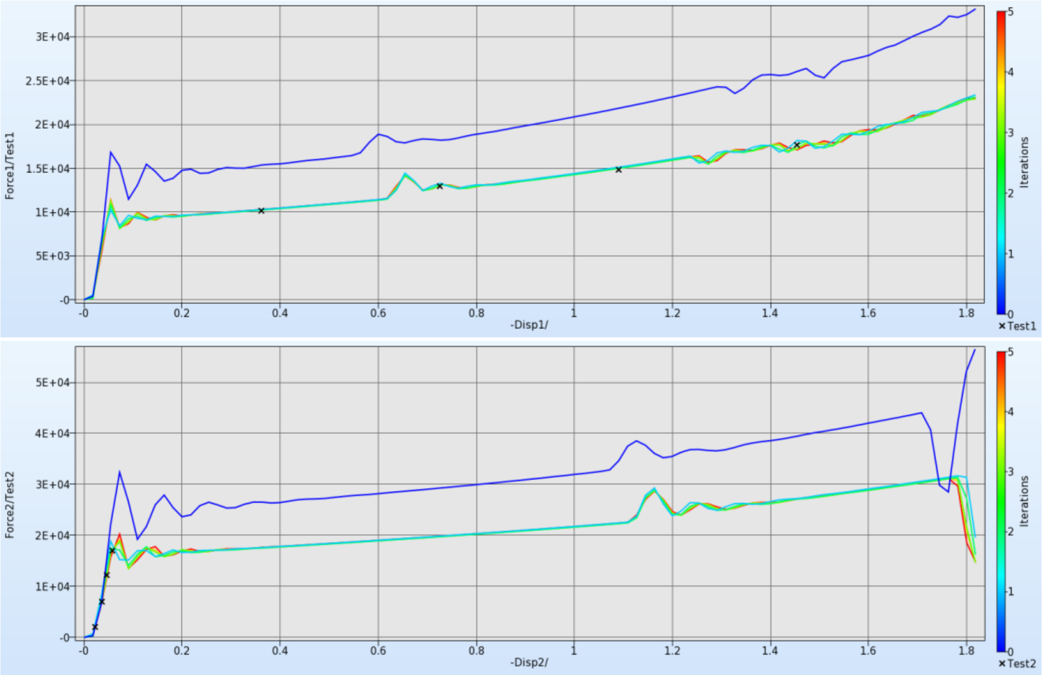


Figure 21-12: Comparison of optimal force-displacement curves and test data. The simulation curves are colored by iteration

21.1.6. Targeted composite formulation

Figure 21-13 visualizes the optimal parameter values (red line) and the respective subregion (blue lines) over the iterations. Figure 21-14 displays the final optimal parameter values with respect to a normalized design space with 95% confidence interval. The larger confidence interval as well as the slower convergence of YMod can be explained by the insignificance of that parameter on the objective function, Figure 21-15.

Note that the optimum Young's modulus differs slightly from the results obtained with the Mean Square Error approach due to its relative insignificance in the optimization as depicted in the Global Sensitivities Plot (Figure 21-15).

Figure 21-16 displays the computed (red square) and predicted (black line) objective values over the iterations. The objective decreases, the quality of the predictions improves and the objective value converges.

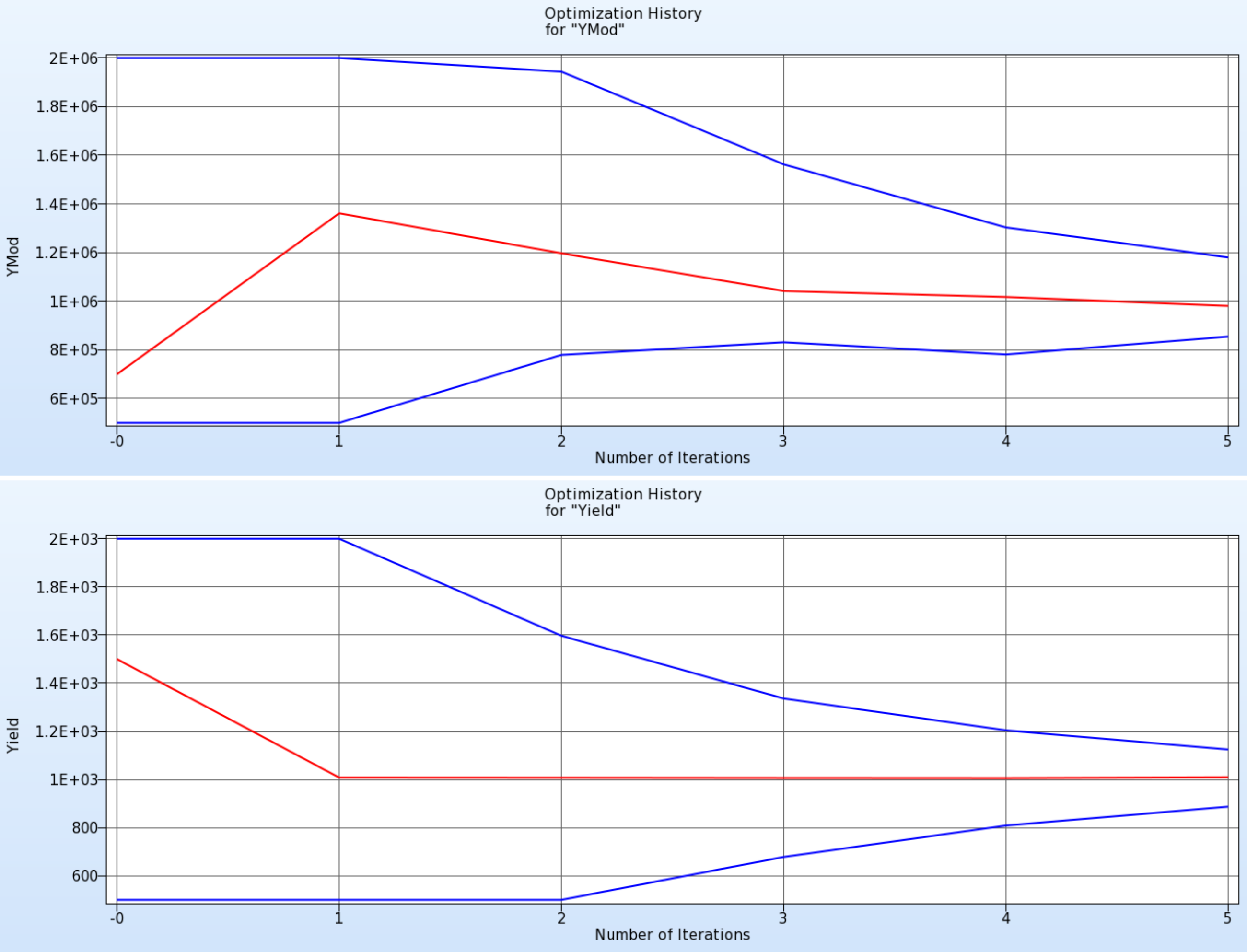


Figure 21-13: Optimization History for YMod and Yield

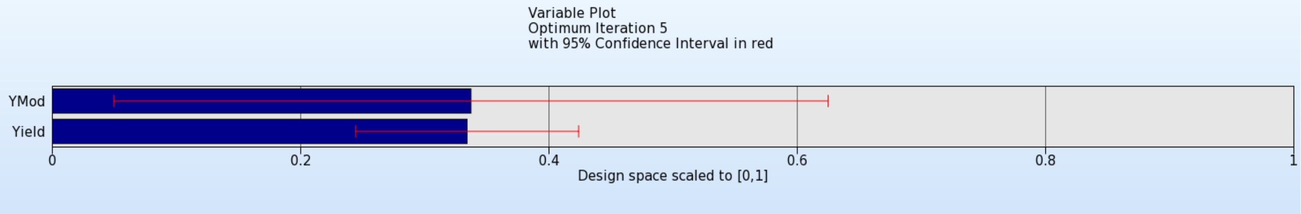


Figure 21-14: Parameter values of optimal point in normalized design space with 95% confidence interval.

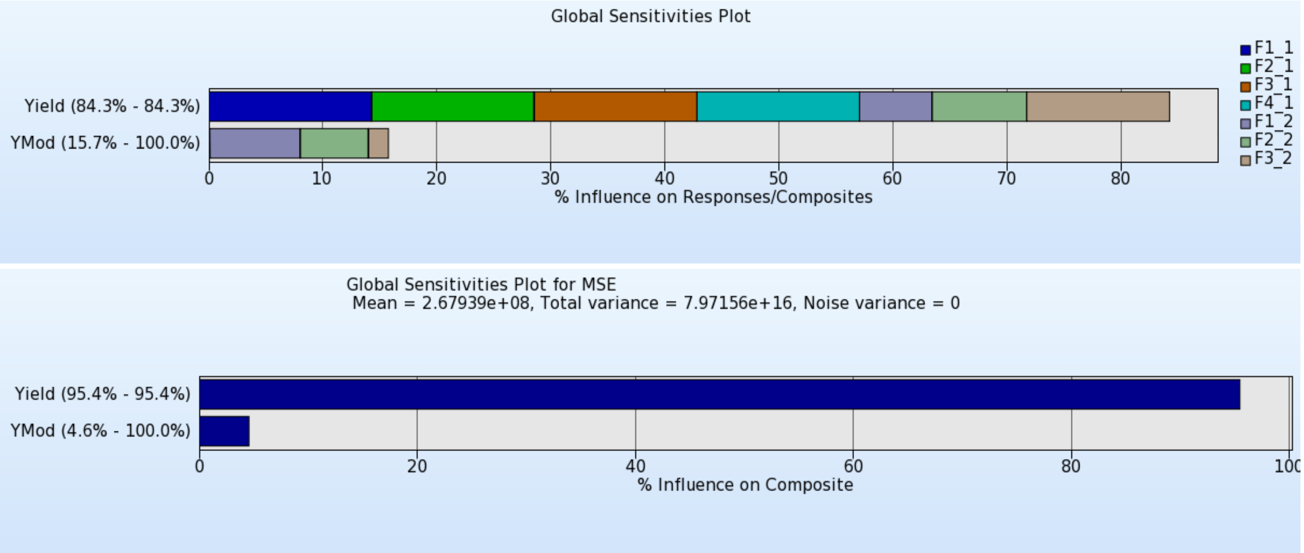


Figure 21-15: Global Sensitivities for all forces and MSE

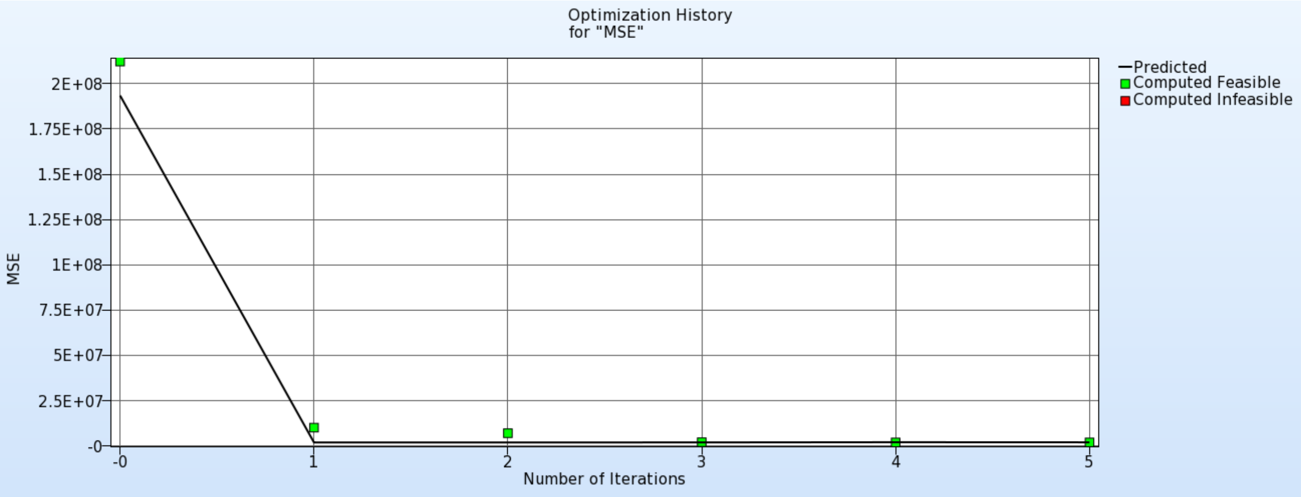


Figure 21-16: Optimization history of MSE. The objectives decrease, and the accuracy of the metamodel improves over the iterations.

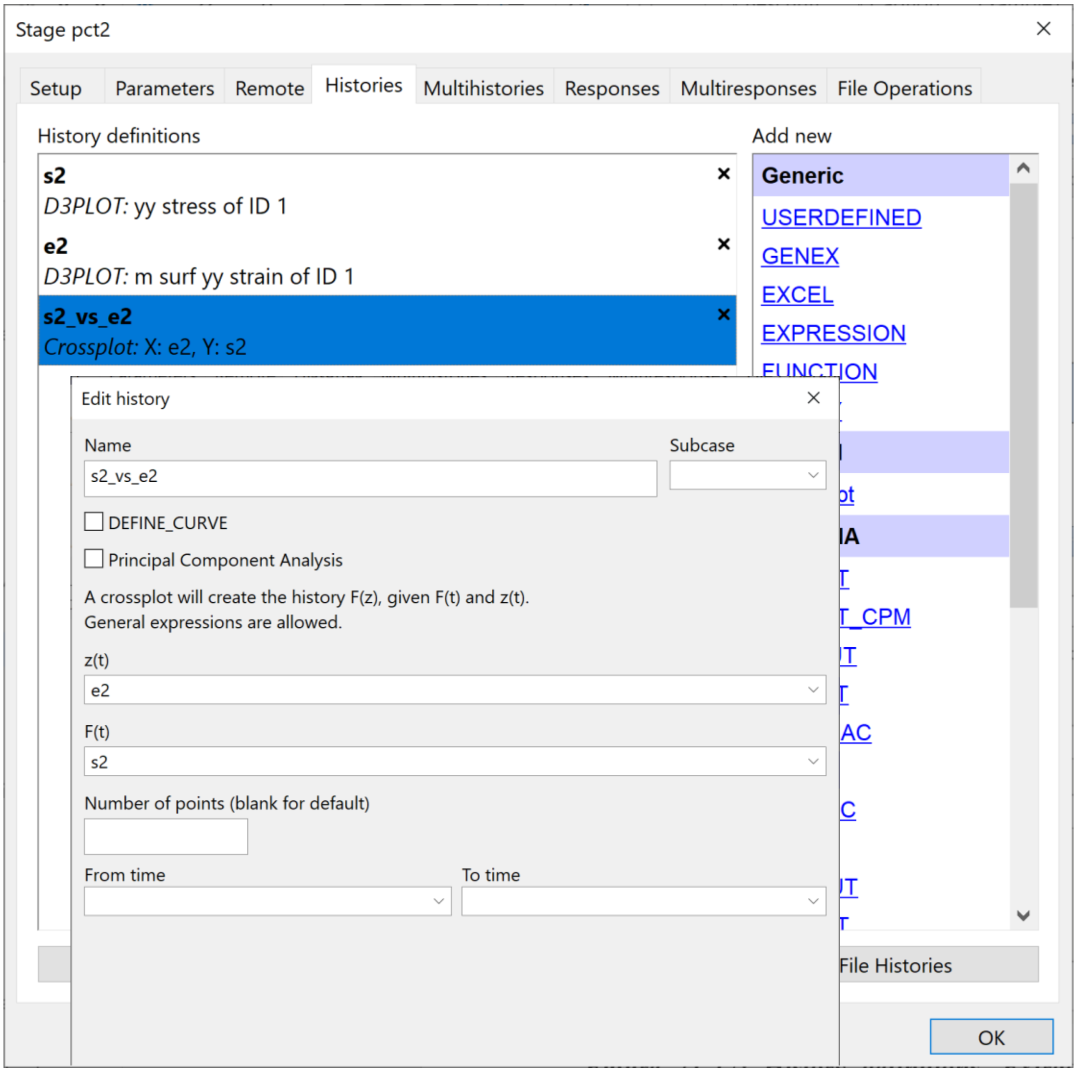
21.2. System identification with hysteretic curves

21.2.1. Problem statement

The Bauschinger effect is significant for automotive sheet steels. The phenomenon is observed under cyclic loading which results in a hysteretic stress-strain curve. The nature of the hysteretic curve complicates the curve matching required to identify the material parameters and therefore an approach which is more sophisticated than the ordinate-based matching is required. For this purpose, a Curve Mapping algorithm is used.

The following example consists of five load cases, each representing a different cyclic loading range as illustrated in the stress-strain diagram in the figure below. The material is defined by 9 parameters.

21.2.2. Solution using Curve Mapping


Stage pct2

History definitions

s2
D3PLOT: yy stress of ID 1

e2
D3PLOT: m surf yy strain of ID 1

s2_vs_e2
Crossplot: X: e2, Y: s2

Edit history

Name: s2_vs_e2

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t): e2

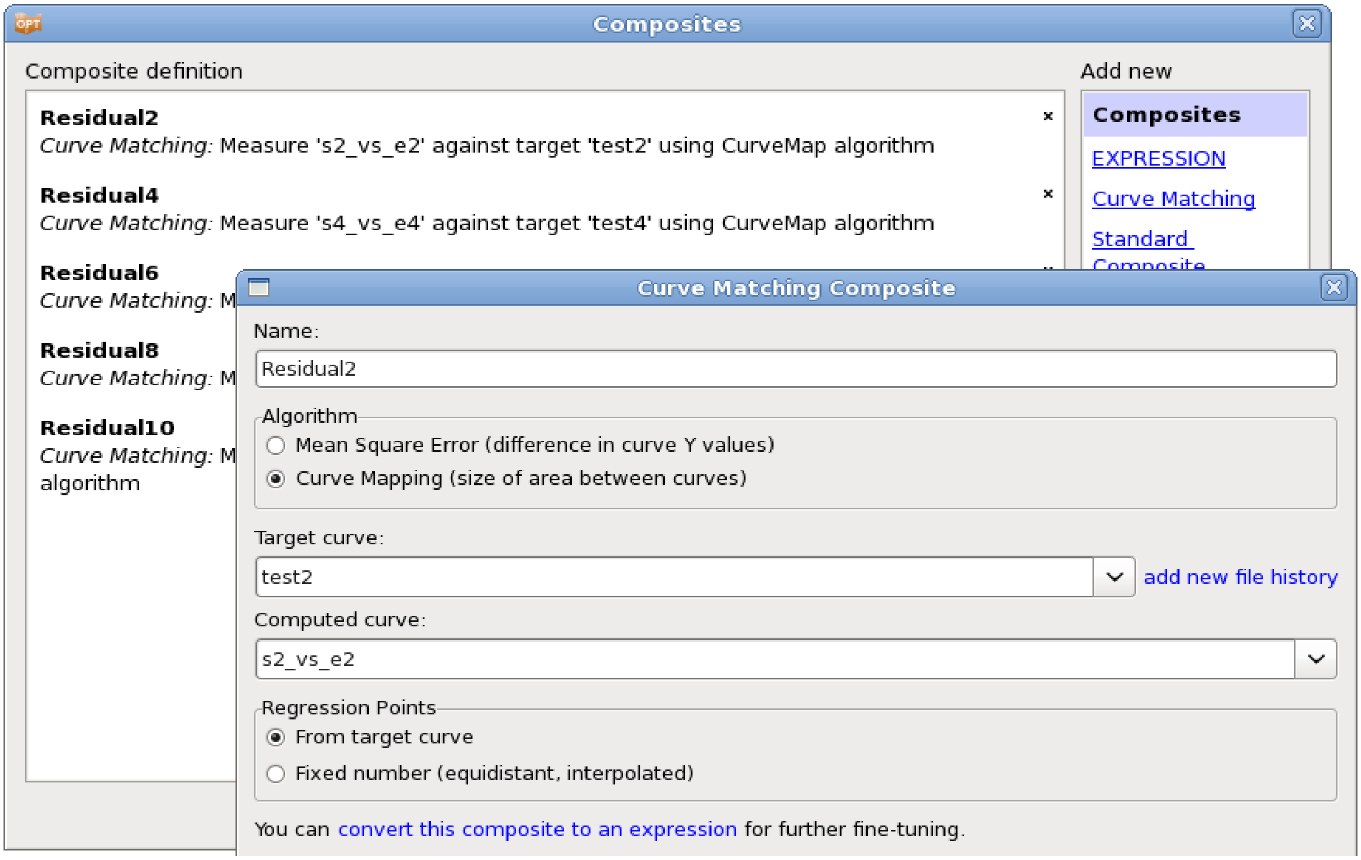
F(t): s2

Number of points (blank for default):

From time:

To time:

Figure 21-17: History definitions. Extract stress and strain using the LS-DYNA d3plot interface. Use the Crossplot interface to generate the stress vs. strain curve



Composite definition

Residual2
Curve Matching: Measure 's2_vs_e2' against target 'test2' using CurveMap algorithm

Residual4
Curve Matching: Measure 's4_vs_e4' against target 'test4' using CurveMap algorithm

Residual6
Curve Matching: M

Residual8
Curve Matching: M

Residual10
Curve Matching: M
algorithm

Name:
Residual2

Algorithm
○ Mean Square Error (difference in curve Y values)
● Curve Mapping (size of area between curves)

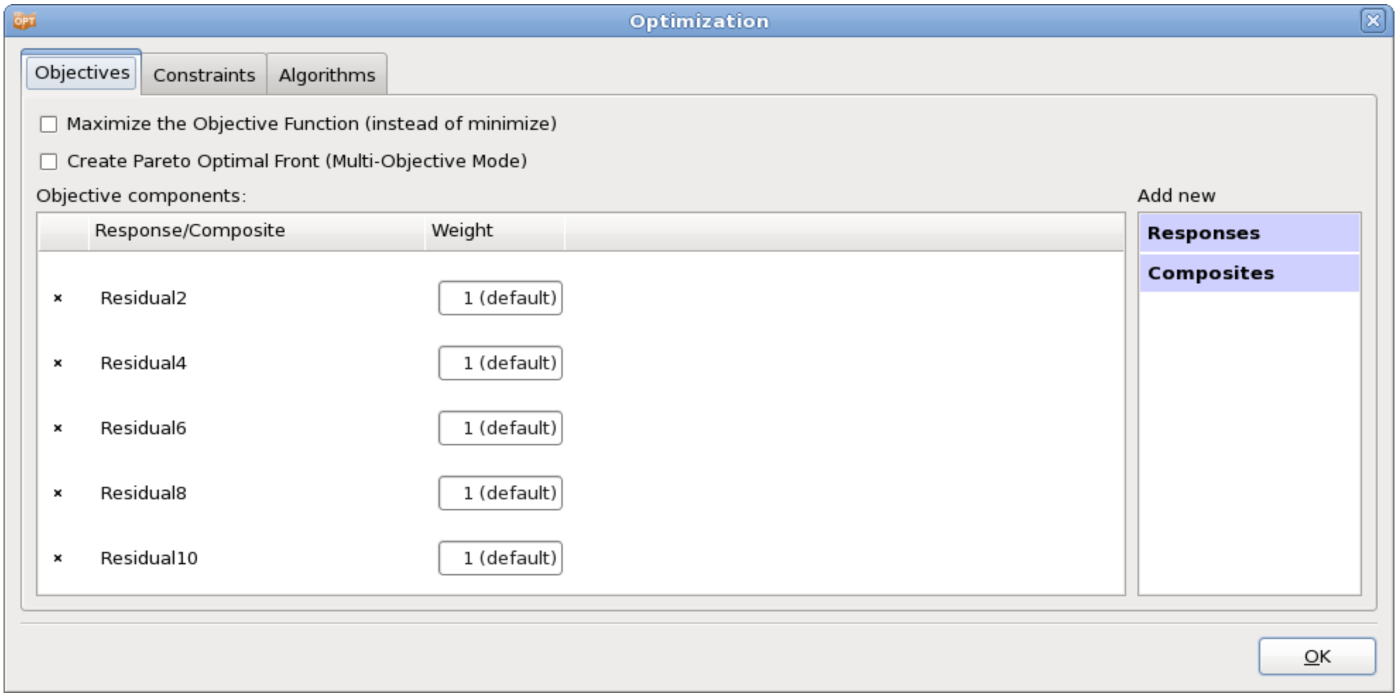
Target curve:
test2

Computed curve:
s2_vs_e2

Regression Points
● From target curve
○ Fixed number (equidistant, interpolated)

You can convert this composite to an expression for further fine-tuning.

Figure 21-18: Define a curve matching composite for each load case.



□ Maximize the Objective Function (instead of minimize)

□ Create Pareto Optimal Front (Multi-Objective Mode)

Objective components:

	Response/Composite	Weight
×	Residual2	1 (default)
×	Residual4	1 (default)
×	Residual6	1 (default)
×	Residual8	1 (default)
×	Residual10	1 (default)

Figure 21-19: Definition of the five objective components.

21.2.3. Results

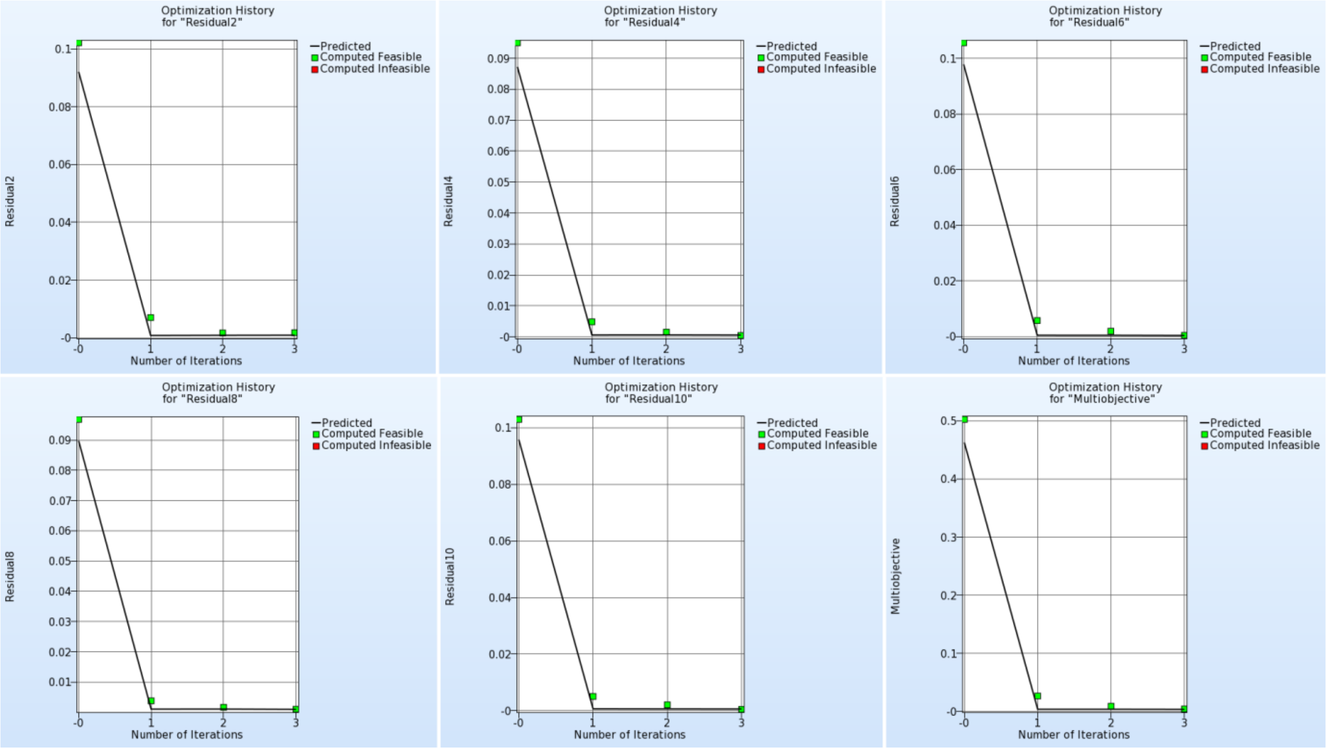


Figure 21-20: Optimization History of objective components for each of the five load cases as well as the multi-objective

Figure 21-20 displays the optimization history plots of all objective components, i.e. the curve mapping composite of each load case, as well as the optimization history of the multi-objective. For all entities, the values decrease rapidly, as the optimal values of the first iteration are already quite small.

Figure 21-21 displays the Global Sensitivities plot for the whole problem. The variable *CB* is by far the most sensitive.

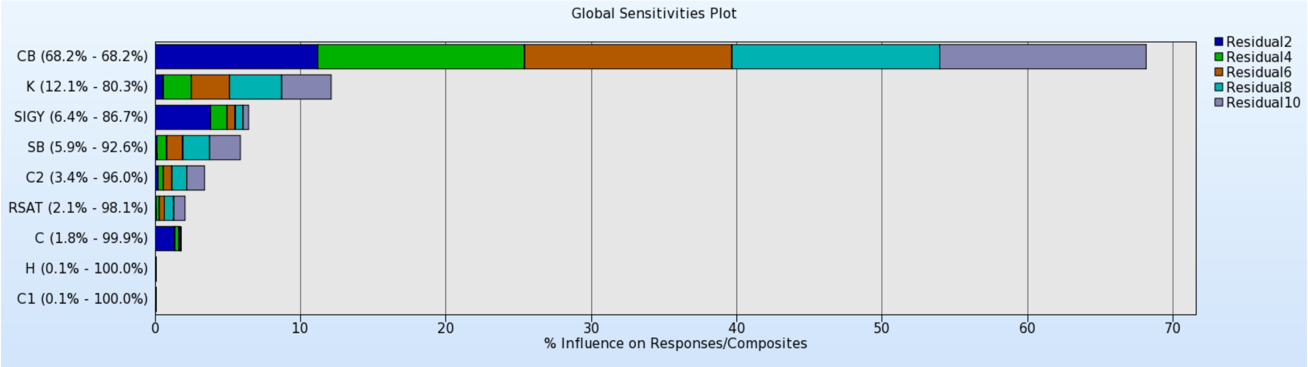


Figure 21-21: Global Sensitivities plot of all objective components

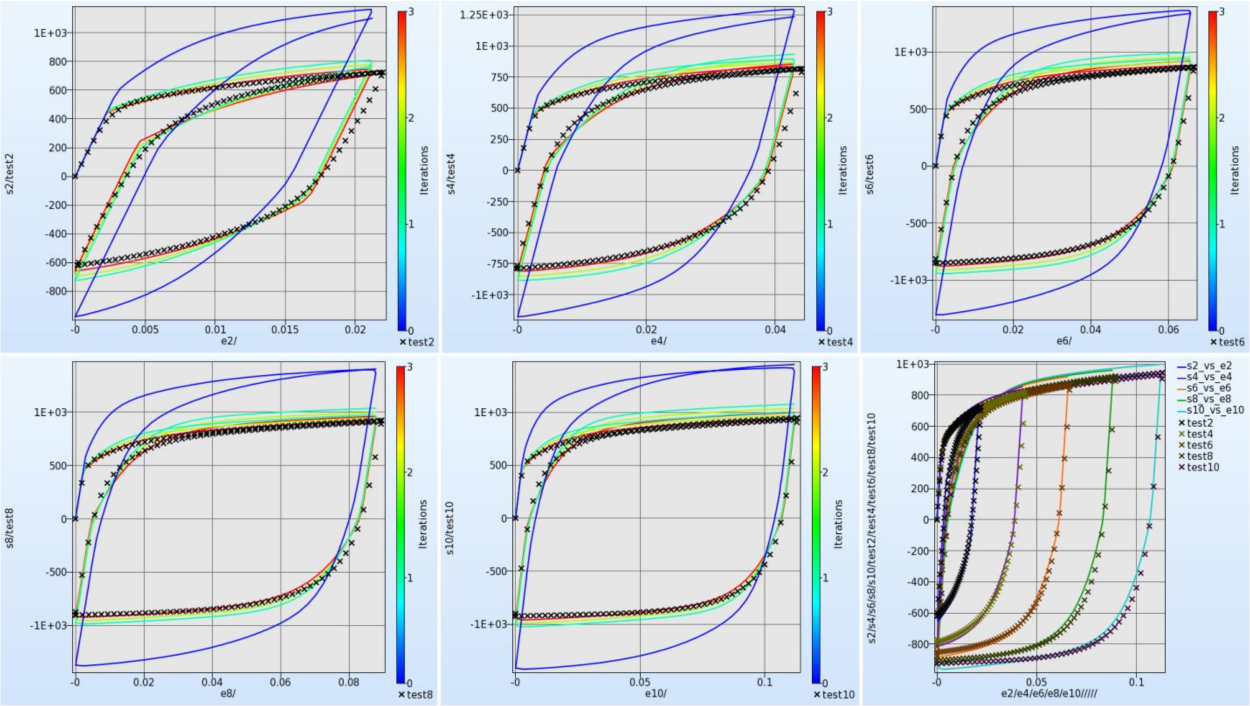


Figure 21-22: Comparison of optimal simulation curves of all iterations and target curves for all load cases; simulation curves are colored by iteration (e.g. baseline in blue. The black crosses represent the target values. The plot at bottom right shows the comparison of the final optimal simulation curves and target curves for all load cases.

Figure 21-22 visualizes the optimal simulation curves for all iterations colored by the iteration together with the target curves for all load cases. Already the optimal curves of the first iteration (turquoise) indicate a good fit. The plot at the bottom on the right shows all target curves (black crosses) and the final optimal simulation curves for all load cases.

21.3. Calibration of the GISSMO model

21.3.1. Problem statement

This example demonstrates the calibration of a GISSMO model to synthetic target curves using three test cases, Figure 21-23. The characteristics of the curves are computational noise and steep failure curves.

The parameters are 7 parameters of the GISSMO damage model.

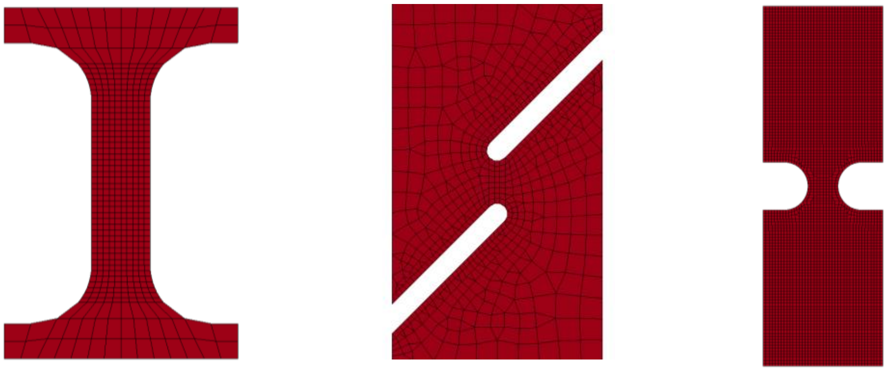
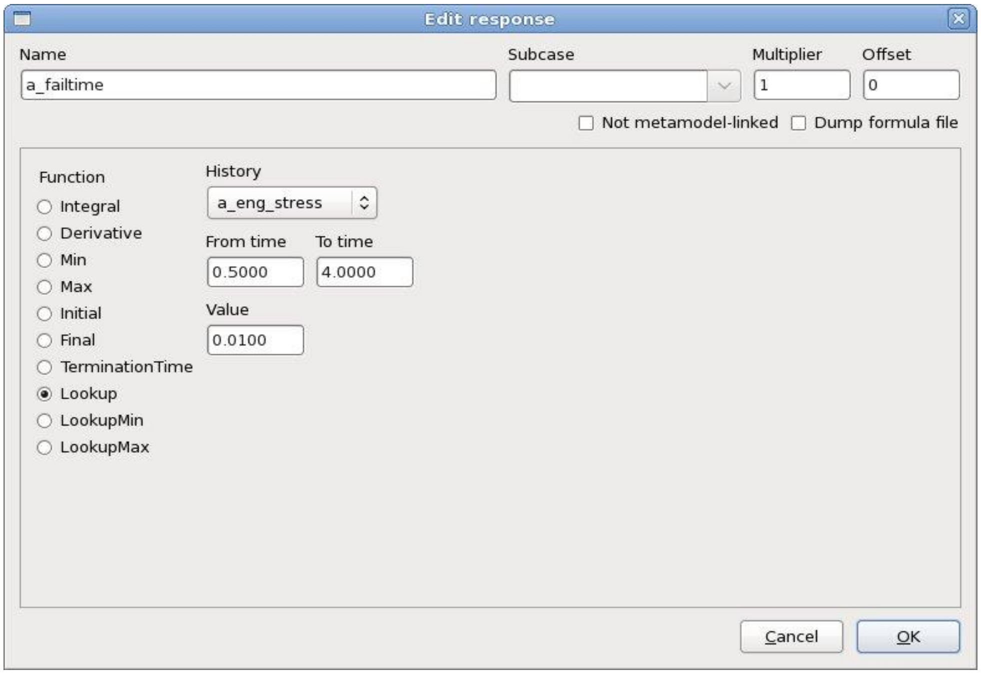


Figure 21-23: Finite Element models of tensile (left), shear (mid) and notched tensile (right) tests.

21.3.2. Solution using Dynamic Time Warping

Since the simulation curves are noisy and have a steep failure part, we use Dynamic Time Warping as a similarity measure, Figure 21-26. To make sure that the length of the target and simulation curves is similar, the force history is truncated at failure load, Figure 21-24 and Figure 21-25.

The optimization method is the Sequential Response Surface method with 15 iterations and 13 simulations per iteration per load case, so 588 simulations in total including the verification runs.



Edit response

Name: a_failtime

Subcase:

Multiplier: 1

Offset: 0

□ Not metamodel-linked □ Dump formula file

Function

○ Integral
○ Derivative
○ Min
○ Max
○ Initial
○ Final
○ TerminationTime
● Lookup
○ LookupMin
○ LookupMax

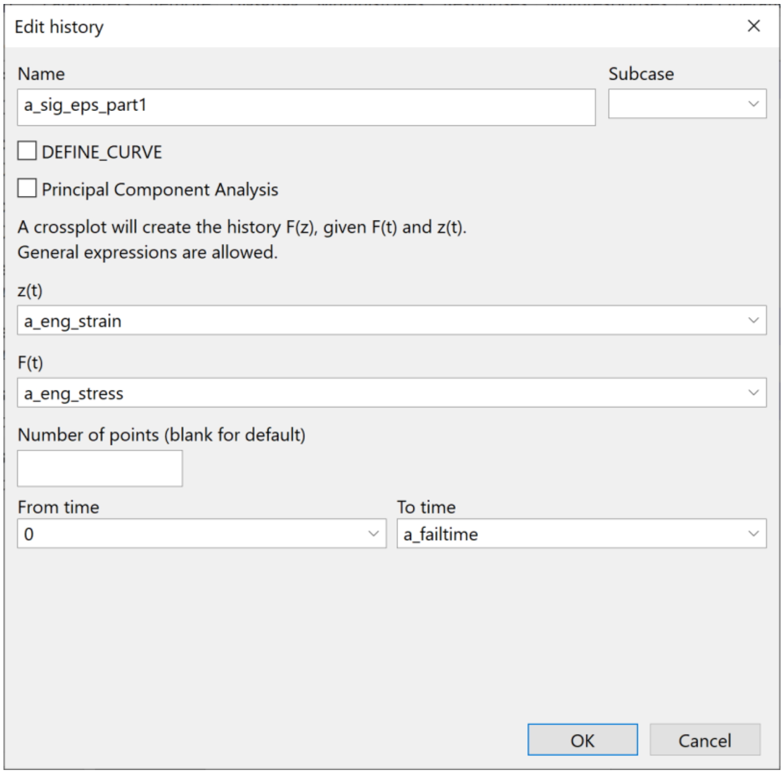
History: a_eng_stress

From time: 0.5000

To time: 4.0000

Value: 0.0100

Figure 21-24: Definition of time of failure using Lookup function



Edit history

Name: a_sig_eps_part1

Subcase:

□ DEFINE_CURVE

□ Principal Component Analysis

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t): a_eng_strain

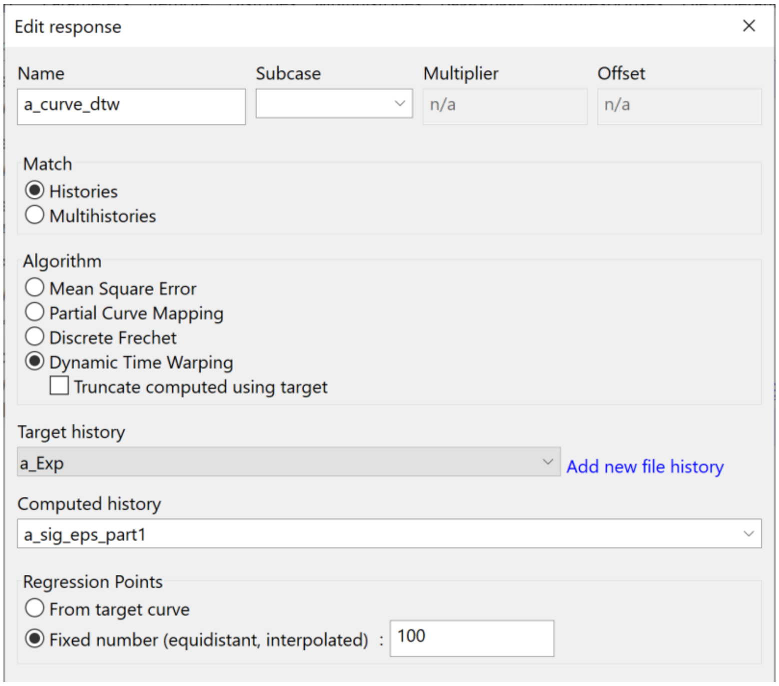
F(t): a_eng_stress

Number of points (blank for default):

From time: 0

To time: a_failtime

Figure 21-25: Definition of stress-strain curve truncated at failure time



Edit response

Name: a_curve_dtw

Subcase:

Multiplier: n/a

Offset: n/a

Match
● Histories
○ Multihistories

Algorithm
○ Mean Square Error
○ Partial Curve Mapping
○ Discrete Frechet
● Dynamic Time Warping
□ Truncate computed using target

Target history: a_Exp

Computed history: a_sig_eps_part1

Regression Points
○ From target curve
● Fixed number (equidistant, interpolated) : 100

Figure 21-26: Definition of similarity measure Dynamic Time Warping

21.3.3. Results

The Optimization histories of the objectives and variables are displayed in Figure 21-27. All DTW residuals improve significantly, the computed values get closer to the metamodel values, and all values converge.

Figure 21-28 shows the History plots of the optimal engineering stress vs. strain curves colored by iteration with the respective target curve. Figure 21-29 shows the final optimal curves compared to the target curves.

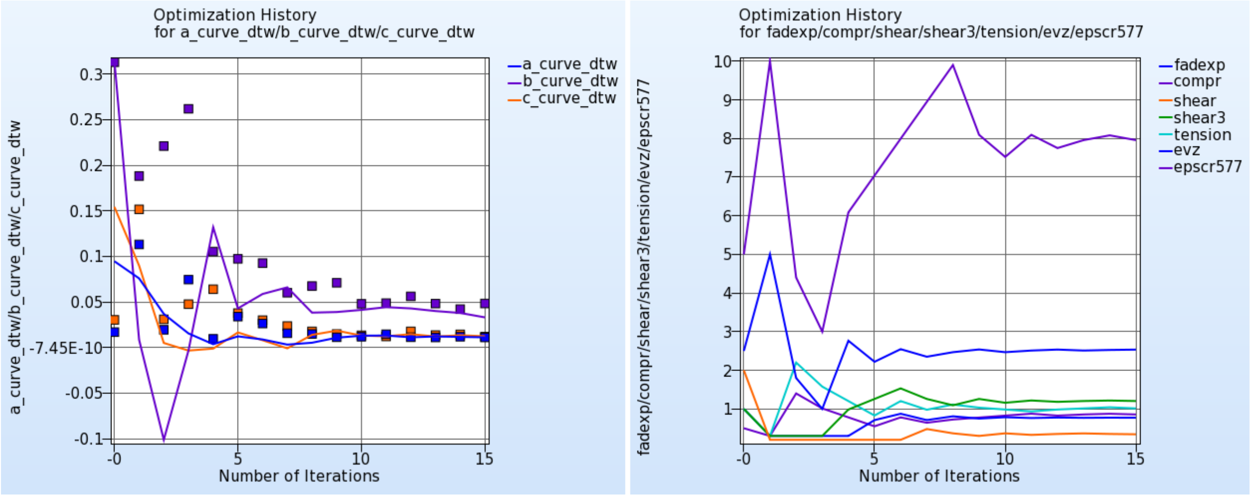


Figure 21-27: Optimization History plot of objectives (left) and variables (right).

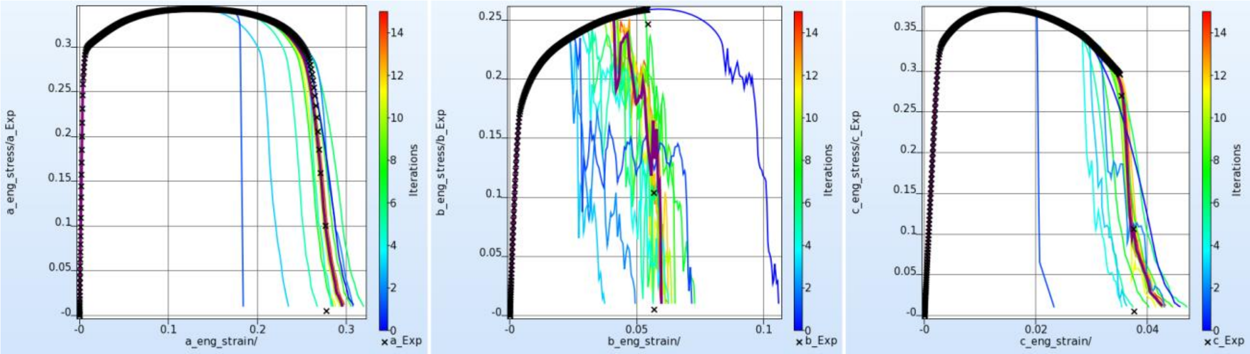


Figure 21-28: History plot displaying the optimal histories colored by iteration and the target curves. The final optimum is highlighted.

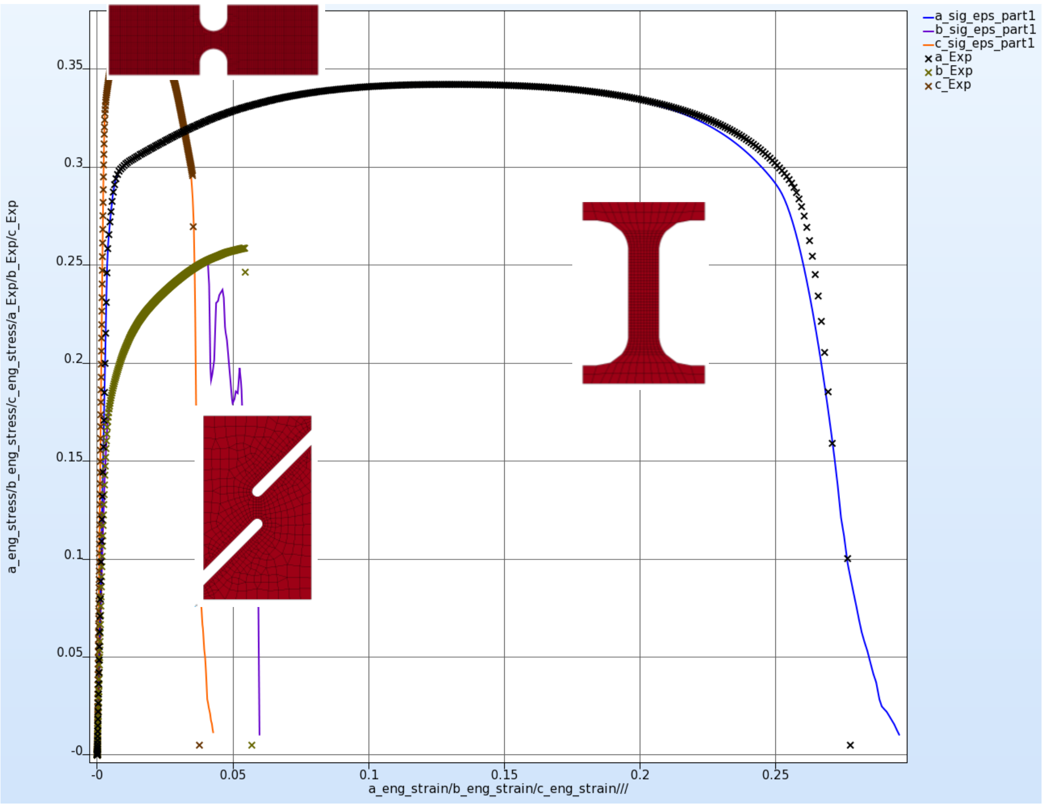


Figure 21-29: Optimal curves and target curves.

21.4. Full-field calibration using a tensile test

21.4.1. Problem statement

This example demonstrates the calibration of the yield parameters of a modified Hockett-Sherby approach to digital imaging results obtained from the optical measurement of a tensile test, (Figure 21-30). A calibration method is used in which the full ε_{xx} strain field is compared with the measured strains over the entire region of measurement. Since the experimental results are obtained both in space and time, point-wise histories are extracted from the simulation database and matched to the experimental curves using Dynamic Time Warping (DTW). The residual computed using this distance norm is minimized to obtain the four material parameters.

21.4.2. Material model

A Finite Element model was created in LS-DYNA® to model a typical coupon (Figure 21-30) subjected to a tensile force. The material model uses the von Mises yield locus and a modified Hockett-Sherby yield curve formula:

$$f(\varepsilon_p) = D + B(1 - e^{-C\varepsilon_{pl}^{N}})$$

where D, B C and N are material parameters.

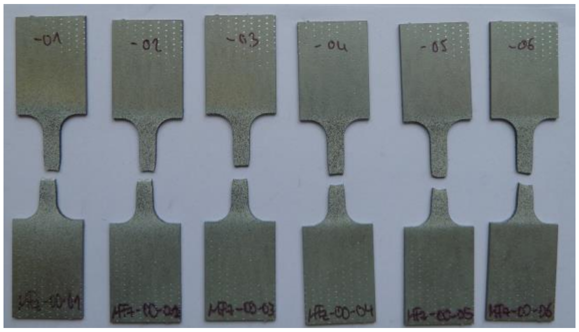


Figure 21-30: Test coupons for a tensile test (0° w.r.t. rolling direction)[19]

21.4.3. Setup

The process inside the optimization cycle consists of a pre-processing step in which the first stage defines the yield curve while the second stage simulates the tensile test using LS-DYNA (see Figure 21-31). The script executed in the first stage is shown in Figure 21-32 and Figure 21-33 shows the resulting yield curve used in the material model in the subsequent simulation.

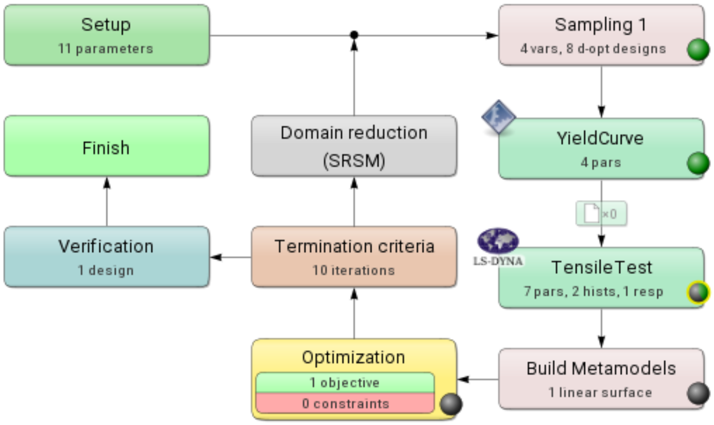


Figure 21-31: Problem setup with user defined yield curve pre-processing and tensile test simulation

```
#CREATE YIELD CURVE WITH MODIFIED HOCKETT-SHERBY
#David Koch, Christian Ilg, Pedro Roehl
#PYTHON 3.5

import sys
import csv
import math
import time

C = float('<<c>>')
N = float('<<n>>')
D = float('<<d>>')
B = float('<<b>>')
```

[19] Dynamore GmbH, Stuttgart, Germany

```
def getYieldCurve():
      yieldcurve_eps = []
      yieldcurve_sig = []
      SamplePoints = 300
      stepsize = 2.0 / (SamplePoints-1)
      for i in range(SamplePoints):
           val = i*stepsize
           yieldcurve_eps.append(val)
           yieldcurve_sig.append(D+B*(1-math.exp(-C*val**N)))
      return yieldcurve_eps, yieldcurve_sig

yieldcurve_eps, yieldcurve_sig = getYieldCurve()

ycfile = 'yield_curve.inc'
f_yc = open(ycfile, 'w')
f_yc.write('*KEYWORD\n')
f_yc.write('$ Created: ' + time.strftime("%d.%m.%Y %H:%M:%S") + '\n')
f_yc.write('*DEFINE_CURVE_TITLE\n')
f_yc.write('yield curve\n')
f_yc.write('$     LCID      SIDR       SFA       SFO      OFFA      OFFO
DATTYP\n')
f_yc.write('       500         0    1.0000    1.0000    0.0000    0.0000\n')
f_yc.write('%20.8e %19.8e\n' %(0.0,  D))
for i in range(len(yieldcurve_eps)):
      if yieldcurve_sig[i] > D:
           f_yc.write('%20.8e %19.8e\n' %(yieldcurve_eps[i],
yieldcurve_sig[i]))
f_yc.close()

print("N o r m a l")
```

Figure 21-32: Script for generating the yield curve as a function of material parameters C, N, D and B. The curve is dumped to an LS-DYNA include file under the *DEFINE_CURVE keyword.

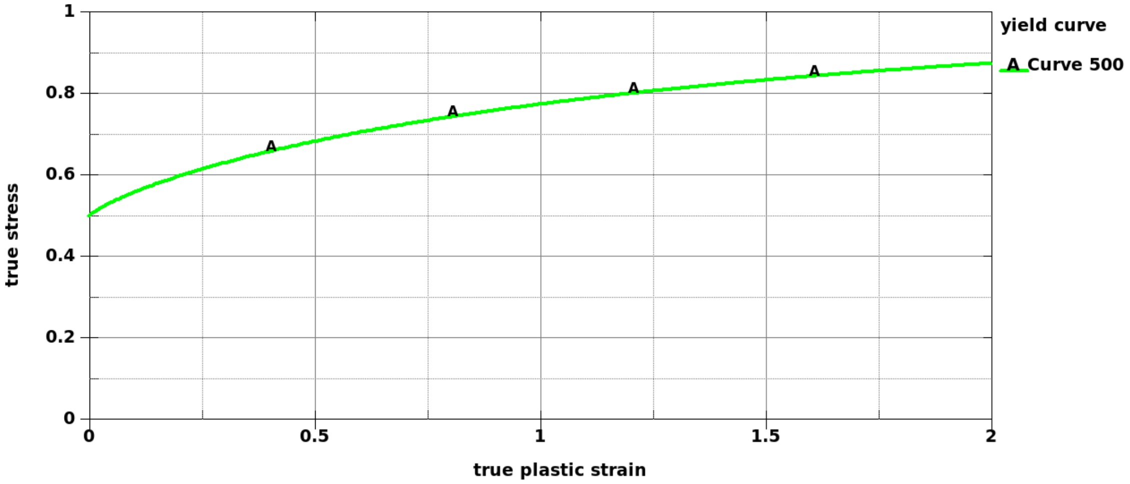
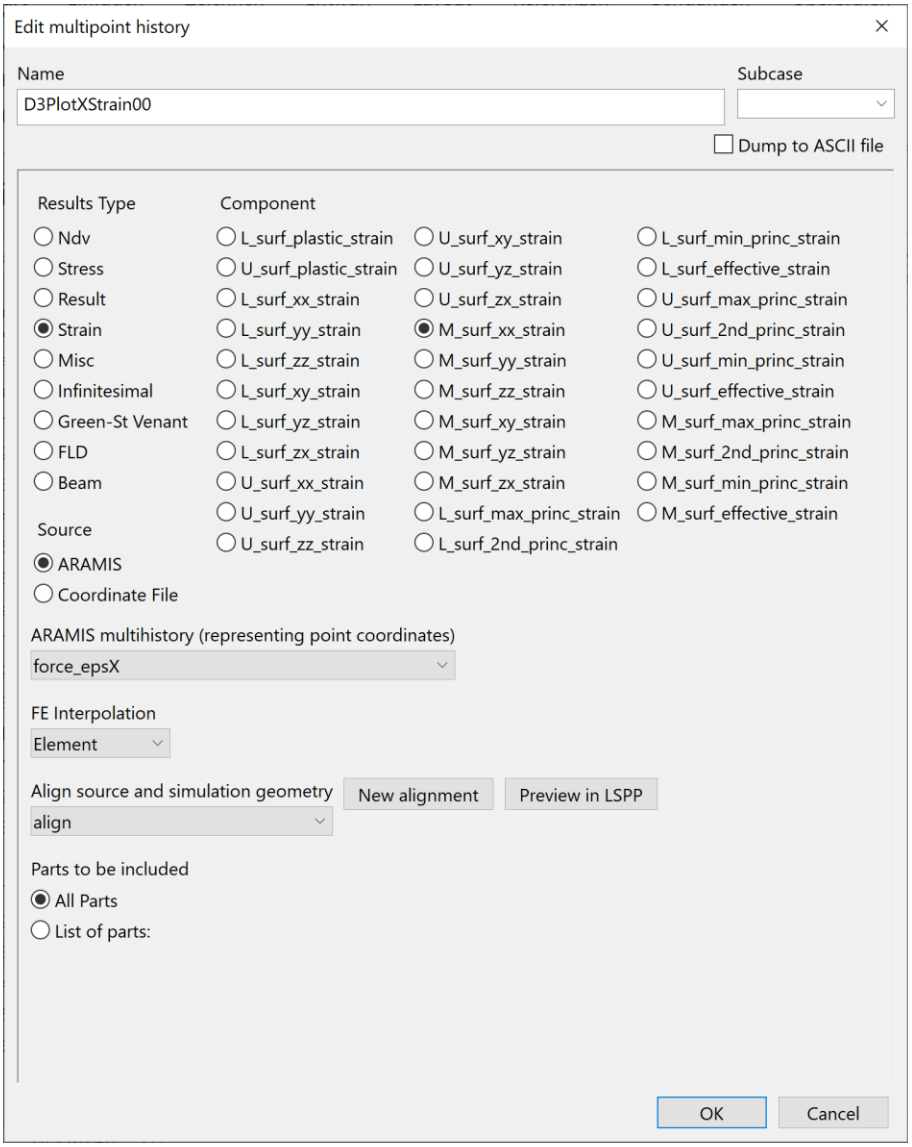


Figure 21-33: Yield curve.



Edit multipoint history

Name: D3PlotXStrain00

Subcase:

□ Dump to ASCII file

Results Type

○ Ndv
○ Stress
○ Result
● Strain
○ Misc
○ Infinitesimal
○ Green-St Venant
○ FLD
○ Beam

Component

○ L_surf_plastic_strain
○ U_surf_plastic_strain
○ L_surf_xx_strain
○ L_surf_yy_strain
○ L_surf_zz_strain
○ L_surf_xy_strain
○ L_surf_yz_strain
○ L_surf_zx_strain
○ U_surf_xx_strain
○ U_surf_yy_strain
○ U_surf_zz_strain
○ U_surf_xy_strain
○ U_surf_yz_strain
○ U_surf_zx_strain
● M_surf_xx_strain
○ M_surf_yy_strain
○ M_surf_zz_strain
○ M_surf_xy_strain
○ M_surf_yz_strain
○ M_surf_zx_strain
○ L_surf_max_princ_strain
○ L_surf_2nd_princ_strain
○ L_surf_min_princ_strain
○ L_surf_effective_strain
○ U_surf_max_princ_strain
○ U_surf_2nd_princ_strain
○ U_surf_min_princ_strain
○ U_surf_effective_strain
○ M_surf_max_princ_strain
○ M_surf_2nd_princ_strain
○ M_surf_min_princ_strain
○ M_surf_effective_strain

Source

● ARAMIS
○ Coordinate File

ARAMIS multihistory (representing point coordinates): force_epsX

FE Interpolation: Element

Align source and simulation geometry: align

Parts to be included

● All Parts
○ List of parts:

Figure 21-34: Multi-point history dialog showing strain extraction attributes. Note that the FE interpolation is within the element. A previewing option can be selected using the 'Preview in LSPP' button.

Since the test point set was initially misaligned with the FE model, an alignment using 3 points was specified. In this case, three FE nodes were aligned with three point coordinate sets (see Figure 21-35 and Figure 21-36). The alignment can be previewed (see Figure 21-38). Finally, a crossplot is created of the global force vs. the ε_{xx} strain field (see Figure 21-37).

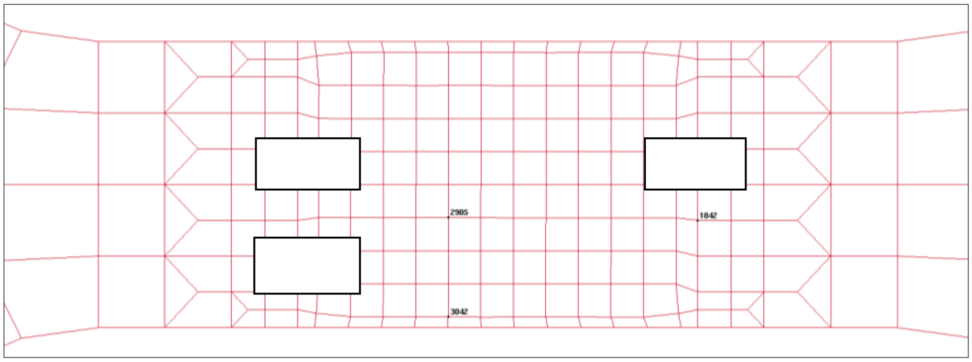
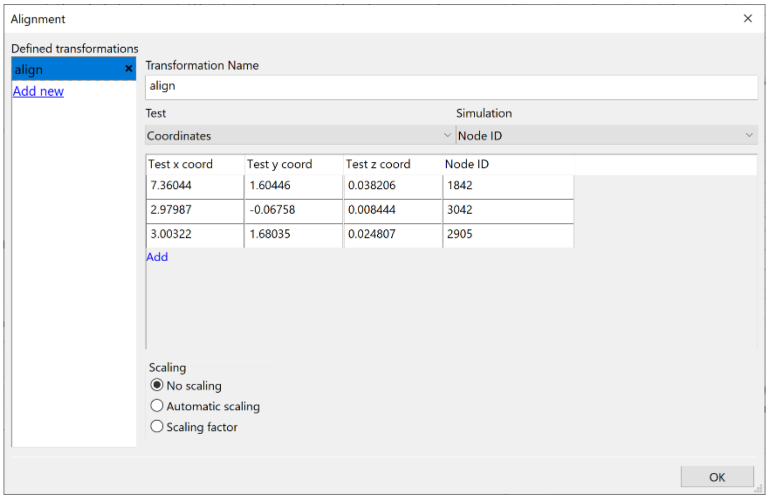


Figure 21-35: Three alignment points on FE mesh



Alignment

Defined transformations

align

Add new

Transformation Name

align

Test	Simulation
Coordinates	Node ID

Test x coord	Test y coord	Test z coord	Node ID
7.36044	1.60446	0.038206	1842
2.97987	-0.06758	0.008444	3042
3.00322	1.68035	0.024807	2905

Add

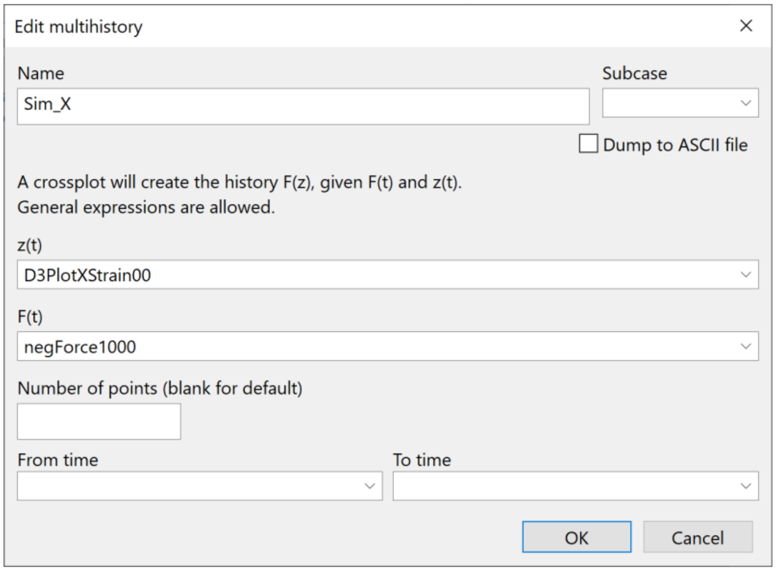
Scaling

● No scaling

○ Automatic scaling

○ Scaling factor

Figure 21-36: Alignment data using 3 points and nodes with IDs 1842, 3042 and 2905. In this case the corresponding source point coordinates were specified.



Edit multihistory

Name: Sim_X

Subcase:

□ Dump to ASCII file

A crossplot will create the history F(z), given F(t) and z(t).
General expressions are allowed.

z(t): D3PlotXStrain00

F(t): negForce1000

Number of points (blank for default):

From time:

To time:

Figure 21-37: Crossplot definition of force vs. full-field strain.

21.4.4. Calibration using DIC data and Dynamic Time Warping

The material model was calibrated using the DIC results of a laboratory experiment[20]. The LS-OPT setup required alignment of 463 measured points with the FE mesh as shown in Figure 21-38. The force vs. point-wise ε_{xx} strain curves are shown in the previewing feature depicted in Figure 21-39. Each curve represents a measuring point as shown in black in Figure 21-38. The use of the force ensures that both statics and kinematics are incorporated into the calibration model.

The optimization problem was set up with the 4 parameters C, N, D and B as variables and with the objective to minimize the DTW distance function f. This defines the ideal of matching ε_{xx} at each spatial point as well as each deformation state. The optimization was conducted using the Sequential Response Surface Method using default settings. Ten iterations were run.

Real experiments present additional challenges to parameter identification implying that the choice of similarity measure is important. In this case, the Dynamic Time Warping method (see the theory in Section 28.2.3 as well as the example of Section 21.3.2) was chosen for the calibration. The Euclidean distance method was considered unsuitable because the vertical drop-off of a large number of curves in the post-yield phase defies the use of an ordinate-based distance measurement (see also Figure 21-39).

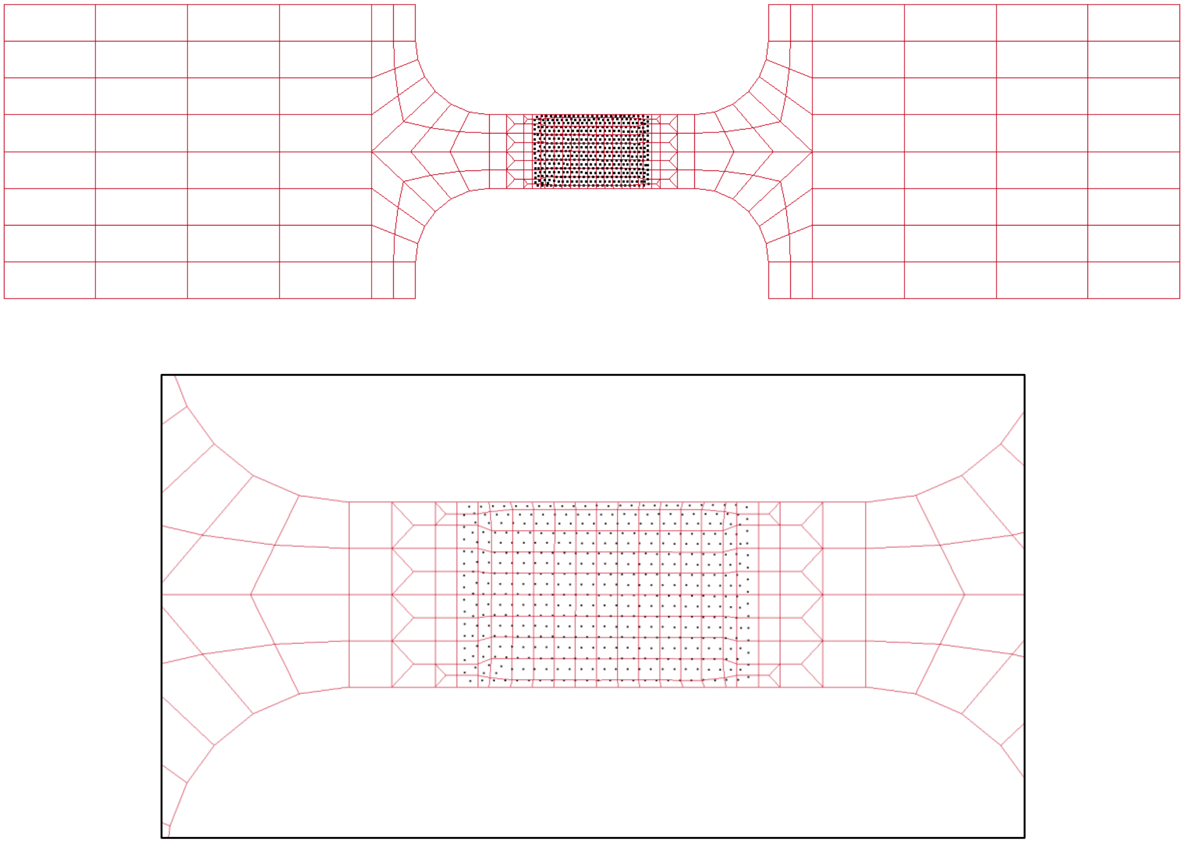
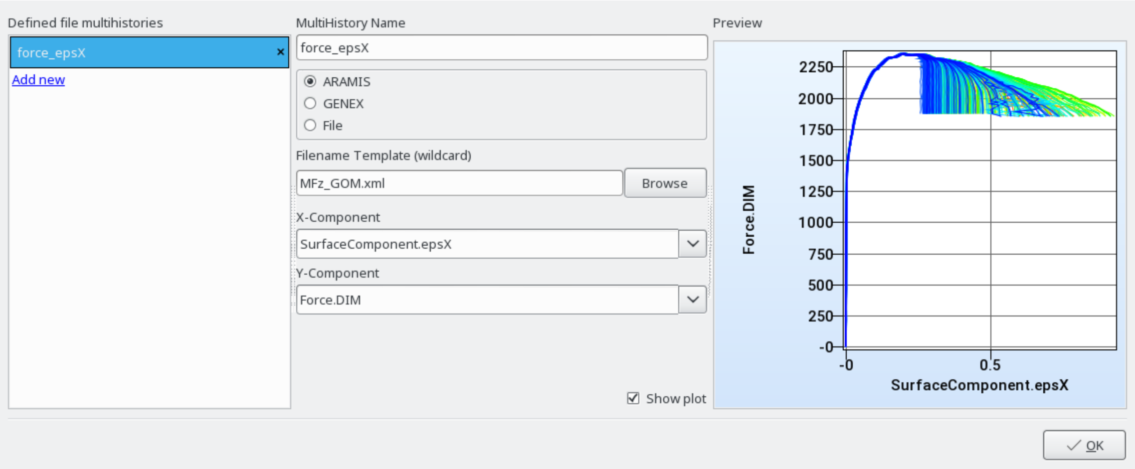


Figure 21-38: Preview of test points superimposed on the Finite Element model (zoomed-in shown at bottom)

[20] Provided by Dynamore GmbH, Stuttgart, Germany



Defined file multihistories

force_epsX ×

Add new

MultiHistory Name: force_epsX

● ARAMIS
○ GENEX
○ File

Filename Template (wildcard): MFz_GOM.xml Browse

X-Component: SurfaceComponent.epsX

Y-Component: Force.DIM

☑ Show plot

✓ OK

Figure 21-39: Preview of the experimental data (force vs. full-field strain) in the file multihistory dialog. This example uses a dedicated interface to the gom/ARAMIS database. Free format text parsing (GENEX) and fixed format (File) options are also available. A single curve represents a spatial point as shown in Figure 21-38.

21.4.5. Results

The results are presented in Figure 21-40 (optimization history of the residual), Figure 21-41 (variable values at the baseline and optimum) and Figure 21-42 (contour plots of the matching strains). The contour plots can be selected from the Viewer after selecting a simulation point from a plot or table.

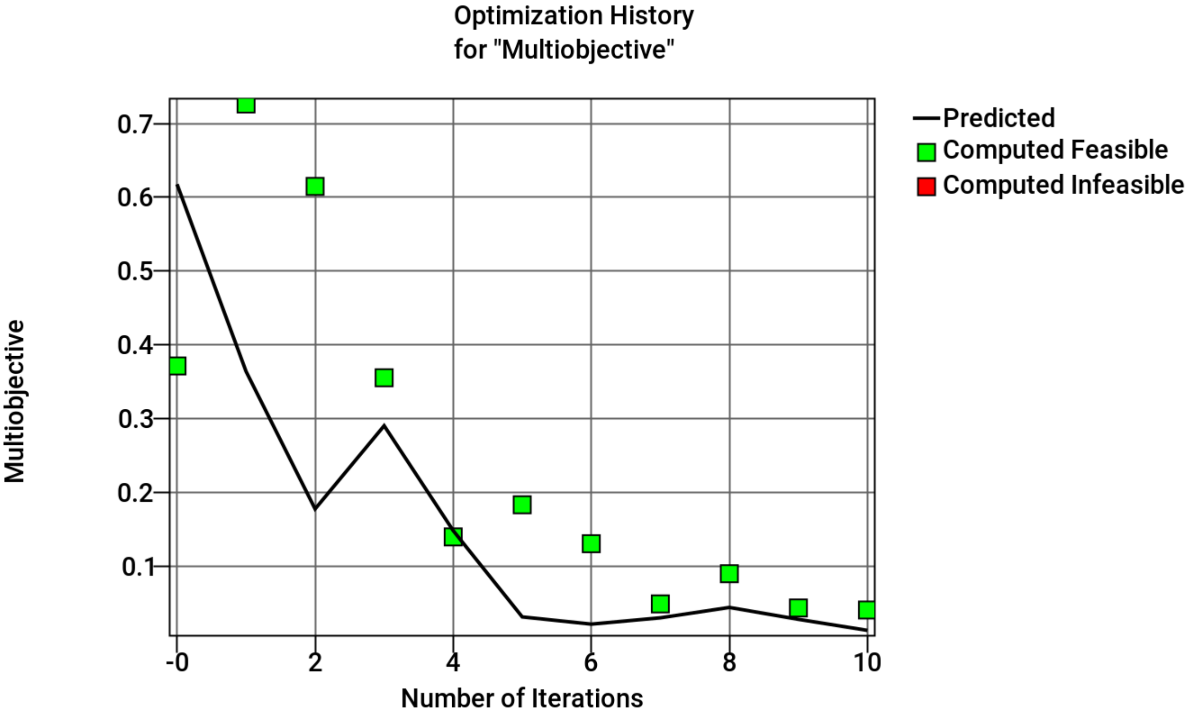


Figure 21-40: Optimization history of the similarity measure (curve mismatch).

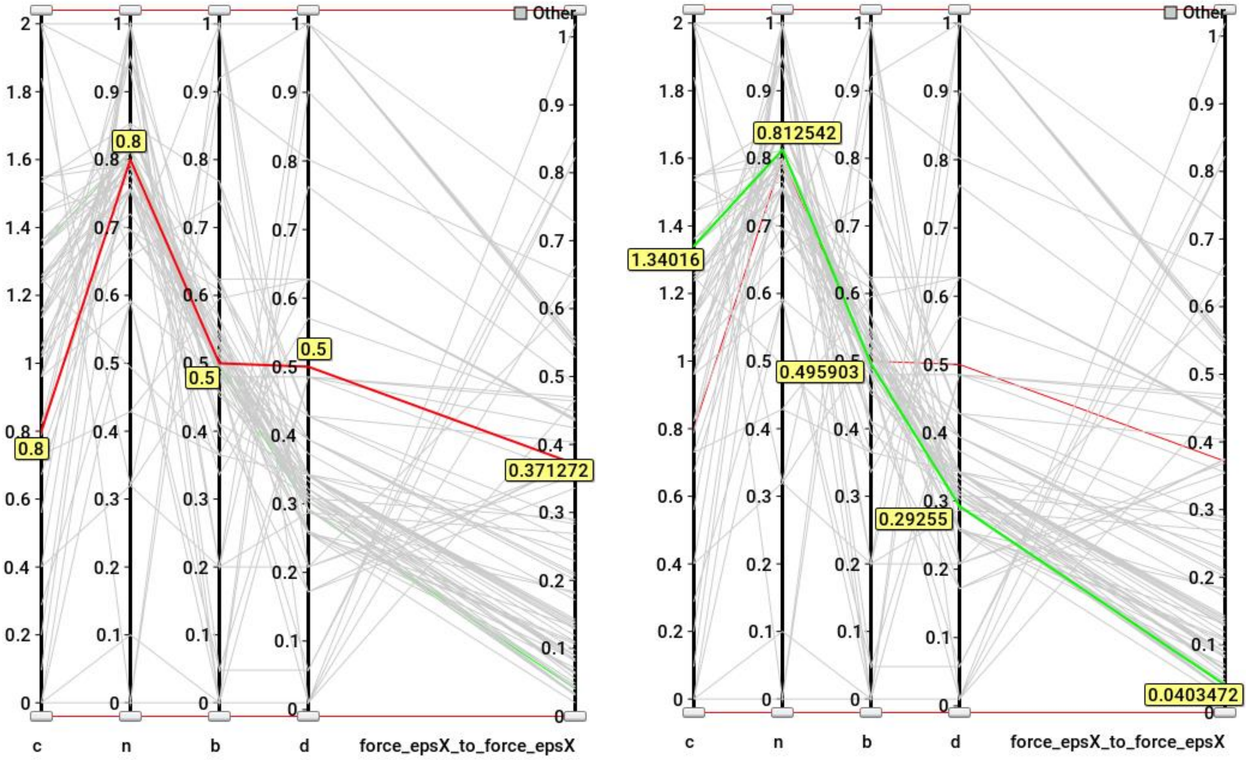


Figure 21-41: Baseline variables C, N, B and D together with objective value (`force_epsX_to_force_epsX`) in red (left) and optimum solution in green (right). The 'categories' option was used to highlight the relevant curve in red/green.

21.4.6. Nodal Mapping using a nearest node cluster

Test results can also be mapped to the FE model nodes instead of the elements. This results in a point cluster of 156 points (out of the original 463). In this case a nodal cluster was created by finding the nearest neighbors of the test points (Figure 21-43). It can be seen in Figure 21-44 that, in spite of the point-node proximity error, the optimal parameter set is very similar to that of the element mapping (Figure 21-41).

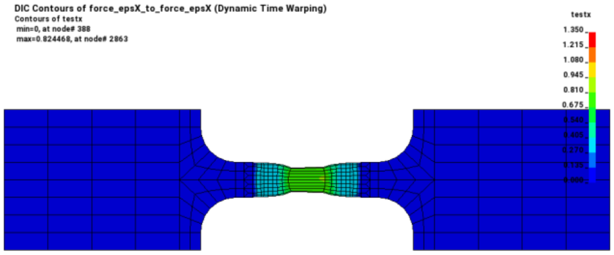
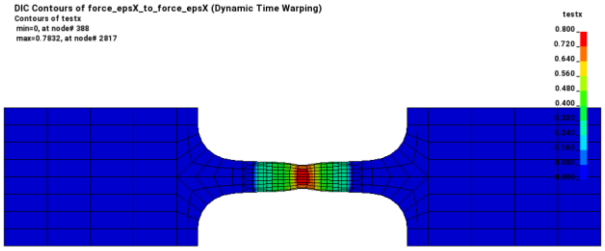
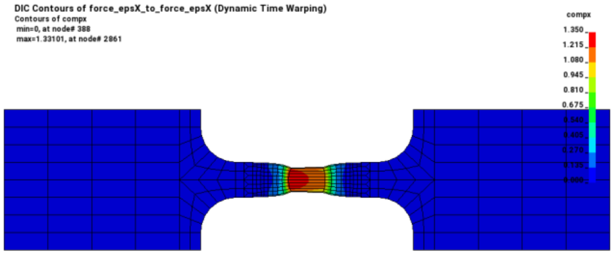
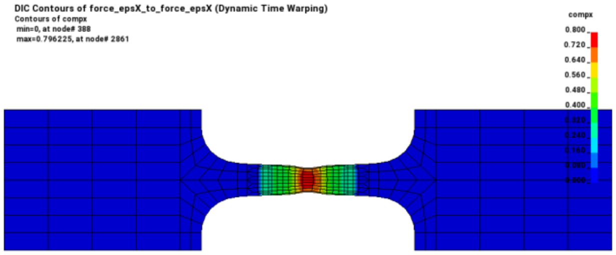
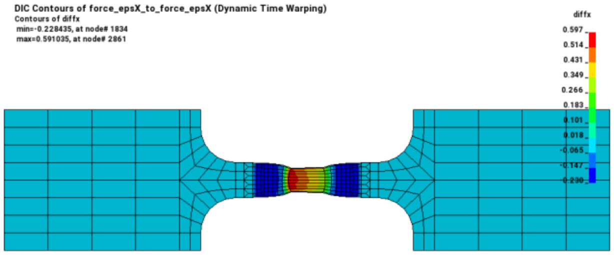
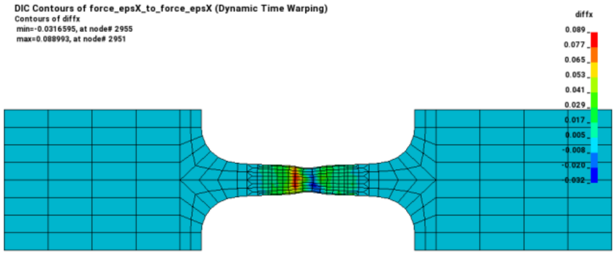
Baseline	**Optimum**
Test	
	
Simulation	
	
Difference = Test – Simulation	
	

Figure 21-42: Comparison of calibrated model with baseline model.The test and simulation plots are on the same scale for the baseline and optimum respectively. The differences are on unique scales [-0.23,0.591] and [-0.032,0.089] for baseline and optimum respectively.

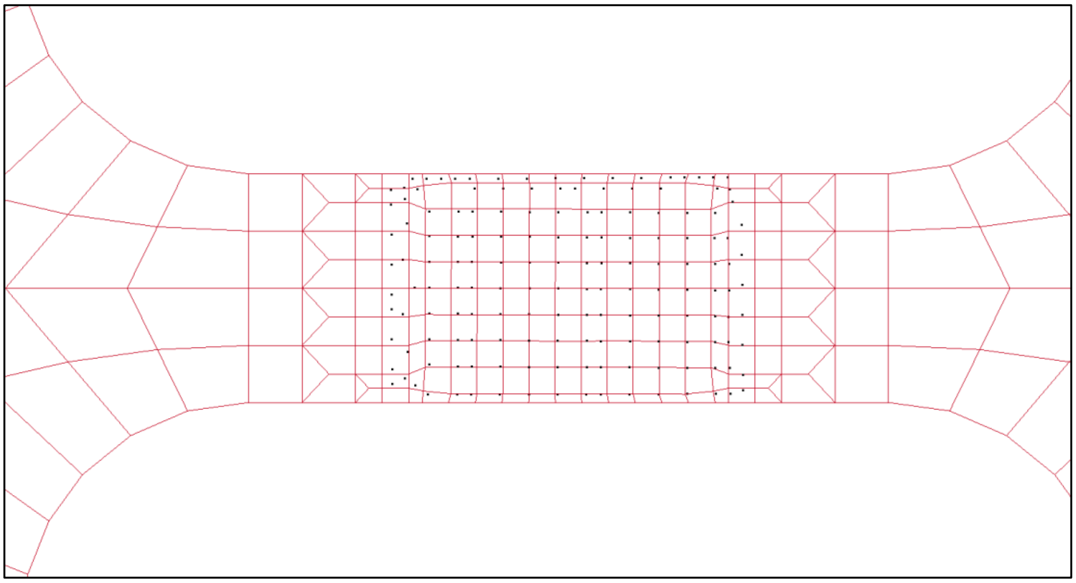


Figure 21-43: Nearest Node point cluster (compare to Figure 21-38).The number of points has been reduced from 463 to 156.

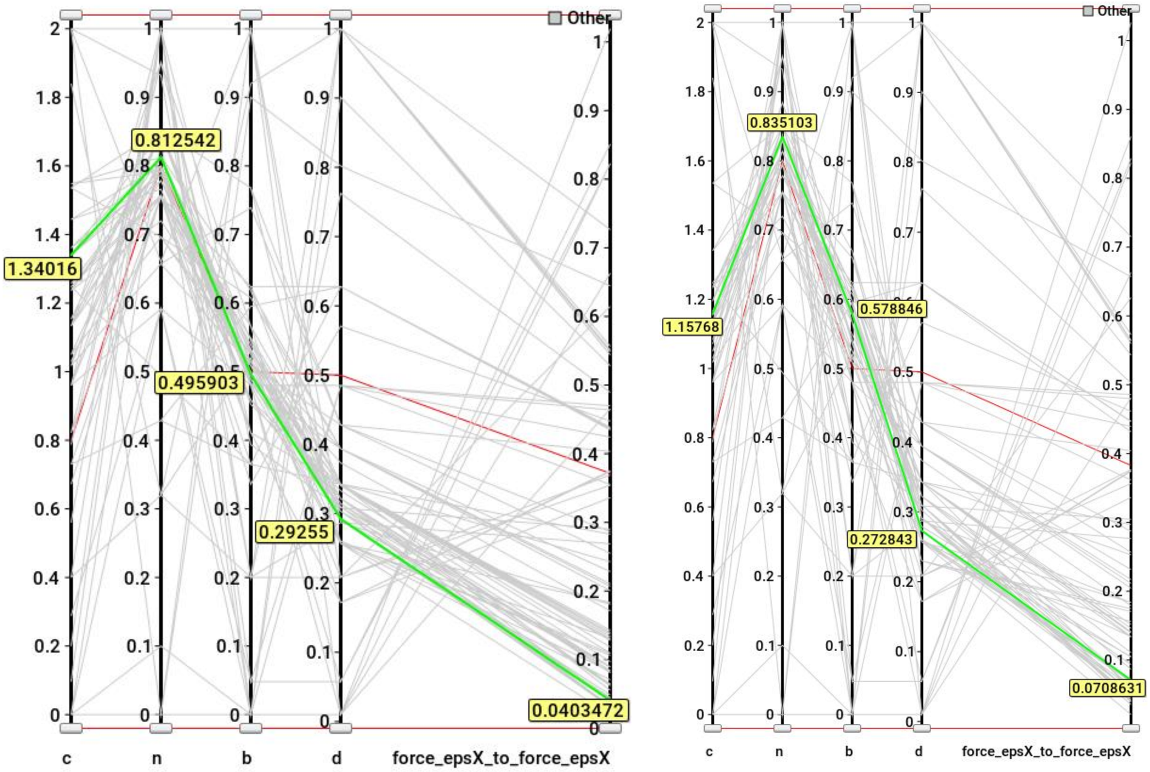


Figure 21-44: Comparison of the optimal material parameters for element mapping (left) and nodal mapping (right) using a nearest nodal neighbor cluster. Optimal variables C, N, B and D are shown together with the objective value `(force_epsX_to_force_epsX)`. Baseline shown in red.

22. Examples − Probabilistic Analysis

22.1. Probabilistic Analysis

22.1.1. Overview

This example has the following features:

- Probabilistic analysis
- Monte Carlo analysis
- Monte Carlo analysis using a metamodel
- Bifurcations analysis

22.1.2. Problem description

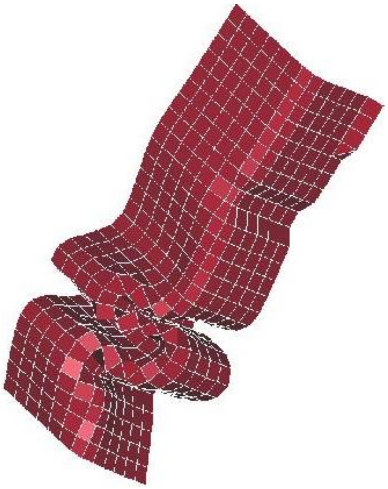


Figure 22-1: Tube impact

A symmetric short crush tube impacted by a moving wall as shown in the figure is considered. The design criterion is the intrusion of the wall into the space initially occupied by the tube (alternatively, how much the structure is shortened by the impact with the wall).

Both the shell thickness and the yield strength of the structure are probabilistic. The shell thickness is normally distributed around a value of 1.0 with a standard deviation of 5% while the yield strength is normally distributed around a value scaled to 1.0 with 10% standard deviation.

The nominal design has an intrusion of 144.4 units. The intrusion is calculated based on the minimum z displacement at node 486 (*NodDisp* response), which lies at the end of the tube. The probability of the intrusion being greater than 150 units is computed. The best-known results are obtained using a Monte Carlo analysis of 1500 runs. The problem is analyzed using a Monte-Carlo evaluation of 60 runs and a quadratic response surface built using a 3^k experimental design. The results from the different methods are similar as can be seen in the following table.

Table 22-1: Comparison of results

Response	**Monte Carlo** **1500 runs**	**Monte Carlo** **60 runs**	**Response Surface** **9 runs**
Average Intrusion	141.3	141.8	141.4
Intrusion Standard Deviation	15.8	15.2	15.0
Probability of Intrusion > 150	0.32	0.33	0.29

Using the response surface, the derivatives of the intrusions with respect to the design variables are computed as given in the following table.

Table 22-2: Derivatives of Intrusion from response surface

Variable	**Intrusion derivative**
Shell Thickness	208
Yield Strength	107

The quadratic response surface also allows the investigation of the dependence of the response variation on each design variable variation. The values of the intrusion standard deviation given in the following table are computed considering the variable as the only source of variation in the structure (the variation of the other design variables are set to zero).

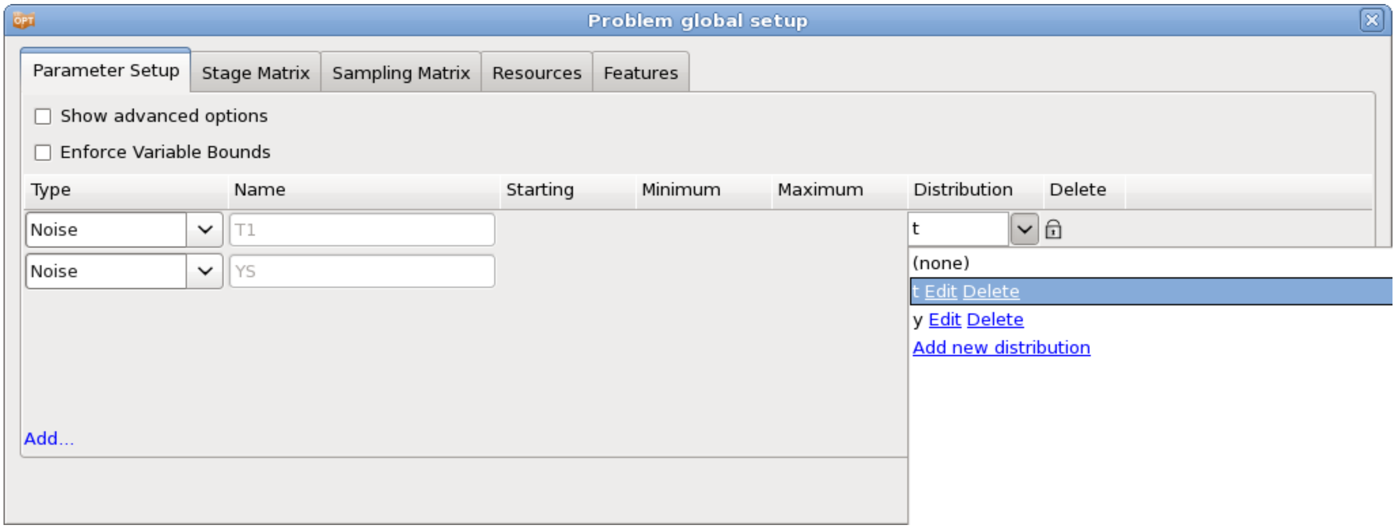
Table 22-3: Standard Deviation of Intrusion

Source of variation	**Intrusion Standard Deviation**
Shell Thickness	10.4
Yield Strength	10.7

The details of the analyses are given the following subsections.

22.1.3. Direct Monte Carlo evaluation

The probabilistic variation is described by specifying statistical distributions, Figure 22-3, and assigning the statistical distributions to noise variables, Figure 22-2. Monte Carlo samples generated based on the distributions are evaluated through the solver to calculate response statistics such as the probability of failure, standard deviation of the responses etc.

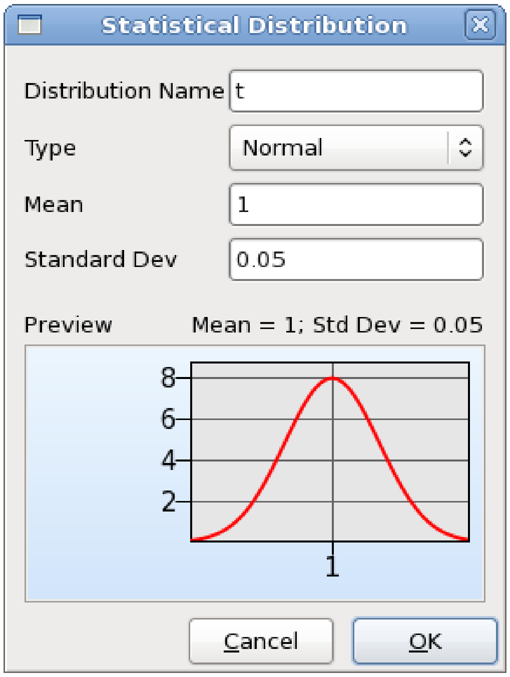


□ Show advanced options

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	T1				t	
Noise	YS					

Figure 22-2: Assigning statistical distributions to noise variables



Distribution Name: t

Type: Normal

Mean: 1

Standard Dev: 0.05

Preview Mean = 1; Std Dev = 0.05

Figure 22-3: Definition of statistical distributions. This dialog is accessible from the Setup dialog's Parameter Setup tab.

Results

The results of the Monte Carlo analysis can be visualized using the **Statistical Tools** plot. The distributions of the variables and responses can be displayed by selecting the plot type **Histogram**, Figure 22-4. The mean value and the standard deviation of the selected entity are displayed in the plot title. The probability of exceeding a bound of a constraint with 95% confidence interval can be displayed by selecting the plot type **Bounds**, Figure 22-5. The bounds can be modified interactively in the viewer.

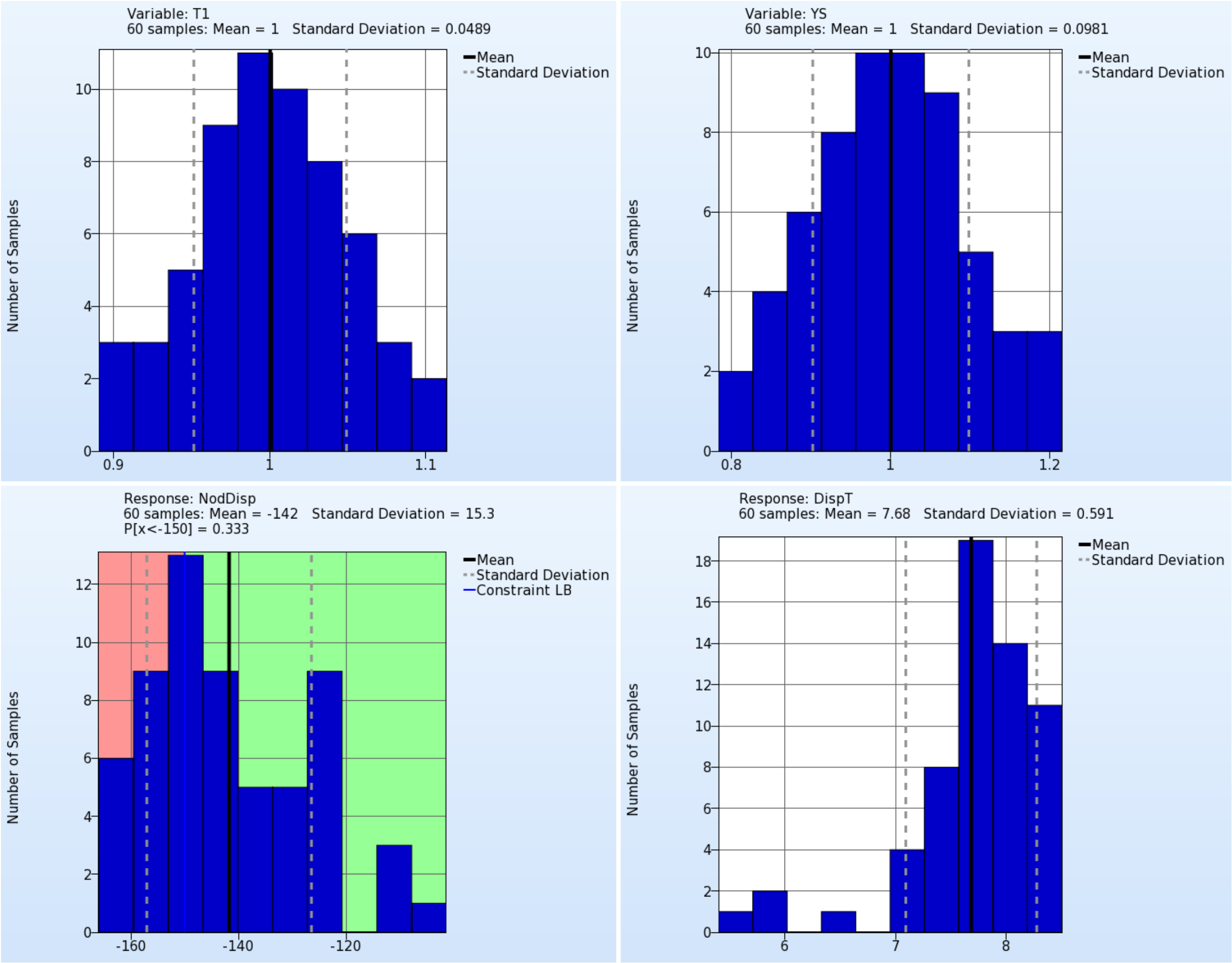


Figure 22-4: Histogram plots of variables and responses with mean value and standard deviation. Feasibility information is displayed for the constraint NodDisp. The values are displayed in the titles.

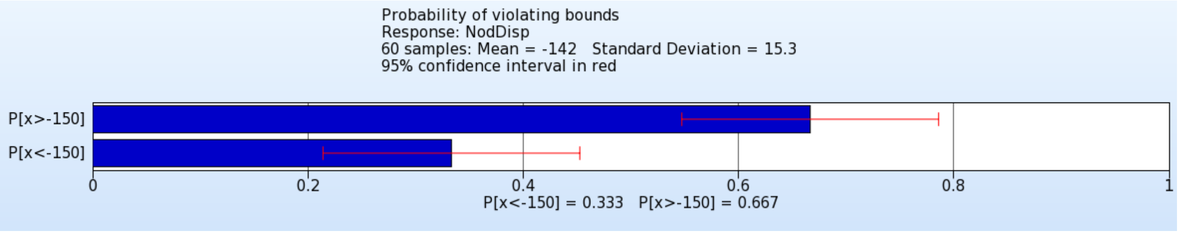
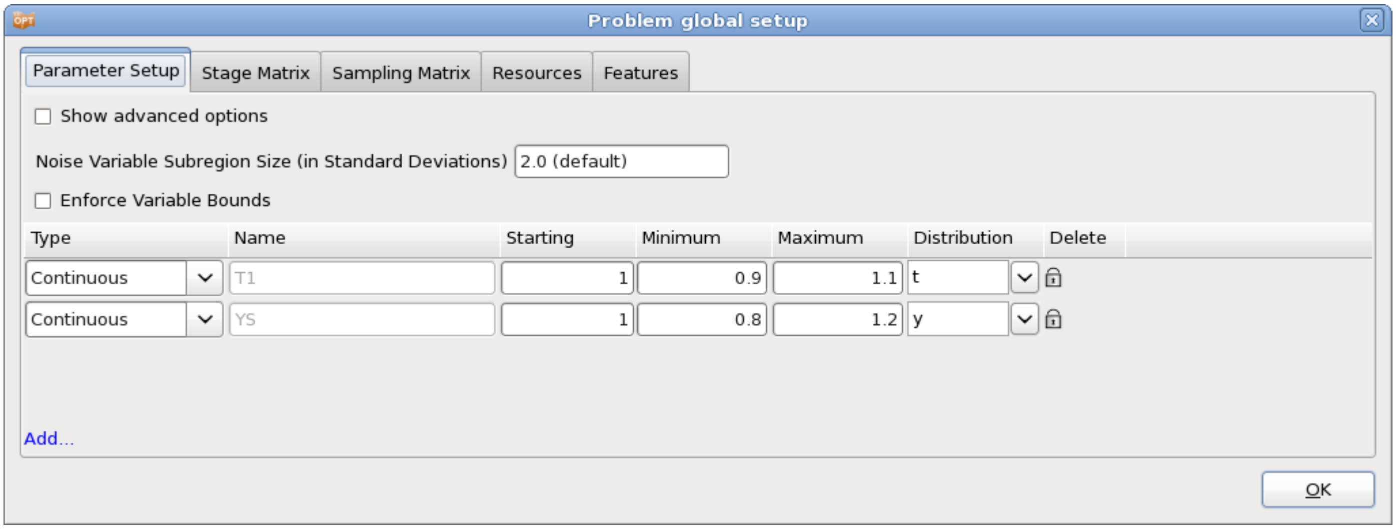


Figure 22-5: Probability of NodDisp < -150 with 95% confidence intervals

22.1.4. Monte Carlo using metamodel

The sampling scheme for metamodel-based Monte Carlo analysis differs from the direct MC method. In the metamodel-based method, the sampling is not completely defined by the variable distributions; specific variable bounds are required to construct the metamodels. If a variable's type is defined as Noise, its bounds are set to be two standard deviations away from the mean (default), Figure 22-6. This multiple can however be changed by the user. In this particular example, noise variables are not used in order to have more control over the variable bounds. If needed we can change the standard deviation of some variables without affecting the variable bounds (the metamodel is computed scaled with respect to the upper and lower bounds on the variables). If noise variables are used instead, then we would only define the design space size in terms of number of standard deviations (default 2), which is same for all noise variables.



□ Show advanced options

Noise Variable Subregion Size (in Standard Deviations) 2.0 (default)

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Continuous	T1	1	0.9	1.1	t	🔒
Continuous	YS	1	0.8	1.2	y	🔒

Figure 22-6: Assigning statistical distributions to control variables

Results

Since the statistical results are evaluated on the metamodel, the accuracy of the response surface is of interest. This can be displayed using the **Accuracy** plot, Figure 22-7. The error measures RMS, SPRESS and R² are displayed in the title.

The probabilistic evaluation results can be visualized using the **Statistical Tools** plot as described in Section 22.1.3, but now, 10000 points evaluated on the metamodel are used to calculate the statistics, Figure 22-9 and Figure 22-8.

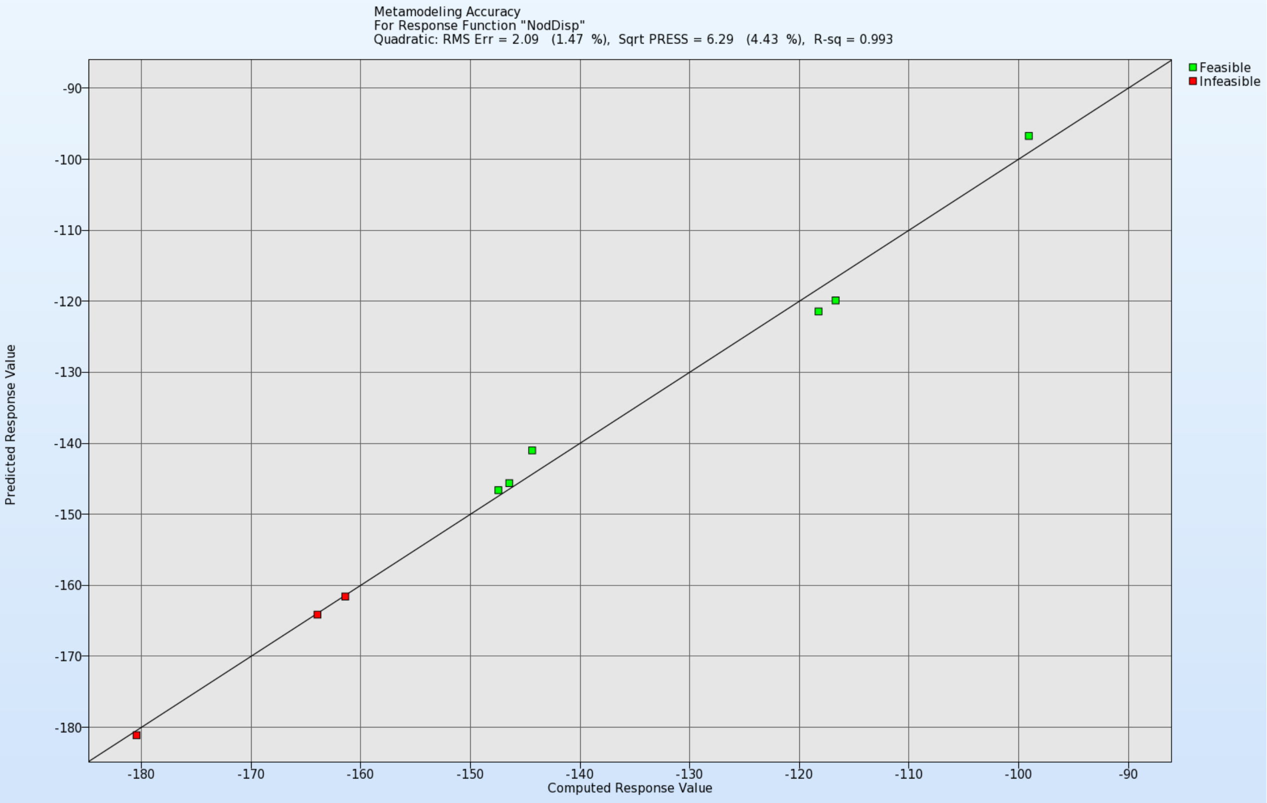


Figure 22-7: Accuracy plot. Computed vs. predicted values; error measures are displayed in the title

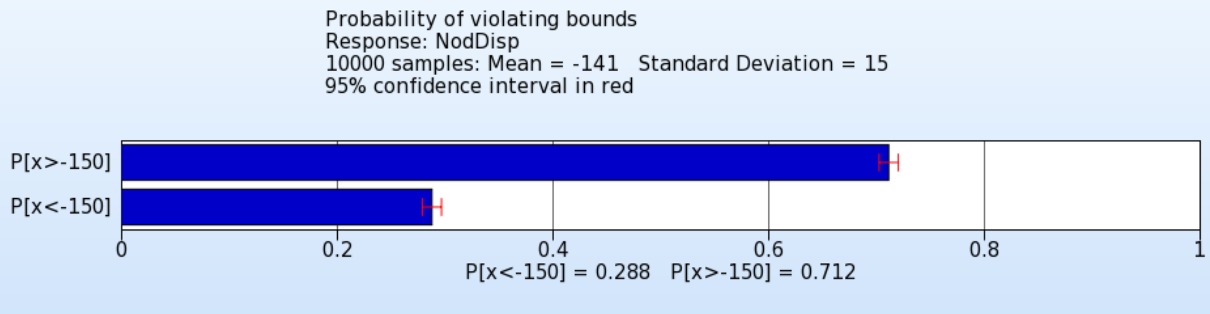


Figure 22-8: Probability of NodDisp < -150 with 95% confidence intervals evaluated on the metamodel using 10000 points.

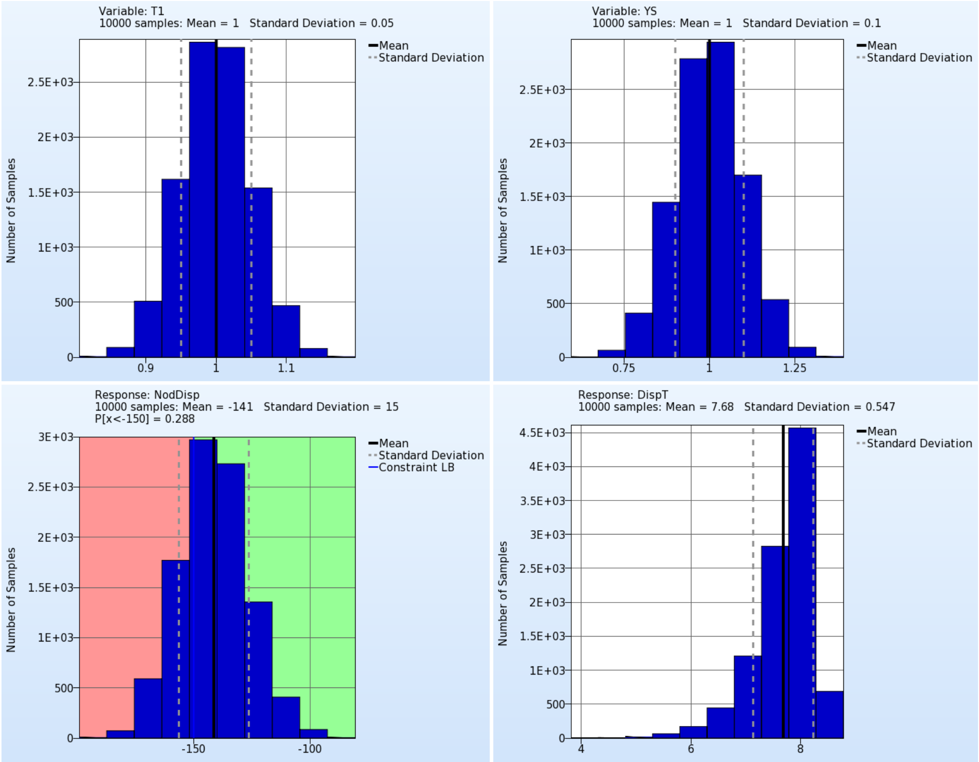
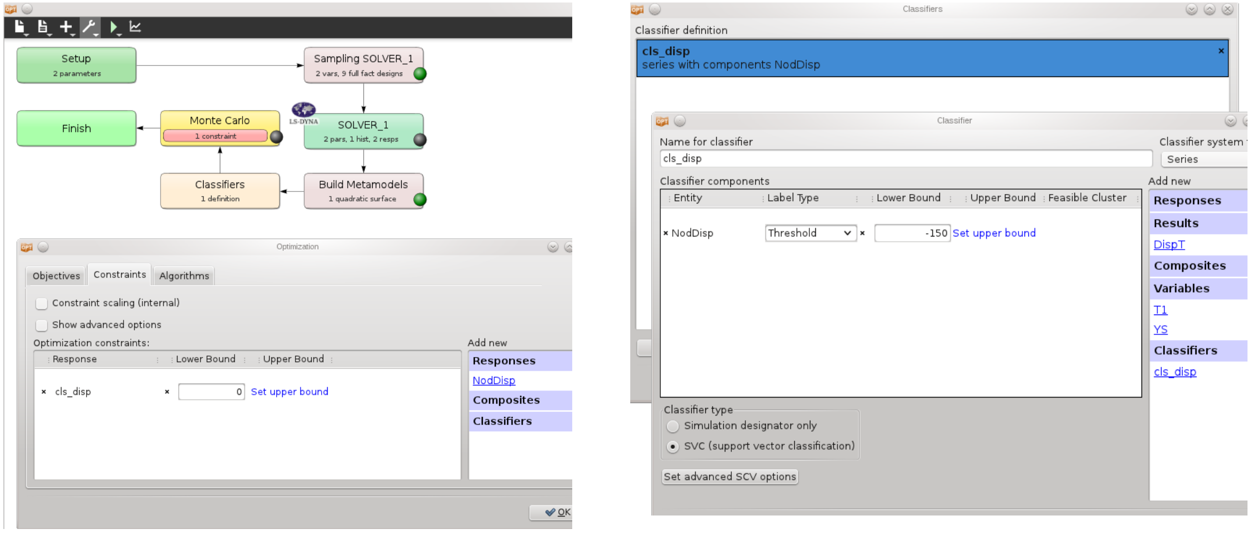


Figure 22-9: Histogram plots of variables and responses. Mean values and standard deviations are displayed in the titles. All values are evaluated on the metamodel using 10000 points.

22.1.5. Monte Carlo using classifier – single iteration

Classification-based Monte Carlo is similar to the metamodel-based method, except for the definition of a classifier that is used as the constraint. The classifier cls_disp is defined based on the response NodeDisp (Figure 22-10). The results are shown in Figure 22-11 and Figure 22-12.



□ Constraint scaling (internal)

□ Show advanced options

Optimization constraints:

Response	Lower Bound	Upper Bound
× cls_disp	× 0	Set upper bound

Classifier definition

cls_disp
series with components NodDisp

Name for classifier: cls_disp

Classifier system: Series

Classifier components

Entity	Label Type	Lower Bound	Upper Bound	Feasible Cluster
× NodDisp	Threshold	× -150	Set upper bound	

Classifier type

○ Simulation designator only

● SVC (support vector classification)

Set advanced SCV options

Figure 22-10: Monte Carlo Setup using a Classifier constraint

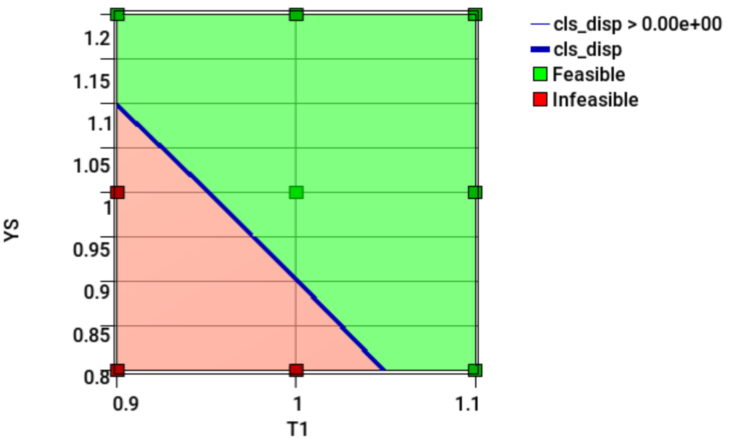


Figure 22-11: Classifier constraint approximation

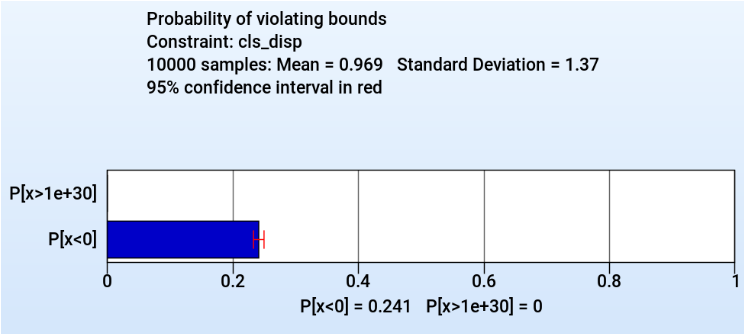


Figure 22-12: Classifier-based failure probability

22.1.6. Monte Carlo using classifier – sequential space-filling sampling

Monte Carlo analysis can also be performed sequentially by selecting the appropriate strategy. Space-filling sampling is recommended for adding samples iteratively. The setup is shown in Figure 22-13. The results are shown in Figure 22-14 and Figure 22-15.

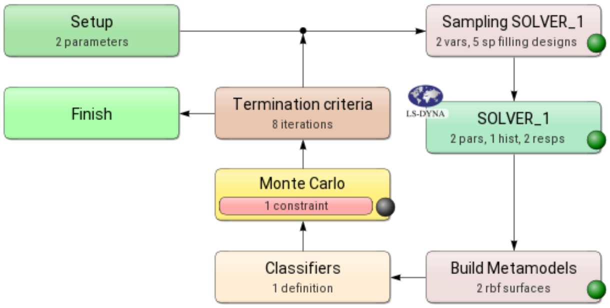


Figure 22-13: Sequential classifier-based Monte Carlo analysis setup

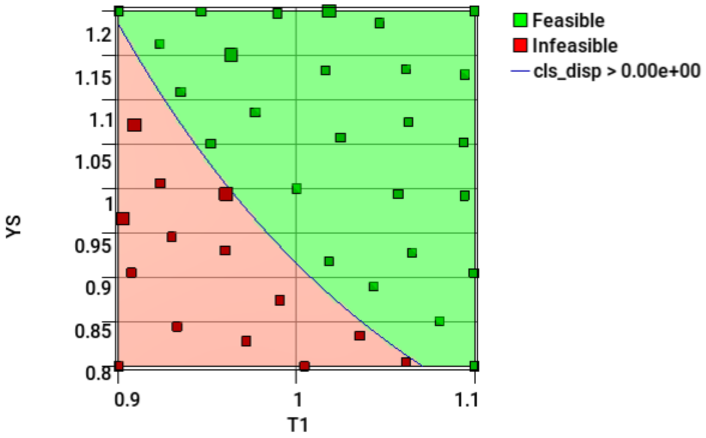


Figure 22-14: Classifier-based constrained using sequential space-filling sampling

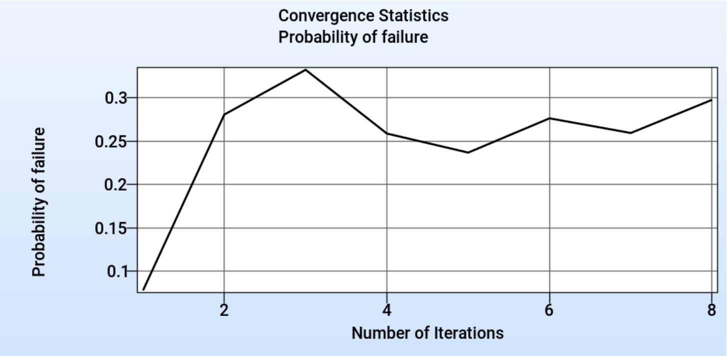


Figure 22-15: Sequential classifier-based failure probability history

22.1.7. Monte Carlo using classifier – sequential adaptive sampling

Adaptive sequential Monte Carlo analysis can be performed using EDSD sampling constraints. The task and strategy are the same as in Section 22.1.6, but the space-filling samples are selected in an adaptive region in the vicinity of the previous classifier estimate. The results are shown in

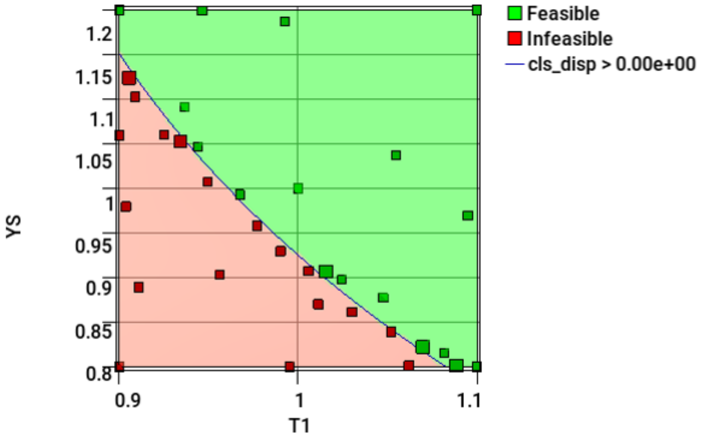


Figure 22-16: Classifier-based constrained using adaptive sequential sampling

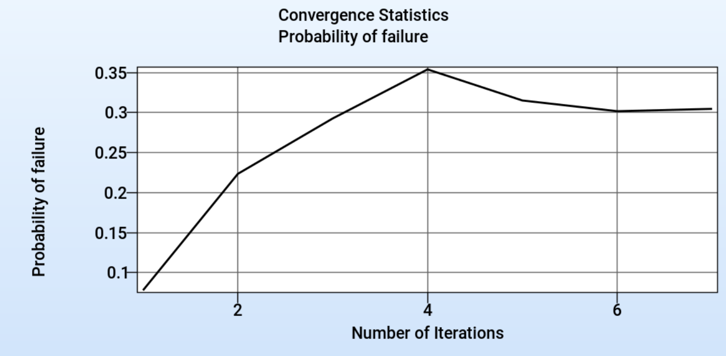


Figure 22-17: Failure probability history using adaptively refined classifier

22.1.8. Bifurcation analysis

A bifurcation analysis of the tube is conducted using the methods described in more detail in Section 29.6, Section 18.8, and Section 22.2. The resulting buckling modes found from the analysis are as shown in Figure 22-18. An extra half wave is formed for one of the designs.

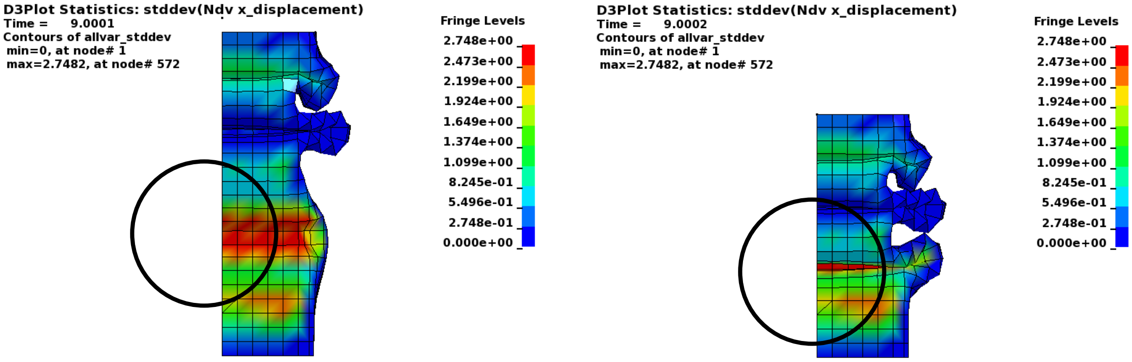


Figure 22-18: Tube Buckling – model with minimal x displacement (left) and maximal x displacement (right)

22.2. Bifurcation/Outlier Analysis

22.2.1. Overview

This example has the following features:

- Monte Carlo analysis
- Identification of different buckling modes in the structure

22.2.2. Problem description

The plate as shown in Figure 22-19 has two buckling modes. Buckling in the positive z-direction occurs with a probability of 80% while buckling in the negative z-direction occurs with a probability of 20%. The statistical distribution of the tip nodes imperfection controls the probability of buckling in a particular mode.

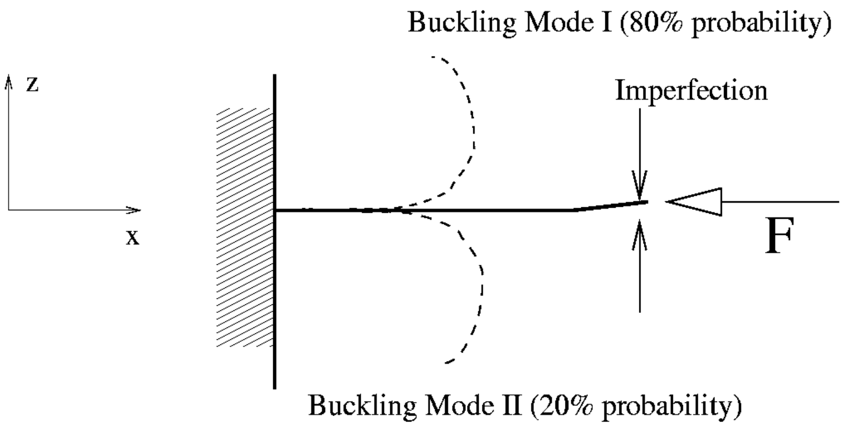
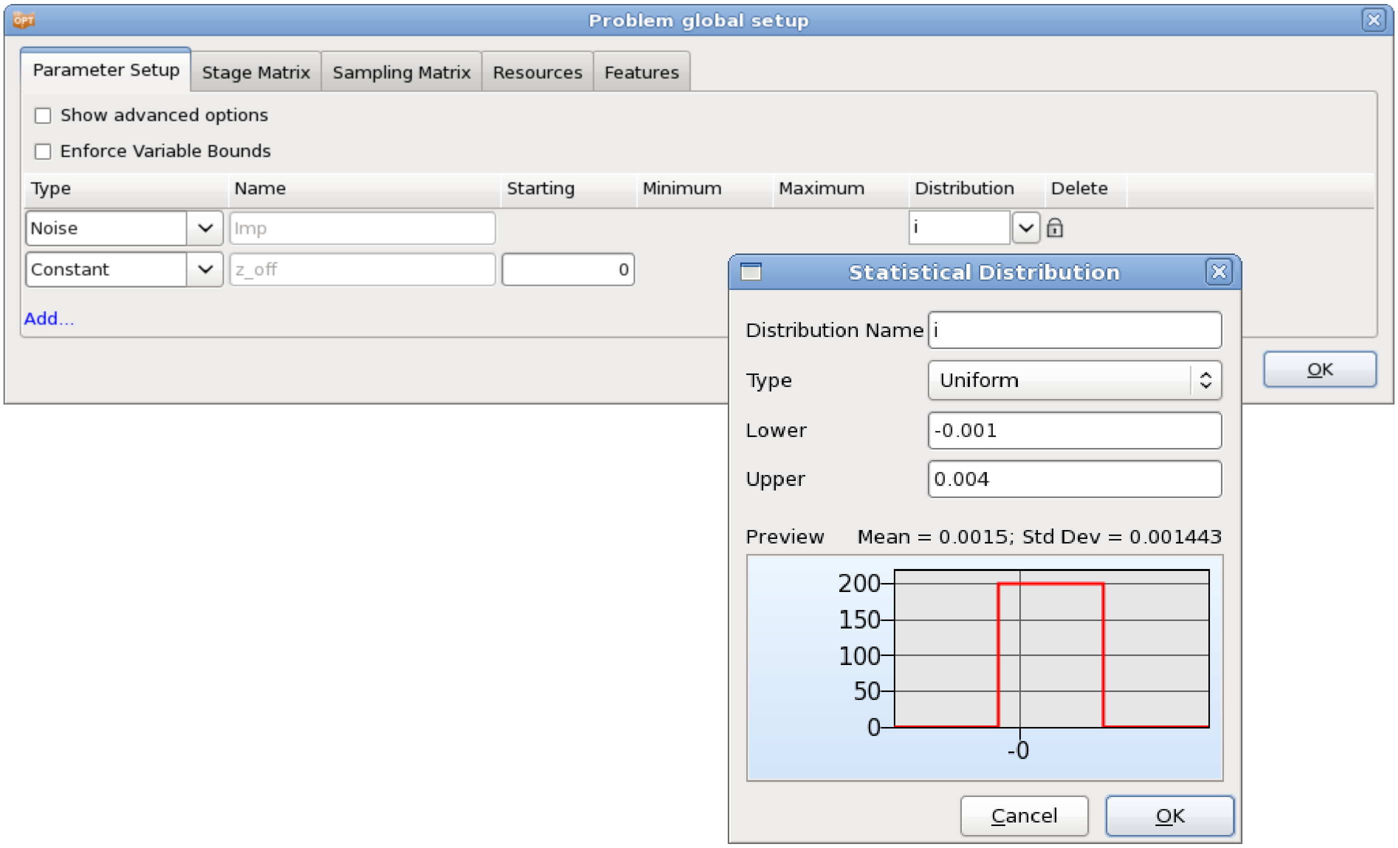

Buckling Mode I (80% probability)
z
x
Imperfection
F
Buckling Mode II (20% probability)

Figure 22-19: Plate Buckling Example



□ Show advanced options

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	Imp				i	
Constant	z_off	0				

Distribution Name: i

Type: Uniform

Lower: -0.001

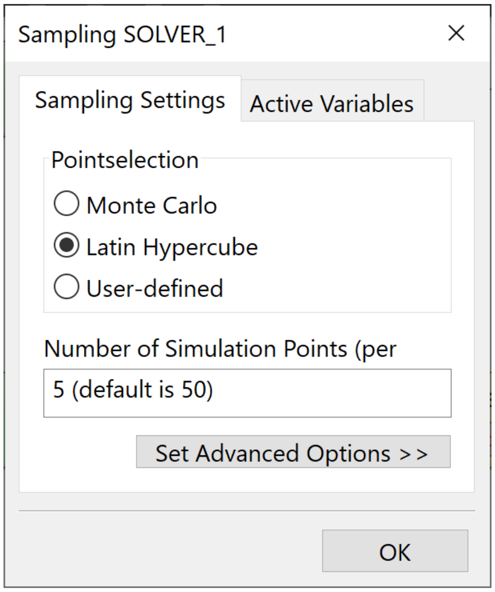
Upper: 0.004

Mean = 0.0015; Std Dev = 0.001443

Figure 22-20: Definition of noise variable with uniform distribution

22.2.3. Monte Carlo evaluation

A Latin hypercube experimental design is used for the Monte Carlo analysis, Figure 22-21. We analyze only five points. Given that the probability of 20% of buckling in the negative z-direction and a Latin hypercube experimental design, one run will buckle in the negative z-direction. The difference between the two modes lies in the z-displacement. Therefore, z-displacement of the tip node is defined as a response, Figure 22-22. The next section will demonstrate how to find out which run contains the different buckling mode.



Sampling SOLVER_1

Pointselection

○ Monte Carlo

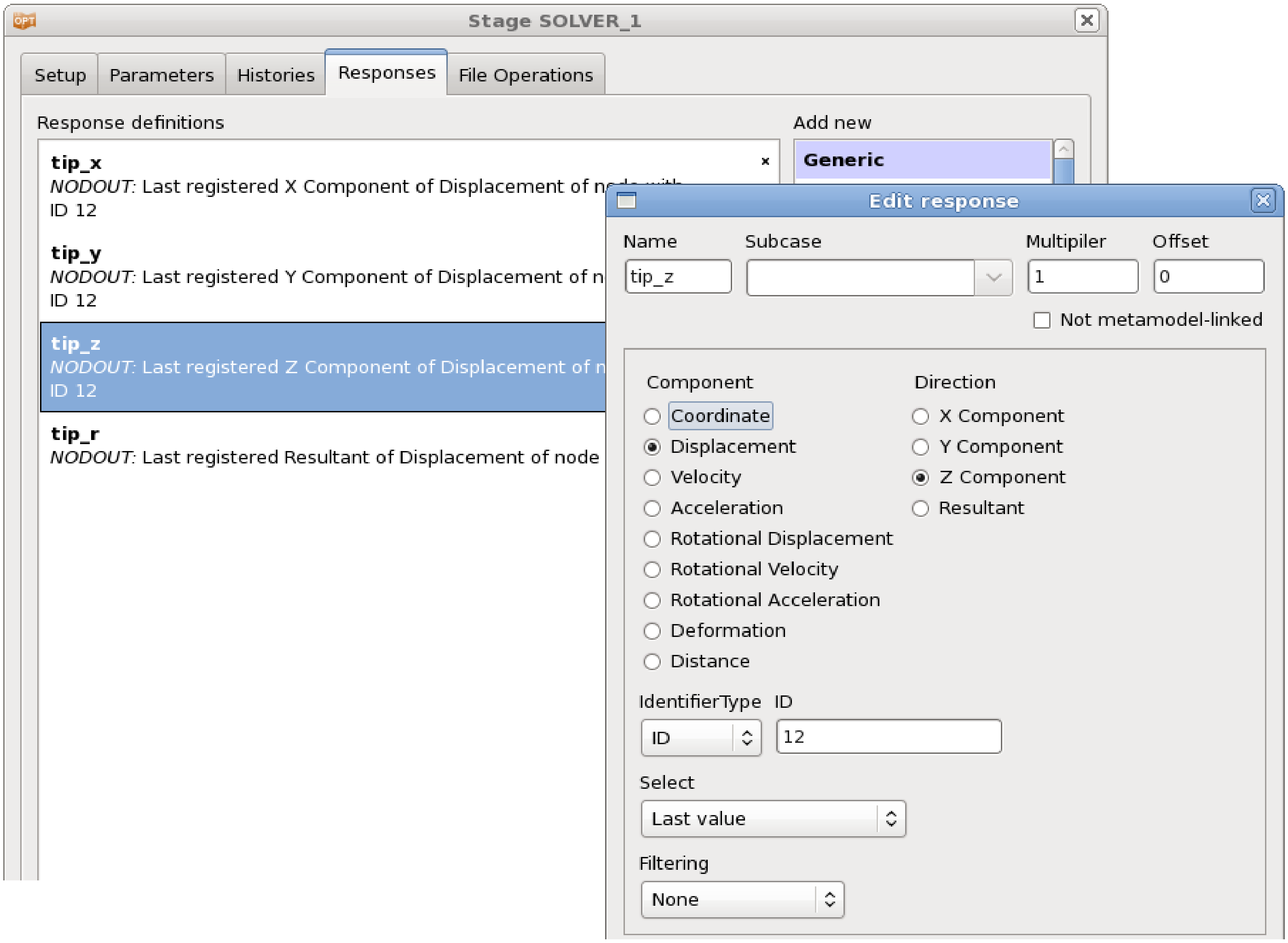
● Latin Hypercube

○ User-defined

Number of Simulation Points (per

5 (default is 50)

Figure 22-21: Definition of Latin Hypercube sampling with 5 points.



Response definitions

tip_x
NODOUT: Last registered X Component of Displacement of node with
ID 12

tip_y
NODOUT: Last registered Y Component of Displacement of n
ID 12

tip_z
NODOUT: Last registered Z Component of Displacement of n
ID 12

tip_r
NODOUT: Last registered Resultant of Displacement of node

Name: tip_z
Subcase:
Multipiler: 1
Offset: 0
☐ Not metamodel-linked

Component
○ Coordinate
● Displacement
○ Velocity
○ Acceleration
○ Rotational Displacement
○ Rotational Velocity
○ Rotational Acceleration
○ Deformation
○ Distance

Direction
○ X Component
○ Y Component
● Z Component
○ Resultant

IdentifierType: ID
ID: 12

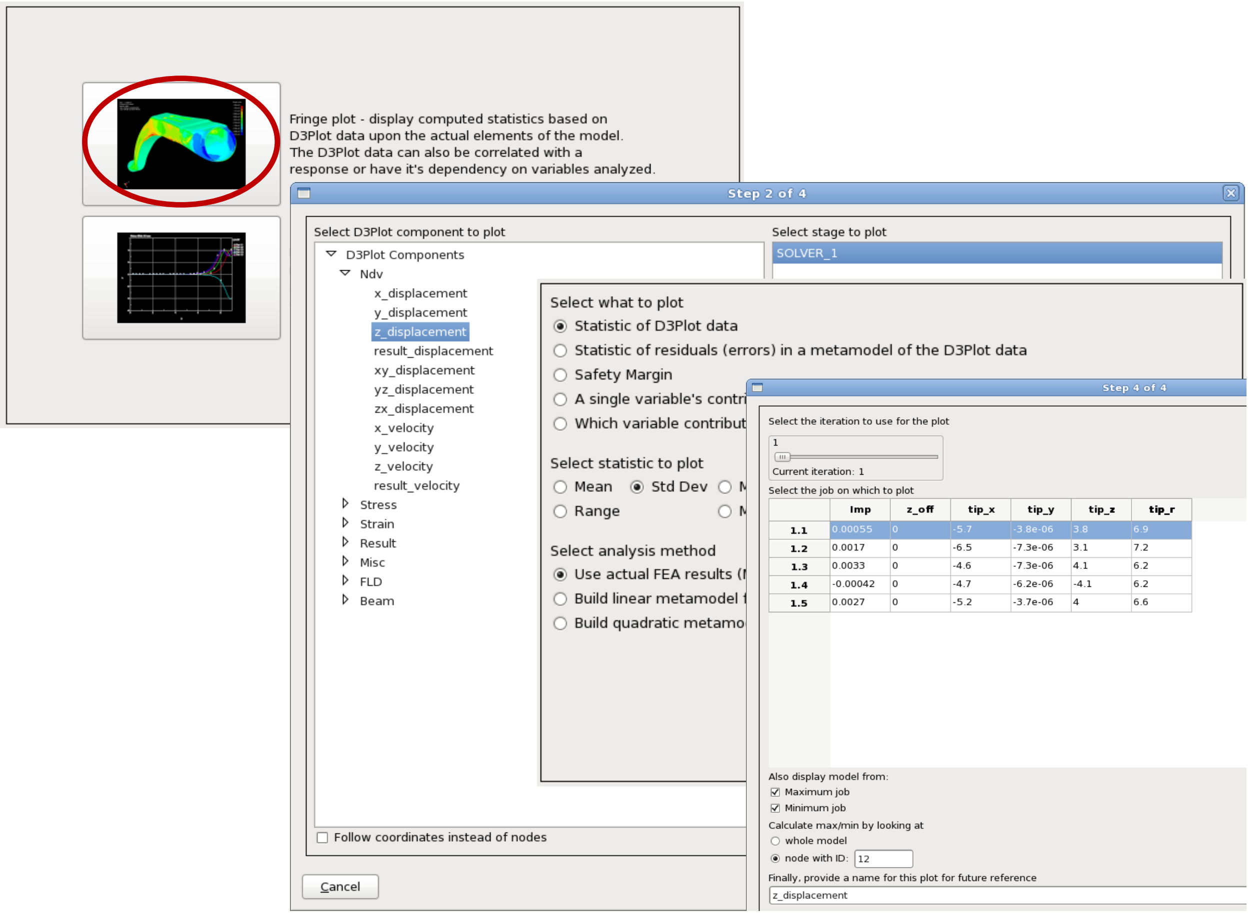
Select: Last value

Filtering: None

Figure 22-22: Response Definition for bifurcation identification

22.2.4. Automatic identification of buckling modes

Different buckling modes can be identified automatically and displayed in LS-PREPOST using DynaStats accessible from the main GUI control bar. To identify bifurcations, we display the FE jobs having the extreme values for a selected d3plot component. For this structure, either the global extreme z-displacement or the tip z-displacement can be considered in order to identify the bifurcation. Automated identification of the bifurcation is done in the DynaStats GUI as shown in **Error! Reference source not found.** with the b ifurcation as displayed using LS-PREPOST as shown in Figure 22-24. Some background on bifurcation identification can be found in Section 18.8. A more user-intensive procedure is described in the next section.



Fringe plot - display computed statistics based on D3Plot data upon the actual elements of the model. The D3Plot data can also be correlated with a response or have it's dependency on variables analyzed.

Select D3Plot component to plot

- D3Plot Components
  - Ndv
    - x_displacement
    - y_displacement
    - z_displacement
    - result_displacement
    - xy_displacement
    - yz_displacement
    - zx_displacement
    - x_velocity
    - y_velocity
    - z_velocity
    - result_velocity
  - Stress
  - Strain
  - Result
  - Misc
  - FLD
  - Beam

☐ Follow coordinates instead of nodes

Select stage to plot

SOLVER_1

Select what to plot

● Statistic of D3Plot data
○ Statistic of residuals (errors) in a metamodel of the D3Plot data
○ Safety Margin
○ A single variable's contri
○ Which variable contribut

Select statistic to plot

○ Mean ● Std Dev ○ M
○ Range ○ M

Select analysis method

● Use actual FEA results (I
○ Build linear metamodel f
○ Build quadratic metamo

Select the iteration to use for the plot

1

Current iteration: 1

Select the job on which to plot

	Imp	z_off	tip_x	tip_y	tip_z	tip_r
1.1	0.00055	0	-5.7	-3.8e-06	3.8	6.9
1.2	0.0017	0	-6.5	-7.3e-06	3.1	7.2
1.3	0.0033	0	-4.6	-7.3e-06	4.1	6.2
1.4	-0.00042	0	-4.7	-6.2e-06	-4.1	6.2
1.5	0.0027	0	-5.2	-3.7e-06	4	6.6

Also display model from:

☑ Maximum job
☑ Minimum job

Calculate max/min by looking at

○ whole model
● node with ID: 12

Finally, provide a name for this plot for future reference

z_displacement

Figure 22-23: Selecting the automated identification of a bifurcation. The user must (i) select to overlay the FE models associated with the maximum and minimum residual and (ii) chose whether the residual is the global residual or a residual at a specific node.

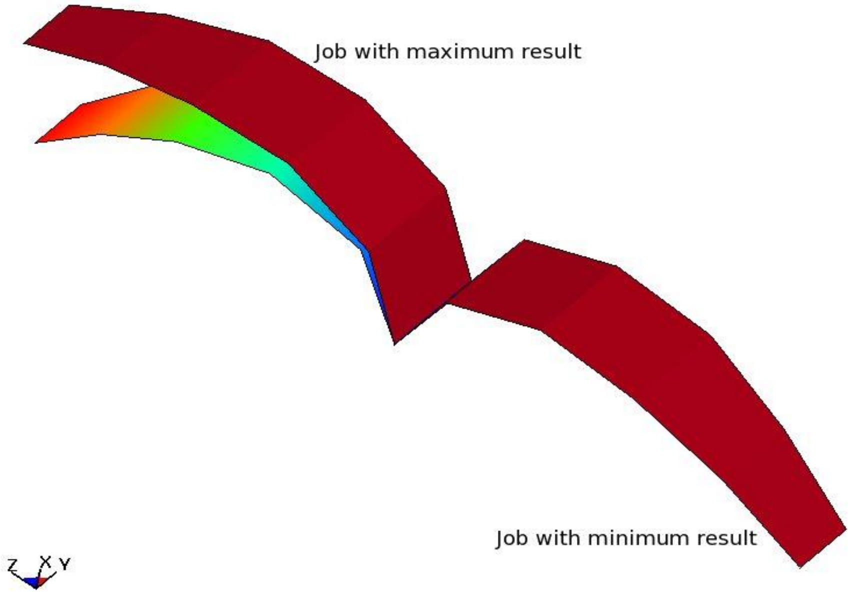


Figure 22-24: LS-OPT identified and displayed this bifurcation automatically using the GUI setting shown in the previous figure.

22.2.5. Manual identification of buckling modes

The different buckling modes are identified using DynaStats accessible from the main GUI control bar.

Next, LS-PREPOST can be launched to investigate the range (or standard deviation) of all the displacement components. From the displacement resultant plot, amongst others, it is clear that the bifurcation is at the tip. Looking at the other component plots, we find the z-displacement has a range of 5.3 and the x-displacement a range of 4.5. The displacement magnitude computed using the maximum vector has a range of 6.9.

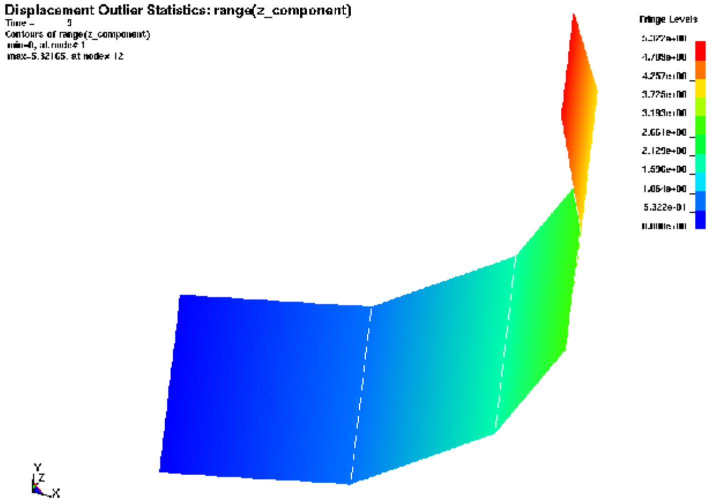


Figure 22-25: Range of z-component displacement

Either the z-displacement or the maximum vector displacement magnitude can therefore be used to identify the buckling modes. Fringe plots of the run index of the maximum and minimum displacement identifies the runs as 2 and 4.

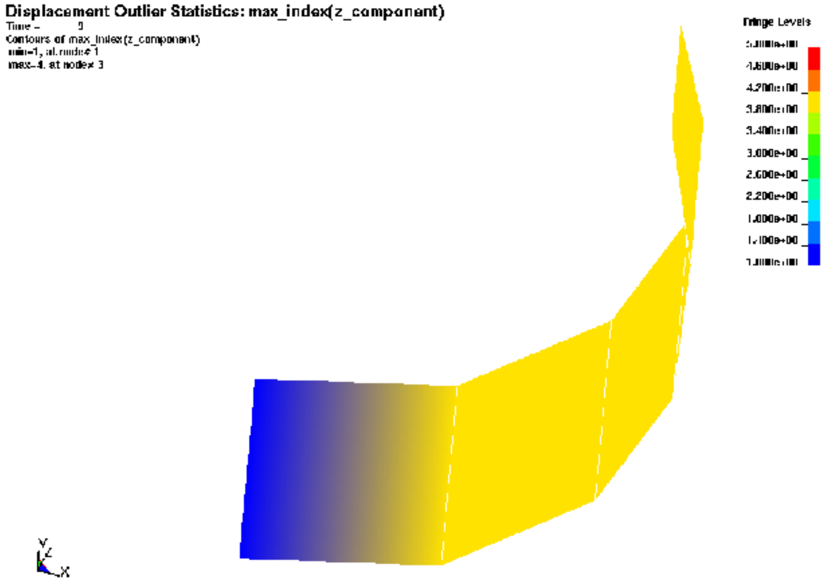


Figure 22-26: Index of run with maximum z-component displacement

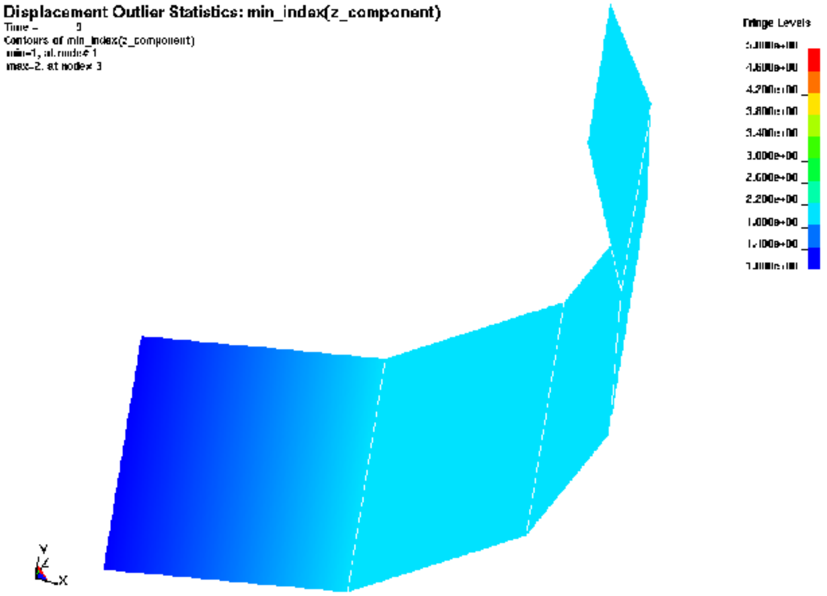


Figure 22-27: Index of run with minimum z-component displacement

LS-OPT allows you to specify the job number to use for the LS-PREPOST plot. Plotting the results of run 2 and 4 we find the second buckling mode as:

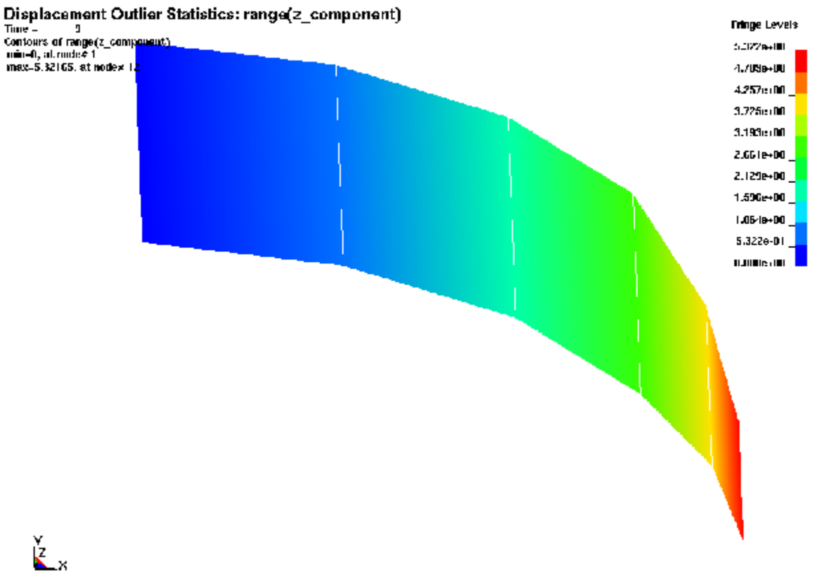


Figure 22-28: Second buckling mode

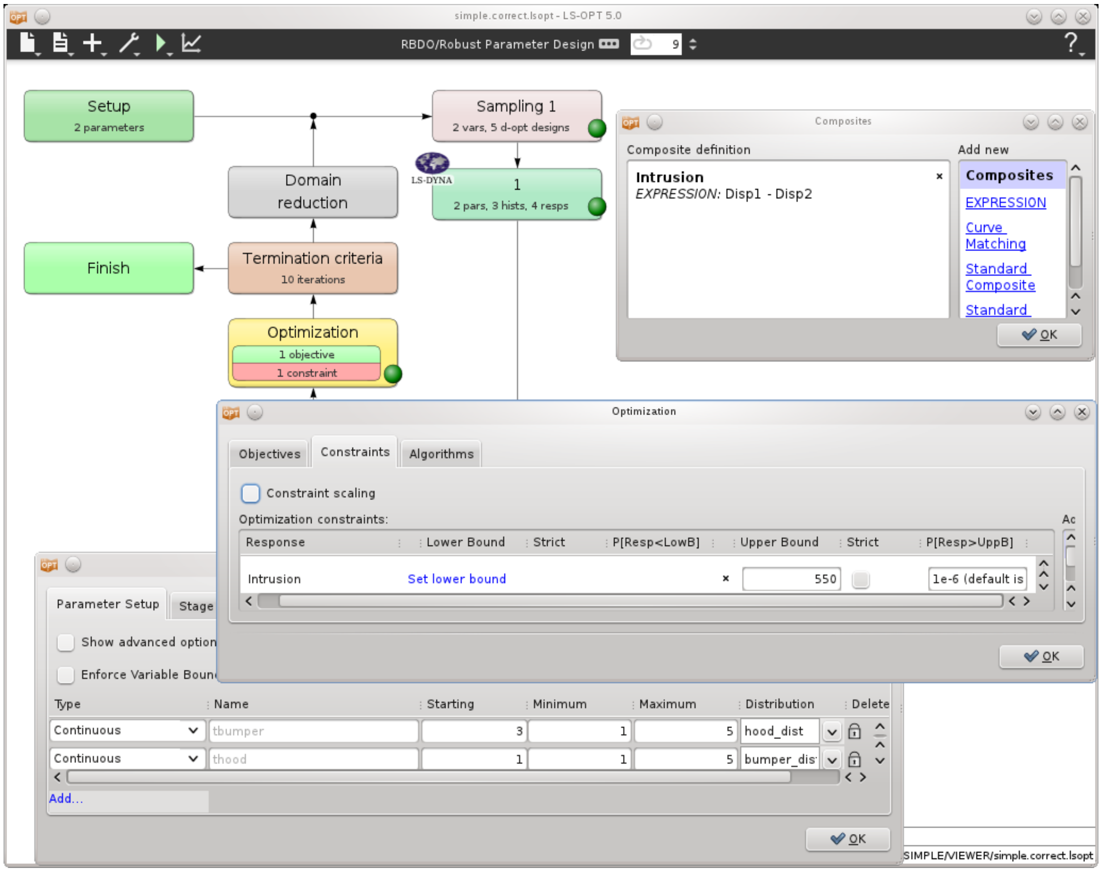
22.3. RBDO (Reliability-based design optimization) using FOSM (First Order Second Moment Method)

This section presents an example of RBDO using the same vehicle problem defined in Section 20.2. The constraint is modified by introducing a target failure probability. The reliability calculations within the optimization loop are done using FOSM. The optimization problem is:

$$\begin{aligned} &\text{min} \quad \text{HIC(15 ms)} \\ &\text{s.t.} \quad \text{Probabilit y[Intrusio n} > 550\text{mm]} \le 10^{-6} \end{aligned} \qquad (22\text{-}1)$$

This formulation implies that the design of the car is made safer such that it has a probability of failure less than 10^{-6}. In Section 20.2 the constraint was deterministic and the intrusion was required to be less than 550 mm. If the same constraint was used on the mean value of intrusion in the presence of uncertainties, that could potentially lead to a large probability of failure. This is avoided by having a probabilistic constraint with a small target probability of failure.

In this example the two variables (t_hood, t_bumber $\in [1,5]$) are assigned identical uniform distributions with lower bound of -0.05 and upper bound of 0.05. The SRSM strategy is used to find the optimum. The variable setup and constraint definition are shown in Figure 22-29.



Composite definition

Intrusion
EXPRESSION: Disp1 - Disp2

☐ Constraint scaling

Optimization constraints:

Response	Lower Bound	Strict	P[Resp<LowB]	Upper Bound	Strict	P[Resp>UppB]
Intrusion	Set lower bound		×	550	☐	1e-6 (default is

☐ Show advanced option

☐ Enforce Variable Boun

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Continuous	tbumper	3	1	5	hood_dist	
Continuous	thood	1	1	5	bumper_dis	

Figure 22-29: Probabilistic variable and constraint definition for RBDO

The results are t_hood =1.7, t_bumper=5, a HIC value of 139, and an intrusion of 545 with standard deviation 1.01.

22.4. Robust Parameter Design

This example has the following features:

- Robust parameter design
- Standard deviation composite

Consider the two-bar truss problem as shown in Figure 22-30. Variable `Area`, the cross-sectional area of the bars, is a noise variable described using a normal distribution with a mean of 2.0 and a standard deviation of 0.1. The half distance between the legs, `Base`, is a control variable which will be adjusted to control the variance of the stress response, Figure 22-32. The standard deviation of stress response is considered as the objective for the robust design process, Figure 22-33.

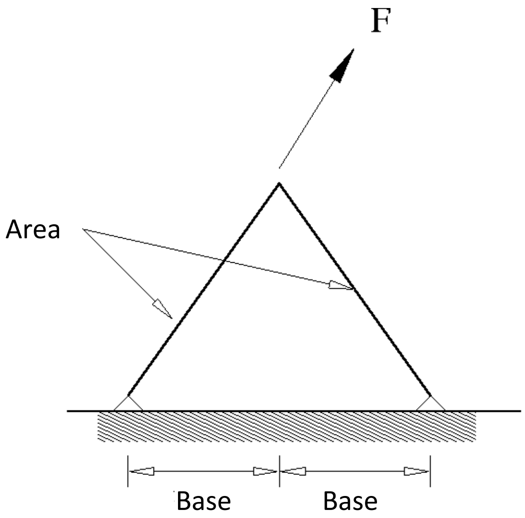
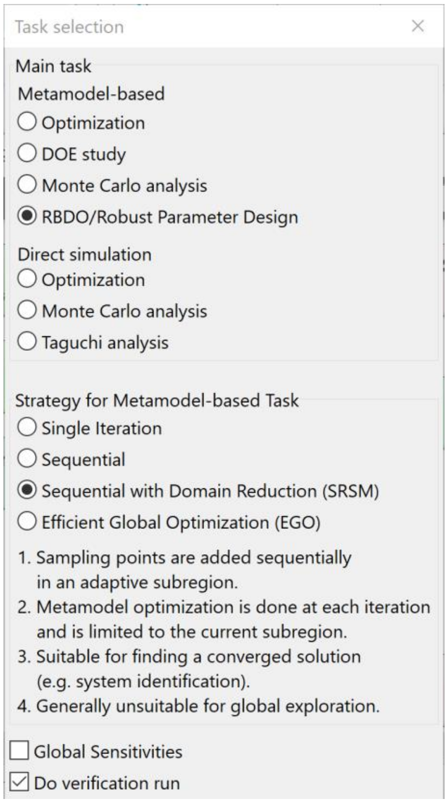


Figure 22-30: The two-bar truss problem. The problem has two variables: the thickness of the bars and the leg widths as shown. The bar thicknesses are noise variables while the leg widths are adjusted (control variables) to minimize the effect of the variation of the bar thicknesses. The maximum stress in the structure is monitored.

The task metamodel-based RBDO/Robust Parameter Design is used, Figure 22-31. A response surface considering the effect of variables and the interaction between variables is used to approximate the stress response.


Task selection

Main task

Metamodel-based

○ Optimization
○ DOE study
○ Monte Carlo analysis
● RBDO/Robust Parameter Design

Direct simulation

○ Optimization
○ Monte Carlo analysis
○ Taguchi analysis

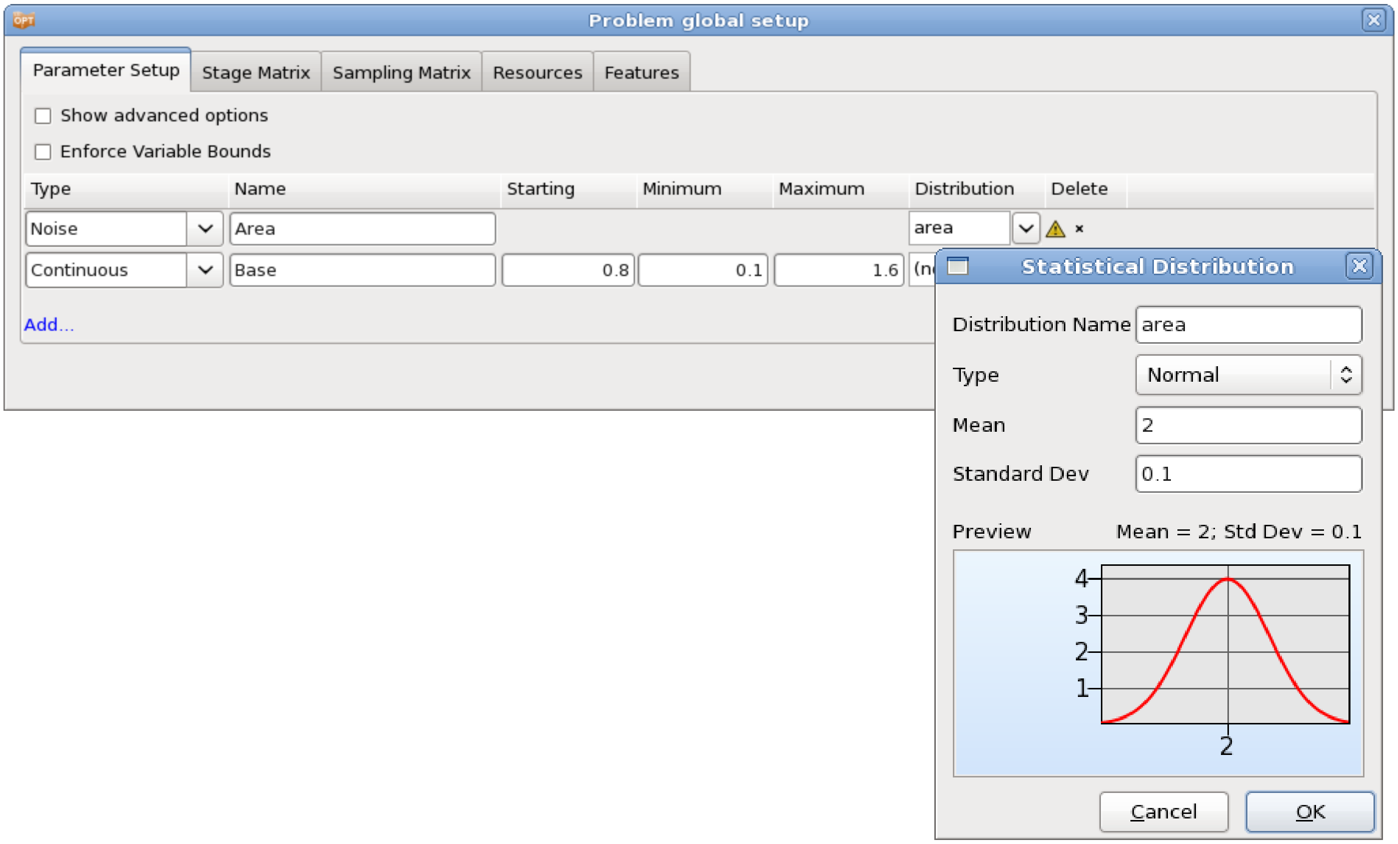
Strategy for Metamodel-based Task

○ Single Iteration
○ Sequential
● Sequential with Domain Reduction (SRSM)
○ Efficient Global Optimization (EGO)

1. Sampling points are added sequentially in an adaptive subregion.
2. Metamodel optimization is done at each iteration and is limited to the current subregion.
3. Suitable for finding a converged solution (e.g. system identification).
4. Generally unsuitable for global exploration.

□ Global Sensitivities
☑ Do verification run

Figure 22-31: Task Reliability based design optimization/Robust Parameter Design



□ Show advanced options

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	Area				area	×
Continuous	Base	0.8	0.1	1.6	(n	

Distribution Name: area

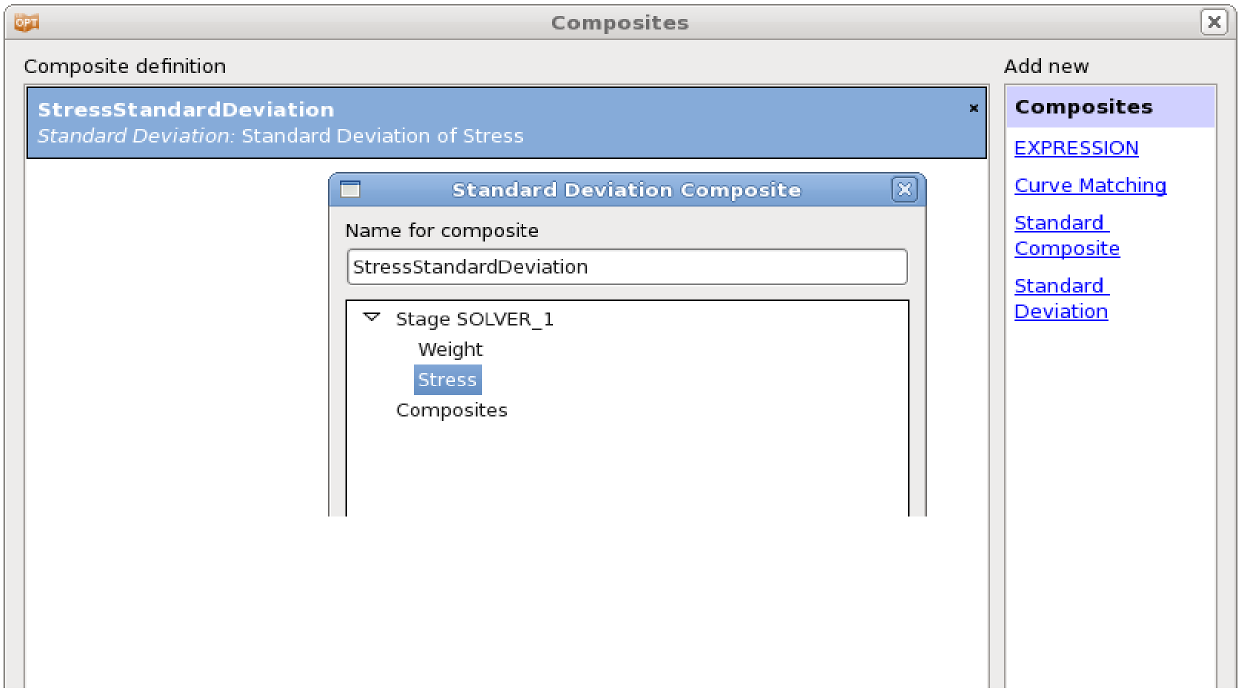
Type: Normal

Mean: 2

Standard Dev: 0.1

Preview: Mean = 2; Std Dev = 0.1

Figure 22-32: Parameter Setup and distribution



Composite definition

StressStandardDeviation

Standard Deviation: Standard Deviation of Stress

Name for composite: StressStandardDeviation

- Stage SOLVER_1
  - Weight
  - Stress
- Composites

Figure 22-33: Definition of Standard deviation composite

The actual stress response is shown in Figure 22-34. From the figure it can be seen that the `Base` variable must be set to values of larger than 0.4 to obtain a minimum variation of the stress considering that the design will then be in the flattest region of the response. A value of 0.5 is obtained in the optimization results as shown in Figure 22-35. Also shown in the optimization results is the design history of the stress standard deviation. Note that the standard deviation response stayed fairly insensitive to changes in the control variable after iteration 4 and that the initial subregion size for the `Base` variable was too large, resulting in initial increase in `Base` variable due to an inaccurate initial response surface.

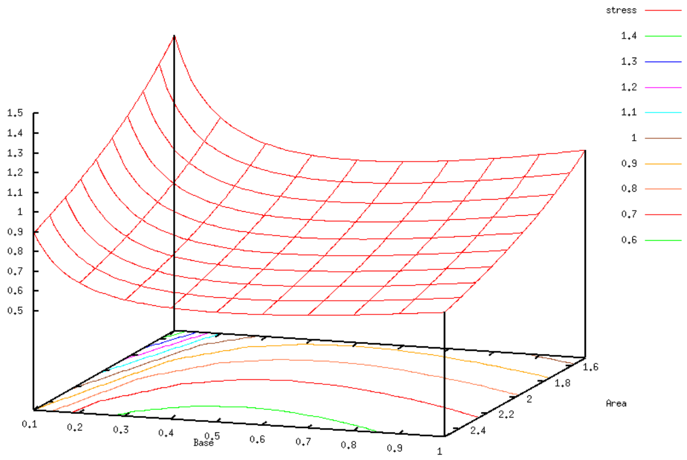


Figure 22-34: Contours of stress response. The flattest part of the response is when variable 'base' = 0.5.

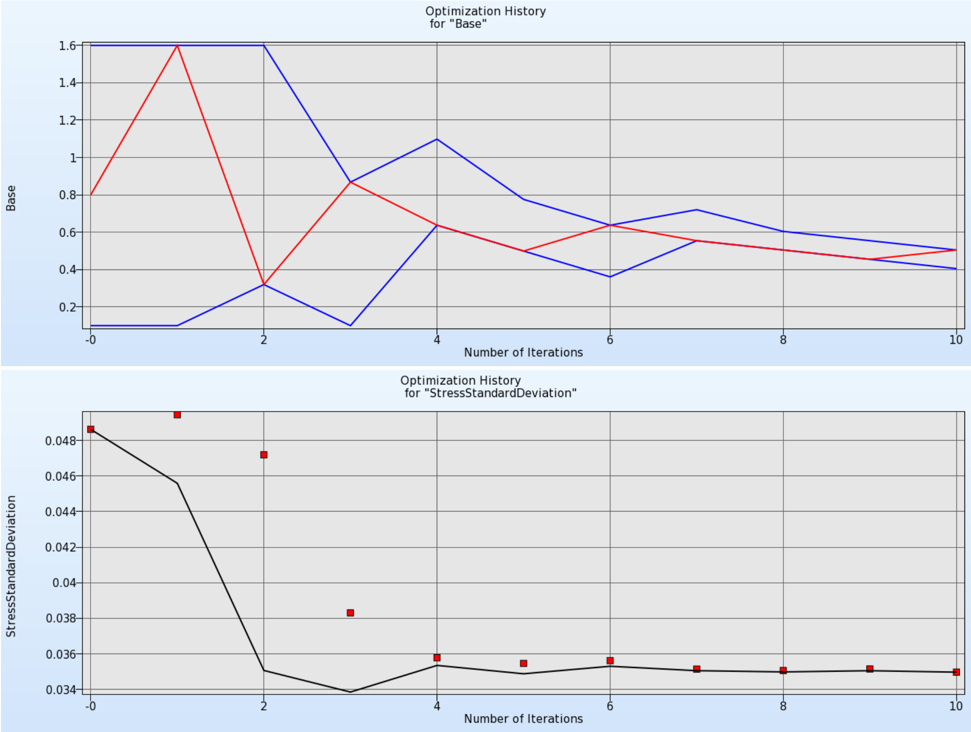


Figure 22-35: Optimization histories. Design variable `Base` is shown on the left and the standard deviation of the stress response is shown on the right.

22.5. Tolerance optimization

22.5.1. Overview

This example has the following features:

- Probabilistic analysis
- Monte Carlo analysis using imported metamodels
- Parametrization of probabilistic distribution parameters
- Extraction of probabilistic analysis results as responses
- Multilevel optimization
- Multi-objective optimization
- Tolerance optimization

22.5.2. Problem description

A simplified vehicle model is subjected to impact in this example. The model is the same as defined in Section 20.2. The goal is to optimize two thickness parameters for the parts `hood` and `bumper`, as well as their associated tolerance values to attain a balance between the design objectives and the robustness of the optimum.

The underlying deterministic optimization problem without considering the tolerances or the effect of uncertainties is:

$$\begin{aligned} &\min_{\mathbf{x}} \quad f_1 = Mass(\mathbf{x}) \\ &\text{s.t.} \\ &g_1(\mathbf{x}) = Intrusion(\mathbf{x}) - 550\,\text{mm} \le 0\,\text{mm} \\ &g_2(\mathbf{x}) = HIC(\mathbf{x}) - 250 \le 0 \\ &1 \le \mathbf{x} \le 5 \end{aligned} \qquad (22\text{-}2)$$

where ***x*** is a vector of design variables, which are the thicknesses `thood` and `tbumper` of the selected parts.

The solution of the above optimization may not be robust, as very often the optimum design lies at the constraint boundaries. Therefore, this issue is addressed by introducing tolerances into the problem. The nominal design variables are controlled so that the associated tolerance can be increased, thus making the design more robust with negligible probability/possibility of failure within the tolerance intervals.

This enhanced robustness may often come at the cost of other design objectives. Thus, the optimization formulation may consist of multiple competing objectives. In this example, the optimization is performed using two objectives. The nominal values of the mass are minimized while the relative tolerance is maximized. The final solution is a Pareto optimal front with a trade-off between the nominal mass and the relative tolerance.

For simplicity, both the thicknesses are assumed to have the same relative tolerance δ or `rel_tol`. It should be noted the optimization variables are the *nominal values* for the thickness, referred to as `nominal_th` and nominal_tb. Overall, the problem consists of three optimization variables - nominal_th, nominal_tb and δ . The tolerance-based optimization problem is:

$$\max_{\bar{\mathbf{x}},\delta} \quad \{\delta, -\bar{f}_1\}$$

s.t.

$$\begin{aligned} &\text{Probability[Intrusion} > 500] \le 10^{-8} && \text{for all } \mathbf{x} \in [\bar{\mathbf{x}}(1-\delta), \bar{\mathbf{x}}(1+\delta)] \\ &\text{Probability[HIC} > 250] \le 10^{-8} && \text{for all } \mathbf{x} \in [\bar{\mathbf{x}}(1-\delta), \bar{\mathbf{x}}(1+\delta)] \end{aligned} \tag{22-3}$$

In the above equation, δ is the relative tolerance, $\overline{f_1}$ represents the nominal mass of the design parts (negative sign indicates minimization), and $\bar{\mathbf{X}}$ is the nominal design. The constraints on intrusion and HIC must be satisfied at all possible designs within the tolerance interval (i.e. $\forall\, \mathbf{x} \in [\bar{\mathbf{x}}(1-\delta), \bar{\mathbf{x}}(1+\delta)]$).

An important feature to note is that the probabilistic constraint functions require the calculation of conditional probabilities with fixed nominal variable values $\bar{\boldsymbol{x}}$ and δ. This can be achieved using a simple Monte Carlo analysis at that fixed nominal design, as fixing the tolerance δ also fixes the distribution bounds. This is demonstrated in Section 22.5.3.

Calculation of failure probability using Monte Carlo analysis is a subproblem of the tolerance optimization problem in Equation (22-3). In Section 22.5.4, the tolerance optimization using a two-level setup is demonstrated. Here the Monte Carlo analysis forms the inner level and is performed iteratively for different combinations of nominal design and tolerance.

In both Section 22.5.3 and 22.5.4, Monte Carlo analysis is performed using pre-constructed global metamodels to reduce the computational cost associated with the calculation of conditional probabilities.

22.5.3. Imported metamodel-based Monte Carlo analysis with a fixed tolerance

In this section the goal is to determine the feasibility of a particular design configuration, with nominal values nominal_th = 1.9, nominal_tb = 3, within a 2% tolerance interval (rel_tol = 0.02). The design has uncertainties associated with it and therefore is considered as feasible only if there is not even a single failure among all the possible configurations within the tolerance interval around the nominal design. It is assumed that accurate global metamodels, previously constructed using LS-OPT, are already available for the required responses and there is no need for additional finite element simulations. The metamodel database is available as an XML file which, for the purpose of this example, is named *DesignFunctionsGlobal_PoleCrash*. Additionally, both the thickness parameters are assumed to have uniform distributions. Thus, the distributions of the two random variables thood and tbumper are as shown in Figure 22-36.

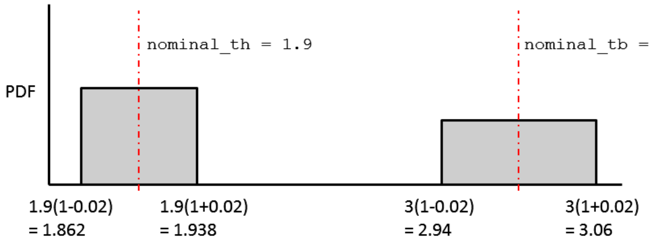
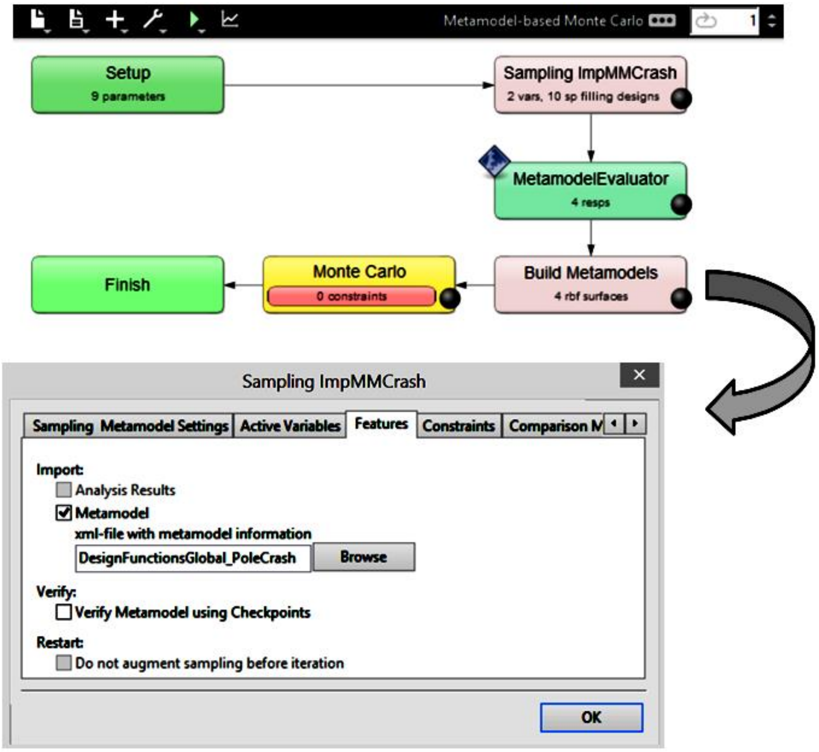


Figure 22-36: Distribution of thood and tbumper based on fixed nominal values and tolerance.

The steps to set up a Monte Carlo analysis problem using an imported metamodel are:

1. Start a new LS-OPT project with metamodel-based Monte Carlo analysis as the task type.
2. Specify the file *DesignFunctionsGlobal_PoleCrash* containing the pre-constructed metamodels using the *Import Metamodel* feature available in the *Build Metamodels* box. Once the file is parsed, the design parameters, responses and the metamodel type are automatically identified in LS-OPT (Figure 22-37).



Sampling ImpMMCrash

Sampling Metamodel Settings | Active Variables | Features | Constraints | Comparison M

Import:

□ Analysis Results

☑ Metamodel

xml-file with metamodel information

DesignFunctionsGlobal_PoleCrash Browse

Verify:

□ Verify Metamodel using Checkpoints

Restart:

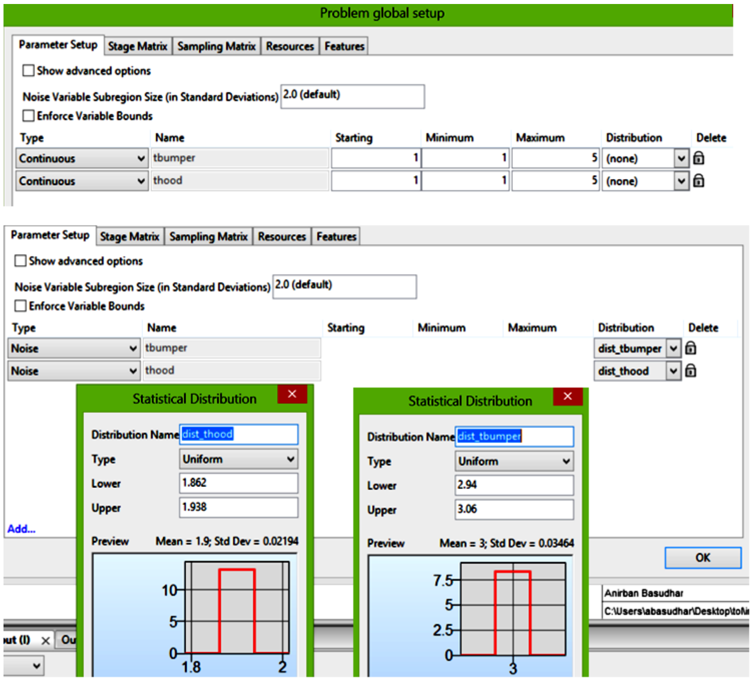
□ Do not augment sampling before iteration

OK

Figure 22-37: Monte Carlo analysis using imported metamodels

3. The part thickness design variables (`tbumper` and `thood`) parsed through the imported metamodel are defined as noise variables with uniform distributions (Figure 22-38 bottom). As the tolerance and nominal values are fixed, the distributions are also fixed as shown in Figure 22-36.
4. However, if one has say 10 different design alternatives (combinations of `nominal_tb`, `nominal_th` and `rel_tol`), there will be 10 different distributions for each noise variable. Thus, for each of the 10 combinations of `nominal_tb`, `nominal_th` and `rel_tol`, the user will need to calculate the lower and upper bounds to define the distribution. This manual work can be avoided by parametrizing the distribution so that the upper and lower bound are automatically calculated in LS-OPT for any given set of the values of the nominal thickness parameters and the relative tolerance. This is achieved using the "*&*" operator (Figure 22-39). The parameterization is done with respect to dependents `tb_l`, `tb_u`, `th_l` and `th_u`, which are defined as functions of manually added constants `nominal_th`, `nominal_tb` and `rel_tol` (δ) with suitable values (1.9, 3 and 0.02 respectively).

5. Parametrization of the distributions is optional for a single Monte Carlo analysis, but it is essential while performing an automated iterative tolerance optimization, as explained in Section 22.5.4 and Section 29.10.



Problem global setup

Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

□ Show advanced options

Noise Variable Subregion Size (in Standard Deviations) 2.0 (default)

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Continuous	tbumper	1	1	5	(none)	
Continuous	thood	1	1	5	(none)	

Variable definitions in the metamodel import file

Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

□ Show advanced options

Noise Variable Subregion Size (in Standard Deviations) 2.0 (default)

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	tbumper				dist_tbumper	
Noise	thood				dist_thood	

Statistical Distribution

Distribution Name dist_thood
Type Uniform
Lower 1.862
Upper 1.938
Preview Mean = 1.9; Std Dev = 0.02194

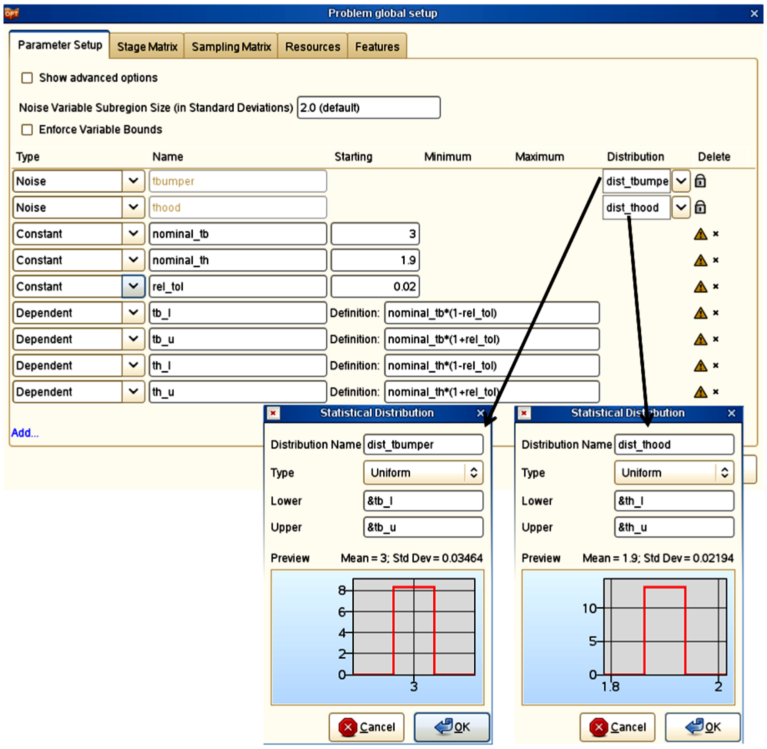
Statistical Distribution

Distribution Name dist_tbumper
Type Uniform
Lower 2.94
Upper 3.06
Preview Mean = 3; Std Dev = 0.03464

Anirban Basudhar
C:\Users\abasudhar\Desktop\toN

Modified variables and their distributions based on nominal_th = 1.9, nominal_tb = 3 and relative tolerance δ = 0.02 (2%)

Figure 22-38: Fixed variable distributions based on constant mean and tolerance



□ Show advanced options

Noise Variable Subregion Size (in Standard Deviations) 2.0 (default)

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	tbumper				dist_tbumpe	
Noise	thood				dist_thood	
Constant	nominal_tb	3				
Constant	nominal_th	1.9				
Constant	rel_tol	0.02				
Dependent	tb_l	Definition: nominal_tb*(1-rel_tol)				
Dependent	tb_u	Definition: nominal_tb*(1+rel_tol)				
Dependent	th_l	Definition: nominal_th*(1-rel_tol)				
Dependent	th_u	Definition: nominal_th*(1+rel_tol)				

Statistical Distribution

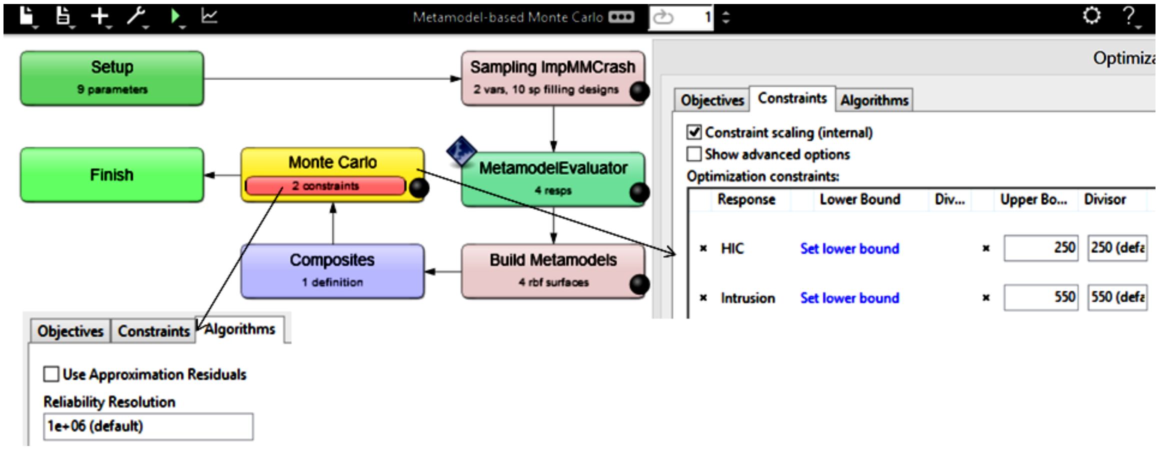
Distribution Name dist_tbumper
Type Uniform
Lower &tb_l
Upper &tb_u
Preview Mean = 3; Std Dev = 0.03464

Statistical Distribution

Distribution Name dist_thood
Type Uniform
Lower &th_l
Upper &th_u
Preview Mean = 1.9; Std Dev = 0.02194

__Figure 22-39: Parameterized variable distributions. As the distributions are parametrized, changing the values of the constants automatically updates the distributions.__

6. This example consists of two upper bound constraints on the intrusion and the HIC (Equation (22-2)).The intrusion needs to be defined as a composite `Disp1-Disp2`, defined in terms of two displacement responses `Disp1` and `Disp2` that are directly obtained from the metamodel import file. The constraints for the Monte Carlo analysis are then defined (Figure 22-40). The number of Monte Carlo samples used for calculations based on the imported metamodel are defined under *Algorithms* (Figure 22-40).



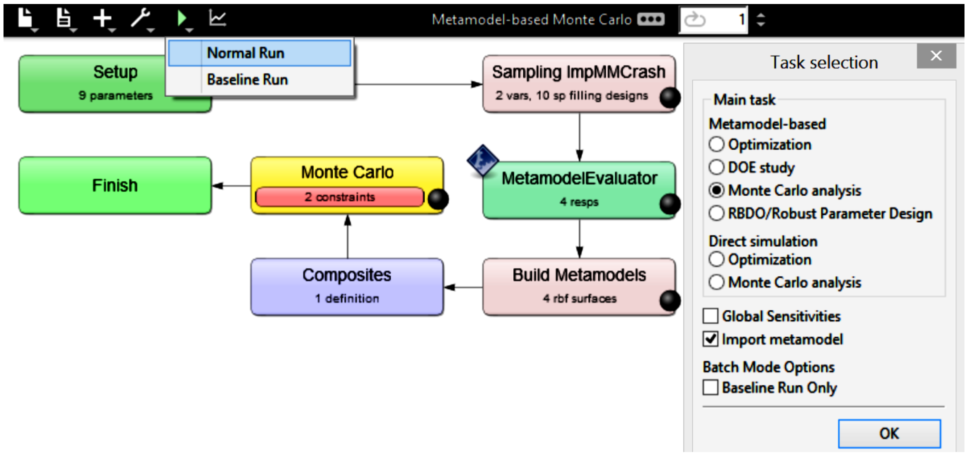
☑ Constraint scaling (internal)
□ Show advanced options
Optimization constraints:

Response	Lower Bound	Div...	Upper Bo...	Divisor
HIC	Set lower bound		250	250 (defa
Intrusion	Set lower bound		550	550 (defa

□ Use Approximation Residuals
Reliability Resolution
1e+06 (default)

__Figure 22-40: Monte Carlo analysis setup. The number of Monte Carlo samples used for calculations based on the imported metamodel are defined as the Reliability Resolution.__

7. The next step is to select the checkbox option *Import Metamodel* in the Task selection menu, followed by a *Normal Run* to perform the Monte Carlo analysis (Figure 22-41). The analysis in this example is done by calculating the responses at 10^6 Monte Carlo samples based on the imported metamodels.



Task selection

Main task

Metamodel-based

○ Optimization

○ DOE study

● Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

□ Global Sensitivities

☑ Import metamodel

Batch Mode Options

□ Baseline Run Only

Figure 22-41: Monte Carlo analysis run with imported metamodel.

The frequency histograms for the two constraint functions are plotted in Figure 22-42, which shows that with mean values `nominal_tb = 3` and `nominal_th = 1.9` there is no failure within the 2% tolerance interval.

Remark

An alternate way to perform the 5th step, instead of selecting the task selection checkbox option Import Metamodel, is to import the metamodels in the DesignFunctionsGlobal_PoleCrash file through a repair operation (Repair → Import Metamodels) before running the metamodel-based Monte Carlo analysis task using a Normal Run. However, such an approach is a two-step process that requires manual intervention to perform the Normal Run after importing the metamodels. This approach is therefore not feasible when the Monte Carlo analysis is a subproblem within a tolerance optimization, during which several Monte Carlo analyses at different design configurations are performed (Section 22.5.4).

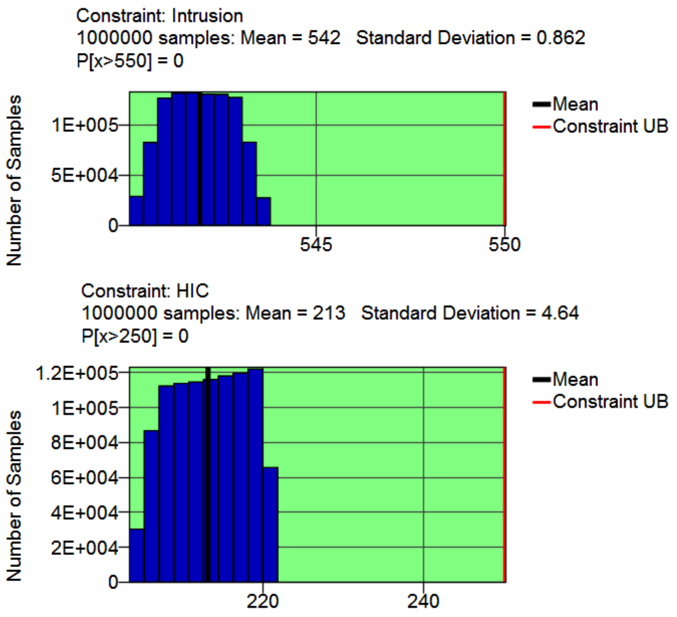


Figure 22-42: Frequency histogram of HIC and Intrusion with feasibility information. The green background shows that the designs are feasible within the tolerance interval.

22.5.4. Tolerance optimization setup and results

In Section 22.5.3 the calculation of probability of failure using a fixed mean design and tolerance value, under the assumption of uniform distribution, was presented. Having fixed the mean variable values and the tolerance, the probability calculated was essentially a conditional probability for those particular values. Going back to Equation (22-3), the same conditional probabilities are part of the constraint definition in the tolerance optimization problem. Thus, the example presented in Section 22.5.3 is a subproblem of the tolerance optimization formulation defined in Equation (22-3). In LS-OPT, such problems can be addressed using a multilevel framework (Section 17.7). The setup and results for the tolerance optimization problem in Equation (22-3) are presented in this section. The LS-OPT setup for tolerance optimization consists of two levels. The basic two level tolerance optimization method is presented in Figure 22-43.

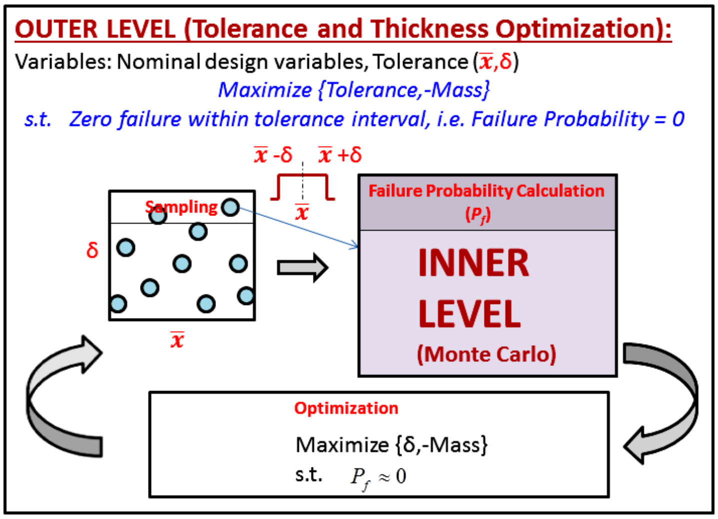

OUTER LEVEL (Tolerance and Thickness Optimization):
Variables: Nominal design variables, Tolerance ($\bar{x}$,δ)
Maximize {Tolerance,-Mass}
s.t. Zero failure within tolerance interval, i.e. Failure Probability = 0
$\bar{x}$ -δ
$\bar{x}$ +δ
$\bar{x}$
Sampling
δ
$\bar{x}$
Failure Probability Calculation (P_f)
INNER LEVEL
(Monte Carlo)
Optimization
Maximize {δ,-Mass}
s.t. $P_f \approx 0$

Figure 22-43: Two-level tolerance optimization methodology.

To reduce the cost of the two-level optimization, the inner level Monte Carlo analysis is performed using imported pre-constructed global metamodels as shown in Section 22.5.3. For that purpose a database of global metamodels needs to be created first. Thus, in summary, the complete solution method consists of two steps:

- *Step 1:* Single iteration metamodel-based optimization to construct the global metamodels and to obtain the deterministic optimum solution for Equation (22-2).
- *Step 2:* Two-level tolerance optimization using imported metamodels. In terms of the LS-OPT setup, the outer level is an optimization (direct optimization in this example) with the nominal thickness values and the associated relative tolerance(s) as the optimization variables and the inner level is an imported metamodel-based Monte Carlo analysis, which is carried out for each outer level sample $(\bar{\boldsymbol{x}}, \delta)$.

The detailed process with the setup steps of both the steps is explained below.

Step 1: Deterministic optimization

The first step consists of a deterministic single iteration metamodel-based optimization. The optimization formulation is given in Equation (22-2). The mass of the selected design parts is minimized subject to constraints on the intrusion and the HIC. The optimization variables are the thicknesses `thood` and `tbumper` of selected parts. It is also possible to perform a DOE task instead of optimization, as the main purpose of this step is to construct high fidelity metamodels of the design responses. These metamodels can be later utilized in lieu of numerous finite element analyses required in step 2.

The design problem is set up in LS-OPT using the single iteration metamodel-based optimization task type. Radial Basis Function networks is selected as the metamodel type with Space Filling as the sampling technique. A high number of design samples (200) were defined so that the resulting global metamodels have sufficient accuracy. The metamodel database *DesignFunctions.1* generated by LS-OPT is saved as *DesignFunctionsGlobal_PoleCrash.*

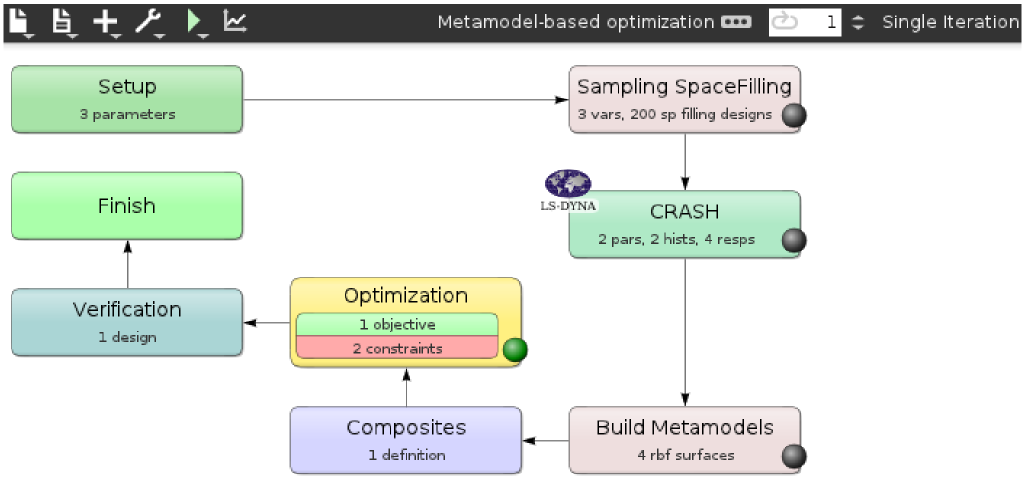


Figure 22-44: LS-OPT setup for Step 1 (global approximation and deterministic optimization)

The metamodel surface plot for the objective function Mass is shown in ***Figure 22-45*** along with the feasible and infeasible domains. Two things are apparent from the plot, which re-emphasize why the effect of uncertainties should be considered during design (such as in the form of tolerance):

- The optimum `tbumper=1, thood=1.734` with mass 0.4554 lies at the boundary of the feasible domain. As a result, a slight perturbation may lead to failure.
- There is a tiny feasible island in the space, which may simply be due to noise or a local metamodeling inaccuracy. If the optimum were to lie in that island, such a design would not be robust at all. Again, a slight perturbation may lead to failure.

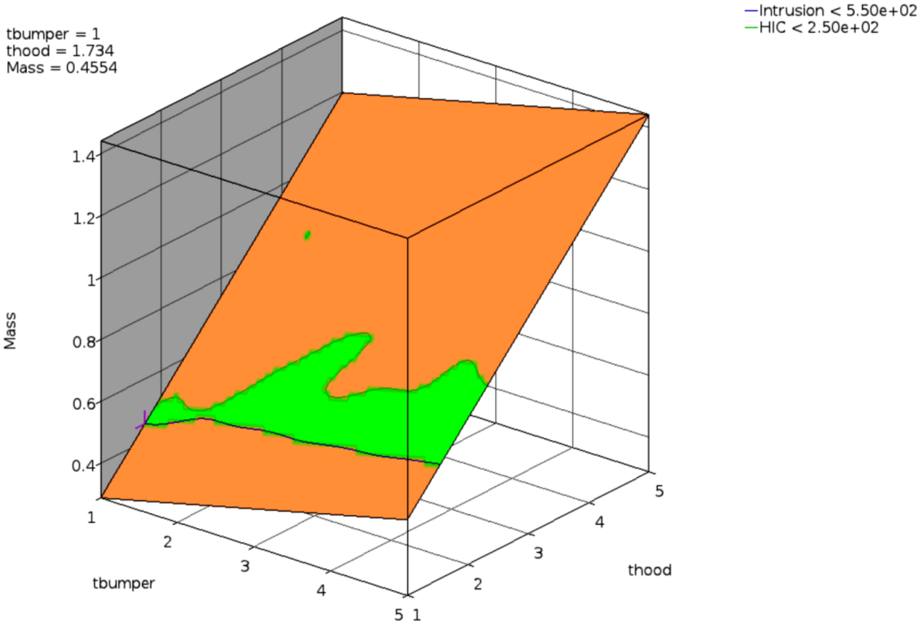


Figure 22-45: Surface plot of Mass. The green region is feasible. The magenta cross represents the optimum.

Step 2: Optimization of thickness and relative tolerance using multilevel setup

While the goal of step 1 was mainly to provide the global approximation to reduce the cost of the overall design process, Step 2 describes the actual tolerance optimization. In summary, the full setup has two levels:

- Outer level:

  Task type: Direct optimization (Equation (22-3)) with population size 100 (direct optimization is used, but metamodels can also be used) and 50 generations.

  Control Variables: nominal_th, nominal_tb, rel_tol (δ)

  Responses: Probability of failure, nominal values (LS-OPT type responses)

  Objectives: Maximize `rel_tol`, Minimize `nominal_mass`

  Constraints: Probabilities of failure ≈ 0 (upper bound lower than the reliability resolution of inner level)

  $$\text{Probabilit y[Intrusio n} > 500] \le 10^{-8} \qquad \text{for all } \mathbf{x} \in \left[\bar{\mathbf{x}}(1-\delta), \bar{\mathbf{x}}(1+\delta)\right]$$

  $$\text{Probabilit y[HIC} > 250] \le 10^{-8} \qquad \text{for all } \mathbf{x} \in \left[\bar{\mathbf{x}}(1-\delta), \bar{\mathbf{x}}(1+\delta)\right]$$

- Inner level:

  Task type: Metamodel-based Monte Carlo at each outer level sample using imported metamodels from *step 1* (Section 22.5.3).

  Noise Variables: `thood, tbumper` (distributions of `thood` and `tbumber` defined as functions of outer level variables)

  Responses: HIC, displacements, mass etc. (Imported metamodel responses)

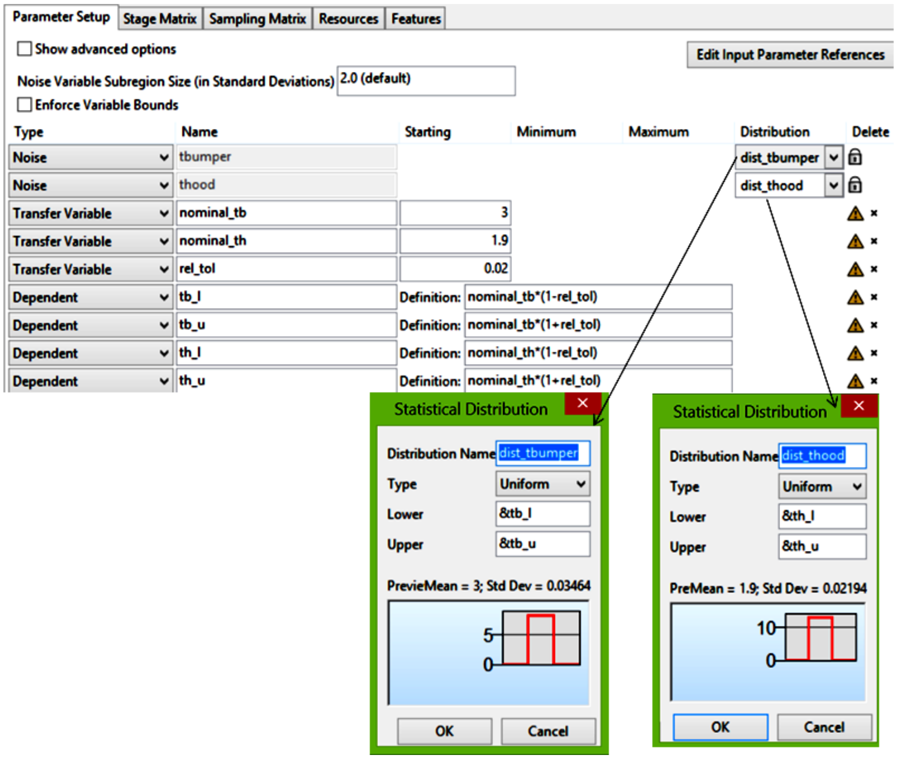
  Constraints: HIC < 250, Intrusion < 550 mm

A detailed step by step procedure of setting up the two level problem in LS-OPT is demonstrated in the following sections. First the inner level is set up, followed by the outer level. The inner level setup is an input file for an LS-OPT type stage in the outer level (Figure 22-46).

- Inner Level Setup (Monte Carlo Analysis):

  Follow the same steps as in Section 22.5.3. The constraint definition needs to be modified however in order to conform to the outer level problem in Equation (22-3).

  - Follow the exact same steps as 1 and 2 in Section 22.5.3 to import the existing metamodels from DesignFunctionsGlobal_PoleCrash.
  - In step3, make sure to parametrize the distributions as in Figure 22-39. However, the constants `nominal_tb, nominal_th` and `rel_tol` (δ) need to be changed to *Transfer Variables*. They can be assigned any numeric value without any effect, as these values are eventually overwritten by the outer level samples. This is explained further in Section 22.5.5.



☐ Show advanced options

Noise Variable Subregion Size (in Standard Deviations) 2.0 (default)

☐ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	tbumper				dist_tbumper	
Noise	thood				dist_thood	
Transfer Variable	nominal_tb	3				×
Transfer Variable	nominal_th	1.9				×
Transfer Variable	rel_tol	0.02				×
Dependent	tb_l	Definition: nominal_tb*(1-rel_tol)				×
Dependent	tb_u	Definition: nominal_tb*(1+rel_tol)				×
Dependent	th_l	Definition: nominal_th*(1-rel_tol)				×
Dependent	th_u	Definition: nominal_th*(1+rel_tol)				×

Statistical Distribution

Distribution Name dist_tbumper
Type Uniform
Lower &tb_l
Upper &tb_u

PrevieMean = 3; Std Dev = 0.03464

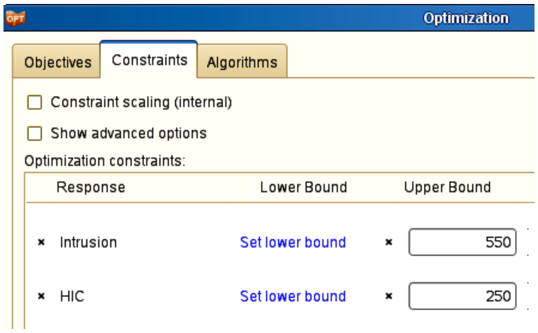
Statistical Distribution

Distribution Name dist_thood
Type Uniform
Lower &th_l
Upper &th_u

PreMean = 1.9; Std Dev = 0.02194

Figure 22-47: Variable distributions parameterized using Transfer Variables. These variables are treated as constants in the inner level Monte Carlo analysis, but may be optimization variable in an outer level (Section 19.7). The values of the Transfer Variables are set as the outer level samples variable values. Thus, for a specific outer level sample the Transfer Variables and their dependents (nominal variables and variable distribution ranges) are fixed, and the failure probabilities calculated in the inner level using these fixed distributions corresponds to that specific outer level sample.

- ○ Follow the same step 4 as in Section 22.5.3 to define the constraints..



☐ Constraint scaling (internal)

☐ Show advanced options

Optimization constraints:

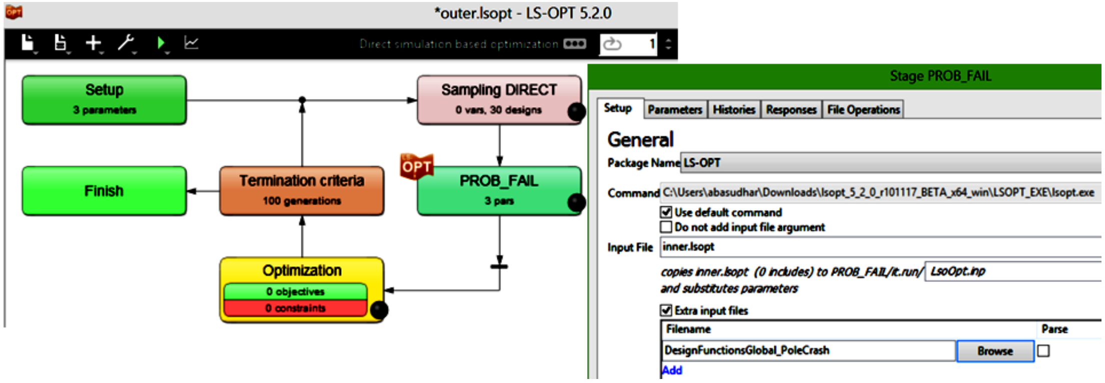
Response	Lower Bound	Upper Bound
Intrusion	Set lower bound	550
HIC	Set lower bound	250

Figure 22-48: Inner level constraints for Monte Carlo Analysis.

- ○ Follow the same step 5 as in Section 22.5.3 to select the checkbox option Import Metamodel in the Task selection menu. However, instead of performing a Normal Run, simply save the inner level setup by any name, say inner.lsopt.

○ Outer Level Setup (Optimization):

- Start a new LS-OPT project with direct optimization as the task type.
- Select LS-OPT as the solver package in the Stage box. Click on ‘use default command’ check box, this will change the command to the full path of the LS-OPT executable. Using the browse option, select the inner level LS-OPT project file inner.lsopt as the input file of the solver. Since the file DesignFunctionsGlobal_PoleCrash is needed in the inner level for metamodel import, this file should be transferred to the outer level run directories using the ‘Extra input files’ option with parsing checked off.



Stage PROB_FAIL

Setup | Parameters | Histories | Responses | File Operations

General

Package Name LS-OPT

Command C:\Users\abasudhar\Downloads\lsopt_5_2_0_r101117_BETA_x64_win\LSOPT_EXE\lsopt.exe

☑ Use default command

☐ Do not add input file argument

Input File inner.lsopt

copies inner.lsopt (0 includes) to PROB_FAIL/it.run/ LsoOpt.inp
and substitutes parameters

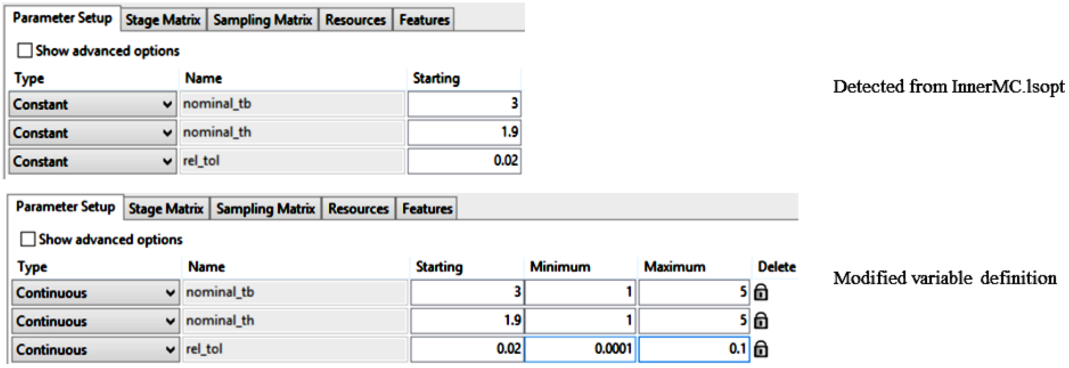
☑ Extra input files

Filename		Parse
DesignFunctionsGlobal_PoleCrash	Browse	☐

Add

Figure 22-49: Outer level stage setup.

- Once the inner level LS-OPT input file is parsed, the transfer variables of the inner level `nominal_tb`, `nominal_th` and `rel_tol` are displayed in the parameter Setup box of the outer level. Define these parameters as continuous variables and specify the upper and lower bounds for each variable.


Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

☐ Show advanced options

Type	Name	Starting
Constant	nominal_tb	3
Constant	nominal_th	1.9
Constant	rel_tol	0.02

Detected from InnerMC.lsopt

Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

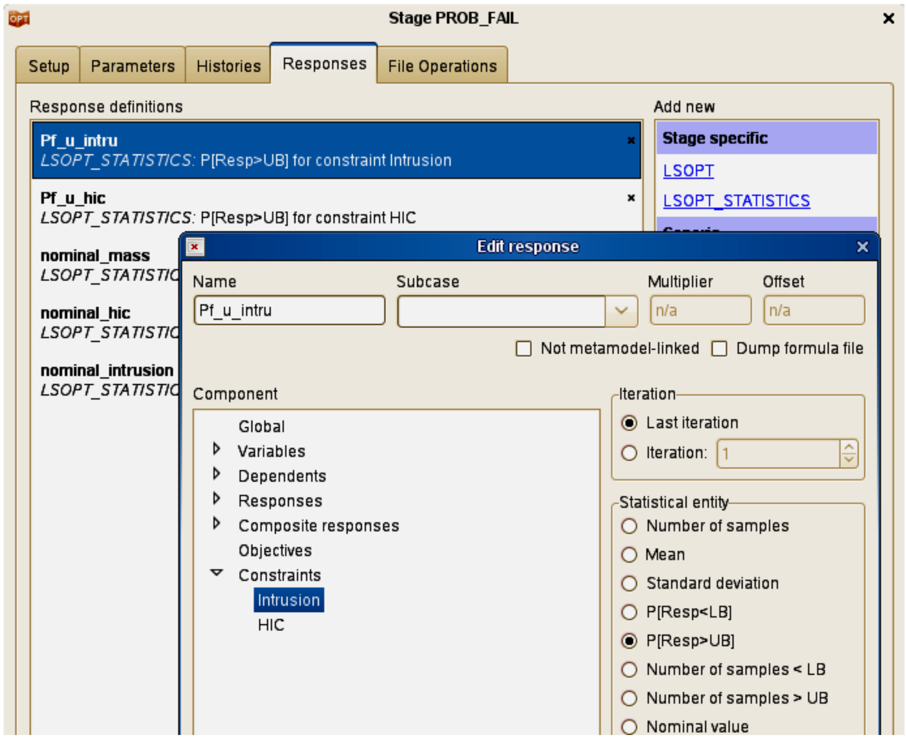
☐ Show advanced options

Type	Name	Starting	Minimum	Maximum	Delete
Continuous	nominal_tb	3	1	5	🔒
Continuous	nominal_th	1.9	1	5	🔒
Continuous	rel_tol	0.02	0.0001	0.1	🔒

Modified variable definition

Figure 22-50: Outer level global variable setup.

- The next step is to define the outer level responses. The statistics from the inner level Monte Carlo analysis can be extracted using LS-OPT Statistics response definition. In this example, the probability of failure of constraints of the inner level Monte Carlo analysis and the response values of the nominal designs are extracted. Therefore, the probability of exceeding the upper bounds and nominal values of intrusion and HIC are defined as outer level responses. It is important to note that each outer level sample corresponds to the nominal design of the inner level. The following figure shows the selection of the probability of failure of intrusion and other outer level response definitions.



Stage PROB_FAIL

Setup | Parameters | Histories | Responses | File Operations

Response definitions

- **Pf_u_intru** — *LSOPT_STATISTICS:* P[Resp>UB] for constraint Intrusion
- **Pf_u_hic** — *LSOPT_STATISTICS:* P[Resp>UB] for constraint HIC
- **nominal_mass** — *LSOPT_STATISTIC*
- **nominal_hic** — *LSOPT_STATISTIC*
- **nominal_intrusion** — *LSOPT_STATISTIC*

Add new: Stage specific — LSOPT — LSOPT_STATISTICS

Edit response

Name: Pf_u_intru | Subcase: | Multiplier: n/a | Offset: n/a

□ Not metamodel-linked □ Dump formula file

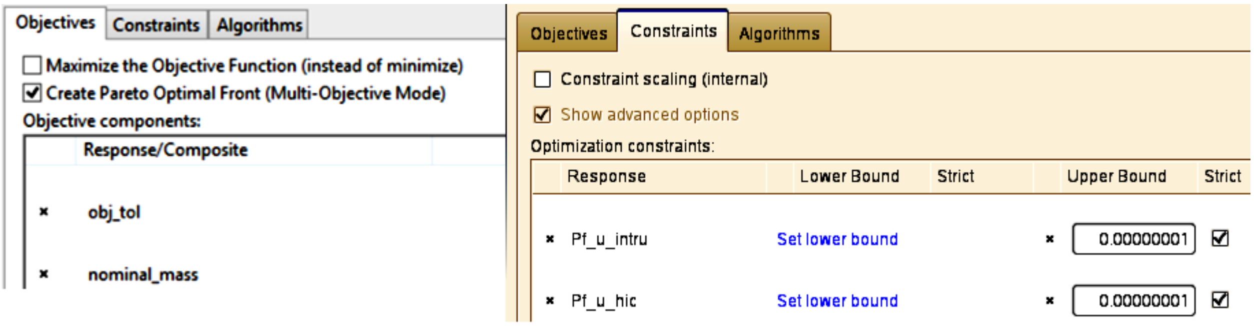
Component: Global, Variables, Dependents, Responses, Composite responses, Objectives, Constraints (Intrusion, HIC) — Intrusion selected

Iteration: ● Last iteration ○ Iteration: 1

Statistical entity:
○ Number of samples
○ Mean
○ Standard deviation
○ P[Resp<LB]
● P[Resp>UB]
○ Number of samples < LB
○ Number of samples > UB
○ Nominal value

Figure 22-51: Outer level response definitions

- These probabilistic responses are later selected as outer level design constraints with an upper bound close to zero (less than the reliability resolution of the inner level Monte Carlo analysis), as shown in the figure below. Maximizing the tolerance and minimizing the nominal mass are the outer level design objectives. As the tolerance is maximized, the negative of tolerance variable can be defined as a composite response (*obj_tol*) using the *Expression composite*. The objective functions to be minimized are *obj_tol* and *nominal_mass*. The '*Create Pareto Optimal Front'* option is selected.



Objectives | Constraints | Algorithms

□ Maximize the Objective Function (instead of minimize)
☑ Create Pareto Optimal Front (Multi-Objective Mode)

Objective components:

	Response/Composite
×	obj_tol
×	nominal_mass

Objectives | Constraints | Algorithms

□ Constraint scaling (internal)
☑ Show advanced options

Optimization constraints:

	Response	Lower Bound	Strict		Upper Bound	Strict
×	Pf_u_intru	Set lower bound		×	0.00000001	☑
×	Pf_u_hic	Set lower bound		×	0.00000001	☑

Figure 22-52: Outer level design objectives and constraints.

- The final step is to specify the required population size and termination criteria for the GA based multiobjective optimization. The overall setup of the outer level is shown in the figure below.

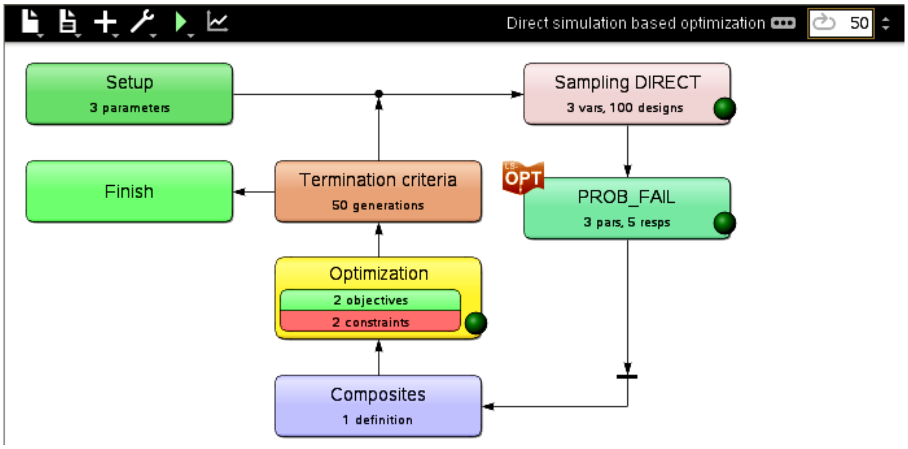


Figure 22-53: LS-OPT setup for outer level multiobjective optimization.

Results

The optimization problem discussed above was solved with a population size of 100 samples for 50 generations. The multiobjective optimization resulted in a set of Pareto optimal designs with mass of the design parts varying from 0.456 to 0.472 units with a maximum tolerance of 0.037. Therefore, a robust design with 3.7% tolerance was obtained with a mass increase of approximately 0.016 units or 3.5%. Since the Pareto optimal designs were based on metamodels, these points were subjected to LS-DYNA analysis to obtain the simulation-based results. The Pareto front of the final generation based on both metamodels and LS-DYNA analysis is shown in Figure 22-54. It is important to note that all the simulation-based optimal designs are feasible, indicating good accuracy of the metamodel obtained in step1.

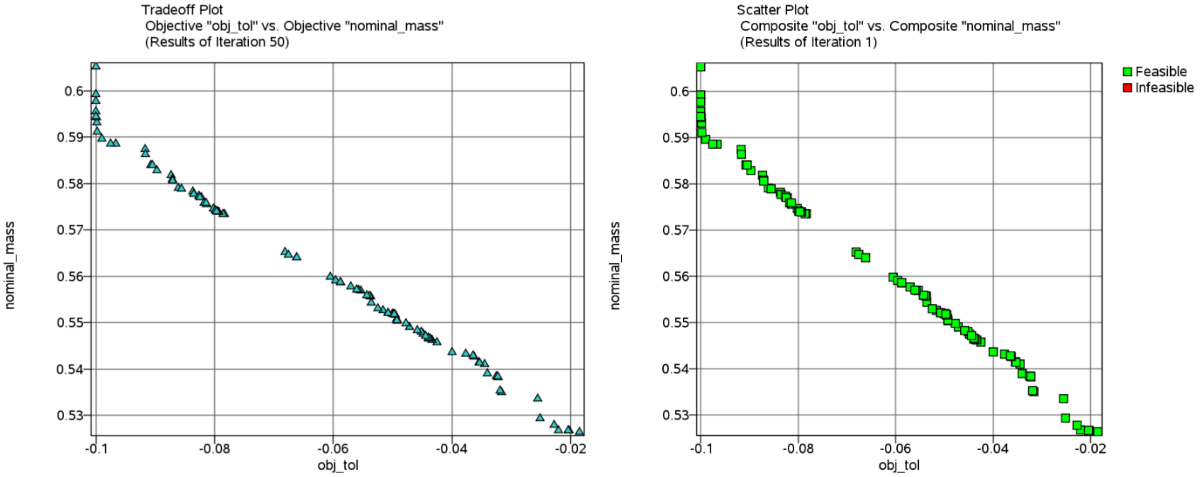


Figure 22-54: Metamodel-based (left) and simulation-based (right) tradeoff.

Figure 22-55 shows the tradeoff between the two objectives and the evolution of the Pareto front over 50 generations.

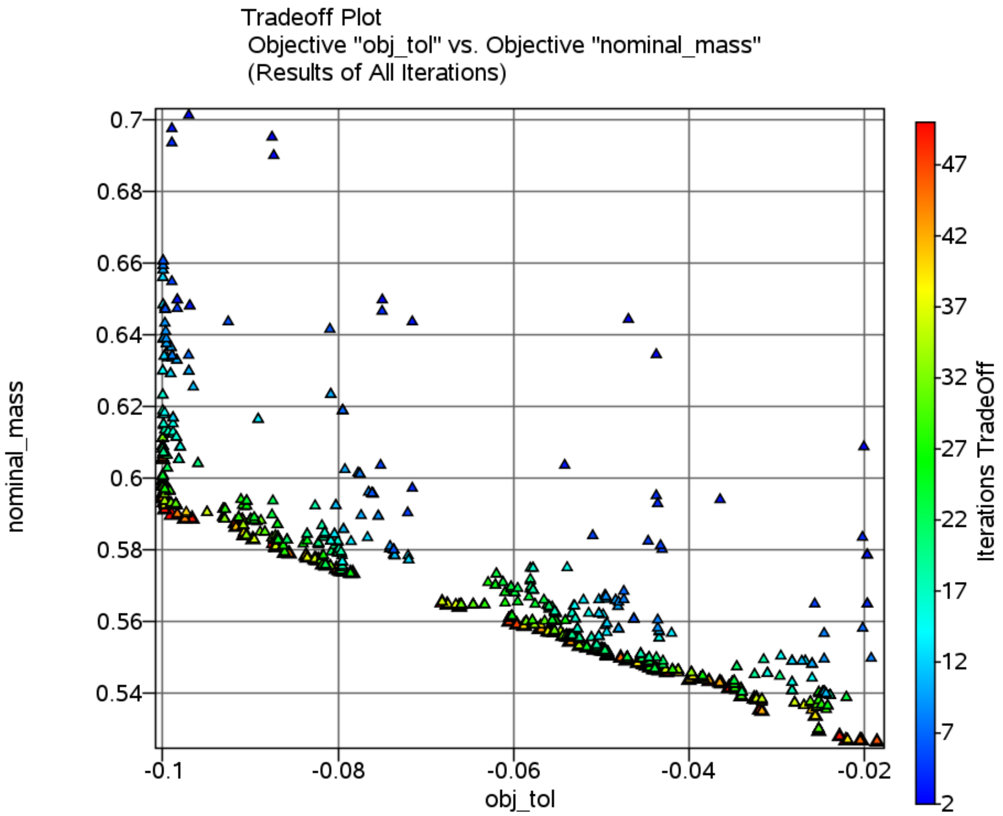


Figure 22-55: Tradeoff between mass and tolerance objectives for all iterations.

Table 22-4: Summary of the values of design variables, objectives and constraints for baseline, deterministic optimum and tolerance-based optimum (at maximum tolerance).

Design point	**tbumper**	**thood**	**rel_tol**	**Mass**	**HIC**	**Intrusion**	**Max Constraint Violation**
Baseline	3	1	-	0.41	68.02	575.7	25.7
Deterministic optimum	1	1.75	-	0.456	222.6	549.4	0
Tolerance-based optimum with maximum tolerance	2.63	1.95	0.1	0.6	229.8	541.6	0

Remark

Sometimes it may desirable to constrain the objective function (mass in this example) also in Equation (22-3). This is however not always necessary. For details one may refer to [1].

22.5.5. Comparison of RBDO and Tolerance Optimization

Tolerance optimization is similar to RBDO in several ways – both involve the minimization of one or more objective functions subject to probabilistic constraints. The variable distributions in tolerance optimization are truncated, and thus, tolerance optimization is similar to RBDO with truncated variable distributions. The main difference lies in the fact that the tolerance value associated with a variable is also optimized here. As a result, the range of the truncated distribution changes during tolerance optimization. This is in contrast to RBDO, in which the range of the variable distribution is known a priori, and only the mean value changes during optimization (also see Section 29.10).

As the variable distribution is fixed a priori in RBDO, calculation of the failure probability at any sample is done using distributions having the same range around the sample. On the other hand, during tolerance optimization the tolerance value is also a variable for the samples (outer level optimization variable), and therefore, the range of variable distribution is different for each sample. For a particular outer level sample, however, the tolerance value and the associated variable distribution range are fixed. Therefore, the failure probability for that sample is calculated using Monte Carlo analysis in the inner level using that fixed distribution.

The link between the outer and the inner level is established using Transfer Variables. Each outer level sample defines the value of the nominal design parameters and the tolerances. The same variables are defined as Transfer Variables in the inner level and treated as constants there. Thus, a specific outer level sample replaces the Transfer Variable values in the inner level. The distributions in the inner level are parametrized using the Transfer Variables, which have constant values in this level. Thus the distributions in the inner level are also fixed for a specific outer level sample, which enables the calculation of the failure probability for the outer level sample.

For a fixed value of tolerance (i.e. if the tolerance values are not optimized and instead feasibility within a pre-determined interval is enforced), the tolerance optimization problem reduces to RBDO with truncated variable distribution.

22.6. Using Stochastic Fields

This example demonstrates:

- Using a stochastic field in a Monte Carlo analysis
- Using a variable and a stochastic field in a Monte Carlo analysis
- Replicating experiments using stochastic fields
- Using fixed stochastic fields

The structure as shown in the Figure 22-56 is considered. This is the compression of a beam with geometrical imperfections modeled using a stochastic field. The result considered is the load at the end of the analysis as shown in the figure.

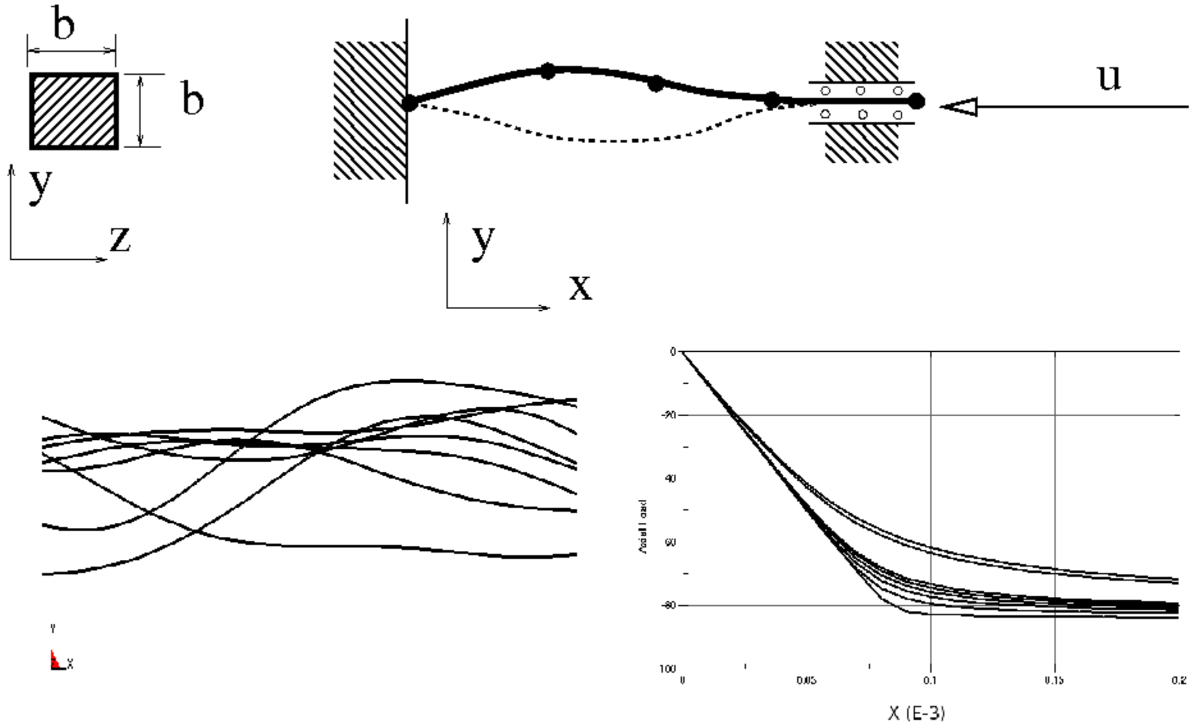


Figure 22-56: Problem with a stochastic field. The structural problem is shown in the top. In the bottom left are sample beams with the perturbation exaggerated by a factor 100, and the corresponding histories are shown in the bottom right.

The beam has a length of 20 and a Young's modulus of 2e8. It is analyzed in using 128 type 2 beam elements using an implicit analysis and 20 increments to compress the end a distance of -0.002.

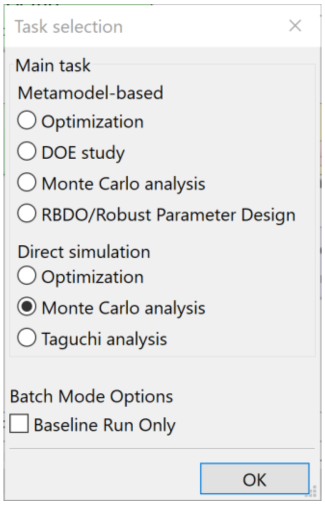
The perturbation is created using the spectral method to have an autocorrelation function described by a Gaussian correlation function. The Gaussian correlation function is $P(x) = e^{-(as)^2}$ with s distance and the constant a = 0.1 in this study. The resulting perturbation is scaled by 0.01. The LS-DYNA® `*PERTURABATION` card is:

```
*PERTURBATION_NODE
$type, nid, scl, cmp, icoord, cid
4, , 1.e-2, 3
$cstype, e1, e2, rnd
1, , ,
1, 1.e-1
```

22.6.1. Using only a stochastic field

Firstly, a Monte Carlo analysis is done (Figure 22-57) considering only the geometric stochastic fields. The stochastic field is set to vary freely using the LS-DYNA® keyword `*PERTURBATION`. Every LS-OPT analysis needs a variable, so we added a dummy variable that does not do anything, Figure 22-58. 50 simulations are run, Figure 22-59. The point selection is arbitrary since the variable value is not used. Note that it is possible to have a variable controlling the random seed in the LS-DYNA® `*PERTURBATION` keyword, which can be useful for many reasons, such as having only certain stochastic fields.

The histogram of the responses is shown in Figure 22-60. Note that the distribution has a characteristic shape.



Task selection

Main task

Metamodel-based

○ Optimization

○ DOE study

○ Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

● Monte Carlo analysis

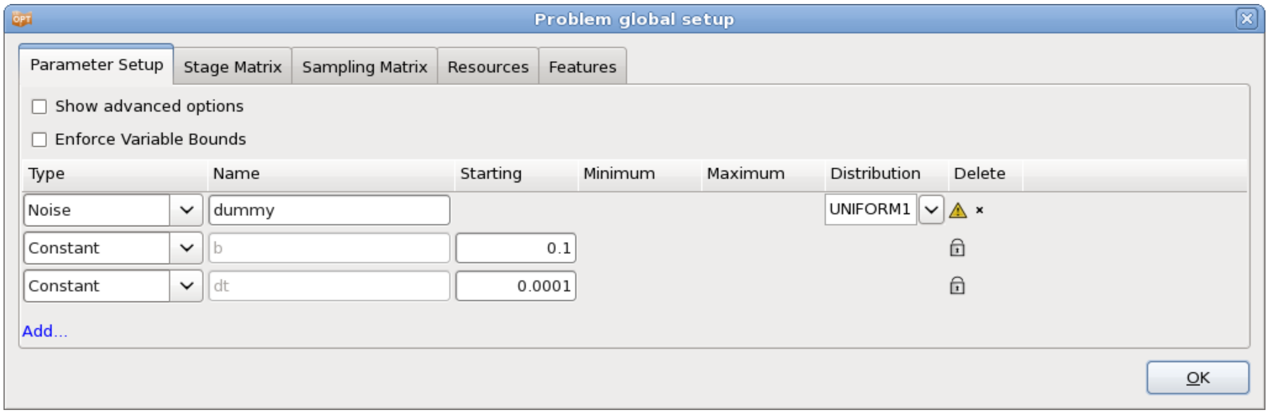
○ Taguchi analysis

Batch Mode Options

□ Baseline Run Only

OK

Figure 22-57: Task dialog. Select the main task Monte Carlo analysis.



Problem global setup

Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

□ Show advanced options

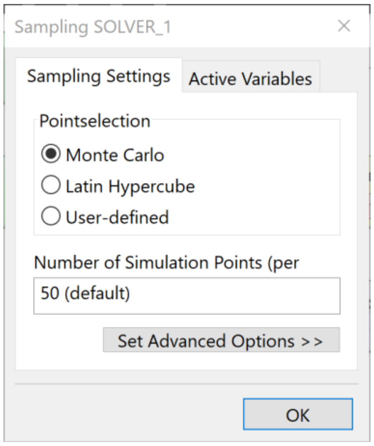
□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	dummy				UNIFORM1	⚠ ×
Constant	b	0.1				🔒
Constant	dt	0.0001				🔒

Add...

OK

Figure 22-58: Parameter setup. Since the variable dummy is not used, an arbitrary distribution can be used here.



Sampling SOLVER_1

Sampling Settings | Active Variables

Pointselection

● Monte Carlo
○ Latin Hypercube
○ User-defined

Number of Simulation Points (per

50 (default)

Figure 22-59: Sampling dialog. Specify the number of simulation points. The point selection doesn't affect the analysis here, since the variable is not used.

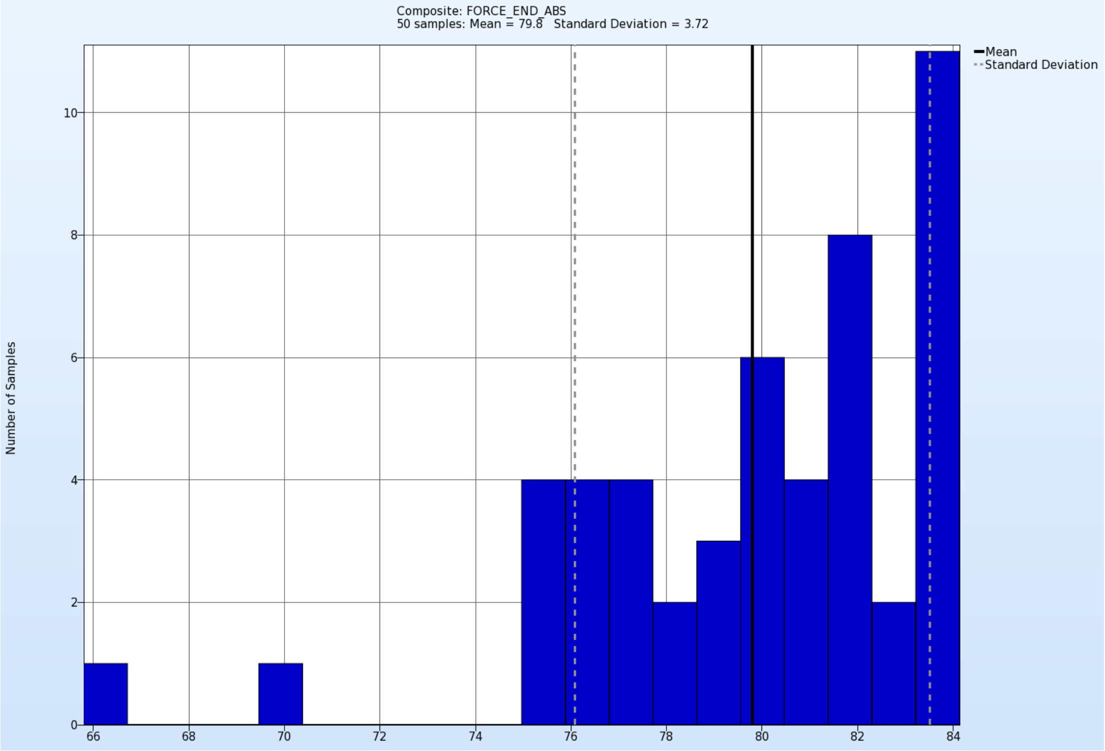
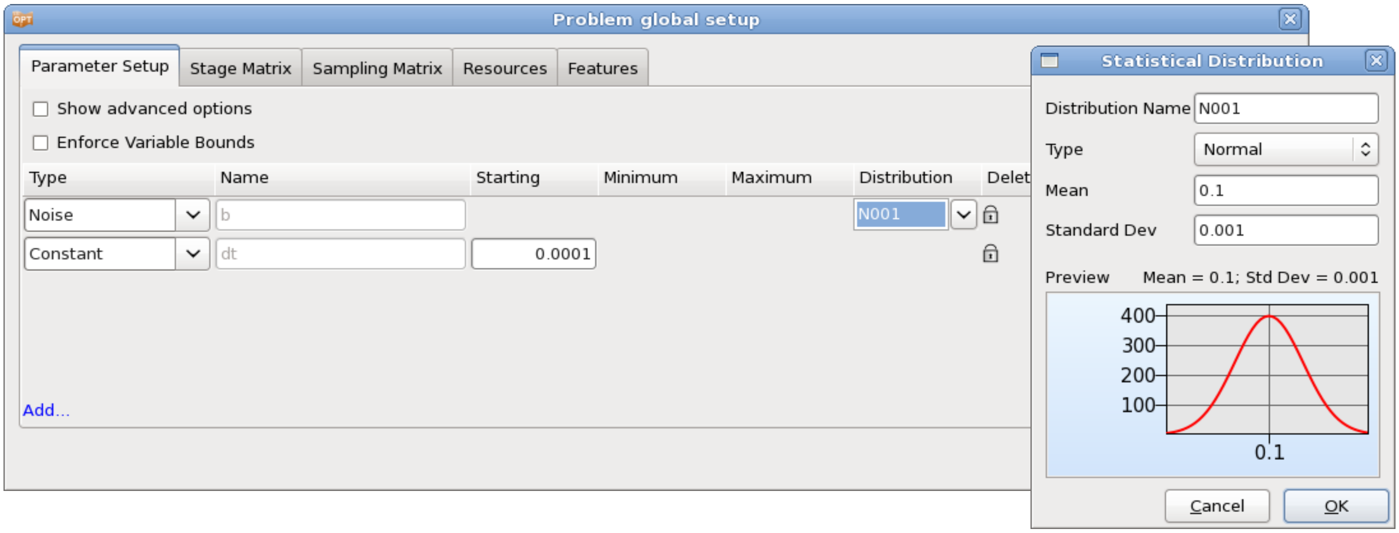


Figure 22-60: Histogram of the responses using only a stochastic field. Mean value and standard deviation are displayed as vertical lines.

22.6.2. A variable and a stochastic field

In this example a variable as well as the stochastic field are used to do the analysis. Figure 22-62 displays the resulting forces. The stochastic field is allowed to vary freely using the *PERTURBATION keyword like in Section 22.6.1, but the dimension *b* of the cross-section is also varied based on a Normal distribution (Figure 22-61).



Problem global setup

Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

□ Show advanced options

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delet
Noise	b				N001	
Constant	dt	0.0001				

Add...

Statistical Distribution

Distribution Name: N001

Type: Normal

Mean: 0.1

Standard Dev: 0.001

Preview Mean = 0.1; Std Dev = 0.001

Figure 22-61: Noise variable distribution definition.

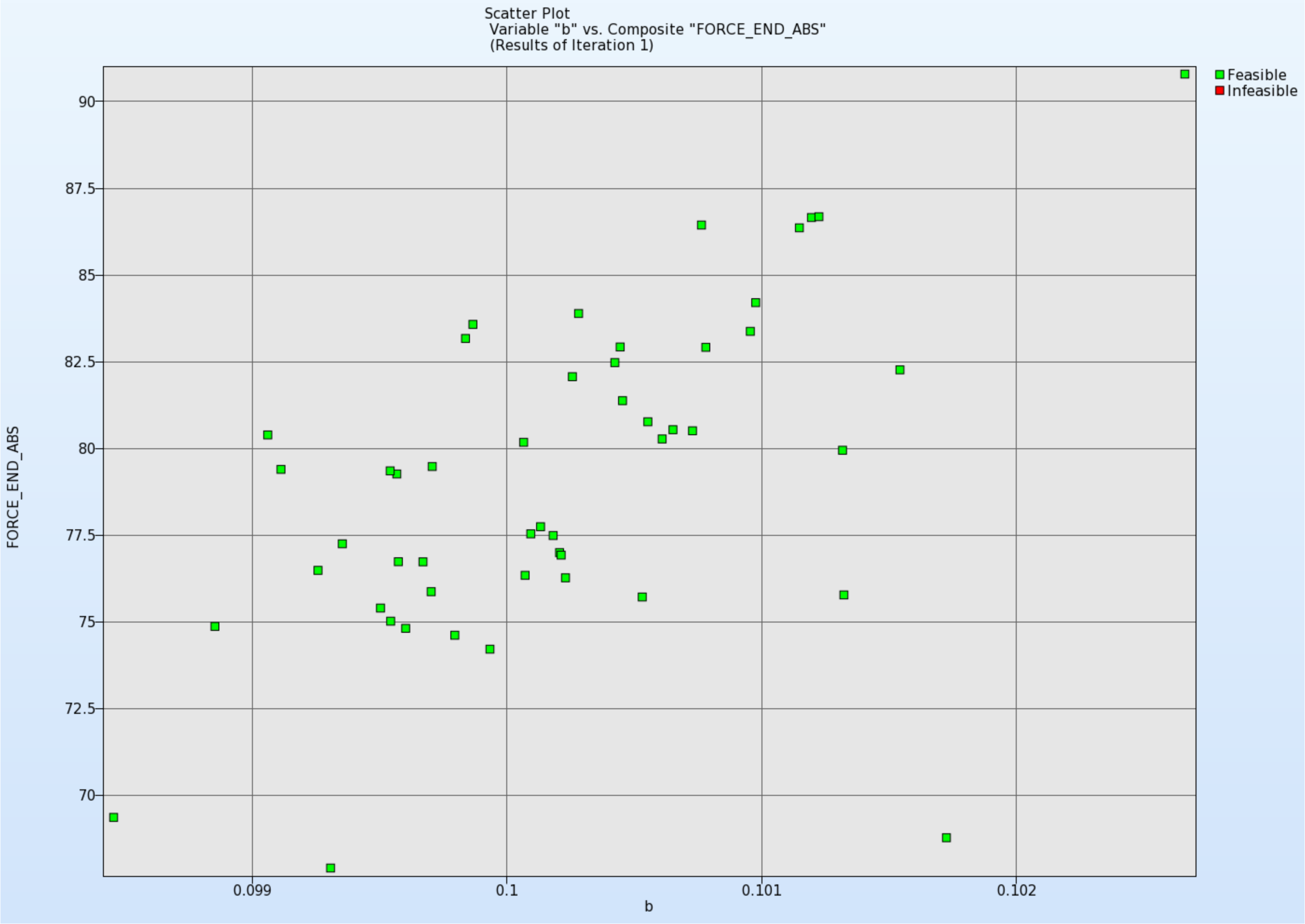


Figure 22-62: Plot of the responses using the thickness variable and a stochastic field.

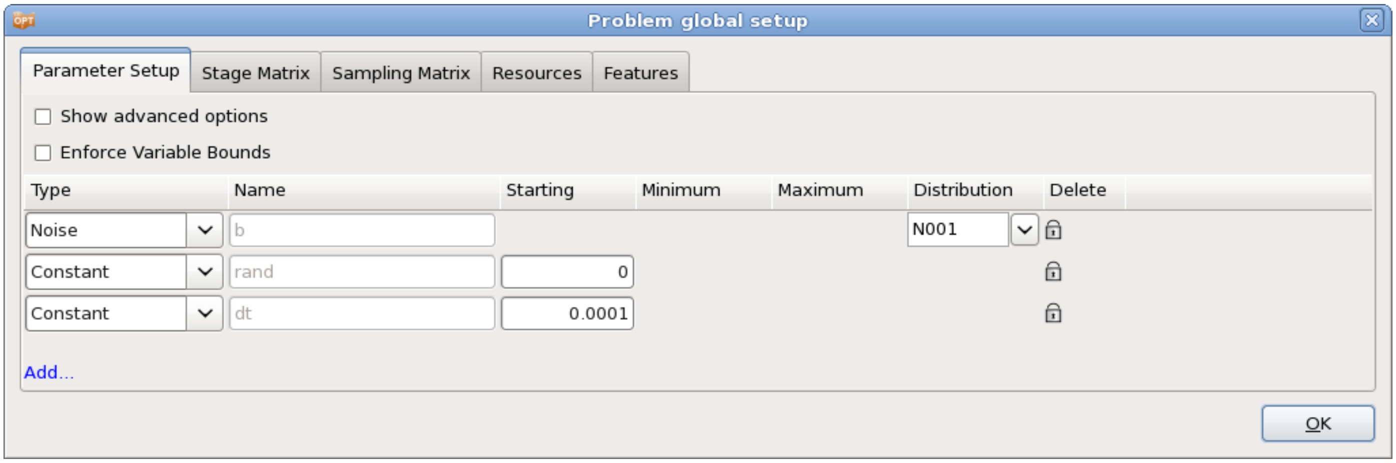
22.6.3. Replicate experiments using stochastic fields

In this example a variable as well as the stochastic field are used to do the analysis. Replicate runs are done at each experimental point with different values of the stochastic field.

In LS-DYNA, we add the random seed of the stochastic field as a variable, Figure 22-63. We let it vary freely by setting the seed to zero:

```
*PARAMETER
irand, 0
$
*PERTURBATION_NODE
$type, nid, scl, cmp, icoord, cid
4, , 1.e-2, 3,
$cstype, e1, e2, rnd
1, , , &rand
$
1, 1.e-1
$
```

In LS-OPT we use replicate experiments to analyze, Figure 22-64.

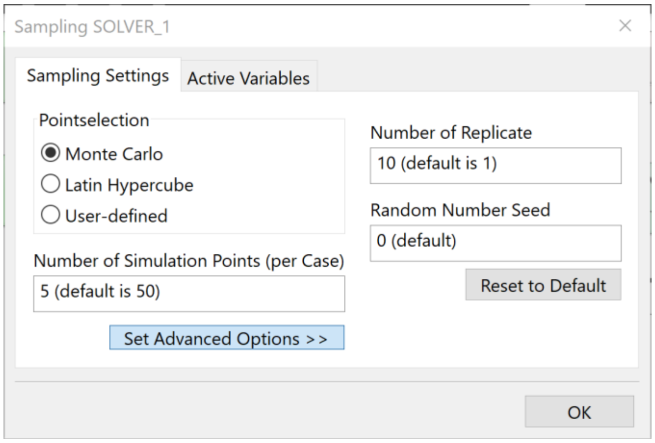


□ Show advanced options

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Distribution	Delete
Noise	b				N001	🔒
Constant	rand	0				🔒
Constant	dt	0.0001				🔒

Figure 22-63: Parameter Setup.



Sampling SOLVER_1

Sampling Settings | Active Variables

Pointselection

● Monte Carlo
○ Latin Hypercube
○ User-defined

Number of Simulation Points (per Case): 5 (default is 50)

Number of Replicate: 10 (default is 1)

Random Number Seed: 0 (default)

Figure 22-64: Sampling dialog. 10 replicate runs are done at each experimental point.

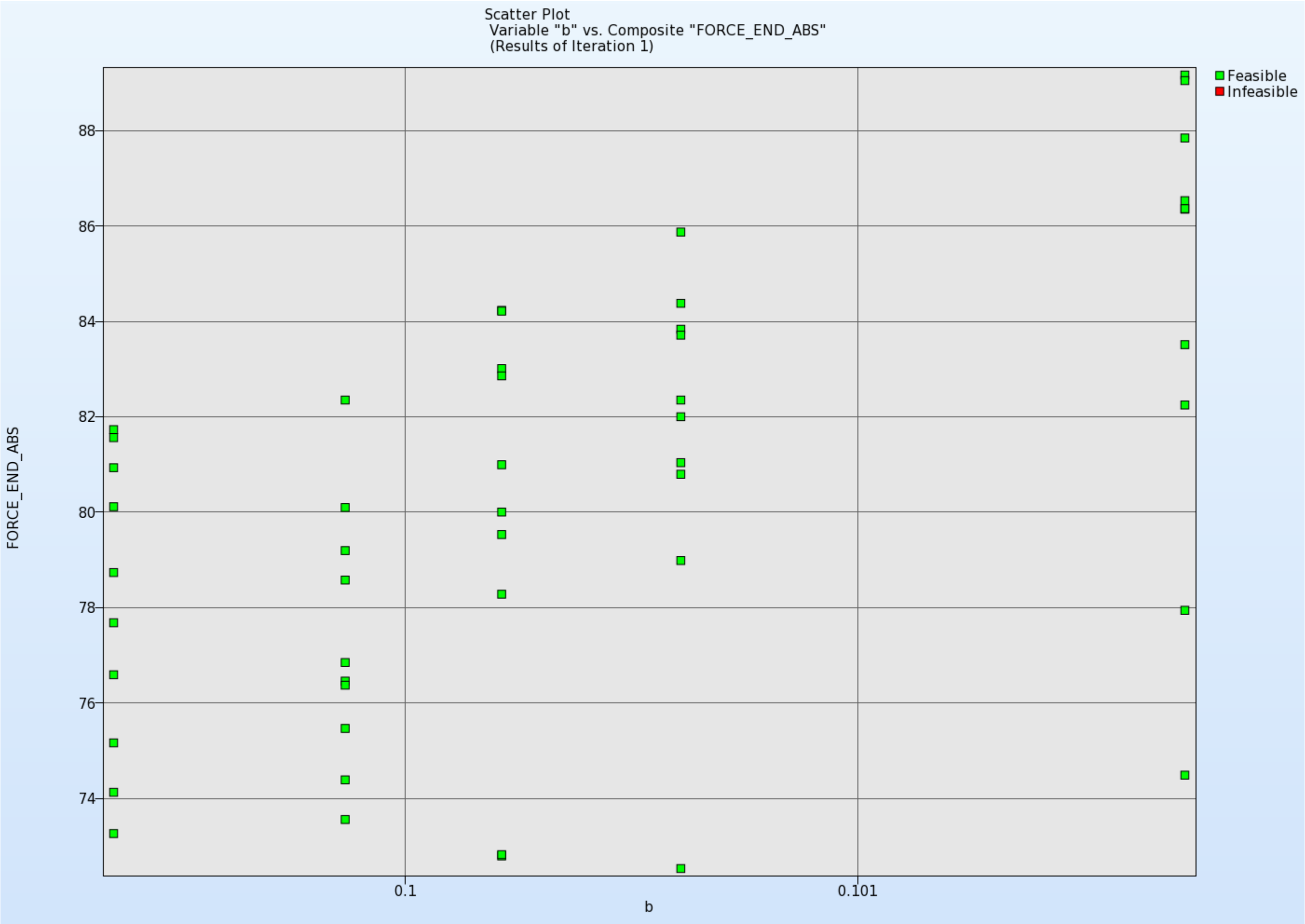
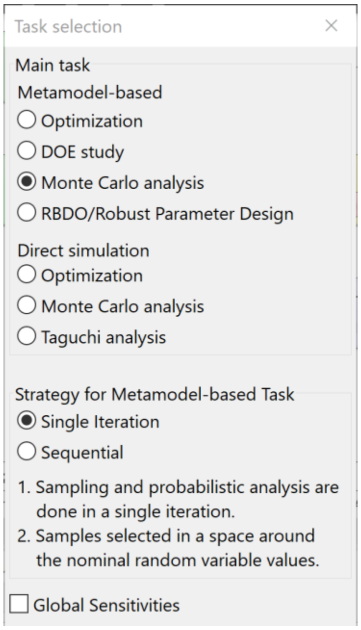


Figure 22-65: Plot of the responses using replicate experiments

22.6.4. Using fixed stochastic fields

In this example a variable as well as the stochastic field are used to do the analysis. Replicate runs are done at each experimental point with the same stochastic fields. By using the seed for the stochastic field as a variable, we are able to specify the stochastic field used, Figure 22-67. The random seed in the *PERTURBATION card can only take integer values. It should be noted that the seed is defined as a discrete control variable without a distribution that takes integer values only, and not a noise variable. Therefore, the task is metamodel-based Monte Carlo analysis (direct Monte Carlo is used to analyze the effect of uncertain variables).



Task selection

Main task

Metamodel-based

○ Optimization

○ DOE study

● Monte Carlo analysis

○ RBDO/Robust Parameter Design

Direct simulation

○ Optimization

○ Monte Carlo analysis

○ Taguchi analysis

Strategy for Metamodel-based Task

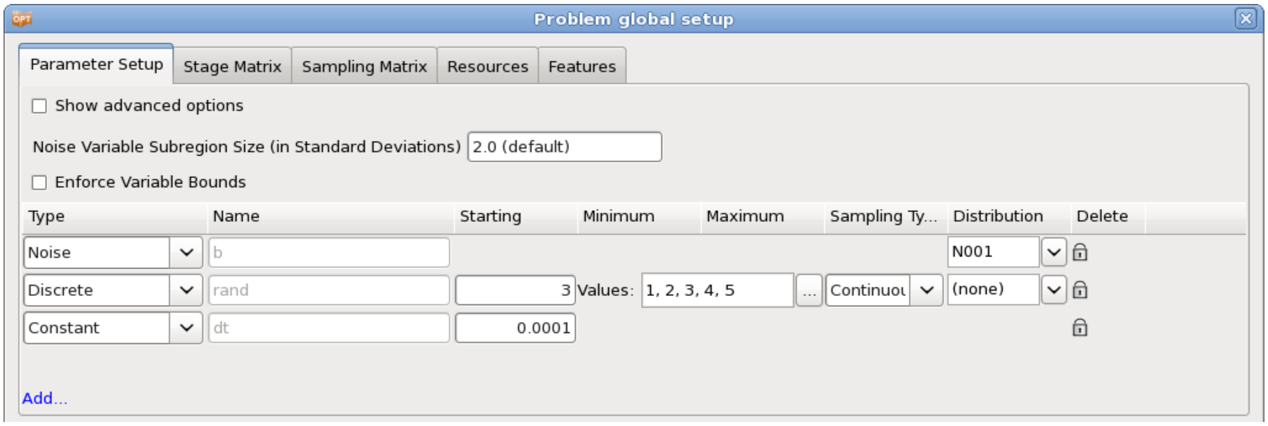
● Single Iteration

○ Sequential

1. Sampling and probabilistic analysis are done in a single iteration.
2. Samples selected in a space around the nominal random variable values.

□ Global Sensitivities

Figure 22-66: Task Metamodel-based Monte Carlo analysis



Problem global setup

Parameter Setup | Stage Matrix | Sampling Matrix | Resources | Features

□ Show advanced options

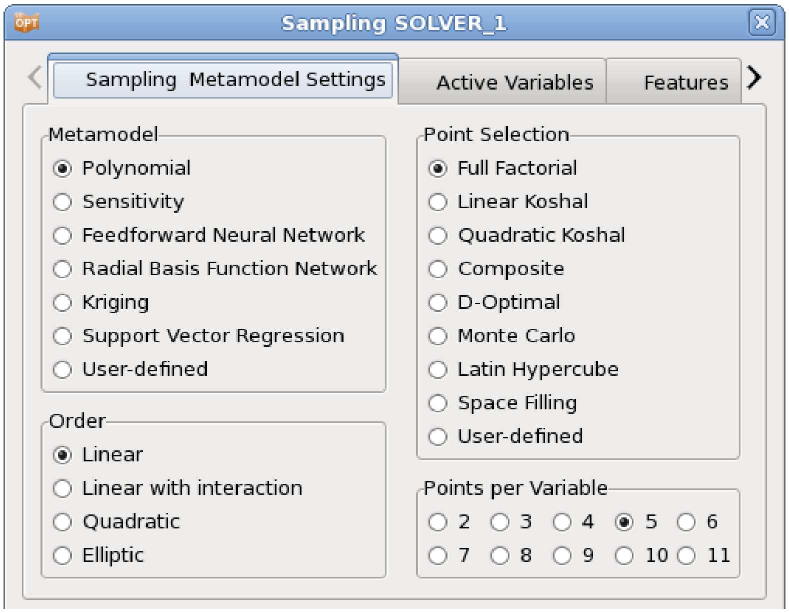
Noise Variable Subregion Size (in Standard Deviations) 2.0 (default)

□ Enforce Variable Bounds

Type	Name	Starting	Minimum	Maximum	Sampling Ty...	Distribution	Delete
Noise	b					N001	
Discrete	rand	3	Values:	1, 2, 3, 4, 5	Continuou	(none)	
Constant	dt	0.0001					

Add...

Figure 22-67: Parameter Setup.


Sampling SOLVER_1

Sampling Metamodel Settings | Active Variables | Features

Metamodel
● Polynomial
○ Sensitivity
○ Feedforward Neural Network
○ Radial Basis Function Network
○ Kriging
○ Support Vector Regression
○ User-defined

Order
● Linear
○ Linear with interaction
○ Quadratic
○ Elliptic

Point Selection
● Full Factorial
○ Linear Koshal
○ Quadratic Koshal
○ Composite
○ D-Optimal
○ Monte Carlo
○ Latin Hypercube
○ Space Filling
○ User-defined

Points per Variable
○ 2 ○ 3 ○ 4 ● 5 ○ 6
○ 7 ○ 8 ○ 9 ○ 10 ○ 11

Figure 22-68: Sampling dialog.

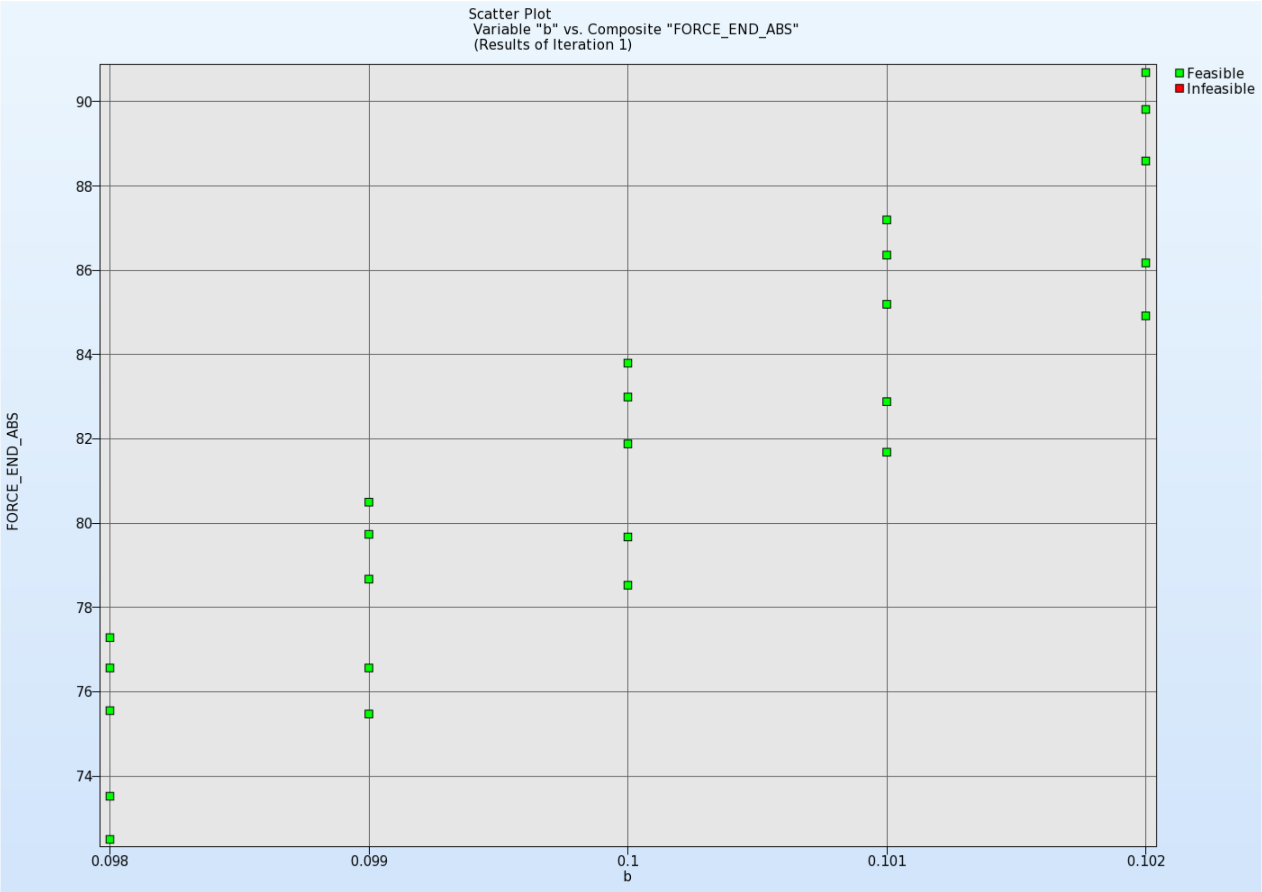


Figure 22-69: Plot of the responses using the same five stochastic fields in the replicates.

22.7. REFERENCES

[1] Basudhar, A, Stander, N, Gandikota, I, Svedin, Å, Witowski, K: “Design Tolerance Optimization using LS-OPT”, 13th LS-DYNA Forum, Bamberg, Germany, 2014

III − Theory

23. Response Surface Methodology

23.1. Introduction

An authoritative text on Response Surface Methodology (RSM) [1] defines the method as “a collection of statistical and mathematical techniques for developing, improving, and optimizing processes.” Although an established statistical method for several decades [2], it has only recently been actively applied to mechanical design [3]. Due to the importance of weight as a criterion and the multidisciplinary nature of aerospace design, the application of optimization and RSM to design had its early beginnings in the aerospace industry. A large body of pioneering work on RSM was conducted in this and other mechanical design areas during the eighties and nineties [3]-[6]. RSM can be categorized as a Metamodeling technique (see Chapter 24 for other Metamodeling techniques namely Neural Networks, and Radial Basis Functions available in LS-OPT).

Although inherently simple, the application of response surface methods to mechanical design has been inhibited by the high cost of simulation and the large number of analyses required for many design variables. In the quest for accuracy, increased hardware capacity has been consumed by greater modeling detail and therefore optimization methods have remained largely on the periphery of the area of mechanical design. In lieu of formal methods, designers have traditionally resorted to experience and intuition to improve designs. This is seldom effective and also manually intensive. Moreover, design objectives are often in conflict, making conventional methods difficult to apply, and therefore more analysts are formalizing their design approach by using optimization.

23.1.1. Approximating the response

Response Surface Methodology (or RSM) requires the analysis of a predetermined set of designs. A design surface is fitted to the response values using regression analysis. Least squares approximations are commonly used for this purpose. The response surfaces are then used to construct an approximate design “subproblem” which can be optimized.

The response surface method relies on the fact that the set of designs on which it is based is well chosen. Randomly chosen designs may cause an inaccurate surface to be constructed or even prevent the ability to construct a surface at all. Because simulations are often time-consuming and may take days to run, the overall efficiency of the design process relies heavily on the appropriate selection of a design set on which to base the approximations. For the purpose of determining the individual designs, the theory of experimental design (Design of Experiments or DOE) is required. Several experimental design criteria are available but one of the most popular for an arbitrarily shaped design space is the *D*-optimality criterion. This criterion has the flexibility of allowing any number of designs to be placed appropriately in a design space with an irregular boundary. The understanding of the *D*-optimality criterion requires the formulation of the least squares problem.

Consider a single response variable *y* dependent upon a number of variables ***x***. The exact functional relationship between these quantities is

$$y = \eta(\mathbf{x}) \tag{23-1}$$

The exact functional relationship is now approximated (e.g. polynomial approximation) as

$$\eta(\mathbf{x}) = f(\mathbf{x}) \tag{23-2}$$

The approximating function *f* is assumed to be a summation of basis functions:

$$f(\mathbf{x}) = \sum_{i=1}^{L} a_i \phi_i(\mathbf{x}) \tag{23-3}$$

where L is the number of basis functions ϕ_i used to approximate the model.

The constants $\mathbf{a} = [a_1, a_2, ..., a_L]^T$ have to be determined in order to minimize the sum of the square error:

$$\sum_{p=1}^{P} \left\{ \left[y(x_p) - f(x_p) \right]^2 \right\} = \sum_{p=1}^{P} \left\{ \left[y(x_p) - \sum_{i=1}^{L} a_i \phi_i(x_p) \right]^2 \right\} \tag{23-4}$$

P is the number of experimental points and *y* is the exact functional response at the experimental points $\boldsymbol{x}_i$.

The solution to the unknown coefficients is:

$$\mathbf{a} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y} \tag{23-5}$$

where $\boldsymbol{X}$ is the matrix

$$\mathbf{X} = \left[X_{ui}\right] = \left[\phi_i(x_u)\right] \tag{23-6}$$

The next critical step is to choose appropriate basis functions. A popular choice is the quadratic approximation

$$\phi = [1, x_1, ..., x_n, x_i^2, x_1 x_2, ..., x_1 x_n, ..., x_n^2]^T \tag{23-7}$$

but any suitable function can be chosen. LS-OPT allows linear, elliptical (linear and diagonal terms), interaction (linear and off-diagonal terms) and quadratic functions.

23.1.2. Factors governing the accuracy of the response surface

Several factors determine the accuracy of a response surface 23.6.

1. The size of the subregion

   For problems with smooth responses, the smaller the size of the subregion, the greater the accuracy. For the general problem, there is a minimum size at which there is no further gain in accuracy. Beyond this size, the variability in the response may become indistinguishable due to the presence of ‘noise’.

2. The choice of the approximating function

   Higher order functions are generally more accurate than lower order functions. Theoretically, over-fitting (the use of functions of too high complexity) may occur and result in suboptimal accuracy, but there is no evidence that this is significant for polynomials up to second order 23.6.

3. The number and distribution of the design points

   For smooth problems, the prediction accuracy of the response surface improves as the number of points is increased. However, this is only true up to roughly 50% oversampling 23.6 (very roughly).

23.1.3. Advantages of the method

Design exploration

As design is a process, often requiring feedback and design modifications, designers are mostly interested in suitable design formulae, rather than a specific design. If this can be achieved, and the proper design parameters have been used, the design remains flexible and changes can still be made at a late stage before verification of the final design. This also allows multidisciplinary design to proceed with a smaller risk of having to repeat simulations. As designers are moving towards computational prototyping, and as parallel computers or network computing are becoming more commonplace, the paradigm of design exploration is becoming more important. Response surface methods can thus be used for global exploration in a parallel computational setting. For instance, interactive trade-off studies can be conducted.

Global optimization

Response surfaces have a tendency to capture globally optimal regions because of their smoothness and global approximation properties. Local minima caused by noisy response are thus avoided.

23.1.4. Other types of response surfaces

Neural and Radial Basis Function networks, Kriging approximations and Support Vector Regression can also be used as response surfaces and are in Sections 24.1, 24.2 and 24.3.

23.2. Experimental design

Experimental design is the selection procedure for finding the points in the design space that must be analyzed. Many different types are available [1]. The factorial, Koshal, composite, *D*-optimal and Latin Hypercube designs are detailed here.

23.2.1. Factorial design

This is a l^n grid of designs and forms the basis of many other designs. l is the number of grid points in one dimension. It can be used as a basis set of experiments from which to choose a *D*-optimal design. In LS-OPT, the 3^n and 5^n designs are used by default as the basis experimental designs for first and second order *D*-optimal designs respectively.

Factorial designs may be expensive to use directly, especially for a large number of design variables.

23.2.2. Koshal design

This family of designs is saturated for modeling of any response surface of order *d*.

First order model

For *n* = 3, the coordinates are:

$$\begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

As a result, four coefficients can be estimated in the linear model

$$\phi = [1, x_1, ..., x_n]^T \tag{23-8}$$

Second order model

For *n* = 3, the coordinates are:

$$\begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

As a result, ten coefficients can be estimated in the quadratic model

$$\phi = [1, x_1, ..., x_n, x_i^2, x_1 x_2, ..., x_1 x_n, ..., x_n^2]^T \tag{23-9}$$

23.2.3. Central composite design

This design uses the 2^n factorial design, the center point, and the 'face center' points and therefore consists of P = $2^n + 2n + 1$ experimental design points. For $n = 3$, the coordinates are:

$$\begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \\ -\alpha & 0 & 0 \\ 0 & -\alpha & 0 \\ 0 & 0 & -\alpha \\ -1 & -1 & -1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

The points are used to fit a second-order function. The value of $\alpha = \sqrt[4]{2^n}$.

23.2.4. D-optimal design

This method uses a subset of all the possible design points as a basis to solve $\max\left|X^T X\right|$. The subset is usually selected from an l^n -factorial design where *l* is chosen a priori as the number of grid points in any particular dimension. Design regions of irregular shape, and any number of experimental points, can be considered [7]. The experiments are usually selected within a sub-region in the design space thought to contain the optimum. A genetic algorithm is used to solve the resulting discrete maximization problem. See References 23.6 and [5].

The numbers of required experimental designs for linear as well as quadratic approximations are summarized in the table below. The value for the D-optimality criterion is chosen to be 1.5 times the Koshal design value plus one. This seems to be a good compromise between prediction accuracy and computational cost [7]. The factorial design referred to below is based on a regular grid of 2^n points (linear) or 3^n points (quadratic).

Table 23-1: Number of experimental points required for experimental designs

Number of Variables *n*	**Linear approximation**			**Quadratic approximation**			**Central Composite**
	Koshal	***D*-optimal**	**Factorial**	**Koshal**	***D*-optimal**	**Factorial**	
1	2	4	2	3	5	3	3
2	3	5	4	6	10	9	9
3	4	7	8	10	16	27	15
4	5	8	16	15	23	81	25
5	6	10	32	21	32	243	43
6	7	11	64	28	43	729	77
7	8	13	128	36	55	2187	143
8	9	14	256	45	68	6561	273
9	10	16	512	55	83	19683	531
10	11	17	1024	66	100	59049	1045

23.2.5. Latin Hypercube Sampling (LHS)

The Latin Hypercube design is a constrained random experimental design in which, for *n* points, the range of each design variable is subdivided into *n* non-overlapping intervals on the basis of equal probability. One value from each interval is then selected at random with respect to the probability density in the interval. The *n* values of the first value are then paired randomly with the *n* values of variable 2. These *n* pairs are then combined randomly with the *n* values of variable 3 to form *n* triplets, and so on, until *k*-tuplets are formed.

Latin Hypercube designs are independent of the mathematical model of the approximation and allow estimation of the main effects of all factors in the design in an unbiased manner. On each level of every design variable only one point is placed. There are the same number of levels as points, and the levels are assigned randomly to points. This method ensures that every variable is represented, no matter if the response is dominated by only a few ones. Another advantage is that the number of points to be analyzed can be directly defined. Let *P* denote the number of points, and *n* the number of design variables, each of which is uniformly distributed between 0 and 1. Latin hypercube sampling (LHS) provides a *P*-by-*n* matrix $\mathbf{S} = S_{ij}$ that randomly samples the entire design space broken down into *P* equal-probability regions:

$$S_{ij} = \left(\eta_{ij} - \zeta_{ij}\right)/P, \qquad (23\text{-}10)$$

where $\eta_{1j},...,\eta_{Pj}$ are uniform random permutations of the integers 1 through *P* and ζ_{ij} independent random numbers uniformly distributed between 0 and 1. A common simplified version of LHS has centered points of *P* equal-probability sub-intervals:

$$S_{ij} = (\eta_{ij} - 0.5)/P \tag{23-11}$$

LHS can be thought of as a stratified Monte Carlo sampling. Latin hypercube samples look like random scatter in any bivariate plot, though they are quite regular in each univariate plot. Often, in order to generate an especially good space filling design, the Latin hypercube point selection ***S*** described above is taken as a starting experimental design and then the values in each column of matrix ***S*** is permuted so as to optimize some criterion. Several such criteria are described in the literature.

Maxi-min

One approach is to maximize the minimal distance between any two points (i.e. between any two rows of S). This optimization could be performed using, for example, Adaptive Simulated Annealing (see Section 26.11). The *maximin* strategy would ensure that no two points are too close to each other. For small P, *maximin* distance designs will generally lie on the exterior of the design space and fill in the interior as P becomes larger. See Section 23.2.6 for more detail.

Centered L2-discrepancy

Another strategy is to minimize the centered L2-discrepancy measure. The discrepancy is a quantitative measure of non-uniformity of the design points on an experimental domain. Intuitively, for a uniformly distributed set in the n-dimensional cube I^n = [0,1]^n, we would expect the same number of points to be in all subsets of I^n having the same volume. Discrepancy is defined by considering the number of points in the subsets of I^n. Centered L2 (CL2) takes into account not only the uniformity of the design points over the *n*-dimensional box region I^n, but also the uniformity of all the projections of points over lower-dimensional subspaces:

$$\begin{aligned} CL_2^2 = (13/12)^n - \frac{2}{P}\sum_{i=1\ldots P}\prod_{j=1\ldots n}\left(1+\frac{|S_{ij}-0.5|}{2}-\frac{|S_{ij}-0.5|^2}{2}\right) \\ +\frac{1}{P_2}\sum_{k=1\ldots P}\sum_{i=1\ldots P}\prod_{j=1\ldots n}\left(1+\frac{|S_{kj}-0.5|}{2}+\frac{|S_{ij}-0.5|}{2}-\frac{|S_{kj}-S_{ij}|}{2}\right). \end{aligned} \tag{23-12}$$

23.2.6. Space-filling designs

In the modeling of an unknown nonlinear relationship, when there is no persuasive parametric regression model available, and the constraints are uncertain, one might believe that a good experimental design is a set of points that are uniformly scattered on the experimental domain (design space). *Space-filling* designs impose no strong assumptions on the approximation model, and allow a large number of levels for each variable with a moderate number of experimental points. These designs are especially useful in conjunction with nonparametric models such as neural networks (feedforward networks, radial basis functions) and Kriging, [8], [9]. Space-filling points can also be submitted as the basis set for constructing an optimal (*D*-Optimal, etc.) design for a particular model (e.g. polynomial). Some space-filling designs are: random Latin Hypercube Sampling (LHS), Orthogonal Arrays, and Orthogonal Latin Hypercubes.

The key to space-filling experimental designs is in generating 'good' random points and achieving reasonably uniform coverage of sampled volume for a given (user-specified) number of points. In practice, however, we can only generate finite pseudo-random sequences, which, particularly in higher dimensions, can lead to a clustering of points, limiting their uniformity. To find a good space-filling design is a nonlinear programming hard problem, which – from a theoretical point of view – is difficult to solve exactly. This problem, however, has a representation, which might be within the reach of currently available tools. To reduce the search time and still generate good designs, the popular approach is to restrict the search within a subset of the general space-filling designs. This subset typically has some good 'built-in' properties with respect to the uniformity of a design.

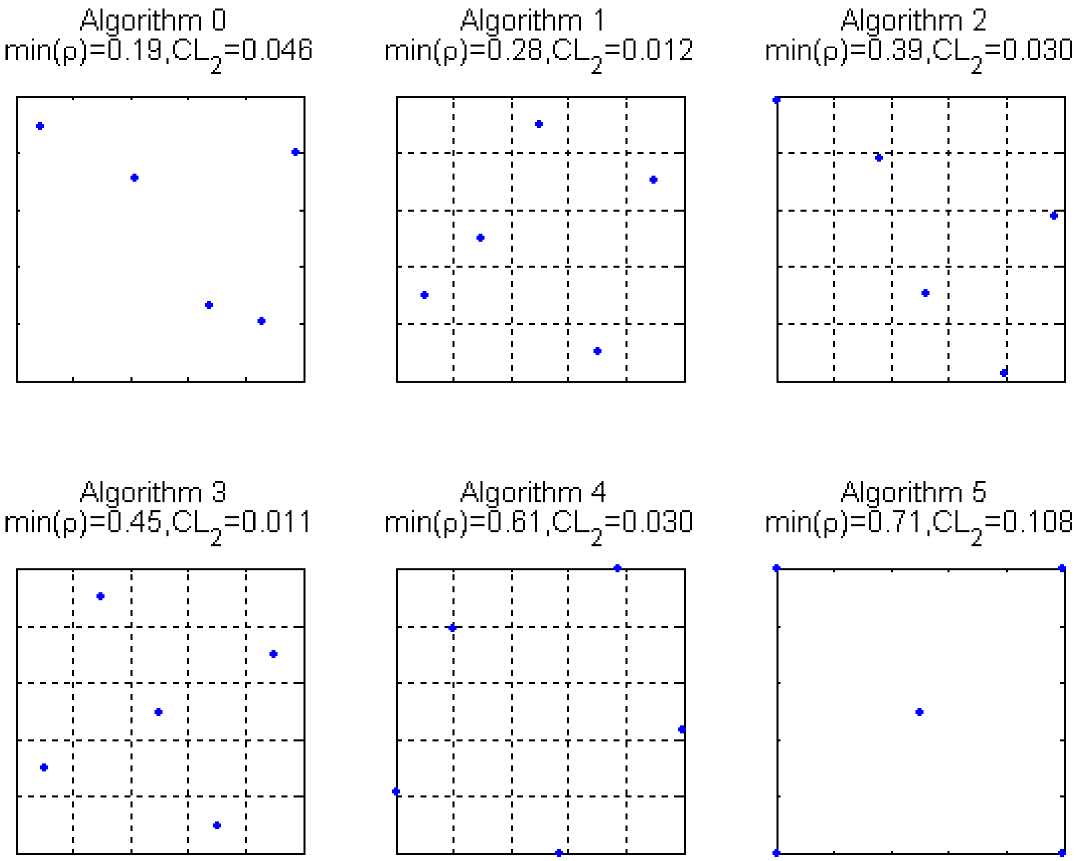
The constrained randomization method termed Latin Hypercube Sampling (LHS) and proposed in [10], has become a popular strategy to generate points on the 'box' (hypercube) design region. The method implies that on each level of every design variable only one point is placed, and the number of levels is the same as the number of runs. The levels are assigned to runs either randomly or so as to optimize some criterion, e.g. so that the minimal distance between any two design points is maximized ('maximin distance' criterion). Restricting the design in this way tends to produce better Latin hypercubes. However, the computational cost of obtaining these designs is high. In multidimensional problems, the search for an optimal Latin hypercube design using traditional deterministic methods (e.g. the optimization algorithm described in [11]) may be computationally prohibitive. This situation motivates the search for alternatives.

Probabilistic search techniques, *adaptive simulated annealing* and genetic algorithms are attractive heuristics for approximating the solution to a wide range of optimization problems. In particular, these techniques are frequently used to solve combinatorial optimization problems, such as the traveling salesman problem. Morris and Mitchell [12] adopted the simulated annealing algorithm to search for optimal Latin hypercube designs.

In LS-OPT, space-filling designs can be useful for constructing experimental designs for the following purposes:

1. The generation of basis points for the *D*-optimality criterion. This avoids the necessity to create a very large number of basis points using e.g. the full factorial design for large n. E.g. for n=20 and 3 points per variable, the number of points = $3^{20} \approx 3.5*10^9$.
2. The generation of design points for all approximation types, but especially for neural networks and Kriging.
3. The augmentation of an existing experimental design. This means that points can be added for each iteration while maintaining uniformity and equidistance with respect to pre-existing points.

LS-OPT contains 6 algorithms to generate space-filling designs (see Table 23-2).



Algorithm 0 $\min(\rho)=0.19, CL_2=0.046$

Algorithm 1 $\min(\rho)=0.28, CL_2=0.012$

Algorithm 2 $\min(\rho)=0.39, CL_2=0.030$

Algorithm 3 $\min(\rho)=0.45, CL_2=0.011$

Algorithm 4 $\min(\rho)=0.61, CL_2=0.030$

Algorithm 5 $\min(\rho)=0.71, CL_2=0.108$

Figure 23-1: Six space-filling designs: 5 points in a 2-dimensional box region

Table 23-2: Description of space-filling algorithms

Algorithm Number	**Description**
0	Random
1	'Central point' Latin Hypercube Sampling (LHS) design with random pairing
2	'Generalized' LHS design with random pairing
3	Given an LHS design, permutes the values in each column of the LHS matrix so as to optimize the maximin distance criterion taking into account a set of existing (fixed) design points. This is done using *simulated annealing*. Fixed points influence the maximin distance criterion, but are not allowed to be changed by Simulated Annealing moves.

4	Given an LHS design, moves the points within each LHS subinterval preserving the starting LHS structure, optimizing the maximin distance criterion and taking into consideration a set of fixed points.
5	Given an arbitrary design (and a set of fixed points), randomly moves the points so as to optimize the maximin distance criterion using simulated annealing (see 26.11).

Discussion

The Maximin distance space-filling algorithms 3, 4 and 5 minimize the energy function defined as the negative minimal distance between any two design points. Theoretically, any function that is a metric can be used to measure distances between points, although in practice the Euclidean metric is usually employed.

The three algorithms, 3, 4 and 5, differ in their selection of random Adaptive Simulated Annealing moves from one state to a neighboring state. For algorithm 3, the next design is always a 'central point' LHS design (Eq. 21.11). The algorithm swaps two elements of *I*, S_{ij} and S_{kj}, where *i* and *k* are random integers from 1 to *N*, and *j* is a random integer from 1 to *n*. Each step of algorithm 4 makes small random changes to a LHS design point preserving the initial LHS structure. Algorithm 5 transforms the design completely randomly - one point at a time. In more technical terms, algorithms 4 and 5 generate a candidate state, S', by modifying a randomly chosen element S_{ij} of the current design, *S*, according to:

$$S_{ij}^{'} = S_{ij} + \xi \tag{23-13}$$

where ξ is a random number sampled from a normal distribution with zero mean and standard deviation $\sigma_\xi \in [\sigma_{\min}, \sigma_{\max}]$. In algorithm 4 it is required that both $S_{ij}^{'}$ and S_{ij} in Eq. (21.13) belong to the same Latin hypercube subinterval.

Notice that maximin distance energy function does not need to be completely recalculated for every iterative step of simulated annealing. The perturbation in the design applies only to some of the rows and columns of *S*. After each step we can recompute only those nearest neighbor distances that are affected by the stepping procedures described above. This reduces the calculation and increased the speed of the algorithm.

To perform an annealing run for the algorithms 3, 4 and 5, the values for $T_{\max}$ and $T_{\min}$ can be adapted to the scale of the objective function according to:

$$T_{\max} := T_{\max} \times \Delta E \,, \tag{23-14}$$

where $\Delta E > 0$ is the average value of $|E'-E|$ observed in a short preliminary run of simulated annealing and $T_{\max}$ and $T_{\min}$ are positive parameters.

The basic parameters that control the adaptive simulated annealing in algorithms 3, 4 and 5 can be summarized as follows:

1. Energy function: negative minimal distance between any two points in the design.
2. Stepping scheme: depends on whether the LHS property is preserved or not.
3. Scalar parameters:

- Parameters for the cooling schedule:
- scaling factor for the initial (maximal) temperature, T_{max},
- scaling factor for the minimal temperature, T_{min},
- ratio of cost temperature and the parameter temperatures,
- number of iterations at each temperature, ν_T.
- parameter temperature update interval
- Parameters that control the standard deviation of ξ in (2.13):
- upper bound, σ_{max},
- lower bound, σ_{min}.
- Termination criterion: maximal number of energy function evaluations, N_{it}.

23.2.7. Orthogonal Arrays

An Orthogonal Array is a type of Fractional Factorial design that only considers a selected subset of the corresponding Full Factorial samples while still giving complete information about the requested variable effects.

An Orthogonal Array is an $(n \times m)$ matrix of experiment, represented by the notation $L_n(s^m)$ where *n* is the number of experiment, *m* the number of factors/variables and *s* the number of levels/values. In an Orthogonal Array, for any pair of columns, all combination of factor levels occur an equal number of time. This is called the balancing property and it implies orthogonality [14]. Table 23-3 provides an Orthogonal Array example consisting of 7 variables with 2 levels each.

Table 23-3: $L_8(2^7)$ ***Orthogonal Array***

Experiment number	**Column and factor assignment**						
	1st 2-level factor	**2nd 2-level factor**	**3rd 2-level factor**	**4th 2-level factor**	**5th 2-level factor**	**6th 2-level factor**	**7th 2-level factor**
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1

7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

Generating the proper Orthogonal Array suitable for the problem of interest is the main difficulty of Taguchi method, as a different algorithm may be required depending on the numer of samples, variables and levels. This is the reason why generally, the method uses a pre-identified list of Orthogonal Arrays. SAS [16] provides a catalog of over 117,000 Orthogonal Arrays regrouping every array listed in Orthogonal Array by Hedayat, Sloane and Stufken (1999) [17] and other arrays discovered afterwards. Table 23-4 is an example of the available Orthogonal Arrays for simple problems.

Table 23-4: Orthogonal Arrays for simple problems [18]

Orthogonal Array	**Number of rows/experiments**	**Max. number of parameters**	**Max. number of columns for the respective levels**			
			2-levels	**3-levels**	**4-levels**	**5-levels**
$L_4(2^3)$	4	3	3	-	-	-
$L_8(2^7)$	8	7	7	-	-	-
$L_9(3^4)$	9	4	-	4	-	-
$L_{12}(2^{11})$	12	11	11	-	-	-
$L_{16}(2^{15})$	16	15	15	-	-	-
$L_{16}(4^5)$	16	5	-	-	5	-
$L_{18}(2^1 3^7)$	18	8	1	7	-	-
$L_{25}(5^6)$	25	6	-	-	-	6
$L_{27}(3^{13})$	27	13	-	13	-	-

The basic Taguchi method using Orthogonal Array only considers the main effect of independent variables. When variable interactions occur, the ability to obtain a good estimation of interaction and main effects is often lost. In that case, more experiments are needed to estimate all effects. This is the reason why it is highly recommended to use Orthogonal Array in case of non-correlated variables. Another solution is to use interaction tables. Currently 2-level factors interaction table are available for $L_8(2^7)$ and $L_{16}(2^{15})$.

An interaction table is a square and symmetric matrix where the dimension is the number of factors.

Table 23-5: Interaction table for $L_8(2^7)$ Orthogonal Array

Column	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	(1)	3	2	5	4	7	6
2		(2)	1	6	7	4	5
3			(3)	7	6	5	4
4				(4)	1	2	3
5					(5)	3	2
6						(6)	1
7							(7)

This matrix indicates which column must be left empty in the Orthogonal Array to allow the proper analysis of main and interaction effect. For example, if we want to add an interaction between the factor in column 2 and the factor in column 4 using $L_8(2^7)$, the column 6 must be left empty (6 is the value of row 2, column 4 in the interaction table).

To select the appropriate Orthogonal Array for the factors and levels of interest, the row number of the matrix of experiments is determined by the matrix degree of freedom. A matrix of experiments with *n* rows has *n* degrees of freedom. The degree of freedom of an orthogonal array is the sum of the overall mean degree of freedom (equal to 1), the degree of freedom of every variable and the degree of freedom of any variable interaction:

$$dof = dof(OverallMean) + \sum dof(Var_i) + \sum dof(Interaction_i) \qquad (23\text{-}15)$$

The degree of freedom of a variable is equal to the number of levels minus one.

For example, in case of a problem with 4 independent factors with 3 levels each, there are 9 degrees of freedom.

$$dof = 1 + \sum_{j=1}^{4}(3-1) = 9.$$

The degree of freedom of an interaction is equal to the multiplication of both interacting factors degree of freedom:

$$dof(Interaction_{k=(i,j)}) = dof(Var_i) \times dof(Var_j)$$

Then the degree of freedom of a problem is:

$$dof = 1 + \sum_{Var\ Number} (Number(Var_i) - 1) + \sum_{Interaction\ Number} (levelNumber(Var_i) - 1) \times (levelNumber(Var_j) - 1)$$

(23-16)

As we use a predefined list of Orthogonal Arrays, it is not always possible to find the perfect Orthogonal Array matching the experimental design of interest. When no perfect match can be found, some basic techniques can be used to select a compatible array while keeping the important Orthogonality property:

- Leave empty columns in a table. A sub-matrix formed by deleting some columns of an orthogonal array is also an orthogonal array [15].
- The Dummy Level Technique which assign a factor with *m* levels to a column that has *n* levels with $m < n$. The resulting array is still proportionally balanced and hence, orthogonal [14].

Results Analysis

When using an Orthogonal Array sampling, it is possible to observe the parameters effect on the result by looking at the level means or the S/N ratio η. Main and interaction effect plots are available in Statistical tools plot, Section 16.2.6.

For one level of one variable, the level mean is the mean overall the results obtained with this variable value.

For the S/N ratio, LS-OPT use two different formulations:

- Smaller-the-better S/N ratio: This type is used in case of a minimization problem. $\eta_i = -10log_{10}({y_i}^2)$
- Larger-the-better S/N ratio: This type is used in case of a maximization problem. $\eta_i = -10\ log_{10}(1/{y_i}^2)$

The S/N ratio of one variable value (or one interaction value) is then the mean over the individual S/N ratio of each experiment using this variable value (or interaction value).

Where y_i represent the transformed results of the observed experiment i. As the smaller-the-better and larger-the-better S/N ratios are only valid for positive values, they are calculated based on transformed responses if the minimum response among all the experiments in non-positive. The transformed response y_i is:

$$y_i = res_i + \left| \min_{j=1,n} res_j \right| + 1$$

where res_i is the actual response. Then the main effect of each parameter and each interaction can be observed using three options (level means, Smaller-the-better S/N ratio, Larger-the-better S/N ratio) available in the Statistics plot. It should be noted that the responses are not transformed if the minimum actual response is positive.

When a Taguchi task is performed, a more detailed analysis is given in the output and report file by the ANOVA table, Appendix E.3.4 .

23.2.8. Random number generator

The Mersenne Twister [13] is used in Neural Network construction and Monte Carlo, Latin Hypercube, Space Filling and *D*-Optimal point selection and probabilistic analysis. The Mersenne Twister (MT19937) is a pseudo-random number generator developed by Matsumoto and Nishimura and has the merit that it has a far longer period and far higher order of equi-distribution than any other implemented generators. It has been proved that the period is 2^{19937}-1, and a 623-dimensional equi-distribution property is assured. Features have been provided to seed the generator to enable sensitivity studies.

23.2.9. Reasonable experimental designs

A 'reasonable' design space refers to a region of interest which, in addition to having specified bounds on the variables, is also bounded by specified values of the responses. This results in an irregular shape of the design space.

In LS-OPT, constrained experimental designs can be obtained for the D-Optimality criterion as well as for Space Filling.

Reasonable experimental designs can only be obtained using explicit constraints, i.e. constraints which are not defined by a metamodel.

23.3. Model adequacy checking

As indicated in the previous sections, response surfaces are useful for interactive trade-off studies. For the trade-off surface to be useful, its capability of predicting accurate response must be known. Error analysis is therefore an important aspect of using response surfaces in design. Inaccuracy is not always caused by random error (noise) only, but modeling error (sometimes called bias error), especially in a large subregion or where there is strong non-linearity present, could play a very significant role. There are several error measures available to determine the accuracy of a response surface.

23.3.1. Residual sum of squares

For the predicted response $\hat{y}_i$ and the actual response y_i, this error is expressed as

$$\varepsilon^2 = \sum_{i=1}^{P} \left(y_i - \hat{y}_i \right)^2 \qquad (23\text{-}17)$$

If applied only to the regression points, this error measure is not very meaningful unless the design space is oversampled e.g., ε = 0 if the number of points P equals the number of basis functions L in the approximation.

23.3.2. RMS error

The residual sum-of-squares is sometimes used in its square root form, ε_{RMS}, and called the "RMS error":

$$\varepsilon_{RMS} = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(y_i - \hat{y}_i\right)^2} \tag{23-18}$$

23.3.3. Maximum residual

This is the maximum residual considered over all the design points and is given by

$$\varepsilon_{\max} = \max\left|y_i - \hat{y}_i\right|. \tag{23-19}$$

23.3.4. Prediction error

The same as the RMS error, but using only responses at preselected prediction points independent of the regression points. This error measure is an objective measure of the prediction accuracy of the response surface since it is independent of the number of construction points. It is important to know that the choice of a larger number of construction points will, for smooth problems, diminish the prediction error.

The prediction points can be determined by adding rows to ***X***

$$\mathbf{X}_a(x_p) = \begin{bmatrix} \mathbf{X} \\ \mathbf{A}(\mathbf{x}_p) \end{bmatrix} \tag{23-20}$$

and solving

$$\max\left|\mathbf{X}_a^T\mathbf{X}_a\right| = \max\left|\mathbf{X}^T\mathbf{X} + \mathbf{A}^T\mathbf{A}\right| \tag{23-21}$$

for $\boldsymbol{x}_p$.

23.3.5. PRESS residuals

The prediction sum of squares residual (PRESS) uses each possible subset of P – 1 points as a regression data set to fit the regression model, and the remaining point in turn is used to form a prediction set [1]. PRESS is defined as the sum of all prediction errors, but can be computed from a single regression analysis of all P points.

$$PRESS = \sum_{i=1}^{P}\left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2, \tag{23-22}$$

where h_{ii} are the diagonal terms of

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T. \tag{23-23}$$

H is the “hat” matrix, the matrix that maps the observed responses to the fitted responses, i.e.

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}. \tag{23-24}$$

The PRESS residual can also be written in its square root form

$$SPRESS = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(\frac{y_i - \hat{y}_i}{1-h_{ii}}\right)^2}. \tag{23-25}$$

For a saturated design, ***H*** equals the unit matrix ***I*** so that the PRESS indicator becomes undefined.

23.3.6. The coefficient of multiple determination R^2

The coefficient of determination R^2 is defined as:

$$R^2 = \frac{\sum_{i=1}^{P}(\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^{P}(y_i - \bar{y}_i)^2} \tag{23-26}$$

where P is the number of design points and $\bar{y}$, $\hat{y}_i$ and y_i represent the mean of the responses, the predicted response, and the actual response, respectively. This indicator, which varies between 0 and 1, represents the ability of the response surface to identify the variability of the design response. A low value of R^2 usually means that the region of interest is either too large or too small and that the gradients are not trustworthy. The value of 1.0 for R^2 indicates a perfect fit. However, the value will not warn against an overfitted model with poor prediction capabilities.

23.3.7. R^2 for Prediction

For the purpose of *prediction* accuracy the $R^2_{prediction}$ indicator has been devised [1].

$$R^2_{prediction} = 1 - \frac{PRESS}{S_{yy}}, \tag{23-27}$$

where

$$S_{yy} = \mathbf{y}^T\mathbf{y} - \frac{1}{P}\left(\sum_{i=1}^{P} y_i\right)^2. \tag{23-28}$$

$R^2_{prediction}$ represents the ability of the model to detect the variability in predicting new responses 23.6.

23.3.8. Iterative design and prediction accuracy

In an iterative scheme with a shrinking region the R^2 value tends to be small at the beginning, then approaches unity as the region of interest shrinks, thereby indicating improvement of the modeling ability. It may then reduce again as the noise starts to dominate in a small region causing the variability to become indistinguishable. In the same progression, the prediction error will diminish as the modeling error fades, but will stabilize at above zero as the modeling error is replaced by the random error (noise).

23.4. ANOVA

Since the number of regression coefficients determines the number of simulation runs, it is important to remove those coefficients or variables which have small contributions to the design model. This can be done by doing a preliminary study involving a design of experiments and regression analysis. The statistical results are used in an analysis of variance (ANOVA) to rank the variables for screening purposes. The procedure requires a single iteration using polynomial regression, but results are produced after every iteration of a normal optimization procedure.

ANOVA is a regression based sensitivity measure with

$$b_j = \frac{\partial f}{\partial x_j} \cdot \Delta x_j, j = 1, \ldots, N,$$

where f is the linear approximation, Δx_j the size of the design space of variable x_j, and N the number of variables, Figure 23-2.

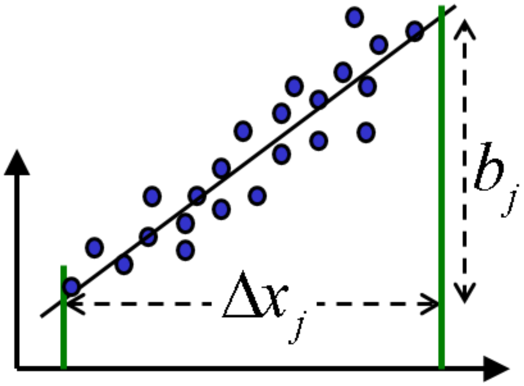


Figure 23-2: Definition of ANOVA value b_j

23.4.1. The confidence interval of the regression coefficients

The 100(1 – α)% confidence interval for the least squares estimators $b_j \wedge j = 0,1,...,L$ is determined by the inequality

$$b_j - 0.5\Delta b_j \le \beta_j \le b_j + 0.5\Delta b_j, \qquad (23\text{-}29)$$

where

$$\Delta b_j(\alpha) = 2t_{\alpha/2,P-L}\sqrt{\hat{\sigma}^2 C_{jj}}, \qquad (23\text{-}30)$$

$\beta_j$is the regression coefficient and $\hat{\sigma}^2$ is an unbiased estimator of the variance σ^2 given by

$$\hat{\sigma}^2 = \frac{\varepsilon^2}{P-L} = \frac{\sum_{i=1}^{P}(y_i - \hat{y}_i)^2}{P-L} \tag{23-31}$$

P is the number of points and *L* is the number of terms in the polynomial while C_{jj} is the diagonal element of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ corresponding to b_j and $t_{\alpha/2,P-L}$ is Student's *t*-Distribution. 100(1 – α)% therefore represents the level of confidence that b_j will be in the computed interval.

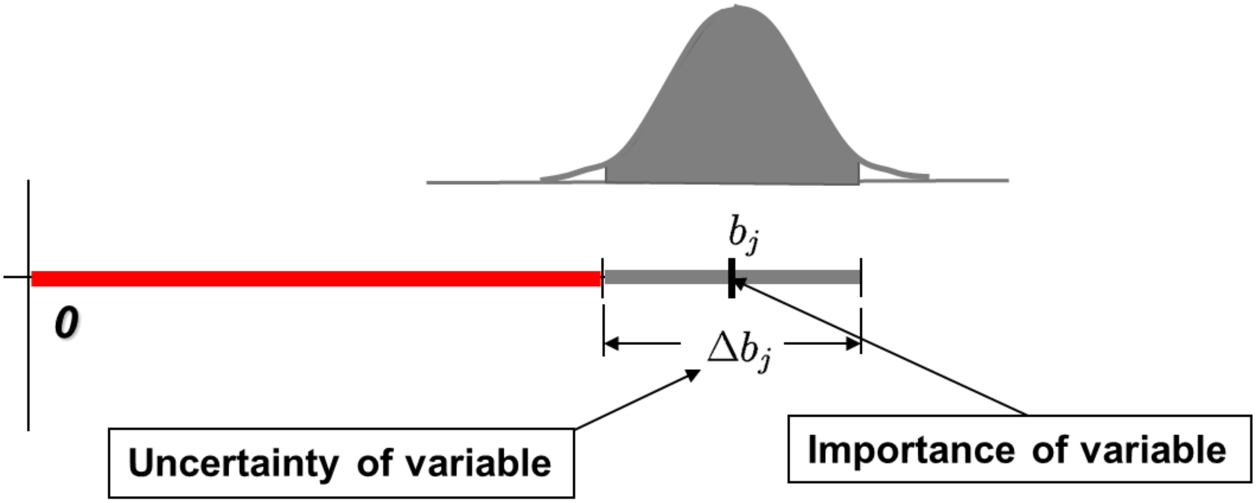


Figure 23-3: ANOVA value with 100(1 – α)% confidence interval

23.4.2. The significance of a regression coefficient b_j

The contribution of a single regressor variable to the model can also be investigated. This is done by means of the *partial F*-test where *F* is calculated to be

$$F = \frac{\left[\varepsilon^2_{reduced} - \varepsilon^2_{complete}\right]/r}{\varepsilon^2_{complete}/(P-L)} \tag{23-32}$$

where $r = 1$ and the reduced model is the one in which the regressor variable in question has been removed. Each of the ε^2 terms represents the sum of squared residuals for the reduced and complete models respectively.

It turns out that the computation can be done without analyzing a reduced model by computing

$$F = \frac{b_j^2/C_{jj}}{\varepsilon^2_{complete}/(P-L)}. \tag{23-33}$$

F can be compared with the *F*-statistic $F_{\alpha,1,P-L}$ so that if $F > F_{\alpha,1,P-L}$, β_j is non-zero with (100 – α)% confidence. The confidence level α that β_j is not zero can also be determined by computing the α for $F = F_{\alpha,1,P-L}$. The importance of β_j is therefore estimated by both the magnitude of b_j as well as the level of confidence in a non-zero β_j.

The significance of regressor variables may be represented by a bar chart of the magnitudes of the coefficients b_j with an error bar of length $2\Delta b_j(\alpha)$ for each coefficient representing the confidence interval for a given level of confidence α. The relative bar lengths allow the analyst to estimate the importance of the variables and terms to be included in the model while the error bars represent the contribution to noise or poorness of fit by the variable.

All terms have been normalized to the size of the design space so that the choice of units becomes irrelevant and a reasonable comparison can be made for variables of different kinds, e.g. sizing and shape variables or different material constants.

23.5. Principal Component Analysis

Principal component analysis (PCA) is a multivariate statistical technique using an orthogonal transformation to convert a set of possibly correlated observations into a set of uncorrelated values called principal components. The principal components are sorted in decreasing order according to their variance, so that the first principal component accounts for the largest variability in the data set [20].

Then, to analyze the effect of parameters on the data set, a method similar to ANOVA is applied to the PCA results [19].

The observations are represented by a matrix $X \in \mathbb{R}^{N \times T}$ where N is the number of experiments and T is the number of observations (number of discrete time values for histories, or number of points for multi-responses).

The observation matrix is obtained using an approximation. For histories, the existing approximation is used whereas for multi-responses a RBF approximation is used to get a full factorial representation of the result.

The principal components are computed using the eigenvalue decomposition of the covariance matrix [21], [22]

$$C = {X_C}^T X_C = V \Lambda V^T.$$

Where X_C is the observation matrix centered and normalized, C is the correlation matrix, V_k are the eigenvectors, and λ_k the eigenvalues (in decreasing order) of the correlation matrix.

The Principal Components are computed as follow:

$$H_k = X_C V_k.$$

And $\|H_k\|^2 = \lambda_k$ is the inertia associated with the k Principal Component.

The sensitivity analysis of the parameters is then performed by computing the variable sensitivity indices associated to the principal components:

$$SS_{W,k} = \|S_W H_k\|^2$$

$$SI_{W,k} = SS_{W,k}/\lambda_k \in [0,1].$$

Where S_W is the orthogonal projection matrix of W (the variable values vector), $SS_{W,k}$ is the sum of squares associated to the variable for the kth principal component, and $SI_{W,k}$ is the sensitivity index of the variable to the kth principal component.

23.6. REFERENCES

[1] Myers, R.H., Montgomery, D.C. Response Surface Methodology. Process and Product Optimization using Designed Experiments. Wiley, 1995.

[2] Box, G.E.P., Draper, N.R. A basis for the selection of a response surface design. Journal of the American Statistical Association, 54, pp. 622-654, 1959.

[3] Toropov, V.V. Simulation approach to structural optimization. Structural Optimization, 1, pp. 37-46, 1989.

[4] Schoofs, A.J.G. Experimental Design and Structural Optimization. PhD thesis, Technische Universiteit Eindhoven, August 1987.

[5] Tu, J. and Choi, K.K. Design potential concept for reliability-based design optimization. Technical Report R99-07. Center for Computer Aided Design and Department of Mechanical Engineering. College of engineering. University of Iowa. December 1999.

[6] Jin, R., Chen, W. and Simpson, T.W. Comparative studies of metamodeling techniques under multiple modeling criteria. AIAA Paper, AIAA-2000-4801.

[7] Roux, W.J. Structural Optimization using Response Surface Approximations. PhD thesis, University of Pretoria, April 1997.

[8] Wilson, B., Cappelleri, D.J., Frecker, M.I. and Simpson, T.W. Efficient Pareto frontier exploration using surrogate approximations. Optimization and Engineering, 2 (1), pp.31-50, 2001.

[9] Ye, K., Li, W., Sudjianto, A., Algorithmic construction of optimal symmetric Latin hypercube designs. Journal of Statistical Planning and Inferences, 90, pp. 145-159, 2000.

[10] McKay, M.D., Conover, W.J., Beckman, R.J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, pp. 239-245, 1979.

[11] Park, J.-S. Optimal Latin-hypercube designs for computer experiments. Journal of Statistical Planning Inference, 39, pp. 95-111, 1994.

[12] Morris, M., Mitchell, T. Exploratory design for computer experiments. Journal of Statistical Planning Inference, 43, pp. 381-402, 1995.

[13] Matsumoto, M. and Nishimura, T., Mersenne Twister: A 623-Dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation, 8(1), pp. 3-30, 1998

[14] Phadke, M. S. Quality Engineering using Robust Design, 1989. Englewood Cliffs, New Jersey: PTR Prentice-Hall Inc.

[15] Kacker, Raghu N., Eric S. Lagergren, and James J. Filliben. Taguchi’s orthogonal arrays are classical designs of experiments. Journal of research of the National Institute of Standards and Technology, 1991, vol. 96, no 5, p. 577.

[16] Kuhfeld, W. F. Orthogonal Arrays. TS-723 (http://support.sas.com/techsup/technote/ts723.html), 2006.

[17] Hedayat, A. Samad, Neil James Alexander Sloane, and John Stufken. Orthogonal arrays: theory and applications. Springer Science & Business Media, 2012.

[18] Kemmler, Stefan, et al. Comparison of Taguchi Method and Robust Design Optimization (RDO): by application of a functional adaptive simulation model for the robust product-optimization of an adjuster unit. 2015.

[19] Lamboni, M., Makowski, D., & Monod, H. Multivariate global sensitivity analysis for discrete-time models. PhD dissestation, auto-saisine, 2008. 2015.

[20] Abdi, H., & Williams, L.J. Principal component analysis. Wiley interdisciplinary reviews: computational statistics, 2(4), 433-459. 2010.

[21] Missoum, S. Probabilistic optimal design in the presence of random fields. Structural and Multidisciplinary Optimization, 35(6), 523-530. 2008.

[22] Turk, M., & Pentland, A. Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), 71-86. 1991.

24. Metamodeling Techniques

Metamodeling techniques allow the construction of surrogate design models for the purpose of design exploration such as variable screening, optimization and reliability. LS-OPT provides the capability of using five standard types of metamodeling techniques, namely polynomial response surfaces (already discussed, see Section 23.1), Neural Networks (NN) (Section 24.1.2), Radial Basis Function Networks (RBF) (Section 24.1.3), Kriging (Section 24.2) and Support Vector Regression (SVR) (Section 24.3). All of these approaches can be useful to provide a predictive capability for optimization or reliability. In addition, linear polynomials, although perhaps less accurate, are highly suitable for variable screening (Section 23.4). At the core, these techniques differ in the regression methods employed to construct the surrogate models. The polynomial response surface method and the RBF's use linear regression, while neural networks use nonlinear regression methods requiring an optimization algorithm.

When using polynomials, the user is faced with the choice of deciding which monomial terms to include. In addition, polynomials, by way of their nature as Taylor series approximations, are not natural for the creation of updateable surfaces. This means that if an existing set of point data is augmented by a number of new points which have been selected in a local subregion (e.g. in the vicinity of a predicted optimum), better information could be gained from a more flexible type of approximation that will keep global validity while allowing refinement in a subregion of the parameter space. Such an approximation would provide a more natural approach for combining the results of successive iterations.

24.1. Neural networks

Neural methods are natural extensions and generalizations of regression methods. Neural networks have been known since the 1940's, but it took the dramatic improvements in computers to make them practical, [3]. Neural networks - just like regression techniques - model relationships between a set of input variables and an outcome. Neural networks can be thought of as computing devices consisting of numerical units ('neurons'), whose inputs and outputs are linked according to specific topologies (see the example in Figure 24-1). A neural model is defined by its free parameters - the inter-neuron connection strengths ('weights') and biases. These parameters are typically 'learned' from the training data by using an appropriate optimization algorithm. The training set consists of pairs of input (design) vectors and associated outputs (responses). The training algorithm tries to steer network parameters towards minimizing some distance measure, typically the mean squared error (MSE) of the model computed on the training data.

Several factors determine the predictive accuracy of a neural network approximation and, if not properly addressed, may adversely affect the solution. For a neural network, as well as for any other data-derived model, the most critical factor is the quality of training data. In practical cases, we are limited to a given data set, and the central problem is that of not enough data. The minimal number of data points required for network training is related to the (unknown) complexity of the underlying function and the dimensionality of design space. In reality, the more design variables, the more training samples are required. In the statistical and neural network literature this problem is known as the ‘curse of dimensionality’. Most forms of neural networks (in particular, feedforward networks) actually suffer less from the curse of dimensionality than some other methods, as they can concentrate on a lower-dimensional section of the high-dimensional space. For example, by setting the outgoing weights from a particular input to zero, a network can entirely ignore that input – see Figure 24-1. Nevertheless, the curse of dimensionality is still a problem, and the performance of a network can certainly be improved by eliminating unnecessary input variables.

It is clear that if the number of network free parameters is sufficiently large and the training optimization algorithm is run long enough, it is possible to drive the training MSE error as close as one likes to zero. However, it is also clear that driving MSE all the way to zero is not a desirable thing to do. For noisy data, this may indicate over-fitting rather than good modeling. For highly discrepant training data, zero MSE makes no sense at all. Regularization means that some constraints are applied to the construction of the neural model with the goal of reducing the 'generalization error', that is, the ability to predict (interpolate) the unobserved response for new data points that are generated by a similar mechanism as the observed data. A fundamental problem in modeling noisy and/or incomplete data is to balance the 'tightness' of the constraints with the 'goodness of fit' to the observed data. This tradeoff is called the *bias-variance tradeoff* in the statistical literature.

A multilayer *feedforward network* and a *radial basis function network* are the two most common neural architectures used for approximating functions. Networks of both types have a distinct layered topology in the sense that their processing units (‘neurons’) are divided into several groups ('layers'), the outputs of each layer of neurons being the inputs to the next layer (Figure 24-1).

In a feedforward network, each neuron performs a biased weighted sum of their inputs and passes this value through a transfer (activation) function to produce the output. Activation function of intermediate ('hidden') layers is generally a Sigmoidal function (Figure 24-2), while network input and output layers are usually linear (transparent). In theory, such networks can model functions of almost arbitrary complexity, see [4] and [5]. All parameters in a feedforward network are usually determined at the same time as part of a single (non-linear) optimization strategy based on the standard gradient algorithms (the steepest descent, RPROP, Levenberg-Marquardt, etc.). The gradient information is typically obtained using a technique called backpropagation, which is known to be computationally effective [6]. For feedforward networks, regularization may be done by controlling the number of network weights (‘model selection’), by imposing penalties on the weights (‘ridge regression’) [7], or by various combinations of these strategies [8].

A radial basis function network has a single hidden layer of radial units, each actually modeling a response function, peaked at the center, and monotonically varying outwards (Figure 24-3). Each unit responds (non-linearly) to the distance of points from its center. The RBF network output layer is typically linear. Intuitively, it is clear that a weighted sum of the sufficient radial units will always be enough to model any set of training data (see Figure 24-4 and Figure 24-5). The formal proofs of this property can be found, for example, in [9] and [10]. An RBF network can be trained extremely quickly, orders of magnitude faster than a feedforward network. The training process typically takes place in two distinct stages. First, the centers and deviations of the radial units (i.e. the hidden layer's weights) must be set; then the linear output layer is optimized. It is important that deviations are chosen so that RBFs overlap with some nearby units. Discovering a sub-optimal ‘spread’ parameter typically implies the preliminary experimental stage. If the RBFs are too spiky, the network will not interpolate between known points (see Figure 24-6). If the RBFs are very broad, the network loses fine detail (Figure 24-7). This is actually another manifestation of the over/under-fitting dilemma.

In the final shape, *after* training, a multilayer neural network with linear output (Figure 24-1) can resemble a general linear regression model - a least squares approximation. The major differences lie in the choice of basis functions and in the algorithms to construct the model (i.e. to adjust model's free parameters). Techniques to identify the systematical errors in the model and to estimate the uncertainty of model's prediction of future observations also become more complex. Unlike polynomial regressors, hidden neurons do not lend themselves to immediate interpretations in terms of input (design) variables.

The next sections discuss various goodness-of-fit assessment approaches applicable to neural networks. We also discuss how to estimate the variance of the neural model and how to compute derivatives of a neural network with respect to any of its inputs. Two neural network types, feedforward and radial basis, are considered.

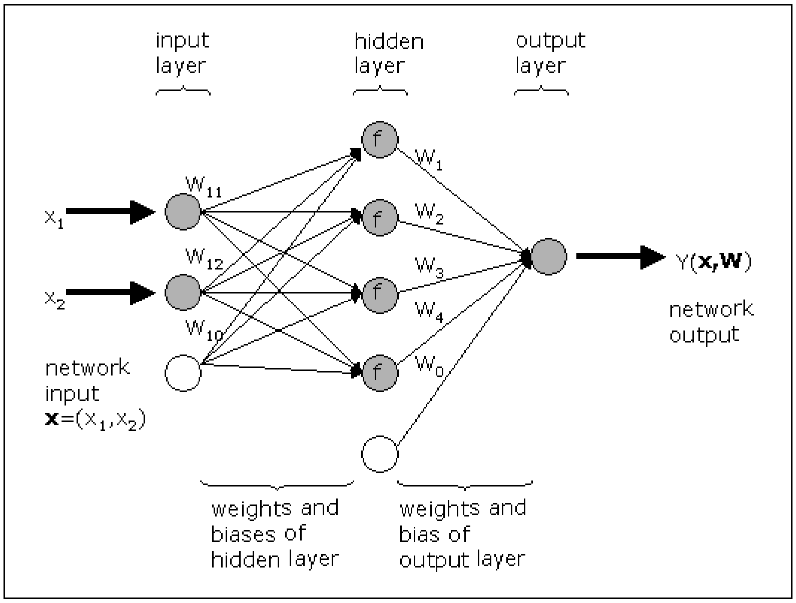


Figure 24-1: Schematic of a neural network with 2 inputs and a hidden layer of 4 neurons with activation function f.

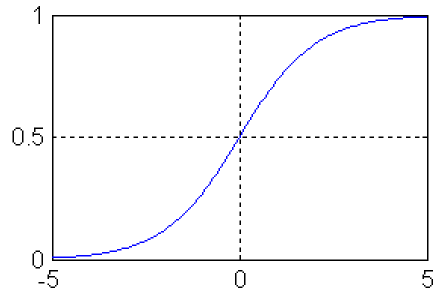


Figure 24-2: Sigmoid transfer function y=1/(1+exp(-x)) typically used with feedforward networks

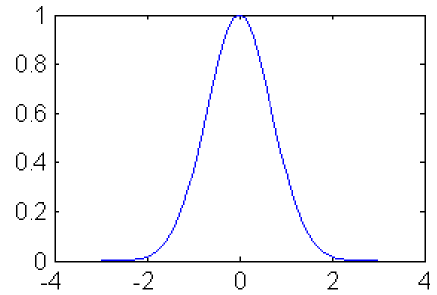


Figure 24-3: Radial basis transfer function y=exp(-x^2)

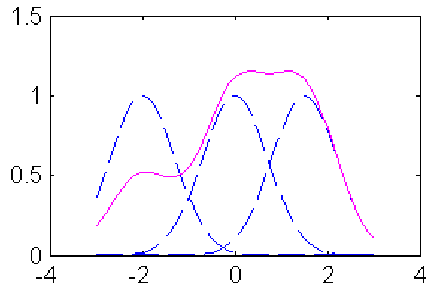


Figure 24-4: Weighted sum of radial basis transfer functions. Three radial basis functions (dashed lines) are scaled and summed to produce a function (solid line).

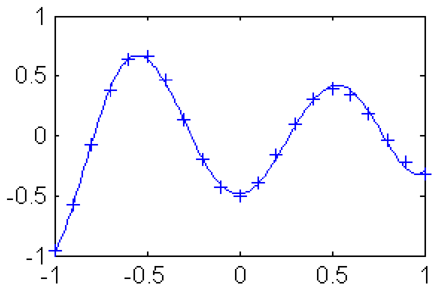


Figure 24-5: A radial basis network approximation (solid line) of the function, which fits the 21 data points (plus symbols).

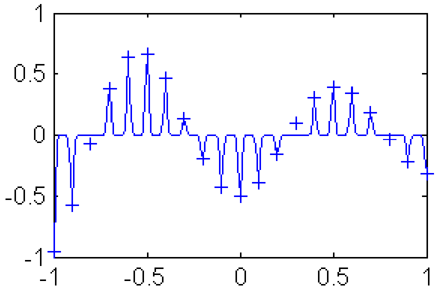


Figure 24-6: The same 21 data points as in Figure 24-5. Test points reveal that the function has been overfit. RBF neuron's spread is too small. RBF network could have done better with a higher spread constant.

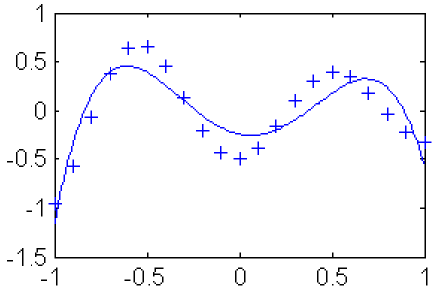


Figure 24-7: The same 21 data points as in Figure 24-5. Approximation with overlapping RBF neurons. The spread of RBF units is too high.

24.1.1. Model adequacy checking

Nature is rarely (if ever) perfectly predictable. Real data never exactly fit the model that is being used. One must take into consideration that the prediction errors not only come from the variance error due to the intrinsic noise and unreliabilities in the measurement of the dependent variables but also from the systematic (bias) error due to model mis-specification. According to George E.P. Box's famous maxim, "all models are wrong, some are useful". To be genuinely useful, a fitting procedure should provide the means to assess whether or not the model is appropriate and to test the goodness-of-fit against some statistical standard. There are several error measures available to determine the accuracy of the model. Among them are:

$$MSE = \sum_{i}^{P} \left(\hat{y}_i - y_i\right)^2 \Big/ P, \tag{24-1}$$

$$RMS = \sqrt{MSE}; \qquad nMSE = MSE \big/ \hat{\sigma}^2; \qquad nRMS = RMS \big/ \hat{\sigma}^2. \tag{24-2}$$

$$R^2 = \frac{\sum_{i}^{P} \left(\hat{y}_i - \bar{y}\right)^2}{\sum_{i}^{P} \left(y_i - \bar{y}\right)^2}, \quad \text{and} \quad R = \frac{\sum_{i}^{P} \left|\hat{y}_i - \bar{\hat{y}}\right| \left|y_i - \bar{y}\right|}{\sum_{i}^{P} \left(y_i - \bar{\hat{y}}\right)^2 \sum_{i}^{P} \left(y_i - \bar{y}\right)^2}, \tag{24-3}$$

where P denotes the number of data points, y_i is the observed response value ('target value'), $\hat{y}_i$ is the model's prediction of response, $\bar{\hat{y}}$ is the mean (average) value of $\hat{y}$, $\bar{y}$ is the mean (average) value of y, and $\hat{\sigma}^2$ is given by

$$\hat{\sigma}^2 = \frac{\varepsilon^2}{P - L} = \frac{\sum_{i=1}^{P} \left(y_i - \hat{y}_i\right)^2}{P - L}. \tag{24-4}$$

Mean squared error (MSE for short) and root mean squared error (RMS) summarize the overall model error. Unique or rare large error values can affect these indicators. Sometimes, MSE and RMS measures are normalized with sample variance of the target value (see formulae for nMSE and nRMS) to allow for comparisons between different datasets and underlying functions. R^2 and R are relative measures. The coefficient of multiple determination R^2 ('R-square') is the explained variance relative to the total variance in the target value. This indicator is widely used in linear regression analysis. R^2 represents the amount of response variability explained by the model. R is the correlation coefficient between the network response and the target. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is a perfect correlation between targets and outputs. Outliers can greatly affect the magnitudes of correlation coefficients. Of course, the larger the sample size, the smaller is the impact of one or two outliers.

Training accuracy measures (MSE, RMS, R^2, R, etc.) are computed along all the data points used for training. As mentioned above, the performance of a good model on the training set does not necessarily mean good prediction of new (unseen) data. The objective measures of the prediction accuracy of the model are test errors computed along independent testing points (i.e. not training points). This is certainly true provided that we have an infinite number of testing points. In practice, however, test indicators are usable, only if treated with appropriate caution. Actual problems are often characterized by the limited availability of data, and when the training datasets are quite small and the test sets are even smaller, only quite large differences in performance can be reliably discerned by comparing training and test indicators.

The generalized cross-validation (GCV) [11] and Akaike's final prediction error (FPE) [12] provide computationally feasible means of estimating the appropriateness of the model. The *k*-fold cross-validation (denoted here as CV-*k*), generalized cross-validation (GCV) [11] and Akaike's final prediction error (FPE) [12] provide computationally feasible means of estimating the appropriateness of the model.

GCV and FPE estimates combine the training MSE with a measure of the model complexity:

$$MSE_{GCV} = MSE \big/ (1-\nu/P)^2 , \tag{24-5}$$

$$RMS_{GCV} = \sqrt{MSE_{GCV}}\,,\; nMSE_{GCV} = MSE_{GCV} \big/ \hat{\sigma}^2 ; \qquad nRMS_{GCV} = RMS_{GCV} \big/ \hat{\sigma}^2 . \tag{24-6}$$

$$MSE_{FPE} = MSE\,(1+\nu/P)\big/(1-\nu/P), \tag{24-7}$$

$$RMS_{FPE} = \sqrt{MSE_{FPE}}\,,\; nMSE_{FPE} = MSE_{FPE} \big/ \hat{\sigma}^2 ; \qquad nRMS_{FPE} = RMS_{FPE} \big/ \hat{\sigma}^2 . \tag{24-8}$$

where ν is the (effective) number of model parameters.

In theory, GCV estimates should be related to ν. As a very rough approximation to ν, we can assume that all of the network free parameters are well determined so that $\nu = M$, where *M* is the total number of network weights and biases. This is what we would expect to be the case for large *P* so that *P* >> *M*. Note that GCV is undefined when ν is equal to the number of training points (*P*). In theory, GCV and FPE estimates should be related to the effective number of model's parameters ν. Techniques to assess ν for neural networks will be discussed later. As a very rough approximation, we can assume that all of the network free parameters are well determined so that ν = *M*, where *M* is the total number of network weights and biases. This is what we would expect to be the case for large *P* so that *P* >> *M*. Note that both GCV and FPE are undefined when the effective number of model's parameters (ν) is equal to the number of training points (*P*). GCV and FPE measures are asymptotically equivalent for large *P*.

In *k*-fold cross-validation the training dataset is divided into *k* randomly selected disjoint subsets of roughly equal size $P^{(j)}$. The model is trained and tested *k* times. Each time $j = 1,\ldots,k$ it is trained on all data except for points from subset *j* and then tested on *j*-th subset. Formally, let $y^{(j)} = (y_i^{(j)}), i = 1,\ldots,P^{(j)}$ be the prediction of such a model for the points from subset *j*. Then the CV-*k* estimates of accuracy

$$MSE_{CV-k} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{P^{(j)}}\left(y_i^{(j)} - \hat{y}_i\right)^2}{P} \tag{24-9}$$

$$RMS_{CV-k} = \sqrt{MSE_{CV-k}}\,,\; nMSE_{CV-k} = MSE_{CV-k} \big/ \hat{\sigma}^2 ; \qquad nRMS_{CV-k} = RMS_{CV-k} \big/ \hat{\sigma}^2 . \tag{24-10}$$

The CV estimate is a random number that depends on the division into folds. Repeating cross-validation multiple times using different splits into folds provides a better approximation to complete *N*-fold cross-validation (leave-one-out). Leave-one-out measure is almost unbiased, but for typical real world datasets it has high variance, leading to unreliable estimates. Small datasets are simply not suitable for CV estimates, since data distribution can change considerably after we separate out even a small portion of data. In addition, the CV approach is usually too expensive. The question is whether the advantages of CV (if any) are big enough to justify the computational cost of training multiple networks rather than one.

Anyway, no accuracy estimation can be correct all the time. Most probably it is impossible to evaluate a model by means of a single descriptive measure. We should always consider several accuracy measures when deciding on the appropriateness of the model, especially if this model is trained on noisy and/or incomplete data. In certain cases the crucial phase of integrating disparate measures into a single judgment could be delegated to a statistical decision-making tool. Of course, when the quantity of data required for statistical methods is simply not available, human experts' knowledge should be used for the really big decisions.

24.1.2. Feedforward neural networks

Feedforward (FF) neural networks have a distinct layered topology. Each unit performs a biased weighted sum of their inputs and passes this value through a transfer (activation) function to produce the output. The outputs of each layer of neurons are the inputs to the next layer. In a feedforward network, the activation function of intermediate (’hidden’) layers is generally a sigmoidal function (Figure 24-3), network input and output layers being linear. Consider a FF network with *K* inputs, one hidden layer with *H* sigmoid units and a linear output unit. For a given input vector $\mathbf{x}=\left(x_1,...,x_K\right)$ and network weights $\mathbf{W}=\left(W_0,W_{1,}...,W_H,W_{10},W_{11},...,W_{HK}\right)$, the output of the network is:

$$\hat{y}(\mathbf{x},\mathbf{W})=W_0+\sum_{h=1}^{H}W_h f\left(W_{h0}+\sum_{k=1}^{K}W_{hk}x_k\right), \qquad (24\text{-}11)$$

Where $f(x)=\frac{1}{1+e^{-x}}$.

The computational graph of Eq. (22-11) is shown schematically in Figure 24-1. The extension to the case of more than one hidden layers can be obtained accordingly. It is straightforward to show that the derivative of the network Eq. (22-11) with respect to any of its inputs is given by:

$$\frac{\partial\hat{y}}{\partial x_k}=\sum_{h=1}^{H}W_h W_{hk} f^{'}\left(W_0+\sum_{h=1}^{H}W_h\right), k=1,...,K. \qquad (24\text{-}12)$$

Neural networks have been mathematically shown to be universal approximators of continuous functions and their derivatives (on compact sets) [4]. In other words, when a network (5) converges towards the underlying function, all the derivatives of the network converge towards the derivatives of this function.

Standard non-linear optimization techniques including a variety of gradient algorithms (the steepest descent, RPROP, Levenberg-Marquardt, etc.) are applied to adjust FF network’s weights and biases. For neural networks, the gradients are easily obtained using a chain rule technique called ‘backpropagation’ [6]. The second-order Levenberg-Marquardt algorithm appears to be the fastest method for training moderate-sized FF neural networks (up to several hundred adjustable weights) [3]. However, when training larger networks, the first-order RPROP algorithm becomes preferable for computational reasons [13].

Regularization: For FF networks, regularization may be done by controlling the number of network weights ('model selection'), by imposing penalties on the weights ('ridge regression'), or by various combinations of these strategies ([7], [8]). Model selection requires choosing the number of hidden nodes and, sometimes, the number of network hidden layers. Most straightforward is to search for an 'optimal' network architecture that minimizes MSE_{GCV}, MSE_{FPE} or $MSE_{CV\text{–}k}$. Often, it is feasible to loop over 1, 2,... hidden nodes and finally select the network with the smallest GCV error. In any event, in order for the GCV measure to be applicable, the number of training points P should not be too small compared to the required network size M.

Over-fitting: To prevent over-fitting, it is always desirable to find neural solutions with the smallest number of parameters. In practice, however, networks with a very parsimonious number of weights are often hard to train. The addition of extra parameters (i.e. degrees of freedom) can aid convergence and decrease the chance of becoming stuck in local minima or on plateaus [14]. Weight decay regularization involves modifying the performance function F, which is normally chosen to be the mean sum of squares of the network errors on the training set (Eq. (24-1)). When minimizing MSE (Eq. (24-1)) the weight estimates tend to be exaggerated. We can impose a penalty for this tendency by adding a term that consists of the sum of squares of the network weights (see also (Eq. (24-1))):

$$F = \beta E_D + \alpha E_W, \tag{24-13}$$

where

$$E_D = 0.5\sum_{i=1}^{P}\left(\hat{y}_i - y_i\right)^2,\; E_W = 0.5\sum_{i=1}^{M} W_i^{\,2},$$

where M is the number of weights and P the number of points in the training set.

Notice that network biases are usually excluded from the penalty term E_W. Using the modified performance function (Eq. 22-13) will cause the network to have smaller weights, and this will force the network response to be smoother and less likely to overfit. This eliminates the guesswork required in determining the optimum network size. Unfortunately, finding the optimal value for α and β is not a trivial task. If we make α/β too small, we may get over-fitting. If α/β is too large, the network will not adequately fit the training data. A rule of thumb is that a little regularization usually helps [15]. It is important that weight decay regularization does not require that a validation subset be separated out of the training data. It uses all of the data. This advantage is especially noticeable in small sample size situations. Another nice property of weight decay regularization is that it can lend numerical robustness to the Levenberg-Marquardt algorithm. The L-M approximation to the Hessian of Eq. (24-13) is moved further away from singularity due to a positive addend to its diagonal:

$$\mathbf{A} = \mathbf{H} + \alpha\mathbf{I} \tag{24-14}$$

where

$$\mathbf{H} = \beta\nabla\nabla E_D \approx \sum_{i=1}^{P} g\left(x^{(i)}\right) g\left(x^{(i)}\right)^T,$$

$$\mathbf{g}(\mathbf{x}) = \left(\frac{\partial \hat{y}}{\partial W_1}, ..., \frac{\partial \hat{y}}{\partial W_M} \right)^T .$$

In [3], [16], [17]and [18] the Bayesian ('evidence framework' or 'type II maximum likelihood') approach to regularization is discussed. The Bayesian re-estimation algorithm is formulated as follows. At first, we choose the initial values for α and β. Then, a neural network is trained using a standard non-linear optimization algorithm to minimize the error function (Eq. (24-13)). After training, i.e. in the minimum of Eq. (24-13), the values for α and β are re-estimated, and training restarts with the new performance function. Regularization hyper-parameters are computed in a sequence of 3 steps:

$$\nu = \frac{\sum_{m=1}^{M} \lambda_m}{\lambda_m + \alpha}, \tag{24-15}$$

where λ_{m}, m = 1,…,M are (positive) eigenvalues of matrix **H** in Eq. (24-14), ν is the estimate of the effective number of parameters of a neural network,

$$\alpha = \nu / 2E_W ;$$

$$\beta = (P - \nu) / 2E_D .$$

It should be noted that the algorithm (Eq. (24-15)) relies on numerous simplifications and assumptions, which hold only approximately in typical real-world problems [[19]]. In the Bayesian formalism a trained network is described in terms of the posterior probability distribution of weight values. The method typically assumes a simple Gaussian prior distribution of weights governed by an inverse variance hyper-parameter $\alpha = 1/\sigma^2_{weights}$. If we present a new input vector to such a network, then the distribution of weights gives rise to a distribution of network outputs. There will be also an addend to the output distribution arising from the assumed $\sigma^2_{noise} = 1/\beta$ Gaussian noise on the output variables:

$$y = y(\mathbf{x}) + N(0, \sigma^2_{noise}).$$

With these assumptions, the negative log likelihood of network weights **W** given P training points ***x***(1), … , ***x***(P) is proportional to MSE (Eq. 22-1)), i.e., the maximum likelihood estimate for **W** is that which minimizes (Eq. 22-1) or, equivalently, E_D. In order for Bayes estimates of α and β to do a good job of minimizing the generalization in practice, it is usually necessary that the priors on which they are based are realistic. The Bayesian formalism also allows us to calculate error bars on the network outputs, instead of just providing a single 'best guess' output $\hat{y}$. Given an unbiased model, minimization of the performance function (Eq. 22-1) amounts to minimizing the variance of the model. The estimate for output variance $\sigma^2_{\hat{y}|x}$ of the network at a particular point **x** is given by:

$$\sigma^2_{\hat{y}|x} \approx \mathbf{g}(\mathbf{x})^T \mathbf{A}^{-1} \mathbf{g}(\mathbf{x}). \tag{24-16}$$

Equation (24-16) is based on a second-order Taylor series expansion of Eq. (24-13) around its minimum and assumes that $\partial\hat{y}/\partial W$ is locally linear.

Variability of Feedforward neural networks

Neural networks have a natural variability because of the following reasons [20]:

1. Local behavior of the neural network training algorithms
2. Uncertainty (noise) in the training data.

The neural network training error function usually has multiple local and global minima. With different initial weights, the training algorithm typically ends up in different (but usually almost equally good/bad) local minima. The larger the amount of noise in the data, the larger is the difference between these NN solutions. The user is allowed to specify a neural network committee to find the average net and quantify the variability (Section 9.1.3). The starting weights for network training are randomly generated using a user-specified seed to ensure repeatability (see Section 23.2.7).

24.1.3. Radial basis function networks

A radial basis function neural network has a distinct 3-layer topology. The input layer is linear (transparent). The hidden layer consists of non-linear radial units, each responding to only a local region of input space. The output layer performs a biased weighted sum of these units and creates an approximation of the input-output mapping over the entire space.

While several forms of radial basis function are considered in the literature, the most common functions are the Hardy's multi-quadrics and the Gaussian basis function. These are given as:

Hardy's multi-quadric:

$$g_h(x_1,...,x_K) = \sqrt{1+\left(r^2/\sigma_h^2\right)}. \tag{24-17}$$

Gaussian:

$$g_h(x_1,...,x_K) = \exp\left[-r^2/2\sigma_h^2\right] \tag{24-18}$$

The activation of h^{th} radial basis function is determined by the Euclidean distance $r = \sqrt{\sum_{k=1}^{K}\left(x_k - W_{hk}\right)^2}$ between the input vector $\mathbf{x} = \left(x_1,...,x_K\right)$ and RBF center $W_h = \left(W_{h1},...,W_{hk}\right)$ in *K*-dimensional space. The Gaussian basis function is a localized function (peaked at the center and descending outwards) with the property that $g_h \to 0$ as $r \to \infty$. Parameter σ_h controls the smoothness properties of the RBF unit.

For a given input vector **x** the output of RBF network with K inputs and a hidden layer with *H* basis function units is given by (see also Eqs. (24-17) and (24-18)):

$$Y\left(\mathbf{x},\mathbf{W}\right) = W_0 + \sum_{h=1}^{H} W_h f\left(\rho_h\right) \tag{24-19}$$

where

$$\rho_h = W_{h0} \sum_{k=1}^{K} \left(x_k - W_{hk}\right)^2; \quad W_{h0} = 1/\left(2\sigma_h^2\right); \quad f\left(\rho\right) = \exp(-\rho).$$

Notice that hidden layer parameters $W_h = \left(W_{h1},...,W_{hk}\right)$ represent the center of h^{th} radial unit, while W_{h0} corresponds to its deviation. Parameters W_0 and $W_1,...,W_H$ are the output layer's bias and weights, respectively.

A linear super-position of localized functions as in (24-13) is capable of universal approximation. The formal proofs of this property can be found, for example, in [9] and [10]. It is straightforward to show that the derivative of the network (24-13) with respect to any of its inputs is given by:

$$\frac{\partial Y}{\partial x_k} = \sum_{h=1}^{H} W_h W_{h0} \cdot 2(x_k - W_{hk}) \cdot f^{'}\left(\rho_h\right), \quad k = 1, ..., K, \tag{24-20}$$

where $f\,'$ denotes the first derivative of the transfer function $f : f^{'}\left(\rho\right) = -\exp(-\rho).$

Theory tells us that when a network (Equation (24-19)) converges towards the underlying function, all the derivatives of the network converge towards the derivatives of this function.

A key aspect of RBF networks, as distinct from feedforward neural networks, is that they can be interpreted in a way which allows the hidden layer parameters (i.e. the parameters governing the radial functions) to be determined by semi-empirical, unsupervised training techniques. Accordingly, although a radial basis function network may require more hidden nodes than a comparable feedforward network, RBF networks can be trained extremely quickly, orders of magnitude faster than FF networks.

For RBF networks, the training process generally takes place in two distinct stages. First, the centers and deviations of the radial units (i.e. hidden layer parameters $W_{11},...,W_{HK}$ and $W_{10},...,W_{H0}$) must be set. All the basis functions are then kept fixed, while the linear output layer (i.e., $W_0,...,W_H$) is optimized in the second phase of training. In contrast, all of the parameters in a FF network are usually determined at the same time as part of a single training (optimization) strategy. Techniques for selecting $W_{11},...,W_{HK}$ and $W_{10},...,W_{H0}$ are discussed at length in following paragraphs. Here we shall assume that the RBF parameters have already been chosen, and we focus on the problem of optimizing the output layer weights.

Mathematically, the goal of output layer optimization is to minimize a performance function, which is normally chosen to be the mean sum of squares of the network errors on the training set (Equation (24-1)). If the hidden layer's parameters $W_{10,}W_{11},...,W_{HK}$ in (24-12) are kept fixed, then the performance function (Equation (24-1)) is a quadratic function of the output layer' parameters $W_0,...,W_H$ and its minimum can be found in terms of the solution of a set of linear equations (e.g., using singular value decomposition). The possibility of avoiding the need for time-consuming and costly non-linear optimization during training is one of the major advantages of RBF networks over FF networks. However, when the number of optimized parameters ($H+1$, in our case) is small enough, non-linear optimization (Levenberg-Marquardt, etc.) may also be cost-effective.

It is clear that the ultimate goal of RBF neural network training is to find a smooth mapping which captures the underlying systematic aspects of the data without fitting the noise. However, for noisy data, the exact RBF network, which passes exactly through every training data point, is typically a highly oscillatory function. There are a number of ways to address this problem. By analogy with FF network training, one can add to (Eq. (24-1)) a regularization term that consists of the mean of the sum of squares of the optimized weights. In conventional curve fitting this form of regularization is called ridge regression. The sub-optimal value for hyperparameters α and β in (24-13) can be found by applying Bayesian re-estimation formulae (Eq. (24-14)) - (Eq. (24-15)). It is also feasible to iterate over several trial values of α and β.

For RBF networks, however, the most effective regularization methods are probably those pertaining to selecting radial centers and deviations in the first phase of RBF training. The commonly held view is that RBF centers and deviations should be chosen so as to form a representation of the probability density of the input data. The classical approach is to set RBF centers equal to all the input vectors from the training dataset. The width parameters σ_h are typically chosen – rather arbitrarily – to be some multiple S_σ of the average spacing between the RBF centers (e.g. to be roughly twice the average distance). This ensures that the RBF's overlap to some degree and hence give a relatively smooth representation of data.

To simplify matters, the same value of the width parameter σ_h for all RBF units is usually considered. Sometimes, instead of using just one value for all RBF's, each RBF unit's deviation σ_h is individually set to the distance to its $N_\sigma << N$ nearest neighbors. Hence, deviations σ_h become smaller in densely populated areas of space, preserving fine detail, and are higher in sparse areas of space, interpolating between points where necessary. Again the choice of N_σ is somewhat arbitrary. RBF networks with individual radial deviations σ_h can be particularly useful in situations where data tend to cluster in only a small subregion of the design space (for example, around the optimum of the underlying system which RSM is searching for) and are sparse elsewhere.

One must take into consideration that after the first phase of RBF training is over, there's no way to compensate for large inaccuracies in radial deviations σ_h by, say, adding a regularization term to the performance function. If the basis functions are too spiky, the network will not interpolate between known points, and thus, will lose the ability to generalize. If the Gaussians are very broad, the network is likely to lose fine detail. The popular approach to find a sub-optimal spread parameter is to loop over several trial values of S_σ and N_σ, and finally select the RBF network with the smallest GCV (FPE, CV-k) error. This is somewhat analogous to searching for an optimal number of hidden nodes of a feedforward neural network.

In order to eliminate all the guesswork required in determining RBF deviations, it might seem natural to treat $W_{10},...,W_{H0}$ ($\sigma_1,...,\sigma_H$, to be precise) in (Eqs. 22-19) as adjustable parameters, which are optimized in the second phase of training along with the output layer's weights and bias. Practical applications of this approach, however, are rare. The reason may be that it requires the use of a non-linear optimization method in combination with a sophisticated regularization scheme specially designed so as to guarantee that the radial functions will remain localized.

It should be noted that RBF networks may have certain difficulties if the number of RBF units (*H*) is large. This is often the case in multidimensional problems. The difficulty arises because for a large number of RBF's, a large number of training samples are required in order to ensure that the neural network parameters are properly determined. A large number of RBF units also increase the computation time spent on optimization of the network output layer and, consequently, the RBF architecture loses its main (if not the only one) advantage over FF networks – fast training.

Radial basis function networks actually suffer more from the curse of dimensionality than feedforward neural networks. To explain this statement, consider the effect of adding an extra, perfectly spurious input variable to a network. A feedforward network can learn to set the outgoing weights of the spurious input to zero, thus ignoring it. An RBF network has no such luxury: data in the relevant lower-dimensional space get ‘smeared’ out through the irrelevant dimension, requiring larger numbers of units to encompass the irrelevant variability.

In principle, the number of RBF's (*H*) need not equal the number of training samples (*P*), and RBF units are not constrained to be centered on the training data points. In fact, when data contain redundant information, we do not need all data points in learning. One simple procedure for selecting RBF centers is to set them equal to a random subset of the input vectors from the training set. Since they are randomly selected, they will 'represent' the distribution of the (redundant) training data in a statistical sense. Of course, *H* and *P* should not be too small in this case.

It is clear, however, that the optimal choice of RBF centers based on the input data alone need not be optimal for representing the input-output mapping as reflected in the observed data. In order to overcome these limitations, the selection procedure should take into account the output values, or at least, approximate estimates (assumptions) of the global behavior of the underlying system. The common neural term for such techniques involving output values is ‘active learning’. In the context of active learning, RBF networks can be thought of as DOE metamodels analogous to polynomials, [16] and [19]. Given a candidate list of points, an active learner is searching for the 'best' points in order to position RBF centers. Popular in neural applications is to treat RBF active learning as 'pruning' technique intended for identifying critical data and discarding redundant points, or more accurately, not selecting some training points as RBF centers. RBF active learning methods are being successfully applied to approximate huge datasets that come from natural stochastic processes. It is questionable, however, whether active learning can be useful for non-redundant datasets, specifically for RSM design sets generated by performing DOE analysis based on low-order polynomial metamodels.

To briefly summarize, parameters governing radial units (radial centers and deviations) play a key role in generalization performance of a RBF model. The appropriate selection of RBF centers implies that we choose a minimal number of training data points that carry enough information to build an adequate input-output representation of the underlying function. Unfortunately, this is easier said than done. Indeed, there is a general agreement that selecting RBF centers and deviations is more Art than Science.

24.2. Kriging*

Kriging is named after D. G. Krige [22], who applied empirical methods for determining true ore grade distributions from distributions based on sampled ore grades. In recent years, the Kriging method has found wider application as a spatial prediction method in engineering design. Detailed mathematical formulations of Kriging are given by Simpson [23] and Bakker [24].

The basic postulate of this formulation [23] is:

$$y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}), \tag{24-21}$$

where *y* is the unknown function of interest, *f(***x***)* is a known polynomial, the trend model, and *Z(***x***) is* the stochastic component with mean zero and covariance:

$$\mathrm{Cov}[Z(\mathbf{x}^i),Z(\mathbf{x}^j)] = \sigma^2\mathbf{R}([R(\mathbf{x}^i,\mathbf{x}^j)]). \qquad (24\text{-}22)$$

With *L* the number of sampling points, **R** is the *L* x *L* correlation matrix with *R(***x***ⁱ,***x***ʲ)* the correlation function between data points $\mathbf{x}^i$ and $\mathbf{x}^j$. **R** is symmetric positive definite with unit diagonal.

Two commonly applied correlation functions used are:

1. Exponential: $R = \prod_{k=1}^{n} \exp\left(-\Theta_k \left|d_k\right|\right)$ and
2. Gaussian: $R = \prod_{k=1}^{n} \exp\left(-\Theta_k d_k^2\right).$

where *n* is the number of variables and $d_k = x_k^{\,i} - x_k^{\,j}$, the distance between the k^{th} components of points $\mathbf{x}^i$ and $\mathbf{x}^j$. There are *n* unknown Θ-values to be determined. The default function in LS-OPT is Gaussian.

Once the correlation function has been selected, the predicted estimate of the response $\hat{y}(\mathbf{x})$ is given by:

$$\hat{y} = \hat{\beta} + r^T(x)R^{-1}(y-f\hat{\beta}) \qquad (24\text{-}23)$$

where $\mathbf{r}^T(\mathbf{x})$ is the correlation vector (length *L*) between a prediction point $\mathbf{x}$ and the *L* sampling points, **y** represents the responses at the *L* points and **f** is the trend model, a *L*-vector of basis functions (ones, if *f(***x***)* is taken as a constant). One can choose either a constant, linear, or quadratic basis function in LS-OPT. The default choice is the constant basis function.

The vector **r** and scalar $\hat{\beta}$ are given by:

$$\mathbf{r}^T(\mathbf{x}) = [R(\mathbf{x},\mathbf{x}^1),R(\mathbf{x},\mathbf{x}^2),\ldots,R(\mathbf{x},\mathbf{x}^L)]^T$$

$$\hat{\beta} = (f^{\,T}R^{\,-1}f)^{-1}f^{\,T}R^{\,-1}y.$$

The estimate of variance from the underlying global model is:

$$\hat{\sigma}^2 = \frac{1}{L}\left(\mathbf{y}-\mathbf{f}\hat{\beta}\right)^T R^{-1}\left(\mathbf{y}-\mathbf{f}\hat{\beta}\right).$$

The maximum likelihood estimates for Θ_k, $k = 1,\ldots, n$ can be found by solving the following constrained maximization problem:

$$\max \Phi(\Theta) = -\frac{1}{2}\left[L\ln\left(\hat{\sigma}^2\right) + \ln\left|\mathbf{R}\right|\right], \text{ subject to } \Theta_k > 0 \ \wedge k.$$

where both $\hat{\sigma}$ and $\left|\mathbf{R}\right|$ | are functions of $\boldsymbol{\Theta}$. This is the same as minimizing

$$\hat{\sigma}^2 \left|\mathbf{R}\right|^{(1/n)},\ subject\ \ to: \mathbf{\Theta} > 0.$$

This optimization problem is solved using the real-coded genetic algorithm (Section 26.9). A small constant number is adaptively added to the diagonal of matrix **R** to avoid ill-conditioning. The net effect is that the approximating functions might not interpolate the observed response values exactly. However, these observations are still closely approximated.

24.3. Support Vector Regression

Support vector regression (SVR) is a sub category of support vector machines (SVM) that can be used for both regression and classification [32]. It is based on the Vapnik-Chervonenkis (VC) theory, and has the ability to have good generalization properties. Instead of empirical risk minimization, it is based on structural risk minimization that balances the model's complexity and its ability to fit known data. The particular implementation of SVR in LS-OPT is referred to as $\varepsilon - SVR$. Given a set of training data that consists of N samples $\mathbf{x}_i \left(i = 1,2,\cdots,N\right)$ and their corresponding response values y_i, $\varepsilon - SVR$ attempts to find a function $\hat{f}\left(\mathbf{x}\right)$ that has at the most ε deviation from the actual response values y_i and is as flat as possible (low complexity) at the same time. In the case of linear functions, the approximated function is:

$$\hat{f}\left(\mathbf{x}\right) = \left\langle \mathbf{w}, \mathbf{x} \right\rangle + b \tag{24-24}$$

where $\mathbf{w}$ is a weight vector and b is a scalar bias. In order to obtain the "flattest" or most simple function approximation (to avoid overfitting), the norm of $\mathbf{w}$ is minimized. Thus the SVR solution is obtained as the result of the following optimization:

$$\begin{array}{ll} \min\limits_{\mathbf{w},b} & \frac{1}{2}\left\|\mathbf{w}\right\|^2 \\ s.t. & \left|\left\langle \mathbf{w}, \mathbf{x}_i \right\rangle + b - y_i\right| \le \varepsilon \qquad \forall i \in \left[1, N\right] \end{array} \tag{24-25}$$

To ensure the feasibility of the optimization, two slack variables ξ and ξ^* are introduced:

$$\begin{array}{ll} \min\limits_{\mathbf{w},b} & \frac{1}{2}\left\|\mathbf{w}\right\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ s.t. & -\left\langle \mathbf{w}, \mathbf{x}_i \right\rangle - b + y_i \le \varepsilon + \xi_i \qquad \forall i \in \left[1, N\right] \\ & \left\langle \mathbf{w}, \mathbf{x}_i \right\rangle + b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{array} \tag{24-26}$$

where C is a cost or penalty parameter that ensures a solution with slack variables as close to zero as possible. In general, the dual formulation of the SVR optimization is solved:

$$\begin{aligned}&\max_{\alpha_i\alpha_i^*} && -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_J-\alpha_J^*)\langle\mathbf{x}_i,\mathbf{x}_j\rangle-\varepsilon\sum_{i=1}^{N}(\alpha_J+\alpha_J^*)+\sum_{i=1}^{N}y_i(\alpha_i-\alpha_i^*)\\ &s.t. && \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0\\ &&& \alpha_i\in[0,C]\qquad\forall i\in[1,N]\end{aligned}\tag{24-27}$$

where α_i and α_i^* are Lagrange multipliers of the original problem that become the optimization variables for the dual formulation, ε is the maximum deviation from the actual response values without any penalty, y_i are the actual response values, C is the cost or penalty parameter and N is the number of training samples. The weight vector **w** eliminated in the dual formulation can be written as:

$$\mathbf{w}=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\mathbf{x}_i\tag{24-28}$$

The support vector regression expression can therefore be written as:

$$f(\mathbf{x})=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\langle\mathbf{x}_i,\mathbf{x}\rangle+b\tag{24-29}$$

where the Lagrange multipliers are obtained by solving the dual problem. The value of b can be obtained using the KKT conditions [33].

In the general nonlinear case, the inner product is replaced by a kernel function K:

$$\begin{aligned}&\max_{\alpha_i\alpha_i^*} && -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_J-\alpha_J^*)K(\mathbf{x}_i,\mathbf{x}_j)-\varepsilon\sum_{i=1}^{N}(\alpha_J+\alpha_J^*)+\sum_{i=1}^{N}y_i(\alpha_i-\alpha_i^*)\\ &s.t. && \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0\\ &&& \alpha_i\in[0,C]\qquad\forall i\in[1,N]\end{aligned}\tag{24-30}$$

The optimization is solved using sequential minimal optimization (SMO) [34]. The final SVR expression is:

$$f(\mathbf{x})=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)K(\mathbf{x}_i,\mathbf{x})+b\tag{24-31}$$

The kernel function must satisfy Mercer's condition [32],[33]. Some of the possible kernels are polynomial, exponential radial basis function, Gaussian radial basis function, hyperbolic tangent etc. Two options are available in LS-OPT for the kernel:

1. Gaussian RBF
2. Polynomial

The polynomial kernel is given as

$$K(\mathbf{x}_i,\mathbf{x})=\left(1+\langle\mathbf{x}_i,\mathbf{x}\rangle\right)^m$$

where *m* is the order of the polynomial that is internally optimized in LS-OPT using 5-fold cross-validation [35].

The Gaussian RBF kernel is given as:

$$K(\mathbf{x}_i, \mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}}$$

Here, σ is the width parameter or spread of the kernel that is internally optimized in LS-OPT using cross-validation. In addition to the kernel parameters, the SVR parameters C and ε are also internally optimized in LS-OPT during cross-validation. The default kernel type is Gaussian RBF; a different kernel can be selected under "Advanced SVR Settings", Section 9.1.5.

24.4. Concluding remarks: which metamodel?

There is little doubt that the polynomial-based response surfaces are very robust, especially for sequential optimization methods. A negative aspect of using polynomials is the fact that the user is obliged to choose the order of polynomial. Also, a greater possibility exists for bias error of a nonlinear response. They are also, in most cases, not suitable for updating in sequential methods. Linear approximations may only be useful within a certain subregion and therefore quadratic polynomials or other higher order approximations such as RBF networks may be required for greater global accuracy. However the linear SRSM method has proved to be excellent for sequential optimization and can be used with confidence [25][26][27].

RBF Networks appear to be generally the best of the neural networks metamodels. They have the following advantages:

- Higher prediction accuracy due to built-in cross validation. Although FF networks may appear more accurate due to a smaller fitting error (RMSE), their prediction error is generally larger than that of RBF networks. An appealing plot of predicted vs. computed responses showing the training points or R^2 values approaching unity or small RMS error values should not be construed as representing a higher accuracy.
- Higher speed due to their linear nature. When sizable FF committees (e.g. with 9 members) are used they may be vastly more expensive to construct than RBF networks. This is true especially for a relatively small number of variables.
- Relative independence of the calculation time with respect to the number of functions. Although there is a slight overhead which depends on this number, the user does not have to be as careful with limiting the number of responses.

FF Neural Networks function well as global approximations and no serious deficiencies have been observed when used as prescribed in Section 26.5. FF networks have been used for sequential optimization [27] and can be updated during the process. A more recent study [28] which focuses on the accuracy comparison for FF neural networks and RBF networks for different types of optimization strategies concluded that, for crashworthiness analysis, RBF and FF metamodels are mostly similar in terms of the accuracy of a large number of checkpoint results. However, the same study showed that Neural Networks are sometimes better than RBF networks for smooth problems. As mentioned earlier, RBF networks have a distinct speed advantage. Reference [28] also assesses the use of FF committees and concludes that, although expensive, there are some cases where they may be necessary.

Although the literature seems to indicate that Kriging is one of the more accurate methods [23], there is evidence of Kriging having fitting problems with certain types of experimental designs [29]. Kriging is very sensitive to noise, since it interpolates the data [30]. The authors of this manual have also experienced fitting problems with non-smooth surfaces ($Z(\mathbf{x})$ observed to peak at data points) in some cases, apparently due to large values of $\mathbf{\Theta}$ that may be due to local optima of the maximum likelihood function. The model construction can be very time consuming [30] (also experienced with LS-OPT). Furthermore, the slight global altering of the Kriging surface due to local updating has also been observed [27]. Some efforts have been made in LS-OPT to reduce the effect of clustering of points.

Support vector regression has also gained significant popularity in the last decade, and literature indicates that it has very good generalization properties, especially in high dimensions. However, the accuracy of SVR depends on the values of the parameters selected. The internal cross-validation technique used in LS-OPT to select these parameter values can be very time consuming. This limits the level of precision with which the best parameters values can be selected.

Reference [27] compares the use of three of the metamodeling techniques for crashworthiness optimization. This paper, which incorporates three case studies in crashworthiness optimization, concludes that while RSM, NN and Kriging were similar in performance, RSM and NN were shown to be the most robust for this application. RBF networks were not available at the time of that study and Kriging has also been improved in the mean time.

24.5. REFERENCES

[1] Daberkow, D.D., Mavris, D.N. An investigation of metamodeling techniques for complex systems design. Symposium on Multidisciplinary Analysis and Design, Atlanta, October 2002.

[2] Redhe, M., Nilsson, L. Using space mapping and surrogate models to optimize vehicle crashworthiness design, AIAA Paper 2002-5607. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, September 4-6, 2003.

[3] Bishop, C.M. Neural Networks for Pattern Recognition. Oxford University Press, 1995.

[4] Hornik, K., Stinchcombe, M., White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Networks, 3, pp. 535-549, 1990.

[5] White, H., Hornik, K., Stinchcombe, M. Universal approximation of an unknown mapping and its derivatives. Artificial Neural Networks: Approximations and Learning Theory, H. White, ed., Oxford, UK: Blackwell, 1992.

[6] Rummelhart, D.E., Hinton, G.E., Williams, R.J. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, Parallel Distributed Processing, Vol. I Foundations, pages 318-362. MIT Press, Cambridge, MA, 1986.

[7] Hoerl, A.H., Kennard, R.W., Ridge regression: Biased estimation for nonorhtogonal problems. Technometrics, 12(3), pp. 55-67, 1970.

[8] Tikhonov, A.N., Arsenin, V.Y., Solutions of Ill-Posed Problems, Winston: Washington, 1977.

[9] Hartman, E.J., Keeler, J.D., Kowalski, J.M. Layered neural networks with Gaussian hidden units as universal approximations. Neural Computation, 2(2), pp. 210-215, 1990.

[10] Park, J., Sandberg, I.W. Approximation and radial basis function networks. Neural Computation, 5(2), pp. 305-316, 1993.

[11] Wahba, G. Spline Models for Observational Data. Volume 59 of Regional Conference Series in Applied Mathematics. SIAM Press, Philadelphia, 1990.

[12] Akaike, H. Statistical predictor identification. Ann.Inst.Statist.Math., 22, pp. 203-217, 1970.

[13] Riedmiller, M., Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In H. Ruspini, editor, Proceedings of the IEEE International Conference on Neural Networks (ICNN), pp. 586 - 591, San Francisco, 1993.

[14] Lawrence, S.C., Lee Giles, Ah Chung Tsoi. What size neural network gives optimal generalization? Convergence Properties of Backpropogation. Technical Report UMIACS-TR-96-22 and CS-TR-3617, University of Maryland, 1996.

[15] Sjöberg, J., Ljung, L. Overtraining, regularization, and searching for minimum in neural networks. Preprints of the 4th IFAC Int. Symp. on Adaptive Systems in Control and Signal Processing, p. 669, July 1992.

[16] MacKay, D. J. C. Bayesian interpolation. Neural Computation, 4(3), pp. 415-447, 1992.

[17] Foresee, F. D., Hagan, M. T. Gauss-Newton approximation to Bayesian regularization. Proceedings of the 1997 International Joint Conference on Neural Networks, pp. 1930-1935, 1997.

[18] Moody, J.E. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. in J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, Advances in Neural Information Processing Systems, 4, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[19] Cohn, D. Neural network exploration using optimal experiment design, Neural Networks, (9)6, pp. 1071-1083, 1996.

[20] Fedorova, N.N. Personal communication, 2004.

[21] Shyy, W., Papila, N. Vaidyanathan, R. and Tucker, P.K. Global design optimization for aerodynamics and rocket propulsion components, Progress in Aerospace Sciences, 37, pp. 59-118, 2001.

[22] Krige, D.G. A Statistical Approach to Some Mine Valuation and Allied Problems on the Witwatersrand. Masters thesis, University of the Witwatersrand, South Africa, 1951.

[23] Simpson, T.W. A Concept Exploration Method for Product Family Design. Ph.D. Thesis, Georgia Institute of Technology, 1998.

[24] Bakker, T.M.D. Design Optimization with Kriging Models. WBBM Report Series 47, Ph.D. thesis, Delft University Press, 2000.

[25] Stander, N., Craig, K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization, Engineering Computations, 19(4), pp. 431-450, 2002.

[26] Stander, N., Reichert, R., Frank, T. 2000: Optimization of nonlinear dynamic problems using successive linear approximations. AIAA Paper 2000-4798.

[27] Stander, N., Roux, W.J., Giger, M., Redhe, M., Fedorova, N. and Haarhoff, J. Crashworthiness optimization in LS-OPT: Case studies in metamodeling and random search techniques. Proceedings of the 4th European LS-DYNA Conference, Ulm, Germany, May 22-23, 2003. (Also www.lstc.com).

[28] Stander, N. Goel, T. Metamodel sensitivity to sequential sampling strategies in crashworthiness design. Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,Victoria, British Columbia, Canada, Sep 10-12, 2008. Submitted.

[29] Xu, Q-S., Liang, Y-Z., Fang, K-T., The effects of different experimental designs on parameter estimation in the kinetics of a reversible chemical reaction. Chemometrics and Intelligent Laboratory Systems, 52, pp. 155-166, 2000.

[30] Jin, R., Chen, W. and Simpson, T.W. Comparative studies of metamodeling techniques under multiple modeling criteria, AIAA Paper, AIAA-2000-4801.

[31] Simpson, T.W., Lin, D.K.J. and Chen, W. Sampling Strategies for Computer Experiments: Design and Analysis. International Journal for Reliability and Applications, Aug. 2001 (Revised Manuscript).

[32] Jin, R., Chen, W. and Sudjianto, A. On sequential sampling for global metamodeling in engineering design, DETC-DAC34092, 2002 ASME Design Automation Conference, Montreal, Canada, September 2002.

[33] Vapnik, V. "Statistical learning theory", 1998.

[34] Smola , A. J., & Schölkopf, B. A tutorial on support vector regression. Statistics and computing, 14(3), 199-222, 2004.

[35] Platt, J. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[36] Geisser, S. Predictive Inference. New York, NY: Chapman and Hall, 1993.

25. Classification Techniques

Classification techniques allow the categorization of design configurations into the specified number of classes. LS-OPT allows binary classification into two classes. In the context of design optimization or reliability assessment, the two classes are typically feasible and infeasible. There are two types of classification – supervised and unsupervised. In the context of simulation-based design, supervised learning is of primary importance, but it may also be combined with unsupervised learning in certain scenarios. The goal of classification techniques is to define a decision boundary in the design space, which has also been referred to as explicit design space decomposition (EDSD) in the literature [1][2][3]. The decision boundary may be used to define a constraint demarcating feasible and infeasible designs or to define a region of interest, e.g. for adaptive sampling [2][4].

25.1. Classification Methodology

Supervised learning is required in order to use classifiers as surrogates for optimization [5] or reliability [6] constraint boundaries. Samples are labeled as feasible or infeasible based on the simulation results, followed by classifier training.

In supervised learning, we are given a set of N samples $\boldsymbol{x}_i$ and their corresponding labels y_i ($i \in [1, N]$). The goal is to construct a decision boundary using this training data that can predict the class of an arbitrary unknown sample $\boldsymbol{x}$. The decision boundary has the form of an analytical equation that can be easily and efficiently be used to approximate optimization or reliability constraints. There are multiple classification algorithms, e.g. k-nearest neighbors (kNN) [7], convex hulls [8], support vector classification (SVC) [9], random forests (RF) [10], neural network classification etc. LS-OPT currently includes SVC.

The basic steps to construct a classifier in the context of design feasibility constraints are listed below and displayed in Figure 24-4.

1. Perform a design of experiments (select samples)
2. Run simulations at these samples
3. Extract simulation results and assign a label +1 or -1 to each sample based on threshold value(s) on the results or using clustering (i.e. unsupervised learning). Note that the labeling may be based on multiple responses simultaneously, e.g. injury criterion, intrusion, energy absorption, mass etc.
4. Construct a classifier using supervised learning (as the labels are assigned already) that can predict the class of any design configuration.

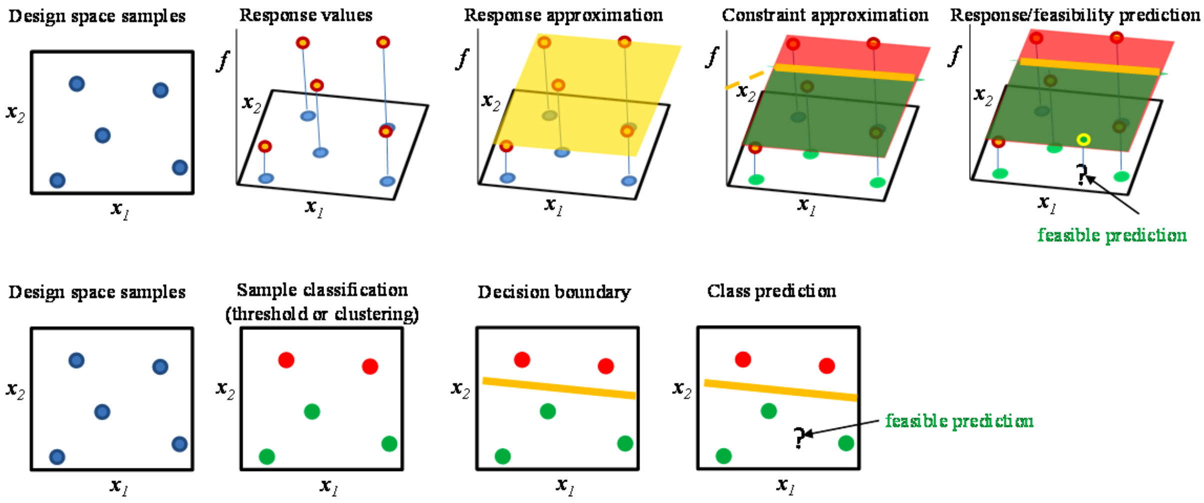


Figure 25-1: Summary of basic classification method (bottom) and comparison to metamodeling (top). The metamdel-based method approximates the responses before projecting the limit state or decision boundary on to the design space, whereas the classification approach directly constructs the decision boundary in the design space.

25.2. Support Vector Classification (SVC)

25.2.1. SVC fundamentals

Support vector classification is subcategory of the class of machine learning techniques known as support vector machines (SVM) [9]. SVM was in fact conceptualized as a classification tool before being extended to regression in the form of support vector regression (SVR). SVC is one of the most popular classification methods, and has been widely used in many fields including computer science and bioinformatics, e.g. for image recognition, gene expression analysis etc. [11][12] The problems in these fields are usually associated with big data. Also, lot of times there is no control over the selection of samples. In the context of simulation-based design optimization and reliability assessment, however, the problems are usually associated with sparse data. It is possible to control the selection of samples for performing simulations, and it is often desirable to run as few simulations as possible without compromising the accuracy of the results. A basic introduction to SVC is provided in this section. More details can be found in several sources [9] [11][13][14].

SVC is applicable to binary classification as well as multi-class classification. However, LS-OPT only supports binary classification as it is particularly interesting in the context of design (e.g. feasible vs infeasible). The two classes are typically labelled as +1 and -1. The basic SVC theory is derived for linearly separable data sets, but it can be extended to nonlinear decision boundaries also.

First, assuming the data to be linearly separable, the equation of an SVM decision boundary in its basic form is given as:

$$s(\boldsymbol{x}) =< \boldsymbol{w}, \boldsymbol{x} > +b = 0 \qquad (25\text{-}1)$$

where $\boldsymbol{x}$ is an arbitrary design, $\boldsymbol{w}$ is a weight vector and b is the bias. Both $\boldsymbol{w}$ and b are determined during the training process, using an optimization method referred to as sequential minimal optimization (SMO) [13]. In a two-dimensional space, i.e. for two variables, this is the equation of a straight line while in a multidimensional space it represents a hyperplane. Figure 25-2 shows a case of linearly separable data. However, there are infinite possible linear separators for this data. Therefore some optimality criterion is required to select the best separator.

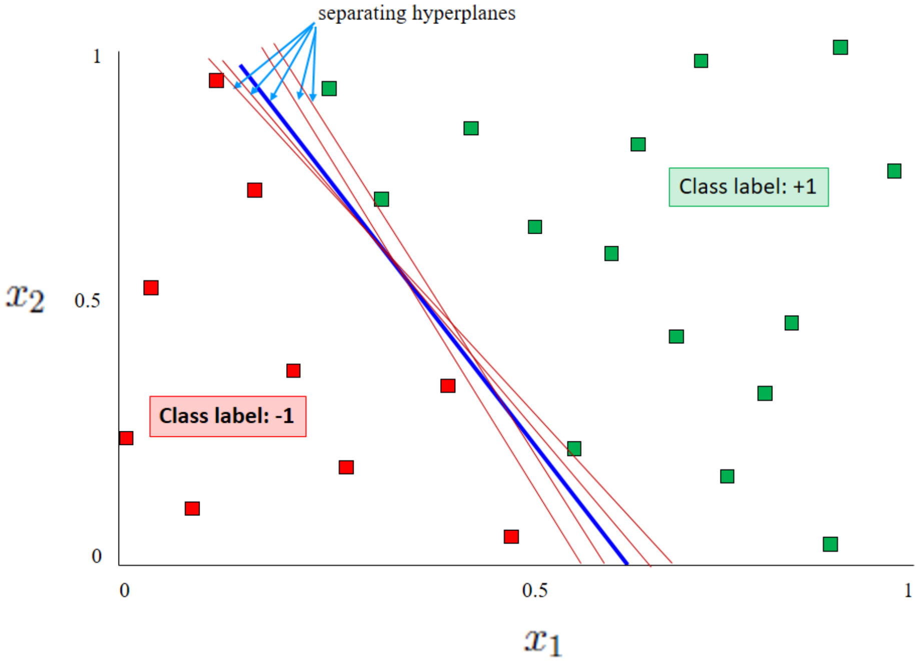


Figure 25-2: Linearly separable binary (red vs green) data. An infinite number of linear separators are possible, some of which are shown using the brown lines. The blue line represents the eventual optimal separator (see Figure 25-3).

Without loss of generality, the +1 and -1 isocontours of $s(\boldsymbol{x})$ play an important role in defining the optimality criterion. The perpendicular distance between $s(\boldsymbol{x}) = +1$ and $s(\boldsymbol{x}) = -1$ is referred to as the margin and a constraint is applied such that there is no training sample within the margin. The optimal SVM decision boundary is obtained by maximizing the margin while ensuring the absence of samples in the space between the hyperplanes $s(\boldsymbol{x}) = +1$ and $s(\boldsymbol{x}) = -1$, also referred to as the support hyperplanes. The training samples that lie on the support hyperplanes are known as support vectors. These are the only samples that affect the equation of the decision boundary. The margin is equal to $\frac{2}{\|\boldsymbol{w}\|}$, resulting in the following optimization problem.

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{25-2}$$

$$s.t. \left|< \boldsymbol{w}, \boldsymbol{x}^i > +b\right| \geq 1 \qquad \forall\, i \in [1, N]$$

The Lagrangian for this optimization is

$$\Phi(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \text{-} \sum_{i=1}^{N} \alpha^i \left\{y^i(< \boldsymbol{w}, \boldsymbol{x}^i > +b) - 1\right\} \tag{25-3}$$

where $\propto^i$ is the non-negative Lagrange multiplier corresponding to the ith sample (i.e. ith constraint in Equation (25-2)). The dual problem is:

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w},b} \Phi(\boldsymbol{w}, b, \boldsymbol{\alpha}) \tag{25-4}$$

Applying stationary conditions on the Lagrangian with respect to ***w*** and *b* gives:

$$\sum_{i=1}^{N} \propto^i y^i = 0$$

$$\boldsymbol{w} = \sum_{i=1}^{N} \propto^i y^i \boldsymbol{x}^i$$

As a result, the dual problem can be re-written as:

$$\max_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j < \boldsymbol{x}^i, \boldsymbol{x}^j > - \sum_{i=1}^{N} \propto^i \left\{ y^i (< \sum_{j=1}^{N} \propto^j y^j \boldsymbol{x}^j, \boldsymbol{x}^i > + b) - 1 \right\}$$

$$\Rightarrow \max_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j < \boldsymbol{x}^i, \boldsymbol{x}^j > - b \sum_{j=1}^{N} \propto^i y^i - \sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j < \boldsymbol{x}^i, \boldsymbol{x}^j > + \sum_{j=1}^{N} \propto^i$$

$$\Rightarrow \max_{\boldsymbol{\alpha}} \ -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j < \boldsymbol{x}^i, \boldsymbol{x}^j > + \sum_{j=1}^{N} \propto^i$$

Thus, reversing the sign, the solution can be written as:

$$\arg\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j < \boldsymbol{x}^i, \boldsymbol{x}^j > - \sum_{j=1}^{N} \propto^i$$

$$s.t. \quad \propto^i \geq 0$$

$$\sum_{i=1}^{N} \propto^i y^i = 0 \qquad \forall\, i \in [1, N]$$

The Lagrange multipliers $\propto$ are typically solved using SMO. Once the $\propto^i$ are known, the optimal ***w*** and *b* are calculated as:

$$\boldsymbol{w}^* = \sum_{i=1}^{N} \propto^i y^i \boldsymbol{x}^i$$

$$b^* = -\frac{1}{2} < \boldsymbol{w}^*, \boldsymbol{x}_{SV+} + \boldsymbol{x}_{SV-} >$$

where $\boldsymbol{x}_{SV+}$ and $\boldsymbol{x}_{SV-}$ are support vectors with $\propto > 0$, $y_{SV+} = +1$ and $y_{SV-} = -1$. The classifier boundary is then given as:

$$s(\boldsymbol{x}) =< \boldsymbol{w}^*, \boldsymbol{x} > +b^* = \sum_{i=1}^{N} \propto^{*i} y^i < \boldsymbol{x^i}, \boldsymbol{x} > +b^* = 0 \qquad (25\text{-}5)$$

The class of an arbitrary sample $\boldsymbol{x}$ can be determined by calculating the sign of $s(\boldsymbol{x})$, i.e. if $s(\boldsymbol{x} > 0)$, the predicted class is +1 and if $s(\boldsymbol{x} < 0)$, the predicted class is -1.

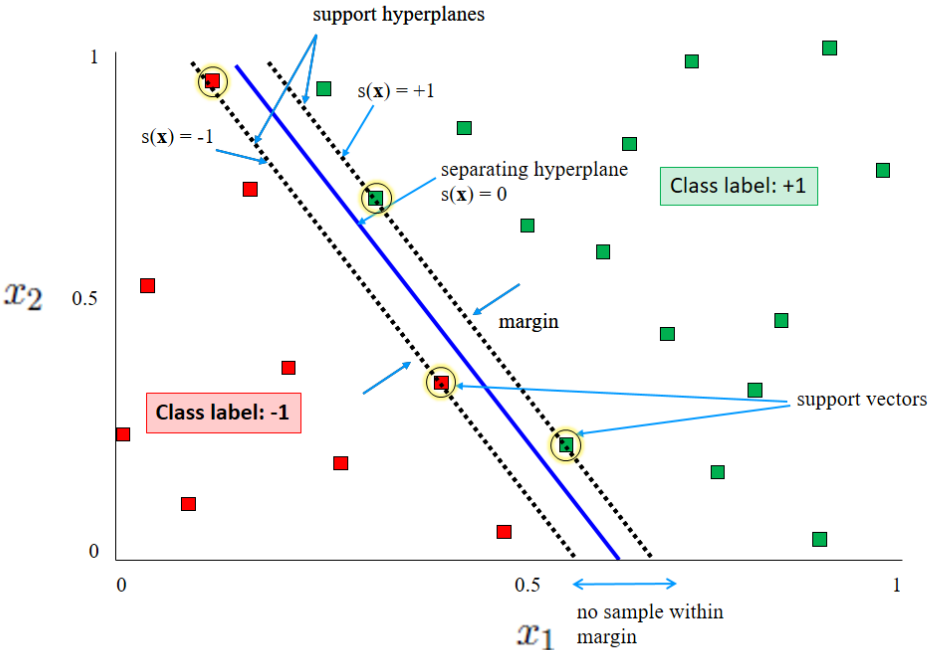


Figure 25-3: SVM boundary separating the two classes of a linearly separable binary (red vs green) data set. The optimal SVM boundary (blue line) is obtained by maximizing the margin, i.e. the perpendicular distance between the two support hyperplanes (dotted lines). The samples lying on the support hyperplanes are known as support vectors (circled).

In general the data may not be linearly separable, e.g. in Figure 25-4. Therefore the above methodology of obtaining the optimal SVM decision boundary cannot be used directly. As the constraint of not having any sample in the margin cannot be satisfied, slack variables are introduced along with a misclassification cost. Thus, the new reformulated problem has an additional goal of minimizing the cost weighted misclassification error.

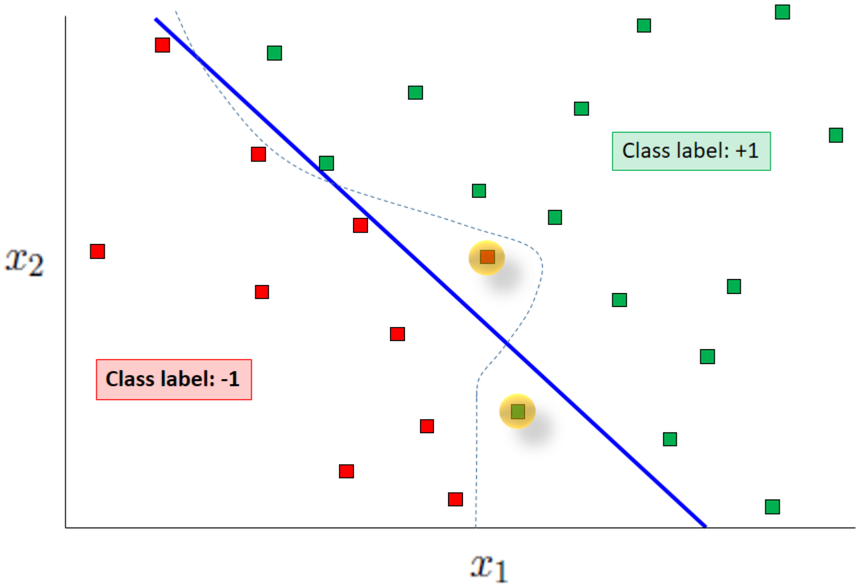


Figure 25-4: Nonlinear data set that cannot be separated using a linear boundary. The blue linear separator leads to the misclassification of the circled samples. A nonlinear separator is possible here (dotted) that will not misclassify any samples.

The solution of the problem is similar to the linearly separable case, except for the introduction of upper bounds on the Lagrange multipliers. Details of the solution can be found in [15]. The solution can be written as:

$$\arg\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j < \boldsymbol{x}^i, \boldsymbol{x}^j > - \sum_{j=1}^{N} \propto^i$$

$$s.t. \quad 0 \le \propto^i \le C$$

$$\sum_{i=1}^{N} \propto^i y^i = 0 \qquad \forall\, i \in [1, N]$$

where C is the misclassification cost specified before the optimization. The equation of the SVM classifier remains unchanged, and it should be noted that this is still a linear decision boundary. In general, however, there is no reason why a decision boundary must always be linear with respect to the variables. Therefore, to extend SVC to problems where a nonlinear boundary is appropriate, a mapping is introduced. In the mapped high dimensional space, known as the feature space, the SVM boundary is linear. However, it can be nonlinear with respect to the original input variables.

The equation of the SVM boundary in the feature space can be derived in the same way as earlier, but that would make the process more complicated due to the high dimensionality. Instead, it is actually possible to derive the SVM equation easily in the input space. This is made possible because only the inner product of the sample position vectors appears in the final equation. Instead of carrying out this inner product in the feature space, it can simply be replaced with a positive semi-definite kernel function defined in terms of the input variables. This is known as the kernel trick. The equation of the SVM boundary becomes:

$$s(\boldsymbol{x}) = \sum_{i=1}^{N} \propto^{*i} y^i K(\boldsymbol{x^i}, \boldsymbol{x}) + b^* = 0 \qquad (25\text{-}6)$$

where $y_{\mathrm{i}} = \pm 1$ (e.g. red vs green) is the class label, α_{i} is the Lagrange multiplier for i^{th} sample, K is the kernel function and b is the bias. The Lagrange multipliers are solved as:

$$\arg\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto^i \propto^j y^i y^j K(\boldsymbol{x^i}, \boldsymbol{x}) - \sum_{j=1}^{N} \propto^i$$

$$s.t. \quad 0 \le \propto^i \le C$$

$$\sum_{i=1}^{N} \propto^i y^i = 0 \qquad \forall\, i \in [1, N]$$

$$< \boldsymbol{w}^*, \boldsymbol{x} >= \sum_{i=1}^{N} \propto^i y^i K(\boldsymbol{x}^i, \boldsymbol{x})$$

$$b^* = -\frac{1}{2}\sum_{i=1}^{N} \propto^i y^i K(\boldsymbol{x}^i, \boldsymbol{x}) < \boldsymbol{w}^*, \boldsymbol{x}_{SV+} + \boldsymbol{x}_{SV-} >$$

The kernel function must satisfy Mercer's condition [14][15]. Some of the possible kernels are polynomial, exponential radial basis function, Gaussian radial basis function, hyperbolic tangent etc. Two options are available in LS-OPT for the kernel:

1. Gaussian RBF
2. Polynomial

The polynomial kernel is given as:

$$K(\mathbf{x}_i, \mathbf{x}) = \left(1 + \langle \mathbf{x}_i, \mathbf{x} \rangle\right)^m \qquad (25\text{-}7)$$

where m is the order of the polynomial that is internally optimized in LS-OPT.

The Gaussian RBF kernel is given as:

$$K(\mathbf{x}_i, \mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}} \qquad (25\text{-}8)$$

Here, σ is the width parameter or spread of the kernel that is internally optimized in LS-OPT using cross-validation. In addition to the kernel parameters, the SVR parameters C and ε are also internally optimized in LS-OPT during cross-validation.

25.2.2. SVC parameter selection

There are different options in LS-OPT to set some of the SVC training parameters. The optimal decision boundary depends on the misclassification cost C as well as on kernel parameters. Additionally, a tolerance ε introduced to quantify the strictness of the constraint satisfaction can also affect the boundary. In LS-OPT the user can either specify fixed values or let the software tune these parameters. The parameter tuning has 3 types of error metric options [17]:

Error rate:

$$E = 1 - \frac{TP + TN}{N_{+} + N_{-}}$$

Weighted error rate:

$$E = 1 - \frac{1}{2}\left(\frac{TP}{N_{+}} + \frac{TN}{N_{-}}\right)$$

ROC curve:

$$E = 1 - \frac{1}{2}\sum_{i=1}^{n_s - 1}\left(TPR^{(i)} + TPR^{(i+1)}\right)\left(FPR^{(i+1)} + FPR^{(i)}\right)$$

In the above equations, *TP* and *TN* refer to the number of true positives and true negatives, N_{+} and N_{-} refer to the number of positive and negative samples, and *TPR* and *FPR* refer to the true positive ratio and the false positive ratio. In the ROC curve method, the error is calculated based on several levels of the SVM value as cutoff and n_s represents the number of such levels. In order to tune the parameters, a grid of possible parameter values is defined and the parameter combination with the lowest error is selected.

25.3. Class labeling

25.3.1. Threshold value-based class labeling

In most engineering applications, the design constraints have specific bounds provided by the user. For instance, the maximum stress in a structure under loading should not exceed certain limit specified based on the material. For such problems, class labeling is straightforward. Once the simulation is run at a particular design, the results are compared to the limit(s) and a class label is assigned depending on whether the design satisfies the limiting value(s).

25.3.2. Unsupervised learning

The purpose of unsupervised learning, in the context of classification, is to assign categories or class labels to the unlabeled data samples. This is done by finding patterns in the dataset, e.g. using clustering. K-means clustering [18], with k = 2, is used in LS-OPT for this purpose currently. This is particularly useful when the response is discontinuous and there is no prior knowledge of a threshold to classify the data (Figure 25-5).

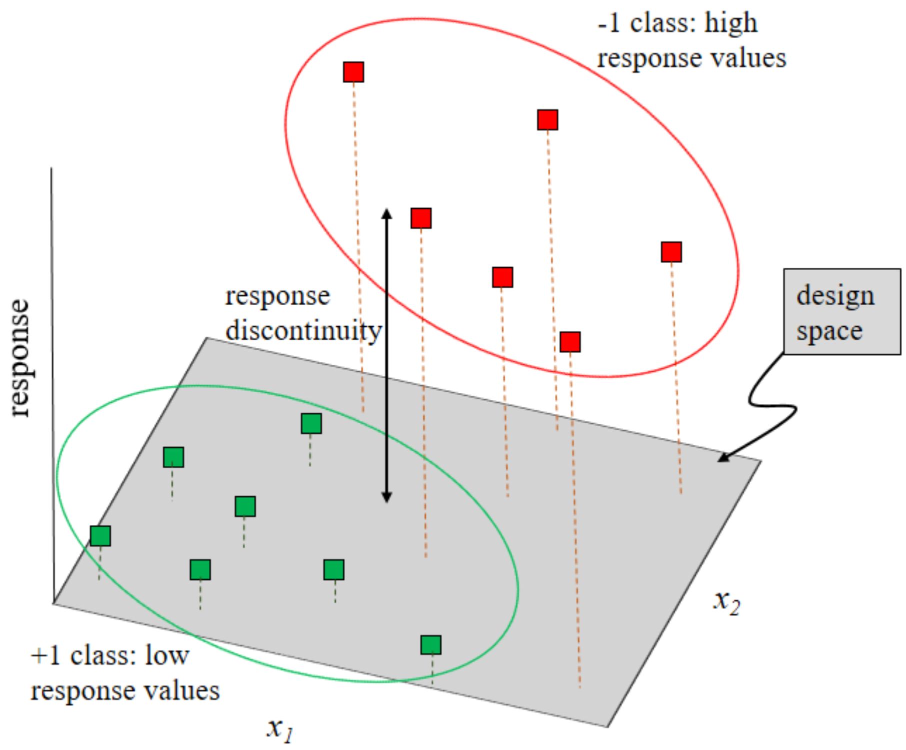


Figure 25-5: Labeling of samples using clustering, an unsupervised learning method to identify patterns in the data. It is especially suited for discontinuous responses due to the lack of a grey area.

The k-means clustering algorithm partitions N data points into k sets or clusters. The j^{th} set S_j contains n_j samples and the only requirement for the set size is that each set must contain at least one sample. The cumulative sum of square distances within the sets is minimized:

$$\min \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j \right\|^2 \qquad (25\text{-}9)$$

25.4. Adaptive Sampling Using Classifiers

The main idea of classification-based methods is to decompose the design space into distinct subregions, which presents a straightforward way to define design constraint boundaries separating feasible and infeasible domains. The same idea is applicable in the context of sample selection; a classifier can be used to define the regions that should be sampled and that should not be sampled. Many unnecessary sample evaluations can be avoided in this manner. The idea is applicable in reliability assessment, global constraint construction as well as in single/multi-objective optimization. EDSD sampling constraints facilitate the use of classifiers in defining the sampling domain.

25.4.1. Adaptive Sampling for Reliability Assessment

Accuracy of the limit state boundary is of paramount importance in the context of reliability assessment. The samples in the vicinity of the decision boundary play an important role in determining its accuracy. Therefore, adding more samples in such regions is expected to obtain important information. In the context of SVM classifiers, the margin $-1 \leq s(\boldsymbol{x}) \leq +1$ around the boundary does not have any samples by construction, and thus, is an automatic choice for sampling. This however does not imply that those regions are the only important ones, as the initial SVM boundary may be inaccurate. Therefore, sampling outside the margin is also necessary. This is the idea behind the advanced EDSD options “First”, “Last” and “Gap” (Section 9.7), which allow some space-filling samples at each iteration to be selected anywhere in the entire design space. Additionally, although the margin $-1 \leq s(\boldsymbol{x}) \leq +1$ is known to be empty and sampling there will therefore change the boundary, it may not necessarily lead to the fastest convergence always. To avoid an arbitrary choice of the sampling bounds, it can be useful to randomize them using the advanced EDSD option “Random” (default is on).

Figure 25-6 shows an initial classifier boundary, and the boundary and sampling after 7 iterations while using a classifier-based sampling constraint. The total number of samples per iteration is 20 and the sampling constraint specification is <edsdconstraint src="x_cls_freq_mac" marginlower="-2" marginupper="2" lastsampleadaptive="16"/>. This implies that 80% of the samples are selected in the vicinity $-\varepsilon \leq s(\boldsymbol{x}) \leq +\varepsilon$ of the classifier boundary. The random value of ε for each of the 16 constrained samples is lower than or equal to 2. It can be seen in the figure that most of the samples are near the boundary and the sparsely populated regions are the ones that do not have much effect on its estimation.

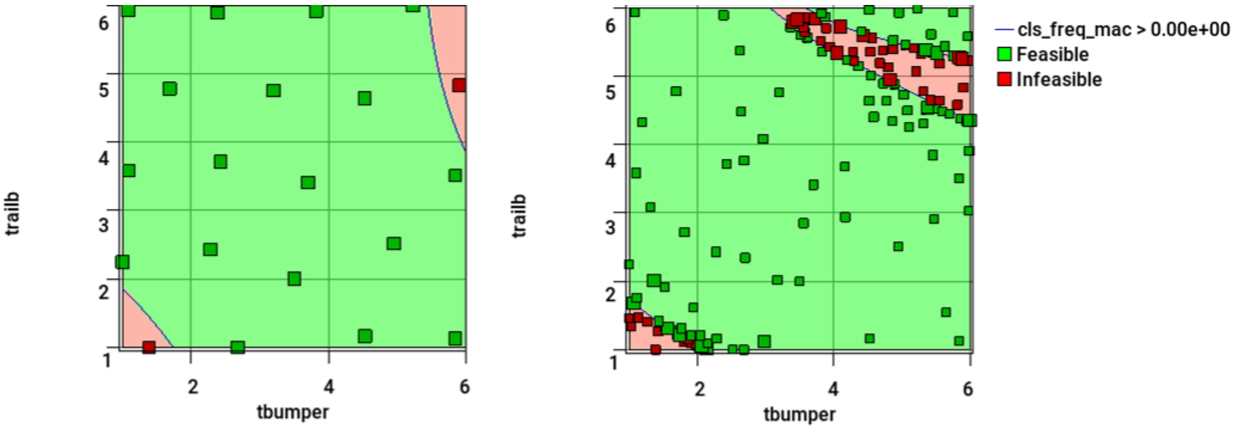


Figure 25-6: Adaptive sampling in the vicinity of the decision boundary using an EDSD sampling constraint. Left plot shows the initial boundary and the right plot shows the samples and the boundary after 7 iterations.

25.4.2. Adaptive Sampling for Optimization

Classifier sampling constraints can be used in the context of optimization to either select samples near the boundary or in the feasible domain. Additionally, a classifier can be used to define the Pareto optimal domain in the context of multi-objective optimization [19][20]. An EDSD constraint for such scenarios can be defined as $0 \leq s(\boldsymbol{x})$. An example of sampling in the feasible domain is shown in Figure 25-7.

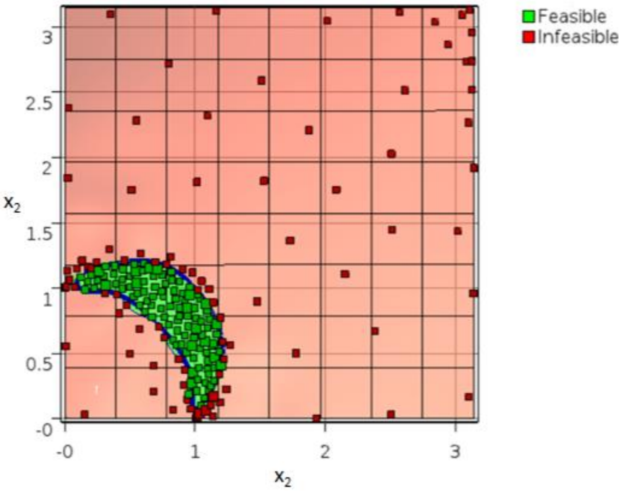


Figure 25-7: Adaptive sampling in the feasible domain.

25.5. Concluding remarks: applications and outlook

One of the obvious applications of classifiers is when system responses are discontinuous or binary. Examples of binary problems include those with pass/fail information, hidden constraints where the responses cannot be evaluated for all samples, combination of multiple data sources that provide similar qualitative information but lack quantitative data or do not agree quantitatively (e.g. combining simulations, experiments and experience) etc. Examples of response discontinuities include problems with geometric nonlinearities, e.g. buckling, impact etc.

Additionally, classifiers can be used to reduce the cost of simulations by avoiding the evaluation of all the load cases during multi-disciplinary analysis and optimization, as they may not be required to determine design feasibility. Such computation features however need some user-defined scripting in LS-OPT.

It should be noted that classifiers use -1^{th} order information to approximate the decision boundaries, and as such do not use all the available information, e.g. the 0 order response values used by metamodels. Therefore, it is natural that for many problems metamodels may be more efficient, especially for smooth problems. On the contrary, a classifier uses the information about the feasibility criteria during training, which is not used directly by a metamodel. As a result a classifier can be more accurate in the vicinity of the decision boundary. The usefulness and efficiency of classifiers can be apparent for some types of problems like the ones mentioned above. Another application of classifiers is in the field of MOO, which will be available in future LS-OPT versions.

25.6. REFERENCES

[1] Basudhar, A., Missoum, S. and Sanchez, A.H., 2008. Limit state function identification using support vector machines for discontinuous responses and disjoint failure domains. *Probabilistic Engineering Mechanics*, *23*(1), pp.1-11.

[2] Basudhar, A. and Missoum, S., 2008. Adaptive explicit decision functions for probabilistic design and optimization using support vector machines. *Computers & Structures*, *86*(19-20), pp.1904-1917.

[3] Basudhar, A. 2011. Computational optimal design and uncertainty quantification of complex systems using explicit decision boundaries. PhD Dissertation, The University of Arizona.

[4] Basudhar, A. and Missoum, S., 2010. An improved adaptive sampling scheme for the construction of explicit boundaries. *Structural and Multidisciplinary Optimization*, *42*(4), pp.517-529.

[5] Basudhar, A., Dribusch, C., Lacaze, S. and Missoum, S., 2012. Constrained efficient global optimization with support vector machines. *Structural and Multidisciplinary Optimization*, *46*(2), pp.201-221.

[6] Hurtado, J.E., 2007. Filtered importance sampling with support vector margin: a powerful method for structural reliability analysis. *Structural Safety*, *29*(1), pp.2-15.

[7] Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), pp.21-27.

[8] Barber, C.B., Dobkin, D.P. and Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, *22*(4), pp.469-483.

[9] Vapnik, V., 1998. *Statistical learning theory* (Vol. 3). Wiley, New York.

[10] Ho, T.K., 1995, August. Random decision forests. In *proceedings of the third international conference on Document analysis and recognition* (Vol. 1, pp. 278-282). IEEE*, 1995.*

[11] Cristianini, N. and Shawe-Taylor, J., 2000. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

[12] Brown MPS, Grundy WN, Lin D, et al. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America*. 2000; 97(1):262-267.

[13] Platt, J., 1998. Sequential minimal optimization: A fast algorithm for training support vector machines.

[14] Burges, C.J., 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), pp.121-167.

[15] Gunn, S.R., 1998. Support vector machines for classification and regression. *ISIS technical report*, *14*(1), pp.5-16.

[16] Vapnik, V. "Statistical learning theory", 1998.

[17] Jiang, P., Missoum, S. and Chen, Z., 2014. Optimal SVM parameter selection for non-separable and unbalanced datasets. *Structural and Multidisciplinary Optimization*, *50*(4), pp.523-535.

[18] Hartigan, J.A. and Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), pp.100-108.

[19] Basudhar, A., 2015. Multi-objective optimization using adaptive explicit non-dominated region sampling. In *11th world congress on structural and multidisciplinary optimization*.

[20] Basudhar A, inventor; Livermore Software Tech Corp, assignee. Multi-objective design optimization using adaptive classification. United States patent US 9,852,235. 2017 Dec 26.

26. Optimization

26.1. Theory of optimization

Optimization can be defined as a procedure for “achieving the best outcome of a given operation while satisfying certain restrictions” [1]. This objective has always been central to the design process, but is now assuming greater significance than ever because of the maturity of mathematical and computational tools available for design.

Mathematical and engineering optimization literature usually presents the above phrase in a standard form as

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{subject to:} \\ &\quad g_j(\mathbf{x}) \le 0;\ j = 1,2,\ldots,m \\ &\quad h_k(\mathbf{x}) = 0;\ k = 1,2,\ldots,l. \end{aligned} \tag{26-1}$$

where *f*, *g* and *h* are functions of independent variables x_1, x_2, x_3, …, x_n. The function *f*, referred to as the cost or objective function, identifies the quantity to be minimized or maximized. The functions *g* and *h* are constraint functions representing the design restrictions. The variables collectively described by the vector **x** are often referred to as design variables or design parameters.

The two sets of functions g_j and h_k define the constraints of the problem. The equality constraints do not appear in any further formulations presented here because algorithmically each equality constraint can be represented by two inequality constraints in which the upper and lower bounds are set to the same number, e.g.

$$h_k(\mathbf{x}) = 0 \approx 0 \le h_k(\mathbf{x}) \le 0 \tag{26-2}$$

Equations (26-1) then become

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{subject to:} \\ &\quad g_j(\mathbf{x}) \le 0;\ j = 1,2,\ldots,m. \end{aligned} \tag{26-3}$$

The necessary conditions for the solution x^* to Eq. (26-3) are the Karush-Kuhn-Tucker optimality conditions:

$$\begin{aligned} &\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^T \nabla g(\mathbf{x}^*) = 0 \\ &\boldsymbol{\lambda}^T g(\mathbf{x}^*) = 0 \\ &g(\mathbf{x}^*) \le 0 \\ &\boldsymbol{\lambda} \ge 0. \end{aligned} \tag{26-4}$$

These conditions are derived by differentiating the Lagrangian function of the constrained minimization problem

$$L(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T g(\mathbf{x}). \tag{26-5}$$

and applying the conditions

$$\nabla^T f \partial \mathbf{x}^* \ge 0 \quad \text{(optimality)} \tag{26-6}$$

and

$$\nabla^T \bar{\mathbf{g}} \partial \mathbf{x}^* \le 0 \quad \text{(feasibility)} \tag{26-7}$$

to a perturbation $\partial \mathbf{x}^*$. λ_j represents the Lagrange multiplier which may be nonzero only if the corresponding constraint is active, i.e. $g_j(\mathbf{x}^*) = 0$. For $\mathbf{x}^*$ to be a local constrained minimum, the Hessian of the Lagrangian function, $\nabla^2 f(\mathbf{x}^*) + \lambda^T \nabla^2 \bar{\mathbf{g}}(\mathbf{x}^*)$ on the subspace tangent to the active constraint $\bar{\mathbf{g}}$ must be positive definite at $\mathbf{x}^*$.

These conditions are not used explicitly in LS-OPT and are not tested at optima. They are more of theoretical interest in this manual, although the user should be aware that some optimization algorithms are based on these conditions.

26.2. Normalization of constraints and variables

It is a good idea to eliminate large variations in the magnitudes of design variables and constraints by normalization.

Constraints. In LS-OPT, a typical constraint is formulated as follows:

$$L_j \le g_j(\mathbf{x}) \le U_j;\ j = 1, 2, ..., m. \tag{26-8}$$

This inequality represents two constraints:

$$\begin{aligned} &L_j \le g_j(\mathbf{x});\ j = 1, 2, ..., m. \\ &g_j(\mathbf{x}) \le U_j;\ j = 1, 2, ..., m. \end{aligned} \tag{26-9}$$

which, when normalized, become:

$$1 \le \frac{g_j(\mathbf{x})}{L_j};\ j = 1, 2, \ldots, m.$$
$$\frac{g_j(\mathbf{x})}{U_j} \le 1;\ j = 1, 2, \ldots, m. \qquad (26\text{-}10)$$

A feature is provided in the GUI to automatically switch on constraint scaling using a single check box. As in Equation (26-10), the values of the bounds, L_j and U_j are used as default scale factors, but can be selected.

Variables. The design variables have been normalized internally by scaling the design space [$\boldsymbol{x}_L$; $\boldsymbol{x}_U$] to [0;1], where $\boldsymbol{x}_L$ is the lower and $\boldsymbol{x}_U$ the upper bound. The formula

$$\xi_i = \frac{x_i - x_{iL}}{x_{iU} - x_{iL}}. \qquad (26\text{-}11)$$

is used to transform each variable x_i to a normalized variable, ξ_i .

26.3. Gradient computation and the solution of optimization problems

Solving the optimization problem requires an optimization algorithm. The list of optimization methods is long and the various algorithms are not discussed in any detail here. For this purpose, the reader is referred to the texts on optimization, e.g. [1] or [2]. It should however be mentioned that the Sequential Quadratic Programming method is probably the most popular algorithm for constrained optimization and is considered to be a state-of-the-art approach for structural optimization [3], [4]. In LS-OPT, the subproblem is optimized by an accurate and robust gradient-based algorithm: the dynamic leap-frog method [5]. Both these algorithms and most others have in common that they are based on first order formulations, i.e. they require the first derivatives of the component functions

$$df / dx_i \text{ and } dg_j / dx_i$$

to construct the local approximations. These gradients can be computed either analytically or numerically. In order for gradient-based algorithms such as SQP to converge, the functions must be continuous with continuous first derivatives.

Analytical differentiation requires the formulation and implementation of derivatives with respect to the design variables in the simulation code. Because of the complexity of this task, analytical gradients (also known as design sensitivities) are mostly not readily available.

Numerical differentiation is typically based on forward difference methods that require the evaluation of *n* perturbed designs in addition to the current design. This is simple to implement but is expensive and hazardous because of the presence of round-off error. As a result, it is difficult to choose the size of the intervals of the design variables, without risking spurious derivatives (the interval is too small) or inaccuracy (the interval is too large). Some discussion on the topic is presented in Reference [1].

As a result, gradient-based methods are typically only used where the simulations provide smooth responses, such as linear structural analysis, certain types of nonlinear analysis or smooth metamodels (mathematical approximations) of the actual response.

In non-linear dynamic analysis such as the analysis of impact or metal-forming, the derivatives of the response functions are mostly severely discontinuous. This is mainly due to the presence of friction and contact. The response (and therefore the sensitivities) may also be highly nonlinear due to the chaotic nature of impact phenomena and therefore the gradients may not reveal much of the overall behavior. Furthermore, the accuracy of numerical sensitivity analysis may also be adversely affected by round-off error. Analytical sensitivity analysis for friction and contact problems is a subject of current research.

It is mainly for the above reasons that researchers have resorted to global approximation methods (also called metamodels) for smoothing the design response. The art and science of developing design approximations has been a popular theme in design optimization research for decades (for a review of the various approaches, see e.g. Reference [6] by Barthelemy). Barthelemy categorizes two main global approximation methods, namely response surface methodology [7] and neural networks [8]. Since then other approximations such as Radial Basis Function networks and Kriging have also become popular metamodels.

In the present implementation, the gradient vectors of general composites based on mathematical expressions of the basic response surfaces are computed using numerical differentiation. A default interval of 1/1000 of the size of the design space is used in the forward difference method.

26.4. Optimization methods

The two basic optimization branches employed in LS-OPT are Metamodel-based optimization and Direct optimization. Metamodel-based optimization is used to create and optimize an approximate model (metamodel) of the design instead of optimizing the design through direct simulation. The metamodel is thus created as a simple and inexpensive surrogate of the actual design. Once the metamodel is created, it can be used to find the optimum or, in the case of multiple objectives, the Pareto Optimal Front. Metamodeling techniques are discussed in Chapter 24.

The nature and capacity of the simulation environment as well as the purpose of the optimization effort typically dictate the strategies for metamodel-based optimization. The strategies depend mostly on whether the user wants to build a metamodel that can be used for global exploration or whether she is only interested in finding an optimal set of parameters. An important criterion for choosing a strategy is also whether the user wants to build the metamodel and solve the problem iteratively or whether he has a "simulation budget" i.e., a certain number of simulations that he wants to use as effectively as possible to build a metamodel and obtain as much information about the design as possible.

26.5. Strategies for metamodel-based optimization

There are three recommended strategies for automating the metamodel-based optimization procedure. These strategies apply to the tasks: Metamodel-based Optimization and RBDO. The setup for each strategy is explained in detail in Section 4.8.

26.5.1. Single stage

In this approach, the experimental design for choosing the sampling points is done only once. A typical application would be to choose a large number of points (as much as can be afforded) to build metamodels such as, RBF networks using the Space Filling sampling method. This is probably the best way of sampling for Space Filling since the Space Filling algorithm positions all the points in a single cycle.

26.5.2. Sequential strategy

In this approach, sampling is done sequentially. A small number of points is chosen for each iteration and multiple iterations are requested. The approach has the advantage that the iterative process can be stopped as soon as the metamodels or optimum points have sufficient accuracy. It was demonstrated in Reference [16] that, for Space Filling, the Sequential approach had similar accuracy compared to the Single Stage approach, i.e. 10×30 points added sequentially is almost as good as 300 points. Therefore both the Single Stage and Sequential Methods are good for design exploration using a surrogate model. For instance when constructing a Pareto Optimal Front, the use of a Single Stage or Sequential strategy is recommended in lieu of a Sequential strategy with domain reduction (see Section 26.5.3).

Both the previous strategies work better with metamodels other than polynomials because of the flexibility of metamodels such as neural networks to adjust to an arbitrary number of points.

26.5.3. Sequential strategy with domain reduction

This approach is the same as that in 26.5.2 but in each iteration the domain reduction strategy is used to reduce the size of the subregion. During a particular iteration, the subregion is used to bind the positions of new points. This method is typically the only one suitable for polynomials. There are two approaches to Sequential Domain Reduction strategies. The first is global and the second, local.

Sequential adaptive metamodeling (SAM)

As for the sequential strategy in Section 26.5.2 *without* domain reduction, sequential adaptive sampling is done and the metamodel constructed using all available points, including those belonging to previous iterations. The difference is that in this case, the size of the subregion is adjusted (usually reduced) for each iteration (see Section 26.6). This method is good for converging to an optimum and *moderately* good for constructing global approximations for design exploration such as a Pareto Optimal front. The user should however expect to have poorer metamodel accuracy at design locations remote from the current optimum.

Sequential response surface method (SRSM)

SRSM is the original LS-OPT automation strategy of Section 26.6 and allows the building of a new response surface (typically linear polynomial) in each iteration. The size of the subregion is adjusted for each iteration (see Section 26.6). Points belonging to previous iterations are ignored. This method is only suitable for convergence to an optimum and should not be used to construct a Pareto optimal front or do any other type of design exploration. Therefore the method is ideal for system identification (see Chapter 28).

26.5.4. Efficient Global Optimization

This approach is the same as that in 26.5.2, but the first points for all the iterations are selected by maximizing an expected improvement (EI) function. The EI not only considers the predicted response value, but also its variance. It strikes a balance between local and global search, and often has the ability to find the approximate region containing the global optimum using only a few iterations. In the current implementation, the other samples are selected using a space-filling algorithm. Therefore, the algorithm primarily focuses on global search. In order to ensure higher accuracy at the optimum, EGO may be followed by SAM. It should be noted that EGO is not suitable for the construction of a Pareto front, and for single objective optimization problems also it is typically used for up to 10-15 variables.

26.5.5. How do I choose an appropriate strategy for metamodel-based optimization?

Selecting the *Strategy* is the main selection for metamodel-based optimization. In the GUI, the three main choices, namely Single stage, Sequential or Sequential with Domain Reduction can be selected. If the Pareto Frontier option has been selected for a multi-objective optimization, Domain Reduction is automatically grayed out so is no longer an option. Hence few choices remain.

In the case of a single objective the user might want to change the design formulation or parameters such as constraint bounds *after* the run. In this case, Sequential (no Domain Reduction) should be used.

The Single Stage approach is intended for users who want to create a globally explorable model and have a fixed budget (e.g. 1000 runs). A very similar globally explorable design model can also be created with the Sequential strategy (without Domain Reduction) but an advantage of Sequential methods is that one can set stopping tolerances. These allow the accuracy of the design model to be maximized if sufficient computing resources are available.

Changing the strategy is flexible, so if, for instance, the user completes a Single Stage run and then decides that a refinement of the design model is needed, he can switch to Sequential and restart. Once the strategy is selected the remaining options are defaulted. For Sequential, the only remaining strategy settings are the convergence tolerances and limits.

26.6. Sequential response surface method (SRSM)

The purpose of the SRSM method is to allow convergence of the single-objective solution to a prescribed tolerance.

The SRSM method [15] uses a region of interest, a subspace of the design space, to determine an approximate optimum. A range is chosen for each variable to determine its initial size. A new region of interest centers on each successive optimum. Progress is made by moving the center of the region of interest as well as reducing its size. Figure 26-1 shows the possible adaptation of the subregion.

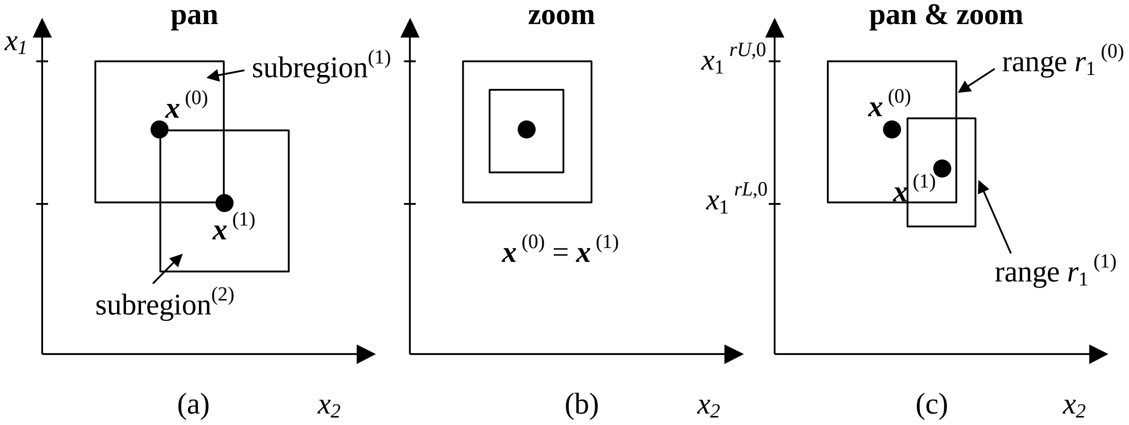


Figure 26-1: Adaptation of subregion in SRSM: (a) pure panning, (b) pure zooming and (c) a combination of panning and zooming

The starting point $\mathbf{x}^{(0)}$ will form the center point of the first region of interest. The lower and upper bounds $(x_i^{rL,0}, x_i^{rR,0})$ of the initial subregion are calculated using the specified initial range value $r_i^{(0)}$ so that

$$x_i^{rL,0} = x_i^{(0)} - 0.5r_i^{(0)} \; and \; x_i^{rU,0} = x_i^{(0)} + 0.5r_i^{(0)}; \;\; i = 1,...,n \tag{26-12}$$

where *n* is the number of design variables. The modification of the ranges on the variables for the next iteration depends on the oscillatory nature of the solution and the accuracy of the current optimum.

Oscillation: A *contraction parameter* γ is firstly determined based on whether the current and previous designs $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k-1)}$ are on the opposite or the same side of the region of interest. Thus an *oscillation indicator c* may be determined in iteration *k* as

$$c_i^{(k)} = d_i^{(k)} d_i^{(k-1)} \tag{26-13}$$

where

$$d_i^{(k)} = 2\Delta x_i^{(k)} / r_i^{(k)}; \;\; \Delta x_i^{(k)} = x_i^{(k)} - x_i^{(k-1)}; d_i^{(k)} \in [-1;1] \tag{26-14}$$

The oscillation indicator (purposely omitting indices *i* and *k*) is normalized as $\hat{c}$ where

$$\hat{c} = \sqrt{|c|} sign(c). \tag{26-15}$$

The contraction parameter γ is then calculated as

$$\gamma = 0.5(\gamma_{pan}(1+\hat{c}) + \gamma_{osc}(1-\hat{c})), \tag{26-16}$$

see Figure 26-2. The parameter γ_{osc} is typically 0.5-0.7 representing shrinkage to dampen oscillation, whereas γ_{pan} represents the pure panning case and therefore unity is typically chosen.

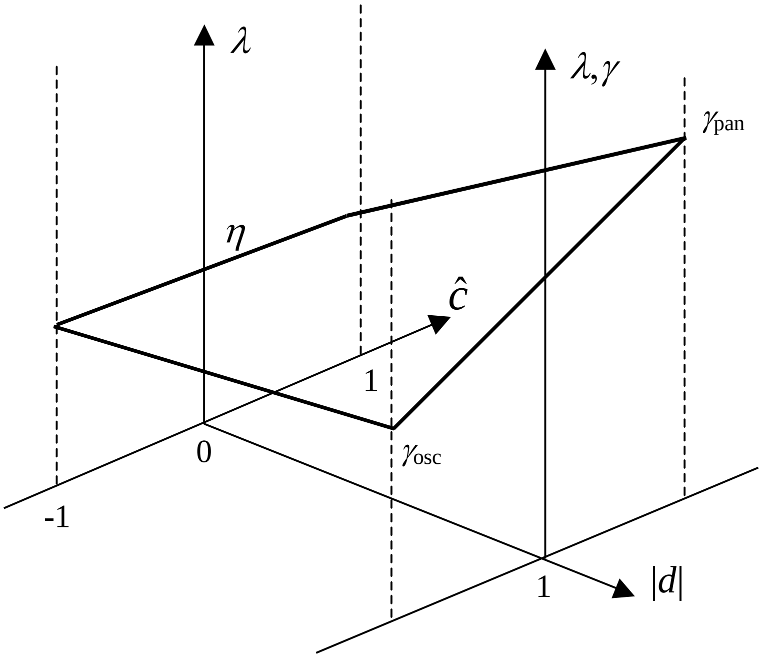


Figure 26-2: The sub-region contraction rate λ as a function of the oscillation indicator $\hat{c}$ and the absolute move distance $|d|$

Accuracy: The accuracy is estimated using the proximity of the predicted optimum of the current iteration to the starting (previous) design. The smaller the distance between the starting and optimum designs, the more rapidly the region of interest will diminish in size. If the solution is on the bound of the region of interest, the optimal point is estimated to be beyond the region. Therefore a new subregion, which is centered on the current point, does not change its size. This is called *panning* (Figure 26-1(a)). If the optimum point coincides with the previous one, the subregion is stationary, but reduces its size (*zooming*) (Figure 26-1(b)). Both panning and zooming may occur if there is partial movement (Figure 26-1(c)). The range $r_i^{(k+1)}$ for the new subregion in the (k + 1)-th iteration is then determined by:

$$r_i^{(k+1)} = \lambda_i r_i^{(k)}; i = 1,...,n;\ k = 0,...,niter \tag{26-17}$$

where λ_i represents the *contraction rate* for each design variable. To determine $\lambda_i, d_i^{(k)}$ is incorporated by scaling according to a *zoom parameter* η that represents pure zooming and the contraction parameter γ to yield the contraction rate

$$\lambda_i = \eta + \left|d_i^{(k)}\right|(\gamma - \eta) \tag{26-18}$$

for each variable (see Figure 26-2).

When used in conjunction with neural networks or Kriging, the same heuristics are applied as described above. However the nets are constructed using all the available points, including those belonging to previous iterations. Therefore the response surfaces are progressively updated in the region of the optimal point.

Refer to Section 4.9.2 for the setting of parameters in the iterative Sequential Response Surface Method.

26.7. Efficient global optimization (EGO)

Efficient global optimization (EGO) [49] is a Gaussian process model-based adaptive algorithm that aims to locate the global optimum with few sample evaluations. It was originally developed for unconstrained optimization, but several constrained formulations have also been developed since then [50][51][52][53]. The algorithm starts with an initial DOE and the corresponding Gaussian process model, and then updates the model by selecting samples with the highest expectation of improving the current solution. It should be noted that this is not same as minimizing the objective function prediction. The computation of the expected improvement (EI) is possible because the Gaussian process approximation, Kriging, not only provides a mean prediction of the underlying function, but also its variance, Section 24.2.

The EI is essentially the expected value by which the predicted objective function value is lower than the current minimum:

$$EI(\boldsymbol{x}) = E\big(I(\boldsymbol{x})\big) = E\left(max\big(0, w^* - w(\boldsymbol{x})\big)\right) = \int_{-\infty}^{w^*} (w^* - w)\, f_{\widehat{w}} dw \qquad (26\text{-}19)$$

where w is the function being approximated and $f_{\widehat{w}}$ is the probability density function based on the Kriging approximation. w^* is the target for improvement, which typically is the current best objective function value. For a Gaussian model, the EI is be expressed analytically as:

$$EI(\boldsymbol{x}) = \big(w^* - \mu_w(\boldsymbol{x})\big)\Phi\left(\frac{w^* - \mu_w(x)}{\sigma_w(x)}\right) + \sigma_w(\boldsymbol{x})\phi\left(\frac{w^* - \mu_w(x)}{\sigma_w(x)}\right) \qquad (26\text{-}20)$$

where Φ and ϕ are the standard normal probability density function and cumulative density function respectively. The point with the maximum expected improvement is evaluated to update the Kriging model.

The variance at any point in the space depends on its proximity to existing samples; it is higher in regions that are sparsely sampled. As a result, even a design point with higher mean objective function prediction can have a greater expectation of improvement compared to a point with better mean prediction lying in a densely populated or low variance region. This is also apparent from Equation (26-22), which shows that the EI can be large due to any of the two additive terms – one governed by the prediction mean and the other by the prediction variance. The same can also be said about the probability of improvement as depicted in Figure 26-3.

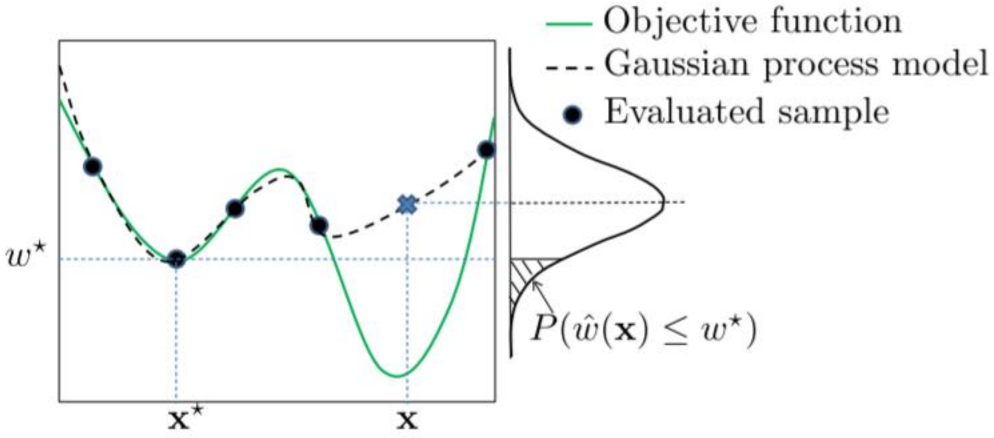


Figure 26-3: Probability of improvement at a design configuration based on Kriging. This point has a higher mean objective function prediction than the current best x*, but has a non-zero probability of improvement due to higher variance. The variance at evaluated samples is zero.

EGO, as introduced in the above discussion, was originally developed for unconstrained optimization. Several constrained approaches have also been developed since then. These are often based on the probability of feasibility based on the Kriging approximation of the constraint functions or the mean constraint prediction [51]. Constrained EGO methods using classification-based constraint handling have also been developed [52][53]. In the current version of LS-OPT, the constraint handling is done simply based on the mean prediction:

$$\max EI(\boldsymbol{x})$$

$$s.t.\, \hat{\boldsymbol{g}}(\boldsymbol{x}) \leq \boldsymbol{0}$$

The original EGO algorithm was developed as a serial adaptive sampling method, i.e. only one sample based on the maximization of EI was selected per iteration. Subsequently, several parallel EGO algorithms have been developed, e.g. based on generalized EI [54], multiple infill criteria [51], Pareto optimality approach [55] etc. These methods are currently not implemented in LS-OPT. The current LS-OPT version uses a rather simplistic parallel EGO approach – the first sample at every iteration is selected by maximizing the EI while the rest are space filling samples that lead to further global exploration and typically reduce the overall prediction variance. It should be noted that this approach focuses more on global search and therefore, while being quite efficient in locating the approximate global optimum, it may be slightly lacking in the final solution accuracy. Supplementing the EGO solution with one or more SRSM iterations can therefore be useful.

26.8. Leapfrog optimizer for constrained minimization (LFOPC)

The optimization algorithm used to solve the approximate subproblem is the LFOPC algorithm of Snyman [5]. It is a gradient method that generates a dynamic trajectory path, from any given starting point, towards a local optimum. This method differs conceptually from other gradient methods, such as SQP, in that no explicit line searches are performed.

The original leap-frog method [9] for unconstrained minimization problems seeks the minimum of a function of *n* variables by considering the associated dynamic problem of a particle of unit mass in an *n*-dimensional conservative force field, in which the potential energy of the particle at point $\boldsymbol{x}(t)$ at time *t* is taken to be the function $f(\boldsymbol{x})$ to be minimized.

The solution to the constrained problem may be approximated by applying the unconstrained minimization algorithm to a penalty function formulation of the original algorithm. The LFOPC algorithm uses a penalty function formulation to incorporate constraints into the optimization problem. This implies that when constraints are violated (active), the violation is magnified and added to an augmented objective function, which is solved by the gradient-based dynamic leap-frog method (LFOP). The algorithm uses three phases: Phase 0, Phase 1 and Phase 2. In Phase 0, the active constraints are introduced as mild penalties through the pre-multiplication of a moderate penalty parameter value. This allows for the solution of the penalty function formulation where the violation of the (active) constraints are pre-multiplied by the penalty value and added to the objective function in the minimization process. After the solution of Phase 0 through the leap-frog dynamic trajectory method, some violations of the constraints are inevitable because of the moderate penalty. In the subsequent Phase 1, the penalty parameter is increased to more strictly penalize violations of the remaining active constraints. Finally, and only if the number of active constraints exceed the number of design variables, a compromised solution is found to the optimization problem in Phase 2. Otherwise, the solution terminates having reached convergence in Phase 1. The penalty parameters have default values as listed in the User’s manual (Section 12.4.1). In addition, the step size of the algorithm and the termination criteria of the subproblem solver are listed.

The values of the responses are scaled with the values at the initial design. The variables are scaled internally by scaling the design space to [0; 1] interval. The default parameters in LFOPC (as listed in Section 12.4.1) should therefore be adequate. The termination criteria are also listed in Section 13.1.

In the case of an infeasible optimization problem, the solver will find the most feasible design within the given region of interest bounded by the simple upper and lower bounds. A global solution is attempted by multiple starts from the experimental design points.

26.9. Genetic algorithm

Genetic algorithms are nature inspired search algorithms that emulate the Darwinian principle of ‘survival of the fittest’. The concept of nature inspired algorithms was first envisaged by Prof. John Holland [10] at the University of Michigan in mid sixties. Later on this theory gained momentum in engineering optimization following the work of Prof. David Goldberg [11] and his students. The differences between genetic algorithms and most conventional optimization methods are:

- GA does not require derivative information to drive the search of optimal points.
- While conventional methods use a single point at each iteration, GA is a population based approach.
- GA is a global optimizer whereas conventional methods may get stuck in local optima.
- GA is a probabilistic optimization method that is, an inferior solution (that may help evolve the correct design variables structure) may also have a non-zero probability of participating in the search process.
- The computational cost of using GA may be high compared to derivative based methods.

26.9.1. Terminology

The Genetic Algorithm imitates nature so some of its terminology is derived from biology:

- **Individual** – Each design variable vector (often known as solution or design point) is called an individual.
- **Population** – A group of individuals is called a population. The number of individuals in a population is termed *population size*.
- **Chromosome** – The binary string used to encode design variables is called chromosome. Chromosomes are used with binary encoding or conventional GA only. There is no direct correspondence of chromosome in real coded GA. The length of a chromosome is the sum of number of bits assigned to each variable.
- **Gene** – In binary encoding, each bit is called a *gene*.
- **Fitness** – The fitness of an individual is analogous to objective function. Each individual is assigned a fitness value based on its objectives and constraints values. The selection process tries to maximize the fitness of a population. The individual with the highest fitness represents the optimal solution to a problem.
- **Generation** – A generation (iteration in general optimization lingo) comprises of application of genetic operators – selection, crossover, and mutation – to create a child population. At the end of each generation, the child population becomes the parent population.

26.9.2. Encoding

To use the genetic algorithm for optimization, design variables of a mathematical optimization problem are encoded into a format required by GA. There are two prominent ways of encoding design variables:

- **Binary encoding** – The conventional approach of using genetic algorithm is to represent an optimization problem into a string of binary numbers (chromosomes). The number of bits assigned to each variable determines the solution accuracy. If p bits are used to represent a variable with lower and upper bounds x_l and x_u, respectively, the accuracy of this variable can be $(x_u\text{-}x_l)/(2^p\text{-}1)$. While binary encoding is the most natural way to use genetic algorithms, it has two main problems: i) discretization of a continuous variable causes loss of accuracy in representation (depends on number of bits), ii) Hamming cliff problem – neighbors in real space may not be close in binary space such that it may be very difficult to find an optimal solution.
- **Real encoding** – To avoid the problems of using binary representation of real variables, researchers have suggested directly using real numbers. This required special methods to perform genetic operations like crossover and mutation.

26.9.3. Algorithm

The steps in a simple genetic algorithm are illustrated with the help of Figure 26-4.

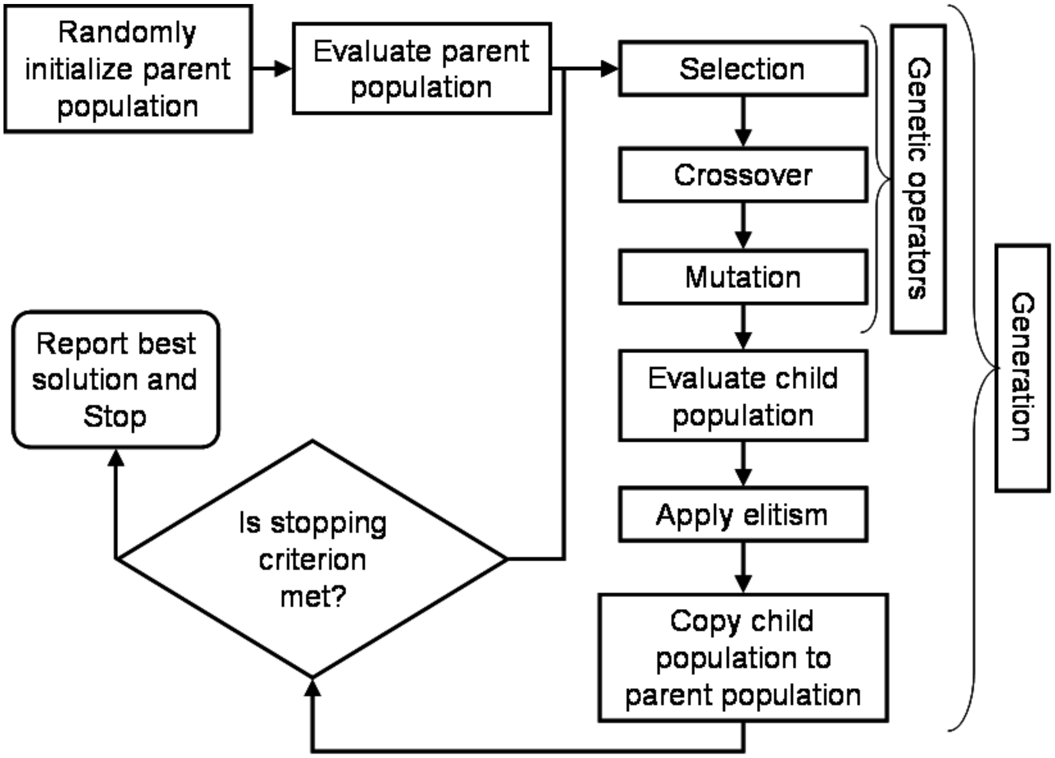

Randomly initialize parent population
Evaluate parent population
Selection
Crossover
Mutation
Genetic operators
Evaluate child population
Apply elitism
Copy child population to parent population
Generation
Report best solution and Stop
Is stopping criterion met?

Figure 26-4: Simple genetic algorithm.

Firstly, problem-specific GA parameters like population size N_{pop}, type of encoding, number of bits per variables for binary coding, number of generations are defined.

Initialization

Next, the population is randomly initialized i.e., binary chromosomes or real variable vectors for N_{pop} individuals are generated randomly.

Function evaluation

For binary encoding, each chromosome (binary string) is decoded to corresponding design variable vector. Next, objective functions, constraints, and constraint violation of each individual in parent population is evaluated and accordingly fitness of each individual is assigned.

Selection or reproduction operator

Selection operator is used to identify individuals with high fitness and to form a mating pool of size N_{pop}. This operator reduces diversity in the population by filtering out low fitness schema. Many reproduction operators are introduced in literature. Three selection operators implemented in LS-Opt are described below.

- **Tournament selection**. In tournament selection, 'N_{tourn}' (N_{tourn} is tournament size) individuals from a population, selected at random, participate in a tournament. The individual with the largest fitness is declared the winner. Mostly, practitioners use N_{tourn} = 2. Increasing the tournament size 'N_{tourn}'

increases selection pressure and might lose diversity in the population that is required to drive the search.

- **Roulette wheel or proportionate reproduction**. In this selection approach, each individual is assigned a probability of selection based on its fitness value. In a population of N_{pop} individuals, the selection probability of the i^{th} individual is

  $$P_i = F_i \Big/ \sum_{j=1}^{N_{pop}} F_j \qquad (26\text{-}21)$$

  where F_i is the fitness of i^{th} individual. High fitness individuals have a high probability of getting selected. This scheme is implemented by considering a roulette wheel with circumference marked by the fitness of each individual. One individual per spin of the wheel is selected. Then, the expected number of copies of the i^{th} individual in the mating pool can be estimated as

  $$N_i = F_i \big/ \bar{F}; \bar{F} = \frac{1}{N_{pop}} \sum_{j=1}^{N_{pop}} F_j . \qquad (26\text{-}22)$$

  This selection operator has a higher selection pressure compared to the tournament selection and can lead to a premature convergence to local optima.

- **Stochastic universal sampling**. The roulette wheel selection operator is often noisy because of multiple spins that correspond to round-off errors in computer simulations. To reduce this noise, it was suggested to use a single spin of the wheel with N_{pop} equi-spaced pointers. This operator also has a high selection pressure.

Crossover

Crossover is the main exploration operator of genetic search. In this operator, μ randomly selected parents mate with a probability (P_c: crossover probability) to create λ children. These children share the attributes from all parents such that they may be better or worse individuals. There are two prominent strategies to create children: i) $(\mu + \lambda)$ strategy selects best individuals from parents and children, and ii) (μ, λ) strategy replaces parents with children irrespective of their fitness values. LS-OPT has adopted a (2, 2) strategy for crossover such that two parents create two children and children replace parents in the new generation. If parents do not create children, they are passed to the next generation.

There are many crossover operators in literature. A few popular crossover operators that have been shown to perform reasonably well are available in LS-OPT. A brief description of these operators is as follows:

- Single point binary crossover

  This crossover operator is used for binary encoding of the individuals. Two parents and a mating site are randomly selected. All genes right to the mating sites are swapped between two parents.

- Uniform binary crossover

  This crossover operator is also used for binary encoded individuals. For a randomly selected parent pair, genes are swapped based on a flip of a coin for each gene in the chromosome.

- Simulated binary real crossover (SBX)

This crossover operator, introduced by Deb and Agrawal in 1995 [12], is used with real encoding i.e., real variables are used as genes. This crossover emulates the single point binary crossover by assigning a probability distribution to each parent. Two children are created from two parents using that probability distribution such that the mean of parents and children are the same. The probability distribution is controlled by a distribution index η_c such that large value of η_c creates children near parents and small value of η_c creates children far from parents. Deb and Beyer [13] showed that SBX possesses self-adaptation capabilities.

- Blend real crossover (BLX-α)

  This crossover operator was introduced by Eshelman and Schaffer in 1993 [14]. In this crossover, a child x is created from two parents $x^{(1)}$ and $x^{(2)}$ ($x^{(2)} > x^{(1)}$) by randomly selecting a value from the interval $[x^{(1)} - \alpha(x^{(2)} - x^{(1)}), x^{(2)} + \alpha(x^{(2)} - x^{(1)})]$. Typically, α is taken as 0.5.

Mutation

Mutation is carried out with a mutation probability (P_m) to bring random changes in the individuals. This operator is very useful when population has lost diversity and the search has become stagnant. Then mutation can help improve diversity in the solutions. The mutation operators for binary and real encoding are given as follows:

- Simple binary mutation

  In simple binary mutation of an individual, a bitwise mutation is carried out by changing a ‘0’ to ‘1’ or vice-versa with a small mutation probability P_m. Typically P_m is taken as the inverse of chromosome length such that on an average, one gene (bit) per chromosome is changed.

- Real mutation

  As was used for the SBX operator, a distribution (defined by mutation distribution index) around each variable is specified and a random variable is selected from that distribution. Large values of the distribution index are recommended to create a child near the parent.

A complete cycle of selection, crossover, and mutation would result in a child population. The population size is kept constant for both parent and child populations.

Elitism in simple genetic algorithm

Due to the disruptive nature of exploration operators, high fitness individuals may get lost while creating a child population from the parent population. Sometimes, it is advantageous to keep these high fitness individuals to preserve favorable genetic information (schema). This process of artificially saving the best individuals is called elitism. To implement this process, the parent and child populations are ranked separately. The worst individuals in the child population are replaced by the best individuals from the parent population. The number of elites should be carefully chosen because a large number of elite solutions may drive the search to local optima by reducing the diversity in the population. On the other hand, too few elites may slow the convergence because favorable schema would spread at a slow rate.

After applying elitism, the child population is transferred to the parent population. The best individual found in the search process is preserved at each generation.

Stopping criterion

Many criteria have been specified in literature to terminate the GA search process. Some researchers have suggested stopping the search when there is no improvement in the last few generations. However, the most common stopping criterion is the fixed number of generations or function evaluations. A user-defined number of generations is used as the stopping criterion in LS-OPT.

At the end of simple genetic algorithm, the best individual (among all searched individuals) is reported as the optimal solution. If enough processing capabilities are carried out, the reported best individual would represent the global optimal solution.

26.10. Multi-objective optimization using genetic algorithms

Multi-objective optimization problems are significantly different than the single-objective optimization problems. MOO problems do not have a single optimal solution. Instead there is a set of solutions that reflects trade-offs among objectives. For MOO problems, population based methods like genetic algorithms are very attractive because many trade-off solutions can be found in a single simulation run. While it is easy to compare multiple designs for a single-objective optimization problem, special considerations are required to compare different designs. Goldberg [11] proposed a non-domination concept to compare different individuals. This idea forms the backbone of most MOGAs and is defined next.

26.10.1. Non-domination criterion

A non-domination criterion is used to identify better individuals without introducing any bias towards any objective ([17]-[19]). To understand the non-domination criterion, a domination criterion is defined as follows.

A solution $\mathbf{x}^{(1)}$ dominates another solution $\mathbf{x}^{(2)}$ $\left(\mathbf{x}^{(1)} \succ \mathbf{x}^{(2)}\right)$, if either of the following three conditions is true.

1. $x^{(1)}$ is feasible and $x^{(2)}$ is infeasible.
2. Both $x^{(1)}$ and $x^{(2)}$ are infeasible but $x^{(2)}$ is more infeasible compared to $x^{(1)}$.
3. When both $x^{(1)}$ and $x^{(2)}$ are feasible, $x^{(1)}$ dominates $x^{(2)}$ $\left(\mathbf{x}^{(1)} \succ \mathbf{x}^{(2)}\right)$ if following two conditions are satisfied
   - $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in 'all' objectives, i.e. $(f_j(\mathbf{x}^{(1)}) \ngtr f_j(\mathbf{x}^{(2)}))\ j \in [1,2,...,M]$.
   - $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in 'at least one' objective, i.e., $(f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)})), \wedge j \in [1,2,...,M]$.

If neither of the two solutions dominates the other, both solutions are non-dominated with respect to each other. An individual **s** is considered non-dominated with respect to a set of solutions ***S***, if no solution in ***S*** dominates **s**.

26.10.2. Pareto optimal solutions

Any non-dominated solution in the entire design domain is a Pareto optimal solution. By definition, all Pareto optimal solutions are non-dominated solutions but vice-versa is not true.

Like single objective optimization problems, there are local and global Pareto optimal solutions. A non-dominated solution is a local Pareto optimal solution with respect to the considered non-dominated solution set, whereas a global Pareto optimal solution is non-dominated with respect to all solutions in the design domain.

26.10.3. Pareto optimal set

The set of all Pareto optimal solutions is the Pareto optimal set for the given problem.

26.10.4. Pareto optimal front

Function space representation of the Pareto optimal set is Pareto optimal front. When there are two conflicting objectives, the POF is a curve, when there are three objectives, POF is a surface, and for higher dimensions, POF is a hyper-surface.

26.10.5. Ranking

Most MOGA search methods assign rank to different individuals based on non-domination criterion. This ranking is used to govern the search process. A rank of one is considered the best rank and low fitness individuals are assigned low ranks (large values of rank are low). Different individuals in a population are assigned rank as follows:

1. Initialize *rnk* = 1. Define a set of individuals **S**, same as the population.
2. Run a non-domination check on all individuals in **S**.
3. All non-dominated individuals are assigned rank = *rnk*.
4. *rnk* = *rnk* + 1.
5. Remove all non-dominated individuals from **S**.
6. If $\mathbf{S} \neq \Phi$, repeat Step 2, else stop.

Note that many individuals can have the same rank.

Different concepts discussed here are illustrated using a two-objective unconstrained minimization problem in Figure 26-5. Each dot represents a solution in the design space that is shown as the shaded area. For each diamond, there is at least one triangle that is better than the diamond in at least one objective without being inferior in other objective. So all individuals represented by diamonds are dominated by the individuals represented by triangles. Similarly, all triangles are dominated by squares and squares are dominated by circular dots. No solution represented by triangles can be said better than any other solution represented by triangles. Thus, they are non-dominated with respect to each other. All individuals represented by circles are non-dominated with respect to any other individual hence they have a rank of one (best rank). If all points represented by circles are removed, the individuals represented by squares are non-dominated with respect to all remaining solutions such that they are assigned a rank of two, and so on. Note that all individuals with the same shape of dots have the same rank. For this example, all individuals with rank one (circular dots) also represent the true Pareto optimal solutions set. The line on the boundary shows the Pareto optimal front.

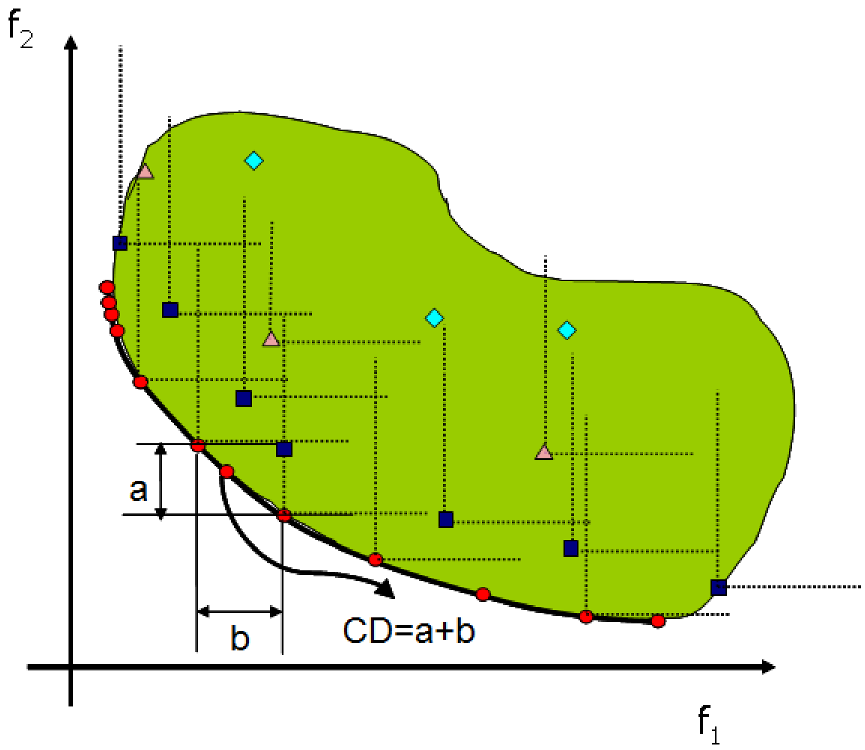


Figure 26-5: Illustration of non-domination criterion, Pareto optimal set, and Pareto optimal front.

26.10.6. Convergence vs. diversity

Different multi-objective optimization algorithms are compared using two criteria. First, convergence to the global Pareto optimal front, and second, diversity on the Pareto optimal front. The convergence criterion requires identifying the global Pareto optimal solution set.

A good multi-objective optimizer is required to maintain diversity (representation of different regions of the Pareto optimal front). This is an important criterion since our goal is to find different trade-off solutions. It is important to note that diversity on the Pareto optimal front (function space) does not mean the diversity in the variable space, i.e., small changes in variables can result in large changes in the function values.

26.10.7. Elitist non-dominated sorting genetic algorithm (NSGA-II)

This algorithm was developed by Prof. Kalyanmoy Deb and his students in 2000 [20]. This algorithm first tries to converge to the Pareto optimal front and then it spreads solutions to get diversity on the Pareto optimal front. Since this algorithm uses a finite population size, there may be a problem of Pareto drift. To avoid that problem, Goel et al. [21] proposed maintaining an external archive.

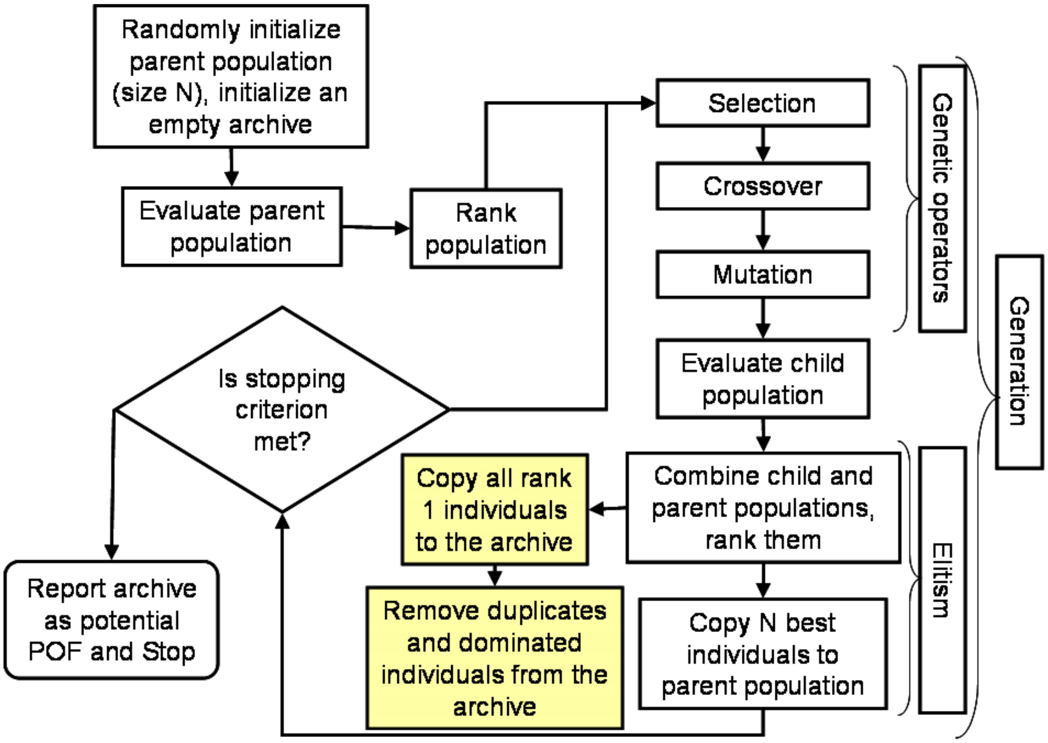

Randomly initialize parent population (size N), initialize an empty archive
Evaluate parent population
Rank population
Selection
Crossover
Mutation
Evaluate child population
Genetic operators
Generation
Is stopping criterion met?
Copy all rank 1 individuals to the archive
Combine child and parent populations, rank them
Elitism
Report archive as potential POF and Stop
Remove duplicates and dominated individuals from the archive
Copy N best individuals to parent population

Figure 26-6: Elitist non-dominated sorting genetic algorithm (NSGA-II). The shaded blocks are not the part of original NSGA-II but additions to avoid Pareto drift.

The implementation of this archived NSGA-II is shown in Figure 26-6, and described as follows:

1. Randomly initialize the parent population (size N_{pop}). Initialize an empty archive.
2. Evaluate the population i.e., compute constraints and objectives for each individual.
3. Rank the population using non-domination criteria. Also compute the crowding distance (this distance finds the relative closeness of a solution to other solutions in the function space and is used to differentiate between the solutions on same rank).
4. Employ genetic operators – selection, crossover & mutation – to create a child population.
5. Evaluate the child population.
6. Combine the parent and child populations, rank them, and compute the crowding distance.
7. Apply elitism (defined in a following section): Select best N_{pop} individuals from the combined population. These individuals constitute the parent population in the next generation.
8. Add all rank = 1 solutions to the archive.
9. Update the archive by removing all dominated and duplicate solutions.
10. If the termination criterion is not met, go to step 4. Otherwise, report the candidate Pareto optimal set in the archive.

26.10.8. Elitism in NSGA-II

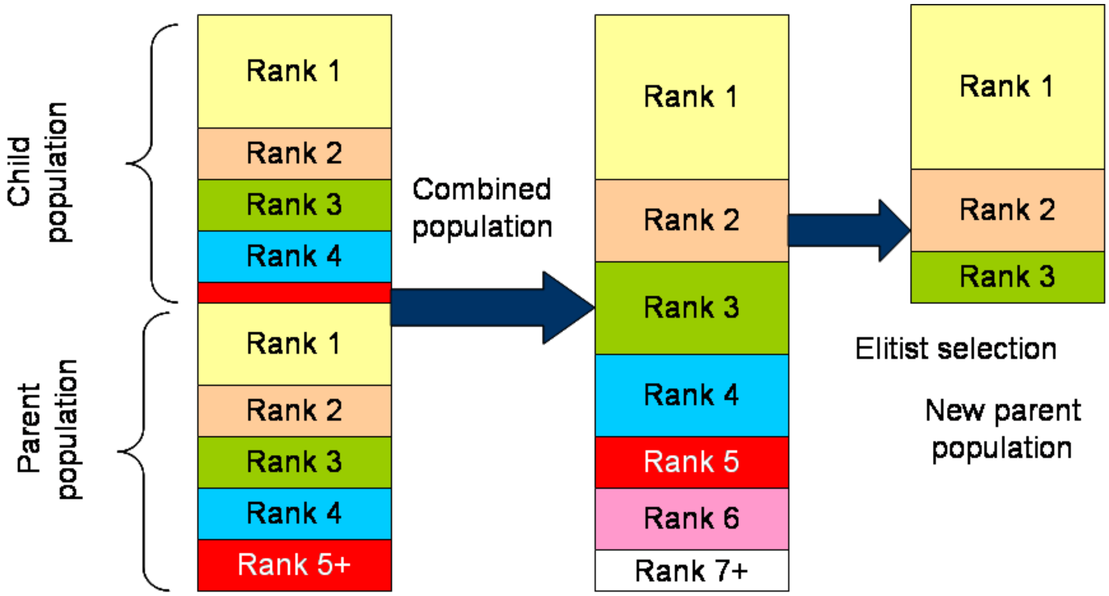


Figure 26-7: Elitism in NSGA-II.

Elitism is applied to preserve the best individuals. The mechanism used by NSGA-II algorithm for elitism is illustrated in Figure 26-7. After combining the child and parent populations, there are $2N_{pop}$ individuals. This combined pool of members is ranked using non-domination criterion such that there are n_i individuals with rank *i*. The crowding distance of individuals with the same rank is computed. Steps in selecting N_{pop} individuals are as follows:

1. Set $i = 1$, and number of empty slots $N_{slots} = N_{pop}$.
2. If $n_i < N_{slots}$,
   - Copy all individuals with rank '*i*' to the new parent population.
   - Reduce the number of empty slots by n_i: $N_{slots} = N_{slots} – n_i$.
   - Increment '*i*': $i=i+1$.
   - Return to Step 2.
3. If $n_i > N_{slots}$,0.
   - Sort the individuals with rank '*i*' in decreasing order of crowding distance.
   - Select N_{slots} individuals.
   - Stop

26.10.9. Diversity preservation mechanism in NSGA-II – crowding distance calculation

To preserve diversity on the Pareto optimal front, NSGA-II uses a crowding distance operator. The individuals with same rank are sorted in ascending order of function values. The crowding distance is the sum of distances between immediate neighbors, such that in Figure 26-5, the crowding distance of selected individual is ‘*a* + *b*’. The individuals with only one neighbor are assigned a very high crowding distance.

Note: It is important to scale all functions such that they are of the same order of magnitude otherwise the diversity preserving mechanism would not work properly.

26.11. Adaptive simulated annealing (ASA)

The Simulated Annealing (SA) is a global stochastic optimization algorithm that mimics the metallurgical annealing process. The original simulated annealing algorithm was developed as a generalization of the Metropolis Monte Carlo integration algorithm [22] to solve various combinatorial problems by Kirkpatrick et al. [23]. The term 'simulated annealing' derives from the rough analogy of the way that the liquefied metals at a high temperature crystallize on freezing. At high temperatures, the atoms in the liquid are at a high energy state and move freely. When the liquid is cooled, the energy of the molecules gradually reduces as they go through many lower energy states, and consequently their motion. If the liquid metal is cooled too quickly or 'quenched', the atoms do not get sufficient time to reach thermal equilibrium at a temperature and might result in a polycrystalline structure with higher energy. This atomic structure of material is not necessarily the most desired. However, if the rate of cooling is sufficiently slow, the atoms are often able to achieve the state of minimum (most stable) energy at each temperature state, resulting in a pure crystalline form. This process is termed as ‘annealing’. Kirkpatrick et al. [23] employed this annealing analogy to develop an efficient search algorithm. Pincus [24], and Cerny [25] also are also independently credited with the development of modern simulated annealing algorithm.

In simulated annealing parlance, the objective function of the optimization algorithm is often called ‘energy’ E and is assumed to be related to the state, popularly known as temperature T, by a probability distribution. The Boltzmann distribution is the most commonly used probability distribution:

$$\text{Probability (E)} \sim \exp(-E / \kappa_B T),$$

where κ_B is the Boltzmann's constant.

26.11.1. Algorithm

The search initializes with the temperature being high and cooling slowly such that the system goes through different energy states in search of the lowest energy state that is the global minima of the optimization problem. A stepwise description of the simulated annealing algorithm is as follows:

Step 1: Initialization

The search starts by identifying the starting state $\boldsymbol{x}^{(0)} \in \boldsymbol{X}$ and corresponding energy $E^{(0)} = E(\boldsymbol{x})$. The temperature T is initialized at a high value: $T^{(0)} = T_{\max}$. A cooling schedule, acceptance function, and stopping criterion are defined. This is iteration $k = 0$. $\boldsymbol{X}^{(0)} = \{\boldsymbol{x}^{(0)}\}$.

Step 2: Sampling

A new point $\boldsymbol{x}' \in \boldsymbol{X}$ is sampled using the candidate distribution $D(\boldsymbol{X}^{(k)})$, and set $\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)}\ U\ \{\boldsymbol{x}'\}$, and corresponding energy is calculated $E' = \mathrm{E}(\boldsymbol{x}')$.

Step 3: Check acceptance

Sample a uniform random number ζ in [0, 1] and set

$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}'$ if $\zeta \leq A(\boldsymbol{E}',\boldsymbol{E}^{(k)},T^{(k)})$ or

$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$ otherwise.

where A(x) is the acceptance function that determines if the new state is accepted.

Step 4: Temperature update

Apply the cooling schedule to the temperature: $T^{(k+1)} = C(\boldsymbol{X}^{(k+1)}, T^{(k)})$.

Step 5: Convergence check

Stop the search if the stopping criterion is met, else set $k = k+1$ and go to Step 2.

As is obvious, the efficiency of the simulated annealing algorithm depends on appropriate choices of the mechanism to generate new candidate states *D*, cooling schedule *C*, acceptance criterion *A*, and stopping criterion. While many options have been proposed in literature, the very fast simulated reannealing methodology proposed by Ingber (1989) [27] has been the most promising. This algorithm is also known as adaptive simulated annealing (ASA) [28]. The different selections along with a very brief historical perspective are outlined as follows.

26.11.2. Acceptance function

Two most prominent *acceptance functions* used to accept a candidate point are the Metropolis criterion and the Barker criterion.

Metropolis criterion : $$A(E^{'},E,T) = \min\{1, \exp(-(E^{'} - E)/T)\}.$$

Barker criterion : $$A(E^{'},E,T) = 1/\{1 + \exp((E^{'} - E)/T)\}.$$

The theoretical motivation for such a restricted choice of acceptance functions can be found in [29]. It is also shown that under appropriate assumptions, many acceptance functions, which share some properties, are equivalent to the above two criteria. The Metropolis criterion is the most commonly used selection criterion and this is chosen as the acceptance function in LS-OPT.

26.11.3. Sampling algorithm

The choice of the next candidate distribution and the cooling schedule for the temperature are typically the most important (and strongly interrelated) issues in the definition of a SA algorithm. The *next candidate state*, $\boldsymbol{x}'$, is usually selected randomly among all the neighbors of the current solution, $\boldsymbol{x}$, with the same probability for all neighbors. The choice of the size of the neighborhood typically follows the idea that when the current function value is far from the global minimum, the algorithm should have more freedom, i.e., larger 'step sizes' are allowed. However, Ingber [27] suggested using a more complicated, non-uniformly selection procedure outlined below to allow much faster cooling rates.

Let i^{th} design variable be bounded as, $x_i \in [A_i, B_i]$. Then the new sample is given by

$$x_i^{'} = x_i^{(k)} + \nu_i (B_i - A_i),$$

where ν_i is estimated as follows.

$$\nu_i = \operatorname{sgn}(u - 0.5) T_{p,i}^{(k)} \left\lfloor (1 + 1/T_{p,i}^{(k)})^{|2u-1|} - 1 \right\rfloor; \; u \in U[0,1].$$

The most important distinction in ASA with standard SA is the use of an independent temperature schedule ($T_{p,i}$) for each parameter along with the temperature associated with the energy function. The cooling schedule for the parameter temperature, used to generate N dimensional design vector, is

$$T_{p,i}^{(k)} = T_{p,i}^{(0)} \exp(-c_i k^{1/N}).$$
$$c_i = m_i \exp(-n_i / N).$$

The control parameter c_i depends on two free parameters m_i and n_i, defined as

$$m_i = \log(T_{p,i}^{\min} / T_{p,i}^{(0)}), n_i = \log(N_{anneal}).$$

The ratio $T_{p,i}^{\min} / T_{p,i}^{(0)}$ is the parameter temperature ratio and the parameter N_{anneal} is linked to the time allowed (number of steps) at each parameter temperature state. Ingber [30] found that the search procedure is sensitive to the choice of the two parameters and should be selected carefully. Relatively, the parameter temperature ratio is the more important of the two parameters.

26.11.4. Cooling schedule

The basic idea of the *cooling schedule* is to start the algorithm off at high temperature and then gradually drop the temperature to zero. The primary goal is to quickly reach the so called effective temperature, roughly defined as the temperature at which low function values are preferred but it is still possible to explore different states of the optimized system, [31]. After that the simulated annealing algorithm lowers the temperature by slow stages until the system 'freezes' and no further changes occur. Geman and Geman [32] found the lower bound on the cooling schedule to be 1/log(t) where ‘t’ is an artificial time measure of the annealing schedule. Hence,

$$T_i^{(k+1)} = T_i^{(0)} / \log(k).$$

This strategy is also known as Boltzmann annealing (Szu and Hartley) [33]. Later van Laarhoven and Aarts [34] modified this strategy to enable a much faster cooling schedule of

$$T_i^{(k+1)} = T_i^{(0)} / k.$$

A straightforward and most popular strategy is to decrement T by a constant factor every ν_T iterations:

$$T := T / \mu_T$$

where μ_T is slightly greater than 1 (e.g. $\mu_T = 1.001$). The value of ν_T should be large enough, so that 'thermal equilibrium' is achieved before reducing the temperature. A rule of thumb is to take ν_T proportional to the size of neighborhood of the current solution.

Nevertheless, the fastest cooling rate was made possible by using Ingber's algorithm that allowed an exponentially faster cooling rate of

$$T_i^{(k)} = T_i^{(0)} \exp(-ck^{1/N})$$
$$c = m \exp(-n/N).$$

As was described in the previous section, the cooling rate is governed by the two free parameters that are linked to the temperature ratio and annealing scale,

$$m = \log(T^{\min} / T^{(0)}),\ n = \log(N_{anneal}).$$

Typically the temperature ratio used to drive the energy (objective) function is linked to the parameter temperature ratio called here as ‘cost-parameter annealing ratio’.

26.11.5. Stopping criterion

Selection of an appropriate stopping criterion is one of the most difficult tasks in stochastic optimization algorithms because it is unknown *a priori* if the algorithm has reached the global optima or is stuck in a hard local optimum. Thus the stopping rules proposed in the literature about SA, all have a heuristic nature and are, in fact, more problem dependent than algorithm dependent. Some common ideas in the heuristics are i) stop when it does not make a noticeable progress over a number of iterations, ii) stop when the number of function evaluations is reached, and iii) stop when the temperature has fallen substantially to a desired minimum level T_{min}. The last two criteria are used to terminate the adaptive simulated annealing search in LS-OPT.

26.11.6. Re-annealing

For multi-dimensional problems, most often the objective function has variable sensitivities for different parameters and at different sampling states. Hence, it is worth while to adjust the cooling rates for different parameters. Ingber [27] used a reannealing algorithm to periodically update the annealing time associated with parameters and the energy function such that the search is more focused in the regions with potential of improvements. For this, he suggested computing the sensitivities of the energy function as,

$$s_i = \partial E / \partial x_i .$$

All the annealing time parameters k are updated by the largest sensitivity s_{max} as follows:

$$T_{p,i}^{(k)} = T_{p,i}^{(k)} (s_{\max} / s_i),$$
$$k_i^{'} = (\log(T_{p,i}^{(0)} / T_{p,i}^{(k)}))^N .$$

The new annealing time associated with the i^{th} parameter is $k_i = k_i^{'}$. Similarly the temperature parameter associated with the energy function is scaled. One can easily deduce from the above formulation that reannealing stretches the ranges of the insensitive parameters relative to the sensitive parameters. More details of reannealing can be obtained elsewhere [30].

Some comments

1. It is difficult to find the initial temperature directly, because this value depends on the neighborhood structure, the scale of the objective function, the initial solution, etc. In [23] a suitable initial temperature is one that results in an average uphill move acceptance probability of about 0.8. This $T^{(0)}$ can be estimated by conducting an initial search, in which all uphill moves are accepted and calculating the average objective increase observed. In some other papers, it is suggested that parameter $T^{(0)}$ is set to a value, which is larger than the expected value of $|E'-E|$ that is encountered from move to move. In [31] it is suggested to spend *most* of the computational time in short sample runs with different $T^{(0)}$ in order to detect the effective temperature. In practice, the optimal control of T may require physical insight and trial-and-error experiments. According to [35], "choosing an annealing schedule for practical purposes is still something of a black art".
2. Simulated annealing has proved surprisingly effective for a wide variety of hard optimization problems in science and engineering. Many of the applications in our list of references attest to the power of the method. This is not to imply that a serious implementation of simulated annealing to a difficult real world problem will be easy. In the real-life conditions, the energy trajectory, i.e. the sequence of energies following each move accepted, and the energy landscape itself can be highly complex. Note that state space, which consists of wide areas with no energy change, and a few "deep, narrow valleys", or even worse, "golf-holes", is not suited for simulated annealing, because in a "long, narrow valley" almost all random steps are uphill. Choosing a proper stepping scheme is crucial for SA in these situations. However, experience has shown that simulated annealing algorithms are more likely trapped in the *largest basin*, which is also often the basin of attraction of the global minimum or of the deep local minimum. Anyway, the possibility, which can always be employed with simulated annealing, is to adopt a multi-start strategy, i.e. to perform many different runs of the SA algorithm with different starting points.
3. Another potential drawback of using SA for hard optimization problems is that finding a good solution can often take an unacceptably long time. While SA algorithms may quickly detect the region of the global optimum, they often require a few iterations to improve its accuracy. For small and moderate optimization problems, one may be able to construct effective procedures that provide similar results much more quickly, especially in cases when most of the computing time is spent on calculations of values of the objective function. However, it should be noted that for large-scale multidimensional problems an algorithm which always (or often) obtains a solution near the global optimum is valuable, since various local deterministic optimization methods allow quick *refinement* of a nearly correct solution.

In summary, simulated annealing is a powerful method for global optimization in challenging real world problems. Certainly, some "trial and error" experimentation is required for an effective implementation of the algorithm. The energy (cost) function should employ some heuristic related to the problem at hand, clearly reflecting how 'good' or 'bad' a given solution is. Random perturbations of the system state and corresponding cost change calculations should be simple enough, so that the SA algorithm can perform its iterations efficiently. The scalar parameters of the simulated annealing algorithm have to be chosen carefully. If the parameters are chosen such that the optimization evolves too fast, the solution converges directly to some, possibly good, solution depending on the initial state of the problem.

26.12. Differential Evolution

Differential Evolution (DE) is a parallel direct search method developed by Storn and Price in 1997 [41] and primarily targeted at unconstrained global optimization problems. It is similar to the Genetic Algorithm and Particle Swarm Optimization in that it uses a population of parameter vectors which evolves over time. DE follows the same basic steps also employed generation-wise in the GA, namely Mutation, Crossover and Selection, although the detail of the implementation is quite different. The basic steps are described in Reference [41] as follows:

Mutation

For each target vector $\boldsymbol{x}_i^k, i = 1,2,3,\dots,NP$, NP the population size, a mutant vector is generated according to

$$\boldsymbol{v}_i^k = \boldsymbol{x}_{r_1}^k + F(\boldsymbol{x}_{r_2}^k - \boldsymbol{x}_{r_3}^k)$$

With random indices $r_1, r_2, r_3 \in \{1,2,3,\dots,NP\}$, integer, mutually different and $F > 0$. The randomly chosen integers r_1, r_2 and r_3 are also chosen to be different from the running index i, so that NP must be ≥ 4 to allow for this condition. F is a real and constant factor ∈ [0,2] which controls the amplification of the differential variation $(\boldsymbol{x}_{r_2}^k - \boldsymbol{x}_{r_3}^k)$.

Crossover

To increase the diversity of the perturbed parameter vectors, crossover is introduced. To this end the trial vector:

$$\boldsymbol{u}_d^{k+1} = (u_{1i}^{k+1}, u_{2i}^{k+1}, \dots, u_{ni}^{k+1})$$

is formed where

$$u_{ji}^{k+1} = \begin{cases} v_{ji}^{k+1} \text{ if } & (randb(j) \leq CR) \text{ or } j = rnbr(i) \\ x_{ji}^k \text{ if } & (randb(j) > CR) \text{ and } j \neq rnbr(i) \end{cases},$$

$$j = 1,2,\dots,n$$

In the last equation, $randb(j)$ is the jth evaluation of a uniform random number generator with outcome ∈ [0,1]. CR is the crossover constant ∈ $[0,1]$ which has to be determined by the user. $rnbr(i)$ is a randomly chosen index ∈ $1,2,\dots,D$ which ensures that u_{ji}^{k+1} gets at least one parameter from v_{jd}^{k+1}.

Selection

To decide whether or not it should become a member of generation $k+1$, the trial vector $\boldsymbol{u}_i^{k+1}$ is compared to the target vector $\boldsymbol{x}_i^k$ using the greedy criterion. If vector $\boldsymbol{u}_i^{k+1}$ yields a smaller cost function value than $\boldsymbol{x}_i^k$ the $\boldsymbol{x}_i^{k+1}$ is set to $\boldsymbol{u}_{ji}^{k+1}$; otherwise the old value $\boldsymbol{x}_i^{k+1}$ is retained.

Other DE variants

In [41] Storn and Price also discuss other variants of DE in which they use the notation: DE/x/y/z where

- *x* specifies the vector to be mutated which can be “rand” (a randomly chosen population vector) or “best” (the vector of lowest cost from the current population).
- *y* is the number of difference vectors used.
- *z* denotes the crossover scheme. The variant described above is known as “bin” (crossover due to independent binomial experiments).

Hence the strategy as described above is known as DE/rand/1/bin. The paper also mentions that DE/best/2/bin could be beneficial. In this case

$$\boldsymbol{v}_i^{k+1} = \boldsymbol{x}_{\text{best}}^k + F.\left(\boldsymbol{x}_{r_1}^k + \boldsymbol{x}_{r_2}^k - \boldsymbol{x}_{r_3}^k - \boldsymbol{x}_{r_4}^k\right).$$

It is concluded that the usage of two difference vectors seems to improve the diversity of the population if the number of population vectors *NP* is high enough.

Because DE is such a simple strategy, the astonishing result from [41] was that several examples showed DE/rand/1/bin to be superior to several other algorithms tested, namely Adaptive Simulated Annealing [28], the Annealed Nelder and Mead approach [43], the Breeder Genetic Algorithm [44], the EASY Evolution Strategy [45] and the method of Stochastic Differential Equations [46].

Constrained Optimization

The DE algorithm implemented in LS-OPT was developed by Kitayama *et al* [42]. The differences between the Kitayama and Storn & Price algorithms are mostly of a minor nature. Whereas Storn and Price used a constant F (mostly 0.5) for all the variables, a randomized $F_i = F + d(rand(0,1) - 0.5)$ where $i = 1,2,\ldots,n$, $F = 0.8$; $d = 0.0001$ was used in Reference [42]. Constraints are handled using a penalty formulation:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + r.\gamma;\ \ r = (1 + |f(x)|)^q;\quad q = 2$$

$$\gamma = \begin{cases} \sum_{j=1} \exp\left(1 + g_j(x)\right);\ \ g_j(x) > 0 \\ 0\ ;\ \ g_j(x) \le 0 \end{cases}.$$

Algorithm

The algorithm used in Reference [42] is as follows:

- 0: Set NP, F and CR. Initialize the iteration counter $k = 1$.
- 1: Generate the particles (parameter vectors) at random.
- 2: Apply the following procedure to all particles:
    - 2-1: Particle i, denoted by $\boldsymbol{x}_i^k$, selects three particles $(\boldsymbol{x}_{r_1}^k, \boldsymbol{x}_{r_2}^k, \boldsymbol{x}_{r_3}^k)$ at random, where $i \neq r_1 \neq r_2 \neq r_3$.
    - 2-2: *Mutation*: A new particle is generated by the mutation: $\boldsymbol{v}_i^k = \boldsymbol{x}_1^k + F(\boldsymbol{x}_2^k - \boldsymbol{x}_3^k)$.
    - 2-3: *Crossover*: A new particle $\boldsymbol{u}_i^k$ is generated by the crossover between $\boldsymbol{x}_i^k$ and $\boldsymbol{v}_i^k$.
    - 2-4: *Selection*: The objective function is evaluated at $\boldsymbol{x}_d^k$ and $\boldsymbol{u}_d^k$ and the particle i is updated according to the following criteria:

$$f(\boldsymbol{u}_i^k) \leq f(\boldsymbol{x}_i^k) \rightarrow \boldsymbol{x}_i^k = \boldsymbol{u}_i^k$$
$$f(\boldsymbol{u}_i^k) > f(\boldsymbol{x}_i^k) \rightarrow \boldsymbol{x}_i^k = \boldsymbol{x}_i^k$$

- 3: $k = k + 1$
- 4: If $k < k_{max}$, go to 2, else terminate.

In addition to 11 analytical benchmark problems with n varying between 2 and 10, Kitayama *et al* analyzed two structural constrained minimization examples (the optimal design of a spring ($n = 3$) and a truss topology optimization ($n = 28$)). As in [41], DE/rand/1/bin was used. The performance of the DE was compared to several other algorithms (Generalized Random Tunneling Algorithm (GRTA) [47], Particle Swarm Optimization (PSO) [36][37], GA [11], Distributed GA and Simulated Annealing) and was concluded to be competitive although GRTA outperformed DE on the topology benchmark.

26.13. Hybrid algorithms

As discussed earlier, the stochastic algorithms like the genetic algorithm (GA) and adaptive simulated annealing (ASA) are designed to find the global optimal solution. However, one of the most difficult aspects of using stochastic algorithms is to identify the correct stopping criterion. A defensive, but likely expensive, approach is to run an algorithm sufficiently long to ensure the global optimal solution. However, the speed of finding the global optimum can be significantly increased by combining the global optimizers with local gradient based optimization methods. This combination, referred to as a *hybrid* algorithm, is based on a very simple idea that the global optimizers reach the basin of the global optimum quickly i.e., they find very high quality solutions, but significant effort is then required to achieve small improvements for refining the solution. On the other hand, gradient based optimization methods like LFOPC can find an optimal solution very quickly when starting from a good solution. Thus, in LS-OPT, a global optimizer such as the GA or ASA is used to find a good starting solution followed by a single LFOPC run to converge to the global optimum. This approach has been found to be both effective and efficient for global optimization. The hybrid algorithms are available for both the GA and ASA options.

26.14. Visualization of the Pareto optimal frontier

Due to the complexity of visualizing the Pareto Optimal Frontier (POF) for high dimensional problems, methods to improve exploration of the Pareto set have been devised. Several methods have been implemented in LS-OPT. These methods are described below:

26.14.1. Trade-off plot

This is the simplest of all plot types. The user creates a scatter plot of different entities, mostly objective functions, in a 3-D space. One can also add fourth entity in the form of the color. An example of the Trade-Off plot in four-dimensional space is shown in Figure 26-8. A serious limitation of this plot type is its inability to simultaneously show more than four dimensions.

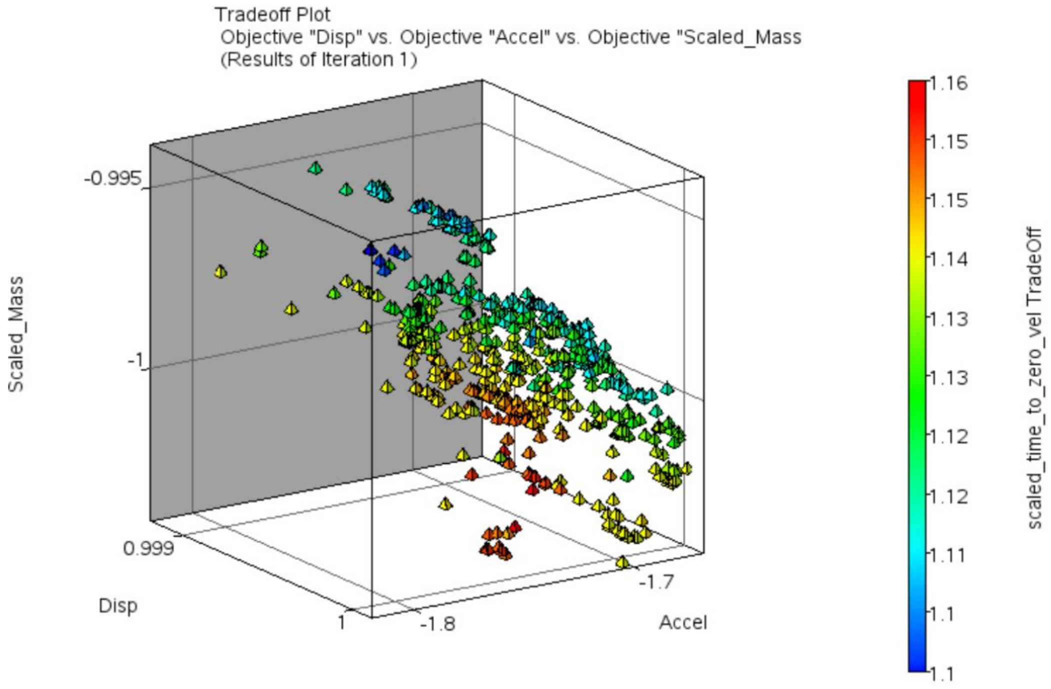


Figure 26-8: Trade-off plot shows all four objectives of Pareto optimal solutions.

26.14.2. Hyper-radial visualization (HRV)

HRV [38] is based on the minimization of the sum of squares of the normalized objective functions which allows the POF to be displayed in two dimensions. HRV is effectively a 2-dimensional mapping of the *n*-dimensional objective function space.

The mathematical form of the multi-objective optimization problem is as follows:

$$\text{Minimize } F(x) = [\, f_1(x),\ f_2(x), ...,\ f_n(x)] \text{ where } x = [x_1, x_2, ..., x_p\,]$$

subject to

$g_j(x) \le 0; \quad j = 1,2,\ldots,m$ inequality constraints

$h_k(x) \le 0; \quad k = 1,2,\ldots,l$ equality constraints

$x_i^l \le x_i \le x_i^u; \quad i = 1,2,\ldots,p$ side constraints

HRV can be seen as a conversion of the multi-objective optimization problem to a single objective optimization problem using the objective:

$$\frac{\sum_{i=1}^{s} W_i \tilde{F}_i^2 + \sum_{i=s+1}^{n} W_i \tilde{F}_i^2}{s}$$

subject to

$$\sum_{i=1}^{n} W_i = 1 \quad \text{and } W_i > 0$$

where *s* = *n/2* and the two additive components represent the objectives assigned to the two axes of the plot (see Figure 26-9). The case where *n* is an odd number is discussed below.

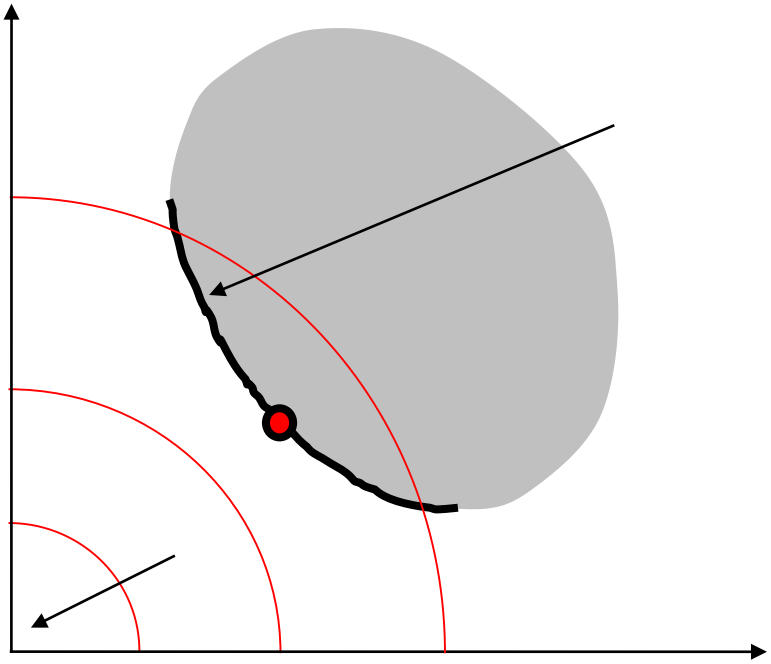


Figure 26-9: The Pareto frontier and indifference curves

The HRV method assumes that the set of Pareto points has already been computed and are available for display. First each objective function F_i is normalized to the range of the Pareto points. Normalization is done by using the lower and upper values of all the computed Pareto points to define the range for each objective.

$$\tilde{F}_i = \frac{F_i - F_{i\ \min}}{F_{i\ \max} - F_{i\ \min}} \quad i = 1,...,n \text{ where } \tilde{F}_i \in [0,1]$$

The coordinate $[F_{1\ \min}, F_{2\ \min}, ..., F_{n\ \min}]$ represents the *Utopian* point (see Figure 26-9), i.e. the point representing the minima of individual objectives. In the HRV representation, this point becomes the origin of the 2-dimensional plot. In addition to normalizing each objective function, the result of the Hyper-Radial Calculation (HRC) must also be normalized:

$$HRC = \sqrt{\frac{\sum_{i=1}^{n} \tilde{F}_i^2}{n}} \text{ where } HRC \in [0,1]$$

Now consider the *n*-objective sample data, corresponding to Pareto point *j* (of a total of *q* Pareto points). The objective functions are grouped into 2 sets controlled by the designer and an HRC value is computed for each group resulting in the coordinates HRC1 and HRC2. Thus *s* objectives are represented by HRC1 while *n-s* objectives are represented by HRC2. The two groups are therefore

Group 1:

$$[F_1, F_2, F_3, ..., F_s] \qquad HRC1 = \sqrt{\frac{\sum_{i=1}^{s} \tilde{F}_i^2}{s}}$$

Group 2:

$$[F_{s+1}, F_{s+2}, F_{s+3}, ..., F_n] \qquad HRC2 = \sqrt{\frac{\sum_{i=s+1}^{n} \tilde{F}_i^2}{n-s}}$$

The formulation is unbiased because the objectives can be grouped in an arbitrary way without sacrificing the unbiased visualization property. This means the radius originating in the Utopian point of any point is preserved irrespective of the objective grouping. The 'best' design is therefore considered to be the one closest to the Utopian point, i.e., the one with the smallest radius in the 2-dimensional plot.

The distance from the Utopian point is not the only criterion of how good a design is since a designer may also have a preference regarding the objectives. Preferences are incorporated by adding weights to the objectives represented in the HRC functions:

Group 1:

$$[F_1, F_2, F_3, ..., F_s] \qquad HRCW1 = \sqrt{\frac{\sum_{i=1}^{s} W_i \tilde{F}_i^2}{s}}$$

Group 2:

$$[F_{s+1}, F_{s+2}, F_{s+3}, ..., F_n] \qquad HRCW2 = \sqrt{\frac{\sum_{i=s+1}^{n} W_i \tilde{F}_i^2}{s}}$$

When (n-s < s) as is the case when, for instance, n is an odd number, (2s-n) dummy objective functions are added to normalize the visualization. This is to avoid producing an elliptical set of indifference curves. A dummy objective is a q-dimensional null vector, q being the number of Pareto points. The addition of such a dummy objective ensures the preservation of the indifference radius, so if the groupings are reselected, a particular Pareto point will move tangent to its current radius and therefore maintain its level of indifference.

26.14.3. Parallel co-ordinate plot (PCP)

The parallel coordinate plot shows all entities of a design by a line such that any number of entities can be simultaneously shown. An example of PCP is shown in Figure 26-10. The user can move the sliders on each entity to filter-out the undesired values and screen the objectives. The screened out solutions are shown as the grey-lines in Figure 26-10.

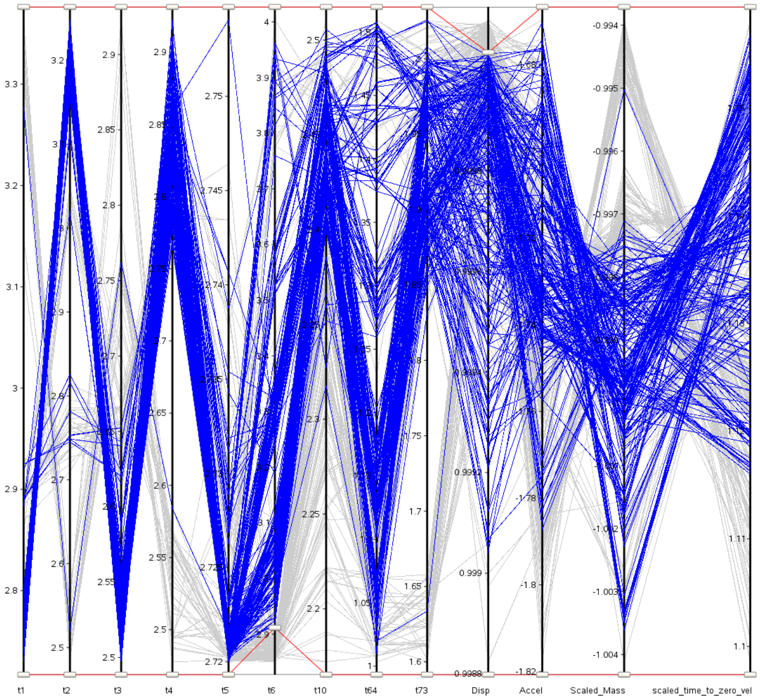


Figure 26-10: Parallel coordinate plot shows objectives and design variables of all points on the Pareto front.

26.14.4. Self organizing maps (SOM)

Self organizing map [39], proposed by Kohonen in early 1980s, is a very powerful technique to represent n-dimensional data in two-dimensional space. The designs that are close in the n-dimensional space remain close to each other in the mapped space as well. These maps allow the user to explore the solution space in many dimensions simultaneously. Figure 26-11 shows an example of a self organizing map. One can simultaneously see design objectives, variables, and constraints.

By default, the network is trained with 12 rows and 9 columns i.e., 108 nodes but the number of units can be controlled in the viewer GUI. With a trained SOM, one can show the following:

1. Component maps: Each component map shows one entity e.g., variables, responses, etc. One can simultaneously plot different component maps to see the variation in data in different regions.
2. D-matrix: This map shows the average distance from the neighboring units in the maps. This feature helps identify sparse sections in the data.
3. U-matrix: The U-matrix map shows the actual distances between the two neighboring units.
4. C-matrix: This plot illustrates the density associated with each SOM unit. For a well trained network, the C-matrix plot would also identify sparsely distributed data.0.

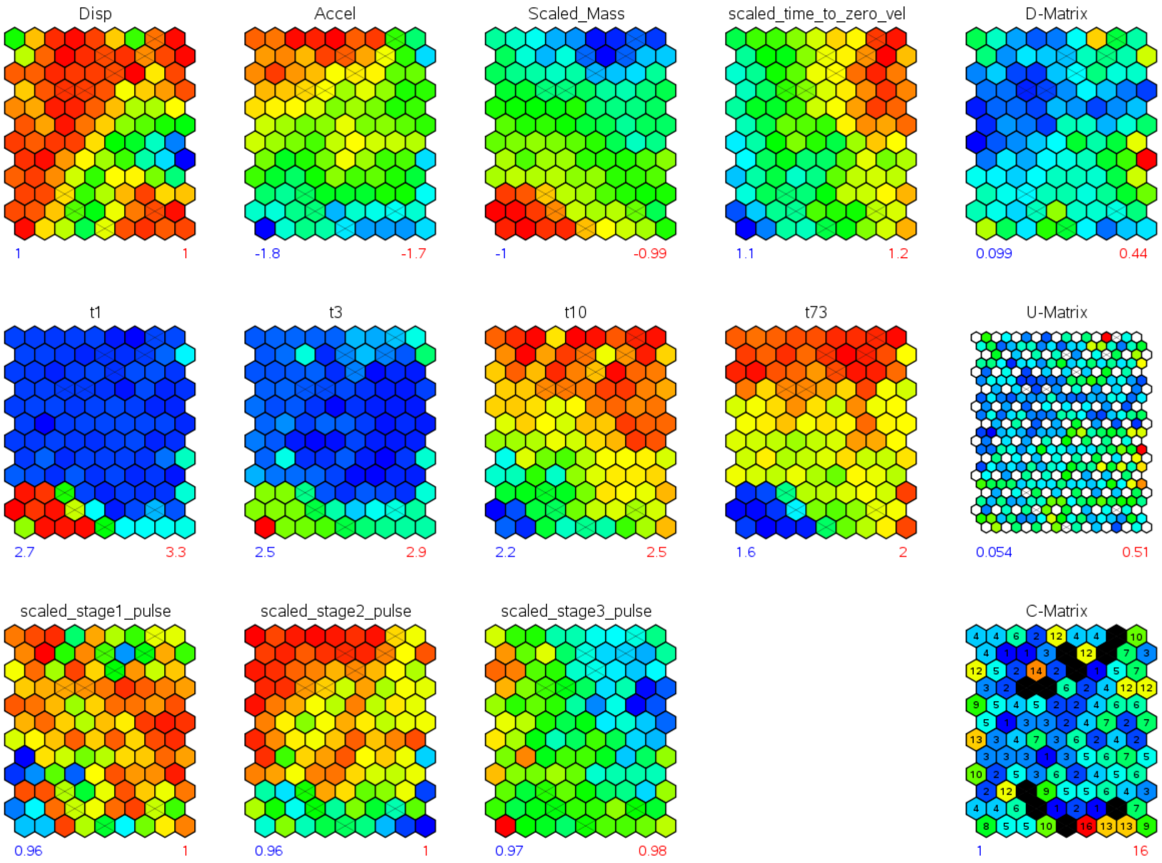


Figure 26-11: Self organizing maps display design objectives, variables, and constraints on the Pareto front.

26.15. Performance metrics for multi-objective optimization

Since multi-objective optimization results in a set of solutions, it requires special metrics to assess the convergence to the Pareto optimal front, diversity on the front, and the spread of the front. While the users can get detailed information on performance metrics for multiobjective optimization problems elsewhere [17], a few metrics available in LSOPT are described here.

26.15.1. Number of nondominated points

This is the number of solutions in the archive of all nondominated solutions at any generation. Usually a higher number of nondominated points are achieved when convergence is good.

26.15.2. Spread

The spread of the front is calculated as the diagonal of the largest hypercube in the function space that encompassed all points. A large spread is desired to find diverse trade-off solutions. The spread measure is derived using the extreme solutions making it susceptible to the presence of a few isolated points that could artificially improve the spread metric. An equivalent criterion might be the volume of such a hypercube.

26.15.3. Standard deviation of crowding distance

This complimentary criterion (to the spread metric) detects the presence of poorly distributed solutions by estimating how uniformly the points are distributed in the Pareto optimal set. This metric is defined as,

$$\delta = \sum_{i=1}^{N} \frac{\left| d_i - \bar{d} \right|}{N}; \bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i .$$

where d_i is the crowding distance of the solution in the function or variable space. The boundary points are assigned a crowding distance of twice the distance to the nearest neighbor. A small value of the uniformity measure is desired to achieve a good distribution of points.

26.15.4. Min/Max of objectives

This represents the range of individual objectives. A wide range represents more choices for the designer.

26.15.5. Hypervolume

A dominated hypervolume metric tries to simultaneously estimate the convergence and spread characteristics by computing the union of the volume between the optimal solutions and a reference point. For practical purposes, the nadir point of all solutions is used as the reference point.

While all the above metrics are obtained on a single set of solutions, the following performance metrics are obtained by comparing multiple sets of solutions. These metrics are helpful in determining the convergence. In LSOPT, the set of non-dominated solutions separated by Δ generations (archive A_i and $A_{i-\Delta}$) are compared and the following metrics are reported. Δ is called generation interval.

26.15.6. Number of common points

This is the number of solutions that exist in both sets A_i and $A_{i-\Delta}$. A large number of common points is indicative of the high quality of solutions. The set of common solutions is represented as,

$$Q = \{a_i : a_{i-\Delta} = a_i\}, a_{i-\Delta} \in A_{i-\Delta}, a_i \in A_i .$$

and *n(Q)* is the size of set *Q*. This is a particularly good metrics when a large generation interval is used.

26.15.7. Number of new nondominated solutions

This metrics denotes the number of nondominated solutions that were evolved during the current generation interval. The set of such solutions is represented as,

$$Q = \{a_i : a_{i-\Delta} \neq a_i\}, a_{i-\Delta} \in A_{i-\Delta}, a_i \in A_i.$$

A large number of new solutions relative to the total archive size indicates that the new solutions are still being evolved and hence convergence is not yet achieved.

26.15.8. Number of old dominated solutions *n*(*Q*)

This metrics denotes the number of nondominated solutions in the older archive $A_{i-\Delta}$ that were dominated by the solutions in the current archive A_i. The set of dominated solutions is,

$$Q = \{a_{i-\Delta} : a_{i-\Delta} \prec a_i\}, a_{i-\Delta} \in A_{i-\Delta}, a_i \in A_i.$$

A large number of dominated solutions represents significant evolution.

26.15.9. Consolidation ratio

This represents the fraction of archive A_i that has evolved up to the i-Δ^{th} generation. This is computed as the ratio of the number of members in archive $A_{i-\Delta}$ that are also present in the archive A_i (non-dominated solutions) to the size of archive A_i. Mathematically,

$$CR = \frac{n(S)}{n(A_i)}; S = \{a_{i-\Delta} : a_{i-\Delta} \not\prec a_i\}, a_{i-\Delta} \in A_{i-\Delta}, a_i \in A_i.$$

This metric represents the proportion of potentially converged solutions in the archive. In the early phase of a MOEA simulation, a large fraction of the non-dominated solutions in the archive $A_{i-\Delta}$ would be dominated by the solutions in archive A_i due to evolution, thus resulting in a small fraction of surviving solutions i.e., small value of the consolidation ratio. However, significantly better solutions evolve in the later phases such that a large proportion of solutions in the archive $A_{i-\Delta}$ remain non-dominated with respect to new solutions leading to a high consolidation ratio. In the limiting case, the consolidation ratio approaches one.

26.15.10. Improvement ratio

This represents the fraction of archive $A_{i-\Delta}$ dominated by the new solutions in archive A_i. This is computed as the ratio of the number of members in archive $A_{i-\Delta}$ that are dominated by the solutions in archive A_i (dominated solutions) to the size of archive A_i. Mathematically,

$$IR = \frac{n(Q)}{n(A_i)}; Q = \{a_{i-\Delta} : a_{i-\Delta} \prec a_i\}, a_{i-\Delta} \in A_{i-\Delta}, a_i \in A_i.$$

The archive A_i includes all non-dominated members of archive $A_{i-\Delta}$ so no member of the archive A_i is dominated. The improvement ratio quantifies the extent of improvement in the quality of evolved solutions. This metric has a high value in the early phase of simulation which gradually converges to zero when convergence is achieved.

More information about these performance metrics can be obtained from [40].

26.16. Discrete optimization

26.16.1. Discrete variables

Discrete variables can have only distinct values; for example, the variable can be a plate thickness having allowable values 1.0, 2.0, 2.5, and 4.5.

26.16.2. Discrete optimization

A very basic method of discrete optimization would be simply evaluating all possible design and selecting the best one. This is not feasible for the general case; consider for example that 30 design variables with variables having 5 possible values of the design variable will result in 10^{21} different designs. Evaluating all the possible designs is therefore not computationally feasible. Note that 30 design variables describe a design space with 10^9 quadrants, so finding the quadrant containing the optimum design is a hard problem. The quadrant containing the optimal design can be found using a gradient based search direction, but discrete optimization problems are not convex, which means that gradient based search directions may lead to local optima. The LS-OPT discrete optimization methodology using LFOPC therefore use gradient based search in conjunction with random search methods. The optimal design found in this manner, cannot be shown to be uniquely the global optimum, but is considered the (practical) optimum because it is known that it is highly unlikely that a better design will be found.

The cost of the discrete optimization is kept affordable by doing the optimization using the values from a response surface approximation. The accuracy of the response surface or metamodel is improved using a sequential strategy described in a later section.

26.16.3. Mixed-discrete optimization

The discrete variables can be used together with continuous variables. This is called mixed-discrete optimization.

The steps followed to compute the mixed-discrete optimum are:

1. Consider all the discrete variables to be continuous and optimize using the gradient based design optimization package. This continuous optimum found is used as the starting design in the next phase.
2. Discrete optimization is done considering only the discrete variables with the continuous variables frozen at the values found in the previous phase.
3. Continuous optimization is done considering only the continuous variables and with the discrete variables frozen at the values found in the previous phase.

26.16.4. Discrete optimization algorithm: genetic algorithm

A GA (genetic algorithm, Section 26.9) is used to do the discrete optimization. A GA mimics the evolutionary process selecting genetic strings. In a GA, the design variable values are coded up into data structure similar to genetic strings. New generations of designs are obtained by combining portions of the genetic strings of the previous generation of designs. Designs that have relatively better values of the objective function have a better chance to contribute a portion of its genetic string to the next generation.

26.16.5. Objective function for discrete optimization

The discrete optimization algorithm used can only consider an objective function (no constraints); the constraints specified by the user are therefore incorporated into this objective function. The resulting objective function has two different behaviors:

1. *A feasible design exists*. In this case all infeasible designs (those violating the constraints) are simply rejected, and only feasible designs are considered inside the optimization algorithm. The objective function used is simply that specified by the user.
2. *A feasible design does not exist*. If the search for the best feasible designs fails due to a lack of feasible designs, then a search is done for the least infeasible constraint. The objective function is a scaled sum of the constraint violations: $\sum \frac{\left|\text{constraint}_{i} - \text{Bound}_{i}\right|}{\left|\text{Bound}_{\text{i}}\right|}$ with the summation done over all the violated constraints.

26.16.6. Sequential strategy

The discrete and the mixed-discrete optimization are done using the response values from the response surfaces or metamodels. The accuracy of the response surface or metamodels is therefore very important. The accuracy of the metamodels is improved by a sequential response surface method (SRSM) (see Section 26.6), in which the size of the subregion over which the designs are evaluated are reduced until convergence. Reducing the size of the subregion is the best known method of obtaining accuracy for optimizing using metamodels.

Discrete optimization introduces the concern that a discrete variable value may not be on the edge of the subregion selected by the SRSM algorithm. The SRSM algorithm was therefore modified to use closest discrete values outside the subregion. This implies that the subregion cannot be smaller than the distance between two successive discrete values.

26.17. Summary of the optimization process

The following tasks may be identified in the process of an optimization cycle using response surfaces.

Table 26-1: Summary of optimization process

Item	Input	Output
DOE	Location and size of the subregion in the design space. The experimental design desired. An approximation order. An affordable number of points.	Location of the experimental points.
Simulation	Location of the experimental points. Analysis programs to be scheduled.	Responses at the experimental points.
Build response surface	Location of the experimental points. Responses at the experimental points. Function types to be fitted.	The approximate functions (response surfaces). The goodness-of-fit of the approximate functions at the construction points.
Check adequacy	The approximate functions (response surfaces). The location of the check points. The responses at the check points.	The goodness-of-fit of the approximate functions at the check points.
Optimization	The approximate functions (response surfaces). Bounds on the responses and variables.	The approximate optimal design. The approximate responses at the optimal design. Pareto optimal curve data.

Two approaches may be taken:

26.17.1. Convergence to an optimal point

- First-order approximations.

  Because of the absence of curvature, it is likely that perhaps 5 to 10 iterations may be required for convergence. The first-order approximation method turns out to be robust thanks to the sequential approximation scheme that addresses possible oscillatory behavior. Linear approximations may be rather inaccurate to study trade-off, i.e., in general they make poor global approximations, but this is not necessarily true and must be assessed using the error parameters.

- Second-order approximations.

  Due to the consideration of curvature, a sequential quadratic response surface method is likely to be more robust, but can be more expensive, depending on the number of design variables.

- Other approximations.

Neural networks (Section 24.1) and Radial Basis Function networks (Section 24.1.3) provide good approximations when many design points are used. A suggested approach is to start the optimization procedure in the full design space, with the number of points at least of the order of the minimum required for a linear approximation. To converge to an optimum, use the iterative scheme with domain reduction as with any other approximations, but choose to update the experimental design and response surfaces after each iteration (this is the default method for non-polynomial approximations). The metamodel will be built using the total number of points.

See Section 26.5 on sequential strategies for optimization and design exploration.

26.18. REFERENCES

[1] Forsberg, J. Simulation Based Crashworthiness Design – Accuracy Aspects of Structural optimization using Response Surfaces. Thesis No. 954. Division of Solid Mechanics, Department of Mechanical Engineering, Linköping University, Sweden, 2002.

[2] Luenberger, D.G. Linear and Nonlinear Programming. Second Edition. Addison Wesley, 1984.

[3] Arora, J.S. Sequential linearization and quadratic programming techniques. In Structural Optimization: Status and Promise, Ed. Kamat, M.P., AIAA, 1993.

[4] Thanedar, P.B., Arora, J.S., Tseng, C.H., Lim, O.K., Park, G.J. Performance of some SQP algorithms on structural design problems. International Journal for Numerical Methods in Engineering, 23, pp. 2187-2203, 1986.

[5] Snyman, J.A. The LFOPC leap-frog algorithm for constrained optimization. Comp. Math. Applic., 40, pp. 1085-1096, 2000.

[6] Barthelemy, J.-F. M. Function Approximation. In Structural Optimization: Status and Promise, Ed. Kamat, M.P., 1993.

[7] Box., G.E.P., Draper, N.R. Empirical Model Building and Response Surfaces. Wiley, New York, 1987.

[8] Hajela, P., Berke L. Neurobiological computational models in structural analysis and design. Proceedings of the 31st AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April, 1990.

[9] Snyman, J.A. An improved version of the original leap-frog dynamic method for unconstrained minimization LFOP1(b). Appl. Math. Modelling, 7, pp. 216-218, 1983.

[10] Holland, J.H., Adaptation in Natural and Artificial Systems. MIT Press, 1992.

[11] Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, 1989.

[12] Deb, K., Agrawal, R.B., Simulated binary crossover for continuous search space. Complex Systems, 9, 115-148, 1995.

[13] Deb, K., Beyer, H.-G., Self adaptive genetic algorithms with simulated binary crossover. Evolutionary Computation Journal, 9(2), 197-221, 2001.

[14] Eshelman, L.J., Schaffer, J.D., Real-coded genetic algorithms and interval-schemata. In Foundations of Genetic Algorithms-2. San Mateo, CA: Morgan Kaufman, 187-202, 1993.

[15] Stander, N., Craig, K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization, Engineering Computations, 19(4), pp. 431-450, 2002.

[16] Stander, N. Goel, T. Metamodel sensitivity to sequential sampling strategies in crashworthiness design. Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,Victoria, British Columbia, Canada, Sep 10-12, 2008. Submitted.

[17] Deb, K., Multiobjective Optimization using Evolutionary Algorithms, Wiley Chichester UK, 2001.

[18] Coello, C.A.C., Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), Springer, 2007.

[19] Miettinen, K.M., Nonlinear Multi Objective Optimization, Kluwer, 1999.

[20] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transaction on Evolutionary Computation, 6(2), 181-197, 2002. an earlier version was presented in Parallel Problem Solving from Nature VI Conference, Paris, France, Sept 2000.

[21] Goel, T., Vaidyanathan, R., Haftka, R.T., Queipo, N.V., Shyy, W., Tucker, K., Response surface approximation of Pareto optimal front in multi-objective optimization. Computer Methods in Applied Mechanics and Engineering,

196(4-6), 879-893, 2007. (also presented at 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, September 2004, Albany, NY).

[22] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., Equation of state calculations by fast computing machines, Journal of Chemical Physics, 21(6), pp. 1087-1092, 1953.

[23] Kirkpatrick, S. Gelatt, C.D., Vecchi, M.P. Science, 220, pp. 671-680, 1983.

[24] Pincus, M., A Monte Carlo method for the approximate solution of certain types of constrained optimization problems, Operations Research, 18, pp. 1225-1228, 1970.

[25] Cerny, V., A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, Report, Bratislave, Czech Republic, Comenius University.

[26] Morris, M., Mitchell, T. Exploratory design for computer experiments. Journal of Statistical Planning Inference, 43, pp. 381-402, 1995.

[27] Ingber, L. Very fast simulated re-annealing, Mathematical Computer Modeling, 12, pp. 967-983, 1989.

[28] Ingber, L., Adaptive simulated annealing (ASA), [ftp.alumni.caltech.edu: /pub/ingber/ASA.tar.gz], Lester Ingber Research, McLean VA, 1993.

[29] Schuur, P.C. Classification of acceptance criteria for the simulated annealing algorithm. Mathematics of Operations Research, 22(2), pp.266-275, 1997.

[30] Ingber, L., Adaptive simulated annealing: Lessons learned, Journal of Control and Cybernetics, 25, pp. 33-54, 1996.

[31] Basu, A., Frazer, L.N. Rapid determination of the critical temperature in simulated annealing inversion, Science, pp. 1409-1412, 1990.

[32] Geman, S., Geman, D., Stochastic relaxation, Gibbs distribution, and the Bayesian restoration in images, IEEE Transactions of Pattern Analysis and Machine Intelligence, 6(6), pp. 721-741, 1984.

[33] Szu, H., Hartley, R., Fast simulated annealing, Physics Letters A, 122 (3-4), pp. 157-162, 1987.

[34] van Laarhoven, P.J.M., Aarts, E.H.L., Simulated annealing: Theory and applications, Dordrecht, The Netherlands, D. Reidel, 1987.

[35] Bounds, D.G. New optimization methods from physics and biology. Nature, 329, pp.215-218, 1987.

[36] Kennedy J, Eberhart RC, “Particle swarm optimization”, Proceedings of the IEEE International Conference on Neural Networks, Perth Australia, Vol. 4, pp. 1942-1948, 1995.

[37] Schutte JF, Applications of Parallel Global Optimization to Mechanics Problems, PhD Thesis, University of Florida, Gainesville, 2005.

[38] Chiu P-W, Bloebaum CL, Hyper-Radial Visualization (HRV) with weighted preferences for multi-objective decision making. Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 10-12 September 2008, Victoria, British Columbia, Canada.

[39] Kohonen T, Self-Organizing Maps, 3rd edition, Springer, Heidelberg, 2001.

[40] Goel T, Stander N, A non-dominance based online stopping criterion for multi-objective evolutionary algorithms, International Journal of Numerical Methods in Engineering, doi: 10.1002/nme.2009.

[41] Storn, R. and Price, K. Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11, 341-359, 1997.

[42] Kitayama, S., Masao, A. and Yamazaki, K. Differential evolution as the global optimization technique and its application to structural optimization. Applied Soft Computing, 11, 3792-3803, 2011.

[43] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. Numerical Recipes in C, Cambridge University Press, 1992.

[44] Mühlenbein, H. and Schlierkamp-Vosen. Predictive models for the Breeder genetic Algorithm, I. Continuous parameter optimizations, Evolutionary Computation, 1 (1), 25-49, 1993.

[45] Voigt, H.-M. Soft Genetic Operators in Evolutionary Computation, Evolution and Bio-computation, Lecture Notes in Computer Science 899, Springer, Berlin, pp. 123-141.

[46] Aluffi-Pentini, F., Parisi, V. and Zirilli, F. Global Optimization and Stochastic Differential Equations, Journal of Optimization Theory and Applications, 47 (1), 1-16.

[47] Kitayama, S. and Yamazaki, K. Generalized random tunneling algorithm for continuous design variables. Trans. ASME J. Mech. Des. 127, 408-414, 2005.

[48] Kitayama, S., Arakawa, M, and Yamazaki, K. Discrete Differential Evolution for Mixed-Discrete Non-Linear Problems. Journal of Civil Engineering and Architecture, 6, 594-605, 2012.

[49] Jones, D.R., Schonlau, M. and Welch, W.J., Efficient global optimization of expensive black-box functions. Journal of Global optimization, 13(4), pp.455-492, 1998.

[50] Schonlau, M.. Computer experiments and global optimization, PhD Thesis, University of Waterloo, 1997

[51] Sasena, M.J., Papalambros, P. and Goovaerts, P. Exploration of metamodeling sampling criteria for constrained global optimization. Engineering optimization, 34(3), pp.263-278, 2002.

[52] Basudhar, A., Dribusch, C., Lacaze, S. and Missoum, S. Constrained efficient global optimization with support vector machines. Structural and Multidisciplinary Optimization, 46(2), pp.201-221, 2012.

[53] Lee, H., Gramacy, R., Linkletter, C. and Gray, G. Optimization subject to hidden constraints via statistical emulation. Pacific Journal of Optimization, 7(3), pp.467-478, 2011.

[54] Schonlau, M., Welch, W.J. and Jones, D.R., 1998. Global versus local search in constrained optimization of computer models. Lecture Notes-Monograph Series, pp.11-25.

[55] Ginsbourger, D., Le Riche, R. and Carraro, L.. Kriging is well-suited to parallelize optimization. In Computational intelligence in expensive optimization problems (pp. 131-162). Springer, Berlin, Heidelberg, 2010

27. Applications of Optimization

27.1. Multicriteria design optimization

A typical design formulation is somewhat distinct from the standard formulation for mathematical optimization (Eq. 2.3). Most design problems present multiple objectives, design targets and design constraints. There are two ways of solving multicriteria design optimization problems.

The first method, discussed in Section 26.10, focused on finding multiple trade-offs, known as Pareto optimal solutions, using multi-objective genetic algorithms. The advantage of this method is that one can find many trade-off designs and the designer does not have to a priori determine the preference structures.

In the second method, the standard mathematical programming problem is defined in terms of a single objective and multiple constraints. The standard formulation of Eq. (23-1) has been modified to represent the more general approach as applied in LS-OPT.

Minimize the function

$$p[\boldsymbol{f}(\mathbf{x})] \qquad (27\text{-}1)$$

subject to the inequality constraint functions

$$L_j \leq g_j(\mathbf{x}) \leq U_j; \quad j = 1,2,\ldots,m.$$

The preference function p can be formulated to incorporate target values of objectives.

Two methods for achieving this are given:

27.1.1. Euclidean distance function

Designs often contain objectives that are in conflict so that they cannot be achieved simultaneously. If one objective is improved, the other deteriorates and *vice versa*. The preference function $p[\boldsymbol{f}(\boldsymbol{x})]$ combines various objectives f_i. The Euclidean distance function allows the designer to find the design with the smallest distance to a specified set of target responses or design variables:

$$p = \sqrt{\sum_{i=1}^{p} W_i \left[\frac{f_i(\mathbf{x}) - F_i}{\Gamma_i} \right]^2} \qquad (27\text{-}2)$$

The symbols F_i represent the target values of the responses. A value Γ_i is used to normalize each response *i*. Weights W_i are associated with each quantity and can be chosen by the designer to convey the relative importance of each normalized response.

27.1.2. Maximum distance

Another approach to target responses is by using the maximum distance to a target value

$$p = \max_i \left[\frac{|f_i(\boldsymbol{x}) - F_i|}{|\Gamma_i|} \right] \qquad (27\text{-}3)$$

This form belongs to the same category of preference functions as the Euclidean distance function [1] and is referred to as the Tchebysheff distance function. A general distance function for target values F_i is defined as

$$p = \left[\sum_{i=1}^{p} \left(\frac{|f_i(\boldsymbol{x}) - F_i|}{|\Gamma_i|} \right)^r \right]^{\frac{1}{r}} \qquad (27\text{-}4)$$

with $r = 2$ for the Euclidean metric and $r \to \infty$ for the min-max formulation (Tchebysheff metric).

The approach for dealing with the Tchebysheff formulation differs somewhat from the explicit formulation.

The alternative formulation becomes:

$$\text{Minimize } \; \text{e} \qquad (27\text{-}5)$$

subject to

$$\frac{F_i}{\Gamma_i} - (1 - \alpha_{jL})e \le \frac{f_i(\mathbf{x})}{\Gamma_i} \le \frac{F_i}{\Gamma_i} + (1 - \alpha_{jL})e; \quad i = 1,\ldots,p, \quad j = 1,\ldots,m$$
$$e \ge 0.$$

In the above equation, Γ_i is a normalization factor, *e* represents the constraint violation or target discrepancy and α represents the strictness factor. If $\alpha = 0$, the constraint is slack (or soft) and will allow violation. If $\alpha = 1$, the constraint is strict (or hard) and will not allow violation of the constraint.

The effect of distinguishing between strict and soft constraints on the above problem is that the maximum violation of the soft constraints is minimized. Because the user is seldom aware of the feasibility status of the design problem at the start of the investigation, the solver will automatically solve the above problem first to find a feasible region. If the solution to *e* is zero (or within a small tolerance) the problem has a feasible region and the solver will immediately continue to minimize the design objective using the feasible point as a starting point.

A few points are notable:

1. The variable bounds of both the region of interest and the design space are always hard. This is enforced to prevent extrapolation of the response surface and the occurrence of impossible designs.
2. Soft constraints will always be *strictly* satisfied if a feasible design is possible.
3. If a feasible design is not possible, the most feasible design will be computed.
4. If feasibility must be compromised (there is no feasible design), the solver will automatically use the slackness of the soft constraints to try and achieve feasibility of the hard constraints. However, even when allowing soft constraints, there is always a possibility that some hard constraints must still be violated. In this case, the variable bounds could be violated, which is highly undesirable as the solution will lie beyond the region of interest and perhaps beyond the design space. If the design is reasonable, the optimizer remains robust and finds such a compromise solution without terminating or resorting to any specialized procedure.

Soft and strict constraints can also be specified for search methods. If there are feasible designs with respect to hard constraints, but none with respect to all the constraints, including soft constraints, the most feasible design will be selected. If there are no feasible designs with respect to hard constraints, the problem is ‘hard-infeasible’ and the optimization terminates with an error message.

In the following cases, the use of the Min-Max formulation can be considered:

1. Minimize the maximum of several responses, e.g. minimize the maximum knee force in a vehicle occupant simulation problem. This is specified by setting both the knee force constraints to have zero upper bounds. The violation then becomes the actual knee force.
2. Minimize the maximum design variable, e.g. minimize the maximum of several radii in a sheet metal forming problem. The radii are all incorporated into composite functions, which in turn are incorporated into constraints which have zero upper bounds.
3. Find the most feasible design. For cases in which a feasible design region does not exist, the user may be content with allowing the violation of some of the constraints, but is still interested in minimizing this violation.

27.2. Multidisciplinary design optimization

There is increasing interest in the coupling of other disciplines into the optimization process, especially for complex engineering systems like aircraft and automobiles [2]. The aerospace industry was the first to embrace multidisciplinary design optimization (MDO) [3], because of the complex integration of aerodynamics, structures, control and propulsion during the development of air- and spacecraft. The automobile industry has followed suit [4]. In [4], the roof crush performance of a vehicle is coupled to its Noise, Vibration and Harshness (NVH) characteristics (modal frequency, static bending and torsion displacements) in a mass minimization study.

Different methods have been proposed when dealing with MDO. The conventional or standard approach is to evaluate all disciplines simultaneously in one integrated objective and constraint set by applying an optimizer to the multidisciplinary analysis (MDA), similar to that followed in single-discipline optimization. The standard method has been called multidisciplinary feasible (MDF), as it maintains feasibility with respect to the MDA, but as it does not imply feasibility with respect to the disciplinary constraints, is has also been called fully integrated optimization (FIO). A number of MDO formulations are aimed at decomposing the MDF problem. The choice of MDO formulation depends on the degree of coupling between the different disciplines and the ratio of shared to total design variables [5]. It was decided to implement the MDF formulation in this version of LS-OPT as it ensures correct coupling between disciplines albeit at the cost of seamless integration being required between different disciplines that may contain diverse simulation software and different design teams.

In LS-OPT, the user has the capability of assigning different variables, experimental designs and job specification information to the different solvers or disciplines. The file locations in Version 2 have been altered to accommodate separate `Experiments`, `AnalysisResults` and `DesignFunctions` files in each solver's directory. An example of job-specific information is the ability to control the number of processors assigned to each discipline separately. This feature allows allocation of memory and processor resources for a more efficient solution process.

Refer to the user's manual (Section 19.3) for the details of implementing an MDO problem. There is one crashworthiness-modal analysis case study in the examples chapter (Section 20.5).

27.3. System Identification

System identification, which includes material parameter identification, is discussed in Chapter 28.

27.4. Worst-case design

Worst-case design involves minimizing an objective with respect to certain variables while maximizing the objective with respect to other variables. The solution lies in the so-called saddle point of the objective function and represents a worst-case design. This definition of a worst-case design is different to what is sometimes referred to as min-max design, where one multi-objective component is minimized while another is maximized, both with respect to the same variables.

There is an abundance of examples of worst-case scenarios in mechanical design.

One class of problems involves minimizing design variables and maximizing case or condition variables. One example in automotive design is the minimization of head injury with respect to the design variables of the interior trim while maximizing over a range of head orientation angles. Therefore the worst-case design represents the optimal trim design for the worst-case head orientation. Another example is the minimization of crashworthiness-related criteria (injury, intrusion, etc.) during a frontal impact while maximizing the same criteria for a range of off-set angles in an oblique impact situation.

Another class of problems involves the introduction of *uncontrollable* variables $z_i, i = 1,...,n$ in addition to the *controlled* variables $y_j, j = 1,...,m$. The controlled variables can be set by the designer and therefore optimized by the program. The uncontrollable variables are determined by the random variability of manufacturing processes, loadings, materials, etc. Controlled and uncontrollable variables can be independent, but can also be associated with one another, i.e. a controlled variable can have an uncontrollable component.

The methodology requires three features:

1. The introduction of a constant range ρ of the region of interest for the uncontrollable variables. This constant represents the *possible variation* of each uncontrollable parameter or variable. In LS-OPT

this is introduced by specifying a lower limit on the range as being equal to the initial range ρ. The lower and upper bounds of the design space are set to $\pm\rho/2$ for the uncontrollable variables.

2. The controlled and uncontrollable variables must be separated as minimization and maximization variables. The objective will therefore be minimized with respect to the controlled variables and maximized with respect to the uncontrollable variables. This requires a special flag in the optimization algorithm and the formulation of Equation (26-1) becomes:

$$\min_{y}\left\{\max_{z} f(\boldsymbol{y},\boldsymbol{z})\right\}; y \in \Re^{p}, z \in \Re^{q}$$

   subject to

   $g_j(\boldsymbol{y},\boldsymbol{z}) \leq 0; j = 1,2,\ldots,l.$

   The algorithm remains a minimization algorithm but with modified gradients:

$$\nabla_{y}^{\mathrm{mod}} := \nabla\mathbf{y},$$

$$\nabla_{z}^{\mathrm{mod}} := -\nabla\mathbf{z}.$$

   For a maximization problem the min and max are switched.

3. The dependent set (the subset of y and z that are dependent on each other) $x = y + z$ must be defined as input for each simulation, e.g. if the manufacturing tolerance on a thickness is specified as the uncontrollable component, it is defined as a variation added to a mean value, i.e. $t = t_{\mathrm{mean}} + t_{\mathrm{deviation}}$, where t is the dependent variable.

27.5. REFERENCES

[1] Daberkow, D.D. Mavris, D.N. An investigation of metamodeling techniques for complex systems design. Symposium on Multidisciplinary Analysis and Design, Atlanta, October 2002.

[2] Lewis, K., Mistree, F. The other side of multidisciplinary design optimization: accommodating a mutiobjective, uncertain and non-deterministic world. Engineering Optimization, 31, pp. 161-189, 1998.

[3] Simpson, T.W. A Concept Exploration Method for Product Family Design. Ph.D. Thesis, Georgia Institute of Technology, 1998.

[4] Sobieszczanski-Sobieski, J., Kodiyalam, S., Yang, R-.J. Optimization of car body under constraints of noise, vibration, and harshness (NVH), and crash. AIAA Paper 2000-1521, 2000.

[5] Zang, T.A., Green, L.L., Multidisciplinary design optimization techniques: Implications and opportunities for fluid dynamics research, AIAA Paper 99-3798, 1999.

28. System and Material Identification

28.1. Introduction

System identification is a general term used to describe the mathematical tools and algorithms that build dynamical models such as systems or processes from measured data. The methodology used in LS-OPT consists of a nonlinear regression procedure to optimize the parameters of a system or material. This procedure minimizes the errors with respect to given experimental results. Two formulations for system identification can be used. The first uses the curve matching metrics as the minimization objective, while the second, the Min-Max formulation, uses the auxiliary problem formulation to minimize the maximum residual. The curve matching approach is commonly used for system identification and has been automated using a single command. Four curve matching measures are available, namely Euclidean (also known as Mean Squared Error or MSE), Partial Curve Mapping, Discrete Fréchet distance, and Dynamic Time Warping. The formulations are outlined below.

The application of optimization to system identification is demonstrated in Chapter 21.

28.2. Similarity measures

28.2.1. Ordinate-based Curve Matching

Figure 28-1 shows a graph containing curve $f(\mathbf{x},z)$ and points $G_p(z)$. The points can be interconnected to form a curve $G(z)$. f is a computed response curve (e.g. stress or force history) computed at a point $\mathbf{x}$ in the parameter space. The variables $\mathbf{x}$ represent unknown parameters in the model. System (e.g. automotive airbag or dummy model) or material constants are typical of parameters used in constructing finite element models. The independent state variable z can represent time, but also any other response type such as strain or deformation. The target curve G is constant with respect to $\mathbf{x}$ and typically represents a test result (e.g. stress vs. strain or force vs. deformation). f may not be readily available from the analysis code if z does not represent time. In this case f must first be constructed using a “crossplot” feature (see Section 6.4.2) and the curve $z(t)$ to obtain a plot that is comparable to G. Each function $f(x,z_p)$ is internally represented by a response surface so that a typical curve $f(x,z)$ is represented by P internal response surfaces.

In Figure 28-1, seven regression points are shown. The residuals at these points are combined into a Mean Squared Error norm:

$$\varepsilon = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\mathbf{x}) - G_p}{s_p}\right)^2 = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{e_p(\mathbf{x})}{s_p}\right)^2 \tag{28-1}$$

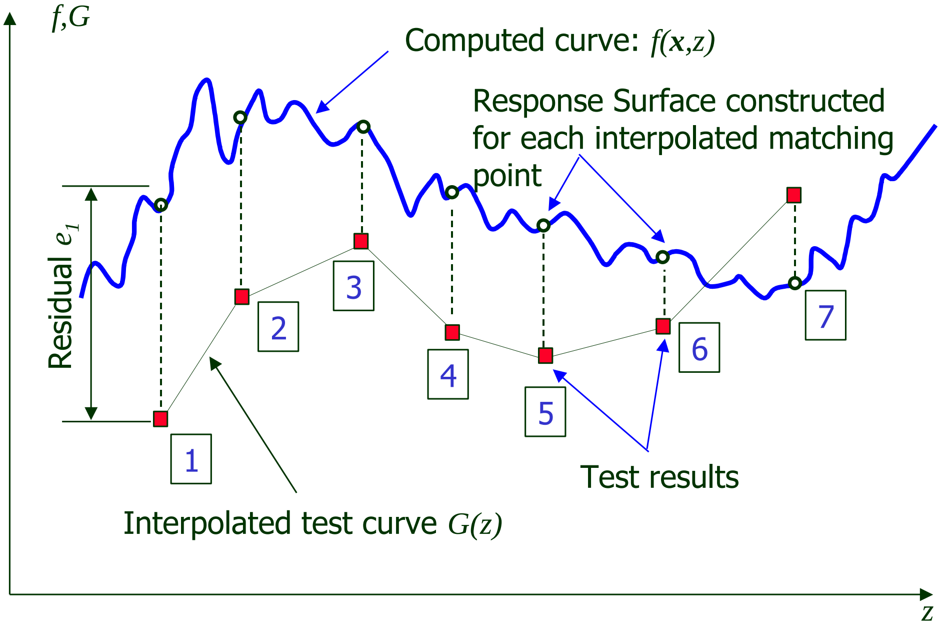

f,G
Computed curve: f(x,z)
Response Surface constructed for each interpolated matching point
Residual e1
1
2
3
4
5
6
7
Test results
Interpolated test curve G(z)
z

Figure 28-1: Entities in Mean Squared Error formulation

The MSE norm is based on a series of P regression points beginning at the start point z_1 and terminating at the end point z_P (see Figure 28-1). The s_p, $p=1,\ldots,P$ are residual scale factors and the W_p, $p=1,\ldots,P$ are weights applied to the square of the scaled residual $(f_p - G_p) / s_p$ at point p.

A major difficulty with ordinate-based curve matching is that steep parts of the curve are difficult to incorporate in the matching. Failure material models typically have the characteristic of a steep decline of the stress-strain curve towards the end of the curve while steep curves also feature in models in which part of the behavior (typically the leading part of the curve) is linear. These kinds of problems present a strong case for incorporation of the abscissa into the curve-matching metric.

A related problem with ordinate-based matching is that the ranges of the computed and target curves often do not coincide horizontally so that some of the points are ignored. It may even happen that at an interim stage of the optimization the two curves do not share any vertical range overlap (there is not a single vertical line which will cross both the computed and the target curves). This type of problem may cause instability of the computation because it becomes impossible to quantify the error.

A third problem is that hysteretic curves (curves with more than one possible *y*-value for some of the *x*-values) cannot be quantified because of the non-uniqueness of the ordinate values of the computed curve with respect to the target curve. I.e. a vertical line may cross the same curve more than once.

28.2.2. Discrete Fréchet Distance

The mathematical literature provides some ideas on curve matching approaches. Two commonly used metrics for curve matching are the *Hausdorff* [1] and *Fréchet* [2][3] distances. The Hausdorff distance measures the mismatch between two point sets so is therefore not suitably general for curve matching as there is no continuous point order. For instance it would not be able to handle a hysteretic curve match. The Fréchet distance is better suited for curve matching because it takes the continuity of the curves into account. The Fréchet distance is formally defined as:

$$Fr(P,Q) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \left\| P(\alpha(t)) - Q(\beta(t)) \right\|$$

where P and Q are polygonal curves, $t \in [0,1]$ represents a position on each curve. The parameters α and β are used to parameterize the distance whereas we can think of t as "time". The analogy is that of a dog walking along the one curve and the dog's owner walking along the other connected by a leash. Both walk continuously and monotonically along the curve from the start point to the end point and can vary their velocities according to α and β. The Fréchet distance is the length of the shortest leash that is sufficient for traversing both curves in this manner.

In 1994 Eiter and Mannila [4] presented a discrete implementation of the continuous Fréchet distance in order to measure the similarity between two polygonal chains P, Q using only the points along the chains [4]. The discrete Fréchet distance (DFD) measures the similarity between curves by taking the minimum of the maximum of all possible edge lengths along a path, which connects all given data points. Mathematically, this can be expressed as

$$DFD(P,Q) = min\left\{ \underset{i=1,\ldots,l}{max} \{ d(p_{a_i}, q_{b_i}) \} \right\},$$

where $d(\cdot\,,\cdot)$ denotes the Euclidean distance. The measure is calculated via the following algorithm [4]:

Input: Two polygonal curves $P = (p_1, \ldots, p_n), Q = (q_1, \ldots, q_m)$

Output: $DFD(P,Q)$

1. Initialize matrix M of size $n \times m$ and set values to -1
2. Calculate the discrete Fréchet distance by calling the function $df(M,n,m)$, where the function df is defined by:

```
function $df(M,i,j)$
    if $M(i,j) > -1$:              return $M(i,j)$
    else if $i = 1$ and $j = 1$:   $M(i,j) := d(p_1, q_1)$
    else if $i > 1$ and $j = 1$:   $M(i,j) := \max\{ df(i-1,1), d(p_i, q_1)\}$
    else if $i = 1$ and $j > 1$:   $M(i,j) := \max\{ df(1,j-1), d(p_1, q_j)\}$
    else if $i > 1$ and $j > 1$:   $M(i,j) := \max\left\{ \min\left\{ \begin{matrix} df(i-1,j) \\ df(i-1,j-1) \\ df(i,j-1) \end{matrix} \right\}, d(p_i, q_j) \right\}$
    else                           $M(i,j) = \infty$
return $M(i,j)$
```

The method takes into account the location and ordering of the points, which makes it applicable in cases where the Euclidean method would fail. However, when dealing with curves of different length, the result may be inaccurate since it considers all points in both curves. Furthermore when points are not uniformly distributed, particularly in case of a big gap between two points on a chain, DFD will focus on the gap. This non-robust behavior can easily distort the distance value. An example for this case can be seen in Figure 28-2.

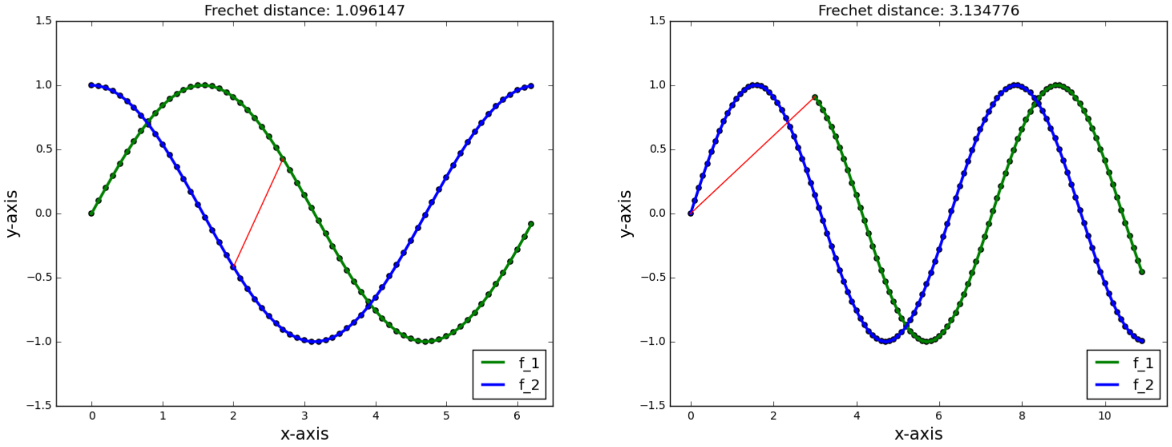


Figure 28-2: The discrete Fréchet distance is calculated between the blue and the green curve. The edge, whose length equals the discrete Fréchet distance is highlighted in red. The plot on the right shows the non-robust behavior of the discrete Fréchet method.

28.2.3. Normalized Dynamic Time Warping

Originating in speech recognition in the 1960s another similarity measure, Dynamic Time Warping (DTW), has been a research topic for scientists of different fields, resulting in many variations of the method [9]. Variations such as FastDTW [10] and the Sakoe-Chiba Band [11] focus on reducing runtime while having an impact on accuracy. In LS-OPT, the normalized DTW approach is implemented where the warping window size remains unchanged and the DTW value is normalized by the length of the warping path.

DTW calculates the distance between two data sets, which may vary in time, via its corresponding warping path $W = (w_1, \dots, w_l)$. This path is the result of the minimum accumulated distance which is necessary to traverse all points in the curves. The normalized DTW distance value can be formulated as

$$DTW(P, Q) = \frac{1}{l} \min_{W} \left\{ \sum_{i=1}^{l} \delta(w_i) \right\},$$

where $\delta(w_i) = d(p_h, q_k)$ if $w_i = (h, k)$, $h \in 1, \dots, n$ and $k \in 1, \dots, m$. Algorithmically, the method calculates a matrix of accumulated distances as follows:

Input: Two polygonal curves $P = (p_1, \dots, p_n)$, $Q = (q_1, \dots, q_m)$

Output: $DTW(P, Q)$

1. Initialize matrix M of size $n \times m$
2. Set the values of M to the Euclidean distances between the points in P and Q
3. Replace the values of M in the following way in order to obtain the minimal accumulated distance:

```
for $i = 1, \dots, n$
```

```
  for $j = 1, \dots, m$
    if $i > 1$ and $j = 1$             $M(i,j) := M(i-1,j) + M(i,j)$
    else if $i = 1$ and $j > 1$        $M(i,j) := M(i,j-1) + M(i,j)$
    else if $i > 1$ and $j > 1$        $M(i,j) := \min\begin{cases} (M(i-1,j) + M(i,j) \\ M(i,j-1) + M(i,j) \\ M(i-1,j-1) + M(i,j) \end{cases}$
return $M(n,m)$
```

The example in Figure 28-3 shows that in contrast to the Mean Squared Error formulation, DTW not only allows one-to-one, but also one-to-many or many-to one mappings.

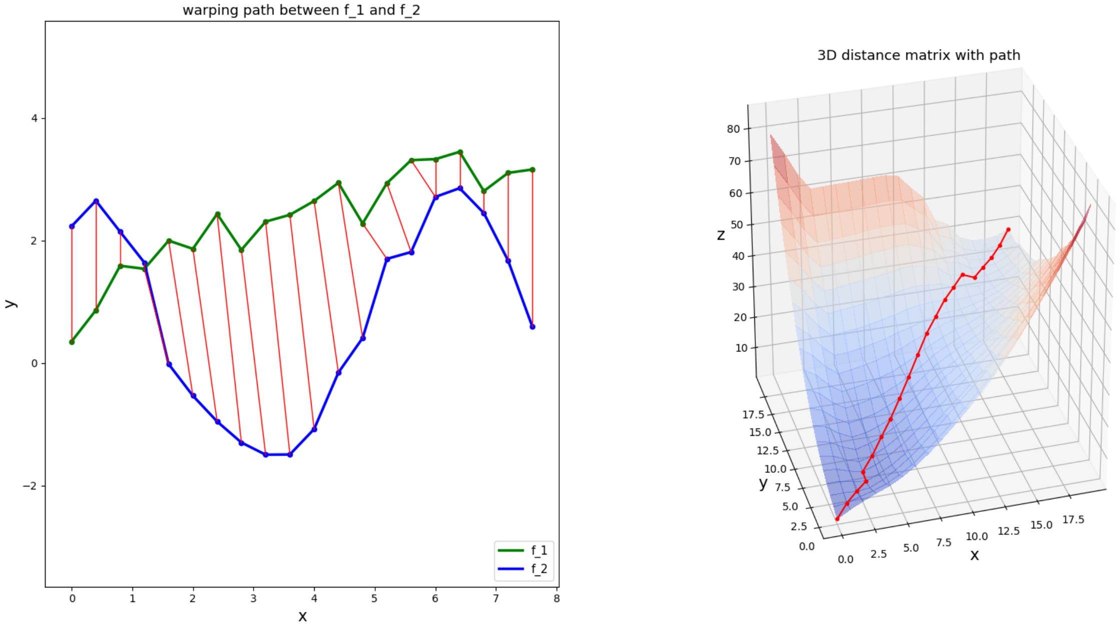


Figure 28-3: The red lines correspond to the warping path created by the DTW method. On the right-hand side the 3D plot illustrates the corresponding matrix of accumulated distances. As described, the method chooses the path of minimum accumulated length.

In general, due to normalization by the length of the path, the normalized DTW distance is unaffected by the density of points and yields substantial results. Nevertheless we strongly suggest using a comparable density of uniformly distributed points on both curves by interpolating missing points where necessary (with the built-in method). This approach avoids insufficient optimization runs and yields comparable results.

In addition, since DTW is required to match all points on both curves, heavy noise can lead to a poor solution in an optimization run as the optimization algorithm may concentrate on the noisy part of the curve if the majority of the distance value is related to it. This effect can be seen when comparing Figure 28-4 and Figure 28-5, where the noisy end of the blue curve increases the distance value by 35%. For such cases Partial Dynamic Time Warping is recommended.

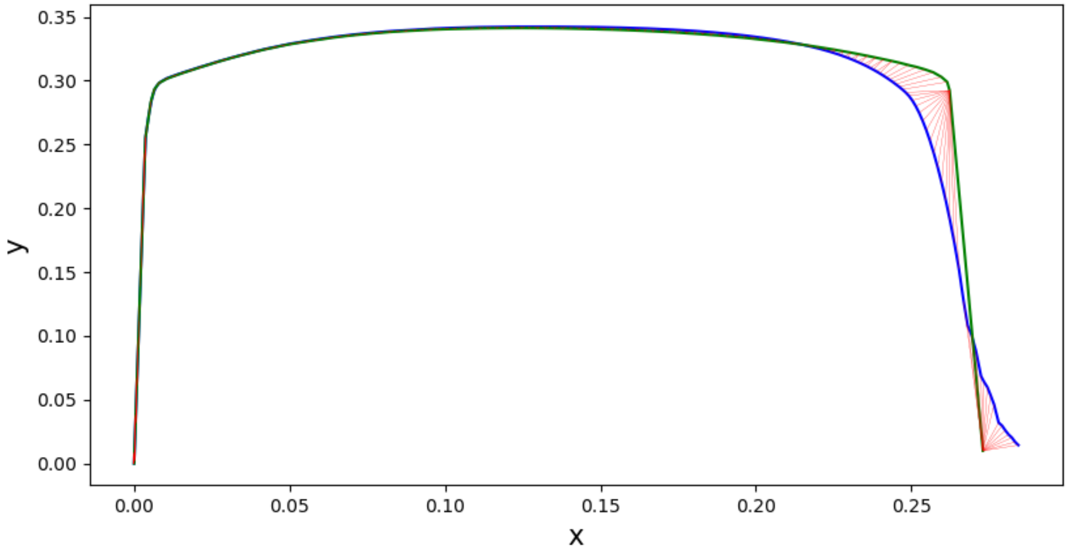


Figure 28-4: Distance value 0.0097849. No noise on the simulated blue curve at the end.

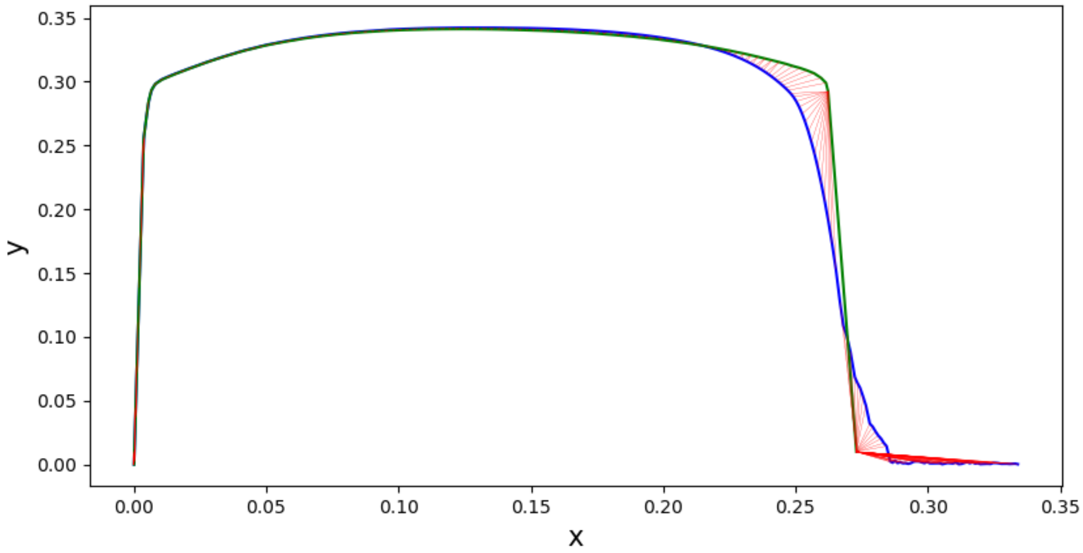


Figure 28-5: Distance value 0.0132290. Exactly the same curves as in figure 3, but the noisy part of the simulated blue curve is not removed.

Partial Dynamic Time Warping

Partial Dynamic Time Warping [22] is a method which can be used to truncate the curves if e.g. the simulation curves have leading or trailing ends that are not desired to be considered for comparison. Such curves are labeled as *incompatible*. As indicated by the name, it uses the Dynamic Time Warping method within a loop which iterates until both curves have the same number of vertices.

If one curve is much longer than the other with respect to the range of ordinate and abscissa, there are multiple end connectors connected to at least one end. An end connector emanates from the end point of the reference curve. Some end connectors may be redundant. These are connectors emanating from the end points of the reference curve, except the innermost one at each end, which is kept to connect the end points of the reference and computed curves.

Partial Dynamic Time Warping recursively trims curves with redundant end connectors until there remain no redundant end connectors. Then the two curves are defined to be compatible and the distance between them can be computed using the standard DTW measure. To make the algorithm function smoothly, the computed curve is *remeshed* in each cycle to have the same length in terms of the number of nodes as the reference curve. The computed partial DTW distance is then the final DTW distance between the compatible curves.

The algorithm for computing the Partial DTW distance follows in Figure 28-6 to Figure 28-10:

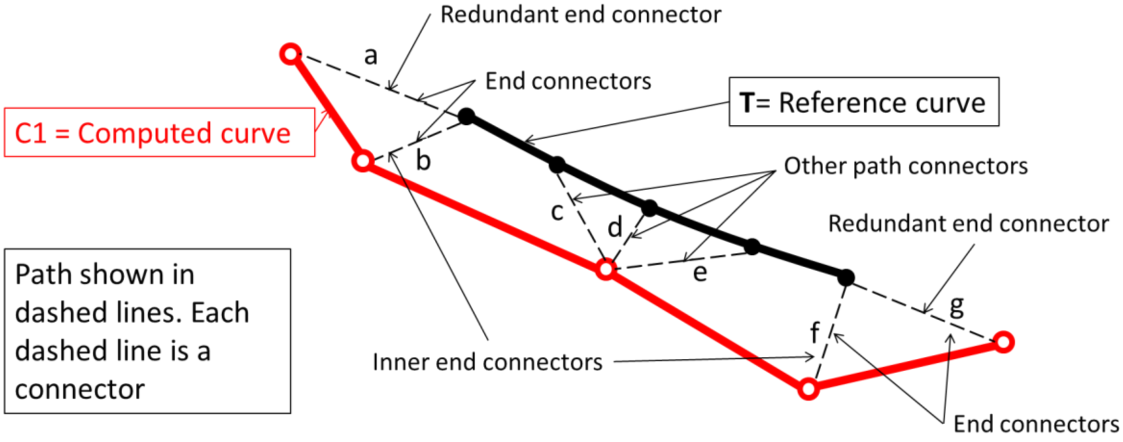


Figure 28-6: DTW path between target curve T and remeshed computed curve C1. The distance between T and C1 is the sum of the path connector lengths (a+b+c+d+e+f+g)/n, where n is the number of connectors.

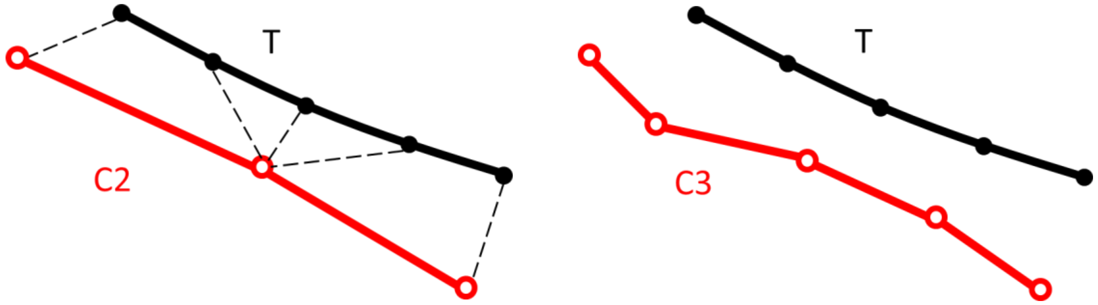


Figure 28-7: Removing redundant end connectors a and g of C1 leads to C2 (left). Remeshing C2 with the same number of nodes that are used by T gives C3 (right).

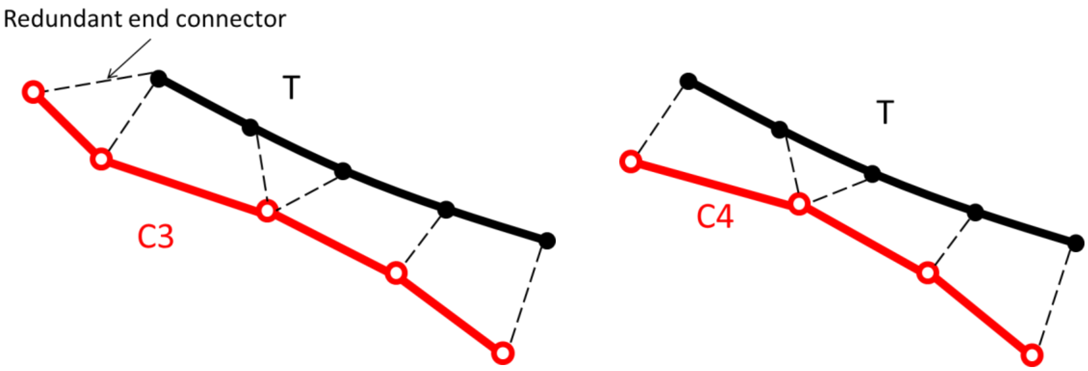


Figure 28-8: The DTW path between T and C3 is calculated. Again there is a redundant end connector remaining (left). C3 is trimmed by removing redundant end connector to get C4 (right).

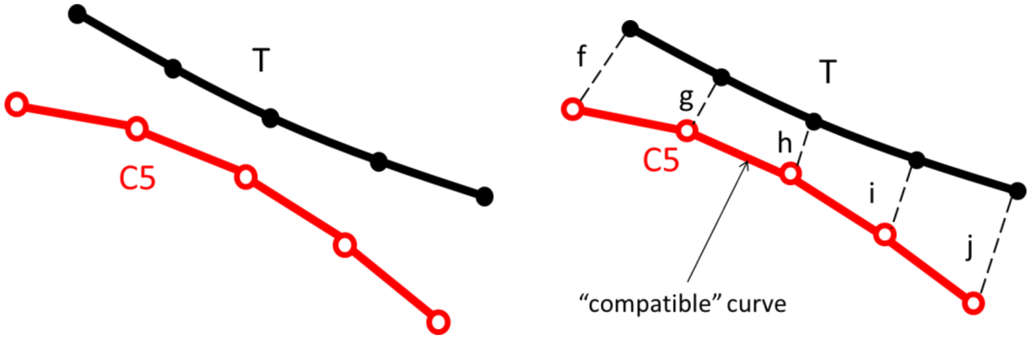


Figure 28-9: C4 is remeshed to get C5, again using the same number of nodes as in T (left). The DTW path between T and C5 is calculated. Since there are no more redundant end connectors, the algorithm is terminated. The curves are now compatible and the final distance between the curves is f+g+h+i+j/m, where m is the number of connectors.

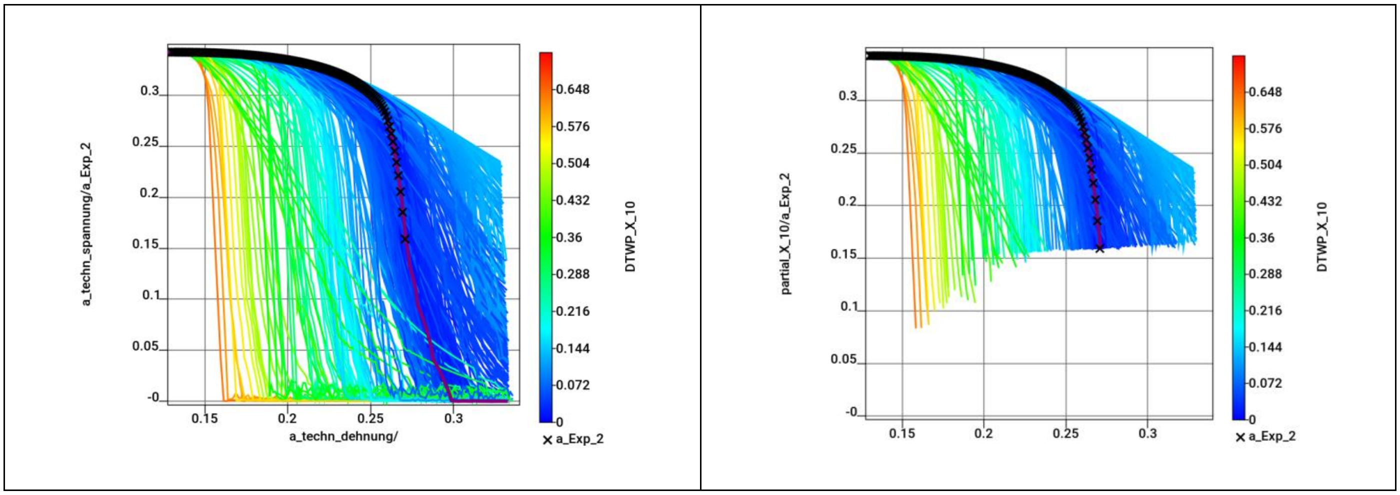


Figure 1-10: Example showing the effect of Partial Dynamic Time Warping applied to a curve set. The heavy magenta line represents the optimum curve closest to the experimental results (black crosses).

28.2.4. Partial Curve Mapping

A logical approach to comparison of the two curves is to map one of the curves onto the other. Two questions which immediately arise are how to scale the curves and how to match two curves of unequal length. Scaling is particularly important since scale changes have an effect on the distances between the two curves. In many cases (e.g. stress vs. strain) there could be several orders of magnitude difference between the values on the abscissa and those of the ordinate.

Curve Mapping maps the points of the one curve onto the second curve and computes the volume (area) between the two curves. When both curves are normalized, this typically yields a mismatch error with value much less than 1 for two reasonably matching curves [12].

A significant problem is that it is not appropriate to map entire curves to one another. A practical reason could be that the test curve, which could be the result of digital output from an experiment, is essentially unedited and therefore contains superfluous points unrelated to the actual behavior of the model. It may also be that the test curve represents only part of the response, perhaps because a full curve could not be obtained from the test. In parameter identification this issue becomes particularly critical as curves are typically computed at widely distributed points throughout the parameter space during the optimization process. This potential disparity of curve length requires partial mapping of the two curves.

The steps for computing the curve mismatch are described in full detail below. The reader should refer to Figure 28-10 which shows a test curve (in thick red) mapped on to a computed curve. The prime symbol (′) is used to denote the curve on which the test curve is being mapped while the double prime symbol (″) is used to denote the finally mapped curve. The test curve is shown inside its smallest bounding box, the boundaries of which are used to normalize the curve. Hence the normalized curve ***a*** is in the [(0,0),(1,1)] range.

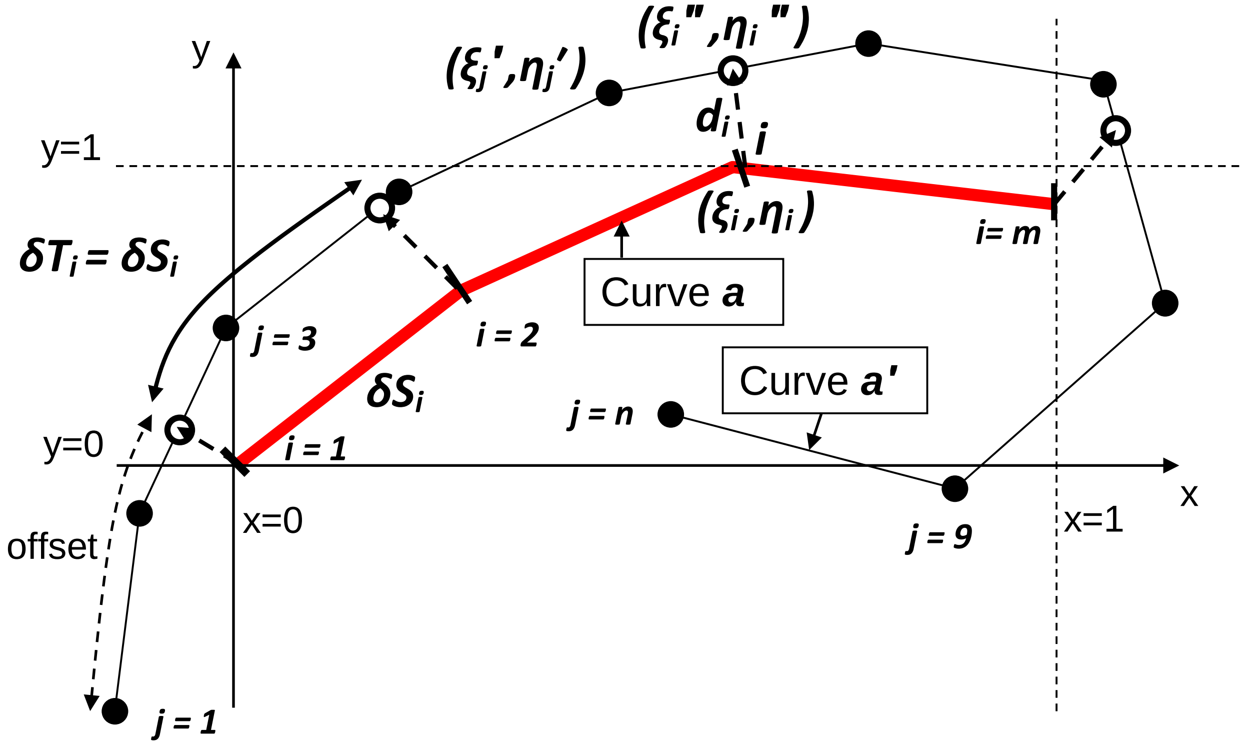


Figure 28-10: Partial curve mapping of Curve a (in red) to Curve a′ with offset. The result is Curve a″. The solid points represent the original vertices of a′ whereas the open circles represent the mapped points representing a″. Curves a and a′ are both normalized to the bounding box of a.

The algorithm for computing the curve mismatch error is as follows:

1. Normalize the *m* point coordinates *i* of the *target* curve ***A*** to its smallest bounding box to create Curve ***a***. See Figure 1.

$$\xi_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \qquad \eta_i = \frac{Y_i - Y_{\min}}{Y_{\max} - Y_{\min}}$$

$$X_{\min} = \min_k X_k \; ; \quad X_{\max} = \max_k X_k \; ; \quad Y_{\min} = \min_k Y_k \; ; \quad Y_{\max} = \max_k Y_k$$

2. Normalize the *n* point coordinates *j* of the *computed* curve ***A′*** to the smallest bounding box of ***A*** to create curve ***a′***. See Figure 1.

$$\xi_j' = \frac{x_j - X_{\min}}{X_{\max} - X_{\min}} \qquad \eta_j' = \frac{y_j - Y_{\min}}{Y_{\max} - Y_{\min}}$$

3. Compute *S*, the total polygon length of ***a***. Also compute the individual segment lengths δS_i:

$$\delta S_i = \sqrt{(\xi_i - \xi_{i-1})^2 + (\eta_i - \eta_{i-1})^2} \; ; \quad i = 2,3,\ldots,m$$

Here a segment is defined as a part of the curve between two consecutive points, connected by a straight line.

4. Scale each segment length to the total curve length *S*:

$$\tilde{s}_i = \delta S_i / S \;; \quad i = 2,3,...,m$$

5. Compute *T*, the total polygon length of ***a'***.
6. If *S* > *T*, rename ***a'*** to ***a*** and ***a*** to ***a'***. Hence ***a*** will always be shorter than ***a'***.
7. Define an *offset* as a starting point of a curve section of total length *S* on curve ***a'***. The offset = λ_p will be varied over *p = 1* to *P* in order to "slide" Curve ***a*** along Curve ***a'***. $\lambda \in [0, T - S]$. Assume *P* increments in this interval so that each increment has size $\Delta\lambda = \dfrac{T-S}{P}$.
8. Set $\lambda_p = \lambda_{p-1} + \Delta\lambda$ to create a new section of the computed curve and create point coordinate pairs by mapping each point of curve ***a*** to curve ***a'***. A typical curve segment *i* on ***a'*** which corresponds to a segment *i* on ***a*** has length $\delta T_i = \delta S_i$ (see Fig. 1). This creates a new set of point pairs ***a"***. The assumption that the length of the mapped section of the long curve is equal to the length of the short curve is critical to the success of the method.
9. Compute the discrepancy (mismatch error) between the two curves ***a*** and ***a"***. This is done by summing the *volumes* v_i representing the individual segment errors. First compute the distances between the point pairs:

$$d_i = \sqrt{(\xi_i" - \xi_i)^2 + (\eta_i" - \eta_i)^2}$$

Then compute the volume component of each segment. (Note for *m* points, there are *m*-1 segments.)

$$v_i = \frac{d_i + d_{i-1}}{2} \times \tilde{s}_i; \quad v_1 = 0; \quad i = 2,3,...,m \;;$$

Then sum the volumes to get the final discrepancy:

$$\varepsilon_p = \sum_{i=1}^{m} v_i$$

10. Set *p* = *p*+1 and repeat from point 8.
11. Find the distance $\varepsilon = \min_p \varepsilon_p$. This is the best match between the curves ***a*** and ***a'***.

28.3. A comparison of similarity measures for curve comparison

The following table summarizes the differences between the methods:

Distance Metric	**Principle**	**Advantage**	**Disadvantage**
Euclidean Distance (MSE)	Mean Squared Error of the difference in curve ordinates	Handles noise, partial curves	Cannot handle hysteresis: the curves must be functions such as time histories.
Partial Curve Mapping (PCM)	Area between curves. Preserves the arc length.	Handles curves of unequal geometric length and hysteresis	Cannot handle noise. Due to arc length preservation.
Discrete Fréchet (DF)	Intuitive definition of a man and dog traversing the two curves. The distance is the length of the shortest leash. Speed can vary, but neither can move backwards.	Handles noise and hysteresis	Cannot handle curves of unequal geometric length (incompatible).
Dynamic Time Warping (DTW)	Minimizes the sum of path connectors between the curves in a one-to-many or many-to-one mapping end to end.	Handles noise and hysteresis	Does not handle incompatible curves efficiently. More complicated to code. More expensive to compute.
Partial DTW (DTW-p)	Uses DTW to recursively reduce the length discrepancy.	Handles noise, hysteresis and incompatible curves (curves with significant length differences)	As the method was conceived to counter the deficiencies of the other methods, there are (apart from the minor disadvantages to DTW) no known prominent disadvantages.

28.4. Full-field material calibration

While simple material models can typically be uniquely identified using a tensile and/or shear test, a major difficulty in parameter identification is ill-posedness of the observation equation[16]:

$$A(x) = y\,;\, x \in \mathrm{X} \tag{28-2}$$

in which the mapping A represents the modeling (e.g. a FE model), $x \in \mathrm{X}$ represents the model parameters (the material constants) bounded by the parameter space X and y represents the data (experimental data) in the observable data space Y. This problem is typically solved as a minimization problem[16]:

$$y = \arg\{\underset{y'}{\mathrm{Min}}\, J(y'; x)\} \equiv A(x) \qquad (28\text{-}3)$$

In Reference [16], Bui lists important issues to check when solving the inverse problem, namely stability of the solution with respect to variations in the data y as well as with respect to modeling errors and small changes in the model space X (robustness). It should be mentioned that *the inverse problem to determine the parameters* x*, for given experimental data* y*, is generally ill-posed*. In material parameter identification, various conditions can cause ill-posedness, for example [16]:

- False experimental data (measurement errors or erroneous data files)
- Incompatible or insufficient data, i.e. there is no set of material parameters that will cause the model to produce results in data space D.
- Noise in the experimental data
- Modeling errors e.g. in the constitutive model or coarse approximation of the model A

These problems are manifested because of the lack of suitable information provided by the chosen test. E.g. providing global information such as a force-displacement curve representing a chosen point or cross-section of a coupon may not be sufficient to characterize a nonlinear material model involving material flow, failure and/or damage. It is also typical of failure behavior to be localized, which eludes capture by global force-displacement data [14]. It is probably safe to assume that the more sophisticated the material model and the phenomena it is intended to model, the more parameters it possesses and the greater the need exists for full-field calibration to capture localized behavior.

While optimization always yields a solution to the unconstrained, but simply bounded, regression problem, the deficiency in the distribution of the input data may result in instability of that solution. It is with this problem, as well as accuracy, in mind that experimental mechanicians, over the last three decades, have introduced full field optical measurement techniques. These allow more comprehensive sampling in order to capture phenomena to which the material parameters have non-zero sensitivity, thereby improving stability of the calibration.

28.4.1. Digital Image Correlation

The purpose of these non-contact methods is to measure the spatial distribution of physical quantities such as displacement or strain. Much of the development in this area arose from the improvements made to camera and computer technology as well as analysis techniques such as the Finite Element Method. Multiple optical measurement methods have been forthcoming such as digital holography and speckle interferometry. Magnetic Resonance Imaging (MRI) has been used to characterize myocardial material. The focus of the current section is an optical measurement method referred to as Digital Image Correlation [21].

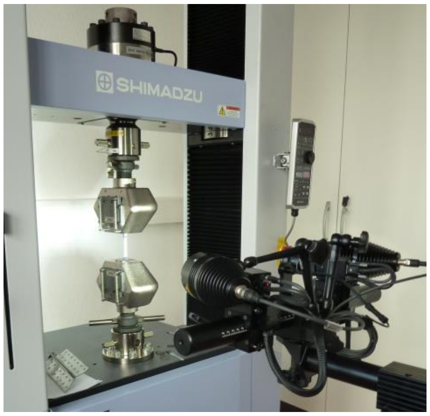


Figure 28-11: Tensile testing equipment and optical measurement system by GOM ARAMIS (courtesy Dynamore GmbH, Stuttgart).

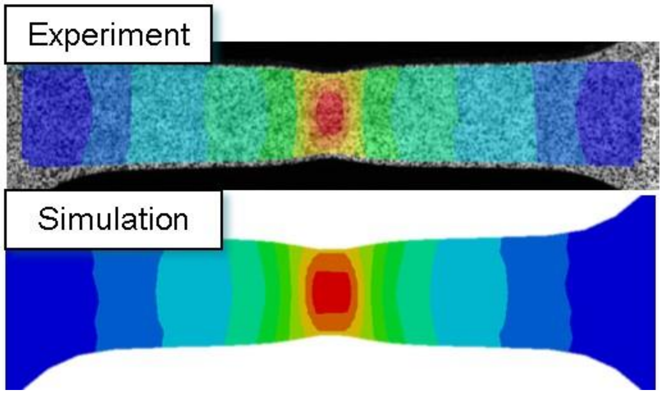


Figure 28-12: Full field measurement of a tensile test using Digital Image Correlation. Above is the DIC image (GOM ARAMIS) while a strain contour plot is featured below (LS-PrePost®).

Digital Image Correlation (DIC) started to appear in the early 1980s and is an optical measurement method (see Figure 28-11 for test setup) which provides full-field displacement measurements for mechanical tests of materials and structures. A specific advantage of this tool is that it exploits numerical images acquired at different stages of loading. As such, it can be used to obtain temporal displacement, deformation or strain fields from an experimental coupon and can be combined with Finite Element Analysis to identify the constitutive properties of a material (see Figure 28-12).

Using DIC results, optimization is used to obtain the parameters which will minimize the distance functional involving the measured field and the computed field as components. As expressed for instance in Mahnken et al [14] the functional used is:

$$f(\boldsymbol{x}) = \sum_{j=1}^{n} \left\| \boldsymbol{\varphi}_j(\boldsymbol{x}) - \widetilde{\boldsymbol{\varphi}}_j \right\|^2$$

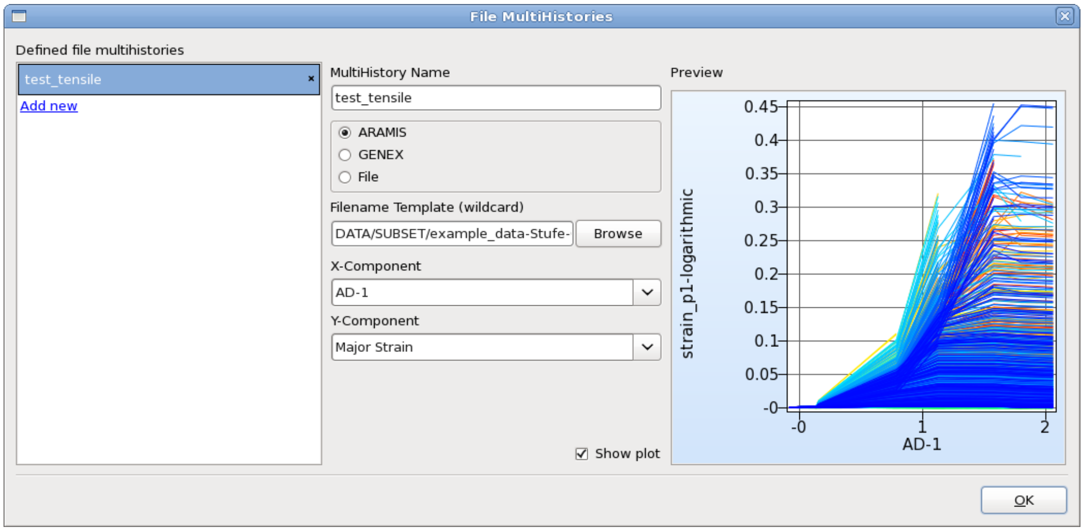
where $\boldsymbol{\varphi_j}(\boldsymbol{x})$ is a vector of nodal displacements or strains with dimension equal to the number of spatially distributed observation points and n is the number of observation states. The functional can also be augmented to incorporate global force-displacement measurements or any other functional resulting in parameter identification based on a multi-scale DIC (see e.g. [19]). While LS-OPT® has included parameter identification features since its first commercial release, the current development goal is to significantly enhance these capabilities to enable LS-OPT to accommodate modern testing techniques such as DIC. While full field calibration has been conducted using LS-OPT in the past, the current implementation generalizes the feature to enable it to address a greater diversity of problems.

A summary of the essential features follows:

1. A core feature of the methodology is the *multi-point history*, a basic mathematical entity representing both spatial and temporal dimensions. Multi-point histories are hyper-curves which are evaluated at multiple locations (possibly hundreds of thousands) and extracted from simulations and experimental data.
2. A binary tree-based exact nearest neighbor algorithm is used to map a test point cloud to an FE model.
3. Both shell and solid elements are supported. The ability to interpolate fields within an individual finite shell element is incorporated into the mapping approach. Solid element quantities are represented by the values at the element center. Depending on the element size, an error will occur as the optical measuring systems only measure at the surface.
4. Any of the available similarity measures (See Section 28.2) can also be applied to multi-point histories.
5. An error analysis enables the user to assess the size of the mismatch as well as the resulting uncertainty of the parameters.
6. Graphical features are available for previewing test results.
7. Distance contour maps are provided for post-processing the discrepancy between computed values and experimental quantities.

Since time is not relevant to the calibration[21], crossplots between kinematic and static quantities are typically used. To enable the creation of cross-plots from the GOM data, both the X- and Y-components shown in Figure 28-13 can be histories or multi-point histories. E.g. a multi-point xx-strain can be crossed with a multi-point yy-strain to create multi-point stress-strain cross-plots or the full-field xx-strain can be crossed with the global force history to create multi-point strain-force cross-plots. The same applies to cross-plots of computational fields

[21] Simulations are often done as quasi-static, using an implicit method to achieve equilibrium at each state of deformation, a notable exception being rate-dependent materials such as those exhibiting visco-elasticity or -plasticity.



Defined file multihistories

test_tensile

Add new

MultiHistory Name: test_tensile

● ARAMIS
○ GENEX
○ File

Filename Template (wildcard): DATA/SUBSET/example_data-Stufe-

X-Component: AD-1

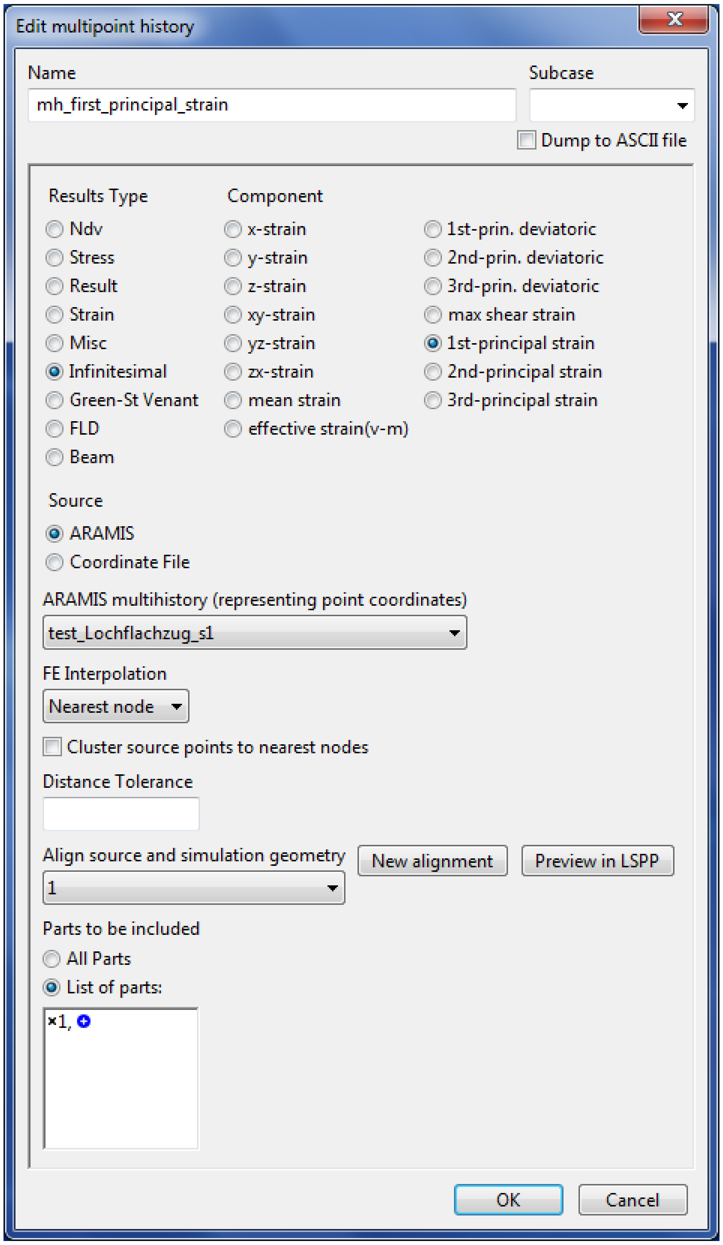
Y-Component: Major Strain

☑ Show plot

Figure 28-13: GOM ARAMIS interface in LS-OPT showing verification plot of selected test value X- and Y-components.

28.4.2. Multi-point histories

In the multi-point histories interface, each history is computed at a location selected from the test point set in the original, undeformed state. Features are provided to preview the test point alignment (see Figure 28-13). E.g. Figure 28-15 shows three representative deformation states representing 4557 points extracted from the DIC database for a plate with a hole while Figure 28-16 shows the FE mesh with corresponding test points. Figure 28-17 shows the outcome of the least squares point alignment (Section 28.4.2).



Edit multipoint history

Name: mh_first_principal_strain

Subcase:

□ Dump to ASCII file

Results Type
○ Ndv
○ Stress
○ Result
○ Strain
○ Misc
● Infinitesimal
○ Green-St Venant
○ FLD
○ Beam

Component
○ x-strain
○ y-strain
○ z-strain
○ xy-strain
○ yz-strain
○ zx-strain
○ mean strain
○ effective strain(v-m)
○ 1st-prin. deviatoric
○ 2nd-prin. deviatoric
○ 3rd-prin. deviatoric
○ max shear strain
● 1st-principal strain
○ 2nd-principal strain
○ 3rd-principal strain

Source
● ARAMIS
○ Coordinate File

ARAMIS multihistory (representing point coordinates): test_Lochflachzug_s1

FE Interpolation: Nearest node

□ Cluster source points to nearest nodes

Distance Tolerance:

Align source and simulation geometry: 1

Parts to be included
○ All Parts
● List of parts:

×1,

Figure 28-14: Multi-point history interface enabling the definition of the response type and source data. FE interpolation can be defined as 'Nearest node' or 'Element'. In the latter option, the result is interpolated at the precise location within the nearest element.

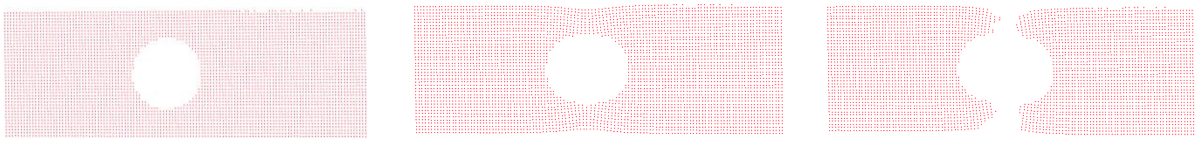


Figure 28-15: Selected deformation states from GOM/ARAMIS output.

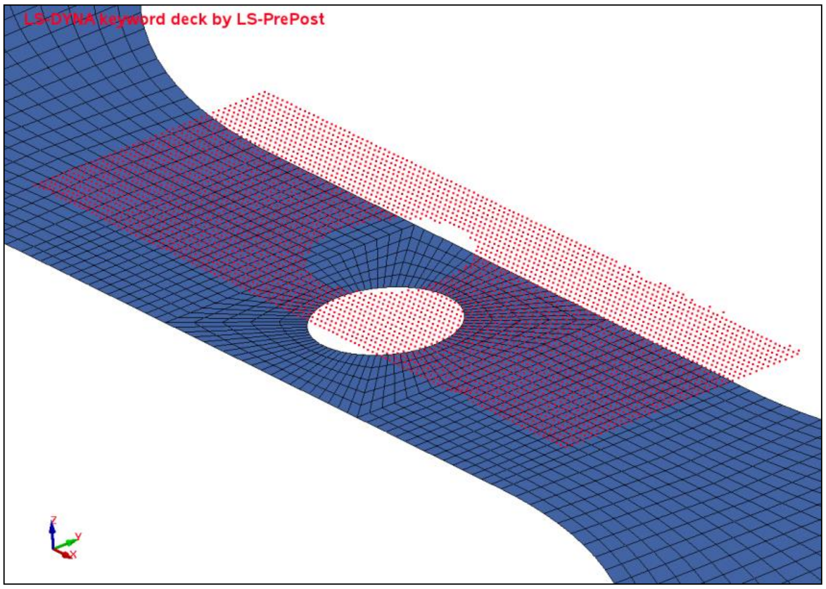


Figure 28-16: Example of test point mapping showing test point set (red) of 4557 points in relation to the Finite Element mesh.

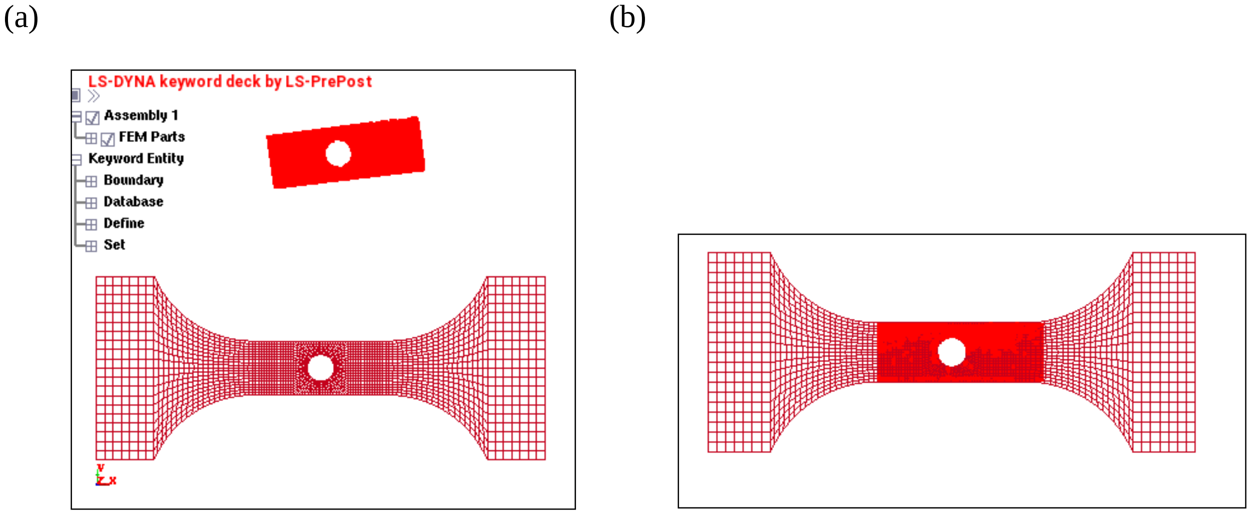


Figure 28-17: Interactive feature for verification of least squares test point alignment. The starting test point location (in bright red) is shown on the left.

28.4.3. Point alignment

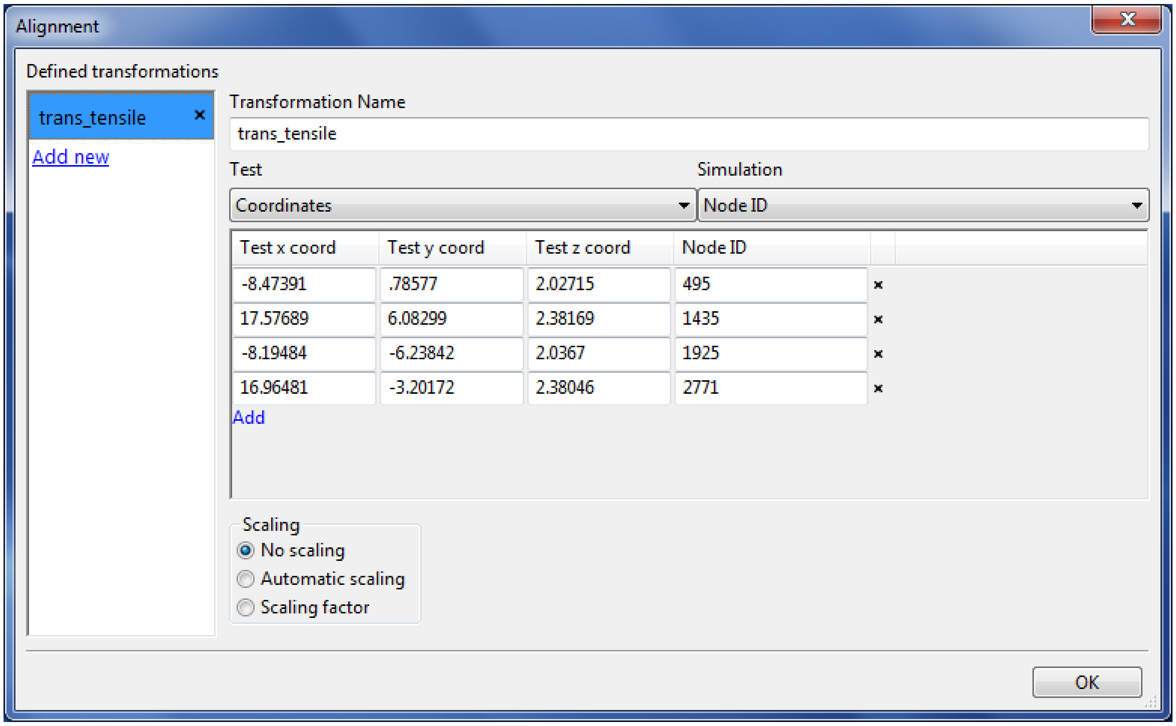
Since the point coordinates obtained along with the experimental results and the coordinate system of the model are typically different, an alignment has to be carried out as a pre-processing step. The alignment is computed using the least squares formulation shown in the equation:

$$\min_{\boldsymbol{T}} \|\boldsymbol{X}_{\text{Test}}\boldsymbol{T} - \boldsymbol{X}_{\text{FE}}\|$$

to match any number of test points $\boldsymbol{X}_{\mathrm{Test}}$ to corresponding coordinates on the FE mesh $\boldsymbol{X}_{\mathrm{FE}}$. The points in $\boldsymbol{X}_{\mathrm{Test}}$ and $\boldsymbol{X}_{\mathrm{FE}}$ respectively do not have to correspond exactly. The solution of the above equation, using Singular Value Decomposition, yields the transformation matrix $\boldsymbol{T}$ which is then used to transform the entire test point cloud (see e.g. Figure 28-17). Three well-distributed points suffice, but any number can be specified.

While the default point alignment algorithm rotates and translates the points, automatic scaling can be selected additionally. In this case T also contains scaling factors calculated from the Singular Values. An option to specify a constant scaling factor is available as well.

Alternatively, the alignment step can be omitted, since LS-PrePost can also be used to pre-align the point set.



Transformation Name: trans_tensile

Test: Coordinates

Simulation: Node ID

Test x coord	Test y coord	Test z coord	Node ID
-8.47391	.78577	2.02715	495
17.57689	6.08299	2.38169	1435
-8.19484	-6.23842	2.0367	1925
16.96481	-3.20172	2.38046	2771

Scaling
● No scaling
○ Automatic scaling
○ Scaling factor

Figure 28-18: Alignment input

28.4.4. Mapping test points to a FE mesh

Mapping: A given aspect of full-field calibration is that, in general, the experimental points and FE nodes do not coincide spatially (see e.g. Figure 28-19(a)). Hence, after the point set has been aligned with the FE mesh, they have to be individually mapped to the nodes or elements. The LS-OPT algorithm enables mapping in 3 dimensions using a binary tree. The approach allows for an exact nearest neighbor search with the ability to map 10^7 query points to 10^7 reference points in reasonable time (suitable for interactive use). The mapping is conducted in parallel with respect to each parameter set (i.e. for each run directory).

Point reduction (cluster): Since points which are remote from nodes may cause inaccuracy of the mapping, an algorithm was implemented to use only those points nearest to nodes. The algorithm proceeds in two steps:

1. For each point, find the nearest node using the nearest neighbor algorithm. This creates a cluster of nodes near the point cloud. Some nodes may be duplicated. After this step, the nodes which are remote from the optical measurement area are excluded.

2. Find the nearest point to each node in the cluster. This eliminates any remote points and yields a one-to-one map between a subset of points and a subset of nodes.

The mapping is illustrated in Figure 28-19.

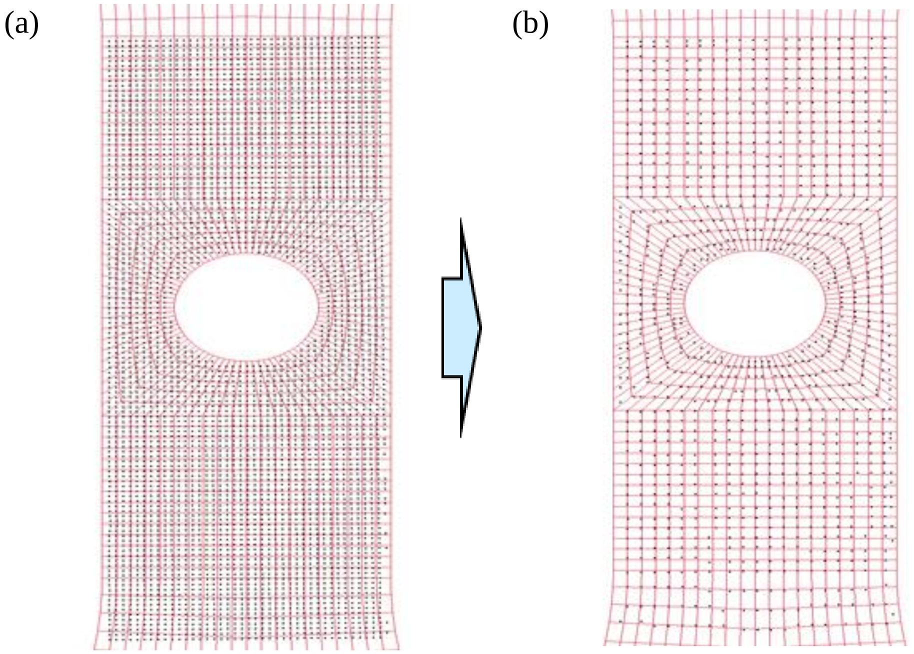


Figure 28-19: Creating (b) a reduced point cluster from (a) the original point set using a nearest nodal neighbor algorithm.

The ideal mesh would be one which exactly coincides with the DIC points.

Tolerance: As a further enhancement to the nearest neighbor nodal mapping, a distance tolerance can be specified so that only data sets associated with nodes close (within tolerance) to experimental points are used to compute the similarity measure.

Element interpolation: An additional measure to circumvent this problem, the ability to interpolate fields *within* each individual finite shell element has also been incorporated into the mapping approach. The scalar result value (e.g. strain) is first averaged at each node. Then the nodal averages of the nearest element are used to interpolate a value at the exact point location within the element.

This is especially important for coarse meshes or high resolution measurement. Obviously, there could still be a discretization error, so the finer the FE mesh, the more accurate the interpolated value.

For solid elements the value is interpolated at the center point of the element.

28.5. Minimizing the maximum residual (Min-Max)

In this formulation, the deviations from the respective target values are incorporated as constraint violations, so that the optimization problem for parameter identification becomes:

$$\text{Minimize} \quad e, \tag{28-4}$$

subject to

$$\left|\frac{f_p(\mathbf{x})-G_p}{s_p}\right| \le e; \quad p = 1,\ldots,P$$

$$e \ge 0.$$

This formulation is automatically activated in LS-OPT when specifying both the lower and upper bounds of f_p / s_p equal to G_p / s_p. There is therefore no need to define an objective function. This is due to the fact that an auxiliary problem is automatically solved internally whenever an infeasible design is found, ignoring the objective function until a feasible design is obtained. When used in parameter identification, the constraint set is in general never completely satisfied due to the typically over-determined systems used.

Since s_p defaults to 1.0, the user is responsible for the normalization in the maximum violation formulation. This can be done by e.g. using the target value to scale the response $f(\boldsymbol{x})$ so that:

$$\left|\frac{f_p(\mathbf{x})}{G_p}-1\right| \le e; \quad p = 1,\ldots,P$$

$$e \ge 0.$$

Omitting the scaling may cause conditioning problems in some cases, especially where constraint values differ by several orders of magnitude. This option has been automated.

28.6. Nonlinear regression: Confidence intervals

Assume the nonlinear regression model:

$G(t) = F(t, \boldsymbol{x}) + \varepsilon,$

where the measured result G is approximated by F and $\boldsymbol{x}$ is a vector of unknown parameters. The nonlinear least squares problem is obtained from the discretization:

$$\min_x \frac{1}{P}\sum_{p=1}^{P}\left(G_p - F_p(\boldsymbol{x})\right)^2$$

is solved to obtain $\boldsymbol{x}^*$. The variance $\sigma^2$is estimated by

$$\sigma^2 = \frac{1}{P-n}\|G - F(\boldsymbol{x}^*)\|^2$$

where $\boldsymbol{F}$ is the P-vector of function values predicted by the model and n is the number of parameters. The $100(1-\alpha)$% confidence interval for each x_i^* is:

$\left(x_i : |x_i^* - x_i| \le \sqrt{\hat{C}_{ii}} t_{P-n}^{\alpha/2}\right),$

where

$$\widehat{\boldsymbol{C}} := \hat{\sigma}^2\{[\nabla\boldsymbol{F}(\boldsymbol{x}^*)]^T[\nabla\boldsymbol{F}(\boldsymbol{x}^*)]\}^{-1}$$

and $t^{\alpha}_{P-n}$is the Student *t*-distribution for α.

$\nabla\boldsymbol{F}$ is the $P \times n$ matrix obtained from the n derivatives of the P response functions representing P points at the optimum $\boldsymbol{x}^*$. The optimal solution is therefore calculated first, followed by the confidence interval.

A critical issue is to ensure that $\nabla\mathbf{F}$ is not based on a gradient obtained from a spurious response surface (e.g. occurring due to noise in the response). Monitoring convergence and selected statistical parameters such as the RMS error and R^2 can help to estimate a converged result. In many cases material identification problems involve smooth functions (e.g. tensile tests) so that spurious gradients would normally not be a problem.

28.7. REFERENCES

[1] Rockafellar, R.T., Wets, R. J-B. Variational Analysis, Springer Verlag, 2005, p. 117

[2] Fréchet, M., Sur quelques points du calcul fonctionnel. Rendiconti del Circolo Mathematico di Palermo, 22:1-74, 1906.

[3] Alt, H. and Godau, M. Computing the Fréchet distance between two polygonal curves, Internat. J. Comput. Geom.. Appl., 5:75-91, 1995.

[4] Eiter, T. and Mannila, H., Computing discrete Fréchet distance. Tech. Report CD-TR 94/64, Information Systems Department, Technical University of Vienna, 1994.

[5] Aach, J. and Church, G., Aligning gene expression time series with time warping algorithms, Bioinformatics. Vol. 17, pp. 495-508, 2001.

[6] Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T. and Simon, I., A new approach to analyzing gene expression time series data, Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology, pp. 39-48, 2002.

[7] Yi, B.K., Jagadish, H. and Faloutsos, C., Efficient retrieval of similar time sequences under time warping, ICDE '98, pp. 23-27, 1998.

[8] Zhang, Z., Huang, K. and Tan, T., Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes, 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, 2006, pp. 1135-1138.

[9] Müller, M., Information Retrieval for Music and Motion. p. 69-84. Springer, 2007.

[10] Salvador, S. and Chan, P., FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. Intell. Data Anal., 11(5):561-580, Oct 2007.

[11] Sakoe, H. and Chiba, S., Dynamic programming algorithm optimization for spoken word recognition, in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, no. 1, pp. 43-49, Feb 1978.

[12] Witowski, K. and Stander, N. Parameter Identification of Hysteretic Models using Partial Curve Mapping. Proceedings of the 12th AIAA Aviation Technology, Integration and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 17-19 September 2012, Indianapolis, Indiana, US

[13] Stander, N., Witowski, K., Ilg, C., Haufe, A., Helbig, M., Koch, D. Application of Digital Image Correlation To Material Parameter Identification, Proceedings of the 12th World Congress on Structural and Multidisciplinary Optimization, Braunschweig, Germany, June 5-9, 2017.

[14] Mahnken, R., Stein, E. Parameter Identification for Finite Deformation Elasto-Plasticity in Principal Directions, Comput. Methods Appl. Mech. Engrg, 147, 17-39, 1997.

[15] Desrues, J., Réthoré, J. Full Field Measurements and Identification in Solid Mechanics, Chapter 14: Characterization of Localized Phenomena, 379-409, Eds. Grédiac, M., Hild, F., Wiley, 2013.

[16] Bui, H.D. Inverse Problems in the Mechanics of Materials, An Introduction, CRC Press, Boca Raton, London, 1994).

[17] Pagnacco, E., Moreau, A., Lemosse, D. Inverse Strategies for the identification of elastic and viscoelastic material properties using full-field measurements, Materials Science and Engineering, Vol. 452-453, 737-745, 2007

[18] Leclerc, H., Périé, J.-N., Roux, S., Hild, F. Integrated Digital Image Correlation for the Identification of Mechanical Properties, MIRAGE 2009, LNCS 5496, 161-171, 2009, A Gagalowicz and W. Philips (Eds.), Springer-Verlag, 2009

[19] Passieux, J.C., Bugarin, F., David, C., Périé, J.-N., Robert, L. Multiscale Displacement Field Measurement Using Digital Image Correlation: Application to the Identification of Elastic properties, https://hal.archives-ouvertes.fr/hal-00949038, 2014.

[20] Gu, J., Cooreman, S., Smits, A., Bossuyt, S., Sol, H., Lecompte, D. and Vantomme, J. Full-field optical measurement for material parameter identification with inverse methods, High Performance Structures and Materials III, WIT Transactions on the Built Environment, 85, 2006. doi10.2495/HPSM06024.

[21] Bornert, M., Hild, F., Orteu, J.-J., Roux, S. Full Field Measurements and Identification in Solid Mechanics, Chapter 6: Digital Image Correlation, 157-190, Eds. Grédiac, M., Hild, F., Wiley, 2013.

[22] Witowski, K., Stander, N. Modified Dynamic Time Warping for Utilizing Partial Curve Data to Calibrate Material Models using LS-OPT. Proceedings of the 16th International LS-DYNA Users Conference, June 2020.

29. Probabilistic Fundamentals

29.1. Introduction

No system will be manufactured and operated exactly as designed. Adverse combinations of design and loading variation may lead to undesirable behavior or failure; therefore, if significant variation exists, a probabilistic evaluation may be desirable.

Sources of variation are:

1. Variation in structural properties; for example: variation in yield stress.
2. Variation in the environment; for example: variation in a load.
3. Variation occurring during the problem modeling and analysis; for example: buckling initiation, mesh density, or results output frequency.

From the probabilistic analysis we want to infer:

1. Distribution of the response values.
2. Probability of failure.
3. Properties of the designs associated with failure.
    - Variable screening - identify important noise factors.
    - Dispersion factors - factors whose settings may increase variability of the responses.
4. Efficient redesign strategies.

29.2. Probabilistic variables

The probabilistic component of a parameter is described using a probability distribution; for example, a normal distribution. The parameter will therefore have a mean or nominal value as specified by the distribution, though in actual use the parameter will have a value randomly chosen according to the probability density function of the distribution.

The relationship between the control variables and the variance can be used to adjust the control process variables in order to have an optimum process. The variance of the control and noise variables can be used to predict the variance of the system, which may then be used for redesign. Knowledge of the interaction between the control and noise variables can be valuable; for example, information such as that the dispersion effect of the material variation (a noise variable), may be less at a high process temperature (a control variable) can be used to select control variables for a more robust manufacturing process.

Three probabilistic associations between variables are possible:

1. Their nominal values and distributions are the same.
2. Their nominal values differ but they refer to the same distribution.
3. Their nominal values are the same but their distributions differ.

29.3. Basic computations

29.3.1. Mean, variance, standard deviation, and coefficient of variation

The mean of a set of responses is

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i .$$

The variance is

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2 .$$

The standard deviation is simply the square root of the variance

$$s = \sqrt{s^2} .$$

The coefficient of variation, the standard deviation as a proportion of the mean, is computed as

$$c.o.v. = s/\bar{y} .$$

29.3.2. Correlation of responses

Whether a variation in displacements in one location causes a variation in a response value elsewhere is not always clear.

The covariance of two responses indicates whether a change in the one is associated with a change in the other.

$$Cov(Y_1, Y_2) = E\left[(Y_1 - \mu_1)(Y_2 - \mu_2)\right],$$

$$Cov(Y_1, Y_2) = E\left[Y_1 Y_2\right] - E(Y_1)E(Y_2).$$

The covariance can be difficult to use because it is unscaled. The standard deviation of the responses can be used for scaling. The coefficient of correlation is accordingly

$$\rho = \frac{Cov(Y_1, Y_2)}{\sigma_1 \sigma_2}.$$

The confidence interval on the coefficient of correlation is described in the next section.

29.3.3. Confidence intervals

The confidence interval on the mean assuming a normal distribution and using s^2 as an estimate to the variance is

$$\bar{y} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} < \mu < \bar{y} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}},$$

with μ the mean, $\bar{y}$ the estimate to the mean, and $t_{\alpha/2,n-1}$ the relevant critical value of the *t*-distribution.

The confidence interval on the variance assuming a normal distribution and using s^2 as an estimate to the variance is

$$\frac{(n-1)s^2}{\chi^2_{\alpha/2,n-1}} < \sigma^2 < \frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}},$$

with σ^2 the variance and $\chi^2_{\alpha/2,n-1}$, $\chi^2_{1-\alpha/2,n-1}$ the relevant critical values of the χ^2 distribution.

The confidence interval on the probability of an event is

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p < \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}},$$

with *p* the probability, $\hat{p}$ the estimate to the probability, and $z_{\alpha/2,n-1}$ the relevant critical value of the normal distribution.

The coefficient of correlation has a confidence interval of

$$\tanh\left[\frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right) - \frac{t_{1-\alpha/2,N}}{\sqrt{N-3}}\right] \le \rho \le \tanh\left[\frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right) + \frac{t_{1-\alpha/2,N}}{\sqrt{N-3}}\right].$$

29.4. Probabilistic methods

The reliability – the probability of not exceeding a constraint value – can be computed using probabilistic methods.

The accuracy can be limited by the accuracy of the data used in the computations as well as the accuracy of the simulations. The choice of methods depends on the desired accuracy and intended use of the reliability information.

More details on probabilistic methods can be found in, for example, the recent text by Haldar and Mahadevan [1].

29.4.1. Monte Carlo simulation

A Monte Carlo simulation aims to compute results with the same scatter as what will occur in practice.

Multiple analyses are conducted using values of the input variables selected considering their probability density function. The results from these analyses should have the scatter expected in practice. Under the law of large numbers the output results will eventually converge.

Applications of a Monte Carlo investigation are:

1. Compute the distribution of the responses, in particular the mean and standard deviation.
2. Compute reliability.
3. Investigate design space – search for outliers.

The approximation to the nominal value is:

$$E[f(X)] = \frac{1}{N}\sum f(X_i).$$

If the X_i are independent, the laws of large numbers allow us any degree of accuracy by increasing N. The error of estimating the nominal value is a random variable with standard deviation

$$\sigma_\theta = \frac{\sigma}{\sqrt{N}}.$$

with σ the standard deviation of $f(\mathbf{x})$ and N the number of sampling points. The error is therefore unrelated to the number of design variables.

The error of estimating p, the probability of an event, is a random value with the following variance

$$\sigma_\theta^2 = \frac{p(1-p)}{N},$$

which can be manipulated to provide a minimum sampling. A suggestion for the minimum sampling size provided by Tu and Choi [2] is:

$$N = \frac{10}{P[G(x) \le 0]}.$$

The above indicates that for a 10% estimated probability of failure; about 100 structural evaluations are required with some confidence on the first digit of failure prediction. To verify an event having a 1% probability; about a 1000 structural analyses are required, which usually would be too expensive.

A general procedure of obtaining the minimum number of sampling points for a given accuracy is illustrated using an example at the end of this section. For more information, a statistics text (for example, reference [3]) should be consulted. A collection of statistical tables and formulae such as the CRC reference [4] will also be useful.

The variance of the probability estimation must be taken into consideration when comparing two different designs. The error of estimating the difference of the mean values is a random variable with a variance of

$$\sigma_\theta^2 = \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2},$$

with the subscripts 1 and 2 referring to the different design evaluations. The error of estimating the difference of sample proportions is a random variable with a variance of

$$\sigma_\theta^2 = \frac{p_1(1-p_1)}{N_1} + \frac{p_2(1-p_2)}{N_2}.$$

The Monte Carlo method can therefore become prohibitively expensive for computing events with small probabilities; more so if you need to compare different designs.

The procedure can be sped up using Latin Hypercube sampling, which is available in LS-OPT. These sampling techniques are described elsewhere in the LS-OPT manual. The experimental design will first be computed in a normalized, uniformly distributed design space and then transformed to the distributions specified for the design variables.

Example

The reliability of a structure is being evaluated. The probability of failure is estimated to be 0.1 and must be computed to an accuracy of 0.01 with a 95% confidence. The minimum number of function evaluations must be computed.

For an accuracy of 0.01, we use a confidence interval having a probability of containing the correct value of 0.95. The accuracy of 0.01 is taken as 4.5 standard deviations large using the Tchebysheff's theorem, which gives a standard deviation of 0.0022. The minimum number of sampling points is therefore:

$$N = \frac{pq}{\sigma^2} = \frac{(0.9)(0.1)}{(0.0022)^2} = 18595.$$

Tchebysheff's theorem is quite conservative. If we consider the response to be normally distributed then for an accuracy of 0.01 and a corresponding confidence interval having a probability of containing the correct value of 0.95, a confidence interval 1.96 standard deviations wide is required. The resulting standard deviation is 0.051 and the minimum number of sampling points is accordingly:

$$N = \frac{pq}{\sigma^2} = \frac{(0.9)(0.1)}{(0.051)^2} = 3457.$$

29.4.2. Monte Carlo analysis using metamodels or classifiers

Performing the Monte Carlo analysis using approximations to the functions instead of FE function evaluations allows a significant reduction in the cost of the procedure.

A very large number of function evaluations (millions) are possible considering that function evaluations using the metamodels are very cheap. Accordingly, given an exact approximation to the responses, the exact probability of an event can be computed.

The choice of the point about which the approximation is constructed has an influence on accuracy. Accuracy may suffer if the metamodel is not accurate close to the failure initiation hyperplane, $G(\mathbf{x}) = 0$. A metamodel accurate at the failure initiation hyperplane (more specifically the Most Probable Point of failure) is desirable in the cases of nonlinear responses. The results should however be exact for linear responses or quadratic responses approximated using a quadratic response surface.

Using approximations to search for improved designs can be very cost-efficient. Even in cases where absolute accuracy is not good, the technique can still indicate whether a new design is comparatively better.

The number of FE evaluations required to build the approximations increases linearly with the number of variables *n* for linear approximations (the default being 1.5(*n*+1) points) and quadratically for quadratic approximations (the default being 0.75(*n*+2)(*n*+1) points).

29.4.3. Sequential Monte Carlo analysis using metamodels or classifiers

Approximation of the limit state function $G(\boldsymbol{x}) = 0$ for probabilistic analysis can be done using samples generated in a single iteration or sequentially. Sequential sampling allows iterative refinement of the limit state boundary as well as a convergence study. As more samples are added, accuracy of the metamodel/classifier used to calculate the failure probability is expected to increase until convergence, which can be based on failure probability, approximation accuracy and maximum iterations. A space filling sampling strategy should be used for sequential sampling.

Adaptive sequential sampling can be performed using classifier-based EDSD sampling constraints. Space-filling samples can then be selected within an irregular subregion, such as in the vicinity of the failure boundary estimate ($-\varepsilon \leq s(\boldsymbol{x}) \leq \varepsilon$). The subregion is updated after every iteration as more samples are added. This can be very cost-efficient, as (potentially large) regions that are not expected to improve the accuracy are avoided. The criteria to define the adaptive subregion can be customized; some of the quantities of interest are proximity to the failure boundary [[5]], joint probability density function value to sample around the most probable point (MPP) [[6], [7], [8]], probability of misclassification to sample potential areas with approximation error [9] etc.

29.4.4. Correlated variables

Considering the correlation $Cov(Y_i, Y_j) = E\left[(Y_i - \mu_i)(Y_j - \mu_j)\right] = \Sigma_{ij}$ between variables, we construct the covariance matrix

$$\begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \dots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \dots & \Sigma_{2n} \\ \dots & \dots & \dots & \dots \\ \Sigma_{n1} & \Sigma_{n2} & \dots & \Sigma_{nn} \end{bmatrix},$$

from which we compute the eigenvalues and eigenvectors as $\sum E = \lambda^2 E$ with E and λ^2 respectively the eigenvectors and the eigenvalues of the covariance matrix.

The correlated variables are created by firstly generating independent variables and transforming them back to being correlated variables using the eigenvalues and eigenvectors of the covariance matrix

$\mathbf{X} = \lambda^1 E^1 iid^{\,1} + \ldots + \lambda^n E^n iid^{\,n}$ with ***X*** the correlated variables and ***iid*** the independent variables. This method is only valid for normally distributed variables.

Consider a function of correlated variables $F = \sum_{i=1}^{n} a_i Y_i$; the statistics of this functions are computed as

$$E(F) = \sum_{i=1}^{n} a_i \mu_i ,$$

$$V(F) = \sum_{i=1}^{n} a_i^2 V(Y_i) + 2\sum_{i=1}^{n} \sum_{j=i+1}^{n} a_i a_j COV(Y_i, Y_j).$$

29.4.5. First-Order Second-Moment Method (FOSM)

For these computations we assume a linear expansion of the response. The reliability index of a response $G(X) < 0$ is computed as:

$$\beta = \frac{E[G(X)]}{D[G(X)]},$$

with *E* and *D* the expected value and standard deviation operators respectively. A normally distributed response is assumed for the estimation of the probability of failure giving the probability of failure as:

$$P_f = \Phi(-\beta) \ or \ 1 - \Phi(\beta),$$

with $\Phi(x)$ the cumulative distribution function of the normal distribution.

The method therefore (*i*) computes a safety margin, (*ii*) scale the safety margin using the standard deviations of the response, and (*iii*) then convert the safety margin to a probability of failure by assuming that the response is normally distributed.

The method is completely accurate when the responses are linear functions of normally distributed design variables. Otherwise the underlying assumption is less valid at the tail regions of the response distribution. Caution is advised in the following cases:

1. Nonlinear responses: Say we have a normally distributed stress responses - this implies that fatigue failure is not normally distributed and that computations based on a normal distribution will not be accurate.
2. The variables are not normally distributed; for example, one is uniformly distributed. In which case the following can have an effect:
   - A small number of variables may not sum up to a normally distributed response, even for a linear response.
   - The response may be strongly dependent on the behavior of a single variable. The distribution associated with this variable may then dominate the variation of the response. This is only of concern if the variable is not normally distributed.

Considering the accuracy of the input data, this method can be reasonable. For example, it should be common that the distribution of the input data can only be estimated using a mean and a standard deviation with a 20% error bound, in which case the results should be understood to have at the least a matching certainty. Interpreting the results in terms of a number of standard deviations can be a reasonable engineering approximation under these circumstances.

29.4.6. Design for six-sigma methods

See the section for FOSM keeping in mind that the reliability index β is the number of standard deviations.

29.4.7. The most probable point

Probabilistic methods based on the most probable point of failure focus on finding the design perturbation most likely to cause failure.

To understand how these methods works, consider the limit state function *G(x)* dividing the responses into two design regions: one consisting of acceptable responses and the other of unacceptable responses. The two regions are separated by the hyperplane described by *G(x)=0*.

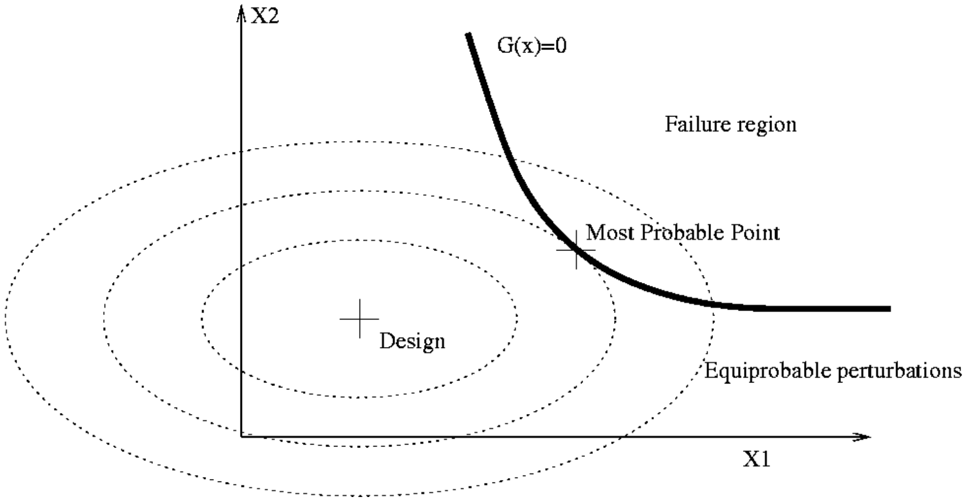


Figure 29-1: Finding the most probable point of failure. The most probable point is the point on the line G(x)=0 closest to the design in the probabilistic sense.

We want to find the design perturbation most likely to cause the design to fail. This is difficult in the problem as shown in Figure 29-1, because all variables will not have an equal influence of the probability of failure due to differences in their distributions. In order to efficiently find this design perturbation, we transform the variables to a space of independent and standardized normal variables, the ***u***-space.

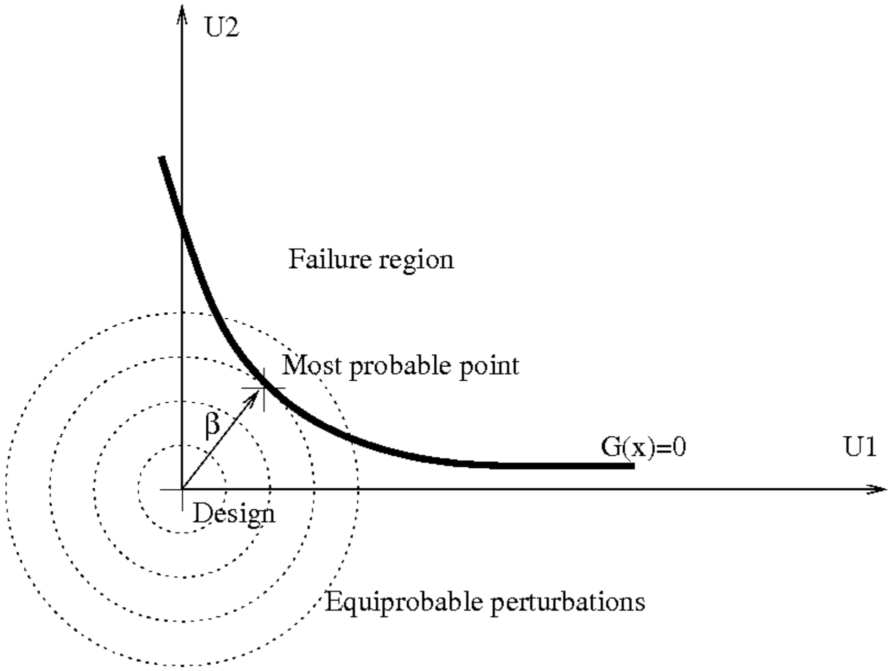


Figure 29-2: Most probable point in the transformed space. In the transformed space the most probable point is the point on the line G(X)=0 the closest to the origin.

The transformed space is shown in Figure 29-2. The point on the limit state function with the maximum joint probability is the point the closest to the origin. It is found by solving the following optimization problem:

$$\text{Minimize: } \sqrt{\sum_{i=1}^{n} u_i^2}$$

$$\text{Subject to: } G(u) = 0.$$

This point is referred to as the *most probable point* (MPP) and the distance from the origin in the ***u***-space is referred to as the first-order probability index β_{FORM}.

The advantages of the most probable point are:

1. The MPP gives an indication of the design most likely to fail.
2. Highly accurate reliability methods utilizing an approximation around the MPP are possible.

29.4.8. FORM (First Order Reliability Method)

The Hasofer-Lind transformation is used to normalize the variables:

$$u_i = \frac{x_i - \mu_i}{\sigma_i}.$$

The minimization problem is accordingly solved in the ***u***-space to find the first-order probability index β_{FORM}. Approximations to the responses are used to solve the optimization problem.

The probability of failure is found assuming a normally distributed response as

$$P_f = \Phi(-\beta_{FORM}),$$

with Φ the cumulative density function of the normal distribution.

The error component of the procedure is due to (*i*) curvature of the constraint, (*ii*) the error component of the approximating function used in the computations, and (*iii*) the assumption of a normal distribution for the computation of failure.

The method is considered conservative considering that disregarding the curvature of the constraint results in an overestimation of the probability of failure.

29.4.9. Design sensitivity of the most probable point

For a probabilistic variable we use the partial derivative as:

$$\frac{\partial P}{\partial x_i} = \frac{\partial P}{\partial \beta}\frac{\partial \beta}{\partial u_i}\frac{\partial u_i}{\partial x_i},$$

with $\partial P / \partial \beta$ the derivative of the CDF function of the normal distribution.

For deterministic variables, which do not have a probabilistic component and therefore no associated ***u*** variables:

$$\frac{\partial P}{\partial x_i} = \frac{\partial P}{\partial \beta}\frac{\partial \beta}{\partial f}\frac{\partial f}{\partial x_i},$$

with $\partial \beta / \partial f$ taken as $\beta / \left(f_{constraint} - f_{nominal} \right)$.

For the pathological case of being at the MPP, the vector associated with β vanishes and we use:

$$\frac{\partial P}{\partial x_i} = 0.4\frac{\partial G}{\partial u_i}\frac{\partial u_i}{\partial x_i},$$

with 0.4 the relevant value derivative of the CDF function of the normal distribution.

29.5. Required number of simulations

29.5.1. Overview

A single analysis of a noisy structural event yields only a single value drawn from an unexplored population. The whole population can be explored and quantified using a probabilistic investigation if the computational cost is reasonable. The cost of this probabilistic analysis is of quite some concern for FEA results and is therefore expounded in the following subsections.

Rough rules of thumb:

- 20 FE evaluation, a maximum of 10 design variables, and a metamodel-based probabilistic analysis for design purposes
- 50 FE evaluations, about 5 design variables, and a metamodel-based probabilistic analysis for a detailed analysis of the scatter in the results and the role of the design variables
- 100 FE evaluations and a Monte Carlo analysis for very noisy behavior or a very large number of potentially significant variables. These would be cases where it is very difficult to associate the variation in results with the design variables and accordingly only quantifying the result is feasible.

29.5.2. Background

The required number of the simulation depends on:

1. Cost of creating an accurate metamodel
2. Cost of estimating the noise variation
3. Cost of observing low-probability events.

If the variation in the responses is mainly due to the variation of the design variables, then the cost of creating an accurate metamodel dominates. The region of interest for a robustness analysis will not be as large as to contain significant curvature; therefore a linear or preferably a quadratic response surface should suffice. In past design optimization work, the number of experiments was successfully taken to be 1.5 times the number of terms (unknowns) in the metamodel to be estimated. For a robustness analysis, being conservative at this point in time, a value of twice the number of terms is recommended. The number of terms for a linear model is $k+1$ with k the number of design parameters. The number of terms for a quadratic response surface is $(k+1)(k+2)/2$.

The variation in the responses may not be mainly due to the variation of the design variables. In this case, enough experiments much be conducted to estimate this noise variation with sufficient accuracy. This cost is additional to the cost of creating the metamodel. The number of experiments required will differ considering the criteria for accuracy used. For example, we can require the error of estimating the noise variation to be less than 10%; however, this requires about 150 experiments, which may be too expensive. Considering the practical constraints for vehicle crash analysis, useful results can be obtained with 25 or more degrees of freedom of estimating the noise variation. This gives a situation where the error bound on the standard deviation is about 20% indicating that it is becoming possible to distinguish the six sigma events from five sigma events.

For design purposes, the variation of the responses and the role of the design variables are of interest. High accuracy may be impossible due to lack of information or unreasonable computational costs. A metamodel-based investigation and 20 FE evaluations can achieve:

1. Investigate up to 10 variable
2. Quantify the contribution of each variable
3. Estimate if the scatter in results is admissible.

If the scatter in FE results is large, then the FE model must be improved, the product redesigned, or a more comprehensive probabilistic investigation performed. The study should indicate which is required.

A study can be augmented to re-use the existing FE evaluations in a larger study.

If higher accuracy is required, then for approximately 50 simulations one can compute:

- Better quantification of the role of the design variables: Investigate the effect of about five variables if a quadratic or neural network approximation is used or about 10 variables using linear approximations.
- Higher accuracy and better understanding of the scatter in the results. Predict effect of frequently occurring variation with a rare chance of being in error. Outliers may occur during the study and will be identified as such for investigation by the analyst. Structural events with a small (5% to 10%) probability of occurring might however not be observed.

The accuracy of these computations must be contrasted to the accuracy to which the variation of the design parameters is known. These limits on the accuracy, though important for the analyst to understand, should not prohibit useful conclusions regarding the probabilistic behavior of the structure.

29.5.3. Competing role of variance and bias

In an investigation the important design variables are varied while other sources are kept at a constant value in order to minimize their influence. In practice the other sources will have an influence. Distinguishing whether a difference in a response value is due to a deterministic effect or other variation is difficult, because both always have a joint effect in the computer experiments being considered.

In general [4] the relationship between the responses **y** and the variables **x** is:

$$y = f(\mathbf{x}) + \delta(\mathbf{x}) + \varepsilon,$$

with $f(\mathbf{x})$ the metamodel; $\delta(\mathbf{x}) = \eta(\mathbf{x}) - f(\mathbf{x})$, the bias, the difference between the chosen metamodel and the true functional response $\eta(\mathbf{x})$; and ε the random deviation.

The bias (fitting error) and variance component both contribute to the residuals. If we compute the variance of the random deviation using the residuals then the bias component is included in our estimate of the variance. The estimate of the variance is usually too large in the case of a bias error.

The bias error is minimized by:

1. Choosing the metamodel to be the same as the functional response. The functional response is however unknown. A reliable approach in the presence of noise is difficult to establish. In particular,

metamodels that can fit exactly to any set of points will fit to the noise thus erroneously stating that the random deviation is zero; inflexible metamodels will ascribe deterministic effects to noise.

2. Reducing the region of interest to such a size that the difference between metamodel and true functional response is not significant.
3. Large number of experimental points. This strategy should be used together with the correct metamodel or a sufficiently small region of interest.

The recommended approach is therefore to use a linear or quadratic response over a subregion small enough that the bias error should be negligible.

29.5.4. Confidence interval on the mean

For multiple regression, the 100(1-α)% confidence limits on the mean value at $\mathbf{X}_0$ are obtained from

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{\mathbf{X}_0 (\mathbf{X}^{'}\mathbf{X})^{-1} \mathbf{X}_0},$$

with s_{n-p}^2 an estimate to σ^2. At the center of the region of interest for the coded variables the confidence interval is

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{C_{11}},$$

with C_{11} the first diagonal element of $(\mathbf{X}^{'}\mathbf{X})^{-1}$. The confidence bound therefore depends on the variance of the response and the quality of the experimental design.

More details can be found in, for example, the text by Myers and Montgomery [11].

29.5.5. Confidence interval on a new evaluation

For multiple regression, the 100(1-α)% confidence limits on a new evaluation at $\mathbf{X}_0$ are obtained from

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{1 + \mathbf{X}_0 (\mathbf{X}^{'}\mathbf{X})^{-1} \mathbf{X}_0}.$$

The confidence interval for new observations of the mean is

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{1 + C_{11}},$$

In the following table we monitor the bounds for a new evaluation of the mean for a linear approximation using five design variables using a 95% confidence interval. The value of C_{11} is computed from D-optimal experimental designs generated using LS-OPT. The error bounds are close to 2σ for more than 25 existing runs (20 degrees of freedom).

Table 29-1: Bounds for a new evaluation of the mean for a linear approximation

n	**p**	**n-p**	**C_{11}**	**Bounds (σ=10% α=5%)**
10	6	4	0.104	±29%
15	6	9	0.070	±23%
20	6	14	0.051	±22%
25	6	19	0.041	±21%
30	6	24	0.034	±21%
50	6	44	0.020	±20%
100	6	94	0.010	±20%

29.5.6. Confidence interval on the noise (stochastic process) variance

The noise (stochastic process) variance can be estimated by considering the residuals of the reponse surface fit. Events such as a different buckling mode or order of contact events will appear in the residuals because they cannot be attributed to the variables in the response surface fit. These residuals can also be due to a bias (lack-of-fit) error, which complicates matters.

The error of estimating the noise variance (σ^2) is minimized by:

1. Large number of points
2. Minimizing the bias error. Ideally one wants to observe many occurrences of the same design.

The residual mean square

$$s^2 = \frac{1}{n-p}\sum_{i=1}^{n}(e_i - \bar{e})^2 = \frac{1}{n-p}\sum_{i=1}^{n} e_i^2,$$

estimates σ^2 with n - p degrees of freedom where n is the number of observations and p is the number of parameters including the mean.

We want to find an interval [b_1, b_2] such that $P[b_1 \le s^2 \le b_2] = 0.95$. We rewrite as $P\left[\frac{n-p}{\sigma^2}b_1 \le \frac{n-p}{\sigma^2}s^2 \le \frac{n-p}{\sigma^2}b_2\right] = 0.95$. We have $(n-p)s^2/\sigma^2$ is a chi-squared distribution with n – p degrees of freedom. From the chi-squared table we can get [a_1, a_2] such that $P\left[a_1 \le \frac{n-p}{\sigma^2}s^2 \le a_2\right] = 0.95$ by reading of the values for 0.975 and 0.025. Having [*a*₁, *a*₂] we can compute for [*b*₁, *b*₂] as $\left(\dfrac{s^2}{n-p}a_1, \dfrac{s^2}{n-p}a_2\right)$. The $100(1-\alpha)\%$ confidence interval on σ^2 is therefore

$$\left(\frac{(n-p)s^2}{\chi^2_{\alpha/2,n-p}}, \frac{(n-p)s^2}{\chi^2_{1-\alpha/2,n-p}}\right)$$

In the table below we monitor the error bounds on the variance for a problem with six parameters (including the mean).

Table 29-2: Error bounds on variance

Noise Variance Confidence Interval								
n	**n-p**	**Lower Bound**			**Value (s)**	**Upper Bound**		
		α=5%	**α=10%**	**α=20%**		**α=20%**	**α=10%**	**α=5%**
10	4	5.99	6.49	7.17	10	19.39	23.72	28.74
15	9	6.88	7.29	7.83	10	14.69	16.45	18.25
20	14	7.32	7.69	8.15	10	13.41	14.60	15.77
25	19	7.605	7.94	8.36	10	12.77	13.70	14.6
30	24	7.81	8.12	8.50	10	12.38	13.16	13.91
50	46	8.31	8.56	8.86	10	11.59	12.10	12.56
106	100	8.78	8.97	9.19	10	11.02	11.33	11.61
206	200	9.11	9.24	9.41	10	10.69	10.92	11.09

In the above it was assumed that the metamodel is a sufficiently accurate approximation to the mechanistic model (the bias error sufficiently small) and that the errors are normally distributed. In general the estimate of σ^2 will be depend on the approximation model. For a model-independent estimate, replicate runs (multiple observations for the same design) are required. If the bias error is significant then the estimate of σ^2 will usually be too large [12].

29.5.7. Probability of observing a specific failure mode

A large number of runs may be required to be sure that an event with a specific probability is observed.

1. Probability that the event will be observed at least once (one or more times):

2. P[observing 0 events] = (1-P[event])^n
3. P[observing 1 or more events] = 1.0 - (1-P[event])^n

Table 29-3: Number of runs required to observe an event with a specific probability

Probability of event	**Required number of runs for observing 1 or more occurrences at 95% probability**
0.45	5
0.26	10
0.14	20
0.095	30
0.06	50
0.03	100

29.6. Outlier analysis

Outliers are values in poor agreement with the values expected or predicted for a specific combination of design variable values. Unexpected values may occur due to different buckling modes or modeling problems. Understanding the cause of the outliers can therefore lead to an improved structure or structural model.

To be considered an outlier, the change in response value computed must not be completely explained by the change in design variable value. An expected value of the response value associated with a certain design is therefore required to judge whether a response is an outlier or not; the value predicted by the metamodel is used as the expected value.

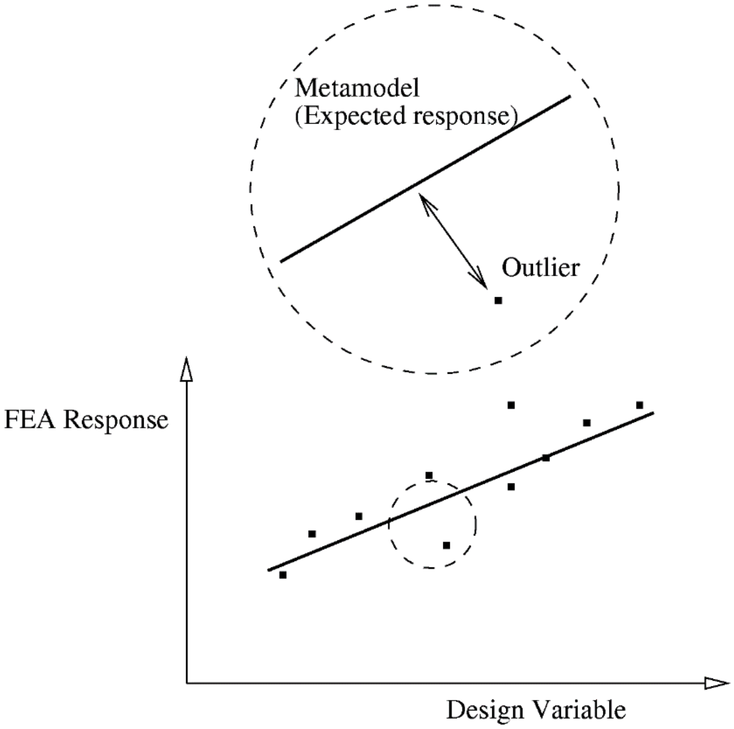


Figure 29-3: Outliers are identified after a metamodel has been fitted. Values in poor agreement of what is predicted by the design variables are considered outliers.

Metamodels are therefore useful to separate the effect of design variable changes from the other types of variation. The design variable cause-effect relationship is contained by the metamodel. The residuals of the fitting procedure are the variation not explained by changes in the design variables. The outliers therefore contain amongst others the bifurcation (buckling) effects.

The outliers are best investigated visually in LS-PrePost by considering the deformation of the structure using DynaStats. A useful metric is computing the outliers for the displacement of every node in the structure and to fringe plot the standard deviation of these outliers. Fringe plots of the index of the run associated with the maximum or minimum displacement outlier can be used to identify different buckling modes.

29.7. Stochastic contribution analysis

The variation of the response can be broken down in contributions from each design variable.

29.7.1. Linear estimation

The contribution can be estimated as:

$$\sigma_{g,i} = \partial G / \partial x \, \sigma_{x,i} ,$$

with $\sigma_{x,i}$ the standard deviation of the variable *i* and $\sigma_{g,i}$ the standard deviation of the variation of function *g* due to the variation of variable *i*.

The variance for all the variables is found as the sum of the variance:

$$\sigma_T^2 = \sum \sigma_i^2$$

where σ_T^2 is the variation of the response due to the variation of all the variables and σ_i^2 is the variation of response due to the variation of variable *i*. In the above it is assumed that the response is a linear response of the design variables and independent variables. If correlation between variables exists, then it is taken into account as documented in section 29.4.3.

29.7.2. Second and higher order estimation

For higher order effects, one must consider the interaction between different design variables as well as curvature. If a variation is due to the interaction of two variables, then the effect of one variable on the variation depends on the current value of the other. This is in contrast with problems described by first order effects, for which the effect of variables can be investigated independently; if interactions exist, this is no longer true.

The effect of a variable can be described in terms of its main or total effect. The main effect of a variable is computed as if it were the only variable in the system, while the total effect considers the interaction with other variables as well. The advantage of using the total effect is that the interaction terms, which can be significant, are included. For linear systems, the main and total effects are therefore the same. The second order effects must be computed, which increases computational costs considerably.

The variance of the response, assuming independent variables, can be written using the Sobol's indices approach [13] [14]. Firstly the function is decomposed as:

$$f(x_1,...,x_n) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n}\sum_{j=i+1}^{n} f_{ij}(x_i, x_j) + \ldots + f_{1,2,...,n}(x_1,...,x_n).$$

From which partial variances are computed as:

$$V_{i,...,j} = \int_0^1 \cdots \int_0^1 f_{i,...,j}^2(x_1,...,x_n)dx_{i.} \ldots dx_j\,,$$

with the variance of the response summed from the partial variances as:

$$V = \sum V_i + \sum_{i<j} V_{ij} + \ldots + V_{1,2,...,n}\,.$$

The sensitivity indices are given as:

$$\begin{aligned} S_i &= V_i / V\,, \quad 1 \le i \le n, \\ S_{ij} &= V_{ij} / V\,, \quad 1 \le i < j \le n, \\ S_{i,i+1,...,n} &= V_{i,i+1,...,n} / V\,. \end{aligned}$$

with the useful property that all of the sensitivity indices sum to 1:

$$\sum S_i + \sum_{i<j} S_{ij} + \ldots + S_{1,2,\ldots,n} = 1.$$

Using Monte Carlo, the main effect can be computed as

$$\hat{D}_i = \frac{1}{N}\sum_{m=1}^{N} f\left(x^{(1)}_{\sim im}, x^{(1)}_{\sim im}\right) f\left(x^{(2)}_{\sim im}, x^{(1)}_{\sim im}\right) - \hat{f}_0^{\,2}$$

with $x_{\sim i}$ is the subset of variables not containing x_i.

The total effect of a variable can also be computed as:

$$S_{Ti} = 1 - S_{\sim i}.$$

Using Monte Carlo, the total effect can be computed by considering the effects not containing x_i

$$\hat{D}_{\sim i} = \frac{1}{N}\sum_{m=1}^{N} f\left(\mathbf{x}^{(1)}_{\sim im}, x^{(1)}_{\sim im}\right) f\left(\mathbf{x}^{(1)}_{\sim im}, x^{(2)}_{\sim im}\right) - \hat{f}_0^{\,2}.$$

For second order response surfaces this can be computed analytically [15] as

$$\sigma_U^2 = \sum_{i\in U}^{n}\left[\beta_{ii}^2(m_{i,4} - \sigma_i^4) + (\beta_i + \beta_{ii}\mu_i + \sum_{j=1}^{n}\beta_{ij}\mu_j)^2\sigma_i^2 + (\beta_i + \beta_{ii}\mu_i + \sum_{j=1}^{n}\beta_{ij}\mu_j)\beta_{ii}m_{1,3}\right] + \sum_{i\in U}^{n}\sum_{i\in U,\, j\geq 1}^{n}\beta_{ij}^2\sigma_i^2\sigma_j^2,$$

with $m_{i,j}$ the *j*th moment about the mean of the distribution *i* and *U* the set of variables under consideration.

Remarks

1. The stochastic contribution is computed analytically only for responses surfaces. For neural networks, Kriging models, Support Vector Regression, and composite functions, Monte Carlo analysis is used. Many points (10,000 or more) are required. Note that a small number of points can results in negative values of the variance; these negative values should be small relative to the maximum variances obtained though and should be treated as zero. Inspecting the values printed for the effects of the variables should clarify the situation, because the effects are scaled values. The default of 10,000 points should give the 1 digit of accuracy needed to compare the effects of variables.
2. Correlations between variables are not considered in the computation of the main and total effects of the variables.

29.8. Reliability-based design optimization (RBDO)*

Reliability-based design optimization (RBDO) is the computation of an optimum design subject to probabilistic bounds on the constraints. The probabilistic bound is usually interpreted in the six-sigma context; for example, the failure of only one part in a million would be acceptable.

RBDO is currently done using First Order Second Moment (FOSM) method of computing the reliability. The requested minimum probability of failure is transformed to a number of standard deviations (sigmas) of the response, and the number of standard deviations (sigmas) is subsequently transformed into a safety margin used in the optimization process. The standard deviation of a response is computed analytically for response surfaces, and for the other metamodels and composites a second order local approximation is created to compute the standard deviation. See Section 29.4.5 for more detail regarding the First Order Second Moment (FOSM) method. The FOSM methodology is currently the default RBDO method, but more sophisticated methods may be available in future versions of LS-OPT.

Discrete variables are allowed in RBDO. The mixed-discrete optimization will be carried out considering the probabilistic bounds on the constraints.

The methods are described in more detail in Section 14.7 with an example in Section 22.3 illustrating the method.

Care must be taken in interpreting the resulting reliability of the responses. Accuracy can be especially poor at the tail ends of the response distribution. What constitutes engineering accuracy at the low probabilities is an open question. A definition such as six-sigma may be the best way of specifying the engineering requirement; a precise numerical value may not be meaningful. Accuracy at low probabilities requires firstly that the input data must be known accurately at these low probabilities, which may be prohibitively expensive to estimate.

29.9. Robust parameter design

Robust parameter design selects designs insensitive to changes in given parameters.

The field of robust design relies heavily on the work of Taguchi. Taguchi's insight was that it costs more to control the sources of variation than to make the process insensitive to these variations [16]. An alternate view of Taguchi [17] is that building quality into a product is preferable to inspecting for quality. Also, in simulation, the actual results of a robust system are more likely to conform to the anticipated results [16].

The robust design problem definition requires considering two sets of variables: (i) the noise variables causing the variation of the response and (ii) the control variables which are adjusted to minimize the effect of the noise variables. The method adjusts the control variables to find a location in design space with reduced gradients so that variation of the noise variable causes the minimum variation of the responses.

29.9.1. Fundamentals

The robustness of a structure depends on the gradient of the response function as shown in Figure 29-4. A flat gradient will transmit little of the variability of the variable to the response, while a steep gradient will amplify the variability of the variable. Robust design is therefore a search for reduced gradients resulting in less variability of the response.

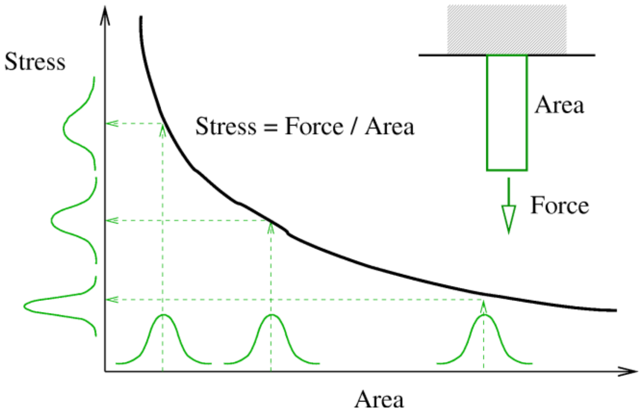


Figure 29-4: Robustness considering a single variable. Larger mean values of the area result in a smaller dispersion of the stress values. Note that the dispersion of the stress depends on the gradient of the stress-area relationship.

The variation of the response is caused by a number of variables, some which are not under the control of the designer. The variables are split in two sets of variables:

1. *Control variables.* The variables (design parameters) under the control of the designer are called control variables,
2. *Noise variables.* The parameter not under the control of the designer are called noise variables.

The relationship between the noise and control variables as shown in Figure 29-5 is considered in the selecting of a robust design. The control variables are adjusted to find a design with a low derivative with respect to the noise variable.

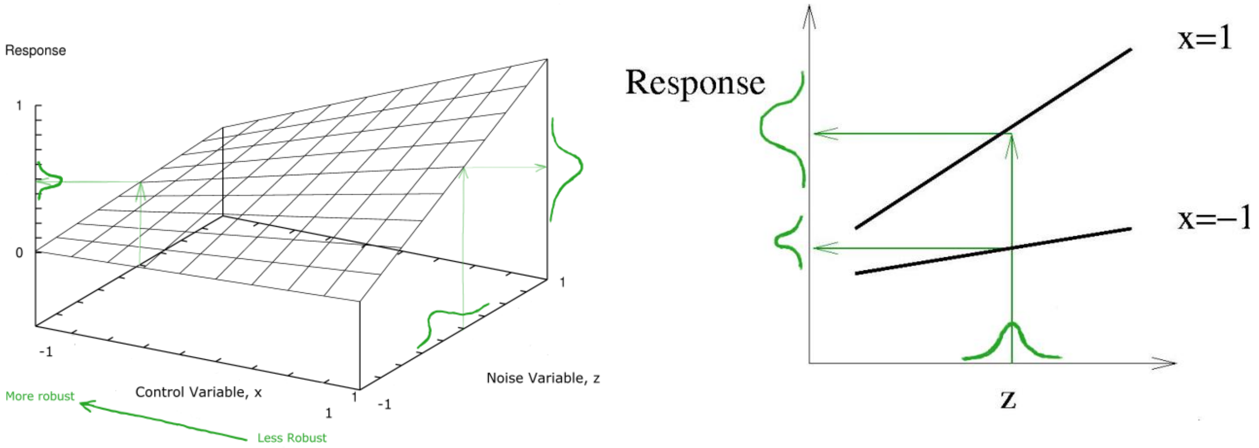


Figure 29-5: Robustness of a problem with both control and noise variables. The effect of the noise variable z on the response variation can be constrained using the control variable x. For robustness, the important property is the gradient of the response with respect to the noise variable. This gradient prescribes the noise in the response and can be controlled using the control variables. The gradient, as shown in the figure, is large for large values of the control variable. Smaller values of the control variable will therefore result in a more robust design, because of the lower gradient and accordingly less scatter in the response.

29.9.2. Methodology

The dual response surface method as proposed by Myers and Montgomery [11] using separate models for process mean and variance is considered. Consider the control variables **x** and noise variables **z** with $Var(\mathbf{z}) = \sigma_z^2 I_{r_z}$. The response surface for the mean is $E_z\left[y(x,z)\right] = \beta + x^{'}\beta + x^{'}\beta x$ considering that the noise variables have a constant mean. Response surface for variance considering only the variance of the noise variables is $Var_z[y(x,z)] = \sigma_z^2 l^{'}(x) l(x) + \sigma^2$ with $Var(\mathbf{z}) = \sigma_z^2 I_{r_z}$, σ^2 the model error variance, and l the vector of partial derivatives $l(x) = \partial y(x,z) / \partial z$.

The search direction required to find a more robust design is requires the investigation of the interaction terms $x_i z_j$. For finding an improved design, the interaction terms are therefore required. Finding the optimum in a large design space or a design space with a lot of curvature requires either an iterative strategy or higher order terms in the response surface.

For robust design, it is required to minimize the variance, but the process mean cannot be ignored. Doing this using the dual response surface approach is much simpler than using the Taguchi approach because multicriteria optimization can be used. Taguchi identified three targets: smaller is better, larger is better, and target is best. Under the Taguchi approach, the process variance and mean is combined into a single objective using a signal-to-noise ratio (SNR). The dual response surface method as used in LS-OPT does not require the use of a SNR objective. Fortunately so, because there is wealth of literature in which SNRs are criticized [11]. With the dual response surface approach both the variance and mean can be used, together or separately, as objective or constraints. Multicriteria optimization can be used to resolve a conflict between process variance and mean as for any other optimization problem.

Visualization is an important part of investigating and increasing robustness. As Myers and Montgomery state: “The more emphasis that is placed on learning about the process, the less important *absolute optimization* becomes.”

29.9.3. Experimental design

One extra consideration is required to select an experimental design for robust analysis: provision must be made to study the interaction between the noise and control variables. Finding a more robust design requires that the experimental design considers the $x_i z_j$ cross-terms, while the x_i^2 and z_j^2 terms can be included for a more accurate computation of the variance.

The crossed arrays of the Taguchi approach are not required in this response surface approach where both the mean value and variance are computed using a single model. Instead combined arrays are used which use a single array considering x and z combined.

29.10. Tolerance optimization

Mechanical parts often have associated tolerance values that determine whether a particular instance of a part is acceptable. A very tight tolerance could increase the associated engineering cost as well as the number of rejected parts. A very loose tolerance on the contrary can lead to undesired system behavior (e.g. loss of performance, failure etc.) and may also lead to mismatch of components during assembly. The effect of a fixed tolerance may also vary depending on the nominal design. Thus, it may be important to optimize the nominal design and the associated tolerance values simultaneously [18].

There can be several ways to account for tolerances. For example, one may choose to use a fixed value for the tolerances and only optimize the nominal design while accounting for those tolerances. Once the tolerances are fixed, the nominal design should be such that there should not be any possibility of failure within those limits. The possibility of failure can be determined by performing a Monte Carlo analysis around the nominal design with the distribution bounds defined based on the tolerance values. Other ways to account for the tolerances include simultaneous optimization of the absolute or the relative tolerances, while performing a worst case analysis to determine the possibility of failure based on a particular combination of nominal values and tolerances. The tolerance optimization problem can also be set up as a multiobjective optimization (e.g. Section 26.10) to simultaneously optimize other performance criteria and obtain a Pareto optimal front.

Tolerance optimization can involve a large number of function evaluations, which is greater than that required in RBDO (Section 29.8). It is so because RBDO with truncated variable distributions can be considered as a special case of tolerance optimization, in which the tolerance values associated with the design variables are known a priori. On the contrary, in the general case of tolerance optimization these values are also optimized. This results in more variables and therefore usually needs more samples. While all samples in RBDO have the same range for truncated variable distributions, during tolerance optimization different samples can have different variable distribution range and shape (Figure 29-6).

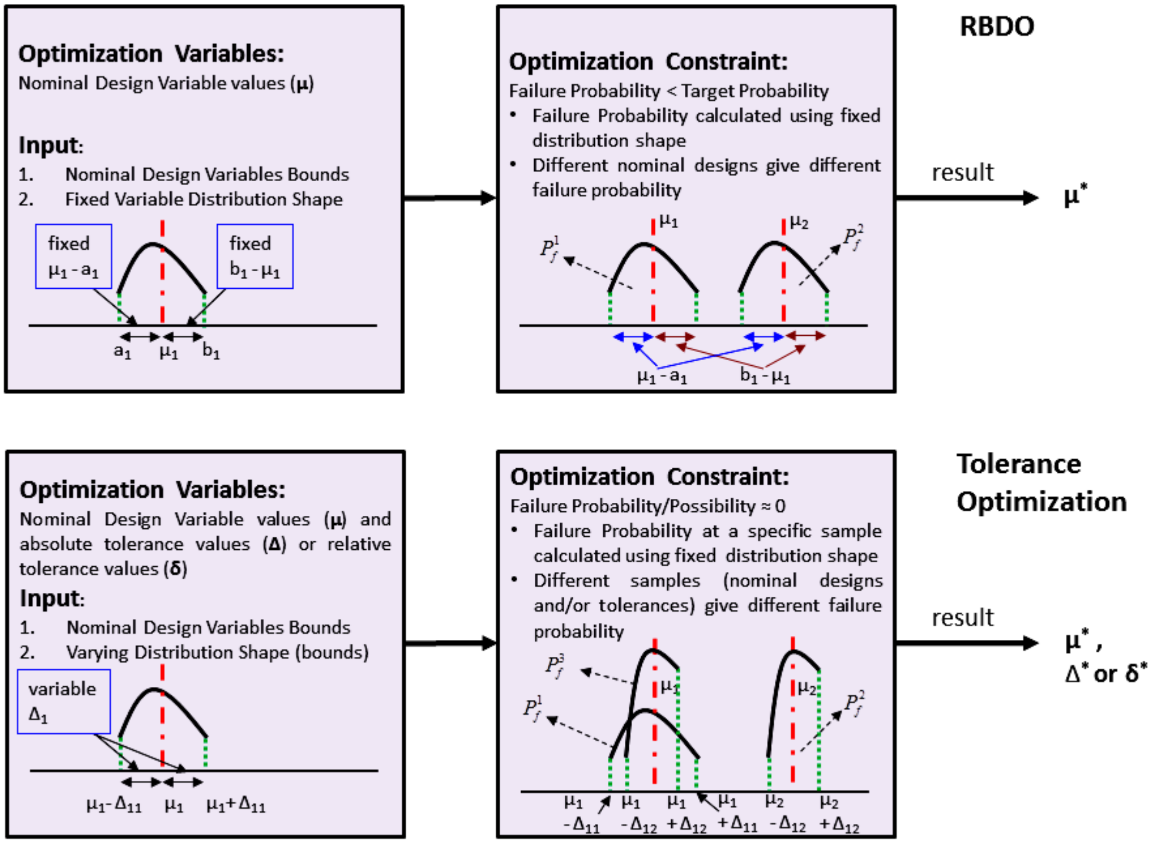

RBDO
Optimization Variables:
Nominal Design Variable values (μ)
Input:
1. Nominal Design Variables Bounds
2. Fixed Variable Distribution Shape
fixed
μ1 - a1
fixed
b1 - μ1
a1
μ1
b1
Optimization Constraint:
Failure Probability < Target Probability
• Failure Probability calculated using fixed distribution shape
• Different nominal designs give different failure probability
μ1
μ2
Pf1
Pf2
μ1 - a1
b1 - μ1
result
μ*
Tolerance Optimization
Optimization Variables:
Nominal Design Variable values (μ) and absolute tolerance values (Δ) or relative tolerance values (δ)
Input:
1. Nominal Design Variables Bounds
2. Varying Distribution Shape (bounds)
variable
Δ1
μ1-Δ11
μ1
μ1+Δ11
Optimization Constraint:
Failure Probability/Possibility ≈ 0
• Failure Probability at a specific sample calculated using fixed distribution shape
• Different samples (nominal designs and/or tolerances) give different failure probability
Pf3
Pf1
Pf2
μ1
μ2
μ1 -Δ11
μ1 -Δ12
μ1 +Δ12
μ1 +Δ11
μ2 -Δ12
μ2 +Δ12
result
μ*,
Δ* or δ*

Figure 29-6: Comparison of RBDO(top) and tolerance optimization (bottom).In tolerance optimization both the nominal value and the variable distribution range may vary from one sample to another. This is

in contrast to control variables in RBDO, for which only the mean can change. The right hand side figures demonstrate two samples for RBDO (top) and three samples for tolerance optimization (bottom).

Due to the usual high cost of tolerance optimization, in order to reduce the number of actual finite element simulations, the problem can be set up such that it consists of two steps.

- In the first step (optional), a deterministic optimization is performed based on the global metamodels generated using a limited number of finite element analyses. These global metamodels are then evaluated instead of the finite element models while performing the otherwise expensive tolerance-based optimization.
- The step 2 consists of a multilevel (Sections 19.7, 20.5.4) and possibly multi-objective tolerance optimization setup with design tolerance and mass of the selected design parts as objectives and the probability of failure close to zero as the constraint. As there is no further finite element simulation during this step, the computational time is significantly reduced.

29.11. REFERENCES

[1] Haldar, A, Mahadevan, S., Probability, Reliability and Statistical Methods in Engineering Design, Wiley, Inc. 2000.

[2] Tu, J., Choi, K.K., Design potential concept for reliability-based design optimization. Technical report R99-07. Center for Computer Aided Design and Department of Mechanical Engineering. College of engineering. University of Iowa. December 1999.

[3] Mendenhall, W., Wackerly, D.D., Scheaffer, R.L., Mathematical Statistics with Applications. PWS Kent, Boston, 1990.

[4] Kokoska, S., Zwillinger, D., CRC Standard Probability and Statistics Tables and Formulae, Student Edition. Chapman & Hall/CRC, New York, 2000.

[5] Basudhar, A. and Missoum, S., 2010. An improved adaptive sampling scheme for the construction of explicit boundaries. Structural and Multidisciplinary Optimization, 42(4), pp.517-529.

[6] G. Gary Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization.Journal ofMechanical Design, 129(4):370–380, 2007.

[7] Basudhar, A. and Missoum, S., 2009. Local update of support vector machine decision boundaries. In 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No (p. 2189).

[8] Lacaze, S. and Missoum, S., 2013, June. Reliability-based design optimization using kriging and support vector machines. In 11th International Conference on Structural Safety & Reliability.

[9] Basudhar, A., Dribusch, C., Lacaze, S. and Missoum, S., 2012. Constrained efficient global optimization with support vector machines. Structural and Multidisciplinary Optimization, 46(2), pp.201-221.

[10] Box, G.E.P., Draper, N.R., Empirical Model Building and Response Surfaces, Wiley, New York, 1987.

[11] Myers, R.H., Montgomery, D.C., Response Surface Methodology. Process and Product Optimization using Design Experiments. Wiley, 1995.

[12] Draper, N.R., Smith, H., Applied Regression Analysis, Second Edition, Wiley, New York, 1981.

[13] Sobol IM, Sensitivity analysis for nonlinear mathematical models, Mathematical Modeling and Computer Experiments, 1(4), pp. 407-413, 1993.

[14] Chan, K., Saltelli, A., Tarantola, S., Sensitivity analysis of model output: variance-based methods make the difference. Proceedings of the 1997 Winter Simulation Conference. 1997, Atlanta, GA.

[15] Chen, W., Jin, R., Sudjianto, A., Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty. Proceedings of DETC'04, Sept 28-October 2, 2004, Salt Lake City, Utah, USA.

[16] Sanchez, S.M., Robust design: seeking the best of all possible worlds. In Proceedings of the 2000 Winter Simulation Conference, ed Joines JA, Barton RR, Kan K, and Fishwick PA. 69-76, Institute of Electrical and Electronic Engineers. Piscataway, NJ.

[17] Roy RK. Design of Experiments Using the Taguchi Approach. Wiley, New York NY. 2001.

[18] Schjaer-Jacobsen, H.; Madsen, K., Algorithms for worst-case tolerance optimization, in Circuits and Systems, IEEE Transactions on , vol.26, no.9, pp.775-783, Sep 1979

[19] Lee, J., Johnson, G. E. (1993/09).Optimal tolerance allotment using a genetic algorithm and truncated Monte Carlo simulation. Computer-Aided Design 25(9): 601-611

IV – Appendix

Appendix A: LS-DYNA Binout Commands

A.1 Binout Histories

```
BinoutHistory –res_type res_type {-sub sub} –cmp component {-
invariant invariant –frame frame –id id (-name name) (-idi id)
(-namei name) –localid id1 id2 id3 (-localname name1 name2
name3) –pos position –side side}
```

Item	Description	Default	Remarks
`res_type`	Result type name	**-**	**1**
`sub`	Result subdirectory	–	1
`cmp`	Component of result	–	2
`invariant`	Invariant of results. Only `MAGNITUDE` is currently available.	–	3
`id`	ID number of entity	–	
`name`	Description (heading) of entity used as label	–	4
`pos`	Through thickness shell position at which results are computed.	1	5
`side`	Interface side for RCFORC data. `MASTER` or `SLAVE`.	`SLAVE`	
`frame`	`GLOBAL|GLOBAL_IN_REF|LOCAL`	`GLOBAL`	6
`localid`	3 Nodal ID's to define local coordinate axes	–	7
`localname`	3 Nodal names (headings) to define local coordinate axes	–	7
`id i`	Multiple ID's *i* = 1,2,3, … *n*	–	8
`name i`	Multiple headings as labels *i* = 1,2,3, … *n*	–	8

Remarks

1. The result types and subdirectories are as documented for the `*DATABASE_`*`OPTION`* LS-DYNA keyword.
2. The component names are as listed in Appendix B: LS-DYNA Binout Components.
3. The individual components required to compute the invariant will be extracted automatically; for example, "`–cmp displacement –invariant MAGNITUDE`" will result in the automatic extraction of the *x*, *y* and *z* components of the displacement.
4. The option "*`–name`*" that allows using the description/heading/name of the entity is valid only with *`nodout`* and *`elout`* result types.
5. For the *shell* and *thickshell* strain results the upper and lower surface results are written to the database using the component names such as `lower_eps_xx` and `upper_eps_xx`.
6. Distances and deformations can be computed in global coordinates, local coordinates or global coordinates in reference frame (*t* = 0), e.g. `–frame GLOBAL_IN_REF`. See Section 6.2.2.
7. The definition of a local coordinate system requires three reference nodes to define the system. These can be defined as integer ID's or as names labels (headings) for example `–localid 231 556 722` or `–localname Thirdnode xBegin xEnd`. The second and third nodes define the direction of the local *x*-axis.
8. Some entities such as deformations or distances require multiple node definitions (two in the case of deformation or distance), e.g. `–id1 529 –id2 718` or `–name1 Measured_node –name2 Reference_node`.

A.2 Averaging, filtering, and slicing Binout histories

```
BinoutHistory {history_options} {-filter filter_type
–filter_freq filter_freq –units units –ave_points ave_points
–start_time start_time –end_time end_time }
```

Item	Description	Default
`history_options`	All available history options	-
`filter_type`	Type of filter to use: `SAE` or `BUTT`	-
`filter_freq`	Filter frequency	60 cycles / time unit
`units`	`S`=seconds `MS`=milliseconds	`S`
`ave_points`	Number of points to average	-
`start_time`	Start time of history interval to extract using slicing	0

`end_time`	End time of history interval to extract using slicing	t_{max}

A.3 Binout Responses

```
BinoutResponse {history_options} –select selection
```

Item	Description	Default	Remarks
`history_options`	All available history options including averaging, filtering, and slicing.	-	
`Selection`	`MAX|MIN|AVE|TIME`	`TIME`	1

Remarks

1. The maximum, minimum, average, or value at a specific time must be selected. If selection is TIME, the *end_time* history value will be used. If *end_time* is not specified, the last value (end of analysis) will be used.

A.3.1 Binout injury criteria

```
BinoutResponse {history_options} –cmp cmp { –units units
–lengthunits lengthunits}
```

Item	Description	Default	Remarks
`history_options`	All available history options including filtering and slicing.	-	
`cmp`	`HIC15, HIC36`, or `CSI`	-	
`lengthunits`	`METER`=meter `MM`=millimeter	`METER`	`1`
`units`	`S`=seconds `MS`=milliseconds	`S`	`1`

Remarks

1. The length and time units are used to compute the gravity value based on 9.81 m/s^2

A.3.2 Bilinear FLD constraint

The values of the principle in-plane strains at the specified surface (upper/lower/center) are used for the FLD constraint.

DynaFLDg location p_1 p_2 ... p_n lcid
DynaFLDf location p_1 p_2 ... p_n l_1 u_1 l_2 u_2 … l_n u_n

The following must be defined for the model and FLD curve:

Table 29-4: DynaFLD item description

Item	**Description**
location	LOWER, CENTER or UPPER
p_1…p_n	Part numbers of the model. Omission implies the entire model.
lcid	Load curve ID
l_1 u_1 … l_n u_n	Load curve IDs to define lower (l) and upper (u) bounds of severity zones

Appendix B: LS-DYNA Binout Components

Airbag Statistics: ABSTAT

Component	Description
Volume	Volume
pressure	Pressure
internal_energy	Internal energy
dm_dt_in	Input mass flow rate
dm_dt_out	Output mass flow rate
total_mass	Mass
gas_temp	Temperature
density	Density
surface_area	Area
reaction	Reaction

Boundary Nodal Forces: BNDOUT

Component	Description
Subdirectory ***discrete/nodes***	
x_force	X-force
y_force	Y-force
z_force	Z-force
x_total	Total X-force
y_total	Total Y-force
z_total	Total Z-force
energy	Energy
etotal	Total Energy

Discrete Element Forces: DEFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
resultant_force	Resultant force
displacement	Change in length

Element Output: ELOUT

Component	Description
Subdirectory ***solid***	
sig_xx	XX-stress
sig_xy	XY-stress
sig_yy	YY-stress
sig_yz	YZ-stress
sig_zx	ZX-stress
sig_zz	ZZ-stress
yield	Yield function
effsg	Effective stress
eps_xx	XX-strain
eps_xy	XY-strain
eps_yy	YY-strain
eps_yz	YZ-strain
eps_zx	ZX-strain
eps_zz	ZZ-strain
Subdirectory ***beam***	
axial	Axial force resultant
shear_s	s-Shear resultant
shear_t	t-Shear resultant
moment_s	s-Moment resultant
moment_t	t-Moment resultant
torsion	Torsional resultant
Subdirectory ***shell***	
sig_xx	XX-stress
sig_yy	YY-stress
sig_zz	ZZ-stress
sig_xy	XY-stress
sig_yz	YZ-stress
sig_zx	ZX-stress
plastic_strain	Plastic strain
upper_eps_xx	XX-strain
lower_eps_xx	
upper_eps_yy	YY-strain
lower_eps_yy	
upper_eps_zz	ZZ-strain
lower_eps_zz	
upper_eps_xy	XY-strain
lower_eps_xy	
upper_eps_yz	YZ-strain
lower_eps_yz	
upper_eps_zx	ZX-strain
lower_eps_zx	

Element Output: ELOUT

Component	Description
Subdirectory ***thickshell***	
sig_xx	XX-stress
sig_yy	YY-stress
sig_zz	ZZ-stress
sig_xy	XY-stress
sig_yz	YZ-stress
sig_zx	ZX-stress
yield	Yield
upper_eps_xx	XX-strain
lower_eps_xx	
upper_eps_yy	YY-strain
lower_eps_yy	
upper_eps_zz	ZZ-strain
lower_eps_zz	
upper_eps_xy	XY-strain
lower_eps_xy	
upper_eps_yz	YZ-strain
lower_eps_yz	
upper_eps_zx	ZX-strain
lower_eps_zx	

Contact Entities Resultants: GCEOUT

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
force_magnitude	Force magnitude
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment
moment_magnitude	Moment magnitude

Global Statistics: GLSTAT

Component	Description
kinetic_energy	Kinetic energy
internal_energy	Internal energy
total_energy	Total energy
energy_ratio	Ratio
stonewall_energy	Stonewall energy
spring_and_damper_energy	Spring & Damper energy
hourglass_energy	Hourglass energy
sliding_interface_energy	Sliding interface energy
external_work	External work
global_x_velocity	Global x-velocity
global_y_velocity	Global y-velocity
global_z_velocity	Global z-velocity
system_damping_energy	System damping energy
energy_ratio_wo_eroded	Energy ratio w/o eroded
eroded_internal_energy	Eroded internal energy
eroded_kinetic_energy	Eroded kinetic energy

Joint Element Forces: JNTFORC

Component	Description
Subdirectory ***joints***	
x_force	X-force
y_force	Y-force
z_force	Z-force
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment
resultant_force	R-force
resultant_moment	R-moment
Subdirectory ***type0***	
d(phi)_dt	d(phi)/dt
d(psi)_dt	d(psi)/dt (degrees)
d(theta)_dt	d(theta)/dt (degrees)
joint_energy	joint energy
phi_degrees	phi (degrees)
phi_moment_damping	phi moment-damping
phi_moment_stiffness	phi moment-stiffness
phi_moment_total	phi moment-total
psi_degrees	psi (degrees)
psi_moment_damping	psi-moment-damping
psi_moment_stiffness	psi-moment-stiffness
psi_moment_total	psi-moment-total
theta_degrees	theta (degrees)
theta_moment_damping	theta-moment-damping
theta_moment_stiffness	theta-moment-stiffness
theta_moment_total	theta-moment-total

Material Summary: MATSUM

Component	Description
kinetic_energy	Kinetic energy
internal_energy	Internal energy
x_momentum	X-momentum
y_momentum	Y-momentum
z_momentum	Z-momentum
x_rbvelocity	X-rigid body velocity
y_rbvelocity	Y-rigid body velocity
z_rbvelocity	Z-rigid body velocity
hourglass_energy	Hourglass energy

Contact Node Forces: NCFORC

Component	Description
Subdirectory ***master_00001*** *and* ***slave_00001***	
x_force	X-force
y_force	Y-force
z_force	Z-force
pressure	Pressure
x	X coordinate
y	Y coordinate
z	Z coordinate

Nodal Point Response: NODOUT

Component	Description
Translational components	
x_displacement	X-displacement
y_displacement	Y-displacement
z_displacement	Z-displacement
x_velocity	X-velocity
y_velocity	Y-velocity
z_velocity	Z-velocity
x_acceleration	X-acceleration
y_acceleration	Y-acceleration
z_acceleration	Z-acceleration
x_coordinate	X-coordinate
y_coordinate	Y-coordinate
z_coordinate	Z-coordinate
Rotational components	
rx_acceleration	XX-rotational acceleration
rx_displacement	XX-rotation
rx_velocity	XX-rotational velocity
ry_acceleration	YY-rotational acceleration
ry_displacement	YY-rotation
ry_velocity	YY-rotational velocity
rz_acceleration	ZZ-rotational acceleration
rz_displacement	ZZ-rotation
rz_velocity	ZZ-rotational velocity
Injury coefficients	
CSI	Chest Severity Index
HIC15	Head Injury Coefficient (15 ms)
HIC36	Head Injury Coefficient (36 ms)
Kinematics	
x_deformation	X-deformation
y_deformation	Y-deformation
z_deformation	Z-deformation
x_distance	X-distance
y_distance	Y-distance
z_distance	Z-distance

Nodal Forces: NODFOR

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
x_total	X-total force
y_total	Y-total force
z_total	Z-total force
energy	Energy
etotal	Total Energy

Rigid Body Data: RBDOUT

Component	Description
Translational components	
global_dx	X-displacement
global_dy	Y-displacement
global_dz	Z-displacement
global_vx	X-velocity
global_vy	Y-velocity
global_vz	Z-velocity
global_ax	X-acceleration
global_ay	Y-acceleration
global_az	Z-acceleration
global_x	X-coordinate
global_y	Y-coordinate
global_z	Z-coordinate
local_dx	Local X-displacement
local_dy	Local Y-displacement
local_dz	Local Z-displacement
local_vx	Local X-velocity
local_vy	Local Y-velocity
local_vz	Local Z-velocity
local_ax	Local X-acceleration
local_ay	Local Y-acceleration

local_az	Local Z-acceleration
Rotational components	
global_rax	X-acceleration
global_ray	Y-acceleration
global_raz	Z-acceleration
global_rdx	X-rotation
global_rdy	Y-rotation
global_rdz	Z-rotation
global_rvx	X-velocity
global_rvy	Y-velocity
global_rvz	Z-velocity
local_rdx	Local X-rotation
local_rdy	Local Y-rotation
local_rdz	Local Z-rotation
local_rvx	Local X-velocity
local_rvy	Local Y-velocity
local_rvz	Local Z-velocity
local_rax	Local X-acceleration
local_ray	Local Y-acceleration
local_raz	Local Z-acceleration
Direction cosines	
dircos_11	11 direction cosine
dircos_12	12 direction cosine
dircos_13	13 direction cosine
dircos_21	21 direction cosine
dircos_22	22 direction cosine
dircos_23	23 direction cosine
dircos_31	31 direction cosine
dircos_32	32 direction cosine
dircos_33	33 direction cosine
Injury coefficients	
CSI	Chest Severity Index
HIC15	Head Injury Coefficient (15 ms)
HIC36	Head Injury Coefficient (36 ms)

Reaction Forces: RCFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
mass	Mass

RigidWall Forces: RWFORC

Component	Description
Subdirectory forces	
normal_force	normal
x_force	X-force
y_force	Y-force
z_force	Z-force

Section Forces: SECFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment
x_centroid	X-center
y_centroid	Y-center
z_centroid	Z-center
total_force	Resultant force
total_moment	Resultant moment
area	Area

Single Point Constraint Reaction Forces: SPCFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
x_resultant	Total X-force
y_resultant	Total Y-force
z_resultant	Total Z-force
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment

Spotweld and Rivet Forces: SWFORC

Component	Description
axial	Axial force
shear	Shear force
failure_flag	Failure flag

Appendix C: LS-DYNA D3Plot Commands

C.1 D3Plot histories

```
D3PlotHistory –res_type res_type {-sub sub} –cmp
component {-
id id -pos position –pids part_ids –loc ELEMENT|NODE –
select_in_region selection –coord x y z –setid setid –
tref
ref_state}
```

Item	Description	Default	Remarks
`res_type`	Result type name	-	1
`cmp`	Component of result	-	1
`id`	ID number of entity	-	2
`pos`	Through thickness shell position	1	
`pids`	One or more part ids.	-	3
`loc`	Locations in model. `ELEMENT` or `NODE`.	-	4
`select_in_region`	`MAX|MIN|AVE`	`-select`	5
`coord`	Coordinate of a point for finding nearest element	-	6
`tref`	Time of reference state for finding nearest element	0.0	6
`setid`	ID of `*SET_SOLID_GENERAL` in LS-DYNA keyword file	-	6

Remarks

1. The result types and components are similar to what is used in LS-PREPOST. The result types and component names are listed in Appendix D.
2. For histories, the *-id* option is mutually exclusive with the *–select_in_region* option.
3. If part ids are specified, the extraction will be done over these parts only. If no part ids and no element or node id are specified, then the extraction will be done considering the whole model.
4. Element results such as stresses will be averaged in order to create the NODE results. Nodal results such as displacements cannot be requested as ELEMENT results.
5. The maximum, minimum, or average over a part can be selected. The *–select_in_region* option is mutually exclusive with the *–id* option. The default value is that of the d3plot response *-select* argument which in turn defaults to MAX.
6. An *x,y,z* coordinate can be selected. The quantity will be extracted from the element nearest to *x,y,z* at time *tref*. Only elements included in the `*SET_SOLID_GENERAL` element set are considered (only the `PART` and `ELEMENT` options).

C.1.1 Slicing D3Plot histories

Slicing of D3Plot histories is possible. Averaging and filtering are not available for D3Plot results.

```
D3PlotHistory {history_options} {-start_time start_time
-end_time end_time }
```

Item	Description	Default
`history_options`	All available history options	-
`start_time`	Start time of history interval to extract using slicing	0
`end_time`	End time of history interval to extract using slicing	$t_{\max}$

C.1.2 D3Plot FLD results

If FLD results are requested then the FLD curve can be specified using (*i*) the *t* and *n* coefficients or (*ii*) a curve in the LS-DYNA input deck. The interpretation of the t and n coefficients is the same as in LS-PREPOST.

```
D3PlotHistory {history_options} {-fld_t fld_t -fld_n
fld_n -
fld_curve fld_curve}
```

Item	Description	Default
`history_options`	All available history options	-
`fld_t`	Fld curve t coefficient	-
`fld_n`	Fld curve n coefficient	-
`fld_curve`	ID of curve in the LS-DYNA input deck	-

C.2 D3Plot responses

A response is extracted from a history – all the history options are therefore applicable and options required for histories are required for responses as well.

```
D3PlotResponse {history_options} –select selection
```

Item	Description	Default	Remarks
`history_options`	All available history options	-	
`select`	`MAX|MIN|AVE|TIME`	`TIME`	1

Remarks

1. The maximum, minimum, average, or value at a specific time must be selected. If *select* is `TIME` then the *end_time* history value will be used. If *end_time* is not specified, the last value (end of analysis) will be used. If the selection must be done over parts as well, then this option is used together with the –*select_in_region* argument as documented for d3plot histories; firstly the maximum, minimum, or average value will be selected for the part as specified by the –*select_in_region* argument, followed by the selection of the maximum, minimum, or average over time as specified by the –*select* argument.

C.3 D3Plot multihistories

Evaluation of histories at multiple coordinate points at the same time is supported. This will generate a multipoint history.

```
D3PlotMultipointHistory {history_options} –coordtype
coordtype –coordfile coordfile –align alignment_name –
cluster [0/1] –tol tolerance_value –select selection
```

Item	Description	Default	Remarks
`coordtype`	Coordinate types `gom` or `file`	-	1
`coordfile`	Name of coordinate file in LS-DYNA `*NODE` format	-	-
`align`	Name of alignment reference	-	2
`cluster`	Creates a one-to-one map of the points and their nearest nodes. 0 or 1.	0	3
`tol`	Proximity tolerance. Points outside the tolerance are ignored.		4

Remarks

1. There are two coordinate type options available, `gom` and `file`. `gom` interfaces to the output of the optical measurement system gom/ARAMIS. In this case the coordinate file is automatically determined from the relevant file multihistory. In the `file` option, the coordinates for evaluation are specified in a text file in the LS-DYNA `*NODE` format, Section 6.19.
2. The simulation geometry (Finite Element mesh) and the coordinates specified in `coordfile` may differ, so a transformation may be specified using the `align` directive, Section 6.2.3.
3. A Nearest Neighbor cluster can be created for efficiency and accuracy. This is typically used to reduce the number of source points to a manageable number, Section 6.2.3
4. A proximity tolerance can be specified to ignore points which are too far away from the nearest node. This option is executed after creation of the cluster to potentially further reduce its size.

Appendix D: LS-DYNA D3Plot Components

The table contains component names for element variables. The result type and component name must be specified in the "*D3Plot*" interface commands to extract response variables.

Result Type	Number	Description	Component name
Stress	1	*xx*, *yy*, *zz*, *xy*, *yz*, *zx* stress	xx_stress
	2		yy_stress
	3		zz_stress
	4		xy_stress
	5		yz_stress
	6		zx_stress
	7	Effective plastic strain	plastic_strain
	8	Pressure or average strain	pressure
	9	von Mises stress	von_mises
	10	First principal deviator maximum	1st_prin_dev_stress
	11	Second principal deviator	2st_prin_dev_stress
	12	Third principal deviator minimum	3rd_prin_dev_stress
	13	Maximum shear stress	max_shear_stress
	14	1st principal maximum stress	1st_principal_stress
	15	2nd principal stress	2st_principal_stress
	16	3rd principal min	3st_principal_stress
Ndv	17	*x*-displacement	x_displacement
	18	*y*-displacement	y_displacement

Result Type	Number	Description	Component name
	19	z-displacement	z_displacement
	20	Displacement magnitude	result_displacement
	21	x-velocity	x_velocity
	22	y-velocity	y_velocity
	23	z-velocity	z_velocity
	24	Velocity magnitude	result_velocity
	64	xy-displacement	xy_displacement
	65	yz-displacement	yz_displacement
	66	zx-displacement	zx_displacement
Result	26	M_{xx} bending resultant	Mxx_bending
	27	M_{yy} bending resultant	Myy_bending
	28	M_{xy} bending resultant	Mxy_bending
	29	Q_{xx} shear resultant	Qxx_shear
	30	Q_{yy} shear resultant	Qyy_shear
	31	N_{xx} normal resultant	Nxx_normal
	32	N_{yy} normal resultant	Nyy_normal
	33	N_{xy} normal resultant	Nxy_normal
	34	Surface stress $N_{xx}/t + 6M_{xx}/t^2$	Nxx/t+6Mxx/t^2
	35	Surface stress $N_{xx}/t – 6M_{xx}/t^2$	Nxx/t-6Mxx/t^2
	36	Surface stress $N_{yy}/t – 6M_{yy}/t^2$	Nyy/t-6Myy/t^2
	37	Surface stress $N_{yy}/t + 6M_{yy}/t^2$	Nyy/t+6Myy/t^2
	38	Surface stress $N_{xy}/t – 6M_{xy}/t^2$	Nxy/t+6Mxy/t^2
	39	Surface stress $N_{xy}/t + 6M_{xy}/t^2$	Nxy/t+6Mxy/t^2
	40	Effective upper surface stress	u_surf_eff_stress
	41	Effective lower surface stress	l_surf_eff_stress
Strain	43	Lower surface effective plastic strain	l_surf_plastic_strain

Result Type	Number	Description	Component name
	44	Upper surface effective plastic strain	u_surf_plastic_strain
	45	Lower surface *xx*, *yy*, *zz*, *xy*, *yz*, *zx* strain	l_surf_xx_strain
	46		l_surf_yy_strain
	47		l_surf_zz_strain
	48		l_surf_xy_strain
	49		l_surf_yz_strain
	50		l_surf_zx_strain
	51	Upper surface *xx*, *yy*, *zz*, *xy*, *yz*, *zx* strain	u_surf_xx_strain
	52		u_surf_yy_strain
	53		u_surf_zz_strain
	45		u_surf_xy_strain
	55		u_surf_yz_strain
	56		u_surf_zx_strain
	57	Middle surface *xx*, *yy*, *zz*, *xy*, *yz*, *zx* strain	m_surf_xx_strain
	58		m_surf_yy_strain
	59		m_surf_zz_strain
	60		m_surf_xy_strain
	61		m_surf_yz_strain
	62		m_surf_zx_strain
	69	Lower, upper, middle principal + effective strains	l_surf_max_princ_strain
	70		l_surf_2nd_princ_strain
	71		l_surf_min_princ_strain
	72		l_surf_effective_princ_strain
	73		u_surf_max_princ_strain
	74		u_surf_2nd_princ_strain
	75		u_surf_min_princ_strain

Result Type	Number	Description	Component name
	76		u_surf_effective_princ_strain
	77		m_surf_max_princ_strain
	78		m_surf_2nd_princ_strain
	79		m_surf_min_princ_strain
	80		m_surf_effective_princ_strain
Misc	25	Temperature	temperature
	63	Internal energy density	internal energy
	67	Shell thickness	shell_thickness
	68	Shell thickness reduction (%)	%_thickness_reduction
	81	History variable 1	history_var#1
FLD	501	Lower, upper, middle, maxima surface eps1/fldc	lower_eps1/fldc
	502		upper_eps1/fldc
	503		middle_eps1/fldc
	504		maxima_eps1/fldc
	505	Lower, upper, middle, maxima surface fldc-eps1	lower_fldc-eps1
	506		upper_ fldc-eps1
	507		middle_ fldc-eps1
	508		maxima_ fldc-eps1
	509	Lower, upper, middle, maxima surface eps1	lower_ eps1
	510		upper_ eps1
	511		middle_ eps1
	512		maxima_ eps1
	513	Lower, upper, middle, maxima surface eps2	lower_ eps1
	514		upper_ eps1
	515		middle_ eps1
	516		maxima_ eps1

Result Type	Number	Description	Component name
Beam	701	Axial Force	axial_force
	702	S Force	s_force
	703	T Force	t_force
	704	SS Moment	ss_moment
	705	TT Moment	tt_moment
	706	Torsion	torsion
	707	Axial_stress	axial_stress
	708	RS Shear Stress	rs_shear_stress
	709	TR Shear Stress	tr_shear_stress
	710	Plastic Strain	plastic_strain
	711	Axial strain	axial_strain

Appendix E: Database and Output Files

E.1 Directory structure

When running an optimization, LS-OPT will generate a directory in the work directory for each sampling and for each stage using the sampling or stage name, respectively. If a sampling and a stage have the same name, the same directory will be used.

In the stage directories a subdirectory will be created for each simulation. These subdirectories, the run directories, are named *mmm.nnnn*, where *mmm* represents the iteration number and *nnnn* is a number starting from 1.

An example of a subdirectory name, defined by LS-OPT, is *side_impact/3.11*, where *3.11* represents the design point number 11 of iteration 3, and *side_impact* is the stage name. The creation of subdirectories is automatic, and the user only needs to deal with the working directory.

The work directory needs to contain at least the lsopt file.

Figure E - 1 visualizes the LS-OPT directory structure. In the case of simulation runs being conducted on remote nodes, a replica of the run directory is automatically created on the remote machine. The response.n, history.n and multihistory.n files will automatically be transferred back to the local run directory at the end of the simulation run. These are the only files required by LS-OPT for further processing. More files can be transferred back by using the recover files options, see Section 5.5.4.

If some of the stages are of type LSOPT (Section 5.3.9) then the subdirectories mmm.nnnn act as the working directories for inner level LS-OPT processes. Therefore, these directories have further sublevel directories. In Figure E - 2, the directory structure is shown for multilevel optimization with a single LSOPT stage named ‘Stage 3’.

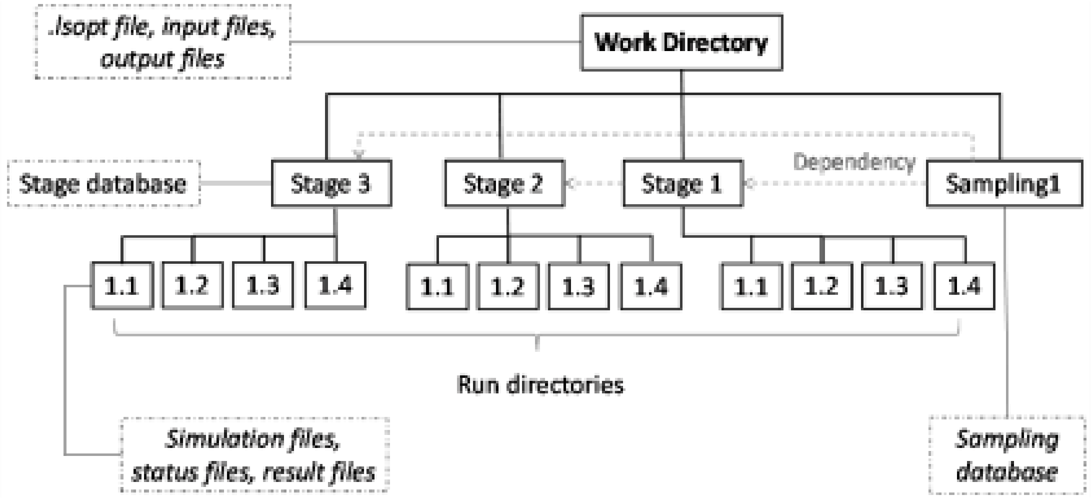


Figure E - 1: Directory structure in LS-OPT.

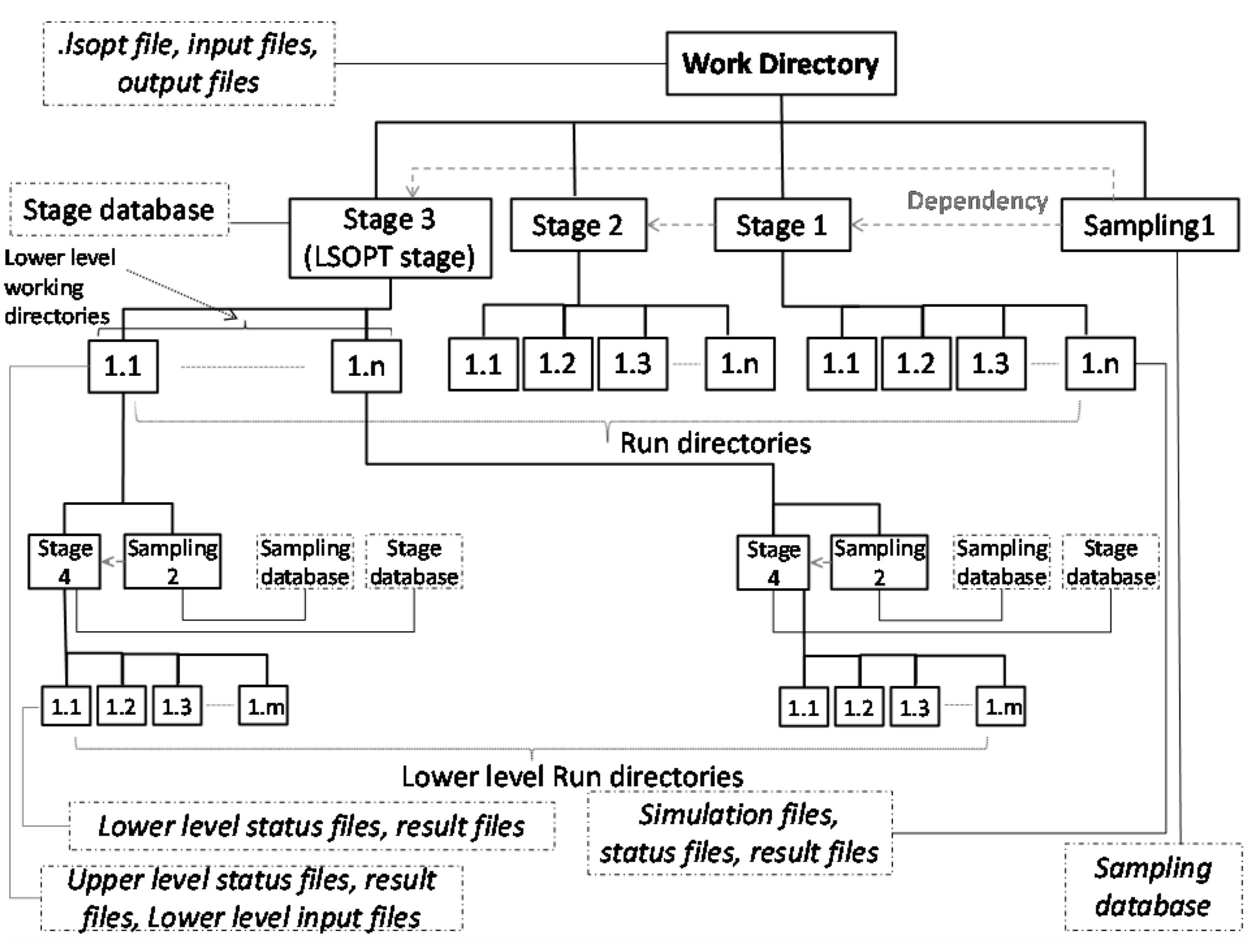

.lsopt file, input files, output files
Work Directory
Stage database
Stage 3 (LSOPT stage)
Stage 2
Stage 1
Dependency
Sampling1
Lower level working directories
1.1
1.n
1.2
1.3
Run directories
Stage 4
Sampling 2
Sampling database
Stage database
1.m
Lower level Run directories
Lower level status files, result files
Simulation files, status files, result files
Upper level status files, result files, Lower level input files
Sampling database

Figure E - 2: Directory structure for multilevel optimization with one LSOPT stage.

E.2 Design flow

Source Database file	**Task**	**Output Database file**	**Directory for output database**
Command file (`.lsopt`)	Point selection	`Experiments_n.csv`	Sampling
`Experiments_n.csv`	Simulation runs	`Solver output files`	Run

Solver output files	Result extraction. Assemble Sampling Results. Compute extended results.	`AnalysisResults_n.lsda`
		`AnalysisResults_n.csv`
		`ExtendedResults.n`
			Sampling

`AnalysisResults_n.csv` `AnalysisResults_n.lsda`	Approximation	`DesignFunctions.n` `VirtualHistories.n`	Sampling

`DesignFunctions.n`
`VirtualHistories.n`
	Optimize	`OptimumResults.n`
		`OptimizationHistory`
		`OptimizerHistory_n.csv`
		`lsopt_results_n.csv`
			Work

E.3 Output files

The following files are intermediate database files containing ASCII data.

E.3.1 Intermediate database files

Database file	**Description**	**Directory**	**Reference**
`AnalysisResults_`*`n`*`.lsda`	Binary file containing all the extracted results including responses, matrices and histories.	Sampling	
`AnalysisResults_`*`n`*`.csv`	Trial designs and the responses extracted	Sampling	

	from the solver database.		
ANOVA.*n*	ANOVA data used in the Viewer Sensitivity feature.	Sampling	
DesignFunctions.*n*	Parameters of the approximate functions	Sampling	E.4.4
Experiments_*n*.csv	Trial designs computed as a result of the experimental design	Sampling	E.4.2
ExtendedResults.*n*	All variables, responses and extended results at each trial design point	Sampling	E.4.7
ExtendedResults_*n*.lsda	Binary file	Sampling	
ExtendedResultsMaster.*n*	ExtendedResults for a master case (sampling). Cases with exactly the same experimental design are grouped in a master case.	Sampling	
ExtendedResultsMaster_*n*.csv	ExtendedResults for a master case (sampling).	Sampling	
ExtendedResultsMaster_*n*.lsda	Binary file for a master case (sampling)	Sampling	
PCA_histories.*n*		Sampling	
PCA_multiresponse.*n*		Sampling	
PRESS_predictions_*n*.lsda		Sampling	
S_AnalysisResults	Temporary intermediate file	Sampling	
SubRegionBounds.*n*	Bounds of the subregion.	Sampling	

VariableMap_*n*.lsox	Variable and response connectivity for each sampling.	Sampling	
VariableMapMaster_*n*.lsox	Variable and response connectivity for master sampling.	Sampling	
VirtualHistoryFunction.*n*	Approximation functions data for histories	Sampling	E.4.5
dh_history.dat	MOO data	Work	
lsopt_results_*n*.binout	MOO data	Work	E.4.10
OptimizationHistory	Variable, response and error history of the successive approximation process	Work	E.4.6
OptimizerHistory	Detailed history of the optimizer	Work	
OptimumResults.*n*	Optimum for a particular iteration	Work	E.4.8
OptimumResults_*n*.lsda	Optimal variables, histories and responses for iteration *n*	Work	
OptiState_*n*.lsox	Optimization state	Work	
Sobol_GSA.*n*	Sobol data	Work	E.4.9
SubRegion_*n*.lsox	Subregion data	Work	
Variables.*n*	The variable values, confidence intervals and bounds	Work	

E.3.2 Database files in .csv (comma separated variables) format

Database file	Description	Directory	Remarks
`Experiments_n.csv`	Experiments (*n* = iteration number)	Sampling	E.4.2
`AnalysisResults_n.csv`	Analysis Results	Sampling	E.4.3
`ExtendedResultsMaster_n.csv`	Extended Results (variables, dependents, responses, composites, objectives, constraints, multiobjective)	Sampling	
`ExtendedResultsMETA_n.csv`	Extended Results file for user-defined Experiments file	Work	Section 8.5.1
`OptimizationHistory.csv`		Work	
`OptimizerHistory_n.csv`	Detailed history of the optimizer for iteration *n*	Work	
`MOOPerformanceMetrics.csv`			

E.3.3 Binary database files

Database file	Description	Directory
`AnalysisResults_n.lsda`	Analysis results	Sampling
`ExtendedResults_n.lsda`	Extended results	Sampling
`ExtendedResultsMaster_n.lsda`	Extended results for master case	Sampling
`PRESS_predictions_n.lsda`	PRESS (Section 23.3.4) predicted results and PRESS residuals (Polynomials and Radial Basis Function networks (Section 24.1.2) only. PRESS residuals are not computed for Feedforward Neural Networks)	Sampling

`VirtualHistoryFunction_`*`n`*`.lsda`	Metamodel-based histories	Sampling
`OptimumResults_`*`n`*`.lsda`	Optimum results for each iteration	Work
`MOOPerformanceMetrics.binout`		
`lsopt_results_`*`n`*`.binout`	All variable, response and extended results of the non-dominated solutions at each iteration	Work

E.3.4 Log files and status files

Status files `started`, `finished`, `multihistory.`*n*, `history.`*n*, `response.`*n*, multiresponse.*n* and `EXIT_STATUS` are placed in the run directories to indicate the status of the solution progress. The directories can be cleaned to free disk space but selected status files must remain intact to ensure that a restart can be executed if necessary.

A brief explanation is given below.

Table 29-5: Status and log files generated by LS-OPT

File	**Description**	**Directory**

`job_log`
`job_log.extractor`
`job_log.solver`
	The simulation run/extraction log is saved in that file in the local run directory.	Simulation home directory

`job_command`
`job_command.extractor`
`job_command.solver`
	Contains the command of the job executed.
	Contains .bat extension when executed on Windows.
		Simulation home directory

`started`
`started.extractor`
`started.solver`
	The run has been started.	Simulation home directory

`finished`
`finished.extractor`
`finished.solver`
	The run has been completed. The completion status is given in the file.	Simulation home directory

response.*n*	Response number *n* has been extracted.	Simulation home directory
history.*n*	History number *n* has been extracted.	Simulation home directory
multiresponse.*n*	Multipoint response number *n* has been extracted	Simulation home directory
multihistory.*n*	Multipoint history number *n* has been extracted	Simulation home directory
lsopt_db	This file communicates the current status of the LSOPT databases to other LSTC programs. The content of this file is subject to change between versions of LS-OPT, E.4.11 .	Project directory
EXIT_STATUS	Error message after termination. The user interface LS-OPT*ui* uses the message in the EXIT_STATUS file as a pop-up message.	Project directory
lfop.log	The file contains a log of the core optimization solver solution.	Project directory
lscheduler.debug	This file is generated by the *lscheduler* executable and is used for debugging purposes.	Project directory
lsopt.debug	Traceback of the solver termination. Used for debugging purposes.	Project directory
lsopt_engine.log		Project directory
lsopt_output	Results and some logging information. Usually a very large file.	Project directory
lsopt_input	Input in a formatted style	Project directory
lsopt_report	A final report of the analysis results. Available for some of the main tasks and most of the Repair tasks	Project directory
lsopt.history		Project directory

lsopt_last_progress	Project directory
lsopt.log	Project directory
lsopt.psinfo	Project directory
process.lsox	Project directory
resource.lsox	Project directory
schedconfig.lsox	Project directory

Remark

For the job_log, job_command, started and finished files, two files are created for each. The solver extension refers to the solver run while the extractor extension refers to the extractor run. The extractor run is a second pass which does not invoke the solver (the so called "no job" mode) but only performs secondary extraction.

E.4 Database file formats

The database consists of text files, text files with comma separated values (.csv format), xml files and binary files. The .csv files have three header lines. The first designates the version name, the second represents the variable names and the third represents the variable types. Variable names are provided for clarity (e.g. the user can import the file into a spreadsheet program) and to verify the consistency between the command file and the run database. The variable types are explained in the table below. The symbol sk is used to ignore certain columns, e.g. the first one which simply contains the point number.

E.4.1 Variable types

Symbol	Explanation
dv	Design variable
nv	Noise variable
dc	Discrete variable
st	String variable
rs	Response
sk	Ignore this column

E.4.2 The Experiments_n.csv file

This file appears in the sampling directory and is used to save the experimental point coordinates for the analysis runs. The file consists of header lines and data lines repeated for each experimental point.

```
lsopt_version 4.1
"Point","tbumper","thood",
"sk","dv","dv",
1,3.0000000000000000e+00,1.0000000000000000e+00,
2,5.0000000000000000e+00,1.0000000000000000e+00,
3,1.0000000000000000e+00,1.0000000000000000e+00,
4,1.0000000000000000e+00,5.0000000000000000e+00,
5,5.0000000000000000e+00,5.0000000000000000e+00,
```

E.4.3 The AnalysisResults_n.csv file

This file is used to save the responses at the experimental design points and appears in the sampling directory. Every line describes an experimental point and gives the variable and response values at the experimental point. The file consists of two header as well as data lines repeated for each experimental point.

```
lsopt_version 4.1
"Point","tbumper","thood","Disp2","Disp1"
"sk","dv","dv","rs","rs"
1,3.0000000000000000e+00,1.0000000000000000e+00,-
7.3670259999999996e+02,-1.6103350000000000e+02
2,5.0000000000000000e+00,1.0000000000000000e+00,-
7.3311230000000000e+02,-1.5946590000000000e+02
3,1.0000000000000000e+00,1.0000000000000000e+00,-
7.4418650000000002e+02,-1.6168279999999999e+02
4,1.0000000000000000e+00,5.0000000000000000e+00,-
6.4731250000000000e+02,-1.5394180000000000e+02
5,5.0000000000000000e+00,5.0000000000000000e+00,-
6.1158939999999996e+02,-1.6078149999999999e+02
```

Values of $2.0*10^{30}$ are assigned to responses of simulations with error terminations. The `AnalysisResults_n.csv` file is synchronous with the `Experiments_n.csv` file.

E.4.4 The DesignFunctions file

The DesignFunctions file, which appears in the sampling directory, is used to save a description of the polynomial design functions. It is an XML file with XML tags chosen such that the file is easy to read. Open a DesignFunction.* file in a text editor to understand the content of the database.

The order of the constants in the database for polynomial design functions is:

```
beta_0, beta_1, ... , beta_n, beta_1_1, beta_1_2, beta_1_3, ...,
beta_1_n,
                        beta_2_2, beta_2_3, ...., beta_2_n,
                        ...., beta_i_n,
                        beta_n_n
```

with

```
 f(x) = beta_0 + beta_1*x_1 + .... + beta_n*x_n +
           beta_1_1*x_1*x_1 + beta_1_2*x_1*x_2 + ... +
beta_1_n*x_1*x_n
           + beta_2_2*x_2*x_2 + ... + beta_2_n*x_2*x_n
           ...
           + beta_n_n*x_n*x_n
```

where $x \in [0,1]^n$.

The following enumerations are used in the database.

Function Types	
NO_SURFACE	0
LINEAR	77
MULT	78
QUADRATIC	79
INTERACTION	80
ELLIPTIC	81
SPHERICAL	82
FEEDFORWARD	83
FF_COMMITTEE	84
RADIALBASIS	85
NEURALNETWORK	86
ANALYTICAL_DSA_SURFACE	87
NUMERICAL_DSA_SURFACE	88
KRIGING	89
USERMETA	90
SVR	91

Response Interface Type		
RESP_INTERF_NULL	0	Interface unknown
USERINTERFACE	700	User defined
BINARY	701	LS-DYNA d3plot
ASCII	702	LS-DYNA ASCII files
REXPRESSION	703	Mathematical expression
XYFILE	704	User specified history file [*t*,*f(t)*]
LSDA_BINARY	705	
FREQUENCY	706	Frequency, Mode #, Generalized Mass
MASSC	707	Mass from d3hsp
D3P_DISP	708	Disp from d3 plot file
METAPOST	711	MetaPost post-processor format
NAST_FREQ	712	Nastran frequency
GENEX	713	Generic extractor
USERPOST	714	User-defined post-processor (uses same format as NAST_FREQ)
response_IMPORT	715	Imported from .csv file

Remarks

1. The flags for active coefficients exclude the constant a_0.
2. The coefficients are based on the independent variables x_i having been normalized to the size of the design space.

E.4.5 The VirtualHistoryFunction file

The *VirtualHistoryFunction* file appears in the main *work* directory and stores the approximation models for all histories at each sampled time-step. One file per iteration is generated. Like the *DesignFunctions* file, this is also a XML database with XML tags chosen such that the file is easy to read. This file stores the approximation type, number of fitting points, bounds and number of design variables, approximation model information (C or Wt), fitting and PRESS residuals (R and

PR) at the fitting points, and global error measures at each time-step (t) of the history curves. The enumerations for the type of the fitting function are the same as used for the *DesignFunction*s.

E.4.6 The OptimizationHistory file

This file is used to save the optimization history results and appears in the *work* directory. Each line contains the values at the optimum point of an iteration.

Entities	**Count**
Objective values	Number of objectives
Variables	Number of variables
Variable lower bounds	Number of variables
Variable upper bounds	Number of variables
RMS errors	Number of responses
Average errors	Number of responses
Maximum errors	Number of responses
R^2 errors	Number of responses
Adjusted R^2 errors	Number of responses
PRESS errors	Number of responses
Prediction R^2	Number of responses
Maximum prediction error	Number of responses
Responses	Number of responses
Multi-objective	1
Constraint values	Number of constraints
Composite values	Number of composites
Responses (computed)	Number of responses
Max. constraint violation	1
Composites (computed)	Number of composites

Constraints (computed)	Number of constraints
Objectives (computed)	Number of objectives
Multi-objective (computed)	1
Max. constraint violation (computed)	1
Constants	Number of constants
Dependents	Number of dependents
RBDO lower bound probability*	Number of constraints
RBDO upper bound probability*	Number of constraints
Generation number[#]	1
Individual number[#]	1

*Only written for RBDO problems.

[#]Only written for Direct GA simulations.

Values of $2.0*10^{30}$ are assigned to responses of error terminations.

E.4.7 The ExtendedResults file

This file contains all points represented in the `AnalysisResults_n.csv` file and appears in the sampling directory. All values are based on the simulation results. A line has the following format:

Entities	**Count**
Objective weights	Number of objectives
Objective values	Number of objectives
Variables	Number of solver variables
Responses	Number of solver responses
Multi-objective	1
Constraint values	Number of constraints
Composite values	Number of composites

Max. constraint violation	1
Constants	Number of constants
Dependents	Number of dependents

The values represent the number of entities in the solver. Values of 2.0*10^{30} are assigned to responses of simulations with error terminations.

E.4.8 The OptimumResults file

This file contains just the optimum design point data and appears in the main work directory. All values are metamodel values, i.e. interpolated.

Entities	**Count**
Objective weights	Number of objectives
Objective values	Number of objectives
Variables	Number of variables
Responses	Number of responses
Multi-objective	1 or 0 (no objectives)
Constraint values	Number of constraints
Composite values	Number of composites
Max. constraint violation	1
Constants	Number of constants
Dependents	Number of dependents

E.4.9 The *Sobol_GSA* file

This file contains the global sensitivity analysis database *Sobol_GSA.n* file and appears in the main *work* directory. One file per iteration is generated. For each response, the partial variance, main sensitivity index, total variance, and total Sobol sensitivity index due to different variables are stored. The mean and variance of the response is also stored. All quantities are based on metamodels. The analytical models are used to compute global sensitivity indices for polynomial approximations and Gaussian RBF functions, where as Monte-Carlo simulations are used for all other metamodels.

E.4.10 The lsopt_results file

This *lsda* binary database contains all the Tradeoff points. A database file *lsopt_results_[n].binout* is created in the main *work* directory for *n*th iteration. The database lists the following information for each TradeOff point.

Entities	**Location**	**Count**
Objective values	Inside directory	Number of objectives
Variables	Inside directory	Number of solver variables
Responses	Inside directory	Number of solver responses
Multi-objective		1
Constraint values	Inside directory	Number of constraints
Composite values	Inside directory	Number of composites
Max. constraint violation		1
Constants	Inside directory	Number of constants
Dependents	Inside directory	Number of dependents
Generation Index		1
Individual Index		1

E.4.11 The *lsopt_db* file

The file should not be used or edited by the user. It is used to communicate the state of the databases between various LS-OPT components. The content of the file is subject to change.

E.4.12 The lsopt_output and lsopt_report file

Taguchi results

When running a Taguchi analysis, the Taguchi ANOVA analysis table is available in *lsopt_output* and *lsopt_report*.

In this table, the following information is available:

- The level mean of each variable value: m_{V_i};

- The sum of square of each variable and interaction: $sumSq_V = \sum_{i=1}^{\#level(V)} \#exp_{V_i} \times (m_{V_i} - m)^2$ and $sumSq_I = \sum_{i=1}^{\#levA} \sum_{j=1}^{\#levB} \left[\# \exp_{A_i B_j} \times \left(m_{A_i B_j} - m\right)^2\right] - sumSq_A - sumSq_B$;
- The degree of freedom of each variable and interaction (see Section 23.2.7);
- The mean square of each variable and interaction: $sumSq/dof$;
- The F-ratio of each variable and interaction: $meanSq_{V\ or\ I}/meanSq_{err}$;
- The error sum of square: $sumSq_{tot} - \sum_{i=1}^{\#var} sumSq_{Var_i} - \sum_{i=1}^{\#inter} sumSq_{Inter_i}$;
- The error degree of freedom: $dof_{err} = dof_{tot} - \sum_{i=1}^{\#var} dof_{var_i} - \sum_{i=1}^{\#inter} dof_{inter_i}$;
- The error mean square: $meanSq_{err} = sumSq_{err}/dof_{err}$;
- The total sum of square: $sumSq_{tot} = \sum_{i=1}^{\#exp} (y_i - m)^2$;
- The total degree of freedom: $dof_{tot} = \#exp - 1$;
- The overall mean: $m = \frac{1}{\#exp} \sum_{i=1}^{\#exp} y_i$.

Where y_i represents the actual experimental result $res_i$for responses and composites, and the transformed logarithmic analysis result for objectives:

- $y_i = -10\ log_{10}((res_i + \left|\min_{j=1,n} res_j\right| + 1)^2)$ in case of a minimization problem;
- $y_i == -10\ log_{10}(1/(res_i + \left|\min_{j=1,n} res_j\right| + 1)^2)$ in case of a maximization problem.

The optimal results are considered in terms of S/N ratio which maximizes the mean over each level of their individual logarithmic analysis result: $\max_{lev}(\frac{1}{n}\sum y_i)$ with n the number of results obtained with this variable value.

Appendix F: Mathematical Expressions

F.1 Syntax rules

- Expressions consist of parameters and constants. A parameter can be any previously defined entity.
- Mathematical expressions can be used for any floating-point number, e.g. upper bound of constraint, convergence tolerance, objective weight, etc.
- An expression is limited to 1024 characters.
- Empty or underscore (_) arguments in functions will generate default values.
- Mathematical expressions can be defined in the input template files if the LS-OPT parameter format is used, e.g. <<Thickness*25.4>>.
- *Note:* Expressions with only integers will evaluate as an integer, e.g. 1 / 2 evaluates as 0. Both 1.0 / 2 and 1 / 2.0 evaluate as 0.5.

F.2 Intrinsic functions

Note: Trigonometric functions use and return degrees, not radians.

Function	**Description**
`int(a)`	integer
`nint(a)`	nearest integer
`abs(a)`	absolute value
`mod(a,b)`	remainder of *a/b*
`sign(a,b)`	transfer of sign from *b* to |*a*|
`max(a,b)`	maximum of *a* and *b*
`min(a,b)`	minimum of *a* and *b*
`sqrt(a)`	square root
`exp(a)`	e^a
`pow(a,b)`	a^b

`log(a)`	natural logarithm
`log10(a)`	base 10 logarithm
`sin(a)`	sine
`cos(a)`	cosine
`tan(a)`	tangent
`asin(a)`	arc sine
`acos(a)`	arc cosine
`atan(a)`	arc tangent
`atan2(a,b)`	arc tangent of *a/b*
`sinh(a)`	hyperbolic sine
`cosh(a)`	hyperbolic cosine
`tanh(a)`	hyperbolic tangent
`asinh(a)`	arc hyperbolic sine
`acosh(a)`	arc hyperbolic cosine
`atanh(a)`	arc hyperbolic tangent
`sec(a)`	Secant
`csc(a)`	cosecant
`ctn(a)`	cotangent
`cnd(a)`	cumulative normal distribution: $\Phi_{0,1}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{u^2}{2}\right) du$

Matrix functions (3×3 only):

Function	**Description**
`inv(A)`	Inverse of matrix ***A***

`tr(A)`	Transpose of matrix ***A***
`rx(angle)`	Rotation about *x*-axis (angle in rad)
`ry(angle)`	Rotation about *y*-axis (angle in rad)
`rz(angle)`	Rotation about *z*-axis (angle in rad)

F.3 Special functions

Special response functions can be specified to apply to response histories/multihistories. These include integration, minima and maxima and finding the time at a specific value of the function, Section F.3.1 . Some special functions such as cross-plotting and Mean Squared Error also apply to multihistories. Histories and multihistories can be mixed in the cross-plotting function while The Mean Squared Error, Discrete Fréchet, Dynamic Time Warping and Partial Curve Mapping *response*s apply to either histories or multihistories.

The arguments used in the expressions are explained in Section F.3.2

History and multihistory arguments must be defined as strings in double quotes and functions of time using the symbol `t`, e.g. `”Velocity(t)”`.

A “generic” argument type implies that the quantity can be an expression, another defined entity or a constant number. An entity (which may be specified in an expression) can be any defined LS-OPT entity. Thus `variable`, `history, multihistory, response` and `composite` are acceptable. If the argument of `TerminationTime` is a general expression containing several history definitions (e.g. `"Displacement1(t) - Displacement2(t)"`), the function will return the earliest termination time over all the histories included.

F.3.1 Functions to apply to response histories

Expression	**Symbols/Description**
`FilterHistory(history[,filtertype, frequency, timeunits, num_average])`	Filtered curve using SAE, Butterworth or running average
`TruncateHistory(history[, lower_limit, upper_limit, omit_point])`	Truncates history curve by removing the trailing end within [lower_limit,upper_limit]. The truncation loop is terminated at the first point outside the limits, starting from the end and then extrapolated up to the limit.

	Point omission can be flagged.
`DTWPHistory(target_curve,computed_curve[, num_reg_points)`	Dynamic Time Warping partial
`PCMHistory(target_curve,computed_curve[, num_reg_points)`	Curve Mapping partial
`RefineHistory(history, num_points)`	Remeshing of the curve
`DerivativeHistory(history, gridsize)`	n-Point rule using a finite difference formula with arbitrary grid spacing.
`Crossplot(history_z, history_F [, numpoints, begin_time, end_time])`	*F(z)* given *F(t)* and *z(t)*
`Integral(expression[,t_lower,t_upper,variable])`	$\int_a^b f(t)dg(t)$
`Derivative(expression[,T_constant])`	3-Point rule using a finite difference formula with arbitrary grid spacing.
`Min(expression[,t_lower,t_upper])`	$f_{\min} = \min_t [f(t)]$
`Max(expression[,t_lower,t_upper])`	$f_{\max} = \max_t [f(t)]$
`Initial(expression)`	First function value on record
`Final(expression)`	Last function value on record
`TerminationTime(expression)`	Termination time. Time of last history value.
`Lookup(expression,value[,t_lower,t_upper])`	Inverse function $t(f = F)$
`LookupMin(expression[,t_lower,t_upper])`	Inverse function $t(f = f_{\min})$
`LookupMax(expression[,t_lower,t_upper])`	Inverse function $t(f = f_{\max})$
`MeanSqErr(target_curve,computed_curve[, num_reg_points, start_point, end_point,`	Mean Squared Error function

weight_type, scale_type, weight_value, scale_value, weight_curve_name, scale_curve_name])	$\frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\boldsymbol{x}) - G_p}{s_p} \right)^2$
CurveMapSegment3(target_curve,computed_curve[,num_reg_points])	Curve Mapping discrepancy

F.3.2 Functions to apply to multipoint histories

Expression	**Symbols/Description**
FilterMultiHistory(multihistory[,filtertype, frequency, timeunits, num_average])	Filtered curve using SAE, Butterworth or running average
TruncateMultiHistory(multihistory[, lower_limit, upper_limit, omit_point])	Truncates multihistory curve by removing the trailing end within [lower_limit,upper_limit]. The truncation loop is terminated at the first point outside the limits, starting from the end and then extrapolated up to the limit. Point omission can be flagged.
DTWPMultiHistory(target_curve, computed_curve[,num_reg_points])	Dynamic Time Warping partials
PCMPMultiHistory(target_curve, computed_curve[,num_reg_points])	Curve Mapping partials

F.3.3 Arguments used in functions

Argument	**Explanation**	**Symbol**	**Type**
expression	history defined as an expression string	$f(t)$	generic
filtertype	Filtering type (SAE, Butterworth, running average)	-	integer

`frequency`	Filtering frequency (Hz)	f	float
`timeunits`	Time units (s or ms)	-	integer
`num_average`	Number of points averaged	-	integer
`gridsize`	Number of grid points in template	-	integer
`history_z, history_F`	History or multihistory names for abscissa and ordinate	$z(t)$, $F(t)$	history or multihistory
`numpoints`	Number of points in curve. Default: Smallest of the numbers of points defining *z* and *F*	-	integer
`begin_time, end_time`	Begin and end times Default: Largest t_0-value of *F* and *z* and smallest t_P-value of *F* and *z*, respectively	t_0, t_P	float
`t_lower`	lower limit of integration or range	a	generic
`t_upper`	upper limit of integration or range	b	generic
`variable`	integration variable	$g(t)$	generic
`T_constant`	specific time	T	generic
`value`	value for which lookup is required	F	generic
`target_curve, computed_curve`	Target, computed curve names	G, f	history or multihistory
`num_reg_points`	Number of regression points If $P < 2$ or not specified: use number of points in target curve between	P	integer

	start_point and *end_point*.		
start_point, end_point	Location of first/last regression points	z_0, z_P	float
weight_type	Weight type (F.3.4): WEIGHTVALUE (default) PROPWEIGHT WEIGHTCURVE		reserved
scale_type	Scale type (F.3.4): SCALEVALUE (default) PROPSCALE MAXISCALE FILESCALE		reserved
weight_value	Uniform weight value (default = 1)	*W*	float
scale_value	Uniform scale value (default = 1)	*s*	float
weight_curve	Weights as a function of *z* (default Weight.*compositename*)	*W(z)*	History
scale_curve	Scale factors as a function of *z* (default Scale.*compositename*)	*s(z)*	History
num_points	Number of points of results curve	-	integer

Remarks

1. Omitting the lower and upper bounds implies operation over the entire available history.
2. The Lookup function allows finding the value of t for a specified value of $f(t) = F$. If such a value cannot be found, the largest value (within the specified bounds) of t in the

history is returned. The LookupMin and LookupMax functions return the value of *t* at the minimum or maximum respectively.

3. *Integration.* The implied variable represented in the first column of any history file is *t*. All history files produced by extraction from the LS-DYNA database are functions of *t* which increases monotonically. The fourth argument of the Integral function defaults to the abscissa.

   There are two ways to compute an integral over a crossplot $f_g\big(g(t)\big)$ of $f(t)$ and $g(t)$:

   - ○ $\int_{t_a}^{t_b} f(t)dg(t)$: Integral("f(t)", t_a, t_b, "g(t)")
   - ○ $\int_{g_b}^{g_a} f(g)dg$: Integral("f_vs_g(t)", g_a, g_b)

   The first formula uses the component functions of a crossplot instead of the crossplot itself. It is more robust since it also allows for the function $g(t)$ to be non-monotonic in t. The values t_a and t_b are *time* values. i.e. the abscissa of the component functions. The two history components $f(t)$ and $g(t)$ must be available.

   The second formula can be used for integration under an arbitrary curve as long as the abscissa is monotonic. The values g_a and g_b are on the g-axis. The functions $f(t)$ and $g(t)$ do not have to be available.

4. *Differentiation.* Finite difference formulas for arbitrarily spaced grids are used for numerical differentiation [1]. For the derivative history an *n*-point rule can be specified. It is recommended to keep n small, but the algorithm is numerically robust and $n = 20$ has been used successfully. The 3- and 5-point rules are routinely applied in industry and hence recommended. The 3-point rule is the default.

   The derivative curve inherits the points of the input curve, so the number of points of the reference curve should not be too small.

   For calculating a scalar derivative value at a specific value of the abscissa, the 3-point rule is used. *T_constant* in the Derivative function defaults to the last abscissa value.

   A future version of derivative functions should allow for higher derivatives since the core algorithm has no inherent limit to the order of differentiation or grid size.

5. If a time is specified smaller than the smallest time value of the computed history, the first value is returned (same as Initial). If a time is specified larger than the largest time value of the computed history, the last value is returned (same as Final).

6. The Crossplot function applies to both histories and multihistories.

F.3.4 Options for **MeanSqErr** arguments

Syntax	Explanation
WEIGHTVALUE	W_i = *value.* Default = 1.0

`PROPWEIGHT`	Use a different weight for each curve point *p*, proportional to the value of $\|G_p\|$. This method emphasizes the large absolute values of the response. The weights are normalized with respect to max $\|G_p\|$
`FILEWEIGHT`	Interpolate the weight from an x-y file: weight vs. *z*
`SCALEVALUE`	$s_i = value$. Default = 1.0
`MAXISCALE`	max $\|G_p\|$.
`PROPSCALE`	Use a different scale factor for each curve point, namely $\|G_p\|$.
`FILESCALE`	Interpolate the scale factor from an x-y file: scale vs. *z*

Remarks on MeanSqErr

1. Only points within range of both curves are included, so P will be automatically reduced during the evaluation if there are missing points. A warning is issued in `WARNING_MESSAGE.`
2. The weight curve and scale curve must be predefined histories (see Section **6.1**) if they are selected. If a weight or scale curve is selected, the name of the curve defaults to ‘`Weight.compositename`’ or ‘`Scale.compositename`’ respectively where compositename is the name of the parent composite being defined.
3. The `MeanSqErr` composite makes use of response surfaces to avoid the nonlinearity (quadratic nature) of the squared error functional. Thus if the response curve f(**x**) is linear in terms of the design variables **x**, the composite function will be exactly represented.
4. The `MeanSqErr response` is also available for both histories and multihistories.
5. Empty or underscore (_) arguments will generate default values.
6. The option names in Section F.3.4 are reserved names and cannot be used as variable names.
7. `MeanSqErr` composites can be added together to make a larger MSE composite (e.g. for multiple test cases).
8. The simplest target curve that can be defined has only one point.

F.4 Matrix functions

Expression	**Symbols/Description**
`Rotate(x1,y1,z1, x2,y2,z2, x3,y3,z3)`	Rotation matrix defined by 3 points, Section 6.4.5.

`Matrix3x3Init(x1,y1,z1, x2,y2,z2, x3,y3,z3)`	Initialize 3×3 matrix		
Argument	**Explanation**	**Symbol**	**Type**
`x1,y1,z1,x2,y2,z2,x3,y3,z3`	Matrix components	-	generic

F.5 Reserved variable names

Name	Explanation
`t`	Time
`pt`	Point; argument for multiresponses used in multiresponse expressions
`LowerLimit`	0.0
`UpperLimit`	Maximum event time over all histories of all solvers
`length`	Intrinsic variable

F.6 Constants associated with histories

The following commands can be given to override defaults for history operations:

Constant	Explanation	Default
`variable fdstepsize`	Finite difference step size for numerical derivatives with respect to variables	0.0001*(Upper bound – Lower bound)
`Historysize (attribute of globals node)`	Number of time points for new history	10000

Remarks

1. The `variable fdstepsize` is used to find the gradients of expression composite functions. These are used in the optimization process.
2. The `historysize` is used when new histories are generated.

F.7 Generic expressions

Expressions can be specified for any floating-point number. In some cases, previously defined parameters can be used as follows:

Number type	Parameter type
Constant	none
Starting variable	constant
Range	variable
Variable bounds	variable
Shift factor for response	variable
Scale factor for response	variable
Constraint bounds	variable
Objective weight	variable
Target value (composite)	variable
Scale factor (composite)	variable
Weight (composite)	variable
Parameters of SRSM	none
Parameters of LFOPC	none

The parameter type represents the highest entity in the hierarchy. Thus constants are included in the variable parameters.

In LS-OPT, expressions can be entered for variables, histories, responses, and composites.

F.8 Examples illustrating syntax of expressions

```
lowerlimit * 1000
upperlimit * 1000
(his1(t) + his2(t))/2
Initial("his1(t)")
Final("his1(t)")
Final("(his1(t) + his2(t))/2")
Max("his1(t)")
Max("his1(t)","ll * 1.0")
Max("his1(t)","ll",ul)
Max("his3(t)",ll,ul)
Min("his3(t)",ll,ul)
Lookup("his1(t)",75)
Lookup("his2(t)",75)
Lookup("his3(t)",75)
Lookup("(his1(t) + his2(t))/2",75)
max(Inverse11,Inverse21)
min(Inverse11,Inverse21)
his3(Inverse31)
his3(66.1) + 0.1
nint(hist66)
int(hist66)
Integral("his1(t)")
Integral("his1(t)",ll,ul,"t")
Integral("his1(t)",ll,UPPER,"t")
Integral("his2(t)",ll,ul,"t")
Integral("his3(t)",ll,ul,"t")
Integral("(his1(t) + his2(t))/2",ll,ul,"t")
Integral("his3(t)")
Integral("his3(t)",ll)
Integral("his3(t)",ll,ul)
Integral("his1(t)",ll,ul,"his2(t)")
Integral("his1(t)",0,30,"his2(t)")
Integral("his1(t)",30,100,"his2(t)")
Integ1 + Integ2
Integral("sin(t) * his1(t) * his2(t)",ll,ul,"t")
Integral("sin(t) * his1(t) * his2(t)")
Derivative(”Displacement(t)”,0.08)
Derivative(”Displacement(t)”)
(Integ3a/(4*Maximum11) + Integ2/2)**.5
(Apillar_velocity_1 +  Apillar_velocity_2)/2}
Lookup("engine_velocity(t)",0)
Apillar_velocity_average(time_to_engine_zero)
Integral("Apillar_velocity_average(t)",0,
time_to_engine_zero)/time_to_engine_zero
Lookup("global_velocity(t)",0)
Apillar_velocity_average(time_to_zero_velocity)
Integral("Apillar_velocity_average(t)",time_to_engine_zero,
     time_to_zero_velocity)/(time_to_zero_velocity -
time_to_engine_zero)
```

F.9 REFERENCES

[1] Fornberg, B. Calculation of weights in finite difference formulas. SIAM Rev., Vol. 40, No. 3, 685-691, 1998

Appendix G: Injury Criteria

G.1 Syntax of Injury Criterion Commands

Expression	Symbols/Description	Reference
`MOC (force, moment [, dummy_type, length_units, force_units, distance])`	$MOC = M - (D \cdot F)$	G.1.1
`NIC (accel_t, accel_h [, time_units, length_units] )`	$NIC = a_{relative} \cdot 0.2 + v_{relative}^2$	G.1.3
`Nij (force,moment,shear [,dummy_type, length_units, force_units, c_force_tension, c_force_compression, c_moment_flexion, c_moment_extension, distance])`	$NIJ = \frac{F}{F_c} + \frac{MOC}{M_c}$	G.1.4
`Nkm (force, moment [, dummy_type, length_units, force_units, criteria_type, distance, critical_force, critical_moment])`	$Nkm(t) = \frac{F(t)}{F_{\text{int}}} + \frac{MOC(t)}{M_{\text{int}}}$	G.1.6
`LNL (axial_force, s_shear, t_shear, s_moment, torsion [, length_units, force_units, critical_moment, critical_shear, critical_tension])`	$LNL = \frac{\sqrt{M_y^2 + M_x^2}}{C_{moment}} + \frac{\sqrt{F_y^2 + F_x^2}}{C_{shear}} + \left\lvert \frac{F_z + off}{C_{tension}} \right\rvert$	G.1.9
`ChestCompression ( relative_rotation, dummy_type, user_constant )`	$C_1 \max_t [\Theta(t)]$	G.1.10
`ViscousCriterion ( history_name, [dummy_type, time_units, length_units, scaling_factor, deformation_constant, user_constant] )`	$-\min \frac{C_1}{C_2} C_3 Y(t) C_3 \frac{dY(t)}{dt}$	G.1.12

`TTI (accel_upper_rib, accel_lower_rib, accel_lower_spine, time_units, length_units, gravity)`	$TTI = \frac{A(\max.rib) + A(lwr.spine)}{2}$	G.1.14
`TibiaIndex (torsion, s_moment, t_shear, dummy_type, length_units, force_units, critical_moment, critical_force)`	$TI = \left\| \frac{M}{M_C} \right\| + \left\| \frac{F}{F_C} \right\|$	G.1.15
`A3ms ( accel_x, [ accel_y, accel_z, time_units, length_units, time_interval, gravity ] )`	-	G.1.17

G.1.1 Options for MOC arguments

Argument name	Description	Symbol	LS-OPT type	Default
`Force`	Neck axial force resultant	`F`	History	
`moment`	Neck s-moment resultant	`M`	History	
`dummy_type`	Dummy type	-	G.1.2	`HYBRID3M50`
`length_units`	Length units	-	`LENG_M` `LENG_MM`	`LENG_MM`
`force_units`	Force units	-	`FORCE_N` `FORCE_KN`	`FORCE_N`
`Distance`	Distance	`D`	Float	G.1.2

G.1.2 MOC Input constants for various dummy types

Dummy Type	D[m]
`Hybrid III, male 95%`	0.01778
`Hybrid III, male 50%`	0.01778
`Hybrid III, female 5%`	0.01778

Hybrid III, 10-year	0.01778
Hybrid III, 6-year	0.01778
Hybrid III, 3-year	0
Crabi 12, 18 month	0.00584
TNO P1,5	0.0247
Crabi 6 month	0.0102
TNO P 3/4, P3	0
ES-2	0
TNO Q series	0
SID-IIs	0.01778
BioRID	0.01778
WORLDSID	0.0195

G.1.3 Options for NIC arguments

Argument name	Description	Symbol	LS-OPT type	Default
accel_t	x-acceleration of first thorax spine	a_x^{T1}	History	
accel_h	x-acceleration at the height of the c.o.g. of the head	a_x^{Head}	History	
time_units	Time units	-	TIME_S TIME_MS	TIME_S
length_units	Length units	-	LENG_M LENG_MM	LENG_MM

G.1.4 Options for Nij arguments

Argument name	Description	Symbol	LS-OPT type	Default
`force`	Neck axial force resultant	See MOC	History	
`moment`	Neck s-moment resultant	See MOC	History	
`shear`	Force at the point of transition from head to neck	`F`	History	
`dummy_type`	Dummy type	-	Table 6-19	`HYBRID3M50`

`length_units`	Length units	-	`LENG_M,`
			`LENG_MM`
				`LENG_MM`

`force_units`	Force units	-	`FORCE_N,`
			`FORCE_KN`
				`FORCE_N`

`c_force_tension`	Critical force tension	F_c	Float	G.1.5
`c_force_compression`	Critical force compression	F_c	Float	G.1.5
`c_moment_flexion`	Critical moment flexion	M_c	Float	G.1.5
`c_moment_extension`	Critical moment extension	M_c	Float	G.1.5
`distance`	Distance	`D` (See MOC)	Float	G.1.5

G.1.5 Nij Input constants for various dummy types

Dummy type	Description	Test	F_C [N] Tension	F_C [N] Compression	M_C [Nm] Flexion	M_C [Nm] Extension
HYBRID3M50	Hybrid III; male 50%	In position	6806	-6160	310	-135
HYBRID3F05	Hybrid III; female 5%	In position	4287	-3880	155	-67
HYBRID3F05	Hybrid III; female 5%	Out of position	3880	-3880	155	-61
HYBRID310Y	Hybrid III; 6-year	Out of position	2800	-2800	93	-37
HYBRID306Y	Hybrid III; 3-year	Out of position	2120	-2120	68	-27
HYBRID303Y	Hybrid III; 12 month	Out of position	1460	-1460	43	-17

G.1.6 Options for Nkm arguments

Argument name	Description	Symbol	LS-OPT type	Default
force	Neck axial force resultant	F	History	
moment	Neck s-moment resultant	See MOC	History	
dummy_type	Dummy type	-	Table 6-15 (MOC)	HYBRID3M50
length_units	Length units	-	LENG_M, LENG_MM	LENG_MM
force_units	Force units	-	FORCE_N, FORCE_KN	FORCE_N

criteria_type	Nfa, Nea, Nfp, Nep	-	FLEXION_ANTERIOR EXTENSION_ANTERIOR FLEXION_POSTERIOR EXTENSION_POSTERIOR	FLEXION_ANTERIOR
distance	Distance	D (See MOC)	Float	Table 6-15 (MOC)
critical_force	Critical force	F_{int}	Float	G.1.7
critical_moment	Critical moment	M_{int}	Float	G.1.7

G.1.7 Nkm Input constants

Criteria	**Description**	**Value**
*_ANTERIOR	Positive Shear F_{int}	845 N
*_POSTERIOR	Negative Shear F_{int}	-845 N
FLEXION_*	Flexion M_{int}	88.1 Nm
EXTENSION_*	Extension M_{int}	-47.5 Nm

G.1.8 Options for LNL arguments

Argument name	**Description**	**Symbol**	**LS-OPT type**	**Default**
axial_force	Axial force resultant	F_y	History	
s_shear	s-Shear resultant	F_x	History	
t_shear	t-Shear resultant	F_z	History	

s_moment	s-Moment resultant	M_y	History	
torsion	Torsional resultant	M_x	History	
length_units	Length units	-	LENG_M, LENG_MM	LENG_MM
force_units	Force units	-	FORCE_N, FORCE_KN	FORCE_N
critical_moment	Critical moment	C_{moment}	Float	G.1.9
critical_shear	Critical force	C_{shear}	Float	G.1.9
critical_tension	Critical force	$C_{tension}$	Float	G.1.9

G.1.9 LNL Input constants

Force/Moment	Description	Value
C_{moment}	Critical moment	15 [Nm]
C_{shear}	Critical force	250 [N]
$C_{tension}$	Critical force	900 [N]

G.1.10 Options for Chest Compression arguments

Argument name	Description	Symbol	LS-OPT type	Default
relative_rotation	relative rotation history	$\Theta(t)$	History	
dummy_type	dummy type	-	HYBRID3M95 HYBRID3M50 HYBRID3F05	HYBRID3M50
constant	Multiplier	C_1	Float	Table 6-25

G.1.11 Chest Comporession Input constants for various dummy types

Dummy type	Description	Scaling factor C_1
`HYBRID3M95`	Hybrid III; male 95%	130.67
`HYBRID3M50`	Hybrid III; male 50%	-139.0
`HYBRID3F05`	Hybrid III; female 5%	-87.58

G.1.12 Options for Viscous Criterion arguments

Argument name	Description	Symbol	LS-OPT type	Default
`history_name`	Thoracic deformation (m)	$Y(t)$	History	
`dummy_type`	Dummy type	-	Table 6-27	`HYBRID3M50`
`time_units`	Time units	-	`TIME_S` `TIME_MS`	`TIME_S (seconds)`

`length_units`	Length units	-	`LENG_M`
			`LENG_MM`
				`LENG_MM`
				`(mm)`

`scaling_factor`	Scaling factor (multiplier)	C_1	Float	G.1.13
`deformation_constant`	Constant: Depth or width of half the rib cage (m)	C_2	Float	G.1.13
`user_constant`	Multiplier of thoracic deformation	C_3	Float	1.0

G.1.13 Viscous Criterion Input constants for various dummy types

Dummy type	Description	Scaling factor	Deformation constant (m)
HYBRID3M95	Hybrid III; male 95%	1.3	0.254
HYBRID3M50	Hybrid III; male 50%	1.3	0.229
HYBRID3F05	Hybrid III; female 5%	1.3	0.187
BIORID2	BioSID	1.0	0.175
EUROSID1	EuroSID-1	1.0	0.140
EUROSID2	EuroSID-2	1.0	0.140
SID2S	SID-IIs	1.0	0.138

G.1.14 Options for TTI arguments

Argument name	Description	Symbol	LS-OPT type	Default
accel_upper_rib	y-acceleration of the upper rib	$A(upr.rib)$	History	
accel_lower_rib	y-acceleration of the lower rib	$A(lwr.rib)$	History	
accel_lower_spine	y-acceleration of the lower spine	$A(lwr.spine)$	History	
time_units	Time units	-	TIME_S TIME_MS	TIME_S
length_units	Length units	-	LENG_M, LENG_MM	LENG_MM

gravity	Gravitational acceleration	g	Float	Depends on time_units and length_units. 9810 mm/s^2 if units undefined

G.1.15 Options for TI arguments

Argument name	Description	Symbol	LS-OPT type	Default
Torsion	Bending moment, torsional resultant	M_x	History	
s_moment	Bending moment, s-moment resultant	M_y	History	
t_shear	Axial compression, t-shear resultant	F	History	
dummy_type	Dummy type	-	Table 6-30	HYBRID3M50
length_units	Length units	-	LENG_M, LENG_MM	LENG_MM
force_units	Force units	-	FORCE_N, FORCE_KN	FORCE_N
critical_moment	Critical bending moment	M_C	Float	G.1.16
critical_force	Critical compression force	F_C	Float	G.1.16

G.1.16 TI Input constants for various dummy types

Dummy type	Description	Critical bending moment [Nm]	Critical compression force [kN]
HYBRID3M95	Hybrid III, male 95%	307.0	44.2

`HYBRID3M50`	Hybrid III, male 50%	225.0	35.9
`HYBRID3F05`	Hybrid III, female 5%	115.0	22.9

G.1.17 Options for a3ms arguments

Argument name	**Description**	**Symbol**	**LS-OPT type**	**Default**
`accel_x`	*x*-acceleration history	$\ddot{x}$	History	
`accel_y`	*y*-acceleration history	$\ddot{y}$	History	no history
`accel_z`	*z*-acceleration history	$\ddot{z}$	History	no history
`time_units`	Time units	-	`TIME_S` `TIME_MS`	`TIME_S (seconds)`

`length_units`	Length units	-	`LENG_M`
			`LENG_MM`
				`LENG_MM`
				`(mm)`

`time_interval`	Time interval for which the acceleration is exceeded	Δt	Float	0.003 s
`gravity`	Gravitational acceleration	g	Float	Depends on `time_units` and `length_units`. 9810 mm/s^2 if units undefined

Appendix H: Installing LS-OPT

This chapter describes the installation of LS-OPT. The description includes remote job scheduling through a queuing system or a proxy system.

H.1 Download

LS-OPT is available at http://lstc.com/download/ls-opt. Please ask LST or your local distributor for login and password. There are several Linux and Windows versions available. Examples are also provided.

H.2 Installation

H.2.1 Linux

Create or select a folder where you would like to install LS-OPT. Unpack the archive using

```
tar -xvfz lsopt_version_revision_system.tar.gz
```

in that directory. When unzipping the file, a directory *LSOPT_EXE* will be created. *LSOPT_EXE* contains all the binaries required. *lsoptui* is the binary to start the graphical user interface of LS-OPT. Create an alias for *lsoptui* or make sure that *lsoptui* is in the path. If you want to use LS-OPT in batch mode, do the same for the binary *lsopt*.

H.2.2 Windows

The recommended way of installing LS-OPT is to use the installer. Simply download the installer and execute it and go through the wizard. It will ask for administration rights on execution. The installer will create necessary shortcuts and associate lsopt-files to open with LS-OPT GUI.

An alternative way of installing LS-OPT is to download a zip-file and extract the contents. This might be preferable in-case you do not have administration rights on the computer or do not want to overwrite a current installation of LS-OPT. When unzipping the file, a directory *LSOPT_EXE* will be created. *LSOPT_EXE* contains all the binaries required. To start LS-OPT, locate the ***lsoptui*** program and double-click it. You may wish to create a short-cut on your desktop rather than remember where the LS-OPT installation folder is located. You might also want to associate lsopt-files to open with *lsoptui.exe*.

In case you want to use the license server for LS-DYNA, please make sure that the LSTC License Client *lstc_client.exe* is available in the directory where the LS-DYNA executable is located.

H.2.3 How to run LS-DYNA from LS-OPT using the license server (Windows)

In case you want to use the license server for LS-DYNA, you need to do the following:

1. Go to the "start" menu of the Windows Operating System and follow the steps:
2. Right click on "My Computer"
   - Choose "Properties"
   - Click "Advanced" tab
   - Click "Environment Variables" button
   - Add the following "User variables":

```
LSTC_LICENSE                    network

LSTC_LICENSE_SERVER        <name of the license server host
machine>
```

The first column above has the variable names and the second column, the variable values, to be filled into the boxes.

You can also start by right-clicking on the "My Computer" icon on your desktop and going through the steps as explained above.

It may be necessary to restart the operating system to initialize the environment variables.

H.3 Remote job scheduling

The solver jobs do not have to be executed on the same machine as where LS-OPT is running. There are several ways of distributing the solver jobs. An example of remote job distribution is when the user is running LS-OPT on a laptop or desktop computer but prefers to run multiple solver jobs in parallel on a computer cluster.

There are five common scenarios that we try to address using various LS-OPT job scheduling options.

1. runqueuer/wrapper option
2. You have a queuing system and you want to submit some or all LS-OPT solver jobs to that queuing system.
3. You can allow remote solver jobs to initiate TCP/IP connections back to the machine where LS-OPT runs.
4. Blackbox or Simple queuer option
5. You have a queuing system and you want to submit some or all LS-OPT solver jobs to that queuing system.
6. You prefer not to allow remote solver jobs to initiate TCP/IP connections back to the machine where LS-OPT runs.

7. SSH option
8. You do not have a queuing system. You would like to run LS-OPT on one machine, but you would like to run all solver jobs on a single, dedicated Linux cluster.
9. Queuing system via SSH proxy
10. Similar to option 1) or 2) but either:
11. Communication with the queuing system is only available from a remote Linux host.
12. There is a firewall blocking runqueuer/wrapper communications
13. Running via LSTCVM proxy.
14. An alternative proxy system to SSH. Although SSH is recommended, this is still supported for existing solutions and for anywhere SSH is not an option.
15. Requires installation of LSTCVM server on a remote system.

H.4 Simple manual setup for running LS-OPT and solvers on different machines

A simple manual setup is one in which LS-OPT runs on e.g. a Windows machine and the solvers are running on a cluster (typically Linux). Such a setup can be created as follows:

1. Install LS-OPT on a Windows (or any desired) machine for preparing the input. Create the problem setup using LS-OPTui. The `solver command` should be created for running jobs on a cluster. This can be done by selecting any of the queuing systems supported by LS-OPT or, if all the jobs will be running on the same cluster where LS-OPT resides, simply by specifying the solver executable name as a *solver command*. The *Resources* should be set for each stage in the **Stage** dialog. Save the input to a file using e.g. the name `project.lsopt.`
2. Install LS-OPT and the solver executables on a cluster node for running LS-OPT in batch mode. Copy the recently created problem setup files from the Windows machine onto a cluster node. This setup should allow the user to run an LS-OPT job in batch mode.
3. Run LS-OPT by executing the command: `lsopt project.lsopt` on the cluster. This is done from the command line.
4. After completion of the LS-OPT run, execute `lsopt -a -H project.lsopt` to create a file `lsopack_h.tar.gz` containing the entire run database. See Section 2.2.1 for a description of the command line options.
5. Unzip `lsopack_h.tar.gz` on the Windows machine to do the post-processing.

H.5 Using an external queuing or job scheduling system

H.5.1 Installation

The LS-OPT Queuing Interface interfaces with queuing systems (e.g. LSF[22], LoadLeveler[23] and others) to enable running simulation jobs across a network. LS-OPT will automatically copy the simulation input files to each remote node, extract the results on the remote directory and transfer the extracted results to the local directory.

To run LS-OPT with a queuing system the following executables files are provided in with the LS-OPT distribution package or installer:

- `wrapper`
- `runqueuer`

The runqueuer executes the command line for the purpose of queuing and must remain in the LS-OPT environment (the same directory as the lsopt executable).

The following instructions should then be followed:

H.5.2 Installation for all remote machines running simulation jobs

1. Create a directory on the remote machine for keeping all the executables. Copy the appropriate executable `wrapper` program to the new directory. e.g. if you are running simulation jobs on a Linux machine, place the `wrapper` appropriate for the architecture and operating system on this machine.
2. Change the script you use to run the solver via the queuing facility by prepending "wrapper" to the solver execution command. Use full path names for both the wrapper and executable or make sure the path on the remote machine includes the directory where the executables are kept.

The argument for the input deck specified in the script must always be the LS-OPT reserved name for the chosen solver, e.g. for LS-DYNA use `DynaOpt.inp`.

H.5.3 Environment variables LSOPT_HOST and LSOPT_PORT

Users typically do not set these. These variables are set on the local side by the `runqueuer` program and their values must be carried to the remote side by the queuing software.

`LSOPT_HOST` : the machine where LS-OPT (and therefore the `runqueuer`) is running. Set this if the wrapper has trouble connecting back to `runqueuer`.

`LSOPT_PORT` : TCP/IP port `runqueuer` listens on for remote connections

The `runqueuer` program does not set LSOPT_HOST if it is already set, but *always* sets LSOPT_PORT. The examples in Section H.5.4 illustrate two methods by which setting of environment variables can be accomplished. Environment variables specified by *Environment*

[22] Registered Trademark of Platform Computing Inc.
[23] Registered Trademark of International Business Machines Corporation

Variables settings in the LS-OPT GUI (see Section H.12) are set by the scheduler. The scheduler runs runqueuer, and runqueuer would be the one to run

```
      qsub -v LSOPT_PORT,LSOPT_HOST <script_name>
```

So, the LSOPT_PORT value passed to the remote side will always be the one set by runqueuer. However, the LSOPT_HOST value may be set through *Environment Variables* or though "`.cshrc`" instead.

In most cases the queuing system will transmit the environment variables to the remote side, so the setting of the variables may not be necessary. The only reason to set LSOPT_HOST would be to compensate for a wrong setting. For example, the machine where LS-OPT is running may be known by several different host names or by different IP addresses. In such a case it might be required to specify which interface should be used for remote connections. It is not permissible for LSOPT_PORT to be changed because *only* the runqueuer knows the right setting.

Notes:

- The LSOPT_PORT number is displayed at the top of the `job_log` file residing in the *job* home directory which can also be viewed by selecting the job progress bar followed by "View log" in the progress monitor.
- The LSOPT_PORT number will be different for each job. Even if the same job is repeated, the LSOPT_PORT number will be different each time.

H.5.4 Examples

Example 1:

This example uses a two-level script. The outer script (submit_pbs), which has to be used as command for the respective stage, sets the values of environment variables in dynscr (the inner script), spawns it and submits it through the queuing system. The script dynscr then sets the environment variables and schedules the solver on the remote machine.

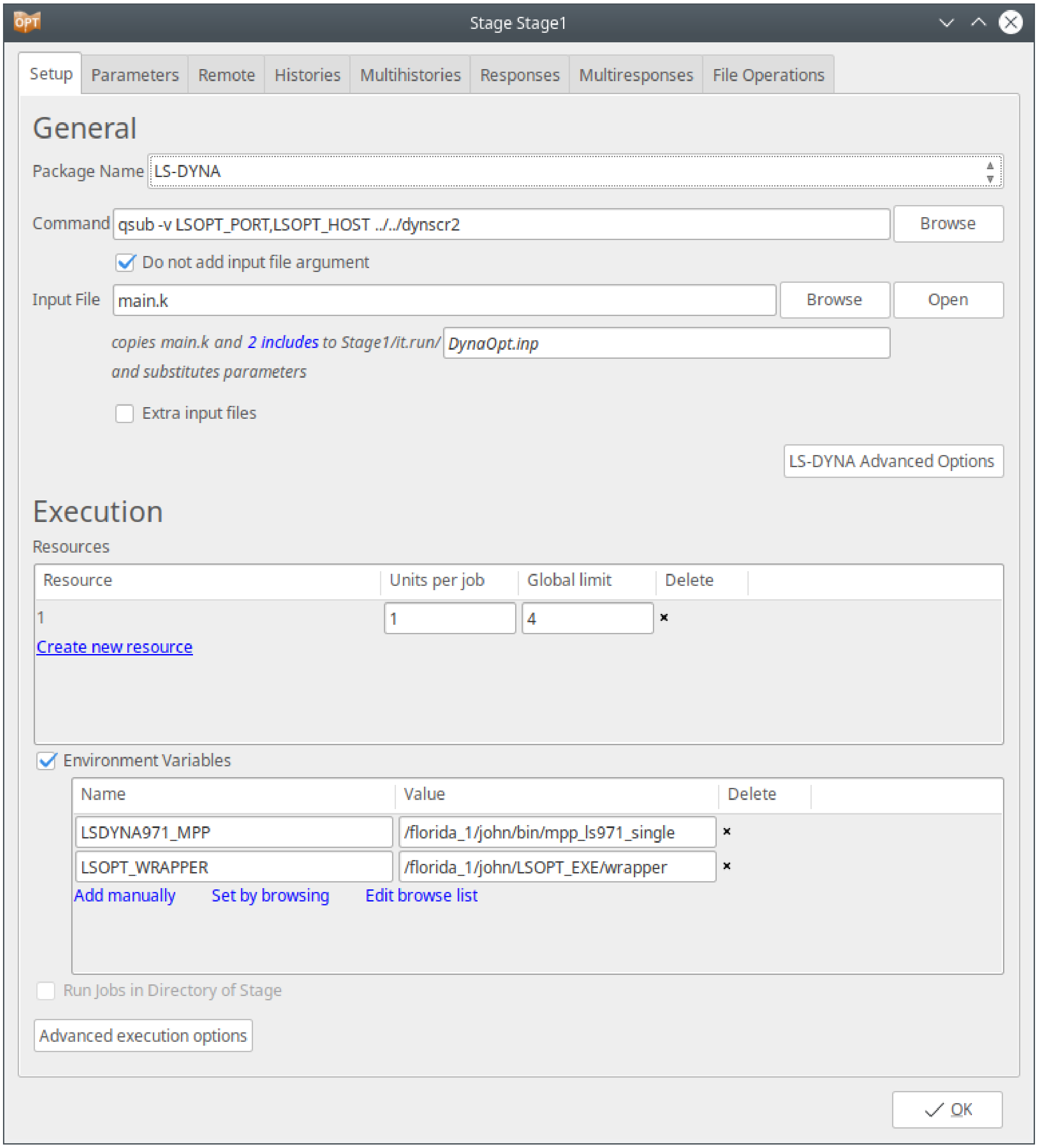
The "submit_pbs" file is:

```
#!/bin/csh -f
#
# Run jobs on a remote processor, remote disk
set newdir=`pwd | sed -n 's/.*\/\(.*\)\/\(.*\)/\1\/\2/p'`
# Run jobs on a remote processor, local disk (no transmission)
# set newdir=`pwd`
echo $newdir
cat > dynscr << EOF
#
# dynscr script
# ============================================================
#!/bin/csh -f
#
#PBS -l nodes=1:ncpus=1
#
setenv LSOPT /nec00a/mike/codes/LSOPT_EXE
setenv LSOPT_HOST $LSOPT_HOST
setenv LSOPT_PORT $LSOPT_PORT
# Run jobs on a remote processor, remote disk
mkdir -p lsopt/$newdir
cd lsopt/$newdir
# The input file name is required for LS-OPT
/nec00a/mike/codes/wrapper /nec00a/mike/codes/ls980.single
i=DynaOpt.inp
EOF
# ============== E N D   O F   S C R I P T =======================
qsub dynscr
```

Example 2:

This example demonstrates how to specify the queuer command directly on the command line. It shows how the required environment variables LSOPT_PORT and LSOPT_HOST set by the runqueuer program are specified on the stage command line whereas the two user variables LSDYNA971_MPP and LSOPT_WRAPPER are defined as environment variables in LS-OPT, see Figure below. These can also be set on the command line using the Linux "setenv" command as specified in for instance the .cshrc script. qsub is a PBS queue submit command and the –v directive defines the names of environment variables to be exported to the job.

The qsub manual pages should be consulted for more details. Please also consult Sections H.5.3 (Environment Variables) and H.12 (Passing Environment Variables through LS-OPT).



General

Package Name: LS-DYNA

Command: qsub -v LSOPT_PORT,LSOPT_HOST ../../dynscr2

☑ Do not add input file argument

Input File: main.k

copies main.k and 2 includes to Stage1/it.run/ DynaOpt.inp
and substitutes parameters

☐ Extra input files

Execution

Resources

Resource	Units per job	Global limit	Delete
1	1	4	×

☑ Environment Variables

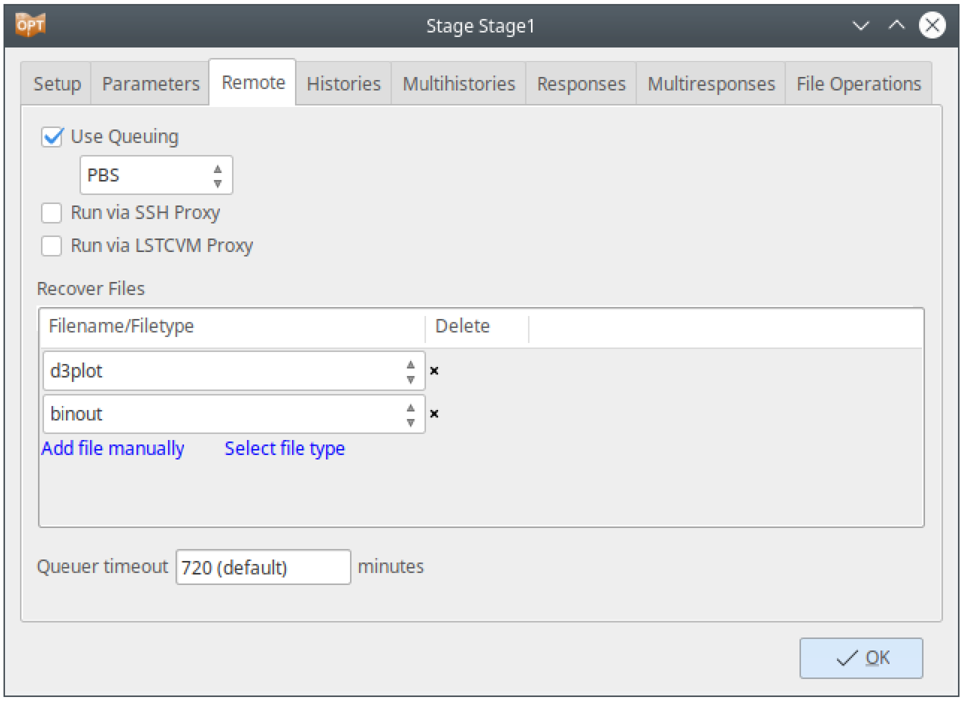
Name	Value	Delete
LSDYNA971_MPP	/florida_1/john/bin/mpp_ls971_single	×
LSOPT_WRAPPER	/florida_1/john/LSOPT_EXE/wrapper	×

☐ Run Jobs in Directory of Stage

Figure H - 1: Definition of queuer command in Stage Command and Environment Variables. The script dynsrc2 is displayed below.

```
# This is the dynscr2 file
#==========================
#!/bin/csh -f
#
#$ -cwd -pe mpi 2
#
setenv NP 2
setenv ROUNDROBIN 0
```

```
#
# Define LSDYNA971_MPP environment variables in lsopt input
# or shell command ("setenv").
# $1 represents i=DynaOpt.inp and is automatically
# tagged on as the last argument of the lsopt "solver command".
#
setenv EXE "$LSDYNA971_MPP $1"
#
rm -f mpd.hostfile mpp.appfile
filter_hostfile < $PE_HOSTFILE > mpd.hostfile
#
# This python script builds an HPMPI specific "appfile" telling it
# exactly what to run on each node.
#
gen_appfile.hpmpi mpd.hostfile $SGE_O_WORKDIR $NP $ROUNDROBIN $EXE >
mpp.appfile
#
# This actually executes the job
#
$LSOPT_WRAPPER /opt/hpmpi/bin/mpirun -f mpp.appfile
#
```



☑ Use Queuing

PBS

□ Run via SSH Proxy

□ Run via LSTCVM Proxy

Recover Files

Filename/Filetype	Delete
d3plot	×
binout	×

Queuer timeout 720 (default) minutes

Figure H - 2: Definition of remote queuing system and recovery of LS-DYNA output databases

H.6 Mechanics of the queuing process

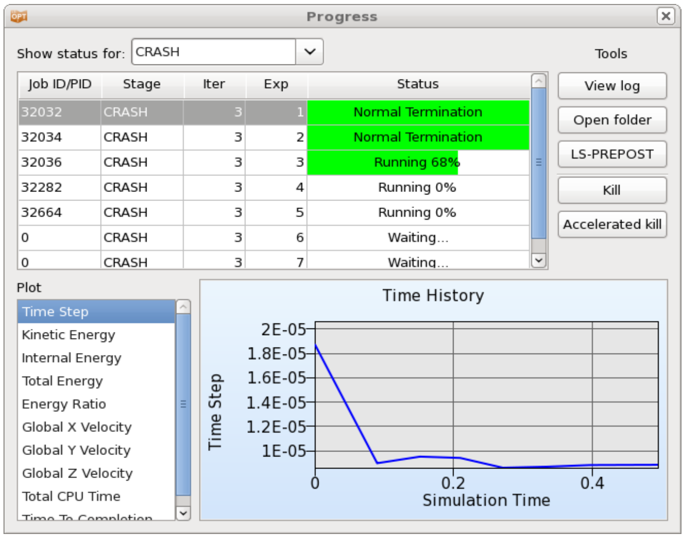
Understanding the mechanics of the queuing process should help to debug the installation:

1. LS-OPT automatically prepends runqueuer to the solver command and executes runqueuer which runs the submit_pbs script.
   - The runqueuer sets the variables LSOPT_HOST and LSOPT_PORT locally.
   - In the first example, the submit_pbs script spawns the dynscr script.
2. In Example 1, the queuing system then submits dynscr (see qsub command at the end of the submit_pbs script above) on the remote node which now has fixed values substituted for LSOPT_HOST and LSOPT_PORT.

   In Example 2, LS-OPT schedules the qsub command directly with LSOPT_HOST and LSOPT_PORT as arguments and i=DynaOpt.inp appended at the end of the command. i=DynaOpt.inp therefore serves as an argument ($1) to dynscr2.
3. The wrapper executes on the same machine as LS-DYNA, opens a socket and connects back to the local host using the host/port information. The standard output is then relayed to the local machine. This output is also written to the job_log file on the local host. To view the log of any particular run, the user can open job_log located in the respective run directory using any text editor, or double click on the status LED of the respective stage, select a run from the list displayed in the Progress dialog, and select the *View Log* button, see Section 15.3. The Progress dialog is shown below, Figure H - 3 , followed by the selected popup log.

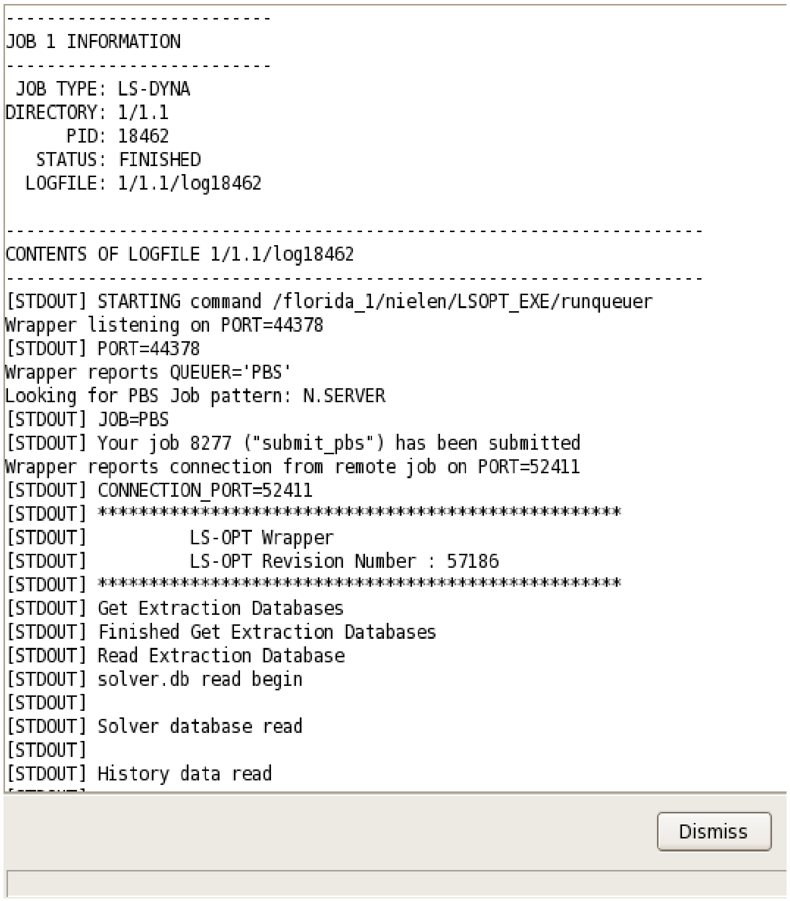
   An example of an error message resulting from a mistype of “wrapper” in the submit script is given in another example job_log file as follows:

   ```
   STARTING command /home/jim/bin/runqueuer

   PORT=56984

   JOB=LoadLeveler

   llsubmit: The job "1/1.1" has been submitted.

   /home/jim/LSOPT_EXE/Xrapper: Command not found.

   Finished with directory

   /home/jim/LSOPT/4.1/optQA/QUEUE/EX4a_remote/remote/1/1.1
   ```
4. The wrapper will also extract the data immediately upon completion on the remote node. Extracted data (the history.n and response.n files) are automatically transferred back to the local sub-subdirectory. If other parts of the database (e.g. d3plot files) are required (e.g. for post-processing with LS-PREPOST), the user has to specify these using appropriate LS-OPT settings (see Section 5.5.4). A log of the database extraction is also provided in the job_log file.



Job ID/PID	Stage	Iter	Exp	Status
32032	CRASH	3	1	Normal Termination
32034	CRASH	3	2	Normal Termination
32036	CRASH	3	3	Running 68%
32282	CRASH	3	4	Running 0%
32664	CRASH	3	5	Running 0%
0	CRASH	3	6	Waiting...
0	CRASH	3	7	Waiting...

Figure H - 3: Progress dialog



```
-------------------------
JOB 1 INFORMATION
-------------------------
 JOB TYPE: LS-DYNA
DIRECTORY: 1/1.1
      PID: 18462
   STATUS: FINISHED
  LOGFILE: 1/1.1/log18462

--------------------------------------------------------------------
CONTENTS OF LOGFILE 1/1.1/log18462
--------------------------------------------------------------------
[STDOUT] STARTING command /florida_1/nielen/LSOPT_EXE/runqueuer
Wrapper listening on PORT=44378
[STDOUT] PORT=44378
Wrapper reports QUEUER='PBS'
Looking for PBS Job pattern: N.SERVER
[STDOUT] JOB=PBS
[STDOUT] Your job 8277 ("submit_pbs") has been submitted
Wrapper reports connection from remote job on PORT=52411
[STDOUT] CONNECTION_PORT=52411
[STDOUT] ***************************************************
[STDOUT]         LS-OPT Wrapper
[STDOUT]         LS-OPT Revision Number : 57186
[STDOUT] ***************************************************
[STDOUT] Get Extraction Databases
[STDOUT] Finished Get Extraction Databases
[STDOUT] Read Extraction Database
[STDOUT] solver.db read begin
[STDOUT]
[STDOUT] Solver database read
[STDOUT]
[STDOUT] History data read
```

Figure H - 4: Opened job_log file using Progress dialog option

H.7 User-defined, Blackbox and “Simple” queuing systems

If the queuing system you want to use is not available in the queuing interfaces list in the **Stage** dialog, Section 5.5.1, there are three options to interface with any queuing system by using the User-Defined, Blackbox or Simple option. Blackbox can be used when the computers running the jobs are separated from the computer running LS-OPT by means of a firewall or you connect to a customized queuing system. Simple queuing system is a variant of Blackbox which has been simplified.

The key differences between User-defined and Blackbox are:

1. For Blackbox, it is the responsibility of the queuing system or the user provided scripts to transfer input and output files for the solver between the queuing system and the workstation running LS-OPT. LS-OPT will not attempt to open any communications channel between the compute node and the LS-OPT workstation.
2. For Blackbox, extraction of responses and histories takes place on the local workstation, whereas for User-Defined this is done on the computer running the job.
3. For Blackbox, LS-OPT will not run local placeholder processes (i.e. extractor/runqueuer) for every submitted job. This makes Blackbox use less system resources, especially when many jobs are run in each iteration.

The requirements needed for the respective approach are explained in the following sections.

H.7.1 User-defined queuing systems

To ensure that the LS-OPT job scheduler can terminate queued jobs, two requirements must be satisfied:

1. The queuer must echo a string

   ```
   Job ”Stringa Stringb Stringc …” has been submitted
   ```

   or

   ```
   Job Stringa has been submitted
   ```

   e.g.

   ```
   Job ”Opteron Aqs4832” has been submitted
   ```

   ```
   Job aqs4832 has been submitted
   ```

   The string will be parsed as separate arguments in the former example or as a single argument in the latter example. The string length is limited to 1024 characters. The syntax of the phrases “`Job `” and “` has been submitted`” must be exactly as specified. If more than one argument is specified without the double quotes, the string will not be recognized and the termination feature will fail.

2. A termination script (or program) `LsoptJobDel` must be placed either in the main working directory (first default location) or in the directory containing the LS-OPT binaries (second default). This script will be run with the arguments *stringA*, *stringB*, etc. and must contain the command for terminating the queue. An example of a Unix C shell termination script that uses two arguments is:

   ```
   #!/bin/csh -f

   aadmin –c $1 –j $2 stop
   ```

H.7.2 Blackbox queuing system

The Blackbox queuing system is another flavor of the User-defined queuing system. It can be used when the computers running the jobs are separated from the computer running LS-OPT by means of a firewall.

When using the Blackbox queuing system, a `LsoptJobDel` script is required, just as in the User-defined case. Furthermore, another script named `LsoptJobCheck` must also be provided. This script takes one parameter, the job ID, as returned by the submission script. The script should return the status of the given job as a string to standard output.

The Blackbox queuer option requires the user to specify a stage command that will queue the job. The command to queue the job must return a *job identifier* that has one of the following two forms:

```
Job "Any Quoted String" has been submitted
Job AnyUnquotedStringWithoutSpaces has been submitted
```

The Word "`Job`" must be the first non-white space on the line and must appear exactly as shown. Any amount of white space may appear between "`Job`" and the job identifier, as well as after the job identifier and before "`has been submitted`".

The Blackbox queuer requires the presence of two executable scripts `LsoptJobCheck` and `LsoptJobDel`. These scripts must be located in either in the current LS-OPT project directory or in the directory where the running LS-OPT program is located. (For Windows, the scripts must have an added extension `.exe`, `.vbs`, `.cmd` or `.bat`). If the Blackbox queuer option is invoked for some stages, then LS-OPT checks for the existence of executable scripts in one of these locations, and refuses to run if the `LsoptJobCheck` and/or `LsoptJobDel` scripts cannot be found or are not executable. The project directory is searched first.

H.7.3 Simple queueing system

The Simple queuing system is a variant of the Blackbox queuing system which does not require the user to write a `LsoptJobCheck` script. Instead termination is controlled by user-defined termination criteria in the LS-OPT GUI under the “Termination” tab in each Stage.

The Simple queuing system supports the same *job identifier* string as Blackbox, which will be used to kill the job by calling a user provided `LsoptJobDel` script. This is however optional. The job identifier may also be used as a reference in user-defined termination criteria. When matching a line containing “_JOBID_”, the actual job identifier will replace “_JOBID_” before finding a match.

Since there is no script to check whether jobs have been started correctly, it is assumed that all jobs are started successfully and running until any of the user-defined termination criteria has been met. Much care must be taken when considering this option since it is possible that a job is never detected as finished when no termination criteria is met but the job has died or was not started for any reason.

H.7.4 LsoptJobCheck script

The user-supplied `LsoptJobCheck` script is run each time LS-OPT tries to update the current status of a job. The `LsoptJobCheck` script is run with a single command line argument:

```
LsoptJobCheck job_identifier
```

The working directory of the `LsoptJobCheck` script is set to the job directory associated with job_identifier.

The script is expected to print a status statement that LS-OPT can use to update its status information. The only valid status statements are:

String	**Description**
`WAITING`	The job has been submitted and is waiting to start
`RUNNING`	The job is running.
`RUNNING` *N/M*	After `RUNNING`, the script may also report the progress as a fraction. `RUNNING 75/100` means that the job has ¼ to go. The progress information will be relayed to the user, but not used in any other way by LS-OPT.
`FAILED`	The job terminated with an error. LS-OPT will update the finished file with error termination. This can also be used when the underlying queuing system reports a problem. A job that has terminated with an error or with an invalid statement is considered as a failed run.
`FINISHED`	The job completed with normal termination and any output files needed for extraction has been copied back to the run directory.
`ABORTED`	If a job reports `ABORTED`, then the status bar in the Progress dialog of LSOPT UI turns yellow, and the number of retries is shown.

Any amount of white space may appear at the beginning of a status statement, and anything may appear after these statements. The optional *N/M* argument for `RUNNING` is interpreted as an estimate of the progress; in this case *N* and *M* are integers and *N/M* is the fractional progress. *N* must not be larger than *M*.

If `LsoptJobCheck` terminates without printing a valid status statement, then it is assumed that `LsoptJobCheck` does not function properly, and LS-OPT terminates the job using the

LsoptJobDel script. All output from the LsoptJobCheck script is logged to the job log file (job_log) in the run directory for debugging purposes.

Note: The LsoptJobCheck script may print more than one status statement, but only the first one will be used to update the status.

H.7.5 LsoptJobDel script

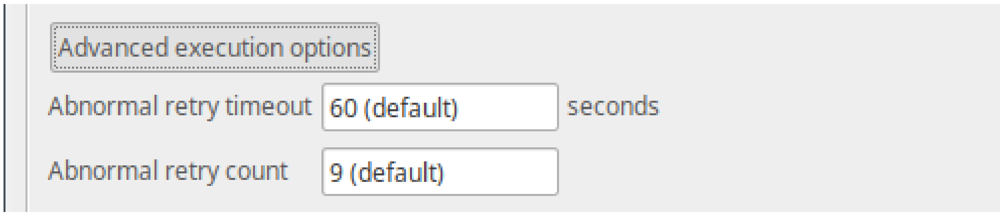
The user-supplied LsoptJobDel script is run whenever the user chooses to terminate a job, or whenever LS-OPT determines that a job should be killed (for example, if LsoptJobCheck fails). The LsoptJobDel script is run with a single command line argument:

```
LsoptJobDel job_identifier .
```

The working directory of the LsoptJobDel script is set to the job directory associated with *job_identifier*.

H.8 Abnormal termination and retrying the job submission

H.8.1 User-defined abnormal termination



Abnormal retry timeout 60 (default) seconds

Abnormal retry count 9 (default)

Figure H - 5: Stage dialog: advanced execution options for abnormal terminations

It may be prudent to retry job submissions for certain types of abnormal termination. For this purpose, the user can specify an Abnormal signal for terminations which are neither normal nor error termination. A job that has terminated in this way can then be retried by the LS-OPT job scheduler. The Abnormal signal should be sent to standard output from the simulation script.

Two parameters can be used to set the number of retries allowed and timeout for each retry. The defaults are 9 retries with a timeout of 60 seconds. These are available under *Advanced execution options*

H.8.2 Queuer timeout

A special case exists in which the LS-OPT job scheduler automatically generates an Abnormal signal. This is whenever the wrapper has not been executed for a specified timeout period. For this case, a queuer timeout can be specified. The queuer timeout is the time it will wait for the wrapper to connect, otherwise it sets an abnormal termination status and writes an Abnormal signal to standard output. In this case the job will be resubmitted for the number of retries specified and using the queuing timeout for each retry. The timeout default is 720 minutes.

H.9 SSH support

A Linux host running an SSH (Secure Shell) server may be used as either a machine to run jobs directly on or as a proxy machine to submit jobs to a Linux cluster. For instance, a Windows host can use a Linux host to submit jobs on a Linux queuing system. LS-OPT supports connections via OpenSSH or plink client. On most Linux distributions the OpenSSH client is by default. On windows, the executables plink and psftp are provided with LS-OPT distribution package. Plink is an SSH client for running a command and return. Psftp is a Secure FTP client which runs over the SSH protocol.

The main component for interacting with a host using SSH is the executable *lrunremote*, which is provided in the LS-OPT distribution package and called from LS-OPT seamlessly. LS-OPT will, with the help of lrunremote, an SSH client and an SFTP client:

1. Check the remote hosts connectivity and information, such as if the run directory is shared between the local and remote host. This check involves copying over a bash script to the remote host and execute it remotely.
2. Copy input files to the remote host as required
3. Run a command on the remote host, along with any stdout/stderr output from the command and the return code of the command.
4. Copy results from the remote host as required
5. Optionally remove the remote run folder of a successful run.

Some important notes to consider:

- Only the local host will need to connect to the remote host, which is typically the TCP port 22.
- Copying will only occur if LS-OPT fails to detect a common filesystem between the local and remote host. Copying however will always be performed as part of checking the remote host, so the SFTP/PSFTP client is required.
- SSH connections must remain open for the whole time LS-OPT is running simulations. Connections cannot be resumed. Great care must be taken to have a stable network environment.
- There might be limitations on the OpenSSH server on how many simultaneous connections that may be accepted or running. LS-OPT will start a lot of connections in short time frames which may stress the remote host. Either adjust the SSH server settings or make use of the built-in feature in LS-OPT to limit number of simulations and/or increase the delay between starting new jobs.
- Administration and configuration of the SSH server, including authentication is not covered in this manual. Refer to the manual of your SSH server.

H.9.1 Authentication methods

In order to use LS-OPT with SSH it is required that an authentication method which does not require human interaction to be available. LS-OPT will always run the SSH clients with batch mode enabled, to avoid hanging prompts. Challenge passwords, known as the server-side setting *ChallengeResponseAuthentication*, are not supported.

Due to security concerns, passwords are not supported at all. Such a solution would require either a multitude of password prompts to be presented to the user or the use of a caching mechanisms and/or storage of password in configuration files. There are better methods of authentication!The recommended method of authentication is to use an SSH agent coupled with a private key. OpenSSH uses the ssh-agent and Putty utilize the pageant executable. This allows a private key to be password protected but unlocked via the agent when required. LS-OPT requires the key to be unlocked for the whole time of running on a cluster. It is also possible to use host-based authentication, Kerberos authentication or any other means of authentication that allows LS-OPT to connect by only providing a username to SSH. Any of these methods of authentication options is referred to as the option “None/SSH-Agent” in the GUI.

Lastly, a private key file can be used for the connection. This private key cannot be password protected for the same reasons as for regular passwords.

H.9.2 Installation on remote host

The remote host needs to have a package of LS-OPT installed which corresponds to the version used locally. It is only guaranteed to work with the same version and possibly revision but different. Version 6.1 of LS-OPT will not work with previous versions due to major changes in the scheduler and utility executables.

It is recommended to extract the **whole** LS-OPT package on the remote host in a folder to not miss future changes. The executables that needs to be installed on the remote host are:

- extractor, for locally runned jobs on the node
- ffshell and ffbuilder, for constructing FFNN metamodels
- runqueuer, for jobs to be queued on a queuing system, as described in H.5.1

H.9.3 Using SSH in a project

First thing to do is to setup SSH by adding a host configuration. Host configurations are added in the “Scheduler” tab in the Settings dialog. Another way to get there is in the Stage dialog, “Remote” tab, by checking the “Run via SSH Proxy” option and then clicking “Manage…”. The settings are described in Section 5.5.2. It is encouraged to test the host configuration by clicking the “Test Connection”. This will tell whether the connection is successful or not. If not, it will show any messages produced from the SSH client with verbose flag on. It is also possible to open a separate prompt to the remote host, which is useful to check whether the host key needs to be accepted locally or to prepare the filesystem on the remote side.

With an SSH host configuration in place, a stage can be set to use the SSH configuration by going in the Stage dialog, “Remote” tab and checking “Run via SSH Proxy”, after which the configuration can be selected. It is a also possible to use it in FFNN metamodels in the Samping “Execution” tab.

If the local and remote host does not share a filesystem, i.e. the local run folder does not correspond to the same folder on the remote host as per the host configuration, cleaning of the remote host folder may occur. This behavior is controlled by the Stage setting “Clean remote folder after run” in the “Remote” tab. The options available are:

- **Default**: Each remote run folder is removed for each successful run. Erroneous runs will be kept on the remote side for inspection. However, this method may remove raw databases such as d3plot unless these have been requested to be copied over in the “Recover files” section in the Stage “Remote” tab.
- **Always**: Always perform removal of the remote run folder regardless of outcome.
- **Never**: Always keep the remote run folder.

The solver command must be adapted to work on the remote host. So, any binary and path should be written as how it needs to be executed on the remote side. No conversion will be made by LS-OPT.

H.10 Enabling LSTCVM job proxy support

LSTCVM is a Proxy Server for distributing solver jobs across a computer cluster, e.g. for running LS-OPT on a Windows machine controlling solver jobs on a Linux cluster. LSTCVM is available for download at https://ftp.lstc.com/user/ls-opt/lstcvm.

H.10.1 LSTCVM options

There are two ways that LS-OPT can work with the LSTCVM job proxy.

1. LSTCVM and LS-OPT share a common file system.

   If LSTCVM and LS-OPT share a common file system, then you may run LS-OPT jobs from within the shared file system by using the stage command

   ```
   lstcvm_run remote_solver_command
   ```

   For example

   ```
   lstcvm_run ls971_single
   ```

   would be the appropriate stage command in LS-OPT if you want to run the "`ls971_single`" command on the remote LSTCVM server.

2. LSTCVM and LS-OPT do not share a common file system.

   In this case, you may still execute remote commands on the LSTCVM server, but you must select the following option in the **Stage** dialog Setup tab, Execution: ***Use LSTCVM proxy***, Section 5.5.3. LS-OPT will take care of prepending the `lstcvm_run` command. So, in this case, if you want to execute "`ls971_single`" on the remote LSTCVM server, then your stage command should simply be

   ```
   ls971_single
   ```

All necessary input files will be transferred to the remote LSTCVM server using LS-OPT runqueuer/wrapper commands. Extraction results are automatically brought back to the local side once the job has finished.

Note: In order for this option to work, you must install the LS-OPT "`wrapper`" on the LSTCVM proxy server, and you must add the following entry to the executable map file "`lstcvm.exemap`"

```
wrapper -> full_path_to_wrapper
```

The "wrapper" command is architecture specific. So be sure to obtain the correct program for the LSTCVM architecture.

REMOTE FILES: We do not currently delete files on the LSTCVM server after the job has completed. This must be done by the LSTCVM proxy server administrator.

H.10.2 LSTCVM server installation

The LSTCVM server is distributed separately from LS-OPT and, in addition to the executables, contains detailed information and installation instructions. This server installation is usually handled by a systems administrator.

H.10.3 Environment Variables

All solver environment variables defined in the LS-OPT **Environment Variables** tab of the Stage dialog (Section **5.4.2**) are automatically passed to the remote job on the LSTCVM server. (PATH is not passed for security reasons). This provides a convenient way to define licensing variables for LS-DYNA. For example, you can pass the following variables to the remote proxy server job:

```
LSTC_LICENSE=network
LSTC_LICENSE_SERVER=license_server_name
```

H.10.4 Configuring the `lstcvm_run` client

In order to configure the "`lstcvm_run`" client, you should execute

```
lstcvm_run -s lstcvm_server_name
```

The information will be saved so that this step never needs to be repeated. If you are running on a Microsoft Windows platform, then you should execute this command from within a command prompt; the server information will be saved in the Windows registry. If you are running on a Linux/UNIX platform, then the server information is stored in $HOME/.lstcvm . If, for some reason, a port other than the default is used, then you must specify the port number N with the command

```
lstcvm_run -s N@lstcvm_server_name
```

After setting the server name, then you can test for connectivity using

```
lstcvm_run -info
```

You should see information about the current configuration of the LSTCVM server.

To test the installation, 'cd' to a directory where you are allowed to run the lstcvm_run client, and issue the command

```
lstcvm_run ls -al
```

It is possible that this command will fail if the LSTCVM administrator does not allow the "ls" command to be run. If that is the case, then check with the administrator about which commands are available.

Once you know that the lstcvm_run command is properly configured and able to execute commands remotely, you are ready to use lstcvm_run with LS-OPT. Only commands which are allowed and enabled by the LSTCVM administrator will function properly. For example, ls971_single is not available unless the remote administrator has enabled this command.

H.11 Getting Started with the LSTCVM Proxy server

The *lstcvm* proxy server facilitates executing LS-OPT on a local Windows/Linux machine, whereas the solver jobs can be submitted to a remote Linux/Unix server. If this remote server has a queuing system, then the solver jobs can also be submitted to the queuing system.

The basic functionality of the proxy server is to map two directories, one in the local system with the other being at the remote end (Linux/Unix) in such a way that the commands executed in the local directory are transferred to the server directory and executed there. For example, consider the mapping between two directories; dir1 and dir2 given as

C:\user\imtiaz\Desktop\dir1	->	/home/imtiaz/Desktop/dir2
(Windows)		(Linux/Unix)

In the above example, dir1 is the local directory (i.e. Windows) and dir2 is the remote server directory. Using this mapping and being in dir1, typing a command using lstcvm to list the files (i.e. by using the lstcvm_run executable as prefix to command '*ls*') will result in listing the files of dir2. In this way, remote commands can be executed from the mapped local directory.

LS-OPT utilizes this capability to execute solver jobs on the remote server by transferring the necessary input files to the server directory (dir2). Once the analysis is completed, the corresponding output files are transferred back to the local directory (dir1) where LS-OPT is running. This is illustrated using an example problem in Figure H-8.

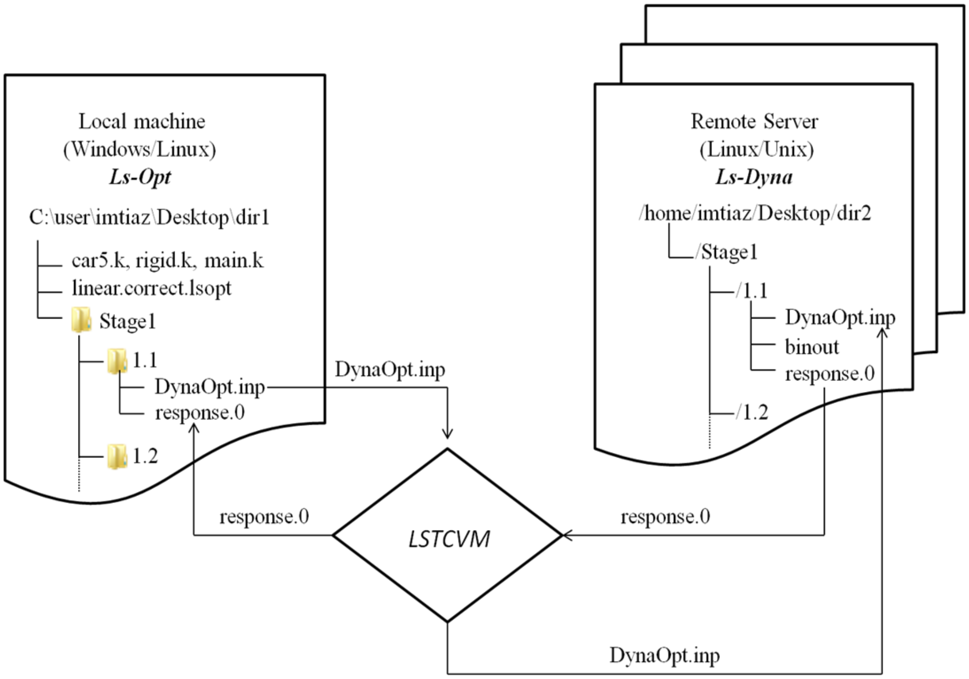


Figure H - 6: Using lstcvm with LS-OPT

DynaOpt.inp, which is the LS-DYNA input file for a design point, is transferred to the remote server hosting the LS-DYNA executable. Once this input file is analyzed, the corresponding response outputs (*response.0*, *response.1* etc) are extracted from the LS-DYNA output files and transferred back to the local directory (dir1).

H.11.1 Installation of LSTCVM

lstcvm is available for download at `http://ftp.lstc.com/user/ls-opt/lstcvm`. Download and install the *lstcvm* build suitable for the remote server.

The *lstcvm* distribution consists of all the necessary executables along with a few sample files used to set up the proxy server. The sample files are provided in the SAMPLE_CONFIG directory. These are *lstcvm.config*, *lstcvm.dirmap*, *lstcvm.exemap*, *lstcvm.usrmap*. All the variables required for setting up the proxy server are initialized in these files.

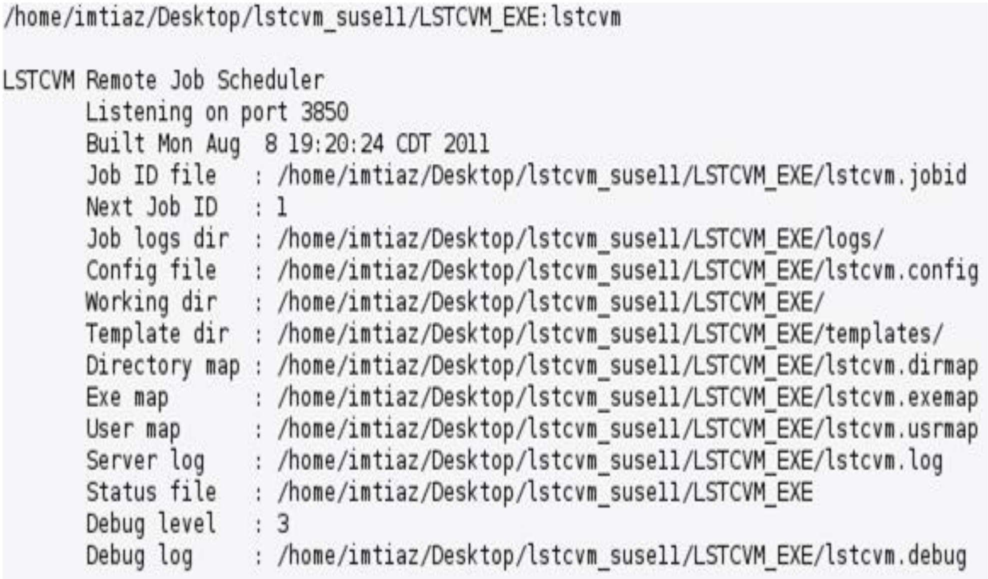
H.11.2 Setting up LSTCVM

From the remote server end (Linux/Unix):

- While setting up the proxy server, the system administrator of the remote server has to define which directories are to be mapped, which executables are allowed to run using the proxy server and who (which user) can execute the commands using *lstcvm*. This information is provided using the *.dirmap*, *.exemap* and *.usrmap* files which are called by the configuration file *lstcvm.config*.
- Note: Mapped directories should have the same subdirectories to execute remote commands from the subdirectories.

- ***lstcvm.dirmap*** defines the information/absolute path of the mapping directories. The local directory should be on the left hand side and the server directory on the right. The following variable maps dir1 with dir2.
- C:\user\imtiaz\Desktop\dir1 -> /home/imtiaz/Desktop/dir2
- ***lstcvm.exemap*** is used to specify the programs that can be executed using the proxy server. For example with the following mapping of the executable,
- ls971_single -> /home/imtiaz/bin/ls971_single
- being in dir1(local), and if dir2 (remote server) has an LS-DYNA keyword file and assuming mapping between dir1 and dir2, executing the command `lstcvm_run ls971_single i= keyword.k` from the local machine will result in an LS-DYNA analysis of the keyword file at the remote end in dir2. Mapping should be provided even to list the files using the *ls* command i.e.
- ls ⇨ /bin/ls
- To use *lstcvm* with LS-OPT, the wrapper provided with the LS-OPT distribution should be mapped with its full path as shown below:
- wrapper -> /home/imtiaz/LSOPT_EXE/wrapper
- ***lstcvm.usrmap*** is used to allow mapped users to execute commands using *lstcvm.* If a user has the same username on both the local machine and the remote server, then no mapping is required. For other cases a mapping rule has to be defined in *lstcvm.usermap*. For example, for a mapping rule given as, (usernames on the left correspond to the local machine while those on the right correspond to the remote server)
- Jones -> Jack
- Fred -> Jack
- * -> nobody
- `*          ->    (none)`
- For the first two rules, the local user *Jones* is mapped with remote user *Jack* and the commands are executed on the server if there is a remote user with the name *Jack.* Similarly, commands executed by the local user *Fred* are run using the remote username *Jack* if that username exists on the server machine. The third rule allows all other local users to execute commands as user *nobody* if that username exists. Any user including * mapped to (none) will not be allowed to run commands on the server.
- ***lstcvm.config*** is the main configuration file which has all the information about the proxy server including the working directory, job log directories, server port, server log and debug log files and all the mapping files apart from other miscellaneous options. The server log and debug log files consist of all the commands executed through the proxy server along with error information (if any). In the sample configuration file provided by the *lstcvm* distribution, the default port number is given as 3850. This is the port number of the remote server through which *lstcvm* listens. If the default port is not open on the remote server, the user has to ask the system administrator to open a port within a range of 1025-65535.

- Note: More detailed information is provided in the self-documented sample files provided with the *lstcvm* distribution.
- Once the configuration and mapping files are defined, the proxy server can be started by running the executable *lstcvm*. Once the server has been started, you should be able see the following information



```
/home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE:lstcvm

LSTCVM Remote Job Scheduler
       Listening on port 3850
       Built Mon Aug  8 19:20:24 CDT 2011
       Job ID file    : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.jobid
       Next Job ID    : 1
       Job logs dir   : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/logs/
       Config file    : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.config
       Working dir    : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/
       Template dir   : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/templates/
       Directory map  : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.dirmap
       Exe map        : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.exemap
       User map       : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.usrmap
       Server log     : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.log
       Status file    : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE
       Debug level    : 3
       Debug log      : /home/imtiaz/Desktop/lstcvm_susell/LSTCVM_EXE/lstcvm.debug
```

Figure H - 7: lstcvm startup

From the local machine (Windows/Linux):

- To execute commands from the local machine, `lstcvm_run` is appended to each command. The executable `lstcvm_run` is available in the *lstcvm* distribution but for using it from a local machine, the *lstcvm* distribution suitable for the local machine architecture should be downloaded (i.e. if the local machine is Windows, the `lstcvm_run` executable for Windows must be used). Once the proxy server has been started from the server end, the environment has to be set on the local machine given as (if Linux):
- `LSTCVM_SERVER` server_host_name
- `export` server_host_name
- For a Windows machine, registry settings are used to store the server name. This can also be done by issuing the following command on Windows,
- `lstcvm_run.exe -s` server_host_name (or)
- `lstcvm_run.exe -s` non_standard_port@server_host_name
- Once these steps have been followed, the proxy server can be tested by issuing the command `lstcvm_run ls` from dir1 which should result in listing the files of dir2 of the remote server.

H.11.3 Example Problem

- Here a simple car crash example similar to the one in Section 20.2 is optimized using LS-OPT with the *lstcvm* proxy server.LS-OPT is executed on a local Windows machine and the solver jobs (LS-DYNA) are scheduled on a remote Linux server.
- The LS-DYNA input files and LS-OPT job file required for this example can be obtained from the LS-OPT training examples set.
- (lsopt_50_training_examples/design_optimization/simple/linear)
- The example requires the following files:
- `main.k` (main solver input file consisting of LS-OPT parameter definitions)
- `car5.k` (include file specified in `main.k`)
- `rigid.k` (include file specified in `main.k`)
- `linear.correct.lsopt` (LS-OPT input file)

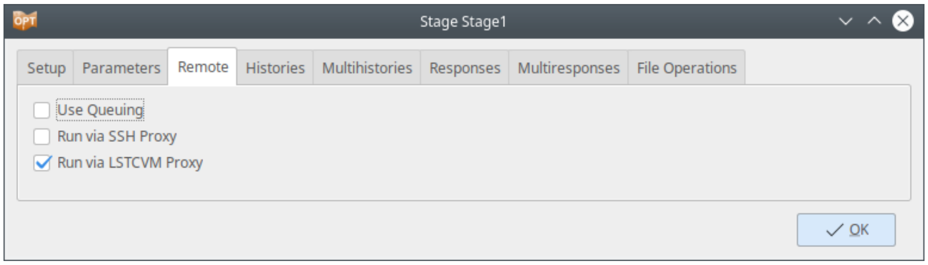
Remote server end:

- The *lstcvm* proxy server is started on the Linux server using the following configuration and mapping files.
- ***lstcvm.config***: (Located in the directory same as *lstcvm* distribution)
- SERVER_WORKING_DIRECTORY = /home/imtiaz/Desktop/dir2
- JOB_LOG_DIRECTORY = /home/imtiaz/Desktop/dir2
- TEMPLATE_DIRECTORY = /home/imtiaz/Desktop/dir2
- JOB_STATUS_FILE = /home/imtiaz/Desktop/dir2
- `SERVER_PORT = 3850`
- SERVER_RUN_IN_FOREGROUND = 1
- SERVER_DEBUG_LOG = lstcvm.debug
- SERVER_LOG = lstcvm.log
- JOB_EXECUTABLE_MAP = lstcvm.exemap
- JOB_DIRECTORY_MAP = lstcvm.dirmap
- JOB_USER_MAP = lstcvm.usrmap
- JOB_ID_FILE = lstcvm.jobid
- (Note: In the above sample files, the files without full path are located in the working directory)
- lstcvm.exemap:
- `ls -> /bin/ls`
- ls971_single -> /home/imtiaz/bin/ls971_single

- wrapper -> /home/imtiaz/LS-OPT/LSOPT_EXE/wrapper
- lstcvm.dirmap:
- C:\user\imtiaz\Desktop\dir1 -> /home/imtiaz/Desktop/dir2
- lstcvm.usrmap:
- imtiaz -> imtiaz
- * -> (none)
- The *lstcvm* proxy server is started using the *lstcvm* executable.

GUI Setup (Local machine):

- In the local Windows machine, the remote server name is stored using the registry settings explained in the setup process.
- The files used in the example should be present on the local Windows machine (dir1) and once LS-OPT is executed with *lstcvm*, the LS-DYNA input files are copied to each run directory on the remote server (dir2).
- Modify `linear.correct.lsopt` by checking the option *Use LSTCVM proxy* in the *Stage setup* dialogue box as shown in Figure 1-3. Using this check box, the user does not have to append *lstcvm_run* on the command line. The defined LS-DYNA command should correspond to the executable mapped in the *lstcvm.exemap* file.



□ Use Queuing
□ Run via SSH Proxy
☑ Run via LSTCVM Proxy

Figure H - 8: lstcvm GUI setup

- Using the example command `ls971_single` represents the LS-DYNA executable present on the remote server. This executable mapping should be defined in the *lstcvm.exemap* file.
- Running the LS-OPT example will copy the LS-DYNA input files `car5.k`, `rigid.k` and `DynaOpt.inp` (originally `main.k`) to all the LS-OPT run directories within dir2 of the remote server.
- Once the LS-DYNA analysis is completed, the response files `response.0`, `response.1` etc. are automatically transferred back to the run directories of dir1 i.e. to the local Windows machine. These files are then utilized by LS-OPT to perform optimization on the local Windows machine. The files for further iterations are copied back and forth from the local machine to the remote server until the LS-OPT process terminates.

H.12 Passing environment variables through LS-OPT

LS-OPT provides a way to define environment variables that will be set before executing a solver command.

Passing environment variables to stage commands can be a convenient way to control the behavior of a command. For example, the command might be a script which queues a job on a remote machine; the environment variable settings might be used by the script to select various queuing options. Or, the environment variable settings might be passed along through the queuing system to set options for the remotely executed job, such as license server locations, input file names, whether to run the MPP version of LS-DYNA, whether to run a single or double precision solver, etc.

Environment variables can be set manually in the GUI, or provided in a file that is loaded to the GUI, Section H.12 .

Executables, `*.env` files and `*.lstcsh` files are currently the only valid file types. The default location of these files is

```
$HOME/LSOPT_SCRIPTS
```

You can redefine the search location for scripts by setting an environment variable LSOPT_SCRIPTS to the desired directory location.

NOTE: Windows does not set a HOME environment variable, even though there is a home directory for each user. A command prompt, for example, opens in the home directory of the user.

H.12.1 .env files

The simplest way to import a group of environment variables into LS-OPT is through the use of an environment variable file. For example, create a file “`test.env`” in $HOME/LSOPT_SCRIPTS with these five lines in it

```
# This is a comment line
LSTC_LICENSE=network
LSTC_LICENSE_SERVER='server1 server2 server3'
LSTC_LICENSE_PORT=31020
LSTC_EXE=ls971_R4
```

Save the file.

There are a few formatting rules that should be observed for a “`.env`” file:

1. Any line which begins with `# ! @ $ % & ;` : is treated as a comment line.
2. NAME=VALUE lines should not contain white space between NAME and =, or between = and VALUE.
3. White space may appear before NAME, at the beginning of a line
4. If VALUE contains white space, then use NAME='VALUE' as shown above. (This is an acceptable form, whether or not spaces appear in VALUE.)

H.12.2 Executables

You can import a group of environment variables by creating an executable that prints a space-separated list of NAME=VALUE pairs to standard output, all on one line. This list must appear in a ***single*** line of output, the last line of output from the program; previous lines of output are ignored. There should be no space between NAME and =, or between = and VALUE. If VALUE must contain spaces, then use NAME='VALUE'. The single quotes are optional if value does not contain spaces.

For example, the single output line shown below is valid (it has been broken for display purposes only):

```
exe=/home/trent/LSTC/PERL/lsdyna-caec01_pbs_sub.pl menu=batch
time=1:00 host=abcdefgh07 procs=1 jobname='My Job' project=isd
email=No delay=No preemptable=No version='LS-DYNA 970 MPP SP 6763'
ioloc=/home/trent inpfile=DynaOpt.inp mem1=auto mem2=auto
pfile=Generic dumpbdb=No dynamore=No clean=No tail=No copycont=No
optimization=LsOpt
```

The main reason to use a program to set variables in bulk, instead of a ".env" file, is that an interactive program can take advantage of the *Edit browse list* feature, which is described in Section **5.4.2**.

.lstcsh files

These types of files are specialized script files which requires the interpreter "lstcsh" (or "lstcsh.exe" on the PC). This interpreter is included in the LS-OPT distribution, and it is designed to generate the output format described for "*Executables*" above. It is also designed to allow interaction with the *Edit browse list* feature, Section **5.4.2**. These scripts define graphical programs with standard user-interface components for selecting, modifying, or specifying environment variables. For example, if you have complex and specialized queuing options, then ".lstcsh" script files could be useful for you. Please contact LSTC for more information.

WARNING: LS-OPT creates a special browse variable when importing a variable list. This variable records the program name used to create the Browse List. The user-supplied browse program should never define the browse variable in its output. The name browse should be treated as a reserved name.

A simple Linux browse command could be a shell script:

```
#!/bin/bash
echo This line is ignored. Only the last line survives
   echo A=B C=D
```

The LS-OPT GUI offers an option **Edit Browse list.** If a valid Browse List is present in the Environment Variables list, then selecting this option will run the original program that created the Browse List, together with all of the current Browse List options passed as command line arguments, one per existing environment variable.

Executing the 'Edit Browse List' will cause the original file to be reread, which is convenient for testing purposes.

Each command-line argument has the form *name=value*. However ‘value’ is not single-quoted because each *name=value* argument is a separate command-line argument. The customer-supplied browse command should offer the user an opportunity to edit the existing variables, and the browse command should return the newly edited list on one line, in the same format as described above. This would normally be done through some sort of graphical user interface. The returned list will be used to replace all of the previous Browse List.

The next example script returns an initial Browse List consisting of two variables, A and C. Invoking the editing feature appends a new variable (tN=N) to the list.

```
#!/bin/bash
echo This line will be ignored. Only the last line survives.
if [ “$1” == “” ]; then
  echo A=B C=D;
else
  echo $* “t”$$”=”$$;
fi
```

When this script is invoked using the “Create by Browse” feature, there are no command-line arguments, and the script prints “A=B C=D” to standard output. However, when the script is invoked using the edit feature for the first time, two command-line arguments “A=B” and “C=D” are passed to the script. This time the return line consists of the original command-line arguments (printed using $*) and tN=N, where N is the PID of the shell process. If the editing feature is invoked a second time, then three command-line arguments are passed to the script (“A=B”, “C=D”, and “tN=N”). Another new variable tN is appended, where N is the newest PID of the script process. This sample script has little practical value, except to illustrate how existing variable settings are passed by command-line to the previous browse command, and to illustrate how one can use the editing feature to modify or add new variables.

H.13 Troubleshooting

1. To increase debug output of the job scheduler and job logs, either set scheduler debug level to DEBUG in the LS-OPT GUI settings dialog or define environment variable LSOPT_DEBUG as “10” for maximum debug output. Re-run any job if possible for the or make an isolated test with the new log level.
2. Diagnostics for a failed run usually appear in the `job_log` file in the run directory. If there is almost no information in this file, the wrapper path may be wrong, or the submission script may have the wrong path or permission. For any job, this file can be viewed from the Progress dialog, Section 15.3. The file lscheduler6.log contains scheduler information. This file may contain a lot of information.

   Please attach the log files (lsopt_output, lscheduler6.log and possibly job_log) when emailing lstsupport@ansys.com.
3. Make sure that the permissions are set for the executables and submission script.
4. Check all paths to executables e.g. "wrapper", etc. No diagnostic can detect this problem.

5. Make sure that the result database is produced in the same directory as where the wrapper is started, otherwise the data cannot be extracted. (E.g. the front-end program such as mpirun may have a specification to change the working directory (`-wd dir`)).

6. *Running on a remote disk.* Make sure that the file "`HostDirectory`" is not copied by a user script to the remote disk if the simulation run is done on a remote disk. The "`HostDirectory`" file is a marker file which is present only on the local disk. Its purpose is to inform the wrapper that it is running on the local disk and, if found on a remote disk, will prevent the wrapper from automatically transferring extracted results back to the local disk. In general the user is not required to do any file copying since input files (including LS-DYNA include files) are copied to the remote disk automatically. The response.*, history.*, result.* and matrix.* files are automatically recovered from the remote disk. Other files can be recovered using the feature in Section **Error! Reference source not found.** .

7. The LSOPT environment variable defines the path of the LSOPT_EXE directory extracted from the distribution .zip or .gz file during installation. This variable is automatically detected when executing LS-OPT and set internally and *should not be set* by the user. Setting this variable externally may have the effect of launching an unintended LS-OPT process (e.g. wrapper, extractor, lsopt, etc.) that could cause errors during execution.

8. A path which is part of the solver command line should be in quotes if the path has spaces e.g.

   "C:\Program Files\MPICH2\bin\mpiexec.exe" -np2 C:\LSDYNA\mpp-dyna\mpp971.exe i=DynaOpt.inp

   is correct whereas

   C:\"Program Files"\MPICH2\bin\mpiexec.exe -np2 C:\LSDYNA\mpp-dyna\mpp971.exe i=DynaOpt.inp

   is *incorrect*. Note that the full path, including the filename, must be in quotes.

9. *Termination of user-defined programs:* LS-DYNA always displays a '`N o r m a l`' at the end of its output. When running a user-defined program which does not have this command displayed for a normal termination, the program has to be executed from a script followed by a command to write '`N o r m a l`' to standard output. The example file *runscript* shown below first runs the user-defined solver and then signals a normal termination.

   ```
   mpiexec -n 2 /home/john/bin/myprogram -i UserOpt.inp
   # print normal termination signal to screen
   echo 'N o r m a l'
   ```

   which is submitted by the wrapper command in submit_pbs as:

   ```
   /home/john/bin/wrapper /home/john/bin/runscript
   ```

   Note: Adding "`echo N o r m a l`" at the end of the wrapper command (after a semicolon) does not work which is why it should be part of the script run by the wrapper.

   Note: This termination criteria can be defined in the termination tab for user-defined solvers or jobs using “Simple” queuing system.

Appendix I: Killing Jobs

I.1 Overview of How Jobs are Killed

There are only a few methods which are used to kill a job, regardless of how the job is run, and regardless of the type of job.

- **A D3KIL file** is created to kill LS-DYNA jobs. LS-DYNA will save its data, update restart files, and exit cleanly. Unfortunately, depending on the type of job, LS-DYNA may not be able to check for the presence of a D3KIL file in any short period of time. This is especially true of large implicit jobs where large matrix solves are not interrupted for efficiency reasons.
- **SIGINT signal (Ctrl+C under Windows).** Many programs are designed to trap such signals and to exit gracefully. Others may deliberately ignore such signals. This signal cannot be used to kill LS-DYNA jobs, because LS-DYNA traps this signal to activate a sense switch.
- **SIGTERM signal (Ctrl+Break under Windows).** This is a little more forceful than SIGINT or Ctrl+C option for killing programs. Programs which ignore the SIGINT or Ctrl+C may exit upon receiving this signal, either voluntarily or involuntarily. Programs are allowed to catch or to ignore such signals.
- **SIGKILL signal (TerminateProcess() process under Windows).** This is the most reliable method for killing a job, but it may result in lost resources because programs cannot generally catch or ignore such kill requests. Linux processes killed in this way will not be able to flush unwritten data, or to close files. Linux jobs killed in this way may result in <defunct> processes. Windows processes killed in this may be unable to properly release DLL resources. This Windows kill option is the same mechanism used by the Windows Task Manager to 'force' kill a job, and the same warnings apply.
- **A queuing system job deletion command** may be issued. This generally results in a sure kill of a remote running job, but there is no opportunity to save files or other data. This is a remote queuer-specific version of the SIGKILL option.

The biggest determining factor in how a job is killed is the manner in which the job is scheduled:

1. Locally.
2. Indirectly through a **runqueuer/wrapper** solution.
3. Indirectly using the **BLACK BOX queuing** option.

There are LS-DYNA and generic jobs. How a job is killed depends on how it is run and queued. Queued jobs can be stuck in strange states that require special handling. LS-DYNA jobs are handled differently in order to preserve data that might be useful.

I.1.1 Case 1 (Killing Local Jobs):

Local jobs are killed using D3KIL files and/or signals.

- The first attempt to kill an LS-DYNA jobs is done by creating a D3KIL file. The SIGTERM method is used if LS-DYNA does not respond within 2 minutes. If that also fails, then a SIGKILL signal is delivered. If that fails, then the job is abandoned.
- Generic jobs are killed using signals, starting with SIGINT. If that fails after about 2 minutes, then SIGTERM is used. If that also fails, then a SIGKILL signal is delivered. If that fails, then the job is abandoned.

I.1.2 Case 2 (Killing Jobs run using runqeuer/wrapper):

This option is used to monitor/control jobs which are run remotely using a queuing system. A local runqueuer process communicates with a remote wrapper process in order to relay terminal I/O, to transfer files, and to deliver signals.

If you try to kill a job started using runqueuer/wrapper before these programs have been able to negotiate a connection, then runqueuer will exit, and the remote job will be unable to start.

If you try to kill a job after runqueuer and wrapper have negotiated a connection, then the method of killing the job is much like that for a locally run job. By design runqueuer will handle any SIGINT and SIGTERM received and relay those signals to the wrapper. Runqueuer can only directly be killed by SIGKILL.

- If the remote job is a LS-DYNA job, then a D3KIL file is created on the remote side. If LS-DYNA fails to exit after about 2 minutes, then the SIGTERM is used to kill LS-DYNA. If that also fails, then the SIGKILL method is used to kill LS-DYNA. If that fails, then the runqueuer/wrapper programs exit, and a queuing system job delete command is issued. The job is then abandoned.
- If the remote job is a non-LS-DYNA job, then the SIGINT (or SIGTERM) is first used. This is done approximately every 30 seconds. If that fails, then the SIGTERM method is used. If that also fails, then the SIGKILL method is used. If that fails, then the runqueuer/wrapper programs exit, and a queuing system job delete command is used. The job is then abandoned.

I.1.3 Case 3 (Jobs are run using the BLACK BOX queuing option):

This option was created to deal with an unknown queuing system, and to provide a general mechanism for progress monitoring using scripts created by the user. Jobs are submitted using a user-supplied submit script to run jobs on remote nodes.

If you try to kill a job managed using this option, then a job id/name must be known, because the only way to kill the job is using a user-supplied LsoptJobDel script. Queuing scripts can hang for a time before finish. Therefore, we wait for 2 minutes to find the job name/id in the script output. After that we abandon the job.

A script named LsoptJobCheck runs periodically for each job. A job for which this script is running is not killed or abandoned until this script completes. This is done to avoid <defunct> processes.

I.2 Killing Jobs using LS-OPT, LS-OPTui, and Scheduler control command.

This is a program supplied with LS-OPT for the purpose of killing jobs managed by the **lscheduler6** process. Jobs may be killed in bulk or killed individually using this program. The **lscheduler6** program resides in the LS-OPT installation directory, along with the other executables.

I.2.1 Scheduler control command

- The scheduler command line utility is provided as an option of the **lscheduler6** executable. The LS-OPTui calls this utility to kill any jobs. Activate the control part of lscheduler6 by typing:

```
lscheduler6 ctrl
```

General syntax of the control command can be shown by executing:

```
lscheduler6.exe ctrl --help
```

A general overview of each command and their use can be shown by executing:

```
lscheduler6.exe ctrl help
```

Notice that in the line above, the “help” is interpreted as a command for the control.

By default the control command will search for and connect to a locally running scheduler in the same folder as it is executed. It is also possible to connect to a specific scheduler by providing host (-h,--host) and/or port (-p,--port) arguments.

The control command fails if no running scheduler could be found and connected to.

I.2.2 Killing All Jobs in Bulk

Jobs can be killed in bulk from

- LS-OPT using the sense switch **sw1.**
- LS-OPTui Run Panel.
- Using the scheduler command line utility.
- The command for killing the all jobs in bulk using the scheduler control command is:

```
lscheduler6 ctrl stop
```

This command will kill any running jobs, flag them for a restart and stop the running scheduler. “killall” is an alias for this command.

I.2.3 Kill One Job

A single job may be killed using LS-OPTui, or using the scheduler command line utility. The basic kill job option requires a **jobid** which is the subdirectory name of the job (job home-folder). In LS-

OPT the subdirectory name of the job is “stagename/iter.exp”, where “stagename” is the stage of the job, “iter” is the current iteration and “exp” is the current experiment.

The basic command is:

```
lscheduler6 ctrl kill jobid
```

Multiple jobs can be killed by specifying several jobids separated by comma:

```
lscheduler6 ctrl kill jobid,jobid2,jobid3
```

The executable will output whether or not the command has been successfully accepted by the scheduler but does not wait for the job to actually be killed.

I.3 Kill Level

Whenever a job is killed using the bulk kill option, or the scheduler kill command described above, the kill operation proceeds in levels or stages. For LS-DYNA there are three stages. For somes jobs, there are only two.

I.3.1 LS-DYNA Jobs (non-queued)

1. A D3KIL file is created in the working directory of the LS-DYNA job.
2. A SIGTERM signal is sent under Linux, or a Ctrl+Break under Windows.
3. A SIGKILL signal (kill -9) is sent under Linux, or is forcibly terminated under Windows, using the same force-kill mechanism as the Windows Task Manager.

I.3.2 LS-DYNA Jobs (queued with runqueuer/wrapper)

1. If the user-defined script failed to return a job name or job id, then a request is made to the runqueuer program to exit. The job is then abandoned.
2. If the user-defined script has returned a job name or job id, but the remote side has not yet connected back to the local side, then the queuer-specific job delete command is issued, and the runqueuer is instructed to exit. The job is then abandoned.
3. If the remote wrapper program has connected back to the local runqueuer, then a D3KIL is created by the remote wrapper, and we wait for LS-DYNA to exit.
4. This level pertains only to the case where the remote wrapper has connected, but the D3KIL file has failed. The wrapper program is now instructed to send a SIGTERM signal to LS-DYNA.
5. This level pertains only to the case where the remote wrapper has connected, but the D3KIL and/or the SIGTERM signal have failed. The wrapper program is now instructed to send a SIGKILL signal to LS-DYNA.

I.3.3 Other Jobs (non-queued)

1. A SIGINT (or SIGTERM) signal is sent under Linux, or a Ctrl+C (or a Ctrl+Break) is delivered under Windows.

2. A SIGTERM signal is sent under Linux, or a Ctrl+Break under Wndows.
3. A SIGKILL signal (kill -9) is sent under Linux, or is forcibly terminated under Windows, using the same force-kill mechanism as the Windows Task Manager.

I.3.4 Other Jobs (queued with runqueuer/wrapper)

1. If the user-defined script failed to return a job name or job id, then a request is made to the runqueuer program to exit. The job is then abandoned.
2. If the user-defined script has returned a job name or job id, but the remote side has not yet connected back to the local side, then the queuer-specific job delete command is issued, and the runqueuer is instructed to exit. The job is then abandoned.
3. If the remote wrapper program has connected back to the local runqueuer, then the wrapper is instructed to deliver a SIGINT (or SIGTERM) signal to the process.
4. This level pertains only to the case where the remote wrapper has connected, but the signal has failed to kill the process. The wrapper program is now instructed to send a SIGTERM signal to the process.
5. This level pertains only to the case where the remote wrapper has connected, but the SIGINT and/or SIGTERM signals have failed to kill the process. The wrapper program is now instructed to send a SIGKILL signal to LS-DYNA.

I.3.5 All BLACKBOX Queued Jobs

1. If the BLACKBOX queuing script has been executed, but no valid job id or job name has yet been detected, then we queue a kill event until the job id or job name can be found, and any running queuing script or LsoptJobCheck script has finished. We then execute LsoptJobDel to kill the job. If this condition has persisted for 120 seconds since starting the queuing script, then the job is abandoned.
2. If the BLACKBOX queuing script has returned a valid job id or job name, but the queuing script or LsoptJobCheck is currently running, then we queue a kill event until the script in question has finished. We then execute LsoptJobDel to kill the job.
3. If the BLACKBOX queuing script has returned a valid job id or job name and has completed, and if there is no queuing script or LsoptJobCheck in progress, then we kill the job using the user-supplied LsoptJobDel script. The job is then abandoned.
4. If a queuing script has not yet reported a valid job id or job name, then the job is abandoned.
5. If an LsoptJobCheck script is stalled, we send a SIGKILL to the script, and the job is abandoned.
6. The job has been abandoned by this point.

I.4 Increasing the Kill Level

The bulk kill may spend up to 2 minutes in the first stage (level 0) before proceeding to level 1. It then executes level 1 kill operations and waits only 30 seconds before proceeding to the final “sure-kill" level 2 kill operations. The kill will remain in the final stage only for a short time before abandoning the job entirely, possibly leaving behind a job that refuses to exit. Some jobs are abandoned at level 1, depending on the type of job and the queuing options. Jobs abandoned at level 1 do not usually result in lost resources.

It not generally advisable to elevate the kill level for a particular job, but this is an option which can be reasonably invoked by the user. For example,

```
lscheduler6 ctrl kill jobid 1
```

will cause the kill routines to skip level 0 kill level and start directly at level 1 operations.

```
lscheduler6 ctrl kill jobid 2
```

will cause the kill routines to go directly to the final level kill operations.

Warnings

- Once a job kill operation is started, it cannot be canceled.
- The kill level 2 can result in lost kernel resources and incorrectly closed files. Linux systems may leave behind a <defunct> job, and files may be incompletely written. Windows systems may lose resources because DLLs are not properly detached and may even destabilize the system; files may also be corrupted or incompletely written.
- LS-DYNA may be unable to check for the presence of a D3KIL file for a considerable period of time. This usually occurs with implicit jobs where long matrix calculations are not interrupted for reasons of efficiency.

I.5 Termination Status for Killed Jobs

This is a tricky issue, where the defaults are usually appropriate, but not always. For that reason, you may wish to set the final job status to ERROR, ABORTED, KILLED using the scheduler command line utility.

You can specify the final job final status as follows:

```
lscheduler6 ctrl kill jobid[,jobid2...] [level] error
lscheduler6 ctrl kill jobid[,jobid2...] [level] aborted
lscheduler6 ctrl kill jobid[,jobid2...] [level] killed
```

Commands marked with ERROR will not be restarted using LS-OPT, and results from these commands are ignored. The finished file will contain the string

```
E r r o r   t e r m i n a t i o n
```

Commands marked with ABORTED will be restarted according to the retry rules defined in LS-OPT. If the process has reached its maximum number of allowed retries (which may be 0) then the process is flagged with a final status of ERROR. Otherwise, the command is retried after a suitable wait period. A finished file is created only in the latter ERROR case and, in that case, will contain

```
E r r o r   t e r m i n a t i o n
```

The KILLED option is used by the bulk kill option so that LS-OPT will retry the job upon restart; however, this job may not restart if LS-OPT has beyond the point where the results of this job are needed. The finished file will contain the string

Forced termination

If a bulk kill is in effect when you kill the job using scheduler control command, then the default final job status is KILLED, which is the same as all other jobs killed during the bulk kill. However, if a bulk kill is not in effect when you kill the job using scheduler control command, then the default final job status is ERROR. The assumption is that any job killed during a bulk kill should be flagged the same as all other jobs; the intent of the user is interpreted only as an action to speed along the bulk kill by targeting certain jobs which are slow to exit. However, if just one job is killed outside of a bulk kill, then the assumption is that the job has failed in some way, and should be flagged with ERROR; that way the job will not be restarted.

I.6 Flagging a Job for Restart

If you kill a job using scheduler control command, then you can also flag the job for restart:

```
lscheduler6 ctrl restart jobid[,jobid2...] [wait_time_seconds]
```

If you do not specify a wait time, then a default value of 0 is used, thereby instructing the scheduler process to reschedule the job as soon as there are resources to do so. The status returned by this command is the same as for the corresponding kill command without the restart option.

This option is designed for cases where a temporary problem has caused a job to hang or misbehave, and you want to kill and restart the job after waiting long enough to correct the problem For example, a queuing system may have failed, and the job was lost; so the submit script failed to return a job id/name. Or perhaps the queuing system discarded the job for some reason, and LS-OPT has no way of knowing that the job cannot complete.

Another designed use of this command is in situations where a job has ERROR terminated because of some resource problem, and you want to schedule the job to restart before the scheduler terminates, so that LSOPT can use the corrected results. To flag a completed job for restart, issue the scheduler control command with the `restart` command described above.

The scheduler process will restart the process as soon after the specified time wait as resources are available to start the job.

This command does not function for running jobs, unless a kill is already scheduled, in which case the restart option and restart time field are updated. If a kill has not yet been scheduled for the job, then the command will report an error.

You can immediately kill a job and put set it to restart by issuing the kill command:

```
lscheduler6 ctrl kill jobid [level] restart [wait_time_seconds]
```

This will kill the job and set the job to restart using the optional wait time.

Remarks on Restarting Jobs

1. Restart events should be considered transient, persisting only during the time that scheduler is running. If you issue a bulk kill of all jobs, then any pending restart events are discarded. The affected jobs are not automatically restarted at a later time.
2. A job which has never been started cannot be flagged for restart.

3. Jobs which are running cannot be flagged for restart. Use the kill command and set the final status to *restart* or issue a separate restart command after the kill is successful.

Appendix J: Document Type Definition (DTD)

<lsoptproject> Root element

root element

<lsoptproject>'s attributes

Name	Values	Default	Description
version	CDATA	Required	Version number for this lsopt project.

Element's model:

(encryption?, head?, globals?, distributions?, filehistories?, multifilehistories?, transformations?, variables?, varcorrelations?, samplings?, evalmeta?, composites?, objectives?, constraints?, schedulerconfig?, resources?, classifiers?, task)

<encryption> Child of lsoptproject

Holder of encrypted data that replaces the current document.

<head> Child of lsoptproject

project header - metadata block

Element's model:

(title?, meta*)

<title> Child of head

project title

<meta/> Child of head

meta (metadata name/content-pair) (modelled after its HTML equivalent)

<meta>'s attributes

Name	Values	Default	Description
content	CDATA	Required	Value for entry

name	Match the NMTOKEN rules.	Required Name for this metadata entry (such as "author")

This element is always empty.

<globals/> Child of lsoptproject

Global parameter definitions

<globals>'s attributes

Name	Values	Default	Description
historysize	CDATA	100,000	Number of time points for new histories

This element is always empty.

<distributions> Child of lsoptproject

Container element for the defined distributions

Element's model:

(distribution*)

<distribution/> Child of distributions

Distribution declaration

<distribution>'s attributes

Name	Values	Default	Description
a	CDATA	Implied	'a' value for beta, triangular, gumbel, frechet
b	CDATA	Implied	'b' value for weibull, beta, erlang, rayleigh, frechet triangular, gumbel
c	CDATA	Implied	'c' value for weibull, erlang, triangular, frechet
filename	CDATA	Implied	File name for user_pdf, user_cdf
lambda	CDATA	Implied	lambda value for poisson
lower	CDATA	Implied	Lower bound for uniform, truncated_normal
mean	CDATA	Implied	Mean value for normal, truncated_normal, lognormal
n	CDATA	Implied	n value for binomial

name	Match the ID rules.	Required	Distribution name, prefixed with "d_" to be unique in the global namespace
nu	CDATA	Implied	nu value for beta
omega	CDATA	Implied	omega value for beta
p	CDATA	Implied	p value for binomial, geometric
scale	CDATA	Implied	Scale value for exponential
stddev	CDATA	Implied	Standard deviation for normal, truncated_normal, lognormal
type	uniform, normal, truncated_normal, lognormal, exponential, weibull, beta, binomial, geometric, poisson, erlang, rayleigh, triangular, gumbel, frechet, user_pdf, user_cdf, user_normaldata, user_weibulldara	Required	Distribution type: uniform, normal, truncated_normal, lognormal, exponential, weibull, beta, binomial, geometric, poisson, erlang, rayleigh, triangular, gumbel, frechet, user_pdf, user_cdf, user_normaldata, user_weibulldara
upper	CDATA	Implied	Upper bound for uniform, truncated_normal

This element is always empty.

<filehistories> Child of lsoptproject

Container element for the defined filehistories

Element's model:

(filehistory*)

<filehistory> Child of filehistories

Declaration of history on file.

<filehistory>'s attributes

Name	Values	Default	Description
genexfile	CDATA	-	Name of genex file if type is genex

name	Match the ID rules.	Required	Name for this file history, prefixed with "x_" to be unique in the global namespace
type	file, genex	file	Interface type
x_component	CDATA	-	x component used for gom interface
y_component	CDATA	-	y component used for gom interface
xent	CDATA	-	x entity of history if type is genex
yent	CDATA	-	y entity of history if type is genex

Element's model:

(file*)

<file> Child of multifilehistory, filehistory

The name of the file is given as the contents of the tag

<multifilehistories> Child of lsoptproject

Container element for the defined multipoint filehistories

Element's model:

(multifilehistory*)

<multifilehistory> Child of multifilehistories

Declaration of multihistory on file

<multifilehistory>'s attributes

Name	Values	Default	Description
file	timestep, point	timestep	Input files are a file per timestep or a file per point if type is genex
genexfile	CDATA	-	Name of genex file if type is genex

regex	CDATA	-	Genex input file template
name	Match the ID rules.	Required	Name for this file history, prefixed with "x_" to be unique in the global namespace
type	file, gom, genex	gom	Interface type
x_component	time, stage, ad0, ad1, ad2, ad3, ad4, ad5, ad6, ad7, ad8, ad9, index_x, index_y, coordundef_x, coordundef_y, coordundef_z, coorddef_x, coorddef_y, coorddef_z, displ_x, displ_y, displ_z, strain_x, strain_y, strain_major, strain_minor, thicknessred	-	x component used for gom interface
xent	CDATA	-	x entity of history if type is genex
y_component	time, stage, ad0, ad1, ad2, ad3, ad4, ad5, ad6, ad7, ad8, ad9, index_x, index_y, coordundef_x, coordundef_y, coordundef_z, coorddef_x, coorddef_y, coorddef_z, displ_x, displ_y, displ_z, strain_x, strain_y, strain_major, strain_minor, thicknessred	-	y component used for gom interface
yent	CDATA	-	y entity of history if type is genex

Element's model:

(file*)

<transformations> Child of lsoptproject

Container element for the defined transformations used for multihistory and multiresponse extraction.

Element's model:

(transformation*)

<transformation> Child of transformations

Container element for the defined alignment points

<transformation>'s attributes

Name	Values	Default	Description
alignSrcType	id, coord	id	Specify gom point or coordinates of alignment points
alignTgtType	id, coord	id	specify Node ID or coordinates of alignment points
name	Match the ID rues	-	Name for transformation, prefixed with "t_" to be unique in the global namespace
scaling	auto no user	no	Use automatic scaling, no scaling, or scaling with scaling factor defined by user
scale	CDATA	1.0	Scale factor

Element's model:

(point*)

<point> Child of transformation

Pair of test and simulation point to be aligned

<transformation>'s attributes

Name	Values	Default	Description
gom	CDATA	-	gom/ARAMIS point id, used if alignSrcType id
node	CDATA	-	Node ID from simulation model, used if alignTgtType id

sim_x	CDATA	-	x coordinate, used if alignTgtType coord
sim_y	CDATA	-	y coordinate, used if alignTgtType coord
sim_z	CDATA	-	z coordinate, used if alignTgtType coord
test_x	CDATA	-	x coordinate, used if alignSrcType coord
test_y	CDATA	-	y coordinate, used if alignSrcType coord
test_z	CDATA	-	z coordinate, used if alignSrcType coord

<variables> Child of lsoptproject

Container element for the defined variables

Element's model:

(variable*)

<variable> Child of variables

A single element is used for all types of variables. This is so that a variable can change type (typically back and forth from constant) without losing data, since a constant can have min/max/range; it's just unused. Element PCDATA is the definition for dependents and user defined.

<variables>'s attributes

Name	Values	Default	Description
dist	Match the IDREF rules.		Distribution for noise variables.
max	CDATA		Maximum value for continuous variables.
maxref	Match the IDREF rules	Implied	Reference max. Reference to a variable of input type
min	CDATA		Minimum value for continuous variables.
minref	Match the IDREF rules	Implied	Reference min. Reference ti a variable of input type

name	Match the ID rules.	Required	Variable name, prefixed with "x_" to be unique in the global namespace
range	CDATA		Initital range for variables
replacementlink	Match the IDREF rules	Implied	Name of the linked response. If type = constant, value will be replaced by respective response value
saddle	minimize, maximize	minimize	Saddle direction
sampling	continuous, discrete	continuous	contiuous/discrete sampling for discrete vars
type	continuous, discrete, noise, constant, sconstant, dependent, userdefined, string, iconstant, istring, responsevariable	continuous	Variable type.
valref	Match the IDREF rules	Implied	Reference value. Reference to a variable of input type
value			Starting value for continuous/discrete. Constant value for constants.

Element's model:

(#PCDATA | dvalue)*

<dvalue> Child of variable

Single value for discrete variable

<varcorrelations> Child of lsoptproject

Container element for the varcorrelation

Element's model:

(varcorrelation*)

<varcorrelation> Child of samplings

The correlation between variables can be specified. This correlation will be considered in Monte Carlo simulation (including metamodel based simulations) as well as in reliability based design optimization. Only correlation between normally distributed variables is allowed.

<varcorrelation>'s attributes

Name	Values	Default	Description
var1	Match the ID rules.	Required	Variable name
var2	Match the ID rules.	Required	Variable name
corr	CDATA	Required	Value of the correlation between the variables var1 and var2.

<samplings> Child of lsoptproject

Container element for the different samplings

Element's model:

(sampling*)

<sampling> Child of samplings

Defines a single sampling design

<sampling>'s attributes

Name	Values	Default	Description
name	Match the ID rules.	Required	Design id, prefixed with "e_" (for experiment) to be unique in the global namespace.
ignore	yes, no	no	Set to yes and this sampling will be ignored

Element's model:

(design?, edsdconstraints?,metamodel?, alternate?, histmetamodel?, myvars?, stages?, importresults?, importdesignfunctions?, checkpoints?, cmp_metamodels?)

<design> Child of sampling, alternate

Specifies the design for this sampling of points

<design>'s attributes

Name	Values	Default	Description
filename	CDATA	Implied	Filename for userdefined
filetype	csv, original	Implied	set to "csv" or "original" for userdefined

firstaugiter	CDATA	Implied	If set, iterations before the one given, that have existing experimental points will not be augmented.
perturbation	CDATA	0.001	Perturbation relative to design space for numerical sensitivity
points	CDATA	Implied	Number of points per iteration for dopt, monte_carlo, latin_hypercube*, maximin_distance
ppv	2, 3, 4, 5, 6, 7, 8, 9, 10, 11	2	Points per variable for full_factorial
replicate	CDATA	1	Number of replicate evaluations of the same design using stochstic fields
seed	CDATA	0	Optional random number seed for methods that use randomness. All point selection schemes are repeatable, but a seed can be provided to create different sets of random points. The feature is particularly useful for Monte Carlo or Latin Hypercube point selections which both directly use random numbers. Because D-Optimal and Space Filling designs also use random numbers, albeit less directly, they may only show small differences due to the occurrence of local minima in the respective optimization procedures.
stype	numerical, analytical	numerical	Subtype for sensitivity: numerical, analytical
subtype	latin_hypercube_central_point		Space Filling 1
	latin_hypercube_generalized		Space Filling 2
	maximin_distance		Space Filling (Algorithm 5)

	maximin_lhd_permute		Space Filling 3
	maximin_lhd_subinterval		Space Filling 4
type	one,	Required	Single Point
	dopt,		D-Optimal
	koshal_linear,		Linear Koshal
	koshal_quad,		Quadratic Koshal
	composite,		Central Composite
	userdefined,		User defined sampling
	plan,		Plan
	random,		Random placement
	latin_hypercube,		Latin Hypercube
	monte_carlo,		Monte Carlo, Space Filling 0
	maximin_distance,		Space Filling
	full_factorial,		Full Factorial Designs
	piercing,		Space filling of Pareto Frontier
	domainpiercing,		Space Filling of Pareto region, not used
	sensitivity,		Sensitivity
	oarray		Orthogonal Array
update	on, off	Implied	If set, causes space-filling method to consider points from previous iterations. Implies metamodel update.

Element's model:

(basis?, levels?)

<basis/> Child of design

Custom basis for D-Optimal design

<basis>'s attributes

Name	Values	Default	Description
points	CDATA	Implied	Number of points per iteration for latin_hypercube, maximin_distance

ppv	2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Implied	Points per variable for full_factorial
type	full_factorial, latin_hypercube, maximin_distance	Implied	Full Factorial Designs Latin Hypercube Space filling

This element is always empty.

<levels> Child of design

Custom levels for Orthogonal Array design. Container element for the different level

<levels>' attributes

Name	Values	Default	Description
interactions	yes, no	no	If set, causes orthogonal array method to consider interactions between variables.
autoOAchoice	yes, no	yes	If not set, the Orthogonal Array is selected by the user.
tablename	CDATA	Implied	If autoOAchoice not set, name of the table selected by the user.

Element's model:

(level*,oainteraction*)

<level> Child of levels

<level>'s attributes

Name	Values	Default	Description
name	IDREF	Required	Name of the variable
numValues	CDATA	Implied	Level number: number of authorized value for this variable

This element is always empty.

<oainteraction> Child of levels

<oainteraction>'s attributes

Name	Values	Default	Description
varname1	IDREF	Required	Name of the 1st variable
varname2	IDREF	Required	Name of the 2nd variable

This element is always empty.

<edsdconstraints> Child of sampling

Element's model:

(edsdconstraint?)

<edsdconstraint> Child of edsdconstraints

EDSD sampling constraints

<edsdconstraint>'s attributes

Name	Values	Default	Description
src	IDREF	Required	Reference to a classifier to use as constraint
marginlower	CDATA	Implied	Classifier margin lower bound
marginupper	CDATA	Implied	Classifier margin upper bound
firstsampleadaptive	CDATA	1	First constrained sample of iteration
lastsampleadaptive	CDATA	Implied	Last constrained sample of iteration
samplegap	CDATA	1	Gap between constrained samples (how many samples ignore the constraint)
marginrandom	yes, no	yes	Flag to specify if the sampling margins are randomized

This element is always empty.

<metamodel> Child of sampling

Controls response metamodel created for this sampling

<metamodel>'s attributes

Name	Values	Default	Description
order	linear, interaction, quadratic, elliptic, spherical	linear	linear, interaction, quadratic, elliptic, spherical (for polynomial)
overwrite	yes, no	no	
type	polynomial, sensitivity, ffnn, rbf, kriging, svr, userdefined	Required	polynomial, sensitivity, ffnn, rbf, kriging, svr or userdefined
update	on, off	Implied	When on, the metamodel will re-use points from previous iterations.

Element's model:

(ffnnopts?, rbfopts?, krigingopts?, svropts?, usermmopts?)

<cmp_metamodels/> Child of sampling

Controls response comparison metamodels created for this sampling

Element's model:

(cmp_metamodel*)

<cmp_metamodel> Child of cmp_metamodels

Controls response comparison metamodel created for this sampling

<cmp_metamodel>'s attributes

Name	Values	Default	Description
active	yes, no	yes	
name	Match the ID rules	Required	Name of comparison metamodel
order	linear, interaction, quadratic, elliptic, spherical	linear	linear, interaction, quadratic, elliptic, spherical (for polynomial)
overwrite	yes, no	no	

type	polynomial, sensitivity, ffnn, rbf, kriging, svr, userdefined	Required	polynomial, sensitivity, ffnn, rbf, kriging, svr or userdefined
update	on, off	Implied	When on, the metamodel will re-use points from previous iterations.

Element's model:

(ffnnopts?, rbfopts?, krigingopts?, svropts?, usermmopts?)

<histmetamodel/> Child of sampling

Controls history metamodels created for this sampling

<histmetamodel>'s attributes

Name	Values	Default	Description
type	linear, quadratic, rbf	Implied	Type of metamodel to use for histories. linear, quadratic or radial basis functions.

This element is always empty.

<ffnnopts/> Child of metamodel

Special options controlling Feed Forward Neural Networks

<ffnnopts>'s attributes

Name	Values	Default	Description
averagetype	mean, median	mean	Averaging function to use, mean or median
command	CDATA	Implied	Command for ffbuilder
discard	CDATA	Implied	Discard these many committee members with the lowest mean squared fitting error and the same number of committee members with the highest MSE.
hiddennodes	CDATA	0,1,2,3,4,5	Space, dash or comma separated list of hidden nodes in ensemble
layers	CDATA	3	Number of layers
members	CDATA	9	Number of Committee members

optimize	gcv, gcv_ratio, rmserror	gcv	Topology Selection Criterion: Leave-one-out, GCV-ratio, Noise variance
regressionalg	levenberg, bfgs, rprop	levenberg	Gradient algorithm: Levenberg-Marquardt, Broyden-Fletcher-Goldfarb-Shanno, Resilient backpropagation
rmserror	CDATA	-1.0	Threshold of RMS training error: The sorting algorithm will pick the first neural net which falls below the specified threshold starting with 0 hidden nodes (linear). That means that, for a truly linear function, the sorting process will be terminated after 0, resulting in a dramatic saving of computational effort.
seed	CDATA	0	Random number seed for generation of a unique set of neural networks.
transfer	linear, sigmoid, gauss, hmq	sigmoid	Transfer function of intermediate layers: Linear, Sigmoid, Gauss, Hardy's Multi Quadrics
Transfer_out	linear, sigmoid, gauss, hmq	linear	Transfer function of output layer: Linear, Sigmoid, Gauss, Hardy's Multi Quadrics
scheduling	yes, no	no	Use scheduling options that are defined for main metamodel

Element's model:

(scheduling?)

<rbfopts/> Child of metamodel

Special options controlling Radial Basis Function networks

<rbfopts>'s attributes

Name	Values	Default	Description
optimize	gcv, gcv_ratio, rmserror	gcv	Topology Selection Criterion: Leave-one-out, GCV-ratio, Noise variance

rmserror	CDATA	-1.0	Threshold of RMS training error
transfer	linear, sigmoid, gauss, hmq	hmq	Transfer function: Linear, Sigmoid, Gauss, Hardy's Multi Quadrics

This element is always empty.

<krigingopts/> Child of metamodel

Special options controlling creation of Kriging metamodels

<krigingopts>'s attributes

Name	Values	Default	Description
correlation	gauss, exponential	gauss	Correlation function
selectone	on, off	off	Set to one to use fixed theta for all responses
trendmodel	constant, linear, quadratic	linear	Trend model

This element is always empty.

<svropts/> Child of metamodel

Special options controlling creation of Support Vector Regression metamodels

<svropts>'s attributes

Name	Values	Default	Description
kernel	gauss, polynomial	gauss	Kernel type

This element is always empty.

<usermmopts> Child of metamodel

Special options controlling user metamodel. Contents of this element is the optional user-defined metamodel parameters, separated by whitespace. This allows the user to send numeric parameters to the user defined metamodel. It is up to the metamodel to specify which, if any, parameters it requires for operation.

<usermmopts>'s attributes

Name	Values	Default	Description
command	CDATA	Implied	Optional string command passed to user code. Allows the user to send one string parameter to the user-defined metamodel, that may be used in any way by the metamodel.
name	CDATA	Implied	Name for user metamodel
path	CDATA	Implied	Optional path to look for user binaries. This has to be specified if the metamodel binaries are not placed in the same directory than the .lsopt file that refers to it, but in a central repository.

<alternate> Child of sampling

When this element is present in a sampling, linear metamodels will be used instead of the ones specified in the metamodel tag, but only up to a certain iteration. An alternate sampling design to be used for these iterations may also be specified.

<alternate>'s attributes

Name	Values	Default	Description
lastiteration	CDATA	1	The number of the last iteration to use linear metamodel. Default is "1".

Element's model:

(design?)

<stages> Child of sampling

Container element for all stages in a design

Element's model:

(stage*)

<stage> Child of stages

Stage definition

<stage>'s attributes

Name	Values	Default	Description
extractor	internal, external	external	Controls whether this stage will extract using the separate binary
ignore	yes, no	no	Set to yes and this stage will be ignored.
name	Match the ID rules.	Required	Stage name, prefixed with "s_" to be unique in the global namespace
type	Match the NMTOKEN rules.	Required	Stage type, dyna960, own ...

Element's model:

(guidata?, runin?, dependson*, fileops?, command?, dbfile?, inputfile?, outputfile?, appendfile?,configfile?, resource?, envvars?, extrainpfiles?, dynaoptions?, exceloptions?,scheduling?, myvars?, histories?, multihistories?, responses?)

<guidata> Child of stage

Metadata used for graphical representation of the problem. Not relevant to the LS-OPT engine

Element's model:

(position?)

<position/> Child of guidata

Position of the entity in 2D Cartesian space

<position>'s attributes

Name	Values	Default	Description
x	CDATA	Required	X position, floating point value
y	CDATA	Required	Y position, floating point value

This element is always empty.

<runin/> Child of stage

By default, stages are run in experiment directories beneath a directory based on the stage name. By specifying this element, the stage can set to run in another directory.

<runin>'s attributes

Name	Values	Default	Descriptions

stage	Match the IDREF rules.	Required	The name of the stage to share directory with, including s_ prefix
ignore	yes, no	no	Set to yes and this stage is run in it's own directory

This element is always empty.

<dependson> Child of stage

Each dependson element specifies that this stage depends on another stage.

<dependson>'s attributes

Name	Values	Default	Description
stage	Match the IDREF rules.	Required	Stage name, including s_ prefix
ignore	yes, no	no	Set to yes and this stage will depend on the upstream stage(s) of *stage*

Element's model:

(filetransfer)*

<filetransfer/> Child of dependson

Each filetransfer element specifies transfer of a file or directory from the stage which the parent dependson element refers to.

<filetransfer>'s attributes

Name	Values	Default	Description
dstfilename	CDATA	Implied	Name of destination file in this stage. If not specified, default is same as srcfile.
onerror	fail, warn, ignore	fail	What to do if file transfer fails. stop, warn or ignore.
op	copy, move, link, copyrecursive	Required	File transfer operation. copy, move, link or copyrecursive
srcfilename	CDATA	Required	Name of file in upstream stage

This element is always empty.

<fileops> Child of stage

Container element for all intra-stage file operations

Element's model:

(fileop)*

<fileop/> Child of fileops

Specifies a local file copy operation

<fileop>'s attributes

Name	Values	Default	Description
dstfilename	CDATA	Required	Destination file name of operation (or file to delete)
onerror	fail, warn, ignore	Fail	What to do if file operation fails. fail, warn or ignore.
op	copy, move, link, copyrecursive, delete	Required	File transfer operation. copy, move, link or copyrecursive or delete
sequence	before, after	Required	Specifies if the file operation is to be done "before" or "after" stage solver execution
srcfilename	CDATA	Implied	Source file name of operation. Not used for delete

This element is always empty.

<command> Child of stage

Specifies a command line

<command>'s attributes

Name	Values	Default	Description
addinputarg	on, off	on	Add input file argument to command line
defaultcmd	yes, no	no	Use default command instead of PCDATA when applicable
displaygraphics	on, off	off	Omit –nographics option, only valid for stage type LS-PREPOST

<dbfile> Child of stage

Database file

<inputfile> Child of gatheropts, extrainpfiles, stage

Specifies an input file

<inputfile>'s attributes

Name	Values	Default	Description
basefilename	CDATA	Implied	The name of the input file after variable substitution. The default varies with stage type. For LS-DYNA it is "DynaOpt.inp". This attribute is only valid in the stage context (eg. not in extrainpfiles)
skip	on, off	off	Skip this file during parsing

<appendfile> Child of stage

Specifies an appended file

<outputfile> Child of stage

Specifies an output file

<configfile> Child of stage

Configuration file

<dynaoptions/> Child of stage

LS-DYNA specific settings

<dynaoptions>'s attributes

Name	Values	Default	Description
checkoutput	on, off	on	Check for needed database cards
compressd3p	on, off	off	Enable d3plot compression
partextract	CDATA	Implied	Write the results for a set of parts, given in filename
xformref	CDATA	Implied	Transform results using reference nodes given in file

This element is always empty.

<exceloptions/> Child of stage

Excel specific settings

<exceloptions>'s attributes

Name	Values	Default	Description
copyfile	yes, no	yes	Copy file to job directory

file	CDATA	Required	Excel document

Element's model:

(excelinput*)

<excelinput/> Child of exceloptions

Excel output definition

<excelinput>'s attributes

Name	Values	Default	Description
cell	CDATA	Required	Target cell(s) or named range
dir	vertical, horizontal	vertical	Direction. Used for types history or userdef
ref	Match the IDREF rules	Implied	Reference to node. Used for types param or history
sheet	CDATA	Implied	Target worksheet
type	param, history, formula, userdef	param	Tape: param, history, formula or userdef

<envvars> Child of stage

Collection of environment variables

Element's model:

(envvar*)

<envvar> Child of envvars

A single environment variable declaration. The contents of the element is the variable value

<envvar>'s attributes

Name	Values	Default	Description
name	CDATA	Required	Name of environment variable

<extrainpfiles> Child of stage

Collection of extra input files

Element's model:

(inputfile*)

<scheduling> Child of ffnnopts, stage

Scheduling options

<scheduling>'s attributes

Name	Values	Default	Description
concurrent	CDATA	Implied	Number of concurrent jobs. If not set, all jobs are run in parallel.
proxy	none, lstcvm, ssh	none	Which proxy to use for job submission: none, lstcvm or ssh
ssh_config	CDATA	Implied	Specify a SSH configuration when proxy is aah. The value is a reference to a configuration name in the global LS-OPT configuration (guiconfig.xml)
ssh_clean	default, always, never	default	Default means remote folder is cleaned after successful transfer of output files
queuer	aqs, loadleveler, lsf, nqs, pbs, pbspro, slurm, sge, blackbox, simple, user	Implied	Enable the use of a queuing system. Valid values are: aqs, loadleveler, lsf, nqs, pbs, pbspro, slurm, sge, blackbox, simple and user. If not set, jobs are run locally.
retries	CDATA	9	Number of retries if submission fails
submittimeout	CDATA	60	Submission script timeout (seconds)
timeout	CDATA	720	A special case exists in which the LS-OPT job scheduler automatically generates an Abnormal signal. This is whenever the wrapper has not been executed for a specified timeout period. For this case a queuer timeout can be specified. The queuer timeout is the time it will wait for the wrapper to connect, otherwise it sets an abnormal termination status and writes an Abnormal signal to standard output. In this case the job will be resubmitted for the number of retries specified and using the queuing timeout for each retry.

Element's model:

(termcriteria*, recover*, resourceref*)

<termcriteria> Child of scheduling

User-defined termination criterial for the solver

Element's model:

(tcriterion*)

<tcriterion> Child of termcriteria

A specific user-defined croterion

<tcriterion>'s attributes

Name	Values	Default	Description
termination	normal, error, abnormal	Required	Type of termination for criteria. There can be only one tcriterion per type.
type	retcode, stdout, file_ex, file_line, constraint	Required	Type of criteria
delay	CDATA	0	Wait this time in seconds after criteria is met

Element's model:

(tc_retcode?, tc_file?, tc_line?)

<tc_retcode> Child of tcriterion

Specifies a termination criterion return code as the contents of the tag

This element is always empty.

<tc_file> Child of tcriterion

Specifies a termination criterion file as the contents of the tag

This element is always empty.

<tc_line> Child of tcriterion

Specifies a termination criterion line in file or stdout as the contents of the tag

This element is always empty.

<recover/> Child of scheduling

Causes files to be recovered from compute node. One of file or dynadb options must be specified.

<recover>'s attributes

Name	Values	Default	Description
dynadb	d3plot, d3eigv, d3hsp, binout, eigout	Implied	Enables recovery of a specific dyna database
file	CDATA	Implied	Enables recovery of a file or glob (wildcard) specification

This element is always empty.

<resourceref/> Child of scheduling

Specifies consumption of a single resource for every job of this stage

<resourceref>'s attributes

Name	Values	Default	Description
resource	CDATA	Required	Resource name
units	CDATA	Required	Number used per job

This element is always empty.

<myvars> Child of sampling, stage

Collection of non-global variables used by this stage. Global variables need not be listed.

Element's model:

(varref | sysref)*

<varref/> Child of myvars

A reference to a variable

<varref>'s attributes

Name	Values	Default	Description
src	Match the IDREF rules.	Required	Variable name, including "x_" prefix for this reference

This element is always empty.

<sysref/> Child of myvars

A reference to a system variable

<sysref>'s attributes

Name	Values	Default	Description
src	runid, iterid, lsprojpath	Required	System variable name

This element is always empty.

<histories> Child of stage

Collection of histories to extract

Element's model:

(history*)

<history> Child of histories

The textual contents of this element is the history definition string, or the expression, based on the type attribute

<history>'s attributes

Name	Values	Default	Description
DatabasePrefix	CDATA	Implied	LS-DYNA/Generic case to use for extraction
dynacase	CDATA	Implied	LS-DYNA case to use for extraction
name	Match the ID rules.	Required	Name for history, prefixed with "x_" to be unique in the global namespace
type	definition, expression, metapost, userpost, metlab	definition	definition for classic LS-OPT definition strings. expression for expressions, metapost for meta post-processor result and userpostpro for user defined post-processor result. More might be added in the future if we'd want full markup for histories
pca	yes, no	no	If set to yes, Principal Component Analysis is performed

Element's model:

(#PCDATA | definecurve)*

<definecurve> Child of history

DEFINE_CURVE definition of history

<definecurve>'s attributes

Name	Values	Default	Description
filename	CDATA	Required	Filename of curve
lcid	CDATA	Required	LCID of curve

This element is always empty.

<multihistories> Child of stage

Collection of multihistories to extract

Element's model:

(multihistory*)

<multihistory> Child of multihistories

The textual contents of this element is the multihistory definition string, or the expression, based on the type attribute

<history>'s attributes

Name	Values	Default	Description
DatabasePrefix	CDATA	Implied	LS-DYNA/Generic case to use for extraction
dynacase	CDATA	Implied	LS-DYNA case to use for extraction
name	Match the ID rules.	Required	Name for history, prefixed with "x_" to be unique in the global namespace
type	definition, expression	definition	definition for classic LS-OPT definition strings. expression for expressions
ascii	off, on	off	If on binary and ascii database is written

<responses> Child of stage

Collection of responses to extract

Element's model:

(response | cresponse)*

<response>	Child of responses

The textual contents of this element is the response definition string, or the expression, based on the type attribute

<response>'s attributes

Name	Values	Default	Description
DatabasePrefix	CDATA	Implied	LS-DYNA/Generic case to use for extraction
dumpformula	yes, no	no	Dump metamodel formula to file
dynacase	CDATA	Implied	LS-DYNA case to use for extraction
metamodel	on, off	on	Set to off to disable metamodel generation
name	Match the ID rules.	Required	Name for response, prefixed with "x_" to be unique in the global namespace
offset	CDATA	0	Offset of response
scale	CDATA	1	Scaling of response
type	definition, expression, matrix, metapost, userpost, import, matlab, multimeansqerr, multidiscretefrechet, multidynamictimewarpin, multitwlcs, |multipartialcurvemapping, multidtwp	definition	definition for classic LS-OPT definition strings. expression for expressions and matrix for matrix expressions.
computed	CDATA	Implied	Computed curve expression for curve mapping resposes. Note that this is an expression and not a reference to a history!
target	CDATA	Implied	Target curve expression for curve mapping responses. Note that this is an expression and not a reference to a history!
numpoints	CDATA	Implied	Number of points for curve mapping responses

<multiresponses> Child of stage

Collection of multiresponses to extract

Element's model:

(multiresponse*)

<multiresponse> Child of multiresponses

The textual contents of this element is the multiresponse definition string, or the expression, based on the type attribute

<response>'s attributes

Name	Values	Default	Description
name	Match the ID rules.	Required	Name for mutiresponse, prefixed with "x_" to be unique in the global namespace
scale	CDATA	1	Scaling of multiresponse
offset	CDATA	0	Offset of multiresponse
dumpformula	yes, no	no	Dump metamodel formula to file
ascii	off, on	off	If on binary and ascii database is written
dynacase	CDATA	Implied	LS-DYNA case to use for extraction
DatabasePrefix	CDATA	Implied	LS-DYNA/Generic case to use for extraction
type	definition, expression	definition	definition for classic LS-OPT definition strings. expression for expressions
metamodel	on, off	on	Set to off to disable metamodel generation
pca	yes, no	no	If set to yes, Principal Component Analysis is performed

<importresults> Child of sampling

If present, activates importing of user results from csv file given as element contents

<importdesignfunctions> Child of sampling

If present, activates importing of user design functions from xml file given as element contents

<checkpoints> Child of sampling

If present, activates checkpoints with csv file given as element contents

<evalmeta> Child of lsoptproject

If present, activates evaluation of metamodel at specific points from the csv file given as element contents

<classifiers> Child of lsoptproject

Collection of classifiers

Element's model:

(classifier)*

<classifier> Child of classifiers

<classifier>'s attributes

Name	Values	Default	Description
name	Match the ID rules	Required	Name of the classifier
systemtype	series, parallel	series	Specifies whether the component conditions are in series or parallel. If series, all the component conditions need to be met for feasibility. If parallel, satisfaction of any one or more component(s) results in a feasible design.
type	none, epssvm	epssvm	Type of classifier. System designator only (none) or support vector machine (epssvm). A "system designator only" classifier only evaluates the sample points, and does not predict the class of arbitrary designs.
ignore	yes, no	no	Set to yes and this classifier will be ignored

Element's model:

(initparam?, classificationcomponent*)

<classificationcomponent> Child of classifier

<classificationcomponent>'s attributes

Name	Values	Default	Description

name	Match the IDREF rules	Required	Classifier component name
labelmethod	threshold, kmeans	threshold	Classifier component condition type - threshold value or clustering
fail	highfail, lowfail	highfail	
lower	CDATA	Implied	Classifier component lower bound condition (in case of labelmethod "threshold")
upper	CDATA	Implied	Classifier component upper bound condition (in case of labelmethod "threshold")
ignore	yes, no	no	Set to yes and this component will be ignored

This element is always empty.

<initparam> Child of classifier

Element's model:

(svcopts)*

<svcopts> Child of initparam

<svcopts>'s attributes

Name	Values	Default	Description
epsilon	CDATA	1.0e-6	Support vector classification training tolerance
c	CDATA	1000.0	Support vector classification misclassification cost
svmparsel	initial, errorrate, balanced_errorrate, roc	errorrate	Support vector classification parameter selection algorithm
useinit	yes, no	yes	
screenDuplicate	yes, no	no	Screen duplicate points
includeFailed	yes, no	no	Include failed points as infeasible

Element's model:

(kernelparam)*

<kernelparam> Child of svcopts

<kernelparam>'s attributes

Name	Values	Default	Description
kernel	Gaussian, polynomial, exp	gaussian	Support vector classification kernel type

Element's model:

(kernelopt)*

<kernelopt> Child of kernelparam

Single value for kernelopt

<kernelopt>'s attributes

Name	Values	Default	Description
value	CDATA	Implied	Support vector classification kernel parameter(s). Degree for polynomial kernel. Spread for Gaussian kernel.

This element is always empty.

<composites> Child of lsoptproject

Collection of composite (global) responses and histories

Element's model:

(cresponse | chistory)*

<cresponse> Child of composites, responses

A composite response. The textual contents of this element is the cresponse def, when using "expression" type

<cresponse>'s attributes

Name	Values	Default	Description
computed	CDATA	Implied	Computed curve expression for meansqerr and curvemap. Note that this is an expression and not a reference to a history!
entity	Match the IDREF rules.	Implied	Entity for stddev-type composite
name	Match the ID rules.	Required	Name for composite, prefixed with "x_" to be unique in the global namespace
numpoints	CDATA	Implied	Number of points for meansqerr and curvemap
target	CDATA	Implied	Target curve expression for meansqerr and curvemap. Note that this is an expression and not a reference to a history!
type	weighted, targeted, standard_mse, expression, userdefined, stddev, meansqerr, curvemap	Required	Type of composite: weighted, targeted, standard_mse, expression, userdefined, stddev, meansqerr or curvemap
ignore	yes, no	no	Set to yes and this composite will be ignored

Element's model:

(#PCDATA | component)*

<component/> Child of cresponse

Composite response component

<component>'s attributes

Name	Values	Default	Description
divisor	CDATA	1	Component divisor, defaults to "1"
entity	Match the IDREF rules.	Required	Reference to variable or response, including "x_" prefix
multiplier	CDATA	1	Component multiplier, defaults to "1"
target	CDATA	Implied	Target value for targeted or standard_mse

ignore	yes, no	no	Set to yes and this component will be ignored

This element is always empty.

<chistory/> Child of composites

Composite/global history

<chistory>'s attributes

Name	Values	Default	Description
filename	CDATA	Implied	Name of file if type=file
name	Match the ID rules.	Required	Name for history, prefixed with "x_" to be unique in the global namespace
type	file	Required	Type of history: file

This element is always empty.

<objectives> Child of lsoptproject

Collection of objectives

<objectives>'s attributes

Name	Values	Default	Description
goal	maximize, minimize	minimize	Sets if we are to strive for minimizing or maximizing of objective function

Element's model:

(objective*)

<objective/> Child of objectives

A single objective

<objective>'s attributes

Name	Values	Default	Description
src	Match the IDREF rules.	Required	Reference to a response or composite to use as objective, including prefix
weight	CDATA	1	Weight for this objective

ignore	yes, no	no	Set to yes and this objective will be ignored

This element is always empty.

<constraints> Child of lsoptproject

Collection of constraints

<constraints>'s attributes

Name	Values	Default	Description
scaling	on, off	on	Set to "on", changes defaults for bound divisor

Element's model:

(constraint*)

<constraint> Child of constraints

Constraint for a single entity. Contains one or two bound elements of different types

<constraint>'s attributes

Name	Values	Default	Description
sampling	stay, move, move_start	stay	Controls how sampling should be affected by this constraint.
src	Match the IDREF rules.	Required	Reference to a response or composite to use as constraint
ignore	yes, no	no	Set to yes and this constraint will be ignored

Element's model:

((bound, bound?)?)

<bound/> Child of constraint, dynastats

A component of a constraint.

<bound>'s attributes

Name	Values	Default	Description
divisor	CDATA	Implied	Defaults to "1" if scaling is off, or **limit** if scaling is on. Not valid in dynastats.
limit	CDATA	Required	Numerical upper/lower limit

probability	CDATA	Implied	Probability of violating constraint, for RDBO. Not valid in dynastats.
strictness	CDATA	0.0	Strictness value, typically 0.0 or 1.0. Not valid in dynastats.
type	upper, lower	Required	upper or lower

This element is always empty.

<schedulerconfig/> Child of lsoptproject

Global scheduler options

<schedulerconfig>'s attributes

Name	Values	Default	Description
globalconcurrencylimit	CDATA	Implied	Maxmimum number of running jobs globally.
printinterval	CDATA	15	Report job status every n:th second
processfilename	CDATA	process.lsox	Name of process configuration file
resourcefilename	CDATA	resource.lsox	Name of resource file
schedconfigfilename	CDATA	schedconfig.lsox	Name of scheduler configuration file

This element is always empty.

<resources> Child of lsoptproject

Definitions of consumable resources

Element's model:

(resource*)

<resource/> Child of resources, stage

Single resource definition

<resource>'s attributes

Name	Values	Default	Description
limit	CDATA	Required	Number of consumable resource units available
name	CDATA	Required	Name of the consumable resource, no prefix

global	true, false	false

This element is always empty.

<task> Child of lsoptproject

Task control section

<task>'s attributes

Name	Values	Default	Description
method	none, metamodel_opt, mc, metamodel_mc, rbdo, metamodel_form, direct_opt, doestudy, exp_design, taguchi	Required	Main task. none is a legacy task that is used when converting com files that used the old tools, dynastats or repair tasks that are now specified in assignment. Task metamodel_form (First Order Reliability Method) will: o Construct the metamodels as prescribed by the user. If the metamodels already exists, then they won’t be recreated. o Conduct a FORM analysis for every constraint using the metamodels. The following are computed in a FORM analysis: o The most probable point (see Section 29.4.7) o The probabilities of exceeding the bounds on the constraint o The derivatives of the probability of exceeding the bound on the constraint with respect to the design variables The method requires very little information additionally to what is required for deterministic optimization. Specify the following: 6. Statistical distributions associated with the design variables 7. Probabilistic bounds on the constraints. Theoretical concerns are discussed in Section 29.4.8.

Element's model:

(optopts?, optalg?, runbaseline?, addmmnoise?, importmetamodel?, mcopts?, usegsa?, assignment?)

<optopts> Child of task

Options for metamodel_opt or rbdo

<optopts>'s attributes

Name	Values	Default	Description
pareto	on, off	off	Set to "on" to enable pareto optimal frontier generation
strategy	singlestage, sequential, domainreduction, ego	domainreduction	Strategy for optimization: singlestage, sequential, domainreduction, Efficient Global Optimization
verify	CDATA	1	Number of verification runs. (Clamped to [0,1] if MOO, not applicable for direct_opt method)

Element's model:

(seqopts?, srsmopts?)

<seqopts/> Child of optopts

Options for any sequential optimization strategy

<seqopts>'s attributes

Name	Values	Default	Description
accuracytol	CDATA	1.00	Response Accuracy Tolerance
designtol	CDATA	0.01	Design Change Tolerance
iterations	CDATA	Implied	Max number of iterations
objecttol	CDATA	0.01	Object Function Tolerance
toloperator	and, or	and	Tolerance operator: and, or.
pftol	CDATA	0.01	Tolerance for probability convergence
pftolrepeat	CDATA	1	Number of times the probability tolerance must be satisfied consecutively to stop

This element is always empty.

<srsmopts/> Child of optopts

Options for Domain Reduction strategy

<srsmopts>'s attributes

Name	Values	Default	Description
eta	CDATA	Implied	Proximity Zoom parameter
freezefromiter	CDATA	Implied	Freeze Range from iteration
gamma	CDATA	Implied	Oscillation Contraction parameter
psi	CDATA	Implied	Panning Contraction parameter
resetiter	CDATA	Implied	Reset to Initial Range on iteration

This element is always empty.

<optalg> Child of task

Optimization algorithm control

<optalg>'s attributes

Name	Values	Default	Description
hybrid	on, off	Implied	Hybrid mode: Switch to lfop after the basin of global optimum has been found
type	lfop, ga, asa, pso, diffev	Implied	Type of algorithm: Leaping Frog, Genetic, Adaptive Simulated Annealing, Particle Swarm Optimization, Differential Evolution

Element's model:

(gaopts?, asaopts?, psoopts?, lfopopts?, paretoopts?)

<gaopts/> Child of optalg

Options for Genetic Algoritm

<gaopts>'s attributes

Name	Values	Default	Description
binarymprob	CDATA	Implied	Mutation probability for binary values
binaryxop	sing_pt, uniform	sing_pt	Binary value crossover type: single point or uniform
binaryxoprob	CDATA	1.0	Crossover probability for binary values
blxalpha	CDATA	0.5	Alpha value for BLX

constrainth	deb_ech, penalty	deb_ech	Constraint Handling
generations	CDATA	Implied	Number of generations
mdistindex	CDATA	100	Mutation distribution index
moeatype	nsga_ii, spea_ii	nsga_ii	MOEA Type
nelite	CDATA	2	Number of elites
paramsel	tournament, roulette, sus	tournament	Selection operator: Tournament, Roulette or Stochastic Universal Sampling
popsize	CDATA	Implied	Population size
realmprob	CDATA	Implied	Mutation probability for real values
realxop	sbx, blx	sbx	Real value crossover type
realxoprob	CDATA	1.0	Crossover probability for real values
repeatlimit	CDATA	Implied	Max repeat optimum generations
restartint	CDATA	Implied	Restart interval
seed	CDATA	Implied	Random number seed
tournsize	CDATA	Implied	Tournament size
xdistindex	CDATA	10	Crossover distribution index (for SBX)

This element is always empty.

<asaopts/> Child of optalg

Options for Adaptive Simulated Annealing

<asaopts>'s attributes

Name	Values	Default	Description
annscale	CDATA	1000	Annealing Scale
funcparamratio	CDATA	1.0	Cost-Parameter Anneal Ratio
maxsim	CDATA	10000	Maximum Function Evaluations
tempratio	CDATA	1e-6	Tmin/Tmax ratio
tempupint	CDATA	1	Function evaluations per time step

This element is always empty.

<paretoopts/> Child of optalg

Options for Multi Objectives

<paretoopts>'s attributes

Name	Values	Default	Description
crthres	CDATA	Implied	Consolidation Ratio Change
dhvthres	CDATA	Implied	Normalized hypervolume change threshold
generationgap	CDATA	Implied	Generation Gap
hypervolume	on, off	on	Dominated hypervolume computation
termcriterion	maxfuncgen, fixedratio, ratiochange, volchange	maxfuncgen	Termination Criterion
utilityfraction	CDATA	Implied	Utility fraction cutof

This element is always empty.

<psoopts/> Child of optalg

Options for Particle Swarm Optimization

<psoopts>'s attributes

Name	Values	Default	Description
cogpressure	CDATA	2.0	Cognitive Pressure (C1)
constrainth	deb_ech, penalty	deb_ech	Constraint Handling
crazinessperiod	CDATA	2000	Craziness interval
initialinertia	CDATA	0.4	Initial inertia
limitnoimprovement	CDATA	Implied	FE Interval
maxevals	CDATA	25000	Maximum function evaluations
mutdistributionindex	CDATA	Implied	Mutation Distribution Index
numparticles	CDATA	Implied	Number of particles
pmut	CDATA	Implied	Mutation Probability
seed	CDATA	-1	Random seed
socpressure	CDATA	2.0	Social Pressure (C2)
stoppingcriterion	maxfunc, noimprovement	maxfunc	Termination Criterion

This element is always empty.

<lfopopts/> Child of optalg

Options for leaping frog algorithm

<lfopopts>'s attributes

Name	Values	Default	Description
nummultistart	CDATA	Implied	Number of multi start points
penaltymu	CDATA	100	Penalty Parameter mu
penaltymumax	CDATA	10000	Penalty Parameter mumax
print	CDATA	10	Print control number
steps	CDATA	1000	Maximum number of steps
stepsize	CDATA	1	Maximum step size
toleg	CDATA	1e-05	Convergence Criterion eg
tolx	CDATA	1e-08	Convergence Criterion xtol

This element is always empty.

<runbaseline/> Child of task

When this element is present, only the baseline will be run

This element is always empty.

<importmetamodel/> Child of task

When this element is present, metamodels will be imported on normal run

This element is always empty.

<addmmnoise/> Child of task

When this element is present, use Approximation Residuals

This element is always empty.

<mcopts/> Child of task

Monte Carlo specific options

<mcopts>'s attributes

Name	Values	Default	Description

createhistogram	CDATA	0	Number of bins in histogram data calculated for standard output display. Histogram data not printed if set to 0.
distbounds	on, off	off	Whether the bounds must be enforced for the probabilistic component of the variables
dsamethod	montecarlo, metamodel	montecarlo	Whether stochastic contribution is calculated using MC sampling or using quadratic approximations
dsaresolution	CDATA	10000	Number of points used in the Monte Carlo simulation for stochastic contributions
reliabilityresolution	CDATA	1e+06	Number of Monte Carlo samples to be analyzed (metamodel evaluations) for probabilistic calculation
subregsize	CDATA	2.0	Noise Variable Subregion Size (in stddevs), for metamodel based Monte Carlo

This element is always empty.

<usegsa/> Child of task

Presence of this element enables GSA calculation

<usegsa>'s attributes

Name	Values	Default	Description
overwrite	yes, no	No	Overwrite results of existing global calculations
points	CDATA	10000	Number of points for Integration

Element's model:

(gsasubregion*)

<gsasubregion> Child of usegsa

GSA Subregion

<gsasubregion>'s attributes

Name	Values	Default	Description
active	yes, no	yes	Whether the subregion should be calculated or not
name	Match the ID rules	Required	Name of subregion, prefixed with “b_” to be unique in the global namespace
overwrite	yes, no	no	Overwrite results of existing calculations

Element's model:

(gsasubbounds*)

<gsasubbounds/> Child of gsasubregion

GSA Subregion variable bounds

<gsasubbounds>'s attributes

Name	Values	Default	Description
lower	CDATA	Implied	Lower bounds
upper	CDATA	Implied	Upper bounds
var	Match the IDREF rules	Required	Reference to variable

This element is always empty.

<assignment> Child of task

Controls the assignment (formerly known as repair/tools). If this element is not present, it is equivalent of <assignment type="full"/>

<assignment>'s attributes

Name	Values	Default	Description
iteration	CDATA	Implied	Iteration number for repair assignments
sampling	Match the IDREF rules.	Implied	Name of the sampling for runjobs, rerunfailed and extract assignments. If absent, all stages for the given iteration are processed.
stage	Match the IDREF rules.	Implied	Name of the stage for runjobs, rerunfailed and extract assignments. If absent, all stages for the given iteration are processed.
type	full, dynastats, readpoints, addmmpoints, addmcpoints, runjobs, rungsa, rerunfailed, extract, import, importmm, buildmm, evalmm, optimize, gather, gatherext, clean, rerunverification, extractalliter	full	Type of assignment. Default "full" means normal run
debug	on, off	off	If on, already existing database (AnalysisResults and DesignFunctions) is deleted on running

Element's model:

(dynastats?, gatheropts?)

<dynastats> Child of assignment

Used for assignment type=dynastats. Controls dynastat assignment. Must contain either dshistory or dsd3plot tag

<dynastats>'s attributes

Name	Values	Default	Description
d3plotprefix	CDATA	-	LS-DYNA/Generic case to use for extraction
dynacase	CDATA	-	ID of the CASE used
iteration	CDATA	1	Iteration number to use
order	none, linear, quadratic	none	Set to linear or quadratic to enable use of metamodel.
stage	Match the IDREF rules.	Required	Which stage to use for calculation

Element's model:

(dshistory?, dsd3plot?, correlate?, bound?)

<gatheropts> Child of assignment

Used for assignment type=gather. Controls options for packing database

<gatheropts>'s attributes

Name	Values	Default	Description
withhistresp	yes, no	no	Set to "yes" to include histories and responses
withinput	yes, no	no	Set to "yes" to include input and extra input files

Element's model:

(inputfile*)

<dshistory/> Child of dynastats

The presence of this tag causes dynastats calculation on histories.

<dshistory>'s attributes

Name	Values	Default	Description
src	Match the IDREF rules.	Required	Reference to a composite or response, complete with "x_" prefix.

This element is always empty.

<dsd3plot> Child of dynastats

The presence of this tag causes dynastats calculation on d3plot data.

<dsd3plot>'s attributes

Name	Values	Default	Description
component	CDATA	Required	D3plot component to extract.
restype	ndv, stress, strain, result, misc, fld, beam	Required	D3plot response to extract

Element's model:

(fld?, coordmap?)

<fld/> Child of dsd3plot

This tag specifies and FLD curve. It's mandatory when FLD response type is used. Either curveid or t and n attributes must be given

<fld>'s attributes

Name	Values	Default	Description
curveid	CDATA	Implied	Curve ID from keyword file
n	CDATA	Implied	n parameter for parametric curve
t	CDATA	Implied	thickness parameter for parametric curve

This element is always empty.

<coordmap/> Child of dsd3plot

This tag enables following of coordinates instead of nodes for a given part

<coordmap>'s attributes

Name	Values	Default	Description
part	CDATA	Required	LS-DYNA part ID

This element is always empty.

<correlate/> Child of dynastats

The presence of this tag enables correlation in a dynastats calculation.

<correlate>'s attributes

Name	Values	Default	Description
entity	Match the IDREF rules.	Required	The entity to correlate with. Must refer to a response or a composite, complete with "x_" prefix.

This element is always empty

Appendix K: Glossary

Analytical Target Cascading. A rational approach to propagate the desired top-level system design targets to appropriate specifications for subsystems and components.

ANOVA. Analysis of variance. Used to perform variable screening by identifying insignificant variables. Variable regression coefficients are ranked based on their significance as obtained through a partial *F*-test. (See also *variable screening*).

ASA. Adaptive Simulated Annealing. An optimization method.

Bias error. The total error – the difference between the exact and computed response - is composed of a random and a bias component. The bias component is a systematic deviation between the chosen model (approximation type) and the exact response of the structure (FEA analysis is usually considered to be the exact response). Also known as the *modeling error*. (See also *random error*).

Binout. The name of the binary output file generated by LS-DYNA (Version 970 onwards).

Committee. A set of Neural Networks of the same order constructed using the same set of results. The nets are usually slightly different because a different weight initiator is typically used for the regression procedure of each individual net.

Compatibility. As applied to curves. A property of compared curves in which one curve is significantly longer than the other. See also *Partial Dynamic Time Warping.*

Composite function. A function constructed by combining responses and design variables into a single value. Symbolized by $\mathcal{F}$.

Concurrent simulation. The running of simulation tasks in parallel without message passing between the tasks.

Confidence interval. The interval in which a parameter may occur with a specified level of confidence. Computed using Student's *t*-test. Typically applied to accompany the significance of a variable in the form of an error bar.

Constraint. An absolute limit on a response variable specified in terms of an upper or lower limit.

Constrained optimization. The mathematical optimization of a function subject to specified limits on other functions.

Conventional Design. The procedure of using experience and/or intuition and/or *ad hoc* rules to improve a design.

Crossplot. A curve obtained by using the two ordinate values at a coinciding abscissa obtained from two separate functions. The two ordinate values are used as the abscissa and ordinate in the new crossplot. In LS-OPT two separate time histories are typically used to construct a single crossplot.

Delimiter. Symbol(s) to separate numeric fields in a text file. Typically spaces, tabs or commas.

Dependent. A function which is dependent on variables. Dependent variable.

Design of Experiments. See experimental design.

Design parameter. See *design variable*.

Design formula. A simple mathematical expression which gives the response of a design when the design variables are substituted. See *response surface*.

Design space. A region in the *n*-dimensional space of the design variables (x_1 through x_n) to which the design is limited. The design space is specified by upper and lower bounds on the design variables. Response variables can also be used to bound the design space.

Design surface. The response variable as a function of the design variables, used to construct the formulation of a design problem. (See also *response surface*, *design rule*).

Design sensitivity. The gradient vector of the response. The derivatives of the response function in terms of the design variables. $\mathrm{d}f/\mathrm{d}x_i$.

Design variable. An independent design parameter which is allowed to vary in order to change the design. Symbolized by (x_i or $\boldsymbol{x}$ (vector containing several design variables)).

DIC. See Digital Image Correlation.

Digital Image Correlation (DIC). An optical method for tracking deformations on a coupon for the purpose of identifying material properties.

Discipline. An area of analysis requiring a specific set of simulation tools, usually because of the unique nature of the physics involved, e.g. structural dynamics or fluid dynamics. In the context of MDO, often used interchangeably with solver.

Discrete Fréchet Distance. A similarity measure for curve matching. Typically used for material identification. See also *Dynamic Time Warping.*

DOE. Design of Experiments. See experimental design.

Domain reduction. The reduction of the region of interest in the design space during the optimization process.

D-**optimal.** The state of an experimental design in which the determinant of the moment matrix $\left|\boldsymbol{X}^T\boldsymbol{X}\right|$ of the least squares formulation is maximized.

DSA. Design sensitivity analysis.

Dynamic Time Warping. A similarity measure for curve matching. Typically used for material identification.

Ensemble. A collection of neural nets of different (usually thought of as ascending) order based on the same set of results.

Elliptic approximation. An approximation in which only the diagonal Hessian terms are used.

Experiment. Evaluation of a single design.

Experimental Design. The selection of designs to enable the construction of a design response surface. Sometimes referred to as the *Point Selection Scheme.*

Feasible Design. A design which complies with the constraint bounds.

Feedforward Neural Network. See *Neural Network.*

Function. A mathematical expression for a response variable in terms of design variables. Often used interchangeably with “response”. Symbolized by *f*.

Functionally efficient. See Pareto optimal.

Function evaluation. Using a solver to analyze a single design and produce a result. See *Simulation*.

Global variable. A variable of which the scope spans across all the design disciplines or solvers. Used in the MDO context.

Global approximation. A design function which is representative of the entire design space.

Global Optimization. The mathematical procedure for finding the global optimum in the design space. E.g. Genetic Algorithm, Particle Swarm, etc.

Global Sensitivity Analysis. A sensitivity analysis method which uses Sobol indices.

Gradient vector. A vector consisting of the derivatives of a function *f* in terms of a number of variables x_1 to x_n. $\mathbf{s} = [\mathrm{d}f / \mathrm{d}x_\mathrm{i}]$. See *Design Sensitivity*.

GSA. See Global Sensitivity Analysis.

History. Response history containing two columns of (usually time) data generated by a simulation.

Importance. See *Weight*.

Infeasible Design. A design which does not comply with the constraint functions. An entire design space or region of interest can sometimes be infeasible.

Isoline. A line representing a constant value of a scalar quantity. In the LS-OPT metamodel plotting feature isolines are used with metamodel functions.

Iteration. A cycle involving an experimental design, function evaluations of the designs, approximation and optimization of the approximate problem.

Kriging. A Metamodeling technique using Bayesian regression.

Latin Hypercube Sampling. The use of a constrained random experimental design as a point selection scheme for response approximation.

Least Squares Approximation. The determination of the coefficients in a mathematical expression so that it approximates certain experimental results by the minimization of the sum of the squares of the approximation errors. Used to determine response surfaces as well as calibrating analysis models.

Local Approximation. See *Gradient vector*.

Local variable. A variable of which the scope is limited to a particular discipline or disciplines. Used in the MDO context.

Material identification. See *parameter identification*.

MDO. Multidisciplinary design optimization.

Metamodeling. The construction of surrogate design models such as polynomial response surfaces, Artificial Neural Networks or Kriging surfaces from simulations at a set of design points.

Min-Max optimization problem. An optimization problem in which the maximum value considering several responses or functions is minimized.

Mode tracking. The identification of a mode closest in shape to a reference mode. LS-OPT uses e.g. the Modal Assurance Criterion to quantify the similarity of two mode shapes.

Model calibration. The optimal adjustment of parameters in a numerical model to simulate the physical model as closely as possible.

Modeling error. See *bias error*.

Multidisciplinary design optimization (MDO). The inclusion of multiple disciplines in the design optimization process. In general, only some design variables need to be shared between the disciplines to provide limited coupling in the optimization of a multidisciplinary target or objective.

Multihistory (also known as *multi-point history*). An ensemble of histories belonging to spatially separated points on a Finite Eelement model. Typically used for material parameter identification.

Multi-level Optimization. A formulation of the optimization problem in which optimization is conducted in a hierarchy of levels.

Multi-objective. An objective function which is constituted of more than one objective. Symbolized by F.

Multi-objective Optimization (MOO). Multi-objective optimization is the procedure for constructing a *Pareto optimal front*.

Multi-criteria. Refers to optimization problems in which several criteria are considered.

MOO. Multi-objective Optimization.

MP. Mathematical Programming. Mathematical optimization.

MSE. Mean Squared Error. Used for system identification.

Neural network approximation. The use of trained feedforward neural networks to perform non-linear regression, thereby constructing a non-linear metamodels *(see metamodeling)*.

Numerical sensitivity. A derivative of a function computed by using finite differences.

Noise. See *random error*.

Objective. A function of the design variables that the designer wishes to minimize or maximize. If there exists more than one objective, the objectives have to be combined mathematically into a single objective. Symbolized by Φ.

Optimal design. The methodology of using mathematical optimization tools to improve a design iteratively with the objective of finding the ‘best’ design in terms of predetermined criteria.

Optimization strategy. A strategy for metamodel-based optimization such as Single Stage, Sequential or Sequential with Domain Reduction.

Parallel Neural Networks. The concurrent solution of Feedforward Neural Networks using the job scheduler.

Parameter identification. See *System identification.*

Pareto optimal. A multi-objective design is Pareto-optimal if none of the objectives can be improved without at least one objective being affected adversely. A Pareto optimal front can be constructed using optimization.

Partial Dynamic Time Warping. A similarity measure for curve matching which also addresses curve incompatibility. Typically used for material identification.

Point selection scheme. Same as *experimental design*.

Preference function. A function of objectives used to combine several objectives into a single one suitable for the standard MP formulation.

Preprocessor. A graphical tool used to prepare the input for a solver.

Process. A series of analysis stages (or steps) designed to produce a result. Multistage process. Example: metal forming analysis which consists of several stages, e.g. gravity loading, stamping, springback, trimming, etc.

Process simulation. The use of computer programming, computer vision, and feedback to simulate manufacturing techniques.

Radial basis function network. The use of radial basis functions (RBFs) to approximate response functions. The LS-OPT default option is the Hardy's multi-quadrics but a user can also select Gaussian function as the radial basis function. This is a global approximation method.

Random error. The total error – the difference between the exact and computed response - is composed of a random and a bias component. The random component is, as the name implies, a random deviation from the nominal value of the exact response, often assumed to be normally distributed around the nominal value. (See also *bias error).*

RBDO. Reliability-based Design optimization.

Reasonable design space. A subregion of the design space within the region of interest. It is bounded by lower and upper bounds of the response values.

Region of interest. A sub-region of the design space. Usually defined by a mid-point design and a range of each design variable. Usually dynamic.

Reliability-based design optimization (RBDO). The performing of design optimization while considering reliability-based failure criteria in the constraints of the design optimization formulation. This implies the inclusion of random variables in the generation of responses and then extracting the standard deviation of the responses about their mean values due to the random variance and including the standard deviation in the constraint(s) calculation.

Residual. The difference between the computed response (using simulation) and the predicted response (using a response surface).

Response. A numerical indicator of the performance of the design. A function of the design variables approximated using a metamodel which can be used for optimization. Symbolized by *f*. Collected over all design iterations for plotting. (See also *history*).

Response quantity. See *response*.

Response Surface. A mathematical expression which relates the response variables to the design parameters. Typically computed using statistical methods.

Response variable. A response which is substituted as an input variable in a downstream stage of a multi-stage process. Response-expressions are allowed.

Result. A numerical indicator of the performance of the design. A result is not associated with a metamodel, but is typically used for intermediate calculations in metamodel-based analysis.

RBF. Radial Basis Function. RBF's are used as basis functions for metamodels (see also *metamodeling*). These functions are typically Gaussian.

RSM. Response Surface Methodology.

Run directory. The directory in which the simulations are done. Two levels below the *Work directory*. The run directory contains status files, the design coordinate file `XPoint` and all the simulation output. The `job_log` file which contains a log of the file transfer, the output log of the solver and a log of the result extraction also resides in this directory.

Saturated design. An experimental design in which the number of points equals the number of unknown coefficients of the approximation. For a saturated design no test can be made for the lack of fit.

Sampling. In the context of the GUI a *Sampling* is the same as a *Case*. It is based on a unique subset of variables.

In general, Sampling is synonymous with Point Selection or Experimental Design.

Scale factor. A factor which is specified as a divisor of a response in order to normalize the response.

Sensitivity. See *Design sensitivity*.

Slack constraint. A constraint with a slack variable. The violation of this constraint can be minimized.

Slack variable. The variable which is minimized to find a feasible solution to an optimization problem, e.g. e in: min e subject to $g_j(x) \leq e; \quad e \geq 0.$ See *Strictness.*

Similarity Measure. A mathematical measure of the distance between two curves.

Simulation. The analysis of a physical process or entity in order to compute useful responses. See *Function evaluation*.

Solver. A computational tool used to analyze a structure or fluid using a mathematical model. The executable software for such a tool. See *Discipline*.

Stage directory. A subdirectory of the work directory that bears the name of a stage and where database files resulting from extraction and the optimization process are stored.

Space Filling Experimental Design. A class of experimental designs that employ an algorithm to maximize the minimum distance between any two points.

Stochastic. Involving or containing random variables. Involving probability or chance.

Stage. A distinct step or operation in a process which typically reads input, processes the input and produces a result. Example: run a solver. Different stages can be dependent on one another.

Stopping Criterion. A mathematical criterion for terminating an iterative procedure.

Strictness. A number between 0 and 1 which signifies the strictness with which a design constraint must be treated. A zero value implies that the constraint *may* be violated. If a feasible design is possible all constraints will be satisfied. Used in the design formulation to minimize constraint violations. See *Slack variable.*

Subproblem. The approximate design subproblem constructed using response surfaces. It is solved to find an approximate optimum.

Subregion. See region of interest.

Successive (or Sequential) Approximation Method. An iterative method using the successive solution of approximate subproblems.

System identification. A procedure in which a numerical model is calibrated by optimizing selected parameters in order to minimize the residual error with respect to certain targeted responses. The targeted responses are usually derived from experimental results.

Target. A desired value for a response. The optimizer will not use this value as a rigid constraint. Instead, it will try to get as close as possible to the specified value.

Template. An input file in which some of the data has been replaced by variable names, e.g. `<<Radius>>`. A template may also contain the LS-DYNA *PARAMETER keyword with corresponding &-parameters. LS-OPT will recognize the parameters defined in the template and display them in the GUI.

Tolerance optimization. Optimization which includes tolerances on the input variables.

Trade-off curve. A curve constructed using *Pareto optimal* designs.

Transfer Variables. Variables transferred from one level to another in a multi-level optimization method.

Transformed variables. Variables which are transformed (mapped) to a different *n*-space using a functional relationship. The experimental design and optimization are performed in this space.

Variable screening. Method to remove insignificant variables from the design optimization process based on a ranking of regression coefficients using analysis of variance (ANOVA). (See also *ANOVA*).

Weight. A measure of importance of a response function or objective. Typically varies between 0 and 1.

Work directory. The directory is which the LS-OPT input files reside and where LS-OPT output is generated. Same as Project Home directory. See also *Run directory*.