

**LS-OPT<sup>®</sup> User's Manual**

**A DESIGN OPTIMIZATION AND  
PROBABILISTIC ANALYSIS TOOL  
FOR THE ENGINEERING ANALYST**

**NIELEN STANDER, Ph.D.  
WILLEM ROUX, Ph.D.  
ANIRBAN BASUDHAR, Ph.D.  
TRENT EGGLESTON, Ph.D.  
TUSHAR GOEL, Ph.D.  
KEN CRAIG, Ph.D.**

**April 2013  
Version 5.0**

Copyright © 2013  
**LIVERMORE SOFTWARE  
TECHNOLOGY CORPORATION**  
All Rights Reserved

**Corporate Address**

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

**Support Addresses**

Livermore Software Technology Corporation  
7374 Las Positas Road  
Livermore, California 94551  
Tel: 925-449-2500 ♦ Fax: 925-449-2507

**Email:** [sales@lstc.com](mailto:sales@lstc.com)

**Website:** [www.lstc.com](http://www.lstc.com)

**LS-OPT support site:** [www.lsoptsupport.com](http://www.lsoptsupport.com)

**LS-OPT Users Group:**

[http://groups.google.com/group/lsopt\\_user\\_group](http://groups.google.com/group/lsopt_user_group)

Livermore Software Technology Corporation

1740 West Big Beaver Road

Suite 100

Troy, Michigan 48084

Tel: 248-649-4728 ♦ Fax: 248-649-6328

**Disclaimer**

Copyright © 1999-2013 Livermore Software Technology Corporation. All Rights Reserved.

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 or 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](mailto:sales@lstc.com).

6/3/2013

# 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 provided 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 4<sup>th</sup> 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, Mr. Christoffer Belestam and Mr. Åke Svedin of Engineering Research AB (Linköping) as well as Dr.-Ing. Heiner Müllerschön, Dipl.-Ing. Marko Thiele and Dipl.-Math. Katharina Witowski of DYNAmore 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<sup>®</sup>/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<sup>®</sup> interface for frequency extraction and mode tracking has been added.

## Version 4.1:

9. Discrete sampling can be done on a variable by variable basis for most sampling schemes including *D*-Optimality, Space Filling and Full Factorial.
10. The Space Filling algorithm has been improved for accuracy and speed.
11. Job scheduling has been significantly improved. Environment variables can be exported through queuing systems.
12. Job data is displayed on the run progress bars with a selection to view the solver log file at any stage of the run.
13. Three injury criteria: a3ms, Chest Compression and Viscous Criterion have been added.
14. SPH, DBBEMAC and NODFOR groups have been added to the LS-DYNA response interface.
15. 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.
16. Responses can be linked to LS-DYNA cases (\*CASE keyword).

17. In addition to polynomials, Radial Basis Functions can now be used for parameter identification.
18. 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.
19. 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.
20. To avoid synchronization errors, the Experiments and AnalysisResults databases have been converted to self-contained .csv files.
21. The Run page has been rationalized. Clean start options are now available for all tasks.
22. 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:

23. The algorithm for constrained experimental design has been greatly improved. An optimization algorithm was introduced to locate design points within specified constraint bounds.
24. 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.
25. 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.
26. 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.
27. Kinematics for NODOUT-based responses and histories. Includes the calculation of deformation and distance in global, local and local-in-reference-frame coordinate systems.
28. DBFSI (fluid structure interaction) is available in the history and response interfaces.
29. 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.
30. Metamodel prediction accuracy based on PRESS error has been added as a stopping criterion for the Sequential Response Surface Method (SRSM).
31. 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.
32. 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.



33. Self-Organizing Maps is available to visualize simulation results.
34. 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.
35. LS-OPT database archiving has been expanded to include extra files such as solver input files.
36. Histories have been added to the GenEx (generic extraction) result extraction feature. In the past, only responses could be extracted.
37. 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).
38. A derivative history function has been added to compute the derivative of a time history, e.g. acceleration from velocity.
39. 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

40. 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.
41. Mode tracking is supported for all versions of LS-DYNA, including LS-DYNA MPP (merged to Version 4.2).
42. 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.
43. 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.
44. Head injury criterion (HIC) using three nodes for the different coordinate directions.
45. Support Vector Regression introduced as a metamodeling type.
46. 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 DYNAmore 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.

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 DYNAmore Nordic, Linköping as well as Dr.-Ing. Heiner Müllerschön and Dipl.-Math. Katharina Witowski of DYNAmore GmbH, Stuttgart, Germany have made major contributions as developers. Dr. Trent Eggleston has redesigned the job scheduler to accommodate the launching and load balancing of jobs with dependencies. Special thanks go to Katharina for patiently editing the manual, a major task in this version.

Nielen Stander and Anirban Basudhar

Livermore, CA

April, 2013



# TABLE OF CONTENTS

LS-OPT® User's Manual .....	1
1. Introduction.....	1
1.1. Overview of the manual.....	3
I – User's Manual .....	4
2. Getting Started .....	5
2.1. Installation of LS-OPT.....	5
2.2. Name conventions in LS-OPT .....	5
2.3. A modus operandi for design using response surfaces .....	6
2.3.1. Preparation for design.....	6
2.3.2. A step-by-step design optimization procedure .....	7
2.4. Recommended test procedure .....	8
2.5. Pitfalls in design optimization .....	9
2.5.1. Global optimality .....	9
2.5.2. Noise .....	9
2.5.3. Non-robust designs .....	9
2.5.4. Impossible designs .....	10
2.5.5. Non-unique designs .....	10
2.6. Setup of a simple optimization problem .....	10
2.6.1. Working directory.....	10
2.6.2. Startup.....	10
2.6.3. Task.....	11
2.6.4. Stage.....	12
2.6.5. Setup .....	14
2.6.6. Sampling and Metamodels.....	15
2.6.7. Optimization .....	16
2.6.8. Termination criteria .....	17
2.6.9. Run.....	17
2.6.10. Viewer.....	17
2.7. REFERENCES .....	17
3. Graphical User Interface .....	18
3.1. LS-OPT user interface (LS-OPTui).....	18
3.2. The GUI main window .....	20
3.2.1. Setting up a Process Flow .....	24
3.2.2. File Transfers between Stages .....	26
3.3. Run LS-OPT .....	27
3.3.1. Normal Run.....	27
3.3.2. Baseline Run .....	27
3.4. Restarting – Clean from Current Iteration .....	28
3.4.1. Augmentation of an existing design .....	28
3.5. Repair or modification of an existing job .....	28
3.6. Archive LS-OPT Database .....	29
4. Task Dialog – Selecting a Task and Strategy .....	31
4.1. Task selection.....	31

4.2.	Metamodel based optimization .....	33
4.3.	DOE study .....	33
4.4.	Direct optimization .....	33
4.5.	Probabilistic Analysis Tasks .....	33
4.5.1.	Direct Monte Carlo Analysis .....	33
4.5.2.	Metamodel-based Monte Carlo Analysis.....	34
4.6.	RBDO/Robust Parameter Design (Probabilistic Optimization Task).....	34
4.7.	Selecting strategies for metamodel-based optimization .....	34
4.7.1.	Single iteration .....	34
4.7.2.	Sequential strategy .....	35
4.7.3.	Sequential strategy with domain reduction.....	35
4.8.	Domain reduction in metamodel-based optimization .....	35
4.8.1.	Changing the behavior of the subdomain .....	36
4.8.2.	Setting the subdomain parameters* .....	37
4.9.	Create Pareto Optimal Front .....	37
4.10.	Global sensitivity analysis .....	37
4.11.	Verification runs .....	38
5.	Stage Dialog – Defining the Solver .....	39
5.1.	Introduction.....	39
5.2.	General Setup.....	39
5.2.1.	Command.....	42
5.2.2.	Input Files .....	42
5.2.3.	Parameterization of Input Files .....	43
5.2.4.	The LS-OPT parameter format .....	45
5.2.5.	System variables .....	46
5.3.	Package Interfaces .....	47
5.3.1.	LS-DYNA.....	47
5.3.2.	MSC-NASTRAN® (SOL 103).....	50
5.3.3.	LS-PREPOST .....	51
5.3.4.	LS-INGRID.....	52
5.3.5.	TrueGrid.....	52
5.3.6.	ANSA (BETA CAE Systems SA) .....	52
5.3.7.	HyperMorph.....	53
5.3.8.	µETA (BETA CAE Systems SA) .....	54
5.3.9.	User-defined program .....	55
5.3.10.	User-defined post-processor .....	55
5.4.	Solver Execution.....	57
5.4.1.	Specifying Computing Resources for Concurrent Processing.....	58
5.4.2.	Interfaces to Queuing Systems.....	59
5.4.3.	Using the LSTCVM secure proxy server.....	60
5.4.4.	Environment Variables .....	61
5.4.5.	Recovering Output Files .....	62
5.5.	File Operations.....	63
5.6.	The ‘N o r m a l’ termination status.....	64
5.7.	Managing disk space during run time.....	64
5.7.1.	Using the clean file to delete solver output files.....	65

5.8.	Alternative setups for running pre-processors .....	66
6.	History and Response Results.....	67
6.1.	Defining histories and responses .....	67
6.1.1.	Result extraction .....	71
6.2.	Extracting history and response quantities: LS-DYNA.....	71
6.2.1.	LS-DYNA binout results .....	71
6.2.2.	Kinematics .....	72
6.2.3.	LS-DYNA d3plot results .....	74
6.2.4.	Mass – Interfacing with d3hsp.....	75
6.2.5.	Frequency – Interfacing with d3eigv .....	76
6.3.	Extracting metal forming response quantities: LS-DYNA.....	79
6.3.1.	Thickness and thickness reduction.....	79
6.3.2.	FLD constraint .....	79
6.3.3.	Principal stress .....	82
6.4.	Generic Interfaces for History and Response extraction .....	83
6.4.1.	Expressions .....	83
6.4.2.	Crossplot history .....	83
6.4.3.	Function Interface .....	84
6.4.4.	Matrix operations .....	86
6.5.	Injury criteria .....	87
6.6.	Head Injury Criteria .....	87
6.6.1.	HIC.....	87
6.7.	Neck Criteria.....	88
6.7.1.	MOC .....	88
6.7.2.	NIC (rear impact).....	90
6.7.3.	Nij (Nce, Ncf, Nte, Ntf) .....	90
6.7.4.	Nkm (Nfa, Nea, Nfp, Nep).....	91
6.7.5.	LNL.....	92
6.8.	Chest Criteria .....	94
6.8.1.	Chest compression .....	94
6.8.2.	Viscous criterion (VC).....	95
6.8.3.	Thoracic Trauma Index (TTI).....	96
6.9.	Criteria for the Lower Extremities.....	97
6.9.1.	Tibia Index (TI).....	97
6.10.	Additional Criteria .....	98
6.10.1.	A3ms.....	98
6.11.	LS-DYNA Binout injury criteria .....	99
6.12.	REFERENCES .....	99
6.13.	The GenEx application for extracting entities from a text file .....	99
6.13.1.	The main window .....	99
6.13.2.	Creating a .g6 file for LS-OPT.....	103
6.13.3.	How to use GenEx from LS-OPT for extracting responses.....	104
6.13.4.	An example using GenEx to extract responses.....	104
6.13.5.	An example using "Repeated anchor vector" to extract histories.....	111
6.13.6.	An example using "Column vector" to extract histories.....	114
6.13.7.	How to extract the histories from LS-OPT .....	115

6.14.	User-defined interface for extracting results.....	116
6.15.	Nastran Frequency .....	118
6.16.	File Histories.....	118
7.	Setup Dialog – Defining the Variables.....	120
7.1.	Parameter Setup .....	120
7.1.1.	Constants.....	123
7.1.2.	Dependent variables.....	123
7.1.3.	Discrete and String variables .....	124
7.1.4.	Probabilistic Variables - Noise and Control Variables.....	124
7.1.5.	Probabilistic distributions .....	125
7.1.6.	Size and location of initial region of interest (range) .....	125
7.1.7.	Saddle direction: Worst-case design.....	125
7.2.	Stage Matrix.....	126
7.3.	Sampling Matrix .....	126
7.4.	Resources .....	127
7.5.	Features.....	128
7.5.1.	Evaluate Metamodel .....	128
8.	Sampling & Metamodel Dialog.....	130
8.1.	Metamodel types.....	131
8.1.1.	Polynomial .....	131
8.1.2.	Sensitivity .....	132
8.1.3.	Feedforward Neural networks and radial basis function networks.....	133
8.1.4.	Kriging parameters.....	137
8.1.5.	Support Vector Regression .....	138
8.1.6.	User-defined metamodel*.....	139
8.2.	General Options for Non-Polynomial Metamodels .....	140
8.2.1.	First Iteration Linear D-Optimal.....	141
8.2.2.	Include points of previous iterations.....	141
8.3.	Point selection schemes .....	141
8.3.1.	Overview.....	141
8.3.2.	D-Optimal point selection.....	143
8.3.3.	Latin Hypercube Sampling .....	144
8.3.4.	Space Filling .....	145
8.3.5.	Space Filling of Pareto Optimal Frontier.....	145
8.3.6.	User-defined point selection .....	146
8.3.7.	Replicate experimental points.....	147
8.3.8.	Remarks: Point selection.....	148
8.4.	Active Variables.....	149
8.5.	Sampling Features.....	150
8.5.1.	Approximate histories.....	151
8.5.2.	Verify Metamodel using Checkpoints .....	151
8.5.3.	Importing user-defined analysis results .....	152
8.5.4.	Changing the number of points on restart* .....	153
8.6.	Sampling Constraints .....	153
9.	Composite Dialog .....	156
9.1.	Introduction.....	156

9.1.1. Composite vs. response expressions .....	156
9.2. Defining Composites .....	156
9.3. Expression composite .....	158
9.4. Standard composite .....	158
9.4.1. Targeted composite (square root of MSE) .....	159
9.4.2. Mean squared error composite .....	161
9.4.3. Weighted composite .....	161
9.5. Curve Matching Composite .....	161
9.5.1. Ordinate-based Curve Matching .....	163
9.5.2. Curve Mapping .....	164
9.6. Standard Deviation Composite .....	165
10. Optimization Dialog – Objectives, Constraints and Algorithms .....	166
10.1. Formulation of the optimization problem .....	166
10.2. Defining objective functions .....	167
10.3. Defining a constraint .....	168
10.3.1. Internal scaling of constraints .....	169
10.3.2. Minimizing the maximum response or violation* .....	170
10.4. Algorithms .....	170
10.4.1. Setting parameters in the LFOPC algorithm* .....	171
10.4.2. Setting parameters in the genetic algorithm* .....	174
10.4.3. Setting parameters in the simulated annealing algorithm* .....	175
10.5. Algorithms for metamodel based Monte Carlo analysis .....	177
11. Termination Criteria .....	179
11.1. Metamodel based methods .....	179
11.1.1. Design Change Tolerance and Objective Function Tolerance .....	180
11.1.2. Response Accuracy Tolerance .....	181
11.1.3. Maximum Number of Iterations .....	181
11.2. Direct Optimization .....	181
12. Probabilistic Modeling and Tasks .....	183
12.1. Probabilistic problem modeling .....	183
12.2. Probabilistic distributions .....	184
12.3. Probabilistic variables .....	193
12.3.1. Setting the nominal value of a probabilistic variable .....	194
12.3.2. Bounds of a probabilistic variable .....	195
12.4. Monte Carlo analysis .....	195
12.5. Monte Carlo analysis using a metamodel .....	196
12.6. RBDO/Robust parameter design .....	197
13. Running the Design Task .....	200
13.1. Running the design task .....	200
13.2. Analysis monitoring .....	200
13.3. Job monitoring – the Progress dialog .....	201
13.3.1. Error termination of a solver run .....	203
13.4. Restarting .....	203
13.5. Directory structure .....	204
13.6. Log files and status files .....	205
14. Viewing Results .....	206



14.1. Viewer overview .....	206
14.1.1. Plot Selector .....	206
14.1.2. General Plot Options .....	207
14.1.3. Plot Rotation .....	210
14.1.4. Point Selection .....	210
14.1.5. Split Window .....	213
14.1.6. Save Plot Setup .....	215
14.1.7. Command line options .....	215
14.1.8. Iteration Panel .....	217
14.1.9. Ranges .....	217
14.2. Visualization of Simulation Results .....	218
14.2.1. Correlation Matrix .....	218
14.2.2. Scatter Plot .....	220
14.2.3. Parallel Coordinate Plot .....	222
14.2.4. History Plot .....	224
14.2.5. Statistical Tools .....	226
14.2.6. Correlation Bars .....	230
14.3. Visualization of Metamodel Results .....	232
14.3.1. Surface Plot .....	232
14.3.2. 2D Interpolator Plot .....	238
14.3.3. Accuracy Plot .....	240
14.3.4. Sensitivities .....	241
14.3.5. History Plot .....	243
14.4. Visualization of Optimization Results .....	246
14.4.1. Optimization History .....	246
14.4.2. Variables Plot .....	247
14.5. Visualization of Pareto Optimal Solutions .....	248
14.5.1. Tradeoff Plot .....	248
14.5.2. Parallel Coordinate Plot .....	249
14.5.3. Hyper-Radial Visualization .....	250
14.5.4. Self-Organizing Maps .....	252
14.6. Stochastic Analysis .....	254
14.6.1. Statistics .....	254
14.6.2. Correlation Bars .....	256
14.6.3. Stochastic Contribution .....	257
14.7. References .....	258
15. LS-DYNA Results Statistics .....	259
15.1. Working with the plots .....	260
15.2. Creation of a plot .....	260
15.2.1. Step 1 – Fringe plot or History plot .....	260
15.2.2. Step 2 – D3Plot component or History .....	262
15.2.3. Step 3 - Statistics .....	263
15.2.4. Step 4 – Visualization in LS-PREPOST .....	265
15.3. Monte Carlo and metamodel analysis .....	267
15.3.1. Monte Carlo .....	267
15.3.2. Metamodels and residuals .....	267

15.4.	Correlation .....	269
15.4.1.	Correlation of fringe plots or histories with responses .....	269
15.4.2.	Correlation of fringe plots or histories with variables .....	270
15.5.	Stochastic contribution of a variable.....	271
15.6.	Safety margin.....	272
15.7.	Viewing LS-OPT histories.....	274
15.8.	Bifurcation investigations .....	276
15.8.1.	Automatic detection .....	278
15.8.2.	Manual detection.....	278
15.9.	Displacement magnitude issues* .....	279
15.10.	Metallforming options.....	281
15.11.	User-defined statistics* .....	283
15.12.	Re-use and persistence of an evaluation methodology* .....	284
16.	Applications of Optimization.....	285
16.1.	Parameter Identification.....	285
16.1.1.	Optimization algorithm.....	285
16.1.2.	Matching scalar values.....	285
16.1.3.	Curve matching metric.....	286
16.1.4.	Sampling constraints.....	286
16.1.5.	Parameterization of solver input curves.....	286
16.1.6.	Viewer.....	286
16.2.	Sensitivity analysis.....	288
16.2.1.	DOE task.....	289
16.2.2.	Sequential.....	289
16.2.3.	Viewer.....	289
16.3.	Multidisciplinary Design Optimization (MDO) .....	292
16.4.	Multi-objective optimization (MOO) .....	293
16.4.1.	Direct Genetic Algorithm .....	293
16.4.2.	Metamodel-based Genetic Algorithm.....	293
16.4.3.	Viewer.....	293
16.5.	Shape Optimization.....	293
16.6.	Worst-case design .....	294
II –	Examples.....	296
17.	Examples – Optimization .....	297
17.1.	Two-bar truss (3 variables).....	297
17.1.1.	Description of problem .....	297
17.1.2.	A first approximation using linear response surfaces.....	300
17.1.3.	Updating the approximation to second order.....	305
17.1.4.	Reducing the region of interest for further refinement .....	308
17.1.5.	Automating the design process .....	311
17.2.	Small car crash (2 variables).....	317
17.2.1.	Introduction.....	317
17.2.2.	Design criteria and design variables .....	317
17.2.3.	Design formulation .....	318
17.2.4.	Modeling.....	321
17.2.5.	Single iteration run using Radial Basis Functions .....	322

17.2.6. Automated run using linear metamodels .....	325
17.2.7. Mixed-discrete optimization .....	329
17.2.8. Optimization using Direct GA simulation .....	331
17.3. Impact of a cylinder (2 variables) .....	334
17.3.1. Problem statement.....	334
17.3.2. A first approximation.....	336
17.3.3. Refining the design model using a second iteration .....	339
17.3.4. Third iteration .....	341
17.3.5. Response filtering: using the peak force as a constraint .....	343
17.4. Sheet-metal forming (3 variables) .....	346
17.4.1. Problem statement.....	346
17.4.2. First Iteration.....	348
17.4.3. Automated design .....	354
17.5. Large vehicle crash and vibration (MDO/MOO) (7 variables) .....	359
17.5.1. FE Modeling .....	359
17.5.2. Design formulation .....	361
17.5.3. Multi-objective optimization using metamodel-based optimization .....	361
17.5.4. Multi-objective optimization using Direct GA simulation .....	368
17.6. Knee impact with variable screening (11 variables).....	372
17.6.1. FE modeling.....	372
17.6.2. Design formulation .....	374
17.6.3. Input preparation.....	375
17.6.4. Variable screening .....	377
17.6.5. Optimization strategy.....	379
17.6.6. Optimization history results .....	380
17.6.7. Summary of results .....	380
17.7. Shape optimization of a front rail using ANSA and $\mu$ ETA .....	384
17.7.1. Problem Statement .....	384
17.7.2. Solution.....	385
17.7.3. Results.....	389
17.8. Optimization with analytical design sensitivities .....	391
17.8.1. Problem Statement .....	391
17.8.2. Solution.....	393
17.8.3. Results.....	396
17.9. REFERENCES .....	398
18. Examples – Parameter Identification.....	399
18.1. Material identification (elastoplastic material) (2 variables).....	399
18.1.1. Problem statement.....	399
18.1.2. Ordinate-based Curve Matching .....	400
18.1.3. Targeted composite formulation .....	403
18.1.4. Results.....	405
18.1.5. Mean Squared Error (MSE) formulation .....	405
18.1.6. Targeted composite formulation .....	408
18.2. System identification with hysteretic curves .....	410
18.2.1. Problem statement.....	410
18.2.2. Solution using Curve Mapping .....	411

18.2.3. Results.....	413
18.3. REFERENCES .....	415
19. Examples – Probabilistic Analysis .....	416
19.1. Probabilistic Analysis .....	416
19.1.1. Overview.....	416
19.1.2. Problem description .....	416
19.1.3. Direct Monte Carlo evaluation.....	418
19.1.4. Monte Carlo using metamodel.....	420
19.1.5. Bifurcation analysis .....	422
19.2. Bifurcation/Outlier Analysis.....	423
19.2.1. Overview.....	423
19.2.2. Problem description .....	423
19.2.3. Monte Carlo evaluation.....	424
19.2.4. Automatic identification of buckling modes.....	426
19.2.5. Manual identification of buckling modes .....	428
19.3. RBDO (Reliability-based design optimization) using FOSM (First Order Second Moment Method).....	430
19.4. Robust Parameter Design.....	431
19.5. Using Stochastic Fields.....	437
19.5.1. Using only a stochastic field.....	438
19.5.2. A variable and a stochastic field .....	440
19.5.3. Replicate experiments using stochastic fields.....	442
19.5.4. Using fixed stochastic fields .....	444
19.6. REFERENCES .....	447
III – Theory.....	449
20. Response Surface Methodology .....	450
20.1. Introduction.....	450
20.1.1. Approximating the response .....	450
20.1.2. Factors governing the accuracy of the response surface.....	452
20.1.3. Advantages of the method .....	452
20.1.4. Other types of response surfaces.....	452
20.2. Experimental design.....	452
20.2.1. Factorial design.....	453
20.2.2. Koshal design.....	453
20.2.3. Central composite design.....	454
20.2.4. D-optimal design.....	454
20.2.5. Latin Hypercube Sampling (LHS).....	455
20.2.6. Space-filling designs.....	456
20.2.7. Random number generator.....	460
20.2.8. Reasonable experimental designs .....	460
20.3. Model adequacy checking.....	461
20.3.1. Residual sum of squares.....	461
20.3.2. RMS error .....	461
20.3.3. Maximum residual .....	461
20.3.4. Prediction error .....	461
20.3.5. PRESS residuals.....	462

20.3.6. The coefficient of multiple determination $R^2$ .....	462
20.3.7. $R^2$ for Prediction .....	463
20.3.8. Iterative design and prediction accuracy.....	463
20.4. ANOVA .....	463
20.4.1. The confidence interval of the regression coefficients .....	463
20.4.2. The significance of a regression coefficient $b_j$ .....	464
20.5. REFERENCES .....	465
21. Metamodeling Techniques.....	466
21.1. Neural networks.....	466
21.1.1. Model adequacy checking.....	470
21.1.2. Feedforward neural networks .....	472
21.1.3. Radial basis function networks .....	475
21.2. Kriging* .....	479
21.3. Support Vector Regression .....	480
21.4. Concluding remarks: which metamodel?.....	482
21.5. REFERENCES .....	483
22. Optimization .....	486
22.1. Theory of optimization .....	486
22.2. Normalization of constraints and variables.....	487
22.3. Gradient computation and the solution of optimization problems.....	488
22.4. Optimization methods.....	489
22.5. Strategies for metamodel-based optimization.....	489
22.5.1. Single stage .....	490
22.5.2. Sequential strategy .....	490
22.5.3. Sequential strategy with domain reduction.....	490
22.5.4. How do I choose an appropriate strategy for metamodel-based optimization?.....	491
22.6. Sequential response surface method (SRSM).....	491
22.7. Leapfrog optimizer for constrained minimization (LFOPC).....	494
22.8. Genetic algorithm.....	494
22.8.1. Terminology.....	495
22.8.2. Encoding .....	495
22.8.3. Algorithm.....	496
22.9. Multi-objective optimization using genetic algorithms .....	499
22.9.1. Non-domination criterion.....	499
22.9.2. Pareto optimal solutions.....	500
22.9.3. Pareto optimal set.....	500
22.9.4. Pareto optimal front .....	500
22.9.5. Ranking.....	500
22.9.6. Convergence vs. diversity .....	501
22.9.7. Elitist non-dominated sorting genetic algorithm (NSGA-II).....	501
22.9.8. Elitism in NSGA-II.....	503
22.9.9. Diversity preservation mechanism in NSGA-II – crowding distance calculation.....	503
22.10. Adaptive simulated annealing (ASA).....	504
22.10.1. Algorithm.....	504
22.10.2. Acceptance function.....	505
22.10.3. Sampling algorithm.....	505

22.10.4. Cooling schedule.....	506
22.10.5. Stopping criterion.....	507
22.10.6. Re-annealing .....	507
22.11. Hybrid algorithms .....	509
22.12. Visualization of the Pareto optimal frontier.....	509
22.12.1. Trade-off plot .....	509
22.12.2. Hyper-radial visualization (HRV).....	510
22.12.3. Parallel co-ordinate plot (PCP) .....	512
22.12.4. Self organizing maps (SOM) .....	513
22.13. Performance metrics for multi-objective optimization.....	514
22.13.1. Number of nondominated points .....	514
22.13.2. Spread .....	514
22.13.3. Standard deviation of crowding distance.....	515
22.13.4. Min/Max of objectives.....	515
22.13.5. Hypervolume.....	515
22.13.6. Number of common points .....	515
22.13.7. Number of new nondominated solutions .....	515
22.13.8. Number of old dominated solutions $n(Q)$ .....	516
22.13.9. Consolidation ratio.....	516
22.13.10. Improvement ratio.....	516
22.14. Discrete optimization .....	517
22.14.1. Discrete variables.....	517
22.14.2. Discrete optimization .....	517
22.14.3. Mixed-discrete optimization .....	517
22.14.4. Discrete optimization algorithm: genetic algorithm .....	517
22.14.5. Objective function for discrete optimization .....	518
22.14.6. Sequential strategy .....	518
22.15. Summary of the optimization process.....	518
22.15.1. Convergence to an optimal point .....	519
22.16. REFERENCES .....	520
23. Applications of Optimization.....	523
23.1. Multicriteria design optimization.....	523
23.1.1. Euclidean distance function .....	523
23.1.2. Maximum distance.....	524
23.2. Multidisciplinary design optimization .....	525
23.3. System identification using nonlinear regression .....	526
23.3.1. Ordinate-based Curve Matching.....	526
23.3.2. Curve Mapping .....	527
23.3.3. Minimizing the maximum residual (Min-Max).....	530
23.3.4. Nonlinear regression: Confidence intervals.....	531
23.4. Worst-case design .....	532
23.5. REFERENCES .....	533
24. Probabilistic Fundamentals.....	534
24.1. Introduction.....	534
24.2. Probabilistic variables.....	534
24.3. Basic computations .....	535

24.3.1. Mean, variance, standard deviation, and coefficient of variation .....	535
24.3.2. Correlation of responses .....	535
24.3.3. Confidence intervals .....	536
24.4. Probabilistic methods.....	536
24.4.1. Monte Carlo simulation .....	537
24.4.2. Monte Carlo analysis using metamodels .....	538
24.4.3. Correlated variables .....	539
24.4.4. First-Order Second-Moment Method (FOSM).....	540
24.4.5. Design for six-sigma methods .....	540
24.4.6. The most probable point .....	541
24.4.7. FORM (First Order Reliability Method).....	542
24.4.8. Design sensitivity of the most probable point.....	543
24.5. Required number of simulations .....	543
24.5.1. Overview.....	543
24.5.2. Background.....	544
24.5.3. Competing role of variance and bias .....	545
24.5.4. Confidence interval on the mean .....	546
24.5.5. Confidence interval on a new evaluation.....	546
24.5.6. Confidence interval on the noise (stochastic process) variance.....	547
24.5.7. Probability of observing a specific failure mode .....	548
24.6. Outlier analysis .....	549
24.7. Stochastic contribution analysis.....	550
24.7.1. Linear estimation .....	550
24.7.2. Second and higher order estimation.....	551
24.8. Reliability-based design optimization (RBDO)* .....	552
24.9. Robust parameter design.....	553
24.9.1. Fundamentals .....	553
24.9.2. Methodology .....	555
24.9.3. Experimental design.....	555
24.10. REFERENCES .....	555
IV – Appendix .....	557
Appendix A: LS-DYNA Binout Commands .....	558
A.1 Binout Histories .....	558
A.2 Averaging, filtering, and slicing Binout histories .....	559
A.3 Binout Responses.....	560
A.3.1 Binout injury criteria.....	560
A.3.2 Bilinear FLD constraint .....	560
Appendix B: LS-DYNA Binout Components .....	562
Appendix C: LS-DYNA D3Plot Commands .....	568
C.1 D3Plot histories .....	568
C.1.1 Slicing D3Plot histories.....	569
C.1.2 D3Plot FLD results .....	569
C.2 D3Plot responses .....	570
Appendix D: LS-DYNA D3Plot Components.....	571
Appendix E: Database Files.....	574
E.1 Design flow .....	574

E.2 Output files .....	574
E.2.1 Intermediate database files .....	575
E.2.2 Database files in .csv (comma separated variables) format .....	576
E.2.3 Text output files.....	576
E.3 Database file formats .....	577
E.3.1 Variable types.....	577
E.3.2 The Experiments_n.csv file .....	577
E.3.3 The AnalysisResults_n.csv file .....	578
E.3.4 The DesignFunctions file .....	578
E.3.5 The VirtualHistoryFunction file .....	580
E.3.6 The OptimizationHistory file .....	580
E.3.7 The ExtendedResults file.....	582
E.3.8 The OptimumResults file .....	583
E.3.9 The <i>Sobol_GSA</i> file .....	583
E.3.10 The <i>lsopt_results</i> file .....	583
E.3.11 The <i>lsopt_db</i> file.....	584
Appendix F: Mathematical Expressions .....	585
F.1 Syntax rules .....	585
F.2 Intrinsic functions .....	585
F.3 Special functions.....	587
F.3.1 Functions to apply to response histories .....	587
F.3.2 Arguments used in functions .....	588
F.3.3 Options for MeanSqErr arguments.....	590
F.4 Matrix functions.....	591
F.5 Reserved variable names .....	591
F.6 Constants associated with histories .....	592
F.7 Generic expressions .....	592
F.8 Examples illustrating syntax of expressions.....	593
Appendix G: Injury Criteria.....	595
G.1 Syntax of Injury Criterion Commands.....	595
G.1.1 Options for MOC arguments.....	596
G.1.2 MOC Input constants for various dummy types .....	596
G.1.3 Options for NIC arguments.....	597
G.1.4 Options for Nij arguments.....	597
G.1.5 Nij Input constants for various dummy types .....	598
G.1.6 Options for Nkm arguments.....	599
G.1.7 Nkm Input constants .....	599
G.1.8 Options for LNL arguments.....	599
G.1.9 LNL Input constants.....	600
G.1.10 Options for Chest Compression arguments .....	600
G.1.11 Chest Comporession Input constants for various dummy types .....	601
G.1.12 Options for Viscous Criterion arguments .....	601
G.1.13 Viscous Criterion Input constants for various dummy types.....	601
G.1.14 Options for TTI arguments .....	602
G.1.15 Options for TI arguments.....	602
G.1.16 TI Input constants for various dummy types .....	603



G.1.17 Options for a3ms arguments .....	603
Appendix H: Installing LS-OPT .....	605
H.1 Download .....	605
H.2 Installation.....	605
H.2.1 Linux .....	605
H.2.2 Windows .....	605
H.2.3 How to run LS-DYNA from LS-OPT using the license server (Windows) .....	606
H.3 Remote job scheduling.....	606
H.4 Simple manual setup for running LS-OPT and solvers on different machines .....	607
H.5 Using an external queuing or job scheduling system.....	608
H.5.1 Installation.....	608
H.5.2 Installation for all remote machines running simulation jobs.....	609
H.5.3 Environment variables LSOPT_HOST and LSOPT_PORT .....	609
H.5.4 Examples.....	610
H.6 Mechanics of the queuing process .....	611
H.7 User-defined and Blackbox queuing systems .....	613
H.7.1 User-defined queuing systems .....	614
H.7.2 Blackbox queuing system .....	614
H.7.3 LsoptJobCheck script.....	615
H.7.4 LsoptJobDel script .....	616
H.8 Honda queuing system.....	616
H.8.1 Mechanics of the Honda queuing process.....	616
H.9 Abnormal termination and retrying the job submission.....	618
H.9.1 User-defined abnormal termination .....	618
H.9.2 Troubleshooting .....	618
H.10 Enabling LSTCVM job proxy support.....	619
H.10.1 LSTCVM options.....	619
H.10.2 LSTCVM server installation.....	620
H.10.3 Environment Variables .....	620
H.10.4 Configuring the lstcvm_run client .....	620
H.11 Passing environment variables through LS-OPT.....	621
H.11.1 .env files .....	622
H.11.2 Executables .....	622
Appendix I: Killing Jobs.....	624
I.1 Overview of How Jobs are Killed .....	624
I.1.1 Case 1 (Killing Local Jobs): .....	625
I.1.2 Case 2 (Killing Jobs run using runqueuer/wrapper):.....	625
I.1.3 Case 3 (Jobs are run using the HONDA queuing option): .....	625
I.1.4 Case 4 (Jobs are run using the BLACK BOX queuing option):.....	626
I.2 Killing Jobs using LS-OPT, LS-OPTui, and LSKILLJOB .....	626
I.2.1 Killing All Jobs in Bulk.....	626
I.2.2 Kill One Job.....	626
I.3 Kill Level.....	627
I.3.1 LS-DYNA Jobs (non-queued).....	627
I.3.2 LS-DYNA Jobs (queued with runqueuer/wrapper).....	627
I.3.3 Other Jobs (non-queued) .....	628

I.3.4 Other Jobs (queued with runqueueer/wrapper) .....	628
I.3.5 All BLACKBOX Queued Jobs.....	628
I.3.6 All HONDA Queued Jobs .....	629
I.4 Increasing the Kill Level .....	629
I.5 Termination Status for Killed Jobs .....	630
I.6 Flagging a Job for Restart.....	631
Appendix J: Document Type Definition (DTD).....	633
Appendix K: Glossary.....	660

# TABLE OF FIGURES

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. ....	11
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. ....	11
Figure 2-3: Task dialog. Select the main task and strategy .....	12
Figure 2-4: Stage dialog - Setup. Select the solver package name, the command and the solver input file ..	13
Figure 2-5: Stage dialog – Parameters. Displays the parameters found in the input file specified in Setup ..	13
Figure 2-6: Stage dialog - Responses page. Select a response type from the list on the right to add a new response definition. ....	14
Figure 2-7: Parameter Setup dialog. Define the parameter type and required values. ....	15
Figure 2-8: Sampling dialog. Select the metamodel type and point selection scheme. ....	15
Figure 2-9: Optimization - Objectives. Select the objective components from the list on the right. ....	16
Figure 2-10: Optimization - Constraints. Select constraints from the list on the right. Specify lower and upper bounds as required. ....	16
Figure 2-11: Termination Criteria dialog. Specify the maximum number of iterations .....	17
Figure 3-1: Startup Dialog of Isoptui .....	19
Figure 3-2: Main LS-OPT GUI window for a setup of a Metamodel-base optimization .....	21
Figure 3-3: Setup of a complex optimization problem .....	25
Figure 3-4: Creating stage dependencies .....	26
Figure 3-5: File transfers between dependent stages .....	26
Figure 3-6: Dialog to specify options for archive LS-OPT database.....	30
Figure 4-1: Task and Strategy selection.....	31
Figure 4-2: Domain reduction dialog.....	36
Figure 5-1: Stage dialog Setup panel .....	41
Figure 5-2: Definition of Extra Input Files .....	43
Figure 5-3: Parameter panel: list of parameters found in stage input files .....	45
Figure 5-4: Stage Setup LS-DYNA advanced options .....	48
Figure 5-5: Stage Setup for ANSA .....	52
Figure 5-6: Stage Setup for HyperMorph .....	53
Figure 5-7: MetaPost interface.....	54
Figure 5-8: Stage dialog Setup panel .....	57
Figure 5-9: Definition of Resources for a Stage .....	58
Figure 5-10: Definition of Environment Variables.....	61
Figure 5-11: Database recovery options .....	62
Figure 5-12: File Operations within a Stage run directory .....	63
Figure 6-1: Histories definition in the GUI.....	68
Figure 6-2: Response extraction: LS-DYNA NODOUT interface.....	72
Figure 6-3: Local and global coordinate systems .....	74
Figure 6-4: Response extraction from d3plot .....	75
Figure 6-5: Interface for extraction of Mass and related entities from LS-DYNA output d3hsp.....	76
Figure 6-6: Interface for extraction of frequencies from LS-DYNA output d3eigv.....	77
Figure 6-7: Thickness or Thickness reduction interface.....	79

Figure 6-8: FLD curve – constraint definition.....	81
Figure 6-9: Definition of General FLD constraint.....	81
Figure 6-10: Principal Stress Interface .....	82
Figure 6-11: Interface to define a crossplot history .....	83
Figure 6-12: Interface to define derivative history .....	84
Figure 6-13: Interface to define a filtered history .....	85
Figure 6-14: Matrix Expression: Initialization of a matrix.....	86
Figure 6-15: GenEx dialog.....	100
Figure 6-16: Definition of a GenEx Response.....	104
Figure 6-17: GenEx dialog; definition of an anchor.....	105
Figure 6-18: GenEx dialog; definition of an entity.....	106
Figure 6-19: GenEx dialog; definition of an entity.....	106
Figure 6-20: GenEx dialog; definition of an anchor.....	107
Figure 6-21: GenEx dialog; definition of an anchor.....	108
Figure 6-22: GenEx dialog; definition of an entity.....	109
Figure 6-23: GenEx dialog; definition of an anchor.....	110
Figure 6-24: GenEx dialog; definition of an entity.....	111
Figure 6-25: GenEx dialog; definition of an entity.....	112
Figure 6-26: GenEx dialog; definition of an repeat anchor vector .....	113
Figure 6-27: GenEx dialog; definition of a history.....	114
Figure 6-28: GenEx dialog; definition of a column vector entity.....	115
Figure 6-29: Interface to define a GenEx History .....	116
Figure 6-30: Extracting a Response using a user-defined program.....	117
Figure 6-31: Interface for Extraction of Frequencies from Nastran results.....	118
Figure 6-32: File Histories .....	119
Figure 7-1: Setup Dialog – Parameter Setup panel in LS-OPTui.....	120
Figure 7-2: Definition of discrete values .....	124
Figure 7-3: Parameter setup panel for probabilistic tasks.....	125
Figure 7-4: Stage Matrix .....	126
Figure 7-5: Sampling Matrix .....	126
Figure 7-6: Setup – Resources .....	127
Figure 7-7: Setup – Features .....	128
Figure 8-1: Sampling dialog – metamodel and point selection settings.....	130
Figure 8-2: Sampling Dialog: Sensitivity options .....	132
Figure 8-3: Feedforward Neural Network Efficiency Options.....	134
Figure 8-4: Radial Basis Function Network Advance Options .....	136
Figure 8-5: Kriging Advanced Options .....	137
Figure 8-6: Metamodel selection Support Vector Regression.....	138
Figure 8-7: User Defined Metamodel Options .....	139
Figure 8-8: Point selection schemes .....	142
Figure 8-9: D-optimal point selection: advanced options.....	144
Figure 8-10: Sampling Dialog: User-defined point selection.....	146
Figure 8-11: Sampling Dialog options for direct Monte Carlo Analysis.....	148
Figure 8-12: Sampling Dialog: Active Variables panel.....	149
Figure 8-13: Sampling Features.....	150
Figure 8-14: Definition of Sampling Constraints by selection from list or new creation.....	154

Figure 8-15: Sampling constrain wizard: definition of an expression and bounds.....	154
Figure 9-1: Composites Dialog.....	157
Figure 9-2: Definition of a Composite Expression.....	158
Figure 9-3: Standard Composite Interface.....	159
Figure 9-4: Definition of targeted (Root MSE) composite response in LS-OPTui.....	160
Figure 9-5: History Matching Composite Dialog.....	162
Figure 9-6: Definition of a Standard Deviation composite.....	165
Figure 10-1: Objective panel in LS-OPTui Optimization dialog.....	167
Figure 10-2: Constraints panel in LS-OPTui.....	169
Figure 10-3: Selecting the optimization algorithm used for the optimization on the metamodel.....	171
Figure 10-4: LFOP settings.....	172
Figure 10-5: GA settings.....	174
Figure 10-6: ASA settings.....	176
Figure 10-7: Algorithm Options for Metamodel based Monte Carlo Analysis.....	177
Figure 11-1: Termination Criteria dialog for metamodel based optimization, strategy sequential or sequential with domain reduction.....	179
Figure 11-2: Termination Criteria dialog for direct optimization.....	181
Figure 12-1: Setup Dialog, Parameter Setup: Definition of Probabilistic Distributions.....	184
Figure 12-2: Beta distribution.....	185
Figure 12-3: Binomial distribution.....	186
Figure 12-4: Lognormal distribution.....	187
Figure 12-5: Normal Distribution.....	188
Figure 12-6: Truncated Normal Distribution.....	189
Figure 12-7: Uniform Distribution.....	190
Figure 12-8: User defined distribution.....	191
Figure 12-9: Weibull distribution.....	193
Figure 12-10: 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. ....	194
Figure 12-11: Bounds exceeded due to variable uncertainty.....	195
Figure 12-12: 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.....	197
Figure 12-13: Probabilistic constraint definition for RBDO. A target failure probability is defined.....	198
Figure 12-14: An example of objective function for RBDO (top) and robust design (bottom). Standard deviation is defined as the objective in latter case.....	199
Figure 13-1: Main GUI while optimization is running.....	200
Figure 13-2: Progress dialog displaying progress of stage runs.....	201
Figure 13-3 : Directory structure in LS-OPT.....	204
Figure 14-1: Plot Selector.....	206
Figure 14-2: Plot Selector with additional plot information.....	207

Figure 14-3: Selection for placement of new plot in the plot selector.....	207
Figure 14-4: General options .....	208
Figure 14-5: Options for printing (right) and saving (left) images.....	210
Figure 14-6: Point selection window for single (front) and multi (background) point selection .....	211
Figure 14-7: Cross-display of selected points.....	213
Figure 14-8: Example for split option.....	214
Figure 14-9: Detachable panels .....	214
Figure 14-10: Plot Selector with previously saved setups.....	215
Figure 14-11: Iteration Panel- only current iteration (left), all previous/ all iterations (middle), iteration range and step size (right).....	217
Figure 14-12: Ranges selections .....	218
Figure 14-13: Correlation matrix with scatter plots, histograms and linear correlation coefficient.....	219
Figure 14-14: Correlation matrix, only correlation coefficients.....	219
Figure 14-15: Scatter plot in View panel in LS-OPTui. The 4 <sup>th</sup> dimension is represented by point color...	221
Figure 14-16: Parallel Coordinate Plot with selected point .....	223
Figure 14-17: History Plot, curves colored by variable.....	225
Figure 14-18: Histories from simulations colored by variable with target curve (File history).....	226
Figure 14-19: Histogram constructed from simulation results .....	227
Figure 14-20: Standard deviation and mean value of selected response constructed from simulation results .....	228
Figure 14-21: Probability of Mass > 0.5 with 95% confidence interval in red constructed from simulation results .....	229
Figure 14-22: Coefficient of Correlation plot with 95% confidence interval in red .....	231
Figure 14-23: Metamodel options.....	232
Figure 14-24: 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. ....	235
Figure 14-25: Metamodel plot showing point color coding for iteration numbers.....	236
Figure 14-26: Surface plot representing only the region of interest of the fourth iteration.....	236
Figure 14-27: 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.....	237
Figure 14-28: Plot showing isolines and points opposite the “Points” tab.....	237
Figure 14-29: Metamodel plot showing standard deviation of the Neural Net committee values. ....	238
Figure 14-30: 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) .....	240
Figure 14-31: Computed vs. Predicted plot in View panel in LS-OPTui. The points are color-coded to represent the feasibility. The largest points represent the most recent iteration.....	241
Figure 14-32: Linear ANOVA plot in View panel in LS-OPTui, sorted.....	242
Figure 14-33: Sorted global sensitivities of all responses and composites.....	243
Figure 14-34: Predicted Histories colored by variable .....	244
Figure 14-35: Predicted History with nearest history and maximal residual.....	245
Figure 14-36: Optimization History plot of a variable – variable values (red) and subregions (blue).....	246
Figure 14-37: Optimization History of a response – computed (red points) and predicted (black) values..	247
Figure 14-38: Variable Plot .....	248
Figure 14-39: Tradeoff plot .....	249
Figure 14-40: Parallel Coordinate Plot for Pareto optimal solutions with selected point (purple line).....	250

Figure 14-41: Hyper-radial visualization, equal weights, points colored by variable .....	252
Figure 14-42: Selection of position for SOM .....	253
Figure 14-43: Self-Organizing Map, component plots of objectives and distance measure .....	253
Figure 14-44: Histogram constructed using metamodel together with the statistical distribution of the variables .....	254
Figure 14-45: Mean value and standard deviation constructed using metamodel together with the statistical distribution of the variables .....	255
Figure 14-46: Probability of Composite Disp > 1 with 95% confidence interval in red constructed using metamodel together with the statistical distribution of the variables .....	256
Figure 14-47: Correlation Bars evaluated on metamodel .....	257
Figure 14-48: Stochastic Contribution plot.....	258
Figure 15-1: Visualization of DYNA results statistics. After plot creation using the wizard, the plot data must be generated by running LS-OPT. The plot can then be displayed in LS-PREPOST. Existing plots can be edited, deleted or investigated for a bifurcation.....	261
Figure 15-2: First step of DynaStats plot definition creation; selection of plot type.....	261
Figure 15-3: Second step of DynaStats plot creation; selection of component or history.....	262
Figure 15-4: Third step of DynaStats plot definition creation.....	264
Figure 15-5: 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 overlaid in order to identify bifurcations as described in Section 15.8.....	266
Figure 15-6: 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.....	268
Figure 15-7: Metamodels can be used to distinguish between changes in the results due to the design variable changes and changes due to bifurcations. ....	268
Figure 15-8 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.....	270
Figure 15-9: 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 can also be viewed... ..	271
Figure 15-10: Viewing the stochastic contribution of a single variable. ....	272
Figure 15-11: The safety margin is the difference, measured in standard deviations, between the mean response and the constraint bound on the response. ....	273
Figure 15-12: Plotting a safety margin or the probability of failure requires that the bound must be specified. ....	274
Figure 15-13: Viewing all the LS-OPT histories.....	275
Figure 15-14: Statistics of an LS-OPT history. ....	276
Figure 15-15: The LS-OPT histories of all the LS-DYNA run can be viewed simultaneously. ....	276
Figure 15-16: 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. ....	277
Figure 15-17: Options to create Bifurcation Plot for an existing plot. ....	277
Figure 15-18: 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.....	279

Figure 15-19: 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. .... 280

Figure 15-20: The displacement magnitude can depend on the alignment 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. .... 280

Figure 15-21: 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. .... 282

Figure 15-22: Interpolation of metal forming results. .... 282

Figure 15-23: Accuracy 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 occur 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. .... 283

Figure 16-1: Optimization history for variables and objective ..... 287

Figure 16-2: Global Sensitivities ..... 287

Figure 16-3: Target and computed curves, only optimal curves are displayed for all iterations..... 288

Figure 16-4: Accuracy plot; computed vs. predicted values; error measures are displayed in the title ..... 290

Figure 16-5: ANOVA values for a single response; Sobol values for multiple responses ..... 291

Figure 16-6: Interpolator plot: 2D surface plots for variables vs. responses; constraints on the metamodel and the predicted value for the selected parameter combination are displayed..... 292

Figure 16-7: Possible setup for a shape optimization. a\_pre interfaces with a preprocessor that generates the geometry of the model depending on parameters. .... 294

Figure 16-8: Parameter definition for a worst-case design optimization..... 295

Figure 17-1: The two-bar truss example ..... 297

Figure 17-2: Stage dialog Setup for a user-defined solver. Parameters are specified in the input file using the LS-OPT parameter format <<>>. .... 299

Figure 17-3: User-defined response definitions..... 300

Figure 17-4: Task dialog; Selection for a metamodel base optimization using a single iteration. .... 301

Figure 17-5: Parameter Setup; specification of design space and starting values or all parameters ..... 301

Figure 17-6: Sampling and Metamodel; Select metamodel type Polynomial with order Linear; use the defaults for Point Selection and number of points..... 302

Figure 17-7: Objectives; select the previously defined response Weight from the list on the right..... 302

Figure 17-8: Constraints; select the respective responses from the list on the right and specify lower and upper bounds. .... 303

Figure 17-9: Accuracy of linear metamodel for response "Weight" ..... 304

Figure 17-10: Accuracy of linear metamodel of responses "StressL" and "StressR" ..... 304

Figure 17-11: Surface plot for objective function Weight; constraints are displayed on the metamodel. ... 305

Figure 17-12: Sampling dialog settings for a quadratic approximation ..... 306

Figure 17-13: Accuracy of quadratic metamodel for response "Weight"..... 307

Figure 17-14: Accuracy of quadratic metamodel of responses "StressL" and "StressR" ..... 307

Figure 17-15: Surface plot for objective function weight; constraints are displayed on the metamodel. .... 308



Figure 17-16: Reducing the design space by specifying an initial range; the starting values are the optimal values found in the previous approach.....	309
Figure 17-17: Accuracy of quadratic metamodel in reduced design space for response "Weight" .....	309
Figure 17-18: Accuracy of quadratic metamodel in reduced design space of responses "StressL" and "StressR" .....	310
Figure 17-19: Surface plot for objective function weight; constraints are displayed on the metamodel. ....	311
Figure 17-20: Task dialog; select strategy SRSM to automate the process.....	312
Figure 17-21: Termination criteria; select 10 iterations for linear, 5 for quadratic approach .....	312
Figure 17-22: Optimization history of design variables; linear (left) and quadratic (right) .....	315
Figure 17-23: Optimization history of responses; linear (left) and quadratic (right) .....	316
Figure 17-24: Small car impacting a pole.....	317
Figure 17-25: Definition of response histories using standard LS-DYNA interfaces. ....	319
Figure 17-26: Definition of responses using standard LS-DYNA interfaces and expressions.....	320
Figure 17-27: Definition of composite expression using previously defined responses. ....	321
Figure 17-28: Parameter Setup; .....	321
Figure 17-29: Sampling dialog; Select metamodel RBF, increase the number of points to 20.....	322
Figure 17-30: Computed vs. predicted responses – RBF approximation .....	323
Figure 17-31: Sensitivities plots; ANOVA with 95% confidence interval (top) and GSA (bottom) .....	323
Figure 17-32: Surface plot for objective function HIC with predicted and computed optimum, simulation points and residuals; constraints are displayed on the surface.....	324
Figure 17-33: History plot for Acceleration; the curves are color-coded using the value of the variable hood. ....	325
Figure 17-34: Task dialog; select Strategy Sequential with Domain Reduction .....	326
Figure 17-35: Sampling Dialog; use the default settings for SRSM for metamodel type and order, point selection scheme and number of points .....	327
Figure 17-36: Termination Criteria dialog; select the maximum number of iterations .....	327
Figure 17-37: Optimization history of HIC and Intrusion .....	328
Figure 17-38: Optimization history of design variables .....	329
Figure 17-39: Parameter Setup dialog; Definition of a discrete variable. ....	329
Figure 17-40: Optimization history of HIC and Intrusion for mixed-discrete optimization .....	330
Figure 17-41: Mixed-discrete variable histories. ....	331
Figure 17-42: Task dialog; Direct Genetic algorithm.....	332
Figure 17-43: Optimization dialog; Specification of advanced GA options .....	332
Figure 17-44: Optimization history of mixed-discrete variable optimization using direct GA simulation..	333
Figure 17-45: Optimization history of HIC and Intrusion .....	333
Figure 17-46: Impacting cylinder .....	334
Figure 17-47: Deformed finite element model (time = 20ms).....	335
Figure 17-48: Prediction accuracy of Internal Energy and Rigid Wall Force (One Quadratic iteration).....	339
Figure 17-49: Prediction accuracy of Internal Energy and Rigid Wall Force (One Quadratic iteration).....	341
Figure 17-50: Optimization history of automated design (filtered force).....	345
Figure 17-51: Cylinder: Constrained rigid wall force: $F(t) < 80000$ (SAE 300Hz filtered).....	346
Figure 17-52: Parameterization of cross-section .....	347
Figure 17-53: Quarter segment of FE model: tools and blank.....	347
Figure 17-54: Optimization history of design variables and responses (automated design) .....	357
Figure 17-55: FLD diagrams of baseline and 10 <sup>th</sup> iteration.....	358
Figure 17-56: Deformed state .....	359

Figure 17-57: Crash model of vehicle showing road and wall a) Undeformed b) Deformed (78ms).....	360
Figure 17-58: Structural components affected by design variables – a) Undeformed and (b) deformed (time = 72ms).....	360
Figure 17-59: Body-in-white model of vehicle in torsional vibration mode (38.7Hz).....	360
Figure 17-60: Main LS-OPT GUI; Metamodel based optimization; two disciplines.....	362
Figure 17-61: Task dialog; Calculating Pareto Optimal solutions using a metamodel-base method using sequential strategy.....	363
Figure 17-62: Sampling dialog; use Radial Basis Functions to get a global approximation.....	363
Figure 17-63: Lookup function; evaluate the value of t for a specified value of the history XDISP.....	364
Figure 17-64: Response Expression; The stage pulses are calculated using the Integral function.....	364
Figure 17-65: Frequency extraction with Mode Tracking.....	365
Figure 17-66: Objectives panel; Select Multi-Objective Mode to create Pareto Optimal Front.....	365
Figure 17-67: Constraints panel; Constraints are scaled using the target values. This is the default.....	366
Figure 17-68: Verification Run; 100 verification runs are performed using results of the Pareto Optimal Front.....	366
Figure 17-69: Pareto optimal front. Comparison of predicted results (left) and verification runs (right)....	367
Figure 17-70: Verification runs color-coded by maximal constraint violation.....	367
Figure 17-71: Self Organizing Maps plot of predicted Pareto optimal solutions.....	368
Figure 17-72: Parallel Coordinate Plot; Predicted Pareto Optimal solutions (top) and verification runs (bottom).....	368
Figure 17-73: Task dialog; Direct genetic algorithm.....	369
Figure 17-74: Options for Genetic Algorithm.....	369
Figure 17-75: Tradeoffs between scaled mass and intrusion (displacement).....	370
Figure 17-76: Self-organizing maps plot of Pareto optimal solutions; results of last generation.....	371
Figure 17-77: Parallel coordinate plot of Pareto optimal solutions; results of last generation.....	371
Figure 17-78: Typical instrument panel prepared for a "Bendix" component test.....	373
Figure 17-79: Typical major components of a knee bolster system and definition of design variables.....	373
Figure 17-80: Constraints for the knee bolster design problem.....	374
Figure 17-81: ANOVA plots for objectives and constraints of knee-bolster design problem.....	378
Figure 17-82: Global sensitivity analysis of objectives and constraints.....	379
Figure 17-83: Optimization history of objectives and maximum constraint violations.....	380
Figure 17-84: Rail with embosses.....	385
Figure 17-85: Main LS-OPT GUI window; Metamodel based design optimization, Strategy SRSM; Optimization of a process chain with 3 stages.....	386
Figure 17-86: Stage dialog interfacing with ANSA.....	387
Figure 17-87: Parameter Setup; variables, values and ranges are imported from the ANSA DV file.....	387
Figure 17-88: File Transfer dialog; the ANSA output file rail.key is copied to the respective LS-DYNA run directory.....	388
Figure 17-89: Stage dialog interfacing with $\mu$ ETApost. $\mu$ ETApost is run in the respective LS-DYNA run directory, since the results are extracted from the LS-DYNA output.....	388
Figure 17-90: Optimization History of variables.....	389
Figure 17-91: Optimization History of objective and constraints.....	390
Figure 17-92: Global Sensitivities.....	390
Figure 17-93: Final design. Optimum of iteration 10.....	391
Figure 17-94: Objective and constraint functions.....	392
Figure 17-95: Defining a user-defined solver.....	394

Figure 17-96: User-defined results extraction. ....	395
Figure 17-97: Sampling definition for optimization using analytical sensitivities.....	396
Figure 17-98: Optimization history for variables .....	397
Figure 17-99: Optimization history for objective .....	397
Figure 17-100: Optimization history for constraints .....	398
Figure 18-1: Sample of elastoplastic material subjected to a controlled vertical displacement .....	399
Figure 18-2: LS-OPT main GUI window .....	401
Figure 18-3: Crossplot definition of force vs. displacement.....	401
Figure 18-4: File History definition. This dialog is accessible from the Histories tab of the Stage dialog or the Curve Matching composite dialog. ....	402
Figure 18-5: Definition of Mean Square Error composite.....	403
Figure 18-6: Evaluation of simulation curves at target t values .....	404
Figure 18-7: Definition of equality constraints using the Standard Composite type MSE. ....	405
Figure 18-8: Optimization History for YMod and Yield.....	406
Figure 18-9: Parameter values of optimal point in normalized design space with 95% confidence interval. ....	406
Figure 18-10: Global Sensitivities for MSE1 and MSE2 .....	406
Figure 18-11: Optimization history of MSE 1 and MSE2. Both objectives decrease, and the accuracy of the metamodel improves over the iterations. ....	407
Figure 18-12: Comparison of optimal force-displacement curves and test data. The simulation curves are colored by iteration .....	408
Figure 18-13: Optimization History for YMod and Yield.....	409
Figure 18-14: Parameter values of optimal point in normalized design space with 95% confidence interval. ....	409
Figure 18-15: Global Sensitivities for all forces and MSE.....	410
Figure 18-16: Optimization history of MSE. The objectives decrease, and the accuracy of the metamodel improves over the iterations.....	410
Figure 18-17: History definitions. Extract stress and strain using the LS-DYNA d3plot interface. Use the Crossplot interface to generate the stress vs. strain curve .....	411
Figure 18-18: Define a curve matching composite for each load case.....	412
Figure 18-19: Definition of the five objective components.....	413
Figure 18-20: Optimization History of objective components for each of the five load cases as well as the multi-objective .....	413
Figure 18-21: Global Sensitivities plot of all objective components.....	414
Figure 18-22: 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 bottom right shows the omparison of the final optimal simulation curves and target curves for all load cases. ....	414
Figure 19-1: Tube impact.....	416
Figure 19-2: Assigning statistical distributions to noise variables .....	418
Figure 19-3: Definition of statistical distributions. This dialog is accessible from the Setup dialog's Parameter Setup tab. ....	418
Figure 19-4: Histogram plots of variables and responses. Mean values and standard deviations are displayed in the titles.....	419
Figure 19-5: Probability of NodDisp < -150 with 95% confidence intervals.....	419
Figure 19-6: Assigning statistical distributions to control variables .....	420

Figure 19-7: Accuracy plot. Computed vs. predicted values; error measures are displayed in the title.....	421
Figure 19-8: 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. ....	422
Figure 19-9: Probability of NodDisp < -150 with 95% confidence intervals evaluated on the metamodel using 10000 points. ....	422
Figure 19-10 Tube Buckling.....	423
Figure 19-11: Plate Buckling Example.....	424
Figure 19-12: Definition of noise variable and distribution .....	424
Figure 19-13: Definition of Latin Hypercube sampling with 5 points. ....	425
Figure 19-14: Response Definition for bifurcation identification .....	426
Figure 19-15: 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. ....	427
Figure 19-16: LS-OPT identified and displayed this bifurcation automatically using the GUI setting shown in the previous figure. ....	428
Figure 19-17: Range of z-component displacement .....	428
Figure 19-18: Index of run with maximum z-component displacement.....	429
Figure 19-19: Index of run with minimum z-component displacement .....	429
Figure 19-20: Second buckling mode .....	430
Figure 19-21: Probabilistic variable and constraint definition for RBDO.....	431
Figure 19-22: 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. ....	432
Figure 19-23: Task Reliability based design optimization/Robust Parameter Design .....	433
Figure 19-24: Parameter Setup and Distribution .....	434
Figure 19-25: Definition of Standard deviation composite .....	435
Figure 19-26: Contours of stress response. The flattest part of the response is when variable 'base' equals 0.5.....	436
Figure 19-27: Optimization histories. Design variable 'base' is shown on the left and the standard deviation of the stress response is shown on the right. ....	436
Figure 19-28: 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. ....	437
Figure 19-29: Task dialog. Select the main task Monte Carlo analysis.....	438
Figure 19-30: Parameter setup. Since the variable dummy is not used, an arbitrary distribution can be used here.....	439
Figure 19-31: Sampling dialog. Specify the number of simulation points. The point selection doesn't affect the analysis here, since the variable is not used.....	439
Figure 19-32: Histogram of the responses using only a stochastic field. ....	440
Figure 19-33: Noise variable and distribution definition.....	441
Figure 19-34: Plot of the responses using the thickness variable and a stochastic field. ....	442
Figure 19-35: Parameter Setup. ....	443
Figure 19-36: Sampling dialog. 10 replicate runs are done at each experimental point.....	443
Figure 19-37: Plot of the responses using replicate experiments .....	444
Figure 19-38: Task Metamodel-based Monte Carlo analysis .....	445

Figure 19-39: Parameter Setup. ....	445
Figure 19-40: Sampling dialog. ....	446
Figure 19-41: Plot of the responses using the same five stochastic fields in the replicates. ....	447
Figure 20-1: Six space-filling designs: 5 points in a 2-dimensional box region ....	458
Figure 21-1: Schematic of a neural network with 2 inputs and a hidden layer of 4 neurons with activation function $f$ . ....	468
Figure 21-2: Sigmoid transfer function $y=1/(1+\exp(-x))$ typically used with feedforward networks. ....	469
Figure 21-3: Radial basis transfer function $y=\exp(-x^2)$ ....	469
Figure 21-4: Weighted sum of radial basis transfer functions. Three radial basis functions (dashed lines) are scaled and summed to produce a function (solid line). ....	469
Figure 21-5: A radial basis network approximation (solid line) of the function, which fits the 21 data points (plus symbols). ....	469
Figure 21-6: The same 21 data points as in Figure 21-5. 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. ....	470
Figure 21-7: The same 21 data points as in Figure 21-5. Approximation with overlapping RBF neurons. The spread of RBF units is too high. ....	470
Figure 22-1: Adaptation of subregion in SRSM: (a) pure panning, (b) pure zooming and (c) a combination of panning and zooming. ....	491
Figure 22-2: The sub-region contraction rate $\lambda$ as a function of the oscillation indicator $\hat{c}$ and the absolute move distance $ d $ ....	493
Figure 22-3: Simple genetic algorithm. ....	496
Figure 22-4: Illustration of non-domination criterion, Pareto optimal set, and Pareto optimal front. ....	501
Figure 22-5: 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. ....	502
Figure 22-6: Elitism in NSGA-II. ....	503
Figure 22-7: Trade-off plot shows all four objectives of Pareto optimal solutions. ....	510
Figure 22-8: The Pareto frontier and indifference curves. ....	511
Figure 22-9: Parallel coordinate plot shows objectives and design variables of all points on the Pareto front. ....	513
Figure 22-10: Self organizing maps display design objectives, variables, and constraints on the Pareto front. ....	514
Figure 23-1: Entities in Mean Squared Error formulation. ....	527
Figure 23-2: 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. ....	529
Figure 24-1: 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. ....	541
Figure 24-2: 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. ....	542
Figure 24-3: Outliers are identified after a metamodel has been fitted. Values in poor agreement of what is predicted by the design variables are considered outliers. ....	550
Figure 24-4: 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. ....	554
Figure 24-5: 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..... 554

# 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.

Modern architectures often feature multiple processors and all indications are that the demand for distributed computing will strengthen into the future. This is causing a revolution in computing as single analyses that took a number of days in the recent past can now be done within a few hours. 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.

The present version of 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. This methodology is currently under development and will be available in future versions of LS-OPT.



## 1.1. Overview of the manual

This LS-OPT<sup>®</sup> manual consists of three parts.

*Part I* is the User's Manual, which guides the user in the use of LS-OPT<sub>ui</sub>, the graphical user interface.

*Part II* is the Examples section, where examples are used to illustrate the application of LS-OPT to a variety of practical applications.

*Part III* is the Theoretical Manual in which the fundamentals are provided for the various features in LS-OPT.

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.

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

*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 17 through 19) are included to demonstrate the features and capabilities and can be read together with Chapters 2 to 16 to help the user to set up a problem formulation.

The Theoretical Manual (Chapters 20 through 24) 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 21-5** (just click on any of the afore-mentioned references). **Alt+Left Arrow** returns to the original reference point.

# **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.

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

*Table 2-1: LS-OPT execution commands*

Command	Description
<code>lsoptui command_file_name</code>	Execute the graphical user interface
<code>lsopt command_file_name</code>	LS-OPT batch execution
<code>lsopt env</code>	Check the LS-OPT environment setting. The LS-OPT environment is automatically set to the location of the <code>lsopt</code> executable.
<code>viewer command_file_name</code>	Execute the graphical postprocessor (also accessible from main GUI)
<code>com2lsopt com.abcde abcde.lsopt</code>	Converts a legacy 'com' file to a .lsopt file in XML format

## 2.2. Name conventions in LS-OPT

Entities such as variables, responses, etc. 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 any of the following characters: `_ .` *Spaces are not allowed.*

For entities that can not be used in mathematical expressions, i.e. `stage` and `sampling`, the name can contain the characters `-+%=` as well.

For entities that can be used in mathematical expressions, i.e. `variable`, `history`, `response`, `composite` and `filehistory`, the leading character must be alphabetical. Those entities must be given unique names, because mathematical expressions can be constructed using various entities in the same formula.

## 2.3. A modus operandi for design using response surfaces

### 2.3.1. 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 21.3 and 22.5.

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 21.3) 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 17.5 and 17.6).

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 22.5.

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:

### **2.3.2. 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 14.3.4).
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 21.1.3) 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 24.8. 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.

## 2.4. 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.

## 2.5. 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.

### 2.5.1. 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 22.8) can be used.

### 2.5.2. 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.

### 2.5.3. 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 20.2.8).

### 2.5.4. 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’.

### 2.5.5. 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 18.1). In this case the response functions are insensitive to  $E$  and will show a very high confidence interval for  $E$  (Section 18.1.4).

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.

## 2.6. Setup of a simple optimization problem

### 2.6.1. 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.

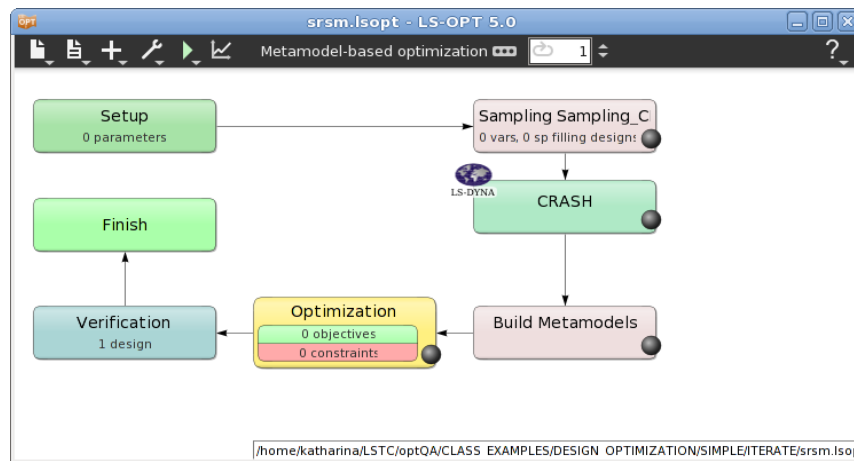
### 2.6.2. Startup

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.





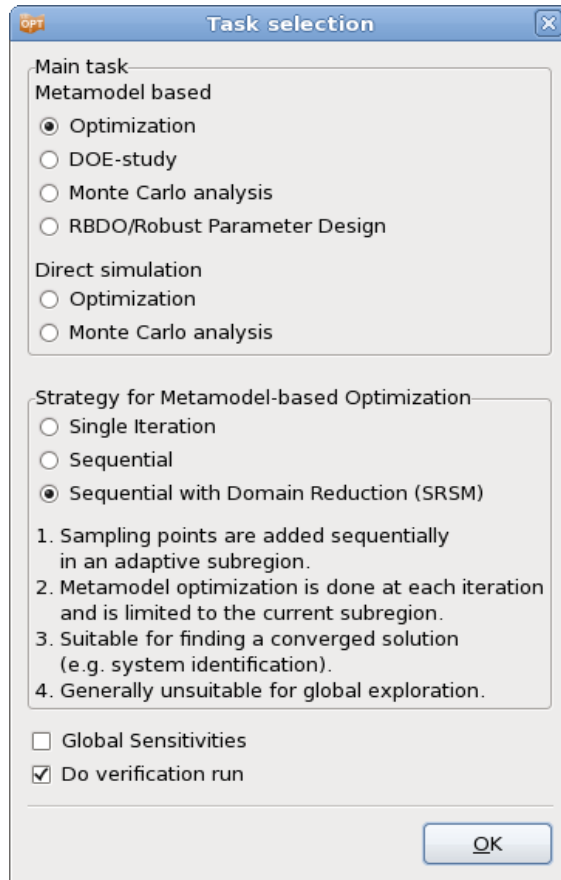
*Figure 2-1: LS-OPT Startup dialog. Select the working directory, enter a name for the LS-OPT project file and a name for the initial sampling and initial stage to generate a new project.*



*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.*

### 2.6.3. 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.



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

#### 2.6.4. 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.

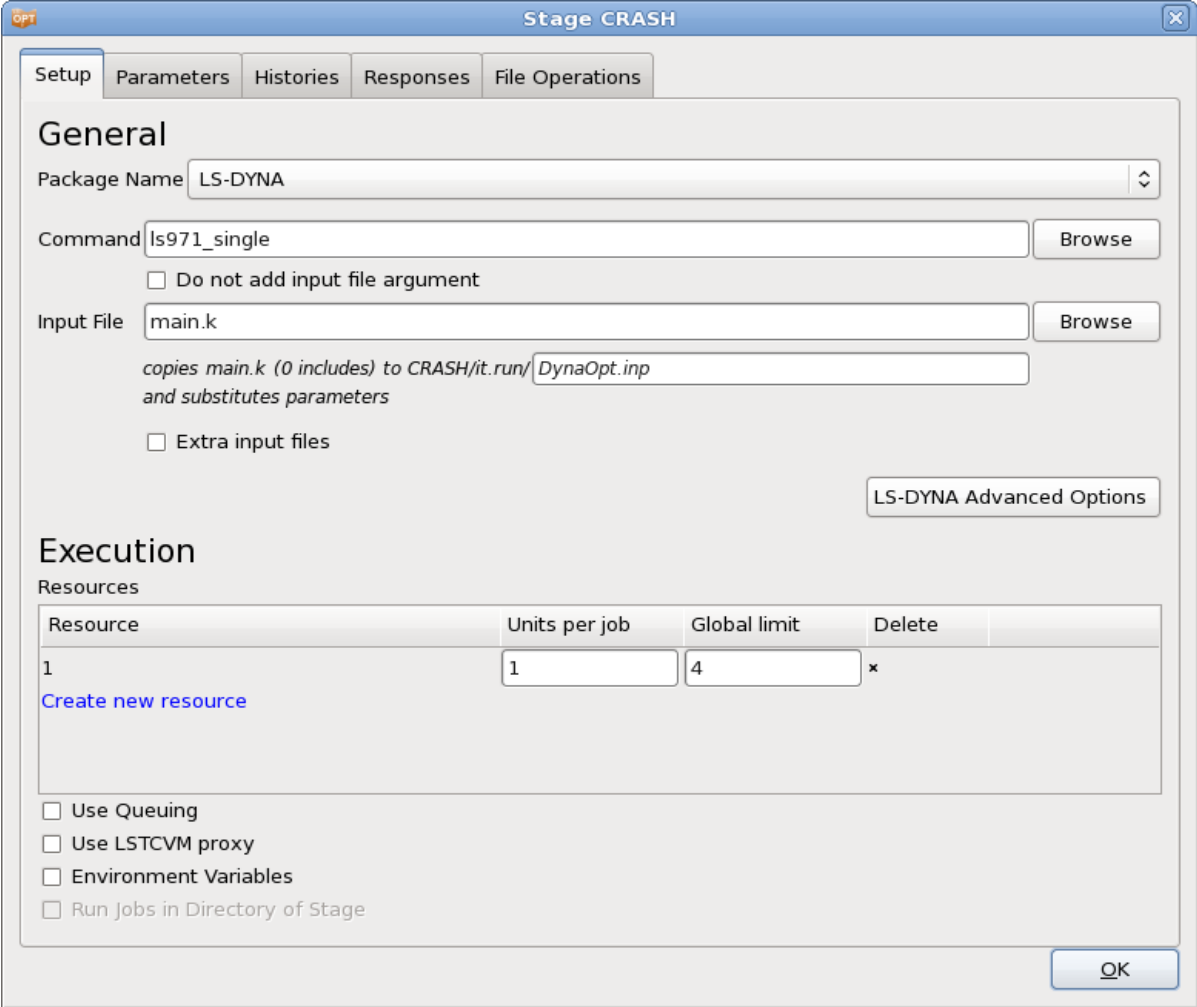


Figure 2-4: Stage dialog - Setup. Select the solver package name, the command and the solver input file

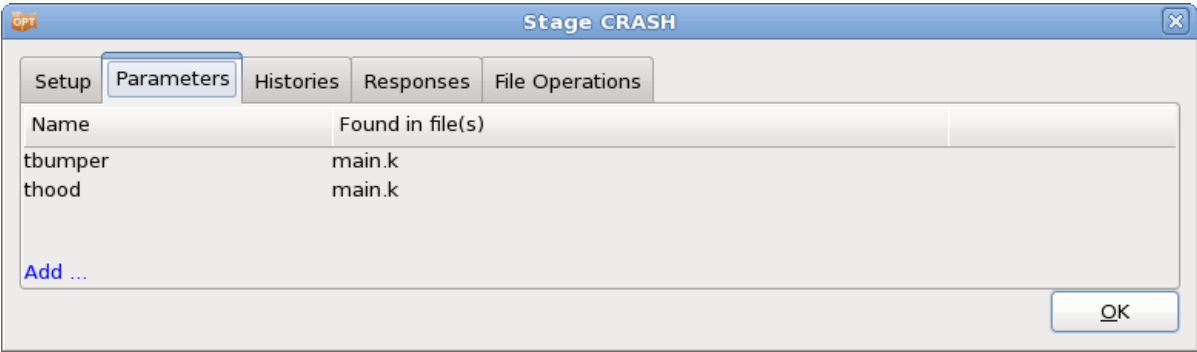
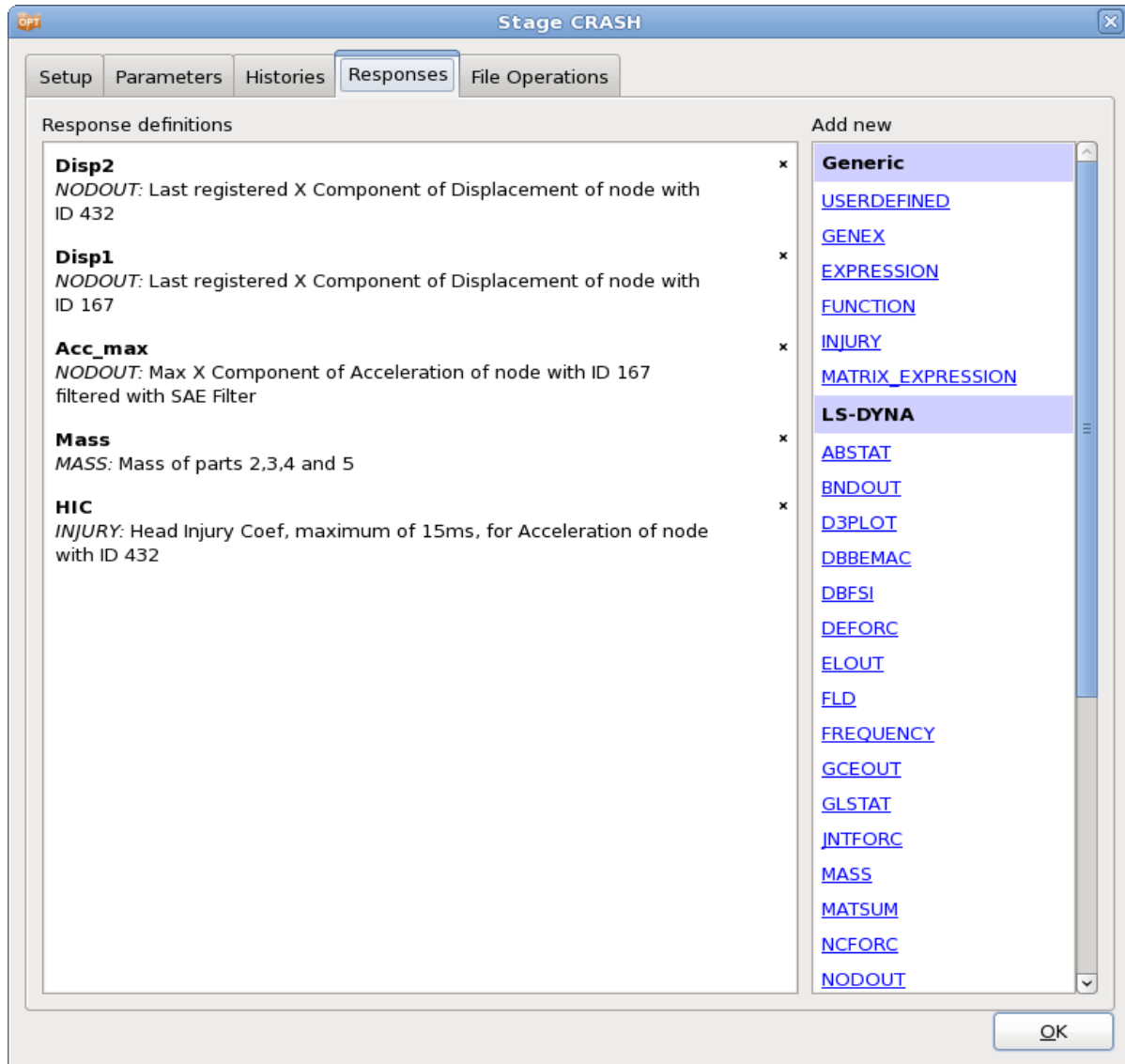


Figure 2-5: Stage dialog – Parameters. Displays the parameters found in the input file specified in Setup



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

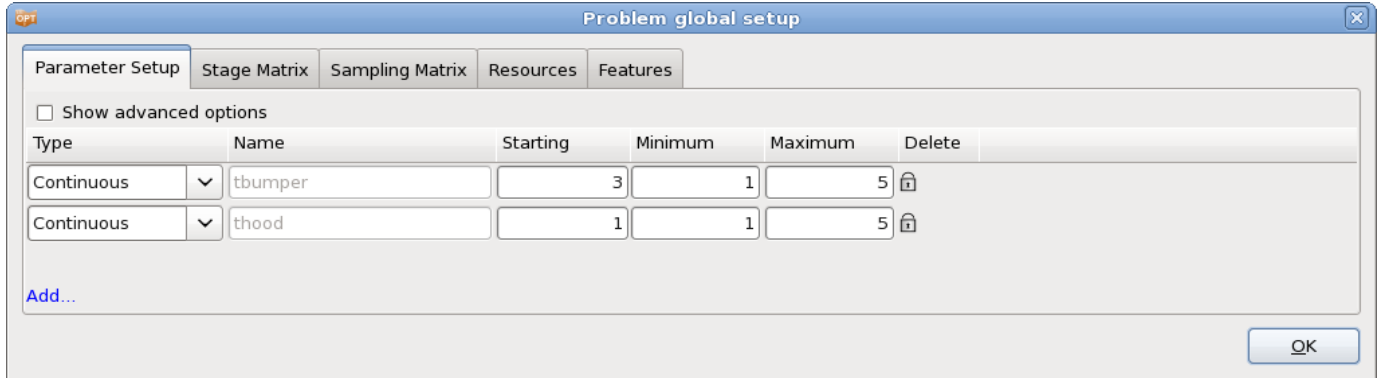
### 2.6.5. Setup

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

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.

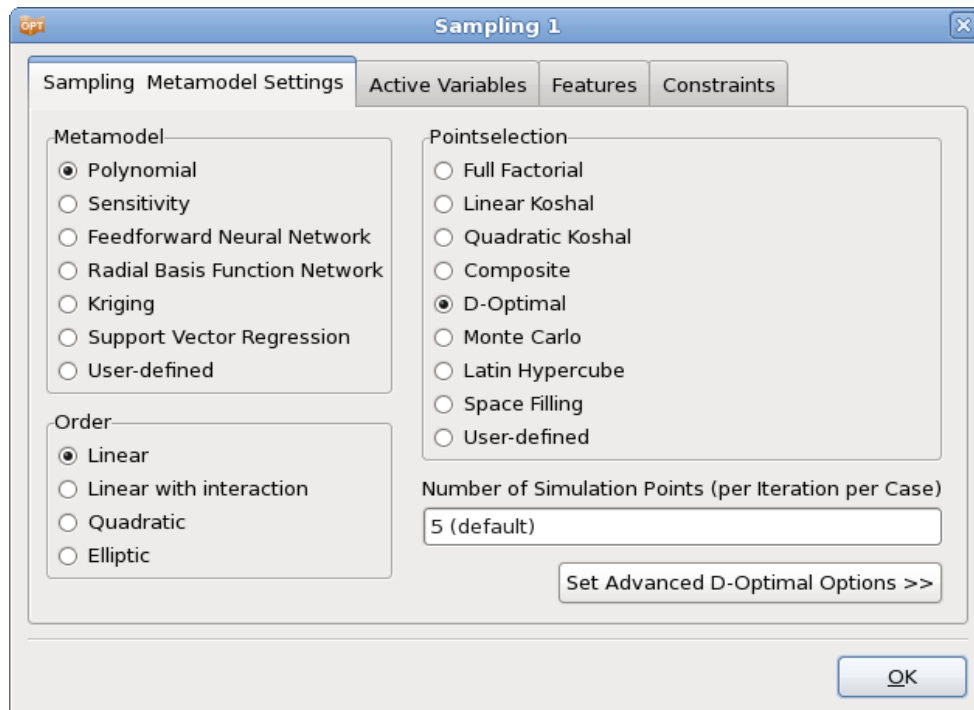


*Figure 2-7: Parameter Setup dialog. Define the parameter type and required values.*

### 2.6.6. Sampling and Metamodels

Select the **Sampling** box, Chapter 8. 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.

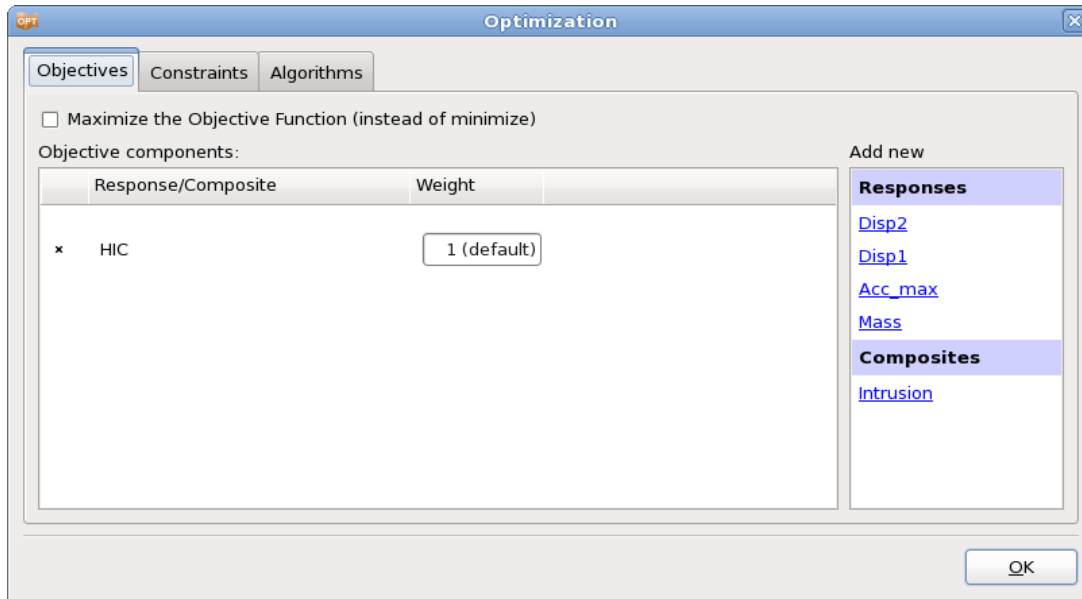


*Figure 2-8: Sampling dialog. Select the metamodel type and point selection scheme.*

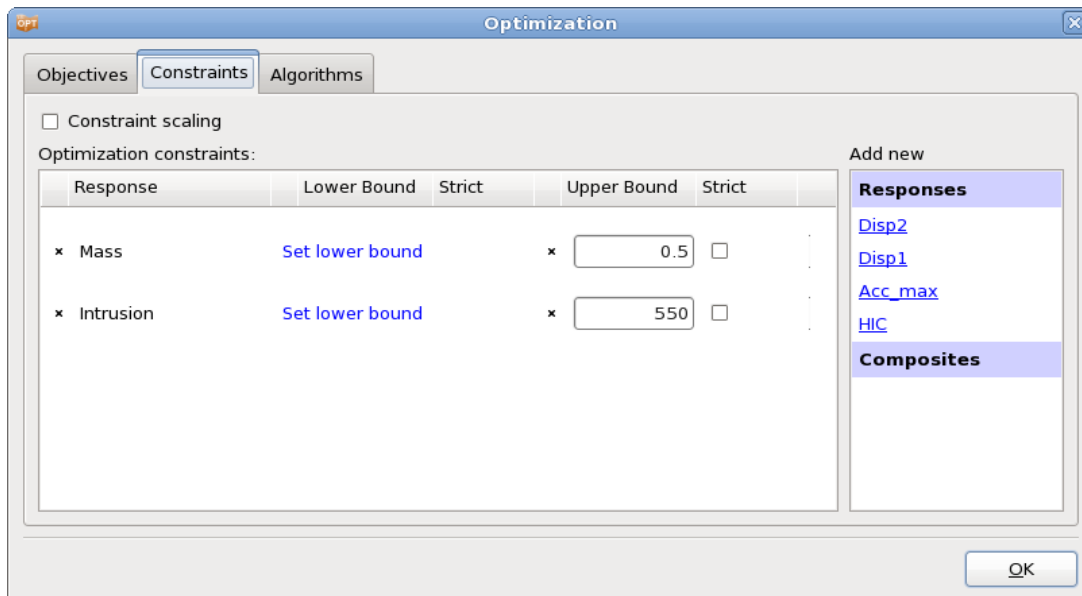
### 2.6.7. Optimization

Select the **Optimization** box, Chapter 10. 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.



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



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

### 2.6.8. Termination criteria

Select the Termination criteria box, Chapter 11. Specify the *Maximum number of Iterations*, e.g. 5 iterations. Use the default values for the other options.

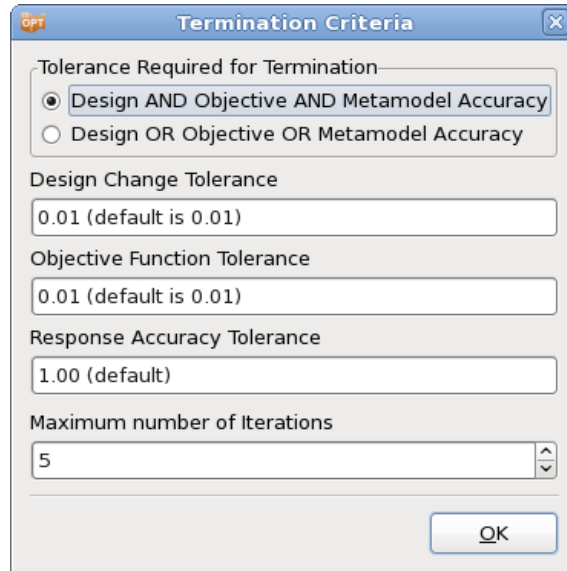



Figure 2-11: Termination Criteria dialog. Specify the maximum number of iterations

### 2.6.9. 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.

### 2.6.10. Viewer

Use the Viewer (Chapter 14) to evaluate the results by selecting  from the the main GUI window control bar. The Viewer provides features to display metamodels and plot simulation results and optimization progress.

## 2.7. REFERENCES

- [1] Stander, N. Goel, T. Metamodel sensitivity to sequential sampling strategies in crashworthiness design. In *Proceedings of the 12<sup>th</sup> AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada, Sep 10-12, 2008*.

# 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 (see Figure 3-2).

Legacy `com.abcde` files generated with previous LS-OPT versions 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
```



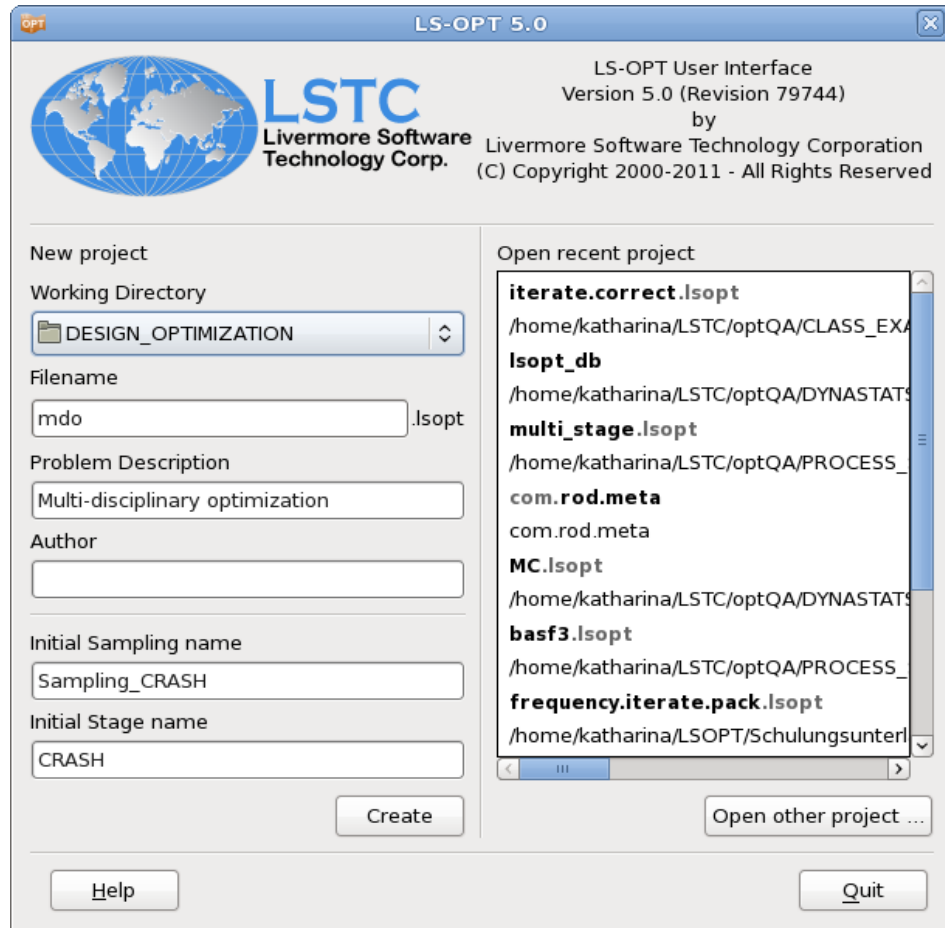


Figure 3-1: Startup Dialog of Isoptui

**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 <code>.lsopt</code> is automatically appended to the selected name.	
Problem Description	A description of the problem can be given. This description is echoed in the <code>lsopt_input</code> and <code>lsopt_output</code> 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 <i>Sampling</i> definition. The name of the first sampling has to be specified here. A default name is provided.	Chapter 8
Initial Stage name	Each LS-OPT project requires at least one <i>Stage</i> 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
Open other project ...	Option to open any existing LS-OPT project	Section 3.2
Quit	Quit <code>lsoptui</code>	

## 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 meta-models, the metamodel optimization, checking of convergence, and domain reduction in one or more loops, and finally the verification run for a meta-model based, sequential optimization. Refer to Chapter 4 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.

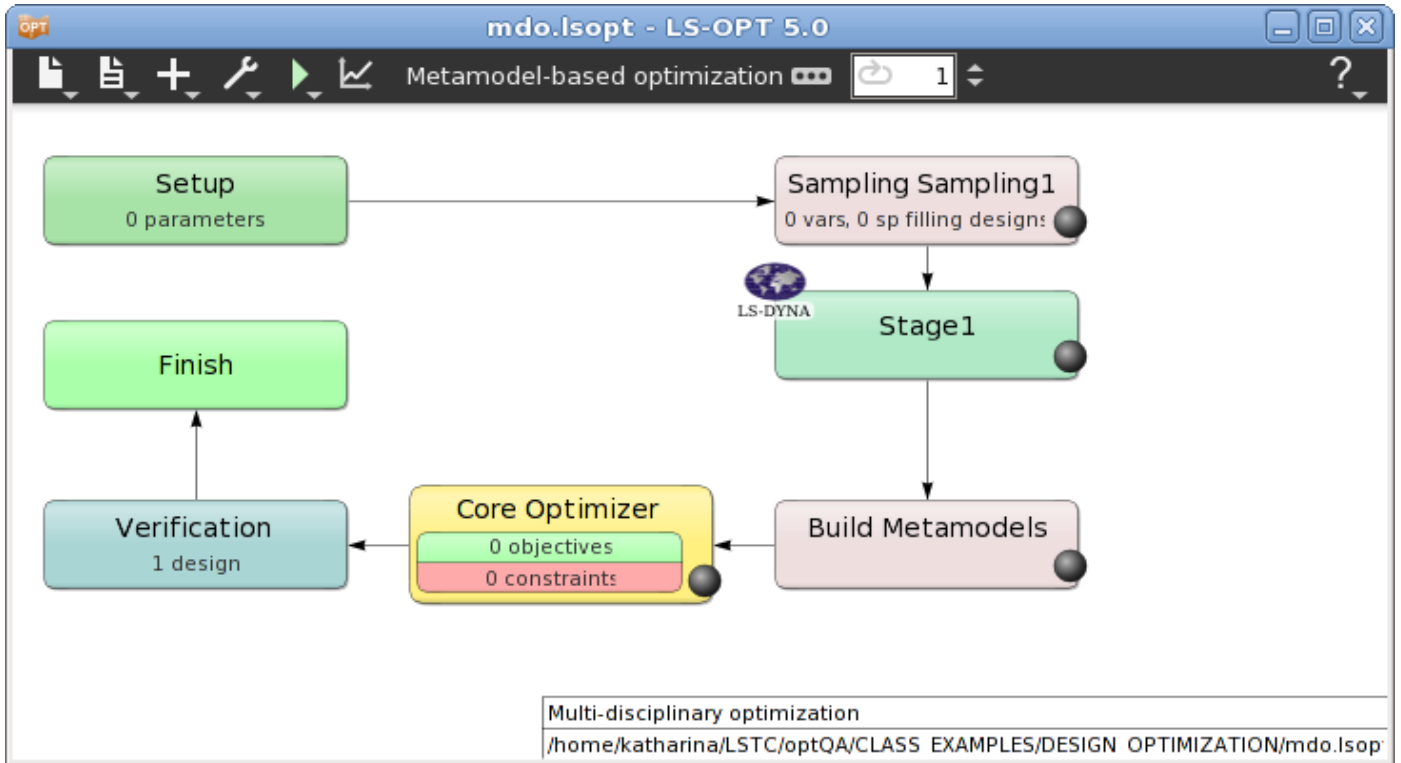












Figure 3-2: Main LS-OPT GUI window for a setup of a Metamodel-base optimization

**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 ...	
	Exit	Exit <code>lsoptui</code>	
	Input	Open the <code>lsopt_input</code> file	
	Output	Open the <code>lsopt_output</code> file	
	Summary Report	Open the <code>lsopt_report</code> file	
	Warnings	Open the <code>WARNING_MESSAGE</code> file	
	Errors	Open the <code>EXIT_STATUS</code> file	
	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 <code>Experiments_n.csv</code> and <code>AnalysisResults_n.lsox</code> .	Chapter 8
	Add Stage in <i>Sampling</i>	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
	Add Composite	Add Composite	Chapter 9
	Add Domain Reduction	Use Domain Reduction (same as <i>Sequential with Domain Reduction</i> option in Task dialog)	Section 4.8
	Add Termination Criteria	Switch to sequential Strategy	Chapter 11

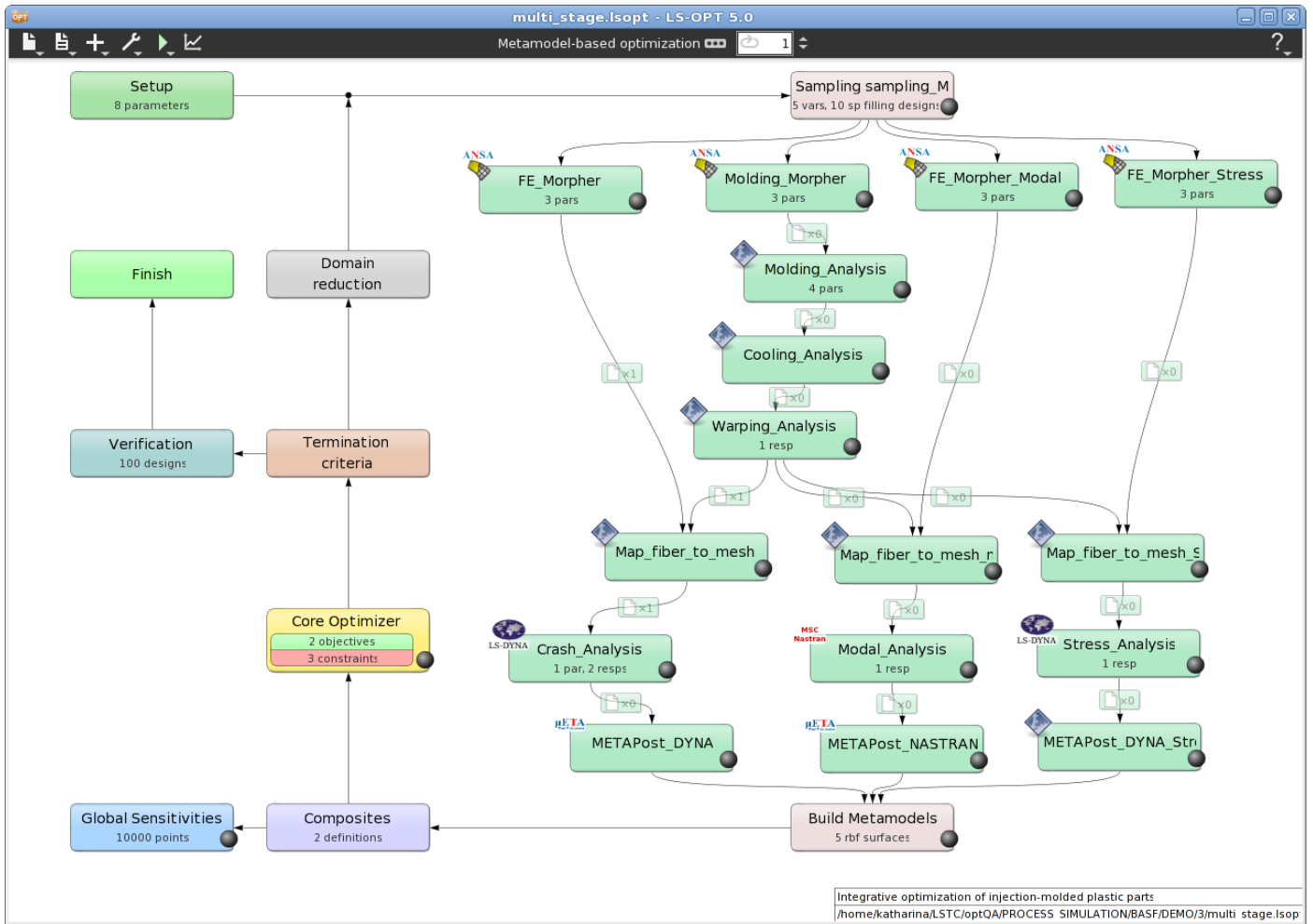
	Add Verification Run	Run an additional simulation using the parameter values of the predicted optimum or Pareto optimal solutions at the end of the optimization run.	Section 4.11
	Add Global Sensitivities	Calculates Global Sensitivities on the meta-model.	Section 4.10
	Re-layout stages	Layout the stage 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 All: The directory structure created by LS-OPT and all the files in this directory structure are deleted.  Clean from current iteration [ <i>iter</i> ]: Removes all simulation data as well as optimization data from the specified iteration <i>iter</i> onwards.	Section 3.4
	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
	DynaStats	Opens DynaStats	Chapter 15
	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 14
	Task	Opens Task Dialog.	Chapter 4
	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
	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 7
Sampling	Point selection and metamodel settings	Chapter 8
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 8
Composites	Define composites	Chapter 9
Global Sensitivities	Calculate global sensitivities	Section 4.10
Optimization	Definition of objectives, constraints and optimization algorithms	Chapter 10
Monte Carlo	Monte Carlo settings	Section 10.5
Termination Criteria	Termination criteria for sequential strategies	Chapter 11
Domain Reduction	Domain reduction settings for strategy sequential with domain reduction	Section 4.8
Verification Run	Perform (specified number of) verification run(s)	Section 4.11

### 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. See Figure 3-3.



**Figure 3-3: Setup of a complex optimization problem**

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.

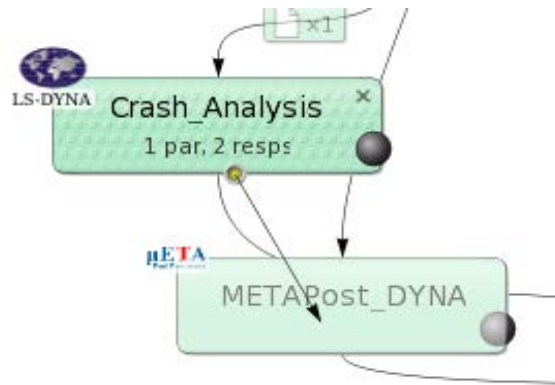
On creating a new optimization project, a first stage is generated. Additional stages can be added using the *Add stage* option of the + menu in the control bar. A sampling has to be selected to which the new stage is assigned. By default, the new stage is added in parallel to the already existing stages.

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.

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.

- A connection will be created between the two boxes.



**Figure 3-4: Creating stage dependencies**

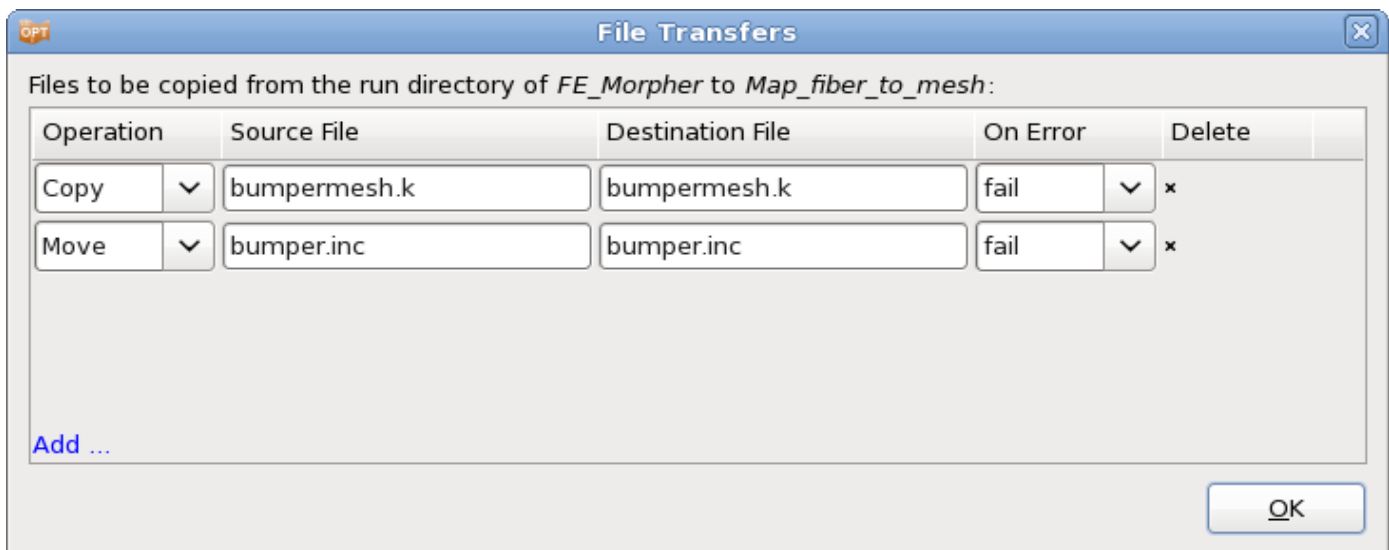
Connections can be deleted using the small icon located on the connection line. This icon also allows the definition of inter-stage file operations, Section 3.2.2.

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. Stages can then be assigned to the relevant samplings.

### 3.2.2. File Transfers between Stages



**Figure 3-5: 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 3-5 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 Link Copy Recursive	Available operations
Source File		Name of source file
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 (See Section 3.4).

### 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 7.1. 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 – 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 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.

### 3.4.1. 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 from current iteration [2]* 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 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.

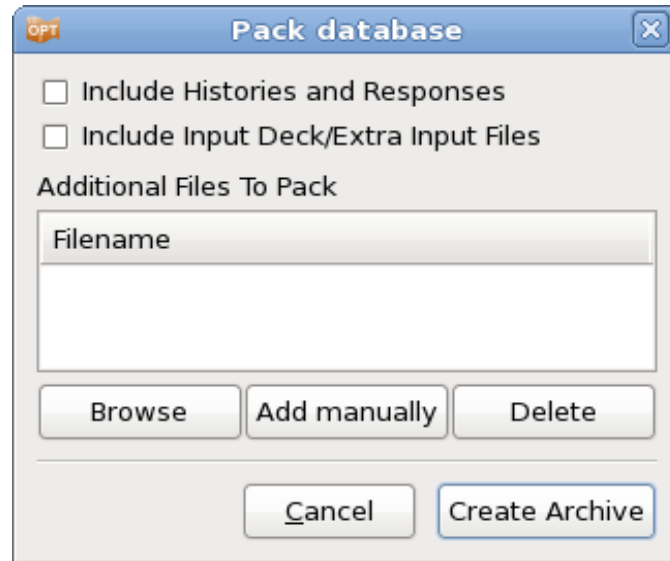
- *Read points.* The `Experiments_n.csv` file is reconstructed from the data in the `XPoint` database files in the run directories.
- *Import results.* Import results from a `.csv` (comma separated variables) file (see Section 8.5.3).
- *Run Jobs.* The stage jobs will be scheduled. Designs previously analyzed will not be analyzed again.
- *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.
- *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 8.5.2) or create a table (`.csv` file) with response values interpolated from a metamodel (Section 7.5.1).
- *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.



**Figure 3-6: Dialog to specify options for archive 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**

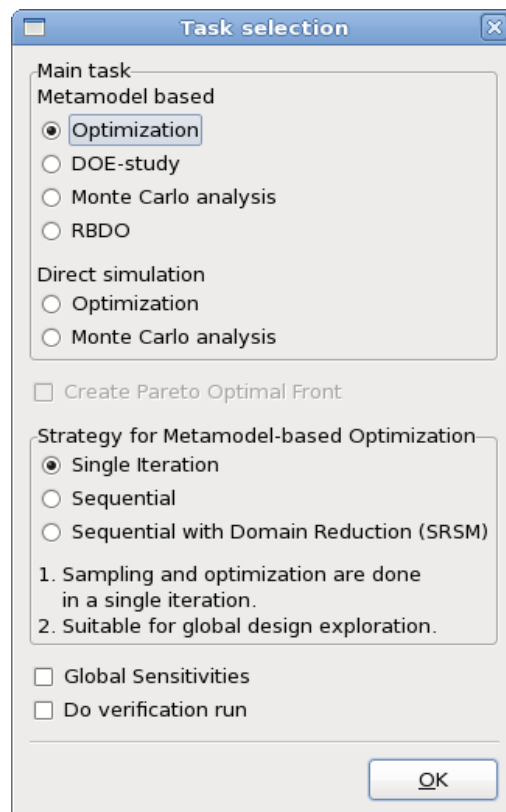
Option	Description
Include Histories and Responses	Also gather the history and response files residing in the run directories. The file produced is <i>lsopack_h.tar.gz</i> ( <i>lsopack_h.zip</i> in Windows). History and response files are only required for the use of DynaStats.
Include Input Deck/Extra 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 <i>lsopack_i.tar.gz</i> ( <i>lsopack_i.zip</i> in Windows).
Additional Files to Pack	List of additional files to pack. Files may be added by browsing or manually.

# 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 and, for an optimization task, the optimization strategy. The two basic branches are Metamodel-based and Direct optimization 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.



*Figure 4-1: Task and Strategy selection*

**Table 4-1: Task selection options**

Option		Description	Reference
Metamodel based	(1) Optimization	Optimization using meta-models	Section 4.2
	(2) DOE-study	DOE study using meta-models	Section 4.3
	(3) Monte Carlo analysis	Monte Carlo analysis using meta-models	Section 4.5.2
	(4) RBDO	Reliability based design optimization using meta-models	Section 4.6
Direct simulation	(5) Optimization	Direct optimization using the Genetic Algorithm	Section 4.4
	(6) Monte Carlo analysis	Direct Monte Carlo analysis	Section 4.5.1
Strategy for Metamodel based optimization (Available for Main Task 1 and 4)	Single Iteration	Sampling and optimization are done in a single iteration. Suitable for global design exploration.	Section 4.7.1
	Sequential	Sampling points are added sequentially in the full design space. Suitable for global design exploration.	Section 4.7.2
	Sequential with Domain Reduction	Sampling points are added sequentially in an adaptive subregion. Metamodel optimization is done at each iteration and is limited to the current subregion. Suitable for finding a converged solution. Generally unsuitable for global exploration.	Section 4.7.3
Available for Main Task 1, 2, 3, 4.	Global Sensitivities	Option to calculate Global Sensitivities on the metamodel.	Section 4.10
Available for Main Task 1 and 4	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 4.11
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 4.9

## 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

## 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 7.1.4). The point selection scheme for a probabilistic analysis depends on whether it is direct or metamodel-based (Section 12.4, Section 12.5). More specific details about the available probabilistic analysis tasks are provided in Section 12.4 and Section 12.5. 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 12.3). 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 12.3). 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 12.6. 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. Selecting strategies for metamodel-based optimization

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 and obtaining as much information about the design as possible.

There are three available strategies for automating the metamodel-based optimization procedure. These strategies only apply to the tasks *Metamodel-based Optimization* and *RBDO*, 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 Stage,
2. Sequential and
3. Sequential with Domain Reduction (SRSM).

A strategy selection resets the *Sampling Dialog* (a warning is given!) with recommended selections for Metamodel type and Point selection scheme, see Chapter 8 .

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

### 4.7.1. 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.



### 4.7.2. 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 11. The approach has the advantage that the iterative process can be stopped as soon as the metamodels or optimum points have achieved sufficient accuracy.

The default settings for sampling follow below (see Sampling dialog, Chapter 8):

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.  $10 \times 30$  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 metamodel. Both these strategies work better with metamodels other than polynomials because of the flexibility of metamodels such as RBF's to adjust to an arbitrary number of points.

### 4.7.3. Sequential strategy with domain reduction

This approach is the same as that in section 4.7.2 but, in order to accelerate convergence, the domain reduction strategy is used to reduce the size of the subregion. During a particular iteration, the subregion is used to locate new points. This strategy is typically only used for optimization in which the user is only interested in the final optimal point (such as parameter identification) and not in any global exploration of the design.

The default domain reduction approach is SRSM which is the original LS-OPT design automation strategy. It allows the building of a new response surface (typically linear polynomial) in each iteration. The size of the subregion is automatically adjusted for each iteration (see Section 22.6) and points belonging to previous iterations are ignored. This method is only suitable for convergence to an optimum, it cannot be used to construct a Pareto Optimal Front, since this needs a global approximation, and is not recommended for any other type of design exploration. The method is ideal for system identification (see Section 23.3).

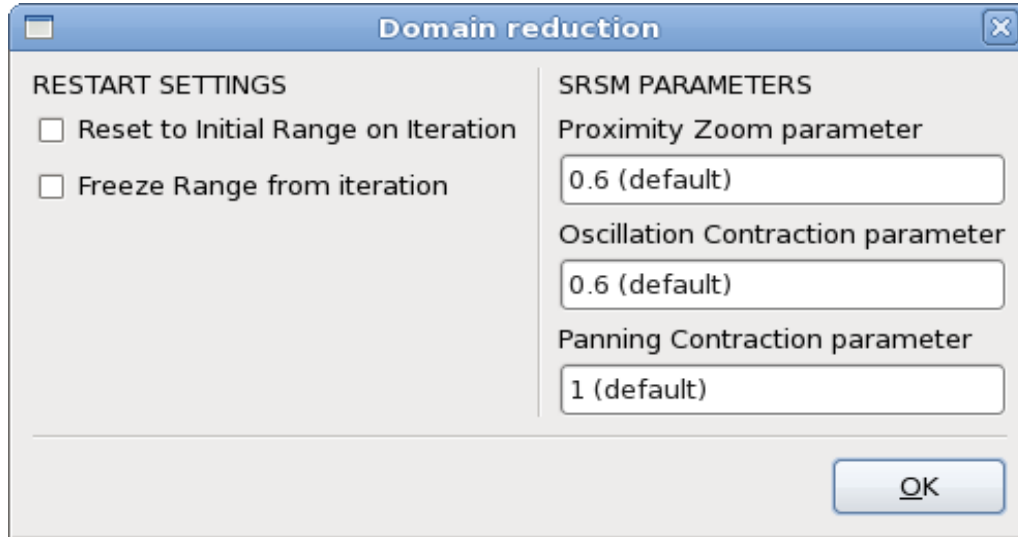
The default settings for sampling are listed below (see *Sampling* dialog, Chapter 8):

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

## 4.8. Domain reduction in metamodel-based optimization

*The Domain reduction dialog is displayed in Figure 4-2.*

Table 4-2 describes the options.



**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 4.8.1
Freeze Range from Iteration	Freeze the subdomain range from a specified iteration	Section 4.8.1
Panning Contraction parameter	$\gamma_{pan}$	Section 4.8.2
Oscillation Contraction parameter	$\gamma_{osc}$	Section 4.8.2
Proximity Zoom parameter	Zoom parameter $\eta$	Section 4.8.2

### 4.8.1. 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. Full adaptivity will be applied 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).

Adaptivity will be applied 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.

### 4.8.2. 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 22.6).

The following parameters can be adjusted in the GUI, Figure 4-2. The options are described in Table 4-2 (refer also to Section 22.6). A suitable default has been provided for each parameter and the user should not find it necessary to change any of these parameters.

## 4.9. 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 22.9.2. If a metamodel-based method is used, selection limits the available strategy options to the *global* strategies *Single Stage* and *Sequential*, see Section 4.7.2 and 4.7.3, and resets the optimization algorithm used on the metamodel to Genetic Algorithm, because this is the only algorithm that has the capability to calculate Pareto optimal solutions.

## 4.10. Global sensitivity analysis

While the ANOVA (Analysis of Variance, Section 20.4) 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 in the 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 the Sobol's method is described in Section 24.7.2. The global sensitivity analysis is carried by selecting the appropriate flag (*Global Sensitivities*) in the Task dialog, Figure 4-1. The selection requires the user to specify the number of Monte-Carlo integration points required to compute sensitivities as suggested by Sobol. The sensitivity indices are stored in the Sobol\_GSA.\* XML database file in the *work* directory.

*Remarks:*

1. In LS-OPT, global sensitivities are evaluated on the metamodels, hence the accuracy depends on the quality of the metamodel.

2. The sensitivities are calculated within the subregion bounds of the variables, i.e. the region adapts itself to the “important” region in the case of domain reduction strategy. However, additional sampling constraints are not considered while calculating the sensitivities.
3. The analytical equations are used to compute sensitivities for polynomials and Gaussian radial basis function metamodels.
4. The composite expressions 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. 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.

# 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 MSC-NASTRAN ANSA LS-INGRID LS-PREPOST HyperMorph TrueGrid META Post User-Defined User-Defined Postprocessor	Section 5.3.1 Section 5.3.2 Section 5.3.6 Section 5.3.4 Section 5.3.3 Section 5.3.7 Section 5.3.5 Section 5.3.8 Section 5.3.9 Section 5.3.10
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
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 5.2.3.	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.  LS-DYNA Include files do not have to be specified as they are automatically and recursively searched by LS-OPT when	Section 5.2.2

given the name of the main input file.

Model Database (ANSA)	ANSA binary database file, typically with the extension <i>.ansa</i>	Section 5.3.6
Output File (HyperMorph, $\mu$ ETA)	HyperMorph: nodal output file produced by Templex $\mu$ ETA: output file used for parsing the history and response names	Section 5.3.7 Section 5.3.8
Session file ( $\mu$ ETA)	File containing information about which results to extract	Section 5.3.8
LS-DYNA Advanced Options	Advanced interfacing options for LS-DYNA.	Section 5.3.1

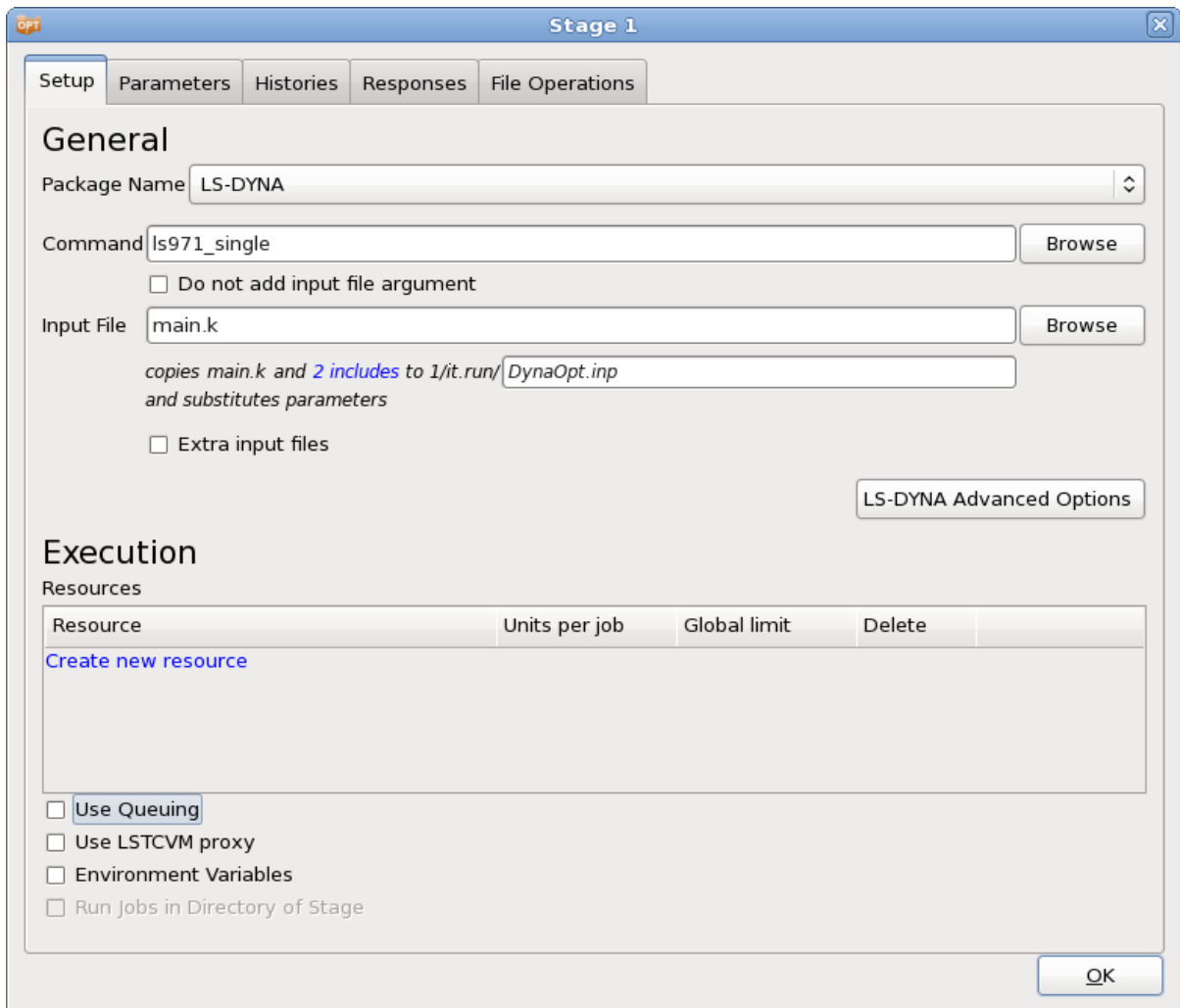


Figure 5-1: Stage dialog Setup panel

### 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 5.8).

*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"`
  - 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. *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.
3. *Linux*: The command name must not be an alias.
4. *Windows*: A path to a program or file cannot contain any blanks or - (dash) symbols.

### 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.

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.



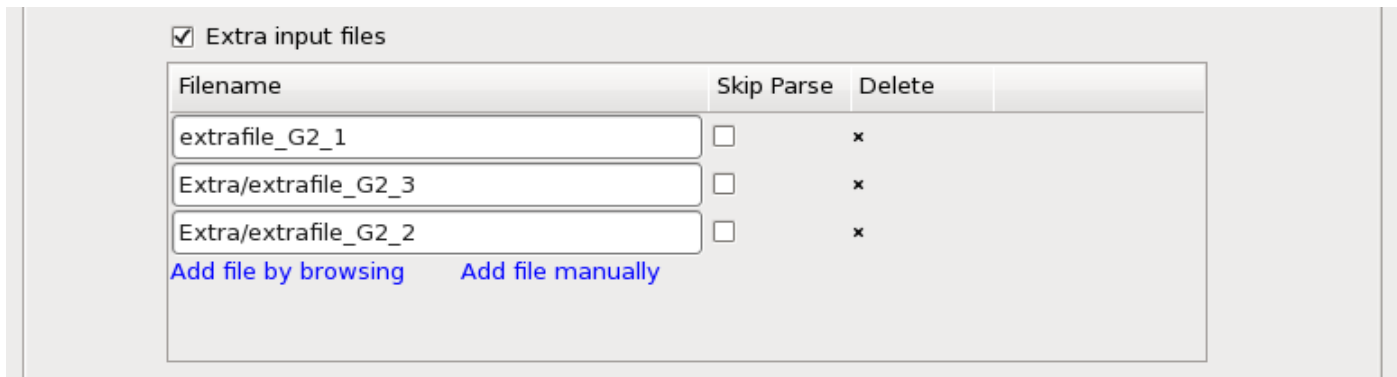
A record of the specified input files and parameters is displayed in the GUI but can also be checked in the `lsopt_input` file.

## 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 5.2.3. LS-OPT will automatically parse the files for variable names and list them on the **Parameters** page and in 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 switched off using the **Skip Parse** checkbox. This feature is typically used to move binary files to the run directories.



**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.

\*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.

### 5.2.3. Parameterization of Input Files

For all stage types, input files can be parameterized using the LS-OPT parameter format, Section 5.2.4. For the packages listed in Table 5-2, LS-OPT supports native parameters, see the respective package interface section for details.

LS-OPTui will automatically recognize the native and LS-OPT 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**

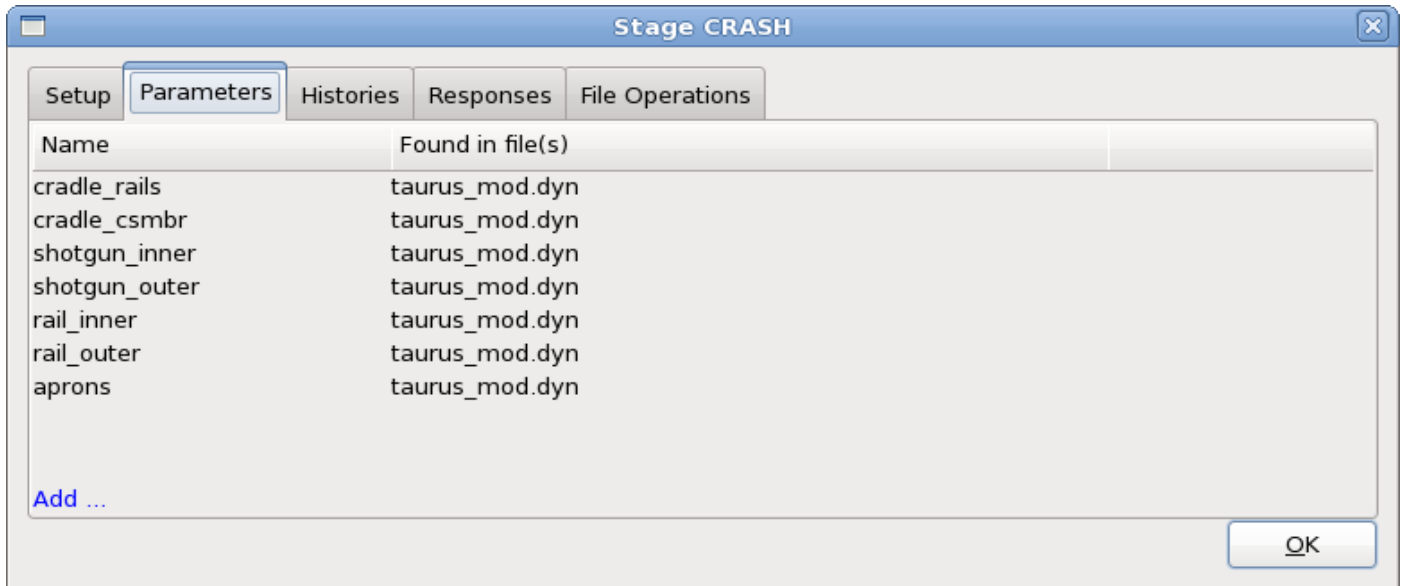
Package	Native parameters recognized in input file	LS-OPT Parameter Format recognized (see Section 5.2.3)	Include files recognized in input file	Reference
LS-DYNA®	Yes	Yes	Yes	Section 5.3.1
LS-PREPOST®	Yes	Yes	Yes	Section 5.3.3
MSC-NASTRAN <sup>1</sup>	Yes	Yes	No	Section 5.3.2
ANSA <sup>2</sup>	Yes	Yes	Yes	Section 5.3.6
HyperMorph <sup>3</sup>	Yes	Yes	No	Section 5.3.7
TrueGrid <sup>4</sup>	No	Yes	Yes	Section 5.3.5
LS-INGRID	No	Yes	Yes	Section 5.3.4
User-defined	N/A	Yes	No	Section 5.3.9

<sup>1</sup> Registered Trademark of MSC Software, Inc.

<sup>2</sup> BETA CAE Systems S.A.

<sup>3</sup> Registered Trademark of Altair Engineering, Inc.

<sup>4</sup> Registered Trademark of XYZ Scientific Applications, Inc.



**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 LS-OPT parameter format described next is recognized in all types of input files.

#### 5.2.4. The LS-OPT parameter format

LS-OPT provides a generic format that allows the user to substitute parameters in any type of 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 field width is 10 (commonly used in e.g. LS-DYNA input files). 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)>>`.

### 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.

## 5.2.5. 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 5.2.4 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 3.2.2.

## 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 LS-OPT 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**” and “**r**” formats for integers and real numbers respectively and will replace the number 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

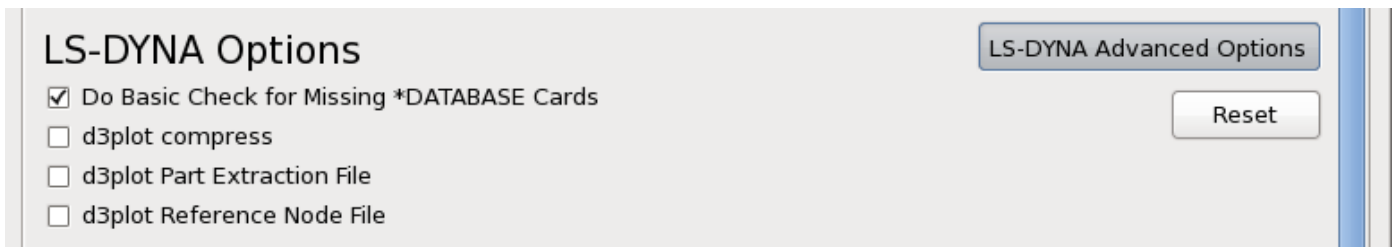
The LS-DYNA MPP (Message Passing Parallel) version can be run using the LS-DYNA option in the "Stage" dialog of LS-OPTui. The following run command is an example of how an MPP command can be specified:

```
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.



**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.
d3plot compress*	Compress the d3plot database. All results except displacements, velocities, and accelerations will be deleted.
d3plot Part Extraction File*	Write the results for a user selected set of parts. A file specifying the list of parts to be included/excluded is required. The file consists of multiple lines with a single entry per line. The syntax of the file is: <i>id</i> includes the part with <i>id</i> , <i>id1-id2</i> includes the parts from <i>id1</i> to <i>id2</i> , - <i>id</i> excludes the part with <i>id</i> . Only parts included with <i>id</i> or <i>id1-id2</i> can be excluded. For example:     5 7-20 -9.
d3plot Reference Node File*	Transforming the results to a local coordinate system specified by three nodes. The first node is the origin and the other two nodes are used to define the coordinate systems. The coordinate system moves with the nodes. A file specified the three nodes consisting of a single line is required. An example of the possible contents of the file: 1001 1002 1003.

**\* Remarks**

1. Altering the d3plot databases does not work with adaptivity.
2. The \*DATABASE\_EXTENT\_BINARY option in LS-DYNA also allows control over the size of the d3plot databases

**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:

1. 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.

- 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. MSC-NASTRAN<sup>®</sup> (SOL 103)

The user can interface with the NASTRAN implicit solver (sol 103) for modal analysis by selecting the MSC-NASTRAN option in the LS-OPT*ui*. The command can either execute a command, or a script. The substituted input file `NastranOpt.inp` will automatically be appended to the command or script. Variable substitution will be performed in the input file (which will be renamed `NastranOpt.inp`). The NASTRAN solver is required to generate a 'N o r m a l' termination command to standard output at the end of simulation. This can be done by executing NASTRAN using a script with its last statement being the command (see remark 2):

```
echo 'N o r m a l'.
```

#### Remarks:

- The NASTRAN solver must not be run in the batch mode. This can be done by specifying the 'batch=no' option with the NASTRAN command.
- A 'N o r m a l T e r m i n a t i o n' statement must be issued after finishing the NASTRAN job. This can be easily done by using the following script as the solver command:

```
#=====
/home/bin/nastran 'batch=no' $1
echo 'N o r m a l T e r m i n a t i o n'
#=====
```

- Design Parameters:** The design parameters can be specified using one of the following two options:
  - defrepsym:** The design variables can be specified using the

```
defrepsym varname default
```

statement. The design variable value is accessed using `%varname%`. The user must be careful to use the appropriate fieldwidth permitted by NASTRAN. This is the preferred option.
  - The LS-OPT parameter format discussed in Section 5.2.3.
- Creating the Database:** In order to facilitate the creation of appropriate LS-OPT readable database, the user must include the following *DMAP* code at the beginning of the input deck.

```
=====
$ open the binary file
```



```

ASSIGN OUTPUT4='nastEigout.op4' UNIT=39 UNFORMATTED DELETE $ binary
$
$ solver
SOL 103
DIAG 5, 6, 8, 56
$
$ Matrix manipulation
MALTER 'call modefsrs' $ after modes are calculated
LAMX,,LAMA/LMAT/-1/0 $ convert eigenvalue table to matrix
MPYAD, MAA, PHA,/MTP/1 $ matrix multiplication
OUTPUT4 PHA, LMAT, MTP,,//-1/39///16 $ output desired matrices
$
CEND

```

```

=====

```

The name of the output file (nastEigout.op4) and matrices (PHA, MAA, LMAT, MTP,...) must not be changed for successful reading of the binary file.

5. **Extracting data:** To extract NASTRAN modal analysis results, the users must use Nastran-Frequency type on the response panel instead of FREQUENCY type that is used for LS-DYNA.

### 5.3.3. LS-PREPOST

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

LS-PREPOST input file example with include:

*test01.cfile:*

```

$# LS-PrePost command file created by LS-PREPOST 3.0(Beta) - 31Mar2010(17:08)
$# Created on Apr-06-2010 (13:42:14)
cemptymodel
openc command "para01.cfile"
genselect target node
occfiler clear
genselect clear
genselect target node
occfiler 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

```

### 5.3.4. 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 LS-OPT parameter format is supported.

### 5.3.5. 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 LS-OPT parameter format is supported.

The TrueGrid input file requires the line:

```

write end
at the very end.

```

### 5.3.6. ANSA (BETA CAE Systems SA)

**General**

Package Name: ANSA

Command: ANSA [Browse]

Do not add input file argument

DV File: ansa\_variable\_def.txt [Browse]

copies ansa\_variable\_def.txt (0 includes) to Stage1/it.run/ ANSAOpt.inp and substitutes parameters

Extra input files

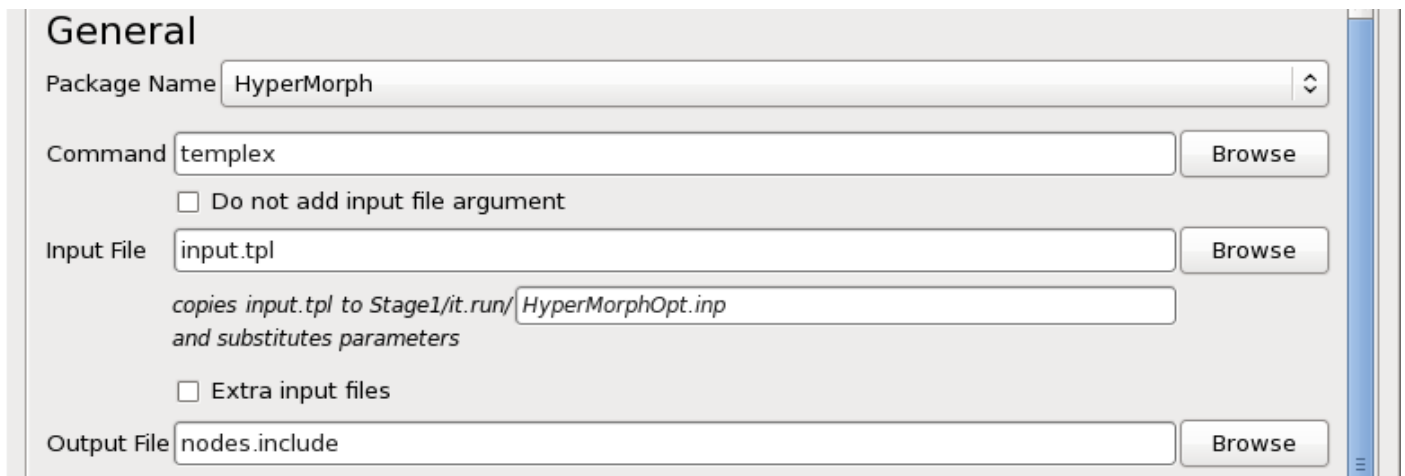
Model Database: data\_base\_name.ansa [Browse]

**Figure 5-5: Stage Setup for ANSA**

1. The ANSA preprocessor can be interfaced with LS-OPT allowing for shape changes to be specified. Several files must be specified:
2. Command: *ANSA executable*, typically named *ansa.sh*. Do not use an alias.
3. 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.
4. 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.

### 5.3.7. HyperMorph



**Figure 5-6: Stage Setup for HyperMorph**

1. To allow the specification of shape variables, the geometric preprocessor HyperMorph<sup>5</sup> has been interfaced with LS-OPT. Several files must be specified:
2. Command: `templex` command
3. Input file: At the top, the variables are defined as:
 

```
{parameter(DVAR1, "Radius_1", 1, 0.5, 3.0)}
```
4. 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:

<sup>5</sup> Registered Trademark of Altair Engineering, Inc.

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

**Remarks:**

1. LS-OPT uses the name of the variable on the DVAR $i$  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 $i$ .

### 5.3.8. $\mu$ ETA (BETA CAE Systems SA)

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

The screenshot shows a software interface titled "General". It contains five rows of input fields, each with a "Browse" button to its right:

- Package Name:** A dropdown menu with "METAPost" selected.
- Command:** A text box containing "METAPost".
- Session File:** A text box containing "sessionfile.txt".
- Output File:** A text box containing "METAPost\_results.txt".
- Database File:** A text box containing ".".

**Figure 5-7: MetaPost interface**

1. Several files must be specified:
2. Command: The  $\mu$ ETA executable
3. Session File: The session file containing information about which results to extract. This can be created interactively using  $\mu$ ETA.
4. 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.

5. Database File: This is the path for finding the solver database. The default `"/"` means that  $\mu$ ETA will look for the database locally. This specification has no effect during the optimization run as LS-OPT will always force  $\mu$ ETA 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  $\mu$ ETA 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  $\mu$ ETA settings in the LS-OPT GUI (see Figure 5-7). The user can browse for the  $\mu$ ETA 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. 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 `prepro own` preprocessor can be used in combination with the design point file: `XPoint` to read the design variables from the run directory.

If the `own` solver 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 `Normal`.
```

### 5.3.10. 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  $\mu$ ETA 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  $\mu$ ETA, 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  $\mu$ ETA as a user-defined post-processor. In this case, the command provided in “fullcommandscript” is:

```

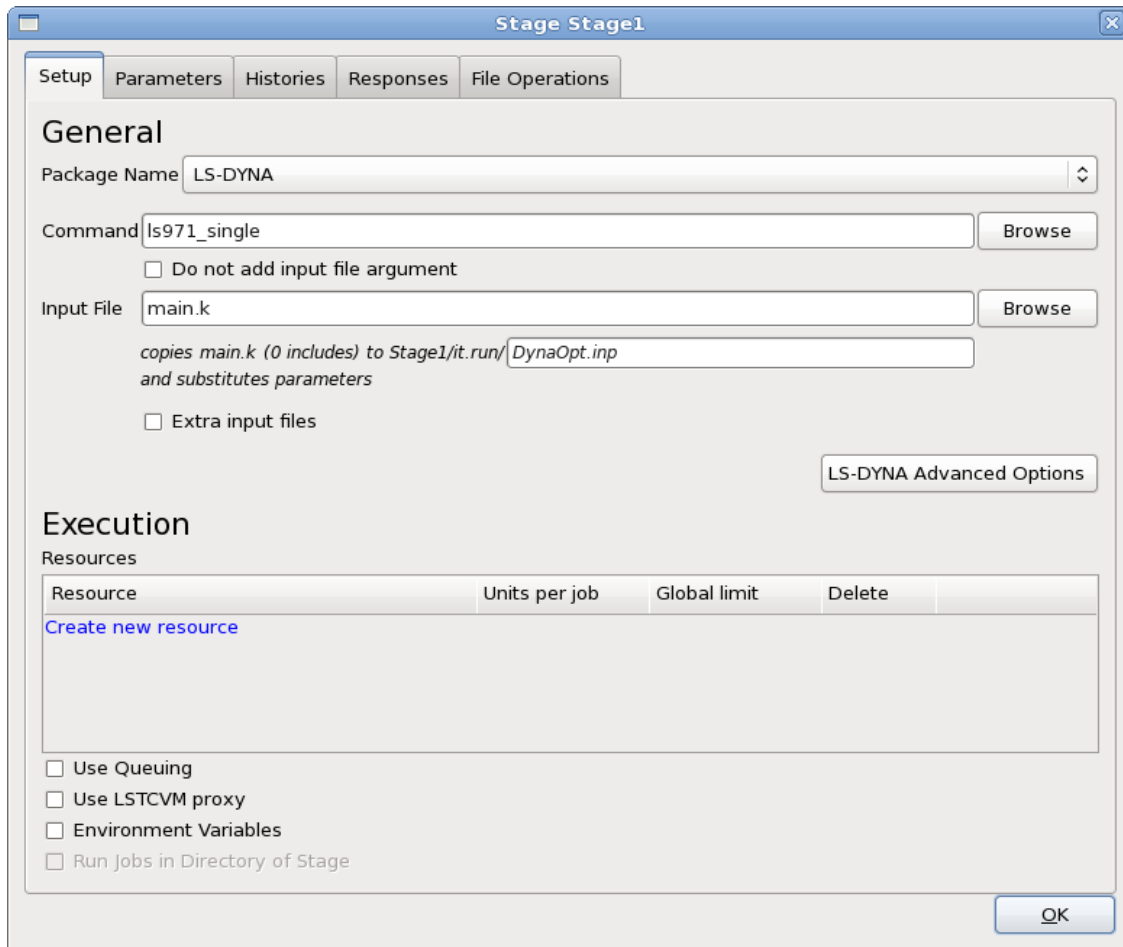
<metapost_executable> -b -s -foregr <path/sessionfile> "<database_path>"
"<path/result_file>"

```

Unlike in the case of  $\mu$ ETA, 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



*Figure 5-8: Stage dialog Setup panel*

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

Option	Description	Reference
Resources	Settings for concurrent processing	Section 5.4.1
Use Queuing	Interfacing with load sharing facilities to enable running simulation jobs across a network.	Section 5.4.2
Use LSTCVM proxy	Enabling LSTCVM, Secure Proxy Server, for distributing solver jobs across a computer cluster.	Section 5.4.3
Environment Variables	Environment variables that will be set before executing a solver command.	Section 5.4.4
Run jobs in Directory of Stage	If multiple stages are defined, the command can be executed in the directory of another stage.	-
Recover Files	List of files to be recovered from remote machine, only available if a queuing system interface is used	Section 5.4.5

### 5.4.1. Specifying Computing Resources for Concurrent Processing

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 5-9). 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 5-9. The job scheduler will launch jobs that will not exceed any of these two limits.

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

[Create new resource](#)

**Figure 5-9: 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 7.4 ). 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.

### **5.4.2. Interfaces to Queuing Systems**

The LS-OPT Queuing Interface interfaces with load sharing facilities (e.g. LSF<sup>6</sup> or LoadLeveler<sup>7</sup>) 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.

---

<sup>6</sup> Registered Trademark of Platform Computing Inc.

<sup>7</sup> Registered Trademark of International Business Machines Corporation

**Table 5-5: Queuing options**

Option	Description	Reference
LSF	LSF	
PBS	PBS <sup>8</sup>	
PBSPRO	PBS PRO	
SLURM	SLURM	
AQS	AQS	
LoadLeveler	LoadLeveler	
NQE	NQE <sup>9</sup>	
NQS	NQS <sup>10</sup>	
Black-Box	Black box	Appendix H.7
Honda	dedicated queuer	Appendix H.8
SGE	SGE	
User-Defined	User Defined	Appendix H.7

### 5.4.3. Using the LSTCVM secure proxy server

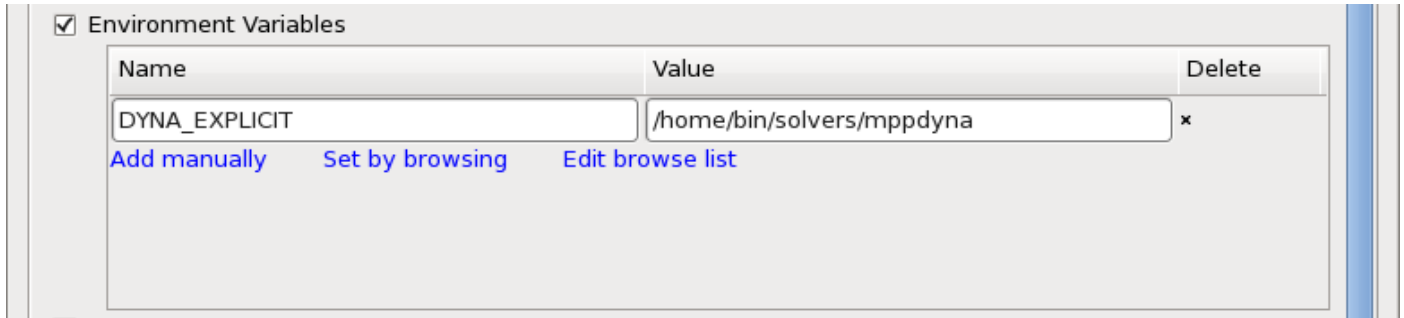
Selecting this option enables the interface to use LSTCVM. LSTCVM is a Secure Proxy Server for distributing solver jobs across a computer cluster, e.g. for running LS-OPT on a Windows machine controlling solver jobs on a Linux cluster. See Appendix H.10 for information on the installation of LSTCVM.

<sup>8</sup> Portable Batch System. Registered Trademark of Veridian Systems

<sup>9</sup> Network Queuing Environment. Registered Trademark of Cray Inc.

<sup>10</sup> Network Queuing System

### 5.4.4. Environment Variables



**Figure 5-10: 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 LSOPT

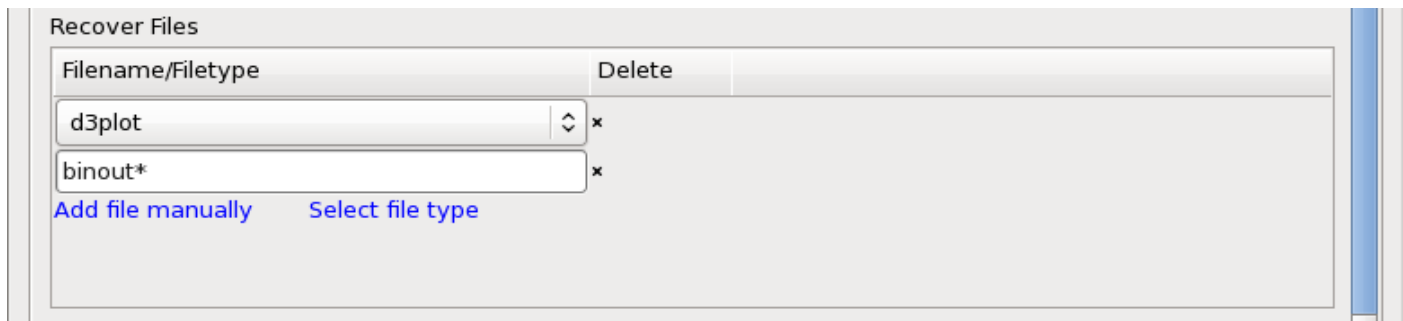
The **Browse List** (indeed, the complete **Environment Variables List**) is used to set environment variables before running the solver command specified by LS-OPT. However, if the first variable returned by the browse command is **exe**, then a pre-processing command is run before running the actual 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 **LSOPT Job Directory**. So, the **LSOPT Stage Directory** will be `".."` and the **LSOPT Project Directory** will be `"../.."`.

### 5.4.5. Recovering Output Files



**Figure 5-11: Database recovery options**

This option is only available if a queuing system interface is used, Section 5.4.2. 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`.

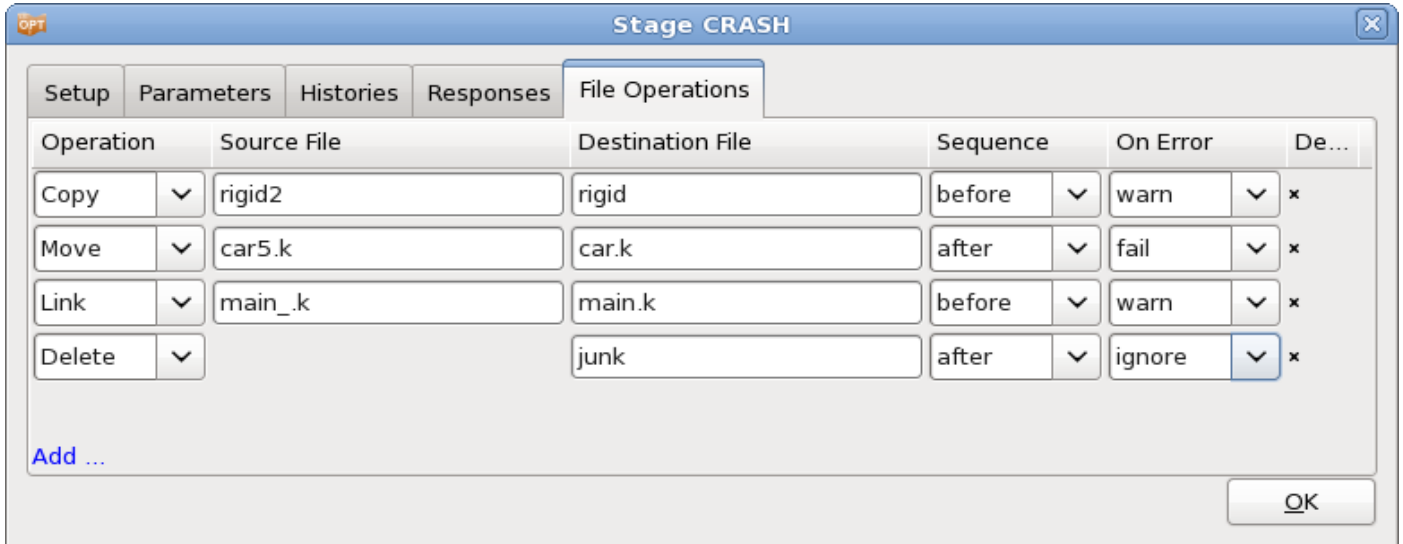
If the user wants to recover additional data to the local machine to do local post-processing (e.g. using LS-PREPOST), the *Recover Files* options can be used.

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 13.3.

## 5.5. File Operations



*Figure 5-12: 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 5-12 and Table 3-4.

File operations between stages are discussed in Section 3.2.2.

**Table 5-6: File Operations**

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

## 5.6. 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. 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 all other solvers, the user has the responsibility to write the status to standard output. This can be accomplished by inserting the solver command into a script in which the ‘N o r m a l’ string is written at the end using a print statement. See also Appendix H.9.1 .

## 5.7. 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.

- 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 5.7.1).
- 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.

### 5.7.1. 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 Section 13.6). For this purpose the user can provide a file named `clean` containing the required erase statements such as:

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

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.

## 5.8. 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 (see Points 1 to 3 below).
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.

1. 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.
2. 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).
3. 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 or response results to be extracted from the stage database. A history is a vector or curve data, whereas a response is a scalar value. Responses can be used to define objectives or constraints (Chapter 10), histories are intermediate entities that can be used to calculate responses or composites (Chapter 9). Interfaces for result extraction from LS-DYNA and MSC-NASTRAN output files are available, as well as mathematical expressions, file import, an interface for extraction of values from ASCII database and a user-defined interface where any program may be used for result extraction. The dialogs are accessible from the **Stage** dialog **Histories** and **Responses** tab, respectively.

## 6.1. Defining histories and responses

A history or a response can be defined by using the interfaces in the **Histories** and **Responses** 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 history or 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 four types of interfaces:

- Standard LS-DYNA or MSC-Nastran result interfaces. This interface provides access to the LS-DYNA binary databases (d3plot or binout, d3hsp or d3eigv). The interface is 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.

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 response is identified by a name, Table 6-2, and the settings to be specified using the respective interface.

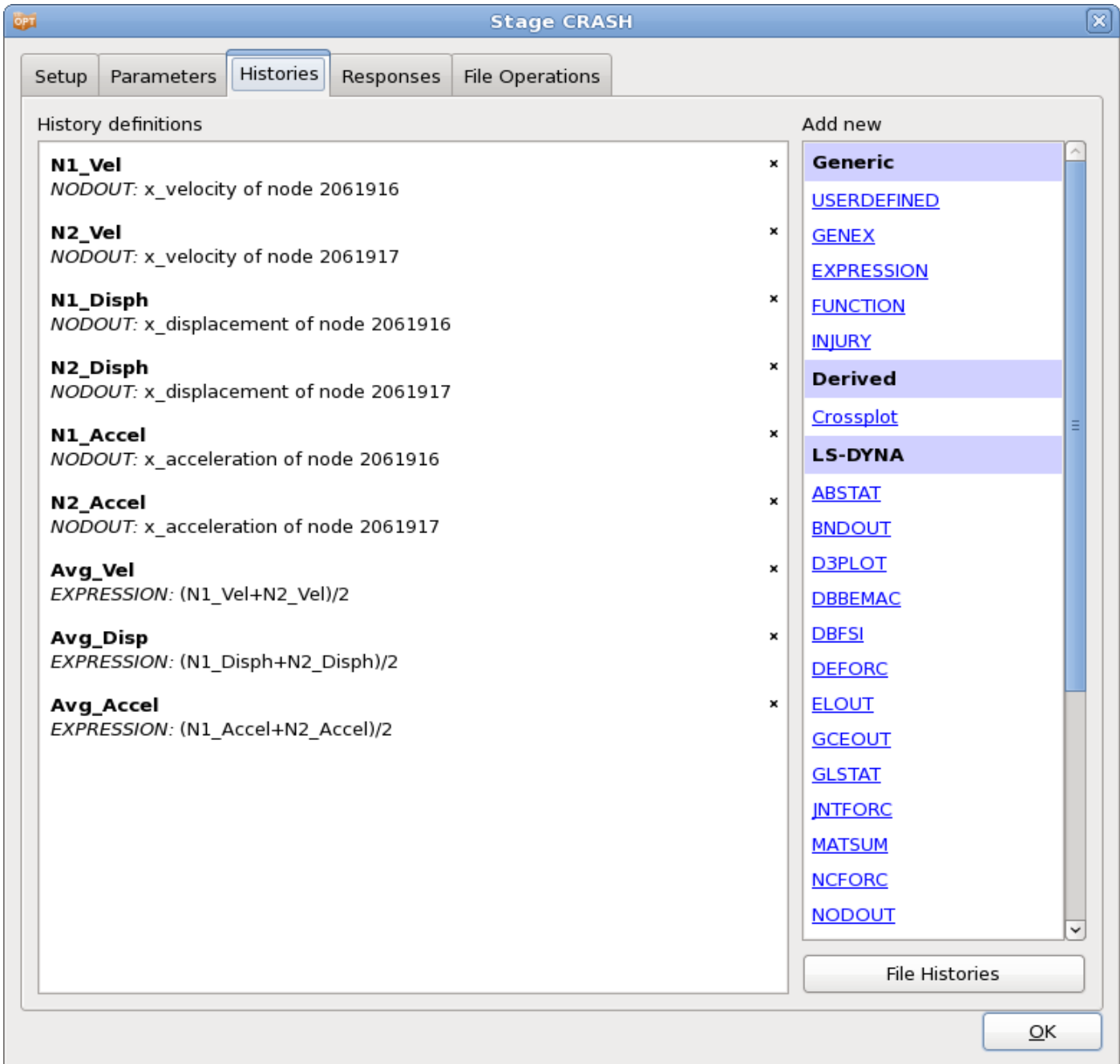


Figure 6-1: Histories definition in the GUI

**Table 6-1: Interfaces for Response and History extraction**

Option		Description	Reference
Generic	USERDEFINED	Result extraction using any script or program	Section 6.14
	GENEX	Tool for extracting results from text files	Section 6.13
	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
	MATRIX_EXPRESSION	(Response only)	Section 6.4.4
Derived	Crossplot	Crossplot (History only)	Section 6.4.1
LS-DYNA	ABSTAT	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 (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
MSC- NASTRAN	NAST_FREQUENCY	(Response only)	Section 6.15
File Histories		Global file histories (History only)	Section 6.16

**Table 6-2: General 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 $\times$ Multiplier ) + Offset.

Not metamodel linked (Response only) In some cases it may be beneficial to create intermediate responses without associated metamodels, but still part of a metamodel-based analysis. For example omitting intermediate neural networks will improve efficiency.

Responses that are not metamodel linked cannot be included directly in composites, since a composite relies on interpolation from a metamodel.○

### 6.1.1. Result extraction

Each simulation run is immediately followed by a result extraction to create the `history.n` and `response.n` files for that particular design point. For distributed simulation runs, this extraction process is executed on the remote machine. The `history.n` and `response.n` files are subsequently transferred to the local run directory. If the extraction on the remote machine is not successful, it is done again 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.lsox` database.

## 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`. The user must ensure that the LS-DYNA program will provide the output files required by LS-OPT.

The options for the extraction of LS-DYNA responses and histories are identical, except for the selection attribute.

Aside of the standard interfaces that are used to extract any particular data item from the database, specialized responses for metal-forming are also available. The computation and extraction of these secondary 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.

The response options are an extension of the history options – a history will be extracted as part of the response extraction.

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

Filtering and averaging options are available for histories and responses.

For responses, the *Select* attribute 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 0 and the end value of the history.

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.

Figure 6-2: Response extraction: LS-DYNA NODOUT interface

## 6.2.2. Kinematics

Additional kinematics such as *distances* 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  $\mathbf{q}$  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 0 to define the rotation matrix  $\mathbf{A}$  where  $\mathbf{A}$  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	$\mathbf{d} = \mathbf{q}$	$\mathbf{u} = \mathbf{q} - \mathbf{q}(0)$
Local	$\mathbf{d}' = \mathbf{A}(t)\mathbf{q}(t)$	$\mathbf{u}' = \mathbf{A}(t)\mathbf{q}(t) - \mathbf{A}(0)\mathbf{q}(0)$
Local in reference	$\mathbf{d}'' = \mathbf{A}^T(0)\mathbf{A}(t)\mathbf{q}(t)$	$\mathbf{u}'' = \mathbf{A}^T(0)\mathbf{A}(t)\mathbf{q}(t) - \mathbf{q}(0)$

The orthogonal matrix  $\mathbf{A}(t)$  is defined by a local coordinate system ( $x'y'z'$  in Figure 6-3) 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 (see Figure 6-3) while Node **1** represents the third node. This is the same convention as defined in Section 0.

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 Isopt\_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.

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

Kinematic quantities are available as both histories and responses.

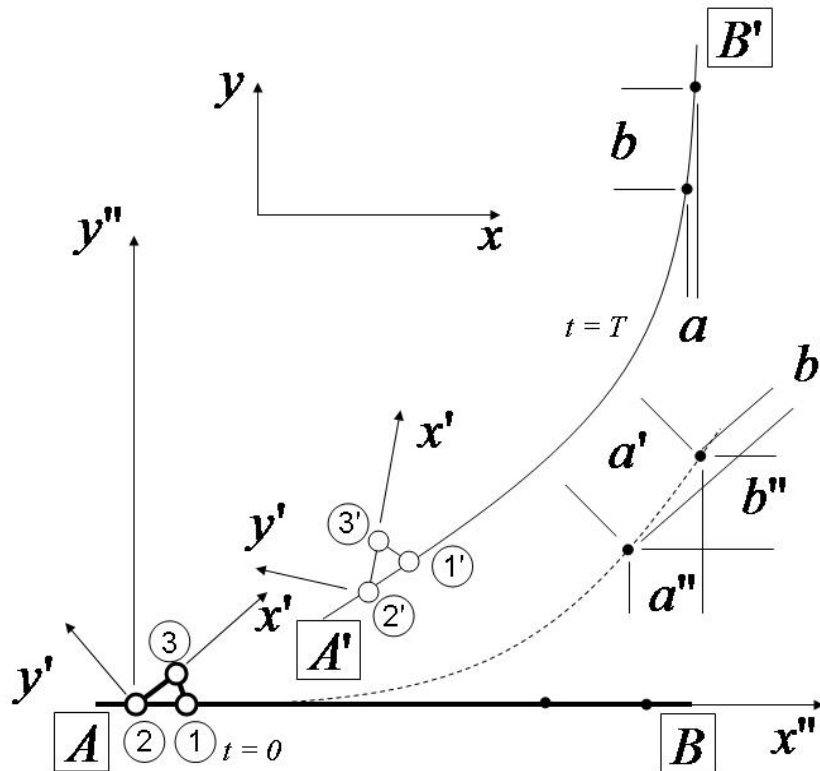


Figure 6-3: Local and global coordinate systems

### 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 as well. Element results such as stresses will be averaged in order to create the NODE 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* attribute 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 0 and the end value of the history. If the selection must be done over parts as well, firstly the maximum value will be selected for the part, 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

Location

Part       ID

Coordinate

Parts to be included

All Parts

List of parts:

Results Type	Component
<input type="radio"/> Ndv	<input checked="" type="radio"/> xx_stress <input type="radio"/> von_mises
<input checked="" type="radio"/> Stress	<input type="radio"/> yy_stress <input type="radio"/> 1st_prin_dev_stress
<input type="radio"/> Result	<input type="radio"/> zz_stress <input type="radio"/> 2nd_prin_dev_stress
<input type="radio"/> Strain	<input type="radio"/> xy_stress <input type="radio"/> 3rd_prin_dev_stress
<input type="radio"/> Misc	<input type="radio"/> yz_stress <input type="radio"/> max_shear_stress
<input type="radio"/> FLD	<input type="radio"/> zx_stress <input type="radio"/> 1st_principal_stress
<input type="radio"/> Beam	<input type="radio"/> plastic_strain <input type="radio"/> 2nd_principal_stress
	<input type="radio"/> pressure <input type="radio"/> 3rd_principal_stress

Select: Maximum Value      From time:      To time:     

Cancel      OK

*Figure 6-4: Response extraction from d3plot*

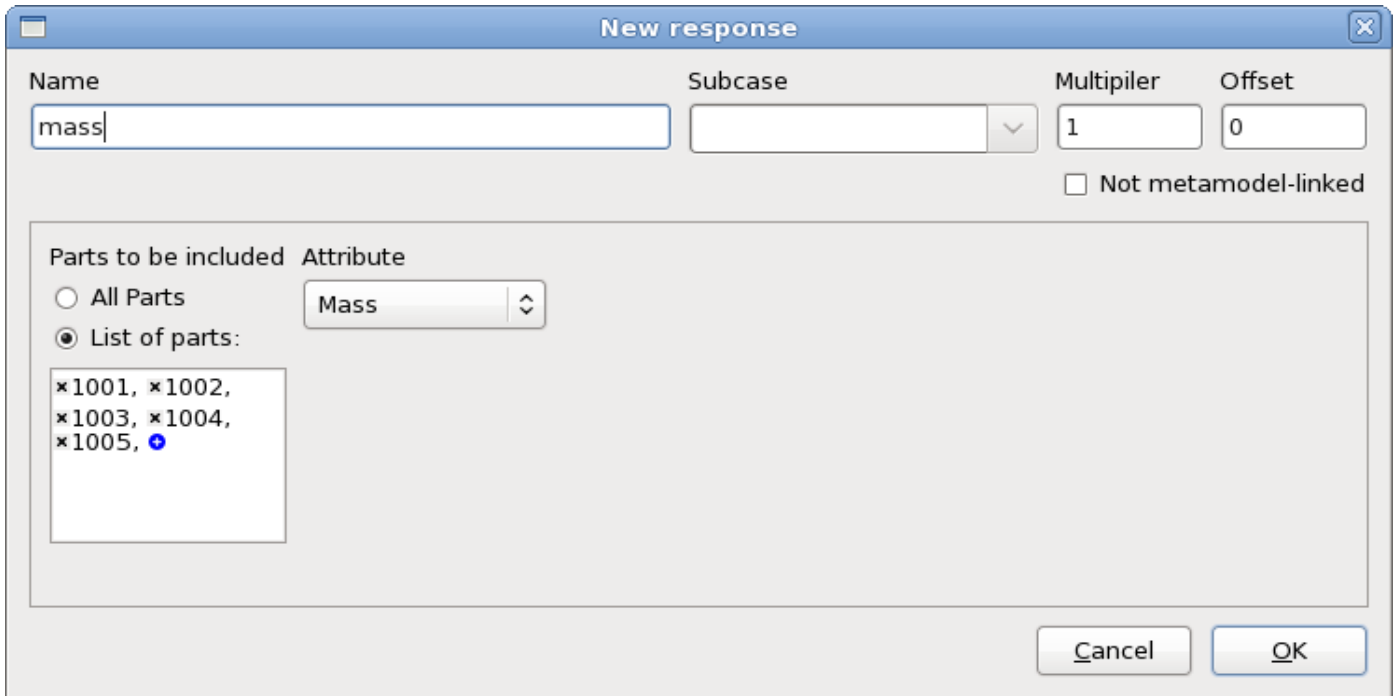
## D3Plot FLD results

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.

### 6.2.4. Mass – Interfacing with d3hsp

The MASS response interfaces with the LS-DYNA output file d3hsp. The Mass and related entities, Figure 6-5 and Table 6-4, can be extracted for the whole model or a list of parts.

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.



**Figure 6-5: Interface for extraction of Mass and related entities from LS-DYNA output d3hsp**

**Table 6-4: 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 6-6. See Table 6-5 for a description of the available extraction options.

**New response**

Name: frequency      Subcase:      Multiplier: 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

Cancel      OK

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

**Table 6-5: 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 <code>d3eigv</code> 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  $\varphi_0$  and the  $j^{\text{th}}$  modal vector of the current design  $\varphi_j$  is calculated as:

$$MAC_j = \frac{\{\varphi_0\}^H \{\varphi_j\} \{\varphi_j\}^H \{\varphi_0\}}{\{\varphi_0\}^H \{\varphi_0\} \{\varphi_j\}^H \{\varphi_j\}} \quad (6-1)$$

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 6-5).

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 6-7.

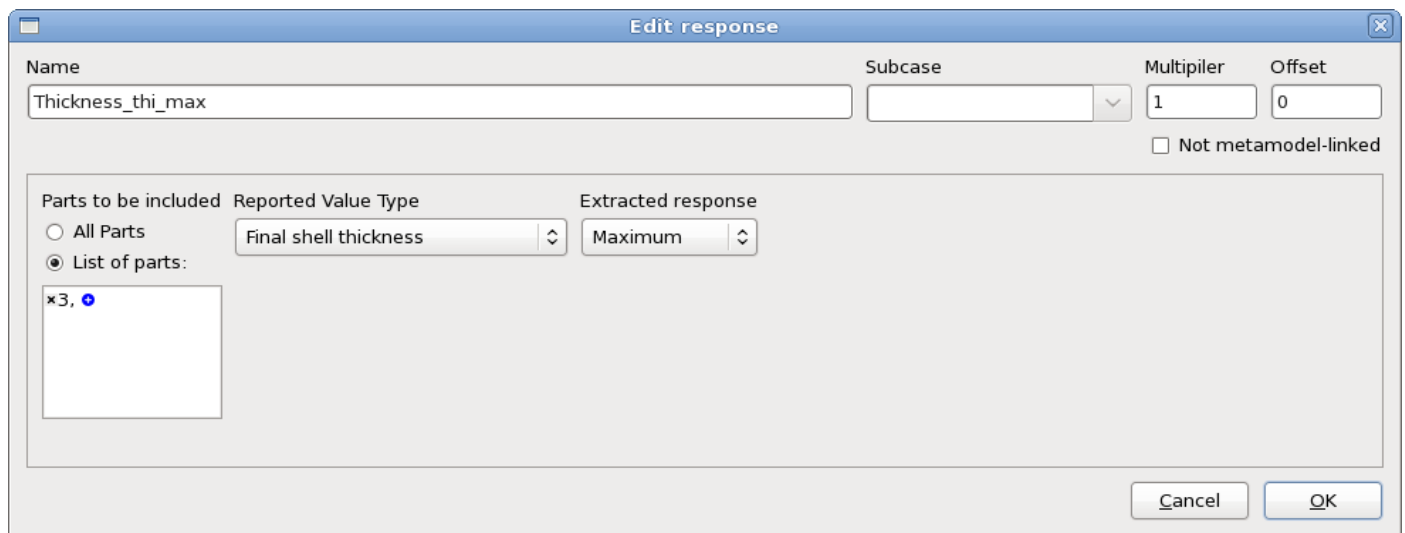


Figure 6-7: Thickness or Thickness reduction interface

Table 6-6: `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 6-8. 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:

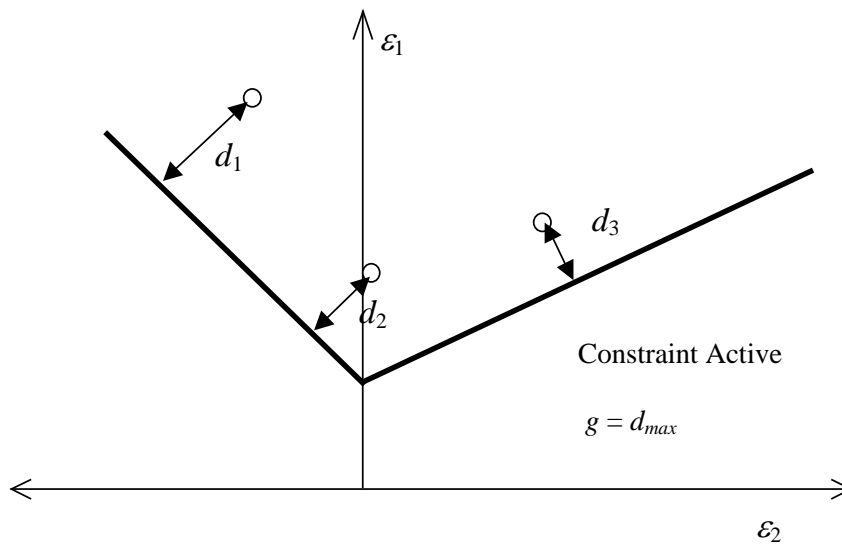
$$g = d_{\max}$$

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

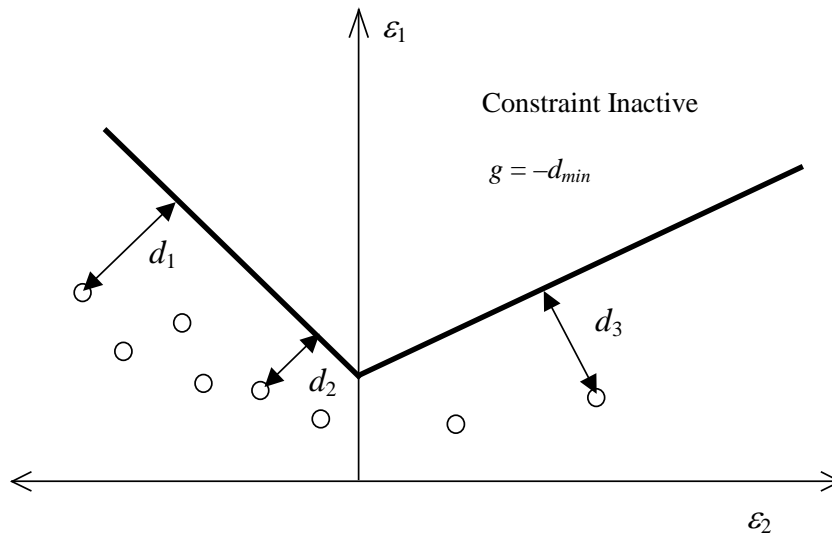
- All the values of the strain points are located below the FLD curve. In this case the constraint is computed as:

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

with  $d_{\min}$  the minimum smallest distance of any strain value to the FLD curve (Figure 6-8).



a) FLD Constraint active



b) FLD Constraint inactive

**Figure 6-8: FLD curve – constraint definition**

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

### 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.

*Remarks:*

- A piece-wise linear curve is defined by specifying a list of interconnected points. The abscissae ( $\varepsilon_2$ ) of consecutive points must increase (or an error termination will occur). Duplicated points are therefore not allowed.
- The curve is extrapolated infinitely in both the negative and positive directions of ( $\varepsilon_2$ ). The first and last segments are used for this purpose.
- The computation of the constraint value is the same as shown in Figure 6-8.

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

**Figure 6-9: Definition of General FLD constraint**

**Table 6-7: LS-DYNA General FLD constraint options description**

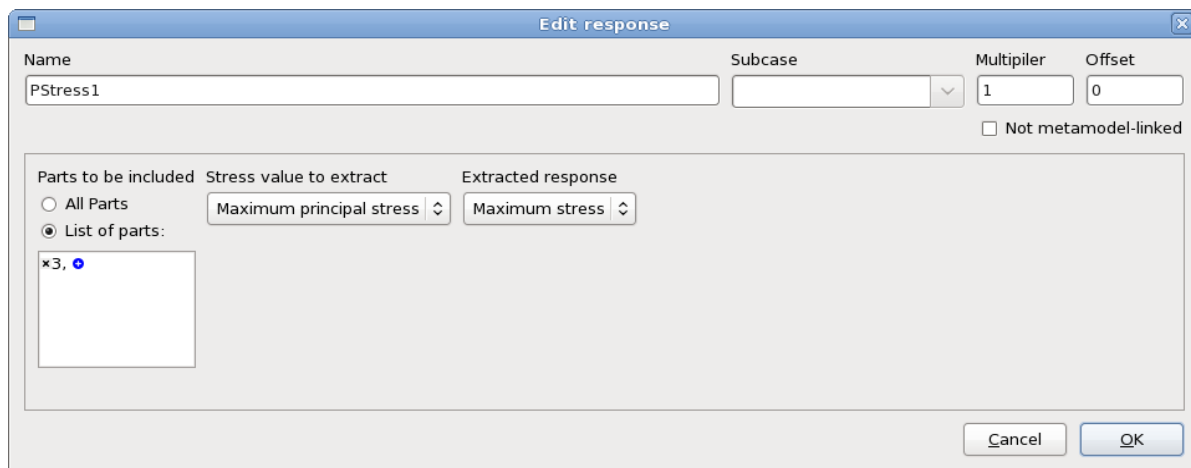
Option	Description
Parts to be included	Entity is extracted for the entire model or for the parts IDs specified in the list.
Sampling location	Lower, middle or upper surface of the sheet
Load curve ID	Identification number of a load curve in the LS-DYNA input file. The <code>*DEFINE_CURVE</code> keyword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

**Remarks:**

- The interface program produces an output file `FLD_curve` which contains the  $\varepsilon_1$  and  $\varepsilon_2$  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.
- 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.

**Figure 6-10: Principal Stress Interface**



**Table 6-8: 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 $\sigma_1$
	Second principal stress $\sigma_2$
	Minimum principal stress $\sigma_3$
	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

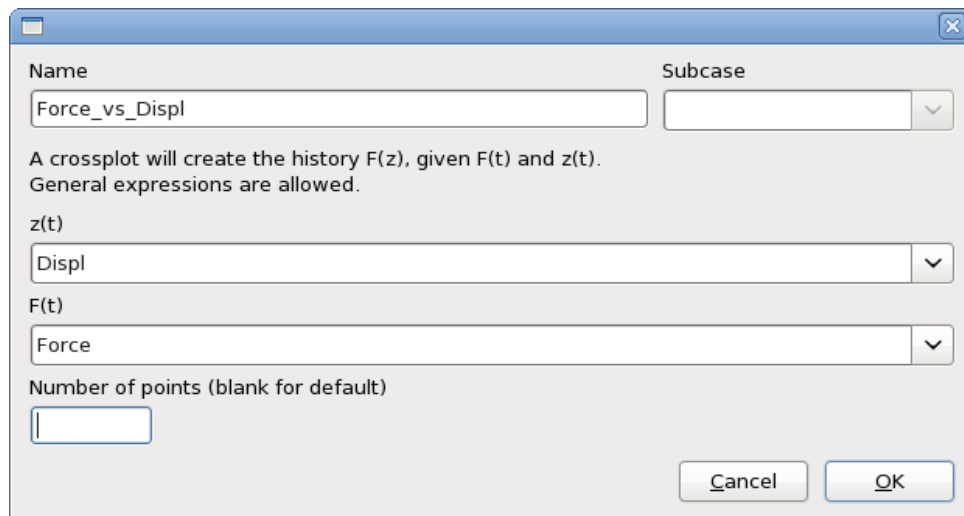
## 6.4. Generic Interfaces for History and Response extraction

### 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 .

### 6.4.2. Crossplot history

A special history function `Crossplot` is provided to construct a curve  $g(f)$  given  $f(t)$  and  $g(t)$ .

**Figure 6-11: Interface to define a crossplot history**

Additional options are available if the Crossplot is defined using a history Expression, see Appendix F.3 for details.

The options are explained in Table 6-9.

**Table 6-9: Description of Crossplot arguments**

Option	Description	Default
z(t)	History of abscissa	-
F(t)	History of ordinate	-
Number of points	Number of points created in crossplot	Smallest of the numbers of points defining $f$ and $g$

### 6.4.3. Function Interface

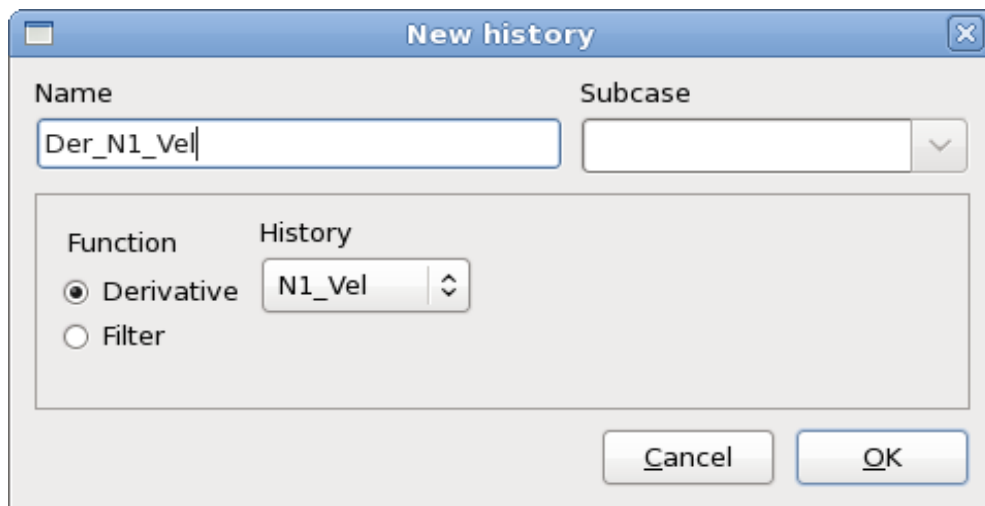
The functions available for the extraction of response values from previously defined histories are explained in Appendix F.3 .

The History functions are described below.

#### Derivative history

A special history function `Derivative` is provided to construct a curve  $\frac{df(t)}{dt}$  given  $f(t)$ ,

$$\frac{df(t)}{dt} \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$



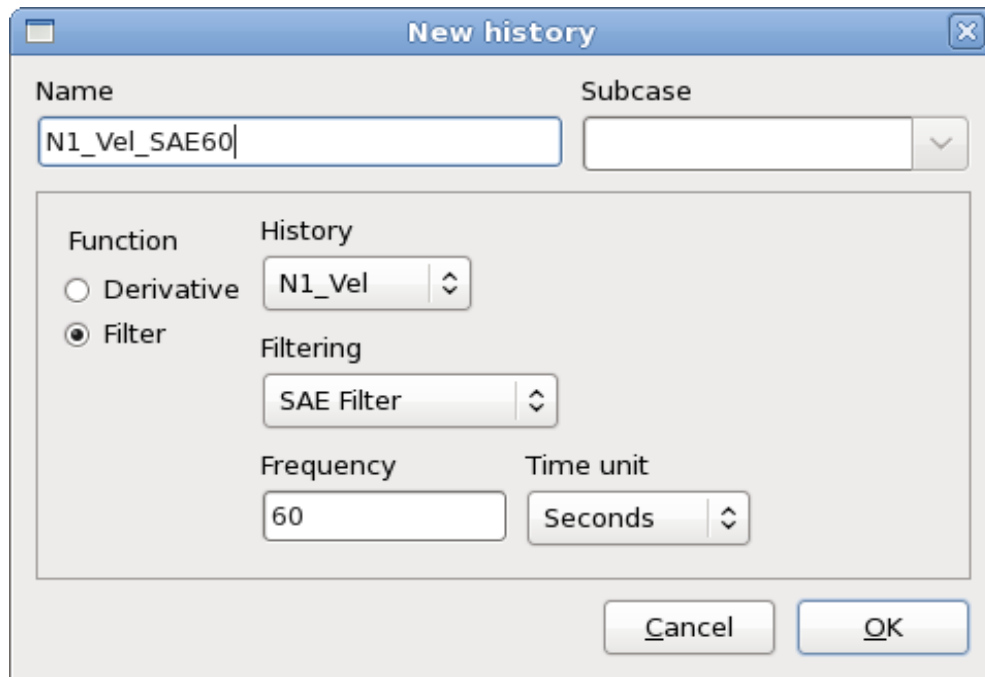
**Figure 6-12: Interface to define derivative history**

*Remarks:*

- The derivatives assume a linear, positive abscissa with equal intervals (typically time history).
- Since the derivative approximation is based on a multipoint scheme, it is recommended to avoid having too few points.
- The derivatives of the first three and last three points are the same as the third and third last points, respectively.

**Filtered history**

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



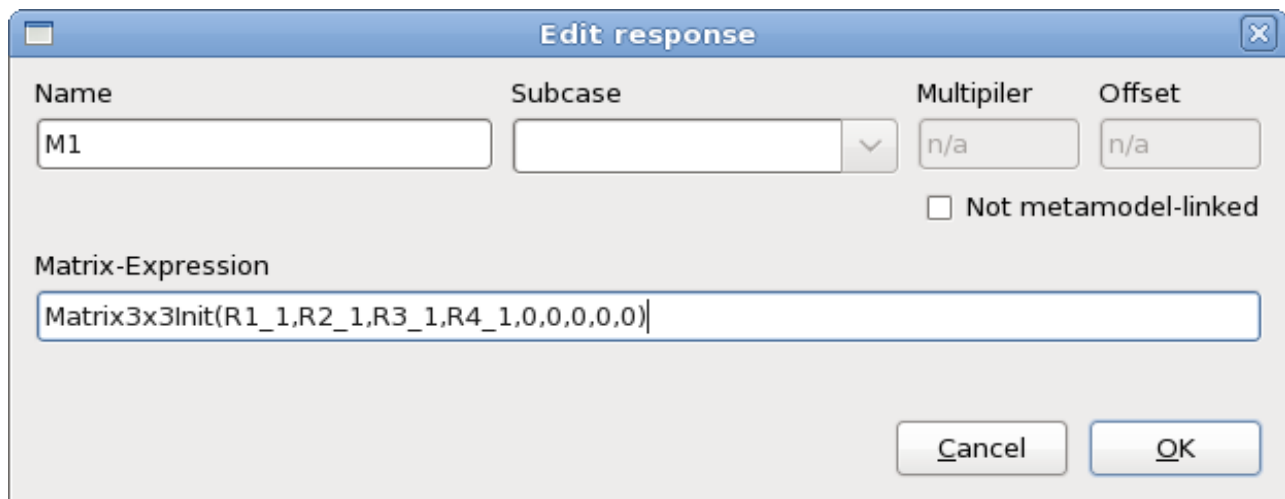
*Figure 6-13: Interface to define a filtered history*

**Table 6-10: Description of *FilterHistory* 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

#### 6.4.4. 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 6-14.

**Figure 6-14: Matrix Expression: Initialization of a matrix**

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

The component of a matrix is extracted using the format `A.ajj` (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`.

#### 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.
- $Z = v_{23} \times v_{21}$
- $Y = Z \times X$

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

$$T = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix}$$

where  $T^T T = I$ .

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

depends on the dummy type, Table 6-12.

**Table 6-11: 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 6-12: 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
WORLD SID	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 6-13: 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.

The Nij value 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 6-14: Options for  $N_{ij}$  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 6-15: 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]:

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

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 6-16: 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 6-17: 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 6-18: Options for LNL arguments**

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

**Table 6-19: Input constants**

Force/Moment	Description	Value
$C_{\text{moment}}$	Critical moment	15 [Nm]
$C_{\text{shear}}$	Critical force	250 [N]
$C_{\text{tension}}$	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 6-20: Options for Chest Compression arguments**

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

**Table 6-21: 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 6-22: 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 6-23: 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:**

- The derivative is computed using the 4<sup>th</sup> order (template size = 5) finite difference approximation:

$$\frac{df}{dt} = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h} + O(h^4)$$

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

- 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:

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

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

- with  $A(\text{upr.rib})$  Maximum y-acceleration of the upper rib  
 $A(\text{lwr.rib})$  Maximum y-acceleration of the lower rib  
 $A(\text{lwr.spine})$  Maximum y-acceleration of the lower spine

The result is divided by the gravitational acceleration g (9810mm/s<sup>2</sup>).

**Table 6-24: Options for TTI arguments**

Option	Description	Symbol
Acceleration upper rib	y-acceleration of the upper rib	$A(\text{upr.rib})$
Acceleration lower rib	y-acceleration of the lower rib	$A(\text{lwr.rib})$
Acceleration lower spine	y-acceleration of the lower spine	$A(\text{lwr.spine})$
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 is 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 6-25: 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 6-26: 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  $\Delta t$  (3ms). The resulting acceleration level is divided by the gravitational acceleration,  $g = 9810mm/s^2$ .



**Table 6-27: 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:

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

## 6.11. LS-DYNA Binout injury criteria

The injury criteria such 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  $9.81 \text{ m/s}^2$

## 6.12. REFERENCES

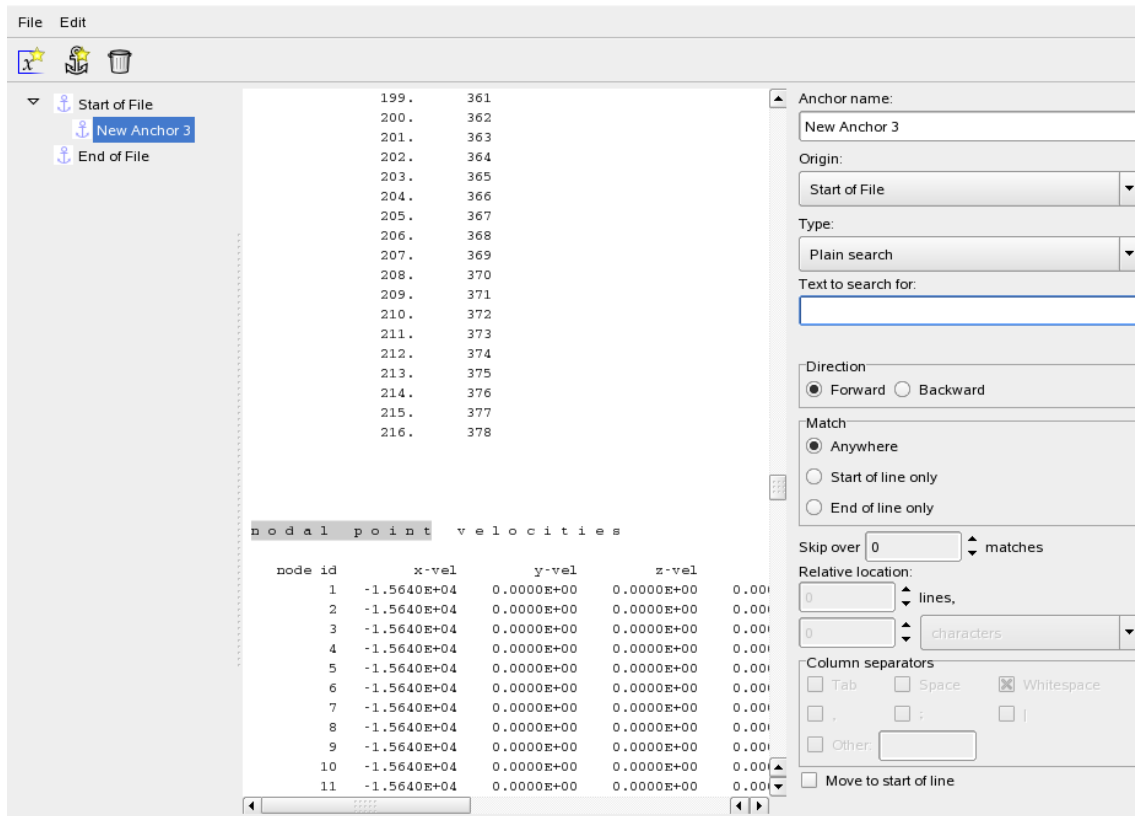
- [1] Data Processing Vehicle Safety Work Group - Crash Analysis Criteria Description. Version 2.1.1 Arbeitskreis Messdatenverarbeitung Fahrzeugsicherheit, May 2008.
- [2] K.-U. Schmitt, M. Muser, How to calculate the  $N_{km}$ , Working Group on Accident Mechanics, Zürich, 2003

## 6.13. The GenEx application for extracting entities from a text file

**GenEx** (Generic Extractor) is a tool to create the .g6 file used by LS-OPT to extract responses and histories. It is included in the LS-OPT distribution as the executable file genex.

### 6.13.1. 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.



**Figure 6-15: GenEx dialog.**

When first starting GenEx, there will be two predefined anchors in the tree on the left, **Start of File** and **End of File**. 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/selecting 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 “Options specific for entities” for the difference between them).

## 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.

### 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

### Type

There are four types of searches. Three of them are keyword-based (search-phrase based).

- **Plain text:** This is the most basic search. It looks for the given text in the file and positions the anchor in front of the match.
- **Glob search:** The glob search main goal is to be able to match 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.

Example:

\*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:** 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.

Examples:

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).

- **Absolute search:** In this search the user positions the anchor simply by telling on which row and in which column the anchor is located in the file.

*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.

### ***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.

### ***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.

### ***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.

### ***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.

## **Options specific for entities**

### ***Relative Location***

This is the position of the entity, relative to the parent anchor.

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.

### *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.

### Maximum number of components

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.

### 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 can’t be repeated.

## **6.13.2. 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’s possible to select a column of numbers to create a column vector.

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.

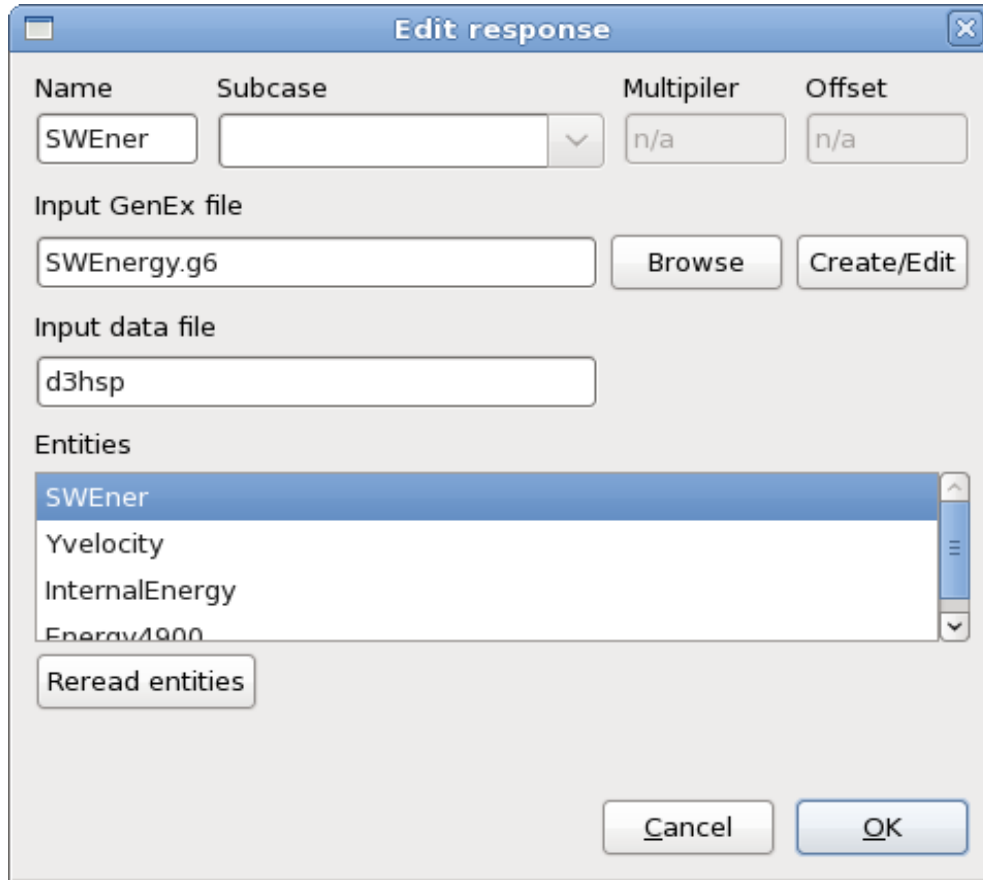
### **Creating an .g6 file without an input file**

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”.

### 6.13.3. How to use GenEx from LS-OPT for extracting responses



*Figure 6-16: 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. LS-OPT looks for the file in the run directory.

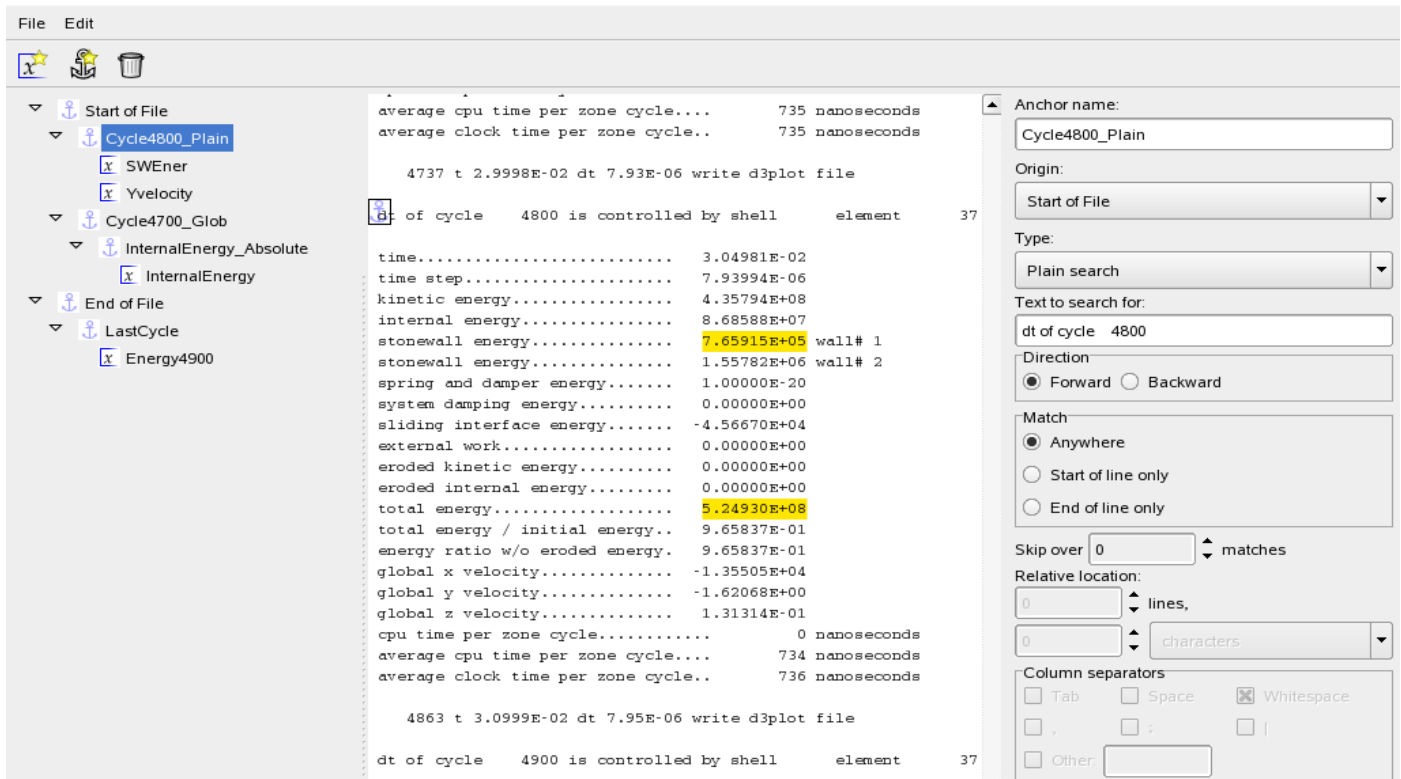
### 6.13.4. An example using GenEx to extract responses

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.



**Figure 6-17: 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.

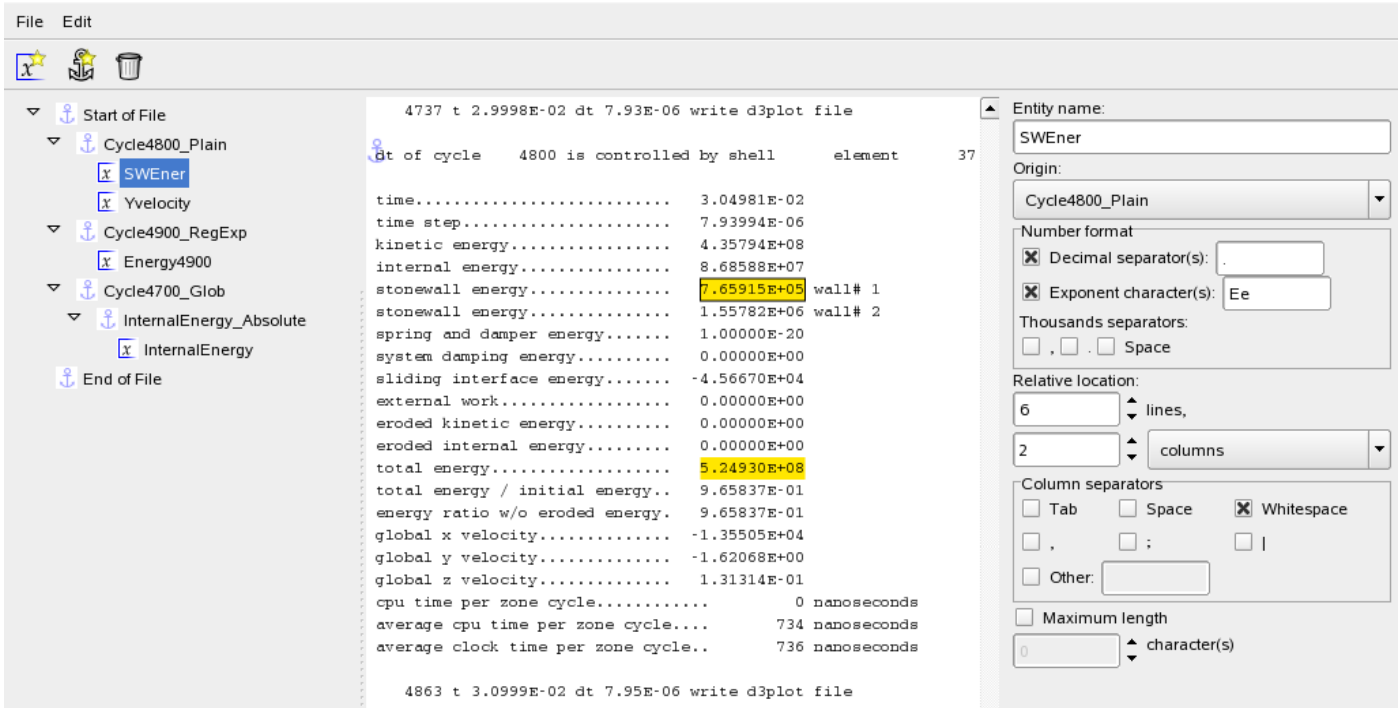


Figure 6-18: GenEx dialog; definition of an entity

- o 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:

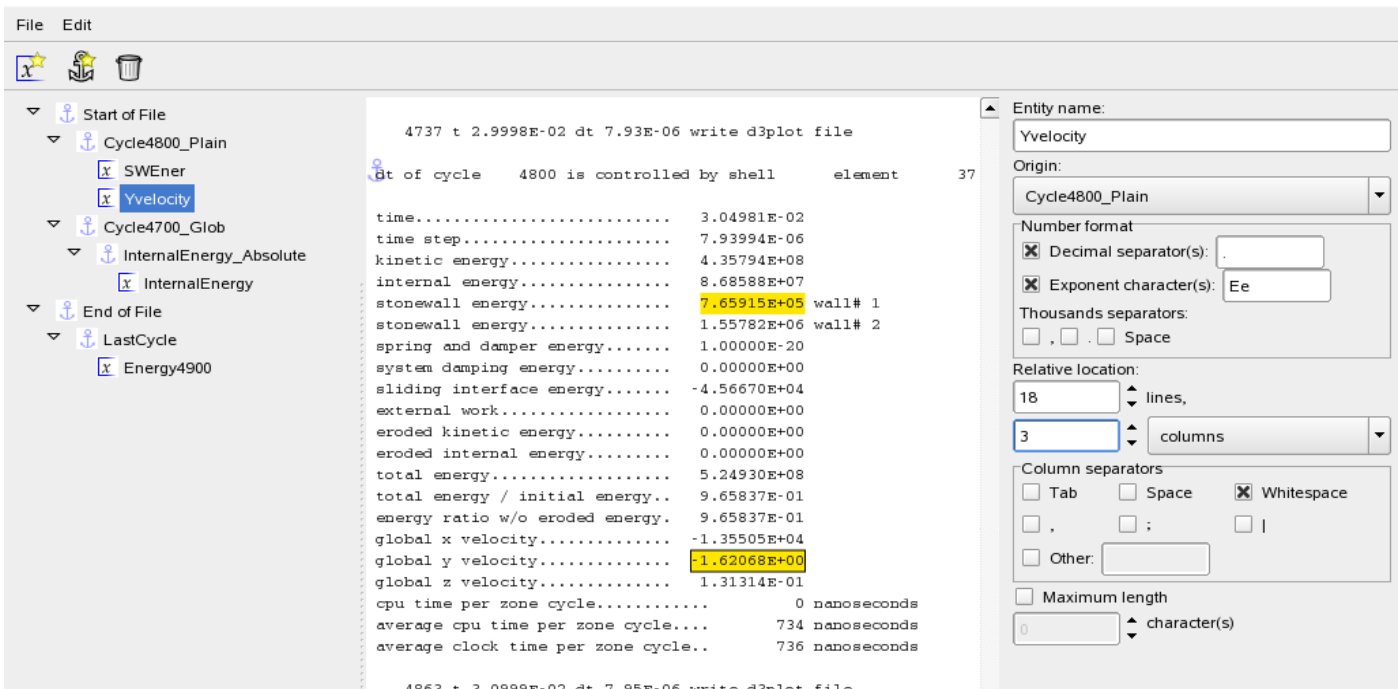
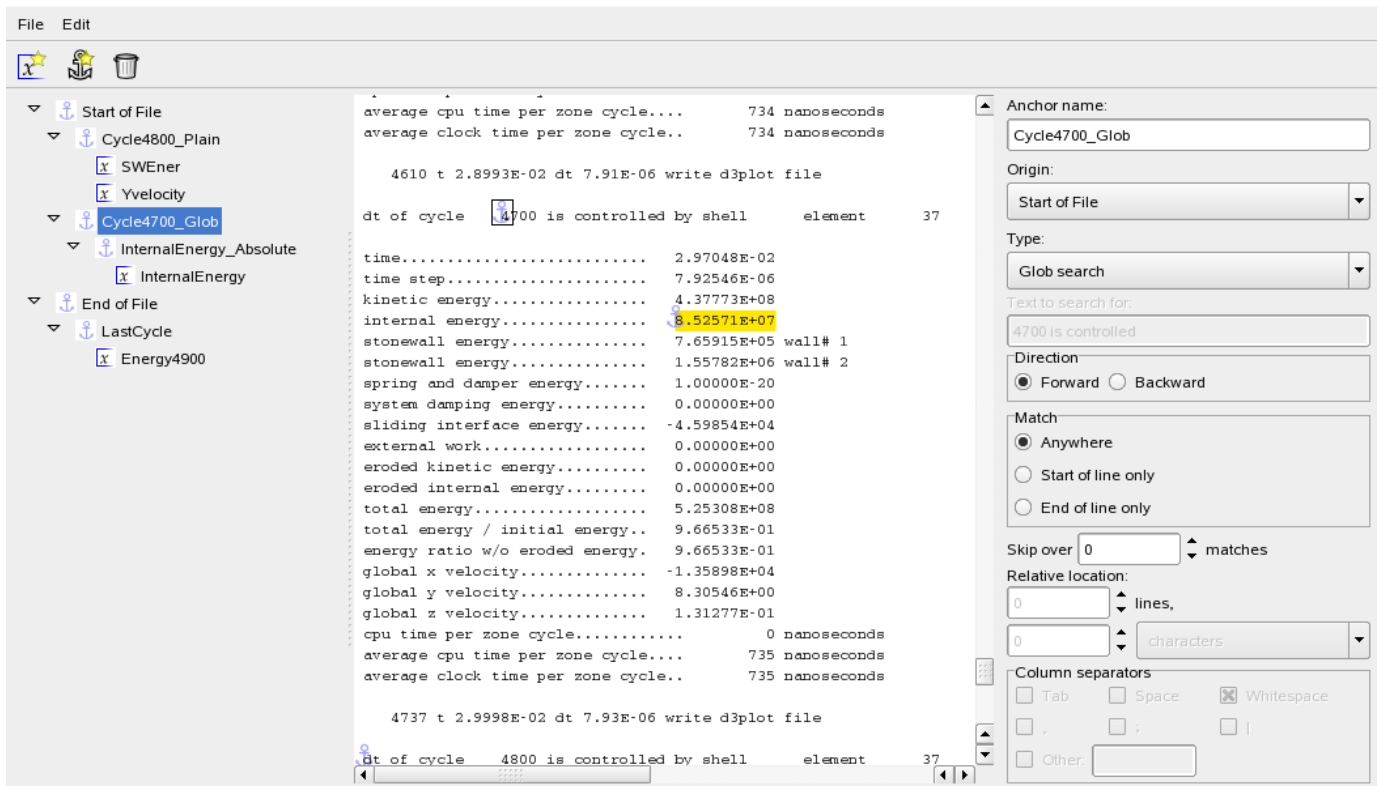


Figure 6-19: GenEx dialog; definition of an entity

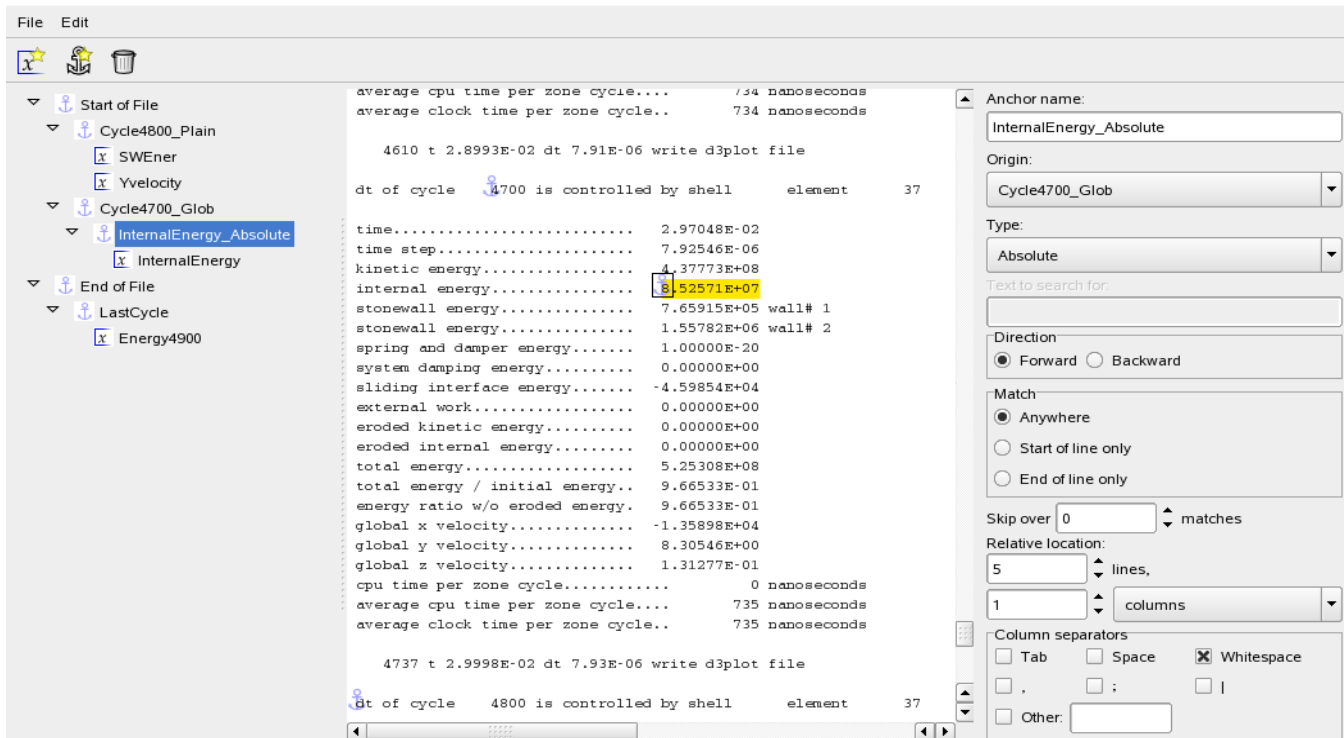


- Define a second anchor using a global search for the string "4700 is controlled". The origin of this anchor is also the start of the file and the search is forward from that point. Note the anchor placement on the figure below just before the string "4700 is controlled".



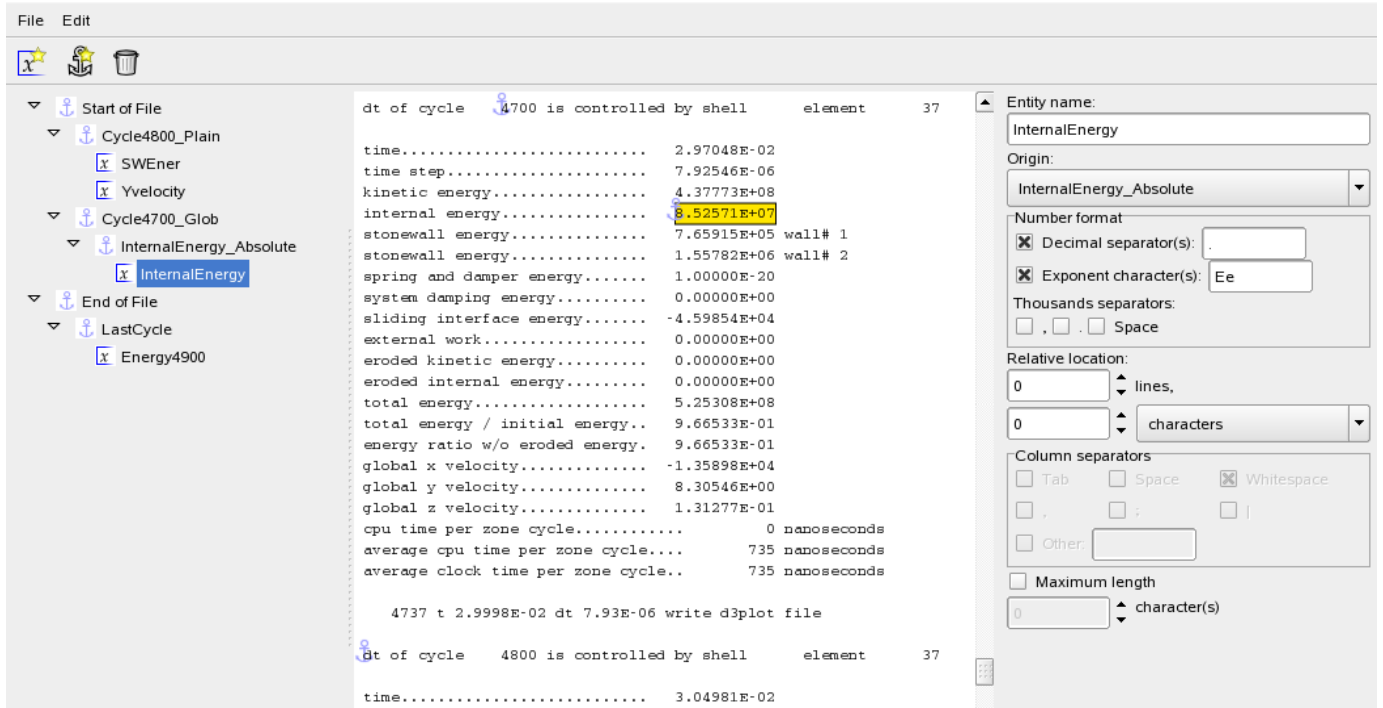
**Figure 6-20: GenEx dialog; definition of an anchor**

- 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.



**Figure 6-21: 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.



**Figure 6-22: 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.

The screenshot shows the GenEx dialog box with the following settings:

- 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
- Column separators:**  Tab  Space  Whitespace
- Move to start of line

The log file content is as follows:

```

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

storage allocation

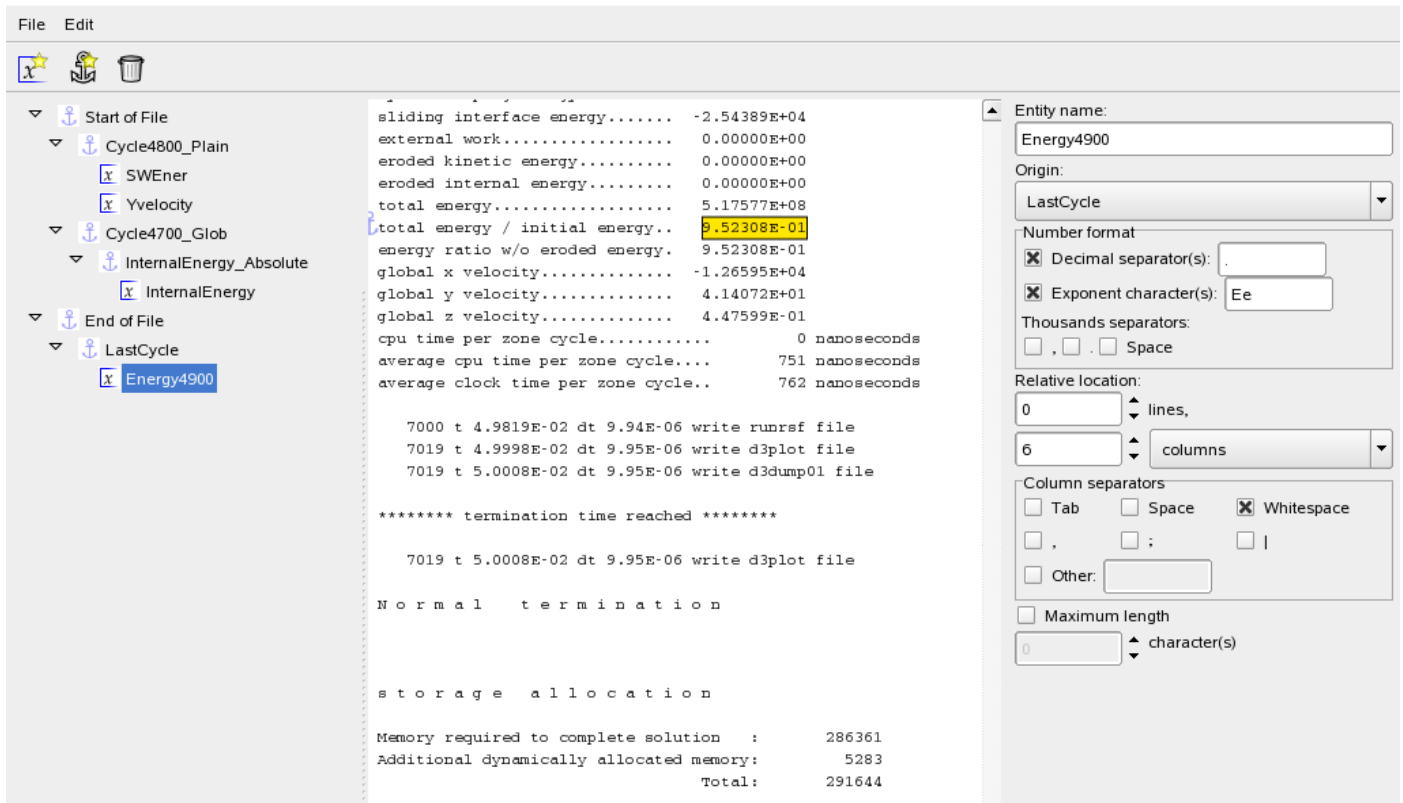
Memory required to complete solution : 286361
Additional dynamically allocated memory: 5283
Total: 291644

Timing information
CPU(seconds) %CPU Clock(seconds) %Clock
-----
Initialization ..... 8.0000E-02 1.27 7.4480E-02 1.27
Element processing ... 3.3000E+00 52.55 3.3588E+00 52.55
Binary databases ..... 2.1000E-01 3.34 1.0776E-01 1.66
ASCII database ..... 6.6000E-01 10.51 7.9649E-01 12.51

```

**Figure 6-23: 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.



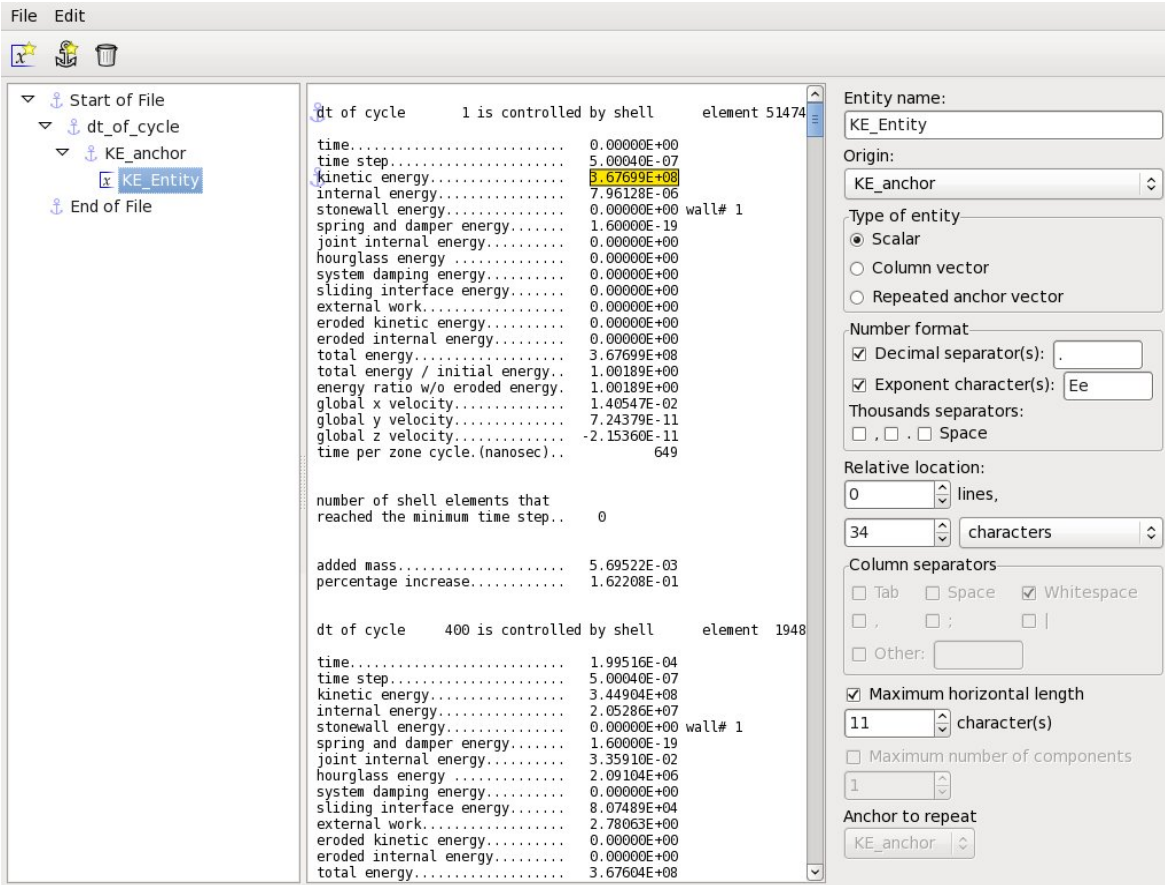
**Figure 6-24: GenEx dialog; definition of an entity**

- 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 `d3h.sp`. 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.

### 6.13.5. An example using "Repeated anchor vector" to extract histories

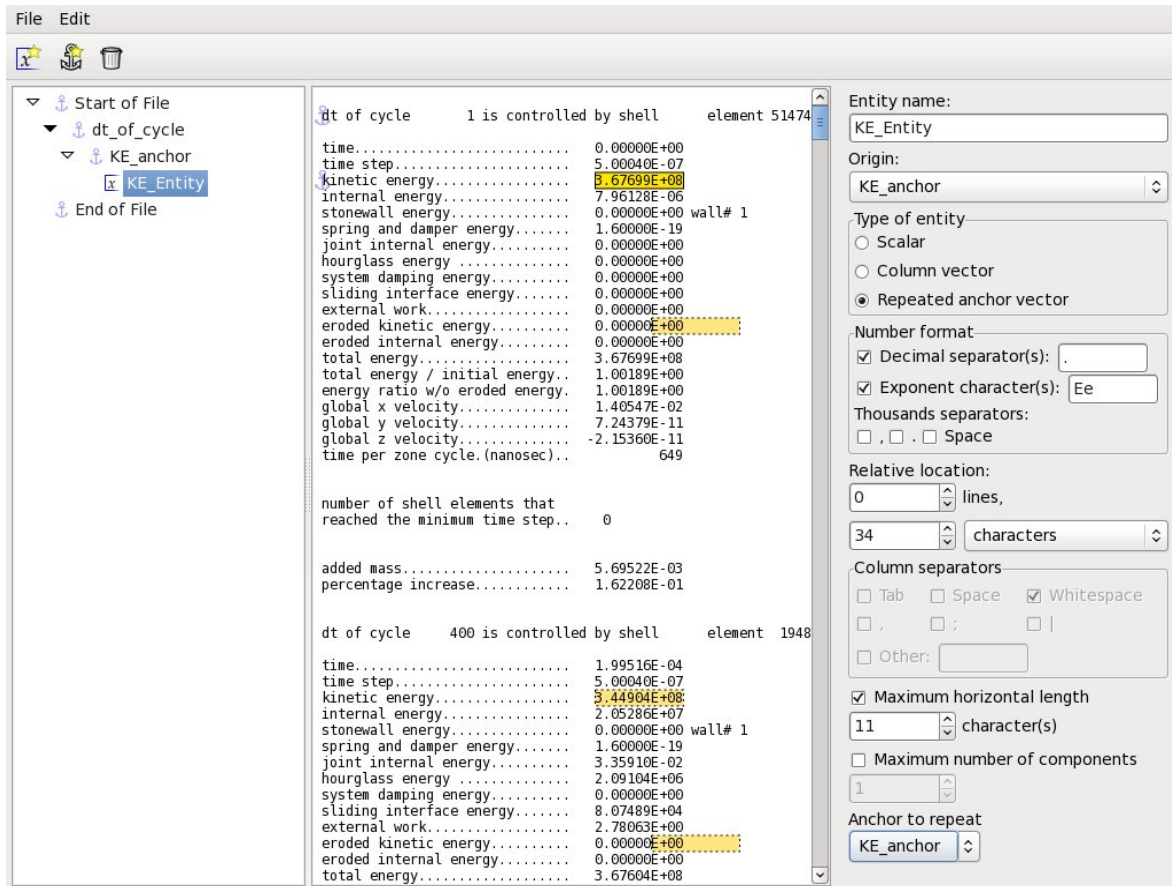
In this example we will use GenEx to extract histories of the value for "kinetic energy" in the "glstat" file created by LS-DYNA. We first start by creating the anchor `dt_of_cycles`. This anchor will be the base for further anchors. With this anchor as parent we now create the `KE_anchor` to search for the string we are looking for, in this case "kinetic energy".



**Figure 6-25: GenEx dialog; definition of an entity**

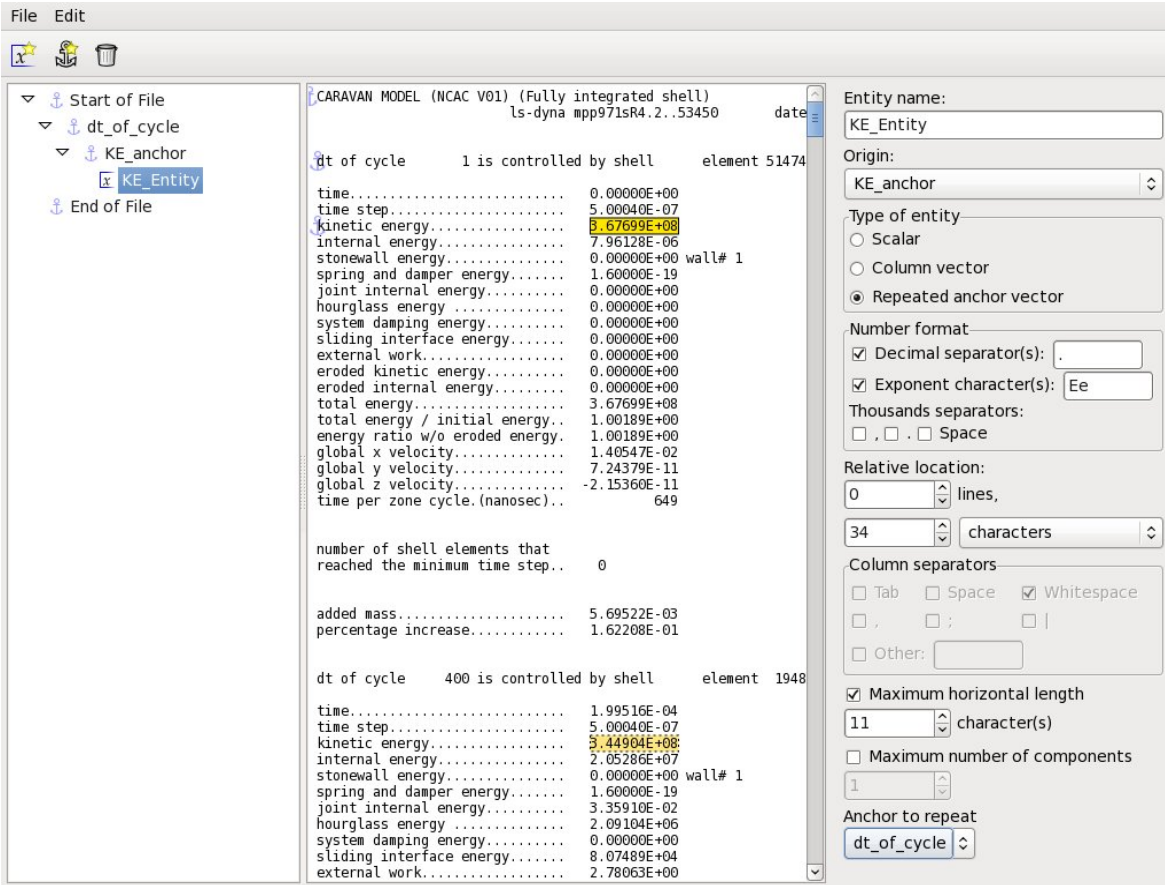
As seen in the screenshot above, this entity is of the Scalar type and needs to be changed to **Repeated anchor vector**. When creating a repeated anchor vector the default value for **Anchor to repeat** is the parent of the entity. Since "kinetic energy" appears twice between every `dt_of_cycle` the 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.



**Figure 6-26: GenEx dialog; definition of an repeat anchor 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 6-27: GenEx dialog; definition of a history*

We are now finished with the GenEx part and the file can be saved.

### 6.13.6. 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**.



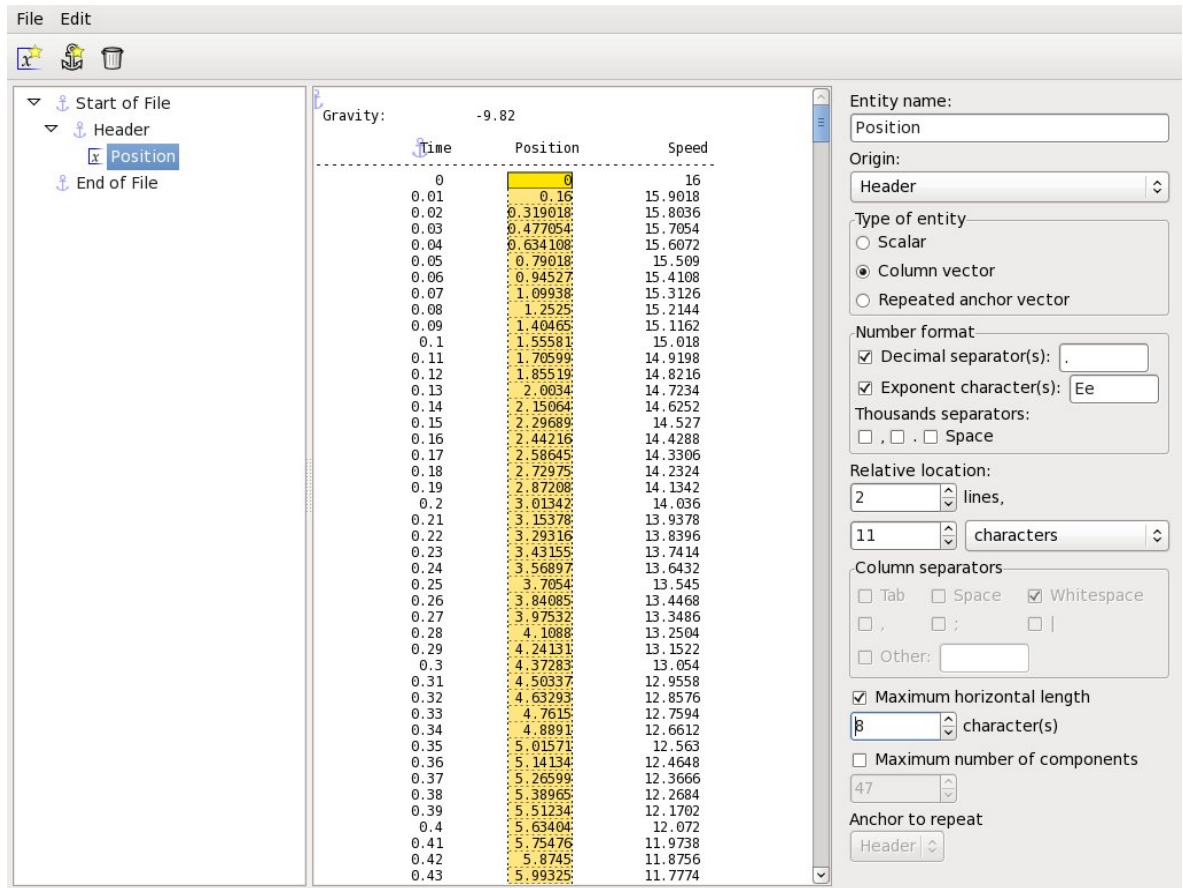
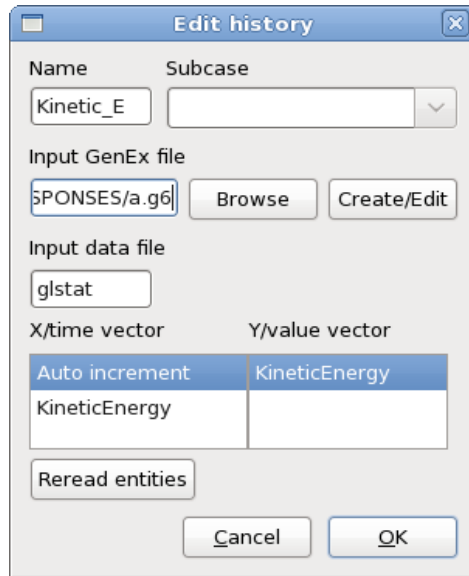


Figure 6-28: GenEx dialog; definition of a column vector entity

### 6.13.7. 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 6-29: 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.

## 6.14. 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 6-30.

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

- o The C language:

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

or

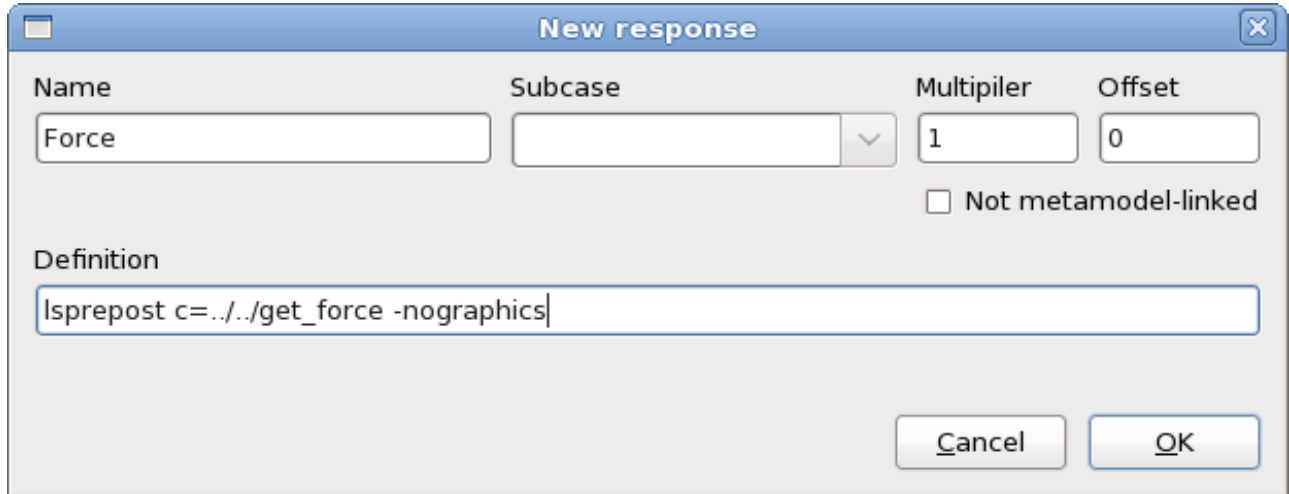
```
fprintf (stdout, "%lf\n", output_value);
```

- o The FORTRAN language:

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

- o The Perl script language:

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



**Figure 6-30: 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 rforc open rforc 0
ascii rforc plot 4 Ma-1
xyplot 1 savefile xypair LsoptHistory 1
deletewin 1
quit
```

produces the `LsoptHistory` file. See Figure 6-30 for the LS-PREPOST command.

*Note :* The `rforc` history in this example can be obtained more easily by direct extraction (see Section 6.2.1 and Appendix A.1 : Binout Histories.)

*Remark:*

1. An alias must not be used for an interface program.

5. The program should be run in batch mode.
6. The program is called from the run directories. This has to be considered if relative paths are used.

## 6.15. Nastran Frequency

The Nastran Frequency feature allows the user to extract the frequency, matched mode number or MAC value from the Nastran database. This interface is similar to the LS-DYNA Frequency interface. Please refer to Section 6.2.5.

The screenshot shows a dialog box titled "New response" with the following fields and options:

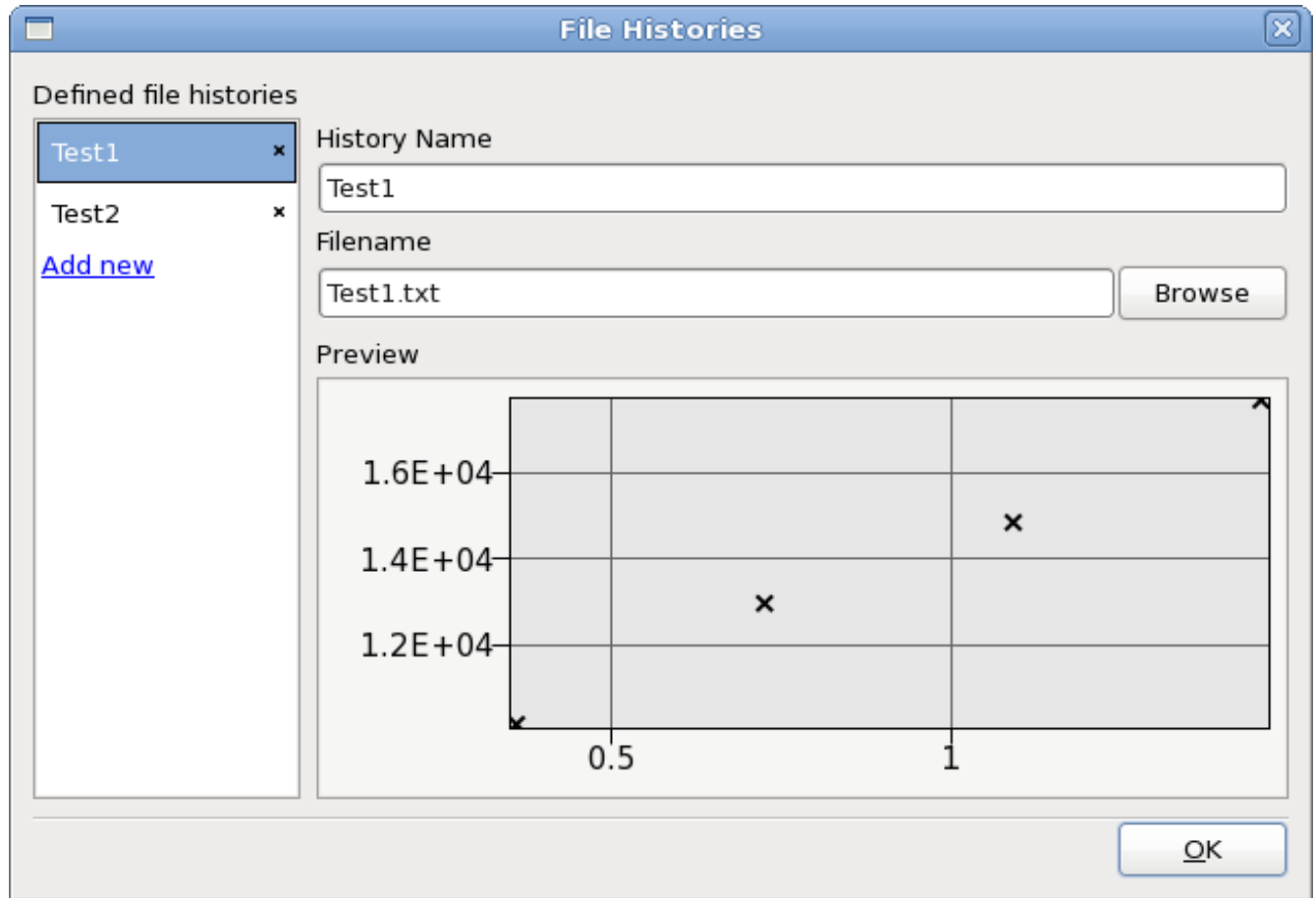
- Name:** Frequency
- Subcase:** (empty)
- Multiplier:** 1
- Offset:** 0
- Not metamodel-linked
- Baseline Mode Number:** 3
- Modal Output Option:**
  - Frequency of Mode
  - New Mode Number
  - Modal Assurance Criterion
- Buttons:** Cancel, OK

*Figure 6-31: Interface for Extraction of Frequencies from Nastran results*

## 6.16. File Histories

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. History files are typically used to import test data for parameter identification problems.

File histories are global curves, they are neither sampling nor stage dependent, hence they are not listed in the *Stage* dialog history list.



**Figure 6-32: File Histories**

*File History Text File Example:*

Time	Displacement
1.2,	143.97
1.4,	156.1
1.7,	923.77

# 7. 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 deactivation of variables for selected samplings.

Resource definitions and other global features are also available in this dialog.

## 7.1. Parameter Setup

Parameters defined in the input files of the stages are automatically displayed in the **Parameter Setup** panel, Figure 7-1. 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.

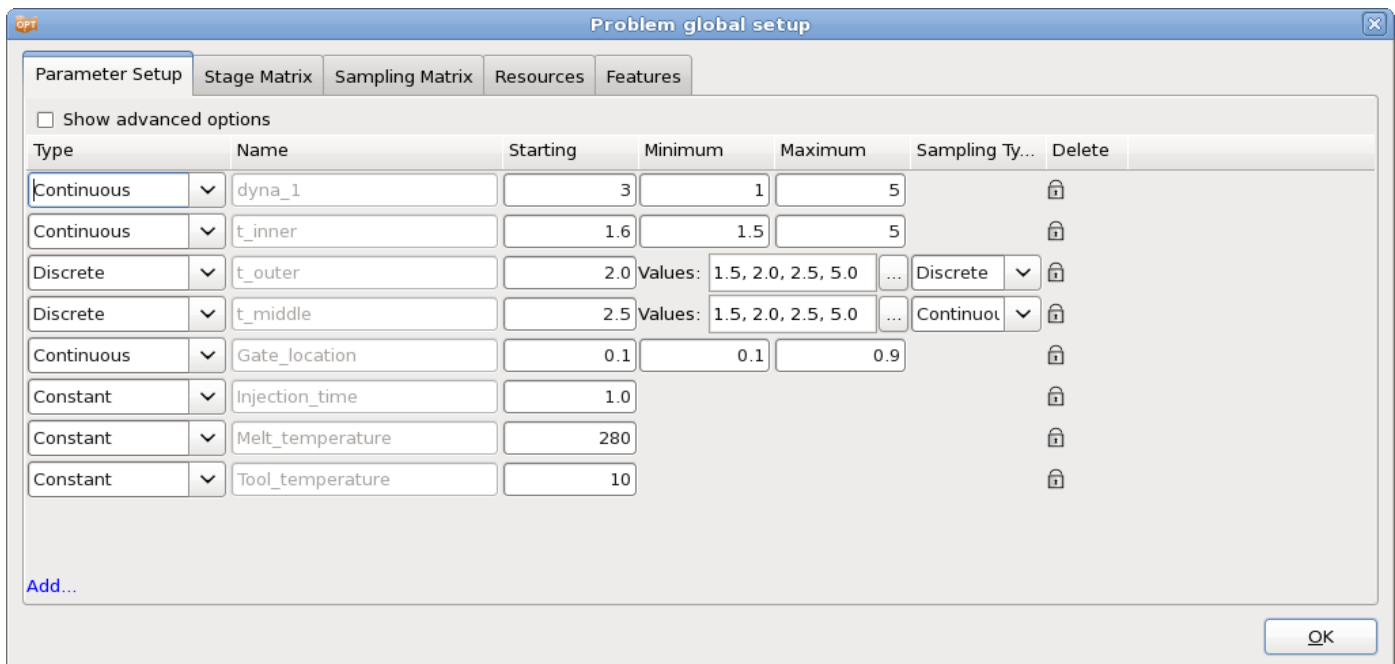


Figure 7-1: Setup Dialog – Parameter Setup panel in LS-OPTui

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 7-1.

Advanced options, such as initial range, that are not required can be specified by selecting the *Show advanced options* checkbox, Table 7-2.

Additional (non-file) parameters, although unusual, can be defined using the *Add...* button at the bottom of the panel.

**Table 7-1: Parameter Setup options to be specified for each parameter**

Option	Description	Reference
Type	Parameter type:	
	Continuous	Continuous variable -
	Constant	Constant value Section 7.1.1
	Dependent	Parameter depending on other parameters Section 7.1.2
	Discrete	Discrete variable Section 7.1.3
	String	Discrete variable using string values Section 7.1.3
	Noise	Probabilistic variable described by a statistical distribution Section 7.1.4
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 7.1.3
Definition	Mathematical expression specifying a dependent parameter	Section 7.1.2
Distribution	Statistical distribution used to define a probabilistic variable	Section 7.1.5
Sampling Type	Sampling type for discrete variable: continuous or discrete	Section 7.1.3

**Table 7-2: Parameter Setup advanced options**

Option	Description	Reference
Init. Range	Size of subregion of the design space used in the first iteration	Section 7.1.6
Saddle Direction	Saddle direction specification used for worst-case design	Section 7.1.7

**Table 7-3: Parameter Setup options**



Option	Description	Reference
Show advanced options	Shows Init. Range and Saddle Direction option for each parameter	Table 7-2
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.	-

### 7.1.1. Constants

Each variable above can be modified to be a constant. Constants are used:

1. to define constant values in the input file such as  $\pi$ ,  $e$  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 7.3)

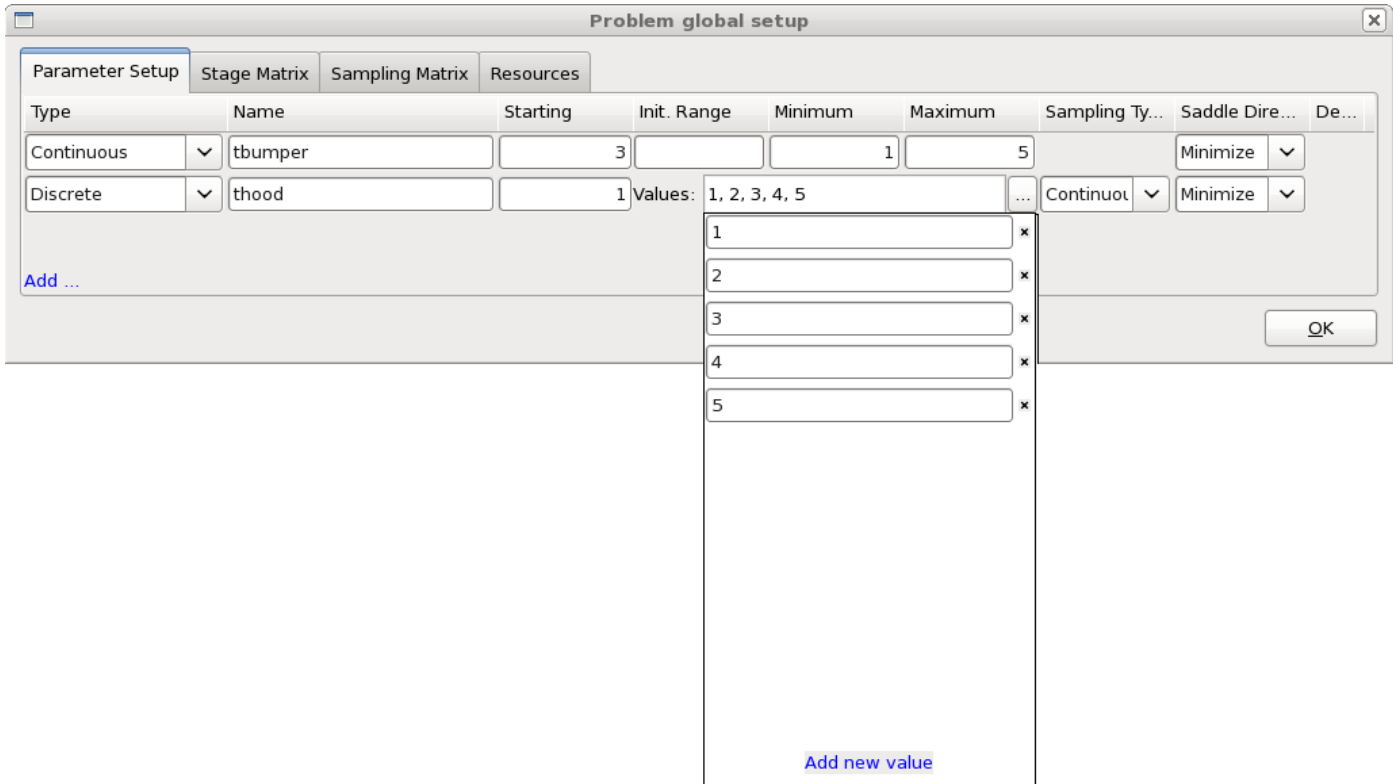
### 7.1.2. Dependent variables

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.

### 7.1.3. Discrete and String variables



*Figure 7-2: Definition of discrete values*

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 7-2. 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.

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.

### 7.1.4. 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 12) 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 7-3).

### 7.1.5. Probabilistic distributions

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 7-3). Several types of distributions are available in LS-OPT. Further details of how to set up probabilistic variables and distributions are provided in Chapter 12.

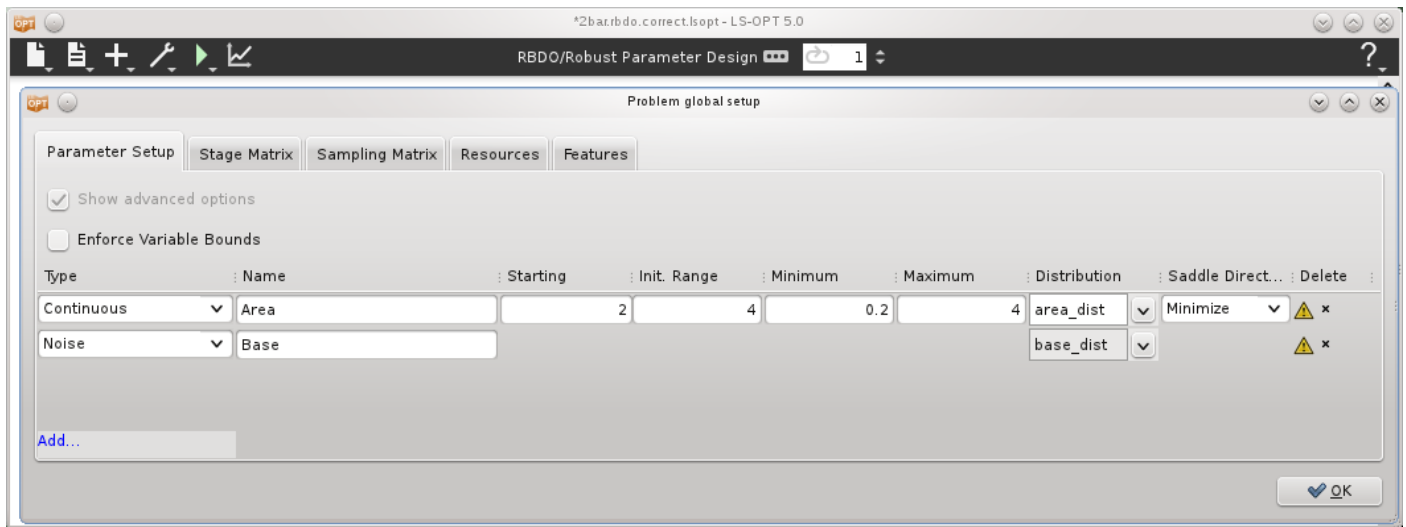


Figure 7-3: Parameter setup panel for probabilistic tasks

### 7.1.6. Size and location of initial region of interest (range)

If an initial range is specified, the initial subregion is defined as  $[starting - range/2, starting + range/2]$ .

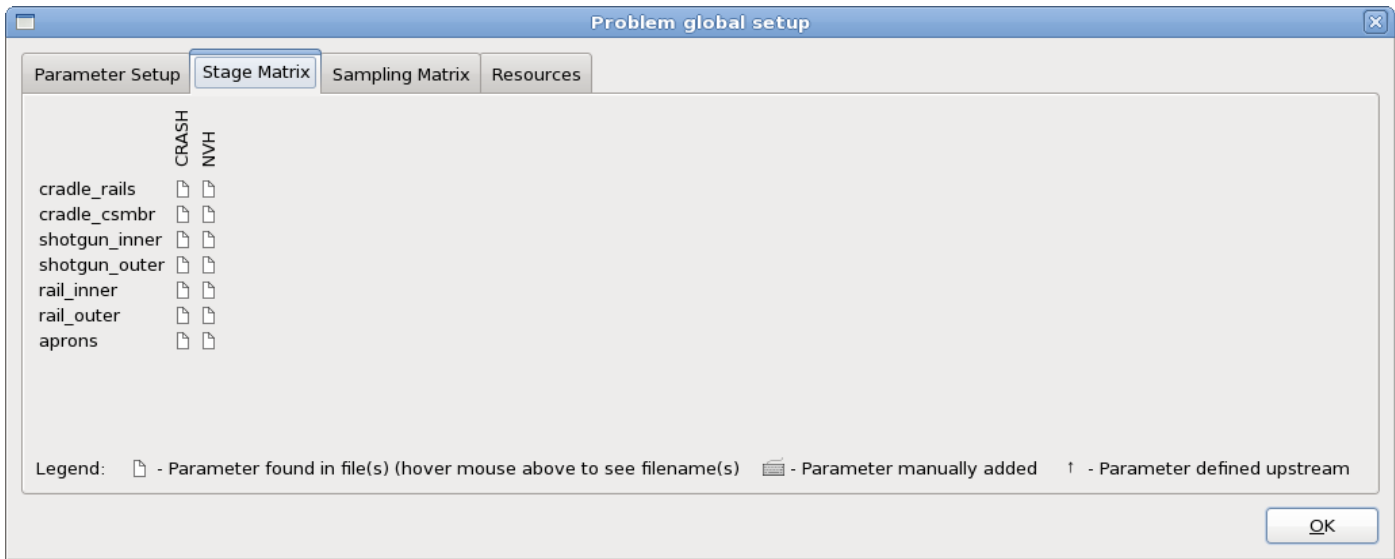
Remarks:

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.

### 7.1.7. 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.

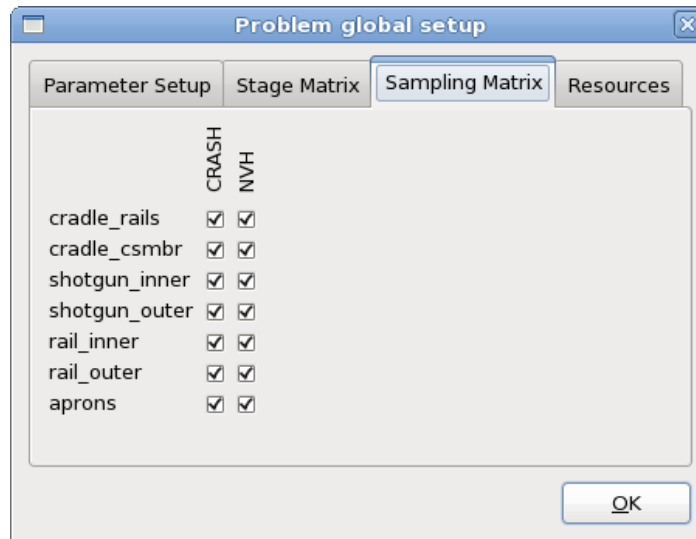
## 7.2. Stage Matrix



**Figure 7-4: 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.

## 7.3. Sampling Matrix



**Figure 7-5: 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

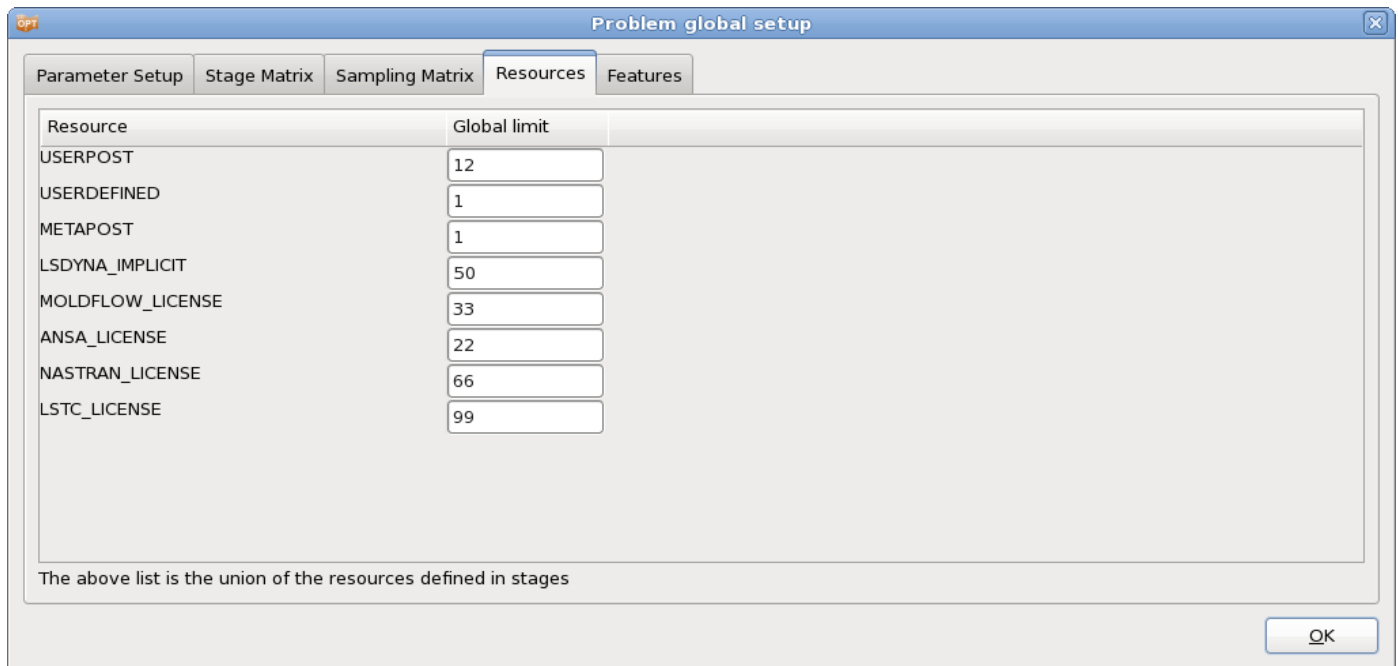
Matrix. If a variable is absent in a particular sampling, it assumes the current global value as generated by the previous iteration for substitution in the input files of the next iteration. The number of variables selected for a sampling directly affects the number of sampling points (and hence the computational effort) required for that sampling. Each column is coupled to the **Active Variables** tab of the respective **Sampling Dialog**, Section 8.4.

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 cannot be changed between iterations.

See Section 17.5 for an MDO example.

## 7.4. Resources



*Figure 7-6: 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.

## 7.5. Features



*Figure 7-7: Setup – Features*

Sampling independent features are available in the **Features** tab of the **Setup** dialog, Figure 7-7.

### 7.5.1. 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. 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 coordinates are specified as one row for each design point. See example below.
2. Use the *Setup* dialog *Repair* option *Evaluate Metamodels*.
  - *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
```

# 8. 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 8-1. The terms *point selection* and *experimental design*, are used interchangeably.

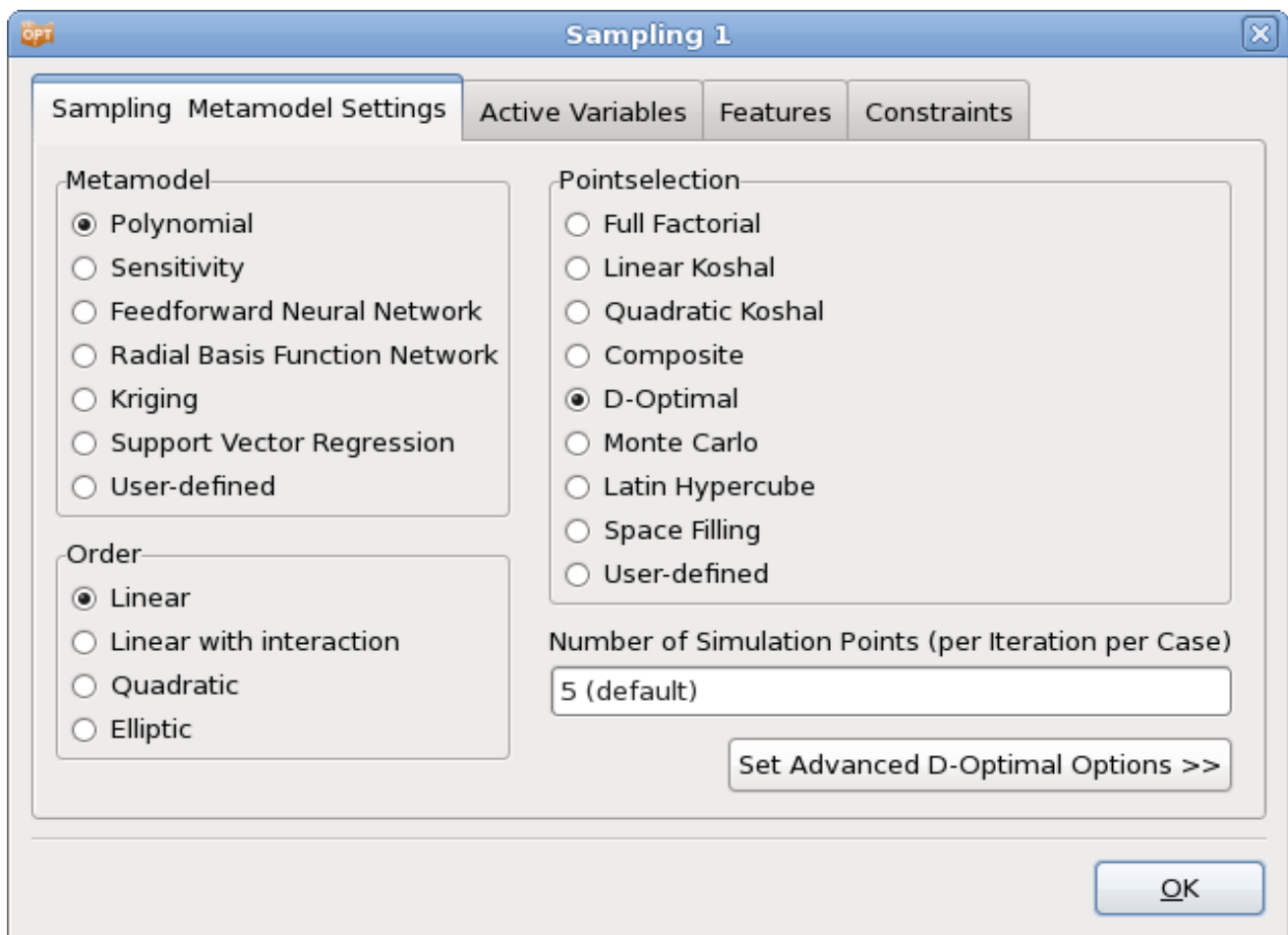


Figure 8-1: Sampling dialog – metamodel and point selection settings



## 8.1. Metamodel types

The user can select one of the metamodel types shown in Figure 8-1 and Table 8-1, 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 8-1.

**Table 8-1: Sampling dialog options – Metamodel types**

Metamodel Type	Description	Reference
Polynomial	Polynomial approximations up to quadratic order	Section 8.1.1
Sensitivity	Uses gradients to determine linear metamodels.	Section 8.1.2
Feedforward Neural Network	An artificial Neural network with sigmoid basis functions	Section 8.1.3
Radial Basis Function Network	A Neural Network with radial basis functions	Section 8.1.3
Kriging	A Gaussian process. Form of Bayesian inference.	Section 8.1.4
Support Vector Regression	Support Vector Regression	Section 8.1.5
User-defined	Interface for user-defined, dynamically linked metamodel.	Section 8.1.6

### 8.1.1. Polynomial

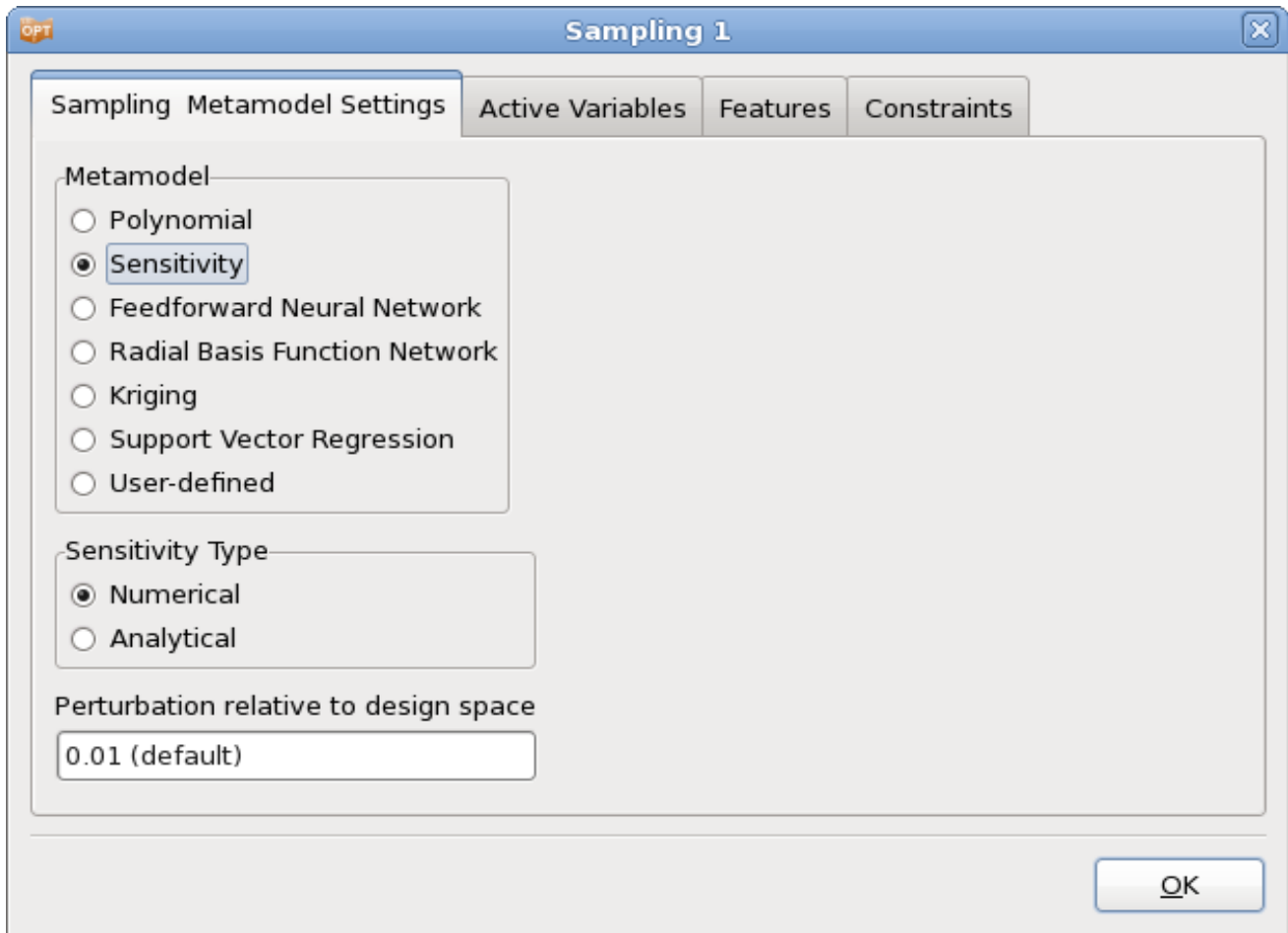
When polynomial response surfaces are constructed, the user can select from different approximation orders. The available options are linear, linear with interaction (linear and off-diagonal terms), elliptic (linear and diagonal terms) and quadratic, Section 20.1.1. In the *Sampling* dialog, the approximation order is set in the *Order* field, Figure 8-1. 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.

The polynomial terms can be used during the variable screening process (see Section 20.4) 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 14.3.4).

The recommended point selection scheme for polynomial response surfaces uses the  $D$ -optimality criterion (Section 8.3.2).

### 8.1.2. Sensitivity

In this approach, sensitivities are used to generate linear metamodels. Both analytical and numerical sensitivities can be used for optimization, Figure 8-2.



*Figure 8-2: 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 17.7.

## 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 8-2.

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  $\varepsilon$  is relative to the design space size. The same value applies to all the variables. The value of  $\varepsilon$  is assumed to be 0.001.

### 8.1.3. 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 8-3 and Figure 8-4, respectively. The recommended Point Selection scheme for feedforward neural networks and radial basis functions is the space filling method (which is also the default).

#### FFNN 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.

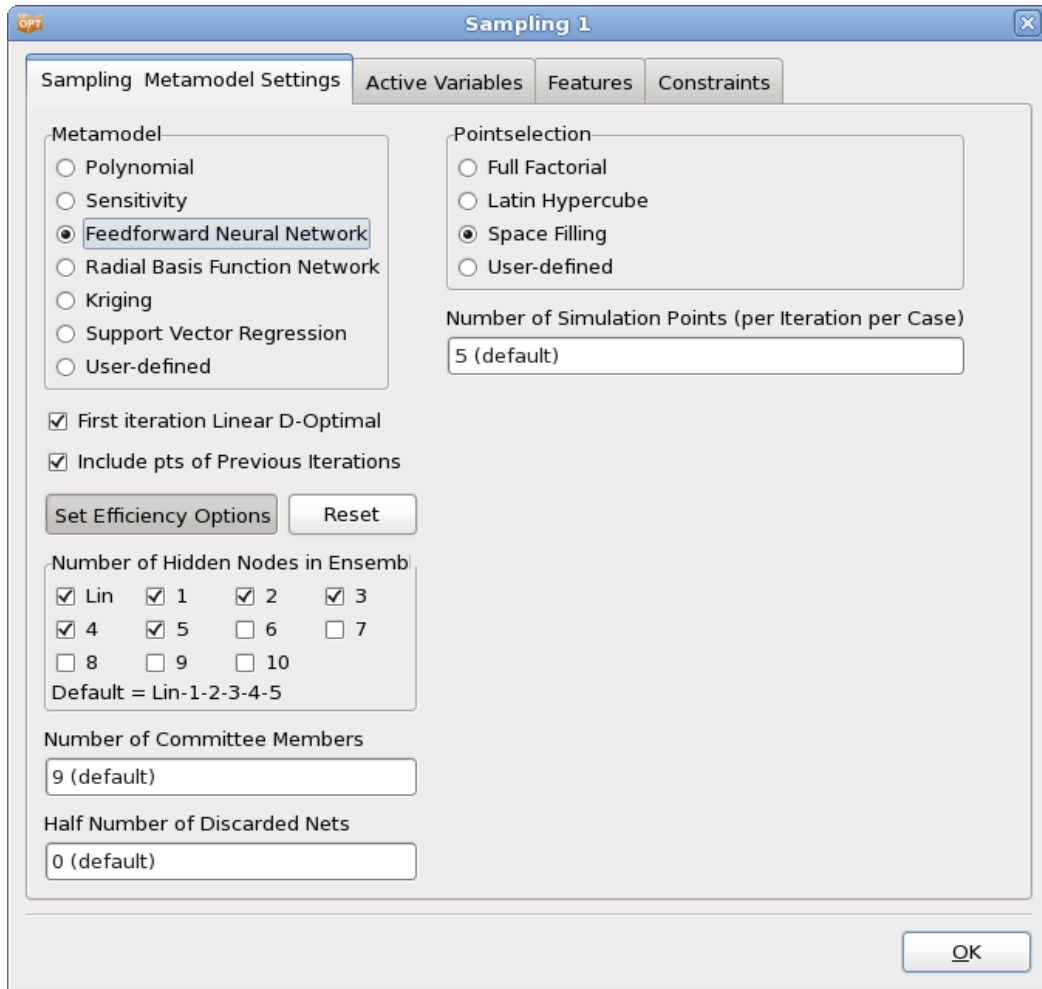
*Committee size.* The default committee size as specified above is largely required because the default number of points when conducting an iterative optimization process is quite small. Because of the tendency of NN's to have larger variability when supplied with fewer points, committees are relied 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 8-3. The available options are explained in Table 8-2.



*Figure 8-3: Feedforward Neural Network Efficiency Options*

**Table 8-2: 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 (see Section 21.1.2), 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.

Please refer to Sections 21.3 and 22.5 for recommendations on how to use metamodels.

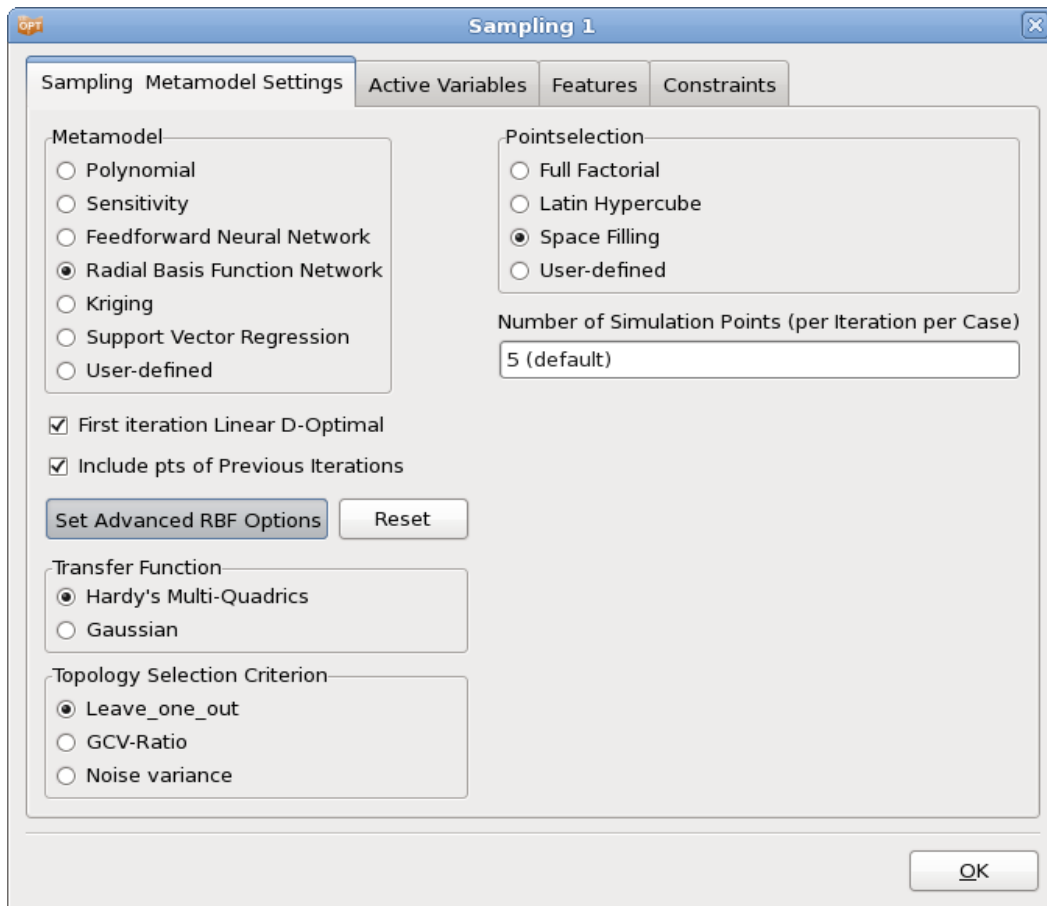
### **Advanced RBF options: Basis functions and optimization criterion for RBF\***

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 8-4.

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 8-4. The available options are described in Table 8-3.

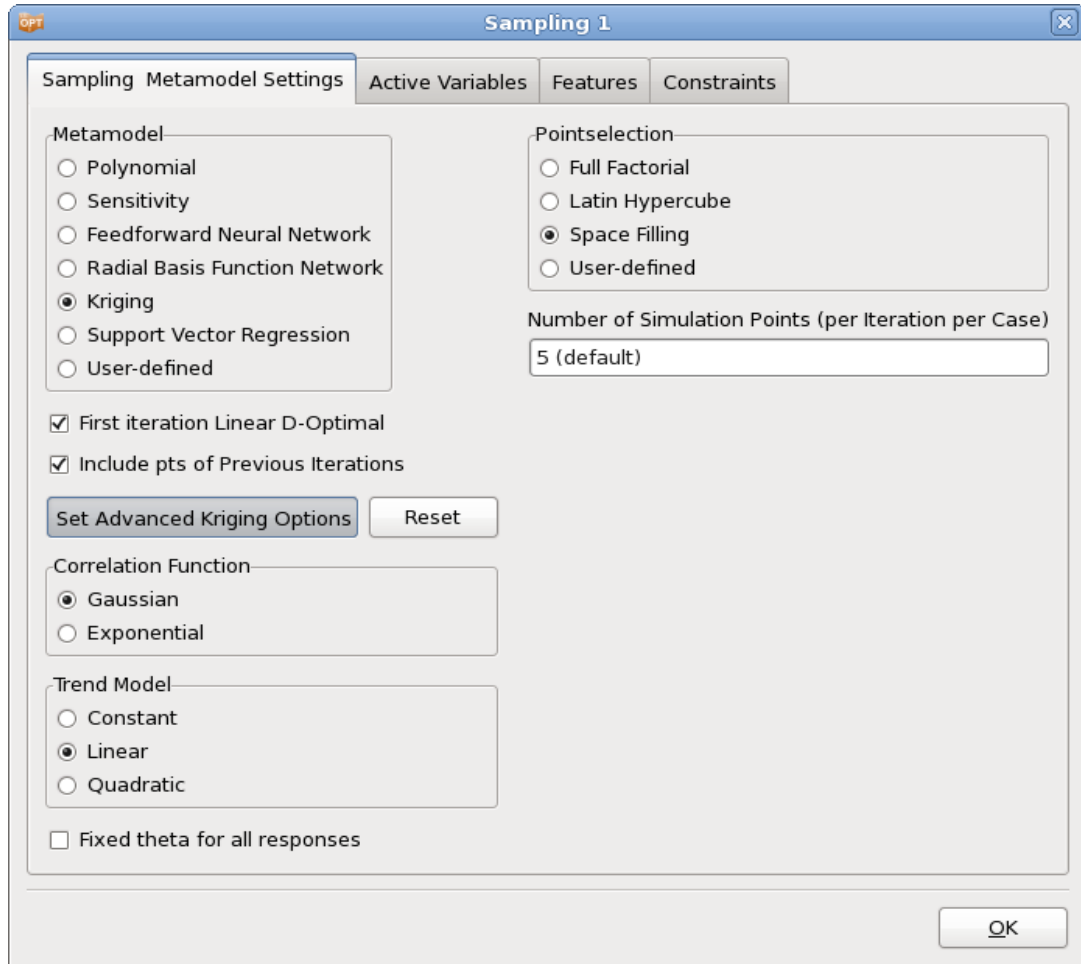
**Table 8-3: RBF Advanced Options**

Option	Description	Option	Description
Transfer Function	Basis function	Hardy's Multi-Quadrics	$g_h(x_1, \dots, x_K) = \sqrt{1 + (r^2 / \sigma_h^2)}$ .
		Gaussian	$g_h(x_1, \dots, x_K) = \exp[-r^2 / 2\sigma_h^2]$
Topology Selection Criterion	Optimization criterion	Leave-one-out	Generalized cross-validation error (PRESS)
		GCV-Ratio	Pointwise ratio of the generalized cross validation error
		Noise variance	Variance of the fitting error



**Figure 8-4: Radial Basis Function Network Advance Options**

### 8.1.4. Kriging parameters



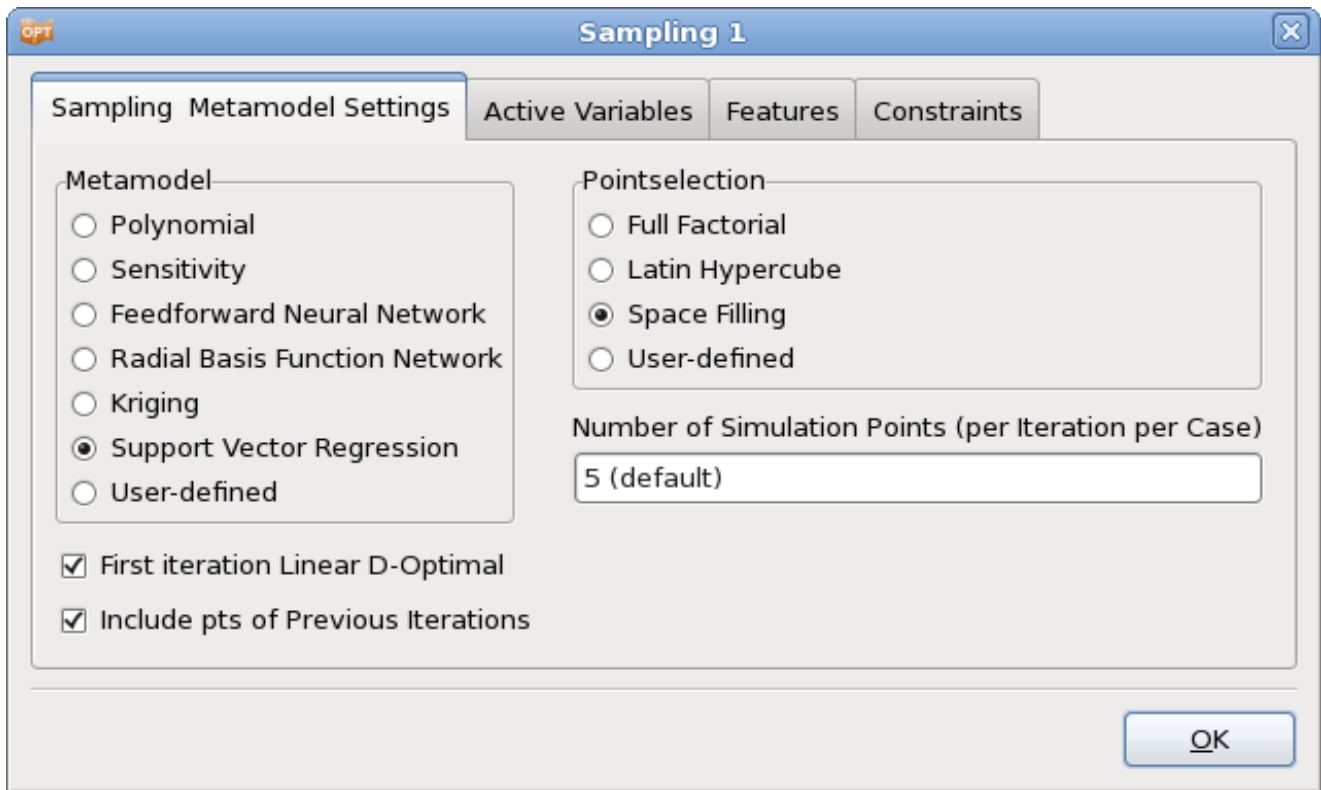
**Figure 8-5: Kriging Advanced Options**

The Kriging fit depends on the choice of appropriate correlation function and the trend model, Section 21.2. 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 8-4.

**Table 8-4: Advanced Kriging Options**

Option	Option	Description
Correlation Function	Gaussian, Exponential	Correlation function used in stochastic component of metamodel function, see Section 21.2.
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.

### 8.1.5. Support Vector Regression

**Figure 8-6: Metamodel selection Support Vector Regression**

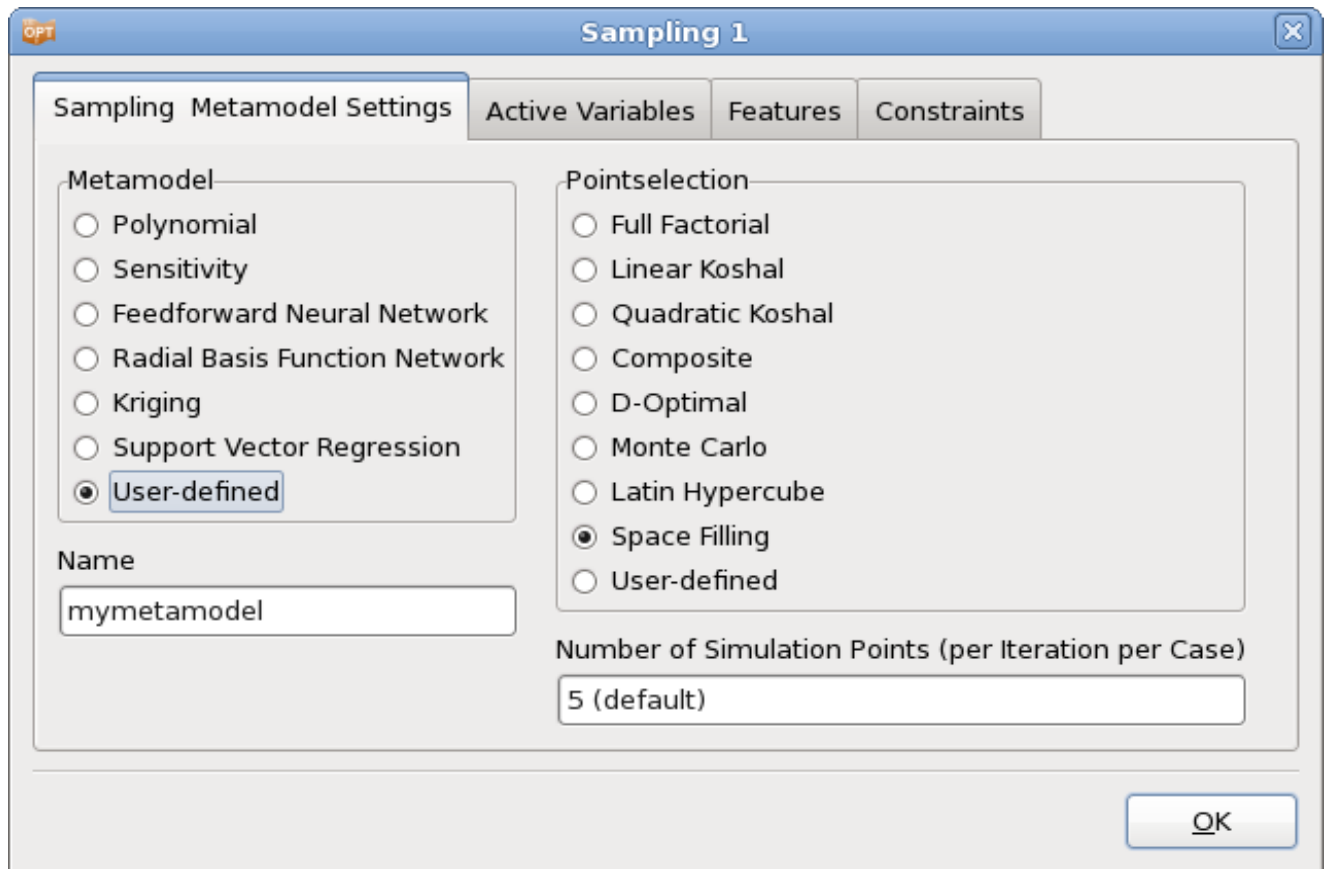


The support vector regression fit depends on the choice of appropriate kernel function (similar to correlation function), Section 21.3. Two kernel functions available for selection are Gaussian and polynomial. The available options are displayed in Table 1-5

**Table 8-5: Advanced Support Vector Regression Options**

Option	Option	Description
Kernel Function	Gaussian, Polynomial	Basis function used in SVR expansion that maps the input variable space to a high dimensional feature space, see Section 21.3 .

### 8.1.6. User-defined metamodel\*



**Figure 8-7: User Defined Metamodel Options**

The user-defined metamodel distribution is available for download at [http://ftp.lstc.com/user/ls-opt/Add\\_On\\_Libraries/](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 8-7.

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.

## 8.2. 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 8-5.

**Table 8-5: FFNN, RBF, Kriging and SVR options**

Option	Description	Reference
First iteration Linear D-Optimal	Use linear metamodels and the <i>D</i> -optimality point selection criterion for the first iteration instead of the selected types.	8.2.1
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.	8.2.2

### 8.2.1. 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.

### 8.2.2. 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).

## 8.3. Point selection schemes

### 8.3.1. Overview

Table 8-6 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.

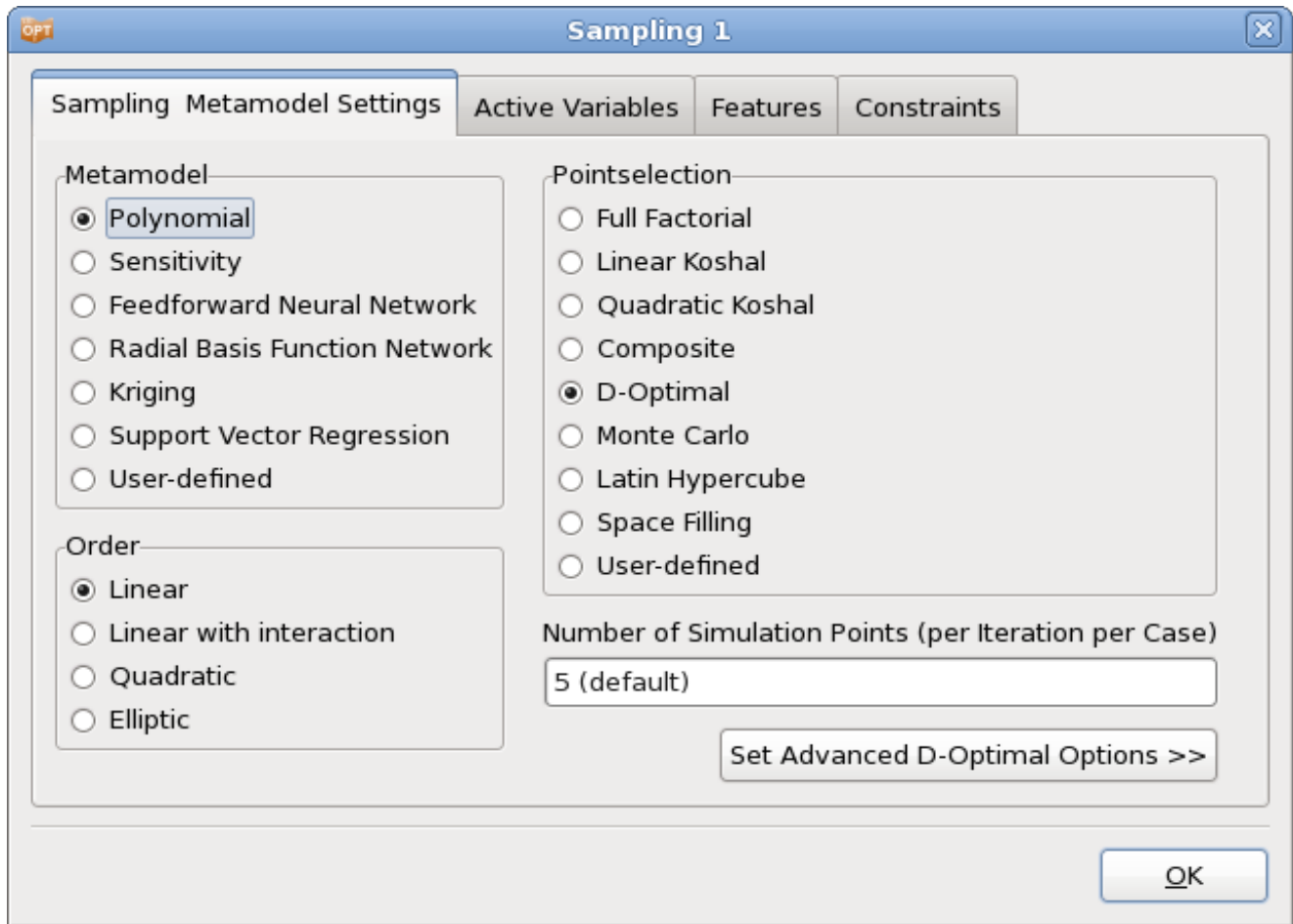
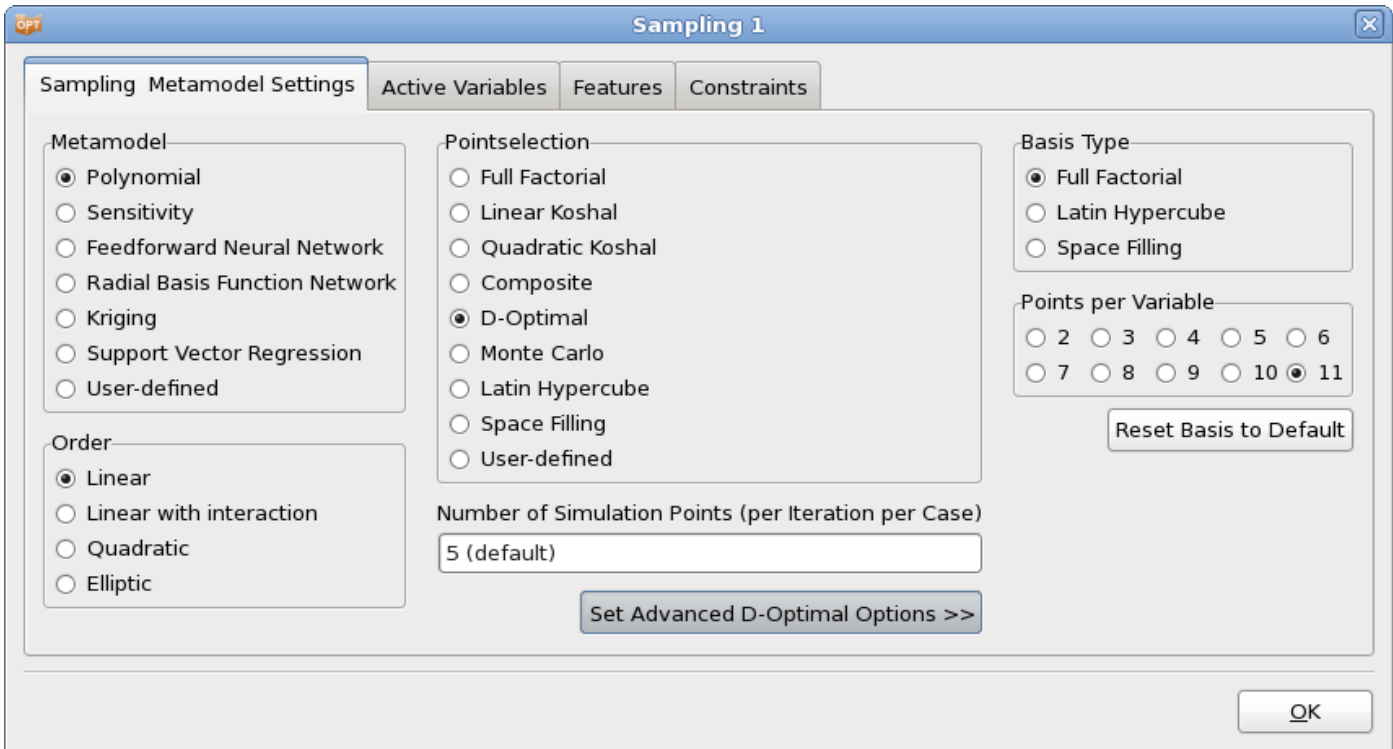


Figure 8-8: Point selection schemes

**Table 8-6: Point selection schemes**

Point Selection Scheme	Description	Reference
Full Factorial	-	Section 20.2.1
Linear Koshal	Saturated design for first order Polynomials	Section 20.2.2
Quadratic Koshal	Saturated design for quadratic Polynomials	Section 20.2.2
Composite	Central Composite design	Section 20.2.3
<i>D</i> -optimal	Design obtained by minimizing the determinant of the moment matrix	Section 8.3.2, Section 20.2.4
Latin Hypercube	Stratified random design	Section 8.3.3, Section 20.2.5
Monte Carlo	Random design	
Space Filling	Design obtained by maximizing the minimum distance between any two points.	Section 8.3.4, Section 20.2.6
Space Filling of Pareto Frontier	Design obtained by maximizing the minimum distance between any two points sampled from the Pareto Optimal Frontier.	Section 8.3.5
User-defined	-	Section 8.3.6

### 8.3.2. D-Optimal point selection



**Figure 8-9: 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 D-Optimal 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  $\text{int}(1.5(n + 1)) + 1$  for linear,  $\text{int}(1.5(2n + 1)) + 1$  for elliptic,  $\text{int}(0.75(n^2 + n + 2)) + 1$  for interaction, and  $\text{int}(0.75(n + 1)(n + 2)) + 1$  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.

### 8.3.3. 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 algorithm, see the description of Algorithm 2 in Section 20.2.6.

### 8.3.4. Space Filling

The 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 20.2.6. 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 8.1.3).

### 8.3.5. 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

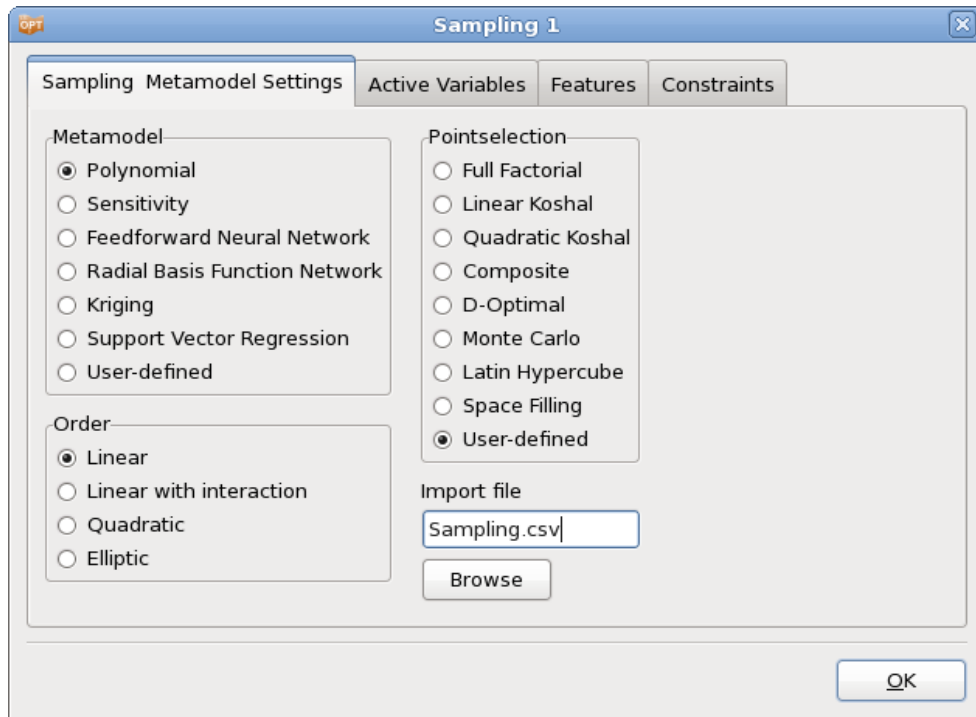
The following procedure can be followed to conduct simulations based on the POF. It is assumed that the user has conducted one or more metamodel-based iterations and that the POF has been created based on the metamodel.

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.
  - 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.

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.

### 8.3.6. User-defined point selection



**Figure 8-10: 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.

#### 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) or string variables (st), respectively. 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:

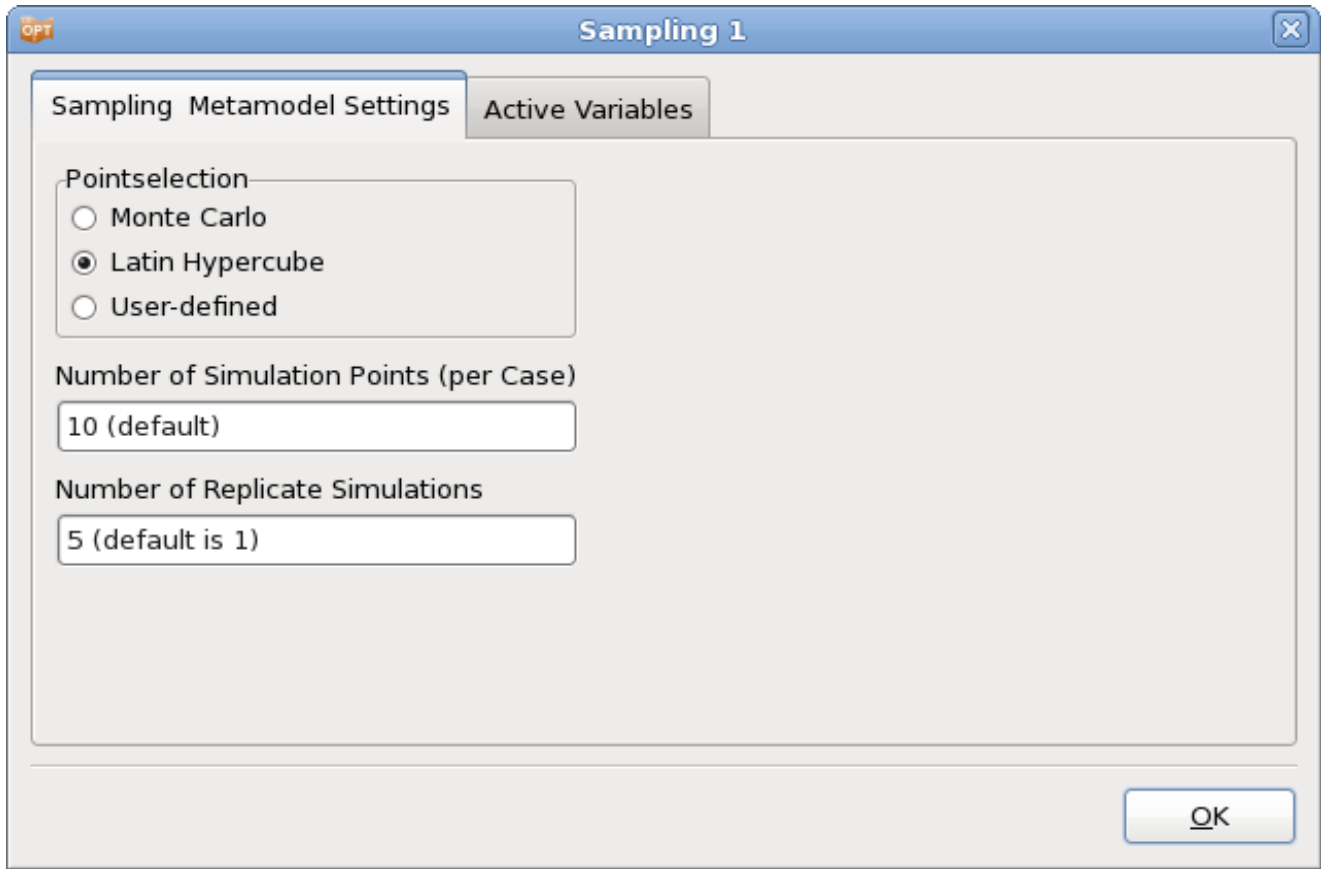
```
lso_numvar 2
lso_numpoints 3
lso_varname          t_bumper    t_hood
lso_vartype          dv            nv
This is a comment    lso_point    1.0        2.0
                    lso_point    2.0        1.0
                    lso_point    1.0        1.0
```

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.

### 8.3.7. 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, Figure 8-11. 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.



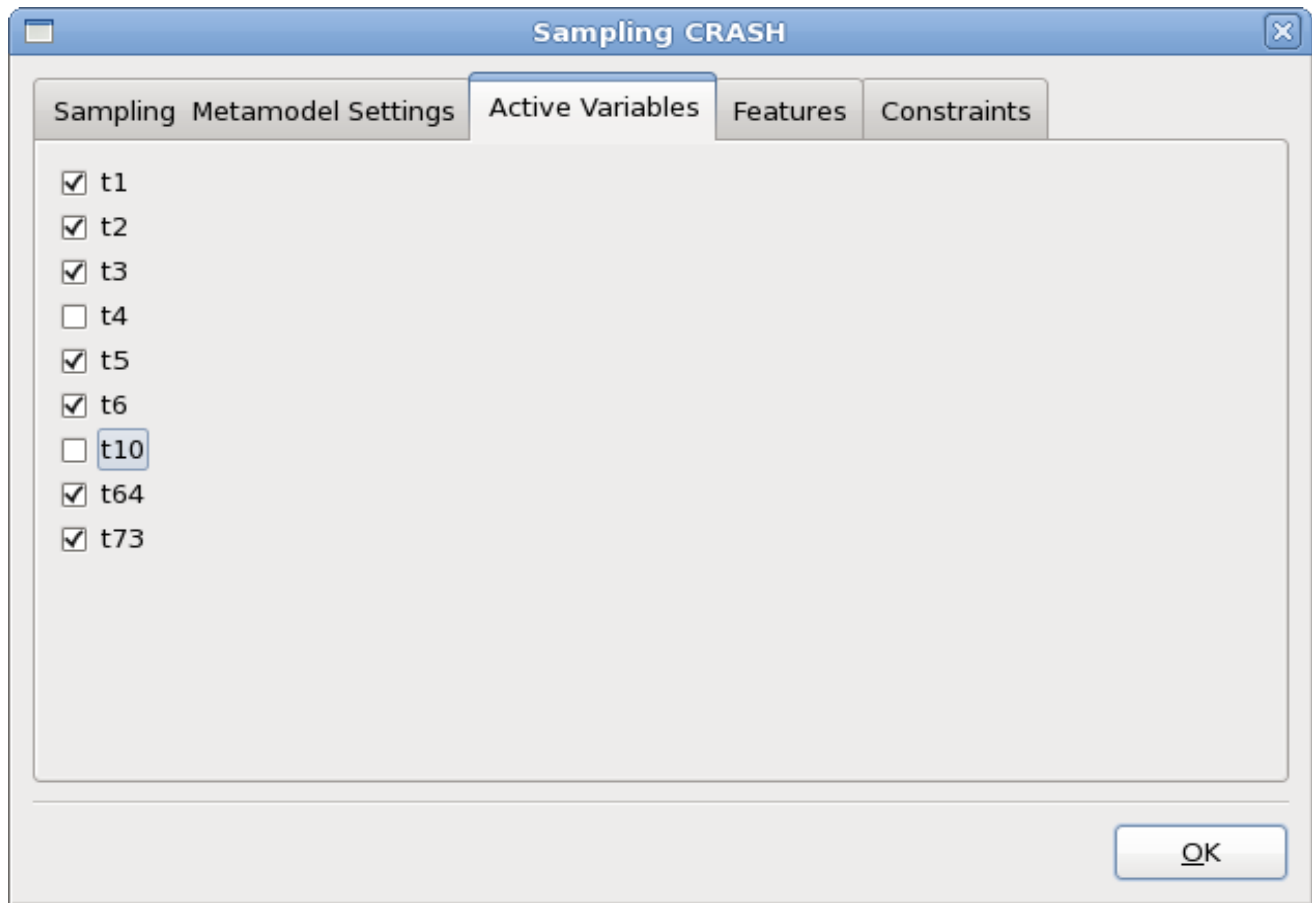
*Figure 8-11: Sampling Dialog options for direct Monte Carlo Analysis*

So, in the above, the original experimental design has 10 point, hence 50 FEA evaluations will be done. See also the example in Section 18.1.

### 8.3.8. Remarks: Point selection

1. The database files `Experiments_n.csv`, `AnalysisResults_n.lsox` 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.

## 8.4. Active Variables



**Figure 8-12: Sampling Dialog: Active Variables panel**

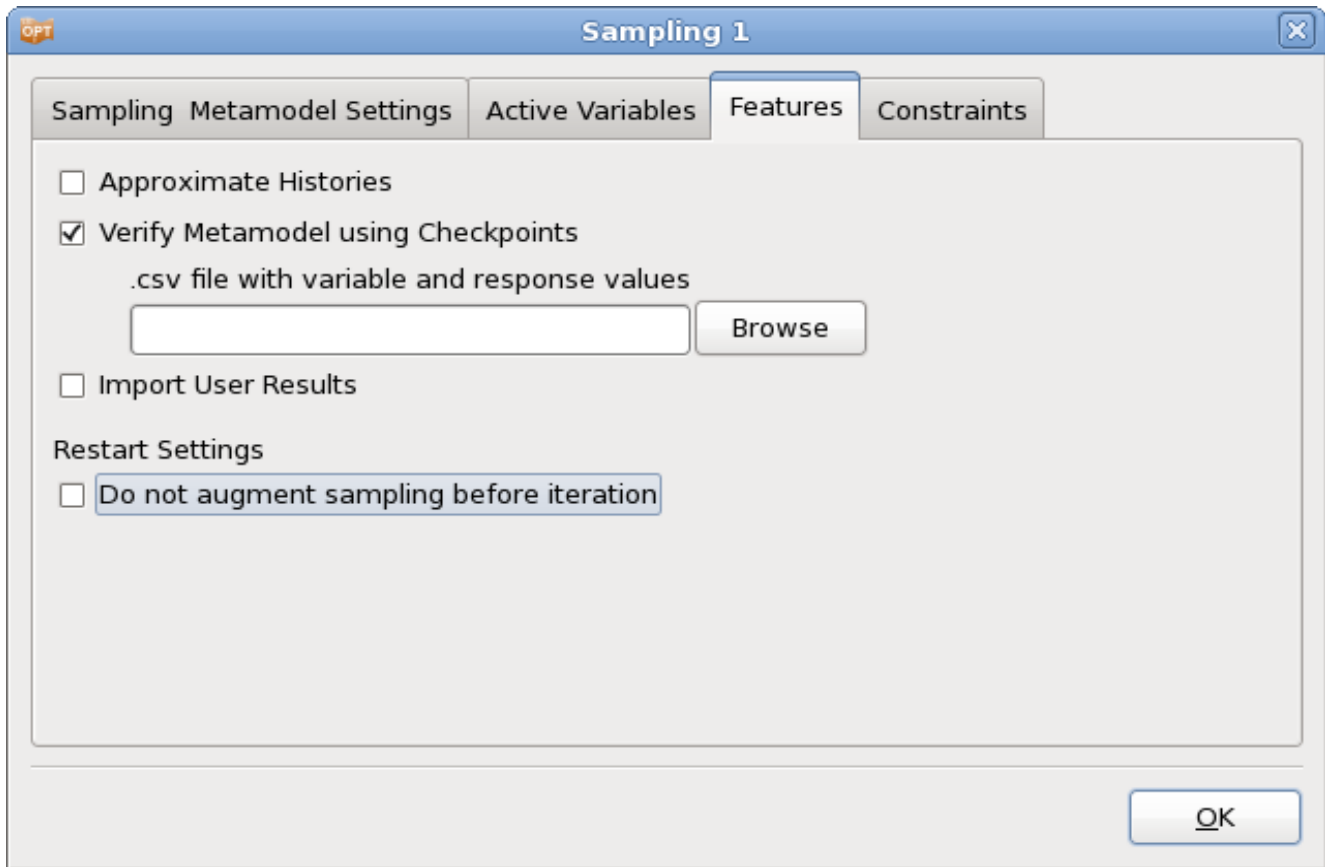
The *Active Variables* panel shows a list of all previously defined variables, Figure 8-12. Each variable has a checkbox that allows the user to select or deselect it for the respective sampling. Deselected variables are treated as constants using the optimal value of the previous iteration.

The selection in the *Active Variables* dialog is coupled to the respective column of the *Sampling Matrix* shown in the *Setup Dialog*, Section 7.3.

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 cannot be changed between iterations.

## 8.5. Sampling Features



*Figure 8-13: Sampling Features*

**Table 8-7: Sampling Features**

Feature	Description	Reference
Approximate Histories	Extension of the metamodel concept to curves.	Section 8.5.1
Verify Metamodel using Checkpoints	Calculate error measures of the metamodel using a given metamodel and set of checkpoints (variables and response values)	Section 8.5.1
Import User Results	Import table of design points (variable and response values)	Section 8.5.3
Do not augment sampling before iteration	Use larger number of sampling points from a specified iteration	Section 8.5.4

### 8.5.1. 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 8-13. 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.

### 8.5.2. 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** panel. The file must be in .csv format although spaces, commas or tabs are allowed as delimiters. 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" and "rs". 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 to run (see Section 3.5).

Cases without checkpoint files will be ignored.

*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
```

### 8.5.3. 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.

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 8-8. The parsing code looks for *double quotes, commas, spaces* and/or *tabs* as delimiters.

**Table 8-8: Variable types**

Symbol	Explanation
dv	Design variable
nv	Noise variable
rs	Response
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 has a preference for `.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". 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.

#### 8.5.4. 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. This option 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. For this example, this can be done by using the *Run with clean start from current iteration* run option, and setting the current iteration to 2. The restart will then generate a *new* starting point for iteration 2 and conduct 10 simulations altogether.

## 8.6. 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.

Sampling constraints are defined in the **Constraints** tab of the **Sampling** dialog, Figure 8-14. Previously defined constraints are available for selection in the *Add new* list, new constraints may be defined using the Sampling constraint wizard, Figure 8-15, 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. A typical explicit constraint could be a simple inequality relationship between the design variables.

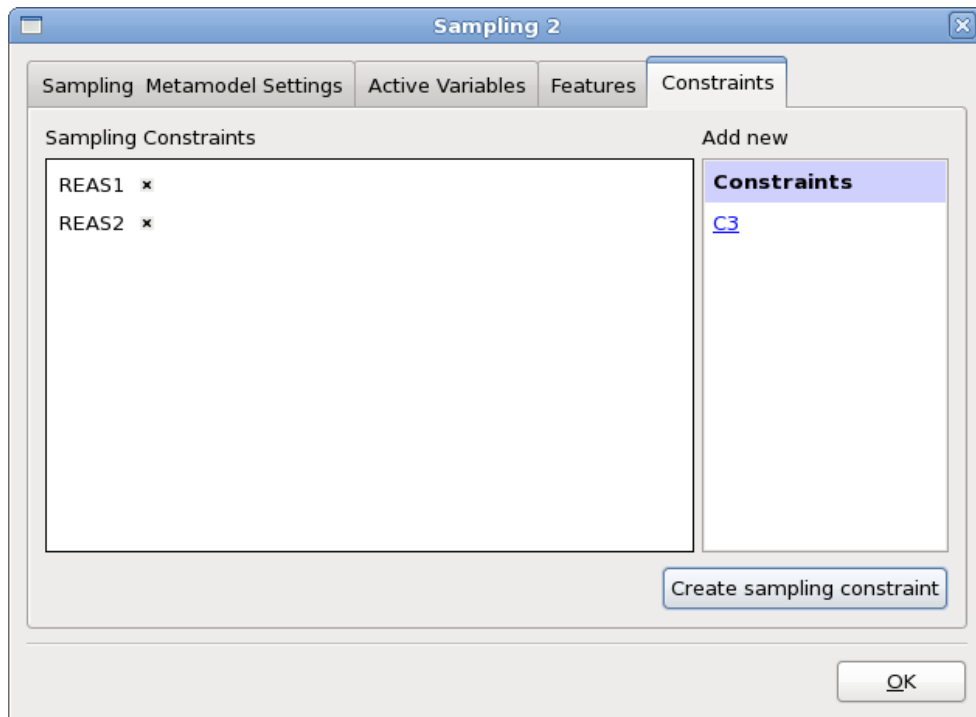


Figure 8-14: Definition of Sampling Constraints by selection from list or new creation.

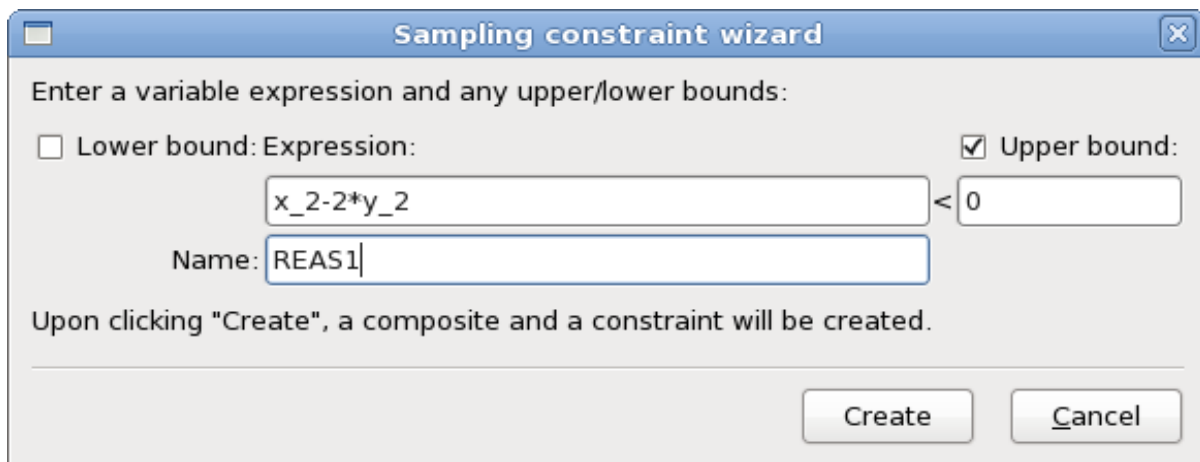


Figure 8-15: Sampling constrain wizard: definition of an expression and bounds



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 commonly used options that accompany the choice of polynomials, Radial Basis Function Networks, Neural Networks or Kriging as metamodels.

# 9. 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.

## 9.1. Introduction

### 9.1.1. 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*

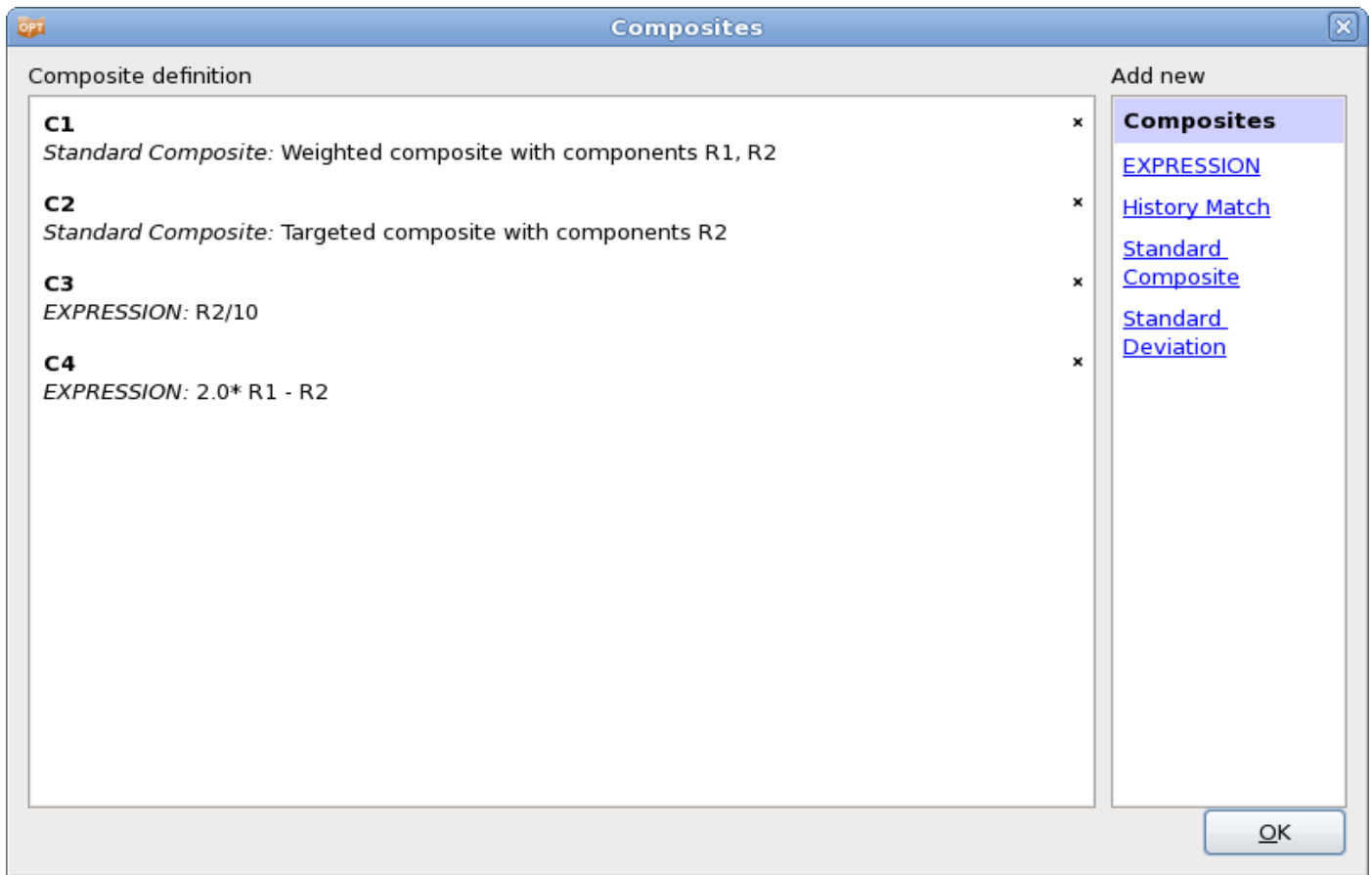
If a response function is defined as  $f(x, y) = xy$  and linear response surfaces are used, the response expression will be a simple linear approximation  $ax + by$  whereas a composite expression specified as  $xy$  will be exact.

## 9.2. Defining Composites

A composite can be defined by using the interfaces in the **Composites** dialog, Figure 9-1. To add a new definition, select the respective interface from the list on the right. The available interfaces are explained in Table 9-1. 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 (see Section 10.2).



*Figure 9-1: Composites Dialog*

**Table 9-1: Composite types**

Composite type	Description	Reference
EXPRESSION	Mathematical expression using previously defined entities	Section 9.3
Curve Matching	Curve matching metrics	Section 9.5
Standard Composite	Weighted or targeted composites	Section 9.4
Standard Deviation	Standard deviation of another response or composite	Section 9.6

### 9.3. Expression composite

**Figure 9-2: Definition of a Composite Expression**

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).

### 9.4. Standard composite

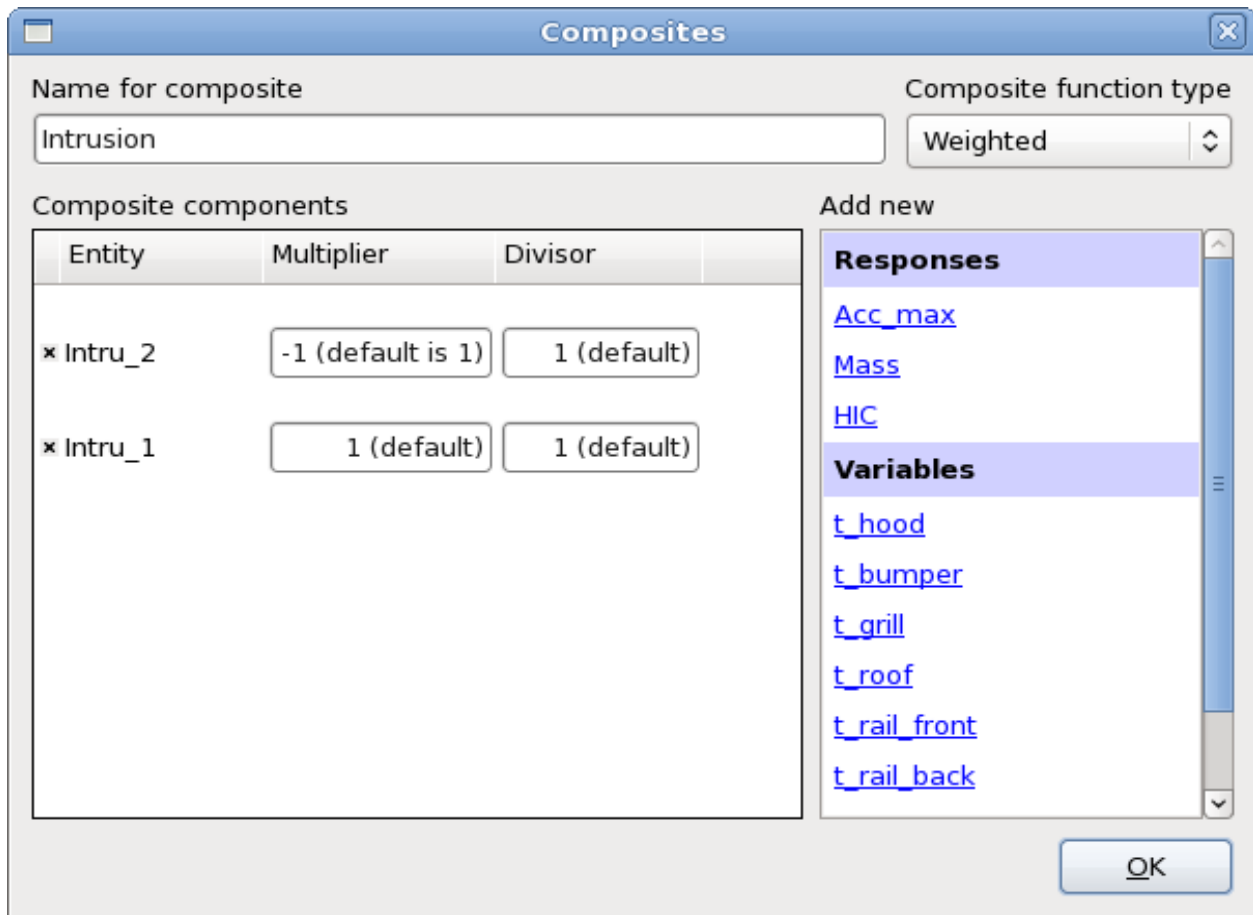
The Standard composite dialog is displayed in Figure 9-3. First the composite function type has to be selected, Table 9-2. Then select the Response or Variable components to be used to calculate the composite from the list on the right. The selected components appear in the list on the left with text 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 9-2.

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.

**Table 9-2: Standard Composite function types**

Composite function type	Reference
Weighted	Section 9.4.3
MSE	Section 9.4.2
Sqrt MSE	Section 9.4.1

**Figure 9-3: Standard Composite Interface**

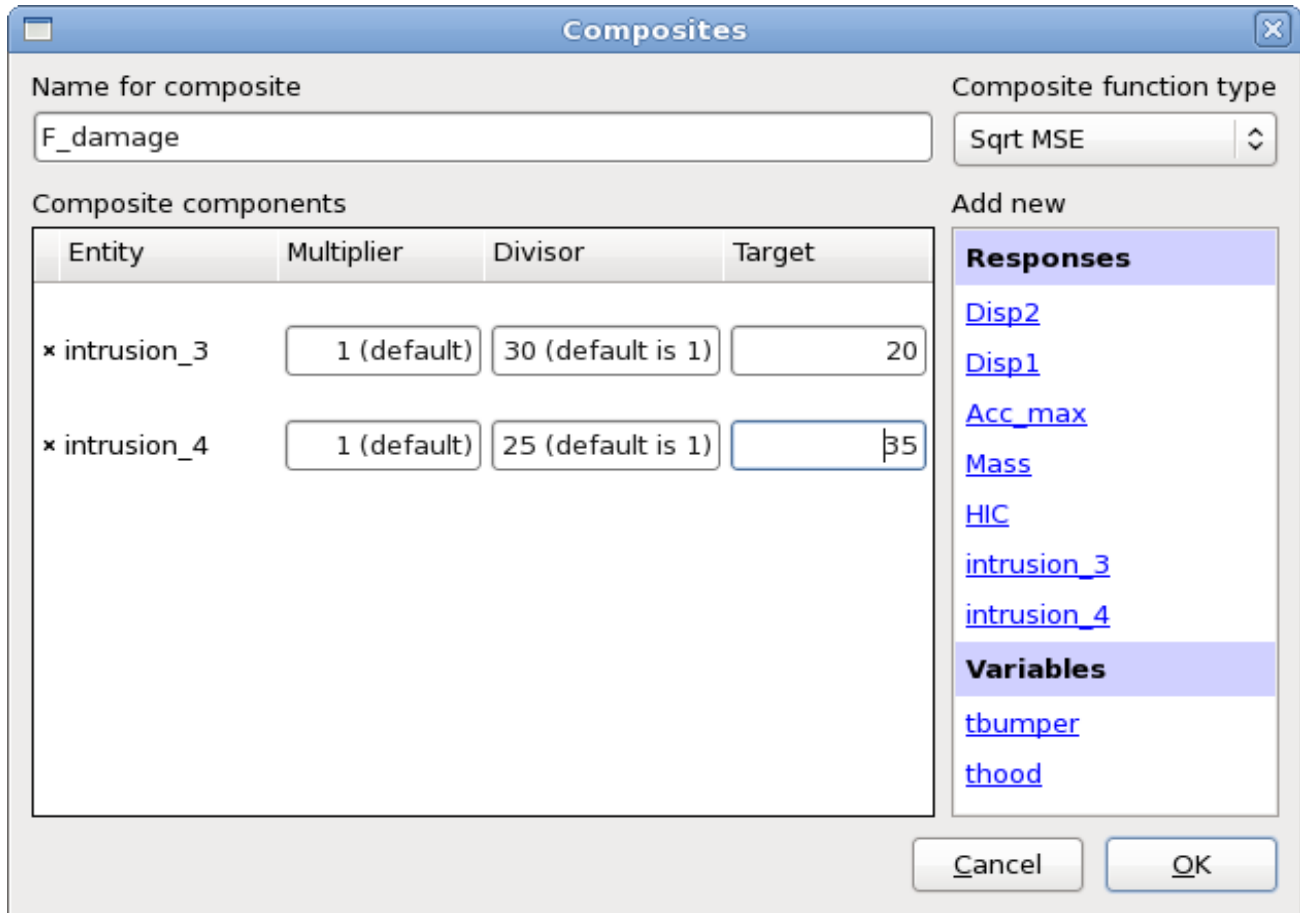
### 9.4.1. Targeted composite (square root of MSE)

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.

$$F = \sqrt{\sum_{j=1}^m W_j \left[ \frac{f_j(\mathbf{x}) - F_j}{\sigma_j} \right]^2 + \sum_{i=1}^n \omega_i \left[ \frac{x_i - X_i}{\chi_i} \right]^2}, \quad (9-1)$$

where  $\sigma$  and  $\chi$  are scale factors (to be specified in the *Divisor* text fields) and  $W$  and  $\omega$  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.



**Figure 9-4: 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  $\sigma_j$  and  $\chi_i$  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 9-4 for an example of a targeted composite response definition. Here,  $F_{\text{damage}}$  will be calculated as

$$F_{\text{damage}} = \sqrt{\left[ \frac{\text{intrusion}_3 - 20}{30} \right]^2 + \left[ \frac{\text{intrusion}_4 - 35}{25} \right]^2}.$$

The equivalent the expression composite is:

```
sqrt(((intrusion_3 - 20)/30)**2 + ((intrusion_4 + 35)/25)**2)}
```

### 9.4.2. 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 9.5.1).

### 9.4.3. 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).

$$F = \sqrt{\sum_{j=1}^m W_j \frac{f_j(\mathbf{x})}{\sigma_j} + \sum_{i=1}^n \omega_i \frac{x_i}{\chi_i}}. \quad (9-2)$$

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 9-3.

The equivalent expression composite is

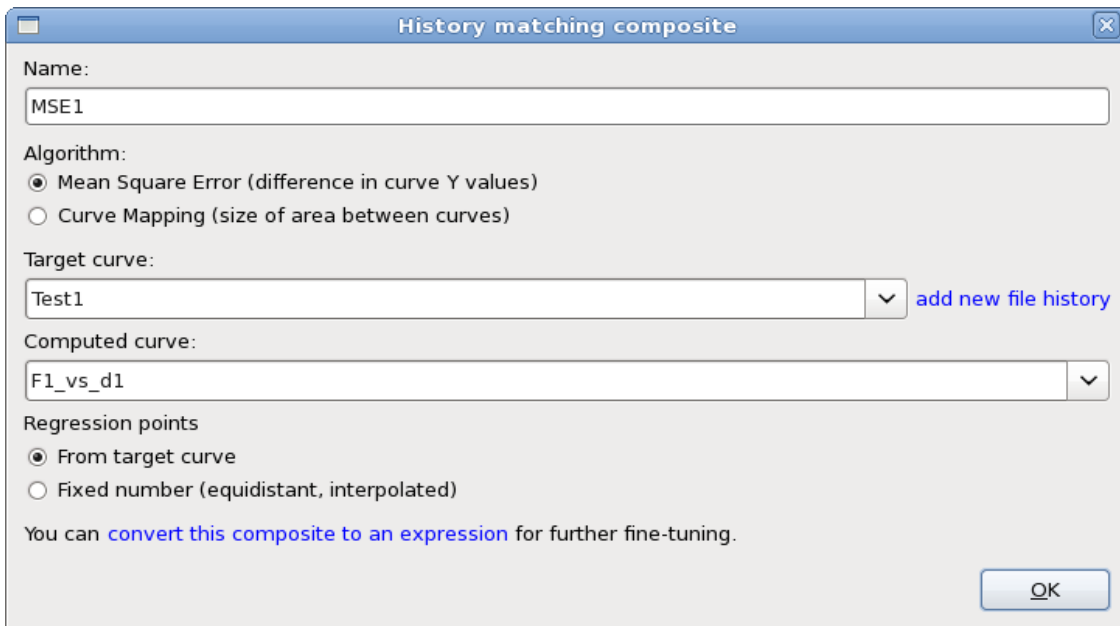
```
Intru_1 - Intru_2.
```

Needless to say, this is the preferable way to define this composite.

## 9.5. Curve Matching Composite

The Curve Matching interface provides two metrics for comparison of a target curve and curves extracted from simulation runs, Figure 9-5. The options are explained in Table 9-3.

To evaluate these composites, predicted histories (histories approximated by metamodels) are used, see Section 8.5.1 for details.



**Figure 9-5: History Matching Composite Dialog**



**Table 9-3: History Match Composite options**

Option	Description	Reference
Algorithm	Curve matching metric to calculate “distance” between target and computed curve: <ul style="list-style-type: none"> <li>○ Mean Square Error (Ordinate-based)</li> <li>○ Curve Mapping</li> </ul>	Section 9.5.1 Section 9.5.2
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:

### 9.5.1. Ordinate-based Curve Matching

A composite function is provided to compute the Mean Squared Error  $\varepsilon$  for the discrepancy between two curves:

$$\varepsilon = \frac{1}{P} \sum_{p=1}^P W_p \left[ \frac{f_p(\mathbf{x}) - G_p}{s_p} \right]^2 = \frac{1}{P} \sum_{p=1}^P W_p \left[ \frac{e_p(\mathbf{x})}{s_p} \right]^2, \quad (9-3)$$

It is constructed so that  $G_p$ ,  $p=1, \dots, P$  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.  $\mathbf{x}$  is the design vector.  $s_p = \max |G_p|$ ,  $p=1, \dots, P$ . By using the default values, the user should obtain a dimensionless error  $\varepsilon$  of the order of unity. See Section 23.3.1 for more detail.

*Note:*

1. Only points within range of both curves are included in Equation (13-3), 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  $f(\mathbf{x})$  is linear in terms of the design variables  $\mathbf{x}$ , 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 9.5.2, Section 23.3.2) is available.

### 9.5.2. Curve Mapping

In contrast to the Mean Square Error curve-matching metric described in Section 9.5.1, 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.

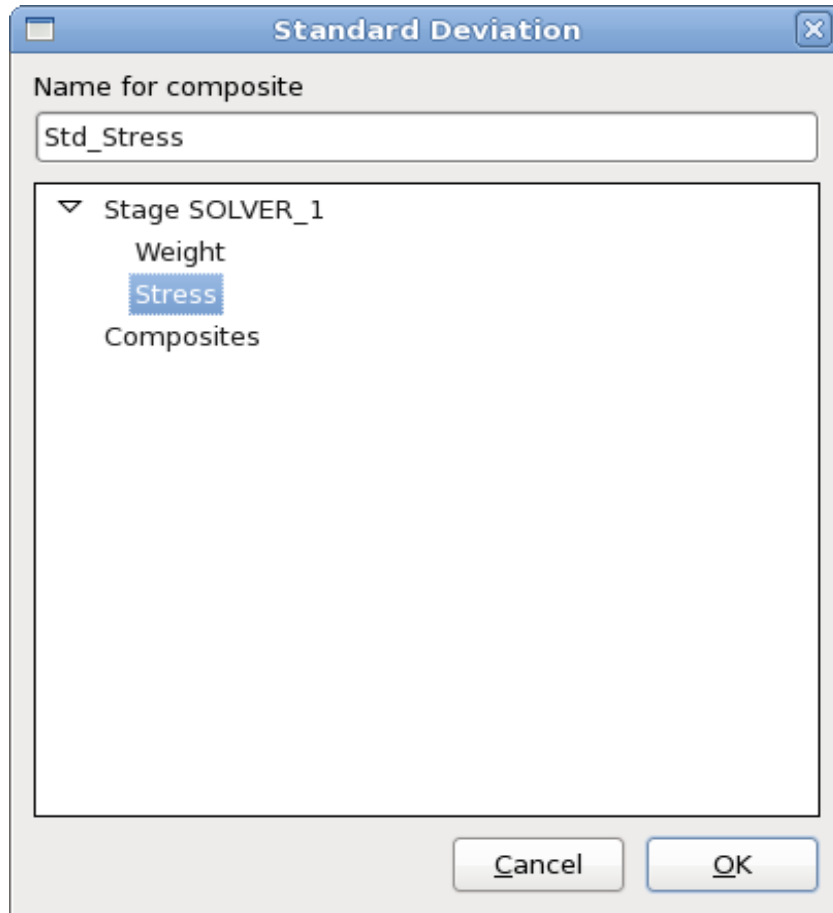
Please refer to Section 23.3.2 for the theory of Curve Mapping.

*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 (see Section 0).

## 9.6. Standard Deviation Composite

The standard deviation of another response or composite can be specified to be a composite, Figure 9-6. 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.



*Figure 9-6: 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 24.7. 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.

# 10. 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.

## 10.1. 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:

$$\begin{aligned} &\text{Minimize} && F(\Phi_1, \Phi_2, \dots, \Phi_N) \\ &\text{subject to} && \\ &&& L_1 \leq g_1 \leq U_1 \\ &&& L_2 \leq g_2 \leq U_2 \\ &&& \vdots \\ &&& L_m \leq g_m \leq U_m \end{aligned}$$

where  $F$  represents the multi-objective function,  $\Phi_i = \Phi_i(x_1, x_2, \dots, x_n)$  represent the various objective functions and  $g_j = g_j(x_1, x_2, \dots, x_n)$  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  $\Phi_i$  must be specified so that the multi-objective

$$F = \sum_{k=1}^N \omega_k \Phi_k. \quad (10-1)$$

A component function must be assigned to each objective function where the component function can be defined as a *composite function*  $F$  (see Chapter 9) or a *response function*  $f$  (see Chapter 6).

## 10.2. Defining objective functions

Objectives are defined in the **Objectives** tab of the **Optimization** dialog, Figure 10-1. 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 10.1. The weight applies to each objective as represented by  $\omega_k$  in Equation (11.1). 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 10-1.

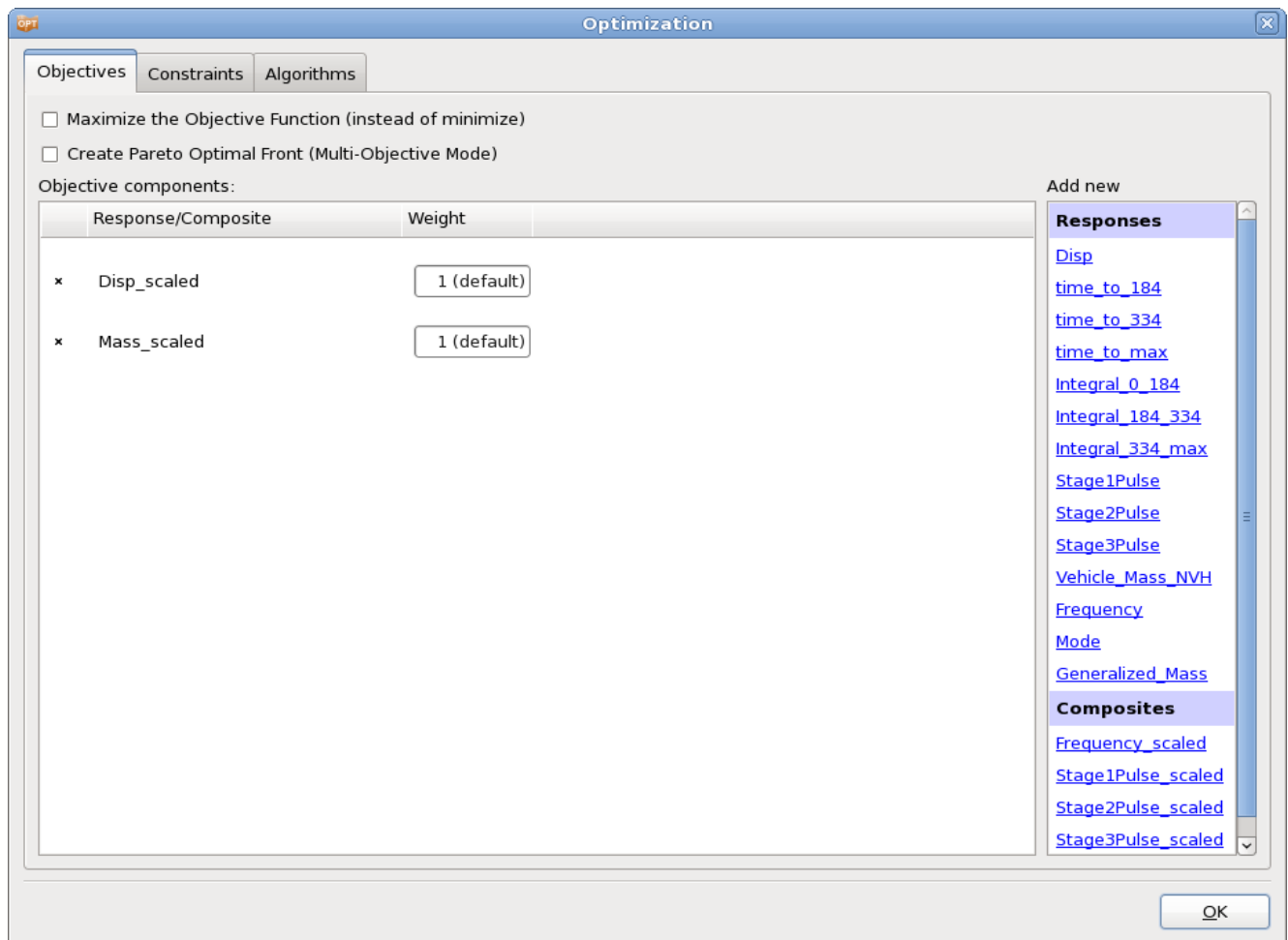


Figure 10-1: Objective panel in LS-OPTui Optimization dialog.

**Table 10-1: 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.

### 10.3. Defining a constraint

Constraints are defined in the **Constraints** tab of the **Optimization** dialog, Figure 10-2. To define a constraint, select a response or composite from the list on the right, that contains all previously defined responses and composites. 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.

Additionally, for Reliability Based Design Optimization, the probability of exceeding a bound on a constraint can be set.

Internal constraint scaling can be defined by selecting the *Constraint scaling* option and defining the respective scaling factors in the Divisor text field, Section 10.3.1.

For details on the *Strict* option, see Section 10.3.2.

To delete a constraint definition or a bound, use the respective *delete* icon.

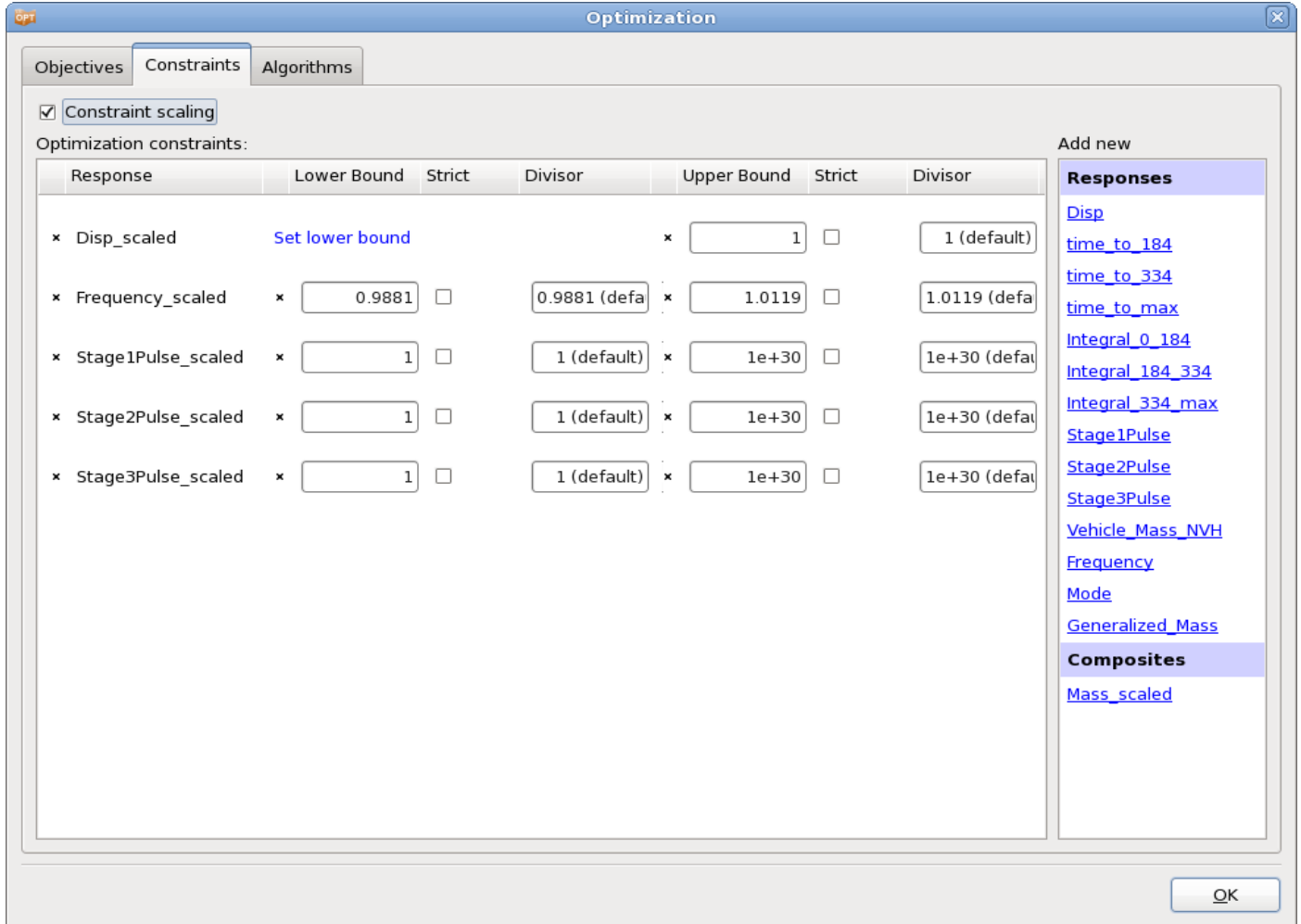


Figure 10-2: Constraints panel in LS-OPTui

### 10.3.1. 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:

$$\frac{-g_j(x) + L_j}{s_j^L} \leq 0; \frac{g_j(x) - U_j}{s_j^U} \leq 0$$

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

$$\frac{-g_j(x)}{L_j} + 1 \leq 0; \frac{g_j(x)}{U_j} - 1 \leq 0$$

internally and in the infeasible phase:

$$\frac{-g_j(x)}{L_j} + 1 \leq e; \frac{g_j(x)}{U_j} - 1 \leq e; e \geq 0.$$

### 10.3.2. Minimizing the maximum response or violation\*

Refer to Section 23.1 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 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.

## 10.4. Algorithms

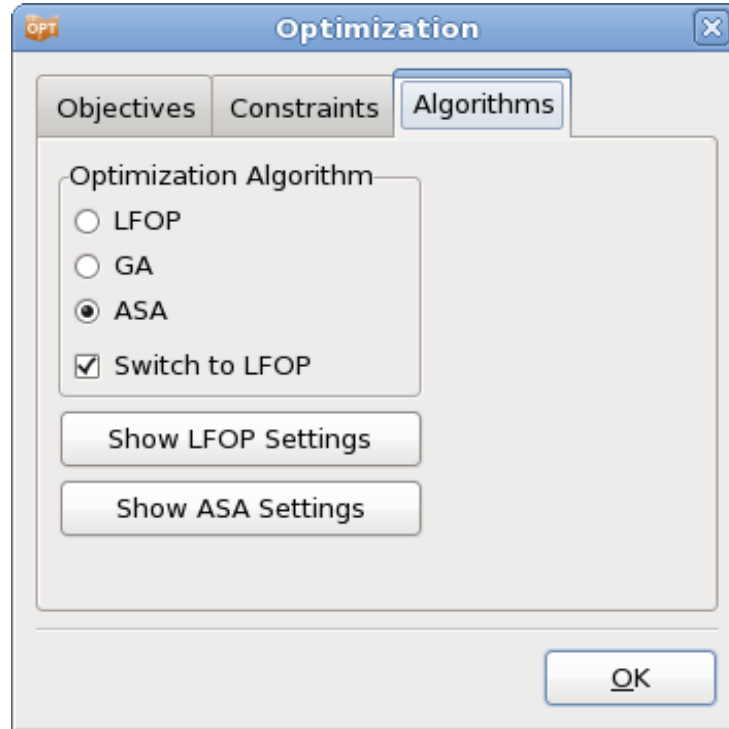
Optimization algorithms for metamodel-based optimization can be selected in the Algorithms tab of the Optimization dialog, Figure 10-3.

The core solvers that can be used for metamodel optimization are LFOP, the Genetic Algorithm (GA) and Adaptive Simulated Annealing (ASA). Hybrid algorithms may also be selected by selecting *Switch to LFOP*, namely the Hybrid GA and Hybrid SA. The hybrid algorithms start with the GA and SA 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 SA).

Hybrid Simulated Annealing is the default.

For each algorithm, advanced settings are available using the respective *Show \*\*\* Settings* button.





*Figure 10-3: Selecting the optimization algorithm used for the optimization on the metamodel*

*Table 10-2: Algorithms options*

Option	Description	Reference
LFOP	Leapfrog Optimizer	Section 10.4.1, Section 22.7
GA	Genetic Algorithm	Section 10.4.2, Section 22.8
ASA	Adaptive Simulated Annealing	Section 10.4.3, Section 22.10
Switch to LFOP	Hybrid version	Section 22.11

### 10.4.1. 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 22.7 for the theory of LFOPC.



*Figure 10-4: LFOF settings*

**Table 10-3: LFOPC parameters and default values**

Option	Parameter	Remark
Number of Multi-Start Points	Number of Multi-Start Points	
Penalty Parameter mu	Initial penalty value $\mu$	
Penalty Parameter mumax	Maximum penalty value $\mu_{\max}$	1
Convergence Criterion xtol	Convergence tolerance $\epsilon_x$ on the step movement	2
Convergence Criteria eg	Convergence tolerance $\epsilon_f$ on the norm of the gradient	2
Maximum Step Size	Maximum step size $\delta$	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

$$\|\Delta(\mathbf{x})\| < \epsilon_x$$

or

$$\|\nabla f(\mathbf{x})\| < \epsilon_f$$

3. It is recommended that the maximum step size,  $\delta$ , be of the same order of magnitude as the “diameter of the region of interest”. To enable a small step size for the successive approximation scheme, the maximum step size has been defaulted to  $\delta = 0.05\sqrt{\sum_{i=1}^n (range)^2}$ .
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.

### 10.4.2. 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 10-5). See Section 22.8 for the theory of the Genetic Algorithm.

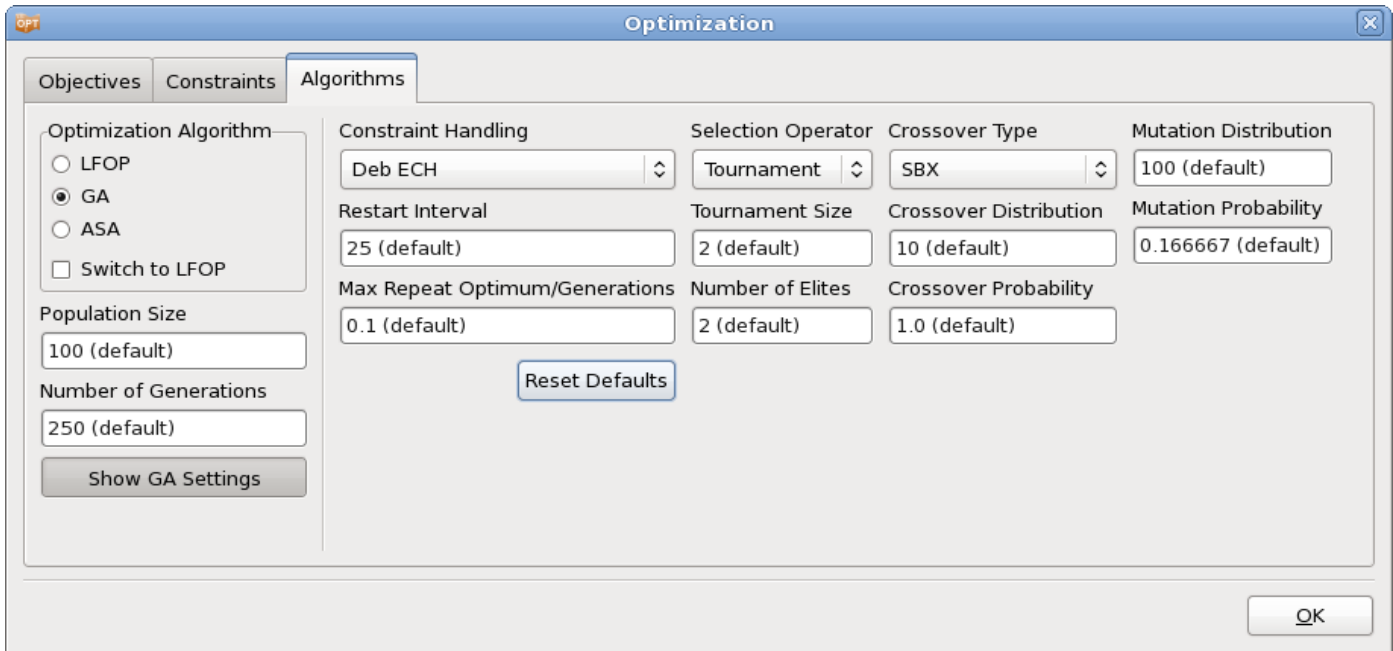


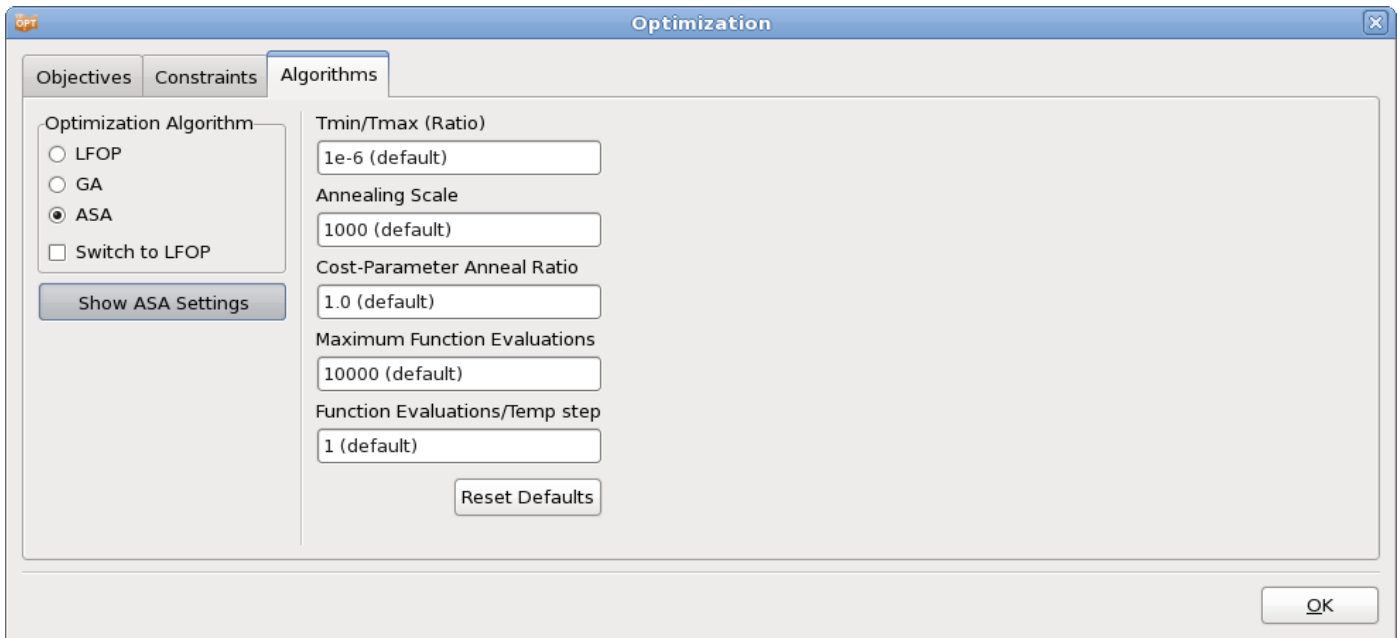
Figure 10-5: GA settings

**Table 10-4: 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 $\alpha$ 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	

### 10.4.3. Setting parameters in the simulated annealing algorithm\*

The adaptive simulated annealing parameters can be modified in the GUI, Figure 10-6. See Section 22.10 for the theory of Adaptive Simulated Annealing.

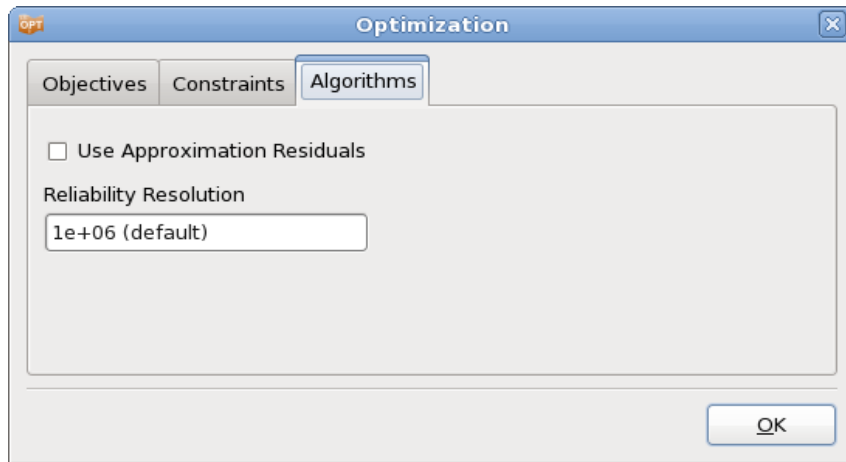


*Figure 10-6: ASA settings*

**Table 10-5: 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
Runction Evaluations/Temp step	Number of function evaluations at some temperature

## 10.5. Algorithms for metamodel based Monte Carlo analysis

**Figure 10-7: Algorithm Options for Metamodel based Monte Carlo Analysis**

**Table 10-6: 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.

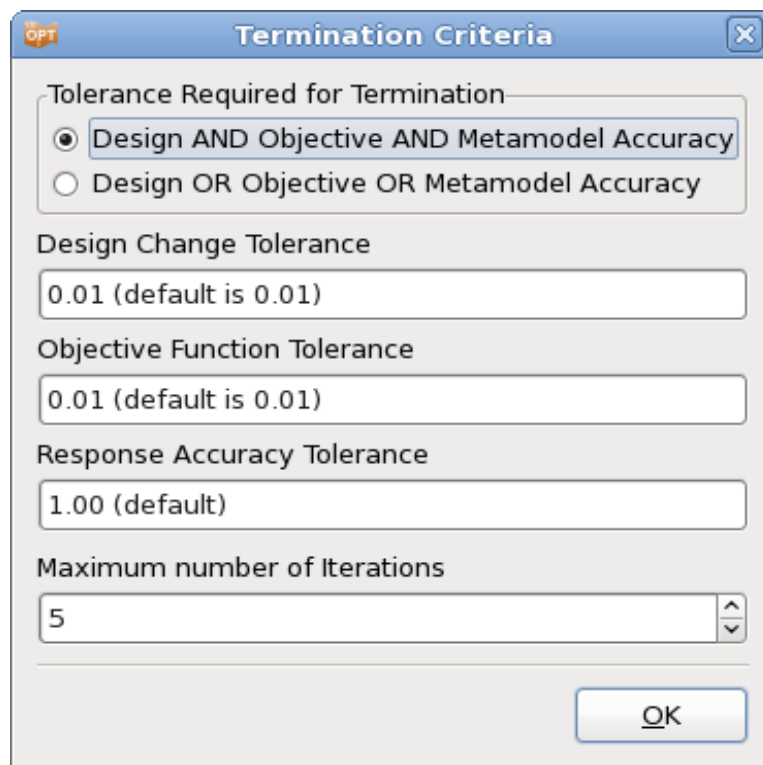


# 11. Termination Criteria

This chapter explains termination criteria for iterative tasks.

## 11.1. Metamodel based methods

The user can specify tolerances on the design change ( $\Delta x_i$ ), the objective function change ( $\Delta f$ ) and 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 11-1. The options are described in Table 11-1.



*Figure 11-1: Termination Criteria dialog for metamodel based optimization, strategy sequential or sequential with domain reduction*

**Table 11-1: Termination Criteria options for metamodel based optimization**

Option	Parameter	Reference
Tolerance Required for Termination	Design AND Objective AND Metamodel Accuracy Design OR Objective OR Metamodel Accuracy	-
Design Change Tolerance	Tolerance on design accuracy $\varepsilon_x$	Section 11.1.1
Objective Function Tolerance	Tolerance on objective function accuracy $\varepsilon_f$	Section 11.1.1
Response Accuracy Tolerance	Tolerance on accuracy of response surface $\varepsilon_r$	Section 11.1.2
Maximum number of Iterations	Maximum Number of Iterations	Section 11.1.3

### 11.1.1. Design Change Tolerance and Objective Function Tolerance

The design change termination criterion becomes active if

$$\frac{\|x^{(k)} - x^{(k-1)}\|}{\|d\|} < \varepsilon_x,$$

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. If the AND option is used, make sure that the response accuracy tolerance value is set to a large value that is always active, e.g. the default of 1. If the OR option is used, set the response accuracy tolerance to a value that will never be active, e.g. 0.

### 11.1.2. Response Accuracy Tolerance

The tolerance on the metamodel accuracy is based on the *change* of the prediction accuracy measure (square root of the PRESS error). 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 criterion becomes active if

$$\left| s_i^{(k)} - s_i^{(k-1)} \right| < \varepsilon_r,$$

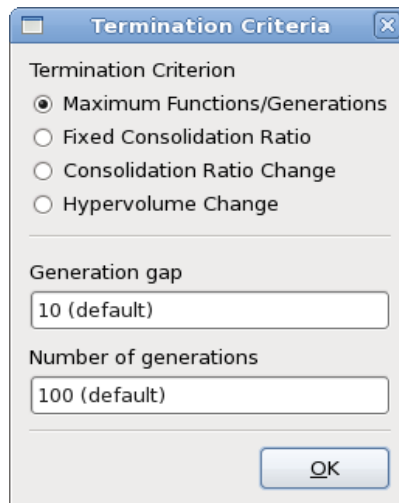
where  $s_i$  denotes the approximation error of  $i^{\text{th}}$  response characterized by the ratio of square root PRESS statistics and the mean value of response and,  $(k)$  and  $(k - 1)$  refer to two successive iteration numbers.

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.

### 11.1.3. 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.

## 11.2. Direct Optimization



*Figure 11-2: Termination Criteria dialog for direct optimization*

Termination criteria are available for multi-objective optimizers. While the default selection is maximum number of function evaluations/generations, one can also use consolidation ratio or hypervolume based metrics to terminate the search as shown in Figure 11-2. The available options are described in Table 11-2.

**Table 11-2: Termination criteria and default values**

Item	Parameter
Termination Criterion	MOO performance metric <sup>*</sup> : Consolidation Ratio   Variable Consolidation Ratio   Hypervolume <sup>*</sup> 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
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.

# 12. 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. The simulation statistical tools (histogram plot, probability bounds), scatter, parallel coordinate and correlation plots 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, and correlation plots.

More background on the probabilistic methods is given in Chapter 24 (the theoretical manual), while example problems can be found in Chapter 19. The LS-DYNA results can be investigated for possible bifurcations using DYNASTats described in Chapter 15.

## 12.1. 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
  - Assigning the statistical distributions to design variables, Section 12.2
2. Definition of the probabilistic task: The available task options are Direct Monte Carlo Analysis, Metamodel-based Monte Carlo Analysis and RBDO/Robust Parameter Design.
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.

## 12.2. 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 12-1. 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 12-1. 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.

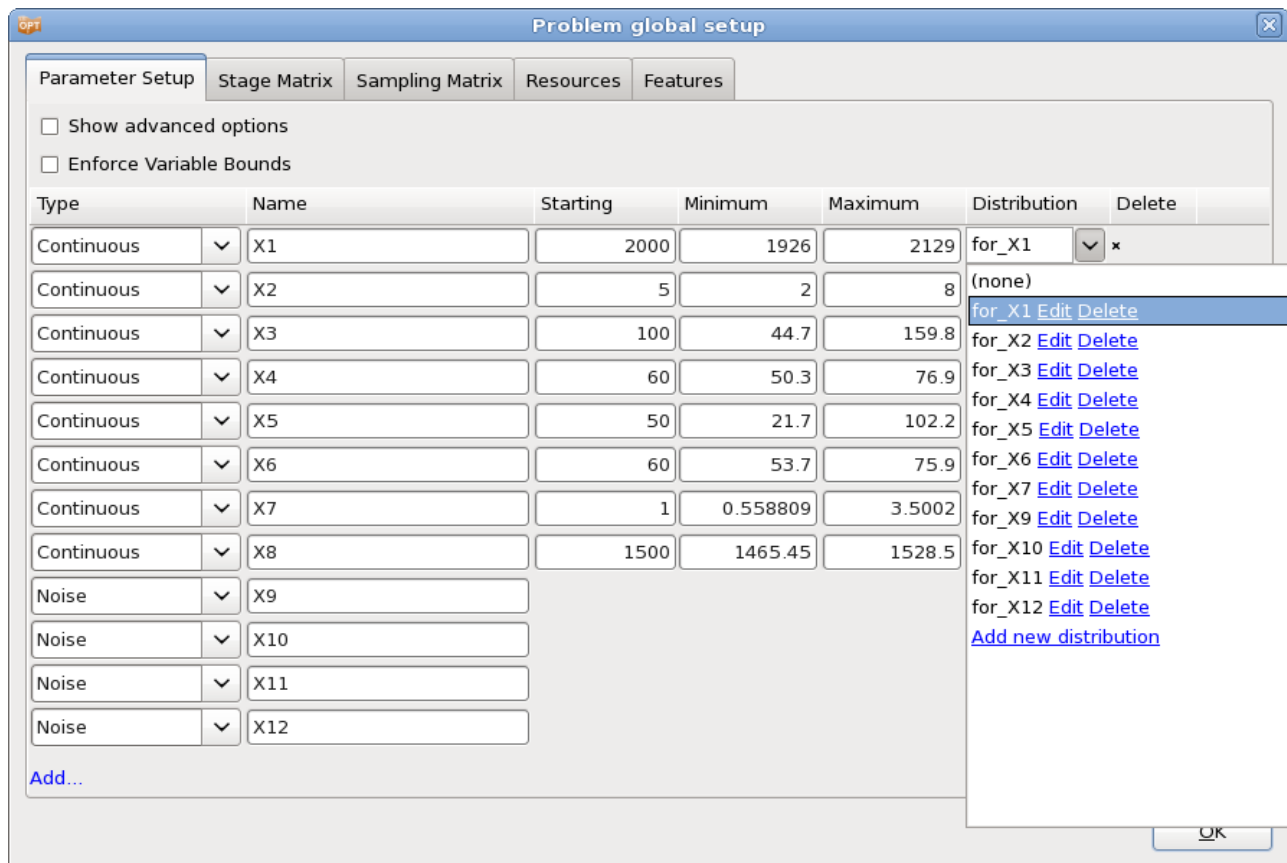


Figure 12-1: Setup Dialog, Parameter Setup: Definition of Probabilistic Distributions

## 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 12-1. Swapping the values of  $q$  and  $r$  produces a mirror image of the distribution.

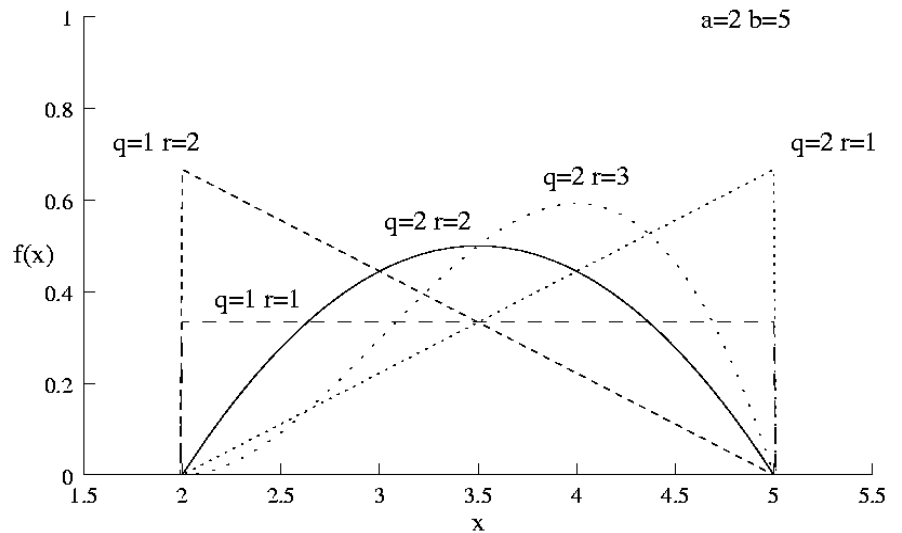
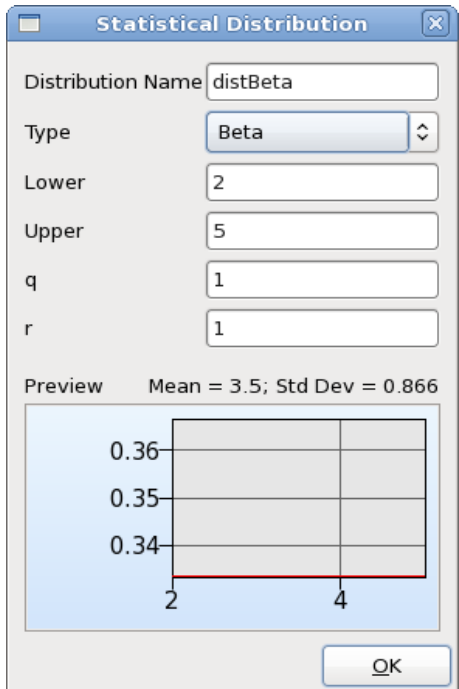


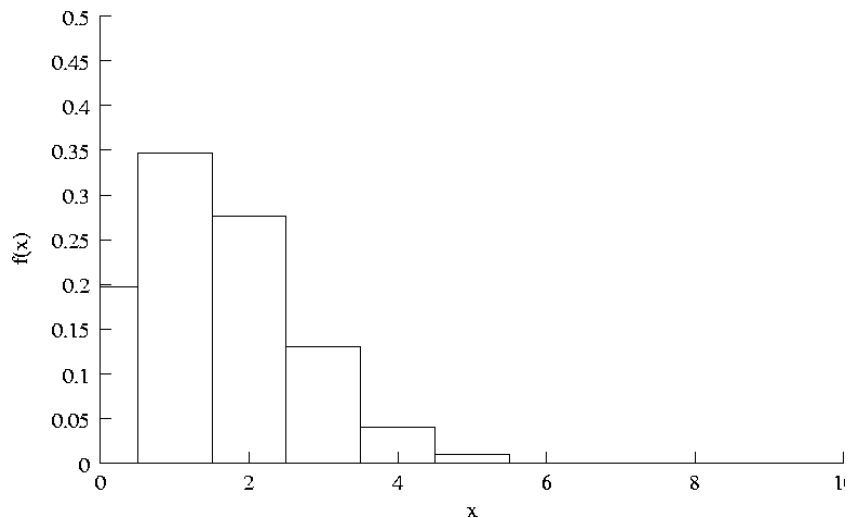
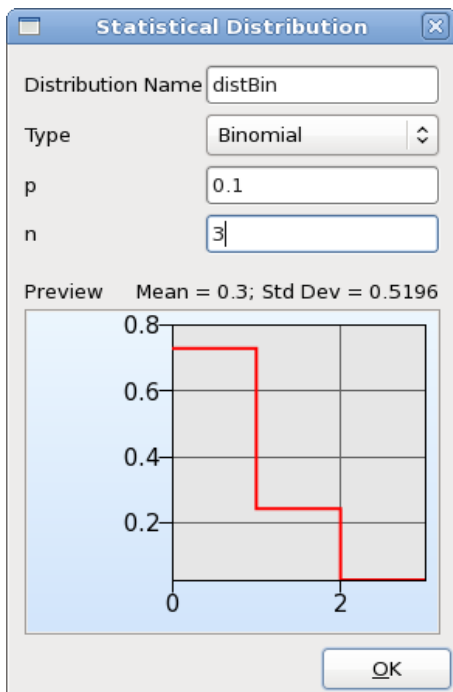
Figure 12-2: Beta distribution

**Table 12-1: Parameters defining a Beta distribution**

Item	Description
Lower	Lower Bound
Upper	Upper Bound
q	Shape parameter q
r	Shape parameter r

## 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 12-2. For  $n=1$ , it is the Bernoulli distribution (experiments with two possible outcomes — success or failure) with probability of success  $p$ .

**Figure 12-3: Binomial distribution**

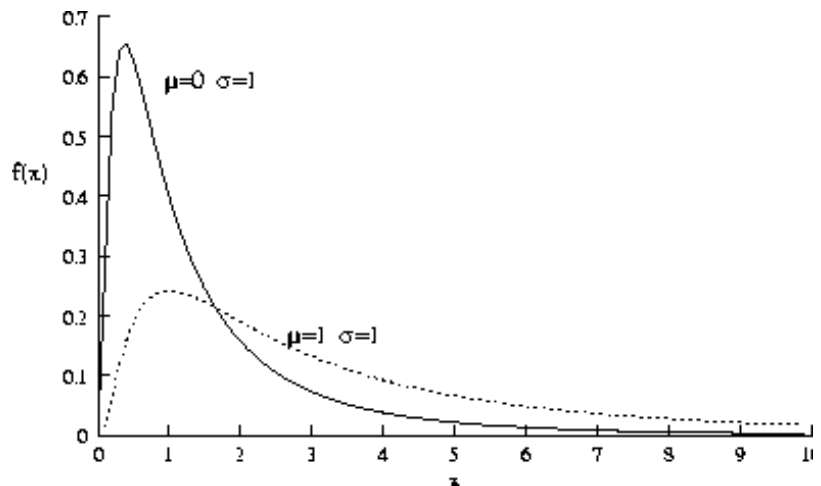
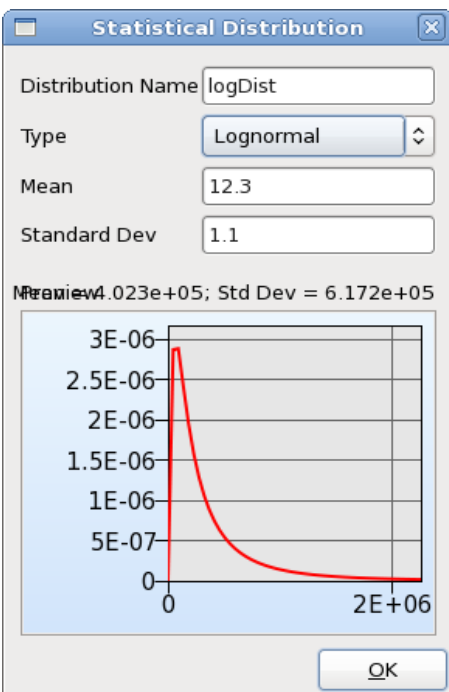


**Table 12-2: Parameters defining a Binomial distribution**

Item	Description
p	Probability of event (Success)
n	Number of trials

## Lognormal distribution

If  $X$  is a lognormal random variable with parameters  $\mu$  and  $\sigma$ , Table 12-3, the random variable  $Y = \ln X$  has a normal distribution with mean  $\mu$  and variance  $\sigma^2$ .

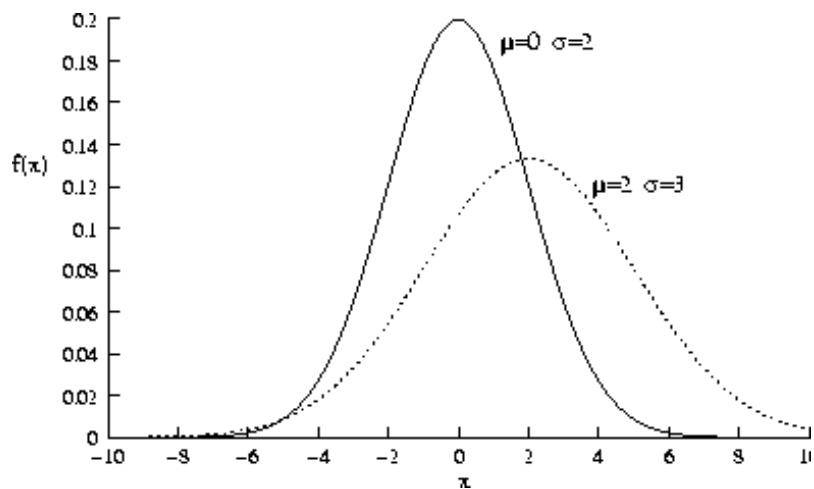
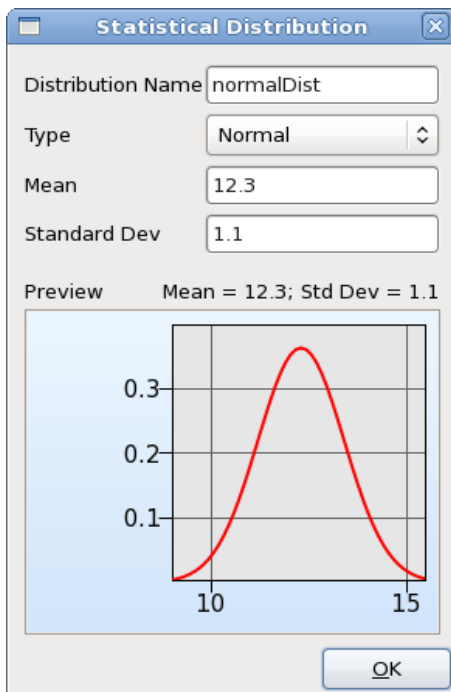
**Figure 12-4: Lognormal distribution**

**Table 12-3: Parameters defining a Lognormal distribution**

Item	Description
Mean	Mean value in logarithmic domain
Standard Dev	Standard deviation in logarithmic domain

## Normal distribution

The normal distribution is symmetric and centered about the mean  $\mu$  with a standard deviation of  $\sigma$ .

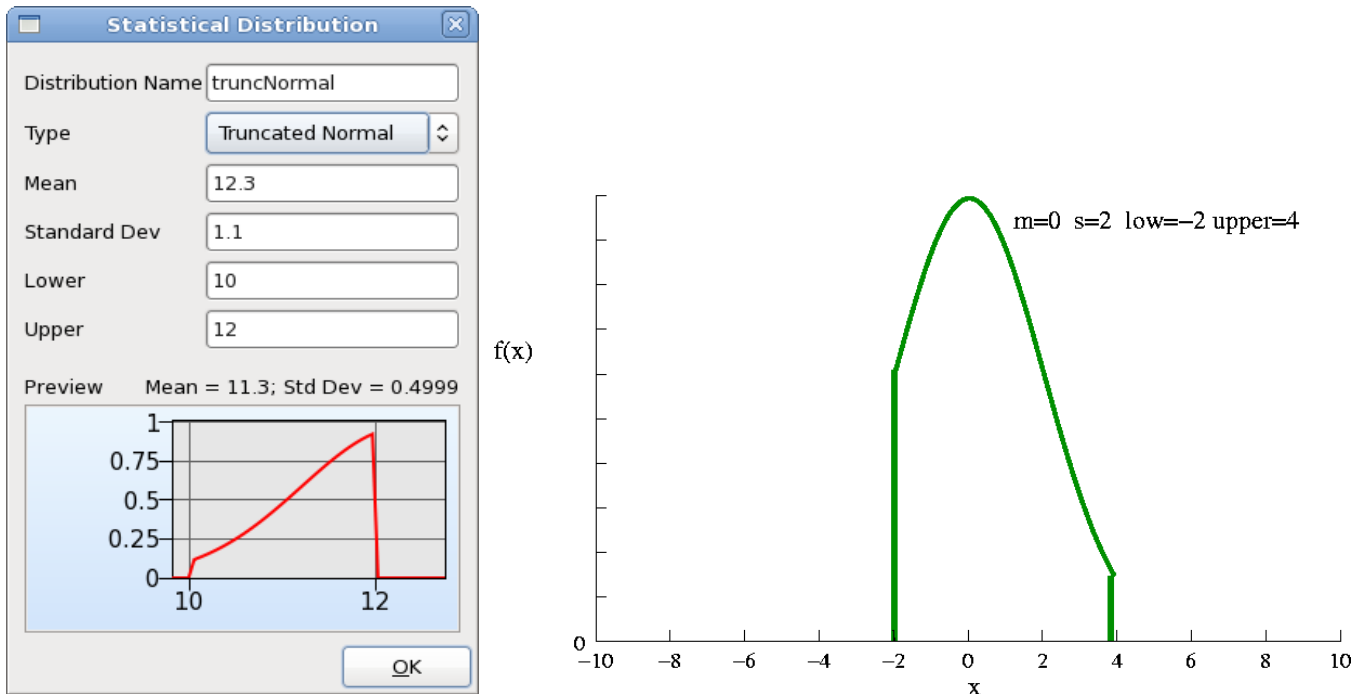
**Figure 12-5: Normal Distribution**

**Table 12-4: 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.

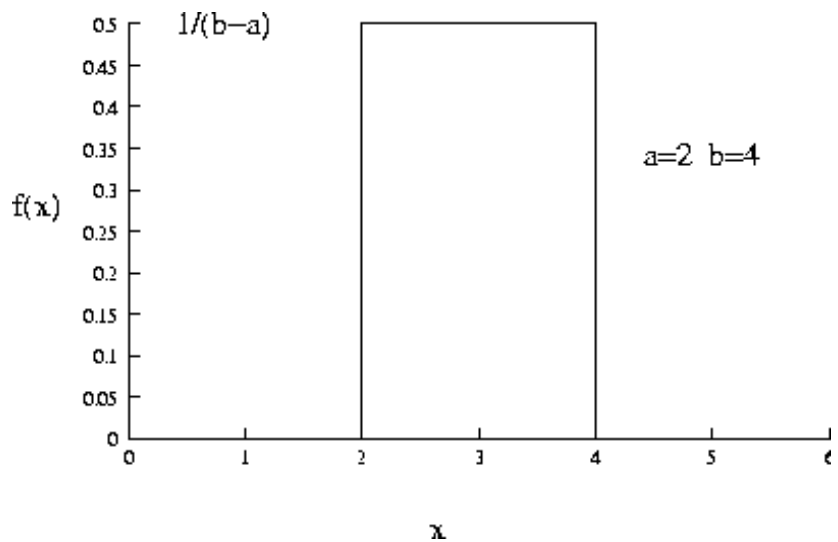
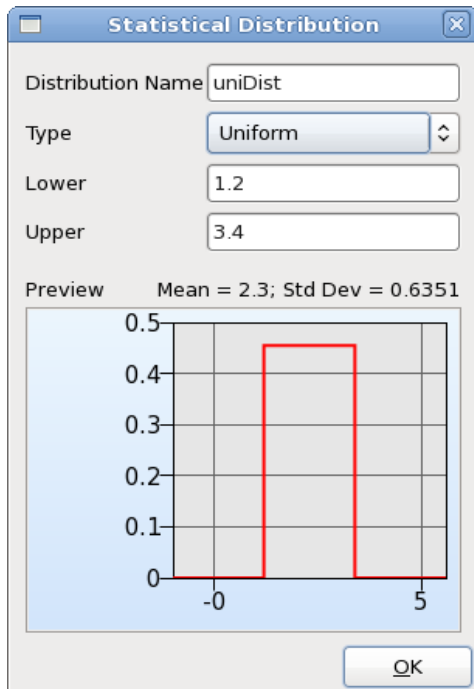
**Figure 12-6: Truncated Normal Distribution**

**Table 12-5: 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

## Uniform distribution

The uniform distribution has a constant value over a given range.

**Figure 12-7: Uniform Distribution**

**Table 12-6: Parameters defining a Uniform distribution**

Item	Description
Lower	Lower bound
Upper	Upper bound

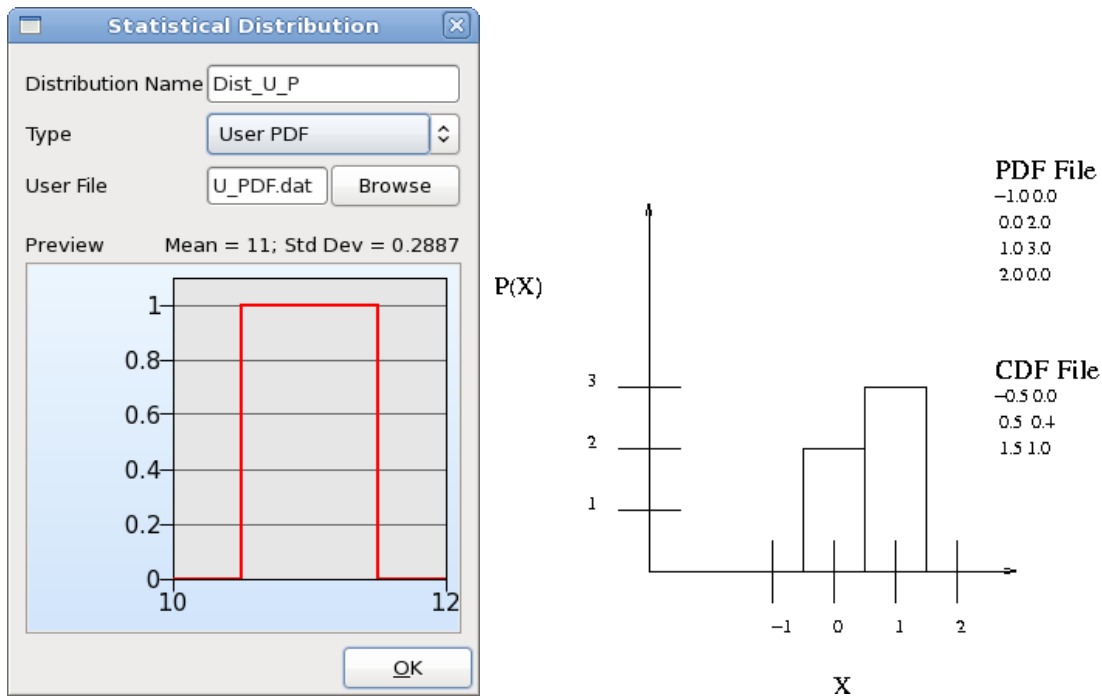
## User defined distribution

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.

**Figure 12-8: User defined distribution**

**Table 12-7: 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*

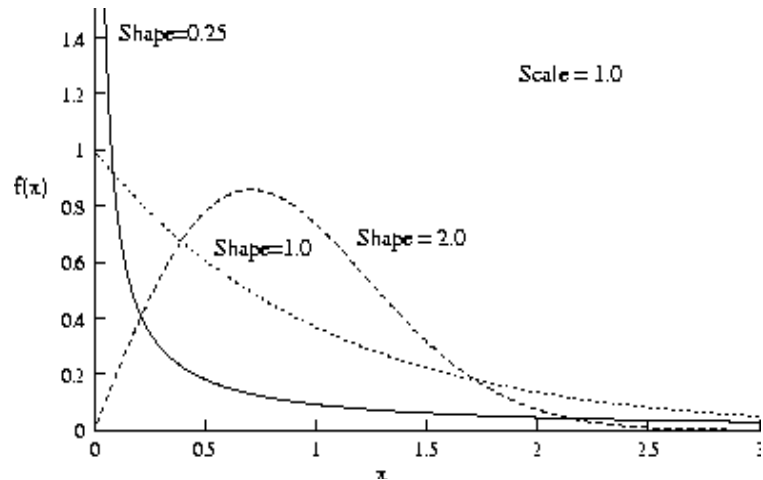
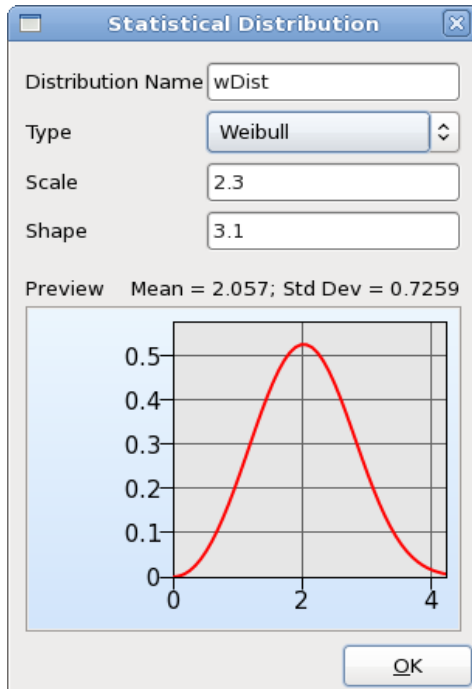
```

$ 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

```

**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.



**Figure 12-9: Weibull distribution**

**Table 12-8: Parameters defining a Weibull distribution**

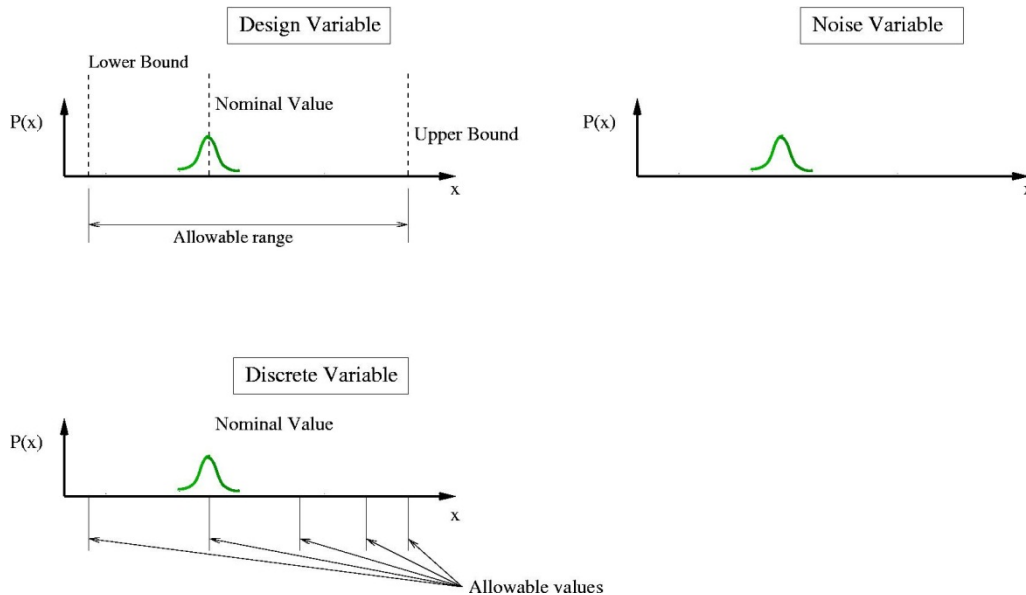
Item	Description
Scale	Scale parameter
Shape	Shape parameter

### 12.3. 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 12.2). 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 12-10) 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 12-10: 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.

### 12.3.1. 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

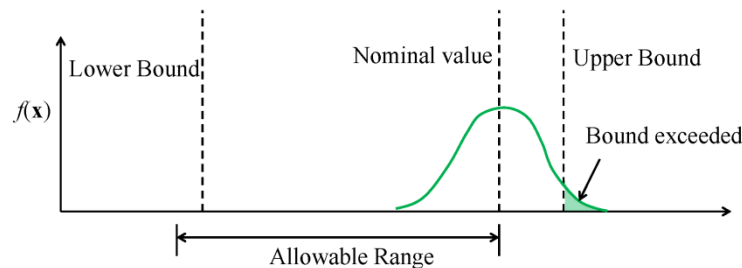


assigned a normal distribution with  $\mu=0$  and  $\sigma=2$ ; the values of the variable will be normally distributed around a nominal value of 7 with a standard deviation of 2.

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.

### 12.3.2. 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 12-11). This is the case by default, unless specified otherwise Using “Enforce Variable Bounds” in the Parameter Setup panel.



**Figure 12-11: 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 7-3). By default, two standard deviations are used on either side of the nominal value.

## 12.4. 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. The Monte Carlo, Latin Hypercube and space-filling 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.

Only variables with a statistical distribution will be perturbed; all other variables will be considered at their nominal value.

The following will be computed for all responses:

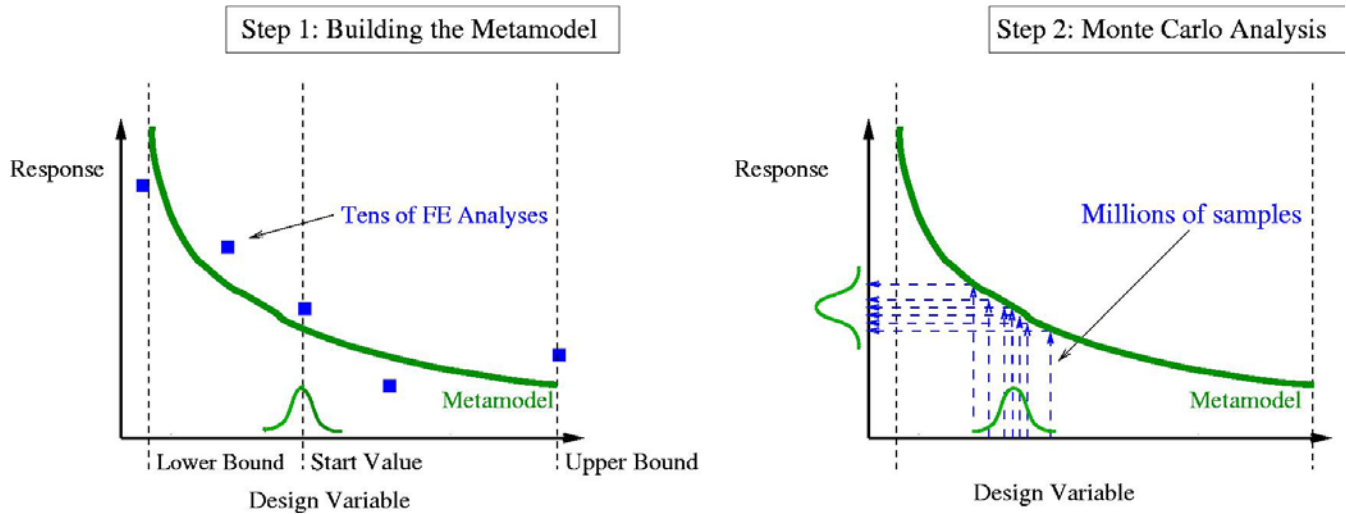
- Statistics such as the mean and standard deviation for all responses and constraints
- Reliability information regarding all constraints:
- The number of times a specific constraint was violated during the simulation
- The probability of violating the bounds and the confidence region of the probability
- A reliability analysis for each constraint assuming a normal distribution of the response.

The 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.

## **12.5. 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. 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.
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. 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.



**Figure 12-12: Metamodel-based Monte Carlo analysis.** The method proceeds 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.

Metamodel-based probabilistic analysis or optimization is accompanied by the calculation of stochastic contributions of the variables. 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 component of the GUI. 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 following data will be collected:

- Statistics such as the mean and standard deviation for all responses, constraints, 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

## 12.6. RBDO/Robust parameter design

To find a robust parameter design, use the task *RBDO/Robust parameter design*, Section 24.8 and Section 24.9, and the strategy *Sequential with Domain Reduction*, Section 4.7.3.

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 and therefore

available for use in a constraint or objective, or in another composite. The theoretical concerns are discussed in Section 24.8.

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 Method (FOSM), Section 24.4.4. No additional FE runs are therefore required for the probabilistic computations.

The method requires very little information additionally to what is required for deterministic optimization. Specify the following:

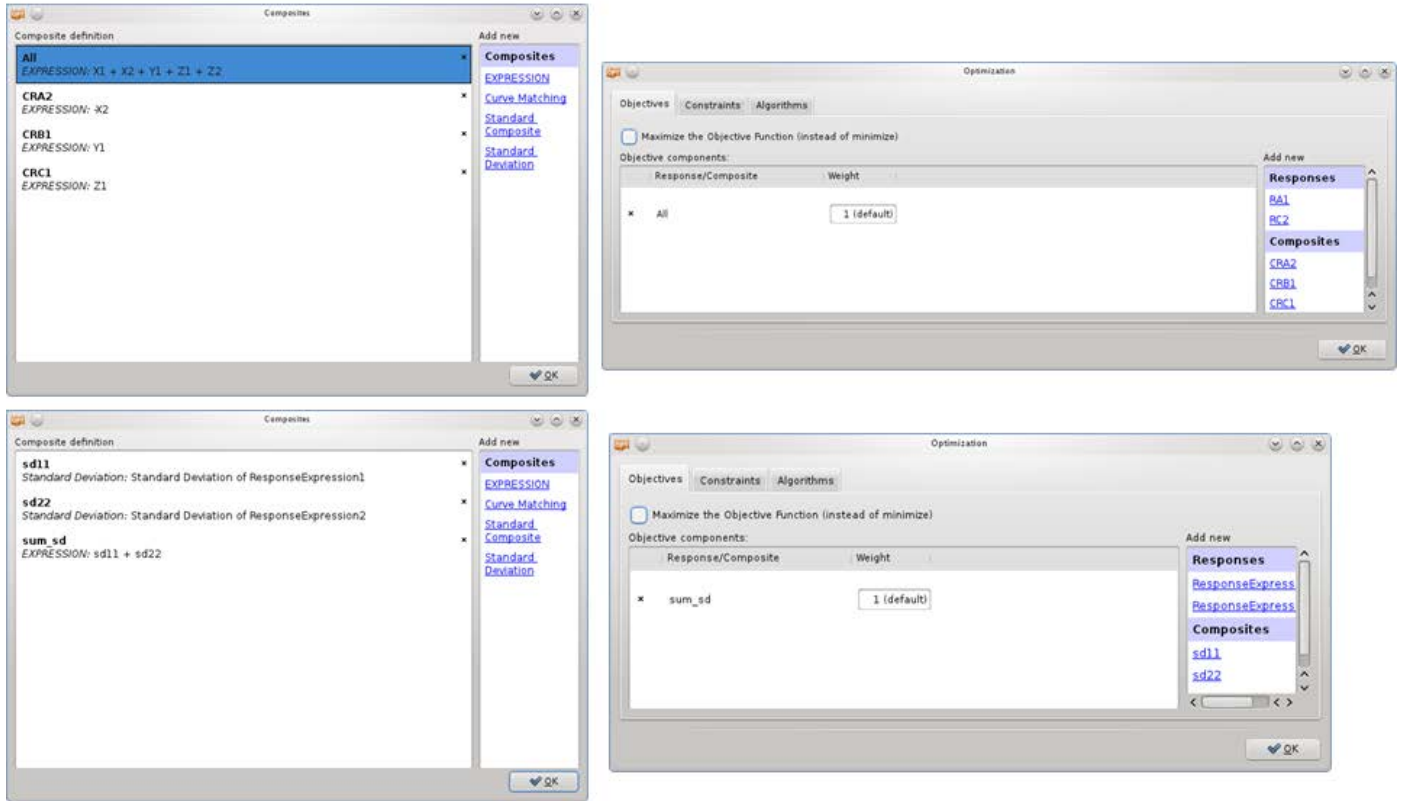
- Statistical distributions associated with the design variables
- Probabilistic bounds on the constraints.

The statistical distributions associated with the design variables are specified in the same manner as for a Monte Carlo analysis using a metamodel.

Response	Lower Bound	Strict	P[Resp<LowB]	Upper Bound	Strict	P[Resp>UppB]
× RA1	× 1		0.158 (default i)	× 1e+30		0 (default)
× RC2	Set lower bound			× -3		0.158 (default i)
× CRA2	Set lower bound			× -2		0.158 (default i)
× CRB1	× 2		0.158 (default i)	× 1e+30		0 (default)
× CRC1	× 3		0.158 (default i)	× 1e+30		0 (default)

**Figure 12-13: 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 12-14).



**Figure 12-14:** An example of objective function for RBDO (top) and robust design (bottom). Standard deviation is defined as the objective in latter case.

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  $x_j$ ), while the  $x_j^2$  and  $x_j^2$  terms can be included for a more accurate computation of the variance. An example is given in Section 19.4.

# 13. 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.

## 13.1. 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.

## 13.2. Analysis monitoring

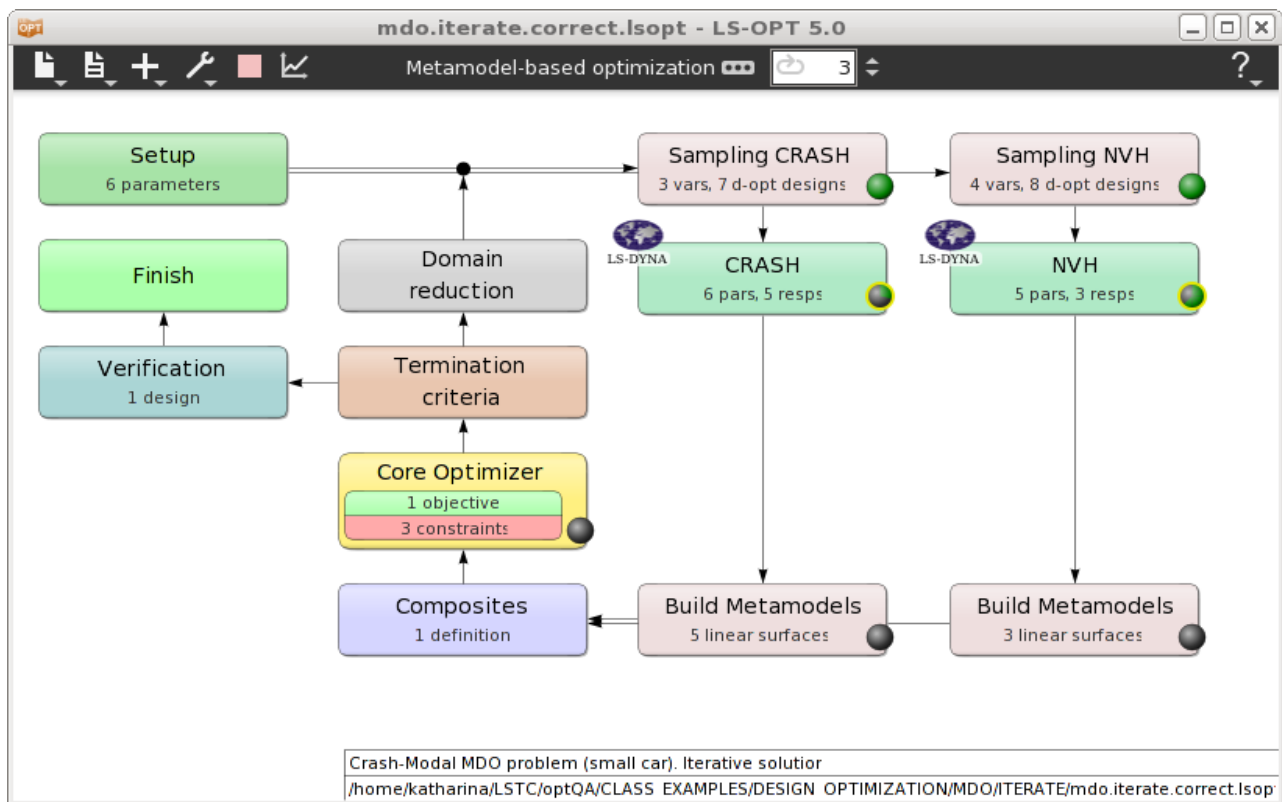


Figure 13-1: Main GUI while optimization is running

While running LS-OPT, the status and progress of the task can be visualized in the main GUI, Figure 13-1.

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 `N o r m a l` and `E r r o r` terminations, respectively. Double-clicking on a stage LED launches the **Progress** dialog described in Section 13.3.

### 13.3. Job monitoring – the Progress dialog

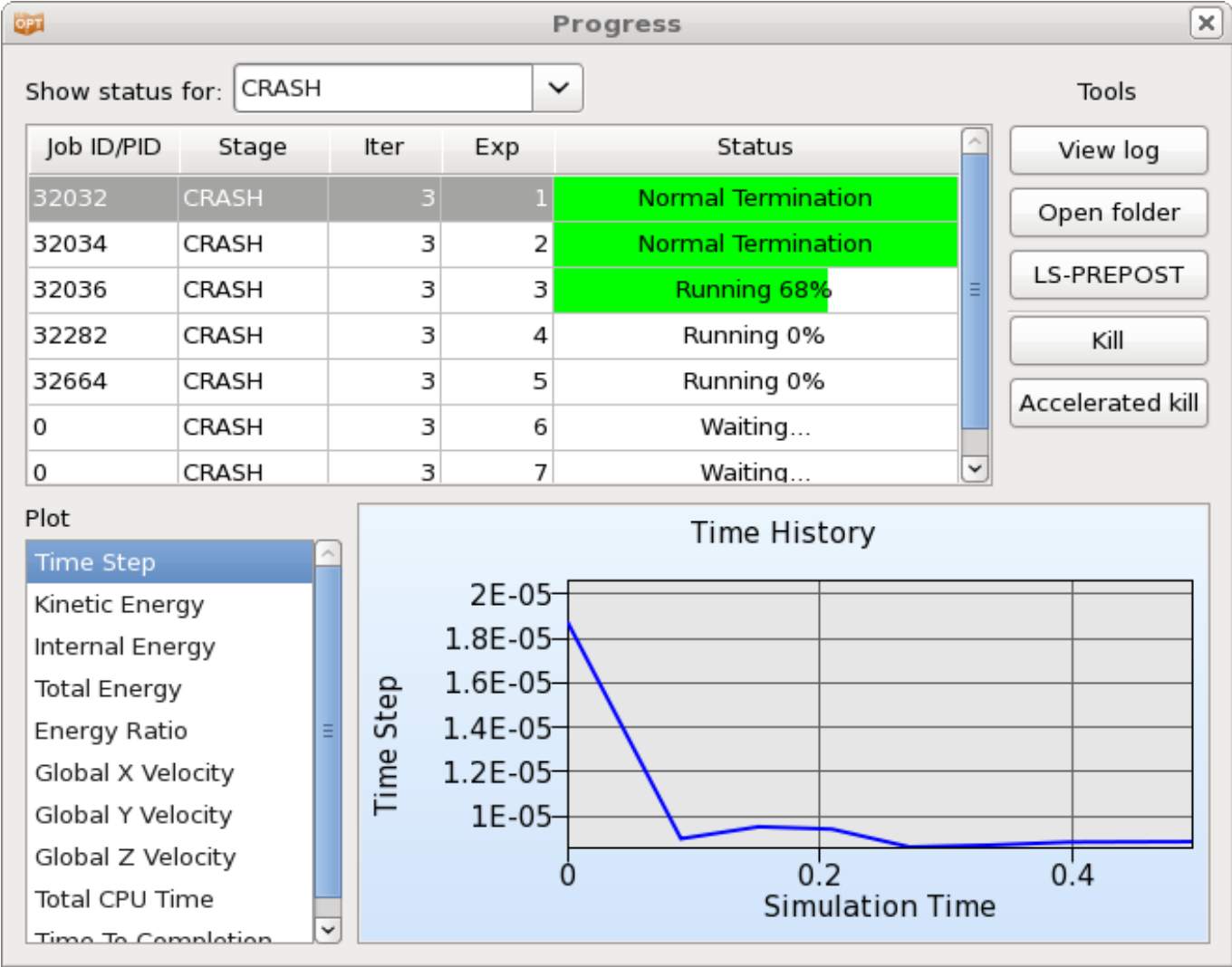


Figure 13-2: Progress dialog displaying progress of stage runs

**Table 13-1: Tools for selected run**

Tool	Description	Reference
View log	Opens <i>job_log</i> file of selected run	Section 13.6
Open folder	Opens run directory of selected job	-
LS-PREPOST	Opens selected run in LS-PREPOST (LS-DYNA only)	-
Kill	Kills selected job	Appendix I.2
Accelerated kill		Appendix I.2

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 13-1 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 13-2.

The **Progress** dialog allows a graphical indication of the job progress with the green horizontal bars linked to estimated completion time, Figure 13-2. This progress is only available for LS-DYNA 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 also reports 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.



### 13.3.1. 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 abnormally 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.

## 13.4. 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 available elsewhere (with the exception of the `AnalysisResults_n.lsox` database in the sampling 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.
  - 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.

After completing the repair functions above, a normal restart can be executed (  ).

*Note:* A restart will only be able to retain the data of the first iteration if more than one iteration was 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 5.6), 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 8.5.4).

## 13.5. 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 sub-directories are named *mmm.nnnn*, where *mmm* represents the iteration number and *nnnn* is a number starting from 1.

The work directory needs to contain at least the command file.

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. The creation of subdirectories is automatic and the user only needs to deal with the working directory.

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* and *history.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 5.4.5.

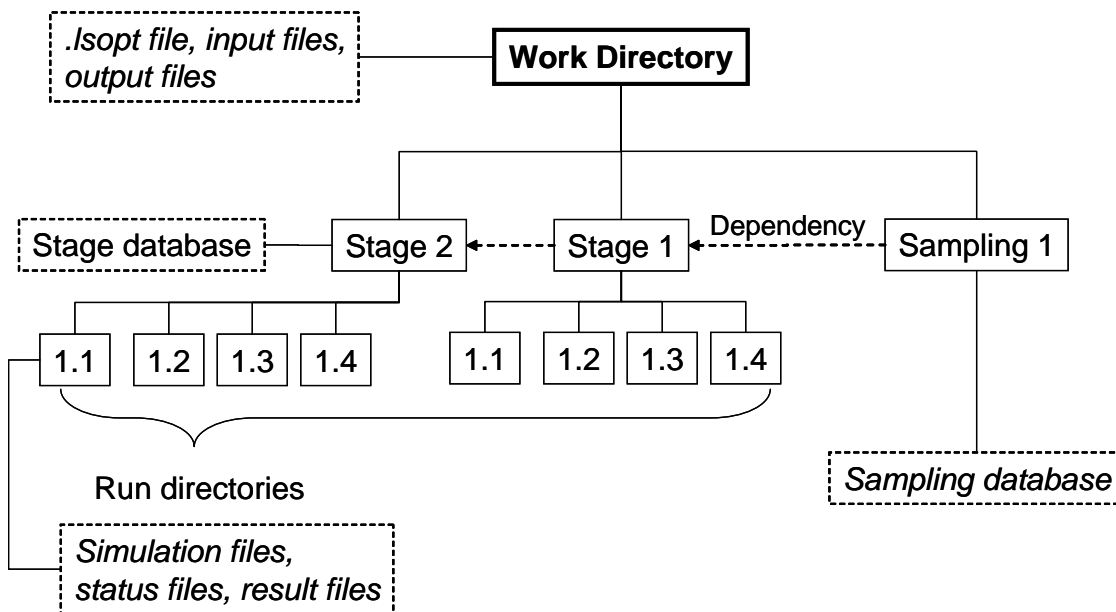


Figure 13-3 : Directory structure in LS-OPT

## 13.6. Log files and status files

Status files `started`, `finished`, `history.n`, `response.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 13-2: Status and log files generated by LS-OPT**

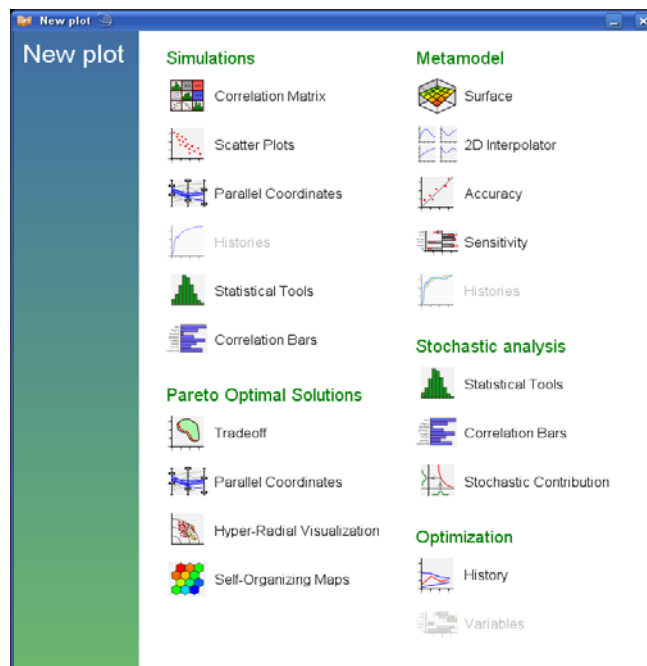
File	Description
<code>job_log</code>	The simulation run/extraction log is saved in that file in the local run directory.
<code>started</code>	The run has been started.
<code>finished</code>	The run has been completed. The completion status is given in the file.
<code>response.n</code>	Response number $n$ has been extracted.
<code>history.n</code>	History number $n$ has been extracted.
<code>EXIT_STATUS</code>	Error message after termination. The user interface <code>LS-OPT<sub>ui</sub></code> uses the message in the <code>EXIT_STATUS</code> file as a pop-up message.
<code>lfop.log</code>	The file contains a log of the core optimization solver solution.
<code>scheduler.debug</code>	This file is generated by the <code>lscheduler</code> executable and is used for debugging purposes.
<code>lsopt.debug</code>	Traceback of the solver termination. Used for debugging purposes.

# 14. Viewing Results

This chapter describes the post-processing of LS-OPT result data using the Viewer.

## 14.1. Viewer overview

### 14.1.1. Plot Selector



*Figure 14-1: Plot Selector*

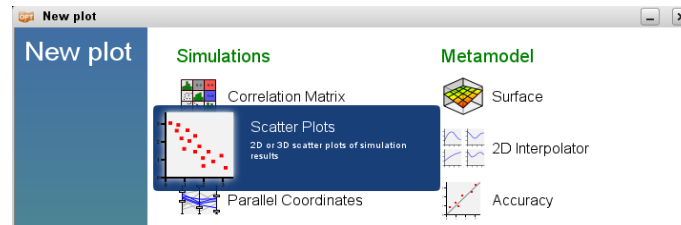
To start the Viewer, select the respective icon from the control bar of the main GUI.

The plots are grouped into five categories, Figure 14-1,

- Simulations,
- Metamodel,
- Optimization,
- Pareto Optimal Solutions and

- Stochastic Analysis.

Depending on the optimization task, the selected options and the database availability, categories are hidden or plots are disabled.



**Figure 14-2: Plot Selector with additional plot information**




Moving the mouse on a plot type gives additional information on the plot, Figure 14-2.



**Figure 14-3: Selection for placement of new plot in the plot selector**

If plots already exist, the placement of the new plot may be specified in the plot selector, Figure 14-3. The default is to create a new plot. All available options are explained in Table 14-1. For details see Section 14.1.5.

**Table 14-1: Plot placement options**

Option	Description
	Create a new plot window
	Replace current plot
	Split window and place new plot at the highlighted position














### 14.1.2. General Plot Options

General plot options are available on the toolbar at the top of the plot window, Figure 14-1. Table 14-2 explains the options.



Figure 14-4: General options

**Table 14-2: General plot options**

Option	Description
 New plot	Opens Plot Selector with placement selection for the new plot, see Section 14.1.1.
 Delete plot	Deletes active plot
 Save plot setup	Saves current plot setup to be reused later, see Section 14.1.6.
 Pointer tool [F1]	Rectangular selection (rubber-banding) in plot or clicking marks points or curves and opens Point selection window, see Section 14.1.4.
 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, see Section 14.1.5.
 Split horizontal	Splits plot window horizontal, see Section 14.1.5.
 Print	Prints the current plot, options see Figure 14-5.
 Save image	Saves the current plot, options see Figure 14-5.
 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 window	Shows or hides a window that shows the values of all entities for the selected points in a table. This window shows up automatically if the point selection changes, see section 14.1.4.

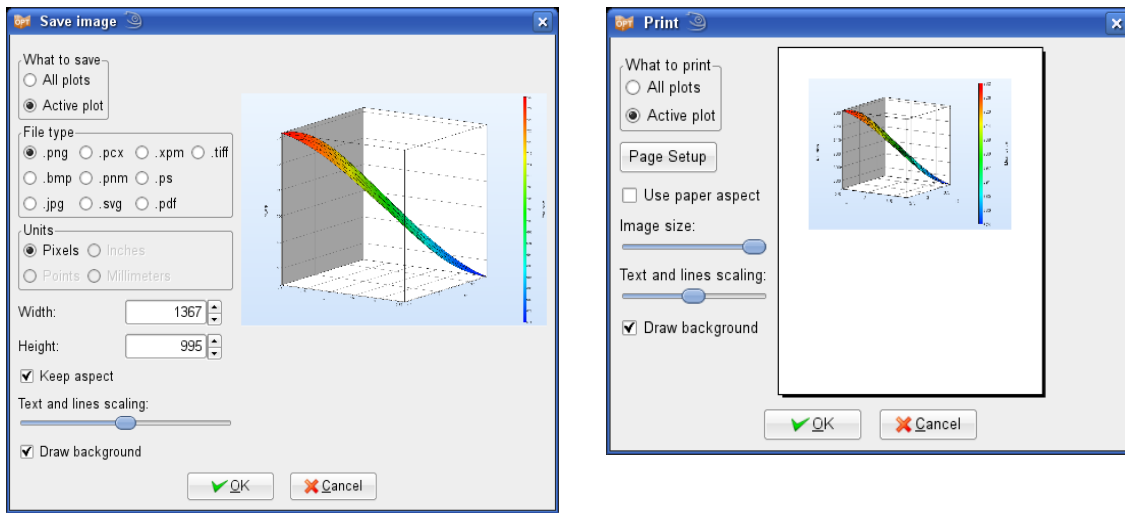


Figure 14-5: Options for printing (right) and saving (left) images

### 14.1.3. Plot Rotation

For all 3D plots, image rotation is performed by holding down the Ctrl key while moving the mouse (same as LS-PREPOST).

### 14.1.4. Point Selection

The points on Scatter, TradeOff, Surface, Accuracy, Optimization History and HRV plots, and lines on PCP, History plots may be selected by clicking on a single point or by selecting several points within a rectangular box. The selected points are highlighted in the plot and the computed and predicted values of all entities for the selected points are displayed in a spreadsheet in a separate plot selection window, Figure 14-6. Options for point selection are explained in Table 14-3. Points may also be selected from the list of all points available in the current plots on the left in the Point selection window.



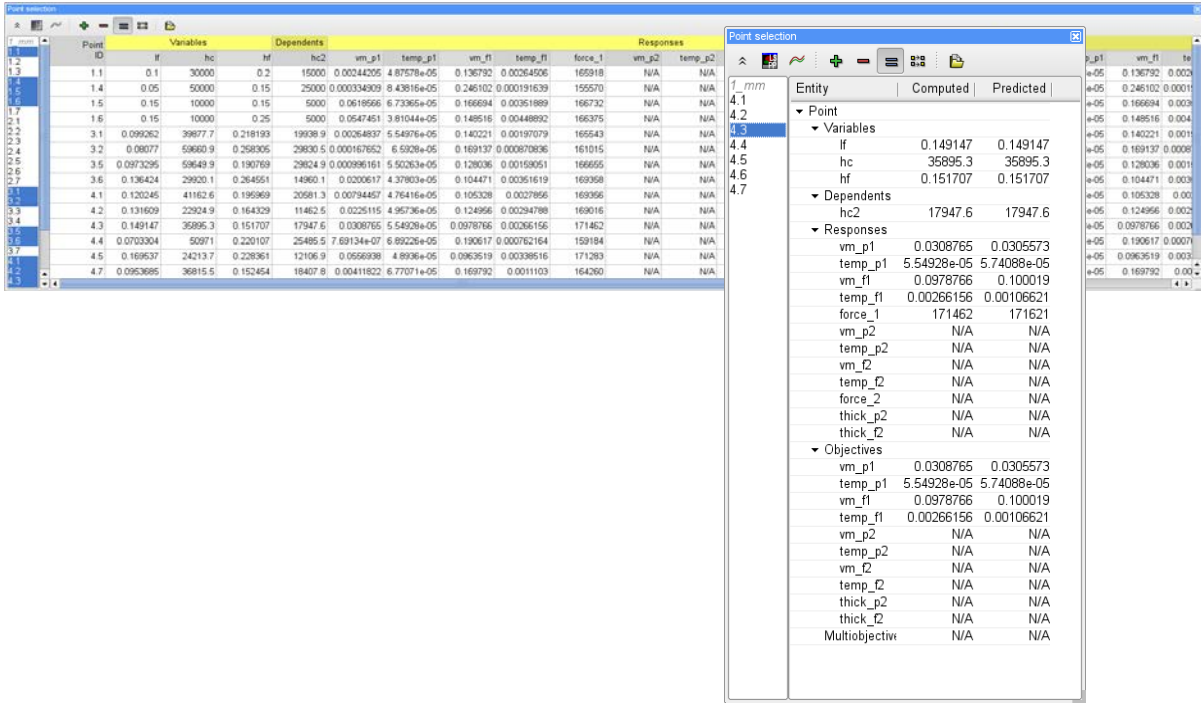









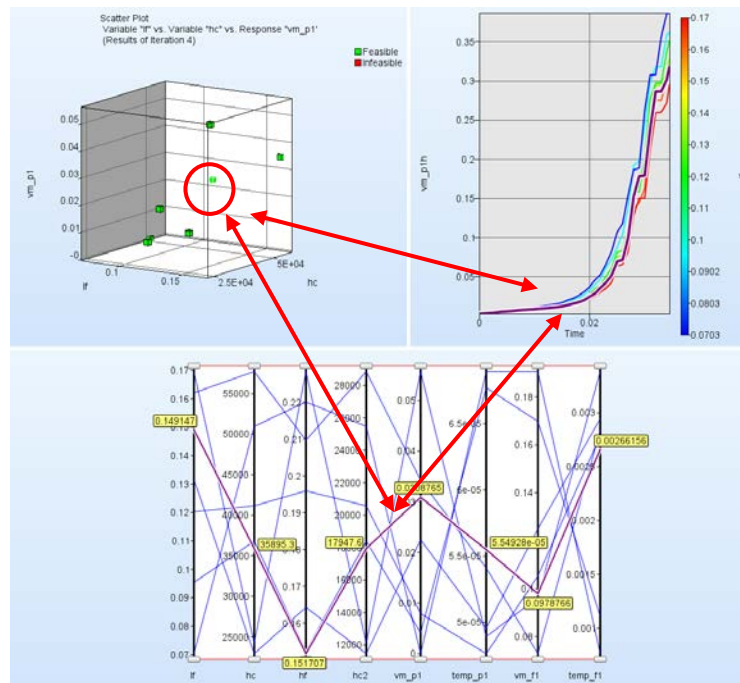
Figure 14-6: Point selection window for single (front) and multi (background) point selection

**Table 14-3: Point selection options**

Option	Description
 Open selected run in LS-PREPOST	This option is only available for LS-DYNA simulations, if the d3plot or d3eigv database is available
 Display histories involved in a curve matching composite	This option is only available if MeanSqErr composites are defined. It is recommended to use the history plot feature (Section 14.2.4) for comparing histories (including file-based histories typically used with curve matching).
 Add to set of selected points	Options for new point selection
 Subtract from set of selected points	
 Replace set of selected points	
 Toggle set of selected points (within rectangle)	
 Export as text file (.csv)	The exported file has the format of a user defined sampling

The SOM plot also supports selection. If a cell is selected, all points that are mapped to the selected cell are displayed in the point selection window.

Point selection is integrated, hence selected points are highlighted in all plots within the same plot window.



*Figure 14-7: Cross-display of selected points*

### 14.1.5. 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 same plot is repeated with the same settings which is useful for e.g., displaying 3D surface plots for different responses side by side, Figure 14-8.

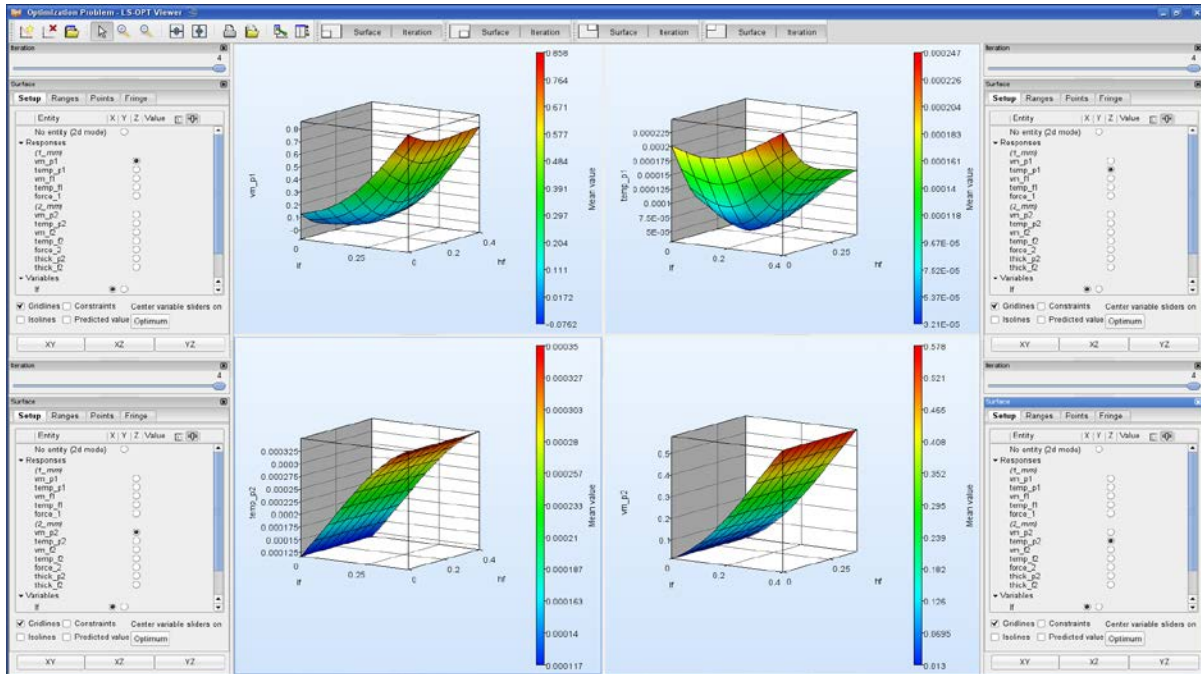


Figure 14-8: Example for split option

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 14-9.

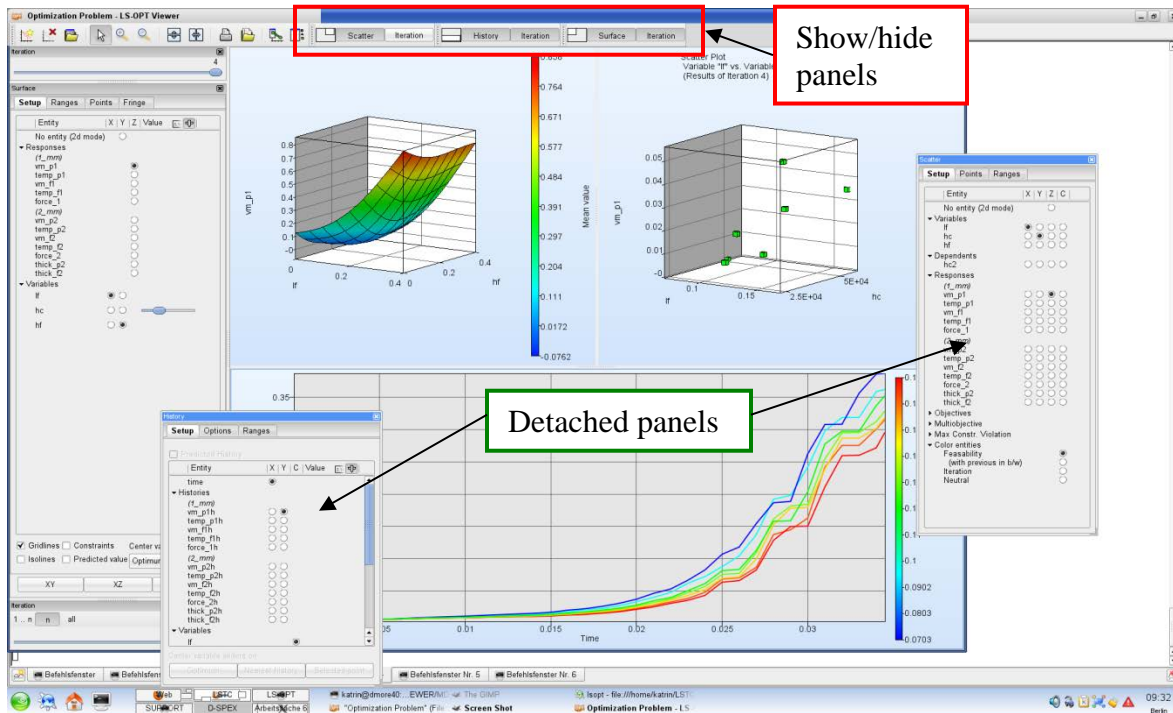


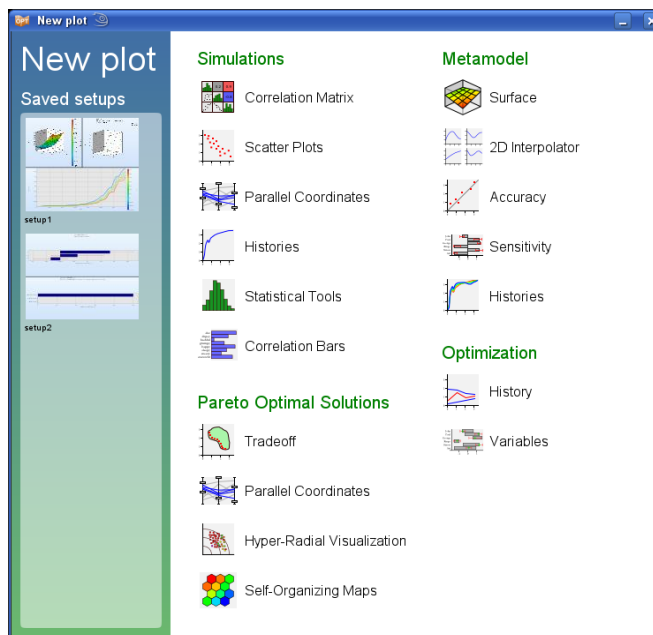
Figure 14-9: Detachable panels

### 14.1.6. 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 14-10.

The plot setup is stored in XML format in `~/LS-OPT Viewer/plotname.plot` on Linux machines. On Windows, the plot setup is stored in `Application Data\LS-OPT Viewer` in the user's home directory. The full path depends on the Windows version and setup, e.g. `C:\Documents and Settings\user\Application Data\LS-OPT Viewer`.

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 14.1.7.



**Figure 14-10:** Plot Selector with previously saved setups

### 14.1.7. 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>] [-h] [--verbose] [com-file]
```

Table 14-4 explains the command line options.

Table 14-4: Command line options

Option	Description
-p <str>, --show-plot=<str>	Open the given plot, valid plot types are
<i>accuracy</i>	Accuracy (Section 14.3.3)
<i>correlation</i>	Correlation Bars (Section 14.2.6)
<i>cormatrix</i>	Correlation Matrix (Section 14.2.1)
<i>history</i>	Histories – Metamodel (Section 14.3.5)
<i>history_ar</i>	Histories – Simulations (Section 14.2.4)
<i>hrv</i>	Hyper-Radial Visualization (Section 14.5.3)
<i>interpol</i>	2D Interpolator (Section 14.3.2)
<i>ophist</i>	Optimization History (Section 14.4.1)
<i>parallelcoord</i>	Parallel Coordinates – Pareto Optimal Solutions (Section 14.5.2)
<i>parallelcoord_ar</i>	Parallel Coordinate – Simulations (Section 14.2.3)
<i>scatter</i>	Scatter Plots (Section 14.2.2)
<i>sensitivities</i>	Sensitivity (Section 14.3.4)
<i>som</i>	Self-Organizing Maps – Pareto Optimal Solutions (Section 14.5.4)
<i>statistics</i>	Statistical Tools (Section 14.2.5)
<i>stoch</i>	Stochastic Contribution (Section 14.6.3)
<i>surface</i>	Surface (Section 14.3.1)
<i>tradeoff</i>	Tradeoff (Section 14.5.1)
<i>variable</i>	Variables (Section 14.4.2)
-l <str>, --load-setup=<str>	Load plot Setup from file, see section <b>Save Plot Setup</b>
-h, --help	show help message for command line options

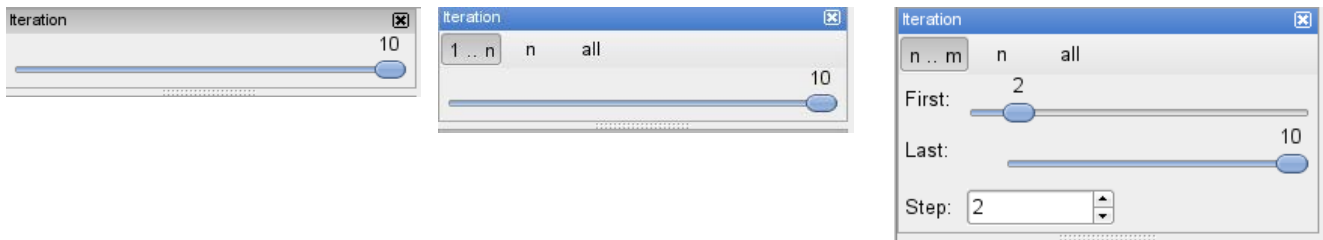
---

<code>--verbose</code>	generate verbose log messages
<code>com-file</code>	LS-OPT command file. By default, the viewer loads the LS-OPT database called <code>lsopt_db</code>

---

### 14.1.8. Iteration Panel

Except for the Optimization 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 14-11).



**Figure 14-11: 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 14-11 plots iterations 2,4,6,8 and 10.

### 14.1.9. Ranges

Most plots allow specifications of the ranges for all plotted entities, Figure 14-12.

The default is **Auto**. In this case, the range is set to the minimal and maximal value to plot.

For surface plots, there are two options for **Auto** range selection. **Auto, Entire design space** plots the surface across the full design space, **Auto, region of interest** uses only the subregion of the selected iteration.

If **never shrink plot range** is selected, the ranges of the previous plot are considered and they are enlarged if the new plot has values outside that range, see the Neural Net plot displayed in Figure 14-26. This option is ignored if entities change.

**Manual** range selection allows the user to specify lower and upper bounds for each plotted entity.

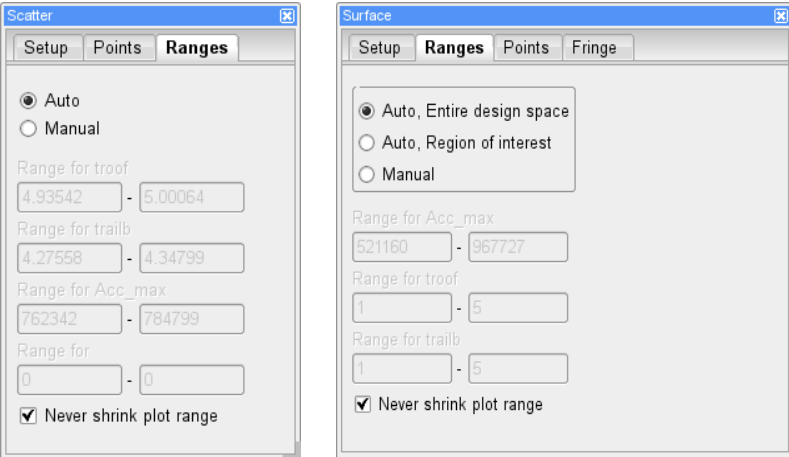


Figure 14-12: Ranges selections

## 14.2. Visualization of Simulation Results

### 14.2.1. 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 14-13.

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's also possible to display e.g. only correlation coefficients, Figure 14-14.

By double-clicking on a scatter plot or histogram, the respective plot may be reached, see Section 14.2.2 or Section 14.2.5, 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.



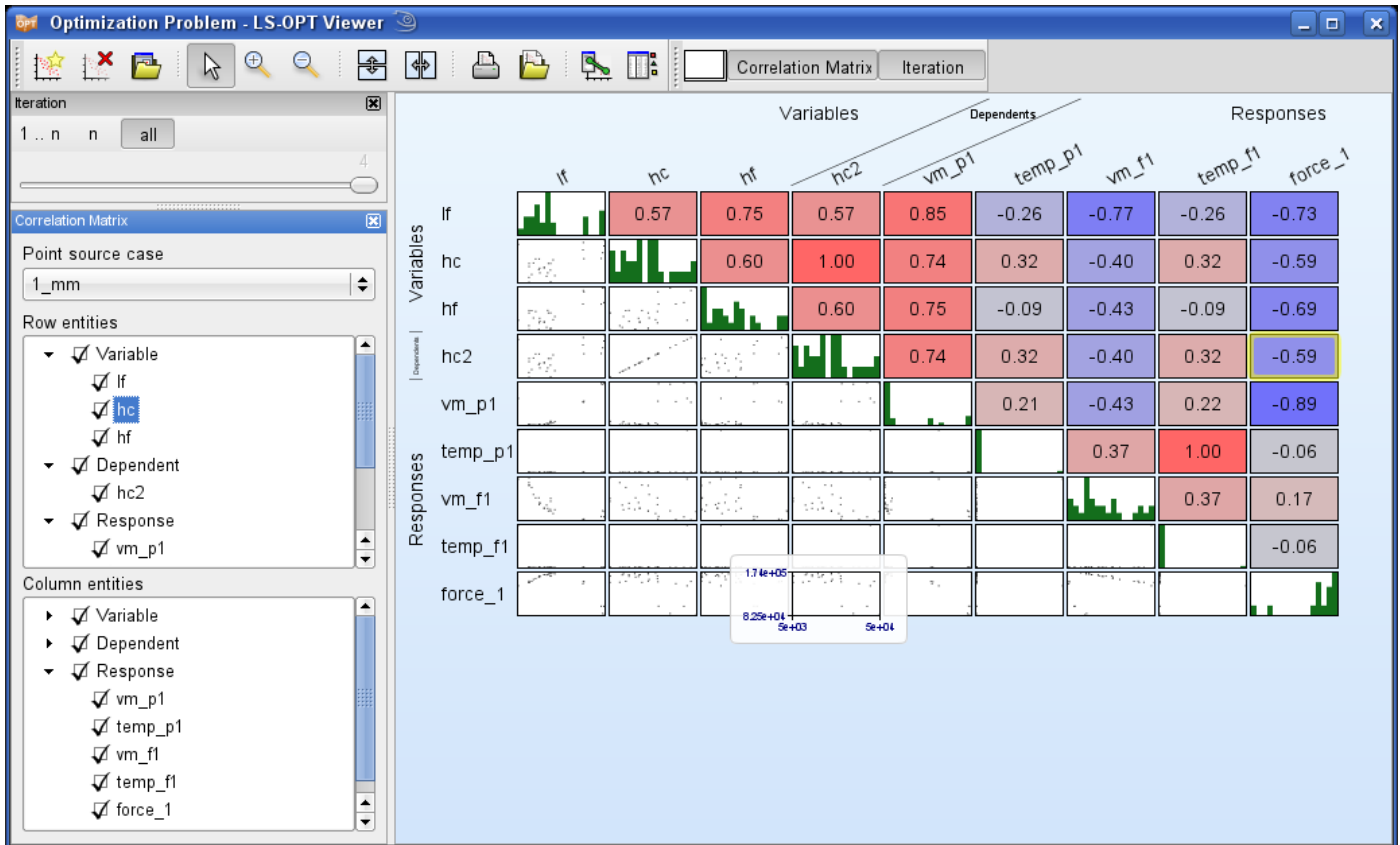


Figure 14-13: Correlation matrix with scatter plots, histograms and linear correlation coefficient

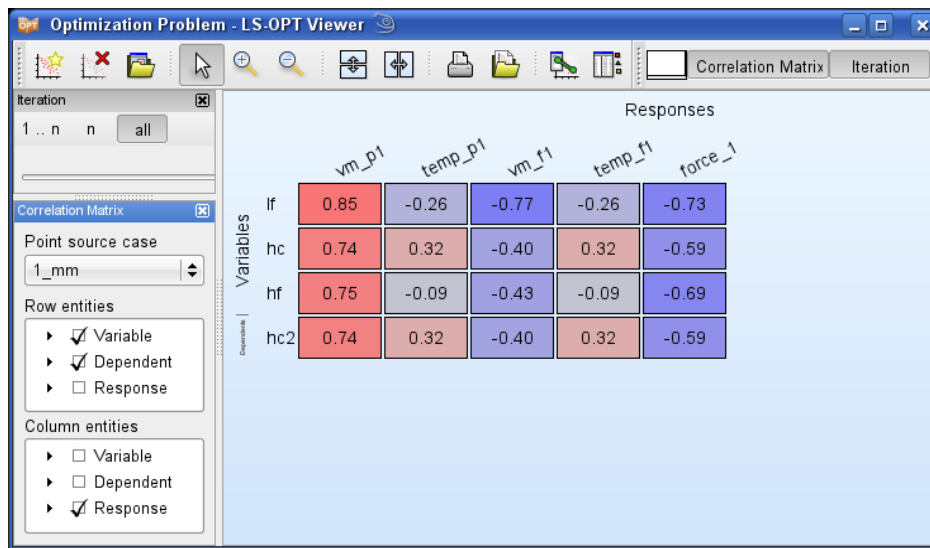


Figure 14-14: Correlation matrix, only correlation coefficients

### 14.2.2. Scatter Plot

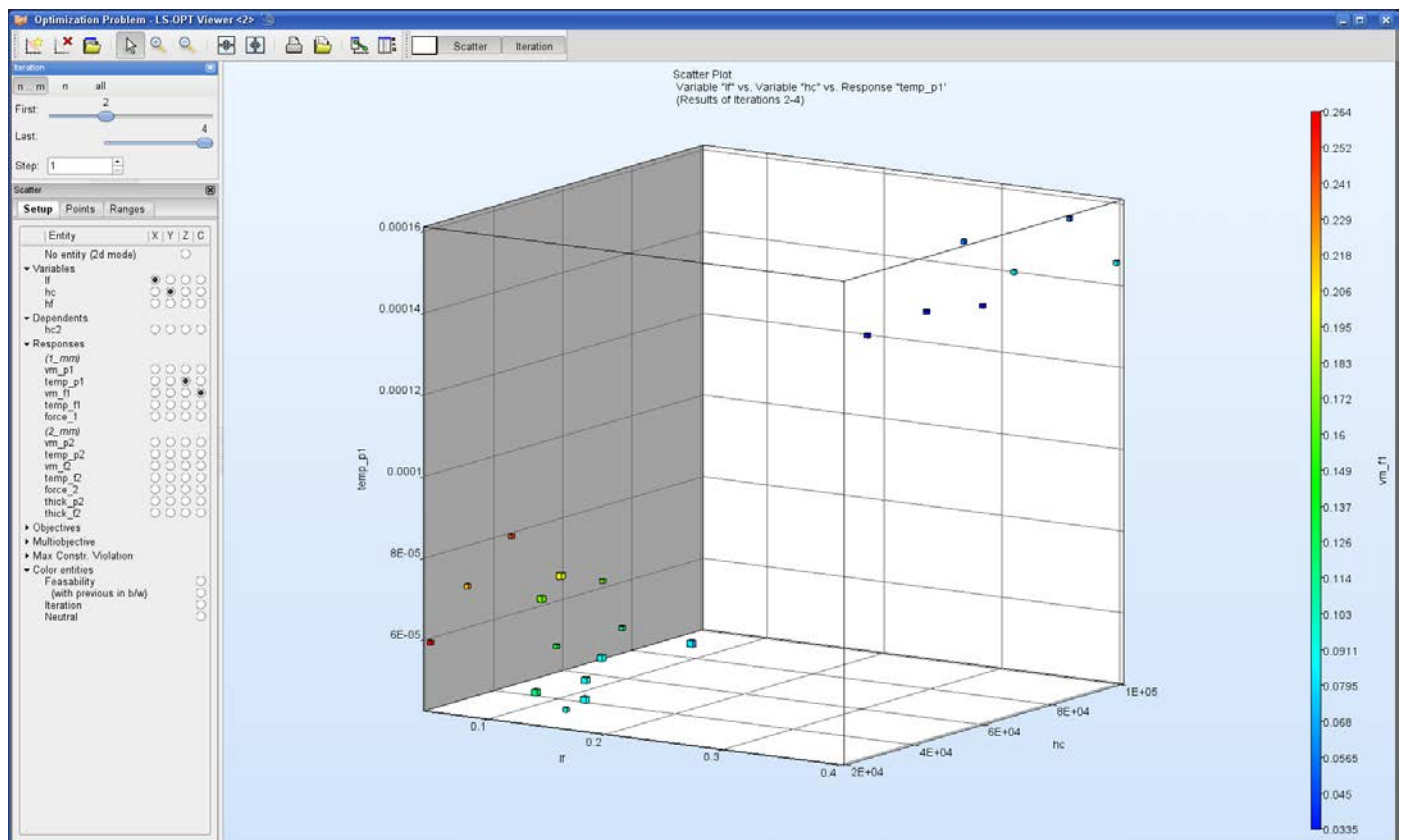
The results of all the simulated points for the selected iterations appear as dots on the scatter plots. This feature allows the three-dimensional plotting of any three entities. A fourth entity may be displayed using the color of the points. Other coloring options are explained below. 2D plots can be 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 stage, the same sampling (Section **23.2**) 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 14-5: Color entity options**

Selection	Description
<i>Feasibility</i>	Feasible points are shown in green, infeasible points in red
<i>Previous b/w</i>	The points for the current iteration are shown in green (feasible) or red (infeasible). Previous points as light grey (feasible) or dark grey (infeasible)
<i>Iterations</i>	The iteration sequence is shown using a color progression from blue through red.
<i>Neutral</i>	All points are shown in blue



**Figure 14-15: Scatter plot in View panel in LS-OPTui. The 4<sup>th</sup> dimension is represented by point color.**

## Points Options

**Table 14-6: Point options**

Selection	Description
<i>Analysis Results</i>	Plot simulation results (Default for the Scatter Plot)
<i>Pareto Optimal Solutions</i>	Plot Pareto optimal solutions (Default for the Tradeoff Plot, Section 14.5.1)
<i>Use reduced set of points</i>	Only active for Pareto optimal solutions, plots 100 uniformly distributed points selected from the Pareto optimal solutions

### 14.2.3. 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 poly-line connecting the respective values on the vertical axis, Figure 14-16. 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.

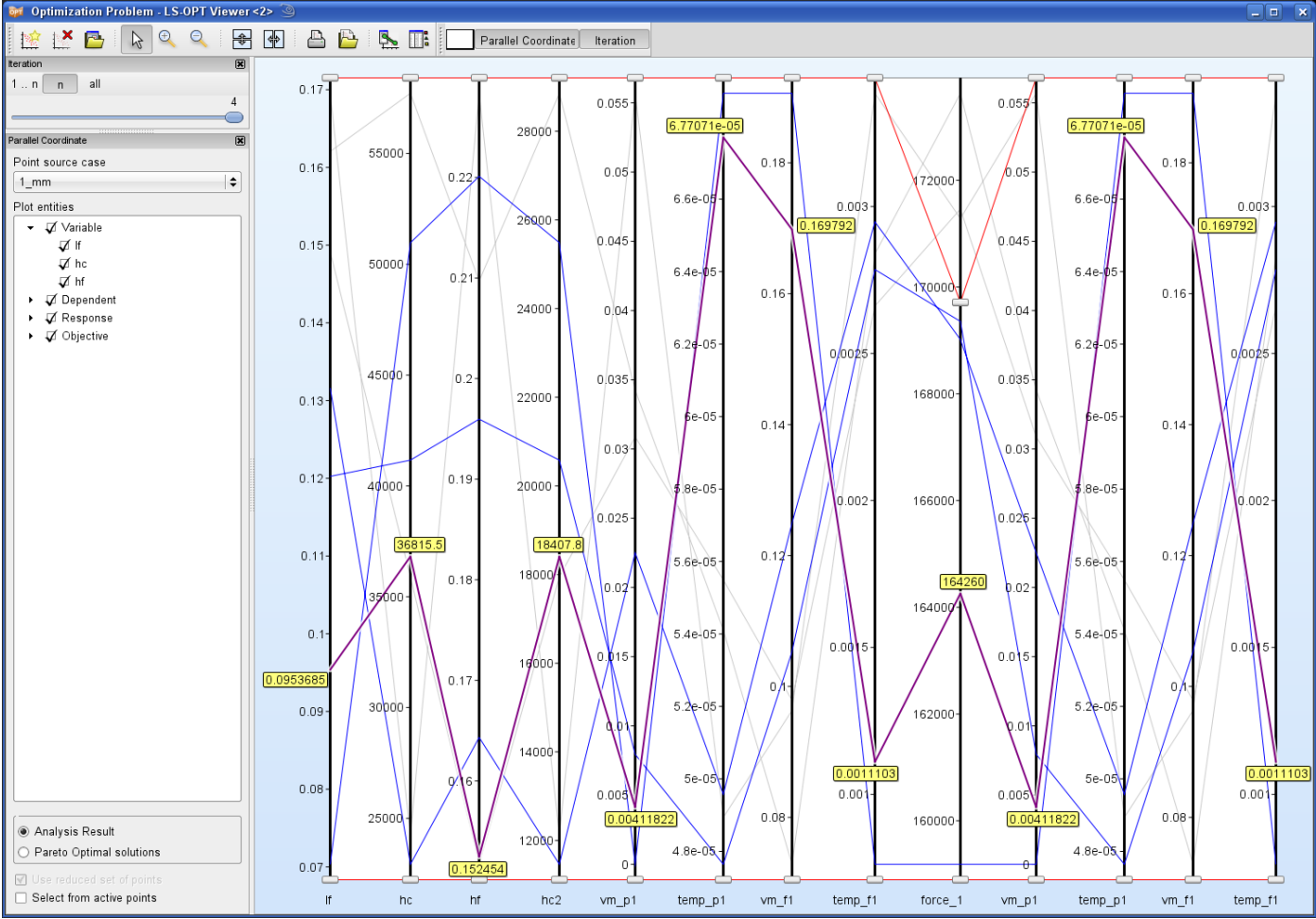


Figure 14-16: Parallel Coordinate Plot with selected point

## Options

**Table 14-7: Parallel Coordinat Plot options**

Selection	Description
<i>Analysis Results</i>	Plots simulation results (Default)
<i>Pareto Optimal Solutions</i>	Plots Pareto optimal solutions (Section 14.5.2)
<i>Use reduced set of points</i>	Only active for Pareto optimal solutions; plots 100 points selected from the Pareto optimal solutions
<i>Select from active points</i>	Selects all points that are not outside the constraints set by the handles, see Section 14.1.4. Useful for visualizing this set of points in another plot.

### 14.2.4. History Plot

This plot visualizes history curves based on time data or crossplots obtained from simulations, Figure 14-17. The coloring options are the same as the point coloring options, see 14.2.2. If histories from files are defined in the optimization problem, they can be visualized in addition to the simulation curves, Figure 14-18.

The **Predicted History** option is explained in Section 14.3.5.

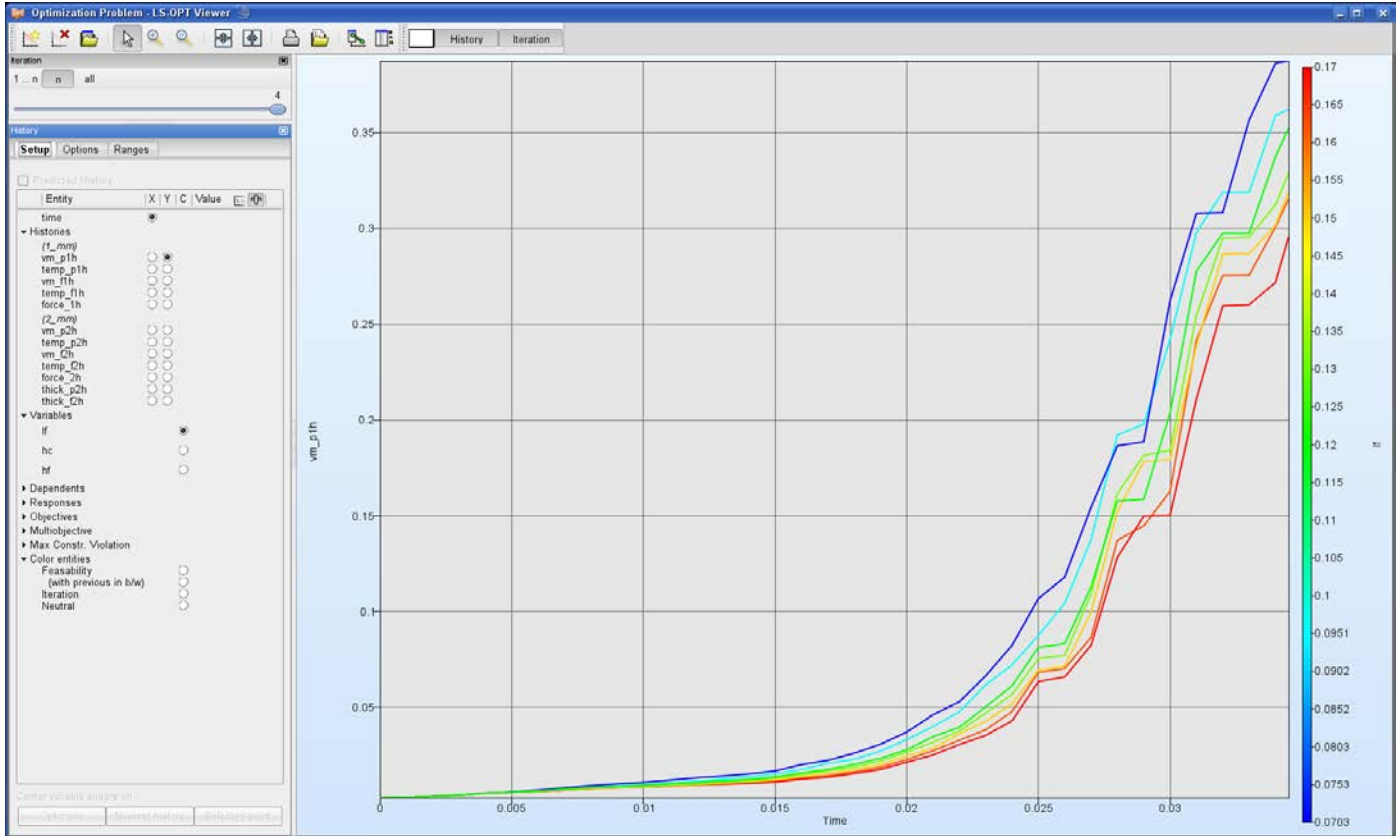
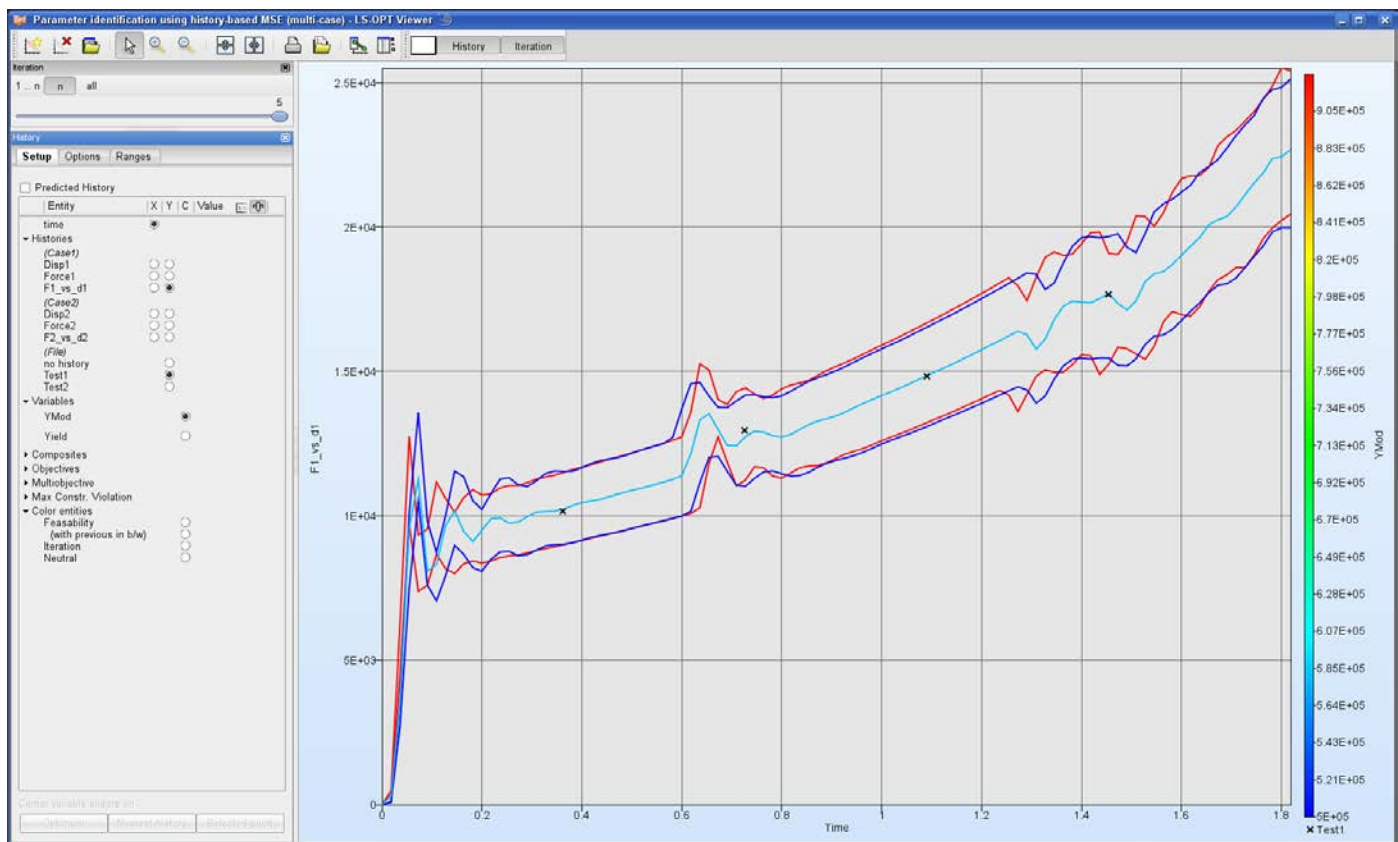


Figure 14-17: History Plot, curves colored by variable

## Options

**Table 14-8: History Plot options**

Selection	Description
<i>Feasible</i>	Plot feasible runs
<i>Infeasible</i>	Plot infeasible runs
<i>Only selected</i>	Plot only selected runs, see 14.1.4., in this case, the selected curves are not highlighted



**Figure 14-18: Histories from simulations colored by variable with target curve (File history)**

### 14.2.5. Statistical Tools

The Statistical Tools option offers three types of plots, Histogram, Summary and Bounds.

The feature enables plotting either (i) the simulation results directly or (ii) using the metamodels together with the statistical distribution of the variables to construct the plots. The simulation results will be read from the ExtendedResults file of the relevant solver. If the use of the metamodels is selected 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 values. The user can control the number of points in this Monte Carlo simulation.

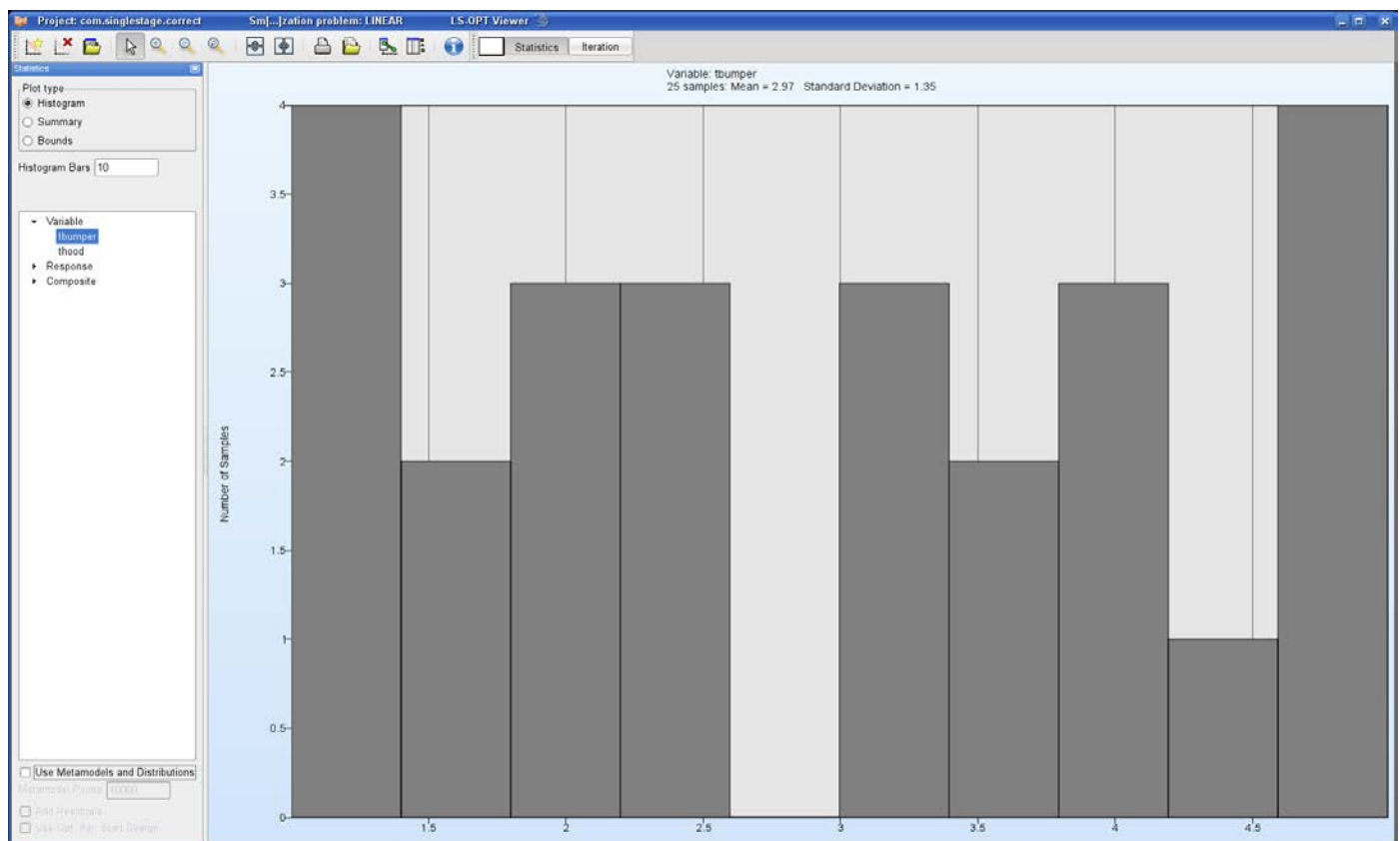
If desired, the residuals of the metamodel fit can be added to results of the Monte Carlo simulation as a normal distribution.

For optimization results, an iteration can be selected, while for probabilistic evaluations the default iteration, iteration 1, will automatically be selected.

## Histogram

Histograms of the variables, dependents, responses, and composites are available. The number of histogram bars may be specified by the user.

The histogram panel is shown in Figure 14-19.



**Figure 14-19: Histogram constructed from simulation results**

### Summary

Here, the standard deviation and the mean value for the selected variable, dependent, response or composite is visualized with the 95% confidence interval in red, Figure 14-20.

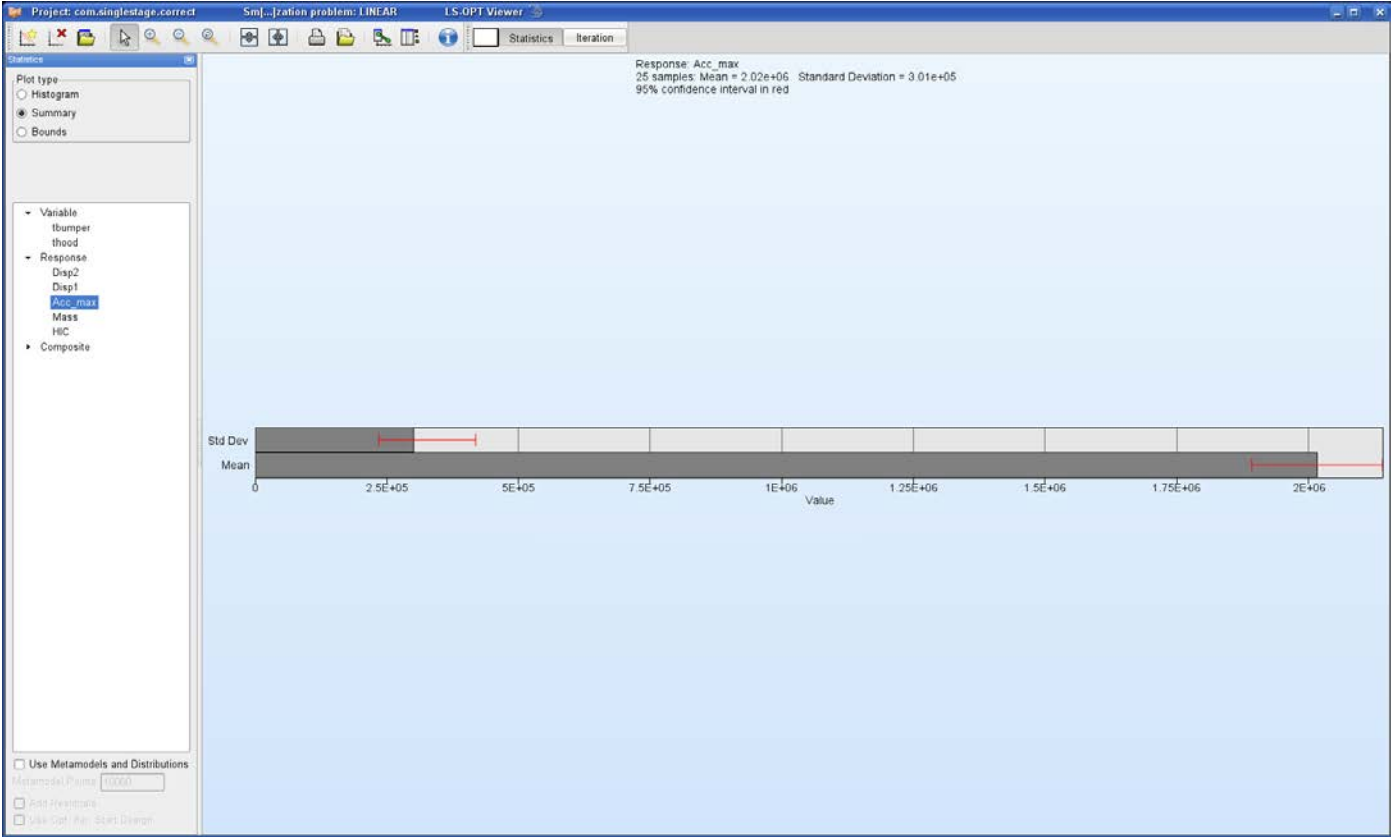
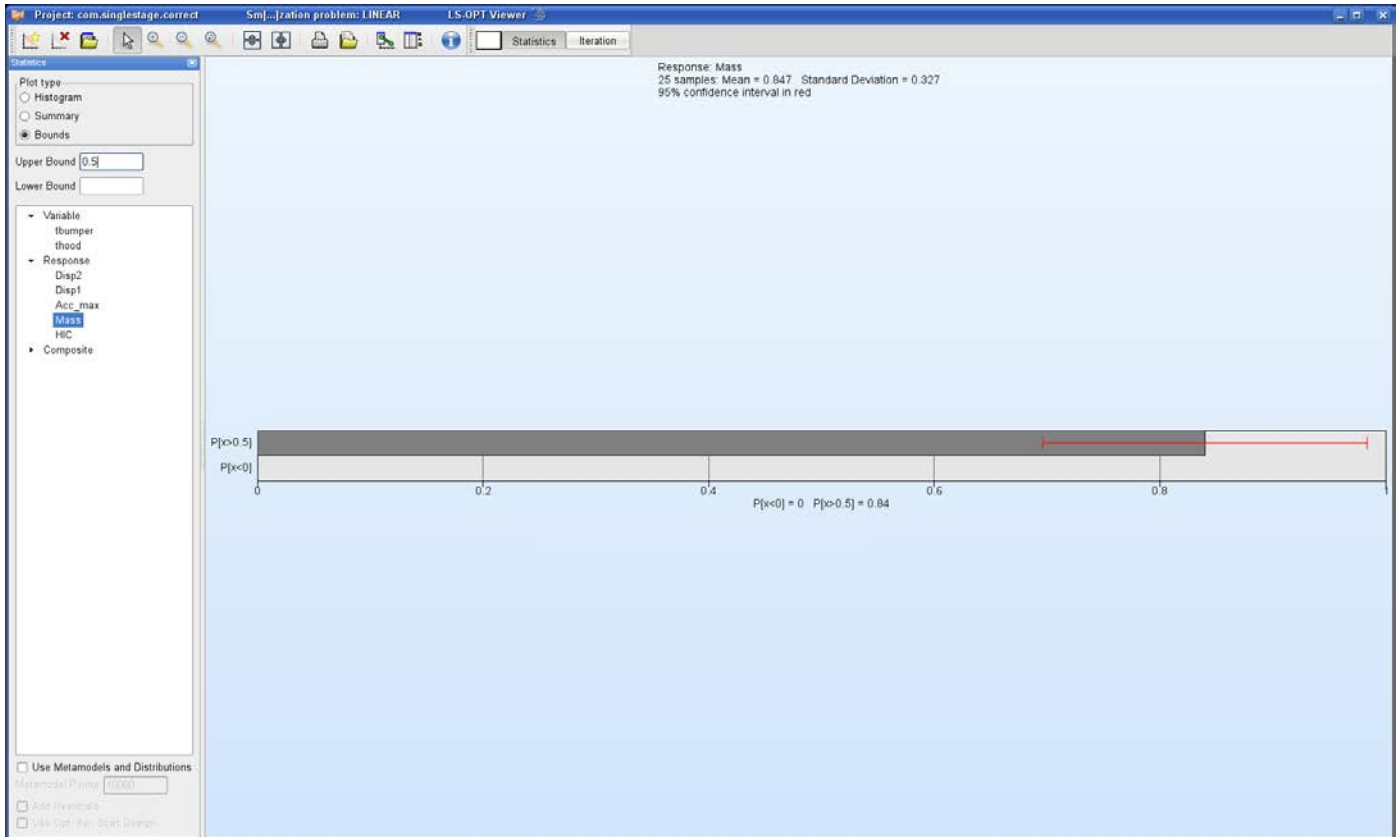


Figure 14-20: Standard deviation and mean value of selected response constructed from simulation results

## Bounds

The user may specify lower and upper bounds, respectively, for the selected variable, dependent, response or composite. The probabilities that the entity exceeds the bounds are visualized with 95% confidence interval in red, Figure 14-21.



**Figure 14-21: Probability of Mass > 0.5 with 95% confidence interval in red constructed from simulation results**

## Histogram Options

*Table 14-9: Histogram options*

Selection	Description
<i>Use Metamodels and Distributions</i>	Use metamodels and statistical distribution of the input variables to construct statistics
<i>Metamodel Points</i>	Number of points used for Monte Carlo Simulation on the metamodel to construct statistics
<i>Add Residuals</i>	Add residuals of the metamodel fit (“noise”) to the results of the Monte Carlo simulation as a normal distribution
<i>Use Opt. Iter. Start Design</i>	Use starting design for iteration to display statistics

### 14.2.6. Correlation Bars

Both the covariance and the coefficient of correlation of the responses and composites with respect to the design variables can be displayed, Figure 14-22.

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 user can control the number of points in the Monte Carlo simulation.

The plot can be used to estimate the stochastic contribution of an analysis without a metamodel.

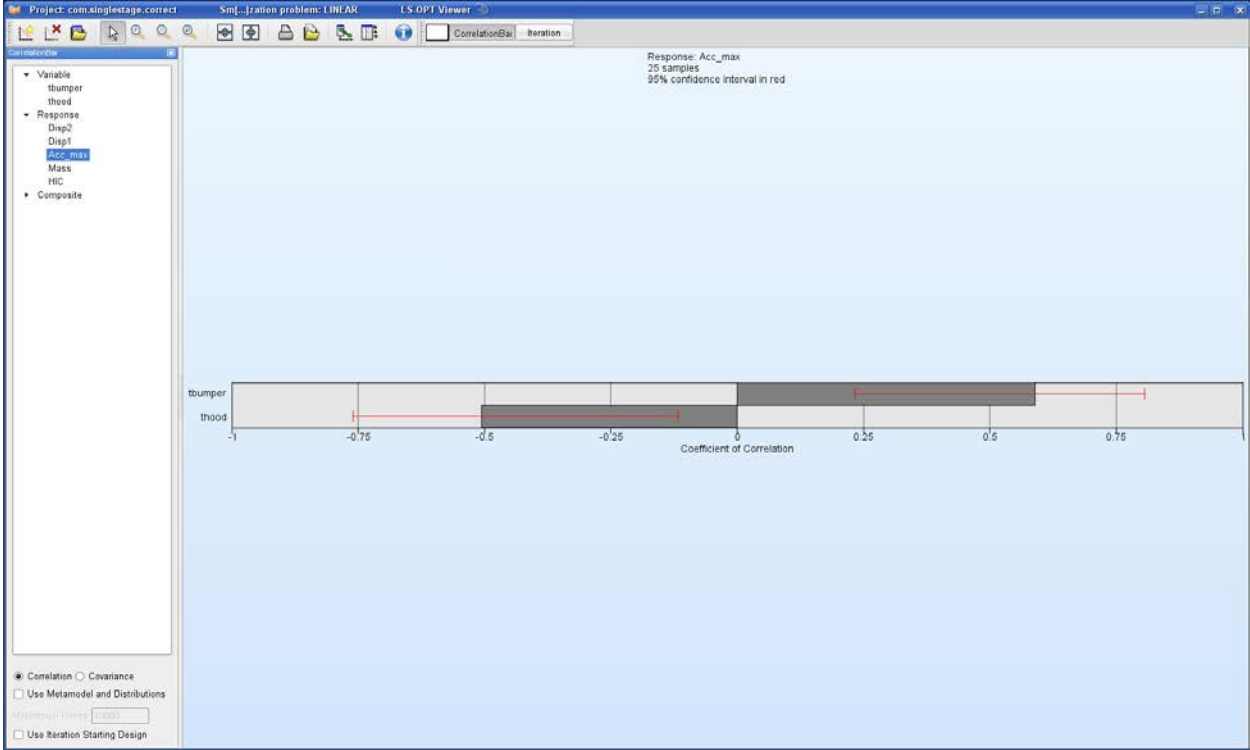


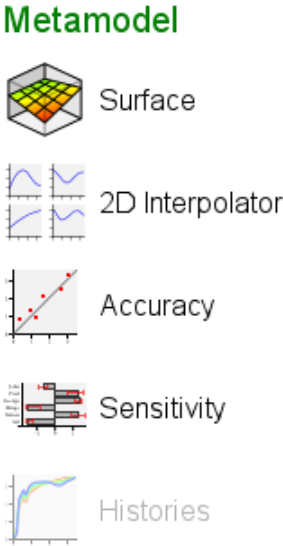
Figure 14-22: Coefficient of Correlation plot with 95% confidence interval in red

**Options**

*Table 14-10: Correlation Bar options*

Selection	Description
<i>Correlation</i>	Plot correlation coefficient
<i>Covariance</i>	Plot covariance
<i>Use Metamodels and Distributions</i>	Use metamodels and statistical distribution of variables to construct statistics
<i>Metamodel Points</i>	Number of points used for Monte Carlo Simulation on the metamodel to construct statistics
<i>Use Opt. Iter. Start Design</i>	Use starting design for iteration to display statistics

**14.3. Visualization of Metamodel Results**



*Figure 14-23: Metamodel options*

**14.3.1. 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 allow changing the variable values for unselected variables (variables not plotted). The slider for the active variables can be activated by selecting the “Predicted Value” option.

**Table 14-11: Surface Plot Setup options**

Selection	Description
<i>Gridlines</i>	Gridlines are displayed on the surface, Figure 14-24
<i>Isolines</i>	Isolines are displayed on the surface, Figure 14-25
<i>Constraints</i>	Constraints are displayed on the surface, Figure 14-27. 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.
<i>Predicted value</i>	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 14-24
<i>Center variable sliders on Optimum</i>	Variable sliders are set to optimal values of selected iteration

## Point plotting options

**Table 14-12: Surface Plot point plotting options**

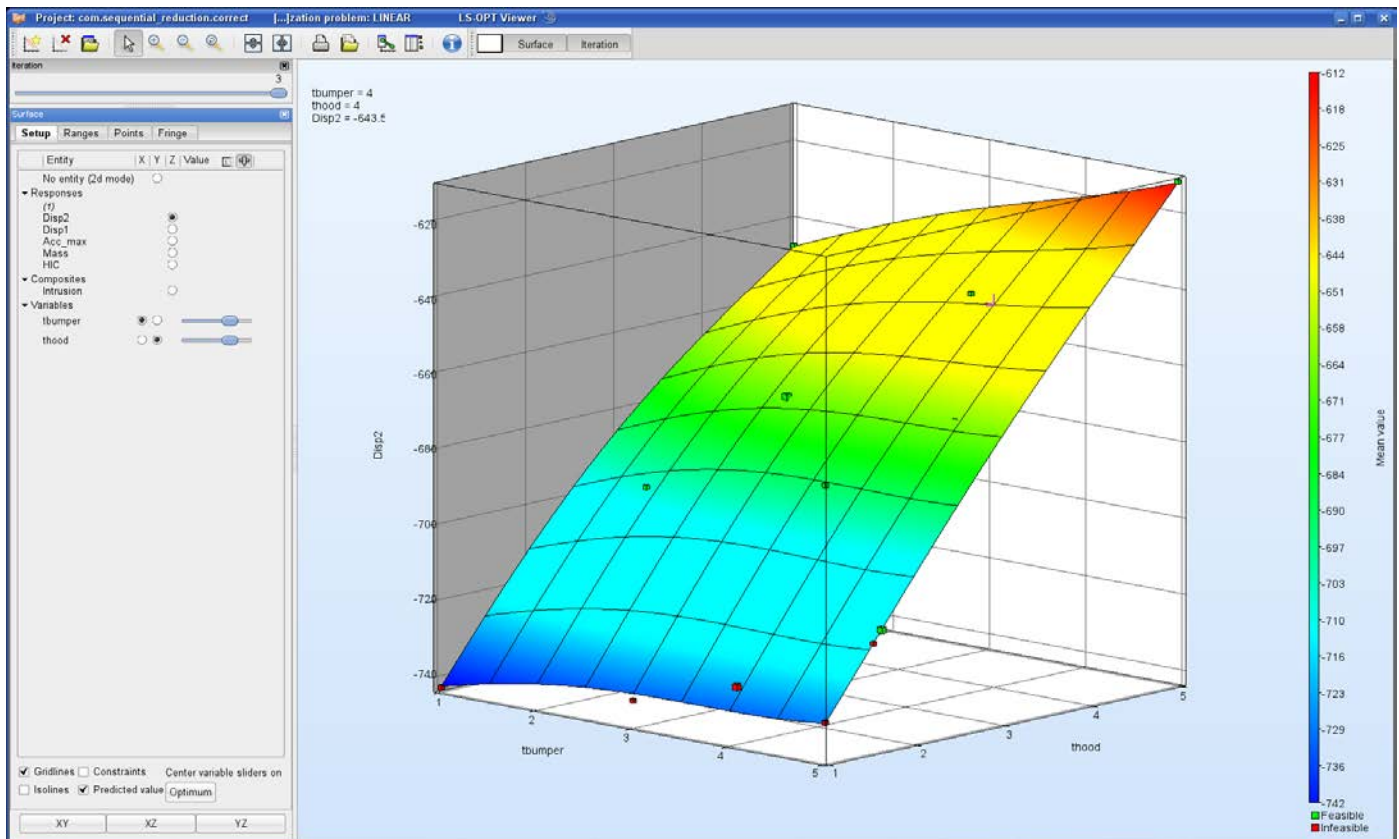
Selection	Description
<i>Feasible</i>	Show feasible runs only
<i>Infeasible</i>	Show infeasible runs only
<i>Predicted Optimum</i>	Show predicted optimum
<i>Computed Optimum</i>	Show computed optimum
<i>Failed runs on surface</i>	Failed runs such as error terminations are projected to the surface in grey
<i>Points only</i>	Show only points without surface
<i>Project points to surface</i>	The points are projected on the surface to improve visibility. Future versions will have a transparency option.
<i>Show Residuals</i>	Shows a black vertical line connecting the computed and predicted values.



## Point status

**Table 14-13: Surface Plot point status options**

Selection	Description
<i>Feasibility</i>	Feasible points are shown in green, infeasible points in red
<i>Previous b/w</i>	The points for the current iteration are shown in green (feasible) or red (infeasible). Previous points as light grey (feasible) or dark grey (infeasible)
<i>Iterations</i>	The iteration sequence is shown using a color progression from blue through red. See Figure 14-25.
<i>Optimum runs</i>	Optimal points are shown in green/red and all other points in white.



**Figure 14-24: 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.**

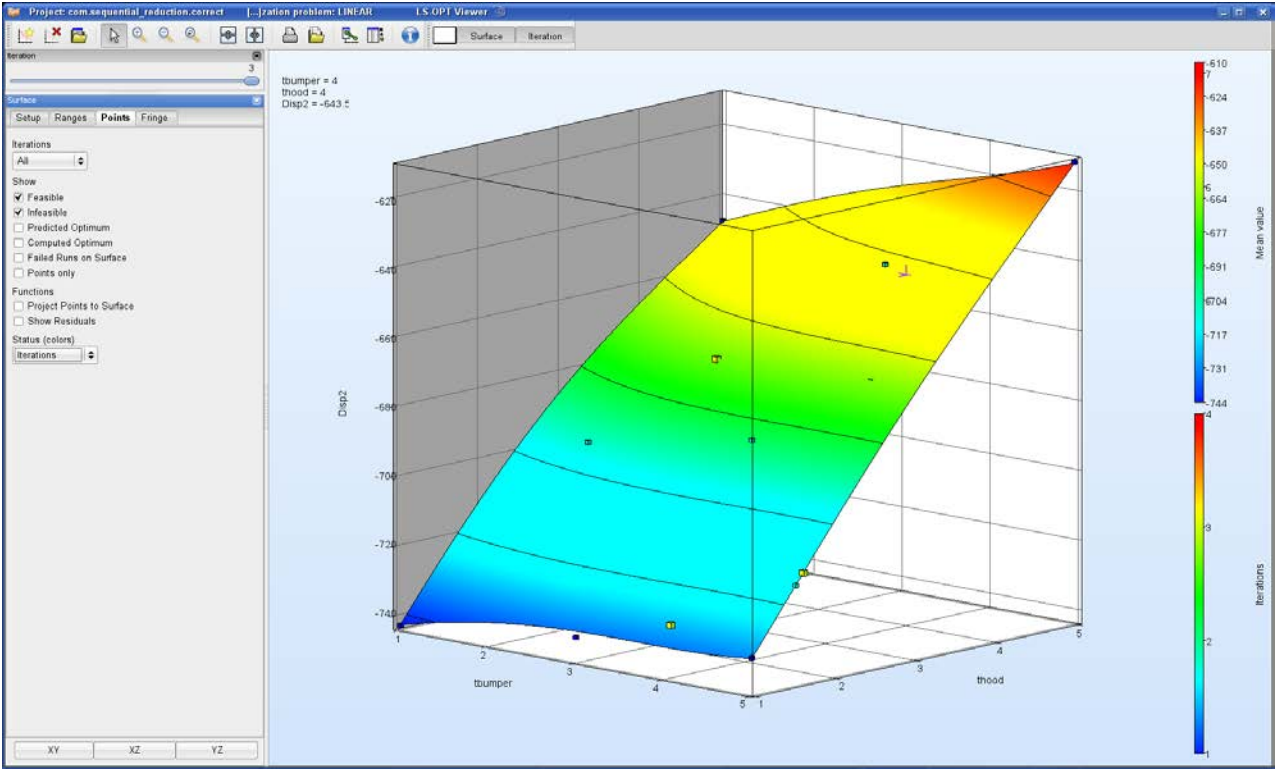


Figure 14-25: Metamodel plot showing point color coding for iteration numbers.

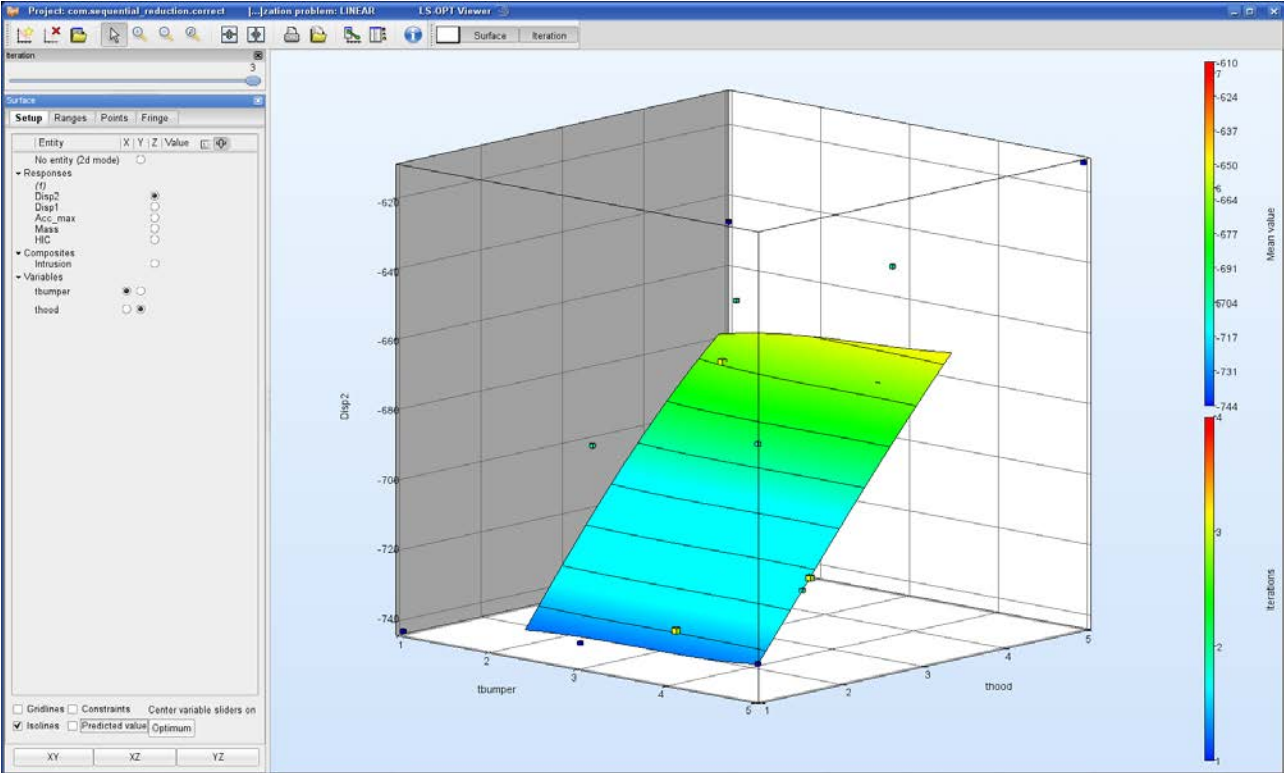
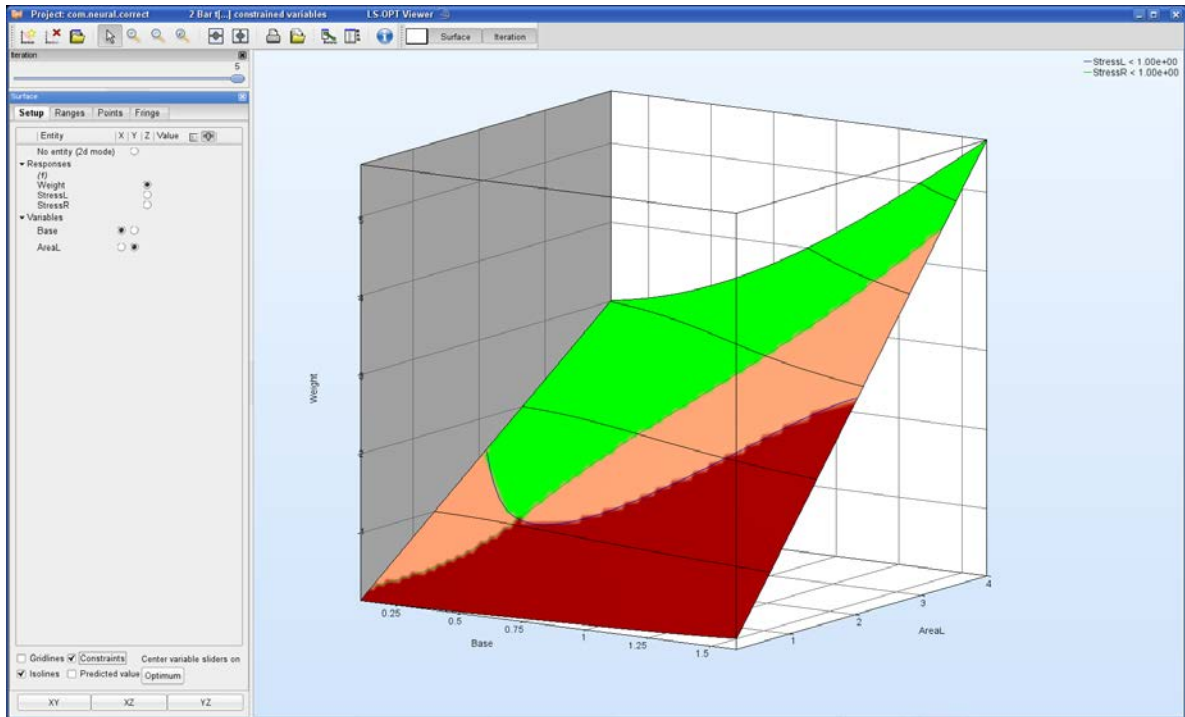
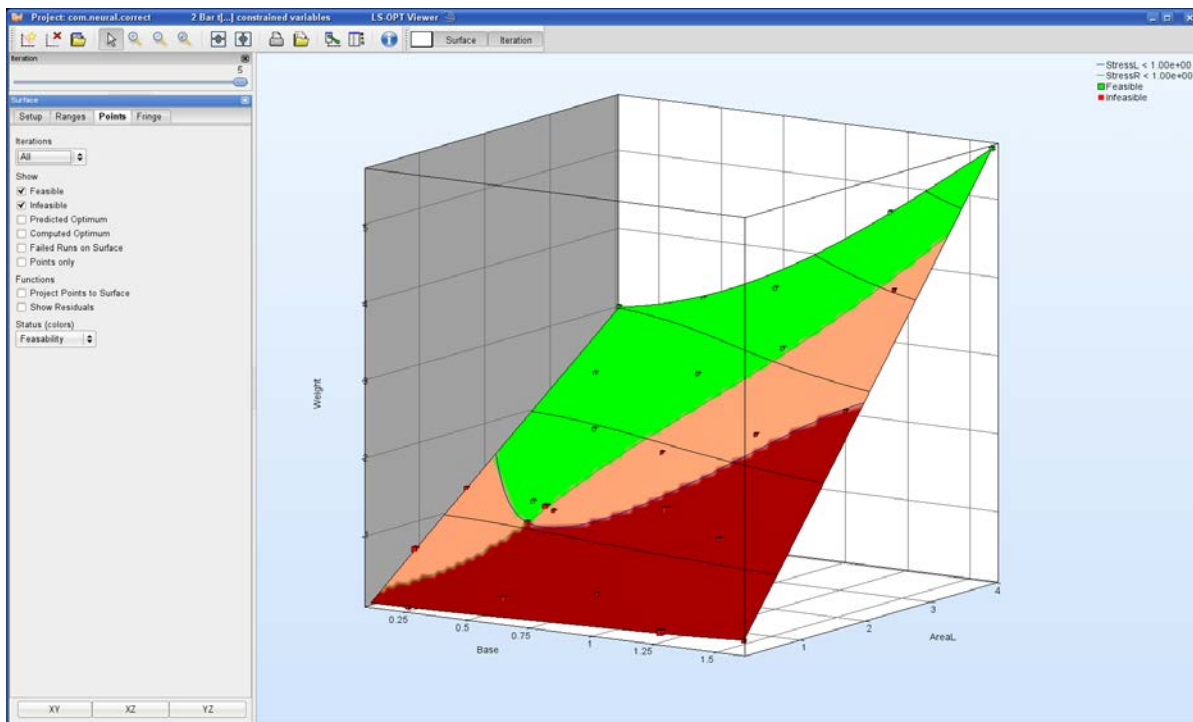


Figure 14-26: Surface plot representing only the region of interest of the fourth iteration.



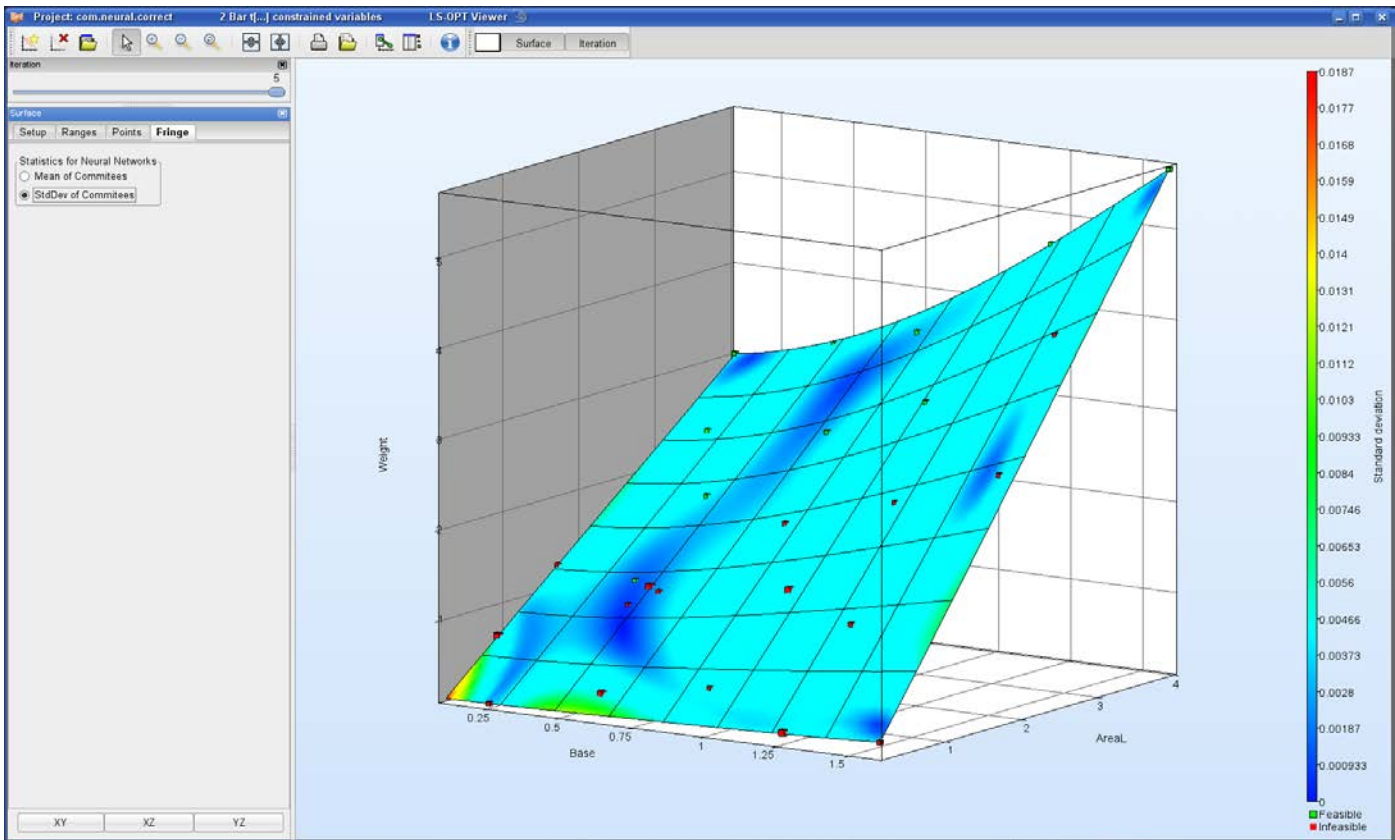
**Figure 14-27:** Plot showing isolines on the objective function as well as constraint contours and feasibility. Feasible regions are in green. Shade of red shows degree of infeasibility (number of violated constraints). Note the legend describing constraints at the top right.



**Figure 14-28:** 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. See Figure 14-29.



*Figure 14-29: Metamodel plot showing standard deviation of the Neural Net committee values.*

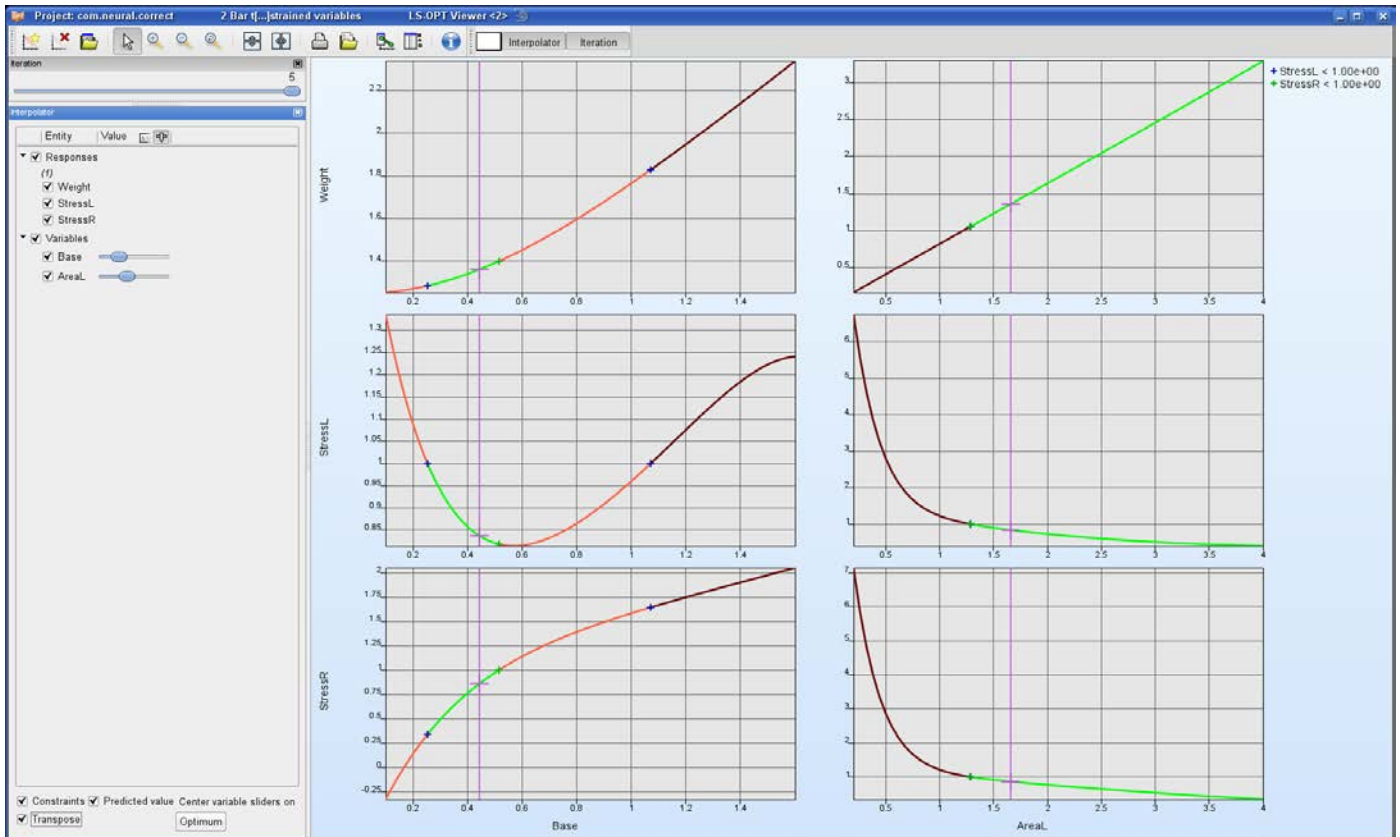
### 14.3.2. 2D Interpolator Plot

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 14-14: 2D Interpolator Plot options**

Selection	Description
<i>Constraints</i>	<p>Constraints are displayed on the surface.</p> <p>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.</p>
<i>Predicted value</i>	<p>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</p>
<i>Transpose</i>	<p>Allows to display each response against all variables in a column.</p>
<i>Center variable sliders on Optimum</i>	<p>Variable sliders are set to optimal values of selected iteration</p>



*Figure 14-30: 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)*

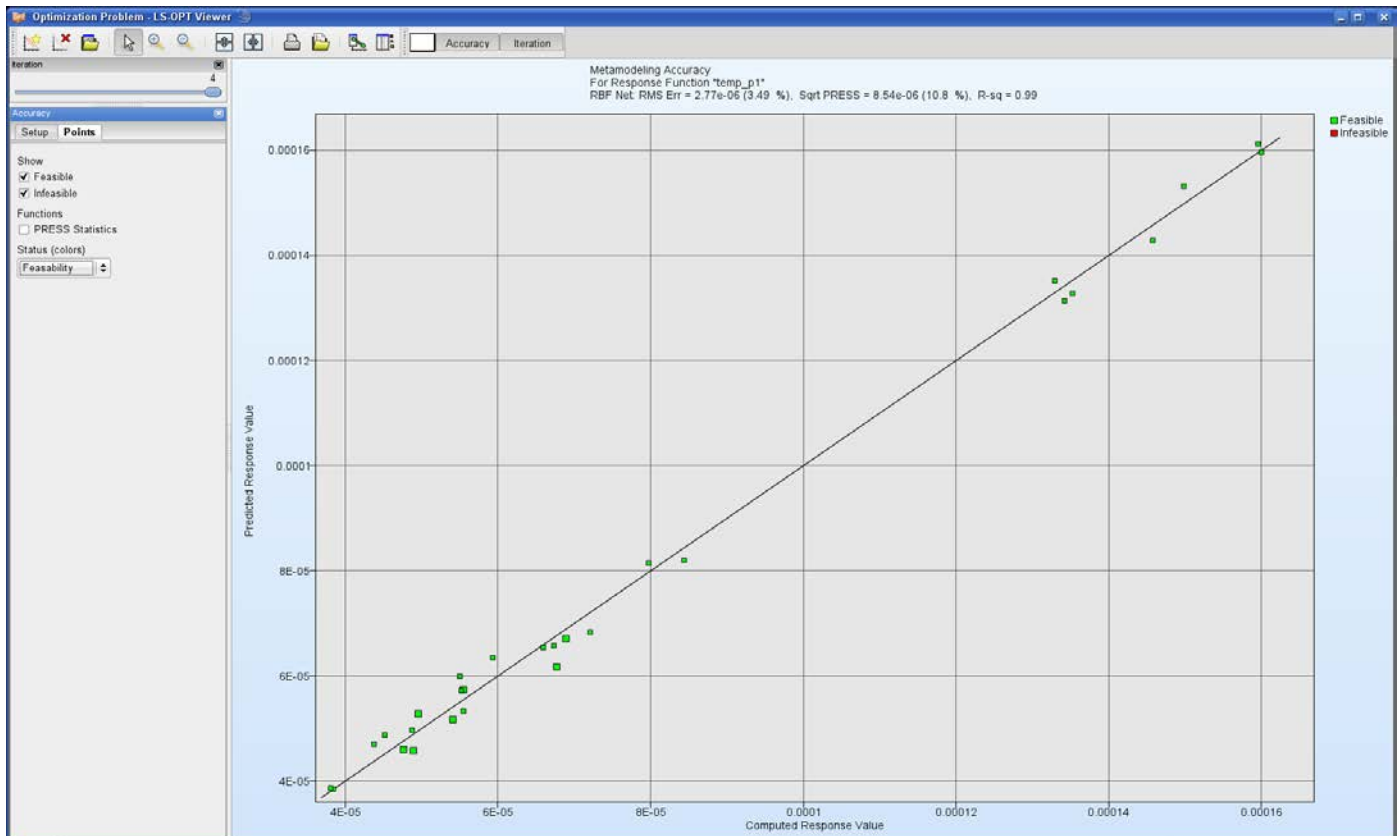
### **14.3.3. Accuracy Plot**

The accuracy of the metamodel fit for the selected response or composite is illustrated in a Predicted vs. Computed plot, Figure 14-31. 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 14-31. The error measures are displayed in the heading.

## Options

**Table 14-15: Accuracy Plot options**

Selection	Description
<i>Feasible</i>	Plot feasible runs
<i>Infeasible</i>	Plot infeasible runs
<i>PRESS statistics</i>	PRESS residuals are plotted against computed values
<i>Status (colors)</i>	Coloring options for points, see 14.2.2



**Figure 14-31: Computed vs. Predicted plot in View panel in LS-OPTui. The points are color-coded to represent the feasibility. The largest points represent the most recent iteration.**

### 14.3.4. 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 20.4) of the approximation to the experimental design is automatically performed if a polynomial response surface method is selected. 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 23.2 (theory) and 17.5 (example)).

If a probabilistic or an RBDO analysis is being done, then the Stochastic Contribution plots (see Section 14.6.3) are recommended.

The ANOVA results are viewed in bar/tornado chart format, Figure 14-32. The **Sort** option sorts the ANOVA values by relevance, the sorting doesn’t consider the 90% confidence interval.

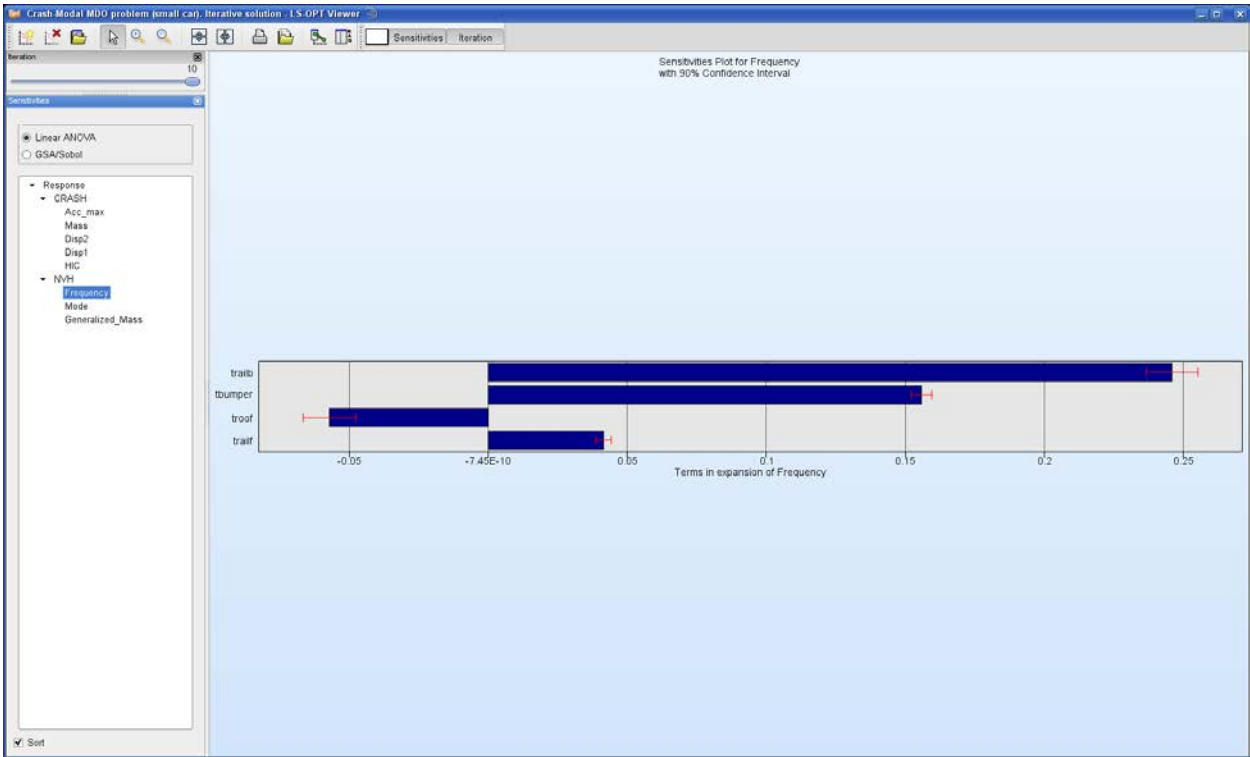


Figure 14-32: Linear ANOVA plot in View panel in LS-OPTui, sorted

### GSA/Sobol

A global sensitivity analysis is only performed if **Compute global Sensitivities** is selected in the Sampling panel of LS-OPTui, see Section 4.10.

Figure 14-33 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.

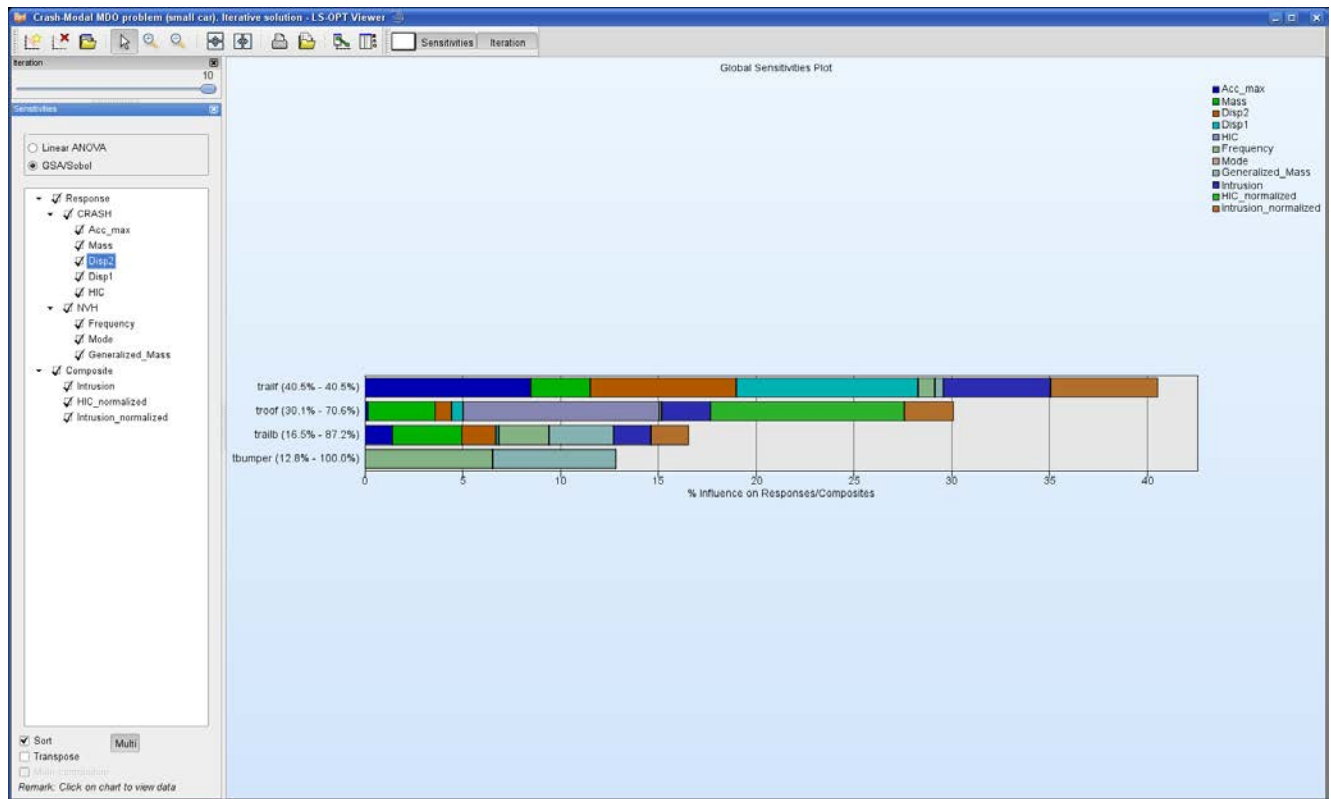


Clicking on the chart displays the respective sensitivity values and variances in the plot.

## Options

**Table 14-16: GSA/Sobol Plot options**

Selection	Description
<i>Sort</i>	Sorts data by relevance
<i>Transpose</i>	Sensitivity values are grouped by response/composite
<i>Main contribution</i>	Main contribution is displayed in addition to total contribution
<i>Multi</i>	Allows selection of multiple responses/composites



**Figure 14-33: Sorted global sensitivities of all responses and composites**

### 14.3.5. History Plot

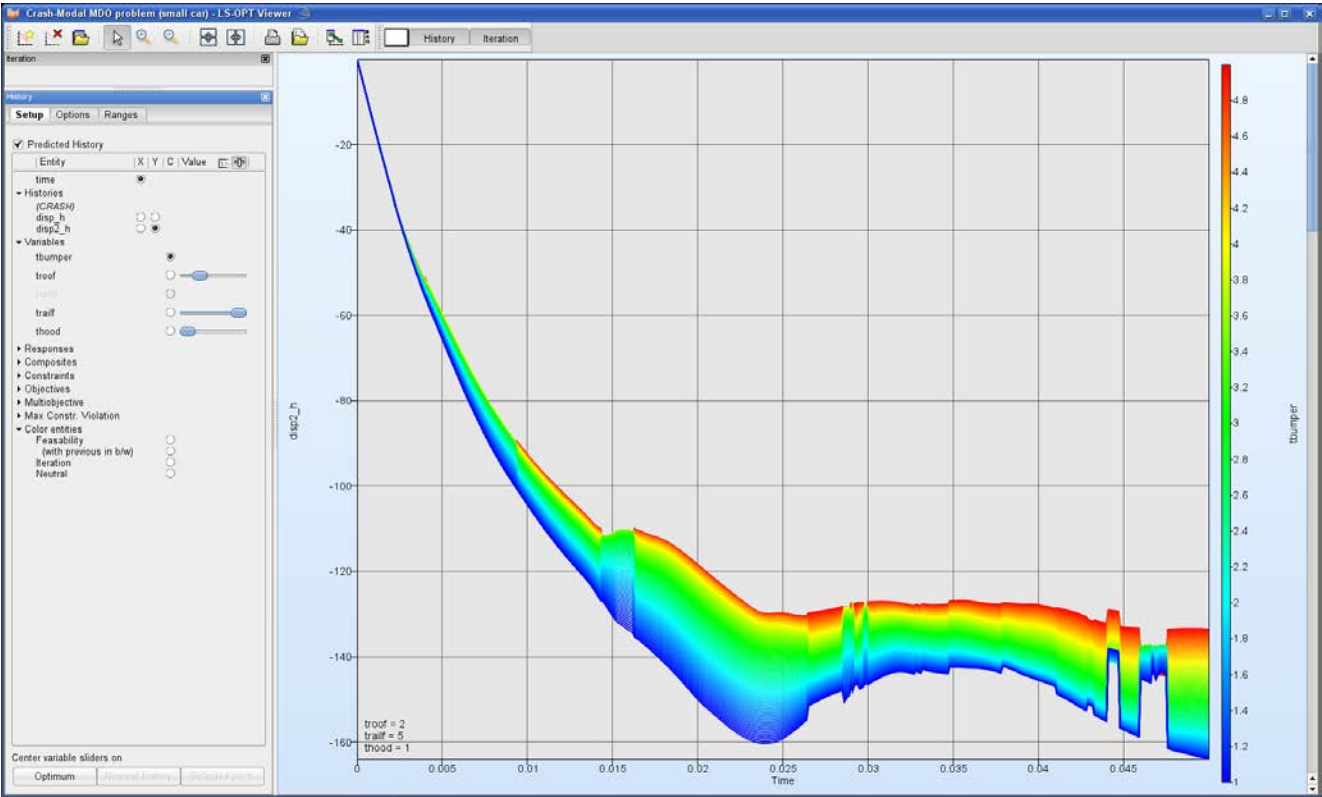
If the **Approximate History** option is set in the Sampling Panel of LS-OPTui, a database that approximates the histories for any design point using metamodels is provided, see Section 8.5.1. If **Predicted History** is selected, the history evaluated on the metamodel for the selected design point is visualized, Figure 14-35. 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 14-18.

The **center variable sliders on** options set the variable sliders to specific values.

**Table 14-17: History Plot center variable sliders on - options**

Selection – center variable sliders 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



**Figure 14-34: Predicted Histories colored by variable**

## Options

**Table 14-18: Predicted History options**

Selection	Description
<i>Nearest</i>	Show computed history with design point closest to selected design point
<i>Maximal Residual</i>	Plot maximal residual
<i>Deviation Residual</i>	Plot standard deviation of residuals
<i>Number of predicted curves</i>	Number of plotted curves if histories are colored by variable



**Figure 14-35: Predicted History with nearest history and maximal residual**

## 14.4. Visualization of Optimization Results

### 14.4.1. 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 14-36. For all the dependents, responses, objectives, constraints and maximum violation, a black solid line indicates the predicted values, while the red squares represent the computed values at the starting point of each iteration, Figure 14-37. 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.

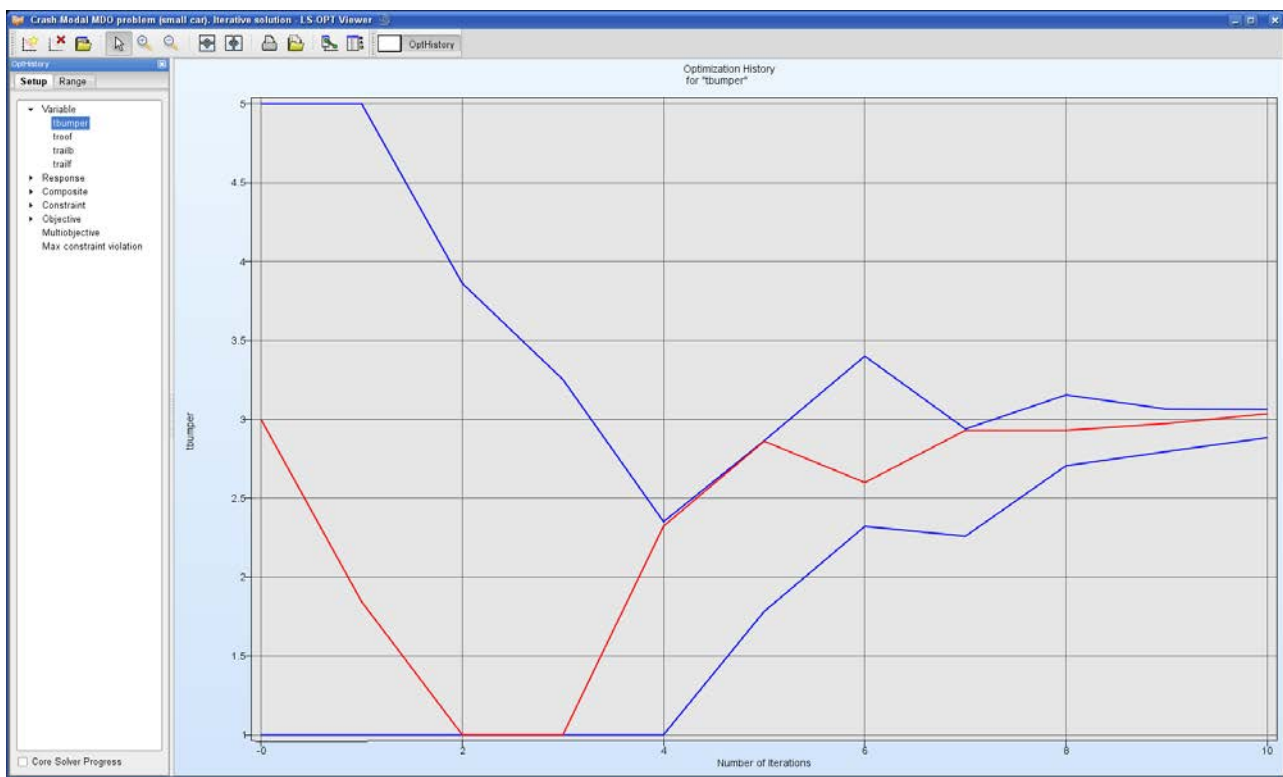
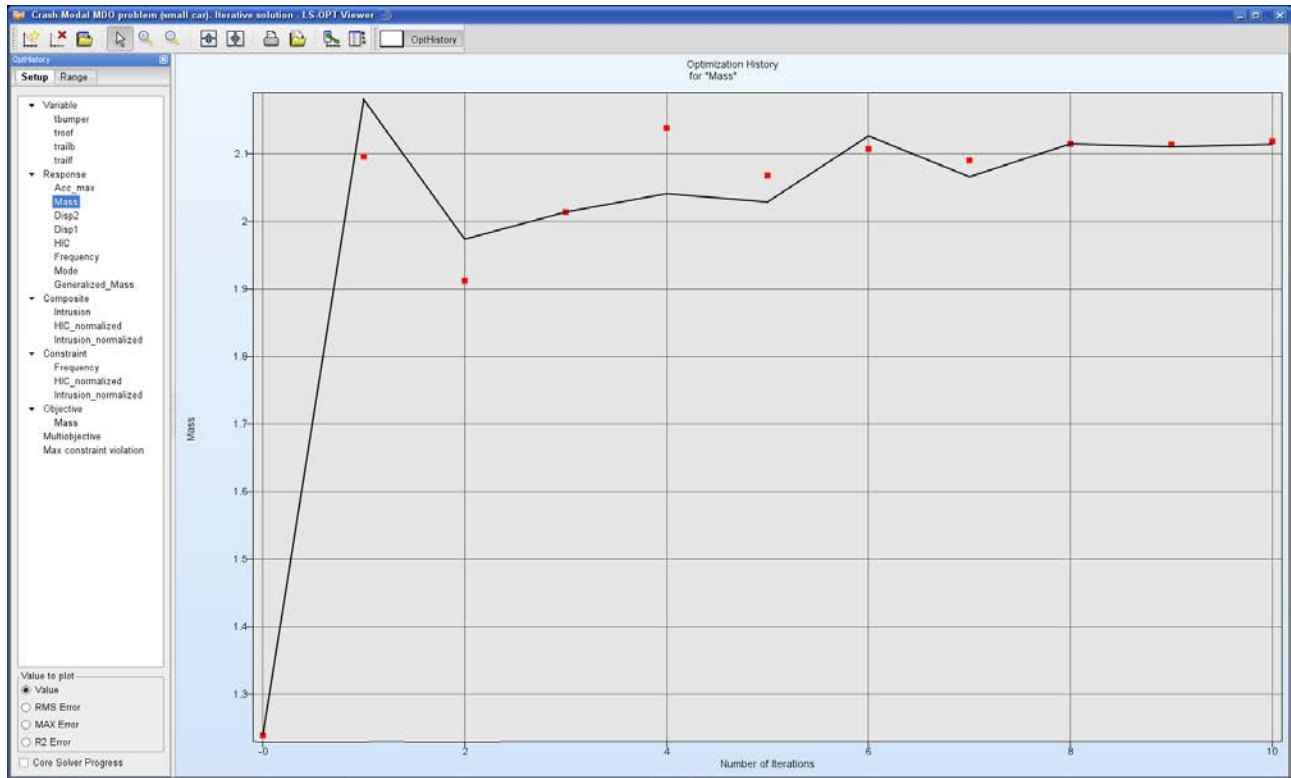


Figure 14-36: Optimization History plot of a variable – variable values (red) and subregions (blue)



*Figure 14-37: Optimization History of a response – computed (red points) and predicted (black) values*

## 14.4.2. 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 14-38. Clicking on the charts displays the actual value and the bounds on the plot.

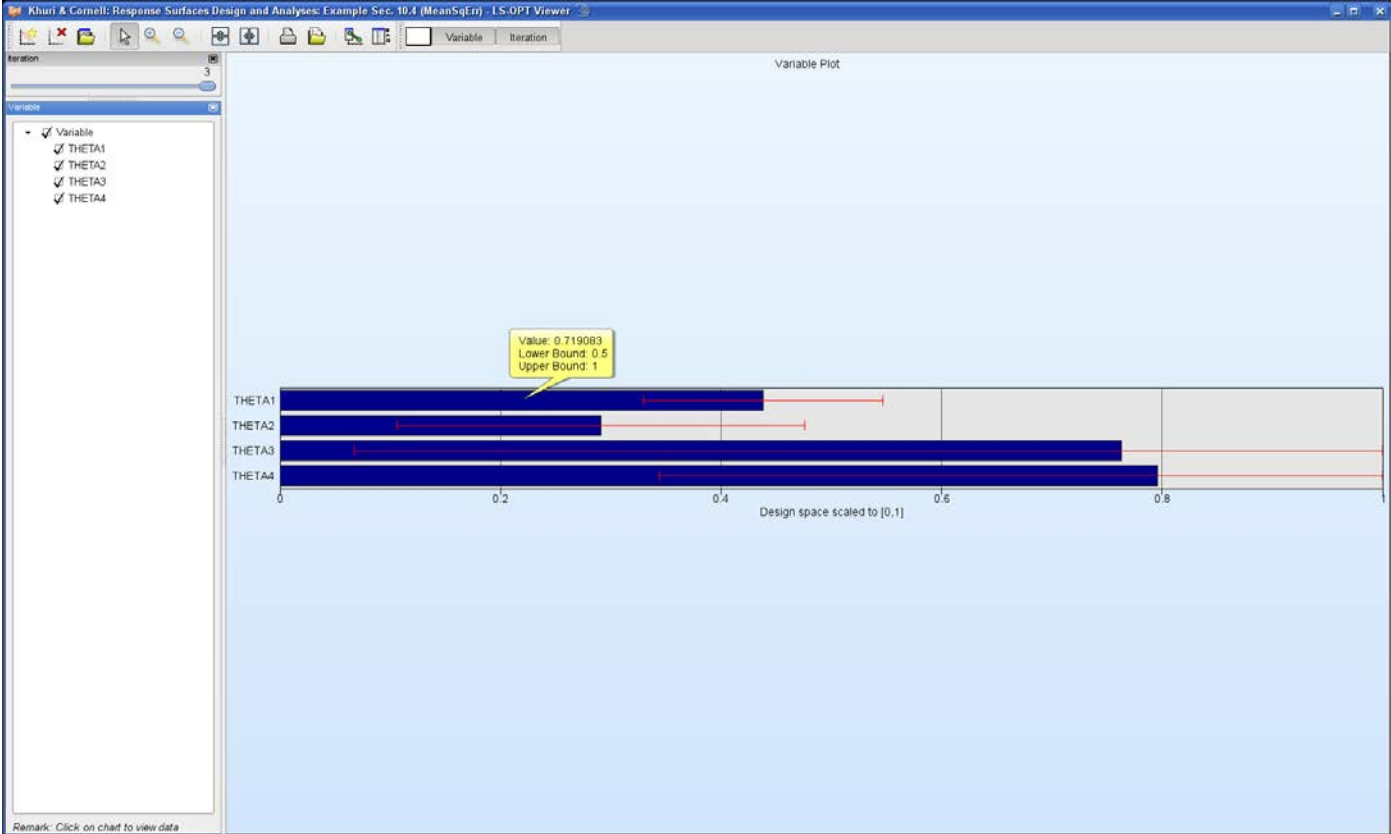
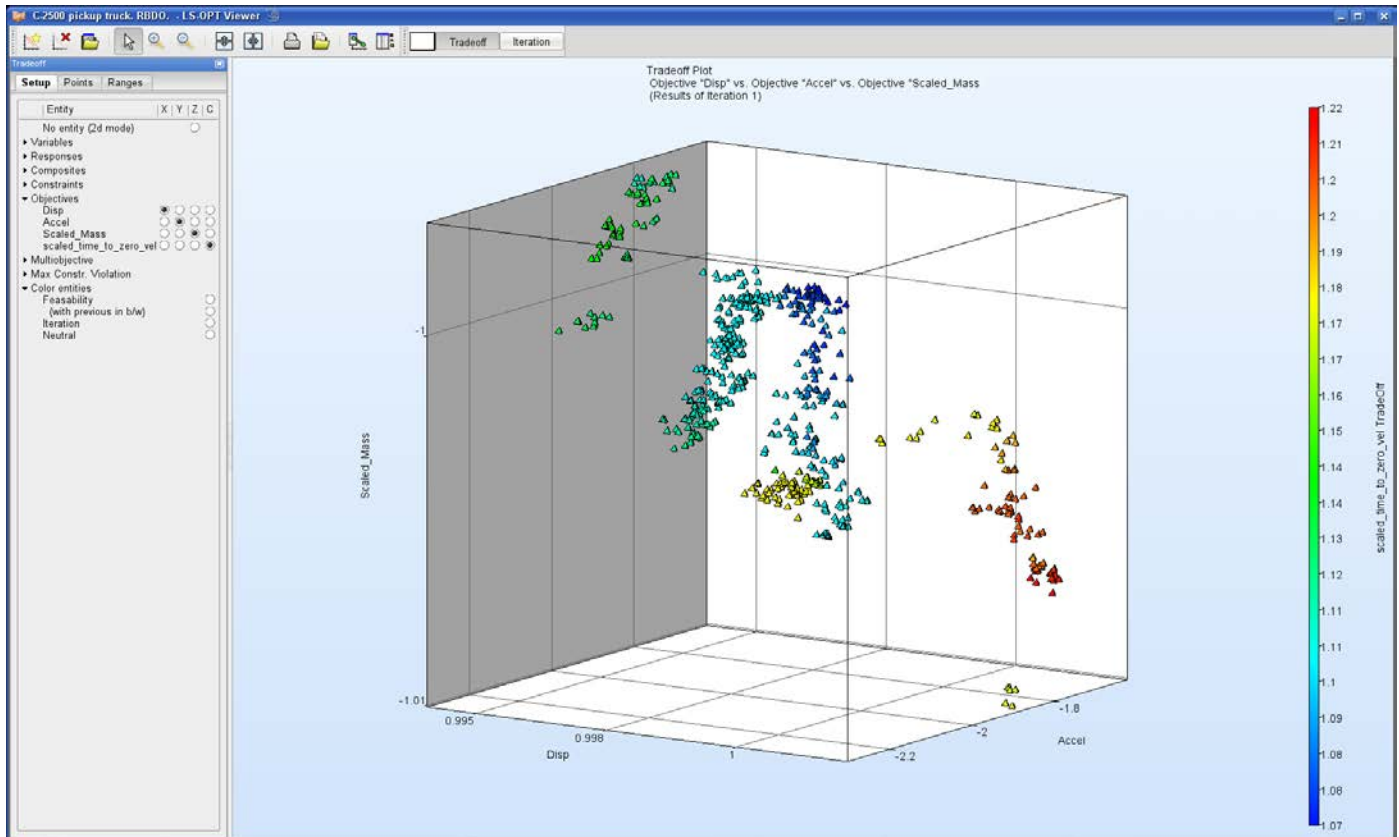


Figure 14-38: Variable Plot

## 14.5. Visualization of Pareto Optimal Solutions

### 14.5.1. Tradeoff Plot

The Tradeoff plot (Section 22.12.1) functions similar to the Scatter plot, Section 14.2.2, but the default setting is here to plot Pareto optimal solution data instead of Analysis Result data.



*Figure 14-39: Tradeoff plot*

## 14.5.2. Parallel Coordinate Plot

The Parallel Coordinate Plot (Section 22.12.3) in the Pareto optimal solutions category functions similar to the Parallel Coordinate Plot described in Section 14.2.3, but here, the default setting is to visualize Pareto data.

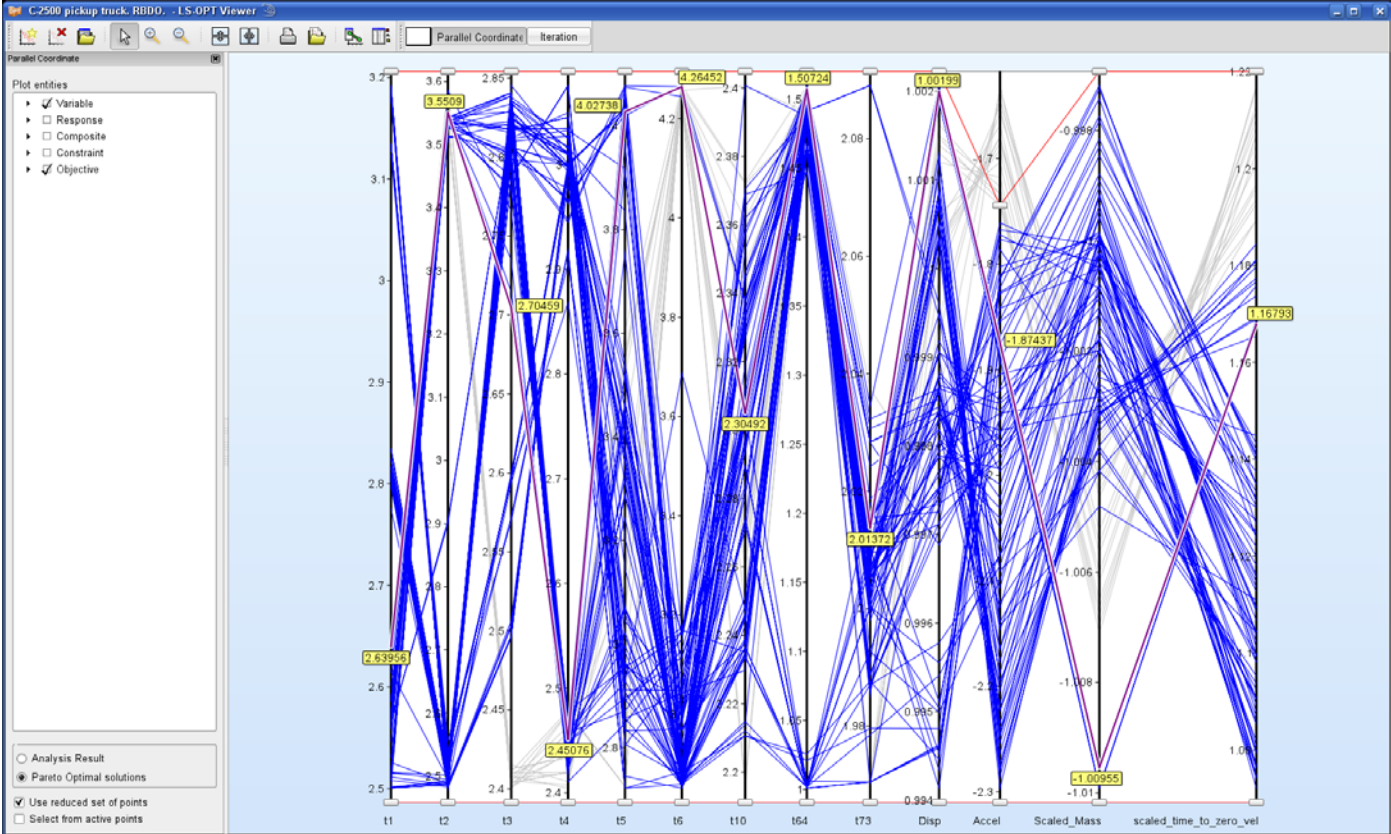


Figure 14-40: Parallel Coordinate Plot for Pareto optimal solutions with selected point (purple line)

### 14.5.3. 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 14-41.




The theory of hyper-radial visualization is explained in Section 22.12.2.

### Grouping

The objectives may be grouped using the 3-state buttons in the **Axis** column.



**Table 14-19: Hyper-Radial Visualization Grouping options**

Selection	Description
	Add objective to the group displayed on the x axis
	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 14-20: 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

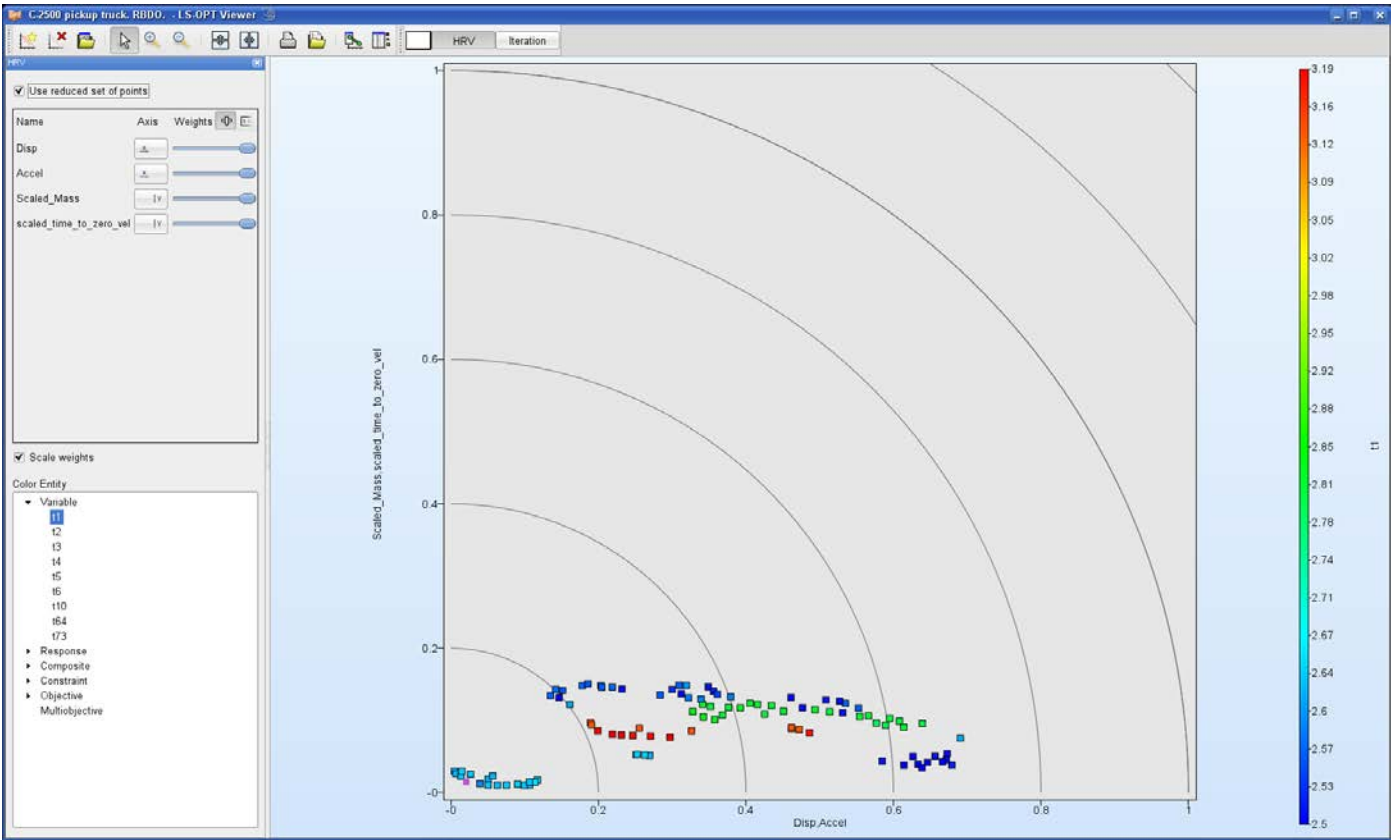


Figure 14-41: Hyper-radial visualization, equal weights, points colored by variable

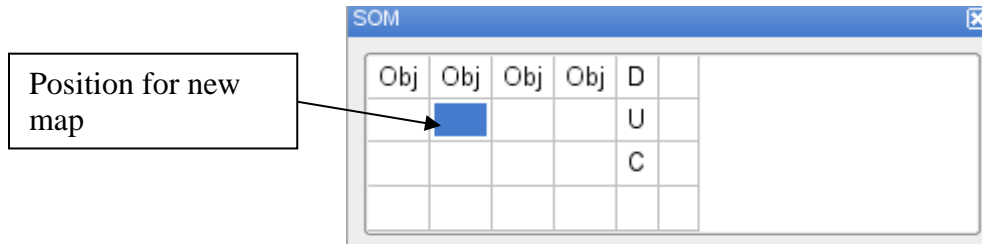
### 14.5.4. Self-Organizing Maps

The theory of Self-Organizing Maps (SOM) is explained in Section 22.12.4

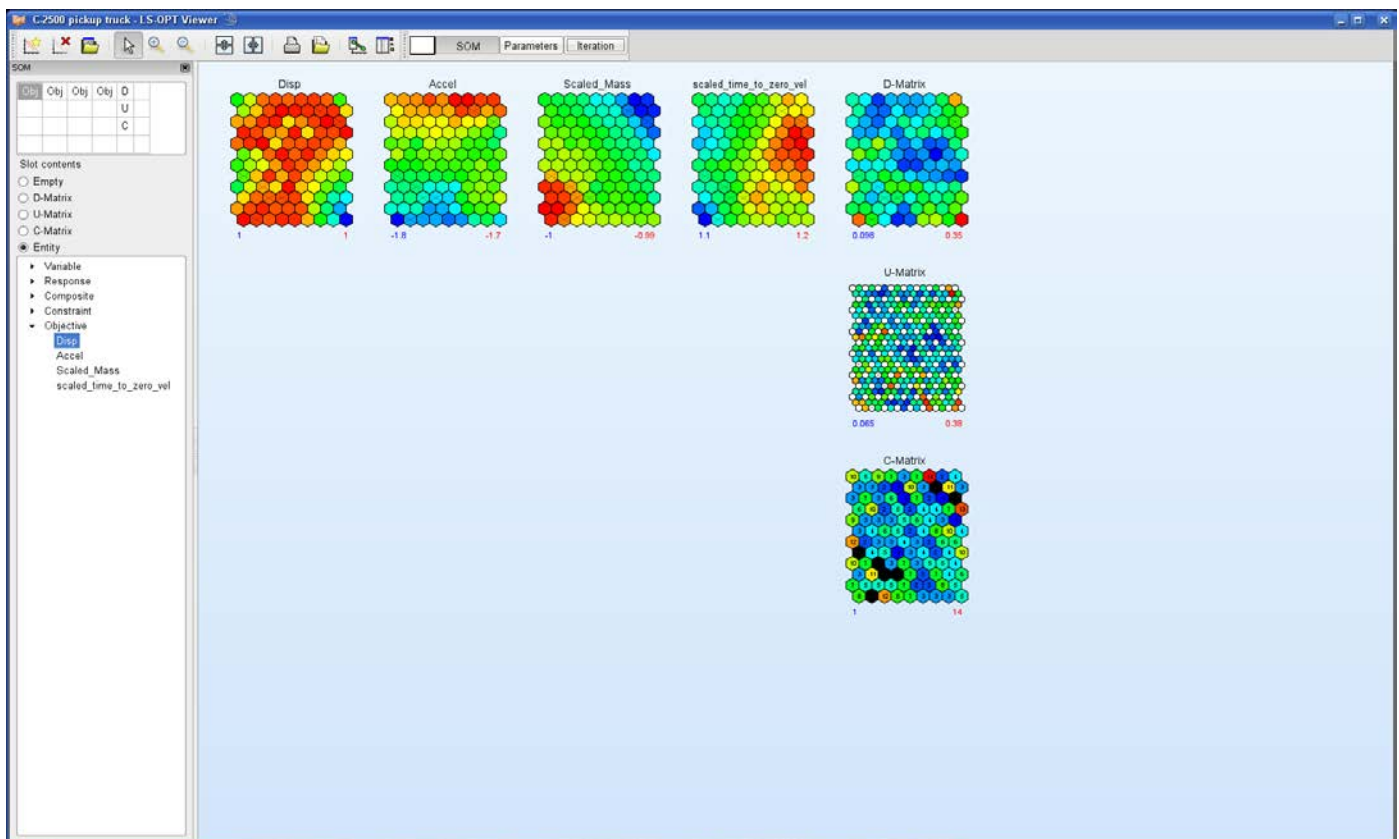
### Component Selection

By default, component maps of all objectives as well as the D-, U- and C-matrices are displayed. To modify the plot, select the position in the dynamic grid, Figure 14-42, and slot content, see Section 22.12.4.

Describe the cell selection behaviour. D-, U-, C- maps. Refer to Section 22.12.4 for an explanation of the map types.



*Figure 14-42: Selection of position for SOM*



*Figure 14-43: Self-Organizing Map, component plots of objectives and distance measure*

## 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 14-21: Self-Organizing Maps parameters**

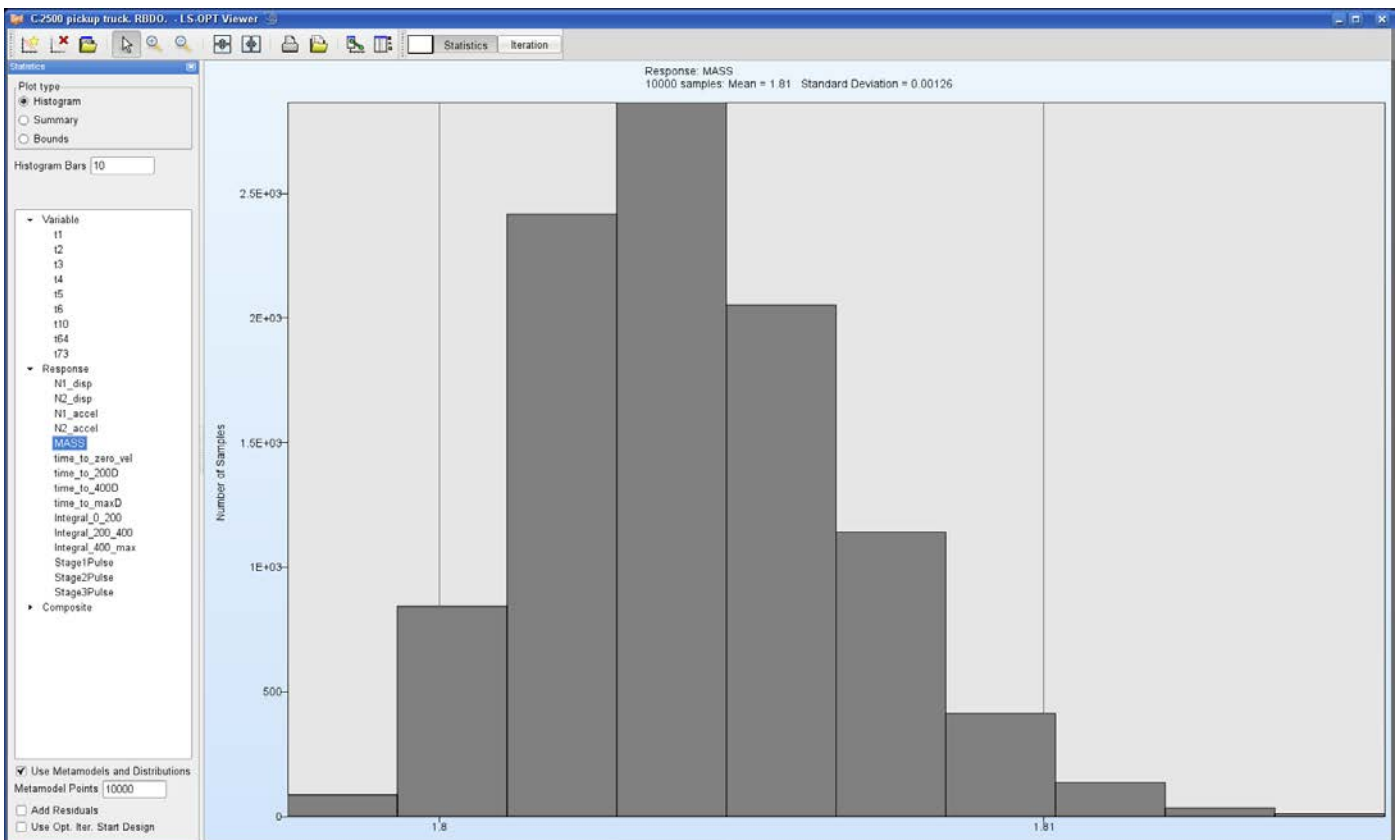
Selection	Description
<i>Training Iterations</i>	Number of iterations performed for training of SOM, default depends on honeycomb dimensions and number of data points
<i>Initial Radius</i>	Initial radius used for training of SOM, default depends on honeycomb dimensions
<i>Honeycomb dimensions</i>	Honeycomb dimensions, default 12x9

## 14.6. Stochastic Analysis

### 14.6.1. Statistics

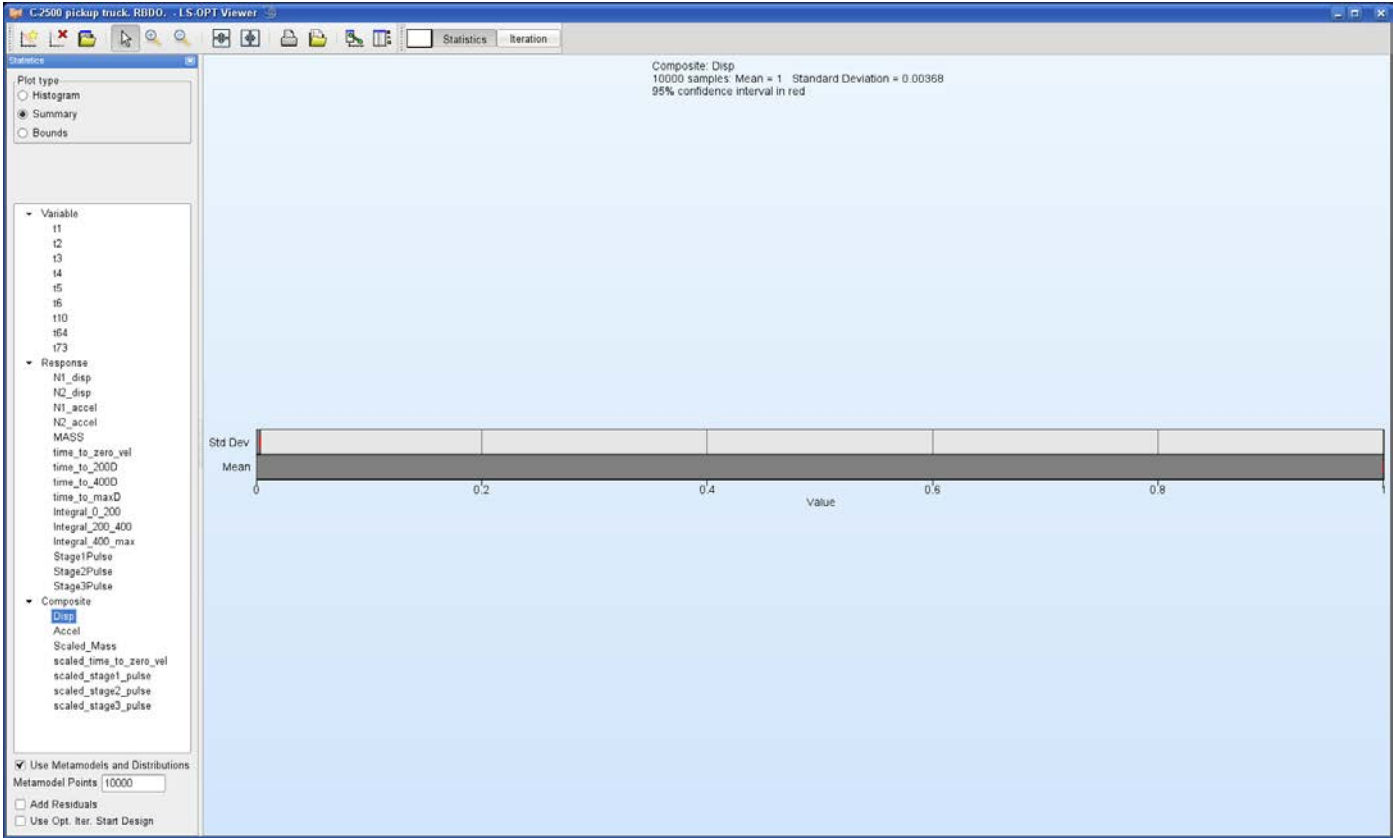
See Section 14.2.5.

### Histogram



**Figure 14-44: Histogram constructed using metamodel together with the statistical distribution of the variables**

### Summary



*Figure 14-45: Mean value and standard deviation constructed using metamodel together with the statistical distribution of the variables*

# Bounds

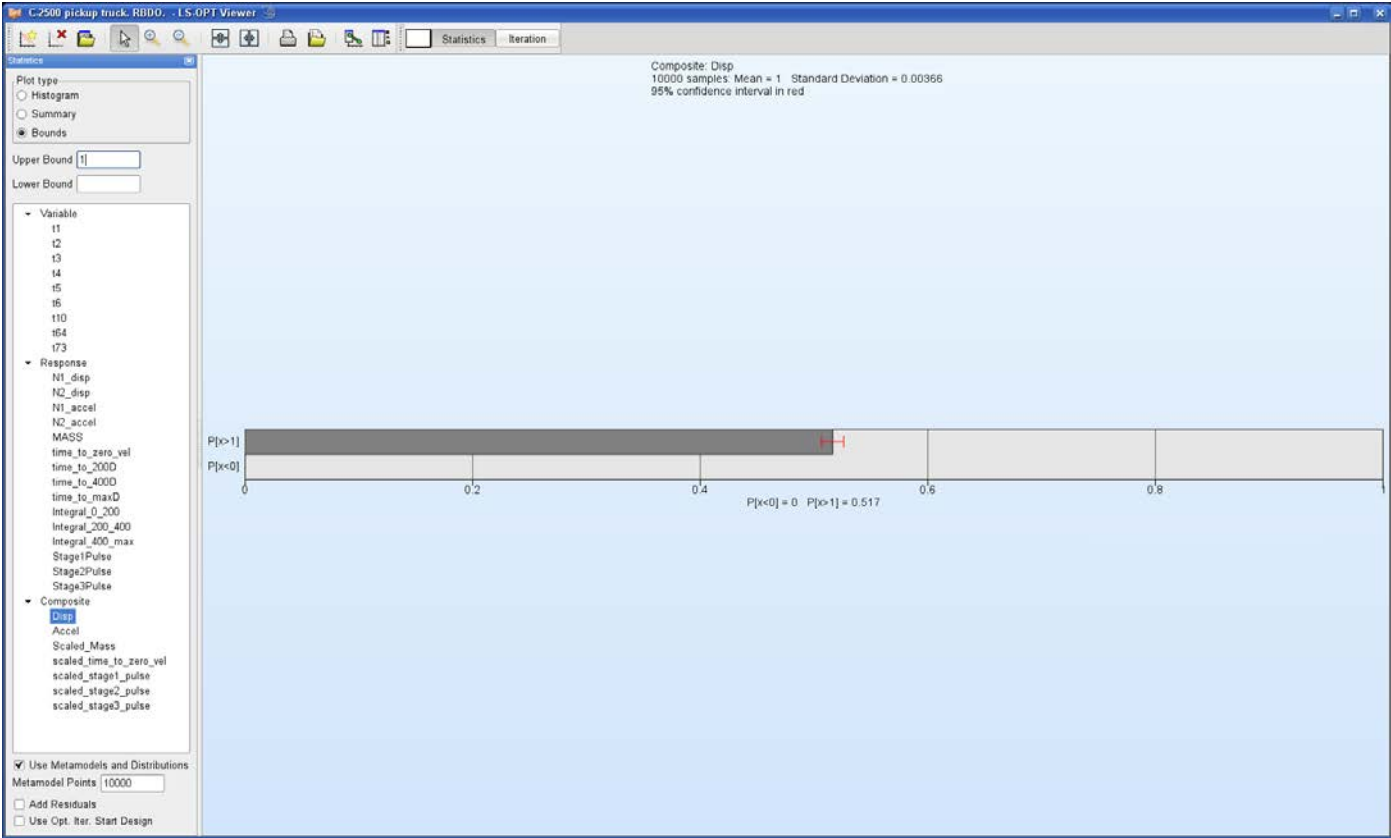


Figure 14-46: Probability of Composite Disp > 1 with 95% confidence interval in red constructed using metamodel together with the statistical distribution of the variables

## 14.6.2. Correlation Bars

See Section 14.2.6.

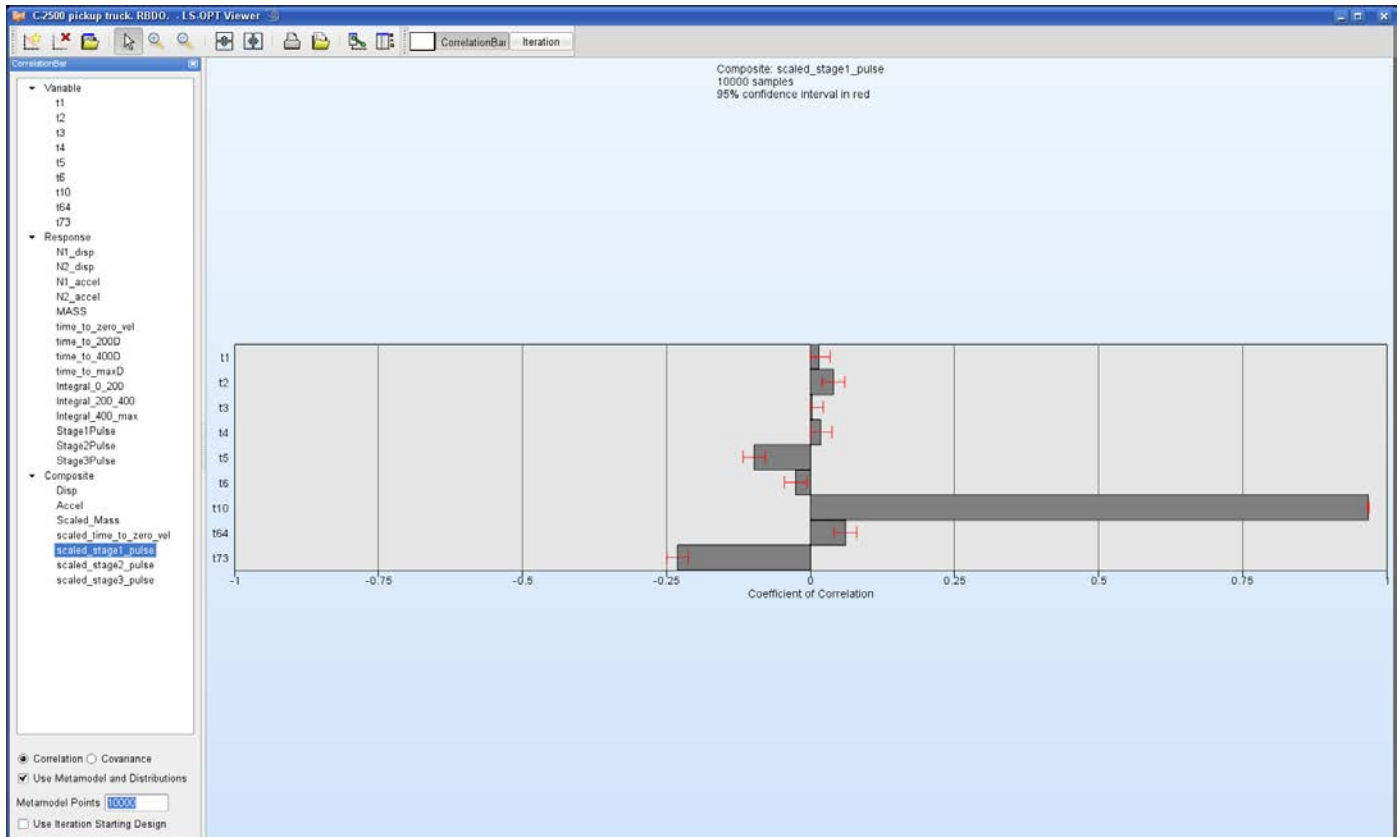


Figure 14-47: Correlation Bars evaluated on metamodel

### 14.6.3. Stochastic Contribution

The stochastic contribution of the variables to the variance of the responses and composites (see Section 24.7) can be displayed as a bar chart.

Optionally the user can elect 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 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 24.7 for the details.

For optimization the stochastic contribution is computed using the optimal design.

The stochastic contribution panel is shown in Figure 14-48.

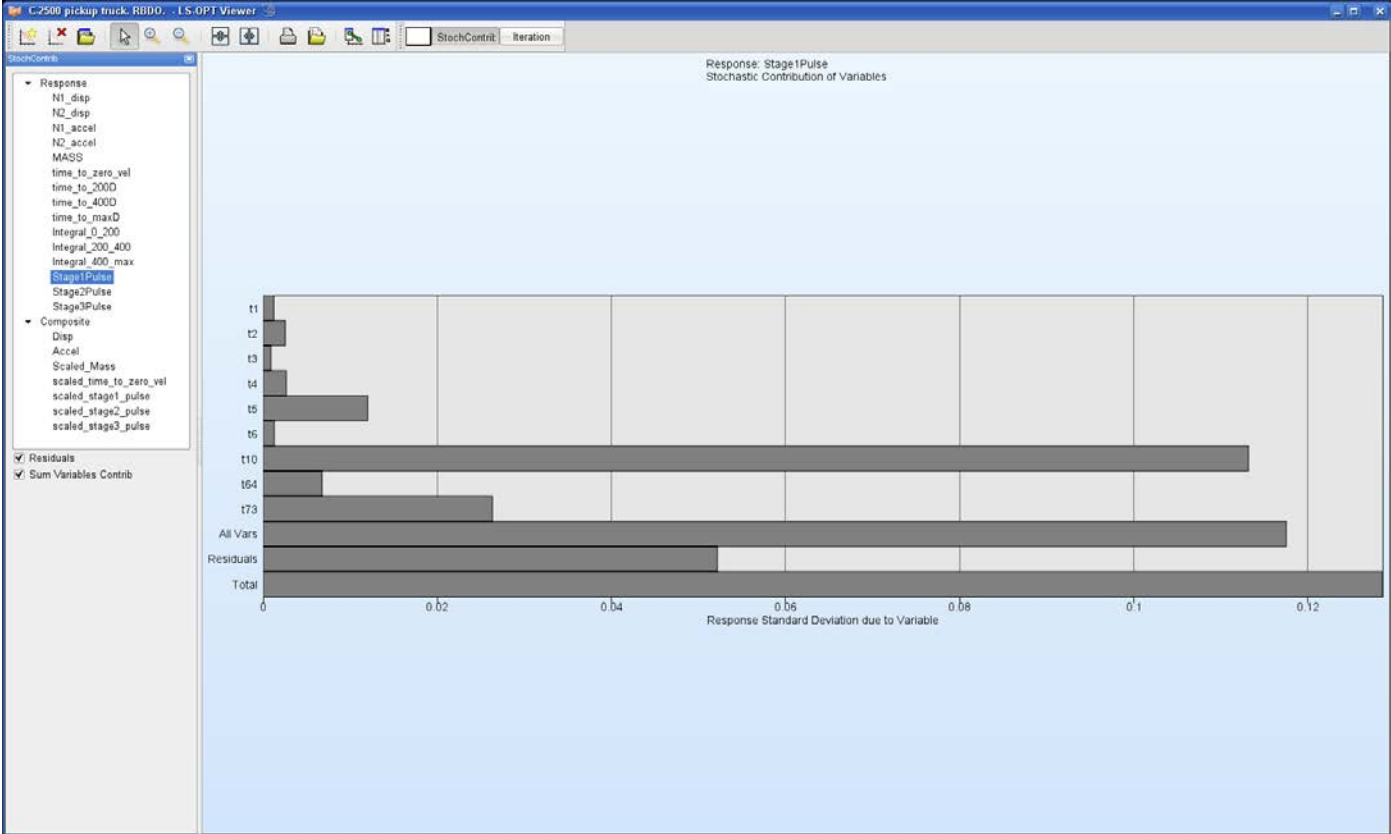


Figure 14-48: Stochastic Contribution plot

### 14.7. References

[1] Myers, R.H. and Montgomery, D.C. *Response Surface Methodology. Process and Product Optimization using Designed Experiments.* Wiley, 1995



# 15. 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” panel 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 24.6 regarding the computation of outliers. This is the process variation is 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 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.

## 15.1. 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 15-1 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 the following sections.
- *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.

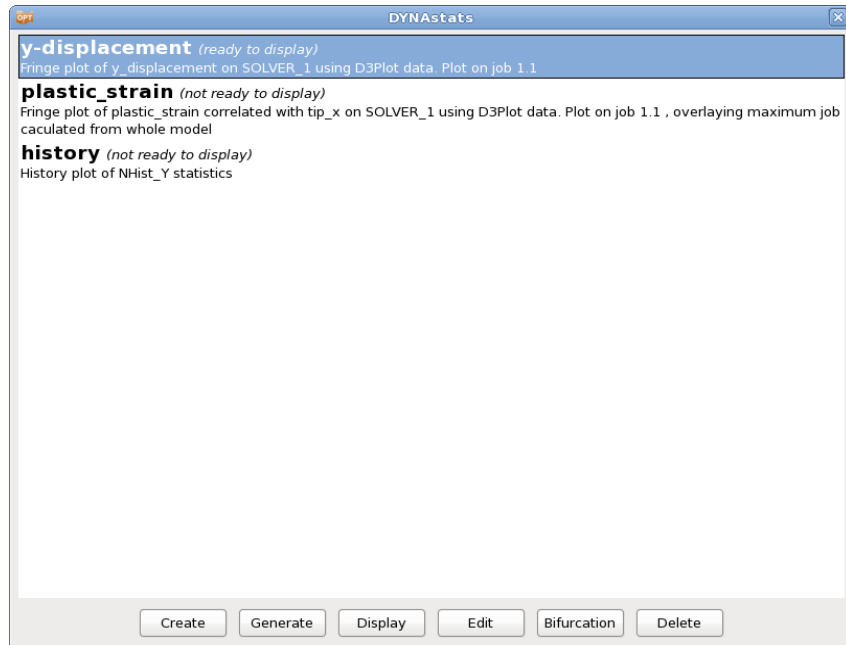
The plot definitions are stored in a file which allows re-use of a methodology in different studies (see Section 15.11).

## 15.2. Creation of a plot

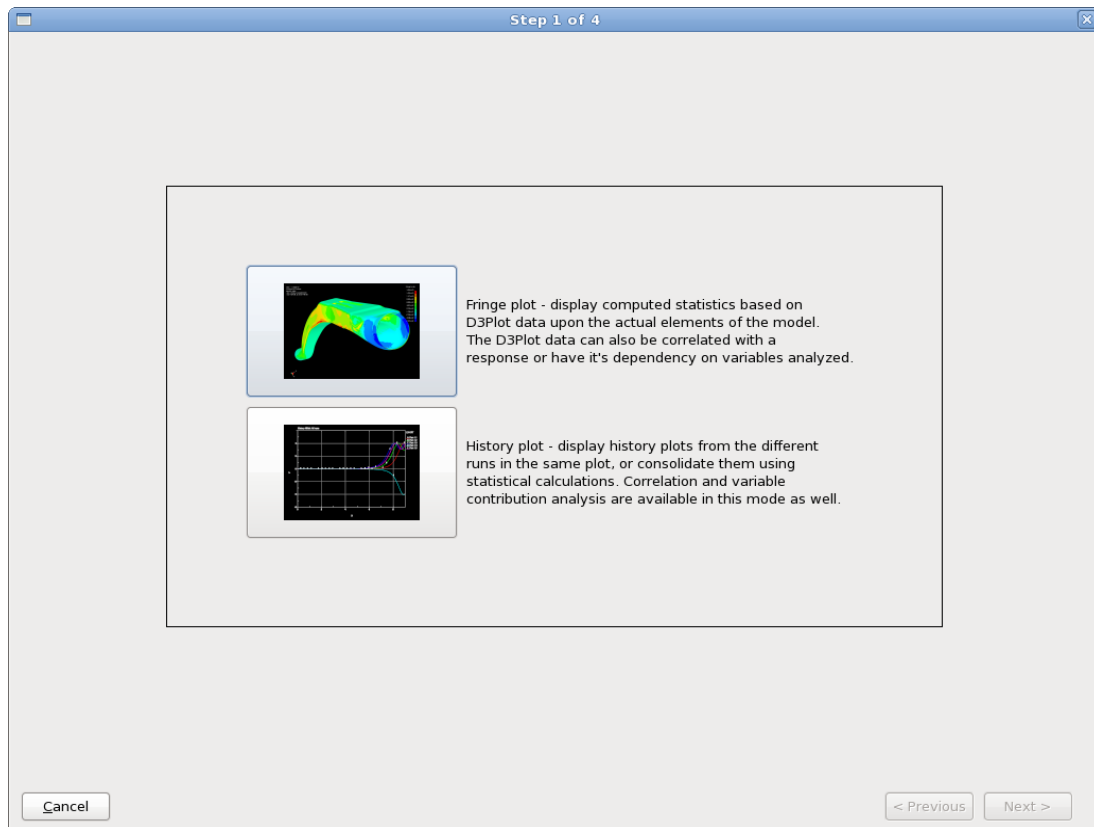
A plot is created in four steps.

### 15.2.1. 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 15-2. Select the respective image to go to the next step.

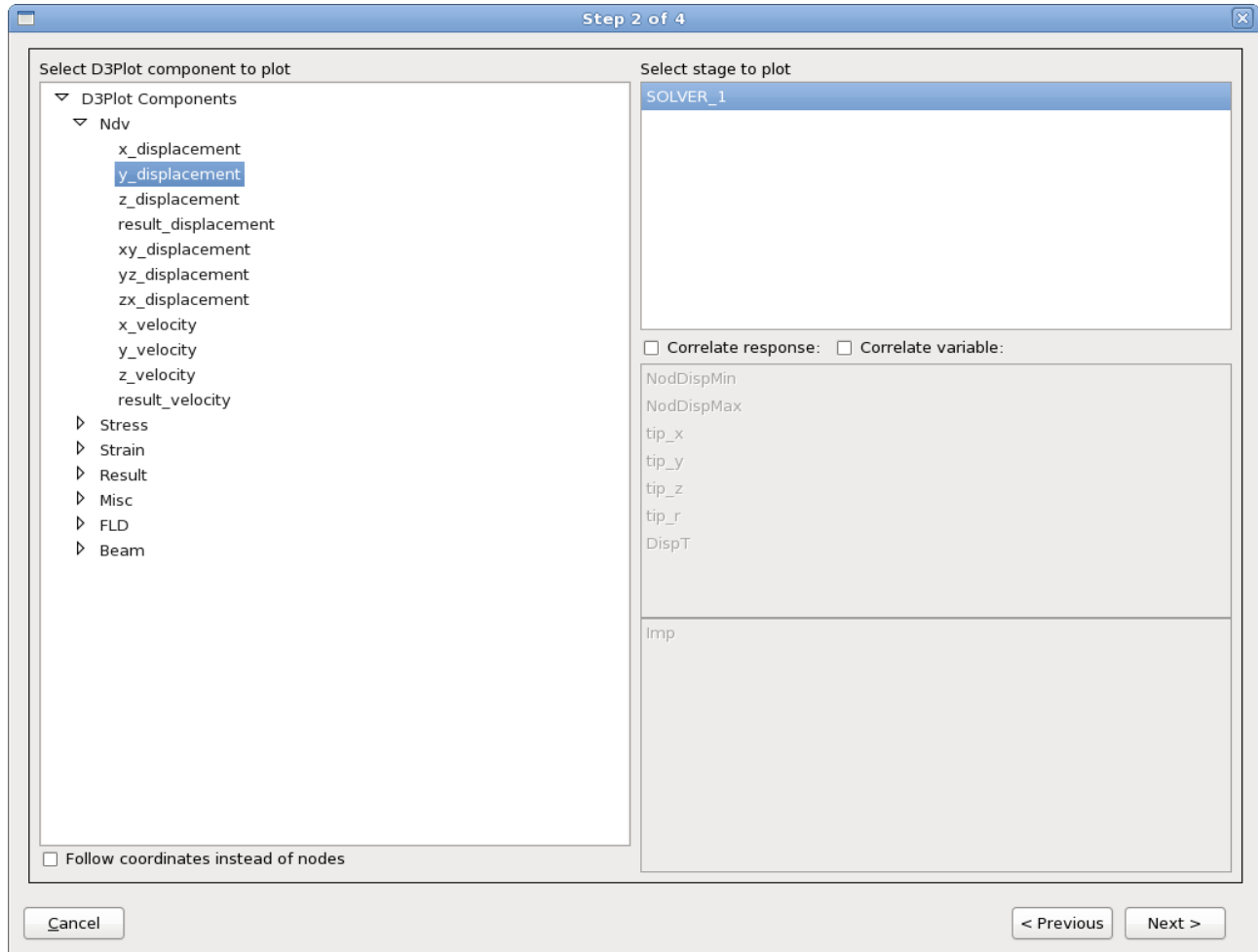


**Figure 15-1: Visualization of DYNA results statistics. After plot creation using the wizard, the plot data must be generated by running LS-OPT. The plot can then be displayed in LS-PREPOST. Existing plots can be edited, deleted or investigated for a bifurcation.**



**Figure 15-2: First step of DynaStats plot definition creation; selection of plot type**

### 15.2.2. Step 2 – D3Plot component or History

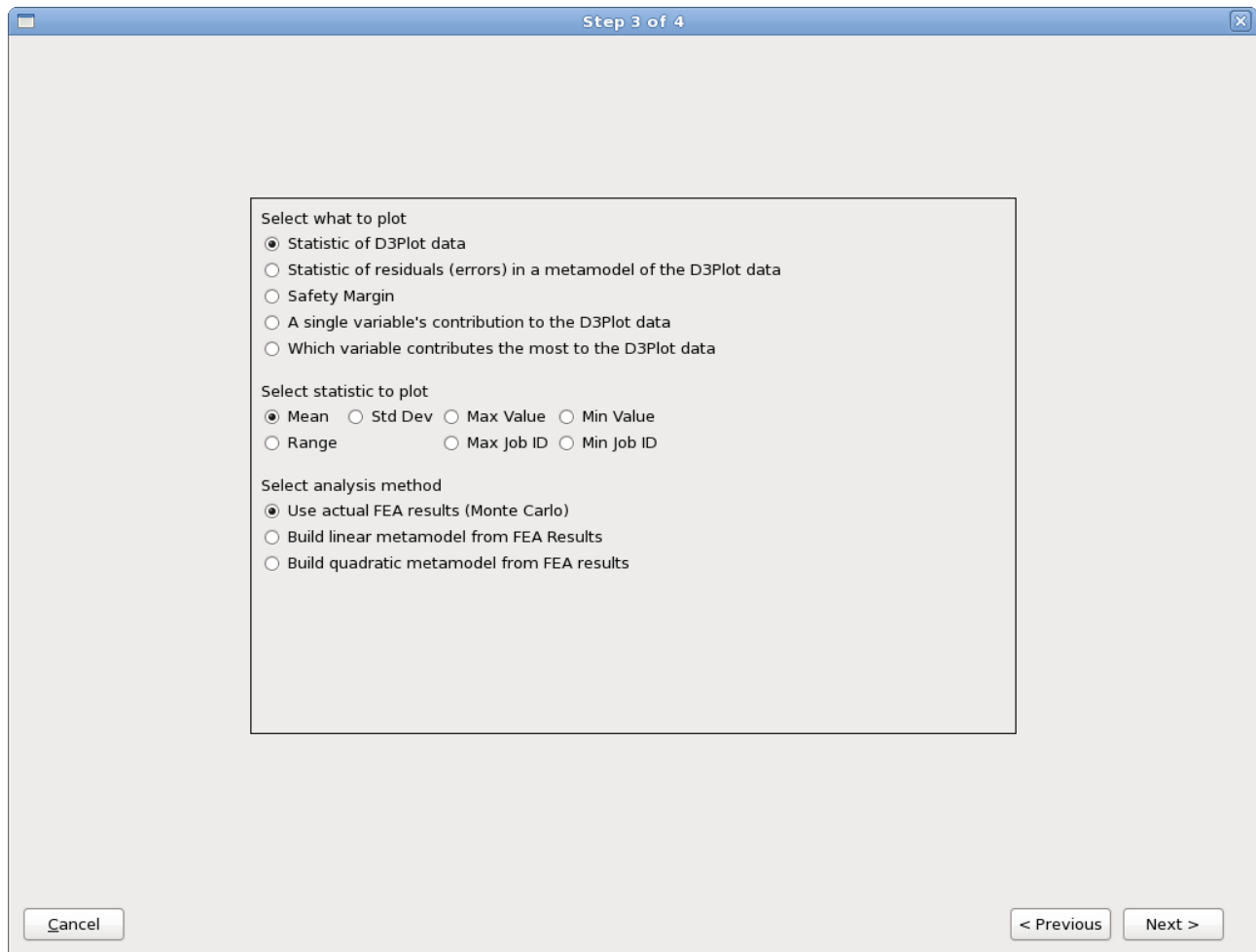


*Figure 15-3: Second step of DynaStats plot creation; selection of component or history*

**Table 15-1: 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 15.7
Select stage to plot	Name of stage	
Follow coordinates instead of nodes	The ID of the part to be mapped has to be specified	Section 15.10
FLC curve	FLC curve specification (for FLD components) Parameteric FLD curve $t$ and $n$ coefficients Provided curve Curve ID in the LS-DYNA file of the FLD curve to be used	Section 15.10
Correlate response	Correlation between an LS-OPT response and a D3Plot component at all states	Section 15.4.1
Correlate variable	Correlation between an LS-OPT variable and a D3Plot component at all states	Section 15.4.2

### 15.2.3. Step 3 - Statistics



*Figure 15-4: Third step of DynaStats plot definition creation*

**Table 15-2: 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 15.6
	A single variable's contribution to the D3Plot data	Section 15.5
	Which variable contributes the most to the D3Plot data	
Select statistics to plot		Section 15.3
Select analysis method	Use actual FEA results (Monte Carlo)	Section 15.3.1
	Build linear metamodel from FEA Results	Section 15.3.2
	Build quadratic metamodel from FEA results	Section 15.3.2

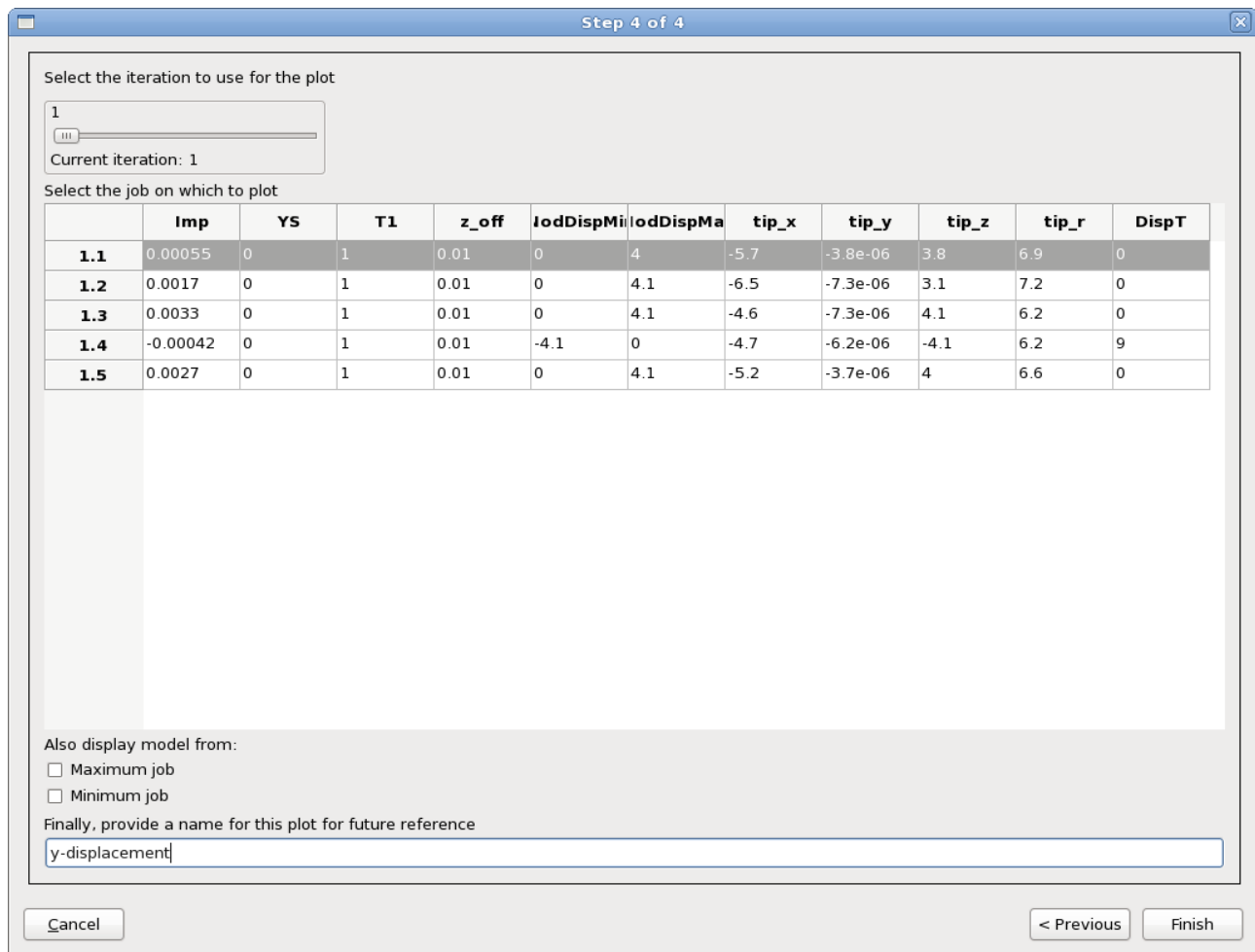
#### 15.2.4. Step 4 – Visualization in LS-PREPOST

The user can select the LS-PREPOST plot details in LS-OPT (Figure 15-5). The GUI options will reflect whether fringe component response or history data is being investigated.

The *Job Index* field specifies the FE model used for the display of the results. Additionally, the FE models containing the maximum and the minimum results can be overlaid in order to spot bifurcations as described in a later section.

**Table 15-3: 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		
Also display model from	Bifurcation investigations	Section 15.8
Name for this plot		



**Figure 15-5: 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 overlaid in order to identify bifurcations as described in Section 15.8.**



## 15.3. 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 in the stage directory with extensions of *.statdb* and *.history*.

The statistics are computed for a single stage and a single iteration.

### 15.3.1. Monte Carlo

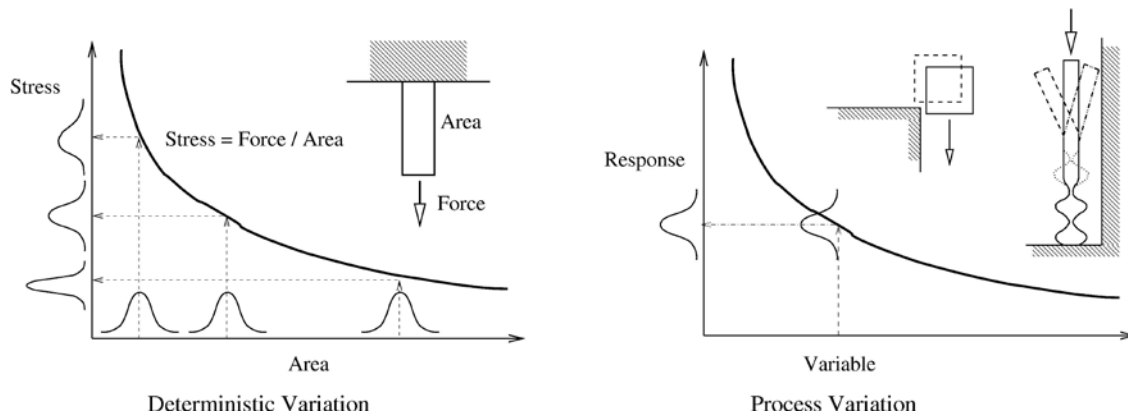
The statistics of the responses from a Monte Carlo procedure can be computed. The task will calculate:

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 indentify the jobs likely to contain a different bifurcation.
  - ID of the LS-DYNA job where the minimum value occurred. This can be used to indentify 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 15.6).

### 15.3.2. Metamodels and residuals

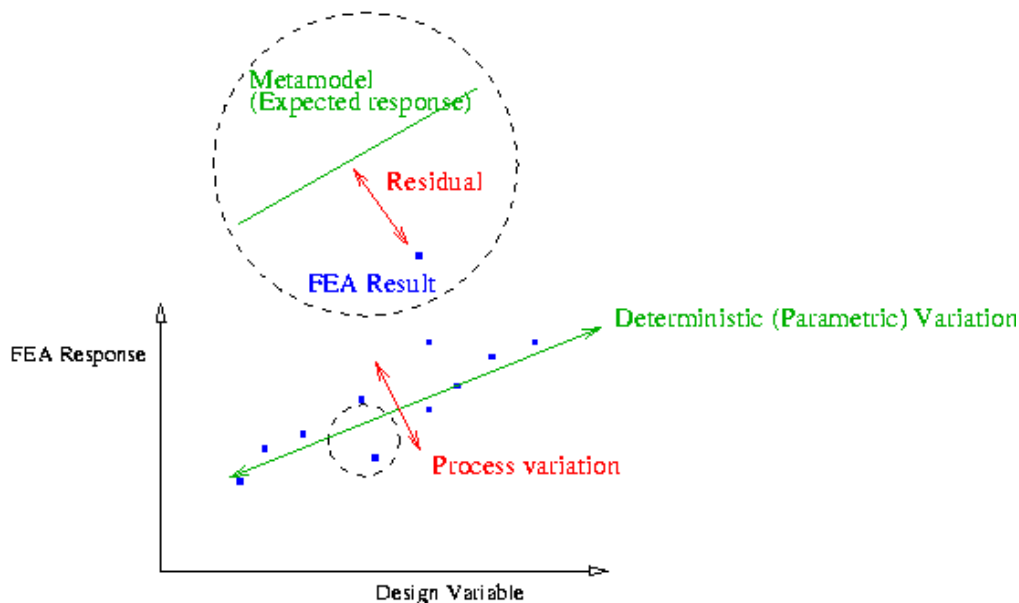
Metamodels (approximations) can be used to predict the statistics of the responses. These metamodels (approximations) will be computed for all results for all nodes for all time steps.

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 15-6.



**Figure 15-6: 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 15-7, a metamodel can only predict the effect of the design variables. Process variation, not predictable by the design variables, becomes residuals.



**Figure 15-7: 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 indentify the jobs likely to contain a different bifurcation.
    - ID of the LS-DYNA job where the minimum residual occurred. This can be used to indentify the jobs likely to contain a different bifurcation.
  3. Stochastic contribution of each individual variable
  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 15.6).
  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 24.7. 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 24.6 for more details).

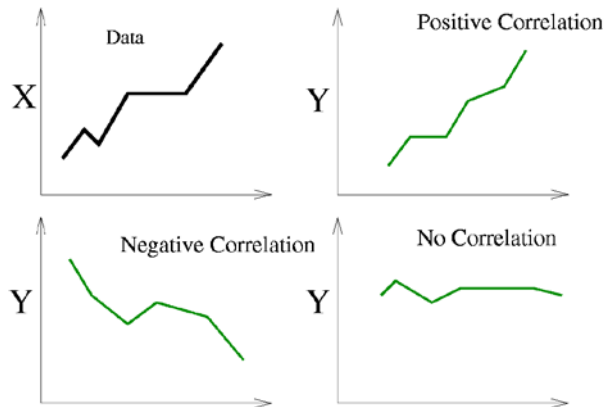
A linear or a quadratic response surface can be used. The metamodel processing speed is approximately  $10^5$  –  $10^6$  finite element nodes a second, where the total nodes to be processed is the number of nodes in the model times the number of states times the number of jobs. FLD computations, which 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.

## 15.4. Correlation

### 15.4.1. 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 15-8 shows examples of a positive, a negative, and zero correlation. If not enough FE

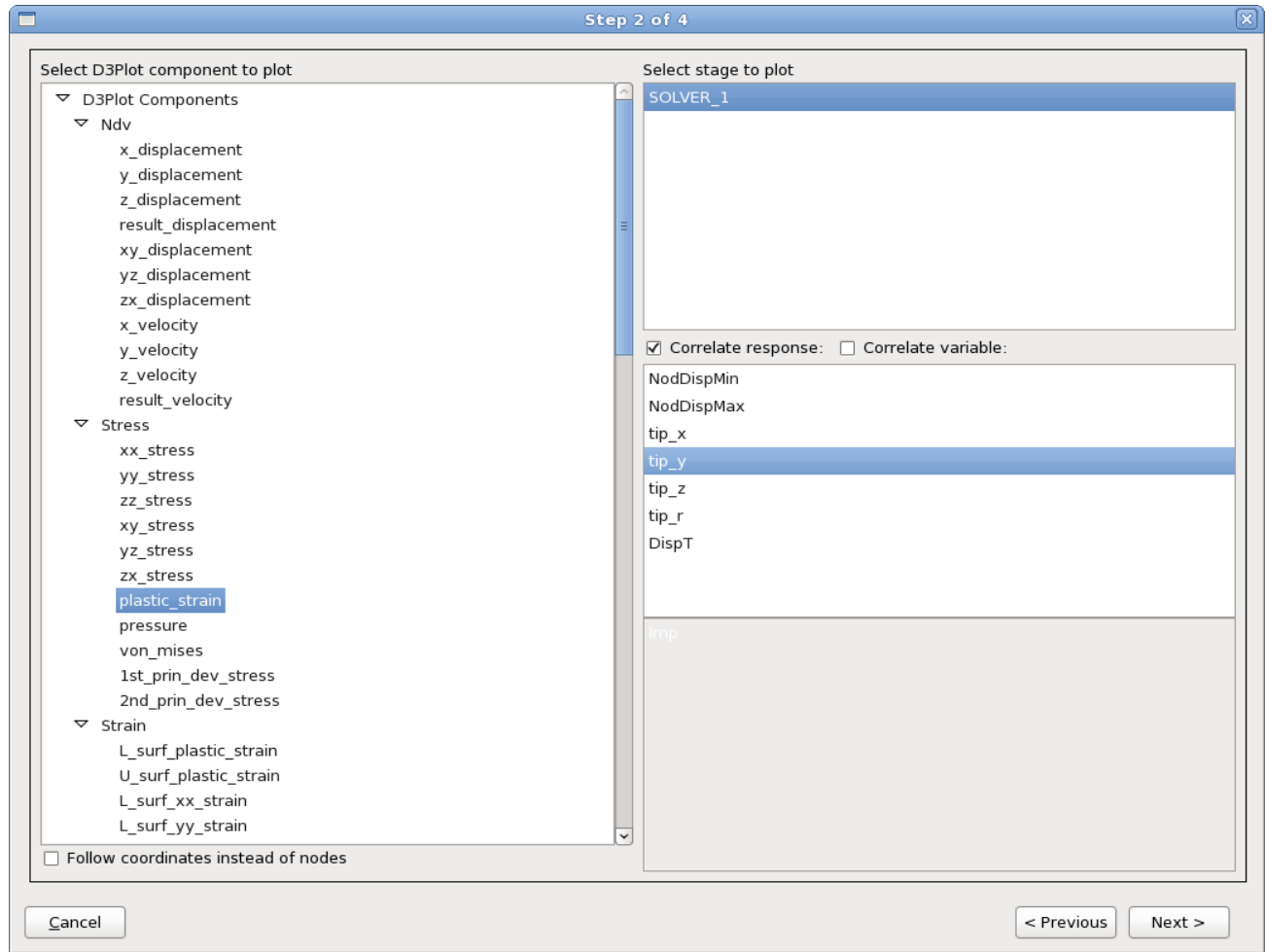
evaluations were conducted, the resulting fringe plot can be visually noisy. Thirty or more FE evaluations may be required. Note that the correlation of history is with respect to a response at a single time instance.



**Figure 15-8** 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.

### 15.4.2. 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.



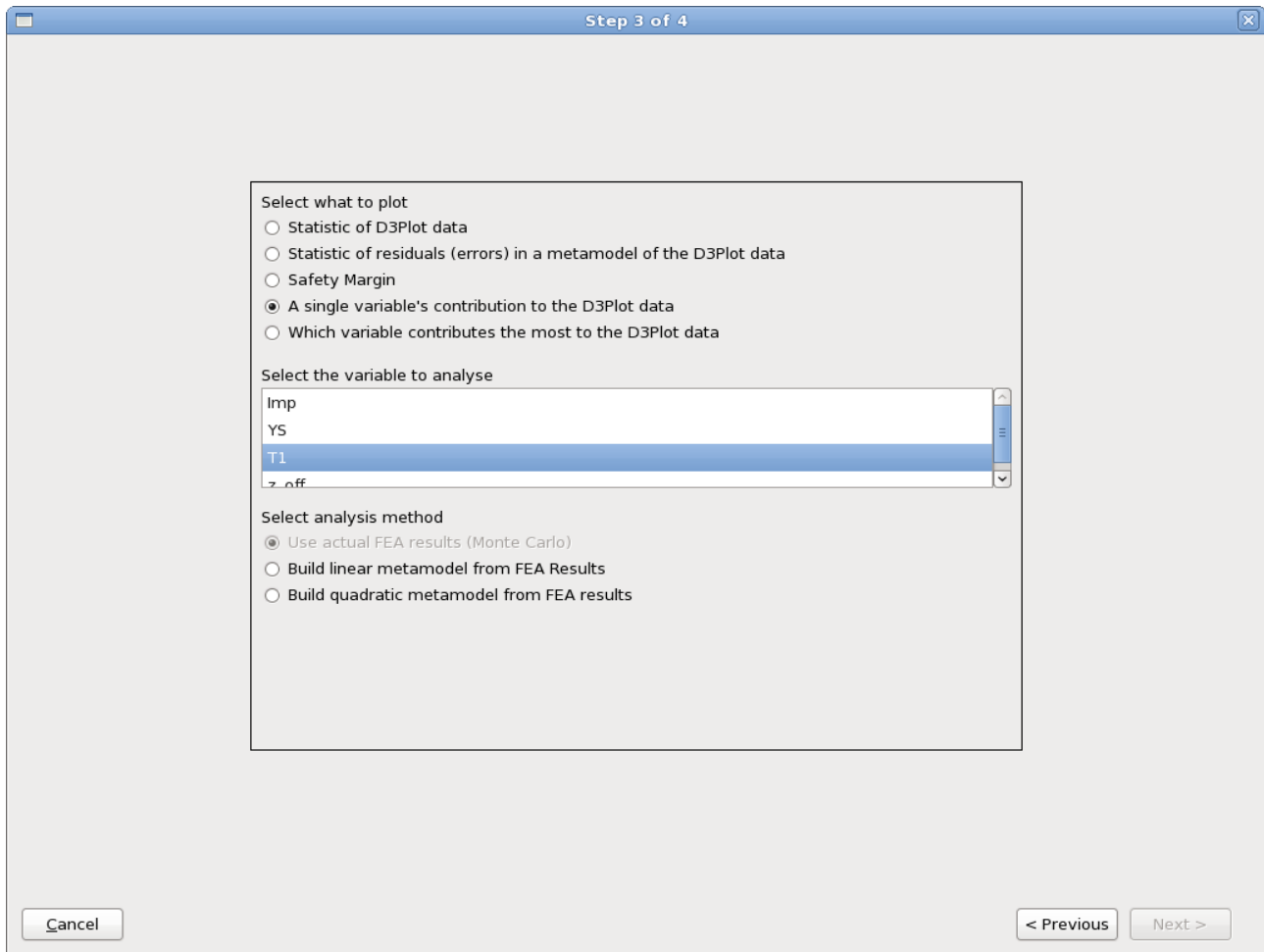
**Figure 15-9:** Viewing the correlation between an LS-DYNA response and an LS-OPT response. Additionally, the correlation between an LS-OPT history and an LS-OPT response can also be viewed.

## 15.5. 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. These results are computed as described in Section 24.7. 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. 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.

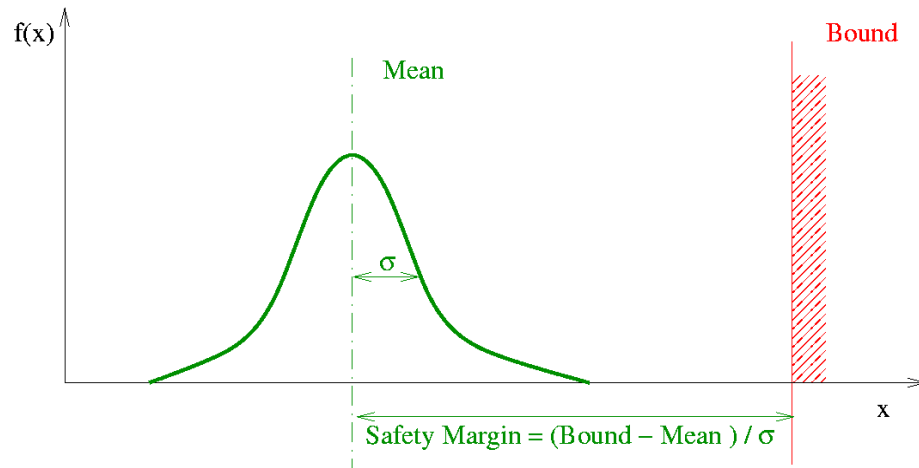


*Figure 15-10: Viewing the stochastic contribution of a single variable.*

## 15.6. Safety margin

The safety margin as shown in Figure 15-11 can be displayed in three ways:

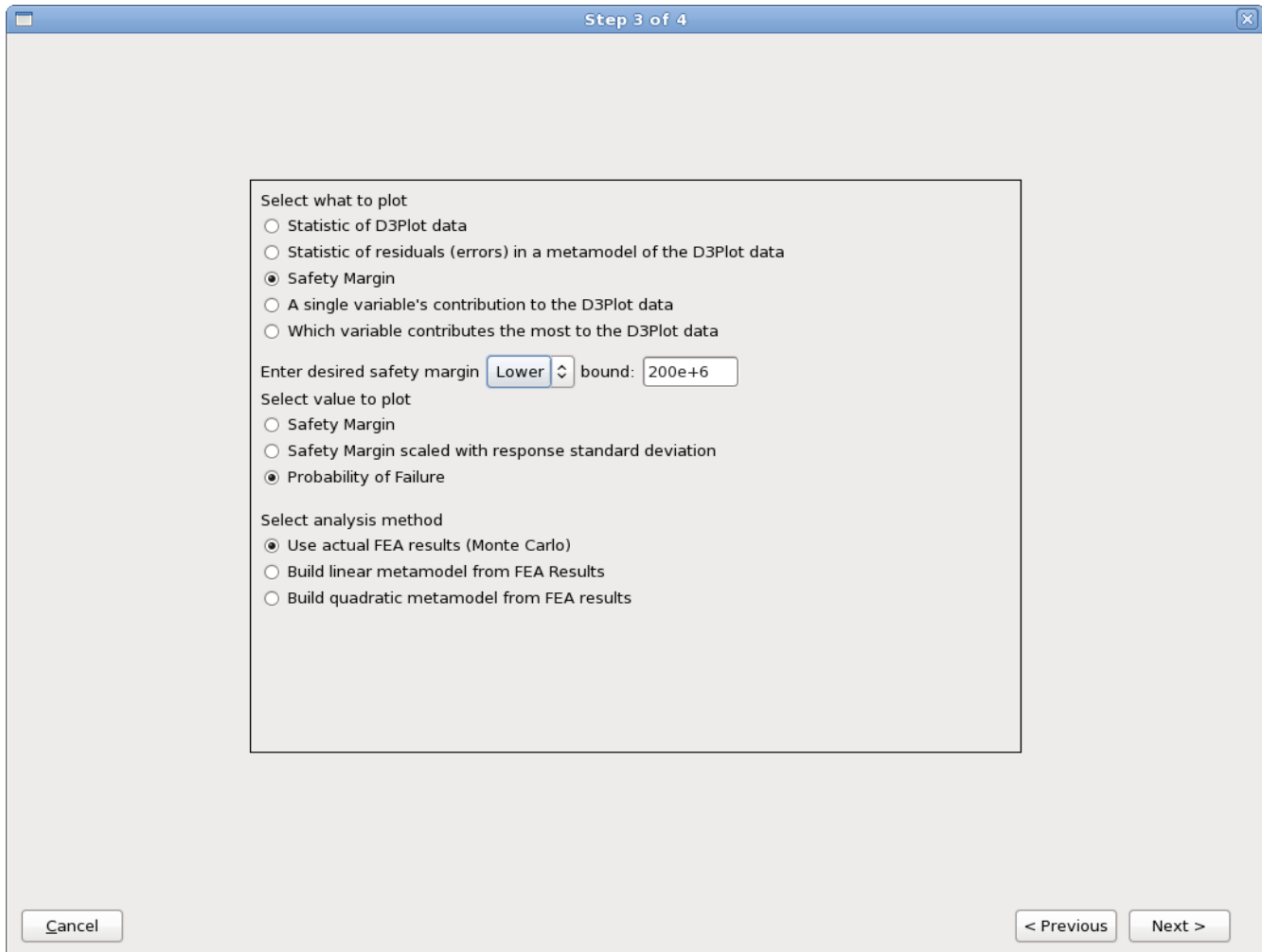
1. The safety margin — the difference between the bound and mean,
2. The safety margin measured (scaled) in standard deviations (sigmas), and
3. The probability of exceeding the bound.



**Figure 15-11:** 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 15-12. 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 24.4.4) using the normal distribution together with the safety margin measured in standard deviations (sigmas). The computation is therefore done in the six-sigma sense interpretation — the number of sigmas (standard deviations) is the unit of measure. If a Monte Carlo computation of the probability is desired, then it must be computed using a response in viewer; if this response was not defined originally then it must be extracted from the d3plot database: first defining a d3plot response, do a repair/extract, and use Viewer.

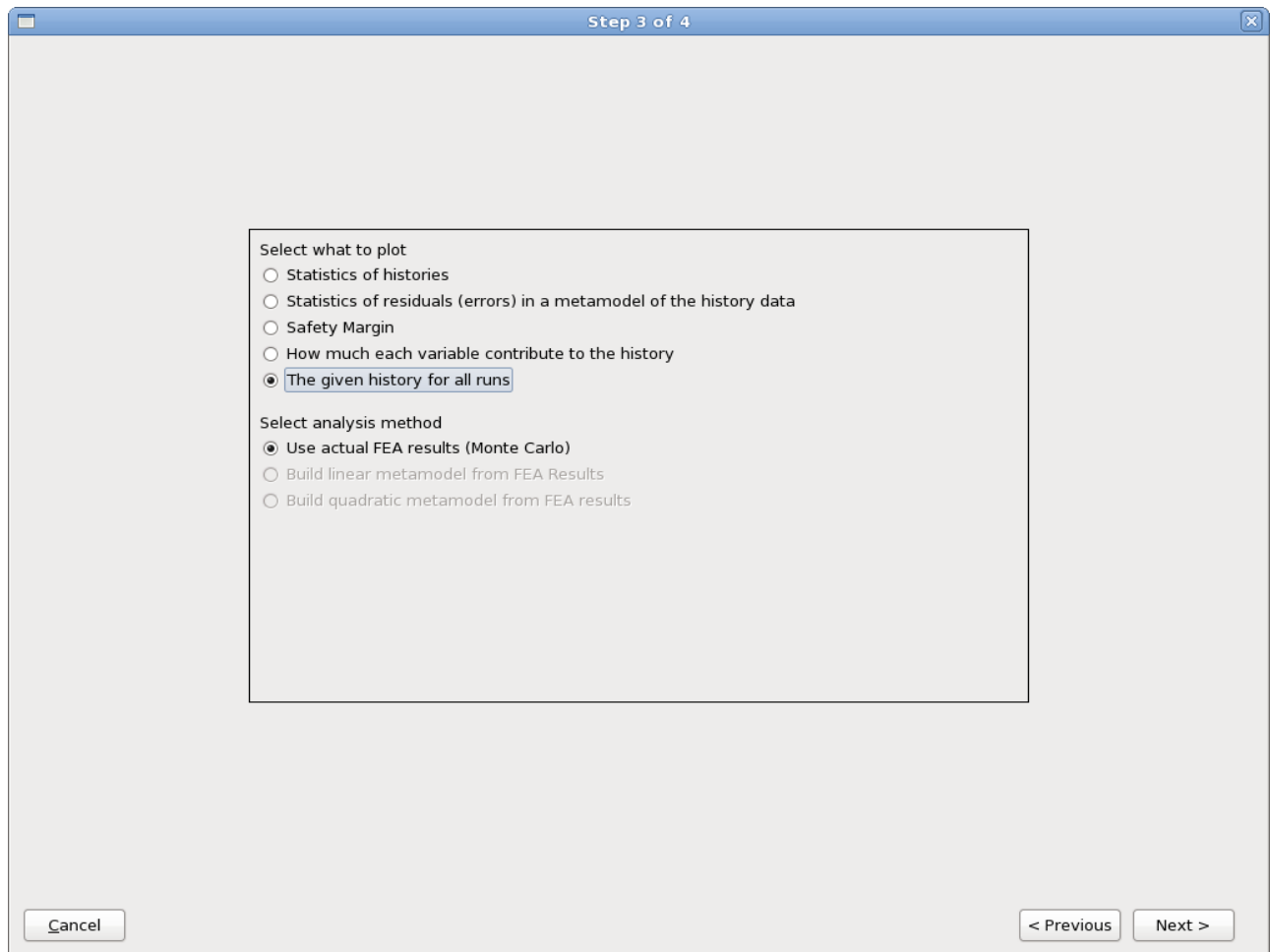


**Figure 15-12:** Plotting a safety margin or the probability of failure requires that the bound must be specified.

## 15.7. Viewing LS-OPT histories

The LS-OPT histories for all the LS-DYNA runs can be viewed simultaneously. See Figure 15-15 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.





*Figure 15-13: Viewing all the LS-OPT histories.*

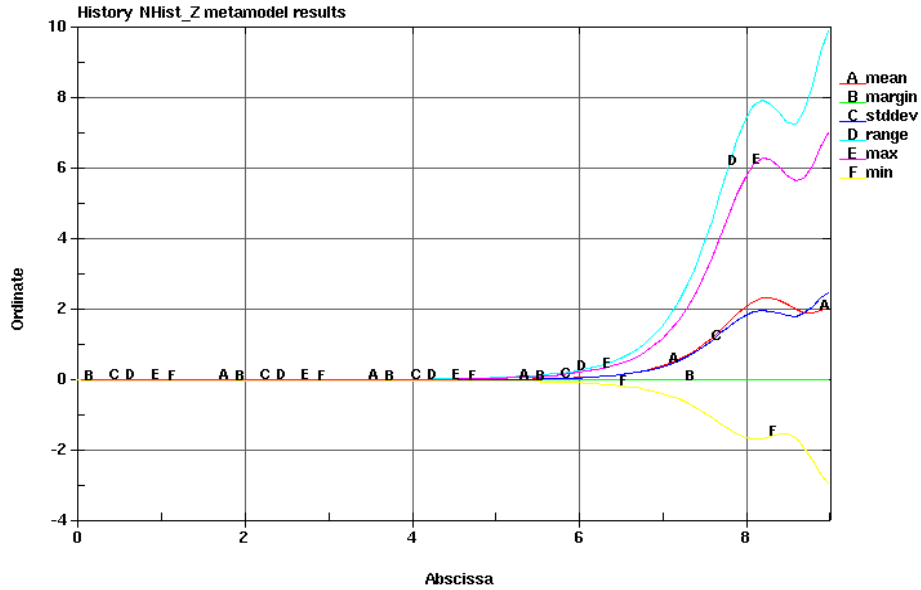


Figure 15-14: Statistics of an LS-OPT history.

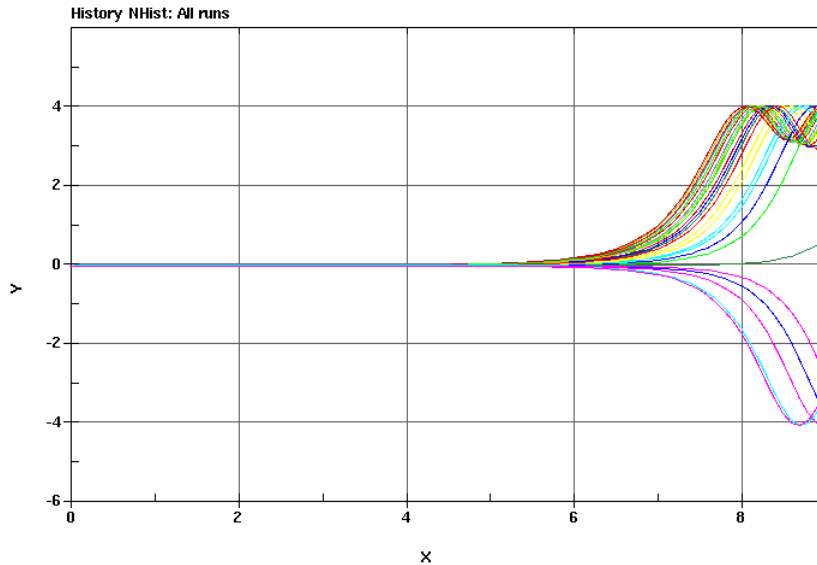
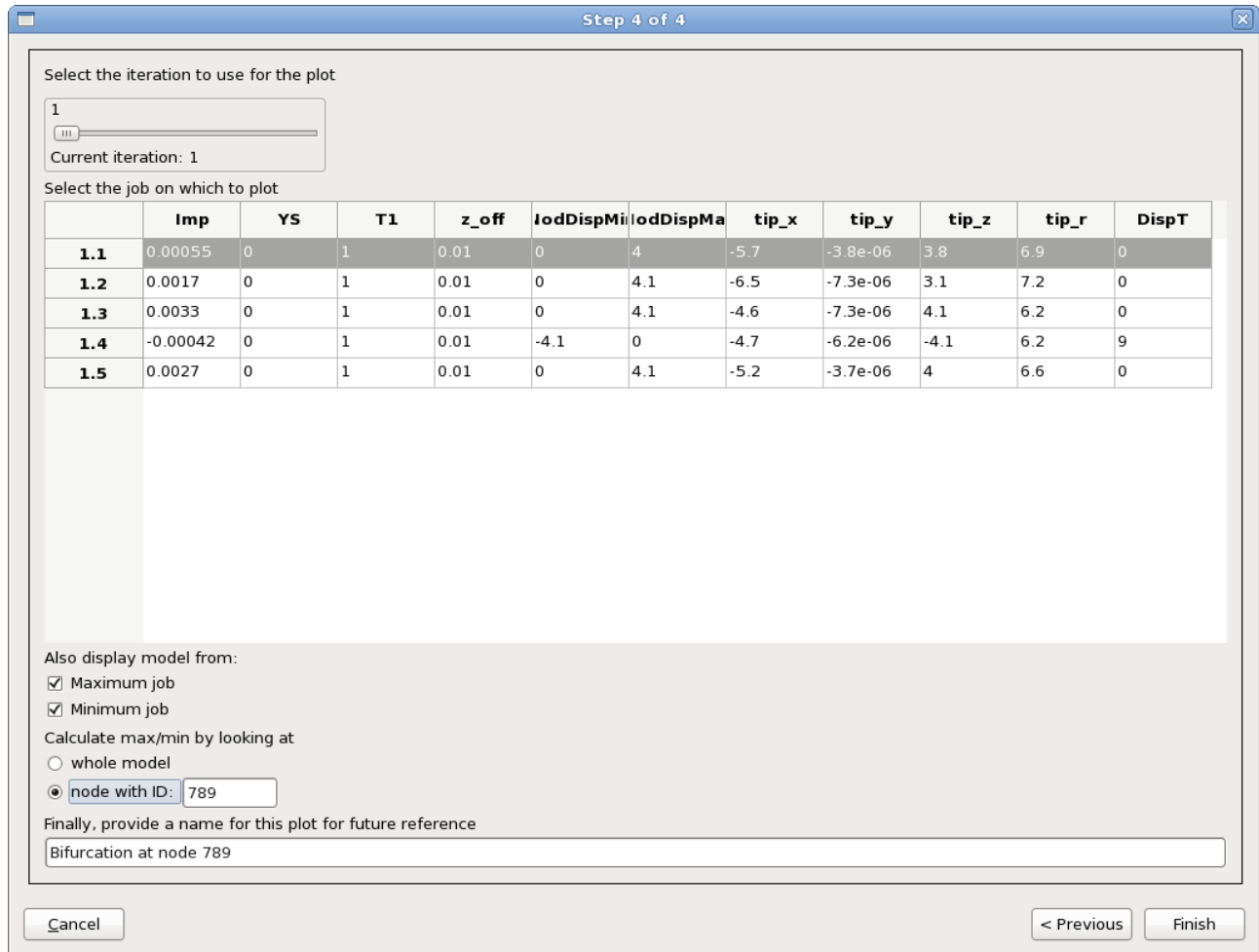


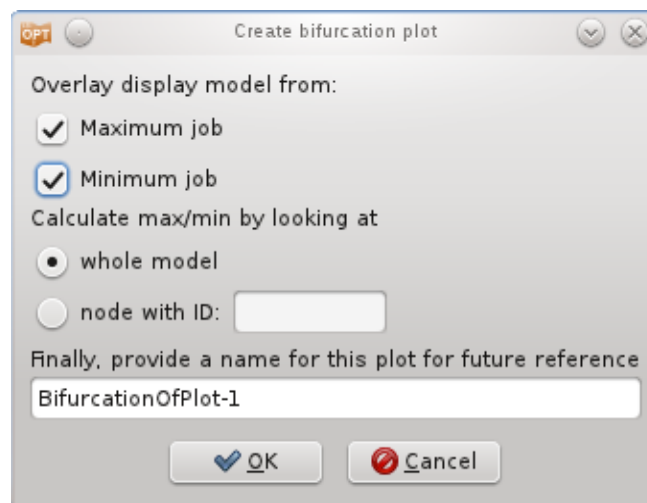
Figure 15-15: The LS-OPT histories of all the LS-DYNA run can be viewed simultaneously.

## 15.8. 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.



**Figure 15-16: 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.



**Figure 15-17: Options to create Bifurcation Plot for an existing plot.**

### 15.8.1. Automatic detection

Automatic detection of the LS-DYNA jobs containing the minimum and maximum outlier can be done as shown in Figure 15-16 and Figure 15-17. 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 **Error! Reference source not found.** for an example of the resulting LS-PREPOST plot.

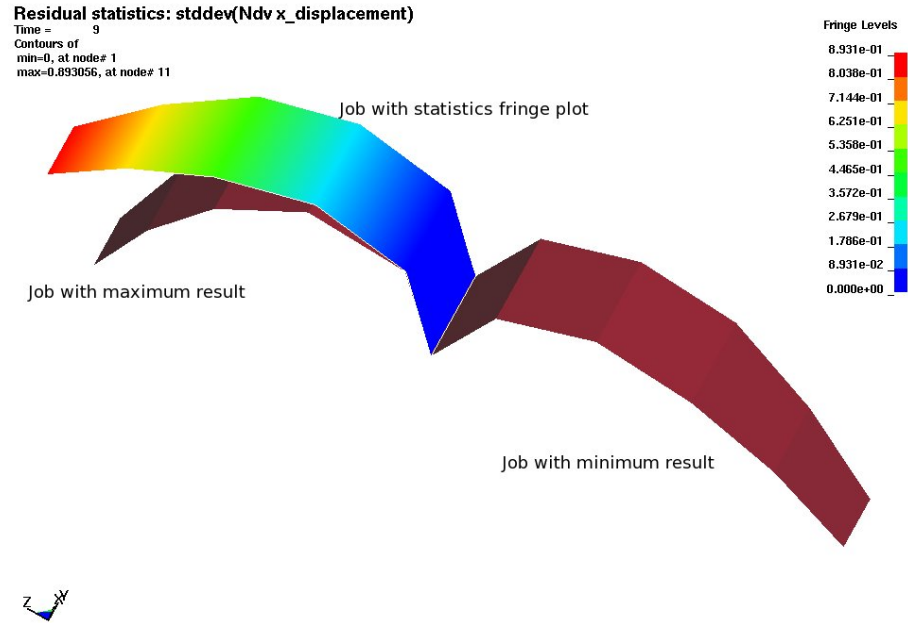
### 15.8.2. Manual detection

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:

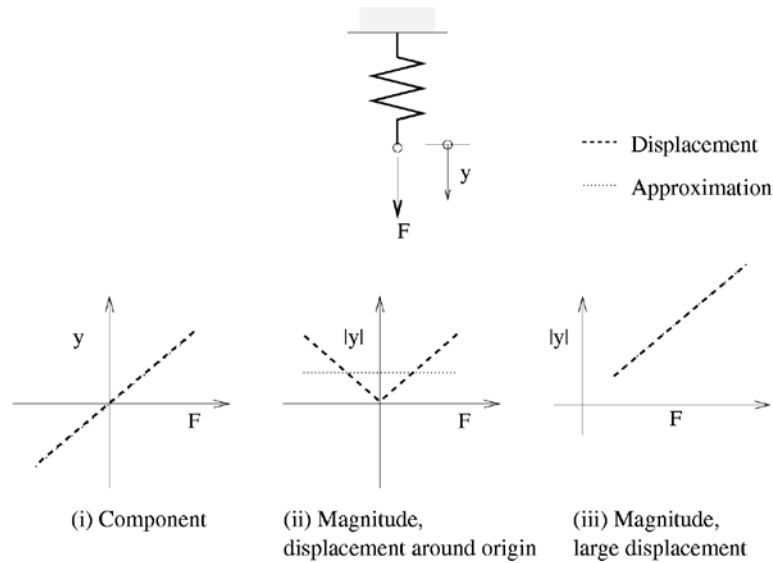
- Engineering knowledge of the structure is important.
- Look at the  $x$ ,  $y$ , and  $z$  components in addition to the displacement magnitude to understand in which direction the bifurcation occurred; most bifurcations are actually best identified considering a displacement component.
- The history 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 15-18: 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.*

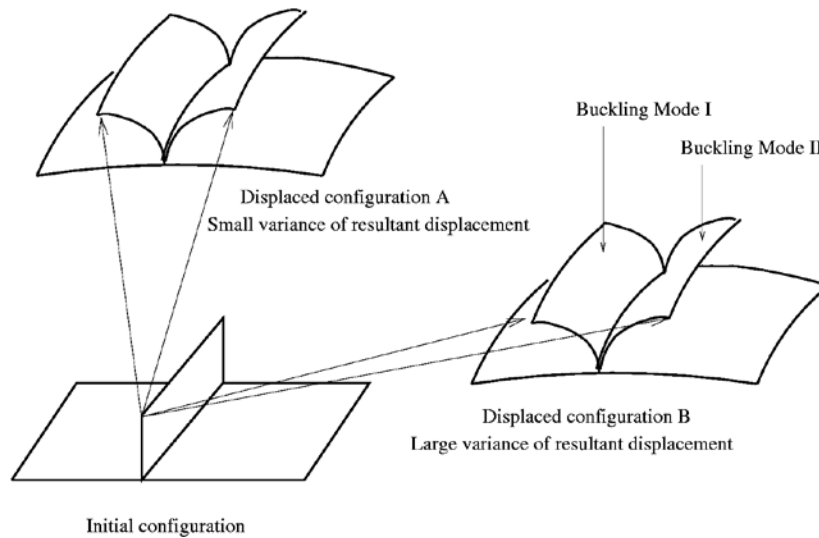
## 15.9. Displacement magnitude issues\*

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 15-19 illustrates. The x, y, and z displacement components do not suffer from this problem.



**Figure 15-19: 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.**

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 15-20 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 15-20: The displacement magnitude can depend on the alignment 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.

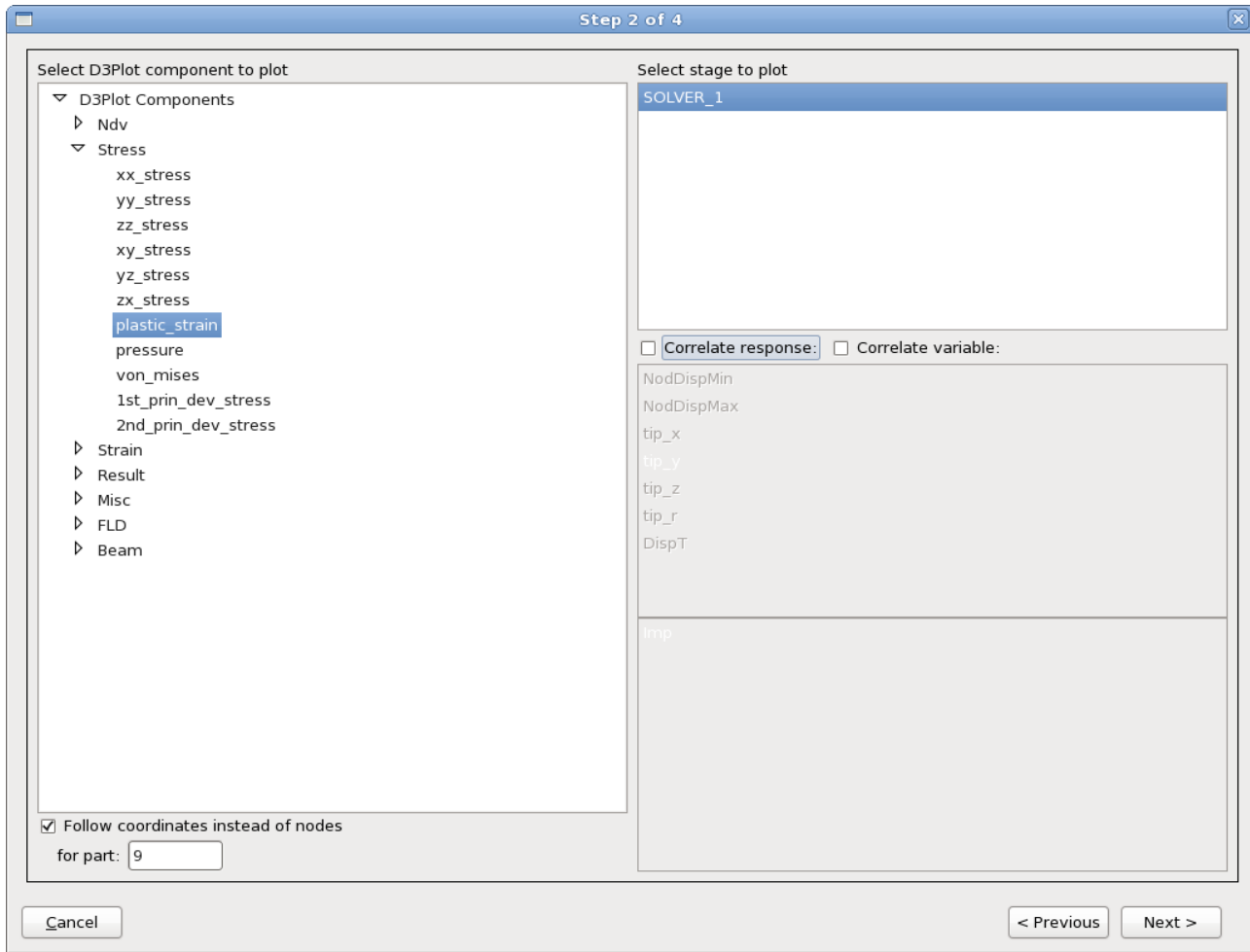
## 15.10. Metalforming options

Metalforming has some special requirements. It is possible to:

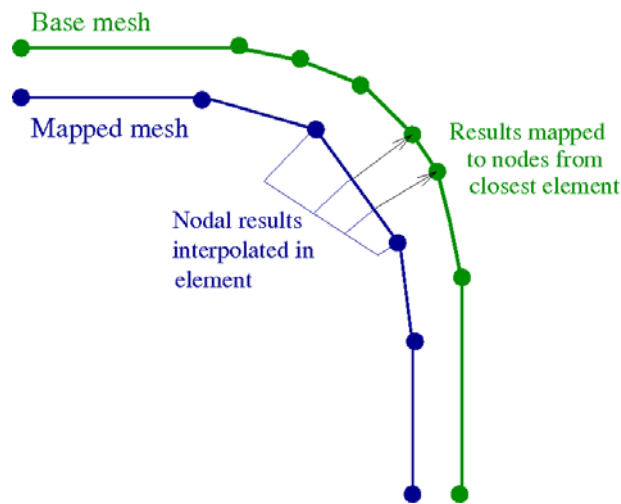
1. Map the results from each iteration to the mesh of the base design. The results will be computed at a specific spatial location instead of a node (Eulerian system). This is required in metalforming because:
  - The adaptivity will result in the different iterations having different meshes.
  - It 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 15-21, Figure 15-22 and Figure 15-23.

2. Specify the FLC curve to be used in the computation of the FLD responses. This can be done by either specifying the number of a curve in the LS-DYNA input deck or using two parameters similar to that being used in LS-PREPOST.

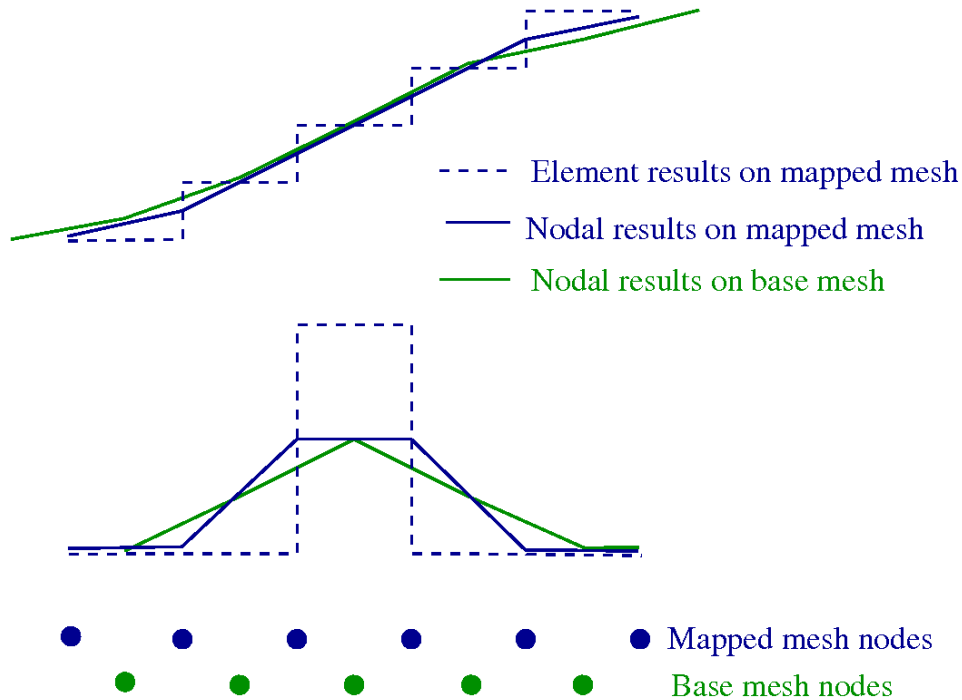


**Figure 15-21:** For metal forming specify that the coordinates instead of the nodes must be followed and specify the part (blank) for which the results must be mapped.



**Figure 15-22:** Interpolation of metal forming results.





*Figure 15-23: Accuracy 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.*

## 15.11. 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 15.2.2 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 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.lsp” for the particular state being run.
3. LS-OPT will open the file “dstats.lsp” 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 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" )
```

## 15.12. 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.

# 16. 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.

## 16.1. 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 6.4.2).

The two main essential components of an algorithm designed for system identification are

- optimization algorithm and
- curve matching metric.

### 16.1.1. 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*, Section 4.7.3. Use linear polynomial metamodels and *D-optimal* point selection which is the default for the selected task and strategy, Section 8.3.2.

### 16.1.2. 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* and using these responses as components associated with the respective target value, Section 9.4. Define this composite as an objective function.

### 16.1.3. Curve matching metric

To calculate the mismatch between the target and the computed curve, define a *Curve Matching* composite, Section 9.5. There are two curve matching metrics available, *Mean Square Error* and *Curve Mapping*. *Mean Square Error* is an ordinate-based curve matching metric. Hence if the curve has steep parts or if the ordinate values are not unique, (the curve is a hysteretic curve), *Curve Mapping* is the metric of choice.

Because *Curve Mapping* uses the length of the curve to calculate the mismatch, filtering of the component history curves is recommended.

### 16.1.4. 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 8.6.

### 16.1.5. 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.

### 16.1.6. Viewer

This section describes some postprocessing options commonly used for parameter identification problems. Further options are described in Chapter 14.

## 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 14.4.1 for further information on the optimization history plot.

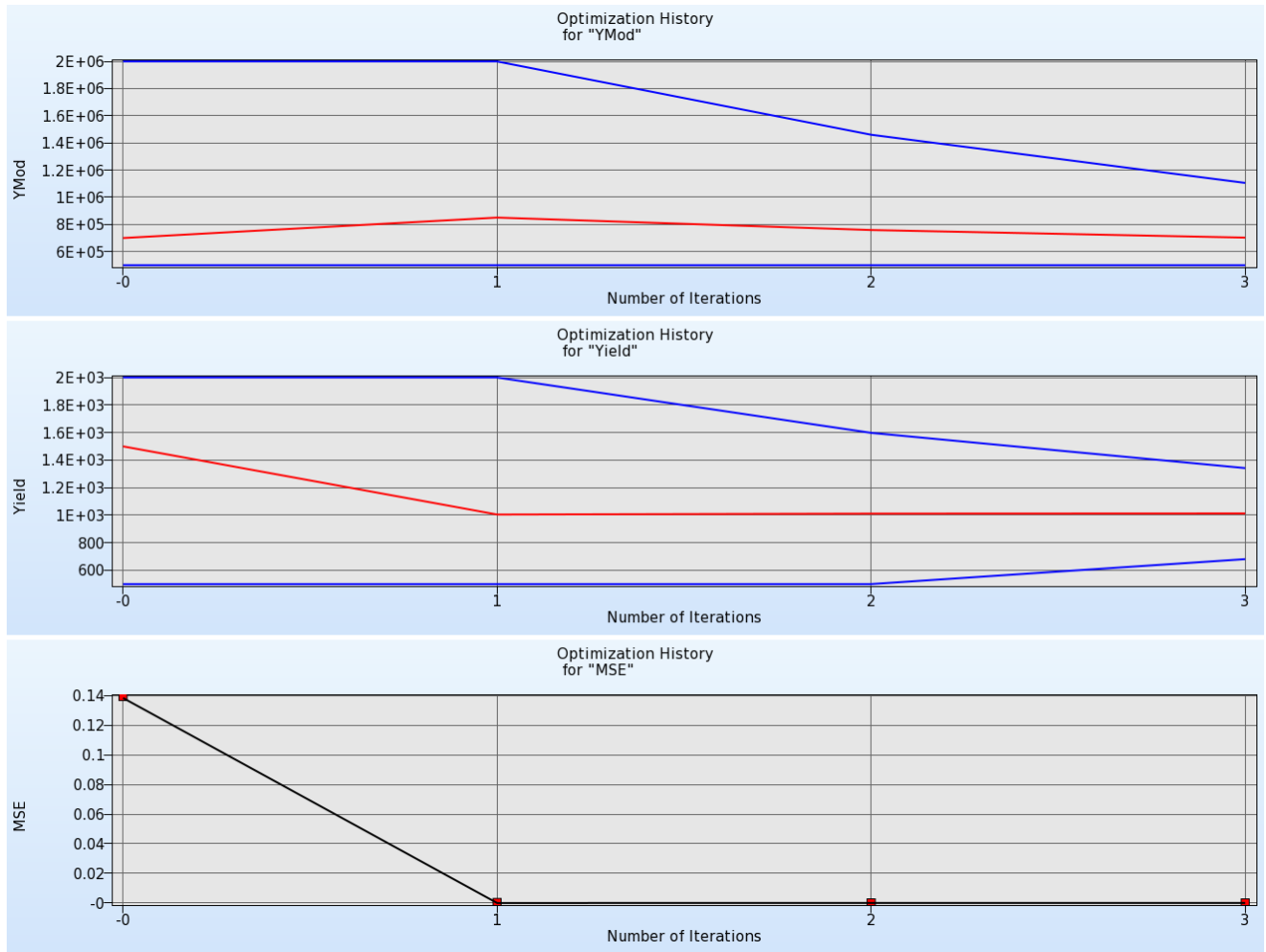


Figure 16-1: Optimization history for variables and objective

### Sensitivities

If there are parameters that do not converge, the sensitivities plot can be used to see if those parameters are insensitive. See Section 14.3.4.

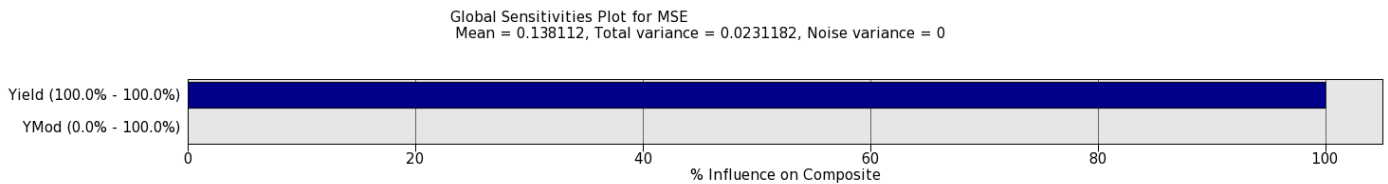


Figure 16-2: Global Sensitivities

## History

The history plot (Section 14.2.4) 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, selection the option *Only optimal* and coloring the curves by iteration visualized the improvement of the optimal curves over the iterations.



*Figure 16-3: Target and computed curves, only optimal curves are displayed for all iterations*

## 16.2. 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.

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 4.10) 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.

### 16.2.1. 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.

### 16.2.2. Sequential

An approach for generating a metamodel to a specified prediction accuracy (using the PRESS metric, see Section 20.3.5) is to use an iterative method.

Select *Metamodel based Optimization* for the main task, and the *Sequential* strategy, Section 4.7.2. 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 8.3.4.

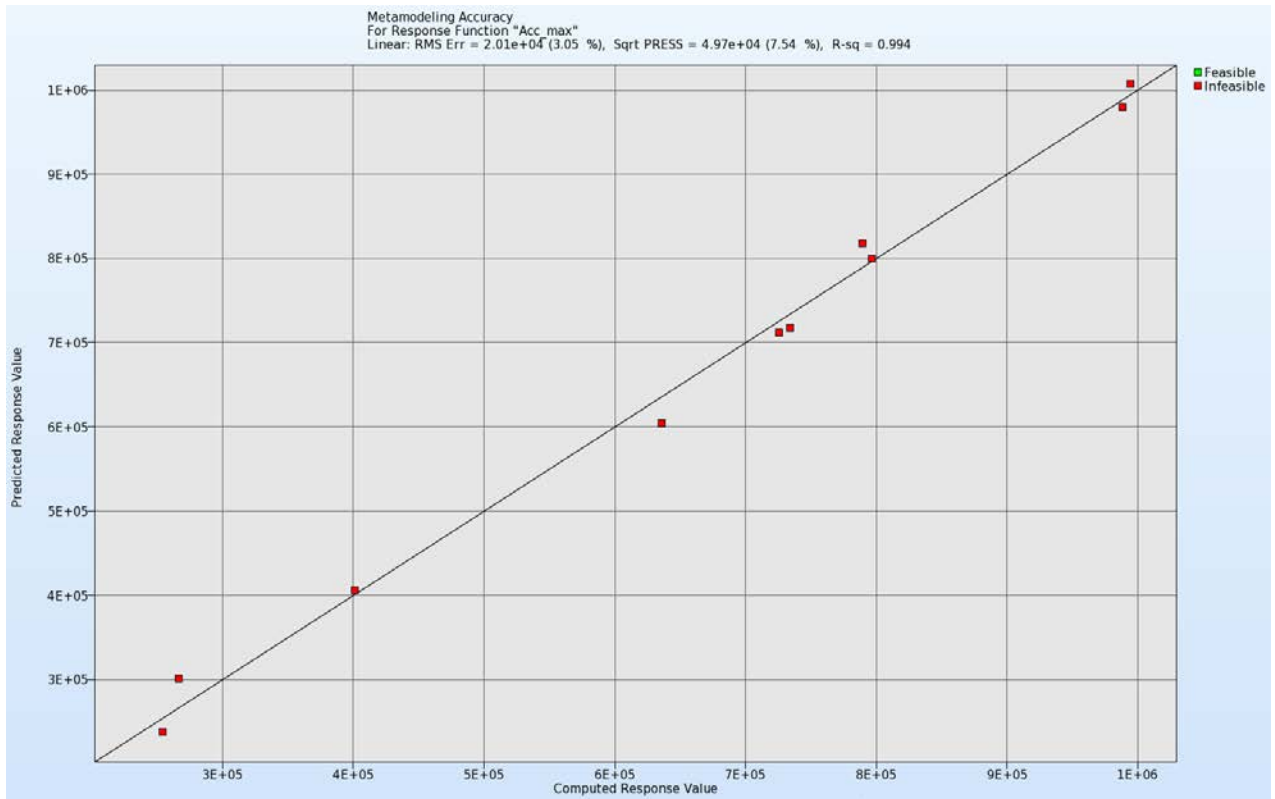
An appropriate termination criterion for a sequential approach is *Response Accuracy Tolerance*, Section 11.1.2. Make sure to use the OR option and set the non-accuracy tolerances to 0. The number of iterations to be performed is again a compromise between accuracy and computational effort.

### 16.2.3. Viewer

This section describes some postprocessing options commonly used for a sensitivity analysis. Further options are described in Chapter 14.

#### Accuracy

Use the accuracy plot (Section 14.3.3) and the error measures displayed in the title to judge the quality of the metamodels.



*Figure 16-4: 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 14.3.4). By default, the values are sorted by significance, hence the ranking of the parameters can be directly taken from the order in the plots.



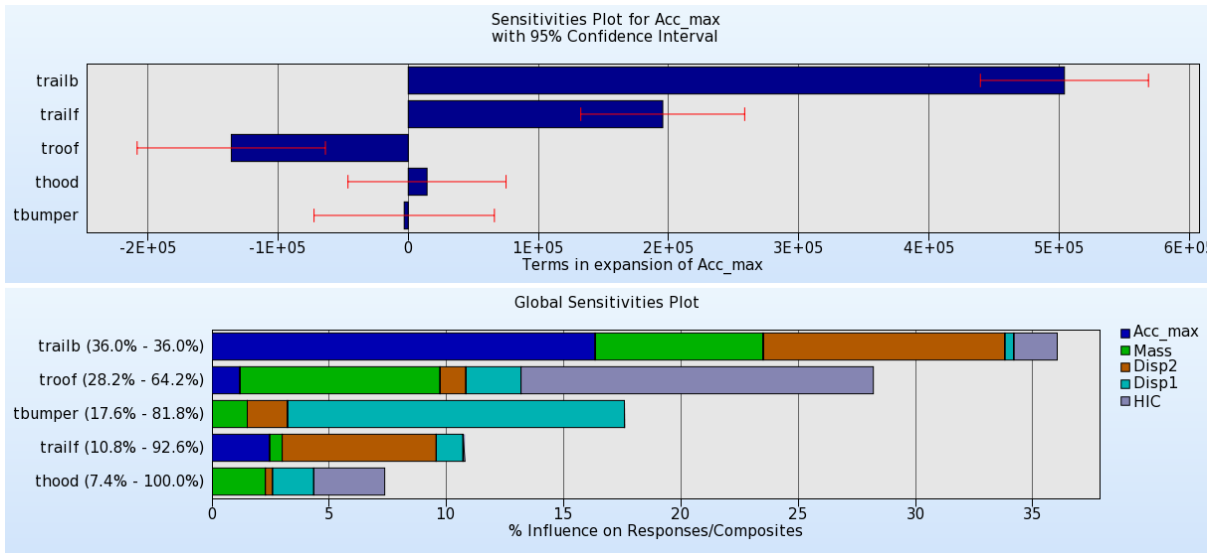
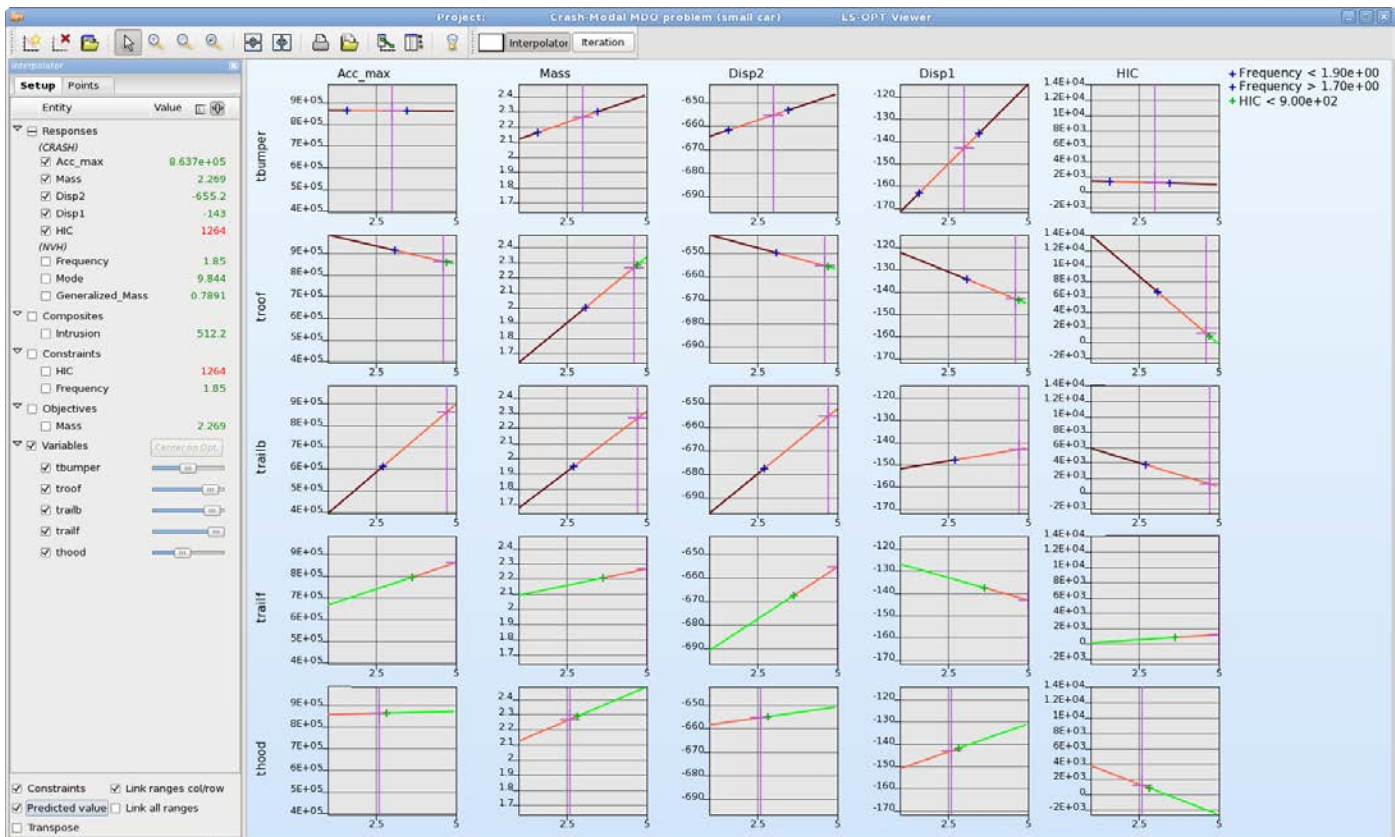


Figure 16-5: ANOVA values for a single response; Sobol values for multiple responses

### Interpolator

The Interpolator plot (Section 14.3.2) 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.



**Figure 16-6:** Interpolator plot: 2D surface plots for variables vs. responses; constraints on the metamodel and the predicted value for the selected parameter combination are displayed.

### 16.3. 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 7.3). 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.

See the examples in Section 17.5 for the command file format.

## 16.4. 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 4.9. 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 4.7.

### 16.4.1. 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.

### 16.4.2. 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.

### 16.4.3. Viewer

Various plot types that are available for the visualization of Pareto optimal solutions described in Section 14.5 can be used to explore those solutions and select the appropriate optimal solution that fits best to the application.

## 16.5. Shape Optimization

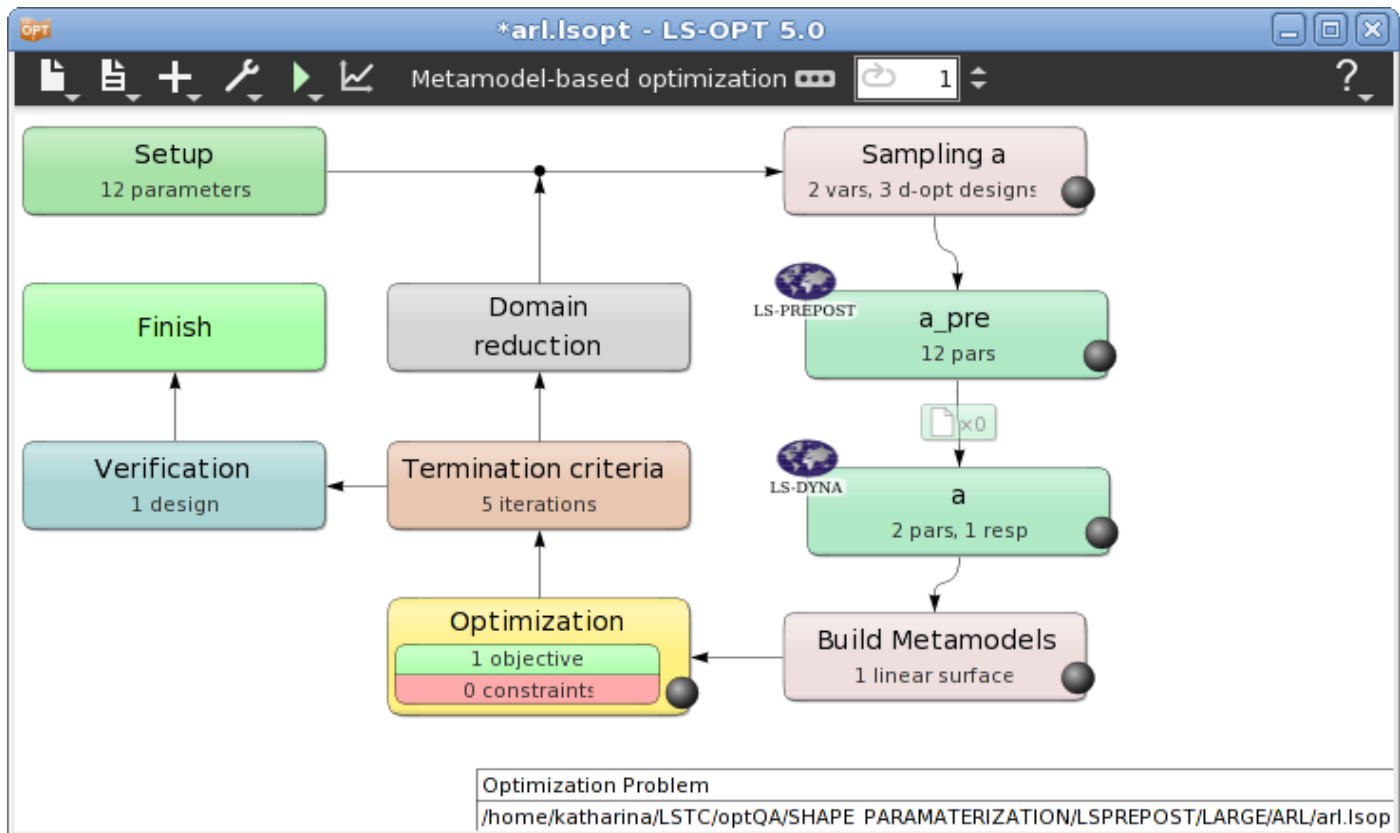
To implement geometrical parameters in LS-OPT, an interface to a preprocessor has to be used. The available interfaces, which include a user-defined option, are described in Chapter 5. The process chain to

be optimized is at least a two-stage process including a preprocessor and a solver, Figure 16-7. 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 pre-processors 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 pre-processor output files.

The recommended task and strategy for single objective optimization is *Metamodel based Optimization and Sequential with Domain Reduction*, Section 4.7.3.

A legacy *com* file (the input file format used before Version 5) containing a pre- or postprocessor definition will automatically translate to a multi-stage process.



**Figure 16-7:** Possible setup for a shape optimization. *a\_pre* interfaces with a preprocessor that generates the geometry of the model depending on parameters.

## 16.6. 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 16-8) 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 7.1.7).

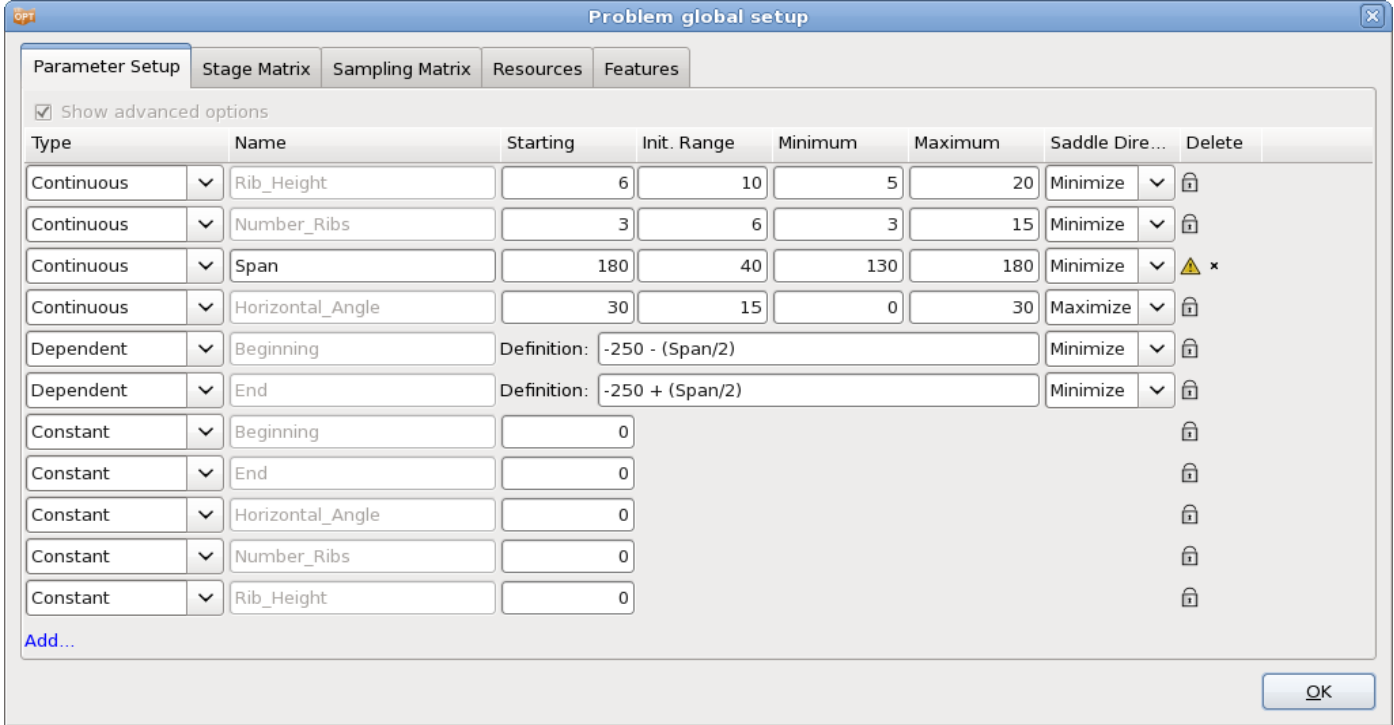


Figure 16-8: Parameter definition for a worst-case design optimization

# II – Examples

# 17. Examples – Optimization

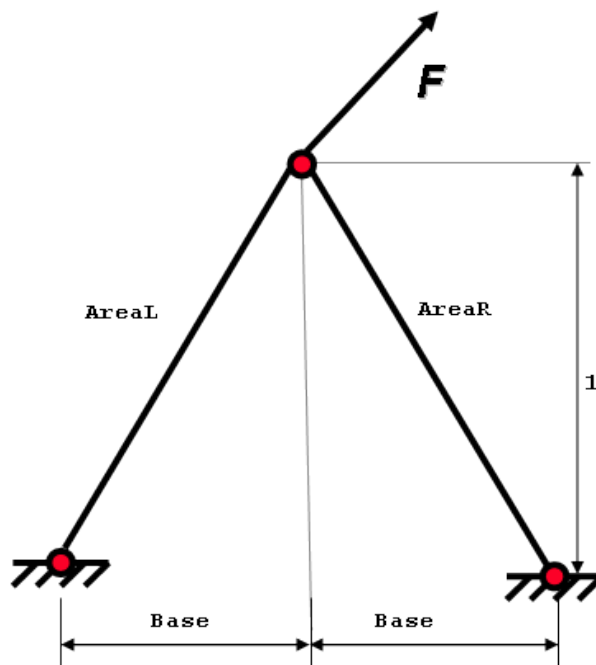
## 17.1. Two-bar truss (3 variables)

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.

### 17.1.1. Description of problem

This example problem as shown in Figure 17-1 has one geometric and two element sizing variables.



*Figure 17-1: 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:  $F = (F_x, F_y) = (24.8kN, 198.4kN)$ .

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,

$$\begin{aligned} -1 \leq \sigma_1(x) &= 0.124 \cdot \sqrt{1 + Base^2 \left( \frac{8}{AreaL} + \frac{1}{Base \cdot AreaL} \right)} \leq 1, \\ -1 \leq \sigma_2(x) &= 0.124 \cdot \sqrt{1 + Base^2 \left( \frac{8}{AreaR} - \frac{1}{Base \cdot AreaR} \right)} \leq 1. \end{aligned}$$

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
```

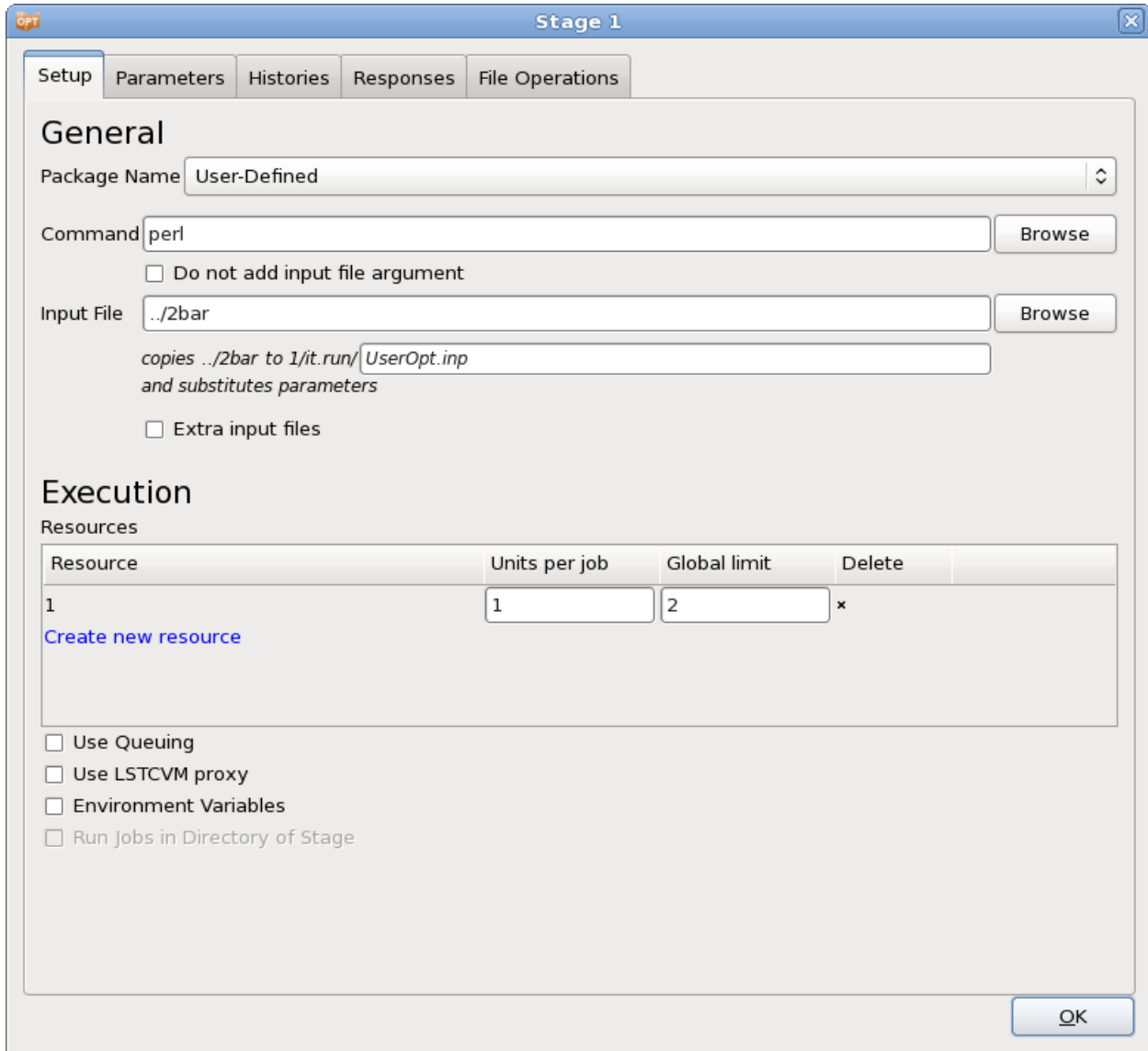


```

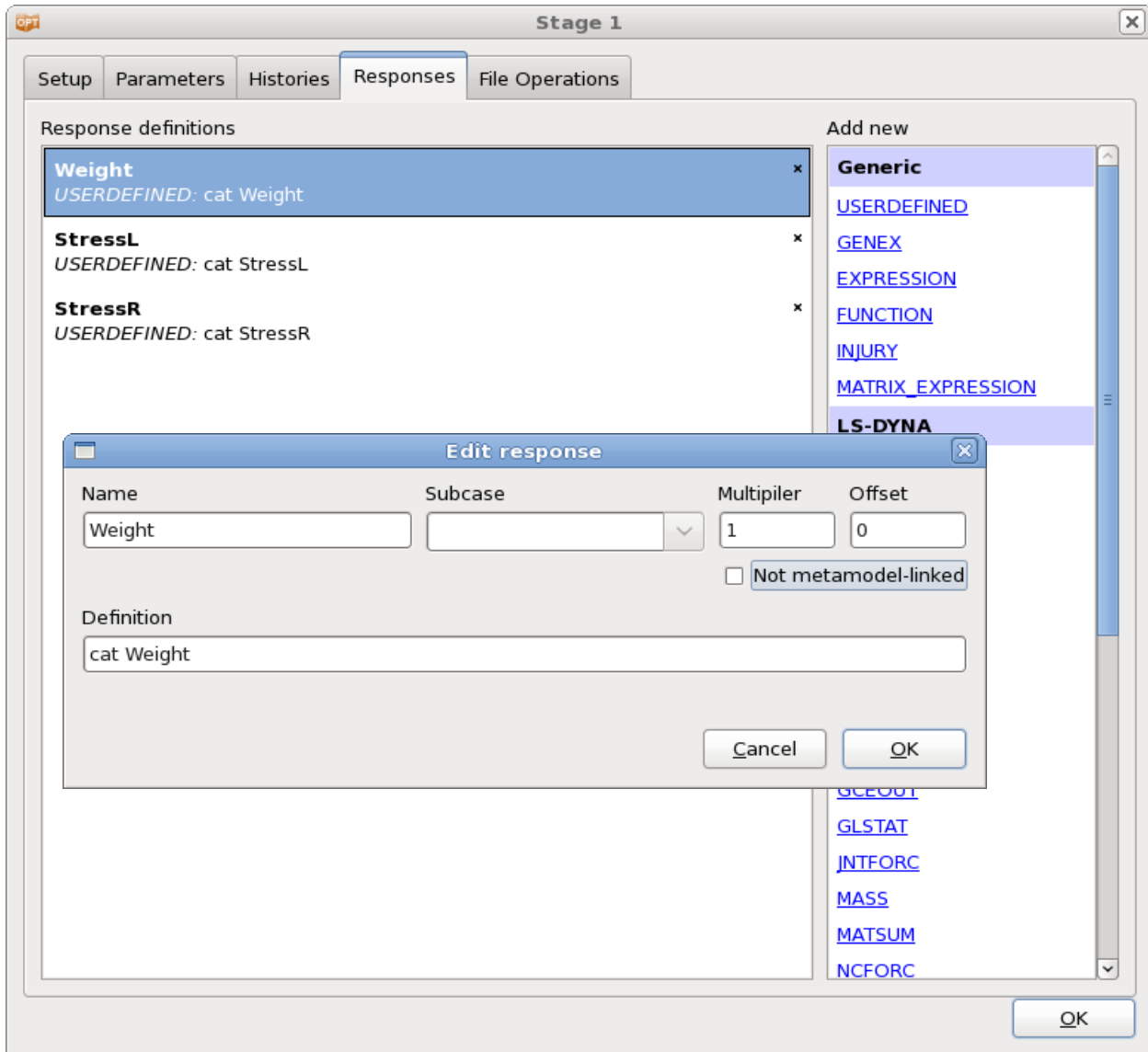
#*****
# print "N o r m a l\n";
#

```

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 17-2. The response values are written to files that are used to define the user-defined responses in LS-OPT, Figure 17-3.



**Figure 17-2:** Stage dialog Setup for a user-defined solver. Parameters are specified in the input file using the LS-OPT parameter format <<>>.

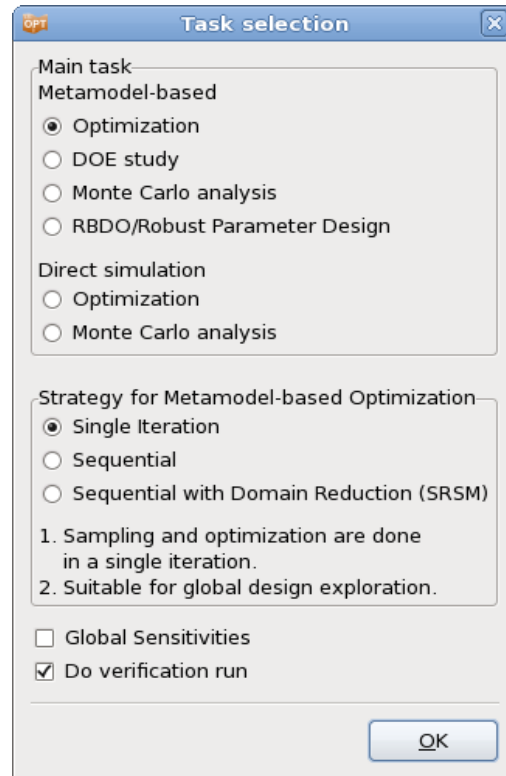


*Figure 17-3: User-defined response definitions*

The problem is solved using metamodel based optimization, Figure 17-4. In Sections 17.1.2 to 17.1.4, a typical semi-automated optimization procedure is illustrated. The last subsection 17.1.5 shows how an automated procedure can be specified for this example problem.

### 17.1.2. A first approximation using linear response surfaces

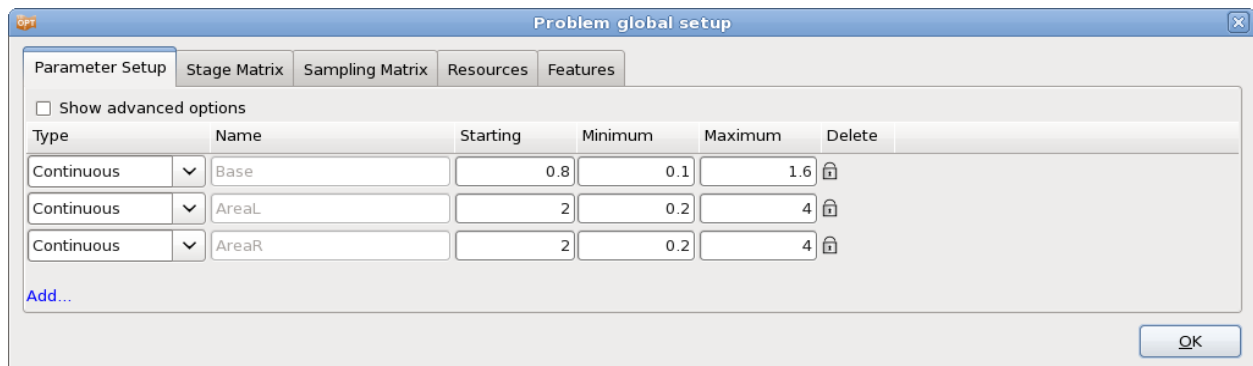
To get a first rough approximation of the problem, a single iteration is run, Figure 17-4.



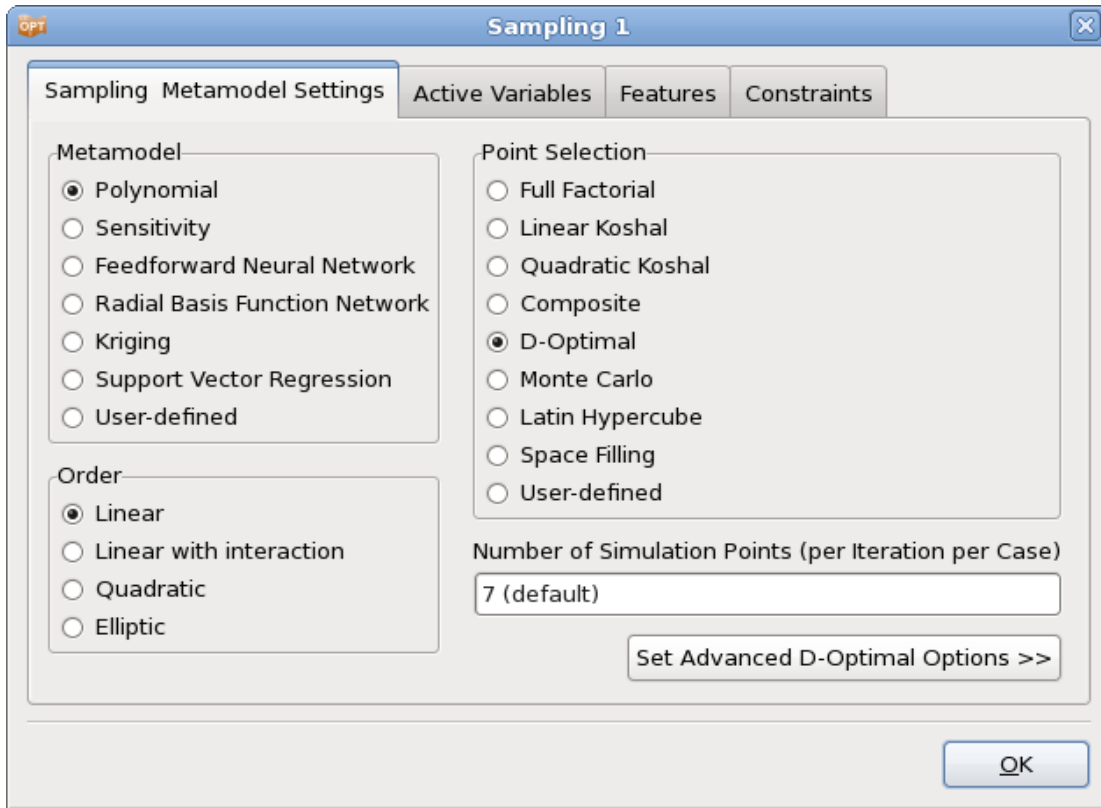
*Figure 17-4: Task dialog; Selection for a metamodel base optimization using a single iteration.*

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 17-5. The starting values are used for the initial design.

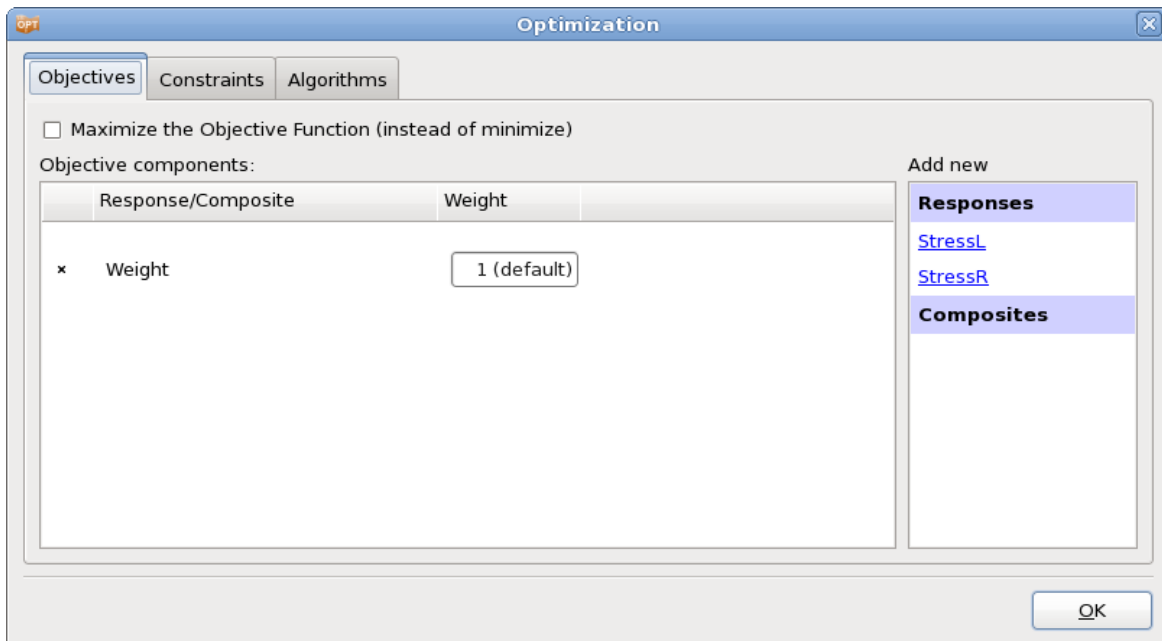
The **Sampling** dialog, allows for setting the metamodel and point selection, Figure 17-6. To get a first rough approximation of the problem, the metamodel type is chosen to be a linear polynomial. The default number of points is automatically adapted to the number of variables and the metamodel type.



*Figure 17-5: Parameter Setup; specification of design space and starting values or all parameters*

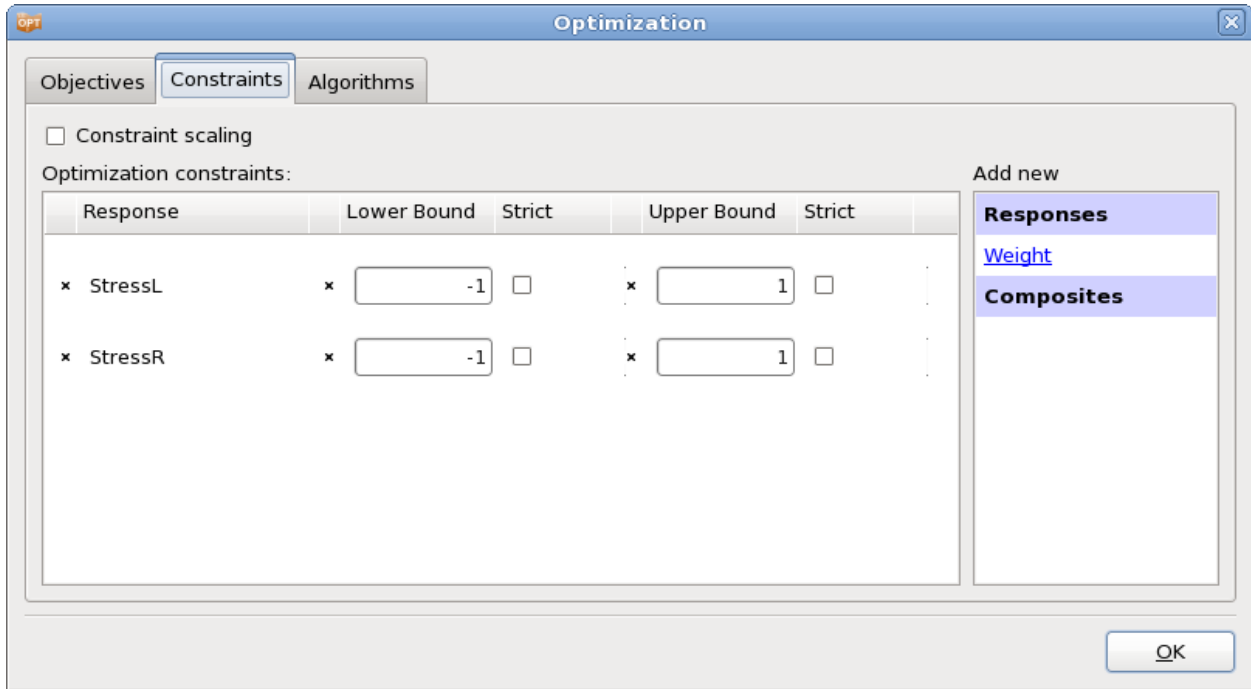


*Figure 17-6: Sampling and Metamodel; Select metamodel type Polynomial with order Linear; use the defaults for Point Selection and number of points*



*Figure 17-7: 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 17-7. 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 17-8.



**Figure 17-8: 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 17-9 and Figure 17-10). 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 17-1. 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 17-11.

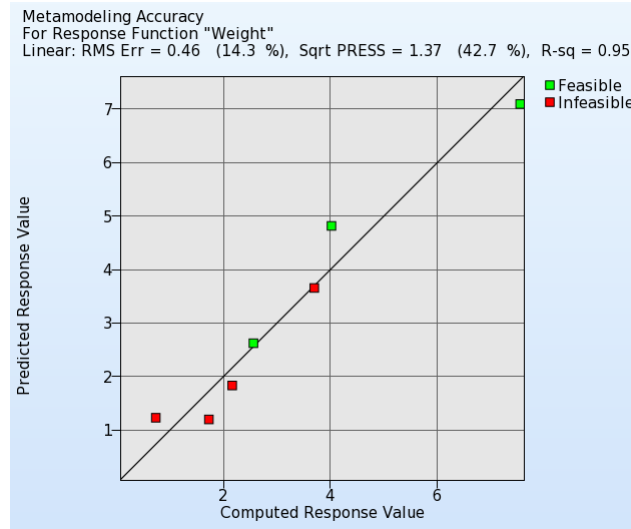


Figure 17-9: Accuracy of linear metamodel for response "Weight"

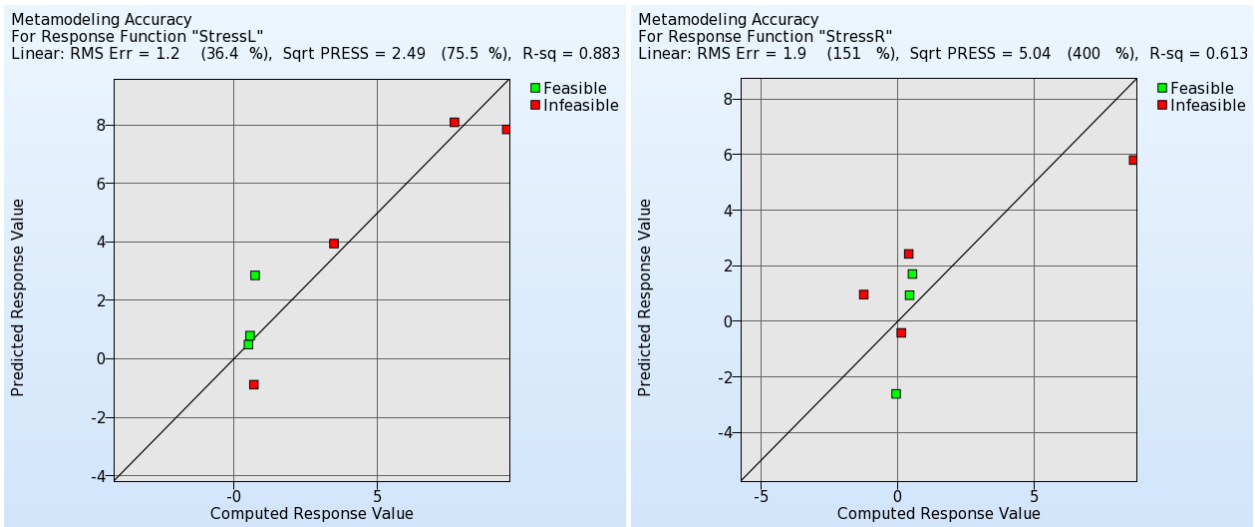
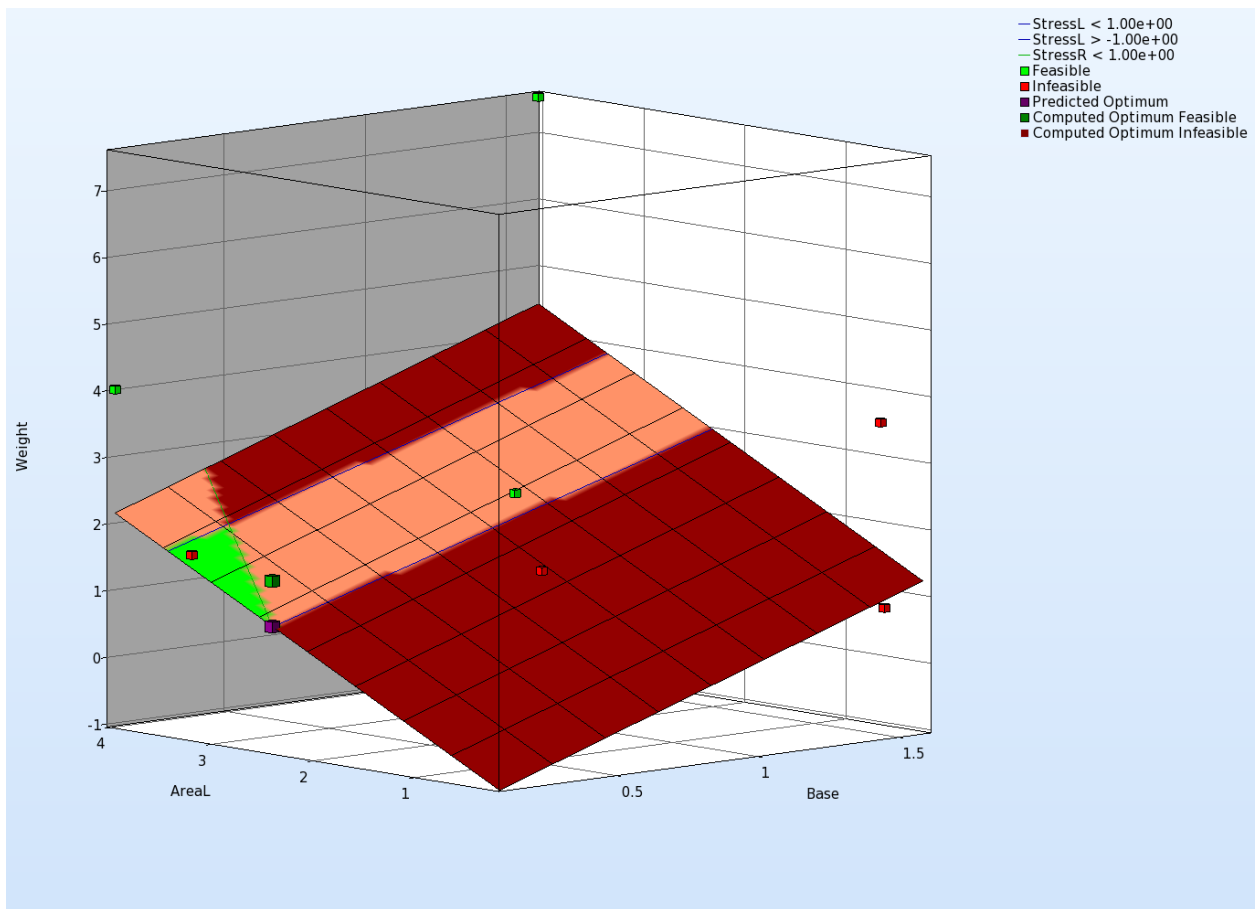


Figure 17-10: Accuracy of linear metamodel of responses "StressL" and "StressR"

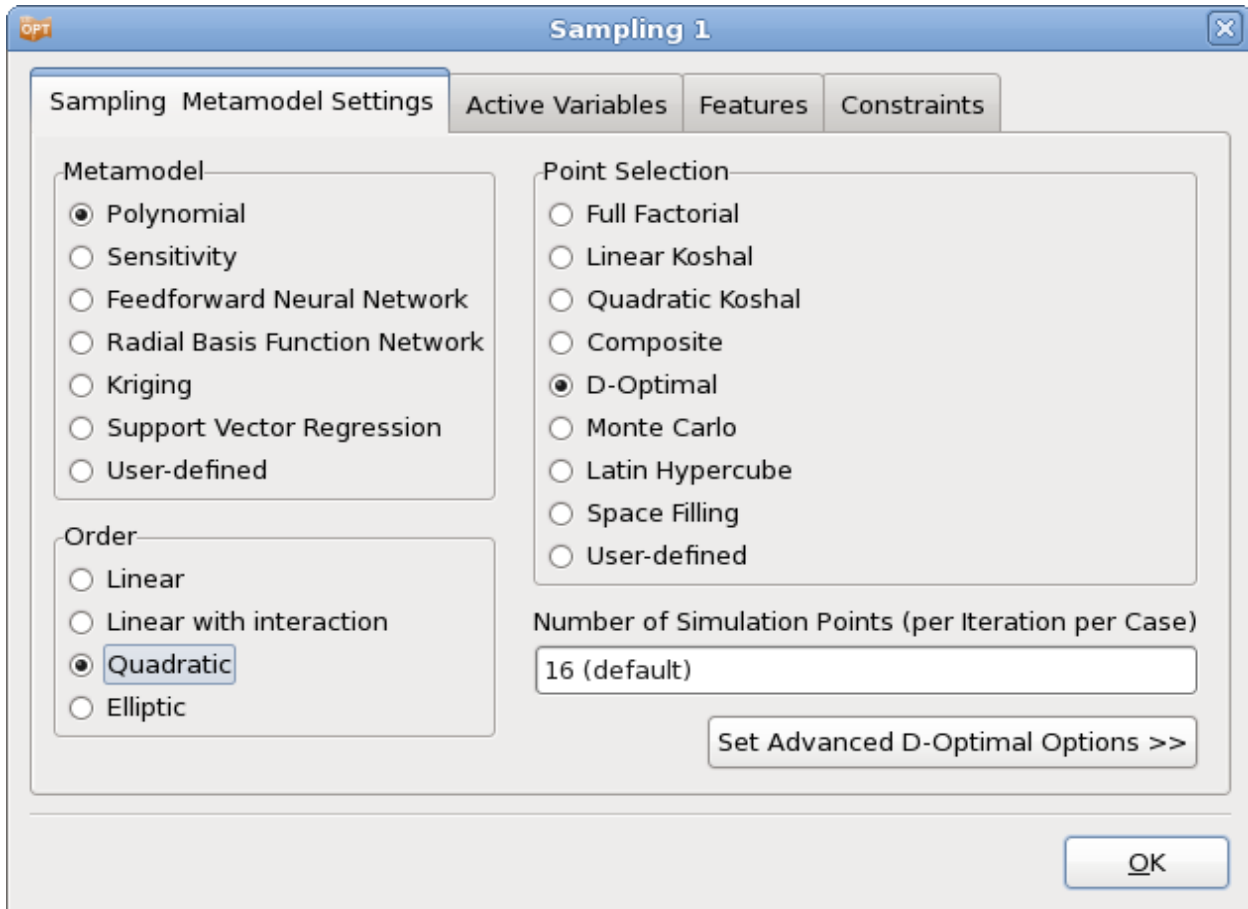
**Table 17-1: Comparison of baseline run and optimum (single iteration, linear metamodel)**

	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 17-11: Surface plot for objective function Weight; constraints are displayed on the metamodel.**

### 17.1.3. 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 17-12. The number of points will automatically update.



*Figure 17-12: 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 17-13 and Figure 17-14.

An improved design is predicted with the approximate constraint values (stress) becoming active, Table 17-2. 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 17-15.



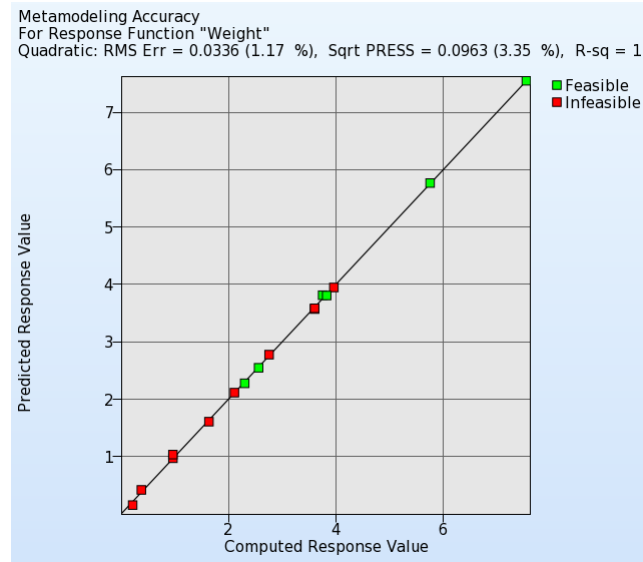


Figure 17-13: Accuracy of quadratic metamodel for response "Weight"

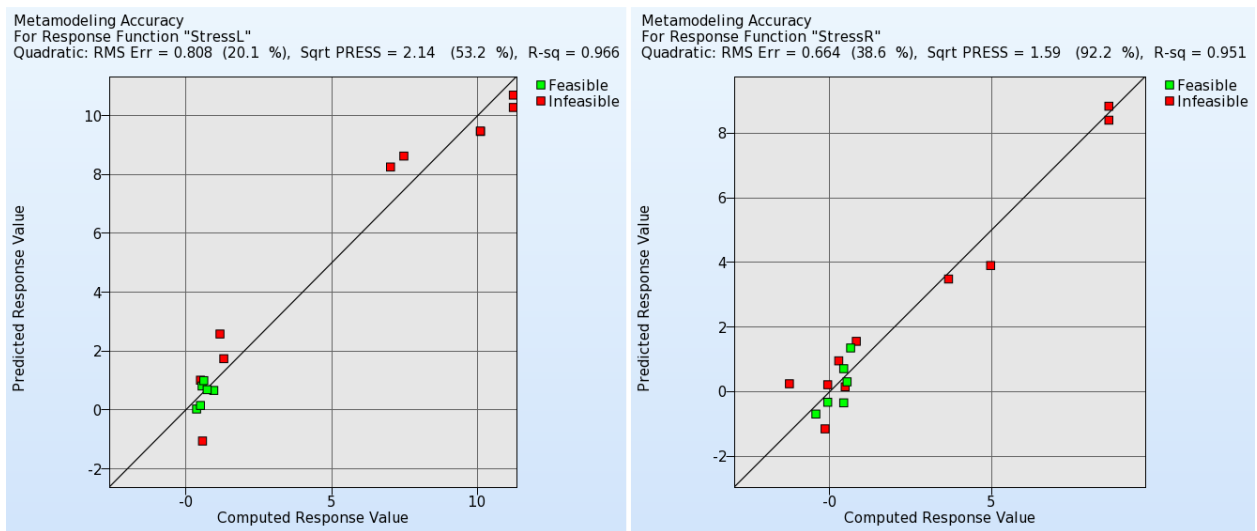
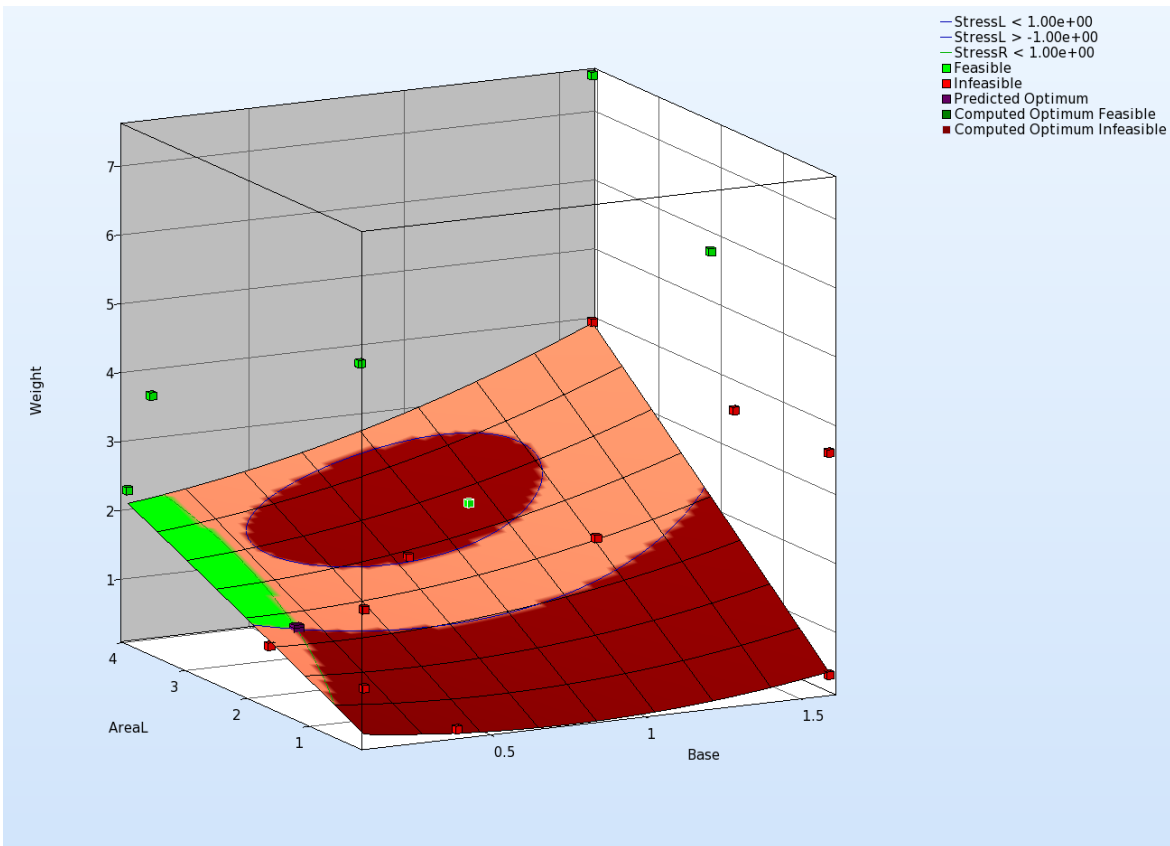


Figure 17-14: Accuracy of quadratic metamodel of responses "StressL" and "StressR"

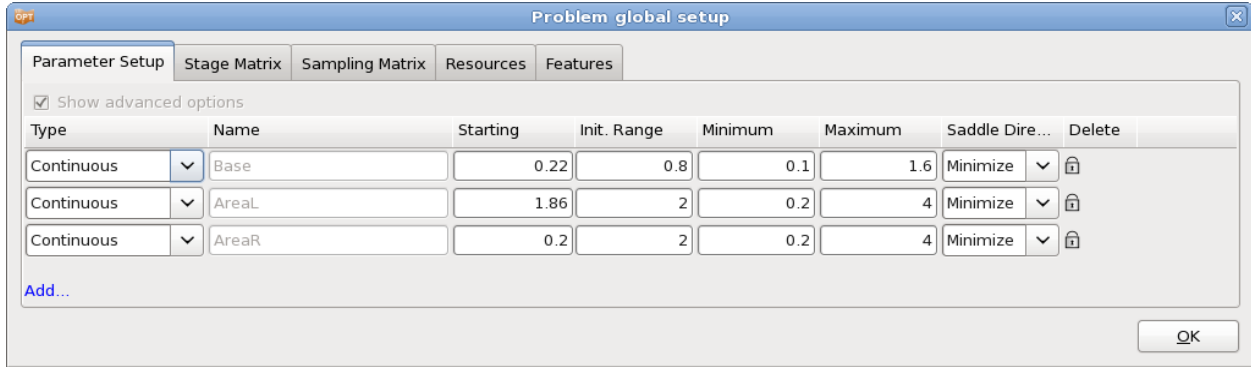
**Table 17-2: Comparison of baseline run and optimum (single iteration, quadratic metamodel)**

	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 17-15: Surface plot for objective function weight; constraints are displayed on the metamodel.**

#### 17.1.4. Reducing the region of interest for further refinement

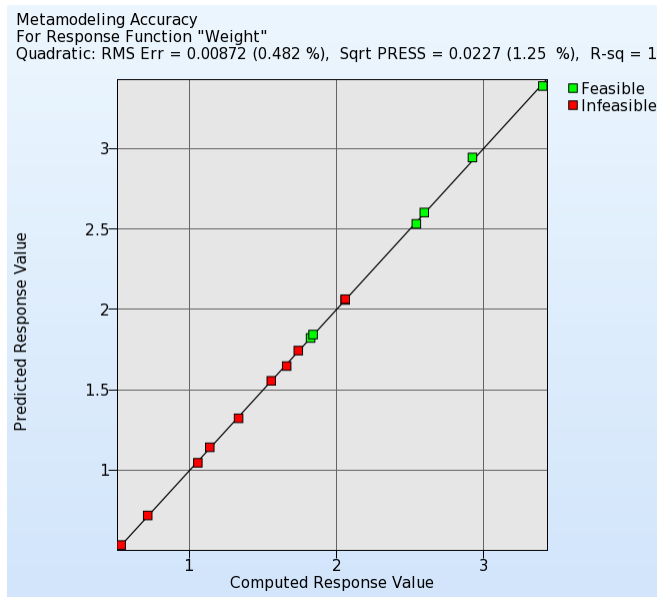
It seems that further accuracy can only be obtained by reducing the size of the subregion. In the following analysis, the current optimum (0.22, 1.86, 0.2) was used as a starting point while the region of interest was cut in half. The order of the approximation is quadratic. The required modifications are illustrated in Figure 17-16.



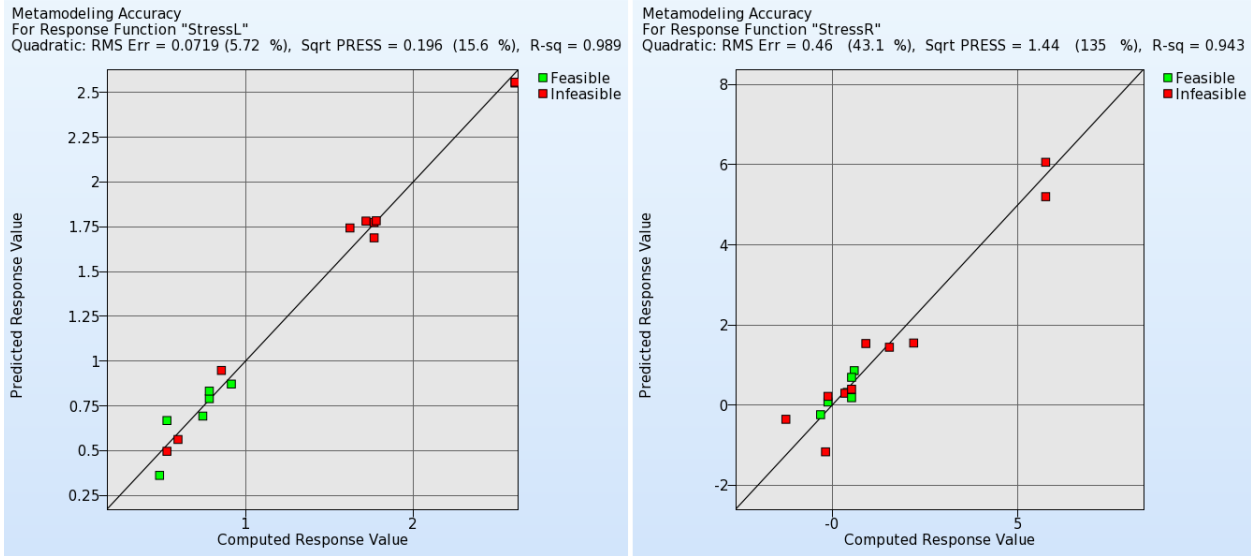
**Figure 17-16:** Reducing the design space by specifying an initial range; the starting values are the optimal values found in the previous approach.

## Results

The approximations are significantly improved, Figure 17-17 and Figure 17-18.



**Figure 17-17:** Accuracy of quadratic metamodel in reduced design space for response "Weight"



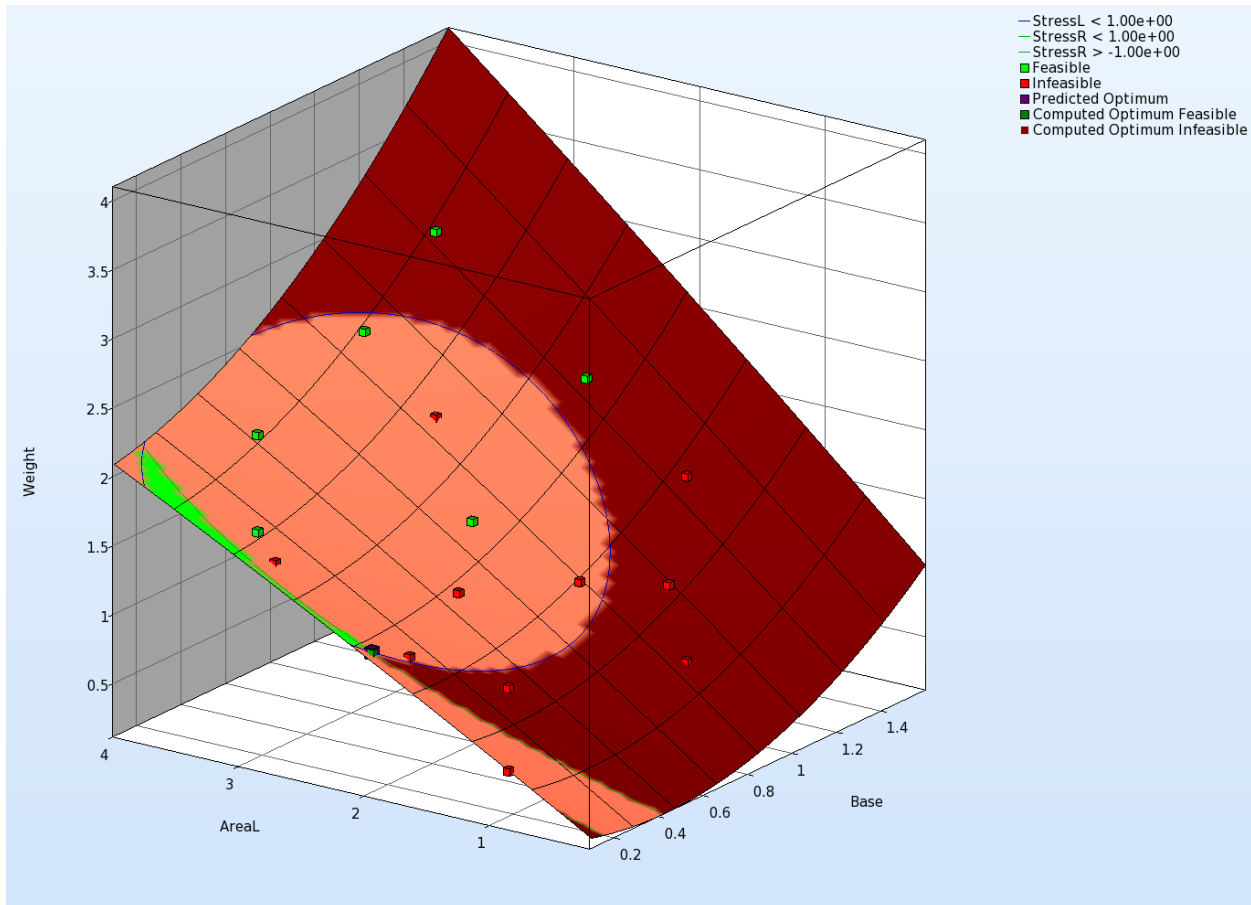
**Figure 17-18: Accuracy of quadratic metamodel in reduced design space of responses "StressL" and "StressR"**

The results are displayed in Table 17-3. 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 17-19.

**Table 17-3: Comparison of baseline run and optimum (single iteration, quadratic metamodel, reduced design space)**

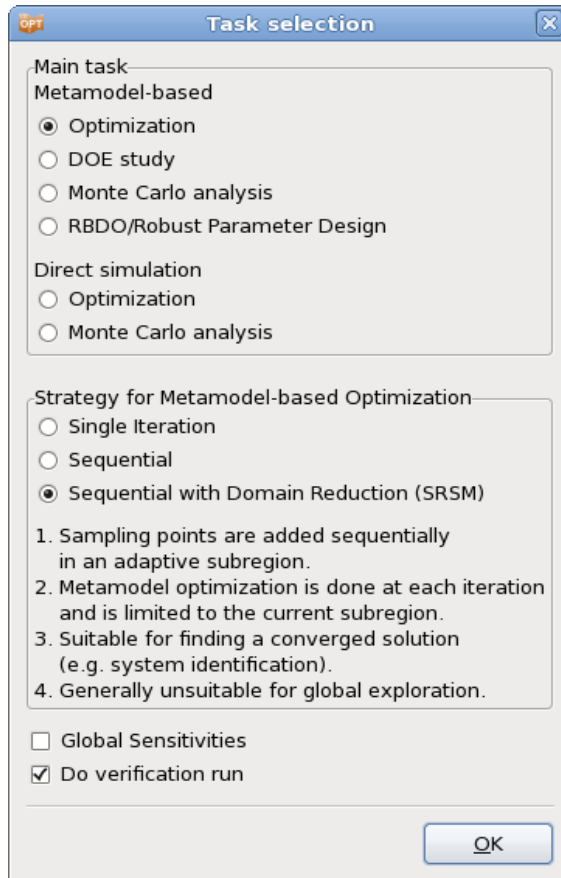
	Baseline (Computed)	Baseline (Predicted)	1. Opt (Computed)	1. Opt (Predicted)
Weight	1.05	1.04	1.12	1.12
StressL	0.86	0.95	0.96	1.00
StressR	2.19	1.55	0.38	1.00



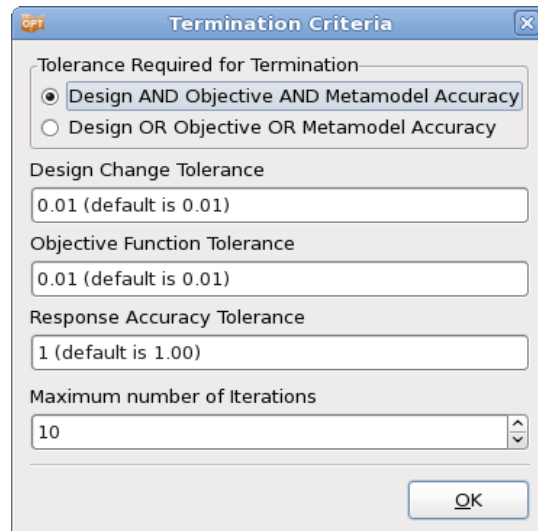
**Figure 17-19: Surface plot for objective function weight; constraints are displayed on the metamodel.**

### 17.1.5. 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 17-20. 10 iterations are performed for the linear approximation, Figure 17-21, with only 5 iterations performed for the more expensive quadratic approximation.



*Figure 17-20: Task dialog; select strategy SRSM to automate the process.*



*Figure 17-21: 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 17-4. The optimization histories have been plotted to illustrate convergence in Figure 17-22 and Figure 17-23. 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 17-4: 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



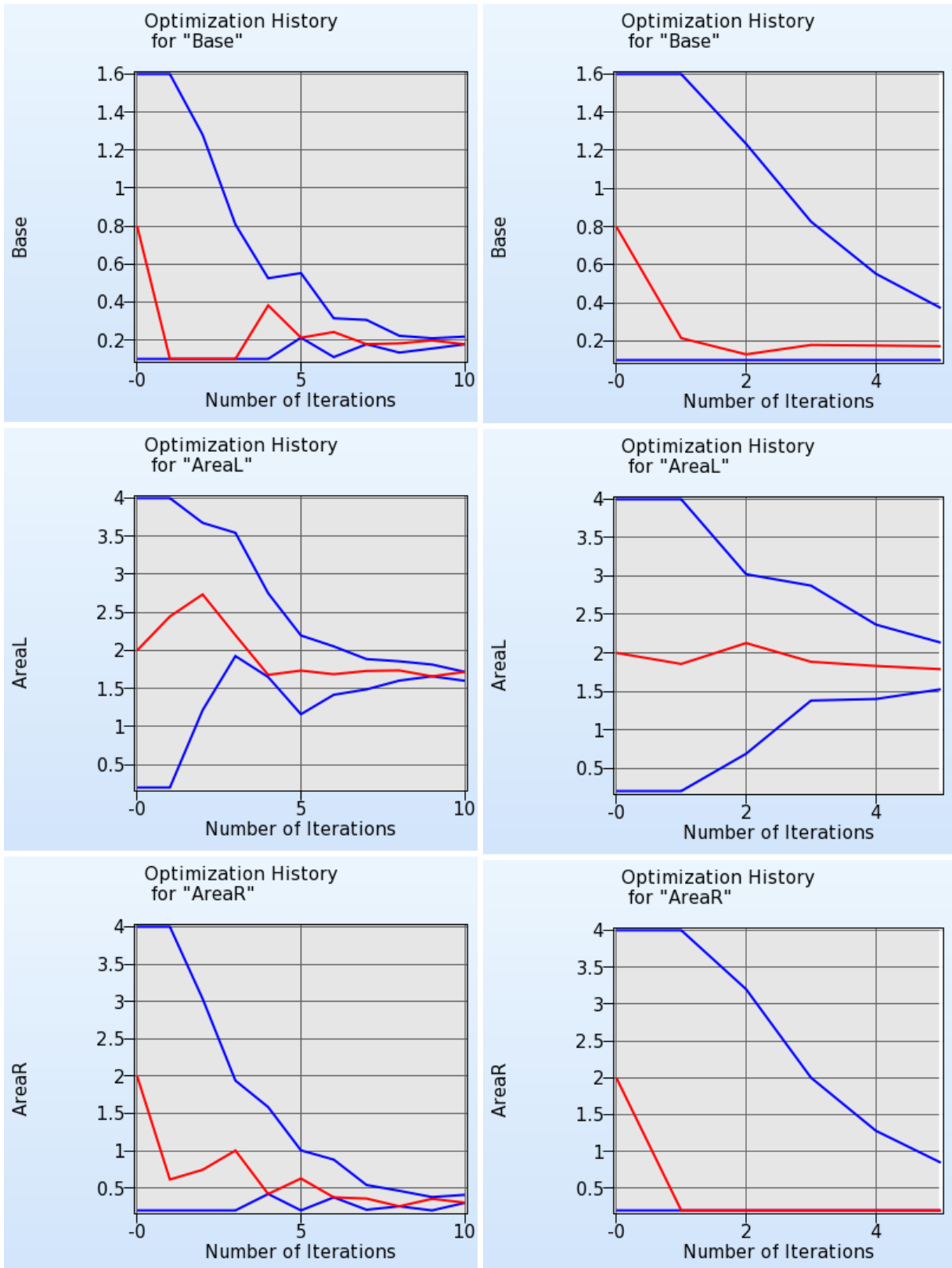


Figure 17-22: Optimization history of design variables; linear (left) and quadratic (right)

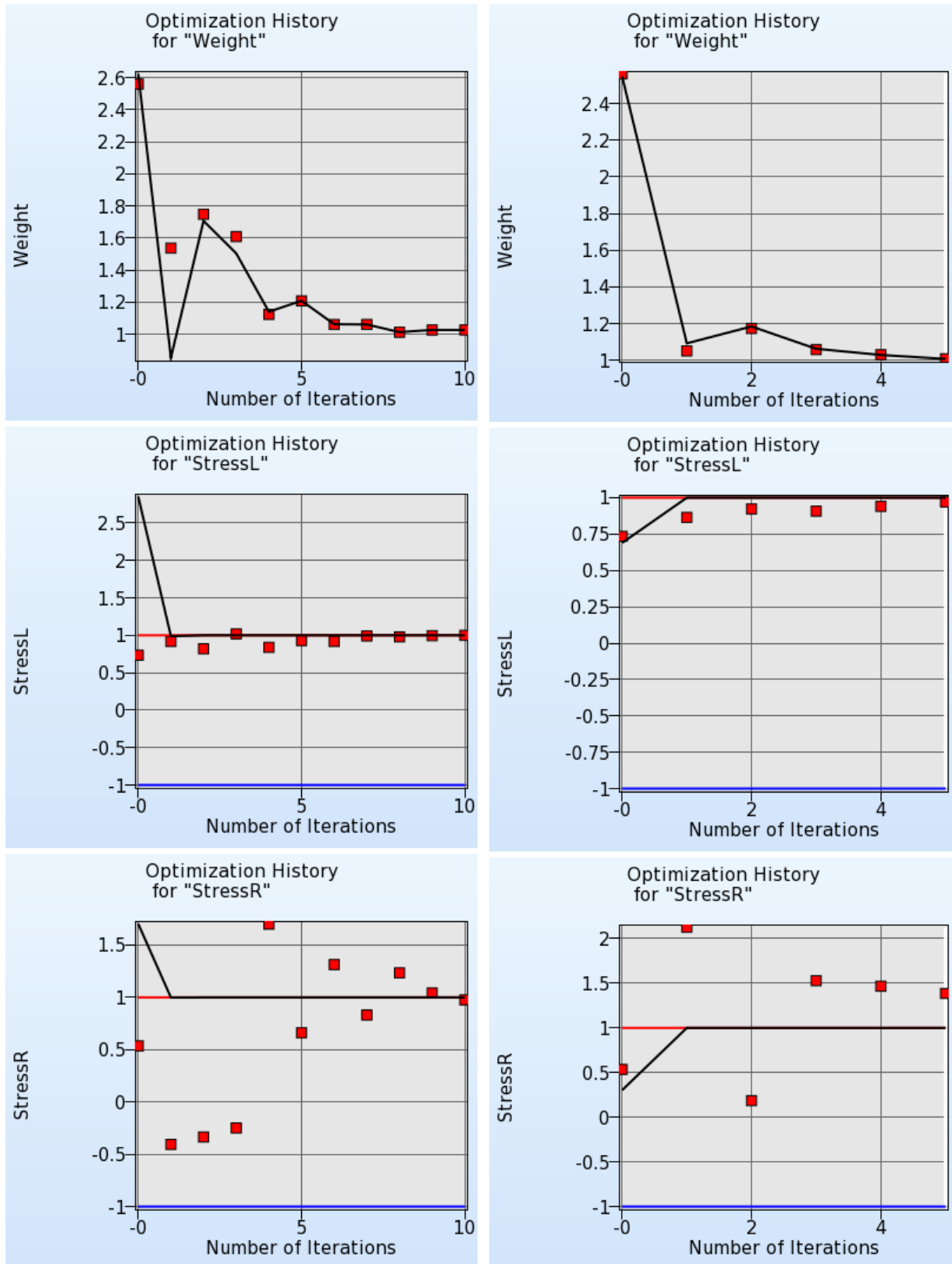


Figure 17-23: Optimization history of responses; linear (left) and quadratic (right)

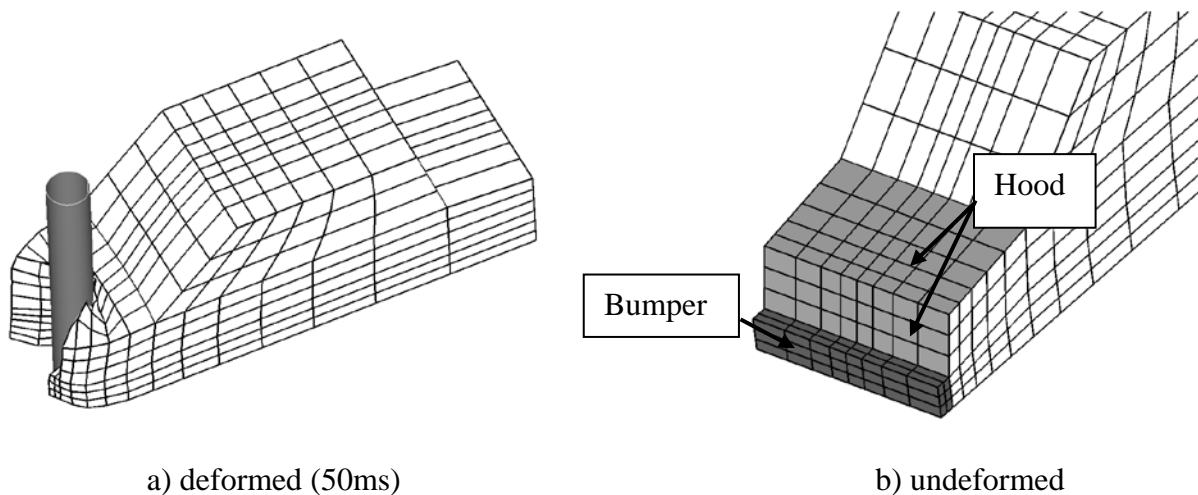
## 17.2. Small car crash (2 variables)

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.

### 17.2.1. Introduction

This example considers the crashworthiness of a simplified small car model. A simplified vehicle moving at a constant velocity of  $15.64\text{m}\cdot\text{s}^{-1}$  (35mph) impacts a rigid pole. See Figure 17-24. 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.



*Figure 17-24: Small car impacting a pole*

### 17.2.2. 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 17-24).

### 17.2.3. Design formulation

The design formulation is as follows:

Minimize

$$\text{HIC (15ms)} \quad (17-1)$$

subject to

$$\text{Intrusion (50ms)} < 550\text{mm}$$

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 17-25. These curves are evaluated at time  $t=50\text{ms}$  using response expressions. The intrusion is defined using a composite expression, Figure 17-27.

The mass is computed using the LS-DYNA MASS interface, Figure 17-26, but not constrained. This is useful for monitoring the mass changes.

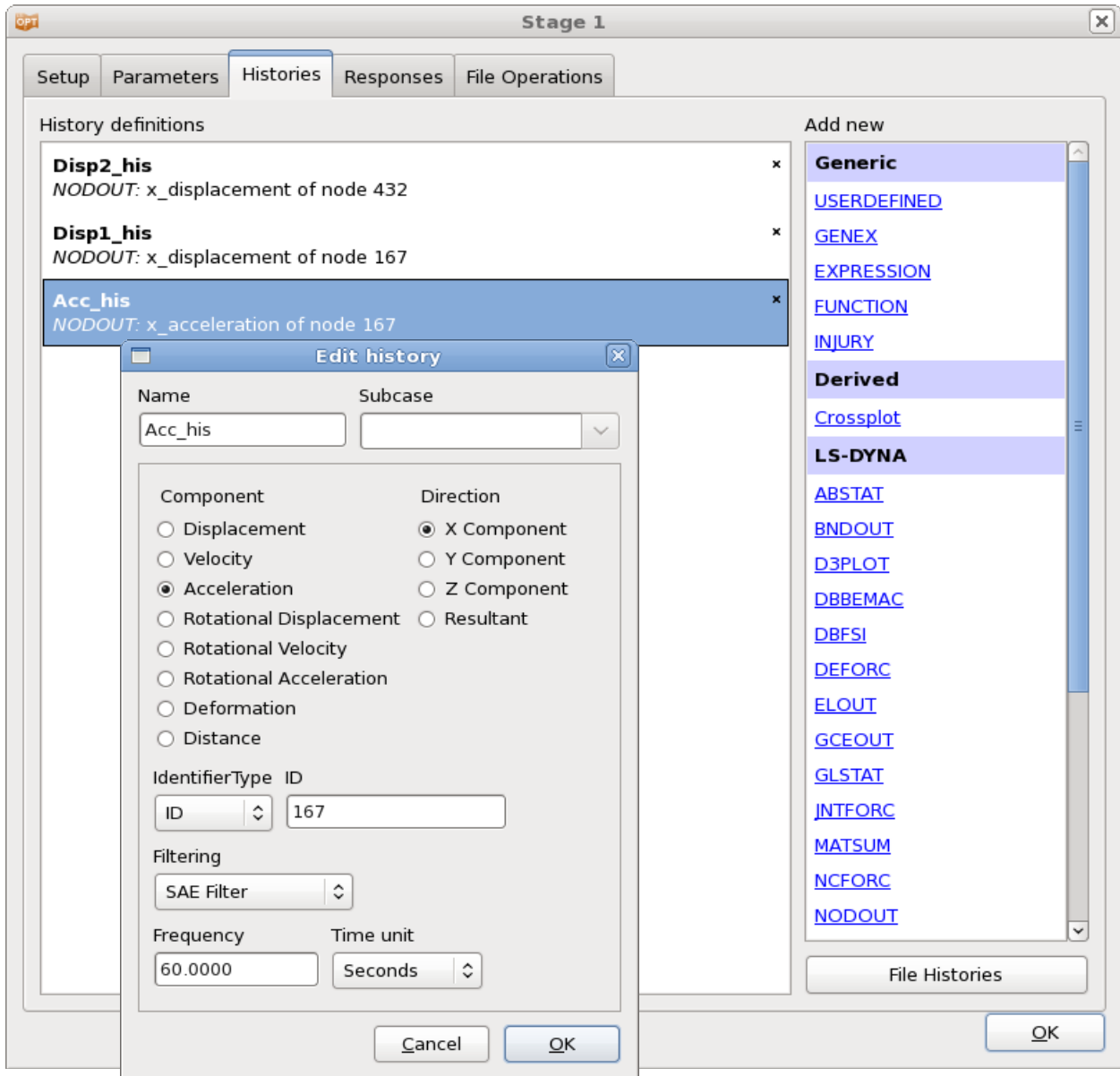


Figure 17-25: Definition of response histories using standard LS-DYNA interfaces.

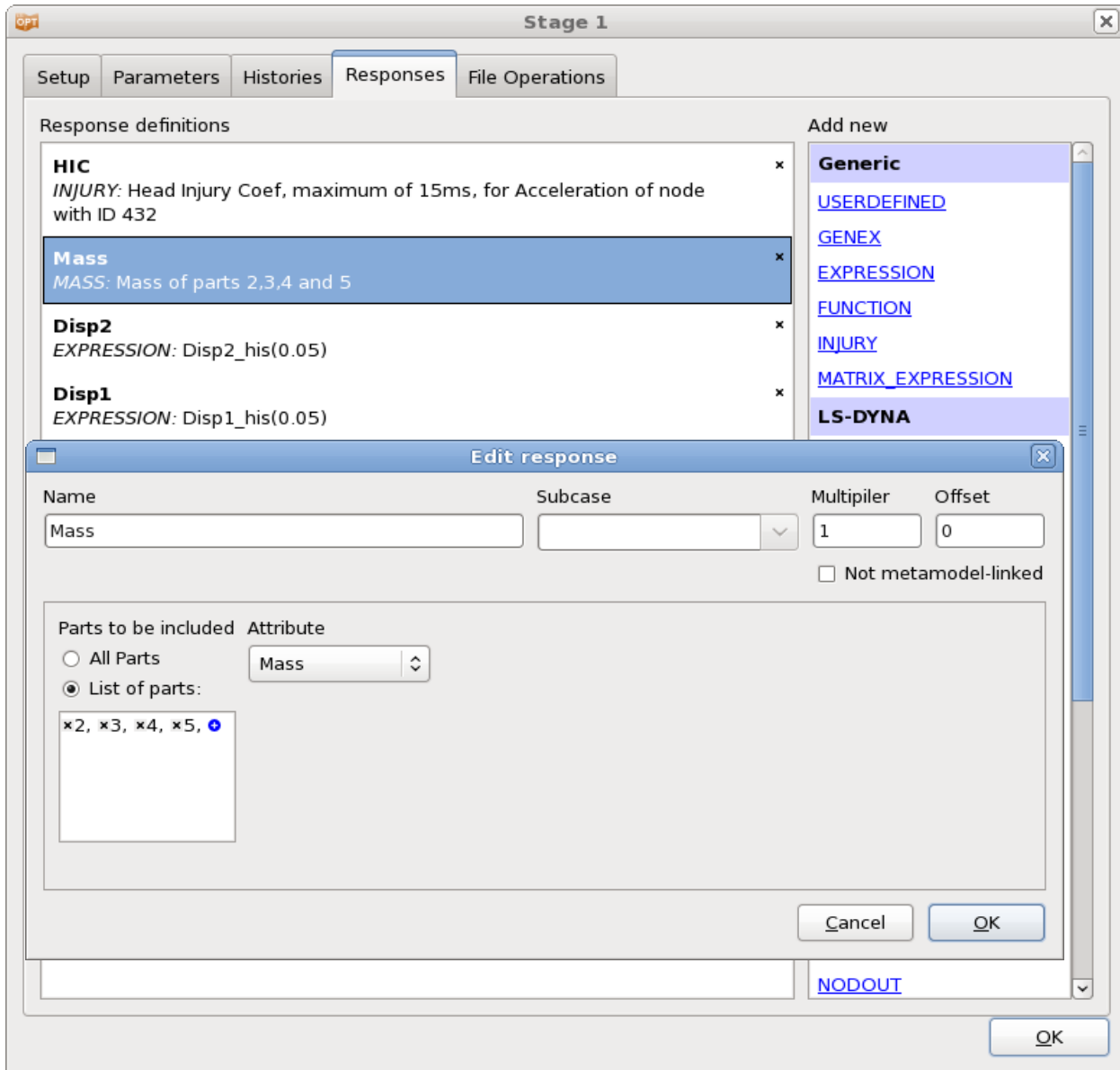
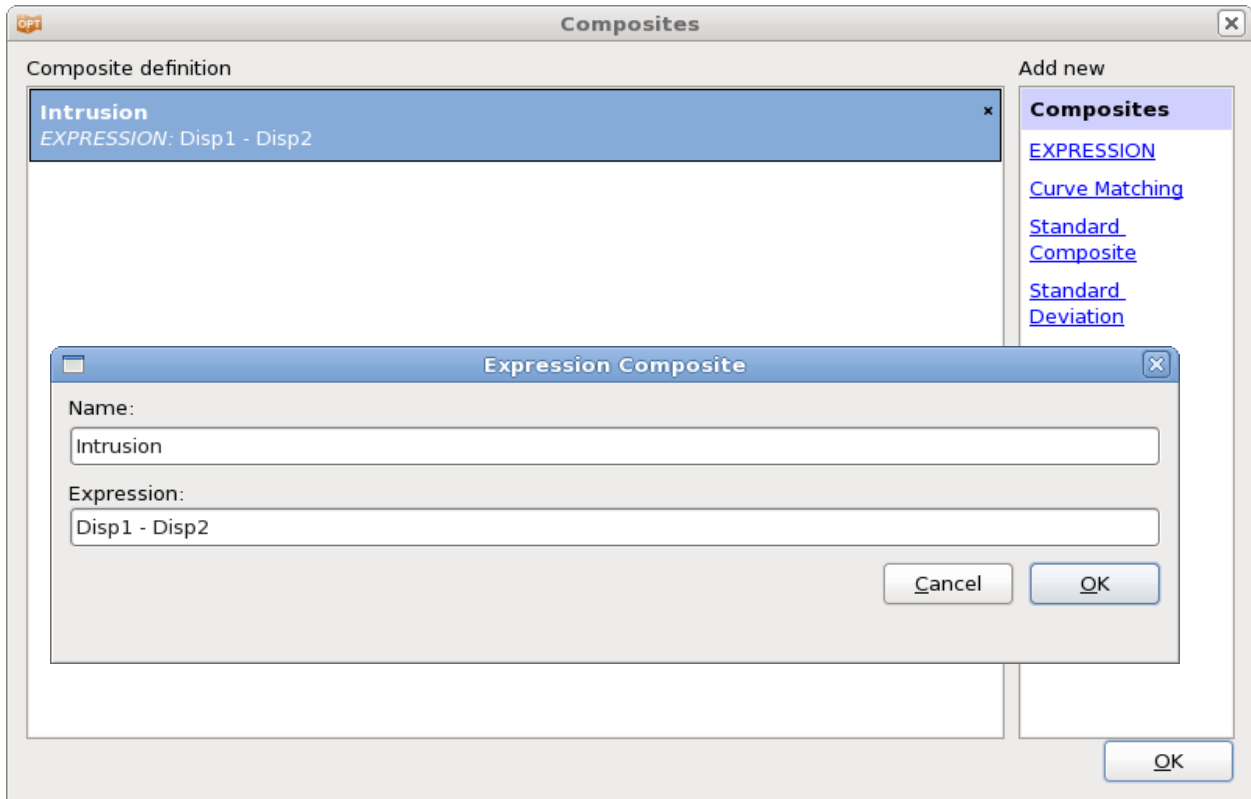


Figure 17-26: Definition of responses using standard LS-DYNA interfaces and expressions.

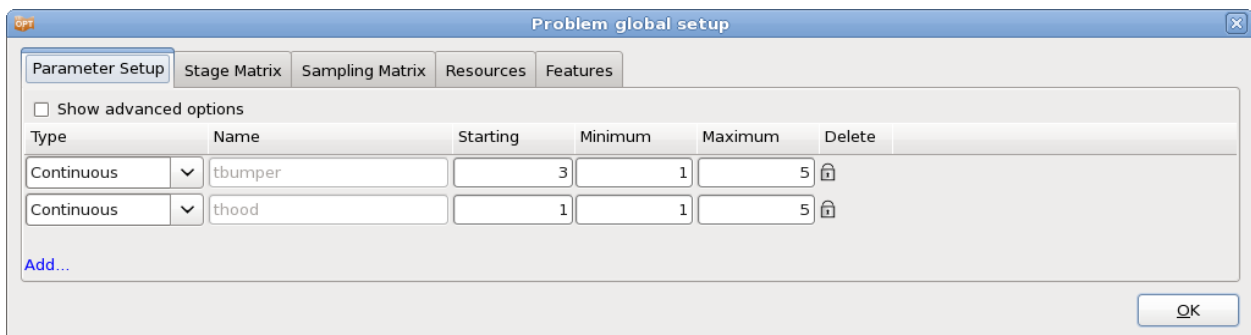


**Figure 17-27: Definition of composite expression using previously defined responses.**

### 17.2.4. 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
```

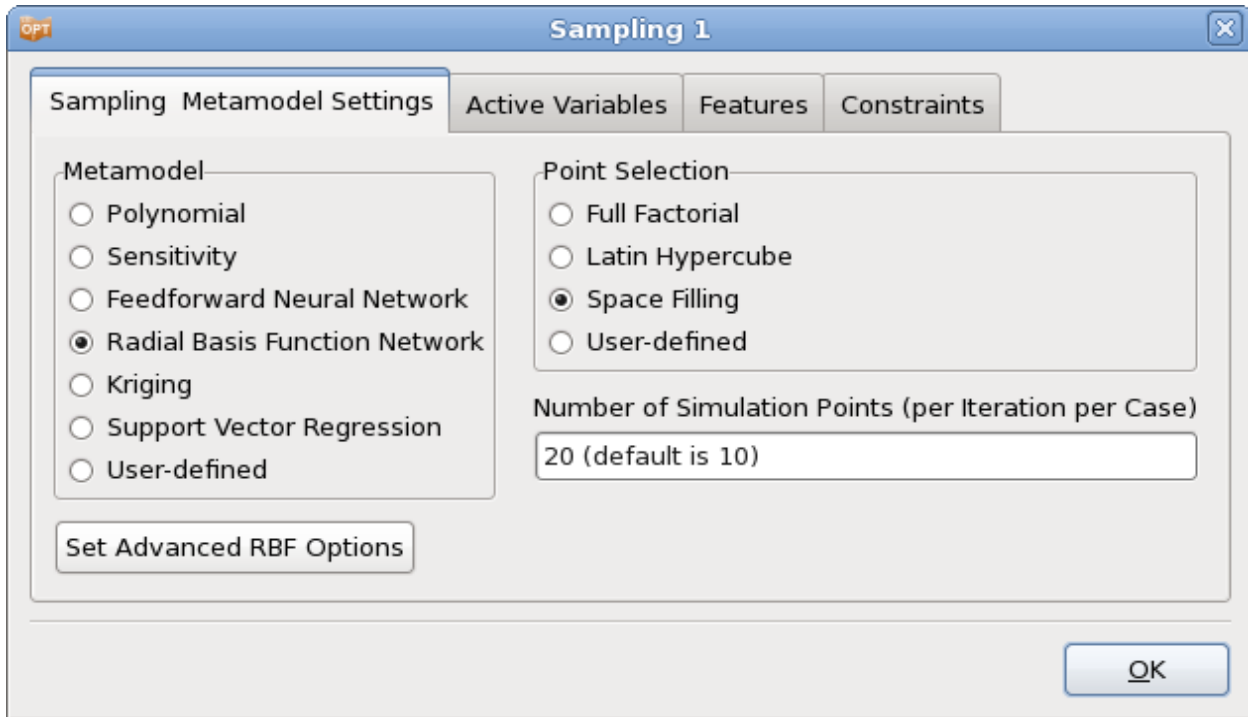


**Figure 17-28: Parameter Setup;**

A design space of [1; 5] is used for both design variables, Figure 17-28.

### 17.2.5. 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.



*Figure 17-29: 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 17-30. 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%.



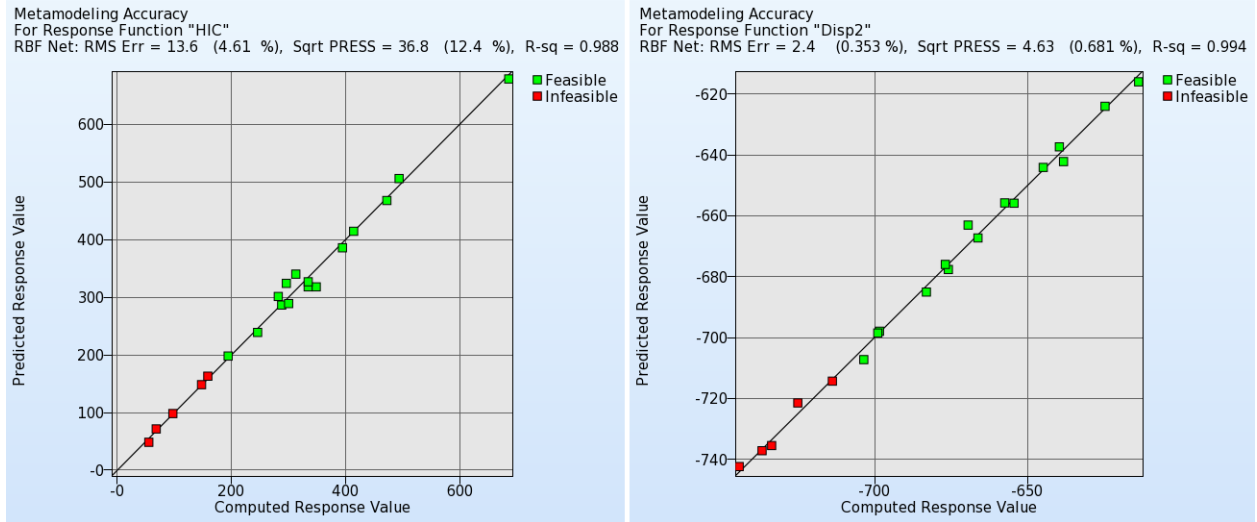


Figure 17-30: Computed vs. predicted responses – RBF approximation

Table 17-5: 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

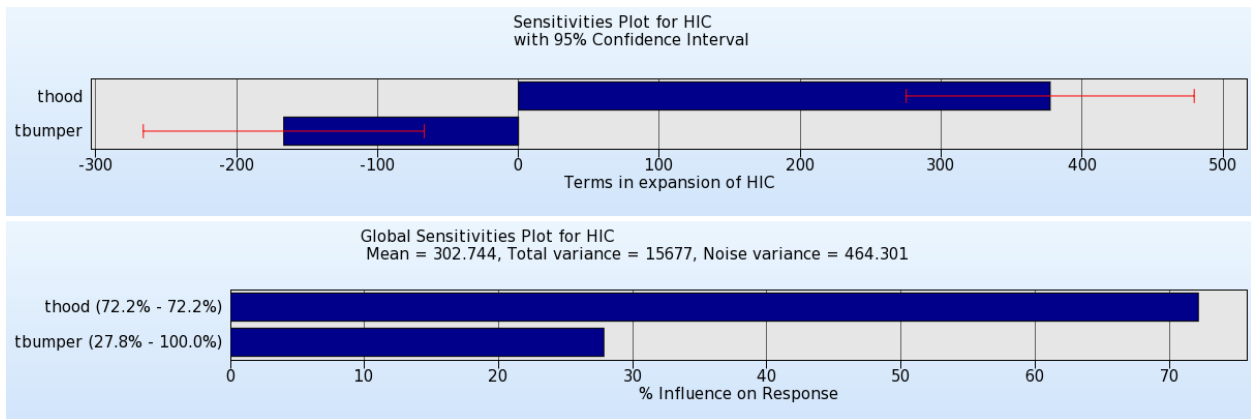
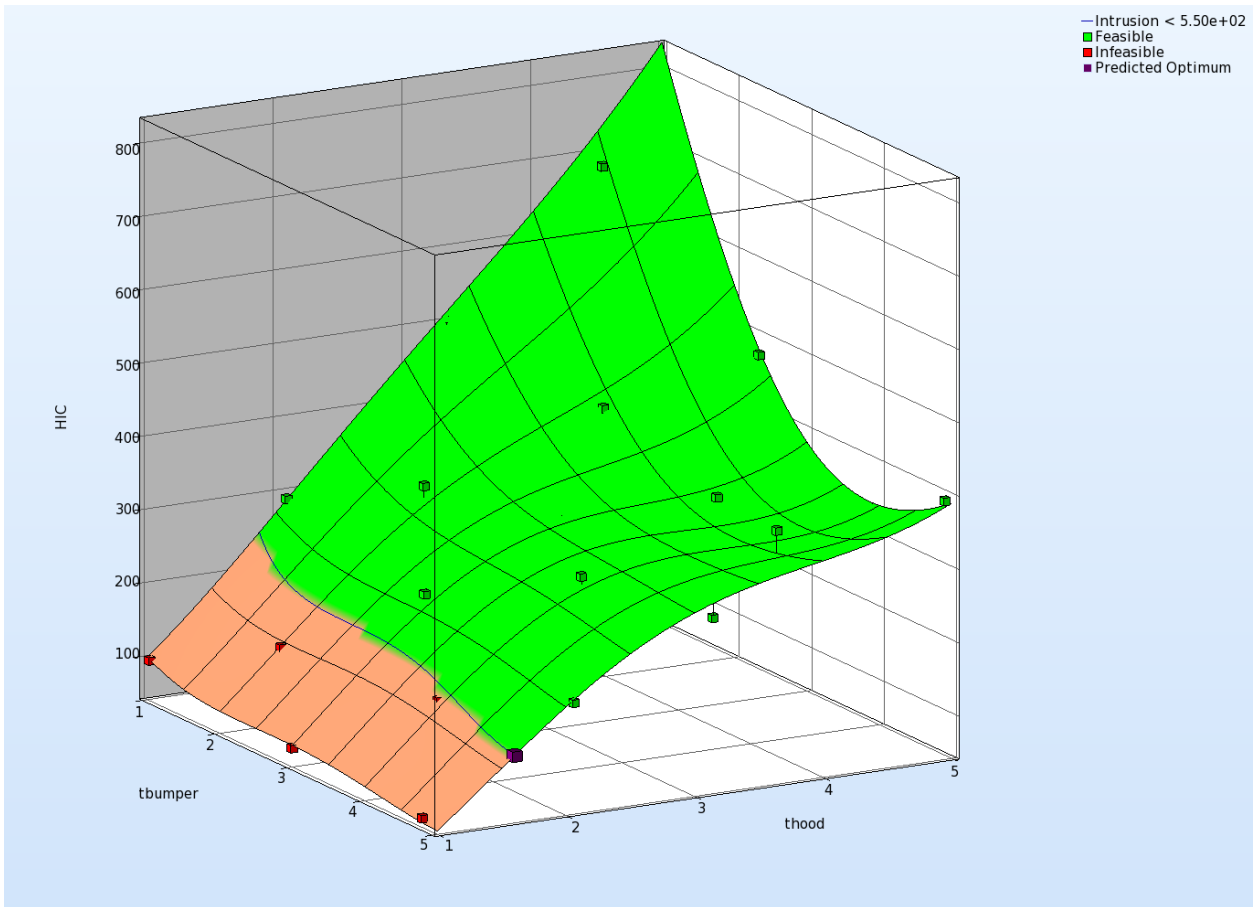
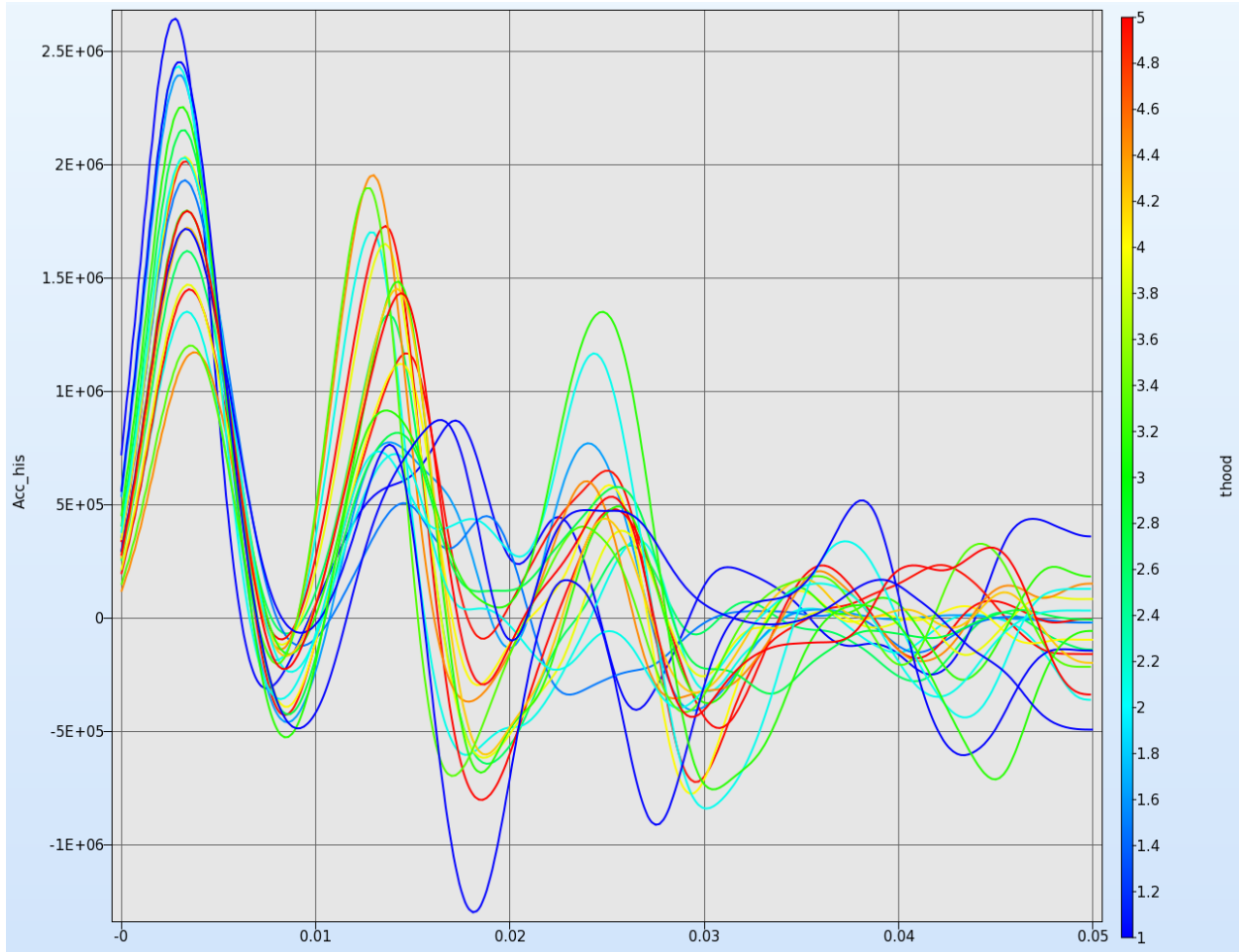


Figure 17-31: Sensitivities plots; ANOVA with 95% confidence interval (top) and GSA (bottom)



*Figure 17-32: Surface plot for objective function HIC with predicted and computed optimum, simulation points and residuals; constraints are displayed on the surface.*

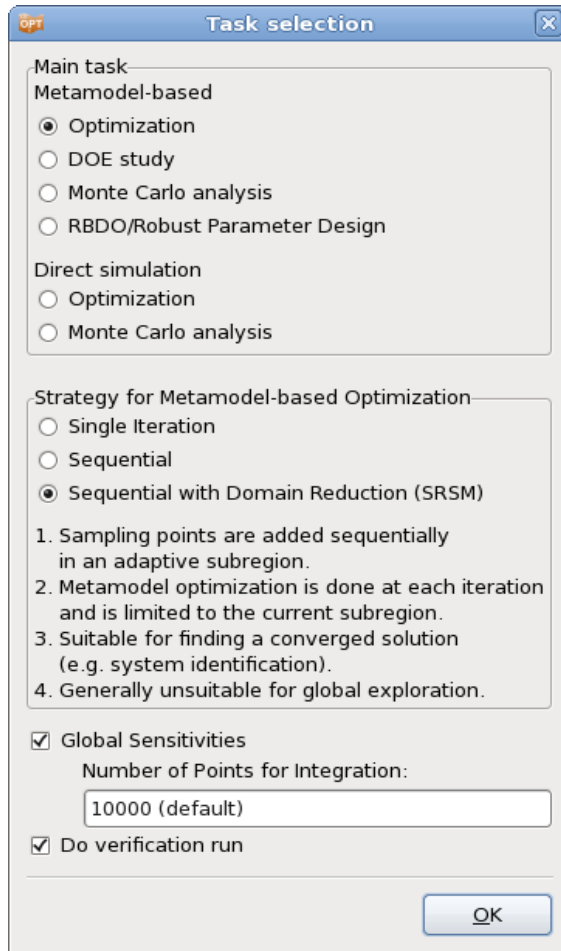


**Figure 17-33:** History plot for Acceleration; the curves are color-coded using the value of the variable *t\_hood*.

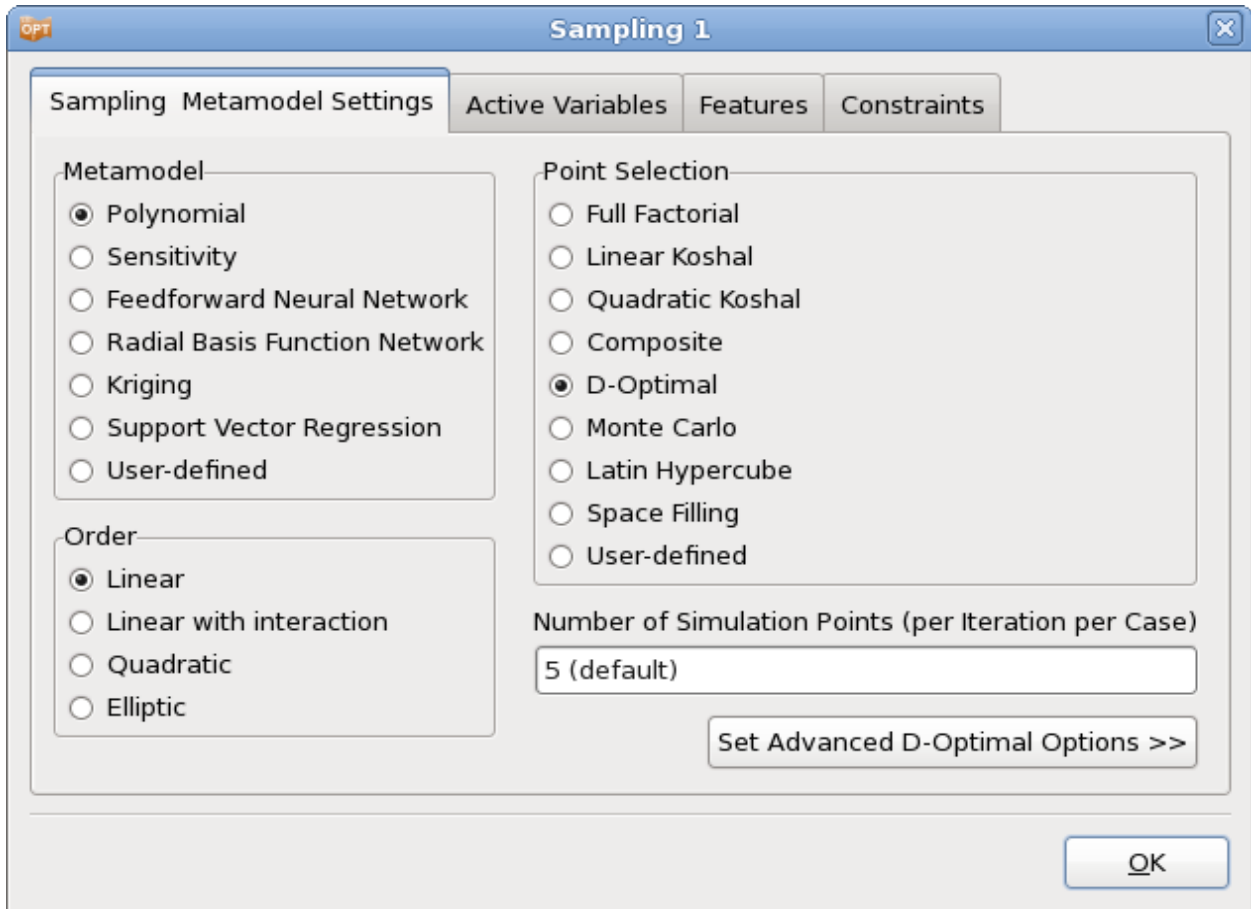
### 17.2.6. Automated run using linear metamodels

An automated optimization is performed with a linear approximation. Select the strategy *Sequential with domain reduction*, Figure 17-34, and switch to the metamodel type *Polynomial linear*, Figure 17-35. Use the default number of points per iteration per case.

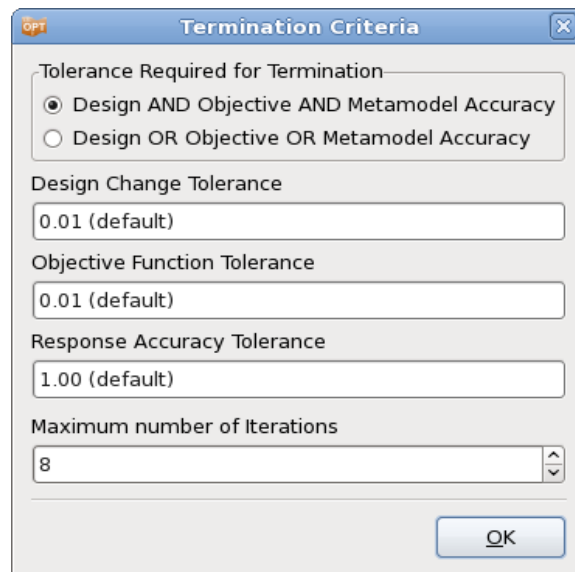
In the **Termination Criteria** dialog, set the maximum number of iterations to 8, Figure 17-36.



*Figure 17-34: Task dialog; select Strategy Sequential with Domain Reduction*



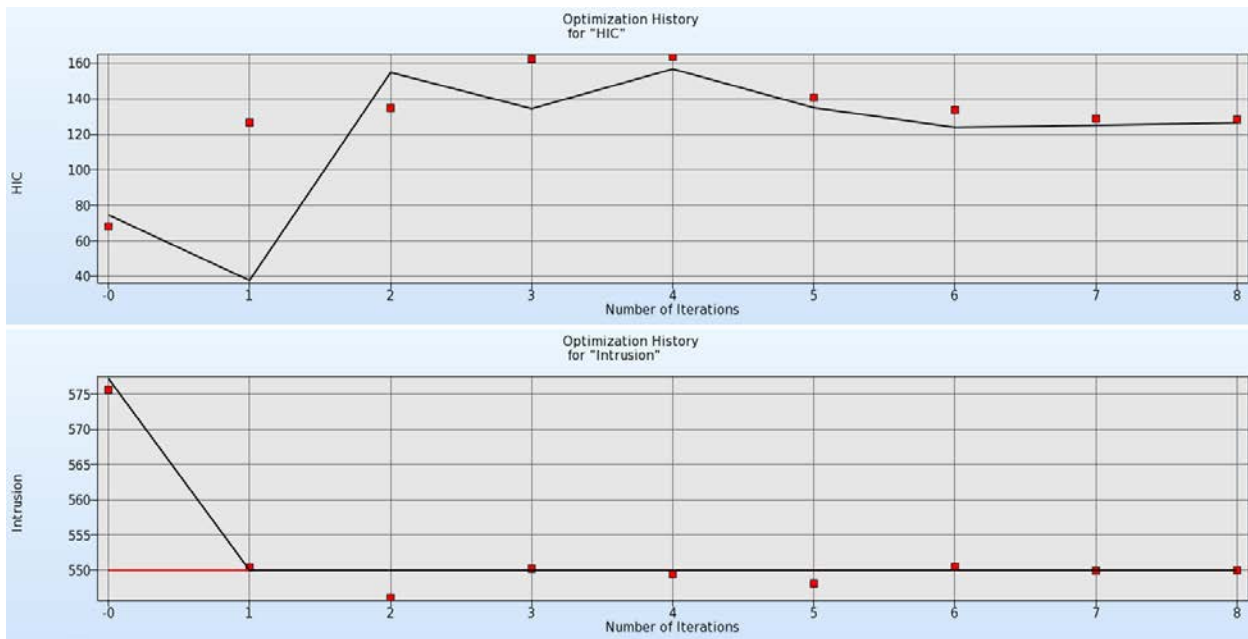
**Figure 17-35: Sampling Dialog; use the default settings for SRSM for metamodel type and order, point selection scheme and number of points**



**Figure 17-36: Termination Criteria dialog; select the maximum number of iterations**

## Results

It can be seen in Figure 17-37 that the objective function (HIC) and intrusion constraint are approximately optimized at the 7<sup>th</sup> 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  $t_{hood}$  never moves to the edge of the subregion during the optimization process, the heuristic in LS-OPT enforces pure zooming (see Figure 17-38). For  $t_{bumper}$ , panning occurs as well due to the fact that the linear approximation predicts a variable on the edge of the subregion.



*Figure 17-37: Optimization history of HIC and Intrusion*

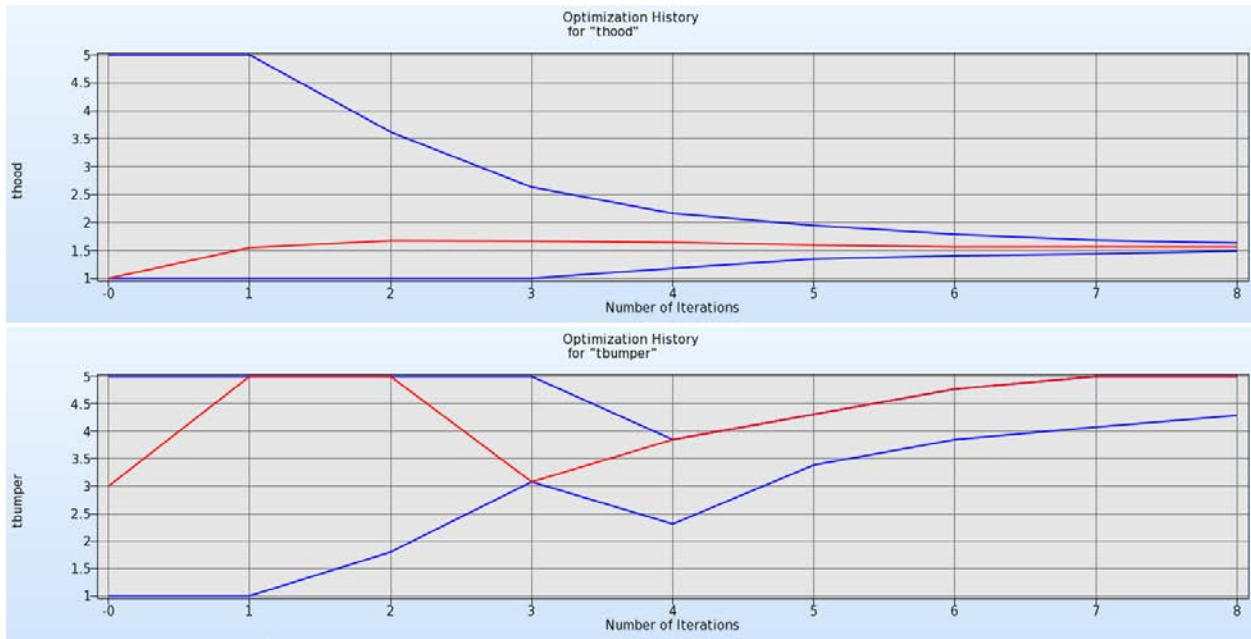


Figure 17-38: Optimization history of design variables

### 17.2.7. 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 17-39.

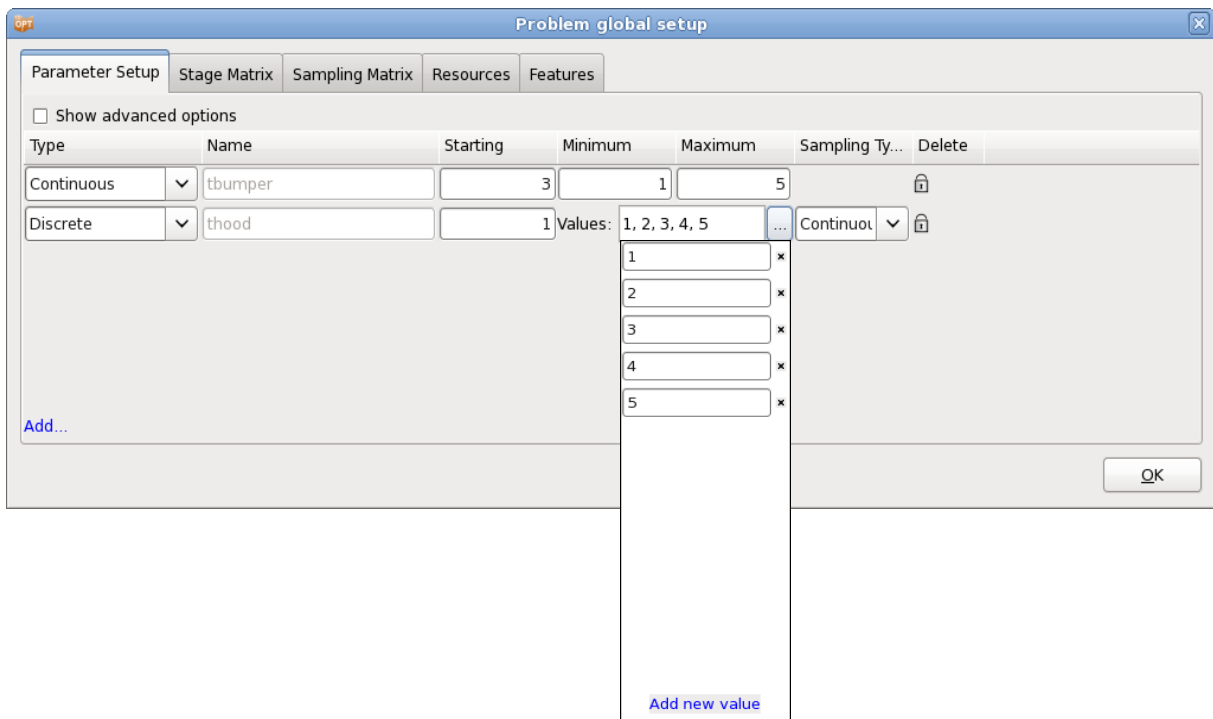
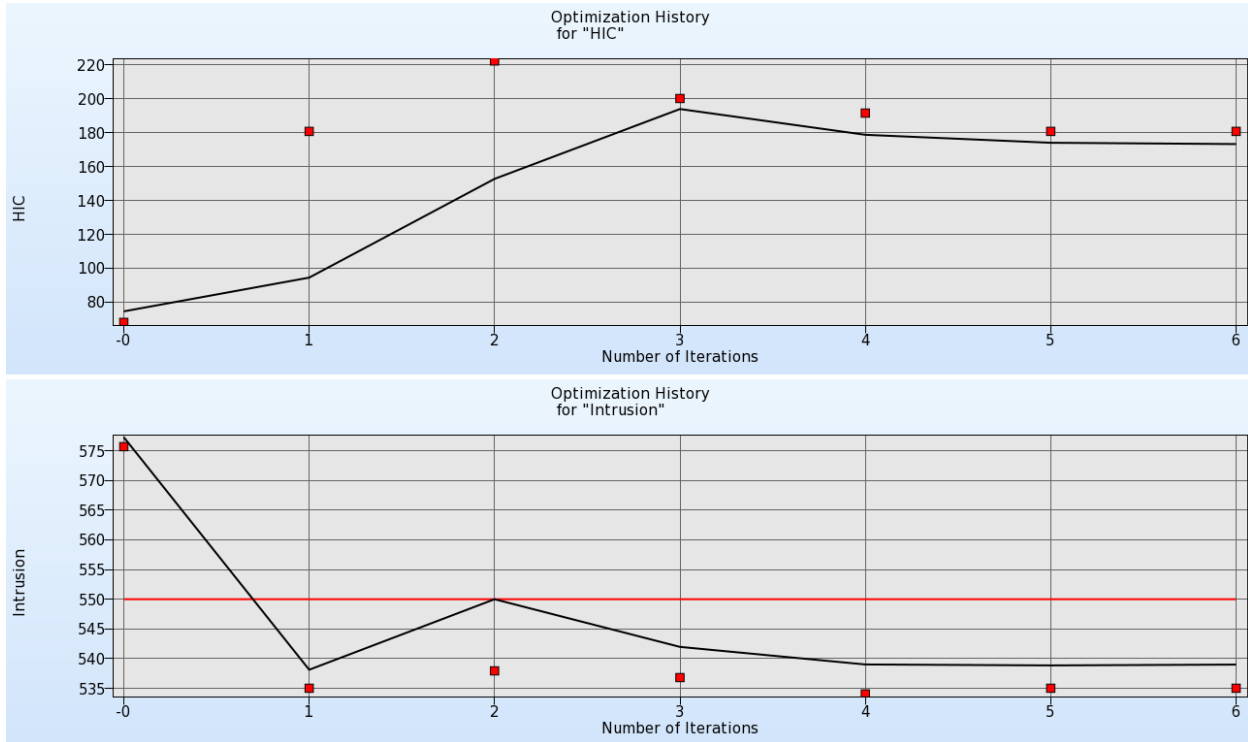


Figure 17-39: Parameter Setup dialog; Definition of a discrete variable.

## Results

The design variables histories are shown in Figure 17-41, the optimization histories for the objective HIC and the constraint Intrusion in Figure 17-40.



*Figure 17-40: Optimization history of HIC and Intrusion for mixed-discrete optimization*



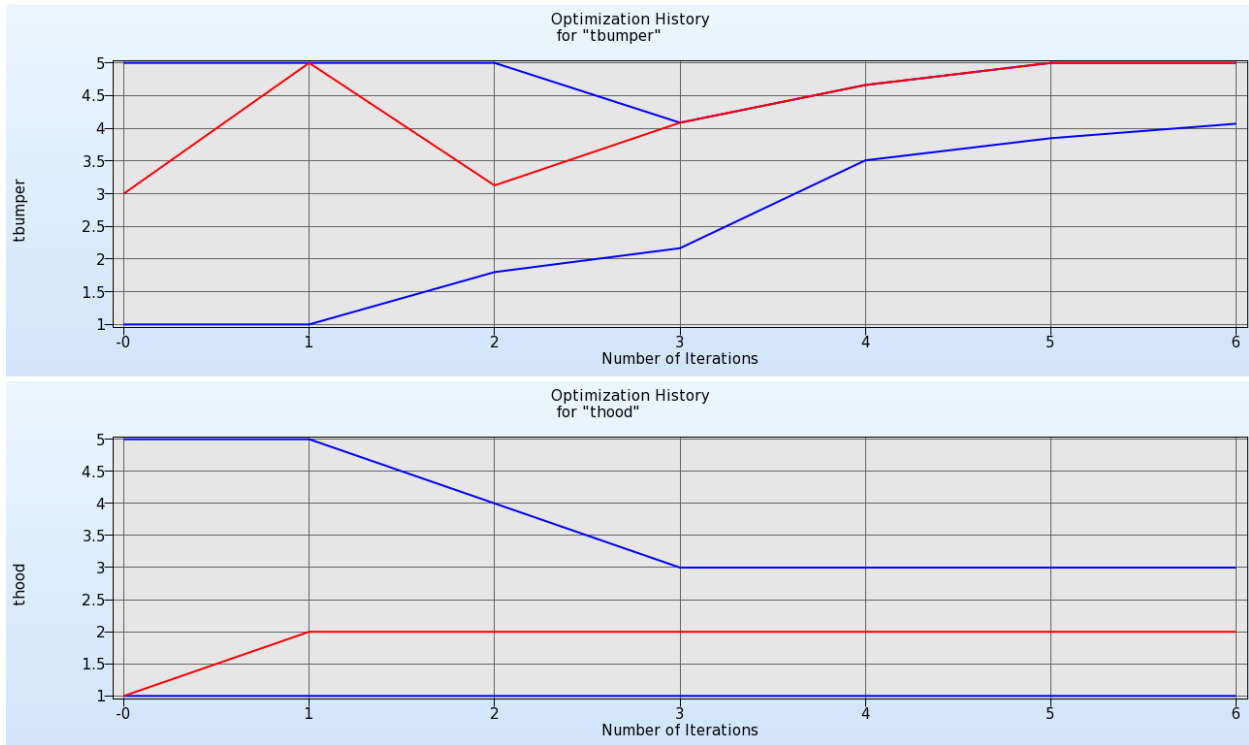
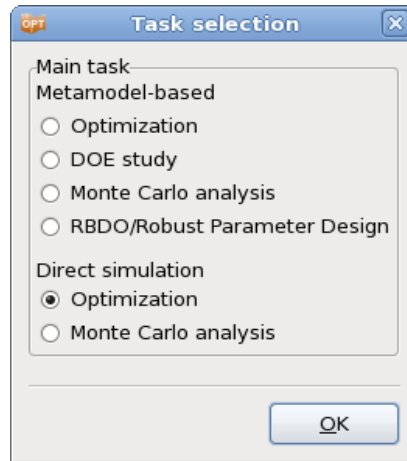


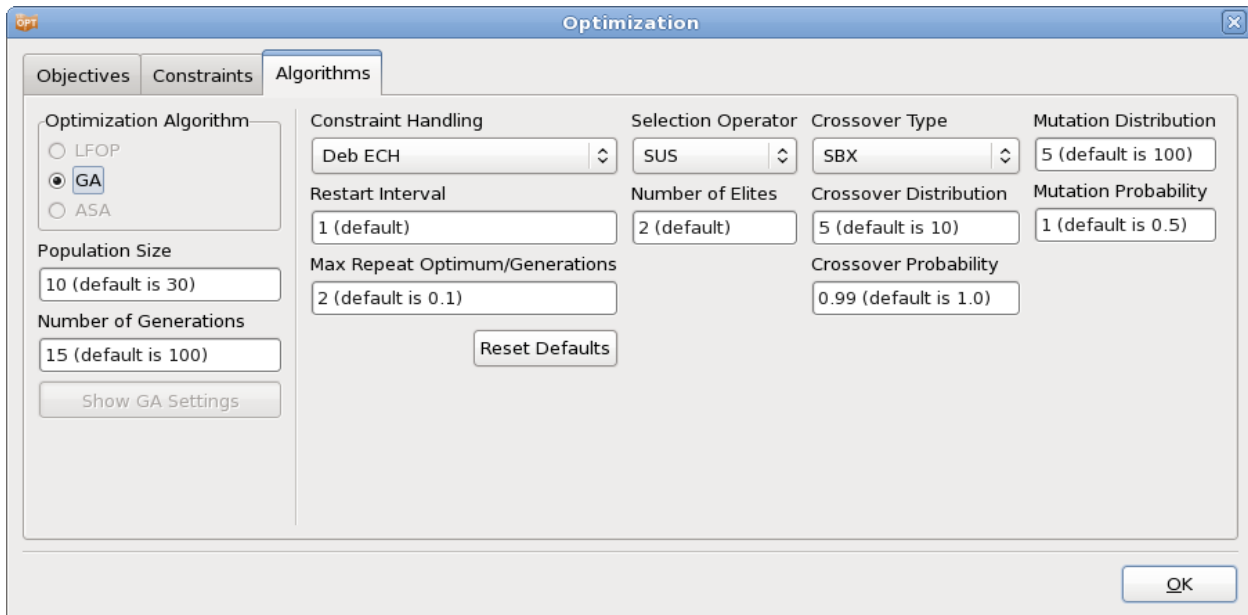
Figure 17-41: Mixed-discrete variable histories.

### 17.2.8. Optimization using Direct GA simulation

The same problem is solved using a direct GA simulation, Figure 17-42. GA specific settings and advanced options may be selected in the Optimization dialog, Figure 17-43. 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.



*Figure 17-42: Task dialog; Direct Genetic algorithm*



*Figure 17-43: Optimization dialog; Specification of advanced GA options*

## Results

The outcome of the optimization is shown in Figure 17-44 and Figure 17-45. 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.

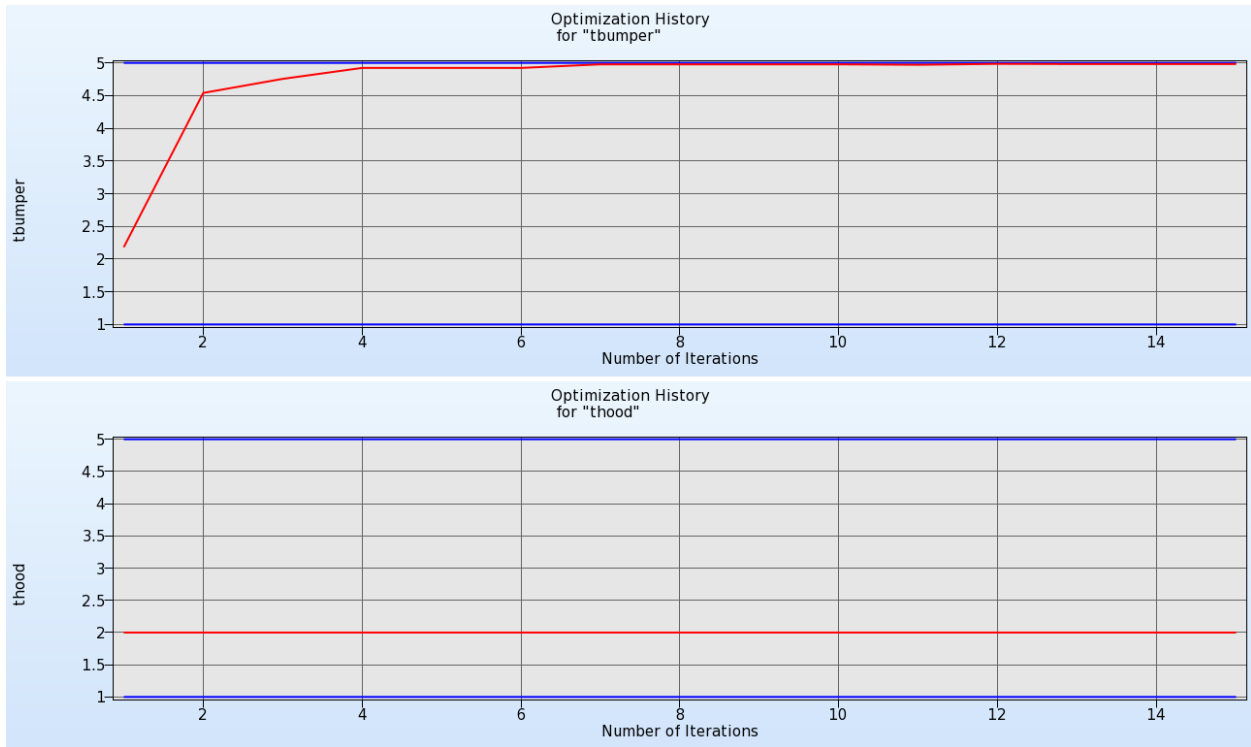


Figure 17-44: Optimization history of mixed-discrete variable optimization using direct GA simulation.

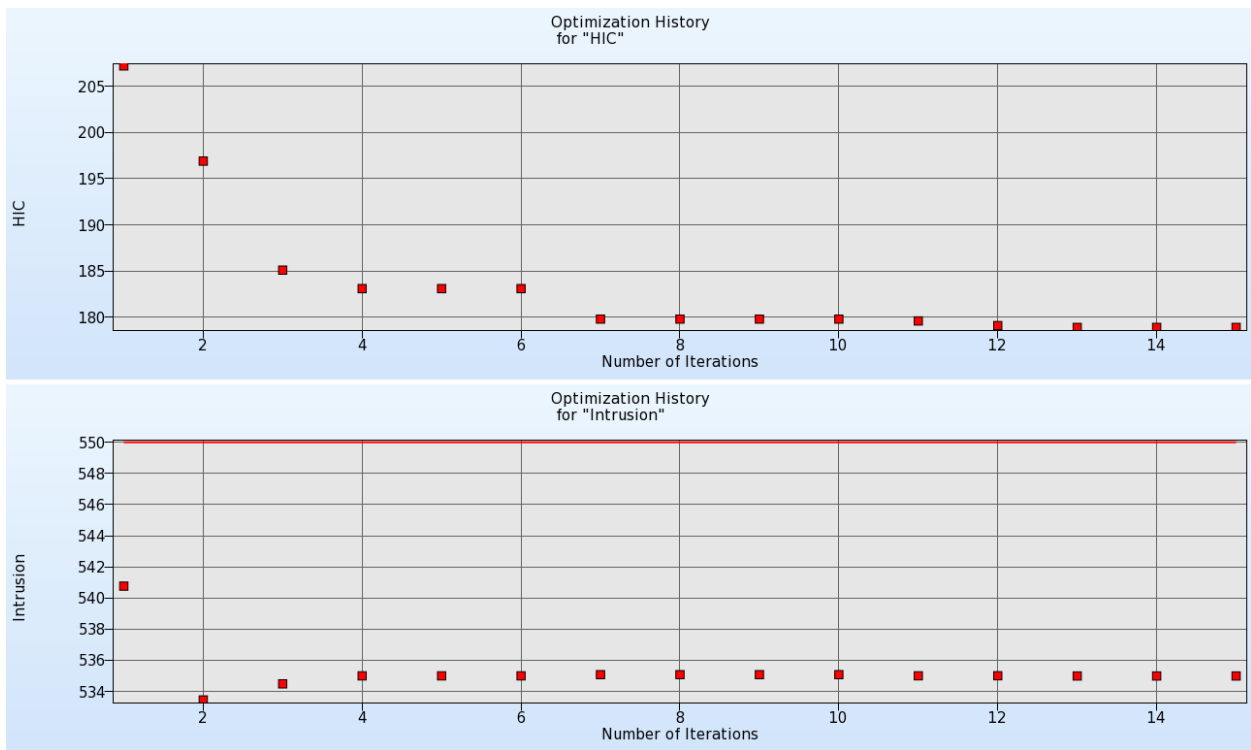


Figure 17-45: Optimization history of HIC and Intrusion

### 17.3. Impact of a cylinder (2 variables)

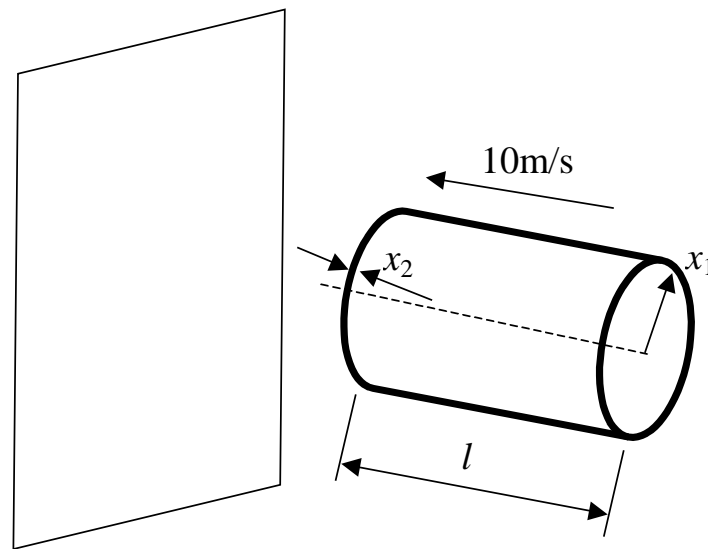
This example has the following features:

- An LS-DYNA explicit impact simulation is performed.
- An independent parametric preprocessor is used to incorporate shape optimization.
- Extraction is performed using standard ASCII LS-DYNA interfaces.
- Second-order response surface approximations are compared using different subregions.
- The design optimization process is automated.
- Noisy response variables are improved using filtering.

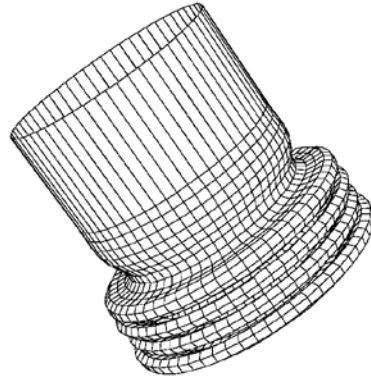
The example in this chapter is modeled on one by Yamazaki [1].

#### 17.3.1. Problem statement

The problem consists of a tube impacting a rigid wall as shown in Figure 17-46. The energy absorbed is maximized subject to a constraint on the rigid wall impact force. The cylinder has a constant mass of 0.54 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 17-47 for a typical design.



*Figure 17-46: Impacting cylinder*



**Figure 17-47: Deformed finite element model (time = 20ms)**

The optimization problem is stated as:

$$\text{Maximize } E_{\text{internal}}(x_1, x_2) \Big|_{t=0.02}$$

subject to

$$F_{\text{normal}}^{\text{wall}}(x_1, x_2) \Big|_{\text{average}} \leq 70000$$

$$l(x) = \frac{0.52}{2\pi\rho x_1 x_2}$$

where the design variables  $x_1$  and  $x_2$  are the radius and the thickness of the cylinder respectively.  $E_{\text{internal}}(x) \Big|_{t=0.02}$  is the objective function and constraint functions  $F_{\text{normal}}^{\text{wall}}(x) \Big|_{\text{average}}$  and  $l(x)$  are the average normal force on the rigid wall and the length of the cylinder, respectively.

The problem is simulated using LS-DYNA. The following TrueGrid input file including the `<<name>>` statements is used to create the FE input deck with the FE model as shown in Figure 17-47. Note that the design variables have been scaled.

```
c cyl2 - crush cylinder - constant volume
lsdyna3d keyword
lsdyopts secforc .00002 rwforc .00002 ;
lsdyopts endtim .02 d3plot dtcycl .0001 ; ;
lsdyopts thkchg 2 ;
lsdyopts elout 0.001
lsdyopts glstat 0.001
lsdymats 1 3 rho 2880 shell elfor bt tsti 4
    e 71.38e9 pr .33 sigy 102.0e6 etan 0.2855e9 ;
lsdymats 2 20 rho 14.3e6 e 7.138e10 pr .33 cmo con 4 7 shell elfor bt tsti 4;
para
r [<<Radius>>/1000.0]
l [3.0e+1/<<Radius>>/<<Wall_Thickness>>]
h [<<Wall_Thickness>>/1000.0]
l2 [75.0/<<Radius>>*0.02]
h2 .002
v0 10.
```

```

n .33
pi 3.14159
;
plane 1 0 0 -.002 0 0 1 .001 ston pen 2. stick ;
sid 1 ldsdi 13 slvmat 1;scoef .4 dcoef .4 sfsp 1.5 ; ; ;
c ***** part 1 mat 1 ***** shell
cylinder
-1; 1 60; 1 50 51;
%r
0 360
0 %1 [%12+%1]
dom 1 1 1 1 2 3
  x=x+.01*%h*sin(%pi*z*57.3/(%pi*(%r*%r*%h*%h/(12*(1-%n*%n)))**.25))
thick %h
thi ;;2 3; %h2
c bi ; ;-3 0 -3; dx 1 dy 1 rx 1 ry 1 rz 1 ;
c interrupt
swi ;; ;1
velocity 0 0 [-%v0]
mate 1
mti ;; 2 3; 2
c element spring block
epb 1 1 1 1 2 3
endpart
merge
stp .000001
write
end

```

### 17.3.2. A first approximation

In the first iteration, a quadratic approximation is chosen from the beginning. The ASCII database is suitable for this analysis as the energy and impact force can be extracted from the `glstat` and `rwforc` databases respectively. Five processors are available. The region of interest is arbitrarily chosen to be about half the size of the design space.

The following LS-OPT command input deck was used to find the approximate optimum solution:

```

"Cylinder Impact Problem"
$ Created on Thu Jul 11 11:37:33 2002
$
$ DESIGN VARIABLES
$
variables 2
  Variable 'Radius' 75
  Lower bound variable 'Radius' 20
  Upper bound variable 'Radius' 100
  Range 'Radius' 50
  Variable 'Wall_Thickness' 3
  Lower bound variable 'Wall_Thickness' 2
  Upper bound variable 'Wall_Thickness' 6
  Range 'Wall_Thickness' 2
solvers 1
responses 2
$
$ NO HISTORIES ARE DEFINED

```

```

$
$
$ DEFINITION OF SOLVER "RUN1"
$
  solver dyna960 'RUN1'
  solver command "lsdyna"
  solver input file "trugrdo"
  prepro truegrid
  prepro command "/net/src/ultra4_4/common/hp/tg2.1/tg"
  prepro input file "cyl2"
$
$ RESPONSES FOR SOLVER "RUN1"
$
  response 'Internal_Energy' 1 0 "DynaASCII Glstat I_Ener 0 Timestep"
  response 'Internal_Energy' quadratic
  response 'Rigid_Wall_Force' 1 0 "DynaASCII rwforc normal 1 ave"
  response 'Rigid_Wall_Force' quadratic
$
$ NO HISTORIES DEFINED FOR SOLVER "RUN1"
$
$
$ OBJECTIVE FUNCTIONS
$
  objectives 1
  maximize
  objective 'Internal_Energy' 1
$
$ CONSTRAINT DEFINITIONS
$
  constraints 1
  constraint 'Rigid_Wall_Force'
  upper bound constraint 'Rigid_Wall_Force' 70000
$
$ EXPERIMENTAL DESIGN
$
  Order quadratic
  Experimental design dopt
  Basis experiment 5toK
  Number experiment 10
$
$ JOB INFO
$
  concurrent jobs 5
  iterate param design 0.01
  iterate param objective 0.01
  iterate 1
STOP

```

The curve-fitting results below show that the internal energy is approximated reasonably well whereas the average force is poorly approximated. The accuracy plots confirm this result (Figure 17-48).

Approximating Response 'Internal\_Energy' using 10 points (ITERATION 1)

-----  
 Global error parameters of response surface  
 -----

Quadratic Function Approximation:  
 -----

Mean response value = 10686.0081

```

RMS error           = 790.3291 (7.40%)
Maximum Residual    = 1538.9208 (14.40%)
Average Error       = 654.4415 (6.12%)
Square Root PRESS Residual = 2213.7994 (20.72%)
Variance            = 1249240.2552
R^2                 = 0.9166
R^2 (adjusted)     = 0.9166
R^2 (prediction)   = 0.3453
Determinant of [X]'[X] = 1.3973
Approximating Response 'Rigid_Wall_Force' using 10 points (ITERATION 1)

```

-----  
Global error parameters of response surface  
-----

Quadratic Function Approximation:  
-----

```

Mean response value      = 121662.9474
RMS error                = 24730.1732 (20.33%)
Maximum Residual        = 48569.4162 (39.92%)
Average Error           = 21111.3307 (17.35%)
Square Root PRESS Residual = 75619.5531 (62.15%)
Variance                = 1223162932.2092
R^2                     = 0.8138
R^2 (adjusted)          = 0.8138
R^2 (prediction)        = -0.7406
Determinant of [X]'[X]  = 1.3973

```

The initial design below shows that the constraint is severely exceeded.

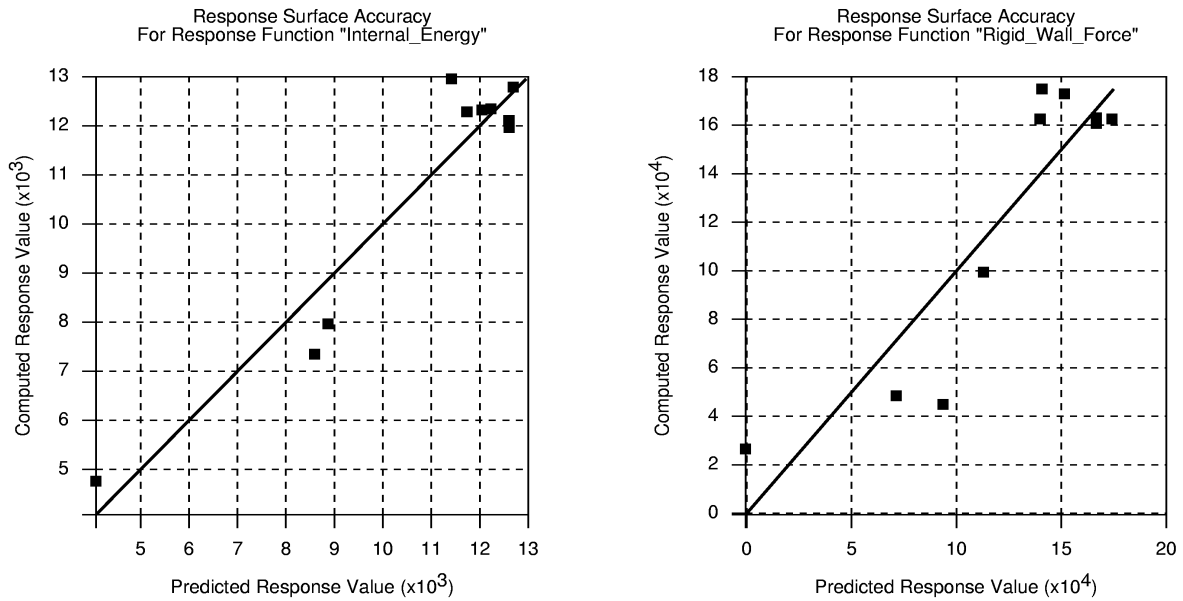
DESIGN POINT  
-----

Variable Name	Lower Bound	Value	Upper Bound
Radius	20	75	100
Wall_Thickness	2	3	6

RESPONSE FUNCTIONS:  
-----

RESPONSE	Scaled		Unscaled	
	Computed	Predicted	Computed	Predicted
Internal_Energy	1.296e+04	1.142e+04	1.296e+04	1.142e+04
Rigid_Wall_Force	1.749e+05	1.407e+05	1.749e+05	1.407e+05





**Figure 17-48: Prediction accuracy of Internal Energy and Rigid Wall Force (One Quadratic iteration)**

Despite the relatively poor approximation a prediction of the optimum is made based on the approximation response surface. The results are shown below. The fact that the optimal Radius is on the lower bound of the subregion specified (Range = 50), suggests an optimal value below 50.

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound
Radius	20	50	100
Wall_Thickness	2	2.978	6

RESPONSE FUNCTIONS:

RESPONSE	Scaled		Unscaled	
	Computed	Predicted	Computed	Predicted
Internal_Energy	7914	8778	7914	8778
Rigid_Wall_Force	4.789e+04	7e+04	4.789e+04	7e+04

### 17.3.3. Refining the design model using a second iteration

During the previous optimization step, the Radius variable was reduced from 75 to 50 (on the boundary of the region of interest). It was also apparent that the approximations were fairly inaccurate. Therefore, in the new iteration, the region of interest is reduced from [50;2] to [35;1.5] while retaining a quadratic approximation order. The starting point is taken as the current optimum: (50,2.978). The modified commands in the input file are as follows:

```
$
$ DESIGN VARIABLES
```

```

$
variables 2
Variable 'Radius' 50
  Lower bound variable 'Radius' 20
  Upper bound variable 'Radius' 100
  Range 'Radius' 35
Variable 'Wall_Thickness' 2.9783
  Lower bound variable 'Wall_Thickness' 2
  Upper bound variable 'Wall_Thickness' 6
  Range 'Wall_Thickness' 1.5

```

As shown below, the accuracy of fit improves but the average rigid wall force is still inaccurate.

Approximating Response 'Internal\_Energy' using 10 points (ITERATION 1)

```

-----
                Global error parameters of response surface
-----
Quadratic Function Approximation:
-----
Mean response value           = 8640.2050
RMS error                     = 526.9459 (6.10%)
Maximum Residual              = 890.0759 (10.30%)
Average Error                  = 388.4472 (4.50%)
Square Root PRESS Residual    = 1339.4046 (15.50%)
Variance                      = 555344.0180
R^2                           = 0.9632
R^2 (adjusted)                = 0.9632
R^2 (prediction)              = 0.7622
Determinant of [X]'[X]        = 0.0556
Approximating Response 'Rigid_Wall_Force' using 10 points (ITERATION 1)

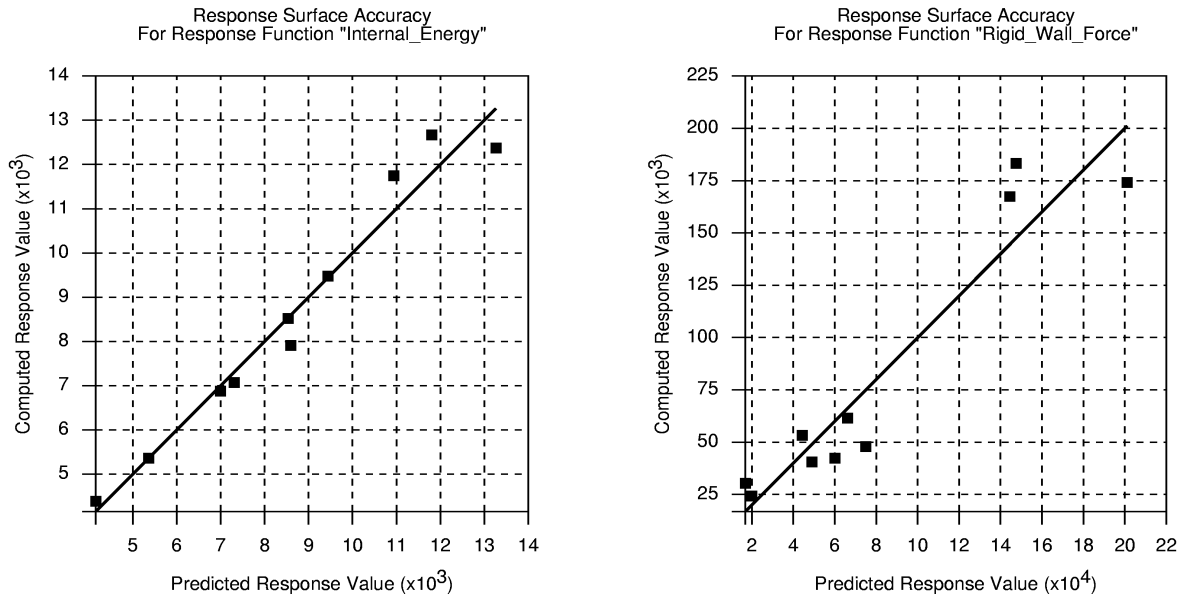
```

```

-----
                Global error parameters of response surface
-----
Quadratic Function Approximation:
-----
Mean response value           = 82483.2224
RMS error                     = 19905.3990 (24.13%)
Maximum Residual              = 35713.1794 (43.30%)
Average Error                  = 17060.6074 (20.68%)
Square Root PRESS Residual    = 54209.4513 (65.72%)
Variance                      = 792449819.5138
R^2                           = 0.8949
R^2 (adjusted)                = 0.8949
R^2 (prediction)              = 0.2204
Determinant of [X]'[X]        = 0.0556

```

The goodness of fit diagrams are shown in Figure 17-49.



**Figure 17-49: Prediction accuracy of Internal Energy and Rigid Wall Force (One Quadratic iteration)**

Nevertheless an optimization is conducted of the approximate subproblem, yielding a much improved feasible result. The objective function increases to 9575 (9777 computed) whereas the constraint is active at 70 000. The computed constraint is lower at 64 170. However the `Wall_Thickness` is now on the upper bound, suggesting an optimal value larger than 3.728.

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound
Radius	20	42.43	100
Wall_Thickness	2	3.728	6

RESPONSE FUNCTIONS:

RESPONSE	Scaled		Unscaled	
	Computed	Predicted	Computed	Predicted
Internal_Energy	9777	9575	9777	9575
Rigid_Wall_Force	6.417e+04	7e+04	6.417e+04	7e+04

### 17.3.4. Third iteration

Because of the large change in the `Wall_Thickness` on to the upper bound of the region of interest, a third iteration is conducted, keeping the region of interest the same. The starting point is the previous optimum:

Variable 'Radius' 42.43  
 Variable 'Wall\_Thickness' 3.728

The approximation improves as shown below:

Approximating Response 'Internal\_Energy' using 10 points (ITERATION 1)

-----  
 Global error parameters of response surface  
 -----

Quadratic Function Approximation:  
 -----

```

Mean response value      = 9801.0070
RMS error                = 439.8326 (4.49%)
Maximum Residual        = 834.5960 (8.52%)
Average Error           = 372.3133 (3.80%)
Square Root PRESS Residual = 1451.3233 (14.81%)
Variance                 = 386905.5050
R^2                     = 0.9618
R^2 (adjusted)          = 0.9618
R^2 (prediction)        = 0.5842
Determinant of [X]'[X]  = 0.0131
    
```

Approximating Response 'Rigid\_Wall\_Force' using 10 points (ITERATION 1)

-----  
 Global error parameters of response surface  
 -----

Quadratic Function Approximation:  
 -----

```

Mean response value      = 81576.0534
RMS error                = 12169.4703 (14.92%)
Maximum Residual        = 26348.0687 (32.30%)
Average Error           = 10539.2275 (12.92%)
Square Root PRESS Residual = 37676.3033 (46.19%)
Variance                 = 296192016.4365
R^2                     = 0.9301
R^2 (adjusted)          = 0.9301
R^2 (prediction)        = 0.3303
Determinant of [X]'[X]  = 0.0131
    
```

Because the size of the region of interest remained the same, the curve-fitting results show only a slight change (because of the new location), in this case an improvement. However, as the optimization results below show, the design is much improved, i.e. the objective value has increased whereas the approximate constraint is active. Unfortunately, due to the poor fit of the Rigid\_Wall\_Force, the simulation result exceeds the force constraint by about 10kN (14%). Further reduction of the region of interest is required to reduce the error, or filtering of the force can be considered to reduce the noise on this response.

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound
Radius	20	36.51	100
Wall_Thickness	2	4.478	6

RESPONSE FUNCTIONS:

RESPONSE	Scaled		Unscaled	
	Computed	Predicted	Computed	Predicted
Internal_Energy	1.129e+04	1.075e+04	1.129e+04	1.075e+04
Rigid_Wall_Force	8.007e+04	7e+04	8.007e+04	7e+04

The table below gives a summary of the three iterations of the step-by-step procedure.

**Table 17-6: Comparison of results (Cylinder impact)**

Variable	Initial	Iteration 1	Iteration 2	Iteration 3
Radius	75	50	42.43	36.51
Wall_thickness	3	2.978	3.728	4.478
Energy (Computed)	12960	7914	9777	11290
Force (Computed)	174900	47890	64170	80070

It is apparent that the result of the second iteration is a dramatic improvement on the starting design and a good approximation to the converged optimum design.

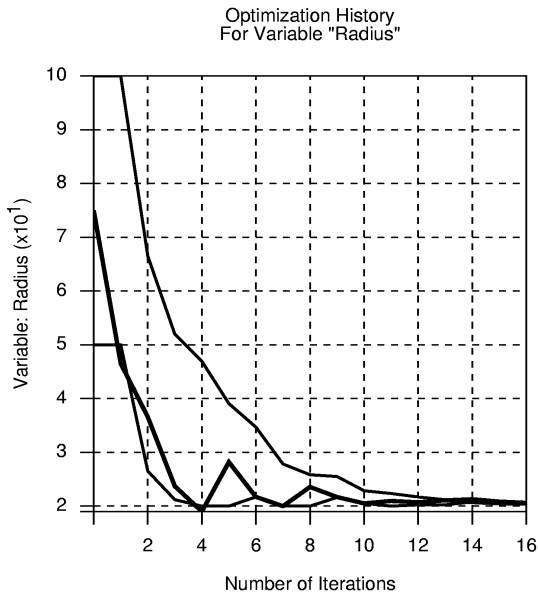
### 17.3.5. Response filtering: using the peak force as a constraint

Because of the poor accuracy of the response surface fit for the rigid wall force above, it was decided to modify the force constraint so that the peak filtered force is used instead. Therefore, the previous response definition for `Rigid_Wall_Force` is replaced with a command that extracts the *maximum* rigid wall force from a response from which frequencies exceeding 300Hz are excluded. The upper bound of the force constraint is changed to 80000.

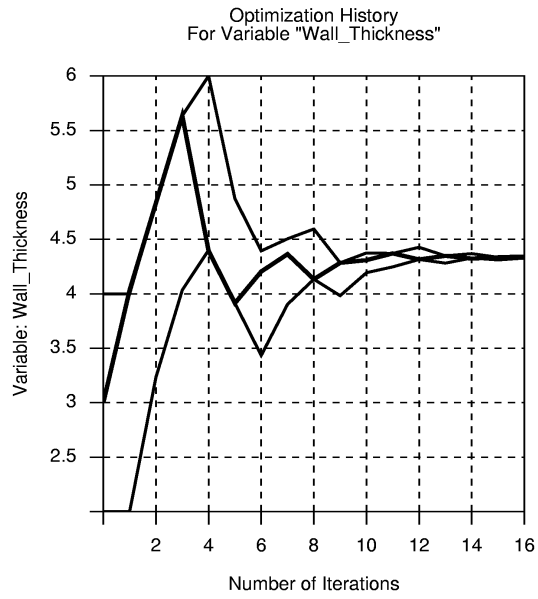
```
response 'Rigid_Wall_Force' "DynaASCII RWForc Normal 1 Max SAE 300"
```

20 iterations are specified with a 1% tolerance for convergence.

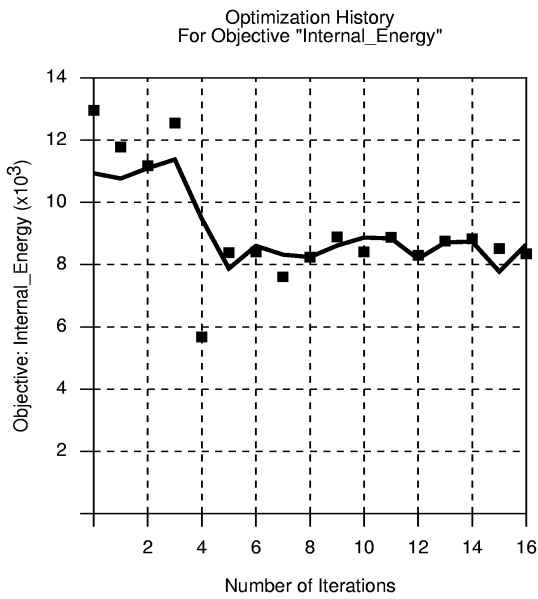
As expected, the response histories (Figure 17-50) show that the baseline design is severely infeasible (the first peak force is about  $1.75 \times 10^6$  vs. the constraint value of  $0.08 \times 10^6$ ). A steady reduction in the error of the response surfaces is observed up to about iteration 5. The optimization terminates after 16 iterations, having reached the 1% threshold for both objective and design variable changes.



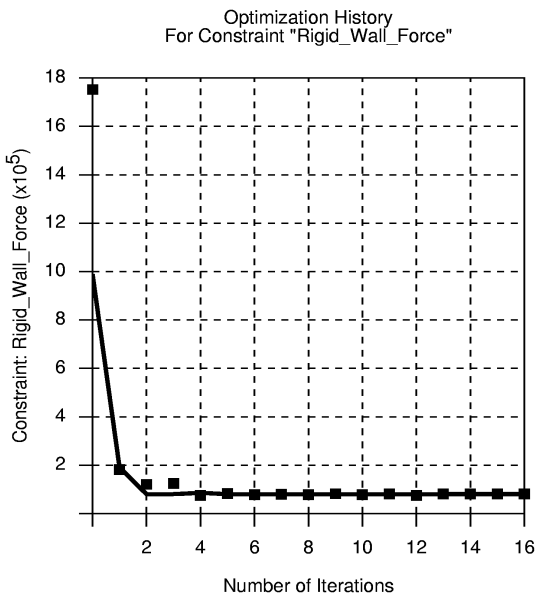
a) Radius



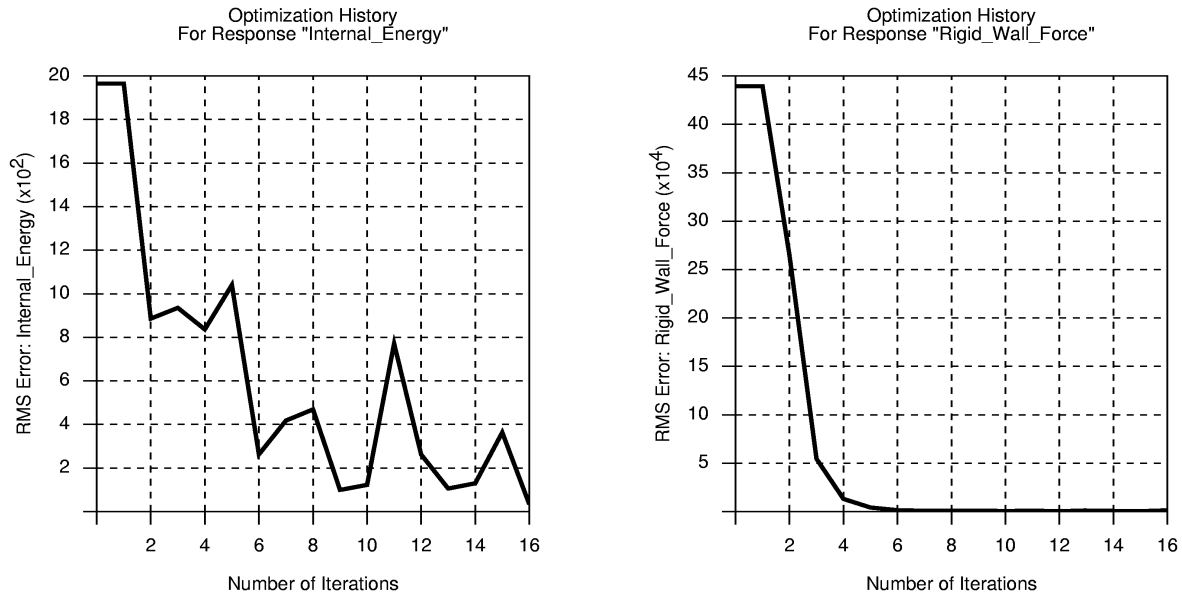
b) Wall\_Thickness



c) Internal\_Energy



d) Rigid\_Wall\_Force



e) RMS error of Internal\_Energy

f) RMS error of Rigid\_Wall\_Force

**Figure 17-50: Optimization history of automated design (filtered force)**

The optimization process steadily reduces the infeasibility, but the force constraint is still slightly violated when convergence is reached. The internal energy is significantly lower than previously:

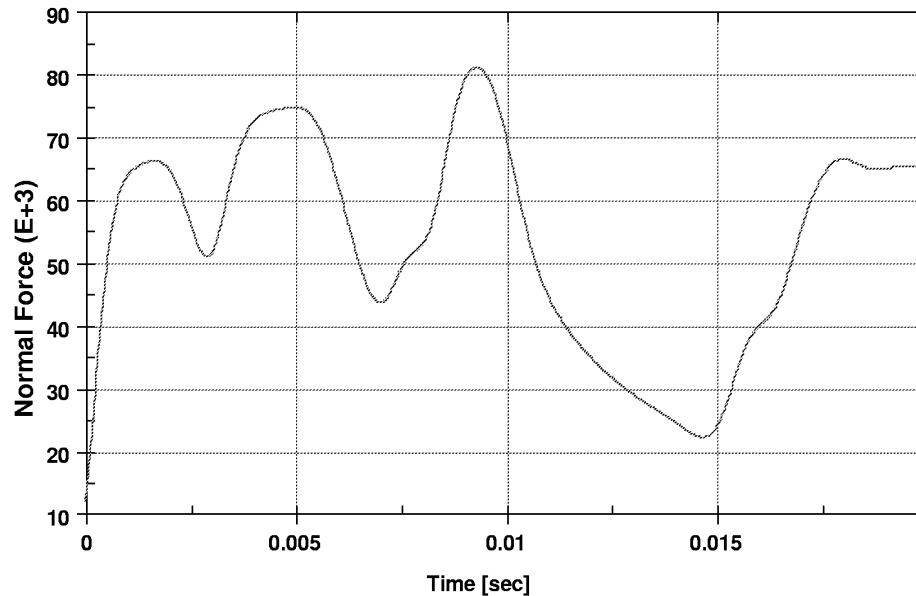
```

DESIGN POINT
-----
Variable Name          Lower Bound  Value  Upper Bound
-----|-----|-----|-----
Radius                 20          20.51  100
Wall_Thickness         2           4.342   6
-----|-----|-----|-----
RESPONSE FUNCTIONS:
-----

```

RESPONSE	Scaled		Unscaled	
	Computed	Predicted	Computed	Predicted
Internal_Energy	8344	8645	8344	8645
Rigid_Wall_Force	8.112e+04	8e+04	8.112e+04	8e+04

Figure 17-51 below confirms that the final design is only slightly infeasible when the maximum filtered force exceeds the specified limit for a short duration at around 9ms.



*Figure 17-51: Cylinder: Constrained rigid wall force:  $F(t) < 80000$  (SAE 300Hz filtered)*

## 17.4. 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:

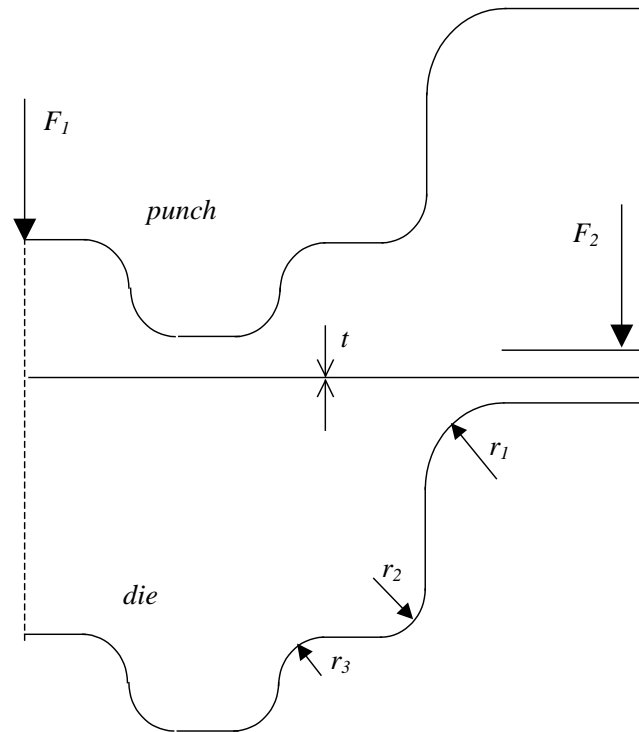
- The maximum of all the design variables is minimized.
- Adaptive meshing is used in the finite element analysis.
- The binary LS-DYNA database is used.
- The example employs the sheet metal forming interface utilities.
- Composite functions are used.
- An appended file containing extra input is used.
- The example utilizes the independent parametric preprocessor, Truegrid<sup>11</sup>.

### 17.4.1. Problem statement

The design parameterization for the sheet metal forming example is shown in Figure 17-52.

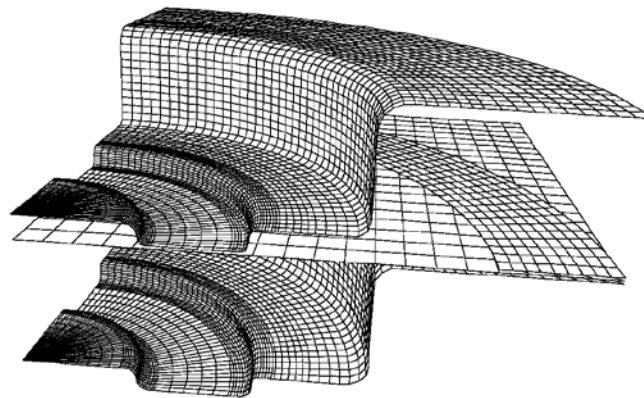
<sup>11</sup> Registered Trademark of XYZ Scientific Applications Inc.





**Figure 17-52: Parameterization of cross-section**

The FE model is shown in Figure 17-53.



**Figure 17-53: Quarter segment of FE model: tools and blank**

The design problem is formulated to minimize the maximum tool radius while also specifying an FLD constraint and a maximum thickness reduction of 20% (thinning constraint). Since the user wants to enforce the FLD and thinning constraints strictly, these constraints are defined as `strict`. To minimize the maximum radius, a small upper bound for the radii has been specified (arbitrarily chosen as a number close to the lower bound of the design space, namely 1.1). The optimization solver will then minimize the maximum difference between the radii and their respective bounds. The radius constraints must not be enforced strictly. This translates to the following mathematical formulation:

Minimize  $e$

with

$$1.5 \leq r_1 \leq 4.5$$

$$1.5 \leq r_2 \leq 4.5$$

$$1.5 \leq r_3 \leq 4.5$$

subject to

$$g^{FLD}(\mathbf{x}) < 0.0$$

$$\Delta t(\mathbf{x}) < 20\%$$

$$r_1 - 1.1 < e$$

$$r_2 - 1.1 < e$$

$$r_3 - 1.1 < e$$

$$e > 0.$$

The design variables  $r_1$ ,  $r_2$  and  $r_3$  are the radii of the work piece as indicated in Figure 17-52.  $\Delta t$  is the thickness reduction which is positive when the thickness is reduced. The FLD constraint is feasible when smaller than zero.

### 17.4.2. First Iteration

The initial run is a quadratic analysis designed as an initial investigation of the following issues:

- The dependency of the through thickness strain constraint on the radii.
- The dependency of the FLD constraint on the radii.
- The location of the optimal design point.

The subregion considered for this study is 2.0 large in  $r_1$ ,  $r_2$  and  $r_3$  and is centered about  $(1.5, 1.5, 1.5)^T$ . The FLD constraint formulation tested in this phase is based on the maximum perpendicular distance of a point violating the FLD constraint to the FLD curve (see Section 6.3.2).

The LS-OPT command file used to run the problem is:

```
"Sheet: Minimization of Maximum Tool Radius"
Author "Aaron Spelling"
$ Created on Wed May 29 19:23:20 2002
$
$ DESIGN VARIABLES
$
variables 3
Variable 'Radius_1' 1.5
Lower bound variable 'Radius_1' 1
Upper bound variable 'Radius_1' 4.5
Range 'Radius_1' 4
```

```

Variable 'Radius_2' 1.5
  Lower bound variable 'Radius_2' 1
  Upper bound variable 'Radius_2' 4.5
  Range 'Radius_2' 4
Variable 'Radius_3' 1.5
  Lower bound variable 'Radius_3' 1
  Upper bound variable 'Radius_3' 4.5
  Range 'Radius_3' 4
solvers 1
responses 2
$
$ NO HISTORIES ARE DEFINED
$
$
$ DEFINITION OF SOLVER "DYNA1"
$
  solver dyna 'DYNA1'
    solver command "lsdyna"
    solver input file "trugrdo"
    solver append file "ShellSetList"
    prepro truegrid
    prepro command "/net/src/ultra4_4/common/hp/tg2.1/tg"
    prepro input file "m3.tg.opt"
$
$ RESPONSES FOR SOLVER "DYNA1"
$
  response 'Thinning' 1 0 "DynaThick REDUCTION MAX"
  response 'Thinning' linear
  response 'FLD' 1 0 "DynaFLDg CENTER 1 2 3 90"
  response 'FLD' linear
$
$ NO HISTORIES DEFINED FOR SOLVER "DYNA1"
$
$
$ HISTORIES AND RESPONSES DEFINED BY EXPRESSIONS
$
  composites 4
  composite 'Rad1' type weighted
    composite 'Rad1' variable 'Radius_1' 1 scale 1
  composite 'Rad2' type weighted
    composite 'Rad2' variable 'Radius_2' 1 scale 1
  composite 'Rad3' type weighted
    composite 'Rad3' variable 'Radius_3' 1 scale 1
  composite 'Thinning_scaled' {Thinning/100}
$
$ NO OBJECTIVES DEFINED
$
  objectives 0
$
$ CONSTRAINT DEFINITIONS
$
  constraints 5
  constraint 'FLD'
    strict
    upper bound constraint 'FLD' 0.0
  constraint 'Rad1'
    slack

```

```

upper bound constraint 'Rad1' 1.1
constraint 'Rad2'
upper bound constraint 'Rad2' 1.1
constraint 'Rad3'
upper bound constraint 'Rad3' 1.1
constraint 'Thinning_scaled'
strict
upper bound constraint 'Thinning_scaled' 0.2
$
$ EXPERIMENTAL DESIGN
$
Order quadratic
Experimental design dopt
Basis experiment 3toK
Number experiment 16
$
$ JOB INFO
$
concurrent jobs 8
iterate param design 0.01
iterate param objective 0.01
iterate 1
STOP

```

The file `ShellSetList` contains commands for LS-DYNA in addition to the preprocessor output. It is slotted into the input file. Adaptive meshing is chosen as an analysis feature for the simulation. The FLD curve data is also specified in this file. The extra commands are:

```

*DATABASE_BINARY_RUNRSF
70
*DATABASE_EXTENT_BINARY
0, 0, 0, 1, 0, 0, 0, 1
0, 0, 0, 0, 0, 0
$
$ SLIDING INTERFACE DEFINITIONS
$
$ TrueGrid Sliding Interface # 1
$
*CONTACT_FORMING_ONE_WAY_SURFACE_TO_SURFACE
$ workpiece vs punch
0.1000000          0.000  0.000
      1          2          3          3          1
      0.0
$
*CONTACT_FORMING_ONE_WAY_SURFACE_TO_SURFACE
$ workpiece vs die
0.1000000          0.000  0.000
      1          1          3          3          3          1
      0.0
$
*CONTACT_FORMING_ONE_WAY_SURFACE_TO_SURFACE
$ workpiece vs blankholder
0.1000000          0.000  0.000
      1          1          1          4          3          3
      0.0
$
*CONTROL_ADAPTIVE
$ ADPFREQ  ADPTOL  ADPOPT  MAXLVL  TBIRTH  TDEATH  LCADP  IOFLAG
0.100E-03  5.000    2      3  0.000E+00  1.0000000  0      1
$ ADPSIZE  ADPASS  IREFLG  ADPENE
0.0000000  1      0      3.0000

```

```

*LOAD_RIGID_BODY
$   rbID          dir      lcID      scale
    2             3        2 1.0000000
*LOAD_RIGID_BODY
$   rbID          dir      lcID      scale
    4             3        3 1.0000000
*DEFINE_CURVE
$   FLD curve
90
$
-1,2.083
0,.25
1,.75
*END

```

The input file (file `m3.tg.opt`) used to generate the FE mesh in Truegrid is:

```

c generate LS-DYNA input deck for sheet metal example
lsdyna keyword
lsdyopts endtim .0009  nodout 1.e-6  d3plot dtcycl .0001 ; ;
lsdyopts istupd 1 ;
c lsdymats 1 37 shell elfor bt rho 7.8e-9 e 2.e5 pr .28
c   sigy 200. etan 572 er 1.4 ;
lsdymats 2 20 shell elfor bt rho 7.8e-9 e 2.e5 pr .28 shth .1
c   cmo con 4 7;
lsdymats 3 20 shell elfor bt rho 7.8e-9 e 2.e5 pr .28 shth .1
c   cmo con 7 7 ;
lsdymats 4 20 shell elfor bt rho 7.8e-9 e 2.e5 pr .28 shth .1
c   cmo con 4 7;
plane 2 0 0 0 1 0 0 .01 symm ;
plane 3 0 0 0 0 1 0 0.01 symm ;
c sid 1 lsdsl a10 slvmat 1;mstmat 2;scoef .1 ; ; ;
c sid 2 lsdsl a10 slvmat 1;mstmat 3;scoef .1 ; ; ;
c sid 3 lsdsl a10 slvmat 1;mstmat 4;scoef .1 ; ; ;
c
lcd 1
    0.0000000000E+00      0.2756000006E+03
    0.665699990E-04      0.2761000006E+03
    0.1365000006E-03      0.276700012E+03
.
.
.
    0.312799990E+00      0.481799988E+03
    0.469900012E+00      0.517200012E+03
    0.705600023E+00      0.555299988E+03
;
c
c die cross-section
para
c
r1 <<Radius_1>> c upper radius  minimum = 2.
r2 <<Radius_2>> c middle radius  minimum = 2.
r3 <<Radius_3>> c lower radius   minimum = 2.
load2 -100000
load3 -20000
th1 1.0          c thickness of blank
th3 .00          c thickness of die and punch

```

```

th2 [1.001*%th1]
l1 20          c length of draw (5-40)
c
z5 [%l1-22]
c Position of workpiece
z4 [%z5+1.001*%th1/2.+%th3/2]
c Position of blankholder
z3 [%z4+1.001*%th1/2.+%th3/2]
n1 [25+4.0*%l1]
n2 [25+8.0*%l1]
c part 2
z6 [%z5+4+%th2]
z7 [%z5+%l1+4+%th2]
;
c
c die cross-section
.
.
.
c punch cross-section (closed configuration)
ld 2
lod 1 [%th2+%th3]

c punch cross-section (withdrawn configuration)
ld 3 lst1 2 0 [%z5+26]
.
.
.

endpart
c ***** part 2 mat 2 ***** punch
cylinder
1 8 35 40 67 76 [76+%n1] [70+%n1+10]; 1 41 ; -1 ;
.001 17. 23. 36. 44. 50. 75. 100.
0. 90.
%z7
.
.
.

thick %th3
mate 2
endpart

c ***** part 3 mat 4 ***** blankholder
cylinder
1 10 ; 1 41 ; -1 ;
80. 100.
0. 90.
[%z3]
b 0 0 0 0 0 0 dx 1 dy 1 rx 1 ry 1 rz 1;
thick %th3
mate 4
endpart

```

```

c ***** part 4 mat 1 workpiece
block
1 21 ; 1 21 ; -1 ;
0. 100.
0. 100.
[%z4]
thick [%th1]
mate 1
endpart
merge
write
end

```

The error parameters for the fitted functions are given in the following output (from `lsopt_output` file):

```
Approximating Response 'Thinning' using 16 points (ITERATION 1)
```

```
-----
Global error parameters of response surface
-----
```

```
Quadratic Function Approximation:
```

```
-----
Mean response value           = 27.8994
RMS error                     = 0.6657 (2.39%)
Maximum Residual              = 1.2932 (4.64%)
Average Error                  = 0.5860 (2.10%)
Square Root PRESS Residual    = 2.0126 (7.21%)
Variance                      = 1.0130
R^2                           = 0.9913
R^2 (adjusted)                = 0.9826
R^2 (prediction)              = 0.9207
Determinant of [X]'[X]        = 2231.5965
Approximating Response 'FLD' using 16 points (ITERATION 1)
-----
```

```
-----
Global error parameters of response surface
-----
```

```
Quadratic Function Approximation:
```

```
-----
Mean response value           = 0.0698
RMS error                     = 0.0121 (17.33%)
Maximum Residual              = 0.0247 (35.35%)
Average Error                  = 0.0103 (14.74%)
Square Root PRESS Residual    = 0.0332 (47.59%)
Variance                      = 0.0003
R^2                           = 0.9771
R^2 (adjusted)                = 0.9542
R^2 (prediction)              = 0.8272
Determinant of [X]'[X]        = 2231.5965
-----
```

The thinning has a reasonably accurate response surface but the FLD approximation requires further refinement. The initial design has the following response surface results which fail the criteria for maximum thinning, but not for FLD:

```
DESIGN POINT
```

```
-----
Variable Name          Lower Bound  Value      Upper Bound
-----|-----|-----|-----
```

Radius_1	1	1.5	4.5		
Radius_2	1	1.5	4.5		
Radius_3	1	1.5	4.5		
----- ----- ----- ----- ----- -----					
CONSTRAINT FUNCTIONS:					
----- ----- ----- ----- ----- -----					
CONSTRAINT NAME	Computed	Predicted	Lower	Upper	Viol?
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
FLD	0.09123	0.1006	-1e+30	0	YES
Rad1	1.5	1.5	-1e+30	1.1	YES
Rad2	1.5	1.5	-1e+30	1.1	YES
Rad3	1.5	1.5	-1e+30	1.1	YES
Thinning_scaled	0.2957	0.3078	-1e+30	0.2	YES
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
CONSTRAINT VIOLATIONS:					
----- ----- ----- ----- ----- -----					
CONSTRAINT NAME	Computed Violation	Predicted Violation			
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
	Lower	Upper	Lower	Upper	
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
FLD	-	0.09123	-	0.1006	
Rad1	-	0.4	-	0.4	
Rad2	-	0.4	-	0.4	
Rad3	-	0.4	-	0.4	
Thinning_scaled	-	0.09567	-	0.1078	
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----

As shown below, after 1 iteration, a feasible design is generated. The simulation response of the optimum is closely approximated by the response surface.

DESIGN POINT					
----- ----- ----- ----- ----- -----					
Variable Name	Lower Bound	Value	Upper Bound		
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	
Radius_1	1	3.006	4.5		
Radius_2	1	3.006	4.5		
Radius_3	1	3.006	4.5		
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	
CONSTRAINT FUNCTIONS:					
----- ----- ----- ----- ----- -----					
CONSTRAINT NAME	Computed	Predicted	Lower	Upper	Viol?
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
FLD	-0.04308	-0.03841	-1e+30	0	no
Rad1	3.006	3.006	-1e+30	1.1	YES
Rad2	3.006	3.006	-1e+30	1.1	YES
Rad3	3.006	3.006	-1e+30	1.1	YES
Thinning_scaled	0.2172	0.2	-1e+30	0.2	no
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
CONSTRAINT VIOLATIONS:					
----- ----- ----- ----- ----- -----					
CONSTRAINT NAME	Computed Violation	Predicted Violation			
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
	Lower	Upper	Lower	Upper	
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----
FLD	-	-	-	-	
Rad1	-	1.906	-	1.906	
Rad2	-	1.906	-	1.906	
Rad3	-	1.906	-	1.906	
Thinning_scaled	-	0.01718	-	-	
----- ----- ----- ----- ----- -----	-----	-----	-----	-----	-----

### 17.4.3. Automated design

The optimization process can also be automated so that no user intervention is required. The starting design, lower and upper bounds, and region of interest is modified from the 1 iteration study above.

The input file is modified as follows:



The variable definitions are as follows:

```
Variable 'Radius_1' 1.5
  Lower bound variable 'Radius_1' 1
  Upper bound variable 'Radius_1' 4.5
  Range 'Radius_1' 1
Variable 'Radius_2' 1.5
  Lower bound variable 'Radius_2' 1
  Upper bound variable 'Radius_2' 4.5
  Range 'Radius_2' 1
Variable 'Radius_3' 1.5
  Lower bound variable 'Radius_3' 1
  Upper bound variable 'Radius_3' 4.5
  Range 'Radius_3' 1
```

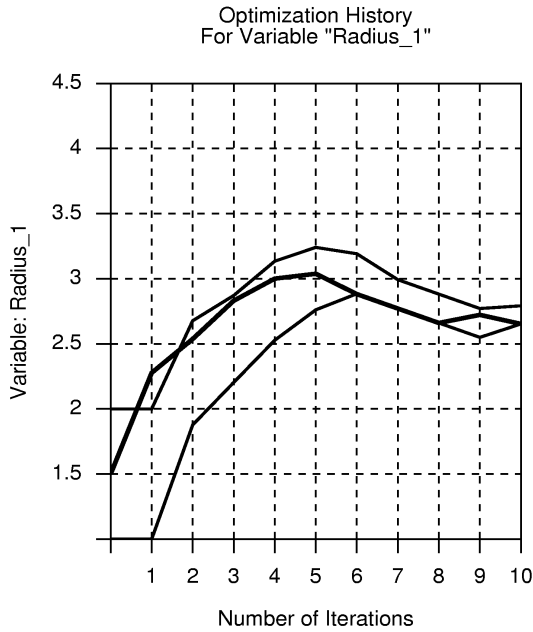
The number of  $D$ -optimal experiments is reduced because of the linear approximation used:

```
Order linear
Experimental design dopt
Basis experiment 3toK
Number experiment 7
```

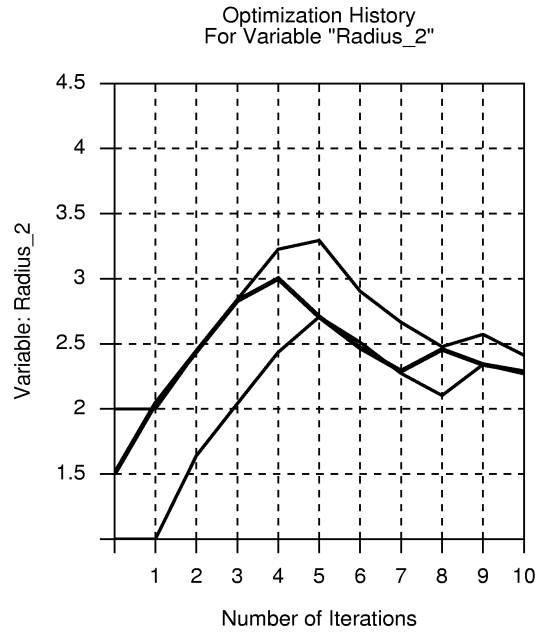
The optimization is run for 10 iterations:

```
iterate 10
```

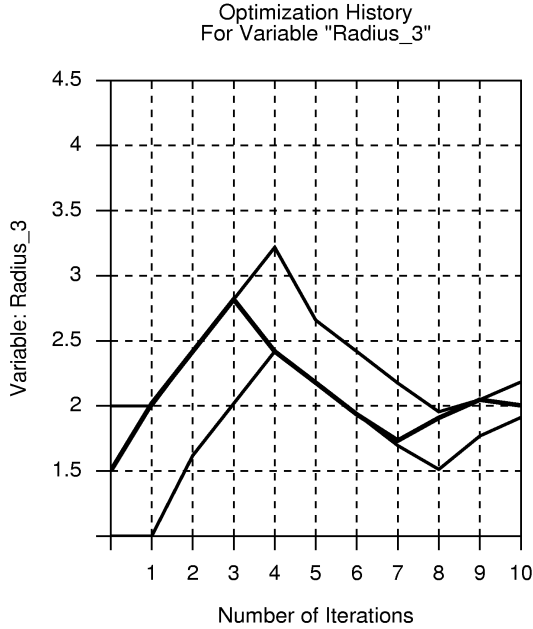
The optimization history is shown in Figure 17-54 for the design variables and responses:



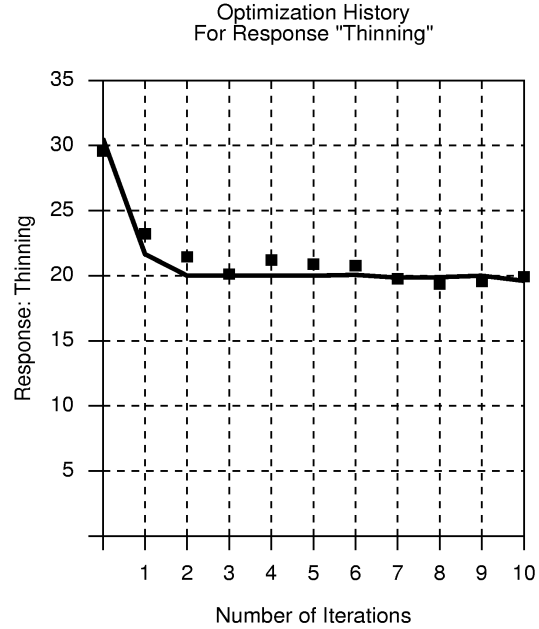
a) Optimization history of variable 'Radius\_1'



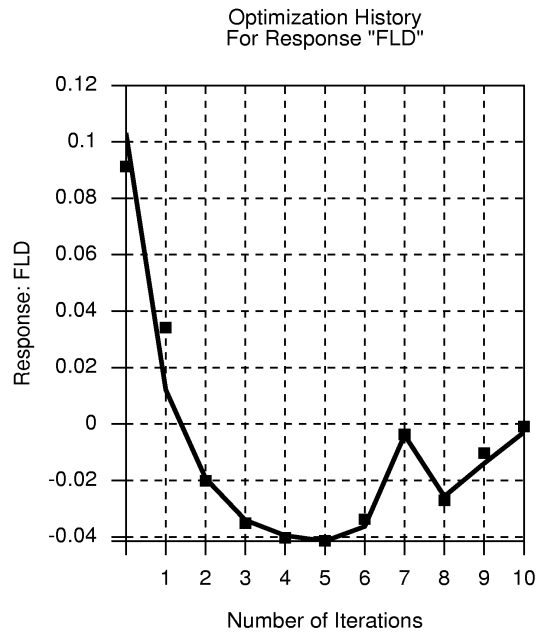
b) Optimization history of variable 'Radius\_2'



c) Optimization history of variable 'Radius\_3'



d) Optimization history of response 'Thinning'



e) Optimization history of response FLD

**Figure 17-54: Optimization history of design variables and responses (automated design)**

The details of the 10<sup>th</sup> iteration have been extracted:

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound
Radius_1	1	2.653	4.5
Radius_2	1	2.286	4.5
Radius_3	1	2.004	4.5

RESPONSE FUNCTIONS:

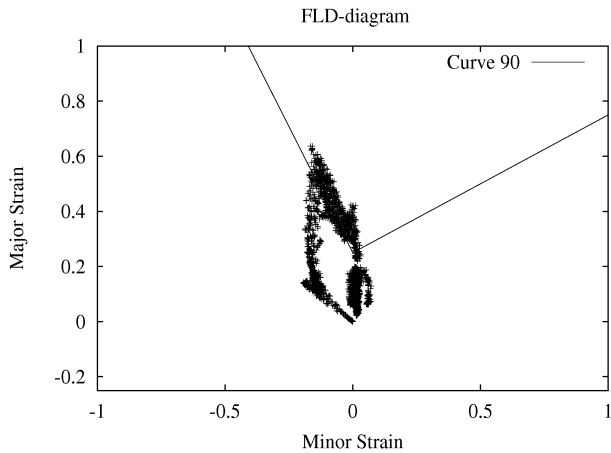
RESPONSE	Scaled		Unscaled	
	Computed	Predicted	Computed	Predicted
Thinning	19.92	19.6	19.92	19.6
FLD	-0.000843	-0.002907	-0.000843	-0.002907

A comparison between the starting and the final values is tabulated below:

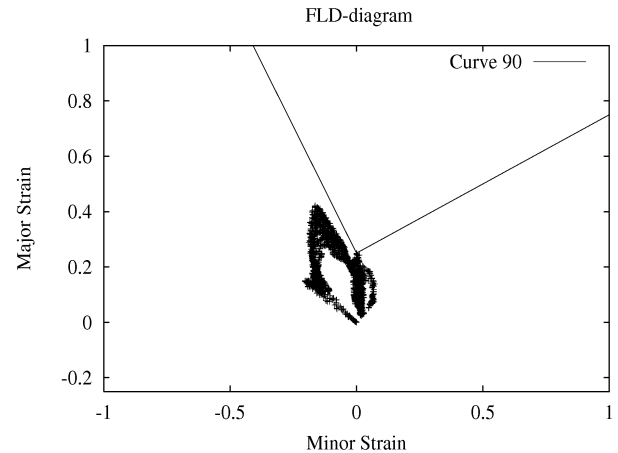
**Table 17-7: Comparison of results (Sheet-metal forming)**

Variable	Start (Computed)	Optimal (Predicted)	Optimal (Computed)
Thinning	29.57	19.92	19.6
FLD	0.09123	-0.000843	-0.002907
Radius_1	1.5	2.653	
Radius_2	1.5	2.286	
Radius_3	1.5	2.004	

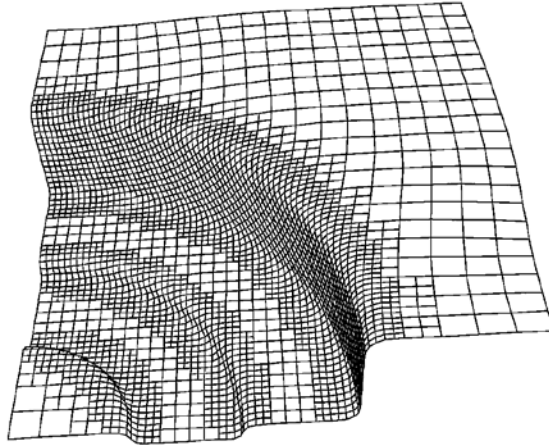
The FLD diagrams (Figure 17-55) for the baseline design and the optimum illustrate the improvement of the FLD feasibility:



Baseline FLD diagram

FLD diagram of 10<sup>th</sup> iteration**Figure 17-55: FLD diagrams of baseline and 10<sup>th</sup> iteration**

A typical deformed state is depicted in Figure 17-56 below.



*Figure 17-56: Deformed state*

## 17.5. Large vehicle crash and vibration (MDO/MOO) (7 variables)

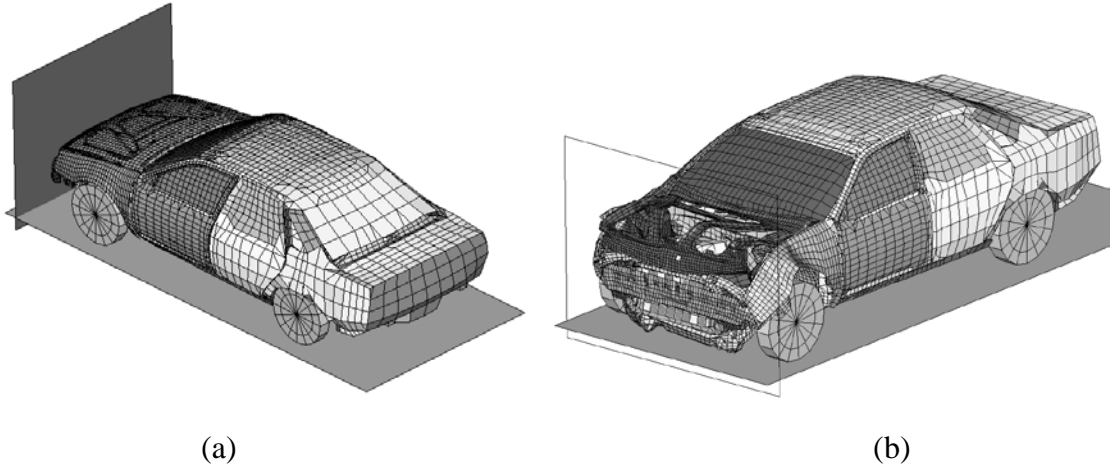
This example has the following features:

- 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.

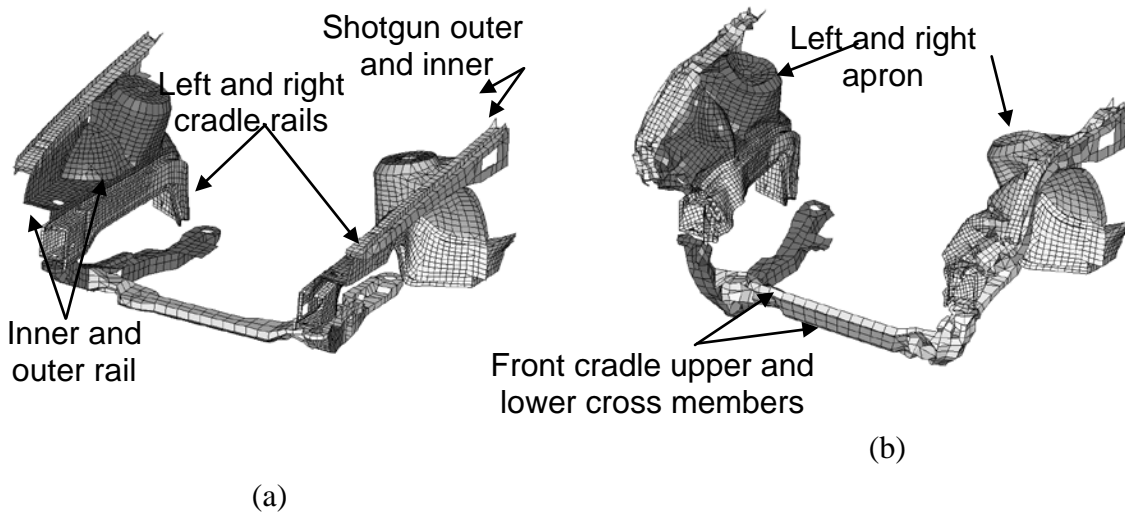
This example illustrates a realistic application of Multidisciplinary Design Optimization (MDO) and concerns the coupling of the crash performance of a large vehicle with one of its Noise Vibration and Harshness (NVH) criteria, namely the torsional mode frequency [2].

### 17.5.1. FE Modeling

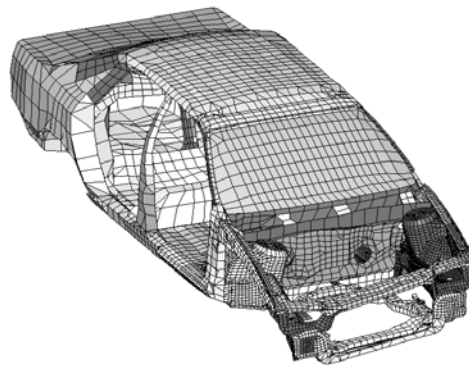
The crashworthiness simulation considers a model containing approximately 30,000 elements of a National Highway Transportation and Safety Association (NHTSA) vehicle [3] undergoing a full frontal impact. A modal analysis is performed on a so-called ‘body-in-white’ model containing approximately 18,000 elements. The crash model for the full vehicle is shown in Figure 17-57 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 17-58. The NVH model is depicted in Figure 17-59 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 17-58), parameterized directly in the LS-DYNA input file. Twelve parts are affected, comprising aprons, rails, shotguns, cradle rails and the cradle cross member (Figure 17-58). LS-DYNA v.971 is used for both the crash and NVH simulations, in explicit and implicit modes respectively.



**Figure 17-57: Crash model of vehicle showing road and wall a) Undeformed b) Deformed (78ms)**



**Figure 17-58: Structural components affected by design variables – a) Undeformed and (b) deformed (time = 72ms)**



**Figure 17-59: Body-in-white model of vehicle in torsional vibration mode (38.7Hz)**

### 17.5.2. Design formulation

The formulation is as follows:

$$\begin{aligned}
 & \text{Minimize} && \text{Mass} \\
 & \text{Minimize} && \text{Maximum intrusion} \\
 & \text{subject to} && \\
 & && \text{Stage 1 pulse}(\mathbf{x}_{\text{crash}}) > 14.51\text{g} \\
 & && \text{Stage 2 pulse}(\mathbf{x}_{\text{crash}}) > 17.59\text{g} \\
 & && \text{Stage 3 pulse}(\mathbf{x}_{\text{crash}}) > 20.75\text{g} \\
 & && 41.38\text{Hz} < \text{Torsional mode frequency}(\mathbf{x}_{\text{NVH}}) < 42.38\text{Hz}
 \end{aligned}$$

The three stage pulses are calculated from the SAE filtered (60Hz) acceleration and displacement of a left rear sill node in the following fashion:

$$\text{Stage } i \text{ pulse} = \frac{-k}{d_2 - d_1} \int_{d_1}^{d_2} a dx ;$$

$$k = 2.0 \text{ for } i = 1, 1.0 \text{ otherwise;}$$

with the limits  $[d_1; d_2] = [0; 184]; [184; 334]; [334; \text{Max}(\text{displacement})]$  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.

### 17.5.3. 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.

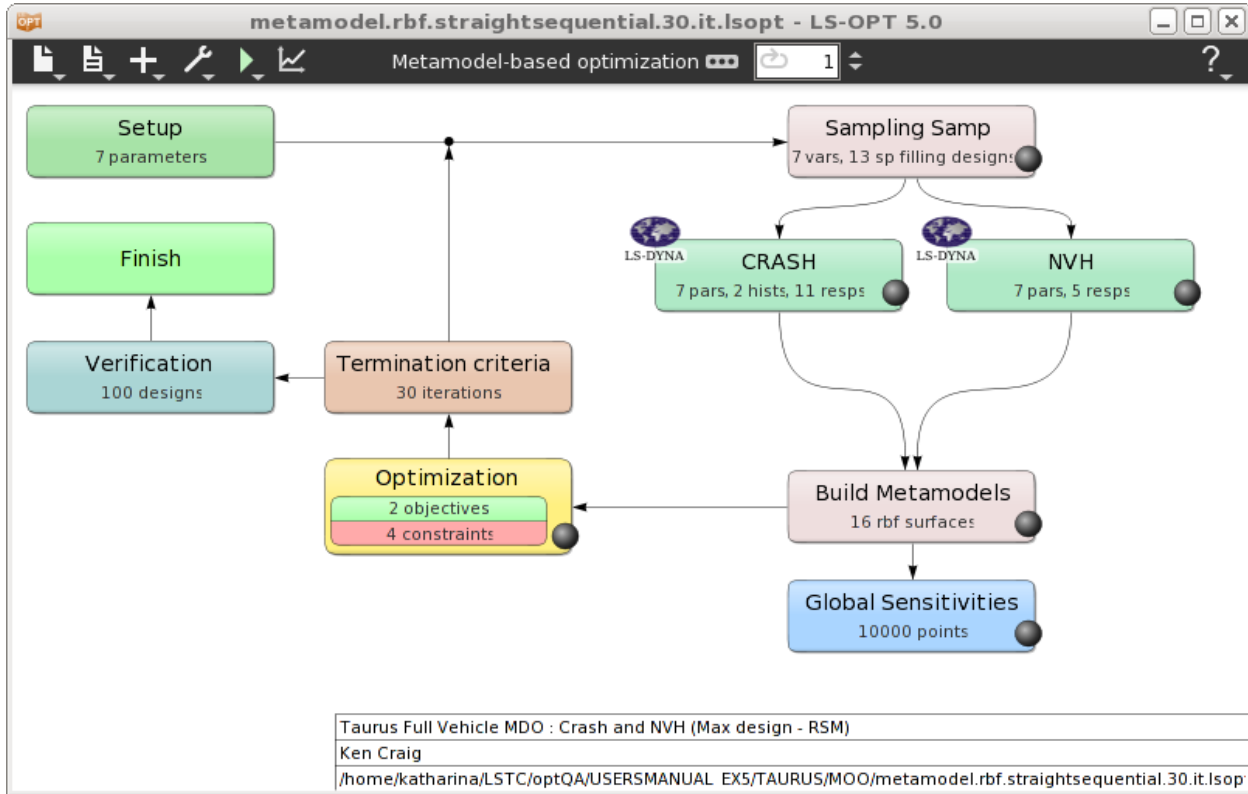
Figure 17-60 shows the LS-OPT main GUI window for a multi-disciplinary optimization using metamodels.

For the main task, select a metamodel-based optimization, Figure 17-61. 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 17-62. 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

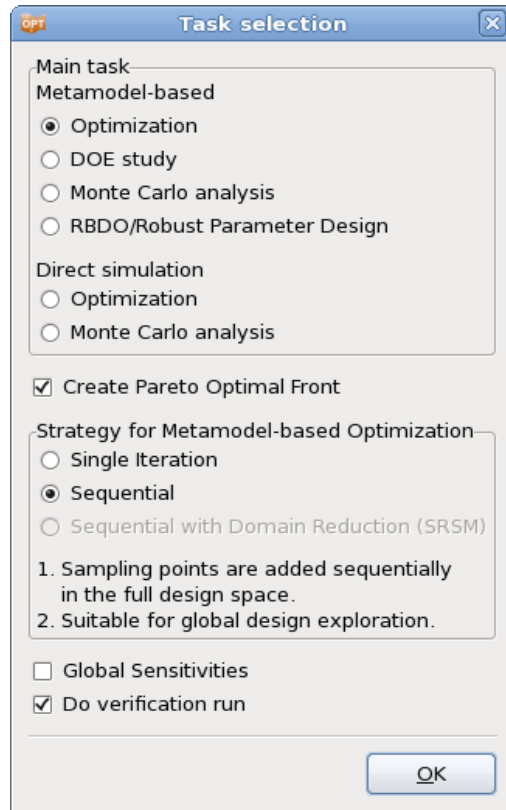
function may be used to get the value of  $t$  for a specified value of the selected history function, Figure 17-63. Then the stage pulses may be calculated using the Integral function, Figure 17-64.

For the NVH load case, the FREQUENCY interface may be used to extract the frequency and related responses, Figure 17-65. Make sure that mode tracking is used.

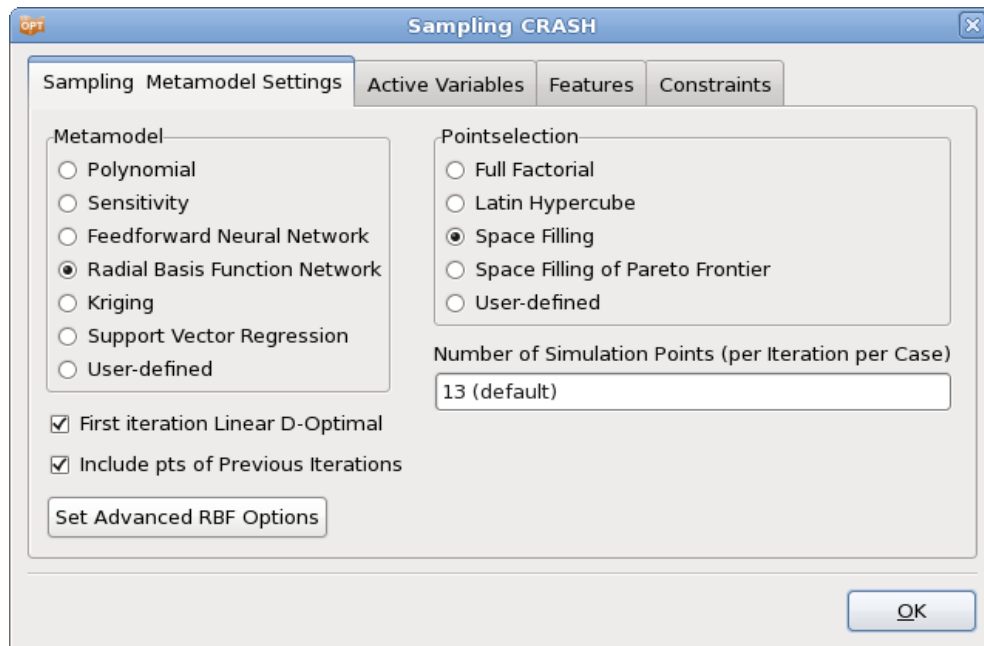


**Figure 17-60: Main LS-OPT GUI; Metamodel based optimization; two disciplines.**

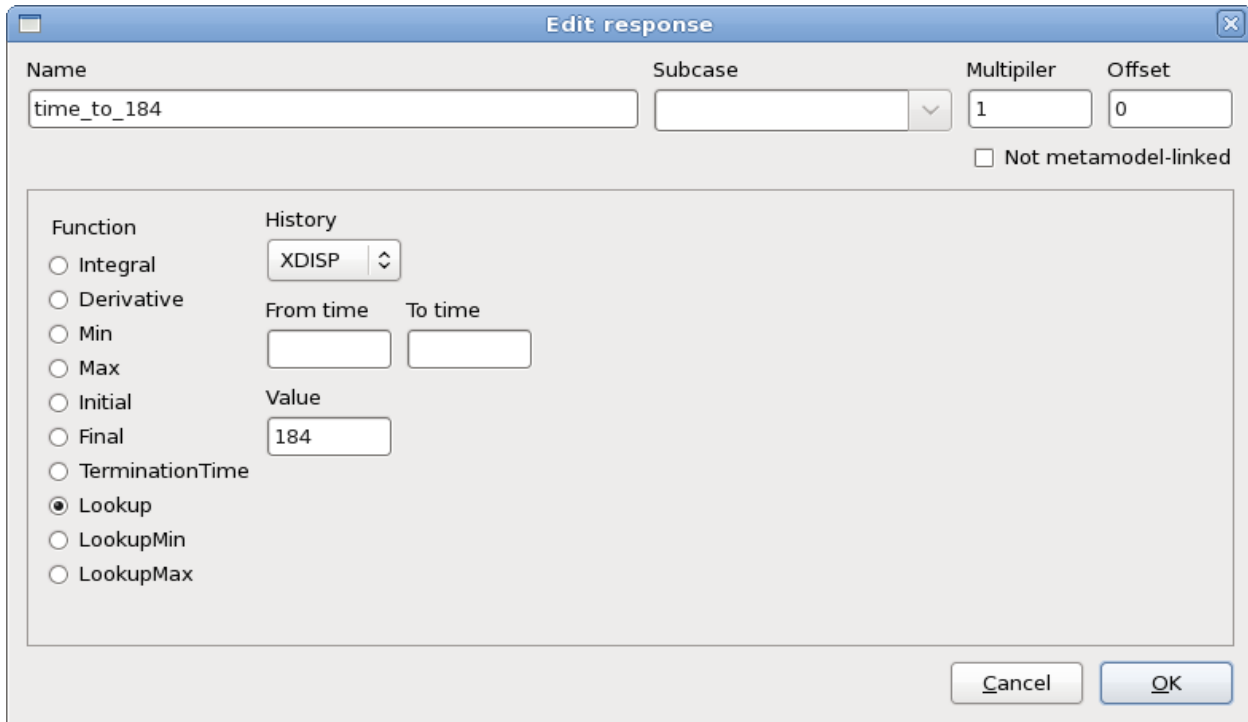




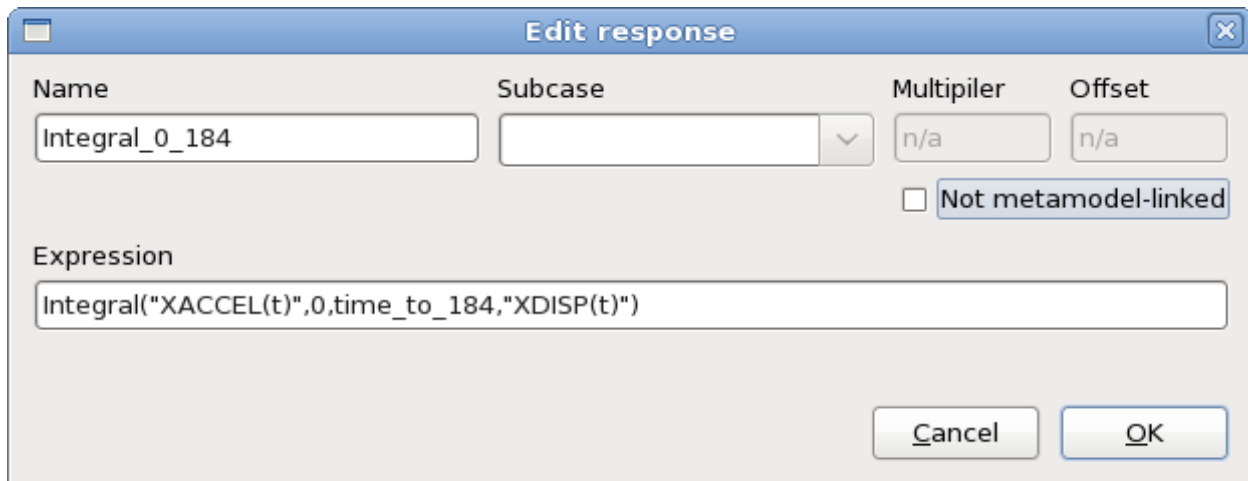
**Figure 17-61:** Task dialog; Calculating Pareto Optimal solutions using a metamodel-base method using sequential strategy.



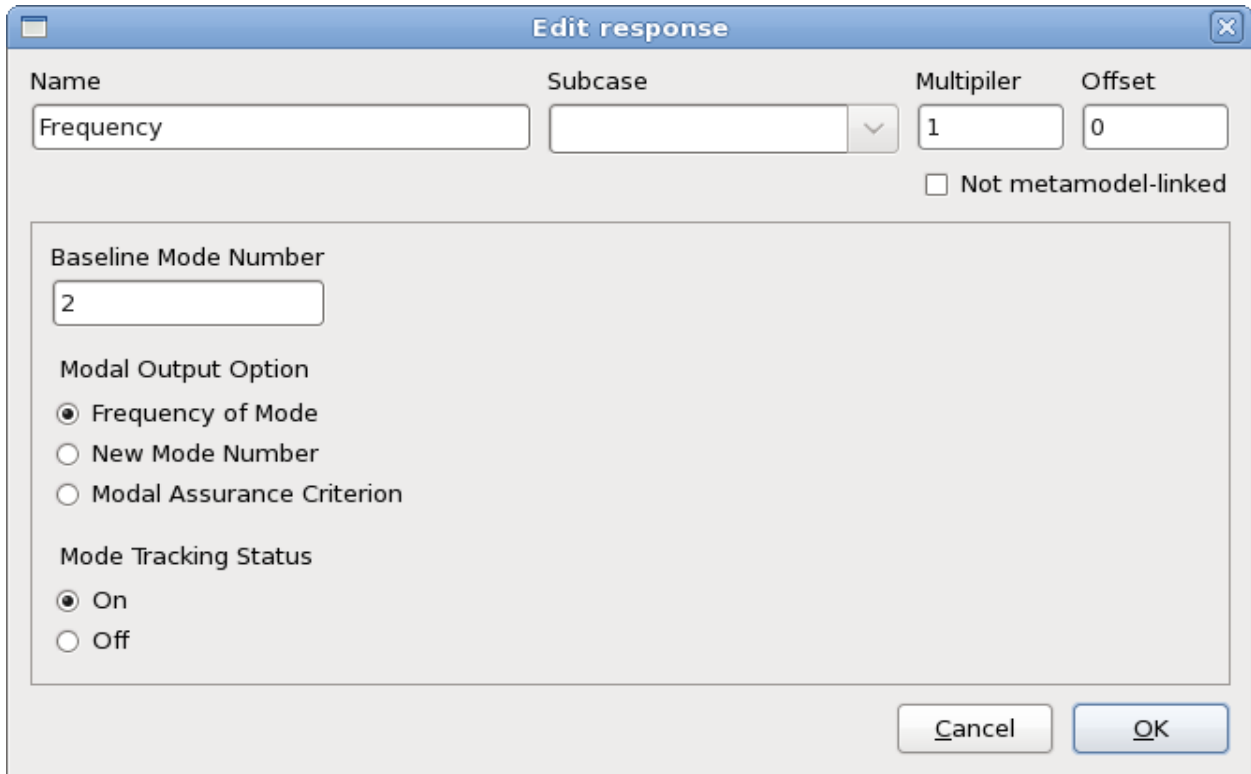
**Figure 17-62:** Sampling dialog; use Radial Basis Functions to get a global approximation.



**Figure 17-63: Lookup function; evaluate the value of  $t$  for a specified value of the history XDISP**

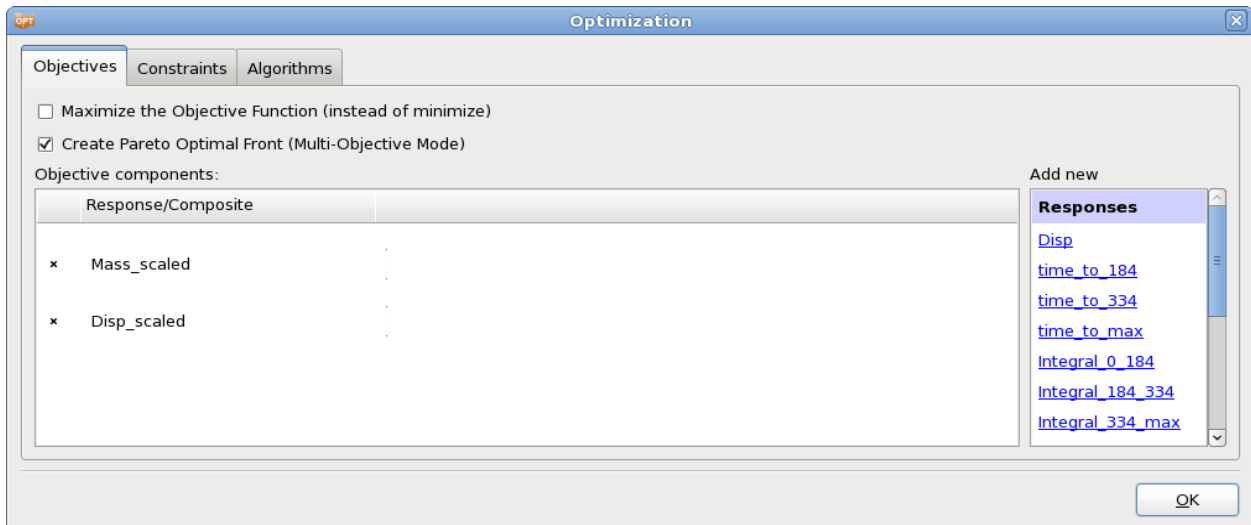


**Figure 17-64: Response Expression; The stage pulses are calculated using the Integral function.**



**Figure 17-65: Frequency extraction with Mode Tracking**

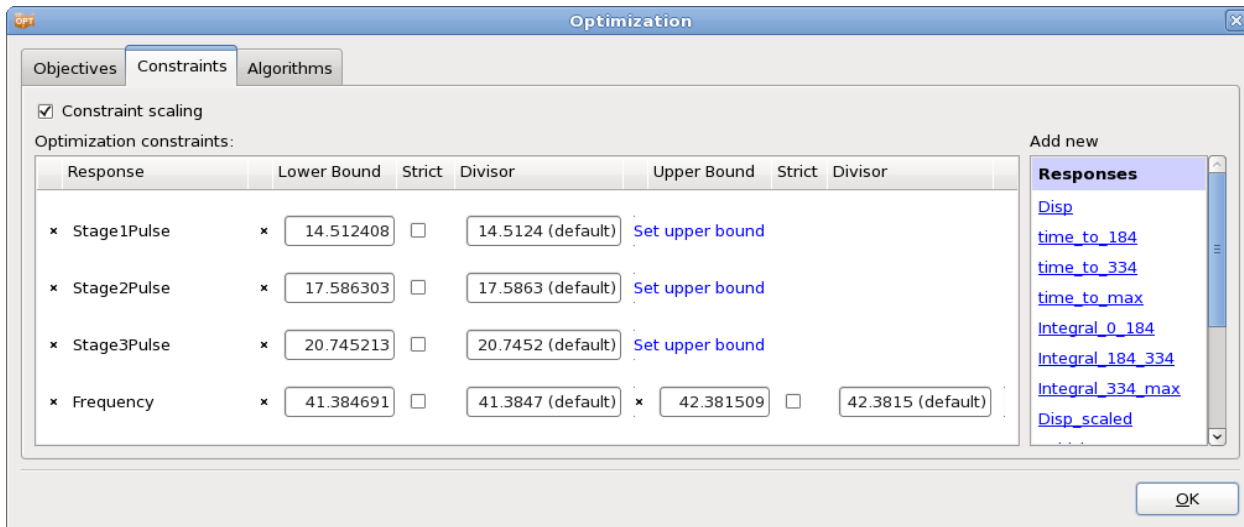
Define the objective and constraint functions in the Optimization dialog. For the objectives, make sure that the multi-objective mode is selected, Figure 17-66.



**Figure 17-66: Objectives panel; Select Multi-Objective Mode to create Pareto Optimal Front**

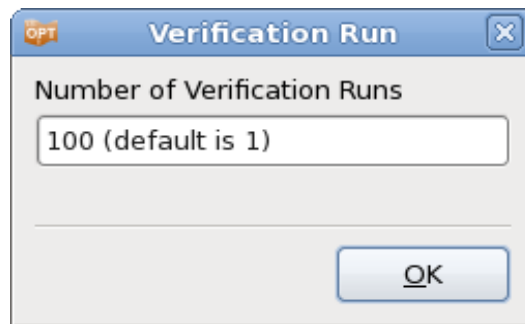
The constraints are scaled using the target values to balance the violations of the different constraints, Figure 17-67. This scaling is activated using a single check box and is only important in cases where

multiple constraints are violated as in the current problem. However, it is a good idea to apply scaling of constraints as a rule.



**Figure 17-67: 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 17-68.



**Figure 17-68: Verification Run; 100 verification runs are performed using results of the Pareto Optimal Front**

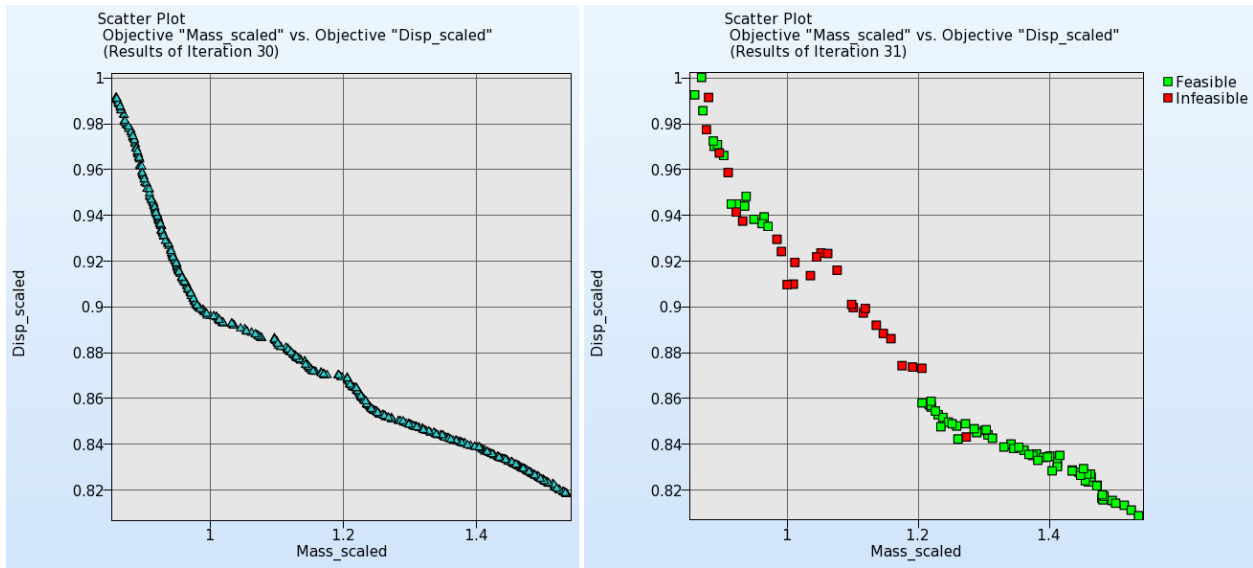
## Results

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 17-69. 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 17-70 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.

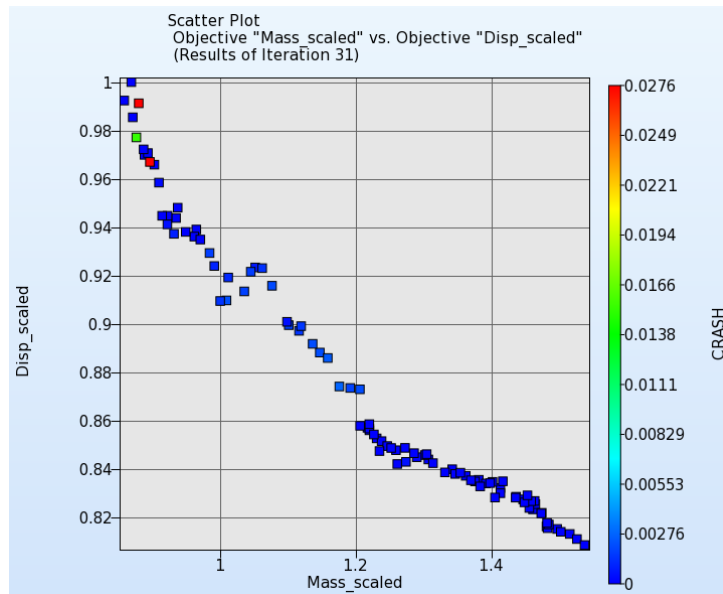
Figure 17-71 show the Self-Organizing maps plot (predicted) for the objective functions, the constraints and the variables. The conflict of the objectives is clearly visible (a blue cell in ‘Mass\_scaled’ corresponds to a

red cell in “Disp\_scaled” and *vice versa*). The corresponding ranges and influences of the variables can also be examined.

Figure 17-72 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 17-69: Pareto optimal front. Comparison of predicted results (left) and verification runs (right)**



**Figure 17-70: Verification runs color-coded by maximal constraint violation.**

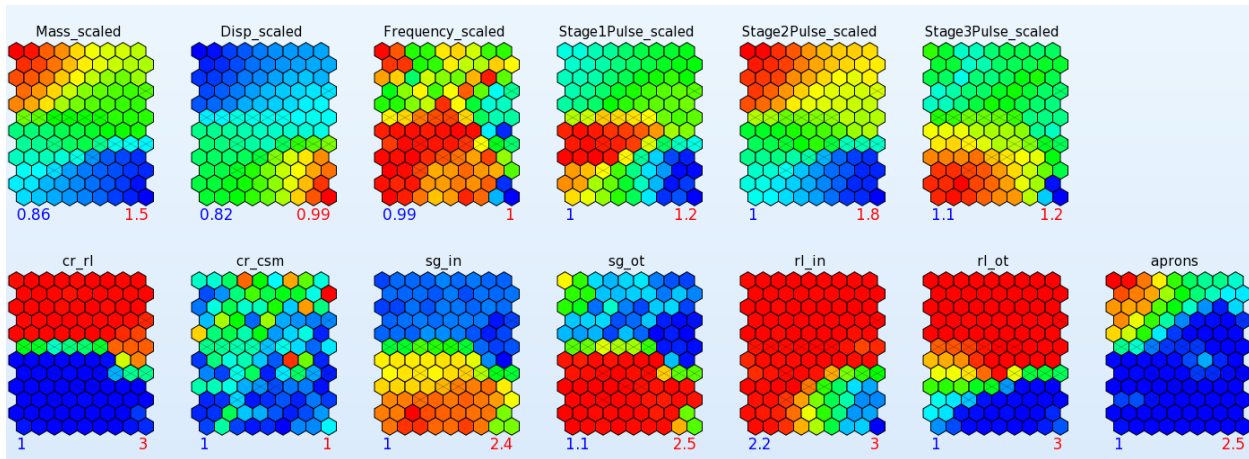


Figure 17-71: Self Organizing Maps plot of predicted Pareto optimal solutions

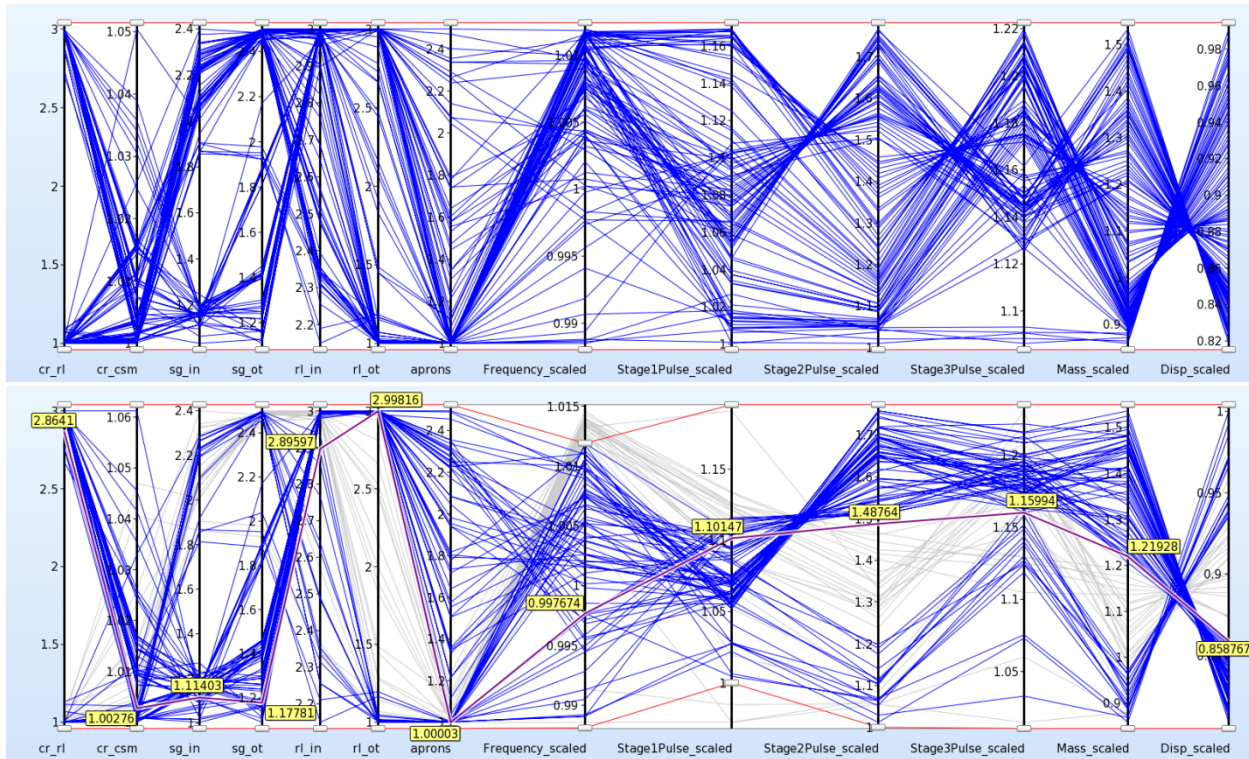
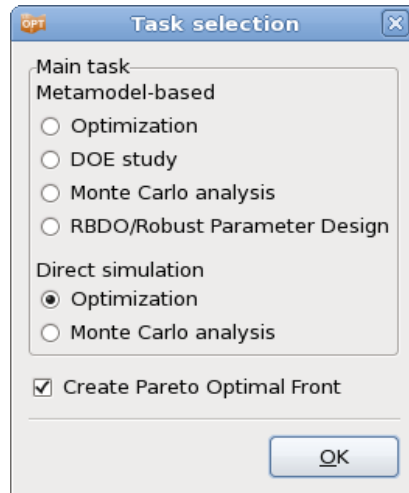


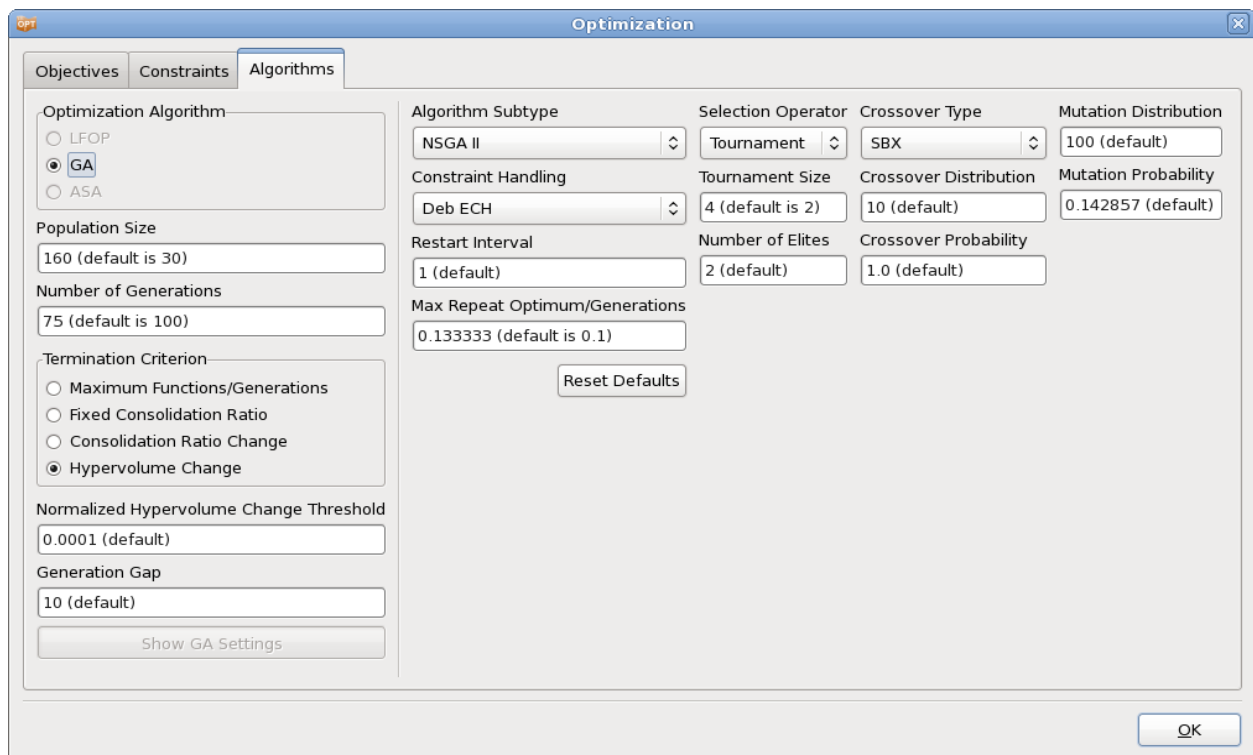
Figure 17-72: Parallel Coordinate Plot; Predicted Pareto Optimal solutions (top) and verification runs (bottom)

### 17.5.4. Multi-objective optimization using Direct GA simulation

Next, the problem is solved using direct GA simulations, Figure 17-73. The GA options used are displayed in Figure 17-74. 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*).



**Figure 17-73: Task dialog; Direct genetic algorithm**



**Figure 17-74: Options for Genetic Algorithm**

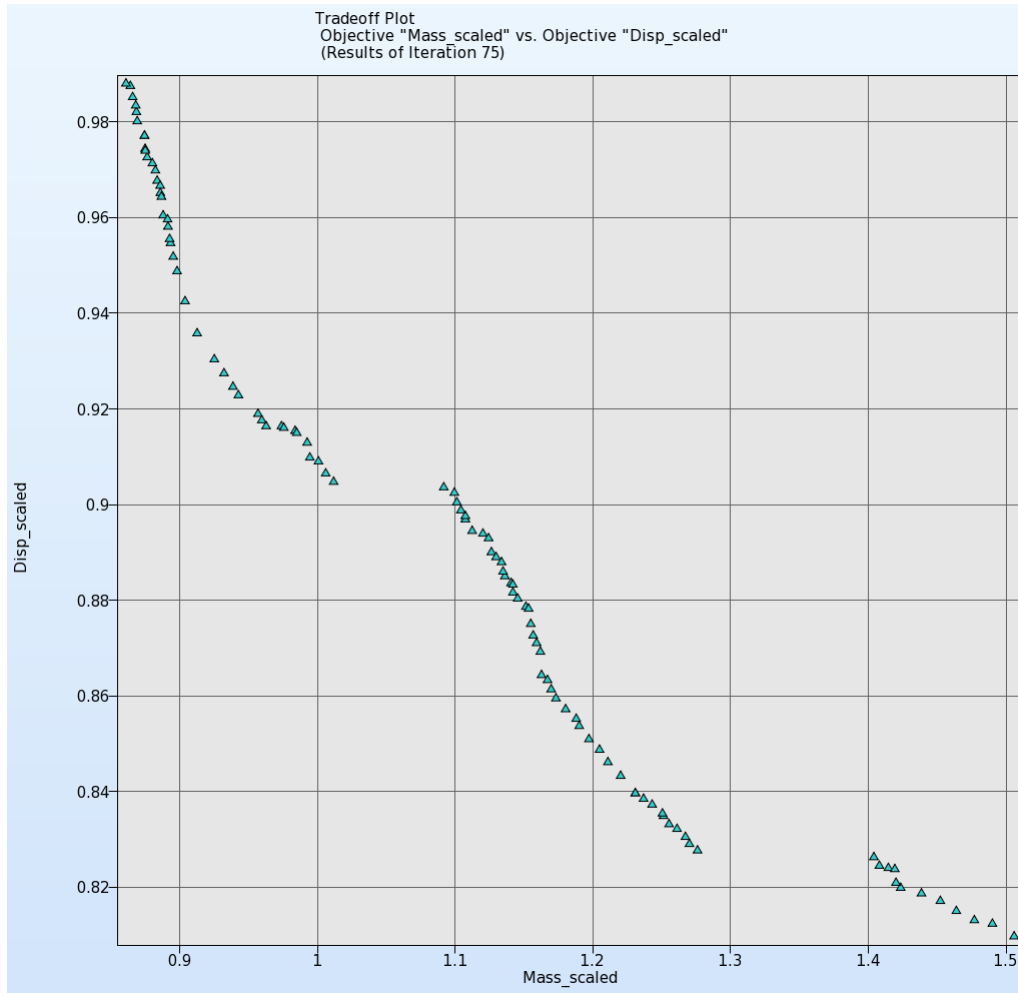
## Results

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 17-75.

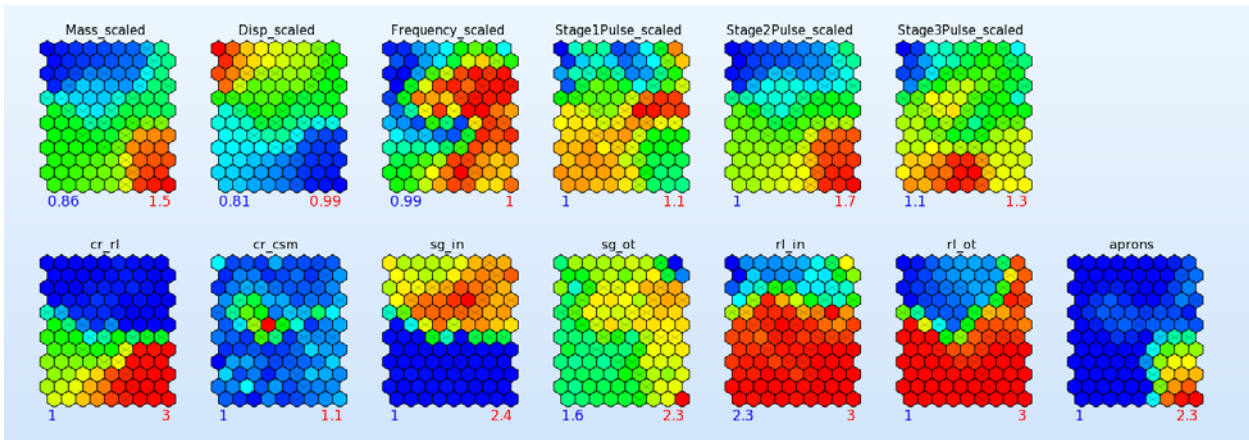
Figure 17-76 shows the Self-Organizing maps plot for the objective functions, the constraints and the variables. As in the metamodel-based optimization, the conflict of the objectives is again clearly visible while the ranges and influences of the variables can be examined.

Figure 17-77 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.

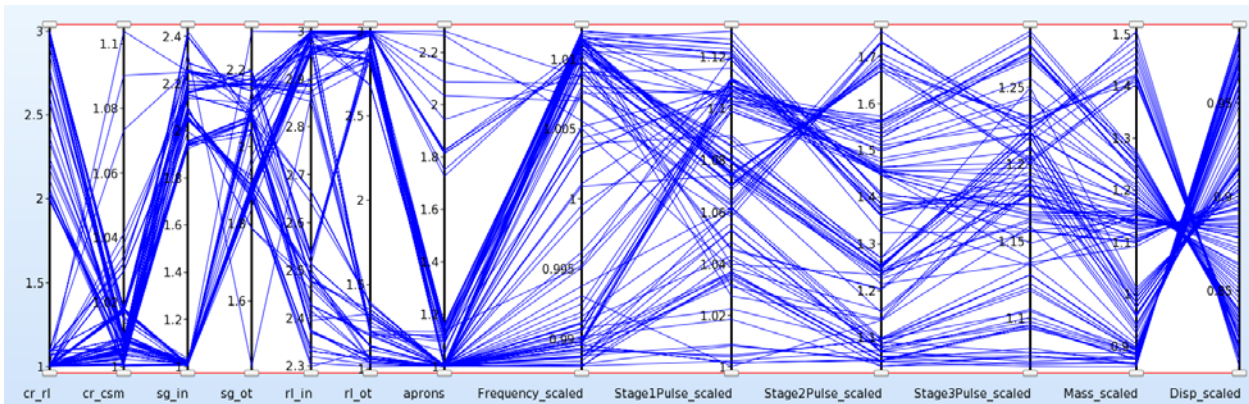


*Figure 17-75: Tradeoffs between scaled mass and intrusion (displacement).*





**Figure 17-76: Self-organizing maps plot of Pareto optimal solutions; results of last generation**



**Figure 17-77: Parallel coordinate plot of Pareto optimal solutions; results of last generation**

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 17-8.

**Table 17-8: 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

## 17.6. Knee impact with variable screening (11 variables)

(Example by courtesy of Visteon and Ford Motor Company)

This example has the following new features:

- A sequential optimization is done using a constant region of interest
- An independent parametric preprocessor is used
- The minimum of two maxima is obtained in the objective (multi-criteria or multi-objective problem). The LFOPC metamodel optimization algorithm (the default algorithm) is used for this purpose.
- A pre-processor is used for shape parameterization.

### 17.6.1. FE modeling

Figure 17-78 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 17-78 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 17-79. The three gauges and the yoke cross-sectional radius are also considered in a separate sizing (4 variable) optimization.

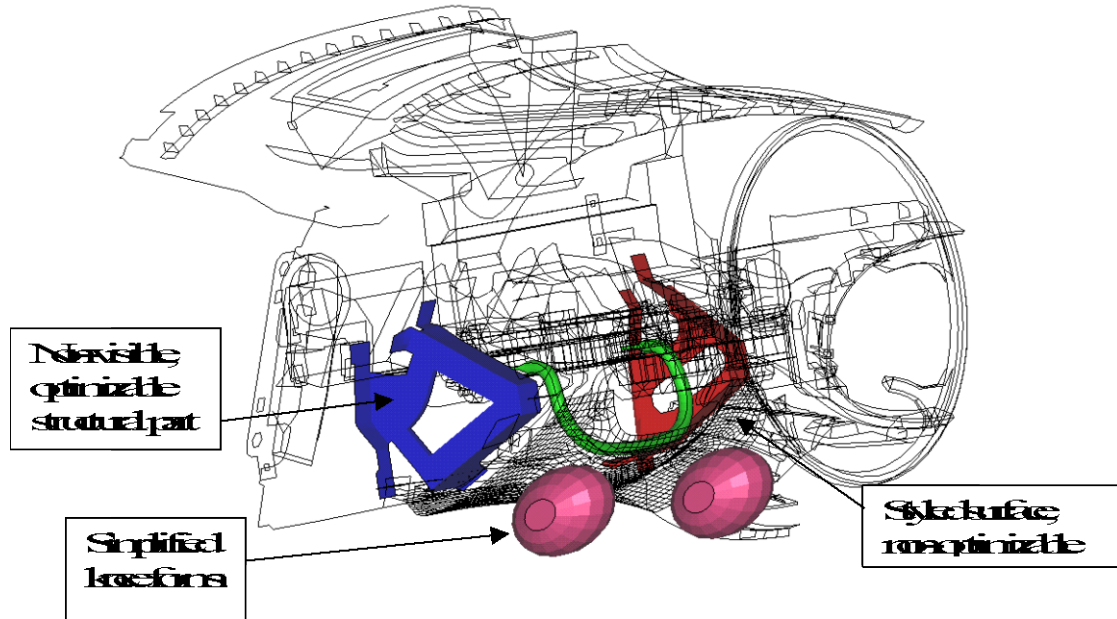


Figure 17-78: Typical instrument panel prepared for a "Bendix" component test

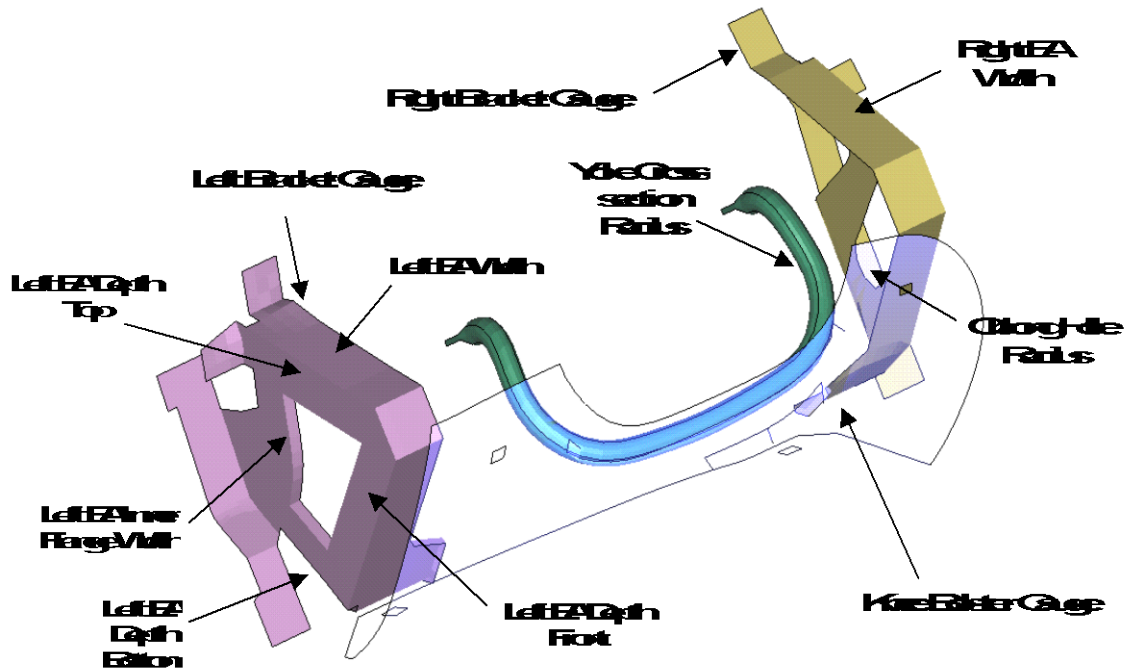


Figure 17-79: Typical major components of a knee bolster system and definition of design variables.

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.

### 17.6.2. Design formulation

The optimization problem is defined as follows:

Minimize  $(\max(\text{Knee\_Force\_Left}, \text{Knee\_Force\_Right}))$

Subject to

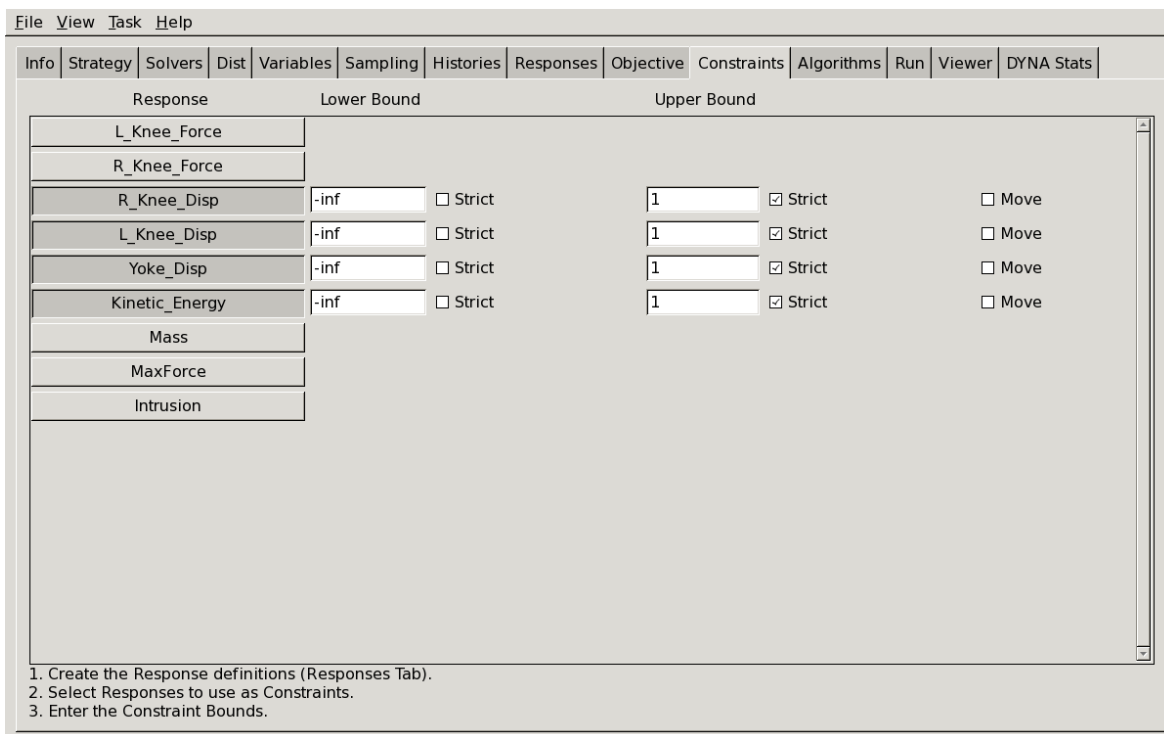
Left Knee intrusion  $< 115\text{mm}$

Right Knee intrusion  $< 115\text{mm}$

Yoke displacement  $< 85\text{mm}$

Kinetic Energy  $< 1.54\text{e}5$

Minimization over both knee forces is achieved by constraining them to impossibly low values. The LFOPC optimization algorithm must be selected since it will therefore always try to minimize the maximum knee force. The constraints other than the knee forces need to be set to “strict” so that if violations occur, only the knee forces will be violated. The “Constraints” panel of the GUI is shown below.



**Figure 17-80: Constraints for the knee bolster design problem.**

The knee forces have been filtered, SAE 60 Hz, to improve the approximation accuracy.





```
$
Metamodel Optimization Strategy SEQUENTIAL
$
  iterate param design 0.01
  iterate param objective 0.01
  iterate param stoppingtype and
$
$$ OPTIMIZATION ALGORITHM
$
Optimization Algorithm hybrid simulated annealing
$
$ SA PARAMETERS
$
  sa maximum simulation 20000
  Use GSA
  Set GSA Resolution 100000
$
$ JOB INFO
$
$ run baseline
  iterate 10
STOP
```

#### 17.6.4. Variable screening

A single iteration is done with a linear approximation to generate the ANOVA and Sobol’s global sensitivity analysis charts. The charts are shown in the figure below. Note the large confidence intervals (low confidence levels) on some of the responses, especially the Left Knee Force and Yoke Displacement.

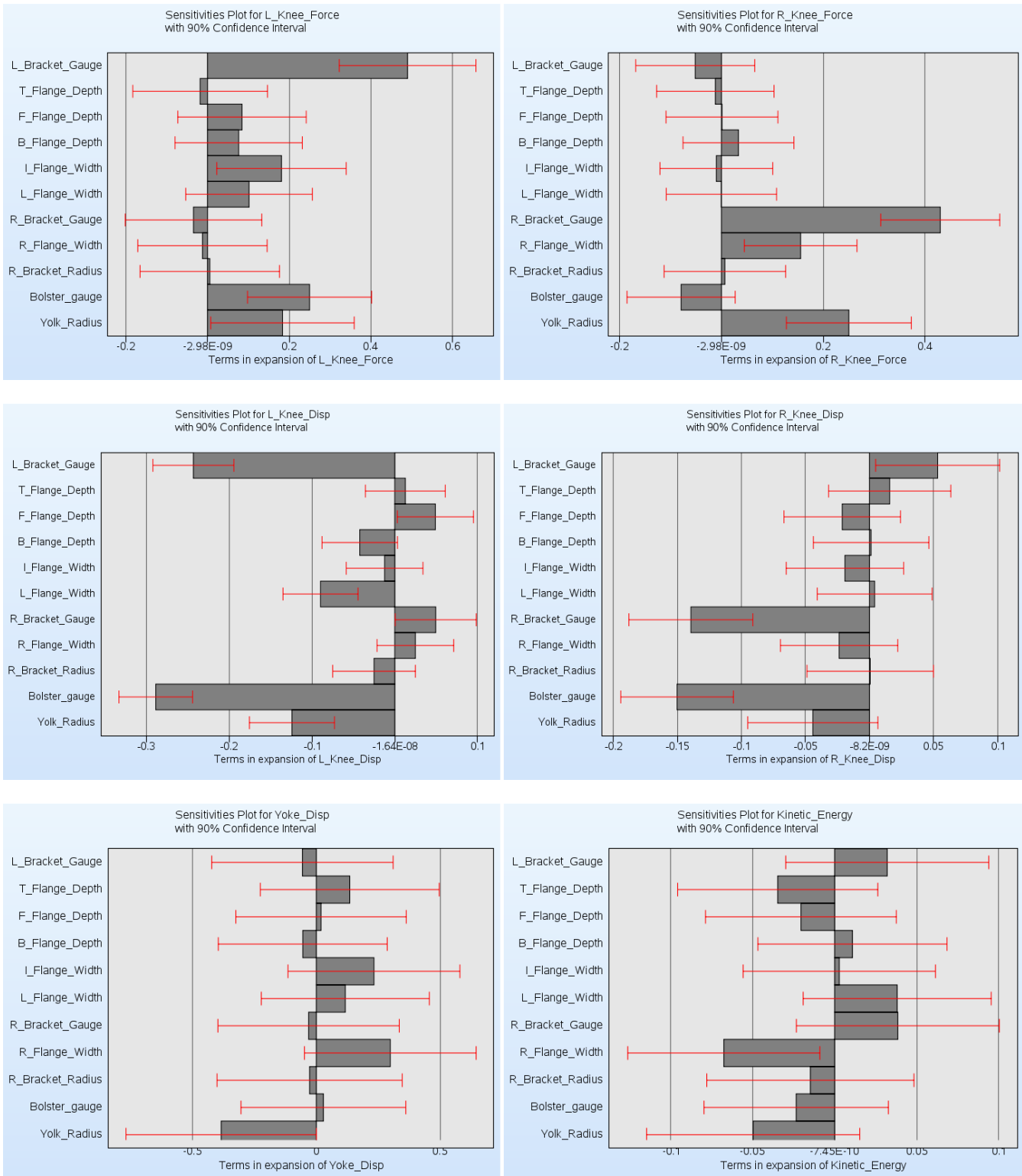
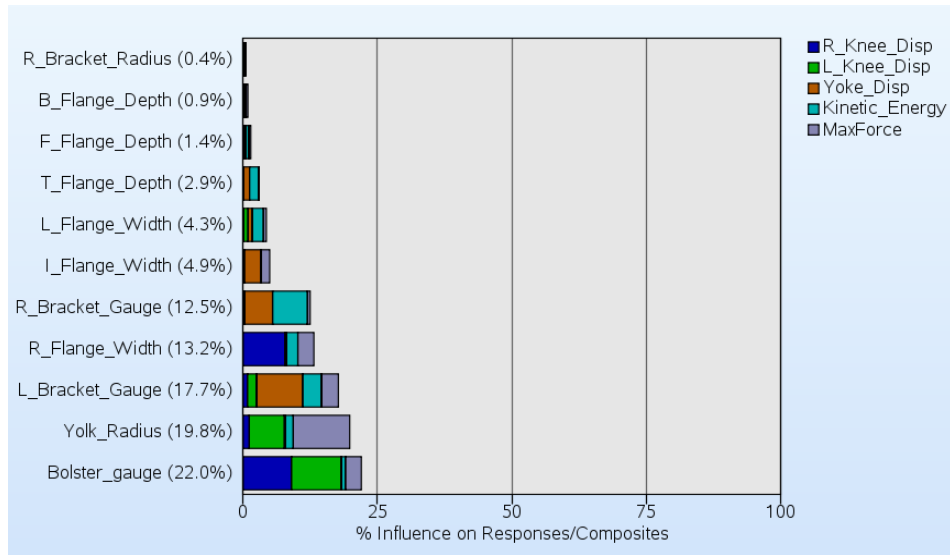


Figure 17-81: ANOVA plots for objectives and constraints of knee-bolster design problem.





**Figure 17-82: Global sensitivity analysis of objectives and constraints.**

The six variables chosen from the charts are:

$\mathbf{x}=[\text{Bolster\_gauge}, \text{Yoke\_Radius}, \text{L\_Bracket\_Gauge}, \text{R\_Bracket\_Gauge}, \text{R\_Flange\_Width}, \text{I\_Flange\_Depth}]^T$ ;

The changes in the input file are as follows:

```
variables 6
Variable 'L_Bracket_Gauge' .7
  Lower bound variable 'L_Bracket_Gauge' .7
  Upper bound variable 'L_Bracket_Gauge' 3.
Variable 'I_Flange_Width' 5.
  Lower bound variable 'I_Flange_Width' 5.
  Upper bound variable 'I_Flange_Width' 25.
Variable 'R_Bracket_Gauge' .7
  Lower bound variable 'R_Bracket_Gauge' .7
  Upper bound variable 'R_Bracket_Gauge' 3.
Variable 'R_Flange_Width' 20.
  Lower bound variable 'R_Flange_Width' 20.
  Upper bound variable 'R_Flange_Width' 50.
Variable 'Bolster_gauge' 3.6
  Lower bound variable 'Bolster_gauge' 1.
  Upper bound variable 'Bolster_gauge' 6.
Variable 'Yolk_Radius' 2.
  Lower bound variable 'Yolk_Radius' 2.
  Upper bound variable 'Yolk_Radius' 8.
$
$ CONSTANTS
$
constants 5
Constant 'T_Flange_Depth' 50
Constant 'F_Flange_Depth' 20
Constant 'B_Flange_Depth' 15
Constant 'L_Flange_Width' 20
Constant 'R_Bracket_Radius' 10
```

### 17.6.5. Optimization strategy

In contrast to the strategy of the full vehicle example, a sequential strategy in which, the region of interest is kept constant, is chosen. 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. LFOPC (the default algorithm) is chosen as the core solver because of the requirement to minimize the maximum knee force.

### 17.6.6. Optimization history results

The plots below show optimization history of the objectives, constraints and some responses. The simulations converged after three iterations. While the baseline design resulted in a maximum force of 1.315 units, the optimum design resulted in only 0.966 units of maximum force. Though intermediate results were infeasible, the final design was feasible.

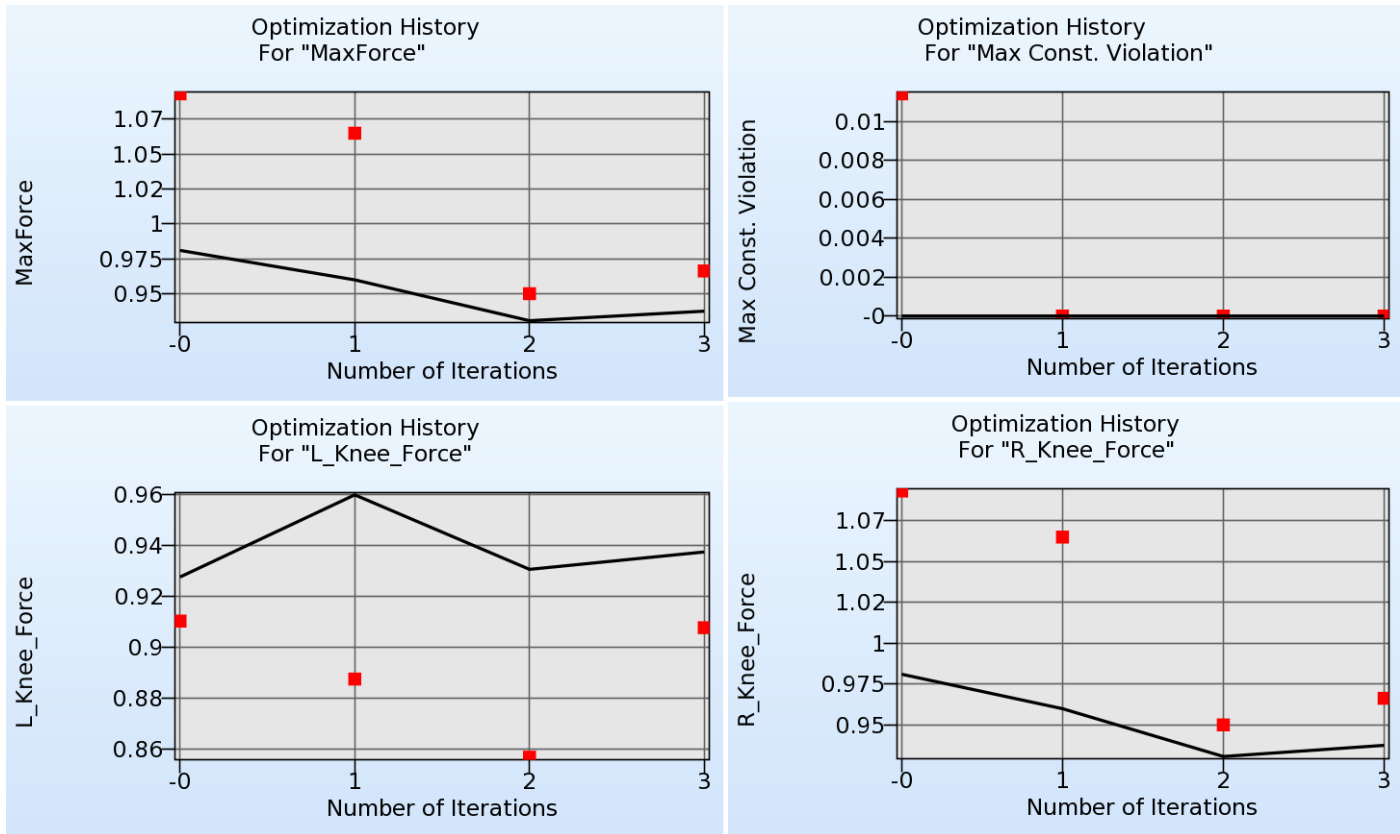


Figure 17-83: Optimization history of objectives and maximum constraint violations.

### 17.6.7. Summary of results

The following is an edited version of the `lsopt_report` file (also viewable by selecting View→Summary).

```

LS-OPT Version      : 4.1
LS-OPT Revision    : 57436
LS-OPT Version Date : Jan 6, 2010

Project Command File : com_srsn

*****
Problem description:
Knee impact with 11 variables
*****

Continuous Variables
-----|
Variable Name   Lower Bound  Upper Bound |
-----|-----|-----|

```

L_Bracket_Gauge	0.7	3
I_Flange_Width	5	25
R_Bracket_Gauge	0.7	3
R_Flange_Width	20	50
Bolster_gauge	1	6
Yolk_Radius	2	8

CONSTRAINT FUNCTIONS

Constraint name	Lower Bound	Upper Bound
R_Knee_Disp	-1e+30	1
L_Knee_Disp	-1e+30	1
Yoke_Disp	-1e+30	1
Kinetic_Energy	-1e+30	1

OPTIMIZATION ALGORITHM

Method ..... Metamodel-based Optimization  
 Strategy ..... Sequential with Domain Reduction  
 Optimization Algorithm ..... Hybrid SA (Simulated Annealing + LFOPC)

Evaluating Starting Design  
 ITERATION 1

COMPUTED vs. PREDICTED

Using Metamodel of Iteration 1

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound
L_Bracket_Gauge	0.7	0.7	3 Active
I_Flange_Width	5	5	25 Active
R_Bracket_Gauge	0.7	0.7	3 Active
R_Flange_Width	20	20	50 Active
Bolster_gauge	1	3.6	6
Yolk_Radius	2	2	8 Active

RESPONSE VALUES:

Name	Computed	Predicted
L_Knee_Force	0.9102	0.9275
R_Knee_Force	1.093	0.981
R_Knee_Disp	0.8355	0.865
L_Knee_Disp	1.011	0.9917
Yoke_Disp	0.609	0.6354
Kinetic_Energy	0.3851	0.3586
Mass	0.6152	0.5795

COMPOSITE FUNCTIONS:

COMPOSITE NAME	Computed	Predicted
MaxForce	1.093	0.981
Intrusion	0.9234	0.9284

OBJECTIVE:

Computed Value = 1.093  
 Predicted Value = 0.981

OBJECTIVE VALUES:

Name	Computed	Predicted	WT.
------	----------	-----------	-----

MaxForce	1.093	0.981	1
----------	-------	-------	---

CONSTRAINT VALUES:

Name	Computed	Predicted	Lower	Upper	Viol?
R_Knee_Displacement	0.8355	0.865	-1e+30		1
L_Knee_Displacement	1.011	0.9917	-1e+30		1
Yoke_Displacement	0.609	0.6354	-1e+30		1
Kinetic_Energy	0.3851	0.3586	-1e+30		1

CONSTRAINT VIOLATIONS:

Name	Computed Violation		Predicted Violation	
	Lower	Upper	Lower	Upper
R_Knee_Displacement	-	-	-	-
L_Knee_Displacement	-	0.01141	-	-
Yoke_Displacement	-	-	-	-
Kinetic_Energy	-	-	-	-

MAXIMUM VIOLATION:

Quantity	Computed		Predicted	
	Constraint	Value	Constraint	Value
Maximum Violation	L_Knee_Displacement	0.01141	R_Knee_Displacement	0
Smallest Margin	L_Knee_Displacement	0.01141	L_Knee_Displacement	0.008291

ERROR MEASURES FOR RESPONSES

ITERATION 3

Number of points = 42

Response Name	Metamodel type	RMS Error	RMS Error (% of mean)	Maximum Residual	Sq. Root PRESS	Sq. Root PRESS (% of mean)	R-Sq.	R-Sq. Predicted
L_Knee_Force	RBF Net	0.0896	6.93	0.249	0.143	11	0.891	0.818
R_Knee_Force	RBF Net	0.0402	3.18	0.093	0.0654	5.17	0.93	0.895
R_Knee_Displacement	RBF Net	0.0232	2.8	0.0709	0.0322	3.89	0.811	0.732
L_Knee_Displacement	RBF Net	0.0132	1.67	0.035	0.041	5.2	0.98	0.923
Yoke_Displacement	RBF Net	0.202	30.3	0.691	0.244	36.6	0.281	0.167
Kinetic_Energy	RBF Net	0.0459	12.7	0.107	0.0531	14.6	0.434	0.461
Mass	RBF Net	0.0181	1.74	0.0514	0.0254	2.45	0.989	0.992

GLOBAL SENSITIVITY ESTIMATES

Cumulative Variable Influence [1]

Variable	AVERAGED		MAXIMUM	
	Main Effect	Total Effect	Main Effect	Total Effect
L_Bracket_Gauge	0.025	0.438	0.0511	0.591
I_Flange_Width	0.00728	0.15	0.0117	0.179
R_Bracket_Gauge	0.0484	0.552	0.11	0.82
R_Flange_Width	0.0383	0.498	0.0676	0.578
Bolster_gauge	0.045	0.445	0.129	0.669
Yolk_Radius	0.025	0.406	0.0498	0.515

FINAL DESIGN  
ITERATION 4

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound	
L_Bracket_Gauge	0.7	0.7	3	Active
I_Flange_Width	5	24.97	25	
R_Bracket_Gauge	0.7	0.7015	3	Active
R_Flange_Width	20	20	50	Active
Bolster_gauge	1	5.156	6	
Yolk_Radius	2	2	8	Active

RESPONSE VALUES:

Name	Computed	Predicted
L_Knee_Force	0.9076	0.9374
R_Knee_Force	0.9662	0.9374
R_Knee_Disp	0.8583	0.8518
L_Knee_Disp	0.8975	0.9037
Yoke_Disp	0.228	0.581
Kinetic_Energy	0.3965	0.3169
Mass	0.8054	0.8007

COMPOSITE FUNCTIONS:

COMPOSITE NAME	Computed	Predicted
MaxForce	0.9662	0.9374
Intrusion	0.8779	0.8777

OBJECTIVE:

Computed Value = 0.9662  
Predicted Value = 0.9374

OBJECTIVE VALUES:

Name	Computed	Predicted	WT.
MaxForce	0.9662	0.9374	1

CONSTRAINT VALUES:

Name	Computed	Predicted	Lower	Upper	Viol?
R_Knee_Disp	0.8583	0.8518	-1e+30		1
L_Knee_Disp	0.8975	0.9037	-1e+30		1
Yoke_Disp	0.228	0.581	-1e+30		1
Kinetic_Energy	0.3965	0.3169	-1e+30		1

CONSTRAINT VIOLATIONS:

Name	Computed Violation		Predicted Violation	
	Lower	Upper	Lower	Upper
R_Knee_Disp	-	-	-	-
L_Knee_Disp	-	-	-	-
Yoke_Disp	-	-	-	-
Kinetic_Energy	-	-	-	-

MAXIMUM VIOLATION:

Quantity	Computed		Predicted	
	Constraint	Value	Constraint	Value

Maximum Violation	R_Knee_Disp	0	R_Knee_Disp	0
Smallest Margin	L_Knee_Disp	0.1025	L_Knee_Disp	0.09635

ANALYSIS COMPLETED

Tue Jan 12 12:17:44 2010

## 17.7. Shape optimization of a front rail using ANSA<sup>12</sup> and $\mu$ ETA<sup>13</sup>

This example has the following features:

- Optimization of a 3-stage process chain using ANSA, LS-DYNA and  $\mu$ ETA
- Shape optimization using the ANSA morphing tool
- Discrete variables
- Result extraction using  $\mu$ ETA

### 17.7.1. Problem Statement

The problem is of a front rail crash simulation. Embosses are to be used to

Minimize acceleration

subject to

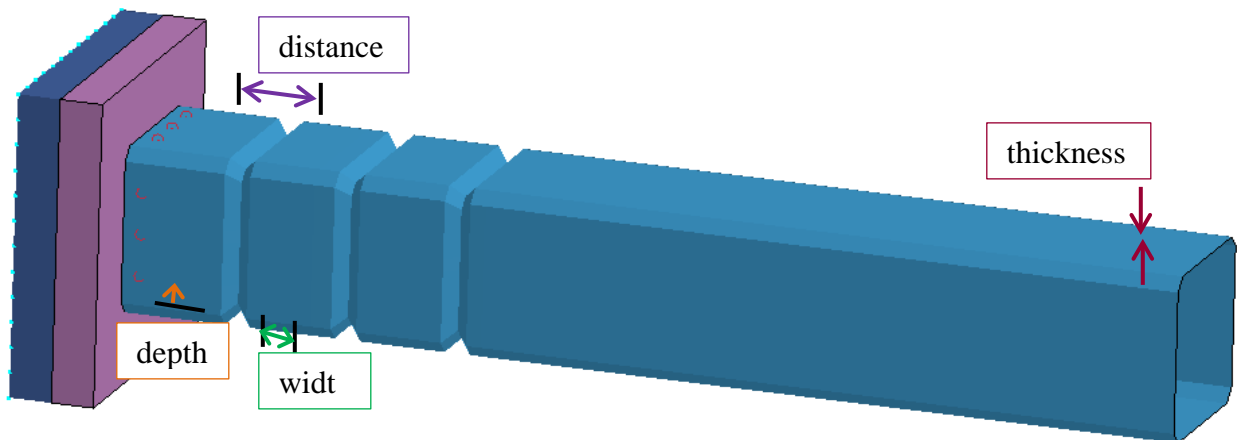
mass < 1.8

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 17-84. Thickness and width are defined as discrete parameters.

<sup>12</sup> BETA CAE Systems S.A.

<sup>13</sup> BETA CAE Systems S.A.

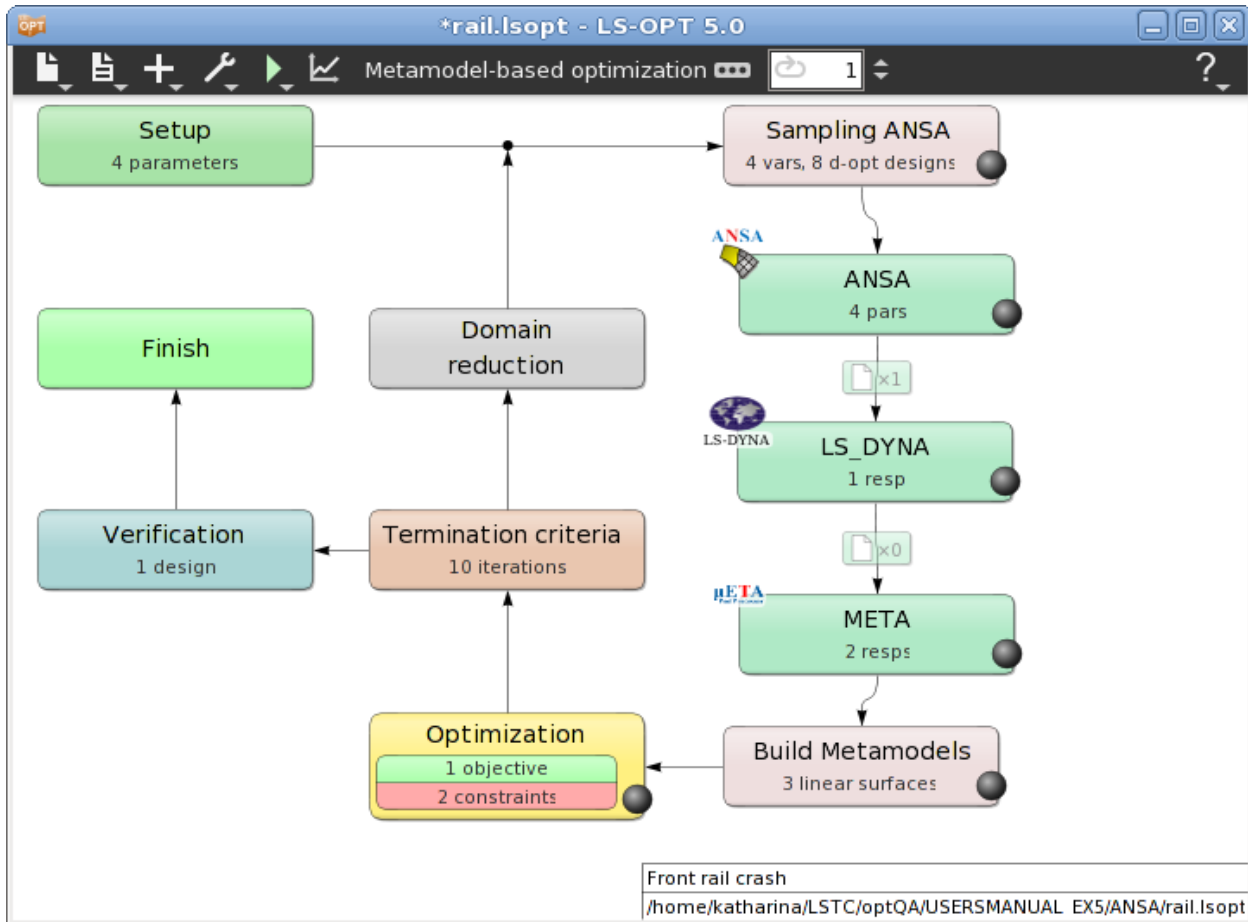


*Figure 17-84: Rail with embosses.*

### 17.7.2. 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, 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 17-85 displays the main LS-OPT GUI window visualizing the optimization process and the ANSA – LS-DYNA –  $\mu$ ETA process chain.



**Figure 17-85: Main LS-OPT GUI window; Metamodel based design optimization, Strategy SRSM; Optimization of a process chain with 3 stages.**

Figure 17-86 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 17-87.

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 17-88. 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 17-89.



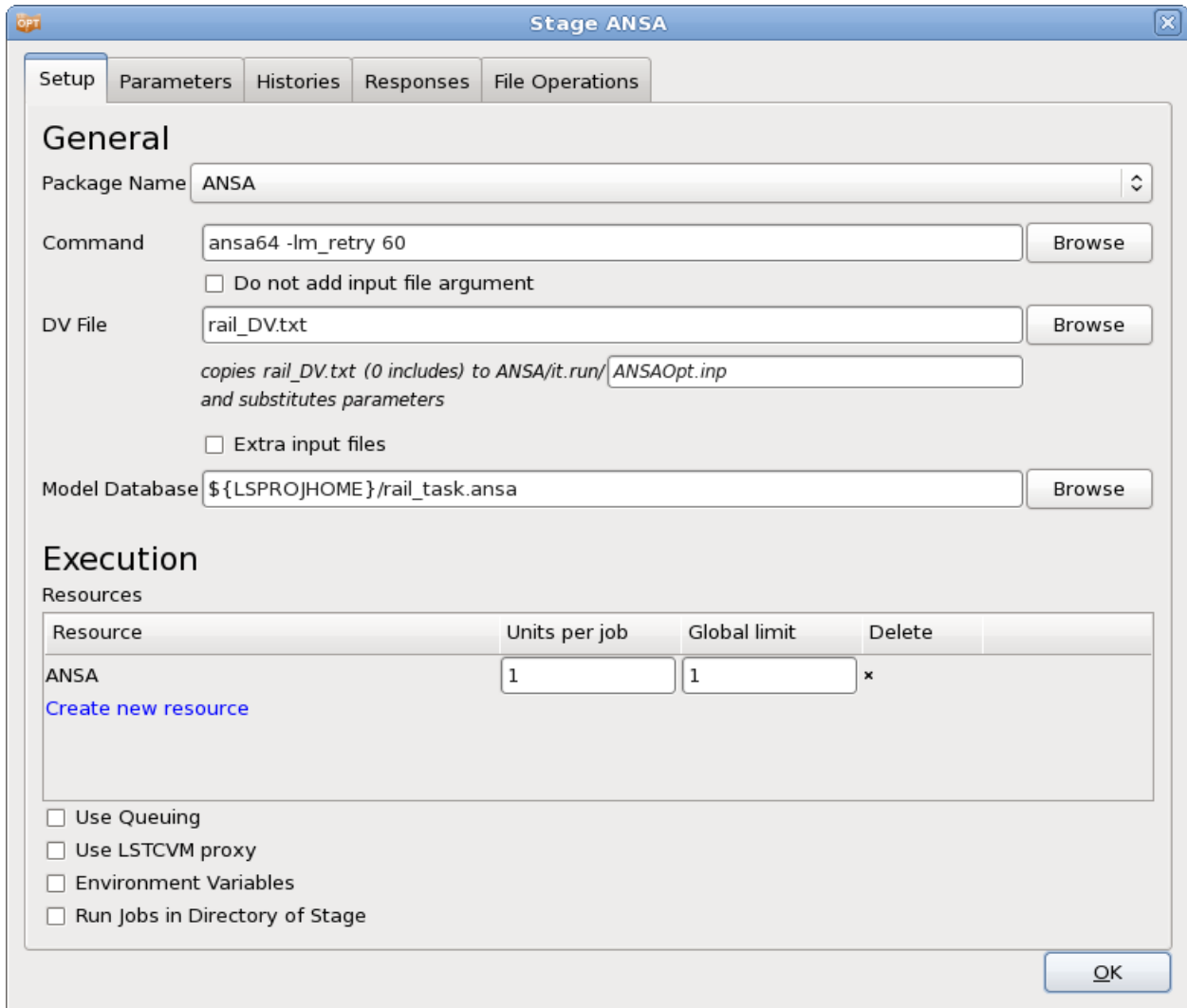


Figure 17-86: Stage dialog interfacing with ANSA

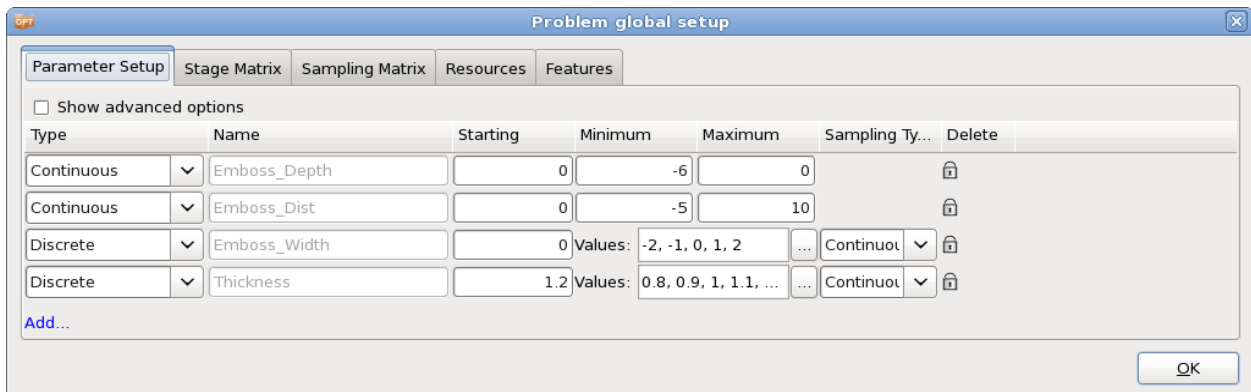
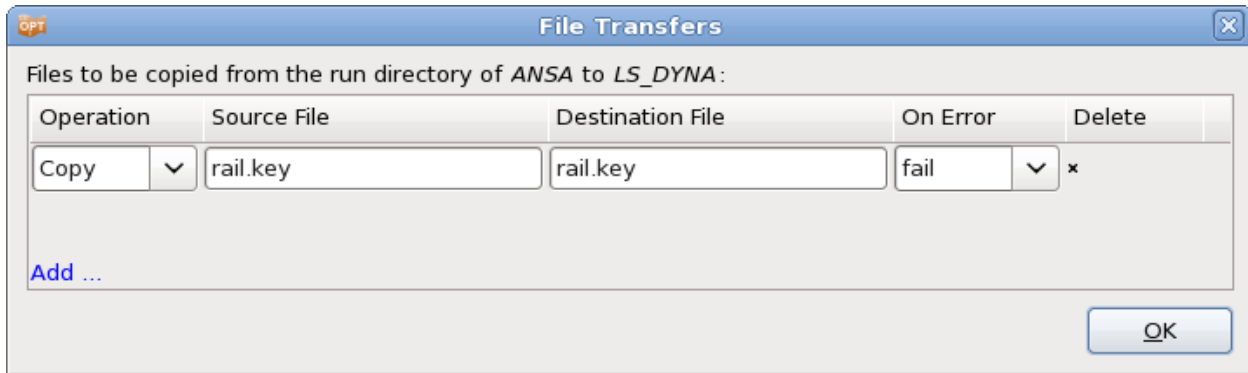
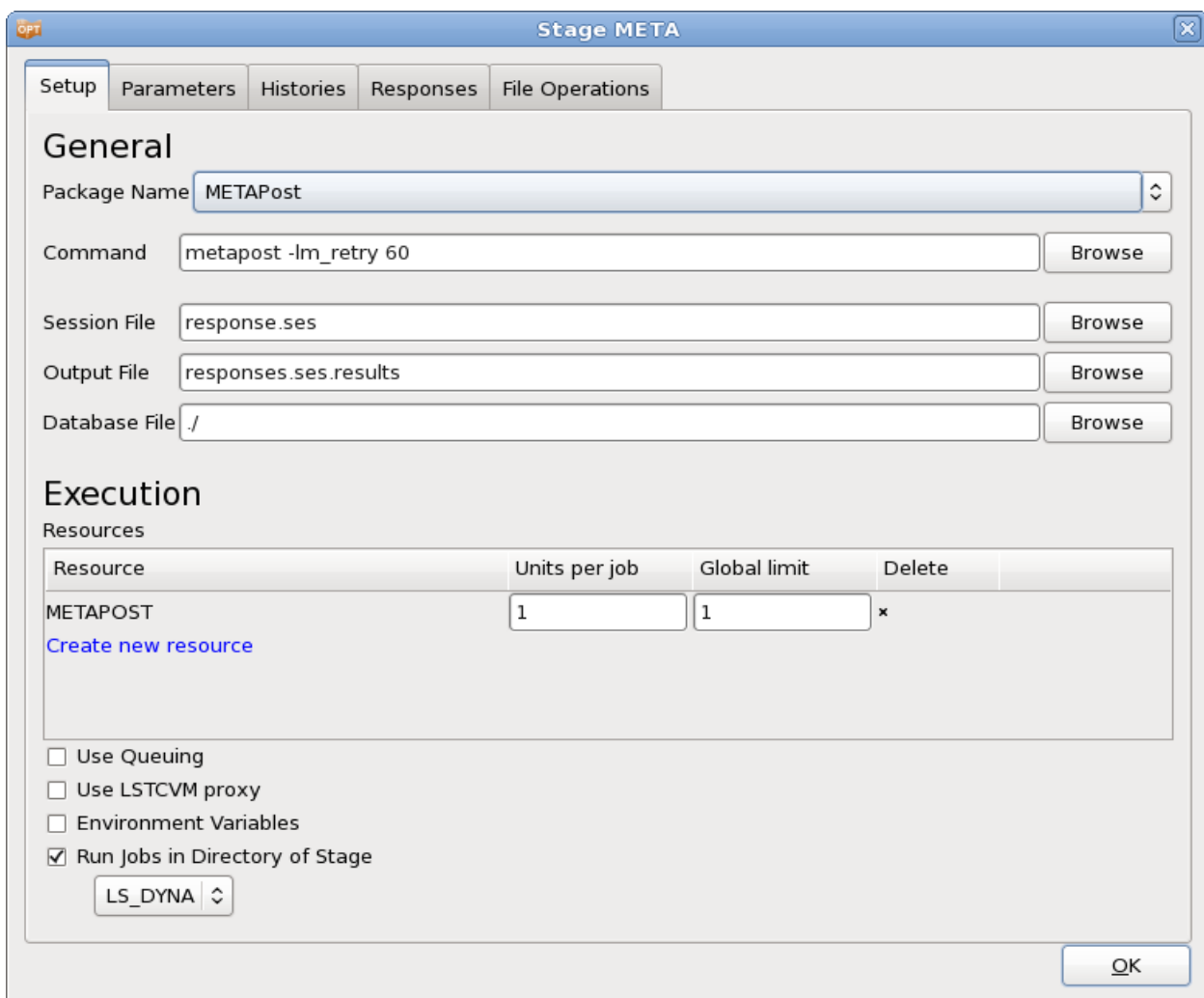


Figure 17-87: Parameter Setup; variables, values and ranges are imported from the ANSA DV file



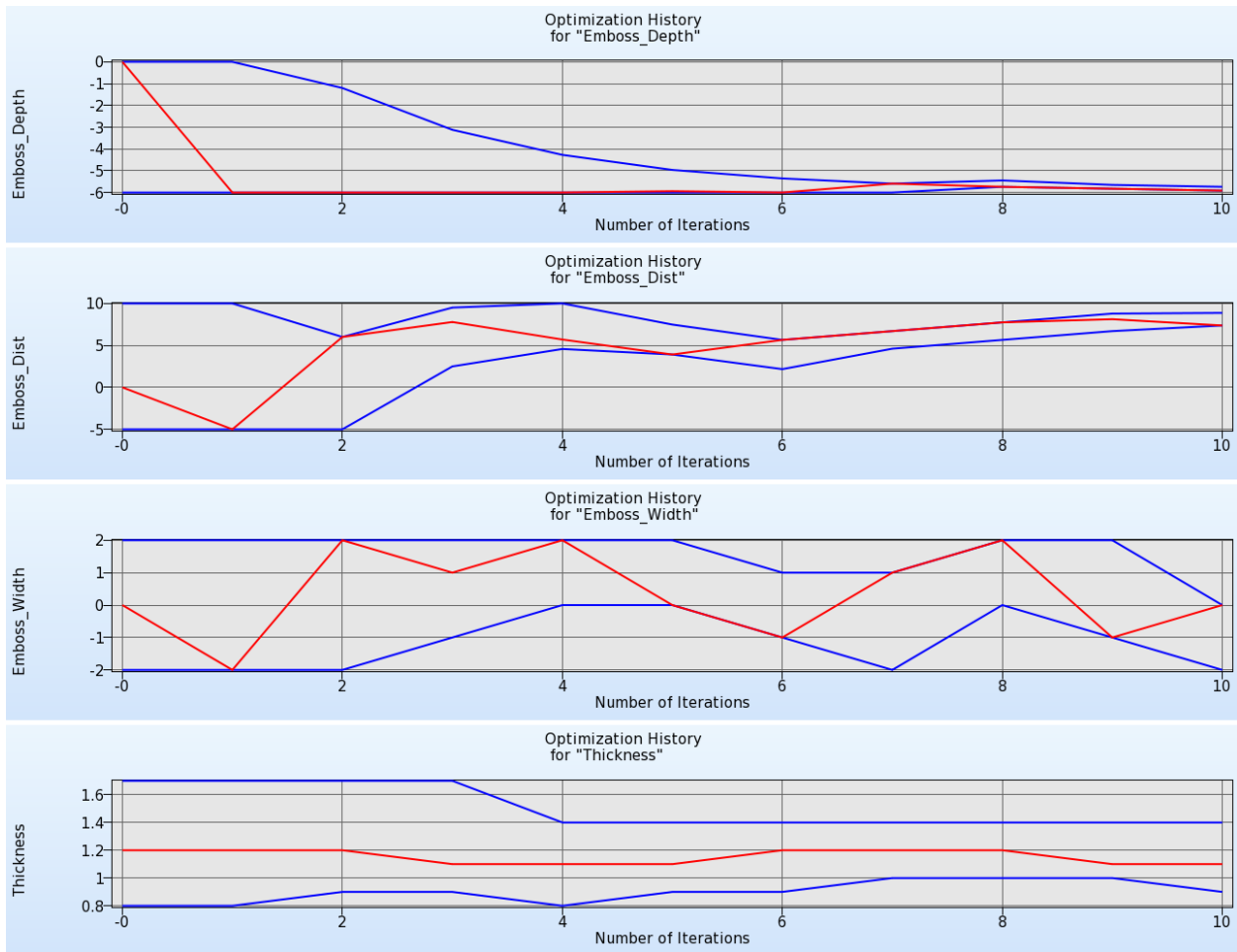
**Figure 17-88:** File Transfer dialog; the ANSA output file *rail.key* is copied to the respective LS-DYNA run directory.



**Figure 17-89:** Stage dialog interfacing with  $\mu$ ETApst.  $\mu$ ETApst is run in the respective LS-DYNA run directory, since the results are extracted from the LS-DYNA output.

### 17.7.3. Results

Figure 17-90 and Figure 17-91 display the optimization history of the variables and the responses, respectively. There is no convergence for the variable `Emboss_Width`, but as the Sensitivities Plot in Figure 17-92 shows, this variable is not sensitive. Due to metamodel inaccuracy, the final design is infeasible displayed in Figure 17-93, but e.g. the optimal value of iteration 9 is feasible, and the acceleration value is similar.



*Figure 17-90: Optimization History of variables.*

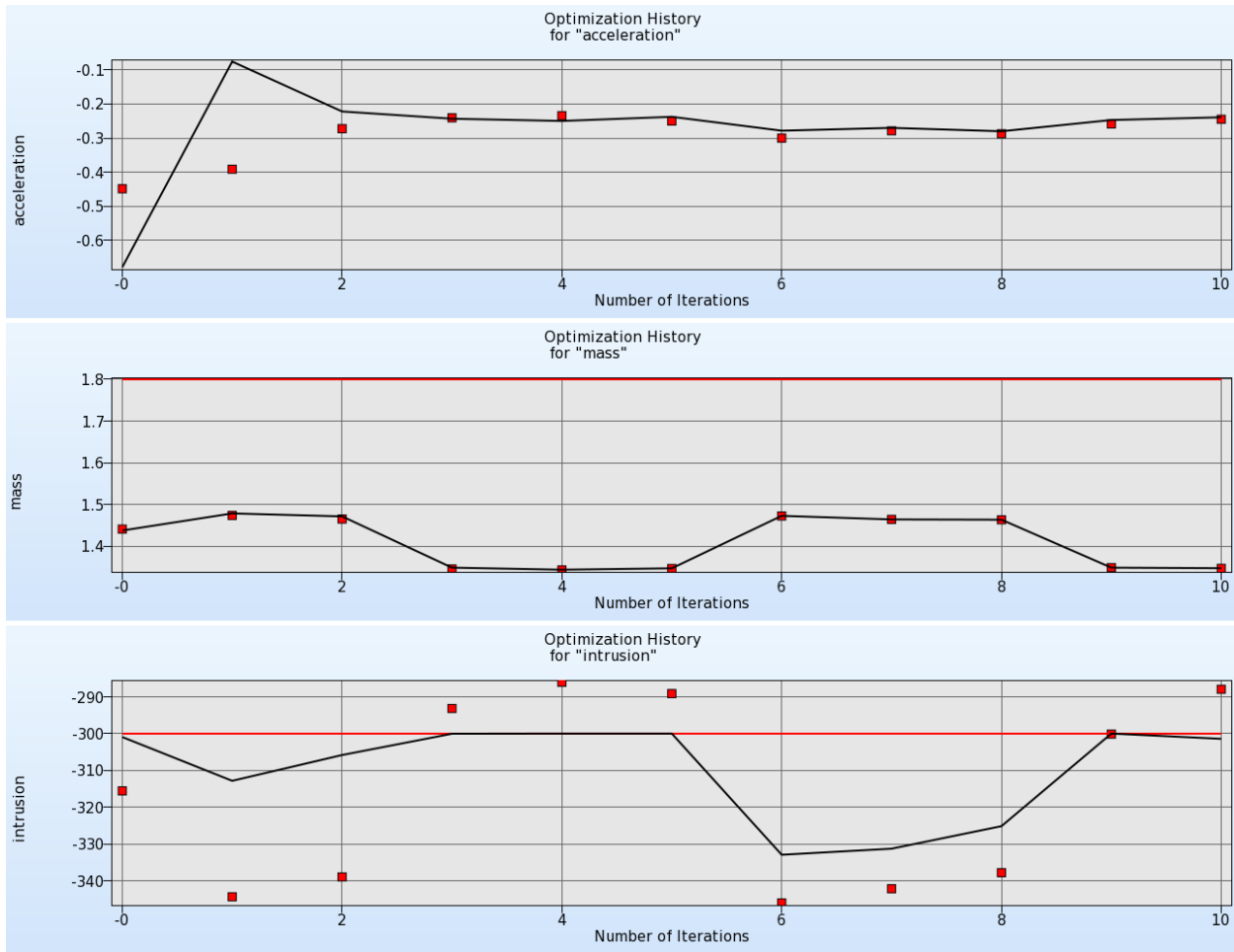


Figure 17-91: Optimization History of objective and constraints.

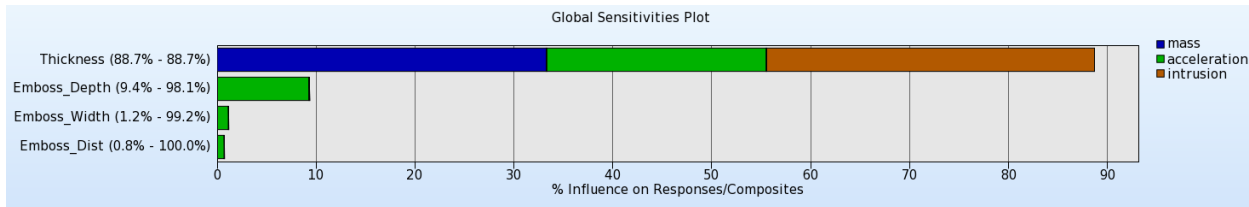


Figure 17-92: Global Sensitivities.



*Figure 17-93: Final design. Optimum of iteration 10.*

## 17.8. Optimization with analytical design sensitivities

This example has the following features:

- Using analytical design sensitivities for optimization
- Defining a user-defined solver

### 17.8.1. Problem Statement

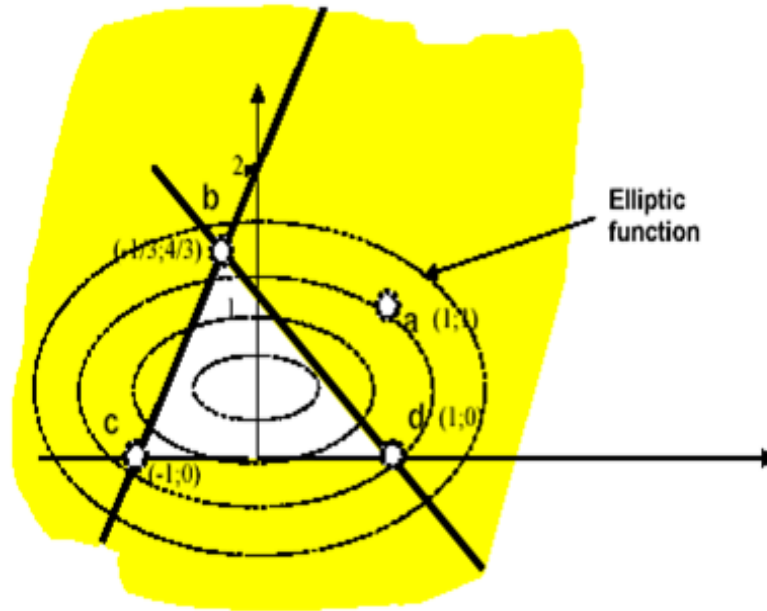
The optimization problem to be solved is

$$\max(x_1^2 + 4(x_2 - 0.5)^2)$$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ -2x_1 + x_2 &\leq 2 \\ x_2 &\geq 0. \end{aligned}$$

Figure 17-94 displays the objective and constraint functions.



**Figure 17-94: Objective and constraint functions.**

This example demonstrates how analytical gradients (Section 0) provided by a solver can be used for optimization using the SLP algorithm and the domain reduction scheme [5] (Section 22.6). 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";

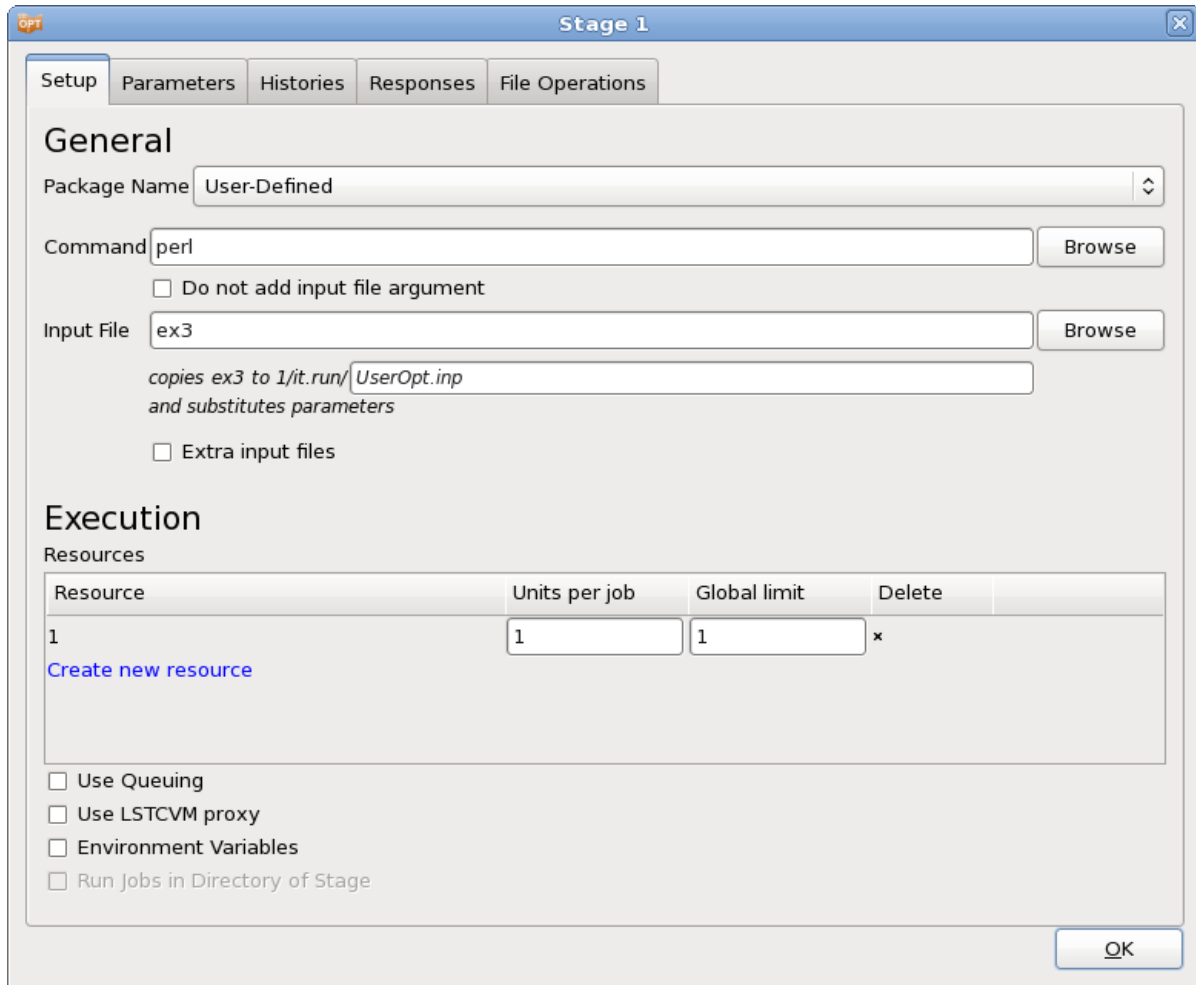
```

### 17.8.2. Solution

Figure 17-95 shows the stage dialog defining the user-defined solver.

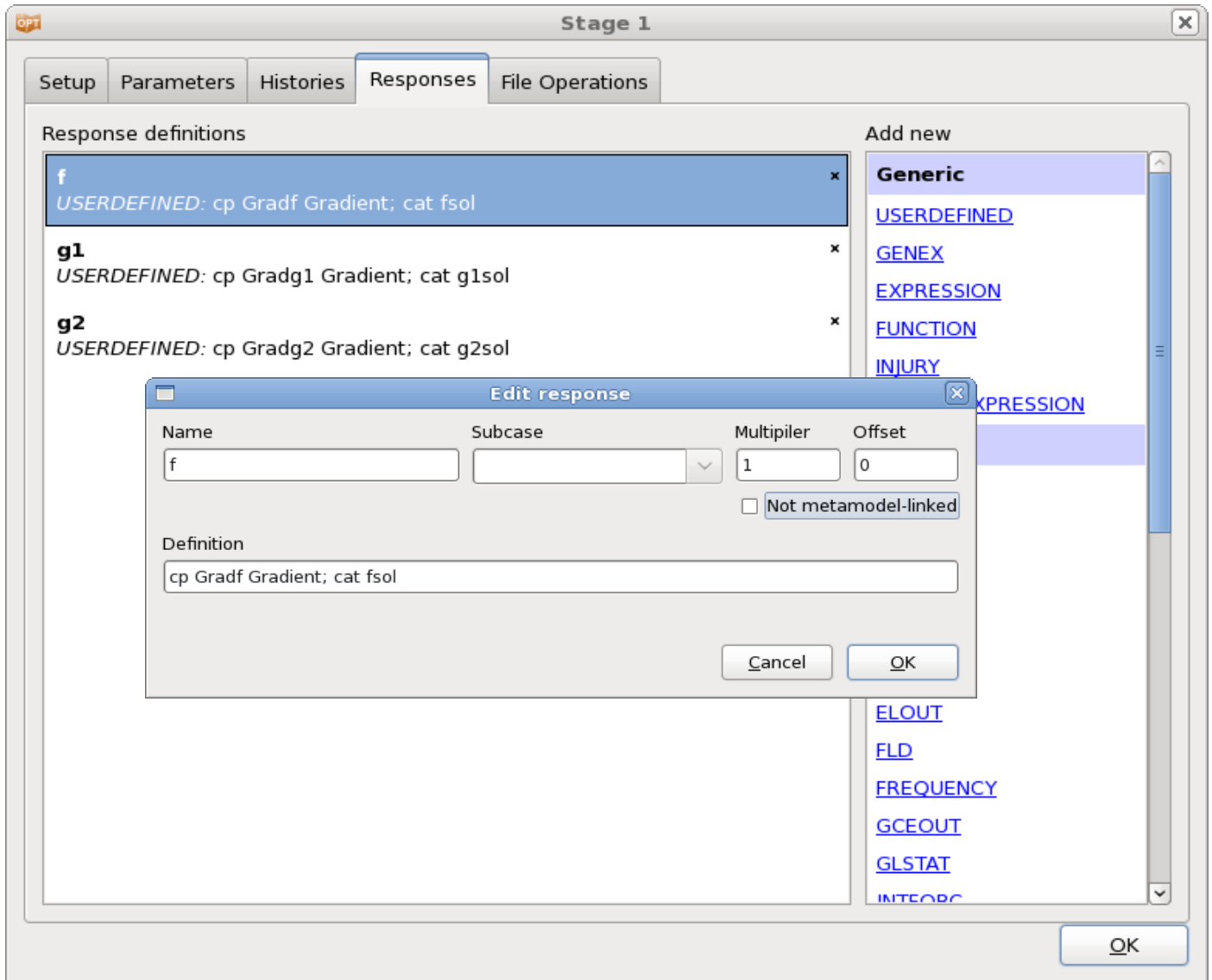
Figure 17-96 displays the response definitions. The gradient files generated by the Perl program need to be copied to a file called `Gradient`, the calculated response values need to be output to standard output.

Use the metamodel type *Sensitivity* to use analytical gradients for optimization, Figure 17-97.

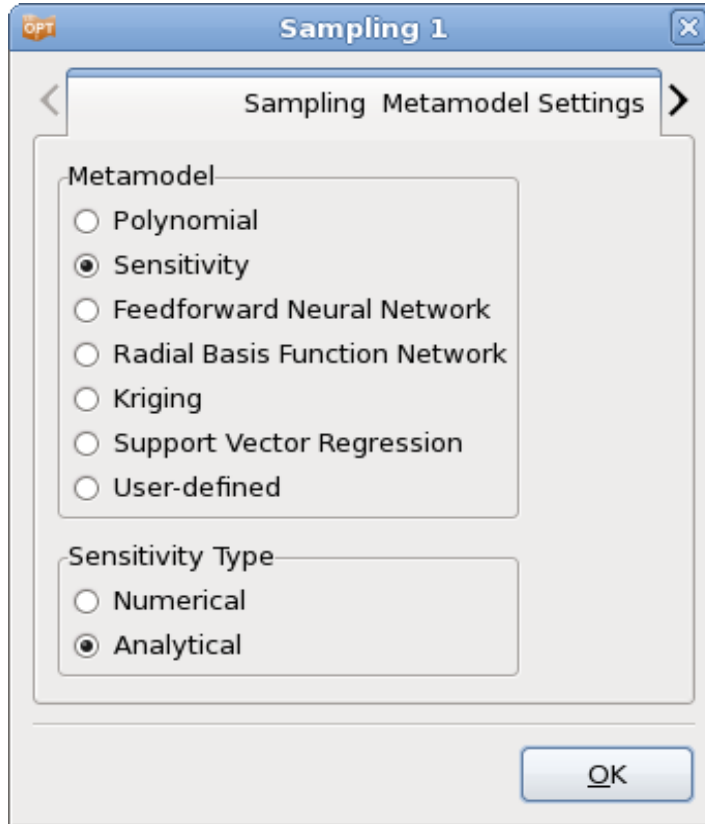


*Figure 17-95: Defining a user-defined solver*





*Figure 17-96: User-defined results extraction.*



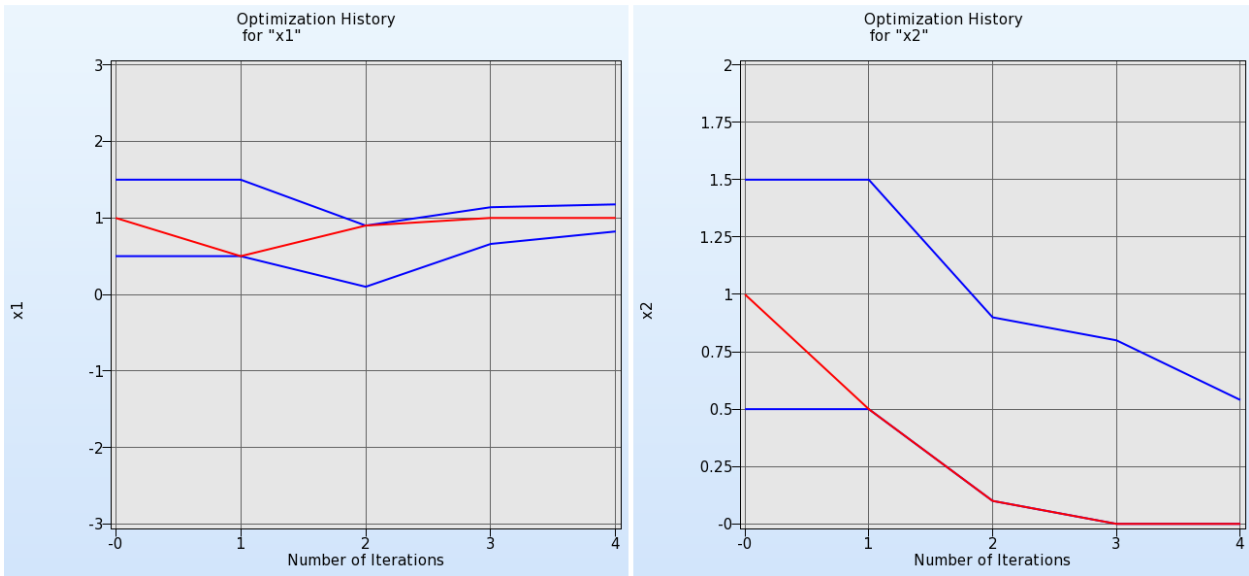
*Figure 17-97: Sampling definition for optimization using analytical sensitivities*

*Typical "Gradient" file (e.g. for f):*

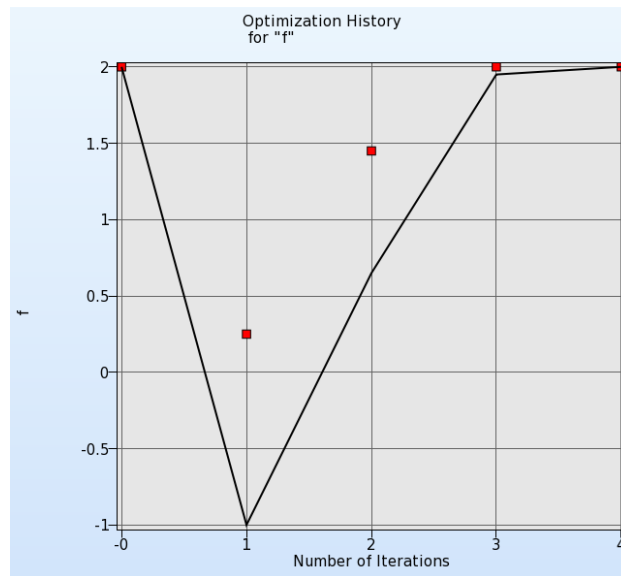
```
1.8000000000 -3.2000000000
```

### 17.8.3. 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 17-98: Optimization history for variables**



**Figure 17-99: Optimization history for objective**

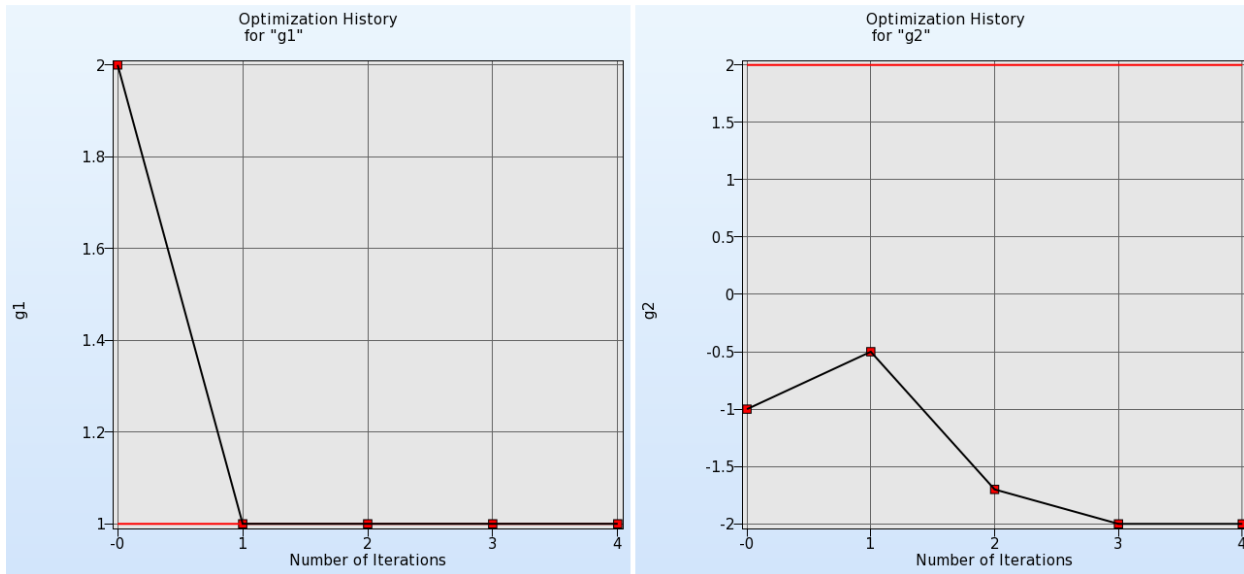


Figure 17-100: Optimization history for constraints

## 17.9. 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](http://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 12<sup>th</sup> 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.

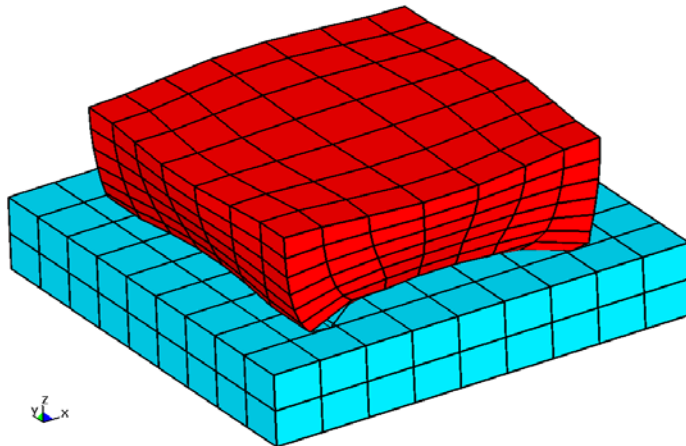
# 18. Examples – Parameter Identification

## 18.1. 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.

### 18.1.1. Problem statement



*Figure 18-1: 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 18-1). 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.

### 18.1.2. Ordinate-based Curve Matching

The LS-OPT main GUI window is displayed in Figure 18-2.

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 18-3. The experimental curves used as target curves are read into LS-OPT as File Histories, Figure 18-4. 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 18-5 (Section 9.5.1):

$$\varepsilon = \frac{1}{P} \sum_{p=1}^P W_p \left( \frac{f_p(\mathbf{x}) - G_p}{s_p} \right)^2 = \frac{1}{P} \sum_{p=1}^P W_p \left( \frac{e_p(\mathbf{x})}{s_p} \right)^2$$

where  $P=10$ ,  $s_p = 1$  and  $W_p = 1$ ,  $p = 1, \dots, 10$ .

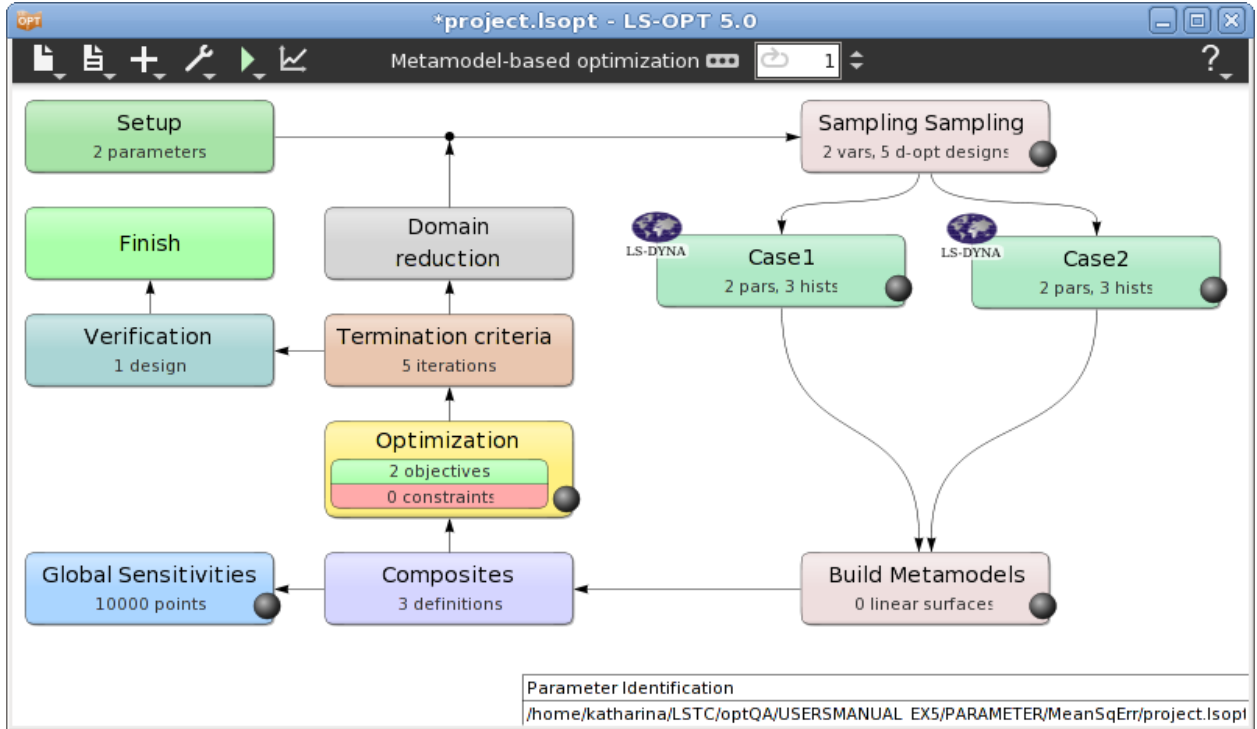
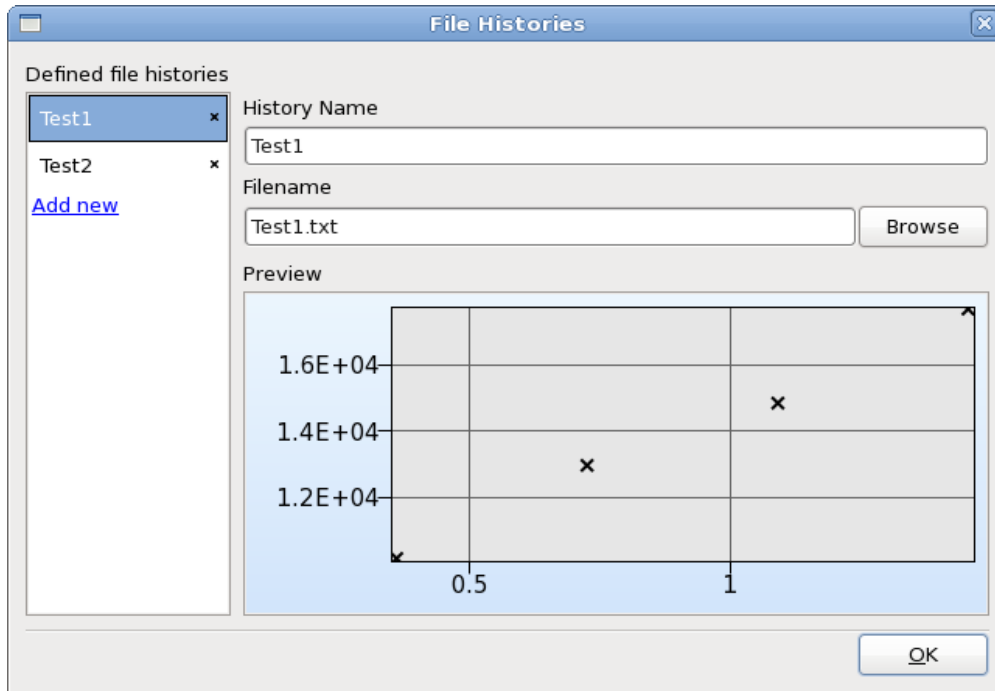


Figure 18-2: LS-OPT main GUI window

The screenshot shows the 'Edit history' dialog box. It contains the following fields and options:

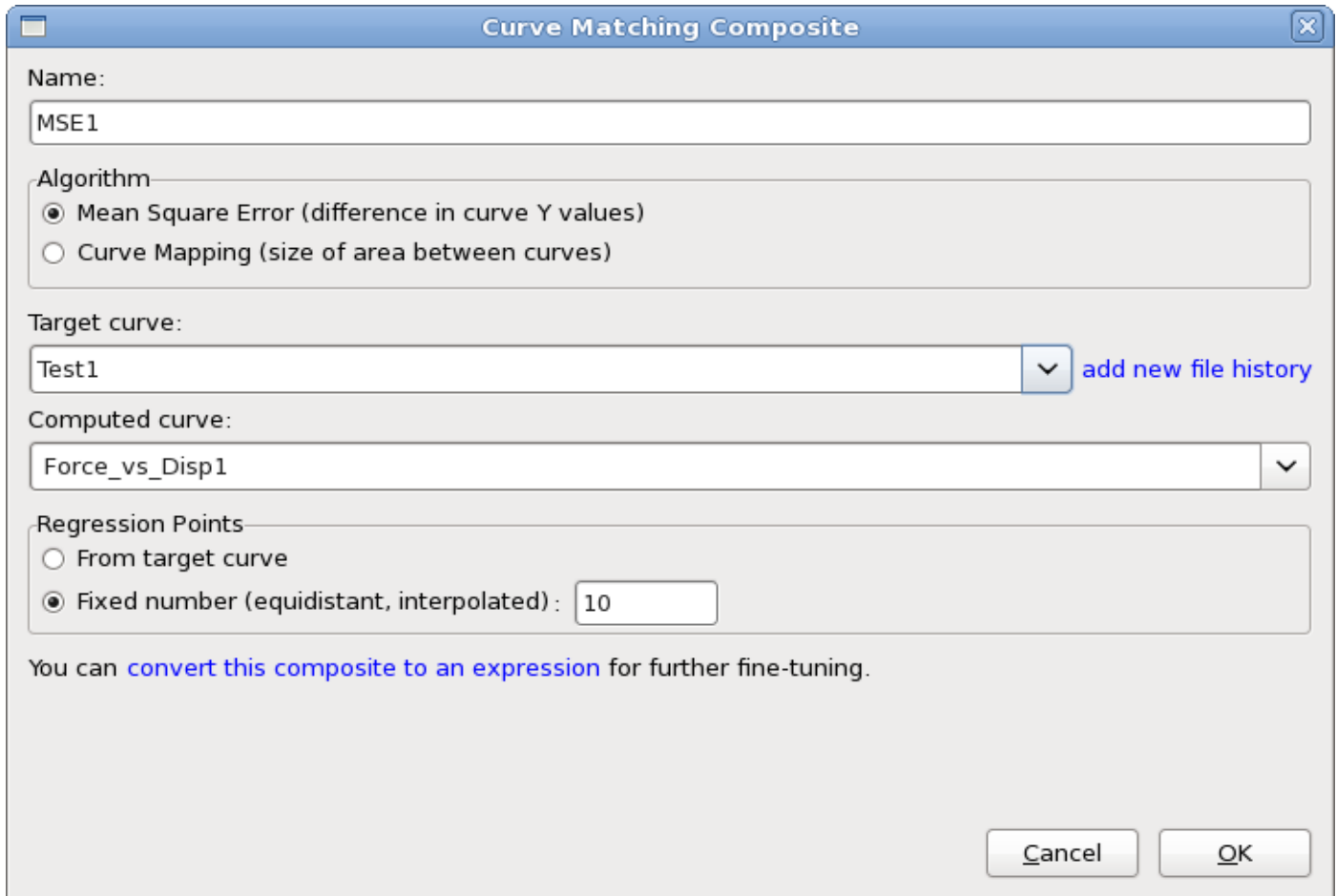
- Name:** Force\_vs\_Displ1
- Subcase:** (empty dropdown menu)
- Description:** A crossplot will create the history  $F(z)$ , given  $F(t)$  and  $z(t)$ . General expressions are allowed.
- z(t):** -Disp1
- F(t):** Force1
- Number of points (blank for default):** (empty text input field)
- Buttons:** Cancel and OK

Figure 18-3: Crossplot definition of force vs. displacement



*Figure 18-4: File History definition. This dialog is accessible from the Histories tab of the Stage dialog or the Curve Matching composite dialog.*





*Figure 18-5: Definition of Mean Square Error composite*

### 18.1.3. 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:

$$\text{Minimize } \sum_{j=1}^7 [f_j(\mathbf{x}) - F_j],$$

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 18-6. The definition of the targeted composite is displayed in Figure 18-7.

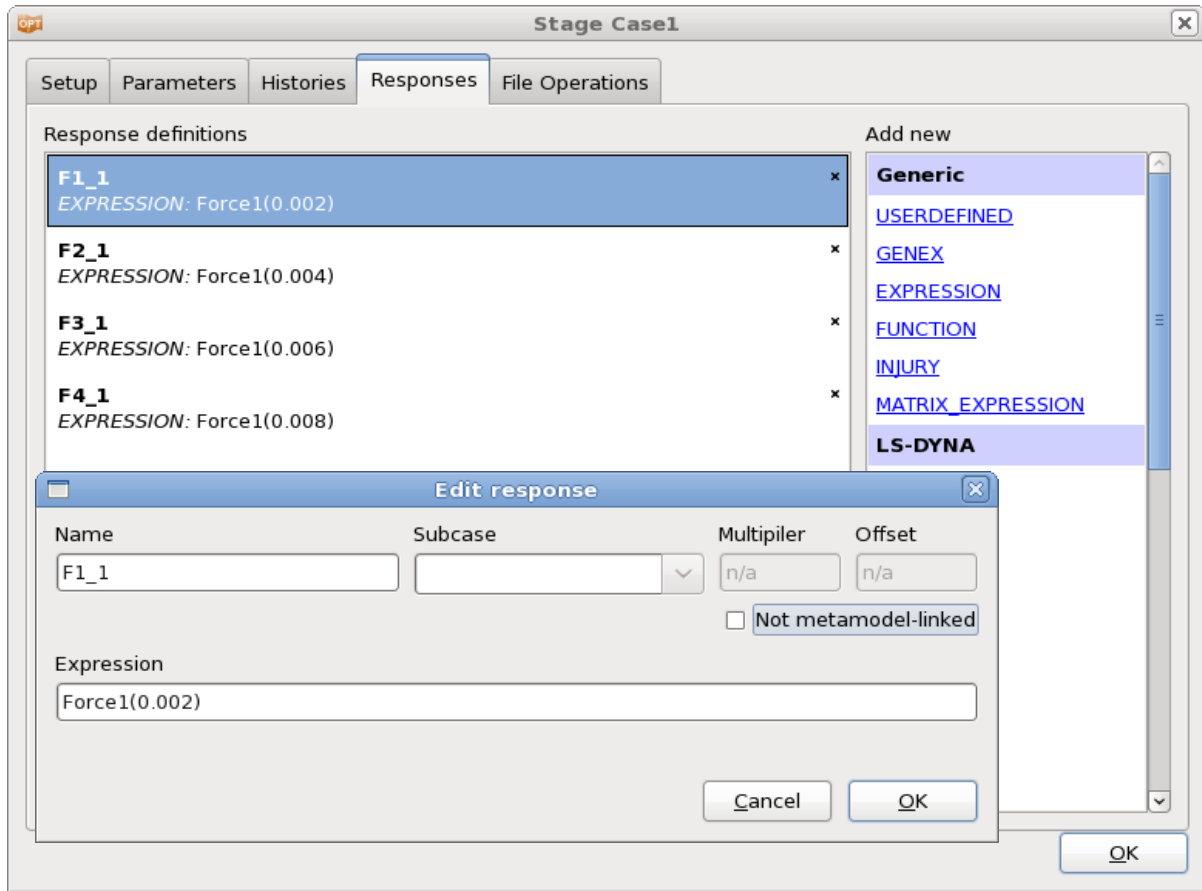
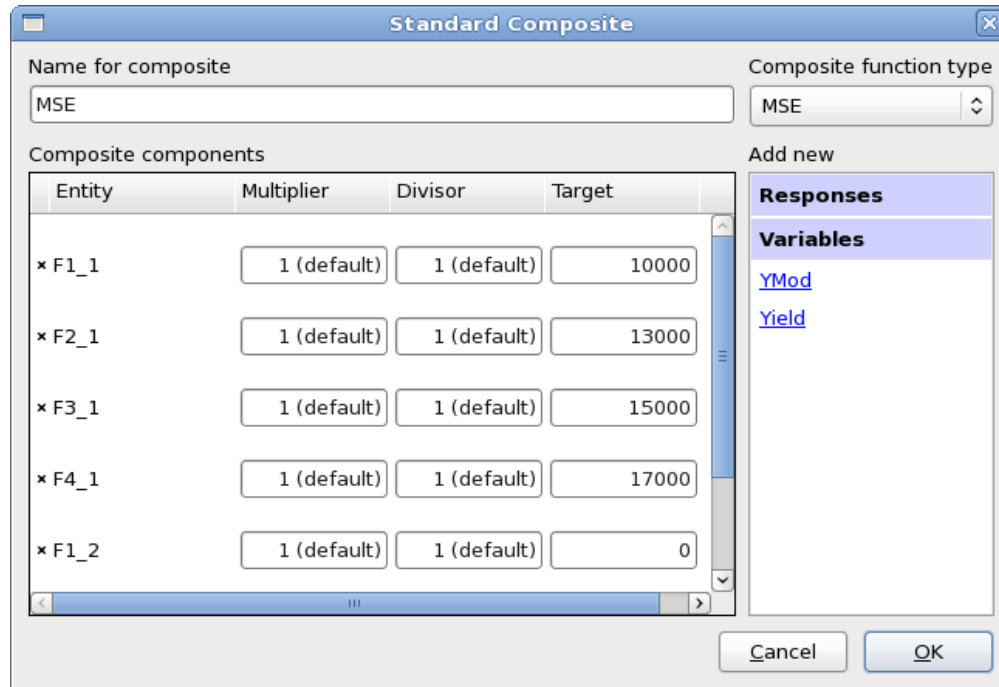


Figure 18-6: Evaluation of simulation curves at target  $t$  values



*Figure 18-7: Definition of equality constraints using the Standard Composite type MSE.*

#### 18.1.4. Results

The results for both methods are compared below.

#### 18.1.5. Mean Squared Error (MSE) formulation

Figure 18-8 visualizes the optimal parameter values (red line) and the respective subregion (blue lines) over the iterations. Figure 18-9 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 18-10.

Figure 18-11 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 18-12 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.

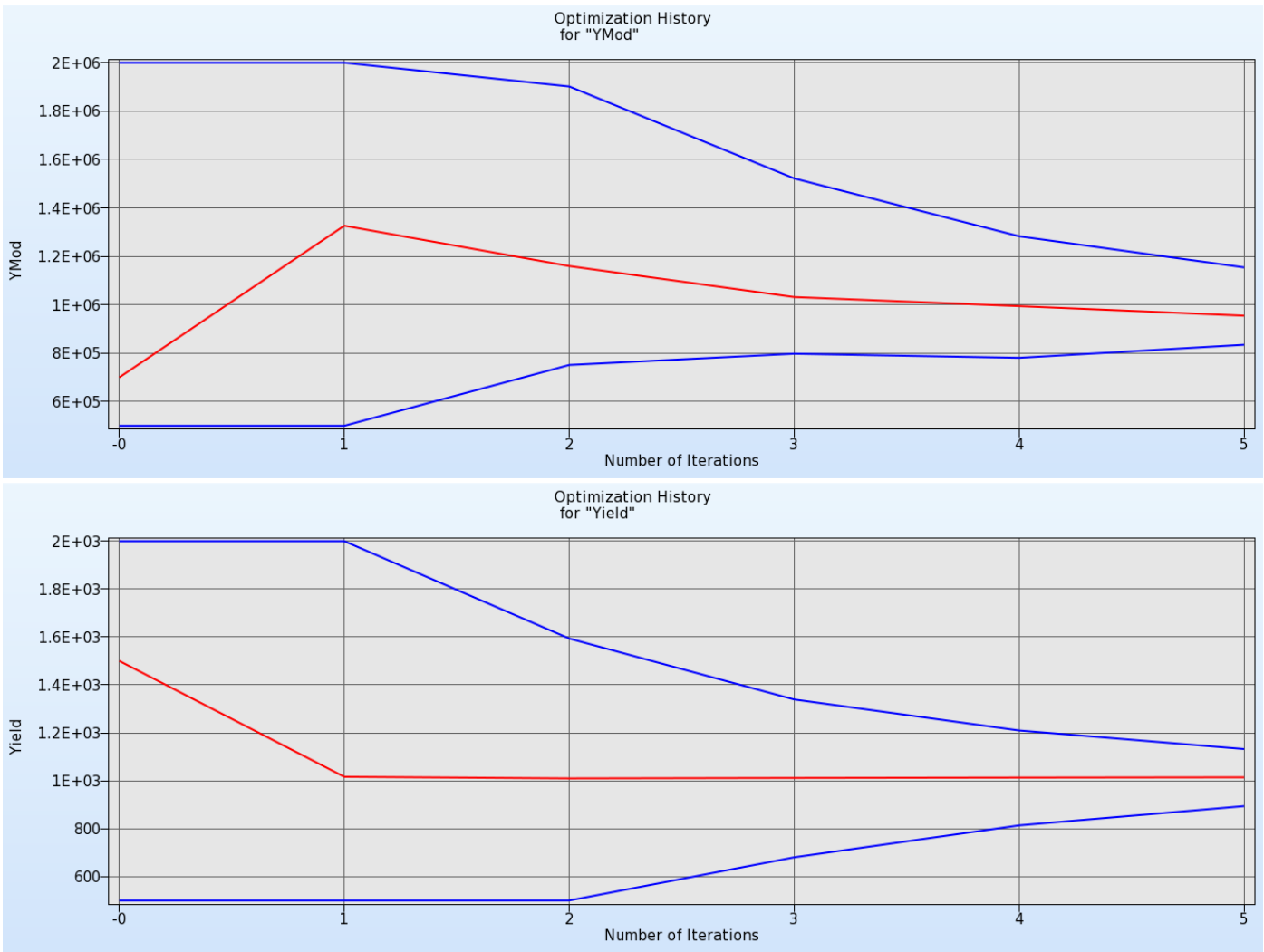


Figure 18-8: Optimization History for YMod and Yield

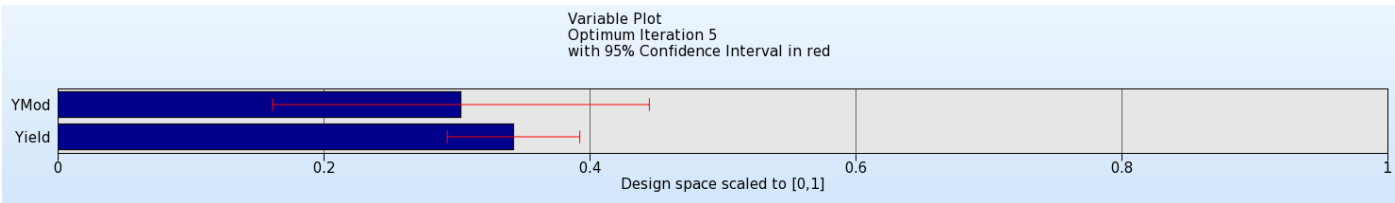


Figure 18-9: Parameter values of optimal point in normalized design space with 95% confidence interval.

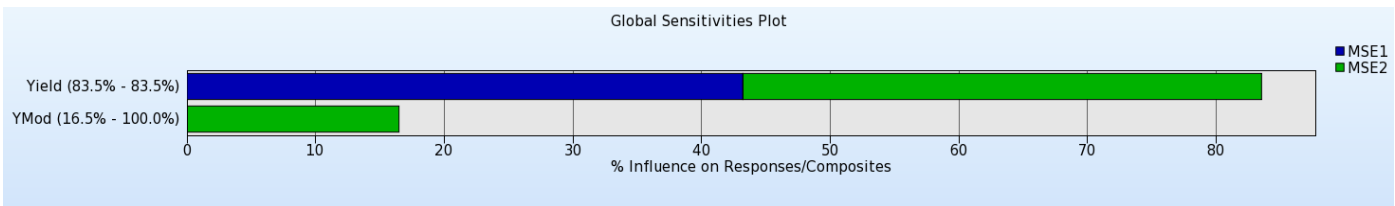
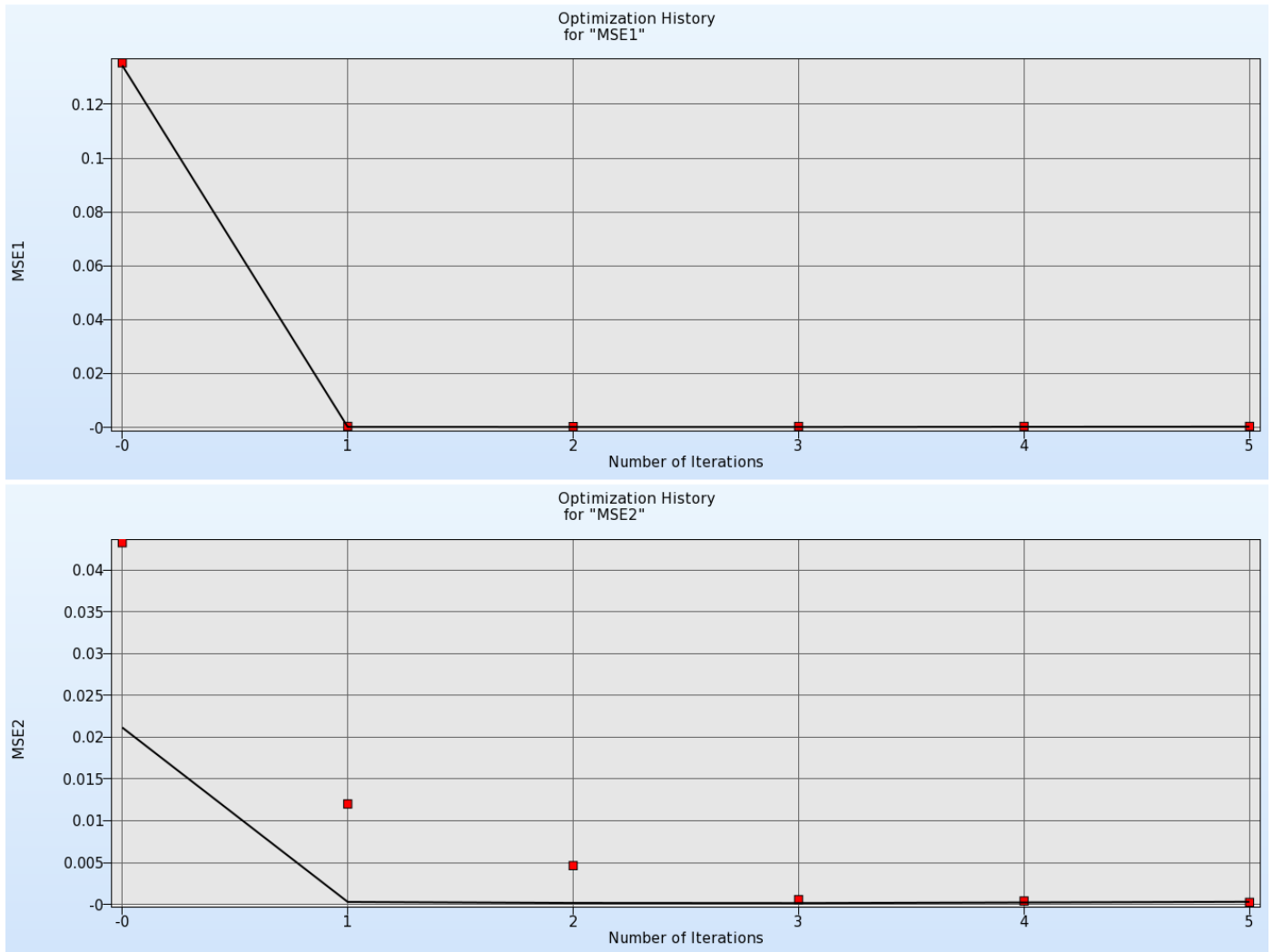


Figure 18-10: Global Sensitivities for MSE1 and MSE2



**Figure 18-11:** Optimization history of MSE 1 and MSE2. Both objectives decrease, and the accuracy of the metamodel improves over the iterations.



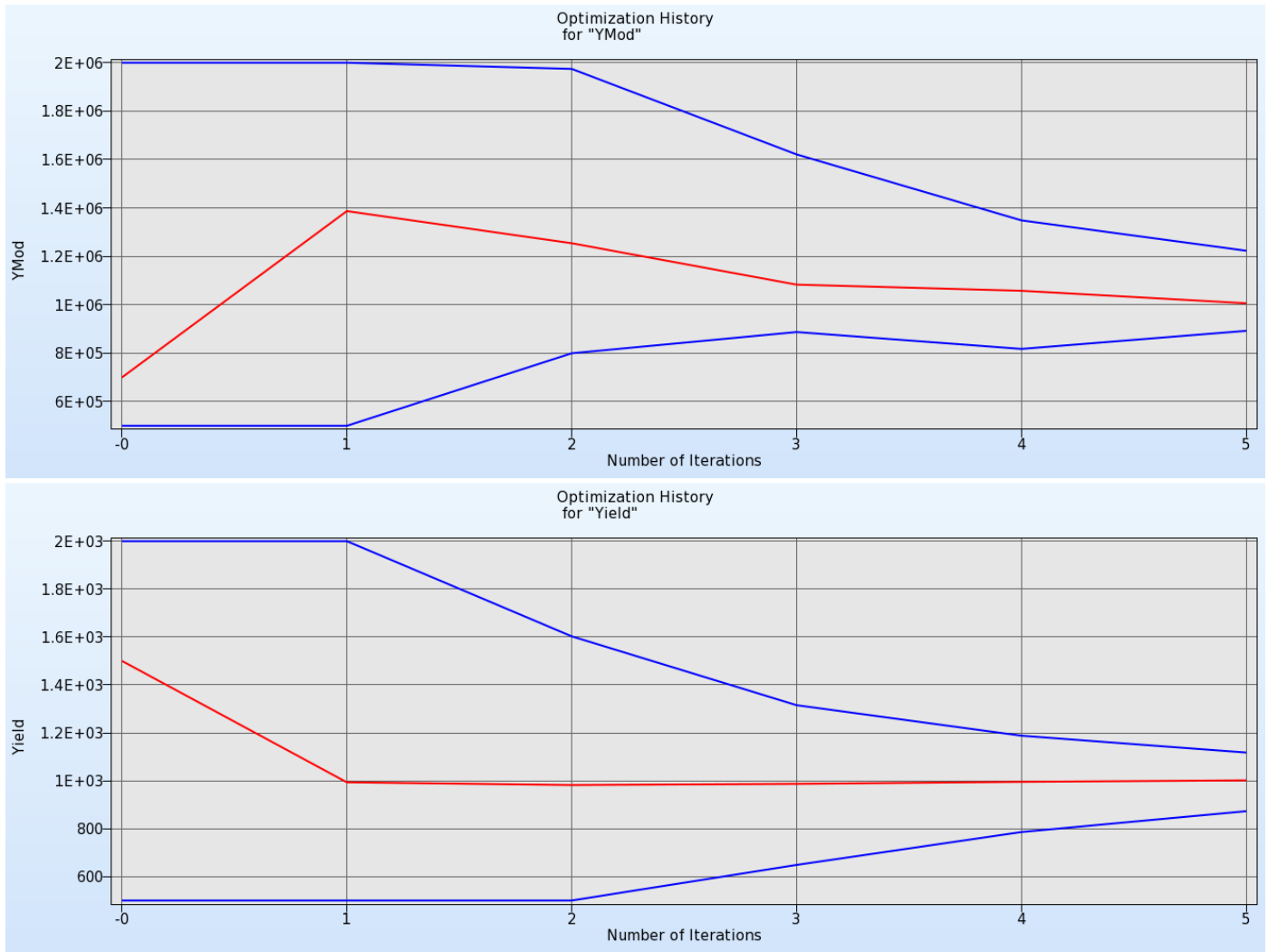
**Figure 18-12:** Comparison of optimal force-displacement curves and test data. The simulation curves are colored by iteration

### 18.1.6. Targeted composite formulation

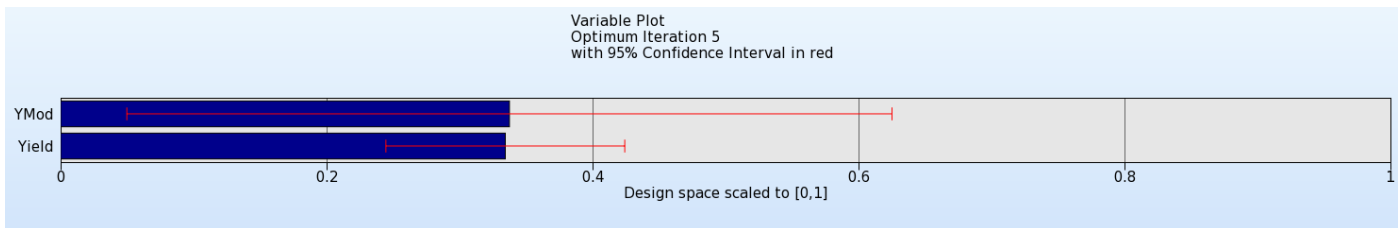
Figure 18-13 visualizes the optimal parameter values (red line) and the respective subregion (blue lines) over the iterations. Figure 18-14 displays the final optimal parameter values with respect to a normalized design space with 95% confidence interval. The larger confidence interval as well as the slower convergence of YMod can be explained by the insignificance of that parameter on the objective function, Figure 18-15.

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 18-15).

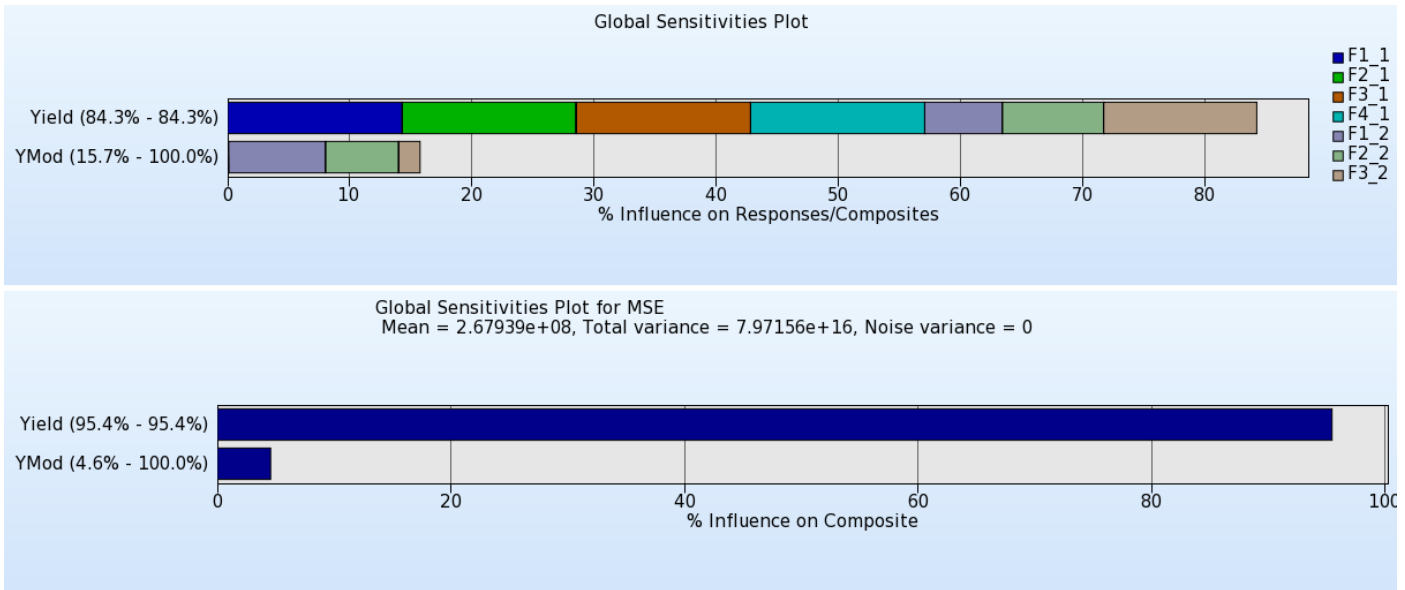
Figure 18-16 displays the computed (red square) and predicted (black line) objective values over the iterations. The objective decreases, the quality of the predictions improves and the objective value converges.



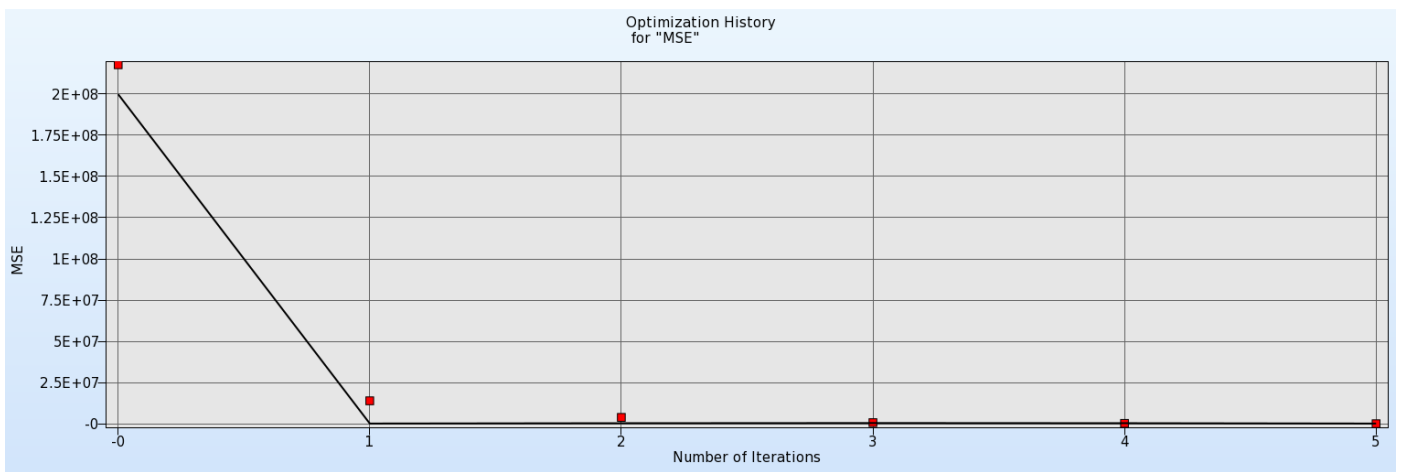
**Figure 18-13: Optimization History for YMod and Yield**



**Figure 18-14: Parameter values of optimal point in normalized design space with 95% confidence interval.**



**Figure 18-15: Global Sensitivities for all forces and MSE**



**Figure 18-16: Optimization history of MSE. The objectives decrease, and the accuracy of the metamodel improves over the iterations.**

## 18.2. System identification with hysteretic curves

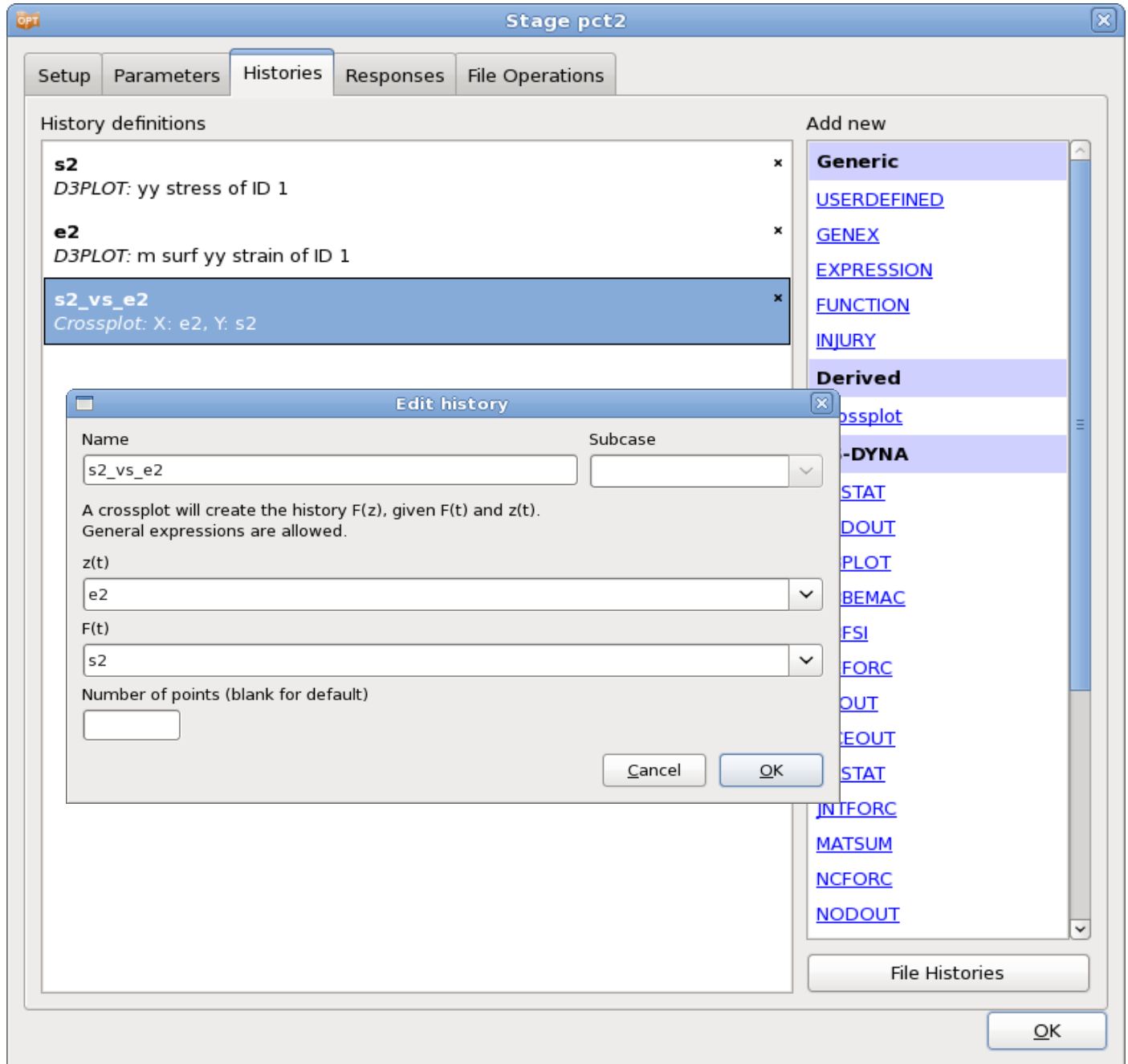
### 18.2.1. Problem statement

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.

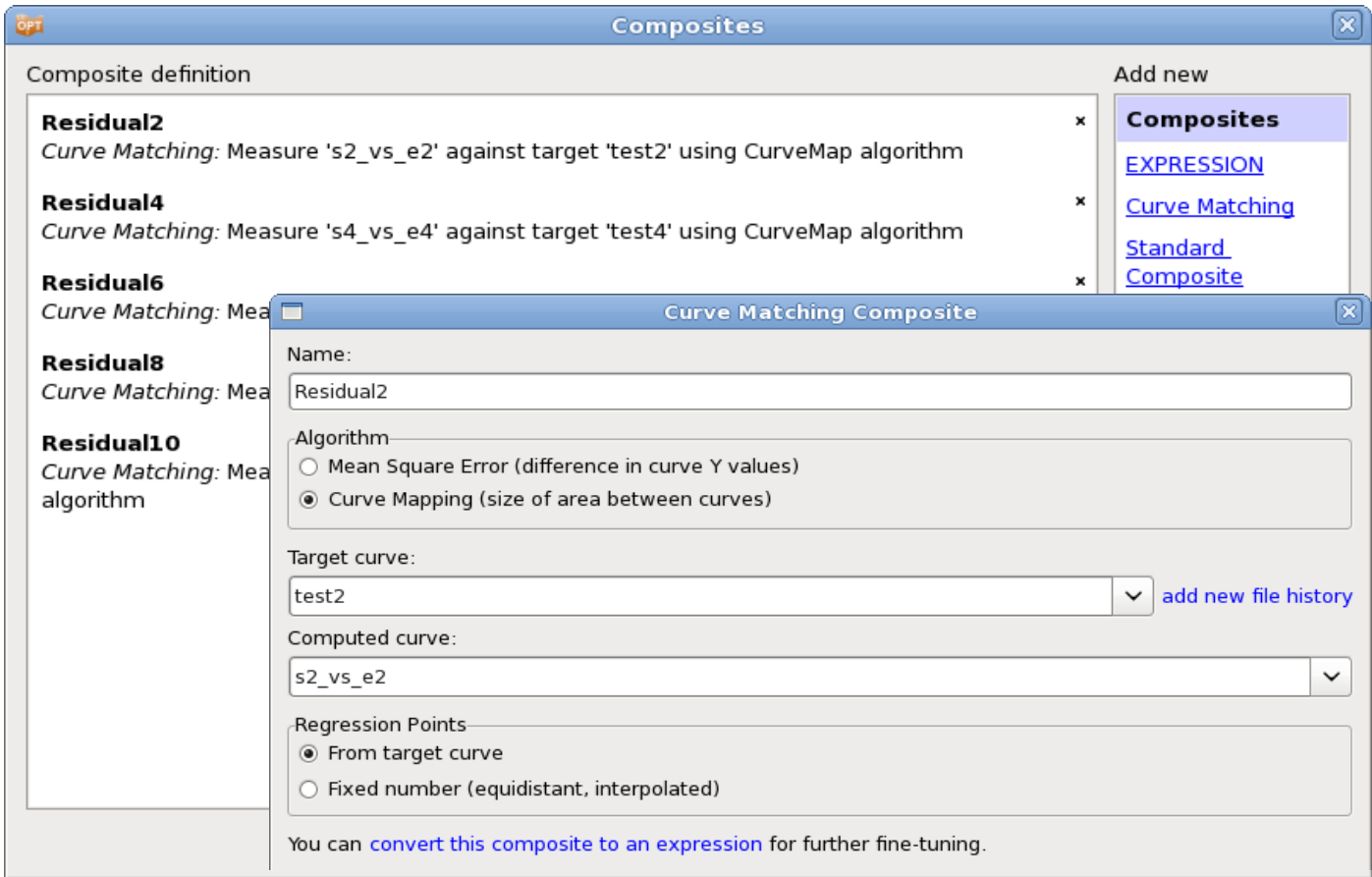


The following example consists of five load cases, each representing a different cyclic loading range as illustrated in the stress-strain diagram in the figure below. The material is defined by 9 parameters.

### 18.2.2. Solution using Curve Mapping



*Figure 18-17: History definitions. Extract stress and strain using the LS-DYNA d3plot interface. Use the Crossplot interface to generate the stress vs. strain curve*



*Figure 18-18: Define a curve matching composite for each load case.*

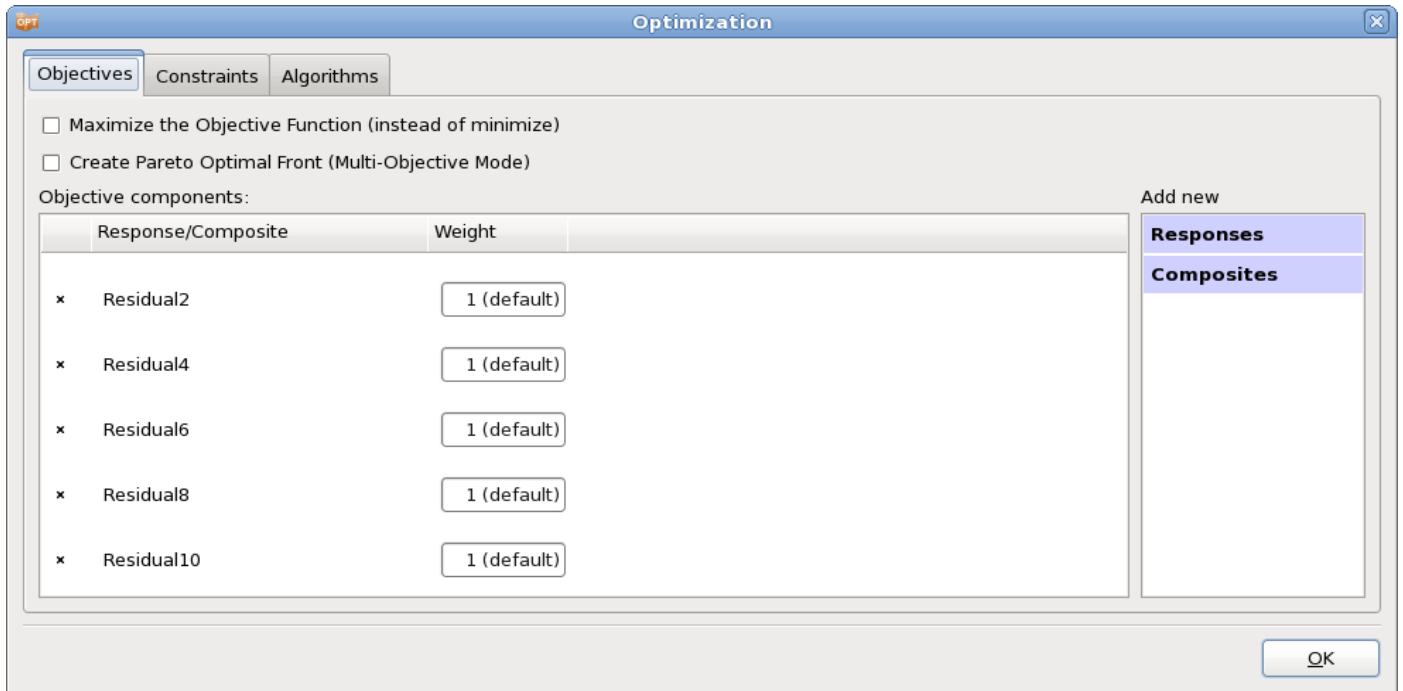


Figure 18-19: Definition of the five objective components.

### 18.2.3. Results

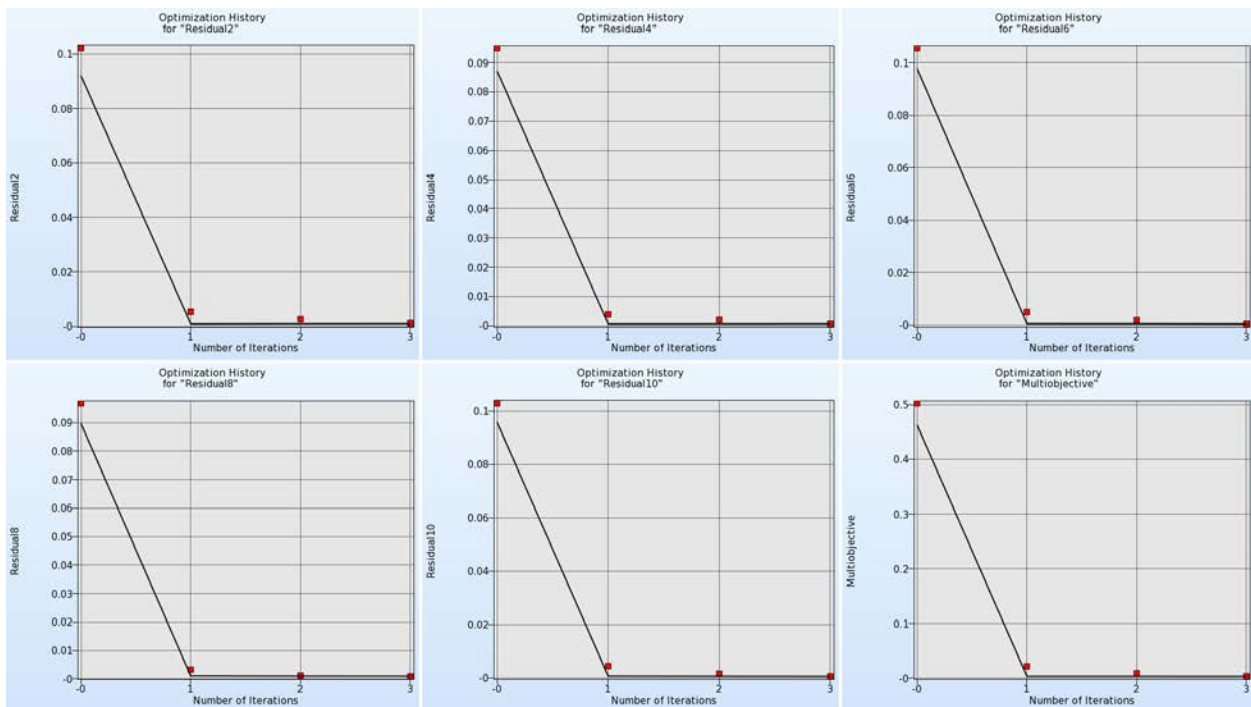


Figure 18-20: Optimization History of objective components for each of the five load cases as well as the multi-objective

Figure 18-20 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 18-21 displays the Global Sensitivities plot for the whole problem. The variable *CB* is by far the most sensitive.

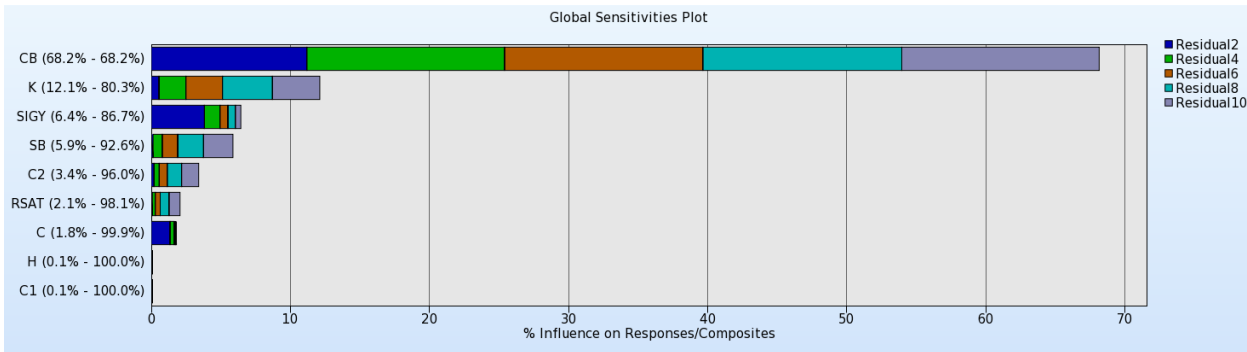


Figure 18-21: Global Sensitivities plot of all objective components

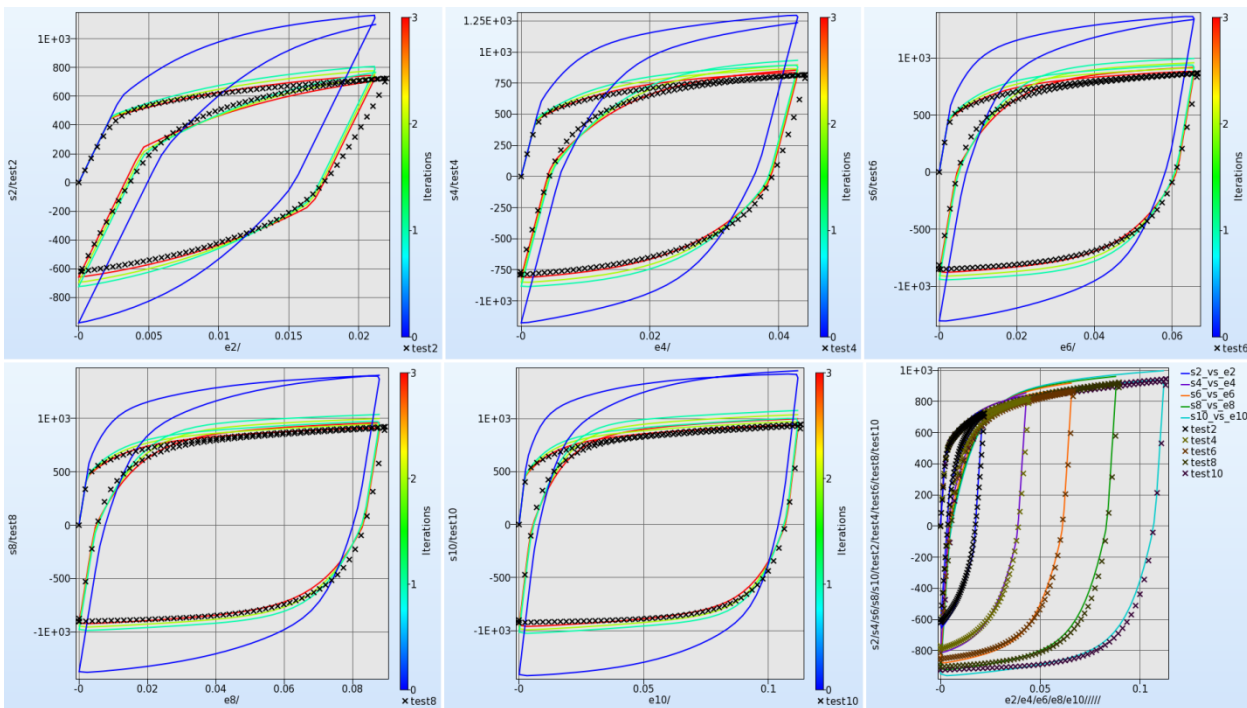


Figure 18-22: Comparison of optimal simulation curves of all iterations and target curves for all load cases; simulation curves are colored by iteration (e.g. baseline in blue). The black crosses represent the target values. The plot bottom right shows the comparison of the final optimal simulation curves and target curves for all load cases.

Figure 18-22 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.

### 18.3. 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](http://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 12<sup>th</sup> 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.

# 19. Examples – Probabilistic Analysis

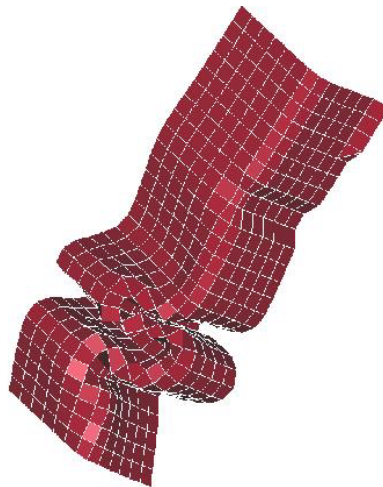
## 19.1. Probabilistic Analysis

### 19.1.1. Overview

This example has the following features:

- Probabilistic analysis
- Monte Carlo analysis
- Monte Carlo analysis using a metamodel
- Bifurcations analysis

### 19.1.2. Problem description



*Figure 19-1: 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 19-1: 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 19-2: 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 19-3: 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.

### 19.1.3. Direct Monte Carlo evaluation

The probabilistic variation is described by specifying statistical distributions, Figure 19-3, and assigning the statistical distributions to noise variables, Figure 19-2. 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.

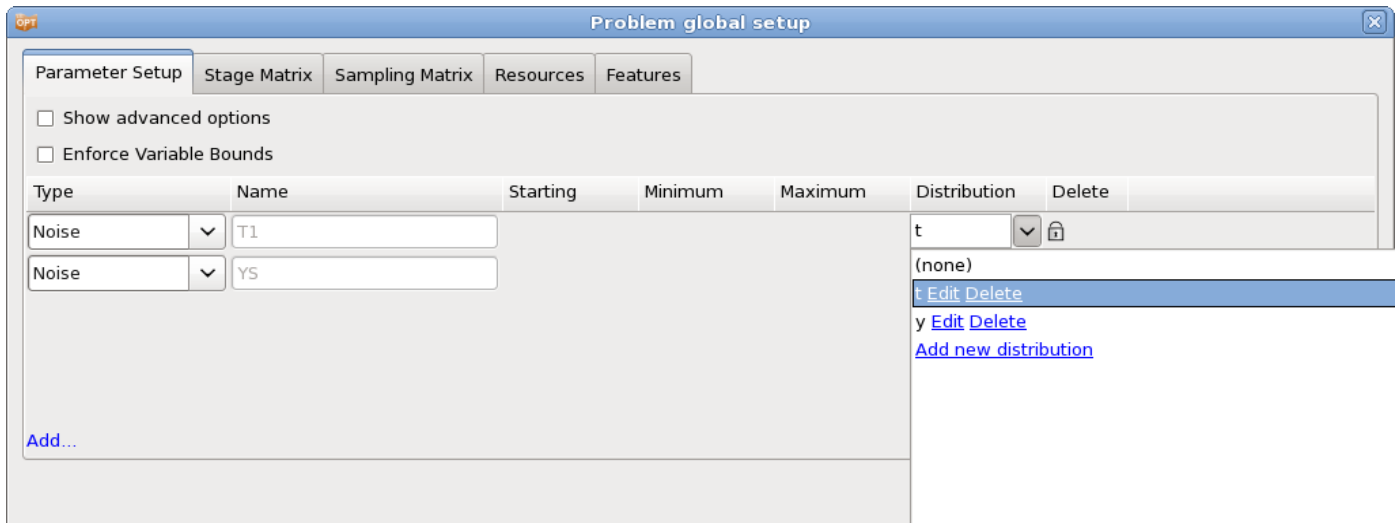


Figure 19-2: Assigning statistical distributions to noise variables

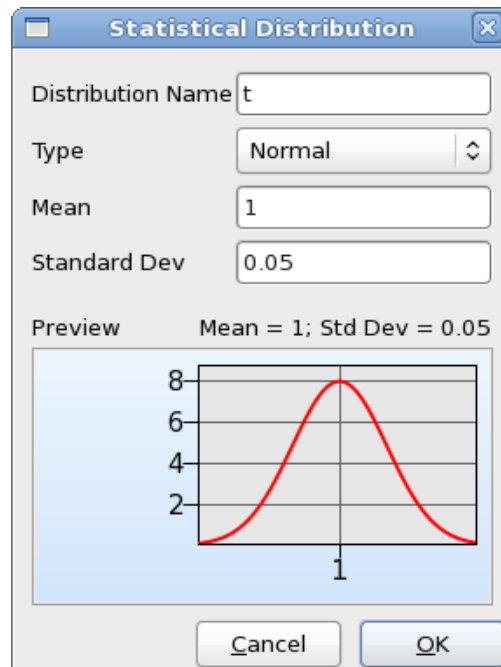


Figure 19-3: Definition of statistical distributions. This dialog is accessible from the Setup dialog's Parameter Setup tab.



### 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 19-4. 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 19-5. The bounds can be modified interactively in the viewer.

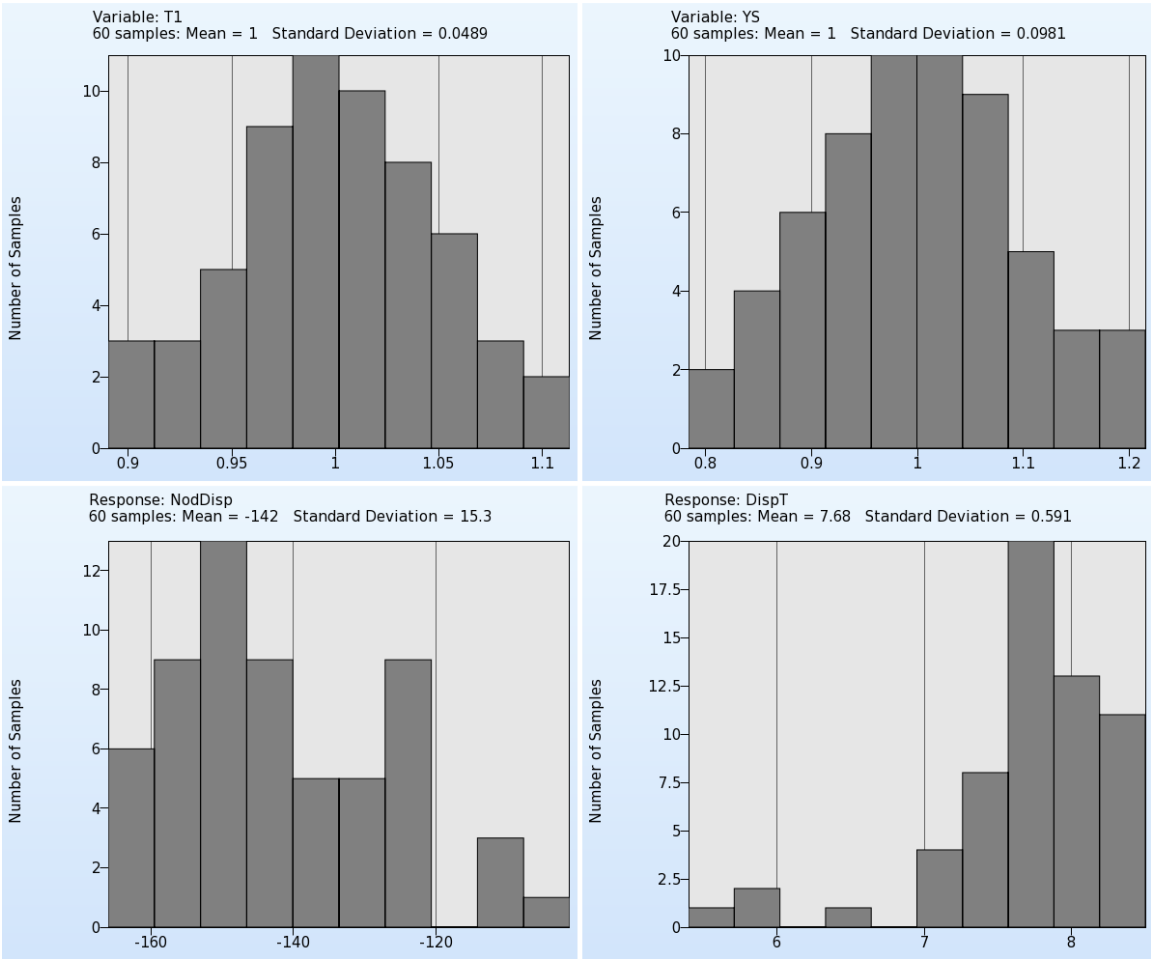


Figure 19-4: Histogram plots of variables and responses. Mean values and standard deviations are displayed in the titles.

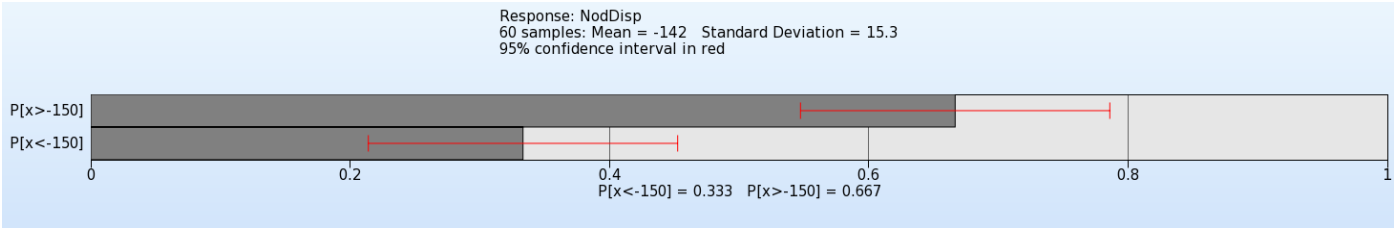
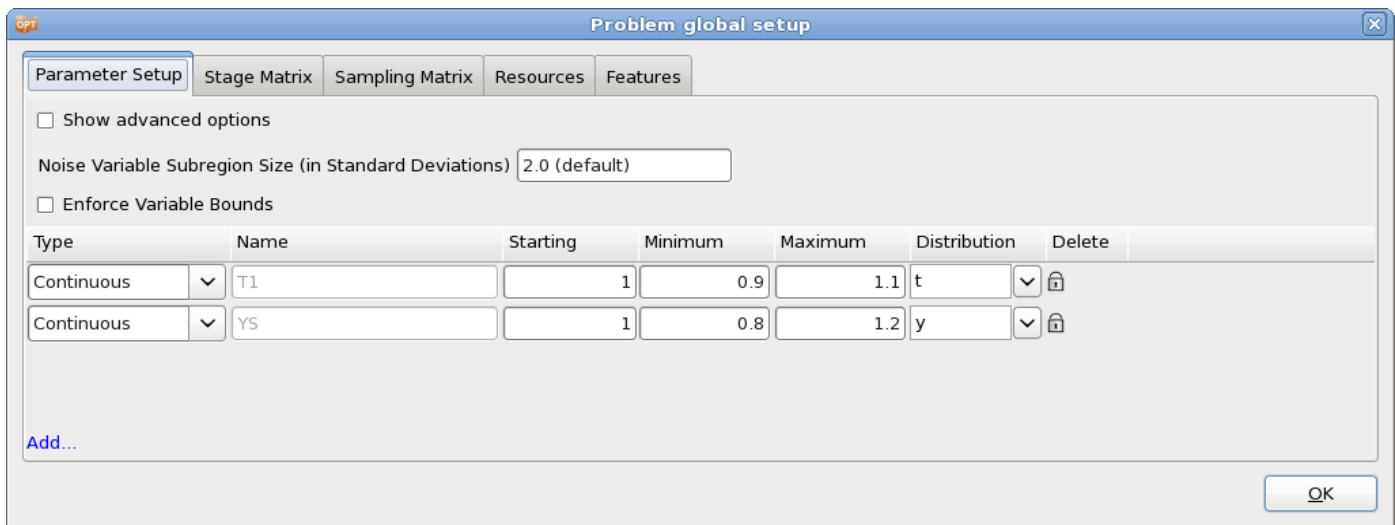


Figure 19-5: Probability of NodDisp < -150 with 95% confidence intervals

### 19.1.4. 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 19-6. 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.

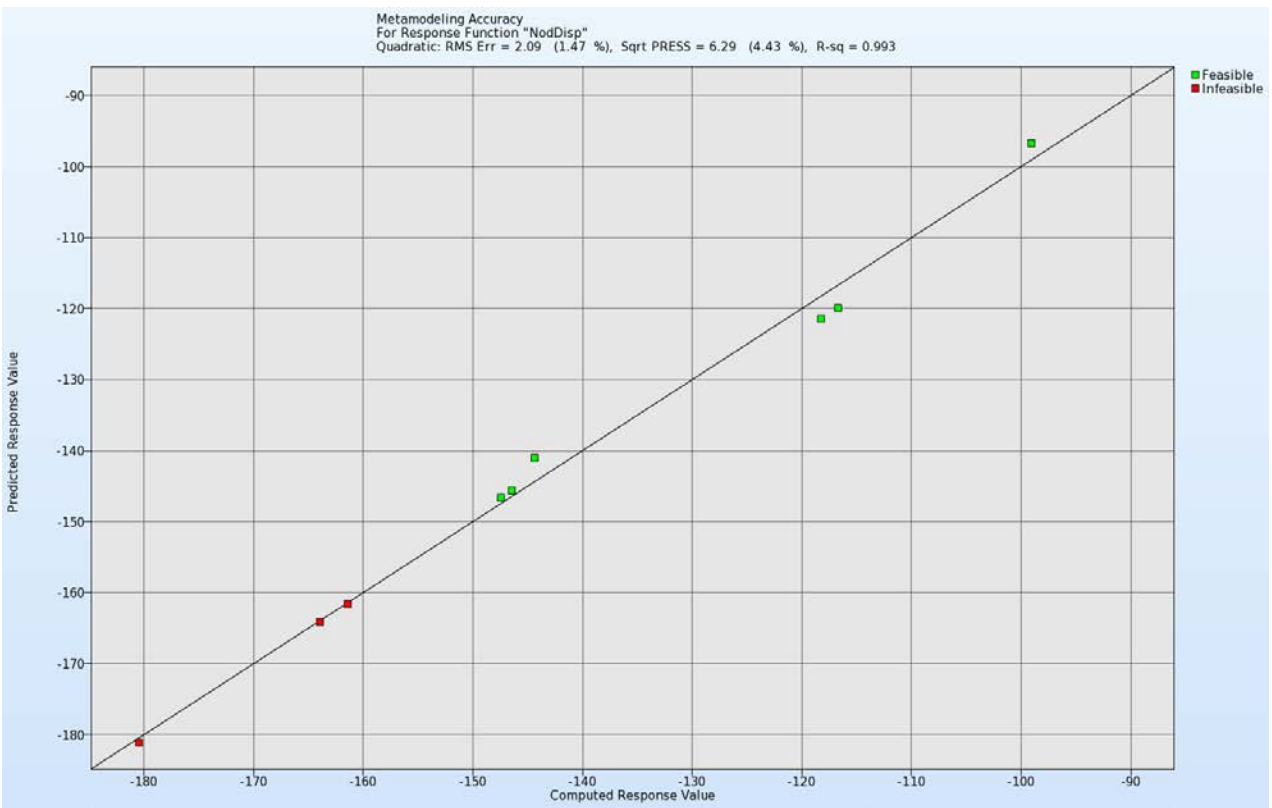


*Figure 19-6: Assigning statistical distributions to control variables*

## 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 19-7. The error measures RMS, SPRESS and  $R^2$  are displayed in the title.

The probabilistic evaluation results can be visualized using the **Statistical Tools** plot as described in Section 19.1.3, but now, 10000 points evaluated on the metamodel are used to calculate the statistics, Figure 19-8 and Figure 19-9.



*Figure 19-7: Accuracy plot. Computed vs. predicted values; error measures are displayed in the title*

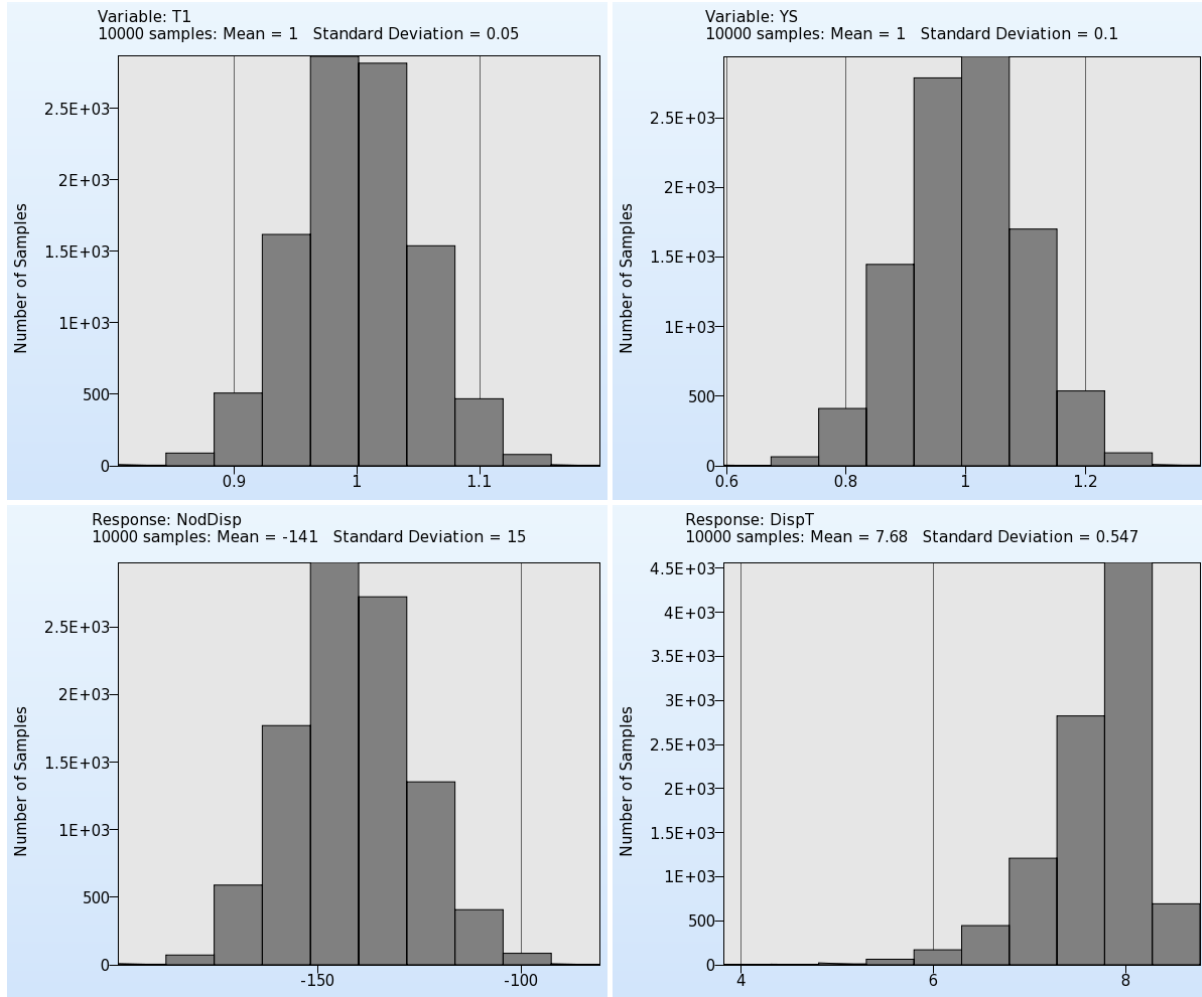


Figure 19-8: Histogram plots of variables and responses. Mean values and standard deviations are displayed in the titles. All values are evaluated on the metamodel using 10000 points.

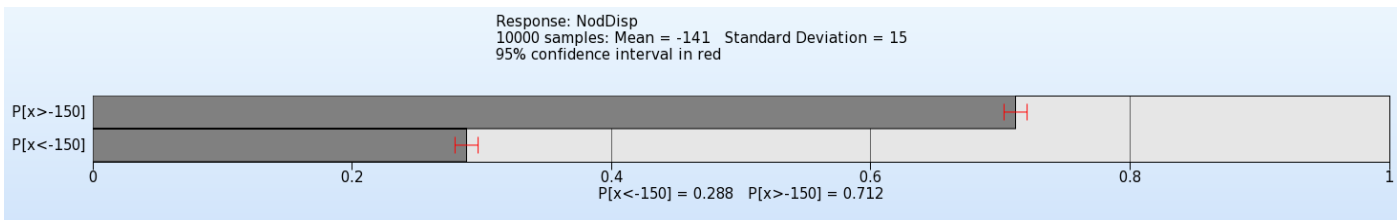


Figure 19-9: Probability of  $NodDisp < -150$  with 95% confidence intervals evaluated on the metamodel using 10000 points.

### 19.1.5. Bifurcation analysis

A bifurcation analysis of the tube is conducted using the methods described in more detail in Section 24.6, Section 15.8, and Section 19.2. The resulting buckling modes found from the analysis are as shown in Figure 19-10. An extra half wave is formed for one of the designs.

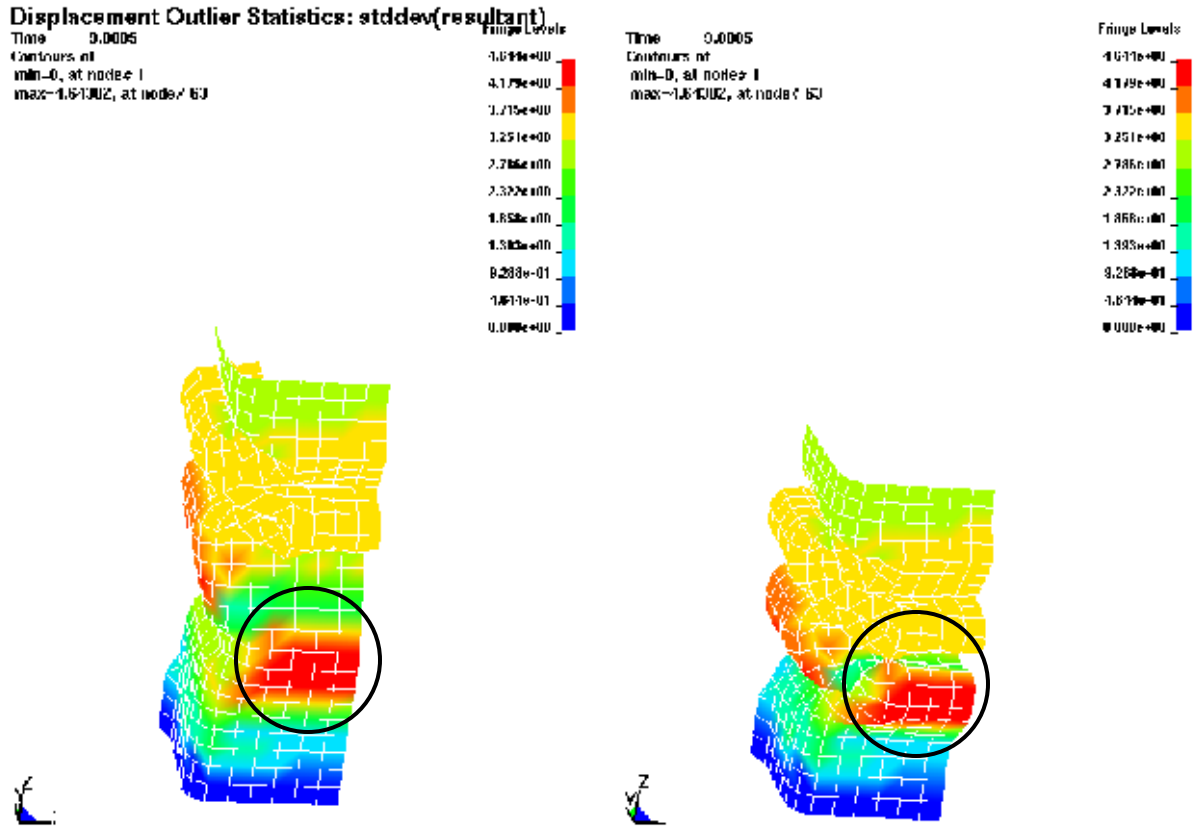


Figure 19-10 Tube Buckling

## 19.2. Bifurcation/Outlier Analysis

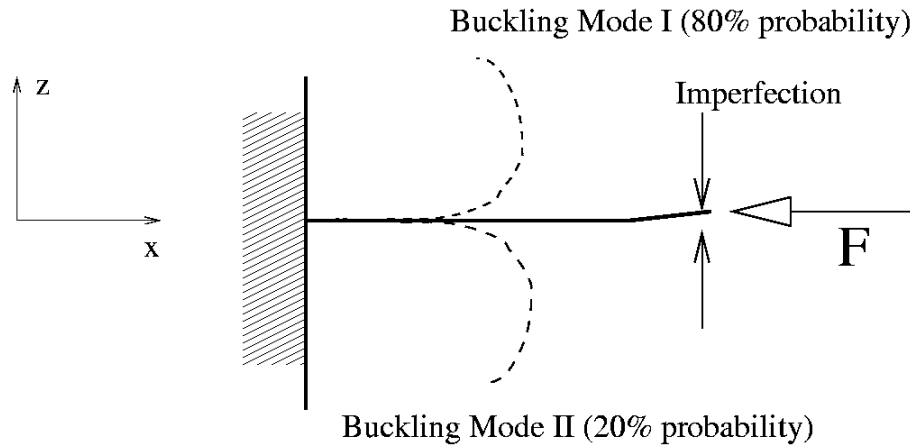
### 19.2.1. Overview

This example has the following features:

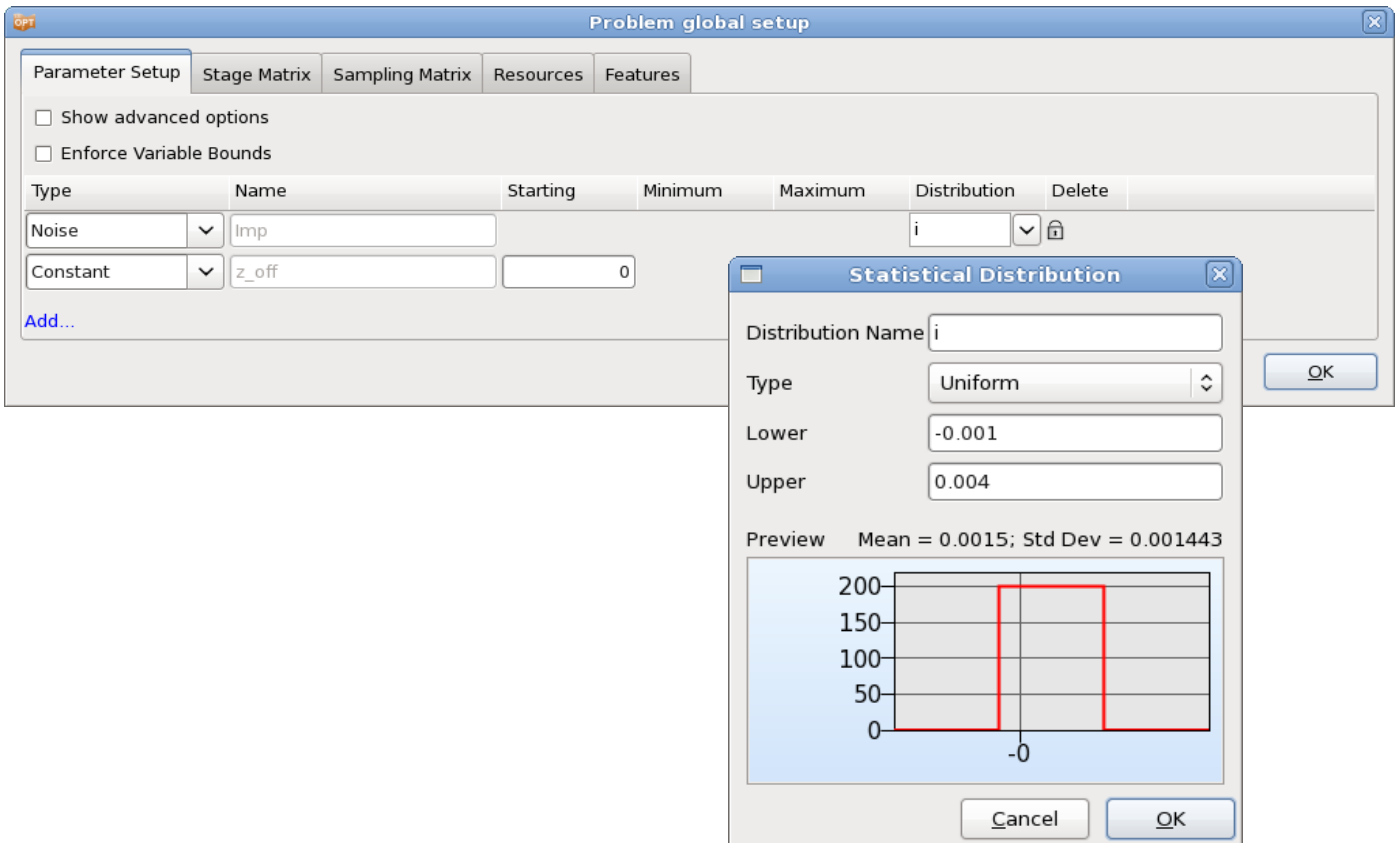
- Monte Carlo analysis
- Identification of different buckling modes in the structure

### 19.2.2. Problem description

The plate as shown in Figure 19-11 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.

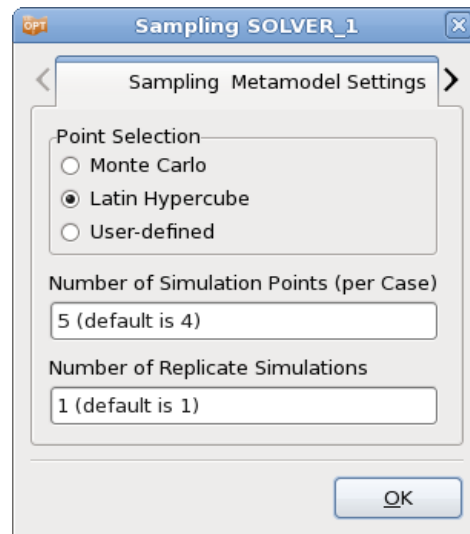


**Figure 19-11: Plate Buckling Example**



**Figure 19-12: Definition of noise variable and distribution**

A Latin hypercube experimental design is used for the Monte Carlo analysis, Figure 19-13. 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 19-14. The next section will demonstrate how to find out which run contains the different buckling mode.



*Figure 19-13: Definition of Latin Hypercube sampling with 5 points.*

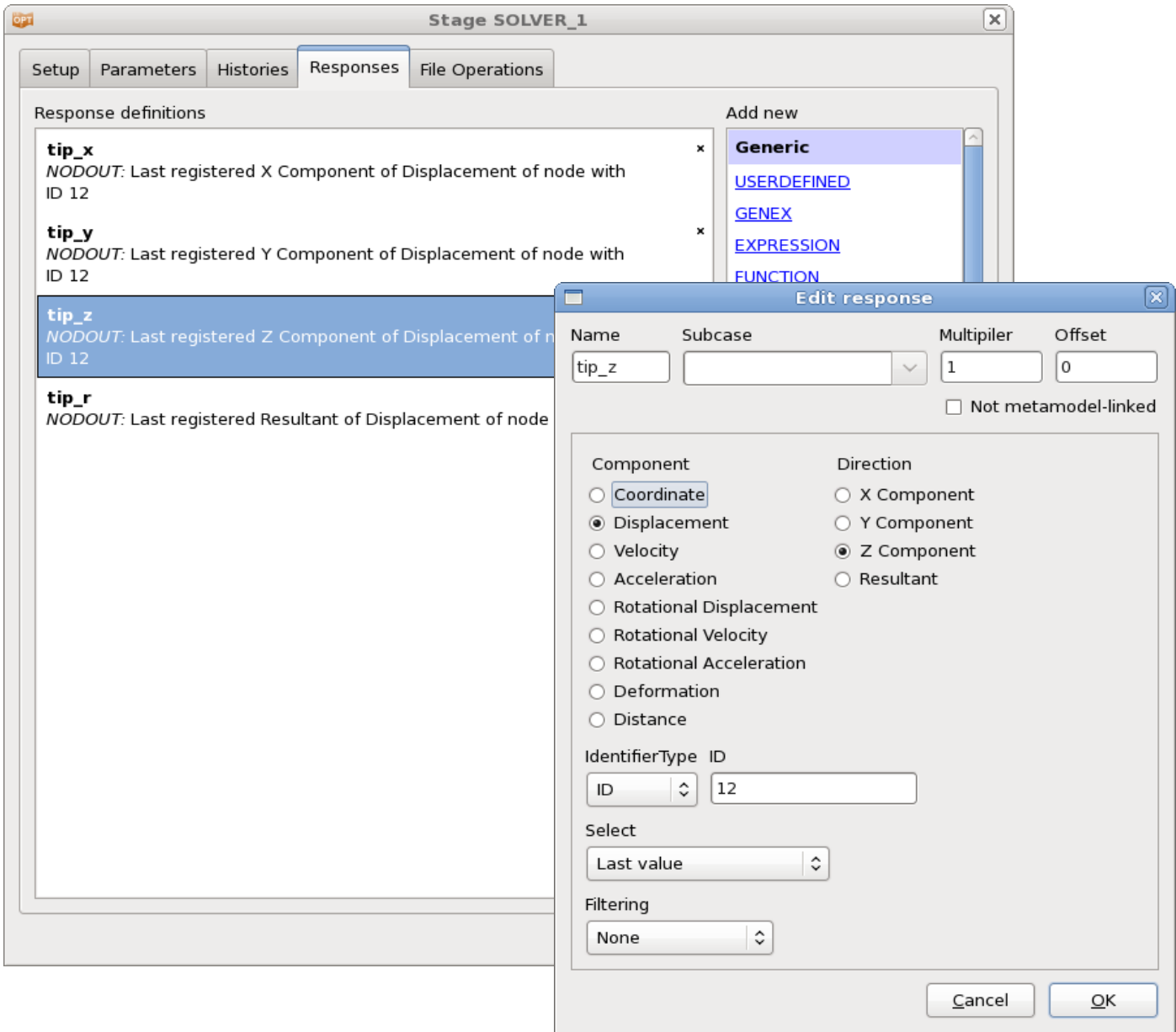
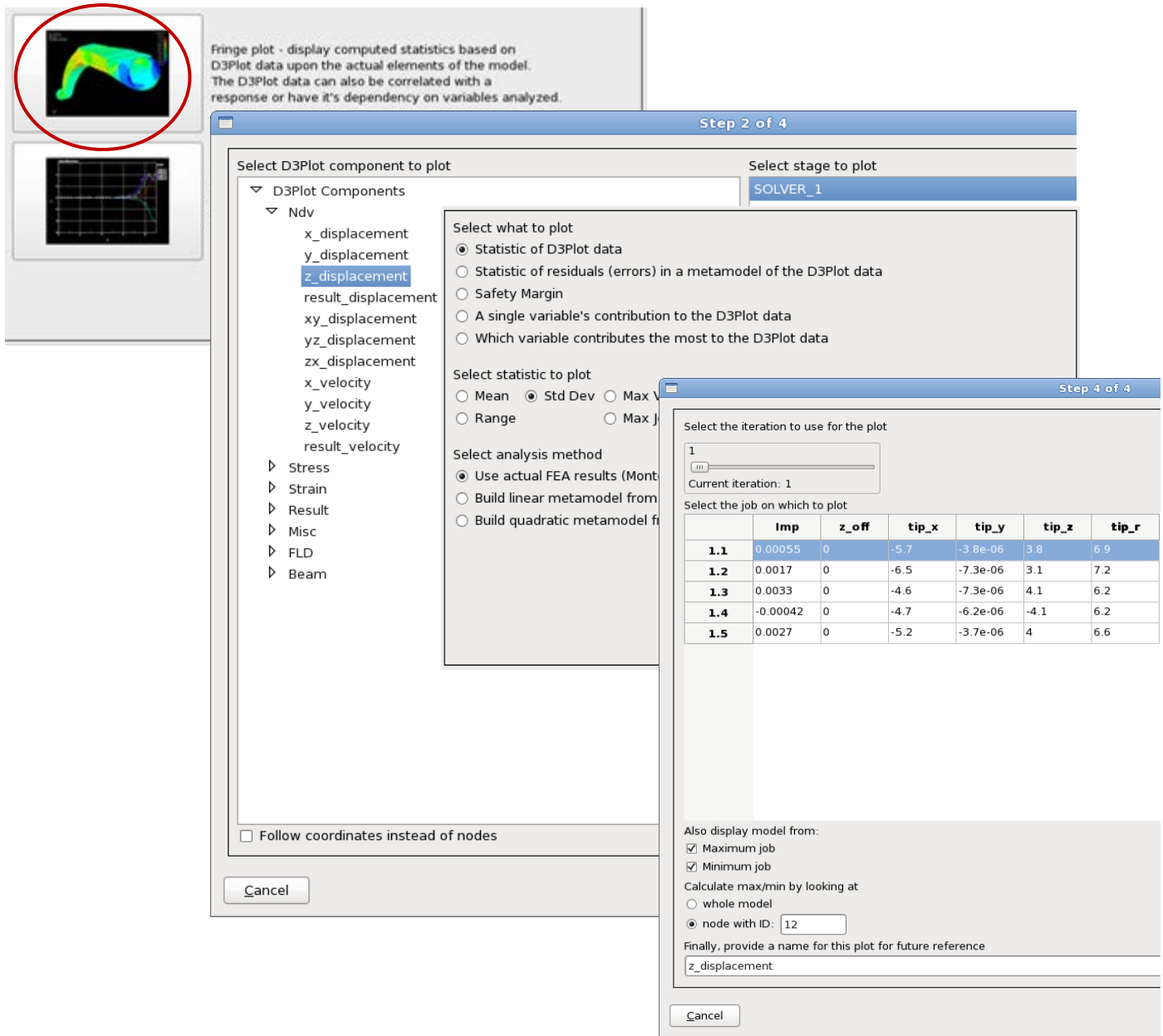


Figure 19-14: Response Definition for bifurcation identification

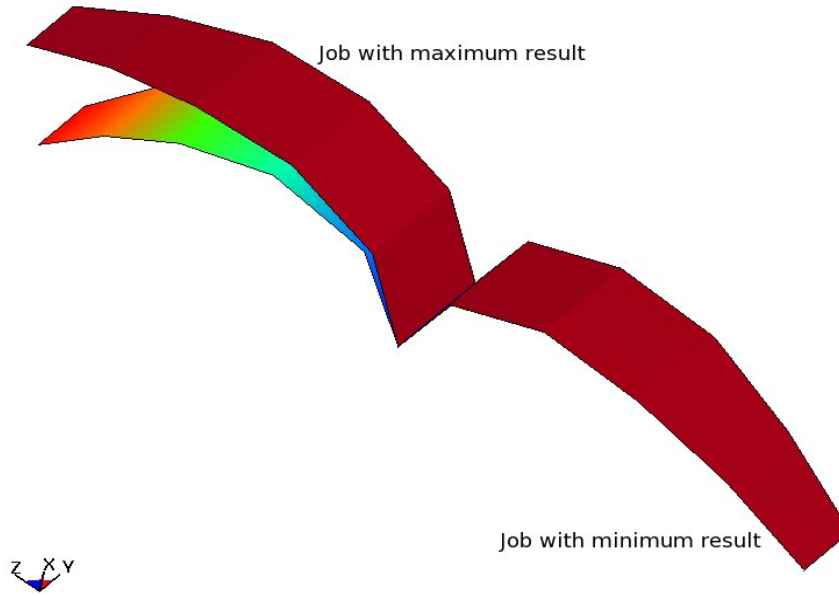
#### 19.2.4. Automatic identification of buckling modes

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 Figure 19-15 with the bifurcation as displayed using LS-PREPOST as shown in Figure 19-16. Some background on bifurcation identification can be found in Section 15.8. A more user-intensive procedure is described in the next section.





**Figure 19-15: 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.**

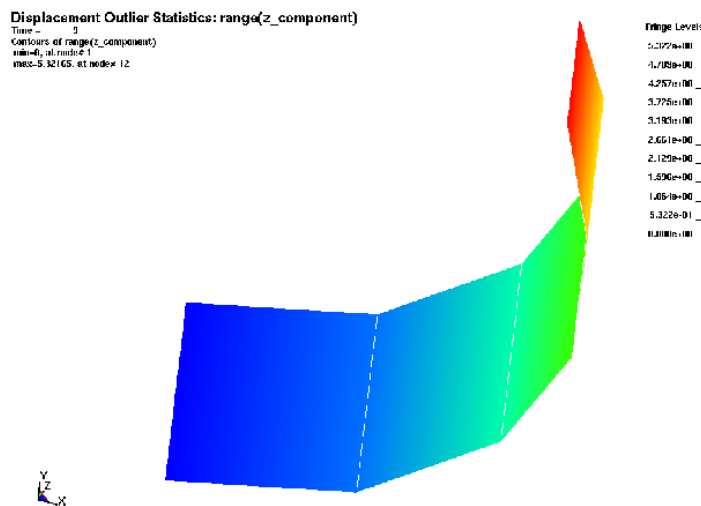


*Figure 19-16: LS-OPT identified and displayed this bifurcation automatically using the GUI setting shown in the previous figure.*

### 19.2.5. 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 19-17: 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.

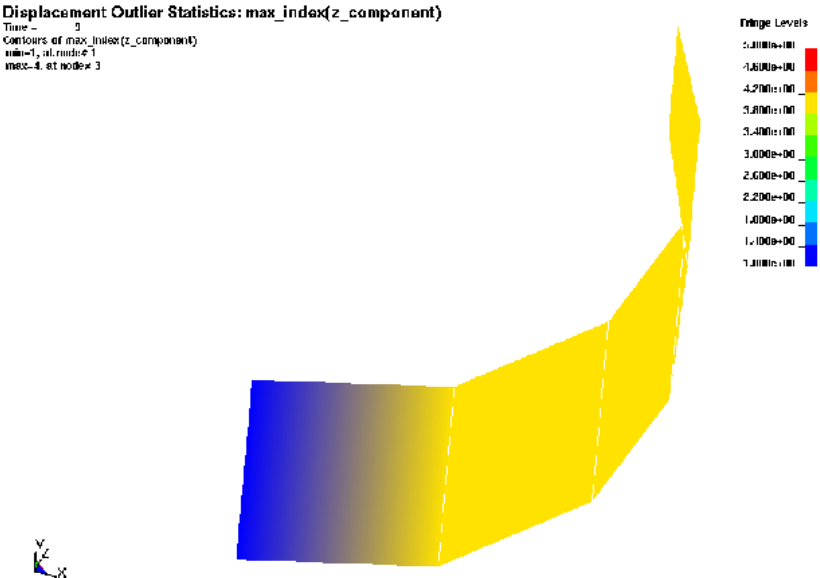


Figure 19-18: Index of run with maximum z-component displacement

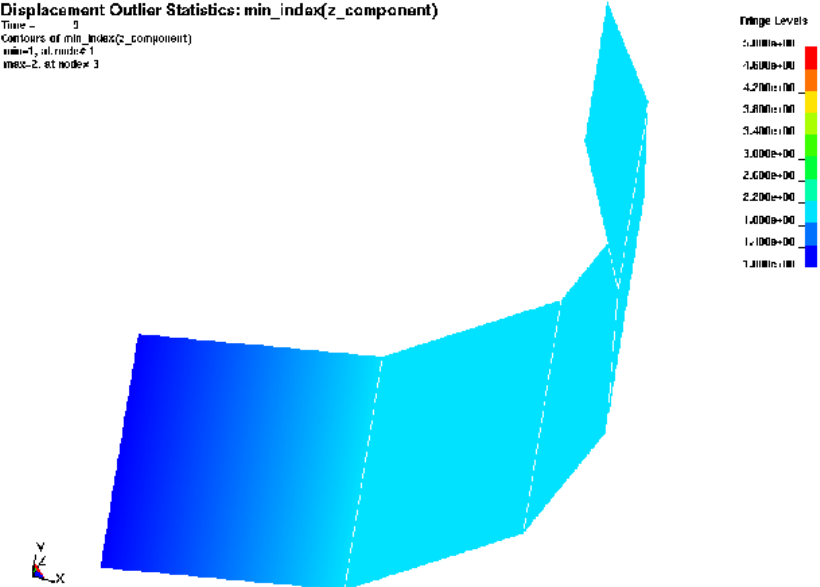


Figure 19-19: Index of run with minimum z-component displacement

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:

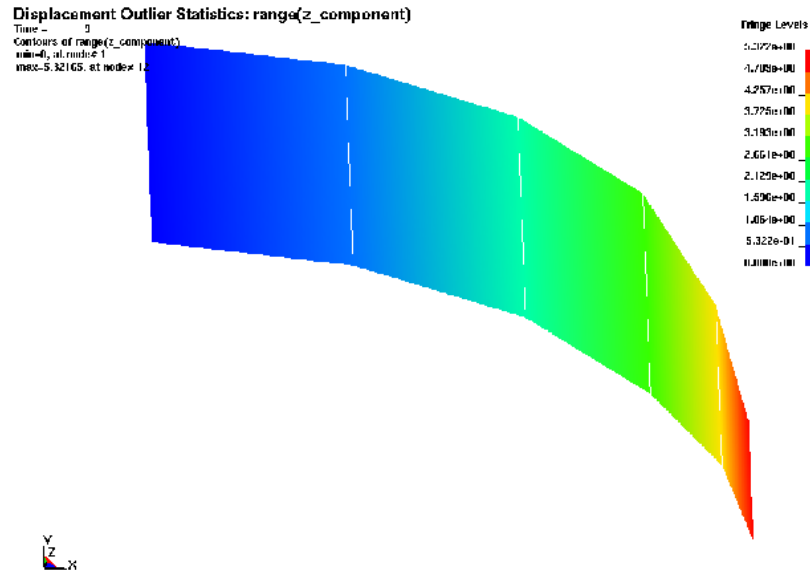


Figure 19-20: Second buckling mode

### 19.3. 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 17.2. 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:

$$\begin{aligned} \min \quad & \text{HIC}(15 \text{ ms}) \\ \text{s.t.} \quad & \text{Probability}[\text{Intrusion} > 550 \text{ mm}] \leq 10^{-6} \end{aligned}$$

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 17.2 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 ( $t_{\text{hood}}, t_{\text{bumper}} \in [1, 5]$ ) 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 19-21.

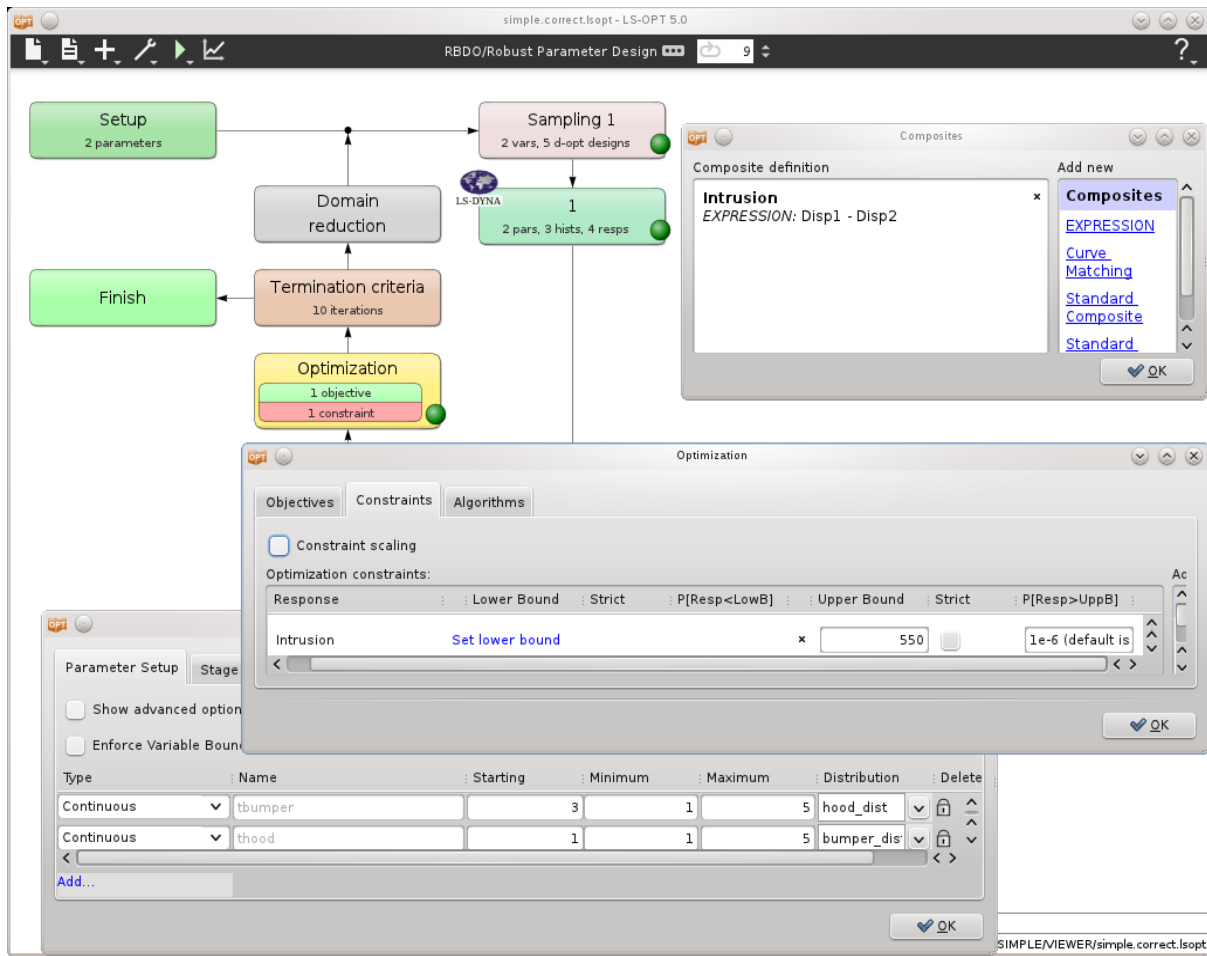


Figure 19-21: Probabilistic variable and constraint definition for RBDO

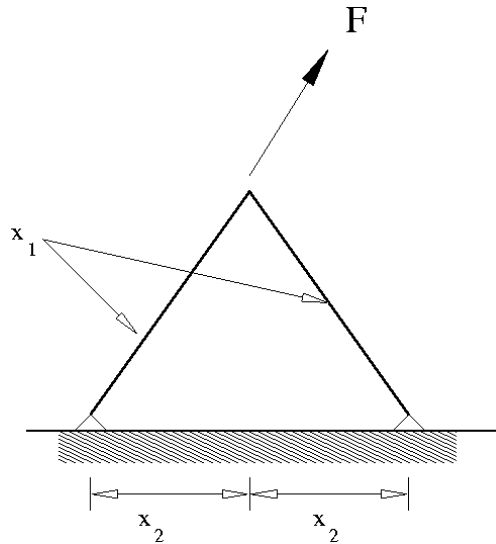
The results are  $t_{hood} = 1.7$ ,  $t_{bumper} = 5$ , a HIC value of 139, and an intrusion of 545 with standard deviation 1.01.

## 19.4. Robust Parameter Design

This example has the following features:

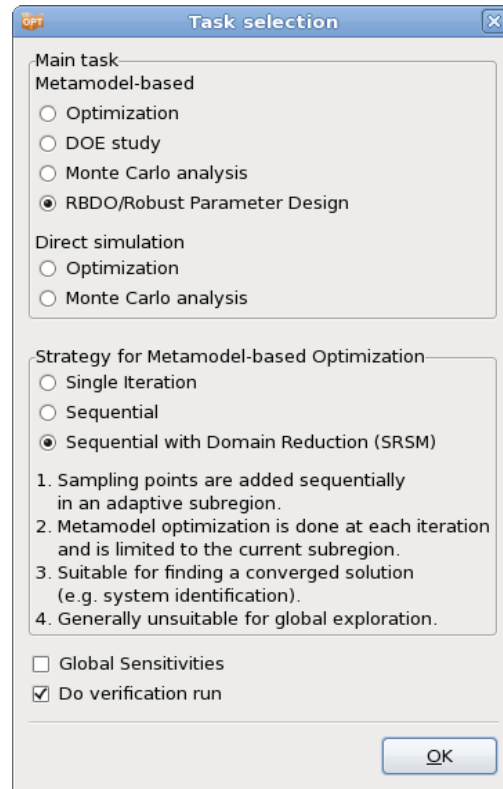
- Reliability based design optimization
- Standard deviation composite

Consider the two-bar truss problem as shown in Figure 19-22. Variable  $x_1$ , the area, 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,  $x_2$ , is a control variable which will be adjusted to control the variance of the stress response, Figure 19-24. The standard deviation of stress response is considered as the objective for the robust design process, Figure 19-25.



**Figure 19-22: 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 19-23. A response surface considering the effect of variables and the interaction between variables is used to approximate the stress response.



*Figure 19-23: Task Reliability based design optimization/Robust Parameter Design*

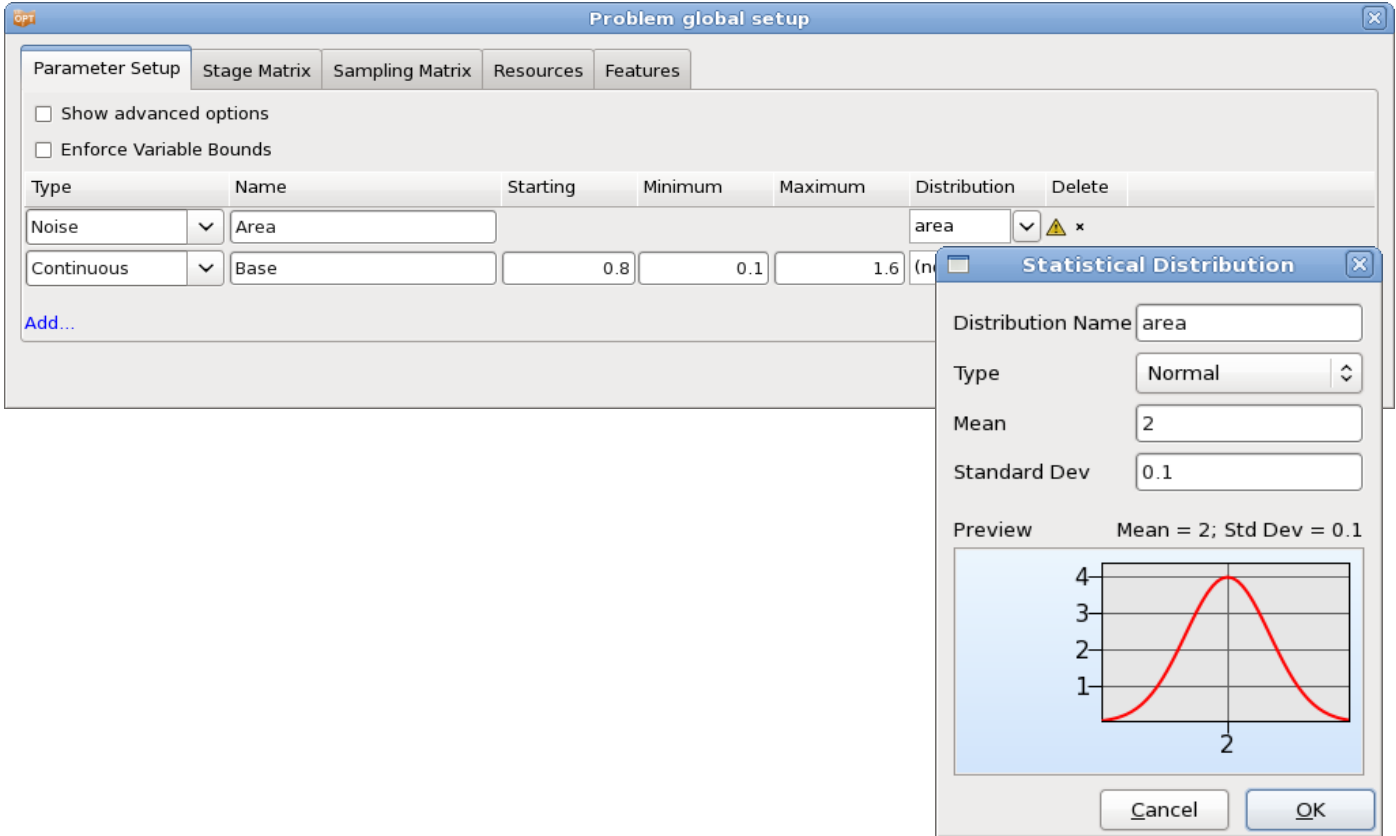
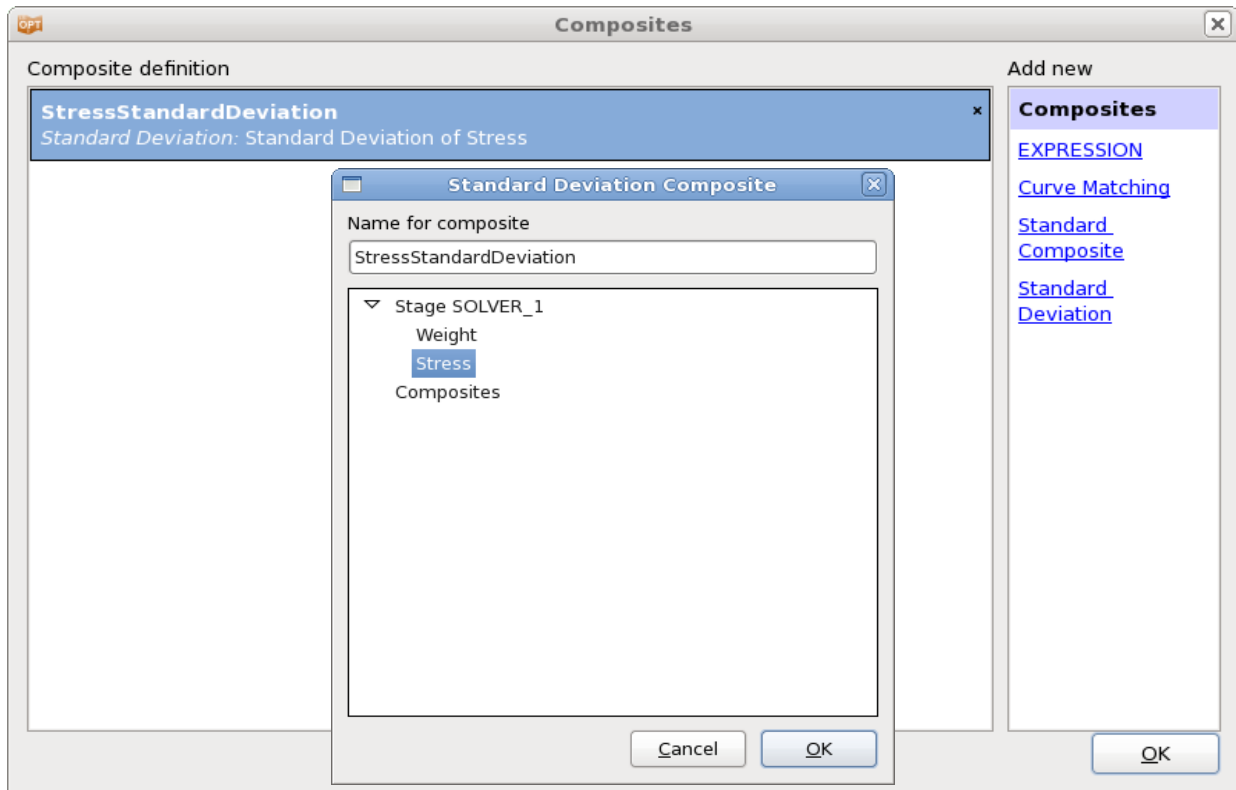


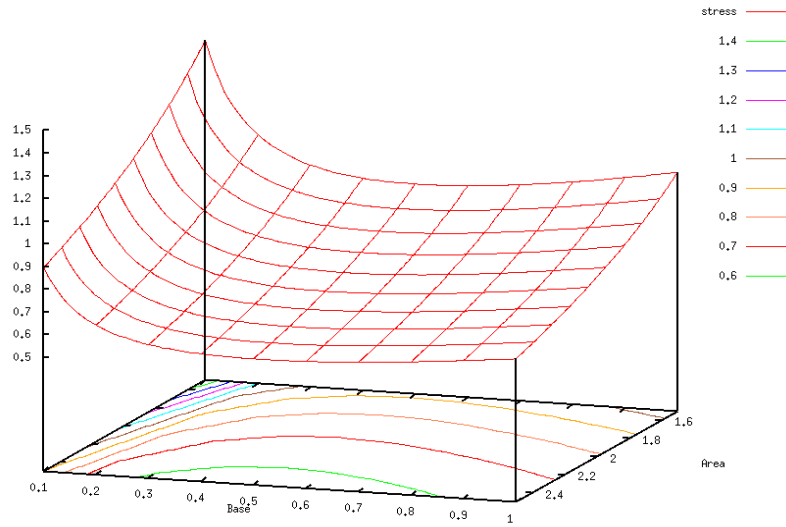
Figure 19-24: Parameter Setup and Distribution



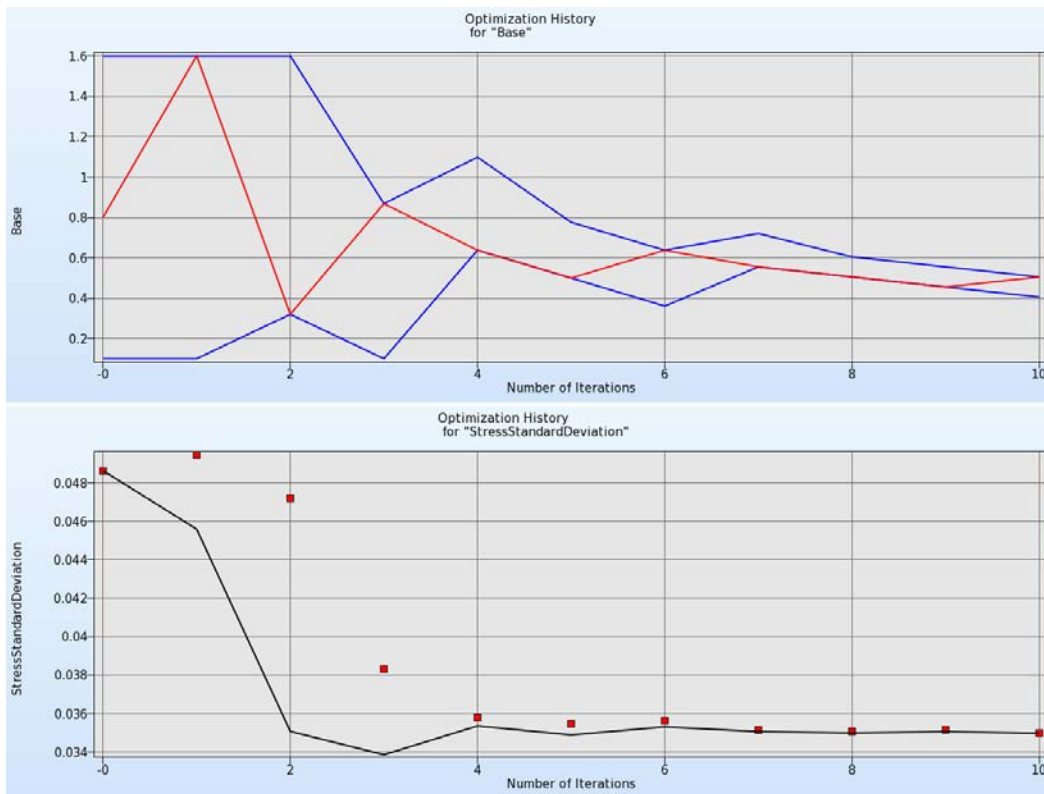


**Figure 19-25: Definition of Standard deviation composite**

The actual stress response is shown in Figure 19-26. 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 19-27. 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 19-26: Contours of stress response. The flattest part of the response is when variable 'base' equals 0.5.**



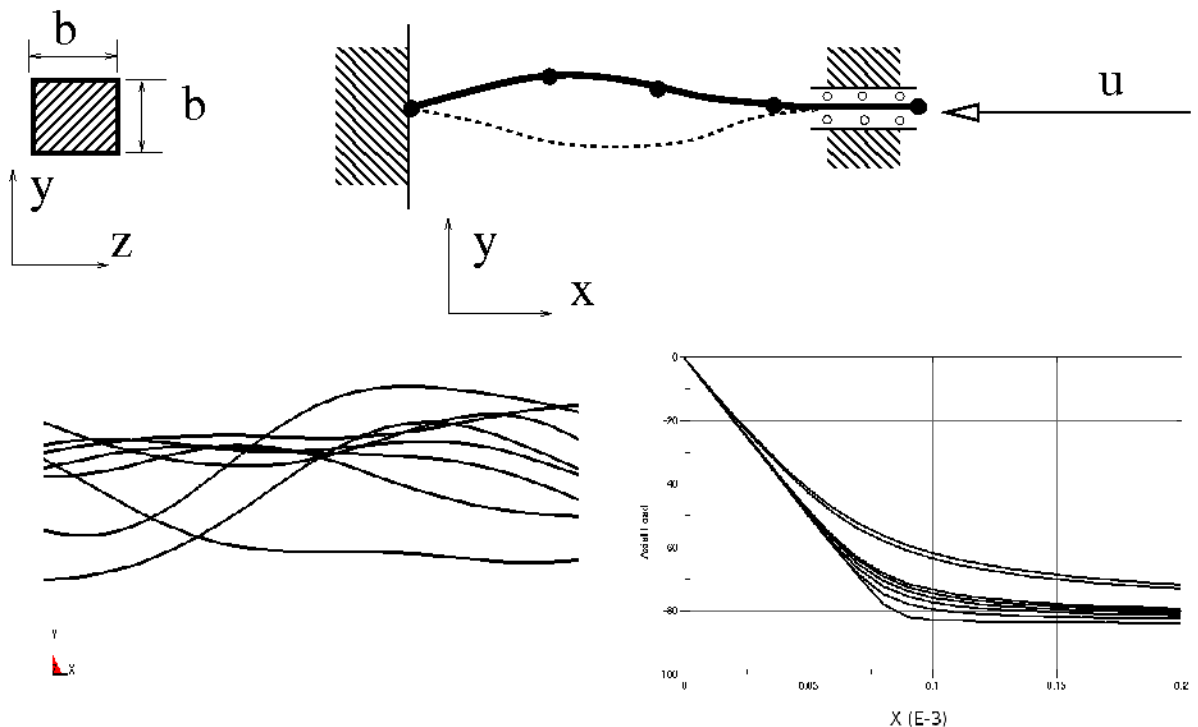
**Figure 19-27: Optimization histories. Design variable 'base' is shown on the left and the standard deviation of the stress response is shown on the right.**

## 19.5. 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 19-28 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 19-28: 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  $-0.002$ .

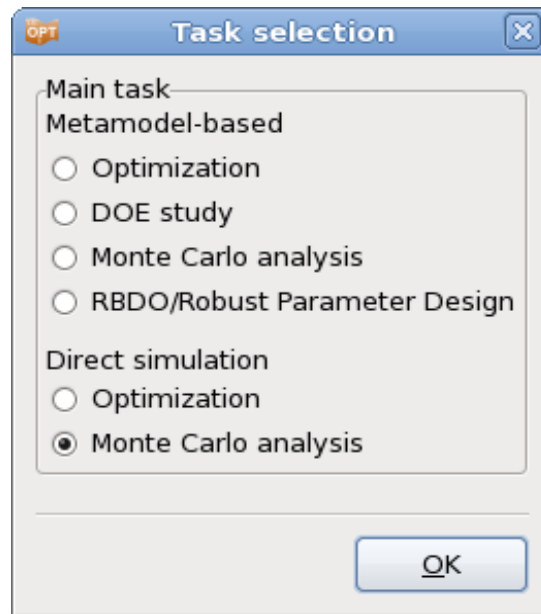
The perturbation is created using the spectral method to have an autocorrelation function described by a Gaussian correlation function. The Gaussian correlation function is  $P(x) = e^{-(ax)^2}$  with  $s$  distance and the constant  $a = 0.1$  in this study. The resulting perturbation is scaled by 0.01. The LS-DYNA® \*PERTURBATION card is:

```
*PERTURBATION_NODE
$type, nid, scl, cmp, icoord, cid
4, , 1.e-2, 3
$cstype, e1, e2, rnd
1, , ,
1, 1.e-1
```

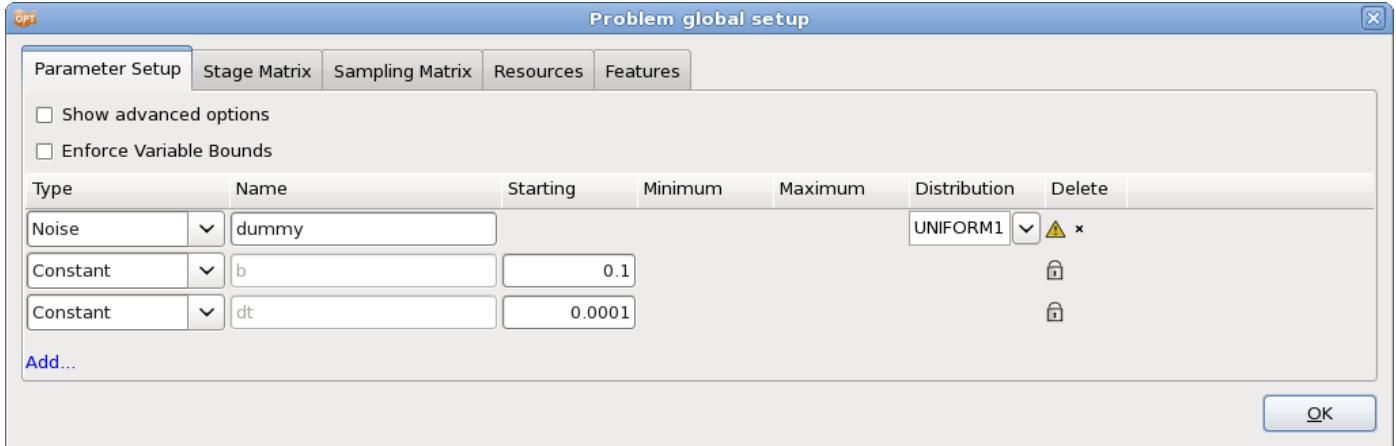
### 19.5.1. Using only a stochastic field

Firstly, a Monte Carlo analysis is done (Figure 19-29) considering only the geometric stochastic fields. The stochastic field is set to vary freely using the LS-DYNA<sup>®</sup> keyword `*PERTURBATION`. Every LS-OPT analysis needs a variable, so we added a dummy variable that does not do anything, Figure 19-30. 50 simulations are run, Figure 19-31. 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<sup>®</sup> `*PERTURBATION` keyword, which can be useful for many reasons, such as having only certain stochastic fields.

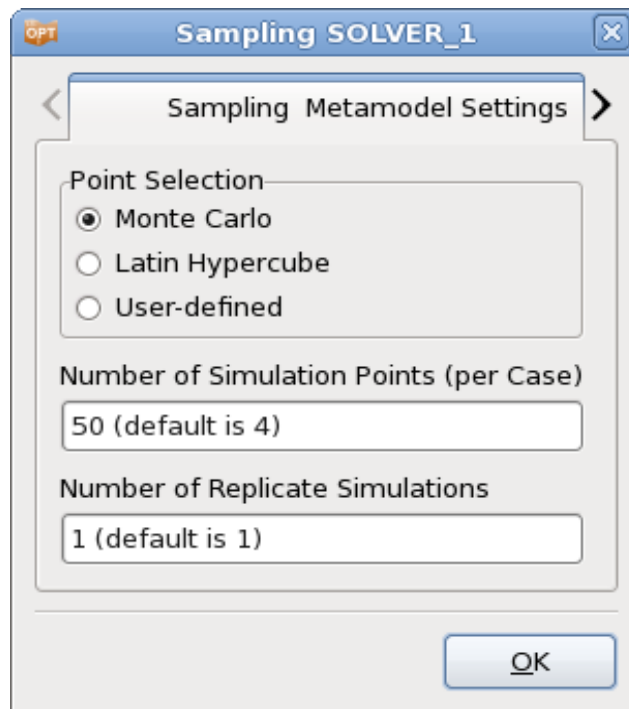
The histogram of the responses is shown in Figure 19-32. Note that the distribution has a characteristic shape.



*Figure 19-29: Task dialog. Select the main task Monte Carlo analysis.*



**Figure 19-30: Parameter setup.** Since the variable *dummy* is not used, an arbitrary distribution can be used here.



**Figure 19-31: Sampling dialog.** Specify the number of simulation points. The point selection doesn't affect the analysis here, since the variable is not used.

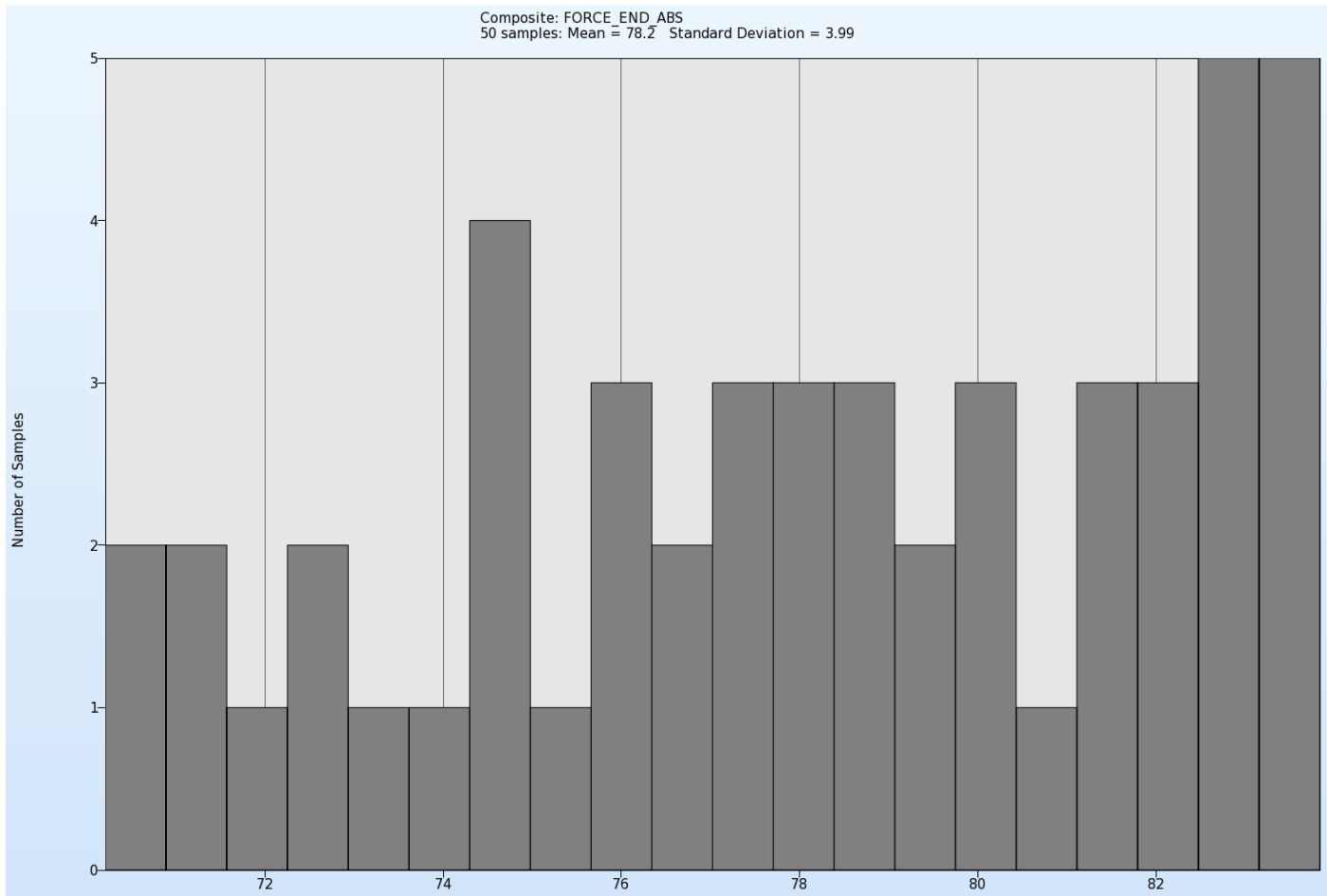


Figure 19-32: Histogram of the responses using only a stochastic field.

### 19.5.2. A variable and a stochastic field

In this example a variable as well as the stochastic field are used to do the analysis, **Error! Reference source not found.** Figure 19-34 displays the resulting forces. The stochastic field is allowed to vary freely using the \*PERTURBATION keyword like in Section 19.5.1, but the dimension  $b$  of the cross-section is also varied based on a Normal distribution, **Error! Reference source not found.**

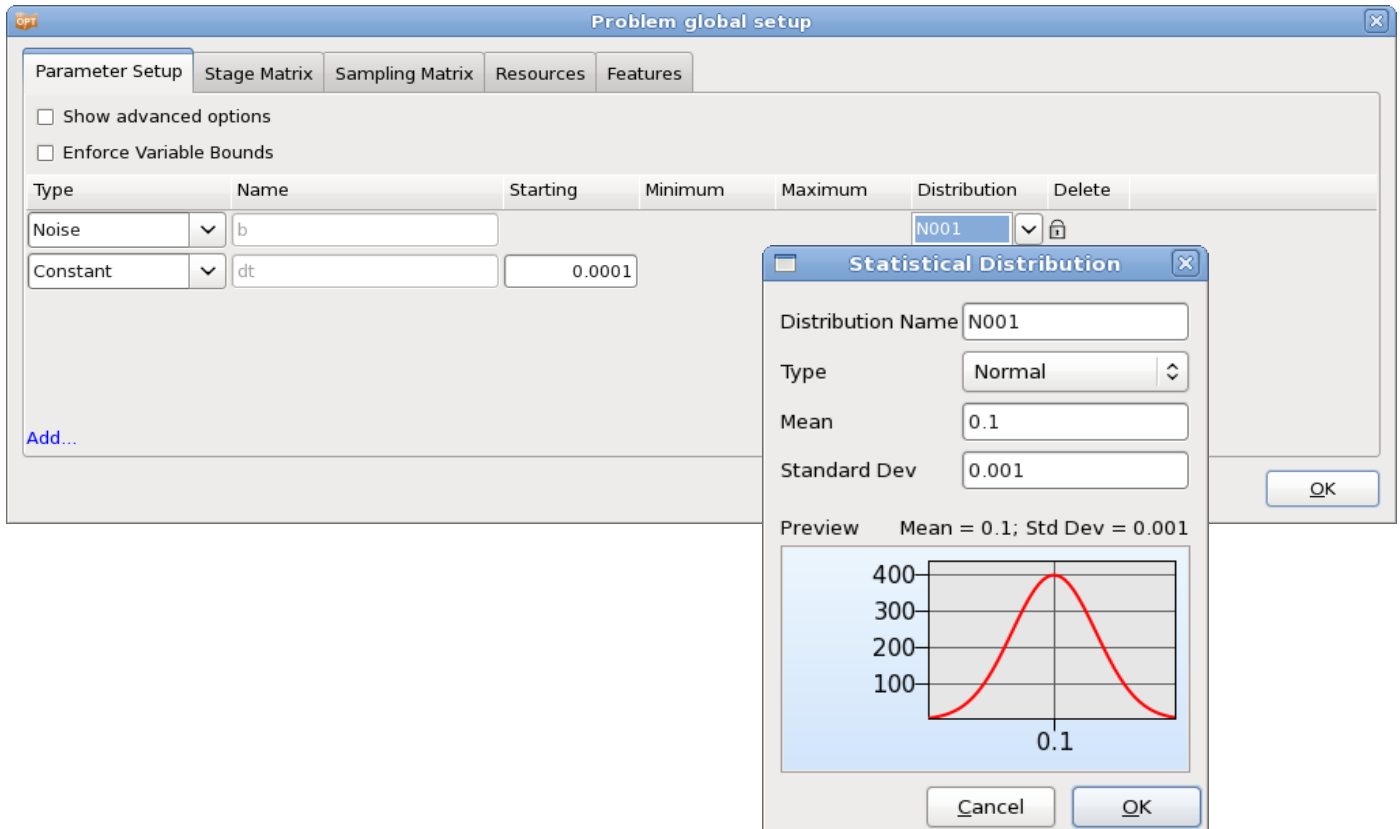
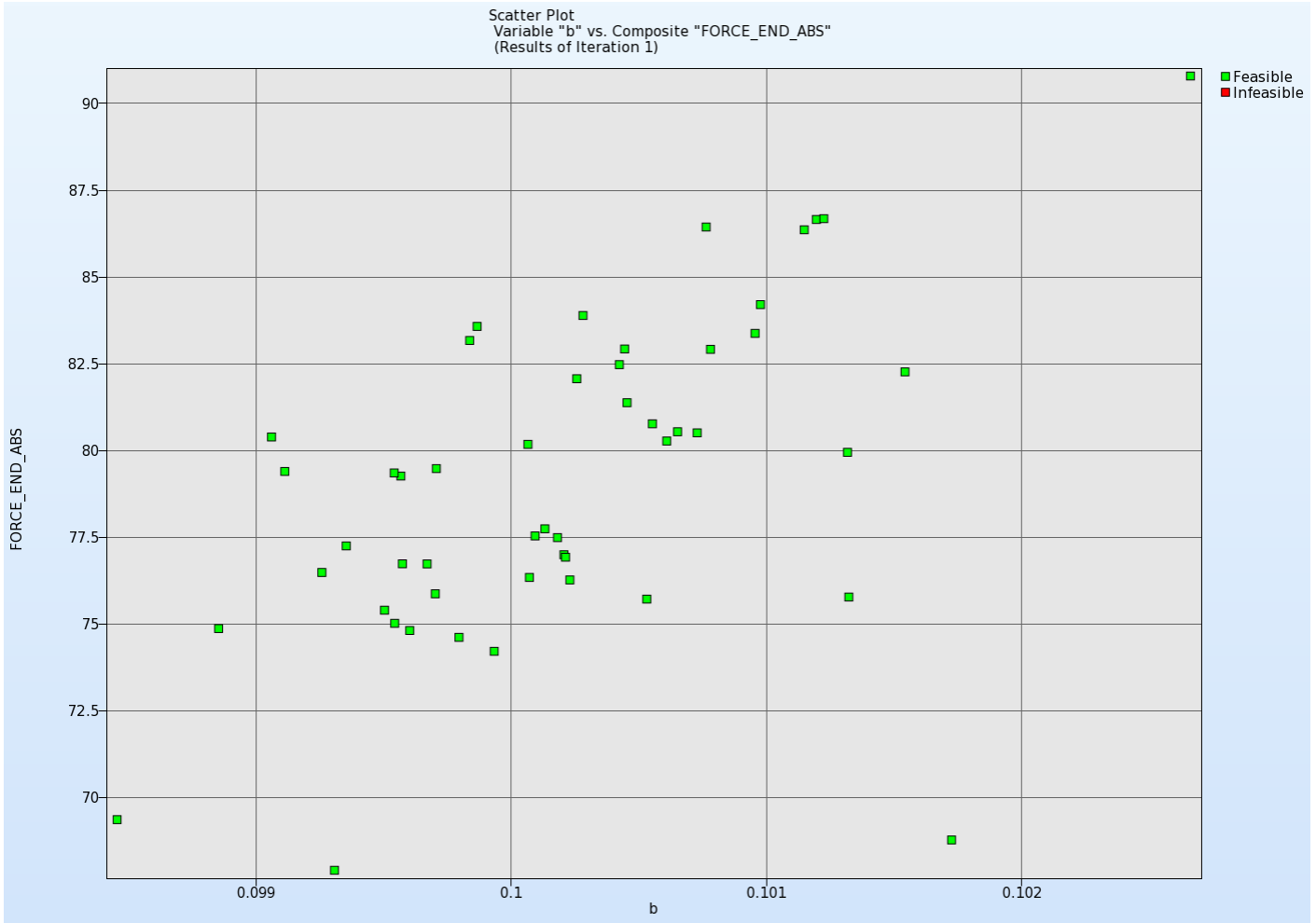


Figure 19-33: Noise variable and distribution definition



**Figure 19-34:** Plot of the responses using the thickness variable and a stochastic field.

### 19.5.3. Replicate experiments using stochastic fields

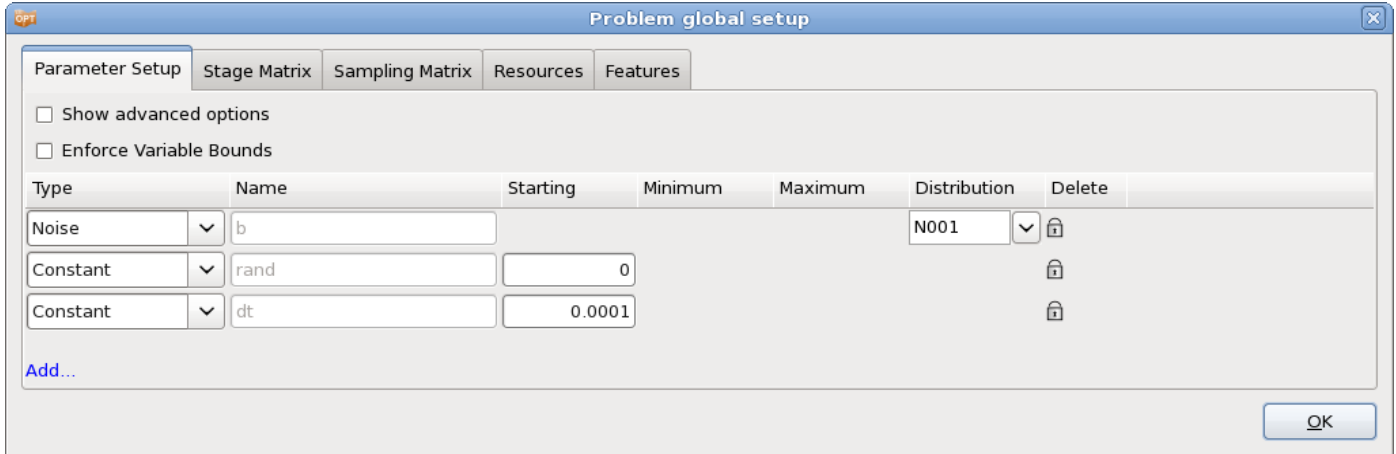
In this example a variable as well as the stochastic field are used to do the analysis. Replicate runs are done at each experimental point with different values of the stochastic field.

In LS-DYNA, we add the random seed of the stochastic field as a variable, Figure 19-35. We let it vary freely by setting the seed to zero:

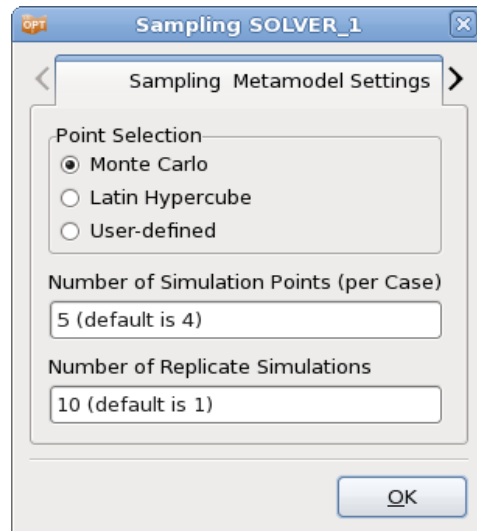
```
*PARAMETER
irand, 0
$
*PERTURBATION_NODE
$type, nid, scl, cmp, icoord, cid
4, , 1.e-2, 3,
$cstype, e1, e2, rnd
1, , , &rand
$
1, 1.e-1
$
```

In LS-OPT we use replicate experiments to analyze, Figure 19-36.





*Figure 19-35: Parameter Setup.*



*Figure 19-36: Sampling dialog. 10 replicate runs are done at each experimental point.*



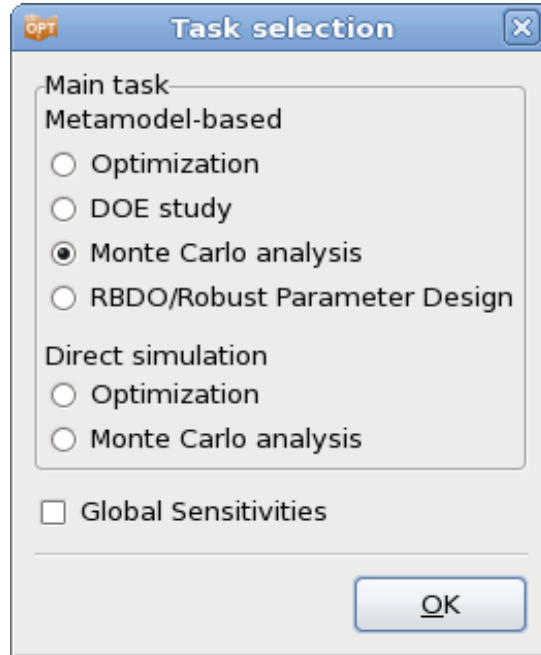


Figure 19-38: Task Metamodel-based Monte Carlo analysis

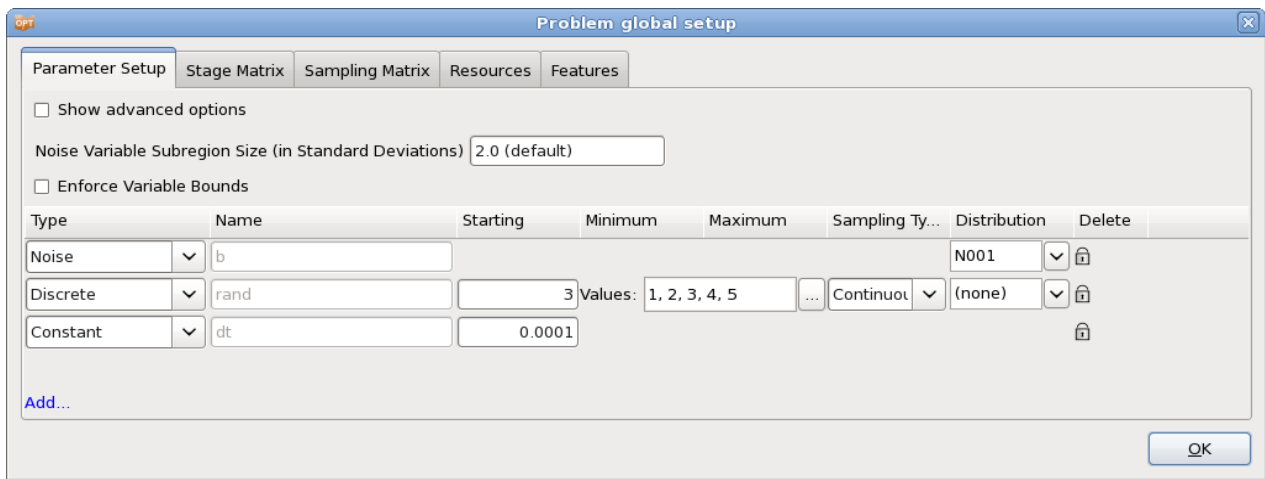


Figure 19-39: Parameter Setup.

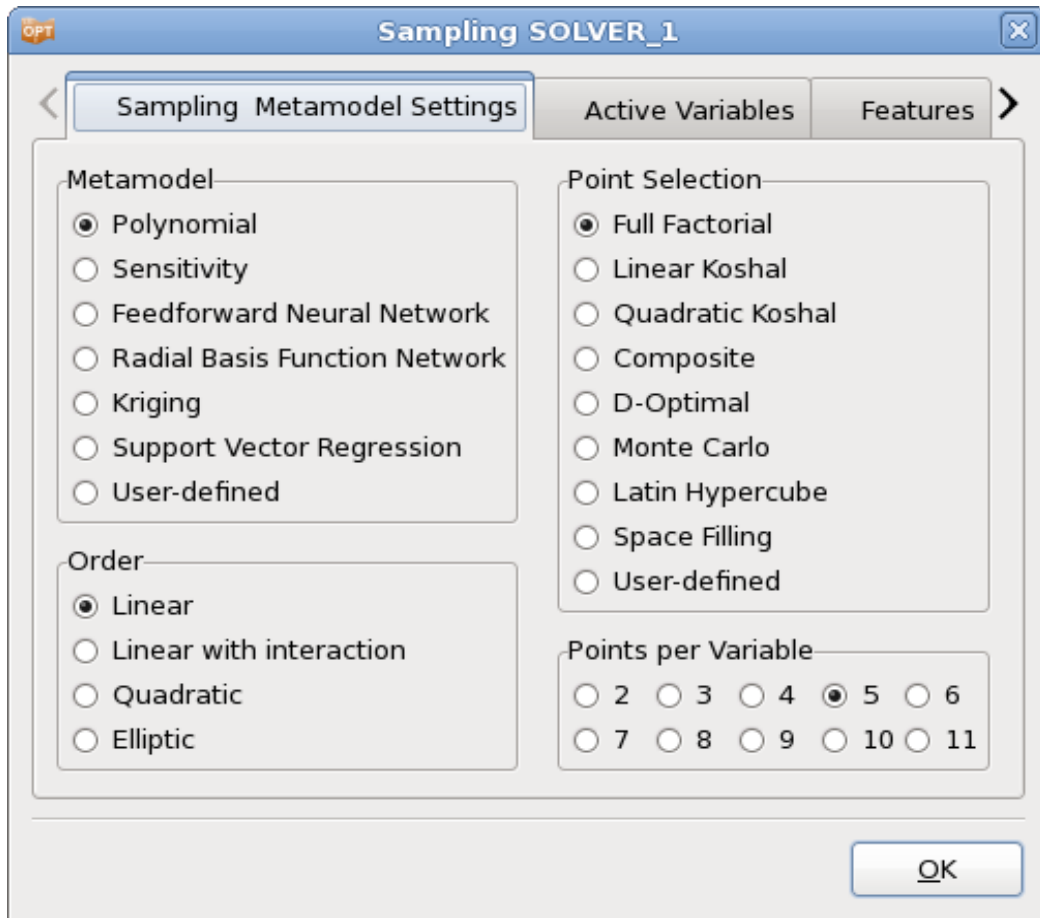


Figure 19-40: Sampling dialog.

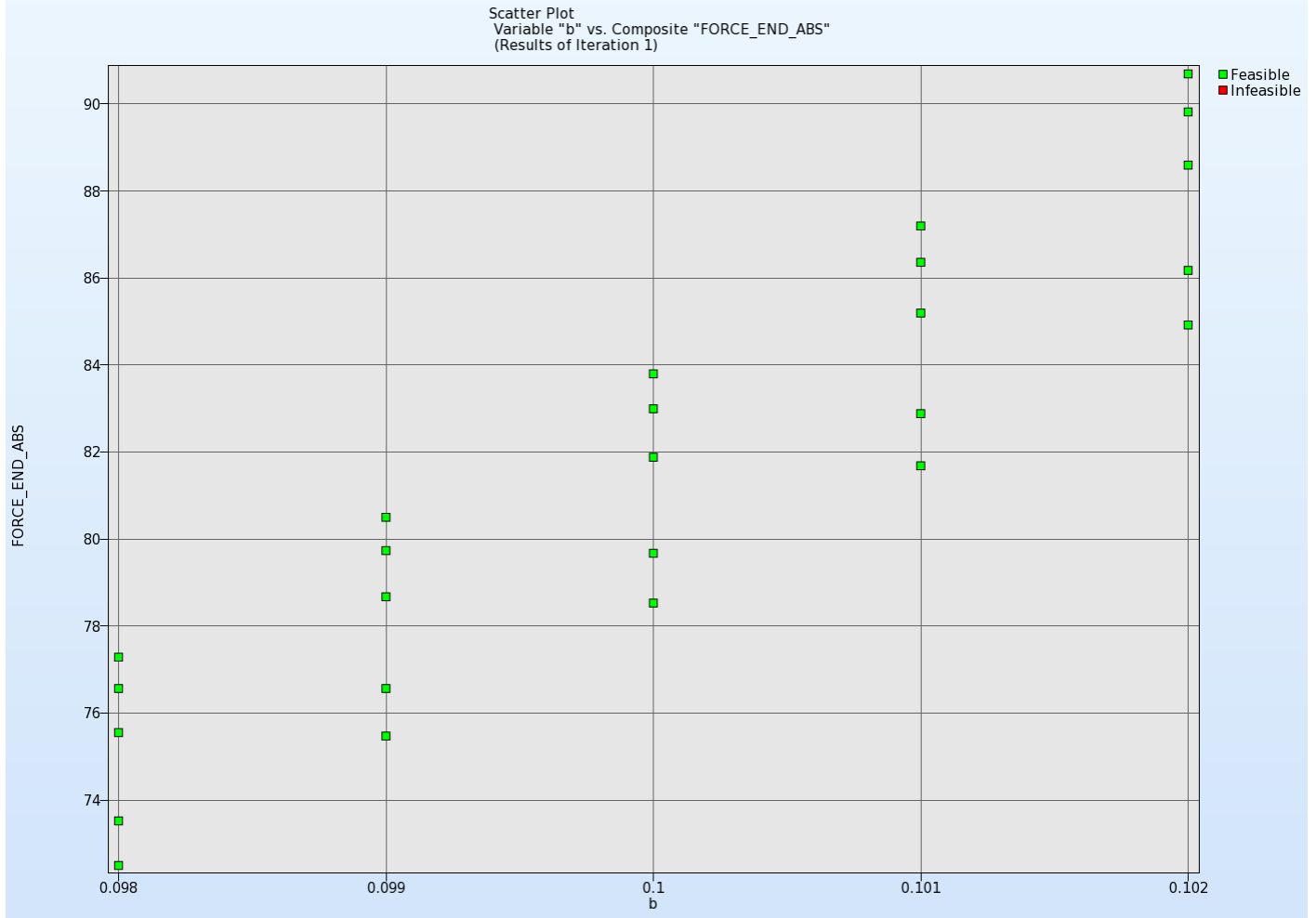


Figure 19-41: Plot of the responses using the same five stochastic fields in the replicates.

## 19.6. REFERENCES

- [1] Yamazaki, K., Han, J., Ishikawa, H., Kuroiwa, Y. Maximization 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](http://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.

# III – Theory

# 20. Response Surface Methodology

## 20.1. Introduction

An authoritative text on Response Surface Methodology (RSM) [7] 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 [8], it has only recently been actively applied to mechanical design [9]. 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 [9]-[12]. RSM can be categorized as a Metamodeling technique (see Chapter 21 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.

### 20.1.1. 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  $\mathbf{x}$ . The exact functional relationship between these quantities is

$$y = \eta(\mathbf{x}) \tag{20-1}$$

The exact functional relationship is now approximated (e.g. polynomial approximation) as

$$\eta(\mathbf{x}) = f(\mathbf{x}) \tag{20-2}$$

The approximating function  $f$  is assumed to be a summation of basis functions:

$$f(\mathbf{x}) = \sum_{i=1}^L a_i \phi_i(\mathbf{x}) \tag{20-3}$$

where  $L$  is the number of basis functions  $\phi_i$  used to approximate the model.

The constants  $\mathbf{a} = [a_1, a_2, \dots, a_L]^T$  have to be determined in order to minimize the sum of the square error:

$$\sum_{p=1}^P \{ [y(x_p) - f(x_p)]^2 \} = \sum_{p=1}^P \left\{ \left[ y(x_p) - \sum_{i=1}^L a_i \phi_i(x_p) \right]^2 \right\} \tag{20-4}$$

$P$  is the number of experimental points and  $y$  is the exact functional response at the experimental points  $\mathbf{x}_i$ .

The solution to the unknown coefficients is:

$$\mathbf{a} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{20-5}$$

where  $\mathbf{X}$  is the matrix

$$\mathbf{X} = [X_{ui}] = [\phi_i(x_u)] \tag{20-6}$$

The next critical step is to choose appropriate basis functions. A popular choice is the quadratic approximation

$$\phi = [1, x_1, \dots, x_n, x_1^2, x_1 x_2, \dots, x_1 x_n, \dots, x_n^2]^T \tag{20-7}$$

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.

### 20.1.2. Factors governing the accuracy of the response surface

Several factors determine the accuracy of a response surface [7].

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, overfitting (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 [7].

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 [7] (very roughly).

### 20.1.3. 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.

### 20.1.4. Other types of response surfaces

Neural and Radial Basis Function networks and Kriging approximations can also be used as response surfaces and are discussed under the heading of *metamodels* in Sections 21.1 and 21.2.

## 20.2. 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 [7]. The factorial, Koshal, composite,  $D$ -optimal and Latin Hypercube designs are detailed here.

### 20.2.1. 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.

### 20.2.2. 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:

$$\begin{array}{ccc} x_1 & x_2 & x_3 \\ \left[ \begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right] \end{array}$$

As a result, four coefficients can be estimated in the linear model

$$\phi = [1, x_1, \dots, x_n]^T \quad (20-8)$$

#### Second order model

For  $n = 3$ , the coordinates are:

$$\begin{array}{ccc} x_1 & x_2 & x_3 \\ \left[ \begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right] \end{array}$$

As a result, ten coefficients can be estimated in the quadratic model

$$\phi = [1, x_1, \dots, x_n, x_i^2, x_1x_2, \dots, x_1x_n, \dots, x_n^2]^T \quad (20-9)$$

### 20.2.3. 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:

$x_1$	$x_2$	$x_3$
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

The points are used to fit a second-order function. The value of  $\alpha = \sqrt[4]{2^n}$ .

### 20.2.4. D-optimal design

This method uses a subset of all the possible design points as a basis to solve  $\max |X^T X|$ . 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 [13]. 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 [7] and [11].

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 [13]. The factorial design referred to below is based on a regular grid of  $2^n$  points (linear) or  $3^n$  points (quadratic).

**Table 20-1: 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

### 20.2.5. 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  $S = S_{ij}$  that randomly samples the entire design space broken down into  $P$  equal-probability regions:

$$S_{ij} = (\eta_{ij} - \zeta_{ij}) / P, \quad (20-10)$$

where  $\eta_{1j}, \dots, \eta_{Pj}$  are uniform random permutations of the integers 1 through  $P$  and  $\zeta_{ij}$  independent random numbers uniformly distributed between 0 and 1. A common simplified version of LHS has centered points of  $P$  equal-probability sub-intervals:

$$S_{ij} = (\eta_{ij} - 0.5) / P \quad (20-11)$$

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 22.10). 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 20.2.6 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:

$$CL_2^2 = (13/12)^n - \frac{2}{P} \sum_{i=1 \dots P} \prod_{j=1 \dots n} \left( 1 + \frac{|S_{ij} - 0.5|}{2} - \frac{|S_{ij} - 0.5|^2}{2} \right) + \frac{1}{P_2} \sum_{k=1 \dots P} \sum_{i=1 \dots P} \prod_{j=1 \dots n} \left( 1 + \frac{|S_{kj} - 0.5|}{2} + \frac{|S_{ij} - 0.5|}{2} - \frac{|S_{kj} - S_{ij}|}{2} \right). \quad (20-12)$$

### 20.2.6. 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, [14], [15]. 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.

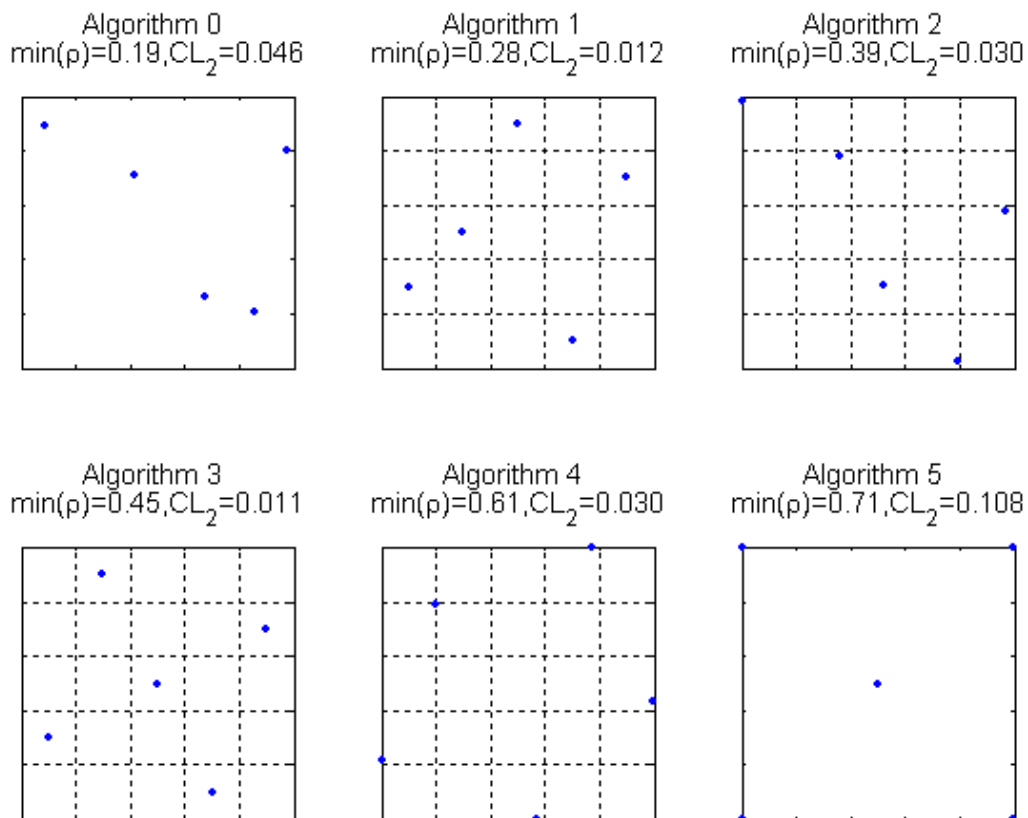
The constrained randomization method termed Latin Hypercube Sampling (LHS) and proposed in [16], has become a popular strategy to generate points on the 'box' (hypercube) design region. The method implies that on each level of every design variable only one point is placed, and the number of levels is 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 [17]) 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 [18] 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 =  $3^{20} \approx 3.5 \cdot 10^9$ .
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 20-2). Only Algorithm 5 has been made available in the graphical interface LS-OPTui.



*Figure 20-1: Six space-filling designs: 5 points in a 2-dimensional box region*



**Table 20-2: 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 <i>simulated annealing</i> . 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 22.10).

### Discussion of algorithms

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. 2.21). 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:

$$S'_{ij} = S_{ij} + \xi \quad (20-13)$$

where  $\xi$  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. (2.23) 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:

$$T_{\max} := T_{\max} \times \Delta E, \quad (20-14)$$

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: 0.
  - 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,  $v_T$ .
    - parameter temperature update interval
  - Parameters that control the standard deviation of  $\xi$  in (2.13):
    - upper bound,  $\sigma_{\max}$ ,
    - lower bound,  $\sigma_{\min}$ .
  - Termination criterion: maximal number of energy function evaluations,  $N_{it}$ .

### 20.2.7. Random number generator

The Mersenne Twister [19] 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.

### 20.2.8. 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.

### 20.3. 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.

#### 20.3.1. Residual sum of squares

For the predicted response  $\hat{y}_i$  and the actual response  $y_i$ , this error is expressed as

$$\varepsilon^2 = \sum_{i=1}^P (y_i - \hat{y}_i)^2 \quad (20-15)$$

If applied only to the regression points, this error measure is not very meaningful unless the design space is oversampled e.g.,  $\varepsilon = 0$  if the number of points  $P$  equals the number of basis functions  $L$  in the approximation.

#### 20.3.2. RMS error

The residual sum-of-squares is sometimes used in its square root form,  $\varepsilon_{RMS}$ , and called the ‘‘RMS error’’:

$$\varepsilon_{RMS} = \sqrt{\frac{1}{P} \sum_{i=1}^P (y_i - \hat{y}_i)^2} \quad (20-16)$$

#### 20.3.3. Maximum residual

This is the maximum residual considered over all the design points and is given by

$$\varepsilon_{\max} = \max |y_i - \hat{y}_i|. \quad (20-17)$$

#### 20.3.4. 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  $\mathbf{X}$

$$\mathbf{X}_a(x_p) = \begin{bmatrix} \mathbf{X} \\ \mathbf{A}(x_p) \end{bmatrix} \quad (20-18)$$

and solving

$$\max |\mathbf{X}_a^T \mathbf{X}_a| = \max |\mathbf{X}^T \mathbf{X} + \mathbf{A}^T \mathbf{A}| \quad (20-19)$$

for  $x_p$ .

### 20.3.5. PRESS residuals

The prediction sum of squares residual (PRESS) uses each possible subset of  $P - 1$  responses as a regression data set, and the remaining response in turn is used to form a prediction set [7]. PRESS can be computed from a single regression analysis of all  $P$  points.

$$PRESS = \sum_{i=1}^P \left( \frac{y_i - \hat{y}_i}{1 - h_{ii}} \right)^2, \quad (20-20)$$

where  $h_{ii}$  are the diagonal terms of

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T. \quad (20-21)$$

$\mathbf{H}$  is the “hat” matrix, the matrix that maps the observed responses to the fitted responses, i.e.

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}. \quad (20-22)$$

The PRESS residual can also be written in its square root form

$$SPRESS = \sqrt{\frac{1}{P} \sum_{i=1}^P \left( \frac{y_i - \hat{y}_i}{1 - h_{ii}} \right)^2}. \quad (20-23)$$

For a saturated design,  $\mathbf{H}$  equals the unit matrix  $\mathbf{I}$  so that the PRESS indicator becomes undefined.

### 20.3.6. The coefficient of multiple determination $R^2$

The coefficient of determination  $R^2$  is defined as:

$$R^2 = \frac{\sum_{i=1}^P (\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^P (y_i - \bar{y}_i)^2} \quad (20-24)$$

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.

### 20.3.7. $R^2$ for Prediction

For the purpose of *prediction* accuracy the  $R^2_{prediction}$  indicator has been devised [7].

$$R^2_{prediction} = 1 - \frac{PRESS}{S_{yy}}, \quad (20-25)$$

where

$$S_{yy} = \mathbf{y}^T \mathbf{y} - \frac{1}{P} \left( \sum_{i=1}^P y_i \right)^2. \quad (20-26)$$

$R^2_{prediction}$  represents the ability of the model to detect the variability in predicting new responses [7].

### 20.3.8. 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).

## 20.4. 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.

### 20.4.1. The confidence interval of the regression coefficients

The  $100(1 - \alpha)\%$  confidence interval for the regression coefficients  $b_j \wedge j = 0, 1, \dots, L$  is determined by the inequality

$$b_j - 0.5\Delta b_j \leq \beta_j \leq b_j + 0.5\Delta b_j, \quad (20-27)$$

where

$$\Delta b_j(\alpha) = 2t_{\alpha/2, P-L} \sqrt{\hat{\sigma}^2 C_{jj}}, \quad (20-28)$$

and  $\hat{\sigma}^2$  is an unbiased estimator of the variance  $\sigma^2$  given by

$$\hat{\sigma}^2 = \frac{\varepsilon^2}{P-L} = \frac{\sum_{i=1}^P (y_i - \hat{y}_i)^2}{P-L} \quad (20-29)$$

$C_{jj}$  is the diagonal element of  $(\mathbf{X}^T \mathbf{X})^{-1}$  corresponding to  $b_j$  and  $t_{\alpha/2, P-L}$  is Student's  $t$ -Distribution.  $100(1 - \alpha)\%$  therefore represents the level of confidence that  $b_j$  will be in the computed interval.

### 20.4.2. 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

$$F = \frac{[\varepsilon_{reduced}^2 - \varepsilon_{complete}^2] / r}{\varepsilon_{complete}^2 / (P-L)} \quad (20-30)$$

where  $r = 1$  and the reduced model is the one in which the regressor variable in question has been removed. Each of the  $\varepsilon^2$  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

$$F = \frac{b_j^2 / C_{jj}}{\varepsilon_{complete}^2 / (P-L)}. \quad (20-31)$$

$F$  can be compared with the  $F$ -statistic  $F_{\alpha, 1, P-L}$  so that if  $F > F_{\alpha, 1, P-L}$ ,  $\beta_j$  is non-zero with  $(100 - \alpha)\%$  confidence. The confidence level  $\alpha$  that  $\beta_j$  is not zero can also be determined by computing the  $\alpha$  for  $F = F_{\alpha, 1, P-L}$ . The importance of  $\beta_j$  is therefore estimated by both the magnitude of  $b_j$  as well as the level of confidence in a non-zero  $\beta_j$ .

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  $\alpha$ . 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.

## 20.5. REFERENCES

- [7] Myers, R.H., Montgomery, D.C. *Response Surface Methodology. Process and Product Optimization using Designed Experiments*. Wiley, 1995.
- [8] 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.
- [9] Toropov, V.V. Simulation approach to structural optimization. *Structural Optimization*, 1, pp. 37-46, 1989.
- [10] Schoofs, A.J.G. *Experimental Design and Structural Optimization*. PhD thesis, Technische Universiteit Eindhoven, August 1987.
- [11] 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.
- [12] Jin, R., Chen, W. and Simpson, T.W. Comparative studies of metamodeling techniques under multiple modeling criteria. *AIAA Paper*, AIAA-2000-4801.
- [13] Roux, W.J. *Structural Optimization using Response Surface Approximations*. PhD thesis, University of Pretoria, April 1997.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] Park, J.-S. Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning Inference*, 39, pp. 95-111, 1994.
- [18] Morris, M., Mitchell, T. Exploratory design for computer experiments. *Journal of Statistical Planning Inference*, 43, pp. 381-402, 1995.
- [19] 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

# 21. 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 20.1), Neural Networks (NN) (Section 21.1.2), Radial Basis Function Networks (RBF) (Section 21.1.3), Kriging (Section 21.2) and Support Vector Regression (SVR) (Section 21.3). 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 20.4). 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.

## 21.1. 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, [5]. 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 21-1). 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 21-1. 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 21-1).

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 21-2), while network input and output layers are usually linear (transparent). In theory, such networks can model functions of almost arbitrary complexity, see [6] and [7]. 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 [8]. For feedforward networks, regularization may be done by controlling the number of network weights (‘model selection’), by imposing penalties on the weights (‘ridge regression’) [9], or by various combinations of these strategies [10].

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 21-3). 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

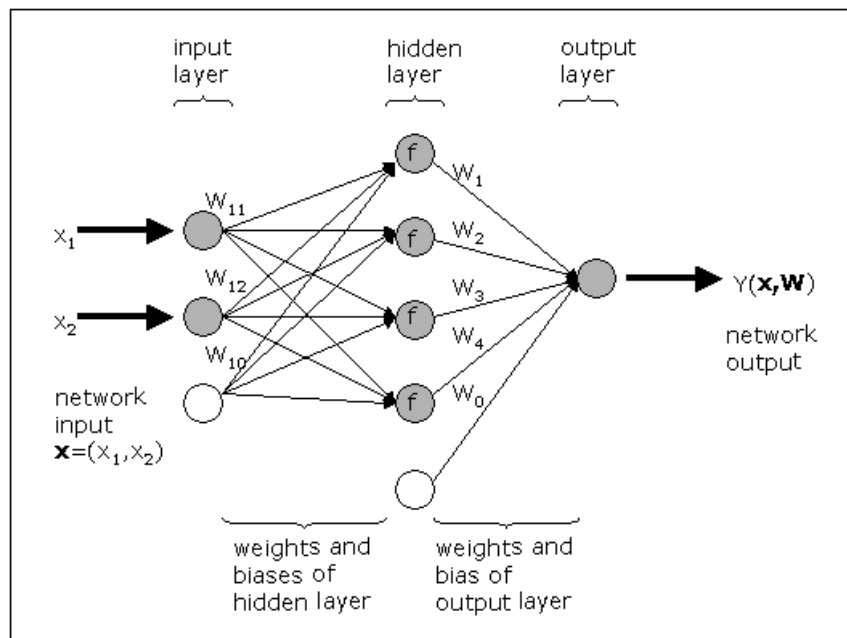
Figure 21-4 and Figure 21-5). The formal proofs of this property can be found, for example, in [11] and [12]. An RBF network can be trained extremely quickly, orders of magnitude faster than a feedforward network. The training process typically takes place in two distinct stages. First, the centers and 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 21-6). If the RBFs are very broad, the network loses fine detail (

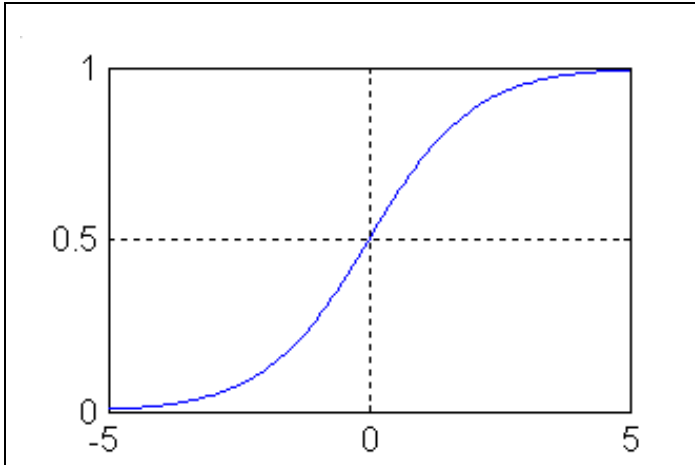
Figure 21-7). This is actually another manifestation of the over/under-fitting dilemma.

In the final shape, *after* training, a multilayer neural network with linear output (Figure 21-1) 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.

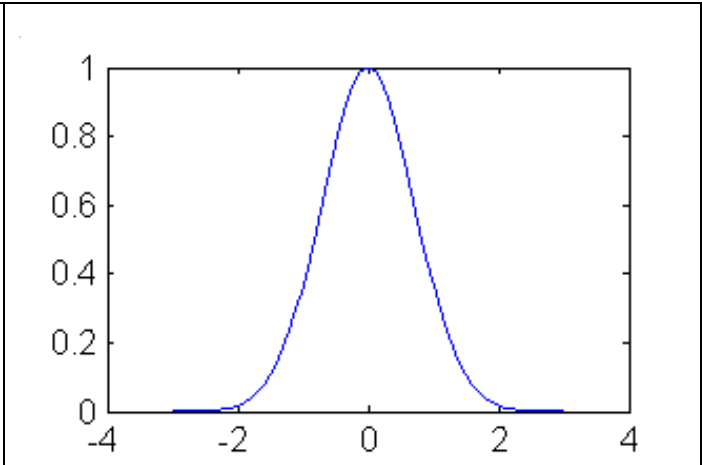
The next sections discuss various goodness-of-fit assessment approaches applicable to neural networks. We also discuss how to estimate the variance of the neural model and how to compute derivatives of a neural network with respect to any of its inputs. Two neural network types, feedforward and radial basis, are considered.



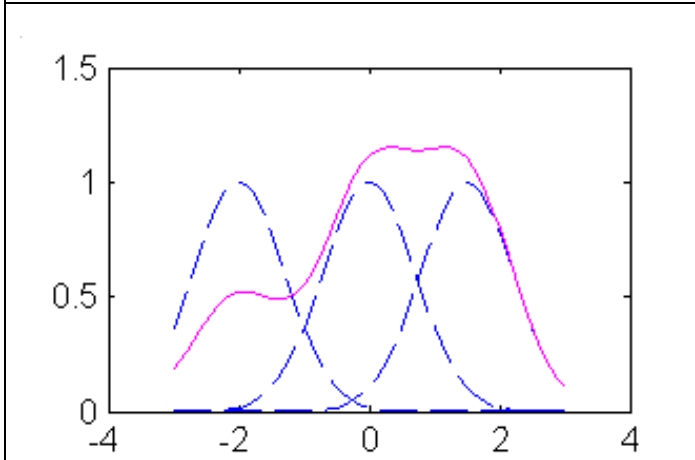
**Figure 21-1: Schematic of a neural network with 2 inputs and a hidden layer of 4 neurons with activation function  $f$ .**



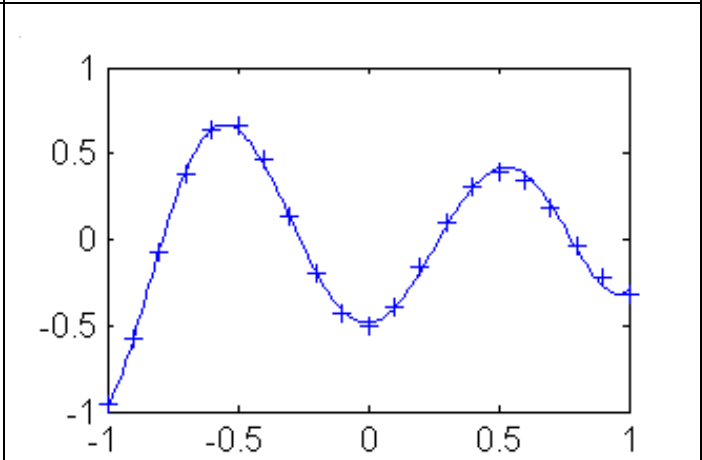
**Figure 21-2:** Sigmoid transfer function  $y=1/(1+\exp(-x))$  typically used with feedforward networks



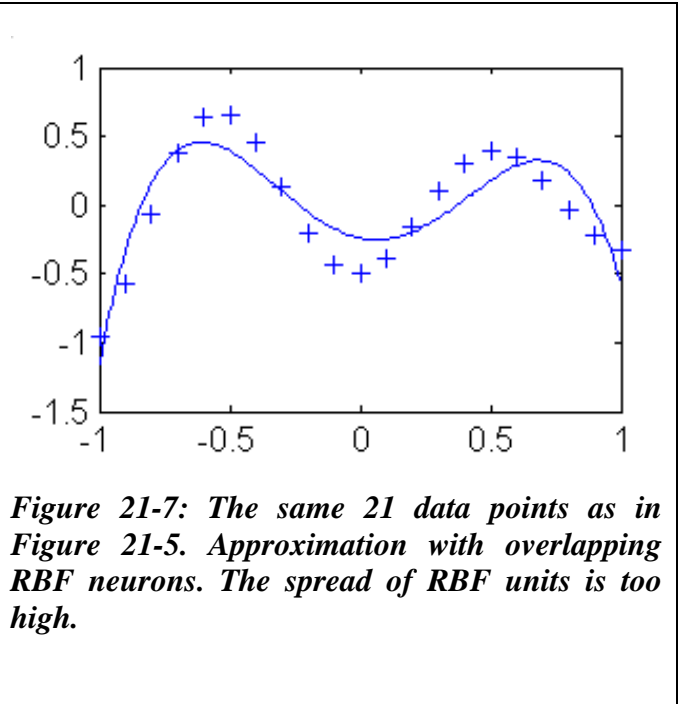
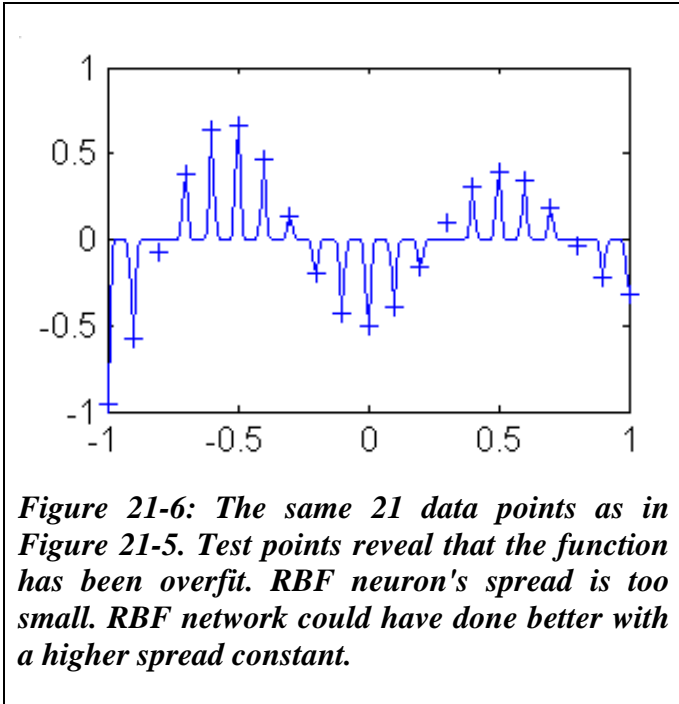
**Figure 21-3:** Radial basis transfer function  $y=\exp(-x^2)$



**Figure 21-4:** Weighted sum of radial basis transfer functions. Three radial basis functions (dashed lines) are scaled and summed to produce a function (solid line).



**Figure 21-5:** A radial basis network approximation (solid line) of the function, which fits the 21 data points (plus symbols).



### 21.1.1. 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:

$$MSE = \sum_i^P (\hat{y}_i - y_i)^2 / P, \tag{21-1}$$

$$RMS = \sqrt{MSE}; \quad nMSE = MSE/\hat{\sigma}^2; \quad nRMS = RMS/\hat{\sigma}. \tag{21-2}$$

$$R^2 = \frac{\sum_i^P (\hat{y}_i - \bar{y})^2}{\sum_i^P (y_i - \bar{y})^2}, \quad \text{and} \quad R = \frac{\sum_i^P |\hat{y}_i - \bar{y}| |y_i - \bar{y}|}{\sum_i^P (y_i - \bar{y})^2 \sum_i^P (y_i - \bar{y})^2}, \tag{21-3}$$

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,  $\bar{y}$  is the mean (average) value of  $\hat{y}$ ,  $\bar{y}$  is the mean (average) value of  $y$ , and  $\hat{\sigma}^2$  is given by

$$\hat{\sigma}^2 = \frac{\varepsilon^2}{P-L} = \frac{\sum_{i=1}^P (y_i - \hat{y}_i)^2}{P-L}. \quad (21-4)$$

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) [13] and Akaike's final prediction error (FPE) [14] 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) [13] and Akaike's final prediction error (FPE) [14] 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:

$$MSE_{GCV} = MSE / (1 - \nu / P)^2, \quad (21-5)$$

$$RMS_{GCV} = \sqrt{MSE_{GCV}}, \quad nMSE_{GCV} = MSE_{GCV} / \hat{\sigma}^2; \quad nRMS_{GCV} = RMS_{GCV} / \hat{\sigma}. \quad (21-6)$$

$$MSE_{FPE} = MSE (1 + \nu / P) / (1 - \nu / P), \quad (21-7)$$

$$RMS_{FPE} = \sqrt{MSE_{FPE}}, \quad nMSE_{FPE} = MSE_{FPE} / \hat{\sigma}^2; \quad nRMS_{FPE} = RMS_{FPE} / \hat{\sigma}. \quad (21-8)$$

where  $\nu$  is the (effective) number of model parameters.

In theory, GCV estimates should be related to  $\nu$ . As a very rough approximation to  $\nu$ , 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  $P \gg M$ . Note that GCV is undefined when  $\nu$  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  $\nu$ . Techniques to assess  $\nu$  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  $v = M$ , 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  $P \gg M$ . Note that both GCV and FPE are undefined when the effective number of model's parameters ( $v$ ) 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  $j = 1, \dots, k$  it is trained on all data except for points from subset  $j$  and then tested on  $j$ -th subset. Formally, let  $y^{(j)} = (y_i^{(j)}), i = 1, \dots, P^{(j)}$  be the prediction of such a model for the points from subset  $j$ . Then the CV- $k$  estimates of accuracy

$$MSE_{CV-k} = \frac{\sum_{j=1}^k \sum_{i=1}^{P^{(j)}} (y_i^{(j)} - \hat{y}_i)^2}{P} \quad (21-9)$$

$$RMS_{CV-k} = \sqrt{MSE_{CV-k}}, nMSE_{CV-k} = MSE_{CV-k} / \hat{\sigma}^2; \quad nRMS_{CV-k} = RMS_{CV-k} / \hat{\sigma}. \quad (21-10)$$

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.

### 21.1.2. 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 21-3), 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  $\mathbf{x} = (x_1, \dots, x_K)$  and network weights  $\mathbf{W} = (W_0, W_1, \dots, W_H, W_{10}, W_{11}, \dots, W_{HK})$ , the output of the network is:

$$\hat{y}(\mathbf{x}, \mathbf{W}) = W_0 + \sum_{h=1}^H W_h f \left( W_{h0} + \sum_{k=1}^K W_{hk} x_k \right), \quad (21-11)$$

where  $f(x) = 1/(1 + e^{-x})$ .

The computational graph of Eq. (3-11) is shown schematically in Figure 21-1. 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. (3-11) with respect to any of its inputs is given by:

$$\frac{\partial \hat{y}}{\partial x_k} = \sum_{h=1}^H W_h W_{hk} f' \left( W_0 + \sum_{h=1}^H W_h \right), k = 1, \dots, K. \quad (21-12)$$

Neural networks have been mathematically shown to be universal approximators of continuous functions and their derivatives (on compact sets) [6]. 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' [8]. 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) [5]. However, when training larger networks, the first-order RPROP algorithm becomes preferable for computational reasons [15].

*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 ([9], [10]). 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  $MSE_{CV-k}$ . 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 [16]. 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. 3-1). When minimizing MSE (Eq. 3-1) 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. 3-1)):

$$F = \beta E_D + \alpha E_W, \quad (21-13)$$

where

$$E_D = 0.5 \sum_{i=1}^P (\hat{y}_i - y_i)^2, \quad E_W = 0.5 \sum_{i=1}^M W_i^2,$$

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. 3-13) 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  $\alpha$  and  $\beta$  is not a trivial task. If we make  $\alpha/\beta$  too small, we may get over-fitting. If  $\alpha/\beta$  is too large, the network will not adequately fit the training data. A rule of thumb is that a little regularization usually helps [17]. 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. (3-13) is moved further away from singularity due to a positive addend to its diagonal:

$$\mathbf{A} = \mathbf{H} + \alpha \mathbf{I} \quad (21-14)$$

where

$$\mathbf{H} = \beta \nabla \nabla E_D \approx \sum_{i=1}^P g(x^{(i)}) g(x^{(i)})^T,$$

$$\mathbf{g}(\mathbf{x}) = \left( \frac{\partial \hat{y}}{\partial W_1}, \dots, \frac{\partial \hat{y}}{\partial W_M} \right)^T.$$

In [5], [18], [19] and [20] 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  $\alpha$  and  $\beta$ . Then, a neural network is trained using a standard non-linear optimization algorithm to minimize the error function (Eq. 3-13). After training, i.e. in the minimum of Eq. (3-13), the values for  $\alpha$  and  $\beta$  are re-estimated, and training restarts with the new performance function. Regularization hyper-parameters are computed in a sequence of 3 steps:

$$\nu = \frac{\sum_{m=1}^M \lambda_m}{\lambda_m + \alpha}, \quad (21-15)$$

where  $\lambda_m$ ,  $m = 1, \dots, M$  are (positive) eigenvalues of matrix  $\mathbf{H}$  in Eq. (3-14),  $\nu$  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. 3-15) relies on numerous simplifications and assumptions, which hold only approximately in typical real-world problems [21]. 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



$\alpha = 1/\sigma_{weights}^2$ . 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  $\sigma_{noise}^2 = 1/\beta$  Gaussian noise on the output variables:

$$y = y(\mathbf{x}) + N(0, \sigma_{noise}^2)$$

With these assumptions, the negative log likelihood of network weights  $\mathbf{W}$  given  $P$  training points  $\mathbf{x}(1), \dots, \mathbf{x}(P)$  is proportional to MSE (Eq. 3-1)), i.e., the maximum likelihood estimate for  $\mathbf{W}$  is that which minimizes (Eq. 3-1) or, equivalently,  $E_D$ . In order for Bayes estimates of  $\alpha$  and  $\beta$  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. 3-1) amounts to minimizing the variance of the model. The estimate for output variance  $\sigma_{\hat{y}|\mathbf{x}}^2$  of the network at a particular point  $\mathbf{x}$  is given by:

$$\sigma_{\hat{y}|\mathbf{x}}^2 \approx \mathbf{g}(\mathbf{x})^T \mathbf{A}^{-1} \mathbf{g}(\mathbf{x}). \quad (21-16)$$

Equation (3-16) is based on a second-order Taylor series expansion of Eq. (3-13) around its minimum and assumes that  $\partial\hat{y}/\partial\mathbf{W}$  is locally linear.

## Variability of Feedforward neural networks

Neural networks have a natural variability because of the following reasons [22]:

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 0). The starting weights for network training are randomly generated using a user-specified seed to ensure repeatability (see Section 20.2.7).

### 21.1.3. 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:

$$g_h(x_1, \dots, x_K) = \sqrt{1 + (r^2 / \sigma_h^2)}. \quad (21-17)$$

Gaussian:

$$g_h(x_1, \dots, x_K) = \exp[-r^2 / 2\sigma_h^2] \quad (21-18)$$

The activation of  $h^{\text{th}}$  radial basis function is determined by the Euclidean distance  $r = \sqrt{\sum_{k=1}^K (x_k - W_{hk})^2}$  between the input vector  $\mathbf{x} = (x_1, \dots, x_K)$  and RBF center  $W_h = (W_{h1}, \dots, W_{hk})$  in  $K$ -dimensional space. The Gaussian basis function is a localized function (peaked at the center and descending outwards) with the property that  $g_h \rightarrow 0$  as  $r \rightarrow \infty$ . Parameter  $\sigma_h$  controls the smoothness properties of the RBF unit.

For a given input vector  $\mathbf{x}$  the output of RBF network with  $K$  inputs and a hidden layer with  $H$  basis function units is given by (see also Eqs. 3-17 and 3-18):

$$Y(\mathbf{x}, \mathbf{W}) = W_0 + \sum_{h=1}^H W_h f(\rho_h) \quad (21-19)$$

where

$$\rho_h = W_{h0} \sum_{k=1}^K (x_k - W_{hk})^2; \quad W_{h0} = 1/(2\sigma_h^2); \quad f(\rho) = \exp(-\rho).$$

Notice that hidden layer parameters  $W_h = (W_{h1}, \dots, W_{hk})$  represent the center of  $h^{\text{th}}$  radial unit, while  $W_{h0}$  corresponds to its deviation. Parameters  $W_0$  and  $W_1, \dots, W_H$  are the output layer's bias and weights, respectively.

A linear super-position of localized functions as in (13) is capable of universal approximation. The formal proofs of this property can be found, for example, in [11] and [12]. It is straightforward to show that the derivative of the network (13) with respect to any of its inputs is given by:

$$\frac{\partial Y}{\partial x_k} = \sum_{h=1}^H W_h W_{h0} \cdot 2(x_k - W_{hk}) \cdot f'(\rho_h), \quad k = 1, \dots, K, \quad (21-20)$$

where  $f'$  denotes the first derivative of the transfer function  $f : f'(\rho) = -\exp(-\rho)$ .

Theory tells us that when a network (Eq. 3-19) 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  $W_{11}, \dots, W_{HK}$  and  $W_{10}, \dots, W_{H0}$ ) must be set. All the basis functions are then kept fixed, while the linear output layer (i.e.,  $W_0, \dots, W_H$ ) 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  $W_{11}, \dots, W_{HK}$  and  $W_{10}, \dots, W_{H0}$  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 (Eq. 3-1). If the hidden layer's parameters  $W_{10}, W_{11}, \dots, W_{HK}$  in (3.4-2) are kept fixed, then the performance function (Eq. 3-1) is a quadratic function of the output layer' parameters  $W_0, \dots, W_H$  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. 3-1) 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  $\alpha$  and  $\beta$  in (3-13) can be found by applying Bayesian re-estimation formulae (Eq. 3-14) - (Eq. 3-15). It is also feasible to iterate over several trial values of  $\alpha$  and  $\beta$ .

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  $\sigma_h$  are typically chosen – rather arbitrarily – to be some multiple  $S_\sigma$  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  $\sigma_h$  for all RBF units is usually considered. Sometimes, instead of using just one value for all RBF's, each RBF unit's deviation  $\sigma_h$  is individually set to the distance to its  $N_\sigma \ll N$  nearest neighbors. Hence, deviations  $\sigma_h$  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  $N_\sigma$  is somewhat arbitrary. RBF networks with individual radial deviations  $\sigma_h$  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  $\sigma_h$  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  $S_\sigma$  and  $N_\sigma$ , 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  $W_{10}, \dots, W_{H0}$  ( $\sigma_1, \dots, \sigma_H$ , to be precise) in (Eqs. 3-17 and 3-18) 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, [18] and [21]. 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.

## 21.2. Kriging\*

Kriging is named after D. G. Krige [24], 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 [25] and Bakker [26].

The basic postulate of this formulation [25] is:

$$y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}), \quad (21-21)$$

where  $y$  is the unknown function of interest,  $f(\mathbf{x})$  is a known polynomial, the trend model, and  $Z(\mathbf{x})$  is the stochastic component with mean zero and covariance:

$$Cov[Z(\mathbf{x}^i), Z(\mathbf{x}^j)] = \sigma^2 \mathbf{R}([R(\mathbf{x}^i, \mathbf{x}^j)]). \quad (21-22)$$

With  $L$  the number of sampling points,  $\mathbf{R}$  is the  $L \times L$  correlation matrix with  $R(\mathbf{x}^i, \mathbf{x}^j)$  the correlation function between data points  $\mathbf{x}^i$  and  $\mathbf{x}^j$ .  $\mathbf{R}$  is symmetric positive definite with unit diagonal.

Two commonly applied correlation functions used are:

1. Exponential:  $R = \prod_{k=1}^n \exp(-\Theta_k |d_k|)$  and
2. Gaussian:  $R = \prod_{k=1}^n \exp(-\Theta_k d_k^2)$ .

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  $\Theta$ -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:

$$\hat{\mathbf{y}} = \hat{\beta} + \mathbf{r}^T(\mathbf{x}) \mathbf{R}^{-1}(\mathbf{y} - \mathbf{f} \hat{\beta}) \quad (21-23)$$

where  $\mathbf{r}^T(\mathbf{x})$  is the correlation vector (length  $L$ ) between a prediction point  $\mathbf{x}$  and the  $L$  sampling points,  $\mathbf{y}$  represents the responses at the  $L$  points and  $\mathbf{f}$  is the trend model, a  $L$ -vector of basis functions (ones, if  $f(\mathbf{x})$  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  $\mathbf{r}$  and scalar  $\hat{\beta}$  are given by:

$$\mathbf{r}^T(\mathbf{x}) = [\mathbf{R}(\mathbf{x}, \mathbf{x}^1), \mathbf{R}(\mathbf{x}, \mathbf{x}^2), \dots, \mathbf{R}(\mathbf{x}, \mathbf{x}^L)]^T$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{f}^T \mathbf{R}^{-1} \mathbf{f})^{-1} \mathbf{f}^T \mathbf{R}^{-1} \mathbf{y}.$$

The estimate of variance from the underlying global model is:

$$\hat{\sigma}^2 = \frac{1}{L} (\mathbf{y} - \mathbf{f} \hat{\boldsymbol{\beta}})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{f} \hat{\boldsymbol{\beta}}).$$

The maximum likelihood estimates for  $\Theta_k$ ,  $k = 1, \dots, n$  can be found by solving the following constrained maximization problem:

$$\max \Phi(\Theta) = -\frac{1}{2} [L \ln(\hat{\sigma}^2) + \ln |\mathbf{R}|], \text{ subject to } \Theta_k > 0 \wedge k.$$

where both  $\hat{\sigma}$  and  $|\mathbf{R}|$  are functions of  $\Theta$ . This is the same as minimizing

$$\hat{\sigma}^2 |\mathbf{R}|^{(1/n)}, \text{ subject to } \Theta > 0.$$

This optimization problem is solved using the real-coded genetic algorithm (Section 22.8). A small constant number is adaptively added to the diagonal of matrix  $\mathbf{R}$  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.

### 21.3. 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  $\mathbf{x}_i (i=1, 2, \dots, N)$  and their corresponding response values  $y_i$ ,  $\varepsilon$ -SVR attempts to find a function  $\hat{f}(\mathbf{x})$  that has at the most  $\varepsilon$  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:

$$\hat{f}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{21-24}$$

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:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & |\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i| \leq \varepsilon \quad \forall i \in [1, N] \end{aligned} \quad (21-25)$$

To ensure the feasibility of the optimization, two slack variables  $\xi$  and  $\xi^*$  are introduced:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \\ \text{s.t.} \quad & -\langle \mathbf{w}, \mathbf{x}_i \rangle - b + y_i \leq \varepsilon + \xi_i \quad \forall i \in [1, N] \\ & \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned} \quad (21-26)$$

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:

$$\begin{aligned} \max_{\alpha, \alpha_i^*} \quad & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \varepsilon \sum_{i=1}^N (\alpha_i + \alpha_i^*) + \sum_{i=1}^N y_i (\alpha_i - \alpha_i^*) \\ \text{s.t.} \quad & \sum_{i=1}^N (\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i \in [0, C] \quad \forall i \in [1, N] \end{aligned} \quad (21-27)$$

where  $\alpha_i$  and  $\alpha_i^*$  are Lagrange multipliers of the original problem that become the optimization variables for the dual formulation,  $\varepsilon$  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  $\mathbf{w}$  eliminated in the dual formulation can be written as:

$$\mathbf{w} = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \mathbf{x}_i \quad (21-28)$$

The support vector regression expression can therefore be written as:

$$f(\mathbf{x}) = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \langle \mathbf{x}_i, \mathbf{x} \rangle + b \quad (21-29)$$

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  $K$ :

$$\begin{aligned} \max_{\alpha, \alpha_i^*} \quad & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon \sum_{i=1}^N (\alpha_i + \alpha_i^*) + \sum_{i=1}^N y_i (\alpha_i - \alpha_i^*) \\ \text{s.t.} \quad & \sum_{i=1}^N (\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i \in [0, C] \quad \forall i \in [1, N] \end{aligned} \quad (21-30)$$

The optimization is solved using sequential minimal optimization (SMO) [34]. The final SVR expression is:

$$f(\mathbf{x}) = \sum_{i=1}^N (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x}) + b \quad (21-311)$$

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:

$$K(\mathbf{x}_i, \mathbf{x}) = (1 + \langle \mathbf{x}_i, \mathbf{x} \rangle)^m$$

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,  $\sigma$  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  $\varepsilon$  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 8.1.5.

## 21.4. 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 [27][28][29].

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 22.5. FF networks have been used for sequential optimization [29] and can be updated during the process. A more recent study [30] 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 [30] 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 [25], there is evidence of Kriging having fitting problems with certain types of experimental designs [31]. Kriging is very sensitive to noise, since it interpolates the data [32]. The authors of this manual have also experienced fitting problems with non-smooth surfaces ( $Z(x)$  observed to peak at data points) in some cases, apparently due to large values of  $\Theta$  that may be due to local optima of the maximum likelihood function. The model construction can be very time consuming [32] (also experienced with LS-OPT). Furthermore, the slight global altering of the Kriging surface due to local updating has also been observed [29]. 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 [29] 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.

## 21.5. REFERENCES

- [3] Daberkow, D.D., Mavris, D.N. An investigation of metamodeling techniques for complex systems design. *Symposium on Multidisciplinary Analysis and Design*, Atlanta, October 2002.
- [4] Redhe, M., Nilsson, L. Using space mapping and surrogate models to optimize vehicle crashworthiness design, *AIAA Paper 2002-5607. 9<sup>th</sup> AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, September 4-6, 2003.
- [5] Bishop, C.M. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [6] 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.

- [7] 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.
- [8] 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.
- [9] Hoerl, A.H., Kennard, R.W., Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3), pp. 55-67, 1970.
- [10] Tikhonov, A.N., Arsenin, V.Y., *Solutions of Ill-Posed Problems*, Winston: Washington, 1977.
- [11] 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.
- [12] Park, J., Sandberg, I.W. Approximation and radial basis function networks. *Neural Computation*, 5(2), pp. 305-316, 1993.
- [13] Wahba, G. *Spline Models for Observational Data*. Volume 59 of Regional Conference Series in Applied Mathematics. SIAM Press, Philadelphia, 1990.
- [14] Akaike, H. Statistical predictor identification. *Ann.Inst.Statist.Math.*, 22, pp. 203-217, 1970.
- [15] 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.
- [16] Lawrence, S.C., Lee Giles, Ah Chung Tsoi. What size neural network gives optimal generalization? Convergence Properties of Backpropagation. *Technical Report UMIACS-TR-96-22 and CS-TR-3617*, University of Maryland, 1996.
- [17] Sjöberg, J., Ljung, L. Overtraining, regularization, and searching for minimum in neural networks. Preprints of the 4<sup>th</sup> IFAC Int. Symp. on Adaptive Systems in Control and Signal Processing, p. 669, July 1992.
- [18] MacKay, D. J. C. Bayesian interpolation. *Neural Computation*, 4(3), pp. 415-447, 1992.
- [19] 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.
- [20] 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.
- [21] Cohn, D. Neural network exploration using optimal experiment design, *Neural Networks*, (9)6, pp. 1071-1083, 1996.
- [22] Fedorova, N.N. Personal communication, 2004.
- [23] 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.
- [24] 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.
- [25] Simpson, T.W. *A Concept Exploration Method for Product Family Design*. Ph.D. Thesis, Georgia Institute of Technology, 1998.
- [26] Bakker, T.M.D. *Design Optimization with Kriging Models*. WBBM Report Series 47, Ph.D. thesis, Delft University Press, 2000.
- [27] 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.

- [28] Stander, N., Reichert, R., Frank, T. 2000: Optimization of nonlinear dynamic problems using successive linear approximations. *AIAA Paper 2000-4798*.
- [29] 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 4<sup>th</sup> European LS-DYNA Conference*, Ulm, Germany, May 22-23, 2003. (Also [www.lstc.com](http://www.lstc.com)).
- [30] Stander, N. Goel, T. Metamodel sensitivity to sequential sampling strategies in crashworthiness design. *Proceedings of the 12<sup>th</sup> AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia, Canada, Sep 10-12, 2008. *Submitted*.
- [31] 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.
- [32] Jin, R., Chen, W. and Simpson, T.W. Comparative studies of metamodeling techniques under multiple modeling criteria, *AIAA Paper*, AIAA-2000-4801.
- [33] 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).
- [34] 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.
- [32] Vapnik, V. "Statistical learning theory", 1998.
- [33] Smola, A. J., & Schölkopf, B. A tutorial on support vector regression. *Statistics and computing*, 14(3), 199-222, 2004.
- [34] Platt, J. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Technical Report MSR-TR-98-14, Microsoft Research, 1998.
- [35] Geisser, S. *Predictive Inference*. New York, NY: Chapman and Hall, 1993.

# 22. Optimization

## 22.1. 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

$$\begin{aligned} & \min f(\mathbf{x}) \\ & \text{subject to :} \\ & g_j(\mathbf{x}) \leq 0; j = 1, 2, \dots, m \\ & h_k(\mathbf{x}) = 0; k = 1, 2, \dots, l. \end{aligned} \tag{22-1}$$

where  $f$ ,  $g$  and  $h$  are functions of independent variables  $x_1, x_2, x_3, \dots, x_n$ . 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  $\mathbf{x}$  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.

$$h_k(\mathbf{x}) = 0 \approx 0 \leq h_k(\mathbf{x}) \leq 0 \tag{22-2}$$

Equations (2.1) then become

$$\begin{aligned} & \min f(\mathbf{x}) \\ & \text{subject to :} \\ & g_j(\mathbf{x}) \leq 0; j = 1, 2, \dots, m. \end{aligned} \tag{22-3}$$

The necessary conditions for the solution  $x^*$  to Eq. (2.3) are the Karush-Kuhn-Tucker optimality conditions:

$$\begin{aligned}
\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^T \nabla g(\mathbf{x}^*) &= 0 \\
\boldsymbol{\lambda}^T g(\mathbf{x}^*) &= 0 \\
g(\mathbf{x}^*) &\leq 0 \\
\boldsymbol{\lambda} &\geq 0.
\end{aligned}
\tag{22-4}$$

These conditions are derived by differentiating the Lagrangian function of the constrained minimization problem

$$L(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T g(\mathbf{x}). \tag{22-5}$$

and applying the conditions

$$\nabla^T f \partial \mathbf{x}^* \geq 0 \text{ (optimality)} \tag{22-6}$$

and

$$\nabla^T \bar{\mathbf{g}} \partial \mathbf{x}^* \leq 0 \text{ (feasibility)} \tag{22-7}$$

to a perturbation  $\partial \mathbf{x}^*$ .  $\lambda_j$  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,  $\nabla^2 f(\mathbf{x}^*) + \boldsymbol{\lambda}^T \nabla^2 \bar{\mathbf{g}}(\mathbf{x}^*)$  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.

## 22.2. 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:

$$L_j \leq g_j(\mathbf{x}) \leq U_j; j = 1, 2, \dots, m. \tag{22-8}$$

This inequality represents two constraints:

$$L_j \leq g_j(\mathbf{x}); j = 1, 2, \dots, m. \tag{22-9}$$

$$g_j(\mathbf{x}) \leq U_j; j = 1, 2, \dots, m.$$

which, when normalized, become:

$$1 \leq \frac{g_j(\mathbf{x})}{L_j}; j = 1, 2, \dots, m. \quad (22-10)$$

$$\frac{g_j(\mathbf{x})}{U_j} \leq 1; j = 1, 2, \dots, m.$$

A feature is provided in the GUI to automatically switch on constraint scaling using a single check box. As in Equation (22-10), 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  $[\mathbf{x}_L; \mathbf{x}_U]$  to  $[0; 1]$ , where  $\mathbf{x}_L$  is the lower and  $\mathbf{x}_U$  the upper bound. The formula

$$\xi_i = \frac{x_i - x_{iL}}{x_{iU} - x_{iL}}. \quad (22-11)$$

is used to transform each variable  $x_i$  to a normalized variable,  $\xi_i$ .

### 22.3. 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.

## **22.4. 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 21.

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.

## **22.5. 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 4.7.

### 22.5.1. 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.

### 22.5.2. 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.  $10 \times 30$  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 22.5.3).

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.

### 22.5.3. Sequential strategy with domain reduction

This approach is the same as that in 22.5.2 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 22.5.2 *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 22.6). 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 22.6 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 22.6). 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 Section 23.3).



### 22.5.4. 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.

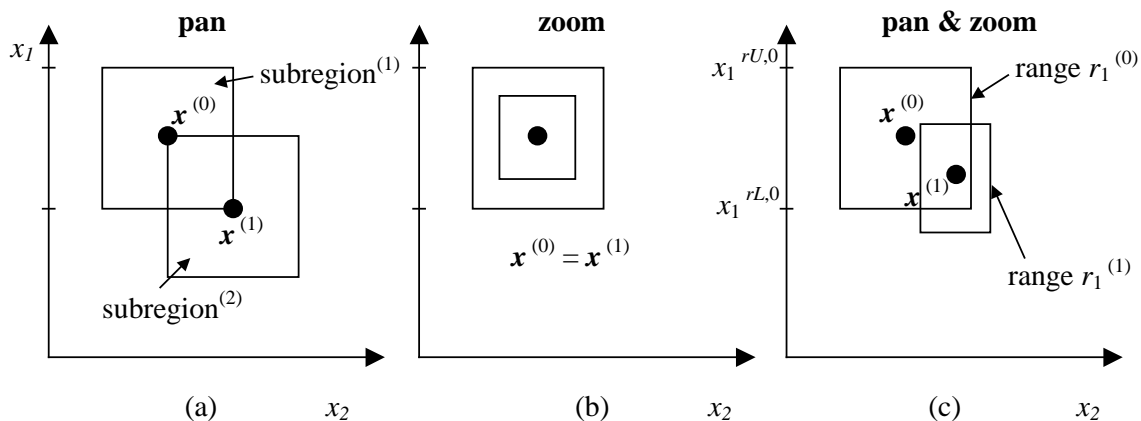
The Single Stage approach is intended for users who want to create a globally explorable model and have a fixed budget (e.g. 1000 runs). A very similar globally explorable design model can also be created with the Sequential strategy (without Domain Reduction) but an advantage of Sequential methods is 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.

## 22.6. 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 22-1 shows the possible adaptation of the subregion.



**Figure 22-1: 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

$$x_i^{rL,0} = x_i^{(0)} - 0.5r_i^{(0)} \text{ and } x_i^{rU,0} = x_i^{(0)} + 0.5r_i^{(0)}; \quad i = 1, \dots, n \quad (22-12)$$

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  $\gamma$  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

$$c_i^{(k)} = d_i^{(k)} d_i^{(k-1)} \quad (22-13)$$

where

$$d_i^{(k)} = 2\Delta x_i^{(k)} / r_i^{(k)}; \quad \Delta x_i^{(k)} = x_i^{(k)} - x_i^{(k-1)}; \quad d_i^{(k)} \in [-1;1] \quad (22-14)$$

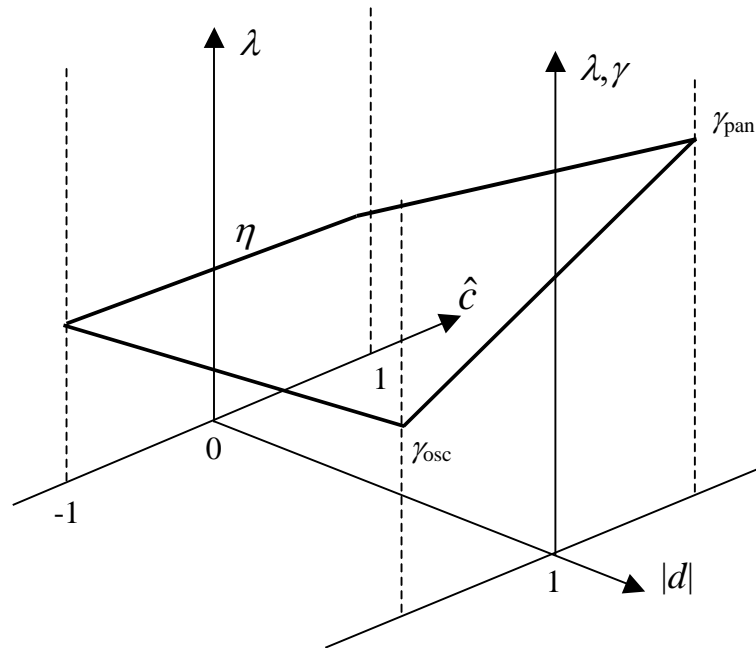
The oscillation indicator (purposely omitting indices  $i$  and  $k$ ) is normalized as  $\hat{c}$  where

$$\hat{c} = \sqrt{|c|} \text{sign}(c).. \quad (22-15)$$

The contraction parameter  $\gamma$  is then calculated as

$$\gamma = 0.5(\gamma_{pan}(1 + \hat{c}) + \gamma_{osc}(1 - \hat{c})). \quad (22-16)$$

See Figure 22-2. The parameter  $\gamma_{osc}$  is typically 0.5-0.7 representing shrinkage to dampen oscillation, whereas  $\gamma_{pan}$  represents the pure panning case and therefore unity is typically chosen.



**Figure 22-2:** The sub-region contraction rate  $\lambda$  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 22-1(a)). If the optimum point coincides with the previous one, the subregion is stationary, but reduces its size (*zooming*) (Figure 22-1(b)). Both panning and zooming may occur if there is partial movement (Figure 22-1(c)). The range  $r_i^{(k+1)}$  for the new subregion in the  $(k + 1)$ -th iteration is then determined by:

$$r_i^{(k+1)} = \lambda_i r_i^{(k)}; i = 1, \dots, n; k = 0, \dots, niter \quad (22-17)$$

where  $\lambda_i$  represents the *contraction rate* for each design variable. To determine  $\lambda_i$ ,  $d_i^{(k)}$  is incorporated by scaling according to a *zoom parameter*  $\eta$  that represents pure zooming and the contraction parameter  $\gamma$  to yield the contraction rate

$$\lambda_i = \eta + |d_i^{(k)}|(\gamma - \eta) \quad (22-18)$$

for each variable (see Figure 22-2).

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 4.8.2 for the setting of parameters in the iterative Sequential Response Surface Method.

## 22.7. 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  $\mathbf{x}(t)$  at time  $t$  is taken to be the function  $f(\mathbf{x})$  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 10.4.1). 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 10.4.1) should therefore be adequate. The termination criteria are also listed in Section 11.1.

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.

## 22.8. 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.

### 22.8.1. 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.

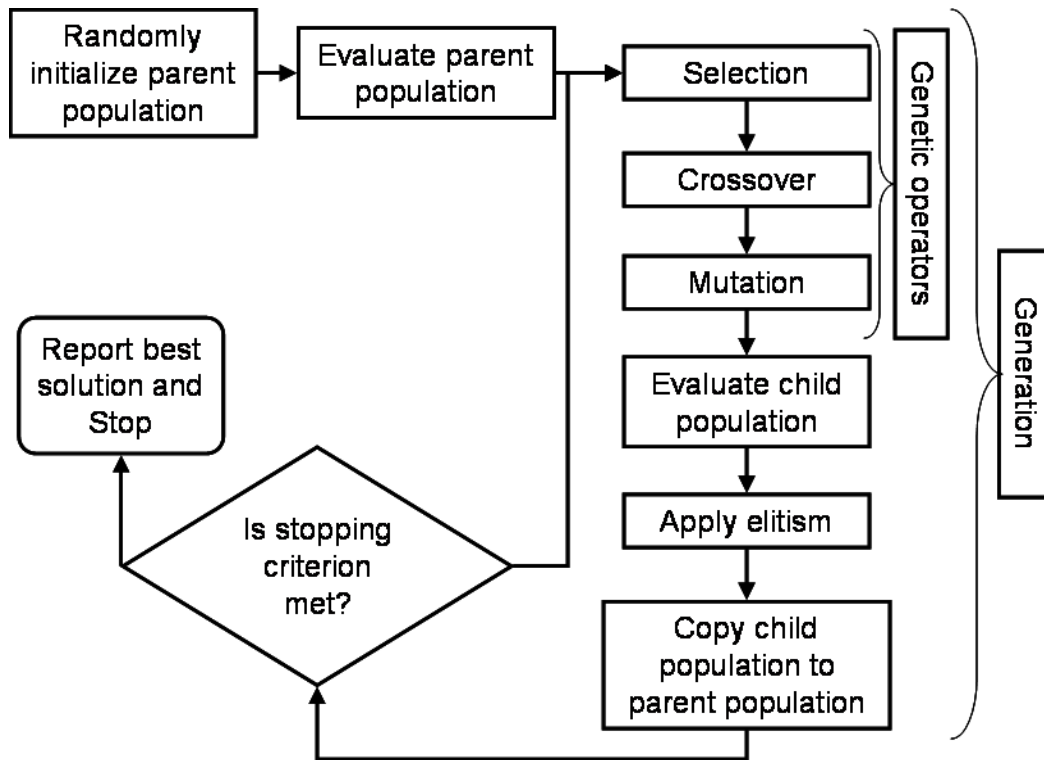
### 22.8.2. Encoding

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  $(x_u - x_l) / (2^p - 1)$ . 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.

### 22.8.3. Algorithm

The steps in a simple genetic algorithm are illustrated with the help of Figure 22-3.



*Figure 22-3: 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, ‘ $N_{\text{tourn}}$ ’ ( $N_{\text{tourn}}$  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  $N_{\text{tourn}} = 2$ . Increasing the tournament size ‘ $N_{\text{tourn}}$ ’ 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  $N_{\text{pop}}$  individuals, the selection probability of the  $i^{\text{th}}$  individual is

$$P_i = F_i / \sum_{j=1}^{N_{\text{pop}}} F_j \quad (22-19)$$

where  $F_i$  is the fitness of  $i^{\text{th}}$  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  $i^{\text{th}}$  individual in the mating pool can be estimated as

$$N_i = F_i / \bar{F}; \bar{F} = \frac{1}{N_{\text{pop}}} \sum_{j=1}^{N_{\text{pop}}} F_j. \quad (22-20)$$

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  $N_{\text{pop}}$  equi-spaced pointers. This operator also has a high selection pressure.

## Crossover

Crossover is the main exploration operator of genetic search. In this operator,  $\mu$  randomly selected parents mate with a probability ( $P_c$ : crossover probability) to create  $\lambda$  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)  $(\mu, \lambda)$  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  $\eta_c$  such that large value of  $\eta_c$  creates children near parents and small value of  $\eta_c$  creates children far from parents. Deb and Beyer [13] showed that SBX possesses self-adaptation capabilities.

- Blend real crossover (BLX- $\alpha$ )

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,  $\alpha$  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.

## 22.9. 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.

### 22.9.1. 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  $\mathbf{x}^{(2)}$  ( $\mathbf{x}^{(1)} \succ \mathbf{x}^{(2)}$ ), if either of the following three conditions is true.

1.  $\mathbf{x}^{(1)}$  is feasible and  $\mathbf{x}^{(2)}$  is infeasible.
2. Both  $\mathbf{x}^{(1)}$  and  $\mathbf{x}^{(2)}$  are infeasible but  $\mathbf{x}^{(2)}$  is more infeasible compared to  $\mathbf{x}^{(1)}$ .
3. When both  $\mathbf{x}^{(1)}$  and  $\mathbf{x}^{(2)}$  are feasible,  $\mathbf{x}^{(1)}$  dominates  $\mathbf{x}^{(2)}$  ( $\mathbf{x}^{(1)} \succ \mathbf{x}^{(2)}$ ) if following two conditions are satisfied
  - $\mathbf{x}^{(1)}$  is no worse than  $\mathbf{x}^{(2)}$  in ‘all’ objectives, i.e.  $(f_j(\mathbf{x}^{(1)}) \leq f_j(\mathbf{x}^{(2)})) \quad j \in [1, 2, \dots, M]$ .
  - $\mathbf{x}^{(1)}$  is strictly better than  $\mathbf{x}^{(2)}$  in ‘at least one’ objective, i.e.,  $(f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)})), \wedge j \in [1, 2, \dots, M]$ .

If neither of the two solutions dominates the other, both solutions are non-dominated with respect to each other. An individual  $\mathbf{s}$  is considered non-dominated with respect to a set of solutions  $S$ , if no solution in  $S$  dominates  $\mathbf{s}$ .

### 22.9.2. 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.

### 22.9.3. Pareto optimal set

The set of all Pareto optimal solutions is the Pareto optimal set for the given problem.

### 22.9.4. 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.

### 22.9.5. Ranking

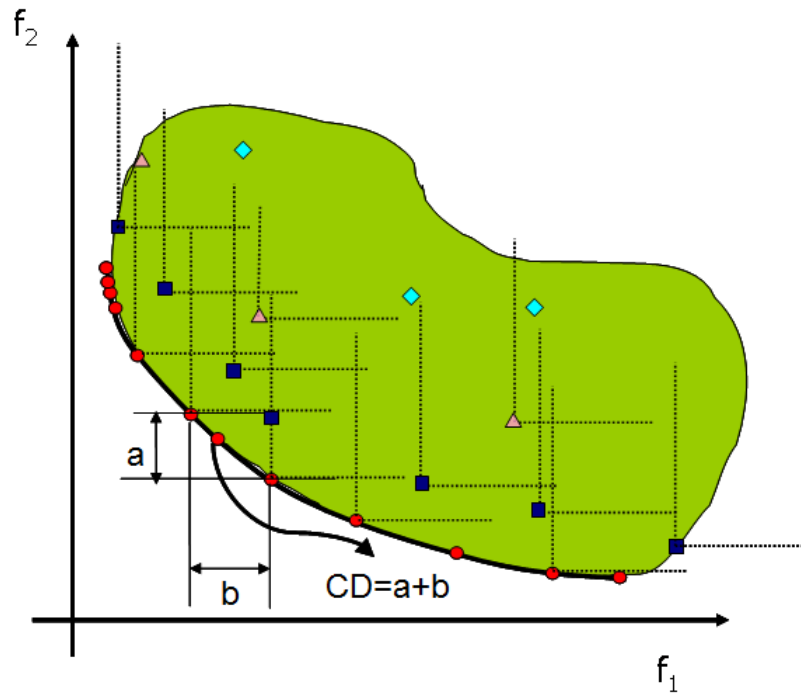
Most MOGA search methods assign rank to different individuals based on non-domination criterion. This ranking is used to govern the search process. A rank of one is considered the best rank and low fitness individuals are assigned low ranks (large values of rank are low). Different individuals in a population are assigned rank as follows:

1. Initialize  $rnk = 1$ . Define a set of individuals  $\mathbf{S}$ , same as the population.
2. Run a non-domination check on all individuals in  $\mathbf{S}$ .
3. All non-dominated individuals are assigned rank =  $rnk$ .
4.  $rnk = rnk + 1$ .
5. Remove all non-dominated individuals from  $\mathbf{S}$ .
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 22-4. 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 22-4: Illustration of non-domination criterion, Pareto optimal set, and Pareto optimal front.*

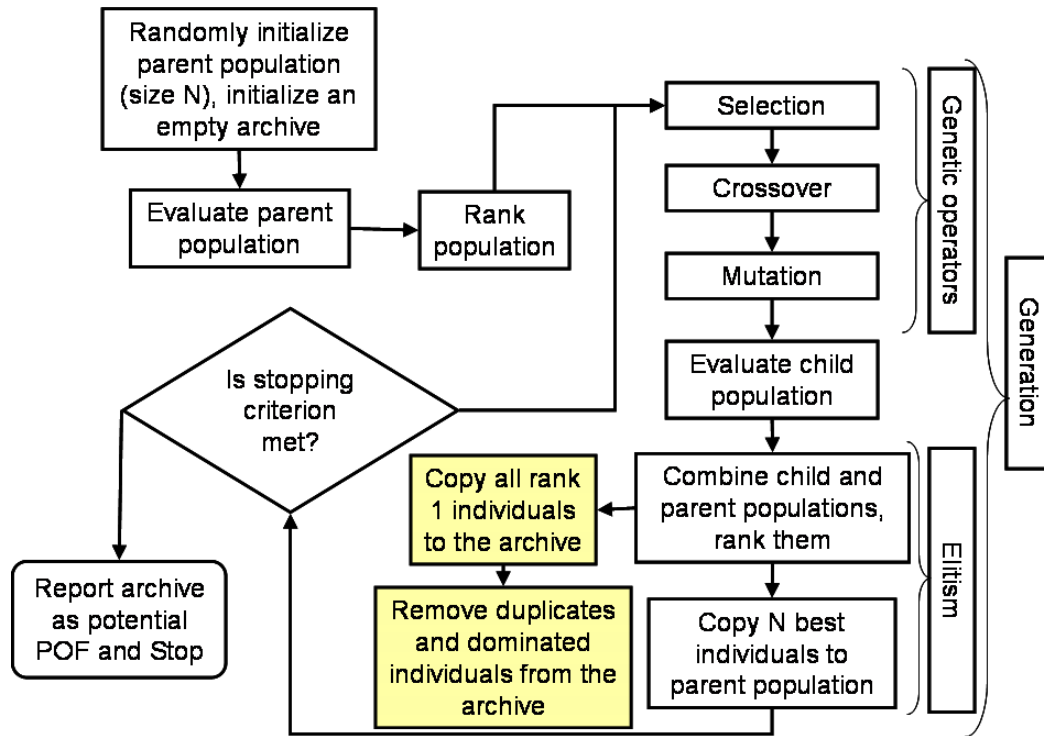
### 22.9.6. Convergence vs. diversity

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.

### 22.9.7. 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.



**Figure 22-5: 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 22-5, 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.

### 22.9.8. Elitism in NSGA-II

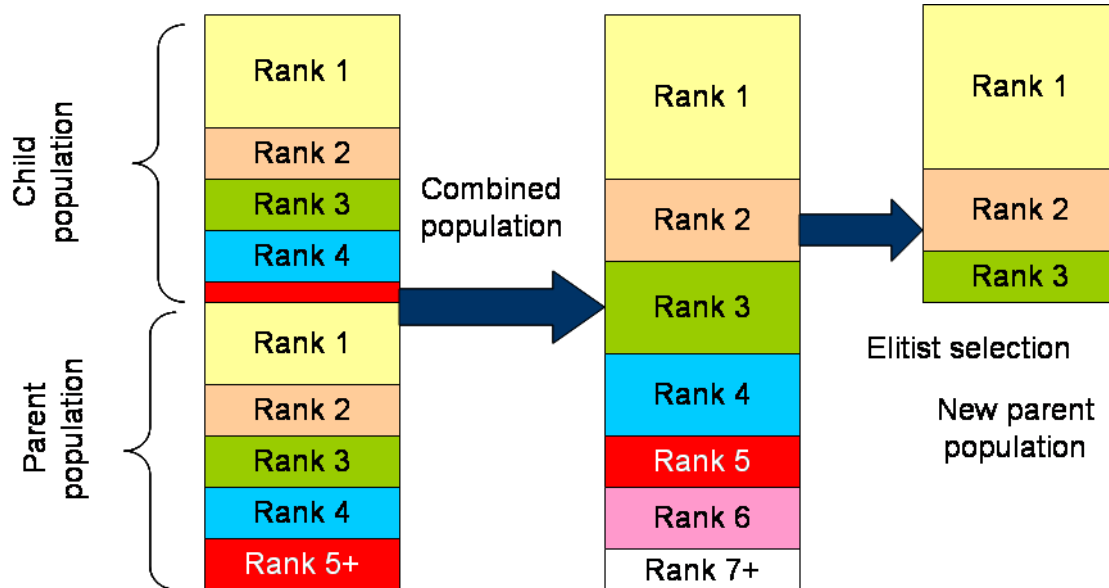


Figure 22-6: Elitism in NSGA-II.

Elitism is applied to preserve the best individuals. The mechanism used by NSGA-II algorithm for elitism is illustrated in Figure 22-6. After combining the child and parent populations, there are  $2N_{pop}$  individuals. This combined pool of members is ranked using non-dominance 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

### 22.9.9. 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 22-4, 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.

## 22.10. 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  $\kappa_B$  is the Boltzmann's constant.

### 22.10.1. 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  $\mathbf{x}^{(0)} \in \mathbf{X}$  and corresponding energy  $E^{(0)} = E(\mathbf{x})$ . 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  $k = 0$ .  $\mathbf{X}^{(0)} = \{\mathbf{x}^{(0)}\}$ .

#### Step 2: Sampling

A new point  $\mathbf{x}' \in \mathbf{X}$  is sampled using the candidate distribution  $D(\mathbf{X}^{(k)})$ , and set  $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} \cup \{\mathbf{x}'\}$ , and corresponding energy is calculated  $E' = E(\mathbf{x}')$ .

**Step 3: Check acceptance**

Sample a uniform random number  $\zeta$  in  $[0, 1]$  and set

$$\mathbf{x}^{(k+1)} = \mathbf{x}' \text{ if } \zeta \leq A(\mathbf{E}', \mathbf{E}^{(k)}, T^{(k)}) \text{ or}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \text{ otherwise.}$$

where  $A(\mathbf{x})$  is the acceptance function that determines if the new state is accepted.

**Step 4: Temperature update**

Apply the cooling schedule to the temperature:  $T^{(k+1)} = C(\mathbf{X}^{(k+1)}, T^{(k)})$ .

**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.

**22.10.2. 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.

**22.10.3. 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*,  $\mathbf{x}'$ , is usually selected randomly among all the neighbors of the current solution,  $\mathbf{x}$ , 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  $i^{\text{th}}$  design variable be bounded as,  $x_i \in [A_i, B_i]$ . Then the new sample is given by

$$x_i' = x_i^{(k)} + v_i(B_i - A_i),$$

where  $v_i$  is estimated as follows.

$$v_i = \text{sgn}(u - 0.5) T_{p,i}^{(k)} \left[ (1 + 1/T_{p,i}^{(k)})^{|2u-1|} - 1 \right], u \in U[0,1].$$

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.

#### 22.10.4. 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  $v_T$  iterations:

$$T := T / \mu_T$$



where  $\mu_T$  is slightly greater than 1 (e.g.  $\mu_T = 1.001$ ). The value of  $v_T$  should be large enough, so that 'thermal equilibrium' is achieved before reducing the temperature. A rule of thumb is to take  $v_T$  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'.

### 22.10.5. 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.

### 22.10.6. 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  $i^{\text{th}}$  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.

## 22.11. 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.

## 22.12. 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:

### 22.12.1. 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 22-7. A serious limitation of this plot type is its inability to simultaneously show more than four dimensions.

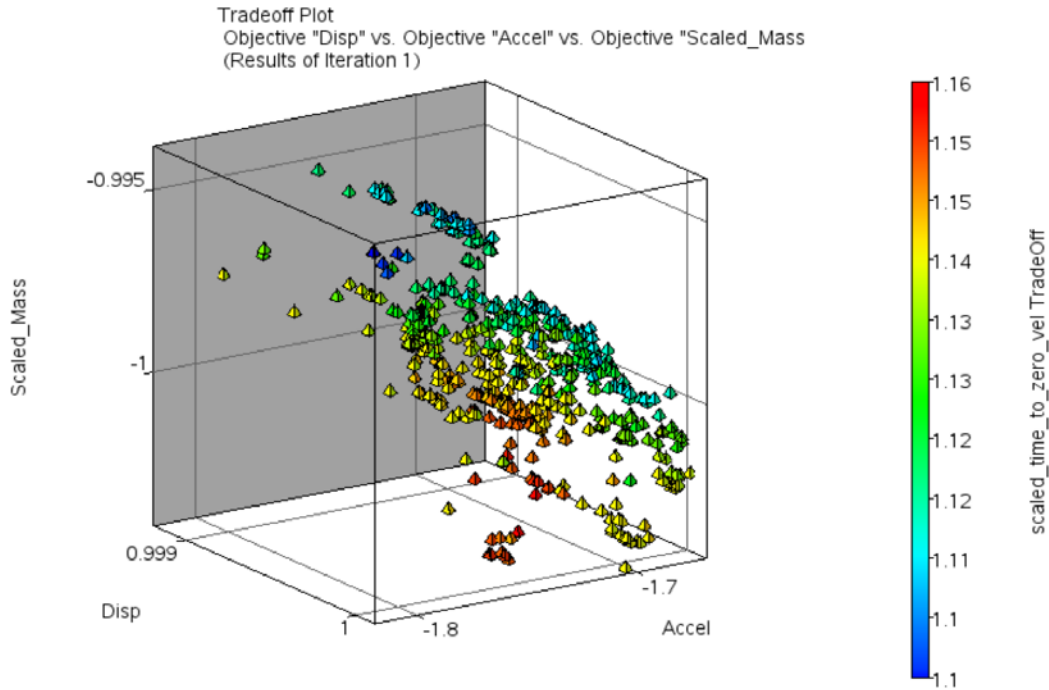


Figure 22-7: Trade-off plot shows all four objectives of Pareto optimal solutions.

### 22.12.2. 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:

$$\text{Minimize } F(x) = [f_1(x), f_2(x), \dots, f_n(x)] \text{ where } x = [x_1, x_2, \dots, x_p]$$

subject to

$$g_j(x) \leq 0; \quad j = 1, 2, \dots, m \quad \text{inequality constraints}$$

$$h_k(x) = 0; \quad k = 1, 2, \dots, l \quad \text{equality constraints}$$

$$x_i^l \leq x_i \leq x_i^u; \quad i = 1, 2, \dots, p \quad \text{side constraints}$$

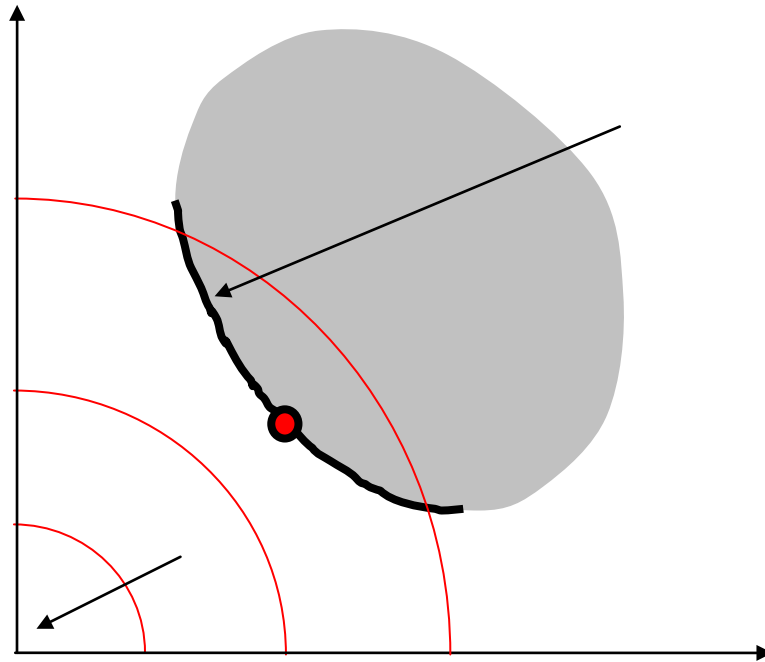
HRV can be seen as a conversion of the multi-objective optimization problem to a single objective optimization problem using the objective:

$$\frac{\sum_{i=1}^s W_i \tilde{F}_i^2 + \sum_{i=s+1}^n W_i \tilde{F}_i^2}{s}$$

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 22-8). The case where  $n$  is an odd number is discussed below.



**Figure 22-8: 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.

$$\tilde{F}_i = \frac{F_i - F_{i \min}}{F_{i \max} - F_{i \min}} \quad i = 1, \dots, n \quad \text{where } \tilde{F}_i \in [0, 1]$$

The coordinate  $[F_{1 \min}, F_{2 \min}, \dots, F_{n \min}]$  represents the *Utopian* point (see Figure 22-8), 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:

$$HRC = \sqrt{\frac{\sum_{i=1}^n \tilde{F}_i^2}{n}} \quad \text{where } HRC \in [0,1]$$

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:  $[F_1, F_2, F_3, \dots, F_s]$   $HRC1 = \sqrt{\frac{\sum_{i=1}^s \tilde{F}_i^2}{s}}$

Group 2:  $[F_{s+1}, F_{s+2}, F_{s+3}, \dots, F_n]$   $HRC2 = \sqrt{\frac{\sum_{i=s+1}^n \tilde{F}_i^2}{n-s}}$

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:

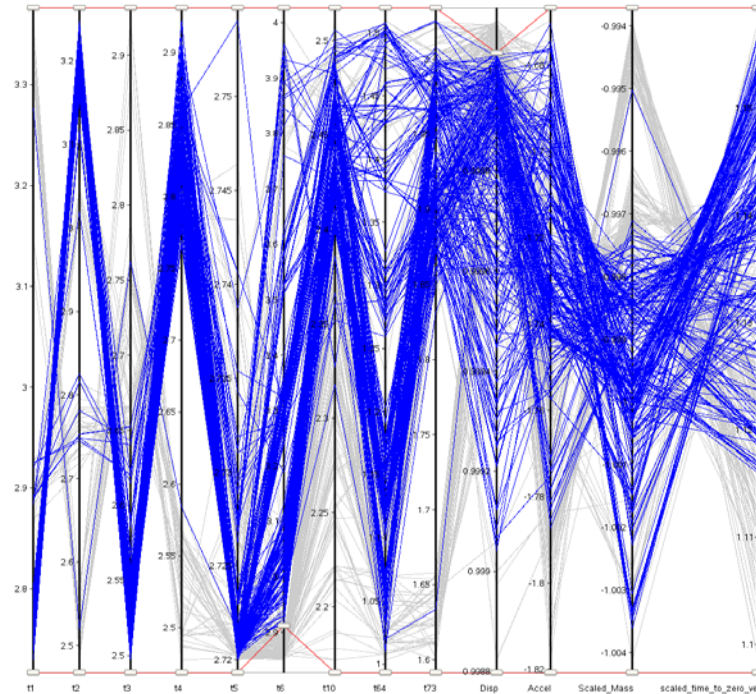
Group 1:  $[F_1, F_2, F_3, \dots, F_s]$   $HRCW1 = \sqrt{\frac{\sum_{i=1}^s W_i \tilde{F}_i^2}{s}}$

Group 2:  $[F_{s+1}, F_{s+2}, F_{s+3}, \dots, F_n]$   $HRCW2 = \sqrt{\frac{\sum_{i=s+1}^n W_i \tilde{F}_i^2}{s}}$

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.

### 22.12.3. 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 22-9. 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 22-9.



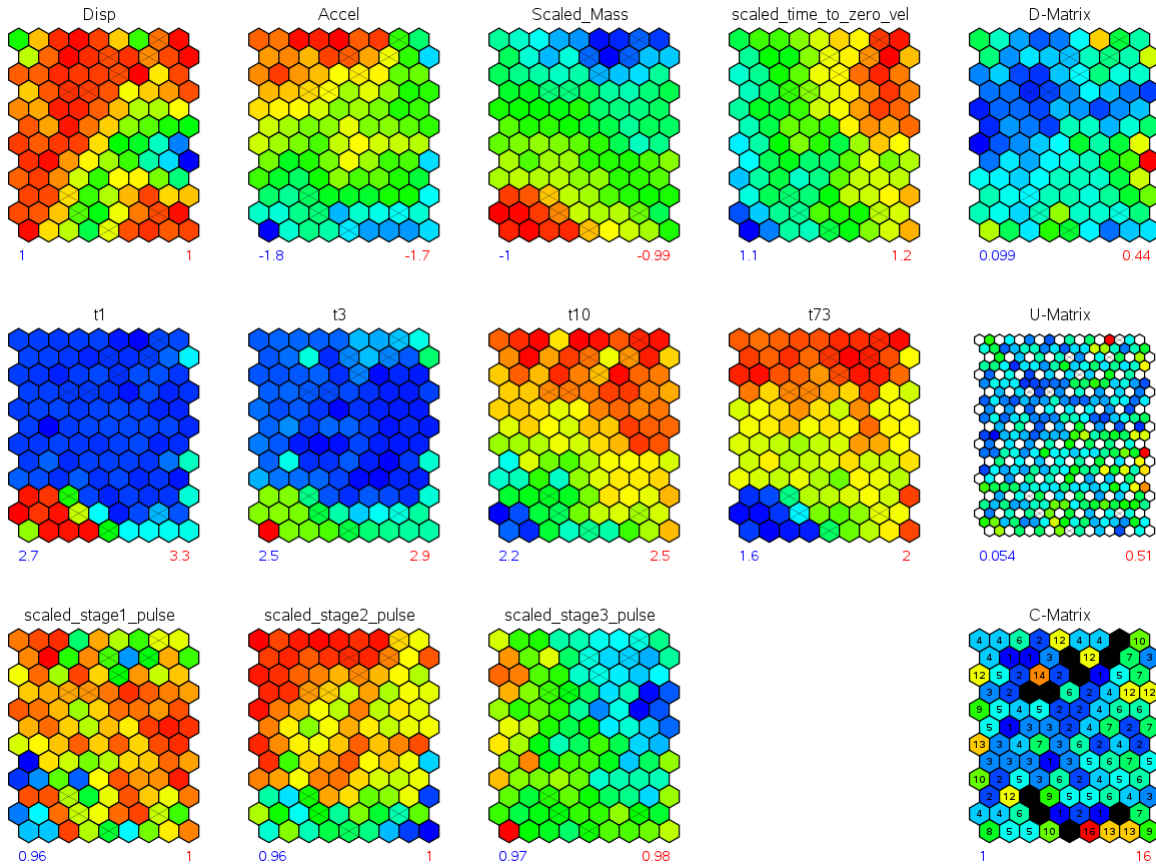
**Figure 22-9:** Parallel coordinate plot shows objectives and design variables of all points on the Pareto front.

#### 22.12.4. 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 22-10 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.



*Figure 22-10: Self organizing maps display design objectives, variables, and constraints on the Pareto front.*

## 22.13. 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.

### 22.13.1. 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.

### 22.13.2. 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.



### 22.13.3. 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,

$$\delta = \sum_{i=1}^N \frac{|d_i - \bar{d}|}{N}; \bar{d} = \frac{1}{N} \sum_{i=1}^N d_i.$$

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.

### 22.13.4. Min/Max of objectives

This represents the range of individual objectives. A wide range represents more choices for the designer.

### 22.13.5. 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  $\Delta$  generations (archive  $A_i$  and  $A_{i-\Delta}$ ) are compared and the following metrics are reported.  $\Delta$  is called generation interval.

### 22.13.6. 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.

### 22.13.7. 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.

### 22.13.8. 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.

### 22.13.9. Consolidation ratio

This represents the fraction of archive  $A_i$  that has evolved up to the  $i-\Delta^{\text{th}}$  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.

### 22.13.10. 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].

## 22.14. Discrete optimization

### 22.14.1. 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.

### 22.14.2. 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.

### 22.14.3. 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.

### 22.14.4. Discrete optimization algorithm: genetic algorithm

A GA (genetic algorithm, Section 22.8) 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.

### 22.14.5. 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:  $\sum \frac{|\text{constraint}_i - \text{Bound}_i|}{|\text{Bound}_i|}$  with the summation done over all the violated constraints.

### 22.14.6. 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 22.6), 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.

## 22.15. Summary of the optimization process

The following tasks may be identified in the process of an optimization cycle using response surfaces.

**Table 22-1: 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:

### 22.15.1. 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 21.1) and Radial Basis Function networks (Section 21.1.3) 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 22.5 on sequential strategies for optimization and design exploration.

## 22.16. 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 31<sup>st</sup> 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 12<sup>th</sup> 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 *10<sup>th</sup> 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 12<sup>th</sup> AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 10-12 September 2008, Victoria, British Columbia, Canada.
- [39] Kohonen T, *Self-Organizing Maps*, 3<sup>rd</sup> 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.



# 23. Applications of Optimization

## 23.1. 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 22.9, 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. (2.3) has been modified to represent the more general approach as applied in LS-OPT.

Minimize the function

$$p[f(\mathbf{x})] \tag{23-1}$$

subject to the inequality constraint functions

$$L_j \leq g_j(\mathbf{x}) \leq U_j; \quad j = 1, 2, \dots, m.$$

The preference function  $p$  can be formulated to incorporate target values of objectives.

Two methods for achieving this are given:

### 23.1.1. 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  $p[f(\mathbf{x})]$  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:

$$p = \sqrt{\sum_{i=1}^p W_i \left[ \frac{f_i(\mathbf{x}) - F_i}{\Gamma_i} \right]^2} \tag{23-2}$$

The symbols  $F_i$  represent the target values of the responses. A value  $\Gamma_i$  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.

### 23.1.2. Maximum distance

Another approach to target responses is by using the maximum distance to a target value

$$p = \max_i \left[ \frac{|f_i(\mathbf{x}) - F_i|}{|\Gamma_i|} \right]. \quad (23-3)$$

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

$$p = \left[ \sum_{i=1}^p \left( \frac{|f_i(\mathbf{x}) - F_i|}{|\Gamma_i|} \right)^r \right]^{1/r}. \quad (23-4)$$

with  $r = 2$  for the Euclidean metric and  $r \rightarrow \infty$  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:

$$\text{Minimize } e \quad (23-5)$$

subject to

$$\frac{F_i}{\Gamma_i} - (1 - \alpha_{jL})e \leq \frac{f_i(\mathbf{x})}{\Gamma_i} \leq \frac{F_i}{\Gamma_i} + (1 - \alpha_{jL})e; \quad i = 1, \dots, p, \quad j = 1, \dots, m$$

$$e \geq 0.$$

In the above equation,  $\Gamma_i$  is a normalization factor,  $e$  represents the constraint violation or target discrepancy and  $\alpha$  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.

## 23.2. 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, it 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 16.3) for the details of implementing an MDO problem. There is one crashworthiness-modal analysis case study in the examples chapter (Section 17.5).

### 23.3. System identification using nonlinear regression

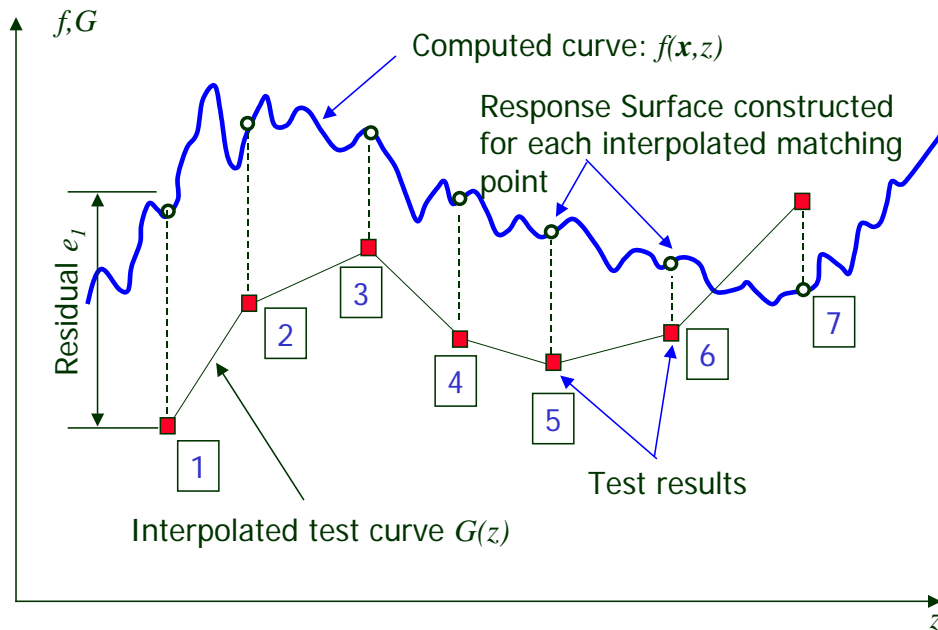
System identification is a general term used to describe the mathematical tools and algorithms that build dynamical models such as systems or processes from measured data. The methodology used in LS-OPT consists of a nonlinear regression procedure to optimize the parameters of a system or material. This procedure minimizes the errors with respect to given experimental results. Two formulations for system identification can be used. The first uses the mean squared error (MSE) as the minimization objective, while the second, the Min-Max formulation, uses the auxiliary problem formulation to minimize the maximum residual. The MSE approach is commonly used for system identification and has been automated using a single command. The two formulations are outlined below.

#### 23.3.1. Ordinate-based Curve Matching

Figure 23-1 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 23-1, seven regression points are shown. The residuals at these points are combined into a Mean Squared Error norm:

$$\varepsilon = \frac{1}{P} \sum_{p=1}^P W_p \left( \frac{f_p(\mathbf{x}) - G_p}{s_p} \right)^2 = \frac{1}{P} \sum_{p=1}^P W_p \left( \frac{e_p(\mathbf{x})}{s_p} \right)^2 \quad (23-6)$$



**Figure 23-1: Entities in Mean Squared Error formulation**

The MSE norm is based on a series of  $P$  regression points beginning at the start point  $z_I$  and terminating at the end point  $z_P$  (see Figure 23-1). The  $s_p$ ,  $p=1, \dots, P$  are residual scale factors and the  $W_p$ ,  $p=1, \dots, P$  are weights applied to the square of the scaled residual  $(f_p - G_p) / s_p$  at point  $p$ .

The application of optimization to system identification is demonstrated in Chapter 18.

### 23.3.2. Curve Mapping

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. 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.

The mathematical literature provides some ideas on curve matching approaches. Two commonly used metrics for curve matching are the *Hausdorff* [6] and *Fréchet* [7] 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:

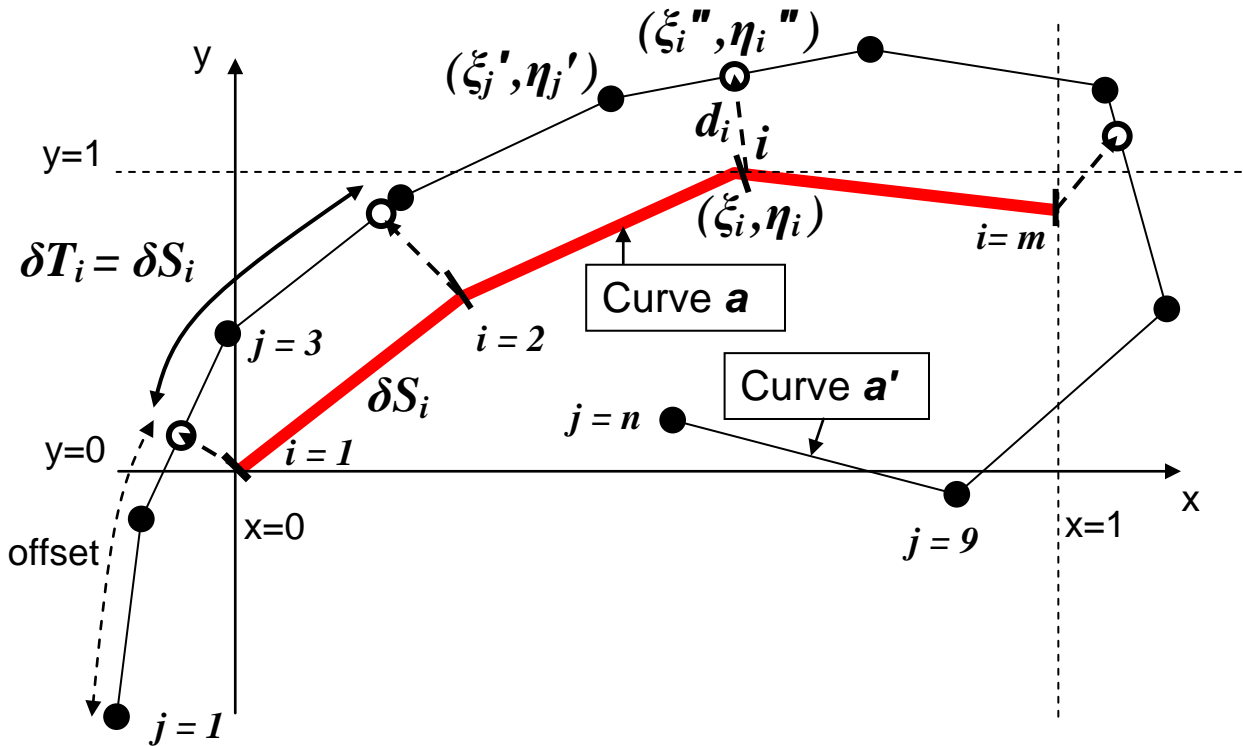
$$Fr(P, Q) = \inf_{\alpha, \beta} \max_{t \in [0,1]} \|P(\alpha(t)) - Q(\beta(t))\|$$

where  $P$  and  $Q$  are polygonal curves,  $t \in [0,1]$  represents a position on each curve. The parameters  $\alpha$  and  $\beta$  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  $\alpha$  and  $\beta$ . The Fréchet distance is the length of the shortest leash that is sufficient for traversing both curves in this manner.

In LS-OPT we map the points of the one curve onto the second curve and compute 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. See Reference [8].

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 23-2 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  $\mathbf{a}$  is in the  $[(0,0),(1,1)]$  range.



**Figure 23-2:** 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.

$$\xi_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \quad \eta_i = \frac{Y_i - Y_{\min}}{Y_{\max} - Y_{\min}}$$

$$X_{\min} = \min_k X_k; \quad X_{\max} = \max_k X_k; \quad Y_{\min} = \min_k Y_k; \quad Y_{\max} = \max_k Y_k$$

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.

$$\xi_j' = \frac{x_j - X_{\min}}{X_{\max} - X_{\min}} \quad \eta_j' = \frac{y_j - Y_{\min}}{Y_{\max} - Y_{\min}}$$

3. Compute  $S$ , the total polygon length of  $a$ . Also compute the individual segment lengths  $\delta S_i$ :

$$\delta S_i = \sqrt{(\xi_i - \xi_{i-1})^2 + (\eta_i - \eta_{i-1})^2}; \quad i = 2, 3, \dots, m$$

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  $S$ :

$$\tilde{s}_i = \delta S_i / S; \quad i = 2, 3, \dots, m$$

5. Compute  $T$ , the total polygon length of  $\mathbf{a}'$ .
6. If  $S > T$ , rename  $\mathbf{a}'$  to  $\mathbf{a}$  and  $\mathbf{a}$  to  $\mathbf{a}'$ . Hence  $\mathbf{a}$  will always be shorter than  $\mathbf{a}'$ .
7. Define an *offset* as a starting point of a curve section of total length  $S$  on curve  $\mathbf{a}'$ . The offset =  $\lambda_p$  will be varied over  $p = 1$  to  $P$  in order to “slide” Curve  $\mathbf{a}$  along Curve  $\mathbf{a}'$ .  $\lambda \in [0, T - S]$ . Assume  $P$  increments in this interval so that each increment has size  $\Delta\lambda = \frac{T - S}{P}$ .
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  $\mathbf{a}$  to curve  $\mathbf{a}'$ . A typical curve segment  $i$  on  $\mathbf{a}'$  which corresponds to a segment  $i$  on  $\mathbf{a}$  has length  $\delta T_i = \delta S_i$  (see Fig. 1). This creates a new set of point pairs  $\mathbf{a}''$ . *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  $\mathbf{a}$  and  $\mathbf{a}''$ . 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.)

$$v_i = \frac{d_i + d_{i-1}}{2} \times \tilde{s}_i; \quad v_1 = 0; \quad i = 2, 3, \dots, m;$$

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  $\mathbf{a}$  and  $\mathbf{a}'$ .

### 23.3.3. 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:



$$\text{Minimize } e, \quad (23-7)$$

subject to

$$\left| \frac{f_p(\mathbf{x}) - G_p}{s_p} \right| \leq e; \quad p = 1, \dots, P$$

$$e \geq 0.$$

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  $f(\mathbf{x})$  so that:

$$\left| \frac{f_p(\mathbf{x})}{G_p} - 1 \right| \leq e; \quad p = 1, \dots, P$$

$$e \geq 0.$$

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.

### 23.3.4. Nonlinear regression: Confidence intervals

Assume the nonlinear regression model:

$$G(t) = F(t, \mathbf{x}) + \varepsilon,$$

where the measured result  $G$  is approximated by  $F$  and  $\mathbf{x}$  is a vector of unknown parameters. The nonlinear least squares problem is obtained from the discretization:

$$\min_x \frac{1}{P} \sum_{p=1}^P (G_p - F_p(\mathbf{x}))^2,$$

is solved to obtain  $\mathbf{x}^*$ . The variance  $\sigma^2$  is estimated by

$$\sigma^2 = \frac{1}{P-n} \|G - F(\mathbf{x}^*)\|^2,$$

where  $F$  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:

$$\left( x_i : |x_i^* - x_i| \leq \sqrt{\hat{C}_{ii}} t_{p-n}^{\alpha/2} \right),$$

where

$$\hat{C} := \hat{\sigma}^2 \left( (\nabla \mathbf{F}(\mathbf{x}^*))^T (\nabla \mathbf{F}(\mathbf{x}^*)) \right)^{-1},$$

and  $t_{p-n}^{\alpha/2}$  is the Student  $t$ -distribution for  $\alpha$ .

$\nabla \mathbf{F}$  is the  $P \times n$  matrix obtained from the  $n$  derivatives of the  $P$  response functions representing  $P$  points at the optimum  $x$ . 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.

## 23.4. 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  $z_i, i = 1, \dots, n$  in addition to the *controlled* variables  $y_j, j = 1, \dots, m$ . 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  $\rho$  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  $\rho$ . 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 (2.1) becomes:

$$\min_y \left\{ \max_z f(\mathbf{y}, \mathbf{z}) \right\}; \quad \mathbf{y} \in \mathfrak{R}^p, \mathbf{z} \in \mathfrak{R}^q \quad (23-8)$$

subject to

$$g_j(\mathbf{y}, \mathbf{z}) \leq 0; \quad j = 1, 2, \dots, l.$$

The algorithm remains a minimization algorithm but with modified gradients:

$$\nabla_y^{\text{mod}} := \nabla \mathbf{y},$$

$$\nabla_z^{\text{mod}} := -\nabla \mathbf{z}.$$

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.  $t = t_{\text{mean}} + t_{\text{deviation}}$ , where  $t$  is the dependent variable.

## 23.5. 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 multiobjective, 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.
- [6] Rockafellar, R.T., Wets, R. J-B. *Variational Analysis*, Springer Verlag, 2005, p. 117
- [7] Alt, H. and Godau, M. Computing the Fréchet distance between two polygonal curves, *Internat. J. Comput. Geom.. Appl.*, 5:75-91, 1995.
- [8] Witowski, K. and Stander, N. Parameter Identification of Hysteretic Models using Partial Curve Mapping. *Proceedings of the 12<sup>th</sup> AIAA Aviation Technology, Integration and Operations (ATIO) Conference and 14<sup>th</sup> AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 17-19 September 2012, Indianapolis, Indiana, USA

# 24. Probabilistic Fundamentals

## 24.1. 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.

## 24.2. 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.

## 24.3. Basic computations

### 24.3.1. 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

$$s^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2.$$

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

$$c.o.v. = s/\bar{y}.$$

### 24.3.2. 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.

$$Cov(Y_1, Y_2) = E[(Y_1 - \mu_1)(Y_2 - \mu_2)],$$

$$Cov(Y_1, Y_2) = E[Y_1 Y_2] - E(Y_1)E(Y_2).$$

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.

### 24.3.3. 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  $\mu$  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

$$\frac{(n-1)s^2}{\chi_{\alpha/2, n-1}^2} < \sigma^2 < \frac{(n-1)s^2}{\chi_{1-\alpha/2, n-1}^2},$$

with  $\sigma^2$  the variance and  $\chi_{\alpha/2, n-1}^2$ ,  $\chi_{1-\alpha/2, n-1}^2$  the relevant critical values of the  $\chi^2$  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

$$\tanh \left[ \frac{1}{2} \ln \left( \frac{1+\rho}{1-\rho} \right) - \frac{t_{1-\alpha/2, N}}{\sqrt{N-3}} \right] \leq \rho \leq \tanh \left[ \frac{1}{2} \ln \left( \frac{1+\rho}{1-\rho} \right) + \frac{t_{1-\alpha/2, N}}{\sqrt{N-3}} \right].$$

## 24.4. 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].

### 24.4.1. 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  $\sigma$  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:

$$N = \frac{10}{P[G(x) \leq 0]}.$$

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:

$$N = \frac{pq}{\sigma^2} = \frac{(0.9)(0.1)}{(0.051)^2} = 3457.$$

### 24.4.2. Monte Carlo analysis using metamodels

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 for linear approximations (the default being  $1.5n$  points) and quadratically for quadratic approximations (the default being  $0.75(n+2)(n+1)$  points).

### 24.4.3. Correlated variables

Considering the correlation  $Cov(Y_i, Y_j) = E[(Y_i - \mu_i)(Y_j - \mu_j)] = \Sigma_{ij}$  between variables, we construct the covariance matrix

$$\begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{bmatrix},$$

from which we compute the eigenvalues and eigenvectors as  $\Sigma E = \lambda^2 E$  with  $E$  and  $\lambda^2$  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

$\mathbf{X} = \lambda^1 E^1 iid^1 + \dots + \lambda^n E^n iid^n$  with  $\mathbf{X}$  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

$$E(F) = \sum_{i=1}^n a_i \mu_i,$$

$$V(F) = \sum_{i=1}^n a_i^2 V(Y_i) + 2 \sum_{i=1}^n \sum_{j=i+1}^n a_i a_j COV(Y_i, Y_j).$$

#### 24.4.4. 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:

$$P_f = \Phi(-\beta) \text{ or } 1 - \Phi(\beta),$$

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.

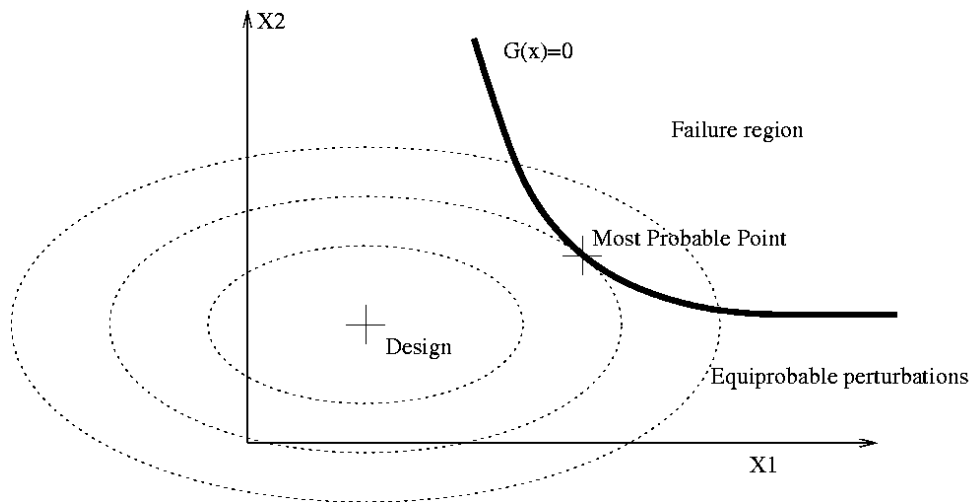
#### 24.4.5. Design for six-sigma methods

See the section for FOSM keeping in mind that the reliability index  $\beta$  is the number of standard deviations.

### 24.4.6. The most probable point

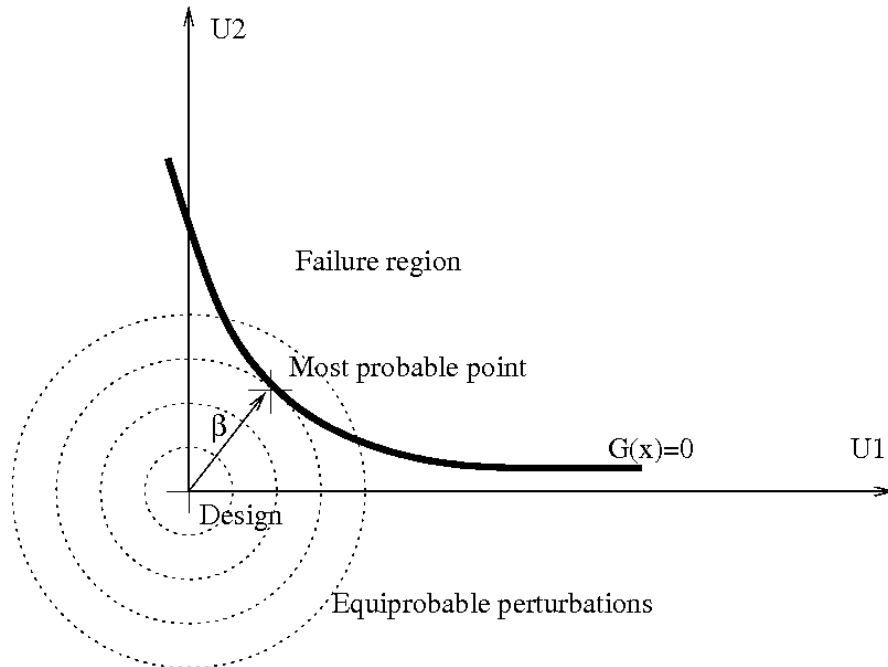
Probabilistic methods based on the most probable point of failure focus on finding the design perturbation most likely to cause failure.

To understand how these methods work, 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 24-1: 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 24-1, 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  $\mathbf{u}$ -space.



**Figure 24-2: 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 24-2. 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:

$$\begin{aligned} \text{Minimize: } & \sqrt{\sum_{i=1}^n u_i^2} \\ \text{Subject to: } & G(u) = 0. \end{aligned}$$

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  $\beta_{\text{FORM}}$ .

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.

#### 24.4.7. 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  $\mathbf{u}$ -space to find the first-order probability index  $\beta_{FORM}$ . 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  $\Phi$  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.

### 24.4.8. 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  $\mathbf{u}$  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  $\beta/(f_{constraint} - f_{nominal})$ .

For the pathological case of being at the MPP, the vector associated with  $\beta$  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.

## 24.5. Required number of simulations

### 24.5.1. 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.

### 24.5.2. 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 must 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.

### 24.5.3. 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  $\mathbf{y}$  and the variables  $\mathbf{x}$  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  $\varepsilon$  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.

#### 24.5.4. Confidence interval on the mean

For multiple regression, the  $100(1-\alpha)\%$  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  $\sigma^2$ . 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 [6].

#### 24.5.5. Confidence interval on a new evaluation

For multiple regression, the  $100(1-\alpha)\%$  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  $2\sigma$  for more than 25 existing runs (20 degrees of freedom).



**Table 24-1: Bounds for a new evaluation of the mean for a linear approximation**

n	p	n-p	C <sub>11</sub>	Bounds ( $\sigma=10\%$ $\alpha=5\%$ )
10	6	4	0.104	$\pm 29\%$
15	6	9	0.070	$\pm 23\%$
20	6	14	0.051	$\pm 22\%$
25	6	19	0.041	$\pm 21\%$
30	6	24	0.034	$\pm 21\%$
50	6	44	0.020	$\pm 20\%$
100	6	94	0.010	$\pm 20\%$

### 24.5.6. Confidence interval on the noise (stochastic process) variance

The noise (stochastic process) variance can be estimated by considering the residuals of the response 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 ( $\sigma^2$ ) 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  $\sigma^2$  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  $P[b_1 \leq s^2 \leq b_2] = 0.95$ . We rewrite as  $P\left[\frac{n-p}{\sigma^2} b_1 \leq \frac{n-p}{\sigma^2} s^2 \leq \frac{n-p}{\sigma^2} b_2\right] = 0.95$ . 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  $P\left[a_1 \leq \frac{n-p}{\sigma^2} s^2 \leq a_2\right] = 0.95$

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

$\left(\frac{s^2}{n-p} a_1, \frac{s^2}{n-p} a_2\right)$ . The  $100(1-\alpha)\%$  confidence interval on  $\sigma^2$  is therefore

$$\left( \frac{(n-p)s^2}{\chi_{\alpha/2, n-p}^2}, \frac{(n-p)s^2}{\chi_{1-\alpha/2, n-p}^2} \right)$$

In the table below we monitor the error bounds on the variance for a problem with six parameters (including the mean).

**Table 24-2: Error bounds on variance**

Noise Variance Confidence Interval								
n	n-p	Lower Bound			Value (s)	Upper Bound		
		$\alpha=5\%$	$\alpha=10\%$	$\alpha=20\%$		$\alpha=20\%$	$\alpha=10\%$	$\alpha=5\%$
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  $\sigma^2$  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  $\sigma^2$  will usually be too large [7].

### 24.5.7. 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.  $P[\text{observing 0 events}] = (1-P[\text{event}])^n$
3.  $P[\text{observing 1 or more events}] = 1.0 - (1-P[\text{event}])^n$

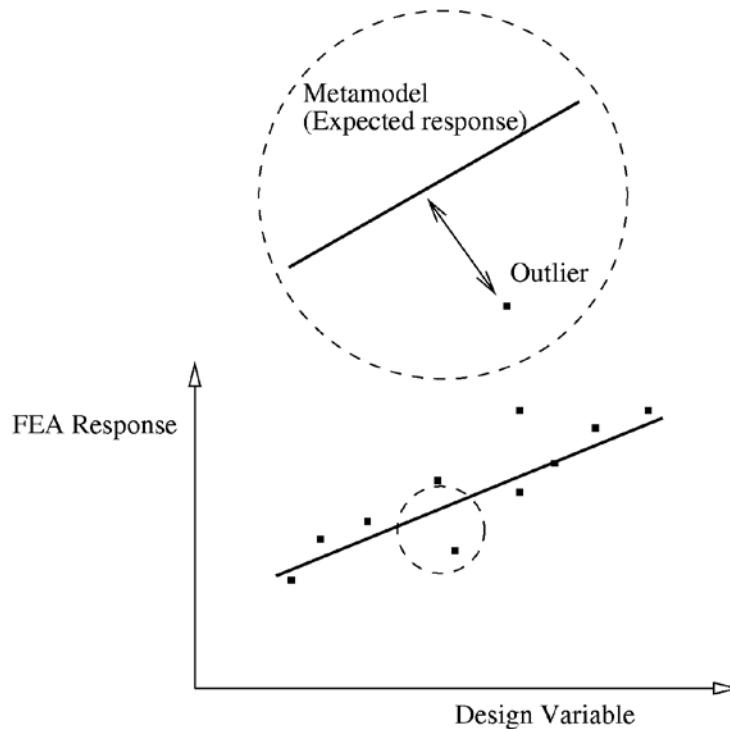
**Table 24-3: Number of runs required to observe an event with a specific probability**

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

## 24.6. 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 24-3:** 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. 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.

## 24.7. Stochastic contribution analysis

The variation of the response can be broken down in contributions from each design variable.

### 24.7.1. 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  $\sigma_T^2$  is the variation of the response due to the variation of all the variables and  $\sigma_i^2$  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 24.4.3.

### 24.7.2. 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 [8] [9]. Firstly the function is decomposed as:

$$f(x_1, \dots, x_n) = f_0 + \sum_{i=1}^n f_i(x_i) + \sum_{i=1}^n \sum_{j=i+1}^n f_{ij}(x_i, x_j) + \dots + f_{1,2,\dots,n}(x_1, \dots, x_n).$$

From which partial variances are computed as:

$$V_{i,\dots,j} = \int_0^1 \dots \int_0^1 f_{i,\dots,j}^2(x_1, \dots, x_n) dx_i \dots dx_j,$$

with the variance of the response summed from the partial variances as:

$$V = \sum V_i + \sum_{i < j} V_{ij} + \dots + V_{1,2,\dots,n}.$$

The sensitivity indices are given as:

$$\begin{aligned} S_i &= V_i / V, & 1 \leq i \leq n, \\ S_{ij} &= V_{ij} / V, & 1 \leq i < j \leq n, \\ S_{i,i+1,\dots,n} &= V_{i,i+1,\dots,n} / V. \end{aligned}$$

with the useful property that all of the sensitivity indices sum to 1:

$$\sum S_i + \sum_{i < j} S_{ij} + \dots + S_{1,2,\dots,n} = 1.$$

Using Monte Carlo, the main effect can be computed as

$$\hat{D}_i = \frac{1}{N} \sum_{m=1}^N f(x_{\sim i}^{(1)}, x_{\sim i}^{(1)}) f(x_{\sim i}^{(2)}, x_{\sim i}^{(1)}) - \hat{f}_0^2$$

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  $x_i$

$$\hat{D}_{\sim i} = \frac{1}{N} \sum_{m=1}^N f(\mathbf{x}_{\sim i}^{(1)}, x_{\sim i}^{(1)}) f(\mathbf{x}_{\sim i}^{(1)}, x_{\sim i}^{(2)}) - \hat{f}_0^2.$$

For second order response surfaces this can be computed analytically [10] as

$$\sigma_U^2 = \sum_{i \in U} \left[ \beta_{ii}^2 (m_{i,4} - \sigma_i^4) + (\beta_i + \beta_{ii} \mu_i + \sum_{j=1}^n \beta_{ij} \mu_j)^2 \sigma_i^2 + (\beta_i + \beta_{ii} \mu_i + \sum_{j=1}^n \beta_{ij} \mu_j) \beta_{ii} m_{1,3} \right] + \sum_{i \in U} \sum_{i \in U, j \geq 1} \beta_{ij}^2 \sigma_i^2 \sigma_j^2,$$

with  $m_{i,j}$  the  $j$ th moment about the mean of the distribution  $i$  and  $U$  the set of variables under consideration.

- 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.

Correlations between variables are not considered in the computation of the main and total effects of the variables.

## 24.8. 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 24.4.4 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 12.6 with an example in Section 19.3 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.

## **24.9. Robust parameter design**

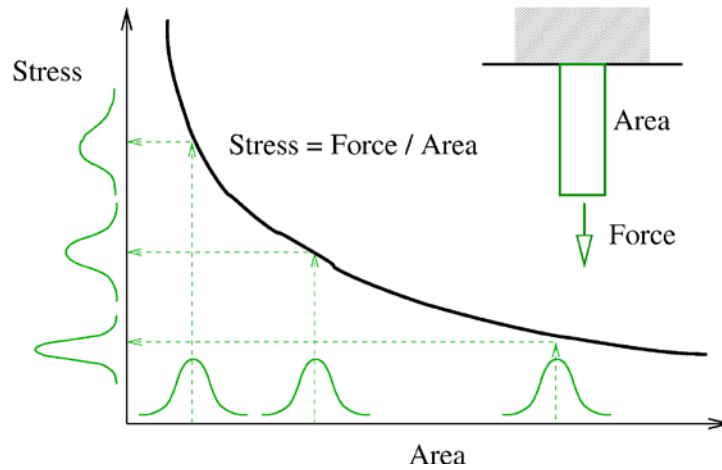
Robust parameter design selects designs insensitive to changes in given parameters.

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 [11]. An alternate view of Taguchi [12] 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 [11].

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.

### **24.9.1. Fundamentals**

The robustness of a structure depends on the gradient of the response function as shown in Figure 24-4. 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.

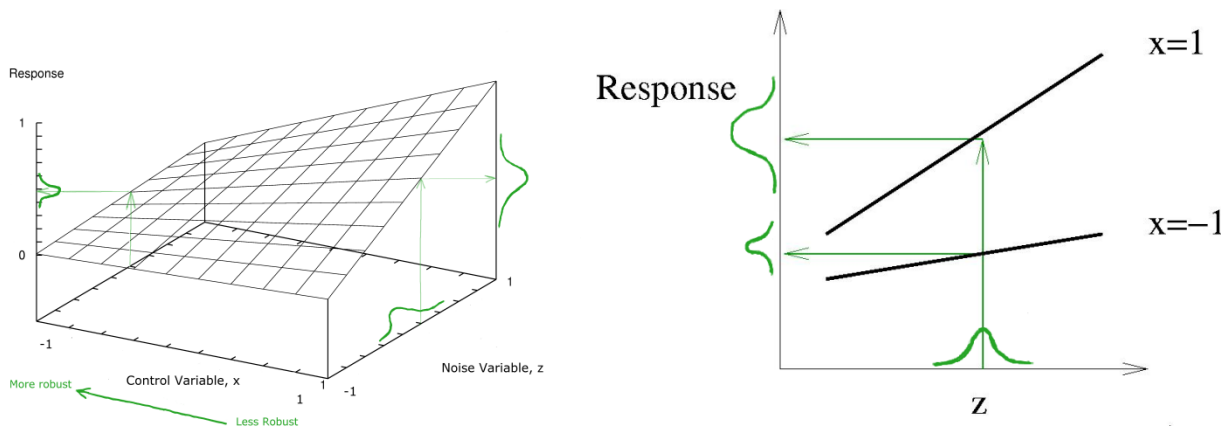


**Figure 24-4: Robustness considering a single variable.** Larger mean values of the area result in a smaller dispersion of the stress values. Note that the dispersion of the stress depends on the gradient of the stress-area relationship.

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 24-5 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 24-5: 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.*

### 24.9.2. Methodology

The dual response surface method as proposed by Myers and Montgomery [6] using separate models for process mean and variance is considered. Consider the control variables  $\mathbf{x}$  and noise variables  $\mathbf{z}$  with  $Var(\mathbf{z}) = \sigma_z^2 I_{r_z}$ . The response surface for the mean is  $E_z[y(x, z)] = \beta + x' \beta + x' \beta x$  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  $Var(\mathbf{z}) = \sigma_z^2 I_{r_z}$ ,  $\sigma^2$  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 [6]. 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.”

### 24.9.3. 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.

## 24.10. 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] Box., G.E.P., Draper, N.R., *Empirical Model Building and Response Surfaces*, Wiley, New York, 1987.
- [6] Myers, R.H., Montgomery, D.C., *Response Surface Methodology. Process and Product Optimization using Design Experiments*. Wiley, 1995.
- [7] Draper, N.R., Smith, H., *Applied Regression Analysis*, Second Edition, Wiley, New York, 1981.
- [8] Sobol IM, Sensitivity analysis for nonlinear mathematical models, *Mathematical Modeling and Computer Experiments*, 1(4), pp. 407-413, 1993.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] Roy RK. *Design of Experiments Using the Taguchi Approach*. Wiley, New York NY. 2001.

# **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
<i>res_type</i>	Result type name	-	<b>1</b>
<i>sub</i>	Result subdirectory	-	1
<i>cmp</i>	Component of result	-	2
<i>invariant</i>	Invariant of results. Only MAGNITUDE is currently available.	-	3
<i>id</i>	ID number of entity	-	
<i>name</i>	Description (heading) of entity used as label	-	4
<i>pos</i>	Through thickness shell position at which results are computed.	1	5
<i>side</i>	Interface side for RCFORC data. MASTER or SLAVE.	SLAVE	
<i>frame</i>	GLOBAL   GLOBAL_IN_REF   LOCAL	GLOBAL	6
<i>localid</i>	3 Nodal ID's to define local coordinate axes	-	7
<i>localname</i>	3 Nodal names (headings) to define local coordinate axes	-	7
<i>id i</i>	Multiple ID's $i = 1, 2, 3, \dots n$	-	8
<i>name i</i>	Multiple headings as labels $i = 1, 2, 3, \dots n$	-	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
<i>history_options</i>	All available history options	-
<i>filter_type</i>	Type of filter to use: SAE or BUTT	-
<i>filter_freq</i>	Filter frequency	60 cycles / time unit
<i>units</i>	S=seconds MS=milliseconds	S
<i>ave_points</i>	Number of points to average	-
<i>start_time</i>	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
<i>history_options</i>	All available history options including averaging, filtering, and slicing.	-	
<i>Selection</i>	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
<i>history_options</i>	All available history options including filtering and slicing.	-
<i>cmp</i>	HIC15, HIC36, or CSI	-
<i>lengthunits</i>	METER=meter MM=millimeter	METER
<i>units</i>	S=seconds MS=milliseconds	S

*Remarks:*

1. The length and time units are used to compute the gravity value based on  $9.81 \text{ m/s}^2$

#### A.3.2 Bilinear FLD constraint

The values of both the principle upper and lower surface in-plane strains are used for the FLD constraint.

---

DynaFLD p1 p2 ... pn intercept negative\_slope positive\_slope

---

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

**Table 24-4: DynaFLD item description**

Item	Description
<i>p1...pn</i>	Part numbers of the model. Omission implies the entire model.
<i>intercept</i>	The FLD curve value at $\varepsilon_2 = 0$
<i>negative_slope</i>	The absolute value of the slope of the FLD curve value at $\varepsilon_2 < 0$
<i>positive_slope</i>	The absolute value of the slope of the FLD curve value at $\varepsilon_2 > 0$

# 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
<i>Subdirectory discrete/nodes</i>	
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
<i>Subdirectory solid</i>	
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
<i>Subdirectory beam</i>	
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
<i>Subdirectory shell</i>	
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
<i>Subdirectory thickshell</i>	
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
<i>Subdirectory joints</i>	
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
<i>Subdirectory type0</i>	
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
<i>Subdirectory master_00001 and slave_00001</i>	
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
<i>Translational components</i>	
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
<i>Rotational components</i>	
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
<i>Translational components</i>	
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
<i>Subdirectory forces</i>	
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
<i>res_type</i>	Result type name	-	1
<i>cmp</i>	Component of result	-	1
<i>id</i>	ID number of entity	-	2
<i>pos</i>	Through thickness shell position	1	
<i>pids</i>	One or more part ids.	-	3
<i>loc</i>	Locations in model. ELEMENT or NODE.	-	4
<i>select_in_region</i>	MAX   MIN   AVE	-select	5
<i>coord</i>	Coordinate of a point for finding nearest element	-	6
<i>tref</i>	Time of reference state for finding nearest element	0.0	6
<i>setid</i>	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 .
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
<i>history_options</i>	All available history options	-
<i>start_time</i>	Start time of history interval to extract using slicing	0
<i>end_time</i>	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
<i>history_options</i>	All available history options	-
<i>fld_t</i>	Fld curve $t$ coefficient	-
<i>fld_n</i>	Fld curve $n$ coefficient	-
<i>fld_curve</i>	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
<i>history_options</i>	All available history options	-	
<i>select</i>	MAX   MIN   AVE   TIME	TIME	1

### Remarks:

- 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.



# 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
	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	

Result Type	Number	Description	Component name
	41	Effective lower surface stress	l_surf_eff_stress
Strain	43	Lower surface effective plastic strain	l_surf_plastic_strain
	44	Upper surface effective plastic strain	u_surf_plastic_strain
	45	Lower surface <i>xx</i> , <i>yy</i> , <i>zz</i> , <i>xy</i> , <i>yz</i> , <i>zx</i> 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 <i>xx</i> , <i>yy</i> , <i>zz</i> , <i>xy</i> , <i>yz</i> , <i>zx</i> 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 <i>xx</i> , <i>yy</i> , <i>zz</i> , <i>xy</i> , <i>yz</i> , <i>zx</i> 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
	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

---

<b>Result Type</b>	<b>Number</b>	<b>Description</b>	<b>Component name</b>
	516		maxima_eps1
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 Files

## E.1 Design flow

Source Database file	Task	Output Database file	Directory for output database
Command file ( <code>.lsopt</code> )	Point selection	<code>Experiments_n.csv</code>	Sampling
<code>Experiments_n.csv</code>	Simulation runs	Solver output files	Run
Solver output files	Result extraction	<code>StageResults_n.lsox</code>	Stage
<code>StageResults_n.lsox</code>	Assemble Sampling Results. Compute extended results.	<code>AnalysisResults_n.lsox</code> <code>AnalysisResults_n.csv</code> <code>ExtendedResults.n</code> <code>ExtendedResults_n.csv</code>	Sampling
<code>AnalysisResults_n.csv</code> <code>AnalysisResults_n.lsox</code>	Approximation	<code>DesignFunctions.n</code> <code>VirtualHistories.n</code> <code>Net.func_name</code>	Sampling
<code>DesignFunctions.n</code> <code>VirtualHistories.n</code>	Optimize	<code>OptimumResults.n</code> <code>OptimizationHistory</code> <code>OptimizerHistory_n.csv</code> <code>lsopt_results_n.csv</code>	Work

## E.2 Output files

The following files are intermediate database files containing ASCII data.

### E.2.1 Intermediate database files

Database file	Description	Directory
StageResults_ <i>n</i> .lsox	Response and history results of a stage.	Stage
Experiments_ <i>n</i> .csv	Trial designs computed as a result of the experimental design	Sampling
AnalysisResults_ <i>n</i> .lsox	.xml file containing all the extracted results including responses, matrices and histories.	Sampling
AnalysisResults_ <i>n</i> .csv	The same trial designs and the responses extracted from the solver database	Sampling
DesignFunctions_ <i>n</i>	Parameters of the approximate functions	Sampling
VirtualHistoryFunction	Approximation functions data for histories	Work
OptimizationHistory	Variable, response and error history of the successive approximation process	Work
OptimizerHistory	Detailed history of the optimizer	Work
OptimumResults_ <i>n</i>	Optimum for a particular iteration	Work
OptiState_ <i>n</i> .lsox	Optimization state.	Work
ExtendedResults_ <i>n</i>	All variables, responses and extended results at each trial design point	Sampling
ExtendedResultsMaster_ <i>n</i>	ExtendedResults for a master case (sampling). Cases with exactly the same experimental design are grouped in a master case.	Sampling
Net_ <i>funcname</i>	Parameters of the metamodel of function with name <i>funcname</i>	Sampling
Variables_ <i>n</i>	The variable values, confidence intervals and bounds	Work
ANOVA_ <i>n</i>	ANOVA data used in the Viewer Sensitivity feature.	Sampling
SubRegionBounds_ <i>n</i>	Bounds of the subregion.	Sampling

VariableMap\_n.lsox      Variable connectivity for each sampling.      Sampling

## E.2.2 Database files in .csv (comma separated variables) format

Database file	Description	Directory	Remarks
Experiments_n.csv	Experiments ( $n$ = iteration number)	Sampling	
AnalysisResults_n.csv	Analysis Results	Sampling	
ExtendedResultsMaster_n.csv	Extended Results (variables, dependents, responses, composites, objectives, constraints, multiobjective)	Sampling	
ExtendedResultsMETAMaster_n.csv	Extended Results file for user-defined Experiments file	Sampling	Section 7.5.1
PRESS_predictions_n.csv	PRESS (Section 20.3.4) predicted results and PRESS residuals (Polynomials and Radial Basis Function networks (Section 21.1.2) only. PRESS residuals are not computed for Feedforward Neural Networks)	Sampling	Use check box to select PRESS in Viewer→ Accuracy→
OptimizerHistory_n.csv	Detailed history of the optimizer for iteration $n$	Work	

## E.2.3 Text output files

Database file	Description	Directory	View option
lsopt_input	Input in a formatted style	Work	Input
lsopt_output	Results and some logging information. Usually a very large file.	Work	Output
lsopt_report	A final report of the analysis results. Available for some of the main tasks and most of the Repair	Work	Summary

tasks			
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.	Work	File
lsopt_results_n.binout	All variable, responses and extended results of the non-dominated solutions at each iteration	Work	-

### E.3 Database file formats

The database consists of text files, text files with comma separated values (.csv format) 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.3.1 Variable types

Symbol	Explanation
dv	Design variable
nv	Noise variable
dc	Discrete variable
st	String variable
rs	Response
sk	Ignore this column

#### E.3.2 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.3.3 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  $2.0 \times 10^{30}$  are assigned to responses of simulations with error terminations. The AnalysisResults\_n.csv file is synchronous with the Experiments\_n.csv file.

### E.3.4 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

$$f(x) = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n + \beta_{1_1} x_1 x_1 + \beta_{1_2} x_1 x_2 + \dots + \beta_{1_n} x_1 x_n + \beta_{2_2} x_2 x_2 + \dots + \beta_{2_n} x_2 x_n + \dots + \beta_{n_n} x_n x_n$$

where  $x \in [0,1]^n$ .



The following enumerations are used in the database.

<b>Function Types</b>	
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:*

- The flags for active coefficients exclude the constant  $a_0$ .
- The coefficients are based on the independent variables  $x_i$  having been normalized to the size of the design space.

**E.3.5 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 *DesignFunctions*.

**E.3.6 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 <sup>#</sup>	1
Individual number <sup>#</sup>	1

\*Only written for RBDO problems.

<sup>#</sup>Only written for Direct GA simulations.

Values of  $2.0 \times 10^{30}$  are assigned to responses of error terminations.

### E.3.7 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  $2.0 \times 10^{30}$  are assigned to responses of simulations with error terminations.

### E.3.8 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.3.9 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.3.10 The Isopt\_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.3.11 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.

# 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
<code>int(a)</code>	integer
<code>nint(a)</code>	nearest integer
<code>abs(a)</code>	absolute value
<code>mod(a,b)</code>	remainder of $a/b$
<code>sign(a,b)</code>	transfer of sign from $b$ to $ a $
<code>max(a,b)</code>	maximum of $a$ and $b$
<code>min(a,b)</code>	minimum of $a$ and $b$
<code>sqrt(a)</code>	square root
<code>exp(a)</code>	$e^a$
<code>pow(a,b)</code>	$a^b$

$\log(a)$	natural logarithm
$\log_{10}(a)$	base 10 logarithm
$\sin(a)$	sine
$\cos(a)$	cosine
$\tan(a)$	tangent
$\text{asin}(a)$	arc sine
$\text{acos}(a)$	arc cosine
$\text{atan}(a)$	arc tangent
$\text{atan2}(a, b)$	arc tangent of $a/b$
$\sinh(a)$	hyperbolic sine
$\cosh(a)$	hyperbolic cosine
$\tanh(a)$	hyperbolic tangent
$\text{asinh}(a)$	arc hyperbolic sine
$\text{acosh}(a)$	arc hyperbolic cosine
$\text{atanh}(a)$	arc hyperbolic tangent
$\sec(a)$	Secant
$\csc(a)$	cosecant
$\text{ctn}(a)$	cotangent
$\text{cnd}(a)$	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
$\text{inv}(A)$	Inverse of matrix $A$



<code>tr(A)</code>	Transpose of matrix $A$
<code>rx(angle)</code>	Rotation about $x$ -axis (angle in rad)
<code>ry(angle)</code>	Rotation about $y$ -axis (angle in rad)
<code>rz(angle)</code>	Rotation about $z$ -axis (angle in rad)

### F.3 Special functions

Special response functions can be specified to apply to response histories. These include integration, minima and maxima and finding the time at a specific value of the function, Section F.3.1 .

The arguments used in the expressions are explained in Section F.3.2

History 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`, `response` and `composite` are acceptable. An expression is given in double quotes, e.g., "4.2 \* C1\_1 \* Displacement( $t$ )".

#### F.3.1 Functions to apply to response histories

Expression	Symbols/Description
<code>FilterHistory(history[,filtertype, frequency, timeunits, num_average])</code>	Filtered curve using SAE, Butterworth or running average
<code>DerivativeHistory(history, order)</code>	$\frac{df}{dt} \approx \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h}$
<code>Crossplot(history_z, history_F [, numpoints, begin_time, end_time])</code>	$F(z)$ given $F(t)$ and $z(t)$
<code>Integral(expression[,t_lower,t_upper,variable])</code>	$\int_a^b f(t)dg(t)$
<code>Derivative(expression[,T_constant])</code>	$\Delta f / \Delta t _{t=T} \sim df/dt _{t=T}$
<code>Min(expression[,t_lower,t_upper])</code>	$f_{\min} = \min_t[f(t)]$
<code>Max(expression[,t_lower,t_upper])</code>	$f_{\max} = \max_t[f(t)]$

<code>Initial(expression)</code>	First function value on record
<code>Final(expression)</code>	Last function value on record
<code>TerminationTime(expression)</code>	Termination time. Time of last history value.
<code>Lookup(expression,value[,t_lower,t_upper])</code>	Inverse function $t(f=F)$
<code>LookupMin(expression[,t_lower,t_upper])</code>	Inverse function $t(f=f_{\min})$
<code>LookupMax(expression[,t_lower,t_upper])</code>	Inverse function $t(f=f_{\max})$
<code>MeanSqErr(target_curve,computed_curve[,num_reg_points,start_point,end_point,weight_type,scale_type,weight_value,scale_value,weight_curve_name,scale_curve_name])</code>	Mean Squared Error function $\frac{1}{P} \sum_{p=1}^P W_p \left( \frac{f_p(\mathbf{x}) - G_p}{s_p} \right)^2$
<code>CurveMapSegment3(target_curve,computed_curve[,num_reg_points])</code>	Curve Mapping discrepancy

### F.3.2 Arguments used in functions

Argument	Explanation	Symbol	Type
<code>expression</code>	history defined as an expression string	$f(t)$	generic
<code>filtertype</code>	Filtering type (SAE, Butterworth, running average)	-	integer
<code>frequency</code>	Filtering frequency (Hz)	$f$	float
<code>timeunits</code>	Time units (s or ms)	-	integer
<code>num_average</code>	Number of points averaged	-	integer
<code>order</code>	Order of differentiation (3 or 5 point template)	-	integer
<code>history_z, history_F</code>	History names for abscissa and ordinate	$z(t)$ , $F(t)$	history
<code>numpoints</code>	Number of points in curve. Default: Smallest of the numbers of	-	integer

	points defining $z$ and $F$		
<i>begin_time, end_time</i>	Begin and end times Default: Largest $t_0$ -value of $F$ and $z$ and smallest $t_P$ -value of $Z$ and $z$ , respectively	$t_0, t_P$	float
<i>t_lower</i>	lower limit of integration or range	$a$	generic
<i>t_upper</i>	upper limit of integration or range	$b$	generic
<i>variable</i>	integration variable	$g(t)$	generic
<i>T_constant</i>	specific time	$T$	generic
<i>value</i>	value for which lookup is required	$F$	generic
<i>target_curve, computed_curve</i>	Target, computed curve names	$G, f$	history
<i>num_reg_points</i>	Number of regression points If $P < 2$ or not specified: use number of points in target curve between <i>start_point</i> and <i>end_point</i> .	$P$	integer
<i>start_point, end_point</i>	Location of first/last regression points	$z_0, z_P$	float
<i>weight_type</i>	Weight type (F.3.3 ): ○ WEIGHTVALUE (default) ○ PROPWEIGHT ○ FILEWEIGHT		reserved
<i>scale_type</i>	Scale type (F.3.3 ): ○ SCALEVALUE ○ PROPSCALE ○ MAXISCALE (default) ○ FILESCALE		reserved
<i>weight_value</i>	Uniform weight value (default = 1)	$W$	float
<i>scale_value</i>	Uniform scale value (default = 1)	$s$	float
<i>weight_curve</i>	Weights as a function of $z$ (default	$W(z)$	History

---

	<code>Weight.compositename)</code>		
<code>scale_curve</code>	Scale factors as a function of $z$ (default <code>Scale.compositename</code> )	$s(z)$	History

---

**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. The implied variable represented in the first column of any history file is  $t$ . Therefore all history files produced by the `DynaASCII` extraction command contain functions of  $t$ . The fourth argument of the `Integral` function defaults to  $t$ . The variable  $t$  must increase monotonically.
4. The derivative assumes a piecewise linear function defined by the points in the `history.n` file.  $T\_constant$  in the `Derivative` function defaults to the end time.
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`). For derivatives the first or last slopes are returned respectively.

**F.3.3 Options for MeanSqErr arguments**


---

Syntax	Explanation
<code>WEIGHTVALUE</code>	$W_i = value$ . Default = 1.0
<code>PROPWEIGHT</code>	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 $
<code>FILEWEIGHT</code>	Interpolate the weight from an x-y file: weight vs. $z$
<code>SCALEVALUE</code>	$s_i = value$ . Default = 1.0
<code>MAXISCALE</code>	$\max  G_p $ .
<code>PROPSCALE</code>	Use a different scale factor for each curve point, namely $ G_p $ .
<code>FILESCALE</code>	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  $f(\mathbf{x})$  is linear in terms of the design variables  $\mathbf{x}$ , the composite function will be exactly represented.
4. Empty or underscore (`_`) arguments will generate default values.
5. The option names in Section F.3.3 are reserved names and cannot be used as variable names.
6. MeanSqErr composites can be added together to make a larger MSE composite (e.g. for multiple test cases).
7. The simplest target curve that can be defined has only one point.

**F.4 Matrix functions**

Expression	Symbols/Description		
<code>Rotate(x1,y1,z1, x2,y2,z2, x3,y3,z3)</code>	Rotation matrix defined by 3 points. See Section 6.4.4.		
<code>Matrix3x3Init(x1,y1,z1, x2,y2,z2, x3,y3,z3)</code>	Initialize 3×3 matrix		
Argument	Explanation	Symbol	Type
<code>x1,y1,z1,x2,y2,z2,x3,y3,z3</code>	Matrix components	-	generic

**F.5 Reserved variable names**

Name	Explanation
<code>t</code>	Time
<code>LowerLimit</code>	0.0
<code>UpperLimit</code>	Maximum event time over all histories of all solvers
<code>length</code>	Intrinsic variable

## F.6 Constants associated with histories

The following commands can be given to override defaults for history operations:

Constant	Explanation	Default
<code>variable fdstepsize</code>	Finite difference step size for numerical derivatives with respect to variables	$0.0001 * (\text{Upper bound} - \text{Lower bound})$
<code>Historysize</code> (attribute of <code>globals</code> 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)", "l1 * 1.0")
Max("his1(t)", "l1", ul)
Max("his3(t)", l1, ul)
Min("his3(t)", l1, 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)", l1, ul, "t")
Integral("his1(t)", l1, UPPER, "t")
Integral("his2(t)", l1, ul, "t")
Integral("his3(t)", l1, ul, "t")
Integral("(his1(t) + his2(t))/2", l1, ul, "t")
Integral("his3(t)")
Integral("his3(t)", l1)
Integral("his3(t)", l1, ul)
Integral("his1(t)", l1, 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)", l1, 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)
```



# Appendix G: Injury Criteria

## G.1 Syntax of Injury Criterion Commands

Expression	Symbols/Description	Reference
<i>MOC</i> ( <i>force</i> , <i>moment</i> [, <i>dummy_type</i> , <i>length_units</i> , <i>force_units</i> , <i>distance</i> ])	$MOC = M - (D \cdot F)$	G.1.1
<i>NIC</i> ( <i>accel_t</i> , <i>accel_h</i> [, <i>time_units</i> , <i>length_units</i> ])	$NIC = a_{relative} \cdot 0.2 + v_{relative}^2$	G.1.3
<i>Nij</i> ( <i>force</i> , <i>moment</i> , <i>shear</i> [, <i>dummy_type</i> , <i>length_units</i> , <i>force_units</i> , <i>c_force_tension</i> , <i>c_force_compression</i> , <i>c_moment_flexion</i> , <i>c_moment_extension</i> , <i>distance</i> ])	$NIJ = \frac{F}{F_c} + \frac{MOC}{M_c}$	G.1.4
<i>Nkm</i> ( <i>force</i> , <i>moment</i> [, <i>dummy_type</i> , <i>length_units</i> , <i>force_units</i> , <i>criteria_type</i> , <i>distance</i> , <i>critical_force</i> , <i>critical_moment</i> ])	$Nkm(t) = \frac{F(t)}{F_{int}} + \frac{MOC(t)}{M_{int}}$	G.1.6
<i>LNL</i> ( <i>axial_force</i> , <i>s_shear</i> , <i>t_shear</i> , <i>s_moment</i> , <i>torsion</i> [, <i>length_units</i> , <i>force_units</i> , <i>critical_moment</i> , <i>critical_shear</i> , <i>critical_tension</i> ])	$LNL = \frac{\sqrt{M_y^2 + M_x^2}}{C_{moment}} + \frac{\sqrt{F_y^2 + F_x^2}}{C_{shear}} + \frac{ F_z + off }{C_{tension}}$	G.1.9
<i>ChestCompression</i> ( <i>relative_rotation</i> , <i>dummy_type</i> , <i>user_constant</i> )	$C_1 \max_t [\Theta(t)]$	G.1.10
<i>ViscousCriterion</i> ( <i>history_name</i> , [ <i>dummy_type</i> , <i>time_units</i> , <i>length_units</i> , <i>scaling_factor</i> , <i>deformation_constant</i> , <i>user_constant</i> ])	$-\min \frac{C_1}{C_2} C_3 Y(t) C_3 \frac{dY(t)}{dt}$	G.1.12
<i>TTI</i> ( <i>accel_upper_rib</i> , <i>accel_lower_rib</i> , <i>accel_lower_spine</i> , <i>time_units</i> , <i>length_units</i> , <i>gravity</i> )	$TTI = \frac{A(\max.rib) + A(lwr.spine)}{2}$	G.1.14
<i>TibiaIndex</i> ( <i>torsion</i> , <i>s_moment</i> , <i>t_shear</i> , <i>dummy_type</i> , <i>length_units</i> , <i>force_units</i> , <i>critical_moment</i> ,	$TI = \left  \frac{M}{M_c} \right  + \left  \frac{F}{F_c} \right $	G.1.15

---

*critical\_force*)

---

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
<i>Force</i>	Neck axial force resultant	F	History	
<i>moment</i>	Neck s-moment resultant	M	History	
<i>dummy_type</i>	Dummy type	-	G.1.2	HYBRID3M50
<i>length_units</i>	Length units	-	LENG_M LENG_MM	LENG_MM
<i>force_units</i>	Force units	-	FORCE_N FORCE_KN	FORCE_N
<i>Distance</i>	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
<i>accel_t</i>	x-acceleration of first thorax spine	$a_x^{Tl}$	History	
<i>accel_h</i>	x-acceleration at the height of the c.o.g. of the head	$a_x^{Head}$	History	
<i>time_units</i>	Time units	-	TIME_S TIME_MS	TIME_S
<i>length_units</i>	Length units	-	LENG_M LENG_MM	LENG_MM

### G.1.4 Options for Nij arguments

Argument name	Description	Symbol	LS-OPT type	Default
<i>force</i>	Neck axial force resultant	See MOC	History	
<i>moment</i>	Neck s-moment resultant	See MOC	History	

<i>shear</i>	Force at the point of transition from head to neck	$F$	History	
<i>dummy_type</i>	Dummy type	-	Table 6-15	HYBRID3M50
<i>length_units</i>	Length units	-	LENG_M, LENG_MM	LENG_MM
<i>force_units</i>	Force units	-	FORCE_N, FORCE_KN	FORCE_N
<i>c_force_tension</i>	Critical force tension	$F_c$	Float	G.1.5
<i>c_force_compression</i>	Critical force compression	$F_c$	Float	G.1.5
<i>c_moment_flexion</i>	Critical moment flexion	$M_c$	Float	G.1.5
<i>c_moment_extension</i>	Critical moment extension	$M_c$	Float	G.1.5
<i>distance</i>	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
<i>force</i>	Neck axial force resultant	$F$	History	
<i>moment</i>	Neck s-moment resultant	See MOC	History	
<i>dummy_type</i>	Dummy type	-	Table 6-11 (MOC)	HYBRID3M50
<i>length_units</i>	Length units	-	LENG_M, LENG_MM	LENG_MM
<i>force_units</i>	Force units	-	FORCE_N, FORCE_KN	FORCE_N
<i>criteria_type</i>	Nfa, Nea, Nfp, Nep	-	FLEXION_ANTERIOR EXTENSION_ANTERIOR FLEXION_POSTERIOR EXTENSION_POSTERIOR	FLEXION_ANTERIOR
<i>distance</i>	Distance	$D$ (See MOC)	Float	Table 6-11 (MOC)
<i>critical_force</i>	Critical force	$F_{int}$	Float	G.1.7
<i>critical_moment</i>	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
<i>axial_force</i>	Axial force resultant	$F_y$	History	

<i>s_shear</i>	s-Shear resultant	$F_x$	History	
<i>t_shear</i>	t-Shear resultant	$F_z$	History	
<i>s_moment</i>	s-Moment resultant	$M_y$	History	
<i>torsion</i>	Torsional resultant	$M_x$	History	
<i>length_units</i>	Length units	-	LENG_M, LENG_MM	LENG_MM
<i>force_units</i>	Force units	-	FORCE_N, FORCE_KN	FORCE_N
<i>critical_moment</i>	Critical moment	$C_{moment}$	Float	<b>G.1.9</b>
<i>critical_shear</i>	Critical force	$C_{shear}$	Float	<b>G.1.9</b>
<i>critical_tension</i>	Critical force	$C_{tension}$	Float	<b>G.1.9</b>

### 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
<i>relative_rotation</i>	relative rotation history	$\Theta(t)$	History	
<i>dummy_type</i>	dummy type	-	HYBRID3M95 HYBRID3M50	HYBRID3M50

HYBRID3F05

<i>constant</i>	Multiplier	$C_1$	Float	Table 6-21
-----------------	------------	-------	-------	------------

### G.1.11 Chest Compression 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
<i>history_name</i>	Thoracic deformation (m)	$Y(t)$	History	
<i>dummy_type</i>	Dummy type	-	Table 6-23	HYBRID3M50
<i>time_units</i>	Time units	-	TIME_S TIME_MS	TIME_S (seconds)
<i>length_units</i>	Length units	-	LENG_M LENG_MM	LENG_MM (mm)
<i>scaling_factor</i>	Scaling factor (multiplier)	$C_1$	Float	G.1.13
<i>deformation_constant</i>	Constant: Depth or width of half the rib cage (m)	$C_2$	Float	G.1.13
<i>user_constant</i>	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
<i>accel_upper_rib</i>	y-acceleration of the upper rib	$A(ubr.rib)$	History	
<i>accel_lower_rib</i>	y-acceleration of the lower rib	$A(lwr.rib)$	History	
<i>accel_lower_spine</i>	y-acceleration of the lower spine	$A(lwr.spine)$	History	
<i>time_units</i>	Time units	-	TIME_S TIME_MS	TIME_S
<i>length_units</i>	Length units	-	LENG_M, LENG_MM	LENG_MM
<i>gravity</i>	Gravitational acceleration	g	Float	Depends on <i>time_units</i> and <i>length_units</i> . 9810 mm/s <sup>2</sup> if units undefined

### G.1.15 Options for TI arguments

Argument name	Description	Symbol	LS-OPT type	Default
<i>Torsion</i>	Bending moment, torsional resultant	$M_x$	History	



<i>s_moment</i>	Bending moment, s-moment resultant	$M_y$	History	
<i>t_shear</i>	Axial compression, t-shear resultant	$F$	History	
<i>dummy_type</i>	Dummy type	-	Table 6-26	HYBRID3M50
<i>length_units</i>	Length units	-	LENG_M, LENG_MM	LENG_MM
<i>force_units</i>	Force units	-	FORCE_N, FORCE_KN	FORCE_N
<i>critical_moment</i>	Critical bending moment	$M_C$	Float	G.1.16
<i>critical_force</i>	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
<i>accel_x</i>	x-acceleration history	$\ddot{x}$	History	
<i>accel_y</i>	y-acceleration history	$\ddot{y}$	History	no history
<i>accel_z</i>	z-acceleration history	$\ddot{z}$	History	no history
<i>time_units</i>	Time units	-	TIME_S TIME_MS	TIME_S (seconds)

<i>length_units</i>	Length units	-	LENG_M LENG_MM	LENG_MM (mm)
<i>time_interval</i>	Time interval for which the acceleration is exceeded	$\Delta t$	Float	0.003 s
<i>gravity</i>	Gravitational acceleration	$g$	Float	Depends on <i>time_units</i> and <i>length_units</i> . 9810 mm/s <sup>2</sup> 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 the LSTCVM Secure Proxy Server.

## H.1 Download

LS-OPT is available at <http://lstc.com/download/ls-opt>. Please ask LSTC 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

There is no installation program required for LS-OPT. Simply create or select a folder where you would like to install LS-OPT, and unzip the contents of the downloaded archive into that folder. Make sure that there are no spaces in the path to the installation directory or in the directory name. 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.

A *.lsopt* file can also be opened in the GUI by double-clicking the *.lsopt* file. When this is done the first time, you will have to select which LS-OPT executable to use (e.g. by browsing). Subsequent requests to open a file by double-clicking will then always be automatic.

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"
  - o Choose "Properties"
  - o Click "Advanced" tab
  - o Click "Environment Variables" button
  - o 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
  - a. You have a *queueing system* and you want to submit some or all LS-OPT solver jobs to that queueing system.
  - b. You can allow remote solver jobs to initiate TCP/IP connections back to the machine where LS-OPT runs.
2. blackbox option
  - a. You have a queueing system and you want to submit some or all LS-OPT solver jobs to that queueing system.
  - b. You prefer not to allow remote solver jobs to initiate TCP/IP connections back to the machine where LS-OPT runs.
3. lstcvm option

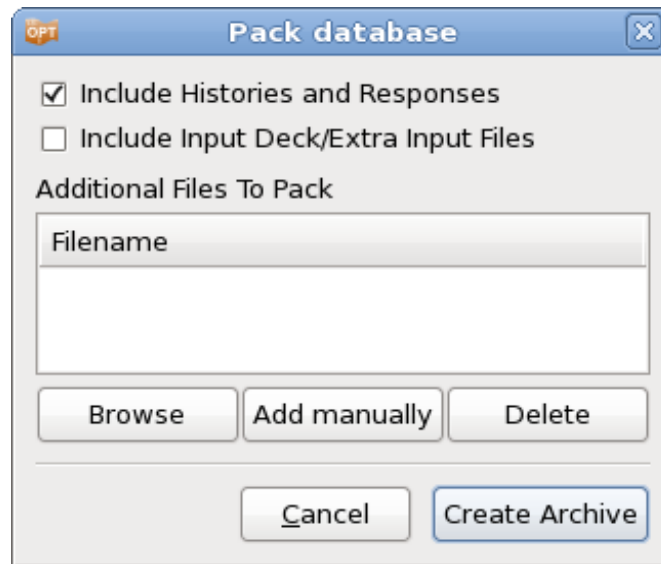
- a. You do not have a queueing system. You would like to run LS-OPT on one machine, but you would like to run all solver jobs on a single, dedicated cluster.
  - b. You can share a file system between LS-OPT and the cluster where the solver jobs are run.
4. `lstcvm/runqueueer/wrapper` option
- a. You do not have a queueing system. You would like to run LS-OPT on one machine, but you would like to run all solver jobs on a single, dedicated cluster.
  - b. You prefer not to share a file system between LS-OPT and the cluster where the solver jobs are run.
  - c. You do allow remote solver jobs on the dedicated cluster to connect via TCP/IP back to the machine where LS-OPT is running.
5. `lstcvm/blackbox` option
- a. You have a queueing system that you would like to use for job submission, but the machine where you would like to run LS-OPT does not have a command line submit utility for the queueing system.
  - b. There is a machine on your system where
    - i. You can install the `lstcvm` job proxy server;
    - ii. You can submit jobs using a command line utility;
    - iii. You can share a file system with the machine where LS-OPT will run.

## H.4 Simple manual setup for running LS-OPT and solvers on different machines

A convenient 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.lstopt`.
2. Open the `lstopt project` file with LS-OPTui and create a second command file e.g. `project.archive.lstopt` by selecting **Archive LS-OPT database** from the control bar **Tools** menu. Select *Include Histories and Responses* and hit *Create Archive* in the **Archive database** dialog, Figure H - 1. The `lstopt_db` file in the work directory contains the settings needed to archive the database later in batch mode. Copy `lstopt_db` to e.g. `project.pack.lstopt`. There are now two command files: `project.lstopt` for running the optimization task and `project.archive.lstopt` for archiving the output data after the run.

3. Install LS-OPT and the solver executables on a cluster node for running LS-OPT in batch mode. Copy the recently created problem setup with the two command files from the Windows machine onto a cluster node. This setup should allow the user to run an LS-OPT job in batch mode.
4. Run LS-OPT by executing the command: `lsopt project.lsopt` on the cluster. This is done from the command line.
5. After completion of the LS-OPT run, execute `lsopt project.archive.lsopt` to create a file `lso_pack_h.tar.gz` containing the entire run database.
6. Unzip `lso_pack_h.tar.gz` on the Windows machine to do the post-processing.



*Figure H - 1: Archive database dialog*

## H.5 Using an external queuing or job scheduling system

### H.5.1 Installation

The LS-OPT Queuing Interface interfaces with load sharing facilities (e.g. LSF<sup>14</sup> or LoadLeveler<sup>15</sup>) 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 (load-sharing) facility the following binary files are provided in the `LSOPT_EXE` directory which un-tars (or unzips) from the distribution during installation of LS-OPT:

```
LSOPT_EXE/wrapper
LSOPT_EXE/runqueuer
```

<sup>14</sup> Registered Trademark of Platform Computing Inc.

<sup>15</sup> Registered Trademark of International Business Machines Corporation

The runqueueer 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 runqueueer 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 runqueueer) is running. Set this if the wrapper has trouble connecting back to runqueueer.

`LSOPT_PORT` : TCP/IP port runqueueer listens on for remote connections

The runqueueer 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 Variables* settings in the LS-OPT GUI (see Section H.11 ) are set by the scheduler. The scheduler runs runqueueer, and runqueueer 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 runqueueer. However, the `LSOPT_HOST` value may be set through *Environment Variables* or through `.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 runqueueer knows the right setting.

## 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 solver command line whereas the two user variables  `LSDYNA971_MPP` and `LSOPT_WRAPPER` are defined and stored as special input entities (see Section H.11 ). 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.11 (Passing Environment Variables through LS-OPT).

```
# 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
#

```

## 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 `runqueueer` to the solver command and executes `runqueueer` which runs the `submit_pbs` script.
  - o The `runqueueer` sets the variables `LSOPT_HOST` and `LSOPT_PORT` locally.
  - o 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 13.3. The Progress dialog is shown below, Figure H - 2, 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/runqueueer

PORT=56984

JOB=LoadLeveler

llsubmit: The job "1/1.1" has been submitted.

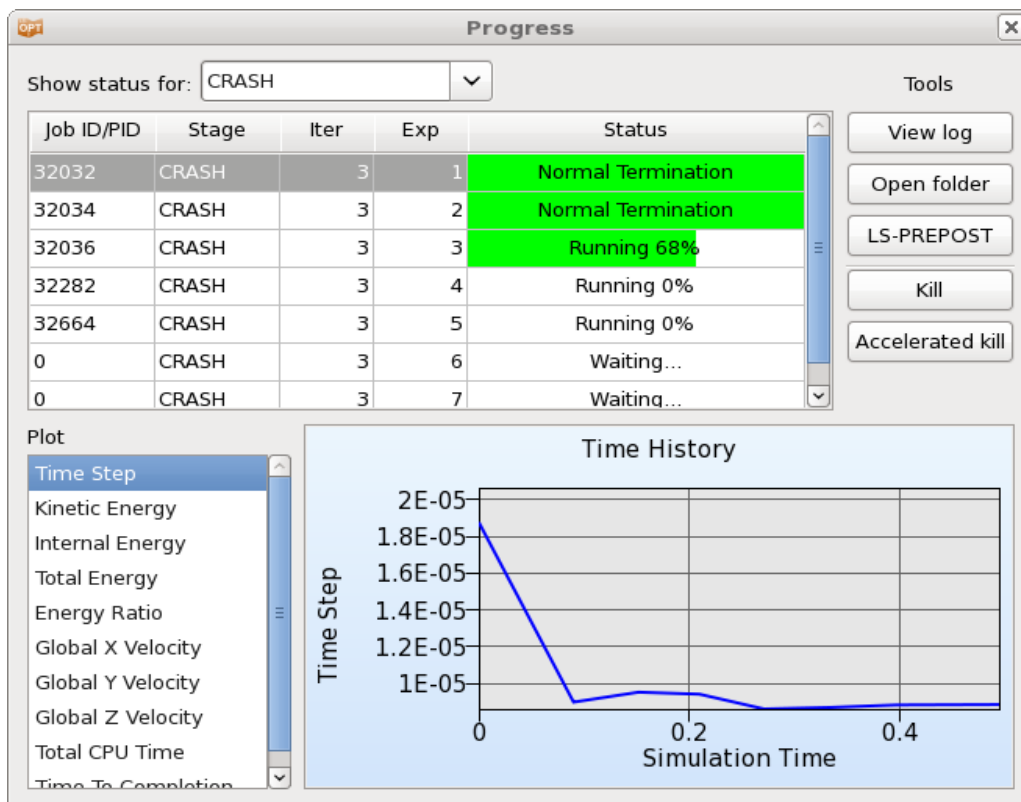
/home/jim/LSOPT_EXE/Xrapper: Command not found.

Finished with directory

/home/jim/LSOPT/4.1/optQA/QUEUE/EX4a_remote/remote/1/1.1

```

- 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 5.4.5). A log of the database extraction is also provided in the `job_log` file.



**Figure H - 2: 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 - 3: Opened job\_log file using Progress dialog option*

## H.7 User-defined and Blackbox queuing systems

If the queuing system you want to use is not available in the queuing interfaces list in the **Stage** dialog, Section 5.4.2, there are two options to interface with any queuing system by using the User-Defined or the Blackbox option. Blackbox can be used when the computers running the jobs are separated from the computer running LS-OPT by means of a firewall.

The key differences between User-defined and Blackbox are:

1. For Blackbox, it is the responsibility of the queueing system or the user provided scripts to transfer input and output files for the solver between the queueing 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.

4. 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 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 <i>N/M</i>	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 failed. This is only to be used when the underlying queuing system reports some kind of problem. Hence, a solver that has terminated in error does not have to be detected by the <code>LsoptJobCheck</code> script.
FINISHED	The job has completed 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 LSOPTui 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 be 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.7.4 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 Honda queuing system

The Honda queuing system interface is based on the Blackbox queuing system, but is dedicated to the particular needs of this system.

### H.8.1 Mechanics of the Honda queuing process

The queuing system generates a status file for which an environment variable has been defined in LS-OPT as:

```
$HONDA_STATUSFILE
```

The status file is the output of the PBS queue check command. During the initialization phase, LS-OPT checks whether this variable setting points to a valid file. If it does not, LS-OPT terminates before starting the scheduler, and prints a standard LS-OPT-style error message.

The line which marks the fields in the status file is used to determine how to parse the file; this line has the form "`----- - ---- ...`". Fields are extracted based on this line which consists solely of space and dash characters. The following fields are used:

Field	Description
4	name
6	status: 'R' for running or 'Q' for queued

---

10	total wall clock time allowed
11	total wall clock time consumed.

---

Fields 10 and 11 are used to set the progress indicator. If the indicator ever reaches 100%, then it will terminate due to total wall clock time restrictions.

If a job cannot be found in the status file, then it is assumed to be dead. The job status entry is not looked for until a minimum of 3 seconds after the job has been started. A status file is searched for a particular job status entry only if the status file has a modification time that is later than the start time of the job.

Since there is no way to determine the exit status of a job by looking only at this status file, the determination of the final exit status depends on whether or not the job is an LS-DYNA job. If the job is an LS-DYNA job, then the message file is parsed for the status statements "N o r m a l" and "E r r o r" termination. If no message file is found *10 seconds* after the job is no longer listed in the status file, then we assume an error termination.

If the job is a non-LS-DYNA job, then `LsoptJobCheck` (see Section H.7 ) is executed just once after the job no longer appears in the status file. `LsoptJobCheck` should print either (a) FINISHED or (b) ERROR in order to communicate the final exit status. If `LsoptJobCheck` cannot be found or cannot be executed, then ERROR is assumed. The job log file will contain a message indicating any problem that may exist which prevents `LsoptJobCheck` from being run.

The HONDA queued jobs do not use `LsoptJobDel` as defined in the Blackbox queuing selection. Jobs are deleted using the standard PBSPro `qdel` command.

Various statements concerning how status information is gathered are logged to the job log files. These are:

1. Job status for LSDYNA jobs found in 'messag' file:

```
[HONDA] Termination status found in 'messag' file
[HONDA] exact termination statement
```

2. The job status line for the current job found in `$HONDA_STATUSFILE` is saved:

```
[HONDA] status line
```

3. The job is assumed finished if there is no status line found:

```
[HONDA] Job 23551 not found in STATUS file - assuming job is finished.
```

4. Indication that `LsoptJobCheck` is run at the end of a non-LS-DYNA job:

```
[HONDA] Non LS-DYNA job. Running LsoptJobCheck to determine exit status.
```

5. Status returned from `LsoptJobCheck`.

```
[HONDA] Job finished - LsoptJobCheck reports normal termination
[HONDA] Job finished - LsoptJobCheck reports error termination
```

Any errors while gathering status information are logged to the `job_log` files..

6. Missing message file after LSDYNA terminates:

```
[HONDA] Failed to find 'messag' file while FINISHING.
[HONDA] Assuming ERROR termination for LSDYNA job.
```

7. Found no termination status statement in message file

```
[HONDA] Found no termination status in 'messag' file
[HONDA] Assuming ERROR termination for LSDYNA job.
```

#### 8. HONDA\_STATUSFILE variable not set

```
[HONDA] *** Error $HONDA_STATUSFILE not set.
```

#### 9. Could not open \$HONDA\_STATUSFILE

```
[HONDA] *** Error Failed to open $HONDA_STATUSFILE=pbsq_status
```

#### 10. LsoptJobCheck script not found for non-LSDYNA job

```
[HONDA] *** Error LsoptJobCheck cannot be found.
[HONDA]     Assuming error termination for non-LSDYNA job.
```

#### 11. LsoptJobCheck script did not print either (a) FINISHED or (b) FAILED.

```
[HONDA] *** Error LsoptJobCheck did not return a valid status.
[HONDA]     Assuming error termination for non-LSDYNA job.
```

If \$HONDA\_STATUSFILE is not updated in a timely fashion, then the scheduler can hang forever, never moving forward. A message is passed to LS-OPT through the communication socket if this happens:

```
*** Warning HONDA_STATUSFILE out of date by more than 5 minutes
*** Job progress monitoring suspended until next update
```

Even though the status file is checked before starting the scheduler, it is still possible for file errors to occur. These are also sent directly to LS-OPT.

```
*** Error $HONDA_STATUSFILE not set
*** Error Failed to open $HONDA_STATUSFILE=pbsq_status
```

## H.9 Abnormal termination and retrying the job submission

### H.9.1 User-defined abnormal termination

It may be prudent to retry job submissions for certain types of abnormal termination. For this purpose, the user can specify an `A b n o r m a l` 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 `A b n o r m a l` signal should be sent to standard output from the simulation script.

### H.9.2 Troubleshooting

1. 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 accessible from the respective **Stage** box LED, Section 13.3.

Please attach the log file (`lsopt_output`) when emailing [support@lstc.com](mailto:support@lstc.com).

2. Make sure that the permissions are set for the executables and submission script.
3. Check all paths to executables e.g. "wrapper", etc. No diagnostic can detect this problem.



4. 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`)).
5. *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.\* and history.\* files are recovered from the remote disk automatically. Other files can be recovered using the feature in Section 5.4.5 .
6. *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.

## H.10 Enabling LSTCVM job proxy support

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. LSTCVM is available for download at <http://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 5.4.3. 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 5.4.4) 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 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.11.

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.11.1 .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.11.2 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/lstdyna-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 5.4.4.

#### *.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 5.4.4. 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.

# 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 way 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 is 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 **HONDA queuing** option;
4. Indirectly using the **BLACK BOX queuing** option.

There are LS-DYNA and non LS-DYNA 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 10 minutes. The SIGKILL method is used if LS-DYNA does not respond to SIGTERM within 15 seconds. If LS-DYNA does not respond shortly afterwards, then the job is abandoned.
- Non LS-DYNA jobs are killed using signals, starting with SIGINT (or SIGTERM). If that fails after about 10 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 runqueuer/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.

- If the remote job is an LS-DYNA job, then a D3KIL file is created on the remote side. If LS-DYNA fails to exit after about 10 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 15 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 HONDA queuing option):**

This queuing option was created to work in environments where TCP/IP connections from compute clusters back to LS-OPT were either impractical, undesirable, or not allowed. Commands are executed using user-supplied queuing scripts to run the 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 the queuing system job delete command. Queuing scripts can hang for a time before finishing. Therefore, we wait for 2 minutes to find the job name/id in the script output. After that we abandon the job.

*NOTE:* There are times when an LsoptJobCheck script may be run. If this script is busy when the job is killed, then we wait indefinitely for the script to complete. That way we prevent <defunct> processes.

### I.1.4 Case 4 (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 LSKILLJOB.

This is a program supplied with LS-OPT for the purpose of killing jobs managed by the **lscheduler** process. Jobs may be killed in bulk or killed individually using this program. The **lskilljob** program resides in the LS-OPT installation directory, along with the other executables.

### I.2.1 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 LSKILLJOB command line utility.

The LSKILLJOB program is a separate executable located in the main directory of the LS-OPT distribution. The syntax to invoke the bulk kill option using LSKILLJOB is:

```
lskilljob -kill
```

All three programs contact the LSCHEMULER job scheduling process in order to kill jobs. The LSKILLJOB program must be able to contact the LSCHEMULER process in order to kill jobs; so it imperative that LSKILLJOB is run in the main project directory for the currently running LS-OPT process. LSKILLJOB consults a file named "lsopt.control" for information about how to connect to the LSCHEMULER process. If "lsopt.control" cannot be found, or if LSCHEMULER is not currently running, then LSKILLJOB prints the following error message and then terminates:

```
Kill request status:Failed to open lscheduler
```

### I.2.2 Kill One Job

A single job may be killed using LS-OPTui, or using the standalone LSKILLJOB program. The basic kill job option requires a job number and no other command line options. As described above, the LSKILLJOB program must be run from the main LSOPT project directory

```
lskilljob job_number
```



There are several errors which can occur during such a request, and by are diagnosed by the following LSKILLJOB error messages

Kill request status:Job not running

Kill request status:Invalid parameter

Kill request status:Target job cannot be killed,  
or job id out of range

Kill request status:Failed to open scheduler

If the command succeeds (which does not mean that the job is immediately killed,) then you should see a success message

Kill request status:Success

This indicates the that the job number is a valid, running job, and that the kill operation has been initiated.

## **I.3 Kill Level**

Whenever a job is killed using the bulk kill option, or the LSKILLJOB kill described above, the kill operation proceeds in levels or stages. For LS-DYNA there are three stages. For some 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 SIGKILL 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 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.3.4 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.3.6 All HONDA Queued Jobs**

1. If the HONDA 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.
2. If the HONDA queuing script has failed to return a valid job id or job name for 120 seconds, then the job is abandoned.
3. If an LsoptJobCheck script is in the process of determining a final exit status for the job, then we ignore the kill in favor of waiting for the job check script to finish.
4. If the HONDA queuing script has returned a valid job id or job name, then the queuing delete command is executed, and the job is abandoned.
5. If the HONDA queuing script has been executed, but still has not returned a valid job id or job name, then 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 10 minutes in the first stage (level 0) before proceeding to level 1. It then executes level 1 kill operations, and waits only 15 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 is 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,

```
lskilljob job_number 1
```

will cause the kill routines to spend only 15 seconds waiting for level 0 kill operations to succeed before moving to level 1 kill operations. LSHEDULER will then try one level 1 kill operations, and wait 15 seconds before trying the final 'sure-kill' level 2 kill operations. Executing

```
lskilljob job_number 2
```

will cause the kill routines to spend only 15 seconds waiting for level 0 kill operations to succeed before moving to the final level kill operations (may be level 2 or, in some cases, may be level 1.)

If you execute LSKILLJOB using level 1, and execute it again immediately afterwards, then the wait period for the level 0 kill is eliminated, and level 1 kill operations are executed immediately. Executing LSKILLJOB immediately again then aborts the 15-second wait period for level 1 kill operations, and level 2 kill operations are immediately executed. This will make the impatient user happy, but may surprise other users. This is by design in order to create a responsive command.

Once the kill level is elevated, then all subsequent LSKILLJOB commands work at the same or higher level according to kill level escalation rules, making it unnecessary to add the kill level argument again. The kill level cannot be lowered by using the option kill level command line option. And a kill operation cannot be aborted.

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. 10 minutes should be sufficient for nearly all explicit LS-DYNA runs.

## 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, or KILLED using the lskilljob command.

You can specify the final exit ERROR, ABORTED or KILLED as follows:

```
lskilljob job_number [level] error
lskilljob job_number [level] aborted
lskilljob job_number [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

```
F o r c e d   t e r m i n a t i o n
```

If a bulk kill is in effect when you kill the job using LSKILLJOB, 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 LSKILLJOB, 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 LSKILLJOB, then **you can also flag the job for restart**

```
lskilljob job_id [kill_level] restart [wait_time_seconds]
```

If you do not specify a wait time, then a default value of 0 is used, thereby instructing the LSCHEUDLER 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 LSCHEUDLER terminates, so that LSOPT can use the corrected results. To flag a completed job for restart, issue the LSKILLJOB command with the `-restart` option

```
lskilljob job_id -restart [wait_time_seconds]
```

The LSCHEUDLER process will restart the process as soon after the specified time wait as resources are available to start the job.

You may **cancel any pending restart** option up until the time the job actually starts. This is done using the `cancel` option for the LSKILLJOB command

```
lskilljob job_id -restart job_to_restart cancel
```

This cancels any pending restart operation initiated with the LSKILLJOB command. This command does not cancel other types of pending restart events caused by an “A b o r t e d” termination status.

You may **modify a pending restart wait time** before the job has restarted by issuing the command

```
lskilljob job_id -restart wait_time_seconds
```

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 LSKILLJOB will report an error

Restart status:Job is running, use kill with restart option

Other possible returns are

Restart status:Success

Restart status:Cannot restart, job never started

Restart status:Job id out of range

Restart status:Already flagged for retry (no action taken)

Restart status:Invalid argument

Restart status:Failed to open lscheduler

Restart status:I/O error while negotiating with lscheduler

*Remarks on Restarting Jobs:*

- Restart events should be considered transient, persisting only during the time that LSCHEUDLER 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.
- In future versions, the user will be able to schedule a restart for a job, even if the LSCHEUDLER is not running. Such restart events will be persistent.
- A job which has never been started cannot be flagged for restart.
- Jobs which are running cannot be flagged for restart without also issuing a kill operation.

# 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. The format is x.y.z, where x.y is the LS-OPT major and minor version numbers. z is the file format revision number, which is 0 for trunk, and >=1 for released versions. When file format revision number is 0, the DTD should be provided in the file itself.

### Element's model:

([head?](#), [distributions?](#), [filehistories?](#), [variables?](#), [varcorrelations?](#), [samplings?](#), [evalmeta?](#), [composites?](#), [objectives?](#), [constraints?](#), [schedulerconfig?](#), [resources?](#), [task](#))

**<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	<i>Match the NMTOKEN rules.</i>	Required	Name for this metadata entry (such as "author")

This element is always empty.

**<distributions>** Child of [lsoptproject](#)  
Container element for the defined distributions

### Element's model:

[\(distribution\\*\)](#)

&lt;distribution/&gt;

Child of [distributions](#)

Distribution declaration

## &lt;distribution&gt;'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	<i>Match the ID rules.</i>	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	Required	Distribution type: uniform, normal, truncated_normal, lognormal, exponential, weibull, beta, binomial, geometric, poisson, erlang, rayleigh, triangular, gumbel, frechet, user_pdf, user_cdf
upper	CDATA	Implied	Upper bound for uniform, truncated_normal

This element is always empty.

&lt;filehistories&gt;

Child of [Isoptproject](#)

Container element for the defined distributions

**Element's model:**



[\(filehistory\\*\)](#)**<filehistory>** Child of [filehistories](#)

Declaration of history on file. The name of the file is given as the contents of the tag

**<filehistory>'s attributes**

Name	Values	Default	Description
name	<i>Match the ID rules.</i>	Required	Name for this file history, prefixed with "x_" to be unique in the global namespace

**<variables>** Child of [Isoptproject](#)

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.

**<variable>'s attributes**

Name	Values	Default	Description
dist	<i>Match the IDREF rules.</i>		Distribution for noise variables.
global	yes, no	yes	If set to no, variable will only be active for the samplings that has it listed in its myvars element. This attribute does not affect the relationship between stages and variables
max			Maximum value for continuous variables.
min			Minimum value for continuous variables.
name	<i>Match the ID rules.</i>	Required	Variable name, prefixed with "x_" to be unique in the global namespace
range			Initial range for variables
saddle	minimize, maximize	minimize	Saddle direction
sampling	continuous, discrete	continuous	continuous/discrete sampling for discrete vars
type	continuous, discrete, noise, constant, dependent, userdefined, string	continuous	Variable type.

value

Starting value for continuous/discrete.  
Constant value for constants.

**Element's model:**(#PCDATA | [dvalue](#))\*

&lt;dvalue&gt;

Child of [variable](#)

Single value for discrete variable

&lt;varcorrelations&gt;

Child of [Isoptproject](#)

Container element for the varcorrelation

**Element's model:**

(varcorrelation\*)

&lt;varcorrelation&gt;

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	<i>Match the ID rules.</i>	Required	.
var2	<i>Match the ID rules.</i>	Required	
corr	CDATA	Required	Value of the correlation between the variables var1 and var2.

&lt;samplings&gt;

Child of [Isoptproject](#)

Container element for the different samplings

**Element's model:**([sampling](#)\*)

&lt;sampling&gt;

Child of [samplings](#)

Defines a single sampling design

**<sampling>'s attributes**

Name	Values	Default	Description
name	<i>Match the ID rules.</i>	Required	Design id, prefixed with "e_" (for experiment) to be unique in the global namespace.

**Element's model:**

([design?](#), [metamodel?](#), [alternate?](#), [histmetamodel?](#), [myvars?](#), [stages?](#), [importresults?](#), [checkpoints?](#))

<design>

Child of [sampling](#), [alternate](#)

Specifies the design for this sampling of points

<design>'s attributes

Name	Values	Default	Description
duplicate	<i>Match the IDREF rules.</i>	Implied	Name of sampling to clone design from
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.
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
seed	CDATA	Implied	Optional random number seed for methods that use randomness
type	one, dopt, koshal_linear, koshal_quad, composite, userdefined, plan, random, latin_hypercube, monte_carlo, latin_hypercube_central_point, latin_hypercube_generalized, maximin_lhd_permute, maximin_lhd_subinterval, maximin_distance, full_factorial, duplicate,	Required	Single Point D-Optimal Linear Koshal Quadratic Koshal Central Composite User defined sampling Plan Random placement Latin Hypercube Monte Carlo, Space Filling 0 Space Filling 1 Space Filling 2 Space Filling 3 Space Filling 4 Space Filling (Algorithm 5) Full Factorial Designs Clone points from another sampling. The duplicate attribute must be set if this type is used.

	piercing, domainpiercing		Space filling of Pareto Frontier Space Filling of Pareto region, not used
update	on, off	Implied	If set, causes space-filling method to consider points from previous iterations. Implies metamodel update.

**Element's model:**

([basis?](#))

<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.

<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)
perturbation	CDATA	0.01	Perturbation relative to design space for numerical sensitivity
stype	numerical, analytical	numerical	Subtype for sensitivity: numerical, analytical
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?](#), [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. <b>linear</b> , <b>quadratic</b> or <b>radial basis functions</b> .

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	median	Averaging function to use, <b>mean</b> or <b>median</b>
discard	CDATA	0	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, rmterror	gcv	Topology Selection Criterion: Leave-one-out, GCV-ratio, Noise variance
regressionalg	levenberg, bfgs, rprop	rprop	Gradient algorithm: Levenberg-Marquardt, Broyden-Fletcher-Goldfarb-Shanno, Resilient backpropagation
rmterror	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. 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.
transfer	linear, sigmoid, gauss, hmq	sigmoid	Transfer function of intermediate layers: Linear, Sigmoid, Gauss, Hardy's Multi Quadrics

This element is always empty.

<rbfopts/>

Child of [metamodel](#)

Special options controlling Radial Basis Function networks

**<rbfopts>'s attributes**

Name	Values	Default	Descriptions
optimize	gcv, gcv_ratio, rmterror	gcv	Topology Selection Criterion: Leave-one-out, GCV-ratio, Noise variance
rmterror	CDATA	-1.0	Threshold of RMS training error
transfer	linear, sigmoid, gauss, hmq	sigmoid	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.

<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	<i>Match the ID</i>	Required	Stage name, prefixed with "s_" to be unique in

type *rules.* the global namespace  
*Match the* Required Stage type, dyna960, own ...  
*NMTOKEN*  
*rules.*

**Element's model:**

([guidata?](#), [runin?](#), [dependson\\*](#), [fileops?](#), [command?](#), [dbfile?](#), [inputfile?](#), [outputfile?](#), [appendfile?](#), [resource?](#), [envvars?](#), [extrainfiles?](#), [dynaoptions?](#), [scheduling?](#), [myvars?](#), [histories?](#), [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
<a href="#">x</a>	CDATA	Required	X position, floating point value
<a href="#">y</a>	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	<i>Match the IDREF rules.</i>	Required	The name of the stage to share directory with, including s_ prefix

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	<i>Match the IDREF rules.</i>	Required	Stage name, including s_ prefix



**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

**<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

**<dynaoptions/>** Child of [stage](#)  
 LS-DYNA specific settings

**<dynaoptions>'s attributes**

Name	Values	Default	Description
checkout	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.

**<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
------	-------	----------	------------------------------

**<extrainfiles>**Child of [stage](#)

Collection of extra input files

**Element's model:**[\(inputfile\\*\)](#)**<scheduling>**Child of [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, lscvm	none	Which proxy to use for job submission: none or lscvm
queuer	lsf, loadleveler, pbs, nqs, user, aqs, slurm, blackbox, msccp, pbspro, honda, sge	Implied	Enable the use of a queuing system. Valid values are: lsf, loadleveler, pbs, nqs, user, aqs, slurm, blackbox, msccp, pbspro, honda, sge. 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	14400	

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:

([recover\\*](#), [resourceref\\*](#))

**<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\\*](#))

**<varref/>** Child of [myvars](#)

A reference to a variable

#### **<varref>'s attributes**

Name	Values	Default	Description
src	<i>Match the IDREF rules.</i>	Required	Variable name, including "x_" prefix for this reference

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
dynacase	CDATA	Implied	LS-DYNA case to use for extraction
name	<i>Match the ID rules.</i>	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. More might be added in the future if we'd want full markup for histories

**<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
dynacase	CDATA	Implied	LS-DYNA case to use for extraction
metamodel	on, off	on	Set to <b>off</b> to disable metamodel generation
name	<i>Match the ID rules.</i>	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	definition	<b>definition</b> for classic LS-OPT definition strings. <b>expression</b> for expressions and <b>matrix</b> for matrix expressions. More might be added in the future if we'd want full markup for responses

**<importresults>**Child of [sampling](#)

If present, activates importing of user results from csv file given as element contents

<checkpoints> Child of [sampling](#)

If present, activates checkpoints with csv file given as element contents

<composites> Child of [Isoptproject](#)

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	<i>Match the IDREF rules.</i>	Implied	Entity for stddev-type composite
name	<i>Match the ID rules.</i>	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

#### 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	<i>Match the IDREF rules.</i>	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

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	<i>Match the ID rules.</i>	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 [Isoptproject](#)

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	<i>Match the IDREF rules.</i>	Required	Reference to a response or composite to use as objective, including prefix
weight	CDATA	1	Weight for this objective

This element is always empty.

**<constraints>** Child of [Isoptproject](#)

Collection of constraints

**<constraints>'s attributes**

Name	Values	Default	Description
------	--------	---------	-------------

scaling on, off	off		Set to "on", changes defaults for bound divisor
-----------------	-----	--	---

**Element's model:**[\(constraint\\*\)](#)

&lt;constraint&gt;

Child of [constraints](#)

Constraint for a single entity. Contains one or two bound elements of different types

## &lt;constraint&gt;'s attributes

Name	Values	Default	Description
sampling	stay, move, move_start	stay	Controls how sampling should be affected by this constraint. Known as "move" in GUI.
src	<i>Match the IDREF rules.</i>	Required Reference	Reference to a response or composite to use as constraint

**Element's model:**[\(\(bound, bound?\)?\)](#)

&lt;bound/&gt;

Child of [constraint](#), [dynastats](#)

A component of a constraint.

## &lt;bound&gt;'s attributes

Name	Values	Default	Description
divisor	CDATA	Implied	Defaults to "1" if scaling is off, or <b>limit</b> 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.

&lt;evalmeta&gt;

Child of [Isoptproject](#)

If present, activates evaluation of metamodel at specific points from the csv file given as element contents

&lt;schedulerconfig/&gt;

Child of [Isoptproject](#)

Global scheduler options

## &lt;schedulerconfig&gt;'s attributes

Name	Values	Default	Description
------	--------	---------	-------------



globalconcurrencylimit	CDATA	Implied	Maximum 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 [Isoptproject](#)

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

This element is always empty.

<task> Child of [Isoptproject](#)

Task control section

**<task>'s attributes**

Name	Values	Default	Description
method	none, metamodel_opt, mc, metamodel_mc, rbd, metamodel_form, direct_opt, doestudy, exp_design	Required	<p>Main task. <i>none</i> 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.</p> <p>Task metamodel_form (First Order Reliability Method) will:</p> <ul style="list-style-type: none"> <li>○ Construct the metamodels as prescribed by the user. If the metamodels already exists, then they won't be recreated.</li> <li>○ Conduct a FORM analysis for every constraint using the</li> </ul>

metamodels.

The following are computed in a FORM analysis:

- The most probable point (see Section **24.4.6**)
- The probabilities of exceeding the bounds on the constraint
- 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:

1. Statistical distributions associated with the design variables
2. Probabilistic bounds on the constraints.

Theoretical concerns are discussed in Section **24.4.7**.

### Element's model:

([optopts?](#), [optalg?](#), [runbaseline?](#), [addmmnoise?](#), [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	domainreduction	Strategy for optimization: singlestage, sequential, domainreduction (not applicable for <b>direct_opt</b> method)
verify	CDATA	1	Number of verification runs. (Clamped to [0,1] if MOO, not applicable for <b>direct_opt</b> 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 0.01	Implied	Response Accuracy Tolerance
designtol	CDATA 0.01	Implied	Design Change Tolerance
iterations	CDATA Implied	Implied	Max number of iterations
objecttol	CDATA 0.01	Implied	Object Function Tolerance
toloperator	and, or	and	Tolerance operator: <b>and</b> , <b>or</b> .

This element is always empty.

<srsmopts/>

Child of [optopts](#)

Options for Domain Reduction strategy

**<srsmopts>'s attributes**

Name	Values	Default	Description
eta	CDATA Implied	Implied	Proximity Zoom parameter
freezefromiter	CDATA Implied	Implied	Freeze Range from iteration
gamma	CDATA Implied	Implied	Oscillation Contraction parameter
psi	CDATA Implied	Implied	Panning Contraction parameter
resetiter	CDATA Implied	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	Implied	Type of algorithm: Leaping Frog, Genetic, Adaptive Simulated Annealing, Particle Swarm Optimization

**Element's model:**

([gaopts?](#), [asaopts?](#), [psoopts?](#), [lfopopts?](#), [paretoopts?](#))

<gaopts/>

Child of [optalg](#)

## Options for Genetic Algorithm

## &lt;gaopts&gt;'s attributes

Name	Values	Default	Description
binarymprob	CDATA	Implied	Mutation probability for binary values
binaryxop	sing_pt, uniform	sing_pt	Binary value crossover type: <b>single point</b> or <b>uniform</b>
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
popsiz	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
tourntsize	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

## &lt;asaopts&gt;'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

## &lt;paretoopts&gt;'s attributes

Name	Values	Default	Description
crthres	CDATA	Implied	Consolidation Ratio Change
dhvthres	CDATA	Implied	Normalized hypervolume change threshold
generationgap	CDATA	Implied	Generation Gap
termcriterion	maxfuncgen, fixedratio, ratiochange, volchange	maxfuncgen	Termination Criterion
utilityfraction	CDATA	Implied	Utility fraction cutoff

This element is always empty.

<psoopts/>

Child of [optalg](#)

## Options for Particle Swarm Optimization

## &lt;psoopts&gt;'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

## &lt;lfopopts&gt;'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.

`<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. Default number of bins used if set to 0. Default number of bins is 20 for more than 500 observations and 8 otherwise.
distbounds	on, off	off	Whether the bounds must be enforced for the probabilistic component of the variables
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
------	--------	---------	-------------

points CDATA 10000 Number of points for Integration

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	<i>Match the IDREF rules.</i>		Implied Name of the sampling for runjobs, rerunfailed and extract assignments. If absent, all stages for the given iteration are processed.
stage	<i>Match the IDREF rules.</i>		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, rerunfailed, extract, import, buildmm, evalmm, optimize, gather, gatherext, clean	full	Type of assignment. Default "full" means normal run

**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
iteration	CDATA	1	Iteration number to use
order	none, linear, quadratic	none	Set to linear or quadratic to enable use of metamodel.
stage	<i>Match the IDREF rules.</i>	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	<i>Match the IDREF rules.</i> Required Reference to a composite or response, complete with "x_" prefix.		

This element is always empty.

**<dsd3plot>** Child of [dynastats](#)

The precense 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	<i>Match the IDREF rules.</i>	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

**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.

**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.  $df/dx_i$ .

**Design variable.** An independent design parameter which is allowed to vary in order to change the design. Symbolized by ( $x_i$  or  $\mathbf{x}$  (vector containing several design variables)).

**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.

**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  $|\mathbf{X}^T \mathbf{X}|$  of the least squares formulation is maximized.

**DSA.** Design sensitivity analysis.

**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  $x_n$ .  $\mathbf{s} = [df/dx_i]$ . 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.

**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.

**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  $\Phi$ .

**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.

**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.

**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 quantity.** See *response*.

**Response Surface.** A mathematical expression which relates the response variables to the design parameters. Typically computed using statistical methods.

**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*).

**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*.

**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. 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.

**Trade-off curve.** A curve constructed using *Pareto optimal* designs.

**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 in which the LS-OPT input files reside and where LS-OPT output is generated. Same as Project Home directory. See also *Run directory*.



