LS-OPT® User's Manual

A DESIGN OPTIMIZATION AND PROBABILISTIC ANALYSIS TOOL FOR THE ENGINEERING ANALYST

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

June, 2008

Version 3.3

Mailing address:
Livermore Software Technology Corporation
2876 Waverley Way
Livermore, California 94551

Support Address:
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, California 94551

FAX: 925-449-2507
TEL: 925-449-2500
EMAIL: sales@lstc.com

June 2, 2008

Contents

Preface to Version 1

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

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

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

Nielen Stander
Livermore, CA
August, 1999

Preface to Version 2

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

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

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

Nielen Stander, Ken Craig, Trent Eggleston and Willem Roux
Livermore, CA
January, 2003

Preface to Version 3

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

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

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) and Professor Ken Craig (Department of Mechanical Engineering, University of Pretoria, South Africa), significant contributions have been made by Dr. Daniel Hilding and Mr. David Björkevik of Engineering Research AB (Linköping) as well Dr.-Ing. Heiner Müllerschön, Dipl.-Ing. Marko Thiele and Dipl.-Math. Katharina Witowski of DYNA*more* GmbH, Stuttgart, Germany.

Nielen Stander, Willem Roux and Tushar Goel
Livermore, CA
May, 2008

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 in making such a decision. 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 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 also been touted as design improvement methods. In a typical approach, the user iteratively selects the best design results of successive stochastic simulations to improve the design. These design methods, being dependent on chance, are generally not as efficient as response surface methods. However, an iterative design improvement method based on stochastic simulation is available in LS-OPT.

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® manual consists of three parts. In the first part, the Theoretical Manual (Chapters 2 through 6), the theoretical background is given for the various features in LS-OPT. The next part is the User's Manual (Chapters 7 through 20), which guides the user in the use of LS-OPT*ui*, the graphical user interface. These chapters also describe the command language syntax. The final part of the manual is the Examples section (Chapter 22), where eight examples are used to illustrate the application of LS-OPT to a variety of practical applications. Appendices contain interface features (Appendix A and Appendix B), database file descriptions (Appendix C), a mathematical expression library (Appendix D), advanced theory (Appendix E), a Glossary (Appendix F) and a Quick Reference Manual (Appendix G).

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 7 (The design optimization process). The Theoretical Manual (Chapters 2 through 6) serves mainly as an in-depth reference section for the underlying methods. The Examples section is included to demonstrate the features and capabilities and can be read together with Chapters 7 to 22 to help the user to set up a problem formulation. The items in the Appendices are included for reference to detail, while the Quick Reference Manual provides an overview of all the features and command file syntax.

Links can be used for cross-referencing and will take the reader to the relevant item such as Section 12.4.3, Reference [4] or Figure 3-5 (just click on any of the afore-mentioned references).

THEORETICAL MANUAL

2. Response Surface Methodology

2.1 Introduction

An authoritative text on Response Surface Methodology (RSM) [1] defines the method as "a collection of statistical and mathematical techniques for developing, improving, and optimizing processes." Although an established statistical method for several decades [2], it has only recently been actively applied to mechanical design [3]. Due to the importance of weight as a criterion and the multidisciplinary nature of aerospace design, the application of optimization and RSM to design had its early beginnings in the aerospace industry. A large body of pioneering work on RSM was conducted in this and other mechanical design areas during the eighties and nineties [3]-[6]. RSM can be categorized as a Metamodeling technique (see Chapter 3 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.

2.1.1 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 $\boldsymbol{x}$. The exact functional relationship between these quantities is

$$y = \eta(\boldsymbol{x}) \tag{2.1-1}$$

The exact functional relationship is now approximated (e.g. polynomial approximation) as

$$\eta(\boldsymbol{x}) \approx f(\boldsymbol{x}) \tag{2.1-2}$$

The approximating function f is assumed to be a summation of basis functions:

$$f(\boldsymbol{x}) = \sum_{i=1}^{L} a_i \phi_i(\boldsymbol{x}) \tag{2.1-3}$$

where L is the number of basis functions ϕ_i used to approximate the model.

The constants $\boldsymbol{a} = [a_1, a_2, \ldots, a_L]^T$ have to be determined in order to minimize the sum of the square error:

$$\sum_{p=1}^{P} \left\{ [y(\boldsymbol{x}_p) - f(\boldsymbol{x}_p)]^2 \right\} = \sum_{p=1}^{P} \left\{ [y(\boldsymbol{x}_p) - \sum_{i=1}^{L} a_i \phi_i(\boldsymbol{x}_p)]^2 \right\} \tag{2.1-4}$$

P is the number of experimental points and y is the exact functional response at the experimental points $\boldsymbol{x}_i$.

The solution to the unknown coefficients is:

$$\boldsymbol{a} = (\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X})^{-1} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{y} \tag{2.1-5}$$

where $\boldsymbol{X}$ is the matrix

$$\boldsymbol{X} = [X_{ui}] = [\phi_i(\boldsymbol{x}_u)] \tag{2.1-6}$$

The next critical step is to choose appropriate basis functions. A popular choice is the quadratic approximation

$$\phi = [1, x_1, \ldots, x_n, x_1^2, x_1 x_2, \ldots, x_1 x_n, \ldots, x_n^2]^T \tag{2.1-7}$$

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.

2.1.2 Factors governing the accuracy of the response surface

Several factors determine the accuracy of a response surface [1].

1. *The size of the subregion*
   For problems with smooth responses, the smaller the size of the subregion, the greater the accuracy. For the general problem, there is a minimum size at which there is no further gain in accuracy. Beyond this size, the variability in the response may become indistinguishable due to the presence of 'noise'.

2. *The choice of the approximating function*
   Higher order functions are generally more accurate than lower order functions. Theoretically, over-fitting (the use of functions of too high complexity) may occur and result in suboptimal accuracy, but there is no evidence that this is significant for polynomials up to second order [1].

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 [1] (very roughly).

2.1.3 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.

2.1.4 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 3.1 and 3.2.

2.2 Experimental design

Experimental design is the selection procedure for finding the points in the design space that must be analyzed. Many different types are available [1]. The factorial, Koshal, composite, *D*-optimal and Latin Hypercube designs are detailed here.

2.2.1 Factorial design

This is an 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 LSOPT, 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.

2.2.2 Koshal design

This family of designs are saturated for modeling of any response surface of order *d*.

First order model

For $n = 3$, the coordinates are:

$$\begin{matrix} x_1 & x_2 & x_3 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

As a result, four coefficients can be estimated in the linear model

$$\phi = [1, x_1, \ldots, x_n]^T \tag{2.2-1}$$

Second order model

For $n = 3$, the coordinates are:

$$\begin{matrix} x_1 & x_2 & x_3 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

As a result, ten coefficients can be estimated in the quadratic model

$$\phi = [1, x_1, \ldots, x_n, x_1^2, x_1 x_2, \ldots, x_1 x_n, \ldots, x_n^2]^T \tag{2.2-2}$$

2.2.3 Central Composite design

This design uses the 2^n factorial design, the center point, and the 'face center' points and therefore consists of $P = 2^n + 2n + 1$ experimental design points. For $n = 3$, the coordinates are:

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \\ -\alpha & 0 & 0 \\ 0 & -\alpha & 0 \\ 0 & 0 & -\alpha \\ -1 & -1 & -1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

The points are used to fit a second-order function. The value of $\alpha = \sqrt[4]{2^n}$.

2.2.4 D-optimal design

This method uses a subset of all the possible design points as a basis to solve

$$\max\left|\boldsymbol{X}^T\boldsymbol{X}\right|.$$

The subset is usually selected from an ℓ^n -factorial design where ℓ is chosen *a priori* as the number of grid points in any particular dimension. Design regions of irregular shape, and any number of experimental points, can be considered [7]. The experiments are usually selected within a sub-region in the design space thought to contain the optimum. A genetic algorithm is used to solve the resulting discrete maximization problem. See References [1] and [5].

The numbers of required experimental designs for linear as well as quadratic approximations are summarized in the table below. The value for the *D*-optimality criterion is chosen to be 1.5 times the Koshal design value plus one. This seems to be a good compromise between prediction accuracy and computational cost [7]. The factorial design referred to below is based on a regular grid of 2^n points (linear) or 3^n points (quadratic).

Table 2.2-1: 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

2.2.5 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 $\boldsymbol{S} = S_{ij}$ that randomly samples the entire design space broken down into P equal-probability regions:

$$S_{ij} = \left(\eta_{ij} - \zeta_{ij}\right)/P, \tag{2.2-3}$$

where $\eta_{1j},\ldots,\eta_{Pj}$ are uniform random permutations of the integers 1 through P and ζ_{ij} independent random numbers uniformly distributed between 0 and 1. A common simplified version of LHS has centered points of P equal-probability sub-intervals:

$$S_{ij} = \left(\eta_{ij} - 0.5\right)/P \tag{2.2-4}$$

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 $\boldsymbol{S}$ 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.

Maximin

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, Simulated Annealing (see Appendix E). 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 2.2.6 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:

$$CL_2^2 = (13/12)^n - \frac{2}{P}\sum_{i=1\ldots P}\prod_{j=1\ldots n}\left(1+\frac{|S_{ij}-0.5|}{2}-\frac{(S_{ij}-0.5)^2}{2}\right) + \frac{1}{P_2}\sum_{k=1\ldots P}\sum_{i=1\ldots P}\prod_{j=1\ldots n}\left(1+\frac{|S_{kj}-0.5|}{2}+\frac{|S_{ij}-0.5|}{2}-\frac{|S_{kj}-S_{ij}|}{2}\right). \quad (2.2\text{-}5)$$

2.2.6 Space-filling designs

In the modeling of an unknown nonlinear relationship, when there is no persuasive parametric regression model available, and the constraints are uncertain, one might believe that a good experimental design is a set of points that are uniformly scattered on the experimental domain (design space). *Space-filling* designs impose no strong assumptions on the approximation model, and allow a large number of levels for each variable with a moderate number of experimental points. These designs are especially useful in conjunction with nonparametric models such as neural networks (feedforward networks, radial basis functions) and Kriging, [8], [9]. Space-filling points can be also 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 pseudorandom sequences, which, particularly in higher dimensions, can lead to a clustering of points, which limits their uniformity. To find a good space-filling design is a

nonlinear programming hard problem, which – from a theoretical point of view – is difficult to solve exactly. This problem, however, has a representation, which might be within the reach of currently available tools. To reduce the search time and still generate good designs, the popular approach is to restrict the search within a subset of the general space-filling designs. This subset typically has some good 'built-in' properties with respect to the uniformity of a design.

The constrained randomization method termed *Latin Hypercube Sampling* (LHS) and proposed in [10], has become a popular strategy to generate points on the 'box' (hypercube) design region. The method implies that on each level of every design variable only one point is placed, and the number of levels is the same as the number of runs. The levels are assigned to runs either randomly or so as to optimize some criterion, e.g. so that the minimal distance between any two design points is maximized ('maximin distance' criterion). Restricting the design in this way tends to produce better Latin Hypercubes. However, the computational cost of obtaining these designs is high. In multidimensional problems, the search for an optimal Latin hypercube design using traditional deterministic methods (e.g. the optimization algorithm described in [11]) may be computationally prohibitive. This situation motivates the search for alternatives.

Probabilistic search techniques, *simulated annealing* and genetic algorithms are attractive heuristics for approximating the solution to a wide range of optimization problems. In particular, these techniques are frequently used to solve combinatorial optimization problems, such as the traveling salesman problem. Morris and Mitchell [12] adopted the simulated annealing algorithm to search for optimal Latin hypercube designs.

In LS-OPT, space-filling designs can be useful for constructing experimental designs for the following purposes:

1. The generation of basis points for the *D*-optimality criterion. This avoids the necessity to create a very large number of basis points using e.g. the full factorial design for large n. E.g. for n=20 and 3 points per variable, the number of points = $3^{20} \approx 3.5*10^9$.
2. The generation of design points for all approximation types, but especially for neural networks and Kriging.
3. The augmentation of an existing experimental design. This means that points can be added for each iteration while maintaining uniformity and equidistance with respect to pre-existing points.

LS-OPT contains 6 algorithms to generate space-filling designs (see Table 2.2-2). Only Algorithm 5 has been made available in the graphical interface. LS-OPT*ui*.

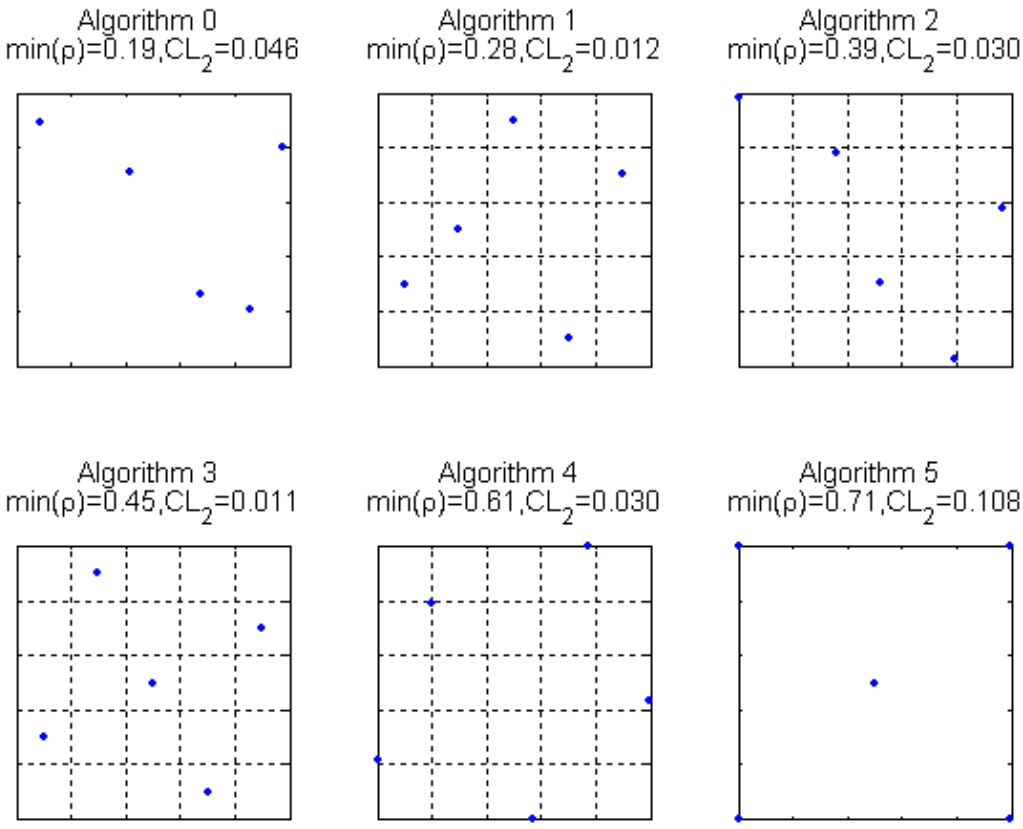


Figure 2-1: Six space-filling designs: 5 points in a 2-dimensional box region

Table 2.2-2: Description of space-filling algorithms

Algorithm Number	Description
0	Random
1	'Central point' Latin Hypercube Sampling (LHS) design with random pairing
2	'Generalized' LHS design with random pairing
3	Given an LHS design, permutes the values in each column of the LHS matrix so as to optimize the maximin distance criterion taking into account a set of existing (fixed) design points. This is done using *simulated annealing*. Fixed points influence the maximin distance criterion, but are not allowed to be changed by Simulated Annealing moves.
4	Given an LHS design, moves the points within each LHS subinterval preserving the starting LHS structure, optimizing the maximin distance criterion and taking into consideration a set of fixed points.
5	given an arbitrary design (and a set of fixed points), randomly moves the points so as to optimize the maximin distance criterion using simulated annealing (see Appendix E).

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 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:

$$S'_{ij} = S_{ij} + \xi \tag{2.2-6}$$

where ξ is a random number sampled from a normal distribution with zero mean and standard deviation $\sigma_\xi \in [\sigma_{\min}, \sigma_{\max}]$. In algorithm 4 it is required that both S'_{ij} and S_{ij} in Eq. (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:

$$\begin{aligned} T_{\max} &:= T_{\max} \times \Delta E \\ T_{\min} &:= T_{\min} \times \Delta E \end{aligned} \tag{2.2-7}$$

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 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:

1. Parameters for the cooling schedule:
   - scaling factor for the initial (maximal) temperature, $T_{\max}$, in (2.24)
   - scaling factor for the minimal temperature, $T_{\min}$, in (2.24),
   - damping factor for temperature, μ_T, in (Eq. (F.5), Appendix E),

- number of iterations at each temperature, ν_T (Appendix E).

2. Parameters that control the standard deviation of ξ in (2.23):
   - upper bound, σ_{max},
   - lower bound, σ_{min}.
3. Termination criteria:
   - maximal number of energy function evaluations, N_{it}.

2.2.7 Random number generator

The Mersenne Twister [13] is used in Neural Network construction and Monte Carlo, Latin Hypercube, Space Filling and *D*-Optimal point selection and probabilistic analysis. The Mersenne Twister (MT19937) is a pseudorandom number generator developed by Matsumoto and Nishimura and has the merit that it has a far longer period and far higher order of equidistribution than any other implemented generators. It has been proved that the period is 2^{19937}-1, and a 623-dimensional equidistribution property is assured. Features have been provided to seed the generator to enable sensitivity studies.

2.2.8 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. Therefore, once the first approximation has been established, all the designs will be contained in the new region of interest. This region of interest is thus defined by approximate bounds.
One way of establishing a reasonable set of designs is to move the points of the basis experimental design to the boundaries of the reasonable design space in straight lines connecting to the central design $\boldsymbol{x}_c$ so that

$$\boldsymbol{x}' = \boldsymbol{x}_c + \alpha(\boldsymbol{x} - \boldsymbol{x}_c) \tag{2.2-8}$$

where α is determined by conducting a line search along $(\boldsymbol{x} - \boldsymbol{x}_c)$.

This step may cause near duplicity of design points that can be addressed by removing points from the set that are closer than a fixed fraction (typically 5%) of the design space size.

The *D*-optimality criterion is then used to attempt to find a well-conditioned design from the basis set of experiments in the reasonable design space. *Using the above approach, a poor distribution of the basis points may result in a poorly designed subset.*

2.3 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.

2.3.1 Residual sum of squares

For the predicted response $\hat{y}_i$ and the actual response y_i, this error is expressed as

$$\varepsilon^2 = \sum_{i=1}^{P} (y_i - \hat{y}_i)^2 \tag{2.3-1}$$

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.

2.3.2 RMS error

The residual sum-of-squares is sometimes used in its square root form, ε_{RMS}, and called the "RMS error":

$$\varepsilon_{\text{RMS}} = \sqrt{\frac{1}{P}\sum_{i=1}^{P} (y_i - \hat{y}_i)^2} \tag{2.3-2}$$

2.3.3 Maximum residual

This is the maximum residual considered over all the design points and is given by

$$\varepsilon_{\max} = \max\left|y_i - \hat{y}_i\right|. \tag{2.3-3}$$

2.3.4 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 $\boldsymbol{X}$

$$\boldsymbol{X}_a(\boldsymbol{x}_p) = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{A}(\boldsymbol{x}_p) \end{bmatrix} \tag{2.3-4}$$

and solving

$$\max\left|\boldsymbol{X}_a^T \boldsymbol{X}_a\right| = \max\left|\boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{A}^T \boldsymbol{A}\right| \tag{2.3-5}$$

for $\boldsymbol{x}_p$.

2.3.5 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 [1]. PRESS can be computed from a single regression analysis of all P points.

$$\text{PRESS} = \sum_{i=1}^{P}\left(\frac{y_i - \hat{y}_i}{1-h_{ii}}\right)^2 \tag{2.3-6}$$

where h_{ii} are the diagonal terms of

$$H = \boldsymbol{X}(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{T} \tag{2.3-7}$$

$\boldsymbol{H}$ is the "hat" matrix, the matrix that maps the observed responses to the fitted responses, i.e.

$$\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y} \tag{2.3-8}$$

The PRESS residual can also be written in its square root form

$$\text{SPRESS} = \sqrt{\sum_{i=1}^{P}\left(\frac{y_i - \hat{y}_i}{1-h_{ii}}\right)^2}\ . \tag{2.3-9}$$

For a saturated design, $\boldsymbol{H}$ equals the unit matrix $\boldsymbol{I}$ so that the PRESS indicator becomes undefined.

2.3.6 The coefficient of multiple determination R^2

The coefficient of determination R^2 is defined as:

$$R^2 = \frac{\sum_{i=1}^{P}(\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^{P}(y_i - \bar{y}_i)^2} \tag{2.3-10}$$

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.

2.3.7 R^2 for Prediction

For the purpose of *prediction* accuracy the $R^2_{prediction}$ indicator has been devised [1].

$$R^2_{prediction} = 1 - \frac{\text{PRESS}}{S_{yy}} \tag{2.3-11}$$

where

$$S_{yy} = \boldsymbol{y}^T \boldsymbol{y} - \frac{\left(\sum_{i=1}^{P} y_i\right)^2}{P} \tag{2.3-12}$$

$R^2_{prediction}$ represents the ability of the model to detect the variability in predicting new responses [1].

2.3.8 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 improving 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).

2.4 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.

2.4.1 The confidence interval of the regression coefficients

The $100(1-\alpha)\%$ confidence interval for the regression coefficients $b_j, j = 0,1,\ldots,L$ is determined by the inequality

$$b_j - \frac{\Delta b_j}{2} \le \beta_j \le b_j + \frac{\Delta b_j}{2} \tag{2.4-1}$$

where

$$\Delta b_j(\alpha) = 2t_{\alpha/2,P-L}\sqrt{\hat{\sigma}^2 C_{jj}} \tag{2.4-2}$$

and $\hat{\sigma}^2$ is an unbiased estimator of the variance σ^2 given by

$$\hat{\sigma}^2 = \frac{\varepsilon^2}{P-L} = \frac{\sum_{i=1}^{P}(y_i - \hat{y}_i)^2}{P-L} \tag{2.4-3}$$

C_{jj} is the diagonal element of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ 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.

2.4.2 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

$$F = \frac{[\varepsilon^2_{reduced} - \varepsilon^2_{complete}]/r}{\varepsilon^2_{complete}/(P-L)} \tag{2.4-4}$$

where $r = 1$ and the reduced model is the one in which the regressor variable in question has been removed. Each of the ε^2 terms represents the sum of squared residuals for the reduced and complete models respectively.

It turns out that the computation can be done without analyzing a reduced model by computing

$$F = \frac{b_j^2/C_{jj}}{\varepsilon^2_{complete}/(P-L)}. \tag{2.4-5}$$

F can be compared with the F-statistic $F_{\alpha,1,P-L}$ so that if $F > F_{\alpha,1,P-L}$, β_j is non-zero with $(100-\alpha)\%$ confidence. The confidence level α that β_j is not zero can also be determined by computing the α for $F = F_{\alpha,1,P-L}$. The importance of β_j is therefore estimated by both the magnitude of b_j as well as the level of confidence in a non-zero β_j.
The significance of regressor variables may be represented by a bar chart of the magnitudes of the coefficients b_j with an error bar of length $2\Delta b_j(\alpha)$ for each coefficient representing the confidence interval for a given level of confidence α. The relative bar lengths allow the analyst to estimate the importance of the variables and terms to be included in the model while the error bars represent the contribution to noise or poorness of fit by the variable.

All terms have been normalized to the size of the design space so that the choice of units becomes irrelevant and a reasonable comparison can be made for variables of different kinds, e.g. sizing and shape variables or different material constants.

2.5 REFERENCES

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

[2] Box, G.E.P., Draper, N.R. A basis for the selection of a response surface design. *Journal of the American Statistical Association*, 54, pp. 622-654, 1959.

[3] Toropov, V.V. Simulation approach to structural optimization. *Structural Optimization*, 1, pp. 37-46, 1989.

[4] Schoofs, A.J.G. *Experimental Design and Structural Optimization*. PhD thesis, Technische Universiteit Eindhoven, August 1987.

[5] Tu, J. and Choi, K.K. Design potential concept for reliability-based design optimization. *Technical Report R99-07. Center for Computer Aided Design and Department of Mechanical Engineering*. College of engineering. University of Iowa. December 1999.

[6] Jin, R., Chen, W. and Simpson, T.W. Comparative studies of metamodeling techniques under multiple modeling criteria. *AIAA Paper,* AIAA-2000-4801.
[7] Roux, W.J. *Structural Optimization using Response Surface Approximations*. PhD thesis, University of Pretoria, April 1997.
[8] Wilson, B., Cappelleri, D.J., Frecker, M.I. and Simpson, T.W. Efficient Pareto frontier exploration using surrogate approximations. *Optimization and Engineering*, 2 (1), pp.31-50, 2001.
[9] Ye, K., Li, W., Sudjianto, A., Algorithmic construction of optimal symmetric Latin hypercube designs. *Journal of Statistical Planning and Inferences*, 90, pp. 145-159, 2000.
[10] McKay, M.D., Conover, W.J., Beckman, R.J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, pp. 239-245, 1979.
[11] Park, J.-S. Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning Inference*, 39, pp. 95-111, 1994.
[12] Morris, M., Mitchell, T. Exploratory design for computer experiments. *Journal of Statistical Planning Inference*, 43, pp. 381-402, 1995.
[13] Matsumoto, M. and Nishimura, T., Mersenne Twister: A 623-Dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1), pp. 3-30, 1998.

3. 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 three types of metamodeling techniques, namely polynomial response surfaces (already discussed, see Section 2.1), Neural Networks (NN) (Section 3.1.2) and Radial Basis Function Networks (RBF) (Section 3.1.3). All three 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 2.4). 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.

3.1 Neural Networks

Neural methods are natural extensions and generalizations of regression methods. Neural networks have been known since the 1940's, but it took the dramatic improvements in computers to make them practical, [3]. Neural networks - just like regression techniques - model relationships between a set of input variables and an outcome. Neural networks can be thought of as computing devices consisting of numerical units ('neurons'), whose inputs and outputs are linked according to specific topologies (see the example in Figure 3-1). 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 3-1. 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 3-1).

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 3-2), while network input and output layers are usually linear (transparent). In theory, such networks can model functions of almost arbitrary complexity, see [4] and [5]. All parameters in a feedforward network are usually determined at the same time as part of a single (non-linear) optimization strategy based on the standard gradient algorithms (the steepest descent, RPROP, Levenberg-Marquardt, etc.). The gradient information is typically obtained using a technique called backpropagation, which is known to be computationally effective [6]. For feedforward networks, regularization may be done by controlling the number of network weights (‘model selection’), by imposing penalties on the weights (‘ridge regression’) [7], or by various combinations of these strategies [8].

A radial basis function network has a single hidden layer of radial units, each actually modeling a response function, peaked at the center, and monotonically varying outwards (Figure 3-3). 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 3-4 and Figure 3-5). The formal proofs of this property can be found, for example, in [9] and [10]. An RBF network can be trained extremely quickly, orders of magnitude faster than a feedforward network. The training process typically takes place in two distinct stages. First, the centers and deviations of the radial units (i.e. the hidden layer's weights) must be set; then the linear output layer is optimized. It is important that deviations are chosen so that RBFs overlap with some nearby units. Discovering a sub-optimal ‘spread’ parameter typically implies the preliminary experimental stage. If the RBFs are too spiky, the network will not interpolate between known points (see Figure 3-6). If the RBFs are

very broad, the network loses fine detail (Figure 3-7). 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 3-1) 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.

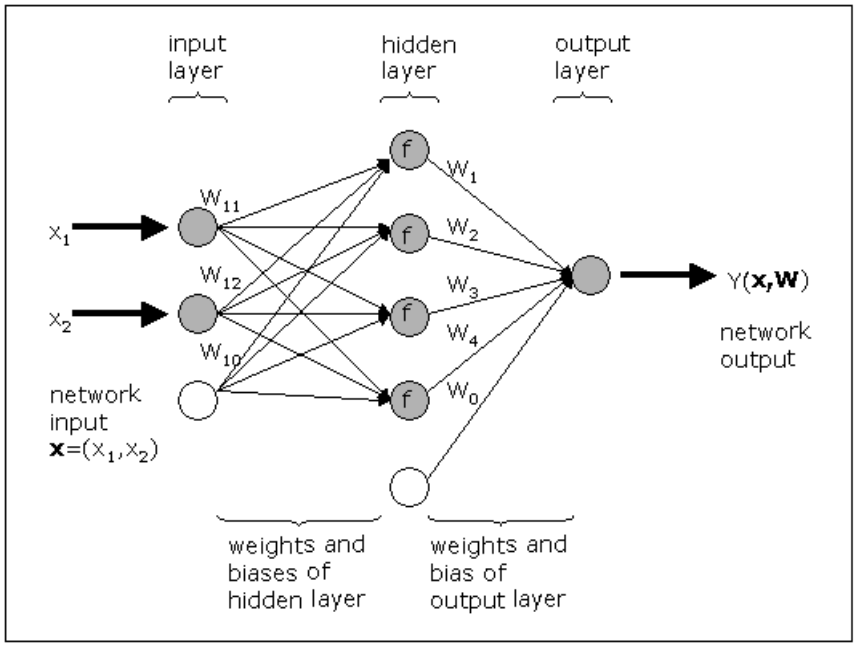


Figure 3-1: Schematic of a neural network with 2 inputs and a hidden layer of 4 neurons with activation function *f*.

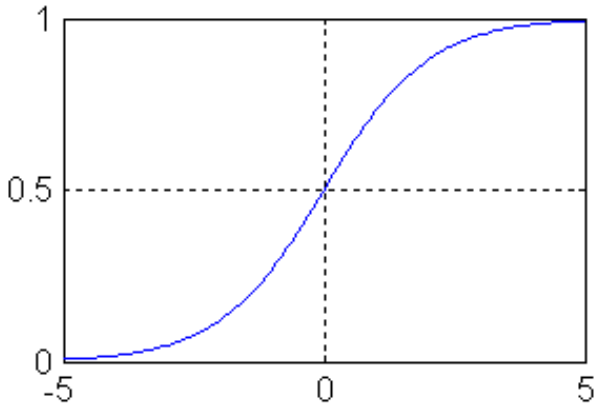


Figure 3-2: Sigmoid transfer function $y = 1/\left(1+e^{-x}\right)$ typically used with feedforward networks

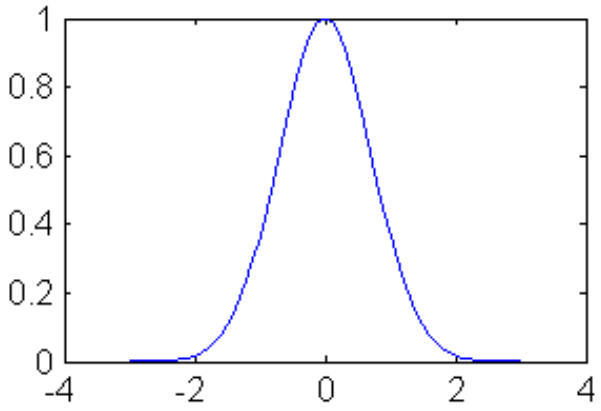


Figure 3-3: Radial basis transfer function $y = \exp[-x^2]$

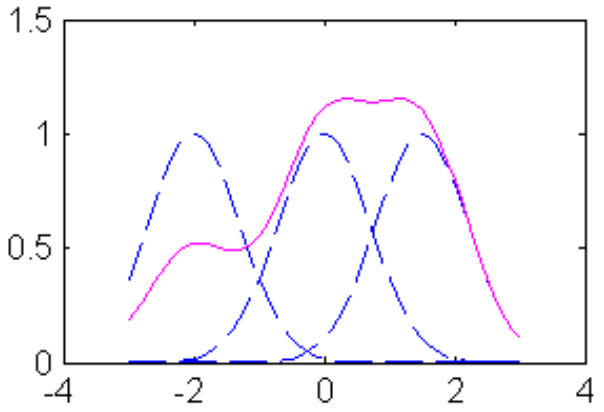


Figure 3-4: Weighted sum of radial basis transfer functions. Three radial basis functions (dashed lines) are scaled and summed to produce a function (solid line).

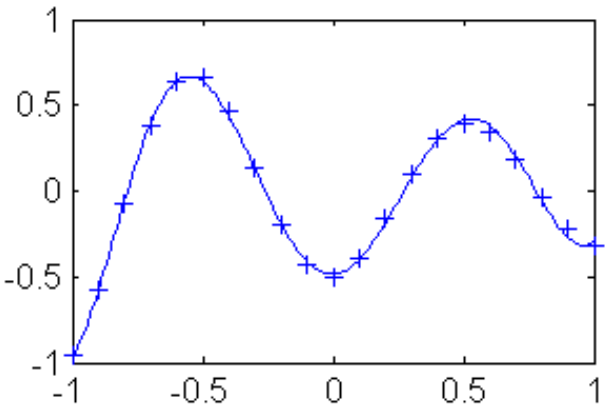


Figure 3-5: A radial basis network approximation (solid line) of the function, which fits the 21 data points (plus symbols).

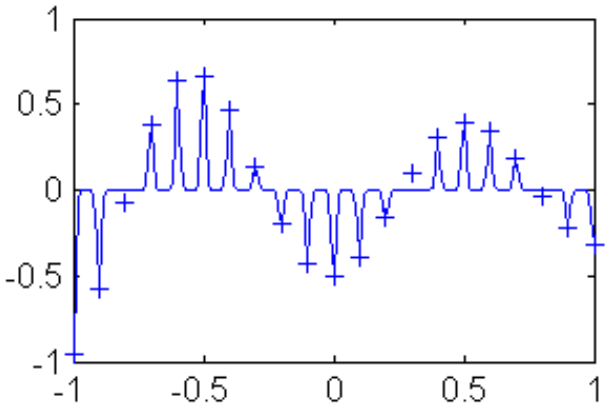


Figure 3-6: The same 21 data points as in Figure 3-5. Test points reveal that the function has been overfit. RBF neuron's spread is too small. RBF network could have done better with a higher spread constant.

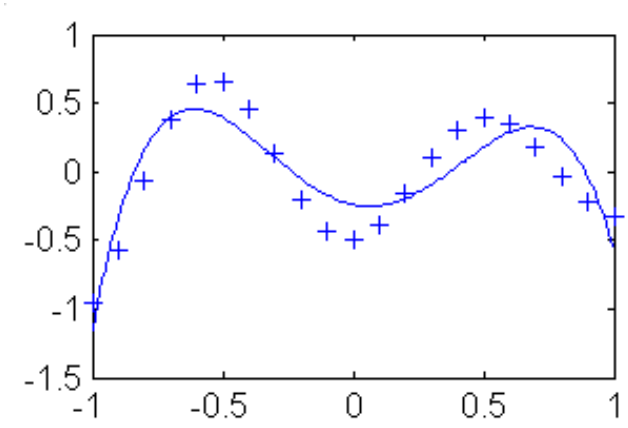


Figure 3-7: The same 21 data points as in Figure 3-5. Approximation with overlapping RBF neurons. The spread of RBF units is too high.

3.1.1 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:

$$MSE = \sum_{i}^{P} (\hat{y}_i - y_i)^2 / P, \tag{1}$$

$$RMS = \sqrt{MSE}\,; \qquad nMSE = \frac{MSE}{\hat{\sigma}^2}\,; \qquad nRMS = \frac{RMS}{\hat{\sigma}^2}$$

$$R^2 = \frac{\sum_{i=1}^{P} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{P} (y_i - \bar{y})^2} \quad \text{and} \quad R = \frac{\sum_{i=1}^{P} \left|\hat{y}_i - \bar{\hat{y}}\right| \left|y_i - \bar{y}\right|}{\sum_{i=1}^{P} (y_i - \bar{\hat{y}})^2 \sum_{i=1}^{P} (y_i - \bar{y})^2}$$

where P denotes the number of data points, y_i is the observed response value ('target value'), $\hat{y}_i$ is the model's prediction of response, $\bar{\hat{y}}$ is the mean (average) value of $\hat{y}$, $\bar{y}$ is the mean (average) value of y, and $\hat{\sigma}^2$ is given by

$$\hat{\sigma}^2 = \frac{\varepsilon^2}{P - L} = \frac{\sum_{i=1}^{P} (y_i - \hat{y}_i)^2}{P - L}$$

Mean squared error (MSE for short) and root mean squared error (RMS) summarize the overall model error. Unique or rare large error values can affect these indicators. Sometimes, MSE and RMS measures are normalized with sample variance of the target value (see formulae for nMSE and nRMS) to allow for comparisons between different datasets and underlying functions. R^2 and R are relative measures. The coefficient of multiple determination R^2 ('R-square') is the explained variance relative to the total variance in the target value. This indicator is widely used in linear regression analysis. R^2 represents the amount of response variability explained by the model. R is the correlation coefficient between the network response and the target. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is a perfect correlation between targets and outputs. Outliers can greatly affect the magnitudes of correlation coefficients. Of course, the larger the sample size, the smaller is the impact of one or two outliers.

Training accuracy measures (MSE, RMS, R^2, R, etc.) are computed along all the data points used for training. As mentioned above, the performance of a good model on the training set does not necessarily mean good prediction of new (unseen) data. The objective measures of the prediction accuracy of the model are test errors computed along independent testing points (i.e. not training points). This is certainly true provided that we have an infinite number of testing points. In practice, however, test indicators are usable, only if treated with appropriate caution. Actual problems are often characterized by the limited availability of data, and when the training datasets are quite small and the test sets are even smaller, only quite large differences in performance can be reliably discerned by comparing training and test indicators.

The generalized cross-validation (GCV) [11] and Akaike's final prediction error (FPE) [12] provide computationally feasible means of estimating the appropriateness of the model. The *k*-fold cross-validation (denoted here as CV-*k*), generalized cross-validation (GCV) [11] and Akaike's final prediction error (FPE) [12] provide computationally feasible means of estimating the appropriateness of the model.

GCV and FPE estimates combine the training MSE with a measure of the model complexity:

$$MSE_{GCV} = MSE/(1-\frac{\nu}{P})^2, \tag{2}$$

$$RMS_{GCV} = \sqrt{MSE_{GCV}}; \qquad nMSE_{GCV} = \frac{MSE_{GCV}}{\hat{\sigma}^2}; \qquad nRMS_{GCV} = \frac{RMS_{GCV}}{\hat{\sigma}}$$

$$MSE_{FPE} = MSE \cdot (1+\frac{\nu}{P})/(1-\frac{\nu}{P}) \tag{3}$$

$$RMS_{FPE} = \sqrt{MSE_{FPE}}; \qquad nMSE_{FPE} = \frac{MSE_{FPE}}{\hat{\sigma}^2}; \qquad nRMS_{FPE} = \frac{RMS_{FPE}}{\hat{\sigma}}$$

where ν is the (effective) number of model parameters.

In theory, GCV estimates should be related to ν. As a very rough approximation to ν, we can assume that all of the network free parameters are well determined so that $\nu = M$, where M is the total number of network weights and biases. This is what we would expect to be the case for large P so that $P >> M$. Note that GCV is undefined when ν is equal to the number of training points (P). In theory, GCV and FPE estimates should

be related to the effective number of model's parameters ν. Techniques to assess v for neural networks will be discussed later. As a very rough approximation, we can assume that all of the network free parameters are well determined so that ν = M, where M is the total number of network weights and biases. This is what we would expect to be the case for large P so that $P >> M$. Note that both GCV and FPE are undefined when the effective number of model's parameters (ν) is equal to the number of training points (P). GCV and FPE measures are asymptotically equivalent for large P.

In k-fold cross-validation the training dataset is divided into k randomly selected disjoint subsets of roughly equal size $P^{(j)}$. The model is trained and tested k times. Each time $j = 1,...,k$ it is trained on all data except for points from subset j and then tested on j-th subset. Formally, let $y^{(j)} = (y_i^{(j)}), i = 1,..., P^{(j)}$ be the prediction of such a model for the points from subset j. Then the CV-k estimates of accuracy

$$MSE_{CV-k} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{P^{(j)}}(y_i^{(j)} - \hat{y}_i)^2}{P} \tag{4}$$

$$RMS_{CV-k} = \sqrt{MSE_{CV-k}}\,; \qquad nMSE_{CV-k} = \frac{MSE_{CV-k}}{\hat{\sigma}^2}\,; \qquad nRMS_{CV-k} = \frac{RMS_{CV-k}}{\hat{\sigma}}$$

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 judgement 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.

3.1.2 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 3-3), 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 $\boldsymbol{x} = (x_1,\ldots,x_K)$ and network weights $\boldsymbol{W} = (W_0, W_1,\ldots, W_H, W_{10}, W_{11},\ldots, W_{HK})$, the output of the network is:

$$\hat{y}(\boldsymbol{x},\boldsymbol{W}) = W_0 + \sum_{h=1}^{H} W_h f(W_{h0} + \sum_{k=1}^{K} W_{hk} x_k), \tag{5}$$

where

$$f(x) = \frac{1}{1+e^{-x}}$$

The computational graph of Eq. (5) is shown schematically in Figure 3-1. 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. (5) with respect to any of its inputs is given by:

$$\frac{\partial \hat{y}}{\partial x_k} = \sum_{h=1}^{H} W_h W_{hk} f'(W_0 + \sum_{h=1}^{H} W_h), \quad k = 1,\ldots,K. \tag{6}$$

Neural networks have been mathematically shown to be universal approximators of continuous functions and their derivatives (on compact sets) [4]. In other words, when a network (5) converges towards the underlying function, all the derivatives of the network converge towards the derivatives of this function.

Standard non-linear optimization techniques including a variety of gradient algorithms (the steepest descent, RPROP, Levenberg-Marquardt, etc.) are applied to adjust FF network's weights and biases. For neural networks, the gradients are easily obtained using a chain rule technique called 'backpropagation' [6]. The second-order Levenberg-Marquardt algorithm appears to be the fastest method for training moderate-sized FF neural networks (up to several hundred adjustable weights) [3]. However, when training larger networks, the first-order RPROP algorithm becomes preferable for computational reasons [13].

Regularization: For FF networks, regularization may be done by controlling the number of network weights ('model selection'), by imposing penalties on the weights ('ridge regression'), or by various combinations of these strategies ([7], [8]). Model selection requires choosing the number of hidden units 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 units and finally select the network with the smallest GCV error. In any event, in order for the GCV measure to be applicable, the number of training points P should not be too small compared to the required network size M.

Over-fitting: To prevent over-fitting, it is always desirable to find neural solutions with the smallest number of parameters. In practice, however, networks with a very parsimonious number of weights are often hard to train. The addition of extra parameters (i.e. degrees of freedom) can aid convergence and decrease the chance of becoming stuck in local minima or on plateaus [14]. Weight decay regularization involves modifying the performance function F , which is normally chosen to be the mean sum of squares of the network errors on the training set (1). When minimizing MSE (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 (1)):

$$F = \beta E_D + \alpha E_W \tag{7}$$

where

$$E_D = \frac{\sum_{i=1}^{P}(\hat{y}_i - y_i)^2}{2}, \qquad E_W = \frac{\sum_{m=1}^{M} W_m^2}{2},$$

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 (7) will cause the network to have smaller weights, and this will force the network response to be smoother and less likely to overfit. This eliminates the guesswork required in determining the optimum network size. Unfortunately, finding the optimal value for α and β is not a trivial task. If we make α/β too small, we may get over-fitting. If α/β is too large, the network will not adequately fit the training data. A rule of thumb is that a little regularization usually helps [15]. It is important that weight decay regularization does not require that a validation subset be separated out of the training data. It uses all of the data. This advantage is especially noticeable in small sample size situations. Another nice property of weight decay regularization is that it can lend numerical robustness to the Levenberg-Marquardt algorithm. The L-M approximation to the Hessian of Eq. (7) is moved further away from singularity due to a positive addend to its diagonal:

$$\boldsymbol{A} = \boldsymbol{H} + \alpha \boldsymbol{I} \tag{8}$$

where

$$\boldsymbol{H} = \beta \nabla \nabla E_D \approx \sum_{i=1}^{P} \boldsymbol{g}(x^{(i)}) \cdot \boldsymbol{g}(x^{(i)})^T$$

$$\boldsymbol{g}(\boldsymbol{x}) = \left(\frac{\partial \hat{y}}{\partial W_1}, \ldots, \frac{\partial \hat{y}}{\partial W_M}\right)^T$$

In [3], [16], [17]and [18] the Bayesian (’evidence framework’ or ’type II maximum likelihood’) approach to regularization is discussed. The Bayesian re-estimation algorithm is formulated as follows. At first, we choose the initial values for α and β. Then, a neural network is trained using a standard non-linear optimization algorithm to minimize the error function (Eq. (7)). After training, i.e. in the minimum of Eq. (7), the values for α and β are re-estimated, and training restarts with the new performance function. Regularization hyperparameters are computed in a sequence of 3 steps:

$$\nu = \frac{\sum_{m=1}^{M} \lambda_m}{\lambda_m + \alpha} \tag{9}$$

where λ_m, $m = 1,\ldots,M$ are (positive) eigenvalues of matrix $\boldsymbol{H}$ in Eq. (8), ν is the estimate of the effective number of parameters of a neural network,

$$\alpha = \frac{\nu}{2E_W}$$

$$\beta = \frac{P - \nu}{2E_D}$$

It should be noted that the algorithm (Eq. 9) relies on numerous simplifications and assumptions, which hold only approximately in typical real-world problems [19]. In the Bayesian formalism a trained network is described in terms of the posterior probability distribution of weight values. The method typically assumes a

simple Gaussian prior distribution of weights governed by an inverse variance hyperparameter $\alpha = 1/\sigma_{weights}^2$. 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:

$$y = y(x) + N(0, \sigma_{noise}^2).$$

With these assumptions, the negative log likelihood of network weights W given P training points $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(P)$ is proportional to MSE (Eq. (1)), i.e., the maximum likelihood estimate for W is that which minimizes (Eq. (1)) or, equivalently, E_D. In order for Bayes estimates of α and β to do a good job of minimizing the generalization in practice, it is usually necessary that the priors on which they are based are realistic. The Bayesian formalism also allows us to calculate error bars on the network outputs, instead of just providing a single 'best guess' output $\hat{y}$. Given an unbiased model, minimization of the performance function (Eq. (1)) amounts to minimizing the variance of the model. The estimate for output variance $\sigma_{\hat{y}|x}^2$ of the network at a particular point x is given by:

$$\sigma_{\hat{y}|x}^2 \approx \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{A}^{-1} \boldsymbol{g}(\boldsymbol{x}) \tag{10}$$

Equation (10) is based on a second-order Taylor series expansion of Eq. (7) around its minimum and assumes that $\partial\hat{y}/\partial\boldsymbol{W}$ is locally linear.

Variability of Feedforward neural networks

Neural networks have a natural variability because of the following reasons [20]:

1. Local behavior of the neural network training algorithms
2. Uncertainty (noise) in the training data

The neural network training error function usually has multiple local and global minima. With different initial weights, the training algorithm typically ends up in different (but usually almost equally good/bad) local minima. The larger the amount of noise in the data, the larger 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 13.1.3). The starting weights for network training is randomly generated using a user-specified seed to ensure repeatability (see Section 2.2.7).

3.1.3 Radial basis function networks

A radial basis function neural network has a distinct 3-layer topology. The input layer is linear (transparent). The hidden layer consists of non-linear radial units, each responding to only a local region of input space. The output layer performs a biased weighted sum of these units and creates an approximation of the input-output mapping over the entire space.

While several forms of radial basis function are considered in the literature, the most common functions are the Hardy's multi-quadrics and the Gaussian basis function. These are given as:

Hardy's multi-quadric:

$$g_h(x_1,...,x_k) = \sqrt{1+\left(r^2/\sigma_h{}^2\right)} \tag{11}$$

Gaussian:

$$g_h(x_1,...,x_k) = \exp\left[-\left(r^2/2\sigma_h{}^2\right)\right] \tag{12}$$

The activation of hth radial basis function is determined by the Euclidean distance $r = \left[\sum_{k=1}^{K}(x_k - W_{hk})^2\right]^{1/2}$ between the input vector $\boldsymbol{x} = (x_1,...,x_K)$ and RBF center $\boldsymbol{W}_h = (W_{h1},...,W_{hk})$ in K-dimensional space. The Gaussian basis function is a localized function (peaked at the center and descending outwards) with the property that $g_h \to 0$ as $r \to \infty$. Parameter σ_h controls the smoothness properties of the RBF unit.

For a given input vector $x = (x_1,...,x_K)$ the output of RBF network with K inputs and a hidden layer with H basis function units is given by (see also Eq.(11)):

$$Y(\mathbf{x},W) = W_0 + \sum_{h=1}^{H} W_h \cdot f(\rho_h) \tag{13}$$

where

$$\rho_h = W_{h0}\sum_{k=1}^{K}(x_k - W_{hk})^2; \quad W_{h0} = 1/2\sigma_h{}^2; \quad f(\rho) = e^{-\rho}$$

Notice that hidden layer parameters $W_{h1},...,W_{hk}$ represent the center of hth radial unit, while W_{h0} corresponds to its deviation. Parameters W_0 and $W_1,...,W_H$ are the output layer's bias and weights, respectively.

A linear super-position of localized functions as in (13) is capable of universal approximation. The formal proofs of this property can be found, for example, in [9] and[10]. It is straightforward to show that the derivative of the network (13) with respect to any of its inputs is given by:

$$\frac{\partial Y}{\partial X_k} = \sum_{h=1}^{H} W_h W_{h0} \cdot 2(x_k - W_{hk}) \cdot f'(\rho_h), \qquad k = 1,...,K \tag{14}$$

where f' denotes the first derivative of the transfer function f: $f'(\rho) = -e^{-\rho}$.

Theory tells us that when a network (13) 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 units than a comparable feedforward network, *RBF networks can be trained extremely quickly, orders of magnitude faster than FF networks*.

For RBF networks, the training process generally takes place in two distinct stages. First, the centers and deviations of the radial units (i.e. hidden layer parameters $W_{11},...,W_{HK}$ and $W_{10},...,W_{H0}$) must be set. All the basis functions are then kept fixed, while the linear output layer (i.e. $W_0,...,W_H$) is optimized in the second phase of training. In contrast, all of the parameters in a FF network are usually determined at the same time as part of a single training (optimization) strategy. Techniques for selecting $W_{11},...,W_{HK}$ and $W_{10},...,W_{H0}$ are discussed at length in following paragraphs. Here we shall assume that the RBF parameters have already been chosen, and we focus on the problem of optimizing the output layer weights.

Mathematically, the goal of output layer optimization is to minimize a performance function, which is normally chosen to be the mean sum of squares of the network errors on the training set (1). If the hidden layer's parameters $W_{10},W_{11},...,W_{HK}$ in (3.4-2) are kept fixed, then the performance function (1) is a quadratic function of the output layer' parameters $W_0,...,W_H$ and its minimum can be found in terms of the solution of a set of linear equations (e.g. using singular value decomposition). The possibility of avoiding the need for time-consuming and costly non-linear optimization during training is one of the major advantages of RBF networks over FF networks. However, when the number of optimized parameters ($H+1$, in our case) is small enough, non-linear optimization (Levenberg-Marquardt, etc.) may also be cost-effective.

It is clear that the ultimate goal of RBF neural network training is to find a smooth mapping which captures the underlying systematic aspects of the data without fitting the noise. However, for noisy data, the exact RBF network, which passes exactly through every training data point, is typically a highly oscillatory function. There are a number of ways to address this problem. By analogy with FF network training, one can add to (1) a regularization term that consists of the mean of the sum of squares of the optimized weights. In conventional curve fitting this form of regularization is called ridge regression. The sub-optimal value for hyperparameters α and β in (7) can be found by applying Bayesian re-estimation formulae (8)-(9). It is also feasible to iterate over several trial values of α and β.

For RBF networks, however, the most effective regularization methods are probably those pertaining to selecting radial centers and deviations in the first phase of RBF training. The commonly held view is that RBF centers and deviations should be chosen so as to form a representation of the probability density of the input data. The classical approach is to set RBF centers equal to all the input vectors from the training dataset. The width parameters σ_h are typically chosen – rather arbitrarily – to be some multiple S_σ of the average spacing between the RBF centers (e.g. to be roughly twice the average distance). This ensures that the RBF's overlap to some degree and hence give a relatively smooth representation of data.

To simplify matters, the same value of the width parameter σ_h for all RBF units is usually considered. Sometimes, instead of using just one value for all RBF's, each RBF unit's deviation σ_h is individually set to the distance to its $N_\sigma << N$ nearest neighbors. Hence, deviations σ_h become smaller in densely populated areas of space, preserving fine detail, and are higher in sparse areas of space, interpolating between points where necessary. Again the choice of N_σ is somewhat arbitrary. RBF networks with individual radial

deviations σ_h can be particularly useful in situations where data tend to cluster in only a small subregion of the design space (for example, around the optimum of the underlying system which RSM is searching for) and are sparse elsewhere.

One must take into consideration that after the first phase of RBF training is over, there's no way to compensate for large inaccuracies in radial deviations σ_h by, say, adding a regularization term to the performance function. If the basis functions are too spiky, the network will not interpolate between known points, and thus, will lose the ability to generalize. If the Gaussians are very broad, the network is likely to lose fine detail. The popular approach to find a sub-optimal spread parameter is to loop over several trial values of S_σ and N_σ and finally select the RBF network with the smallest GCV (FPE, CV-k) error. This is somewhat analogous to searching for an optimal number of hidden units of a feedforward neural network.

In order to eliminate all the guesswork required in determining RBF deviations, it might seem natural to treat $W_{10},...,W_{H0}$ ($\sigma_1,...,\sigma_H$, to be precise) in (12) 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 Gaussians will remain localized.

It should be noted that RBF networks may have certain difficulties if the number of RBF units (H) is large. This is often the case in multidimensional problems. The difficulty arises because for a large number of RBF's, a large number of training samples are required in order to ensure that the neural network parameters are properly determined. A large number of RBF units also increase the computation time spent on optimization of the network output layer and, consequently, the RBF architecture loses its main (if not the only one) advantage over FF networks – fast training.

Radial basis function networks actually suffer more from the curse of dimensionality than feedforward neural networks. To explain this statement, consider the effect of adding an extra, perfectly spurious input variable to a network. A feedforward network can learn to set the outgoing weights of the spurious input to zero, thus ignoring it. An RBF network has no such luxury: data in the relevant lower-dimensional space get ‘smeared’ out through the irrelevant dimension, requiring larger numbers of units to encompass the irrelevant variability.

In principle, the number of RBF's (H) need not equal the number of training samples (P), and RBF units are not constrained to be centered on the training data points. In fact, when data contain redundant information, we do not need all data points in learning. One simple procedure for selecting RBF centers is to set them equal to a random subset of the input vectors from the training set. Since they are randomly selected, they will 'represent' the distribution of the (redundant) training data in a statistical sense. Of course, H and P should not be too small in this case.

It is clear, however, that the optimal choice of RBF centers based on the input data alone need not be optimal for representing the input-output mapping as reflected in the observed data. In order to overcome these limitations, the selection procedure should take into account the output values, or at least, approximate estimates (assumptions) of the global behavior of the underlying system. The common neural term for such techniques involving output values is ‘active learning’. In the context of active learning, RBF networks can be thought of as DOE metamodels analogous to polynomials, [16] and [19]. Given a candidate list of points,

an active learner is searching for the 'best' points in order to position RBF centers. Popular in neural applications is to treat RBF active learning as 'pruning' technique intended for identifying critical data and discarding redundant points, or more accurately, not selecting some training points as RBF centers. RBF active learning methods are being successfully applied to approximate huge datasets that come from natural stochastic processes. It is questionable, however, whether active learning can be useful for non-redundant datasets, specifically for RSM design sets generated by performing DOE analysis based on low-order polynomial metamodels.

To briefly summarize, parameters governing radial units (radial centers and deviations) play a key role in generalization performance of a RBF model. The appropriate selection of RBF centers implies that we choose a minimal number of training data points that carry enough information to build an adequate input-output representation of the underlying function. Unfortunately, this is easier said than done. Indeed, there is a general agreement that selecting RBF centers and deviations is more Art than Science.

3.2 Kriging*

Kriging is named after D.G. Krige [22], who applied empirical methods for determining true ore grade distributions from distributions based on sampled ore grades. In recent years, the Kriging method has found wider application as a spatial prediction method in engineering design. Detailed mathematical formulations of Kriging are given by Simpson [23] and Bakker [24].

The basic postulate of this formulation [23] is :

$$y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x})$$

where y is the unknown function of interest, $f(\mathbf{x})$ is a known polynomial and $Z(\mathbf{x})$ the stochastic component with mean zero and covariance:

$$Cov[Z(\mathbf{x}^i),Z(\mathbf{x}^j)] = \sigma^2\mathbf{R}([R(\mathbf{x}^i,\mathbf{x}^j)]).$$

With L the number of sampling points, $\mathbf{R}$ is the L x 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:

$$\text{Exponential:} \quad R=\prod_{k=1}^{n} e^{-\Theta_k|d_k|} \quad \text{and}$$

$$\text{Gaussian:} \quad R=\prod_{k=1}^{n} e^{-\Theta_k d_k^2}$$

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 θ-values to be determined. The default function in LS-OPT is Gaussian.

Once the correlation function has been selected, the predicted esitimate of the response $\hat{y}(\boldsymbol{x})$ is given by:

$$\hat{y} = \hat{\beta} + \mathbf{r}^T(\boldsymbol{x})\mathbf{R}^{-1}(\mathbf{y}-\mathbf{f}\hat{\beta})$$

where $\mathbf{r}^T(\boldsymbol{x})$ is the correlation vector (length L) between a prediction point $\boldsymbol{x}$ and the L sampling points, $\mathbf{y}$ represents the responses at the L points and $\mathbf{f}$ is an L-vector of ones (in the case that $f(\boldsymbol{x})$ is taken as a constant). The vector $\mathbf{r}$ and scalar $\hat{\beta}$ are given by:

$$\mathbf{r}^T(\boldsymbol{x}) = [R(\boldsymbol{x},\boldsymbol{x}^1),R(\boldsymbol{x},\boldsymbol{x}^2),\ldots,R(\boldsymbol{x},\boldsymbol{x}^L)]^T$$

$$\hat{\beta} = (\mathbf{f}^T\mathbf{R}^{-1}\mathbf{f})^{-1}\mathbf{f}^T\mathbf{R}^{-1}\mathbf{y}.$$

The estimate of variance from the underlying global model is:

$$\hat{\sigma}^2 = \frac{(\mathbf{y}-\mathbf{f}\hat{\beta})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{f}\hat{\beta})}{L} \quad .$$

The maximum likelihood estimates for Θ_k, $k = 1,\ldots, n$ can be found by solving the following constrained maximization problem:

$$\text{Max } \Phi(\boldsymbol{\Theta}) = \frac{-[L\ln(\hat{\sigma}^2)+\ln|\mathbf{R}|]}{2}, \text{ subject to } \boldsymbol{\Theta}>0.$$

where both $\hat{\sigma}$ and $|\mathbf{R}|$ are functions of $\boldsymbol{\Theta}$. This is the same as minimizing

$$\hat{\sigma}^2 |\mathbf{R}|^{1/n}, \text{ s.t. } \boldsymbol{\Theta}>0$$

This optimization problem is solved using the LFOPC algorithm (Section 4.4). Because of the possible ill-conditioning of $\mathbf{R}$, a small constant number is adaptively added to its diagonal during optimization. The net effect is that the approximating functions no longer interpolate the observed response values exactly. However, these observations are still closely approximated.

3.3 Concluding remarks: which metamodel?

There is little doubt that the polynomial-based response surfaces are the most robust, especially for sequential optimization methods. A negative aspect of using polynomials is the fact that the user is obliged

to choose the order of polynomial. Also, a greater possibility exists for bias error of a nonlinear response. They are also, in most cases, not suitable for updating in sequential methods. Linear approximations may only be useful within a certain subregion and therefore quadratic polynomials or other higher order approximations such as RBF networks may be required for greater global accuracy. However the linear SRSM method has proved to be excellent for sequential optimization and can be used with confidence [25][26][27].

RBF Networks appear to be generally the best of the neural networks metamodels. They have the following advantages:

- Higher prediction accuracy due to built-in cross validation. Although FF networks may appear more accurate due to a smaller fitting error (RMSE), their prediction error is generally larger than that of RBF networks. An appealing plot of predicted vs. computed responses showing the training points or R^2 values approaching unity or small RMS error values should not be construed as representing a higher accuracy.
- Higher speed due to their linear nature. When sizable FF committees (e.g. with 9 members) are used they may be vastly more expensive to construct than RBF networks. This is true especially for a relatively small number of variables.
- Relative independence of the calculation time with respect to the number of functions. Although there is a slight overhead which depends on this number, the user does not have to be as careful with limiting the number of responses.

FF Neural Networks function well as global approximations and no serious deficiencies have been observed when used as prescribed in Section 4.7. FF networks have been used for sequential optimization [27] and can be updated during the process. A more recent study [28] which focuses on the accuracy comparison for FF neural networks and RBF networks for different types of optimization strategies concluded that, for crashworthiness analysis, RBF and FF metamodels are mostly similar in terms of the accuracy of a large number of checkpoint results. However, the same study showed that Neural Networks are sometimes better than RBF networks for smooth problems. As mentioned earlier, RBF networks have a distinct speed advantage. Reference [28] also assesses the use of FF committees and concludes that, although expensive, there are some cases where they may be necessary.

Although the literature seems to indicate that Kriging is one of the more accurate methods [23], there is evidence of Kriging having fitting problems with certain types of experimental designs [29]. Kriging is very sensitive to noise, since it interpolates the data [30]. The authors of this manual have also experienced fitting problems with non-smooth surfaces ($Z(\mathbf{x})$ observed to peak at data points) in some cases, apparently due to large values of $\boldsymbol{\Theta}$ that may be due to local optima of the maximum likelihood function. The model construction can be very time consuming [30] (also experienced with LS-OPT). Furthermore, the slight global altering of the Kriging surface due to local updating has also been observed [27]. Work is under way to improve the Kriging implementation in LS-OPT.

Reference [27] compares the use of three of the metamodeling techniques for crashworthiness optimization. This paper, which incorporates three case studies in crashworthiness optimization, concludes that while RSM, NN and Kriging were similar in performance, RSM and NN were shown to be the most robust for this application. RBF networks were not available at the time of that study.

3.4 REFERENCES

[1] Daberkow, D.D., Mavris, D.N. An investigation of metamodeling techniques for complex systems design. *Symposium on Multidisciplinary Analysis and Design*, Atlanta, October 2002.

[2] Redhe, M., Nilsson, L. Using space mapping and surrogate models to optimize vehicle crashworthiness design, *AIAA Paper 2002-5607. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, September 4-6, 2003.

[3] Bishop, C.M. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] Hornik, K., Stinchcombe, M., White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3, pp. 535-549, 1990.

[5] White, H., Hornik, K., Stinchcombe, M. Universal approximation of an unknown mapping and its derivatives. *Artificial Neural Networks: Approximations and Learning Theory*, H. White, ed., Oxford, UK: Blackwell, 1992.

[6] Rummelhart, D.E., Hinton, G.E., Williams, R.J. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, Vol. I Foundations, pages 318-362. MIT Press, Cambridge, MA, 1986.

[7] Hoerl, A.H., Kennard, R.W., Ridge regression: Biased estimation for nonorhtogonal problems. *Technometrics*, 12(3), pp. 55-67, 1970.

[8] Tikhonov, A.N., Arsenin, V.Y., *Solutions of Ill-Posed Problems*, Winston: Washington, 1977.

[9] Hartman, E.J., Keeler, J.D., Kowalski, J.M. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation,* 2(2), pp. 210-215, 1990.

[10] Park, J., Sandberg, I.W. Approximation and radial basis function networks. *Neural Computation,* 5(2), pp. 305-316, 1993.

[11] Wahba, G. *Spline Models for Observational Data*. Volume 59 of Regional Conference Series in Applied Mathematics. SIAM Press, Philadelphia, 1990.

[12] Akaike, H. Statistical predictor identification. *Ann.Inst.Statist.Math.*, 22, pp. 203-217, 1970.

[13] Riedmiller, M., Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In H. Ruspini, editor, *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, pp. 586 - 591, San Francisco, 1993.

[14] Lawrence, S.C., Lee Giles, Ah Chung Tsoi. What size neural network gives optimal generalization? Convergence Properties of Backpropogation. *Technical Report UMIACS-TR-96-22 and CS-TR-3617*, University of Maryland, 1996.

[15] Sjöberg, J., Ljung, L. Overtraining, regularization, and searching for minimum in neural networks. Preprints of the *4th IFAC Int. Symp. on Adaptive Systems in Control and Signal Processing*, p. 669, July 1992.

[16] MacKay, D. J. C. Bayesian interpolation. *Neural Computation*, 4(3), pp. 415-447, 1992.

[17] Foresee, F. D., Hagan, M. T. Gauss-Newton approximation to Bayesian regularization. *Proceedings of the 1997 International Joint Conference on Neural Networks*, pp. 1930-1935, 1997.

[18] Moody, J.E. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. in J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, 4, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[19] Cohn, D. Neural network exploration using optimal experiment design, *Neural Networks*, (9)6, pp. 1071-1083, 1996.

[20] Fedorova, N.N. Personal communication, 2004.

[21] Shyy, W., Papila, N. Vaidyanathan, R. and Tucker, P.K. Global design optimization for aerodynamics and rocket propulsion components, *Progress in Aerospace Sciences*, 37, pp. 59-118, 2001.

[22] Krige, D.G. *A Statistical Approach to Some Mine Valuation and Allied Problems on the Witwatersrand*. Masters thesis, University of the Witwatersrand, South Africa, 1951.

[23] Simpson, T.W. *A Concept Exploration Method for Product Family Design*. Ph.D. Thesis, Georgia Institute of Technology, 1998.

[24] Bakker, T.M.D. *Design Optimization with Kriging Models*. WBBM Report Series 47, Ph.D. thesis, Delft University Press, 2000.

[25] Stander, N., Craig, K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization, *Engineering Computations*, 19(4), pp. 431-450, 2002.

[26] Stander, N., Reichert, R., Frank, T. 2000: Optimization of nonlinear dynamic problems using successive linear approximations. *AIAA Paper 2000-4798*.

[27] Stander, N., Roux, W.J., Giger, M., Redhe, M., Fedorova, N. and Haarhoff, J. Crashworthiness optimization in LS-OPT: Case studies in metamodeling and random search techniques. *Proceedings of the 4th European LS-DYNA Conference*, Ulm, Germany, May 22-23, 2003. (Also www.lstc.com).

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

[29] Xu, Q-S., Liang, Y-Z., Fang, K-T., The effects of different experimental designs on parameter estimation in the kinetics of a reversible chemical reaction. *Chemometrics and Intelligent Laboratory Systems*, 52, pp. 155-166, 2000.

[30] Jin, R., Chen, W. and Simpson, T.W. Comparative studies of metamodeling techniques under multiple modeling criteria, *AIAA Paper, AIAA-2000-4801*.

[31] Simpson, T.W., Lin, D.K.J. and Chen, W. Sampling Strategies for Computer Experiments: Design and Analysis. *International Journal for Reliability and Applications*, Aug. 2001 (Revised Manuscript).

[32] Jin, R., Chen, W. and Sudjianto, A. On sequential sampling for global metamodeling in engineering design, DETC-DAC34092, 2002 ASME Design Automation Conference, Montreal, Canada, September 2002.

4. Optimization Algorithms

4.1 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

$$\min \ f(\boldsymbol{x}) \tag{4.1-1}$$

subject to

$$g_j(\boldsymbol{x}) \le 0 \quad ; \quad j = 1,2,\ldots,m$$

and

$$h_k(\boldsymbol{x}) = 0 \quad ; \quad k = 1,2,\ldots,l$$

where f, g and h are functions of independent variables x_1, x_2, x_3, …, x_n. The function f, referred to as the cost or objective function, identifies the quantity to be minimized or maximized. The functions g and h are constraint functions which represent the design restrictions. The variables collectively described by the vector $\boldsymbol{x}$ 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.

$$h_k(\boldsymbol{x}) = 0 \ \sim \ 0 \le h_k(\boldsymbol{x}) \le 0 \tag{4.1-2}$$

Equations (2.1) then become

$$\min \ f(\boldsymbol{x}) \tag{4.1-3}$$

subject to

$$g_j(\boldsymbol{x}) \le 0 \quad ; \quad j = 1,2,\ldots,m$$

The necessary conditions for the solution x^* to Eq. (2.3) are the Karush-Kuhn-Tucker optimality conditions:

$$\begin{aligned} \nabla f\left(\boldsymbol{x}^*\right) + \boldsymbol{\lambda}^T \nabla \boldsymbol{g}\left(\boldsymbol{x}^*\right) &= \boldsymbol{0} \\ \boldsymbol{\lambda}^T \boldsymbol{g}\left(\boldsymbol{x}^*\right) &= \boldsymbol{0} \end{aligned} \tag{4.1-4}$$

$$\boldsymbol{g}(\boldsymbol{x}^*) \le 0$$
$$\boldsymbol{\lambda} \ge \mathbf{0}.$$

These conditions are derived by differentiating the Lagrangian function of the constrained minimization problem

$$L(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) \tag{4.1-5}$$

and applying the conditions

$$\nabla^T f \, \partial \boldsymbol{x}^* \ge 0 \text{ (optimality)} \tag{4.1-6}$$

and

$$\nabla^T \overline{\boldsymbol{g}} \, \partial \boldsymbol{x}^* \le \mathbf{0} \text{ (feasibility)} \tag{4.1-7}$$

to a perturbation $\partial \boldsymbol{x}^*$.

λ_j are the Lagrange multipliers which may be nonzero only if the corresponding constraint is active, i.e. $g_j(\boldsymbol{x}^*) = 0$.

For x^* to be a local constrained minimum, the Hessian of the Lagrangian function, $\nabla^2 f(\boldsymbol{x}^*) + \lambda^T \nabla^2 \overline{\boldsymbol{g}}(\boldsymbol{x}^*)$ on the subspace tangent to the active constraint $\overline{\boldsymbol{g}}$ must be positive definite at x^*.

These conditions are not used explicitly in LS-OPT and are not tested for 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.

4.2 Normalization of constraints and variables

It is a good idea to eliminate large variations in the magnitudes of design variables and constraints by normalization.

In LS-OPT, the typical constraint is formulated as follows:

$$L_j \le g_j(\boldsymbol{x}) \le U_j \quad ; \quad j = 1,2,\ldots,m \tag{4.2-1}$$

which, when normalized becomes:

$$\frac{L_j}{g_j(\boldsymbol{x_0})} \le \frac{g_j(\boldsymbol{x})}{g_j(\boldsymbol{x_0})} \le \frac{U_j}{g_j(\boldsymbol{x_0})} \quad ; \quad j = 1,2,\ldots,m \tag{4.2-2}$$

where $\boldsymbol{x}_0$ is the starting vector. The normalization is done internally.

The design variables have been normalized internally by scaling the design space [$\boldsymbol{x}_L$; $\boldsymbol{x}_U$] to [0;1], where $\boldsymbol{x}_L$ is the lower and $\boldsymbol{x}_U$ the upper bound. The formula

$$\xi_i = \frac{x_i - x_{iL}}{x_{iU} - x_{iL}} \tag{4.2-3}$$

is used to transform each variable x_i to a normalized variable, ξ_i.

When using LS-OPT to minimize maximum violations, the constraints must be normalized by the user. This method is chosen to give the user the freedom in selecting the importance of different responses when e.g. performing parameter identification. Section 5.3.2 will present this application in more detail.

4.3 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 and certain types of nonlinear analysis.

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 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].

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.

4.4 Core optimization algorithm (LFOPC)

The optimization algorithm used to solve the approximate subproblem is the LFOPC algorithm of Snyman [5]. It is a gradient method that generates a dynamic trajectory path, from any given starting point, towards a local optimum. This method differs conceptually from other gradient methods, such as SQP, in that no explicit line searches are performed.

The original leap-frog method [9] for unconstrained minimization problems seeks the minimum of a function of n variables by considering the associated dynamic problem of a particle of unit mass in an n-dimensional conservative force field, in which the potential energy of the particle at point $\boldsymbol{x}(t)$ at time t is taken to be the function $f(\boldsymbol{x})$ to be minimized.

The solution to the unconstrained 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 premultiplied 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 20.4). In addition, the step size of the algorithm and 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 the [0; 1] interval. The default parameters in LFOPC (as listed in Section 20.4) should therefore be adequate. The termination criteria are also listed in Section 20.4.

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.

4.5 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.

4.5.1 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.

4.5.2 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.

4.5.3 Algorithm

The steps in a simple genetic algorithm are illustrated with the help of Figure 4-1.

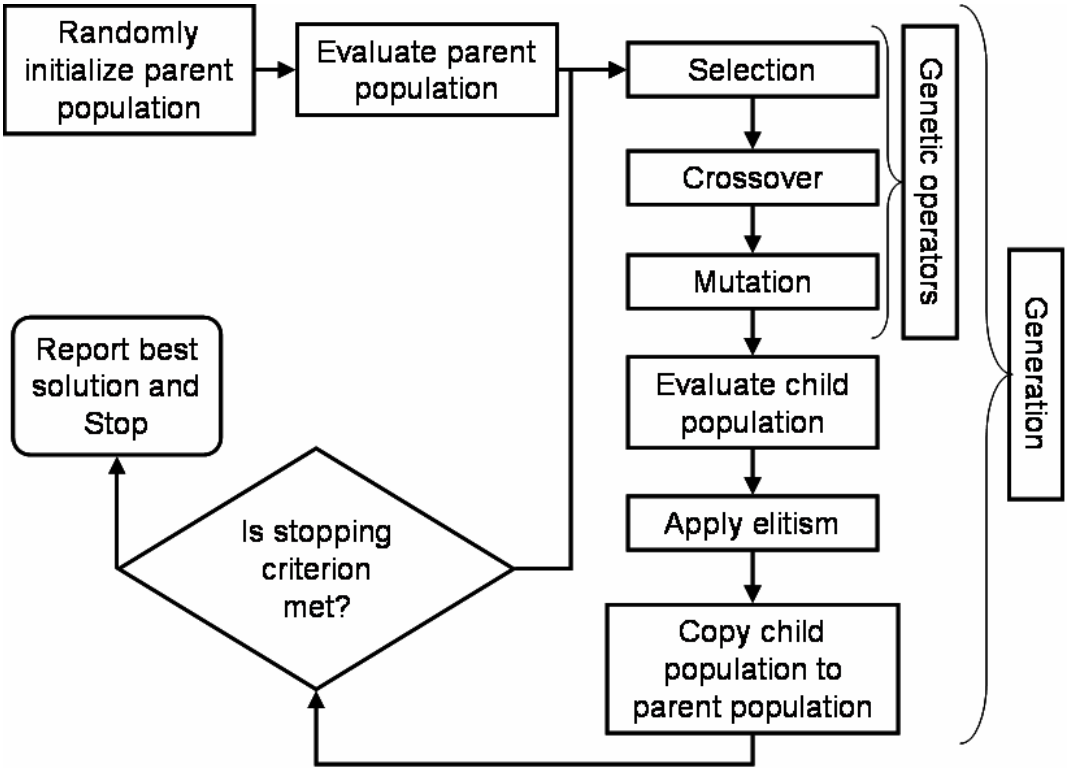


Figure 4-1: Simple genetic algorithm.

Firstly, problem-specific GA parameters like population size N_{pop}, type of encoding, number of bits per variables for binary coding, number of generations are defined.

Initialization

Next, the population is randomly initialized i.e., binary chromosomes or real variable vectors for N_{pop} individuals are generated randomly.

Function evaluation
For binary encoding, each chromosome (binary string) is decoded to corresponding design variable vector. Next, objective functions, constraints, and constraint violation of each individual in parent population is evaluated and accordingly fitness of each individual is assigned.

Selection or reproduction operator
Selection operator is used to identify individuals with high fitness and to form a mating pool of size N_{pop}. This operator reduces diversity in the population by filtering out low fitness schema. Many reproduction operators are introduced in literature. Three selection operators implemented in LS-Opt are described below.

Tournament selection. In tournament selection, 'N_{tourn}' (N_{tourn} is tournament size) individuals from a population, selected at random, participate in a tournament. The individual with the largest fitness is declared the winner. Mostly, practitioners use $N_{tourn} = 2$. Increasing the tournament size 'N_{tourn}' increases selection pressure and might lose diversity in the population that is required to drive the search.

Roulette wheel or proportionate reproduction. In this selection approach, each individual is assigned a probability of selection based on its fitness value. In a population of N_{pop} individuals, the selection probability of the i^{th} individual is

$$P_i = F_i \Big/ \sum_{j=1}^{N_{pop}} F_j \,, \qquad (4.5\text{-}1)$$

where F_i is the fitness of i^{th} individual. High fitness individuals have a high probability of getting selected. This scheme is implemented by considering a roulette wheel with circumference marked by the fitness of each individual. One individual per spin of the wheel is selected. Then, the expected number of copies of the i^{th} individual in the mating pool can be estimated as

$$N_i = F_i / \overline{F} \,; \overline{F} = \frac{1}{N_{pop}} \sum_{j=1}^{N_{pop}} F_j . \qquad (4.5\text{-}2)$$

This selection operator has a higher selection pressure compared to the tournament selection and can lead to a premature convergence to local optima.

Stochastic universal sampling. The roulette wheel selection operator is often noisy because of multiple spins that correspond to round-off errors in computer simulations. To reduce this noise, it was suggested to use a single spin of the wheel with N_{pop} equi-spaced pointers. This operator also has a high selection pressure.

Crossover
Crossover is the main exploration operator of genetic search. In this operator, μ randomly selected parents mate with a probability (P_c: crossover probability) to create λ children. These children share the attributes from all parents such that they may be better or worse individuals. There are two prominent strategies to create children: i) $(\mu + \lambda)$ strategy selects best individuals from parents and children, and ii) (μ, λ) strategy replaces parents with children irrespective of their fitness values. LS-OPT has adopted a $(2,2)$ strategy for crossover such that two parents create two children and children replace parents in the new generation. If parents do not create children, they are passed to the next generation.
There are many crossover operators in literature. A few popular crossover operators that have been shown to perform reasonably well are available in LS-OPT. A brief description of these operators is as follows.

Single point binary crossover
This crossover operator is used for binary encoding of the individuals. Two parents and a mating site are randomly selected. All genes right to the mating sites are swapped between two parents.

Uniform binary crossover
This crossover operator is also used for binary encoded individuals. For a randomly selected parent pair, genes are swapped based on a flip of a coin for each gene in the chromosome.

Simulated binary real crossover (SBX)
This crossover operator, introduced by Deb and Agrawal in 1995 [12], is used with real encoding i.e., real variables are used as genes. This crossover emulates the single point binary crossover by assigning a probability distribution to each parent. Two children are created from two parents using that probability distribution such that the mean of parents and children are the same. The probability distribution is controlled by a distribution index η_c such that large value of η_c creates children near parents and small value of η_c creates children far from parents. Deb and Beyer [13] showed that SBX possesses self-adaptation capabilities.

Blend real crossover (BLX-α)
This crossover operator was introduced by Eshelman and Schaffer in 1993 [14]. In this crossover, a child x is created from two parents $x^{(1)}$ and $x^{(2)}$ ($x^{(2)} > x^{(1)}$) by randomly selecting a value from the interval $[x^{(1)} - \alpha(x^{(2)} - x^{(1)}), x^{(2)} + \alpha(x^{(2)} - x^{(1)})]$. Typically, α is taken as 0.5.

Mutation
Mutation is carried out with a mutation probability (P_m) to bring random changes in the individuals. This operator is very useful when population has lost diversity and the search has become stagnant. Then mutation can help improve diversity in the solutions. The mutation operators for binary and real encoding are given as follows:

Simple binary mutation
In simple binary mutation of an individual, a bitwise mutation is carried out by changing a '0' to '1' or vice-versa with a small mutation probability P_m. Typically P_m is taken as the inverse of chromosome length such that on an average, one gene (bit) per chromosome is changed.

Real mutation
As was used for the SBX operator, a distribution (defined by mutation distribution index) around each variable is specified and a random variable is selected from that distribution. Large values of the distribution index are recommended to create a child near the parent.

A complete cycle of selection, crossover, and mutation would result in a child population. The population size is kept constant for both parent and child populations.

Elitism in simple genetic algorithm
Due to the disruptive nature of exploration operators, high fitness individuals may get lost while creating a child population from the parent population. Sometimes, it is advantageous to keep these high fitness individuals to preserve favorable genetic information (schema). This process of artificially saving the best individuals is called elitism. To implement this process, the parent and child populations are ranked separately. The worst individuals in the child population are replaced by the best individuals from the parent population. The *number of elites* should be carefully chosen because a large number of elite solutions may drive the search to local optima by reducing the diversity in the population. On the other hand, too few elites may slow the convergence because favorable schema would spread at a slow rate.

After applying elitism, the child population is transferred to the parent population. The best individual found in the search process is preserved at each generation.

Stopping criterion
Many criteria have been specified in literature to terminate the GA search process. Some researchers have suggested stopping the search when there is no improvement in the last few generations. However, the most common stopping criterion is the fixed number of generations or function evaluations. A user defined number of generations is used as the stopping criterion in LS-OPT.

At the end of simple genetic algorithm, the best individual (among all searched individuals) is reported as the optimal solution. If enough processing capabilities is carried out, the reported best individual would represent the global optimal solution.

4.6 Sequential response surface method (SRSM)

The purpose of the SRSM method is to allow convergence of the 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 4-2 shows the possible adaptation of the subregion.

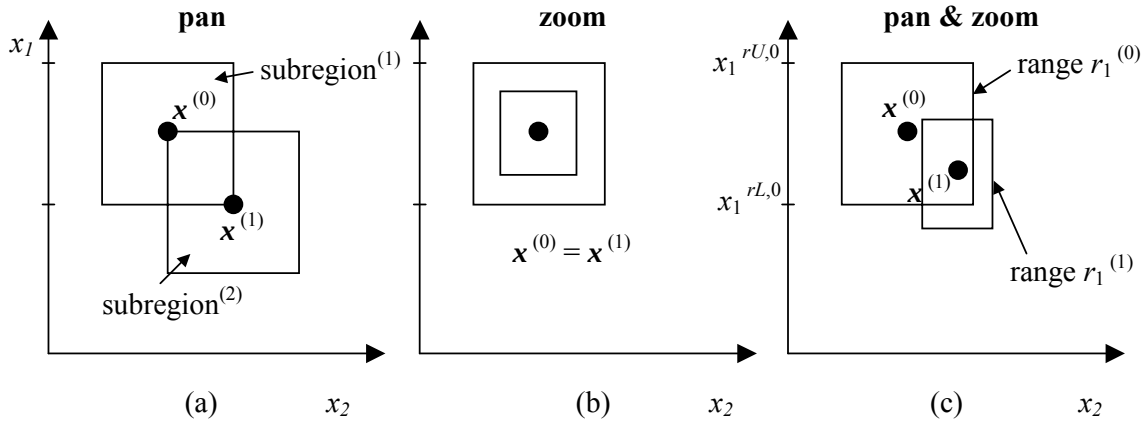


Figure 4-2: Adaptation of subregion in SRSM: (a) pure panning, (b) pure zooming and (c) a combination of panning and zooming

The starting point $\boldsymbol{x}^{(0)}$ 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

$$x_i^{rL,0} = x_i^{(0)} - 0.5r_i^{(0)} \text{ and } x_i^{rU,0} = x_i^{(0)} + 0.5r_i^{(0)} \qquad i = 1,\ldots n \tag{4.6-1}$$

where n is the number of design variables. The modification of the ranges on the variables for the next iteration depends on the oscillatory nature of the solution and the accuracy of the current optimum.

Oscillation: A *contraction parameter* γ is firstly determined based on whether the current and previous designs $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(k-1)}$ are on the opposite or the same side of the region of interest. Thus an *oscillation indicator c* may be determined in iteration *k* as

$$c_i^{(k)} = d_i^{(k)} d_i^{(k-1)} \tag{4.6-2}$$

where

$$d_i^{(k)} = 2\Delta x_i^{(k)} / r_i^{(k)}; \quad \Delta x_i^{(k)} = x_i^{(k)} - x_i^{(k-1)}; \quad d_i^{(k)} \in [-1;1] \tag{4.6-3}$$

The oscillation indicator (purposely omitting indices *i* and *k*) is normalized as $\hat{c}$ where

$$\hat{c} = \sqrt{|c|}\, sign(c)\,. \tag{4.6-4}$$

The contraction parameter γ is then calculated as

$$\gamma = \frac{\gamma_{\text{pan}}(1+\hat{c}) + \gamma_{\text{osc}}(1-\hat{c})}{2}\,. \tag{4.6-5}$$

See Figure 4-3. The parameter γ_{osc} is typically 0.5-0.7 representing shrinkage to dampen oscillation, whereas γ_{pan} represents the pure panning case and therefore unity is typically chosen.

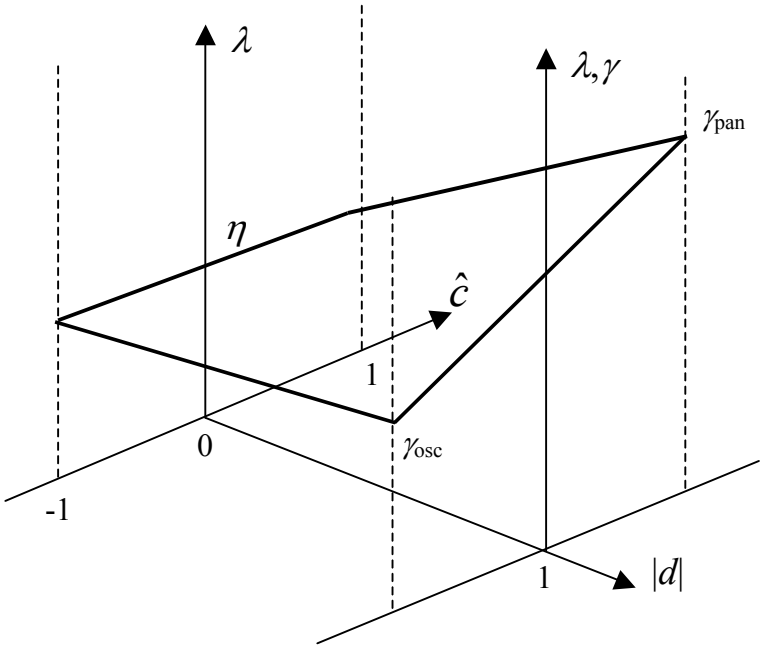


Figure 4-3: The sub-region contraction rate λ as a function of the oscillation indicator $\hat{c}$ and the absolute move distance $|d|$

Accuracy: The accuracy is estimated using the proximity of the predicted optimum of the current iteration to the starting (previous) design. The smaller the distance between the starting and optimum designs, the more rapidly the region of interest will diminish in size. If the solution is on the bound of the region of interest,

the optimal point is estimated to be beyond the region. Therefore a new subregion, which is centered on the current point, does not change its size. This is called *panning* (Figure 4-2(a)). If the optimum point coincides with the previous one, the subregion is stationary, but reduces its size (*zooming*) (Figure 4-2(b)). Both panning and zooming may occur if there is partial movement (Figure 4-2(c)). The range $r_i^{(k+1)}$ for the new subregion in the (k + 1)-th iteration is then determined by:

$$r_i^{(k+1)} = \lambda_i r_i^{(k)}; \quad i = 1,\ldots,n; \quad k = 0,\ldots,niter \tag{4.6-6}$$

where λ_i represents the *contraction rate* for each design variable. To determine λ_i, $d_i^{(k)}$ is incorporated by scaling according to a *zoom parameter* η that represents pure zooming and the contraction parameter γ to yield the contraction rate

$$\lambda_i = \eta + \left|d_i^{(k)}\right|(\gamma - \eta) \tag{4.6-7}$$

for each variable (see Figure 4-3).

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 20.3.1 for the setting of parameters in the iterative Sequential Response Surface Method.

4.7 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 20.6.

4.7.1 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.

4.7.2 Sequential strategy

In this approach, sampling is done sequentially. A small number of points is chosen for each iteration and multiple iterations are requested. The approach has the advantage that the iterative process can be stopped as soon as the metamodels or optimum points have sufficient accuracy. It was demonstrated in Reference [16] that, for Space Filling, the Sequential approach had similar accuracy compared to the Single Stage approach, i.e. 10 × 30 points added sequentially is almost as good as 300 points. Therefore both the Single Stage and

Sequential Methods are good for design exploration using a surrogate model. For instance when constructing a Pareto Optimal Front, the use of a Single Stage or Sequential strategy is recommended in lieu of a Sequential strategy with domain reduction (see Section 4.7.3).

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.

4.7.3 Sequential strategy with domain reduction

This approach is the same as that in 4.7.2 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 bound 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 4.7.2 *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 4.6). 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 4.6 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 4.6). 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 5.3).

4.8 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.

4.8.1 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
   a. $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in 'all' objectives, i.e. $f_j(\mathbf{x}^{(1)}) \not> f_j(\mathbf{x}^{(2)}), \quad j \in [1,2,...,M]$.
   b. $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in 'at least one' objective, i.e., $f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)}), \wedge j \in [1,2,...,M]$.

If neither of the two solutions dominates the other, both solutions are non-dominated with respect to each other. An individual **s** is considered non-dominated with respect to a set of solutions ***S***, if no solution in ***S*** dominates **s**.

4.8.2 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.

4.8.3 Pareto optimal set

The set of all Pareto optimal solutions is the Pareto optimal set for the given problem.

4.8.4 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.

4.8.5 Ranking

Most MOGA search methods assign rank to different individuals based on non-domination criterion. This ranking is used to govern the search process. A rank of one is considered the best rank and low fitness individuals are assigned low ranks (large values of rank are low). Different individuals in a population are assigned rank as follows:

1. Initialize *rnk* = 1. Define a set of individuals **S**, same as the population.
2. Run a non-domination check on all individuals in **S**.
3. All non-dominated individuals are assigned rank = *rnk*.
4. *rnk* = *rnk* + 1.
5. Remove all non-dominated individuals from **S**.
6. If $\mathbf{S} \neq \Phi$, repeat Step 2, else stop.

Note that many individuals can have the same rank.

Different concepts discussed here are illustrated using a two-objective unconstrained minimization problem in Figure 4-4. 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 it 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.

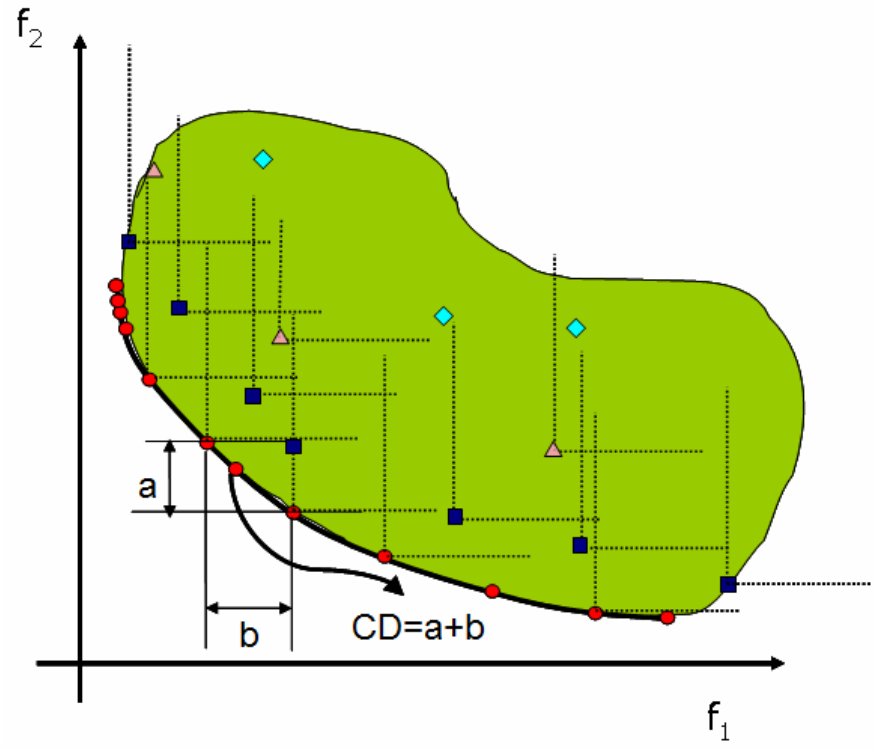


Figure 4-4: Illustration of non-domination criterion, Pareto optimal set, and Pareto optimal front.

4.8.6 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.

4.8.7 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 a problem of Pareto drift. To avoid that problem, Goel et al. [21] proposed maintaining an external archive.

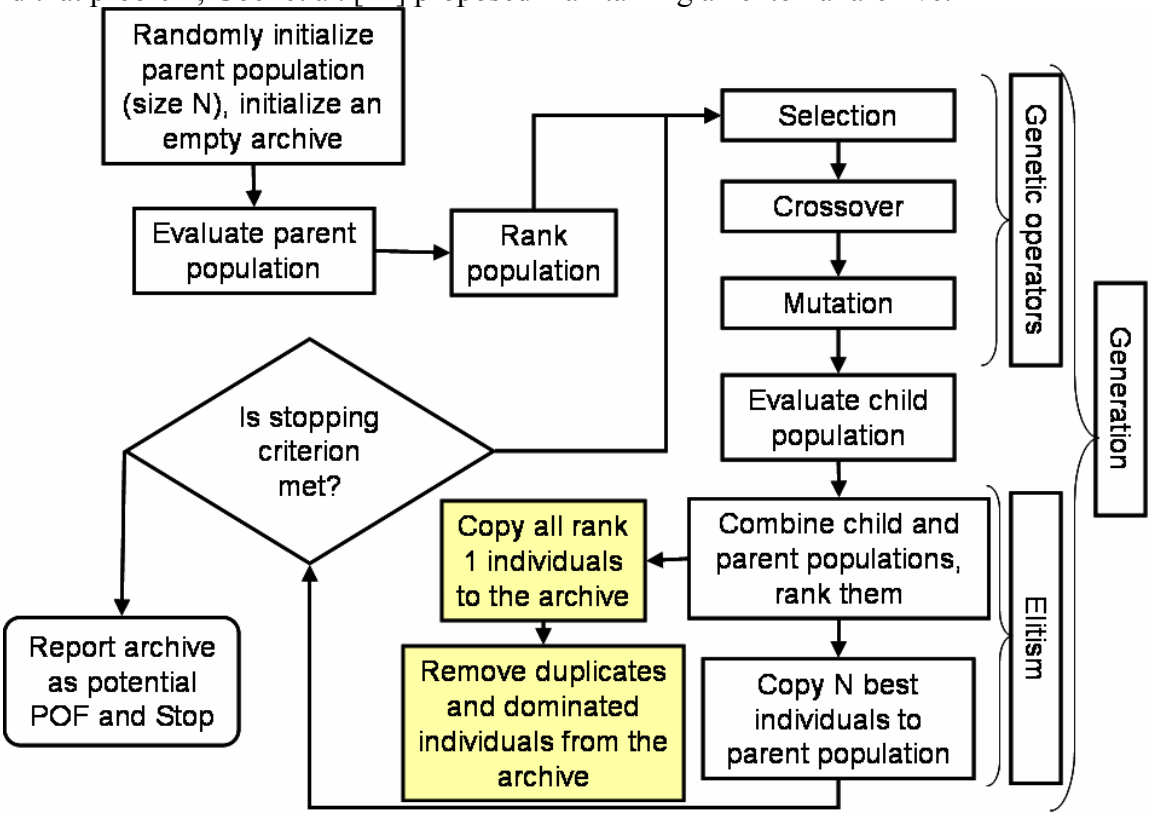


Figure 4-5: Elitist non-dominated sorting genetic algorithm (NSGA-II). The shaded blocks are not the part of original NSGA-II but additions to avoid Pareto drift.

The implementation of this archived NSGA-II is shown in Figure 4-5, 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.

Elitism in NSGA-II

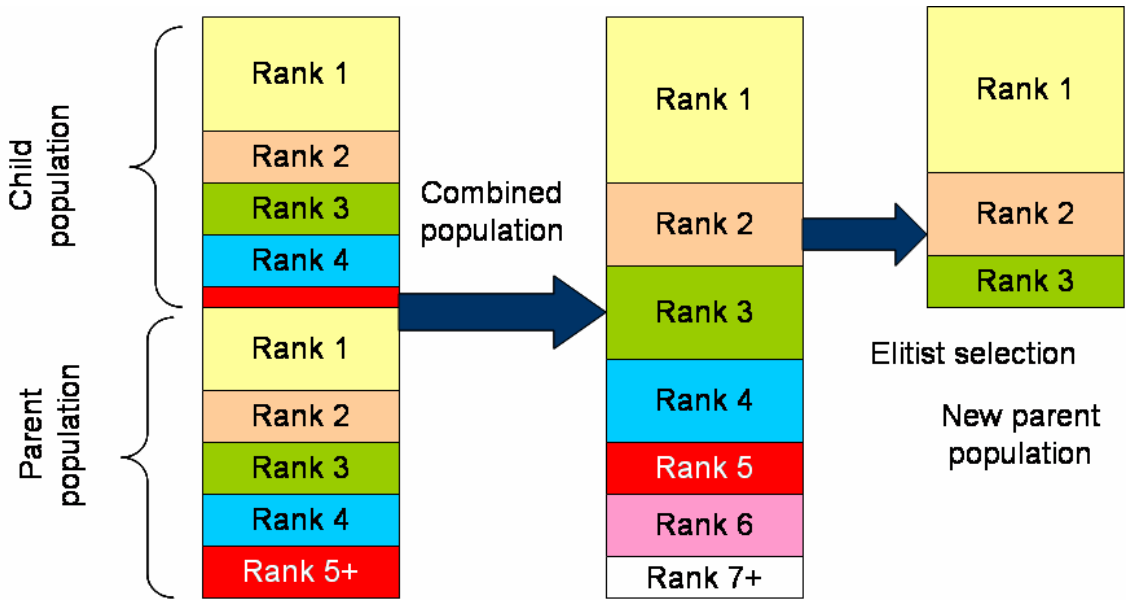


Figure 4-6: Elitism in NSGA-II.

Elitism is applied to preserve the best individuals. The mechanism used by NSGA-II algorithm for elitism is illustrated in Figure 4-6. After combining the child and parent populations, there are $2N_{pop}$ individuals. This combined pool of members is ranked using non-domination criterion such that there are n_i individuals with rank i. The crowding distance of individuals with the same rank is computed. Steps in selecting N_{pop} individuals are as follows:

1. Set $i = 1$, and number of empty slots $N_{slots} = N_{pop}$.
2. If $n_i < N_{slots}$,
    a. Copy all individuals with rank 'i' to the new parent population.
    b. Reduce the number of empty slots by n_i: $N_{slots} = N_{slots} - n_i$.
    c. Increment 'i': $i=i+1$.
    d. Return to Step 2.
3. If $n_i > N_{slots}$,
    a. Sort the individuals with rank 'i' in decreasing order of crowding distance.
    b. Select N_{slots} individuals.
    c. Stop

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 4-4, 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.

4.9 Discrete optimization

4.9.1 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.

4.9.2 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 describes 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 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.

4.9.3 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.

4.9.4 Discrete optimization algorithm: genetic algorithm

A GA (genetic algorithm, Section 4.5) 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.

4.9.5 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.

4.9.6 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 are therefore very important. The accuracy of the metamodels are improved by a sequential response surface method (SRSM) (see Section 4.6), 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.

4.10 Summary of the optimization process

The following tasks may be identified in the process of an optimization cycle using response surfaces.

Table 4.10-1: 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:

4.10.1 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 3.1) and Radial Basis Function networks (Section 3.1.3) 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 neural nets). The metamodel will be built using the total number of points.

See Section 4.7 on sequential strategies for optimization and design exploration.

4.10.2 Design exploration

Conduct one iteration, usually by utilizing second-order approximations with a large range. Then assess the adequacy of the surfaces using the error parameters. If the user is satisfied with the accuracy of the metamodel, a trade-off study can be conducted to visualize how a design might change in response to modified design criteria in the design formulation. Solutions to the trade-off optimization problem falling outside the region of interest are connected by dotted lines in the GUI to indicate extrapolation of the metamodel. To avoid extrapolation, points can be added to the design space and adaptable response surfaces such as neural networks can be used to improve prediction. See also Section 4.7.

4.11 REFERENCES

[1] Forsberg, J. *Simulation Based Crashworthiness Design – Accuracy Aspects of Structural optimization using Response Surfaces*. Thesis No. 954. Division of Solid Mechanics, Department of Mechanical Engineering, Linköping University, Sweden, 2002.

[2] Luenberger, D.G. *Linear and Nonlinear Programming*. Second Edition. Addison Wesley, 1984.

[3] Arora, J.S. Sequential linearization and quadratic programming techniques. In *Structural Optimization: Status and Promise*, Ed. Kamat, M.P., AIAA, 1993.

[4] Thanedar, P.B., Arora, J.S., Tseng, C.H., Lim, O.K., Park, G.J. Performance of some SQP algorithms on structural design problems. *International Journal for Numerical Methods in Engineering*, 23, pp. 2187-2203, 1986.

[5] Snyman, J.A. The LFOPC leap-frog algorithm for constrained optimization. *Comp. Math. Applic.*, 40, pp. 1085-1096, 2000.

[6] Barthelemy, J.-F. M. Function Approximation. In *Structural Optimization: Status and Promise*, Ed. Kamat, M.P., 1993.

[7] Box., G.E.P., Draper, N.R. *Empirical Model Building and Response Surfaces*. Wiley, New York, 1987.

[8] Hajela, P., Berke L. Neurobiological computational models in structural analysis and design. *Proceedings of the 31st AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference*, Long Beach, CA, April, 1990.

[9] Snyman, J.A. An improved version of the original leap-frog dynamic method for unconstrained minimization LFOP1(b). *Appl. Math. Modelling*, 7, pp. 216-218, 1983.

[10] Holland, J.H., *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[11] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[12] Deb, K., Agrawal, R.B., Simulated binary crossover for continuous search space. *Complex Systems*, 9, 115-148, 1995.
[13] Deb, K., Beyer, H.-G., Self adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation Journal*, 9(2), 197-221, 2001.
[14] Eshelman, L.J., Schaffer, J.D., Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms-2*. San Mateo, CA: Morgan Kaufman, 187-202, 1993.
[15] Stander, N., Craig, K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization, *Engineering Computations*, 19(4), pp. 431-450, 2002.
[16] Stander, N. Goel, T. Metamodel sensitivity to sequential sampling strategies in crashworthiness design. *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,Victoria, British Columbia, Canada,* Sep 10-12, 2008. *Submitted*.
[17] Deb, K., *Multiobjective Optimization using Evolutionary Algorithms*, Wiley Chichester UK, 2001.
[18] Coello, C.A.C., *Evolutionary Algorithms for Solving Multi-Objective Problems* (Genetic and Evolutionary Computation), Springer, 2007.
[19] Miettinen, K.M., *Nonlinear Multi Objective Optimization*, Kluwer, 1999.
[20] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6(2), 181-197, 2002. an earlier version was presented in *Parallel Problem Solving from Nature VI Conference*, Paris, France, Sept 2000.
[21] Goel, T., Vaidyanathan, R., Haftka, R.T., Queipo, N.V., Shyy, W., Tucker, K., Response surface approximation of Pareto optimal front in multi-objective optimization. *Computer Methods in Applied Mechanics and Engineering*, 196(4-6), 879-893, 2007. (also presented at *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September 2004, Albany, NY).

5. Applications of Optimization

5.1 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 0, focused on finding multiple trade-offs, known as Pareto optimal solutions, using multi-objective genetic algorithms [Section 0]. The advantage of this method is that one can find many trade-off designs and the designer does not have to a prioi 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

$$p[\boldsymbol{f}(\boldsymbol{x})] \tag{5.1-1}$$

subject to the inequality constraint functions

$$L_j \le g_j(\boldsymbol{x}) \le U_j \quad ; \quad j = 1,2,\ldots,m$$

The preference function p can be formulated to incorporate target values of objectives.

Two methods for achieving this are given:

5.1.1 Euclidean Distance Function

Designs often contain objectives that are in conflict so that they cannot be achieved simultaneously. If one objective is improved, the other deteriorates and *vice versa*. The preference function $p[\boldsymbol{f}(\boldsymbol{x})]$ combines various objectives f_i. The Euclidean distance function allows the designer to find the design with the smallest distance to a specified set of target responses or design variables:

$$p = \sqrt{\sum_{i=1}^{p} W_i \left[\frac{f_i(\boldsymbol{x}) - F_i}{\Gamma_i} \right]^2} \tag{5.1-2}$$

The symbols F_i represent the target values of the responses. A value Γ_i is used to normalize each response i. Weights W_i are associated with each quantity and can be chosen by the designer to convey the relative importance of each normalized response.

5.1.2 Maximum distance

Another approach to target responses is by using the maximum distance to a target value

$$p = \max_i \left[\frac{|f_i(\mathbf{x}) - F_i|}{|\Gamma_i|} \right] \tag{5.1-3}$$

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

$$p = \left[\sum_{i=1}^{p} \left(\frac{|f_i(\mathbf{x}) - F_i|}{|\Gamma_i|} \right)^r \right]^{1/r} \tag{5.1-4}$$

with $r = 2$ for the Euclidean metric and $r \to \infty$ for the min.-max. formulation (Tchebysheff metric).

The approach for dealing with the Tchebysheff formulation differs somewhat from the explicit formulation. The alternative formulation becomes:

$$\begin{aligned} &\text{Minimize} && e && (5.1\text{-}5)\\ &\text{subject to} \end{aligned}$$

$$\frac{F_i}{\Gamma_i} - (1-\alpha_{jL})e \le \frac{f_i(\mathbf{x})}{\Gamma_i} \le \frac{F_i}{\Gamma_i} + (1-\alpha_{jU})e \quad ; \quad i = 1,\ldots,p, \quad j=1,..,m$$

$$e \ge 0$$

In the above equation, Γ_i is a normalization factor, e represents the constraint violation or target discrepancy and α represents the strictness factor. If $\alpha = 0$, the constraint is slack (or soft) and will allow violation. If $\alpha = 1$, the constraint is strict (or hard) and will not allow violation of the constraint.

The effect of distinguishing between strict and soft constraints on the above problem is that the maximum violation of the soft constraints is minimized. Because the user is seldom aware of the feasibility status of the design problem at the start of the investigation, the solver will automatically solve the above problem first to find a feasible region. If the solution to e is zero (or within a small tolerance) the problem has a feasible region and the solver will immediately continue to minimize the design objective using the feasible point as a starting point.

A few points are notable:

- 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.
- Soft constraints will always be *strictly* satisfied if a feasible design is possible.
- If a feasible design is not possible, the most feasible design will be computed.
- 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.

5.2 Multidisciplinary Design Optimization

There is increasing interest in the coupling of other disciplines into the optimization process, especially for complex engineering systems like aircraft and automobiles [2]. The aerospace industry was the first to embrace multidisciplinary design optimization (MDO) [3], because of the complex integration of aerodynamics, structures, control and propulsion during the development of air- and spacecraft. The automobile industry has followed suit [4]. In [4], the roof crush performance of a vehicle is coupled to its Noise, Vibration and Harshness (NVH) characteristics (modal frequency, static bending and torsion displacements) in a mass minimization study.

Different methods have been proposed when dealing with MDO. The conventional or standard approach is to evaluate all disciplines simultaneously in one integrated objective and constraint set by applying an optimizer to the multidisciplinary analysis (MDA), similar to that followed in single-discipline optimization. The standard method has been called multidisciplinary feasible (MDF), as it maintains feasibility with

respect to the MDA, but as it does not imply feasibility with respect to the disciplinary constraints, is has also been called fully integrated optimization (FIO). A number of MDO formulations are aimed at decomposing the MDF problem. The choice of MDO formulation depends on the degree of coupling between the different disciplines and the ratio of shared to total design variables [5]. It was decided to implement the MDF formulation in this version of LS-OPT as it ensures correct coupling between disciplines albeit at the cost of seamless integration being required between different disciplines that may contain diverse simulation software and different design teams.

In LS-OPT, the user has the capability of assigning different variables, experimental designs and job specification information to the different solvers or disciplines. The file locations in Version 2 have been altered to accommodate separate `Experiments`, `AnalysisResults` and `DesignFunctions` files in each solver's directory. An example of job-specific information is the ability to control the number of processors assigned to each discipline separately. This feature allows allocation of memory and processor resources for a more efficient solution process.

Refer to the user's manual (Section 19.1) for the details of implementing an MDO problem. There is one crashworthiness-modal analysis case study in the examples chapter (Section 22.6).

5.3 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.

5.3.1 Nonlinear regression: minimizing Mean Squared Error (MSE)

Figure 5-1 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 14.1.1) and the curve $z(t)$ to obtain a plot that is comparable to G. Each function $f(\mathbf{x},z_p)$ is internally represented by a response surface so that a typical curve $f(\mathbf{x},z)$ is represented by P internal response surfaces.

In Figure 5-1, seven regression points are shown. The residuals at these points are combined into a Mean Squared Error norm:

$$\varepsilon = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\boldsymbol{x}) - G_p}{s_p} \right)^2 = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{e_p(\boldsymbol{x})}{s_p} \right)^2 \qquad (5.3\text{-}1)$$

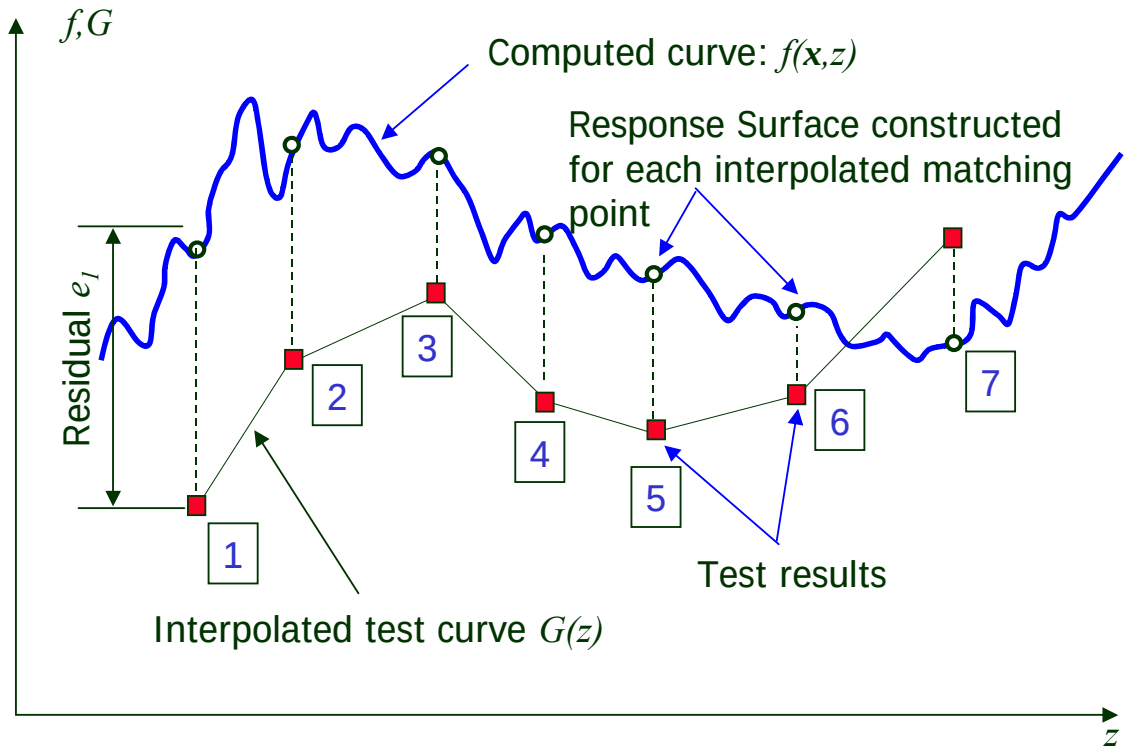


Figure 5-1: Entities in Mean Squared Error formulation

The MSE norm is based on a series of P regression points beginning at the start point z_1 and terminating at the end point z_P (see Figure 5-1). The s_p, p=1,...,P are residual scale factors and the W_p, p=1,...,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 Section 22.5.

5.3.2 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:

Minimize e, (5.3-2)

subject to

$$\left|\frac{f_p(\boldsymbol{x}) - G_p}{s_p}\right| \le e \quad ; \quad p = 1,\ldots,P$$

$$e \ge 0$$

This formulation is automatically activated in LS-OPT when specifying both the lower and upper bounds of f_p / s_p equal to G_p / s_p. There is therefore no need to define an objective function. This is due to the fact that an auxiliary problem is automatically solved internally whenever an infeasible design is found, ignoring the objective function until a feasible design is obtained. When used in parameter identification, the constraint set is in general never completely satisfied due to the typically over-determined systems used.

Since s_p defaults to 1.0, the user is responsible for the normalization in the maximum violation formulation. This can be done by e.g. using the target value to scale the response $f(\boldsymbol{x})$ so that:

$$\left|\frac{f_p(\boldsymbol{x})}{G_p} - 1\right| \le e; \quad p = 1\ ,\ldots,\ P$$

$$e \ge 0$$

Omitting the scaling may cause conditioning problems in some cases, especially where constraint values differ by several orders of magnitude.

This option will also be automated in future versions.

5.3.3 Nonlinear regression: confidence intervals

Assume the nonlinear regression model:

$$G(t) = F(t, \boldsymbol{x}) + \varepsilon$$

where the measured result G is approximated by F and $\boldsymbol{x}$ is a vector of unknown parameters. The nonlinear least squares problem is obtained from the discretization:

$$\min_{\boldsymbol{x}} \frac{1}{P}\sum_{p=1}^{P}(G_p - F_p(\boldsymbol{x}))^2$$

is solved to obtain x^*. The variance σ^2 is estimated by

$$\hat{\sigma}^2 = \frac{\left\|\boldsymbol{G} - \boldsymbol{F}(\boldsymbol{x}^*)\right\|^2}{P - n}$$

where $\boldsymbol{F}$ is the P-vector of function values predicted by the model and n is the number of parameters. The 100(1-α)% confidence interval for each x^*_i is:

$$\left(\left[x_i : \left|x_i^* - x_i\right| \le \sqrt{\hat{C}_{ii}}\, t_{P-n}^{\alpha/2}\}\right]\right)$$

where

$$\hat{\boldsymbol{C}} := \hat{\sigma}^2 \left(\nabla \boldsymbol{F}\left(x^*\right)^T \nabla \boldsymbol{F}\left(x^*\right)\right)^{-1}$$

and $t_{P-n}^{\alpha/2}$ is the Student t-distribution for α.

$\nabla\boldsymbol{F}$ is the $P \times n$ matrix obtained from the n derivatives of the P response functions representing P points at the optimum x. The optimal solution is therefore calculated first, followed by the confidence interval.

A critical issue is to ensure that $\nabla\boldsymbol{F}$ 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.

5.4 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 $z_i, i = 1,\ldots,n$ in addition to the *controlled* variables $y_j, j = 1,\ldots,m$. The controlled variables can be set by the designer and therefore optimized by the program. The uncontrollable variables are determined by the random variability of manufacturing processes, loadings, materials, etc. Controlled and uncontrollable variables can be independent, but can also be associated with one another, i.e. a controlled variable can have an uncontrollable component.

The methodology requires three features:

1. The introduction of a constant range ρ of the region of interest for the uncontrollable variables. This constant represents the *possible variation* of each uncontrollable parameter or variable. In LS-OPT this is introduced by specifying a lower limit on the range as being equal to the initial range ρ. The lower and upper bounds of the design space are set to $\pm\rho/2$ for the uncontrollable variables.

2. The controlled and uncontrollable variables must be separated as minimization and maximization variables. The objective will therefore be minimized with respect to the controlled variables and maximized with respect to the uncontrollable variables. This requires a special flag in the optimization algorithm and the formulation of Equation (2.1) becomes:

$$\min_{\boldsymbol{y}}\{\max_{\boldsymbol{z}} f(\boldsymbol{y},\boldsymbol{z})\}, \boldsymbol{y}\in\Re^p, \boldsymbol{z}\in\Re^q \tag{5.4-1}$$

   subject to

$$g_j(\boldsymbol{y},\boldsymbol{z})\le 0 \quad ; \quad j=1,2,\ldots,l$$

   The algorithm remains a minimization algorithm but with modified gradients:

$$\nabla_{\boldsymbol{y}}^{\text{mod}} := \nabla\boldsymbol{y}$$

$$\nabla_{\boldsymbol{z}}^{\text{mod}} := -\nabla\boldsymbol{z}$$

   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.

5.5 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. In the FOSM method, the standard deviation of a response is computed using the metamodel gradients and variable standard deviations; no additional computational costs are therefore incurred to compute the reliability information. See Section 6.4.3 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.

The standard deviations are assumed to be constant over the sub-region, but the method should converge to a fixed value of the standard deviation if an iterative scheme is used.

Discrete variables are allowed in RBDO. The mixed-discrete optimization will be carried out considering the probabilitistic bounds on the constraints.

The methods are described in more detail in Section 19.3 with an example in Section 22.2.11 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 be 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.

5.6 REFERENCES

[1] Daberkow, D.D. Mavris, D.N. An investigation of metamodeling techniques for complex systems design. *Symposium on Multidisciplinary Analysis and Design*, Atlanta, October 2002.

[2] Lewis, K., Mistree, F. The other side of multidisciplinary design optimization: accommodating a mutiobjective, uncertain and non-deterministic world. *Engineering Optimization*, 31, pp. 161-189, 1998.

[3] Simpson, T.W. *A Concept Exploration Method for Product Family Design*. Ph.D. Thesis, Georgia Institute of Technology, 1998.

[4] Sobieszczanski-Sobieski, J., Kodiyalam, S., Yang, R-.J. Optimization of car body under constraints of noise, vibration, and harshness (NVH), and crash. *AIAA Paper 2000-1521*, 2000.

[5] Zang, T.A., Green, L.L., Multidisciplinary design optimization techniques: Implications and opportunities for fluid dynamics research, *AIAA Paper 99-3798*, 1999.

6. Probabilistic Fundamentals

6.1 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:

- Variation in structural properties; for example: variation in yield stress.
- Variation in the environment; for example: variation in a load.
- 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:

- Distribution of the response values.
- Probability of failure.
- Properties of the designs associated with failure.
    - Variable screening - identify important noise factors.
    - Dispersion factors - factors whose settings may increase variability of the responses.
- Efficient redesign strategies.

6.2 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 selected control variables for a more robust manufacturing process.

6.2.1 Variable linking

A single design parameter can apply to several statistically independent components in a system; for example: one joint design may be applicable to several joints in the structure.

The components will then all follow the same distribution but the actual value of each component will differ. Each duplicate component is in effect an additional variable and will result in additional computational cost (contribute to the curse of dimensionality) for techniques requiring an experimental design to build an approximation or requiring the derivative information such as FORM. Direct Monte Carlo simulation on the other hand does not suffer from the curse of dimensionality but is expensive when evaluating events with a small probability.

Design variables can be linked to have the same expected (nominal) value, but allowed to vary independently according to the statistical distribution during a probabilistic analysis. One can therefore have one design variable associated with many probabilistic variables.

Three probabilistic associations between variables are possible:

- Their nominal values and distributions are the same.
- Their nominal values differ but they refer to the same distribution.
- Their nominal values are the same but their distributions differ.

6.3 Basic computations

6.3.1 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

$$c.o.v = \frac{s}{\bar{y}}$$

6.3.2 Correlation of responses

Whether a variation in displacements in one location cause 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 coefficient of correlation is usually considered significant [1] if the absolute value is larger than 0.3.

6.3.3 Confidence intervals

The confidence interval on the mean assuming a normal distribution and using s^2 as an estimate to the variance is

$$\bar{y} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} < \mu < \bar{y} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}$$

with μ the mean, $\bar{y}$ the estimate to the mean, and $t_{\alpha/2,n-1}$ the relevant critical value of the t distribution.

The confidence interval on the variance assuming a normal distribution and using s^2 as an estimate to the variance is

$$\frac{(n-1)s^2}{\mathrm{X}^2_{\alpha/2,n-1}} < \sigma^2 < \frac{(n-1)s^2}{\mathrm{X}^2_{1-\alpha/2,n-1}}$$

with σ^2 the variance and $\mathrm{X}^2_{\alpha/2,n-1}, \mathrm{X}^2_{1-\alpha/2,n-1}$ the relevant critical values of the X^2 distribution.

The confidence interval on the probability of an event is

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p < \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

with p the probability, $\hat{p}$ the estimate to the probability, and $z_{\alpha/2,n-1}$ the relevant critical value of the normal distribution.

The coefficient of correlation has a confidence interval of

$$\tanh\left[\frac{1}{2}\ln\left[\frac{1+\rho}{1-\rho}\right]-\frac{t_{1-\alpha/2,N}}{\sqrt{N-3}}\right]\le\rho\le\tanh\left[\frac{1}{2}\ln\left[\frac{1+\rho}{1-\rho}\right]+\frac{t_{1-\alpha/2,N}}{\sqrt{N-3}}\right].$$

6.4 Probabilistic methods

The reliability – the probability of not exceeding a constraint value – can be computed using probabilistic methods.

The current version of LS-OPT provides only Monte Carlo evaluation of using approximations. Methods considering the Most Probable Point (MPP) of failure will be included in future versions of LS-OPT.

The accuracy can be limited by the accuracy of the data used in the computations as well as the accuracy of the simulation. The choice of methods depends on the desired accuracy and use of the reliability information.

More details on probabilistic methods can be found in, for example, the recent text by Haldar and Mahadevan [1].

6.4.1 Monte Carlo analysis

In a Monte Carlo analysis, we approximate the nominal value of a response using the mean of a number of computer experiments. The values of the random variables are selected considering their probability density function. Under the law of large numbers the solution will eventually converge.

Applications of a Monte Carlo investigation are:

- Compute the distribution of the responses, in particular the mean and standard deviation.
- Compute reliability.
- 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

$$\sigma_{\hat{\theta}}=\frac{\sigma}{\sqrt{N}}$$

with σ the standard deviation of $f(\mathbf{x})$ and N the number of sampling points. The error is therefore unrelated to the number of design variables.

The error of estimating p, the probability of an event, is a random value with the following variance

$$\sigma_{\hat{\theta}}^2 = \frac{p(1-p)}{N}$$

which can be manipulated to provide a minimum sampling. A suggestion for the minimum sampling size provided by Tu and Choi [2] is:

$$N = \frac{10}{P[G(x) \le 0]}$$

The above indicates that for a 10% estimated probability of failure; about 100 structural evaluations are required with some confidence on the first digit of failure prediction. To verify an event having a 1% probability; about a 1000 structural analyses are required, which usually would be too expensive.

A general procedure of obtaining the minimum number of sampling points for a given accuracy is illustrated using an example at the end of this section. For more information, a statistics text (for example, reference [3]) should be consulted. A collection of statistical tables and formulae such as the CRC reference [4] will also be useful.

The variance of the probability estimation must be taken into consideration when comparing two different designs. The error of estimating the difference of the mean values is a random variable with a variance of

$$\sigma_{\hat{\theta}}^2 = \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}$$

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

$$\sigma_{\hat{\theta}}^2 = \frac{p_1(1-p_1)}{N_1} + \frac{p_2(1-p_2)}{N_2}$$

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, the 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:

$$N = \frac{pq}{\sigma^2} = \frac{(0.9)(0.1)}{(0.051)^2} = 3457$$

6.4.2 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).

6.4.3 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:

$$P_f = \Phi(-\beta) \text{ or } \quad 1 - \Phi(\beta)$$

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:

- 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.
- 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 an 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.

6.4.4 Design for six-sigma methods

See the section for FOSM keeping in mind that the reliability index β is the number of standard deviations.

6.4.5 The most probable point

Probabilistic methods based on the most probable point of failure focus on finding the design perturbation most likely to cause failure.

To understand how these methods works, consider the limit state function *G(**x**)* dividing the responses into two design regions: one consisting of acceptable responses and the other of unacceptable responses. The two regions are separated by the hyperplane described by *G(**x**)=0*.

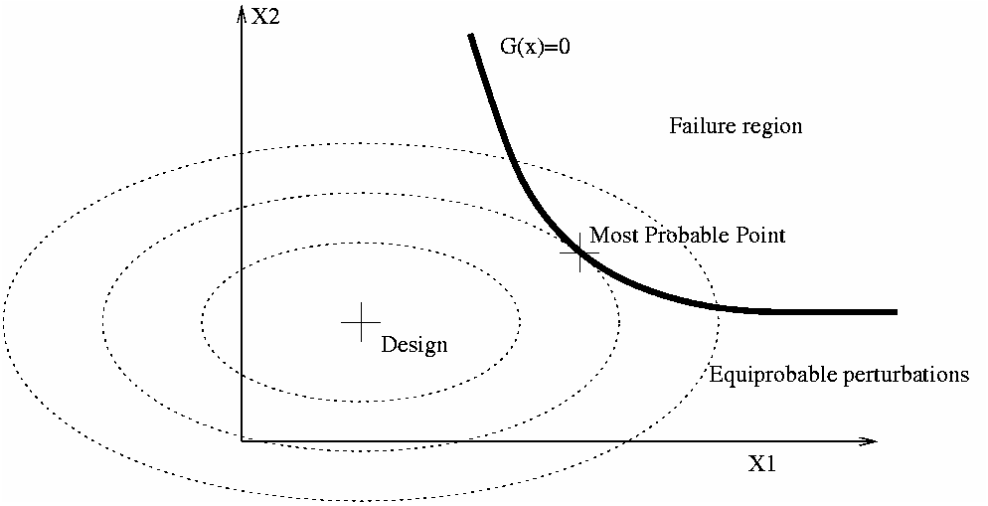


Figure 6-1 **Finding the most probable point of failure. The most probable point is the point on the line G(x)=0 closest to the design in the probabilistic sense.**

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 6-1, because all variables will not have an equal influence of the probability of failure due to differences in their distributions. In order to efficiently find this design perturbation, we transform the variables to a space of independent and standardized normal variables, the ***u***-space.

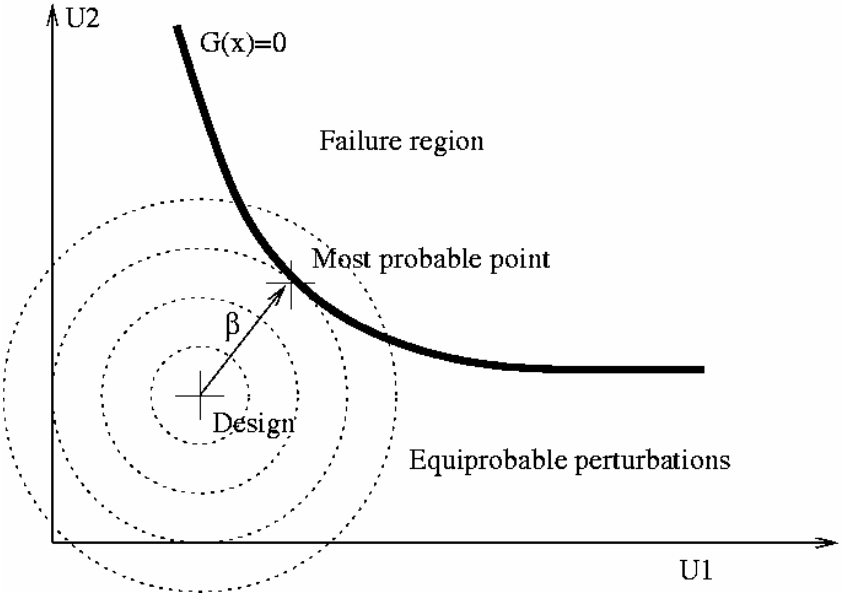


Figure 6-2 **Most probable point in the transformed space. In the transformed space the most probable point is the point on the line G(X)=0 the closest to the design.**

The transformed space is shown in Figure 6-2. 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:

Minimize: $\sqrt{\sum_{i=1}^{n} u_i^2}$

Subject to: $G(u) = 0$

This point is referred to as the *most probable point* (MPP) and the distance from the origin in the ***u***-space is referred to as the first-order probability index β_{FORM}.

The advantages of the most probable point are:

- The MPP gives an indication of the design most likely to fail.
- Highly accurate reliability methods utilizing an approximation around the MPP are possible.

6.4.6 FORM (First Order Reliability Method)

The Hasofer-Lind transformation is used to normalize the variables:

$$u_i = \frac{x_i - \mu_i}{\sigma_i}$$

The minimization problem is accordingly solved in the ***u***-space to find the first-order probability index β_{FORM}. Approximations to the responses are used to solve the optimization problem.

The probability of failure is found assuming a normally distributed response as

$$P_f = \Phi(-\beta_{FORM})$$

with Φ the cumulative density function of the normal distribution.

The error component of the procedure is due to (*i*) curvature of the constraint, (*ii*) the error component of the approximating function used in the computations, and (*iii*) the assumption of a normal distribution for the computation of failure.

The method is considered conservative considering that disregarding the curvature of the constraint results in an overestimation of the probability of failure.

6.4.7 Design sensitivity of the most probable point

For a probabilistic variable we use the partial derivative as:

$$\frac{\partial P}{\partial x_i} = \frac{\partial P}{\partial \beta}\frac{\partial \beta}{\partial u_i}\frac{\partial u_i}{\partial x_i}$$

with $\partial P / \partial \beta$ the derivative of the CDF function of the normal distribution.

For deterministic variables, which do not have a probabilistic component and therefore no associated ***u*** variables:

$$\frac{\partial P}{\partial x_i} = \frac{\partial P}{\partial \beta}\frac{\partial \beta}{\partial f}\frac{\partial f}{\partial x_i}$$

with $\partial\beta/\partial f$ taken as $\beta/(f_{\text{constraint}} - f_{\text{nominal}})$.

For the pathological case of being at the MPP, the vector associated with β vanishes and we use:

$$\frac{\partial P}{\partial x_i} = 0.4\frac{\partial G}{\partial u_i}\frac{\partial u}{\partial x_i}$$

with 0.4 the relevant value derivative of the CDF function of the normal distribution.

6.5 Required number of simulations

6.5.1 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.

6.5.2 Background

The required number of the simulation depends on:

- Cost of creating an accurate metamodel
- Cost of estimating the noise variation
- Cost of observing low-probability events

If the variation in the responses is mainly due to the variation of the design variables, then the cost of creating an accurate metamodel dominates. The region of interest for a robustness analysis will not be as large as to contain significant curvature; therefore a linear or preferably a quadratic response surface should

suffice. In past design optimization work, the number of experiments was successfully taken to be 1.5 times the number of terms (unknowns) in the metamodel to be estimated. For a robustness analysis, being conservative at this point in time, a value of twice the number of terms is recommended. The number of terms for a linear model is $k+1$ with k the number of design parameters. The number of terms for a quadratic response surface is $(k+1)(k+2)/2$.

The variation in the responses may not be mainly due to the variation of the design variables. In this case, enough experiments much be conducted to estimate this noise variation with sufficient accuracy. This cost is additional to the cost of creating the metamodel. The number of experiments required will differ considering the criteria for accuracy used. For example, we can require the error of estimating the noise variation to be less than 10%; however, this requires about 150 experiments, which may be too expensive. Considering the practical constraints for vehicle crash analysis, useful results can be obtained with 25 or more degrees of freedom of estimating the noise variation. This gives a situation where the error bound on the standard deviation is about 20% indicating that it is becoming possible to distinguish the six sigma events from five sigma events.

For design purposes, the variation of the responses and the role of the design variables are of interest. High accuracy may be impossible due to lack of information or unreasonable computational costs. A metamodel-based investigation and 20 FE evaluations can achieve:

- Investigate up to 10 variable
- Quantify the contribution of each variable
- 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.

6.5.3 Competing role of variance and bias

In an investigation the important design variables are varied while other sources are kept at a constant value in order to minimize their influence. In practice the other sources will have an influence. Distinguishing whether a difference in a response value is due to a deterministic effect or other variation is difficult, because both always have a joint effect in the computer experiments being considered.

In general [4] the relationship between the responses **y** and the variables **x** is:

$$\mathbf{y} = f(\mathbf{x}) + \delta(\mathbf{x}) + \boldsymbol{\varepsilon}$$

with f(**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 $\boldsymbol{\varepsilon}$ 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:

- 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.
- Reducing the region of interest to such a size that the difference between metamodel and true functional response is not significant.
- 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.

6.5.4 Confidence interval on the mean

For multiple regression, the 100(1-α)% confidence limits on the mean value at $\mathbf{X}_0$ are obtained from

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{\mathbf{X}_0(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}_0}$$

with s^2_{n-p} an estimate to σ^2. At the center of the region of interest for the coded variables the confidence interval is

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{C_{11}}$$

with C_{11} the first diagonal element of $(\mathbf{X}'\mathbf{X})^{-1}$. The confidence bound therefore depends on the variance of the response and the quality of the experimental design.

More details can be found in, for example, the text by Myers and Montgomery [6].

6.5.5 Confidence interval on a new evaluation

For multiple regression, the 100(1-α)% confidence limits on a new evaluation at $\mathbf{X}_0$ are obtained from

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{1 + \mathbf{X}_0(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}_0}$$

The confidence interval for new observations of the mean is

$$Y_0 \pm t_{\alpha/2,n-p} s_{n-p} \sqrt{1+C_{11}}$$

In the following table we monitor the bounds for a new evaluation of the mean for a linear approximation using five design variables using a 95% confidence interval. The value of C_{11} is computed from D-optimal experimental designs generated using LS-OPT. The error bounds are close to 2σ for more than 25 existing runs (20 degrees of freedom).

n	p	n-p	C_{11}	Bounds (σ=10% α=5%)
10	6	4	0.104	±29%
15	6	9	0.070	±23%
20	6	14	0.051	±22%
25	6	19	0.041	±21%
30	6	24	0.034	±21%
50	6	44	0.020	±20%
100	6	94	0.010	±20%

6.5.6 Confidence interval on the random deviation (σ^2)

The error of estimating the random deviation is minimized by:

- Large number of points
- Minimizing the bias error. Ideally one wants to observe many occurrences of the same design.

The residual mean square

$$s^2 = \frac{\sum_{i=1}^{n}(e_i - \bar{e})^2}{(n-p)} = \frac{\sum_{i=1}^{n} e_i^2}{(n-p)}$$

estimates σ^2 with $n-p$ degrees of freedom where n is the number of observations and p is the number of parameters including the mean.

We want to find an interval $[b_1, b_2]$ such that $P[b_1 \le s^2 \le b_2] = 0.95$. We rewrite as $P\left[\frac{n-p}{\sigma^2} b_1 \le \frac{n-p}{\sigma^2} s^2 \le \frac{n-p}{\sigma^2} b_2\right] = 0.95$. We have $(n-p)s^2/\sigma^2$ is a chi-squared distribution with $n-p$ degrees of freedom. From the chi-squared table we can get $[a_1, a_2]$ such that $P\left[a_1 \le \frac{n-p}{\sigma^2} s^2 \le a_2\right] = 0.95$ by reading of the values for 0.975 and 0.025. Having $[a_1, a_2]$ we can compute for $[b_1, b_2]$ as $\left(\frac{s^2}{n-p} a_1, \frac{s^2}{n-p} a_2\right)$. The $100(1-\alpha)\%$ confidence interval on σ^2 is therefore

$$\left(\frac{(n-p)s^2}{X^2_{\alpha/2,n-p}}, \frac{(n-p)s^2}{X^2_{1-\alpha/2,n-p}}\right)$$

In the table below we monitor the error bounds on the variance for a problem with six parameters (including the mean).

Noise Variance Confidence Interval								
n	**n-p**	**Lower Bound**			**Value (s)**	**Upper Bound**		
		α=5%	**α=10%**	**α=20%**		**α=20%**	**α=10%**	**α=5%**
10	4	5.99	6.49	7.17	10	19.39	23.72	28.74
15	9	6.88	7.29	7.83	10	14.69	16.45	18.25
20	14	7.32	7.69	8.15	10	13.41	14.60	15.77
25	19	7.605	7.94	8.36	10	12.77	13.70	14.6
30	24	7.81	8.12	8.50	10	12.38	13.16	13.91
50	46	8.31	8.56	8.86	10	11.59	12.10	12.56
106	100	8.78	8.97	9.19	10	11.02	11.33	11.61
206	200	9.11	9.24	9.41	10	10.69	10.92	11.09

In the above it was assumed that the metamodel is a sufficiently accurate approximation to the mechanistic model (the bias error sufficiently small) and that the errors are normally distributed. In general the estimate of σ^2 will be depend on the approximation model. For a model-independent estimate, replicate runs (multiple observations for the same design) are required. If the bias error is significant then the estimate of σ^2 will usually be too large [7].

6.5.7 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.

Probability that the event will be observed at least once (one or more times):
P[observing 0 events] = $(1\text{-P[event]})^n$
P[observing 1 or more events] = $1.0 - (1\text{-P[event]})^n$

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

6.6 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.

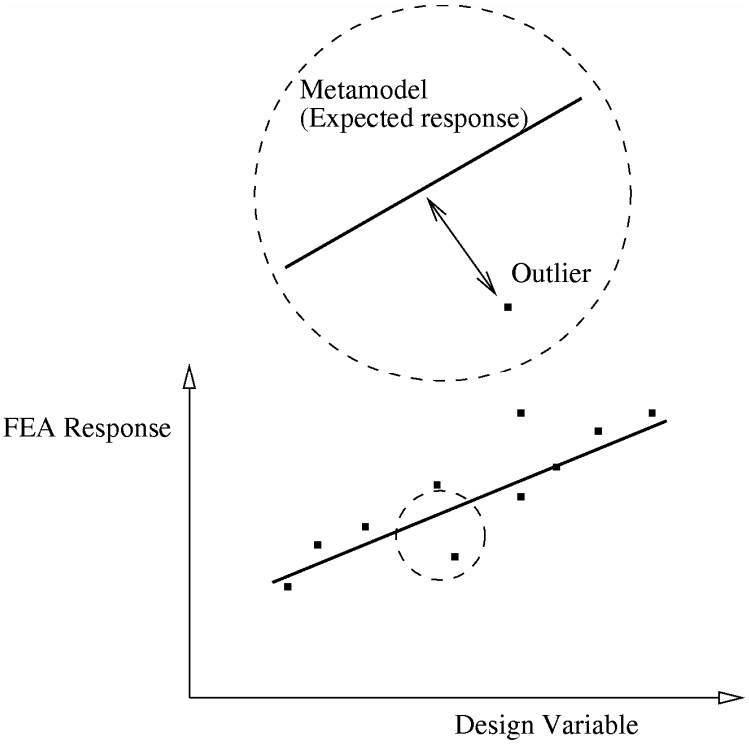


Figure 6-3 Outliers are identified after a metamodel has been fitted. Value in poor agreement of what is predicted by the design variables are considered outliers.

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.

6.7 Stochastic contribution analysis

The variation of the response can be broken down in contributions from each design variable.

6.7.1 Linear Estimation

The contribution can be estimated as:

$$\sigma_{g,i} = \partial G / \partial x \, \sigma_{x,i}$$

with $\sigma_{x,i}$ the standard deviation of the variable *i* and $\sigma_{g,i}$ the standard deviation of the variation of function *g* due to the variation of variable *i*.

The variance for all the variables is found as the sum of the variance:

$$\sigma_T^2 = \sum \sigma_i^2$$

where σ_T^2 is the variation of the response due to the variation of all the variables and σ_i^2 is the variation of response due to the variation of variable *i*. In the above it is assumed that the response is a linear response of the design variables and independent variables.

6.7.2 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' indices approach [8]. Firstly the function is decomposed as:

$$f(x_1,\ldots,x_n) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n}\sum_{j=i+1}^{n} f_{ij}(x_i,x_j) + \ldots + f_{1,2,\ldots,n}(x_1,\ldots,x_n)$$

From which partial variances are computed as:

$$V_{i,\ldots j} = \int_0^1 \ldots \int_0^1 f_{i\ldots j}^2(x_1,\ldots,x_n)dx_i \ldots dx_j$$

with the variance of the response summed from the partial variances as:

$$V = \sum V_i + \sum_{i<j} V_{ij} + \cdots + V_{1,2,\cdots,n}$$

The sensitivity indices are given as:

$S_i = V_i / V, \quad 1 \le i \le n$

$S_{ij} = V_{ij} / V, \quad 1 \le i < j \le n$

$S_{i,i+1,\ldots,n} = V_{i,i+1,\ldots,n} / V$

with the useful property that all of the sensitivity indices sum to 1:

$$\sum S_i + \sum_{i<j} S_{ij} + \cdots + S_{1,2,\cdots,n} = 1$$

Using Monte Carlo, the main effect can be computed as

$$\hat{D}_i = \frac{1}{N}\sum_{m=1}^{N} f(\mathbf{x}_{\sim im}^{(1)}, x_{im}^{(1)}) f(\mathbf{x}_{\sim im}^{(2)}, x_{im}^{(1)}) - \hat{f}_0^2$$

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

$$\hat{D}_{\sim i} = \frac{1}{N}\sum_{m=1}^{N} f(\mathbf{x}_{\sim im}^{(1)}, x_{im}^{(1)}) f(\mathbf{x}_{\sim im}^{(1)}, x_{im}^{(2)}) - \hat{f}_0^2$$

For second order response surfaces this can be computed analytically [9] as

$$\sigma_U^2 = \sum_{i\in U}^{n}\left[\beta_{ii}^2(m_{i,4} - \sigma_i^4) + (\beta_i + \beta_{ii}\mu_i + \sum_{j=1}^{n}\beta_{ij}\mu_j)^2\sigma_i^2 + (\beta_i + \beta_{ii}\mu_i + \sum_{j=1}^{n}\beta_{ij}\mu_j)\beta_{ii}m_{i,3}\right] + \sum_{i\in U}^{n}\sum_{i\in U,\, j\ge i}^{n}\beta_{ij}^2\sigma_i^2\sigma_j^2$$

with $m_{i,j}$ the jth 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, and composite functions, two options are currently available:

1. Approximate using second order response surface. The response surface is built using three times the number of terms in the response surface using a central points Latin hypercube experimental design over a range of plus/minus two standard deviations around the mean.
2. Using a Monte Carlo analysis. Many points are required.

Later version of LS-OPT may contain analytical computations for neural networks and Kriging models.

6.8 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 [10]. An alternate view of Taguchi [11] 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 [10].

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.

6.8.1 Fundamentals

The robustness of a structure depends on the gradient of the response function as shown in Figure 6-4. 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.

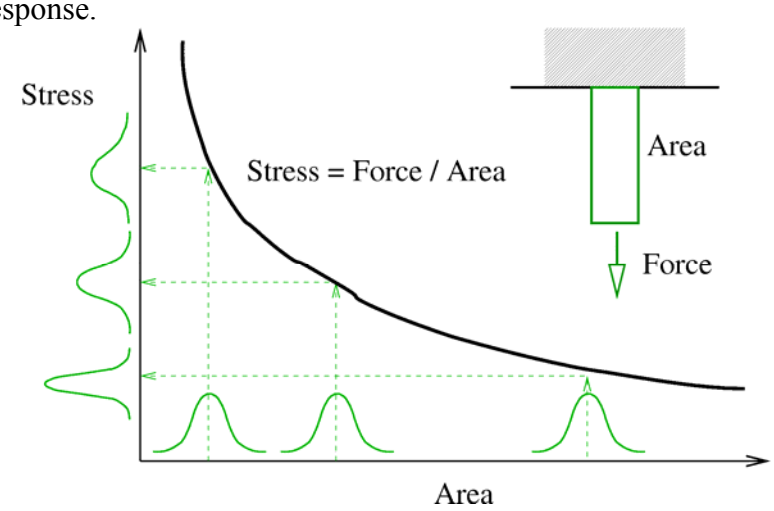


Figure 6-4 Robustness considering a single variable. Larger mean values of the area result in a smaller dispersion of the stress values. Note that the dispersion of the stress depends on the gradient of the stress-area relationship.

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:

- *Control variables.* The variables (design parameters) under the control of the designer are called control variables,
- *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 6-5 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.

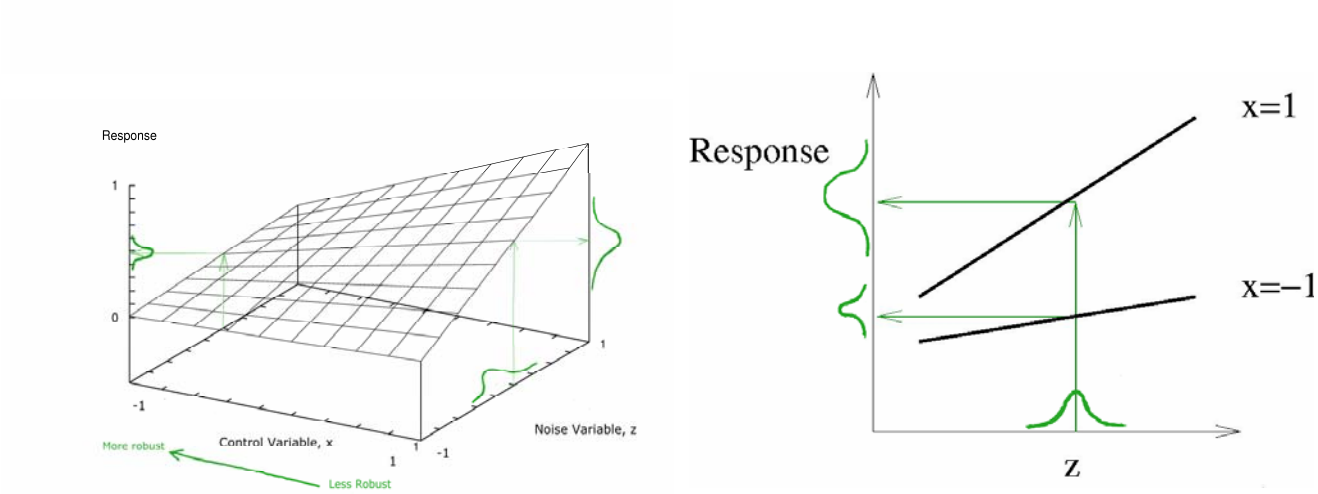


Figure 6-5 Robustness of a problem with both control and noise variables. The effect of the noise variable z on the response variation can be constrained using the control variable x. For robustness, the important property is the gradient of the response with respect to the noise variable. This gradient 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.

6.8.2 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 **x** and noise variables **z** with $Var(\mathbf{z}) = \sigma_z^2 \mathbf{I}_{r_z}$. 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 $Var(\mathbf{z}) = \sigma_z^2 \mathbf{I}_{r_z}$, σ^2 the model error variance, and l the vector of partial derivatives $l(x) = \dfrac{\partial y(x,z)}{\partial z}$.

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."

6.8.3 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_i^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.

6.9 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] 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.

[9] 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.

[10] 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.

[11] Roy RK. *Design of Experiments Using the Taguchi Approach*. Wiley, New York NY. 2001.

USER'S MANUAL

7. Design Optimization Process

7.1 A modus operandi for design using response surfaces

7.1.1 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. 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.

   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.

10. 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 3.3 and 4.7.

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 3.3) or whether to firstly perform variable screening (through ANOVA) 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 22.6 and 22.7).

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 4.7.

However 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 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:

7.1.2 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 results. This will highlight insignificant variables that may be removed from the problem. An ANOVA is simply a single iteration run, typically using a linear response surface to investigate main and/or interaction effects. The ANOVA results can be viewed in the post-processor (see Section 18.5).

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 3.1.3) 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:

   (a) There is significant noise in the design response.
   (b) 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 5.5. 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.

7.2 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 '0' as the number of iterations in the Run panel.

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 7.1.

7.3 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.

- **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 4.5) can be used.

- **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.

- **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 2.2.8).

- **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'.

- **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 probems, 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 22.5). In this case the response functions are insensitive to E and will show a very high confidence interval for E (Section 22.5.2).

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.

7.4 REFERENCES

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

8. Graphical User Interface and Command Language

This chapter introduces the graphical user interface, the command language and describes syntax rules for names of variables, strings and expressions.

8.1 LS-OPT user interface (LS-OPT*ui*)

LS-OPT can be operated in one of two modes. The first is through a graphical user interface, LS-OPT*ui*, and the second through the command line using the Design Command Language (DCL).

The user interface is launched with the command

`lsoptui` [*command_file*]

The layout of the menu structure (Figure 8-1) mimics the optimization setup process, starting from the problem description, through the selection of design variables and experimental design, the definition and responses, and finally the formulation of the optimization problem (objectives and constraints). The run information (number of processors, monitoring and termination criteria) is also controlled via LS-OPT*ui*.

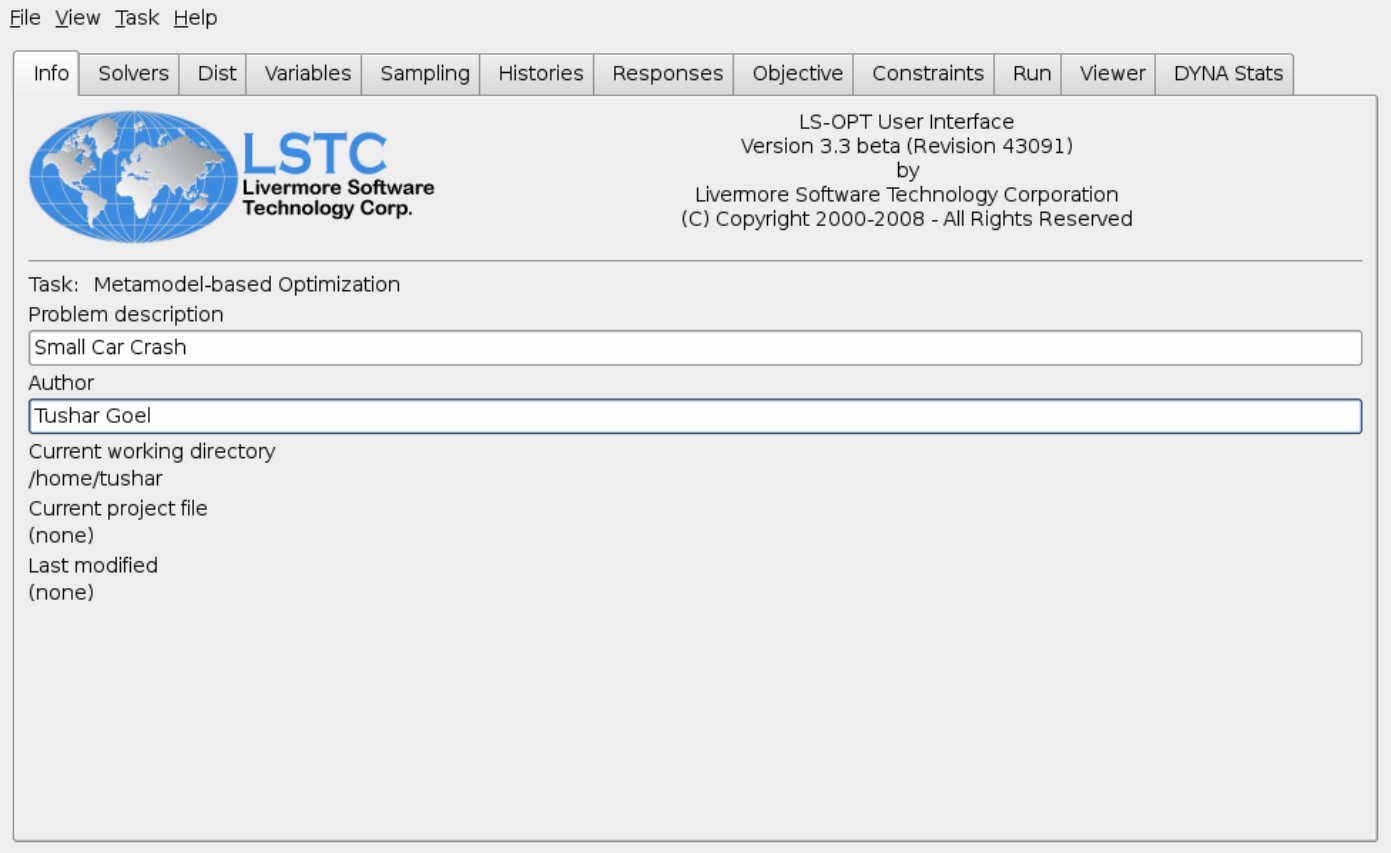


Figure 8-1: Information panel in LS-OPT*ui*

8.2 Problem description and author name

In LS-OPT*ui*, the Info (main) panel has fields for the entering of the problem description and author information.

Command file syntax:

```
problem_description
author author_name
```

A description of the problem can be given in double quotes. This description is echoed in the `lsopt_input` and `lsopt_output` files and in the plot file titles.

Example:

```
"Frontal Impact"
author "Jim Brown"
```

The number of variables and constraints are echoed from the graphical user input. These can be modified by the user in the command file.

Command file syntax:

```
solvers number_of_solvers <1>
constants number_of_constants <0>
variables number_of_variables
dependents number_of_dependent_variables <0>
histories number_of_response_histories <0>
responses number_of_responses
composites number_of_composites <0>
objectives number_of_objectives <0>
constraints number_of_constraints <0>
distributions number_of_distributions <0>
```

Example:

```
variable 2
constraint 1
responses 2
objectives 2
```

The most important data commands are the definitions. These serve to define the various entities which constitute the design problem namely solvers, variables, results, matrices, responses, objectives, constraints and composites. The definition commands are:

```
solver package_name
constant name value
variable name value
dependent name value
result name string
history name string
matrix name string
response name string
composite name type type
composite name string
objective name entity weight
constraint name entity name
```

Each definition identifies the entity with a name. “Results” and “matrices” do not require a count. Other entities will be phased out in future.

8.3 Command Language

The command input file is a sequence of text commands describing the design optimization process. It is also written automatically by LS-OPT*ui*.

The Design Command Language (DCL) is used as a medium for defining the input to the design process. This language is based on approximately 200 command phrases drawing on a vocabulary of about 200 words. Names can be used to describe the various design entities. The command input file combines a sequence of text commands describing the design optimization process. The command syntax is not case sensitive.

8.3.1 Names

Entities such as variables, responses, etc. are identified by their names. The following entities must be given unique names:

```
solver
constant
variable
dependent
result
history
matrix
response
composite
objective
constraint
```

A name is specified in single quotes, e.g.

```
solver dyna 'DYNA_side_impact'
constant 'Young_modulus' 50000.0
variable 'Delta' 1.5
dependent 'new_modulus' {Young_modulus + Delta}
result 'x_acc' "BinoutResponse -res_type rcforc -cmp z_force -id 1
                -side SLAVE -select TIME -end_time 0.002"
Matrix 'strain' {Matrix3x3Init(0.001,0.002,0.0035, a,b,c, d,e,f)}
History 'y_vel' "DynaASCII nodout Y_VEL 187705 TIMESTEP 0 SAE 30"
Response 'x_acc' "DynaASCII rbdout X_ACC 21 AVE"
composite 'deformation' type targeted
composite 'sqdef' {sqrt(deformation)}
objective 'deformation' composite 'deformation' 1.0
constraint 'Mass' response 'Mass'
```

In addition to numbers `0-9`, upper or lower case letters, a name can contain any of the following characters:

`_.`

The leading character must be alphabetical. *Spaces are not allowed*.

Note:

Because mathematical expressions can be constructed using various entities in the same formula, duplication of names is not allowed.

8.3.2 Command lines

Preprocessor commands, solver commands or response extraction commands are enclosed in double quotes, e.g.

```
$ SPECIFICATION OF PREPROCESSOR AND SOLVER
preprocessor command "/usr/ls-dyna/ingrid"
solver command "/alpha6_2/usr/ls-dyna/bin/ls-dyna_9402_dec_40"
$ IDENTIFICATION OF THE RESPONSE
response 'displacement' "DynaRelativeDisp 0.2"
response 'Force' "Myforce"
```

In addition to numbers `0-9`, upper or lower case letters and spaces, a command line can contain any of the following characters:

```
_=-.'/<>;`
```

In the command input file, a line starting with the character `$` is ignored.

A command must be specified on a single line.

8.3.3 File names

Input file names for the solver and preprocessor must be specified in double quotes.

```
prepro input file "p11i"
solver input file "side_impact"
```

8.3.4 Command file structure

The commands are arranged in two categories:

- problem data
- solution tasks

There are several commands for specifying the available tasks. The remaining commands are for the specification of problem data. A solution task command serves to execute a solver or processor while the other commands store the design data in memory.

In the following chapters, the command descriptions can be easily found by looking for the large typescript bounded by horizontal lines. Otherwise the reader may refer to the quick reference manual that also serves as an index. The default values are given in angular brackets, e.g. < 1 >.

8.3.5 Environments

Environments have been defined to represent all dependent entities that follow. The only environments in LS-OPT are for

- `solver` *`identifier_name`*
  All responses, response histories, solver variables, solver experiments and solver-related job information defined within this environment are associated with the particular solver.
- `strict, slack/soft` Pertains to the strictness of constraints. See Sections 16.5.
- `move, stay` Pertains to whether constraints should be used to define a reasonable design space or not for the experimental design. See Section 13.6.

8.3.6 Expressions

Each entity can be defined as a standard formula, a mathematical expression or can be computed with a user-supplied program that reads the values of known entities. The bullets below indicate which options apply to the various entities. Variables are initialized as specified numbers.

Table 8.3-1: Expression options of optimization entities

Entity	Standard	Expression	User-defined
Variable			
Dependent		●	
Result	●	●	●
Matrix		●	
History	●	●	●
Response	●	●	●
Composite	●	●	

A list of mathematical and special function expressions that may be used is given in Appendix D : Mathematical Expressions.

9. Program Execution

This chapter describes the directory structure, output and status files, and logistical handling of a simulation-based optimization run.

9.1 Work directory

Create a work directory to keep the main command file, input files and other command files as well as the LS-OPT program output.

9.2 Execution commands

`lsoptui` *`command_file_name`*	Execute the graphical user interface
`lsopt` *`command_file_name`*	LS-OPT batch execution
`lsopt info`	Create a log file for licensing
`lsopt env`	Check the LSOPT environment setting
`viewer` *`command_file_name`*	Execute the graphical postprocessor

The LSOPT environment is automatically set to the location of the `lsopt` executable.

9.3 Directory structure

When conducting an analysis in which response evaluations are done for each of the design points, a sub-directory will automatically be created for each analysis.

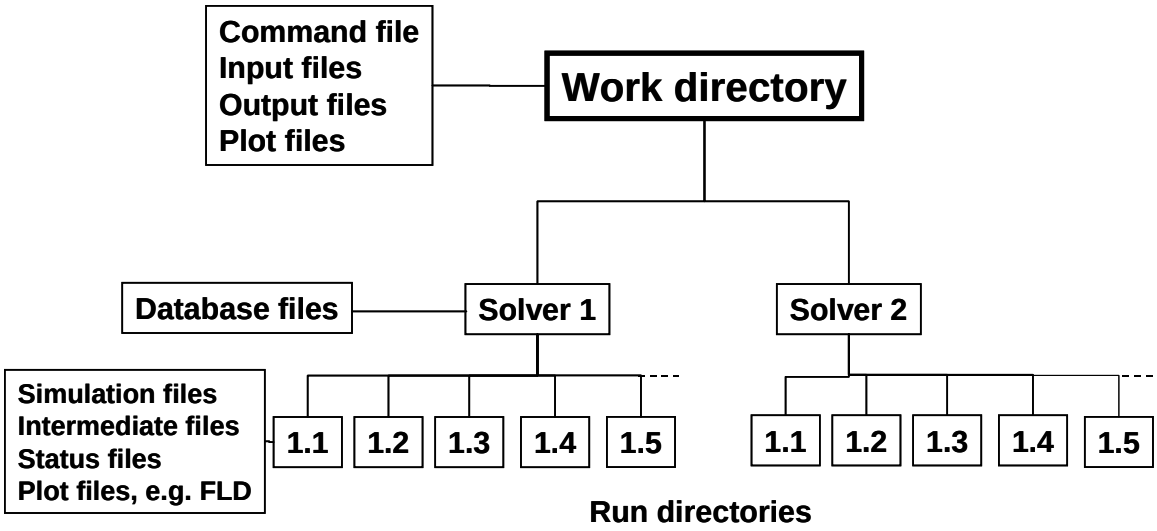


Figure 9-1 : Directory structure in LS-OPT

These sub-directories are named *solver_ name/*mmm.nnnn, where mmm represents the iteration number and nnnn is a number starting from `1`. *solver_ name* represents the solver interface specified with the command, e.g.

```
solver dyna 'side_impact'
```

In this case `dyna` is a reserved package name and `side_impact` is the name of an analysis case chosen by the user. The work directory needs to contain at least the command file and the template input files. Various other files may be required such as a command file for a preprocessor. An example of a sub-directory name, defined by LS-OPT, is `side_impact/3.11`, where `3.11` represents the design point number of iteration 3. The creation of subdirectories is automated 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.

9.4 Job Monitoring

The job status is automatically reported at a regular interval. The user can also specify the interval. The interface, LS-OPTui reports the progress of the jobs in the Run panel (see Section 17.6). 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.

9.5 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.

9.6 Restarting

Restarting is conducted by giving the command:

`lsopt` *`command_file_name`*, or by selecting the Run button in the Run panel of LS-OPT*ui*.

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 is not available elsewhere.

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. A single iteration has been carried out but the design formulation is incorrect and must be changed.
b. Incorrect data was extracted, e.g., for the wrong node or in the wrong direction.
c. The user wants to change the response surface type, but keep the original experimental design.

In the above cases, all the `history.`*n* and `response.`*n* files must be deleted. After restarting, the data will then be newly extracted and the subsequent phases will be executed. 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. Unless the database was deleted (by, e.g., using the `clean` file, see Section 9.9), no simulations will be unnecessarily repeated, and the simulation run should continue normally.

3. A restart can be made from any particular iteration by selecting the 'Specify Starting Iteration' button on the Run panel, and entering the iteration number. The subdirectories representing this iteration and all higher-numbered iterations will be deleted after selecting the Run button and confirming the selection.

4. The number of points can be changed for a restart (see Section 13.11).

9.7 Output files

The following files are intermediate database files containing ASCII data.

Table 9.7-1: Intermediate ASCII database files

Database file	Description	Directory
`Experiments`	Trial designs computed as a result of the experimental design	Case
`AnalysisResults`	The same trial designs and the responses extracted from the solver database	Case
`DesignFunctions`	Parameters of the approximate functions	Case
`OptimizationHistory`	Variable, response and error history of the successive approximation process	Work
`TradeOff`	All variable, responses and extended results of the non-dominated solutions at each iteration	Work
`ExtendedResults`	All variables, responses and extended results at each trial design point	Case
`Net.`*`funcname`*	Parameters of the metamodel of function with name *funcname*	Case

A more detailed description of the database is available in Appendix C.

The output files are as follows:

Table 9.7-2: Output files

Database file	Description	Directory	View option
`lsopt_input`	Input in a formatted style	Work	Input
`lsopt_output`	Results and some logging information	Work	Output
`lsopt_report`	A final report of the analysis results. Available for some of the main tasks and most of the Repair tasks	Work	Summary
`history_design`	Table of the objective and constraint values for each iteration (e.g. for plotting)	Work	File
`history_variables`	Table of the design variables, responses and composites for each iteration (e.g. for plotting)	Work	File
`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 LSOPT.	Work	File

The following files are in a .csv (comma separated variables) format:

Table 9.7-3: Result files in .csv format

Database file	Description	Directory	View option
`AnalysisResults_`*`n`*`.csv`	Analysis Results (*n* = iteration number)	Case	
`PRESS_predictions_`*`n`*`.csv`	PRESS (Section 2.3.4) predicted results and PRESS residuals (Polynomials and Radial Basis Function networks (Section 3.1.2) only. PRESS residuals are not computed for Feedforward Neural Networks)	Case	Use check box to select PRESS in Viewer→ Accuracy→

9.8 Log files and status files

Status files `prepro`, `replace`, `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 9.8-1: Status files generated by LS-OPT

`prepro`	The preprocessing has been done.
`replace`	The variables have been replaced in the input files.
`started`	The run has been started.
`finished`	The run has been completed. The completion status is given in the file.
`response.`*n*	Response number *n* has been extracted.
`history.`*n*	History number *n* has been extracted.
`EXIT_STATUS`	Error message after termination.

The user interface LS-OPT*ui* uses the message in the `EXIT_STATUS` file as a pop-up message.

The `lfop.log` file contains a log of the core optimization solver solution.

The simulation run/extraction log is saved in a file called `log`*nnnnnn* in the local run directory, where *nnnnnn* represents the process ID number of the run. An example of a logfile name is `log234771`.

Please refer to Section 9.6 for restarting an optimization run.

9.9 Managing disk space during run time

During a successive approximation procedure, superfluous data can be erased after each run while keeping all the necessary data and status files (see above and example below). 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 `prepro`, `started`, `replace`, `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*. E.g., the file `response.2` contains the extracted value of response 2. 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:

```
prepro
XPoint
replace
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.

9.10 Error termination of a solver run

The job scheduler will mark an error-terminated job to avoid termination of LS-OPT. Results of abnormally terminated jobs are ignored. If there are not enough results to construct the approximate design surfaces, LS-OPT will terminate with an appropriate error message.

9.11 Parallel processing

Runs can be executed simultaneously. The user has to specify how many processors are available.
Command file syntax:

```
concurrent jobs number_of_jobs
```

If a parallel solver is used, the number of concurrent jobs used for the solution will be *number_of_jobs* times the number of cpu's specified for the solver.

Example:

```
concurrent jobs 16
```

If the number of concurrent jobs is specified as 0, all the jobs will be run simultaneously. This can be used to transfer all the jobs to a queuing system (see Section 9.12) at once.

9.12 Using an external queuing or job scheduling system

9.12.1 Introduction

The LS-OPT Queuing Interface interfaces with load sharing facilities (e.g. LSF[1] or LoadLeveler[2]) 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*. The `README.queue` file should be consulted for the most up to date information about the queuing interface.

Command file syntax:

`Solver queue [`*`queue_name`*`]`

Table 9.12-1: Queuing options

queuer_ name	Description
`lsf`	LSF
`loadleveler`	LoadLeveler
`pbs`	PBS[3]
`nqe`	NQE[4]
`aqs`	AQS
`slurm`	SLURM
`user`	User Defined
`blackbox`	Black box
`msccp`	MS Windows Compute Cluster Server

9.12.2 Installation

To run LS-OPT with a queuing (load-sharing) facility the following binary files are provided in the `/bin` directory which un-tars (or unzips) from the distribution during installation of LS-OPT:

```
bin/wrappers/wrapper_*
bin/runqueuer
```

[1] Registered Trademark of Platform Computing Inc.
[2] Registered Trademark of International Business Machines Corporation
[3] Portable Batch System. Registered Trademark of Veridian Systems
[4] Network Queuing Environment. Registered Trademark of Cray Inc.

The * represents platform details, e.g. wrapper_hp or wrapper_suse91. The runqueuer executes the command line for the purpose of queuing and must remain in the LS-OPT environment (the same directory as the lsopt executable).

The following instructions should then be followed:

Installation for all remote machines running LS-DYNA

1. Create a directory on the remote machine for keeping all the executables including lsdyna. Copy the appropriate executable wrapper_* program located in the bin/wrappers directory to the new directory. E.g. if you are running lsdyna on HPUX, place wrapper_hp on this machine. Rename it to "wrapper".

Installation on the local machine

2. Select the queuer option in LS-OPTui or add a statement in the LS-OPT command file to identify the queuing system, e.g.

   ```
   queuer lsf
   ```
   or
   ```
   solver queuer loadleveler
   ```

   for each solver.

   To pass all the jobs to the queuing system at once, select zero concurrent jobs in the GUI or command file, e.g.

   ```
   concurrent jobs 0
   ```

   Example:

   ```
   solver command "rundyna.hp DynaOpt.inp single 980"
    solver input file "car6_crash.k"
    solver queuer loadleveler
   ```

   In this example, the arguments to the rundyna.hp script are optional and can be hard-coded in the script.

3. 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 .

9.12.3 Example

An example using a script follows:

The LS-OPT command file part relating to the queue is:

```
solver dyna960 'Case1'
$ ---- PBS Script
  solver command "/nec00a/mike/project/submit_pbs"
$ ---- Input file with variable substitution
  solver input file "input.k"
$ ---- Queuing specification
  solver queue pbs
```

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
#!/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
cd $newdir
/nec00a/mike/codes/wrapper /nec00a/mike/codes/ls980.single i=DynaOpt.inp
EOF
qsub dynscr
```

9.12.4 Mechanics of the queuing process

Understanding the mechanics of the queuing process should help to debug the installation.
:

1. LS-OPT automatically prepends `runqueuer` to the solver command and executes `runqueuer` which runs the `submit_pbs` script.
    a. The `runqueuer` sets the variables `LSOPT_HOST` and `LSOPT_PORT` locally.
    b. The `submit_pbs` script spawns the `dynscr` script.
2. 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 most cases the queuing system will transmit the environment variables to the remote side, so the setting of the variables may not be necessary.

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 written to the logxxxx file (where xxxx is the process number) on the local host (look in the local sub-subdirectory, e.g. CRASH/1.7). An example of an error message resulting from a mistype of "wrapper" in the submit script is given in the log file as follows:

```
STARTING command /home/jim/bin/runqueuer
PORT=56984
JOB=LoadLeveler
llsubmit: The job "1/1.1" has been submitted.
/home/jim/LSOPT_EXE/Xrapper: Command not found.
finished with directory
/home/jim/LSOPT/___3.1___/optQA/QUEUE/EX4a_remote/remote/1/1.1
```

4. The wrapper will also extract the data immediately upon completion on the remote node. Extracted data (the history.n and response.n files) are automatically transferred back to the local sub-subdirectory. If other parts of the database (e.g. d3plot files) are required (e.g. for post-processing with LS-PREPOST), the user has to specify these in the command file using appropriate LS-OPT commands (see Section 9.12.9). A log of the database extraction is provided in the logxxxx file.

9.12.5 Environment variables

Users typically do not need to set these. However these variables are set on the local side and their values must be carried to the remote side by the queuing software. If you do not know if this is being done, try setting them in the submit script as in the example above or please contact your system administrator.

LSOPT_HOST : the machine where LS-OPT (and therefore the runqueuer) is running. Set this if wrapper_* has trouble connecting back to runqueuer.

LSOPT_PORT : TCP/IP port runqueuer listens on for remote connections

9.12.6 Abnormal termination and retrying the job submission

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. The following two parameters can be used to set the number of retries allowed and timeout for each retry. The defaults are shown in square brackets

Command file syntax:

```
Solver job retry [number_of_retries_allowed[9]]
Solver job timeout [timeout for retry in seconds[60]]
```

Queuer timout
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.

Command file syntax:

```
Solver queue timeout [number_of_minutes[720]]
```

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.

9.12.7 Troubleshooting

1. Diagnostics for a failed run usually appear in the `logxxxx` 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.

   Please attach the log file when emailing 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 9.12.11 .

A system under development for future versions of LS-OPT will significantly simplify and standardize the scripting for running remote jobs.

9.12.8 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
   ```

9.12.9 Blackbox queueing system

The Blackbox queueing system is another flavor of the User-defined queueing system. It 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:

- 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.
- Extraction of responses and histories takes place on the local workstation instead of on the computer running the job.
- 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.

When using the Blackbox queueing 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 following strings are accepted:

String	Description

WAITING	The job has been submitted and is waiting to start
RUNNING	The job is running. 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 queueing system reports some kind of problem. Hence, a solver that has terminated in error does not have to be deteceted by the LsoptJobCheck script.
FINISHED	The job has completed and any output files needed for extraction has been copied back to the run directory.

Note that under Windows, the LsoptJobCheck and LsoptJobDel scripts have the suffix .bat.

9.12.10 Microsoft Windows Compute Cluster Server

LS-OPT supports submission of jobs to the Microsoft Compute Cluster Pack Scheduler. Two scripts called submit.cmd and submit.vbs, that work together, are available to interface LS-OPT with CCP. The script can be downloaded from ftp://ftp.lstc.com/ls-opt. Before using the scripts the variables in the beginning of the file submit.cmd needs to be changed to fit your local environment. Most users do not need to change the submit.vbs file.

The example shows how the queue-related parts of an LS-OPT command file look when using the CCP scripts, when they are placed in the same directory as the command file:

Example:

```
solver dyna960 '1'
      solver command "..\..\submit.cmd \\fileserver\bin\ls971.exe"
      solver input file "key.k"
      solver queue msccp
```

9.12.11 Database recovery

When distributing the simulation runs, the data can be recovered to the local machine. There are two commands: a LS-DYNA specific command and a general command.

LS-DYNA:

Command file syntax:

```
Solver recover dyna [d3plot|d3hsp|binout|d3eigv|eigout]
```

The LS-DYNA database can be recovered by using the above command. The requested database file will appear in the local directory. Each name is a prefix, so that e.g. `d3plot01, d3plot02, …` will be recovered when specifying `d3plot`. The details of the recovery procedure is logged in a local directory file.

Example:

```
Solver recover dyna d3plot
Solver recover dyna binout
```

The recovery of the LS-DYNA database is only required if the user wants to do local post-processing (e.g. using LS-PREPOST). Otherwise the results are automatically extracted and transferred to the local node in the form of files `response.`*n* and/or `history.`*n*.

User-defined :

Command file syntax:

`Solver recover file "[`*file_wildcard*`]"`

Any database can be recovered by using the above command. The requested database file will appear in the local directory. Each name is a wildcard.

Example:

```
Solver recover file "d3plot*"
Solver recover file "*"
```

The first command will recover the full `d3plot` database.

The last command will recover all the files from the run directory on the remote node to the run directory on the local node, hence the local directory will mirror the remote directory.

A log of the database recovery is available in the `logxxxx` file in the run directory on the local machine.

10. Interfacing to a Solver or Preprocessor

This chapter describes how to interface LS-OPT with a simulation package and/or a parametric preprocessor. Standard interfaces as well as interfaces for user-defined executables are discussed.

10.1 Labeling design variables in a solver and preprocessor

Parameters specified in input files are automatically identified for the following packages:

Package	Native parameters recognized in input file	LS-OPT Parameter Format recognized (see Section 10.1.1)	include files recognized in input file
LS-DYNA	Yes	Yes	Yes
ANSA	Yes	Yes	Yes
DEP Morpher[5]	Yes	Yes	No
HyperMorph[6]	Yes	Yes	No
TrueGrid[7]	No	Yes	Yes
LS-INGRID	No	Yes	Yes
User-defined	N/A	Yes	No

LS-OPTui will automatically recognize the native and LS-OPT parameters for the formats indicated in the table and display them as 'Constants' against a blue background in the 'Variables' 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 solver input file. A gray background indicates that the parameter name was specified in the GUI by the user or read from the LS-OPT command file and is not available in any of the input or include files.

The 'include' files are also scanned wherever this feature is available making it nonessential to append any 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

[5] Registered Trademark of Detroit Engineering Products
[6] Registered Trademark of Altair Engineering, Inc.
[7] Registered Trademark of XYZ Scientific Applications, Inc.

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 Section 9.12), do not contain any path specifications. These files are automatically transmitted to the relevant nodes where the solver will be executed.

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

10.1.1 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*)>>**.

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

```
-----------------------------------------------------------------------
 L I S T   O F   I N P U T   F I L E S   U S E D   B Y   L S - O P T
-----------------------------------------------------------------------

 SOLVER: 1
--------------------------------|-----------|---------|-----------------|
File name                          Type      Utility  Parameter Occur.
                                                       -----------------
                                                        Native   LS-OPT
--------------------------------|-----------|---------|--------|--------|
main.k                           LS-DYNA 960 Rootfile  2        0
../../car5.k                     LS-DYNA 960 Include   0        0
--------------------------------|-----------|---------|-----------------|

-------------------------------------------------------------------------
  L I S T   O F   I N C L U D E   F I L E S   A N D   P A R A M E T E R S
-------------------------------------------------------------------------
=========================================================================
File Name        Include Parameters Status     Time Stamp
```

```
                  Files
=========================================================================
main.k            1       2          OLD        Thu Apr  1 14:39:11 2004
=========================================================================

                List of Include Files for "main.k"

                -----------------
                Include File Name
                -----------------
                ../../car5.k
                -----------------

                List of Parameters found in "main.k"

                ----------------------------------------
                Parameter Name   Value      Type
                ----------------------------------------
                tbumper          1          *PARAMETER
                thood            3          *PARAMETER
                ----------------------------------------
```

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 'Variables' panel or command file.

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.

10.2 Interfacing to a Solver

In LS-OPT*ui*, solvers are specified in the Solver panel (Figure 10-1):

Both the preprocessor and solver input and append files are specified in this panel. Multiple solvers (as used in multi-case or multi-disciplinary applications) are defined by selecting 'Add'. The 'Replace' button must be used after the modification of current data.

The name of the analysis case is used as the name for the subdirectory.

Execution command. The command to execute the solver must be specified. The command depends on the solver type and could be a script, but typically excludes the solver input file name argument as this is specified using a separate command. The execution command may include any number of additional arguments.

Input template 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. During run-time, LS-OPT appends a standard input deck name to the end of the execution command. In the case of the standard solvers, the appropriate syntax is used (e.g. `i=DynaOpt.inp` for LS-DYNA). For a user-defined solver, the name `UserOpt.inp` is appended. The specification of an input file is not required for a user-defined solver.

Appended file. Additional solver data can be appended to the input deck using the *solver_append_file_name* file. This file can contain variables to be substituted.

Include files. These do not have to be specified as they are automatically and recursively searched by LS-OPT when given the name of the main input file (root file).

Extra input files. Extra files can be specified as input files. These files will be parsed for variables and copied to the run directories. Variables will be displayed in the Variables panel.

.

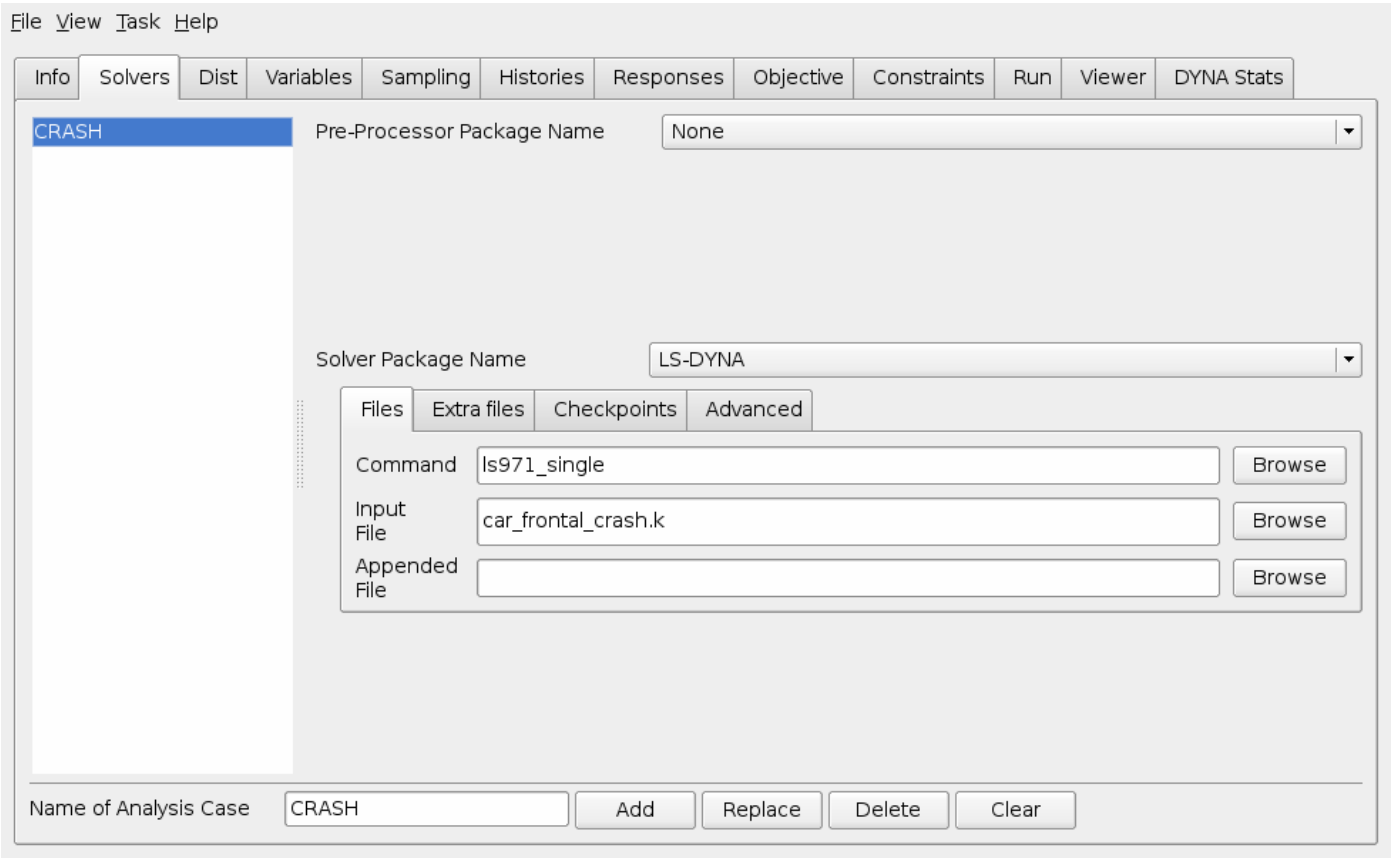


Figure 10-1: Solver panel in LS-OPT*ui*

Command file syntax:

```
solver software_package_identifier name_of_analysis_case
solver input file "solver_input_file_name"
solver command "solver_program_name"
solver append file "solver_append_file_name"
interval Time_interval_between_progress_reports < 15 > (not available in LS-OPTui)
```

The following software package identifiers are available:

`own`	user-defined solver
`dyna`	LS-DYNA Versions prior to 960
`dyna960`	LS-DYNA Version 960/970

10.2.1 Interfacing with LS-DYNA

The first command demarcates the beginning of the solver environment.

Example:

```
$ Define the solver software to be used.
solver dyna960 'SIDE_IMPACT'
$ the data deck to be read by the solver.
  solver input file "ingrido"
$ the command to execute the solver.
  solver command "/alpha6_2/usr/ls-dyna/bin/ls970.single"
$ Extra commands to the solver.
  solver append file "ShellSetList"
```

More than one analysis case may be run using the same solver. If a new solver is specified, the data items not specified will assume previous data as default. All commands assume the current solver.

Remarks:

- The name of the solver will be used as the name of the sub-directory to the working directory.
- The command `solver package_identifier` *name* initializes a new solver environment. All subsequent commands up to the next "`solver` *name*" command will apply to that particular solver. This is particularly important when specifying `response` *name commandline* commands as each response is assigned to a specific solver and is recovered from the directory bearing the name of the solver. (See Section 14).
- 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.
- The `solver command` name *must not be an alias*. The full path name (or the full path name of a script which contains the full solver path name) must be specified.

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

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 Variable 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.

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

Check of the *DATABASE cards

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

- Whether the required binout data types are requested in the LS-DYNA input deck. For example, if LS_OPT uses airbag data, then the LS-DYNA deck should contain a ***DATABASE_ABSTAT** card requesting binout output.
- 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 be flagged incorrectly as missing; switch off the checking in this case.

The GUI allows this to be set as an advanced solver option.

Command file syntax:

```
solver check output on/off
```

Altering the d3plot databases

The following options are available:

- Compress the d3plot database. All results except displacements, velocities, and accelerations will be deleted.
- 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 is required. An example of the possible contents of the file: `1001 1002 1003`. The file therefore consists of a single line.
- 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:
  - *–id* excludes part with *id*
  - *id* includes part with *id*
  - *id1-id2* includes parts from *id1* to *id2*.

  For example:
  ```
  5
  7-20
  -9.
  ```

The GUI allows this to be set as an advanced solver option.

This capability does not work with adaptivity.

The *DATABASE_EXTENT_BINARY option in LS-DYNA also allows control over the size of the d3plot databases.

Command file syntax:

```
solver compress d3plot on/off
solver compress d3plot nodes nodrel_filename
solver compress d3plot extract parts_filename
```

Example:

```
$ set d3plot compress options
solver compress d3plot on
solver compress d3plot nodes "nodrel_nodes.txt"
solver compress d3plot extract "part_list.txt"
```

10.2.2 Interfacing with LS-DYNA/MPP

The LS-DYNA MPP (Message Passing Parallel) version can be run using the LS-DYNA option in the "Solver" window of LS-OPTui (same as the `dyna` option for the solver in the command file). However, the run commands must be specified in a script, e.g. the UNIX script `runmpp`:

```
mpirun -np 2 lsdynampp i=DynaOpt.inp
cat dbout.* > dbout
dumpbdb dbout
```

The solver specification in the command file is as follows:

```
solver dyna960 'crash'
solver command "../../runmpp"
solver input file "car5.k"
solver append file "rigid2"
```

Remarks:

1. `DynaOpt.inp` is the *reserved* name for the LS-DYNA MPP input file name. This file is normally created in the run directory by LS-OPT after substitution of the variables or creation by a preprocessor. The original template file can have a different name and is specified as the input file in the `solver input file` command.

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

3. The file `dumpbdb` for creating the ASCII database must be executable.
4. The script must be specified in one of the following formats:

   (a) path relative to the run directory: two levels above the run directory (see example above).
   (b) absolute path, e.g. "`/origin/users/john/crash/runmpp`"
   (c) in a directory which is in the path. In this case the command is:

```
solver command "runmpp".
```

10.2.3 Interfacing with a user-defined solver

An own solver can be specified using the `solver own` *solvername* command, or selecting User-defined in LS-OPT*ui*. The `solver 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 `solver input file` (which will be renamed `UserOpt.inp)` and the `solver append file`. 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 'N o r m a l'.
```

Example:

```
solver own 'Analyzer'
  solver command "../../run_this_script"
  solver input file "setup.jou"
```

10.3 Preprocessors

The preprocessor must be identified as well as the command used for the execution. The command file executed by the preprocessor to generate the input deck must also be specified. The preprocessor specification is valid for the current solver environment.

Command file syntax:

```
prepro software_package_identifier
prepro command "prepro_program_name"
prepro input file "pre_file_name"
```

The interfacing of a preprocessor involves the specification of the design variables, input files and the preprocessor run command. Interfacing with LS-INGRID, TrueGrid[8], AutoDV and HyperMorph[9] and the ANSA Morpher[10] is detailed in this section. The identification of the design variables in the input file is detailed in Section 10.1.

10.3.1 LS-INGRID

The identifier in the `prepro` section for the use of LS-INGRID is `ingrid`. The file `ingridopt.inp` is created from the LS-INGRID input template file.

[8] Registered Trademark of XYZ Scientific Applications, Inc.
[9] Registered Trademark of Altair Engineering, Inc.
[10] Registered Trademark of Detroit Engineering Products

Example:

```
$ the preprocessor software to be used.
prepro ingrid
$ the command to execute the preprocessor
  prepro command "ingrid"
$ the input file to be used by the preprocessor
  prepro input file "p9i"
```

This will allow the execution of LS-INGRID using the command "`ingrid i=ingridopt.inp -d TTY`". The file `ingridopt.inp` is created by replacing the << *name* >> keywords in the `p9i` file with the relevant values of the design variables.

10.3.2 TrueGrid

The identifier in the `prepro` section for the use of TrueGrid is `truegrid`. This will allow the execution of TrueGrid using the command "*`prepro program_name`* `i=TruOpt.inp`". The file `TruOpt.inp` is created by replacing the << *name* >> keywords in the TrueGrid input template file with the relevant values of the design variables.

Example:

```
$ the preprocessor software to be used.
prepro truegrid
$ the command to execute the preprocessor
  prepro command "tgx"
$ the input file to be used by the preprocessor
  prepro input file "cyl"
```

These lines will execute TrueGrid using the command "`tgx i=cyl`" having replaced all the keyword names << *name* >> in `cyl` with the relevant values of the design variables.

The TrueGrid input file requires the line:

```
write end
```

at the very end.

10.3.3 ANSA

The ANSA preprocessor can be interfaced with LS-OPT allowing for shape changes to be specified. The identifier in the `prepro` section for ANSA is `ANSA`. Several files must be specified:

1. *ANSA executable*, typically named *ansa.sh*. Do not use an alias.
2. *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 and values from this file. Parameters defined in the parameter file will become constants with the same name and value in LS-OPT. The user can

change them to be design variables instead of constants in the variable panel of the GUI. If LS-OPT already has a design variable with the same name then this variable will be used to drive the value of the ANSA parameter.

3. *ANSA binary database*, typically with the extension *.ansa*.
4. *LS-DYNA executable.*
5. *LS-DYNA input file.* ANSA automatically produces a LS-DYNA keyword file called `ansaout` . This file will therefore automatically appear as the LS-DYNA input file in the GUI. However this file can also be used as an appended file or include file (specified under `*INCLUDE`). In this case an input file name has to be specified for LS-DYNA.

Example:

```
$
$ DEFINITION OF SOLVER "1"
$ Solver "1" uses ANSA
solver dyna '1'

$
prepro ANSA
prepro command "/home/jane/bin/ansa.sh"
prepro input file "model.txt"
propro database file "model.ansa"
$
solver command "lsdyna"
solver input file "ansaout"
$
```

10.3.4 AutoDV

The geometric preprocessor AutoDV can be interfaced with LS-OPT which allows shape variables to be specified. The identifier in the `prepro` section for the use of AutoDV is `templex` (the name of an auxiliary product: Templex[11]). The use of AutoDV requires several input files to be available.

1. *Input deck:* At the top, the variables are defined as `DVAR1`, `DVAR2`, etc. along with their current values. The default name is `input.tpl`. This file is specified as the `prepro input file`.

2. *Control nodes file:* This is a nodal template file used by Templex to produce the nodal output file using the current values of the variables. This file is specified using the `prepro controlnodes` command. The default name is `nodes.tpl`.

3. A coefficient file that contains original coordinates and motion vectors specified in two columns must be available. The command used is `prepro coefficient file` and the default file name is `nodes.shp`.

[11] Registered Trademark of Altair Engineering, Inc.

4. Templex produces a nodal output file that is specified under the `solver append file` command. The default name is `nodes.include`.

Example:

```
$
$ DEFINITION OF SOLVER "1"
$
solver dyna '1'
  solver command "lsdyna"
  solver append file "nodes.include"
  solver input file "dyna.k"
  prepro templex
  prepro command "/origin_2/user/mytemplex/templex"
  prepro input file "a.tpl"
  prepro coefficient file "a.dynakey.node.tpl"
  prepro controlnodes file "a.shp"
```

In the example, several files can be defaulted.

Table 10.3-1: Templex solver and prepro files and defaults

Command	Description	Default
`prepro input file`	Templex input file	`input.tpl`
`prepro coefficient file`	Coefficient file	`nodes.shp`
`prepro controlnodes file`	Control Nodes file	`nodes.tpl`
`solver append file`	Append file (same as templex output file)	`nodes.include`

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

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

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

```
/origin 2/user/mytemplex/templex a.tpl > nodes.include
```

Remarks:

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

   ```
   {DVAR1=23.77}
   {DVAR2=49.05}
   ```

   to replace the variables and bounds at the end of each line by the current values. The name DVAR1 (or DVAR2) is recognized by LS-OPT and displayed in the 'Variables' panel.

10.3.5 HyperMorph

To allow the specification of shape variables, the geometric preprocessor HyperMorph[12] has been interfaced with LS-OPT. The identifier in the `prepro` section for the use of HyperMorph is `hypermorph`.

1. *Input deck:* At the top, the variables are defined as:

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

   This file is specified as the `prepro input file`.

2. Templex produces a nodal output file that is specified under the `prepro output file` command.

Example:

```
$
$ DEFINITION OF SOLVER "1"
$
solver dyna '1'
  solver command "ls970.single"
  solver append file "nodes.include"
  solver input file "dyna.k"
  prepro hypermorph
   prepro command "/origin_2/user/mytemplex/templex"
   prepro input file "a.tpl"
   prepro output file "h.output"
```

Table 10.3-2: HyperMorph preprocessor input files and defaults

Command	Description
`prepro input file`	Templex input file
`prepro output file`	Output file produced by Templex (can e.g. be used as an include file in the analysis)

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

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

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

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

[12] Registered Trademark of Altair Engineering, Inc.

Remarks:

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

   ```
   {parameter(DVAR1,"Radius_1",1,0.5,3.0)}
   {parameter(DVAR2,"Radius_2",1,0.5,3.0)}
   ```

   to replace the variables and bounds at the end of each line by the current values. This name, e.g. `Radius_1` is recognized by LS-OPT and automatically displayed in the 'Variables' panel. 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*.

10.3.6 User-defined preprocessor

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. Only the `prepro command` statement will therefore be used, and no input file (`prepro input file`) will be specified.

The user-defined `prepro command` will be executed with the standard preprocessor input file `UserPreproOpt.inp` appended to the command. The `UserPreproOpt.inp` file is generated after performing the substitutions in the `prepro input file` specified by the user.

Example:

```
prepro own
prepro command "gambit -r1.3 -id ../../casefile -in "
prepro input file "setup.jou"
```

The executed command is:

```
gambit -r1.3 -id ../../casefile –in setup.jou
```

Alternatively, a script can be executed with the `prepro command` to perform any number of command line commands that result in the generation of a file called: `UserOpt.inp` for use by an own solver, or `DynaOpt.inp` for use by LS-DYNA.

10.4 Extra input files

A list of extra input files can be provided for the preprocessor or solver. A different set can be specified for each analysis case. The files must be placed in the main working directory and are copied from the main directory to the run directories before the start of the preprocessing. Parameters can be specified in the extra files using the LS-OPT parameter format (`<<parameter>>`) (see Section 10.1.1). Note that LS-DYNA include files do not have to be specified as extra files, since these are automatically processed. The files are specified in the GUI under the "Solvers" tab ("Extra files" sub-tab).

Command file syntax:

```
Solver extra file "extra_file_name"
```

Example:

```
solver extra file "inputfile1.txt"
solver extra file "inputfile2.txt"
solver extra file "inputfile3.txt"
```

11. Design Variables, Constants, and Dependents

This chapter describes the definition of the input variables, constants and dependents, design space and the initial subregion.

All the items in this chapter are specified in the Variables panel in LS-OPT*ui* (Figure 11-1). Shown is a multidisciplinary design optimization (MDO) case where not all the variables are shared. E.g., `t_bumper` in Figure 11-1 is only associated with the solver CRASH.

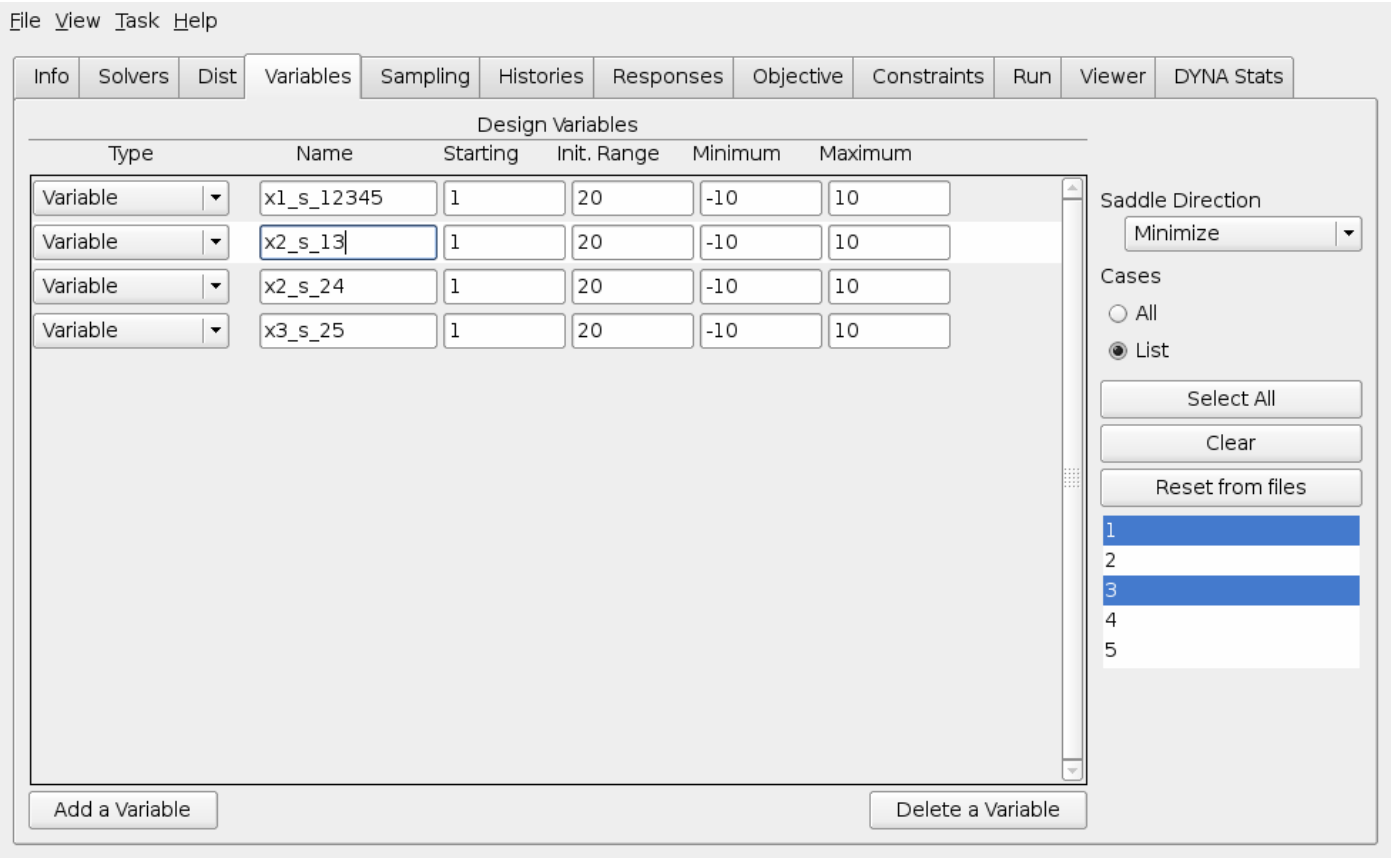


Figure 11-1: Variables panel in LS-OPT*ui*

11.1 Selection of design variables

The `variable` command is the identification command for each variable.

Command file syntax:

`variable` ***`variable_name`*** *value*

Example:

```
$ DEFINE THE VARIABLE: 'Area'
Variable 'Area' 0.8
```

The value assigned is the initial value of the variable.

11.2 Definition of upper and lower bounds of the design space

Command file syntax:

`Lower bound variable` ***`variable_name`*** *value* <-10^{+30}>
`Upper bound variable` ***`variable_name`*** *value* <$+10^{+30}$>

Example:

```
Lower bound variable 'Area' 0.1
Upper bound variable 'Area' 2.0
```

Both the lower and upper bounds must be specified, as they are used for scaling.

11.3 Size and location of region of interest (range)

Command file syntax:

`range` ***`variable_name subregion_size`***

Example:

```
$ RANGE OF 'Area'
range 'Area' 0.4
```

This will allow '`Area`' to vary from 0.6 to 1.0.

Remarks:

1. A value of 25-50% of the design space can be chosen if the user is unsure of a suitable value.

2. The full design space is used if the range is omitted.

3. 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.

11.4 Local variables

For multidisciplinary design optimization (MDO) certain variables are assigned to some but not all solvers (disciplines). In the command file the following syntax defines the variable as local:

Command file syntax:

```
local variable_name
```

See Section 22.6 for an example.

11.5 Discrete Variables

Discrete variables are defined using (i) a name, (ii) a starting value, and (iii) a list of allowable values. Specifying an initial range for the construction of a response surface is optional; the allowable values will be used to compute a default range. The following commands are therefore required to define a discrete variable:

Command file syntax:

```
variable variable_name value
variable variable_name discrete {discrete_value_1 … discrete_value_n}
```

Example:

```
variable 'Area' 3.1
variable 'Area' discrete {2.0 3.1 4.0 5}
```

11.6 Assigning variable to solver

If a variable has been flagged as local, it needs to be assigned to a solver. The command file syntax is:

Command file syntax:

```
Solver variable variable_name
```

See Section 22.6 for an example.

11.7 Constants

Each variable above can be modified to be a constant. See Figure 11-2 where this is the case for `t_bumper`.

Constants are used:

1. to define constant values in the input file such as π, e or any other constant that may relate to the optimization problem, e.g. initial velocity, event time, integration limits, etc.

2. 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.

Command file syntax:

`constant` *`constant_name value`*

Example:

```
constant 'Youngs_modulus' 2.07e8
constant 'Poisson_ratio' 0.3
dependent 'Shear_modulus' {Youngs_modulus/(2*(1 + Poisson_ratio))}
```

In this case, the dependent is of course not a variable, but a constant as well.

11.8 Dependent Variables

Dependent variables (see Figure 11-2 for example of definition in Variables panel) 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 D).

Command file syntax:

`dependent` *`variable_name expression`*

The string must conform to the rules for expressions and be placed in curly brackets. The dependent variables can be specified in an input template and will therefore be replaced by their actual values.

Example:

```
variable 'Youngs_modulus' 2.0e08
variable 'Poisson_ratio' 0.3
dependent 'Shear_modulus' {Youngs_modulus/(2*(1 + Poisson_ratio))}
```

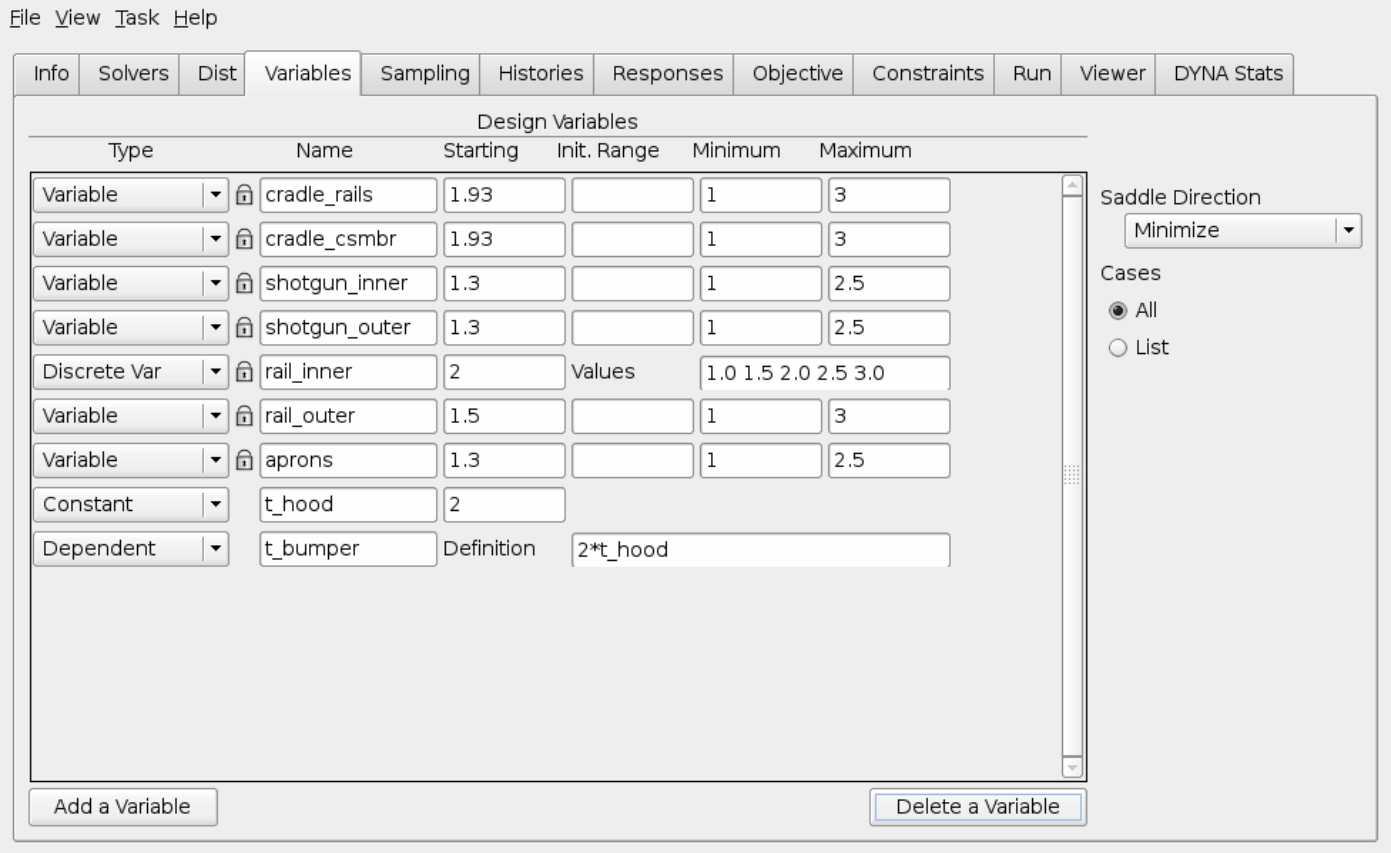


Figure 11-2: Variables panel in LS-OPT*ui* with Constants and Dependents. “Lock” symbols (before name) indicate that variables were automatically imported from input files.

11.9 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 case, but which are re-used in the current case. 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 case will always include the file in the corresponding directory in Case1

The `i0` format forces an integer specification (see Section 10.1.1 for a more detailed description).

Unfortunately the feature cannot be used with LS-DYNA `*PARAMETER` parameters.

11.10 Worst-case design

Worst-case or saddle-point design is where the objective function is minimized (or maximized) with respect to *some* variables, while it is maximized (or minimized) 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 Variables panel. The default selection is Minimize.

Command file syntax:

`Variable` *`variable_name`* `max`

Example:

```
variable 'head_orientation' max
```

12. Probabilistic Modeling and Monte Carlo Simulation

Probabilistic evaluations investigate the effects of variations of the system parameters on the system responses.

The variation of the system parameters is described using variables and probabilistic distributions describing their variation. Accordingly, the variation of the system responses, including information such as the nominal value of the response, reliability, and extreme values, can be computed. The source of the variation can be the variation of the design variables (control variables) as well as the variation of noise variables, whose the value is not under the control of the analyst such as the variation in a load.

More background on the probabilistic methods is given in Chapter 6 (the theoretical manual), while example problems can be found in Chapter 22.

12.1 Probabilistic problem modeling

Introducing the probabilistic effects into analysis requires the specification of:

1. Statistical distributions.
2. Assigning the statistical distributions to design variables.
3. Specification of the experimental design. For a Monte Carlo analysis a suitable strategy for selecting the experimental points must be specified; for example, a Latin Hypercube experimental design can be used to minimize the number of runs required to approximate the mean and standard deviation. However, if the Monte Carlo analysis is done using a metamodel, then the experimental design pertains to the construction of the metamodel.
4. The probabilistic analysis to be executed; for example, a Monte Carlo reliability analysis.

12.2 Probabilistic distributions

The probabilistic component of a design variable is described using a probabilistic distribution. The distributions are created without referring to a variable. Many design variables can refer to a single distribution.

12.2.1 Beta distribution

The beta distribution is quite versatile as well as bounded by two limits: *a* and *b*. The shape of the distribution is described by two parameters: *q* and *r*. Swapping the values of q and r produces a mirror image of the distribution.

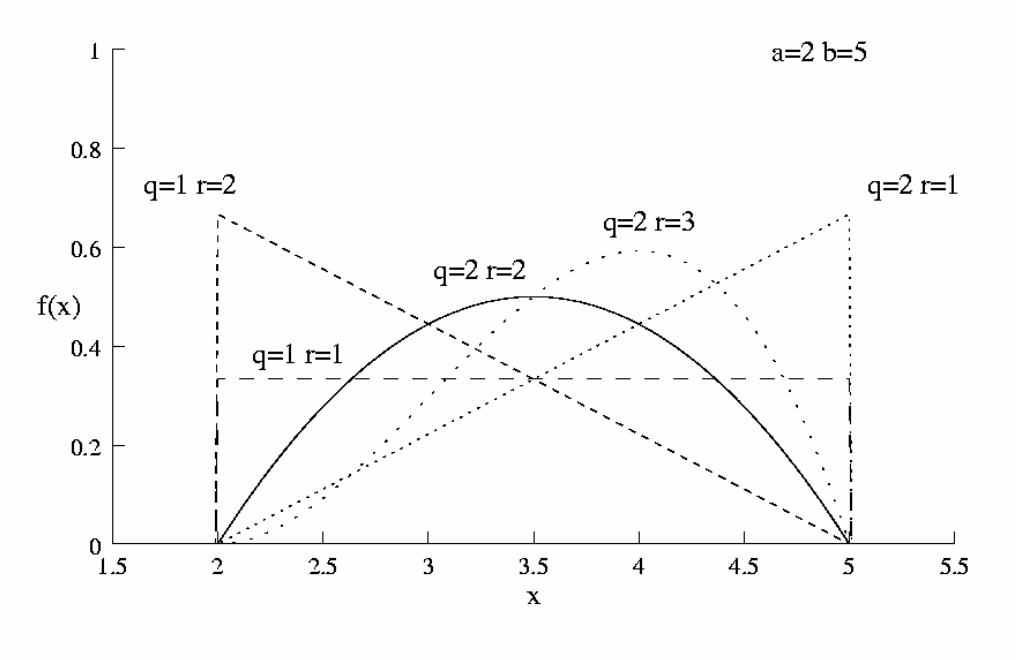


Figure 12-1 Beta distribution

Command file syntax:

```
distribution 'name' BETA a b q r
```

Item	Description
name	Distribution name
a	Lower Bound
b	Upper Bound
q	Shape parameter q
r	Shape parameter r

Example:

```
distribution 'distBeta' BETA 2.0 5.0 1.0 1.0
```

12.2.2 Binomial distribution

The binomial distribution is a discrete distribution describing the expected number of events for an event with probability *p* evaluated over *n* trails. For *n=1*, it is the Bernoulli distribution (experiments with two possible outcomes — success or failure) with probability of success *p*.

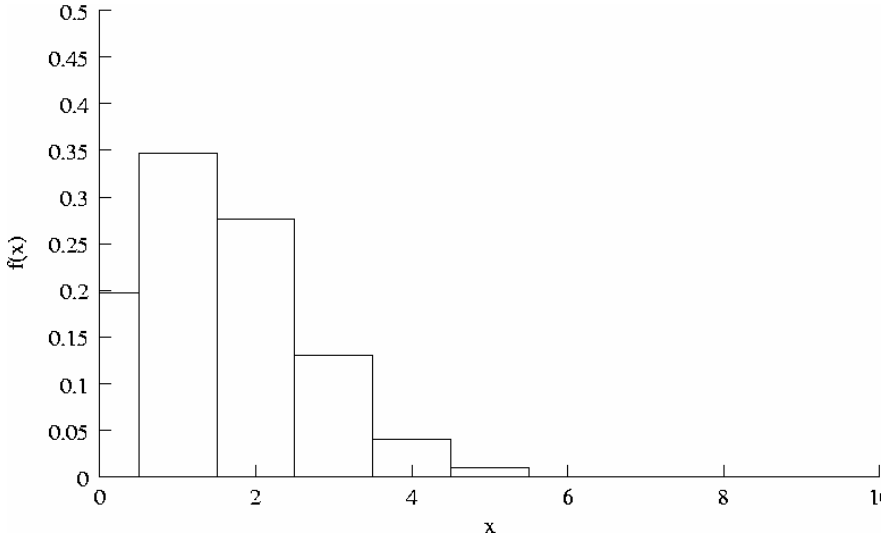


Figure 12-2 Binomial distribution

Command file syntax:

```
distribution 'name' BINOMIAL p n
```

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

Example:

```
distribution 'distBin' BINOMIAL 0.1 3
```

12.2.3 Lognormal distribution

If X is a lognormal random variable with parameters μ and σ, then the random variable $Y = ln\ X$ has a normal distribution with mean μ and variance σ^2.

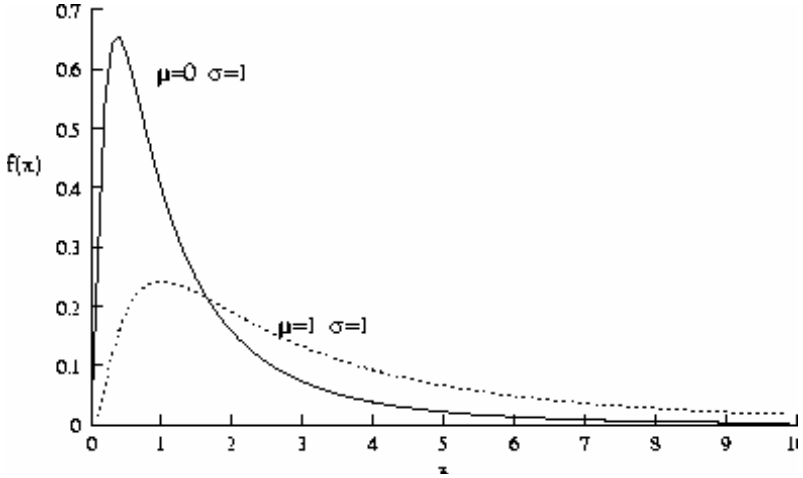


Figure 12-3 Lognormal distribution

Command file syntax:

```
distribution 'name' LOGNORMAL mu sigma
```

Item	Description
name	Distribution name
mu	Mean value in logarithmic domain
sigma	Standard deviation in logarithmic domain

Example:

```
distribution 'logDist' LOGNORMAL 12.3 1.1
```

12.2.4 Normal distribution

The normal distribution is symmetric and centered about the mean μ with a standard deviation of σ.

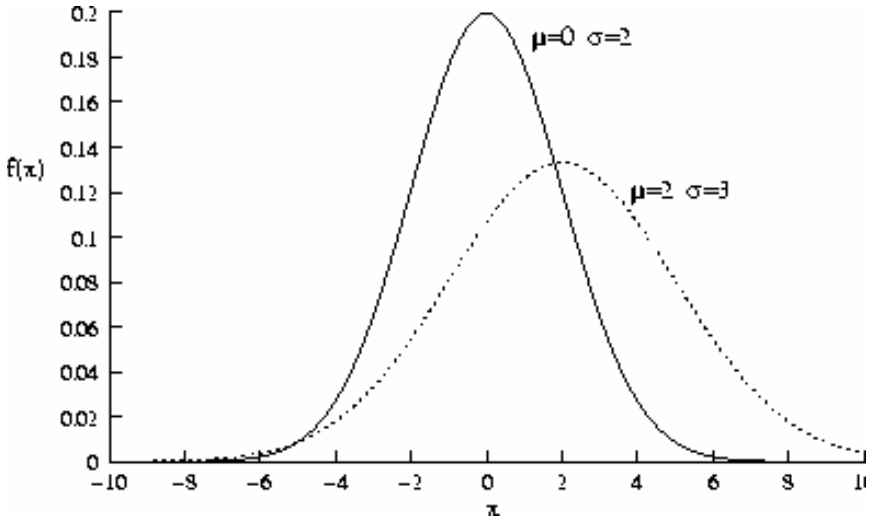


Figure 12-4 Normal Distribution

Command file syntax:

```
distribution 'name' NORMAL mu sigma
```

Item	Description
name	Distribution name
mu	Mean value
sigma	Standard deviation

Example:

```
distribution 'normalDist' NORMAL 12.2 1.1
```

12.2.5 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.

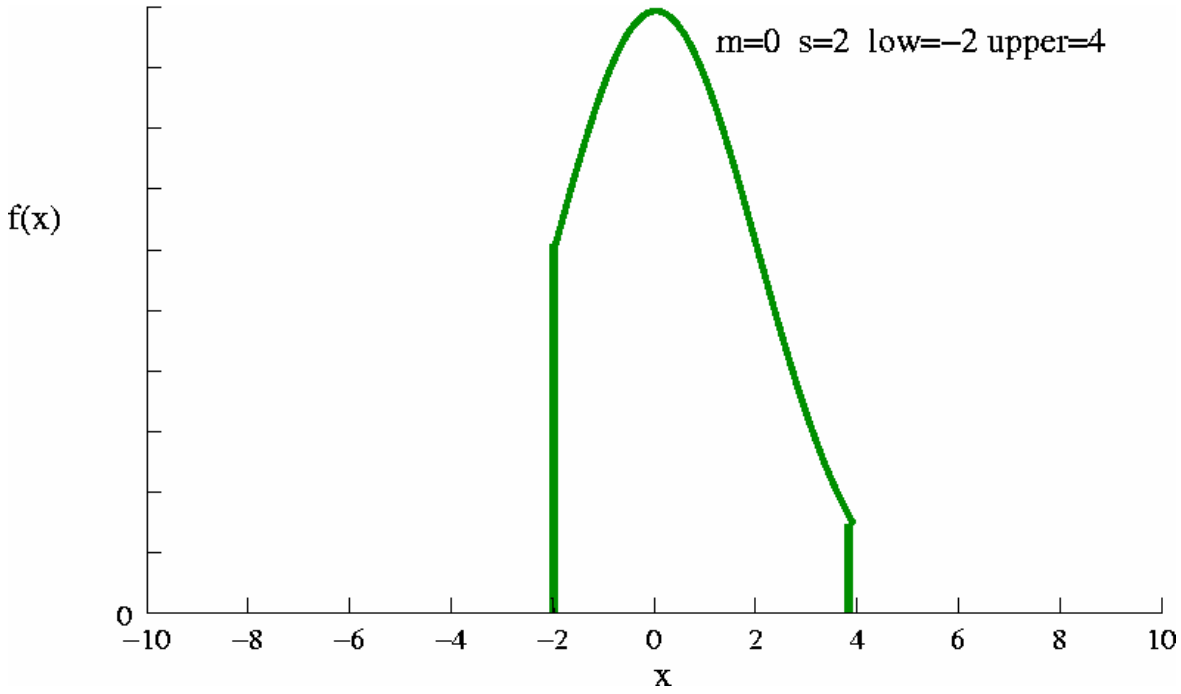


Figure 12-5 Truncated Normal Distribution

Command file syntax:

```
distribution 'name' TRUNCATED_NORMAL mu sigma low upper
```

Item	Description
name	Distribution name
mu	Mean value
sigma	Standard deviation
low	Lower bound on values
upper	Upper bound on values

Example:

```
distribution 'truncNormalDist' TRUNCATED_NORMAL 12.2 1.1 10.0 12.0
```

12.2.6 Uniform distribution

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

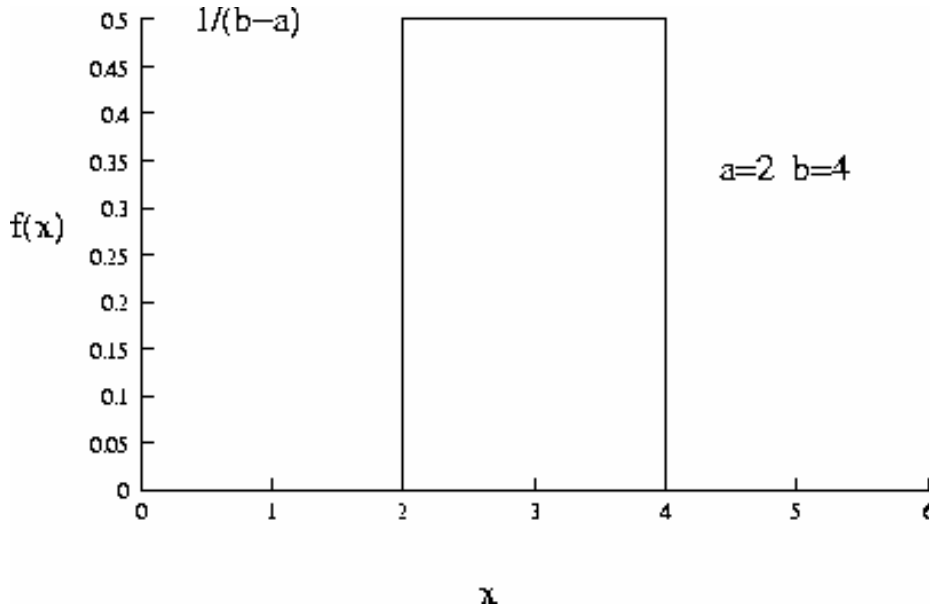


Figure 12-6 Uniform Distribution

Command file syntax:

```
distribution 'name' UNIFORM lower upper
```

Item	Description
name	Distribution name
lower	Lower bound
upper	Upper bound

Example:

```
distribution 'rangeX' UNIFORM 1.2 3.4
```

12.2.7 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.

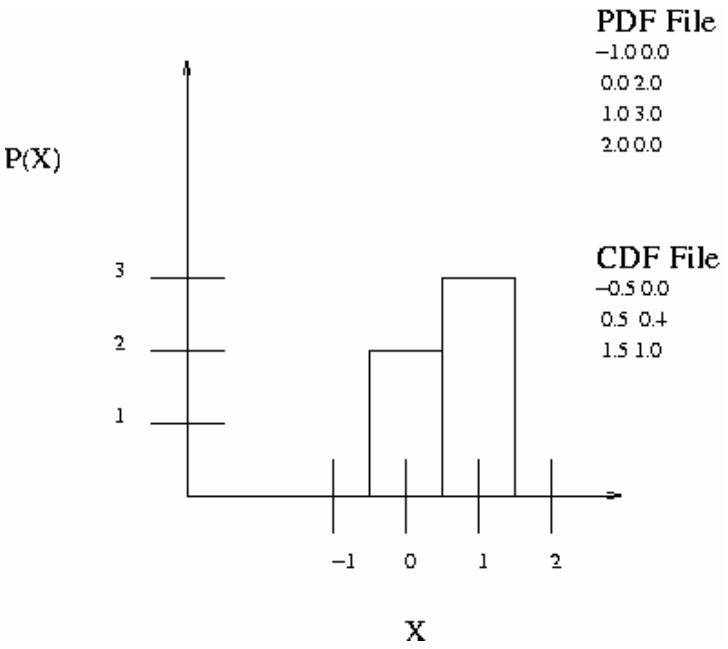


Figure 12-7 User defined distribution

Lines in the data file starting with the character '$' will be ignored.

Command file syntax:

```
distribution 'name' USER_DEFINED_PDF "fileName"
distribution 'name' USER_DEFINED_CDF "fileName"
```

Item	Description
name	Distribution name
filename	Name of file containing the distribution data

Example:

```
distribution 'bendDist' USER_DEFINED_PDF "bendingTest.pdf"
distribution 'testDat' USER_DEFINED_CDF "threePointTest.dat"
```

The file “bendingTest.pdf” contains:

```
$ 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
```

The file “threePointTest.dat” contains:

```
$ 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
4.5             1.00000
```

12.2.8 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.

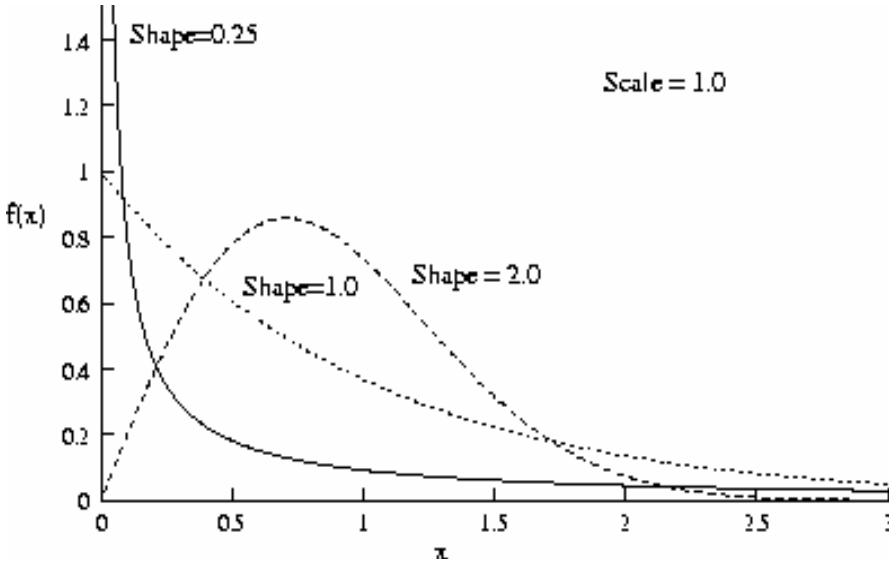


Figure 12-8 Weibull distribution

Command file syntax:

```
distribution 'name' WEIBULL scale shape
```

Item	Description
name	Distribution name
scale	Scale parameter
shape	Shape parameter

Example:

```
distribution 'wDist' WEIBULL 2.3 3.1
```

12.3 Probabilistic variables

A probabilistic variable is completely described using a statistical distribution. From this statistical distribution defines the mean or nominal value as well as the variation around this nominal value. Note that some special rules apply to control variables, the mean of which can be adjusted by the optimization algorithm.

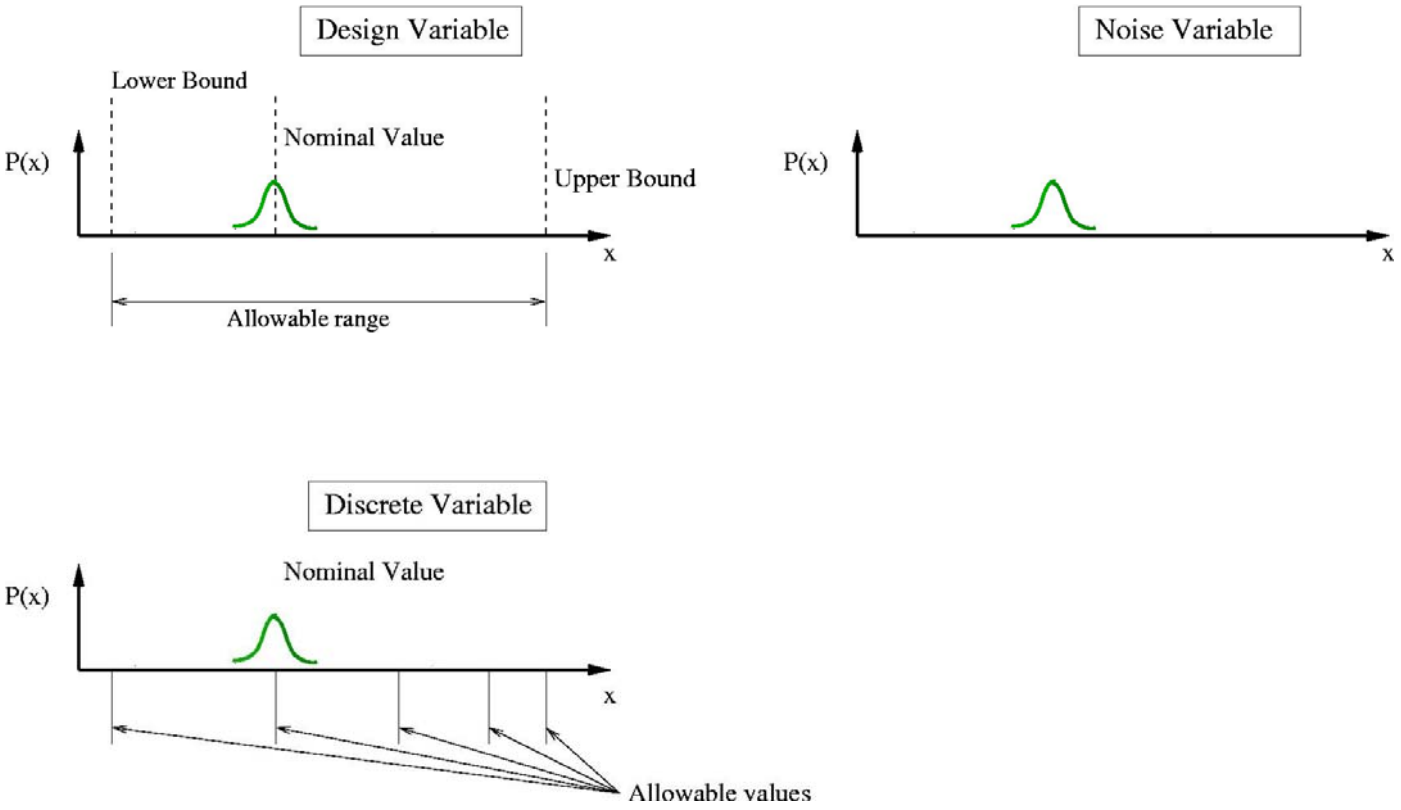


Figure 12-9 Probabilistic variables. The nominal value of a control variable can be adjusted by the optimization algorithm between the lower and upper bound; the probabilistic variation of a design variable is around this nominal value. A noise variable is described completely by the statistical distribution. A discrete variable, like design variable has a nominal value selected by the optimization algoritm; the probabilistic variation of the discrete variable is around this nominal value.

A distinction is made between control and noise variables:

- **Control variables**: Variables that can be controlled in the design, analysis, and production level; for example: a shell thickness. It can therefore be assigned a nominal value and will have a variation around this nominal value. The nominal value can be adjusted during the design phase in order to have a more suitable design. A discrete variable is a special case of a control variable.
- **Noise variables:** Variables that are difficult or impossible to control at the design and production level, but can be controlled at the analysis level; for example: loads and material variation. A noise variable will have the nominal value as specified by the distribution, that is follow the distribution exactly.

A variable is declared probabilistic by:

- Creating it as a noise variable or
- Assigning a distribution to a control variable or
- Creating it as linked to an existing probabilistic 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.

Command file syntax:

```
noise variable 'variableName' distribution 'distributionName'
variable 'variableName' distribution 'distributionName'
variable 'variableName' link variable 'variableName'
```

Item	Description
variableName	Variable identifier
distributionName	Distribution identifier

Example:

```
$ Create a noise variable
Noise Variable 'windLoadScatter' distribution 'windLoadData'
$ Assigning a distribution to an existing control variable
Variable 'Var-D-1' Distribution 'dist-1'
$ Creating a variable by linking it to another.
Variable 'Var-D-2' Link variable 'Var-D-1'
```

12.3.1 Setting the nominal value of a probabilistic variable

If no nominal value is specified for a control variable, then the nominal value of the distribution is used.

If the nominal value of a control variable is specified, then this value is used; the associated distribution will be used to describe the variation around this nominal value. For example: a variable with a nominal value of 7 is assigned a normal distribution with μ=0 and σ=2; the results 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.

12.3.2 Bounds on probabilistic variable values

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. The user can control this behavior.

A noise variable is bounded by the distribution specified and does not have upper and lower bounds similar to control variables. However bounds are required for the construction of the approximating functions and are chosen as described in the next subsection.

Command file syntax:

```
set variable distribution bound state
```

Item	Description
state	Whether the bounds must be enforced for the probabilistic component of the variable.

Example:

```
$ ignore bounds on control variables
set variable distribution bound 0
$ Respect bounds on control variables
set variable distribution bound 1
```

12.3.3 Noise variable subregion size

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. The number of standard deviations can however be set by the user. In general, a noise variable is bounded by the distribution specified and does not have upper and lower bounds similar to control variables.

Command file syntax:

```
set noise variable range standardDeviations
```

Item	Description
standardDeviations	The subregion size in standard deviations for the noise variable.

Example:

```
$ Set noise var bounds to 1.5 standard deviations
$ for defining subregion for creating approximation
set noise variable range 1.5
```

12.4 Probabilistic simulation

The following simulation methods are provided:

- Monte Carlo.
- Monte Carlo using metamodels.

The upper and lower bounds on constraints will be used as failure values for the reliability computations.

12.4.1 Monte Carlo analysis

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. Sampling schemes must be duplicated across disciplines if composite functions must be computed for each point, because if the experimental designs differ across disciplines, then composite functions referring to responses in more than one discipline can not be computed.

Command file syntax:

```
analyze Monte Carlo
```

Example:

```
analyze Monte Carlo
```

12.4.2 Monte Carlo analysis using a metamodel

The Monte Carlo analysis will be done using the metamodels – response surfaces, neural networks, or Kriging – as prescribed by the user.

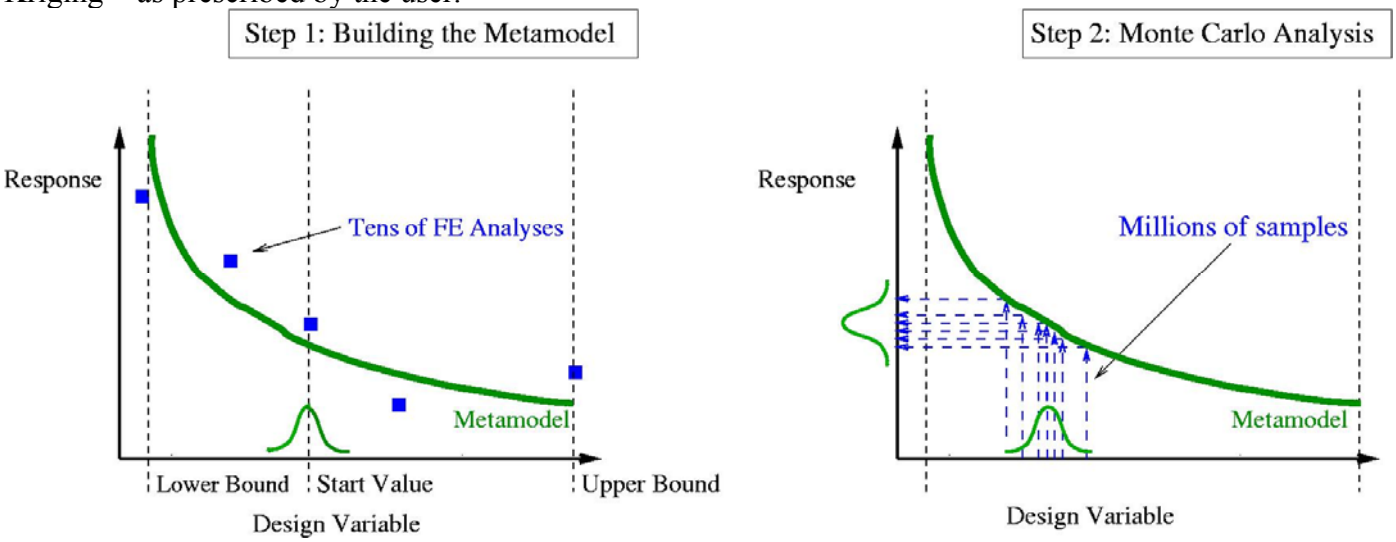


Figure 12-10 Metamodel-based Monte Carlo analysis. The method proceed in two steps: firstly a metamodel is created, and then the Monte Carlo simulation is done using the metamodel and the statistical distribution of the variable. Note that the metamodel for a design/control variable is constructed considering 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.

The number of function evaluations can be set by the user. The default value is 10^6. The function evaluations are done using designs chosen randomly respecting the distributions of the design variables and are evaluated using the metamodels.

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.

Command file syntax:

```
analyze metamodel monte carlo
```

Example:

```
analyze metamodel monte carlo
```

12.4.3 FORM (First Order Reliability Method) analysis

A FORM evaluation will:

- Construct the metamodels – response surfaces, neural networks, or Kriging – as prescribed by the user. If the metamodels already exists, then they won't be recreated.
- Conduct a FORM analysis for every constraint using the metamodels.

The following are computed in a FORM analysis:

- The most probable point (see Section 6.4.5)
- 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 6.4.6. See also Section 19.3 for more information about Reliability Based Design Optimization (RBDO).

Command file syntax:

```
analyze metamodel FORM
```

Example:

```
analyze metamodel FORM
```

12.4.4 Accuracy of metamodel based Monte Carlo

The number of function evaluations to be analyzed can be set by the user. The default value is 10^6.

Command file syntax:

```
set reliability resolution m
```

Item	Description
m	Number of sample values

Example:

```
set reliability resolution 1000
```

12.4.5 Histograms of responses

This option outputs histograms of the variables, dependend variables, responses, and composite functions. This feature is only available for the Monte Carlo analysis procedures. It can be memory intensive for metamodel-based computations because all the responses values must be kept in memory.

The default is not to output any histograms. The histograms can always be viewed in Viewer as a post-processing operation.

Command file syntax:

```
set histogram intervals
```

Item	Description
intervals	Number of intervals in histogram

Example:

```
set histogram 12
```

12.4.6 Adding the noise component to metamodel Monte Carlo computations

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. The default is not to add the noise to the computations.

Command file syntax:

```
set metamodel noise true_false
```

Item	Description
true_false	0 for not adding noise; 1 otherwise

Example:

```
set metamodel noise 0 $ default: noise not added in computation
set metamodel noise 1 $ noise included in computation
```

12.5 Stochastic Contribution Analysis (DSA)

It can be useful to know how the variation of each design variable contributes to the variation of a response.

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 stochastic contribution is computed analytically for response surfaces. For neural networks, Kriging models, and composite functions, two options are available:

1. Approximate using second order response surface (default). The response surface is built using three times the number of terms in the response surface using a central points Latin hypercube experimental design over a range of plus/minus two standard deviations around the mean.
2. Use Monte Carlo. The number of points used will be the same as used for a metamodel based Monte Carlo analysis. A large number of points (10 000 or more) is required. This option, using 100 000 points, is the default method.

Theoretical concerns are discussed in Section 6.7.

Command file syntax:

```
set dsa method monte carlo
set dsa method  meta model
set dsa resolution m
```

Item	Description
m	Number of sample values

Example:

```
set dsa method  meta model
$ Use Monte Carlo simulation
set dsa method monte carlo
$ use 1000 points in the Monte Carlo simulation
set dsa resolution 1000
```

12.6 Covariance

The covariance and coefficient of correlation of the responses will be printed for a Monte Carlo analysis.

The covariance and coefficient of the responses can also be examined in the Viewer part of the GUI.

Theoretical concerns are discussed in Section 6.3.2.

12.7 Robust Design

The implementation of robust design in LS-OPT only required that the variation of a response be available as a composite. The standard deviation of a response is therefore available for use in a constraint or objective, or in another composite.

The LS-OPT command defining the standard deviation of another response or composite to be a composite is:
composite 'var x11' noise 'x11'

The variation of response approximated using response surfaces is computed analytically as documented for the LS-OPT stochastic contribution analysis. 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.*

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_i^2 terms can be included for a more accurate computation of the variance.

13. Metamodels and Point Selection

This chapter describes the specification of the metamodel types and point selection schemes (design of experiments or DOE). The terms *point selection* and *experimental design* are used interchangeably.

13.1 Metamodel definition

The user can select from three metamodel types in LS-OPT. The standard and default selection is the polynomial response surface method (RSM) where response surfaces are fitted to results at data points using polynomials. For global approximations, neural network or Kriging approximations are available. Sensitivity data (analytical or numerical) can also be used for optimization. This method is more suitable for linear analysis solvers.

Command file syntax:

```
Solver order [linear|interaction|elliptic|quadratic|FF|RBF|user]
```

The linear, interaction (linear with interaction effects), elliptic and quadratic options are for polynomials. `FF` represents the Feedforward Neural network and `RBF` represents the radial basis function network.

13.1.1 Response Surface Methodology

When polynomial response surfaces are constructed, the user can select from different approximation orders. The available options are linear, linear with interaction, elliptic and quadratic. Increasing the order of the polynomial results in more terms in the polynomial, and therefore more coefficients. In LSOPT*ui*, the approximation order is set in the Order field. See Figure 13-1.

The polynomial terms can be used during the variable screening process (see Section 2.4) 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 18.5).

The recommended point selection scheme for polynomial response surfaces is the *D*-optimal scheme (Section 13.2.2).

13.1.2 Neural Networks and Radial Basis Function Networks

To apply neural network or radial basis functions approximations, select the appropriate option in the Metamodel field in LS-OPT*ui*. See Figure 13-2. The recommended Point Selection Scheme for neural networks and radial basis functions is the space filling method. The user can select either a sub-region (local) approach, or update the set of points for each iteration to form a global approximation. An updated network is fitted to *all* the points. See Section 13.7 for more detail on updating.

13.1.3 Variability of Neural Networks*

Because of the natural variability of neural networks (see Section 3.1.2), the user is allowed to select the number of members in a neural net committee and the centering (averaging) procedure to be used. To ensure distinct members, the regression procedure uses new randomly selected starting weights for generating each committee member. The syntax is shown below.

Command file syntax:

```
solver FF_committee size [number_of_members]
solver FF_committee discard [number_of_members]

solver FF_committee use [MEAN|MEDIAN]

solver FF_committee seed [integer_value]
```

The selected attributes apply to the current solver. A seed can be provided to the random number generator (see Section 2.2.7) to ensure a repeatable (but different) committee.

The `discard` option allows the user to discard *number_of_members* committee members with the lowest mean squared fitting error and the *number_of_members* 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 in the `MEAN` or `MEDIAN` calculation is therefore 2* *number_of_members*.

The `discard` feature is activated during the regression procedure whereas the averaging function (mean/median) is only used during the evaluation procedure.The use of the `MEDIAN` option simply finds the median value of all the member values obtained at a point, *so different points in the parameter space may not be represented by the same member and the neural net surface plot may be discontinuous*. If a single median neural net is desired, the user must generate an uneven committee size *n* and then `discard` the truncated integer value of *n*/2 members, e.g. size=5 and discarded=2, 9 and 4, 17 and 8, etc. Size=1 and discarded=0 is the least expensive.

The `seed` feature allows the generation of a unique set of neural networks. This feature can be used for sensitivity studies. In this case the user must provide a different seed for each new set of networks for the specific solver.

The default attributes of committees are given in Table 13.1-1. This selection creates a committee of 5 nets and finds the mean value during evaluation. The data for all 5 nets appears in the database file for each

specific net, e.g. `Net.<variable_name>.<iteration_number>` in the solver subdirectory.

The variance of the predicted result is reported.

Table 13.1-1: Default values for Neural Net committees

Option	Default
Size	9
Discard	(int) (Size + 3/2)/4
Averaging type	MEAN
Seed	0

Please refer to Sections 3.3 and 4.7 for recommendations on how to use metamodels.

13.1.4 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, and Gaussian RBF. 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. The syntax is shown below.

Command file syntax:

```
solver RBF transfer [HMQ|GAUSS]
solver RBF optimize [GCV|GCV Ratio]
```

13.1.5 Efficiency of Neural Networks*

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. When a large number of points have been simulated however, the committee size can be reduced to a single neural net using

```
solver FF_committee size 1
```

Ensemble size. The ensemble size can be reduced in two ways: (i) by exactly specifying the architecture of the ensemble and (ii) by providing a threshold to the RMS training error. The architecture is specified as follows:

Command file syntax:

```
Solver FF_committee ensemble add number_of_hidden_nodes
```

e.g.

```
Solver FF_committee ensemble add 0
Solver FF_committee ensemble add 1
Solver FF_committee ensemble add 2
```

represents an ensemble of 0 (linear), 1 and 2 hidden nodes or 0-1-2 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. Higher order neural nets are more expensive to compute.

The threshold for the RMS error is specified as:

Command file syntax:

```
Solver FF_committee rmserror threshold
```

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.

Example:

```
Solver FF_committee rmserror 0.1
```

for a 10% threshold.

See Figure 13-2 for how to specify efficiency options in the GUI.

13.1.6 User-defined metamodel

The user-defined metamodel distribution is contained in the `user_metamodel_dist` directory in the LS-OPT distribution.

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. It can be placed either in a central repository (which needs to be pointed out by the "`solver user metamodel path`" command (see below), or in the same directory as the command file that refers to it.

Referring to user-defined metamodels in LS-OPT command files

In order to use a user-defined metamodel for a certain solver, add the command "solver order user" to the command file, under the appropriate solver.

The following commands apply for user defined metamodels:

Command file syntax:

```
Solver order user
```

The command enables the use of a user-defined metamodel for the current solver.

Solver user metamodel *'name'*

Example:

```
Solver user metamodel 'mymetamodel'
```

Gives the name of the user-defined metamodel (e.g. umm_**mymetamodel**_linux_i386.so). Note this 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.

Solver user metamodel path "*path*"

Example:

```
solver user metamodel path "/home/joe/metamodels"
```

specifies where the user defined metamodel may be found. If it is not found in the given directory (or that directory does not exist), LS-OPT will look in the same directory as the current command file. This parameter is optional.

Solver user metamodel command "*string*"

Example:

```
Solver user metamodel command "do it right"
```

Allows the user to send one string parameter to the user-defined metamodel, that may be used in any way by the metamodel. This parameter is optional.

Solver user metamodel param *value*

Example:

```
solver user metamodel param 1.3
```

Allows the user to send a numeric parameter to the user defined metamodel. This statement may be given multiple times for one solver in order to pass many parameters to the metamodel. It is up to the metamodel to specify which, if any, parameters it requires for operation.

13.2 Point Selection Schemes

13.2.1 Overview

Table 13.2-1 shows the available point selection schemes (experimental design methods).

Table 13.2-1: Point selection schemes

Experiment Description	Identifier	Remark
Linear Koshal	`lin_koshal`	For polynomials
Quadratic Koshal	`quad_koshal`	
Central Composite	`composite`	

D-optimal designs		
D-optimal	`dopt`	Polynomials
Factorial Designs		
2^n	`2toK`	
3^n	`3toK`	
⋮	⋮	⋮
11^n	`11toK`	

Random designs		
Latin Hypercube	`latin_hypercube`	For probabilistic analysis
Monte Carlo	`monte_carlo`	
Space filling designs		
Space filling 5 (recommended)	`space_filling`	Algorithm 5 (Section 2.2.6)
Space filling 0	`monte_carlo`	-
Space filling 1	`lhd_centralpoint`	-
Space filling 2	`lhd_generalized`	-
Space filling 3	`maximin_permute`	-
Space filling 4	`maximin_subinterval`	-

User defined designs		
User-defined	`user`	
Plan	`plan`	

Command file syntax:

```
Solver order [linear|interaction|elliptic|quadratic|FF|kriging|user]
Solver experimental design point_selection_scheme
Solver basis experiment basis_experiment
Solver number experiment number_experimental_points
Solver number basis experiments number_basis_experimental_points
```

Example 1:

```
Solver order quadratic
Solver experimental design dopt
Solver basis experiment 5toK
```

Example 2:

```
Solver order linear
Solver experimental design dopt
Solver number experiments 40
Solver basis experiment latin_hypercube
Solver number basis experiments 1000
```

In Example 1, the default number of experiments will be selected depending on the number of design variables. In Example 2, 40 points are selected from a total number of 1000.

In LS-OPT*ui*, the point selection scheme is selected using the Point Selection panel (Figure 13-1).

The default options are preset and are based on the number of variables, 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 (Figure 13-1), and the space-filling scheme is the default for the Neural Net and Kriging methods (Figure 13-2).

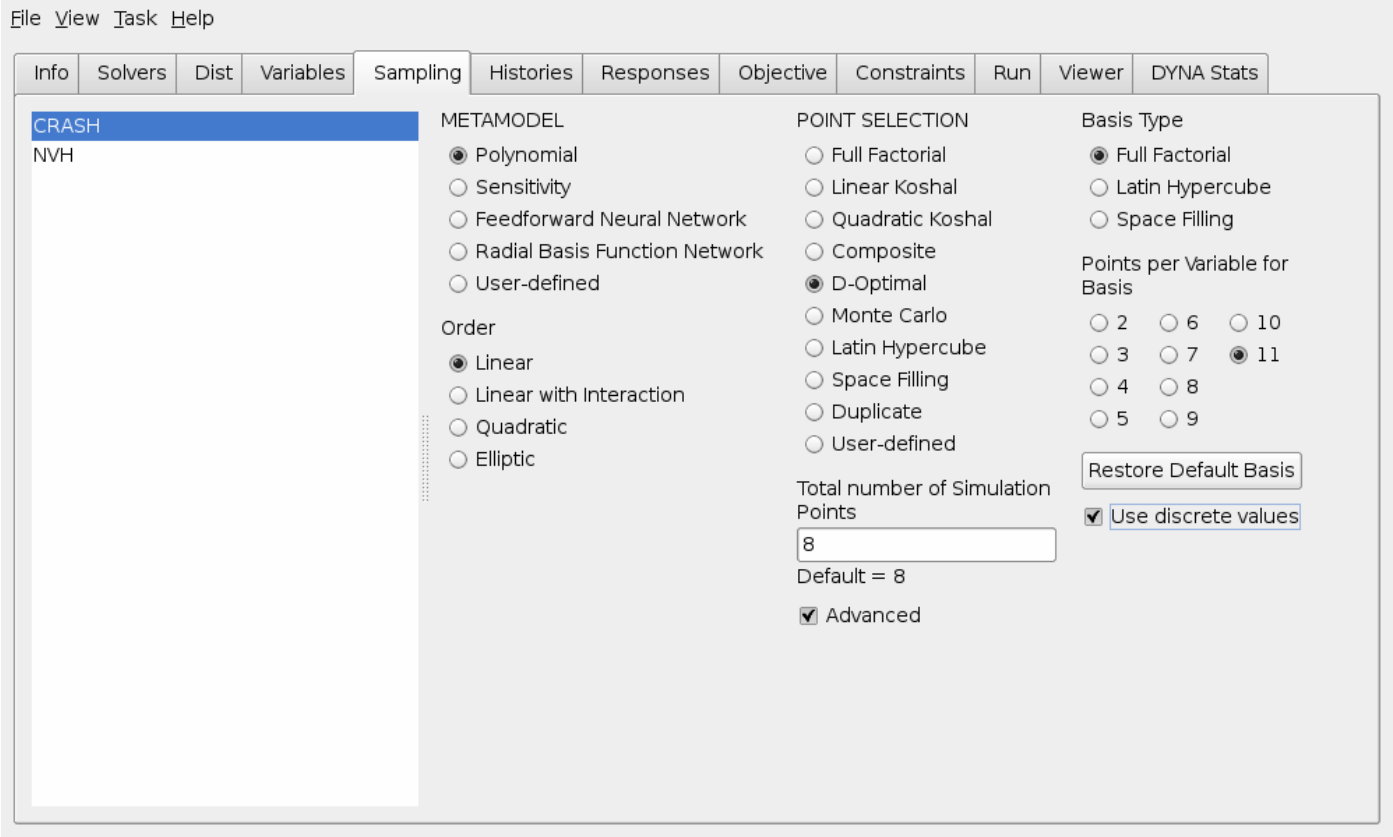


Figure 13-1: Metamodel and Point Selection panel in LS-OPT*ui* (Advanced options (basis experimental design) displayed)

13.2.2 *D*-Optimal point selection

The *D*-optimal design criterion 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 and is referred to here as the *basis experiment*. The *order* of the functions used has an influence on the distribution of the optimal experimental design.

The following must be defined to select *D*-optimal points.

`Order`	The order of the functions that will be used. Linear, linear with interaction, elliptic or quadratic.
`Number experiments`	The number of experimental points that must be selected.
`Basis experiment`	The set of points from which the *D*-optimal design points must be chosen, e.g. `3tok`
`Number basis experiments`	The number of basis experimental points (only `random`, `latin hypercube` and `space filling`).

The default number of points selected for the *D*-optimal design is `int`$(1.5(n+1))+1$ for linear, `int`$(1.5(2n+1))+1$ for elliptic, `int`$(0.75(n^2+n+2))+1$ for interaction, and `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, the command "`solver number experiments`" is required.

The default basis experiment for the *D*-optimal design is based on the number of variables. For small values of *n*, the full factorial design is used, whereas larger *n* employs a space filling method for the basis experiment. The basis experiment attributes can be overridden using the commands: `solver basis experiment` and `solver number basis experiments`.

13.2.3 Latin Hypercube Sampling

The Latin Hypercube point selection scheme is typically used for probabilistic analysis.

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^n design is more than 14 million.

The Monte Carlo, Latin Hypercube and Space-Filling point selection schemes require a user-specified number of experiments.

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 during the regression procedure. It is therefore better to use the *D*–optimal experimental design for RSM.

Example:

```
Solver order linear
Solver experimental design latin_hypercube
Solver number experiment 20
```

13.2.4 Space filling

Only algorithm 5 (see Section 2.2.6) is available in LS-OPTui. This algorithm maximizes the minimum distance between experimental design points. The only item of information that the user must provide for this point selection scheme, is the number of experimental design points. Space filling is useful when applied in conjunction with the Neural Net (neural network) and Kriging methods (see Section 13.1.2).

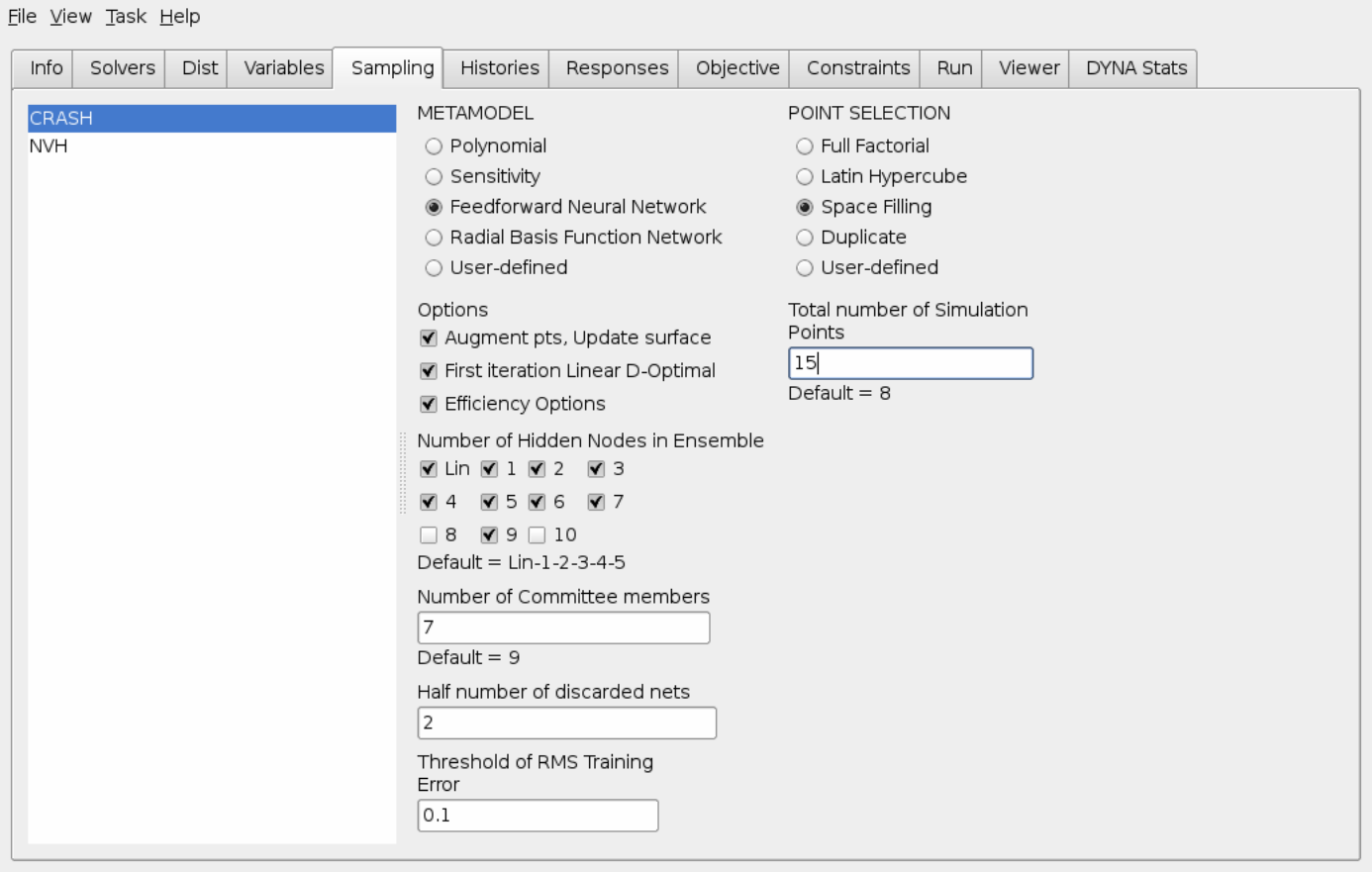


Figure 13-2: Selecting the Feedforward neural network approximation method in the Point Selection panel (Efficiency options displayed).

13.2.5 User-defined point selection

A user-defined experimental design can be specified in a text file using a free format. The `user` option ("User-defined" in the GUI) allows the user to import a table from a text file with the following keyword-based 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 "Variables" panel. 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) if the user switches to the "Variables" panel only after firstly selecting the file to be imported in the "Sampling" panel.

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.

13.3 Sampling at discrete points

A flag is provided to select the sampling points at specified discrete values of the variables. Discrete sampling will also handle discrete-continuous problems correctly. In the GUI, a check box is located as a *D*-Optimal advanced option for each case (See Figure 13-1). Discrete sampling is based on selecting a discrete basis set for *D*-Optimality and is therefore not available for other point selection schemes. Discrete sampling is only available if discrete variables are specified.

See Section 11.5 for how to specify a discrete variable.

Command file syntax:

```
Solver basis experiment discrete
```

13.4 Duplicating an experimental design

When executing a search method (see e.g. Section 4.6) for a multi-case or multidisciplinary optimization problem, the design points of the various disciplines must be duplicated so that all the responses and composites can be evaluated for any particular design point. The command must appear in the environment of the solver requiring the duplicate points. An experimental design can therefore be duplicated as follows:

Command file syntax:

```
solver experiment duplicate string
```

where *string* is the name of the master solver in single quotes, e.g.

```
Solver experiment duplicate 'CRASH'
```

'CRASH' is the master experimental design that must be copied exactly.

Multi-case composites not accompanied by case duplication cannot be visualized in 2-D or 3-D point plots. This is a mandatory step for using 'Direct GA' solver with multiple cases.

See also the example in Section 22.5.

13.5 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 design with additional experimental points. This can be performed by constructing a user-defined experiment as follows.

User-defined experiments can be placed in a file named `Experiments.PRE.`*`casename`* in the work directory. These will be used in the first iteration only for the case with name *casename*. The user can augment this set *D*-optimally by requesting a number of experiments greater than the number of lines in `Experiments.PRE.`*`casename`*. Each experiment must appear on a separate line with spaces, commas or tabs between values.

13.6 Specifying 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. Therefore, once the first approximation has been established, all the designs will be contained in the new region of interest. This region of interest is thus defined by approximate constraint bounds and by variable bounds. The purpose of an irregular design space is to avoid designs which may be impossible to analyze.

The `move/stay` commands can be used to define an environment in which the constraint bound commands (Section 16.4) can be used to double as bounds for the reasonable design space.

If a reasonable experimental design is required from the start, a `DesignFunctions.PRE.`*`case_name`* file can be provided by the user. This is however not necessary if explicit constraints, i.e. constraints that do not require simulations, are specified for the reasonable design space. An explicit constraint may be a simple relationship between the design variables.

The `move start` option moves the designs to the starting point instead of the center point (see Section 2.2.8). This option removes the requirement for having the center point inside the reasonable design space.

Command file syntax:

```
move
stay
move start
```

Example 1:

```
$ SET THE BOUNDS ON THE REASONABLE DESIGN SPACE
Lower bound constraint 'Energy' 4963.0
move
Upper bound constraint 'Energy' 5790.0
```

```
stay
Lower bound constraint 'Force' -1.2
Upper bound constraint 'Force' 1.2
```

The example above shows the lines required to determine a set of points that will be bounded by an upper bound on the Energy.

Example 2:

```
Variable 'Radius_1' 20.0
Variable 'Radius_2' 20.0
.
.
Composite 'TotalR' {Radius_1 + Radius_2}
move
Constraint 'TotalR'
Upper bound constraint 'TotalR' 50
```

This specification of the `move` command ensures that the points are selected such that the sum of the two variables does not exceed 50.

Remarks:

1. For constraints that are dependent on simulation results, a reasonable design space can only be created if response functions have already been created by a previous iteration. The mechanism is as follows:

   - *Automated design:* After each iteration, the program converts the database file `DesignFunctions` to file `DesignFunctions.PRE` in the solver directory. `DesignFunctions.PRE` then defines a reasonable design space and is read at the beginning of the next design iteration.
   - *Manual (semi-automated) Procedure:* If a reasonable design space is to be used, the user must ensure that a file `DesignFunctions.PRE.`*`case_name`* is resident in the working directory before starting an iteration. This file can be copied from the `DesignFunctions` file resulting from a previous iteration.

2. A reasonable design space can only be created using the *D*-optimal experimental design.
3. The reasonable design space will only be created if the center point (or the starting point in the case of `move start`) of the region of interest is feasible.
   Feasibility is determined within a tolerance of $0.001*|f_{max} - f_{min}|$ where f_{max} and f_{min} are the maximum and minimum values of the interpolated response over all the points.
4. The move feature should be used with extreme caution, since a very tightly constrained experimental design may generate a poorly conditioned response surface.

13.7 Automatic updating of an experimental design

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 or Kriging surfaces in combination with a space filling scheme.

Command file syntax:

```
solver update doe
```

The new points have the following properties:

- They are located within the current region of interest.
- The minimum distance between the new points and between the new and existing points, is maximized (space filling only).

13.8 Using design sensitivities for optimization

Both analytical and numerical sensitivities can be used for optimization. The syntax for the `solver experimental design` command is as follows:

Experiment Description	Identifier
Numerical Sensitivity	`numerical_DSA`
Analytical Sensitivity	`analytical_DSA`

13.8.1 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 LS-OPT*ui*, the Metamodel (Point Selection panel) must be set to Sensitivity Type → Analytical. See Figure 13-3.

Example:

```
Solver experimental design analytical_DSA
```

A complete example is given in Section 0.

13.8.2 Numerical sensitivities

To use numerical sensitivities, select Numerical Sensitivities in the Metamodel field in LS-OPT*ui* and assign the perturbation as a fraction of the design space.

Numerical sensitivities are computed by perturbing n points relative to the current design point $\mathbf{x}_0$, where the j-th perturbed point is:

$$x_i^j = x_i^0 + \delta_{ij}\varepsilon\left(x_{iU} - x_{iL}\right)$$

$\delta_{ij} = 0$ if $i \neq j$ and 1.0 if $i = j$. The perturbation constant ε is relative to the design space size. The same value applies to all the variables and is specified as:

Command file syntax:

```
Solver perturb perturbation_value
```

Example:

```
Solver experimental design numerical_DSA
Solver perturb 0.01
```

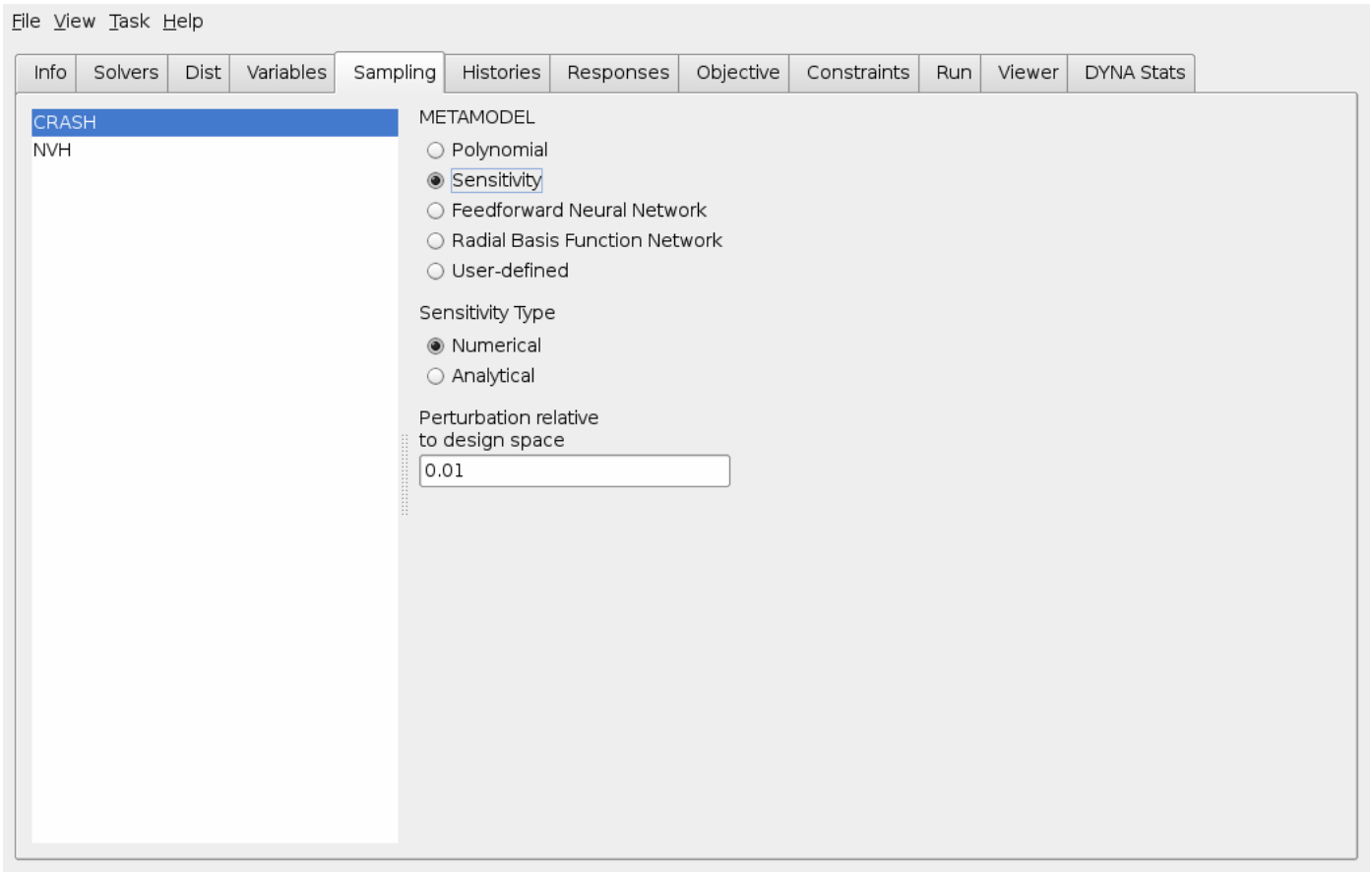


Figure 13-3: Selecting Sensitivities in the Point Selection panel

13.9 Checkpoints

The error measures of any number of designs (checkpoints) can be evaluated using an existing metamodel. There are two simple step to obtaining a table with error data.

1. Browse for the file with the checkpoint information using the **Checkpoints** tab in the **Solvers** panel. The file must be in the format of the AnalysisResults file (see Appendix C.2) pertaining to the selected case.
2. Use the **Analyze checkpoints** option in the **Repair** task menu and run the task (Section 17.7).

Cases without checkpoint files will be ignored.

Command file syntax:

```
solver check file file_name
```

Example:

```
solver check file "checkpoints2"
solver check file "/user/bob/lsopt/checkpoints2"
```

13.10 Alternative point selection

Alternative point selection schemes can be specified to replace the main scheme in the first iteration. The main purpose is to provide an option to use linear *D*-optimality in the first iteration 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.

Command file syntax:

```
solver alternate experiment 1
 solver alternate order linear
 solver alternate experimental design point_selection_scheme
 solver alternate number experiment number_experimental_points
 solver alternate basis experiment basis_experiment
 solver alternate number basis experiments
                    number_basis_experimental_points
```

The defaults are as follows:

Attribute	*Default*
Order	`Linear` (only option available)
Experimental design	*D*-Optimal
Number of experiments	Number of experiments of main experimental design
Basis experimental design	type depends on number of variables (only *D*-optimal)
Number of basis experiments	depends on basis experiment type and number of experiments (only *D*-optimal)

Example:

```
Solver order FF
Solver experimental design space filling
Solver number experiments 5
Solver update doe
Solver alternate experiment 1
```

In the above example a linear surface based on *D*-optimal point selection will be used in the first iteration instead of a neural network based on Space Filling. The number of points is 5, the same as for the main experimental design. In the second iteration all the points created in the first and second iterations will be used to fit a neural network (because of `update doe`). The single additional line is typically all that is needed when using neural networks.

Example:

```
Solver order FF
Solver experimental design space filling
Solver number experiments 5
Solver alternate experiment 1
 Solver alternate experimental design dopt
 Solver alternate order linear
 Solver alternate basis experiments space_filling
 Solver alternate number basis experiments 100
```

13.11 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 case-specific, so must be added to all the case definitions.

Command file syntax:

`Solver experiment augment iteration` *`iteration_number`*

Example 1:

In the first analysis, the following sampling scheme was specified:

```
Solver experiment design dopt
Solver number experiment 5
Solver basis experiment 3toK
.
.
.
Iterate 1
```

By default, a single verification run is done in iteration 2.

After the first analysis, the user wants to restart, using a larger number of points

```
Solver experiment design dopt
Solver number experiment 10
Solver basis experiment 5toK
Solver experiment augment iteration 2
.
.
.
Iterate 3
```

Iterations 2 and 3 will then be conducted with 10 points each while iteration one will be left intact.

Example 2:

Starting with:

```
Solver experiment design dopt
Solver number experiment 5
.
.
.
Iterate 1
```

and *restarting* with:

```
Solver experiment design dopt
Solver number experiment 10
Solver experiment augment iteration 1
.
.
```

```
.
Iterate 3
```

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 entering "2" in the box for "Specifying Starting Iteration" in the Run panel. The restart will then generate a *new* starting point for iteration 2 and conduct 10 simulations altogether.

13.12 Repeatability of point selection

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 selection 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. The seed is of the type "`unsigned long`", so the value typically has to be between 0 and 4,294,967,295 (depending on the machine architecture). The syntax is as follows:

Command file syntax:

`Solver experiment seed` ***`integer_value`***

The default value is 0 (zero).

```
Solver experimental design lhd_generalized
Solver number experiments 30
Solver experiment seed 349177
```

13.13 Remarks: Point selection

1. The number of points specified in the "`solver number experiment` ***`num`***" command is reduced by the number already available in the `Experiments.PRE.`***`case_name`*** or `AnalysisResults.PRE.`***`case_name`*** files.
2. The files `Experiments` and `AnalysisResults` are synchronous, i.e. they will always have the same experiments after extraction of results. Both these files also mirror the result directories for a specific iteration.
3. Design points that replicate the starting point are omitted.

14. History and Response Results

This chapter describes the specification of the history or response results to be extracted from the solver database. The chapter focuses on the standard response interfaces for LS-DYNA.

14.1 Defining a response history (vector)

A response history can be defined by using the `history` command with an extraction, a mathematical expression or file import. The extraction of the result can be done using a standard LS-DYNA interface (see Section 14.4) or with a user-defined program.

Command file syntax:

```
history history_name string
history history_name expression math_expression
history history_name file string
```

The *string* is an interface definition (in double quotes), while the *math_expression* is a mathematical expression (in curly brackets).

Example 1:

```
history 'displacement_1' "BinoutHistory -res_type nodout -cmp x_displacement -
id 12789 -filter SAE -filter_freq 60"
history 'displacement_2' "BinoutHistory -res_type nodout -cmp x_displacement -
id 26993 -filter SAE -filter_freq 60"
history 'deformation' expression {displacement_2 - displacement_1}
response 'final_deform' expression {deformation(200)}
```

Example 2:

```
constant 'v0' 15.65
history 'bumper_velocity'    "BinoutHistory -res_type nodout -cmp x_velocity -
id 73579 -filter SAE -filter_freq 30"
history 'Apillar_velocity_1' "BinoutHistory -res_type nodout -cmp x_velocity -
id 41195 -filter SAE -filter_freq 30"
history 'Apillar_velocity_2' "BinoutHistory -res_type nodout -cmp x_velocity -
id 17251 -filter SAE -filter_freq 30"
history 'global_velocity'    "BinoutHistory -res_type glstat -cmp X_VEL "
history 'Apillar_velocity_average' expression {
```

```
                                (Apillar_velocity_1 + Apillar_velocity_2)/2}
$
response 'time_to_bumper_zero' expression {Lookup("bumper_velocity(t)",0)}
response 'vel_A_bumper_zero' expression {Apillar_velocity_average
(time_to_bumper_zero)}
response 'PULSE_1' expression {Integral
                                  ("Apillar_velocity_average(t)",
                                  0,
                                  time_to_bumper_zero)
                                  /time_to_bumper_zero}
response 'time_to_zero_velocity'expression {Lookup("global_velocity(t)",0)}
response 'velocity_final' expression
{Apillar_velocity_average(time_to_zero_velocity)}
response 'PULSE_2' expression    {Integral
                                  ("Apillar_velocity_average(t)"
                                  time_to_bumper_zero,
                                  time_to_zero_velocity)
                             /(time_to_zero_velocity - time_to_bumper_zero)}
```

Example 3:

```
constant 'Event_time' 200
$ Results from a physical experiment
history 'experiment_vel' file "expdata"
$ LS-DYNA results
history 'velocity' "DynaASCII nodout X_VEL 12667 TIMESTEP"
response 'RMS_error' expression {Integral("(experiment_vel-
velocity)**2",0,Event_time}
```

Example 4:

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

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

produces the `LsoptHistory` file.

```
history 'Force' "lsprepost c=../../get_force"
response 'Force1' expression {Force(.002)}
response 'Force2' expression {Force(.004)}
response 'Force3' expression {Force(.006)}
response 'Force4' expression {Force(.008)}
```

Note :

1. The rcforc history in Example 4 can be obtained more easily by direct extraction (see Section 14.5.1 and Appendix B)

Remarks:

1. Histories are used by response definitions (see Section 14.1.1) to define a response surface. They are therefore intermediate entities and cannot be used directly to define a response surface. Only `response` can define a response surface.
2. For LS-DYNA history definition and syntax, please refer to Section 14.4.

In LS-OPT*ui*, histories are defined in the Histories panel (Figure 14-1):

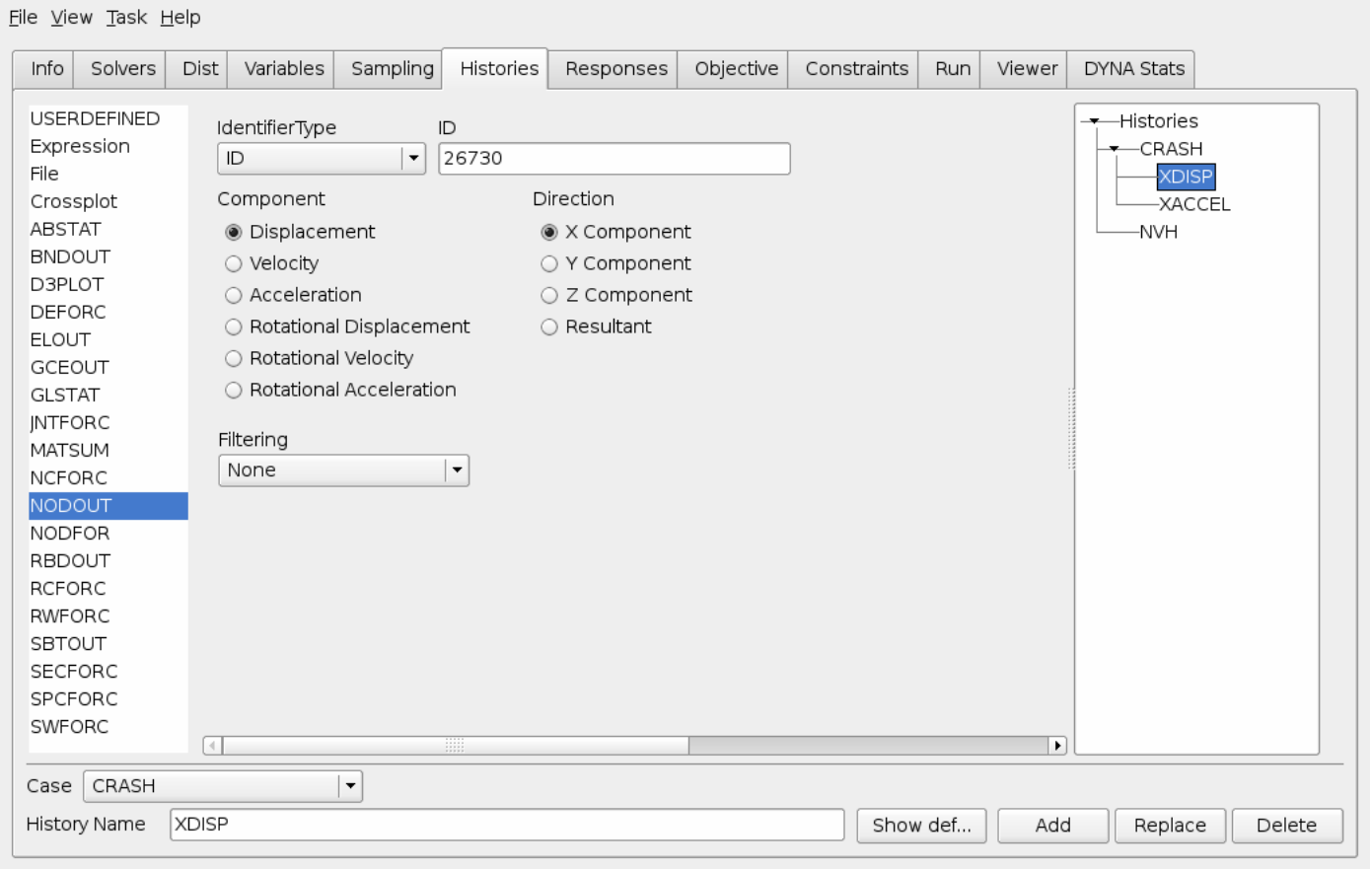


Figure 14-1: Histories panel in LS-OPT*ui*

14.1.1 Crossplot history

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

Expression syntax:

```
History 'curvename' {Crossplot (abscissa_history, ordinate_history,
    [numpoints, begin, end] )}
```

Argument name	Description	Symbol	LS-OPT Type	Default
abscissa history	History of abscissa	*f(t)*	Expression	-
ordinate history	History of ordinate	*g(t)*	Expression	-
numpoints	Number of points created in crossplot	*P*	Int	10,000
begin	Begin *t*-value	t_1	Float	Smallest *t*-value
end	End *t*- value	t_P	Float	Largest *t*-value

Table 14.1-1: Description of `Crossplot` arguments

Examples:

```
$ ------ CROSSPLOT CURVES ---------------------------------------------------
history 'Force_Disp_Dflt' expression {Crossplot("-Disp2", "Force2") }
history 'Force_Disp_to_Num' expression {Crossplot("-Disp2", "Force2", 2) }
history 'Force_Disp_to_Beg' expression {Crossplot("-Disp2", "Force2", 4, 0.002) }
history 'Force_Disp_to_End' expression {Crossplot("-Disp2", "Force2", 4, 0.002, End) }
```

14.1.2 History files

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.

Command file syntax:

```
history name file filename
```

Example:

```
History 'Test1' file "Test1.txt"
```

where `Test1.txt` contains:

```
Time    Displacement
1.2, 143.97
1.4   156.1
1.7 , 923.77
```

14.2 Defining a response (scalar)

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 14.4) or a user-defined interface.

Each extracted response is identified by a name and the command line for the program that extracts the results. The command line must be enclosed in double quotes. If scaling and/or offsetting of the response is required, the final response is computed as (the extracted response × *scale factor*) + *offset*. This operation can also be achieved with a simple mathematical expression.
A mathematical expression for a response is defined in curly brackets after the response name.

Command file syntax:

```
response response_name {scale_factor offset} string
response response_name expression math_expression
```

Example:

```
response 'Displacement_x' 25.4 0.0 "DynaASCII nodout 'r disp' 63 TIMESTEP 0.1"
response 'Force' "$HOME/ownbin/calculate force"
response 'Displacement_y' "calc constraint2"
response 'Disp' expression {Displacement_x + Displacement_y}
```

Remarks:

1. The first command will use a standard interface for the specified solver package. The standard interfaces for LS-DYNA are described in Section 14.4.

2. The middle two commands are used for a user-supplied interface program (see Section 14.10). The interface name must either be in the path or the full path name must be specified. Aliases are not allowed.

3. For the last command, the second argument `expression` is a reserved name.

14.3 Specifying the metamodel type

The metamodel type can be specified for an individual reponse.

Command file syntax:

```
response response_name
[linear|interaction|elliptic|quadratic|FF|kriging]
```

The default is the metamodel specified in Section 13.1. `FF` refers to the feedforward neural network approximation method (see Section 3.1).

Example:

```
response 'Displacement' kriging
```

In LS-OPT*ui*, responses are defined in the Responses panel (Figure 14-2):

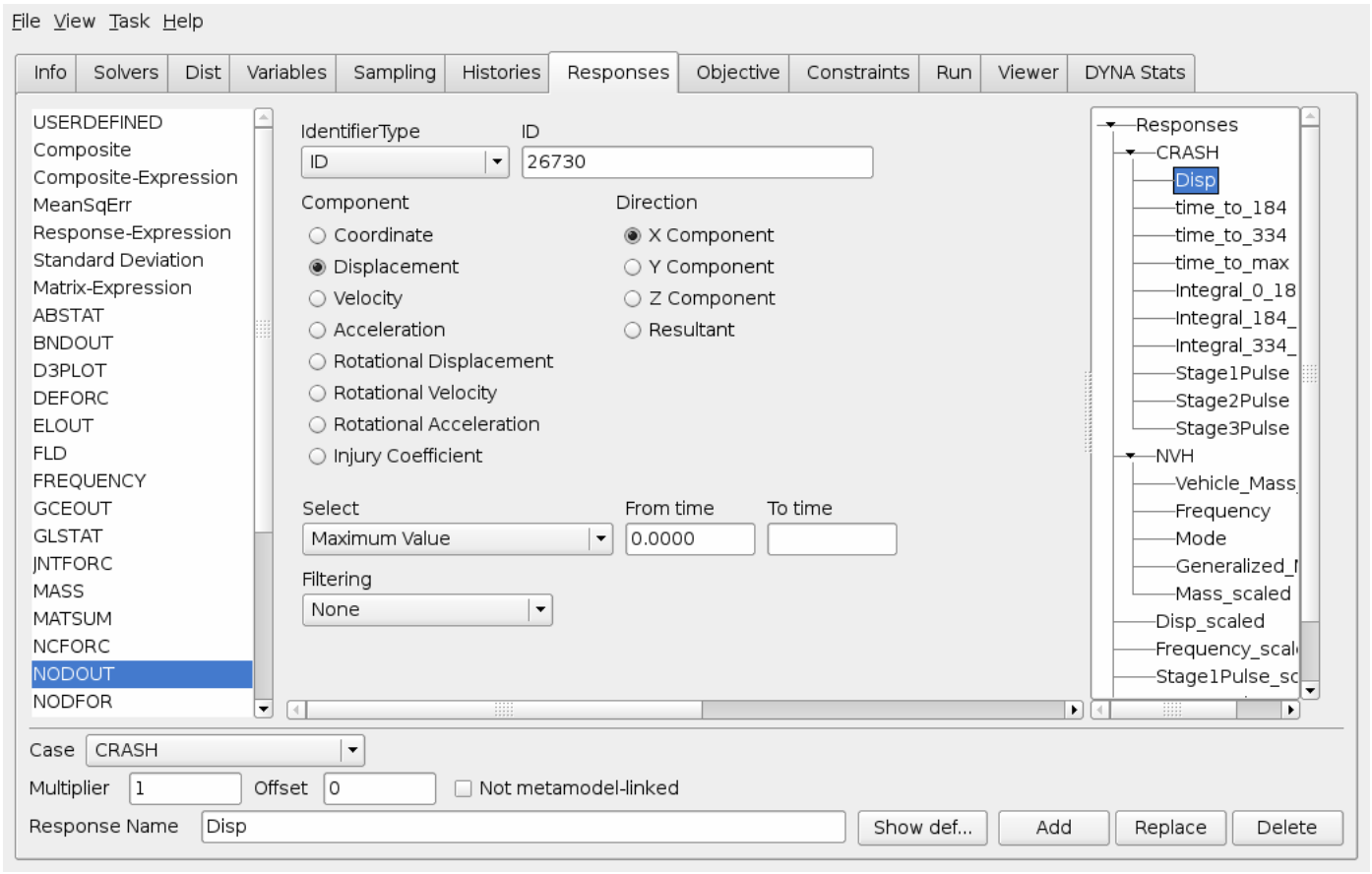


Figure 14-2: Reponses panel in LS-OPT*ui*

14.4 Extracting history and response quantities: LS-DYNA

In LS-OPT the general functionality for reading histories and responses from the simulation output is achieved through the `history` and `response` definitions (see Section 14.1 and Section 14.1.1 respectively). The syntax for the extraction commands for LS-DYNA responses and histories is identical, except for the selection attribute. The history function is included so that operations (such as subtracting two histories) can first be performed, after which a scalar (such as maximum over time) can be extracted from the resulting history.

There are two types of interfaces:

1. Standard LS-DYNA result interfaces. This interface provides access to the LS-DYNA binary databases (d3plot or Binout). The interface is an integral part of LS-OPT.

2. User specified interface programs. These can reside anywhere. The user specifies the full path.

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 14.9.

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

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 if requested by the user (`clean` file (see also Section 9.9)).
- 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.

For more specialized responses the Perl programs provided can be used as templates for the development of own routines.

All the utilities can be specified through the command:

```
response response_name {scale_factor offset } command_line.
```

or

```
history history_name command_line.
```

14.5 LS-DYNA Binout results

From Version 970 of LS-DYNA the ASCII output can be written to a binary file: the `Binout` file.

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

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

14.5.1 Binout histories

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.

Command file syntax:

```
BinoutHistory -res_type res_type {-sub sub} -cmp component {-invariant
invariant -id(-name) id(NAME) -pos position -side side}
```

Item	Description	Default	Remarks
res_type	Result type name	-	1
sub	Result subdirectory	-	1
cmp	Component of result	-	2
invariant	Invariant of results. Only MAGNITUDE is currently available.	-	3
id	ID number of entity	-	
name	Description (heading) of entity	-	4
pos	Through thickness shell position at which results are computed.	1	5
side	Interface side for RCFORC data. MASTER or SLAVE.	SLAVE	

Example:

```
history 'ELOUT1' "BinoutHistory -res_type Elout -sub shell -cmp sig_xx
-id 1 -pos 1"
history 'invarHis' "BinoutHistory -res_type nodout -cmp displacement
-invariant MAGNITUDE -name RAIL15"
```

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 A: LS-DYNA Binout Result 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`.

Averaging, filtering, and slicing Binout histories

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

Command file syntax:

```
BinoutHistory {history_options} {-filter filter_type
-filter_freq filter_freq -units units -ave_points ave_points
-start_time start_time -end_time end_time }
```

Item	Description	Default
history_options	All available history options	-
filter_type	Type of filter to use: `SAE` or `BUTT`	-
filter_freq	Filter frequency	60 cycles / time unit
units	`S`=seconds `MS`=milliseconds	`S`
ave_points	Number of points to average	-
start_time	Start time of history interval to extract using slicing	0
end_time	End time of history interval to extract using slicing	t_{max}

Example:

```
history 'ELOUT12' "BinoutHistory -res_type Elout -sub shell -cmp sig_xx
-name RAIL15 -pos 2 -filter SAE -start_time 0.02 -end_time 0.04"
history 'nodHist432acc_AVE' "BinoutHistory -res_type nodout
-cmp x_acceleration -id 432 -ave_points 5"
```

14.5.2 Binout 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.

Command file syntax:

```
BinoutResponse {history_options} -select selection
```

Item	Description	Default	Remarks
history_options	All available history options including averaging, filtering, and slicing.	-	

selection	MAX|MIN|AVE|TIME	TIME	1

Example:

```
response 'eTime'  "BinoutResponse -res_type glstat -cmp kinetic_energy
-select TIME -end_time 0.015"
$
response 'nodeMax' "BinoutResponse -res_type nodout -cmp x_acceleration
-id 432 -select MAX -filter SAE -filter_freq 10"
```

Remarks:

1. The maximum, minimum, average, or value at a specific time must be selected. If *selection* 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.

Binout injury criteria

Injury criteria such as HIC can be specified as the result component. The acceleration components will be extracted, the magnitude computed, and the injury criteria computed from the acceleration magnitude history.

Command file syntax:

```
BinoutResponse {history_options}  -cmp cmp {-gravity gravity
-units units}
```

Item	Description	Default
history_options	All available history options including filtering and slicing.	-
cmp	HIC15, HIC36, or CSI	-
gravity	Gravitational acceleration	9.81
units	S=seconds MS=milliseconds	S

Example:

```
response 'HIC_ms' 1 0 "BinoutResponse -res_type Nodout -cmp HIC15
-gravity 9810. -units MS -name RAIL15"
```

14.6 LS-DYNA D3Plot results

The D3Plot interface is related to the Binout interface. The D3Plot commands differ from the Binout commands in that a response or history can be collected over a whole part. For example, the maximum stress in a part or over the whole model.

The available results types and components are listed in Appendix A.

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

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

14.6.1 D3Plot histories

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.

Command file syntax:

```
D3PlotHistory -res_type res_type {-sub sub} -cmp component {-id id
-pos position -pids part_ids -loc ELEMENT|NODE -select selection
-coord x y z -setid setid -tref ref_state}
```

Item	Description	Default	Remarks
res_type	Result type name	-	1
cmp	Component of result	-	1
id	ID number of entity	-	2
pos	Through thickness shell position at which results are computed.	1	
pids	One or more part ids.	-	3
loc	Locations in model. ELEMENT or NODE.	-	4
selection	MAX|MIN|AVE	MAX	5
coord	Coordinate of a point for finding nearest element	-	6
tref	Time of reference state for finding nearest element	0.0	6
setid	ID of *SET_SOLID_GENERAL in LS-DYNA keyword file	-	6

Example:

```
history 'ELOUT1' "D3PlotHistory -res_type Elout -sub shell -cmp sig_xx
-id 1 -pos 1"
```

```
history 'invarHis' "D3PlotHistory -res_type nodout -cmp displacement
-invariant MAGNITUDE -id 432"
history 'd3ploth4' "D3PlotHistory -res_type ndv -cmp x_displacement -
pids 2 3 -select MAX"
```

Remarks:

1. The result types and components are similar to what is used in LS-PREPOST. The result types and component names are listed in Appendix A:LS-DYNA D3Plot Result Components.
2. For histories, the *-id* option is mutually exclusive with the *–select* 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. For D3Plot histories, the *-select* option is mutually exclusive with the *–id* option.
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).

Slicing D3Plot histories

Slicing of D3Plot histories is possible. Averaging and filtering are not available for D3Plot results.

Command file syntax:

`D3PlotHistory {`*`history_options`*`} {-start_time `*`start_time`*` -end_time `*`end_time`*` }`

Item	Description	Default
history_options	All available history options	-
start_time	Start time of history interval to extract using slicing	0
end_time	End time of history interval to extract using slicing	t_{max}

Example:

```
history 'ELOUT12' "D3PlotHistory -res_type stress -cmp xx_stress
-id 1 -pos 2 -start_time 0.02 -end_time 0.04"
```

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.

Command file syntax:

```
D3PlotHistory {history_options} {-fld_t fld_t -fld_n fld_n -fld_curve fld_curve}
```

Item	Description	Default
history_options	All available history options	-
fld_t	Fld curve t coefficient	-
fld_n	Fld curve t coefficient	-
fld_curve	ID of curve in the LS-DYNA input deck	-

Example:

```
history 'ELOUT12' "D3PlotHistory -res_type stress -cmp xx_stress
-id 1 -pos 2 -start_time 0.02 -end_time 0.04"
```

14.6.2 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.

Command file syntax:

```
D3PlotResponse {history_options} -select selection
```

Item	Description	Default	Remarks
history_options	All available history options	-	
selection	MAX|MIN|AVE|TIME	TIME	1

Example:

```
Response 'nodeMax' "D3PlotResponse -res_type ndv -cmp x_displacement -id
432 -select MAX"
```

Remarks:

1. The maximum, minimum, average, or value at a specific time must be selected. If *selection* 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 part ids as well, then the maximum, minimum, or average value will be selected for the part, followed by the selection of the maximum, minimum, or average over time.

14.7 Mass

Command file syntax:

`DynaMass` *`p1 p2 p3 ... pn mass_attribute`*

Table 14.7-1: Mass item description

Item	Description
p1 ... pn	Part numbers of the model. Omission implies the entire model.
Mass_attribute	Type of mass quantity (see table below).

Table 14.7-2: Mass attribute description

Attribute	Description
`MASS`	Mass
`I11`	Principal inertias
`I22`	
`I33`	
`IXX`	Components of inertia tensor
`IXY`	
`IXZ`	
`IYX`	
`IYY`	
`IYZ`	
`IZX`	
`IZY`	
`IZZ`	
`X_COORD`	*x*-coordinate of mass center
`Y_COORD`	*y*-coordinate of mass center
`Z_COORD`	*z*-coordinate of mass center

Example:

```
$ Specify the mass of material number 13, 14 and 16 as
$ the response 'Component_mass'.
response 'Component_mass' "DynaMass 3 13 14 16 Mass"
$ Specify the total principal inertial moment about the x-axis.
response 'Inertia' "DynaMass Ixx"
```

Remarks:

1. The output file d3hsp must be produced by LS-DYNA.
2. 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.

14.8 Frequency of given modal shape number

Command file syntax:

`DynaFreq` *`mode_original modal_attribute`*

Table 14.8-1: Frequency item description

Item	Description
mode_original	The number (sequence) of the baseline modal shape to be tracked.
modal_attribute	Type of modal quantity. (See table below).

Table 14.8-2: Frequency attribute description

Attribute	Description
FREQ	Frequency of current mode corresponding in modal shape to baseline mode specified.
NUMBER	Number of current mode corresponding in modal shape to baseline mode specified.
GENMASS	$\max_j \left\{ \left(M_0^{\frac{1}{2}} \phi_0 \right)^T \left(M_j^{\frac{1}{2}} \phi_j \right) \right\}$

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 performs a scalar product between the baseline modal shape (mass-orthogonalized eigenvector) and each mode shape of the current design. The maximum scalar product indicates the mode most similar in shape to the original mode selected. To adjust for the mass orthogonalization, the maximum scalar product is found in the following manner:

$$\max_j \left\{ \left(M_0^{\frac{1}{2}} \phi_0 \right)^T \left(M_j^{\frac{1}{2}} \phi_j \right) \right\} \qquad (14.8\text{-}1)$$

where $\boldsymbol{M}$ is the mass matrix (excluding all rigid bodies), ϕ is the mass-orthogonalized eigenvector and the subscript 0 denotes the baseline mode. This product can be extracted with the GENMASS attribute (see Table 14.8-2). Rigid body inertia and coupling will be incorporated in a later version.

Example:

```
$ Obtain the frequency of the current mode corresponding to the
$ baseline mode shape number 15 as the response 'Frequency'.
response 'Frequency' "DynaFreq 15 FREQ"
$ Obtain the number (sequence) of the current mode corresponding to
$ the baseline mode shape number 15 as the response 'Number of mode'.
response 'Modal number' "DynaFreq 15 NUMBER"
```

Remarks:

1. The user must identify which baseline mode is of interest by viewing the baseline `d3eigv` file in LS-PrePost. The baseline mode number should be noted.

2. The optimization run can now be started with the correct `DynaFreq` command (or select the Baseline Mode Number in the GUI).

3. Additional files are generated by LS-DYNA and placed in the run directories to perform the scalar product and extract the modal frequency and number.

4. *mode_original* cannot exceed 999.

14.9 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 *D3plotResponse* commands are an alternative.

14.9.1 Thickness and thickness reduction

Either thickness or thickness reduction can be specified as follows.

Command file syntax:

`DynaThick` `[THICKNESS|REDUCTION]` ***p1 p2 ... pm*** `[MIN|MAX|AVE]`

Table 14.9-1: DynaThick item description

Item	Description
`THICKNESS`	Final thickness of part
`REDUCTION`	A percentage thickness reduction of the part
p1…pn	The parts as defined in LS-DYNA. If they are omitted, all the parts are used.
`MIN|MAX|AVE`	Minimum, maximum or average computed over all the elements of the selected parts

Example:

```
Response 'Thickness 1' "DynaThick THICK 1 2 MAXIMUM"
Response 'Thickness 1' "DynaThick REDU 1 2 MINIMUM"
```

14.9.2 FLD constraint

The FLD constraint is shown in Figure 14-3.

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:

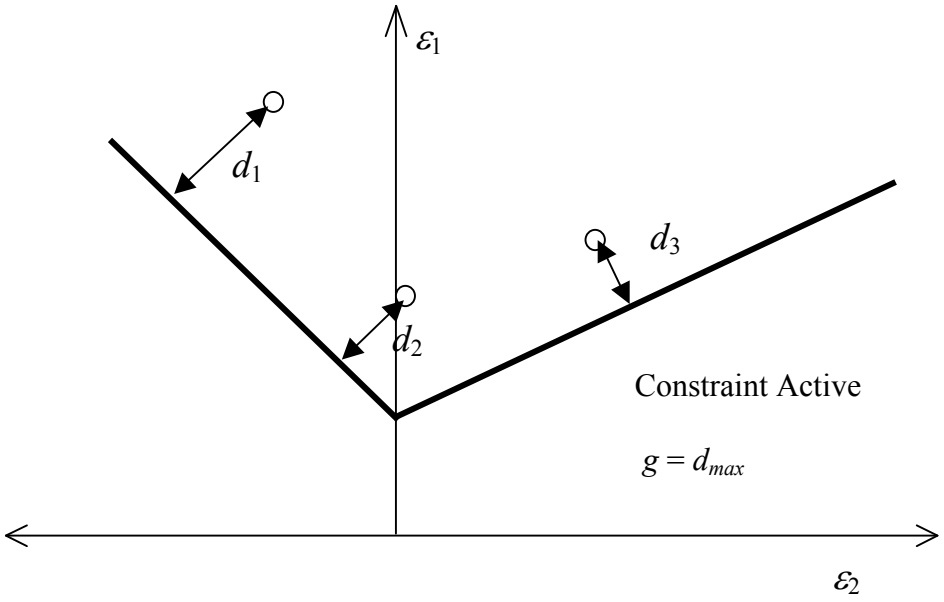
  $$g = d_{max}$$

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

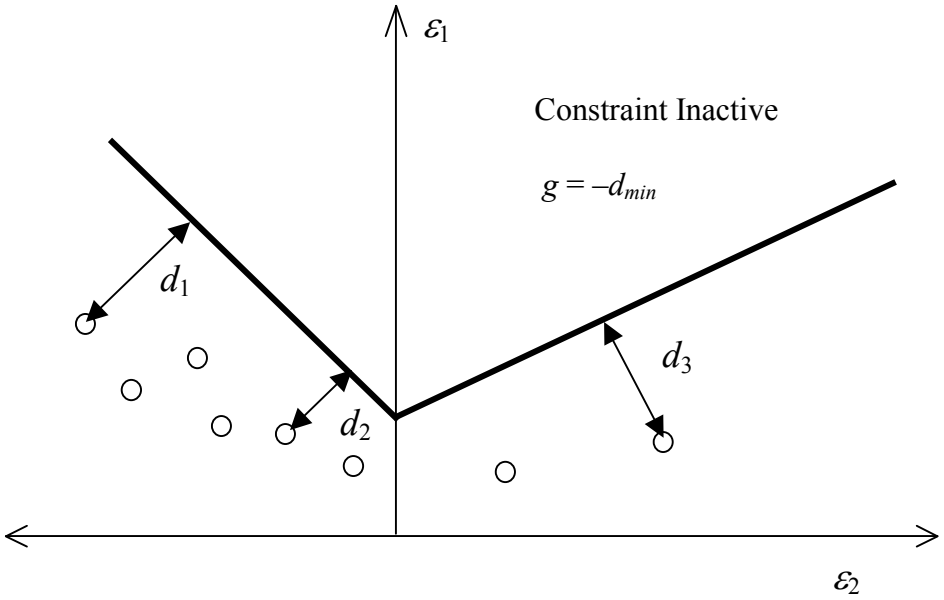
- All the values of the strain points are located below the FLD curve. In this case the constraint is computed as:

  $$g = -d_{min}$$

  with d_{min} the minimum smallest distance of any strain value to the FLD curve (Figure 14-3).



a) FLD Constraint active



b) FLD Constraint inactive

Figure 14-3: FLD curve – constraint definition

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

Bilinear FLD constraint

The values of both the principle upper and lower surface in-plane strains are used for the FLD constraint.

Command file syntax:

```
DynaFLD p1 p2 ... pn intercept negative_slope positive_slope
```

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

Table 14.9-2: DynaFLD item description

Item	Description
p1...pn	Part numbers of the model. Omission implies the entire model.
intercept	The FLD curve value at $\varepsilon_2 = 0$
negative_slope	The absolute value of the slope of the FLD curve value at $\varepsilon_2 < 0$
positive_slope	The absolute value of the slope of the FLD curve value at $\varepsilon_2 > 0$

Example:

```
$ Specify the FLD Constraint to be used
Response 'FLD' "DynaFLD 1 2 3 0.25 1.833 0.5"
```

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:

1. A piece-wise linear curve is defined by specifying a list of interconnected points. The abscissae (ε_2) of consecutive points must increase (or an error termination will occur). Duplicated points are therefore not allowed.

2. The curve is extrapolated infinitely in both the negative and positive directions of ε_2. The first and last segments are used for this purpose.

3. The computation of the constraint value is the same as shown in (Figure 14-3).

Command file syntax:

`DynaFLDg [LOWER|CENTER|UPPER] p1 p2 ... pn load_curve_id`

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

Table 14.9-3: DynaFLDg item description

Item	Description
LOWER	Lower surface of the sheet
UPPER	Upper surface of the sheet
CENTER	Middle surface of the sheet
p1...pn	Part numbers of the model. Omission implies the entire model.
load_curve_id	Identification number of a load curve in the LS-DYNA input file.

	The `*DEFINE_CURVE` keyword must be used. Refer to the LS-DYNA User's Manual for an explanation of this keyword.

Example:

```
$ Specify the general FLD Constraint to be used
Response 'FLDL' "DynaFLDg LOWER 1 2 3 23"
Response 'FLDU' "DynaFLDg UPPER 1 2 3 23"
Response 'FLDC' "DynaFLDg CENTER 23"
```

For all three specifications load curve 23 is used. In the first two specifications, only parts 1, 2 and 3 are considered.

Remarks:

1. The interface program produces an output file `FLD_curve` which contains the ε_1 and ε_2 values in the first and second columns respectively. Since the program first looks for this file, it can be specified in lieu of the keyword specification. *The user should take care to remove an old version of the* `FLD_curve` *if the curve specification is changed in the keyword input file.* If a structured input file is used for LS-DYNA input data, `FLD_curve` *must* be created by the user.

2. The scale factor and offset values feature of the `*DEFINE_CURVE` keyword are not utilized.

14.9.3 Principal stress

Any of the principal stresses or the mean can be computed. The values are nodal stresses.

Command file syntax:

`DynaPStress [S1|S2|S3|MEAN]` ***p1 p2 ... pn*** `[MIN|MAX|AVE]`

Table 14.9-4: DynaPStress item description

Item	Description
`S1, S2, S3`	$\sigma_1, \sigma_2, \sigma_3$
`MEAN`	$(\sigma_1 + \sigma_2 + \sigma_3)/3$
p1 ... pn	Part numbers of the model. Omission implies the entire model.
`MIN|MAX|AVE`	Minimum, maximum or average computed over all the elements of the selected parts

Example:

```
Response 'Stress 1' "DynaPStress MEAN 14 17 MAX"
```

14.10 Userdefined interface for extracting results

The user may provide an own extraction routine to output a single floating-point number to standard output.

Examples of the output statement in such a program are:

- The C language:

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

  or

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

- The FORTRAN language:

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

- The Perl script language:

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

The string "`N o r m a l`" must be written to the standard error file identifier (`stderr` in C) to signify a normal termination. (See Section 22.1 for an example).

The command to use a user-defined program to extract a response is:

Command file syntax:

`response` *`response_name`* `{` *`scale_factor offset`* `}` *`command_line`*

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.

   The relevant response definition command must therefore be as follows:
   ```
   response 'Force' "$HOME/own/bin/ExtractForce"
   ```

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

```
response 'Acc' "$LSOPT/perl $LSOPT/DynaFLD2 0.5 0.25 1.833"
```

Remark:

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

14.11 Responses without metamodels

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. The selection is simply made in a check box in the "Responses" panel (labeled "Not metamodel-linked"). Except for the metamodel linking, "Results" are identical to "Responses" and can be defined using a standard LS-DYNA interface, a mathematical expression or a command for a user-defined program.

Command file syntax:

```
result name string
result name math_expression
result name command_line
```

Remark:

14.12 Matrix operations

1. "Results" cannot be included directly in composites, since a composite relies on interpolation from a metamodel.

Matrix operations can be performed by initializing a matrix, performing multiple matrix operations, and extracting components of the matrix as response functions or results.

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

The component of a matrix is extracted using the format *A*.a*ij* (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 $\boldsymbol{A} - \boldsymbol{I}$ (where $\boldsymbol{I}$ is the unit matrix) is coded as `A-1`.

Command file syntax:

```
matrix name math_expression
```

Examples:

In the following example the user constructs a matrix from scalar results, performs matrix operations and uses the final matrix components in an optimization run:

```
Constant 'X2' 0.0
Constant 'Y2'  0.0
Constant 'Z2' -1.0
Constant 'X3'  0.0
Constant 'Y3'  0.0
```

```
Constant 'Z3'  8.0
$
$                       Extract results
$
  result 'Fd11_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#11 -select TIME -end_time 0.04"
  result 'Fd12_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#14 -select TIME -end_time 0.04"
  result 'Fd13_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#17 -select TIME -end_time 0.04"
  result 'Fd21_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#12 -select TIME -end_time 0.04"
  result 'Fd22_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#15 -select TIME -end_time 0.04"
  result 'Fd23_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#18 -select TIME -end_time 0.04"
  result 'Fd31_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#13 -select TIME -end_time 0.04"
  result 'Fd32_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#16 -select TIME -end_time 0.04"
  result 'Fd33_2' "D3PlotResponse -setid 10 -tref 0.04 -coord 0 -1.858 1.858
-res_type misc -cmp history_var#19 -select TIME -end_time 0.04"
$
$                       Matrix expressions
$
$           1. Initialization
$
  matrix 'Fd_2'
   {Matrix3x3Init(Fd11_2,Fd12_2,Fd13_2,Fd21_2,Fd22_2,Fd23_2,Fd31_2,Fd32_2,Fd33_2)}
  matrix 'Fs_2'
   {Matrix3x3Init(Fs11_2,Fs12_2,Fs13_2,Fs21_2,Fs22_2,Fs23_2,Fs31_2,Fs32_2,Fs33_2)}
  matrix 'R_2' {Rotate(0, -1.858, 1.858,  X2,Y2,Z2,  X3,Y3,Z3)}
$
$           2. Matrix operations
$
$              Updated deformation gradient Fs
  matrix 'FSD_2' {Fs_2 * inv (Fd_2)}
$              Updated Lagrange strain using Fs and Fd
  matrix 'epsGlobal_2' {.5 * ( tr ( FSD_2 ) * FSD_2  -  1  )}
$              Tensor transformation to local coordinates
  matrix 'epsCyl_2' {tr(R_2) * epsGlobal_2 * R_2}
$
$           3. Extract matrix components as response surfaces
$
  response 'Ell_2' expression {epsCyl_2.a11}
  response 'Ecc_2' expression {epsCyl_2.a33}
  response 'Elc_2' expression {epsCyl_2.a13}
  response 'Elr_2' expression {epsCyl_2.a12}
  response 'Ecr_2' expression {epsCyl_2.a32}
```

14.12.1 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).

14.12.2 Creating a rotation matrix using 3 specified points

The command 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 $\boldsymbol{X}$ direction.
- $\boldsymbol{Z} = v_{23} \times v_{21}$
- $\boldsymbol{Y} = \boldsymbol{Z} \times \boldsymbol{X}$

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

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

where $\boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}$.

15. Composite Functions

Composite functions can be used to combine response surfaces and variables as well as other composites. The objectives and constraints can then be constructed using the composite functions.

15.1 Introduction

15.1.1 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.

There are three types of composites:

15.2 Expression composite

15.2.1 General expressions

A general expression can be specified for a composite. The composite can therefore consist of constants, variables, dependent variables, responses and other composites (see Appendix D).

15.2.2 Special expressions

There is one special function for composites namely `MeanSqErr` (see Section 15.6).

15.3 Standard composite

15.3.1 Targeted composite (square root of MSE)

This is a special composite in which a target is specified for each response or variable. The composite is formulated as the 'distance' to the target using a Euclidean norm formulation. The components can be weighted and normalized.

$$\mathcal{F} = \sqrt{\sum_{j=1}^{m} W_j \left[\frac{f_j(\boldsymbol{x}) - F_j}{\sigma_j} \right]^2 + \sum_{i=1}^{n} \omega_i \left[\frac{x_i - X_i}{\chi_i} \right]^2} \qquad (15.3\text{-}1)$$

where σ and χ are scale factors and W and ω are weight factors. These are typically used to formulate a multi-objective optimization problem in which $\mathcal{F}$ is the distance to the target values of design and response variables.

A suitable application is parameter identification. In this application, the target values F_j are the experimental results that have to be reproduced by a numerical model as accurately as possible. The scale factors σ_j and χ_i are used to normalize the responses. The second component, which uses the variables can be used to regularize the parameter identification problem. Only independent variables can be included. See Figure 15-1 for an example of a targeted composite response definition.

In the GUI this type is now selected as the "Root MSE" type.

15.3.2 Mean Squared Error composite

This special 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-based `MeanSqErr` composite in Section 15.6).

15.3.3 Weighted composite

Weighted response functions and independent variables are summed in this standard composite. Each function component or variable is scaled and weighted.

$$\mathcal{F} = \sum_{j=1}^{m} W_j \frac{f_j(\boldsymbol{x})}{\sigma_j} + \sum_{i=1}^{n} \omega_i \frac{x_i}{\chi_i} \qquad (15.3\text{-}2)$$

These are typically used to construct objectives or constraints in which the responses and variables appear in linear combination.

The expression composite is a simple alternative to the weighted composite.

Remarks:

1. An expression composite can be a function of any other composite.

2. An objective definition involving more than one response or variable requires the use of a composite function.

3. In addition to specifying more than one function per objective, multiple objectives can be defined (see Section 16.2).

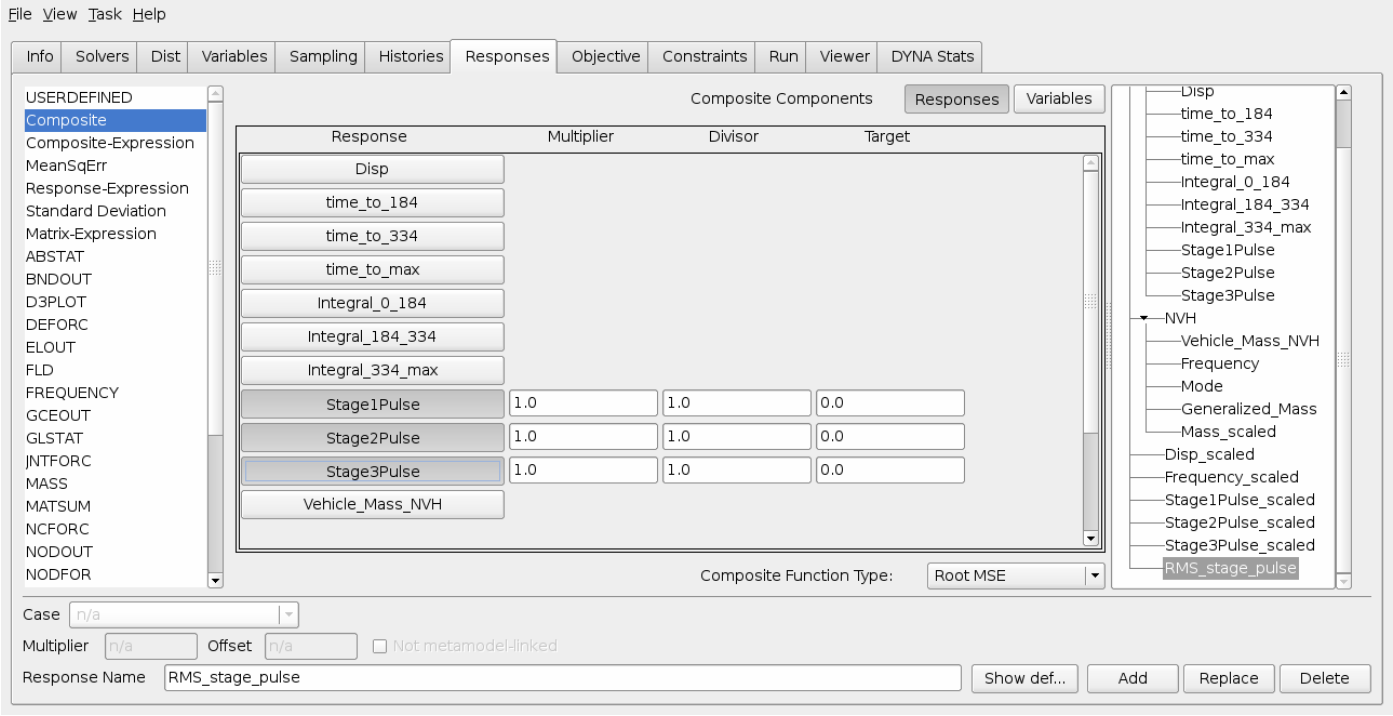


Figure 15-1: Definition of targeted (Root MSE) composite response in LS-OPT*ui*

15.4 Defining the composite function

This command identifies the composite function. The type of composite is specified as `weighted`, `targeted` or `expression`. The expression composite type does not have to be declared and can simply be stated as an expression.

Command file syntax:

```
composite composite_name type [standardMSE|targeted|weighted]
```

Example:

```
composite 'Damage' type targeted
composite 'Acceleration' type weighted
```

The expression composite is defined as follows:

Command file syntax:

```
composite composite_name math_expression
```

The *math_expression* is a mathematical expression given in curly brackets (see Appendix D).

The number of composite functions to be employed must be specified in the problem description.

15.5 Assigning design variable or response components to the composite

Command file syntax:

```
composite name response response_name value <1> { scale scale_factor <1> }
composite name variable variable_name value { scale scale_factor <1> }
```

The *value* is the target value for type: `targeted` and the weight value for the type: `weighted`. The *scale_factor* is a divisor.

Example:

```
composite 'damage' type targeted
composite 'damage' response 'intrusion_3' 20. scale 30.
composite 'damage' response 'intrusion_4' -35. scale 25.
```

for the composite function $\mathcal{F}_{damage} = \sqrt{\left(\frac{f_3 - 20}{30}\right)^2 + \left(\frac{f_4 - 35}{25}\right)^2}$.

The equivalent code using the expression composite is:

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

Example:

```
$----- x10 > x9 --------------------------
composite 'C9' type weighted
composite 'C9' variable 'x_9'      -1.
composite 'C9' variable 'x_10'     1.
constraint 'C9'
Lower bound constraint 'C9'        0.
```

for the composite function which defines the inequality $x_{10} > x_9$.

The equivalent code using the expression composite is:

```
$----- x10 > x9 --------------------------
composite 'C9' {x_10 - x_9}
constraint 'C9'
Lower bound constraint 'C9'        0.
```

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

If weights are required for the targeted function, an additional command may be given.

Command file syntax:

`weight` *weight value* <1>

Example:

```
composite 'damage' type targeted
composite 'damage' response 'intrusion_3'    20.
weight 1.5
composite 'damage' response 'intrusion_4'    -35.
```

is used to specify $\mathcal{F}_{damage} = \sqrt{1.5(f_3 - 20)^2 + (f_4 - 35)^2}$.
The weight applies to the last specified composite and response.

15.6 Mean Squared Error

A special function `MeanSqErr` is provided to compute the Mean Squared Error:

$$\varepsilon = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\boldsymbol{x}) - G_p}{s_p} \right)^2 = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{e_p(\boldsymbol{x})}{s_p} \right)^2 \qquad (15.6\text{-}1)$$

It is constructed so that G_p , $p=1,...,P$ are the values on the target curve G and $f_p(\boldsymbol{x})$ the corresponding components of the computed curve f. $f_p(\boldsymbol{x})$ are represented internally by response surface values. $\boldsymbol{x}$ is the design vector. By using the default values, the user should obtain a dimensionless error ε of the order of unity. See Section 5.3.1 for more detail.

Expression syntax:

```
MeanSqErr (target_curve, computed_curve,
     [num_regression_points, start_point, end_point,
     weight_type, scale_type,
     weight_value, scale_value,
     weight_curve, scale_curve])
```

Table 15.6-1: `MeanSqErr` arguments. Arguments in bold are obligatory.

Argument name	Description	Symbol	LS-OPT Type	Default
target_curve	Target Curve name	$G(z)$	History	-
computed_curve	Computed curve name	$f(x,z)$	History	-
num_regression_ points	Number of regression points	P	Int	If $P < 2$ or not specified: use number of points in target curve between lower limit and upper limit
lower_limit	Lower limit on z	z_L	Float	z-Location of first target point
upper_limit	Upper limit on z	z_U	Float	z-Location of last target point
weight_ type	Weighting type	-	Reserved option name: `WEIGHTVALUE` `PROPWEIGHT`, `FILEWEIGHT`	`WEIGHTVALUE` (Value=1.0)
scale_type	Scaling type	-	Reserved option name: `SCALEVALUE`, `PROPSCALE` `MAXISCALE`, `FILESCALE`	`MAXISCALE`
weight_value	Weight value	W	Float	1
scale_value	Scale value	s	Float	1
weight_curve	Weights as a function of z	$W(z)$	History	`Weight.`***`compositename`***
scale_curve	Scale factors as a function of z	$s(z)$	History	`Scale.`***`compositename`***

Table 15.6-2: Options for `MeanSqErr` arguments

Syntax	Explanation
`WEIGHTVALUE`	W_i = *value.* Default = 1.0

PROPWEIGHT	Use a different weight for each curve point p, proportional to the value of $\|G_p\|$. This method emphasizes the large absolute values of the response. The weights are normalized with respect to max $\|G_p\|$
FILEWEIGHT	Interpolate the weight from an x-y file: weight vs. z
SCALEVALUE	s_i = *value*. Default = 1.0
MAXISCALE	max $\|G_p\|$.
PROPSCALE	Use a different scale factor for each curve point, namely $\|G_p\|$.
FILESCALE	Interpolate the scale factor from an x-y file: scale vs. z

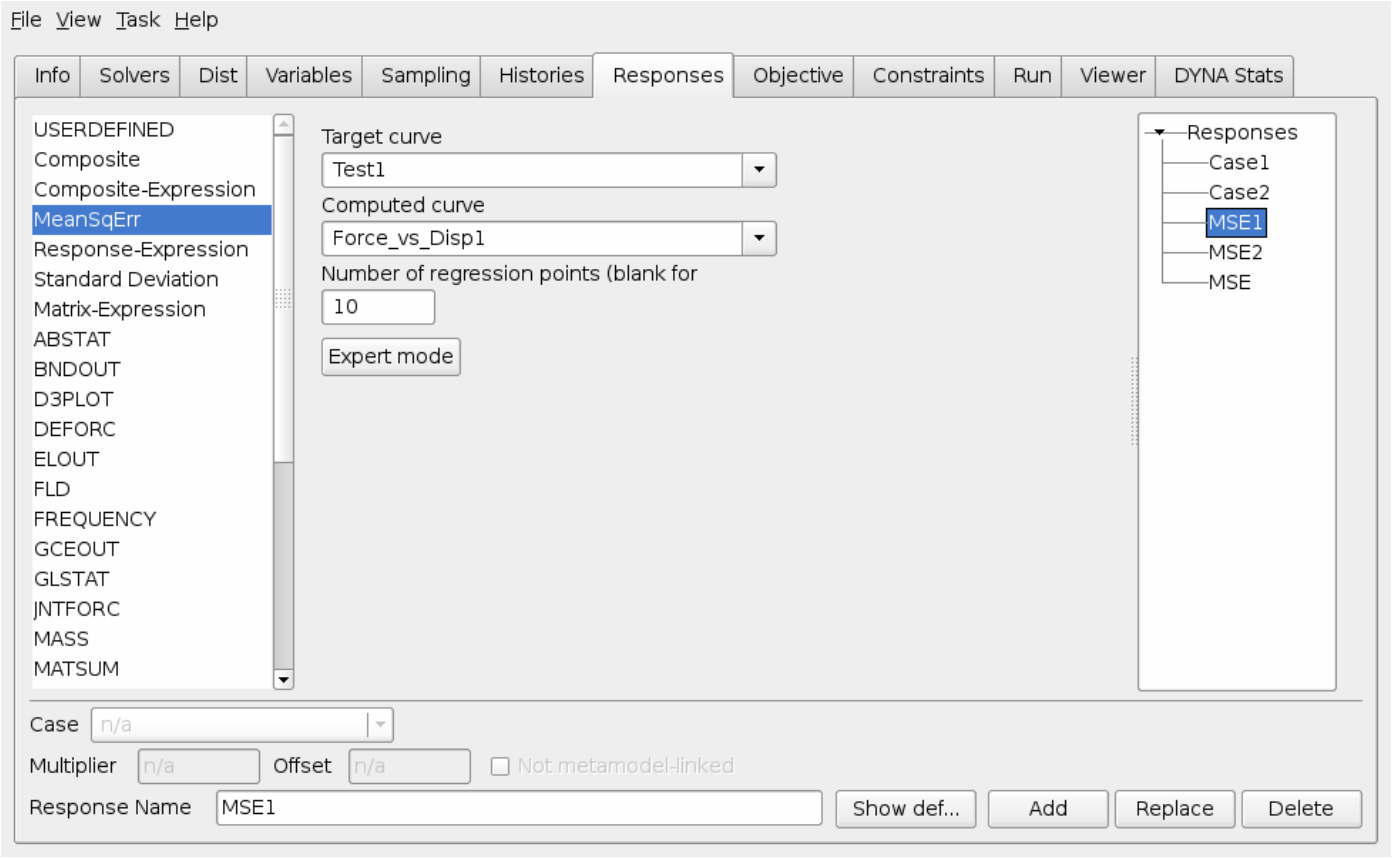


Figure 15-2: Responses panel showing a MeanSqErr selection

Note:

1. The MeanSqErr function can only be used as a composite.
2. 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.
3. If *num_regression_points* is unspecified, P equals the number of target points bounded by *lower_limit* and *upper_limit*.
4. The weight curve and scale curve must be predefined histories (see Section 14.1) 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.

5. The MeanSqErr composite makes use of response surfaces to avoid the nonlinearity (quadratic nature) of the squared error functional. Thus if the response curve $f(\boldsymbol{x})$ is linear in terms of the design variables $\boldsymbol{x}$, the composite function will be exactly represented.
6. Empty or underscore (_) arguments will generate default values.
7. The option names in Table 15.6-2 are reserved names and cannot be used as variable names.
8. MeanSqErr composites can be added together to make a larger MSE composite (e.g. for multiple test cases).

The simplest case, and probably the one used most frequently, is where the user simply defines only the target curve and corresponding computed curve (therefore only the first two arguments). In this case all the points in the target curve are taken as regression points (provided they have corresponding computed points). The simplest target curve that can be defined has only one point.

Examples:

```
$ ------ CONSTANTS ----------------------------------------------------------
 Constant 'Begin' 0.002
 Constant 'End' 0.008
 Constant 'numpoints' 4
$ ------ HISTORIES FROM BINOUT ----------------------------------------------
 history 'Force1' "BinoutHistory -res_type rcforc -cmp z_force -id 1 -side SLAVE"
 history 'Force2' "BinoutHistory -res_type rcforc -cmp z_force -id 1 -side SLAVE"
 history 'Disp2'  "BinoutHistory -res_type nodout -cmp z_displacement -id 288"
$ ------ HISTORIES FROM CROSSPLOTS ------------------------------------------
 history 'Force_Disp_Dflt' expression { Crossplot ("-Disp2", "Force2") }
 history 'Force_Disp_to_Num' expression { Crossplot ("-Disp2", "Force2", 2) }
 history 'Force_Disp_to_Beg' expression { Crossplot ("-Disp2", "Force2", 4, 0.002) }
 history 'Force_Disp_to_End' expression { Crossplot ("-Disp2", "Force2", 4, 0.002,
End) }
$ ------ HISTORIES FROM FILES -----------------------------------------------
 history 'Test1' file "Test1"
 history 'Test2' file "Test2"
 history 'Test3' file "Test3"
 history 'Weight.Weight_Scale_Curves' file "Weight.Weight_Scale_Curves"
 history 'Scale.Weight_Scale_Curves' file "Scale.Weight_Scale_Curves"
 history 'Scale.Wt_Scale_Curves2' file "Scale.Weight_Scale_Curves2"
 history 'Weight_1' file "Weight_1"
 history 'Scale_1' file "Scale_1"
 history 'UnitWeight' file "UnitWeight"
$ ------ COMPOSITES ---------------------------------------------------------
 composite 'Constant_weight' { MeanSqErr ( Test1, Force1,4, Begin, 8./1000,
WEIGHTVALUE, SCALEVALUE, 2.0, 1.0) }
 composite 'Unit_weight_curve' { MeanSqErr ( Test1, Force1,4, Begin, .008,
WEIGHTCURVE, SCALEVALUE, 2.0, 1.0, UnitWeight) }
 composite 'Weight_Scale_Curves' { MeanSqErr ( Test1, Force1, 4, Begin, .008,
WEIGHTCURVE, SCALECURVE) }
 composite 'Wt_Scale_Curves2' { MeanSqErr ( Test1, Force1, 4, Begin, .008,
WEIGHTCURVE, SCALECURVE, _, _,Weight_1 ) }
 composite 'Wt_Scale_Curves3' { MeanSqErr ( Test1, Force1, 4, Begin,  End,
WEIGHTCURVE, SCALECURVE, _, _,Weight_1, Scale_1 ) }
 composite 'Weight_Propscale' { MeanSqErr ( Test1, Force1, 4, Begin,  End,
WEIGHTCURVE, PROPSCALE , _, _,Weight_1) }
```

```
 composite 'Dfltwt_Scalecurve' { MeanSqErr ( Test1, Force1, 4, Begin, End, ,
SCALECURVE, _, _,Weight_1, Scale_1 ) }
 composite 'Dfltwt_Propscale' { MeanSqErr ( Test2, Force2, 4, 0.002, , , PROPSCALE) }
 composite 'Dfltwt_Propscale2' { MeanSqErr ( Test2, Force2, 4, , .008, , PROPSCALE) }
 composite 'Unitwt_Unitscale1' { MeanSqErr ( Test1, Force1, numpoints, Begin, .008,
WEIGHTVALUE , SCALEVALUE) }
 composite 'Unitwt_Unitscale2' { MeanSqErr ( Test2, Force2, numpoints, Begin, .008,
WEIGHTVALUE , SCALEVALUE) }
 composite 'Unitscale' { MeanSqErr ( Test2, Force2, 4, Begin, .008, _  , SCALEVALUE) }
 composite 'Defaults_to_end' { MeanSqErr ( Test2, Force2, 4, Begin, .008) }
 composite 'Defaults_to_begin' { MeanSqErr ( Test2, Force2, 4, Begin) }
 composite 'Defaults_to_num' { MeanSqErr ( Test2, Force2, 4) }
 composite 'Defaults1' { MeanSqErr ( Test1, Force1 ) }
 composite 'Defaults2' { MeanSqErr ( Test2, Force2 ) }
 composite 'Defaults3' { MeanSqErr ( Test3, Force_Disp_Dflt ) }
```

16. Objectives and Constraints

This chapter describes the specification of objectives and constraints for the design formulation.

16.1 Formulation

Multi-criteria optimal design problems can be formulated. These typically consist of the following:

- Multiple objectives (multi-objective formulation)
- Multiple constraints

Mathematically, the problem is defined as follows:

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

where F represents the multi-objective function, $\Phi_i = \Phi_i(x_1,x_2,\ldots,x_n)$ represent the various objective functions and $g_j = g_j(x_1,x_2,\ldots,x_n)$ represent the constraint functions. The symbols x_i represent the n design variables.

In order to generate a trade-off design curve involving objective functions, more than one `objective` Φ_i must be specified so that the multi-objective

$$\mathsf{F} = \sum_{k=1}^{N} \omega_k \Phi_k . \qquad (16.1\text{-}1)$$

A component function must be assigned to each objective function where the component function can be defined as a *composite function* $\mathcal{F}$ (see Section 0) or a *response function* f . The number of objectives, N, must be specified in the problem description (see Section 8.2).

16.2 Defining an objective function

This command identifies each objective function. The name of the objective is the same as the component, which can be a response or composite.

Command file syntax:

```
objective name { weight <1> }
```

Examples:

```
objective 'Intrusion_1'
objective 'Intrusion_2' 2.
objective 'Acceleration' 3.
```

for

$$\text{Multi-objective} = \mathsf{F} = \Phi_1 + 2\Phi_2 + 3\Phi_3$$
$$= \mathcal{F}_1 + 2\mathcal{F}_2 + 3f_2$$

Remarks:

1. The distinction between objectives is made solely for the purpose of constructing a Pareto-optimal curve involving multiple objectives. *However it is still better to construct a Pareto optimal curve using a varying constraint bound instead of varying weights. See Sections 16.4 and 18.4.*

2. Objectives can be specified in terms of composite functions and/or response functions.

3. The weight applies to each objective as represented by ω_k in Equation (11.1).

The default is to minimize the objective function. The program can however be set to maximize the objective function. In LS-OPT*ui*, maximization is activated in the Objective panel.

Command file syntax:

```
Maximize
```

Example:

```
Response 'Mass' "DynaMass 3 13 14 16 MASS"
Maximize
Objective 'Mass'
Constraint 'Acceleration'
```

In LS-OPT*ui*, objectives are defined in the Objective panel (Figure 16-1):

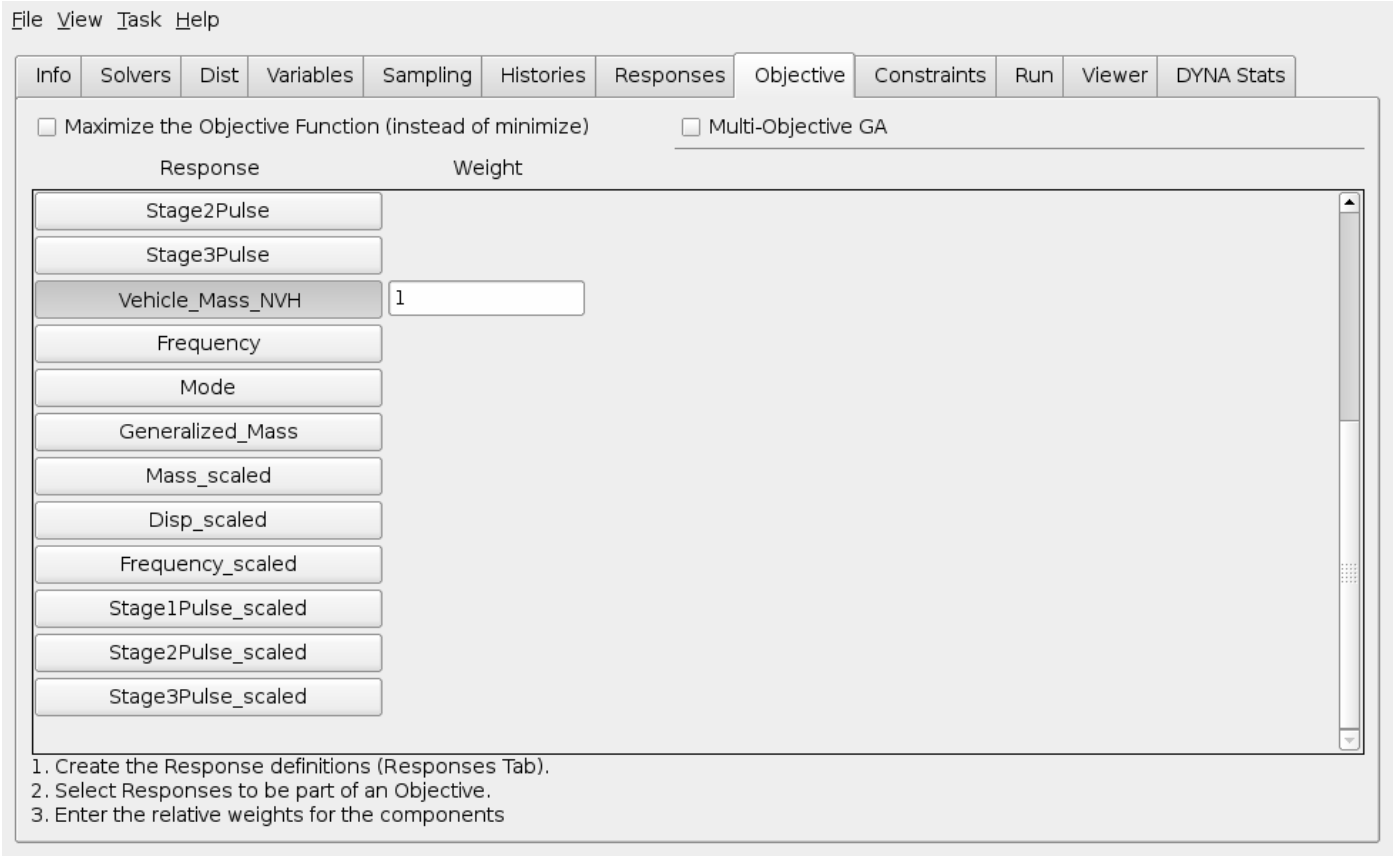


Figure 16-1: Objective panel in LS-OPT*ui*

16.3 Defining a constraint

This command identifies each constraint function. The constraint has the same name as its component. A component can be a response or composite.

Command file syntax:

```
constraint constraint_name
```

Examples:

```
history 'displacement_1'  "DynaASCII nodout 'r_disp' 12789 TIMESTEP 0.0 SAE 60"
history 'displacement_2'  "DynaASCII nodout 'r_disp' 26993 TIMESTEP 0.0 SAE 60"
history 'Intrusion'       {displacement_2 - displacement_1}
response Intrusion_80     {Intrusion(80)}
```

```
constraint 'Intrusion_80'
```

Remark:

1. Constraints can be specified in terms of response functions or composite functions.

In LS-OPT*ui*, constraints are defined in the Constraints panel (Figure 16-2):

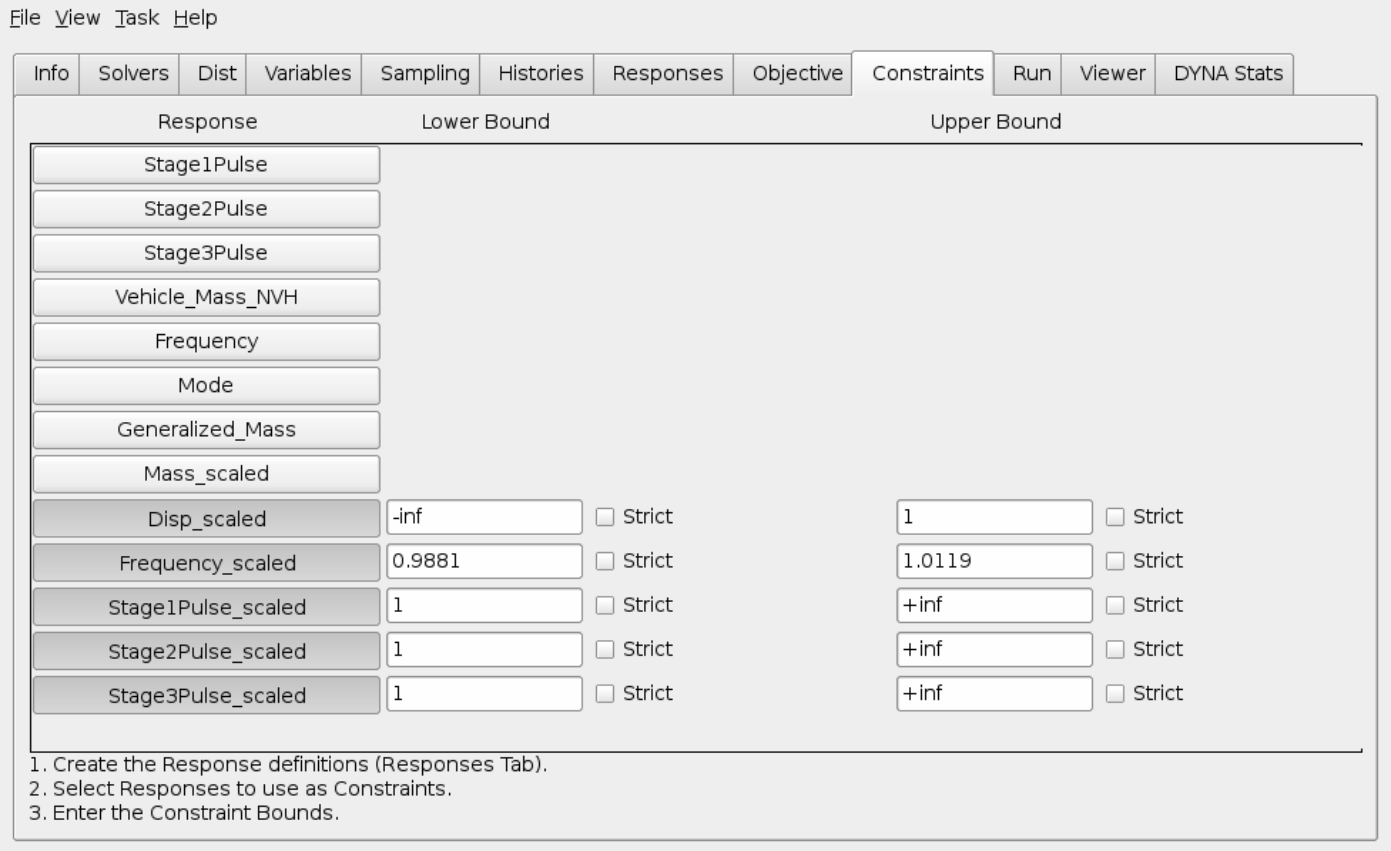


Figure 16-2: Constraints panel in LS-OPT*ui*

16.4 Bounds on the constraint functions

Upper and lower bounds may be placed on the constraint functions.

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

Command file syntax:

```
lower bound constraint constraint_name value <-10^+30>
```

```
upper bound constraint constraint_name value <+10^+30>
probability lower bound constraint constraint_name prob_value
probability upper bound constraint constraint_name prob_value
```

Example:

```
Lower bound constraint 'Stress' 1.e-6
Upper bound constraint 'Stress' 20.2
```

Remark:

1. A flag can be set to identify specific constraint bounds to define a reasonable design space. For this purpose, the `move` environment must be specified (See Section 13.6).

16.5 Minimizing the maximum response or violation*

Refer to Section 5.1 for the theory regarding strict and slack constraints. To specify hard (strict) or soft (slack) constraints, the following syntax is used:

Command file syntax:

```
strict strictness_factor <1>
slack
```

Each command functions as an environment. Therefore all `lower bound constraint` or `upper bound constraint` commands which appear after a `strict/slack` command will be classified as `strict` or `slack`.

In the following example, the first two constraints are slack while the last three are strict. The purpose of the formulation is to compromise only on the knee forces if a feasible design cannot be found.

Example:

```
$ This formulation minimizes the average knee force but
$ constrains the forces to 6500.
$ If a feasible design is not available, the maximum violation
$ will be minimized.
$
$ Objective:
$-----------
composite 'Knee_Forces' type weighted
composite 'Knee_Forces' response 'Left_Knee_Force'   0.5
composite 'Knee_Forces' response 'Right_Knee_Force'  0.5
objective 'Knee_Forces'
$
$ Constraints:
$-------------
SLACK
Constraint 'Left_Knee_Force'
Upper bound constraint 'Left_Knee_Force'            6500.
```

```
$
Constraint 'Right_Knee_Force'
Upper bound constraint 'Right_Knee_Force'            6500.
$
STRICT
Constraint 'Left_Knee_Displacement'
Lower bound constraint 'Left_Knee_Displacement'      -81.33
$
Constraint 'Right_Knee_Displacement'
Lower bound constraint 'Right_Knee_Displacement'     -81.33
$
Constraint 'Kinetic_Energy'
Upper bound constraint 'Kinetic_Energy'              154000.
```

The `composite` function is explained in Section 0. Note that the same response functions appear both in the objective and the constraint definitions. This is to ensure that the violations to the knee forces are minimized, but if they are both feasible, their average will be minimized (as defined by the composite).

The constraint bounds of all the soft constraints can also be set to a number that is impossible to comply with, e.g. zero. This will force the optimization procedure to always ignore the objective and it will minimize the maximum response.

In the following example, the objective is to minimize the maximum of '`Left Knee Force`' or '`Right Knee Force`'. The displacement and energy constraints are strict.

Example:

```
$ This formulation minimizes the maximum knee force
$ Because the knee forces are always positive,
$ the objective will be ignored and the knee force
$ minimized
$
$ Objective:
$-----------
composite 'Knee_Forces' type weighted
composite 'Knee_Forces' response 'Left_Knee_Force'   0.5
composite 'Knee_Forces' response 'Right_Knee_Force'  0.5
objective 'Knee_Forces'
$
$ Constraints:
$-------------
SLACK
Constraint 'Left_Knee_Force'
Upper bound constraint 'Left_Knee_Force'             0.
$
Constraint 'Right_Knee_Force'
Upper bound constraint 'Right_Knee_Force'            0.
$
STRICT
Constraint 'Left_Knee_Displacement'
Lower bound constraint 'Left_Knee_Displacement'      -81.33
$
```

```
Constraint 'Right_Knee_Displacement'
Lower bound constraint 'Right_Knee_Displacement'     -81.33
$
Constraint 'Kinetic_Energy'
Upper bound constraint 'Kinetic_Energy'              154000.
```

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.

16.6 Internal scaling of constraints

Command file syntax:

```
Constraint constraint_name scale lower bound value <1.0>
Constraint constraint_name scale upper bound value <1.0>
```

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 is applied internally to constraint j as follows:

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

A logical choice for the selection of *s* is $s_j^L = L_j$ and $s_j^U = U_j$, so that the above inequalities become

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

internally and in the infeasible phase:

$$\frac{-g_j(x)}{L_j} + 1 \le e\,; \quad \frac{g_j(x)}{U_j} - 1 \le e\,; \; e \ge 0$$

Example:

```
Constraint 'Left_Knee_Displacement'
Lower bound constraint 'Left_Knee_Displacement'       -81.33
Constraint 'Left_Knee_Displacement' scale lower bound 81.33
```

17. Running the Design Task

This chapter explains simulation job-related information and how to execute a design task from the graphical user interface.

The available tasks are optimization, probabilistic evaluation, and repair of an existing job.

17.1 Optimization

The optimization process is triggered by the `iterate` command in the input file or by the Run command in the Run panel in LS-OPT*ui* (Figure 17-1). The optimization history is written to the `OptimizationHistory` file and can be viewed using the View panel.

17.1.1 Number of optimization iterations

The number of optimization iterations are specified in the appropriate field in the Run panel. 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 below are reached first, LS-OPT will terminate and not perform the maximum number of iterations.

Command file syntax:

`iterate` *maximum_number_of_iterations*

17.1.2 Optimization termination criteria

The user can specify tolerances on both the design change (Δx_i) and the objective function change (Δf) and whether termination is reached if either, or both these criteria are met. The default selection is *and*, but the user can modify this by selecting *or*.

Refer to Section 20.1 for the modification of the stopping type in the Command File.

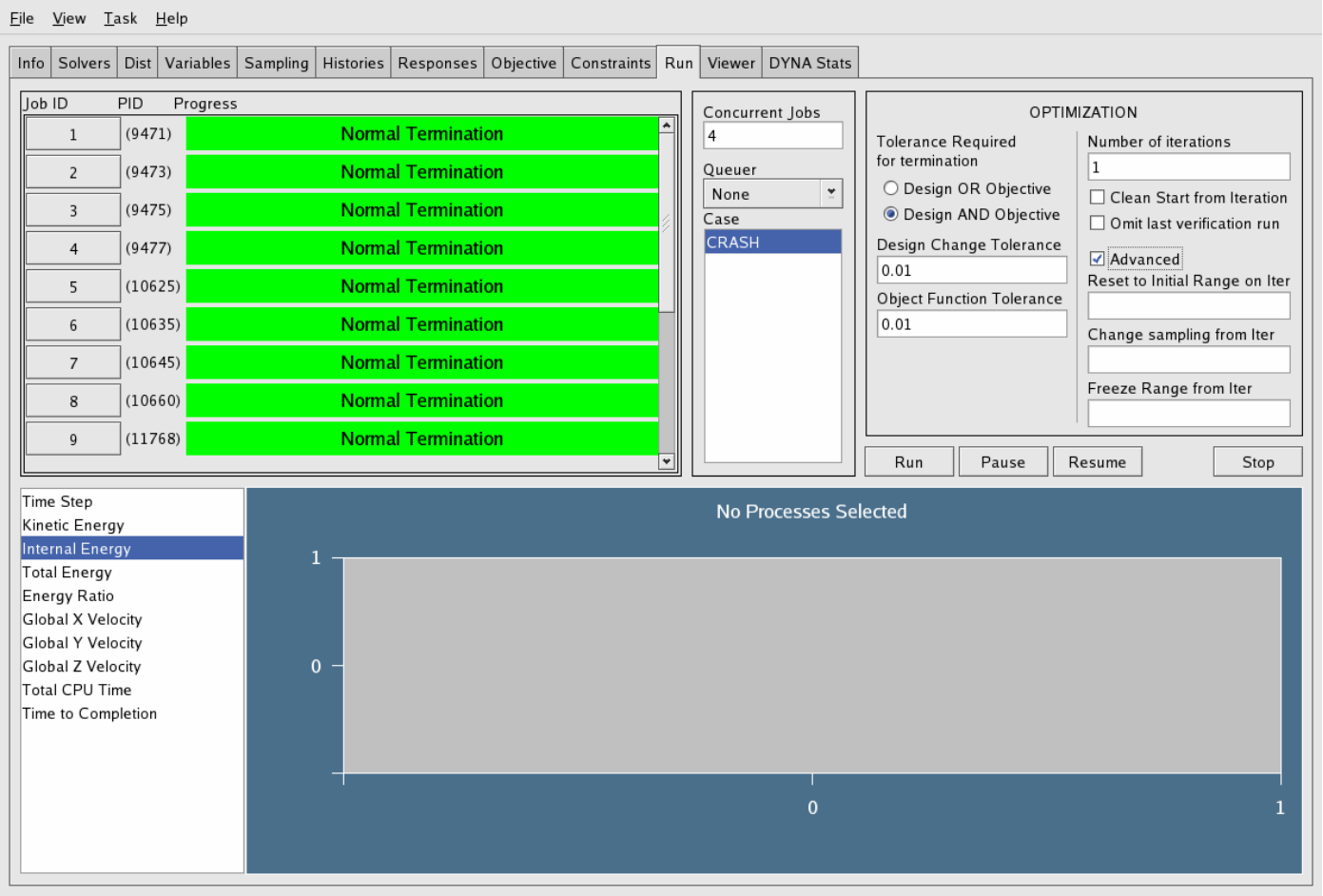


Figure 17-1: Run panel in LS-OPT*ui* (Advanced options displayed)

17.2 Probabilistic Evaluation

Both a Monte Carlo and a metamodel-based Monte Carlo evaluation can be scheduled from the user interface. The task must be set to the relevant procedure.

Section 12.4 regarding probabilistic evaluation contains more details on the available options.

The results can be viewed using the View panel. The histogram, tradeoff, and covariance plots are pertinent to a pure Monte Carlo analysis. For a metamodel-based Monte Carlo evaluation, the accuracy, ANOVA, and stochastic contribution plots are relevant in addition to the histogram, tradeoff, and covariance plots.

The LS-DYNA results can be investigated for possible bifurcations using the tools described in chapter 21.

17.3 Restarting

When a solution is interrupted (through the Stop button) or if a previous optimization run is to be repeated from a certain starting iteration, this can be specified in the appropriate field in the Run panel (Figure 17-1).

17.4 Job concurrency

When LS-OPT is run on a multi-processor machine, the user can select how many simulations (jobs) can run concurrently on different processors (see Figure 17-1). Only the solver process and response extraction are parallellized. The preprocessor processes run serially. The number of Concurrent Jobs is ignored for jobs that are run by a queuing system.

17.5 Job distribution

When a queuing system is available, its operation can be specified in the Run panel (Figure 17-1).

17.6 Job and analysis monitoring

The Run panel allows a graphical indication of the job progress with the green horizontal bars linked to estimated completion time. 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.

When using LS-DYNA, the user can also view the progress (time history) of the analysis by selecting one of the available quantities (Time Step, Kinetic Energy, Internal Energy, etc.).

17.7 Repair or modification of an existing job

Several kinds of repairs and modifications are possible for an existing optimization iteration or a probabilistic analysis. The repair depends on the LSOPT database files as described in Section 9.7. The available repair tasks are:

- *Read points*. The *CASE*/Experiments.*iteration* file is reconstructed from the runs executed. The experimental points can be extracted from the database in the job directories and the experimental design thereby reconstructed.
- *Augment points of a Metamodel-based analysis*. Points are added to the existing experimental design. This option is only available for the following experimental designs types: D-Optimal, space-filling, random, and Latin Hypercube. The D-Optimal and space-filling experimental designs will be computed taking in consideration the previously computed points. Both the random and 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 an experimental design does not exist, new points will be created.
- *Augment Points of a Monte Carlo analysis*. Points are added to the existing experimental design. This option is only available for the following experimental designs types: random and Latin

Hypercube. Both the random and the Latin Hypercube experimental design points will be computed using the number of previously computed points as a seed to random number generator.

- *Run Jobs*. The LS-DYNA 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 LS-DYNA input file used will be regenerated from the files specified in the main directory. The preprocessor, if one is specified, will be rerun.
- *Extract Results*. The results will be extracted from the runs. This option also allows the user to change the responses for an existing iteration or Monte Carlo analysis.
- *Read user results*. Extract results from `AnalysisResults.PRE.<casename>`. The `AnalysisResults.PRE.<casename>` file will be generated if the analysis results are imported from a `.csv` or `.txt` file (see Section 17.9).
- *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 *Read user results* above).
- *Analyze checkpoints*. Create a table with the error measures of a given set of points. See Section 13.9.
- *Optimize*. The metamodels are used for metamodel optimization. A new optimum results database is created. The "ExtendedResults" file will be updated.

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.

The use of *.PRE.* databases for Experiments and DesignFunctions are not supported by the repair facility. See Sections 13.5, and 13.6 for the use of these databases.

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 as *n*+1 for a new run.

Command file syntax:

```
read experiments iteration_number
design more metamodel iteration_number
design more monte carlo iteration_number
run iteration_number
run failed iteration_number
extract results iteration_number
read user results iteration_number
approximate iteration_number
check file iteration_number
optimize iteration_number
```

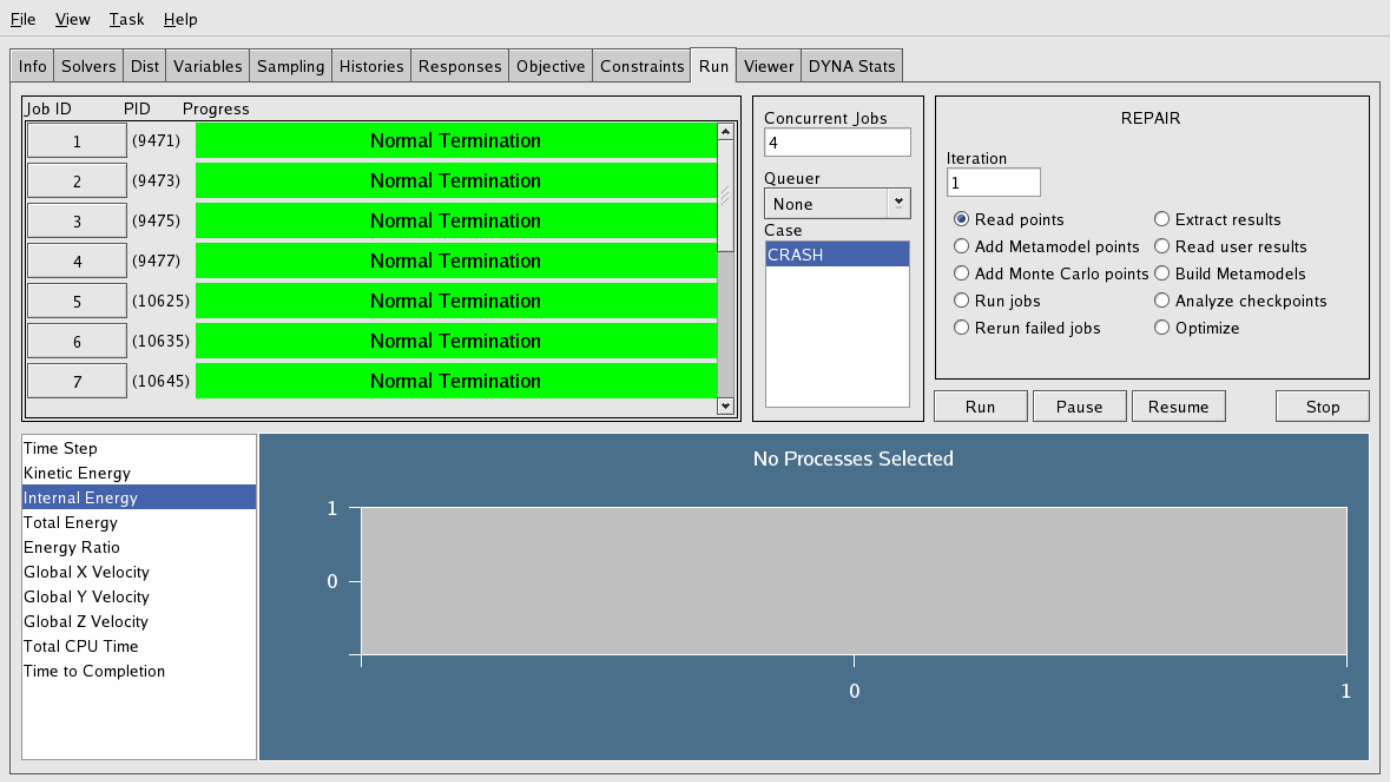


Figure 17-2: Repair panel

17.8 Tools

A number of tools are available for miscellaneous operations:

- *Clean*. The directory structure created by LS-OPT and all files in this directory structure are deleted.
- *Update database*. LS-OPT Version 2.2 function databases can be updated to Version 3 format.
- *Gather LS-OPT database.* See Section 17.10.

Command file syntax:

```
clean
pack database
update database
```

17.9 Importing user-defined analysis results

A table (in text form) of existing analysis results can be used for analysis. The command to import the file is given as:

Command file syntax:

```
solver response user filename_csv_format
```

Example:

```
solver response user "/home/test/ImportResults/crash2.csv"
```

An example of a analysis results file (with 2 dimulation 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
```

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. The types are defined as:

Symbol	Explanation
`dv`	Design variable
`nv`	Noise variable
`rs`	Response

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

The steps for importing user-defined analysis result files using the GUI are as follows:

1. *Solvers panel:* Browse for the text file in the "Import user results" tab. The browser has a preference for `.csv` and `.txt` files.
2. Specify a name for the analysis case and "Add" the case.
3. *Variables and Responses panels.* It is recommended to check the variables and responses in these panels. The variables and responses will be displayed and automatically associated with the correct analysis case.
4. *Menu bar.* Choose the "Repair" task.
5. *Run panel.* Select "Read user results" and "Run".
6. Select "Build Metamodels" and "Run".
7. Optimization.
    1. Define the Objectives and/or constraints.
    2. Change to the "Metamodel-based Optimization" task and "Run". Select the "Omit verification run" option in the "Run" panel to avoid attempting a verification run. An optimization history is created.

17.10 Saving/compressing the LS-OPT database after completing a run

Using the Tools function, the database can be gathered up and compressed in a file called `lsopack.tar.gz (lsopack.zip` in Windows). The packed database is suitable for post-processing on any computer platform. The repair selection is: *Gather LS-OPT database.* The gathered database cannot be used to visualize results stored in sub-sub-directories (e.g. some `MeanSqErr` post-pocessing).

18. Viewing Results

This chapter describes the viewing of metamodeling surfaces, metamodeling accuracy, optimization history, trade-off plots, ANOVA results, as well as statistical plots such as histograms, stochastic contribution of the variables, covariance, and coefficient of correlation plots.

The View panel in LS-OPTui is used to view the results of the optimization process. The results include the metamodelling accuracy data, optimization history of the variables, dependents, responses, constraints and objective(s). Trade-off data can be generated using the existing response surfaces, and ANOVA results can be viewed.

There are three options for viewing accuracy and tradeoff (anthill plots), namely viewing data for the *current* iteration, for *all previous* iterations simultaneously, *all* iterations (see e.g. Figure 18-7). The last option will also show the last verification point (optimal design) in green.

18.1 Metamodel

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 following options are available:

18.1.1 Setup

The selection of the 2 variables and the response function is done here. The sliders allow changing of the variable values for unselected variables (variables not plotted). The slider for the active variables can be activated by selecting the “Show Predicted Value” option under the Points tab.

18.1.2 Ranges

A selection can be made to plot the surface across either the full design space or the subregion. The region size can also be adjusted manually. The check box prevents shrinking of the view box when changing to a different (usually higher) iteration. See Neural Net plot in Figure 18-3.

18.1.3 Points

Point plotting options

Selection	Description
Analysis Results	Points are plotted for current iteration
All iterations	Points for previous iterations are added
Project points to surface	The points are projected on the surface to improve visibility. Future versions will have a transparency option.
Residuals	Shows a black vertical line connecting the computed and predicted values.
Feasible runs	Show feasible runs only
Infeasible runs	Show infeasible runs only
Failed runs on surface	Failed runs such as error terminations are projected to the surface in grey

Point status

Selection	Description
Feasibility	Feasible points are shown in green, infeasible points in red (Figure 18-1).
Previous b/w	The points for the current iteration are shown in green (feasible) or red (infeasible). Previous points as light grey (feasible) or dark grey (infeasible)
Iterations	The iteration sequence is shown using a color progression from blue through red. See Figure 18-2.
Optimum runs	Optimal points are shown in green/red and all other points in white.

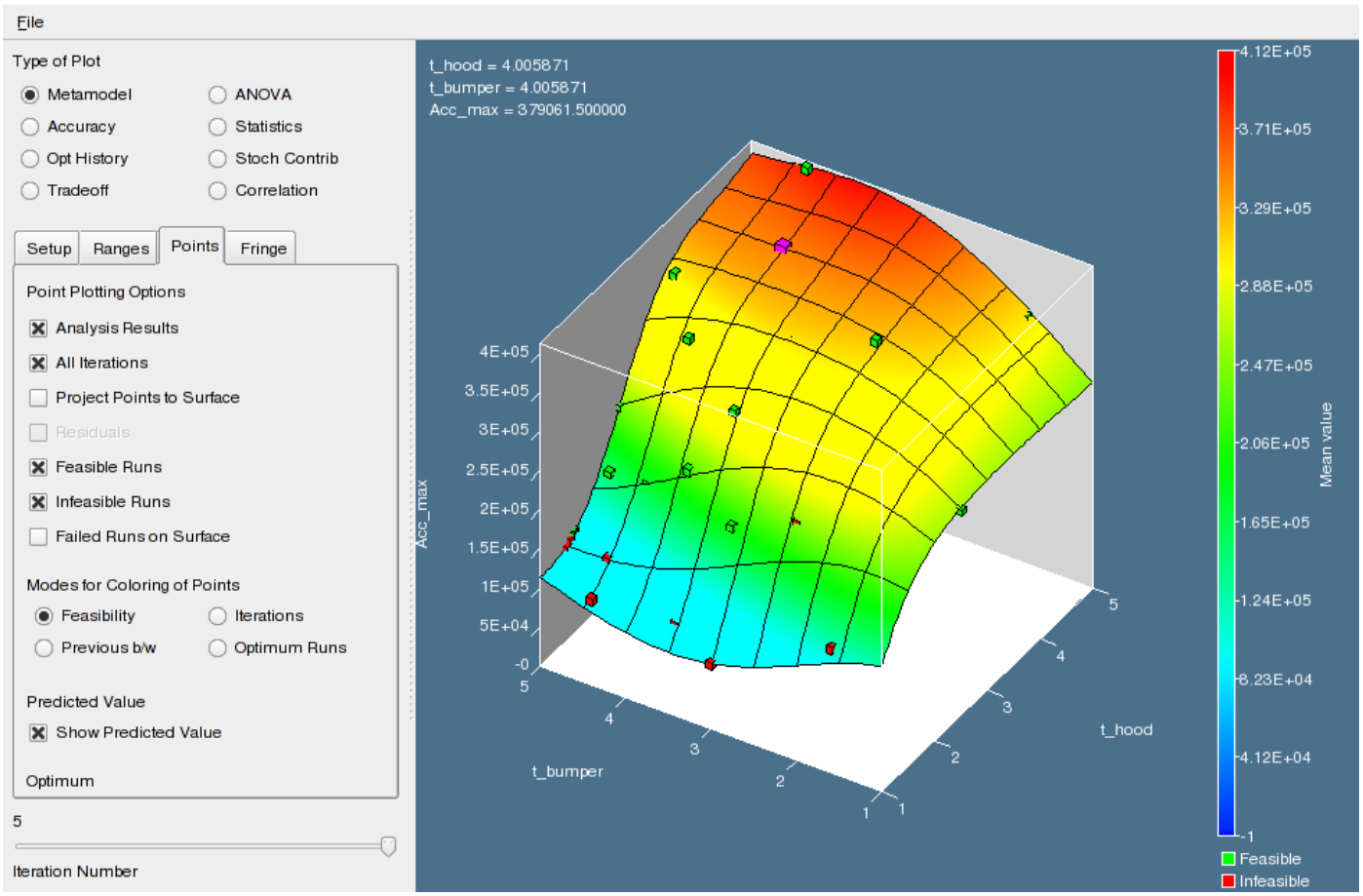


Figure 18-1: Metamodel plot showing feasible (green) and infeasible (red) points. The predicted point is shown in violet (*t_hood* = 4, *t_bumper* = 4) with the values displayed at the top left.

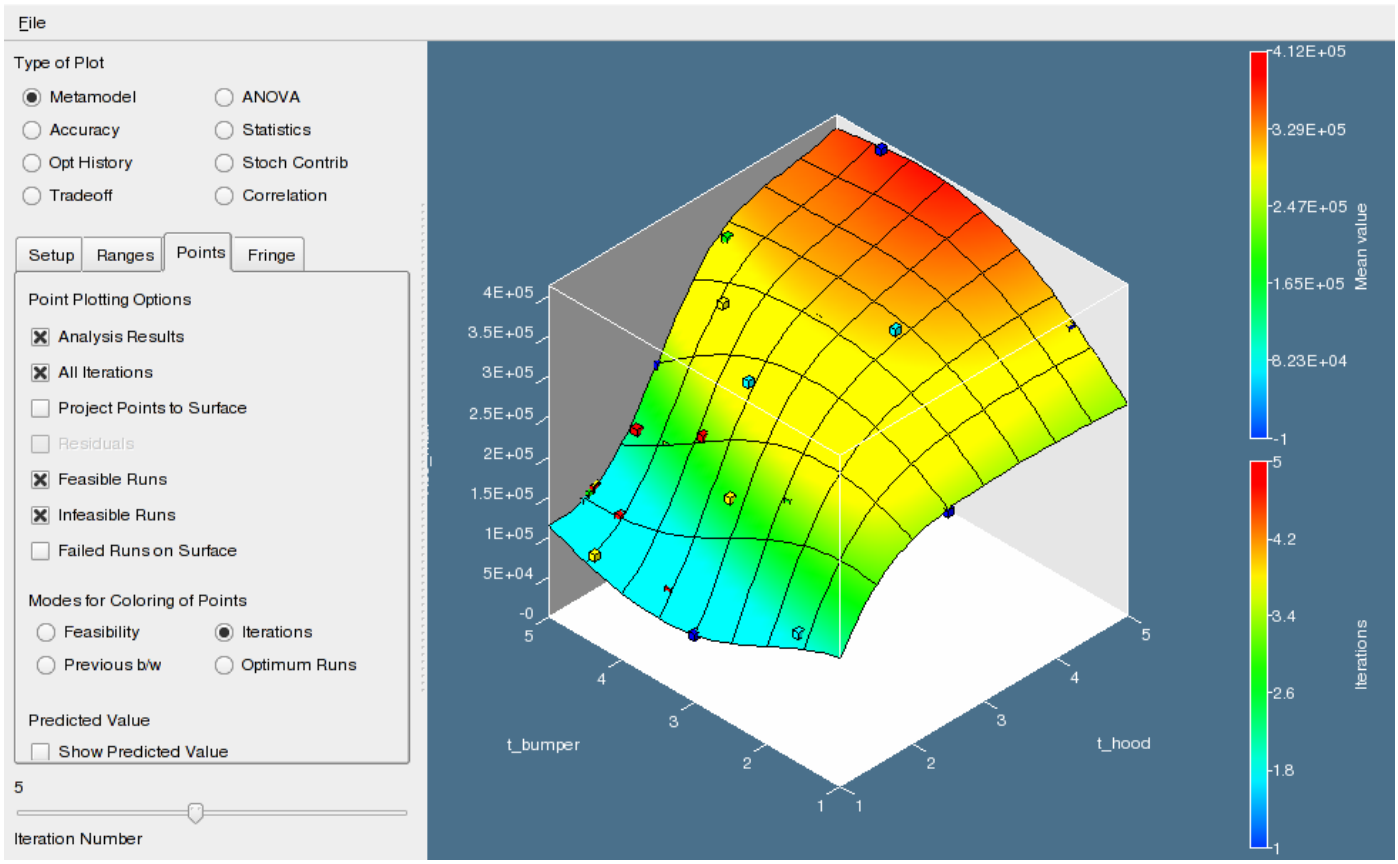


Figure 18-2: Metamodel plot showing point color coding for iteration numbers.

Predicting a value

Predicted values can be obtained by selecting the “Predicted Value” option and moving the sliders in the “Setup” menu. The predicted value is displayed in the top left corner (Figure 18-1).

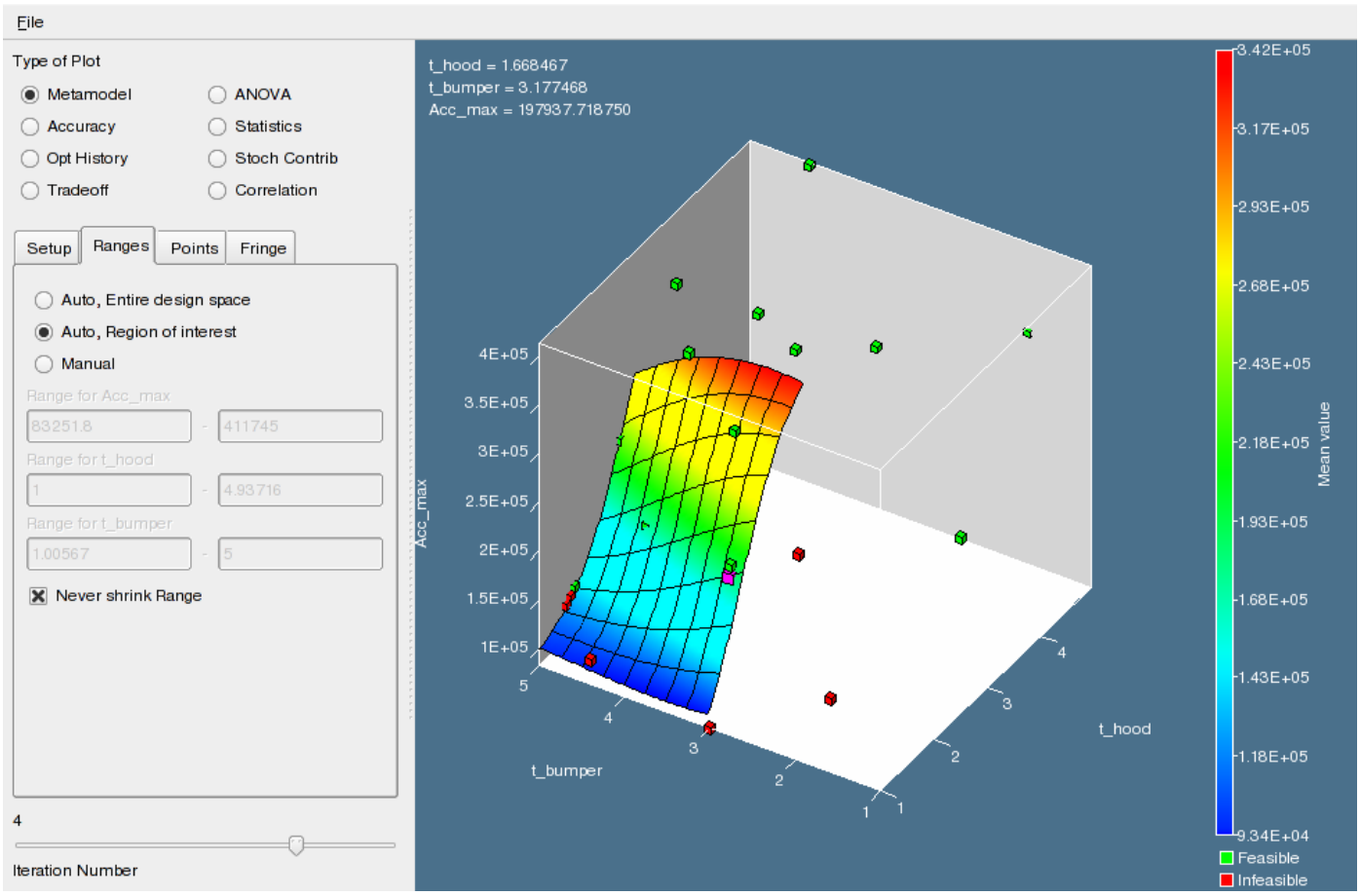


Figure 18-3: Surface plot representing only the region of interest of the fourth iteration.

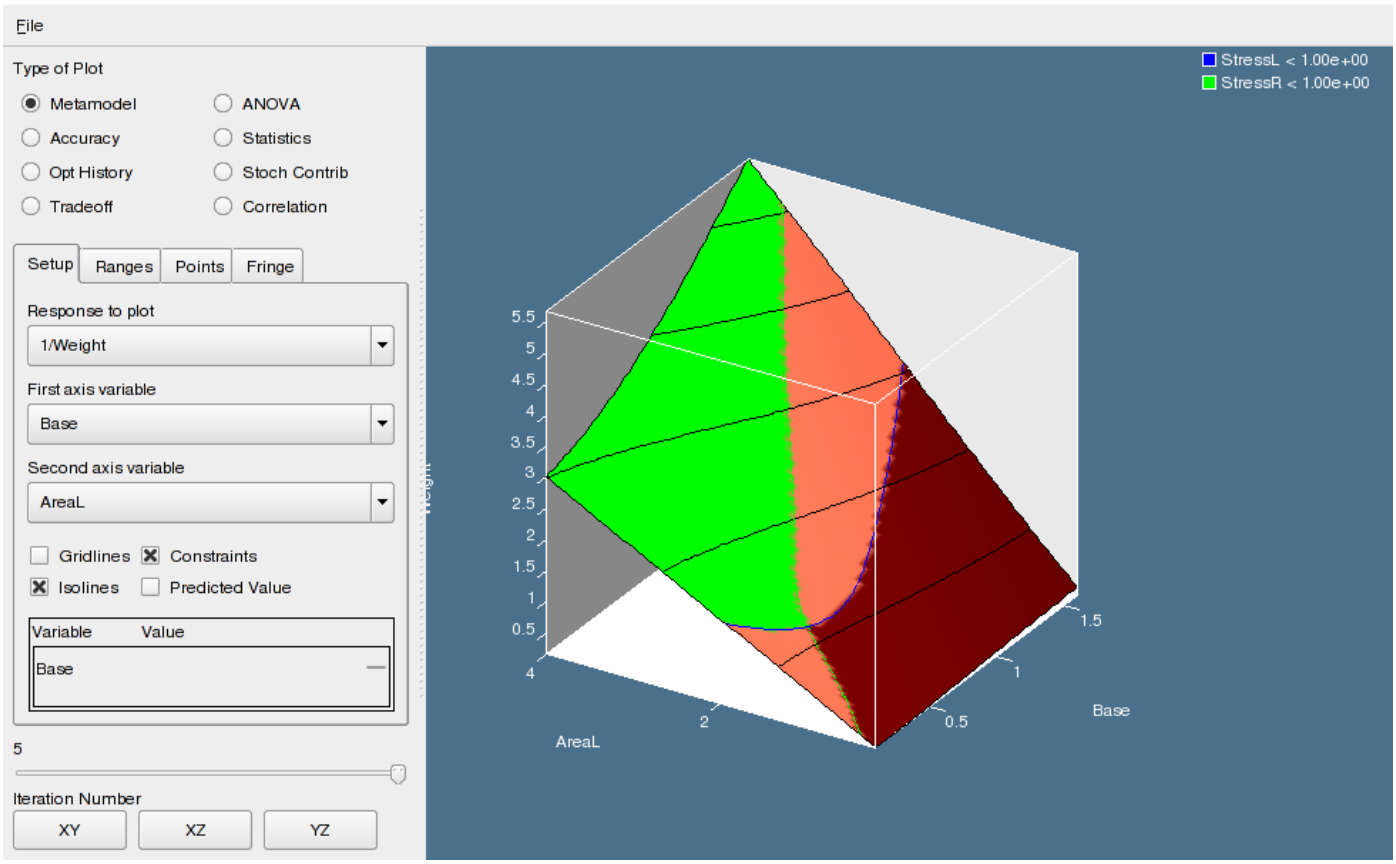


Figure 18-4: Plot showing isolines on the objective function as well as constraint contours and feasibility. Feasible regions are in green. Shade of red shows degree of infeasibility (number of violated constraints). Note legend describing constraints at the top right.

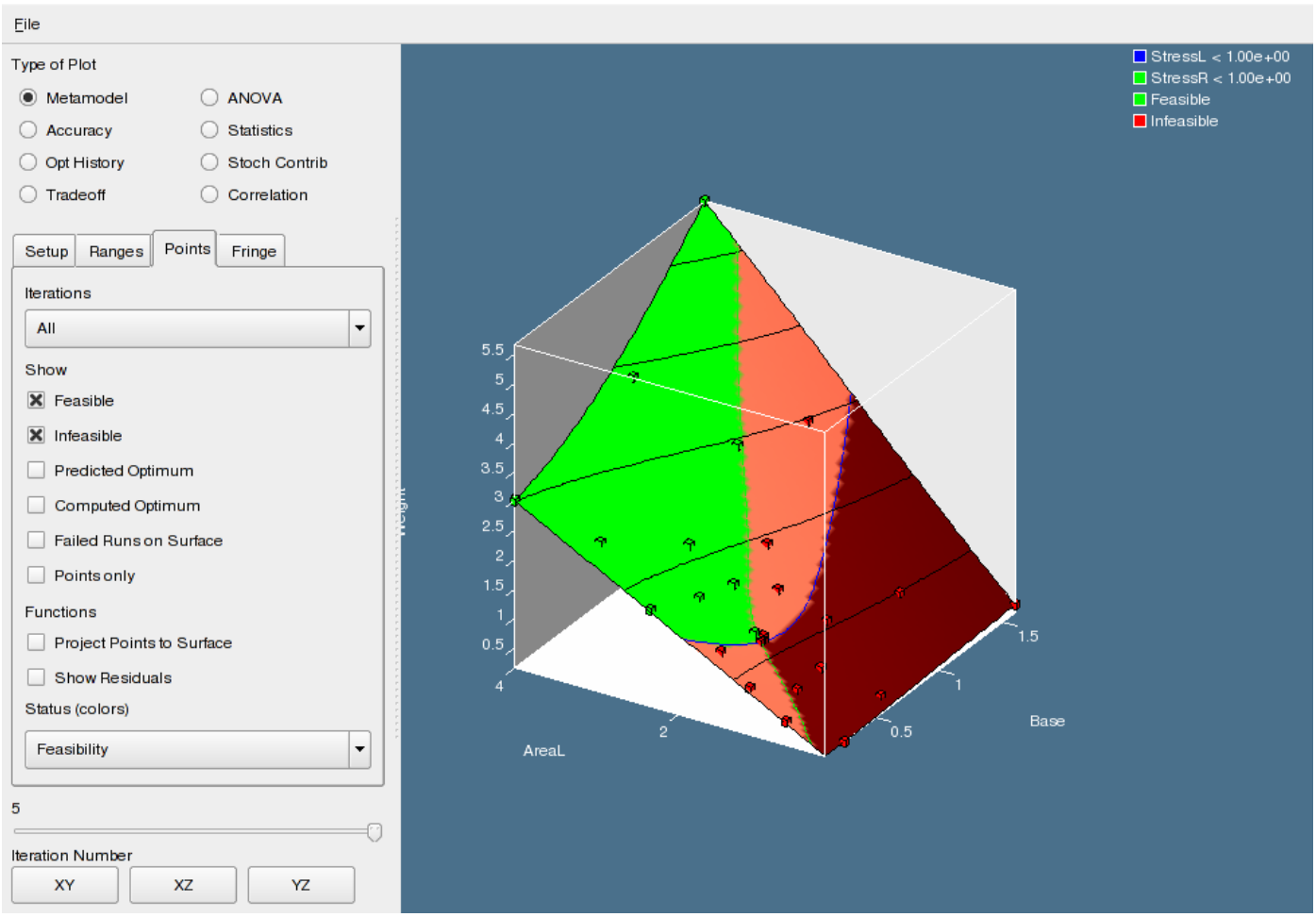


Figure 18-5: Plot showing isolines and points opposite the “Points” tab.

18.1.4 Fringe plot options for neural nets

The options are function value or standard deviation of the Neural Net committee values. See Figure 18-6.

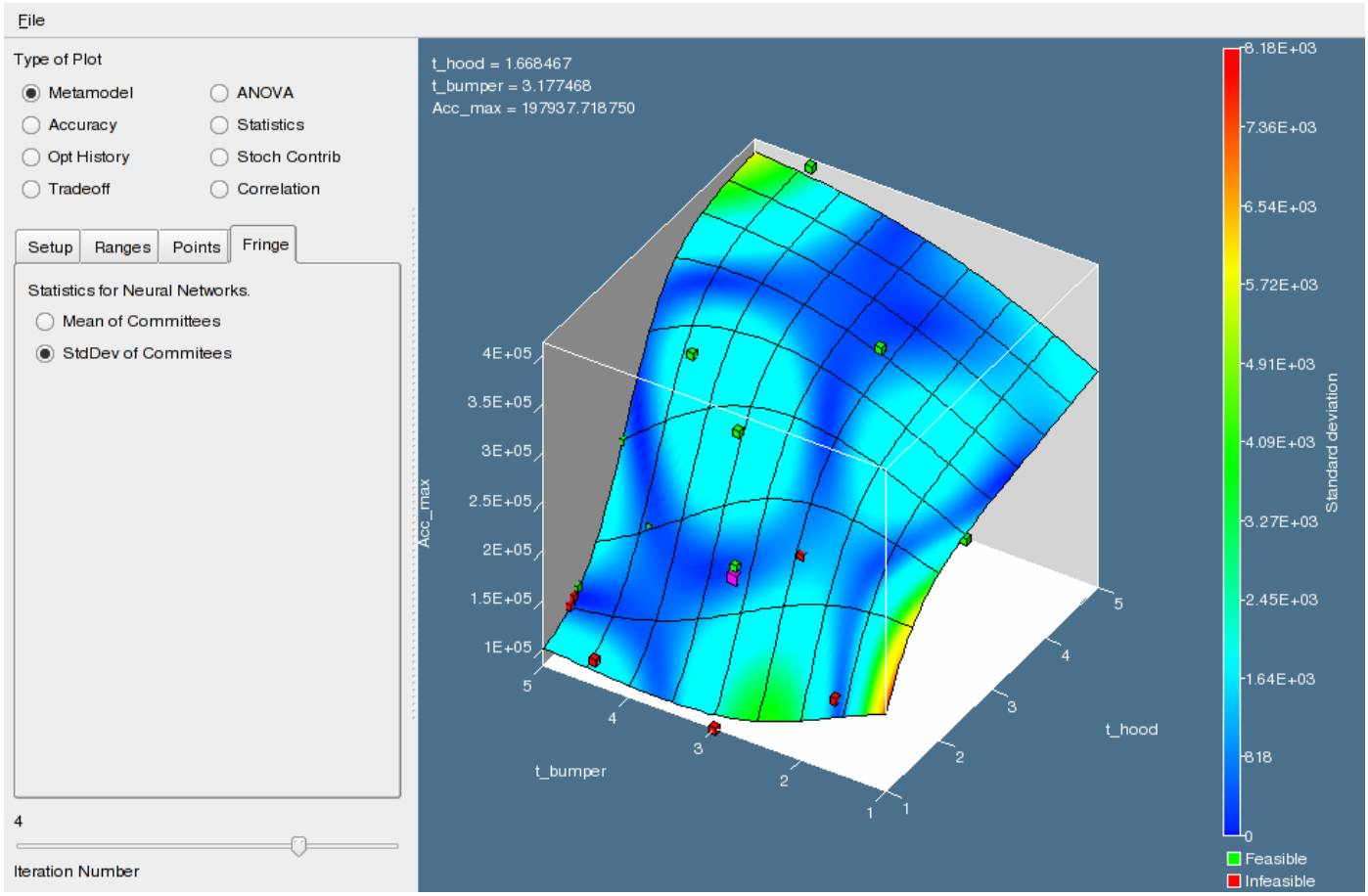


Figure 18-6: Metamodel plot showing standard deviation of the Neural Net committee values.

18.2 Metamodel accuracy

The accuracy of the metamodel fit is illustrated in a Computed vs. Predicted plot (Figure 18-7). By clicking on any of the red squares, the data of the selected design point is listed. For LS-DYNA results, LS-PREPOST can then be launched to investigate the simulation results. The results of each iteration are displayed separately using the slider bar. The iterations can be viewed simultaneously by selecting **All Previous** or **All**. The **All** selection shows the final verification point in green (see Figure 18-7). The error measures are displayed in the heading.

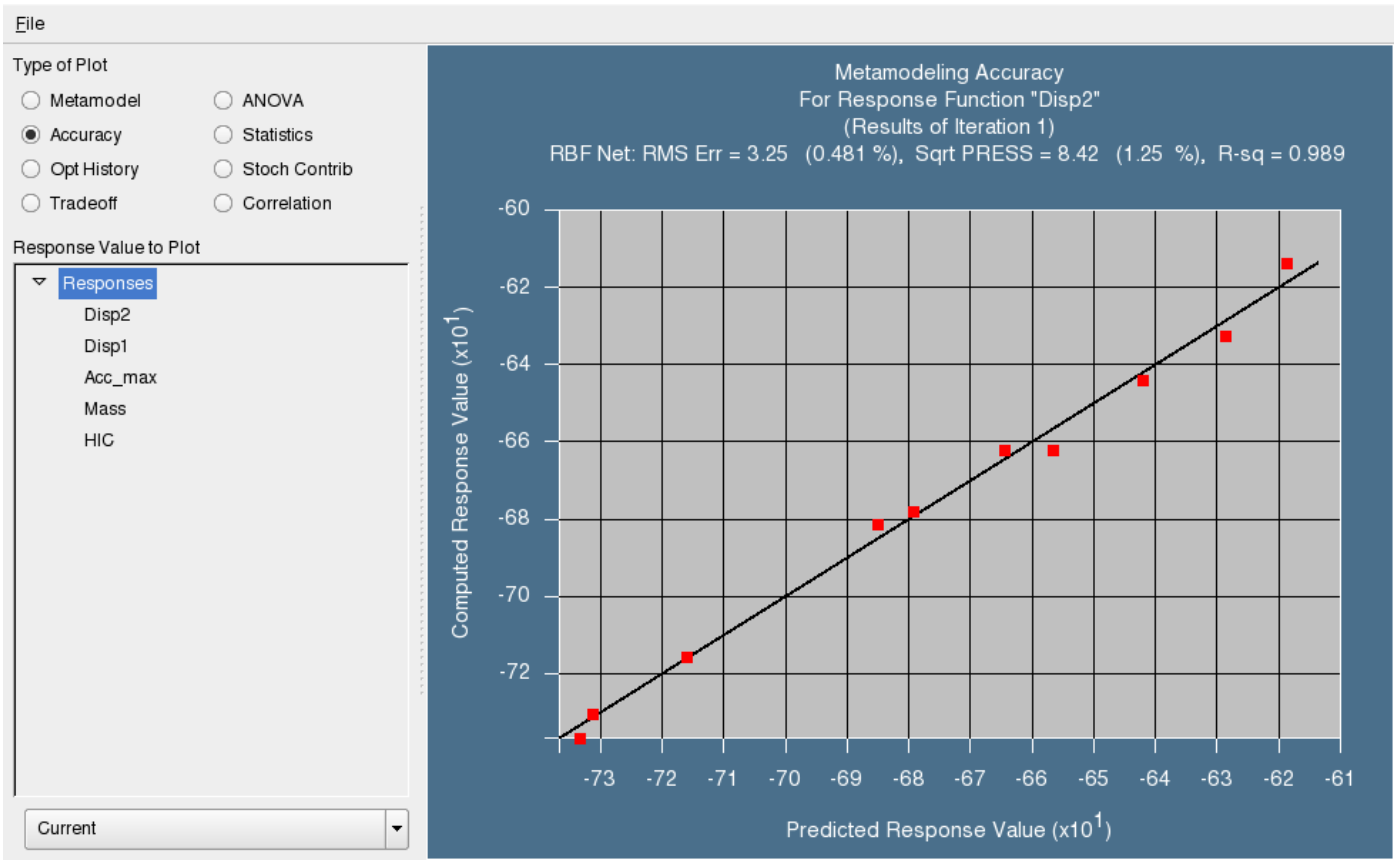


Figure 18-7: Computed vs. Predicted plot in View panel in LS-OPT*ui*

18.3 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) can be plotted by clicking on the Optimization History button (Figure 18-8). For the variables, the upper and lower bounds (subregion) are also displayed. 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. For the error parameters, only one solid red line of the optimization history is plotted. RMS, Maximum and R^2 error indicators are available.

By clicking on any of the red squares, the data of the selected design point is listed. For LS-DYNA results, LS-PREPOST can then be launched to investigate the simulation results.

MeanSqErr composites in the history list are depicted with special icons to emphasize their additional functionality. By clicking near any of the iterations, the point values are given as well as a selection button for viewing the history comparison using LS-PREPOST.

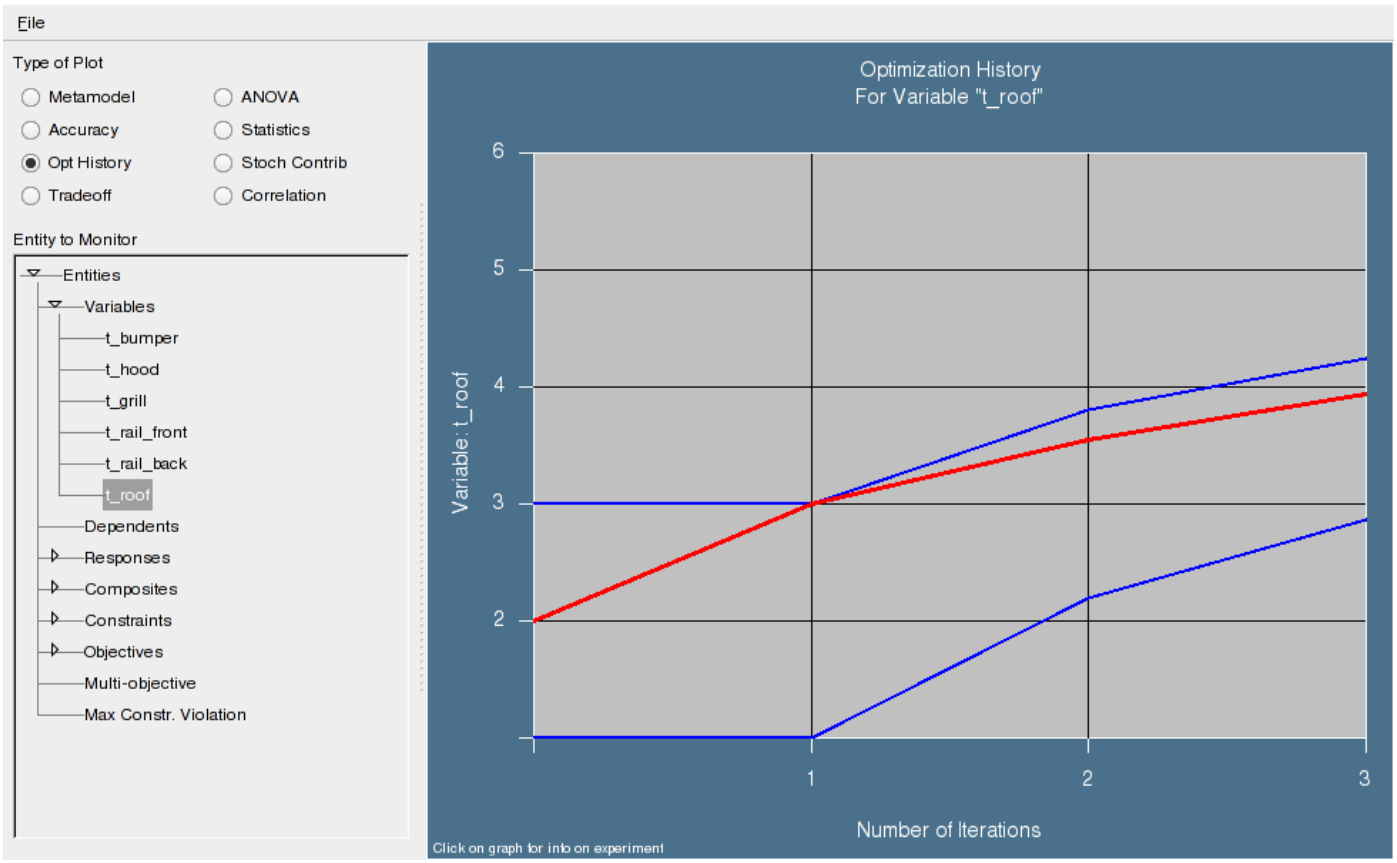


Figure 18-8: Optimization History plot in View panel in LS-OPT*ui*

18.4 Trade-off and anthill plots

The results of all the simulated points appear as dots on the trade-off plots. This feature allows the two-dimensional plotting of any variable/response against any other variable/response.

Trade-off studies can also be conducted based on the results of an optimization run. This is because the response surfaces for each response are at that stage available at each iteration for rapid evaluation.

Trade-off is performed in LS-OPT*ui* using the View panel and selecting Trade-off (Figure 18-9).

Trade-off curves can be developed using either constraints or objectives. The curve can be plotted with any of the variables, responses, composites, constraints or objectives on either of the two axes. Care should be taken when selecting e.g. a certain constraint for plotting, as it may also be either a response or composite, and that this value maybe different from the constraint value, depending on whether the constraint is active during the trade-off process. The example in the picture below has Constraint: Intrusion selected for the X-Axis Entity, and not Composite: Intrusion.
Solutions to the trade-off optimization problem falling outside the region of interest are connected by dotted lines to indicate extrapolation of the metamodel.

To be able to view the results of composite functions spanning two or more disciplines or cases, the duplicate sampling method (Section 5.2) 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.

An example of trade-off is given in Section 22.1 and 22.2.

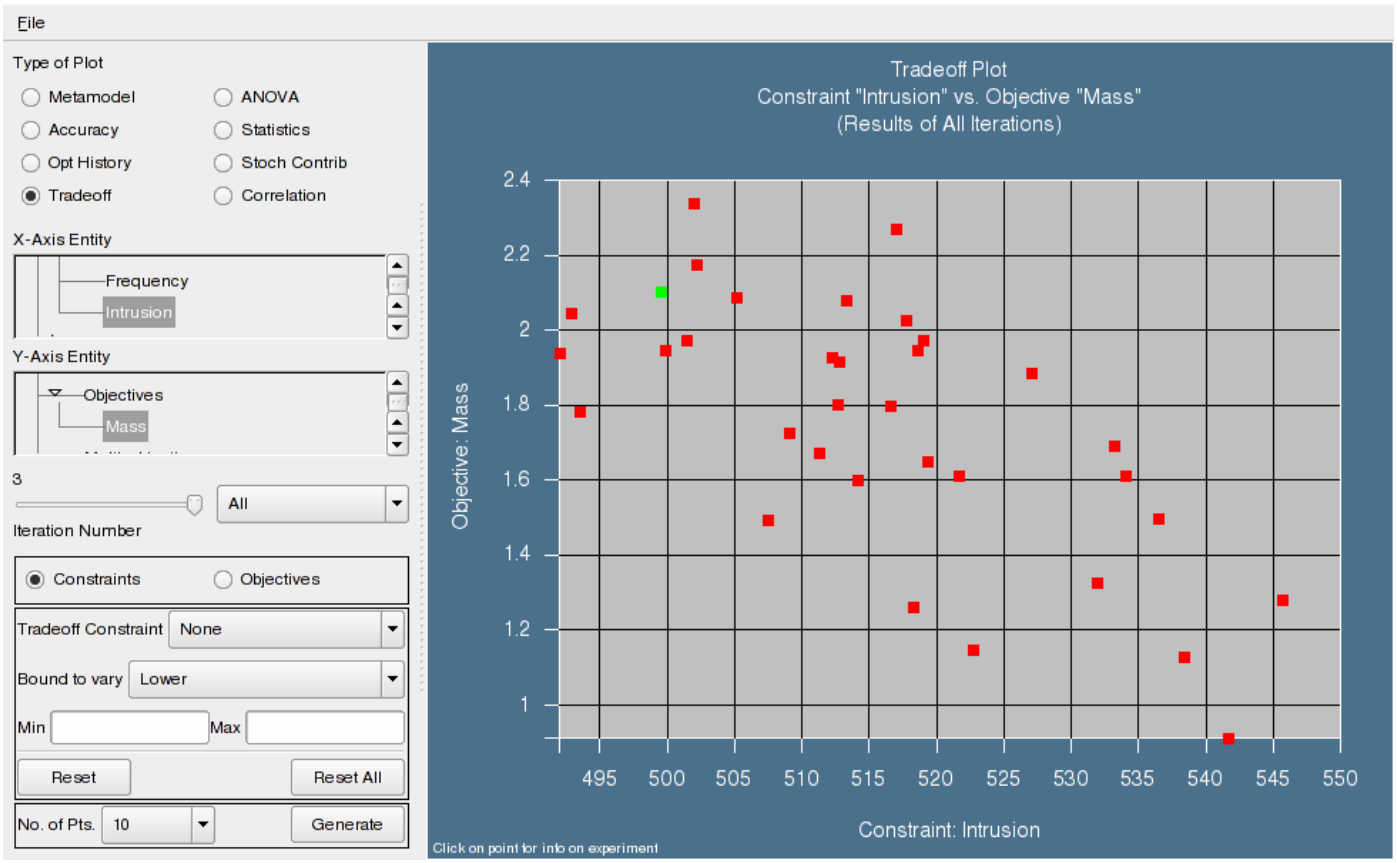


Figure 18-9: Trade-off plot in View panel in LS-OPT*ui*

18.5 Variable screening

The Analysis of Variance (ANOVA) (refer to Section 2.4) 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 5.2 (theory) and 22.6 (example)).

The ANOVA results are viewed in bar chart format by clicking on the ANOVA button. The ANOVA panel is shown in Figure 18-10.

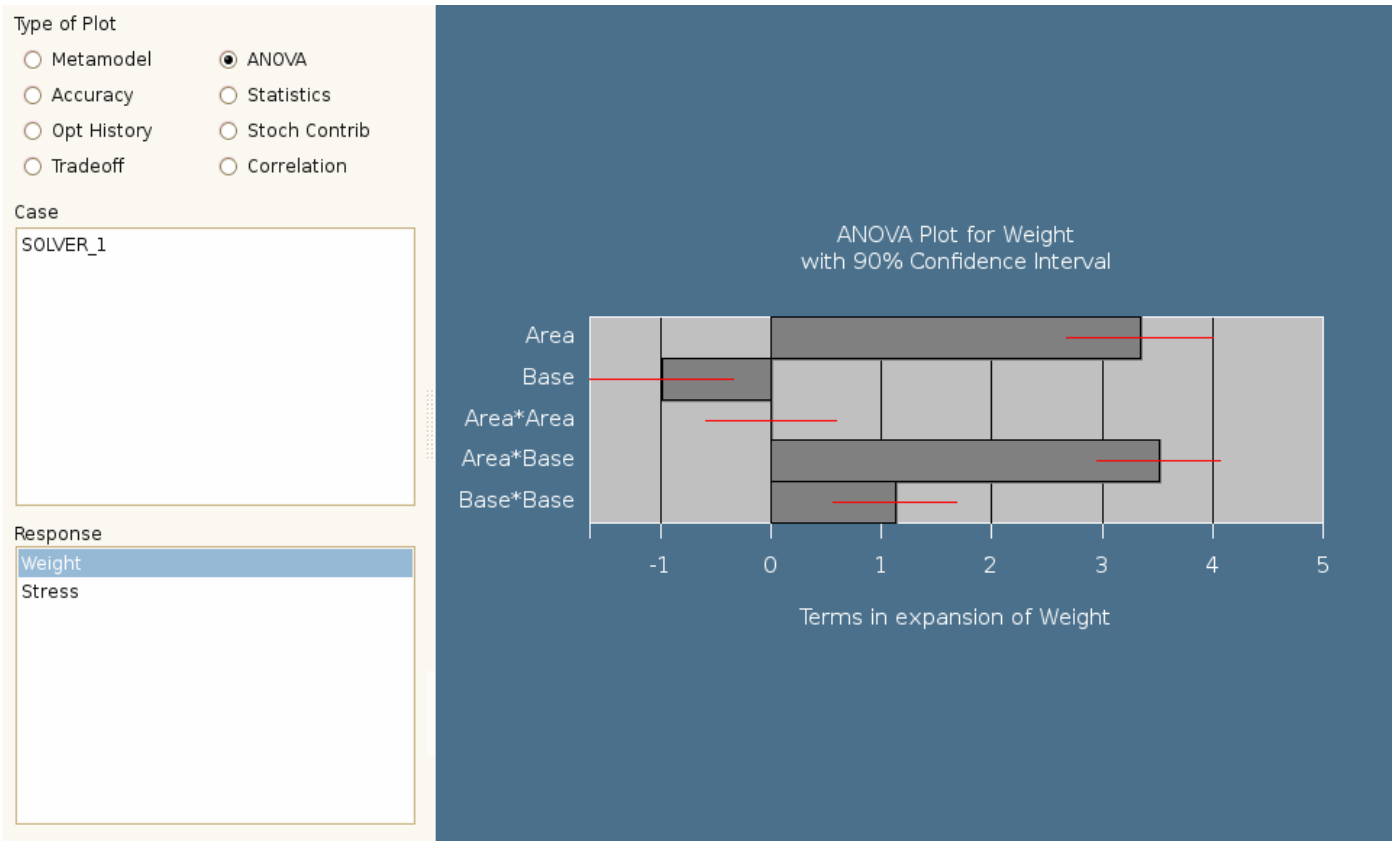


Figure 18-10: ANOVA plot in View panel in LS-OPT*ui*

18.6 Histograms

Histograms of the variables, dependents, responses, and composites are available.

Either the simulation results directly or the metamodels together with the statistical distribution of the variables can be used to construct the histogram. 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 histogram. The user can control the number of points in this Monte Carlo simulation; the default value should however suffice for most cases. 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.

The histogram panel is shown in Figure 18-11.

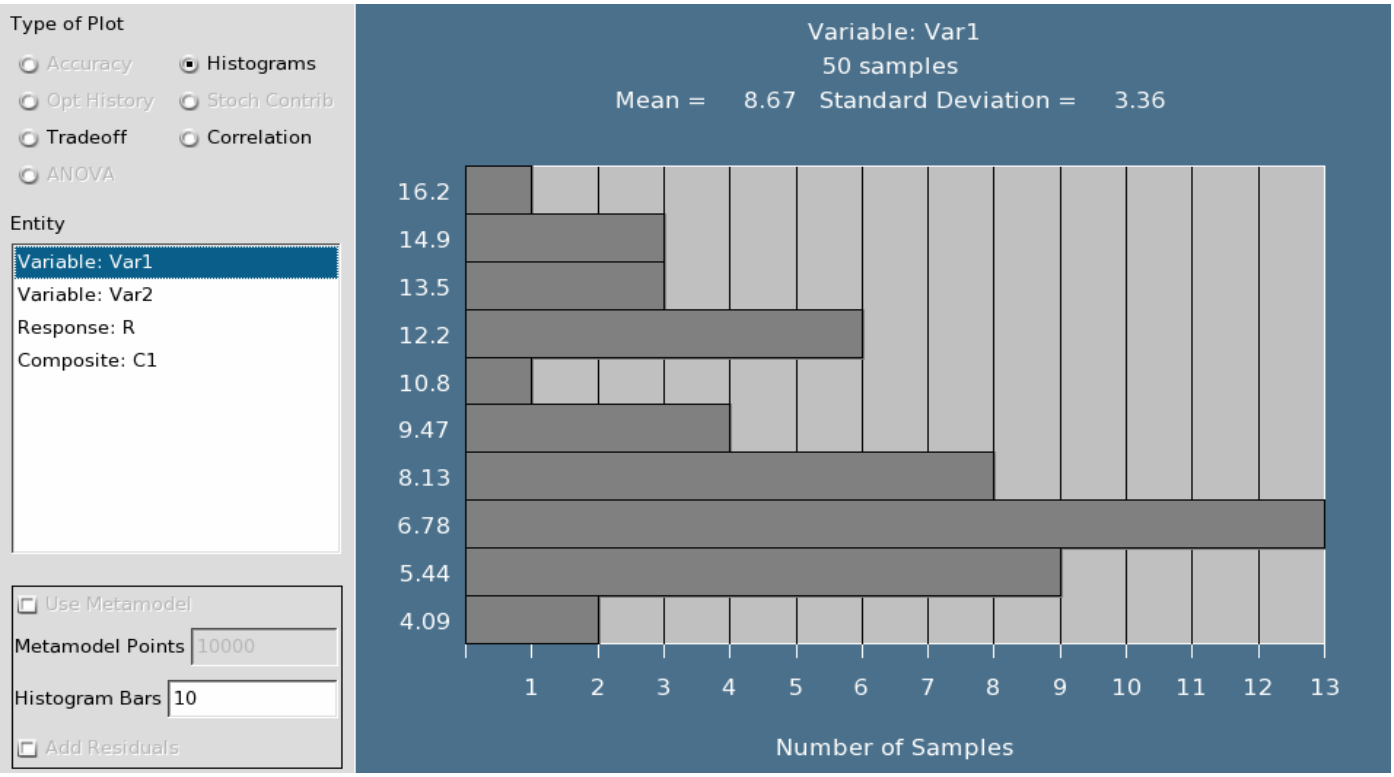


Figure 18-11 Histogram plot

18.7 Stochastic Contribution

The stochastic contribution of the variables to the variance of the responses and composites (see Section 6.7) 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.

The stochastic contribution panel is shown in Figure 18-12.

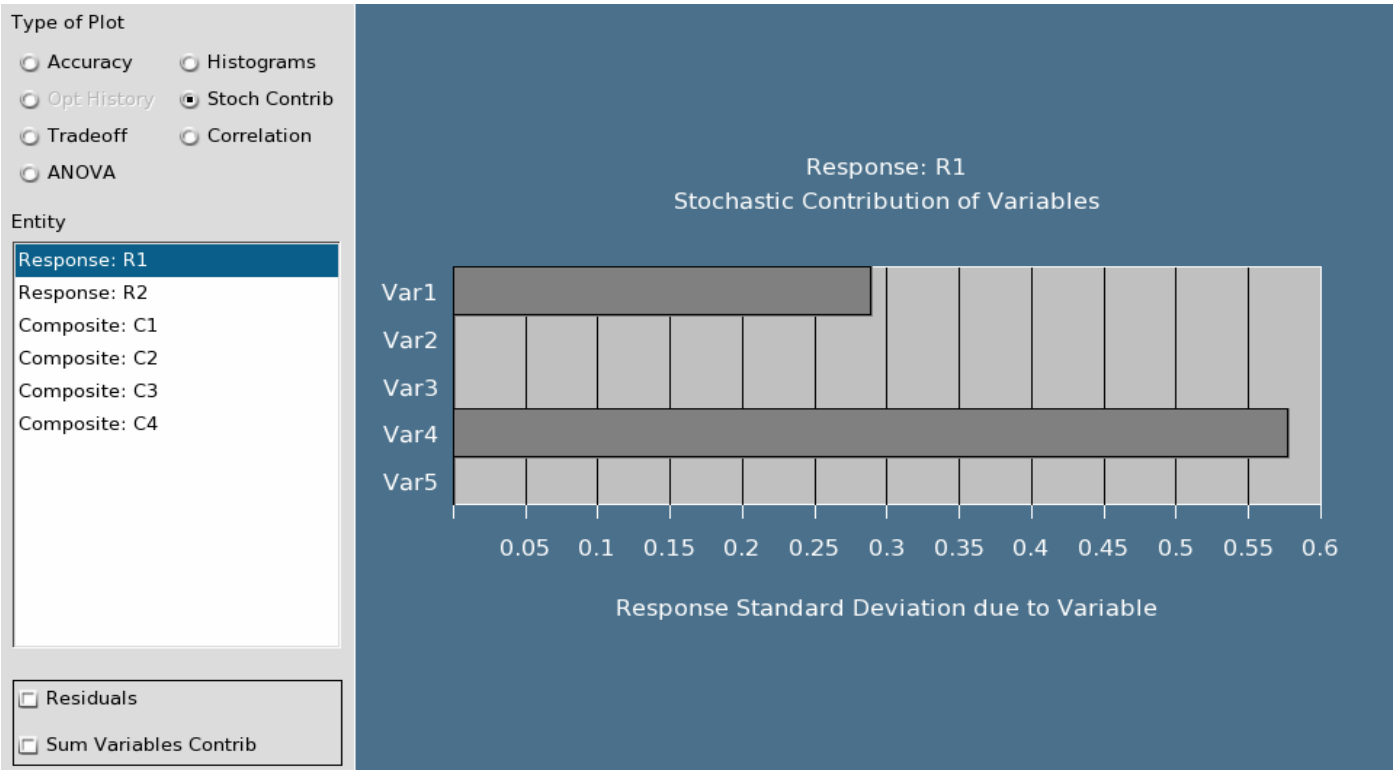


Figure 18-12 Stochastic Contribution plot

18.8 Covariance and Correlation

Both the covariance and the coefficient of correlation of the responses and composites with respect to the design variables can be displayed.

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 this Monte Carlo simulation; the default value should however suffice for most cases.

The plots can be used to estimate the stochastic contribution for an analysis without a metamodel.

The covariance panel is shown in Figure 18-13.

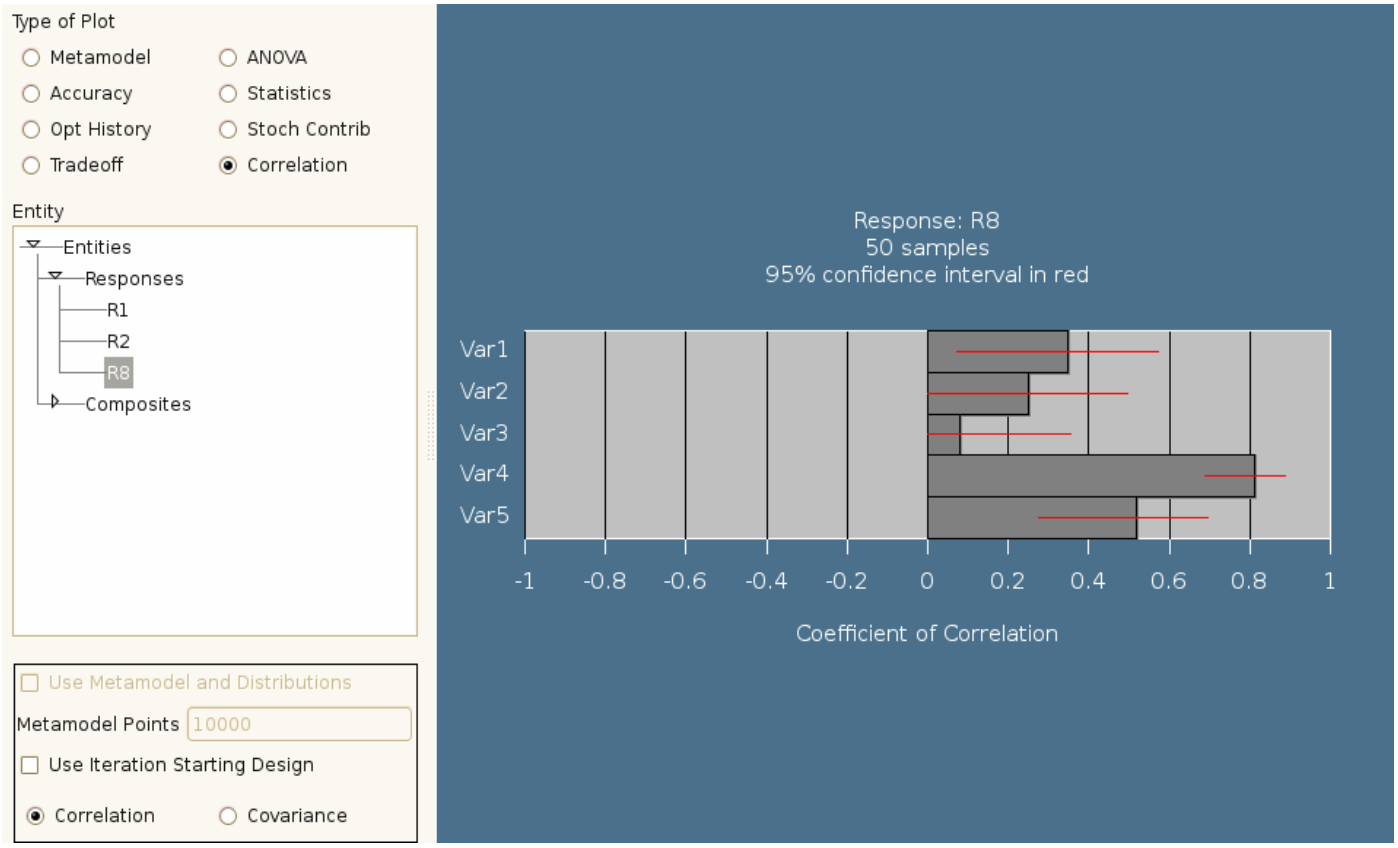


Figure 18-13 Coefficient of Correlation plot

18.9 Plot generation

Plots can be generated in LS-OPT*ui* by selecting File>Export. The current supported format is postscript, both color and monochrome, either to a device or file.

18.10 References

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

19. Applications of Optimization

This chapter provides a brief description of some of the applications of optimization that can be performed using LS-OPT. It should be read in conjunction with Chapter 22, the Examples chapter, where the applications are illustrated with practical examples.

19.1 Multidisciplinary Design Optimization (MDO)

The MDO capability in LS-OPT implies that the user has the option of assigning different variables, experimental designs and job specification information to different solvers or disciplines. The directory structure change that has been incorporated in this version, separates the number of experiments that needs to be run for each solver by creating separate `Experiments`, `AnalysisResults`, `DesignFunctions` and `ExtendedResults` files in each solver directory.

Command file syntax:

`mdo` *`mdotype`*

The only *`mdotype`* available is `mdf`, or multidisciplinary feasible.

19.1.1 Command file

All variable definitions are defined first, as when solving non-MDO problems, regardless of whether they belong to all disciplines or solvers. This means that the variable starting value, bounds (minimum and maximum) and range (sub-region size) are defined together. If a variable is *not* shared by all disciplines, however, i.e., it belongs to some but not all of the disciplines (solvers), then it is flagged using the syntax `local variable_name`. At this stage, no mention is made in the command file to which solver(s) the particular variable belongs. This reference is made under the solver context, where the syntax `Solver variable variable_name` is used.

See the examples in Section 22.6 for the command file format.

19.2 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 Variables panel (see Figure 11-1) by toggling the required variables from 'Minimize' to 'Maximize'.

19.3 Reliability-based design optimization (RBDO)*

LS-OPT has a reliability-based design capability based on the computation of the standard deviation of any response. The theoretical concerns are discussed in Section 5.5.

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) or First Order Reliability Method (FORM) method. 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:

1. Statistical distributions associated with the design variables
2. 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.

The current GUI support is limited to what is available for deterministic design optimization and Monte Carlo analysis.

Command file syntax:

```
probability upper bound constraint 'con_name' upper_bound
probability lower bound constraint 'con_name' lower_bound
iterate number_of_iterations
```

An example is given in Section 22.2.11.

20. Optimization Algorithm Selection and Settings

This chapter describes the parameter settings for the domain reduction and LFOPC methods that are used in LS-OPT. The default parameters for both the domain reduction scheme and the core optimization algorithm (LFOPC) should be sufficient for most optimization applications. The following sections describe how to modify the default settings. These can only be modified using the command language.

20.1 Selecting an optimization methodology

There are two optimization methods available namely the Sequential Response Surface Method (SRSM) and the Genetic Algorithm. The syntax is as follows:

Command file syntax:

```
Optimization method [srsm|genalg]
```

SRSM is the default. Note that the choice of Genetic Algorithm methodology may require a large number of simulations.

20.2 Selecting an optimization algorithm

The two optimization algorithms that can be used with the SRSM approach are the LFOPC and the Genetic Algorithm. The syntax is as follows:

Command file syntax:

```
Optimization algorithm [lfopc|genalg]
```

LFOPC is the default.

20.3 Subdomain reduction

20.3.1 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, for SRSM, also on the occurrence of oscillation (see theory in Section 4.6).

The following parameters can be adjusted (refer also to Section 4.6). A suitable default has been provided for each parameter but the user should not find it necessary to change any of these parameters.

Table 20.3-1: Subdomain parameters and default values

Item	Parameter	Default	
		SRSM	SRSM (NN)
`objective`	Tolerance on objective function accuracy ε_f	0.01	0.01
`design`	Tolerance on design accuracy ε_x	0.01	0.01
`stoppingtype`	`and: objective` *and* `design;` `or: objective` *or* `design`	`and`	`and`
`psi`	γ_{pan}	1.0	1.0
`gamma`	γ_{osc}	0.6	1.0
`eta`	Zoom parameter η	0.6	0.75
`rangelimit`	Minimum range	0.0	0.0
`repeatlimit`	Limit on number of times solution is repeated (SRS only)	5	5

* Applied when the design has not changed.

Command file syntax:

```
iterate param parameter_identifier value
iterate param rangelimit 'variable_name' value
```

The iterative process is terminated if the following convergence criteria become active:

$$\left|\frac{f^{(k)}-f^{(k-1)}}{f^{(k-1)}}\right|<\varepsilon_f$$

and/or

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

where $\boldsymbol{x}$ refers to the vector of design variables, $\boldsymbol{d}$ is the size of the design space, f denotes the value of the objective function and, (k) and $(k-1)$ refer to two successive iteration numbers. The `stoppingtype` parameter is used to determine whether (`and`) or (`or`) will be used, e.g.

```
iterate param design 0.001
iterate param objective 0.001
iterate param stoppingtype or
```

implies that the optimization will terminate when either criterion is met.

The range limit can be used to specify the minimum size of the region of interest. This is not a stopping criterion so that the solver will still continue to iterate until any of the other stopping criteria are met.

An application of the range limit is to maintain a constant tolerance on the random variables.

Command file syntax:

```
iterate param rangelimit variable_name value
```

Example:

```
iterate param rangelimit 'thickness 1' 0.5
iterate param rangelimit 'Radius' 10
```

20.3.2 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.

Command file syntax:

```
iterate param reset range iteration iteration_number
```

Example:

```
iterate param reset range iteration 3
```

The point selection of iteration 3 will be conducted in the initial range around the most recent optimum point. Full adaptivity will be applied again starting with iteration 4.

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.

Command file syntax:

```
iterate param adapt off iteration iteration_number
```

Example:

```
iterate param adapt off iteration 3
```

Adaptivity will be applied up to the second iteration. Therefore iterations 3 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.

20.4 Verification run

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.

Command file syntax:

```
iterate noverify
```

20.5 Setting parameters in the LFOPC optimization 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 are only available in the command input file.

Table 20.5-1: LFOPC parameters and default values

Item	Parameter	Default value	Remark
`mu`	Initial penalty value μ	1.0E+2	
`mumax`	Maximum penalty value $\mu_{\max}$	1.0E+4	1
`xtol`	Convergence tolerance ε_x on the step movement	1.0E-8	2
`eg`	Convergence tolerance ε_f on the norm of the gradient	1.0E-5	2
`delt`	Maximum step size δ	See remark	3
`steps`	Maximum number of steps per phase	1000	1
`print`	Printing interval	10	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 value of `steps` must then be increased as well.

2. The optimization is terminated when either of the convergence criteria becomes active that is when

$$\|\Delta(\boldsymbol{x})\| < \varepsilon_x$$

or

$$\|\nabla f(\boldsymbol{x})\| < \varepsilon_f$$

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

4. If `print` = `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 `print` < 0.

Command file syntax:

`lfop param` *`parameter_identifier value`*

Example:

```
lfop param eg 1.0e-6
```

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 a set of random points.

20.6 Setting parameters for metamodel-based optimization strategies

There are three recommended strategies for automating the metamodel-based optimization procedure. These strategies apply to the tasks: Metamodel-based Optimization and RBDO.

20.6.1 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.

Example:

```
solvers 1
responses 1
variables 2
 Variable 'tbumper' 3
  Lower bound variable 'tbumper' 1
  Upper bound variable 'tbumper' 5
 Variable 'thood' 1
  Lower bound variable 'thood' 1
  Upper bound variable 'thood' 5
Optimization Method SRSM
 solver dyna960 '1'
  solver command "ls971_single"
  solver input file "main.k"
  solver order RBF
  solver experiment design space_filling
```

```
   solver number experiments 15
  response 'HIC' 1 0 "BinoutResponse -res_type Nodout  -cmp HIC15 -gravity 9810.00000
-units S -id 432 "
 objectives 1
 objective 'HIC' 1
constraints 0
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype or
 iterate 1
STOP
```

Required settings:

1. Use RBF networks or FF neural networks. RBF networks are much quicker and in some cases more accurate than FF networks.
2. Select the number of sampling points.
3. Since there is only one iteration, *unset* the selection for "First iteration Linear, *D*-Optimal".
4. Adjust the iteration limit in the GUI Run panel to 1.

The following options should default to the settings indicated:

5. Space Filling sampling.

The GUI settings are also explained in Figure 20-1.

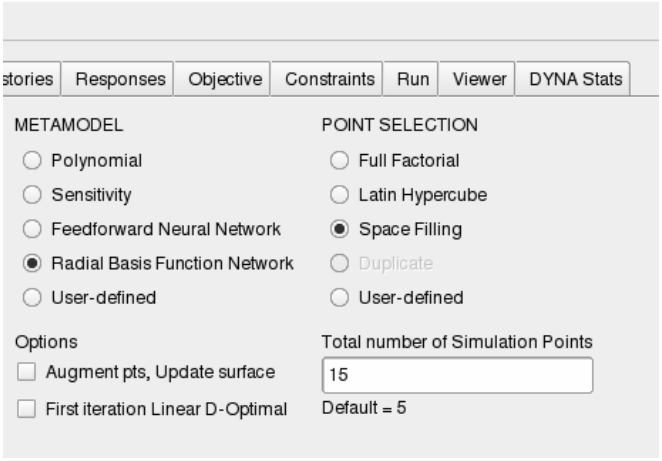


Figure 20-1: Required settings in Sampling panel for Single Stage strategy

20.6.2 Sequential strategy

In this approach, sampling is done sequentially. A small number of points is chosen for each iteration and multiple iterations are requested. The approach has the advantage that the iterative process can be stopped as soon as the metamodels or optimum points have sufficient accuracy. It was demonstrated in Reference [16] that, for Space Filling, the Sequential approach had similar accuracy compared to the Single Stage approach,

i.e. 10 × 30 points added sequentially is almost as good as 300 points. Therefore both the Single Stage and Sequential Methods are good for design exploration using a surrogate model. For instance when constructing a Pareto Optimal Front, the use of a Single Stage or Sequential strategy is recommended in lieu of a Sequential strategy with domain reduction (see Section 20.6.3).

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.

Example:

```
Optimization Method SRSM
 solver dyna960 '1'
  solver command "ls971_single"
  solver input file "main.k"
  solver order RBF
  solver experiment design space_filling
   solver update doe
   solver alternate experiment 1
  response 'HIC' 1 0 "BinoutResponse -res_type Nodout  -cmp HIC15 -gravity 9810.00000
-units S -id 432 "
objectives 1
 objective 'HIC' 1
constraints 0
iterate param design 0.01
 iterate param objective 0.01
 iterate param adapt off iteration 1
 iterate param stoppingtype or
 iterate 3
STOP
```

Required settings:

1. Choose either RBF networks or FF neural networks. RBF networks are much quicker and in some cases more accurate than FF networks.
2. Adjust the iteration limit.
3. Set the Advanced option in the Run panel named “Freeze range from iter” to 1. This selection will ensure that the region of interest remains the full design space from iteration to iteration. (`iterate param adapt off iteration 1`)

The following options should default to the settings indicated:

4. Space Filling sampling.
5. The first iteration is Linear D-Optimal.
6. Use adaptive sampling. This implies that the positions of points belonging to previous iterations are taken into account when choosing new Space Filling points. Metamodels are also built using all available points, including those of previous iterations. The GUI check box is “Augment points, update surface”.
7. Choose the number of points per iteration to not be less than the default for a linear approximation $(1.5(n+1)+1)$.

The GUI settings are also explained in Figure 20-2.

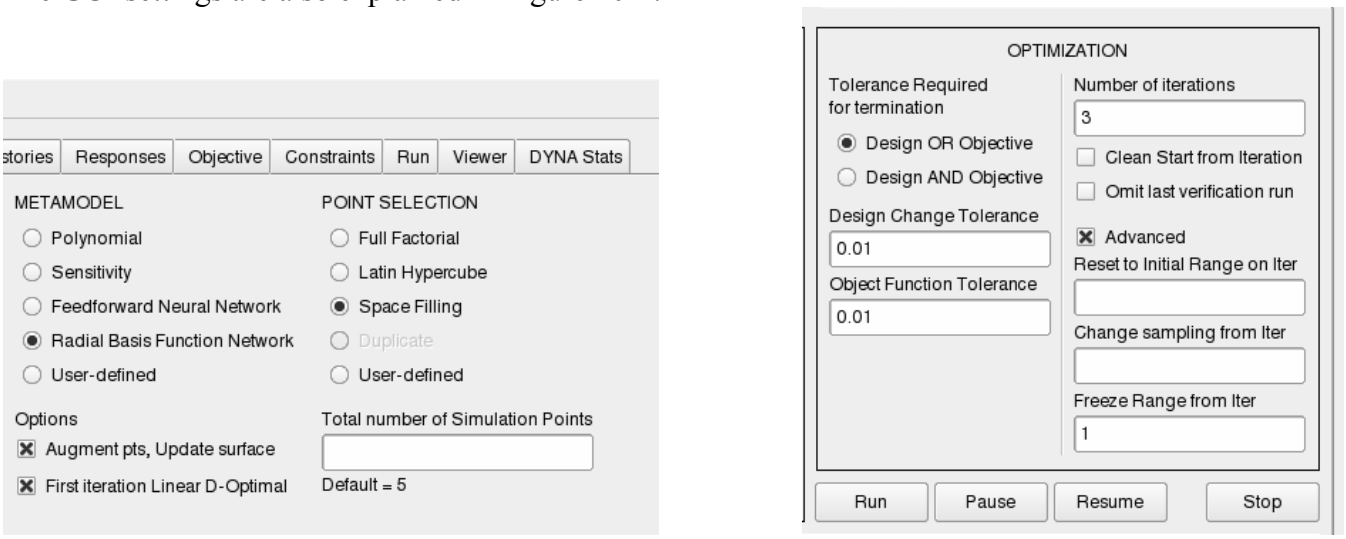


Figure 20-2: Required settings in Sampling panel (left) and Run panel (right) for sequential strategy without domain reduction

20.6.3 Sequential strategy with domain reduction

This approach is the same as that in 20.6.2 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 bound 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 4.7.2 *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 4.6). 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.*

Example:

```
Optimization Method SRSM
solver dyna960 '1'
  solver command "ls971_single"
  solver input file "main.k"
  solver order RBF
  solver experiment design space_filling
   solver update doe
   solver alternate experiment 1
  response 'HIC' 1 0 "BinoutResponse -res_type Nodout  -cmp HIC15 -gravity 9810.00000
-units S -id 432 "
objectives 1
 objective 'HIC' 1
```

```
constraints 0
iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype or
 iterate 3
STOP
```

Required settings:

1. Choose either RBF or FF neural networks. RBF networks are much quicker than FF networks. RBF networks have shown some sensitivity to domain reduction methods [16].
2. Set the iteration limit.

The following options should default to the settings indicated:

3. Space Filling sampling.
4. Choose the number of points per iteration to not be less than the default for a linear approximation $(1.5(n+1)+1)$.
5. Check the box for the first iteration to be Linear D-Optimal.
6. Use adaptive sampling. This implies that the positions of points belonging to previous iterations are taken into account when choosing new Space Filling points. Metamodels are also built using all available points, including those of previous iterations.

In the GUI, the sampling panel setting is the same as in Figure 20-2 (left). However, the "Freeze range from Iter" option in the Run panel is *not* used.

Sequential Response Surface Method (SRSM)
SRSM is the original LS-OPT automation strategy 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 4.6). 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 5.3).

Example:

```
Optimization Method SRSM
solver dyna960 '1'
  solver command "ls971_single"
  solver input file "main.k"
  solver order linear
  solver experiment design dopt
response 'HIC' 1 0 "BinoutResponse -res_type Nodout  -cmp HIC15 -gravity 9810.00000  -
units S -id 432 "
objectives 1
 objective 'HIC' 1
constraints 0
iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype or
 iterate 3
```

```
STOP
```

Required settings:

1. Set the iteration limit in the Run panel.

The following options should default to the settings indicated:

2. Linear polynomial
3. *D*-optimal sampling
4. Default number of sampling points (see Table 2.2-1).

20.7 Setting parameters in the Genetic algorithm

The default parameters in GA should therefore be adequate for most problems. However, if the user needs to explore different methods, the following parameters may be set for GA. These are only available in the command input file.

Table 20.7-1: GA parameters and default values

Item	Parameter	Default value	Type	Remark
`popsize`	Population size	30/100	Integer	1
`generation`	Number of generations	100/250	Integer	1
`selection`	Selection operator: Tourn=1, Roullette=2, SUS=3	1	Integer	2
`Tourn Size`	Tournament size for tournament selection operator	2	Integer	2
`Elitism`	Switch elitism for single objective GA	1	Integer	
`NumElites`	Number of elites passed to next generation	2	Integer	
`Encoding variable`	Type of encoding for a variable: Binary=1, Real=2	2	Integer	2
`Numbits variable`	Number of bits assigned to a binary variable	15	Integer	2
`Binary crossover type`	Type of binary crossover: Single point=1, Uniform=2	1	Integer	
`Binary crossover probability`	Binary crossover probability	1.00	Real	
`Real crossover type`	Type of real crossover: SBX=1, BLX=2	1	Integer	
`Real crossover probability`	Real crossover probability	1.00	Real	
`BLX alpha param`	Value of α for BLX operator	0.5	Real	
`Real crossover distribution index`	Distribution index for SBX crossover operator	10.0	Real	
`Binary mutation probability`	Mutation probability for binary mutation	1/number of binary digits	Real	
`Real mutation probability`	Mutation probability in real-space	1/number of real variables	Real	
`Real mutation distribution index`	Distribution index for mutation operator	10.0	Real	
`Restart interval`	Frequency of writing restart file for direct GA. For multi-objective problems, this parameter governs the frequency of writing TradeOff files	10	Integer	
`seed`	Random seed		Long	

Remarks:

Command file syntax:

GA parameter *parameter_identifier value*

Example:

```
GA parameter popsize 100
```

For direct GA, the default population size is 30 and number of generations is 100. For SRSM, the default population size is 100 and number of generations is 250.

2. Command file syntax:

Encoding variable *variable_name value*

Example:

```
Encoding variable 'x1' 1
Numbits variable 'x1' 20
```

21. LS-DYNA Results Statistics

Various statistics of the LS-DYNA d3plot results and LS-OPT history data can be computed using LS-OPT for viewing in LS-PREPOST on the FE model. 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 timestep in the history.

A more complete list of the statistics that can be computed and visualized is:

- 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.
- Statistics of the results considering the variation of the design variables using the approximations (**metamodels**) created from the LS-DYNA jobs. 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, then 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 6.6 regarding the computation of outliers. This is the p**rocess 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.
- 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.

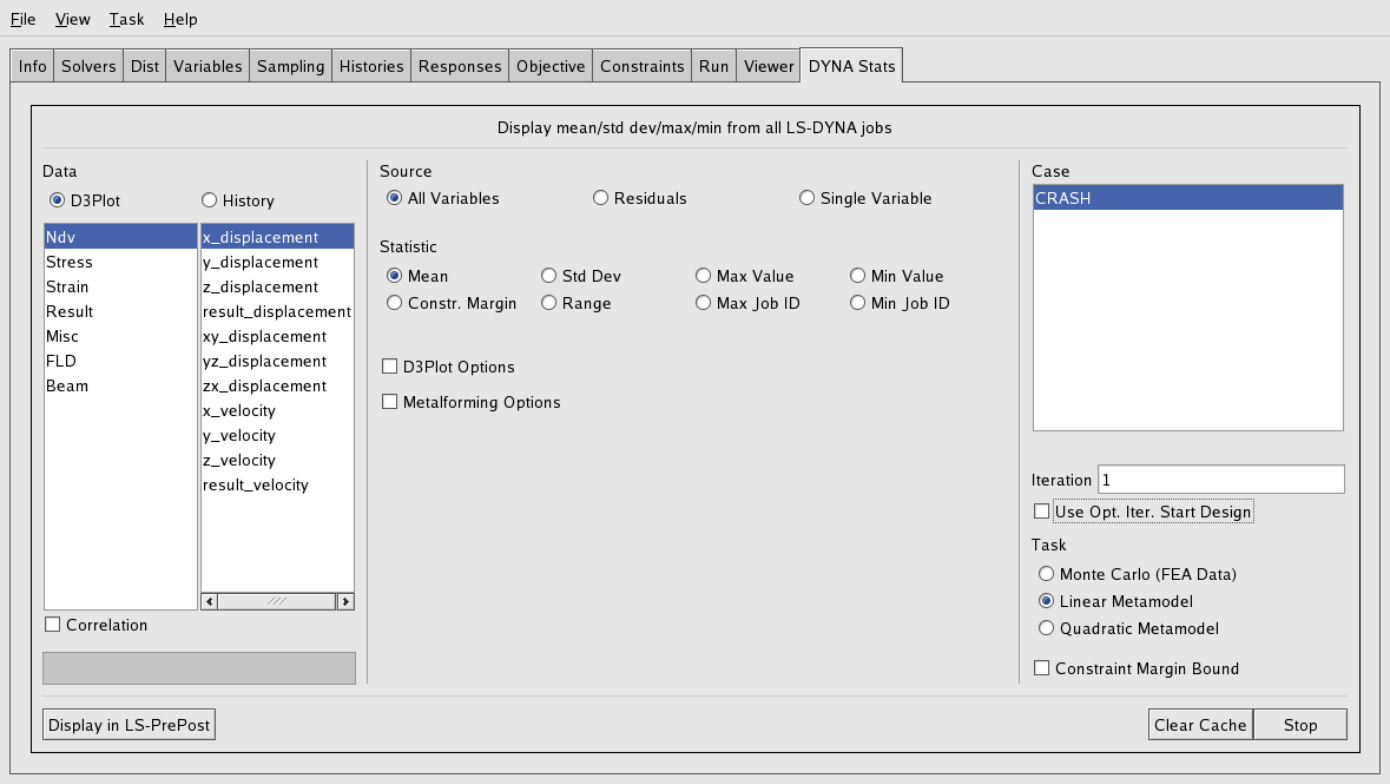


Figure 21-1 Computation of DYNA results statistics. The left hand side of the GUI is dedicated to the selection of the LS-DYNA result; the center is dedicated to the required statistics; and the right hand side is used for infrequently used options such as the solver, the task type, and the constraint margin bound.

21.1 Monte Carlo

The statistic of the responses from a Monte Carlo procedure can be computed.

This Monte Carlo task will calculate:

- 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.
- 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 21.4).

21.2 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 21-2.

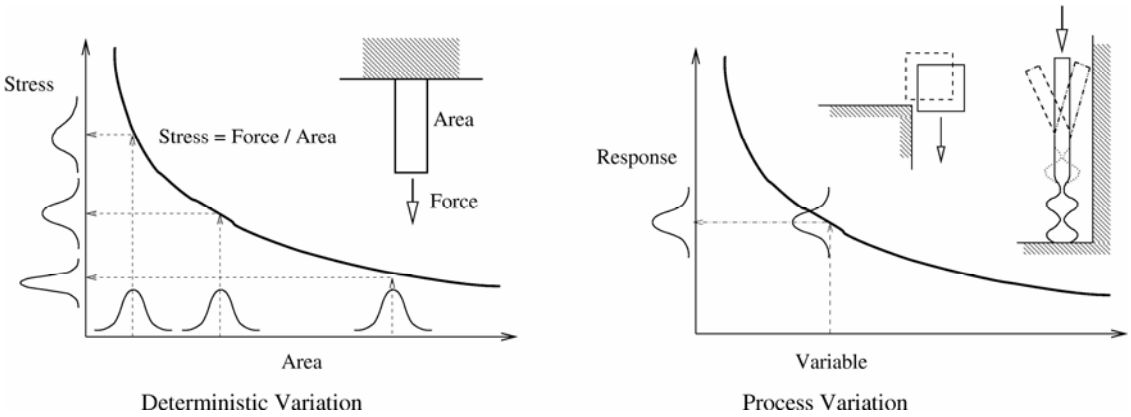


Figure 21-2 Different types of variation that can occur in a structure. The deterministic variation, predicted using the metamodel, is due to changes in the design variable values. The process variation, not associated with change in the design variable values, shows up in the residuals of the metamodel fit.

Metamodels are able to distinguish the process variation because, as shown in Figure 21-3, a metamodel can only predict the effect of the design variables. Process variation, not predictable by the design variables, becomes residuals.

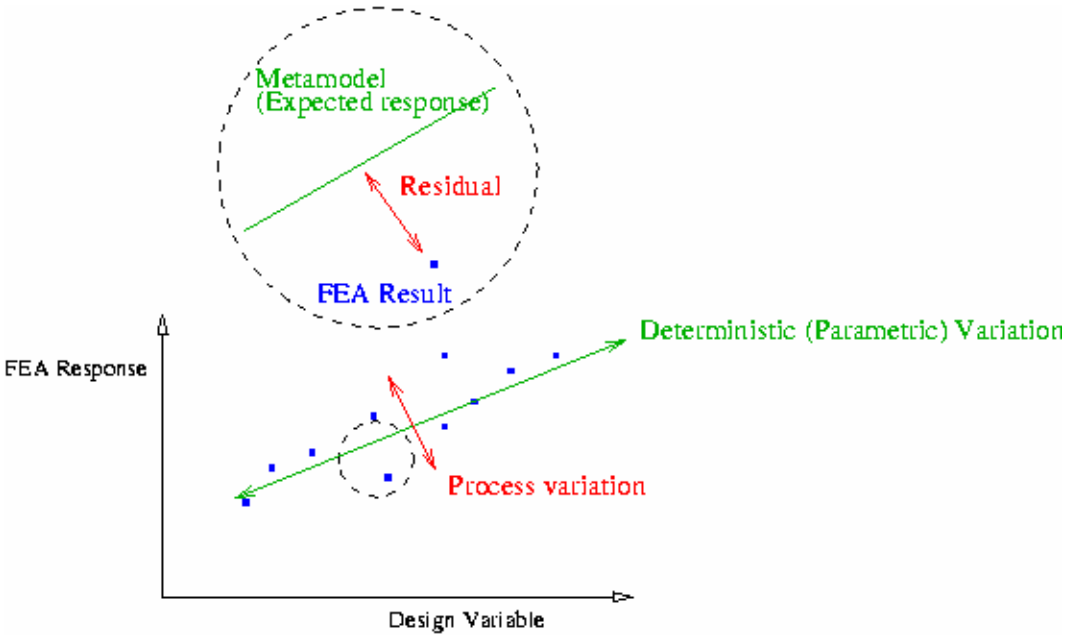


Figure 21-3 Metamodels can be used to distinguish between changes in the results due to the design variable changes and changes due to bifurcations.

The metamodel task will calculate:

- 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)
- 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.
- Stochastic contribution of each individual variable
- 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 21.4).
- 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 6.7. 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 meaningfull 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 6.6 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 number of nodes to be processed are the number of nodes in the model times the number of states times the number of jobs. FLD computations, which requires 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, which means accessing files over a network will be slow.

21.3 Stochastic contribution of a variable (Design sensitivity analysis)

The 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 Section 6.7.

The most important variable, 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.

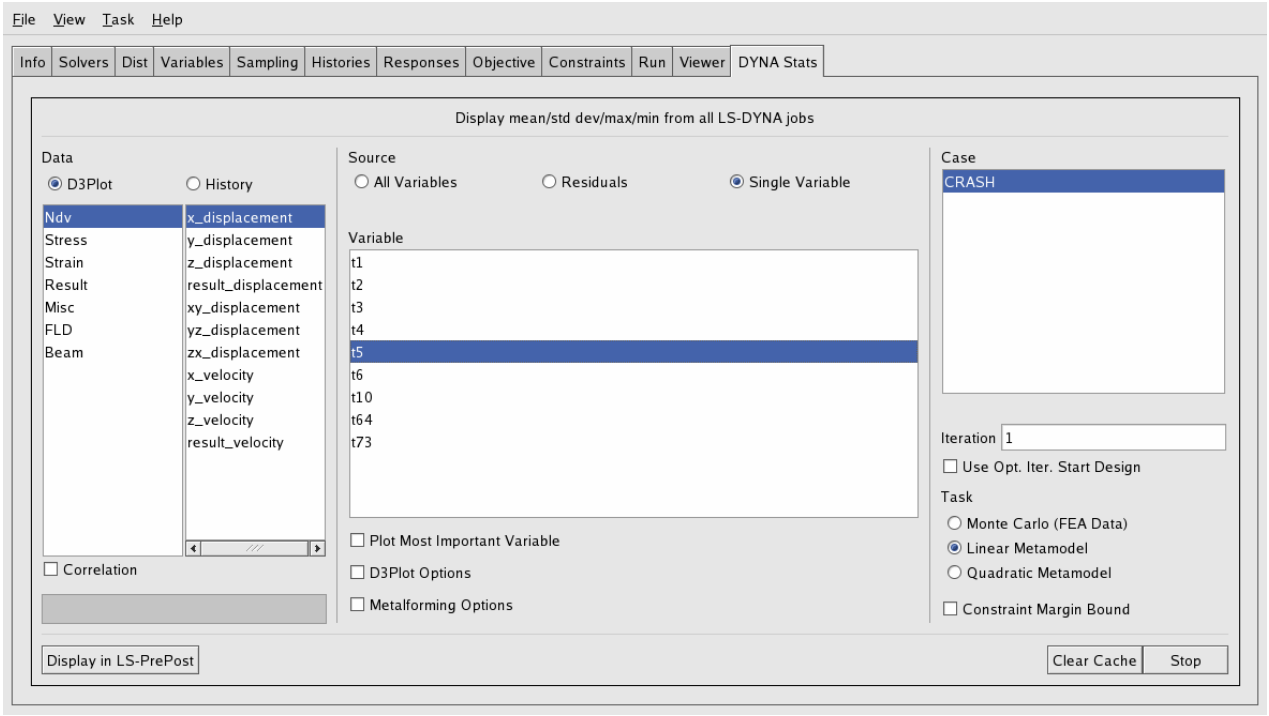


Figure 21-4 Viewing the stochastic contribution of a single variable.

21.4 Safety margin

The safety margin is simply the difference, measured in standard deviations, between the mean response and the constraint bound on the response as shown in Figure 21-5. The bound must therefore be specified when the statistics are computed. Obtaining the safety margin for a different bound requires the recomputation of the statistic.

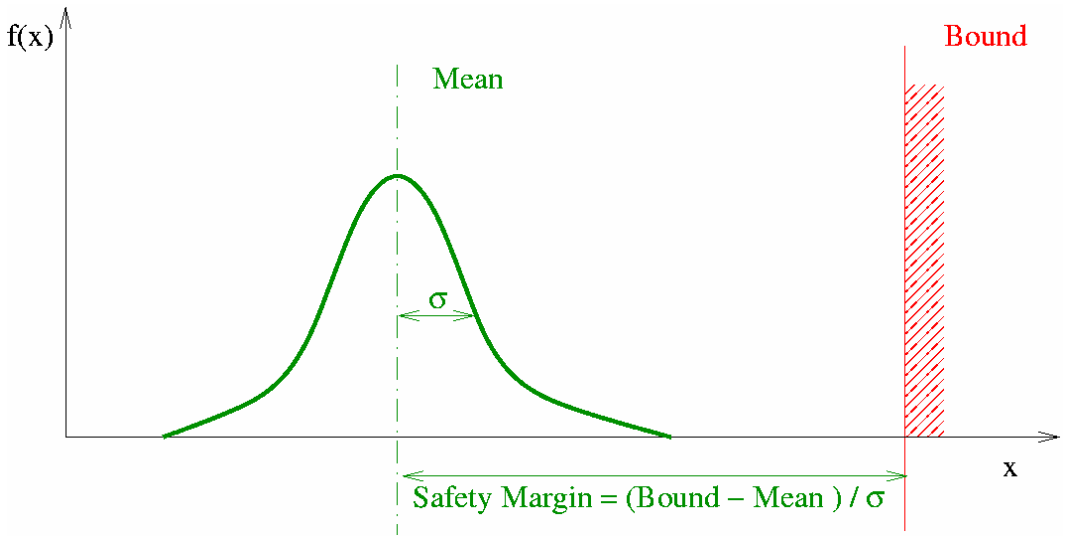


Figure 21-5 The safety margin is the difference, measured in standard deviations, between the mean response and the constraint bound on the response.

21.5 Monte Carlo and metamodel analysis commands

This section gives the commands 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 solver directory with extensions of *.statdb* and *.history*.

The statistic are computed for a single solver and a single iteration.

Command file syntax:

Example:

```
dynastat order linear
```

Command file syntax:

```
dynastat solver 'case_name'
```

```
dynastat iteration iteration_number
dynastat order approx_order
analyze dynastat
analyze dynastat d3plot 'result_type' 'component'
analyze dynastat d3plot 'FLD' 'fld_component' parameters fld_t
fld_n
analyze dynastat d3plot 'FLD' 'fld_component' fld_curve_id
analyze dynastat history 'history_name'
```

Item	Description	Default
case_name	Name of analysis case	The first or only case specified
iteration_number	Iteration number	1
approx_order	linear | quadratic	Do not use a metamodel
result_type	The available result types are listed Appendix A	
component	The available components are listed Appendix A	
fld_t	FLD curve *t* coefficient	
fld_n	FLD curve *n* coefficient	
fld_curve_id	ID in the LS-DYNA file of the FLD curve to be used	
history_name	Name of LS-OPT history	

Example:

```
$ analyze displacement using a metamodel
dynastat solver 'CRASH'
dynastat iteration 1
analyze dynastat
dynastat order linear
$
$ analyze history using a metamodel
dynastat solver 'CRASH'
dynastat iteration 1
dynastat order linear
analyze dynastat history 'nHist'
```

21.6 Correlation

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 displacement or history. Figure 21-6 shows examples of a positive, a negative, and a lack of correlation.

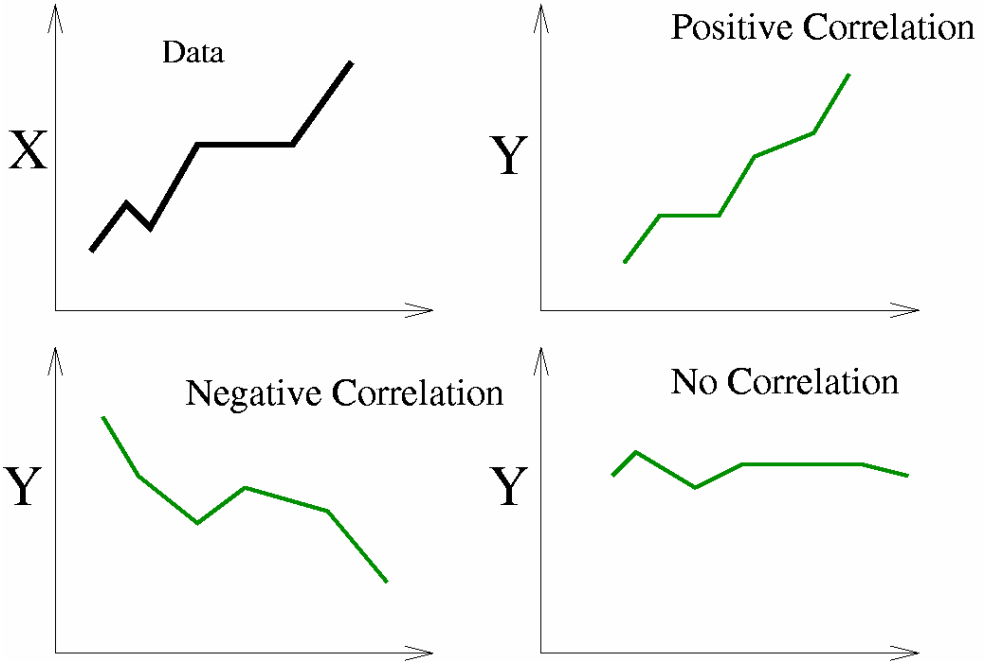


Figure 21-6 Correlation between X, shown in the upper left corner, and different responses Y. Different responses Y with a positive, a negative, and no correlation are shown.

If not enough FE evaluations were conducted, the resulting fringe plot can be visually noisy. 30 or more FE evaluations may be required.

Note that the correlation of history is with respect to a response at a single time instance.

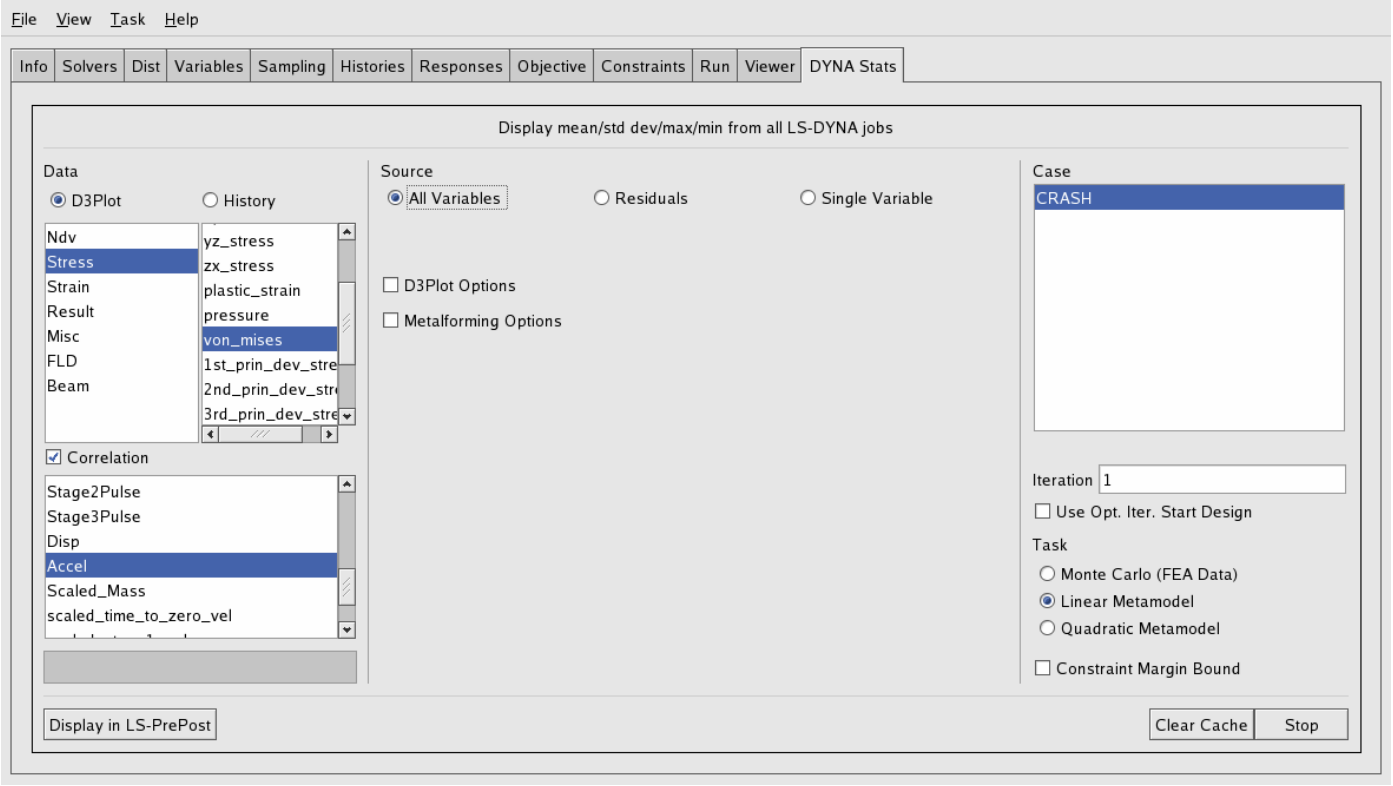


Figure 21-7 Viewing the correlation between an LS-DYNA response and an LS-OPT response. Additionally, the correlation between an LS-OPT history and an LS-OPT response can also be viewed.

Command file syntax:

```
dynastat correlation response 'name'
```

Item	Description
name	Name of response or composite

Example:

```
dynastat correlation response 'node_max'
```

21.7 Visualization in LS-PREPOST

The user can select the LS-PREPOST plot details in LS-OPT (Figure 21-8). The GUI options will reflect whether displacements or history data is being investigated and whether coefficient of correlation results are requested.

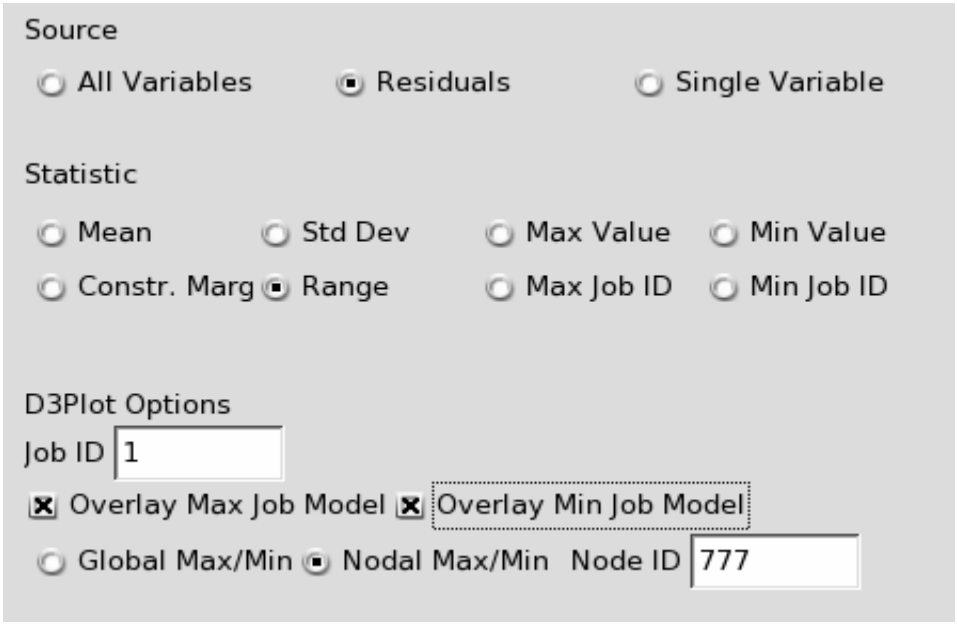


Figure 21-8 The statistics viewing options. The statistics contributed by all the variables, the residuals, and a single variable can be viewed. The statistics will be shown in LS-PREPOST using the FE model from the LS-DYNA job specified using the *Job ID* field. If the residuals are viewed, then the FE models of the jobs containing the maximum and minimum residuals can be overlayed in order to identify bifurcations as described in Section 21.9.

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 overlayed in order to spot bifurcations as described in a later section.

The LS-PREPOST executable must be named *lsprepost*. The LS-PREPOST executable must be newer than December 2003.

21.8 Viewing LS-OPT histories

The LS-OPT histories for all the LS-DYNA run can be viewed simultaneously. See Figure 21-11 for an example.

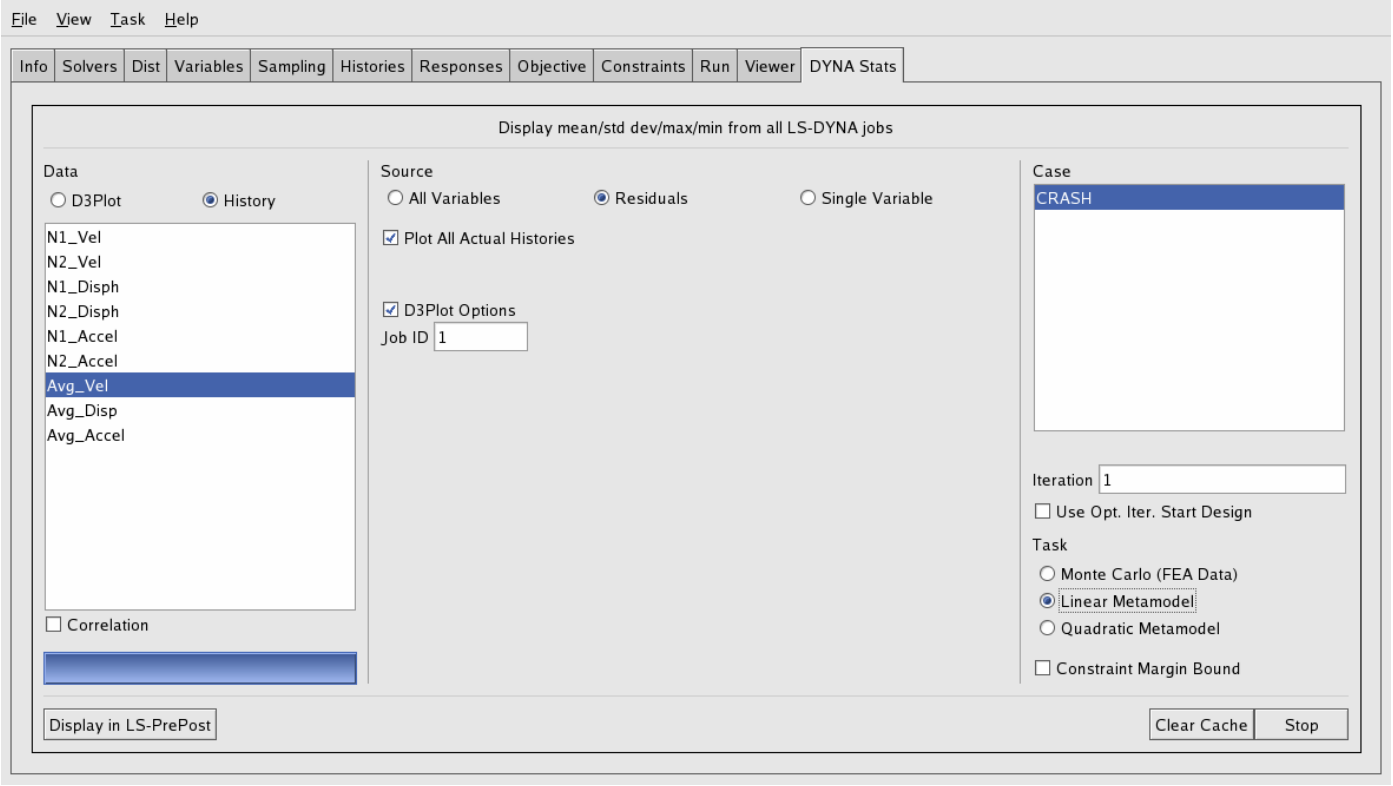


Figure 21-9 Viewing LS-OPT histories.

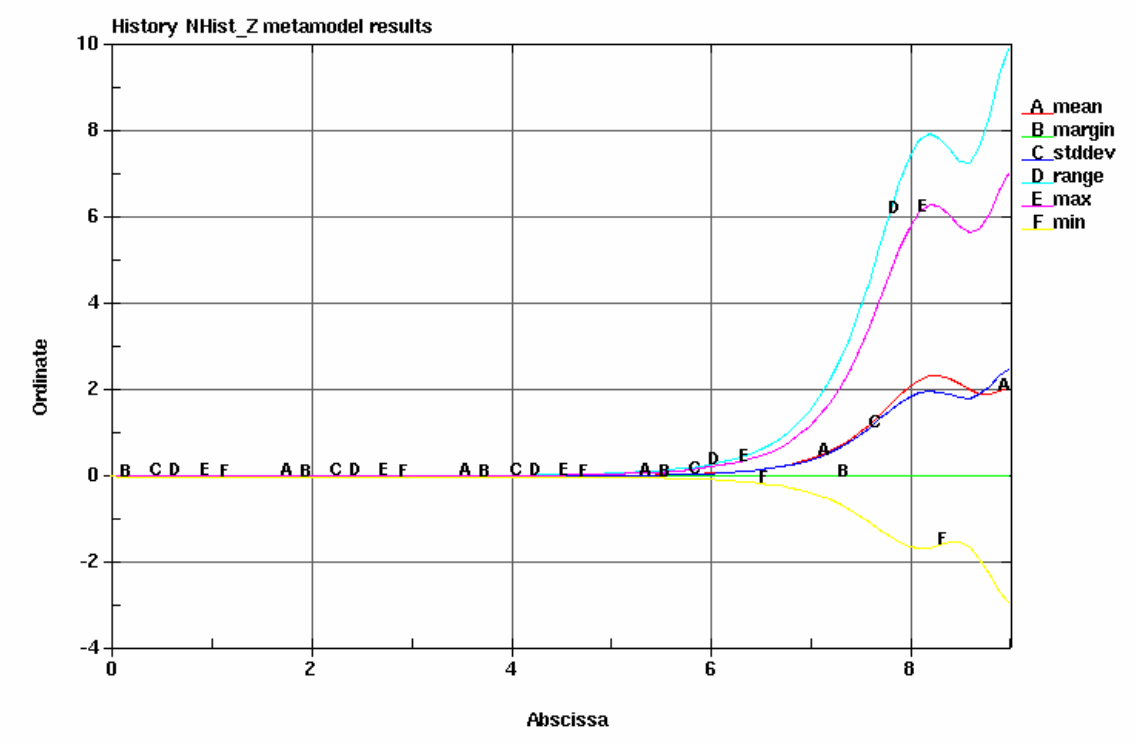


Figure 21-10 Statistics of an LS-OPT history.

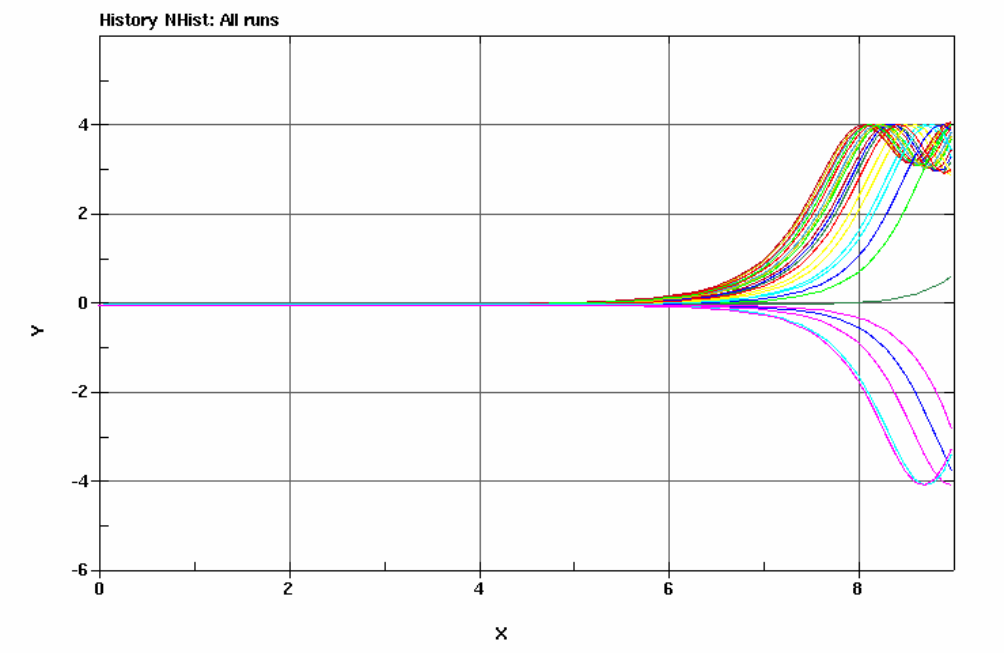


Figure 21-11 The LS-OPT histories of all the LS-DYNA run can be viewed simultaneously.

21.9 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.

21.9.1 Automatic detection

Automatic detection of the LS-DYNA jobs containing the minimum and maximum outlier can be done as shown in Figure 21-8. The GUI the user must select (i) overlay of the FE models containing the maximum and minimum results and (ii) whether the global minimum or the minimum at specific node must be used. Viewing the maximum and minimum job simultaneously allows the bifurcation to be identified. See Figure 21-8 for an example of the resulting LS-PREPOST plot.

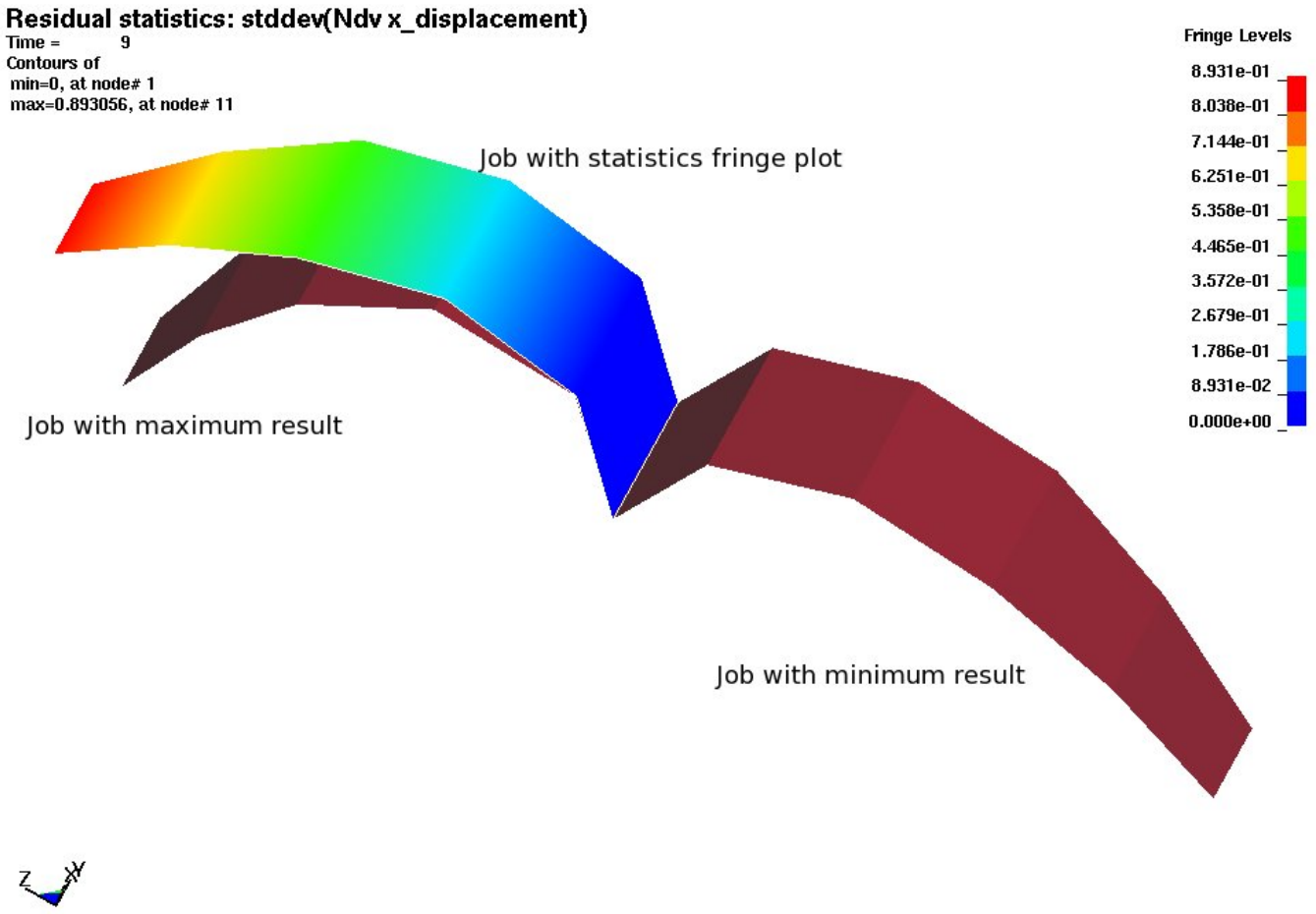


Figure 21-12 Viewing a bifurcation. The structure is a plate that can buckle either left or right. Three FE models are shown, and the two distinctly different solution modes are clearly visible. The creation and display of the plot containing all three models are automated in LS-OPT.

21.9.2 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.

21.10 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 21-13 illustrates. The x, y, and z displacement components do not suffer from this problem.

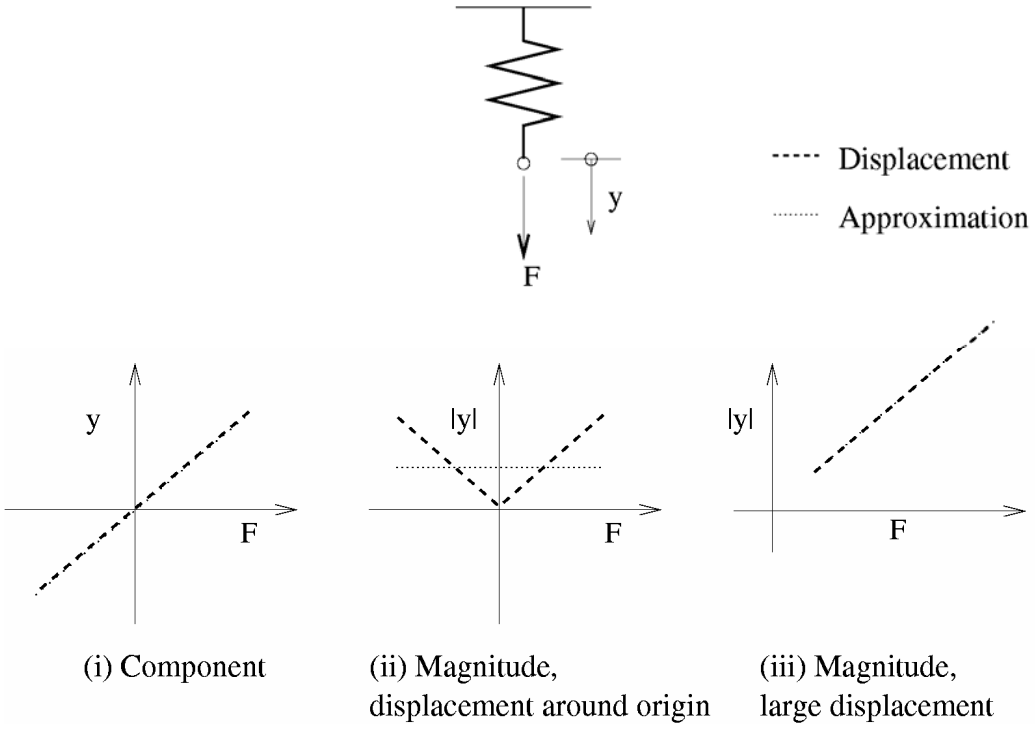


Figure 21-13 Displacement approximation scenarios. The displacement magnitude, being always larger than zero, cannot be approximated accurately around the origin if some of the displacement components can have a negative value.

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 21-14 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.

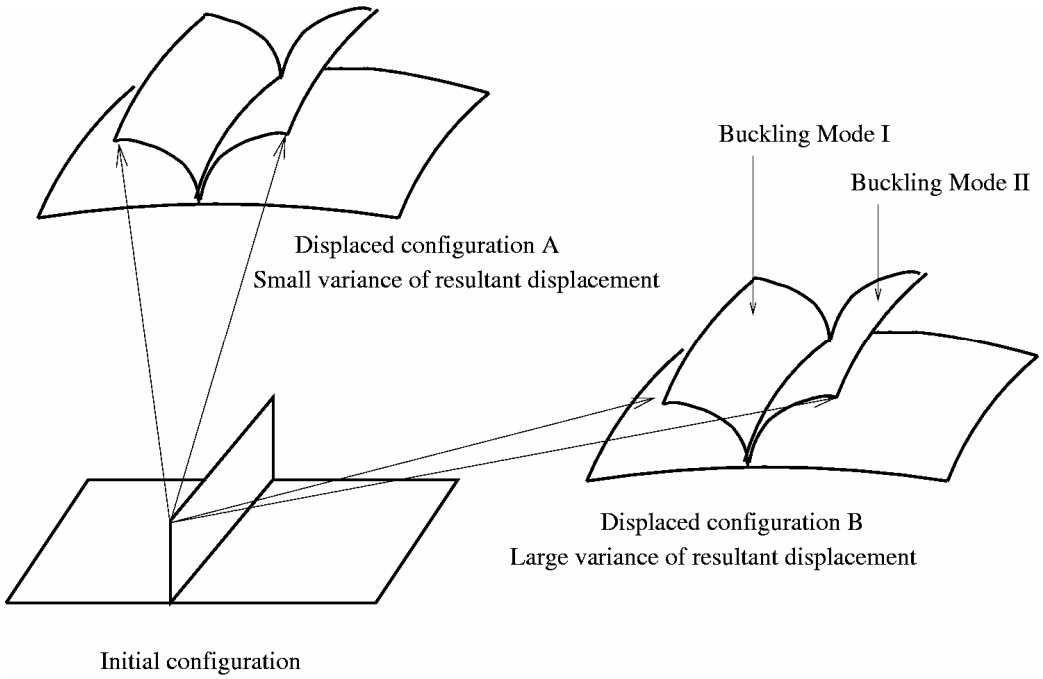


Figure 21-14 The displacement magnitude can depend on the aligment of the flange with the axis. The buckling will be difficult to spot if it is aligned with the position of the axis. For configuration A, the two vectors have nearly the same length, while for configuration B, they clearly have different lengths.

Recommendations:

- Use the x, y, and z displacement components.

21.11 Metalforming options

Metalforming has some special requirements. It is possible to:

- 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:
  i. The adaptivitity will result in the different iterations having different meshes.
  ii. It is more natural in metalforming to consider the results at a specific geometric location than at a specific node.
- 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 parameter similar to that being used in LS-PREPOST.

Command file syntax:

```
dynastat map part
analyze dynastat d3plot 'FLD' 'fld_component' parameters fld_t fld_n
analyze dynastat d3plot 'FLD' 'fld_component' fld_curve_id
```

Item	Description
part	ID of part to be mapped
fld_t	FLD curve *t* coefficient
fld_n	FLD curve *n* coefficient
fld_curve_id	ID in the LS-DYNA file of the FLD curve to be used

Example:

```
dynastat map 8
analyze dynastat 'FLD' 'lower_eps1/fldc' parameters 0.8 0.21
```

EXAMPLES

22. Example Problems

22.1 Two-bar truss (2 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.
- A trade-off study is performed.
- The design optimization process is automated.

22.1.1 Description of problem

This example problem as shown in Figure 22-1 has one geometric and one element sizing variable.

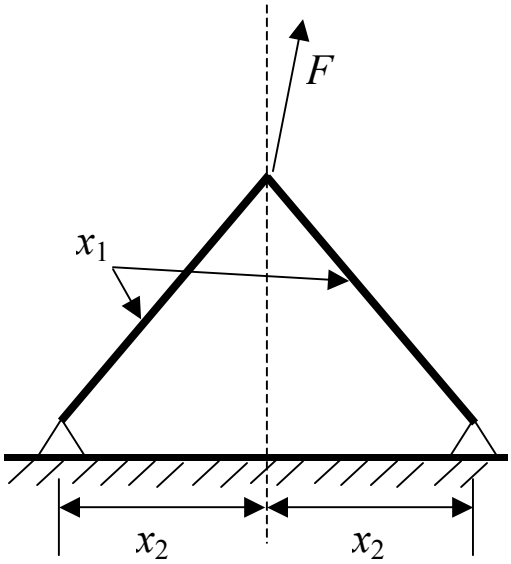


Figure 22-1: The two-bar truss example

The problem is statically determinate. The forces on the members depend only on the geometric variable.

Only one load case is considered: $\boldsymbol{F} = (F_x, F_y) = (24.8kN, 198.4kN)$.

There are two design variables: x_1 the cross-sectional area of the bars, and x_2 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.

$$f(x) = C_1 x_1 \sqrt{1 + x_2^2} \tag{22.1-1}$$

The stresses in the members are constrained to be less than 100 MPa.

$$\sigma_1(x) = C_2 \sqrt{1 + x_2^2} \left(\frac{8}{x_1} + \frac{1}{x_1 x_2} \right) \le 1 \tag{22.1-2}$$

$$\sigma_2(x) = C_2 \sqrt{1 + x_2^2} \left(\frac{8}{x_1} - \frac{1}{x_1 x_2} \right) \le 1 \tag{22.1-3}$$

where $C_1 = 1.0$ and $C_2 = 0.124$.

Only the first stress constraint is considered since it will always have the larger value.

The C language is used for the simulation program. The following two programs simulate the weight response and stress response respectively.

gw.c

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define NUMVAR 2

main (int argc, char *argv[])
{
int i, flag;
double x[NUMVAR], val;

    for (i=0; i<NUMVAR; i++) {
      flag = sscanf (argv[i+1], "%lf", &x[i]);
      if (flag != 1)     {
          printf ("Error in calculation of Objective Function\n");
          exit (1);
          }
      }

    val =  x[0] * sqrt(1 + x[1]*x[1]);

    printf ("%lf\n", val);
    fprintf (stderr, "N o r m a l\n");
```

```
    exit (0);
}
```

gs.c

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define NUMVAR 2

main (int argc, char *argv[])
{
int i, flag;
double x[NUMVAR], val;
double x2;

    for (i=0; i<NUMVAR; i++) {
      flag = sscanf (argv[i+1], "%lf", &x[i]);
      if (flag != 1)    {
          printf ("Error in calculation of constraint1\n");
          exit (1);
          }
      }
    x2 = 1 + x[1]*x[1];
    val = 0.124 * sqrt (x2) * (8/x[0] + 1/x[0]/x[1]);

    printf ("%lf\n", val);
    fprintf (stderr, "N o r m a l\n");

    exit (0);
}
```

The UNIX script program `2bar_com` runs the C-programs `gw` and `gss` using the design variable file `XPoint` which is resident in each run directory, as input. For practical purposes, `2bar_com`, `gw` and `gs` have been placed in a directory above the working directory (or three directories above the run directory). Hence the references `../../../2bar_com`, `../../../gw`, etc. in the LS-OPT input file.

Note the output of the string `"N o r m a l"` so that the completion status may be recognized.

2bar_com:

```
../../../gw `cat XPoint` >wt; ../../../gss `cat XPoint` >str
```

The UNIX extraction scripts `get_wt` and `get_str` are defined as user interfaces:

get_wt:

```
cat wt
```

get_str:

```
cat str
```

In Sections 22.1.2 to 22.1.4, a typical semi-automated optimization procedure is illustrated. Section 22.1.5 shows how a trade-off study can be conducted, while the last subsection 22.1.6 shows how an automated procedure can be specified for this example problem.

22.1.2 A first approximation using linear response surfaces

The first iteration is chosen to be linear. The input file for LS-OPT given below. The initial design is located at $\boldsymbol{x}$ = (2.0, 0.8).

```
"2BAR1: Two-Bar Truss: A first approximation (linear)"
$ Created on Wed Jul 10 17:41:03 2002
$
$ DESIGN VARIABLES
$
variables 2
 Variable 'Area' 2
  Lower bound variable 'Area' 0.2
  Upper bound variable 'Area' 4
  Range 'Area' 4
 Variable 'Base' 0.8
  Lower bound variable 'Base' 0.1
  Upper bound variable 'Base' 1.6
  Range 'Base' 1.6
solvers 1
responses 2
$
$ NO HISTORIES ARE DEFINED
$
$
$ DEFINITION OF SOLVER "RUNS"
$
 solver own 'RUNS'
  solver command "../../../2bar_com"
$
$ RESPONSES FOR SOLVER "RUNS"
$
 response 'Weight' 1 0 "cat wt"
 response 'Weight' linear
 response 'Stress' 1 0 "cat str"
 response 'Stress' linear
$
$ NO HISTORIES DEFINED FOR SOLVER "RUNS"
$
$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'Weight' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 1
 constraint 'Stress'
  upper bound constraint 'Stress' 1
$
$ EXPERIMENTAL DESIGN
$
 Order linear
 Experimental design dopt
```

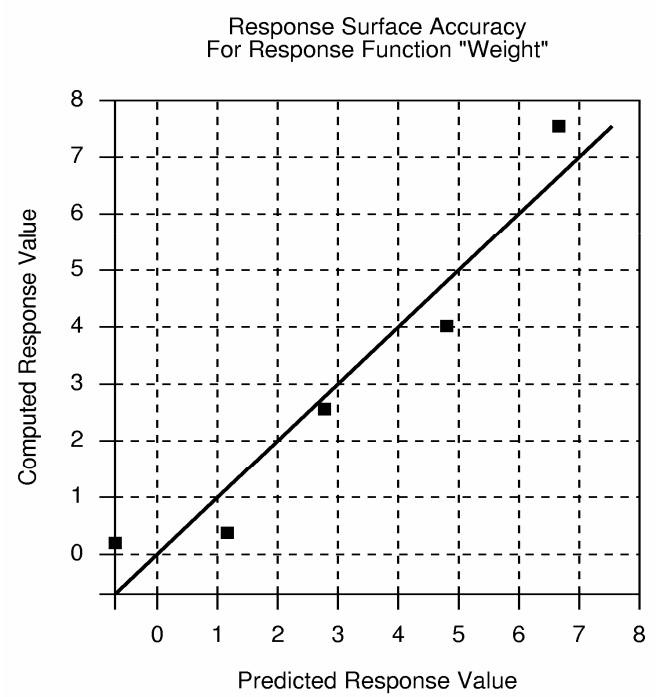
```
 Basis experiment 3toK
 Number experiment 5
$
$ JOB INFO
$
 concurrent jobs 4
 iterate param design 0.01
 iterate param objective 0.01
 iterate 1
STOP
```

The input is echoed in the file `lsopt_input`.
The output is given in `lsopt_output` and in the View panel of LS-OPT*ui*.

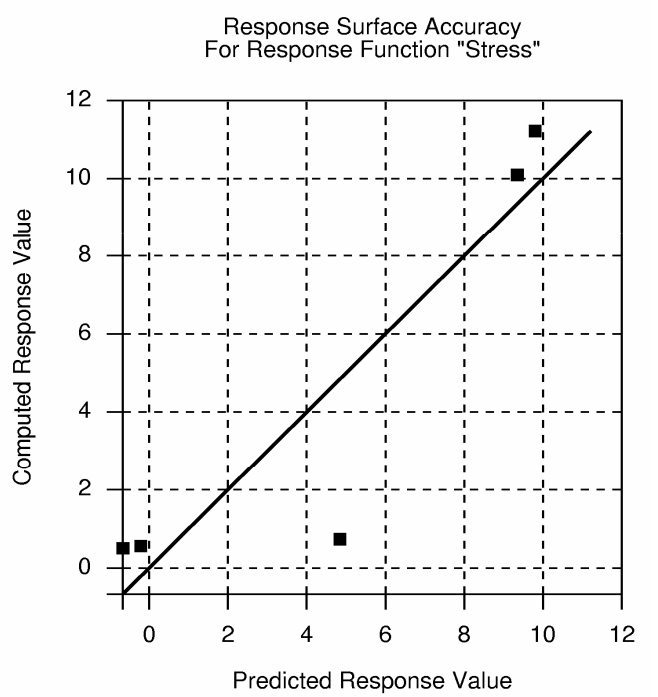
A summary of the response surface statistics from the output file is given:

```
Approximating Response 'Weight' using 5 points (ITERATION 1)
------------------------------------------------------------------
          Global error parameters of response surface
          -------------------------------------------
Linear Function Approximation:
------------------------------
Mean response value           =     2.9413

RMS error                     =     0.7569 (25.73%)
Maximum Residual              =     0.8978 (30.52%)
Average Error                 =     0.7131 (24.24%)
Square Root PRESS Residual    =     2.5054 (85.18%)
Variance                      =     0.9549
R^2                           =     0.9217
R^2 (adjusted)                =     0.9217
R^2 (prediction)              =     0.1426
Determinant of [X]'[X]        =     3.5615

Approximating Response 'Stress' using 5 points (ITERATION 1)
------------------------------------------------------------------
          Global error parameters of response surface
          -------------------------------------------
Linear Function Approximation:
------------------------------
Mean response value           =     4.6210

RMS error                     =     2.0701 (44.80%)
Maximum Residual              =     4.1095 (88.93%)
Average Error                 =     1.6438 (35.57%)
Square Root PRESS Residual    =     3.9077 (84.56%)
Variance                      =     7.1420
R^2                           =     0.8243
R^2 (adjusted)                =     0.8243
R^2 (prediction)              =     0.3738
Determinant of [X]'[X]        =     3.5615
```

The accuracy of the response surfaces can also be illustrated by plotting the predicted results vs. the computed results (Figure 22-2).



Prediction accuracy of Weight
(Iteration 1 - Linear)



Prediction accuracy of Stress
(Iteration 1 - Linear)

Figure 22-2: Prediction accuracy of Weight and Stress (Iteration 1 – Linear)

The R^2 values are large. However the prediction accuracy, especially for weight, seems to be poor, so that a higher order of approximation will be required.

Nevertheless an improved design is predicted with the constraint value (stress) changing from an approximate 4.884 (severely violated) to 1.0 (the constraint is active). Due to inaccuracy, the actual constraint value of the optimum is 0.634. The weight changes from 2.776 to 4.137 (3.557 computed) to accommodate the violated stress:

```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
--------------------------------|-----------|----------|-----------
Area                                    0.2      3.539          4
Base                                    0.1        0.1        1.6
--------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                |       Scaled         |       Unscaled       |
                                |----------------------|----------------------|
RESPONSE                        | Computed   Predicted| Computed   Predicted|
--------------------------------|----------|----------|----------|----------|
Weight                          |     3.557      4.137|     3.557      4.137|
Stress                          |    0.6338          1|    0.6338          1|
--------------------------------|----------|----------|----------|----------|
```

```
               OBJECTIVE:
               ---------
Computed Value  =      3.557
Predicted Value =      4.137


OBJECTIVE FUNCTIONS:
-------------------
OBJECTIVE NAME                   | Computed   Predicted  WT.
---------------------------------|----------|----------|----
Weight                           |     3.557      4.137|   1
---------------------------------|----------|----------|----

CONSTRAINT FUNCTIONS:
--------------------
CONSTRAINT NAME                  | Computed | Predicted|  Lower    |  Upper    |Viol?
---------------------------------|----------|----------|-----------|-----------|-----
Stress                           |    0.6338          1|    -1e+30           1|no
---------------------------------|----------|----------|-----------|-----------|-----


CONSTRAINT VIOLATIONS:
---------------------
                                 |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                  |----------|----------|----------|----------|
                                 |  Lower   |  Upper   |  Lower   |  Upper   |
---------------------------------|----------|----------|----------|----------|
Stress                           |    -          -     |    -          -     |
---------------------------------|----------|----------|----------|----------|


MAXIMUM VIOLATION:
-----------------
                    |         Computed          |        Predicted          |
     Quantity       |---------------------------|---------------------------|
                    |   Constraint      Value   |   Constraint      Value   |
--------------------|---------------|----------|---------------|----------|
Maximum Violation   |Stress                    0|Stress            6.995e-08|
Smallest Margin     |Stress               0.3662|Stress            6.995e-08|
--------------------|---------------|----------|---------------|----------|
```

22.1.3 Updating the approximation to second order

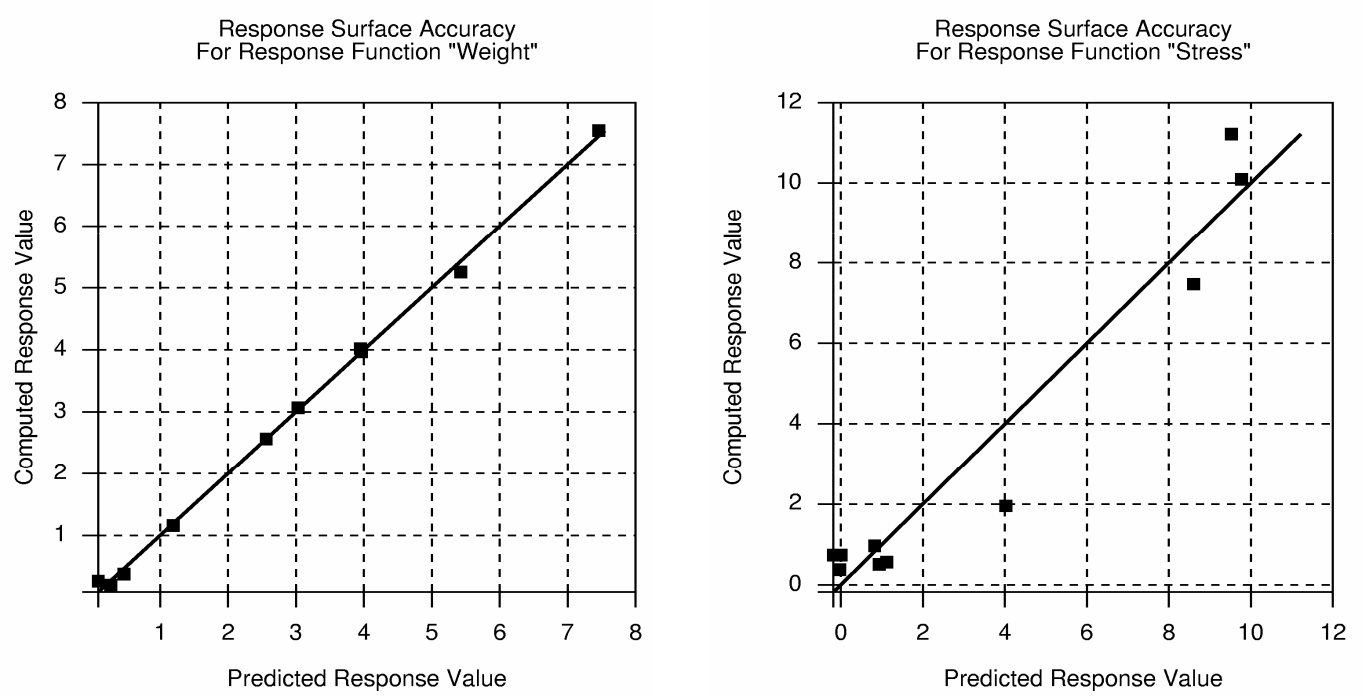
To improve the accuracy, a second run is conducted using a quadratic approximation. The following statements differ from the input file above:

```
"2BAR2: Two-Bar Truss: Updating the approximation to 2nd order"
response 'Weight' quadratic
response 'Stress' quadratic
$
$ EXPERIMENTAL DESIGN
$
 Order quadratic
 Experimental design dopt
 Basis experiment 5toK
 Number experiment 10
```

The approximation results have improved considerably, but the stress approximation is still poor.

```
Approximating Response 'Weight' using 10 points (ITERATION 1)
-------------------------------------------------------------
          Global error parameters of response surface
          -------------------------------------------
Quadratic Function Approximation:
---------------------------------
Mean response value           =     2.8402

RMS error                     =     0.0942 (3.32%)
Maximum Residual              =     0.1755 (6.18%)
Average Error                 =     0.0737 (2.59%)
Square Root PRESS Residual    =     0.2815 (9.91%)
Variance                      =     0.0177
R^2                           =     0.9983
R^2 (adjusted)                =     0.9983
R^2 (prediction)              =     0.9851
Determinant of [X]'[X]        =    14.6629



Approximating Response 'Stress' using 10 points (ITERATION 1)
-------------------------------------------------------------
          Global error parameters of response surface
          -------------------------------------------
Quadratic Function Approximation:
---------------------------------
Mean response value           =     3.4592

RMS error                     =     1.0291 (29.75%)
Maximum Residual              =     2.0762 (60.02%)
Average Error                 =     0.8385 (24.24%)
Square Root PRESS Residual    =     2.4797 (71.68%)
Variance                      =     2.1182
R^2                           =     0.9378
R^2 (adjusted)                =     0.9378
R^2 (prediction)              =     0.6387
Determinant of [X]'[X]        =    14.6629
```

The fit is illustrated below in Figure 22-3:



Prediction accuracy of Weight (Iteration 1 - Quadratic)

Prediction accuracy of Stress (Iteration 1 - Quadratic)

Figure 22-3: Prediction accuracy of Weight and Stress (Iteration 1 – Quadratic)

An improved design is predicted with the constraint value (stress) changing from a computed 0.734 to 1.0 (the approximate constraint becomes active). Due to inaccuracy, the actual constraint value of the optimum is a feasible 0.793. The weight changes from 2.561 to 1.925 (1.907 computed).

```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------
Area                                     0.2     1.766          4
Base                                     0.1    0.4068        1.6
---------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Weight                           |    1.907      1.925|    1.907      1.925|
Stress                           |   0.7927          1|   0.7927          1|
---------------------------------|----------|----------|----------|----------|

               OBJECTIVE:
               ---------
Computed Value  =      1.907
Predicted Value =      1.925
```

```
OBJECTIVE FUNCTIONS:
--------------------
OBJECTIVE NAME                   | Computed   Predicted  WT.
---------------------------------|----------|----------|----
Weight                           |     1.907      1.925|   1
---------------------------------|----------|----------|----

CONSTRAINT FUNCTIONS:
---------------------
CONSTRAINT NAME                  | Computed | Predicted|  Lower   |  Upper   |Viol?
---------------------------------|----------|----------|----------|----------|-----
Stress                           |    0.7927          1|    -1e+30          1|YES
---------------------------------|----------|----------|----------|----------|-----


CONSTRAINT VIOLATIONS:
----------------------
                                 |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                  |----------|----------|----------|----------|
                                 |  Lower   |  Upper   |  Lower   |  Upper   |
---------------------------------|----------|----------|----------|----------|
Stress                           |    -          -     |    -       1.033e-06|
---------------------------------|----------|----------|----------|----------|


MAXIMUM VIOLATION:
------------------
                   |         Computed          |        Predicted          |
     Quantity      |---------------------------|---------------------------|
                   |   Constraint      Value   |   Constraint      Value   |
-------------------|----------------|----------|----------------|----------|
Maximum Violation  |Stress                   0|Stress            1.033e-06|
Smallest Margin    |Stress              0.2073|Stress            1.033e-06|
-------------------|----------------|----------|----------------|----------|
```

22.1.4 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 (1.766; 0.4086) was used as a starting point while the region of interest was cut in half. The order of the approximation is quadratic. The modified statements are:

```
"2BAR3: Two-Bar Truss: Reducing the region of interest"
$ Created on Thu Jul 11 07:46:24 2002
$
$ DESIGN VARIABLES
  Range 'Area' 2
  Range 'Base' 0.8
```

The approximations have been significantly improved:

```
Approximating Response 'Weight' using 10 points (ITERATION 1)
------------------------------------------------------------
          Global error parameters of response surface
          -------------------------------------------
Quadratic Function Approximation:
---------------------------------
```

```
Mean response value           =     2.0282

RMS error                     =     0.0209 (1.03%)
Maximum Residual              =     0.0385 (1.90%)
Average Error                 =     0.0157 (0.77%)
Square Root PRESS Residual    =     0.0697 (3.44%)
Variance                      =     0.0009
R^2                           =     0.9995
R^2 (adjusted)                =     0.9995
R^2 (prediction)              =     0.9944
Determinant of [X]'[X]        =     0.0071

Approximating Response 'Stress' using 10 points (ITERATION 1)
------------------------------------------------------------
          Global error parameters of response surface
          -------------------------------------------
Quadratic Function Approximation:
---------------------------------
Mean response value           =     1.2293

RMS error                     =     0.0966 (7.85%)
Maximum Residual              =     0.1831 (14.89%)
Average Error                 =     0.0826 (6.72%)
Square Root PRESS Residual    =     0.3159 (25.69%)
Variance                      =     0.0186
R^2                           =     0.9830
R^2 (adjusted)                =     0.9830
R^2 (prediction)              =     0.8182
Determinant of [X]'[X]        =     0.0071
```

The results after one iteration are as follows:

```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------
Area                                     0.2      1.444          4
Base                                     0.1     0.5408        1.6
---------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                 |       Scaled        |       Unscaled      |
                                 |---------------------|---------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Weight                           |     1.642      1.627|     1.642      1.627|
Stress                           |    0.9614          1|    0.9614          1|
---------------------------------|----------|----------|----------|----------|

               OBJECTIVE:
               ---------
Computed Value  =      1.642
Predicted Value =      1.627
```

```
OBJECTIVE FUNCTIONS:
--------------------
OBJECTIVE NAME                   | Computed   Predicted  WT.
---------------------------------|----------|----------|----
Weight                           |     1.642      1.627|   1
---------------------------------|----------|----------|----

CONSTRAINT FUNCTIONS:
---------------------
CONSTRAINT NAME                  | Computed | Predicted|  Lower   |  Upper   |Viol?
---------------------------------|----------|----------|----------|----------|-----
Stress                           |    0.9614          1|    -1e+30          1|no
---------------------------------|----------|----------|----------|----------|-----


CONSTRAINT VIOLATIONS:
----------------------
                                 |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                  |----------|----------|----------|----------|
                                 |  Lower   |  Upper   |  Lower   |  Upper   |
---------------------------------|----------|----------|----------|----------|
Stress                           |    -          -     |    -          -     |
---------------------------------|----------|----------|----------|----------|
```

An improved design is predicted with the constraint value (stress) changing from an approximate 0.8033 (0.7928 computed) to 1.0 (the approximate constraint becomes active). Due to inaccuracy, the actual constraint value of the optimum is a feasible 0.961. This value is now much closer to the value of the simulation result. The weight changes from 1.909(1.907 computed) to 1.627 (1.642 computed).

22.1.5 Conducting a trade-off study

The present region of interest (2; 0.8) is chosen in order to conduct a study in which the weight is traded off against the stress constraint. The trade-off is performed by selecting the Trade-off option in the View panel of LS-OPT*ui*.

The upper bound of the stress constraint is varied from 0.2 to 2.0 with 20 increments. Select Constraint as the Trade-off option and enter the bounds and number of increments. Generate the trade-off. This initiates the solution of a series of optimization problems using the response surface generated in Section 22.1.4, with the constraint in each (constant coefficient of the constraint response surface polynomial) being varied between the limits selected. The resulting curve is also referred to as a Pareto optimality curve. When plotting, select the ‘Constraint’ Stress, and not the ‘Response’ Stress, as the latter represents only the left-hand side of the constraint equation (17.2).

The resulting trade-off diagram (Figure 22-4) shows the compromise in weight when the stress constraint is tightened.

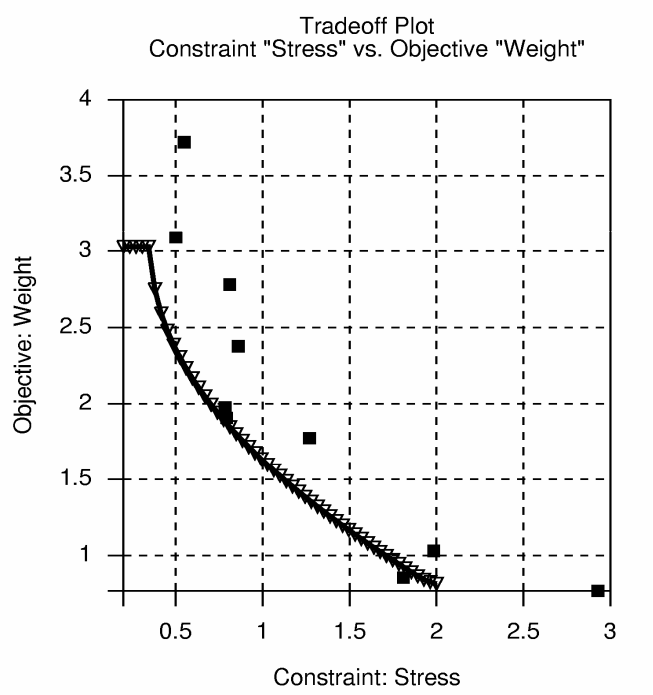


Figure 22-4: Trade-off of stress and weight

22.1.6 Automating the design process

This section illustrates the automation of the design process for both a linear and a quadratic response surface approximation order. 10 iterations are performed for the linear approximation, with only 5 iterations performed for the more expensive quadratic approximation.

The modified statements in the input file are as follows:

```
Variable 'Area' 2
  Range 'Area' 4
 Variable 'Base' 0.8
  Range 'Base' 1.6
$
$ EXPERIMENTAL DESIGN
$
Order linear
Number experiment 5
$
$ JOB INFO
$
iterate 10
```

for the linear approximation, and

```
$
$ EXPERIMENTAL DESIGN
$
 Order quadratic
```

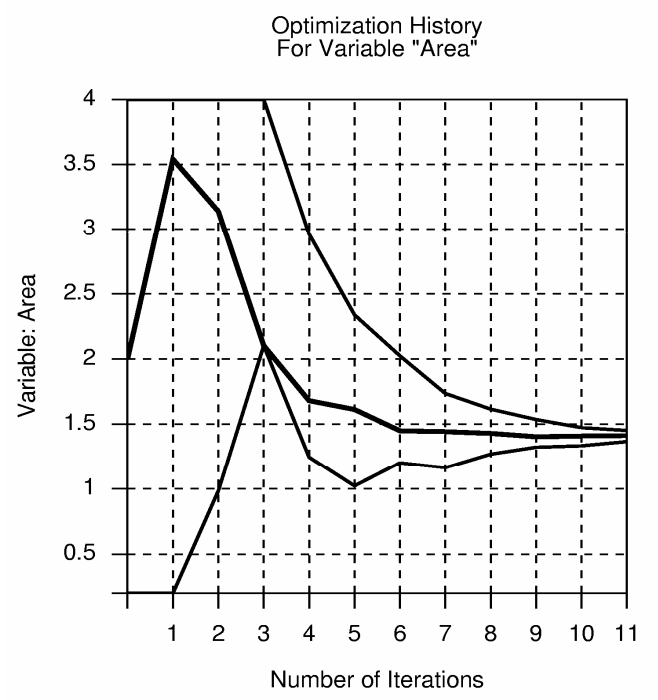
```
 Number experiment 10
$
$ JOB INFO
$
iterate 5
```

The final results of the two types of approximations are as follows:

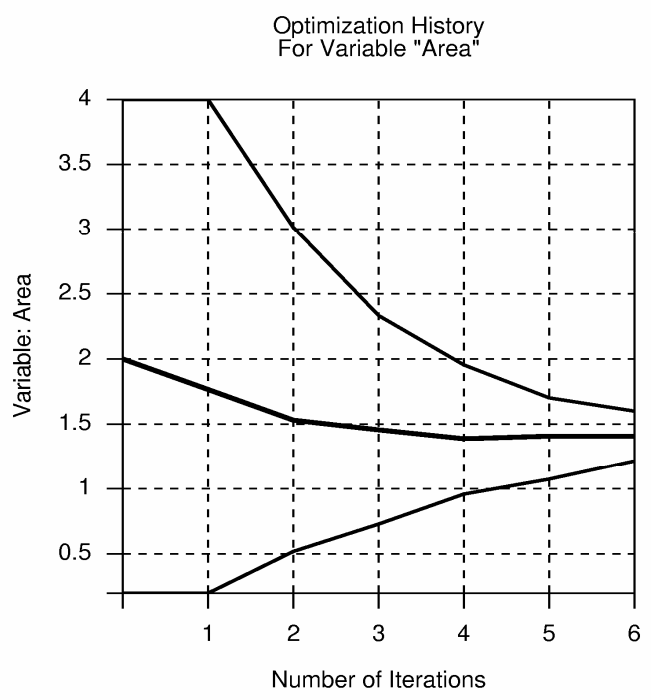
Table 22.1-1: Summary of final results (2-bar truss)

	Linear	Quadratic
Number of iterations	10	5
Number of simulations	51	51
Area	1.414	1.408
Base	0.3737	0.3845
Weight	1.51	1.509
Stress	0.9993	1.000

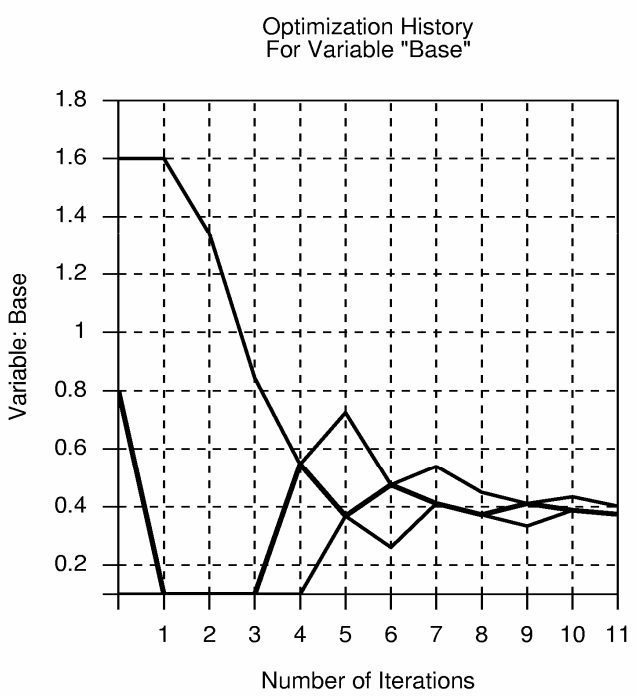
The optimization histories have been plotted to illustrate convergence in Figure 22-5.



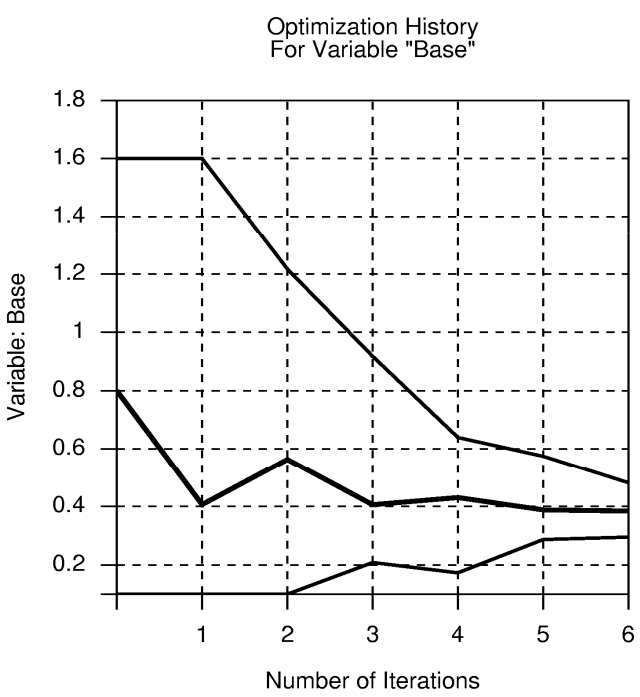
a) Optimization history of Area (Linear)



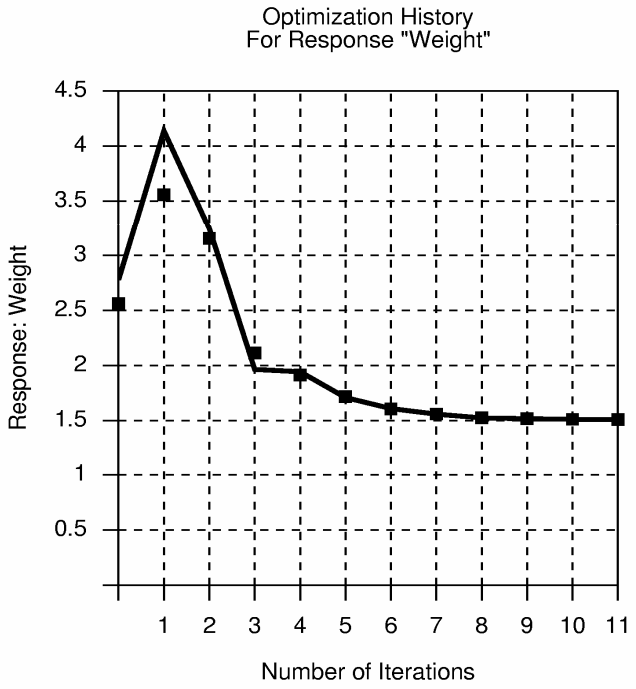
b) Optimization history of Area (Quadratic)



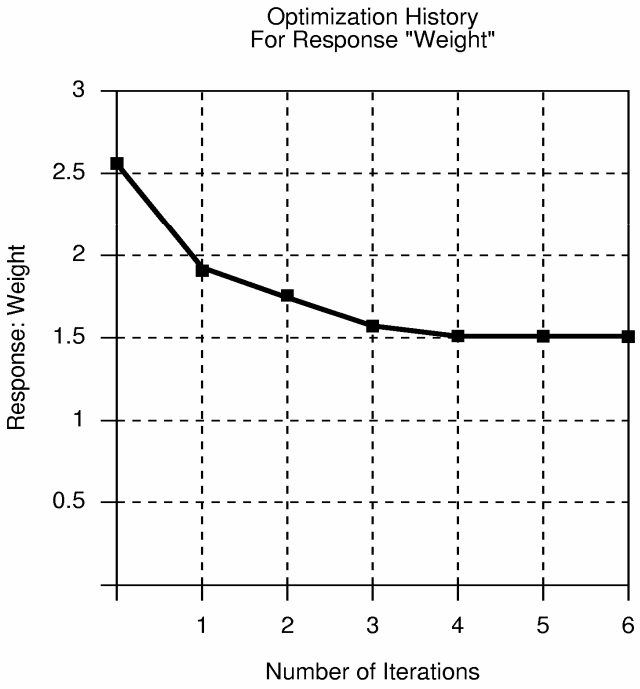
c) Optimization history of Base (Linear)



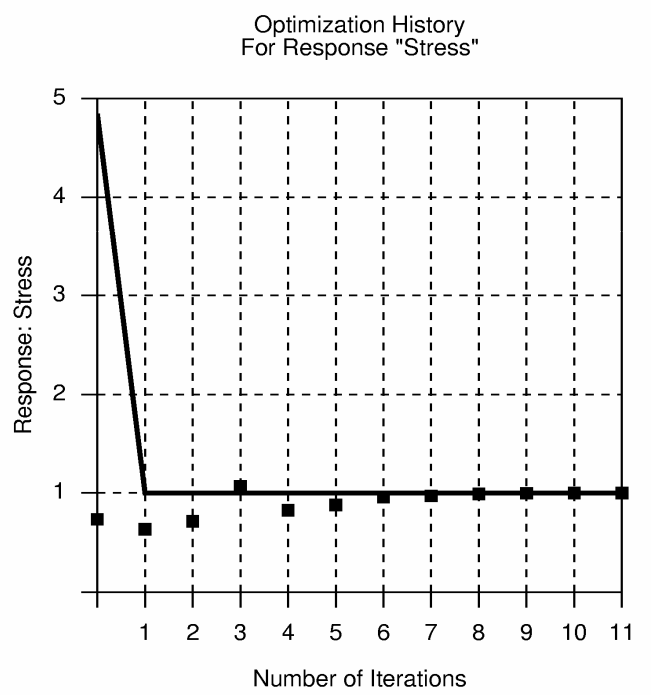
d) Optimization history of Base (Quadratic)



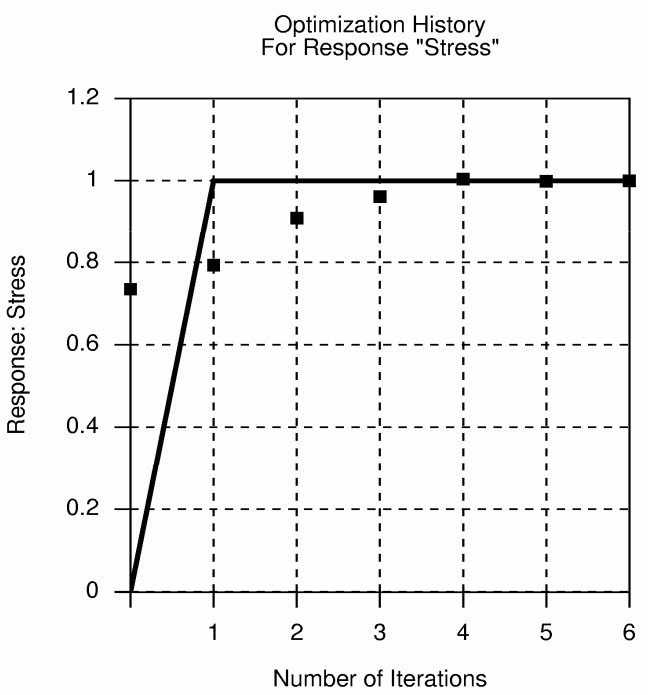
e) Optimization history of Weight (Linear)



f) Optimization history of Weight (Quadratic)



g) Optimization history of Stress (Linear)



h) Optimization history of Stress (Quadratic)

Figure 22-5: Optimization history of design variables and responses (Linear and Quadratic)

Remarks:

1. Note that the more accurate but more expensive quadratic approximation converges in about 3 design iterations (30 simulations), while it takes about 7 iterations (35 simulations) for the objective of the linear case to converge.

2. In general, the lower the order of the approximation, the more iterations are required to refine the optimum.

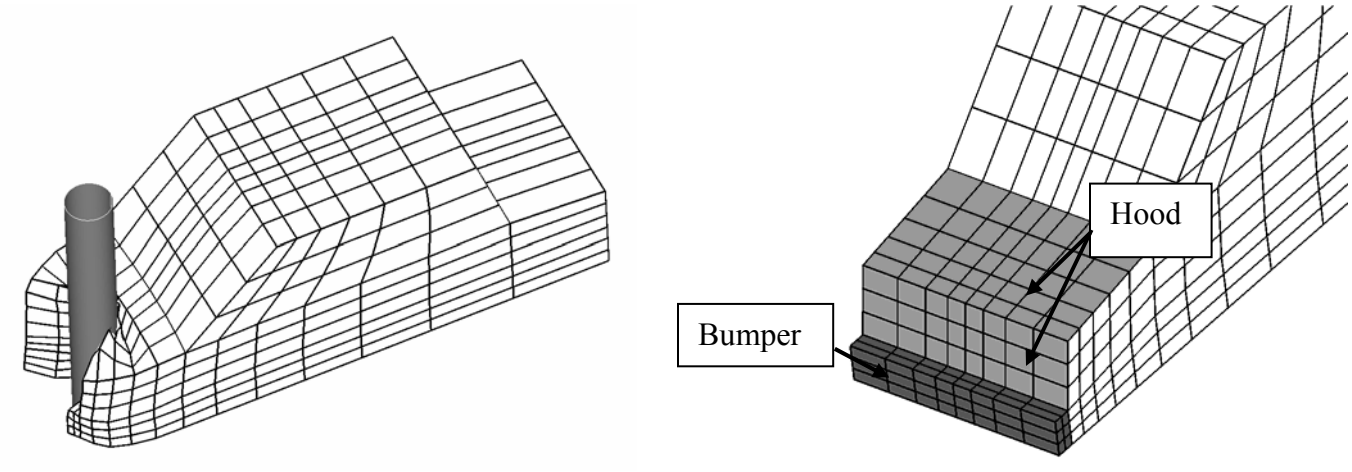
22.2 Small car crash (2 variables)

This example has the following features:

- An LS-DYNA explicit crash simulation is performed.
- Extraction is performed using standard LS-DYNA interfaces.
- First- and second-order response surface approximations are compared.
- The design optimization process is automated.
- A trade-off study is performed using both a quadratic and neural network approximation.
- A limited reliability-based design optimization study is performed.

22.2.1 Introduction

This example considers the crashworthiness of a simplified small car model. A simplified vehicle moving at a constant velocity of 15.64m.s^{-1} (35mph) impacts a rigid pole. See Figure 22-6. 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 a simulation duration of 50ms.



a) deformed (50ms) b) undeformed

Figure 22-6: Small car impacting a pole

22.2.2 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 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 (`t_hood`) and the shell thickness of the bumper (`t_bumper`) (see Figure 22-6).

22.2.3 Design formulation

The design formulation is as follows:

Minimize

$$\text{HIC (15ms)} \tag{22.2-1}$$

subject to

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

The intrusion is measured as the difference between the displacement of nodes **167** and **432**.

Remark:

- The mass is computed but not constrained. This is useful for monitoring the mass changes.

22.2.4 Modeling

The simulation is performed using LS-DYNA. An extract from the parameterized input deck is shown below. Note how the design variables are labeled for substitution through the characters << >>. The cylinder for impact is modeled as a rigid wall.

```
$
$ DEFINITION OF MATERIAL     1
$
*MAT_PLASTIC_KINEMATIC
1,1.000E-07,2.000E+05,0.300,400.,0.,0.
0.,0.,0.
*HOURGLASS
1,0,0.,0,0.,0.
*SECTION_SHELL
1,2,0.,0.,0.,0.,0
2.00,2.00,2.00,2.00,0.
*PART
material type # 3 (Kinematic/Isotropic Elastic-Plastic)
1,1,1,0,1,0
$
$ DEFINITION OF MATERIAL     2
$
*MAT_PLASTIC_KINEMATIC
2,7.800E-08,2.000E+05,0.300,400.,0.,0.
```

```
0.,0.,0.
*HOURGLASS
2,0,0.,0,0.,0.
*SECTION_SHELL
2,2,0.,0.,0.,0.,0
<<t_bumper>>,<<t_bumper>>,<<t_bumper>>,<<t_bumper>>,0.
*PART
material type # 3 (Kinematic/Isotropic Elastic-Plastic)
2,2,2,0,2,0
$
$ DEFINITION OF MATERIAL     3
$
*MAT_PLASTIC_KINEMATIC
3,7.800E-08,2.000E+05,0.300,400.,0.,0.
0.,0.,0.
*HOURGLASS
3,0,0.,0,0.,0.
*SECTION_SHELL
3,2,0.,0.,0.,0.,0
<<t_hood>>,<<t_hood>>,<<t_hood>>,<<t_hood>>,0.
*PART
material type # 3 (Kinematic/Isotropic Elastic-Plastic)
3,3,3,0,3,0
$
$ DEFINITION OF MATERIAL     4
$
*MAT_PLASTIC_KINEMATIC
4,7.800E-08,2.000E+05,0.300,400.,0.,0.
0.,0.,0.
*HOURGLASS
4,0,0.,0,0.,0.
*SECTION_SHELL
4,2,0.,0.,0.,0.,0
<<t_hood>>,<<t_hood>>,<<t_hood>>,<<t_hood>>,0.
*PART
material type # 3 (Kinematic/Isotropic Elastic-Plastic)
4,4,4,0,4,0
$
$ DEFINITION OF MATERIAL     5
$
*MAT_PLASTIC_KINEMATIC
5,7.800E-08,2.000E+05,0.300,400.,0.,0.
0.,0.,0.
*HOURGLASS
5,0,0.,0,0.,0.
*SECTION_SHELL
5,2,0.,0.,0.,0.,0
<<t_hood>>,<<t_hood>>,<<t_hood>>,<<t_hood>>,0.
*PART
material type # 3 (Kinematic/Isotropic Elastic-Plastic)
5,5,5,0,5,0
$
```

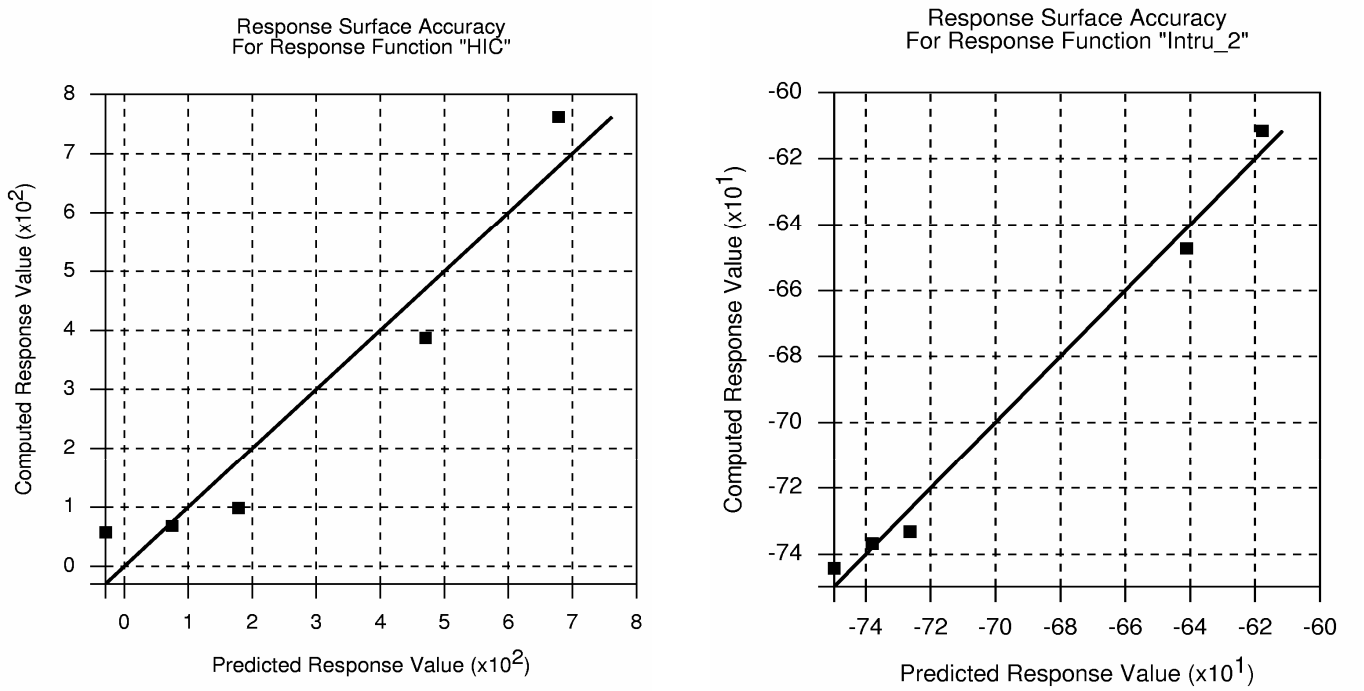
22.2.5 First linear iteration

A design space of [1; 5] is used for both design variables with no range specified. This means that the range defaults to the whole design space. The LS-OPT input file is as follows:

```
"Small Car Problem: EX4a"
$ Created on Mon Aug 26 19:11:06 2002
solvers 1
responses 5
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 2
 Variable 't_hood' 1
  Lower bound variable 't_hood' 1
  Upper bound variable 't_hood' 5
 Variable 't_bumper' 3
  Lower bound variable 't_bumper' 1
  Upper bound variable 't_bumper' 5
$
$ DEFINITION OF SOLVER "1"
$
 solver dyna '1'
  solver command "lsdyna"
  solver input file "car5.k"
  solver append file "rigid2"
  solver order linear
  solver experiment design dopt
  solver number experiments 5
  solver basis experiment 3toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "1"
$
 response 'Acc_max' 1 0 "DynaASCII Nodout X_ACC   432 Max SAE 60"
 response 'Acc_max' linear
 response 'Mass' 1 0 "DynaMass 2 3 4 5 MASS"
 response 'Mass' linear
 response 'Intru_2' 1 0 "DynaASCII Nodout X_DISP 432 Timestep"
 response 'Intru_2' linear
 response 'Intru_1' 1 0 "DynaASCII Nodout X_DISP 167 Timestep"
 response 'Intru_1' linear
 response 'HIC' 1 0 "DynaASCII Nodout HIC15 9810. 1 432"
 response 'HIC' linear
$
$ NO HISTORIES DEFINED FOR SOLVER "1"
$
$
$ HISTORIES AND RESPONSES DEFINED BY EXPRESSIONS
$
 composites 1
 composite 'Intrusion' type weighted
```

```
  composite 'Intrusion' response 'Intru_2' -1 scale 1
  composite 'Intrusion' response 'Intru_1' 1 scale 1
$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'HIC' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 1
 constraint 'Intrusion'
  upper bound constraint 'Intrusion' 550
$
$ JOB INFO
$
 iterate param design 0.01
 iterate param objective 0.01
 iterate 1
STOP
```

The computed vs. predicted HIC and `Intru_2` responses are given in Figure 22-7. The corresponding R^2 value for HIC is 0.9248, while the RMS error is 27.19%. For Intru_2, the R^2 value is 0.9896, while the RMS error is 0.80%.



a) HIC response b) `Intru_2` response

Figure 22-7: Computed vs. predicted responses – Linear approximation

The summary data for the first iteration is:

Baseline:

```
---------------------------------------
ITERATION NUMBER (Baseline)
---------------------------------------


DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------
t_hood                                     1          1          5
t_bumper                                   1          3          5
---------------------------------|-----------|----------|-----------


RESPONSE FUNCTIONS:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Acc_max                          | 8.345e+04  1.162e+05| 8.345e+04  1.162e+05|
Mass                             |    0.4103     0.4103|    0.4103     0.4103|
Intru_2                          |    -736.7       -738|    -736.7       -738|
Intru_1                          |      -161     -160.7|      -161     -160.7|
HIC                              |     68.26      74.68|     68.26      74.68|
---------------------------------|----------|----------|----------|----------|
```

and 1st optimum:

```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------
t_hood                                     1      1.549          5
t_bumper                                   1          5          5
---------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Acc_max                          | 1.248e+05  1.781e+05| 1.248e+05  1.781e+05|
Mass                             |    0.6571      0.657|    0.6571      0.657|
Intru_2                          |    -713.7     -711.4|    -713.7     -711.4|
Intru_1                          |    -164.6     -161.4|    -164.6     -161.4|
HIC                              |     126.7      39.47|     126.7      39.47|
---------------------------------|----------|----------|----------|----------|
```

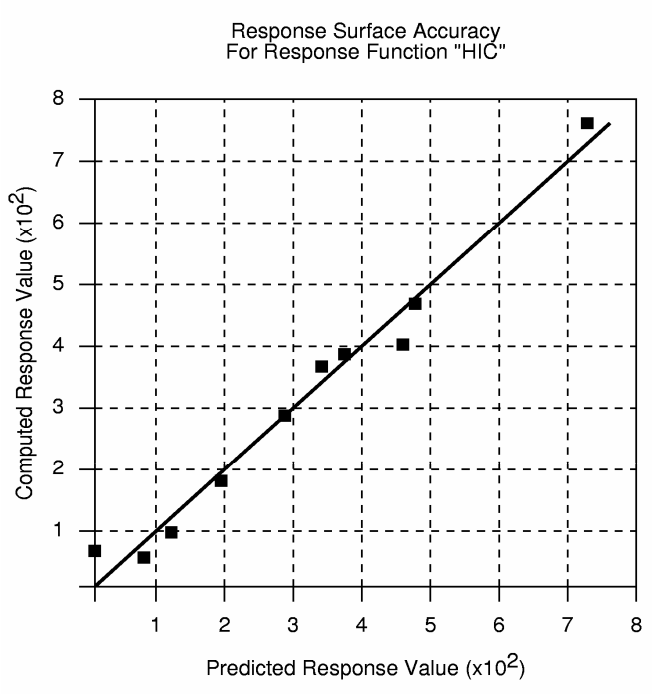
22.2.6 First quadratic iteration

The LS-OPT input file is modified as follows (the response approximations are all `quadratic` (not shown)):

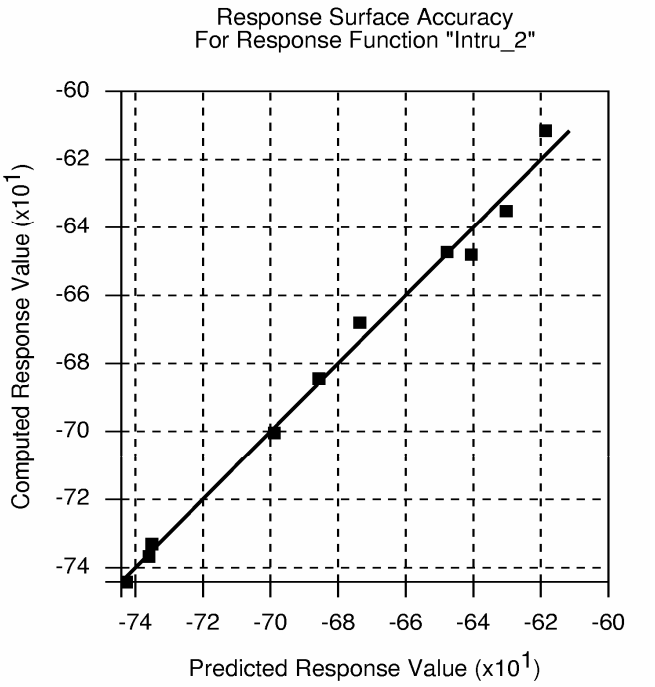
```
Order quadratic
 Experimental design dopt
 Basis experiment 5toK
 Number experiment 10
```

For very expensive simulations, if previous extracted simulation is available, as, e.g., from the previous linear iteration in Section 22.2.5, then these points can be used to reduce the computational cost of this quadratic approximation. To do this, the previous `AnalysisResults.1` file is copied to the current work directory and renamed `AnalysisResults.PRE.1`.

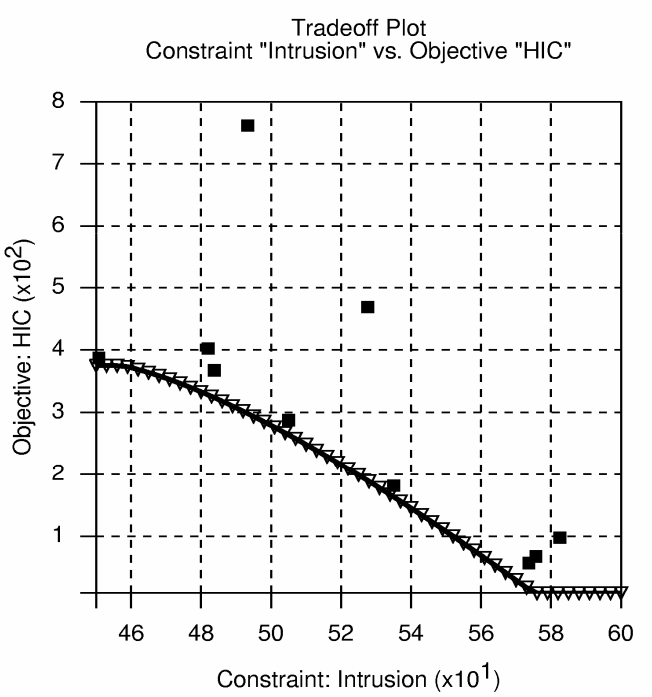
As is shown in the results below, the computed vs. predicted HIC and `Intru_2` responses are is now improved from the linear approximation. The accuracy of the HIC and `Intru_2` responses are given in Figure 22-8. The corresponding R^2 value for HIC is 0.9767, while the RMS error is 10.28%. For `Intru_2`, the R^2 value is 0.9913, while the RMS error is 0.61%. When conducting trade-off studies, a higher-order approximation like the current one will be preferable. See trade-off of HIC versus intrusion in a range 450mm to 600mm, in Figure 22-8c).



a) HIC response



b) `Intru_2` response



c) Trade-off of HIC versus Intrusion

Figure 22-8: Computed vs. predicted responses and trade-off – Quadratic approximation

The summary data for the first iteration is:

Baseline:

```
---------------------------------------
ITERATION NUMBER (Baseline)
---------------------------------------

DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------
t_hood                                     1          1           5
t_bumper                                   1          3           5
---------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                 |       Scaled        |       Unscaled      |
                                 |---------------------|---------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Acc_max                          | 8.345e+04  1.385e+05| 8.345e+04  1.385e+05|
Mass                             |    0.4103     0.4103|    0.4103     0.4103|
Intru_2                          |    -736.7       -736|    -736.7       -736|
Intru_1                          |      -161     -160.3|      -161     -160.3|
HIC                              |     68.26      10.72|     68.26      10.72|
---------------------------------|----------|----------|----------|----------|
```

and 1st optimum:

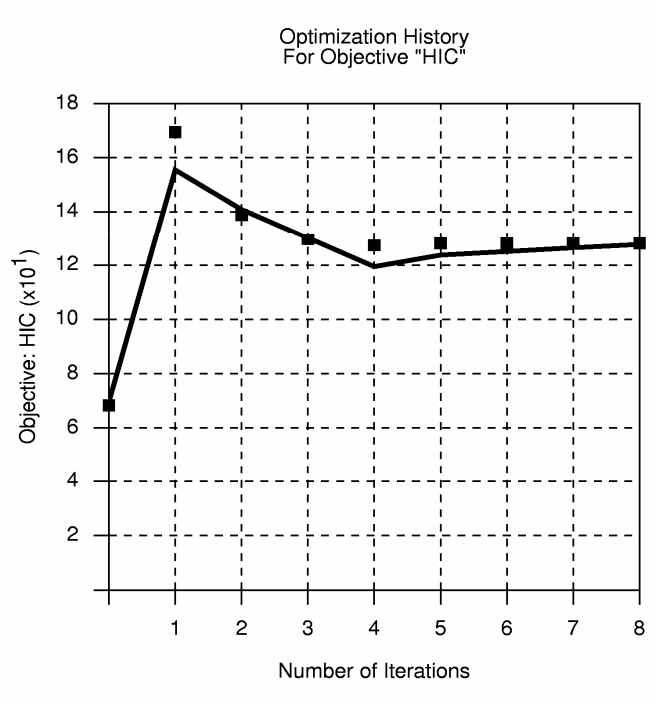
```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------
t_hood                                     1      1.653          5
t_bumper                                   1      3.704          5
---------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Acc_max                          | 1.576e+05  1.985e+05| 1.576e+05  1.985e+05|
Mass                             |    0.6017     0.6018|    0.6017     0.6018|
Intru_2                          |    -712.7     -711.9|    -712.7     -711.9|
Intru_1                          |    -163.3     -161.9|    -163.3     -161.9|
HIC                              |     171.4      108.2|     171.4      108.2|
---------------------------------|----------|----------|----------|----------|
```

22.2.7 Automated run

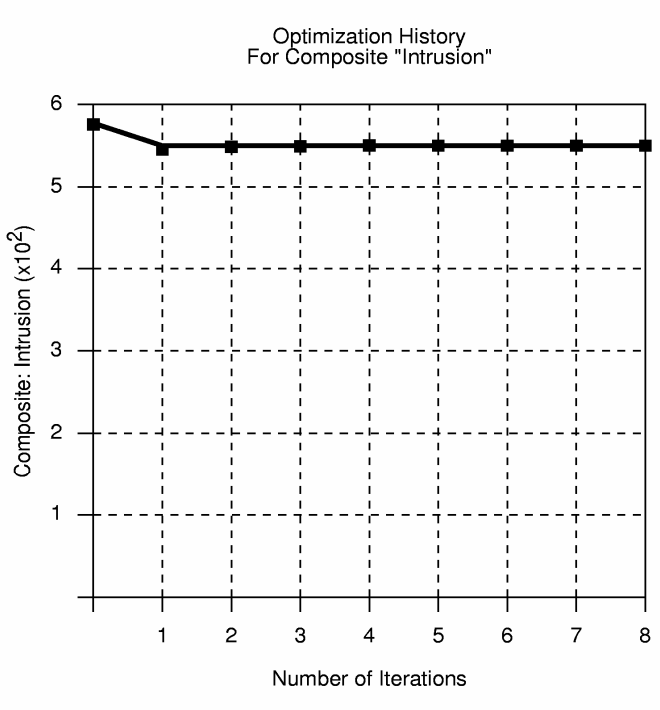
An automated optimization is performed with a linear approximation. The LS-OPT input file is modified as follows:

```
Order linear
 Experimental design dopt
 Basis experiment 3toK
 Number experiment 5

iterate 8
```

It can be seen in Figure 22-9 that the objective function (HIC) and intrusion constraint are approximately optimized at the 5th 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 22-10). For `t_bumper`, panning occurs as well due to the fact that the linear approximation predicts a variable on the edge of the subregion.

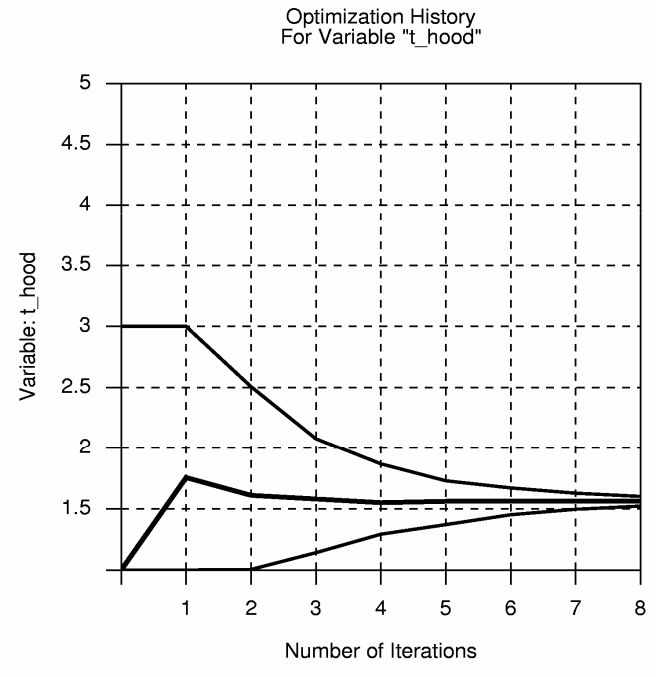


a) Optimization history of HIC

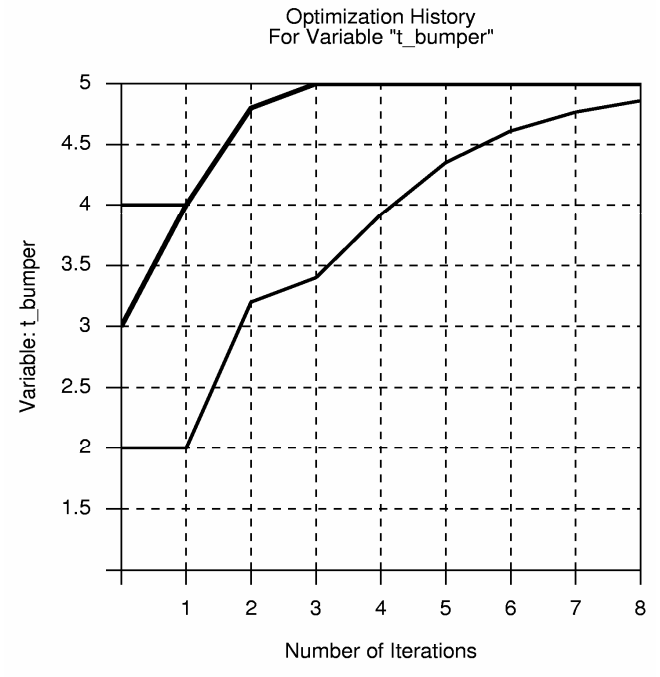


b) Optimization history of Intrusion

Figure 22-9: Optimization history of HIC and Intrusion



a) Optimization history of `t_hood`



b) Optimization history of `t_bumper`

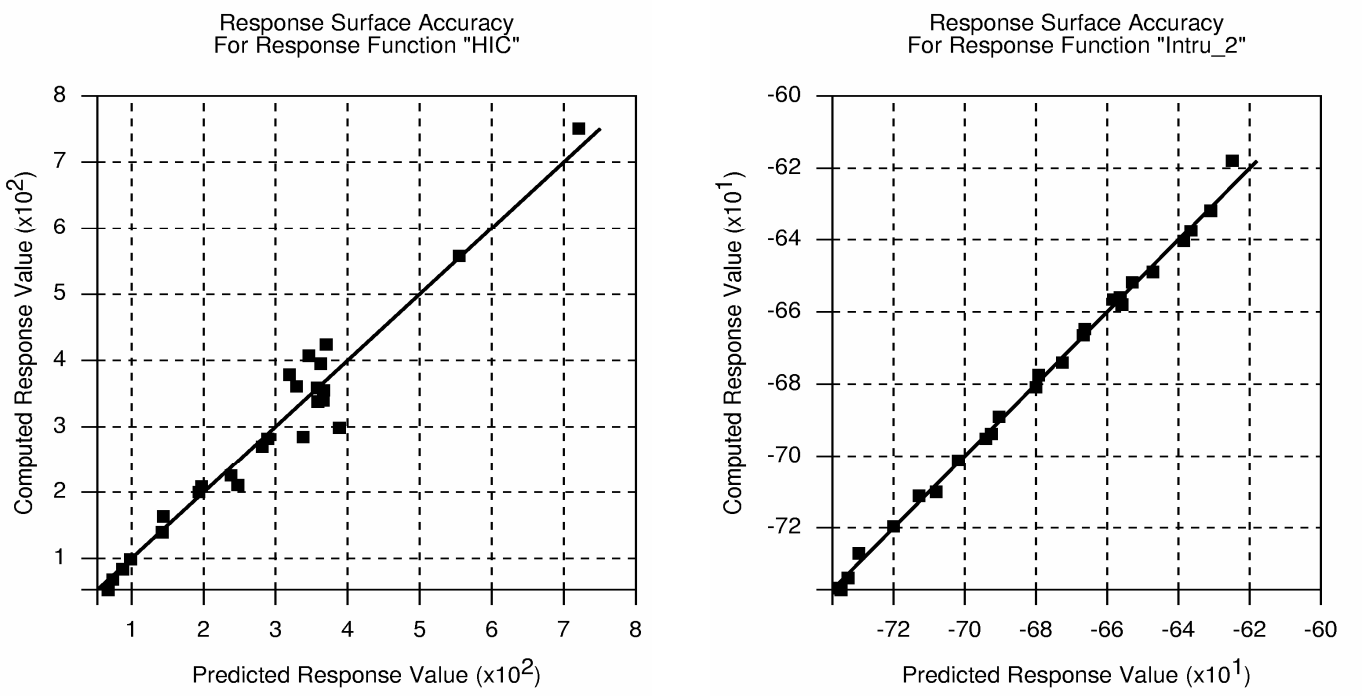
Figure 22-10: Optimization history of design variables

22.2.8 Trade-off using neural network approximation

In order to build a more accurate response surface for trade-off studies, the Neural Net method is chosen under the ExpDesign panel. This results in a feedforward (FF) neural network (Section 3.1) being solved for the points selected. The recommended point selection scheme (Space Filling) is used. One iteration is performed to analyze only one experimental design with 25 points. The modifications to the command input file are as follows:

```
$
$ DEFINITION OF SOLVER "1"
$
 solver dyna '1'
  solver command "lsdyna"
  solver input file "car5.k"
  solver append file "rigid2"
  solver order FF
  solver update doe
  solver experiment design space_filling
  solver number experiments 25
iterate 1
```

The response surface accuracy is illustrated in Figure 22-11 for the HIC and `Intru_2` responses. The HIC has more scatter than `Intru_2` for the 25 design points used.



a) HIC response b) `Intru_2` constraint

Figure 22-11: Response surface accuracy using neural network approximation

A trade-off study considers a variation in the Intrusion constraint (originally fixed at 550mm) between 450 and 600mm, the same as in Figure 22-8c). The experimental design used for the responses in Figure 22-11

is shown in Figure 22-12. The effect of the Space-Filling algorithm in maximizing the minimum distance between the experimental design points can clearly be seen from the evenly distributed design. The resulting Pareto optimality curves for HIC and the two design variables (`t_hood` and `t_bumper`) can be seen in Figure 22-13. It can be seen that a tightening of the Intrusion constraint increases the HIC value through an increase of the hood thickness in the optimal design.

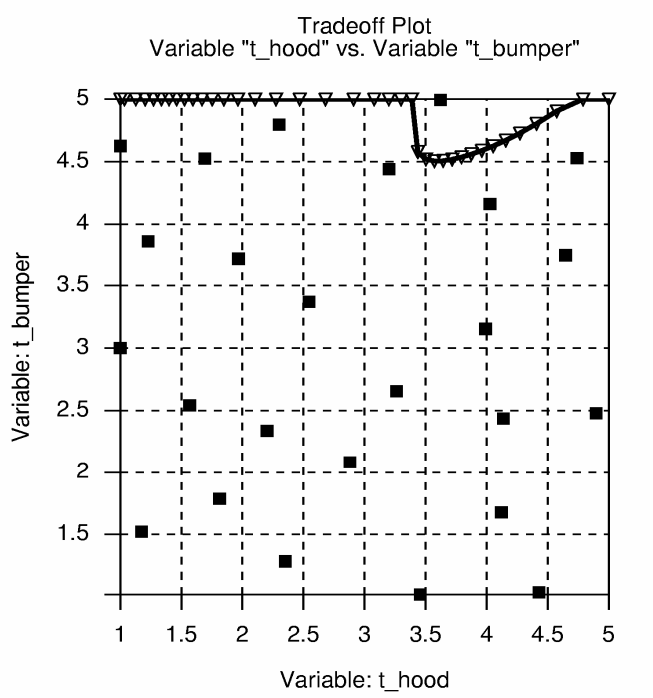
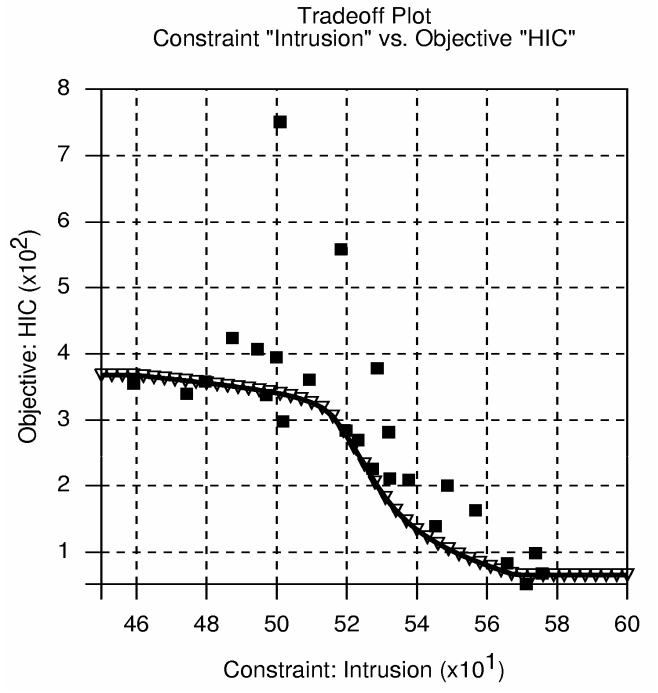
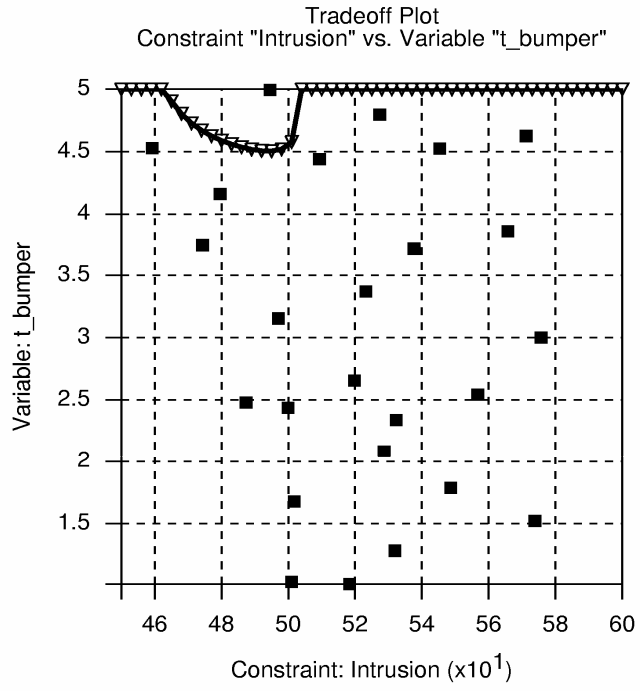
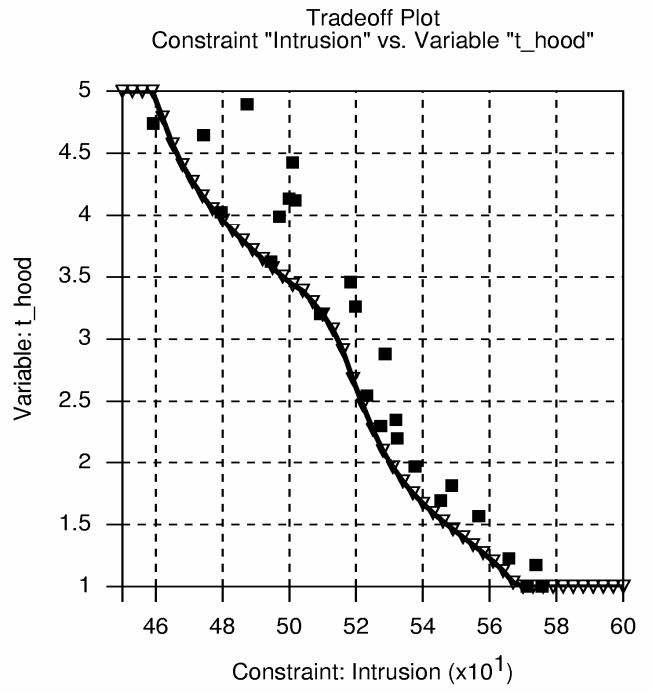


Figure 22-12: Experimental design points used for trade-off





a) Objective (HIC) versus Intrusion constraint b) `t_bumper` versus Intrusion constraint



c) `t_hood` versus Intrusion constraint

Figure 22-13: Trade-off results – Small car (2 variables)

22.2.9 Mixed-discrete optimization

Mixed discrete optimization is achieved simply by setting the `t_hood` variable to be discrete with possible values of 1.0, 2.0, 3.0, 4.0, and 5.0. The input file commands describing the variables are:

```
$
$ DESIGN VARIABLES
$
variables 2
 Variable 't_bumpr' 1
  Lower bound variable 't_bumpr' 1
  Upper bound variable 't_bumpr' 5
  Range 't_bumpr' 4
 Variable 't_hood' 1
  Variable 't_hood' discrete {1 2 3 4 5 }
$
```

The results design variables histories are shown in `Figure 22-14`.

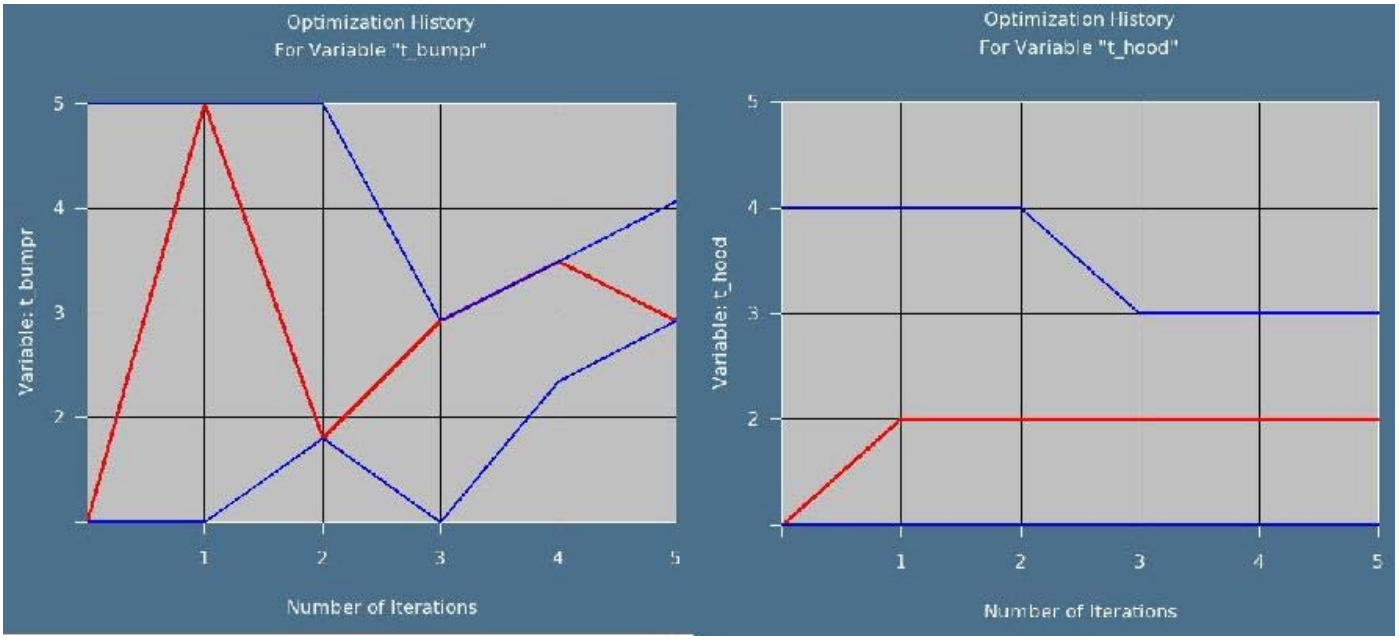


Figure 22-14 Mixed-discrete variable histories.

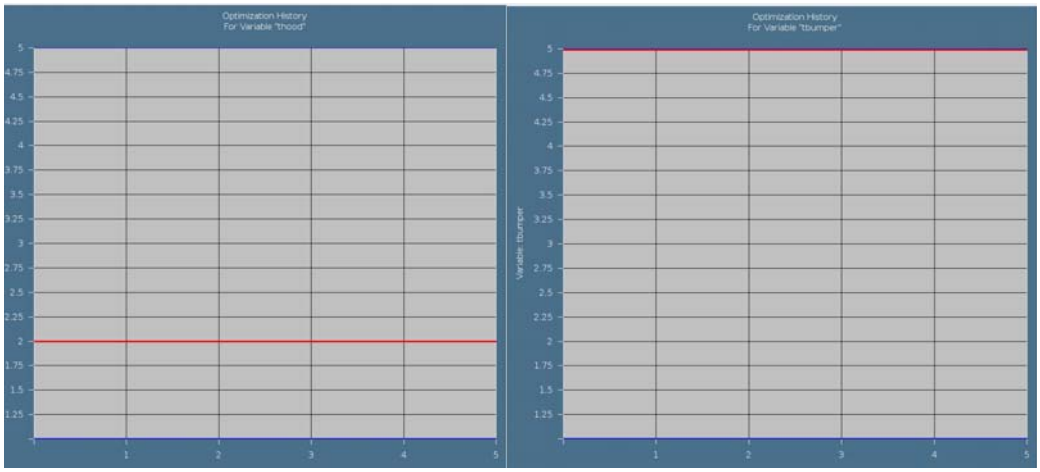
22.2.10 Optimization using Direct GA simulation

The same problem is solved using a direct GA simulation. For illustration, the population size is taken as 6 (*Popsize*) and number of generations is limited to 5 (*Generation*). The continuous variable 'bumper thickness' is treated as binary variable (*Encoding variable*), where 20 bits are used to discretize the variable (*Number of bits*). The Stochastic Universal Sampling method is used as selection operator (*Selection*). Elitism is switched on (*Elitism*) and two elite members (*NumElites*) are used in each generation. Since, both real and binary encoding is used for different variables, the operators have to be specified in both genotype spaces. For real crossover, SBX operator is used (*Real Crossover Type*) with a distribution index of 5 (*Real Crossover Distribution Index*) and crossover probability of 0.99 (*Real Crossover Probability*). Uniform crossover operator is used for binary variables (*Binary Crossover Type*), with a crossover probability of 1.0 (*Binary Crossover Probability*). While the real mutation probability (*Real Mutation Probability*) is 1.0, binary mutation probability (*Binary Mutation Probability*) is 0.05.

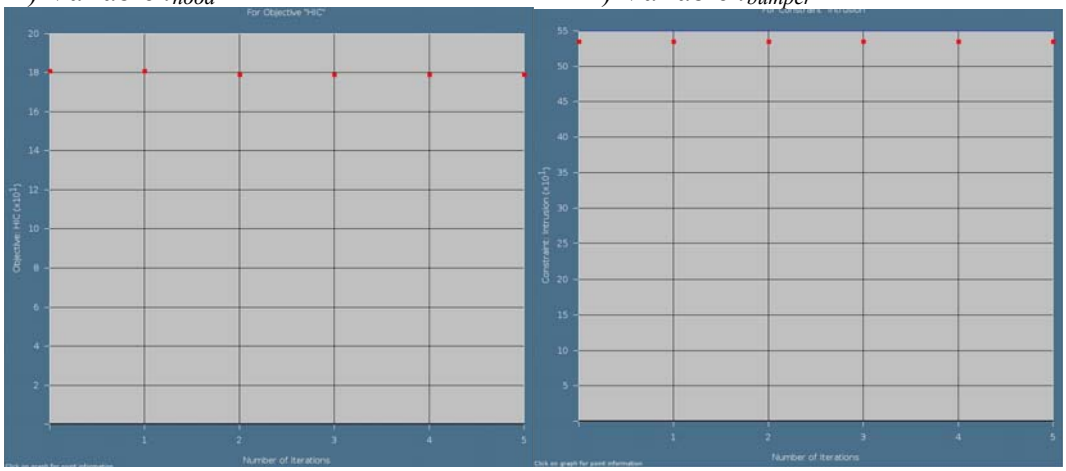
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method GA

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Genetic Algorithm Parameters
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 GA Parameter Popsize 6
 GA Parameter Generation 5
 Encoding Variable 't_bumper' 2
 Number of Bits variable 't_bumper' 20
 GA Parameter Selection 3
 GA Parameter Elitism 1
 GA Parameter NumElites 2
```

```
GA Parameter Real Crossover Type 1
GA Parameter Real Crossover Probability 0.99
GA Parameter Real Crossover Distribution Index 5.0
GA Parameter Binary Crossover Type 2
GA Parameter Binary Crossover Probability 1.0
GA Parameter Real Mutation Probability 1.0
GA Parameter Real Mut Dist Index 5.0
GA Parameter Binary Mutation Probability 0.05
GA Parameter Restart Status 0
GA Parameter Seed 854526

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

The outcome of the optimization is shown in Figure 22-15. In the chosen example, there is small variation in the optimized results with generation. The discrete variable was fixed at 2 units and the variations in the bumper thickness were very small. Consequently, the reduction in HIC and intrusion values are not visible in the optimization history, though there were small improvements. Note that the optimization history treats 'generation' as 'iteration' to display results.



A) Variable t_{hood} B) Variable t_{bumper}



C) Objective HIC D) Constraint Intrusion

Figure 22-15 Optimization history of mixed-discrete variable optimization using direct GA simulation.

22.2.11 RBDO (Reliability-based design optimization) using FOSM (First Order Second Moment Method)*

The First Order Second Moment reliability-based design optimization in LS-OPT is illustrated in this example. The optimization problem is modified as follows:

Minimize

$$\text{HIC} \tag{22.2-2}$$

subject to $\text{Probability}[\text{Intrusion} > 550\text{mm}] < 10^{-6}$

The formulation in Eq. HIC (22.2-2 implies that the car is made safer by 6 standard deviations of the intrusion.

The following commands must be added to the LS-OPT input file used for the automated run (Section 22.2.7):

```
$
$ Define distributions
$
Distributions 2
 distribution 'hood_dist' UNIFORM -0.05 0.05
 distribution 'bumper_dist' UNIFORM -0.05 0.05
$
$ Assign distributions to variables
$
variable 't_hood' distribution 'hood_dist'
variable 't_bumper' distribution 'bumper_dist'
$
$ Assign probabilistic bounds to constraints
$
 probability upper bound constraint 'Intrusion' 1e-6
```

The results are: x = <1.78, 3.44>, a HIC value of 182, and an intrusion of 545 with a standard deviation of 1.06.

22.3 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].

22.3.1 Problem statement

The problem consists of a tube impacting a rigid wall as shown in Figure 22-16. 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 22-17 for a typical design.

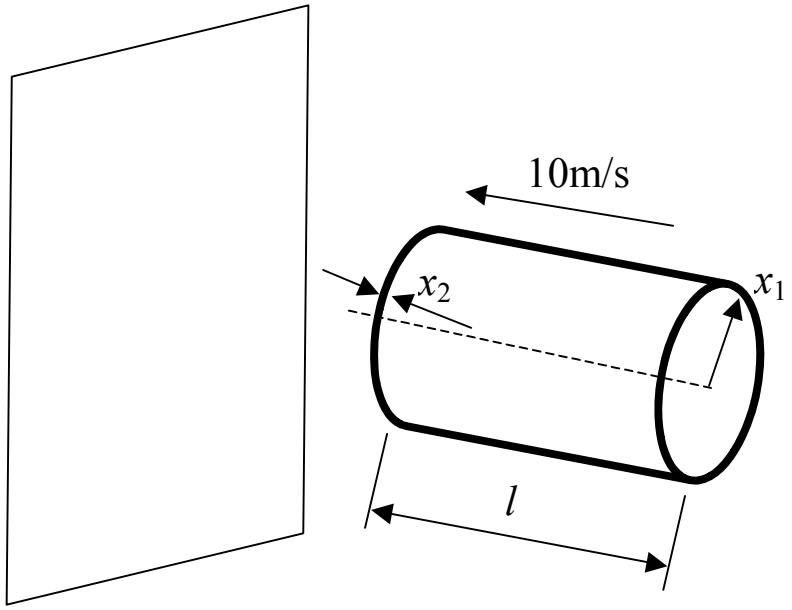


Figure 22-16: Impacting cylinder

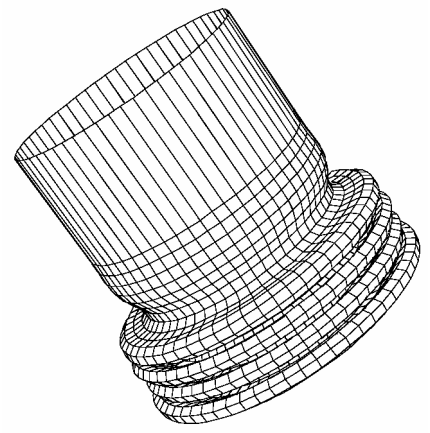


Figure 22-17: Deformed finite element model (time = 20ms)

The optimization problem is stated as:

$$\text{Maximize } E_{\text{internal}}(x_1, x_2)\big|_{t=0.02}$$

subject to

$$F_{normal}^{wall}(x_1, x_2)\big|_{average} \le 70\,000$$

$$l(x) = \frac{0.52}{2\pi\rho x_1 x_2}$$

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 $F_{normal}^{wall}(x)\big|_{average}$ and $l(\mathbf{x})$ 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 22-17. 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 lsdsi 13 slvmat 1;scoef .4 dcoef .4 sfsps 1.5 ; ; ;
c ************** part 1 mat 1 ************* shell
cylinder
-1; 1 60; 1 50 51;
%r
0 360
0 %l [%l2+%l]
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
```

22.3.2 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 22-18).

```
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:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | 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|
---------------------------------|----------|----------|----------|----------|
```

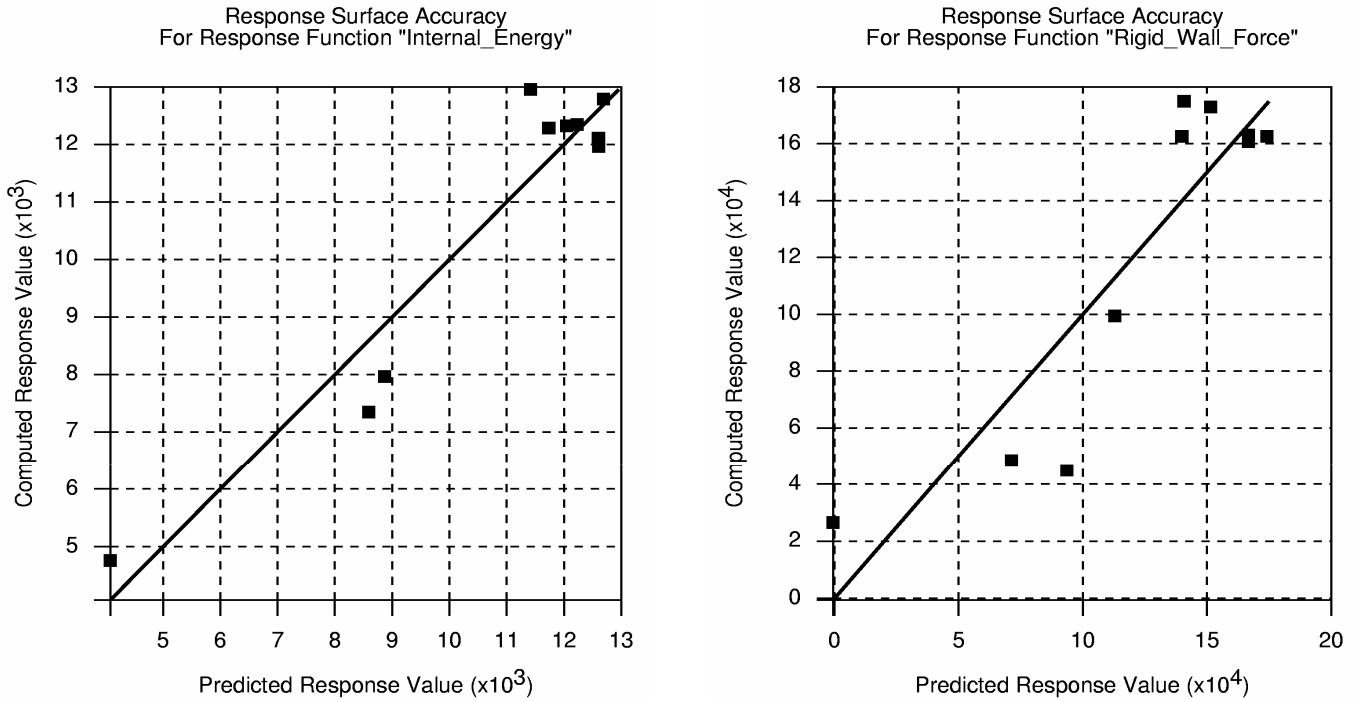


Figure 22-18: Prediction accuracy of Internal Energy and Rigid Wall Force (One Quadratic iteration)

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:
-------------------
                                 |       Scaled        |       Unscaled      |
                                 |---------------------|---------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Internal_Energy                  |      7914       8778|      7914       8778|
Rigid_Wall_Force                 | 4.789e+04      7e+04| 4.789e+04      7e+04|
---------------------------------|----------|----------|----------|----------|
```

22.3.3 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 22-19.

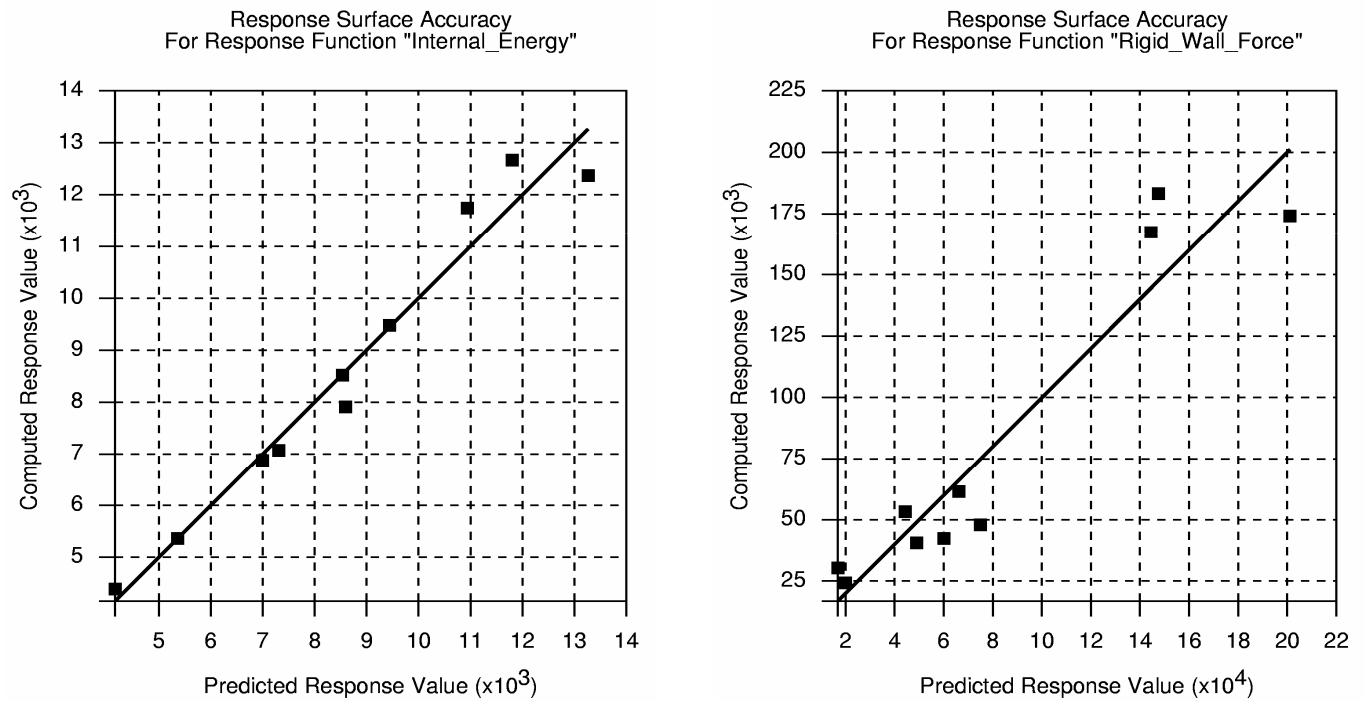


Figure 22-19: Prediction accuracy of Internal Energy and Rigid Wall Force (One Quadratic iteration)

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:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Internal_Energy                  |      9777       9575|      9777       9575|
Rigid_Wall_Force                 | 6.417e+04      7e+04| 6.417e+04      7e+04|
---------------------------------|----------|----------|----------|----------|
```

22.3.4 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:
-------------------
                                |       Scaled         |       Unscaled       |
                                |----------------------|----------------------|
RESPONSE                        | 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 22.3-1: 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.

22.3.5 Response filtering: using the peak force as a constraint

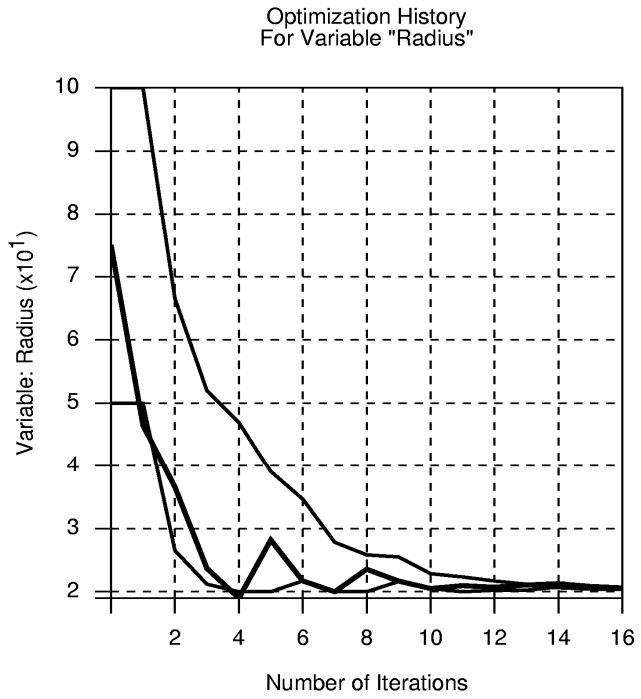
Because of the poor accuracy of the response surface fit for the rigid wall force above, it was decided to modify the force constraint so that the peak filtered force is used instead. Therefore, the previous response definition for `Rigid_Wall_Force` is replaced with a command that extracts the *maximum* rigid wall force from a response from which frequencies exceeding 300Hz are excluded.

The upper bound of the force constraint is changed to 80000.

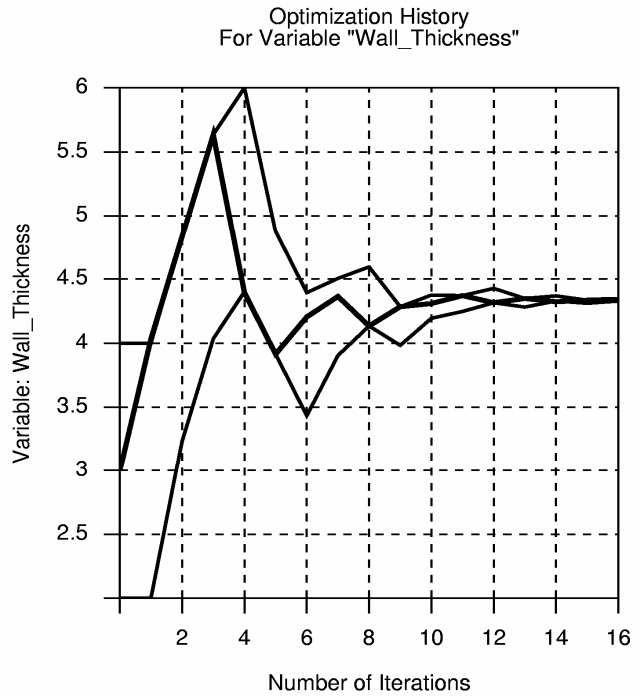
```
response 'Rigid_Wall_Force' "DynaASCII RWForc Normal 1 Max SAE 300"
```

20 iterations are specified with a 1% tolerance for convergence.

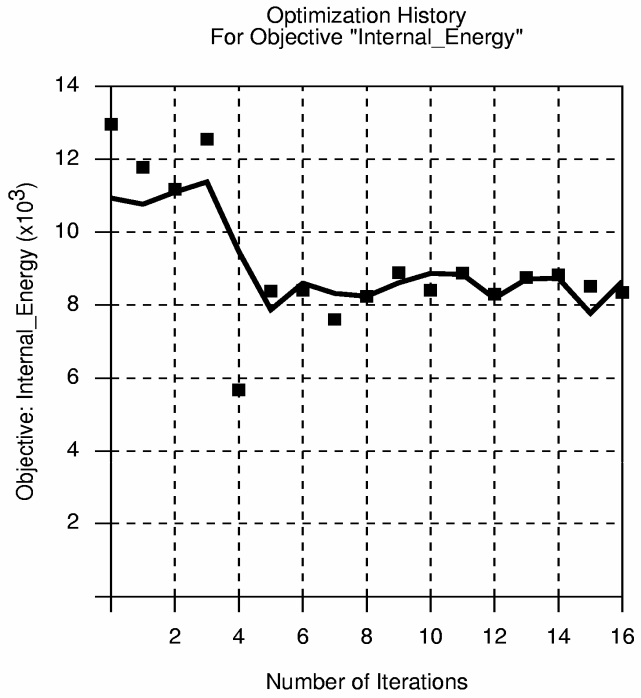
As expected, the response histories (Figure 22-20) show that the baseline design is severely infeasible (the first peak force is about 1.75×10^6 vs. the constraint value of 0.08×10^6. A steady reduction in the error of the response surfaces is observed up to about iteration 5. The optimization terminates after 16 iterations, having reached the 1% threshold for both objective and design variable changes.



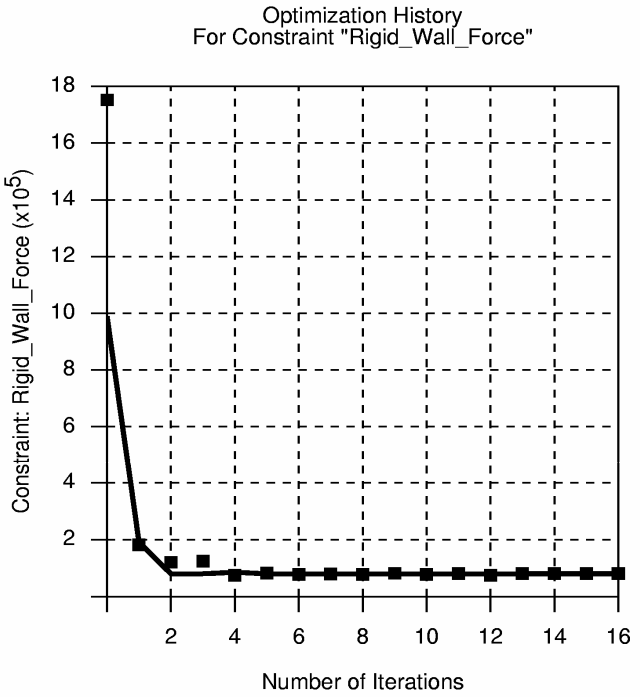
a) `Radius`



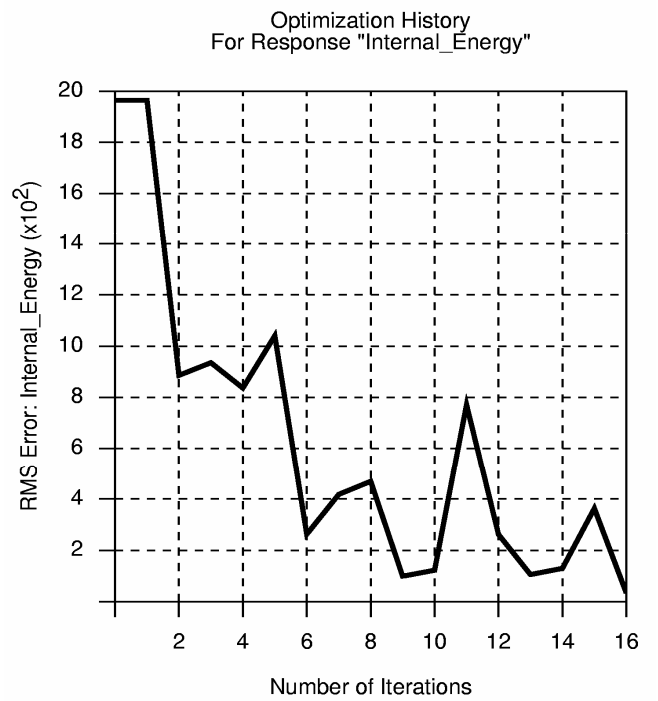
b) `Wall_Thickness`



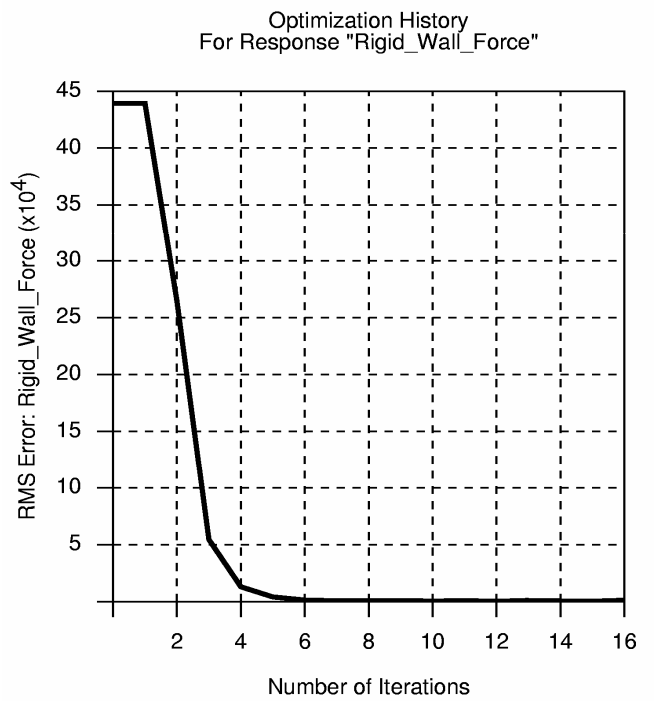
c) `Internal_Energy`



d) `Rigid_Wall_Force`



e) RMS error of `Internal_Energy`



f) RMS error of `Rigid_Wall_Force`

Figure 22-20: 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:
-------------------
                                 |       Scaled         |       Unscaled       |
                                 |----------------------|----------------------|
RESPONSE                         | Computed   Predicted| Computed   Predicted|
---------------------------------|----------|----------|----------|----------|
Internal_Energy                  |      8344       8645|      8344       8645|
Rigid_Wall_Force                 | 8.112e+04      8e+04| 8.112e+04      8e+04|
---------------------------------|----------|----------|----------|----------|
```

Figure 22-21 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.

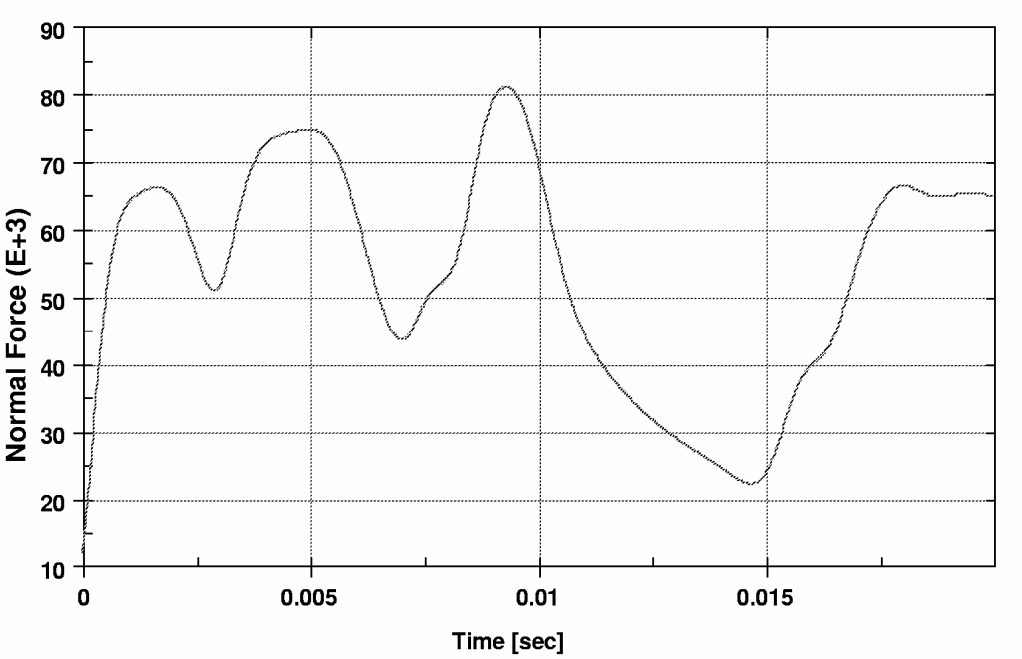


Figure 22-21: Cylinder: Constrained rigid wall force: F(t) < 80000 (SAE 300Hz filtered)

22.4 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[13].

22.4.1 Problem statement

The design parameterization for the sheet metal forming example is shown in Figure 22-22.

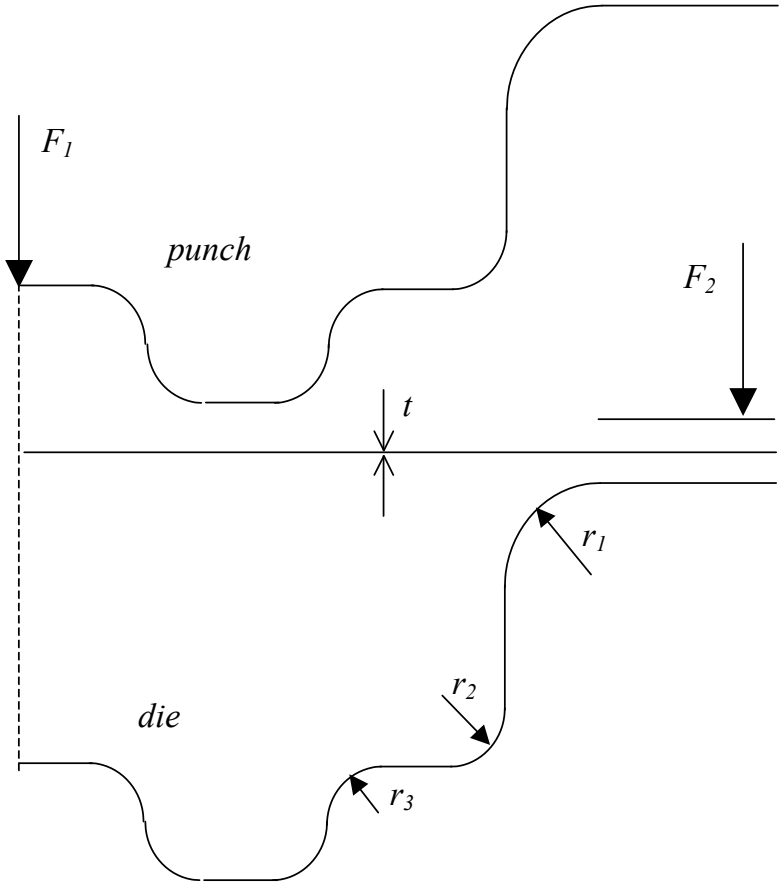


Figure 22-22: Parameterization of cross-section

[13] Registered Trademark of XYZ Scientific Applications Inc.

The FE model is shown in Figure 22-23.

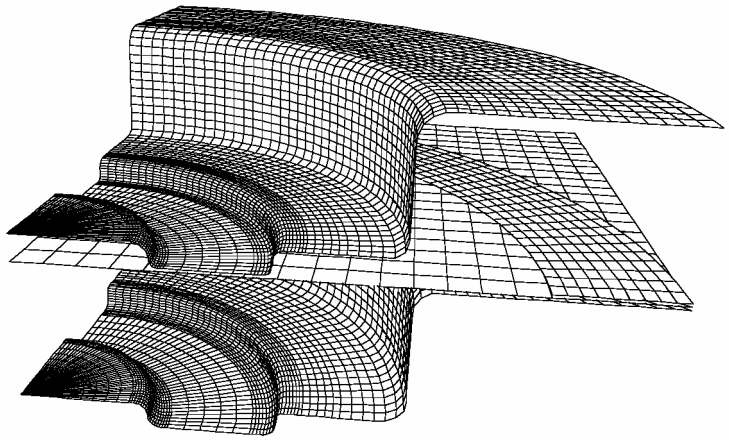


Figure 22-23: Quarter segment of FE model: tools and blank

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:

$$\text{Minimize } e$$

with

$$1.5 \le r_1 \le 4.5$$
$$1.5 \le r_2 \le 4.5$$
$$1.5 \le r_3 \le 4.5$$

subject to

$$g^{FLD}(\boldsymbol{x}) < 0.0$$
$$\Delta t(\boldsymbol{x}) < 20\%$$
$$r_1 - 1.1 < e$$
$$r_2 - 1.1 < e$$
$$r_3 - 1.1 < e$$
$$e > 0.$$

The design variables r_1, r_2 and r_3 are the radii of the work piece as indicated in Figure 22-22. Δt is the thickness reduction which is positive when the thickness is reduced. The FLD constraint is feasible when smaller than zero.

22.4.2 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 14.9.2).

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
          1           3           3           3                         1         1
   0.1000000                                                      0.000     0.000
         0.0
$
*CONTACT_FORMING_ONE_WAY_SURFACE_TO_SURFACE
$ workpiece vs blankholder
          1           4           3           3                         1         1
   0.1000000                                                      0.000     0.000
         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
    cmo con 4 7;
lsdymats 3 20 shell elfor bt rho 7.8e-9 e 2.e5 pr .28 shth .1
    cmo con 7 7 ;
lsdymats 4 20 shell elfor bt rho 7.8e-9 e 2.e5 pr .28 shth .1
    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 lsdsi a10 slvmat 1;mstmat 2;scoef .1 ; ; ;
c  sid 2 lsdsi a10 slvmat 1;mstmat 3;scoef .1 ; ; ;
c  sid 3 lsdsi a10 slvmat 1;mstmat 4;scoef .1 ; ; ;
c
lcd 1
     0.000000000E+00     0.275600006E+03
     0.665699990E-04     0.276100006E+03
     0.136500006E-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 lstl 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:
----------------------
                                |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                 |----------|----------|----------|----------|
                                |  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:
----------------------
                                |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                 |----------|----------|----------|----------|
                                |  Lower   |  Upper   |  Lower   |  Upper   |
--------------------------------|----------|----------|----------|----------|
FLD                             |    -          -      |    -          -      |
Rad1                            |    -           1.906|    -           1.906|
Rad2                            |    -           1.906|    -           1.906|
Rad3                            |    -           1.906|    -           1.906|
Thinning_scaled                 |    -         0.01718|    -          -      |
--------------------------------|----------|----------|----------|----------|
```

22.4.3 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
```

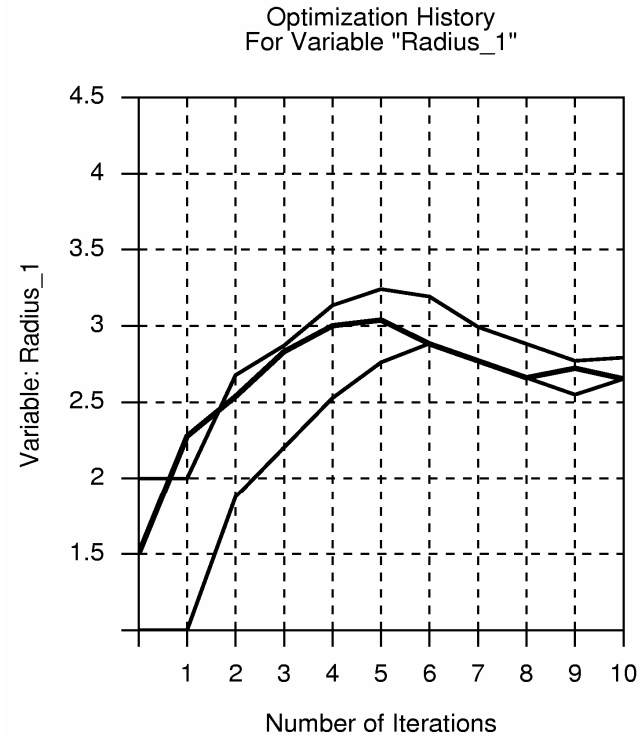
The number of *D*-optimal experiments is reduced because of the linear approximation used:

```
Order linear
 Experimental design dopt
 Basis experiment 3toK
 Number experiment 7
```

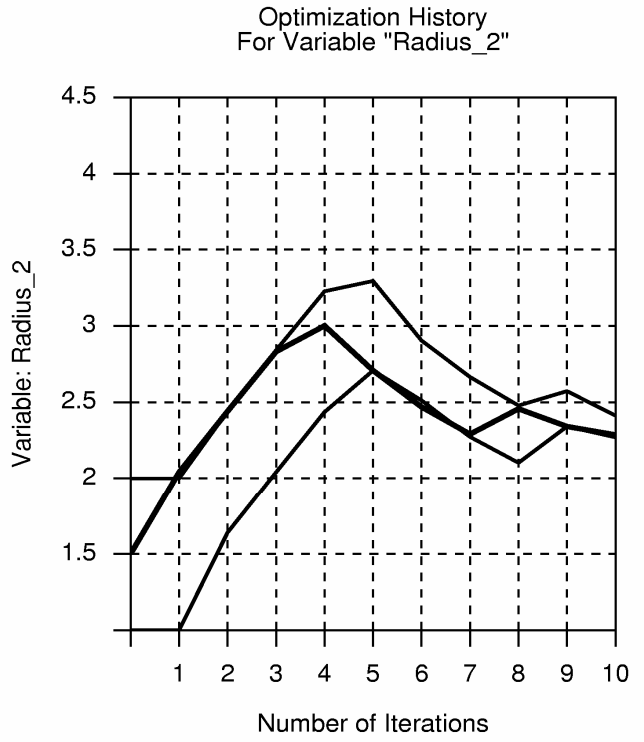
The optimization is run for 10 iterations:

```
iterate 10
```

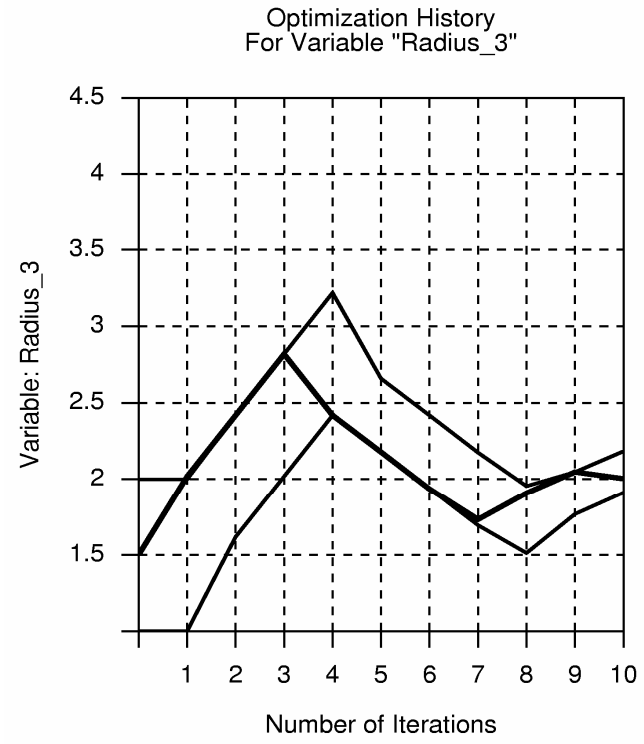
The optimization history is shown in Figure 22-24 for the design variables and responses:



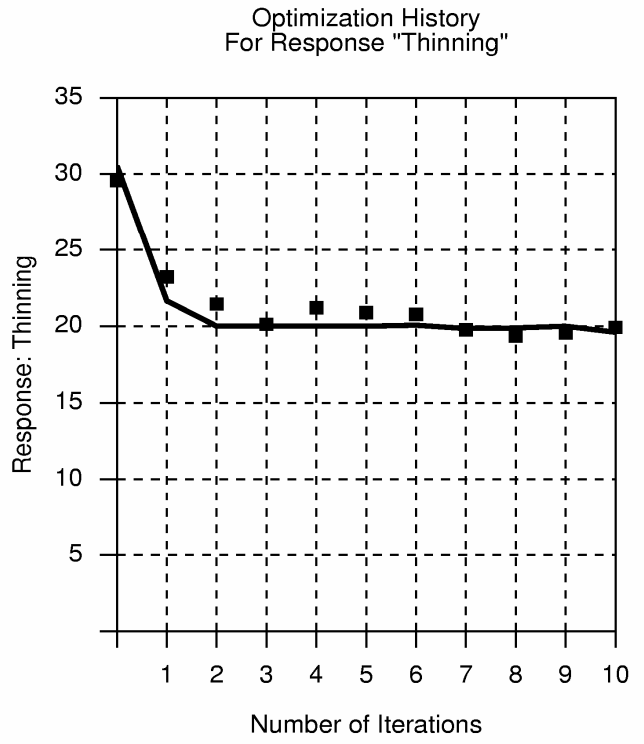
a) Optimization history of variable '`Radius_1`'



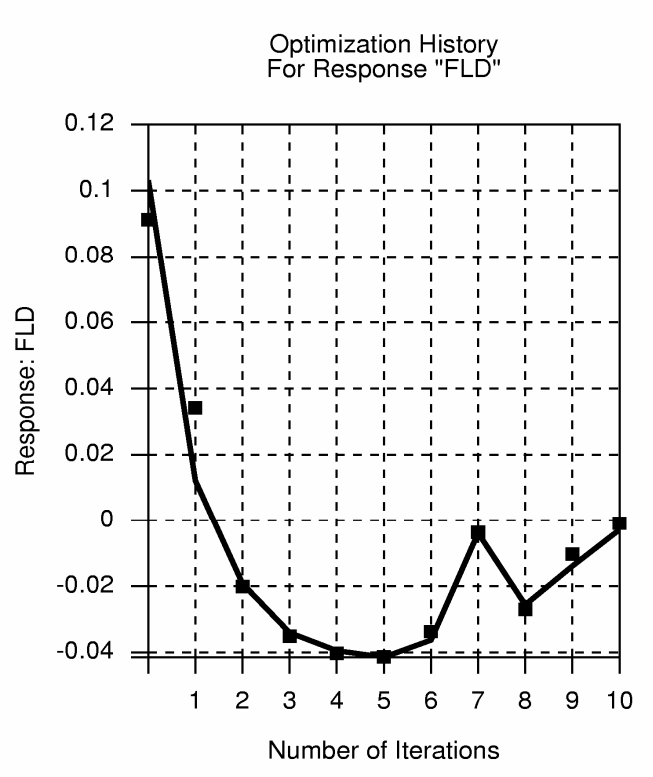
b) Optimization history of variable '`Radius_2`'



c) Optimization history of variable '`Radius_3`'



d) Optimization history of response '`Thinning`'



e) Optimization history of response `FLD`

Figure 22-24: Optimization history of design variables and responses (automated design)

The details of the 10th iteration have been extracted:

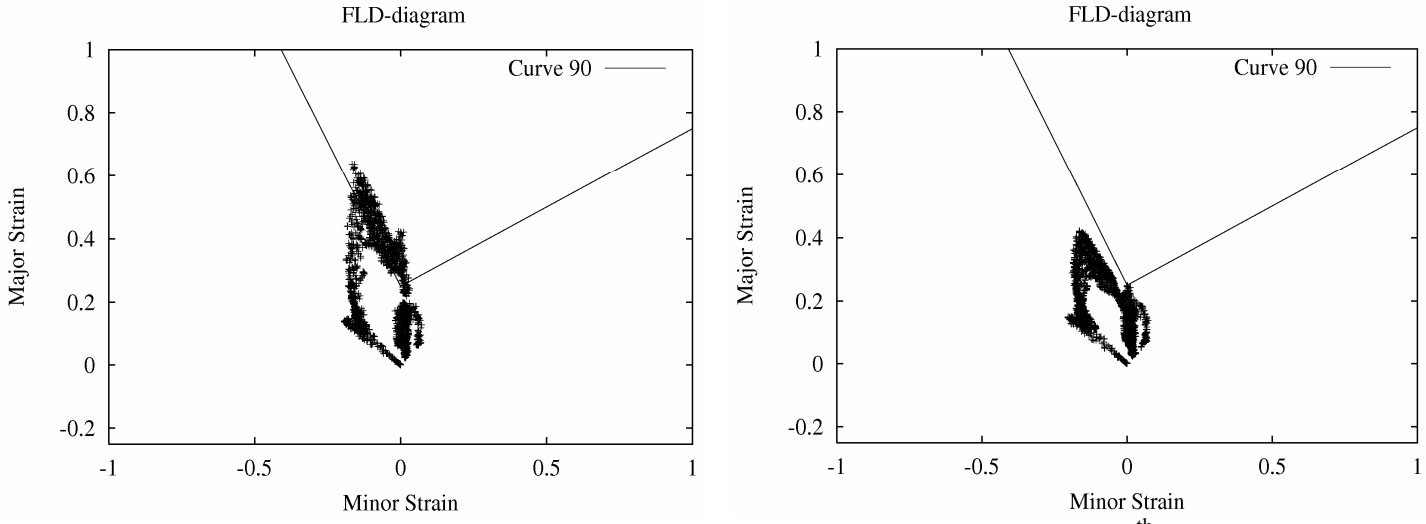
```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
--------------------------------|-----------|----------|-----------
Radius_1                                  1      2.653        4.5
Radius_2                                  1      2.286        4.5
Radius_3                                  1      2.004        4.5
--------------------------------|-----------|----------|-----------

RESPONSE FUNCTIONS:
-------------------
                                |       Scaled         |       Unscaled      |
                                |----------------------|---------------------|
RESPONSE                        | 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 22.4-1: 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 22-25) for the baseline design and the optimum illustrate the improvement of the FLD feasibility:



Baseline FLD diagram FLD diagram of 10th iteration

Figure 22-25: FLD diagrams of baseline and 10th iteration

A typical deformed state is depicted in Figure 22-26 below.

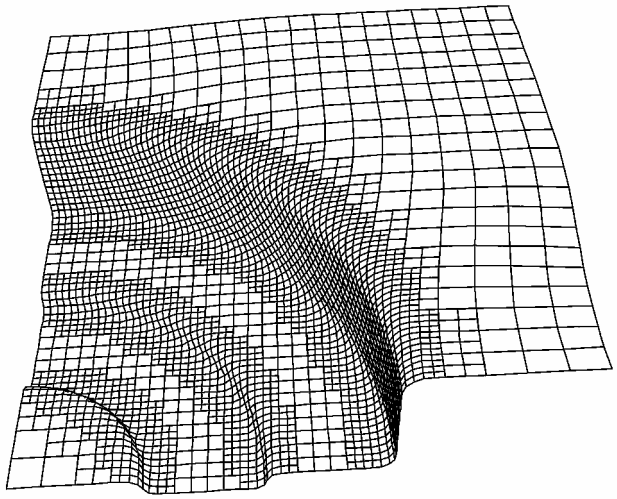


Figure 22-26: Deformed state

22.5 System identification (elastoplastic material) (2 variables)

A methodology for deriving system or material parameters from experimental results, known as system identification, is applied here using optimization. The example has the following features:

- The `MeanSqErr` composite function is used
- The `Crossplot` history is used
- The Min-Max formulation is demonstrated
- Multiple test cases are employed
- The confidence intervals of the optimal parameters is reported.

22.5.1 Problem statement

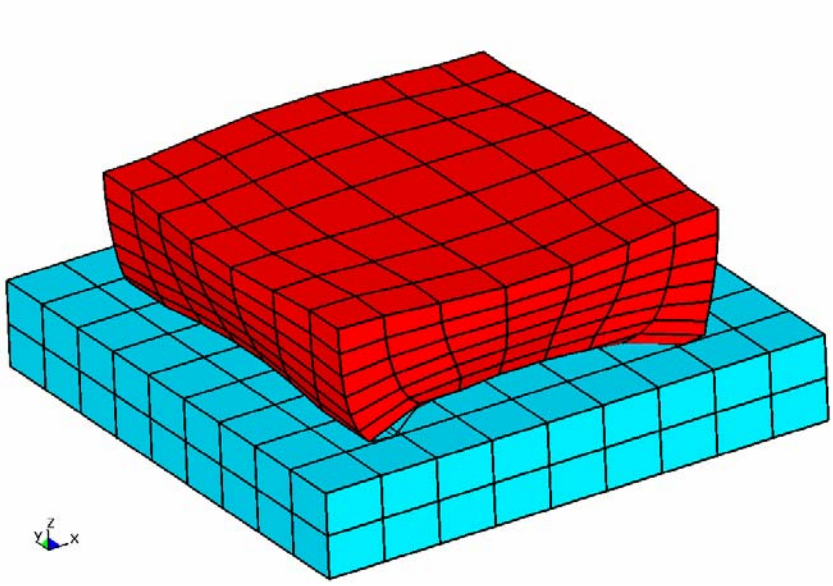


Figure 22-27: Sample of elastoplastic material subjected to a controlled vertical displacement

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 22-27). 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:

`Test1.txt`

```
0.36168 10162
0.72562 12964
1.0903  14840
1.4538  17672
```

Test2.txt

```
  0.36168 17393
  0.72562 19559
  1.0903  22098
  1.4538  26833
```

The finite element models for the two cases are represented in the keyword files `foam1.k` and `foam2.k` respectively.

Mean Squared Error (MSE) formulation

The LS-OPT command file is given below. The displacement and force histories are used to construct a force vs. displacement crossplot for the two cases. 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 `MeanSqErr` (Section 15.6) composite:

$$\varepsilon = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\boldsymbol{x}) - G_p}{s_p} \right)^2 = \frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{e_p(\boldsymbol{x})}{s_p} \right)^2$$

where P=10, $s_p = 1$ and $W_p = 1$. The representative MSE command is:

```
composite 'MSE1' { MeanSqErr ( Test1, Force_vs_Disp1, 10 ) }
```

```
"Example 6"
$ Created on Mon Nov 28 10:42:41 2005
solvers 2
$
$ WARNING -- NO RESPONSES ARE DEFINED
$
histories 8
$
$ DESIGN VARIABLES
$
variables 2
 Variable 'Youngs_Modulus' 700000
  Lower bound variable 'Youngs_Modulus' 500000
  Upper bound variable 'Youngs_Modulus' 2e+06
  Local 'Youngs_Modulus'
 Variable 'Yield_Stress' 1500
  Lower bound variable 'Yield_Stress' 500
  Upper bound variable 'Yield_Stress' 2000
$
$ CONSTANTS
$
constants 3
 Constant 'Begin' 0.002
 Constant 'End' 0.008
 Constant 'numpoints' 4

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Case1"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Case1"
$
 solver dyna960 'Case1'
  solver command "ls970.single"
  solver input file "foam1.k"
$ ------ Pre-processor --------
$   NO PREPROCESSOR SPECIFIED
$ ------ Sampling -------------
  solver order linear
  solver experiment design dopt
  solver basis experiment 5toK
$ ------ Job information ------
  solver concurrent jobs 1
$
$ WARNING - NO RESPONSES DEFINED FOR SOLVER "Case1"
$
$
$ HISTORIES FOR SOLVER "Case1"
$
 history 'Force1' "BinoutHistory -res_type rcforc -cmp z_force -id 1 -side SLAVE"
 history 'Disp1' "BinoutHistory -res_type nodout -cmp z_displacement -id 296"
$
$ HISTORY EXPRESSIONS FOR SOLVER "Case1"
$
 history 'Force_vs_Disp1' expression { Crossplot ("-Disp1", "Force1") }

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Case2"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Case2"
$
 solver dyna960 'Case2'
  solver command "ls970.single"
  solver input file "foam2.k"
$ ------ Pre-processor --------
$   NO PREPROCESSOR SPECIFIED
$ ------ Sampling -------------
  solver order linear
  solver experiment design dopt
  solver basis experiment 5toK
$ ------ Job information ------
  solver concurrent jobs 1
$
$ LOCAL DESIGN VARIABLES FOR SOLVER "Case2"
$
  solver variable 'Youngs_Modulus'
$
$ WARNING - NO RESPONSES DEFINED FOR SOLVER "Case2"
$
$
$ HISTORIES FOR SOLVER "Case2"
$
 history 'Force2' "BinoutHistory -res_type rcforc -cmp z_force -id 1 -side SLAVE"
 history 'Disp2' "BinoutHistory -res_type nodout -cmp z_displacement -id 288"
$
$ HISTORY EXPRESSIONS FOR SOLVER "Case2"
$
 history 'Force_vs_Disp2' expression { Crossplot ("-Disp2", "Force2") }

$
$ HISTORIES FROM FILES
$
 history 'Test1' file "Test1.txt"
 history 'Test2' file "Test2.txt"
composites 3
```

```
$
$ COMPOSITE EXPRESSIONS
$
 composite 'MSE1' { MeanSqErr ( Test1, Force_vs_Disp1, 10 ) }
 composite 'MSE2' { MeanSqErr ( Test2, Force_vs_Disp2, 10 ) }
 composite 'MSE' { sqrt(MSE1 + MSE2) }
$
$ OBJECTIVE FUNCTIONS
$
 objectives 2
 objective 'MSE1' 1
 objective 'MSE2' 1
$
$ THERE ARE NO CONSTRAINTS!!!
$
 constraints 0
$
$ JOB INFO
$
 concurrent jobs 1
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype and
 iterate 2
STOP
```

Maximum residual formulation

In this formulation, the deviations from the respective target values are incorporated as constraint violations, so that the optimization problem for parameter identification becomes:

$$\text{Minimize } e,$$

subject to

$$\left|\frac{f_j(\boldsymbol{x}) - G_j}{s_j}\right| \leq e; \quad j = 1,\ldots,8$$

$$e \geq 0$$

This formulation is automatically activated in LS-OPT by specifying the individual responses in equality constraints, i.e. without specifying the objective function as the maximum constraint violation. This is due to the fact LS-OPT automatically minimizes the infeasibility e, ignoring the objective function until a feasible design is found. When used in parameter identification, all the constraints are in general never completely satisfied due to typically over-determined systems that are used and therefore the objective function specification may be omitted.

As a method of second choice, the Minmax method presently requires a more laborious input preparation than the MSE approach. It will be simplified, using a single command, in a later version of LS-OPT.

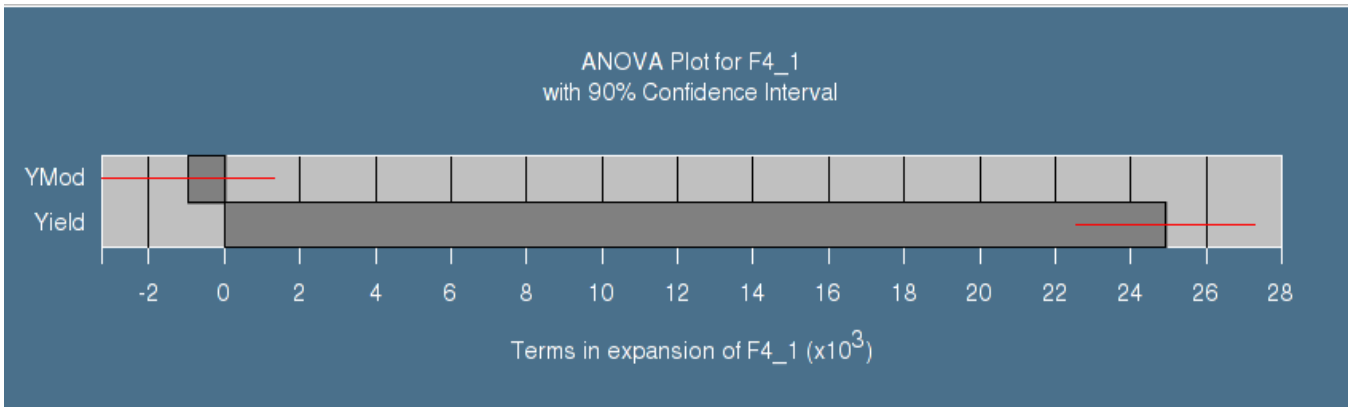
```
"Example 6c"
$ Created on Sun Apr  4 18:00:20 2004
solvers 2
responses 8
histories 2
$
$ DESIGN VARIABLES
$
```

```
variables 2
 Variable 'Youngs_Modulus' 700000
  Lower bound variable 'Youngs_Modulus' 500000
  Upper bound variable 'Youngs_Modulus' 2e+06
 Variable 'Yield_Stress' 1500
  Lower bound variable 'Yield_Stress' 500
  Upper bound variable 'Yield_Stress' 2000

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Case1"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Case1"
$
 solver dyna960 'Case1'
  solver command "ls970.single"
  solver input file "foam1.k"
  solver order linear
  solver experiment design dopt
  solver number experiments 5
  solver basis experiment 3toK
  solver concurrent jobs 1
$
$ WARNING - NO RESPONSES DEFINED FOR SOLVER "Case1"
$
$
$ HISTORIES FOR SOLVER "Case1"
$
 history 'Force1' "BinoutHistory -res_type rcforc -cmp z_force -id 1  -side SLAVE "
$
$ RESPONSE EXPRESSIONS FOR SOLVER "Case1"
$
 response 'F1_1' expression {Force1(0.002)}
 response 'F2_1' expression {Force1(0.004)}
 response 'F3_1' expression {Force1(0.006)}
 response 'F4_1' expression {Force1(0.008)}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Case2"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Case2"
$
 solver dyna960 'Case2'
  solver command "ls970.single"
  solver input file "foam2.k"
  solver order linear
  solver experiment duplicate 'Case1'
  solver concurrent jobs 1
$
$ WARNING - NO RESPONSES DEFINED FOR SOLVER "Case2"
$
$
$ HISTORIES FOR SOLVER "Case2"
$
 history 'Force2' "BinoutHistory -res_type rcforc -cmp z_force -id 1  -side SLAVE "
$
$ RESPONSE EXPRESSIONS FOR SOLVER "Case2"
$
 response 'F1_2' expression {Force2(0.002)}
 response 'F2_2' expression {Force2(0.004)}
 response 'F3_2' expression {Force2(0.006)}
 response 'F4_2' expression {Force2(0.008)}

composites 1
```

```
$
$ COMPOSITE RESPONSES
$
 composite 'Residual' type targeted
  composite 'Residual' response 'F1_1' 10162 scale 1
   weight 1
  composite 'Residual' response 'F2_1' 12964 scale 1
   weight 1
  composite 'Residual' response 'F3_1' 14840 scale 1
   weight 1
  composite 'Residual' response 'F4_1' 17672 scale 1
   weight 1
  composite 'Residual' response 'F1_2' 17393 scale 1
   weight 1
  composite 'Residual' response 'F2_2' 19559 scale 1
   weight 1
  composite 'Residual' response 'F3_2' 22098 scale 1
   weight 1
  composite 'Residual' response 'F4_2' 26833 scale 1
   weight 1
$
$ NO OBJECTIVES DEFINED
$
 objectives 0
$
$ CONSTRAINT DEFINITIONS
$
 constraints 8
 constraint 'F1_1'
  lower bound constraint 'F1_1' 10162
  upper bound constraint 'F1_1' 10162
 constraint 'F2_1'
  lower bound constraint 'F2_1' 12964
  upper bound constraint 'F2_1' 12964
 constraint 'F3_1'
  lower bound constraint 'F3_1' 14840
  upper bound constraint 'F3_1' 14840
 constraint 'F4_1'
  lower bound constraint 'F4_1' 17672
  upper bound constraint 'F4_1' 17672
 constraint 'F1_2'
  lower bound constraint 'F1_2' 17393
  upper bound constraint 'F1_2' 17393
 constraint 'F2_2'
  lower bound constraint 'F2_2' 19559
  upper bound constraint 'F2_2' 19559
 constraint 'F3_2'
  lower bound constraint 'F3_2' 22098
  upper bound constraint 'F3_2' 22098
 constraint 'F4_2'
  lower bound constraint 'F4_2' 26833
  upper bound constraint 'F4_2' 26833
$
$ JOB INFO
$
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype or
 iterate 5
STOP
```

22.5.2 Results

The results for both methods are compared below. Note that the optimum Young's modulus differs slightly due to its relative insignificance in the optimization as depicted in the following ANOVA plot representing the 4th point of the history plot and demonstrated by the size of its confidence interval (see table).



Mean Squared Error (MSE) formulation

Printout of the `lsopt_report` file:

```
=========================================================================
           M E A N   S Q U A R E D   E R R O R   V A L U E S

                            ITERATION 5
=========================================================================

-------------------------------------------
Objective name                       MSE
-------------------------------------------
MSE1                              .000221574
MSE2                              .000175544
-------------------------------------------
Total                             .000397118
-------------------------------------------


=========================================================================
           M E A N   S Q U A R E D   E R R O R   R E S I D U A L S

                           ITERATION 5
=========================================================================


   COMPOSITE : MSE1

     "Force_vs_Disp1"  calibrated to  "Test1"
-------------------------------------------------------------------------------------------------------

     Computed MSE Value  = 0.00026367
     Predicted MSE Value = 0.000221574

-------------------------------------------------------------------------------------------------------
       TEST DATA            |             COMPUTED RESULTS              |
------|----------|----------|----------|----------|----------|----------|----------|----------
 Point  Point       Target   | Computed   Computed | Predicted  Predicted|  Weight     Scale
 No.    Location    Value    | Value      Error    | Value      Error    |  Value      Value
------|----------|----------|----------|----------|----------|----------|----------|----------
     1     0.3617  1.016e+04| 1.027e+04      107.9| 1.026e+04      98.01|         1  1.767e+04
     2      0.483   1.11e+04|  1.08e+04     -298.9|  1.08e+04     -299.4|         1  1.767e+04
     3     0.6044  1.203e+04| 1.143e+04     -605.1| 1.157e+04       -458|         1  1.767e+04
     4     0.7257  1.296e+04| 1.283e+04     -129.6| 1.276e+04     -204.2|         1  1.767e+04
     5     0.8471  1.359e+04| 1.317e+04     -422.7| 1.314e+04       -447|         1  1.767e+04
     6     0.9684  1.421e+04| 1.397e+04       -240| 1.397e+04     -242.4|         1  1.767e+04
```

```
     7       1.09  1.484e+04|  1.49e+04      58.49|  1.49e+04      58.67|         1  1.767e+04
     8      1.211  1.578e+04| 1.592e+04      136.1| 1.591e+04        131|         1  1.767e+04
     9      1.332  1.673e+04| 1.688e+04      148.9| 1.674e+04      16.58|         1  1.767e+04
    10      1.454  1.767e+04| 1.743e+04     -243.2| 1.742e+04     -248.4|         1  1.767e+04
-------------------------------------------------------------------------------------------


   COMPOSITE : MSE2

     "Force_vs_Disp2"  calibrated to  "Test2"
-------------------------------------------------------------------------------------------

     Computed MSE Value  = 9.06349e-05
     Predicted MSE Value = 0.000175544

-------------------------------------------------------------------------------------------
       TEST DATA            |             COMPUTED RESULTS              |
------|----------|----------|----------|----------|----------|----------|----------|----------
 Point  Point       Target   | Computed   Computed | Predicted  Predicted|  Weight     Scale
 No.    Location    Value    | Value      Error    | Value      Error    |  Value      Value
------|----------|----------|----------|----------|----------|----------|----------|----------
     1     0.3617  1.739e+04| 1.753e+04      138.8| 1.762e+04      223.9|         1  2.683e+04
     2      0.483  1.812e+04| 1.823e+04      112.7| 1.824e+04      127.4|         1  2.683e+04
     3     0.6044  1.884e+04| 1.897e+04      130.5| 1.896e+04      121.3|         1  2.683e+04
     4     0.7257  1.956e+04| 1.973e+04      170.2| 1.972e+04      165.1|         1  2.683e+04
     5     0.8471   2.04e+04| 2.053e+04      120.7| 2.052e+04      118.2|         1  2.683e+04
     6     0.9684  2.125e+04| 2.137e+04      123.3| 2.137e+04      119.3|         1  2.683e+04
     7       1.09  2.209e+04| 2.228e+04        184| 2.228e+04      184.3|         1  2.683e+04
     8      1.211  2.367e+04| 2.438e+04      705.9| 2.471e+04       1037|         1  2.683e+04
     9      1.332  2.525e+04| 2.519e+04     -59.33| 2.539e+04      132.9|         1  2.683e+04
    10      1.454  2.683e+04| 2.674e+04     -95.55| 2.684e+04      5.068|         1  2.683e+04
-------------------------------------------------------------------------------------------

===========================================================================

===========================================================================
                C O N F I D E N C E   I N T E R V A L S

                             ITERATION 5
===========================================================================

-------------------------------------------------------------
 90% Confidence intervals for individual optimal parameters
-------------------------------------------------------------
Name                  Value              Confidence
                                          Interval
                                      Lower      Upper
-------------------------------------------------------------
Youngs_Modulus   739559.415      72970.5803 1406148.25
Yield_Stress     1009.14575      978.501323 1039.79017
-------------------------------------------------------------

===========================================================================
```

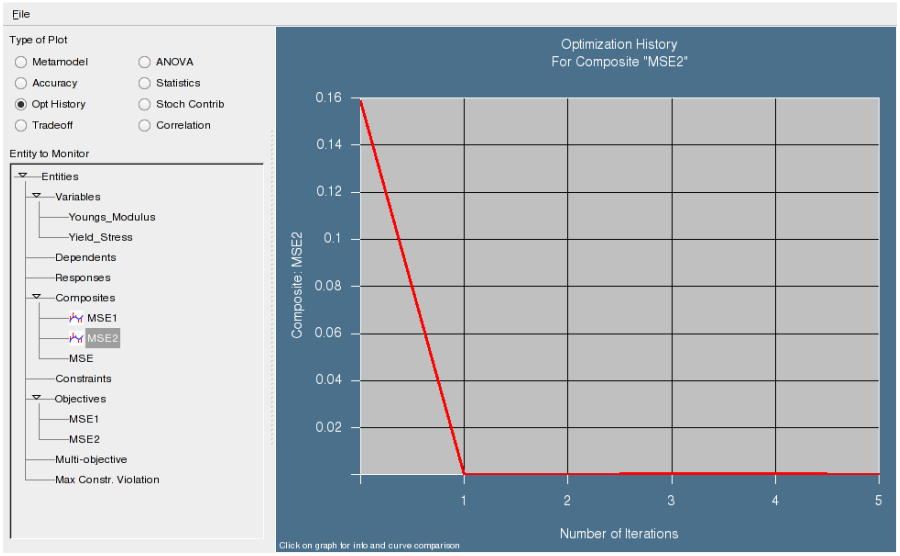


Figure 22-28: Optimization history of MSE2. The history plots comparing the response to the test data are selected by clicking near the selected iteration on the plot and then on the MeanSqErr button.

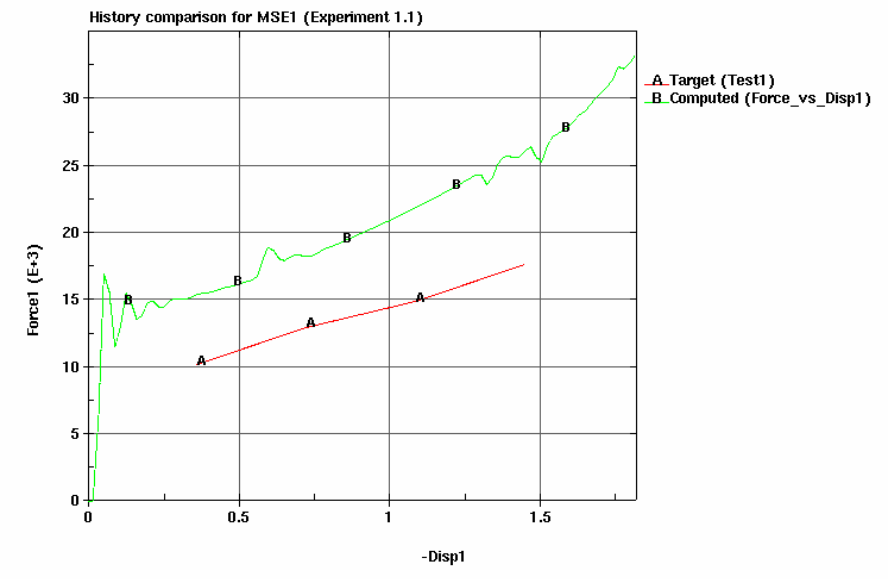


Figure 22-29: Comparison of force-displacement and data from Test1 (baseline)

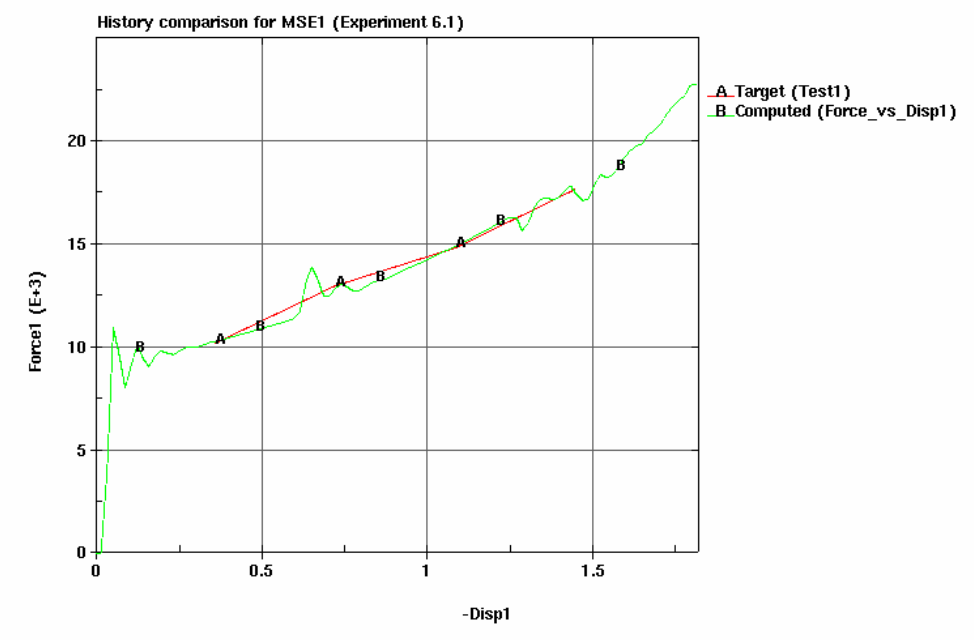


Figure 22-30: Comparison of force-displacement and data from Test1 (optimum)

Maximum residual formulation

```
DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
--------------------------------|-----------|----------|-----------|--------
Youngs_Modulus                         5e+05  7.083e+05       2e+06
Yield_Stress                             500       1001        2000
--------------------------------|-----------|----------|-----------|--------

RESPONSE FUNCTIONS:
-------------------
                                |       Scaled         |       Unscaled      |
                                |----------------------|---------------------|
RESPONSE                        | Computed   Predicted| Computed   Predicted|
--------------------------------|----------|----------|----------|----------|
F1_1                            |  1.02e+04   1.02e+04|  1.02e+04   1.02e+04|
F2_1                            | 1.273e+04  1.295e+04| 1.273e+04  1.295e+04|
F3_1                            | 1.478e+04  1.477e+04| 1.478e+04  1.477e+04|
F4_1                            | 1.735e+04  1.748e+04| 1.735e+04  1.748e+04|
F1_2                            | 1.743e+04  1.748e+04| 1.743e+04  1.748e+04|
F2_2                            | 1.957e+04  1.956e+04| 1.957e+04  1.956e+04|
F3_2                            |  2.21e+04  2.234e+04|  2.21e+04  2.234e+04|
F4_2                            | 2.653e+04  2.678e+04| 2.653e+04  2.678e+04|
--------------------------------|----------|----------|----------|----------|

COMPOSITE FUNCTIONS:
--------------------
COMPOSITE NAME                  | Computed | Predicted|
--------------------------------|----------|----------|
Residual                        |     505.2      332.9|
--------------------------------|----------|----------|


CONSTRAINT FUNCTIONS:
---------------------
CONSTRAINT NAME                 | Computed | Predicted|  Lower   |  Upper   |Viol?
--------------------------------|----------|----------|----------|----------|-----
F1_1                            |  1.02e+04   1.02e+04| 1.016e+04  1.016e+04|YES
```

F2_1	1.273e+04	1.295e+04	1.296e+04	1.296e+04	YES
F3_1	1.478e+04	1.477e+04	1.484e+04	1.484e+04	YES
F4_1	1.735e+04	1.748e+04	1.767e+04	1.767e+04	YES
F1_2	1.743e+04	1.748e+04	1.739e+04	1.739e+04	YES
F2_2	1.957e+04	1.956e+04	1.956e+04	1.956e+04	
F3_2	2.21e+04	2.234e+04	2.21e+04	2.21e+04	YES
F4_2	2.653e+04	2.678e+04	2.683e+04	2.683e+04	YES

CONSTRAINT VIOLATIONS:

CONSTRAINT NAME	Computed Violation		Predicted Violation	
	Lower	Upper	Lower	Upper
F1_1	-	37.3	-	35.91
F2_1	230.2	-	10.99	-
F3_1	61.33	-	65.56	-
F4_1	326.2	-	194.5	-
F1_2	-	40.46	-	85.06
F2_2	-	10.74	0.9383	-
F3_2	-	2.992	-	240.1
F4_2	298.1	-	49.21	-

MAXIMUM VIOLATION:

Quantity	Computed		Predicted	
	Constraint	Value	Constraint	Value
Maximum Violation	F4_1	326.2	F3_2	240.1
Smallest Margin	F3_2	2.992	F2_2	0.9383

22.6 Large vehicle crash and vibration (MDO/MOO) (7 variables)

(Example by courtesy of DaimlerChrysler)

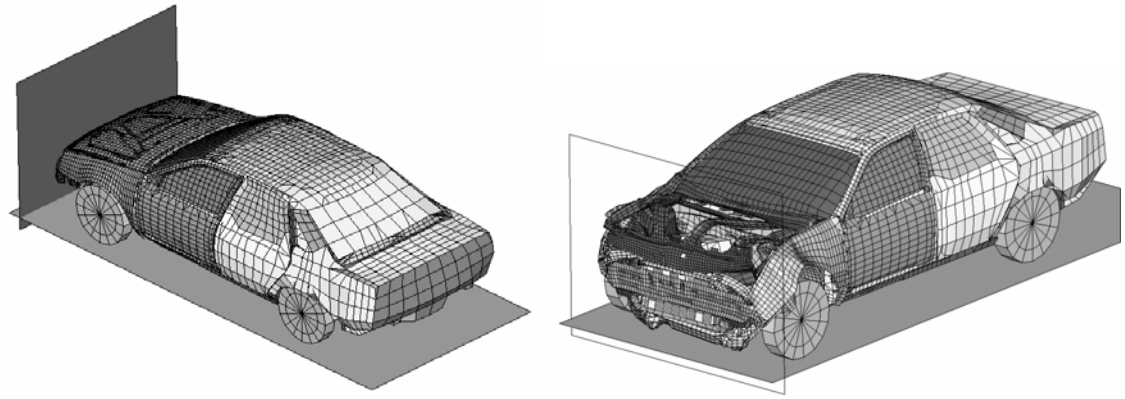
This example has the following features:

- 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.

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].

22.6.1 FE Modeling

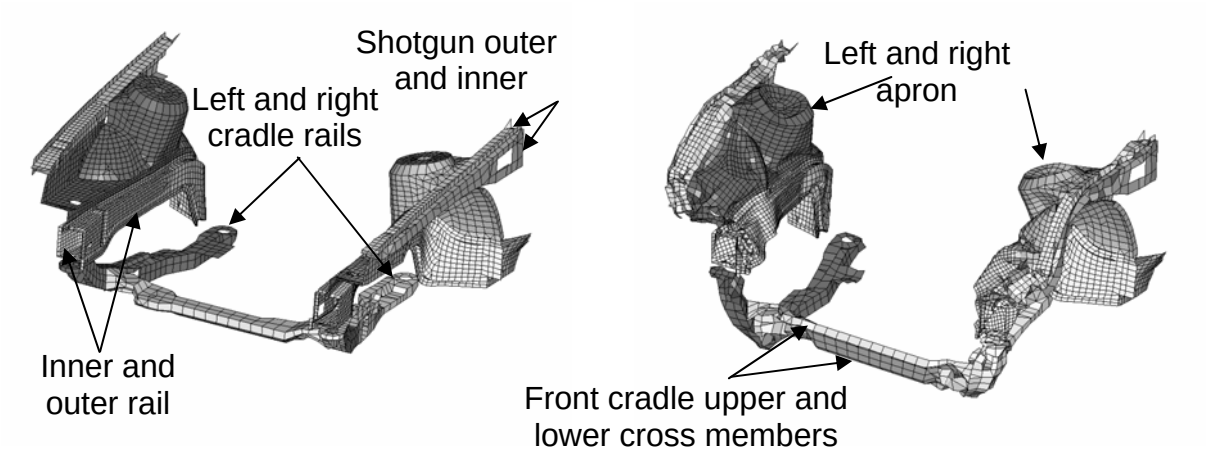
The crashworthiness simulation considers a model containing approximately 30,000 elements of a National Highway Transportation and Safety Association (NHTSA) vehicle [3] undergoing a full frontal impact. A modal analysis is performed on a so-called 'body-in-white' model containing approximately 18,000 elements. The crash model for the full vehicle is shown in Figure 22-31 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 22-32. The NVH model is depicted in Figure 22-33 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 22-32), parameterized directly in the LS-DYNA input file. Twelve parts are affected, comprising aprons, rails, shotguns, cradle rails and the cradle cross member (Figure 22-32). LS-DYNA v.971 is used for both the crash and NVH simulations, in explicit and implicit modes respectively.



(a) (b)

Figure 22-31: Crash model of vehicle showing road and wall

(a) Undeformed (b) Deformed (78ms)



(a) (b)

Figure 22-32: Structural components affected by design variables –

(a) Undeformed and (b) deformed (time = 72ms)

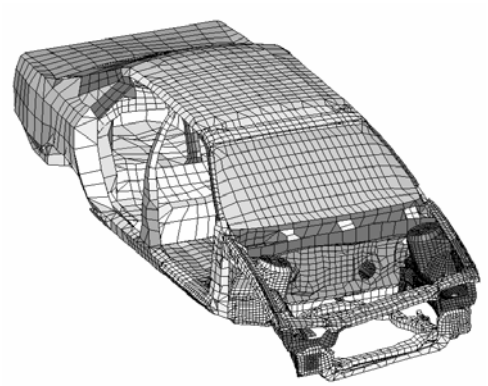


Figure 22-33: Body-in-white model of vehicle in torsional vibration mode (38.7Hz)

22.6.2 Design formulation

This example illustrates the following:

- Multidisciplinary optimization
- Discrete optimization
- Multi-objective optimization
- Complex mathematical expressions

The formulation is as follows:

Minimize Mass
Minimize Maximum intrusion

subject to

Maximum intrusion($\boldsymbol{x}_{crash}$) < 551.27mm

Stage 1 pulse($\boldsymbol{x}_{crash}$) > 14.51g
Stage 2 pulse($\boldsymbol{x}_{crash}$) > 17.59g
Stage 3 pulse($\boldsymbol{x}_{crash}$) > 20.75g

41.38Hz < Torsional mode frequency($\boldsymbol{x}_{NVH}$) < 42.38Hz

Variables:
$\boldsymbol{x}_{crash}$ = [*rail_inner*, *rail_outer*, *cradle_rails*, *aprons*, *shotgun_inner*, *shotgun_outer*, *cradle_crossmember*]^T
$\boldsymbol{x}_{NVH}$ = [*rail_inner*, *rail_outer*, *cradle_rails*, *aprons*, *shotgun_inner*, *shotgun_outer*, *cradle_crossmember*]^T.

The three stage pulses are calculated from the SAE filtered (60Hz) acceleration and displacement of a left rear sill node in the following fashion:

$$\text{Stage } i \text{ pulse} = \frac{-k}{d_2 - d_1} \int_{d_1}^{d_2} a\,dx \ ;$$

$$k = 0.5 \text{ for } i = 1,\ 1.0 \text{ otherwise;}$$

with the limits $[d_1;d_2] = [0;184]$; $[184;334]$; $[334;\text{Max(displacement)}]$ for $i = 1,2,3$ respectively, all displacement units in mm and the minus sign to convert acceleration to deceleration. The Stage 1 pulse is represented by a triangle with the peak value being the value used.

The constraints are scaled using the target values to balance the violations of the different constraints. This scaling 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.

22.6.3 Input preparation

The MDO and MOO features are specified as follows:

- **MDO.** The two disciplines (crash and NVH) are treated separately. Variables are flagged as local with the `Local` *`variable_name`* statement, and then linked to a solver using the `Solver variable` *`variable_name`* command.
- **MOO.** Two design objectives (Intrusion and mass) are stated. The weight of the mass has been set to 1.0 whereas the weight on the intrusion has been set to 0.0. These weights are specified in the "Objectives" panel of the GUI. This implies that the optimization path is based on minimal mass alone while the Pareto optimal front is constructed based on both objectives. The GA must be selected (also in the Objectives panel) as metamodel optimizer to obtain the Pareto optimal front.
- **Discrete variables.** These are specified as an array of space delimited values.

The command file is given below:

```
"Taurus Full Vehicle MDO : Crash and NVH, all variables"
Author "Ken Craig"
$ Created on Fri Feb  1 17:43:39 2008
solvers 2
responses 15
histories 2
$
$ DESIGN VARIABLES
$
variables 7
 Variable 'cradle_rails' 1.93
  Lower bound variable 'cradle_rails' 1
  Upper bound variable 'cradle_rails' 3
 Variable 'cradle_csmbr' 1.93
  Lower bound variable 'cradle_csmbr' 1
  Upper bound variable 'cradle_csmbr' 3
  Local 'cradle_csmbr'
 Variable 'shotgun_inner' 1.3
  Lower bound variable 'shotgun_inner' 1
  Upper bound variable 'shotgun_inner' 2.5
```

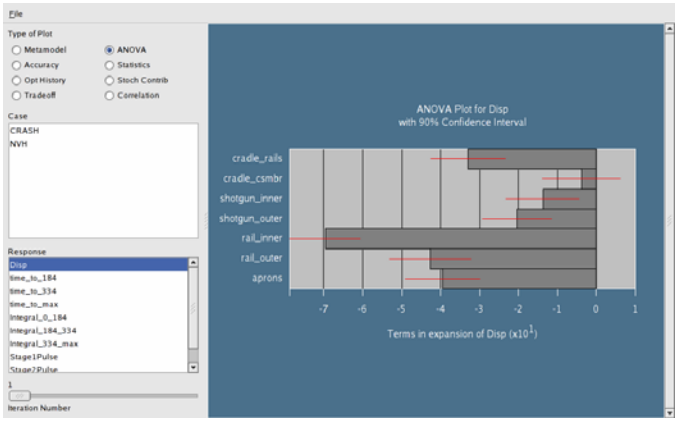
```
  Local 'shotgun_inner'
 Variable 'shotgun_outer' 1.3
  Lower bound variable 'shotgun_outer' 1
  Upper bound variable 'shotgun_outer' 2.5
  Local 'shotgun_outer'
 Variable 'rail_inner' 2
  Variable 'rail_inner' discrete {1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 }
 Variable 'rail_outer' 1.5
  Variable 'rail_outer' discrete {1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 }
 Variable 'aprons' 1.3
  Lower bound variable 'aprons' 1
  Upper bound variable 'aprons' 2.5
$
Optimization Method SRSM
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "CRASH"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "CRASH"
$
 solver dyna960 'CRASH'
  solver command "ls971_single"
  solver input file "taurus_mod.dyn"
  solver check output on
  solver compress d3plot off
$ ------ Pre-processor --------
$   NO PREPROCESSOR SPECIFIED
$ ------ Metamodeling ---------
  solver order RBF
  solver experiment design space_filling
   solver update doe
   solver alternate experiment 1
$ ------ Job information ------
  solver concurrent jobs 4
$
$ RESPONSES FOR SOLVER "CRASH"
$
 response 'Disp' 1 0 "BinoutResponse -res_type nodout -cmp x_displacement -id 26730  -select MAX "
$
$ HISTORIES FOR SOLVER "CRASH"
$
 history 'XDISP' "BinoutHistory -res_type nodout  -cmp x_displacement -id 26730"
 history 'XACCEL' "BinoutHistory -res_type nodout  -cmp x_acceleration -id 26730 -filter SAE -filter_freq
60.0000"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "CRASH"
$
 response 'time_to_184' expression {Lookup("XDISP(t)",184)}
 response 'time_to_334' expression {Lookup("XDISP(t)",334)}
 response 'time_to_max' expression {LookupMax("XDISP(t)")}
 response 'Integral_0_184' expression {Integral("XACCEL(t)",0,time_to_184,"XDISP(t)")}
 response 'Integral_184_334' expression {Integral("XACCEL(t)",time_to_184,time_to_334,"XDISP(t)")}
 response 'Integral_334_max' expression {Integral("XACCEL(t)",time_to_334,time_to_max,"XDISP(t)")}
 response 'Stage1Pulse' expression {(Integral_0_184/(-9810))*2/184}
 response 'Stage2Pulse' expression {(Integral_184_334/(-9810))/(334-184)}
 response 'Stage3Pulse' expression {(Integral_334_max/(-9810))/(Disp-334)}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "NVH"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "NVH"
$
 solver dyna960 'NVH'
  solver command "ls971_double"
  solver input file "taurus_biw.dyn"
  solver check file "/nexus4_1/nielen/LSOPT/___3.3___/RELEASE/TAURUS/CHECKPOINTS/NVH/AnalysisResults.1"
  solver check output on
  solver compress d3plot off
$ ------ Pre-processor --------
$   NO PREPROCESSOR SPECIFIED
```

```
$ ------ Metamodeling ---------
  solver order RBF
  solver experiment design space_filling
   solver update doe
   solver alternate experiment 1
$ ------ Job information ------
  solver concurrent jobs 2
$
$ LOCAL DESIGN VARIABLES FOR SOLVER "NVH"
$
  solver variable 'cradle_csmbr'
  solver variable 'shotgun_inner'
  solver variable 'shotgun_outer'
$
$ RESPONSES FOR SOLVER "NVH"
$
 response 'Vehicle_Mass_NVH' 2204.62 0 "DynaMass 29 30 32 33 34 35 79 81 82 83 MASS"
 response 'Frequency' 1 0 "DynaFreq 2 FREQ"
 response 'Mode' 1 0 "DynaFreq 2 NUMBER"
 response 'Generalized_Mass' 1 0 "DynaFreq 2 GENMASS"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "NVH"
$
 response 'Mass_scaled' expression {Vehicle_Mass_NVH/99.078}

composites 5
$
$ COMPOSITE RESPONSES
$
 composite 'Disp_scaled' type targeted
  composite 'Disp_scaled' response 'Disp' 0 scale 551.27
 composite 'Frequency_scaled' type targeted
  composite 'Frequency_scaled' response 'Frequency' 0 scale 41.8831
$
$ COMPOSITE EXPRESSIONS
$
 composite 'Stage1Pulse_scaled' {Stage1Pulse/14.512408}
 composite 'Stage2Pulse_scaled' {Stage2Pulse/17.586303}
 composite 'Stage3Pulse_scaled' {Stage3Pulse/20.745213}
$
$ OBJECTIVE FUNCTIONS
$
 objectives 2
 optimization algorithm GA
 objective 'Disp' 0
 objective 'Vehicle_Mass_NVH' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 5
 constraint 'Disp_scaled'
  upper bound constraint 'Disp_scaled' 1
 constraint 'Frequency_scaled'
  lower bound constraint 'Frequency_scaled' 0.9881
  upper bound constraint 'Frequency_scaled' 1.0119
 constraint 'Stage1Pulse_scaled'
  lower bound constraint 'Stage1Pulse_scaled' 1
 constraint 'Stage2Pulse_scaled'
  lower bound constraint 'Stage2Pulse_scaled' 1
 constraint 'Stage3Pulse_scaled'
  lower bound constraint 'Stage3Pulse_scaled' 1
$
$ JOB INFO
$
 concurrent jobs 4
 iterate param design 0
 iterate param objective 0
 iterate param stoppingtype and
 iterate 10
STOP
```

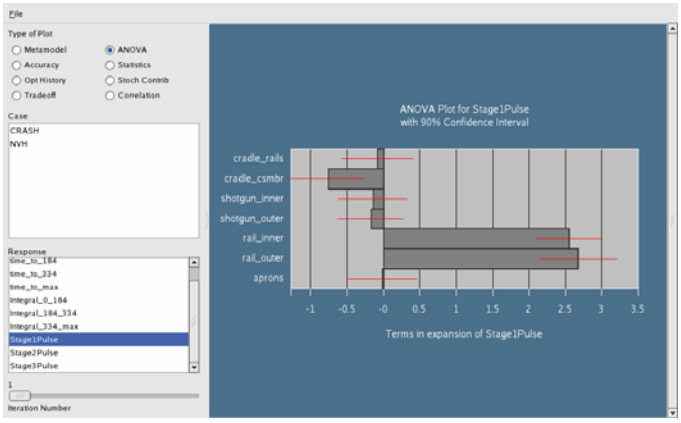
22.6.4 Variable screening

The plots below show the ANOVA charts for the 7 design responses. The plots are based on a single iteration with a linear approximation and *D*-optimality criterion as sampling scheme. From these plots, the most important subsets of variables are chosen. All the variables are kept for the NVH analysis because of its relatively small computational cost.

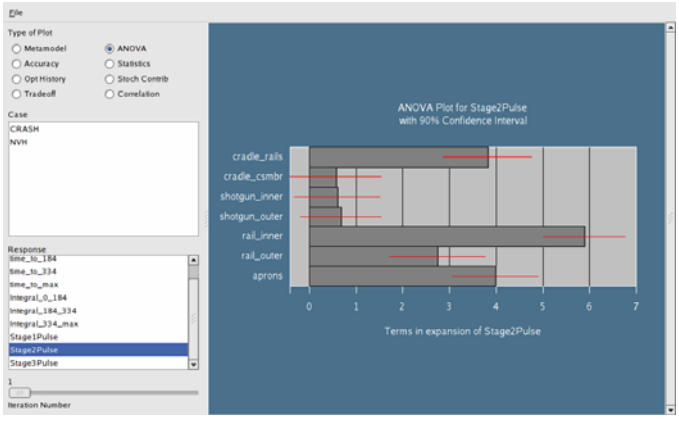
$\boldsymbol{x}_{\text{crash}} = [rail\_inner, rail\_outer, cradle\_rails, aprons]^{T}$;
$\boldsymbol{x}_{\text{NVH}} = [rail\_inner, rail\_outer, cradle\_rails, aprons, shotgun\_inner, shotgun\_outer, cradle\_crossmember]^{T}$.



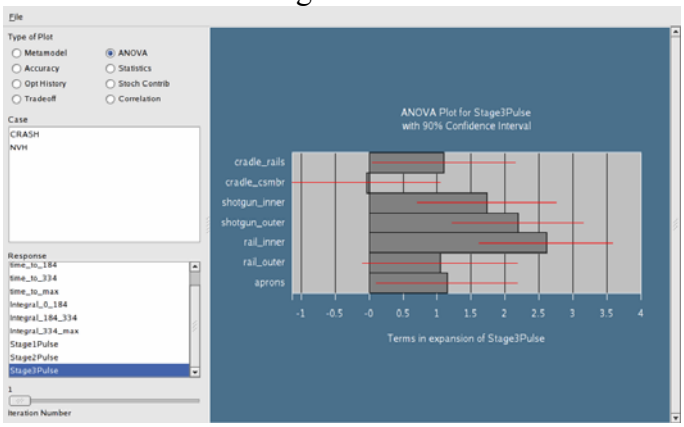
Intrusion



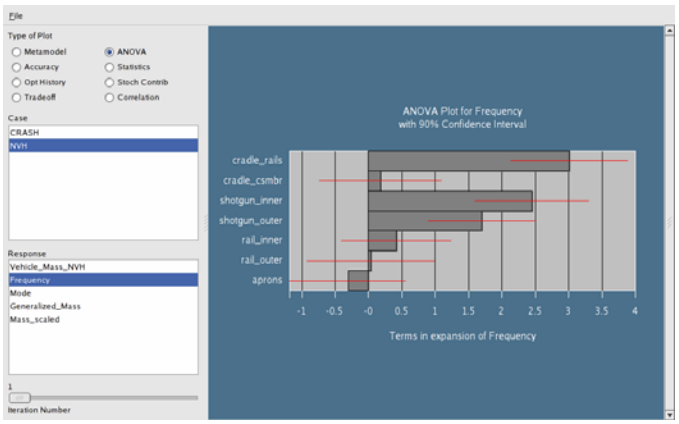
Stage 1 Pulse



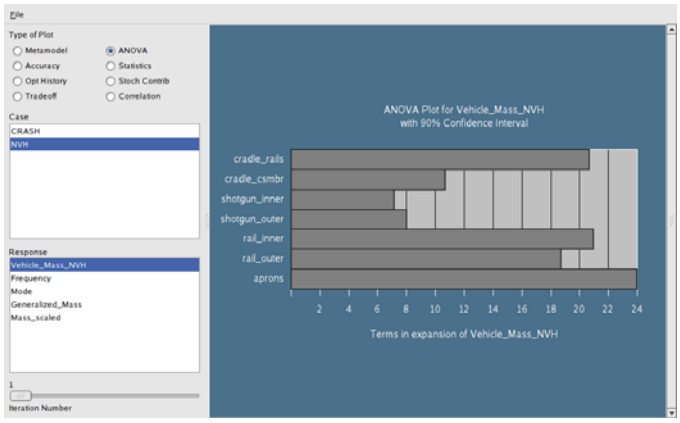
Stage 2 Pulse



Stage 3 Pulse



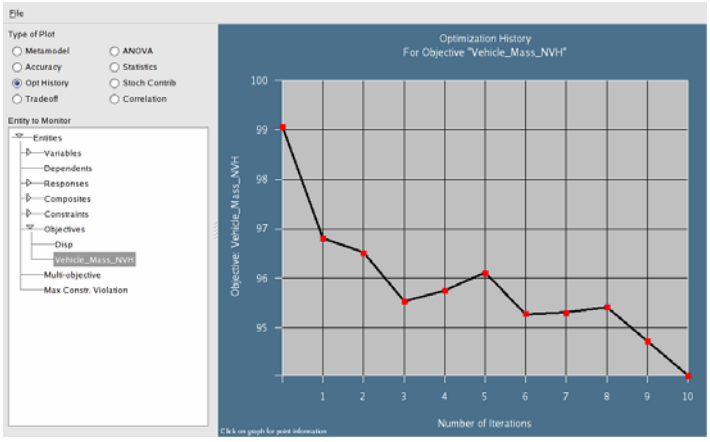
Frequency



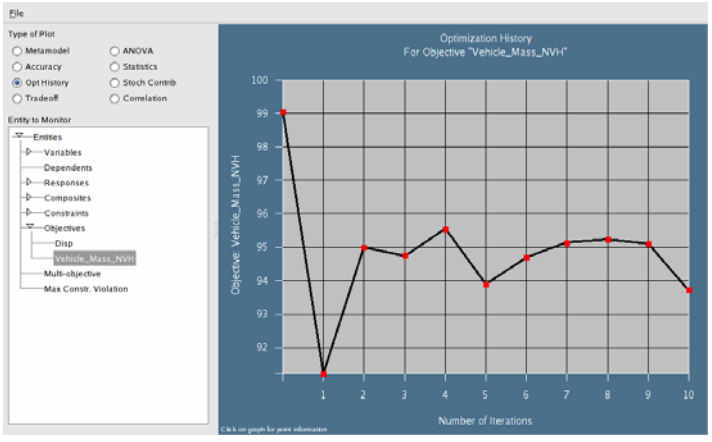
Mass

22.6.5 Optimization history results and Pareto optimal front

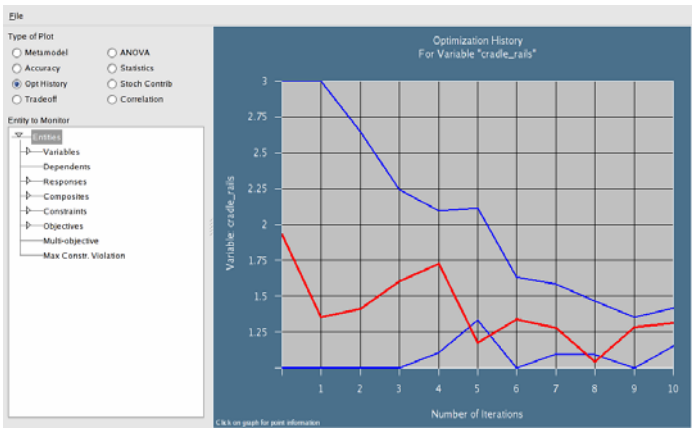
The figure shows the optimization history of the mass (objective function). For the purpose of comparison, two optimization runs were conducted, one with the full variable set and the other with the screened variables. Note the similarity of the minimal mass for both cases (*Note:* Scale of *y*-axis is different). The history of the cradle rails is also shown. The blue lines represent the upper and lower bounds of the region of interest for this variable. The plot (bottom right) shows the Pareto optimal front for the two objectives. The final two plots depict the Mass function in the [inner rail, outer rail] and [aprons, outer rail] spaces respectively with constraint isolines. The feasible region is green whereas the infeasible regions have increasingly darker shades of red as more constraints are violated.



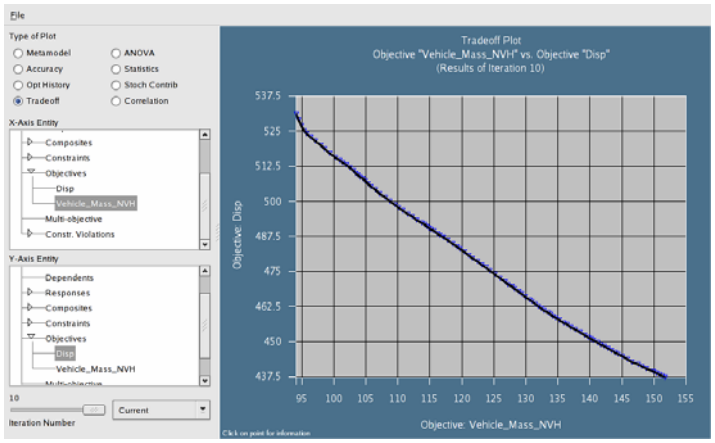
Optimization history of mass for full set of variables



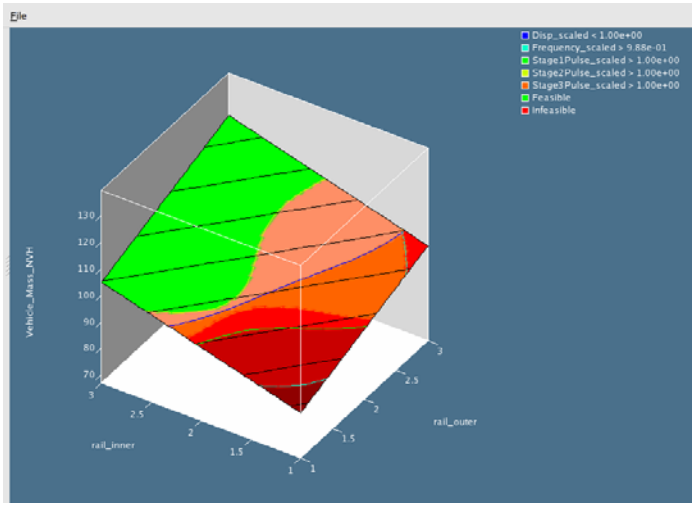
Optimization history of mass for screened variables



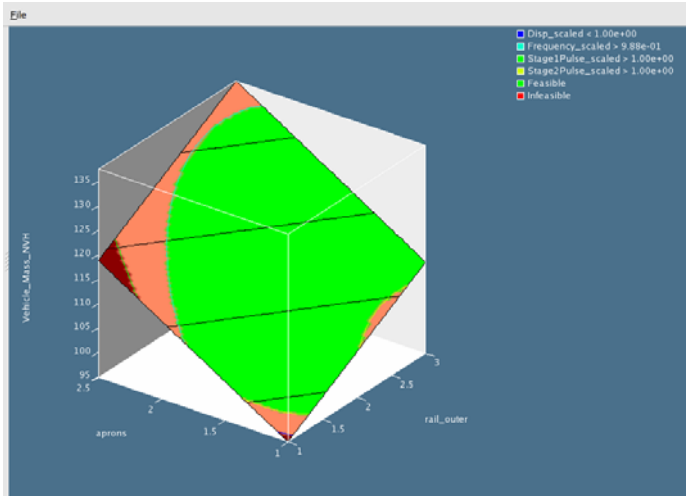
Optimization history of cradle_rails variable



Pareto optimal front: Mass vs. Intrusion



Constraint isolines superimposed on Mass approximation. Feasible region in green. Darker red shades for increasing number of violations. Space shown in [*inner rail*; outer rail]



Constraint isolines superimposed on Mass approximation. Feasible region in green. Darker red shades for increasing number of violations. Space shown in [*aprons*; outer rail]

22.6.6 Summary of results

The file reported below is the `lsopt_report` file which is viewable using the View→Summary GUI selection in the top menu bar and can also be found in the main working directory. The gradient information (derivatives of the responses with respect to the variables) is also available from the file, but is omitted here for brevity.

```
LS-OPT Version          : 3.3
LS-OPT Revision         : 42875
LS-OPT Version Date     : Feb 1, 2008

File name               : lsopt_report
This file created on    : Fri Feb  1 19:51:07 2008
```

```
Project Command File  : lsopt_db

************************************************************************
Problem description:
 Taurus Full Vehicle MDO : Crash and NVH, all variables
************************************************************************

------------------------------------------------------------------------
              N U M B E R S   O F   E A C H   E N T I T Y
------------------------------------------------------------------------
Number of design variables ..............................     7
Number of response functions ............................    15
Number of constraint functions ..........................     5
Number of objective functions ...........................     2
------------------------------------------------------------------------
                  D E S I G N   V A R I A B L E   D A T A
------------------------------------------------------------------------

Continuous Variables
-------------------------------------------|
Variable Name     Lower Bound   Upper Bound
---------------|-------------|-------------|
cradle_rails               1           3
cradle_csmbr               1           3
shotgun_inner              1         2.5
shotgun_outer              1         2.5
aprons                     1         2.5
------------------------------------------------------------------------

Discrete Variables
-----------------------------------
Variable Name    Discrete Values
---------------|-------------------
rail_inner       1 1.25 1.5 1.75 2 2.25 2.5 2.75 3
rail_outer       1 1.25 1.5 1.75 2 2.25 2.5 2.75 3
------------------------------------------------------------------------------

------------------------------------------------------------------------
                   O B J E C T I V E   F U N C T I O N S
------------------------------------------------------------------------
Objective ..................... MINIMIZE

-------------------------------------------
Objective name                   Weights
-------------------------------------------
Disp                             0
Vehicle_Mass_NVH                 1

-------------------------------------------

------------------------------------------------------------------------
                  C O N S T R A I N T   F U N C T I O N S
------------------------------------------------------------------------
Constraint name                  Lower Bound Upper Bound
------------------------------------------------------------------------
Disp_scaled                      -1e+30      1
Frequency_scaled                 0.9881      1.012
Stage1Pulse_scaled               1           1e+30
Stage2Pulse_scaled               1           1e+30
Stage3Pulse_scaled               1           1e+30
------------------------------------------------------------------------
------------------------------------------------------------------------
                     O P T I M I Z A T I O N   A L G O R I T H M
------------------------------------------------------------------------
 Method ............................ Sequential Response Surface Method
 Optimization Algorithm ............. Genetic Algorithm
------------------------------------------------------------------------

================================================================
```

```
              ------------------------------
              | Evaluating Starting Design |
              |        ITERATION  1        |
              ------------------------------


          C O M P U T E D  vs.  P R E D I C T E D
          ---------------------------------------

Using Metamodel of Iteration 1

-----------------------------------------------


DESIGN POINT
------------
Variable Name                     Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------|--------
cradle_rails                                1       1.93           3
cradle_csmbr                                1       1.93           3
shotgun_inner                               1        1.3         2.5
shotgun_outer                               1        1.3         2.5
rail_inner                                  1          2           3
rail_outer                                  1        1.5           3
aprons                                      1        1.3         2.5
---------------------------------|-----------|----------|-----------|--------


RESPONSE FUNCTIONS:
-------------------
RESPONSE                          | Computed   Predicted|
----------------------------------|----------|----------|
Disp                              |     551.6      552.4|
time_to_184                       |   0.01476    0.01476|
time_to_334                       |   0.02893    0.02883|
time_to_max                       |   0.07095    0.06982|
Integral_0_184                    |-1.318e+07 -1.308e+07|
Integral_184_334                  |-2.609e+07 -2.472e+07|
Integral_334_max                  |-4.407e+07 -4.539e+07|
Stage1Pulse                       |     14.61       14.5|
Stage2Pulse                       |     17.73       16.8|
Stage3Pulse                       |     20.65      21.24|
Vehicle_Mass_NVH                  |     99.06      99.06|
Frequency                         |     41.88      40.83|
Mode                              |         2      2.022|
Generalized_Mass                  |         1     0.9825|
Mass_scaled                       |    0.9998     0.9998|
----------------------------------|----------|----------|

COMPOSITE FUNCTIONS:
--------------------
COMPOSITE NAME                    | Computed | Predicted|
----------------------------------|----------|----------|
Disp_scaled                       |     1.001      1.002|
Frequency_scaled                  |         1     0.9748|
Stage1Pulse_scaled                |     1.007     0.9988|
Stage2Pulse_scaled                |     1.008     0.9554|
Stage3Pulse_scaled                |    0.9952      1.024|
----------------------------------|----------|----------|


               OBJECTIVE:
               ----------
Computed Value  =      99.06
Predicted Value =      99.06


OBJECTIVE FUNCTIONS:
--------------------
```

OBJECTIVE NAME	Computed	Predicted	WT.
Disp	551.6	552.4	0
Vehicle_Mass_NVH	99.06	99.06	1

CONSTRAINT FUNCTIONS:

CONSTRAINT NAME	Computed	Predicted	Lower	Upper	Viol?
Disp_scaled	1.001	1.002	-1e+30	1	YES
Frequency_scaled	1	0.9748	0.9881	1.012	
Stage1Pulse_scaled	1.007	0.9988	1	1e+30	
Stage2Pulse_scaled	1.008	0.9554	1	1e+30	
Stage3Pulse_scaled	0.9952	1.024	1	1e+30	YES

CONSTRAINT VIOLATIONS:

CONSTRAINT NAME	Computed Violation		Predicted Violation	
	Lower	Upper	Lower	Upper
Disp_scaled	-	0.0006012	-	0.002016
Frequency_scaled	-	-	0.01335	-
Stage1Pulse_scaled	-	-	0.00118	-
Stage2Pulse_scaled	-	-	0.04461	-
Stage3Pulse_scaled	0.004776	-	-	-

MAXIMUM VIOLATION:

Quantity	Computed		Predicted	
	Constraint	Value	Constraint	Value
Maximum Violation	Stage3Pulse_scaled	0.004776	Stage2Pulse_scaled	0.04461
Smallest Margin	Disp_scaled	0.0006012	Stage1Pulse_scaled	0.00118

E R R O R M E A S U R E S F O R R E S P O N S E S

ITERATION 10

Response Name	Metamodel type	RMS Error	RMS Error (% of mean)	Maximum Residual	Sq. Root PRESS	Sq. Root PRESS (% of mean)	R-Sq.
Disp	RBF Net	3.53	0.658	10.7	4.44	0.826	0.967
time_to_184	RBF Net	1.64e-06	0.0111	1.09e-05	6.8e-06	0.046	0.989
time_to_334	RBF Net	2.81e-05	0.0964	8.69e-05	4.03e-05	0.138	0.987
time_to_max	RBF Net	0.000511	0.75	0.00102	0.000734	1.08	0.945
Integral_0_184	RBF Net	5.24e+04	0.38	3.12e+05	1.2e+05	0.869	0.992
Integral_184_334	RBF Net	2.73e+05	1.03	7.28e+05	4.48e+05	1.69	0.985
Integral_334_max	RBF Net	2.7e+05	0.629	7.12e+05	3.78e+05	0.881	0.989
Stage1Pulse	RBF Net	0.0581	0.38	0.345	0.133	0.873	0.992
Stage2Pulse	RBF Net	0.186	1.03	0.502	0.304	1.68	0.985
Stage3Pulse	RBF Net	0.344	1.59	0.872	0.42	1.94	0.871
Vehicle_Mass_NVH	RBF Net	0.00707	0.00691	0.0187	0.00743	0.00726	1
Frequency	RBF Net	0.0463	0.111	0.148	0.139	0.332	0.994
Mode	RBF Net	0.14	6.47	0.594	0.236	10.9	0.822
Generalized_Mass	RBF Net	0.0434	4.43	0.247	0.0465	4.75	0.313
Mass_scaled	RBF Net	7.14e-05	0.00691	0.000189	7.5e-05	0.00726	1

```
          F I N A L   D E S I G N
          ITERATION 11
================================================


DESIGN POINT
------------
Variable Name                    Lower Bound   Value    Upper Bound
---------------------------------|-----------|----------|-----------|--------
cradle_rails                               1      1.576           3
cradle_csmbr                               1          1           3 Active
shotgun_inner                              1      1.584         2.5
shotgun_outer                              1      1.416         2.5
rail_inner                                 1        2.5           3
rail_outer                                 1        1.5           3
aprons                                     1      1.054         2.5
---------------------------------|-----------|----------|-----------|--------

RESPONSE FUNCTIONS:
-------------------
RESPONSE                         | Computed   Predicted|
---------------------------------|----------|----------|
Disp                             |     537.4      540.2|
time_to_184                      |   0.01479    0.01479|
time_to_334                      |   0.02908    0.02904|
time_to_max                      |   0.06765    0.06839|
Integral_0_184                   |-1.418e+07 -1.415e+07|
Integral_184_334                 | -2.58e+07 -2.588e+07|
Integral_334_max                 | -4.32e+07  -4.32e+07|
Stage1Pulse                      |     15.71      15.68|
Stage2Pulse                      |     17.53      17.59|
Stage3Pulse                      |     21.65       21.4|
Vehicle_Mass_NVH                 |     93.72      93.71|
Frequency                        |     41.37      41.41|
Mode                             |         2      2.008|
Generalized_Mass                 |    0.9933     0.9925|
Mass_scaled                      |    0.9459     0.9458|
---------------------------------|----------|----------|

COMPOSITE FUNCTIONS:
--------------------
COMPOSITE NAME                   | Computed | Predicted|
---------------------------------|----------|----------|
Disp_scaled                      |    0.9748     0.9798|
Frequency_scaled                 |    0.9878     0.9888|
Stage1Pulse_scaled               |     1.083      1.081|
Stage2Pulse_scaled               |     0.997          1|
Stage3Pulse_scaled               |     1.044      1.032|
---------------------------------|----------|----------|


              OBJECTIVE:
              ---------
Computed Value  =      93.72
Predicted Value =      93.71


OBJECTIVE FUNCTIONS:
--------------------
OBJECTIVE NAME                   | Computed   Predicted  WT.
---------------------------------|----------|----------|----
Disp                             |     537.4      540.2|   0
Vehicle_Mass_NVH                 |     93.72      93.71|   1
---------------------------------|----------|----------|----

CONSTRAINT FUNCTIONS:
---------------------
CONSTRAINT NAME                  | Computed | Predicted|  Lower   |  Upper   |Viol?
---------------------------------|----------|----------|----------|----------|-----
Disp_scaled                      |    0.9748     0.9798|    -1e+30          1|
Frequency_scaled                 |    0.9878     0.9888|    0.9881      1.012|YES
```

```
Stage1Pulse_scaled              |     1.083      1.081|         1      1e+30|
Stage2Pulse_scaled              |     0.997          1|         1      1e+30|YES
Stage3Pulse_scaled              |     1.044      1.032|         1      1e+30|
--------------------------------|----------|----------|----------|----------|-----


CONSTRAINT VIOLATIONS:
----------------------
                                |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                 |----------|----------|----------|----------|
                                |  Lower   |  Upper   |  Lower   |  Upper   |
--------------------------------|----------|----------|----------|----------|
Disp_scaled                     |    -          -     |    -          -     |
Frequency_scaled                | 0.0003405     -     |    -          -     |
Stage1Pulse_scaled              |    -          -     |    -          -     |
Stage2Pulse_scaled              |  0.003032     -     |    -          -     |
Stage3Pulse_scaled              |    -          -     |    -          -     |
--------------------------------|----------|----------|----------|----------|


MAXIMUM VIOLATION:
------------------
                    |         Computed          |        Predicted          |
     Quantity       |---------------------------|---------------------------|
                    |   Constraint      Value   |   Constraint      Value   |
--------------------|---------------|----------|---------------|----------|
Maximum Violation   |Stage2Pulse_scaled   0.003032|Disp_scaled               0|
Smallest Margin     |Frequency_scaled  0.0003405|Stage2Pulse_scaled  0.0001336|
--------------------|---------------|----------|---------------|----------|


                        ANALYSIS COMPLETED

                                    Sat Feb  2 18:33:23 2008
```

22.6.7 Multi-objective optimization using Direct GA simulation

Next, this MDO problem is solved to study the trade-off between mass and intrusion. The problem statement is given as:

Minimize Mass
Minimize Maximum intrusion

subject to

Stage 1 pulse($\boldsymbol{x}_{\text{crash}}$) > 14.51g
Stage 2 pulse($\boldsymbol{x}_{\text{crash}}$) > 17.59g
Stage 3 pulse($\boldsymbol{x}_{\text{crash}}$) > 20.75g

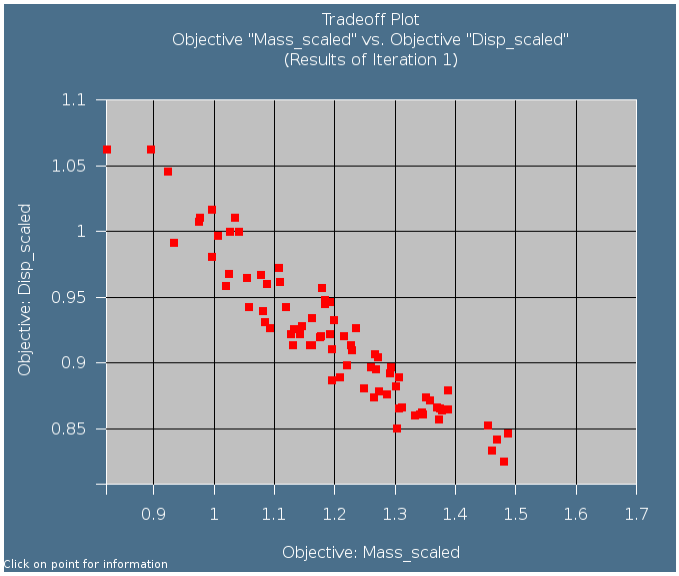
41.38Hz < Torsional mode frequency($\boldsymbol{x}_{\text{NVH}}$) < 42.38Hz *(Fully-shared variables)*

The problem is solved using direct GA simulations. For this problem, all seven design variables were used for both disciplines. The NSGA-II algorithm (*MOEA*) was used in conjunction with real encoding of design variables. Tournament selection operator (*Selection*), with a tournament size of two (*Tourn Size*), was used to remove individuals with low fitness values. The simulated binary crossover (*Real Crossover Type*) and mutation operators were used to create child populations. The distribution index for crossover and mutation

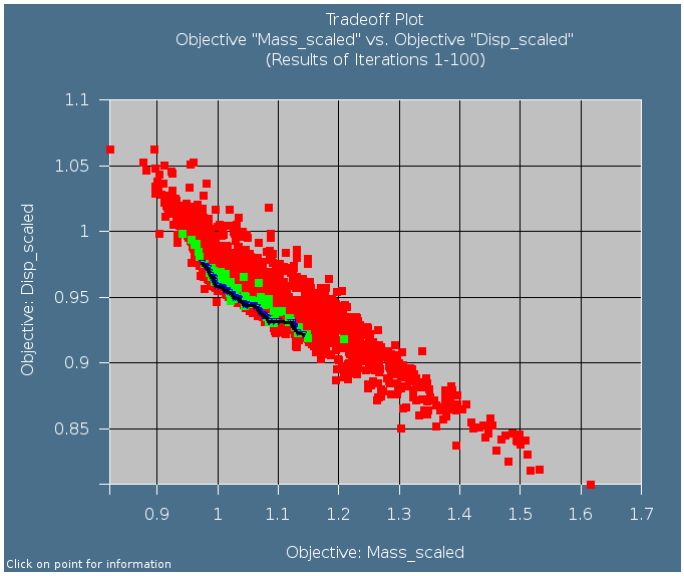
were taken as 5 (*Crossover Distribution Index, Mutation Distribution Index*). The trade-off files were generated at each generation (*Restart Interval*). The GA parameters are implemented as follows:

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method GA

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Genetic Algorithm Parameters
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 GA Parameter Popsize 80
 GA Parameter Generation 100
 GA Parameter MOEA 1
 GA Parameter Selection 1
 GA Parameter Tourn Size 2
 GA Parameter Real Crossover Type 1
 GA Parameter Real Crossover Probability 0.99
 GA Parameter Real Crossover Distribution Index 5.0
 GA Parameter Real Mutation Probability 0.15
 GA Parameter Real Mut Dist Index 5.0
 GA Parameter Restart Status 0
 GA Parameter Restart Interval 1
 GA Parameter Seed 854526

$
$ COMPOSITE RESPONSES
$
 composite 'Disp_scaled' type targeted
  composite 'Disp_scaled' response 'Disp' 0 scale 551.27
 composite 'Frequency_scaled' type targeted
  composite 'Frequency_scaled' response 'Frequency' 0 scale 41.8831
$
$ COMPOSITE EXPRESSIONS
$
 composite 'Stage1Pulse_scaled' {Stage1Pulse/14.512408}
 composite 'Stage2Pulse_scaled' {Stage2Pulse/17.586303}
 composite 'Stage3Pulse_scaled' {Stage3Pulse/20.745213}
$
$ OBJECTIVE FUNCTIONS
$
 objectives 2
 objective 'Mass_scaled' 1
 objective 'Disp_scaled' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 4
 constraint 'Frequency_scaled'
  lower bound constraint 'Frequency_scaled' 0.9881
  upper bound constraint 'Frequency_scaled' 1.0119
 constraint 'Stage1Pulse_scaled'
  lower bound constraint 'Stage1Pulse_scaled' 1
 constraint 'Stage2Pulse_scaled'
  lower bound constraint 'Stage2Pulse_scaled' 1
 constraint 'Stage3Pulse_scaled'
  lower bound constraint 'Stage3Pulse_scaled' 1
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

The outcome of the optimization is shown in Figure 22-34. Initial population did not have any feasible design but after running for 100 generations, the population resulted into 81 unique candidate Pareto optimal designs. These designs are shown by blue dots (connected by the line) on the left hand-side.



Initial population – no feasible design

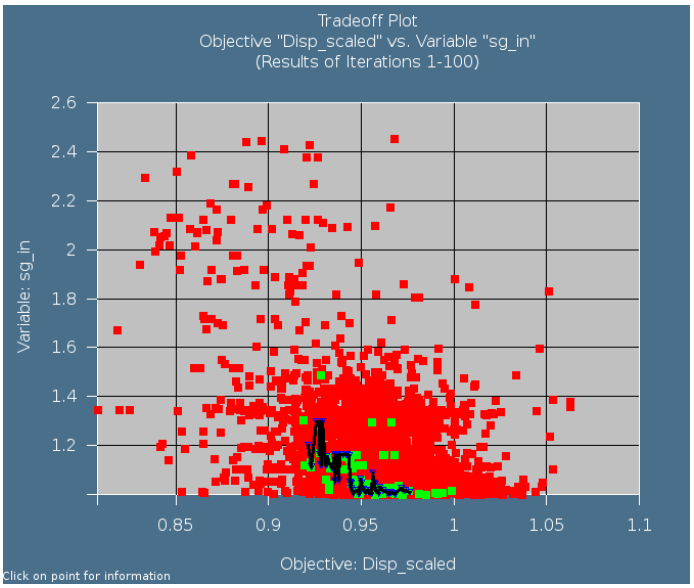


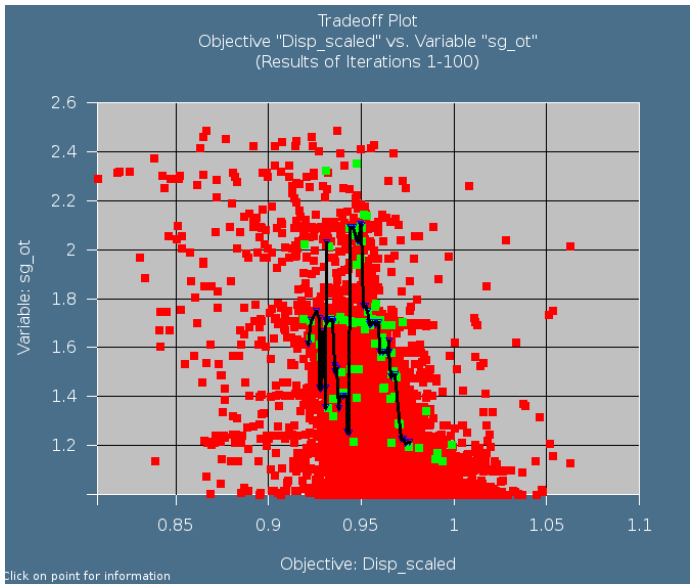
Final population after 100 generations

Figure 22-34: Tradeoff between mass and scaled intrusion (displacement) – Initial population is shown on the left hand side and the final population is shown on the right hand side graph. The non-dominated solutions are shown by blue triangles (connected by a line on right hand side graph).

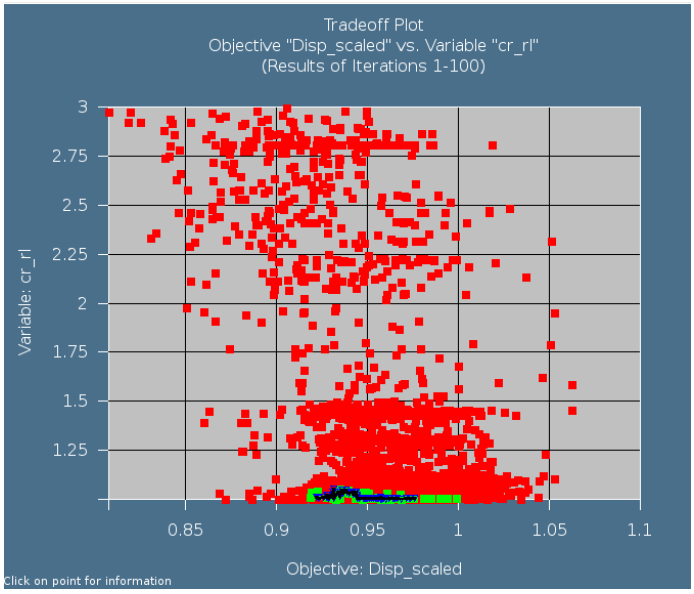
The results show that the potential of improvement by using multi-objective optimization. Trade-off between the two objectives show that intrusion can be reduced by increasing the mass. The trade-off curve clearly illustrates that reduction in intrusion (from 0.922 to 0.976) might require proportionate increase in mass (from 0.974 to 1.14). A trade-off design (0.974, 0.976) can achieve nearly 2.5% reduction in both objectives.

The optimal design variables corresponding to different designs on the candidate Pareto optimal front are shown in Figure 22-35. Quite interestingly, the variations in different design variables is fairly small.

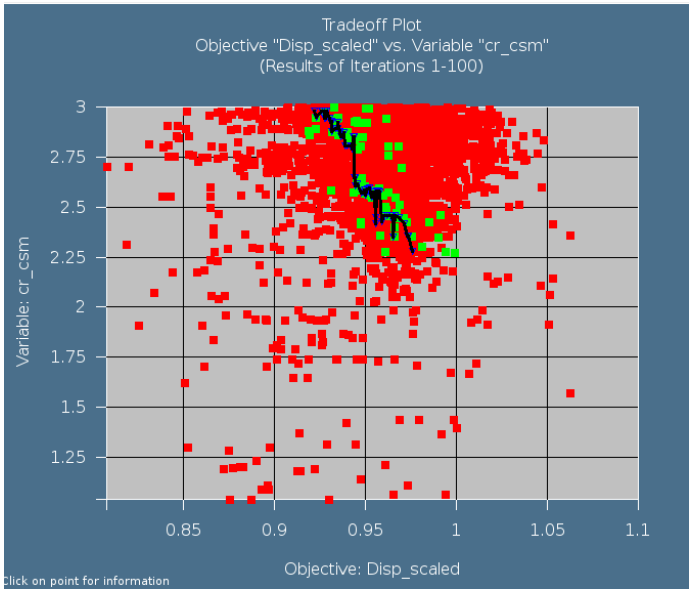




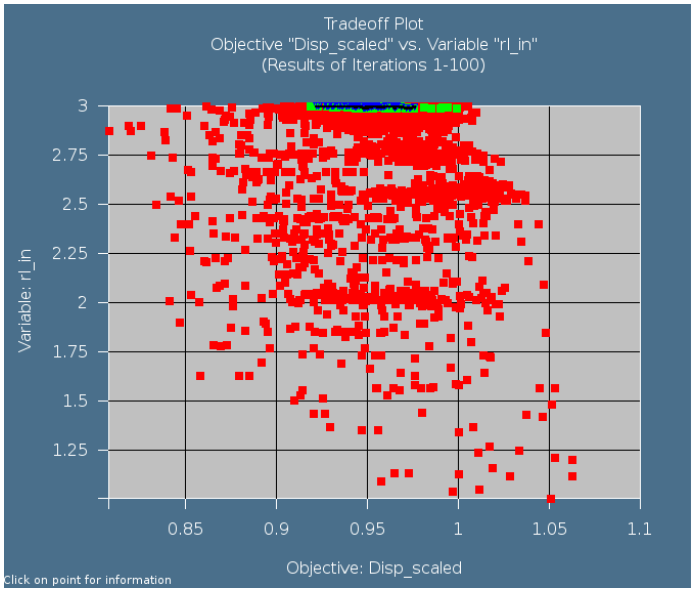
Inner shotgun



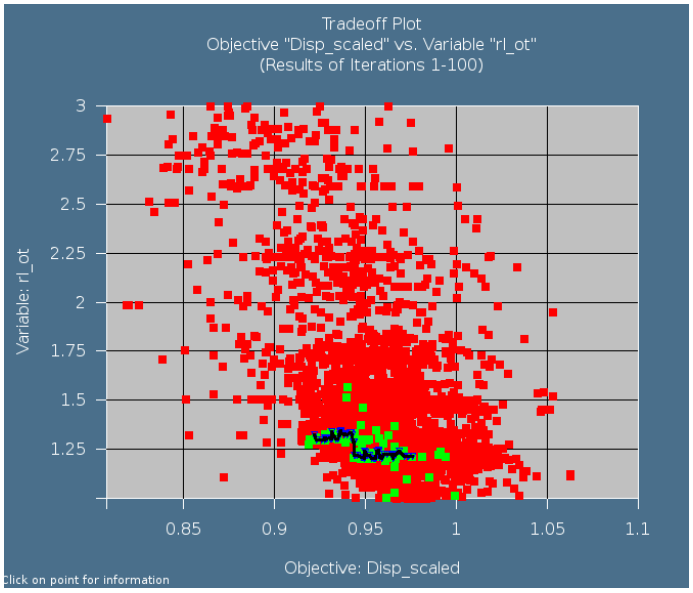
Outer shotgun



Cradle rails

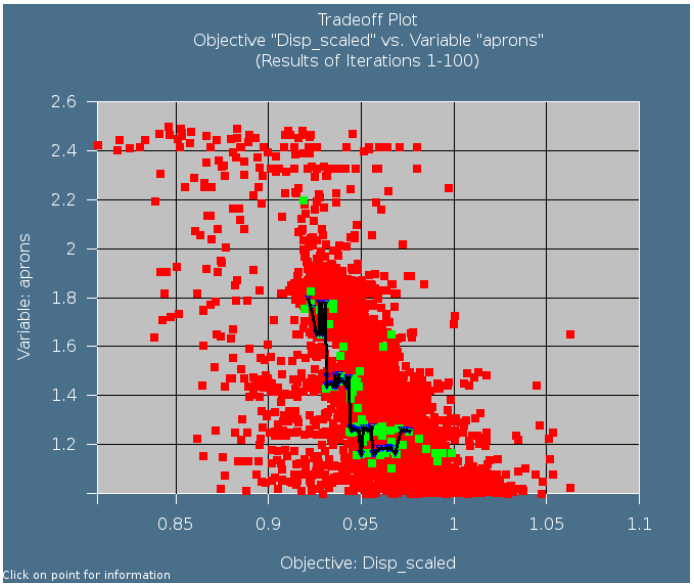


Cradle crossmembers



Inner rails

Outer rails



Aprons

Figure 22-35: Scaled intrusion (displacement) (x-axis) vs. design variables (y-axis) at final generation.

22.7 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.

22.7.1 FE modeling

Figure 22-36 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 22-36 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 22-37. The three gauges and the yoke cross-sectional radius are also considered in a separate sizing (4 variable) optimization.

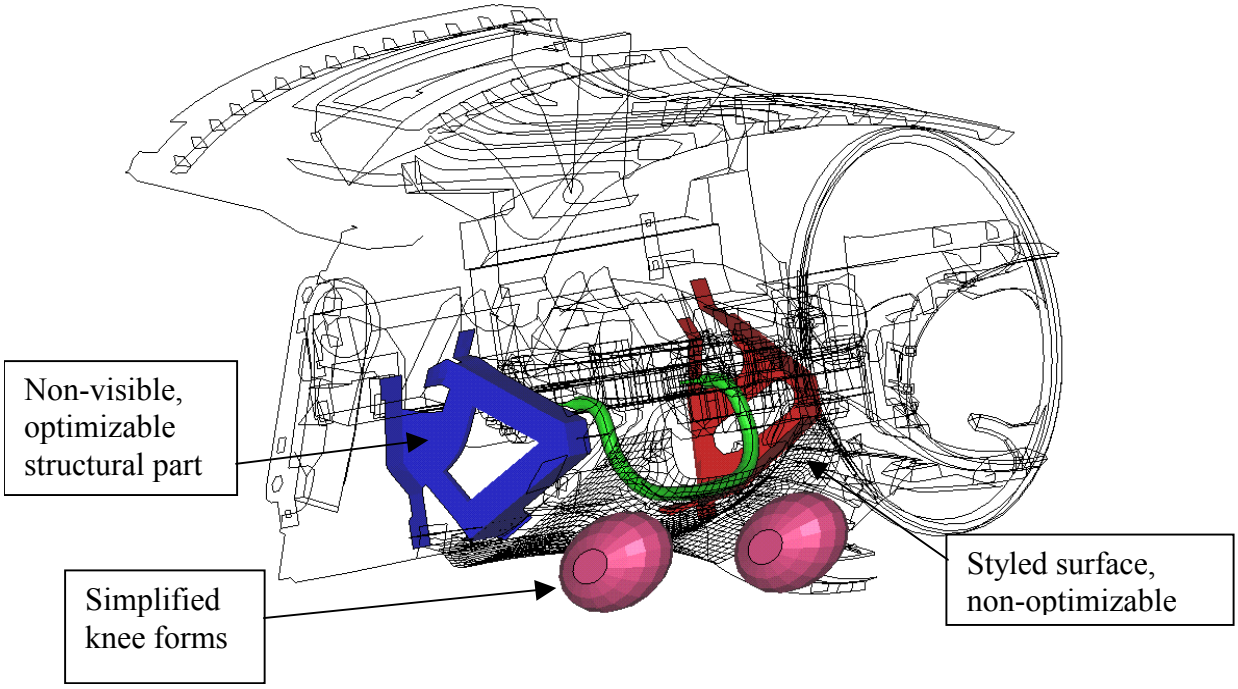


Figure 22-36: Typical instrument panel prepared for a "Bendix" component test

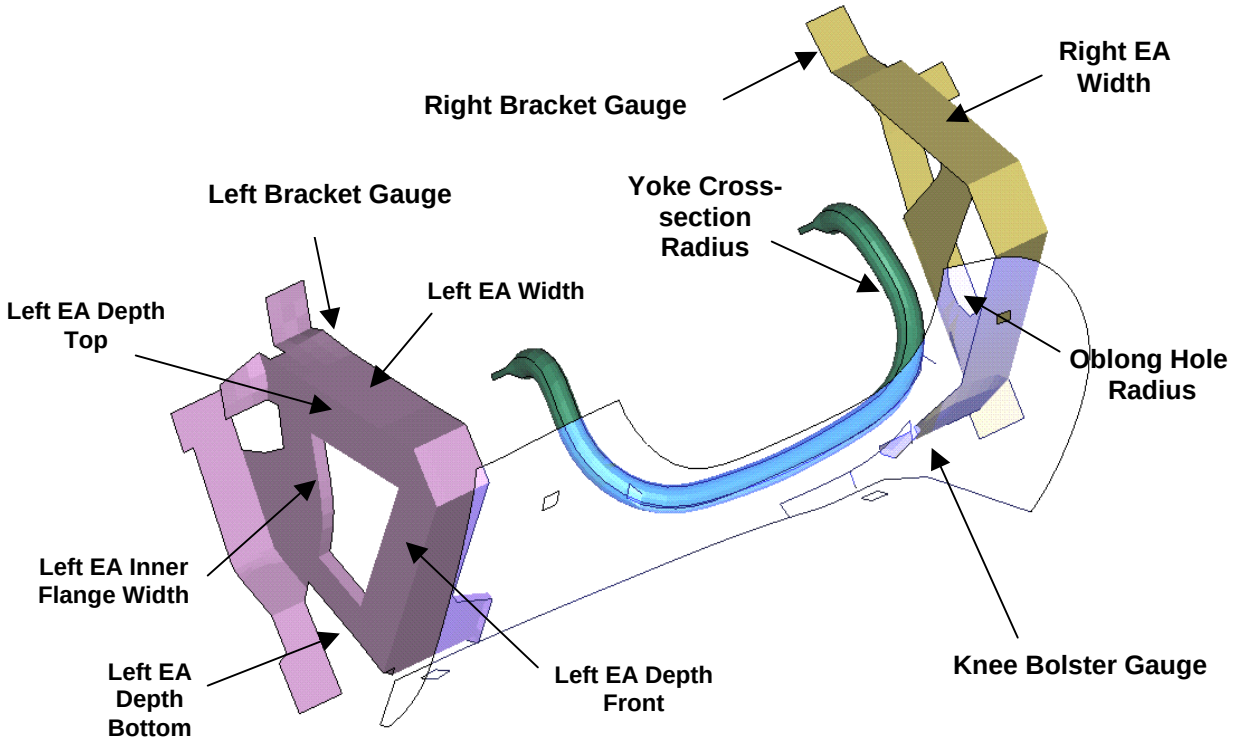


Figure 22-37: Typical major components of a knee bolster system and definition of design variables

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.

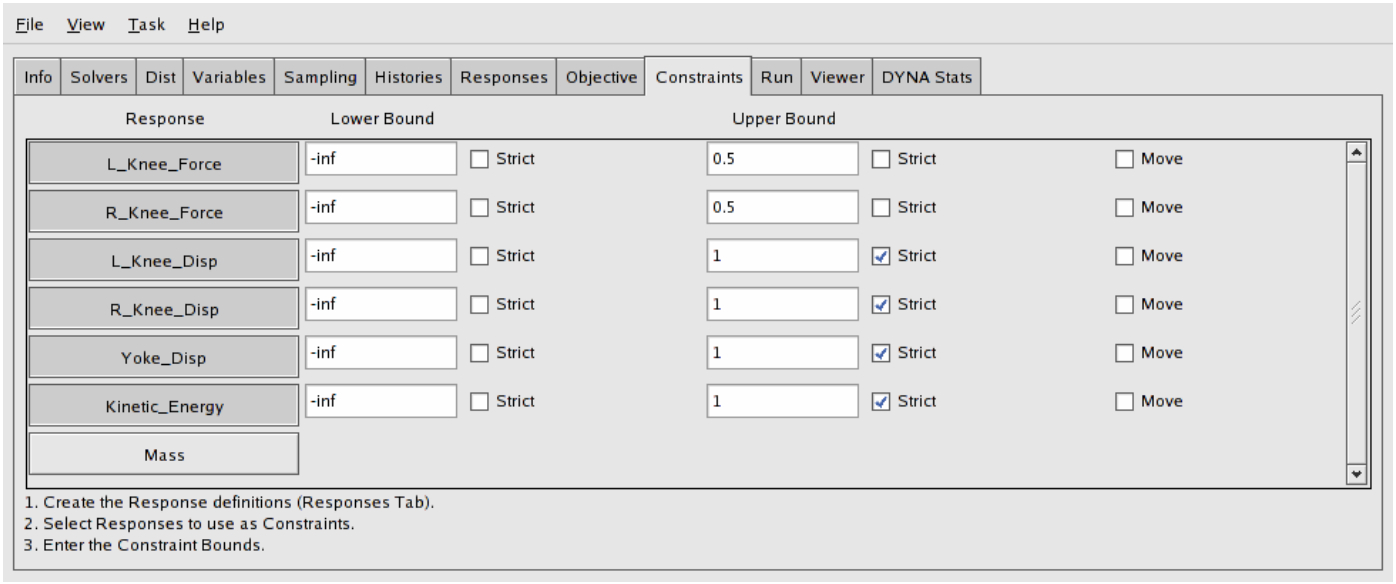
22.7.2 Design formulation

The optimization problem is defined as follows:

Minimize (max (Knee_Force_Left, Knee_Force_Right))
Subject to

Left Knee intrusion < 115mm
Right Knee intrusion < 115mm
Yoke displacement < 85mm

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.



The knee forces have been filtered, SAE 60 Hz, to improve the approximation accuracy.

22.7.3 Input preparation

Truegrid is used to parameterize the geometry. The section of the Truegrid input file (s7.tg) where the design variables are substituted, is shown below.

```
para
  w1 <<L_Flange_Width>>             c Left EA flange width
  w2 <<R_Flange_Width>>             c Right EA flange width
  thick1 <<L_Bracket_Gauge>>        c Left bracket gauge
  thick2 <<R_Bracket_Gauge>>        c Right bracket gauge
  thick3 <<Bolster_gauge>>          c Knee bolster gauge
  f1  <<T_Flange_Depth>>            c Left EA Depth Top
  f2  <<F_Flange_Depth>>            c Left EA Depth Front
  f3  <<B_Flange_Depth>>            c Left EA Depth Bottom
  f4  <<I_Flange_Width>>            c Left EA Inner Flange Width
  r1  <<Yoke_Radius>>               c Yoke bar radius
  r2  <<R_Bracket_Radius>>          c Oblong hole radius
```

The LS-OPT input file is shown below for the 11-variable shape optimization case:

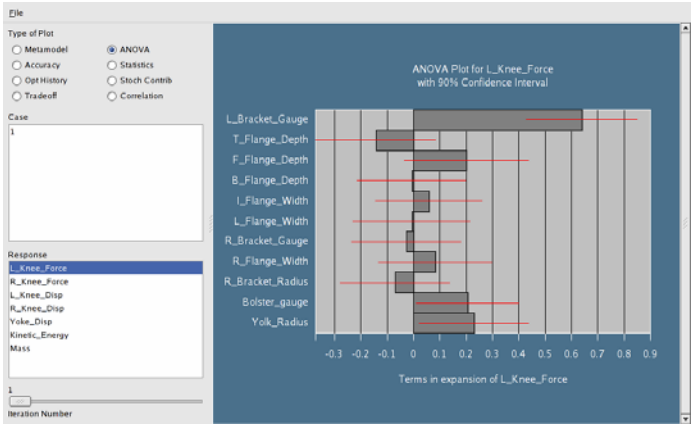
```
"Knee impact with 11 variables"
$ Created on Tue Feb  5 15:43:40 2008
solvers 1
responses 7
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 11
 Variable 'L_Bracket_Gauge' 1.1
  Lower bound variable 'L_Bracket_Gauge' 0.7
  Upper bound variable 'L_Bracket_Gauge' 3
 Variable 'T_Flange_Depth' 28.3
  Lower bound variable 'T_Flange_Depth' 20
  Upper bound variable 'T_Flange_Depth' 50
 Variable 'F_Flange_Depth' 27.5
  Lower bound variable 'F_Flange_Depth' 20
  Upper bound variable 'F_Flange_Depth' 50
 Variable 'B_Flange_Depth' 22.3
  Lower bound variable 'B_Flange_Depth' 15
  Upper bound variable 'B_Flange_Depth' 50
 Variable 'I_Flange_Width' 7
  Lower bound variable 'I_Flange_Width' 5
  Upper bound variable 'I_Flange_Width' 25
 Variable 'L_Flange_Width' 32
  Lower bound variable 'L_Flange_Width' 20
  Upper bound variable 'L_Flange_Width' 50
 Variable 'R_Bracket_Gauge' 1.1
  Lower bound variable 'R_Bracket_Gauge' 0.7
  Upper bound variable 'R_Bracket_Gauge' 3
 Variable 'R_Flange_Width' 32
  Lower bound variable 'R_Flange_Width' 20
  Upper bound variable 'R_Flange_Width' 50
 Variable 'R_Bracket_Radius' 15
  Lower bound variable 'R_Bracket_Radius' 10
  Upper bound variable 'R_Bracket_Radius' 25
 Variable 'Bolster_gauge' 3.5
  Lower bound variable 'Bolster_gauge' 1
  Upper bound variable 'Bolster_gauge' 6
 Variable 'Yoke_Radius' 4
  Lower bound variable 'Yoke_Radius' 2
  Upper bound variable 'Yoke_Radius' 8

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "1"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "1"
$
 solver dyna960 '1'
  solver command "ls971_single"
  solver input file "ford7.k"
  solver check output off
  solver compress d3plot off
$ ------ Pre-processor --------
  prepro truegrid
  prepro command "cp ../../curves .; cp ../../node .; cp ../../elem .; cp ../../elem-bar .; /truegrid/tg"
  prepro input file "s7.tg"
$ ------ Metamodeling ---------
  solver order RBF
  solver experiment design space_filling
   solver update doe
   solver alternate experiment 1
$ ------ Job information ------
  solver concurrent jobs 4
$
$ RESPONSES FOR SOLVER "1"
$
 response 'L_Knee_Force' 0.000153846 0 "BinoutResponse -res_type RCForc -cmp force -invariant MAGNITUDE
        -id 1  -side MASTER -select MAX -start_time 0.0000 -filter SAE  -filter_freq 60.0000"
 response 'R_Knee_Force' 0.000153846 0 "BinoutResponse -res_type RCForc -cmp force -invariant MAGNITUDE
        -id 2  -side MASTER -select MAX -start_time 0.0000 -filter SAE  -filter_freq 60.0000"
 response 'L_Knee_Disp' 0.00869565 0 "BinoutResponse -res_type Nodout  -cmp displacement
        -invariant MAGNITUDE -id 24897 -select MAX -start_time 0.0000"
 response 'R_Knee_Disp' 0.00869565 0 "BinoutResponse -res_type Nodout  -cmp displacement
        -invariant MAGNITUDE -id 25337 -select MAX -start_time 0.0000"
 response 'Yoke_Disp' 0.0117647 0 "BinoutResponse -res_type Nodout  -cmp displacement
        -invariant MAGNITUDE -id 28816 -select MAX -start_time 0.0000"
 response 'Kinetic_Energy' 6.49351e-06 0 "BinoutResponse -res_type GLStat -cmp kinetic_energy -select TIME "
 response 'Mass' 638.162 0 "DynaMass 7 8 48 62 MASS"

$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'Mass' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 6
 constraint 'L_Knee_Force'
  upper bound constraint 'L_Knee_Force' 0.5
 constraint 'R_Knee_Force'
  upper bound constraint 'R_Knee_Force' 0.5
$
$ Strict constraints
$
 constraint 'L_Knee_Disp'
  strict
  upper bound constraint 'L_Knee_Disp' 1
 constraint 'R_Knee_Disp'
  slack
  strict
  upper bound constraint 'R_Knee_Disp' 1
 constraint 'Yoke_Disp'
  slack
  strict
  upper bound constraint 'Yoke_Disp' 1
 constraint 'Kinetic_Energy'
  slack
  strict
  upper bound constraint 'Kinetic_Energy' 1
$
$ JOB INFO
```

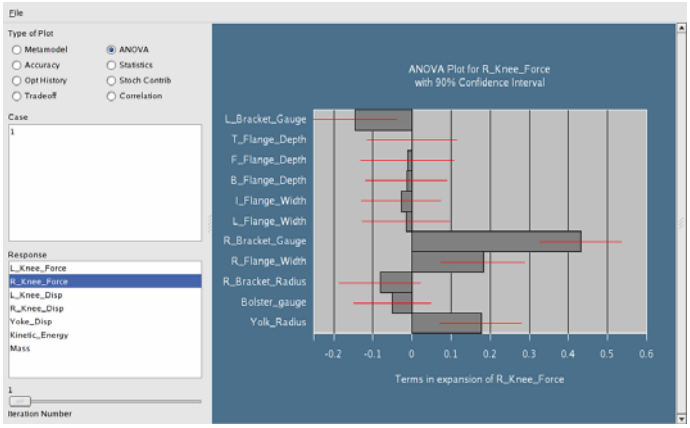
```
$
 concurrent jobs 4
 iterate param design 0.01
 iterate param objective 0.01
$
$ Switch off domain reduction
$
 iterate param adapt off iteration 1
 iterate param stoppingtype and
 iterate 10
STOP
```

22.7.4 Variable screening

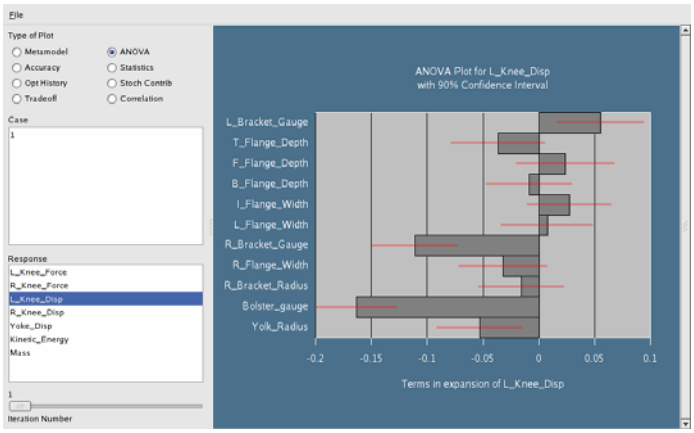
A single iteration is done with a linear approximation to generate the ANOVA charts. The charts are shown in the figure below. Note the large confidence intervals (low confidence levels) on some of the responses, especially the Left Knee Force and Yoke Displacement.



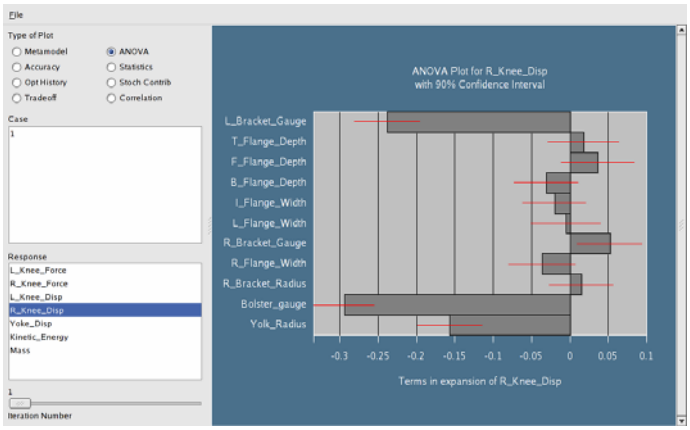
Left Knee Force



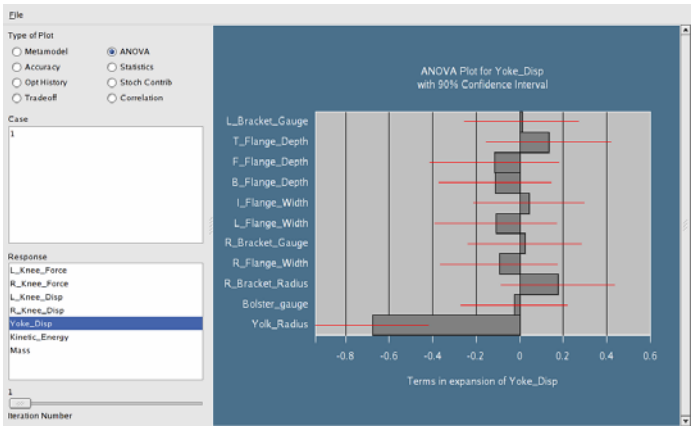
Right Knee Force



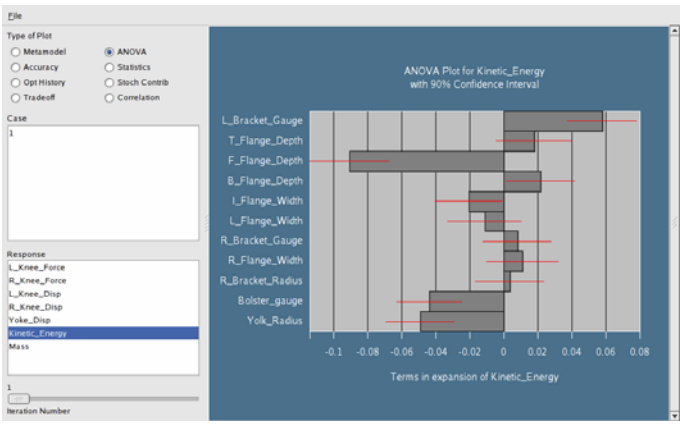
Left Knee Intrusion



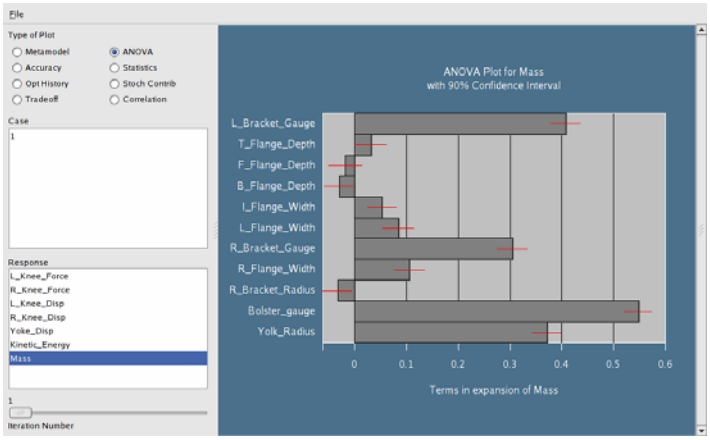
Right Knee Intrusion



Yoke displacement



Kinetic Energy



Mass

The variables chosen from the charts are:

$\boldsymbol{x}=[L\_Bracket\_Gauge, T\_Flange\_Depth, R\_Bracket\_Gauge, R\_Flange\_Width, Bolster\_gauge, Yoke\_Radius]^T$;

The changes in the input file are as follows:

```
variables 6
 Variable 'L_Bracket_Gauge' 1.1
  Lower bound variable 'L_Bracket_Gauge' 0.7
  Upper bound variable 'L_Bracket_Gauge' 3
 Variable 'T_Flange_Depth' 28.3
  Lower bound variable 'T_Flange_Depth' 20
  Upper bound variable 'T_Flange_Depth' 50
 Variable 'R_Bracket_Gauge' 1.1
  Lower bound variable 'R_Bracket_Gauge' 0.7
  Upper bound variable 'R_Bracket_Gauge' 3
 Variable 'R_Flange_Width' 32
  Lower bound variable 'R_Flange_Width' 20
  Upper bound variable 'R_Flange_Width' 50
 Variable 'Bolster_gauge' 3.5
  Lower bound variable 'Bolster_gauge' 1
  Upper bound variable 'Bolster_gauge' 6
 Variable 'Yoke_Radius' 4
```

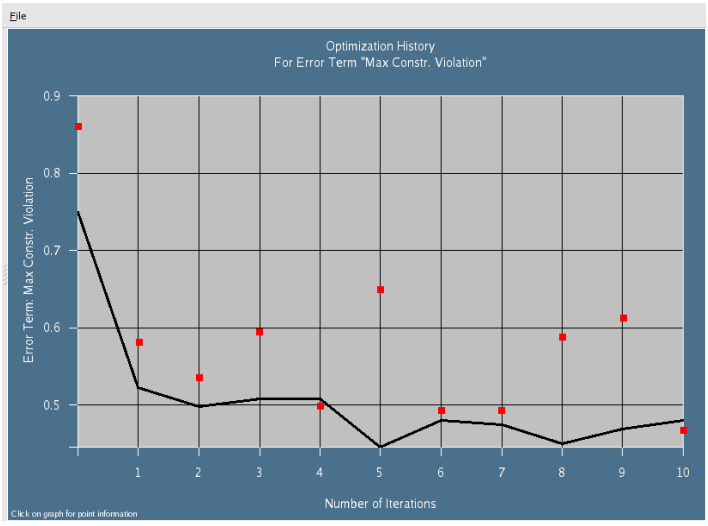
```
  Lower bound variable 'Yoke_Radius' 2
  Upper bound variable 'Yoke_Radius' 8
$
$ CONSTANTS
$
constants 5
 Constant 'F_Flange_Depth' 27.5
 Constant 'B_Flange_Depth' 22.3
 Constant 'I_Flange_Width' 7
 Constant 'L_Flange_Width' 32
 Constant 'R_Bracket_Radius' 15
```

22.7.5 Optimization strategy

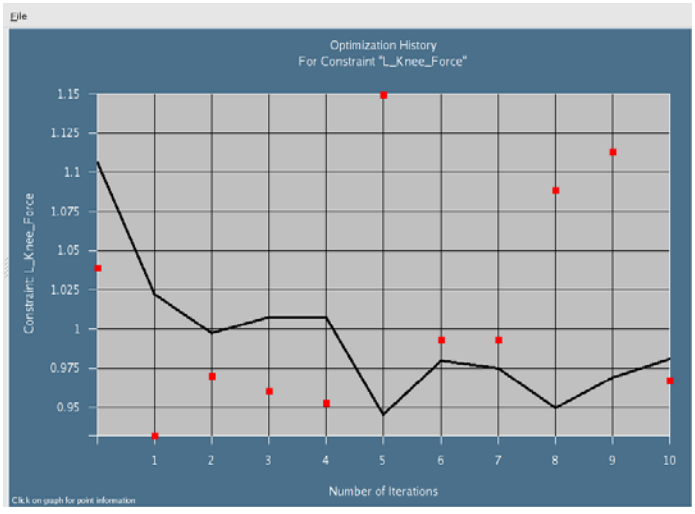
In contrast to the strategy of the full vehicle example, a sequential strategy in which the region of interest is kept constant, is chosen. The reader is also referred to [5] for a discussion of the accuracy and purpose of the various sequential sampling strategies available in LS-OPT. LFOPC (the default algorithm) is chosen as the core solver because of the requirement to minimize the maximum knee force.

22.7.6 Optimization history results

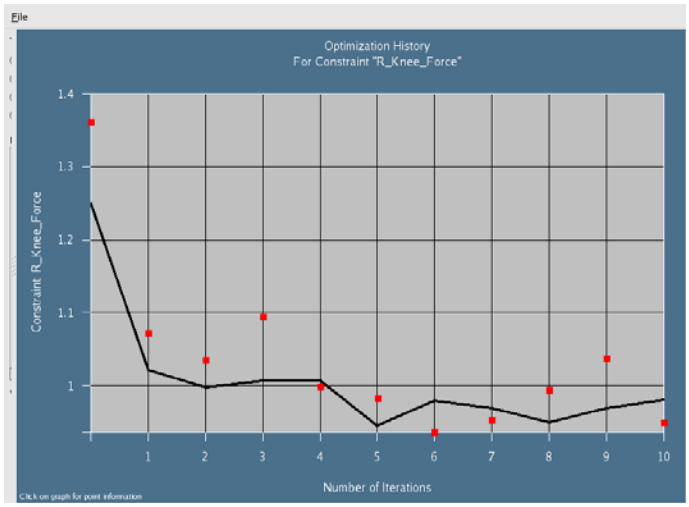
The plots below show optimization history and accuracy as well as a plot of the metamodel in a 2-dimensional subspace. Note the steep gradient of the left knee force with respect to the Left Bracket Gauge. A contour plot shows the constraint isolines superimposed on the Left Knee Force approximation. Note that for this example, there are no feasible regions, since the problem formulation attempts to minimize the maximum constraint violation with respect to both the knee force targets (= 0.5). The darker the shade of red, the more constraints are violated.



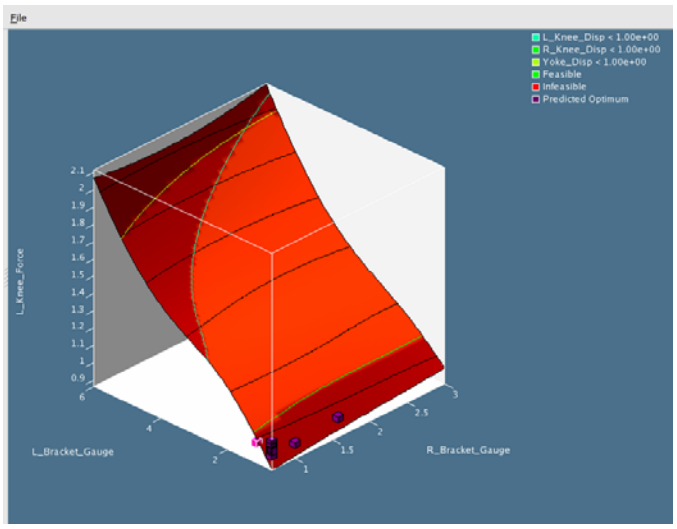
Optimization history of Maximum Constraint Violation



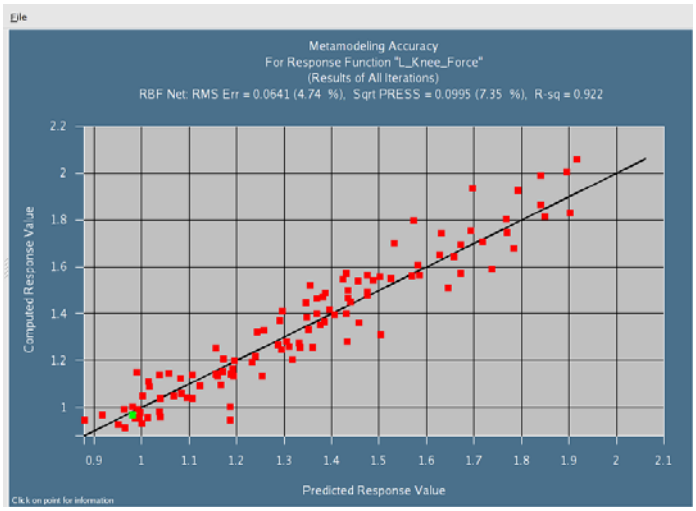
Optimization history of Left Knee Force



Optimization history of Right Knee Force



Constraint isolines (green) superimposed on contoured (in black) Left Knee Force in the space: [Left Bracket Gauge; Right Bracket Gauge]. Lightest shade is most feasible



Left Knee Force: Computed values vs. values predicted from metamodel. Optimum shown in green.

22.7.7 Summary of results

The following is an edited version of the `lsopt_report` file (also viewable by selecting View→Summary).

```
Continuous Variables
--------------------------------------------|
Variable Name      Lower Bound   Upper Bound
---------------|-------------|-------------|
L_Bracket_Gauge          0.7             3
T_Flange_Depth            20            50
R_Bracket_Gauge          0.7             3
```

```
R_Flange_Width            20          50
Bolster_gauge              1           6
Yoke_Radius                2           8
-------------------------------------------------------------------------

-------------------------------------------------------------------------
                  O B J E C T I V E   F U N C T I O N S
-------------------------------------------------------------------------
Objective ..................... MINIMIZE

-----------------------------------------
Objective name                  Weights
-----------------------------------------
Mass                            1

-----------------------------------------

-------------------------------------------------------------------------
                C O N S T R A I N T   F U N C T I O N S
-------------------------------------------------------------------------
Constraint name                 Lower Bound                 Upper Bound
-------------------------------------------------------------------------
L_Knee_Force                    -1e+30      0.5
R_Knee_Force                    -1e+30      0.5
L_Knee_Disp                     -1e+30      1
R_Knee_Disp                     -1e+30      1
Yoke_Disp                       -1e+30      1
Kinetic_Energy                  -1e+30      1
-------------------------------------------------------------------------
-------------------------------------------------------------------------
                   O P T I M I Z A T I O N   A L G O R I T H M
-------------------------------------------------------------------------
 Method ............................ Sequential Response Surface Method
 Optimization Algorithm ............. LFOPC
-------------------------------------------------------------------------

=================================================================

              -------------------------------
              | Evaluating Starting Design |
              |        ITERATION  1        |
              -------------------------------


          C O M P U T E D  vs.  P R E D I C T E D
          ---------------------------------------

Using Metamodel of Iteration 1

-----------------------------------------
DESIGN POINT
------------
Variable Name                   Lower Bound   Value    Upper Bound
--------------------------------|-----------|----------|-----------|--------
L_Bracket_Gauge                         0.7       1.1           3
T_Flange_Depth                           20      28.3          50
R_Bracket_Gauge                         0.7       1.1           3
R_Flange_Width                           20        32          50
Bolster_gauge                             1       3.5           6
Yoke_Radius                               2         4           8
--------------------------------|-----------|----------|-----------|--------

               OBJECTIVE:
               ---------
Computed Value  =     0.8093
Predicted Value =     0.8222


OBJECTIVE FUNCTIONS:
--------------------
OBJECTIVE NAME                  | Computed   Predicted  WT.
```

Mass	0.8093	0.8222	1

CONSTRAINT FUNCTIONS:

CONSTRAINT NAME	Computed	Predicted	Lower	Upper	Viol?
L_Knee_Force	1.039	1.106	-1e+30	0.5	YES
R_Knee_Force	1.361	1.249	-1e+30	0.5	YES
L_Knee_Disp	0.8555	0.8583	-1e+30	1	
R_Knee_Disp	0.8431	0.8631	-1e+30	1	
Yoke_Disp	0.4511	0.7248	-1e+30	1	
Kinetic_Energy	0.3733	0.3761	-1e+30	1	

CONSTRAINT VIOLATIONS:

CONSTRAINT NAME	Computed Violation		Predicted Violation	
	Lower	Upper	Lower	Upper
L_Knee_Force	-	0.5391	-	0.6059
R_Knee_Force	-	0.861	-	0.7494
L_Knee_Disp	-	-	-	-
R_Knee_Disp	-	-	-	-
Yoke_Disp	-	-	-	-
Kinetic_Energy	-	-	-	-

MAXIMUM VIOLATION:

Quantity	Computed		Predicted	
	Constraint	Value	Constraint	Value
Maximum Violation	R_Knee_Force	0.861	R_Knee_Force	0.7494
Smallest Margin	L_Knee_Disp	0.1445	R_Knee_Disp	0.1369

==========

E R R O R M E A S U R E S F O R R E S P O N S E S

ITERATION 10

==========

Response Name	Metamodel type	RMS Error	RMS Error (% of mean)	Maximum Residual	Sq. Root PRESS	Sq. Root PRESS (% of mean)	R-Sq.
L_Knee_Force	RBF Net	0.0641	4.74	0.156	0.0995	7.35	0.922
R_Knee_Force	RBF Net	0.0264	2	0.0677	0.0571	4.32	0.967
L_Knee_Disp	RBF Net	0.0163	1.97	0.0474	0.0233	2.81	0.914
R_Knee_Disp	RBF Net	0.012	1.56	0.0416	0.0179	2.33	0.986
Yoke_Disp	RBF Net	0.162	24.3	0.701	0.261	39.2	0.634
Kinetic_Energy	RBF Net	0.00938	2.36	0.0362	0.0433	10.9	0.927
Mass	RBF Net	0.0032	0.291	0.00798	0.00477	0.434	0.999

==========

F I N A L D E S I G N

ITERATION 11

==========

DESIGN POINT

Variable Name	Lower Bound	Value	Upper Bound
L_Bracket_Gauge	0.7	1.144	3

```
T_Flange_Depth                          20      39.54          50
R_Bracket_Gauge                        0.7        0.7           3 Active
R_Flange_Width                          20      39.38          50
Bolster_gauge                            1      1.078           6
Yoke_Radius                              2      4.427           8
--------------------------------|----------|----------|----------|--------

              OBJECTIVE:
              ---------
Computed Value  =     0.5272
Predicted Value =     0.5291


OBJECTIVE FUNCTIONS:
--------------------
OBJECTIVE NAME                  | Computed   Predicted  WT.
--------------------------------|----------|----------|----
Mass                            |    0.5272     0.5291|   1
--------------------------------|----------|----------|----

CONSTRAINT FUNCTIONS:
---------------------
CONSTRAINT NAME                 | Computed | Predicted|  Lower   |  Upper   |Viol?
--------------------------------|----------|----------|----------|----------|-----
L_Knee_Force                    |    0.9672     0.9805|    -1e+30        0.5|YES
R_Knee_Force                    |    0.9505     0.9805|    -1e+30        0.5|YES
L_Knee_Disp                     |     1.002     0.9451|    -1e+30          1|
R_Knee_Disp                     |     1.003          1|    -1e+30          1|YES
Yoke_Disp                       |    0.5018     0.5661|    -1e+30          1|
Kinetic_Energy                  |    0.3676     0.3765|    -1e+30          1|
--------------------------------|----------|----------|----------|----------|-----

CONSTRAINT VIOLATIONS:
----------------------
                                |  Computed Violation | Predicted Violation |
CONSTRAINT NAME                 |----------|----------|----------|----------|
                                |  Lower   |  Upper   |  Lower   |  Upper   |
--------------------------------|----------|----------|----------|----------|
L_Knee_Force                    |    -          0.4672|    -          0.4805|
R_Knee_Force                    |    -          0.4505|    -          0.4805|
L_Knee_Disp                     |    -        0.001601|    -          -     |
R_Knee_Disp                     |    -        0.002817|    -       2.899e-06|
Yoke_Disp                       |    -          -     |    -          -     |
Kinetic_Energy                  |    -          -     |    -          -     |
--------------------------------|----------|----------|----------|----------|

MAXIMUM VIOLATION:
------------------
                   |         Computed          |        Predicted          |
     Quantity      |---------------------------|---------------------------|
                   |   Constraint      Value   |   Constraint      Value   |
-------------------|----------------|----------|----------------|----------|
Maximum Violation  |L_Knee_Force         0.4672|R_Knee_Force         0.4805|
Smallest Margin    |L_Knee_Disp        0.001601|R_Knee_Disp       2.899e-06|
-------------------|----------------|----------|----------------|----------|


                        ANALYSIS COMPLETED

                                    Wed Feb  6 02:59:06 2008
```

22.8 Optimization with analytical design sensitivities

This example demonstrates how analytical gradients (Section 13.8) provided by a solver can be used for optimization using the SLP algorithm and the domain reduction scheme [5] (Section 4.6). The solver, a Perl program is shown below, followed by the command file for optimization. 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";
```

Command file:

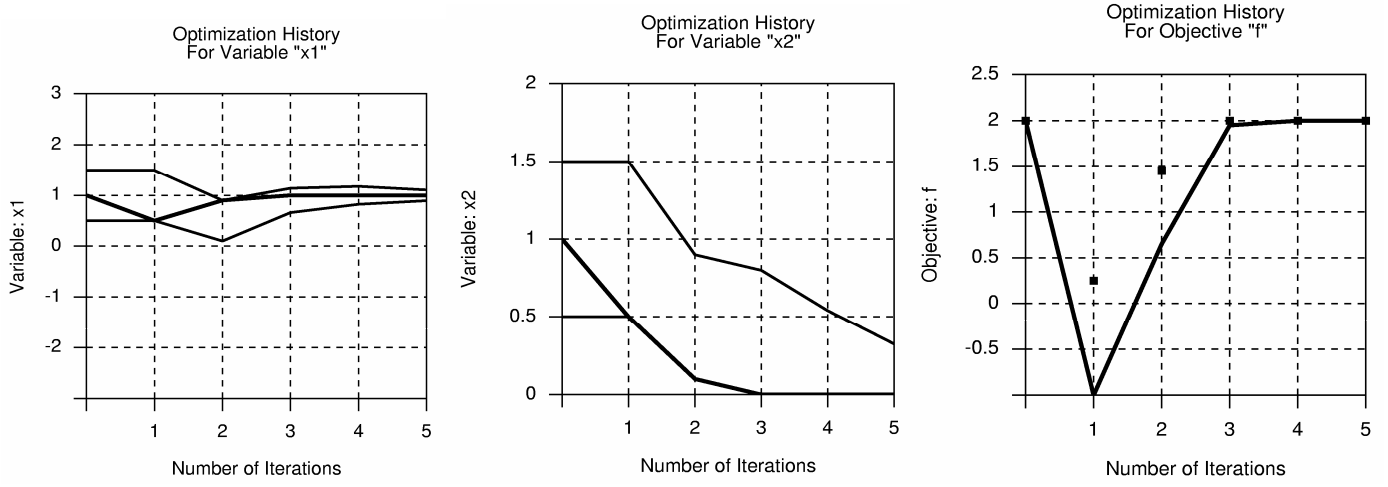
```
"Example 2b: QP problem (analytical sensitivity analysis)"
```

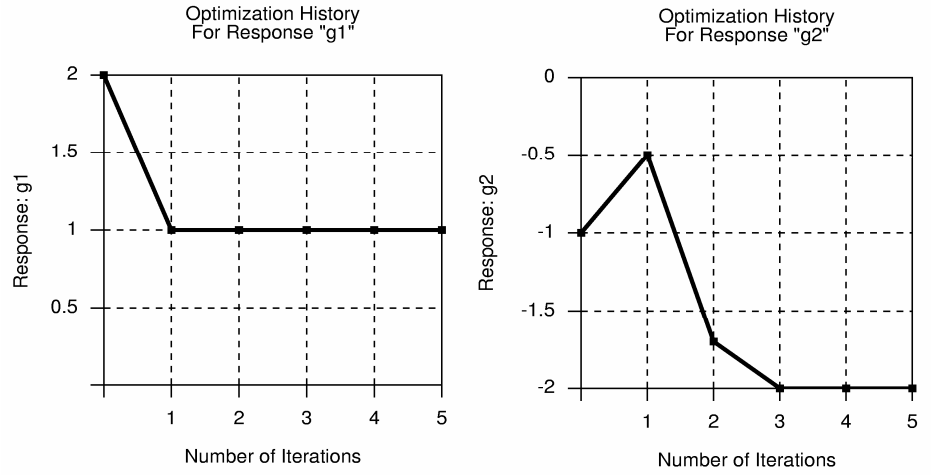
```
solvers 1
responses 3
$
$ DESIGN VARIABLES
$
variables 2
 Variable 'x1' 1
  Lower bound variable 'x1' -3
  Upper bound variable 'x1' 3
  Range 'x1' 1
 Variable 'x2' 1
  Lower bound variable 'x2' 0
  Upper bound variable 'x2' 2
  Range 'x2' 1
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "1"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
solver own '1'
  solver command "/home/LSOPT_EXE/perl ../../ex2"
  solver experimental design analytical_DSA
$
$ RESPONSES FOR SOLVER "1"
$ The Gradf, Gradg1 and Gradg2 files are individually copied to "Gradient"
 response 'f' 1 0 "cp Gradf Gradient; cat fsol"
 response 'g1' 1 0 "cp Gradg1 Gradient; cat g1sol"
 response 'g2' 1 0 "cp Gradg2 Gradient; cat g2sol"
$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 maximize
 objective 'f' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 2
 constraint 'g1'
  upper bound constraint 'g1' 1
 constraint 'g2'
  upper bound constraint 'g2' 2
$
$ JOB INFO
$
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype and
 iterate 5
STOP
```

Typical `"Gradient"` file (e.g. for f):

```
1.8000000000 -3.20000000000
```

The optimization results are shown in the plots below. An iteration represents a single simulation. The dots represent the computed results while the solid line represents a linear approximation constructed from the gradient information of the previous point.





22.9 Probabilistic Analysis

22.9.1 Overview

This example has the following features:

- Probabilistic analysis
- Monte Carlo analysis
- Monte Carlo analysis using a metamodel
- Bifurcations analysis

22.9.2 Problem description

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).

Figure 22-38: Tube impact

Both the shell thickness and the yield strength of the structure follow a probabilistic distribution. 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 a standard deviation of 10%.

The nominal design has an intrusion of 144.4 units. 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 build using a 3^k experimental design. The results from the different methods are close to each other as can be seen in the following table.

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.

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).

Source of variation	**Intrusion Standard Deviation**
Shell Thickness	10.4
Yield Strength	10.7

The details of the analyses are given the following subsections.

22.9.3 Monte Carlo evaluation

The probabilistic variation is described by specifying statistical distributions and assigning the statistical distributions to noise variables.

```
"Tube Crush Monte Carlo "
$ Created on Tue Apr  1 11:26:07 2003
solvers 1
$
distribution 2
 distribution 't' NORMAL 1.0 0.05
 distribution 'y' NORMAL 1.0 0.10
```

```
$
$ DESIGN VARIABLES
$
variables 2
 noise variable 'T1' distribution 't'
 noise variable 'YS' distribution 'y'
$
$ DEFINITION OF SOLVER "SOLVER_1"
$
 solver dyna960 'SOLVER_1'
  solver command "ls970.single"
  solver input file "tube.k"
  solver experiment design lhd centralpoint
  solver number experiments 60
$
$ HISTORIES FOR SOLVER "SOLVER_1"
$
histories 1
 history 'NHist' "BinoutHistory -res_type nodout -cmp z_displacement -id 486"
$
$ RESPONSES FOR SOLVER "SOLVER_1"
$
responses 2
 response 'NodDisp' 1 0 "BinoutResponse -res_type nodout -cmp z_displacement -id 486
-select MIN "
 response 'DispT' {LookupMin("NHist(t)")}
$
$
$
 constraints 1
  constraint 'NodDisp'
  lower bound constraint 'NodDisp' -150
$
$ JOB INFO
$
 analyze monte carlo
STOP
```

The LS-OPT output:

```
############################################################
Direct Monte Carlo simulation considering 2 stochastic variables.
############################################################


#######################################################
STATISTICS OF VARIABLES
#######################################################

Variable 'T1'
Distribution Information
------------------------
Number of points   :     60
Mean Value         :      1
Standard Deviation : 0.04948
```

```
Coef of Variation  : 0.04948
Maximum Value      :   1.12
Minimum Value      : 0.8803


Variable 'YS'
Distribution Information
------------------------
Number of points   :     60
Mean Value         :      1
Standard Deviation : 0.09895
Coef of Variation  : 0.09895
Maximum Value      :  1.239
Minimum Value      : 0.7606


#########################################################
STATISTICS OF RESPONSES
#########################################################

Response 'NodDisp'
Distribution Information
------------------------
Number of points   :     60
Mean Value         : -141.8
Standard Deviation :  15.21
Coef of Variation  : 0.1073
Maximum Value      : -102.3
Minimum Value      : -168.9



Response 'DispT'
Distribution Information
------------------------
Number of points   :     60
Mean Value         :  7.726
Standard Deviation : 0.6055
Coef of Variation  : 0.07837
Maximum Value      :    8.4
Minimum Value      :    5.5



#########################################################
STATISTICS OF COMPOSITES
#########################################################


#########################################################
STATISTICS OF CONSTRAINTS
#########################################################


Constraint 'NodDisp'
Distribution Information
------------------------
```

```
Number of points   :     60
Mean Value         : -141.8
Standard Deviation :  15.21
Coef of Variation  : 0.1073
Maximum Value      : -102.3
Minimum Value      : -168.9

Lower Bound:
------------
      Bound ..................................   -150
      Evaluations exceeding this bound .........     20
      Probability of exceeding bound ........... 0.3333
      Confidence Interval on Probability.
            Standard Deviation of Prediction Error: 0.06086
            Lower Bound | Probability | Higher Bound
                 0.2116 |      0.3333 |      0.455
            Confidence Interval of 95% assuming Normal Distribution
            Confidence Interval of 75% using Tchebysheff's Theorem



Reliability Assuming Normal Distribution
========================================
Lower Bound:
------------
      Bound ............................   -150
      Probability of exceeding Bound ... 0.2956
      Reliability Index (Beta) ......... 0.5372



                          ANALYSIS COMPLETED
```

22.9.4 Monte Carlo using metamodel

The bounds on the design variables are set to be two standard distributions away from the mean (the default for noise variables). Noise variables are not used because of the need to have more control over the variable bounds — specifically we want to 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).

The command file for using a metamodel is:

```
$
"Tube Crush Metamodel Monte Carlo"
$ Created on Tue Apr  1 11:26:07 2003
solvers 1
$
distribution 2
 distribution 't' NORMAL 1.0 0.05
 distribution 'y' NORMAL 1.0 0.10
$
$ DESIGN VARIABLES
$
variables 2
 variable 'T1' 1.0
```

```
  upper bound variable 'T1' 1.1
  lower bound variable 'T1' 0.9
  variable 'T1' distribution 't'
 variable 'YS' 1.0
  upper bound variable 'YS'  1.2
  lower bound variable 'YS'  0.8
  variable 'YS' distribution 'y'
$
$ DEFINITION OF SOLVER "SOLVER_1"
$
 solver dyna960 'SOLVER_1'
  solver command "ls970.single"
  solver input file "tube.k"
  solver experiment design 3toK
  solver order quadratic
$
$ HISTORIES FOR SOLVER "SOLVER_1"
$
histories 1
 history 'NHist' "BinoutHistory -res_type nodout -cmp z_displacement -id 486"
$
$ RESPONSES FOR SOLVER "SOLVER_1"
$
responses 2
 response 'NodDisp' 1 0 "BinoutResponse -res_type nodout -cmp z_displacement -id 486 -
select MIN"
 response 'DispT' {LookupMin("NHist(t)")}
$
$
$
 constraints 1
  constraint 'NodDisp'
  lower bound constraint 'NodDisp' -150.0
$
$ JOB INFO
$
 analyze metamodel monte carlo
STOP
```

The accuracy of the response surface is of interest:

```
Approximating Response 'NodDisp' (ITERATION 1)
------------------------------------------------------------------
Polynomial approximation: using 9 points

          Global error parameters of response surface
          -------------------------------------------
Quadratic Function Approximation:
---------------------------------
Mean response value           =  -142.0087

RMS error                     =     2.0840 (1.47%)
Maximum Residual              =     3.3633 (2.37%)
Average Error                 =     1.6430 (1.16%)
```

```
Square Root PRESS Residual    =     6.2856 (4.43%)
Variance                      =    13.0296
R^2                           =     0.9928
R^2 (adjusted)                =     0.9856
R^2 (prediction)              =     0.9346
```

The probabilistic evaluation results:

```
################################################################
Monte Carlo simulation considering 2 stochastic variables.
Computed using 1000000 simulations
################################################################



---------------------------------------------------------------
Results for reliability analysis using approximate functions
---------------------------------------------------------------


#####################################################
STATISTICS OF VARIABLES
#####################################################


Variable 'T1'
Distribution Information
------------------------
Number of points   : 1000000
Mean Value         :       1
Standard Deviation : 0.04997
Coef of Variation  : 0.04997
Maximum Value      :  1.227
Minimum Value      : 0.7505



Variable 'YS'
Distribution Information
------------------------
Number of points   : 1000000
Mean Value         :       1
Standard Deviation : 0.09994
Coef of Variation  : 0.09994
Maximum Value      :  1.472
Minimum Value      : 0.5187



#####################################################
STATISTICS OF RESPONSES
#####################################################
Response 'NodDisp'
Distribution Information
------------------------
Number of points   : 1000000
```

```
Mean Value         : -141.4
Standard Deviation :  14.95
Coef of Variation  : 0.1058
Maximum Value      :  -68.5
Minimum Value      : -206.3
Response 'DispT'
Distribution Information
------------------------
Number of points   : 1000000
Mean Value         :   7.68
Standard Deviation :  0.546
Coef of Variation  : 0.0711
Maximum Value      :  9.267
Minimum Value      :  2.565


#######################################################
STATISTICS OF COMPOSITES
#######################################################


#######################################################
STATISTICS OF CONSTRAINTS
#######################################################


Constraint 'NodDisp'
Distribution Information
------------------------
Number of points   : 1000000
Mean Value         : -141.4
Standard Deviation :  14.95
Coef of Variation  : 0.1058
Maximum Value      :  -68.5
Minimum Value      : -206.3

Lower Bound:
------------
      Bound ..................................   -150
      Evaluations exceeding this bound ......... 285347
      Probability of exceeding bound ........... 0.2853
      Confidence Interval on Probability.
            Standard Deviation of Prediction Error: 0.0004516
            Lower Bound | Probability | Higher Bound
                 0.2844 |      0.2853 |     0.2863
            Confidence Interval of 95% assuming Normal Distribution
            Confidence Interval of 75% using Tchebysheff's Theorem


                          ANALYSIS COMPLETED
```

22.9.5 Bifurcation analysis

A bifurcation analysis of the tube is conducted as described in Section 6.6, Section 21, and Example 22.10. The resulting buckling modes found for the metamodel-based analysis are as shown in Figure 22-39. An extra half wave is formed for the one design.

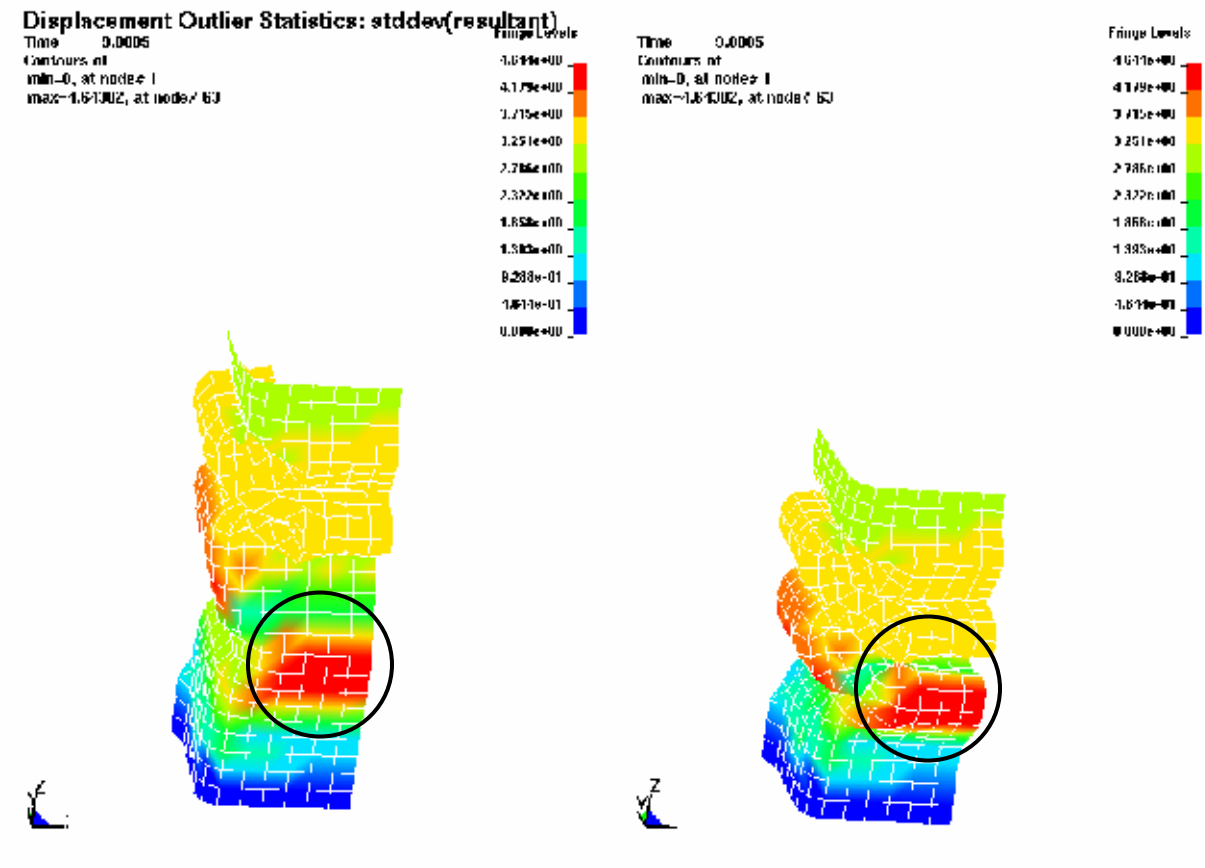


Figure 22-39 Tube Buckling

22.10 Bifurcation/Outlier Analysis

22.10.1 Overview

This example has the following features:

- Monte Carlo analysis
- Identification of different buckling modes in the structure

22.10.2 Problem description

The plate as shown in Figure 22-40 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.

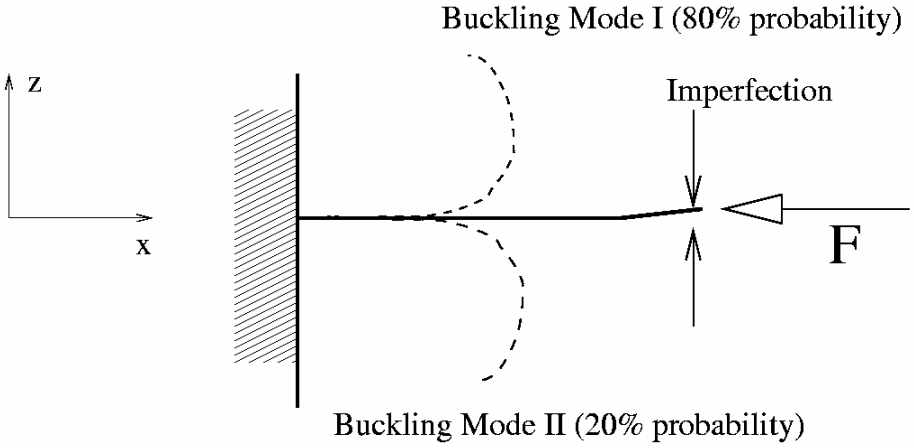


Figure 22-40 Plate Buckling Example

22.10.3 Monte Carlo evaluation

A Latin hypercube experimental design is used for the Monte Carlo analysis. 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 next section will demonstrate how to find out which run contains the different buckling mode.

```
"Monte Carlo Analysis; 2 buckling modes"
$
solvers 1
$
distribution 1
 distribution 'i' UNIFORM -0.001 0.004
$
$ DESIGN VARIABLES
$
variables 1
 noise variable 'Imp' distribution 'i'
$
$
$ SOLVER_1
```

```
$
solver dyna960 'SOLVER_1'
  solver command "ls970.single"
  solver input file "plate.k"
  solver experiment design lhd centralpoint
  solver number experiments 5
$
$ RESPONSES
$
responses 4
 response 'tip_x' 1 0 "BinoutResponse -res_type nodout -cmp x_displacement -id 12  -
select TIME "
 response 'tip_y' 1 0 "BinoutResponse -res_type nodout -cmp y_displacement -id 12  -
select TIME "
 response 'tip_z' 1 0 "BinoutResponse -res_type nodout -cmp z_displacement -id 12  -
select TIME "
 response 'tip_r' 1 0 "BinoutResponse -res_type nodout -cmp displacement  -invariant
MAGNITUDE -id 12  -select TIME "
$
$
$ JOB
$
 analyze monte carlo
STOP
```

22.10.4 Automatic identification of buckling modes

Different buckling modes can be identified automatically and displayed in LS-PREPOST. To identify bifurcations, we display the FE jobs having the extreme values. 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 GUI as as shown in Figure 22-41 with the bifurcation as displayed using LS-PREPOST as shown in Figure 22-42. Some background on bifurcation indentification can be found in Section 21.9. A more user-intensive procedure is described in the next section.

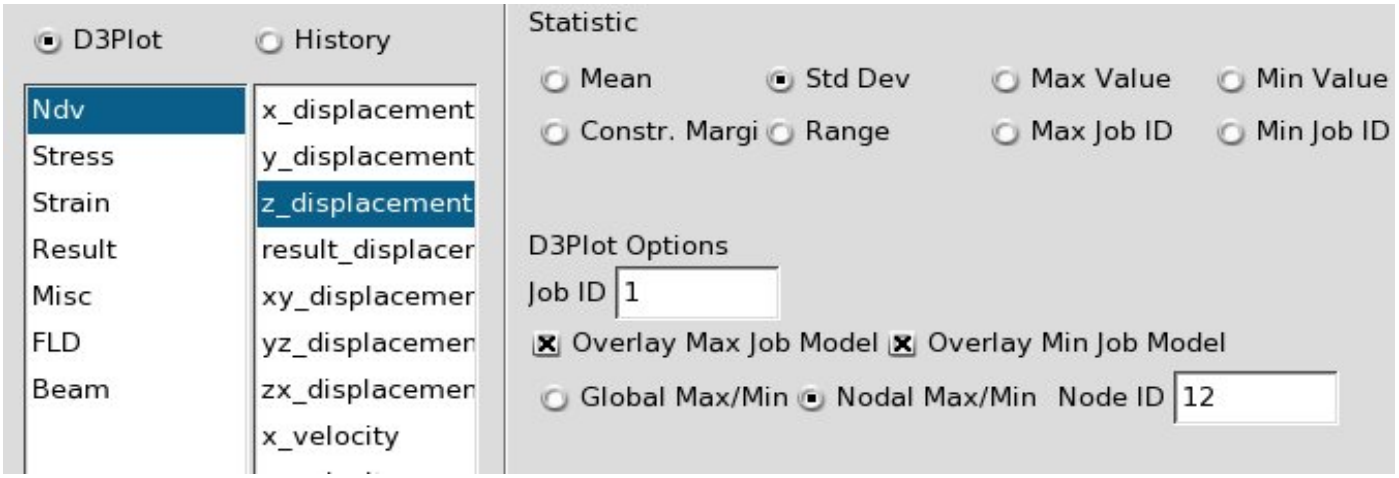


Figure 22-41 Selecting the automated identification of a bifurcation. The user must (i) select to overlay the FE models associated with the maximum and minimum residual and (ii) chose whether the residual is the global residual or a residual at a specific node.

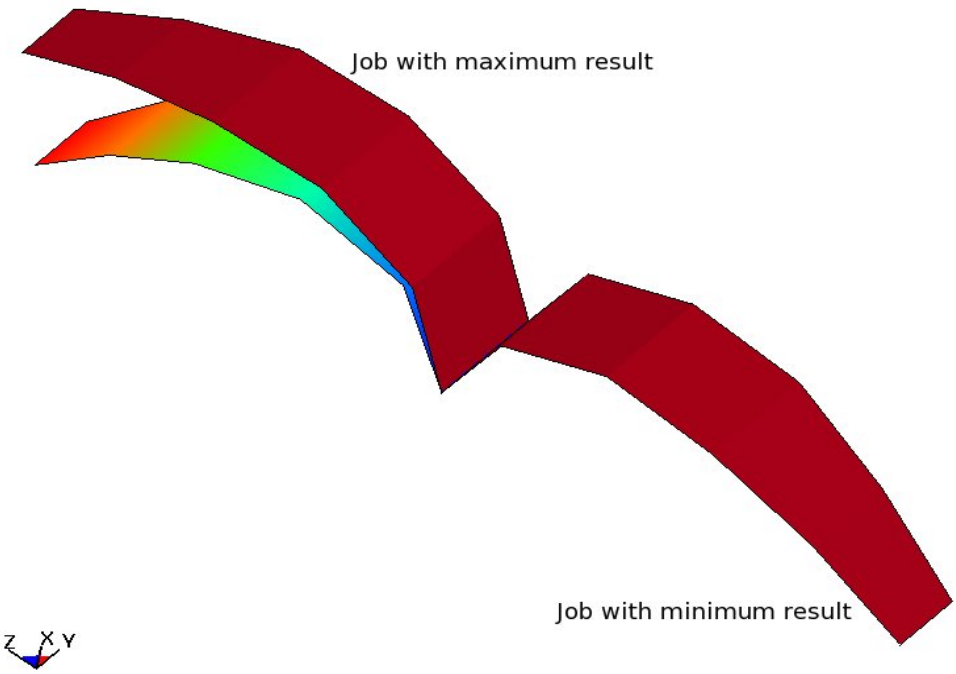


Figure 22-42 LS-OPT identified and displayed this bifurcation automatically using the GUI setting shown in the previous figure.

22.10.5 Manual identification of buckling modes

The different buckling modes are identified using the DYNA Stats panel in LS-OPT.

Next, LS-PREPOST can be launched to investigate the range (or standard deviation) of all the displacement components. From the displacement resultant plot, amongst others, it is clear that the bifurcation is at the tip. Looking at the other component plots, we find the z-displacement has a range of 5.3 and the x-displacement a range of 4.5. The displacement magnitude computed using the maximum vector has a range of 6.9.

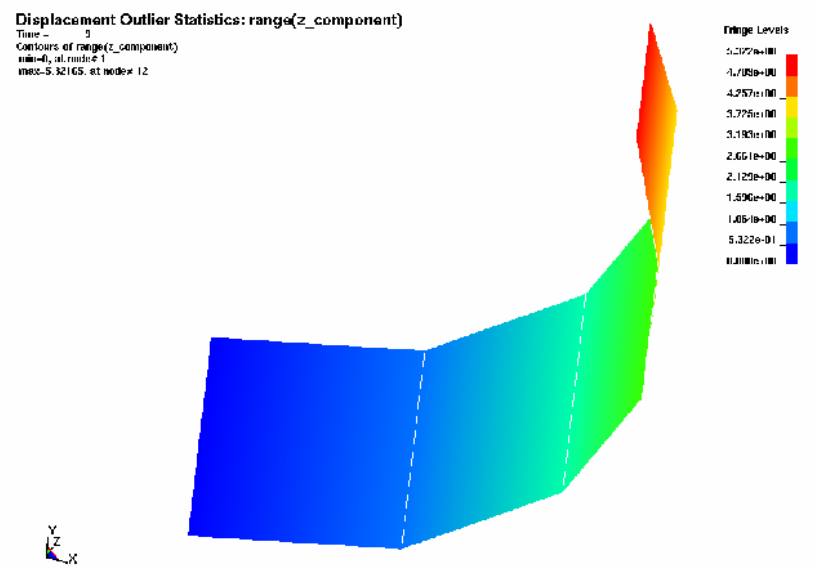


Figure 22-43 Range of z-component displacement

Either the z-displacement or the maximum vector displacement magnitude can therefore be used to identify the buckling modes. Fringe plots of the run index of the maximum and minimum displacement identifies the runs as 2 and 4.

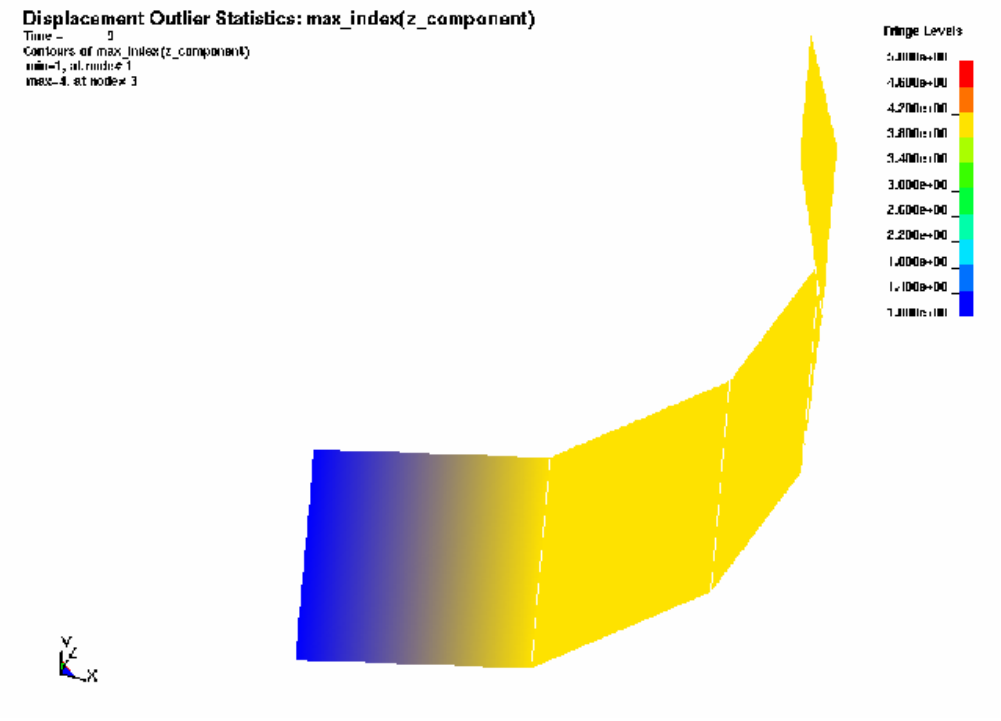


Figure 22-44 Index of run with maximum z-component displacement

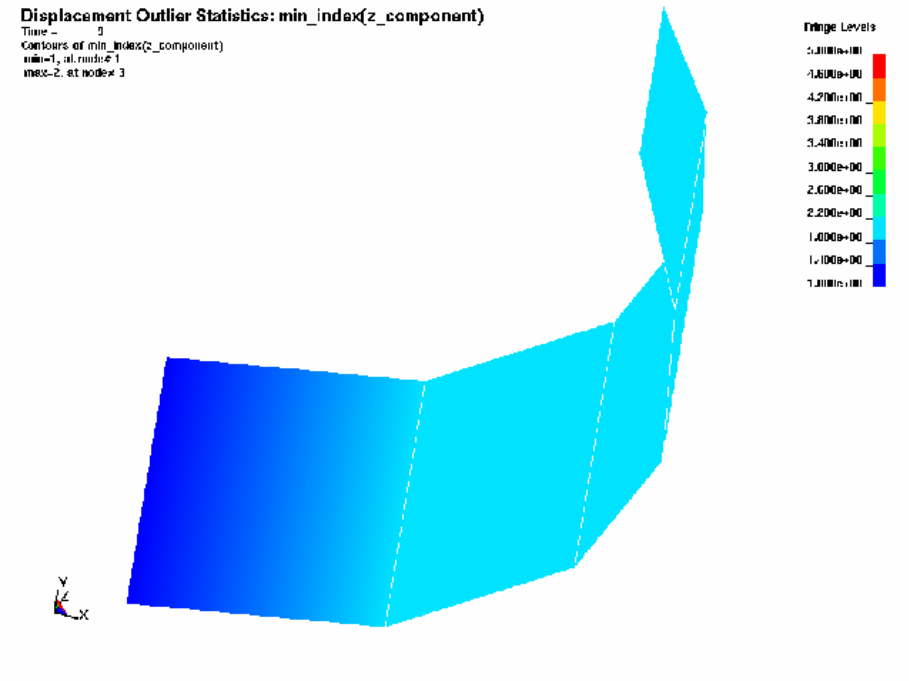


Figure 22-45 Index of run with minimum z-component displacement

LS-OPT allows you to specify the job number to use for the LS-PREPOST plot. Plotting the results of run 2 and 4 we find the second buckling mode as:

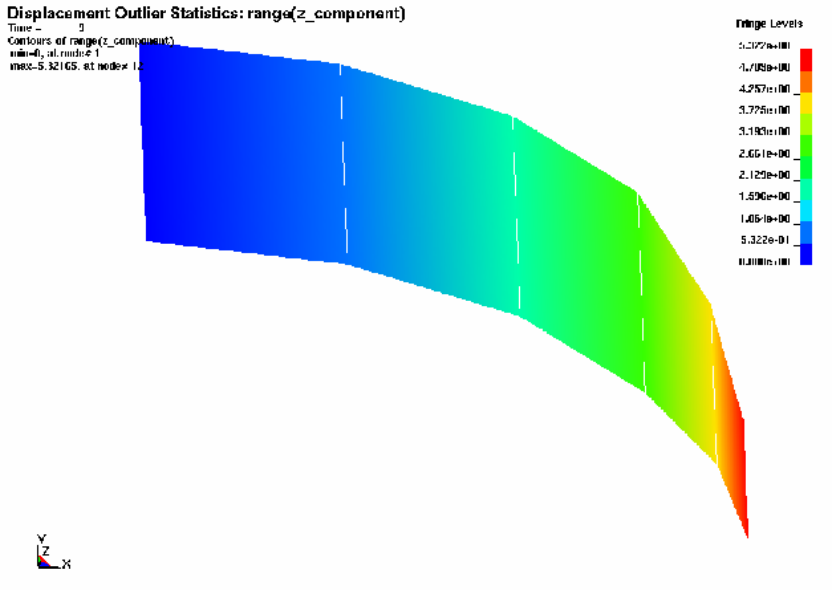


Figure 22-46 Second buckling mode

22.11 Robust Parameter Design

Consider the two-bar truss problem as shown in Figure 22-47. Variable x1, 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 distance between the legs, x2, is a control variable which will be adjusted to control the variance of the responses. The maximum stress is considered as the objective for the robust design process.

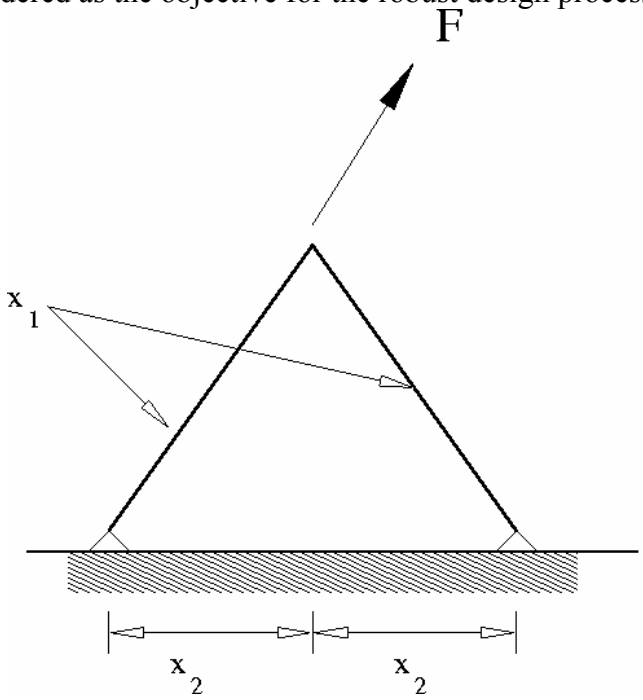


Figure 22-47 The two-bar truss problem. The problem has two variables: the thickness of the bars and the leg widths as shown. The bar thicknesses are noise variables while the leg widths are adjusted (control variables) to minimize the effect of the variation of the bar thicknesses. The maximum stress in the structure is monitored.

An response surface considering the effect of variables and the interaction between variables is used to approximate the stress response.

```
"Two-bar Truss"
$
solvers 1
responses 2
$
$ PROBABILISTIC DISTRIBUTIONS
$
distribution 1
 distribution 'area' NORMAL  2.0  0.1
$
$ DESIGN VARIABLES
$
variables 2
 Noise variable 'Area' distribution 'area'
 Variable 'Base' 0.8
  Lower bound variable 'Base' 0.1
```

```
  Upper bound variable 'Base' 1.6
  Range 'Base' 1.6

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "SOLVER_1"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "SOLVER_1"
$
 solver own 'SOLVER_1'
  solver command "echo N o r m a l"
$ ------ Pre-processor --------
$   NO PREPROCESSOR SPECIFIED
$ ------ Metamodeling ---------
  solver order interaction
  solver experiment design 3toK
$ ------ Job information ------
  solver concurrent jobs 1
$
$ RESPONSE EXPRESSIONS FOR SOLVER "SOLVER_1"
$
 response 'Weight' expression { Area * sqrt(1+Base*Base) }
 response 'Stress' expression { 0.124 * sqrt(1+Base*Base) * (8/Area + 1./Area/Base) }
$
composites 1
$
$ STD DEV COMPOSITES
$
 composite 'StressStandardDeviation' noise 'Stress'
$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'StressStandardDeviation' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 0
$
$ JOB INFO
$
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype and
 iterate 10
STOP
```

The stress response is shown in Figure 22-48. From the figure it can be seen that the ‘base’ variable must be set to values of large than 0.4 to obtain a minimum variation of the stress considering that the design will then be in the flattest region of the response. A value of 0.5 is obtained in the optimization results as shown in Figure 22-49. Also shown in the optimization results is the design history of the stress standard deviation. Note that the standard deviation response stayed fairly insensitive to changes in the control variable after iteration 4 and that the initial subregion size for the ‘base’ variable was too large, resulting in initial increase in ‘base’ variable due to an inaccurate initial response surface.

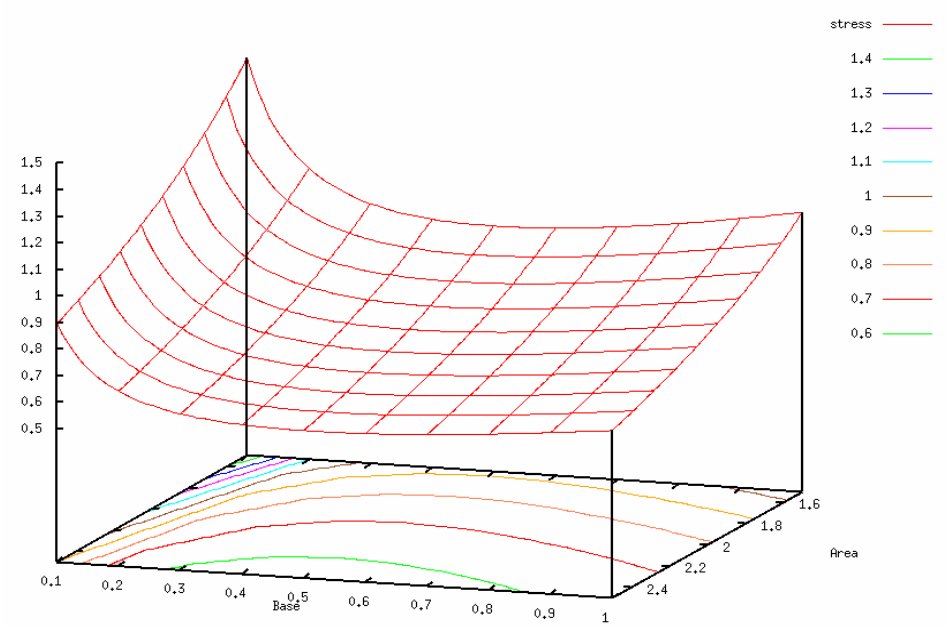


Figure 22-48 Contours of stress response. The flattest part of the response is when variable 'base' equals 0.5.

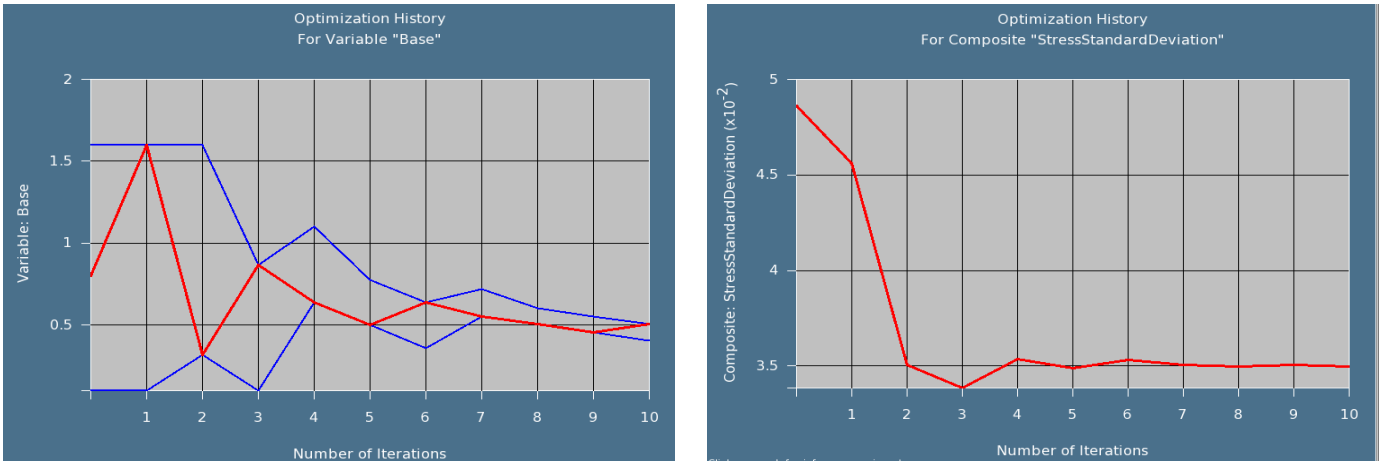


Figure 22-49 Optimization histories. Design variable ‘base’ is shown on the left and the standard deviation of the stress response is shown on the right.

22.12 REFERENCES

[1] Yamazaki, K., Han, J., Ishikawa, H., Kuroiwa, Y. Maximation of crushing energy absorption of cylindrical shells – simulation and experiment, Proceedings of the OPTI-97 Conference, Rome, Italy, September 1997.

[2] Craig K.J., Stander, N., Dooge, D., Varadappa, S. MDO of automotive vehicle for crashworthiness and NVH using response surface methods. Paper AIAA2002_5607, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 4-6 Sept 2002, Atlanta, GA.

[3] National Crash Analysis Center (NCAC). Public Finite Element Model Archive, www.ncac.gwu.edu/archives/model/index.html 2001.

[4] Akkerman, A., Thyagarajan, R., Stander, N., Burger, M., Kuhn, R., Rajic, H. Shape optimization for crashworthiness design using response surfaces. Proceedings of the 1st International Workshop on Multidisciplinary Design Optimization, Pretoria, South Africa, 8-10 August 2000, pp. 270-279.

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

[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.

Appendix A

LS-DYNA D3Plot Result 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

Result Type	Number	Description	Component name
	35	Surface stress $N_{xx}/t - 6M_{xx}/t^2$	Nxx/t-6Mxx/t^2
	36	Surface stress $N_{yy}/t - 6M_{yy}/t^2$	Nyy/t-6Myy/t^2
	37	Surface stress $N_{yy}/t + 6M_{yy}/t^2$	Nyy/t+6Myy/t^2
	38	Surface stress $N_{xy}/t - 6M_{xy}/t^2$	Nxy/t+6Mxy/t^2
	39	Surface stress $N_{xy}/t + 6M_{xy}/t^2$	Nxy/t+6Mxy/t^2
	40	Effective upper surface stress	u_surf_eff_stress
	41	Effective lower surface stress	l_surf_eff_stress
Strain	43	Lower surface effective plastic strain	l_surf_plastic_strain
	44	Upper surface effective plastic strain	u_surf_plastic_strain
	45	Lower surface *xx*, *yy*, *zz*, *xy*, *yz*, *zx* strain	l_surf_xx_strain
	46		l_surf_yy_strain
	47		l_surf_zz_strain
	48		l_surf_xy_strain
	49		l_surf_yz_strain
	50		l_surf_zx_strain
	51	Upper surface *xx*, *yy*, *zz*, *xy*, *yz*, *zx* strain	u_surf_xx_strain
	52		u_surf_yy_strain
	53		u_surf_zz_strain
	45		u_surf_xy_strain
	55		u_surf_yz_strain
	56		u_surf_zx_strain
	57	Middle surface *xx*, *yy*, *zz*, *xy*, *yz*, *zx* strain	m_surf_xx_strain
	58		m_surf_yy_strain
	59		m_surf_zz_strain
	60		m_surf_xy_strain
	61		m_surf_yz_strain
	62		m_surf_zx_strain
	69	Lower, upper, middle principal + effective strains	l_surf_max_princ_strain
	70		l_surf_2nd_princ_strain
	71		l_surf_min_princ_strain
	72		l_surf_effective_princ_strain
	73		u_surf_max_princ_strain
	74		u_surf_2nd_princ_strain
	75		u_surf_min_princ_strain
	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

Result Type	Number	Description	Component name
	512		maxima_ eps1
	513	Lower, upper, middle, maxima surface eps2	lower_ eps1
	514		upper_ eps1
	515		middle_ eps1
	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 B

LS-DYNA Binout Result Components

Airbag Statistics: ABSTAT

Component	Description
Volume	Volume
pressure	Pressure
internal_energy	Internal energy
dm_dt_in	Input mass flow rate
dm_dt_out	Output mass flow rate
total_mass	Mass
gas_temp	Temperature
density	Density
surface_area	Area
reaction	Reaction

Boundary Nodal Forces: BNDOUT

Component	Description
Subdirectory ***discrete/nodes***	
x_force	X-force
y_force	Y-force
z_force	Z-force
x_total	Total X-force
y_total	Total Y-force
z_total	Total Z-force
energy	Energy
etotal	Total Energy

Discrete Element Forces: DEFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
resultant_force	Resultant force
displacement	Change in length

Element Output: ELOUT

Component	Description
Subdirectory ***solid***	
sig_xx	XX-stress
sig_xy	YY-stress
sig_yy	ZZ-stress
sig_yz	XY-stress
sig_zx	YZ-stress
sig_zz	ZX-stress
yield	Yield function
effsg	Effective stress
eps_xx	XX-strain
eps_xy	YY-strain
eps_yy	ZZ-strain
eps_yz	XY-strain
eps_zx	YZ-strain
eps_zz	ZX-strain
Subdirectory ***beam***	
axial	Axial force resultant
shear_s	s-Shear resultant
shear_t	t-Shear resultant
moment_s	s-Moment resultant
moment_t	t-Moment resultant
torsion	Torsional resultant

Element Output: ELOUT

Component	Description
Subdirectory ***shell***	
sig_xx	XX-stress
sig_yy	YY-stress
sig_zz	ZZ-stress
sig_xy	XY-stress
sig_yz	YZ-stress
sig_zx	ZX-stress
plastic_strain	Plastic strain
upper_eps_xx	XX-strain
lower_eps_xx	
upper_eps_yy	YY-strain
lower_eps_yy	
upper_eps_zz	ZZ-strain
lower_eps_zz	
upper_eps_xy	XY-strain
lower_eps_xy	
upper_eps_yz	YZ-strain
lower_eps_yz	
upper_eps_zx	ZX-strain
lower_eps_zx	
Subdirectory ***thickshell***	
sig_xx	XX-stress
sig_yy	YY-stress
sig_zz	ZZ-stress
sig_xy	XY-stress
sig_yz	YZ-stress
sig_zx	ZX-stress
yield	Yield
upper_eps_xx	XX-strain
lower_eps_xx	
upper_eps_yy	YY-strain
lower_eps_yy	
upper_eps_zz	ZZ-strain
lower_eps_zz	
upper_eps_xy	XY-strain
lower_eps_xy	
upper_eps_yz	YZ-strain
lower_eps_yz	
upper_eps_zx	ZX-strain
lower_eps_zx	

Contact Entities Resultants: GCEOUT

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
force_magnitude	Force magnitude
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment
moment_magnitude	Moment magnitude

Global Statistics: GLSTAT

Component	Description
kinetic_energy	Kinetic energy
internal_energy	Internal energy
total_energy	Total energy
energy_ratio	Ratio
stonewall_energy	Stonewall energy
spring_and_damper_energy	Spring & Damper energy
hourglass_energy	Hourglass energy
sliding_interface_energy	Sliding interface energy
external_work	External work
global_x_velocity	Global x-velocity
global_y_velocity	Global y-velocity
global_z_velocity	Global z-velocity
system_damping_energy	System damping energy
energy_ratio_wo_eroded	Energy ratio w/o eroded
eroded_internal_energy	Eroded internal energy
eroded_kinetic_energy	Eroded kinetic energy

Joint Element Forces: JNTFORC

Component	Description
Subdirectory ***joints***	
x_force	X-force
y_force	Y-force
z_force	Z-force
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment
resultant_force	R-force
resultant_moment	R-moment
Subdirectory ***type0***	
d(phi)_dt	d(phi)/dt
d(psi)_dt	d(psi)/dt (degrees)
d(theta)_dt	d(theta)/dt (degrees)
joint_energy	joint energy
phi_degrees	phi (degrees)
phi_moment_damping	phi moment-damping
phi_moment_stiffness	phi moment-stiffness
phi_moment_total	phi moment-total
psi_degrees	psi (degrees)
psi_moment_damping	psi-moment-damping
psi_moment_stiffness	psi-moment-stiffness
psi_moment_total	psi-moment-total
theta_degrees	theta (degrees)
theta_moment_damping	theta-moment-damping
theta_moment_stiffness	theta-moment-stiffness
theta_moment_total	theta-moment-total

Material Summary: MATSUM

Component	Description
kinetic_energy	Kinetic energy
internal_energy	Internal energy
x_momentum	X-momentum
y_momentum	Y-momentum
z_momentum	Z-momentum
x_rbvelocity	X-rigid body velocity
y_rbvelocity	Y-rigid body velocity
z_rbvelocity	Z-rigid body velocity
hourglass_energy	Hourglass energy

Contact Node Forces: NCFORC

Component	Description
Subdirectory ***master_00001*** *and* ***slave_00001***	
x_force	X-force
y_force	Y-force
z_force	Z-force
pressure	Pressure
x	X coordinate
y	Y coordinate
z	Z coordinate

Nodal Point Response: NODOUT

Component	Description
Translational components	
x_displacement	X-displacement
y_displacement	Y-displacement
z_displacement	Z-displacement
x_velocity	X-velocity
y_velocity	Y-velocity
z_velocity	Z-velocity
x_acceleration	X-acceleration
y_acceleration	Y-acceleration
z_acceleration	Z-acceleration
x_coordinate	X-coordinate
y_coordinate	Y-coordinate
z_coordinate	Z-coordinate
Rotational components	
rx_acceleration	XX-rotation
rx_displacement	YY-rotation
rx_velocity	ZZ-rotation
ry_acceleration	XX-rotational velocity
ry_displacement	YY-rotational velocity
ry_velocity	ZZ-rotational velocity
rz_acceleration	XX-rotational acceleration
rz_displacement	YY-rotational acceleration
rz_velocity	ZZ-rotational acceleration
Injury coefficients	
CSI	Chest Severity Index
HIC15	Head Injury Coefficient (15 ms)
HIC36	Head Injury Coefficient (36 ms)

Nodal Forces: NODFOR

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
x_total	X-total force
y_total	Y-total force
z_total	Z-total force
energy	Energy
etotal	Total Energy

Rigid Body Data: RBDOUT

Component	Description
Translational components	
global_dx	X-displacement
global_dy	Y-displacement
global_dz	Z-displacement
global_vx	X-velocity
global_vy	Y-velocity
global_vz	Z-velocity
global_ax	X-acceleration
global_ay	Y-acceleration
global_az	Z-acceleration
global_x	X-coordinate
global_y	Y-coordinate
global_z	Z-coordinate
local_dx	Local X-displacement
local_dy	Local Y-displacement
local_dz	Local Z-displacement
local_vx	Local X-velocity
local_vy	Local Y-velocity
local_vz	Local Z-velocity
local_ax	Local X-acceleration
local_ay	Local Y-acceleration
local_az	Local Z-acceleration
Rotational components	
global_rax	X-rotation
global_ray	Y-rotation
global_raz	Z-rotation
global_rdx	X-velocity
global_rdy	Y-velocity
global_rdz	Z-velocity
global_rvx	X-acceleration
global_rvy	Y-acceleration
global_rvz	Z-acceleration
local_rdx	Local X-rotation
local_rdy	Local Y-rotation
local_rdz	Local Z-rotation
local_rvx	Local X-velocity
local_rvy	Local Y-velocity
local_rvz	Local Z-velocity
local_rax	Local X-acceleration
local_ray	Local Y-acceleration
local_raz	Local Z-acceleration
Direction cosines	
dircos_11	11 direction cosine
dircos_12	12 direction cosine
dircos_13	13 direction cosine
dircos_21	21 direction cosine
dircos_22	22 direction cosine
dircos_23	23 direction cosine
dircos_31	31 direction cosine
dircos_32	32 direction cosine
dircos_33	33 direction cosine
Injury coefficients	
CSI	Chest Severity Index
HIC15	Head Injury Coefficient (15 ms)
HIC36	Head Injury Coefficient (36 ms)

Reaction Forces: RCFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
mass	Mass

RigidWall Forces: RWFORC

Component	Description
Subdirectory ***forces***	
normal_force	normal
x_force	X-force
y_force	Y-force
z_force	Z-force

Section Forces: SECFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment
x_centroid	X-center
y_centroid	Y-center
z_centroid	Z-center
total_force	Resultant force
total_moment	Resultant moment
area	Area

Single Point Constraint Reaction Forces: SPCFORC

Component	Description
x_force	X-force
y_force	Y-force
z_force	Z-force
x_resultant	Total X-force
y_resultant	Total Y-force
z_resultant	Total Z-force
x_moment	X-moment
y_moment	Y-moment
z_moment	Z-moment

Spotweld and Rivet Forces: SWFORC

Component	Description
axial	Axial force
shear	Shear force
failure_flag	Failure flag

Appendix C

Database files

C.1 Design flow

Source Database file	Process	Output Database file	Level of directory for output database
Command file (**com**)	Point selection	**Experiments**	Solver
Experiments	Simulation runs	Solver output files	Run
Solver output files	Result extraction	**AnalysisResults** **ExtendedResults**	Solver
AnalysisResults	Approximation	**DesignFunctions** **Net**	Solver
DesignFunctions	Optimize	**OptimumResults**	Work
		OptimizationHistory	Work

C.2 Database file formats

The *Experiments* file

This file appears in the solver directory and is used to save the experimental point coordinates for the analysis runs. The file consists of lines having the following format repeated for each experimental point.

```
x[1], x[2], ..., x[n]
```

where $x[1]$ to $x[n]$ are the values of the n solver design variables at the experimental point.

The *AnalysisResults* file

This file is used to save the responses at the experimental points and appears in the solver directory. Every line describes an experimental point and gives the response values at the experimental point. The file consists of lines having the following format repeated for each experimental point.

```
x[1], x[2], ..., x[n],RespVal[1], RespVal[2], ..., RespVal[m]
```

where $x[1]$ to $x[n]$ are the values of the n solver design variables at the experimental point. *RespVal[1]* to *RespVal[m]* are the values of the m solver responses. Values of $2.0*10^{30}$ are assigned to responses of simulations with error terminations. The `AnalysisResults` file is synchronous with the `Experiments` file.

The *DesignFunctions* file

The DesignFunctions file, which appears in the solver 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 is for polynomial design functions is:

beta_0, beta_1, ... , beta_n, beta_1_1, beta_1_2, beta_1_3, ..., beta_1_n,
beta_2_2, beta_2_3, ..., beta_2_n,
...., beta_i_n,
beta_n_n

with

f(x) = beta_0 + beta_1*x_1 + ... + beta_n*x_n +
beta_1_1*x_1*x_1 + beta_1_2*x_1*x_2 + ... + beta_1_n*x_1*x_n
+ beta_2_2*x_2*x_2 + ... + beta_2_n*x_2*x_n
...
+ beta_2_n*x_n*x_n

The following enumerations are used in the database.

Function Types	
NO_SURFACE	0
LINEAR	77
MULT	78
QUADRATIC	79
INTERACTION	80
ELLIPTIC	81
SPHERICAL	82
FEEDFORWARD	83
FF_COMMITTEE	84
RADIALBASIS	85
NEURALNETWORK	86
ANALYTICAL_DSA_SURFACE	87
NUMERICAL_DSA_SURFACE	88
KRIGING	89

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

The flags for active coefficients exclude the constant a_0.

The *OptimizationHistory* file

This file is used to save the optimization history results and appears in the *work* directory. Each line contains the values at the optimum point of an iteration.

Entities	Count
Objective values	Number of objectives
Variables	Number of variables
Variable lower bounds	Number of variables
Variable upper bounds	Number of variables
RMS errors	Number of responses
Average errors	Number of responses
Maximum errors	Number of responses
R^2 errors	Number of responses
Adjusted R^2 errors	Number of responses
PRESS errors	Number of responses
Prediction R^2	Number of responses
Maximum prediction error	Number of responses
Responses	Number of responses
Multi-objective	1
Constraint values	Number of constraints
Composite values	Number of composites
Responses (computed)	Number of responses
Max. constraint violation	1
Composites (computed)	Number of composites
Constraints (computed)	Number of constraints
Objectives (computed)	Number of objectives
Multi-objective (computed)	1
Max. constraint violation (computed)	1
Constants	Number of constants
Dependents	Number of dependents
RBDO lower bound probability*	Number of constraints
RBDO upper bound probability*	Number of constraints
Generation number[#]	1
Individual number[#]	1

*Only written for RBDO problems.

[#]Only written for Direct GA simulations.

Values of $2.0*10^{30}$ are assigned to responses of error terminations.

The *ExtendedResults* file

This file contains all points represented in the `AnalysisResults` file and appears in the solver directory. All values are based on the simulation results. A line has the following format:

Entities	Count
Objective weights	Number of objectives
Objective values	Number of objectives
Variables	Number of solver variables
Responses	Number of solver responses
Multi-objective	1
Constraint values	Number of constraints
Composite values	Number of composites
Max. constraint violation	1
Constants	Number of constants
Dependents	Number of dependents

The values represent the number of entities in the solver. Values of $2.0*10^{30}$ are assigned to responses of simulations with error terminations.

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

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 D

Mathematical Expressions

Mathematical expressions are available for the following entities:

```
Dependent
result
matrix
history
response
composite
multiobjective
```

D.1 Syntax rules

1. Mathematical expressions are placed in curly brackets in the command file or in double angular brackets (e.g. <<Thickness*25.4>>) in the input template files.
2. Expressions consist of parameters and constants. A parameter can be any previously defined entity.
3. Expressions can be wrapped to appear on multiple lines.
4. Mathematical expressions can be used for any floating-point number, e.g. upper bound of constraint, convergence tolerance, objective weight, etc.
5. An expression is limited to 1024 characters.
6. Empty or underscore (_) arguments in functions will generate default values.

D.2 Intrinsic functions

Note: Trigonometric functions use and return degrees, not radians.

`int(a)`	integer
`nint(a)`	nearest integer
`abs(a)`	absolute value
`mod(a,b)`	remainder of a/b
`sign(a,b)`	transfer of sign from b to $\|a\|$
`max(a,b)`	maximum of a and b
`min(a,b)`	minimum of a and b
`sqrt(a)`	square root
`exp(a)`	e^a
`pow(a,b)`	a^b
`log(a)`	natural logarithm
`log10(a)`	base 10 logarithm
`sin(a)`	sine
`cos(a)`	cosine
`tan(a)`	tangent
`asin(a)`	arc sine
`acos(a)`	arc cosine
`atan(a)`	arc tangent
`atan2(a,b)`	arc tangent of a/b
`sinh(a)`	hyperbolic sine
`cosh(a)`	hyperbolic cosine
`tanh(a)`	hyperbolic tangent
`asinh(a)`	arc hyperbolic sine
`acosh(a)`	arc hyperbolic cosine
`atanh(a)`	arc hyperbolic tangent
`sec(a)`	secant
`csc(a)`	cosecant
`ctn(a)`	cotangent

Matrix functions (3×3 only):

`inv(A)`	Inverse of matrix $\boldsymbol{A}$
`tr(A)`	Transpose of matrix $\boldsymbol{A}$
`rx(angle)`	Rotation about x-axis (angle in rad)
`ry(angle)`	Rotation about y-axis (angle in rad)
`rz(angle)`	Rotation about z-axis (angle in rad)

D.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. General expressions (in double quotes) can be used for limits and for the integration variable. Histories must be defined as strings in double quotes and functions of time using the symbol `t`, e.g. `"Velocity(t)"`.

Expression	Symbols
`Integral(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`,`*`variable`*`])`	$\int_a^b f(t)dg(t)$
`Derivative(`*`expression`*`[,`*`T_constant`*`])`	$\Delta f/\Delta t\|_{t=T} \sim \mathrm{d}f/\mathrm{d}t\|_{t=T}$
`Min(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	$f_{\min} = \min_t[f(t)]$
`Max(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	$f_{\max} = \max_t[f(t)]$
`Initial(`*`expression`*`)`	First function value on record
`Final(`*`expression`*`)`	Last function value on record
`Lookup(`*`expression,value`*`[,`*`t_lower,t_upper`*`])`	Inverse function $t(f = F)$
`LookupMin(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	Inverse function $t(f = f_{\min})$
`LookupMax(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	Inverse function $t(f = f_{\max})$
`MeanSqErr(`*`target_curve,computed_curve`*`[,`*`num_reg_points, start_point, end_point, weight_type, scale_type, weight_value, scale_value, weight_curve_name, scale_curve_name`*`])`	Mean Squared Error function $\frac{1}{P}\sum_{p=1}^{P} W_p \left(\frac{f_p(\boldsymbol{x}) - G_p}{s_p} \right)^2$
`Crossplot (`*`history_z, history_F`*` [, numpoints, begin_time, end_time])`	$F(z)$ given $F(t)$ and $z(t)$
`Rotate(`*`x1,y1,z1, x2,y2,z2, x3,y3,z3`*`)`	Rotation matrix defined by 3 points. See Section 14.
`Matrix3x3Init(`*`x1,y1,z1, x2,y2,z2, x3,y3,z3`*`)`	Initialize 3×3 matrix

The arguments used in the expressions have the following explanations:

Argument	Explanation	Symbol	Type
t_lower	lower limit of integration or range	a	generic
t_upper	upper limit of integration or range	b	generic
variable	integration variable	$g(t)$	generic
expression	history defined as an expression string	$f(t)$	generic
value	value for which lookup is required	F	generic
T_constant	specific time	T	generic
target_curve,computed_curve	Target, computed curve names	G	history
Num_reg_points	Number of regression points	n	integer
Start_point, end_point	Location of first/last regression points	z_0, z_P	float
Weight_type, Scale_type	Weight and scale types		reserved
Weight_value, scale_value	Uniform weight and scale values	W, s	float
History_z, history_F	History names for abscissa and ordinate	$z(t), F(t)$	history
numpoints	Number of points in curve	-	integer
Begin_time, end_time	Begin and end times	t_1, t_P	float
x1,y1,z1,x2,y2,z2,x3,y3,z3	Matrix components	-	generic

"Generic" 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 `constant`, `variable`, `dependent`, `history`, `response` and `composite` are acceptable. An expression is given in double quotes, e.g., `"4.2 * C1_1 * Displacement(t)"`.

D.4 Reserved variable names

Name	Explanation
`t`	Time
`LowerLimit`	0.0
`UpperLimit`	Maximum event time over all histories of all solvers

Omitting the lower and upper bounds implies operation over the entire available history.
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 of t in the history (within the specified bounds) is returned. The `LookupMin` and `LookupMax` functions return the value of t at the minimum or maximum respectively.

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.

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.

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.

D.5 Constants associated with histories

The following commands can be given to override defaults for history operations:

Constant	Explanation	Default
`variable fdstepsize`	Finite difference step size for numerical derivatives with respect to variables	0.0001*(Upper bound – Lower bound)
`historysize`	Number of time points for new history	10000

Command file syntax:

```
variable fdstepsize value
historysize integer value
```

- The `variable fdstepsize` is used to find the gradients of expression composite functions. These are used in the optimization process.
- The `historysize` is used when new histories are generated.

D.6 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, constants, dependents, histories, responses constraints and objectives.

Example:

```
constant 'Target1' {12756.333/1000.}
constant 'Target2' {966002/1000.}
variable 'Emod' 1e7
composite 'Residual' type targeted
composite 'Residual' response 'F1' {Target1} scale {Target1}
composite 'Residual' response 'F2' {Target2} scale {Target2}
objective 'Residual'
$
variable fdstepsize {1/500.}
time fdstepsize {1/300.}
history size 10000
```

D.7 Examples illustrating syntax of expressions

Example 1:

The following example shows a simple evaluation of variables and functions. The histories are specified in plot files `his1` and `his2`. A third function `his3` is constructed from the files by averaging.

File `his1`**:**

```
0 0.0
100 1000
200 500
300 500
```

File `his2`**:**

```
0 0.0
100 2000
200 2000
300 2000
```

Input file:

```
"Mathematical Expressions"
$
$ CONSTANTS
$
constants 3
constant 'lowerlimit' 0
constant 'upperlimit' .200
```

```
constant 'angle' 30
$
$ DESIGN VARIABLE DEFINITIONS
$
variables 2
Variable 'x1' 45
Lower bound variable 'x1' -10
Upper bound variable 'x1' 50
Variable 'x2' 45
Lower bound variable 'x2' -10
Upper bound variable 'x2' 50
$
$ DEPENDENT VARIABLES
$
dependents 2
dependent 'll' {lowerlimit * 1000}
dependent 'ul' {upperlimit * 1000}
$
.
.
.
```

```
$
$ HISTORIES
$
history 3
history 'his1' file "../../his1"
history 'his2' file "../../his2"
history 'his3' {(his1(t) + his2(t))/2}
$
$ RESPONSES
$
responses 42
response 'LOWER'     expression {LowerLimit}
response 'UPPER'     expression {UpperLimit}
response 'UL'        expression {ul}
response 'First'     expression {Initial("his1(t)")}
response 'Last'      expression {Final("his1(t)")}
response 'Last3'     expression {Final("(his1(t) + his2(t))/2")}
response 'Max1'      expression {Max("his1(t)")}
response 'Max2'      expression {Max("his1(t)","ll * 1.0")}
response 'Maximum11' expression {Max("his1(t)","ll",ul)}
response 'Maximum32' expression {Max("his3(t)",ll,ul)}
response 'Minimum32' expression {Min("his3(t)",ll,ul)}
response 'Inverse11' expression {Lookup("his1(t)",75)}
response 'Inverse21' expression {Lookup("his2(t)",75)}
response 'Inverse31' expression {Lookup("his3(t)",75)}
response 'Inverse33' expression {Lookup("(his1(t) + his2(t))/2",75)}
response 'MaxI'      expression {max(Inverse11,Inverse21)}
response 'MinI'      expression {min(Inverse11,Inverse21)}
response 'hist'      expression {his3(Inverse31)}
response 'hist66'    expression {his3(66.1) + 0.1}
response 'nhist66'   expression {nint(hist66)}
response 'ihist66'   expression {int(hist66)}
response 'Integ11'   expression {Integral("his1(t)")}
```

```
response 'Integ14'   expression {Integral("his1(t)",ll,ul,"t")}
response 'Integ15'   expression {Integral("his1(t)",ll,UPPER,"t")}
response 'Integ22'   expression {Integral("his2(t)",ll,ul,"t")}
response 'Integ32'   expression {Integral("his3(t)",ll,ul,"t")}
response 'Integ33'   expression {Integral("(his1(t) + his2(t))/2",ll,ul,"t")}
response 'Integ34'   expression {Integral("his3(t)")}
response 'Integ35'   expression {Integral("his3(t)",ll)}
response 'Integ36'   expression {Integral("his3(t)",ll,ul)}
$
$ Cross-functional integrals
$
response 'Integ2'    expression {Integral("his1(t)",ll,ul,"his2(t)")}
response 'Integ3a'   expression {Integral("his1(t)",0,30,"his2(t)")}
response 'Integ3b'   expression {Integral("his1(t)",30,100,"his2(t)")}
response 'Integ4'    expression {Integ1 + Integ2}
response 'Integ5' expression {Integral("sin(t) * his1(t) * his2(t)",ll,ul,"t")}
response 'Integ7'    expression {Integral("sin(t) * his1(t) * his2(t)")}
response 'Velocity1' expression {Derivative("Displacement(t)",0.08)}
response 'Velocity2' expression {Derivative("Displacement(t)")}
$
$ COMPOSITE FUNCTIONS
$
composites 1
composite 'Integ6' {(Integ3a/(4*Maximum11) + Integ2/2)**.5}
$
$ OBJECTIVE FUNCTIONS
$
objectives 1
objective 'Integ6'
$
$ CONSTRAINT FUNCTIONS
$
constraints 1
constraint 'Integ1'
$
iterate 0
STOP
```

Example 2:

```
constant 'v0' 15.65
$-------------------------------------------------------------------------
$ Extractions
$-------------------------------------------------------------------------
history 'engine_velocity'    "DynaASCII nodout X_VEL 73579 TIMESTEP 0.0 SAE 30"
history 'Apillar_velocity_1' "DynaASCII nodout X_VEL 41195 TIMESTEP 0.0 SAE 30"
history 'Apillar_velocity_2' "DynaASCII nodout X_VEL 17251 TIMESTEP 0.0 SAE 30"
history 'global_velocity'    "DynaASCII glstat X_VEL 0 TIMESTEP 0.0"
$-------------------------------------------------------------------------
$ Mathematical Expressions for dependent histories
$-------------------------------------------------------------------------
history 'Apillar_velocity_average' {(Apillar_velocity_1 +
                                     Apillar_velocity_2)/2}
$
$ Find the time when the engine velocity = 0
$
response 'time_to_engine_zero' expression {Lookup("engine_velocity(t)",0)}
$
```

```
$ Find the average velocity at time of engine velocity = 0
$
response 'vel_A_engine_zero' expression {Apillar_velocity_average
                                  (time_to_engine_zero)}
$
$ Integrate the average A-pillar velocity up to zero engine velocity
$ Divide by the time to get the average
$
response 'PULSE_1' expression    {Integral
                      ("Apillar_velocity_average(t)",
                      0,
                      time_to_engine_zero
                      )
                     /time_to_engine_zero}
$
$ Find the time at which the global velocity is zero
$
response 'time_to_zero_velocity' expression {Lookup("global_velocity(t)",0)}
$
$ Find the average A-pillar velocity where global velocity is zero
$
response 'velocity_final' {Apillar_velocity_average(time_to_zero_velocity)}
response 'PULSE_2'    expression {Integral
                      ("Apillar_velocity_average(t)",
                      time_to_engine_zero,
                      time_to_zero_velocity
                      )
                      /(time_to_zero_velocity - time_to_engine_zero)}
```

Appendix E

Simulated Annealing

The Simulated Annealing (SA) algorithm for global optimization can be viewed as an extension to local stochastic optimization techniques. The basic idea is very simple. SA takes a (biased) random walk through the space and aims to find a global optimum from among multiple local solutions. In trying to minimize a function, instead of *always* going downhill, SA algorithm goes downhill *most* of the time. It means that the SA process sometimes goes *uphill*. This allows simulated annealing to move consistently towards lower function values, yet still 'jump' out of local minima and globally explore different states of the optimized system. The SA algorithm was first formulated for various combinatorial problems, [1]. The approach was later extended to continuous optimization problems. In [2] the simulated annealing algorithm was adopted to search for optimal Latin hypercube designs.

The term 'simulated annealing' derives from the rough analogy of the way that the liquids freeze and crystallize, or metals cool and anneal, starting at a high temperature, [1]. When the liquid is hot, the molecules move freely, and very many changes of energy can occur. When the liquid is cooled, this thermal mobility is partially lost. If the rate of cooling is sufficiently slow, the atoms are often able to line themselves up and form a pure crystal, which is the state of minimum (most stable) energy for this physical system. If a liquid metal is cooled quickly or 'quenched', it usually does not reach this state but rather ends up in a polycrystalline or amorphous state having somewhat higher energy. So the essence of the whole process is *slow* cooling.

Nature's minimization algorithm is based on the fact that a system in thermal equilibrium at temperature T has its energy, E, probabilistically distributed among all different energy states as determined by the Boltzmann distribution:

$$\text{Probability}(\, E \,) \sim \exp(-E / \kappa_B T \,). \qquad \text{(F.1)}$$

Hence, even at low temperature, there is a chance, albeit very small, of a system being in a high-energy state. This slight probability of choosing a state that gives higher energy is what allows the physical system to get out of local (i.e. amorphous) minima in favor of finding a better, more stable, orientation. The quantity κ_B (Boltzmann's constant) is a constant of nature that relates temperature to energy.

In simulated annealing algorithm parlance, the objective function of the optimization problem is often called 'energy'. The optimization algorithm proceeds in small iterative steps. At each iteration, SA algorithm randomly generates a candidate state and, through a random mechanism (controlled by a parameter called temperature in view of the analogy with the physical process) decide whether to move to the candidate state or to stay in the current one at the next iteration. More formally, a general SA algorithm can be described as follows.

Step 0. Let $\boldsymbol{x}^{(0)} \in \boldsymbol{X}$ be a given starting state of the optimized system, $E = E(\boldsymbol{x})$.
Start the sequence of observed states: $\boldsymbol{X}^{(0)}=\{\boldsymbol{x}^{(0)}\}$.
Set the starting temperature $T^{(0)}$ to a high value: $T^{(0)} = T_{\max}$, and initialize the counter of iterations to $k = 0$.

Step 1. Sample a point x' from the candidate distribution, $D(\boldsymbol{X}^{(k)})$, and set $\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} \; U \; \{\boldsymbol{x}'\}$.
The sequence $\boldsymbol{X}^{(k+1)}$ contains all the states observed up to iteration k.

Step 2. Sample a uniform random number ζ in [0,1] and set

$$\begin{aligned} &\boldsymbol{x}^{(k+1)} = \boldsymbol{x}' \text{ if } \zeta \le A(\boldsymbol{x}',\boldsymbol{x}^{(k)},T^{(k)}) \text{ or} \\ &\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} \text{ otherwise.} \end{aligned} \tag{F.2}$$

Step 3. Apply the cooling schedule to the temperature, i.e. set $T^{(k+1)} = C(\boldsymbol{X}^{(k+1)}, T^{(k)})$.

Step 4. Check a stopping criterion and if it fails set $k := k+1$ and go back to Step 1.

The distribution of the next candidate state, D, the acceptance function, A, the cooling schedule, C, and the stopping criterion must be specified in order to define the SA algorithm. Appropriate choices are essential to guarantee the efficiency of the algorithm. Many different definitions of the above entities have been given in the existing literature about SA. These will be discussed in the next few paragraphs, trying to emphasize some key ideas that have driven the choices of the researches in this field.

In the existing literature about SA algorithms very few *acceptance functions* have been employed. In most cases the acceptance function is the so-called Metropolis function:

$$A(x',x,T) = \min\left\{1, \frac{\exp(-(E(x')-E(x))}{T}\right\} \tag{F.4}$$

Another possibility is the so-called Barker criterion:

$$A(x',x,T) = \frac{1}{\left(1+\frac{\exp((E(x')-E(x))}{T}\right)} \tag{F.5}$$

The theoretical motivation for such a restricted choice of acceptance functions can be found in [3]. It is shown that under appropriate assumptions, many acceptance functions, which share some properties, are equivalent to (F.4) or (F.5) after a monotonic transformation of the temperature T.

Due to the difficult nature of the problems solved by SA algorithms, it is hard, if not impossible, to define a general *stopping rule*, which guarantees to stop when the global optimum has been detected or when there is a sufficiently high probability of having detected it. Thus the stopping rules proposed in the literature about SA all have a heuristic nature and are, in fact, more problem dependent than SA algorithm dependent. These

heuristics are usually based on the idea to stop the iterative algorithm when it does not make a noticeable progress over a number of iterations.

The choice of the next candidate distribution and the cooling schedule for the temperature are typically the most important (and strongly interrelated) issues in the definition of a SA algorithm. The *next candidate state*, $\boldsymbol{x}'$, is usually selected randomly among all the neighbors of the current solution, $\boldsymbol{x}$, with the same probability for all neighbors. However, with a complicated neighbor structure, a non-uniformly random selection might be appropriate. 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.

The basic idea of the *cooling schedule* is to start the algorithm off at high temperature and then gradually to 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, [4]. After that the simulated annealing algorithm lowers the temperature by slow stages until the system 'freezes' and no further changes occur. A straightforward and most popular strategy is to decrement T by a constant factor every ν_T iterations:

$$T := T/\mu_T \tag{F.6}$$

where μ_T is slightly greater than 1 (e.g. $\mu_T = 1.001$).

The value of ν_T should be large enough, so that 'thermal equilibrium' is achieved before reducing the temperature. A rule of thumb is to take ν_T proportional to the size of neighborhood of the current solution. Often, the cooling schedule (F.6) also provides a condition for terminating SA iterations:

$$T < T_{\min} \tag{F.7}$$

Some of the convergence results for SA rely on the fact that the support of the next candidate distribution is the whole feasible region (though in some cases the probability of sampling states far from the current one decreases to 0 as the iteration counter increases). For these convergence results it is often only required that the temperature decreases to 0, no matter at which rate. For some other convergence results the support of the next candidate distribution is only a neighborhood of the current state, and to make the algorithm able to climb the barriers separating the different local minima, it is required that the temperature decreases to 0 slowly enough.

It is clear that the selection of the initial temperature, $T_{\max}$, has a profound influence on the *rate* of convergence of the SA algorithm. At temperatures much higher than the effective temperature, the algorithm behaves very much like a random search, while at temperatures much lower than the effective temperature it behaves like (an inefficient implementation of) a deterministic algorithm for local optimization. Intuitively, the cooling schedule (F.6) should begin one order of magnitude higher than the effective temperature and end one order of magnitude lower, [4].

It is difficult to give the initial temperature directly, because this value depends on the neighborhood structure, the scale of the objective function, the initial solution, etc. In [1] a suitable initial temperature is one that results in an average uphill move acceptance probability of about 0.8. This $T_{\max}$ 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_{max} is set to a value, which is larger than the expected value of $|E'-E|$ that is encountered from move to move. In [4] it is suggested to spend *most* of the computational time in short sample runs with different T_{max} 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 [5], "choosing an annealing schedule for practical purposes is still something of a black art".

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 terrifically 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 get 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 multistart strategy, i.e. to perform many different runs of the SA algorithm with different starting points.

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 detect quickly the region of the global optimum, they often require many iterations to improve its approximation. 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. But it should be noted that for the 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' is a given solution. Random perturbations of the system state and corresponding cost change calculations should be simple enough, so that SA algorithm can perform its iterations very fast. The scalar parameters of the simulated annealing algorithm (T_{max}, μ_T, ν_T, in particular) 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.

References

[1] Kirkpatrick, S. Gelatt, C.D., Vecchi, M.P. *Science*, 220, pp. 671-680, 1983.
[2] Morris, M., Mitchell, T. Exploratory design for computer experiments. *Journal of Statistical Planning Inference*, 43, pp. 381-402, 1995.
[3] Schuur, P.C. Classification of acceptance criteria for the simulated annealing algorithm. *Mathematics of Operations Research*, 22(2), pp.266-275, 1997.

[4] Basu, A., Frazer, L.N. Rapid determination of the critical temperature in simulated annealing inversion, *Science*, pp. 1409-1412, 1990.
[5] Bounds, D.G. New optimization methods from physics and biology. *Nature*, 329, pp.215-218, 1987.

Appendix F

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*).

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. $\mathrm{d}f/\mathrm{d}x_i$.

Design variable. An independent design parameter which is allowed to vary in order to change the design. Symbolized by (x_i or $\boldsymbol{x}$ (vector containing several design variables)).

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.

***D*-optimal.** The state of an experimental design in which the determinant of the moment matrix $\left|\boldsymbol{X}^T\boldsymbol{X}\right|$ of the least squares formulation is maximized.

DSA. Design sensitivity analysis.

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.

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*.

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.

MP. Mathematical Programming. Mathematical optimization.

MSE. Mean Squared Error. Used for system identification.

Neural network approximation. The use of trained feedforward neural networks to perform non-linear regression, thereby constructing a non-linear metamodels *(see metamodeling)*.

Numerical sensitivity. A derivative of a function computed by using finite differences.

Noise. See *random error*.

Objective. A function of the design variables that the designer wishes to minimize or maximize. If there exists more than one objective, the objectives have to be combined mathematically into a single objective. Symbolized by Φ.

Optimal design. The methodology of using mathematical optimization tools to improve a design iteratively with the objective of finding the 'best' design in terms of predetermined criteria.

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.

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).*

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 `logxxxx` 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.

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 $g_j(x) \le e; \quad e \ge 0$. 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*.

Solver directory. A subdirectory of the work directory that bears the name of a solver 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.

Space Mapping. A technique which uses a fine design model to improve a coarse surrogate model. The hope is, that if the misalignment between the coarse and fine models is not too large, only a few fine model simulations will be required to significantly improve the coarse model. The coarse model can be a response surface.

Stochastic. Involving or containing random variables. Involving probability or chance.

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 is which the input files reside and where output is produced. See also *Run directory*.

Appendix G

LS-OPT Commands: Quick Reference Manual

Note:

All commands are case insensitive.
The commands which are definitions are given in boldface.
Page reference numbers of the syntax definition are given in the last column.
Command phrases in { } are optional.
Names cannot start with a number.

string:	Extraction command, solver/preprocessor command, filename (pathname) in double quotes
name:	Name in single quotes
expression:	Mathematical expression in curly brackets

G.1 Problem description

Constants *number*	The number of constants in the problem	105
Variables *number*	The number of variables in the problem	105
Dependents *number*	The number of dependent variables	105
Histories *number*	The number of histories	105
Responses *number*	The number of responses	105
Composites *number*	The number of composite functions	105
Objectives *number*	The number of objectives	105
Constraints *number*	The number of constraints	105
Solvers *number*	The number of solvers	105
Distribution *number*	The number of probabilistic distributions	105

G.2 Parameter definition

Constant *name value*	constant	144

G.3 Probabilistic distributions

Distribution *name type values* 147

type	*values*
NORMAL	mu sigma
UNIFORM	lower upper
USER_DEFINED_PDF	filename
USER_DEFINED_CDF	filename
LOGNORMAL	mu sigma
WEIBULL	scale shape
BETA	lower upper shape1 shape2

G.4 Design space and region of interest

Variable *name value*	Starting value for design variable	142
Range *name value*	Range of variable to define region of interest	142
Lower bound variable *name value*	Lower bound of Variable	142
Upper bound variable *name value*	Upper bound of Variable	142
Dependent *name expression*	Dependent variable	144
Variable *name max*	Saddle direction flag	146
Constant *name value*	Value of constant	144
Local *name*	Variable is not global	143

G.5 Multidisciplinary or multi-case environment

Solver *package_name name*	software package identifier	129
Solver input file *string*	solver input file name	129
Solver command *string*	solver command line	129
Solver append file *string*	name of file to be appended to input	129
Solver check file *string*	name of checkpoints file	183
Solver extra file *string*	names of extra files (can be repeated)	138
Prepro *name*	software package identifier	133
Prepro command *string*	pre-processor command file	133
Prepro input file *name*	pre-processor input file	133
Prepro output file *name*	pre-processor output file name for Templex	135
Queue *queue type*	queue for workload scheduling	116
Interval *value*	time interval for progress reports	129
Solver concurrent jobs *number*	number of concurrent jobs	115
Solver variable	Flag for solver variable	143

G.6 Package identifiers

ingrid	LS-INGRID		133
truegrid	TrueGrid		134
ansa	ANSA		134
hypermorph	HyperMorph		137
dyna	LS-DYNA	(versions prior to 960)	130
dyna960	LS-DYNA	Version 960/970	130
own	user-defined		133
depmorpher	DEP-Morpher		134

G.7 Queuer identifiers

lsf	Load Sharing Facility
loadleveler	IBM LoadLeveler
pbs	PBS
nqe	NQE
nqs	NQS
aqs	AQS
slurm	SLURM
blackbox	Blackbox
msccp	MS Windows Compute Cluster Server

G.8 Metamodel

Solver order `[linear|elliptic|interaction|quadratic|FF|RBF|user]`	Type of approximating function	174
Solver RBF transfer [`HMQ`|`GAUSS`]	Type of transfer function	168
Solver FF_committee size *number*	Size of a FFNN committee	168
Solver FF_committee discard *number*	Discard 2**number* committee members	168
Solver FF_committee use `[MEAN|MEDIAN]`	Centering procedure for NN evaluation	168
Solver user metamodel *name*	Name (without pre-/suffix)	171
Solver user metamodel path *path*	Metamodel library path	171
Solver user metamodel command *string*	String used by metamodel	171
Solver user metamodel param *value*	Input value	171

G.9 Point selection

Solver experimental design *design*	Experimental design type	174
Solver basis experiment *design*	Basis experiment for *D*-optimal design points selection scheme	174
Solver number basis experiments *number*	Number of experimental points	174
Solver number experiment *number*	Number of experimental points	174
Solver update `doe`	Updating of experimental points	181
Solver experiment duplicate *name*	Duplicate previously defined experiment	178
Solver alternate experiment 1	Alternative experimental design required for first iteration	183
Solver alternate order `[linear]`	Type of alternative approximating function	183
Solver alternate experimental design *design*	Alternative experimental design type	183
Solver alternate basis experiment *design*	Alternative basis experiment for *D*-optimal design points selection scheme	183
Solver alternate number basis experiments *number*	Alternative number of experimental points	183
Solver alternate number experiment *number*	Alternative number of experimental points	183
Solver experiment augment iteration *number*	Change number of points starting with iteration	184

G.10 Point selection types

Experiment Description	Identifier	Default approximation
Linear Koshal	`lin_koshal`	linear
Quadratic Koshal	`quad_koshal`	quadratic
Central Composite	`composite`	quadratic
Latin Hypercube	`latin_hypercube`	linear
Monte Carlo	`monte_carlo`	linear
Plan	`plan`	linear
User-defined	`user`	linear
D-optimal	`dopt`	linear
Space filling	`space_filling`	-
Duplicate	`duplicate`	-
Factorial Designs		
2^n	`2toK`	Linear
3^n	`3toK`	quadratic
⋮	⋮	⋮
11^n	`11toK`	quadratic

G.11 Database recovery

Solver recover dyna `[d3plot|d3hsp|binout|d3eigv]`	Recover DYNA database files of a remote job for given prefix	122
Solver recover file *file_wildcard*	Recover database file(s) of a remote job	123

G.12 Design problem formulation

History *name string*	Defines history function	187
History *name expression*	Defines history function	187
History *name* file *string*	History from file	187
Historysize *number*	Defines maximum number of data points in history function	190
Result *name string*	Defines a result	208
Result *name expression*	Defines a result	208
Matrix *name expression*	Defines a matrix	208
Response *name string*	Defines response function	191
Response *name expression*	Defines response function	191
Response `[linear|elliptic|quadratic|FF|kriging]`	Type of approximation	214
Composite *name* type `[weighted|targeted]`	Type of composite function	213
Composite *name expression*	Defines composite function	214
Composite *name* response *name value** { scale *factor* }	Component definition	214
Composite *name* variable *name value** { scale *factor* }	Component definition	214
Weight *value*	Weight (only `targeted`)	215
Maximize	Maximize objective	222
Objective *name* { *weight* }	Objective definition	222
Constraint *name*	Constraint definition	223
`[Lower|upper]` bound constraint *name value*	Bound on constraint	224
Strict / slack	Slack variable omission status	225
Move / stay / move start	Reasonable space sampling	179
Constraint *name* scale `[lower|upper]` bound *factor*	Constraint scale factor	227

* *value* = target value for `type = MSE`, weight for `type = weighted`

G.13 LS-DYNA result interfaces

`DynaMass p1 p2 p3 ... pn mass_attribute`	Mass	200
`DynaThick [THICKNESS|REDUCTION] p1 p2 ... pm [MIN|MAX|AVE]`	Shell thickness	202
`DynaFLD p1 p2 ... pn intercept neg_slope pos_slope`	FLD	204
`DynaFLDg [LOWER|CENTER|UPPER] p1 p2 ... pn load_curve_id`	General FLD	205
`DynaPStress [S1|S2|S3|MEAN] p1 p2 ... pn [MIN|MAX|AVE]`	Principal stress	206
`DynaFreq mode_original [FREQ|NUMBER|GENMASS]`	Modal data	201
`BinoutHistory –res_type res_type {-sub sub} –cmp component {-invariant invariant –id id –pos position –side side –filter filter_type –filter_freq filter_freq –units units –ave_points ave_points -start_time -start_time start_time –end_time end_time}`	Binout	194
`BinoutResponse {history_options} -select MAX|MIN|AVE|TIME`	Binout	195
`D3PlotHistory –res_type res_type {-sub sub} –cmp component {-id id –pos position -pids part_ids -loc ELEMENT|NODE –select selection -coord x y z -tref ref_state -setid setid}{-start_time start_time –end_time end_time }`	d3plot	197
`D3PlotResponse {history_options} -select selection`	d3plot	199

G.14 Solution tasks

Iterate *n*	Iterate over *n* successive approximations	229
Analyze Monte Carlo	Monte Carlo evaluation	160
Analyze Metamodel Monte Carlo	Monte Carlo evaluation with metamodel	161

G.15 LS-DYNA Results Statistics

analyze dynastat {history *name*}	Compute LS-DYNA results statistics	272
dynastat order *approx_order*	Use metamodels; order of metamodel	268
dynstat outlier *ON/OFF*	Report metamodel outliers	268
dynastat max vector *ON/OFF*	Displacement magnitude formulation	280
dynastat component vector *ON/OFF*	Displacement magnitude formulation	280
dynastat correlation response *name*	Correlation	274
dynstat solver *name*	Solver	272
dynastat iteration *number*	Iteration	272

G.16 Intrinsic functions for mathematical expressions

Note: Trigonometric functions use and return degrees, not radians.

`int(a)`	integer
`nint(a)`	nearest integer
`abs(a)`	absolute value
`mod(a,b)`	remainder of *a/b*
`sign(a,b)`	transfer of sign from *b* to |*a*|
`max(a,b)`	maximum of *a* and *b*
`min(a,b)`	minimum of *a* and *b*
`sqrt(a)`	square root
`exp(a)`	e^a
`pow(a,b)`	a^b
`log(a)`	natural logarithm
`log10(a)`	base 10 logarithm
`sin(a)`	sine
`cos(a)`	cosine
`tan(a)`	tangent
`asin(a)`	arc sine
`acos(a)`	arc cosine
`atan(a)`	arc tangent
`atan2(a,b)`	arc tangent of *a/b*
`sinh(a)`	hyperbolic sine
`cosh(a)`	hyperbolic cosine
`tanh(a)`	hyperbolic tangent
`asinh(a)`	arc hyperbolic sine
`acosh(a)`	arc hyperbolic cosine
`atanh(a)`	arc hyperbolic tangent
`sec(a)`	secant
`csc(a)`	cosecant
`ctn(a)`	Cotangent

3×3 Matrix functions:

`inv(A)`	Inverse of matrix ***A***
`tr(A)`	Transpose of matrix ***A***
`rx(angle)`	Rotation about *x*-axis (angle in rad)
`ry(angle)`	Rotation about *y*-axis (angle in rad)
`rz(angle)`	Rotation about *z*-axis (angle in rad)

G.17 Special functions for mathematical expressions

Expression	Symbols	Type
`Integral(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`,`*`variable`*`])`	$\int_a^b f(t)dg(t)$	Resp.
`Derivative(`*`expression`*`[,`*`T_constant`*`])`	$\Delta f/\Delta t\|_{t=T} \sim \mathrm{d}f/\mathrm{d}t\|_{t=T}$	Resp.
`Min(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	$f_{\min} = \min_t[f(t)]$	Resp.
`Max(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	$f_{\max} = \max_t[f(t)]$	Resp.
`Initial(`*`expression`*`)`	First function value on record	Resp.
`Final(`*`expression`*`)`	Last function value on record	Resp.
`Lookup(`*`expression,value`*`[,`*`t_lower,t_upper`*`])`	Inverse function $t(f=F)$	Resp.
`LookupMin(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	Inverse function $t(f=f_{\min})$	Resp.
`LookupMax(`*`expression`*`[,`*`t_lower`*`,`*`t_upper`*`])`	Inverse function $t(f=f_{\max})$	Resp.
`Crossplot(`*`expr_f,expr_g`*`[,`*`numpts,t_lower,t_upper`*`])`	Crossplot *g(t)* vs. *f(t)*	History
`MeanSqErr(target_G,history_f[,`*`numpts,z_low,z_up, wgt_typ,scl_typ,wgt_val,scl_val, wgt_curve,scl_curve`*`])`	$\frac{1}{P}\sum_{p=1}^{P} W_p\left(\frac{f_p(\mathbf{x})-G_p}{s_p}\right)^2$	Comp.
`Matrix3x3Init(`*`x1,y1,z1, x2,y2,z2, x3,y3,z3`*`)`	Initialize 3×3 matrix	Matrix
`Rotate(`*`x1,y1,z1, x2,y2,z2, x3,y3,z3`*`)`	Rotation matrix defined by 3 points.	Matrix

G.18 Selecting an optimization method

`Optimization method srsm`	Sequential Response Surface Method (SRSM)	253
`Optimization method genalg`	Genetic Algorithm	253

G.19 Setting parameters for optimization algorithm

iterate param *identifier value*	Define parameters in LFOPC	254
iterate param rangelimit variable *value*	Define minimum range of variable in SRSM	255

G.20 Selecting an optimization algorithm for SRSM

`Optimization algorithm lfopc`	Leap Frog Optimizer (LFOPC)
`Optimization algorithm genalg`	Genetic Algorithm