An Overview of New Features in LS-OPT® Version 3.3

Nielen Stander*, Tushar Goel*, David Björkevik**

*Livermore Software Technology Corporation, Livermore, CA94551, USA

**Engineering Research Nordic AB, Linköping, Sweden

Summary:

This paper summarizes the development status of LS-OPT® Version 3.3 and focuses mainly on the following new features: (i) Radial Basis Function Networks, (ii) Multi-objective Optimization (MOO), (iii) Metamodel-based MOO, (iv) User-defined metamodeling, (v) User-defined queuing

Keywords:

LS-OPT, Multi-Objective Optimization, Combinatorial Optimization, Radial Basis Function Networks, User-defined metamodel

1 Introduction: LS-OPT overview

In today's CAE environment it is unusual to make engineering decisions based on a single physics simulation. A typical user conducts multiple analyses by varying the design and uses the combined results for design improvement. LS-OPT® [1] provides an environment for design and is tightly interfaced to LS-DYNA® and LS-PREPOST® with the goal of allowing the user to organize input for multiple simulations and gather and display the results and statistics. More specifically, LS-OPT has capabilities for improving design performance in an uncertain environment and conducting system and material identification. These objectives can be achieved through the use of statistical tools and optimization. The individual tasks that can thus be accomplished are:

- Identify important design variables
- Find the optimal surface (Pareto front) for multi-objective problems
- Explore the design space using surrogate design models
- Identify sources of uncertainty in FE models
- Visualize statistics of multiple runs
- Optimize the design with consideration of uncertainties
- Conduct robust parameter design

The typical applications are: Multidisciplinary Design Optimization (crashworthiness, modal analysis, durability analysis, etc.), system and material identification (biomaterials, metal alloys, concrete, airbag properties, etc.) and process design (metal forming).

The main technologies available in LS-OPT are:

Experimental Design (DOE). D-Optimal design, Latin Hypercube sampling, Space Filling and others. DOE allows the user to automatically select a set of different designs to be analyzed. The main types mentioned here are each suited to a different type of analysis: *D*-Optimal for polynomials and sequential optimization, Latin Hypercube for stochastic analysis and Space Filling for Neural Networks.

Metamodels (approximations). Response Surface Methodology and Neural Networks are provided. Both Feedforward Neural Networks and Radial Basis Function Networks are available. With these tools, the user can explore the design space and quantify the predictability of a response, i.e. identify sources of noisy response. A user-defined metamodel can also be specified by creating a dynamically linked library.

Variable screening [4] provides information on the relative importance of design variables.

Optimization. Used for automated design improvement. The Successive Response Surface Method (SRSM) [5] is the principal iterative tool for finding a converged optimum and is very efficient. A similar methodology is used for finding a converged result using Neural Net updating with adaptive Space Filling. Multi-objective optimization can be activated by selecting more than one objective together with the GA core solver. This combination will produce a Pareto front. Optimization with discrete variables is possible, as is combinatorial optimization. A direct multi-objective GA optimizer is used for the latter.

Probabilistic analysis includes Reliability Analysis, Outlier Analysis, Robust Parameter Design and Reliability-Based Design Optimization (RBDO)[3]. Reliability analysis allows the user to evaluate the probability of failure while Outlier Analysis allows the identification of parts of a model that contribute to noisy response and therefore may affect the overall predictability of the results. Robust Parameter Design and RBDO allow for defining robustness as an objective and the consideration of the probability of failure as a constraint option in optimization. The outlier analysis uses integrated LS-PREPOST features to visualize structural zones with unpredictable behavior.

Features are available to distribute simulation jobs across a network, using a variety of standard and user-defined queuing systems.

In the sections that follow, a number of new features are discussed, namely Radial Basis Function Networks, User-defined metamodels, a GA for direct combinatorial optimization and multi-objective optimization.

2 New Metamodels

Two new metamodels have been introduced. The first, Radial Basis Function Networks, has been added to enhance the speed of building metamodels while the User-defined metamodel has been added for the benefit of commercial users with proprietary metamodel codes as well as for researchers who want to test new metamodels, using the optimization and viewing features available in LS-OPT. A detailed description of RBFN's follows.

2.1 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.

2.1.1 Formulation

Several forms of radial basis function are considered in the literature, the most common being the Gaussian

$$g_h(x_1,...,x_k) = \exp\left[-\sum_{k=1}^{K}(x_k - W_{hk})^2 / 2\sigma_h^{\,2}\right] \qquad (1)$$

The activation of the *h*th Gaussian unit 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 is a localized function (peaked at the center and descending outwards – see Fig. 1) with the property that $g_h \to 0$ as $r \to \infty$. Parameter σ_h controls the smoothness properties of the RBF unit.

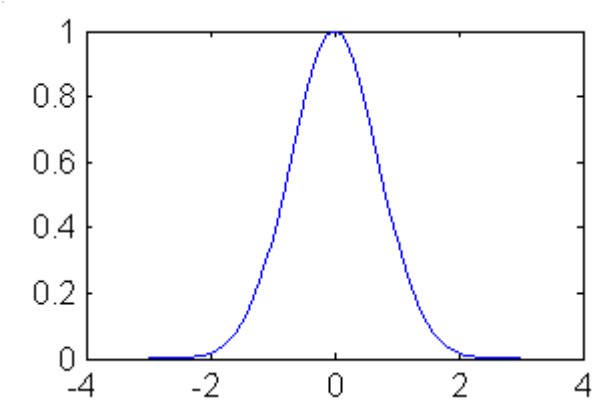

Fig. 1: Radial basis transfer function $y = \exp[-x^2]$

For a given input vector $x = (x_1,...,x_K)$ the output of RBF network with K inputs and a hidden layer with *H* Gaussian units is given by:

$$Y(\boldsymbol{x},W) = W_0 + \sum_{h=1}^{H} W_h \cdot f(\rho_h) \qquad (2)$$

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 the *h*th radial unit, while W_{h0} corresponds to its deviation. Parameters W_0 and $W_1,...,W_H$ are the output layer's bias and weights, respectively.

A 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.
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. Here it is assumed that the RBF parameters have already been chosen, and the focus is 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. If the hidden layer's parameters $W_{10},W_{11},...,W_{HK}$ in (2) are kept fixed, then the performance function $MSE = \sum_{i}^{P} (\hat{y}_i - y_i)^2 / P,$ 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 symbol y represents the function values, $\hat{y}$ the interpolated (approximate) values and P is the number of points.

2.1.2 Locating centers and radii of the basis functions

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. There are a number of ways to address this problem. By analogy with FF network training, one can add to the MSE a regularization term that consists of the mean of the sum of squares of the optimized weights.

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.

In LS-OPT, instead of using just one value of the width parameter σ_h for all RBF's, each RBF unit's deviation is individually set to the distance to its $N_\sigma << N$ nearest neighbors where N is the number of simulation points. 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. RBF networks with individual radial deviations σ_h can be particularly useful in situations (such as the sequentially updated network in LS-OPT) where data tend to cluster in 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.

After the first phase of RBF training has been done, there is no way to compensate for large inaccuracies in radial deviations σ_h by, say, adding a regularization term to the performance function. If the Gaussians 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 (generalized cross validation) error. This is somewhat analogous to searching for an optimal number of hidden units of a feedforward neural network.

2.1.3 Efficiency

It should be noted that RBF networks may have certain difficulties if the number of RBF units (H) is large. A large number of RBF units increases the computation time spent on optimization of the network output layer and, consequently, the RBF architecture loses its main (if not the only) advantage over FF networks – fast training. An active learning procedure has therefore been introduced in LS-OPT in which the RMS training error threshold can be prescribed. The algorithm loops through the numbers of centers which represent the fractions (0.1, 0.25, 0.5 and 1.0) of the total number of training points. If a particular fraction satisfies the threshold, the algorithm exits from the loop. Linear functions have been added to the RBF's and therefore if a function is exactly linear, it will immediately satisfy the threshold criterion and the RBF's are not constructed. A random selection is used to determine the subset of participating centers.

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.

2.2 User-defined metamodel

The user can now integrate an own metamodel with LS-OPT. Building tools are provided to enable the building of a library that is dynamically linked to LS-OPT (LS-OPT object code is therefore not required for the building process). The user metamodel toolkit is distributed as a separate file which includes a source code template and building tools for Microsoft Windows (Visual C) and Linux. The following optional parameters can be specified in the LS-OPT input file:

1. The metamodel path
2. Any number of numerical parameters
3. A command string

This feature adds one more user-defined facility to the user-defined experimental design, solver, response extraction and queuing systems already in place.

3 Constrained Multi-Objective Optimization

3.1 Overview

When two or more objective functions in a design formulation are in conflict, the resulting optimum becomes a hypersurface instead of a single point in the function space. This hypersurface is known as the Pareto optimal front. A typical example of conflicting objectives in crashworthiness design is the deceleration of the vehicle and the intrusion into the vehicle. Classical optimization methods are typically able to only find one optimal solution at a time (using for instance the ε-constraint method or the weighted objective method). Some methods e.g. the weighted objective method are not able to

find the full Pareto front if it is non-convex (it typically finds some values at the extremes). In this case a more sophisticated procedure is required. Implemented in LS-OPT is the multi-objective genetic algorithm. Many variants of this algorithm have been presented in the literature, but they all have in common the dual objective of (i) finding optimal solutions to a linear combination of the objective functions and (ii) maximizing the diversity of the solutions to form the Pareto optimal front. The last goal is entirely specific to multi-objective evolutionary optimization. The book by Prof. Kalyanmoy Deb [6] is wholly dedicated to the subject of multi-objective optimization and has a detailed discussion of the most important evolutionary algorithms for achieving both optimality and diversity simultaneously.

LS-OPT features the NSGA-II (non-dominated sorting genetic algorithm II) [7] for multi-objective optimization. The capabilities are the following:

1. Mixed discrete-continuous optimization. The discrete variables may also be integer variables, e.g. material types.
2. Multi-objective optimization. The solution converges to the Pareto optimal front.
3. Global optimization. Due to the fact that random solutions are generated by the GA, the design space is widely (not just locally) explored for new solutions.
4. NSGA-II can be applied both using *direct simulation* and *metamodel-based analysis*.

3.2 Sequential updating procedure

The multi-objective procedure is best applied using either a single iteration with a large number of points but can also be used with the sequential updating procedure available in LS-OPT. This procedure typically starts with a linear response surface based on a *D*-optimal experimental design and in each new iteration adds Space Filling [1] points for the purpose of enriching the metamodel regionally. LS-OPT uses a linear combination of the objectives as a multi-objective to determine a single optimum point which serves as a center for the new region of interest. The objective weighting is uniform, but can be set individually by the user. It is therefore a good idea to normalize the objectives. The Pareto front is automatically updated after each iteration, so it will be *automatically* displayed in the viewer (Tradeoff selection) for each iteration (see example).

3.3 Post-processing

The Pareto curve can be amended in accordance with design changes (such as changes in the constraints and their bounds, changes in the objectives themselves or changes in the metamodel selection – any design or analysis changes for that matter). The limited “Tradeoff” feature previously available in the viewer has been rescinded in favor of using the existing GUI “Constraints” and “Objectives” panels to enable the user to make these changes interactively. After making the desired changes, the new Pareto optimal front can be computed using the “Optimization” selection in the “Repair” task.

4 Example: Multidisciplinary, constrained Multi-Objective Optimization using sequentially updated Radial Basis Function Networks

4.1 Modeling

The crashworthiness simulation considers a model containing approximately 30 000 elements of a National Highway Transportation and Safety Association (NHTSA) vehicle 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 2. The vibration model is depicted in Figure 3 in the first torsional vibration mode. The tracking methodology applied to the torsional mode is described in Reference [1]. The design variables all represent gauges of structural components in the engine compartment of the vehicle (Figure 4), parameterized directly in the solver input file. Twelve parts are affected, comprising aprons, rails, shotguns, cradle rails and the cradle cross member. Seven gauge variables namely apron, rail-inner, rail-outer, shotgun-inner, shotgun-outer, cradle rail and cradle cross member are selected to represent the parts.

LS-DYNA [2] is used for both the crash and modal analysis simulations, in explicit and implicit analysis modes respectively.

4.2 MDO multi-objective formulation of optimization problem

The following optimization problem is considered:

Minimize (Mass, Intrusion)

Subject to:

	Lower bound	Upper bound
Maximum intrusion ($\boldsymbol{x}_{\text{crash}}$)	-	551.27mm
Stage 1 pulse($\boldsymbol{x}_{\text{crash}}$)	14.512 *g*	-
Stage 2 pulse($\boldsymbol{x}_{\text{crash}}$)	17.586 *g*	-
Stage 3 pulse($\boldsymbol{x}_{\text{crash}}$)	20.745 *g*	-
Torsional mode frequency($\boldsymbol{x}_{\text{NVH}}$)	38.27Hz	39.27Hz

Table 1 – Design constraints

Variable Name	Lower Bound	Baseline Design	Upper Bound	Discrete set	Vibration Set After Screening	Crash Subset After Screening
Rail inner	-	2	-	{1,1.25,1.5,1.75,2 ,2.25,2.5,2.75,3}	●	●
Rail outer	-	1.5	-	{1,1.25,1.5,1.75,2 ,2.25,2.5,2.75,3}	●	●
Cradle rails	1	1.93	3	-	●	●
Aprons	1	1.3	3	-	●	●
Shotgun inner	1	1.3	3	-	●	
Shotgun outer	1	1.3	3	-	●	
Cradle cross member	1	1.93	3	-	●	

Table 2 – Starting values and bounds on design variables.

The bounds on the design variables are given in Table 2 together with the starting design.

The three stage pulses are calculated from the SAE filtered (60Hz) acceleration $\ddot{x}(t)$ and displacement $x(t)$ of a left rear sill node in the following fashion:

$$\text{Stage } j \text{ pulse} = -k\int_{d_1}^{d_2} \ddot{x}\mathrm{d}x; k = 0.5 \text{ for } j = 1, k = 1.0 \text{ otherwise;}$$

with the limits (d_1;d_2) = (0;184); (184;334); (334;Max(displacement)) for i = 1,2,3 respectively, all displacement units in mm and the minus sign to convert acceleration to deceleration.

In summary, the optimization problem aims to minimize the mass and intrusion without compromising the crash and vibration criteria. The constraints are scaled using the target values to balance the violations of the different constraints. This scaling is important in cases where multiple constraints may be violated as in the current problem. The inner and outer rails are discrete variables.

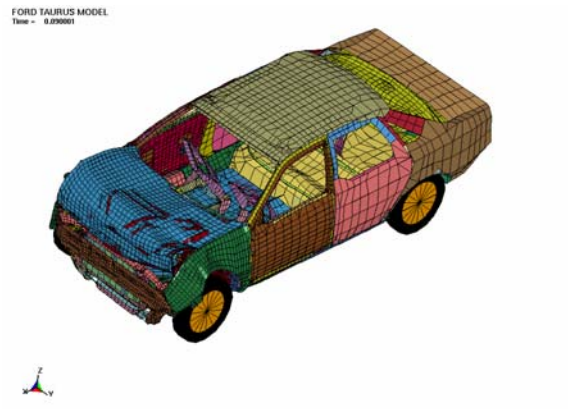

Fig. 2: Finite element crash model

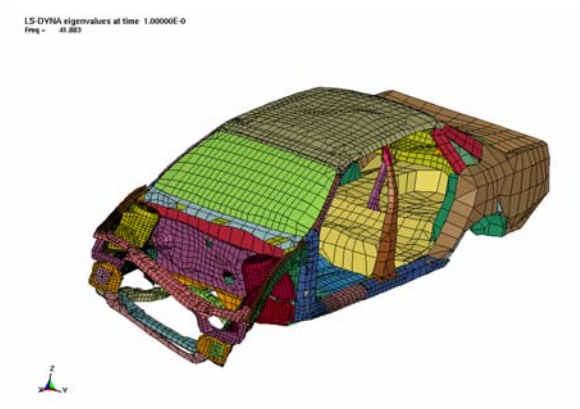

Fig. 3: Finite element vibration model

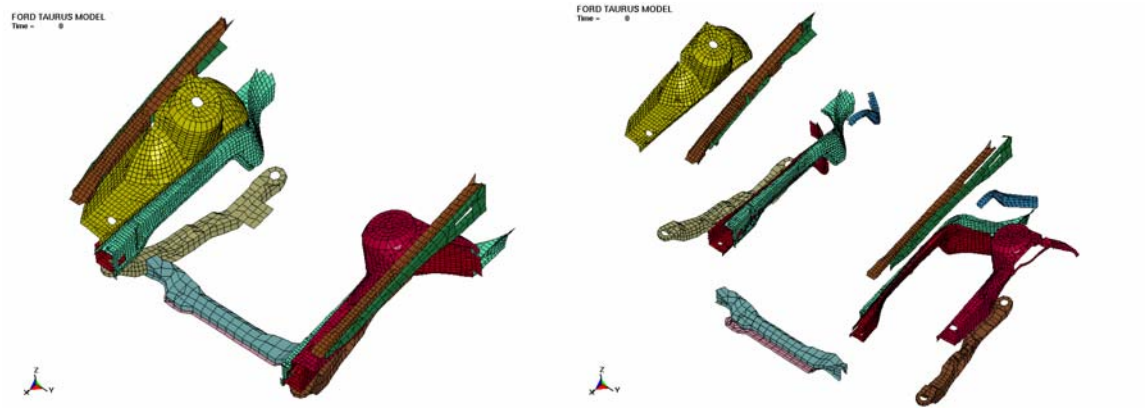

Fig. 4: Thickness design variables (with exploded view)

4.3 Results

The following plots compare the results of the RBF and FFNN. The intrusion metamodel is shown in Figure 5. The plots have identical scale settings for the vertical axis. The Pareto optimal front (Figure 6) is based on the dual objectives of predicted intrusion and mass. Figure 7 has been introduced to illustrate the reason for the uneven Pareto optimal front for the FFNN which occurs as a result of a sudden jump of the optimum when tightening the intrusion constraint. The transition also exists for the RBF but, for some reason, is smoother. Finally, Figure 8 depicts the optimization history of the mass.

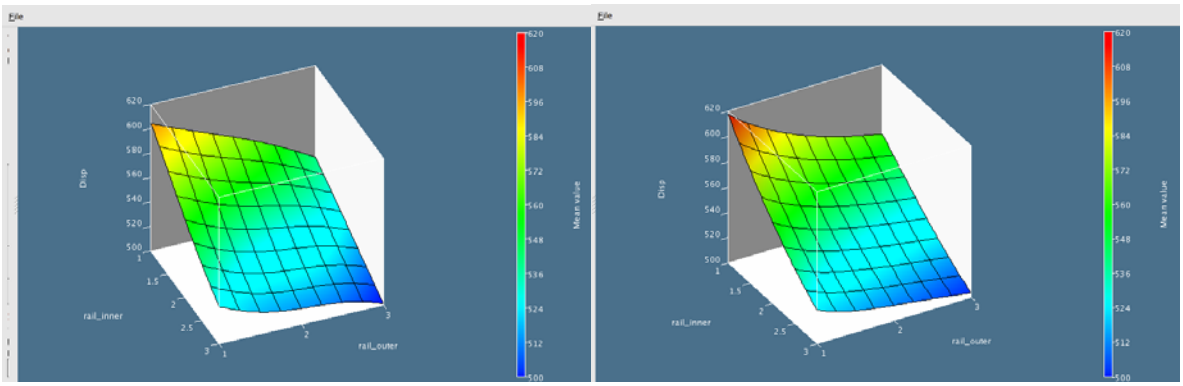

Fig. 5: Metamodel of the intrusion as a function of the inner and outer rail thicknesses (a) Radial Basis Function Networks and (b) Feedforward NN.

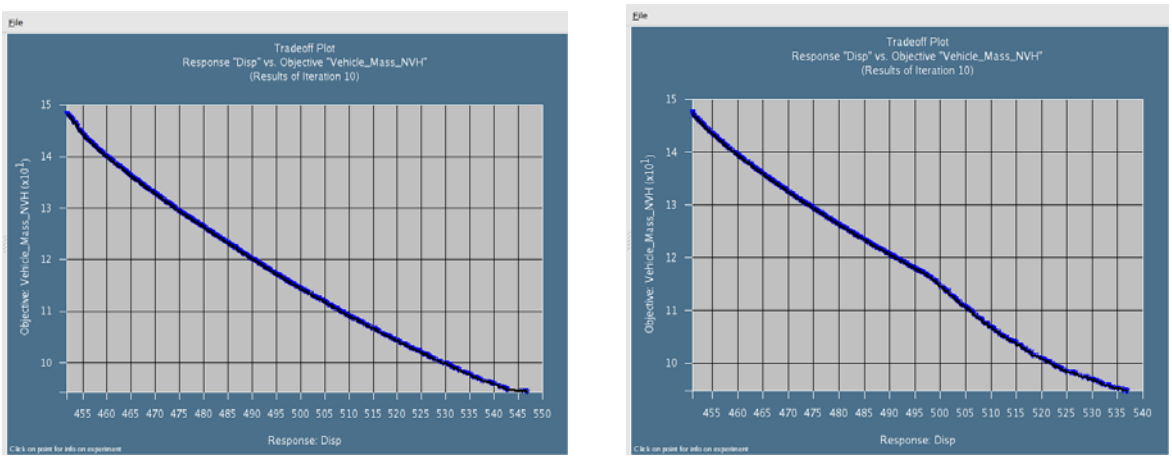

Fig. 6: Pareto optimal front using (a) Radial Basis Function Networks and (b) Feedforward NN

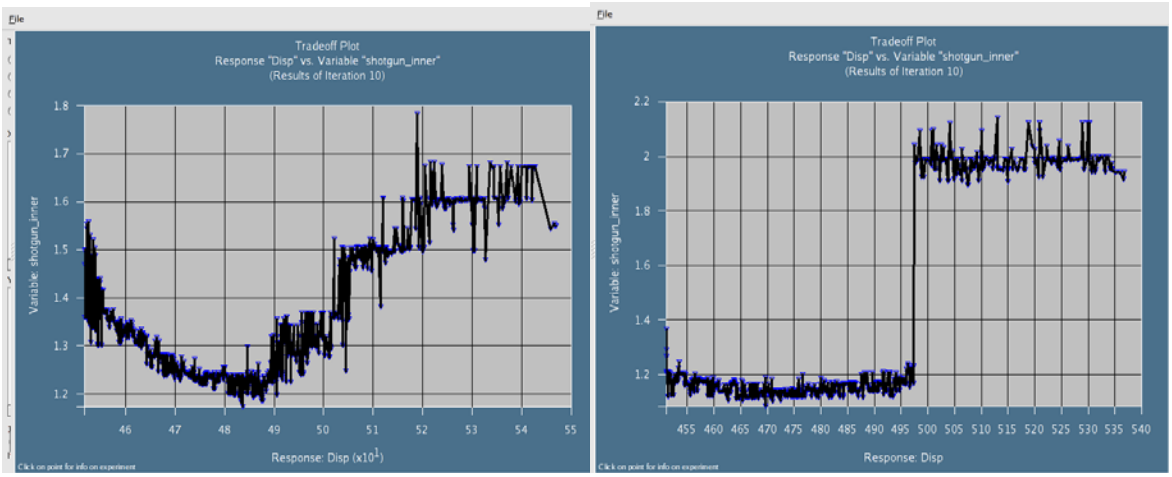

Fig. 7: Plot of the Pareto optimal inner shotgun thickness vs. Intrusion to illustrate the reason for the 'kink' in the Feedforward NN, not present in the RBF surrogate.

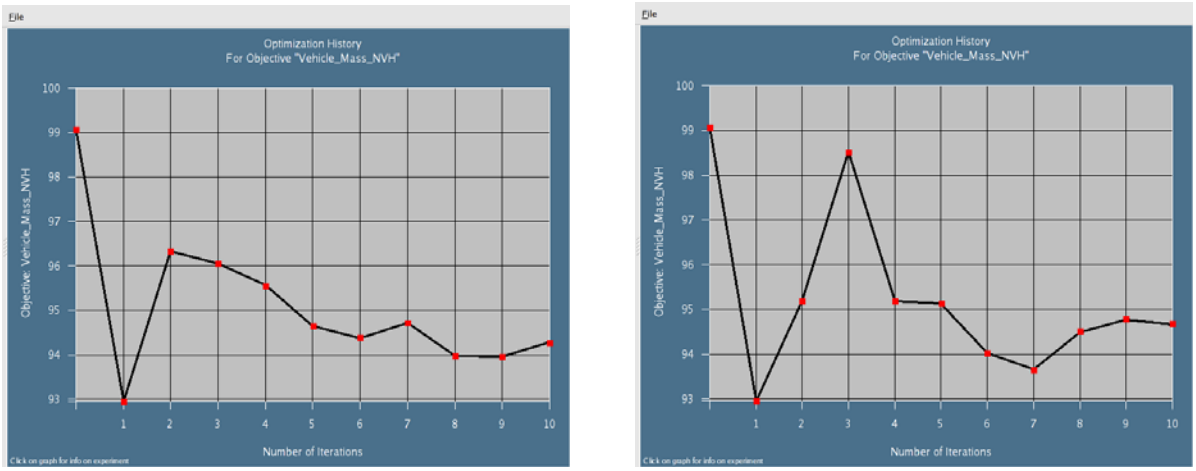

Fig. 8: Mass convergence for (a) Radial Basis Function Networks and (b) Feedforward NN

The final optimum results after 10 iterations is presented in Table 3.

	RBF	FFNN
Rail inner	2.25	2.75
Rail outer	1.5	1.5
Cradle rails	2.191	1.358
Aprons	1.004	1.024
Shotgun inner	1.55	1.94
Shotgun outer	1	1.29
Cradle cross member	1	1
Mass	94.28	94.68
Maximum intrusion	550.5	535.1
Stage 1 pulse	15.41	15.82
Stage 2 pulse	17.95	17.94
Stage 3 pulse	20.22*	21.61
Torsional mode frequency	41.41	41.36*
Max. constraint violation (fraction of constr. value)	0.0251	0.00059

Tables 3 – Optimum results (10 iterations). The * indicates a violated constraint

5 'Blackbox' user-defined queuing 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.

6 Conclusion

LS-OPT Version 3.3 presents a significant step forward for optimization with LS-DYNA by providing powerful new methods for combinatorial optimization and multi-objective optimization. New metamodels have been added to improve speed and flexibility.

7 Acknowledgement

The authors would like to acknowledge the work of Nely Fedorova and Serge Terekhoff who developed the Neural Network and Radial Basis Function Network codes.

8 References

1. Stander, N., Roux, W.J., Goel, T., Eggleston T. and Craig, K.J. LS-OPT® Version 3.3 User's Manual, Livermore Software Technology Corporation, October 2007.

2. Hallquist, J.O. LS-DYNA® User's Manual, Version 971.

3. Roux, W.J., Stander, N., Günther, F. and Müllerschön, H. Stochastic analysis of highly nonlinear structures, *International Journal for Numerical Methods in Engineering,* Vol. 65:1221-1242, 2006.

4. Craig, K.J. and Stander, N., Dooge and Varadappa, S. Automotive crashworthiness design using response surface-based variable screening and optimization, *Engineering Computations,* Vol. 22:38-61, 2005.

5. Stander, N. and Craig, K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization, *Engineering Computations*, Vol. 19:431-450, 2002.

6. Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001

7. Deb, K and Goel, T. Controlled elitist non-dominated sorting genetic algorithms for better convergence. *Proceedings of the First International Conference on Evolutionary Multicriterion Optimization (EMO-2001),* pp 67-81, 2001

