Metamodel Sensitivity to Sampling Strategies: A Crashworthiness Design Study

Nielen Stander and Tushar Goel
Livermore Software Technology Corporation
7374 Las Positas Road, Livermore, CA 94551

Abstract

A study is conducted to determine the sensitivity of 2 topologically distinct metamodel types to variations in the experimental design brought about by sequential adaptive sampling strategies. The study focuses on examples encountered in crashworthiness design. Three sampling strategies are considered for updating the experimental designs, namely (i) a single stage approach, (ii) a sequential approach and (iii) a sequential approach, but with higher densities in local regions. The experimental design type is the Space Filling Method based on maximizing the minimum distance between any two design points within a subdomain. Feedforward Neural Networks (NN) and Radial Basis Function Networks (RBF) are compared with respect to their sensitivity when applied to these strategies. A large set of independent checkpoints, constructed using a Latin Hypercube Sampling method is used to evaluate the accuracy of the various strategies. Four examples are used in the evaluation, namely (i) simple two-variable two-bar truss, (ii) the 21 variable Svanberg problem, (iii) a 7 variable full vehicle crash example and (iv) a 11 variable knee impact crash example. The example, analyzed using LS-OPT® for metamodeling and LS-DYNA® for FE modeling, reveal the following: while expensive to construct, NN committees tend to be superior in predictability whereas the much cheaper RBF networks, can sometimes be highly sensitive to irregularity of experimental designs caused by subdomain updating. However, this conclusion cannot be extended to the crash problems tested, since the RBF networks performed consistently well for these examples.

Introduction

Along with experimental design, metamodels play a crucially important role in simulation-based optimization. Due to (i) the presence of noise in the response, (ii) the typical unavailability of analytical gradients and (iii) the requirement of obtaining a global surrogate model for design exploration, crashworthiness analysis requires the use of metamodeling techniques for design. FE models for crashworthiness analysis are highly complex and include highly non-linear phenomena such as contact, buckling, frictional sliding and material nonlinearity. Hence, nonlinear dynamic analysis is used and will always produce a displacement, velocity or acceleration response which is noisy to some degree. Factors such as mesh adaptivity, rounding and platform dependency are additional contributors to noisy behavior. To enable optimization or probabilistic analysis of the response, an accurate metamodel is imperative. After doing the initial expensive simulations, the subsequent metamodel becomes the surrogate and will represent the mechanical design for the remainder of the design process.

The most basic approach to metamodeling, and the earliest to be used, is response surface methodology (RSM) [1]. RSM is polynomial-based, typically uses a full or fractional factorial or *D*-optimal experimental design as sampling scheme and relies only on linear regression to solve for the function coefficients. Because of the highly nonlinear nature of crash analysis, other metamodel types have been introduced to deal with the complexity of the response functions. Two well-established methods are Feedforward Neural Networks (FFNN) [2][3] and Radial Basis Function Networks (RBFN) [4][5]. These methods have some similarities, but differ greatly in their choice of basis functions and solution approach. While FFNN's require nonlinear regression, RBFN's allow the regression process to be split into multiple levels, the innermost being linear while multiple outer line search loops deal with the nonlinearity induced by some of the parameters notably the spread and regularization factors and the topology (number of basis functions).

Metamodel comparisons have been made by several researchers. Jin *et al* [6] investigated the effect of various sequential schemes (Entropy approach, the Integrated Mean Squared Error (IMSE) approach, and the Maximin Distance approach) on the relative merits of Kriging and RBF metamodels. They came to the conclusion that Kriging tended to be better for smooth functions whereas if the unknown function is irregular or unpredictable, RBF networks performed better. An earlier study by Jin *et al* [7] included a comparative study of Polynomial Regression, Multivariate Adaptive Regression Splines (MARS), Radial Basis Functions and Kriging. In this study it was concluded that RBF excels in most categories of degree of nonlinearity and problem scale. Hence the RBF Networks were also chosen as a suitable component of the current study. The RBF Networks metamodel type is a new addition to LSOPT® Version 3.3 .

The design environment often requires the application of a non-standard, irregular experimental design. While an optimization run is ideally based on a single stage of runs, i.e. a series of runs based on a standard experimental design method the user is sometimes obliged to use a sequential method because of the flexibility it provides to run simulations until the metamodel converges. These sequential adaptive schemes can augment the set of designs locally or globally, possibly resulting in irregular experimental designs with higher point densities in a certain area or areas. Apart from sequential adaptive schemes, poor experimental designs may also arise when a number of runs fail due to finite element modeling deficiencies. In this case it is important to construct a surrogate with good predictive capability using the available points.

This study investigates the robustness of the two afore-mentioned metamodeling types: FFNN's and RBFN's to variations in the experimental design method. A sequential adaptive domain reduction scheme [8] is used as a realistic irregular experimental design for detecting sensitivity of the sampling scheme. As reference cases, the Max-Min distance designs [9] are used. Five examples are used to investigate the relevant metamodel properties. Two of these are analytical namely (i) the two-bar truss, (ii) the 21 variable Svanberg problem. The crash examples are (iii) a 7 variable full vehicle crash and vibration multidisciplinary example and (iv) an 11 variable knee impact example.

Theory

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, 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:

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

where

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

Standard non-linear optimization techniques including a variety of gradient algorithms (the steepest descent, RPROP, Levenberg-Marquardt, etc.) can be applied to obtain the FF network's weights and biases. 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) [2]. However, when training larger networks, the first-order RPROP algorithm becomes preferable for computational reasons [10].

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 ([11], [12]). 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 the PRESS (Prediction Sum of Squares computed using a leave-one-out approach). For FF networks, this procedure is usually too expensive and an approximate quantity: GCV = $MSE/(1-\nu/P)^2$ is used, where ν is the effective number of model parameters. A common scheme is 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 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 [13]. 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 MSE. When minimizing MSE, the weight estimates tend to be exaggerated. A penalty is imposed for this reason by adding a term that consists of the sum of squares of the network weights (see also (1)):

$$F = \beta \cdot MSE + \alpha\, E_W$$

where

$$MSE = \frac{1}{P}\sum_{p=1}^{P}\left(\hat{y}_p - y_p\right)^2$$

$$E_W = \sum_{m=1}^{M} {W_m}^2$$

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 F will cause the network to have smaller weights, and this will force the network response to be smoother and less likely to overfit thus eliminating the guesswork required in determining the optimum network size. A description of how to determine α and β can be found in [14].

Neural networks have a natural variability for the following reasons [15]:

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. In this study, NN committees consisting of a membership on 9 neural nets were used to find the average NN [2]. Each member of the committee is constructed by starting the nonlinear regression procedure at a different starting point of the weight values ${W_m}^{(0)}$.

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. The most common basis functions are Hardy's multi-quadrics functions and the Gaussian function. These are given as:

Hardy's multi-quadric:

$$g_h(x_1,...,x_k) = \sqrt{1+\frac{r^2}{{\sigma_h}^2}}$$

Gaussian:

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

The activation of the 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:

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

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 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.

Sequential domain reduction

The purpose of sequential domain reduction is to allow convergence of the solution to a prescribed tolerance. The SRSM (Sequential Response Surface Method) [8] 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 1 shows the possible adaptation of the subregion. The nominal reduction of the region of interest is 0.25 (i.e. the range of each variable reduces to 0.75 of its former range) for each iteration. However, heuristic rules dependent on the amount of oscillation and the proximity of the new optimal design with respect to the previous one are used to determine a reduction factor between 0 and 0.25. The experimental design is augmented within this new region of interest, maintaining the maximin distance with respect to all existing points as well as new points. The maximin distance is obtained using simulated annealing.

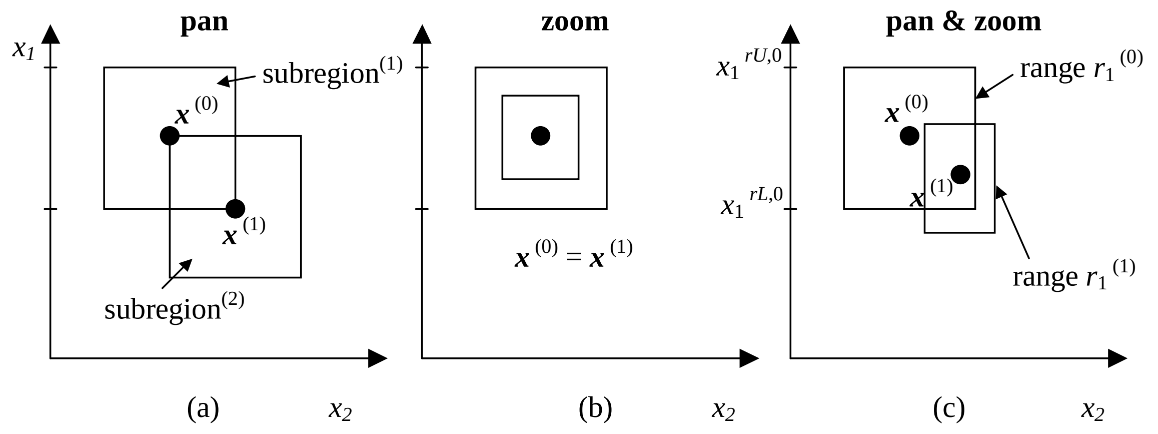

Figure 1: Adaptation of subregion in SRSM: (a) pure panning, (b) pure zooming and (c) a combination of panning and zooming

The sequential sampling method *without* domain reduction simply adds new points to the full design space by using the criterion of maximizing the minimum distance between any of the new points as well as between new and existing points.

Examples

A number of examples are provided to compare the different metamodels. There are two analytical examples, the 3 variable 2-bar truss problem and the Svanberg 21 variable problem. The remaining three examples are: (i) a full vehicle crash and vibration analysis with 7 sizing variables and (ii) a knee impact analysis with 11 sizing and shape variables. Two metamodel types are compared: (i) Feedforward Neural Networks (NN) and (ii) Radial Basis Function Networks (RBF). Both 9- and single member committees are used for the NN's and denoted NN-9 and NN-1 respectively. The RBF networks use Hardy's Multi-quadric basis functions. LS-OPT® [14] was used to build the respective metamodels while LS-DYNA® [16] was used to conduct the crash and vibration simulations.

The single stage global approximation is an approximation based on a single stage maximin experimental design constructed by maximizing the minimum distance between any two points. The sequential subdomain updating is conducted as follows: The first iteration uses a *D*-optimal experimental design with linear approximation. From the second iteration onwards, the region of interest is reduced to a sub-domain (see Reference [8]). This results in an increasingly dense sampling as the sub-domain continues to reduce in size. 10 Iterations are used for all the examples. The number of simulations per iteration is represented by the thumb rule $1.5(n+1)+1$ where n is the number of design variables. For the single stage method, the total number of points is the same as for the iterative method. This thumb rule ensures that the number of sampling points in the first iteration represents a slight oversampling for a linear approximation.

The metamodel types and sampling schemes are compared by means of a large set of independent checkpoints constructed using the Latin Hypercube Sampling scheme. An RMS error value is computed for each function as follows:

$$RMSE = \sqrt{\frac{1}{P}\left(y_p - \hat{y}_p\right)^2}$$

Where P is the number of checkpoints, y_p is the function value at the checkpoint and $\hat{y}_p$ is the function value predicted using the metamodel. The reported RMSE values are also normalized with respect to the mean value:

$$\frac{1}{P}\sum_{p=1}^{P} y_p \ .$$

2-Bar truss with 3 variables

The first problem is of a simple two-bar truss. A linear analysis is conducted using simple user defined formulas The height of the structure = 1. The force components are: *Fx* = +24.8kN, *Fy*=198.4kN. The criteria are weight and stress in each of the bars. Three design variables are chosen, namely the cross-sectional area of the bars and the base measurement between the supports. It should be noted that the StressL and StressR functions are highly nonlinear functions of the cross-sectional areas AreaL and AreaR respectively.

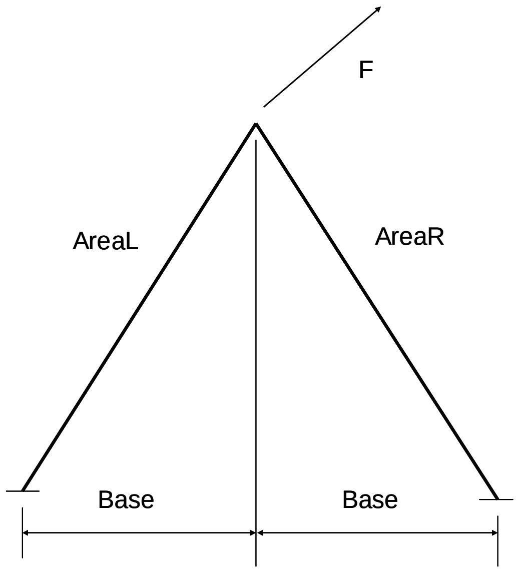

Figure 2: Two bar truss

The results are presented in tables Table 1 and Table 2. Table 1 represents the RMSE while Table 2 represents the Maximum errors. The statistics are based on 1000 checkpoint simulations.

	RMSE (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
Weight	**0.0891**	0.0985	0.522	**0.0648**	0.106	0.0966	0.356	**0.332**	1.12
StressL	9.86	**8.56**	35.8	**6.01**	7.49	24.2	**11.2**	22.3	57.2
StressR	**8.46**	9.24	43.6	8.04	**5.89**	42	**17.2**	22.6	86.3

Table 1: 2-Bar truss. 70 Simulation points. RMSE of a set of 1000 LHS checkpoints as a percentage of the mean value for NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network. Best fits are shown in bold.

	Maximum error (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
Weight	0.774	**0.763**	3.54	**0.506**	0.641	0.651	**1.68**	2.19	5.34
StressL	135	**119**	343	**36.9**	55.8	158	**100**	264	269
StressR	**108**	115	390	79.2	**57.4**	344	**110**	129	439

Table 2: 2-Bar truss. 70 Simulation points. Maximum error of 1000 LHS checkpoints as a percentage of the mean value for NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function. Best fits are shown in bold.

The most obvious and unexpected result for the two-bar truss is the poor performance of the RBF's, even when using a single stage experimental design.

Svanberg problem with 21 variables

This example has 21 design variables and was proposed by Svanberg in 1995 [17]. The formulation is as follows:

$$f = -\frac{1}{2}\sum_{i=0}^{10} x_i x_{i+1} \sin\left(\frac{\pi}{180} x_{i+11}\right)$$

$$g_1 = x_0 + x_{10} + \sum_{i=0}^{10} \sqrt{x_i^{\,2} + x_{i+1}^{\,2} - 2x_i x_{i+1} \cos\frac{\pi}{180} x_{i+11}}$$

Table 3 and Table 4 represent the results for the Svanberg problem. In contrast to the previous example, the RBF performs significantly better than the neural network. Table 3 represents the RMSE while Table 4 represents the Maximum errors.

	RMSE (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
f	12.6	16.9	**9.24**	14.3	17.8	**8.66**	17.3	26.3	**15.6**
g_1	10.7	15.1	**5.86**	12.6	14.9	**6.11**	14.5	17	**9.67**

Table 3: Svanberg problem: 340 Simulation points. RMS error of a set of 1000 LHS checkpoints as a percentage of the mean value for two types of experimental designs. NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network. Best fits are shown in bold

	Maximum error (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
f	47.6	63.8	**37.2**	64.1	77.9	**43.5**	65.2	137	**58.4**
g_1	48.8	54	**20.1**	44.6	52.7	**26.5**	52.9	71.3	**41.7**

Table 4: Svanberg problem: 340 Simulation points. Maximum error of a set of 1000 LHS checkpoints as a percentage of the mean value for two types of experimental designs. NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network

In this example, the RBF's perform better than the FFNN's. The example also demonstrates the considerably better performance achieved when using committees, especially for subdomain updating.

Vehicle crash and vibration (7 variables)

The crashworthiness simulation considers a model containing approximately 30,000 elements of a National Highway Transportation and Safety Association (NHTSA) Ford Taurus vehicle [17] 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 3 for the deformed (time = 90ms) state, and only the structural components affected by the design variables in Figure 4. The NVH model used to compute the first torsion vibrational mode is not shown here. 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 of the vehicle parameterized directly in the LS-DYNA [16] input file. Twelve parts are affected, comprising aprons, rails, shotguns, cradle rails and the cradle cross member (Figure 4). LS-DYNA Version 971 [16] is used for both the crash and NVH simulations, in explicit and implicit modes respectively.

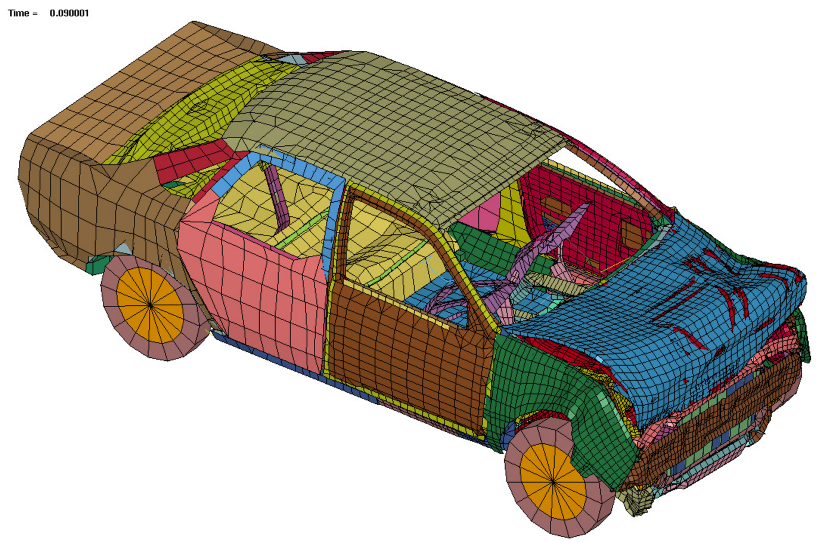

Figure 3: Deformed crash model of Taurus (90ms)

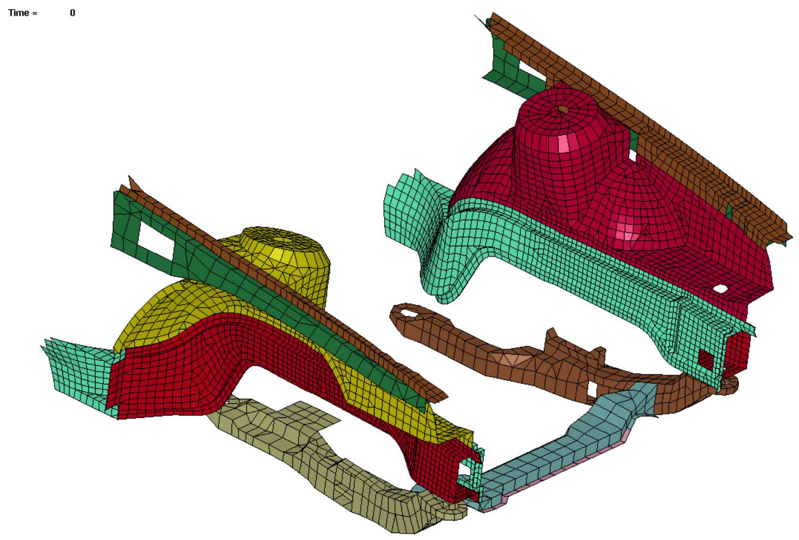

Figure 4: Parts containing thickness design variables

The response functions are defined as follows:

Maximum intrusion(***x***)
Stage 1 pulse(***x***)
Stage 2 pulse(***x***)
Stage 3 pulse(***x***)
Torsional mode frequency(***x***)

with **x** = [*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 *a* and displacement of a left rear sill node in the following fashion:

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

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. The Stage 1 pulse is represented by a triangle with the peak value being the value used.

Table 5 and Table 6 represent the RMS errors and Maximum errors respectively of the checkpoints. 300 Checkpoints are used on the metamodels built using 130 training points.

	RMSE (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
Intrusion	**0.680**	0.735	0.729	**0.796**	0.890	0.902	1.14	1.7	**0.963**
Pulse (Stage 1)	**0.637**	0.691	0.704	**0.695**	0.756	0.758	**1.52**	1.62	1.60
Pulse (Stage 2)	**1.31**	1.46	1.44	**1.38**	1.82	1.40	2.33	4.52	**2.21**
Pulse (Stage 3)	1.94	2.26	**1.90**	**2.04**	2.32	2.26	3.43	3.54	**2.5**
Frequency (twisting mode)	0.202	0.200	**0.188**	**0.179**	0.244	0.236	0.521	0.552	**0.461**

Table 5: Taurus MDO problem: 130 Simulation points. RMS error of a set of 300 LHS checkpoints as a percentage of the mean value for two types of experimental designs. NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network. Best fits are shown in bold.

	Maximum error (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
Intrusion	2.34	2.57	**2.12**	**2.59**	2.87	2.95	3.18	6.33	**2.86**
Pulse (Stage 1)	**1.67**	1.78	2.66	3.72	3.96	**2.43**	4.99	**4.52**	4.75
Pulse (Stage 2)	**4.02**	5.71	5.39	**3.97**	6.47	4.16	8.59	18.8	**7.1**
Pulse (Stage 3)	6.44	7.95	**5.93**	**5.78**	6.61	6.34	9.93	12.7	**6.78**
Frequency (twisting mode)	0.905	0.861	**0.755**	**0.943**	1.03	0.981	1.76	1.66	**1.61**

Table 6: Taurus MDO problem: 130 Simulation points. Maximum error of a set of 300 LHS checkpoints as a percentage of the mean value for two types of experimental designs. NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network. Best fits are shown in bold.

For this crash problem RBF networks generally perform the best, except for the sequential method where NN-9 is marginally better.

Knee impact example (11 variables)

Figure 5 shows the finite element model of a typical automotive instrument panel (IP) [19]. For model simplification and reduced per-iteration computational times, only the driver's side of the IP is used in the analysis as shown, and consists of around 25,000 shell elements. Also shown in Figure 5 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 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 design variables are shown in Figure 6. 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.

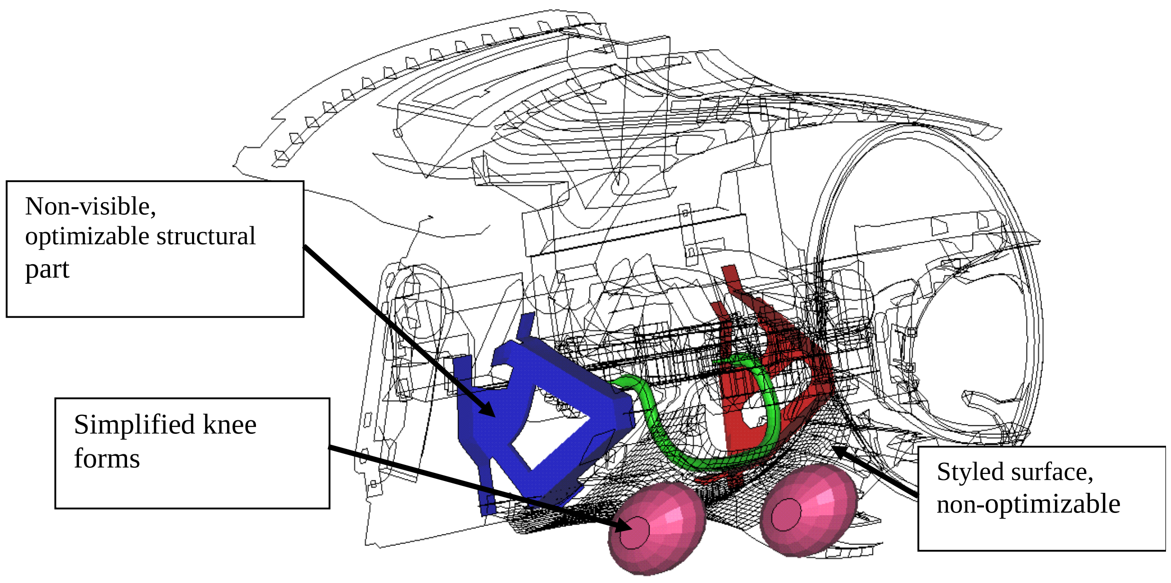
Non-visible, optimizable structural part
Simplified knee forms
Styled surface, non-optimizable

Figure 5: Instrument panel with knee bolster system (highlighted)

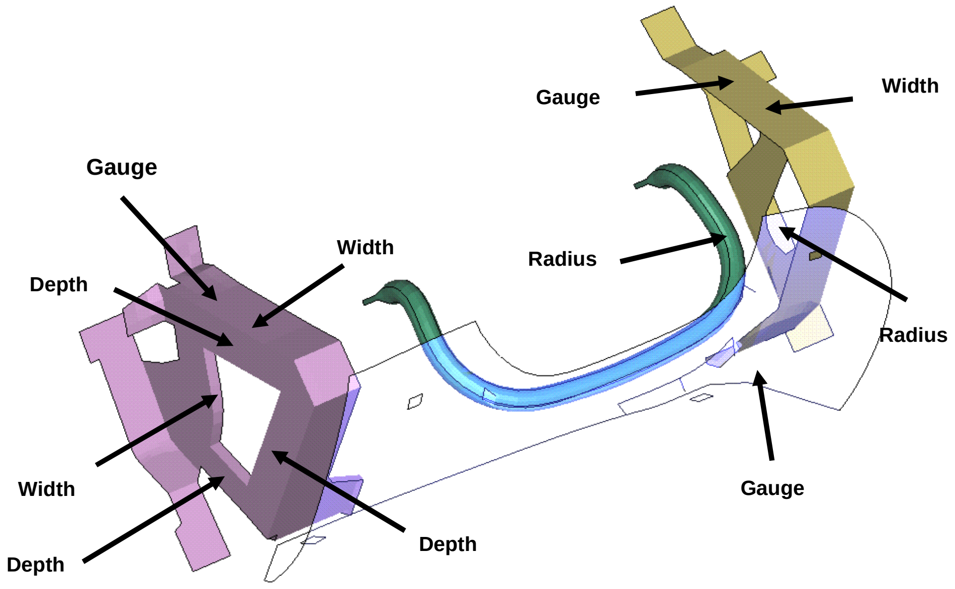

Figure 6: Eleven design variables of the knee bolster system

Table 7 represents the RMSE while Table 8 represents the Maximum checkpoint errors of 300 checkpoints.

	RMSE (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
Left knee force	**6.15**	7.02	6.94	**5.56**	6.67	7.02	10.1	47.5	**9.11**
Right knee force	3.33	**3.09**	3.92	**2.94**	3.21	3.74	6.92	26.6	**5.17**
Left Knee intrusion	**2.3**	2.38	2.85	**2.14**	2.18	2.58	**3.88**	33.7	4.48
Right Knee intrusion	**2.05**	2.25	2.94	**2.22**	2.29	2.81	**3.61**	39.9	5.43
Yoke displacement	36.4	54.7	**29.7**	**26.4**	27.6	32.2	64.0	142	**35.7**
Kinetic Energy	18.0	21.7	**12.1**	**12.0**	15.0	12.4	14.9	21.7	**13.4**

Table 7: Knee impact problem: 190 Simulation points. RMSE of a set of 300 LHS checkpoints as a percentage of the mean value for two types of experimental designs. NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network. Best fits are shown in bold.

	Maximum error (% of mean)								
Sampling:	Single Stage			Sequential			Sequential Subdomain		
Metamodel	NN-9	NN-1	RBF	NN-9	NN-1	RBF	NN-9	NN-1	RBF
Left knee force	**19.5**	24.5	29.4	**20.6**	22.0	24.0	31.1	105	**26.4**
Right knee force	**9.93**	10.4	12.4	**9.77**	10.7	10.3	29.6	46.7	**18**
Left Knee intrusion	9.9	**8.49**	10.8	7.09	**6.44**	8.6	**12.5**	87.7	14.1
Right Knee intrusion	**8.15**	8.13	11.4	**8.39**	8.8	10.5	**11.0**	63.7	15.8
Yoke displacement	199	388	**115**	**114**	147	120	403	206	**121**
Kinetic Energy	143	234	**47.1**	**40.9**	64.8	50.2	53.5	79.3	**53.4**

Table 8: Knee impact problem: 190 Simulation points. Maximum error of a set of 300 LHS checkpoints as a percentage of the mean value for two types of experimental designs. NN-9: Feedforward Neural Net with 9 member committee, NN-1: Feedforward Neural Net with 1 member, RBF: Radial Basis Function Network. Best fits are shown in bold.

An obvious result is that the single member NN (NN-1) performs relatively poorly on the subdomain approach whereas the RBF and NN-9 perform consistently well. RBF does particularly well for the Subdomain method.

Conclusions

Except for the fact that single stage global sampling will – as expected – always give a more accurate metamodel, the comparative strengths of the metamodels are otherwise somewhat problem dependent.

The NN-9 metamodel performs consistently well for all the examples and sampling strategies but is also the most expensive metamodel, having to rely on a non-linear regression algorithm for solution. Although it performs mostly better than NN-1, the single member model, the difference does not seem to be very large. Since the cost of building NN-9 is about 9 times that of building NN-1, the centering operation is very robust, but may not be justified. A glaring exception to this observation was the knee impact problem with subdomain reduction in which most of the responses deteriorated significantly when relying on a single net.

For the sequential methods, both the RBF and NN-9 do well for the crash problems, but, exceptionally, RBF is significantly worse for the 2-Bar truss problem. Since the 2-Bar truss is a highly nonlinear problem, involving reciprocal functions, RBF methods may perform especially poorly on such problems when using an iterative scheme with subdomain updating.

For the examples considered, there is no obvious degradation of accuracy for subdomain updating as a function of the number of variables. This is in spite of the fact that the reduction factor for each variable remains the same (about 0.25) irrespective of the number of design variables, thereby contributing to a greatly reduced volume fraction populated by new points for each new iteration. E.g the 21 variable Svanberg problem is less sensitive to the experimental design than the 3-variable 2-bar truss example. This may, of course, depend on the nature of the problem.

Due to their linear property, the radial basis functions have a major speed advantage over Neural Networks, even when the latter method uses only one committee member. It is however of some concern that, for highly nonlinear (but smooth) functions, the RBF networks may perform rather poorly, affecting the error measures by almost an order of magnitude. This result is aggravated by the use of subdomain updating. Further investigation is needed regarding this issue.

References

[1] Myers and Montgomery. *Response Surface Methodology – Process and Product optimization Using Designed Experiments*, Wiley, 1995.

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

[3] Papadrakakis, M. Lagaros, M. and Tsompanakis, Y. Structural optimization using Evolution Strategies and Neural Networks, *Computer Methods in Applied Mechanics and Engineering*, 156(1-4), pp. 309-333, 1998.

[4] Dyn, N., Levin, D. and Rippa, S. Numerical procedures for surface fitting of scattered data by Radial Basis Functions, *SIAM Journal of Scientifica and Statistical Computing*, 7(2), pp. 639-659, 1986.

[5] Fang, H. and Horstemeyer, M.F. Global response approximation with Radial Basis Functions, *Journal of Engineering Optimization*, 38(4), pp. 407-424, 2006.

[6] 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.

[7] Jin, R., Chen, W. and Simpson, T. Comparative studies of metamodeling techniques under multiple modeling criteria, *Journal of Structural Optimization,* 23(1), pp 1-13, 2001.

[8] 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.

[9] Johnson, M., Moore, L. and Ylvisaker, D. "Minimax and maximin distance designs," *Journal of statistical planning and inference,* Vol. 26, pp. 131-148.

[10] 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.

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

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

[13] 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.

[14] Stander, N., Roux, W.J., Goel, T., Eggleston, T. and Craig, K.-J. *LS-OPT User's Manual*, Version 3.3, Livermore Software Technology Corporation, Livermore, CA, 2008.

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

[16] Hallquist, J.O. *LS-DYNA Users Manual Version 971*. Livermore Software Technology Corporation, Livermore, CA, 2007.

[17] Svanberg. K. A globally convergent version of MMA without line search, Proceedings of the First World Congress of Structural and Multidisciplinary Optimization, Goslar, Pergamon Press, 6-9, 1995

[18] National Crash Analysis Center (NCAC). Public Finite Element Model Archive, www.ncac.gwu.edu/archives/model/index.html 2001.

[19] Ackerman, A., Thyagarajan, R., Stander, N., Burger, M., Kuhn, R. and Rajic, H. Shape optimization for crashworthiness design using response surfaces. *Proceedings of the International Workshop on Multidisciplinary Optimization, Pretoria, South Africa, August 8-10, 2000*