ASME 2003 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
Chicago, Illinois, USA, September 2-6, 2003

DETC2003/DAC-48759

ANALYSIS OF SUPPORT VECTOR REGRESSION FOR APPROXIMATION OF COMPLEX ENGINEERING ANALYSES

Stella M. Clarke
Department of Industrial & Manufacturing Engineering
The Pennsylvania State University
University Park, PA 16802 USA

Jan H. Griebsch
Doctoral Candidate
Lehrstuhl für Effiziente Algorithmen
The Technical University of Munich

Timothy W. Simpson[1]
Departments of Mechanical & Nuclear and
Industrial & Manufacturing Engineering
The Pennsylvania State University
University Park, PA 16802 USA

ABSTRACT

A variety of metamodeling techniques have been developed in the past decade to reduce the computational expense of computer-based analysis and simulation codes. Metamodeling is the process of building a "model of a model" that provides a fast surrogate for a computationally expensive computer code. Common metamodeling techniques include response surface methodology, kriging, radial basis functions, and multivariate adaptive regression splines. In this paper, we present Support Vector Regression (SVR) as an alternative technique for approximating complex engineering analyses. The computationally efficient theory behind SVR is presented, and SVR approximations are compared against the aforementioned four metamodeling techniques using a testbed of 22 engineering analysis functions. SVR achieves more accurate and more robust function approximations than these four metamodeling techniques and shows great promise for future metamodeling applications.

Keywords: Support Vector Regression, Support Vector Machines, Kriging, Response Surfaces, Metamodels

1. INTRODUCTION

Much of today's engineering analysis requires running complex and computationally expensive analysis and simulation codes, such as finite element analysis and computational fluid dynamics models. Despite continuing increases in computer processor speeds and capabilities, the huge time and computational costs of running complex engineering codes maintains pace. A way to overcome this problem is to generate an approximation of the complex analysis code that describes the process accurately enough, but at a much lower cost. Such approximations are often called *"metamodels"* in that they provide a "model of the model" [1]. Mathematically, if the inputs to the actual computer analysis are supplied in vector **x**, and the outputs from the analysis in vector **y**, then the true computer analysis code evaluates:

$$y = f(x) \tag{1}$$

where $f(x)$ is a complex engineering analysis function. The computationally efficient metamodel approximation is:

$$\hat{y} = g(x) \tag{2}$$

such that:
$$y = \hat{y} + \varepsilon \tag{3}$$

where ε includes both approximation and random errors.

There currently exists a number of metamodeling techniques to approximate $f(\mathbf{x})$ with $g(\mathbf{x})$, such as polynomial response surface models, multivariate adaptive regression splines, radial basis functions, kriging, and neural networks, and a recent comparison of the first four of these metamodeling techniques can be found in Ref. [2]. All of these techniques are capable of function approximation. In particular, although neural networks are able to approximate very complex models well, they have the two disadvantages of (i) being a "black box" approach and (ii) having a computationally expensive training process [3]. "Black box" means that little can be seen and understood about the model, because an exact function is not generated, only a trained "box" that accepts inputs and returns outputs.

[1] Please direct all correspondences to tws8@psu.edu.

In this paper, Support Vector Regression (SVR) is investigated as an alternative technique for approximating complex, computationally expensive engineering analyses. SVR is a particular implementation of Support Vector Machines (SVM), a "principled and very powerful method that in the few years since its introduction has already outperformed most other systems in a wide variety of applications" [4]. In many applications, SVMs are known to produce equally good, if not better, results than neural networks, while being computationally cheaper and producing an actual mathematical function (i.e., no "black box"). Hearst [5] comments that "SV learning is based on some beautifully simple ideas and provides a clear intuition of what learning from examples is about" and that SVMs "can lead to high performances in practical applications." In addition, Takeuchi, et al. [6] state that "SVMs have been applied very successfully in the past to several traditional classification tasks such as text classification." Other successful applications of SVMs have included handwritten character and digit recognition, face detection, text categorization, and object detection in machine vision [7].

The SVM algorithm is a non-linear generalization of the *Generalized Portrait* algorithm developed in Russia in the sixties [8]. In its present form, the SVM was developed at AT&T Bell Laboratories by Vapnik and co-workers in the early nineties [9, 10]. Smola, et al. [8] acknowledge the success of SVMs since this time and also add that "in regression and time series prediction applications, excellent performances were soon obtained." The application of the support vector approach to regression retains much of the elegance of SVMs but adds the capability to approximate functions. Hence, in this paper we investigate the performance of SVR in comparison to four metamodeling techniques: kriging, response surfaces, radial basis functions, and multivariate adaptive regression splines.

The remainder of this paper is as follows. First, an overview of the four metamodeling techniques is given. In Section 3, Support Vector Regression is introduced, beginning with linear function approximations and later non-linear approximations. A simple one-dimensional function approximation example is presented for illustration purposes in Section 3.3. A more thorough analysis is documented in Section 4 wherein 22 different engineering analysis functions are approximated using all five metamodeling methods (including SVR). The approximation testbed is introduced, and the method for comparing the techniques is explained, followed by a discussion of the results. Section 5 contains closing remarks and outlines future research in using SVR for function approximation.

2. EXISTING METAMODELING TECHNIQUES

This section briefly overviews the four metamodeling techniques against which SVR is compared: (1) response surface methodology, (2) radial basis functions, (3) kriging, and (4) multivariate adaptive regression splines. These methods were chosen based on their current widespread use and because algorithms for each method were readily available.

2.1. Response Surface Methodology

Response surface methodology (RSM) approximates functions by using the least squares method on a series of points in the design variable space. Low-order polynomials are the most widely used response surface approximating functions [11]. Equation (4) is a first-order polynomial that can be used for approximating functions with little to no curvature. Equation (5) is a second-order polynomial that includes all two-factor interactions.

$$\hat{y} = b_0 + \sum_{i=1}^{k} b_i x_i \qquad (4)$$

$$\hat{y} = b_0 + \sum_{i=1}^{k} b_i x_i + \sum_{i=1}^{k} b_{ii} x_i^2 + \sum_{i<j}\sum b_{ij} x_i x_j \qquad (5)$$

The constants (b_0, b_i, b_{ii}, b_{ij}) are determined by least squares regression; more information can be found in Ref. [12].

2.2. Radial Basis Functions

Radial basis functions (RBF) attempt approximation by using a linear combination of radially symmetric functions. Mathematically, the model can be expressed as:

$$\hat{y} = \sum_i a_i \left\| \mathbf{X} - \mathbf{X_{0i}} \right\| \qquad (6)$$

where a_i is a real-valued weight and $\mathbf{X_{0i}}$ is the input vector. Radial basis functions have produced good approximations to arbitrary contours. For example, they have been successfully applied to electronic circuit simulation models [13] and the construction of metamodels for a desk lamp example [14].

2.3. Kriging

The kriging model postulates a combination of a known function and departures of the form:

$$y(x) = f(x) + Z(x) \qquad (7)$$

where $y(x)$ is the unknown function of interest, $f(x)$ is a known polynomial function, which is often taken as a constant, and $Z(x)$ is called the correlation function and is a realization of a stochastic process with mean zero and variance σ^2, and nonzero covariance. Flexibility in kriging is achieved through a variety of spatial correlation functions, but the Gaussian correlation function is most frequently used [15]. More information on constructing kriging models can be found in Ref. [16].

2.4. Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) is a non-parametric regression procedure that makes no assumption about the underlying functional relationship between the dependent and independent variables. Instead, MARS constructs this relation from a set of coefficients and basis functions that are determined from regression data. The input space is divided into regions containing their own regression equation; thus, MARS is suitable for problems with high input dimensions, where the curse of dimensionality would likely create problems for other techniques. More information on MARS can be found in Ref. [17].

3. SUPPORT VECTOR REGRESSION

3.1. Linear Regression using SVR

There are two basic aims in SVR. The first is to find a function $f(\mathbf{x})$ that has at most ε deviation from each of the targets of the training inputs. For the linear case, f is:

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + \mathbf{b} \tag{8}$$

where $\langle \mathbf{a} \cdot \mathbf{b} \rangle$ is the dot product between **a** and **b**.

At the same time we would like this function to be as *flat* as possible. Smola, et al. [18] showed that choosing the flattest function in the feature space leads to a smooth function in the input space. Flatness in this sense means a small **w** in Eq. (8). This second aim is not as immediately intuitive as the first but is nevertheless important in the formulation of the optimization problem used to construct the SVR approximation:

$$\text{Minimize} \quad \frac{1}{2}|\mathbf{w}|^2 \tag{9}$$
$$\text{subject to} \quad \begin{cases} y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b \le \varepsilon \\ \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b - y_i \le \varepsilon \end{cases}$$

A key assumption in this formulation is that there exists a function *f(x)* that can approximate all input pairs *($\mathbf{x}_i$, y_i)* with ε precision; however, this may not be the case or perhaps some error allowance is desired. Thus slack variables ξ_i and ξ_i^* can be incorporated into the optimization problem to yield the following formulation [19]:

$$\text{Minimize} \quad \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{\ell}\left(\xi_i + \xi_i^*\right) \tag{10}$$
$$\text{subject to} \quad \begin{cases} y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b \le \varepsilon + \xi_i \\ \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

where the constant C > 0 determines the tradeoff between flatness (small *w*) and the degree to which deviations larger than ξ are tolerated (see Fig. 1), and ℓ is the number of samples.

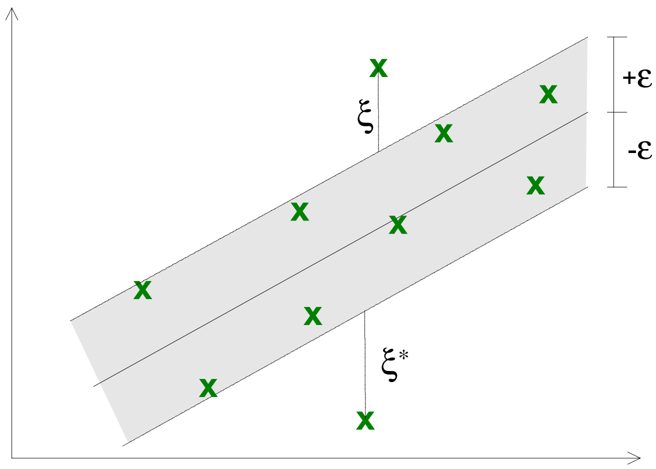


Figure 1 - Accounting for Slack Variables

The optimization function and linear constraints in Eq. (10) can be written as the Lagrangian function:

$$\begin{aligned} L = {} & \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{\ell}\left(\xi_i + \xi_i^*\right) \\ & - \sum_{i=1}^{\ell}\alpha_i\left(\varepsilon + \xi_i - y_i + \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b\right) \\ & - \sum_{i=1}^{\ell}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b\right) \\ & - \sum_{i=1}^{\ell}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right) \end{aligned} \tag{11}$$

Through Lagrangian Theory, necessary conditions for α to be a solution to the original optimization problem are:

$$\partial_b L = \sum_{i=1}^{\ell}\left(\alpha_i^* - \alpha_i\right) = 0 \tag{12}$$

$$\partial_w L = \mathbf{w} - \sum_{i=1}^{\ell}\left(\alpha_i^* - \alpha_i\right)\mathbf{x_i} = 0 \tag{13}$$

$$\partial_{\xi_i} L = C - \alpha_i - \eta_i = 0 \tag{14}$$

$$\partial_{\xi_i^*} L = C - \alpha_i^* - \eta_i^* = 0 \tag{15}$$

Substituting Eqs. (12-15) into Eq. (11) yields the optimization problem in dual form:

$$\text{Maximize} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)\langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle \\ -\varepsilon\sum_{i=1}^{\ell}\left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{\ell}y_i\left(\alpha_i - \alpha_i^*\right) \end{cases} \tag{16}$$
$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{\ell}\left(\alpha_i - \alpha_i^*\right) = 0 \\ \left(\alpha_i - \alpha_i^*\right) \in [0, C] \end{cases}$$

From Eq. (13),

$$\mathbf{w} = \sum_{i=1}^{\ell}\left(\alpha_i^* - \alpha_i\right)\mathbf{x_i} \tag{17}$$

and so the linear regression in Eq. (8) becomes:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell}\left(\alpha_i - \alpha_i^*\right)\langle \mathbf{x_i} \cdot \mathbf{x} \rangle + b \tag{18}$$

Thus the training algorithm and the regression function *f*(**x**) can be expressed in terms of the dot product $\langle \mathbf{x}_i \cdot \mathbf{x} \rangle$.

Transforming the optimization problem into dual form yields two advantages. First, the optimization problem is now a quadratic programming problem with linear constraints and a positive definite Hessian matrix, ensuring a unique global optimum. In addition, highly efficient and thoroughly tested quadratic solvers exist. Second, the input data only appears in the dot product, and regardless of the dimension of the input vector this dot product always yields a matrix. Effectively, *the computational complexity of high dimensional spaces is hidden from the optimization problem and the regression function.*

3.2. Nonlinear Regression using SVR

Another benefit of the dual form is that nonlinear function approximations can be achieved by replacing the dot product of input vectors with a nonlinear transformation on the input vectors. This transformation is referred to as the *kernel function* and is represented by $k(\mathbf{x},\mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are each input vectors. Table 1 lists common kernel functions [20]. Importantly, the kernel function substitution maintains the elegance of the optimization method for linear SVR.

Table 1 - Common Kernel Functions [20]

Linear	$k(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$
Polynomial	$k(\mathbf{x},\mathbf{x}') = \langle \mathbf{x} \cdot \mathbf{x}' \rangle^{d}$
Gaussian	$k(\mathbf{x},\mathbf{x}') = \exp\left(-\dfrac{\lVert \mathbf{x}-\mathbf{x}' \rVert^2}{2\sigma^2} \right)$
Sigmoid	$k(\mathbf{x},\mathbf{x}') = \tanh\left(\kappa \langle \mathbf{x} \cdot \mathbf{x}' \rangle + \vartheta\right)$
Inhomogeneous Polynomial	$k(\mathbf{x},\mathbf{x}') = \left(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + c \right)^{d}$

Applying the kernel function to the dot product of input vectors, we obtain:

$$\text{Maximize} \quad \begin{cases} -\dfrac{1}{2}\displaystyle\sum_{i,j=1}^{\ell}\left(\alpha_i - \alpha_i^{*}\right)\left(\alpha_j - \alpha_j^{*}\right)k\left(\mathbf{x_i},\mathbf{x_j}\right) \\ -\varepsilon\displaystyle\sum_{i=1}^{\ell}\left(\alpha_i + \alpha_i^{*}\right) + \sum_{i=1}^{\ell} y_i\left(\alpha_i - \alpha_i^{*}\right) \end{cases} \tag{19}$$

$$\text{Subject to} \quad \begin{cases} \displaystyle\sum_{i=1}^{\ell}\left(\alpha_i - \alpha_i^{*}\right) = 0 \\ \alpha_i, \alpha_i^{*} \in [0, C] \end{cases}$$

Replacing the dot product in Eq. (18), the SVR approximation becomes:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell}\left(\alpha_i - \alpha_i^{*}\right)k\left(\mathbf{x_i},\mathbf{x}\right) + b \tag{20}$$

The kernel function $k(\mathbf{x_i},\mathbf{x})$ can be precomputed, and the results are stored in the *kernel matrix*, K= $\left(k\left(\mathbf{x_i},\mathbf{x_j}\right)\right)_{i,j=1}^{n}$.

In order to guarantee a unique optimal solution to the quadratic optimization problem, this kernel matrix must be positive definite. The kernel functions presented in Table 1 yield positive definite kernel matrices [4]. Thus by using the kernel matrix, nonlinear function approximations can be achieved while maintaining the simplicity and computational efficiency of linear SVR approximations.

Now that the theory behind SVR has been presented, we can begin to investigates the application of SVR to engineering analyses. First, a simple one-dimensional example is presented next for illustration purposes, and then a testbed of more realistic engineering problems is discussed in Section 4.

3.3. A Simple One-Dimensional Example

This section illustrates the application of SVR to a one-dimensional example, which means that the length of each input vector is only one. The function to be approximated comes from Su and Renaud [21] and is shown in Fig. 2. The five training points noted in the figure are used to fit the SVR.

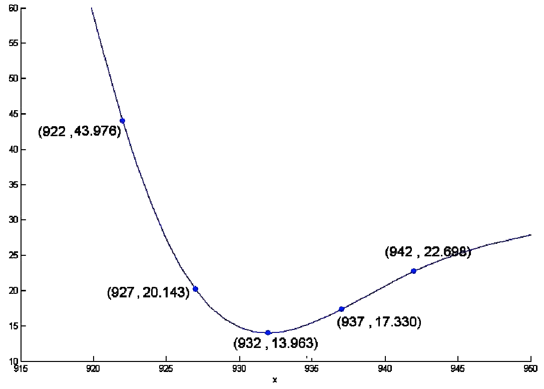


Figure 2 – Eighth-Order Test Function [21]

The exact eighth-order function is given by:

$$f(x) = \sum_{i=1}^{9} a_i (x_i - 900)^{(i-1)} \tag{21}$$

where:

a_1 = -659.23, a_2 = 190.22, a_3 = -17.802, a_4 = 0.82691, a_5 = -0.021885, a_6 = 0.0003463, a_7 = -3.2446 x 10^{-6}, a_8 = 1.6606 x 10^{-8}, a_9 = -3.5757 x 10^{-11}

The Matlab code developed by Steve Gunn was used to execute the SVR algorithm [22]. Figure 3 shows a flowchart of the implementation, which is discussed as follows.

As seen in Fig. 3, the kernel matrix is first calculated from the training points. Using the Gaussian kernel function presented in Table 1 for each combination of input vectors, the following kernel matrix is obtained:

$$K = \begin{bmatrix} 1.0000 & 0.8749 & 0.5858 & 0.3003 & 0.1178 \\ 0.8749 & 1.0000 & 0.8749 & 0.5858 & 0.3003 \\ 0.5858 & 0.8749 & 1.0000 & 0.8749 & 0.5858 \\ 0.3003 & 0.5858 & 0.8749 & 1.0000 & 0.8749 \\ 0.1178 & 0.3003 & 0.5858 & 0.8749 & 1.0000 \end{bmatrix}$$

This matrix is used in the quadratic optimization problem stated in Eq. (19). The resulting solution yields all the variables required to calculate the approximating function. The vector of differences of Lagrange multipliers is:

$$\alpha_i^{*} - \alpha_i = \begin{Bmatrix} 433.8411 \\ -924.0233 \\ 999.9999 \\ -647.1798 \\ 229.4055 \end{Bmatrix}$$

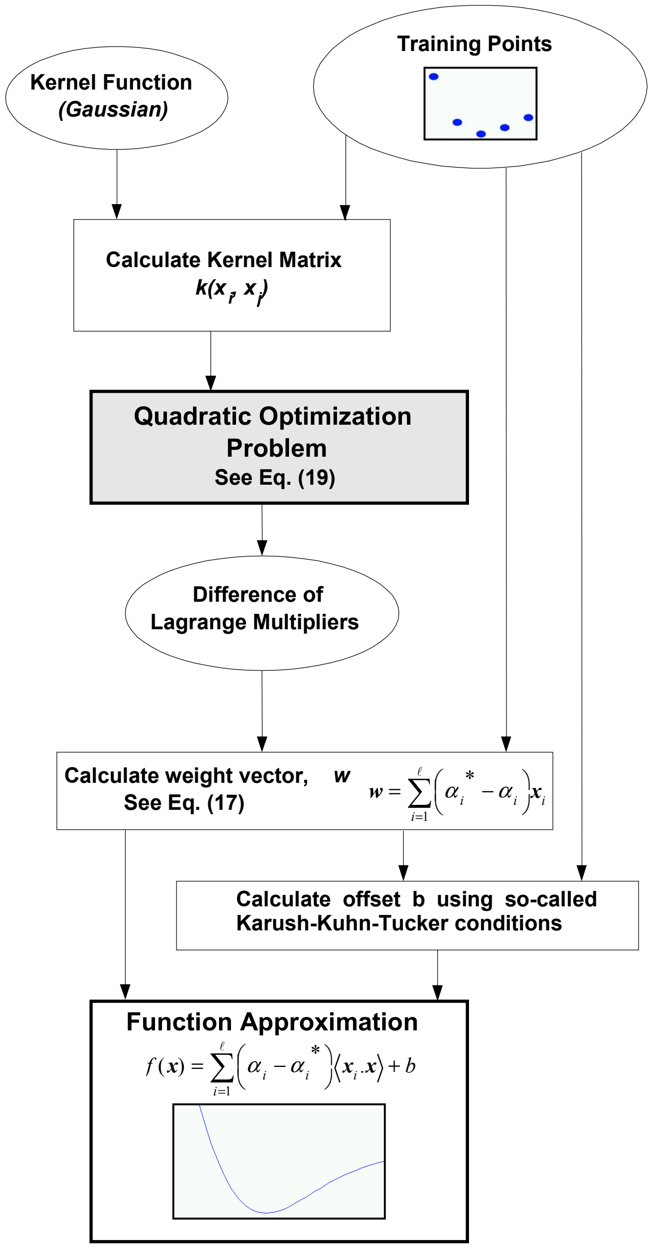

Kernel Function (Gaussian)
Training Points
Calculate Kernel Matrix $k(x_i, x_j)$
Quadratic Optimization Problem See Eq. (19)
Difference of Lagrange Multipliers
Calculate weight vector, w See Eq. (17) $w=\sum_{i=1}^{\ell}(\alpha_i^*-\alpha_i)x_i$
Calculate offset b using so-called Karush-Kuhn-Tucker conditions
Function Approximation $f(x)=\sum_{i=1}^{\ell}(\alpha_i-\alpha_i^*)\langle x_i.x\rangle+b$

Figure 3 - Flowchart of the SVR Algorithm

The difference of Lagrange multipliers is then used with the training points to calculate the weight vector, $\mathbf{w}=\sum_{i=1}^{\ell}\left(\alpha_i^*-\alpha_i\right)\mathbf{x_i}$, and the offset, b. The offset b is calculated using Karush-Kuhn-Tucker conditions (see Ref. [23] for more details), and all of these variables are then substituted into Eq. (20) to yield the SVR approximation, $f(\mathbf{x})$.

The radius of the Gaussian kernel was manually optimized to a value of 9.7 for the given training data. This involved continually updating the Gaussian radius and running the SVR algorithm until the root mean square error (RMSE) was close to minimum. Figure 4 shows the SVR function approximation compared to the actual function. We can see a very close fit between the SVR approximation and the actual function within the range of the training data, but the fit starts to deteriorate outside of this range as one might expect.

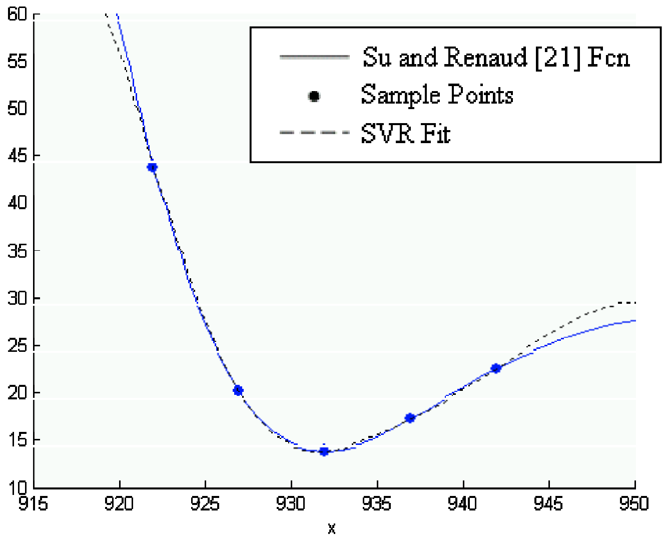


Figure 4 - Fit of 1D Function by SVR

To assess the accuracy of these results, the three error equations in Table 2 are calculated, where n_{error} is the number of random test points used, y_i is the actual function value and $\hat{y}_i$ is the predicted value from the function approximation method. For comparison, the response surface and kriging models constructed in Ref. [24] are included in this assessment. The resulting errors for SVR, second-order RS model and kriging are listed in Table 3, based on 16 evenly spaced test points between 920 and 945. The SVR method has achieved the lowest error measures in all three categories; hence, SVR provides a very accurate approximation for this 1D example.

Table 2 - Error Measures for Accuracy Assessment

Name	Equation
Max. Absolute Error (MAE)	max. $\lvert y_i-\hat{y}_i\rvert$, i = 1, ..., n_{error}
Average Absolute Error (AAE)	$\frac{1}{n_{error}}\sum_{i=1}^{n_{error}}\lvert y_i-\hat{y}_i\rvert$
Root Mean Square Error (RMSE)	$\sqrt{\frac{\sum_{i=1}^{n_{error}}(y_i-\hat{y}_i)^2}{n_{error}}}$

Table 3 - Error Comparisons between Approximation Methods for 1D Example

Error	SVR	RSM	Kriging
MAE	2.044	3.134	2.507
AAE	0.4356	1.911	0.776
RMSE	0.8266	2.155	1.004

4. APPROXIMATION TESTBED AND COMPARISON

The testbed of engineering analyses used to benchmark SVR against other approximation methods is listed in the Appendix. This testbed was initially created in Ref. [24] to compare the predictive capability of kriging models. The 22 functions to be approximated are derived from five engineering problems typically used to test optimization algorithms: a two-bar truss [25], a three-bar truss [25], a two-member frame [26], a helical compression spring [27], and a welded beam [28].

4.1. Training and Testing Procedure

The procedure used to train and test each approximation technique is summarized as follows:

- Step 1 - Generate training data and test data
- Step 2 - Construct approximations using training data
- Step 3 - Compare the accuracy and robustness of each approximation

Step 1 - Generate training data and test data: To minimize interactions between metamodel type and the choice of experimental design, six different sets of training data were generated for each function using six different experimental designs as listed in Table 4. A review and detailed comparison of these experimental designs can be found in Ref. [29]. The number of training points generated for each problem (n_1) is also shown. Since there are six sets of training data for each of the 22 functions, a total of 6x22=132 function approximations are constructed using *each* approximation technique.

Table 4 – Generation of Training Data and Test Data

Experimental Design Used to Generate Training Data	# Training Points (n_1)		
	2 Variable Problems	3 Variable Problems	4 Variable Problems
Central Composite Design [12]	13	15	25
Inscribed CCD [12]	13	15	25
Hammersley Sampling [30]	13	15	25
Latin Hypercube Design [31]	13	15	25
Orthogonal Latin Hypercube [32]	13	15	25
Uniform Design [33]	13	15	25
# Test Points	1000	1500	2000

Additional test data is generated as shown in Table 4 using a Latin Hypercube Design. The number of test points is very large to ensure a thorough analysis of the accuracy of the resulting approximation throughout the performance space.

Step 2 - Construct approximations using training data: Every training data set is submitted to each metamodeling algorithm to generate a function approximation. For SVR, Steve Gunn's Support Vector Machine Matlab code [22] is implemented as in the one-dimensional example (see Fig. 3). For kriging, RSM, RBF and MARS, Fortran algorithms from Ref. [24] were utilized. Thus, five algorithms were employed to test each of the five approximation techniques.

Step 3 - Compare the accuracy and robustness of each approximation: Additional test data is used to test the accuracy of the function approximation generated by each of the five metamodeling techniques. For a given function and a given metamodel, the inputs of the test data are submitted to the function approximation generated in Step 2. The outputs are the predicted values of the original function according to the corresponding metamodeling technique. The difference between each predicted value, $\hat{y}_i$, and the actual function value, y_i, is calculated as the error for that test point.

As in the one-dimensional example, three error measures are calculated to assess accuracy: (1) RMSE, (2) MAE, and (3) AAE, see Table 2. The RMSE gives an indication of how accurate the approximation was overall, while the maximum error can reveal the presence of regional areas of poor approximation; however, due to the different magnitudes of the 22 test functions, these error statistics enable comparisons of algorithms only within each function. It is desirable to compare the effectiveness of each algorithm across *all* of the functions so that conclusions can be drawn. A common normalized statistic is the correlation coefficient (R-squared); however, it is only suited for linear approximations. Although correlation coefficients have been devised for non-linear approximations (see Ref. [34]), they apply only to simple functions and not the complex equations in the testbed. Thus to enable comparisons of SVR to the four other metamodeling techniques, the average percentage differences in error values between each algorithm and SVR is computed. In this manner, the error in the SVR approximations becomes the benchmark to which all other metamodels are compared. Positive percentage values imply that the corresponding approximation had larger errors than SVR, while negative values imply that the approximation had smaller errors.

The **robustness** of an approximation method is indicated by the variance between its error values across different sample sets [2]. The capability to continually repeat a function approximation with similar accuracies (i.e., similar errors) increases the reliability of the results and the robustness of the approximation method. To test for robustness, SVR is again used as a benchmark to examine the robustness of each approximation technique, and the standard deviation is computed to indicate robustness using the following steps:

1. For a given approximation technique, find the standard deviation of each error (i.e., RMSE, MAE, AAE) across the six different training sets for each function.
2. Normalize each standard deviation against the standard deviation for SVR for the same function. This normalized standard deviation (Norm_Std_Dev) is calculated as:

$$\text{Norm\_Std\_Dev} = \frac{\text{Std. Dev. for Given Technique} - \text{Std. Dev. for SVR}}{\text{Std. Dev. for SVR}}$$

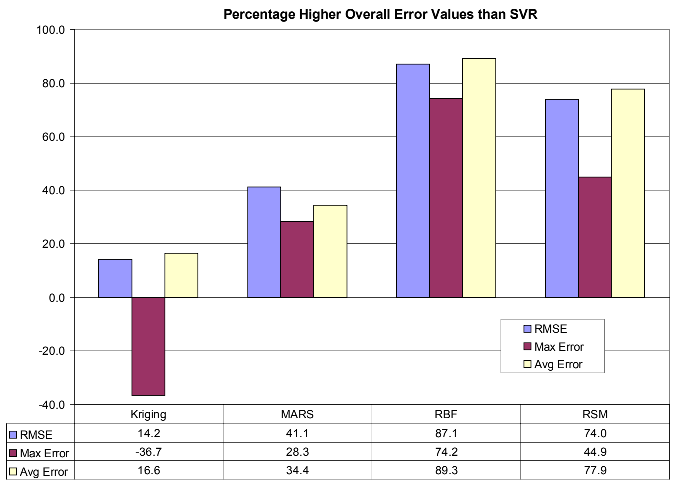
3. Average Norm_Std_Dev across all functions for a given metamodel.

The normalized standard deviation reflects the variance in the error for a given approximation technique, relative to SVR. A positive value indicates a greater variance than SVR and hence lower robustness, while a negative value indicates a lower variance than SVR and a more robust approximation method.

4.2. ANALYSIS OF RESULTS

4.2.1. Accuracy Results

Figure 5 shows the percentages by which average error values were higher than the corresponding error value for SVR. These percentage differences have been averaged over all 22 approximated functions. Except for the overall maximum error for kriging, SVR has achieved lower average error values (i.e., RMSE, average error, and maximum error) compared to the four other approximation techniques. As previously mentioned, lower overall RMSE values represent good global function approximation across the performance space, while lower overall maximum error values reflect the absence of poorly approximated regions in the performance space. Thus the results imply that kriging models avoid areas of poor approximation, but they do not perform as well globally as the SVR approximations. In general, SVR has outperformed the four other approximation techniques, giving lower overall error values. Kriging achieved the next best overall performance, followed by MARS, RSM, and finally RBF. These trends are consistent with those observed in Ref. [24], but we note that RBF has performed much better in other studies [2,14]. We are still investigating why this has occurred.

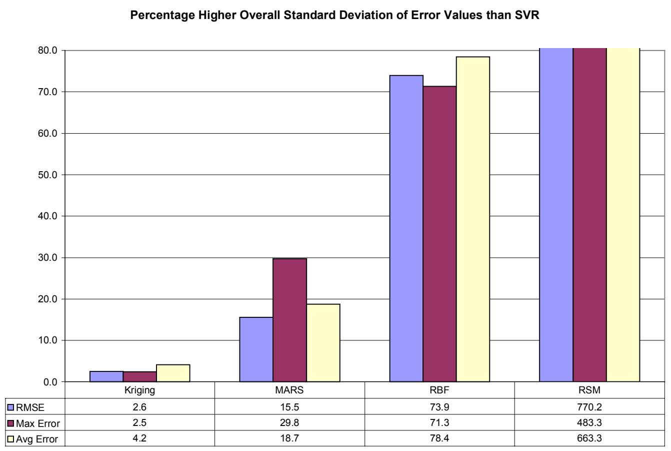


	Kriging	MARS	RBF	RSM
RMSE	14.2	41.1	87.1	74.0
Max Error	-36.7	28.3	74.2	44.9
Avg Error	16.6	34.4	89.3	77.9

Figure 5 - Comparisons of Errors Between Metamodeling Techniques

4.2. Robustness Results

Figure 6 presents the normalized standard deviations of the errors for each approximation technique, relative to SVR. All of the relative standard deviations have positive values, indicating large variance between error values for each technique, relative to SVR. Hence, the results indicate that SVR is the most robust of the five approximation techniques, reinforcing the validity of the error values obtained in Figure 5. Kriging is the second most robust method. The variance between errors for RSM is very large, when measured relative to SVR, indicating very inconsistent performance. This could be reflective of the fact that RSM is most suitable for linear approximations only, performing well in these cases and poorly in nonlinear cases. The relatively high robustness of SVR was followed by kriging, MARS, RBF, and finally RSM.



	Kriging	MARS	RBF	RSM
RMSE	2.6	15.5	73.9	770.2
Max Error	2.5	29.8	71.3	483.3
Avg Error	4.2	18.7	78.4	663.3

Figure 6 - Comparisons of Standard Deviations of Errors between Metamodels

5. CONCLUSIONS AND FUTURE WORK

In conclusion, the theory behind SVR has been presented and shown to possess the desirable qualities of mathematical and algorithmic elegance and producing an actual approximating function as opposed to a trained "black box". In comparison to four common approximating techniques, SVR had the best overall performance for the testbed of 22 engineering analysis functions. Only kriging out-performed SVR in the category of average lowest maximum error. The strong performance of the SVR approximations was reinforced through relatively small variances between error values, indicating that SVR also yields a more robust approximation.

The SVR implementation employed produced successful results; however, better results using SVR are anticipated through increased attention to the SVR algorithm itself and the model parameters selected. The Matlab algorithm with which the SVR results were obtained is not as efficient as other available algorithms, but it could easily be manipulated. Other SVM and SVR algorithms exist, such as SVMlight and mySVM <http://www-ai.cs.uni-dortmund.de/>, which are available in C and are more efficient than the relatively slower Matlab implementation; however, users are limited in the amount of "tweeking" they can do. A better solution would be to code an entire SVR algorithm in C, employing a commercially available quadratic solver such as LOQO [35].

In addition, the SVR implemented in our experiments consistently used a Gaussian kernel function and a constant precision value, ε. Optimizing these parameters during training will improve results in most cases. For example, data suspected to be polynomial will probably be modeled more accurately with the use of a polynomial kernel function instead of a Gaussian one; however, the optimal choice for the kernel function is still an area of active research. Theoretically, any symmetric function that results in a positive definite kernel matrix can be used. The radius of the Gaussian kernel function was also manually optimized in this study for each function. Adopting a method to automatically optimize this radius (such as the simulated annealing algorithm in kriging [16]) could further improve the results.

ACKNOWLEDGMENTS

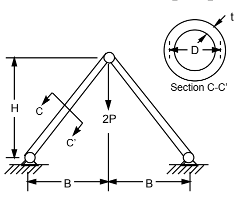
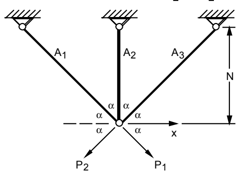
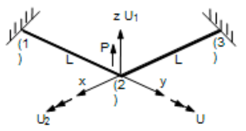
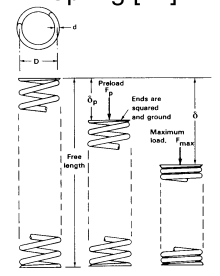
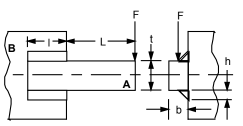
We gratefully acknowledge support from the U.S. Department of Transportation and the Federal Transit Administration. This work was conducted as part of the development of a web-based information management system for storing, displaying, and analyzing bus test data.

REFERENCES

[1] Kleijnen, J. P. C., 1987, *Statistical Tools for Simulation Practitioners*, Marcel Dekker, New York, NY.
[2] Jin, R., Chen, W., and Simpson, T. W., 2001, "Comparative Studies of Metamodeling Techniques under Multiple Modelling Criteria," *Journal of Structural Optimization*, **23**(1), pp. 1-13.
[3] Kinnebrook, W., 1994, *Neuronale Netze*. Oldenbourg Verlag, München.
[4] Cristianni, N., and Shawe-Taylor, J., 2000, *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*, Cambridge University Press, Cambridge, UK.
[5] Hearst, M.A., 1998, "Trends controversies: Support vector machines," *IEEE Intelligent System*, **13**(4), pp. 18-28.

[6] Takeuchi, K., Collier, N., 2002, “Use of Support Vector Machines in Extended Named Entity,” *CoNLL-2002*, Taipei, Taiwan.
[7] Dumais, S.T., Platt, J., Heckerman, D., Saharni, M., 1998, “Inductive Learning Algorithms and Representations for Text Categorization,” *Proceedings of ACM- CIKM98*, pp. 148-155.
[8] Smola, A. J., and Schölkopf, B., 1998, “A Tutorial on Support Vector Regression,” *NeuroCOLT2 Technical Report Series*, NC2-TR-1998-030, Berlin, Germany.
[9] Vapnik, V., and Lerner, A., 1963, “Pattern Recognition Using Generalized Portrait Method.” *Automation and Remote Control*, 24.
[10] Vapnik, V, and Chervonenkis, A., 1964, A Note On One Class of Perceptrons, *Automation and Remote Control*, 25.
[11] Simpson, T. W., Peplinski, J., Koch, P. N. and Allen, J. K., 2001, "Metamodels for Computer-Based Engineering Design: Survey and Recommendations," *Engineering with Computers,* **17**(2), pp. 129-150.
[12] Myers, R. H., Khuri, A. I., and Carter, W. H., 1995, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments,* John Wiley & Sons, New York, NY.
[13] Tu, C. H., and Barton, R. R., 1997, “Production Yield Estimation by the Metamodel Method with a Boundary–focused Experiment Design,” *Design Theory and Methodology Conference – DTM’97*, ASME, Paper No. DETC97/DTM3870.
[14] Meckesheimer, M., Barton, R. R., Simpson, T. W., Limayem, F. and Yannou, B., 2001, "Metamodeling of Combined Discrete/Continuous Responses," *AIAA Journal*, **39**(10), pp. 1955-1959.
[15] Sacks, J. Welch, W.J., Mitchell, T.J., Wynn, H.P., 1989, “Design and Analysis of Computer Experiments.” *Statistical Science*, **4**(4), pp. 409-435.
[16] Simpson, T. W., Mauery, T. M., Korte, J. J. and Mistree, F., 2001, "Kriging Metamodels for Global Approximation in Simulation-Based Multidisciplinary Design Optimization," *AIAA Journal*, **39**(12), pp. 2233-2241.
[17] Friedman, J.H., 1991, “Multivariate Adaptive Regression Splines,” *The Annals of Statistics*, **19**(1), pp. 1-141.
[18] Smola, A. J., Schölkopf, B., and Müller, K. R., 1998, “The Connection Between Regularization Operators and Support Vector Kernels,” *Neural Networks*, **11**, pp. 637-649.
[19] Vapnik, V., 1995, *The Nature of Statistical Learning Theory*, Springer, New York, NY.
[20] Schölkopf, B., Smola A.J., 2002, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA.
[21] Su, J. and Renaud, J. E., 1997, "Automatic Differentiation in Robust Optimization," *AIAA Journal*, **35**(6), pp. 1072-1079.
[22] Gunn, S.R., 1997, *Support Vector Machines for Classification and Regression*. Technical Report, Image Speech and Intelligent Systems Research Group, University of Southampton, UK.
[23] Markowetz, F., 2001, *Support Vector Machines in Bioinformatics*, Diploma Thesis in Mathematics, University of Heidelberg, Germany.
[24] Simpson, T. W., 1998, "A Concept Exploration Method for Product Family Design," *Ph.D. Dissertation,* G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA.
[25] Schmit, L. A., 1981, "Structural Synthesis—Its Genesis and Development," *AIAA Journal,* **19**(10), pp. 1249-1263.
[26] Arora, J. S., 1989, *Introduction to Optimum Design*, McGraw-Hill, New York, NY.
[27] Sandgren, E., 1990, "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization," *Journal of Mechanical Design,* **112**(2), pp. 223-229.
[28] Ragsdell, K. M. and Phillips, D. T., 1976, "Optimal Design of a Class of Welded Structures Using Geometric Programming," *Journal of Engineering for Industry,* Series B **98**(3), pp. 1021-1025.
[29] Simpson, T. W., Lin, D. K. J. and Chen, W., 2001, "Sampling Strategies for Computer Experiments: Design and Analysis," *International Journal of Reliability and Applications*, **2**(3), pp. 209-240.
[30] Kalagnanam, J. R. and Diwekar, U. M., 1997, "An Efficient Sampling Technique for Off-Line Quality Control," *Technometrics*, **39**(3), pp. 308-319.
[31] McKay, M. D., Beckman, R. J. and Conover, W. J., 1979, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, **21**(2), pp. 239-245.
[32] Owen, A. B., 1992, "Orthogonal Arrays for Computer Experiments, Integration and Visualization," *Statistica Sinica*, **2**, pp. 439-452.
[33] Fang, K.-T., Lin, D. K. J., Winker, P. and Zhang, Y., 2000, "Uniform Design: Theory and Application," *Technometrics*, **42**, pp. 237-248.
[34] Cameron, A.C. and Windmeijer, F.A.G., 1997, "An R-squared Measure of Goodness of Fit for some common Nonlinear Regression Models", *Journal of Econometrics*, **77**, pp. 329-342.
[35] Vanderbei, R. J., 1999, “LOQO user's manual (Version 3.10)”, *Optimization Methods and Software*, **12**, pp. 485-514.

APPENDIX: APPROXIMATION TESTBED FOR COMPARING METAMODELING TECHNIQUES

	Problem	Input (Design) Variables	Output (Response) Variables	Function to be Approximated
2 Variable Problems	2-bar Truss [25] 	D H	W	$W(\mathbf{x}) = 2\rho\pi DT(B^2 + H^2)^{1/2}$
			g_1	$g_1(\mathbf{x}) = \frac{\pi^2 E(D^2+T^2)}{8(B^2+H^2)} - \frac{P(B^2+H^2)^{1/2}}{\pi TDH}$
			g_2	$g_2(\mathbf{x}) = \sigma_y - \frac{P(B^2+H^2)^{1/2}}{\pi TDH}$
	3-bar Truss [25] 	A_1 A_2	W*	$W(\mathbf{x}) = \rho N(2\sqrt{2}A_1 + A_2)$
			g_1	$g_1(\mathbf{x}) = 20{,}000 - \left[\frac{1}{A_1} - \frac{A_2}{2A_1A_2+\sqrt{2}A_1^2}\right]$
			g_2	$g_2(\mathbf{x}) = 20{,}000 - \frac{20{,}000\sqrt{2}A_1}{2A_1A_2+\sqrt{2}A_1^2}$
			g_3	$g_1(\mathbf{x}) = 15{,}000 - \frac{20{,}000A_2}{2A_1A_2+\sqrt{2}A_1^2}$
3 Variable Problems	2-member Frame [26] 	d h t	V*	$V(\mathbf{x}) = 2L(2dt + 2ht - 4t^2)$
			g_1	$g_1(\mathbf{x}) = (\sigma_1^2 + 3\tau^2)^{1/2}$
			g_2	$g_2(\mathbf{x}) = (\sigma_2^2 + 3\tau^2)^{1/2}$
	Spring [27] 	N D d	V*	$V(\mathbf{x}) = \pi^2 Dd^2(N + 2)/4$
			g_1	$g_1(\mathbf{x}) = S - 8C_fF_{max}D/(\pi d^3)$
			g_2*	$g_2(\mathbf{x}) = l_{max} - l_f$
			g_3*	$g_3(\mathbf{x}) = \delta_{pm} - \delta$
			g_4*	$g_4(\mathbf{x}) = (F_{max} - F_{load})/K - \delta_w$
			g_5*	$g_5(\mathbf{x}) = D_{max} - D - d$
			g_6*	$g_6(\mathbf{x}) = C - 3$
4 Variable Problem	Welded Beam [28] 	h l t b	F*	$F(\mathbf{x}) = (1 + c_3)h^2l + c_4tb(L + l)$
			τ	$\tau(\mathbf{x}) = [(\tau')^2 + 2\tau'\tau''\cos\theta + (\tau'')^2]^{1/2}$
			σ	$\sigma(\mathbf{x}) = 6FL/(bt^2)$
			P	$P_c(\mathrm{x}) = \frac{4.013\sqrt{EI\alpha}}{L^2}\left[1 - (\frac{t}{2L})\sqrt{\frac{EI}{\alpha}}\right]$
			DEFL	$P_c(\mathrm{x}) = \frac{4.013\sqrt{EI\alpha}}{L^2}\left[1 - (\frac{t}{2L})\sqrt{\frac{EI}{\alpha}}\right]$