EngOpt 2008 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 01 - 05 June 2008.

Response Surface Methods Applied to Scarce and Small Sets of Training Points – A Comparative Study

COLAÇO[1], M. J., SILVA[1], W. B., MAGALHÃES[1], A. C., DULIKRAVICH[2], G. S.

[1]Military Institute of Engineering
Dept. of Mechanical and Materials Engineering
Pç. Gen.Tibúrcio,80, Rio de Janeiro, RJ, 22290-270, Brazil
colaco@ime.eb.br

[2]Florida International University
Dept. of Mechanical and Materials Engineering
10555 West Flagler St., EC 3474, Miami, FL, 33174, USA
dulikrav@fiu.edu

1. Abstract

In this paper we perform a comparison among several methods for generating response surface models. Response surfaces are often used to replace very complicated physical models, to generate correlations of experimental data and to be used in optimization problems in order to reduce the computational cost involved. Based on the extensive testing performed on 296 linear and non-linear analytical test functions, the accuracy, robustness, efficiency, transparency and conceptual simplicity of the different methods are discussed. The methods compared in this work are singular value decomposition (SVD), K-nearest, Kriging, parametric, Gaussian processes, neural networks, radial basis functions (RBF) and the fittest polynomial RBF method. All, except the basic RBF methods and the fittest polynomial RBF method were taken from the modeFRONTIER® software, which was kindly provided through the Esteco and ESSS companies. Conclusion of this thorough comparative analyses performed on scarce and small sized training data sets is that the fittest polynomial RBF method is the most accurate and the most robust overall method among the 15 response surface fitting methods analyzed.

2. Keywords: Response surfaces, Polynomial fit.

3. Introduction

In many optimization problems, evaluation of the objective function is extremely expensive and time consuming. For example, optimizing chemical concentrations of each of the alloying elements in a multi-component alloy requires manufacturing each candidate alloy and evaluating its properties using classical experimental techniques. Even with the most efficient optimization algorithms[1], this means that often hundreds of alloys having different chemical concentrations of their constitutive elements would have to be manufactured and tested. This is understandably too expensive to be economically acceptable.

Therefore, for problems where objective function evaluations are already expensive and where the number of design variables is large thus requiring many such objective functions evaluations, the only economically viable approach to optimization is to use a cheap and accurate surrogate model (a metamodel) instead of the actual high fidelity analysis method. Such low fidelity surrogate models are known as response surfaces which in case of more than three dimensions become high dimensional hypersurfaces that need to be fitted through the available often small original set of high fidelity values of the objective function. Once the response surface (hypersurface) it created using an appropriate analytic formulation, it is very easy and fast to search such a surface for its minima given a set of values of design variables supporting such a response surface.

From the viewpoint of kernel interpolation/approximation techniques, many response surface methods are based on linear and non-linear regression and other variants of least square technique. This group of mesh-free methods has been successfully applied to many practical, but difficult problems in engineering that are to be solved by the traditional mesh-based methods.

One of the most popular mesh-free kernel approximation techniques is radial basis functions (RBFs). Initially, RBFs were developed for multivariate data and function interpolation. It was found that RBFs were able to construct an interpolation scheme with favorable properties such as high efficiency, good quality and capability of dealing with scattered data, especially for higher dimension problems. A convincing comparison[2] of a RFB based response surface method and a wavelet based artificial neural network method[3] demonstrated superiority of RBF based methods especially for high dimensionality response surfaces.

In the present study, we compare several of RBF formulations to generate response surfaces. Besides the numerous RBF based formulation, we also compare the Singular Value Decomposition (SVD), the K-Nearest, the Kriging, the Parametric, Gaussian Processes, and the Neural Networks methods. These last methods, except the basic RBF methods (Table 1) and the fittest polynomial RBF method were taken from the modeFRONTIER® general purpose optimization software, which was kindly provided through the Esteco and ESSS companies. The objective is to compare the performance (accuracy of fitting the data provided) and computing time for all of these methods.

3.1. Radial Basis Functions

The use of RBFs followed by collocation, a technique first proposed by Kansa[4], after the work of Hardy[5] on multivariate approximation, is now becoming an established approach. Various applications to problems in mechanics have been made in recent years – see, for example Refs. [6] and [7].

Kansa's method (or asymmetric collocation) starts by building an approximation to the field of interest (normally displacement components) from the superposition of RBFs (globally or compactly supported) conveniently placed at points in the domain and/or at the boundary.

The unknowns (which are the coefficients of each RBF) are obtained from the approximate enforcement of the boundary conditions as well as the governing equations by means of collocation. Usually, this approximation only considers regular RBFs, such as the globally supported multiquadrics or the compactly supported Wendland functions[8].

RBFs may be classified into two main groups:

1. The globally supported ones namely the multiquadrics (MQ, $\sqrt{\left(x-x_j\right)^2+c_j^2}$, where c_j is a shape parameter), the inverse multiquadrics, thin plate splines, Gaussians, etc;
2. The compactly supported ones such as the Wendland[8] family (for example, $(1-r)_+^n + p(r)$ where $p(r)$ is a polynomial and $(1-r)_+^n$ is 0 for r greater than the support).

Let us suppose that we have a function of L variables x_i, $i = 1,\ldots, L$. The globally supported RBF model used in this work has the following form

$$s(\mathbf{x}) = f(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi\left(\left|\mathbf{x}-\mathbf{x}_j\right|\right) + \sum_{k=1}^{M}\sum_{i=1}^{L} \beta_{i,k}\, p_k(x_i) + \beta_0 \tag{1}$$

where $\mathbf{x} = \{x_1,\ldots, x_i,\ldots, x_L\}$ and $f(\mathbf{x})$ is known for a series of points $\mathbf{x}$. Here, N is the number of collocation (training) points, $p_k(x_i)$ is one of the M terms of a given basis of polynomials[9]. This approximation is solved for the α_j and $\beta_{i,k}$ unknown coefficients from the system of N + M*L + 1 linear equations, subject to

$$\sum_{j=1}^{N} \alpha_j\, p_k(x_i) = 0 \quad \vdots \quad \sum_{j=1}^{N} \alpha_j\, p_k(x_L) = 0 \qquad \text{and} \qquad \sum_{j=1}^{N} \alpha_j = 0 \tag{2),(3}$$

For each of the basic RBF models given in Table 1 (except for the fittest polynomial RBF model) used in this work, the polynomial part of Eq. (1) as well as the constant $\beta_{o,}$ in such equation was not used. Thus, Eqs. (2) and (3) were not used either except for the fittest polynomial RBF method. Fifteen different RBF models were used in this comparison, which are summarized below in Table 1. In this table, a short name of each approximation is provided, which will be used later in the comparisons. It should be noted that the shape parameter c_j was kept constant as $1/N$. The shape parameter c_j is used to control the smoothness of the RBF. Figure 1 shows the influence of the choice for c_j on the multiquadrics RBF.

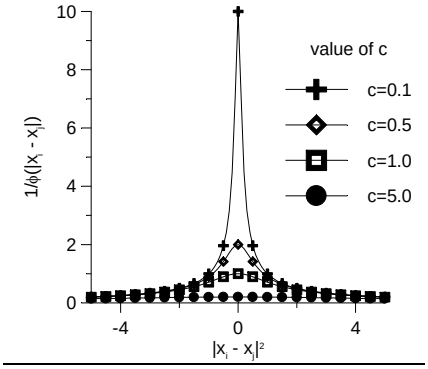

Figure 1. Influence of the shape parameter on the smoothness of an RBF.

Table 1. Basic RBF approximations used in this comparative analysis.

Short name	$\phi\left(\left\|x_i-x_j\right\|\right)$ See Eq. (1)	Short name	$\phi\left(\left\|x_i-x_j\right\|\right)$ See Eq. (1)
RBF1	$\phi\left(\left\|x_i-x_j\right\|\right)=\sqrt{\left(x_i-x_j\right)^2+c_j^2}$	RBF9	$\phi\left(\left\|x_i-x_j\right\|\right)=\left[\sqrt{\left(x_i-x_j\right)^2+c_j^2}\right]^{-3}$
RBF2	$\phi\left(\left\|x_i-x_j\right\|\right)=\exp\left[-c_j^2\left(x_i-x_j\right)^2\right]$	RBF10	$\phi\left(\left\|x_i-x_j\right\|\right)=\left\{\exp\left[-c_j^2\left(x_i-x_j\right)^2\right]\right\}^{-1}$
RBF3	$\phi\left(\left\|x_i-x_j\right\|\right)=\left(x_i-x_j\right)^2+c_j^2$	RBF11	$\phi\left(\left\|x_i-x_j\right\|\right)=J_0\left(\sqrt{\left(x_i-x_j\right)^2}\right)$
RBF4	$\phi\left(\left\|x_i-x_j\right\|\right)=\left[\sqrt{\left(x_i-x_j\right)^2+c_j^2}\right]^3$	RBF12	$\phi\left(\left\|x_i-x_j\right\|\right)=J_1\left(\sqrt{\left(x_i-x_j\right)^2}\right)$
RBF5	$\phi\left(\left\|x_i-x_j\right\|\right)=0$	RBF13	$\phi\left(\left\|x_i-x_j\right\|\right)=K_0\left(\sqrt{\left(x_i-x_j\right)^2}\right)$

RBF6	$\phi\left(\left\|x_i - x_j\right\|\right) = \left[\sqrt{\left(x_i - x_j\right)^2 + c_j^2}\right]^{-1}$	RBF14	$\phi\left(\left\|x_i - x_j\right\|\right) = \sqrt{\left(x_i - x_j\right)^2} K_1\left(\sqrt{\left(x_i - x_j\right)^2}\right)$
RBF7	$\phi\left(\left\|x_i - x_j\right\|\right) = \left[\left(x_i - x_j\right)^2 + c_j^2\right] \ln\left[\sqrt{\left(x_i - x_j\right)^2 + c_j^2}\right]$	RBF15	$\phi\left(\left\|x_i - x_j\right\|\right) = \dfrac{\left[\sum_{k=1}^{n} \dfrac{(n)!}{k!(n+k)!}\left(2c_j\sqrt{\left(x_i - x_j\right)^2}\right)^k\right] e^{\left[-c_j\sqrt{\left(x_i - x_j\right)^2}\right]\left[c_j\sqrt{\left(x_i - x_j\right)^2}\right]^n}}{(2n-1)!}$ $n = 9$
RBF8	$\phi\left(\left\|x_i - x_j\right\|\right) = \left[\left(x_i - x_j\right)^2 + c_j^2\right]^{-1}$		

3.2. The Fittest Polynomial RBF Method for Constructing Response Surfaces

The fittest polynomial method, which is proposed in this work, consists of using the best interpolation possible among the different RBF models presented in Table 1. In this model, the polynomial part of Eq. (2) was taken as

$$p_k\left(x_i\right) = x_i^k \tag{4}$$

and the RBFs are selected among the ones presented in Table 1.

From Eq. (4) one can notice that a polynomial of order *M* is added to the RBF. *M* was limited to an upper value of 6. After inspecting Eqs. (1)-(4), one can realize that the final linear system has [(*N*+*M***L*) + 1] equations. Some tests were also made using the cross-product polynomials ($x_i\, x_j\, x_k$ …), but the improvements of the results were insignificant. Also, other types of RBFs were used, but no improvement of the accuracy of the interpolation was observed.

The choice of which polynomial order and which RBF are the best for fitting a specific function, was made based on a cross-validation procedure. Let us suppose that we have P_{TR} training points, which are located in the multidimensional space where the values of the function are known. Such set of training points is then equally subdivided into two subsets of points, named P_{TR1} and P_{TR2}. The Eqs. (1)-(4) are solved for a polynomial of order zero and for the first RBF expression given at Table 1 using the subset P_{TR1}. Then, the values of the interpolated function are checked against the known values of the function for the subset P_{TR2} and the error is recorded as

$$RMS_{PTR1,M=0,RBF1} = \sum_{i=1}^{P_{TR2}}\left[s\left(x_i\right) - f\left(x_i\right)\right]^2 \tag{5}$$

Then, the same procedure is repeated, using the subset P_{TR2} to solve Eqs. (1)-(4) and the subset P_{TR1} to calculate the error as

$$RMS_{PTR2,M=0,RBF1} = \sum_{i=1}^{P_{TR1}}\left[s\left(x_i\right) - f\left(x_i\right)\right]^2 \tag{6}$$

Finally, the total error for the polynomial of order zero and the first RBF expression given in Table 1 is obtained as

$$RMS_{M=0,RBF1} = \sqrt{RMS_{PTR1,m=0,RBF1} + RMS_{PTR2,m=0,RBF1}} \tag{7}$$

This procedure is repeated for all polynomial orders, up to *M* = 6 and for each one of the RBF expressions presented in Table 1. The best combination of the polynomial formulation and its order is the one that returns the lowest value of the RMS error. Although this cross-validation procedure is quite simple, it worked very well for all test cases analyzed in this paper. Some other procedures to improve the interpolation were implemented in the fittest polynomial RBF scheme, such as the optimization of the best shape factor *c* and an auto-scale and auto-normalized procedure, depending on the topology of the function being interpolated.

4. Performance Measurements

In accordance with having multiple metamodeling criteria, the performance of each metamodeling technique is measured from the following aspects[10].

- Accuracy – the capability of predicting the system response over the design space of interest.
- Robustness – the capability of achieving good accuracy for different problem types and sample sizes.
- Efficiency – the computational effort required for constructing the metamodel and for predicting the response for a set of new points generated by metamodels.
- Transparency – the capability of illustrating explicit relationships between input variables and responses.
- Conceptual Simplicity – ease of implementation. Simple methods should require minimum user input and be easily adapted to each problem.

For accuracy, the goodness of fit obtained from "training" data is not sufficient to assess the accuracy of newly predicted points. For this reason, additional confirmation samples are used to verify the accuracy of the metamodels. To provide a more complete picture of metamodel accuracy, three different metrics are used: R Square, Relative Average Absolute Error (RAAE), and Relative Maximum Absolute Error (RMAE)[10].

R Square (R2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} = 1 - \frac{MSE}{\text{variance}} \tag{8}$$

Here, $\hat{y}_i$ is the corresponding predicted value for the observed value y_i, while $\bar{y}$ is the mean of the observed values. While *MSE* (Mean Square Error) represents the departure of the metamodel from the real simulation model, the variance captures how irregular the problem is. The larger the value of R2, the more accurate the metamodel.

Relative Average Absolute Error (RAAE)

$$RAAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n * STD} \tag{9}$$

Here, *STD* stands for standard deviation. The smaller the value of RAAE, the more accurate the metamodel.

Relative Maximum Absolute Error (RMAE)

$$RMAE = \frac{\max(|y_1 - \hat{y}_1|, |y_2 - \hat{y}_2|, ..., |y_n - \hat{y}_n|)}{STD} \tag{10}$$

Large RMAE indicates large error in one region of the design space even though the overall accuracy indicated by R2 and RAAE can be very good. Therefore, a small RMAE is preferred. However, since this metric cannot show the overall performance in the design space, it is not as important as R2 and RAAE.

Although the R2, RAAE and RMAE are usefull to ascertain the accuracy of the interpolation, they can fail in some cases. For the R2 metric, for example, if one of the testing points has a huge deviation from the exact value, such discrepancy might affect the entire sum appearing in Eq. (8) and, even if all the other testing points are accurately interpolated, the R2 result can be very bad. For this reason, we also calculate the percentage deviation from the exact value of each testing point. Such deviations are collected according to six ranges of errors: 0-10%; 10-20%; 20-50%; 50-100%; 100-200%; >200%. Thus, an interpolation that has all testing points within the interval of 0 to 10% of relative error might be considered good in comparison to another one where the points are all spread along the intervals from 10 to 200%.

5. Test Functions

In order to test the accuracy of the RBF models proposed, 296 test cases were used, representing linear and non-linear problems with up to 100 variables. Such problems were selected from a collection of 395 problems (actually 296 test cases), proposed by Hock and Schittkowski[11] and Schittkowski[12]. Figure 2 shows the number of variables of each one of the problems. Note that there are 395 problems, but some of them were not used because problem numbers ranging from 120 to 200 were not defined in the original publications.

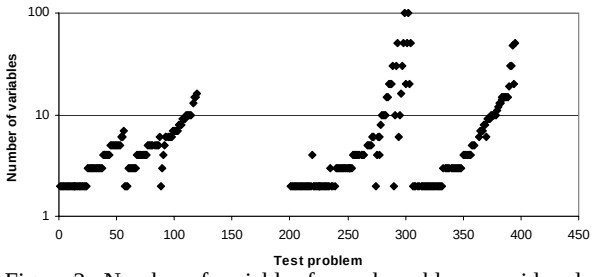

Figure 2. Number of variables for each problem considered.

In order to verify the accuracy of the interpolation over different number of training points, two sets were defined; scarce and small. Also, the number of corresponding testing points varied according to the number of training points. Table 2 presents these two sets, based on the number of dimensions (variables) *L* of the particular problem.

Table 2. Number of training and testing points where L is the number of variables.

	Number of training points	Number of testing points
Scarce set	3 *L*	300 *L*
Small set	10 *L*	1000 *L*

For example, if we are trying to fit a function of say, 500 variables, then a scarce set will have 3*500 = 1500 training points and 300*500 = 150,000 testing points. The large matrix resulting from Eq. (1) assuming that we use polynomial of order M = 6, will be of size (6*500 + 1,500 + 1 = 4501) squared. Such large resulting matrices have been solved iteratively using bi-conjugate iterative method[13]. Besides, cross-validation is performed for each combination of these, say, 4501 x 4501 coefficients.

5.1. Results with Scarce Sets (3*L) of Training Points

Figure 3 shows the R2 metric for all test cases, using the scarce set (3*L) of training points only for the interpolation functions that presented some meaningful results. It can be noticed that the results are all spread from 0 (completely inadequate interpolation) to 1 (very accurate interpolation). Qualitatively, the darkest pictures mean better results. It is worth noting that the fittest polynomial RBF method provides the best interpolation among all test functions. The RBF1, RBF4 and RBF7 (refer to Table 1) also performed a good interpolation, based on the R2 metric.

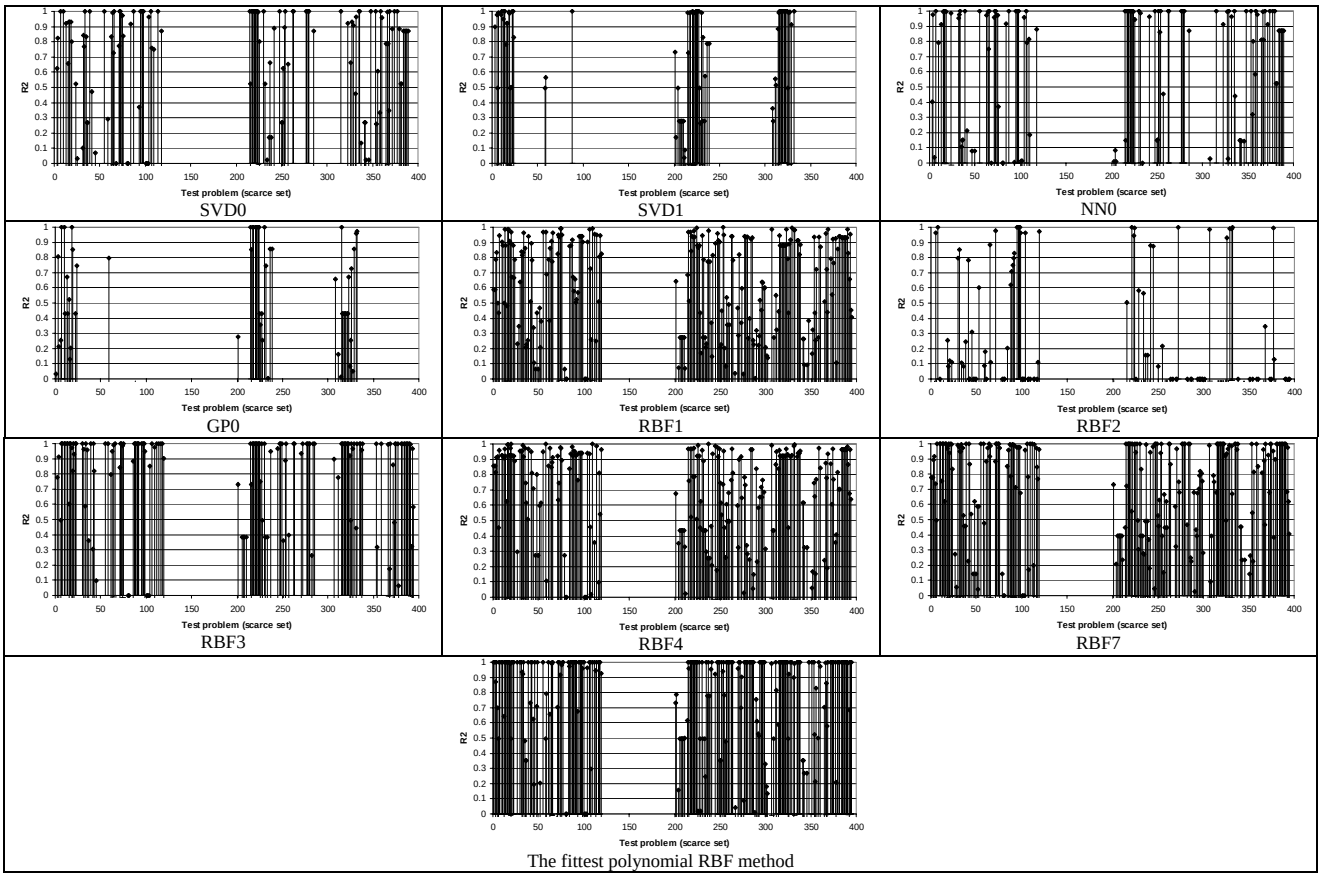

Figure 3. R2 metric for various fitting method using scarse set (3*L) of training points.

Figure 4 shows the CPU time required to interpolate each test function, using the scarce set (3*L) of training points for the fittest polynomial RBF method. For most of the cases, the computing time was less than 10 seconds, using a Pentium IV 3 GHz with 1Gb RAM running Rocks 4.2.1 (Cydonia) and the Intel© ifort (IFORT) 9.1 20060323 compiler with the directives "-static-libcxa -static -O3 -r8". In fact, the highest dimensional test case, which has 100 variables required only 50 seconds to be interpolated. The computing times for the other methods will be presented later.

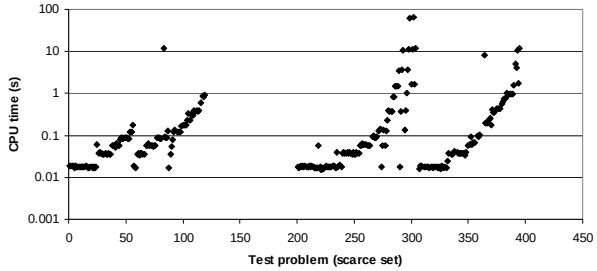

Figure 4. Computing time for the scarce set (3*L) of training points using the fittest polynomial RBF method.

Although the R2 might indicate some performance behavior of the interpolation function, we decided to use a different measure of accuracy. Figure 5 shows the percentage of hypersurface fits having errors less than 10 percent when fitting all of the 296 test cases, for the scarce set of testing points. It is worth noting that not all test cases ran in modeFRONTIER®. In such cases, either the program crashed or the computational time was so high (greater than three hours) that we aborted the interpolation. The ordinate in each figure shows the number of test cases that ran for each method. For example, from Figure 5, it can be noticed that for the fittest polynomial RBF method, 6 percent of all the points in 11 percent of all of the 296 test functions had relative error less than 10 percent. Similarly, for the fittest polynomial RBF method, in more than 40 percent of all test functions, the relative errors were less than 10 percent for more than 94 percent of the points. This is a very good result, considering the extremely small number of training points used in the scarce set (only three times the number of variables). For the RBF7 method, more than 30 percent of all 296 test functions had relative errors less than 10 percent. From this analysis, RBF1 and RBF4 had a very poor performance, showing that this type of analysis might be more complete than the R2 metric alone.

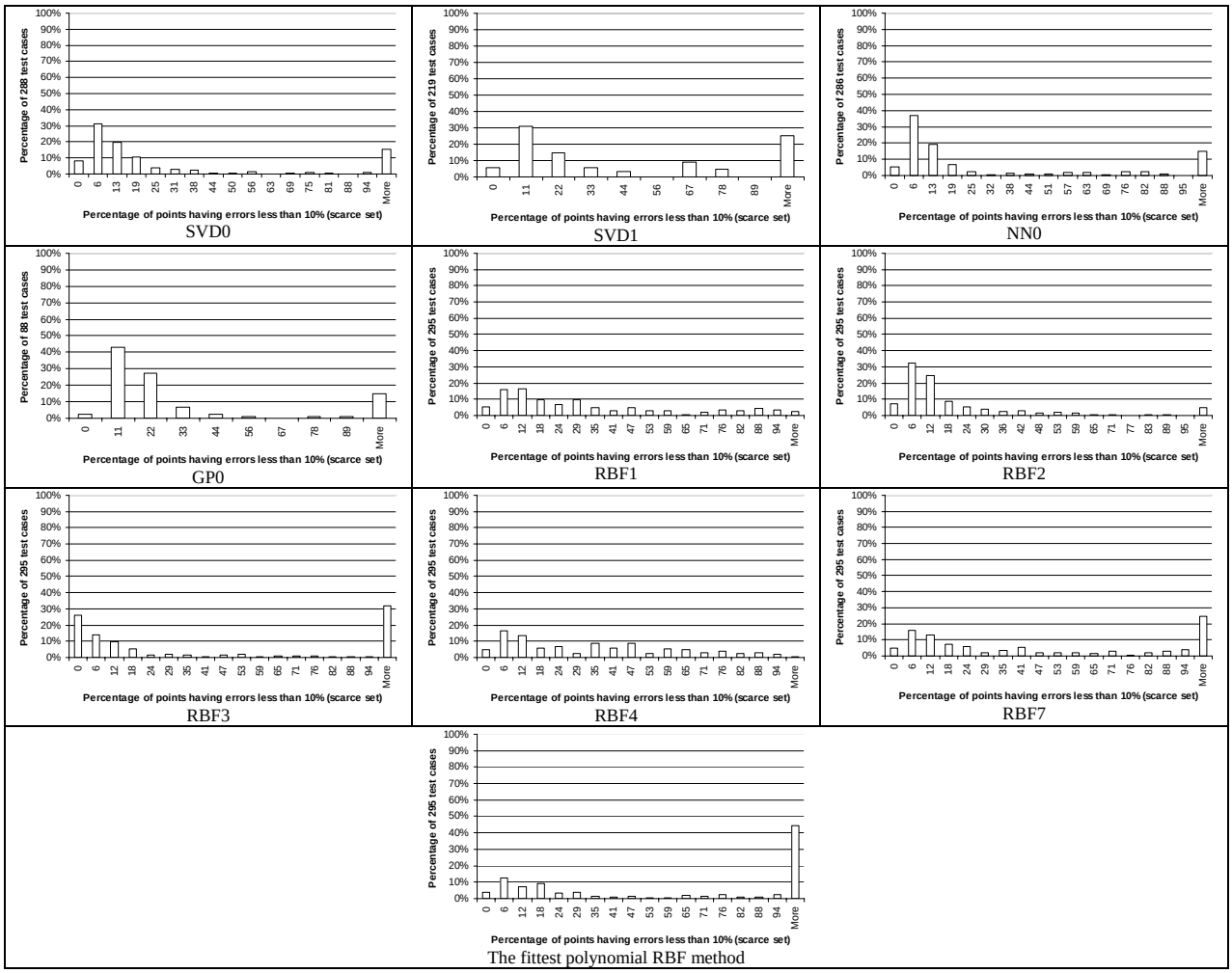

Figure 5. Testing points with less than 10% error, for the scarce set (3*L) of training points.

Table 3 shows the average value and the standard deviation of the R2 metric for all interpolation functions as well as for the percentage of functions where relative errors were less than 10 percent, for the scarce set of data. It shows also the average and standard deviation of the computing time for the RBFs and the fittest polynomial RBF method. Although the computing times for the test cases ran with modeFRONTIER are not shown, because the code was written in a different language, in general they are much higher than the ones with the basic RBFs and the fittest polynomial RBF method. In all test cases the values of the RMAE and RAAE metrics are not shown, because they varied from zero to infinity and no such comparison was possible among all methods for these metrics. From Table 3 it is possible to confirm that the fittest polynomial RBF method had the best performance for the R2 metric and also the highest number of functions with relative errors less then 10 percent. Besides the fittest polynomial RBF method, the SVD1 and the RBF7 also had a very good performance. For some methods (KR0, RBF5, RBF13 and RBF14, for example) the results were completely inadequate, with a R2 result equal to zero. It is worth noting that the computing time for the fittest polynomial RBF method was of the same order of magnitude as for the other RBFs.

Table 3. Comparison among all interpolation functions for the scarce set of data.

Response Surface Model	Number of test functions successfully evaluated	R2		Percentage of functions with relative errors less than 10%		CPU time (s)	
		Avg. Value	Std. Dev.	Average Value	Standard Deviation	Avg. Value	Std. Dev.
SVD0	288	0.31	0.42	26.07	35.17		
SVD1	219	0.64	0.39	41.53	39.86		
NN0	286	0.30	0.43	28.16	36.25	----------	
KR0	252	0.00	0.06	6.17	10.77		
GP0	88	0.37	0.39	26.93	33.24		
RBF1	296	0.52	0.38	30.03	29.25	0.31	1.68
RBF2		0.11	0.29	17.05	24.34	0.33	1.79
RBF3		0.44	0.47	39.87	44.39	0.32	1.70
RBF4		0.54	0.39	31.63	26.65	0.32	1.70
RBF5		0.00	0.06	0.68	7.65	0.33	1.78
RBF6		0.04	0.16	12.93	17.90	0.32	1.68
RBF7		0.58	0.41	45.02	39.72	0.33	1.76

RBF8		0.03	0.14	12.56	17.32	0.31	1.68
RBF9		0.02	0.13	12.40	17.13	0.32	1.69
RBF10		0.06	0.23	6.41	21.51	0.49	2.69
RBF11		0.05	0.21	13.80	21.44	0.33	1.74
RBF12		0.04	0.17	7.32	18.93	0.34	1.78
RBF13		0.00	0.06	0.96	7.76	0.33	1.81
RBF14		0.00	0.00	0.53	5.31	0.36	1.92
RBF15		0.08	0.24	6.80	18.33	0.72	3.85
The fittest polynomial RBF method		0.66	0.42	58.15	42.47	0.96	5.48

5.2. Results with Small Sets (10*L) of Training Points

Figure 6 shows the R2 metric for the small set (10*L) of interpolation data for all functions that presented some meaningful result. It is clear that the results improved when compared with the ones presented in Fig. 2 (for the scarce set 3*L of training data). Remembering that, qualitatively, darker figures mean better results, the SVD1, NN0, RBF4, RBF7 and the fittest polynomial RBF functions are better then ones presented in Fig. 2. For this small set of data, the SVD1 and RBF15 were capable of presenting meaningful results, although the results with RBF15 are still not satisfactory. The other functions, at least from the observation of the R2 metrics in Fig. 6, did not improve too much by going from scarce set (3*L) to small set (10*L) of training points and the corresponding numbers of testing points.

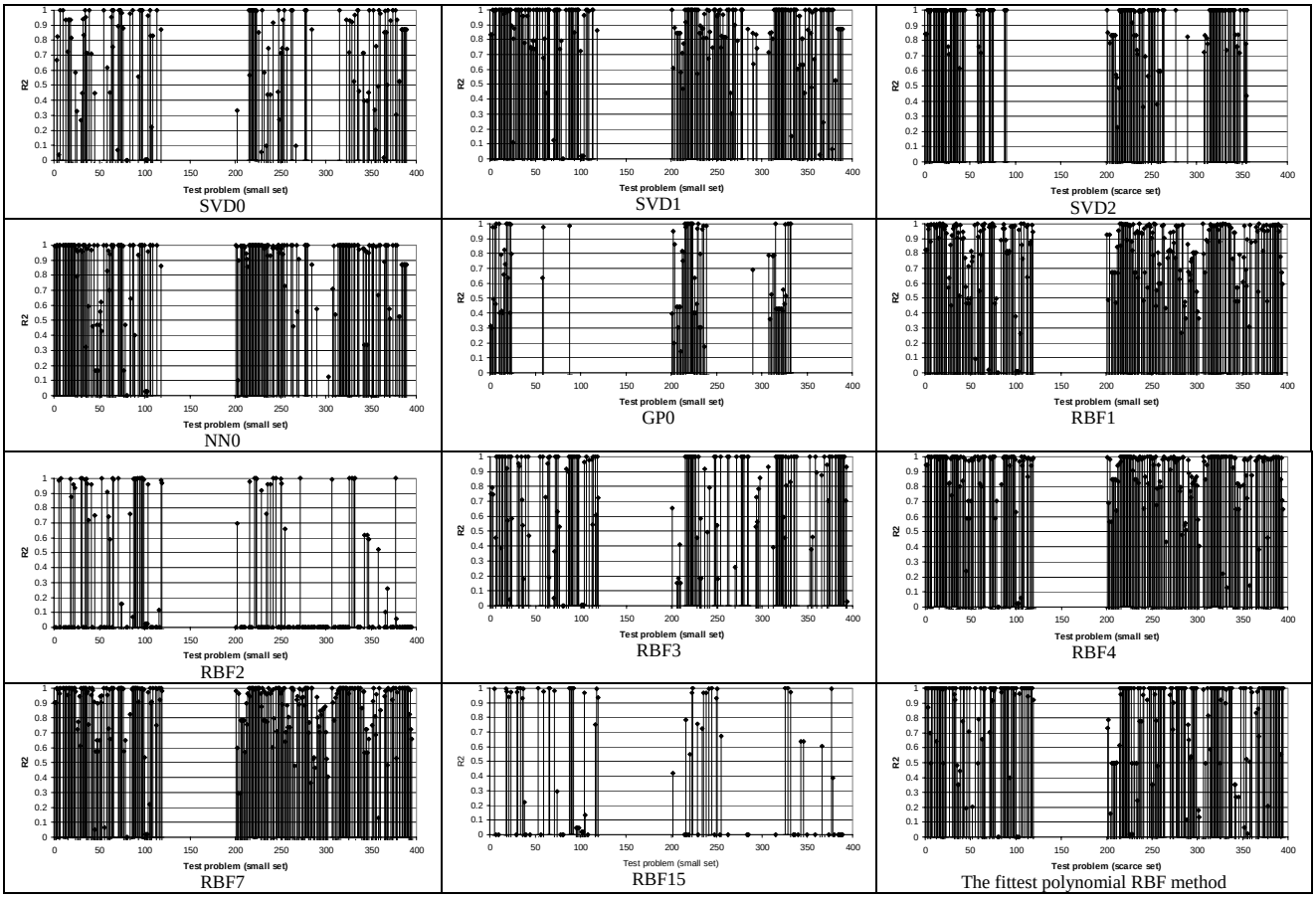

Figure 6. R2 metric for the small set (10*L) of training points.

Figure 7 shows the computing time required to interpolate each test function, using the small set (10*L) of training points for the fittest polynomial RBF method. For most of the cases, the computing time was less than 100 seconds, using a a Pentium IV 3 GHz with 1Gb RAM running Rocks 4.2.1 (Cydonia) and the Intel© ifort (IFORT) 9.1 20060323 compiler with the directives "-static-libcxa -static -O3 -r8". The highest dimensional test cases, which had 100 variables, required only 500 seconds to be interpolated.

Figure 8 shows the percentage of testing points having errors less than 10 percent, against the percentage of all 296 test cases, for the small set (10*L) of testing points. Again, not all test cases ran with modeFRONTIER®. The ordinate in each figure shows the number of test cases that ran for each method. Comparing these figures with the ones presented for the scarce set of data, Fig. 4, one can see that the overall number of functions interpolated with modeFRONTIER® increased. The quality of the results for the SVD0 and GP0 methods did not improve too much, while the SVD1 and NN0 methods had a considerable increase in performance regarding the number of test functions that were fitted with errors less than 10 percent. The RBF1, RBF2 and RBF3 did not improve too much, while the RBF4, RBF7 and the fittest polynomial RBF method had a very good improvement in their performances. Again, the best approach was the fittest polynomial RBF method, where more then 60 percent of all 296 test functions were fitted with a relative error less than 10 percent. Looking back at Fig. 6, it is worth mentioning again that the analysis of the R2 metric alone

could not reveal all these details. However, from both the results (Fig. 6 and Fig. 7) the fittest polynomial RBF method was the best for the small set (10*L) of training data.

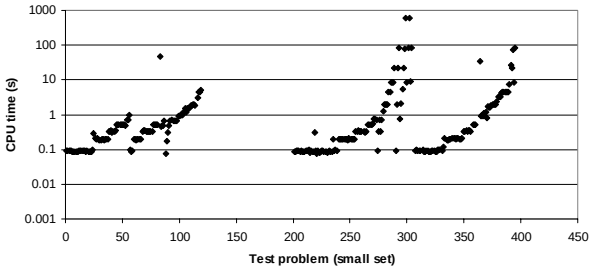

Figure 7. Computing time for the small set (10*L) of training points using the fittest polynomial RBF method.

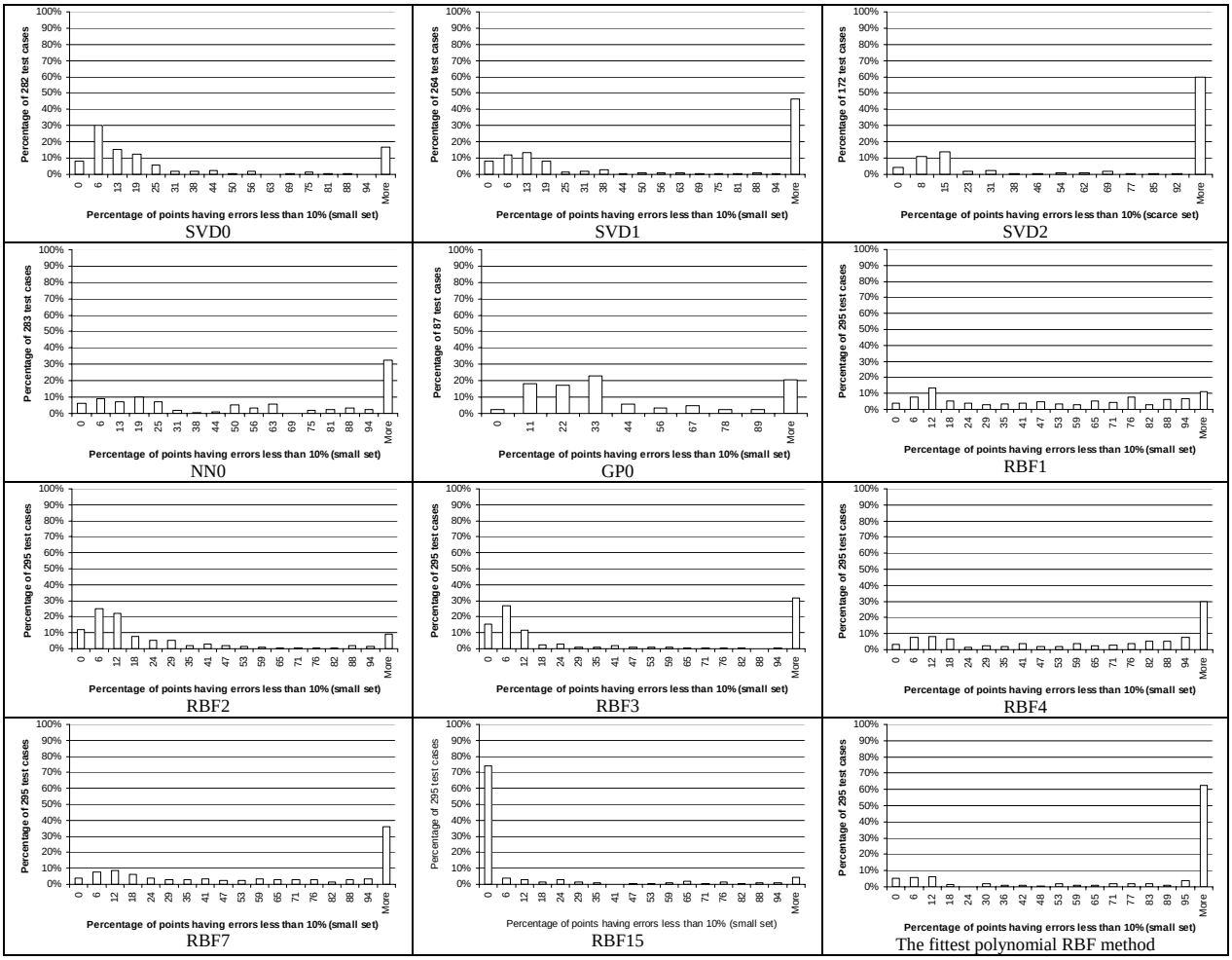

Figure 8. Testing points with less than 10 percent error, for the small set of training points.

Table 4 shows the average value and the standard deviation of the R2 metric for all 296 interpolation functions as well as for the percentage of functions where relative errors were less than 10 percent, for the small set of data (10*L). This table also shows the average and standard deviation of the computing times. From this table, again it is evident that the fittest polynomial RBF method was the one with the higher value of the R2 and the number test of functions where fitting errors were less than 10 percent. Performance of the fittest polynomial RBF method was followed by the RBF4, RBF7 and SVD1. Again, some of the methods had a very poor performance with R2 metrics equal to zero. It is remarkable that the average value of the R2 metric for the fittest polynomial RBF method was 0.83 for this test case where the number of training functions was equal to ten times the number of variables (10*L). Also, for the fittest polynomial RBF method, 73.83 percent of all test functions had a fitting error less than 10 percent, while for the second best, the RBF4, this was the case for only 59.97 percent of the test functions. The average computing time for the fittest polynomial RBF method, again, was of the same order of magnitude as for the other RBFs.

Table 4. Comparison among all interpolation functions for the small set (10*L) of data.

Response Surface Model	Number of test functions succesfully evaluated	R2		Percentage of functions with relative errors less than 10%		CPU time (s)	
		Avg. Value	Std. Dev.	Average value	Standard Deviation	Avg. Value	Std. Dev.
SVD0	282	0.36	0.43	28.16	35.69	----------	
SVD1	264	0.76	0.36	55.03	44.25		
SVD2	172	0.83	0.30	66.92	42.56		
NN0	283	0.64	0.44	53.20	39.07		
KR0	138	0.00	0.00	6.73	11.96		
GP0	87	0.65	0.31	40.29	34.70		
RBF1	295	0.74	0.33	48.20	34.42	1.97	14.43
RBF2		0.19	0.37	22.97	31.13	2.06	15.15
RBF3		0.44	0.46	38.53	44.12	2.08	15.48
RBF4		0.80	0.33	59.97	37.42	2.02	14.91
RBF5		0.00	0.06	1.14	8.34	2.19	15.69
RBF6		0.05	0.18	13.69	19.58	2.00	14.73
RBF7		0.78	0.33	58.08	38.97	2.10	15.14
RBF8		0.03	0.14	13.14	18.61	1.96	14.40
RBF9		0.02	0.13	12.95	18.30	2.01	14.80
RBF10		0.14	0.34	12.10	29.84	3.86	24.97
RBF11		0.06	0.22	14.49	23.39	2.15	15.43
RBF12		0.06	0.23	9.92	23.36	2.32	16.47
RBF13		0.00	0.06	0.77	7.66	2.09	15.05
RBF14		0.00	0.00	0.77	6.09	2.53	16.31
RBF15		0.14	0.33	11.75	27.14	6.52	37.78
The fittest polynomial RBF method		0.84	0.33	75.41	37.77	7.32	51.90

Finally, Figs. 9 and 10 summarize all the results previously shown for the R2 metric and for the test cases with errors less then 10 percent. In these figures, we also added a new set of training data, called “medium set”. For this case, the number of training data was equal to 50 *L*, while the number of testing data was equal to 5000 *L*. From Fig. 9, one can see that the average value of the R2 metric for the fittest polynomial RBF method is the highest among all the methods considered in this paper. Although the GP0 has high values of the R2 metrics, one can see from Tables 3 and 4 that it did not run for all functions, but only for a reduced number of them. From this comparison, the SVD1, SVD2 (although it did not run for the scarce set of data), NN0, RBF1, RBF4 and RBF7 also performed very well.

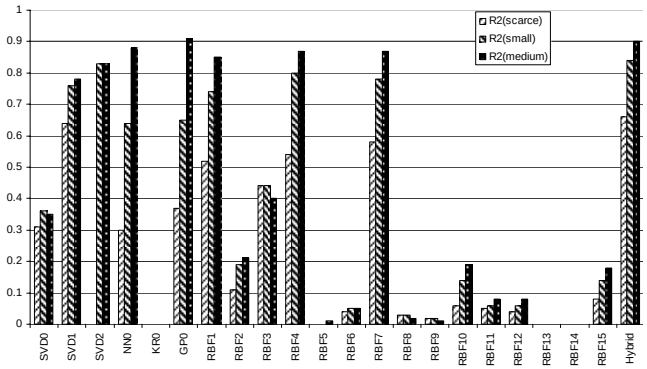

Figure 9. Results when using all methods in terms of R2 metric over all test cases.

Figure 10 shows the percentage of functions with errors less then 10 percent over all test cases, for all functions tested. Again, the fittest polynomial RBF method is the one with the best performance among all methods analyzed. From this analysis, the NN0, RBF4 and RBF7 also performed very well.

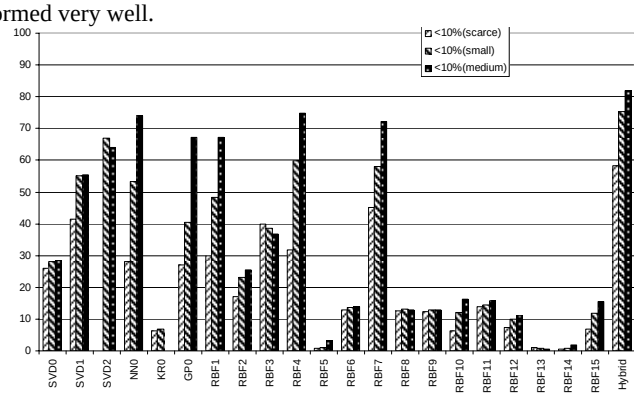

Figure 10. Results for the number of functions with errors less then 10 percent over all test cases.

6. Conclusions

In this paper we compared several methodologies to generate response surfaces for different kinds of linear and non-linear functions. The total number of such test functions was 296. Two different sets of training points were analyzed, corresponding to scarce and small set of data. The fittest polynomial radial basis function method was proposed which proves to be superior to using a single approach. This multi-dimensional surface fitting method can be applied to the solution of algebraic and partial differential equations, as well as to use as a surrogate model for complex physical problems.

7. Acknowledgments

This work was partially funded by CNPq and FAPERJ (Brazilian councils for scientific development). M. J. Colaço is very gratefull to the financial support from FIU as well as the hospitally of the Dulikravich's family during his staying in Miami from September to November of 2006, April of 2007 and January of 2008. The authors would also like to thank the ESTECO and ESSS Companies, specially Dr. Carlo Poloni, Dr. Nader Fateh, and Mr. Rodrigo Ferraz for providing free access to modeFRONTIER® software. This work was also partially sponsored by the US Air Force Office of Scientific Research under grant FA9550-06-1-0170 monitored by Dr. Todd E. Combs, Dr. Fariba Fahroo and Dr. Donald Hearn and by the US Army Research Office/Materials Division under the contract number W911NF-06-1-0328 monitored by Dr. William M. Mullins. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Air Force Office of Scientific Research, the US Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

8. References

[1]Egorov-Yegorov, I. N. and Dulikravich, G. S., "Chemical Composition Design of Superalloys for Maximum Stress, Temperature and Time-to-Rupture Using Self-Adapting Response Surface Optimization", *Materials and Manufacturing Processes*, vol. 20, No. 3, pp. 569-590, 2005.

[2]Colaço, M. J., Dulikravich, G. S. and Sahoo, D., "A Comparison of Two Methods for Fitting High Dimensional Response Surfaces", *Proc. of International Symposium on Inverse Problems, Design and Optimization (IPDO-2007)*, (eds.: Dulikravich, G. S., Orlande, H. R. B., Tanaka, M. and Colaco, M. J.), Miami Beach, Florida, U.S.A., April 16-18, 2007.

[3]Sahoo, D. and Dulikravich, G. S., "Evolutionary Wavelet Neural Network for Large Scale Function Estimation in Optimization", *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA Paper AIAA-2006-6955, Portsmouth, VA, September 6-8, 2006.

[4]Kansa, E.J., "Multiquadrics – A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics – II: Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations", *Comput. Math. Applic.*, vol. 19, pp. 149-161, 1990.

[5]Hardy, R.L., "Multiquadric Equations of Topography and Other Irregular Surfaces", *Journal of Geophysics Res.*, vol. 176, pp. 1905-1915, 1971.

[6]Leitão, V.M.A., "A Mesheless Method for Kirchhoff Plate Bending Problems", *International Journal of Numerical Methods in Engineering*, vol. 52, pp. 1107-1130, 2001.

[7]Leitão, V.M.A., "RBF-Based Meshless Methods for 2D Elastostatic Problems", *Engineering Analysis with Boundary Elements*, vol. 28, pp. 1271-1281, 2004.

[8]Wendland, H., "Error Estimates for Interpolation by Compactly Supported Radial Basis Functions of Minimal Degree", *Journal of Approximation Theory*, vol. 93, pp. 258-272, 1998.

[9]Buhmann, M.D., "Radial Basis Functions on Grids and Beyond", *International Workshop on Meshfree Methods*, Lisbon, Portugal, 2003.

[10]Jin, R., Chen, W. and Simpson, T.W., "Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria", *Proc. of the 8th AIAA / USAF / NASA / ISSMO Multidisciplinary Analysis & Optimization Symposium*, AIAA 2000-4801, Long Beach, CA, September 6-8, 2000.

[11]Hock, W. and Schittkowski, K., *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, vol. 187, Springer, 1981.

[12]Schittkowski, K., *More Test Examples for Nonlinear Programming*, Lecture Notes in Economics and Mathematical Systems, vol. 282, Springer, 1987.

[13]Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., *Numerical Recipes: in Fortran*, Cambridge University Press, U.K, 1992.