Inverse Problems, Design and Optimization Symposium
Miami, Florida, U.S.A., April 16-18, 2007

A COMPARISON OF TWO METHODS FOR FITTING HIGH DIMENSIONAL RESPONSE SURFACES

Marcelo J. Colaço
Department of Mechanical and Materials Eng.
Military Institute of Engineering
Rio de Janeiro, RJ, Brazil
colaco@asme.org

George S. Dulikravich
Department of Mechanical and Materials Eng.
Florida International University
Miami, Florida, USA
dulikrav@fiu.edu

Debasis Sahoo
Infinite Cloud, LLC
48 Wall St. New York, NY
dsahoo@infinitecloud.com

ABSTRACT

In this paper we compare two methodologies to interpolate linear as well as highly non-linear functions in multidimensional spaces having up to 500 dimensions. Two methodologies are compared: the Radial Basis Function (RBF) and the Wavelet based Neural Network (WNN). The accuracy, robustness, efficiency, transparency and conceptual simplicity are discussed. Based on the extensive testing performed on 13 test functions, the RBF approach seems easy to implement computationally and results are better interpolation for higher dimensional spaces than the WNN, requiring lesser computing time.

INTRODUCTION

The use of RBFs followed by collocation, a technique first proposed by Kansa [Kansa, 1990], after the work of Hardy [Hardy, 1971] on multivariate approximation, is now becoming an established approach. Various applications to problems in mechanics have been made in recent years – see, for example Leitão [Leitão, 2001; Leitão, 2004].

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 [Wendland, 1998].

RBF 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 [1998] 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,…,L. The RBF model used in this work has the following form

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

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, $p_k(x_i)$ is one of the M terms of a given basis of polynomials [Buhmann, 2003]. This approximation is solved for the α_j and $\beta_{i,k}$ unknowns from the system of N linear equations, subject to the conditions (for the sake of uniqueness)

$$\sum_{j=1}^{N}\alpha_j p_k(x_i)=0 \quad (2.a)$$

$$\vdots$$

$$\sum_{j=1}^{N}\alpha_j p_k(x_L)=0$$

$$\sum_{j=1}^{N}\alpha_j=0 \quad (2.b)$$

In this work, the polynomial part of Eq. (1) was taken as

$$p_k(x_i)=x_i^k \quad (3)$$

and the multiquadrics radial function

$$\phi\left(\left|x_i-x_j\right|\right)=\sqrt{\left(x_i-x_j\right)^2+c_j^2} \quad (4)$$

was used with the shape parameter c_j which is kept constant as 1/*N*. Thus, a polynomial of order *M* is added to the radial basis function. *M* was limited to an upper value of 10. After inspecting Eqs. (1)-(4), one can easily check that the final linear system has [(*N*+*M***L*)+1] equations.

The second method used for fitting high dimensional functions was the Wavelets based Neural Network (WNN) model presented by Sahoo and Dulikravich [2006] with 5 neural subnets. Training the WNN for response surface generation was done using a random sequence dataset of Sobol and Levitan [1976]. Typically, the mother wavelet used in the WNN is Mexican Hat wavelet given by

$$\psi(x) = \left(\frac{2}{\sqrt{3}}\pi^{-1/4}\right)\left(1 - x^2\right) exp\left(\frac{-x^2}{2}\right) \tag{5}$$

Gaussian wavelets were also used along with this mother wavelet to construct the WNN. For each node of the WNN, genetic algorithm was used to search the best Mexican Hat wavelet and the best Gaussian wavelet. The one having a lower norm of residue after performing multiple linear regression was selected and used in the WNN architecture.

TEST FUNCTIONS

In order to test the accuracy of the RBF model proposed, 13 test cases were used, representing linear and non-linear problems with up to 16 variables. These test cases, defined as problems 1 to 13 were selected by Jin *et al.* [2000] in a comparative study among different kinds of meta-models. Such problems were selected from a collection of 395 problems (actually 296 test cases), proposed by Hock and Schittkowski [1981] and Schittkowski [1987]. The first 12 problems do not have random errors added to the original function, while the problem no. 13 has a noise added with the following form

$$\varepsilon(x_1, x_2) = \sigma r \tag{6}$$

where σ is the standard deviation and *r* is a random number with Gaussian distribution and zero mean. In accordance with having multiple metamodeling criteria, the performance of each metamodeling technique is measured from the following aspects [Jin *et al.,* 2000].

- 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 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) [Jin *et al.,* 2000].

a) R Square (R2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{MSE}{\text{variance}} \tag{7}$$

where $\hat{y}_i$ is the corresponding predicted value for the observed value y_i; $\bar{y}$ 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.

b) Relative Average Absolute Error (RAAE)

$$RAAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n * STD} \tag{8}$$

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

c) Relative Maximum Absolute Error (RMAE)

$$RMAE = \frac{\max\left(|y_1 - \hat{y}_1|, |y_2 - \hat{y}_2|, \ldots, |y_n - \hat{y}_n|\right)}{STD} \tag{9}$$

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.

RESULTS

The RBF model presented here was compared WNN method for the 13 selected test cases. Table 1 gives the number of training points, testing points, minimum and maximum value of each test function, as well as the standard deviation and average value of each test function.

Table 1 – Parameters for the 13 test functions

		Number of training and testing points				Non Linearity	Minimum value of *f*	Maximum value of *f*	Standard deviation of *f*	Average value of *f*
		Training			Testing					
PB #	# Vars	Scarce	Small	Large						
PB1	10	30	100	198		High	-21.01	29.25	8.92	8.60
PB2	10	30	100	198		Low	-1717.43	-539.36	187.40	-1146.82
PB3	10	30	100	198		High	-1327241.16	-1.14	224561.73	-213124.84
PB4	10	30	100	198		Low	258.78	4779.01	1002.27	2185.25
PB5	16	48	160	459		Low	7136.99	195608.81	27294.72	69060.73
PB6	2		9	100		High	98.94	187.53	17.75	127.73
PB7	2		9	100	1000	High	-0.26	0.26	0.15	0.00
PB8	2		9	100		Low	-1.51	8.91	2.08	1.66
PB9	3	N/A	27	125		High	2.15	146369.38	21527.31	10842.11
PB10	3		27	125		High	3.42	32.84	7.70	29.13
PB11	3		27	125		Low	-40829.97	-40749.42	21.60	-40790.41
PB12	2		9	100		Low	-12.72	13.75	5.74	0.52
PB13	2		9	100		Low	-54.77	77.80	30.94	12.60

Three methodologies were used to solve the linear algebraic system resulting from Eqs. (1)-(4): LU decomposition, SVD and the Generalized Minimum Residual (GMRES) iterative solver. When the number of equations was small (less than 40), the LU solver was used. However, when the number of variables increased over 40, the resulting matrix becomes too ill-conditioned and the SVD solver had to be used. For more than 80 variables, the GMRES iterative method with the Jacobi preconditioner was used.

In order to check the accuracy of the metamodel when different samples were employed, three different sets of training points were used, as suggested by Jin *et al.* [2000].

Initially, the accuracy of the RBF expansion was tested for a large set of training points as defined in Table 1. Figure 1 shows the results for different orders of the polynomial part of the RBF expansion presented in Eqs. (1)-(3). From this figure, one can see that by increasing the order of the polynomial the results become much better, except for the problem no. 9, where the R2 metric decreases when M increases. In fact, for all problems, except no. 10, the RBF expansion achieves the R2 metric over 0.9, showing a very good global accuracy. For a large set of training points, the main conclusion is that a high polynomial order should be used in order to achieve high accuracy.

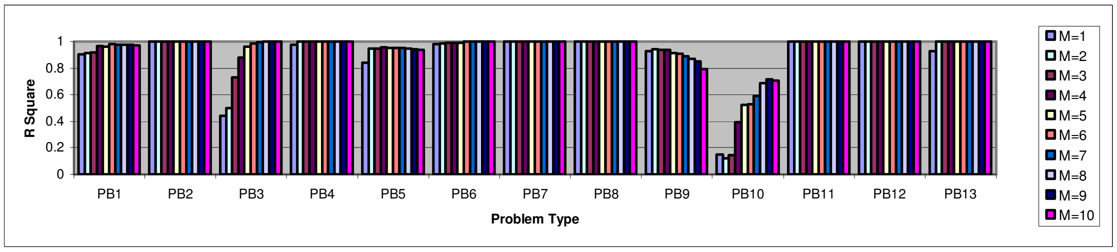

Figure 1 – Influence of the polynomial order on the R2 metric for a large set of training points

Next, the same comparison was made for a small number of training points, as defined in Table 1. Figure 2 shows the results of the R2 metric for this comparison. It can be observed that problem no. 1 is almost insensitive to the order of the polynomial, except for M = 9. Problem no. 11 is also insensitive except for M greater than 8, where the performance drops rapidly. Problems no. 2 and 4 are insensitive to the value of M, while problem number 7 shows a small decrease of the R2 value for high order polynomials. Problem no. 3 shows an increase in the R2 values when *M* increases, just as in the case with a large set of training points. However, problems no. 5, 6, 9, 10 and 13 show a drop in the R2 value when the polynomial order is increased so that some metrics were way below zero. From this figure, in order to keep the method robust, we conclude that, if the number of training points is small, the order of the polynomial should be kept small.

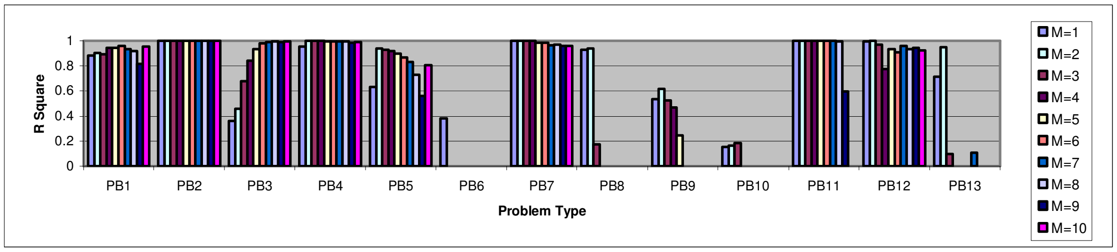

Figure 2 – Influence of the polynomial order on the R2 metric for a small set of training points

Figure 3 shows the comparison of the R2 metrics for several polynomial orders for a scarce set of training points. Only problems no. 1 to 5 were tested for a scarce set of training points, as suggested by Jin *et al.* [2000]. Problems no. 2 and 3 have the same behavior as for a small set of training points. However, problem no. 1 rapidly drops its value of R2 for a high polynomial order. Again, the minimum value of the scale was limited to zero, because some of the R2 values were way below zero. The performance of problem no. 5 oscillates when *M* is varied. Again, we concluded that for a scarce number of training points, a lower polynomial order should be used in order to keep the method more robust.

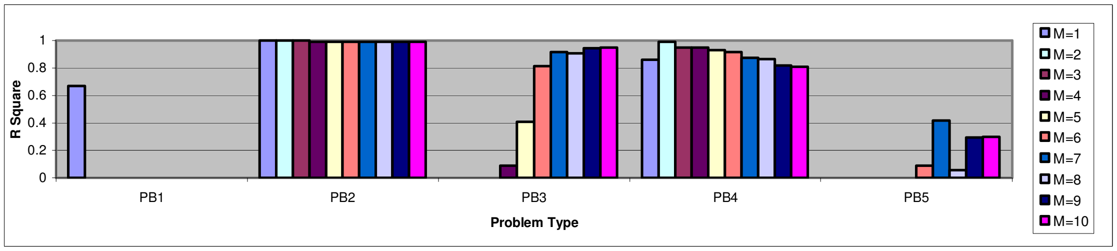

Figure 3 – Influence of the polynomial order on the R2 metric for a scarce set of training points

Next, the results obtained with a RBF polynomial of order 10 using a large number of training points and the results obtained with a polynomial of order 1 for small and scarce sets of training points were compared with the results obtained by using WNN method [Sahoo and Dulikravich, 2006]. Again, only problems no. 1 to 5 were tested for a scarce set of training points, as suggested by Jin *et al.* [2000].

The results for the R2 metric are presented in Fig. 4, where one can see that the RBF formulation performed better than the WNN for a scarce number of training points for problems no. 1 and 2. For problem no. 4 the value of R2 is a little small for the RBF when compared to WNN. For problems no. 3 and 5 the RBF performed quite poorly, while the WNN was able to obtain some results. For a small set of training points, the RBF was better than the WNN for problems no. 1, 7, 8, 9, 12 and 13, while the WNN performed better for problems no. 3, 5, and 10. The performance was practically the same for problems no. 2, 4, 6 and 11. For a large number of training points, the WNN performed better for problems no. 9 and 10, while the RBF had a better performance for problems no. 1, 3 and 13. For problems no. 2, 4, 5, 6, 7, 8, 11 and 12 the accuracy of both methods was almost the same. Figure 5 shows the comparisons of RAAE for all 13 test functions, both for RBF and WNN. Recall that for this metric, lower values are better than high values. For a scarce set of training points, the RBF performed better for problem no. 2, while the WNN was better for problems no. 2, 3 and 4. For problem no. 1, RAAE values for both of the methods were comparable.

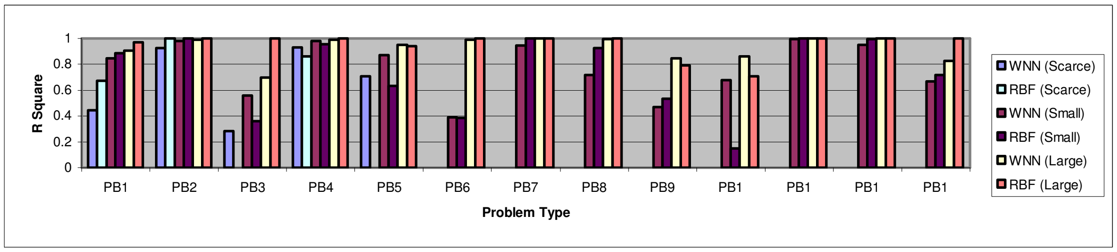

Figure 4 –R2 metric for WNN and RBF

It is interesting to note that the R2 value for problems no. 3 and 4 (see Fig. 4) were bad when the RBFs were used. However, the RAAE metrics for these two problems are better when compared to the WNN. Actually, the RAAE metric for problem no. 3 is approximately 15% greater for RBF than for WNN, while the R2 metrics for this problem was under zero for RBF. For small set of training points, the RBF performed better for problems no. 1, 2, 6, 7, 8, 9, 12 and 13, while the WNN performed better for problems no. 3, 4, 5 and 10. The values of RAAE for both methods were almost equal for problem no. 11. For a large set of training points, the RBF performed better than the WNN, except for problems no. 5 and 10. Thus, as it was observed in the R2 metric, for large and small sets of training points, the RBF was better than the WNN, while for a scarce number of training points, the WNN performed better.

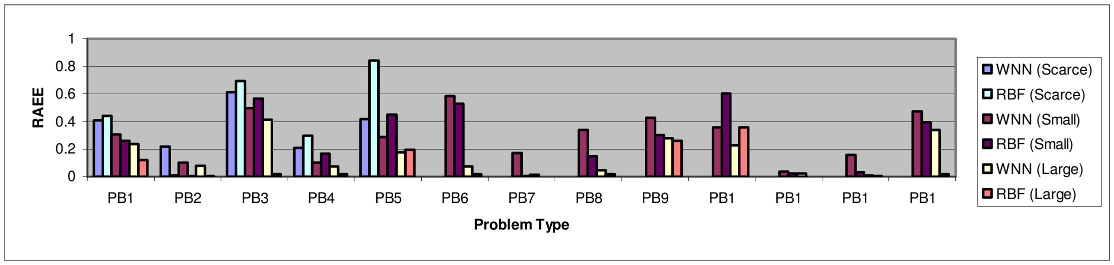

Figure 5 –RAAE metric for WNN and RBF

Finally, Fig. 6 shows the results for the relative maximum absolute error (RMAE) for all 13 test functions. A high value of the RMAE means a bad local estimate. For a scarce set of training points, the general trend is very close to the previous one, related to the RAAE metric, showing a better performance for RBF in the problem no. 2, while the WNN performed better for problems no. 3, 4 and 5. However, for problem no. 1, the RBF performed much better than WNN, indicating that the WNN had some inaccurate local estimates. For a small set of training points, the RBF performed better for problems no. 2, 7, 8 and 12, while the WNN performed better for problems no. 3, 5, 6, 10 and 13. For problems no. 1, 4, 9 and 11 the values of RMAE were close to each other. For a large set of training points, the RBF was better for problems no. 1, 2, 3, 4, 7, 8, 9, 11 and 13, while the WNN was better for the problems no. 5, 10. For problems no. 6 and 12 the performance was practically the same for both of them.

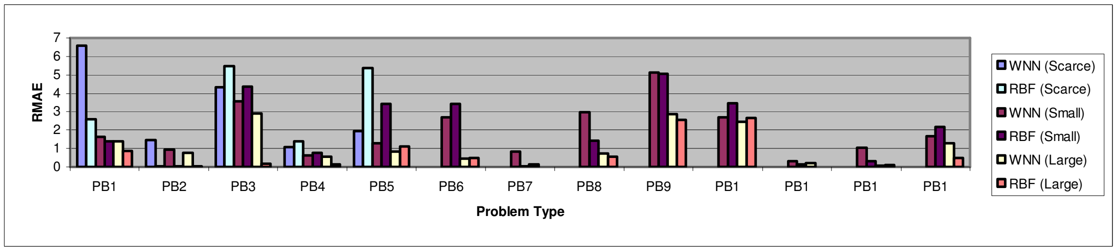

Figure 6 – RMAE metric for WNN and RBF

In order to check the robustness of the two models when noise is added, test problem no. 13 was used with several values of σ. For this test, 100 training points and 1000 testing points were used. Figure 7 shows how the R2 metric decreases when the added noise in the original function increases. It is worth to note that the RBF performed better than the WNN. In fact, even for a high level of noise, the RBF still shows a value of 0.8 for the R2 metric when the order of the polynomial is equal to M = 2. It is quite interesting that when no noise is added, the R2 metric decreases when the order of the polynomial increases, which is exactly the opposite trend to the one presented in Fig. 1 for the function number 13 without noise.

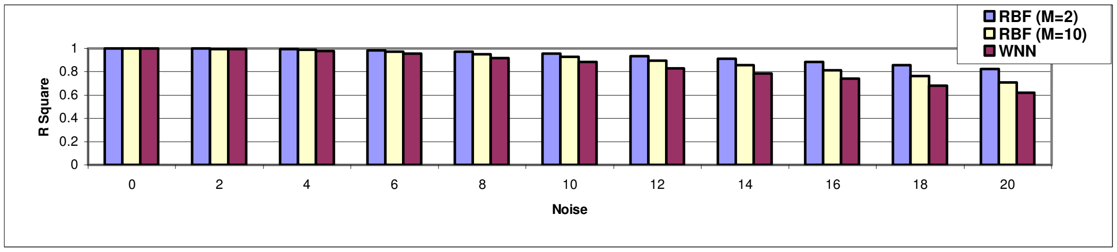

Figure 7 – Influence of noise on the R2 metric

Finally, a test case with progressively large number of variables was proposed. For this test case, test function no. 2 was chosen with 25, 100 and 400 training points and 1000 testing points for various problem dimensions. Figure 8 shows the results for the R2 metrics when the RBF was used with a polynomial of order M = 10. It is surprising that the RBF is able to maintain a high level of R2, even for a problem with 500 variables. It is worth to note that when the number of training points decreases, the value of R2 also decreases, but not significantly.

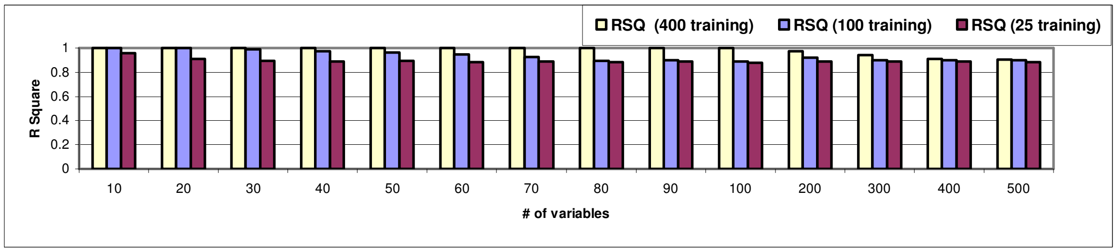

Figure 8 – R2 results for a large number of variables (RBF with M=10)

Figure 9 shows the results for the same test case presented in Fig. 8, but now for a polynomial of order M = 1. Since it was observed in Figs. 1-3, the problem no. 2 is insensitive to the order of the polynomial for scarce, small and large set of training points. It is interesting to note the similar results when a high order polynomial was used (see Fig. 8). In fact, the results are even better for 500 variables..

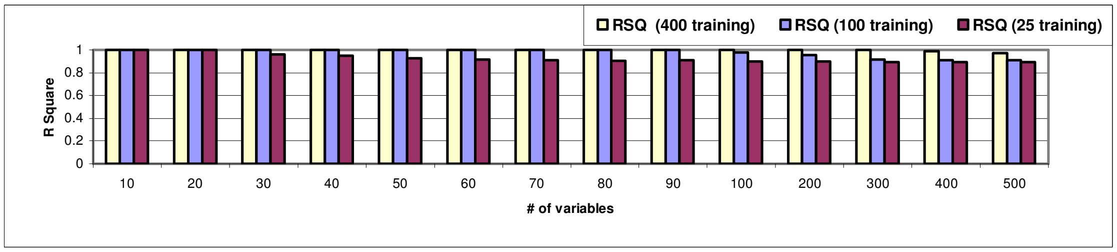

Figure 9 – R2 results for a large number of variables (RBF with M=1)

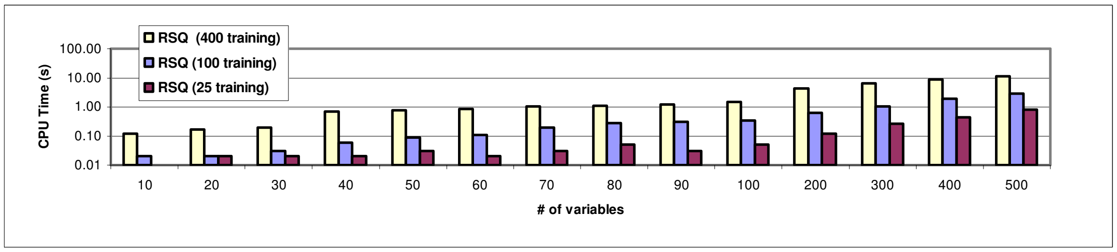

Figure 10 – CPU time for a large number of variables (RBF with M=1)

The reason for this is that the linear system resulting from the RBF approximation has [(N+M*L)+1] equations, where N is the number of training points, L is the number of variables and M is the order of the polynomial. Thus, when a high order polynomial is used, the matrix becomes too large and might become more ill-conditioned.

Finally, Fig. 10 shows the computational time required to run this test case. It is quite surprising that the computational time was lower than 10 seconds for all test cases, meaning that the RBF approximation is very fast. The code was written in Fortran 90 and the CPU was an Intel T2300 1.66Ghz (Centrino Duo) with 1Gb RAM.

Figure 11 shows the results for test problem no. 2 for a large number of variables, using the WNN. One can see that the accuracy, given by the R2 metric, decreases rapidly when using 100 training points. Also, for 400 training points, the R2 goes to a negative value for more than 100 variables, while the RBF (see Figs. 8 and 9) maintains good accuracy even when there are 500 variables.

Figure 12 shows the computational time required by the WNN where one can notice the high computational cost. In fact, for 100 variables, the time required was about 4 hours, while for 300 variables it was more than 11 hours. The code for the WNN was written in Matlab 7.0.4 and the CPU was an Intel T2300 1.66Ghz (Centrino Duo) with 1Gb RAM. Some improvement in the performance can be obtained by converting this code to Fortran90 or C++ and this should be investigated. However, the computational cost for the WNN for a problem with 300 variables and 400 training points, even with different languages (Matlab and Fortran90) was approximately 6000 times greater than for the RBF.

Finally, the same problem was run again using WNN with only one sub-net and 400 training points, using the Mexican Hat function. The results are presented in Fig. 13.

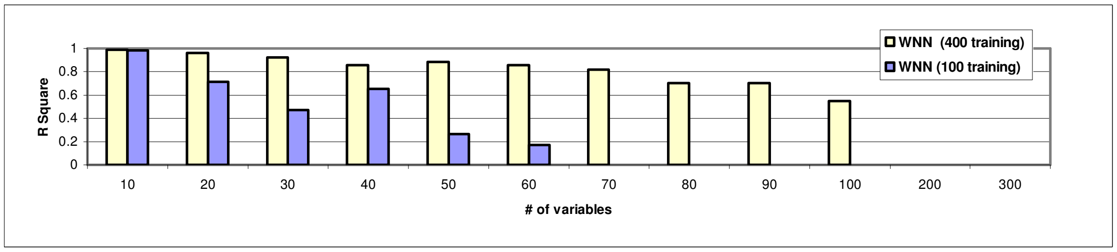

Figure 11 – R2 results with WNN for a large number of variables (WNN with five subnets)

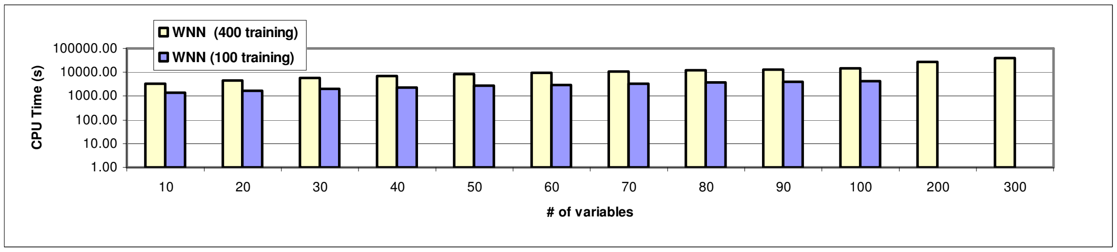

Figure 12 – CPU time for a large number of variables (WNN with five subnets)

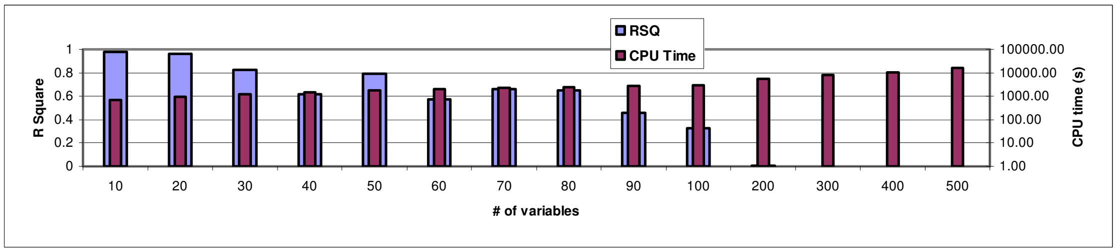

Figure 13 – R2 results and CPU time for a large number of variables (WNN with one subnet)

One can see that the computational times decrease when compared with the ones presented in Fig. 12 by a factor of five, but the R2 metrics also decreases. It is interesting to note that, again, after 100 variables, the R2 metrics goes to a negative value when the WNN is used. Thus, it is not recommended to reduce the number of neural sub-nets in order to speed-up the training process, because the accuracy goes down. In conclusion, at least for the problem no. 2, the RBF is more robust than the WNN when a very large number of variables is used.

CONCLUSIONS

In this work we performed a comparison of two interpolation techniques for highly non-linear functions where large number of variables were involved. The RBF technique seems to be quite powerfull regarding its accuracy and reduced CPU time. Even when the number of variables were as large as 500, the RBF approximation was very fast and robust. This is a promising technique for real time interpolations such as target tracking or image recognition. When only a scarce sets of data are available, WNN method typically offers higher accuracy.

ACKNOWLEDGEMENTS

This work was partially funded by CNPq (a Brazilian council for scientific development). The first author 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 2006.

REFERENCES

Buhmann, M.D., 2003, "Radial Basis Functions on Grids and Beyond", International Workshop on Meshfree Methods, Lisbon.

Hardy, R.L., 1971, "Multiquadric Equations of Topography and Other Irregular Surfaces", Journal of Geophysics Res., Vol. 176, pp. 1905-1915.

Hock, W. and Schittkowski, K., 1981, "Test Examples for Nonlinear Programming Codes", Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer.

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

Kansa, E.J., 1990, "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.

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

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

Sahoo, D. and Dulikravich, G. S., 2006, "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, Sept. 6-8.

Schittkowski, K., 1987, "More Test Examples for Nonlinear Programming", Lecture Notes in Economics and Mathematical Systems, Vol. 282, Springer.

Sobol, I. and Levitan, 1976, "The Production of Points Uniformly Distributed in a Multidimensional Cube," Preprint IPM Akad. Nauk SSSR, Number 40, Moscow, Russia.

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