Observations and Guidelines on Interpolation with Radial Basis Function Network for One Dimensional Approximation Problem

Romyaldy **Marcelo Ang Jr. ***

Department of Mechanical & Production Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260, phone (65) 874 2555
engp8939@nus.edu.sg mpeangh@nus.edu.sg (correspondence author)

Abstract

This paper reports observations on the form and behavior of the coefficient matrix involved in the training of Radial Basis Function (RBF) network for one dimensional learning (interpolation) problem. Based on these, the paper first introduces a faster way for training this particular RBF. Then it proposes a guideline on choosing the RBF spread value to ensure not only a good approximation quality, but also the least sensitivity to perturbations in training data and numerical inaccuracies during evaluation. With these results, a single dimensional approximation with RBF network becomes straightforward. Several function approximation examples are included to show the results of this proposed spread value.

Index Terms—Radial basis function networks, interpolation, toeplitz matrices, condition number.

1 Introduction

Radial Basis Function (RBF) is one of the most commonly used supervised neural networks algorithm. Compared to Multi Layer Perceptron (MLP), RBF, especially the one with Gaussian nonlinearity, is interesting mainly due to its local support property. This property enables it represent input-output mapping with more local variations and discontinuities better than MLP's globalized representation. It is even more interesting by considering properties such as faster convergence with more guaranteed optimality, ease of use and ease of interpretation thanks to huge amounts of literature and analytical frameworks already available in this subject. Unfortunately, these properties are achieved with resource-hungry characteristics of RBF and a sometime-difficult-to-solve problem (to be elaborated below) as a cost (see e.g. [1] pg. 293, [2] Ch. 6).

In this paper we mainly deal with the basic Gaussian RBF network as described e.g. in [1], and utilized in commercial software such as Matlab NN toolbox™ [2]. This kind of RBF has one hidden layer with gaussian nonlinearity and one linear output layer. The output of hidden neuron *k* given training pattern *j* is:

$$\varphi_k(x(j)) = \exp\left(-\frac{\left\|x(j)-t_k\right\|^2}{2\sigma^2}\right) \qquad (1)$$

where the t_k and σ are the Gaussian's centers and spread respectively, while ||.|| is the euclidean norm. Output from all hidden neurons are combined linearly in the output layer as:

$$y(x(j)) = \sum_{k=1}^{m} w_k\,\varphi_k(x(j)) \qquad (2)$$

In this basic architecture, the supervised RBF network learns a mapping by a strategy so-called Linear Interpolation. This is a relatively simple yet fast and powerful strategy to get a zero training error (not necessarily good generalization) compared to its counterparts such as non-linear interpolation, linear and non-linear regression. However, it is not plausible for use with a highly noisy training data.

By this strategy, the following two restrictions are imposed to the RBF algorithm:

1. Centers (t_k-s) and Spreads *(σ)* must be pre-determined prior to training to ensure linearity in the output layer.
2. Every actual output *y(j)* due to training pattern *x(j)* in (2) must theoretically have the same value with the desired output *d(j)* of training patterns. Hence, the output layer must be "solved" (interpolation problem) rather than "approached" (regression problem).

The second restriction requires the number of centers (m), hence the number of hidden layer neurons, to be equal to the number of training patterns (N). Therefore, the common approach that we consider here is by using all training patterns as the centers. As for σ, a uniform choice of σ is used for all hidden neurons, since it is enough to achieve universal approximation capability of RBF network while maintaining simplicity [3].

By the first restriction the output of the network for input *x(j)* can be written as:

$$y(x(j)) = \sum_{k=1}^{N} w_k \exp\left(-\frac{\left\|x(j)-x_k\right\|^2}{2\sigma^2}\right) \qquad (3)$$

Imposing the second restriction, the training problem to find w_k such that (3) represents the mapping problem correctly can be rewritten in the matrix form as:

$$\begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix} = \begin{bmatrix} \varphi_1(x(1)) & \varphi_2(x(1)) & \cdots & \varphi_N(x(1)) \\ \varphi_1(x(2)) & \varphi_2(x(2)) & \cdots & \varphi_N(x(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x(N)) & \varphi_2(x(N)) & \cdots & \varphi_N(x(N)) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} \qquad (4)$$

or simply as:

$$\mathbf{d} = \mathbf{\Phi}\,\mathbf{w} \qquad (5)$$

Note that, mathematically, Equation (5) is just a system of linear equations with **w** as unknowns. So the training problem is actually a process of solving linear equations (5) for **w**. Provided that could be done, we would get a unique and optimal mapping which passes training patterns exactly (zero training error theoretically).

Several difficulties arise in designing an RBF network with good overall performance. First, is how to choose σ as the approximation quality is known to be quite sensitive to this variable [4]. In [1], there is a formula suggested. Unfortunately, it is unjustified and unclear on how to use. The other proposals require significant modification on the original RBF architecture and algorithm, which make it loose its simplicity, hence absorbing even more computational resources (e.g., [6], [9]). Therefore, the common practice is through trial and error.

Secondly is how to avoid the condition number of Φ being so large. As for this ill-conditioned problem, a more difficult yet less accurate approach to solve (5) such as regularization should be taken (e.g., [7], [8]). Thirdly, provided (5) is not so ill conditioned, how to solve it efficiently. In RBF network literature, this problem is usually left to the reader to refer to general numerical linear algebra literature.

In this paper, by considering only a particular learning problem, we found that these issues can be addressed at once by analyzing the property of Φ in (5).

That particular learning problem can be formulated as: "Given a set of N different patterns $\{x(j),\ d(j)\ \in\ \Re^1\}$, find w_k in (3) or **w** in (5) such that network in (3) will map $F{:}x(j)\rightarrow d(j)$ exactly in cases where $x(j)=0,\ T,\ 2T,\ \ldots,\ (N-1)T$." Thus, it is a single input and single output learning problem, whose training cases are obtained by sampling the input space at a fixed period T.

We will start by analyzing the form of Φ in (5). As the result, it is known that it will always be in the form of a specially structured matrix called Toeplitz, with additional regularities in that it is also real, symmetric, positive definite and nonsingular. Having recognized this, we then suggest some special purposes algorithm to achieve a faster training process.

Next, we observe the condition number of Φ with respect to σ, *T*, and *N*. Based on this, we propose a guideline on choosing σ given *T* and *N*, to obtain an RBF network with good approximation quality as well as the least sensitivity to perturbations in data and numerical inaccuracies of the process.

Since the matrix concerned in this paper is highly structured, then they will be written in a condensed form as commonly used in mathematical field. For instance, an $N\times N$ matrix **A**, whose element at row j and column k is A_{jk}=(j-k), is written as: $\mathbf{A}=[\mathbf{A}_{jk}]_{j,k=0,1,\ldots,N-1}=\{(j-k),\ j,k=0,1,2,\ldots,N-1\}$.

2 First Observation: It is a Toeplitz system

In single input and single output approximation problem, the most common and sensible way of generating cases for training the network is by uniformly sampling the input space. Thus, if sampled with period T, then the training set $\{x(j),\ d(j)\ \in\ \Re^1,\ j=1,\ \ldots,\ N\}$ will contain $\{(0,d(0)),\ (T,\ d(T)),\ (2T,\ d(2T)),\ \ldots,\ ((N-1)T,\ d((N-1)T)\}$ as cases.

Following definition in (3), (4) and (5), matrix Φ can be written as:

$$\mathbf{\Phi}=\exp(-\mathbf{A}/2\sigma^2) \qquad (6)$$

where:

$$\begin{aligned}\mathbf{A}&=[\mathbf{A}_{jk}]_{j,k=0,1,\ldots,N-1}\\&=\left\{\left\|x(j)-x_k\right\|^2,\ j,k=0,1,\ \ldots,N-1\right\}\\&=\left\{(jT-kT)^2\right\}=\left\{T^2(j-k)^2\right\}\end{aligned} \qquad (7)$$

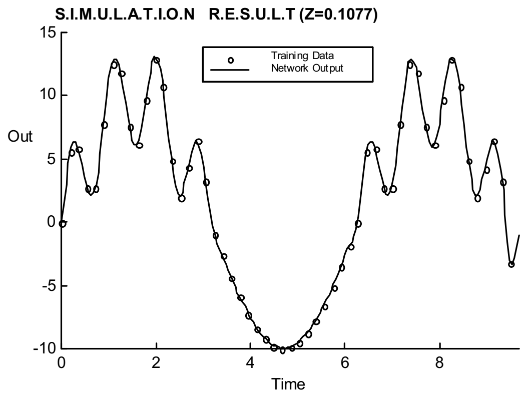
is a matrix called a distance matrix. As can be seen from the existence of component $(j-k)^2$ in (7), then matrix **A** is a matrix with a highly regular structure called symmetric Toeplitz Matrix. This type of matrix has entries that are constant along each diagonal. Hence, Φ, a Gaussian function of the distance matrix **A**, is a symmetric Toeplitz, with additional regularity in that it is always real, symmetric, positive definite, and guaranteed to be non-singular [5]. Therefore, (5) is a Toeplitz system.

This recognition should be taken advantage of to achieve a faster learning process. As for a general Toeplitz system, there are already some fast solvers for (5) available (only $O[N^2]$ floating point operations needed compared to standard methods with $O[N^3]$ complexities, see e.g. [10], [11]). By taking advantage of Φ's additional regularities further simplification may be possible.

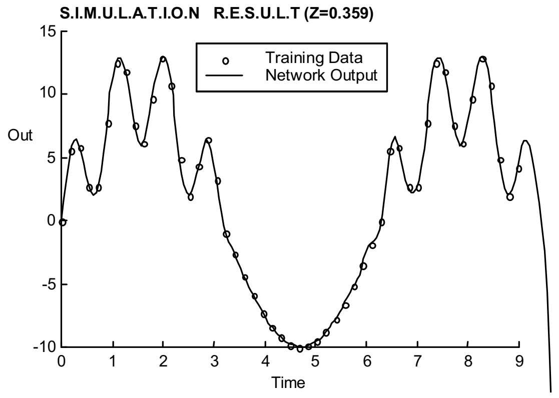
3 Second Observation: Condition number of Φ w.r.t. T, σ and N

Fig. 1 shows a one dimensional function approximation with an RBF network, trained using two different choices of spread (spread, Z or σ are used interchangeably throughout the text). Note that this sample of approximation will be used consistently throughout the text to clarify the idea presented.

The training patterns are generated by sampling the input space (in this case the time domain) at period T=0.18. The function to be approximated has a total period (*Tsim)* of 9.54 s. Therefore, 54 training cases are obtained.



(a)



(b)

Fig. 1 Approximation result for different choice of spread (Z)

The figure deceivingly suggest that both values of spread has given the same quality of approximation (indicated by low deviation between target and output of the network). Hence, they can be used interchangeably.

However, if the values of condition number (CN) of Φ involved in both cases are compared, then we see that both have totally different qualities. CNs of Φ in Fig. 1(a) and (b) are 41.7804 and $5.1746\text{x}10^8$ respectively. Recalling the definition of CN, we can infer that the network resulted in Fig 1(b) has more probability to be perturbed either by noise inherent in the training cases or by numerical inaccuracies involved in the calculation of (5).

Unfortunately, perturbation is an inevitable condition on training data. Hence, a network that is very sensitive to this is clearly not so useful in practice. Therefore, it is important for us to investigate the behavior of this CN in order to find a way to improve it.

3.1 Condition Number

As a review, condition number $k(\Phi)$ is defined as:

$$k(\mathbf{\Phi}) = k_p(\mathbf{\Phi}) = \left\|\mathbf{\Phi}\right\|_p \left\|\mathbf{\Phi}^{-1}\right\|_p \qquad (8)$$

where p represents the norm used in calculation. For example, in this paper we will present some sample simulations resulted for norm p=1 which is defined as:

$$\left\|\mathbf{\Phi}\right\|_1 = \max_{1\le r\le n} \sum_{q=1}^{n} \left|\varphi_{qr}\right| \qquad (9)$$

Note that new indexes $q=j+1$ and $r=k+1$ are used to clearly denote the index of a matrix element.

The importance of CN can be seen in:

$$\left\|\frac{\delta\mathbf{w}}{\mathbf{w}}\right\| \le k(\mathbf{\Phi}) \left\|\frac{\delta\mathbf{d}}{\mathbf{d}}\right\| \qquad (10)$$

where $\delta\mathbf{d}$ represents a change in **d** due to perturbations, while $\delta\mathbf{w}$ represents the possible changes to the solution (**w**) in (5) due to $\delta\mathbf{d}$.

Hence, one interpretation of $k(\Phi)$ is that it is a measure of how big a relative error inherent in training cases will be maximally amplified during the calculation of **w** in (5). The lowest value of $k(\Phi)$ is 1, which reflects a well-conditioned system. Large $k(\Phi)$ reflects an ill-conditioned linear system. The highest, which reflects the singularity of Φ, is infinity. Readers are referred to e.g. [12] pp. 96, and [13] pp.53 for more details on CN.

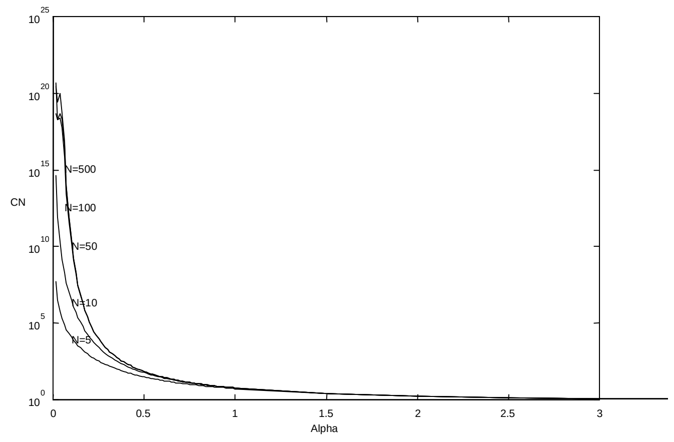
Now let us collect all external factors that could possibly affect this $k(\Phi)$. Let us rewrite (6) as:

$$\mathbf{\Phi} = \left\{ e^{-\alpha(j-k)^2} \; j,k = 0,1,...,N-1 \right\} \text{ where } \alpha = T^2/2\sigma^2 \qquad (11)$$

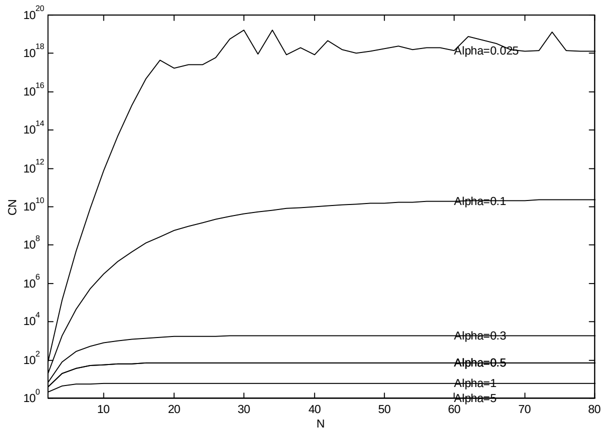
Thus, the objective now can be stated as to observe the behavior of CN of Φ due to the change of external factors, α and N.

There are several papers in mathematical field that provide an analysis to estimate the bound of CN for a certain class of function of distance matrix **A** (e.g., [5], [14]). However, none seems suitable for Gaussian function of **A** such as in our case. Hence, we rely on simulations to observe it.

To that objective, we have done quite extensive simulations to observe the behavior of CN using norms 1,2, and frobenius norm with respect to α and N. Note that since matrix Φ is symmetric then $\left\|\mathbf{\Phi}\right\|_1 = \left\|\mathbf{\Phi}\right\|_\infty$. First we do it with respect to α from α=0 to α=30 for N=5, 10, 50, 100, 500, and 1000. Next we do it with respect to N from N=0 to 500 for different α-s. Two representative plots of those simulations are displayed in Fig. 2 below by using norm 1.



(a) CN versus α plot



(b) CN versus N

Fig. 2 Behavior of CN of Φ w.r.t. α and N

Note that the non smoothness of some curves indicates that the actual value of CN at those points are so inaccurate as the matrices have already been very close to singular.

There are two motivating observations we can infer from the two figures. Fig. 2.(a) reveals that the CN will decrease monotonically w.r.t. the increase of α until it reaches its presumably asymptotic value CN=1, which means very well conditioned. This fact gives a direction of our next search for improving the CN of Φ by searching for the **highest value** of α.

Fig. 2.(b) reveals some presumably asymptotic values which acts as the upper bound of CN for each α. These asymptotic values, if it can be proven analytically, are really motivating. As they give assurance that once the asymptotic value (or the upper bound) value is known to be low enough, then however large the N (training data) used, we can always solve (5) as safe as for low number of N.

These two presumably monotonicity behaviors are newly observed, according to Micchelli[1], as literatures in the field of mathematics usually report CN only in the form of upper and lower bound. Further research needs to be set up to really prove this. The investigation is now proceeded by searching for **the highest value of α**.

By recalling that $\alpha=T^2/2\sigma^2$, there are two variables that should be considered in order to get the highest value of α, i.e. T, and σ.

The choice of T is clearly problem dependent. Lower value of T (tenth of second or even smaller) is usually required to generate training data from a possibly high-frequency system, in order to ensure that the training data is sufficient to represent the mapping of the original system. Consequently, an RBF network designer can not freely choose the value for this. In order to remove the effect of T from the choice of α, let us choose $\sigma=cT$ such that: $\alpha=1/2c^2$. So now our objective for improving CN of Φ can be further refined to finding the lowest value of spread (σ) or similarly the lowest value of c so as to obtain the highest value of α.

3.2 The Best Value of Spread

However, it is commonly known that there is a certain lower bound of σ not to be exceeded, in order that the RBF network can generalize (approximate values in between training cases). Matlab™ ([2], pg. 6-6), for instance, state this guideline as: "…make sure that the spread is large enough so that active input regions of the RBF neurons overlap enough so that RBF neuron always have fairly large output at any given moment."

Regarding generalization, there are many statistically proven methods available in literature to measure this aspect. However, for our purpose, a visual measure is considered sufficient. To be precise, we regard an RBF network producing a good generalization if the function it produces connects training cases (points) smoothly, without having too much inflection or local maxima/minima in between.

To illustrate this, Fig. 1 above has shown two approximation results with good generalization due to the use of wide enough spread values. In contrast, Fig. 3 below shows a bad generalization due to a too narrow spread (although it gives a well conditioned Φ).

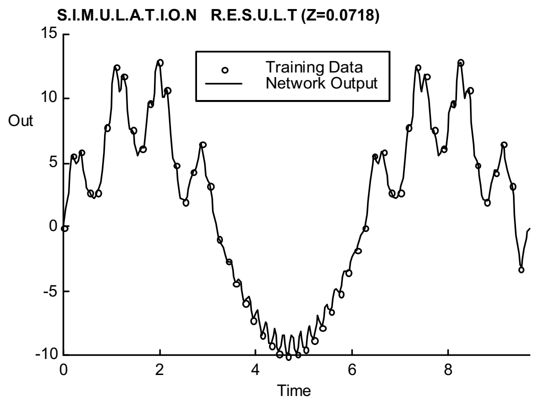


Fig. 3 Approximation result for too small choice of spread

[1] Private conversation with Charles A. Micchelli, Mathematics and Statistics Dept., University at Albany, New York.

By considering this lower bound requirement for spread, then the value of spread that we are looking for is the one at the lower bound. In other words, since $\sigma=cT$ then the value of c we are looking for is the lowest value that is enough to produce a smooth curve.

So our objective now can be refined to find **the lowest value of spread** or similarly to find the lowest value of ***c*** that is sufficient to produce a smooth curve connecting training cases. We will rely on the oncoming simulation in order to find this.

For the first simulation, recall that the output of our trained RBF network for input x is:

$$y(x)=\sum_{k=1}^{N} w_k \exp(-\frac{(x-kT)^2}{2\sigma^2}) \tag{12}$$

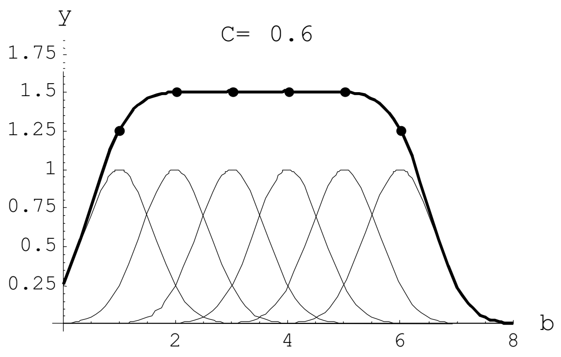
Recall that we can always regard the input x as $x=bT$ following $\sigma=cT$ such that (12) becomes:

$$y(x)=\sum_{k=1}^{N} w_k \exp(-\frac{(b-k)^2}{2c^2}) \tag{13}$$

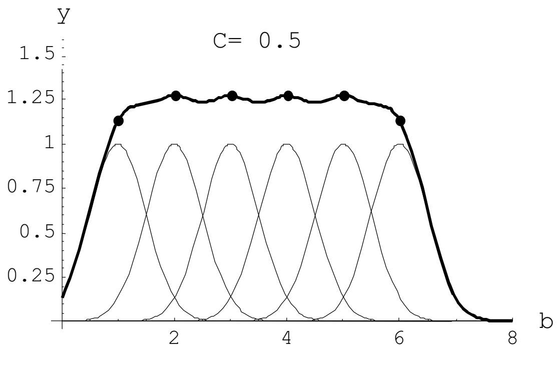
Equation (13) shows that c is inherently independent from the characteristics of training data (T). Hence, we can proceed to simulation in a search for the lowest value of c by using equation (13) instead of (12).

We choose $N=6$ (6 hidden neurons/basis functions), in order to give enough length for visual observation. Regarding this arbitrary choice of N, recall that mathematically this RBF interpolation as in (13) is a scheme of representation of a continuous function by the sum of N basis functions of $\exp(-(b-k)^2/2c^2)$, or the Gaussian functions [4]. Variable w_k determines the vertical scaling of each basis function. While c determines the spread of each basis function, which in turn determine the amount of overlapping between each basis function. Since a Gaussian function goes to zero as its input x goes away from the center (which makes it said to have a local support), then the choice of N should have no effect on the observation other than larger N eases the observation. To clarify this notion, in Fig. 4 below, the output network (bold line) is drawn along with the unscaled basis function (thin line).

We start the simulation of (13) by using $w_k=1$. Then by observing the smoothness of the output, we choose the lowest value of c that gives a smooth enough curve. The sample of this process is displayed in Fig. 4 below. Note that the dots in the bold line are the training patterns as well as the centers of the basis functions.



(a)



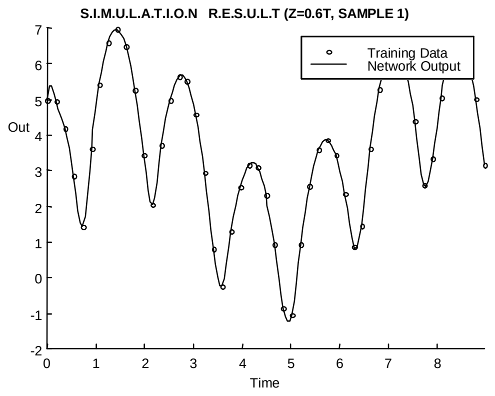
(b)

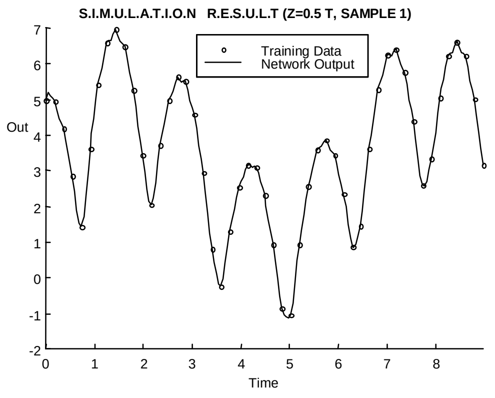
Fig. 4 Smallest value of *c* that produces good generalization

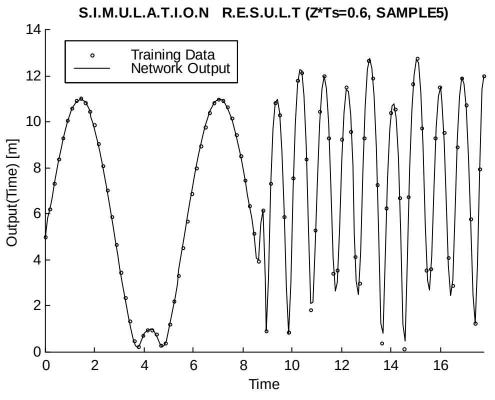
From this simulation, we found *c=0.6* to be the best candidate. We then proceed by using random values of w_k and repeat the whole experimentation. From this, we again found that *c=0.6* is still the best candidate. This result suggests the use of $\sigma=0.6\ T$ as the value of the spread we are looking for. With this choice of *c* the value of $\alpha=1/2c^2=1.39$, which according to our simulation gives asymptotic value of CN=2.9553 for N ≥18 for norm 1. This value of CN tells that this choice of *c* will always make Φ very well conditioned however large the number of training data is.

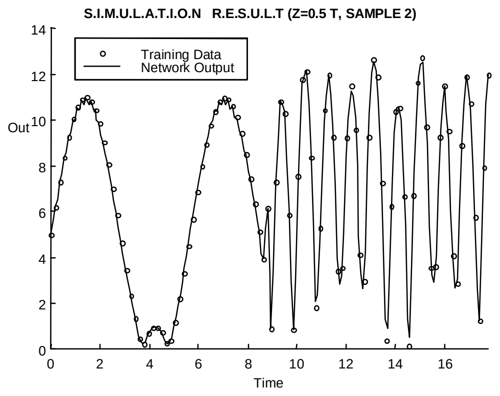
We finally proceed to test this value of σ to real approximation cases. To do this, we generate four other training sets of artificial functions besides the one we have already used. We then try to approximate these training sets using values of σ from $\sigma=0.6\ T$ to $\sigma=T$ with $0.1T$ increment.

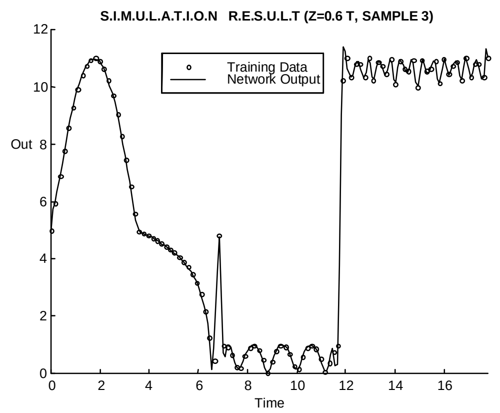
From this second simulation, it is confirmed that $\sigma=0.6\ T$ is the lowest value of spread that gives the best approximation results for these arbitrary sample problems. As it below $\sigma=0.6\ T$ the quality reduces quite significantly. In Fig. 5, we include the approximation results of three training cases using both $\sigma=0.6\ T$ and $\sigma=0.5\ T$.











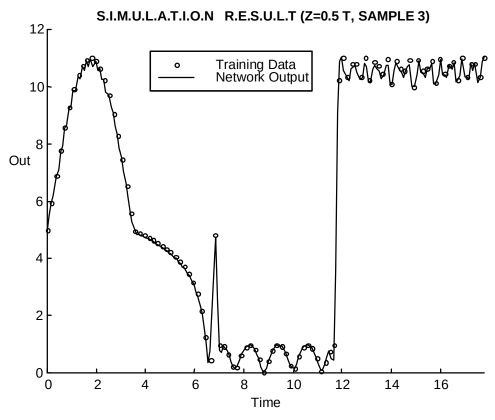


Fig. 5 Results of approximation using value $\sigma=0.6\ T$ and below $\sigma=0.6\ T$

This result confirms the previous simulation's result. Hence, in conclusion, we suggest the use of $\sigma=0.6\ T$ as the spread for this type of approximation. With this choice, the RBF network is observed to produce an optimal reconstruction of mapping from training cases.

4 Conclusions

This paper has suggested several improvements to make the learning process of a single input single output mapping fast and straight forward.

The coefficient matrix of an RBF network employed on this one-dimensional interpolation problem is observed to have a real, symmetric, positive definite and guaranteed nonsingular toeplitz structure. This special structure can and should be taken advantage of, in order to make training process much more efficient.

Relation between ill-conditioning of problem (5) with spread, T (sampling period) and N (number of training cases used) has been observed. The result suggests the use of $\sigma=0.6\ T$ to produce an RBF network with optimal approximation quality due to better conditioning of (5). With this proposal, interpolation with RBF network is already a straightforward procedure. Given a one dimensional training set with fixed sampling period, an RBF network which optimally interpolates the data can be directly constructed without the need for trial and error.

Moreover, with the previously observed asymptotic behavior of CN, this choice of spread will produce Φ with CN maximum 2.9553 regardless however large the training data is. According to (10), this ensure that perturbation in training data will be amplified maximally 2.9553 times.

The nice thing of the result presented here is that it is totally independent on the type of function or mapping problem dealt with. The only condition for its applicability is that the one dimension input space is sampled at a fixed sampling period.

Noting how our proposed value of spread is related to *T*, then Franke's conjecture that the value of spread depends on the training data seems to be strengthened ([4], pp. 191).

Further work is needed to simplify algorithm in [11] in light of additional specialty in matrix Φ and also to mathematically prove the monotonicity behavior of CN of Φ w.r.t. α and *N*.

Acknowledgements

We thank Prof. C.A. Micchelli for the inspiring discussions throughout the investigations and the writing of this paper.

References

[1] S. Haykin, *Neural Networks-A Comprehensive Foundation*, 2nd ed., NJ: Prentice Hall, 1994.

[2] Mathworks Inc., *Matlab User Guide version 3.0*, Mathworks, 1998.

[3] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," in *Neural Comput.*, vol. 3, no.4, pp.566-578, 1991.

[4] R.Franke, "Scattered Data Interpolation: Tests of Some Method", in *Mathematics of Computation*, vol. 38, issue 157, Jan 1982, pp. 181-200.

[5] B.J.C.Baxter, *The Interpolation Theory of Radial Basis Functions*, PhD Thesis, Cambridge University, UK.

[6] M.A.Kavchak, Hector M. Budman, "Adaptive neural network architectures for Nonlinear Function Estimation", in *proceedings of the American Control Conference, Pehiladephia, Pennsylvania*, June 1998, pp. 63-67.

[7] A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," *SIAM Rev.,* vol. 40, no.3, pp. 636-666, 1998.

[8] F. Girosi, M. Jones and T. Pogio, "Regularization theory and neural networks architectures," in *Neural Computation*, vol. 7, pp. 219-269, 1995.

[9] T.F. Junge, Heinz Unbehauen, "On-line identification of nonlinear time-variant systems using structurally adaptive radial basis function networks", in *Proceedings of the American Control Conference Albuquerque, New Mexico,* June 1997, pp. 1037-1041.

[10] D. Bini, "Toeplitz Matrices, Algorithms and Applications." in *ECRIM News Online Edition,* No. 22, July 1995..

[11] R. W. Freund, "A Look-Ahead Bareiss Algorithm for General Toeplitz Matrices", in *Numerische Mathematik,* Vol. 68, pp. 35--69, 1994.

[12] David S.Watkins, *Fundamentals of matrix computations,* WA: John Wiley&Sons, 1991.

[13] Philip E. Gill, W.Murray, M.H.Wright, *Numerical linear algebra and Optimization*, Vol. 1, CA:Addison-Wesley, 1991.

[14] F.J. Narcowich, J.D. Ward, "Norms of Inverses and Condition Numbers for Matrices Associated with Scattered Data" in *Journal of Approximation Theory,* 64, pp. 69-94, 1991.

[15] W. H. Press, B. P. Flannery, S. A.Teukolsky and W.T. Vetterling, "Vandermonde Matrices and Toeplitz Matrices" in *Numerical Recipes in FORTRAN: The Art of Scientific Computing,* 2nd ed., England: Cambridge University Press, pp. 82-89, 1992.