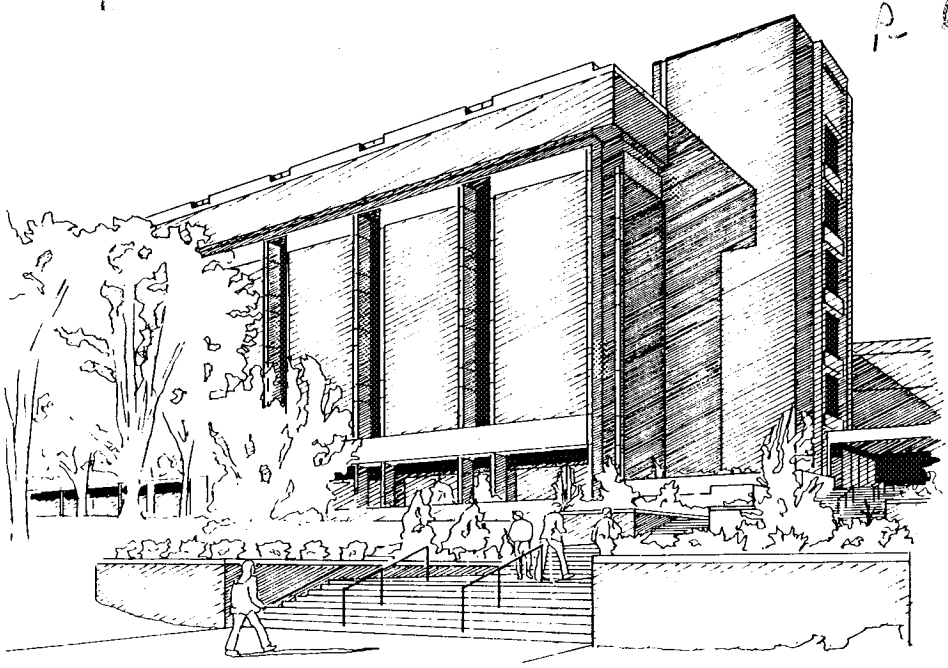


UNIVERSITY OF CINCINNATI
COLLEGE OF ENGINEERING

Radial Basis Function Neural Networks Applied to NASA SSME Data

Technical Report: TR-154/6/93/ECE

Kevin R. Wheeler and Atam P. Dhawan

This work was partially supported by grants from the NASA Lewis Research Center, and the NASA Space Engineering Research Center for System Health Management Technology at the University of Cincinnati.

Department of Electrical and Computer Engineering

Radial Basis Function Neural Networks Applied to NASA SSME Data

Technical Report # TR154/6/93 [1]

Kevin R. Wheeler and Atam P. Dhawan
Department of Electrical and Computer Engineering
University of Cincinnati
Cincinnati, Ohio 45221 ML 30

[1]This work was partially supported by grants from the NASA Lewis Research Center, and NASA-UC Space Engineering Research Center for System Health Management Technology at the University of Cincinnati.

Abstract

This paper presents a brief report on the application of Radial Basis Function Neural Networks (RBFNN) to the prediction of sensor values for fault detection and diagnosis of the Space Shuttle's Main Engines (SSME). The location of the Radial Basis Function (RBF) node centers was determined with a K-means clustering algorithm. A neighborhood operation about these center points was used to determine the variances of the individual processing nodes.

1 Introduction

In test firing and during on-line operation of the Space Shuttle's Main Engines (SSME), an efficient method of detecting anomalous sensor values is needed for detection and diagnosis of engine and sensor faults. Due to the volume of data acquired during a single test, and the fact that the nominal ranges of sensor values are dependent upon system parameters, the analysis currently consumes many man hours. To automate the analysis of anomalous sensor values, the use of Radial Basis Function Neural Networks (RBFNN) is being investigated. This report shows that by using the last five samples in time of a select group of sensor values, it is possible to have a RBFNN predict the value of a particular sensor at the next discrete instant of time. The predicted value can then be compared to the actual value. If the difference is greater then some threshold (which could be based upon the standard deviation of the data), then an anomalous sensor value has been detected.

A brief review of RBFNNs is given next, after which a description of the SSME data is presented. This is followed by a presentation of the implementations and experiments performed, and suggestions for future work.

2 Radial Basis Function Neural Networks

The basic radial basis function neural network contains an input layer for input signal distribution, a single hidden layer of processing units, and an output summation unit as shown in Figure 1.

The input vector $\vec{x}$ with components 1 to n is presented to each processing unit. Each processing unit has a centroid vector $\vec{c_i}$ which determines the location of the center of the radial basis function. The radial basis function is applied to the Euclidean distance between the input vector and its own centroid. The output of each unit is then weighted by w_i and summed together by the unit in the third layer:

$$f(\vec{x}) = \sum_{i=1}^{K} w_i \phi(\|\vec{x} - \vec{c_i}\|)$$

where ϕ is the radial basis function (typically Gaussian), $\vec{c_i}$ are the K centers, and w_i are the K weighted connections from the units to the third layer.

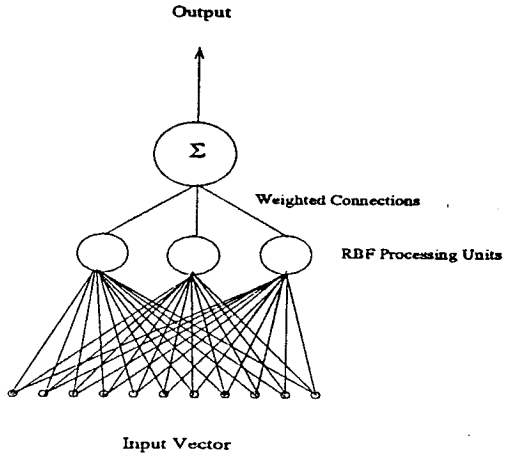


Figure 1: Radial Basis Function Neural Network

With this basic topology, the system can be represented in state space form:

$$\vec{y} = (F \cdot \vec{w})$$

where F is the matrix of activations:

$$F = \begin{pmatrix} f(\|\vec{x_1} - \vec{c_1}\|) & \cdots & f(\|\vec{x_1} - \vec{c_K}) \\ \vdots & \ddots & \vdots \\ f(\|\vec{x_N} - \vec{c_1}\|) & \cdots & f(\|\vec{x_N} - \vec{c_K}) \end{pmatrix}$$

The weights can be calculated using the Moore-Penrose inverse:

$$\vec{w} = (F^T \cdot F + \alpha I)^{-1} \cdot F^T \cdot \vec{y}$$

where α is much less than 1, and I is the identity matrix. αI is added in the event that the square matrix $F^T \cdot F$ is close to being singular. This singularity will only occur if redundant data is applied to the input of the neural system. It is also possible to calculate the weights using an iterative gradient descent algorithm.

Internal normalization can be implemented amongst the Gaussian nodes themselves. Without normalization the output of each node is calculated as:

$$\phi(\|\vec{x} - \vec{c_i}\|)$$

The following equation incorporates normalization across all of the Gaussian nodes upon presentation of each input vector:

$$\frac{\phi(\|\vec{x} - \vec{c_i}\|)}{\sum_{i=1}^{K} \phi(\|\vec{x} - \vec{c_i}\|)}$$

The normalization requires that the sum of the outputs of all nodes be 1 for any input vector. This allows for the output of smaller units to have a greater impact on the overall output. Thus when an input vector falls between two nodes, the system will be able to better interpolate [1].

The parameters which need to be established are the locations of the centroids (means), the weight values, the widths of the radial basis functions (variances), and the type of radial basis function to be used. It may be desirable to alter the topology of the network to include direct weighted connections from input to output, and to change the linear unit in layer three to use a nonlinear function. Although the modifications to the topology may reduce output error, it is no longer possible to use the simple weight calculation mentioned above, so learning time will increase.

The locations and widths of the radial basis functions depends upon the data being presented to the network. One way to find these parameters is to use a clustering algorithm such as K-means on the data. Once the data has been clustered, the centers of these clusters can be used as the centers of the Gaussian nodes. Note that this method requires some a priori information as to how many clusters should be performed. This number is not as important as one might believe. This will be demonstrated when the results of the experiments are presented.

3 Description of Data

Three sets of startup data were available, each of which contained a different number of sensor parameters. The parameters used in each list were selected in the work of [2]. The sensors I.D. numbers used in the training and recall

Table 1: Training-based Parameter Lists

Parameter List	*Number of PIDs*	*Parameters used in training*
1	6	21 58 209 734 951 1050
2	7	21 58 209 327 734 951 1058
3	8	21 52 58 209 327 734 951 1050

are depicted in Table 1. Descriptions of the sensor I.D. numbers are given in Table 2. In this table a "T" indicates that a set of vectors was used for training, a "V" indicates a set used only for validation. Only List1 was used in this report. It was the easiest of the three lists to learn in terms of the extent of required generalization necessary for adequate performance on the validation sets. List 1 uses 6 sensor values which, when combined into a single input vector representing the last five sampled sensor values for each sensor, becomes a 30 dimensional vector. The sensor value that was to be predicted for all sets is the SSME's High Pressure Oxidizer Turbine (HPOT) discharge temperature, which has Parameter Identification (PID) number 233. It is important to note that all of the data was transient startup data. This data contained many transients which varied from test firing to test firing. These transients increase the difficulty that any learning paradigm will have in accurately interpolating between the training and validation data sets. The HPOT sensor values for the four training sets are plotted against time in Figure 2, the HPOT values for the validation sets are plotted in Figure 3. From these figures it can be seen that there was a wide range of nominal sensor values.

4 Implementations and Experiments

Six different variations of RBFNNs were implemented in C and tested on List 1: RBF1, RBF11, RBF6, RBF8, RBF10 and RBF12. All simulations were executed on a SPARC IPX. These implementations mainly differ in the way that they calculate the variance associated with each Gaussian node. All of implementations used Gaussian based radial basis functions. The means of

Table 2: Parameter Descriptions

PID	*Description*
21	Main Combustion Chamber Oxidizer Injection Temperature
40	Oxidizer Preburner Oxidizer Valve Actuator Position
42	Fuel Preburner Oxidizer Valve Actuator Position
52	High Pressure Fuel Pump Discharge Pressure
58	Fuel Preburner Chamber Pressure
59	Preburner Boost Pump Discharge Pressure
209	High Pressure Oxidizer Pump Inlet Pressure
231	High Pressure Fuel Turbine Discharge Temperature
233†	High Pressure Oxidizer Turbine Discharge Temperature
327	High Pressure Oxidizer Pump Balance Cavity Pressure
480	Oxidizer Preburner Chamber Pressure
734	Low Pressure Oxidizer Pump Shaft Speed
951	High Pressure Oxidizer Pump Primary Seal Drain Pressure
1050	Oxidizer Tank Discharge Temperature
1058	Engine Oxidizer Inlet Temperature
1205	Facility Fuel Flow
1212	Facility Oxidizer Flow
O/Cs	Dummy Parameter indicating Open/Closed Loop Operation
OPBs	Dummy Parameter indicating Oxidizer Preburner Prime Time

† the modeled parameter

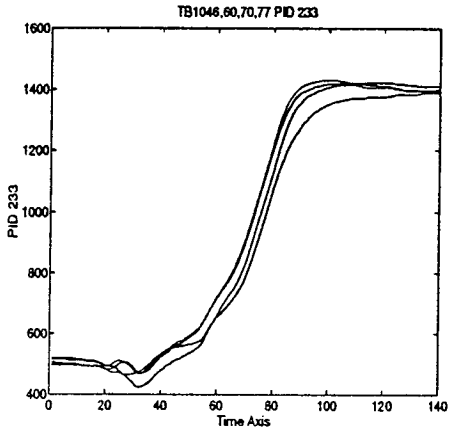


Figure 2: P.I.D. 233 Training sets

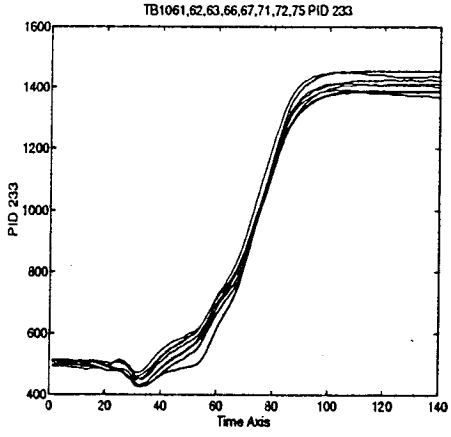


Figure 3: P.I.D. 233 Validation sets

the functions were chosen by the K-means algorithm. The number of clusters to be formed by this algorithm was selected by the user. It will be shown that the number of nodes selected was not that important as long as it fell within a reasonable range.

In the experiments, the start up data was sampled from 0.4 to 5.96 seconds at 25 hertz. This resulted in 140 vectors per set, or 560 vectors total for the 4 training sets:

$$(5.96 - 0.4) * 25 \; cycles/sec. + 1 = 140 \; vectors$$

If one Gaussian node were to be placed on every data vector, this would result in 560 nodes. The following number of nodes were used in the experiments: 28, 56, 112, 140, 560. These correspond to using 5%, 10%, 20%, and 100% of all possible nodes.

Note that if 560 nodes were used, then no data clustering was performed. 560 nodes were only used with RBF11, in order to see what error would be associated with Gaussian function interpolation on the data with fixed variance. Training occurred by presenting the four training sets to the system: B1046, B1060, B1070 and B1077. The error statistic were then generated by individually presenting each set and recording the errors. The experimental results are presented in the tables and plots at the end of this text. In the plots, both the output of the network and the desired output are plotted, next to these are plots of the associated error. Note that in the plots the solid line represents the output of the neural network, and the dashed line represents the actual value of the P.I.D. 233 sensor. All the plots used between node normalization unless otherwise noted on the plot.

All of the implementations used an iterative singular value decomposition to invert the matrix in the weight calculation except for RBF6. RBF6 used gradient descent to solve for the weights. This was implemented only for processing time considerations. All of the data had originally been mapped into the range [-0.5,0.5], although it was not necessary to have the data in the [-0.5,0.5] range. The data was then normalized to have zero mean and unit variance before it was presented to the K-means preprocessor. The data was denormalized back to the [-0.5,0.5] range before it was presented to the RBF algorithm. All algorithms randomized the order of the data vectors so that K-means could better cluster the data. The randomization made a dramatic improvement on the uniformity of the center distributions found by K-means.

RBF1 and RBF11 used a global variance parameter, in this case all nodes used a variance of 0.01. This parameter was chosen heuristically to work best with the given data. Two other forms of RBFs were created to avoid having to choose a global variance parameter, and to allow for the variance to vary with each node. In order to determine the effect of normalizing across the Gaussian nodes, RBF1 was implemented without the between node normalization, and RBF11 was implemented with the between node normalization described previously.

RBF8 used the square of the mean of the 50 nearest vectors to a centroid (the mean of a Gaussian node) for the variance of the node which belongs to that centroid.

RBF10 used the square median of the 50 nearest vectors to a centroid as the variance of the node. The size of the neighborhoods used in determining the variance was selected by trial and error. A larger size neighborhood helps reduce the effect of data point outliers. Too large a neighborhood will cause all nodes to have essentially the same variance. The performance of the system was sensitive to the size of the neighborhood.

RBF12 used the kmeans algorithm to calculate the variance of each cluster. Note that a Euclidean distance metric was used in calculating the distance of each vector in a cluster for the centroid of that cluster. Otherwise, it would have been necessary to use a covariance matrix approach.

RBF6 was the same as RBF8 except that the weights were calculated using gradient descent. The error statistics of this method will be as good as those of the algebraic method as long as enough iterations are performed. Thus, an important consideration is the processor time used in convergence.

The tables at the end of this text contain three columns of numbers: the Root Mean Square (RMS) of the error between the output of the network and the actual prediction value, the Normalized Root Mean Square Error (NRMS), and the Maximum percentage of the error (Max %). The last two columns will be used for comparison between the experiments.

The error statistics for RBF1 with global variance of 0.01 and 25% nodes (i.e. 140 nodes were used out of 560 possible nodes) without between node normalization is shown in Table 3. The Normalized Root Mean Square Error (NRMS) was always under 6%, and the maximum percent error was as high as 55%. Clearly this is unacceptable for most applications. When between node normalization was added, the greatest maximum percent error dropped to 11% as shown in Table 4.

All algorithms described from here on will incorporate between node normalization. Table 5 shows the error statistics when a node was located at every data point (560 nodes), with a global variance of 0.01. The errors were higher in this case then when 140 nodes were used. This can be explained by the fact that with a node on every data point, individual variances were needed from node to node. With less nodes than data points, the clusters may overlap without adverse affect on the function approximation capabilities. It should be noted that this conjecture is only valid for the data used in these experiments.

The greatest difficulty with the above method involves the heuristic selection of two parameters: variance, and the number of nodes. In order to avoid having to pick a variance parameter, and to provide for the ability of each cluster to have an individual variance, other methods are being investigated. The next two methods use a neighborhood operation to assign a variance to each cluster. Unfortunately, this still involves heuristic selection of a neighborhood size, and of the number of nodes to be used in the network.

RBF8 used a mean neighborhood operation to determine the variance of each node individually. To determine the variance of a node, the algorithm calculates the mean of the vectors within some neighborhood about the centroid (mean of the Gaussian) of that node. If we let N represent the size of the neighborhood, then this can be written as:

$$variance = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{dim} (x_i - c_i)^2$$

where x_i is the i^{th} component of the input vector $\vec{x}$, and dim refers to the dimension of the input vectors. Thus if N = 50, this means that the 50 input data vectors with the smallest square distance from the centroid will be used in calculating the variance for that centroid.

Table 6 shows the error statistics for Rbf8 with 140 nodes and a neighborhood size of 50. This neighborhood size was arbitarily chosen. The effects of this selection will be discussed later. The errors in this table are slightly higher than for the Rbf11 with 140 nodes in Table 4. However, the problem of selecting a suitable global variance has been traded for selecting a suitable neighborhood size. This is only a benefit if the neighborhood size selection is less sensitive than the variance parameter.

Since in Rbf8 a mean neighborhood operation was used, it is only natural

to determine if a median neighborhood operation would produce better results. Rbf10 used the median of the square of the distance to the N nearest input vectors to a centroid, to calculate the variance for that centroid. The error statistics for Rbf10 using 140 nodes with a neighborhood size of 50 is shown in Table 7. Note that the mean and median implementations produce very similar error statistics, with the mean operation being slightly favorable.

The effects of changing the neighborhood size and the effects of changing the number of nodes will now be discussed. Since both the mean and median implementations produce nearly identical results, only the mean implementation RBF8 will be used to show the results. Tables 8, 9, and 10 represent the error statistics for Rbf8 with 140 nodes and neighborhood sizes of: 5, 60, and 95. This represents 10%, 120%, and 190% of the original value of 50. Looking at both the NRMS and the maximum % error, it can be seen that the errors were lower for the training sets and higher for the validation sets. When considering only the validation sets, a neighborhood size of 5 was the best. But even with a 90% change in the value of the neighborhood size, the errors still remained low except for set B1071 with a neighborhood size of 95. It should be noted that B1071 requires the most generalization of all of the validation sets. This indicates that the selection of neighborhood size was not critical as long as it fell within some reasonable range. This is only significant if the output of RBF1 is sensitive to similar variations in the value of the global variance parameter.

Tables 11, 12, and 13 show the error statistics for the global variance (RBF11) values of 0.001, 0.012, and 0.19. This represents a variation of 10%, 120%, and 190%. Comparing these tables with Table 4 reveals that the errors don't really change that much even with a 90% change in the variance parameter.

The bottom line of this is that neither the selection of the neighborhood size nor the global variance parameter are critical as long as they fall within some reasonable range. Thus, there is no advantage to using the neighborhood operation to determine the variance.

The other heuristic to be investigated is the selection of the number of nodes in the RBFNN. Tables 14, 15, and 16 specify the error statistics for RBF8 with 28, 56, and 112 nodes with a neighborhood size of 50 respectively. This represents 5%, 10%, and 20% of the possible nodes. Compare these tables with Table 6 which had 140 nodes (25%). The error did increase when only 5% of the nodes were used, especially the maximum percent error

for set B1071. The difference in error between 20% and 25% nodes seems insignificant.

Tables 17, 18, and 19 represent the error statistics for RBF11 with global variance parameter of 0.01 with 28, 56, and 112 nodes respectively. As the variance increases the error for the training sets increases, but overall, the errors remain small. This is because as the variance increases, the overlap becomes too great for an individual set to be learned exactly. The RBF1 implementation seems to be a little more sensitive to the selection of the number of nodes than the neighborhood operation RBF8.

Another performance consideration was the computation time required by the various implementations. Since most of the algorithms were virtually the same, only RBF8 and RBF6 will be compared. RBF8 used the weight matrix inverse approach and RBF6 used gradient descent to calculate the weights. RBF8 takes approximately 225 cpu seconds whereas the gradient descent approach RBF6 takes hours when working with 112 out of 560 possible nodes.

The gradient descent approach can be greatly improved by performing only localized weight update. Therefore each epoch will involve only updating a small number of weights directly having influence on the output of the given input vector, rather than updating all of the weights. Since the error statistics for the gradient descent approach change with the number of iterations performed, and those iterations are currently too expensive to justify performing, no error statistics have been generated for the gradient descent approach.

The latest implementation (Rbf12) involved using the kmeans algorithm to determine the variance of each cluster and assigning this value to each node associated with that cluster. Since it is possible that a cluster will only contain one or two vectors, it was necessary to create an artificial lower bound on the calculated variance. In this case the lowest the variance was allowed to be was 0.01. This value was chosen so that this method could be compared to the the global variance method. In other words, the variance of each node will be 0.01 unless a larger value is required. Since it was desirable to have enough overlap between the nodes to provide for sufficient generlization, all of the calculated variances of this procedure were multiplied by a constant. The error statistics for this method are presented in 20. Plots of this appear at the end. The error statistics for this implementation are similar to those presented for Rbf11. The plots also show that the behavior of the output of the network is oscillatory about the desired value.

Timing comparisons have also been made between the two clustering algorithms: K-means and Kohonen's LVQ [3]. The two took approximately the same amount of C.P.U. time.

5 Conclusions and Future Directions

This report has shown that combination of the K-means algorithm with Radial Basis Function Neural Network with variable variance has allowed for successful sensor value prediction.

Work is being done on automating the two main heuristics of this approach: the variance of the processing nodes, and the number of nodes to use on the data. The automation is being done so that this approach may be used as part of an on-line diagnostic, fault detection system. It should be noted that the number of nodes and the value of the variance(s) was not really that important as long as it fell within a reasonable range.

Investigation of the K-means algorithm as a stand-alone predictor using vector component labeling is also proceeding. The purpose of this approach is to determine if the weight calculations can be avoided, and to form a comparison to component labeling with Kohonen's LVQ.

Two approaches that are currently under investigation are the implementation of a regularization network using Radial Basis Functions, and the implementation of a mulitplicative gaussian bar network. All of the approaches described in this paper require that the variance be the same for all of the dimensions of each node. The multiplicative gaussian bar network will allow a varying variance not only for each node, but also for each dimension of each node. It is hoped that this approach will allow the network to have a smoother prediction output. Currently, as can be seen from the plots, the output of the network tends to jump when a different group of nodes becomes active. This is bad when trying to use the system for fault detection because it forces the use of large confidence intervals. Hopefully with variable variance in each dimension, the confidence interval will be very small.

References

[1] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley Publishing Company, 1991.

[2] Charles C. Peck, Atam P. Dhawan, and Claudia M. Meyer. Selection of input variables for SSME parameter model ing using genetic algorithms and neural networks. In *Proceedings of the Fourth Annual Space System Health Ma nagement Technology Conference*, pages 104–118, Cincinnati, OH, November 1992. NASA Space Engineering Center for System Health Management Technology.

[3] Teuvo Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume vol. 78, no. 9, pages 1464–1480. IEEE, September 1990.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. Technical report, University of Illinois, Urbana, IL, December 1988.

[5] Federico Girosi and Tomaso Poggio. Networks and the best approximation property. A.i. memo no. 1164, c.b.i.p. paper no. 45, MIT Artificial Intelligence Laboratory and Center for Biological Information Processing Whitaker College, October 1989.

[6] I.R.H. Jackson. Convergence properties of radial basis functions. *Constructive Approximation*, 4:243–264, 1988.

[7] I.R.H. Jackson. Radial basis functions: a survey and new results. In D.C. Handscomb, editor, *The Mathematics of Surfaces III*, pages 115–133. Clarendon Press, Oxford, 1989.

[8] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, September 1990.

Table 3: Error Statistics from Parameter List 1 for Rbf1, 140 out of 560 possible nodes, global variance 0.01, no normalization across nodes

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	51.59860	0.05187	55.43803
B1060	T	28.21390	0.02770	43.82708
B1061	V	29.12591	0.02831	37.07641
B1062	V	39.23239	0.03863	47.86150
B1063	V	33.72927	0.03320	31.26490
B1066	V	53.21917	0.05209	46.77677
B1067	V	42.51067	0.04179	29.82455
B1070	T	22.83335	0.02182	27.27271
B1071	V	56.68070	0.05435	40.25718
B1072	V	41.90969	0.03953	38.49752
B1075	V	48.30697	0.04708	44.38711
B1077	T	36.09605	0.03450	34.11739

Table 4: Error Statistics from Parameter List 1 for Rbf11, 140 out of 560 possible nodes, global variance 0.01, with normalization across nodes.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	9.08039	0.00913	5.05390
B1060	T	15.11922	0.01484	8.18955
B1061	V	16.56278	0.01610	7.55070
B1062	V	23.88363	0.02352	8.80874
B1063	V	24.06748	0.02369	6.74517
B1066	V	28.07834	0.02748	10.30139
B1067	V	30.19188	0.02968	8.40462
B1070	T	7.80751	0.00746	4.90432
B1071	V	36.64726	0.03514	11.16738
B1072	V	22.29507	0.02103	4.61144
B1075	V	24.04426	0.02343	9.24767
B1077	T	10.94992	0.01047	5.01032

Table 5: Error Statistics from Parameter List 1 for Rbf11, 560 out of 560 possible nodes, global variance 0.01, with normalization across nodes.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	0.00721	0.00001	0.00164
B1060	T	5.83704	0.00573	0.86627
B1061	V	21.17175	0.02058	8.06793
B1062	V	26.14354	0.02574	8.81908
B1063	V	47.42858	0.04669	19.71690
B1066	V	25.07479	0.02454	10.11135
B1067	V	55.24511	0.05431	24.98962
B1070	T	4.89017	0.00467	0.71944
B1071	V	42.73459	0.04097	13.92102
B1072	V	37.87623	0.03572	16.95097
B1075	V	27.00886	0.02632	12.82674
B1077	T	7.48279	0.00715	0.96961

Table 6: Error Statistics from Parameter List 1 for Rbf8, 140 out of 560 possible nodes, neighborhood size of 50.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	2.38426	0.00240	1.39597
B1060	T	12.59777	0.01237	8.94835
B1061	V	17.01483	0.01654	10.51706
B1062	V	27.73178	0.02731	7.73836
B1063	V	36.67299	0.03610	8.43292
B1066	V	30.88582	0.03023	9.61599
B1067	V	35.97727	0.03537	8.59062
B1070	T	7.00646	0.00670	4.22006
B1071	V	38.58204	0.03699	13.70739
B1072	V	23.79091	0.02244	7.77105
B1075	V	30.23295	0.02946	12.56111
B1077	T	9.82221	0.00939	7.98302

Table 7: Error Statistics from Parameter List 1 for Rbf10, 140 out of 560 possible nodes, neighborhood size of 50.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	2.83764	0.00285	2.27082
B1060	T	13.01944	0.01278	10.44688
B1061	V	16.99191	0.01652	8.35701
B1062	V	27.63844	0.02721	10.59826
B1063	V	35.93978	0.03538	7.85795
B1066	V	30.68942	0.03004	8.84875
B1067	V	34.05498	0.03348	7.91770
B1070	T	7.27333	0.00695	4.86389
B1071	V	40.26593	0.03861	12.87269
B1072	V	23.99103	0.02263	7.29484
B1075	V	31.83439	0.03102	14.72255
B1077	T	9.86484	0.00943	6.64400

Table 8: Error Statistics from Parameter List 1 for Rbf8, 140 out of 560 possible nodes, neighborhood size of 5

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	12.59036	0.01266	5.91643
B1060	T	18.86700	0.01852	8.26020
B1061	V	15.54236	0.01511	8.09296
B1062	V	23.84036	0.02347	7.95695
B1063	V	29.50691	0.02905	6.01330
B1066	V	27.19326	0.02661	8.17823
B1067	V	27.51375	0.02705	6.54893
B1070	T	12.13479	0.01160	4.74145
B1071	V	37.65979	0.03611	14.24448
B1072	V	20.79559	0.01961	6.86008
B1075	V	25.46773	0.02482	9.03917
B1077	T	11.67591	0.01116	5.18719

Table 9: Error Statistics from Parameter List 1 for Rbf8, 140 out of 560 possible nodes, neighborhood size of 60

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	1.99468	0.00201	1.47569
B1060	T	12.11123	0.01189	7.26255
B1061	V	16.98131	0.01651	9.71268
B1062	V	26.68269	0.02627	8.88408
B1063	V	35.55722	0.03500	8.21124
B1066	V	31.38819	0.03072	9.40951
B1067	V	39.94933	0.03927	9.00299
B1070	T	6.99072	0.00668	4.03056
B1071	V	39.72683	0.03809	14.84076
B1072	V	25.54100	0.02409	7.60504
B1075	V	27.51913	0.02682	10.99728
B1077	T	9.43180	0.00902	5.93759

Table 10: Error Statistics from Parameter List 1 for Rbf8, 140 out of 560 possible nodes, neighborhood size of 95

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	2.52465	0.00254	1.01239
B1060	T	5.89194	0.00578	2.65420
B1061	V	14.11111	0.01372	8.36682
B1062	V	25.79183	0.02540	14.42955
B1063	V	32.26750	0.03177	14.87512
B1066	V	28.31700	0.02771	15.87984
B1067	V	40.62741	0.03994	11.26816
B1070	T	5.18299	0.00495	2.84313
B1071	V	65.97996	0.06326	32.48063
B1072	V	39.38403	0.03715	21.43067
B1075	V	20.39765	0.01988	9.73600
B1077	T	5.62804	0.00538	2.67773

Table 11: Error Statistics from Parameter List 1 for Rbf11, 140 out of 560 possible nodes, global variance 0.001, with normalization across nodes.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	17.39792	0.01749	9.41842
B1060	T	19.24398	0.01889	8.72347
B1061	V	14.86860	0.01445	9.14472
B1062	V	22.17325	0.02183	7.69661
B1063	V	25.62952	0.02523	6.37022
B1066	V	24.79422	0.02427	8.00535
B1067	V	25.81279	0.02538	6.63005
B1070	T	12.97352	0.01240	5.38796
B1071	V	39.82024	0.03818	15.11610
B1072	V	26.84831	0.02532	6.29536
B1075	V	25.14532	0.02450	9.98049
B1077	T	11.28997	0.01079	5.32295

Table 12: Error Statistics from Parameter List 1 for Rbf11, 140 out of 560 possible nodes, global variance 0.012, with normalization across nodes.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	7.53711	0.00758	4.02564
B1060	T	14.65475	0.01439	8.15110
B1061	V	16.48162	0.01602	7.53241
B1062	V	24.26593	0.02389	8.67870
B1063	V	25.48573	0.02509	8.54002
B1066	V	27.90006	0.02731	10.27619
B1067	V	31.02996	0.03050	8.69295
B1070	T	7.65893	0.00732	4.90027
B1071	V	36.54959	0.03504	10.38719
B1072	V	23.02699	0.02172	4.71767
B1075	V	23.63481	0.02303	9.14347
B1077	T	10.89041	0.01041	4.90642

Table 13: Error Statistics from Parameter List 1 for Rbf11, 140 out of 560 possible nodes, global variance 0.019, with normalization across nodes.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	3.77056	0.00379	1.80409
B1060	T	13.21858	0.01298	7.99314
B1061	V	18.48718	0.01797	7.70497
B1062	V	26.25988	0.02586	8.18643
B1063	V	29.54284	0.02908	11.77689
B1066	V	26.13952	0.02558	9.94572
B1067	V	31.32031	0.03079	9.47755
B1070	T	6.90393	0.00660	4.77293
B1071	V	37.59862	0.03605	9.40809
B1072	V	29.65932	0.02797	7.29677
B1075	V	24.94946	0.02431	9.01241
B1077	T	10.57636	0.01011	5.07506

Table 14: Error Statistics from Parameter List 1 for Rbf8, 28 out of 560 possible nodes, neighborhood size of 50.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	14.41434	0.01449	9.00289
B1060	T	28.22631	0.02771	15.22439
B1061	V	20.45557	0.01988	15.12137
B1062	V	27.14355	0.02673	13.25370
B1063	V	31.14893	0.03066	7.27571
B1066	V	25.22222	0.02469	9.23387
B1067	V	31.01706	0.03049	6.86656
B1070	T	18.86116	0.01803	6.47004
B1071	V	42.92448	0.04116	23.21731
B1072	V	25.89541	0.02442	7.66308
B1075	V	29.25733	0.02851	14.23219
B1077	T	14.43709	0.01380	7.00140

Table 15: Error Statistics from Parameter List 1 for Rbf8, 56 out of 560 possible nodes, neighborhood size of 50

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	4.45642	0.00448	2.73107
B1060	T	19.72829	0.01937	8.06534
B1061	V	13.82834	0.01344	7.98474
B1062	V	27.61247	0.02719	6.78202
B1063	V	30.51164	0.03004	5.60983
B1066	V	27.69112	0.02710	7.22070
B1067	V	28.52224	0.02804	6.90655
B1070	T	9.74294	0.00931	4.63169
B1071	V	36.70488	0.03519	11.93740
B1072	V	19.51650	0.01841	4.33229
B1075	V	23.80098	0.02319	8.70032
B1077	T	13.61962	0.01302	4.74346

Table 16: Error Statistics from Parameter List 1 for Rbf8, 112 out of 560 possible nodes, neighborhood size of 50.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	2.83689	0.00285	1.87113
B1060	T	15.10454	0.01483	7.26094
B1061	V	15.33540	0.01491	9.99841
B1062	V	27.02377	0.02661	7.66580
B1063	V	31.69655	0.03120	6.39428
B1066	V	30.97042	0.03031	9.15610
B1067	V	30.84572	0.03032	7.25803
B1070	T	7.02387	0.00671	4.29946
B1071	V	36.32853	0.03483	11.16077
B1072	V	19.99030	0.01885	7.18022
B1075	V	20.72743	0.02020	9.70741
B1077	T	13.02576	0.01245	7.72375

Table 17: Error Statistics from Parameter List 1 for Rbf11, 28 out of 560 possible nodes, global variance 0.01, with normalization across nodes.

Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	27.18175	0.02732	15.67179
B1060	T	25.14493	0.02468	11.02959
B1061	V	16.65054	0.01619	10.56191
B1062	V	24.08476	0.02372	8.69066
B1063	V	30.18663	0.02972	6.02264
B1066	V	22.65658	0.02217	6.00475
B1067	V	30.09099	0.02958	6.73048
B1070	T	18.25288	0.01745	6.61461
B1071	V	39.08114	0.03747	15.32269
B1072	V	22.67000	0.02138	5.39135
B1075	V	26.91294	0.02623	11.02567
B1077	T	12.99691	0.01242	5.55255

Table 18: Error Statistics from Parameter List 1 for Rbf11, 56 out of 560 possible nodes, global variance 0.01, with normalization across nodes.

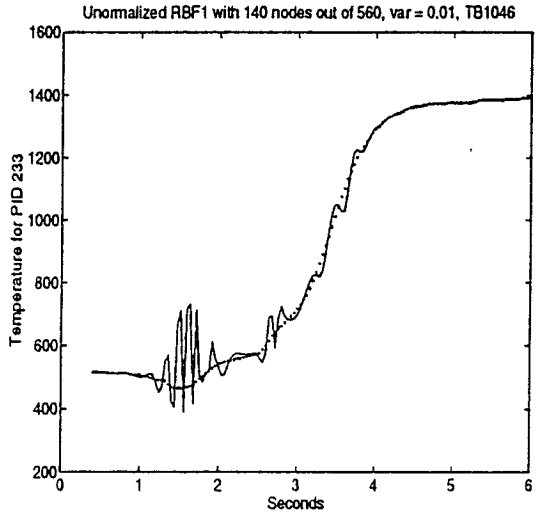
Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	16.89184	0.01698	9.67346
B1060	T	22.36876	0.02196	10.31860
B1061	V	14.56266	0.01416	10.03354
B1062	V	20.76527	0.02045	8.12575
B1063	V	23.57223	0.02321	5.79424
B1066	V	21.93067	0.02146	5.90312
B1067	V	25.39526	0.02497	6.24095
B1070	T	15.45565	0.01477	5.73375
B1071	V	40.40978	0.03875	13.09041
B1072	V	23.68318	0.02234	5.45226
B1075	V	24.96156	0.02433	10.16316
B1077	T	11.03639	0.01055	5.47923

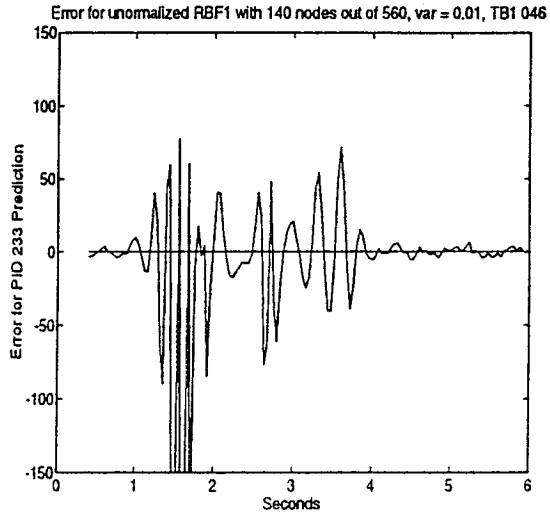
Table 19: Error Statistics from Parameter List 1 for Rbf11, 112 out of 560 possible nodes, global variance 0.01, with normalization across nodes.

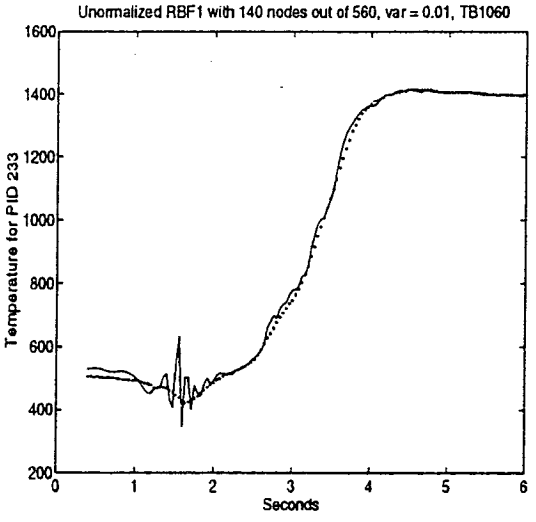
Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	9.19158	0.00924	5.07136
B1060	T	18.42548	0.01809	8.74580
B1061	V	14.63324	0.01422	7.68326
B1062	V	22.38574	0.02204	7.07681
B1063	V	22.37034	0.02202	5.83572
B1066	V	24.51416	0.02399	7.59422
B1067	V	27.92617	0.02745	7.61870
B1070	T	9.62823	0.00920	5.21438
B1071	V	37.67616	0.03612	11.96297
B1072	V	26.63420	0.02512	4.70912
B1075	V	25.96042	0.02530	9.57053
B1077	T	12.05393	0.01152	4.43430

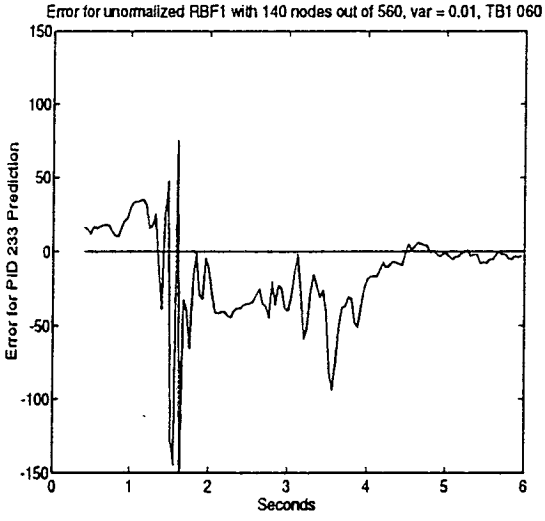
Table 20: Error Statistics from Parameter List 1 for Rbf12, 140 out of 560 possible nodes

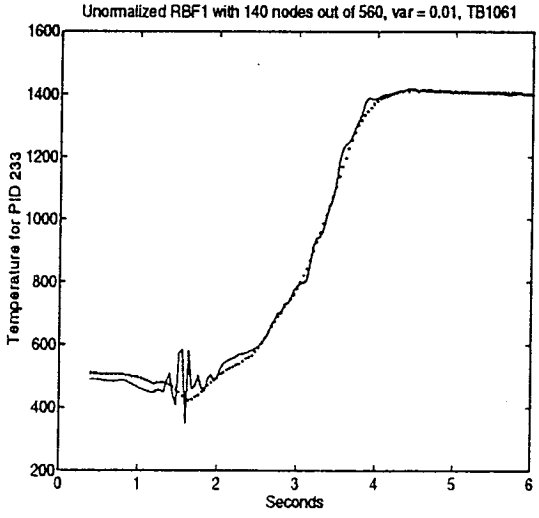
Test Firing	*Training/ Validation*	*RMS*	*NRMS*	*Max. % Error*
B1046	T	6.01873	0.00605	3.15011
B1060	T	14.86046	0.01459	7.73349
B1061	V	16.48907	0.01603	7.48909
B1062	V	23.49133	0.02313	8.62899
B1063	V	23.95228	0.02358	6.73804
B1066	V	28.18141	0.02758	9.58983
B1067	V	30.06580	0.02956	8.39829
B1070	T	7.49930	0.00717	4.29970
B1071	V	36.89098	0.03537	11.30262
B1072	V	22.69011	0.02140	4.92904
B1075	V	24.03571	0.02342	9.16707
B1077	T	11.20852	0.01071	6.01282

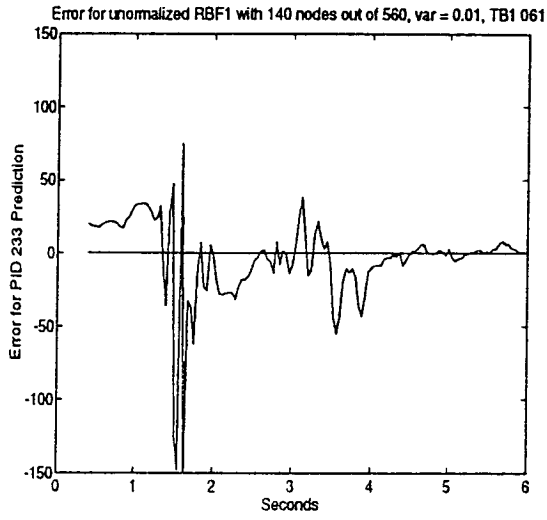

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1046
Temperature for PID 233
Seconds

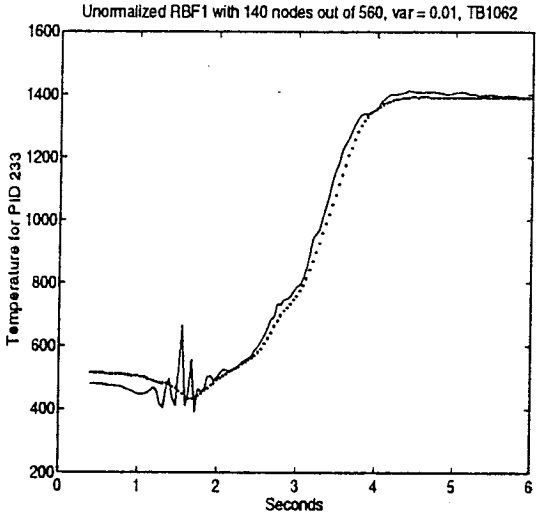

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 046
Error for PID 233 Prediction
Seconds

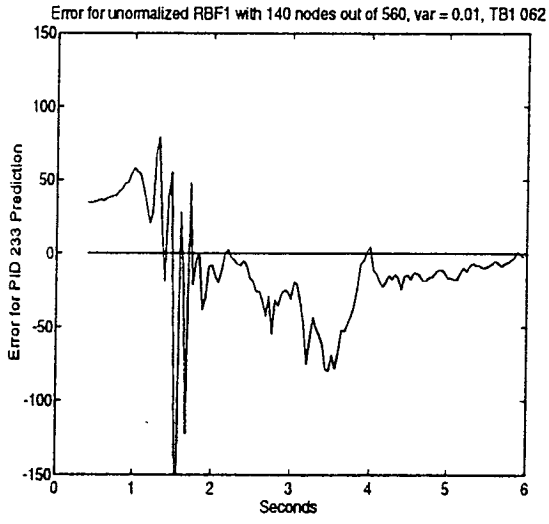

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1060
Temperature for PID 233
Seconds

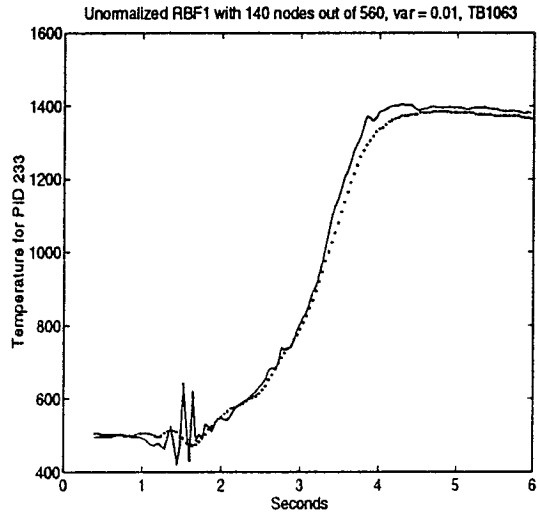

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 060
Error for PID 233 Prediction
Seconds

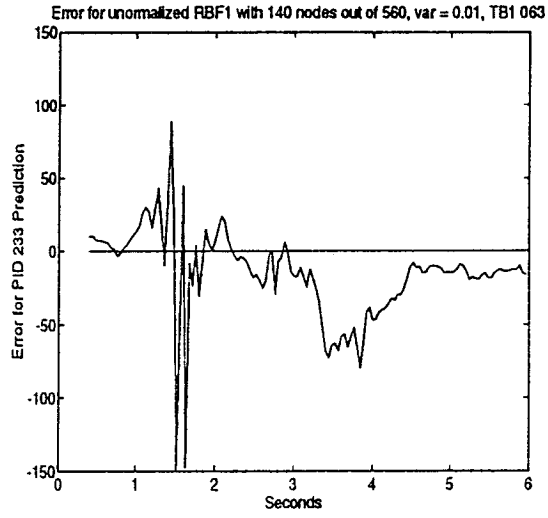

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1061
Temperature for PID 233
Seconds

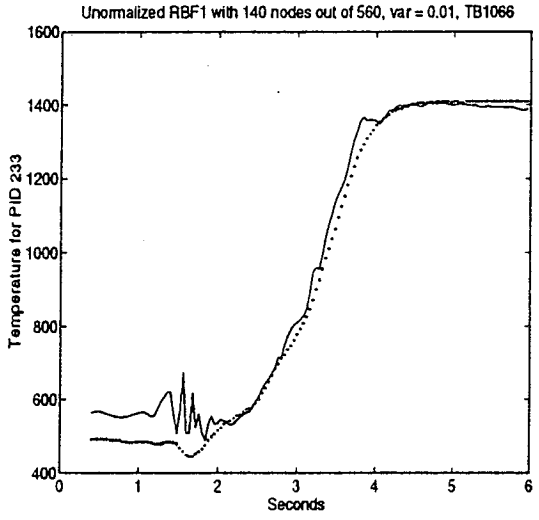

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 061
Error for PID 233 Prediction
Seconds

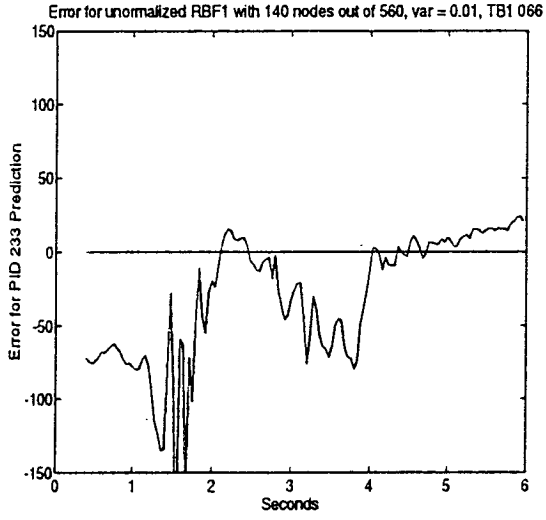

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1062
Temperature for PID 233
Seconds

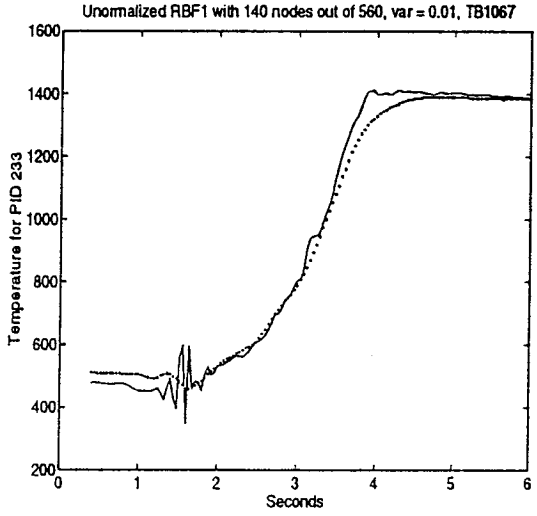

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 062
Error for PID 233 Prediction
Seconds

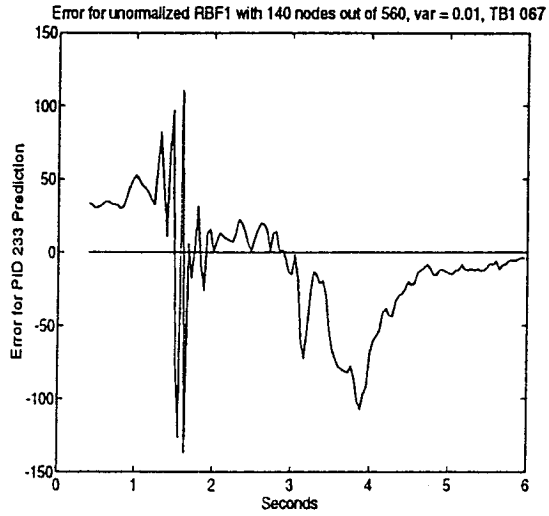

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1063
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

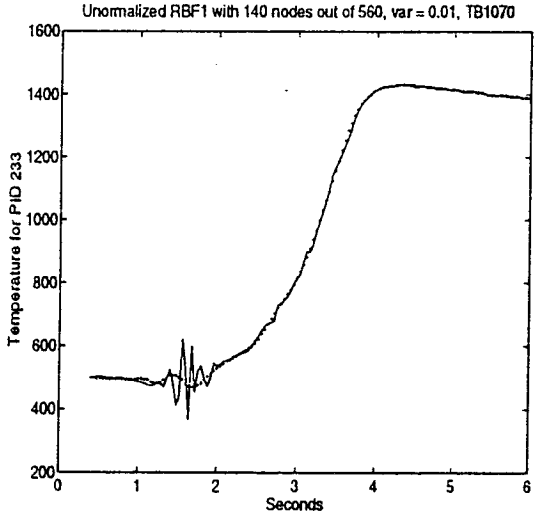

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 063
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

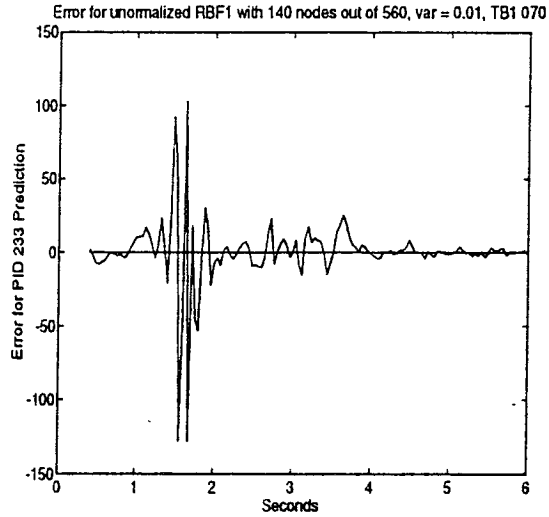

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1066
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

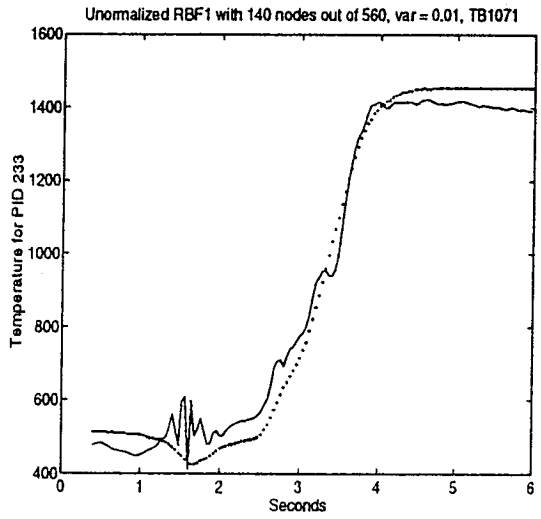

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 066
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

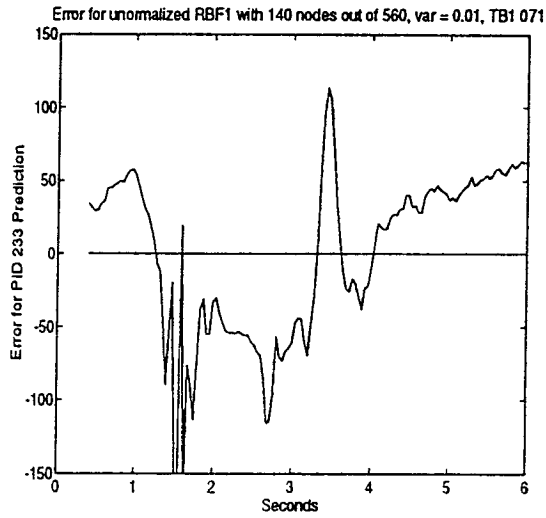

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1067
Temperature for PID 233
Seconds

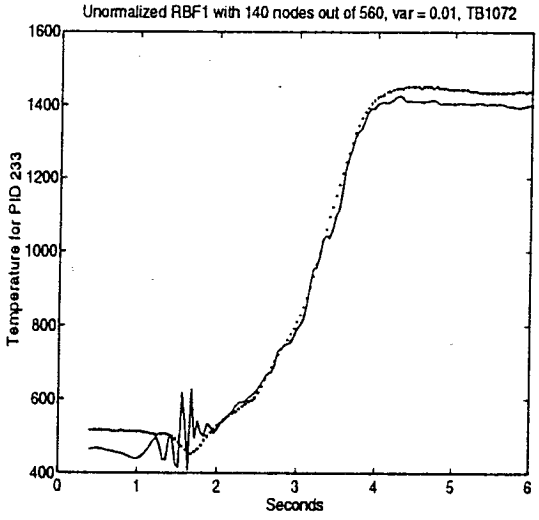

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 067
Error for PID 233 Prediction
Seconds

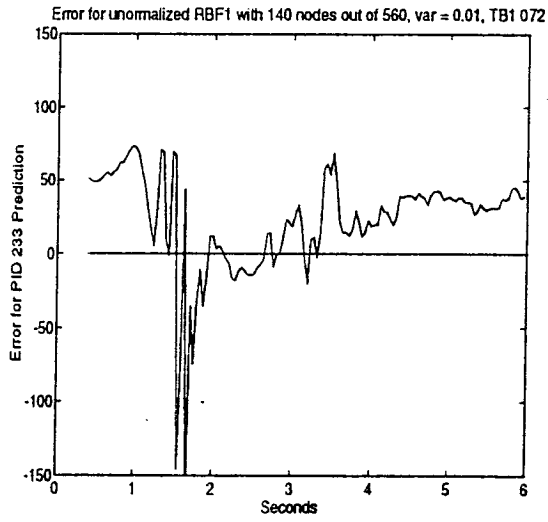

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1070
Temperature for PID 233
Seconds

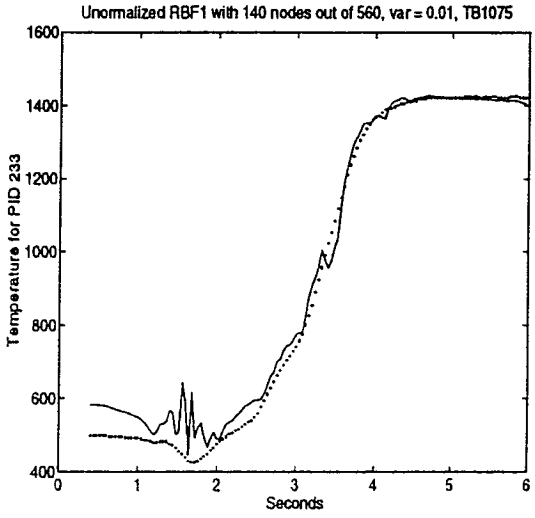

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 070
Error for PID 233 Prediction
Seconds

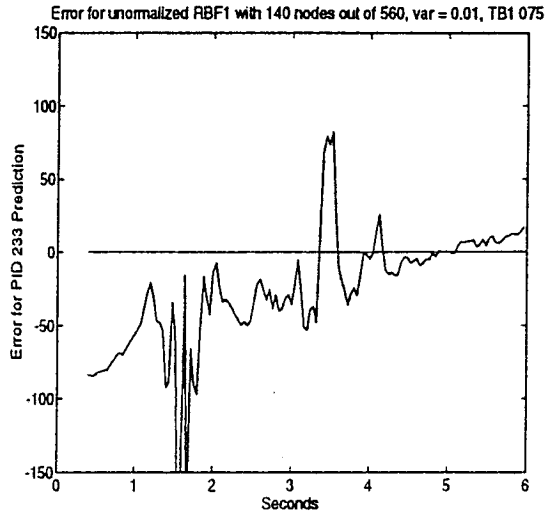

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1071
Temperature for PID 233
Seconds

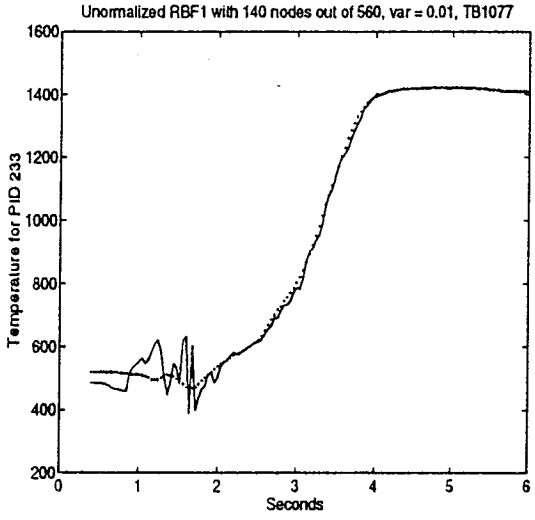

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 071
Error for PID 233 Prediction
Seconds

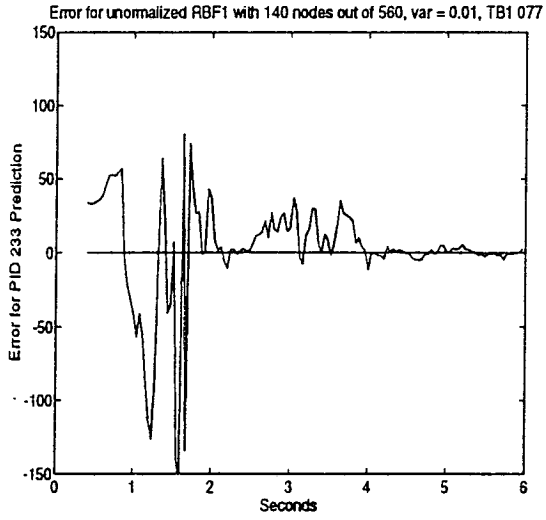

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1072
Temperature for PID 233
Seconds

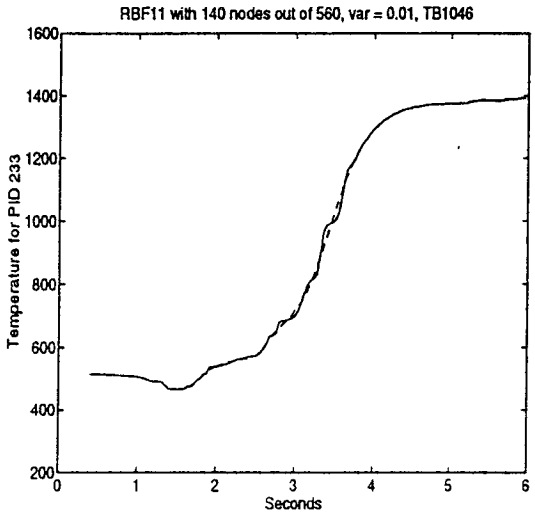

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 072
Error for PID 233 Prediction
Seconds

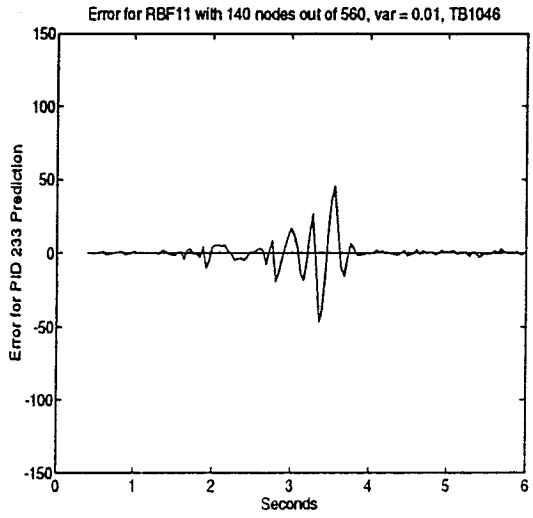

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1075
Temperature for PID 233
Seconds

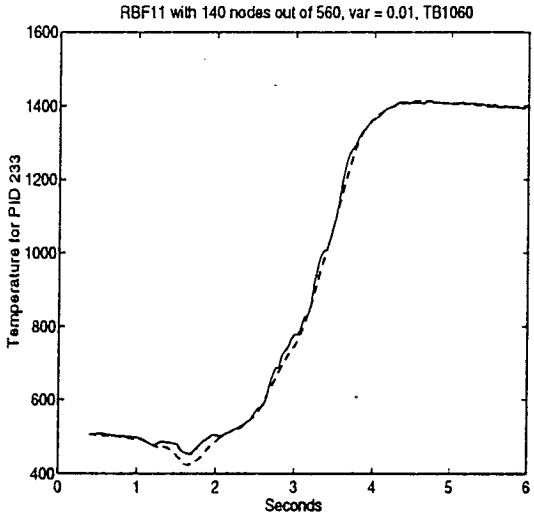

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 075
Error for PID 233 Prediction
Seconds

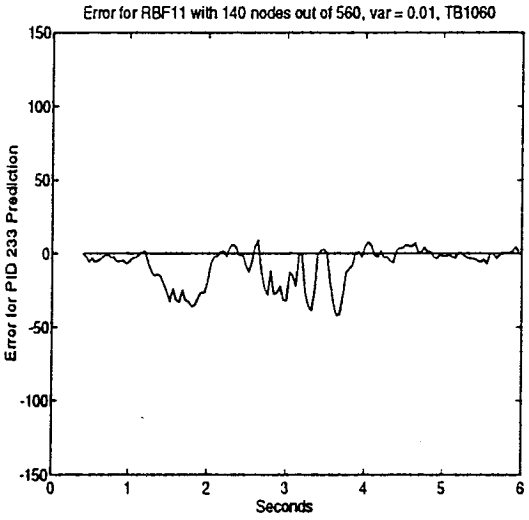

Unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1077
Temperature for PID 233
Seconds

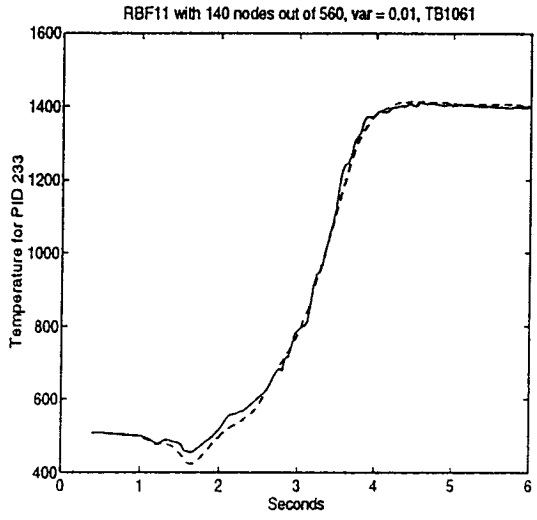

Error for unormalized RBF1 with 140 nodes out of 560, var = 0.01, TB1 077
Error for PID 233 Prediction
Seconds

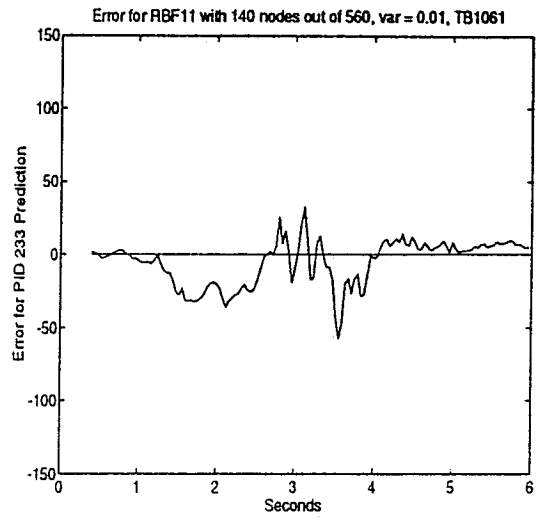

RBF11 with 140 nodes out of 560, var = 0.01, TB1046
Temperature for PID 233
Seconds

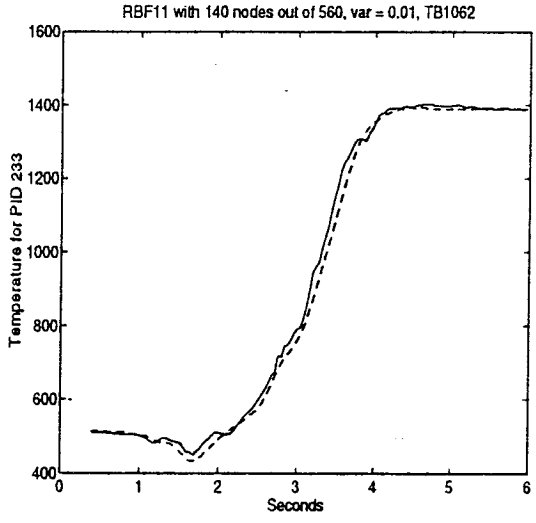

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1046
Error for PID 233 Prediction
Seconds

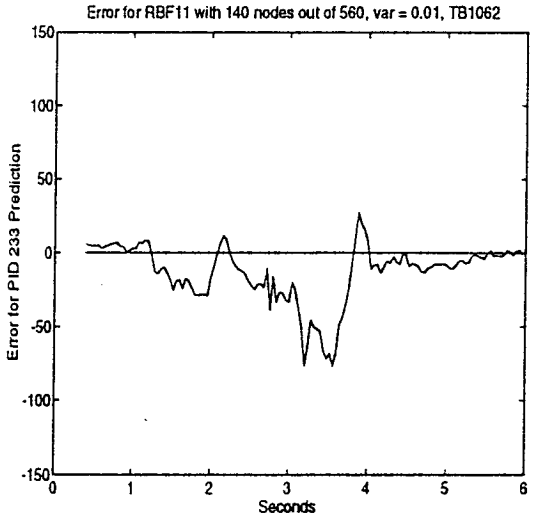

RBF11 with 140 nodes out of 560, var = 0.01, TB1060
Temperature for PID 233
Seconds

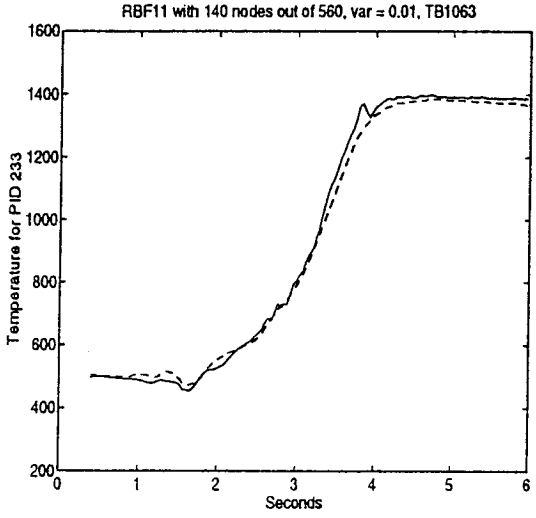

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1060
Error for PID 233 Prediction
Seconds

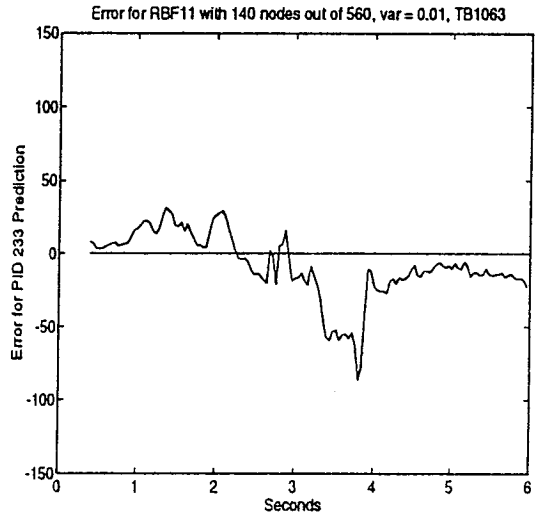

RBF11 with 140 nodes out of 560, var = 0.01, TB1061
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

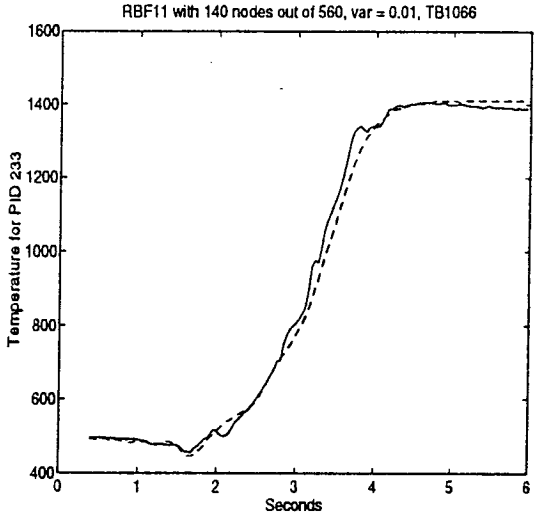

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1061
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

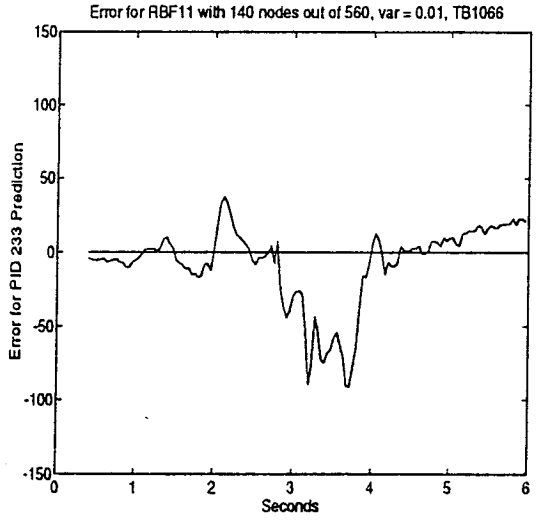

RBF11 with 140 nodes out of 560, var = 0.01, TB1062
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

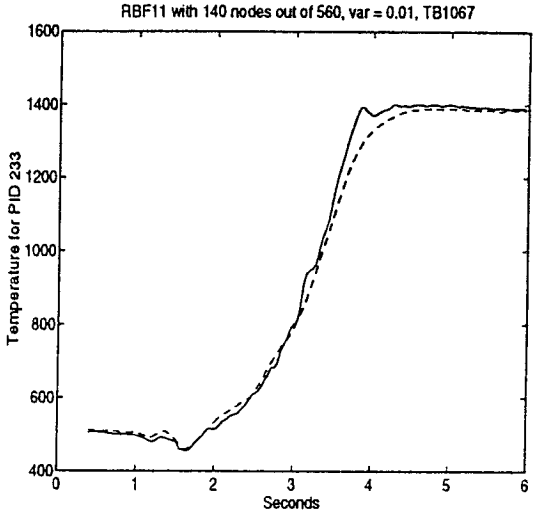

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1062
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

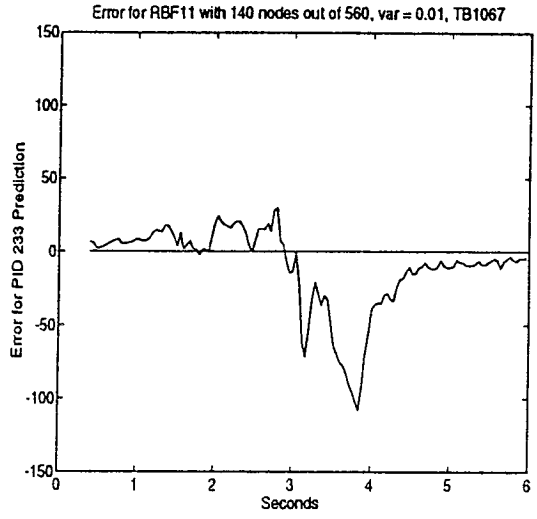

RBF11 with 140 nodes out of 560, var = 0.01, TB1063
Temperature for PID 233
Seconds

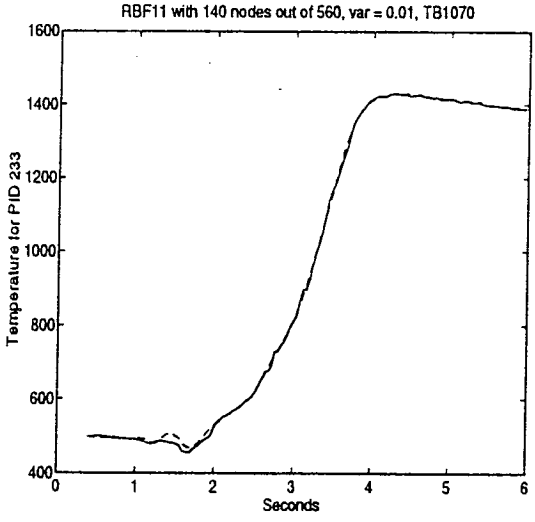

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1063
Error for PID 233 Prediction
Seconds

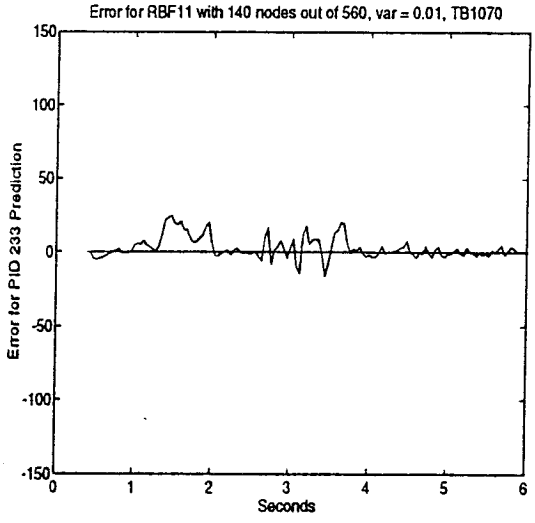

RBF11 with 140 nodes out of 560, var = 0.01, TB1066
Temperature for PID 233
Seconds

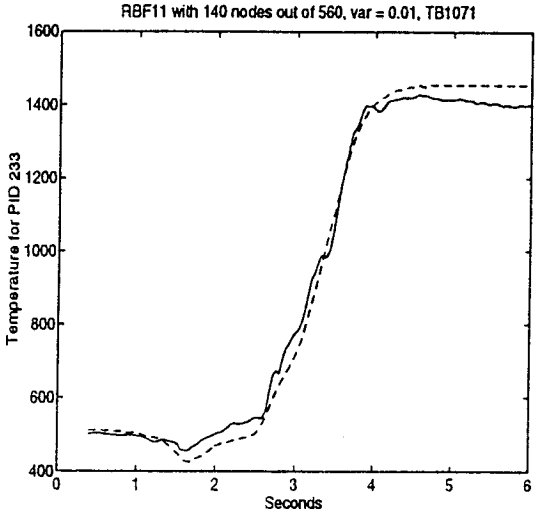

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1066
Error for PID 233 Prediction
Seconds

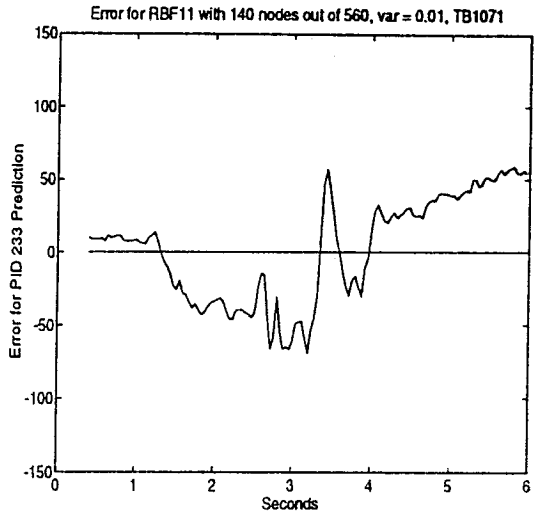

RBF11 with 140 nodes out of 560, var = 0.01, TB1067
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

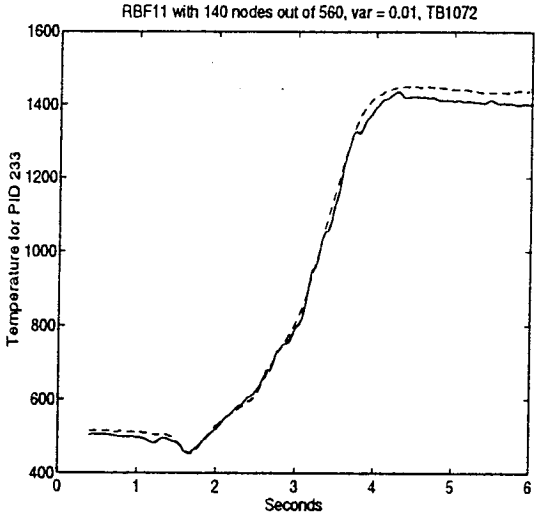

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1067
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

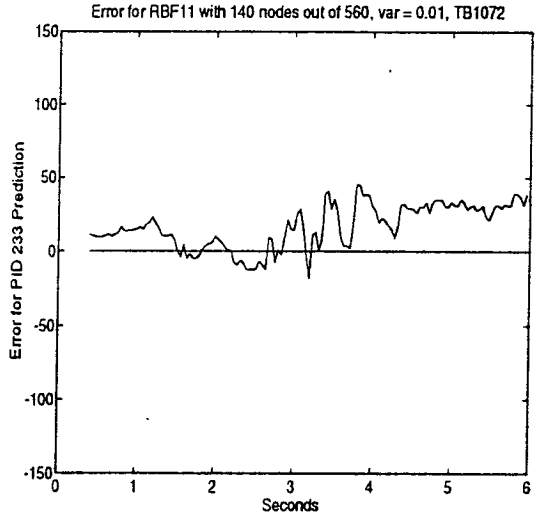

RBF11 with 140 nodes out of 560, var = 0.01, TB1070
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

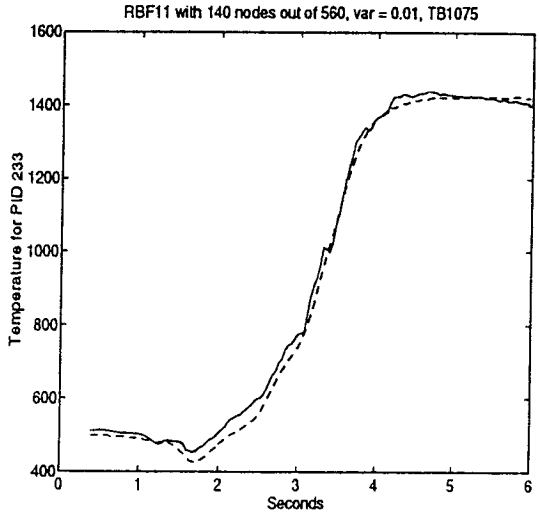

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1070
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

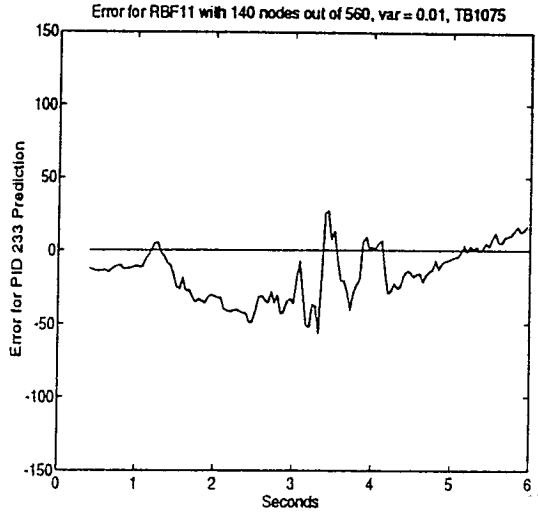

RBF11 with 140 nodes out of 560, var = 0.01, TB1071
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

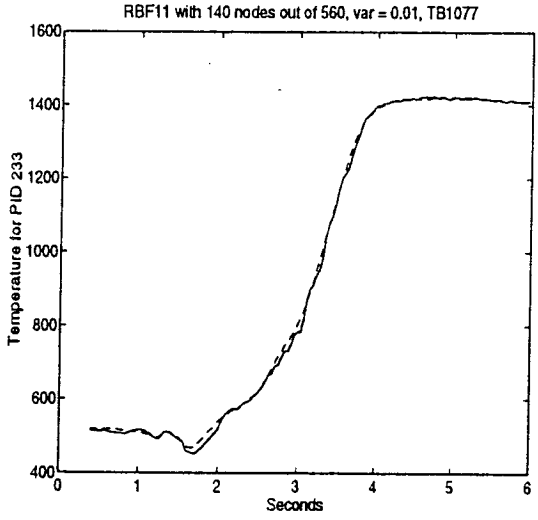

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1071
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

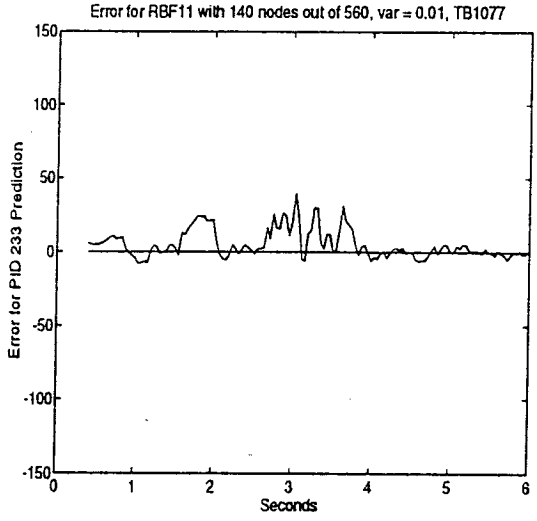

RBF11 with 140 nodes out of 560, var = 0.01, TB1072
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

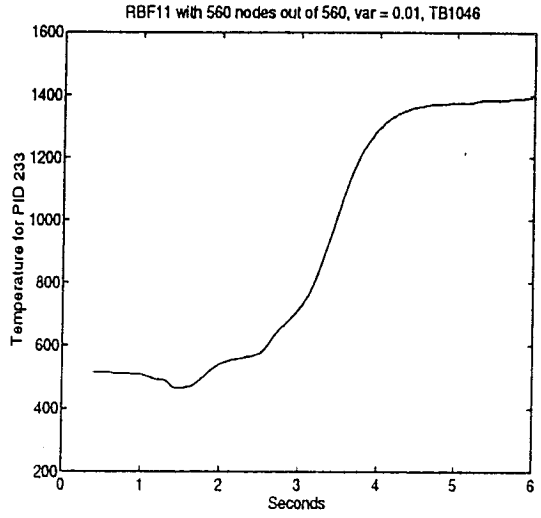

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1072
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

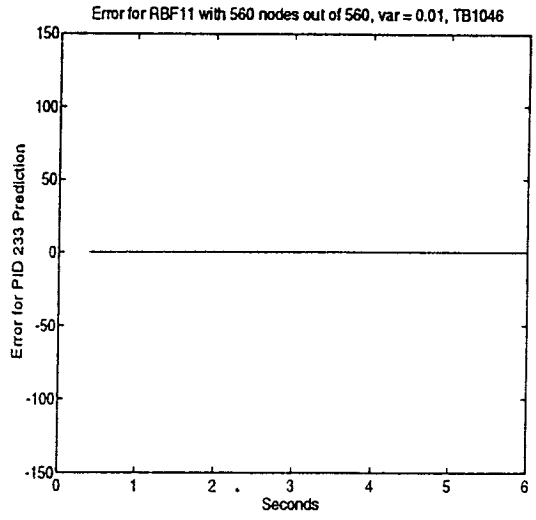

RBF11 with 140 nodes out of 560, var = 0.01, TB1075
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

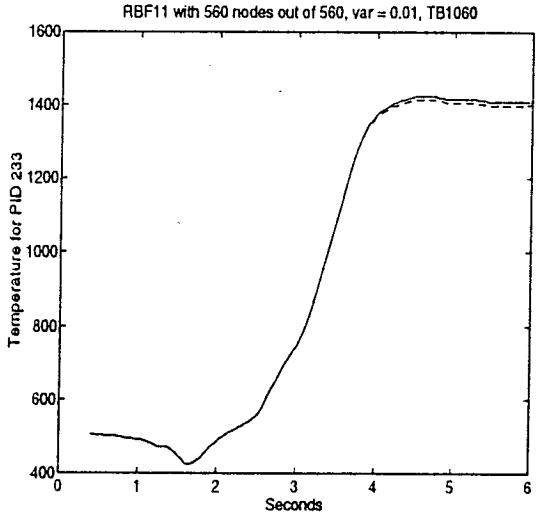

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1075
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

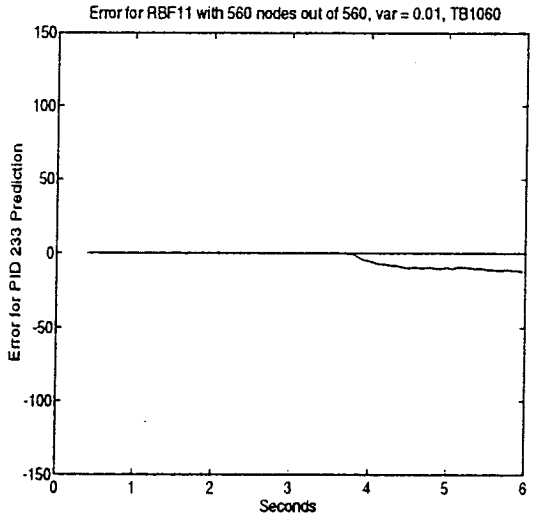

RBF11 with 140 nodes out of 560, var = 0.01, TB1077
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

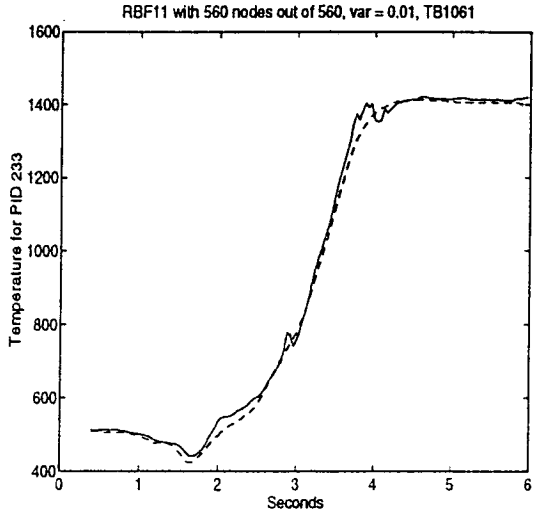

Error for RBF11 with 140 nodes out of 560, var = 0.01, TB1077
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

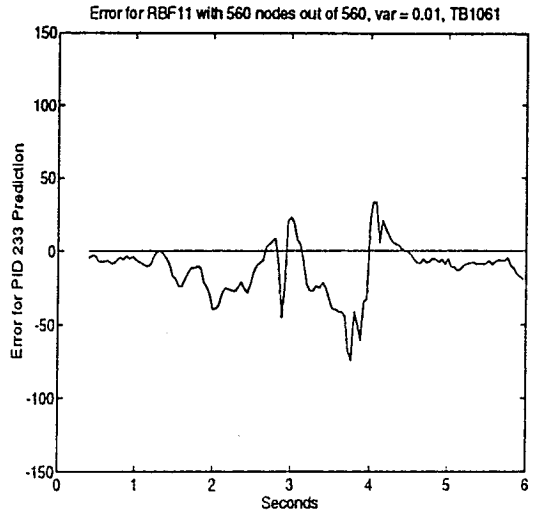

RBF11 with 560 nodes out of 560, var = 0.01, TB1046
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
6

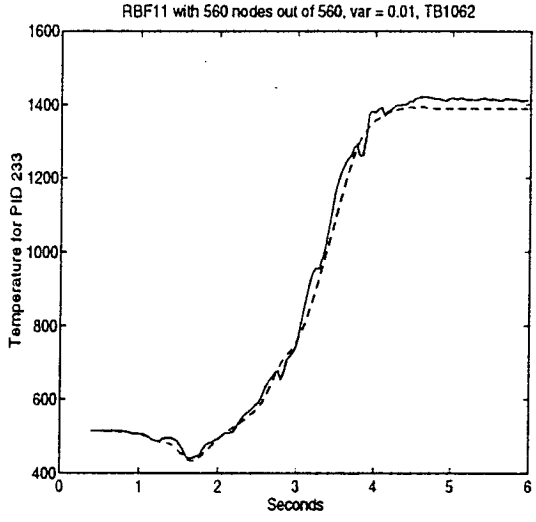

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1046
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

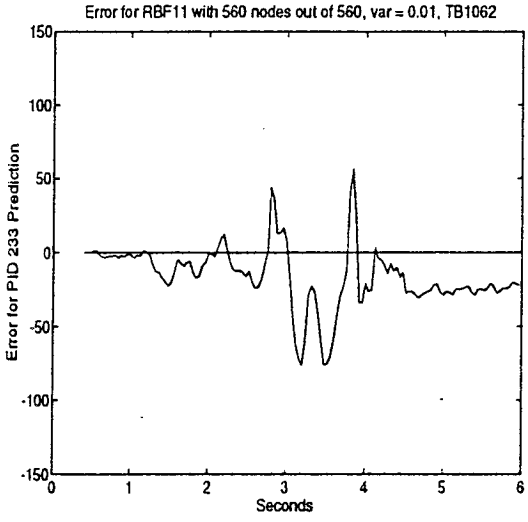

RBF11 with 560 nodes out of 560, var = 0.01, TB1060
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

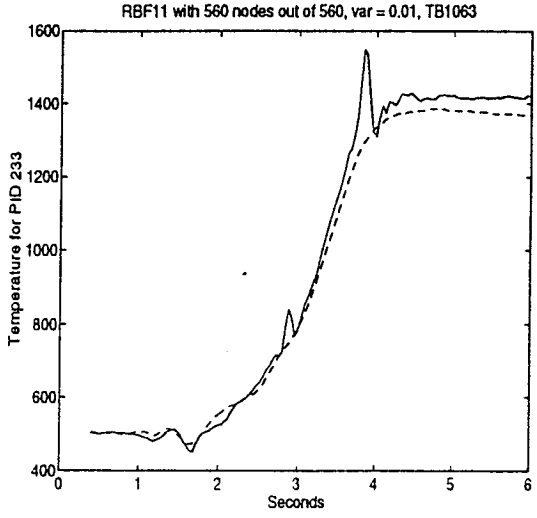

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1060
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

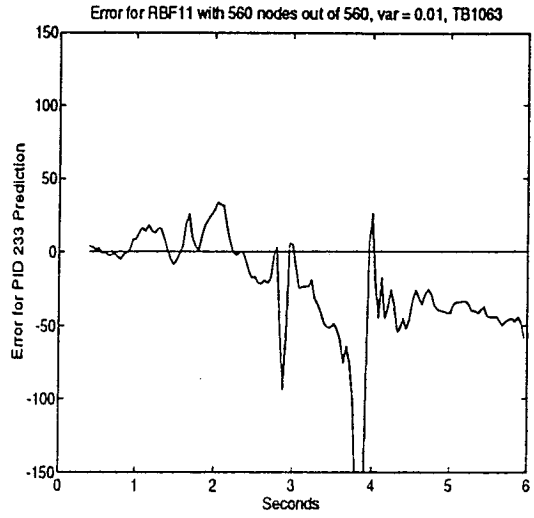

RBF11 with 560 nodes out of 560, var = 0.01, TB1061
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

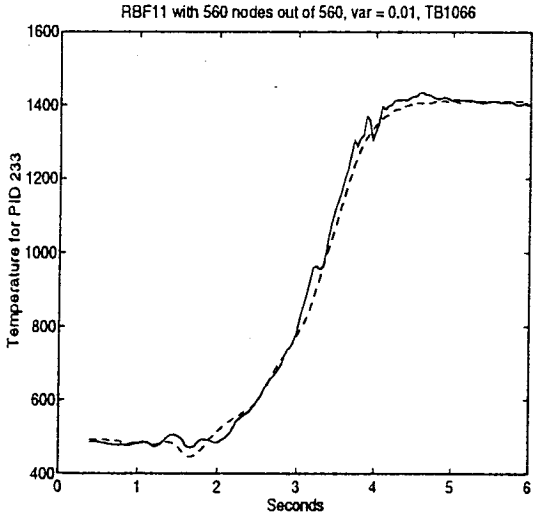

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1061
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

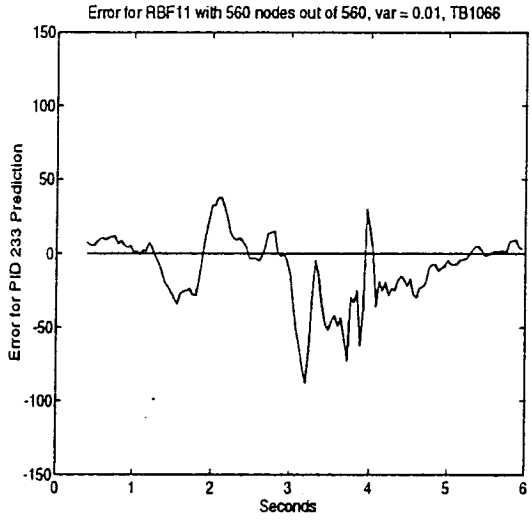

RBF11 with 560 nodes out of 560, var = 0.01, TB1062
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

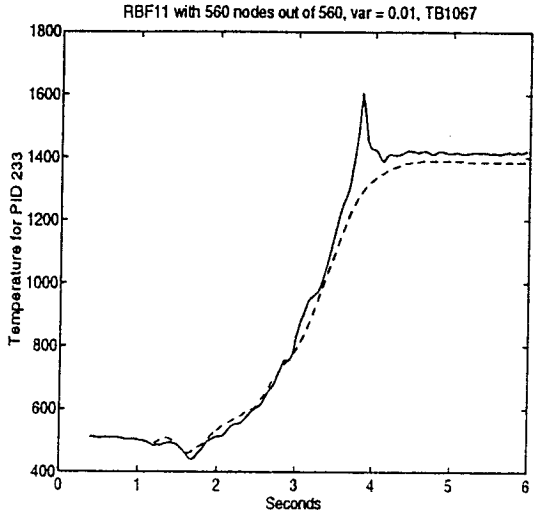

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1062
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

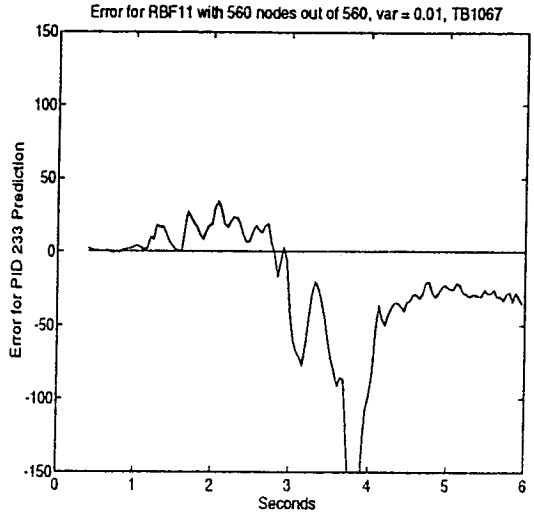

RBF11 with 560 nodes out of 560, var = 0.01, TB1063
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

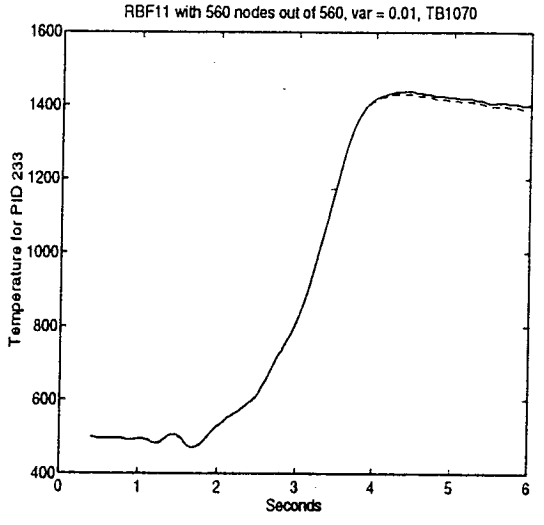

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1063
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

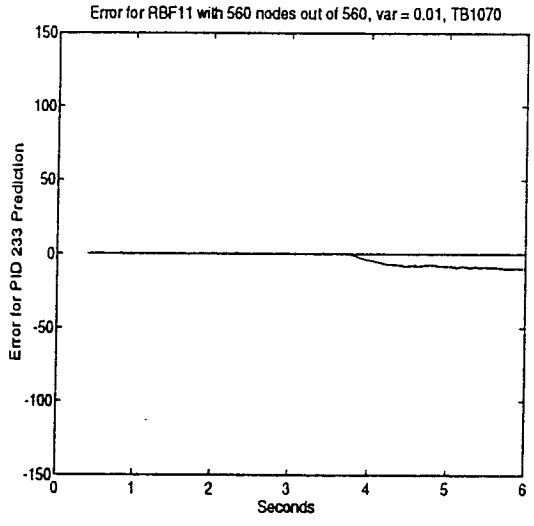

RBF11 with 560 nodes out of 560, var = 0.01, TB1066
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

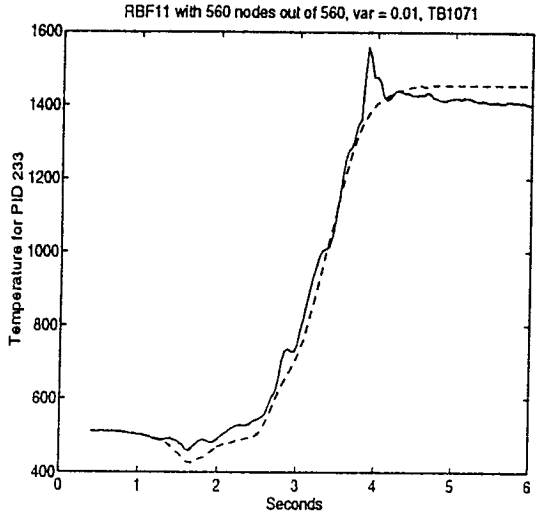

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1066
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

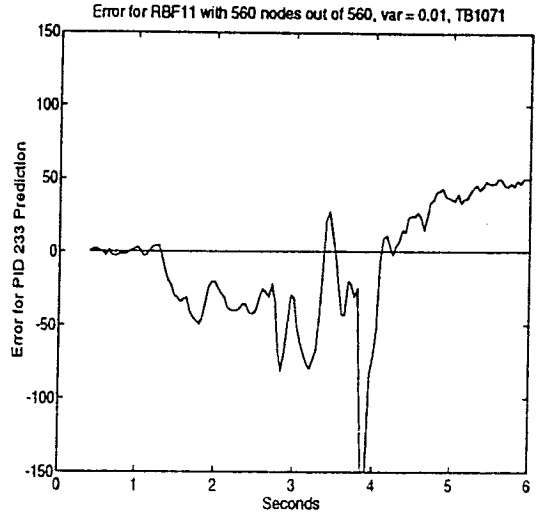

RBF11 with 560 nodes out of 560, var = 0.01, TB1067
Temperature for PID 233
Seconds
1800
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

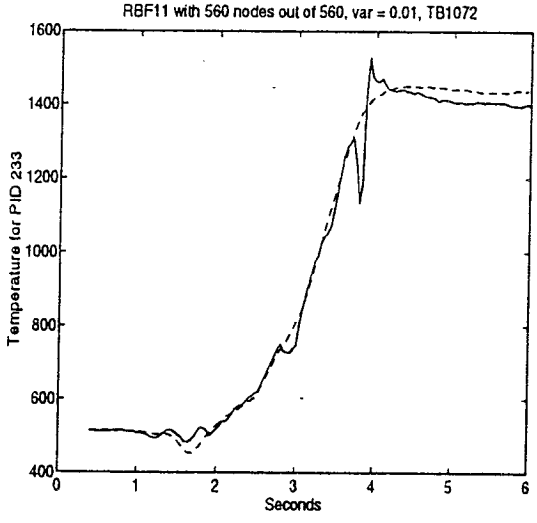

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1067
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

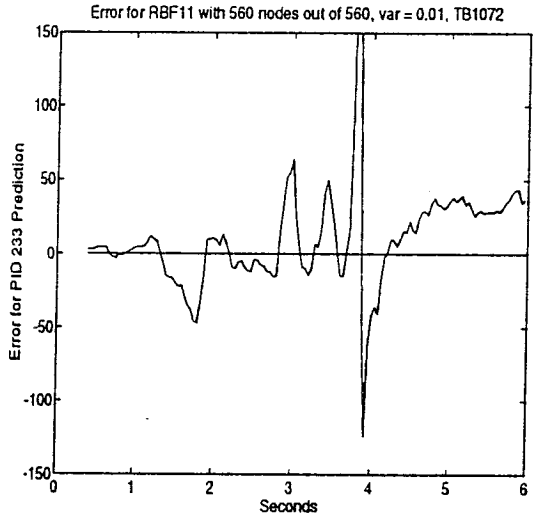

RBF11 with 560 nodes out of 560, var = 0.01, TB1070
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

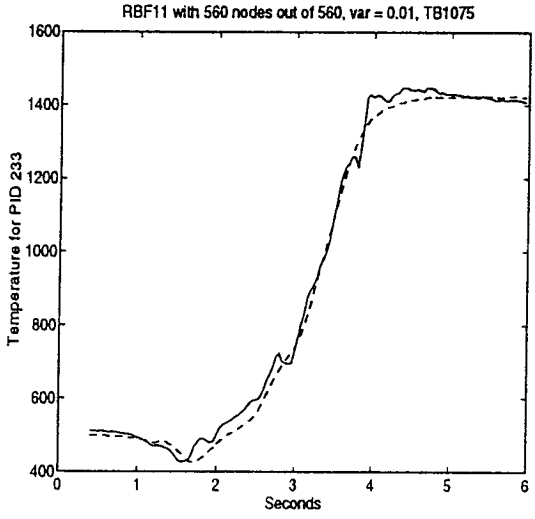

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1070
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

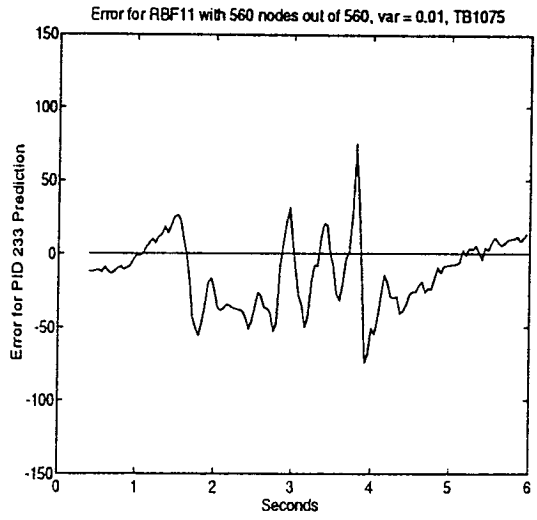

RBF11 with 560 nodes out of 560, var = 0.01, TB1071
Temperature for PID 233
Seconds

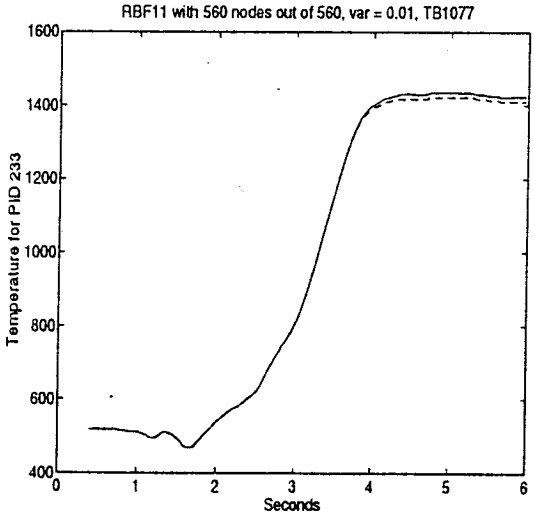

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1071
Error for PID 233 Prediction
Seconds

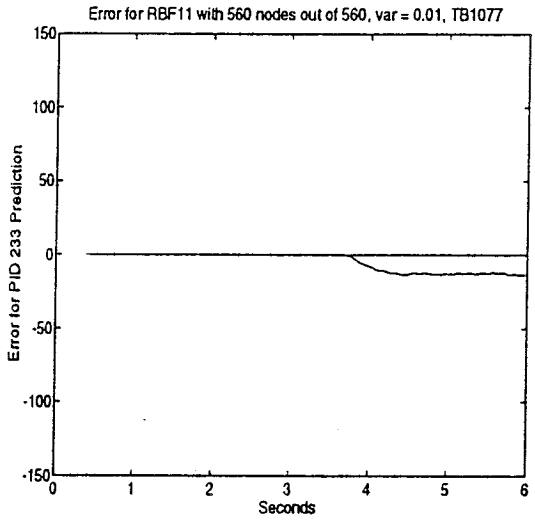

RBF11 with 560 nodes out of 560, var = 0.01, TB1072
Temperature for PID 233
Seconds

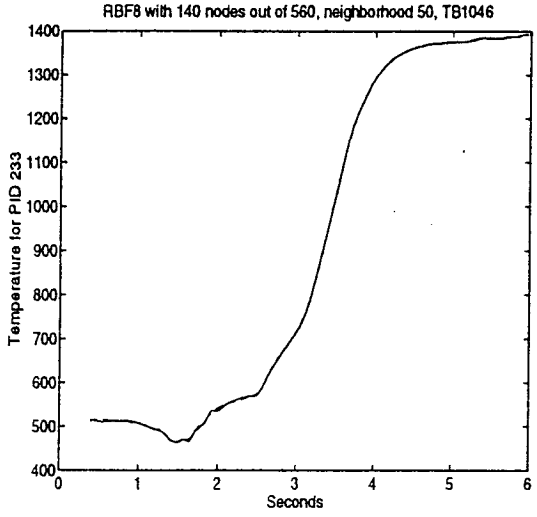

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1072
Error for PID 233 Prediction
Seconds

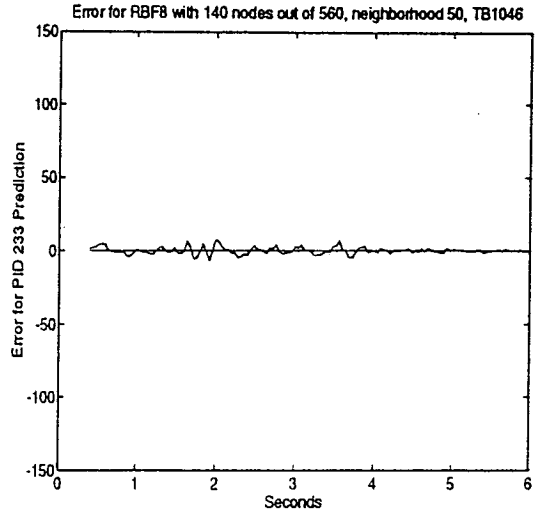

RBF11 with 560 nodes out of 560, var = 0.01, TB1075
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

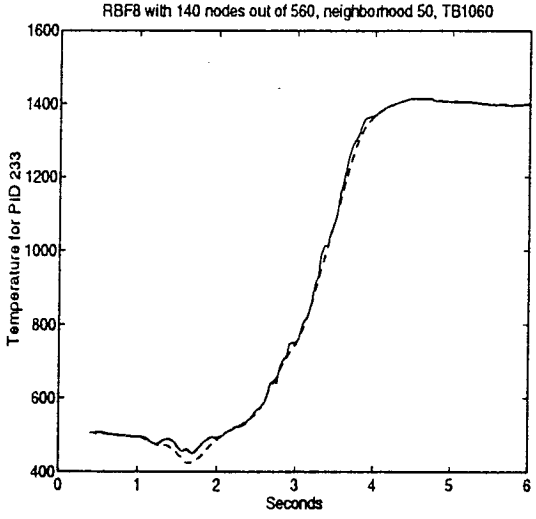

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1075
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

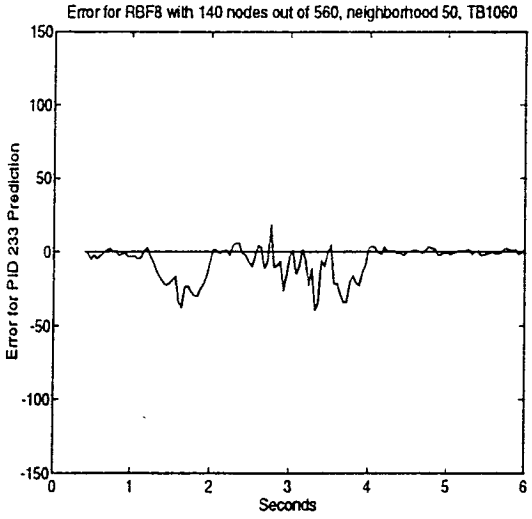

RBF11 with 560 nodes out of 560, var = 0.01, TB1077
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

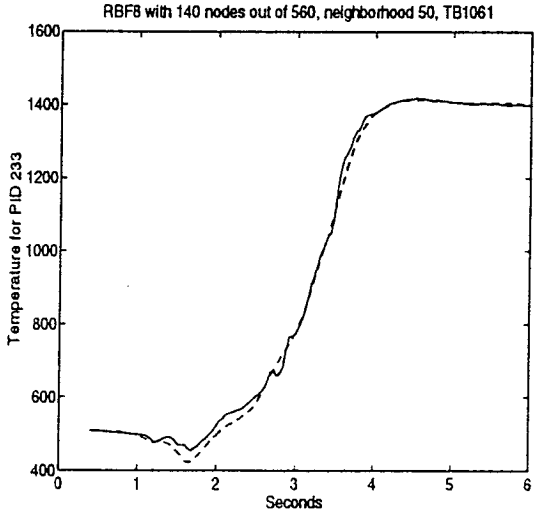

Error for RBF11 with 560 nodes out of 560, var = 0.01, TB1077
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

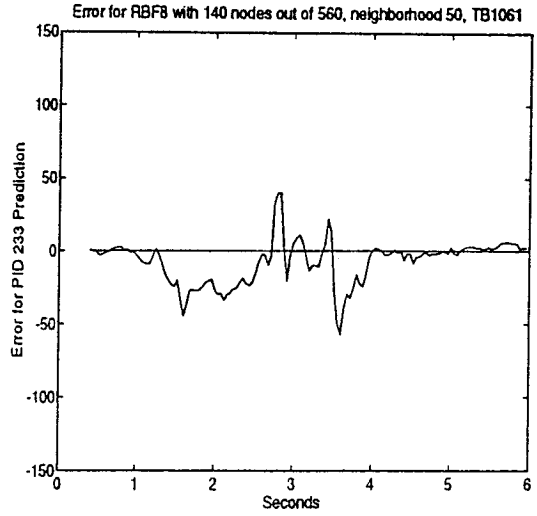

RBF8 with 140 nodes out of 560, neighborhood 50, TB1046
Temperature for PID 233
Seconds

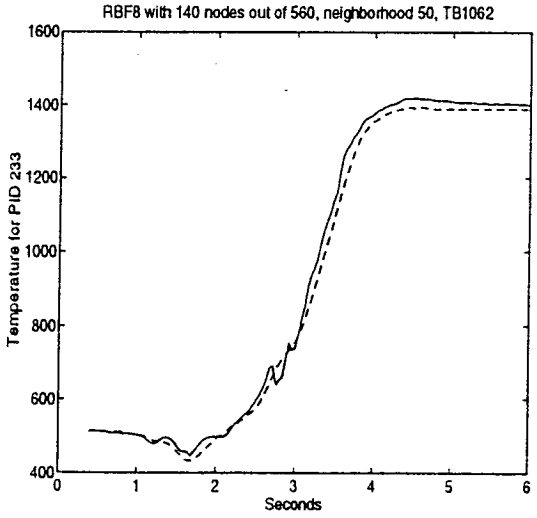

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1046
Error for PID 233 Prediction
Seconds

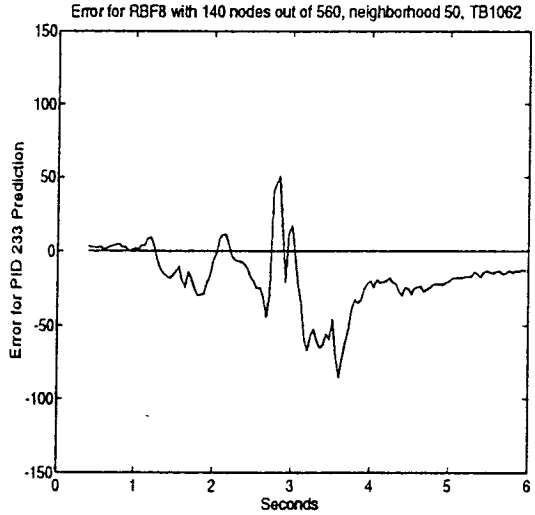

RBF8 with 140 nodes out of 560, neighborhood 50, TB1060
Temperature for PID 233
Seconds

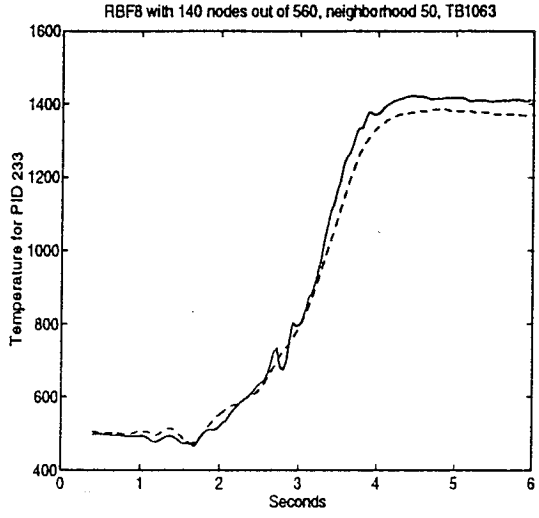

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1060
Error for PID 233 Prediction
Seconds

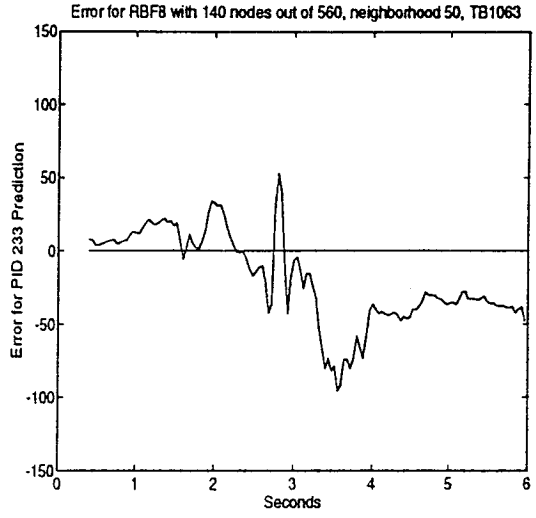

RBF8 with 140 nodes out of 560, neighborhood 50, TB1061
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

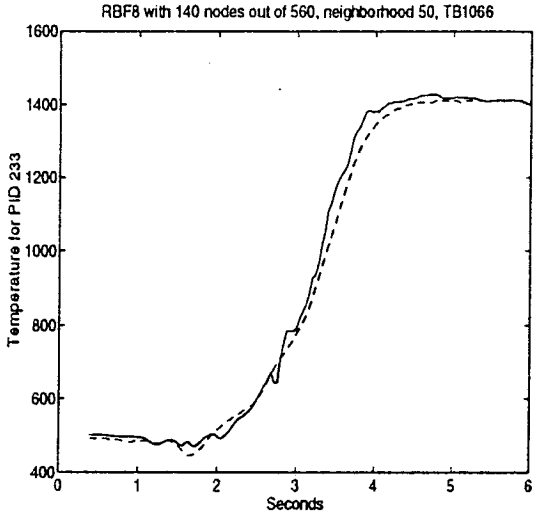

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1061
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

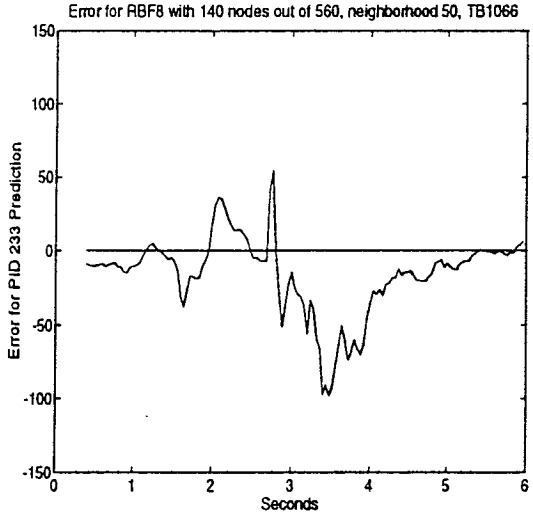

RBF8 with 140 nodes out of 560, neighborhood 50, TB1062
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

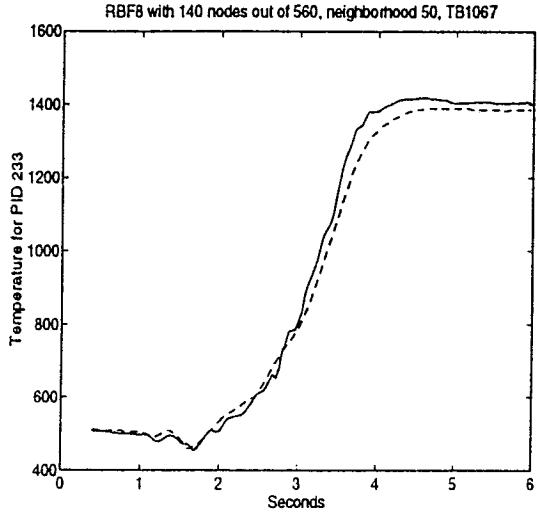

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1062
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

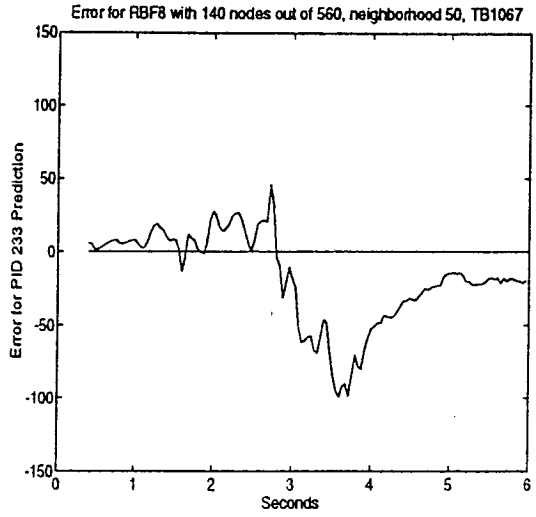

RBF8 with 140 nodes out of 560, neighborhood 50, TB1063
Temperature for PID 233
Seconds

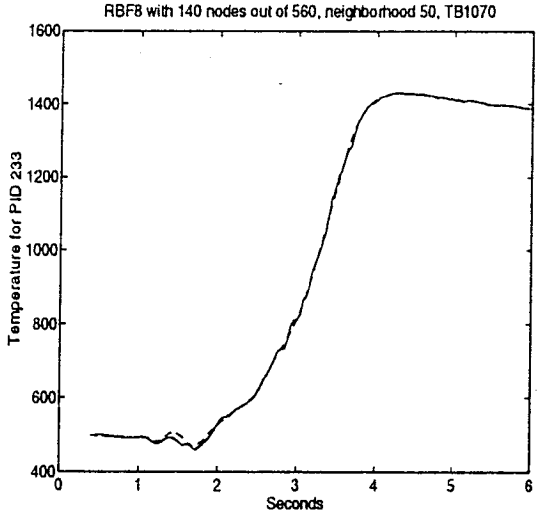

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1063
Error for PID 233 Prediction
Seconds

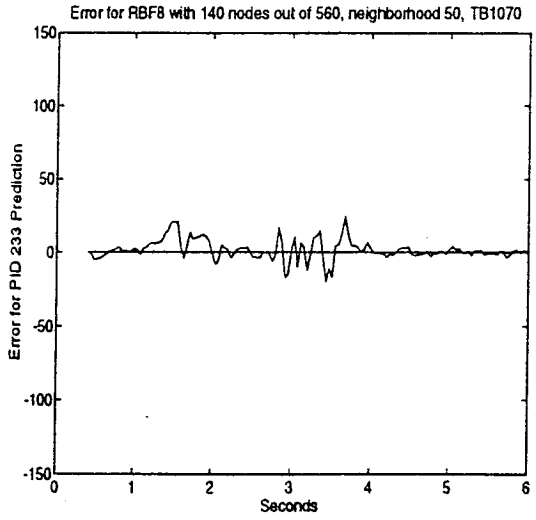

RBF8 with 140 nodes out of 560, neighborhood 50, TB1066
Temperature for PID 233
Seconds

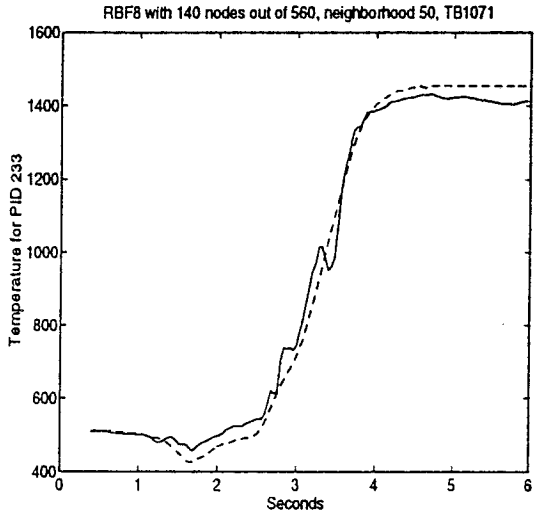

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1066
Error for PID 233 Prediction
Seconds

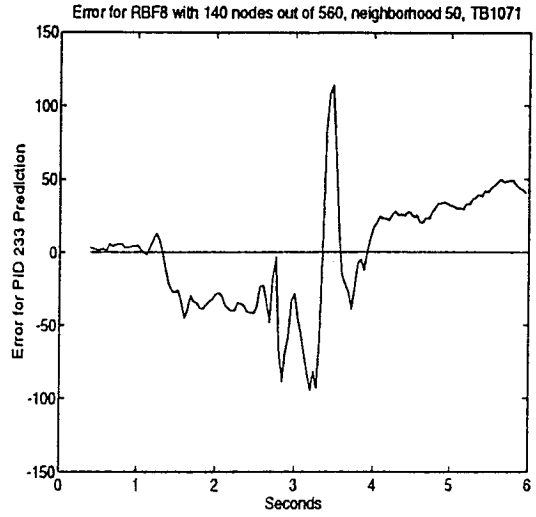

RBF8 with 140 nodes out of 560, neighborhood 50, TB1067
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

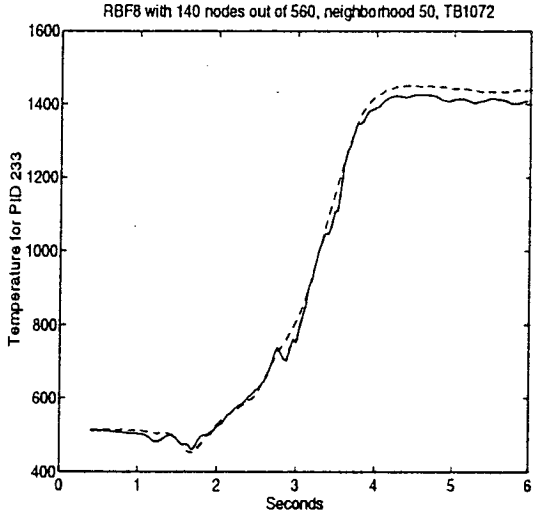

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1067
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

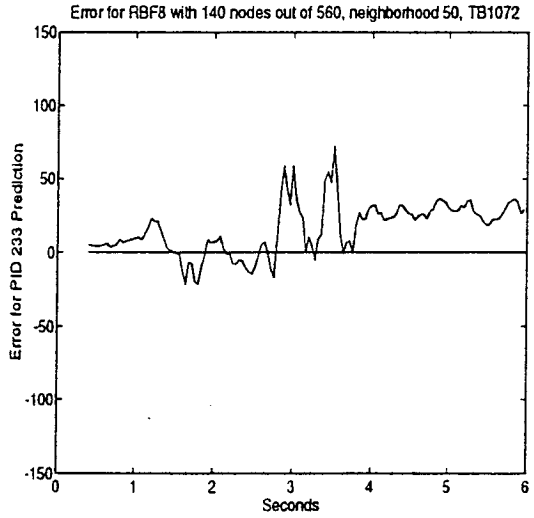

RBF8 with 140 nodes out of 560, neighborhood 50, TB1070
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

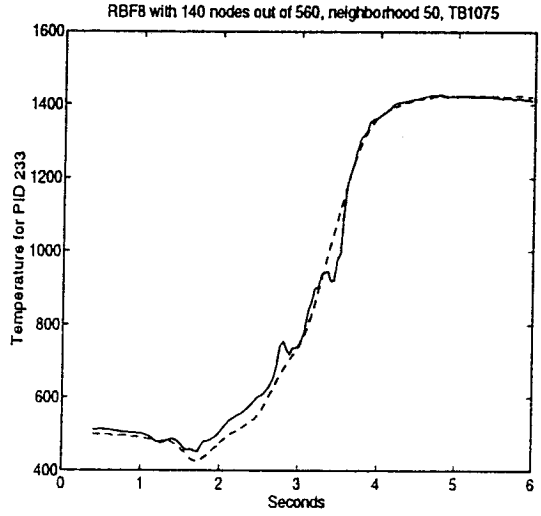

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1070
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

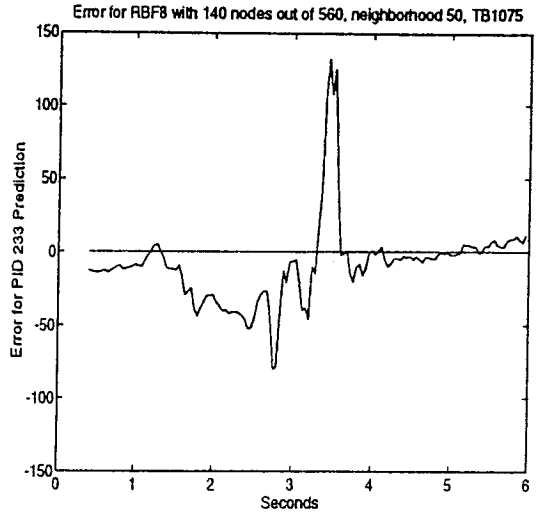

RBF8 with 140 nodes out of 560, neighborhood 50, TB1071
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

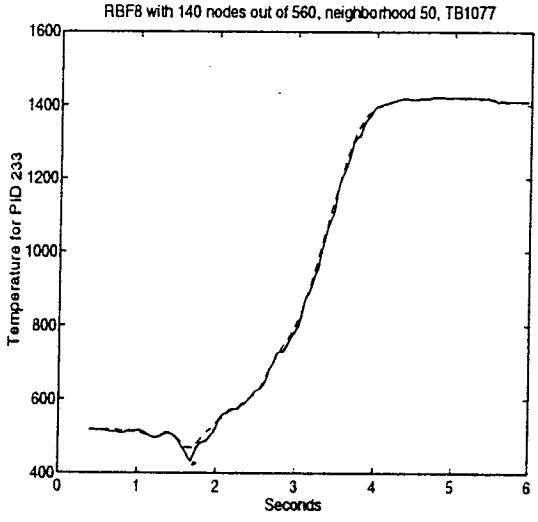

Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1071
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

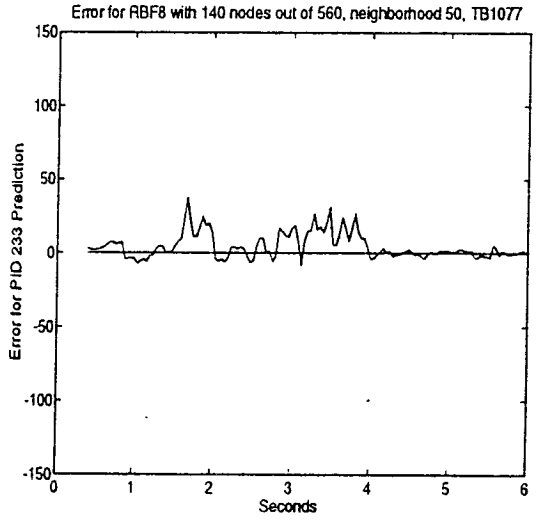

RBF8 with 140 nodes out of 560, neighborhood 50, TB1072
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6


Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1072
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6


RBF8 with 140 nodes out of 560, neighborhood 50, TB1075
Temperature for PID 233
Seconds


Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1075
Error for PID 233 Prediction
Seconds


RBF8 with 140 nodes out of 560, neighborhood 50, TB1077
Temperature for PID 233
Seconds


Error for RBF8 with 140 nodes out of 560, neighborhood 50, TB1077
Error for PID 233 Prediction
Seconds

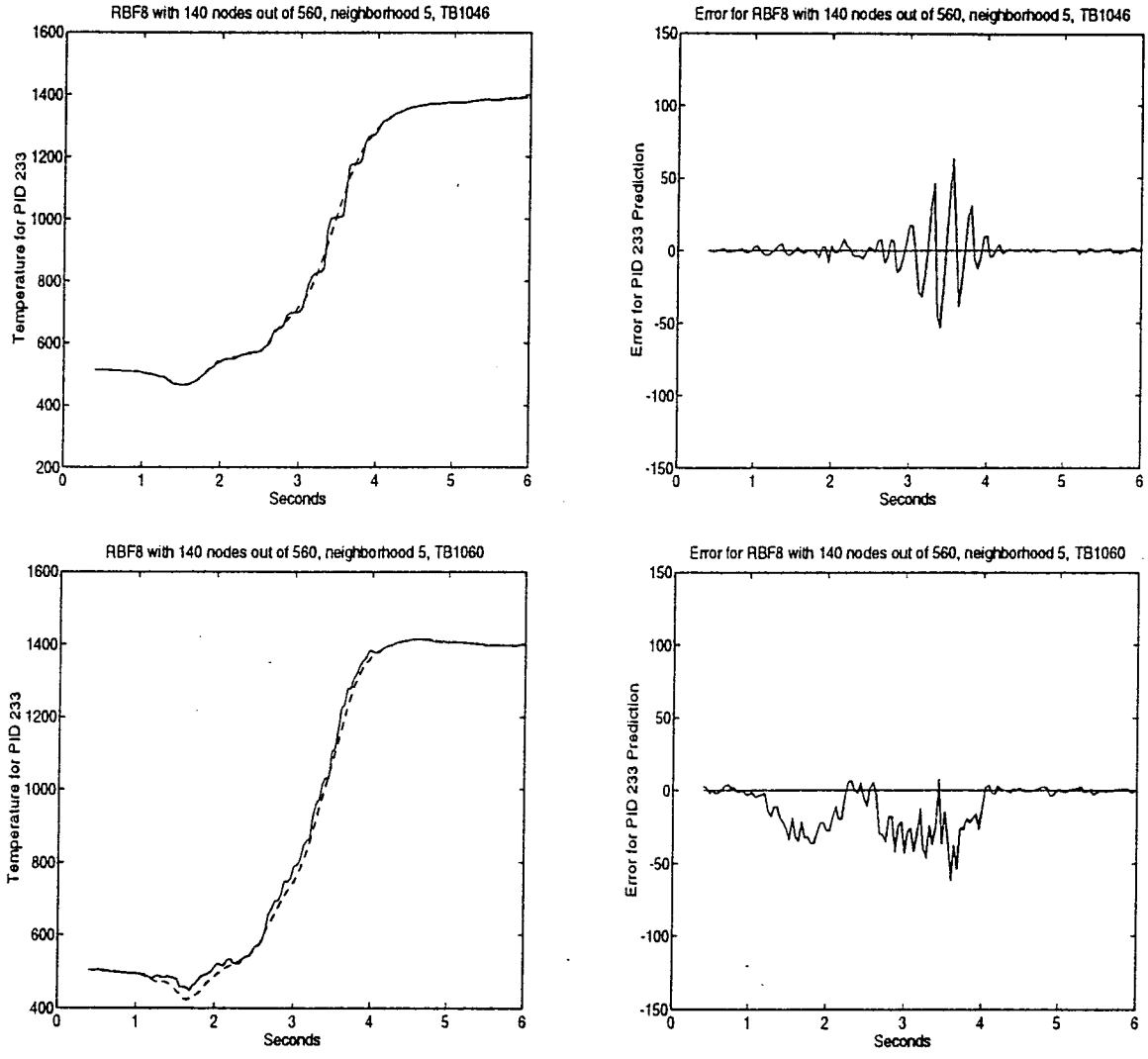


Figure 1: Rbf8 with 140 nodes, neighborhood 5

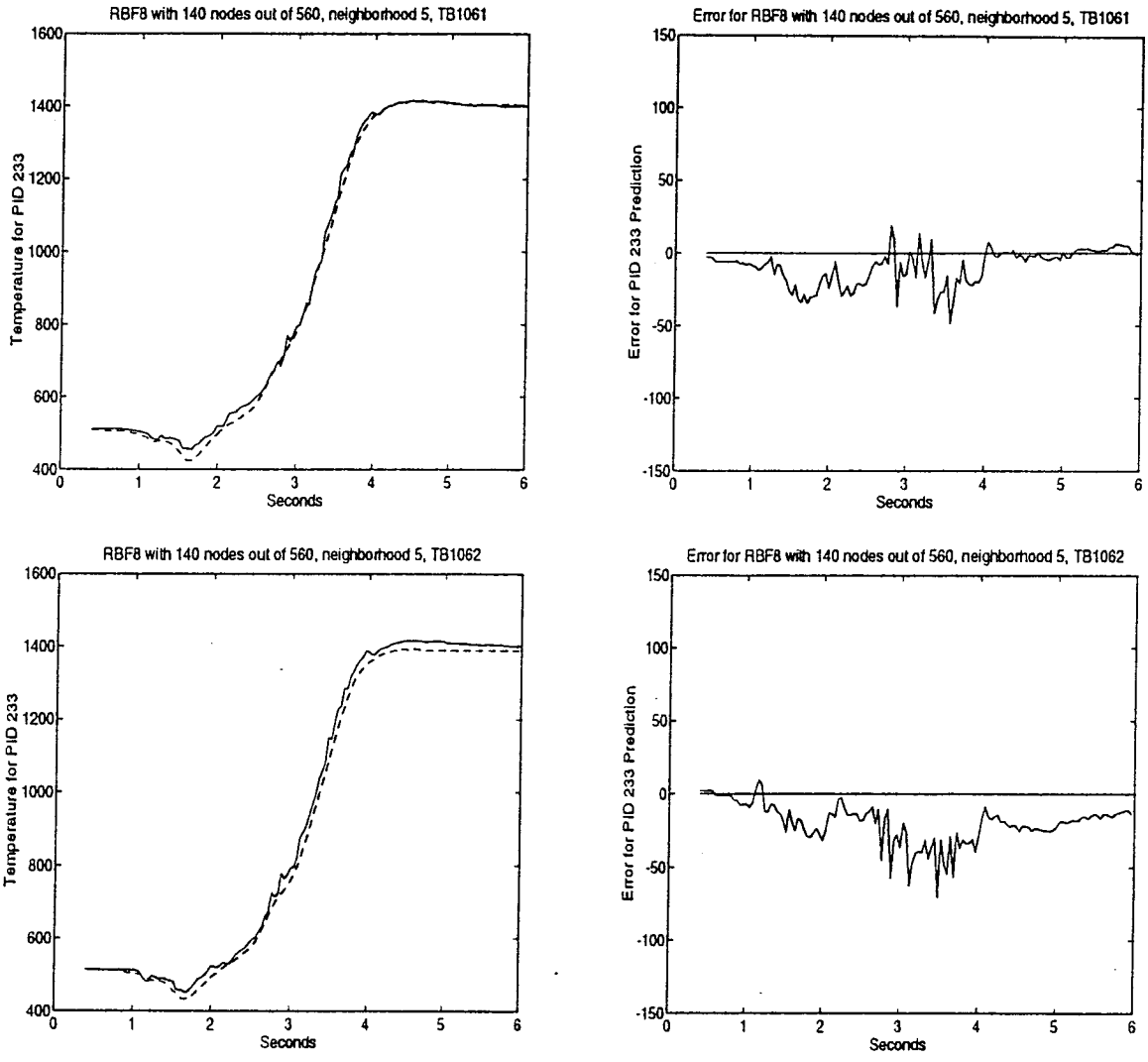


Figure 2: Rbf8 with 140 nodes, neighborhood 5

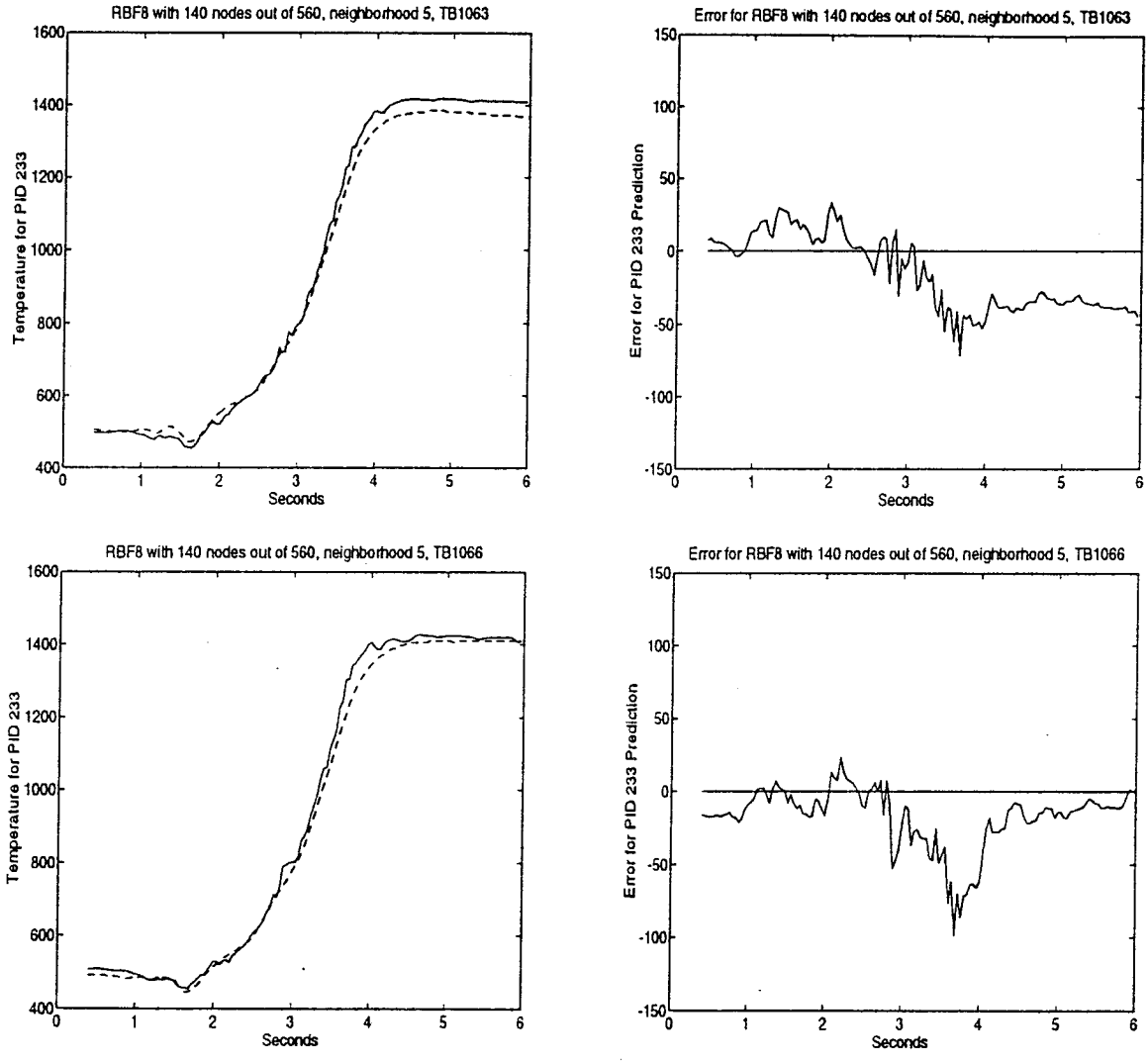


Figure 3: Rbf8 with 140 nodes, neighborhood 5

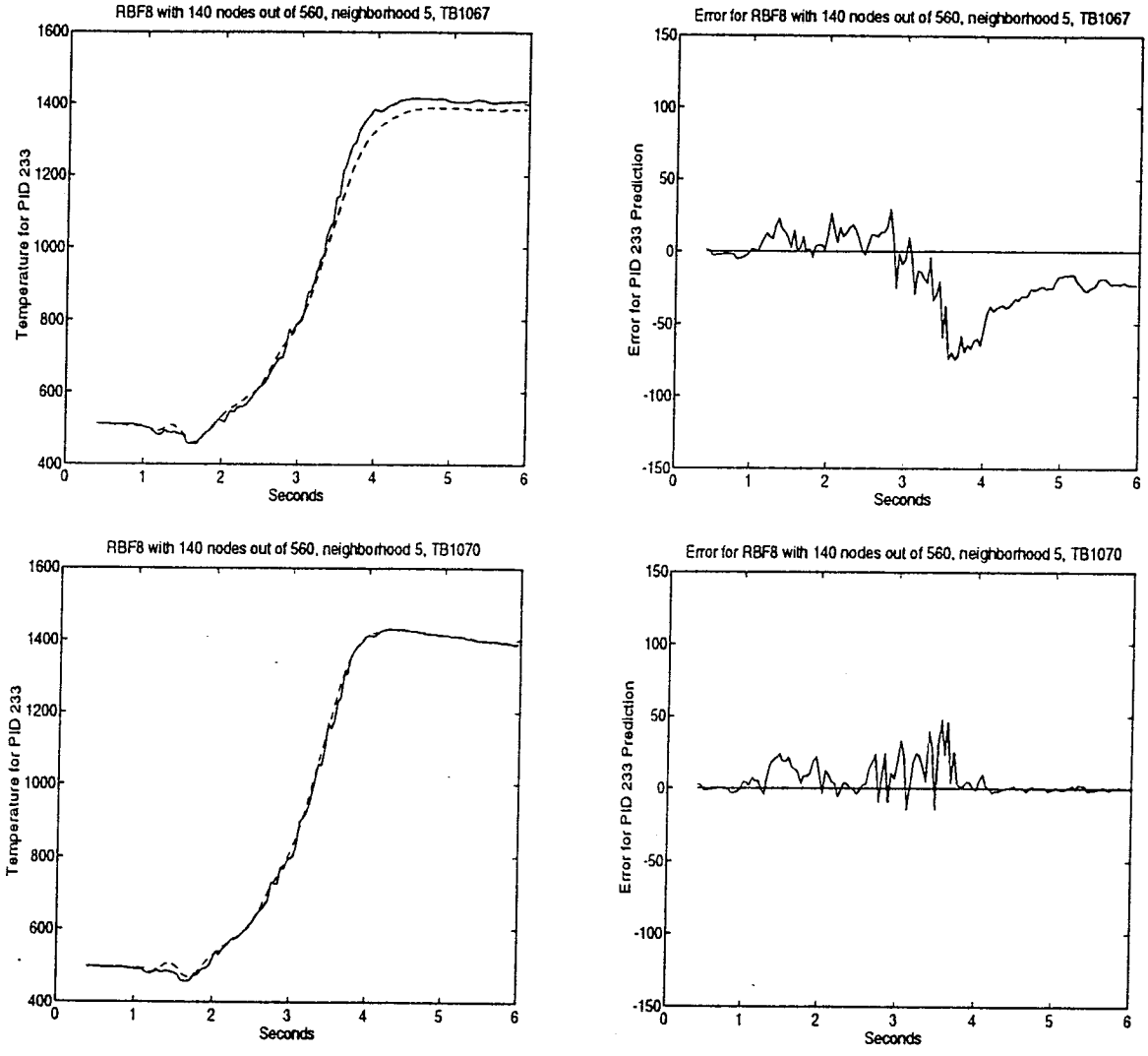


Figure 4: Rbf8 with 140 nodes, neighborhood 5

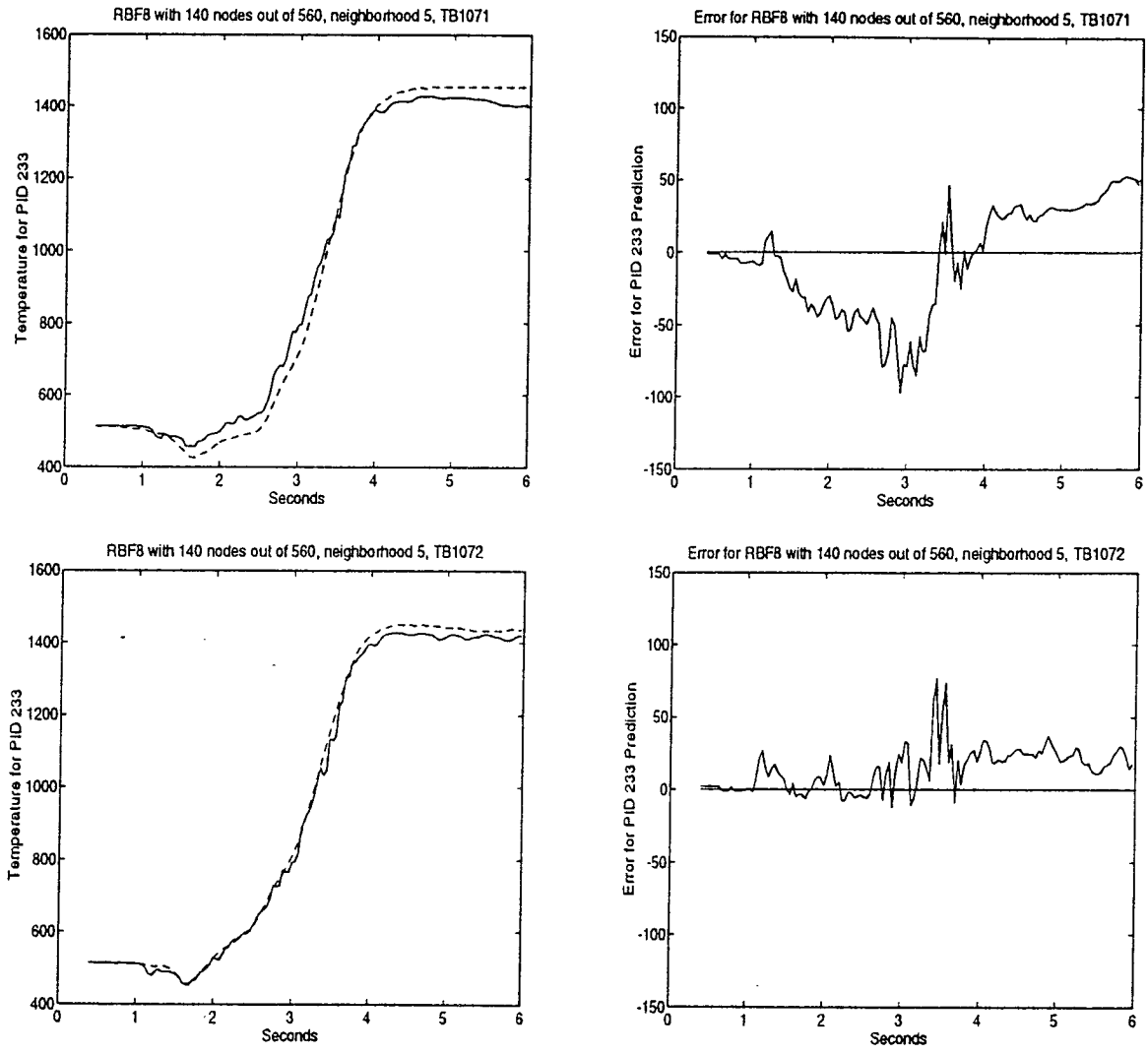


Figure 5: Rbf8 with 140 nodes, neighborhood 5

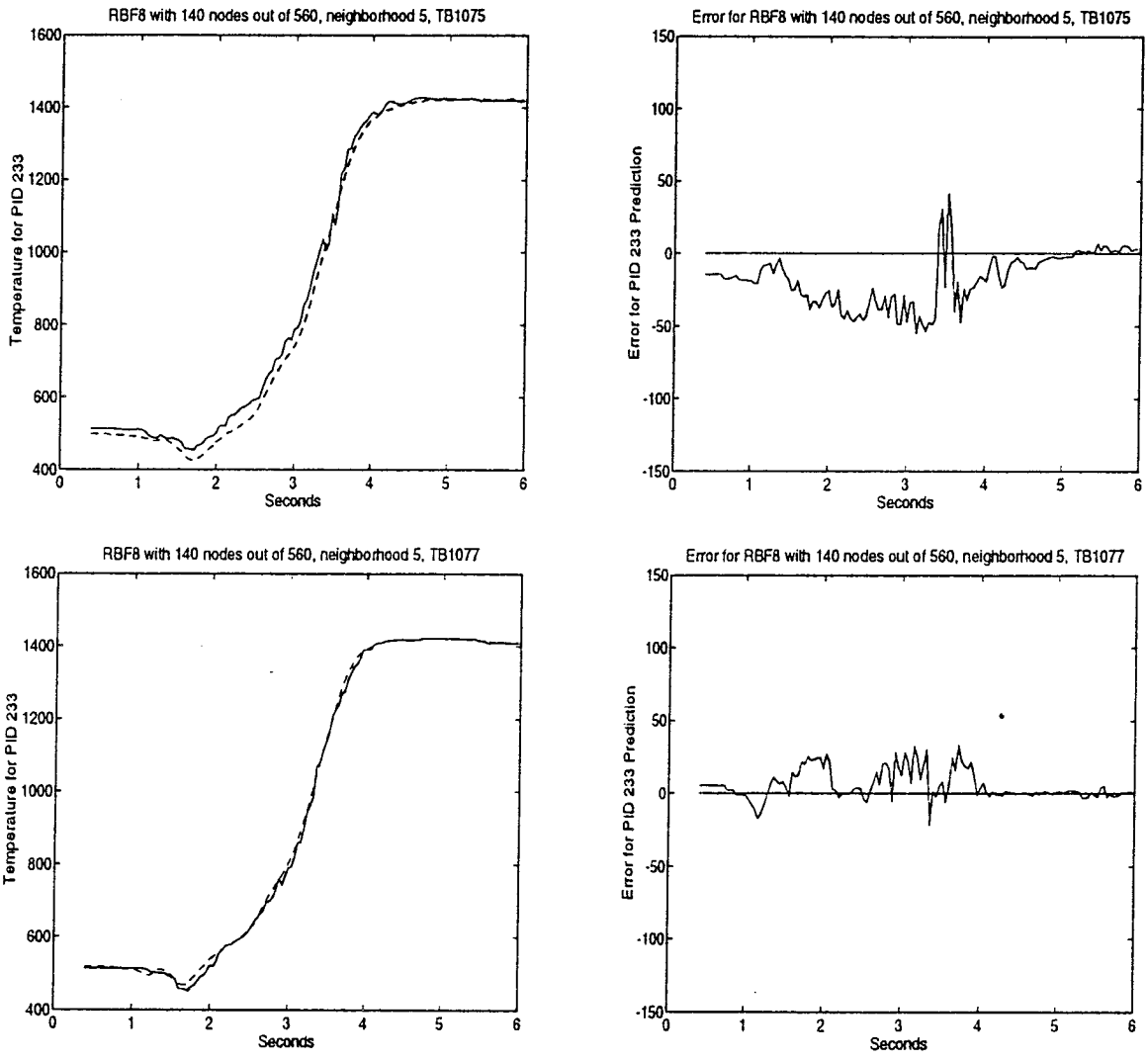
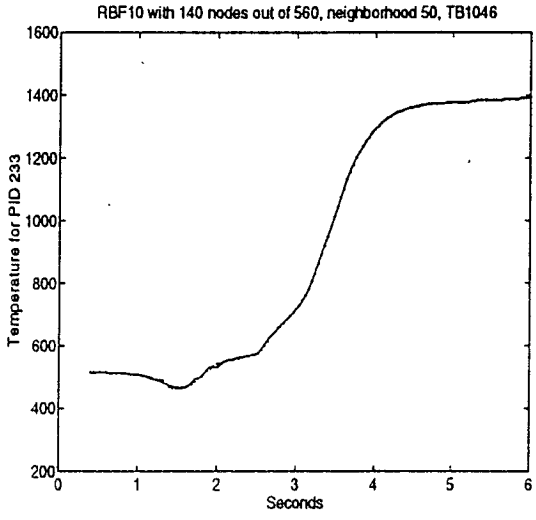
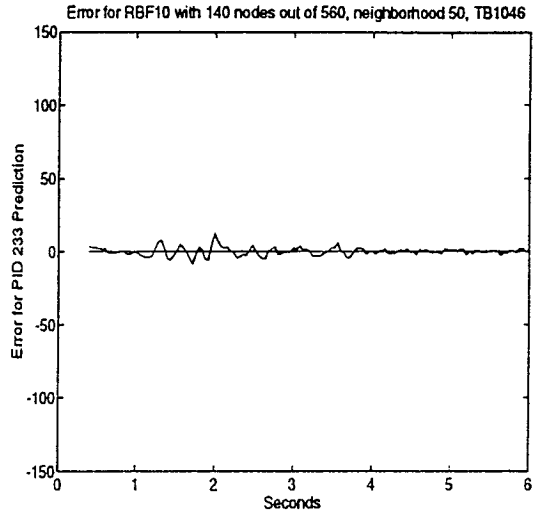
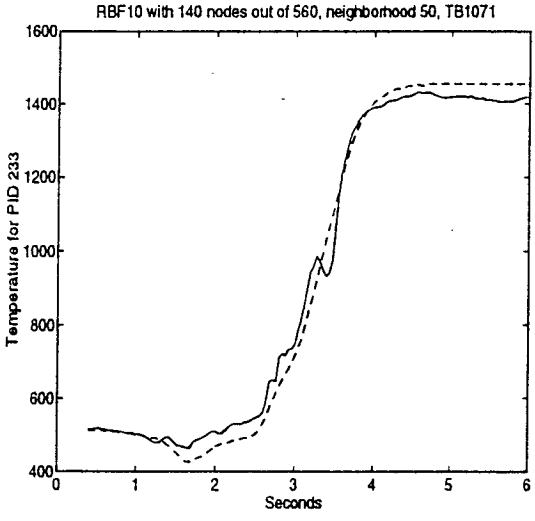
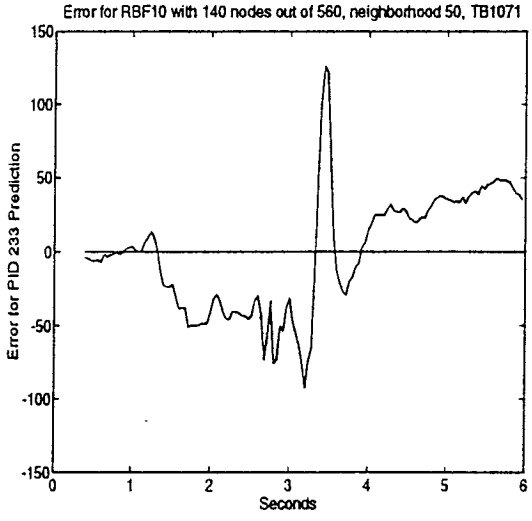


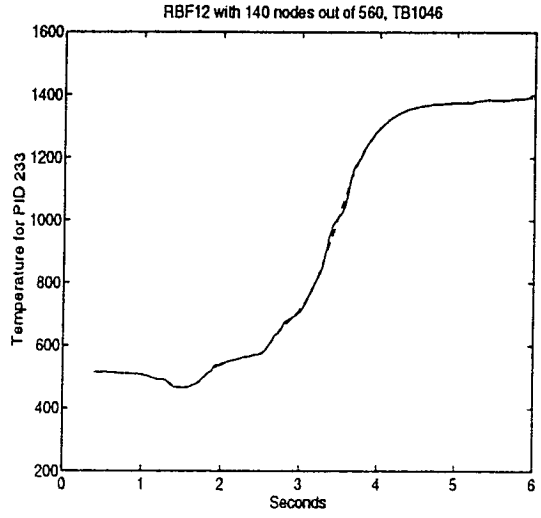
Figure 6: Rbf8 with 140 nodes, neighborhood 5

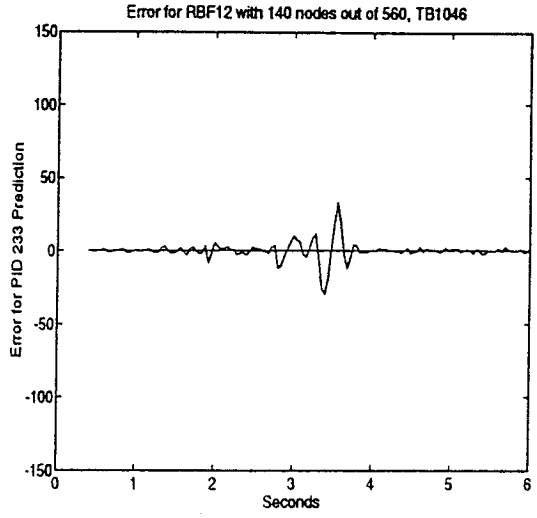

RBF10 with 140 nodes out of 560, neighborhood 50, TB1046
Temperature for PID 233
Seconds

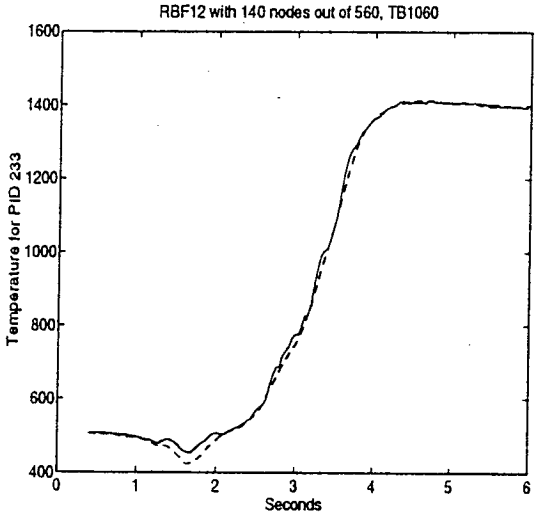

Error for RBF10 with 140 nodes out of 560, neighborhood 50, TB1046
Error for PID 233 Prediction
Seconds

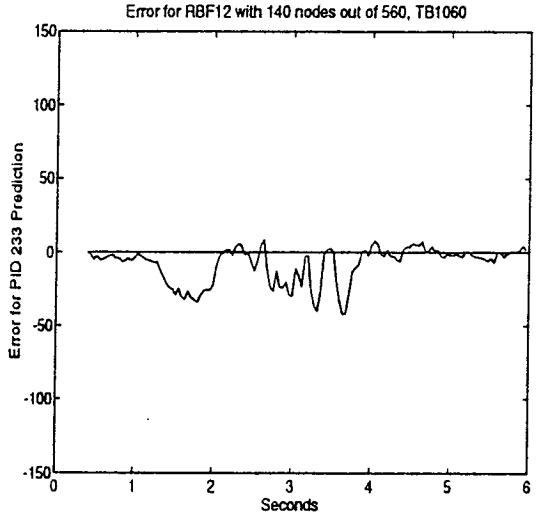

RBF10 with 140 nodes out of 560, neighborhood 50, TB1071
Temperature for PID 233
Seconds

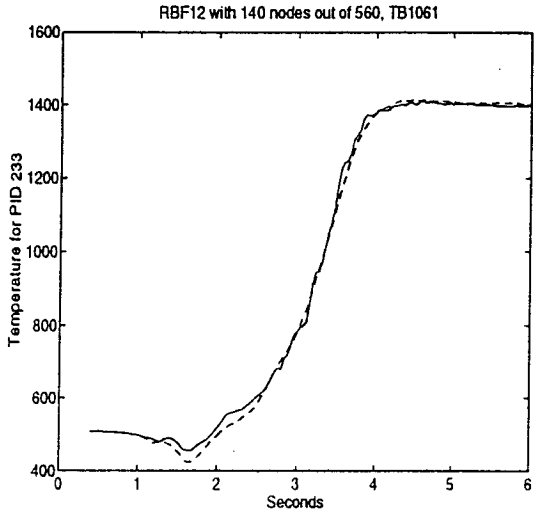

Error for RBF10 with 140 nodes out of 560, neighborhood 50, TB1071
Error for PID 233 Prediction
Seconds

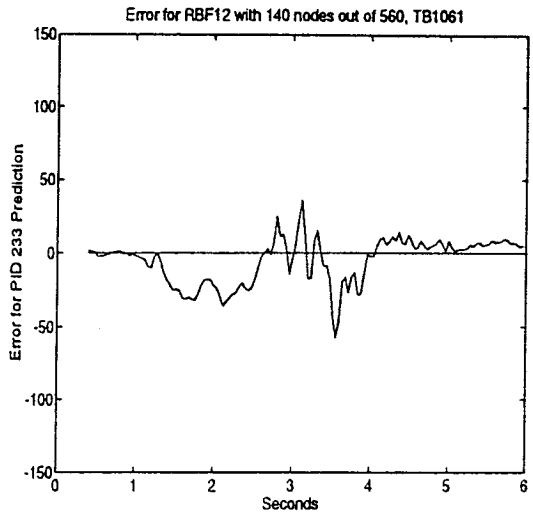

RBF12 with 140 nodes out of 560, TB1046
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
6

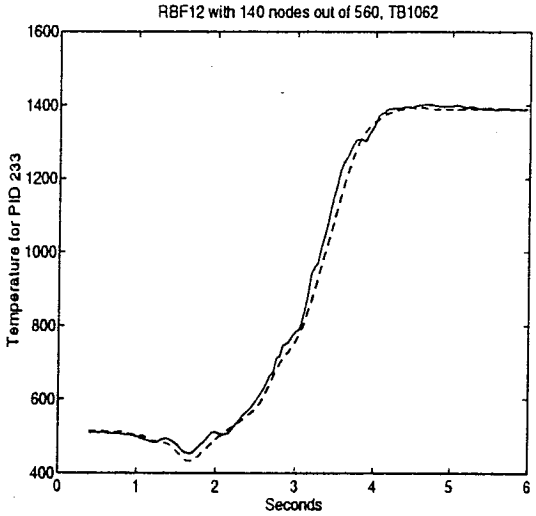

Error for RBF12 with 140 nodes out of 560, TB1046
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

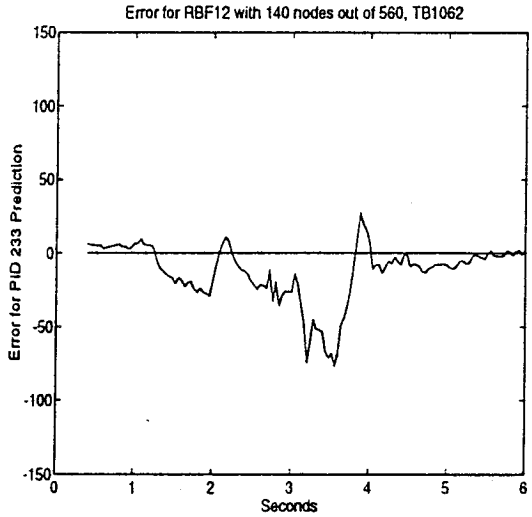

RBF12 with 140 nodes out of 560, TB1060
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

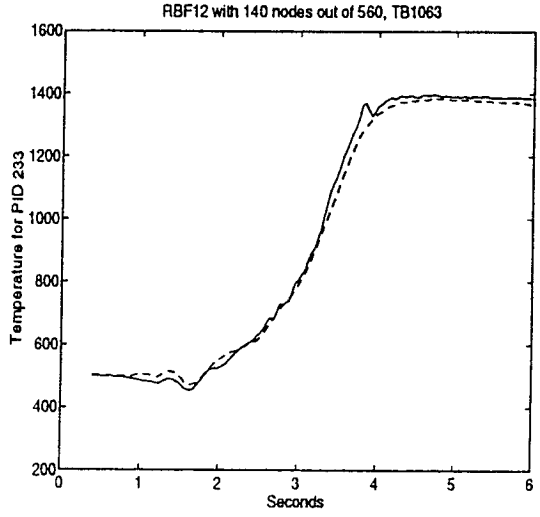

Error for RBF12 with 140 nodes out of 560, TB1060
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

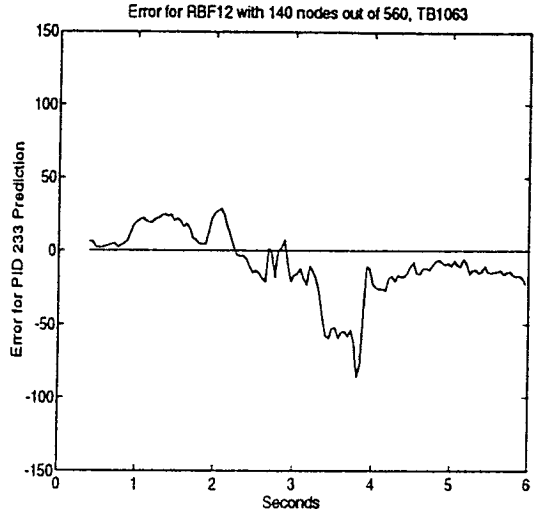

RBF12 with 140 nodes out of 560, TB1061
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

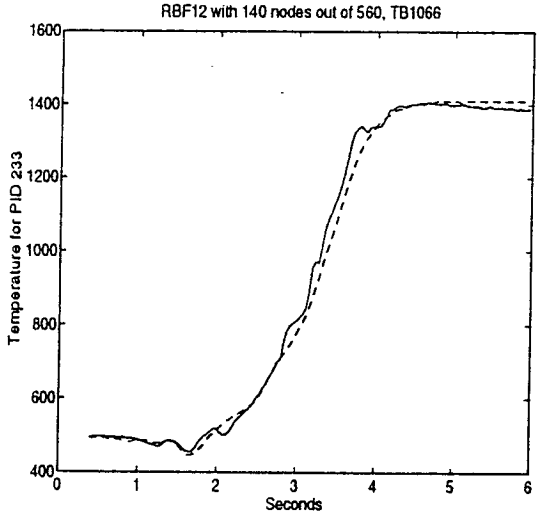

Error for RBF12 with 140 nodes out of 560, TB1061
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

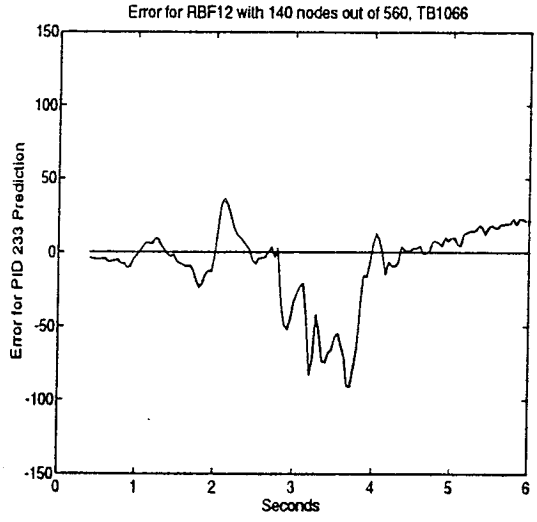

RBF12 with 140 nodes out of 560, TB1062
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

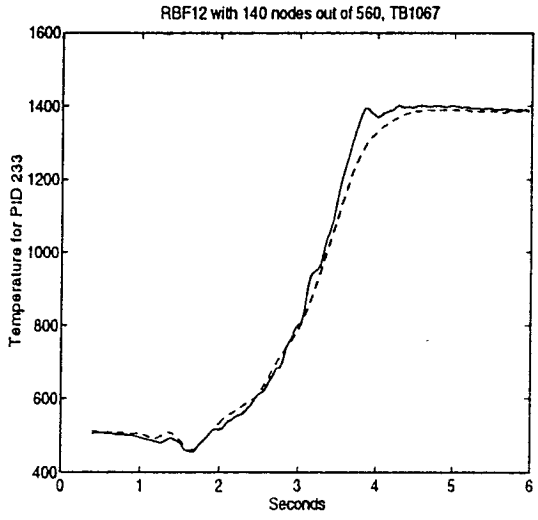

Error for RBF12 with 140 nodes out of 560, TB1062
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

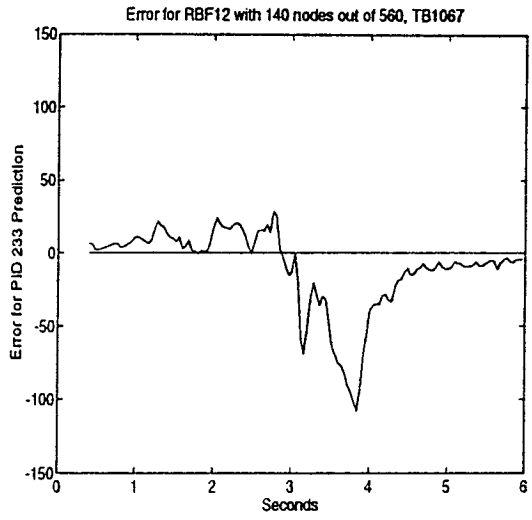

RBF12 with 140 nodes out of 560, TB1063
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
6

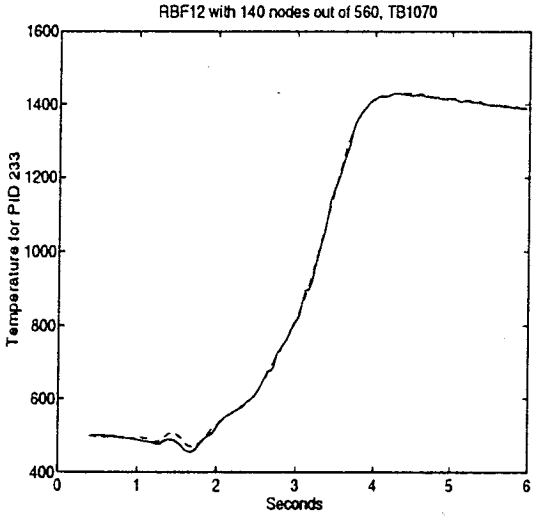

Error for RBF12 with 140 nodes out of 560, TB1063
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

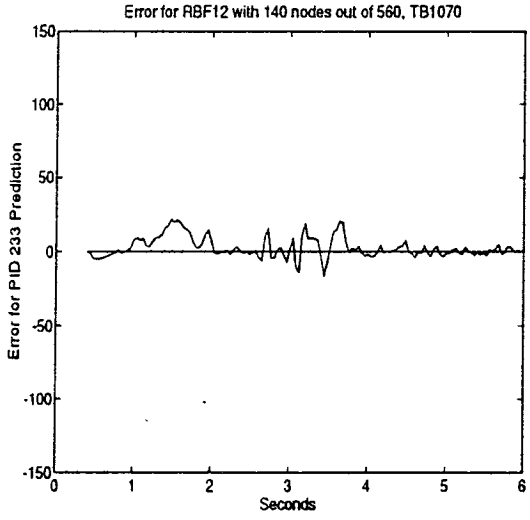

RBF12 with 140 nodes out of 560, TB1066
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

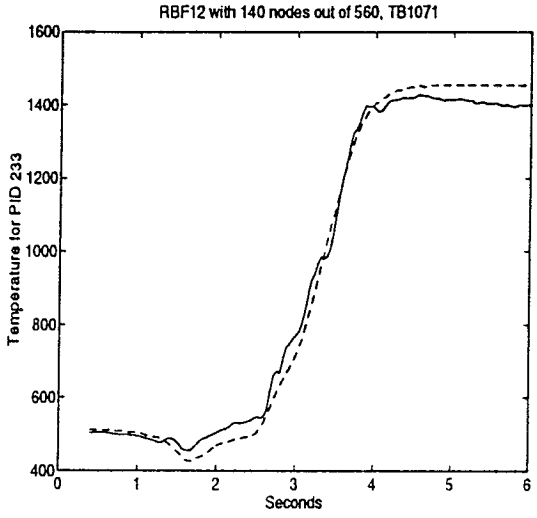

Error for RBF12 with 140 nodes out of 560, TB1066
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

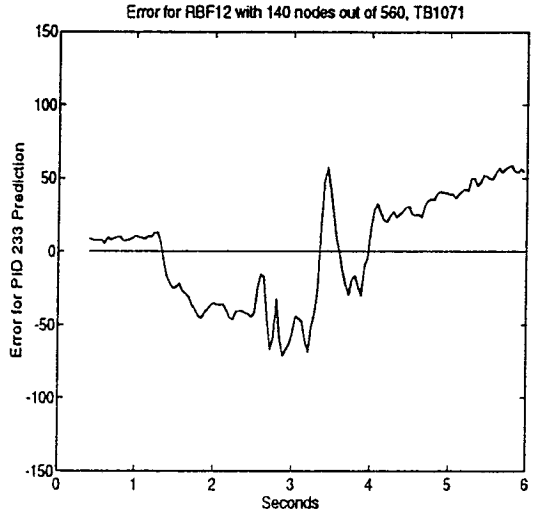

RBF12 with 140 nodes out of 560, TB1067
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

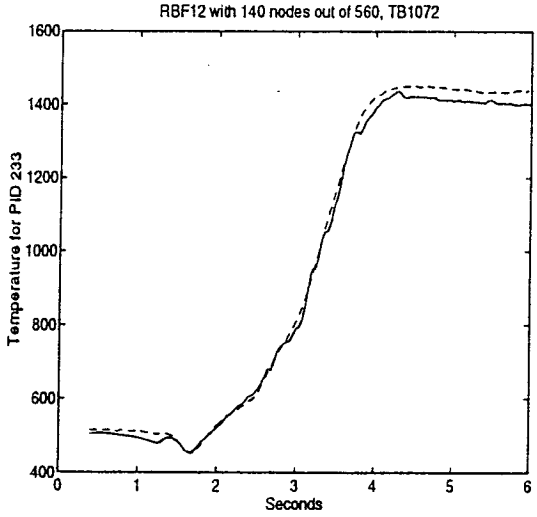

Error for RBF12 with 140 nodes out of 560, TB1067
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

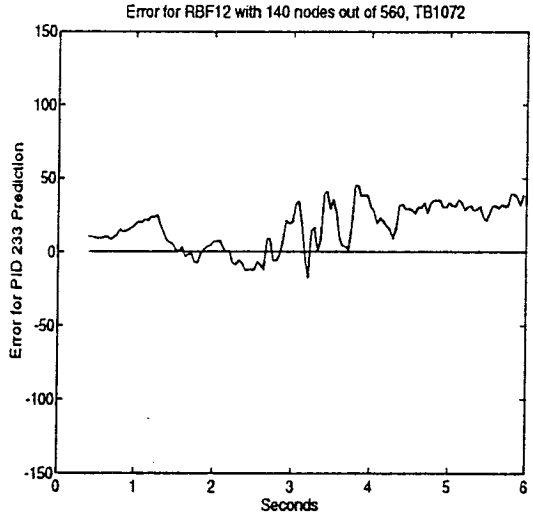

RBF12 with 140 nodes out of 560, TB1070
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

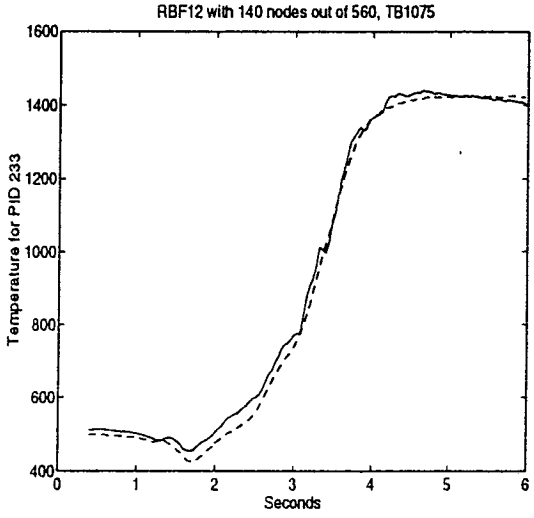

Error for RBF12 with 140 nodes out of 560, TB1070
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

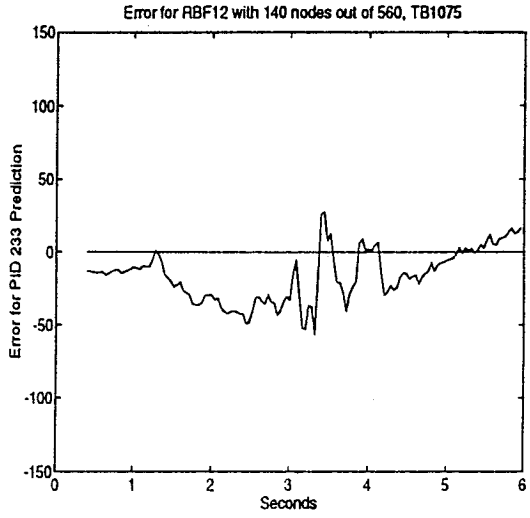

RBF12 with 140 nodes out of 560, TB1071
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6

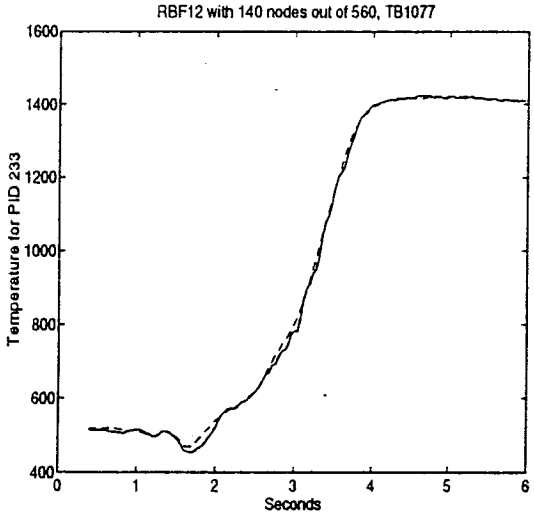

Error for RBF12 with 140 nodes out of 560, TB1071
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

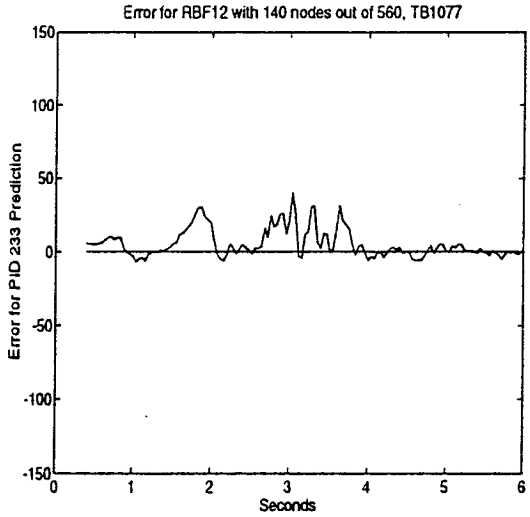

RBF12 with 140 nodes out of 560, TB1072
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6


Error for RBF12 with 140 nodes out of 560, TB1072
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6


RBF12 with 140 nodes out of 560, TB1075
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6


Error for RBF12 with 140 nodes out of 560, TB1075
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6


RBF12 with 140 nodes out of 560, TB1077
Temperature for PID 233
Seconds
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5
6


Error for RBF12 with 140 nodes out of 560, TB1077
Error for PID 233 Prediction
Seconds
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1993	Final Contractor Report

4. TITLE AND SUBTITLE

Radial Basis Function Neural Networks Applied to NASA SSME Data

5. FUNDING NUMBERS

WU–584–03–11
NCC3–308

6. AUTHOR(S)

Kevin R. Wheeler and Atam P. Dhawan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Cincinnati
Department of Electrical and Computer Engineering
Cincinnati, Ohio 45221 ML 30

8. PERFORMING ORGANIZATION REPORT NUMBER

E–9347

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR–195417

11. SUPPLEMENTARY NOTES

Project manager, June F. Zakrajsek, Space Propulsion Technology Division, NASA Lewis Research Center, organization code 5310, (216) 433–7470.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 15

This publication is available from the NASA Center for Aerospace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This paper presents a brief report on the application of Radial Basis Function Neural Networks (RBFNN) to the prediction of sensor values for fault detection and diagnosis of the Space Shuttle's Main Engines (SSME). The location of the Radial Basis Function (RBF) node centers was determined with a K-means clustering algorithm. A neighborhood operation about these center points was used to determine the variances of the individual processing nodes.

14. SUBJECT TERMS

Neural networks; Space shuttle main engine

15. NUMBER OF PAGES

63

16. PRICE CODE

A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	