MODIFICATION OF RBF NETWORK ARCHITECTURE

Mohd Yusoff Mashor

School of Electrical and Electronic Engineering
University Science Malaysia, Perak Branch Campus
31750 Tronoh, Perak
Malaysia

ABSTRACT

This paper introduces a modified RBF network with additional linear input connections together with a hybrid training algorithm. The training algorithm is based on k -means clustering with square root updating method and Given least squares algorithm with additional linear input connections features. The capability of the proposed RBF network architecture and the new hybrid algorithm was demonstrated using simulated and real data sets. The proposed RBF network provides significant improvement over standard RBF network especially when the numbers of hidden nodes are small. These additional linear input connections do not significantly increase the complexity of RBF network since only a few connections are required. In fact by using the additional linear input connections, the number of hidden nodes required for the normal RBF network model can be reduced, which in turn reduce computational load.

1. INTRODUCTION

Neural networks have theoretically been proved to be capable of representing complex non-linear mappings^{1,2,3}. Multilayered feed forward neural networks offer an alternative to conventional modelling methods, and networks with various training laws have successfully been used to model non-linear systems⁴⁻¹⁴. Modelling using a feed forward neural network can be viewed as performing a curve fitting operation in a multi-dimensional space. Thus, training is a process of producing a best fit surface in a multi-dimensional space to a training data set and prediction is an interpolation of the testing data set on the fitted surface. In general, neural network models are highly non-linear in the unknown-parameters. A drawback of this property is that the network training algorithm must be based on a non-linear optimisation technique which is associated with the following problems:

- slow parameter convergence,
- intensive computation, and
- poor local minima.

Radial Basis Function (RBF) networks that overcome some of these problems have been introduced by several authors^{15,16}. RBF is a traditional technique of strict

interpolation in multi-dimensional space that is capable of representing almost any system up to a certain accuracy¹⁶. Broomhead and Lowe⁸ showed that if the strict interpolation restriction is relaxed, the RBF can be realised as a special one-hidden-layer feed forward neural network. Once the centres, widths and the non-linear functions have been fixed, the adjustable weights of the network (between the hidden and output layer) can be estimated using a class of linear least squares techniques. Thus, a faster and efficiently-trained-network can be achieved while partially avoiding the problem of local minima. Tsoi¹⁷ presented a full derivation of how the RBF technique can be used as a multilayered feed forward neural network.

Training algorithms for RBF networks normally comprise of a procedure to position the RBF centres and a linear least squares technique to estimate the weights. The *orthogonal least squares* algorithm can be employed to train the RBF network⁹⁻¹¹. This algorithm will automatically select the appropriate RBF centres from the training data and estimate the weights. Moody and Darken¹⁸ proposed a hybrid algorithm based on *k*-means clustering to position the centres and least mean squares to estimate the weights in real-time. Chen et al¹² introduced a recursive hybrid algorithm based on *k*-means clustering algorithm to position the centres and Givens least squares to estimate the weights.

In the present study, the convergence properties of the hybrid algorithms are further improved by proposing a RBF network with additional linear input connections. As an alternative to the hybrid algorithms^{12,18}, a new hybrid algorithm based on *k*-means clustering using square root updating method and Givens least squares algorithm (with additional linear input connections features) is proposed to train RBF networks. Givens least squares has been selected because of the superior numerical stability and accuracy of this method¹⁹. The capability of the RBF network trained using the new hybrid algorithm was tested using simulated and real data.

2. RADIAL BASIS FUNCTION NETWORKS

A RBF network as illustrated in Figure 1 is a one-hidden-layered neural network. The first layer consists of input nodes connected to a hidden layer via a set of unity connections. The output of each hidden node or radial basis function node is given by:

$$z_j(t) = \phi(\|v(t) - c_j(t)\|); \quad j=1, \dots, n_h \quad (1)$$

where $c_j(t)$ and n_h represent the RBF centres and number of hidden nodes or centres respectively, $v(t)$ is the input vector to the RBF network composed of lagged input and lagged output and $\phi(\bullet)$ is a non-linear basis function. $\|\bullet\|$ denotes a distance measure that is normally taken to be the Euclidean norm. The response of the output nodes is governed by the following equation.

$$y_i(t) = w_{i0} + \sum_{j=1}^{n_h} w_{ij} z_j(t); \quad i=1, \dots, m \quad (2)$$

where w_{ij} , w_{i0} and m are the connection weights, bias connection weights and number of outputs respectively. Therefore, a RBF network with m outputs and n_h hidden nodes can be expressed as:

$$y_i(t) = w_{i0} + \sum_{j=1}^{n_h} w_{ij} \phi(\|v(t) - c_j(t)\|); \quad i=1, \dots, m \quad (3)$$

The weights, w_{ij} of the RBF network appear linearly as in Equation 3, hence after the centres and non-linear function have been fixed the weight estimation problem reduces to a linear least squares optimisation problem. This is the main advantage of a RBF network over a multilayered perceptron network. Since the weight estimation process is based on linear optimisation techniques, the weights will not be trapped at poor local minima whereas in multilayered perceptron networks this problem often becomes the main obstacle. Furthermore, the linear optimisation technique will ensure fast and efficient weight convergence.

Figure 1. Radial Basis Function Network.

The non-linear function $\phi(\bullet)$ can be selected from a set of non-linear basis functions and in principle each hidden node can have different non-linear functions. The typical choices of the non-linear function are linear, cubic, thin-plate-spline, multi-quadratic, inverse multi-quadratic and Gaussian functions. In the present study, the thin-plate-spline function has been selected as the non-linear function because this function has a good modelling capability¹⁶. Thin-plate-spline is given by:

$$\phi(a) = a^2 \log(a) \quad (4)$$

where $a = \|v(t) - c_j(t)\|$. The thin-plate-spline function has been used in references⁹⁻¹², the Gaussian function in references^{15,18,20} and the inverse multi-quadratic in reference²¹.

3. RADIAL BASIS FUNCTION NETWORKS WITH LINEAR INPUT CONNECTIONS

Since neural networks are highly non-linear, even a linear system has to be approximated using the non-linear neural network model. However, modelling a linear system using a non-linear model can never be better than using a linear model. Considering this argument, in the present study an RBF network with linear input connections is proposed. The proposed network allows the network inputs to be connected directly to the output node via weighted connections to form a linear model in parallel with the non-linear original RBF model as shown in Figure 2.

Figure 2. The RBF network with linear input connections.

The new RBF network with m outputs, n inputs, n_h hidden nodes and n_l linear input connections can be expressed as:

$$y_i(t) = w_{i0} + \sum_{j=1}^{n_l} \lambda_{ij} v_l(t) + \sum_{j=1}^{n_h} w_{ij} \phi(\|v(t) - c_j(t)\|); \quad i=1,2,\dots,m \quad (5)$$

where the λ 's and vl 's are the weights and the input vector for the linear connections respectively. The input vector for the linear connections may consist of past inputs, outputs and noise lags. Since the λ 's (the linear connection weights) appear to be linear within the network, the λ 's can be estimated using the same algorithm as for the w 's. In fact both the λ and w values can be estimated at the same time by rearranging Equation 5 to give:

$$y_i(t) = w_{i0} + \sum_{j=1}^{h_i+n_i} \Gamma_{ij} v(t); \quad i=1,2,\dots,m \quad (6)$$

$$\text{where } \Gamma_{ij} = \begin{cases} w_{ij} & \text{if } j \leq n_h \\ \lambda_{ij} & \text{if } j > n_h \end{cases} \quad \text{and} \quad V(t) = \begin{cases} z_j(t) & \text{if } j \leq n_h \\ vl(t) & \text{if } j > n_h \end{cases}$$

Notice that Equation 6 is in the same form as Equation 2 and hence any algorithm that can be used to estimate the w 's in Equation 2 can also be used to estimate Γ_{ij} in Equation 6. Since the additional linear connections only introduce a linear model, no significant computational load is added to the standard RBF network training. Furthermore, the number of required linear connections are normally much smaller than the number of hidden nodes in the RBF network.

4. MODELLING NON-LINEAR SYSTEMS USING RBF NETWORKS

There are a number of studies that have been accomplished on modelling non-linear systems using radial basis function networks. Lowe²², Potts and Broomhead²³, Katayama et al²⁴ and Longinov²⁵ used RBF networks to model non-linear time series. Modelling of SISO non-linear systems has been reported in references⁹⁻¹². Modelling using RBF networks can be considered as fitting a surface in a multi-dimensional space to represent the training data set and using the surface to predict over the testing data set. Therefore, RBF networks require all the future data of the system to lie within the domain of the fitted surface to ensure a correct mapping so that good predictions can be achieved. This is normal for the non-linear modelling where the model is only valid over a certain amplitude range.

A wide class of non-linear systems can be represented by the non-linear auto-regressive moving average with exogenous input (NARMAX) model²⁶. The NARMAX model can be expressed in terms of a non-linear function expansion of lagged input, output and noise terms as follows:

$$y(t) = f_s(y(t-1), \dots, y(t-n_y), u(t-1), \dots, u(t-n_u), e(t-1), \dots, e(t-n_e)) + e(t) \quad (7)$$

where

$$y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_m(t) \end{bmatrix}, \quad u(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_r(t) \end{bmatrix} \quad \text{and} \quad e(t) = \begin{bmatrix} e_1(t) \\ \vdots \\ e_m(t) \end{bmatrix}$$

are the system output, input and noise vector respectively; n_y , n_u and n_e are the maximum lags in the output, input and noise vector respectively.

The non-linear function $f_s(\bullet)$ is normally very complicated and rarely known a priori for practical systems. If the mechanism of a system are known the function $f_s(\bullet)$ can be derived from the function that govern those mechanism. In the case of an unknown system, $f_s(\bullet)$ is normally constructed based on the observation of the input and output data. In the present study, RBF networks will be used to model the input-output relationship. In other words, $f_s(\bullet)$ will be modelled by using Equation 3 where $\phi(\bullet)$ is chosen to be the thin-plate-spline function. The network input vector, $v(t)$ is formed from legged input, output and noise terms which are denoted as $u(t-1)\cdots u(t-n_u)$, $y(t-1)\cdots y(t-n_y)$ and $e(t-1)\cdots e(t-n_e)$ respectively in Equation 7.

Another method to include a noise model in a RBF network is to use only linear noise connections as in Figure 2. This approach can reduce the complexity of the RBF network and hence accelerate the training process. However, this approach only allows linear noise models and may not be sufficient if the data is highly corrupted by non-linear noise.

5. THE NEW HYBRID ALGORITHM

Given a set of input-output data, $u(t)$ and $y(t)$, ($t=1,2,\dots,N$), the connection weights, centres and widths may be obtained by minimising the following cost function:

$$J = \sum_{t=1}^N (y(t) - \hat{y}(t))^T (y(t) - \hat{y}(t)) \quad (8)$$

where $\hat{y}(t)$ is the predicted output generated by using the RBF network given by Equation 3. Equation 8 can be solved using a non-linear optimisation or gradient descent technique. However, estimating the weights using such algorithm will destroy the advantage of linearity in the weights. Thus, the training algorithm is normally split into two parts:

- (i) locate the RBF centres, $c_j(t)$ and
- (ii) estimate the weights, w_{ij} .

This approach will allow an independent algorithm to be employed for each task. The centres are normally located using an unsupervised algorithm such as k-means clustering, fuzzy clustering and Gaussian classifier whereas the weights are normally estimated using a class of linear least squares algorithm. Moody and Darken¹⁸ used the k-means clustering method to position the RBF centres and a least means squares routine to estimate the weights, Chen et al¹² used a k-means clustering to positioned the centres and a Givens least squares algorithm to estimate the weights. In the present study, the k-means clustering using square root updating method is used to position the RBF centres and a Givens least

squares algorithm with additional linear input connections features will be used to estimate the weights. A detailed description of the k -means clustering using square root updating method can be found in Darken and Moody²⁷.

After the RBF centres and the non-linear functions have been selected, the weights of the RBF network can be estimated using a least squares type algorithm. In the present study, exponential weighted least squares were employed based on the Givens transformation. The estimation problem using weighted least squares can be described as follows:

Define a vector $z(t)$ at time t as:

$$\mathbf{z}(t) = [z_1(t), \dots, z_{n_h}(t)] \quad (9)$$

where are the output of the hidden nodes vector and the number of hidden nodes to the RBF network respectively. If linear input connections are used, Equation 9 should be modified to include linear terms as follows:

$$\mathbf{z}(t) = [z_1(t) \quad \dots \quad z_{n_h}(t) \quad zl_1(t) \quad \dots \quad zl_{n_l}(t)] \quad (10)$$

where zl 's is the linear input connection vector vl in Figure 2. Any vector or matrix size n_h should be increased to $n_h + n_l$ in order to accommodate the new structure of the network. A bias term can also be included in the RBF network in the same way as the linear input connections.

Define a matrix $\mathbf{Z}(t)$ at time t as:

$$\mathbf{Z}(t) = \begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(t) \end{bmatrix}; \quad (11)$$

and an output vector, $\mathbf{y}(t)$ given by:

$$\mathbf{y}(t) = [y(1), \dots, y(t)]^T, \quad (12)$$

then the normal equation can be written as:

$$\mathbf{y}(t) = \mathbf{Z}(t)\Theta(t) \quad (13)$$

where $\Theta(t)$ is a coefficient vector given by:

$$\Theta(t) = [w_1(t), \dots, w_{n_h}(t)]^T \quad (14)$$

The weighted least squares algorithm estimates $\Theta(t)$ by minimising the sum of weighted squared errors, defined as:

$$e(t)_{wls} = \sum_{i=1}^t \beta(t-1) [\mathbf{y}(i) - \mathbf{Z}(i-1)\Theta(t)]^2 \quad (15)$$

where β , $0 < \beta < 1$, is an exponential forgetting factor. The solution for the Equation 13 is given by

$$\Theta(t) = [\mathbf{Z}^T(t) \mathbf{Q}(t) \mathbf{Z}(t)]^{-1} \mathbf{Z}^T(t) \mathbf{Q}(t) \mathbf{y}(t) \quad (16)$$

where $\mathbf{Q}(t)$ is $n_t \times n_t$ diagonal matrix defined recursively by

$$\mathbf{Q}(t) = [\beta(t) \mathbf{Q}(t-1) \quad \mathbf{1}], \quad \mathbf{Q}(1) = \mathbf{1}; \quad (17)$$

and $\beta(t)$ and n_t are the forgetting factor and the number of training data at time t respectively.

Many solutions have been suggested to solve the weighted least squares problem (16) such as recursive modified Gram Schmidt, fast recursive least squares, fast Kalman algorithm and Givens least squares. In the present study, Givens least squares without square roots was used. The application of the Givens least squares algorithm to adaptive filtering and estimation have stimulated much interest due to superior numerical stability and accuracy¹⁹.

Given Least Squares without square roots is summarised below. Introduce a $n_t \times n_k$ diagonal matrix $\mathbf{D}(t)$ as:

$$\mathbf{D}(t) = \text{diag} [d_1(t) \quad \cdots \quad d_{(n_k-1)}(t) \quad \sigma_\epsilon^2(t)]; \quad n_k = n_h + 1 \quad (18)$$

where n_h d 's and $\sigma_\epsilon(t)$ are the number of the estimated RBF weights, the least squares estimation errors and the standard deviation at time t respectively. Define a $n_k \times n_k$ dimensional upper triangular matrix $\mathbf{R}(t)$ as:

$$\mathbf{R}(t) = \begin{bmatrix} 1 & r_{12}(t) & r_{13}(t) & \cdots & \cdots & r_{1n_k}(t) \\ 0 & 1 & r_{23}(t) & \ddots & \ddots & r_{2n_k}(t) \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & r_{(n_k-1)n_k}(t) \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix} \quad (19)$$

where the r 's are the least squares estimated coefficients after estimating $x_k^{(i)}(t)$ from $x_k^{(i-1)}(t)$.

The Givens Least Squares algorithm solves for $\Theta(t)$ in equation (16) by performing a Givens transformation:

$$\begin{bmatrix} \mathbf{D}^{1/2}(t-1) & [\mathbf{R}(t-1)] \\ (\delta^{(0)})^{1/2} & [x_1^{(3)}(t) \dots x_{n_k}^{(3)}(t)] \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{D}^{1/2}(t) & [\mathbf{R}(t)] \\ 0 & [0 \dots 0] \end{bmatrix} \quad (20)$$

where

$$[x_1^{(0)}(t) \dots x_{n_k}^{(0)}(t)] = [z_1(t) \dots z_{n_h}(t) y(t)] \quad (21)$$

and $\delta^{(0)}$ is called a maximum likelihood factor¹⁹, initialised to $1/\beta(t)$, where $\beta(t)$ is updated using Equation 27.

After i -th steps, Given transformation transfers

$$\begin{bmatrix} 0 & \dots & 0 & d_i^{1/2}(t-1) & d_i^{1/2}(t-1)r_{ii+1}(t-1) & \dots & d_i^{1/2}(t-1)r_{in_k}(t-1) \\ 0 & \dots & 0 & (\delta^{(i-1)})^{1/2}x_i^{(i-1)}(t) & (\delta^{(i-1)})^{1/2}x_{i+1}^{(i-1)}(t) & \dots & (\delta^{(i-1)})^{1/2}x_{n_k}^{(i-1)}(t) \end{bmatrix} \quad (22)$$

into

$$\begin{bmatrix} 0 & \dots & 0 & d_i^{1/2}(t) & d_i^{1/2}(t)r_{ii+1}(t) & \dots & d_i^{1/2}(t)r_{in_k}(t) \\ 0 & \dots & 0 & 0 & (\delta^{(i)})^{1/2}x_{i+1}^{(i)}(t) & \dots & (\delta^{(i)})^{1/2}x_{n_k}^{(i)}(t) \end{bmatrix} \quad (23)$$

The explicit computation of Givens transformation can be expressed as follows:

$$\left. \begin{aligned} d_i(t) &= d_i(t-1) + \delta^{(i-1)}(x_i^{(i-1)}(t))^2 \\ c &= \frac{d_i(t-1)}{d_i(t)} \\ b &= \delta^{(i-1)} \frac{x_i^{(i-1)}(t)}{d_i(t)} \\ \delta^{(i)} &= c\delta^{(i-1)} \end{aligned} \right\} \quad (24)$$

$$\left. \begin{aligned} x_k^{(i)}(t) &= x_k^{(i-1)}(t) - x_k^{(i-1)}(t)r_{ik}(t-1) \\ r_{ik}(t) &= cr_{ik}(t-1) + bx_k^{(i-1)}(t) \end{aligned} \right\} \quad k = i+1, \dots, n_k \quad (25)$$

The algorithm is initialised by setting

$$\begin{aligned} \sigma_e(0) &= 0 \\ \mathbf{R}(0) &= \mathbf{I} \\ \mathbf{D}(0) &= \frac{1}{\rho} \end{aligned} \quad (26)$$

where $\sigma_e(0)$ and ρ are the initial standard deviation of the estimated parameters and a large positive scalar respectively; and $\mathbf{I}$ is an $n_k \times n_k$ identity matrix. The forgetting factor $\beta(t)$ is normally computed according to:

$$\beta(t) = \beta_0 \beta(t-1) + 1 - \beta_0 \quad (27)$$

where β_0 and $\beta(0)$ are typically chosen to be less but close to one and β_0 is normally larger than $\beta(0)$

After Givens transformation is completed, the estimated parameter vector, $\Theta(t)$ can be calculated by back substitution.

$$w_{(n_k-1)}(t) = r_{(n_k-1)n_k}(t) \quad (28)$$

and

$$w_i(t) = r_{in_k}(t) - \sum_{j=i+1}^{(n_k-1)} r_{ij}(t) w_j(t); \quad i = (n_k - 2), (n_k - 3), \dots, 1 \quad (29)$$

The RBF network using the hybrid algorithm based on k -means clustering using square root updating method and Givens Least Squares with additional linear input connections feature can be used to model non-linear systems. Initially the RBF centres are positioned using the k -means clustering using square root updating method and then the Givens least squares algorithm is used to estimate the weights of the RBF network, w 's.

6. MODEL VALIDATION

There are several ways of testing a model such as one step ahead predictions (OSA), model predicted outputs (MPO), mean squared error (MSE), correlation tests, and chi-squares tests. In the present study the first four tests were used to justify the performance of the fitted models. OSA is a common measure of predictive accuracy of a model that has been considered by many researchers. OSA can be expressed as:

$$\hat{y}(t) = f_s(u(t-1), \dots, u(t-n_u), y(t-1), \dots, y(t-n_y), \varepsilon(t-1, \hat{\theta}), \dots, \varepsilon(t-n_\varepsilon, \hat{\theta})) \quad (30)$$

and the residual or prediction error is defines as

$$\hat{\varepsilon}(t, \hat{\theta}) = y(t) - \hat{y}(t) \quad (31)$$

where $f_s(\bullet)$ is a non-linear function, in this case the RBF network.

Another test that often gives a better measurement of the fitted model predictive capability is the model predicted output. Generally model predicted output can be expressed as:

$$\hat{y}_d(t) = f_s(u(t-1), \dots, u(t-n_u), \hat{y}_d(t-1), \dots, \hat{y}_d(t-n_y), 0, \dots, 0) \quad (32)$$

and the deterministic error or deterministic residual is

$$\varepsilon_d(t) = y(t) - \hat{y}_d(t) \quad (33)$$

A good model will normally give a good prediction, however a model that has a good one step ahead prediction and model predicted output may not always be unbiased. The model may be significantly biased and prediction over a different set of data often reveals this problem. This can be tested by splitting the data into two sets, a training set and a testing set.

MSE is an iterative method of model validation where the model is tested by calculating the mean squared errors after each training step. MSE test will indicate how fast a prediction error or residual converges with the number of training data. A good model normally converges rapidly and has a relatively low final MSE value. The mean squared error at the t -th training step, is given by

$$E_{MSE}(t, \Theta(t)) = \frac{1}{n_d} \sum_{i=1}^{n_d} (y(i) - \hat{y}(i, \Theta(t)))^2 \quad (34)$$

where $E_{MSE}(t, \Theta(t))$ and $\hat{y}(i, \Theta(t))$ are the mean squared error and one step ahead prediction for a given set of estimated parameters $\Theta(t)$ after t training steps respectively, and n_d is the number of data that were used to calculate the MSE. The data for calculating MSE can be the same as the training data set or testing data set. In general, a good model should produce a good MSE but this cannot always be used to imply that the model is good. For instance, an overfitted model will normally produce a good MSE although the model cannot predict very well and may be biased. This can be tested by splitting the data into two sets, a training set and a testing set. If the MSE generated using both the training and the testing data sets are good then more confidence can be given to the model estimation.

An alternative method of model validation is to use correlation tests to determine if there is any predictive information in the residual after model fitting²⁸. The residual defined in Equation 31 will be unpredictable from all linear and non-linear combinations of past inputs and outputs if the following hold:

$$\begin{aligned} \Phi_{\varepsilon\varepsilon}(\tau) &= E[\varepsilon(t-\tau)\varepsilon(t)] = \delta(t) && \text{for all } \tau \\ \Phi_{u\varepsilon}(\tau) &= E[u(t-\tau)\varepsilon(t)] = 0 && \text{for all } \tau \\ \Phi_{\varepsilon(u)}(\tau) &= E[\varepsilon(t)(\varepsilon(t-\tau))] = 0 && \text{for all } \tau \geq 0 \\ \Phi_{u^2\varepsilon}(\tau) &= E[(u^2(t-\tau) - \bar{u}^2(t))\varepsilon^2(t)] = 0 && \text{for all } \tau \\ \Phi_{u^2\varepsilon^2}(\tau) &= E[(u^2(t-\tau) - \bar{u}^2(t))\varepsilon^2(t)] = 0 && \text{for all } \tau \end{aligned} \quad (35)$$

where $\bar{u}^2(t)$ and $E[\bullet]$ are the mean value of $u^2(t)$ and the expectation respectively. In practice if the correlation tests lie within the 95% confidence limits $\pm 1.96/\sqrt{N}$ then the model is regarded as adequate, where N is the number of data used to train the network.

7. APPLICATION EXAMPLES

The RBF network trained using the hybrid algorithm based on k -means clustering using square root updating method and Given least squares algorithm with additional linear input connections features, derived in Section 5 was used to model three systems. In all the examples the thin-plate-spline was selected as the non-linear function in the RBF network and the centres were initialised to the first few samples of the input-output data. During the calculation of the mean squared error, the noise model was excluded from the model since the noise model will normally cause the mean squared error to become unstable in the early stage of training. This is because if the noise model is included, the MSE will consist of model predicted output of the predicted noise, $\hat{\varepsilon}(t)$ and one step ahead predicted output of the process model. Since the process model is very bad at the early training stage, $\hat{\varepsilon}(t)$ will be very large. So model predicted output of the noise model based on $\hat{\varepsilon}(t)$ may become unstable. In all examples, all the 1000 data were used to calculate the MSE and the designing parameters were taken as $\rho = 1000.0$, $\beta_0 = 0.99$, and $\beta(0) = 0.95$.

Example 1

System S1 was a simulated system defined by the following difference equation S1:- $y(t) = 0.3y(t-1) + 0.6y(t-2) + u^3(t-1) + 0.3u^2(t-1) - 0.4u(t-1) + e(t)$ where $e(t)$ was a Gaussian white noise sequence with zero mean and variance 0.05 and the input, $u(t)$ was a uniformly random sequence $[-1, +1]$. System S1 was used to generate 1000 pairs of data input and output. The first 600 data were used to train the network and the remaining 400 data were used to test the fitted model.

The RBF centres were initialised to the first few samples of input and output data and the network was trained based on the following configuration

$$\begin{aligned} v(t) &= [u(t-1) \quad y(t-1) \quad y(t-2)] \\ vl(t) &= [u(t-1) \quad y(t-1) \quad y(t-2)] \quad n_h = 30 \end{aligned}$$

One step ahead prediction and model predicted output of the fitted model over both the training and testing data sets are shown in Figures 3 and 4 respectively. These plots show that the model predicts very well over both training and testing data sets. The correlation tests in Figure 5 are very good, all the correlation tests are inside the 95% confidence limits. The evolution of the mean squared error obtained from the fitted model is shown in Figure 6. During the learning process, the mean squared error of the RBF network model was reduced from an initial value of 6.52dB to a noise floor of -21.17dB. The good prediction, MSE and correlation tests suggest that the model is unbiased and adequate to represent the identified system.

Figure 3. One step ahead prediction superimposed on actual output for Example 1.

Figure 4. Model predicted output superimposed on actual output for Example 1.

Figure 5. Correlation tests for Example 1.

Figure 6. Mean square error for Example 1.

Example 2

The second data set, S2 was taken from a heat exchanger system and consists of 1000 samples. A detailed description of the process can be found in Billings and Fadhil²⁹. The first 500 data were used to identify the system and the remaining 500 data were used to test the fitted RBF network model. The RBF centres were initialised to the first few data samples and the network has been trained using the following specifications:

$$v(t) = [u(t-1) \quad u(t-2) \quad y(t-1) \quad y(t-4)] \text{ with a bias input}$$

$$vl(t) = [u(t-1) \quad u(t-2) \quad y(t-1) \quad y(t-4) \quad e(t-3) \quad e(t-4) \quad e(t-5)]$$

$$nh = 20$$

Notice that $v_l(t)$ denotes the linear input connections vector and the e 's represent the linear noise terms. One step ahead prediction and model predicted outputs generated by the network model over both the training and testing data sets are shown in Figures 7 and 8 respectively. The plots show that the model predicts very well over both the training and testing data sets. Correlation tests shown in Figure 9 are quite reasonable where $\Phi_{u2e^2}(\tau)$ and $\Phi_{u2e}(\tau)$ plots are marginally outside the 95% confidence limits. The evolution of MSE obtained from the network model is shown in Figure 10. During the learning process, the MSE was reduced from 19.6dB to the final value of -14.6dB. As the model predicts very well and has reasonable correlation tests, the model was considered to be sufficient to represent the identified system.

Figure 7. One step ahead prediction superimposed on actual output for Example 2.

Figure 8. Model predicted output superimposed on actual output for Example 2.

Figure 9. Correlation tests for Example 2.

Figure 10. Mean square error for Example 2.

Example 3

A data set of 1000 input-output samples were taken from system S3 which is a tension leg platform. The first 600 data were used to train the network and the rest were used to test the fitted network model. The RBF centres were initialised to the first few input-output data samples and the network was trained using the following structure

$$v(t) = [u(t-1) \ u(t-3) \ u(t-4) \ u(t-6) \ u(t-7) \ u(t-8) \ u(t-11) \ y(t-1) \ y(t-3) \ y(t-4)]$$

$$v(t) = [y(t-1) \ y(t-2) \ e(t-1) \ e(t-2) \ e(t-3) \ e(t-5)] \text{ with bias input; } n_h = 40$$

One step ahead prediction and model predicted output generated by the fitted model are shown in Figures 11 and 12 respectively. The plots show that the model predicts reasonably over both the training and testing data sets. All the correlation tests, shown in Figure 13,

are well inside the 95% confidence limits except for $\Phi_{u^2, \varepsilon^2}(\tau)$ which is marginally outside the confidence limits at lag 7. The evolution of the MSE plot is shown in Figure 14. During the learning process, the MSE was reduced from 16.1dB initially to a final value of -3.1dB. Since the model predicts reasonably and has good correlation tests, the model can be considered as an adequate representation of the identified system.

Figure 11. One step ahead prediction superimposed on actual output for Example 3.

Figure 12. Model predicted output superimposed on actual output for Example 3.

Figure 13. Correlation tests for Example 3.

Figure 14. Mean square error for Example 3.

8. ADVANTAGES OF THE RBF NETWORKS WITH LINEAR INPUT CONNECTIONS

The performance of the proposed RBF architecture (RBF network with additional linear input connections) can be demonstrated using the data sets that have been used in the previous section. The MSE will be used for this comparison since it is hard to notice the performance advantage by using other model validity tests. Both of the standard RBF network and the proposed RBF network were used the same structure except that the proposed RBF network will have additional linear input connections, $v_l(t)$. The non-linear function was selected to be a thin-plate-spline function and the RBF centres were initialised to the first few samples of the input-output data. In this comparison, the network specifications are the same as in the previous section except that the number of centres is varied.

Figure 15. MSE generated using normal RBF network and the modified RBF network for Example 1.

The MSE plots for the Examples 1, 2 and 3 are shown in Figures 15, 16 and 17 respectively. These results show that a few additional linear input connections can significantly improve the performance of the RBF network. The performance improvement is very remarkable when the number of RBF centres are small. The plots also show that by using the additional linear input connections the modified RBF network can produce the same result as the normal RBF network with a larger number of hidden nodes or centres. Therefore, the RBF network with additional linear input connections can significantly reduce the required number of hidden nodes which will shorten the training time.

Figure 16. MSE generated using the normal RBF network and the modified RBF network for Example 2.

Figure 17. MSE generated using the normal RBF network and the modified RBF network for Example 3.

9. CONCLUSION

A RBF network with additional linear input connections and a hybrid training algorithm based on square roots k -means clustering and Givens least squares (with additional linear input connections feature) has been introduced. Three examples were used to test the efficiency of the modified RBF networks and the hybrid training algorithm. In all the examples the fitted RBF network models yield good predictions and correlation tests. Hence, the proposed modified RBF networks and the hybrid training algorithm is considered as an adequate representation of the identified system.

The mean squared error plots in Section 8 proved that the RBF networks with additional linear input connections significantly improve RBF network performance. The improvement is remarkable when the numbers of RBF centres are small. Since the additional linear input connections are connected directly to the output node and only a few are required, the linear input connections do not significantly increase the complexity of the normal RBF network. In fact by using the additional linear input connections the number of hidden nodes required for the normal RBF network model can be reduced which will also reduce computational load.

10. REFERENCES

1. Cybenko, G., 1989, "Approximations by superposition of a sigmoidal function", *Mathematics of Control, Signal and Systems*, 2, 303-314.
2. Funahashi, K., 1989, "On the approximate realisation of continuous mappings by neural networks", *Neural Networks*, 2, 183-192.
3. Hornik, K., 1991, "Approximation capabilities of multilayer feed forward networks", *Neural Networks*, 4, 251-257.

4. Back, A.D., and Tsoi, A.C., 1992, "Non-linear system identification using multilayer perceptrons with locally recurrent synaptic structure", *Neural Networks for Signal Processing II, Proc. of the IEEE-SP Workshop*, 444-453.
5. Bhat, N.V., Minderman, Jr.P.A., McAvoy, T., and Wang, N.S., 1990, "Modelling chemical process system via neural computation", *IEEE Control Systems*, (April).
6. Billings, S.A., and Chen S., 1989, "Extended model set, global data and threshold model identification of severely non-linear systems", *Int. J. Control*, 50, 1897-1923.
7. Billings, S.A., Jamaluddin, H.B. and Chen, S., 1990, "A comparison of the back propagation and recursive prediction error algorithms for training neural networks", *Mechanical systems and signal processing*, 5, 233-255.
8. Billings, S.A., Jamaluddin, H.B. and Chen, S., 1992, "Properties of neural networks with applications to modelling non-linear dynamical systems", *Int. J. of Control*, 55, 193-224.
9. Chen, S., Billings, S.A., Cowan, C.F.N., and Grant, P.M., 1990, "Practical identification of NARMAX models using radial basis functions", *Int. J. Control*, 52, 1327-1350.
10. Chen, S., Billings, S.A., Cowan, C.F.N., and Grant, P.M., 1990, "Non-linear systems identification using radial basis functions", *Int. J. Systems Science*, 21, 2513-2539.
11. Chen, S., Cowan, C.F.N., and Grant, P.M., 1991, "Orthogonal least squares learning algorithm for radial basis function networks", *IEEE Trans. on Neural Networks*, 2, 302-309.
12. Chen, S., Billings, S.A. and Grant, P.M., 1992, "Recursive hybrid algorithm for non-linear system identification using radial basis function networks", *Int. J. of Control*, 55, 1051-1070.
13. Elanayar V.T., S., and Shin, Y.C., 1994, "Radial basis function neural network for approximation and estimation of non-linear stochastic dynamic systems", *IEEE Trans. on Neural Networks*, 5 (4), 594-603.
14. Fernandez, B., Parlos, A.G., Tsai, W.K., 1990, "Non-linear dynamic system identification using artificial neural networks: The discrete time case", *Proc. of the Fourth Annual Parallel Processing Symp.*, 1, 192-227.
15. Broomhead, D.S., and Lowe, D., 1988, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, 2, 321-355.
16. Powell, M.J.D., 1987, "Radial basis function approximations to polynomials", *Proc. 12th Biennial Numerical Analysis Conf.*, Dundee, 223-241.
17. Tsoi, A.C., 1989, "Multilayer perceptron trained using radial basis functions", *Elec. Letters*, 25, 1296-1297.
18. Moody, J., and Darken, C.J., 1989, "Fast learning in neural networks of locally-tuned processing units", *Neural Computation*, 1, 281-294.
19. Ling, F., 1991, "Givens rotation based on least squares lattice and related algorithms", *IEEE Trans. on Signal Processing*, 39, 1541-1551.
20. Poggio, T., and Girosi, F., 1990, "Network for approximation and learning", *Proc. of IEEE*, 78 (9), 1481-1497.
21. Pottmann, M., and Seborg, D.E., 1992, "Identification of non-linear processes using reciprocal multiquadratic functions", *J. of Process Control*, 2 (4), 189-203.
22. Lowe, D., 1989, "Adaptive radial basis function non-linearities and the problem of generalisation", *IEE Conf. on Artificial Intelligence and Neural Networks*.
23. Potts, M.A.S., and Broomhead, D.S., 1991, "Time series prediction with a radial basis

- function neural network", *Proc. of the SPIE - The Int. Society for Optical Eng.*, 1565, 255-266.
24. Katayama, R., Kajitani, Y., Kuwata, K., and Nishida, Y., 1993, "Self generating radial basis function as neuro-fuzzy model and its application to non-linear prediction of chaotic time series", *Second IEEE Int. Conf. on Fuzzy Systems*, 1, 407-414.
 25. Longinov, N.E., 1994, "Predicting pilot look-angle with a radial basis function network", *IEEE Trans. on Systems, Man and Cybernetics*, 24 (10), 1511-1518.
 26. Leontaritis, I.J., and Billings, S.A., 1985, "Input-output parametric models for non-linear systems. Part I - Deterministic non-linear systems. Part II - Stochastic non-linear systems", *Int. J. Control*, 41, 303-359.
 27. Darken, C., and Moody, J., 1990, "Fast adaptive k -means clustering: Some empirical results", *Int. Joint Conf. on Neural Networks*, 2, 233-238.
 28. Billings, S.A., and Voon, W.S.F., 1986, "Structure detection and model validity tests in the identification of non-linear systems", *Proc. IEE, Part D*, 127, 272-285.
 29. Billings, S.A., and Fadhil, M.B., 1985, "The practical identification of system with non-linearities", *Proc. 7th IFAC Symp. on Identification and System Parameter Estimation*, York, U.K., 155-160.