A NEW FAST ALGORITHM FOR IDENTIFICATION OF NON-LINEAR DYNAMIC SYSTEMS USING RADIAL BASIS FUNCTION NETWORKS

K. Li

Intelligent Systems and Control Group
School of Electrical and Electronic Engineering
Queen's University Belfast
Ashby Building, Stranmillis Rd., Belfast BT9 5AH, UK
E-mail: K.Li@qub.ac.uk; Fax: +44 (0)28 90 667023

Keywords: System identification, non-linear system, radial basis function network, network construction

Abstract

A new fast estimation algorithm is derived for identification of non-linear dynamic systems using radial basis function networks. The new algorithm is able to both select the hidden nodes and to compute weights of the output layer in the radial basis function neural networks (RBFN) simultaneously in a stepwise forward fashion.

1 Introduction

Radial basis function (RBF) neural networks [1,2,4,5,9,14,19,20,21,22,23] have been proved to have the capacity of approximating a wide range of multivariate functions to an arbitrary degree of accuracy under certain conditions [11,19]. RBF networks have also been applied in non-linear system identification and time series prediction [1,2,7,9,20,23].

The theoretical basis of the RBF approach lies in the field of interpolation of multivariate functions. The RBF approach in interpolating a function say $f:\Re^m \to \Re^1$ is to use an interpolating function F which is a linear combination of basis functions:

$$F(x) = \theta_0 + \sum_{j=1}^{p} \theta_i \varphi(\|x - c_i\|) \qquad (1)$$

where $\|\bullet\|$ denotes the Euclidean norm, $\theta_i, i = 0,...,M$ are real numbers, φ are real valued function also called radial basis function. The most popular radial basis function is the Guassian basis functions

$$\varphi(x) = \exp(-\|x - c\|^2 / 2\sigma^2) \qquad (2)$$

with peak at centre $c \in \Re^m$ and decreasing as the distance from the centre increase. σ is a positive real number called the scaling parameter or the width of the radial basis functions. Other radial basis functions are $\varphi(x) = (\|x-c\|^2 + \sigma^2)^{1/2}$ and $\varphi(x) = (\|x-c\|^2 + \sigma^2)^{-(1/2)}$. In this paper, Guassian basis function is used throughout.

The solution to exact interpolation is to use the same number of Gaussian functions as that of the data set, which is impractical for a number of reasons [21], such as sensitivity to noise, and immense computation burden for the solution. Instead, a practical procedure is used, i.e. to approximate the function using much less number of radial basis functions. This method of RBF approximation has been interpreted as a RBF neural network with one hidden layer by Broomhead and Lowe [4], where the activation functions for hidden nodes are RBFs. In this case, the function f can be represented by a RBFN as

$$f(x) = \theta_0 + \sum_{j=1}^{p} \theta_i \varphi(\|x - c_i\|) + \varepsilon(x) \qquad (3)$$

where $\varepsilon(x)$ is the modelling error.

There are different training schemes for Radial basis function neural networks (RBFN) [21], they are one-, two, and three-phase learning schemes. Two-phase learning is very commonly applied, where the training of the hidden layer and that of the output layer is performed separately. Two-phase training is not the optimal training scheme, and a three-phase training scheme by introducing another phase of back-propagation-like training can be used to overcome the problem. Support vector learning [22] for RBFN can be considered as an one-phase method, where only a subset of the training data is used for centres of RBFs and the selection of the data points and the training of weights for output layers are integrated in an optimisation problem. Another training scheme can also be regarded as one-phase training scheme is the orthogonal-least-square method (OLS) [5,6,23], where the selection of hidden nodes and the training of weights for output layers are integrated in one algorithm.

In this paper, a new fast one-phase training algorithm is proposed for identification of non-linear dynamic systems using RBFN, where the selection of hidden nodes and the training of weights for the output layer are integrated in one algorithm. Unlike the OLS method, the algorithm does not apply matrix transformation and decomposition, and it is also efficient in implementation.

2 Review of the OLS method

Given a data set of *N* samples, the RBFN in Equation (3) can be re-formulated as

$$\boldsymbol{Y} = \boldsymbol{\Phi\Theta} + \boldsymbol{\Xi} \tag{4}$$

where

$\boldsymbol{Y}^{\mathrm{T}} = [y_1, y_2, ..., y_N]$, $\boldsymbol{\Phi} = [\varphi_0, \varphi_1, \varphi_2, .., \varphi_p]$,

$\varphi_i = [\varphi_i(x_1),\ \varphi_i(x_2), ..., \varphi_i(x_N)]^{\mathrm{T}}$, $i = 1,2,..,p$

$\varphi_0 = [1, ..., 1]^T$, $\boldsymbol{\Xi}^T = [\varepsilon(x_1), \varepsilon(x_2), ..., \varepsilon(x_N)]$

$\boldsymbol{\Theta} = [\theta_0, \theta_1, \theta_2, .., \theta_p]^{\mathrm{T}}$

The cost function is generally defined as

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{5}$$

where $\hat{y}_i\ i = 1, ..., N$ are the RBFN outputs. (5) can be re-formulated more concisely in the matrix form, which gives

$$\mathrm{E} = (\boldsymbol{\Phi\Theta} - \boldsymbol{Y})^{\mathrm{T}} (\boldsymbol{\Phi\Theta} - \boldsymbol{Y}) \tag{6}$$

The least squares method aims to find the estimates of $\boldsymbol{\Theta}$ that minimizes the cost function, which is, according to Gauss's theorem of least squares [3,12,15, 16,17]

$$\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \left\| \boldsymbol{Y} - \boldsymbol{\Phi\Theta} \right\|_2 = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{Y} \tag{7}$$

where $\left\| \bullet \right\|_2$ denotes the Euclidean norm. $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is also called information matrix and the minimal cost function is calculated as

$$\mathrm{E}(\hat{\boldsymbol{\Theta}}) = \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} - \hat{\boldsymbol{\Theta}}^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{Y} \tag{8}$$

There are several numerical methods to solve $\hat{\boldsymbol{\Theta}}$ and $E(\hat{\boldsymbol{\Theta}})$ in the least-squares problem [10, 12, 15, 18]:

1) To compute $\hat{\boldsymbol{\Theta}}$ by solving $[\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} : \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{Y}]$ using Guassian-Jordan elimination.
2) Perform decomposition of $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ with **LDU** decomposition, where **L** is a unit lower triangular, **U** is the unit triangular, and **D** is the diagonal.
3) Perform **QR** decomposition on $\boldsymbol{\Phi}$, that is $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\Phi} = \boldsymbol{R} = \boldsymbol{DU}$, where **Q** is an orthogonal matrix, $\boldsymbol{R}$ is an upper triangular with nonnegative diagonal elements, **D** is a diagonal with nonnegative diagonal elements, and $\boldsymbol{U}$ is a unit upper triangular. Householder transformations or modified Gram-Schmidt methods are the two examples for **QR** decomposition.
4) Perform singular value decomposition (SVD) on $\boldsymbol{\Phi}$, i.e. $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{V} = \boldsymbol{\Lambda}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal elements are singular values for $\boldsymbol{\Phi}$.

Various matrix decompositions methods (**QR** decomposition on $\boldsymbol{\Phi}$ or **LDU** decomposition on $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$) have been widely used to solve the least-squares problem in signal processing and system identification [18,6]. Matrix decomposition methods can be used simultaneously for parameter estimation as well as for model selection [6] in a forward selection fashion [8]. In particular, **QR** decompositions on $\boldsymbol{\Phi}$ that are numerically implemented by Householder transforms or the modified Gram-Schmidt method have resulted in orthogonal least squares method [6], which is summarized as follows.

By applying an orthogonal transformation to Equation (3) gives

$$\mathrm{f}(\mathrm{x}) = \sum_{\mathrm{i}=1}^{\mathrm{p}} \mathrm{g_i}\psi_{\mathrm{i}}(\mathrm{x}) \tag{9}$$

When Equation (9) is used to fit the two data sets $\boldsymbol{X} = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_{\mathrm{N}}\}$ and $\boldsymbol{Y} = \{\mathrm{y}_1, \mathrm{y}_2, ..., \mathrm{y}_{\mathrm{N}}\}$, the estimated parameters in Equation (9) are given as follows

$$\hat{\boldsymbol{G}} = [\hat{\mathrm{g}}_1 \cdots \hat{\mathrm{g}}_{\mathrm{p}}]^{\mathrm{T}} = [\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi}]^{-1}\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{Y} \tag{10}$$

and

$$\begin{cases} \boldsymbol{\Psi} = \boldsymbol{\Phi}\boldsymbol{A}^{-1} = \left[\psi_1 \quad \cdots \quad \psi_{\mathrm{p}}\right] \\ \psi_{\mathrm{k}} = \left[\psi_{\mathrm{k}}(\mathrm{x}_1) \cdots \psi_{\mathrm{k}}(\mathrm{x}_{\mathrm{N}})\right]^{\mathrm{T}} \end{cases} \tag{11}$$

is an orthogonal regression matrix with the following properties

$$\begin{cases} \boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi} = \mathrm{diag}\left\{ \left\|\psi_1\right\|_2^2 \quad \cdots \quad \left\|\psi_{\mathrm{p}}\right\|_2^2 \right\} \\ (\psi_{\mathrm{i}})^{\mathrm{T}}\psi_{\mathrm{j}} = 0, \mathrm{i} \neq \mathrm{j} \end{cases} \tag{12}$$

and

$$\boldsymbol{A} = \begin{bmatrix} 1 & \alpha_{12} & \cdots & \alpha_{1\mathrm{p}} \\ 0 & 1 & \cdots & \alpha_{2\mathrm{p}} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

is a unit upper triangular matrix, and the parameter estimates for Equation (7) can be recovered by

$$\hat{\Theta} = \boldsymbol{A}^{-1}\hat{\boldsymbol{G}} \tag{14}$$

and the cost function is computed as

$$\mathrm{E}(\hat{\Theta}) = \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} - \sum_{\mathrm{i}=1}^{\mathrm{p}} \frac{(\boldsymbol{Y}^{\mathrm{T}}\psi_{\mathrm{i}})^2}{(\psi_{\mathrm{i}}^{\mathrm{T}}\psi_{\mathrm{i}})} \tag{15}$$

According to Equation (15), the contribution of an orthogonal term ψ_{i} in decreasing the cost function is explicitly expressed as $\delta \mathrm{E}_{\mathrm{k}} = -\frac{(\boldsymbol{Y}^{\mathrm{T}}\psi_{\mathrm{k}})^2}{(\psi_{\mathrm{k}}^{\mathrm{T}}\psi_{\mathrm{k}})}$, therefore an efficient forward regression model selection can be proposed [6]. However, in the orthogonal least-square algorithm the orthogonal terms in $\boldsymbol{\Psi}$ has to be computed recursively based on Equations (11) and (13), coupled with the computation of elements in $\mathbf{A}$ and $\hat{\boldsymbol{G}}$, which is summarised as follows

$$\begin{cases} \psi_{\mathrm{k}}(\mathrm{x}) = \varphi_{\mathrm{k}}(\mathrm{x}) - \sum_{\mathrm{i}=1}^{\mathrm{k}-1} \alpha_{\mathrm{ik}}\psi_{\mathrm{i}}(\mathrm{x}) \\ \alpha_{\mathrm{ik}} = \dfrac{\overline{\varphi_{\mathrm{k}}(\mathrm{x})\psi_{\mathrm{i}}(\mathrm{x})}}{\overline{(\psi_{\mathrm{i}}(\mathrm{x}))^2}} \\ \hat{\mathrm{g}}_{\mathrm{k}} = \dfrac{\overline{\mathrm{y}(\mathrm{x})\psi_{\mathrm{k}}(\mathrm{x})}}{\overline{(\psi_{\mathrm{k}}(\mathrm{x}))^2}} \\ \delta \mathrm{E}_{\mathrm{k}} = \dfrac{\hat{\mathrm{g}}_{\mathrm{k}}^2\overline{\psi_{\mathrm{k}}^2}}{\overline{\mathrm{y}^2}} = \dfrac{\overline{(\mathrm{y}(\mathrm{x})\psi_{\mathrm{k}}(\mathrm{x}))^2}}{\overline{(\psi_{\mathrm{k}}(\mathrm{x}))^2}\,\overline{(\mathrm{y}(\mathrm{x}))^2}} \end{cases} \tag{16}$$

where $\overline{(\bullet)}$ stands for the mean value. And the estimates of model parameters in Equation (4) can be computed as follows:

$$\hat{\theta}_{\mathrm{i}} = \hat{\mathrm{g}}_{\mathrm{i}} - \sum_{\mathrm{k}=\mathrm{i}+1}^{\mathrm{p}} \alpha_{\mathrm{ik}}\hat{\theta}_{\mathrm{k}},\ \hat{\theta}_{\mathrm{p}} = \hat{\mathrm{g}}_{\mathrm{p}} \tag{17}$$

The elegance of this algorithm is that the error reduction by introducing a new term into the model can be explicitly expressed. Meanwhile, once the last term is selected, the model parameter can be recursively computed in a back-forward fashion. However, strictly speaking, this algorithm is not the right one for identifying the model terms and parameters simultaneous. Since the model selection and estimation of parameters are recursively performed in two opposite directions, i.e. model selection is done in forward fashion, while the parameter selection is done in a backward way. Moreover, the procedure of computing the error reduction $\delta \mathrm{E}_{\mathrm{k}}$ is complicated with the computation of elements in $\boldsymbol{\Psi}$, $\mathbf{A}$, and $\hat{\boldsymbol{G}}$.

In this following, a fast algorithm is proposed for the identification of non-linear dynamic model using RBFN. This algorithm can select the hidden nodes and compute the weights of the output layer in the RBFNs simultaneously in one direction simultaneously, i.e. in the forward fashion.

3 The fast algorithm

Without losing generalization, all vectors in $\boldsymbol{\Phi}$ in Equation (4) are normalised so that $\varphi_{\mathrm{i}}^{\mathrm{T}}\varphi_{\mathrm{i}} = 1, \mathrm{i} = 1,...,\mathrm{p}$, and also that $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}=1$. Then the constant term θ_0 in Equation (3) can be removed, and $\boldsymbol{\Phi}$ becomes $\boldsymbol{\Phi} = [\varphi_1, \varphi_2, .., \varphi_{\mathrm{p}}]$, and Θ becomes $\Theta = [\theta_1, \theta_2, .., \theta_p]^T$.

Let $\mathrm{M} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$, $\mathrm{M}_{\mathrm{k}} = \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}, \mathrm{k} \in \{1,2,...,\mathrm{p}\}$ where $\boldsymbol{\Phi}_{\mathrm{k}}$ contains the first k columns (vectors) in $\boldsymbol{\Phi}$, $\hat{\boldsymbol{\Theta}}_{\mathrm{k}} = \mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{Y}$, and $\mathrm{E}_{\mathrm{k}} = 1 - \hat{\Theta}_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{Y}$.

Since

$$\begin{aligned} \mathrm{M}_{\mathrm{k}+1} &= \boldsymbol{\Phi}_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}+1} = \begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}} \\ \varphi_{\mathrm{k}+1}^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{k}} & \varphi_{\mathrm{k}+1} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}} & \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1} \\ \varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}} & \varphi_{\mathrm{k}+1}^{\mathrm{T}}\varphi_{\mathrm{k}+1} \end{bmatrix} = \begin{bmatrix} \mathrm{M}_{\mathrm{k}} & \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1} \\ \varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}\mathrm{k} & 1 \end{bmatrix} \end{aligned} \tag{18}$$

Let

$$\begin{aligned} &\mathrm{M}_{\mathrm{k}+1}\begin{bmatrix} \boldsymbol{F}_{\mathrm{k}} & \boldsymbol{G}\mathrm{k} \\ \boldsymbol{G}_{\mathrm{k}}{}^{\mathrm{T}} & \mathrm{f}_{\mathrm{k}} \end{bmatrix} = \\ &\begin{bmatrix} \mathrm{Mk} & \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1} \\ \varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}} & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{F}_{\mathrm{k}} & \boldsymbol{G}_{\mathrm{k}} \\ \boldsymbol{G}_{\mathrm{k}}{}^{\mathrm{T}} & \mathrm{f}_{\mathrm{k}} \end{bmatrix} \\ &= \boldsymbol{I}_{(\mathrm{k}+1)\times(\mathrm{k}+1)} \end{aligned} \tag{19}$$

where $\boldsymbol{F}_{\mathrm{k}} \in \mathrm{R}^{\mathrm{k}\times\mathrm{k}}$, $\boldsymbol{G}_{\mathrm{k}} \in \mathrm{R}^{\mathrm{k}\times 1}$, $\mathrm{f}_{\mathrm{k}} \in \mathrm{R}^{1\times 1}$ are the matrices to be determined, then we have

$$\begin{cases} \mathrm{M}_{\mathrm{k}}\boldsymbol{F}_{\mathrm{k}} + \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1}\boldsymbol{G}_{\mathrm{k}}{}^{\mathrm{T}} = \boldsymbol{I}_{\mathrm{k}\times\mathrm{k}} \\ \varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{F}_{\mathrm{k}} + \boldsymbol{G}_{\mathrm{k}}{}^{\mathrm{T}} = 0 \\ \varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{G}_{\mathrm{k}} + \mathrm{f}_{\mathrm{k}} = 1 \end{cases} \tag{20}$$

therefore

$$\begin{cases} [\mathrm{M}_{\mathrm{k}} - \boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1}\varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}]\boldsymbol{F}_{\mathrm{k}} = \boldsymbol{I}_{\mathrm{k}\times\mathrm{k}} \\ \boldsymbol{G}_{\mathrm{k}} = -\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1} \\ \mathrm{f}_{\mathrm{k}} = (1 + \varphi_{\mathrm{k}+1}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k}+1}) \end{cases} \tag{21}$$

Lemma 1 (Ljung [17]) Let **A, B, C** and **D** be matrices with appropriate dimensions, so that the product **BCD**, the sum **A+BCD** and the inverses of **A**, **C** exist, then the following is true:

$$[\boldsymbol{A}+\boldsymbol{BCD}]^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{B}[\boldsymbol{DA}^{-1}\boldsymbol{B}+\boldsymbol{C}^{-1}]^{-1}\boldsymbol{DA}^{-1} \quad (22)$$

To compute $\boldsymbol{F}_{\mathrm{k}}=[\mathrm{M}_{\mathrm{k}}-\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}]^{-1}$, let $\mathbf{A}=M_k$, $\boldsymbol{B}=\boldsymbol{D}^{\mathrm{T}}=\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}$, $\mathbf{C}=-\mathbf{I}_{N\times N}$, according to Lemma 1, we have:

$$\begin{aligned}\boldsymbol{F}_{\mathrm{k}}&=\mathrm{M}_{\mathrm{k}}^{-1}-\\&\mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}[\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}-1]^{-1}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{M}_{\mathrm{k}}^{-1} \quad (23)\\&=\mathrm{M}_{\mathrm{k}}^{-1}+\frac{\mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}^{-1}}{1-\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}}\end{aligned}$$

Hence

$$\begin{cases}\mathrm{M}_{\mathrm{k+1}}^{-1}=\begin{bmatrix}\boldsymbol{F}_{\mathrm{k}} & \boldsymbol{G}_{\mathrm{k}}\\ \boldsymbol{G}_{\mathrm{k}}^{\mathrm{T}} & \mathrm{f}_{\mathrm{k}}\end{bmatrix}\\ \boldsymbol{F}_{\mathrm{k}}=\mathrm{M}_{\mathrm{k}}^{-1}+\dfrac{\mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}^{-1}}{1-\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}{}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}}\\ \boldsymbol{G}_{\mathrm{k}}=-\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\\ \mathrm{f}_{\mathrm{k}}=(1+\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}})\\ \text{for } \mathrm{k}=1,2,...,(\mathrm{p}-1) \text{ and } \mathrm{M}_1=1\end{cases} \quad (24)$$

Based on the solutions of $\mathbf{F}_{\mathrm{k}}$, $\mathbf{G}_{\mathrm{k}}$, f_{k}, $\hat{\boldsymbol{\Theta}}_{k+1}$ and E_{k+1} can be reformulated as follows:

$$\begin{cases}\hat{\boldsymbol{\Theta}}_{\mathrm{k+1}}=\begin{pmatrix}\hat{\boldsymbol{\Theta}}_{\mathrm{k+1}}{}'\\ \hat{\theta}_{\mathrm{k+1}}\end{pmatrix}\\ \hat{\boldsymbol{\Theta}}_{\mathrm{k+1}}{}'=\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\mathrm{Y}+\boldsymbol{G}_{\mathrm{k}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{Y}\\ =\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}[\mathbf{I}-\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}]\boldsymbol{Y}\\ \hat{\theta}_{\mathrm{k+1}}=\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{Y}-\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}[\mathbf{I}-\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}]\boldsymbol{Y}\\ \text{for } \mathrm{k}=1,2,...,(\mathrm{p}-1) \text{ and } \hat{\boldsymbol{\Theta}}_1=\varphi_1^{\mathrm{T}}\boldsymbol{Y}\end{cases} \quad (25)$$

$$\begin{cases}\mathrm{M}_{\mathrm{k+1}}^{-1}=\begin{bmatrix}\boldsymbol{F}_{\mathrm{k}} & \boldsymbol{G}_{\mathrm{k}}\\ \boldsymbol{G}_{\mathrm{k}}^{\mathrm{T}} & \mathrm{f}_{\mathrm{k}}\end{bmatrix}\\ \boldsymbol{F}_{\mathrm{k}}=\mathrm{M}_{\mathrm{k}}^{-1}+\dfrac{\mathrm{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}^{-1}}{1-\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\mathrm{M}_{\mathrm{k}}{}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}}\\ \boldsymbol{G}_{\mathrm{k}}=-\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\\ \mathrm{f}_{\mathrm{k}}=(1+\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{F}_{\mathrm{k}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\varphi_{\mathrm{k+1}})\\ \text{for } \mathrm{k}=1,2,...,(\mathrm{p}-1) \text{ and } \mathrm{M}_1=1\end{cases} \quad (26)$$

$$\mathrm{E}_{\mathrm{k+1}}=\frac{1}{\mathrm{N}}(1-\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}\hat{\boldsymbol{\Theta}}_{\mathrm{k}}{}'-\boldsymbol{Y}^{\mathrm{T}}\varphi_{\mathrm{k+1}}\hat{\theta}_{\mathrm{k+1}}) \quad (27)$$

The recursive algorithm to compute $\hat{\boldsymbol{\Theta}}$ and E according to (25), (26) and (27) is not intuitive and simplified formulas are required. Before introduce the simple non-orthogonal algorithm to compute E, the following theorem is introduced.

Theorem 1 Let $\boldsymbol{K}_{\mathrm{k}}=\mathbf{I}_{\mathrm{N\times N}}-\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}$, then we have

$$\begin{cases}\boldsymbol{K}_{\mathrm{k+1}}=\boldsymbol{K}_{\mathrm{k}}-\dfrac{\boldsymbol{K}_{\mathrm{k}}\varphi_{\mathrm{k+1}}\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{k}}^{\mathrm{T}}}{\varphi_{\mathrm{k+1}}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{k}}\varphi_{\mathrm{k+1}}}\\ \boldsymbol{K}_0=\boldsymbol{I}_{\mathrm{N\times N}}\end{cases} \quad (28)$$

(28) can be derived by substituting (26) into $\mathrm{M}_{\mathrm{k+1}}$ and $\boldsymbol{K}_{\mathrm{k+1}}=\mathbf{I}_{\mathrm{N\times N}}-\boldsymbol{\Phi}_{\mathrm{k+1}}\mathrm{M}_{\mathrm{k+1}}^{-1}\boldsymbol{\Phi}_{\mathrm{k+1}}^{\mathrm{T}}$.

Now by introducing $\boldsymbol{K}_{\mathrm{k}}=\mathbf{I}_{\mathrm{N\times N}}-\boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{M}_{\mathrm{k}}^{-1}\boldsymbol{\Phi}_{\mathrm{k}}^{\mathrm{T}}$ and submit (25) and (26) into (27), we have

$$\begin{cases}\mathrm{E}_{\mathrm{k}}-\mathrm{E}_{\mathrm{k-1}}=-\dfrac{(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{k-1}}\varphi_{\mathrm{k}})^2}{\varphi_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{k-1}}\varphi_{\mathrm{k}}}\\ \boldsymbol{K}_{\mathrm{k}}=\boldsymbol{K}_{\mathrm{k-1}}-\dfrac{\boldsymbol{K}_{\mathrm{k-1}}\varphi_{\mathrm{k}}\varphi_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{k-1}}^{\mathrm{T}}}{\varphi_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{k-1}}\varphi_{\mathrm{k}}}\\ \boldsymbol{K}_0=\boldsymbol{I}_{\mathrm{N\times N}},\ \mathrm{E}_0=1\end{cases} \quad (29)$$

Furthermore, let $\varphi_{\mathrm{i}}^{(\mathrm{k})}=\mathbf{K}_{\mathrm{k}}\varphi_{\mathrm{i}}$, $\delta\varphi_{\mathrm{i}}^{(\mathrm{k})}=\varphi_{\mathrm{i}}-\varphi_{\mathrm{i}}^{(\mathrm{k})}$, $\mathrm{i}=1,\cdots\mathrm{p}$, since $\mathbf{K}_{\mathrm{k}}=\mathbf{K}_{\mathrm{k}}^{\mathrm{T}}$ and $\mathbf{K}_{\mathrm{k}}\mathbf{K}_{\mathrm{k}}=\mathbf{K}_{\mathrm{k}}$ then (25) and (29) may be computed in an integrated framework:

$$\begin{cases}\delta\mathrm{E}_{\mathrm{k}}=\mathrm{E}_{\mathrm{k}}-\mathrm{E}_{\mathrm{k-1}}=-\dfrac{(\boldsymbol{Y}^{\mathrm{T}}\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^2}{(\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^{\mathrm{T}}\varphi_{\mathrm{k}}^{(\mathrm{k-1})}},\\ \mathrm{E}_0=1\\ \hat{\boldsymbol{\Theta}}_{\mathrm{k}}=\begin{pmatrix}\hat{\boldsymbol{\Theta}}_{\mathrm{k}}'\\ \hat{\theta}_{\mathrm{k}}\end{pmatrix}\\ \hat{\theta}_{\mathrm{k}}=\dfrac{1+2(\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^{\mathrm{T}}\varphi_{\mathrm{k}}^{(\mathrm{k-1})}}{1-(\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^{\mathrm{T}}\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})}}\varphi_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{Y}\\ \hat{\boldsymbol{\Theta}}_{\mathrm{k}}'=\dfrac{(\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^{\mathrm{T}}(\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})}+2\varphi_{\mathrm{k}}^{(\mathrm{k-1})})}{(\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^{\mathrm{T}}\delta\varphi_{\mathrm{k}}^{(\mathrm{k-1})}-1}\hat{\boldsymbol{\Theta}}_{\mathrm{k-1}}\\ \mathrm{k}=1,\cdots,\mathrm{p},\ \ \hat{\boldsymbol{\Theta}}_0=0\\ \varphi_{\mathrm{i}}^{(\mathrm{k})}=\varphi_{\mathrm{i}}^{(\mathrm{k-1})}-\dfrac{(\varphi_{\mathrm{i}})^{\mathrm{T}}\varphi_{\mathrm{k}}^{(\mathrm{k-1})}}{(\varphi_{\mathrm{k}}^{(\mathrm{k-1})})^{\mathrm{T}}\varphi_{\mathrm{k}}^{(\mathrm{k-1})}}\varphi_{\mathrm{k}}^{(\mathrm{k-1})}\\ \delta\varphi_{\mathrm{i}}^{(\mathrm{k})}=\varphi_{\mathrm{i}}-\varphi_{\mathrm{i}}^{(\mathrm{k})},\mathrm{i}=1,\cdots\mathrm{p},\delta\varphi_{\mathrm{i}}^{(0)}=0\\ \mathrm{i},\mathrm{k}=1,\cdots,\mathrm{p},\varphi_{\mathrm{i}}^{(0)}=\varphi_{\mathrm{i}},\ \mathrm{i}=1,2,...,\mathrm{p}\end{cases} \quad (30)$$

Remark 1 (30) give a fast algorithm to compute the lost function as well as the estimates of model parameters. According to (30), the error reduction as a new term is added into the regression model is computed as

$$\delta E_k = \frac{(Y^T \varphi_k^{(k-1)})^2}{(\varphi_k^{(k-1)})^T \varphi_k^{(k-1)}}$$ where $\varphi_i^{(k-1)}, i = 1,..., p, k = 1,..., (p-1)$ is the only intermediate matrix to be recursively updated. Therefore it is simpler than the orthogonal least square method. Moreover, according to (30) the estimates of model parameters are simultaneously computed as a new term is added into the model.

Remark 2 There are various other recursive least square method, such as the fast recursive least square method (Ljung and Soderstrom, 1983). However, they are not applicable to regression model selection.

4 Simulation example

The RBFN approach for identification of a non-linear dynamic model is to select significant *p* hidden nodes for model (3) from a saturated model set of *q* terms (*q*>>*p*). When the orthogonal algorithm is used to select model structure, error reduction ration is introduced to select terms, which is defined as

$$ERR_p = [err_1 \cdots err_p]^T = \frac{\hat{G}^T \Psi^T \Psi \hat{G}}{Y^T Y} \qquad (31)$$

To find *p* significant model terms from a saturated model term set, a step-wise procedure is applied. At each step the model term with maximal err_i value from all of the remaining terms in the saturated model term set is selected. The selection procedure is terminated as the p^{th} step when

$$1 - \sum_{i=1}^{p} err_i \qquad (32)$$

reaches a desired tolerance or the reduction rate of cost function err_i by introducing a new term is below certain value. The term selection procedure can be regarded as a series of steps to reduce the model residual or residual to output ratio.

Under the new developed fast algorithm for computing the cost function, ERR_p is reformulated as follows

$$ERR_p = [err_1 \cdots err_p]^T = \text{diag}\left\{ \frac{\left\langle Y, \varphi_k^{(k-1)} \right\rangle^2}{\left\| \varphi_k^{(k-1)} \right\|^2} \right\} \qquad (33)$$

$i = 1,..., p$

where $\langle \bullet \rangle$ is inner product of two vectors, $\| \bullet \|$ denotes the Euclidean norm. And in each step, the model term, which maximises $\max_j \delta E_k = \frac{(Y^T \varphi_j^{(k-1)})^2}{(\varphi_j^{(k-1)})^T \varphi_j^{(k-1)}}$ from all the remaining terms in the saturated model term set, is selected. The estimates of weights for the RBFN output layer is automatically computed simultaneously according to (30).

Example. Consider the following non-linear dynamic system from Haber and Unbehauen [13].

$$\begin{aligned} y(t) &= -0.6377y(t-1) + 0.07298y(t-2) \\ &+ 0.03597u(t-1) + 0.06622u(t-2) \\ &+ 0.06568u(t-1)y(t-1) + 0.02375u^2(t-1) \\ &+ 0.05939 + e(t) \end{aligned} \qquad (34)$$

where $y(t)$, $y(t-1)$ and $y(t-2)$ are system outputs at time t, t-1, t-2 respectively; $u(t-1)$, $u(t-2)$ are system inputs in time t-1, t-2; $e(t)$ is white noise of range of [0, 0.01].

By simulating (34) with $u(t)$ uniformly generated within the range [-0.5, 0.5], a data set of 500 samples is acquired for model selection. The following set of variables is selected as the RBFN inputs $u(t-1), u(t-2), y(t-1), y(t-1)$. The width for the Gaussian basis functions is selected to be $\sigma = 0.45$.

All 500 samples are used as centres in the RBFN, and both OLS and the proposed algorithm are then used to select the best RBF hidden nodes. Interestingly, the selection results are all the same by both algorithms. However the computation complexity is quite different. Matlab simulation shows that OLS takes 2.05×10^8 flops to select the best 20 hidden nodes, while the fast algorithm takes 3.93×10^7 flops to get the same result. The experimental tests show that the proposed algorithm outperforms OLS method. The selection results (the first 15 nodes) are listed in Table 1.

Nodes Index	δE_k	E_k
208	0.5744	0.4256
153	0.0764	0.3492
328	0.0221	0.3271
158	0.1049	0.2222
79	0.0225	0.1997
372	0.0153	0.1844
353	0.0109	0.1735
401	0.0165	0.1570
330	0.0345	0.1225
447	0.0265	0.0959
87	0.0386	0.0573
422	0.0102	0.0470
459	0.0054	0.0416
289	0.0044	0.0372
90	0.0022	0.0350

Table 1 The error reduction as the RBFN hidden nodes selection proceeds

According to Table 1, when the 459th candidate node is selected, the decreased cost function is less than 0.01, therefore only the first 12 nodes are selected into the RBFN. Figure 1 illustrates the model prediction of RBFN in contrast the original system output, as well as the modelling error.

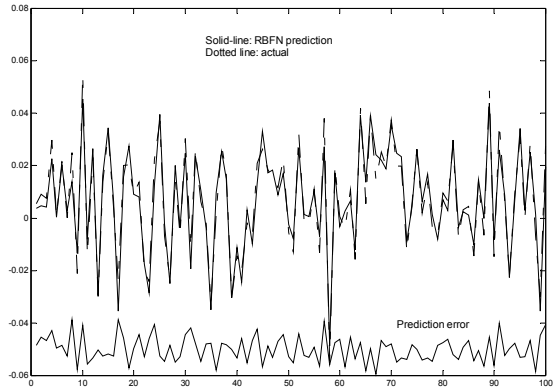

Fig. 1 RBFN prediction
Solid – Prediction, dotted – actual, bottom- error

5 Conclusion

In this paper, a new fast estimation algorithm is derived for identification of non-linear dynamic systems using radial basis function networks. The new algorithm is able to both select the hidden nodes and compute the weights in the radial basis function neural networks (RBFN) simultaneously in a stepwise forward fashion. The practical simulation shows that it outperforms the OLS in terms of computation complexity.

References

[1] K.M. Adeney, M.J. Korenberg. Iterative fast orthogonal search algorithm for MDL-based training of generalized single-layer networks, *Neural Networks,* **13**, pp. 787-799, (2000).

[2] S.A Billings, X. Hong. Dual-orthogonal radial basis function networks for nonlinear time series prediction. *Neural Networks,* **11**, pp. 479–493, (1998).

[3] A. Bjorck. Numerical methods for least squares problems. Frontiers in Applied Mathematics, Soc. Ind. Applied Math., Philadelphia, (1995).

[4] D. S. Broomhead, D. Lowe. Multivariable functional interpolation and adaptive networks, *Complex Systems* **2**, pp. 321 –355, (1988).

[5] S. Chen, C. F. N. Cowan, P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks,* **2**, pp. 302 –309, (1991).

[6] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to nonlinear system identification. *International Journal of Control*, **50**, 1873-1896, (1989).

[7] E. S. Chng, S. Chen, & B. Mulgrew. Gradient radial basis function networks for nonlinear and nonstationary time series prediction. *IEEE Transactions on Neural Networks*, **7** (1), pp.190–194, (1996).

[8] Christensen, R.. Log-linear models and logistic regression. 2nd edition, Springer-Verlag, New York, (1990).

[9] C. Drioli, D. Rocchesso. Orthogonal least squares algorithm for the approximation of a map and its derivatives with a RBF network. *Signal Processing*, **83**, pp. 283 – 296, (2003).

[10] N. R. Draper, and H. Smith. Applied Regression Analysis. New York: Wiley, (1981).

[11] F. Girosi, and T. Poggio. Neural networks and the best approximation property. *Biol. Cybernetics*, **63**, pp. 169-176, (1990).

[12] G. H. Golub, and C. F. van Loan. Matrix Computations. Baltimore, MD: The Johns Hopkins University Press, 2nd Ed, (1989).

[13] R. Haber, and H. Unbehauen. Structure identification of nonlinear dynamic systems – a survey on input/output approaches. *Automatica,* **26**, pp. 651-67, (1990).

[14] R. J. Howlett, L. C. Jain (Eds.). Radial Basis Function Networks: Design and Applications, Physica-Verlag, Wurzbury, (2000).

[15] C. L. Lawson, and R. J. Hanson. Solving Least Squares Problem. Englewood Cliffs, NJ: Prentice-Hall, (1974).

[16] L. Ljung, and T. Soderstrom. Theory and Practice of Recursive Identification. Cambridge, Massachusetts, MIT Press, (1983).

[17] L. Ljung,. System identification: theory for the user. Prentice Hall, Englewood Cliffs, N.J., (1987).

[18] S. Niu, L. Ljung, and A. Bjorck. Decomposition methods for solving least-squares parameter estimation. *IEEE Transactions on Signal Processing*, **44**, pp. 2847-2852, (1996).

[19] J. Park, I.W. Sandberg. Approximation and radial-basis-function networks, *Neural Comput*. **5**, pp. 305–316, (1993).

[20] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. In J.C. Mason and M.G. Cox (Eds.), Algorithms for approximation, pp 143-167. Oxford: Clarendon Press, (1987).

[21] F. Schwenker, H. A. Kestler and G. Palm. Three learning phases for radial-basis-function networks, *Neural Networks*, **14**: 439-458, (2001)

[22] V. Vapnik, S. E. Golowich, A. Smola, Support vector method for function approximation, regression estimation, and signal processing, in: M. Mozer, M. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems 9,The MIT Press,Cambridge,MA, pp.281 –287, (1997).

[23] G. L. Zheng, & S. A. Billings. Radial basis function network configuration using mutual information and the orthogonal least squares algorithm. *Neural Networks*, **9**, pp. 1619–1637, (1996).