Structure Selective Updating for Nonlinear Models and Radial Basis Function Neural Networks

W. Luo * and S. A. Billings +

* Department of Chemical and Process Engineering
University of Newcastle upon Tyne
Merz Court, Newcastle upon Tyne NE1 7RU, U.K.
Tel: +44 191 222 6000 ext 5207

+ Department of Automatic Control and Systems Engineering
University of Sheffield
Mappin Street, Sheffield S1 4DU, U.K.
Tel: +44 114 2768555 ext 5232

June 1995
Research Report No. 580

Abstract

Selective model structure and parameter updating algorithms are introduced for both the on-line estimation of NARMAX models and training of radial basis function neural networks. Techniques for on-line model modification which depend on the vector-shift properties of regression variables which are widely used for linear models cannot be applied when the model is nonlinear. In the present paper two new methods for on-line model modification are developed. The first is based on selectively updating the nonlinear model structure and this leads to a reduction in computational cost. The second is based on selecting specified regression variables to efficiently obtain nonlinear models for applications such as one-step-ahead controllor design. Experiments on both simulated and real data sets are used to demonstrate the performance of the new algorithms.

Keywords

adaptive structure detection, on-line processing, system identificatiion, neural networks

1. Introduction

Lattice algorithms and related techniques that are widely applied for the on-line modification of model structure for linear models cannot generally be applied in the nonlinear cases because most nonlinear models do not satisfy the simple regressor-shift properties on which lattice type techniques are based. It is therefore necessary to develop new efficient methods for nonlinear systems. Two adaptive model selection methods for nonlinear systems have been developed in previous studies using numerically stable orthogonal routines with forward selection including both exponential and sliding data windows [20,21]. Both these algorithms can minimize the loss function at each step by selecting significant variables while maintaining the orthogonality of the vector space as each new observation is processed. These algorithms can be used to determine the model structure and parameters on-line and have demonstrated a satisfactory performance in experiments.

However both these algorithms modify the model structure at every sample instant in a point-updating manner. But many systems have a stable structure over long time periods and therefore the operation of adjusting the model structure at every computational interval becomes unnecessary. The main focus of the present study is therefore to introduce new algorithms which only selectively update the model structure when the underlying system shows significant change according to some specified criterion, so that if model structure is stable only the parameters are updated based on the model structure determined at the previous selection step. It is therefore important to develop methods for selectively updating the model structure. The use of these methods should also reduce the computational cost. Sometimes it may be difficult to specify a tolerance which can be used to determine whether the system structure has changed. When the structure of the system is stable over a long period, the tolerance may be considered as the mean of the residuals in a large data window and this leads to an adaptive search scheme for updating the model structure.

In some applications the selected regressors must be constrained to include some specified variables. One such application is the one-step-ahead control of nonlinear systems. This requires that the control term $u(t)$ must be maintained in the model at every time point. The models fitted using such methods reflect both the system dynamics and simplify the design

of controllors. The estimator is therefore required to select the optimal regressors with some restrictions on candidate regression variables and it is shown that conditional-selection can be realized using an extension of the adaptive orthogonal algorithms based on either point- or selective-updating.

Radial basis function neural networks (RBFN) have been extensively applied in system modelling and signal processing. In radial basis functions network every output of the network is a linear combination of the outputs of neurones from a single hidden layer. Since the NARMAX model may involve different functions, called the extended model set [2], RBFN can be considered as part of an extended set of the polynomial NARMAX model. The on-line algorithms derived for the polynomial NARMAX model can therefore be extended to the training of RBFN. The present study will focus on applying the selective structure updating algorithms presented in this paper to radial basis function networks.

The paper is organized as follows. Section 2 describes the NARMAX model. The Square-Root-Free version of Givens rotation with Forward selection and EXponential windowing algorithm (GFEX) will be briefly derived in Section 3. Section 4 introduces methods for model selective structure updating to save computational cost for systems with stable model structures. In Section 5 the conditional selection of regressors is discussed for one-step-ahead control of nonlinear systems represented by the NARMAX model. In Section 6 the results from the previous sections are extended to RBFN. The final two sections contain experimental results and conclusions. For clarity only the exponential windowing algorithm will be considered here, but the principles described can easily be carried over to the sliding window algorithm [21].

2. The NARMAX Model and the Extended Model Set

A single-input single-output NARMAX [18] can be defined as

$$y(t) = F[y(t-1),\ldots,y(t-n_y),u(t-d),\ldots,u(t-d-n_u+1),\varepsilon(t-1),\ldots,\varepsilon(t-n_\varepsilon)]+\varepsilon(t). \tag{2-1}$$

where $y(t)$, $u(t)$ and $\varepsilon(t)$ represent the output, input, and residual respectively at time interval t, n_y, n_u and n_ε are the corresponding orders, $F[.]$ is some nonlinear function and d is the

minimum time delay of the input. In practice the noise $e(t)$ cannot usually be measured and therefore all variables related to the noise have been replaced in (2-1) by the residual which is defined as

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{2-2}$$

where $\hat{}$ denotes estimate of. The polynomial NARMAX model often involves power and cross-product terms of the inputs, outputs and residuals. For example, a polynomial NARMAX model with a nonlinearity $n_l = 2$, $n_y = 2$, $n_u = n_\varepsilon = 1$ and $d = 1$ can be expressed as

$$\begin{aligned} y(t) = {} & \theta_0(t) + y(t-1)\theta_1(t) + y(t-2)\theta_2(t) + u(t-1)\theta_3(t) + y^2(t-1)\theta_4(t) + y(t-1)y(t-2)\theta_5(t) \\ & + y^2(t-2)\theta_6(t) + y(t-1)u(u-1)\theta_7(t) + y(t-2)u(t-1)\theta_8(t) + u^2(t-1)\theta_9(t) \\ & + \varepsilon(t-1)\theta_{10}(t) + \varepsilon^2(t-1)\theta_{11}(t) + \varepsilon(t-1)y(t-1)\theta_{12}(t) \\ & + \varepsilon(t-1)y(t-2)\theta_{13}(t) + \varepsilon(t-1)u(t-1)\theta_{14}(t) + \varepsilon(t). \end{aligned}$$

and can be represented concisely as a pseudo-linear regression model

$$y(t) = \sum_{i=0}^{m} \phi_i(t)\theta_i(t) + \varepsilon(t) \tag{2-3}$$

where $\phi_i(t)$ expresses the i'th regression variable (regressor) which are monomials of lagged $u(t)$, $y(t)$ and/or $\varepsilon(t)$, $\phi_0(t) = 1$, $\theta_i(t)$ is the unknown parameter corresponding to $\phi_i(t)$ and m is the number of the regressors (in the above example m =15). A variable structure NARMAX model can be obtained by replacing m with a variant $m(t)$ in Eqn (2-3).

The choice of functions $F[.]$ in (2-1) are various and the generalised polynomial NARMAX models may involve different functions, such as absolute value, exponential, logarithmic, $sgn(.)$ etc. [2]. Since every output of the radial basis function network can be expressed as a regressor which may involve different kinds of activation functions (radial basis functions), the RBFN can be considered as an extended set of the polynomial NARMAX model and methods developed for the polynomial NARMAX model can therefore be applied to RBFN's. The extension to the RBFN will be considered in detail in Section 7.

3. Square Root Free - GFEX algorithm

The on-line model selection and parameter estimation algorithm derived previously by Luo et al [20] involves square-root Givens rotation routines that are considered to involve

higher computational costs than the square-root-free routines [13,14]. Unfortunately the square-root-free Givens rotation routines given by Gentleman [13,14] are not so easy to apply for structure detection, because these routines change the scaling factors and hence affect the structure detection. A square-root-free version of the GFEX algorithm will therefore be derived by modifying the original square-root-free Givens routines of Gentleman. Full details are given in Appendix I. The new routines can be used in recursive orthogonal transformation, retriangularization and on-line structure detection.

The regression equations for data at the time points *1*, *2*, ..., *t* can be given as

$$\mathbf{y}(t) = \Phi(t)\theta(t) + \varepsilon(t) \tag{3-1}$$

where $\Phi(t)$, $\theta(t)$, $\mathbf{y}(t)$ and $\varepsilon(t)$ have corresponding dimensions. The least squares solution for $\theta(t)$ can be obtained by solving the normal equations

$$\Phi^T(t)\Phi(t)\theta(t) = \Phi^T(t)\mathbf{y}(t) \quad . \tag{3-2}$$

Unfortunately, the matrix $\Phi^T\Phi$ is often ill-conditioned and can be greatly affected by roundoff errors. The problem of ill-conditioning can be particularly serious in overparameterized polynomial nonlinear models which involve a large number of candidate regressors where the nonlinearity is of a high degree. One solution to this problem is to use orthogonal decomposition

$$\Phi(t) = \mathbf{Q}(t)\mathbf{R}(t) \tag{3-3}$$

where $\mathbf{Q}(t)$ is an orthonormal matrix and $\mathbf{R}(t)$ is an upper triangular matrix. Substituting (3-3) into (3-2) yields

$$\Phi^T(t)\Phi(t)\theta(t) = \mathbf{R}^T(t)\mathbf{Q}^T(t)\mathbf{Q}(t)\mathbf{R}(t)\theta(t) = \mathbf{R}^T(t)\mathbf{R}(t)\theta(t) = \mathbf{R}^T(t)\mathbf{Q}^T(t)\mathbf{y}(t)$$

and therefore $\mathbf{R}(t)\theta(t) = \mathbf{Q}^T(t)\mathbf{y}(t)$. If $\mathbf{R}(t)$ is non-singular the estimate of $\theta(t)$ can easily be obtained by solving the above triangular system.

Such an orthogonal decomposition process may be performed in a recursive manner. Assume that the regression equations corresponding to m time points and m regressors have been decomposed to form

$$\mathbf{R}(m)\hat{\theta}(m) = \mathbf{v}_m(m) - \mathbf{Q}^T(m)\varepsilon_m(m)$$

where $\mathbf{v}_m(m)=\mathbf{Q}^T(m)\mathbf{y}(m)$ and $\varepsilon_m(m)$ is the residual vector. When data at time point $m+1$ needs to be added to improve the previous estimates, the procedure may be executed using an augmented form of the above equations given by

$$\begin{bmatrix}\mathbf{R}(m)\\ \phi(m+1)\end{bmatrix}\theta(m)=\begin{bmatrix}\mathbf{v}_m(m)\\ y(m+1)\end{bmatrix}-\begin{bmatrix}\mathbf{Q}^T(m)\\ \mathbf{O}_Q\end{bmatrix}\varepsilon_m(m)-\begin{bmatrix}\mathbf{O}_{\varepsilon_m}\\ \underline{\varepsilon}(m+1)\end{bmatrix},$$

where $\underline{\varepsilon}(m+1)$ is the a priori prediction error defined by $\underline{\varepsilon}(m+1)=y(m+1)-\phi(m+1)\hat{\theta}(m)$ and $\mathbf{O}_Q$ is a zero row vector. Further orthogonal decomposition yields

$$\begin{bmatrix}\mathbf{R}(m+1)\\ \mathbf{O}_1\end{bmatrix}\theta(m+1)=\begin{bmatrix}\mathbf{v}_m(m+1)\\ v(m+1)\end{bmatrix}-\mathbf{Q}^T(m+1)\begin{bmatrix}\varepsilon_m(m+1)\\ \varepsilon(m+1)\end{bmatrix},$$

where $\mathbf{O}_1$ is a zero row vector. Since $v(m+1)$ is not used to determine $\hat{\theta}(m+1)$ using the backsubtitution operation, this may be replaced by a zero element, $\mathbf{O}_2$. Later data are located sequentially in the positions of $\mathbf{O}_1$ and $v(m+1)$ and re-transformed using orthogonal routines so that the estimate of $\hat{\theta}(t)$, $(t=m+2,m+3,.....)$, can be obtained sequentially. The implementation of the decomposition is recursive and the transformed form at time point t should be

$$\begin{bmatrix}\mathbf{R}(t)\\ \mathbf{O}_1\end{bmatrix}\hat{\theta}(t)=\begin{bmatrix}\mathbf{v}_m(t)\\ \mathbf{O}_2\end{bmatrix}-\mathbf{Q}^T(t)\begin{bmatrix}\varepsilon_m(t)\\ \varepsilon(t)\end{bmatrix} \tag{3-4}$$

The corresponding residual sum of squares (RSS) can be computed from

$$RSS(t)=\|\varepsilon(t)\|^2=\|\mathbf{y}(t)\|^2-\|\mathbf{v}_m(t)\|^2=\mathbf{y}^T(t)\mathbf{y}(t)\mathbf{v}_m^T(t)\mathbf{v}_m(t) \tag{3-5}$$

Combining $\mathbf{R}(t)$ and $\mathbf{v}_m(t)$ to form an augmented matrix

$$\begin{bmatrix}\mathbf{R}(t) & \mathbf{v}_m(t)\\ \mathbf{O}_1 & \mathbf{O}\end{bmatrix} \tag{3-6}$$

Using the square-root-free algorithm (A-1), the row vectors of $\mathbf{R}(t)$ mentioned above should be scaled by the corresponding diagonal elements, namely,

$$\mathbf{R}(t)=\mathbf{D}^{1/2}\mathbf{R}^*(t) \tag{3-7}$$

and correspondingly

$$\mathbf{v}_m(t)=\mathbf{D}^{1/2}\mathbf{v}_m^*(t) \tag{3-8}$$

where $\mathbf{D}(t)$ is diagonal and $\mathbf{R}^*(t)$ is an $m\times m$ upper triangular matrix with unit diagonal elements. Since the procedures for selecting regressors, which will be described later, involve the exchange of column vectors and retriangularization of the upper triangular matrix, the matrix (3-6) is modified to the following form

$$\begin{bmatrix} \mathbf{D}^* \mathbf{R}^* \mathbf{v}_m^* \\ \mathbf{O}_3 \mathbf{O}_1 \mathbf{O}_2 \end{bmatrix} = \begin{bmatrix} d_1 & r_{11}^* & r_{12}^* & \cdots & r_{1m}^* & v_1^* \\ d_2 & 0 & r_{22}^* & \cdots & r_{2m}^* & v_2^* \\ \cdot & \cdot & 0 & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ d_m & 0 & 0 & \cdots & r_{mm}^* & v_m^* \\ \mathbf{O}_3 & & \mathbf{O}_1 & & \mathbf{O}_2 & \end{bmatrix} \qquad (3-9)$$

where $\mathbf{D}^* = [d_1,...,d_m]^T$ consists of m diagonal elements of $D(t)$ and $\mathbf{O}_3$ is the zero element. In the exponential windowing methods a forgetting factor λ should be used to multiply $d_1,...,d_m$ when a row of new data are added into the bottom row of the matrix (3-9). To clearly express the operations the notation "t" has been ignored in (3-9) and such a simplification will be used throughout the remainder of this section.

In the initial stage of processing many candidate regressors may emerge but most of these are often insignificant or linearly dependent. This means $\mathbf{R}$ may be singular and therefore there is no unique solution to the normal equations. But if adequate techniques are applied to $\mathbf{R}$, the linear independent regressors can be decomposed and the significant regressors which should be in the model can be determined based on a pre-set tolerance of *RSS*. Dividing (3-5) by $\mathbf{y}^T\mathbf{y}$ gives the normalized *RSS* (*NRSS*), given by

$$NRSS = 1 - \frac{\mathbf{v}_m^T \mathbf{v}_m}{\mathbf{y}^T \mathbf{y}} = 1 - \sum_{i=1}^{m} ERR_i \qquad (3\text{-}10)$$

where $ERR_i = d_i (v_i^*)^2 / \mathbf{y}^T\mathbf{y}$. This is defined as the error reduction ratio (*ERR*) of the orthogonal vector $\mathbf{q}_i$ [19,2]. In the exponential windowing methods $\mathbf{D}^*$ is exponentially weighted at every time interval and therefore $\mathbf{v}_m$ ($=\mathbf{D}^{1/2}\mathbf{v}_m^*$) is a weighted quantity. Correspondingly $\mathbf{y}^T\mathbf{y}$ should be taken as the weighted value in the exponential window. The values of $d_i (v_i^*)^2$ or ERR_i can conveniently be utilized to select significant regressors from all the candidate regression variables by using a forward search procedure so that every selected variable minimizes *NRSS(t)* at every selection step. The number of selected regressors at time t, named

m_s, will normally be less than the number of candidate variables, m. Therefore the selection will be continued for $m_s(t)$ steps until

$$NRSS_{m_s(t)} = 1 - \sum_{i=1}^{m_s(t)} ERR_i(t) \le \xi_s \tag{3-11}$$

where ξ_s is a pre-set tolerance. Generally, the selected variables at every computational interval are different and the number of these regressors, $m_s(t)$, is time-varying, .

Assume that the j'th optimal regressor is being selected. Combining the bottom m-j+1 elements of the column vectors of $\mathbf{R}^*$, $\mathbf{r}_p^*$, some elements of $\mathbf{v}_m^*$, v_p^*, and some elements of $\mathbf{D}^*$, d_p^*, $(p = j,...,m)$, forms the m-j+1 computational matrix given by

$$\begin{bmatrix} d_j & r_{jp}^* & v_j^* \\ . & . & . \\ . & . & . \\ . & . & . \\ d_m & r_{mp}^* & v_m^* \end{bmatrix} \tag{3-12}$$

where $p = j,...,m$. Using the orthogonal rotation in Eqn (A-1) to eliminate $r_{j+1,p}^*$, $r_{j+2,p}^*$, ..., $r_{m,p}^*$ gives the result $v_{j(p)}^*$ located at the position of v_j^*. The practical operation of the above procedure does not involve the storage of the m-j+1 computational matrices above and a simple subroutine to implement this is given in Appendix II. According to (3-10) the regressor with the maximum $d_p(v_{j(p)}^*)^2$ is selected as the j'th optimal regressor so that the *NRSS* can be minimized at the j'th selection step. Then the column vector corresponding to the selected optimal regressor is exchanged to the position of the j'th column vector of $\mathbf{R}^*$. To maintain the orthogonality of the vector space, a retriangularization procedure must be performed using the formula given in (A-1). The selection stops if the critical value is satisfied for Eqn (3-11), where $m_s(t) = j$ and the resulting model is the sub-model available at time point t.

The operation may be summarized briefly into follows: -

(i) Add a row of new data into the bottom row of the augmented matrix (3-9), multiply d_1, ..., d_m by the forgetting factor λ ;

(ii) Update $\mathbf{y}^T\mathbf{y}$ by multiplying the previous vale of $\mathbf{y}^T\mathbf{y}$ by λ and then adding $y^2(t)$

(iii) Perform an orthogonal transform using Givens rotation to produce a new matrix that is of the form of (3-9);

(iv) Using the subroutine in Appendix II, compute $d_i(v_i^*)^2$ and select the optimal regressor with the maximum $d_i(v_i^*)^2$ by selecting the columns of $\mathbf{R}^*$;

(v) Exchange the column vectors of $\mathbf{R}^*$ and then retriangularize $\mathbf{R}^*$ and $\mathbf{v}_m^*$ in (3-9);

(vi) Continue the selection until (3-11) is satisfied, e.g. m_s regression variables are selected;

(vii) Use back-substitution to obtain estimates of the parameters θ_i, $i=1,\ldots,m_s$ from $\mathbf{R}_{m_s}^*$ (this is the top left triangular portion of the final $\mathbf{R}^*$), $\mathbf{v}_{m_s}^*$ (which consists of the first m_s elements of $\mathbf{v}_m^*$) and $\mathbf{d}_{m_s}$ (which consists of the first m_s elements of $\mathbf{d}_m$).

(viii) Compute the residual at the time instant t

$$\varepsilon(t) = y(t) - \sum_{i=0}^{m_s} \phi_i(t)\theta_i(t)$$

The prediction error $\varepsilon(t)$ will become part of the next input signal if the model specification includes noise terms. Notice that if the columns of $\mathbf{R}^*$ have been permuted new data added in the next computational interval must be input to the corresponding columns.

The candidate regressors can involve many terms and these should be sufficient to describe the system dynamics in a wide range of operation. At the beginning of the computation the algorithm also permits the addition of "empty" regressors by assigning very small numbers to these. Variables which have not been involved as candidate regressors can then replace these "empty" variables later in the computational process. Removing some useless candidate variables can also be easily realized by substituting all the elements of the associated columns of $\mathbf{R}^*$ with very small numbers. An alternative method of adjusting candidate variables is to extend or contract on-line the dimension of the augmented matrix (3-9). All the new elements are initialized to very small numbers at time t and then the data associated with these new variables are added successively to the computations.

4. Selective Model Structure Updating

Under a basic assumption that the noise signal is stationary, some methods for model structure selective-updating are considered in this section.

When the system structure changes suddenly, the prediction error increases rapidly. This is because the effect of past input-output data decays based on the exponential factor λ and estimators need time to track the change in the system. If slowly time-varying systems have a change in the structure, the estimates based on the previously detected structure cannot always match the system dynamics very well and an appreciable deviation may appear. If the structure does not change, the time-varying parameters can induce a deviation, but such errors often reduce quickly when the correct model structure is applied. Based on these observations a measurement for selectively updating the structure can be defined as the mean of the sum of the squares of the residuals in an interval

$$\overline{\varepsilon^2_{M_s(t)}}(t) = \frac{1}{M_s} \sum_{i=0}^{M_s-1} \varepsilon^2(t-i), \tag{4-1}$$

where M_s is the length of the specified interval. In every computational interval, new data are first transformed into the augmented matrix (3-9). The parameter estimation is then computed based on the structure produced at time t-1. After considering the present error, if

$$\overline{\varepsilon^2_{M_s(t)}}(t) \geq \xi_{M_s}, \tag{4-2}$$

where ξ_{M_s} is a pre-set critical value, the system structure will be updated at the present or next computational interval.

The choice of M_s should minimize the effect of large additive noise signals and cancel out the periodicity of the prediction error if the signal to be processed has periodicity. But a large value of M_s reduces the capacity for detecting a change in the structure. Ideally M_s should be smaller than the asymptotic memory length N_a $(=\frac{1}{1-\lambda})$ [12].

Although both ξ_s in (3-11) and ξ_{M_s} in (4-2) are based on the sum of squares of the residuals, the role of each is different. ξ_s controls the reasonableness of sub-models subject to the least squares principle and uses data based on the dimension of the exponential window, namely the asymptotic memory length N_a. But ξ_{M_s} determines the update of the structure based on a time average over a measurement length M_s. Therefore ξ_{M_s} is similar to a bound on the errors. Since the GFEX algorithm always uses stable Givens rotations to completely transform the new measurement data into the augmented matrix (3-9), the complete information about the system structure and parameters can be represented concisely in this

matrix. Thus at a particular time instant both the point-updating and selective-updating method should result in the same model, but the latter saves computational cost.

Ideally ξ_{M_s} should be selected close to the mean of the sum of the squares of the noise given as

$$\overline{e^2_{M_s(t)}}(t) = \frac{1}{M_s}\sum_{i=0}^{M_s-1} e^2(t-i), \tag{4-3}$$

Since $\overline{e^2_{M_s(t)}}(t)$ is unknown, ξ_{M_s} will be considered as a criterion of the square of the residuals. Usually, it is difficult to design the optimal value of ξ_{M_s}, but an approximate value can be found in an adaptive search scheme.

Define

$$\overline{\varepsilon^2_{M_a(t)}}(t) = \frac{1}{M_a}\sum_{i=0}^{M_a-1} \varepsilon^2(t-i) \tag{4-4}$$

where M_a is a large number compared with the asymptotic memory length N_a so that $\overline{\varepsilon^2_{M_a(t)}}(t)$ is the time average over a large period and $e(t)$ has been replaced by $\varepsilon(t)$. Take the minimum of $\overline{\varepsilon^2_{M_a(t)}}(i)$, $i = t - M_a + 1,\ldots,t$, as the optimal approximate value of ξ_{M_s}. Then design

$$\xi_{M_s} = \min(\overline{\varepsilon^2_{M_a(t)}}(i)) \times \alpha \qquad i = t - M_a + 1,\ldots,t \tag{4-5}$$

where the coefficient α is chosen typically as 1.0~2.5. Since $\overline{\varepsilon^2_{M_a(t)}}(i)$ is usually computed recursively, to avoid the effect of very large residuals and to ensure the correct structure, it is necessary to detect the structure regularly. For example, a regular detection manipulation should be added to the adaptive search process and executed every M_r computational intervals where M_r is a pre-set value.

Since different systems may have different levels of residuals, the design of the absolute value of ξ_{M_s} can become difficult. It is possible to replace the critical function $\overline{\varepsilon^2_{M_s(t)}}(t)$ in (4-1) by the normalized value $\overline{\varepsilon^2_{M_s(t)}}(t) / \overline{y^2_{M_a(t)}}(t)$, where

$$\overline{y^2_{M_a(t)}}(t) = \frac{1}{M_a}\sum_{i=0}^{M_a-1} y^2(t-i) \tag{4-6}$$

Consequently, (4-4) is replaced by $\overline{\varepsilon^2_{M_a(t)}}(t) / \overline{y^2_{M_a(t)}}(t)$.

In the computation of (4-1), $\varepsilon(t-i)$ may be replaced by the priori prediction error

$$\underline{\varepsilon}(t-i) = y(t-i) - \phi(t-i)\hat{\theta}(t-i-1) \tag{4-7}$$

5. Reservation of Input Terms for Control Problems

In some applications, there are some requirements on the selection of regressors. For example, one-step-ahead adaptive controllers [16] usually require that the control term $u(t)$ can be easily expressed as a non-zero monomial in control equations given in

$$u(t) = F[\underline{y}(t+d), y(t),\ldots, y(t-n_y), u(t-1),\ldots, u(t-n_u)] \qquad (5\text{-}1)$$

where $F[.]$ is some function, $\underline{y}(t+d)$ is the desired output and d is the time delay. If nonlinear models of the form of (5-1) can be estimated on-line, good one-step-ahead control performance should be possible for complex nonlinear systems. This can easily be realized using the GFEX algorithm with a judicious selection of the regressors. Assume a predictor is represented by the following NARX model

$$y(t) = \sum_{i=1}^{m} \phi_i(y(t-d),\ldots, y(t-d-n_y), u(t-d),\ldots, u(t-d-n_u))\theta_i(t-d) + \varepsilon(t). \qquad (5\text{-}2)$$

The form of $u(t\text{-}d)$ terms in sub-models produced by the on-line estimators can be constrained using the GFEX algorithm so that after simple transformation a fitted sub-model can have the equivalent form

$$y(t) = P_1[.]\ F_u[u(t-d)] + P_2[.] \qquad (5\text{-}3)$$

where $F_u[.]$ is one term selected from $u(t-d)$ or $u^2(t-d)$ or $u^3(t-d)$, and $P_1[.]$, $P_2[.]$ are polynomials excluding $F_u[.]$. Such a restriction can reduce computational costs and avoids cases where the control equation has imaginary solutions. For example, the solution of a sub-model given by $y(t) = a_1 y(t-1)u(t-1) + b_1 u^2(t-1)$ may involve imaginary numbers for some choices of the coefficients a_1 and b_1, and hence this form of sub-model should be avoided. Similarly, the form with $u^2(t-d)$ must be used carefully. Using (5-3) the control law (5-1) is easily obtained. For example, for the two models

$$y(t) = a_2 y(t-1) + b_2 y(t-1)u(t-1) + c_2 y(t-2)u(t-1)$$

and $\quad y(t) = a_3 y(t-2)u^3(t-2) + b_3 y(t-3)u(t-4)$

where a_2, b_2, c_2, a_3 and b_3 are coefficients, the one-step-ahead control law is given as

$$u(t) = \frac{\underline{y}(t+1) - a_2 y(t)}{b_2 y(t) + c_2 y(t-1)}$$

and

$$u(t) = (\frac{y(t+2) - b_3 y(t-1)u(t-2)}{a_3 y(t)})^{1/3}$$

respectively.

In the initial design of candidate variables two methods may be adopted. The first involves considering a single form of control terms, say *u(t-d)*. In the selection of regressors every fixed sub-model must contain one or more terms which are monomials or products of *u(t-d)*. Another method is to design different forms of control terms, but the selection is still constrained to obey the above restriction on the form of terms. Under this restriction optimal sub-models cannot always be produced at every computational interval, but sub-optimal sub-models can usually be found by means of the design of candidate variables and the flexibility in selecting regressors using the GFEX algorithm.

The method for conditional selection of regressors can be extended to multi-step-ahead control and other applications. In the above discussion the problems of dividing by zero and control stability, were not considered. These issues can be solved by combining the GFEX algorithm with other techniques [16].

6. Extension of the Algorithms to Radial Basis Function Networks

RBF approximations were originally investigated for numerical interpolation in multidimensional space [24,25]. A generalized form of the RBF expansion was introduced by Broomhead and Lowe [5] and this provides a more suitable basis for system modelling and other applications of signal processing. The basic structure of a RBFN consists of three layers an input layer, one hidden layer and an output layer. The neurones in the hidden layer perform a nonlinear mapping but the neurones in the input and output layer just effect a linear transformation. A RBF approximation with n_{in} input variables (e.g. $n_{in} = n_y + n_u$) and a constant input term takes the following form

$$F(\mathbf{x}) = \theta_0 + \sum_{i=1}^{n_c} \theta_i \ \psi(\|\mathbf{x} - \underline{\mathbf{x}}_i\|) \qquad (6\text{-}1)$$

where $\mathbf{x} \in \Re^{n_{in}}$, represents the input data vector, $\underline{\mathbf{x}}_i \in \Re^{n_{in}}$, is the i th centre, $0 < i \le n_c$, n_c is the number of RBF centres (neurones), $\|.\|$ denotes the Euclidean norm, $\psi(\|.\|)$ represents the

activation function (RBF), θ_0 and θ_i are the parameters associated with the constant input and the activation function corresponding to the i th centre respectively. The centres are usually chosen as a sub-set of the input data or distributed uniformly in the input space. The choice of activation function can be varied but popular choices are the thin plate spline function

$$\psi(\mu_i) = \mu_i^2 \log \mu_i \quad ,$$

and the gaussian function

$$\psi(\mu_i) = \exp(-\mu_i^2 / \beta^2)$$

or the multiquadric or inverse multiquadric function, where $\mu_i = \|\mathbf{x} - \mathbf{x}_i\|$, and β is the width. It has been shown that RBF posses the property of the best approximation but multilayered perceptron neural networks do not [15]. Many studies of RBFN, e.g., Chen et al [9,10], Leonard and Kramer [17], Park and Sandberg [22], etc., reveal very good performance in a wide range of applications. To use RBFN suitable neurones (centres) of the hidden layer have to be determined from a large number of candidate centres. Off-line solutions to this problem have been suggested based on modified Gram-Schmidt orthogonal transformations with forward selection procedures [9,10,3]. These methods can usually be used to provide good global solutions for time-invariant systems but they are inappropriate for time-varying systems. Consequently on-line estimation should be considered. Most of the published results relevant to on-line processing methods work in such a way that the structure of RBFN, that is the number and position of the centres, is determined initially using off-line methods, and then the parameters, i.e. connection weights, are estimated on-line. A recursive hybrid algorithm based on this principle was suggested by Chen et al [11] and this algorithm has been extended and applied in adaptive noise cancellation by Billings and Fung [4]. In the hybrid algorithm k-means clustering is used to locate the positions of the centres and Givens rotation is used to update the parameter estimates at every computational step. Since the number of the centres is fixed at the beginning of the computation, the ability to model time-varying structure systems is restricted. Poggio and Girosi [23] showed that the performance of the network given in (6-1) can be improved by including direct links of the input signals to give

$$F(\mathbf{x}) = \theta_0 + \sum_{i=1}^{n_c} \theta_i \ \psi(\|\mathbf{x} - \underline{\mathbf{x}}_i\|) + \sum_{i=1+n_c}^{n_m+n_c} \theta_i \ x_{i-n_c} \qquad (6\text{-}2)$$

In fact Eqn (6-2) can be considered as an extended polynomial NARMAX model and can be concisely represented in the form of Eqn (2-3) as a pseudo-linear regression model

$$F(\mathbf{x}) = \sum_{i=0}^{m_R} \phi_i(t)\theta_i(t) + \varepsilon(t) \qquad (6\text{-}3)$$

where $m_R = n_c + n_{in} + 1$, $\phi_i(t)$ denotes regressors which are either the radial basis functions or direct links x_{i-n_c}, and the parameters can be time-varying $\theta_i(t)$. The GFEX algorithm and variants derived in Section 3, 4 and 5 can therefore be readily extended for on-line structure detection and parameter estimation of RBFN.

The algorithms derived in Section 3 and 4 can be applied directly to the RBF problem and will not be repeated here. In applications where control terms are required for one-step-ahead control, the same control terms should not be used to compute the norms μ_i. Initial centres can be arranged in a grid in the input space based on a knowledge of underlying system or selected randomly from the first few data points and then adjusted on-line using "empty" regressors or by modifying the dimension of the computational matrix (3-9). To simplify the problem in the present study the candidate centres will be considered to be fixed in advance. The derivation of more flexible training methods will be addressed in subsequent papers.

7. Experimental Results

Two experiments will be used to illustrate the methods described above. The first experiment was performed on a simulated system and the second relates to a pilot scale liquid level system. For convenience, in this section the Point-Updating GFEX algorithm will be referred to as the PU method, and the Selective Updating methods with a fixed Tolerance and with an Adaptive search scheme will be referred as the SUT and SUA methods respectively.

7.1 Experiment 1: A Simulated System

To illustrate the new algorithms, a time-varying system will be simulated where both the structure and parameters change suddenly. The first 250 data points were generated using a linear model

$$z(t) = 0.5z(t-1) + u(t-1)$$

$$y(t) = z(t) + e(t)$$

and the second 250 data points by a NARMAX model

$$z(t) = 0.2z(t-1) + 0.8u(t-1) + 0.1u^2(t-1)$$

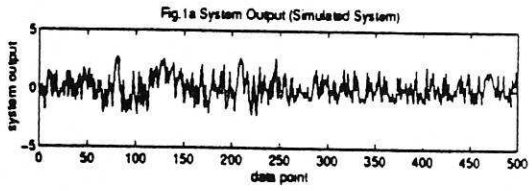
$$y(t) = z(t) + e(t)$$

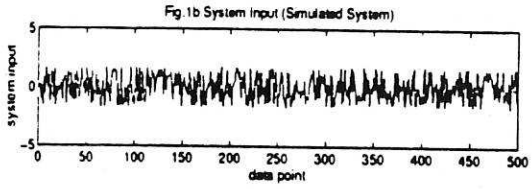
Notice that the models relating $y(t)$ to $u(t)$ will involve coloured noise terms.

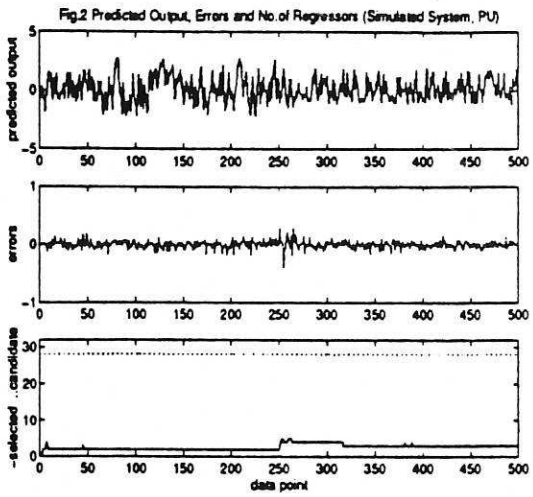
In the test, the input signal $u(t)$ was an independent sequence of uniform distribution with zero mean and variance of 1.03, and the noise signal $e(t)$ was a Gaussian white noise with zero mean and variance 0.005. The system output is illustrated in Fig.1. The first five hundred data samples were used to fit a time-varying NARMAX model using the PU method with initial parameters

$$n_y = n_u = n_\varepsilon = n_l = 2, \quad \xi_s = 0.01, \quad \lambda = 0.97.$$

The total number of the candidate variables is 28. The results are tabulated in Table 1 and illustrated in Fig. 2 and 3. Fig.2 clearly shows the change in the number of selected regression variables from point 250 onwards. Fig. 3 illustrates a similar effect showing how the $u^2(t-1)$ term is selected and the parameters of the linear terms are updated. The estimator tracks both the change in model structure and the changes in individual parameter values.







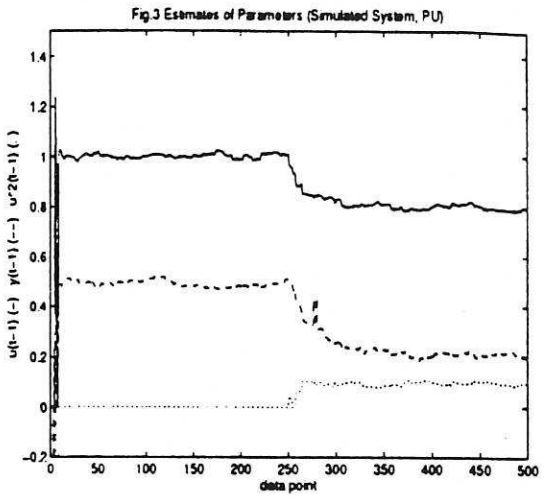


Table 1: Simulated System (point-updating, tolerance)

point	term	θ	$\hat{\theta}$	*ERR*
250'th	$u(t-1)$	1.0000	1.0116	0.7332
	$y(t-1)$	0.5000	0.5110	0.2636
	$u^2(t-1)$	none	none	*
500'th	$u(t-1)$	0.8000	0.7927	0.9236
	$y(t-1)$	0.2000	0.2015	0.0519
	$u^2(t-1)$	0.1000	0.0981	0.0210
$sse = 2.3338$	$\sigma_\varepsilon^2 = 0.0047$		$\varepsilon^T\varepsilon / y^T y = 0.0043$	

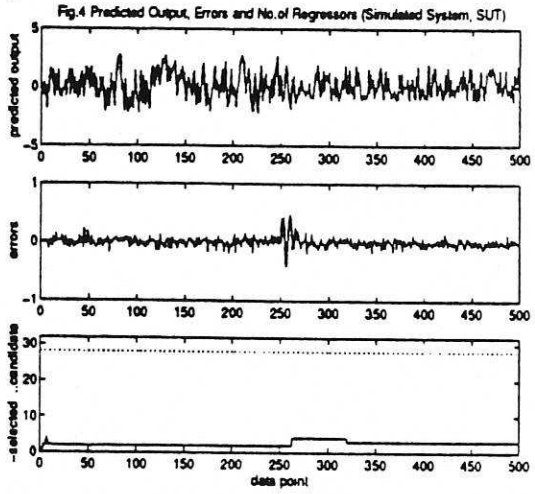
The same data set was processed using the SUT algorithm with

$n_y = n_u = n_\varepsilon = n_l = 2, \quad \xi_s = 0.01, \quad \lambda = 0.97,$

$M_s = 20, \quad \xi_{M_s} = 0.05.$

Model structure updating was only triggered at 43 data points and this is much less than in the point-updating method. Most of the updating operations happened in the initial stages of processing and the period when the system structure appears to change, Fig. 4. Inspection of the results, Table 2 shows that the estimator produces the correct system structure and good parameter estimates. The sum of squared errors ($sse = \sum_{t=1}^{500} \varepsilon^2(t)$) was 3.395 which is larger than in the point-updating methods, Table 1. Since the SUT method selectively updates the model structure based on the critical value ξ_{M_s} and does not perform point by point operations based on ξ_s, the errors may be slightly larger over the period when the structure changes. But such errors reduce continuously when the structure is stable. It is worth noting that at the 250'th and 500'th point the sub-models from PU and SUT are the same although

the number of structure updates has been considerably reduced in SUT. The parameters profiles



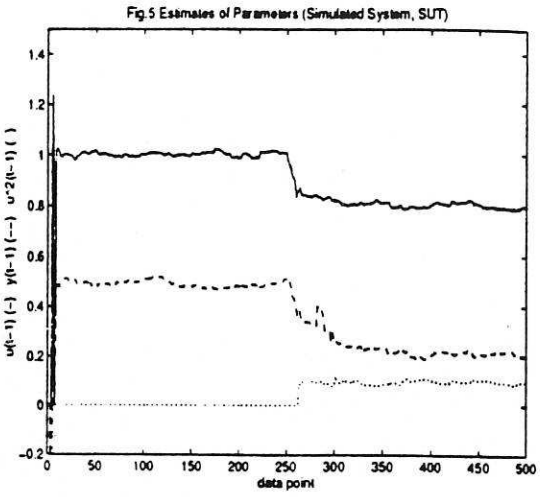


Table 2: Simulated System (selective-updating, tolerance)

point	term	θ	$\hat{\theta}$	*ERR*
250'th	$u(t-1)$	1.0000	1.0116	0.7332
	$y(t-1)$	0.5000	0.5110	0.2636
	$u^2(t-1)$	none	none	*
500'th	$u(t-1)$	0.8000	0.7927	0.9236
	$y(t-1)$	0.2000	0.2015	0.0519
	$u^2(t-1)$	0.1000	0.0981	0.0210
Structure Upd. No.= 43	$sse = 3.3951$	$\sigma_\varepsilon^2 = 0.0068$	$\varepsilon^T\varepsilon / y^T y = 0.0062$	

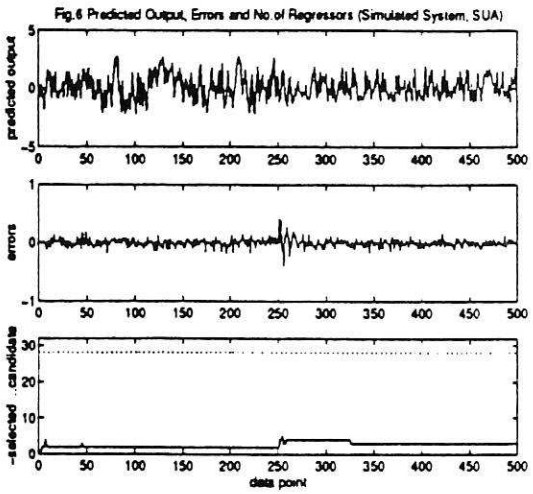
in Fig. 5 show that the transition between the 250'th and 350'th point is still smooth, especially for the parameter $y(t\text{-}1)$.

The SUA method was also used to process this data set. The initial parameters were

$$n_y = n_u = n_\varepsilon = n_l = 2, \quad \xi_s = 0.01, \quad \lambda = 0.97,$$

$$M_s = 20, \quad M_r = 30, \quad M_a = 70, \quad \alpha = 1.1.$$

The sub-models at the 250'th and 500'th points indicate that the adaptive search scheme has performed well and the fitting is good, Table 3 and Fig. 6. The *sse* is 2.621 which is lower than that from the SUT method. In the period where the structure changes the transition of $y(t\text{-}1)$ (Fig. 7) is not very smooth, compared with the PU and SUT routines. This is due to the adaptive search process used for the tolerance. Although the total number of data points was 500 the operation of updating the structure was only performed at 218 points.



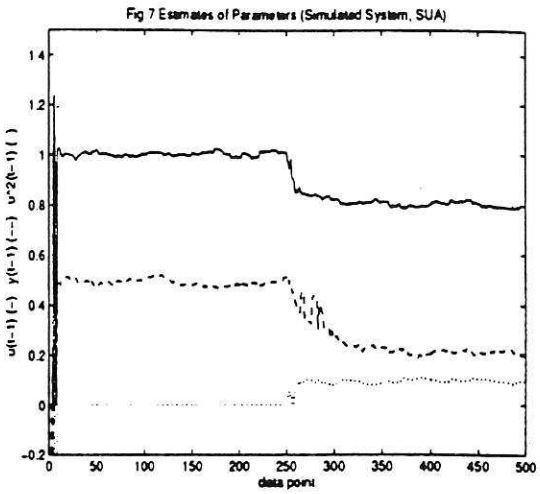


Table 3: Simulated System (selective-updating, adaptation)

point	term	θ	$\hat{\theta}$	*ERR*
250'th	$u(t-1)$	1.0000	1.0116	0.7332
	$y(t-1)$	0.5000	0.5110	0.2636
	$u^2(t-1)$	none	none	*
500'th	$u(t-1)$	0.8000	0.7927	0.9236
	$y(t-1)$	0.2000	0.2015	0.0519
	$u^2(t-1)$	0.1000	0.0981	0.0210
Structure Upd. No.= 218 $sse = 2.6205$ $\sigma_\varepsilon^2 = 0.0053$ $\varepsilon^T\varepsilon / \mathbf{y}^T\mathbf{y} = 0.0048$				

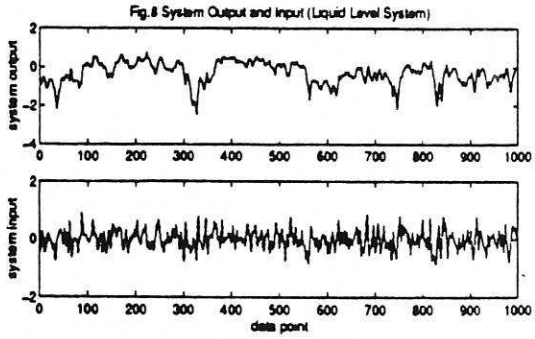
7.2 Experiment 2: A Pilot Scale Liquid Level System

The system consists of a DC water pump feeding a conical flask which in turn feeds a square tank. The system input is the voltage to the pump motor and the system output is the water level in the conical flask. 1000 data samples were collected from this system and are used in the present study to demonstrate how the new algorithm performs for a real system. The input and output data are illustrated in Fig. 8.

First the PU method was applied to process the 1000 data samples. The initial parameters were

$$n_y = n_u = n_\varepsilon = n_l = 2, \quad \xi_s = 0.005, \quad \lambda = 0.99.$$

and the number of candidates were 28. The predicted output and residuals are shown in Fig. 9 and the sub-model at the 1000'th point is tabulated in Table 4. Inspection of the result shows that the predicted output matches the real output giving a sum of squared errors ($sse = \sum_{t=1}^{1000} \varepsilon^2(t)$) of 1.817. The variance of the residuals σ_ε^2 (0.0018) was also small, compared



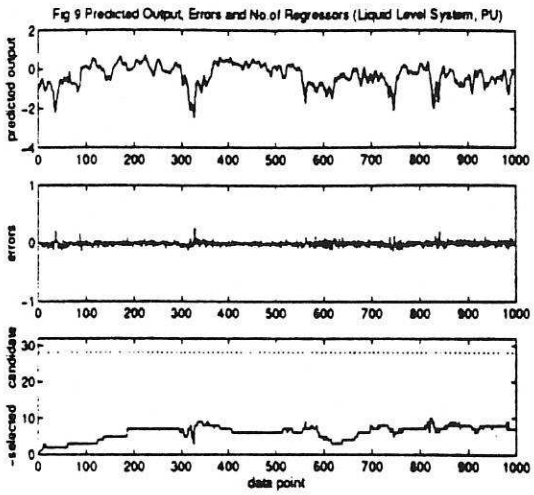


Table 4: Liquid Level System (point-updating, tolerance)

point	term	$\hat{\theta}$	*ERR*
1000'th	$y(t-1)$	1.0177	0.9586
	$u(t-1)$	0.4660	0.0286
	$u(t-2)$	-0.2134	0.0043
	$y^2(t-2)$	0.0705	0.0013
	$u(t-2)u(t-1)$	-0.1157	0.0007
	$e(t-1)$	-0.2286	0.0002
sse =1.8167	σ_ε^2 =0.0018	$\varepsilon^T\varepsilon / y^T y$ = 0.0044	

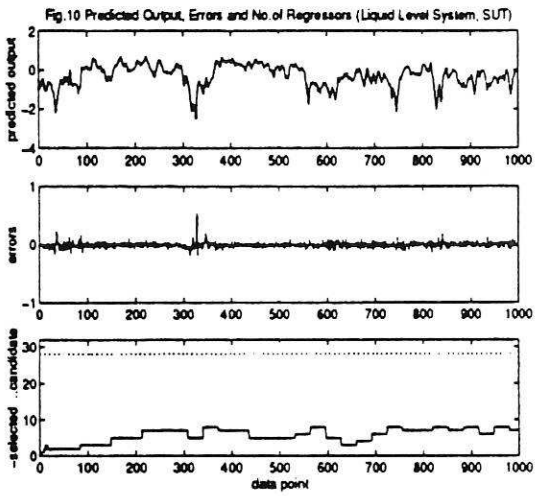
with the results (0.0020) produced from a recursive prediction error parameter estimator [6]. Although the system output shows a large change at the 325'th point the estimator tracks the change quickly and the residuals only increase over a short interval. The adjustment behaviour can be observed from the changes of the number of regression variables in Fig. 9.

The SUT method was also used to process the same data set. The initial parameters were

$$n_y = n_u = n_\varepsilon = n_l = 2, \quad \xi_s = 0.005, \quad \lambda = 0.99,$$

$$M_s = 20, \quad \xi_{M_s} = 0.05.$$

The results tabulated in Table 5 shows that the *sse* becomes 2.396 and σ_ε^2 0.0024. Both are just higher than the results obtained using the PU method. The maximum residual is slightly high but is only 0.536, while in the PU method this was 0.270. It is more significant that the points at which the structure was updated have been reduced to 48, only one twentieth of the updates required by the PU method. The variation of the model structure can be observed from the number of the regressors illustrated in Fig. 10.



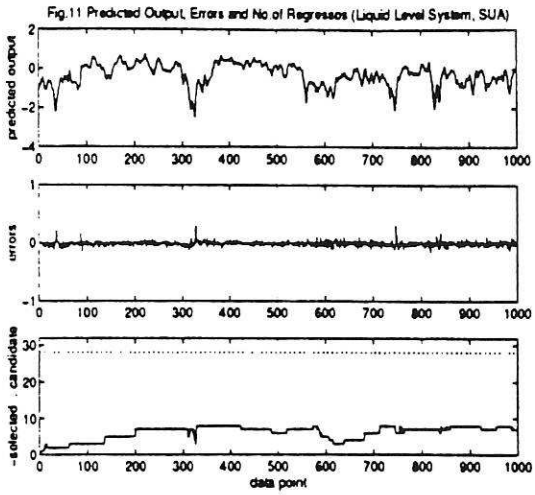


Table 5: Liquid Level System (selective-updating, tolerance)

point	term	$\hat{\theta}$	*ERR*
1000'th	$y(t-1)$	1.0155	0.9586
	$u(t-1)$	0.4207	0.0286
	$u(t-2)$	-0.2162	0.0043
	$y^2(t-2)$	0.0712	0.0013
	$u^2(t-1)$	-0.0726	0.0006
	$e(t-1)$	-0.2810	0.0003
	$u(t-1)y(t-1)$	-0.0982	-0.0004
Structure Upd. No.= 48			
sse = 2.3963	σ_ε^2 = 0.0024	$\varepsilon^T\varepsilon / \mathbf{y}^T\mathbf{y}$ = 0.0059	

Table 6: Liquid Level System (selective-updating, adaptation)

point	term	$\hat{\theta}$	*ERR*
1000'th	$y(t-1)$	1.0196	0.9586
	$u(t-1)$	0.4309	0.0286
	$u(t-2)$	-0.2205	0.0043
	$y^2(t-2)$	0.0727	0.0013
	$u(t-1)u(t-2)$	-0.0750	0.0007
	$e(t-1)$	-0.3257	0.0004
	$u(t-1)y(t-1)$	-0.0861	0.0003
Structure Upd. No.= 294			
sse = 1.9865	σ_ε^2 = 0.0020	$\varepsilon^T\varepsilon / \mathbf{y}^T\mathbf{y}$ = 0.0049	

In the third part of this experiment, the SUA method was employed. The initial parameters were designed as

$$n_y = n_u = n_\varepsilon = n_l = 2, \quad \xi_s = 0.005, \quad \lambda = 0.99,$$

$$M_s = 20, \quad M_r = 30, \quad M_a = 65, \quad \alpha = 1.1.$$

With this initialization a time-varying model was fitted and the sub-model at the 1000'th point is shown in Table 6. The results from this algorithm are illustrated in Fig. 11. The *sse* is only

1.987 and σ_ε^2 is 0.0020. Comparing the selected terms at the 1000' th point with those from the PU and SUT method, it is found that the first four terms of the three sub-models are the same and have the same ERR_i value (the contribution to the output). It is worth noting that the sum of the ERR_i values for the first four terms is about 0.9928. This means that the rest of the terms in these sub-models only make a very small contribution even though these terms are different. This difference in the model structures comes from the generation of the critical values and the fact that the model structures are updated at different time instants. These differences also induce different residual sequences and hence affect whether noise terms are included in the sub-models. For example, the sub-models at the 1000'th point from the PU, SUT and SUA methods involve a noise term. The points at which the model structure was updated were reduced to 294 in the SUA algorithm.

This data set was also used to test the method of constraining a control input term in the model for the one-step-ahead control problem based on the nonlinear polynomial model in Eqn. (2-1). The test investigated whether there was a possibility of fitting a time-varying model containing specified control terms. Using the point-updating method with the initial parameters

$$n_y = n_u = n_l = 2, \quad n_\varepsilon = 0, \quad \xi_s = 0.005, \quad \lambda = 0.99,$$

but excluding $u^2(t-1)$, a time-varying NARX model with 14 candidate variables was fitted to the liquid level system. The results are illustrated in Fig. 12 and the sub-model at the 1000'th point is tabulated in Table 7. The *sse* is 1.980 and this value is close to the result, 1.817, obtained using the PU algorithm in the first part of the experiment, Table 4. It is easy, to obtain the control equation at every time instant from the sub-models produced. At the 1000'th point, for example, the control equation is

$$u(t) = \frac{y(t+1) - 1.01y(t) + 0.210u(t-1) - 0.065y^2(t-1)}{0.424 - 0.0865u(t-1) - 0.0756y(t)}.$$

Similarly, using a set of initial parameters given as

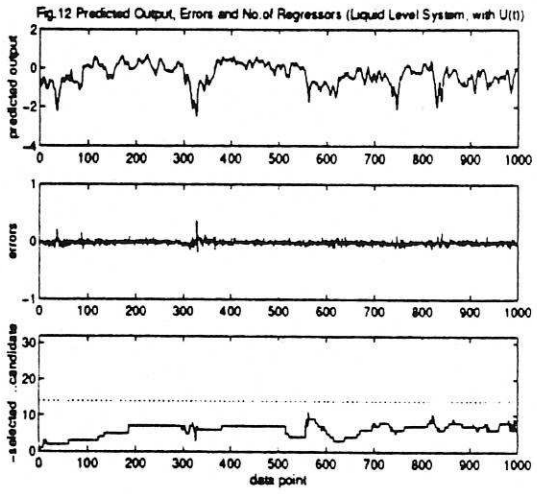
$$n_y = n_u = 2, \quad n_l = 3, \quad n_\varepsilon = 0, \quad \xi_s = 0.005, \quad \lambda = 0.99,$$

but excluding all the terms containing $u(t\text{-}1)$ except $u^2(t-1)$, $u^2(t-1)u(t-2)$, $u^2(t-1)y(t-1)$ and $u^2(t-1)y(t-2)$, to fit one time-varying model with the term $u^2(t-1)$ and

24 candidate regressors, the result indicated that the *sse* rises, Table 8, because the term $u(t-1)$ is one of the optimal regression variables for this system. But the amplitude of the residuals do not rise greatly, see Fig. 13. The control equation corresponding to the sub-model at the 1000'th point is given as

$$u(t) = \left(\frac{y(t+1) - 1.442y(t) + 0.481y(t-1) + 0.345u^3(t-1) - 0.043y^2(t-1) + 0.020}{-0.089 + 0.673u(t-1)}\right)^{1/2}$$

respectively. The above results demonstrate that using the GFEX algorithm it is easy to fit a time-varying model containing specified control input terms to a nonlinear system and thus to perform one-step-ahead control.



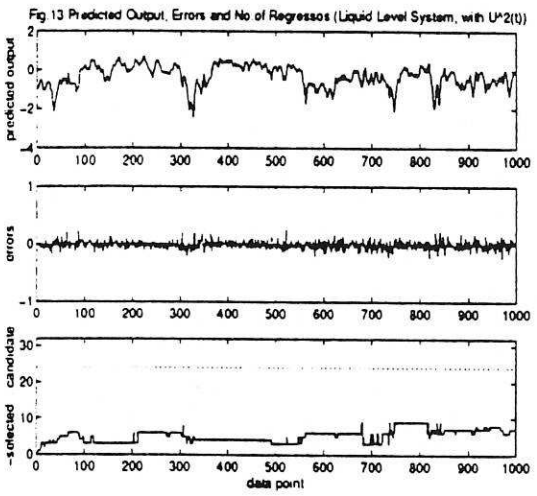


Table 7: Liquid Level System (one-step-ahead control with $u(t)$)

point	term	$\hat{\theta}$	*ERR*
1000'th	$y(t-1)$	1.0104	0.9586
	$u(t-1)$	0.4236	0.0286
	$u(t-2)$	-0.2099	0.0043
	$y^2(t-2)$	0.0650	0.0013
	$u(t-2)\,u(t-1)$	-0.0865	0.0007
	$u(t-1)\,y(t-1)$	-0.0756	0.0002
sse =1.9804	σ_ε^2 =0.0020	$\varepsilon^T\varepsilon / y^T y$ = 0.0048	

Table 8: Liquid Level System (one-step-ahead control with $u^2(t)$)

point	term	$\hat{\theta}$	*ERR*
1000'th	$y(t-1)$	1.4422	0.9586
	$y(t-2)$	-0.4808	0.0212
	$u^2(t-1)$	-0.0892	0.0018
	$u^2(t-1)u(t-2)$	0.6732	0.0012
	$u^3(t-2)$	-0.3447	0.0014
	$y^2(t-2)$	0.0429	0.0006
	constant	-0.0197	0.0003
sse =4.5349	σ_ε^2 =0.0045	$\varepsilon^T\varepsilon / \mathbf{y}^T\mathbf{y}$ = 0.0111	

The PU, SUT and SUA were applied to training a radial basis function neural networks of the liquid level system. The main aim of this experiment was to demonstrate the ability of RBFN's to track the system dynamics. The first 40 data points were considered as candidate centres, and the significant centres to the output were selected from these candidate centres. The input signals were not linked directly to the output layer and the thin-plate-spline function was considered as the activation function. The initial parameters for three algorithms are indicated as follows

PU: $n_y = 3,\ n_u = 4,\ n_\varepsilon = 0,\ n_l = 1,$ No.of Centres =40, ξ_s = 0.005, $\lambda = 0.99,$

SUT: $n_y = 3,\ n_u = 4,\ n_\varepsilon = 0,\ n_l = 1,$ No.of Centres =40, ξ_s = 0.005, $\lambda = 0.99,$

$M_s = 20,\ \xi_{M_s} = 0.05.$

SUA: $n_y = 3,\ n_u = 4,\ n_\varepsilon = 0,\ n_l = 1,$ No.of Centres =40, ξ_s = 0.005, $\lambda = 0.99,$

$M_s = 20,\ M_r = 30,\ M_a = 65,\ \alpha$ = 1.1.

From the numerical results shown in Table 9, it is observed that the *sse* from PU (5.238) is higher that the value obtained from PU based on the NARX model. Large errors appeared in several periods when the system output changed very quickly, but the estimator could still track such variations, see Fig.14. It is worth noting that the first 40 data points were used as candidate centres. If these candidate centres are distributed properly in the input space, the prediction errors should reduce greatly. But the result shows that the distribution of these candidate centres is only sub-optimal. To improve the accuracy of the prediction, on-line processing methods for optimizing the candidate centres should be considered. The relevant issues will not be considered here but will be addressed in forthcoming papers. It is also worth noting that the computation did not involve any direct input links. The direct links usually

improve the linear mapping of RBF networks. From the liquid level NARX model experiments it was shown that $y(t-1)$, $u(t-1)$ and $u(t-2)$ are three of the optimal regressors. If the existing network was augmented by adding some of these global variables, the performance should improve. This consideration has been confirmed by the result in Tables 9 obtained from the method relevant to one-step-ahead control where one direct link, $u(t-1)$, was used. The number of updates to the network structure were 52 in SUT and 343 in SUA respectively. These are only a fraction of the updates used in PU, but the cost is an increase in the errors, illustrated in Fig.15, 16 and Tables 9. Therefore the use of SUT and SUA should be balanced between the accuracy of the prediction and the requirement of the constraints for control.

The data set was used to investigate the method of constraining a control input term in the RBFN for one-step-ahead control. The initial parameters were given as

$n_y = 3,\ n_u = 4,\ n_\varepsilon = 0,\ n_l = 1,$ No.of Centres =40, $\xi_s =$ 0.005, $\lambda = 0.99,$

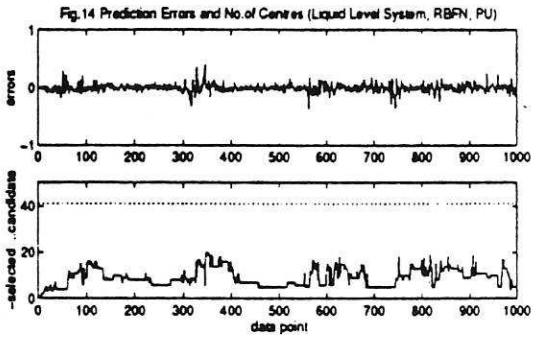
and the first 40 data points were considered as the candidate centres. Only one input $u(t-1)$ was linked directly to the output node. Consequently the control input can easily be computed using the equation

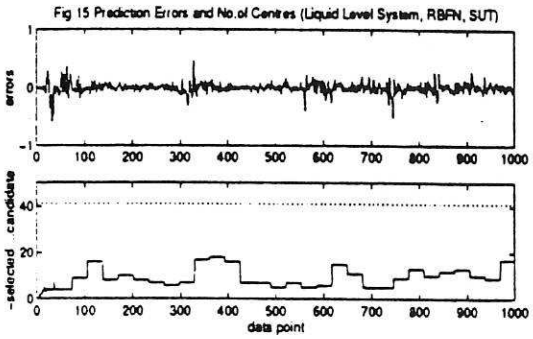
$$u(t) = \frac{\underline{y}(t+1) - \hat{\theta}_0 - \sum_{i=1}^{n_c} \hat{\theta}_i \psi(\|\mathbf{x}_t - \underline{\mathbf{x}}_i\|)}{\hat{\theta}_u}$$

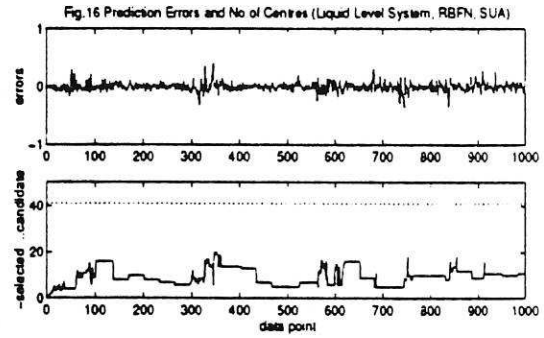
where $\mathbf{x}_t$ is the data point available up to time t, that lies in the input space made up of $y(t)$, $y(t-1)$, $y(t-2)$, $u(t-1)$, $u(t-2)$ and $u(t-3)$. Fig.17 shows that the estimator can track the system dynamics well. Since the input signal was linked directly to the output, the performance of the estimator was improved and the *sse* shown in Table 9 is less than the values obtained from other RBFN's without direct links. But the value is still higher than the level obtained from the prediction for one-step-ahead control using the NARX model, Table.7. This is because the initialzation of the candidate centres has not been optimized as mentioned above.

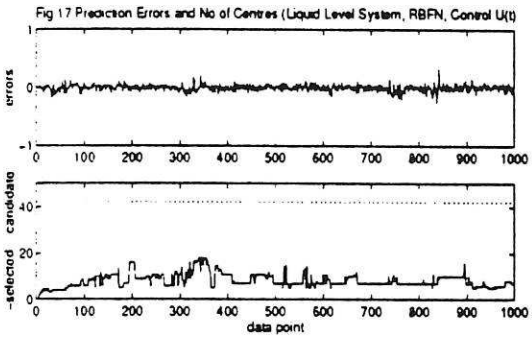
Table 9: Liquid Level System (RBFN with PU, SUT, SUA and Control Relevant method)

Method	*sse*	σ_e^2	$\varepsilon^T\varepsilon / y^T y$	Upd. No.
PU	5.2380	0.0052	0.0128	997
SUT	8.3698	0.0084	0.0204	52
SUA	5.7469	0.0057	0.0140	343
Control Relevant	2.9920	0.0030	0.0073	996









8. Conclusions

A square-root-free GFEX algorithm and variants of it have been derived for nonlinear NARMAX models and RBF networks. The operation of these methods does not depend on regressor-shift properties nor is there a need to store all regression data and orthogonal vectors. The numerical results show that the algorithm can determine the model structure on-line and update estimates of the corresponding parameters, so that the sum of squared errors is minimized.

The selective structure updating method can greatly reduce the computational cost using both a fixed tolerance or an adaptive search procedure. The estimates produced by these methods often exhibit a smooth transient when structure changes. The fixed tolerance method has a relatively low computational cost but the design depends on a knowledge of the

underlying system. The adaptive search method avoids this problem and can provide more flexibility for adaptive signal processing.

A conditional selection of regressors for one-step-ahead control problems has also been discussed. The GFEX algorithm provides an easy method for selecting the regressors and designing one-step-ahead controllers. A real system was used to demonstrate the performance of this algorithm which easy be extended to other adaptive control applications.

The application of GFEX to training radial basis function networks provides an ability to track changing systems. However the initialization of the candidate centres is very important. Methods which use the fixed centres selected from the initial data set may not be the best choice, unless these points are distributed uniformly in the input space. Worthwhile improvement can be obtained using either k-means clustering or adaptive generation of the candidate centres and these issues will be addressed in subsequent papers.

Appendix I: A Modified Version of Square-Root-Free Givens Routine

The transformation of two rows, for example, the i'th and j'th rows, is presented as follows

$$\textit{row } i:\ 0,\ldots,0,\sqrt{d_i}\,\underline{c^*_{i,k}},\sqrt{d_i}\,\underline{c^*_{i,k+1}},\ldots,\sqrt{d_i}\,\underline{c^*_{i,m+1}} \ \Rightarrow\ 0,\ldots,0,\sqrt{d_i'}\,\underline{c^{*'}_{i,k}},\sqrt{d_i'}\,\underline{c^{*'}_{i,k+1}},\ldots,\sqrt{d_i'}\,\underline{c^{*'}_{i,m+1}}$$

$$\textit{row } j:\ 0,\ldots,0,\sqrt{d_j}\,\underline{c^*_{j,k}},\sqrt{d_j}\,\underline{c^*_{j,k+1}},\ldots,\sqrt{d_j}\,\underline{c^*_{j,m+1}} \ \Rightarrow\ 0,\ldots,0,\ 0,\ \sqrt{d_j'}\,\underline{c^{*'}_{j,k+1}},\ldots,\sqrt{d_j'}\,\underline{c^{*'}_{j,m+1}}$$

where ' denotes transformed quantities. Note that the above transformation does not require $\underline{c^*_{i,k}} = 1$.

$$d_i' = (\sqrt{d_i}\,\underline{c^*_{i,k}})^2 + (\sqrt{d_j}\,\underline{c^*_{jk}})^2 = d_i(\underline{c^*_{i,k}})^2 + d_j(\underline{c^*_{jk}})^2 \qquad \text{(A-1a)}$$

$$a = \frac{d_i\,\underline{c^*_{i,k}}}{d_i'} \qquad b = \frac{d_j\,\underline{c^*_{jk}}}{d_i'} \qquad \text{(A-1b, A-1c)}$$

$$\underline{c^{*'}_{ip}} = a\,\underline{c^*_{ip}} + b\,\underline{c^*_{jp}} \qquad \text{(A-1d)}$$

$$d_j' = \frac{d_i d_j}{d_i'}\left(= \frac{a d_j}{\underline{c^*_{jk}}}\right) \qquad \text{(A-1e)}$$

$$\underline{c^{*'}_{jp}} = \underline{c^*_{i,k}}\,\underline{c^*_{jp}} - \underline{c^*_{jk}}\,\underline{c^*_{ip}} \qquad \text{(A-1f)}$$

where $p = k,\ldots,m+1$. It is obvious that if $\underline{c^*_{i,k}} = 1$ this is equivalent to the formulas in (Gentleman 1973, 1974).

Appendix II: A Subroutine for Selecting Regressors

Assume that the j'th optimal regressor is being selected. The column vectors of $\mathbf{R}^*$, $\mathbf{r}_p^*$, $p = j,\ldots,m$, have the form

$$\mathbf{r}_p^* = [r_{1p}^* r_{2p}^* \ldots r_{pp}^* 0 \ldots 0]^T .$$

set $v_{j(j)}^2 = d_j^* (v_j^*)^2$ and the three auxiliary variables

$$d_{j(p)}^{(j)} = d_j^* , \quad r_{j(p)}^{(j)} = r_{jp}^* , \quad v_{j(p)}^{(j)} = v_j^* .$$

for $i = j+1,\ldots,p$. Based on (A-1) calculate

$$d_{j(p)}^{(i)} = d_{j(p)}^{(i-1)} (r_{j(p)}^{(i-1)})^2 + d_i^* (r_{ip}^*)^2$$

$$v_{j(p)}^{(i)} = \frac{d_{j(p)}^{(i-1)} r_{j(p)}^{(i-1)}}{d_{j(p)}^{(i)}} v_{jp}^{(i-1)} + \frac{d_{j(p)}^{(i-1)} r_{i(p)}^*}{d_{j(p)}^{(i)}} r_{ip}^*$$

and finally

$$v_{j(p)}^2 = d_{j(p)}^{(p)} (v_{j(p)}^{(p)})^2 .$$

References

[1] S.A. Billings and W.S.F. Voon, 'A prediction error and stepwise regression estimation algorithm for nonlinear systems', *Int. J. Control*, **44**, 803-822 (1986).

[2] S.A. Billings and S. Chen, 'Extended model set, global data and threshed model identification of severely nonlinear systems', *Int. J. Control*, **50**, 1897-1923 (1989).

[3] S.A. Billings and S. Chen, 'Neural networks and system identification', in *Neural Networks for Control and Systems*, K. Warwick, G.W. Irwin and K.J. Hunt (eds), P. Peregrinus, London, 181-205 (1992).

[4] S.A. Billings and C.F. Fung, 'Recurrent radial basis function networks for adaptive noise cancellation', *J. Neural Networks*, **8**, 273-290 (1995).

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

[6] S. Chen and S.A. Billings, 'Recursive prediction error parameter estimator for nonlinear models', *Int. J. Control*, **49**, 1989, 569-594 (1989).

[7] S. Chen and S.A. Billings, 'Representations of nonlinear systems the NARMAX model', *Int. J. Control*, **48** (1989).

[8] S. Chen, S.A. Billings and W. Luo, 'Orthogonal least squares methods and their application to non-linear system identification', *Int. J. Control*, **50**, 1873-1896 (1989).

[9] S. Chen, S.A. Billings, C.F.N. Cowan and P.M. Grant, 'Non-linear system identification using radial basis functions', *Int. J. Systems Sci.*, **21**, 2513-2539 (1990).

[10] S. Chen, S.A. Billings, C.F.N. Cowan and P.M. Grant, 'Practical identification of NARMAX models using radial basis functions', *Int. J. Control*, **52**, 1327-1350 (1990).

[11] S. Chen, S.A. Billings and P.M. Grant, 'Recursive hybrid algorithm for nonlinear system identification using radial basis function networks', *Int. J. Control*, **55**, 1051-1070 (1992).

[12] D. Clarke and P.J. Gawthrop, 'Self-tuning controller', *Proc. Inst. Elec. Eng.*, **122**, 929-934 (1975).

[13] W.M. Gentleman, 'Least squares computation by Givens transformation without square roots', *J. Inst. Math. and Appl.*, **12**, 329-336 (1973).

[14] W.M. Gentleman, 'Regression problems and the QR decomposition', *Bulletin Inst. Math. and Appl.*, **10**, 195-197 (1974).

[15] F. Girosi and T. Poggio, 'Networks and the best approximation property', *Biological Cybernetics*, **63**, 169-179 (1990).

[16] G.C. Goodwin and K.S. Sin, *Adaptive Filtering, Prediction and Control*, Englewood Cliffs, Prentice-Hall Inc., New Jersey, 1984.

[17] J.A. Leonard and M.A. Kramer (1991): 'Radial basis function networks for classifying process faults', *IEEE Control Systems Mag.* 31-38, (1991).

[18] I.J. Leontaritis and S.A. Billings, 'Input-output parametric models for nonlinear systems, Part I: Deterministic nonlinear systems; Part II: Stochastic nonlinear systems', *Int. J. Control*, **41**, 303-344 (1985).

[19] Y.P. Liu, M.J. Korenberg, S.A Billings. and M.B. Fadzil, 'The non-linear identification of a heat exchanger', 26th IEEE Conference on Decision and Control, Dec.9-11, Los Angeles, U.S.A (1987).

[20] W. Luo, S.A. Billings and K.M. Tsang, 'On-line structure detection and parameter estimation with exponential windowing for nonlinear systems', (1994). (submitted for publication).

[21] W. Luo and S.A. Billings, 'Adaptive model selection and estimation for nonlinear systems using a sliding data window', (1994) (submitted for publication).

[22] J. Park and I.W. Sandberg, 'Universal approximation using radial basis function networks', *Neural Computation*, **3**, 246-257 (1991).

[23] T. Poggio and F. Girosi, 'Networks for application and learning', *Proc. IEEE*, **78**, 1481-1496 (1990).

[24] M.J.D. Powell, 'Radial basis functions for multivariable interpolation: a review', in *Algorithms for Approximation*, J.C. Mason and M.G. Cox (Eds.), Oxford University Press, Oxford, 143-167 (1985).

[25] M.J.D. Powell, 'Radial basis function approximations to polynomials', in Proc. 12th biennial Numerical Analysis Conf., Dundee, 223-241 (1987).

List of Figure Captions

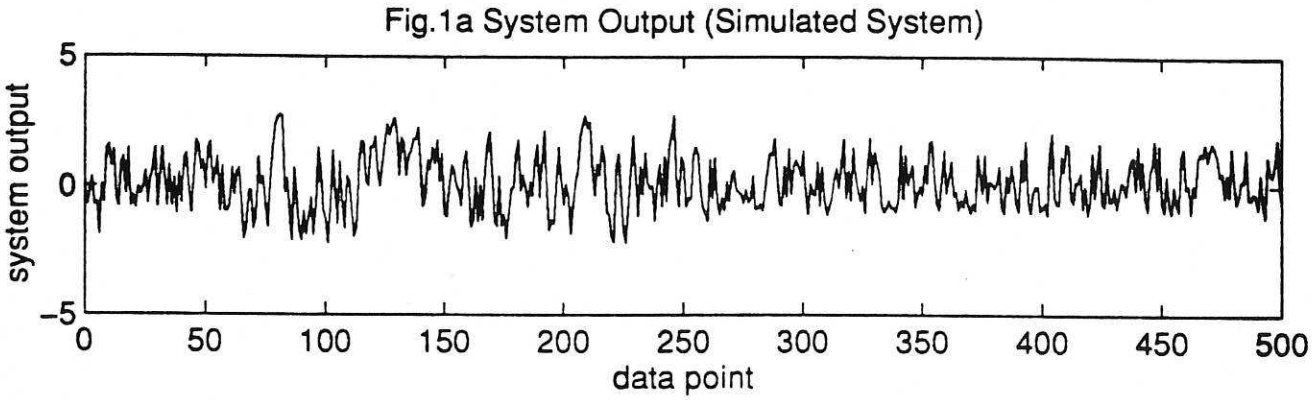


Fig.1a System Output (Simulated System)

Fig.1b System Input (Simulated System)

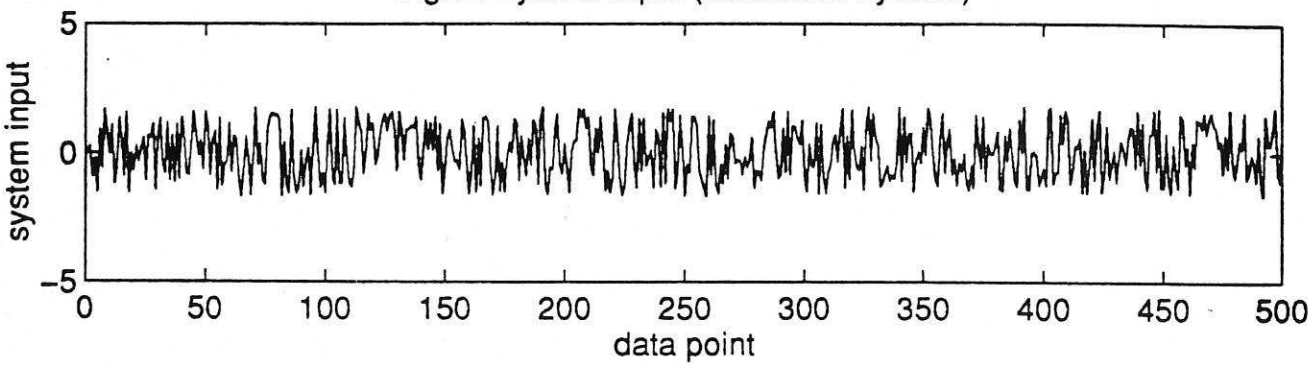


Fig.2 Predicted Output, Errors and No.of Regressors (Simulated System, PU)

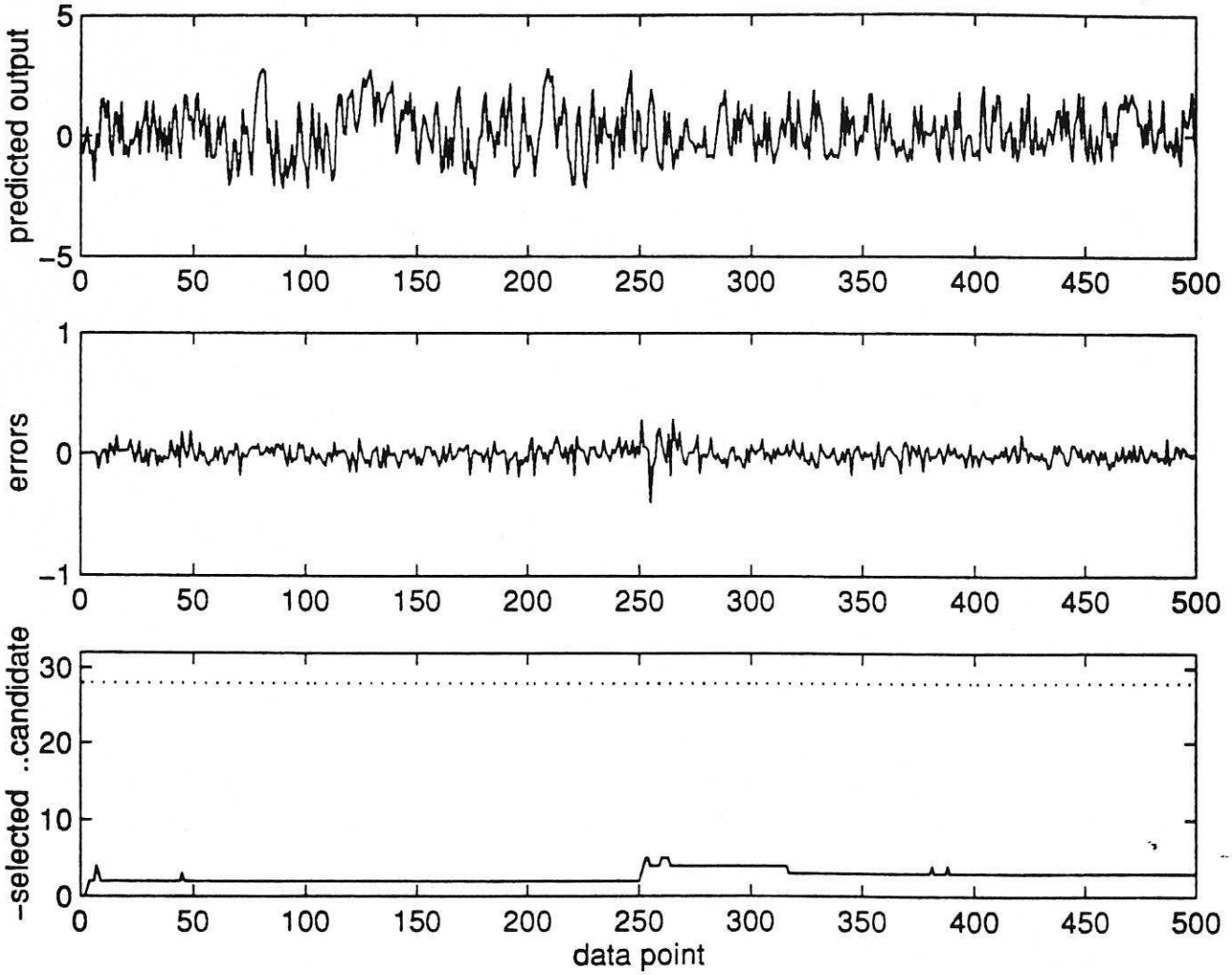


Fig.3 Estimates of Parameters (Simulated System, PU)

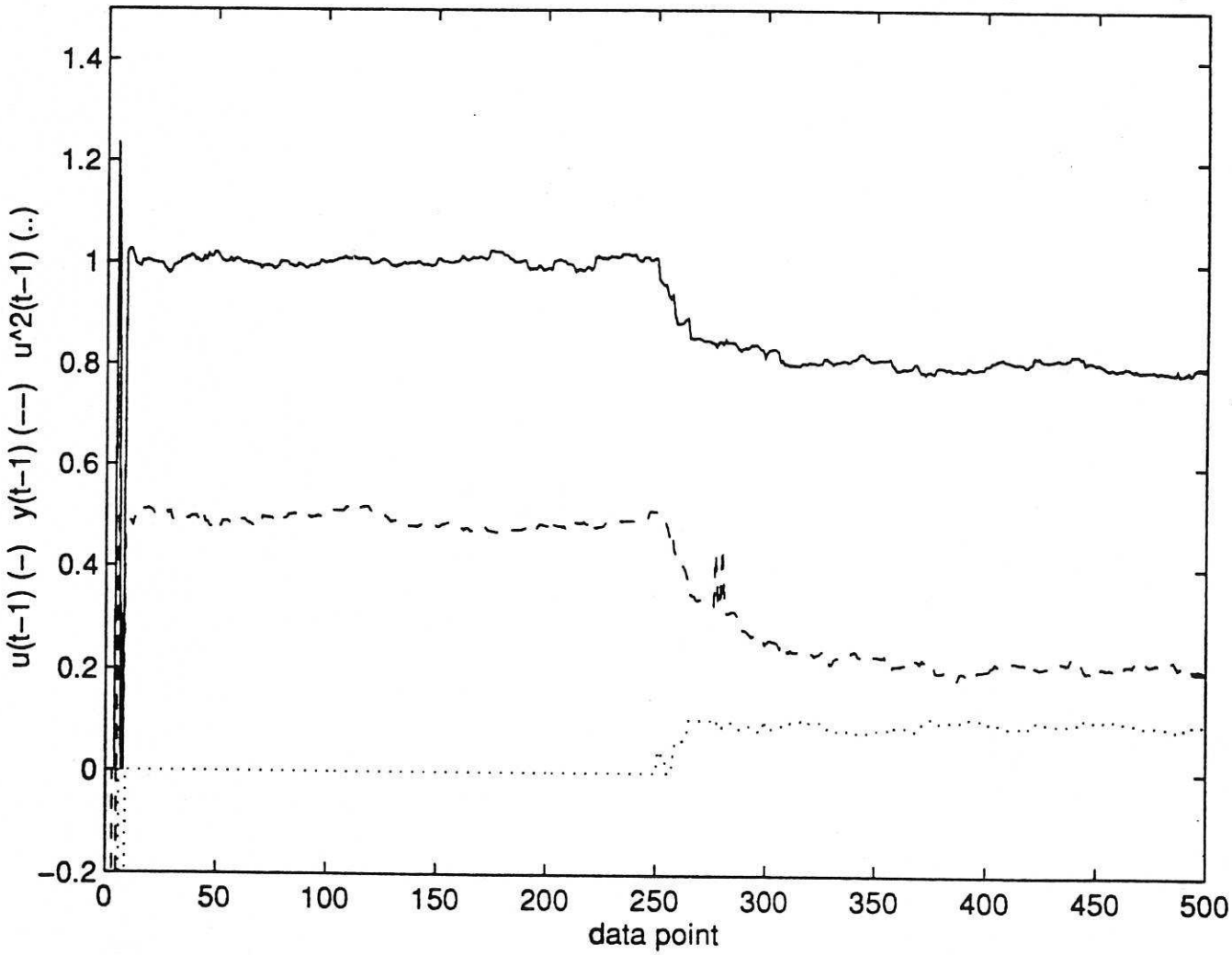


Fig.4 Predicted Output, Errors and No.of Regressors (Simulated System, SUT)

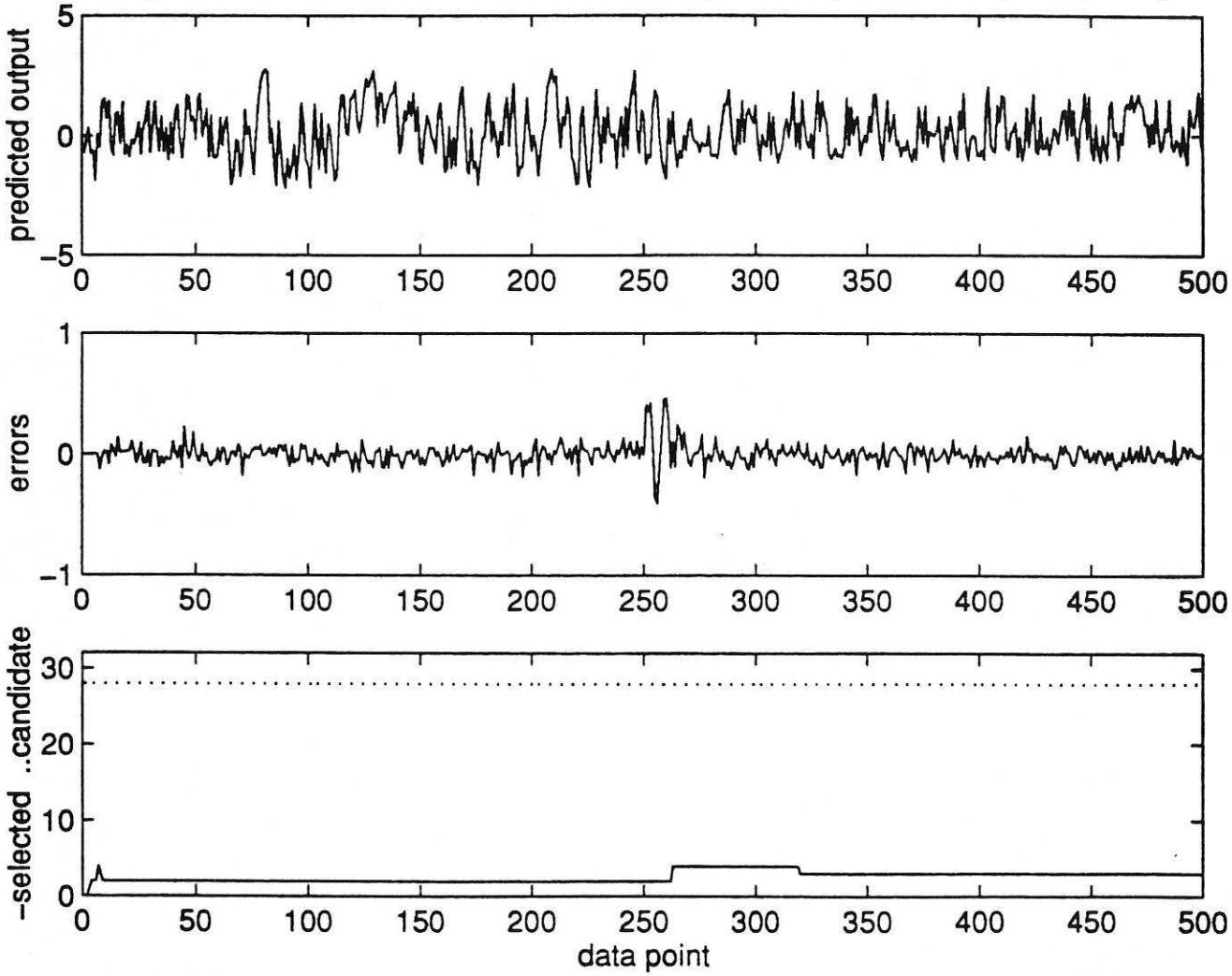


Fig.5 Estimates of Parameters (Simulated System, SUT)

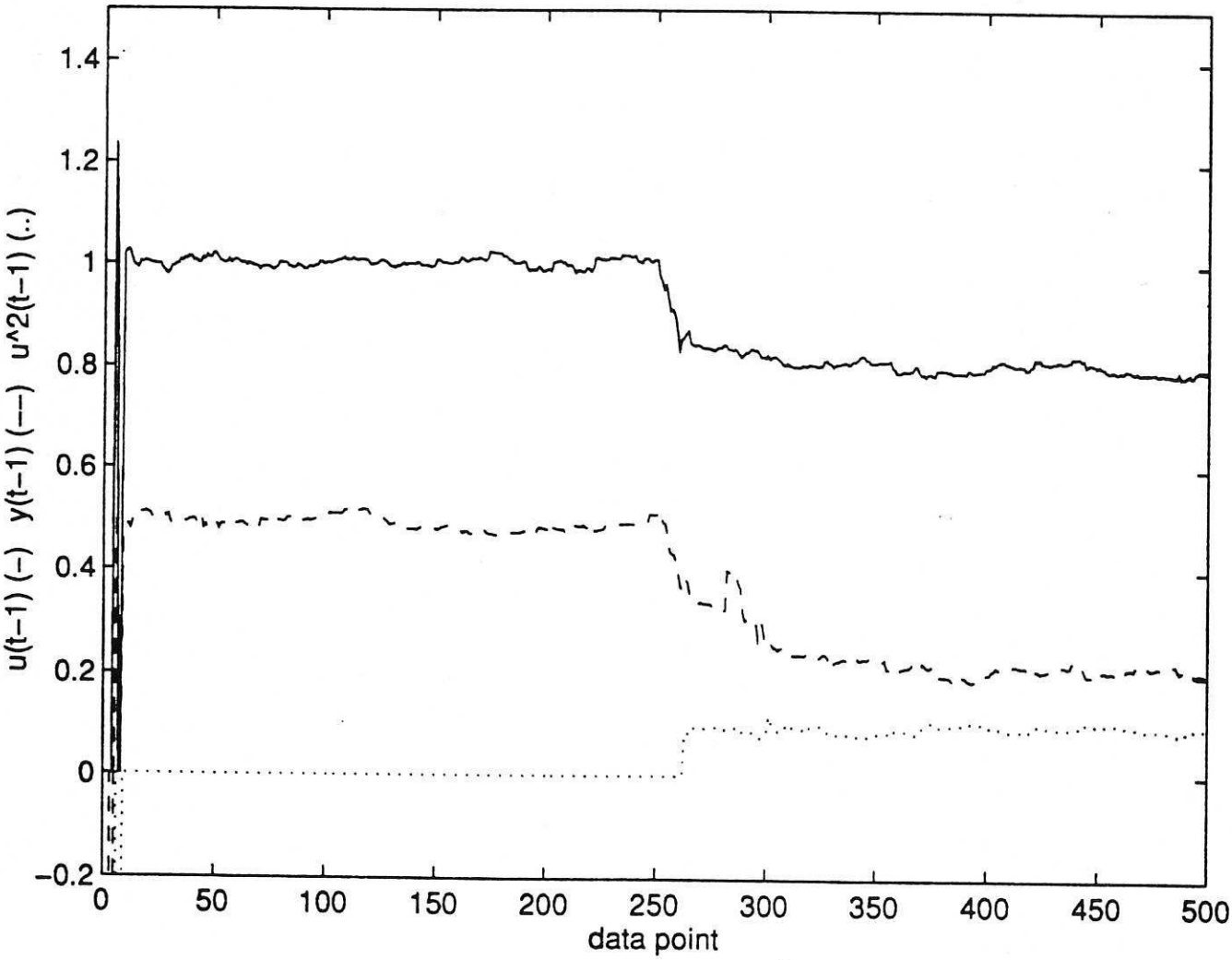


Fig.6 Predicted Output, Errors and No.of Regressors (Simulated System, SUA)

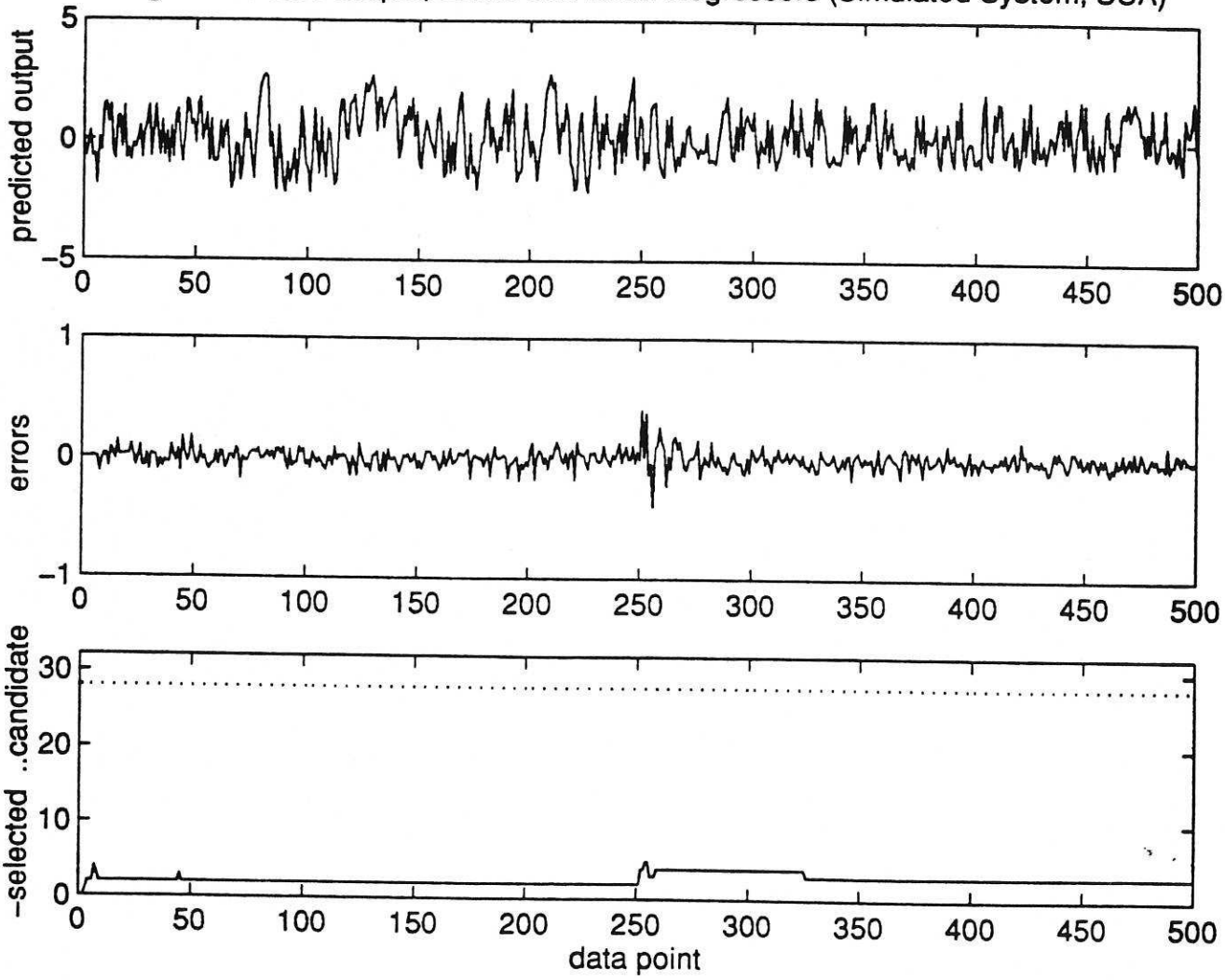


Fig.7 Estimates of Parameters (Simulated System, SUA)

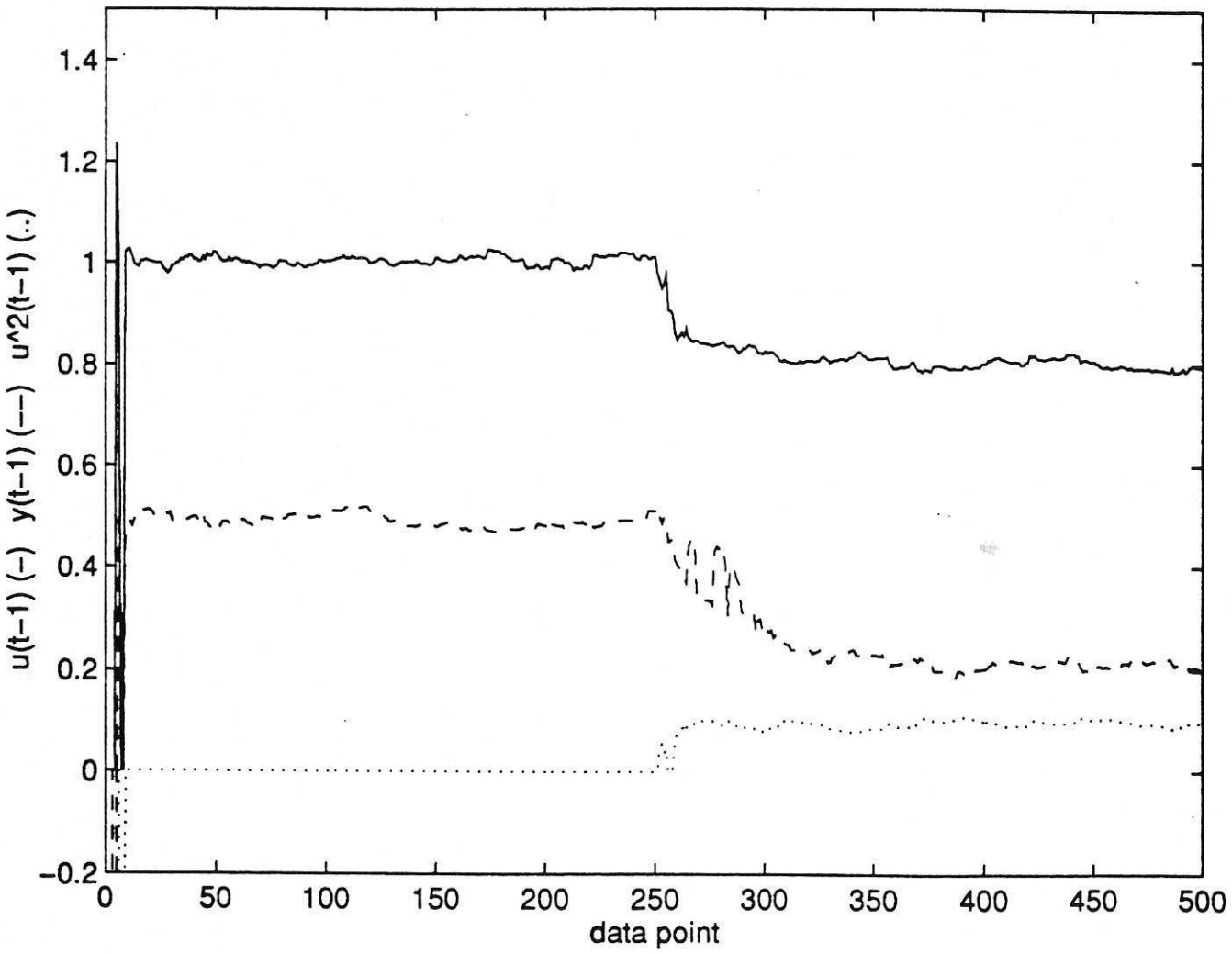


Fig.8 System Output and Input (Liquid Level System)

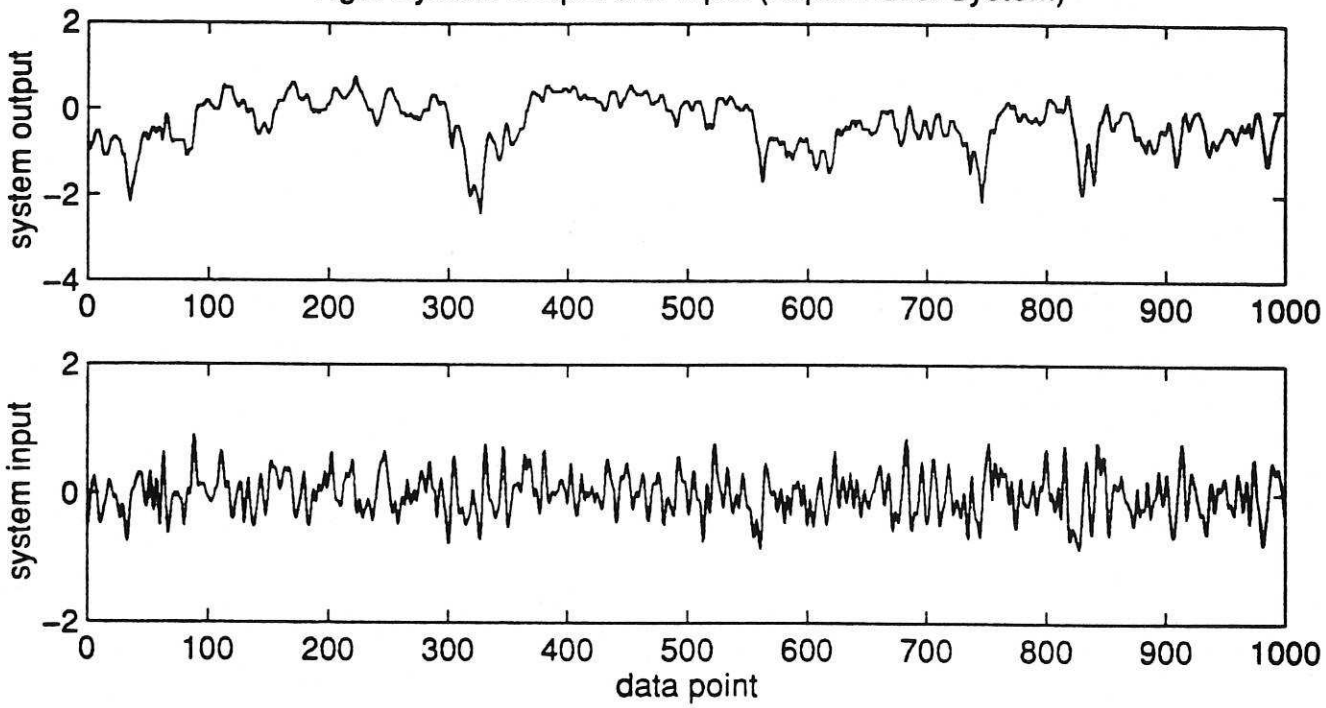


Fig.9 Predicted Output, Errors and No.of Regressors (Liquid Level System, PU)

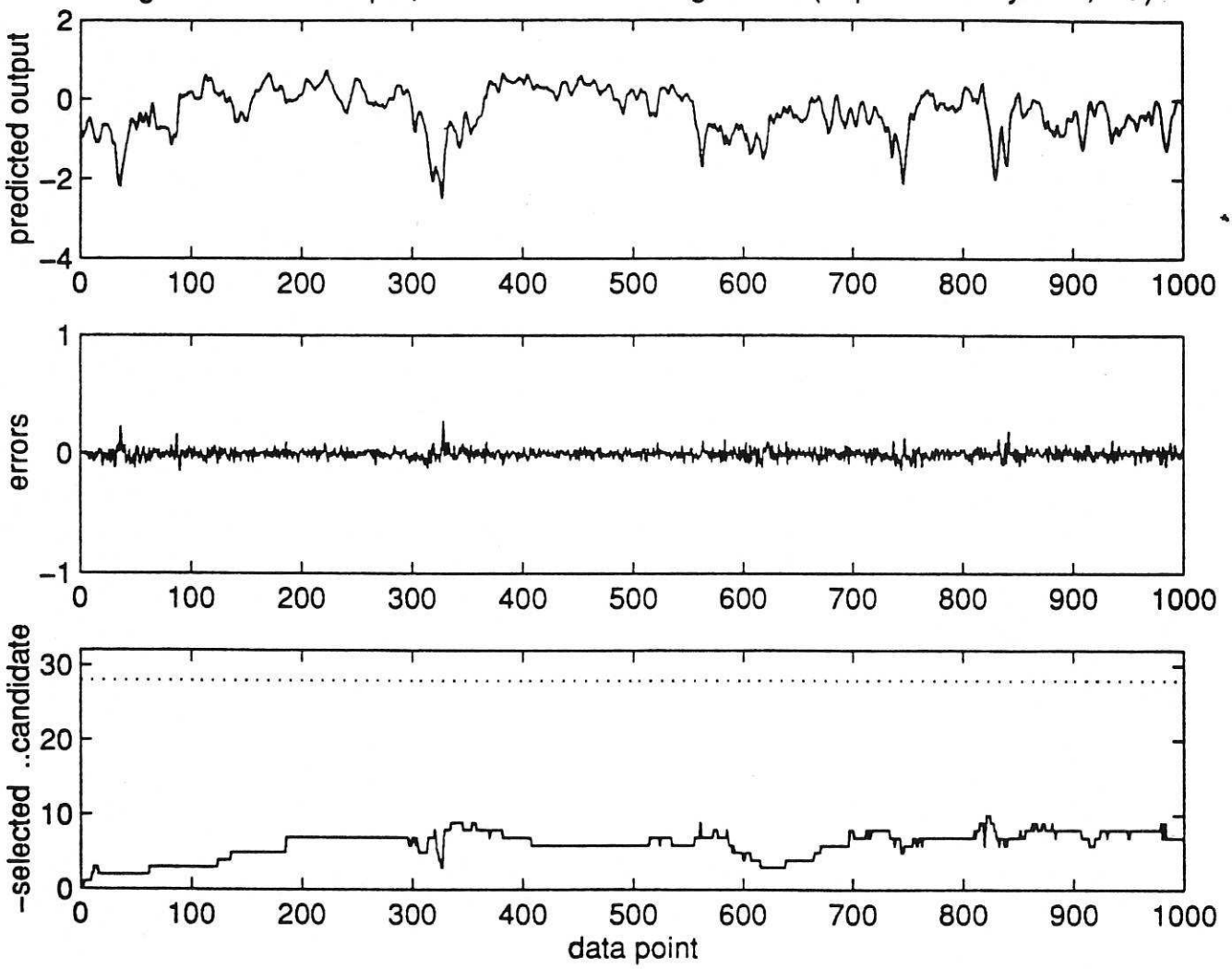


Fig.10 Predicted Output, Errors and No.of Regressors (Liquid Level System, SUT)

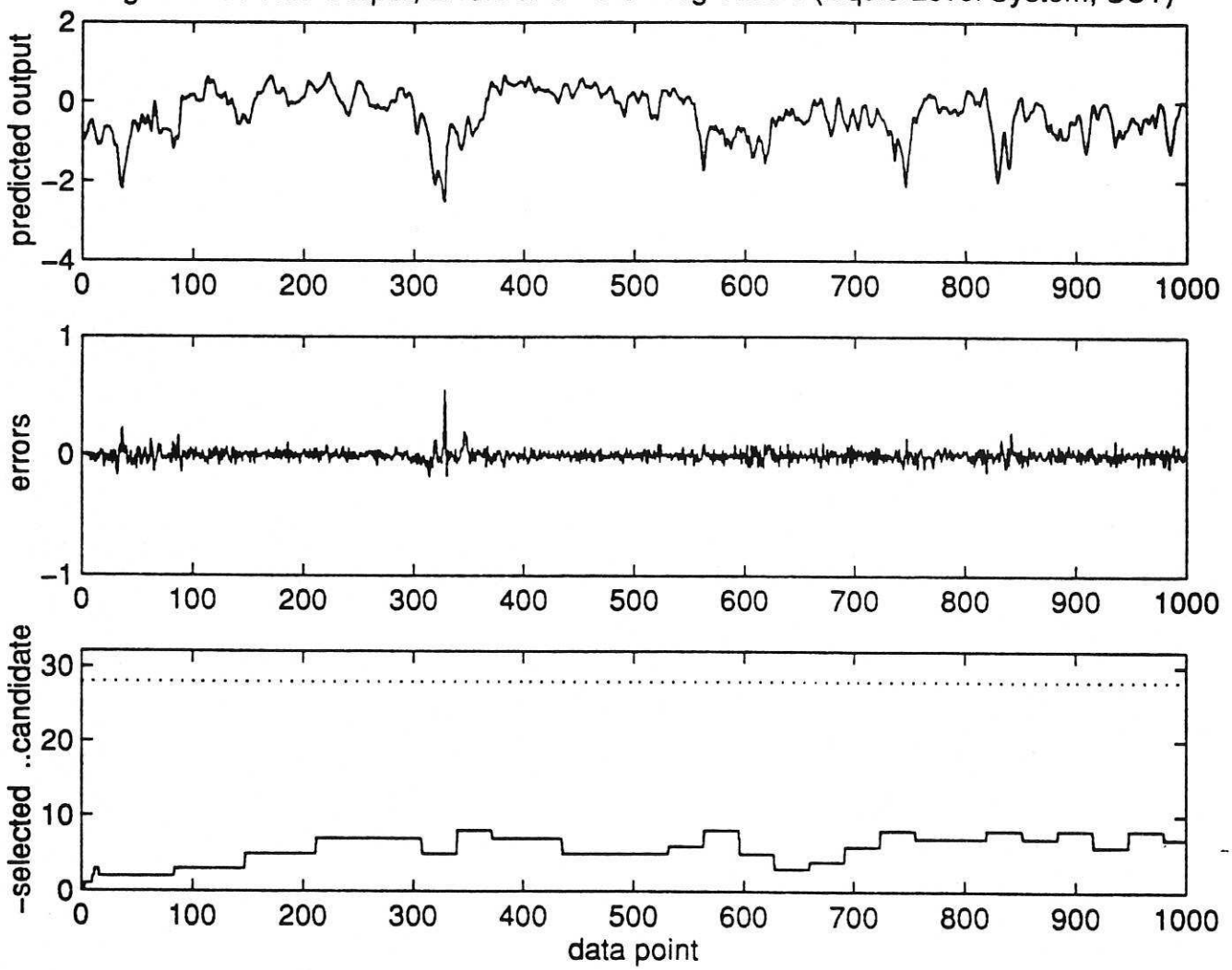


Fig.11 Predicted Output, Errors and No.of Regressos (Liquid Level System, SUA)

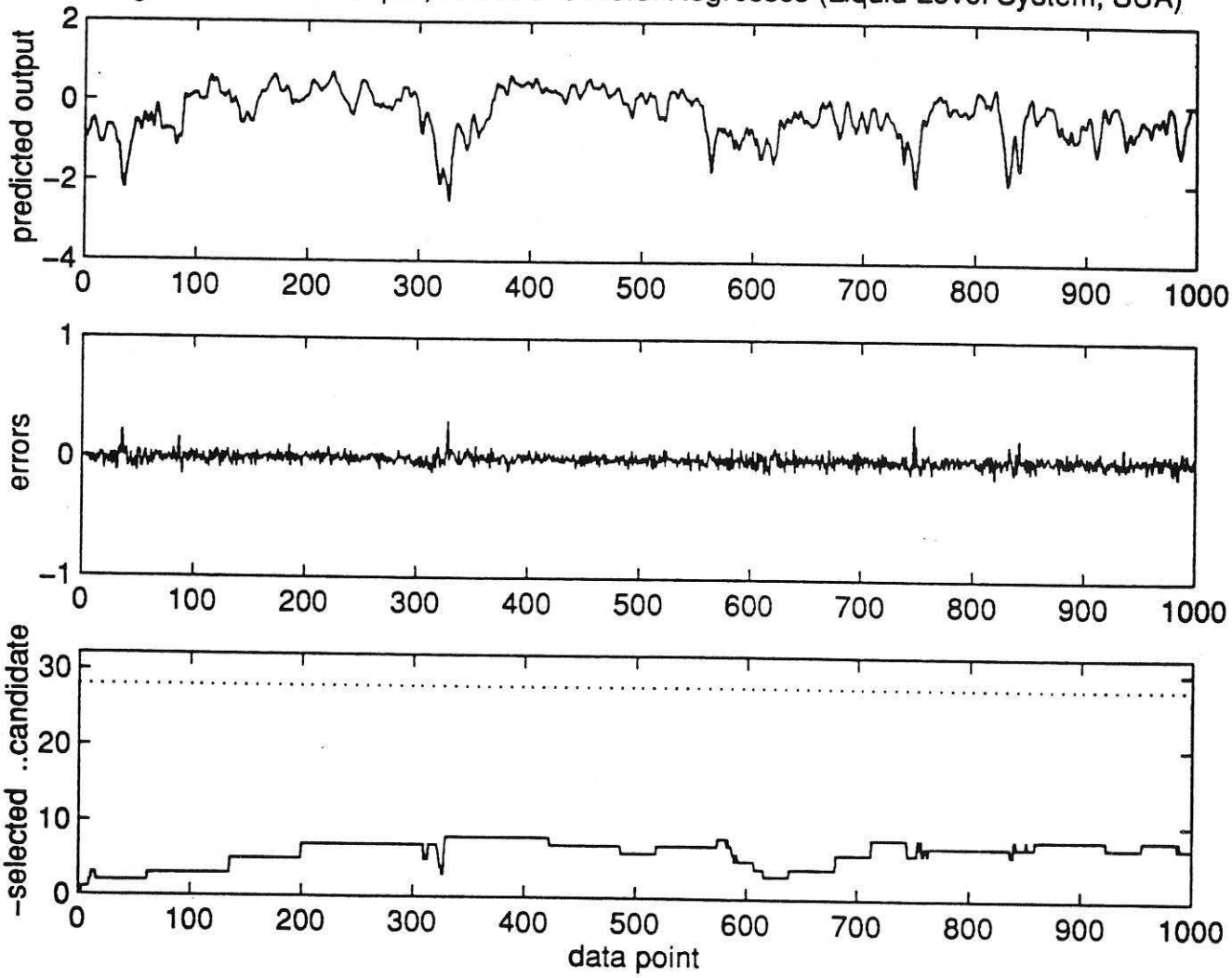


Fig.12 Predicted Output, Errors and No.of Regressors (Liquid Level System, with U(t))

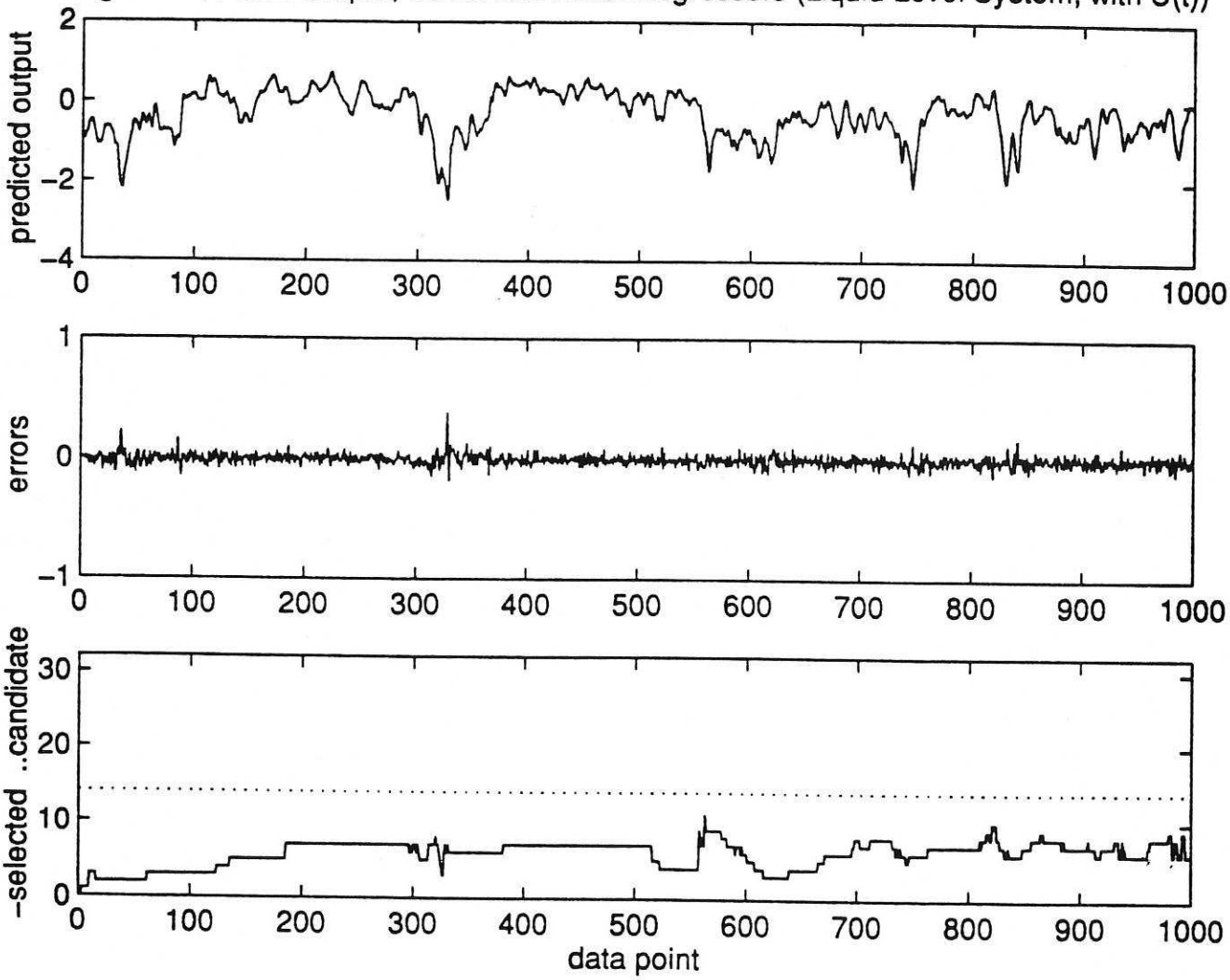


Fig.13 Predicted Output, Errors and No.of Regressos (Liquid Level System, with U^2(t))

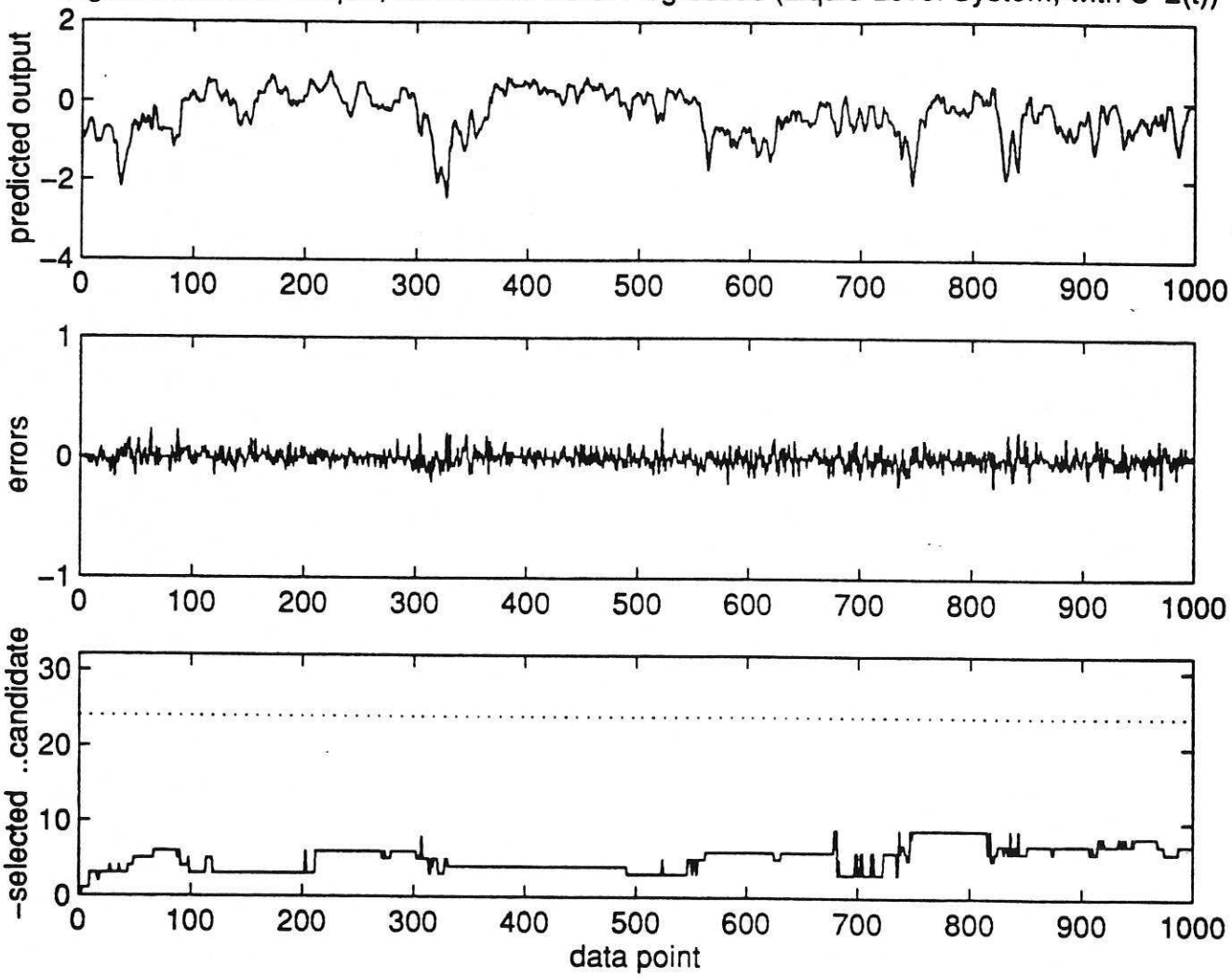


Fig.14 Prediction Errors and No.of Centres (Liquid Level System, RBFN, PU)

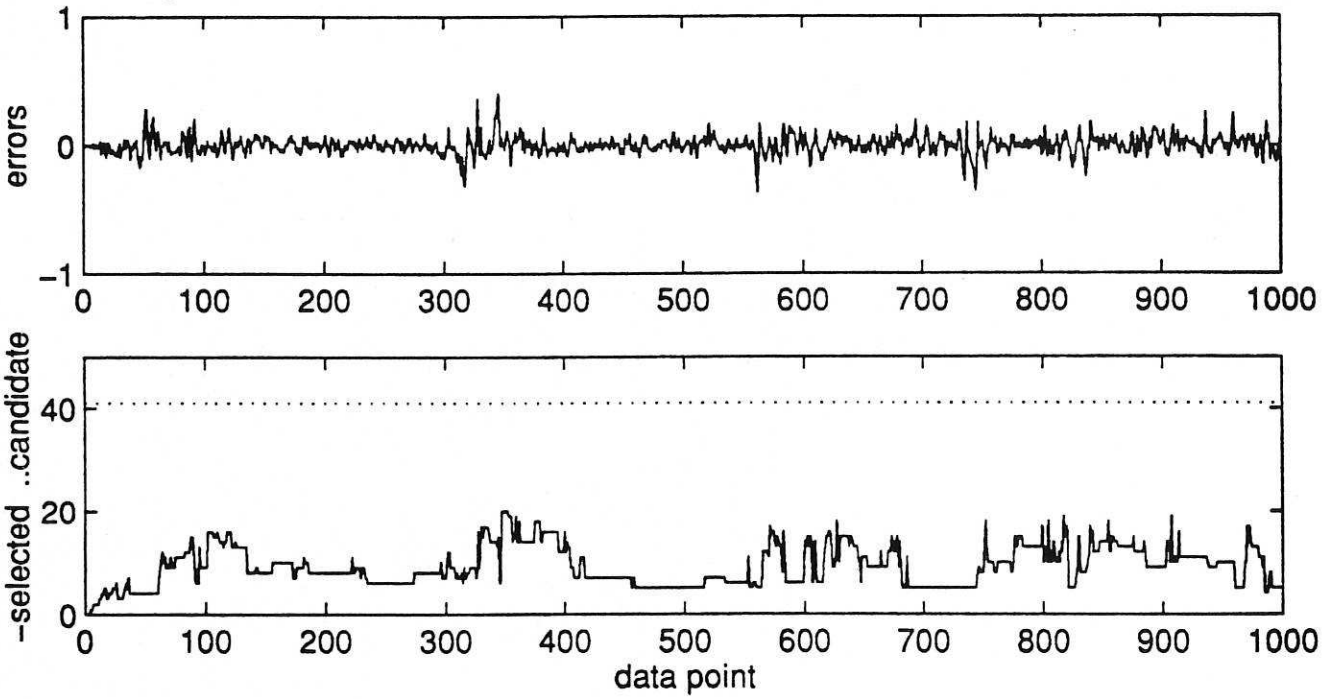


Fig.15 Prediction Errors and No.of Centres (Liquid Level System, RBFN, SUT)

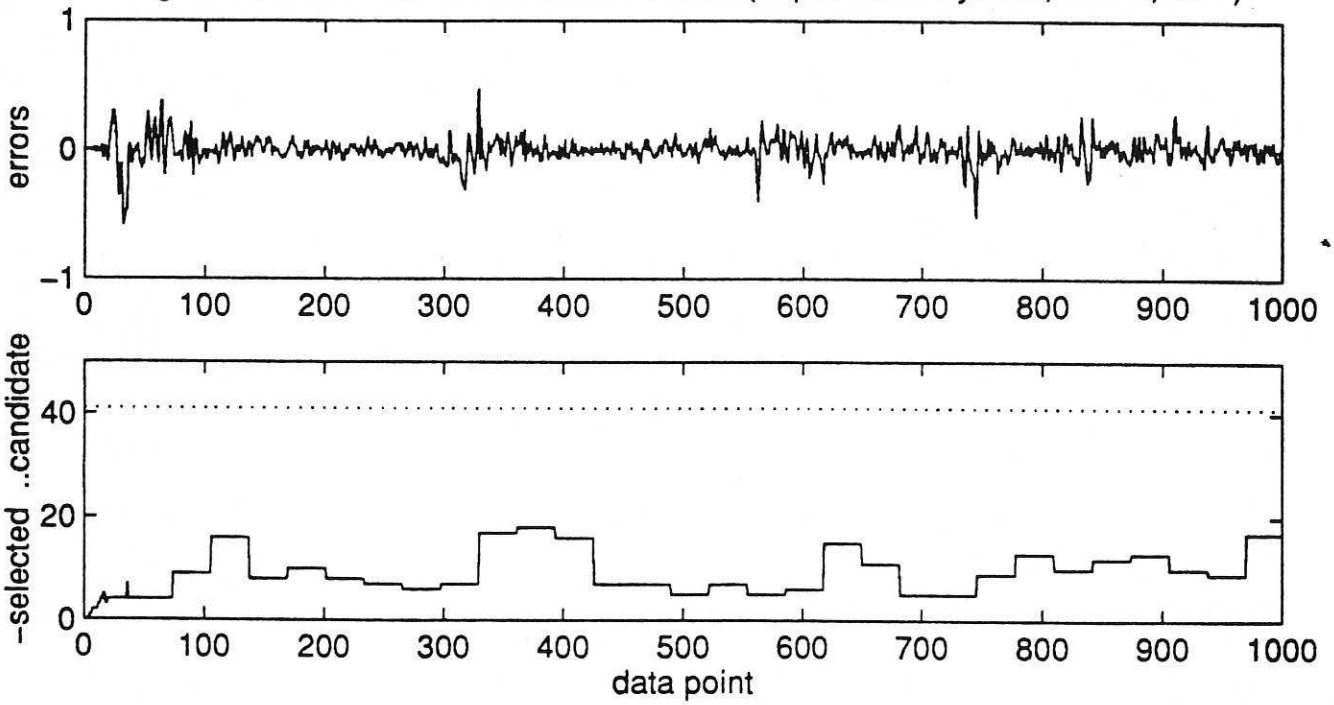


Fig.16 Prediction Errors and No.of Centres (Liquid Level System, RBFN, SUA)

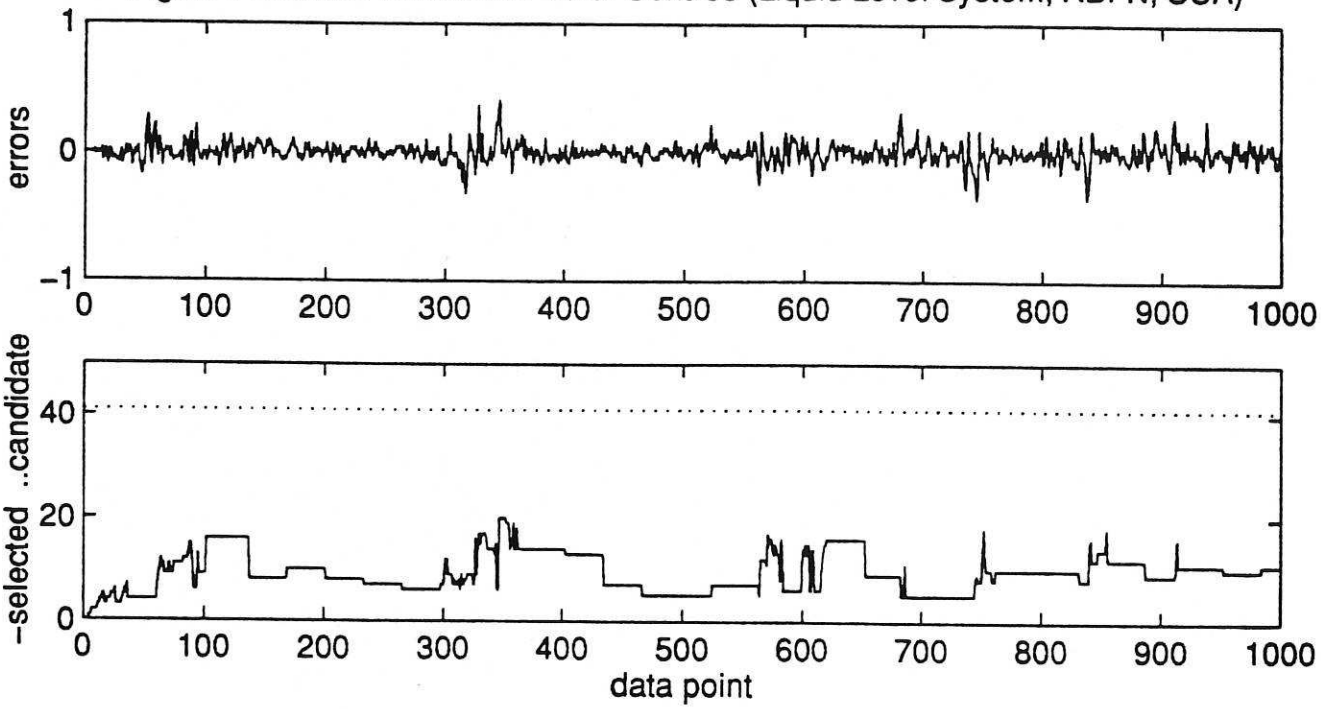


Fig.17 Prediction Errors and No.of Centres (Liquid Level System, RBFN, Control U(t)