Neural network architectures for parameter estimation of dynamical systems

Revised and uploaded

J.R. Raol
H. Madhuranath

Indexing terms: Neural networks, Parameter estimation, Parallel computing, Dynamical systems

Abstract: Various recurrent neural network architectures for solving the problems of parameter estimation in dynamical systems are presented. The architectures based on precomputation of weight/bias information (Hopfield neural network), direct gradient computation with and without normalisation and output error method are developed. A typical computer simulation result is given.

1 Introduction

Estimation of parameters based on the empirical data plays a crucial role in modelling and identification of dynamical systems, e.g. aerospace vehicles. Often equation error or output error methods are used for parameter estimation and very successful applications of output error method are reported [1–31]. These methods, in general, are batch-iterative, meaning that a set of data is processed to arrive at the gradient of a cost function and compute the estimation error. Then as a first step the parameter estimates are obtained using this information. Subsequently the estimates are refined via an iterative process based on the improved estimates of gradients and error (computed using the same data set and the previous estimates of the parameters).

Such an approach is not directly suitable for online real time application of the parameter estimation methods. There are several situations where online estimation might be required: (i) model-based approach to (aircraft) sensor failure detection, (ii) determination of lift and drag characteristics of an aircraft from its dynamic manoeuvres using a mathematical model.

Recently there has been a great deal of activity in the area of artificial neural networks and their possible applications to problems in science and engineering [4, 5]. Artificial neural networks (ANN) provide new/alternative solutions to the problems of signal processing and control, of which parameter estimation is a specialised branch. It has been shown that the so called recurrent neural networks (RNN) can be programmed to estimate the parameters of dynamical systems [6–11].

However, in order to obtain fast solutions to the parameter estimation problem, a system of parallel computers can be used. This will require the parallelisation of the conventional parameter estimation codes.

Since ANNs have massively parallel information processing capacity, they can be naturally adapted and utilised for solving parameter estimation problems [12]. In this paper some recurrent neural network architectures are presented which can process input/output data from dynamical systems and obtain estimates of the parameters very quickly, depending upon the speed of the basic processing units (as realised in hardware).

2 Parameter estimation problem

Consider the state space representation of a dynamical system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (1)$$

The aim is to solve for $\mathbf{A}$, $\mathbf{B}$ the system parameter matrices knowing $\mathbf{x}$, $\dot{\mathbf{x}}$ and u . Here $\mathbf{x}$ is $n \times 1$ state vector, $u = u(t)$ is the control input (scalar), $\mathbf{A} = [a_{ij}]$ is $n \times n$ matrix and $\mathbf{B} = [b_i]$ is $n \times 1$ vector.

The parameters a_{ij} (elements of $\mathbf{A}$) and b_i (elements of $\mathbf{B}$) to be estimated are denoted by $n_p \times 1$

$$\mathbf{v} = [a_{11}, a_{12}, \dots, a_{1n}, a_{21}, a_{22}, \dots, a_{2n}, \dots, a_{n1}, a_{n2}, \dots, a_{nn}, b_1, b_2, \dots, b_n] \quad (2)$$

where n_p is the number of parameters to be estimated i.e. $(n^2 + n)$. Thus

$$v_s = a_{ij} \quad \text{for } s = (i-1)n + j \quad (3)$$

and

$$v_s = b_i \quad \text{for } s = n^2 + i \quad (4)$$

Conversely, if $1 \leq s \leq n^2$ then

$$v_s = a_{ij} \quad (5)$$

where $i = (s-1) \bmod(n)j = s - (i-1)n$. If $n^2 + 1 \leq s \leq n^2 + n$ then

$$v_s = b_i \quad \text{where } i = s - n^2 \quad (6)$$

The equation error vector $\mathbf{e}(\mathbf{v})$ is defined as

$$\mathbf{e}(\mathbf{v}) = \dot{\mathbf{x}} - \mathbf{A}\mathbf{x} - \mathbf{B}u \quad (7)$$

We define a cost function $E(\mathbf{v})$ as the sum of squares of the error over all m data points

$$E(\mathbf{v}) = \frac{1}{2} \sum_{k=1}^m \mathbf{e}^T(k) \mathbf{e}(k) \quad (8)$$

Using the result from the optimisation theory [12], one obtains

$$\frac{d\mathbf{v}}{dt} = -\mu(t) \nabla E(\mathbf{v}) \quad (9)$$

Here $\nabla E(v)$ denotes the gradient and $\mu = [\mu_{ij}]$ is a $n_p \times n_p$ matrix. Often $\mu(t)$ is diagonal

$$\begin{aligned} \frac{dv_s}{dt} &= -\mu_s \frac{\partial E}{\partial v_s} \quad 1 \leq s \leq n_p \\ v_s &= \mu_s \int -\frac{\partial E}{\partial v_s} dt \end{aligned} \quad (10)$$

Here μ_s can be replaced by or absorbed in a nonlinear function f_s (e.g. $[1 - e^{-\lambda x_i}]/[1 + e^{-\lambda x_i}]$) so that

$$\begin{aligned} v_s &= f_s \left(\int -\frac{\partial E}{\partial v_s} dt \right) \\ v_s &= f_s(\beta_s) \end{aligned} \quad (11)$$

where β_s is defined as

$$\beta_s = - \frac{\partial E}{\partial v_s} dt$$

and

$$\frac{d\beta_s}{dt} = - \frac{\partial E}{\partial v_s} \quad (12)$$

The algorithm for digital computer implementation based on eqns. 11 and 12 is given below

(i) start with initial values of $\beta: \beta_s(0)$

(ii) use eqn. 11 to compute v_s

(iii) compute $\partial E / \partial v_s$

(iv) integrate eqn. 12 to compute β

(v) go to (ii) until convergence is reached.

An alternative procedure is derived as follows. Differentiating eqn. 11 with respect to t gives

$$\begin{aligned} \frac{dv_s}{dt} &= f'_s(\beta_s) \frac{d\beta_s}{dt} \\ \frac{d\beta_s}{dt} &= \frac{1}{f'_s(\beta_s)} \frac{dv_s}{dt} \end{aligned} \quad (13)$$

Using eqn. 10, we obtain

$$\frac{d\beta_s}{dt} = - \frac{\mu_s}{f'_s(\beta_s)} \frac{\partial E}{\partial v_s} \quad (14)$$

The digital computer algorithm for this method is similar to the previous one except that in step (iv), eqn. 14 is used instead of eqn. 12.

Thus the parameter estimation rules given by eqns. 12 and 14 are generalisations of eqn. 10, because of the use of the nonlinear function (e.g. sigmoidal nonlinearity) which can improve the quality of the parameter estimates in the presence of outliers.

3 RNN architecture based on precomputation of weight bias information

We assume that measurements of $\dot{x}(t)$ and $x(t)$ are available [6, 11]. Let $\int f$ denote $\int (1/t_f) f(t) dt$ where t_f is the total time. The cost function is given as

$$\begin{aligned} E(v) &= \frac{1}{2} \int e^T(v) e(v) \\ &= \frac{1}{2} \int (\dot{x} - Ax - Bu)^T (\dot{x} - Ax - Bu) \end{aligned} \quad (15)$$

$$\begin{aligned} \frac{\partial E}{\partial A} &= \int (\dot{x} - Ax - Bu)(-\dot{x}^T) \\ &= A \int x x^T + B \int u x^T - \int \dot{x} x^T \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial a_{ij}} &= \left[\sum_{p=1}^n a_{ip} \int x_p x_j \right] + b_i \int u x_j - \int \dot{x}_i x_j \\ \frac{\partial E}{\partial B} &= \int (\dot{x} - Ax - Bu)(-u) \\ &= v u + B \int u^2 - \int \dot{x} u \end{aligned}$$

$$\frac{\partial E}{\partial a_i} = \left[\sum_{p=1}^n a_{ip} \int x_p u \right] + b_i \int u^2 - \int \dot{x}_i u \quad (17)$$

Eqns. 16 and 17 can be written in terms of v using eqns. 2-12 to give, for $1 \leq s \leq n^2$

$$v_{(i-1)n+p}(-\int x_p x_j) + v_{n^2+i}(-\int u x_j) + \int \dot{x}_i x_j$$

where i and j are determined from eqn. 5

For $n^2 + 1 \leq s \leq n_p$

$$\frac{d\beta_s}{dt} = \left[\sum_{p=1}^n v_{(i-1)n+p}(-\int x_p u) \right] + v_{n^2+i}(-\int u^2) + \int \dot{x}_i u \quad (19)$$

where i is determined from eqn. 6.

Eqns. 18 and 19 can be written as

$$\frac{d\beta_s}{dt} = \sum_{r=1}^{n_p} W_{sr} v_r + I_s \quad (20)$$

where $W = [W_{sr}]$ is a $n_p \times n_p$ weight matrix and $I = [I_s]$ is a $n_p \times 1$ bias/input vector. The weight matrix W is defined, for $1 \leq s \leq n^2$, as

$$\begin{aligned} W_{sr} &= -\int x_p x_j \quad 1 \leq p \leq n; \quad r = (i-1)n + p \\ &= -\int u x_j \quad r = n^2 + i \\ &= 0 \quad \text{for all other } r \quad 1 \leq r \leq n_p \end{aligned} \quad (21)$$

for $n^2 + 1 \leq s \leq n_p$, as

$$\begin{aligned} W_{sr} &= -\int x_p u \quad 1 \leq p \leq n \quad r = (i-1)n + p \\ &= -\int u^2 \quad r = n^2 + i \\ &= 0 \quad \text{for all other } r \quad 1 \leq r \leq n_p \end{aligned} \quad (22)$$

The bias vector I is defined, for $1 \leq s \leq n^2$, as

$$I_s = \int \dot{x}_i x_j \quad (23)$$

for $n^2 + 1 \leq s \leq n_p$, as

$$I_s = \int \dot{x}_i u \quad (24)$$

Eqn. 20 can be written in vector notation as

$$\frac{d\beta}{dt} = W v + I \quad (25)$$

The general schematic diagram of the RNN architecture based on eqns. 11 and 25 is shown in Fig. 1. The W block generates the elements of the weight matrix and the I block generates the elements of the I vector. The matrix multiplication and addition in Fig. 1 are carried out for each parameter β_s ; eqn. 20, by the M

block shown in Fig. 2. Since the elements of the W matrix can be precomputed, it is sufficient to use constant multipliers instead of expensive analogue multipliers. In general each M block carries out matrix multiplication and addition to give one element of the matrix product in all the diagrams where it is used. In Fig. 2, the input variables to the M block have appropriate meaning.

Fig.1 Schematic block diagram of neural network architecture for weight/bias method

Fig.2 Definition of the M block
... 1 to n

Fig.3 Schematic diagram of W block
... 1 to n
... 1 to n blocks

The detailed schematic diagram of the W block is shown in Fig. 3. From the definition of the W matrix, eqns. 21 and 22, the total number of non zero distinct elements of the W matrix is computed as shown in Table 1. Thus the number of elements of the W matrix to be generated is of the order of the dimension of the

matrix rather than the order of its size. This reduces the size of the W block considerably. The I block is shown in Fig. 4. The complete RNN architecture for this method constituted by the blocks from Figs. 2 to 4 is shown in Fig. 5.

Fig.4 Schematic diagram of I block
... 1 to n
... 1 to n blocks

Fig.5 Schematic architecture of recurrent neural network for parameter estimation (weight/bias method)
... 1 to n
... 1 to n blocks

Table 1: Count of hardware elements

	Constant multipliers	Summers	Integrators	Multipliers
Weight-based neural network architecture				
M block (Figs. 2 and 5)	$\cdot (n^2 + n)(n + 1)$	$(n^2 + n)$	—	—
W block (Fig. 3)				
$\{x_i x_j\}$	—	—	$n(n + 1)/2$	$n(n + 1)/2$
$\{u x_j\}$	—	—	n	n
$\{u^2\}$	—	—	1	1
Total	—	—	$(n + 1)(n + 2)/2$	$(n + 1)(n + 2)/2$
I block (Fig. 4)				
$\{\dot{x}_i x_j\}$	—	—	n^2	n^2
$\{\dot{x}_i u\}$	—	—	n	n
Total	—	—	$(n^2 + n)$	$(n^2 + n)$
Parameter integrators	—	—	$(n^2 + n)$	—
Total	$n(n + 1)^2$	$n(n + 1)$	$(n + 1)(5n + 2)/2$	$(n + 1)(3n + 2)/2$
Gradient-based neural network architecture				
Error block				
eqn. 29	$mn(n + 1)$	nm	—	—
Gradient block				
eqn. 30	$m(n^2)$	n^2	—	—
eqn. 31	mn	n	—	—
Total	$mn(n + 1)$	$n(n + 1)$	—	—
Parameter integrators	—	—	$(n^2 + n)$	—
Total	$2mn(n + 1)$	$n(m + n + 1)$	$n(n + 1)$	—
Gradient-based neural network architecture with normalisation				
Gradient-based neural network	$2mn(n + 1)$	$n(m + n + 1)$	$n(n + 1)$	—
For normalisation*				
eqn. 40	—	—	—	$n(n + 1)$
Total	$2mn(n + 1)$	$n(m + n + 1) + 1$	$n(n + 1)$	$n(n + 1)$

*this requires $n(n + 1)$ dividers also.

4 RNN architecture based on direct gradient computation

We assume that $\dot{x}$ and x vectors are known at discrete instants of time $k = 1, 2, \dots, m$. From eqns. 7 and 8, the equation error cost function is given by

$$E(v) = \frac{1}{2} \sum_{k=1}^m (\dot{x}(k) - Ax(k) - Bu(k))^T (\dot{x}(k) - Ax(k) - Bu(k)) \quad (26)$$

$$\frac{\partial E}{\partial A} = \sum_{k=1}^m (\dot{x}(k) - Ax(k) - Bu(k))(-x(k))^T$$

$$\begin{aligned} \frac{\partial E}{\partial B} &= \sum_{k=1}^m (\dot{x}(k) - Ax(k) - Bu(k))(-u(k)) \\ &= \sum_{k=1}^m e(k)(-u(k))^T \end{aligned} \quad (28)$$

The eqns. 7, 27, 28 and 9 can be written in the parametric form:

$$e_i(k) = \dot{x}_i(k) - \sum_{p=1} a_{ip} x_p(k) - b_i u(k) \quad (29)$$

$$\frac{\partial E}{\partial a_{ij}} = \sum_{k=1}^m (-e_i(k))(x_j(k)) \quad (30)$$

$$\frac{\partial E}{\partial b_i} = \sum_{k=1}^m (-e_i(k))u(k) \quad (31)$$

$$\frac{dv_s}{dt} = - \sum_{r=1}^{(n^2+n)} \mu_{sr} \frac{\partial E}{\partial v_r} \quad (32)$$

The general block diagram based on the above equations is shown in Fig. 6. The error block generates the error vector, the gradient block generates the gradient of the energy function while the adaptive block causes the parameters to change according to the learning rule of eqn. 9. The nonlinear activation function in Fig. 6 is introduced in order to limit the effect of the outliers on the parameter estimates.

Fig.6 Schematic block diagram of neural network architecture for gradient-based method

Fig. 7 Schematic neural network-architecture for gradient-based method
 1 to m
 ... 1 to n
 ... 1 to n blocks
 1 to n_p

The RNN architecture as shown in Fig. 7 can be implemented using **M** blocks. Each **M** block carries out the operations in eqns. 29–32 for a parameter. The nonlinear function is not shown explicitly in Fig. 7. The detailed RNN architecture is shown in Fig. 8, where the μ matrix is assumed to be diagonal for simplicity.

Fig. 8 Neural network architecture for gradient-based method
 μ is a diagonal matrix
 1 to m
 ... 1 to n
 ... 1 to n blocks

5 RNN architecture based on direct gradient computation with normalisation

The μ matrix was constant in the gradient based method. Theoretically, it takes infinite time for v to converge to the true solution v^* . In some real time

applications, it may be necessary that the solution be obtained within a prescribed period of time t_r i.e. $v(t_r) = v^*$, i.e. $E(v)$ should reach a minimum with a specified error δ at $t = t_r$ where δ is a very small positive number and it should remain in the δ neighbourhood for $t > t_r$ [12]. This can be achieved by making μ matrix adaptive during the minimisation process.

Let $\mu_0(t)$ denote the adaptive parameter in eqn. 9, defined as

$$\mu_0(t) = \mu_{0 \max}$$

when $E(v(t)) = 0$, otherwise

$$\mu_0(t) = \frac{\mu}{tr \left\{ \left(\sum_{k=1}^m e(k) x^T(k) \right) \left(\sum_{k=1}^m x(k) e^T(k) \right) + \left(\sum_{k=1}^m e(k) u(k) \right) \left(\sum_{k=1}^m u(k) e^T(k) \right) \right\}} \quad (33)$$

where $\mu = 1/\tau > 0$ (τ is the time constant of integration).

The time derivative of the cost function $E(v(t))$ is

$$\begin{aligned} \frac{dE}{dt} &= \sum_{s=1}^{n_p} \frac{\partial E}{\partial v_s} \frac{dv_s}{dt} \\ &= [\nabla E(v)]^T \frac{dv}{dt} \end{aligned} \quad (34)$$

If $dv/dt \neq 0$ using eqn. 9, we obtain

$$\begin{aligned} \frac{dE}{dt} &= -\mu_0(t) \sum_{s=1}^{n_p} \left(\frac{\partial E}{\partial v_s} \right)^2 \\ &= -\mu_0(t) \left\{ \sum_{j=1}^n \sum_{i=1}^n \left(\frac{\partial E}{\partial a_{ij}} \right)^2 + \sum_{i=1}^n \left(\frac{\partial E}{\partial b_i} \right)^2 \right\} \\ &= -\mu_0(t) \left\{ tr \left(\frac{\partial E}{\partial A} \right) \left(\frac{\partial E}{\partial A} \right)^T + tr \left(\frac{\partial E}{\partial B} \right) \left(\frac{\partial E}{\partial B} \right)^T \right\} \end{aligned} \quad (35)$$

Fig.9 Neural network architecture for gradient-based method with normalisation
 1 to m
 ... 1 to n
 ... 1 to n blocks

Using eqns. 27 and 28

$$\begin{aligned} \frac{dE}{dt} &= -\mu_0(t) \\ &\times \text{tr} \left\{ \left(\sum_{k=1}^m e(k)(-x^T(k)) \right) \left(\sum_{k=1}^m e(k)(-x^T(k)) \right)^T \right. \\ &\quad \left. + \left(\sum_{k=1}^m e(k)(-u^T(k)) \right) \left(\sum_{k=1}^m e(k)(-u^T(k)) \right)^T \right\} \\ &= -\mu_0(t) \times \text{tr} \left\{ \left(\sum_{k=1}^m e(k)x^T(k) \right) \left(\sum_{k=1}^m x(k)e^T(k) \right) \right. \\ &\quad \left. + \left(\sum_{k=1}^m e(k)u(k) \right) \left(\sum_{k=1}^m u(k)e^T(k) \right) \right\} \\ &= \mu < 0 \end{aligned} \quad (36)$$

If $dv/dt = 0$, eqn. 34 becomes $dE/dt = 0$.

Thus the cost function decreases linearly with time during the minimisation process. Eqn. 36 can be written in integral form as

$$E(v(t)) = E(v(0)) - \mu t \quad (37)$$

It is assumed that the initial value $E(v(0))$ and the final value $E(v^*)$ are known. Let us assume that at the start $v(0) = 0$ i.e. $A = 0, B = 0$, then using eqn. 26, the value of $E(v(0))$ can be computed as

$$\begin{aligned} E(v(0)) &= \frac{1}{2} \sum_{k=1}^m (\dot{x}^T(k))(\dot{x}(k)) \\ &= \frac{1}{2} \sum_{k=1}^m \sum_{i=1}^n (\dot{x}_i(k))^2 \end{aligned} \quad (38)$$

Assuming that $E(v^*) = 0$, from eqn. 37 we obtain

$$t_r = \frac{E(v(0)) - 0}{\mu} \quad (39)$$

Thus μ can be determined from eqn. 39 for the prescribed settling time t_r .

The implementation of $\mu_0(t)$ in the adaptive block of Fig. 8 can be easily done from the alternative expression for $\mu_0(t)$ derived from eqns. 35 and 36,

$$\mu_0(t) = \frac{\mu}{\sum_{s=1}^{n_p} \left(\frac{\partial E}{\partial v_s} \right)^2} \quad (40)$$

The detailed RNN architecture with normalisation is shown in Fig. 9. The disadvantage of this implementation is that it uses analogue multipliers and dividers which are expensive.

6 RNN architecture for output error method

The more general problem is to estimate the parameters of the matrices A, B, C and D in the equations

$$\dot{x} = Ax + Bu \quad (41)$$

$$y = Cx + Du \quad (42)$$

$$z(k) = y(k) + v(k) \quad k = 1, \dots, m$$

where the variables have appropriate dimension. The parameter vector v consists of the elements of A, B, C and D . The output error cost function E is

$$E(v) = \frac{1}{2} \sum_{k=1}^m (z(k) - y(k))^T (z(k) - y(k)) \quad k = 1, \dots, m \quad (43)$$

From eqn. 9, we obtain

$$\frac{dv}{dt} = -\mu \sum_{k=1}^m \left(\frac{\partial y}{\partial v}(k) \right)^T (z(k) - y(k)) \quad (44)$$

The convergence of v is speeded up using the adaptive μ matrix, defined as

$$\mu = \mu_0 \left\{ \sum_{k=1}^m \left(\frac{\partial y}{\partial v}(k) \right)^T \left(\frac{\partial y}{\partial v}(k) \right) \right\}^{-1} \quad (45)$$

Using eqn. 45 in eqn. 44, we obtain

$$\frac{dv}{dt} = -\mu_0 \left\{ \sum_{k=1}^m \left(\frac{\partial y}{\partial v}(k) \right)^T \left(\frac{\partial y}{\partial v}(k) \right) \right\}^{-1} \left\{ \sum_{k=1}^m \left(\frac{\partial y}{\partial v}(k) \right)^T (z(k) - y(k)) \right\} \quad (46)$$

The block schematic RNN architecture for the output error method constructed using **M** blocks (of Fig. 2) is shown in Fig. 10. The parameter vector v is updated after every m samples of the data using discrete integration. Thus the well known parameter estimation rule (eqn. 46) based on the maximum likelihood output error method can be naturally adapted to parallelisation using RNN architecture. The gross estimate of hardware computational elements required for some of the architectures is given in Table 1. With VLSI (very large scale integration) technology, it might be feasible to fabricate microelectronics networks of high complexity [12, 13] for solving parameter estimation problems for dynamical systems for which recurrent neural network architectures have been formulated in this paper.

7 Numerical simulation result

Some of the schemes for which RNN architectures (for realisation into hardware circuits) are presented in this paper, have been studied by computer simulations for stable and unstable/augmented dynamical systems [7, 8, 10, 14].

Consider a second order state-space model of a system

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 + b_1u \quad (47)$$

$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2 + b_2u \quad (48)$$

Feedback variable x_1 was utilised to close the loop. For parameter estimation, the signals u , x , and $\dot{x}$ were used in digital computer simulation of an RNN architecture based on direct gradient computation with normalisation. The results are compared with the equation error method (EE) as shown in Table 2. Based on the param-

eter estimation error norm (PEEN) it can be seen that the performance of the RNN scheme is better than that of EE. Also, reasonably accurate estimates have been obtained. Based on many other simulation exercises, it can be inferred that the RNN-based schemes either give similar or better results depending upon the effect of noise or outliers on the data [7, 8, 10, 14].

Table 2 Parameter estimation results

Parameter	True value	Equation error method	Recurrent neural network
a_{11}	-1.425	-1.461	-1.607
	-1.477	-1.275	-1.402
a_{21}	0.216	0.143	0.157
a_{22}	-3.707	-3.295	-3.619
b_1	-6.263	-5.839	-6.387
b_2	-12.780	-11.917	-12.817
PEEN%		7.8	2.16

PEEN: parameter estimation (relative)error (L2) norm
signal/noise ratio = 10

The main theme of this paper has been to present RNN architectures which can be exploited to build a parallel computer to solve parameter estimation (PE) problems of dynamical systems for fast throughput and online/real time applications. In addition, many of the conventional PE methods can be shown to be the special cases of the RNN-based schemes presented in this paper. Some of the merits of the RNN schemes come from the use of a nonlinear activation function which tries to reduce the adverse effect of outliers/spikes in the data and improve the convergence of the algorithms.

8 Conclusions

Recurrent neural network architectures for estimation of parameters of state-space models of dynamical systems have been described. Specifically, the neuron-like schematic architectures based on weight/bias information, the direct gradient method and the output error method have been developed and presented in this paper. A typical computer simulation result has been given to compare the performance of one of the proposed schemes with a conventional method. These architectures have the appealing features of a parallel computer and can be implemented with analogue adaptive circuits/VLSI technology

Fig. 10 Schematic architecture of neural network for output error method

... 1 to m
... 1 to n
... 1 to n blocks
... 1 to n_p

9 Acknowledgments

The authors are grateful to Dr. S. Srinathkumar, Head of Flight Mechanics & Control Division and Deputy Director of National Aerospace Laboratories (NAL) for his encouragement during the course of this work. H. Madhuranath is grateful to the director of NAL for his permission to pursue this work at NAL.

10 References

- 1 MENDEL, J.M.: 'Discrete techniques for parameter estimation — the equation error formulation' (Marcel Dekkar, 1973)
- 2 MAINE, R.E., and ILIFF, K.W.: 'Application of parameter estimation to aircraft stability and control — the output error approach'. NASA report 1168, 1986
- 3 GIRIJA, G., PARAMESWARAN, V., RAOL, J.R., and BALAKRISHNA, S.: 'Estimation of aircraft performance and stability characteristics by dynamic manoeuvres using maximum likelihood estimation technique'. Proceedings of 9th IFAC/IFORS symposium on *System identification and parameter estimation*, Budapest, Hungary, 8–12 July 1991
- 4 WARWICK, K., IRWIN, G.W., and HUNT, K.J. (Eds.): 'Neural networks for control and systems' (IEE, 1992)
- 5 HAYKIN, S.: 'Neural networks — a comprehensive foundation' (IEEE, 1994)
- 6 CHU, S.R., RAHAMAT, S., and TENORIO, M.: 'Neural networks for system identification', *IEEE Control Systems Mag.*, 1990, 10, (4), pp. 31–35
- 7 RAOL, J.R., RAMESH, S.V., and KAVITHA, B.: 'Neural network schemes for parameter estimation of state space models'. Proceedings of 3rd IEEE symposium on *Intelligent systems*, 22–23 Bangalore, India, November 1993
- 8 RAOL, J.R.: 'Parameter estimation in dynamic systems using neural networks'. Proceedings of 4th IEEE symposium on *Intelligent systems*, Bangalore, India, 21–22 November 1994
- 9 RAOL, J.R., and JATEGAONKAR, R.V.: 'Artificial neural networks for aerodynamic modelling'. DLR IB. 111-94/41, 1994
- 10 RAOL, J.R.: 'Parameter estimation of state space models by recurrent neural networks', *IEE Proc. Contr. Theory Appl.*, 1995, 142, (2), pp. 114–118
- 11 CETINKUNT, S., and CHIN, H.T.: 'Estimation of model parameters of linear structural systems using Hopfield neural networks', *J. Guid. Control Meas.*, 1994, 17, (3), pp. 621–624
- 12 CICHOCKI, A., and UNBEHAUEN, R.: 'Neural networks for solving systems of linear equations and related problems', *IEEE Trans. Circuits Syst.*, 1992, 39, (2), pp. 124–138
- 13 ZURADA, J.M.: 'Introduction to artificial neural systems' (West Publishing Company, 1992)
- 14 RAOL, J.R.: 'Neural network based parameter estimation of unstable aerospace dynamic systems: *IEE Proc., Control Theory Appl.*, 1994, 141, (6), pp. 385–388