# Back Propagation Through Adjoints for the Identification of Nonlinear Dynamic Systems Using Recurrent Neural Models

B. Srinivasan, U. R. Prasad, *Member, IEEE,* and N. J. Rao

***Abstract*—In this paper, back propagation is reinvestigated for an efficient evaluation of the gradient in arbitrary interconnections of recurrent subsystems. It is shown that the error has to be back-propagated through the adjoint model of the system and that the gradient can only be obtained after a delay. A faster version, accelerated back propagation, that eliminates this delay, is also developed. Various schemes including the sensitivity method are studied to update the weights of the network using these gradients. Motivated by the Lyapunov approach and the adjoint model, the predictive back propagation and its variant, targeted back propagation, are proposed. A further refinement, predictive back propagation with filtering is then developed, where the states of the model are also updated. The convergence of this scheme is assured. It is shown that it is sufficient to back propagate as many time steps as the order of the system for convergence. As a preamble, convergence of on-line batch and sample-wise updates in feedforward models is analyzed using the Lyapunov approach.**

## I. Introduction

NEURAL networks have recently emerged as a successful tool in the fields of pattern classification [1] and control of dynamic systems [2]–[8]. This is due to the computational efficiency of the back propagation algorithm [10], [11], and the versatility of the three layer feedforward neural network in approximating an arbitrary static nonlinearity [12], [13]. In this paper, we further exploit these features for the identification of nonlinear dynamic systems using neural networks. Algorithms proposed in this paper are for time invariant systems but can be directly applied to the identification of plants that are slowly time varying.

Identification of a system has two distinct steps; 1) choosing a proper model and 2) adjusting the parameters of the model so as to minimize a certain fit criterion. In the first part of this paper, we deal with the issue of choosing a neural network model for identification purposes. Since dynamic systems are described by differential or difference equations, in contrast to static systems that are described by algebraic equations, it is important to understand how general nonlinear dynamic systems can be modeled using neural networks that are versatile static maps. The choice of the model depends on whether all the states of the system or only the outputs are measured. If all the state variables are available, then a multilayer perceptron is sufficient to model a dynamic system. Even if only the outputs are available for measurement, it is possible under certain assumptions, to predict the output from delayed inputs and outputs using a multilayer perceptron [3], [4]. But for modeling a general nonlinear system, we illustrate through an example that feedforward models are inadequate and hence propose dynamic recurrent neural models for this purpose.

As for the second step of identification, i.e., parameter update, we use the gradient method for optimization. This procedure has two parts; 1) the gradient evaluation and 2) the update law. Though the gradient for static systems can be obtained using the standard back propagation algorithm, it fails in the presence of feedback. In such cases, the sensitivity method [9], [14], [15] and "back propagation through time" [2], [5] have been proposed for evaluating the gradient. The major drawback of the sensitivity method is that it is computationally intensive. Though the computational efficiency can be improved in linear time invariant dynamic systems as in [16], an extension of that result to nonlinear systems is not possible. Also, back propagation through time as available in literature, is not applicable to all types of interconnections of dynamic elements. So in this paper, we reinvestigate back propagation in the context of identification of dynamic systems and establish that the gradient for any dynamic system or for any interconnection of dynamic subsystems can be obtained by back propagating the error through the adjoint model [17]. Also, under the assumption of local observability, we show that, it is sufficient for the sake of convergence of parameter update laws, to back propagate as many time steps as the order of the system. As we go back in time with the adjoint, a delay is associated with the calculation of the gradient. This is avoided by accelerating the back propagation in a scheme termed "accelerated back propagation." This accelerated variant is adopted as the standard for back propagation in our work.

In recurrent models, though the gradient of the present cost with respect to a past parameter can be obtained using the adjoint, updating a past parameter in an online scheme is not possible. A few update schemes that reflect the various levels of approximation that can be made in this context are considered, the sensitivity method being one of them. Motivated by the Lyapunov approach and the adjoint model, a predictive scheme, predictive back propagation is proposed. A variant, targeted back propagation, which avoids explicit

Manuscript received February 3, 1993; revised July 26, 1993.

The authors are with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore-560012, India.

IEEE Log Number 9214799.

prediction, is also presented. A further refinement, predictive back propagation with filtering, is developed, in which the states of the system are also updated. Updating the states allows convergence to be established for a general nonlinear system. As a preamble, the problem of model matching in static systems is considered and the convergence to local and global minima with batch and sample-by-sample updates is analysed using the Lyapunov approach. The analysis shows that, convergence to the global minimum (if the initial conditions are in its vicinity) or to a small enough ball around a local minimum can be guaranteed with sample-wise updates.

Section II introduces different neural network models for the identification of nonlinear dynamical systems. In Section III, a discussion on computing the gradient in recurrent models is undertaken. The main algorithm for back propagation in dynamic systems through the adjoint and its accelerated variant are developed in Section IV. Having obtained the gradient, the convergence of gradient descent update laws in feedforward and feedback models is analyzed in Sections V and VI respectively. Simulation results presented in Section VII illustrate the applicability of various methods in the context of identification and control of dynamic systems.

## II. Models for Identification of Nonlinear Dynamic Systems

### A. The State-Output Model

It is well known in system theory that the state-output model, which relates the past and the present states, can represent a fairly large class of nonlinear dynamic systems. The state-output model is given by,

$$x(k) = f(x_p(k-1), u(k-1)) \text{ and } y(k) = g(x_p(k), u(k)) \tag{1}$$

where $u(k)$: input, $x(k)$: state of the model, $y(k)$: output of the model, $x_p(k)$: state of the plant, $y_p(k)$: output of the plant, and $k$: discretized time. The nonlinear functions $f()$ and $g()$ are static and hence can be approximated using feedforward neural networks [12], [13].

If all the states of the plant are measured in addition to its outputs, then the problem of learning $f()$ and $g()$ are decoupled. The plant states, $x_p(k)$, can be made the targeted outputs of the $f()$ network and the plant output, $y_p(k)$, for the $g()$ network. Any supervised learning algorithm such as back propagation [10] can be used for learning.

### B. The Nonlinear ARMA Models (NARMA)

Though the state-output model is quite general, all the plant states are not usually available for measurement. In such cases, an extension of the Auto Regressive Moving Average (ARMA) model, the Nonlinear ARMA model, which predicts the present output as a nonlinear function of the past inputs and outputs is proposed [3] [4] (Fig. 2).

$$\begin{aligned} y(k) = f(&y_p(k-1), y_p(k-2), \cdots, y_p(k-n), \\ &u(k), u(k-1), \cdots, u(k-m), W_f(k)) \end{aligned} \tag{2}$$

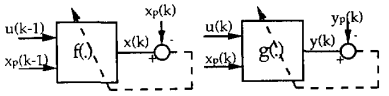



Fig. 1. The state—output model.

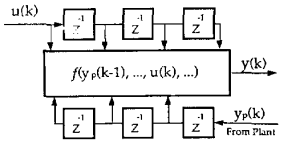



Fig. 2. NARMA—equation error model.

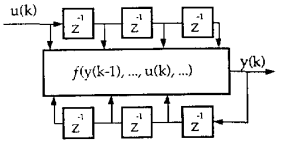



Fig. 3. NARMA—output error model.

This model is referred to as the prediction or series-parallel or equation error model or the model with teacher forcing [15]. Even though the overall system is dynamic, the nonlinear map $f()$ in (2) is static and hence can be modeled by a multilayer perceptron. Also, since the targeted output $y_p(k)$ is available, standard back propagation [10] can be used for learning.

The representation capability of NARMA models is only a subset of the state-output models. This is due to the fact that a NARMA model doesn't store any state information and relies on the delayed values of the inputs and the outputs to reconstruct the states. If such a reconstruction is not possible, then the NARMA model will not be capable of representing the given system. More precisely, NARMA models work only when the mapping $g$ of (1) is invertible with respect to the states $x(k)$. If $g$ is not invertible, then there is an ambiguity in reconstructing the states from the outputs. As an example consider the state-output model and its input-output representation given by:

$$x(k) = ax(k-1) + bu(k-1) \text{ and } y(k) = x^2(k) \tag{3}$$

$$y(k) = (a\sqrt{y_p(k-1)} + bu(k-1))^2 \tag{4}$$

Since $\sqrt{y_p(k-1)}$ is non-unique, no NARMA model can be used to represent (3).

The representation capability can be improved by interconnecting NARMA models. However, since only the model outputs are available for auto-regression at the subsystem level, the following NARMA model has to be used in interconnected systems: (Fig. 3)

$$\begin{aligned} y(k) = f(&y(k-1), y(k-2), \cdots, y(k-n), u(k), \\ &u(k-1), \cdots, u(k-m), W_f(k)) \end{aligned} \tag{5}$$

This model, which uses the model output for autoregression instead of the plant output, is referred to as the estimation or parallel or output error model. It is crucial to note that, the map $f$ in (5) depends on the past values of its own output $y(k)$,

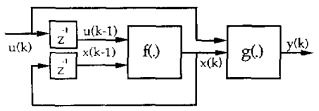



Fig. 4. The recurrent state model.

rendering the network non-feedforward. Hence, standard back propagation cannot be used for learning (5).

### C. Recurrent State Model

We note that the problem of recurrence has to be explicitly tackled when a general nonlinear system is to be learned. So the recurrent version of the state-output model is proposed where multilayer perceptrons are used with feedback as shown in Fig. 4. This can represent any arbitrary nonlinear dynamic system, but learning with generalized delta rule (back propagation) [10] is ruled out due to recursion. Explicitly showing the parametrization of the nonlinearities $f$ and $g$ in terms of their respective neural network weights $W_f$ and $W_g$, the recurrent state model can be written as:

$$x(k) = f(x(k-1), u(k-1), W_f(k)) \text{ and}$$
$$y(k) = g(x(k), u(k), W_g(k)) \tag{6}$$

In this section, we pointed out that if only the output information is available then the dynamics and nonlinearities cannot always be decoupled as in the NARMA-equation error models. Hence, this forces us to use recurrence in our models. In the rest of the paper, problems encountered in updating parameters in dynamic systems and adaptation amidst recurrence are discussed. Since these issues are generic to any interconnections of dynamic systems, the algorithms developed are directly extendable to adaptive control of nonlinear systems. This topic will be pursued elsewhere.

## III. Calculation of Gradient in Recurrent Models

If the model is purely feedforward (e.g., state-output and NARMA-equation error models), then the gradient can be obtained by standard back propagation. However, if feedback or recurrence is present in the model (e.g., Recurrent State and NARMA-output error models), it is necessary to calculate the "gradient amidst dynamics" for adaptation. In other words, due to the present output depending upon the past output of the network, the present error depends not only on the present parameter set but also on the past parameter values. So these dependencies have to be considered in the calculation of the gradient.

One method to calculate the "gradient amidst dynamics" is the sensitivity method [9] [14]. This method is also termed "on-line recurrent back propagation" [15] and the "dynamic back propagation" [9]. However, we stick to the terminology of "sensitivity method," since this can be considered to be "forward propagation" [2] as it calculates the gradient forward in signal flow and time. Assuming that $W_f$ and $W_g$ do not vary with time, (6) can be differentiated to obtain the derivative amidst dynamics with respect to representative components $w_f$ and $w_g$ of $f(.)$ and $g(.)$ respectively. We use $\nabla_x J$ to denote the gradient of the scalar $J$ with respect to a vector $x$ and $\Im_x f$ to denote the Jacobian. The gradient is considered as a column vector of dimension $\dim(x) \times 1$ and the Jacobian has the dimension $\dim(f) \times \dim(x)$. Let us use $f(k)$ to denote the network whose output is $x(k)$.

$$\frac{\partial x(k)}{\partial w_f} = \Im_{x(k-1)} f(k) \frac{\partial x(k-1)}{\partial w_f} + \frac{\partial f(k)}{\partial w_f} \tag{7}$$

$$\frac{\partial y(k)}{\partial w_f} = \Im_{x(k)} g(k) \frac{\partial x(k)}{\partial w_f}, \frac{\partial y(k)}{\partial w_g} = \frac{\partial g(k)}{\partial w_g} \tag{8}$$

From (7), we see that the "derivative amid dynamics" is the output of another recurrent network. This auxiliary network, called the sensitivity network, is similar to the original network (Fig. 4), the differences being 1) the network is a linearized version of the original, 2) the inputs of the system $u(k)$ do not enter this network, and 3) the partial derivative of the map $f(.)$ is injected as its input. Linearization can be achieved by replacing the sigmoid units by linear gain blocks, the gain being equal to the gradient of the sigmoid in the original network. This method requires as many sensitivity networks as there are parameters in the map $f(.)$ making it computationally intensive. This is a major drawback of the sensitivity method. Also, the gradient is obtained under the assumption that the weights remain constant over time. Since we adapt the weights, this assumption is not valid and hence neither the proper gradient is obtained nor can the convergence be assured.

In the sensitivity method the derivatives are calculated in the same direction as the signal flows and hence the necessity for one sensitivity model for each partial derivative. On the contrary, back propagation implements the chain rule of differentiation by the derivative of the error flowing in a direction opposite to that of the signal flow. Working backwards allows updating of all parameters of the network in a single run making it computationally efficient.

If the recurrent model (6) is unfolded in time, it is possible to calculate the gradient by back propagating the error in the time axis also [10], [11]. Due to the fact that in the recurrent model (6), the present state, $x(k)$, is a function of only the immediate past state, $x(k-1)$, the network unfolded in time is layered (cascade interconnection of $f()$ blocks). This is referred to as "back propagation through time" algorithm [2], [5]. The network unfolded in time for "$N$" time steps is given by,

$$y(k) = g\big[f\big(f\big(f\big(\cdots f(x(k-N), u(k-N))\cdots\big), u(k-2),\big)u(k-1)\big)\big] \tag{9}$$

However, in models such as the NARMA-output error model, where the present output $y(k)$ is a function of not just one but a number of its past values, unfolding and back propagation are not straightforward.

## IV. Back Propagation in Dynamic Systems

To obtain the proper gradient amidst dynamics for arbitrary interconnections of dynamic subsystems, back propagation is reinvestigated leading to the following result. Though the motivation for our approach is its applicability to more complex model structures, the result is stated for the recurrent model as it is a more versatile representation than the NARMA-output

error model. However, the proof is constructed by viewing the $f(.)$ map in (6) as a special case of the $f(.)$ map of (5) so as to emphasize a wider applicability of the approach.

*Theorem 1: Consider a dynamic system represented by the recurrent state model (6) with input: $u(k) \in \mathcal{R}^p$, states: $x(k) \in \mathcal{R}^n$, model and plant output: $y(k), y_p(k) \in \mathcal{R}^q$. Let the instantaneous scalar cost be $J(k) = \frac{1}{2}e(k)^T e(k)$ with $e(k) = y(k) - y_p(k)$. If the error is back-propagated through the adjoint model, whose initial conditions are set to zero with the adjoint input $= e(k)$ at time instant "$k$" and zero elsewhere, i.e.,*

$$\lambda(0) = \Im^T_{x(k)} g(k)\ e(k), \lambda(i) = \Im^T_{x(k-i)} f(k-i+1)\lambda(i-1), \quad i = 1, 2, \cdots, \phi \tag{10}$$

*then, the partial derivative of the instantaneous cost with respect to a representative component $w_f$ of $f(.)$ with time tag $(k-\phi), \phi \in \{1, .., k\}$, $\frac{\partial J(k)}{\partial w_f(k-\phi)}$, is the product of, 1) the input entering the weight "$w_f$" in the original network at time $(k-\phi)$ and 2) the input entering the weight "$w_f$" after $\phi$ retrograde time units in the adjoint network. In other words, the gradient $\nabla_{W_f(k-\phi)} J(k)$ and $\nabla_{W_g(k)} J(k)$ can be obtained using*

$$\nabla_{W_f(k-\phi)} J(k) = \Im^T_{W_f(k-\phi)} f(k-\phi)\ \lambda(\phi) \tag{11}$$

$$\nabla_{W_g(k)} J(k) = \Im^T_{W_g(k)} g(k)\ e(k) \tag{12}$$

*where $\Im^T_{x(k-\phi-1)} f(k-\phi)$ and $\Im^T_{W_f(k-\phi)} f(k-\phi)$ are the Jacobians obtainable from the map $f(.)$ at the time instant $(k-\phi)$, and $\Im^T_{x(k)} g(k)$ and $\Im^T_{W_g(k)} g(k)$ from the map $g(\cdot)$ at time instant "$k$."* □

We precede the proof of this theorem by some discussion and remarks.

*Discussion on Theorem 1:* To calculate the gradient amidst dynamics from the observed Jacobian, the adjoint model is seen to be useful. The adjoint of a dynamic system is constructed using the following rules [18]: 1) Reverse all signal flow, redefining nodes as summing junctions and vice versa. This converts inputs to outputs and vice versa. 2) Replace "$t$" in the arguments of all time varying coefficients by $(t_f - t)$, where "$t_f$" is the terminal time and "$t$" the forward time. Here "$t_f$" is the time of observation, "$k$." 3) If nonlinear blocks are present, they have to be successively linearized. Linearization in our case is achieved by using the derivative of the nonlinearity.

*Remark 1:* From the construction of the adjoint system described above, it is clear that the network through which back propagation is performed in static systems [10] is the adjoint of the forward system, the gradient being calculated at $\phi = 0$.

*Remark 2:* Among various models that can be proposed for nonlinear system identification and control, the neural network models proposed here have an edge over others due to the fact that the adjoint of the nonlinearity can be constructed with no additional computational burden.

*Remark 3:* Though similar adjoints are popular in the optimal control and missile guidance literature [18], [19], they only deal with the sensitivity of the cost or the output with respect to signals in the system. In such cases, the sensitivities are dependent only on the backward run. In the above theorem we extend this concept to deal with sensitivities of the cost with respect to the parameters of the system also. The point that has to be noted is that sensitivities with respect to the parameters do depend on the forward run also. Though sensitivities with respect to parameter variations using adjoints have been addressed to in circuit theory literature [20], the parameters are considered time invariant. In the present development such an assumption is not required.

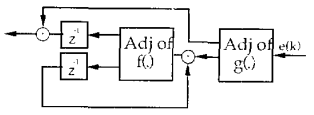



Fig. 5. Adjoint of the recurrent state model.

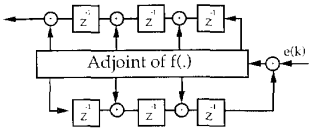



Fig. 6. Adjoint of the NARMA—output error model.

*Remark 4:* Note that by back propagating the error through the adjoint network, we apportion the error among weights spread over time. This is achieved by unfolding the network in time as in "back propagation through time." However, the adjoint approach gives a systematic methodology of unfolding any given configuration in time and back propagating the error through it.

*Remark 5:* In comparison with the sensitivity method, which decimates the time tag of the weight, back propagation through the adjoint retains it. Also, the adjoint method does not assume the weights to remain constant, and is thus better suited in an adaptive context like ours. The adjoint method uses only one model to calculate the derivatives with respect to all the adjustable parameters of the map, which is a striking improvement over the sensitivity method.

*Remark 6:* As mentioned earlier, though Theorem 1 is only stated for the recurrent state model, the concept of adjoints is applicable to any kind of interconnection of subsystems. For an interconnected system, the adjoint of the configuration is first formed (by reversing signal flow between the subsystems and exchanging the nodes and summing junctions) and every subsystem is then replaced by its adjoint. Adjoints of the recurrent state and the NARMA-output error models are shown in Figs. 5 and 6. Note that if a NARMA—equation error model is a part of the interconnected system, the error is back propagated through the moving-average part only, where as for the output error model, back propagation has to be done through the auto-regressive part also.

We first look into a more general result as far as the recurrent part of a dynamic system is concerned by considering the output to be dependent on a number of its past values as in (5).

*Lemma 1: Consider a dynamic system represented by the NARMA-output error model (5) with input: $u(k) \in \mathcal{R}^p$, model output and plant output: $y(k), y_p(k) \in \mathcal{R}^q$. Let the instantaneous scalar cost be $J(k) = \frac{1}{2}e(k)^T e(k)$ with $e(k) = y(k) - y_p(k)$. If the error is back-propagated through the adjoint model, with the*

*adjoint input* $= e(k)$ *at time instant "k" and zero elsewhere, i.e.,*

$$\lambda(0) = e(k), \lambda(\phi) = \sum_{i=0}^{\phi-1} \Im^T_{y(k-\phi)} f(k-i)\ \lambda(i) \qquad (13)$$

*then, the gradient* $\nabla_{W_f(k-\phi)} J(k)$*, can be obtained using*

$$\nabla_{W_f(k-\phi)} J(k) = \Im^T_{W_f(k-\phi)} f(k-\phi)\lambda(\phi) \qquad (14)$$

*where* $\Im^T_{y(k-\phi)} f(k-i)$ *and* $\Im^T_{W_f(k-\phi)} f(k-\phi)$ *are the Jacobians obtainable from* $f(.)$. □

We precede the proof of this lemma by some preliminary results.

*Proposition 1: For the dynamic system* (5) *the gradient of the instantaneous cost* $J(k)$ *with respect to* $W_f(k-\phi)$*, the weights of* $f(.)$ *with time tag* $(k-\phi), \phi \in \{0, 1, 2, .., k\}$, $\nabla_{W_f(k-\phi)} J(k)$*, is given by,*

$$\nabla_{W_f(k-\phi)} J(k) = \Im^T_{W_f(k-\phi)} f(k-\phi) \Im^T_{y(k-\phi)} y(k)\ e(k) \qquad (15)$$

*where the Jacobian* $\Im^T_{y(k-\phi)} y(k)$ *is obtained by the recursive relation,*

$$\Im^T_{y(k-\phi)} y(k) = \sum_{i=0}^{\phi-1} \Im^T_{y(k-\phi)} f(k-i)\ \Im^T_{y(k-i)} y(k) \qquad (16)$$

*where* $\Im^T_{y(k-\phi)} f(k-i)$ *and* $\Im^T_{W_f(k-\phi)} f(k-\phi)$ *are the Jacobians obtainable from* $f(.)$. □

*Proof:* Equation (15) follows from the chain rule of differentiation by noting that,

$$\Im^T_{W_f(k-\phi)} f(k-\phi) = \Im^T_{W_f(k-\phi)} y(k-\phi) \text{ and}$$
$$\nabla_{y(k)} J(k) = e(k), \qquad (17)$$

Equation (16) follows from the concept of total partial derivatives, where the effect of a change in $y(k-\phi)$ on $y(k)$ through all $y(k-i), i = 0, 1, 2, \cdots, (\phi-1)$, are summed, the direct effect of $y(k-\phi)$ on $y(k-i)$ being given by $\Im^T_{y(k-\phi)} f(k-i)$. □

*Proposition 2: With initial conditions set to zero in the adjoint model, and with the input* $= e(k)$ *at time instant "k" and zero elsewhere, the input to the adjoint of map* $f(.)$ *after* $\phi$ *retrograde time units in the adjoint run; i.e.,* $\lambda(\phi)$*, is given by the recursive relation* (13). □

*Proof:* Since the initial conditions are zeros, first part of (13) is apparent. Since the Jacobian is linearized version of the $f(.)$ map, the output of the adjoint of $f(.)$ at a position separated by "$j$" delays from the adjoint input, (Fig. 6) at retrograde time "$i$," is of the form $\Im^T_{y(k-i-j)} f(k-i)\ \lambda(i)$. From the construction of the adjoint (Fig. 6), it is evident that this signal will take another "$j$" retrograde time steps to reach the input of the adjoint. So at a given retrograde time "$\phi$," the signals at the input of the adjoint, will be such that $i + j = \phi$. The second part of (13) is a summation over all such signals. □

*Proposition 3: The vector* $\lambda(\phi)$ *is equal to the product* $\Im^T_{y(k-\phi)} y(k)\ e(k)$. □

*Proof by Induction:* The proposition holds for $\phi = 0$,

$$\Im^T_{y(k)} y(k) e(k) = e(k) = \lambda(0) \qquad (18)$$

Assume the proposition to hold for all $i \leq \phi - 1$. i.e.,

$$\Im^T_{y(k-i)} y(k) e(k) = \lambda(i),\ i = 0, 1, \cdots, \phi - 1 \qquad (19)$$

Then from (13) and (16), the proposition holds for $\phi$. □

*Proof of Lemma 1:* Applying Proposition 3 in (15) we get (14). Hence the gradient can be obtained by back propagating $\lambda(\phi)$ through the map $f()$ linearized at $k - \phi$. □

*Proof of Theorem 1:* For the proof of the theorem, (15)–(16) of Proposition 1 has to be changed to,

$$\nabla_{W_f(k-\phi)} J(k) = \Im^T_{W_f(k-\phi)} f(k-\phi) \times \Im^T_{x(k-\phi)} x(k) \Im^T_{x(k)} g(k)\ e(k) \qquad (20)$$

$$\Im^T_{x(k-i)} x(k) = \Im^T_{x(k-i)} f(k-i+1) \Im^T_{x(k-i+1)} x(k),$$
$$i = 1, 2, \cdots, \phi, \Im^T_{x(k)} x(k) = \mathbf{I} \qquad (21)$$

Consequently, Proposition 2 will include (10) instead of (13) and in Proposition 3, $\lambda(\phi) = \Im^T_{x(k-\phi)} x(k)\ \Im^T_{x(k)} g(k) e(k)$. Equation (11) of the theorem can be obtained along the same lines noting that the state of the recurrent model, $x(k)$, depends only on the immediate past state, $x(k-1)$, and $\Im^T_{x(k)} g(k) = \Im^T_{x(k)} y(k)$. Equation (12) is obvious as there is no dynamics in $g(.)$.

*Accelerated Back Propagation:* We will show later on that it is necessary to go back a finite number of steps, "$N$," through the adjoint for calculating the gradient necessary for adaptation. From Theorem 1 it is clear that this gradient can be obtained only after a delay of "$N$" adjoint time units. However, since back propagation is only through a model of the system, the basic time unit for the adjoint can be much smaller than that of the plant. In other words, back propagation can be performed through a time scaled model so as to obtain the gradient without any delay. Then, the computational complexity will depend on "$N$" and not on the number of weights. In comparison with the sensitivity method, which does parallel computations, the computations performed here are serial. In the rest of our discussions, this accelerated variant will be used for back propagation wherever needed.

## V. Parameter Update Process in Feedforward Models

Before we attack the problem of parameter update in recurrent models, let us first consider the update of parameters in models with no feedback such as the state-output and NARMA-equation error models. In these systems, the gradient can be obtained using the standard back propagation. Using this gradient, the parameters can be updated either off-line or on-line. For convergence to a local minimum in an off-line update scheme, it is sufficient to show that, away from the minimum, the cost function is reduced in every iteration. Hence updating along the negative gradient is sufficient for global convergence. However, if the update is on-line, the cost function, (which in our case is the integral squared error) depends on the input applied raising an important question. If the parameter is so chosen that cost incurred with this particular input is reduced, will the cost incurred with other inputs also be uniformly better? For this, if we are in a position to associate a Lyapunov function independent of the inputs with the dynamics of the weights, then convergence can be assured. As we are interested in on-line update schemes, we address the choice of such Lyapunov functions in this section.

The Lyapunov function to be chosen is for the dynamics of the weights and hence should be independent of the states

and inputs of the system. If the weights are updated along the negative gradient of the instantaneous cost, $W(k+1) = W(k) - \mu\nabla_{W(k)}J(k)$, then the change in Lyapunov function $\Delta V(k)$ will be given by,

$$\Delta V(k) = -(\mu + O(\mu))\nabla^T_{W(k)}J(k)\nabla_{W(k)}V(k) \qquad (22)$$

Note that $\Delta V(k) \propto -\nabla^T_{W(k)}J(k)\nabla_{W(k)}V(k)$ need not, in general, be sign definite. However, if the output is linear in parameters and $V(k)$ is the Euclidean norm of the parameter error, $\Delta V(k) \leq 0$. Such a sign definiteness does not come through if the output and the error are nonlinear in parameters. Then a parameter update law should resemble $W(k+1) = W(k) - \mu\nabla_{W(k)}V(k)$. However, if we define $V(k)$ as the Euclidean norm of the parameter error, then the gradient cannot be evaluated. So, taking the motivation from off-line approaches, we integrate the squared error with respect to the inputs and states over the complete range of interest to get a candidate Lyapunov function. In other words, this can be viewed analogous to an off-line scheme where the costs with respect to all possible inputs are considered. The weight dynamics is separated out from the dynamics of the states by integrating the squared error over the entire state space of interest.

Consider the weight dynamics, $W_g(k+1) = W_g(k) - \mu\nabla_{W_g(k)}J(k)$ along with the output maps of the state-output model $y(k) = g(x_p(k), u(k), W_g(k))$, and that of the plant $y_p(k) = g(x_p(k), u(k), W_g^*)$, where $J(k) = \frac{1}{2}(y(k) - y_p(k))^2$. A candidate for the Lyapunov function is obtained by integrating the error criterion with respect to the state $x_p(k)$ and the input $u(k)$ over the set of interest $\aleph$ weighting it by a suitable distribution $D(x_p, u)$:

$$V(k) = \frac{1}{2}\int_{(x_p,u\in\aleph)} \|g(x_p, u, W_g(k)) - g(x_p, u, W_g^*)\|^2 D(x_p, u)dx_p du \qquad (23)$$

Now three issues need to be addressed: 1) Can the gradient of $V(k)$ with respect to $W_g$ be calculated from a finite data set? 2) If so, which update law will assure negative definiteness of the $\Delta V(k)$ ? 3) At every instant the gradient for only one sample can be evaluated. Is it meaningful to use this for update?

Answering these questions in the general framework of functional approximation with process noise is a formidable task. So we limit ourselves to the problem of matching two functions, where we assume that the plant generates its data using a similar network whose weights are unknown. To answer the first question, let us extend the concept of informative data sets [21], which is similar to "persistently existing inputs" and "general enough inputs." Given a model $g(.)$ parameterized by $W_g$, and the set of interest $\aleph$, then a set $Z \subset \aleph$ is said to be informative with respect to the model $g(.)$ over the set $\aleph$, if an optimum over $Z$ is an optimum over $\aleph$.

*Proposition 4: If the number of parameters $W_g$ is finite then a finite informative data set exists.* □

*Proof:* In the problem of matching the outputs of two functions, the global minimum over the entire state-input set of interest $\aleph$ occurs when the weights of the networks generating them match. Once the weights match, the output of the two networks match for any subset of $\aleph$. In other words, the global minimizer of the Lyapunov function obtained by integrating the error criterion over $\aleph$ is also a global minimizer if the integration is performed over any subset of $\aleph$. The statement of the proposition is the converse of this statement. For the converse to be true, it should be possible to uniquely define the global minimum with the given subset. The number of independent conditions necessary to uniquely define a point in a finite dimensional space equals the cardinality of the parameter space. Since each input-output data is a condition on the optimal set of parameters, the global minimum can be uniquely defined by as many input points as there are parameters. Hence a finite informative data set exists. □

*Remark 7:* Though we have shown the existence of a finite data set, we need a consistent algorithm to take us to the global minimum. But, only convergence to a local minimum can be addressed when a gradient descent algorithm is used. But if we assume that the initial condition is in the vicinity of or in the attraction region of the global minimum, and if back propagation is used for optimization, then with a finite informative data set as above, global minimum with respect to that set will be reached, which by Proposition 4 is the desired global minimum.

Having shown the existence of a finite informative data set at least in the vicinity of the global minimum, whose length is $M$ (say), let us assume that every data set of length $M$ is informative. Then the integration over the set $\aleph$ can be replaced by summation of the squared error over $M$ terms. The distribution, $D(x_p, u) \approx D(z)$, does not appear explicitly in the summation as it is imbedded in the sequence of states and inputs over the interval $[k-M+1, k]$. Let us also assume that the informative sets are so chosen that the Lyapunov function is independent of the input set over which summation is performed.

$$V(k) = \frac{1}{2}\sum_{i=0}^{M-1} \|g(x_p(k-i), u(k-i), W_g(k)) - g(x_p(k-i), u(k-i), W_g^*)\|^2 = \sum_{i=0}^{M-1} J(k-i) \qquad (24)$$

*Proposition 5: If 1) every input data set of length $M$ is informative with respect to the model $g(.)$ over the set of interest $\aleph$, 2) the second partials of $V$ are bounded and 3) the update is done once in $M$ time steps using:*

$$W_g(k+1) - W_g(k-M+1) = -\mu\nabla_{W_g(k-M+1)}V(k) = -\mu\sum_{i=0}^{M-1}\nabla_{W_g(k-M+1)}J(k-i) \qquad (25)$$

*then there exists a $\mu > 0$ such that the batch update converges to a minimum.* □

*Proof:* Since every data set of length $M$ is informative, (24) can be chosen as a candidate Lyapunov function for the parameter update process. Since the weights have not changed during $[k-M+1, k]$, $V(k) = V(k-M+1)$. Using the

mean value theorem for $V(W_g(k+1))$ in the neighborhood of $W_g(k)$,

$$\begin{aligned} V(k+1) =& V(k) + (\Delta W_g)^T \nabla_{W_g(k)} V(k) \\ &+ \frac{1}{2}(\Delta W_g)^T \nabla^2_{W_g} V(k)(\Delta W_g) \qquad (26) \\ \Delta V(k) =& V(k+1) - V(k) \\ &\leq -\mu \|\nabla_{W_g(k)} V(k)\|^2 (1 - \frac{\mu}{2} \|\nabla^2_{W_g} V(k)\|) \qquad (27) \end{aligned}$$

Since the second derivative is bounded, $\mu$ can be so chosen that the second term is nonnegative. Then the Lyapunov function is non-increasing in successive batches and hence the batch update converges. □

Having proved the convergence of a batch processing type of update, now we turn to the third question *viz.*, whether updating every time instant also converges. If update is done every instant, sample by sample, the update law $W_g(k+1) = W_g(k) - \mu \nabla_{W_g(k)} J(k)$ can be rewritten as:

$$W_g(k+1) = W_g(k-M+1) - \mu \sum_{i=0}^{M-1} \nabla_{W_g(k-i)} J(k-i) \qquad (28)$$

Here $\nabla_{W_g(k-i)} J(k-i)$ is being used instead of $\nabla_{W_g(k-M+1)} J(k-i)$ of the batch update law. If we expand the gradient in the neighborhood of $W_g(k-M+1)$, noting that the difference, $(W_g(k-i) - W_g(k-M+1))$ is proportional to $\mu$, we get:

$$\begin{aligned} \nabla_{W_g(k-i)} J(k-i) &= \nabla_{W_g(k-M+1)} J(k-i) \\ &\quad + \nabla^2_{W_g} J(k-i)(W_g(k-i) \\ &\quad - W_g(k-M+1)) \\ &= \nabla_{W_g(k-M+1)} J(k-i) + \mu \rho(k-i) (\text{ say }) \end{aligned} \qquad (29)$$

$$\begin{aligned} W_g(k+1) = W_g(k-M+1) &- \mu \nabla_{W_g(k-M+1)} V(k) \\ &- \mu^2 \sum_{i=0}^{M-1} \rho(k-i) \end{aligned} \qquad (30)$$

As $\nabla_{W_g(k-M+1)} V(k)$, the sum of all sample gradients is bounded, the term $\rho(k-i)$, which consists of partial sums of sample gradients is bounded. So, if $\mu$ is so chosen that $\nabla_{W_g(k-M+1)} V(k)$ always dominates the summation of $\rho(k-i)$, then convergence is ensured. But as the weights approach an optimum, the summed gradient, $\nabla_{W_g(k-M+1)} V(k)$ tends to zero while the sample gradients need not do so. In one sense the sample gradients may contradict each other in such a way that the mean gradient goes to zero. So convergence to a local optimum cannot be assured with constant step size. However if the initial conditions are in the attraction region of the global minimum, convergence to the global minimum can be assured with a sample-by-sample update.

*Proposition 6: If 1) every input data set of length $M$ is informative with respect to the model $g(.)$ over the set of interest $\aleph$, 2) the second partials of $V$ are bounded (iii) the initial conditions are in the attraction region of the global minimum and (iv) the update is done every time step using:*

$$W_g(k+1) \quad = W_g(k) - \mu \nabla_{W_g(k)} J(k) \qquad (31)$$

*then there exists a $\mu > 0$ such that the sample-by-sample update converges to the global minimum.* □

*Proof:* In the case of matching two functions, by construction $\nabla_{W_g(k-M+1)} J(k-i)$ tends to zero for all "$i$" in the vicinity of the global minimum. For if the summation of these gradients, $\nabla_{W_g(k-M+1)} V(k)$ vanishes without every sample gradient vanishing, it means that it has not matched the function for some input, and hence is not a global minimum contradicting our assumption. So the gradient $\nabla_{W_g(k-M+1)} V(k)$ vanishes iff $\nabla_{W_g(k-M+1)} J(k-i)$ vanishes for all "$i$." This means that $\|\rho(k-i)\|$ for all "$i$" goes to zero iff $\|\nabla_{W_g(k-M+1)} V(k)\|$ goes to zero. So there exists a constant $\alpha$ such that, summation of $\|\rho(k-i)\| \leq \alpha \|\nabla_{W_g(k-M+1)} V(k)\|$. Since the initial conditions are in the attraction region of the global minimum, $\mu < \frac{1}{\alpha}$ leads to the second term of (30) dominating the third, and the update performed every time step converges to the global minimum. □

Now we analyze through this proposition how the asymptotic solution will behave if the initial conditions are in the vicinity of a local minimum.

*Proposition 7: Let $\beta$ be the upper bound of the sum $\sum_{i=0}^{M-1} \|\rho(k-i)\|$ and $\gamma$ for that of $\|\nabla^2_{W_g} V\|$. If 1) the learning parameter $\mu > 0$ is small enough to avoid the effect of the second partials of $V$ and 2) the update is done every time step using (31), then the norm of the gradient is confined to a ball $\|\nabla_{W_g(k-M)} V(k)\| \leq \beta\mu + \gamma\beta\mu^2$.* □

*Proof:* Given $\mu$, consider a ball around the optimum, where $\|\nabla_{W_g} V\| \leq \beta\mu$, and the last term of (30) dominates. Outside this ball the Lyapunov function will be non-increasing and hence it will enter this ball. Having entered, the negative definiteness of the time derivative is lost and hence the norm of the gradient may increase. To calculate how much the increase can be, we apply the mean value theorem considering the effect of only the last term in (30) as it is the only perturbing term.

$$\begin{aligned} \|\Delta(\nabla_{W_g} V(k))\| &= \|\nabla_{W_g} V(k+1) - \nabla_{W_g} V(k)\| \\ &= \|\nabla^2_{W_g} V(k) \Delta W_g\| \\ &\leq \|\Delta W_g\| \times \|\nabla^2_{W_g} V(k)\| \\ &\leq \mu^2 \| \sum_{i=0}^{M-1} \rho(k-i)\| \, \|\nabla^2_{W_g} V(k)\| \leq \beta\gamma\mu^2 \end{aligned} \qquad (32)$$

So the gradient will always be confined to a ball where $\|\nabla_{W_g(k-M)} V(k)\| \leq \beta\mu + \gamma\beta\mu^2$. The size of this ball can be made arbitrarily small by choosing a proper $\mu$. □

The convergence of $f(.)$ maps in the state-output model (1) follows the same arguments. We conclude this section by noting that in the vicinity of the global minimum sample-by-sample update converges to the global minimum, while in the vicinity of a local minimum convergence to a small-enough ball around it can be assured.

## VI. Parameter Update Process in Recurrent Models

So far we have been dealing with systems in which there is no feedback. The learning problem then was an unconstrained optimization problem. But, if feedback is present in the system then the learning problem is one of constrained optimization,

the constraints originating from the dynamics of the system. Further, the past parameters can also affect the present output. So the gradients with respect to past and present parameter sets have to be obtained using the adjoint model described in Section IV. Updating the present parameter set using these gradients in an on-line scheme is discussed here. In particular, the recurrent map $f(.)$ of (6) will be considered (The learning of the $g(.)$ map of (6) is identical to the problem discussed in the last section). In this section we will discuss through different schemes the various levels of approximation that can arise in this problem. We will initially assume that the Lyapunov function introduced in the last section (24) is the cost function to be minimized. Also, in Proposition 6 we saw that the samplewise update and batch update are equivalent if the initial conditions are in the vicinity of the global minimum. We will assume the initial conditions to be so and discuss the convergence to the global minimum with constant step size and samplewise update resembling (31).

*Scheme 1—Direct Adaptation:* The first approximation is to assume that the effect of any past parameter set on the present error is negligible. So it is sufficient to adapt only the present parameter set.

$$W_f(k+1) = W_f(k) - \mu \nabla_{W_f(k)} J(k), \mu > 0 \qquad (33)$$

With this approximation, which is equivalent to ignoring the dynamics, the algorithm boils down to standard back propagation. This is computationally inexpensive and works when the dynamics are insignificant.

*Scheme 2—Sensitivity Method (Forward Implementation):* The next approximation is to assume that the parameters remain constant as in the sensitivity calculation. We use $\nabla_{W_f(k-\phi)} J(k), \forall\ \phi \in [0, k]$, to adapt the weights $W_f(k-\phi)$. Since we are looking for a constant parameter set that would minimize $J(k)$, we resort to averaging that gives:

$$W_f(k+1) = W_f(k) - \mu \sum_{\phi=0}^{k} \nabla_{W_f(k-\phi)} J(k), \mu > 0 \qquad (34)$$

This update involves a summation that increases in length with time. In the forward implementation this summation is automatically taken care of. However, in the backward implementation, the length of adjoint run keeps increasing with time making the backward implementation impractical. Note that the update is not a batch update but a samplewise update.

*Scheme 3—Sensitivity Method (Backward Implementation):* Here we will modify Scheme 2 so that it is implementable in the backward sense. For this, we run the adjoint only for a finite period of time. For the sensitivity method to converge, a necessary condition is that the series

$$g^{k+1} = \sum_{\phi=0}^{k} \nabla_{W_f(k-\phi)} J(k) < \infty \text{ as } k \to \infty. \qquad (35)$$

Under this necessary condition, the effect of truncation of the summation $g^{k+1}$ is studied in the following proposition.

*Proposition 8: If (35) holds, then $\exists$ a finite $N$ such that, the finite horizon update,*

$$W_f(k+1) = W_f(k) - \mu \sum_{\phi=0}^{N-1} \nabla_{W_f(k-\phi)} J(k) \qquad (36)$$

*can be used instead of the infinite horizon update (34) for the purpose of optimization.* □

*Proof:* For optimization, it is not necessary to follow the negative gradient, $-g^{k+1}$, but is sufficient to choose a descent direction [22], $-d^{k+1}(N)$ such that, $g^{(k+1)T} d^{(k+1)}(N) > 0$. Consider the sequence $g^{(k+1)T} d^{(k+1)}(N)$, where

$$d^{k+1}(N) = \sum_{\phi=0}^{N-1} \nabla_{W_f(k-\phi)} J(k). \qquad (37)$$

Since $g^{k+1}$ is a convergent series, either the last term or the sum of last few terms tend to zero as $\phi \to k$ and $k \to \infty$. Also as $k \to \infty$ and $N \to \infty, g^{(k+1)T} d^{(k+1)}(N) \to g^{\infty T} g^{\infty} > 0, \exists$ a finite $N$ such that, $g^{(k+1)T} d^{(k+1)}(N) > 0$. □

With this proposition we prove that it is sufficient to go back a finite number of steps $(N)$ through the adjoint making the backward implementation of the sensitivity method feasible. Note that the truncation in this context is not a further approximation over Scheme 2. On the contrary, the assumption that the parameters remain constant over a finite period can actually be met when batch processing is performed as shown below:

$$W_f(k+1) = W_f(k-N+1) - \mu \sum_{\phi=0}^{N-1} \nabla_{W_f(k-\phi)} J(k) \qquad (38)$$

The major drawback in implementing this update is that it uses only $\frac{1}{N}$th of the information available as we disregard $J(k-i), i = 1, 2, \cdots, N-1$. On the contrary, if sample by sample update is done as in (36), then we are adapting a parameter that has a totally different time tag.

*Scheme 4—Predictive Back Propagation:* In a recurrent network the present weight $W_f(k)$ affects the future cost $J(k+i)$ for all $i \geq 0$. So for analyzing how an error caused by the present set of weights evolves due to dynamics of the system, it is desirable to choose the criterion function that looks into the entire time scale up to infinity. However, since the state space of the system is finite, it is not necessary to analyse till infinity, a finite window of length "$N$" say being sufficient. Also for the sake of implementation it is required that the analysis interval be finite. Hence the gradient that has to be used is

$$\sum_{i=0}^{N-1} \nabla_{W_f(k)} J(k+N-i),$$

which can be obtained only at time instant $(k+N)$. However, this gradient cannot be used to update the then weight, $W_f(k+N)$. To resolve the problem of using derivatives with respect to past parameters to correct the present ones, we resort to prediction in this scheme. In the feedforward case, the Lyapunov function was chosen so that a weight gets tested over the entire input set of interest. In the presence of feedback, the criterion function is so chosen that it not only evaluates the

weights over the entire input space but also over the evolution of those inputs over the time axis. For this, we look ahead in time using an $N$ step prediction at every time instant and obtain the effect of the present parameters on the future costs. Let $J_p^*(k)$ be the $N$ step prediction cost defined by

$$J_p^* = \frac{1}{2}\sum_{\phi=0}^{N-1} \|g(x(k+N-\phi), u(k+N+\phi)) - g(x_p(k+N-\phi), u(k+N-\phi))\|^2 \quad (39)$$

under the constraint

$$x(k+N-\phi) = f(x(k+N-\phi-1), u(k+N-\phi-1), W_f(k)) \quad (40)$$

Since the dynamics act as constraints, Lagrange multipliers are introduced to get

$$J_p(k) = \sum_{\phi=0}^{N-1} \frac{1}{2}\|g(x(k+N-\phi)) - g(x_p(k+N-\phi))\|^2 - \lambda^T(\phi)[x(k+N-\phi) - f(k+N-\phi)] \quad (41)$$

Differentiating (41) with respect to $x(k+N-\phi)$ and equating it to zero leads to the adjoint equation:

$$\lambda(\phi) = \Im^T_{x(k+N-\phi)} f(k+N-\phi+1)\lambda(\phi-1) + \Im^T_{x(k+N-\phi)} g(k+N-\phi) [g(x(k+N-\phi)) - g(x_p(k+N-\phi))] \quad (42)$$

$$\nabla_{W_f(k)} J_p(k) = \Im^T_{W_f(k)} f(k)\lambda(N) \quad (43)$$

As was seen earlier, to obtain the "gradient amidst dynamics," $\nabla_{W_f(k)} J_p(k)$, we first go ahead in time $N$ steps by prediction and then go back in time through the adjoint of the system. In Theorem 1, back propagation of a criterion that is of the form $\frac{1}{2}[y(k) - y_p(k)]^2$ was discussed, where the adjoint input was zero $\forall \phi \neq 0$. However, if the cost is a sum of squared errors, as in (39), then the adjoint input at retrograde time "$\phi$" will be as in the second term of (42).

*Choice of N:* In the following proposition, we show that the size of the prediction window "$N$" should be at least the order of the system "$n$" and predicting over a larger interval is superfluous.

*Proposition 9:* If the system is locally observable at all operating points then it is sufficient to back propagate as many time steps as the order of the system. □

*Proof:* The gradient $\nabla_{W_f(k)} J_p(k)$ gives the decoupled effect of the weights at time "$k$," $W_f(k)$, on $J_p(k)$, regardless of the weights used at other time instants during prediction, $W_f(k+i)$ for $i = 1, 2, \cdots, N-1$. So let us assume that $W_f(k+1) = W_f^*$ for $i = 1, 2, \cdots, N-1$. Then, given $x(k-1) = x_p(k-1)$, the cost function $J_p(k)$ is just the cost associated with the time evolution of the error in the state $(x(k) - x_p(k)) = e_x$ (say), which is induced by the mismatch in weights at time instant "$k$." Considering the system linearized around an operating point and assuming that the operating point does not change, the cost and the sensitivities can be calculated as follows. For the simplicity of notation, let $\Im_{x(k)} f(k) = A$ and $\Im_{x(k)} g(k) = C$.

$$J_p^*(k) = \frac{1}{2}\sum_{\phi=0}^{N-1} \|CA^{(N-1-\phi)} e_x\|^2 \quad (44)$$

$$\lambda(0) = 0, \lambda(\phi) = A^T\lambda(\phi-1) + C^T(CA^{(N-\phi-1)}e_x) \quad (45)$$

$$\lambda(N) = \sum_{\phi=0}^{N-1} A^{T(N-1-\phi)} C^T C A^{(N-1-\phi)} e_x \quad (46)$$

If we assume that the system is observable, then the observability matrix $\Theta_n$ is full rank and so is $\Theta_N$ for any $N \geq n$. Hence $\Theta_N^T\Theta_N$ is positive definite. The observability matrix and the sensitivity are given by,

$$\Theta_n^T = [C^T|, A^TC^T|A^{2T}C^T| \cdots |A^{(n-1)T}C^T] \quad (47)$$

$$\lambda(N) = \Theta_N^T\Theta_N e_x \quad (48)$$

Ideally, we would like to evaluate the weights over the entire time axis. In such a case we would have calculated $\lambda_\infty = \Theta_\infty^T\Theta_\infty e_x$. But since we go only for a finite length of time, we have $\lambda(N)$ as in (48). These two adjoint outputs are related by, $\lambda(N) = (\Theta_N^T\Theta_N)(\Theta_\infty^T\Theta_\infty)^{-1}\lambda_\infty$. However, both $(\Theta_N^T\Theta_N)$ and $(\Theta_\infty^T\Theta_\infty)^{-1}$ are positive definite for $N \geq n$. So if $-\lambda_\infty$ is a descent direction, so is $-\lambda(N)$. This means that it is sufficient to back propagate as many time steps as are required to make $\Theta_N$ full rank, which is the order of the system from the observability assumption. □

*Remark 8:* From the above proposition we see that as a dynamic system cannot be assessed with a single sample, we need to wait for a certain minimum number of time units equal to the order of the system. Also we see that a longer delay does not yield any further information. This waiting time is circumvented in the present scheme by the use of prediction.

In the context of adaptation in dynamic systems, we see that there are two issues that have to be addressed. The first issue is the error caused by the mismatch of weights. In Proposition 9, we considered the propagation of the state error arising due to the mismatch in the weights, in order to judge how a change in the present weights will affect the long term cost. The second issue is how we can control (keep within certain bounds) the propagation of the state error. In the present scheme and the next we will assume that the propagation of the state error is somehow kept under control. In the last scheme (Scheme 6—Predictive Back Propagation with Filtering), we propose a methodology to keep the propagation of the error under control.

Now assuming that every data set of length $M$ is informative, in this case even over the time axis, we define the Lyapunov function:

$$\Psi(k) = \sum_{i=0}^{M-1} J_p^*(k-i) \quad (49)$$

*Proposition 10: If 1) the system is observable, 2) every input data set of length $M$ is informative with respect to the model $f(.)$ over the set of interest $\aleph$, 3) If the initial states of the plant and the model match at the beginning of every batch of length $M$, 4) the second partials of $\Psi$ are bounded and 5) the update is done once in $M$ time steps using:*

$$W_f(k+1) - W_f(k-M+1) = -\mu \nabla_{W_{f(k-M+1)}} \Psi(k) = -\mu \sum_{i=0}^{M-1} \nabla_{W_f(k-M+1)} J_p(k-i) \tag{50}$$

$$\lambda_i(\phi) = \Im^T_{x(k-i+N-\phi)} f(k-i+N-\phi+1)\lambda_i(\phi-1) + \Im^T_{x(k-i+N-\phi)} g(k-i+N-\phi) \times [g(x(k-i+N-\phi)) - g(x_p(k-i+N-\phi))] \tag{51}$$

$$\nabla_{W_f(k-M+1)} J_p(k-i) = \Im^T_{W_f(k-M+1)} f(k-i)\lambda_i(N) \tag{52}$$

*then there exists a $\mu > 0$ such that the batch update process converges to a minimum.* □

*Proof:* The batch cost $\Psi(k)$ arises due to 1) mismatch in initial conditions and 2) mismatch in weights. By the assumption on the initial conditions, the batch cost depends only on the mismatch of weights. From Proposition 9, the finite horizon gradient evaluated over $N$ steps, $\nabla_{W_f(k-M+1)}\Psi(k)$, is a descent direction. Under the assumption of informative sets, the proof follows from Proposition 5, where the convergence of a batch update along a descent direction is proved. □

*Proposition 11: If 1) the system is observable, 2) every input data set of length $M$ is informative with respect to the model $f(.)$ over the set of interest $\aleph$, 3) the states of the plant and model somehow match every sample, 4) the second partials of $\Psi$ are bounded and 5) the update is done every time step using:*

$$W_f(k+1) = W_f(k) - \mu \nabla_{W_f(k)} J_p(k) \tag{53}$$

*where $\nabla_{W_f(k)} J_p(k)$ is calculated as in (42)–(43), then, there exists a $\mu > 0$ such that the sample by sample update process either converges to the global minimum or to a close enough neighborhood of a local minimum.* □

*Proof:* Follows from Propositions 6, 7, and 10. □

*Remark 9:* The key problem in using prediction is that the future values are not available to us for correction. To overcome this we place our model "$N$" steps behind in time, i.e., at $(k-N)$, and predict from $(k-N+1)$ to "$k$." In a samplewise update this should be carried out every sampling instant with the new weights. In other words, the adaptation will not be in real time but behind it by "$N$" time units. This is only an implementation aspect and does not affect convergence.

*Scheme 5—Targeted Back Propagation:* The major problem in Scheme 4 is that it requires explicit prediction that is computationally expensive. This was done so that the time tag of the weight and the time tag of the gradient are matched. In this scheme we look into an alternative that avoids explicit prediction. The key idea is that in static maps, the gradient can be calculated independent of time, once the target inputs and outputs are available. In particular, if in the recurrent state model (6), good estimates of the states are available, then the problem gets decoupled as in the case of state-output model (1).

So, instead of obtaining the gradient with respect to the past weights (the problem is that the weights have changed and we don't know how to use these gradients), we use the adjoint to calculate the gradient with respect to the state $x(k-N)$ and correct it. Also note that in the predictive back propagation scheme, gradient calculation in (43) is just back propagating $\lambda(N)$ through a static map. Let $x'(k-N)$ be used to represent the corrected state. Then,

$$x'(k-N) = x(k-N) - \eta \nabla_{x(k-N)} J_p(k-N) = x(k-N) - \eta\lambda(N), \eta > 0. \tag{54}$$

In this scheme the state variables are considered as adjustable parameters that are updated to minimize the cost function $J_p$. By updating along the gradient, in an off-line scheme, the state converges to its desired value. Assuming that such a convergence is also possible in an on-line update like ours, the corrected states, $x'(k-N)$ and $x'(k-N-1)$ can be taken to be the true values of $x_p$. Then we have an input-output relationship, $[x'(k-N-1), u(k-N-1)] \rightarrow x'(k-N)$, which can be presented to the static map $f(.)$ for the calculation of the gradient. So we reapply $x'(k-N-1), u(k-N-1)$ to the network at time "$k$" and use the error to correct the present map. The gradient obtained is:

$$\nabla_{W_f(k)} J_t(k) = \Im^T_{W_f(k)} f(k)(f(x'(k-N-1), u(k-N-1), W_f(k)) - x'(k-N)) \tag{55}$$

$$J_t(k) = \frac{1}{2}\| f(x'(k-N-1), u(k-N-1), W_f(k)) - x'(k-N)\|^2 \tag{56}$$

Note that, in the absence of changes in weights, if $x'(k-N-1) = x(k-N-1)$, then the error term is equal to $\eta\lambda(N)$. Hence the same gradient as in (42)–(43) is obtained. $[\nabla_{W_f(k)} J_t(k) = \nabla_{W_f(k)} J_p(k)]$. However, with the previous state and weights updated the gradient is altered so as to account for the change in the weights. Though the gradient of Scheme 5 can be obtained as $\eta \rightarrow 0$, it is seen that a non-zero $\eta$ speeds up convergence. In comparison with the earlier scheme, this algorithm requires only an additional rerun of the static map instead of an $N$ step prediction.

*Scheme 6—Predictive Back Propagation with Filtering:* In the convergence analysis of the predictive scheme, it is not realistic to assume that the plant and the model states match at every sampling instant. Further, any mismatch in weights leads to a mismatch in the states. To take care of this spill-over from one sampling instant to another, the states should also be adapted. In the context of parameter adaptation, correction of the states is quite logical and improves the speed of adaptation. Consider a situation in which a state was pushed into error due to a wrong parameter. If we do not adapt the states, they will not return to their true values even if the parameters converge to their correct values. The error in the states will then be attributed to the parameters leading to instability in the adaptation mechanism.

In the targeted back propagation scheme a methodology for the correction of the states was discussed. In the present scheme we combine the last two schemes, by first correcting

the state and then using the corrected value for further prediction. Hence, this scheme requires prediction at every time instant. The convergence result with the parameter and state updates is stated in the following theorem.

We first introduce the notation that will be used. Let a transformation $\overline{T} : A \rightarrow \overline{A}$, be defined by, $\overline{A} = E^{-1}\overline{D}E$, where $A = E^{-1}DE, E$ the matrix of eigenvectors, $D$ the diagonal matrix containing the eigenvalues and $\overline{D}$ a diagonal matrix defined by,

$$\bar{d}_{ii} = \begin{cases} d_{ii}, \text{ if } |d_{ii}| \geq \frac{1}{\sqrt{2}} \\ \frac{d_{ii}}{\sqrt{2}|d_{ii}|}, \text{ if } |d_{ii}| < \frac{1}{\sqrt{2}} \end{cases} \text{ and } \overline{A} = E^{-1}\overline{D}E. \quad (57)$$

*Theorem 2:* **If** *1) the system is observable, 2) the second partials of $\Psi$ are bounded, 3) every input data set of length $M$ (say) is informative with respect to the model $f(.)$ over the input-state set of interest $\aleph$, 4) if the gradient is obtained using the adjoint equations (42)–(43), 5) the parameters are updated every time step using (53), and 6) the states updated using:*

$$\xi(k-N) = (2+\kappa)\left[\overline{A}\xi'(k-N-1)\overline{A}^T + \frac{\Gamma\Gamma^T}{\Gamma^T\overline{A}^{-T}(\xi')^{-1}\overline{A}^{-1}\Gamma}\right] \quad (58)$$

$$\xi'(k-N) = (\xi^{-1}(k-N) + \eta(\Theta_N^T\Theta_N))^{-1} \quad (59)$$

$$x'(k-N) = x(k-N) - \eta\xi'(k-N)\lambda(N) \quad (60)$$

*where $\xi(k)$ is an estimate of the covariance of the error in the states, $\Gamma$ the upper bound that describes the influence of the weight mismatch in the dynamics of the estimated norm, $\overline{A} = \overline{T}(\Im_{x(k-N)}f(k-N)), \Theta_N$ the observability matrix of the linearized transformed system at time instant $(k-N)$ and $\kappa = \eta\|\xi'\| \ \|\Theta_N^T\Theta_N\|$, 7) the error is such that the linearization is valid,* **then**, *there exist constants $\eta, \mu > 0$ such that the sample-wise update converges either to the global minimum or to a close enough neighborhood of the local minimum.* □

*Proof:* If the states of the model and the plant do not match at every sampling instant, then $J_p(k)$, and hence its batch sum consists of two components: that caused due to a mismatch in the parameters, $\Psi_w(k)$, and that caused due to an error in the initial state, $\Psi_x(k)$, i.e., $\Psi(k) = \Psi_w(k) + \Psi_x(k)$, where $\Psi_x(k)$ has the summation form of $e_x(\Theta_N{}^T\Theta_N)e_x$ from (46). From Proposition 11 it can be seen that the error caused by the parameters, $\Psi_w(k)$, keeps decreasing in successive batches. The proof will be completed if we show that some function of the error in states $e_x$ also reduces in successive batches. In this proof we construct a function that decreases in every time step and hence the result is directly applicable to sample-wise updates.

Let $e_x(k-N)$ be the actual error in the states and $\xi(k-N)$ the estimated covariance of it. A candidate Lyapunov function that can be associated with the dynamics of the error in the states [23] is:

$$V_x(k) = \frac{1}{2}e_x{}^T(k-N)\xi^{-1}(k-N)e_x(k-N) + \frac{\nu^2(k-N)}{2}\Gamma^T\xi^{-1}(k-N)\Gamma \quad (61)$$

$$\nu^2(k-N) = (1+\kappa)\Psi_w^2(k-N) \text{ and } \nu'^2(k-N) = \Psi_w^2(k-N) \quad (62)$$

The error in the states depends upon the past corrected error and the error induced by the parameter errors.

$$e_x(k) = \Im_{x(k)}f(k)e_x'(k-1) + \Gamma\Psi_w(k) \leq \overline{A}e_x'(k-1) + \Gamma\Psi_w(k) \quad (63)$$

Let the covariance be updated by using the recursive equation (58). With the updates (58) and (63) (time update—update between sampling instants), we will show that the function $V_x$ is non-increasing; i.e., $V_x(k) \leq V_x'(k-1)$.

Equation (64) can be obtained by noting: 1) $\Psi_w$ is monotonically non-increasing, i.e., $\Psi_w^2(k-N-1) + (1+\kappa)\Psi_w^2(k-N) \leq (2+\kappa)\Psi_w^2(k-N-1)$, 2) $\|\overline{A}^{-1}\Gamma\| \leq \sqrt{2}\|\Gamma\|$, and 3) $(e_x'^T\xi'^{-1}e_x' - 2\Psi_w e_x'^T\xi'^{-1}\overline{A}^{-1}\Gamma + \Psi_w^2\Gamma^T\overline{A}^{-T}\xi'^{-1}\overline{A}^{-1}\Gamma)$ is perfect square.

Though $V_x(k) \leq V_x'(k-1)$, the covariance $\xi(k-N)$ may become unbounded with time. To avoid this, we correct the states along the gradient of the Lyapunov function and also update the covariance $\xi(k-N)$ in such a way that $V_x$ keeps decreasing with the measurement update (correction process); i.e., $V_x'(k) \leq V_x(k)$, and $\xi$ is bounded. From (46) the state correction (60) can be written as,

$$x'(k-N) = x(k-N) - \eta\,\xi'(k-N)(\Theta_N^T\Theta_N)e_x(k-N) \quad (65)$$

If the covariance be updated using (59) and $\nu$ as in (62), the change in $V_x$ will be,

$$V_x'(k) - V_x(k) = \frac{1}{2}(e_x'^T(k-N)\xi'^{-1}(k-N)e_x'(k-N)$$

$$\begin{aligned} V_x(k) - V_x'(k-1) &= \frac{1}{2}\left(e_x^T(k-N)\xi^{-1}(k-N)e_x(k-N) - e_x'^T(k-N-1)\xi'^{-1}(k-N-1)e_x'(k-N-1)\right) \\ &\quad + \frac{1}{2}(\nu^2(k-N)\Gamma^T\xi^{-1}(k-N)\Gamma - \nu'^2(k-N-1)\Gamma^T\xi'^{-1}(k-N-1)\Gamma) \\ &\leq -\frac{(1+2\kappa)e_x'^T\xi'^{-1}e_x' + (e_x'^T\xi'^{-1}e_x' - 2\Psi_w e_x'^T\xi'^{-1}\overline{A}^{-1}\Gamma + \Psi_w^2\Gamma^T\overline{A}^{-T}\xi'^{-1}\overline{A}^{-1}\Gamma)}{4(2+\kappa)} \\ &\quad - \frac{1}{2}\left[\Psi_w^2(k-N-1)\Gamma^T\xi'^{-1}\Gamma - \left(\frac{\Psi_w^2(k-N-1) + (1+\kappa)\Psi_w^2(k-N)}{(2+\kappa)}\right)\left(\frac{\Gamma^T\overline{A}^{-T}\xi'^{-1}\overline{A}^{-1}\Gamma}{2}\right)\right] \leq 0 \quad (64) \end{aligned}$$

$$
\begin{aligned}
&- e_x^T(k-N)\xi^{-1}(k-N)e_x(k-N)) \\
&+ \frac{1}{2}(\nu'^2(k-N)\Gamma^T\xi'^{-1}(k-N)\Gamma \\
&- \nu^2(k-N)\Gamma^T\xi^{-1}(k-N)\Gamma) \\
&= e_x^T\xi'^{-1}\Delta e_x + \frac{1}{2}e_x^T\Delta\xi^{-1}e_x \\
&+ \frac{1}{2}\Gamma^T(\Delta\nu^2\xi'^{-1} + \nu'^2\Delta\xi^{-1})\Gamma \\
&\leq -e_x^T\xi'^{-1}\eta\xi'(\Theta_N^T\Theta_N)e_x + \frac{1}{2}e_x^T\eta(\Theta_N^T\Theta_N)e_x \\
&\quad - \frac{1}{2}\Psi_w^2(k-N)\frac{\Gamma^T\Gamma}{\|\xi'\|}(\kappa - \eta\|\xi'\|\,\|\Theta_N^T\Theta_N\|) \leq 0 \quad (66)
\end{aligned}
$$

We see that $V_x$ is non-increasing during both estimation and correction; i.e., $V_x'(k) \leq V_x(k) \leq V_x'(k-1)$. Also $\xi$ is bounded from above by $\frac{1}{\eta}(\Theta_N^T\Theta_N)^{-1}$. Since $(\Theta_N^T\Theta_N)$ is positive definite from the observability assumption and $\eta > 0, \xi$ does not blow up to infinity. With the covariance appearing in the update equation, whenever the estimated covariance of the error is large then a large step is taken along the descent direction. This is necessary as our correction should at least compensate for the increase in error due to dynamics. By our bound on $\xi$, we find that $\xi'\lambda < e_x$, which is acceptable in a quadratic programming problem like ours. So, if we redefine the Lyapunov function as

$$\Psi'(k) = \Psi_w(k) + \Psi_{x\xi}(k) \quad (67)$$

where $\Psi_{x\xi}(k)$ is the sum of $V_x(k)$ over the batch, then the updates mentioned in the theorem decrease it monotonically and hence the batch update converges. Arguments for samplewise update follow from Propositions 6 and 7. □

*Extended Kalman filtering:* For the adaptation of states, we go back in time through the adjoint and correct a past state. From there an "$N$" step prediction is done to correct the present state. Due to the dynamics we go through a backward and a forward pass. This can be solved without going back and forth in time by using a Kalman filter [24]. Targeted back propagation is a scheme that performs parameter adaptation without prediction. Using the concept of Extended Kalman filtering the state adaptation can be performed without explicit prediction. These two concepts can be combined to yield a scheme "targeted back propagation with filtering," which approximates the above scheme and is computationally less expensive.

*Remark 10:* Note that if the system is stable, the system dynamics itself reduces the state error. In such cases, state update as in Theorem 2 is not very essential.

*Remark 11:* All the algorithms developed here are applicable to cases where the model is some arbitrary interconnection of subsystems (NARMA—output error model). The steps that have to be used in learning such an interconnected system are: 1) Construct the adjoint of the interconnecting loop replacing every subsystem by its corresponding adjoint to obtain the adjoint of the overall system. 2) Back propagate the error through the overall adjoint to calculate the gradient with respect to the weights and signals. 3) Using the gradient, the adaptation can be performed using any one of the schemes developed earlier.

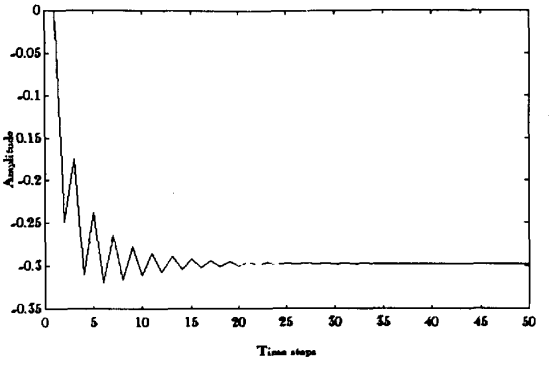



Fig. 7. Response for a unit step input.

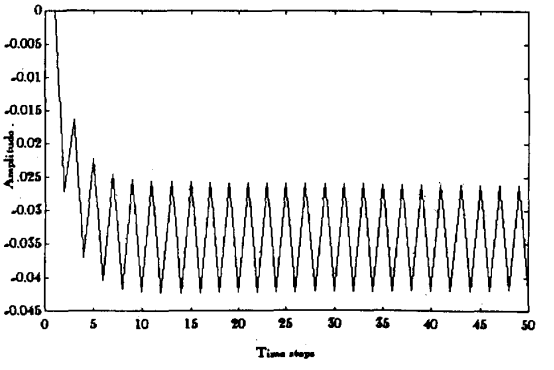



Fig. 8. Response for a 0.1 unit step input.

We close this section by noting that the algorithms presented here are quite computationally intensive compared to those used for feedforward systems, which is inevitable if the dynamics are significant.

## VII. Simulation Results

Here we present a simple example to illustrate the ideas reported in this paper. The plant considered is a second order plant described by the state and output equations:

$$
\begin{aligned}
x_1(k) &= f(x_1(k-1), x_2(k-1), u(k-1)), \\
y(k) &= x_2(k) = x_1(k-1)
\end{aligned} \quad (68)
$$

The nonlinear mapping $f(.)$ with which the plant generates its output is chosen to be a three layer perceptron with 3 inputs, 3 hidden nodes and one output. Bipolar sigmoid nonlinearity is used for activation in all units. Weights for the plant are arbitrarily chosen so that its dynamic behavior is interesting. For the choice of weights used in this simulation, the responses for step inputs of 0.1 unit and 1 unit are shown in Figs. 7 and 8. It can be seen that the response for a 0.1 unit step enters a limit cycle while that for an unit step is well damped. Also note that the DC gain of the system is negative.

For the model, a similar network is used. Also we assume that full state information is not available and hence the learning situation is similar to that of the recurrent state model.

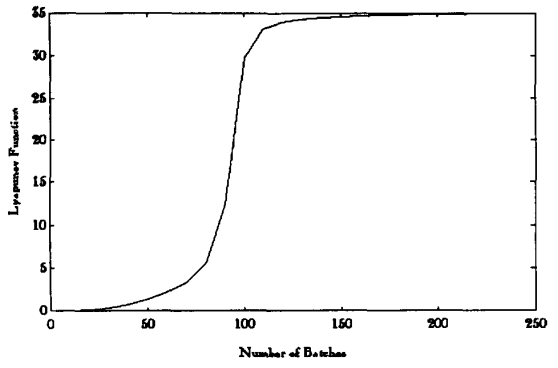



Fig. 9. Direct adaptation.

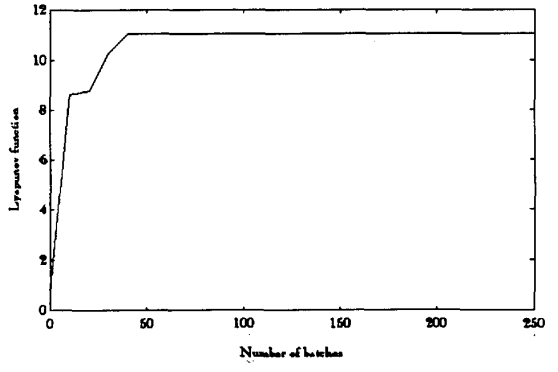



Fig. 10. Sensitivity method—forward.

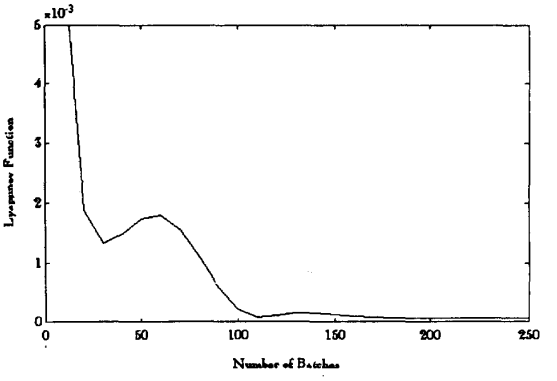



Fig. 11. Sensitivity method—backward.

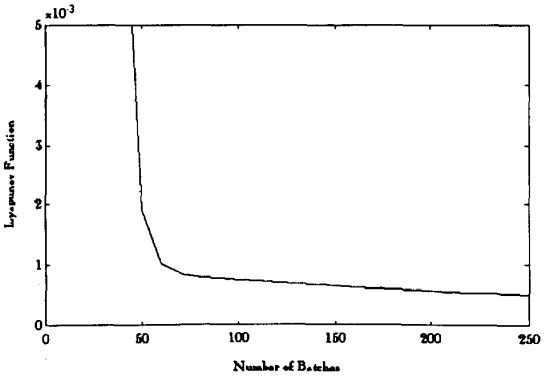



Fig. 12. Predictive back propagation.

During adaptation the weights are kept bounded by using a hard saturation on the weights. The initial conditions of the model are chosen randomly and the input to the plant is a random signal of length 50 repeated over and again. The update is done in batches with the batch length "$M$" being 50. The number of back propagation steps, "$N$," is chosen to be the order of the system, which is 2. Since the number of weights in this network is 16, to uniquely define it amidst second order dynamics the batch size of $50 > 16(2 + 1)$ was chosen. For a proper comparison of various schemes, the plant, the initial conditions, the learning rate, and the input to the system are kept the same in all simulation trials. The comparison is made by studying the time behavior of the fitness criterion, which is a Lyapunov function over different batches.

Since the dynamics is quite significant, the direct update scheme is incapable of decreasing the Lyapunov function as shown in Fig. 9. The Lyapunov function saturates at a high value due to the boundedness of the sigmoid nonlinearity. The sensitivity method implemented in the forward sense also does not converge under the conditions of the present simulation (Fig. 10). The weights start blowing up, only to be limited by their saturation limits. This may be due to the fact that once the summation for the sensitivity calculation starts increasing in length, the step size used is too large to assure a decrease in cost. However, the backward implementation of sensitivity method converges. Though it is capable of eventually reducing the Lyapunov function, it does not do so monotonically (Fig. 11). The extent of the increase of the Lyapunov function in the divergent phases keeps reducing with time. This can be attributed to the fact that, as the changes in weights keep reducing with time, the constancy of weights assumption required for the sensitivity method is met with increasing fidelity. Ultimately, its performance is comparable to that of other methods.

For the other three schemes (Figs. 12, 13, and 14) monotonic decrease of the Lyapunov function is observed indicating that these schemes are globally convergent. Monotonic convergence in the predictive and targeted back propagation cases means that for this system the state error is reduced automatically and the spill-over across batches is not large enough to cause instability. Hence filtering is not very essential and this fact can be used to reduce the computational burden. However the convergence is faster with filtering and is the best among all schemes discussed. The convergence properties of the Targeted scheme lie in between those of the Predictive schemes with and without filtering, since the weight update of the Targeted scheme is accomplished only indirectly through filtering.

As a second example, illustrative of how control problems can be attacked using the algorithms reported here, we consider

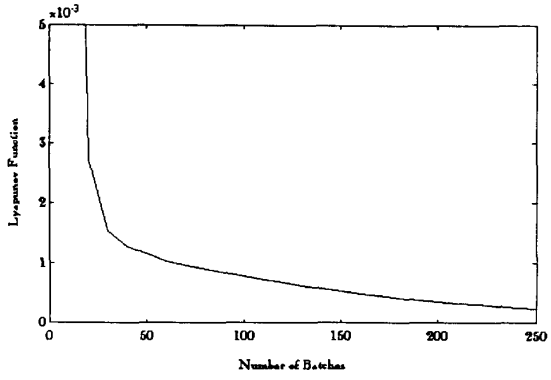



Fig. 13. Targeted back propagation.

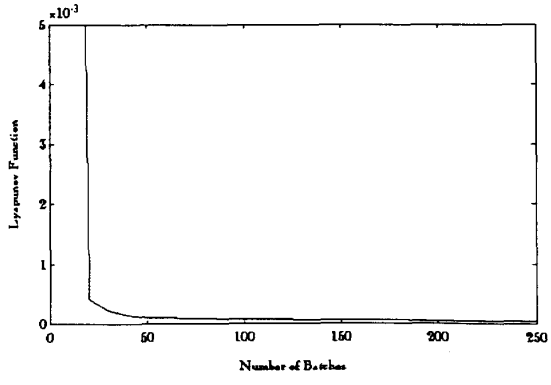



Fig. 14. Predictive back propagation with filtering.

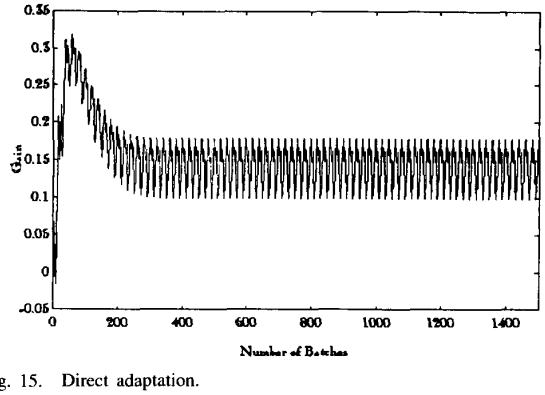



Fig. 15. Direct adaptation.

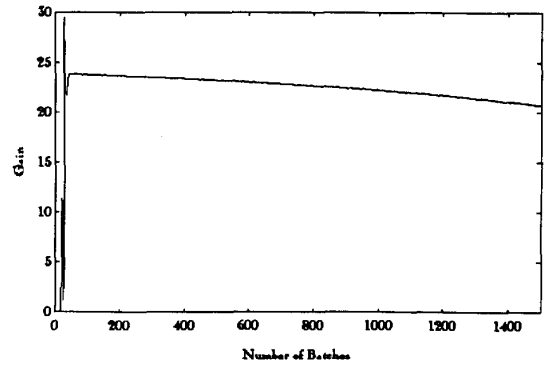



Fig. 16. Sensitivity method—forward.

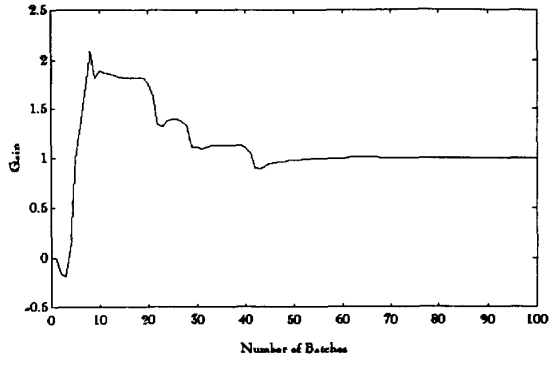



Fig. 17. Sensitivity method—backward.

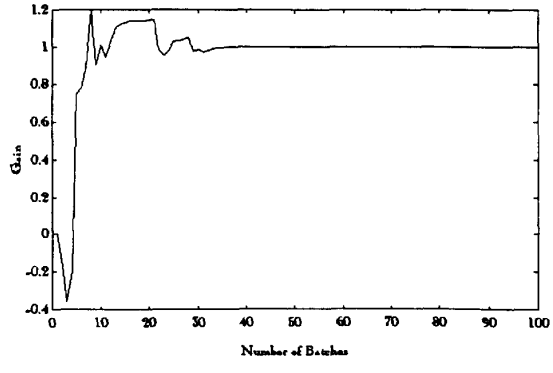



Fig. 18. Predictive back propagation.

an unit feedback system around the same plant with an unity gain proportional controller. In the simulation we assume that the system configuration and the plant dynamics are known while the gain of the controller is unknown. The gain has to determined from the reference input and the plant output. The derivative of the cost function with respect to the controller gain is obtained by back propagating the error through the adjoint of the plant. The time evolution of the estimated gain in the various schemes are shown in Figs. 15 to 20. The direct adaptation leads to a limit cycle, while the forward implementation of the sensitivity method initially overshoots and converges very slowly. The convergence of the backward implementation of the sensitivity method is quite fast but is characterized by a large overshoot. Barring a few differences, the convergence characteristics of the Predictive and Targeted back propagation schemes are similar. With filtering, the convergence is faster with no overshoot.

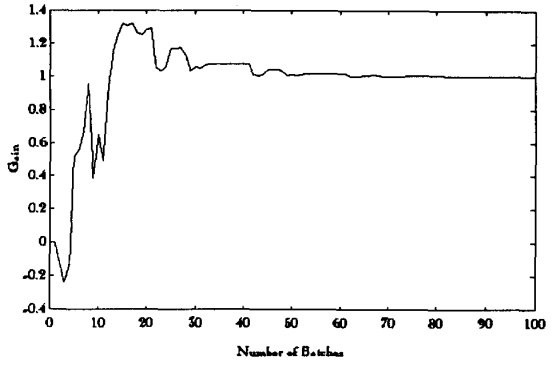



Fig. 19. Targeted back propagation.

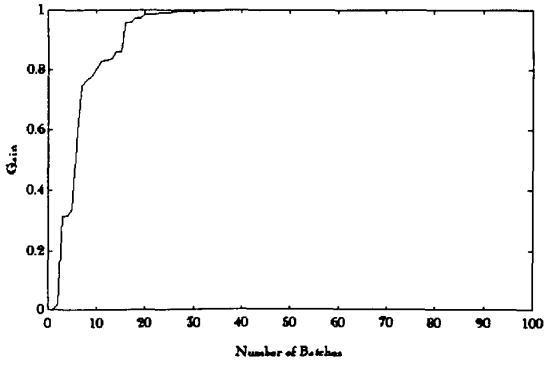



Fig. 20. Predictive back propagation with filtering.

## VIII. Conclusion

In this paper, it was shown that for identification of general nonlinear dynamic systems the use of recurrent models is inevitable. In cases where the model is an interconnection of dynamic subsystems, it was illustrated that the adjoint model can be used to calculate the gradient. To update the weights of the network using the gradient various schemes were proposed with increasing levels of computational complexity. The convergence of predictive back propagation with filtering was established. As a preamble, convergence issues in feedforward models was analyzed using a Lyapunov approach.

The update schemes presented here are directly extendable to any interconnection of dynamic subsystems. So, the adaptive control problem can be solved using the algorithms reported here by casting it as a problem of identification with an interconnected model. Though the algorithms discussed here are computationally more expensive compared to their feedforward counterparts, they are competitive in comparison with the algorithms available in the current literature for recurrent models. An increase in computational complexity is naturally expected due to the dynamics and is inevitable if the dynamics encountered is significant.

## References

[1] R. P. Lipmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine* vol. 2, pp. 4–22, Apr. 1987.

[2] R. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, "Neural networks for control systems—A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[4] S. Chen and S. A. Billings, "Nonlinear system identification using neural networks," *International Journal of Control*, vol. 51, pp. 1191–1214, 1990.

[5] P. J. Werbos, "Back propagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct. 1990.

[6] A. G. Barto, R. S. Sutton and C. W. Anderson, "Neuron-like adaptive elements that can solve difficult learning control problems," *IEEE Trans. System Man, Cybernetics*, vol. 13, pp. 834–846, 1983.

[7] M. Kawato, Y. Uno, M. Isobe and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE control systems magazine*, vol. 8, no. 2, pp. 8–16, 1988.

[8] D. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE control systems magazine*, vol. 10, no. 3, pp. 18–23, Apr. 1990.

[9] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 252–262, 1991.

[10] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press, ch. 8, vol. 1: Foundations, pp. 318–362.

[11] P. J. Werbos, "Beyond regression: new tools for prediction and analysis in behavioral sciences," *Ph.D Thesis*, Harvard University, 1974.

[12] G. Cybenko, "Continuous value neural networks with two hidden layers are sufficient," *Math. Control Signals and Systems*, vol. 2, pp. 303–314, 1989.

[13] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[14] J. B. Cruz Jr., Ed., *System Sensitivity Analysis*, Stroudsburg PA: Dowder, Hutchinson and Ross, 1973.

[15] R. J. Williams and D. Zipser, "A Learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[16] K. S. Narendra and L. E. McBride. Jr., "Multi-parameter self-optimization using correlation techniques," *IEEE Trans. Auto. Contr.*, vol. AC-9, pp. 31–38, 1964.

[17] B. Srinivasan, U. R. Prasad and N. J. Rao, "Improved back propagation methods for identification of nonlinear dynamical systems using neural networks," *IJCNN92*, vol. 2, pp. 59–63, Beijing, 1992.

[18] P. Zarchan, "Strategic and tactical missile guidance," vol. 124, *AIAA Publication* 1990.

[19] A. E. Bryson and Y. C. Ho, *Applied Optimal Control: Optimisation, Estimation and Control.* Blaisdel Publishing Company, 1969.

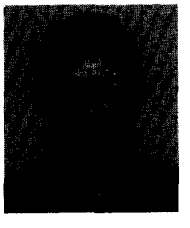
[20] S. W. Director and R. A. Rhorer, "The generalised adjoint network and network sensitivities," *IEEE Transaction Circuit Theory, CT-16*, pp. 318–323, 1969.

[21] L. Ljung, *System Identification: Theory for the User.* Prentice Hall, 1987.

[22] R. Fletcher, *Practical Methods of Optimization.* Vol. 1, New York: J. Wiley & Sons, 1980.

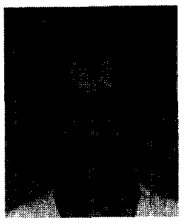
[23] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

[24] R. C. K. Lee, "Optimal estimation, identification and control," Research monograph 28, Cambridge, MA: MIT Press, 1964.



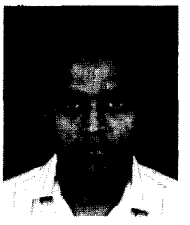
**Balasubrahmanyan Srinivasan** received the B.E. degree in Electronics and Communication Engineering from the Bharathiar University, Coimbatore, India in 1988. He received the M.Tech. in Electronics Design and Technology from the Indian Institute of Science, Bangalore, India in 1990, where he is currently pursuing the Ph.D. degree in the Department of Computer Science and Automation.

He spent an year in the Control Laboratory at the Swiss Federal Institute of Technology, Lausanne, Switzerland. His research interests include adaptive and optimal control of nonlinear systems, neural networks, and motion control.



**Upadrasta Ravikirana Prasad** (M'68) was born in Guntur, India in 1944. He received the B.Sc. degree in Physics and the B.E. (Hons.) degree in Electrical Engineering from Andhra University, Visakhapatnam, India, and the Ph.D. degree from the Indian Institute of Technology, Kanpur, India, in 1960, 1965 and 1970 respectively.

He was a research associate at the Indian Institute of Technology, Kanpur, India during 1970–71. In 1972, he joined the Indian Institute of Science, Bangalore, India where he is currently a Professor in the Department of Computer Science and Automation. He was a Nuffield Fellow at the Department of Electronic System Design, Cranfield Institute of Technology, Cranfield, England during 1979–80, and a Visiting Scientist at the Institute of Flight Systems Dynamics, German Aerospace Research Organization, Oberpfaffenhofen, Germany during 1987–88. His research interests include control systems, differential games, neural networks, and applications.



**Nalam Jaganmohan Rao**, (S'68–M'70), received the B.E. degree in Telecommunication Engineering from Andhra University, Visakhapatnam, India in 1964, the M.Tech. degree in Industrial Electronics from the Indian Institute of Technology, Bombay, India in 1966 and the Ph.D. degree in Control Theory from the Indian Institute of Technology, Kanpur, India in 1972.

In 1972, he joined the School of Automation at the Indian Institute of Science, Bangalore, India where he is currently a Professor. Since 1981, he is the Chairman of the Centre for Electronics Design and Technology at the same Institute. His areas of interest are digital systems, motion control systems and engineering education.