A98-37196 AIAA-98-4510

AN INFORMATION FILTER APPROACH TO RAPID SYSTEM IDENTIFICATION: CONVERGENCE SPEED AND NOISE SENSITIVITY

David C. Hyland
Department of Aerospace Engineering
The University of Michigan, Ann Arbor, MI

Keith K. Denoyer
Air Force Research Laboratory,
Space Vehicles Directorate
Kirtland AFB, NM

Abstract

In the development of neural network-based systems for the autonomous identification and control of space platforms, there remain important issues associated with the avoidance of inordinately slow convergence. Focusing here on autonomous identification of space systems, one can investigate techniques for faster convergence following two basic strategies. One strategy involves the use of a Principal Component Analysis (PCA) algorithm to transform the regressor vector. An alternative strategy, and the one pursued in this paper, seeks to improve the adaptation speed of neural adaptive identifiers by means of a basic revision of the fundamental learning mechanism. A neural network algorithm that affords parallel implementation of the information filter form of the Kalman filter has been found to eliminate the dependence of convergence behavior upon the correlation matrix of the net input. Analysis and computational studies show that this algorithm can produce many orders of magnitude improvement in convergence speed. A drawback is that the algorithm also tends to amplify the effect of sensor noise on the identification errors. However, detailed analysis of noise sensitivity shows that the solution is to combine this approach with the PCA procedure. Numerical studies, confirm that when this is done, the combined system achieves very rapid convergence with low noise sensitivity.

1. Introduction

The pursuit of a higher degree of autonomous behavior that provides constant health monitoring and fault tolerance for space systems with minimum human intervention has motivated the development of autonomous neural network controllers based solely on on-board instrumentation, that are capable of self-optimization, on-line adaptation, and autonomous fault detection and controller reconfiguration [1]. As part of this development, a processing architecture for neural algorithms in identification (ID) and control [2] was devised some years ago and the basic capabilities of this architecture have been demonstrated experimentally. For example, in the Adaptive Neural Control Program for the USAF Phillips Lab, autonomous control algorithms for vibration suppression were demonstrated on the ASTREX testbed [3,4]. These experiments showed both the autonomous convergence to a high performance controller as well as automatic controller reconfiguration/recovery following the disengagement (simulated failure) of randomly selected subsets of the actuator hardware units.

The above advances and testing experiences directly indicate that one of the most important issues in autonomous neural net-based system identification is the attainment of predictably rapid convergence. It can often happen in practice that LMS-style identifiers exhibit inordinately slow convergence in a manner that depends in a very complicated way on both the system and identifier parameters. In the next section, we illustrate this phenomenon using a very simple and apparently benign example involving just three lightly damped structural resonances. This example also serves to show how slow convergence is connected with ill-conditioning of the correlation matrix of the identifier inputs.

In [5] we considered one technique for achieving more rapid convergence in connection with a standard form of our series-parallel, ARMA model neural identifier. In this approach we used a variant of the Principal Component Analysis (PCA) algorithm [6,7] to transform the regressor vector inputs to render their components mutually uncorrelated and of equal variance. With the conditioning of the correlation matrix thus improved, we showed orders of magnitude increase in the convergence speed.

In this paper, we consider an entirely complementary approach. While the idea of the earlier PCA approach was to transform the regressor vector we now consider the transformation of the weight increment given by the customary backpropagation update formula. We show that this can be done in a way that eliminates the deleterious dependence of the convergence rate on the correlation matrix of the regressor.

In section 3, we devise a recursive algorithm for the automatic generation of the matrix associated with the above weight increment transformation. This algorithm comprises entirely parallel computations and can be readily implemented via a neural network with a

Hebbian learning rule for adjusting the weights. The algorithm can also be viewed as the information filter form of the Kalman filter which adheres to a purely parallel computational approach. In the system identification examples we have investigated, this algorithm , called here the *information filter algorithm*, exhibits qualitatively superior convergence speed as compared even with the PCA approach. Specifically, for the "three-mode " example mentioned above, convergence to very high accuracy can be obtained in only twelve time steps (0.06 second at 200 Hz sample rate). Further studies, however, showed that the algorithm entails heightened sensitivity to sensory input noise. When inevitable instrumentation noise corrupts the measurement inputs, any adaptive system identifier will exhibit both fluctuating and steady state bias errors in the identified coefficients. Ideally, the ID algorithm should be able to attenuate the magnitudes of both error components. Unfortunately, the fast adaptive algorithm tends to amplify the error components due to measurement noise. However,the detailed analysis presented in Section 4 shows that the noise amplification is inversely proportional to the smallest eigenvalue of the regressor correlation matrix. Thus, the same ill-conditioning of the this matrix that causes slow convergence for standard LMS-style identification algorithms is also the source of noise sensitivity for the fast adaptive algorithm. This immediately suggests the solution: Combine the PCA and the present adaptive approaches. Specifically, we first apply a PCA network to transform the regressor, then with the new regressor, transform the weight increment using the information filter algorithm. Our simulation results on a number of examples in Section 4, including the three-mode example, confirm that the combined system gives the same rapid convergence as the information filter algorithm while eliminating its sensitivity to measurement noise.

2. Basic System Identification Algorithm and Convergence Speed Issues

In order to more clearly motivate the subject of this paper, we first discuss the nature of convergence speed issues encountered with backpropagation-based neural networks or similar LMS-style adaptive signal processing schemes. Moreover the issues with which we shall be mainly concerned do not depend upon whether or not the control system or identifier is linear or nonlinear. Therefore, for simplicity and to focus ideas, we consider the problem of identifying a linear plant. The structure of the neural identifier depends upon the model form with which the plant is represented. To address the most popular model form, suppose the output, y(k), of a linear SISO system with input x(k), can be represented by the ARMA model;

$$y(k+1) = Wp^T X(k) \quad (1.a)$$

$$X(k) = [y(k), y(k-1), \ldots y(k-N+1), x(k), x(k-1),\ldots x(k-N+1)]^T \quad (1.b)$$

where $(.)^T$ denotes the transpose, N is the order , X is the 2N dimensional "regressor vector", assumed to be measured and Wp is the vector of ARMA coefficients.

Now, given X, the simplest neural net for representing the above system, has an output in the form [2]:

$$z(k+1) = W(k)^T X(k) \quad (2)$$

where W(k) is the 2N dimensional weight vector. This is then subtracted from y(k+1) to obtain the output error:

$$e(k+1) = y(k+1)-z(k+1) \quad (3)$$

Following [2], the weight vector is updated according to:

$$W(k+1) = W(k) + \mu(k)e(k+1)X(k) \quad (4.a)$$

where:

$$\mu(k) = \alpha / \|X(k)\|^2 \quad (4.b)$$

Here, $\mu(k)$ is a time varying adaptive speed and α is a constant. The definition of μ allows us to choose the constant α once and for all and, under quite general conditions, guarantee convergence. These matters are considered in some detail in [2]. In particular, if there exist weight values such that z can duplicate y exactly and if the square of the norm of the error , considered as a function of the weights, is a homogeneous function of degree M, then $\alpha<M$ implies that e(k) converges to zero. In the above linear problem, the square of the error is obviously a quadratic function of the weights (M = 2). Hence, if y can be represented exactly by a system of form (2) for some value(s) of W, then $\alpha<2$ implies convergence of |e(k)|. Because it results in one-step convergence in the one-dimensional case, we most often use the value $\alpha = 1$.

Next, in order to say more about the rate of convergence, we need to recast (4.a) into a more suitable form. Define:

$$w(k) = Wp - W(k) \quad (5)$$

Then:

$$e(k+1) = w(k)^T X(k) \quad (6)$$

Using this expression in (4.a) and replacing W in favor of w, we get:

$$w(k+1) = [\, I - \mu(k)X(k)X(k)^T]w(k) \quad (7)$$

where I denotes the 2N dimensional identity. Equation (7) describes the evolution of the deviation, w, of the weight vector from the values that permit z to exactly match y. The behavior of the mean of w or the norm of w gives some notion of the rapidity of convergence of (4). There are very few exact results, particularly when α is of order unity (α not small). However, for small α, results such as those reported in [8] show that the

expected value of w(k) asymptotically approaches wa(k), where:

$$wa(k+1) = [\,I - \alpha R]wa(k) \qquad (8)$$

where:

$$R = E[X(k)X(k)^T / \|X(k)\|^2] \qquad (9)$$

We see that R (often called the "confluence matrix") is the second moment matrix of the normalized net input, X/ ||X||. The sum of the eigenvalues of R , trace(R), is unity. Thus the rates of geometric convergence have the form $(1 - \alpha\lambda)$, where λ is any one of the (non-negative) eigenvalues of R. We cannot make this factor small by increasing α indefinitely because, in general, $\alpha<2$ is necessary to ensure that W(k) is bounded. Thus, if R is ill-conditioned, i.e., has a set of small eigenvalues, then $|1 - \alpha\lambda|$ will be nearly unity, and convergence can be very slow. Equations (8-9) pertain to the case $\alpha << 1$, but it is still arguable that (8-9) gives the leading term in a equence of asymptotic approximations and adequately portrays the dominant factors in convergence speed even when α is of order unity.

Slowness of convergence is a problem that depends on a very complicated way on the system dynamic parameters, sample rate, the number of delays in the ARMA model, etc. The sensitivity of convergence speed to system parameters and the fact that slow convergence can even occur for relatively low order, apparently innocuous systems are well illustrated by the following "three mode example". This is a single-input, single-output system with just three lightly damped resonances - apparently a trivial example upon which to apply autonomous system identification. The discrete-time model is given by:

$$y(k+1)=[1, 0]x1(k+1)+[1, 0]x2(k+1)+[1, 0]x3(k+1) \qquad (10)$$

where:

$$x1(k+1) = \rho_1 \begin{bmatrix} c1 & -s1 \\ s1 & c1 \end{bmatrix} x1(k) + \begin{bmatrix} 0 \\ 2.0 \end{bmatrix} \chi(k) \qquad \text{a. (11)}$$

$$x2(k+1) = \rho_2 \begin{bmatrix} c2 & -s2 \\ s2 & c2 \end{bmatrix} x2(k) + \begin{bmatrix} 0 \\ 2.0 \end{bmatrix} \chi(k) \qquad \text{b.}$$

$$x3(k+1) = \rho_3 \begin{bmatrix} c3 & -s3 \\ s3 & c3 \end{bmatrix} x3(k) + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \chi(k) \qquad \text{c.}$$

Here, ck and sk (k = 1,2,3) denote $\cos(\theta_k)$ and $\sin(\theta_k)$, respectively. θ_k and ρ_k are parameters of a balanced modal representation. The values of these parameters are given by:

$$\theta_1 = 2\pi/50,\ \theta_2 = 2\pi/30,\ \theta_3 = 2\pi/10 \qquad (12.a,b,c)$$

$$\rho_1 = 0.995,\ \rho_2 = 0.995,\ \rho_3 = 0.990 \qquad (13.a,b,c)$$

Finally, in (11) χ (k) is a discrete time white noise process that is normally distributed with unit variance.

Given the above system, we construct a simulation T steps long in which the regressor vector, X, is assembled and the system identification algorithm, (2) - (4) is implemented. Although rigorous requirements for convergence are satisfied, a typical case with N=6, α =1 and T=1000 results in very little decline in the output error. Figure 1 shows the system output and the output error, e(k), versus time steps for these values of parameters and for an initially zero weight vector. Apparently, e(k) persists throughout the entire 1000 steps without any evidence of further improvement. For the same case, Figure 2 shows the time variation of the elements of the weight vector. Again, there is no appreciable tendency toward convergence. The standard algorithm, (2)-(4) *does* converge but it would require a simulation over a much larger time period to reveal this numerically.

For the above simulation of length T = 1000, the eigenvalues of R are found to be:

$$\lambda(R) = [2.910e\text{-}09\ \ 5.813e\text{-}06\ \ 3.373e\text{-}04\ \ 1.880e\text{-}03\ \ 1.812e\text{-}03\ \ 1.607e\text{-}03\ \ 2.539e\text{-}03\ \ 3.827e\text{-}03\ \ 4.189e\text{-}03\ \ 1.308e\text{-}02\ \ 2.311e\text{-}01\ \ 7.375e\text{-}01] \qquad (14)$$

In view of relations (8) and (9) and the eight orders of magnitude spread in the above eigenvalues, it is not surprising that convergence is glacially slow. Considering that this sort of phenomenon can occur for such a simple example it is vital to address the root cause of the problem in order to assure reliable operation of neural adaptive systems for applications of practical importance.

3. A Fast Adaptive Algorithm via an Information Filter: Advantages and Issues

In [5] we sought to improve convergence speed by transforming the regressor using PCA. Here we consider a complementary approach in which only the increment in the weights is transformed. The basic idea of the PCA work was to transform the regressor, X, replacing it by QX wherever it appears in (2)-(4). Thus W is adjusted according to:

$$W(k+1) = W(k) + \mu(k)QX(k)e(k+1) \qquad (15.a)$$

$$e(k+1) = y(k+1) - W(k)^T QX(k) \qquad (15.b)$$

where $\mu(k)$ is appropriately modified from its expression in (4.b). In contrast, the alternative considered here is to transform only W(k+1) - W(k). Hence, in place of (15):

$$W(k+1) = W(k) + \mu(k)Q(k+1)X(k)e(k+1) \qquad (16.a)$$

$$e(k+1) = y(k+1) - W(k)^T X(k) \qquad (16.b)$$

where, again, $\mu(k)$ is assumed to be given by expression (4.b) of the original formulation. In (16), e(k+1) is the same as the original quantity in (3), but the increment in W is different. As will be seen, Q(k+1) is not a constant matrix but is a recursively updated estimate of the inverse of the correlation matrix $E[XX^T]$.

At this point, we give a simple, heuristic derivation of what Q ought to be, based on two main considerations: (1) the choice of Q(k+1) should permit the fastest possible convergence for W and (2) the calculation of Q(k+1) should involve "parallelizable" computations suitable for neural network realization. To begin, defining w(k) as in equation (5), we have:

$$w(k+1) = [I - \mu(k)Q(k+1)X(k)\,X^T(k)]w(k) \quad (17)$$

Now a reasonable measure of the effectiveness of the learning algorithm in reducing the size of w (which represents the error in the identified parameters) is the magnitude of the component of w(k+1) along X(k) as compared with the component of w(k) along X(k). Taking the inner product of both sides of (17) with X(k), we get:

$$X^T(k)w(k+1) = [1 - \mu(k)\,X^T(k)Q(k+1)X(k)]\,X^T(k)w(k) \quad (18)$$

The quantity $[1 - \mu(k)\,X^T(k)Q(k+1)X(k)]$ gives the ratio of the projection of w(k+1) along X(k) to what it was one time step earlier and so we would like to make this as small as possible, particularly when μ is set to a large value. In other words, considering $\mu(k)X^T(k)Q(k+1)X(k)$ as a function of $\mu(k)$, we would like this quantity to be approximately unity for large values of $\mu(k)$. Small values of μ, on the other hand, signify large averaging times and slow changes in W. In this case, we would expect to see $X^T(k)Q(k+1)X(k)$ approximately equal to $X^T(k)Q(k)X(k)$ for small $\mu(k)$. The lowest order rational function of $X^T(k)Q(k)X(k)$ that we can form for $X^T(k)Q(k+1)X(k)$ that has the desired behavior at large and at small values of μ (k) is:

$$X^T(k)Q(k+1)X(k) = X^T(k)Q(k)X(k) \,/\, [1+\mu(k)\,X^T(k)Q(k)X(k)] \quad (19)$$

Next, (19) implies that for large μ, the component of Q(k+1) along X(k) is reduced to nearly zero. Thus a rule for getting Q(k+1) that is consistent with (19) is to subtract off from Q(k) the "component" of Q(k) that is aligned along X(k). A little reflection shows that this component is $Q(k)X(k)(Q(k)X(k))^T$. Thus we form Q(k+1) according to:

$$Q(k+1) = Q(k) - d(k)Q(k)X(k)X^T(k)Q^T(k) \quad (20)$$

where d(k) is some nonnegative scalar. Note that formula (20) entails only a parallel type of computation. Indeed, the (k,j)th element of Q(k+1)-Q(k) is proportional to the product of the kth element of Y=Q(k)X(k) and the jth element of Y. This is very much analogous to a Hebbian learning rule. To determine what d(k) ought to be, we determine the product $X^T(k)Q(k+1)X(k)$ from (20):

$$X^T(k)Q(k+1)X(k) = X^T(k)Q(k)X(k)[1 - d(k)X^T(k)Q(k)X(k)] \quad (21)$$

and compare the right hand side with (19) to get:

$$d(k) = \mu(k) / (1 + \mu(k)X^T(k)Q(k)X(k)) \quad (22)$$

Finally, we initialize Q so that at the initial time it corresponds to the original algorithm (wherein there is no transformation)- i.e. Q(0) = I. Putting all these relationships together, we have, in summary:

$$Q(k+1) = Q(k) - d(k)Q(k)X(k)X^T(k)Q(k)^T, \quad Q(0) = I \quad (23.a, b)$$

$$d(k) = \mu(k) / (1 + \mu(k)X^T(k)Q(k)X(k)) \quad (23.c)$$

$$W(k+1) = W(k) + \mu(k)Q(k+1)X(k)e(k+1) \quad (24.a)$$

$$\mu(k) = \alpha / \|X(k)\|^2 \quad (24.b)$$

(23)-(24) constitute the simplest parallel computation-based algorithm that can yield very rapid convergence of W. Below, we show that this is also an optimal filter when one assumes a noise model in which additive sensor noise is uncorrelated with the regressor vector. Unfortunately this noise model does not accord with reality, as will be explained. However, at present, we provide detailed analysis of the convergence behavior of the above algorithm.

Consider the analysis of (23). We can use the "tearing" formula [9] on (23.a,b) to obtain:

$$Q(k+1) = (Q^{-1}(k) + \mu(k)X(k)\,X^T(k))^{-1}, \quad Q(0) = I \quad (25.a, b)$$

Examining the above equation, it is easily deduced that:

$$Q(k) = \left(I + \sum_{m=0}^{k-1} \mu(m)X(m)X^T(m)\right)^{-1} \quad (26)$$

Next, using (24.a) to evaluate w(k+1), we get:

$$w(k+1) = w(k) - \mu(k)Q(k+1)X(k)X^T(k)w(k) \quad (27)$$

Then, using expression (26) for Q(k+1) in the right hand side of (27), we deduce that:

$$Q^{-1}(k+1)w(k+1) = Q^{-1}(k)w(k) \quad (28)$$

This shows that $Q(k)^{-1}\,w(k)$ is a constant matrix. Setting k=0 and noting that Q(0)=I, we find that:

$$w(k) = Q(k)w(0) \quad (29)$$

Hence, in summary, we can write:

$$Q(k) = \left(I + \sum_{m=0}^{k-1} \mu(m)X(m)X^T(m)\right)^{-1} \quad (30)$$

$$w(k) = \left(I + \sum_{m=0}^{k-1} \mu(m)X(m)X^T(m)\right)^{-1} w(0) \quad (31)$$

where, again, $\mu(m)$ is given by (24.b).

From (30)-(31), it is clear that Q and W converge for all positive α, not just for α's bounded by some number of order unity. In further contrast with the previous weight adjustment schemes, (23)-(24) are particularly effective for very large values of α, say α = 1e12. Consider such large values of α and examine the

convergence of ||w(k)||. Looking at the situation at k=1, and letting:

$$x(k) = X(k) / \|X(k)\| \qquad (32)$$

we see from (31) that

$$|w(1)^T x(0)| / |w(0)^T x(0)| = 1 / [1+\alpha]$$

Thus, if $w_{||}(k)$ is $w(k)^T x(0)$ and $w_{\perp}(k)$ is the component of w(k) in the subspace orthogonal to x(0), then:

$$\|w(1)\|^2 = (1 / [1+\alpha])^2 (w_{||}(1))^2 + \|w_{\perp}(0)\|^2$$
$$\approx \|w_{\perp}(0)\|^2 \quad \text{for large } \alpha$$

as compared with the initial value:

$$\|w(1)\|^2 = (w_{||}(1))^2 + \|w_{\perp}(0)\|^2$$

Proceeding to examine k=2, 3, etc. we see that every time the span of {x(0), x(1), ..., x(k)} effectively increases in dimension, the norm of w(k) and hence the magnitude of e(k+1) (since $|e(k+1)| \le \|w(k)\| \|X(k)\|$) undergoes an approximately step - change reduction. This continues until the maximum dimension of span{x(0), ...,x(k)} is reached. If X(k) is persistently exciting so that for some k>Te, span{x(0), ...,x(k)} is of dimension 2N, then from (31):

$$\|w(k)\| / \|w(0)\| \le 1 / \sqrt{1 + \alpha\, gk}$$
$$\approx 1 / \sqrt{\alpha\, k} \qquad (33)$$

where g is a positive scalar of order unity. Thus the magnitude of e(k) eventually declines as $1/\sqrt{k}$.

To summarize: (31) shows that the magnitude of the output error at first undergoes nearly step - change reductions, and, in the case of persistent excitation, eventually is reduced to something of order $1/\sqrt{\alpha}$ and thereafter declines in inverse proportion to the square root of time. Thus, the convergence of (23)-(24) tends to be faster than exponential at first but much slower (in proportion to $1/\sqrt{k}$) over the long term.

The above remarks can be verified by simulations implementing (23)-(24) on the above three-mode example with: N = 6 and α = 1e12 and assuming zero additive sensor noise. Figure 3 shows a plot of the system output and the identifier error versus time for this case. It is seen that the error, e(k), becomes very small after only about 12 ~ 2N steps. On the other hand, plots of the elements of W given in Figure 4 show that the weights settle down only after about 40 steps. Notice that each component of W seems to execute roughly step changes with intervening periods of nearly constant values. Finally, a semilog plot of |e(k)| over a longer time interval shown in Figure 5 reveals the convergence behavior described above - the error magnitude decreases from order unity at t=0 to under 1e-4 in about 12 time steps but thereafter declines slowly, behaving as $1/\sqrt{t}$ over the long term.

It would seem that the convergence behavior of the algorithm (23)-(24) is quite spectacular - convergence in a dozen steps on an example problem that completely defeated the original adaptation algorithm. However there are two fundamental problems. First, although very fast at first, the new algorithm "falls asleep" after an initial period, i.e. assumes very slow response to subsequent perturbations. The second major difficulty is that the algorithm is very sensitive to sensor noise. In the following we discuss both these problems and their potential remedies in some detail.

The problem of slow subsequent response is understood by noticing the similarities between (24.a) and the original adaptation formula, (4.a). At least past time Te, when Q becomes full rank, the role of the adaptive speed, $\mu(k)$, in (4.a) is played by the quantity $\mu(k)Q(k+1)$ in (24.a). Overall, the effective adaptive speed in (24.a) is $\mu(k)\|Q(k+1)\|$. But noting the similarity of (30) and (31), we see that analogous to (33), for k>Te:

$$\|Q(k)\| \le 1 / \sqrt{1 + \alpha\, gk} \qquad (34)$$

Thus the behavior of (24.a) is analogous to that of (4.a) with an adaptive speed that continually decreases with time. This means that the speed of response of the identifier to system parameter changes becomes slower and slower: the algorithm "falls asleep". To see an illustration of this, one can run the three mode example simulation for a few hundred steps, allowing the identifier to completely converge, then introduce a step change in one of the parameters at ,say, k = 200 and watch the algorithm attempt to re-converge. The second reconvergence occurs much more slowly that the first. Any subsequent repetitions of parameter perturbations result in slower and slower response. Thus, in contrast to the original algorithm, the behavior of (23)-(24) has a unique time reference and the identifier response is not equally rapid at all times.

From what has been described above, if Q(k) is reset to identity at k = k* then W(k) enters into its faster-than-exponential convergence behavior immediately following k*. Hence the above problem can be rectified by some kind of "attentional algorithm" which resets Q to identity in (23) whenever the identifier response is deemed too slow. For example, the algorithm could monitor the background noise levels and detect when the output error rises above a certain critical threshold relative to the estimated noise level and then reset Q to I. A similar algorithm and convergence problem was treated in our previous work on adaptive tonal noise cancellation for the ASTREX testbed [3,4], so we have confidence that successful "attentional" algorithms can be devised in the present case. For the present, we now consider the second and far more serious problem with (23)-(24): the sensitivity of the algorithm to sensor noise.

To assess the effects of sensor noise, we modify (1.a) to read:

$$y(k+1) = Wp^T X(k) + v(k) \qquad (35)$$

where v(k) is a zero mean random time series with statistics independent of the disturbance noise exciting the system being identified. v(k) is normally distributed white noise with:

$$E[v(k)] = 0, \quad E[v(k)^2]) = \sigma^2 \qquad (36.a, b)$$

To illustrate the sensitivity of (23)-(24) to v(k), we compute the three mode system and identifier response and assuming N=6, α=1e12 and T = 1000 and various values of σ. In each run, convergence of W occurs for k < 50. Therefore, in each case, we compute steady state statistics of various quantities using the portions of the simulations involving k>=100. There are various steady state statistics of interest. For example, consider the output error minus the term directly due to sensor noise:

$$er(k) = e(k) - v(k) \qquad (37)$$

The standard deviation of this is a measure of how much sensor noise leaks through to produce a steady state fluctuation in the estimated system output. Also, consider the difference between the converged value of W and its "exact" value, Wp (which one can determine directly from results for σ = 0). In each case, we take the converged value of W to be W(1000) or W(T). The norm of W(T)-Wp relative to the norm of Wp gives a measure of the permanent bias error incurred by the identifier because of sensor noise. Calculating these quantities for various values of σ, we get the following results:

Table 3.1: Steady state fluctuation and bias errors for algorithm (23)-(24)

σ	$\sqrt{E[(er)^2]}$	$\|W(T)-Wp\| / \|Wp\|$
1.0e-4	2.899e-3	1.609e-2
5.0e-4	1.374e-2	1.239e-1
1.0e-3	2.446e-2	5.419e-1
5.0e-3	5.162e-2	0.9854
1.0e-2	1.397e-1	0.9868
5.0e-2	3.203e-1	0.9971
1.0e-1	4.261e-1	1.0032

The above table shows that, besides the direct noise term , e(k) exhibits noise induced fluctuations that are approximately an order of magnitude larger than the sensor noise. Not only is there this amplified fluctuating component of error , the estimated model parameters show very large, permanent bias errors. As the right column of the Table shows, the norm of the bias error becomes comparable to the exact values for σ = 1.0e-3 and above. Clearly, algorithm (23)-(24) has difficulties in the face of even modest sensor noise.

The above results are disappointing particularly in view of the fact that under certain conditions, (23)-(24) constitute an optimal Kalman filter for estimating Wp. In fact, (39) below presents the *information filter* form of the Kalman filter. We have the following result.

Theorem

Suppose that Wp is a random vector with elements normally distributed such that $E[Wp] = Wp_0$ and $E[(Wp-Wp_0)(Wp- Wp_0)^T] = Q_0$ and consider:

$$y(k+1) = Wp^T X(k) + v(k) \qquad (38)$$

where X(k) is a known 2N dimensional vector time series and v(k) is a white Gaussian process with zero mean and variance σ^2 that is statistically independent of X(k). Then the least mean square estimation of Wp, call it W(k), is determined according to:

$$Q(k+1) = (Q^{-1}(k) + (1/\sigma^2)X(k)X^T(k))^{-1}, \quad Q(0) = Q_0 \qquad (39.a, b)$$

$$W(k+1) = W(k) + (1/\sigma^2)Q(k+1)X(k)(y(k+1) - W(k)^T X(k)) \qquad (40.a)$$

$$W(0) = Wp_0 \qquad (40.b)$$

Proof: This is obtained from the more general result shown in [10] by specializing to the case $x(k) \rightarrow Wp$, w(k)=0, $C(k) \rightarrow X(k)$, A(k)=I and $V(k) = \sigma^2$. ❑

(39)-(40) closely resemble (23)-(24) except that $\mu(k)$ is replaced by $1/\sigma^2 >> 1$. The above result may help to explain the rapid convergence capability of (23)-(24) in the absence of noise, but the theorem is actually inapplicable. It is inapplicable because the regressor vector depends on the measurements, making the resulting filter a nonlinear function of the measurements. The Kalman filter is derived assuming that the sensitivity matrix (regressor vector) is a deterministic function, and results in the filter being a linear function of the measurements.

In the next section, we devise various ways to alleviate the noise sensitivity of (23)-(24). We also give a careful analysis of the causes of this noise sensitivity, showing clearly the role that is played by the correlation between v and X. The results of this analysis also suggest a fundamental way to eliminate the noise sensitivity of algorithm (23)-(24).

4. Information Filter Algorithm: Reduction of Noise Sensitivity

Here we consider various ways to alleviate the noise sensitivity of the information filter algorithm. First we examine various "quick fixes" that seem plausible but turn out to be unsuccessful. Next, we examine the underlying reasons for the algorithm's

noise sensitivity by conducting a detailed analysis. The analysis points the way to a specific approach that appears to remedy the problem, allowing us to exploit the rapid convergence properties of the algorithm without suffering undue noise sensitivity.

One possible source of the noise sensitivity that comes to mind is the scaling of $\mu(k)$ by the square of the norm of X(k) in equation (24.b). When the norm of W is small, this scaling makes the adaptive speed large, thereby emphasizing the importance of regressor vectors that contain a relatively large component of noise. We can readily test this hypothesis by trying simulations wherein (24.b) is replaced by $\mu(k)= \alpha$. Let us define steady state fluctuation and bias errors as in the last section, including er(k). With only the μ formula modified, and identical conditions as were assumed to compute the results in Table 3.1 we find that the error statistics are practically identical to the earlier case and the overall behaviors of the output error and W as functions of time, while different in detail, are qualitatively the same. These and similar results for other cases would seem to indicate that the scaling of $\mu(k)$ with $\|X(k)\|^2$ is not the source of the noise sensitivity of the algorithm.

A second approach to reducing noise sensitivity is motivated by the observation that algorithm, (23)-(24) is a gradient descent algorithm using the instantaneous gradient of the square of the error. Since instantaneous gradient values are used, one is continually injecting into the weight updates the disruptive disturbances associated with sensor noise, without any form of smoothing or averaging. This suggests that one way to reduce noise sensitivity is to devise a "batch processing" or time averaged version of the original algorithm. Following the same rationale as was used to derive (23)-(24) we can define such a time averaged algorithm as follows.

In the time averaging version, both W and Q are held constant over each time averaging period of duration Tav and are not updated except at the end points of each period. Let "n" denote the index of the averaging period and the corresponding update of Q and W. Then at the end of each interval of Tav time steps, we update W and Q according to:

$$Q(n+1) = (Q^{-1}(n) + \mu(n) \sum_{r=k-Tav+1}^{k} X(r)\, X^T(r))^{-1} \quad (41)$$

$$W(n+1) = W(n) + \mu(n)Q(n+1) \sum_{r=k-Tav+1}^{k} e(r+1)X(r)) \quad (42)$$

where the starting value of Q is still identity, e(r+1) is given by:

$$e(r+1) = y(r+1) - W(n)^T X(r) \quad (43)$$

and

$$\mu(n) = \alpha \,/\, \sum_{r=k-Tav+1}^{k} X^T(r)X(r) \quad (44)$$

Here, (41) and (42) replace (25.a) and (24.a), respectively. e(k) is actually evaluated just as before, except that W is kept constant every Tav time steps. Note that μ is now scaled by the sum of the magnitudes squared of X over the previous Tav time steps. The above formulae are tantamount to a gradient descent using the gradient of a mean square error estimate involving the previous Tav time steps. Obviously, (41) and (42) involve the processing of prior data over Tav - sized batches and for Tav sufficiently large, this algorithm guarantees a great deal of smoothing.

The above is a fairly direct extension of the algorithm introduced in the last section. In particular, when Tav = 1, the algorithm reduces back to the original algorithm. For Tav > 1, we can still prove results analogous to those demonstrated above. For example, if as in Section 2, we define w(n) as Wp - W(n), then (41)-(44) imply:

$$Q^{-1}(n+1)w(n+1) = Q^{-1}(n)w(n) \quad (45)$$

- a result analogous to (28). Similarly, we may work out explicit formulae for w(n) and Q(n) similar to (30)-(31). It follows from all this that the modified algorithm, what we might call the "batch" version of the information filter algorithm, has analogous convergence behavior. Convergence is secured for all real positive values of α, the error norm converges to zero at a faster than exponential rate at first and then at a much slower rate, proportional to $1/\sqrt{k}$. Unlike the original algorithm, however, the initial, very rapid, period of convergence is not of order N time steps, but rather requires approximately NTav steps. This is the price one pays for additional time averaging.

If we use the above and run simulations using exactly the same parameter values as in the studies for Tables 3.1 we find that the new algorithm still retains the same large levels of fluctuation and bias error. Indeed, our numerical study indicates that the effect of larger averaging times is to make the situation progressively worse. Thus, the new, "batch" version of the algorithm does not succeed in reducing the sensitivity to additive noise in the output. The reason for this paradoxical situation can only be found through a careful analysis of the effect of noise on the output error in the original algorithm.

Return now to (23), (24) and (25) with Y(k) corrupted by sensor noise as in (35). The output error is now given by:

$$e(k+1) = w(k)X(k) + v(k) \quad (46)$$

where:

$$w(k) = Wp - W(k) \quad (47)$$

Using these expressions in (24), we get the following equation describing the evolution of w(k):

$$w(k+1) = w(k) - \mu(k)Q(k+1)X(k)(X^T(k)w(k) + v(k)) \quad (48)$$

At this point, we recall initial discussion at the beginning of this section that pointed out that a constant μ did not significantly affect the steady state noise sensitivity or bias error. Therefore, for simplicity in our noise sensitivity analysis we assume:

$$\mu(k) = \alpha \quad (49)$$

Note that the additional term, v(k), in e(k+1) does not affect (23) or (25). Therefore, one may use (25.a) to demonstrate :

$$Q^{-1}(k+1)w(k+1) = Q^{-1}(k)w(k) - \mu(k)X(k)v(k) \quad (50)$$

From this and the initial condition, Q(0) = I, we can deduce that:

$$w(k) =(I + \alpha \sum_{m=0}^{k-1} X(m) X^T(m))^{-1} [w(0) - \alpha \sum_{m=0}^{k-1} X(m)v(m)] \quad (51)$$

This generalizes (31) for the sensor noise case.

Next, using the definition of X(k) in (1),we see that X(k) can be expressed as:

$$X(k) = \overline{X}(k) + B(k)V(k) \quad (52)$$

where $\overline{X}$ is independent of v. B(k) is a matrix sequence that is independent of w(k) and V(k) is a vector consisting of the past values of v:

$$V(k) = [v(k-1), v(k-2),..., v(0)]^T \quad (53)$$

With expression (52) and the relation (51), we may expand w(k) in ascending powers of the elements of V. In particular, the linear terms explicitly display the nature of the sensitivity of the weight error, w(k), to the sensor noise.

Substituting (52) into (51) and expanding in powers of V, we obtain:

$$w(k) = \overline{Q}(k)[I - \alpha \sum_{m=0}^{k-1} (\overline{X}(m) V^T(m) B^T(m)) + B(m)V(m) \overline{X}^T) \overline{Q}(k) + H.O.T.] *[w(0) - \alpha \sum_{m=0}^{k-1} \overline{X}(m)v(m)] \quad (54)$$

$$\overline{Q}(k) = (I + \alpha \sum_{m=0}^{k-1} \overline{X}(m) \overline{X}^T(m))^{-1} \quad (55)$$

where $\overline{Q}$ can be recognized as the value of Q(k) in the absence of sensor noise. Let us now consider the limiting form of the above expression for large k. Assuming the ergodicity of all processes, each sum in (54) and (55) approaches an ensemble average times k. When we consider as higher order terms those whose variance goes as 1/k, we obtain for large k:

$$w(k) \rightarrow - R^{-1}(k)[I - (1/k) \sum_{m=0}^{k-1} (\overline{X}(m)V^T(m)B^T(m) + B(m)V(m) \overline{X}^T) R^{-1}(k)] *[(1/k) \sum_{m=0}^{k-1} \overline{X}(m)v(m)] + H.O.T. \quad (56)$$

$$R(k) = (1/k) \sum_{m=0}^{k-1} \overline{X}(m) \overline{X}^T(m) \quad (57)$$

In (56), we see that since V involves past values of v, V(m) and v(m) are almost always correlated and E[w(k)] does not vanish as k increases without bound. Consequently, there is a nonzero steady state bias error. Note also that for large k, R(k) approaches the correlation matrix of the regressor without sensor noise. It is evident from (57) that very large contributions to w(k) will arise from those eigen components of R associated with very small eigenvalues. Thus in cases, such as our three mode example, in which there is a large spread in the eigenvalues of R it should not be surprising to see considerable amplification of the impact of sensor noise.

The above analysis shows that noise sensitivity is exacerbated by a large spread in the eigenvalues of the regressor correlation matrix. Thus we are finally led back to the possibility of reducing noise sensitivity by transforming the regressor so as to equalize the eigenvalues of $E[XX^T]$. To demonstrate this, we generate X(k) over a thousand time step interval and use the PCA algorithm developed in detail in [5] to determine matrix U which defines a transformation of the regressor that approximately renders its correlation matrix equal to unity. Specifically, having run the PCA algorithm, we determine U and then define the transformed regressor, Z(k), by:

$$Z(k) = UX(k) \quad (58)$$

Z(k) replaces X(k) in (16), (23) and (24). By virtue of the properties of the PCA algorithm, $E[ZZ^T]$ is approximately identity. With these additions to the algorithm, we run simulations with T=1000, N=6, α=1e12 and for various values of σ and generate the same quantities characterizing steady state fluctuation and bias error as were shown in Table 3.1. The results are shown in Table 4.1.

Table 4.1: Steady state fluctuation and bias errors for algorithm (23)-(24), with the regressor replaced by Z = UX.

σ	$\sqrt{E[(er)^2]}$	$\|W(T)-Wp\| / \|Wp\|$
1.0e-4	2.621e-3	2.040e-5
5.0e-4	1.210e-2	2.173e-4

1.0e-3	2.097e-2	6.793e-4
5.0e-3	4.464e-2	1.312e-3
1.0e-2	8.119e-2	1.712e-3
5.0e-2	2.764e-1	9.187e-3
1.0e-1	4.598e-1	1.130e-2

Contrasting these results with the numbers shown in Table 3.1, we see that while the variance of the fluctuation error has been reduced by a modest 10%, the bias errors are reduced by approximately three orders of magnitude. This tends to qualitatively confirm the analysis of the noise induced errors given above. The results for $\sqrt{E[(er)^2]}$ remind us that since the regressor always contains a component arising from sensor noise, there is an irreducible minimum magnitude to the fluctuating output error due to noise. The numbers given appear to reflect this irreducible minimum. It should be noted that while the bias error in W has been essentially eliminated, the modified algorithm retains the extremely rapid initial convergence of the original algorithm developed in Section 3. This is illustrated in Figure 6 which shows the time variation of the system output and the identifier output error in the presence of sensor noise with σ=0.1 (the largest noise value studied). It is seen that the error converges down to the additive noise term in about 20 time steps. Figure 7 shows the corresponding behavior of the elements of the weight vector. As in the noise free case, these converge to essentially constant values in less than 40 time steps. For comparison, the stars in Figure 7 show the converged values for W in the zero noise case; i.e. the exact values of the elements of W. From this it may be concluded that despite the high level of sensor noise, the identifier weights have converged with negligible bias error.

Finally, we illustrate the effectiveness of the information filter algorithm on a more realistic example - the UltraLITE model considered in [5].

The original data on UltraLITE provided to this study was in the form of the standard [A,B,C,D] matrices for a continuous-time model. Specifically, the 200x200 A matrix was in 2x2 block diagonal modal form, the C matrix pertains to the position sensors and the B matrix corresponds to the co-located piezo actuators. This continuous-time model was then transformed into a discrete time model assuming a 200Hz sample rate and then truncated to a ten mode model as discussed in [5]. To set up a suitable identification problem, we assume that the system represented by the hundred-mode model is stimulated with discrete-time white noise injected through actuator number 1. It is also assumed that we monitor response through the (co-located) position sensor number 1. Comparisons of response of the 100-mode model and a truncated model obtained by including the first 10 modes showed that ten mode model afforded acceptable accuracy.

The PCA algorithm was exercised on the above ten mode UltraLITE model and the transformation U extracted. Then the fast adaptive algorithm was employed. As a result, the identification output error converges to zero in approximately 28 time steps (0.14 second). This is illustrated by Figure 8. Convergence of the weights takes somewhat longer - approximately 60 steps or 0.3 second, as indicated in Figure 9. Thus, as found in the three mode example, this algorithm is truly fast. This is reassuring, particularly in view of the considerably greater complexity of the UltraLITE example.

In summary, we conclude that an effective way to reduce the noise sensitivity of the extended Kalman algorithm is to first apply the PCA algorithm to transform the regressor vector, then proceed with the information filter algorithm.

5. Concluding Remarks

In this paper, we have investigated a technique for attaining faster and more predictable convergence of neural network algorithms for autonomous system identification. The approach developed here seeks to improve the adaptation speed of neural adaptive identifiers by means of a basic revision of the fundamental learning mechanism involving a transformation of the increment in the synaptic weights. A neural network algorithm has been found which eliminates the dependence of convergence behavior upon the correlation matrix of the net input. Analysis and computational studies show that this algorithm can produce many orders of magnitude improvement in convergence speed beyond the performance available from the earlier LMS type algorithms. Although the algorithm can also be shown to be the optimal estimator for a particular sensor noise model (unfortunately an inapplicable model in the present case), a drawback is that the algorithm also tends to amplify the effect of sensor noise on the identification errors. This is manifested in both amplified noise in the output errors and large steady state bias errors in the network weights. However, analysis of the root cause of the noise sensitivity shows that the solution is to combine this approach with the procedure discussed in [5] wherein a Principal Component Analysis algorithm is first used to transform the regressor to a new regressor input having uncorrelated, equal variance components. Numerical results presented in the last section, show that the combined approach succeeds in

achieving very rapid convergence with low noise sensitivity.

References

1. G.G. Yen, "Autonomous neural control in flexible space structures," *Control Engineering Practice,* 3(4), April 1995, pp. 471-483.

2. D. C. Hyland, "Connectionist Algorithms for Identification and Control:
System Structure and Convergence Analysis", Paper No. AIAA 97-0686, 35th Aerospace Sciences Meeting, Reno, NV, January, 1997.

3. Hyland, D.C., Davis, L.D., Das, A., and Yen, G. "Autonomous Neural Control for Structure Vibration Suppression", AIAA -96-3923, AIAA Guidance, Navigation and Control Conference, San Diego, CA, July 1996.

4. Hyland, D.C. and Davis,L.D., "Adaptive Neural Control Program - Final Report", A F Phillips Lab. contractor report, February, 1997.

5. Hyland, D.C. and Davis, L.D., "Accelerated Convergence of Neural Network System Identification Algorithms Via Principal Component Analysis", AIAA Guidance, Navigation and Control Conference, Boston, MA., August, 1998.

6. Jolliffe, I. T., Principal Component Analysis. New York: Springer-Verlag. 1986.

7. Haykin, S., Neural Networks: A Comprehensive Foundation, New Jersey: Prentice-Hall . pp. 370-380. 1994.

8. R.R. Bitmead, "Persistence of Excitation Conditions and the Convergence
of Adaptive Schemes", *IEEE Trans. On Infor. Theory,* vol. IT-30, No. 2, March 1984.

9. Noble, B. Applied Linear Algebra, Prentice-Hall, New Jersey, p.147. 1969.

10. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Section 7.3, pp.200ff, Academic Press,1970.

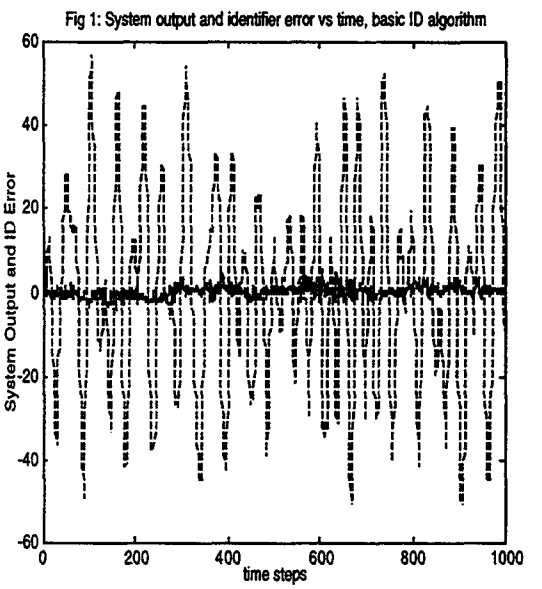

Figure 1: System output, y(k) (dashed line), and identifier output error, e(k) (solid line) versus time for the three mode example, basic identification algorithm with N = 6, and alpha = 1. The initial weight vector is zero.

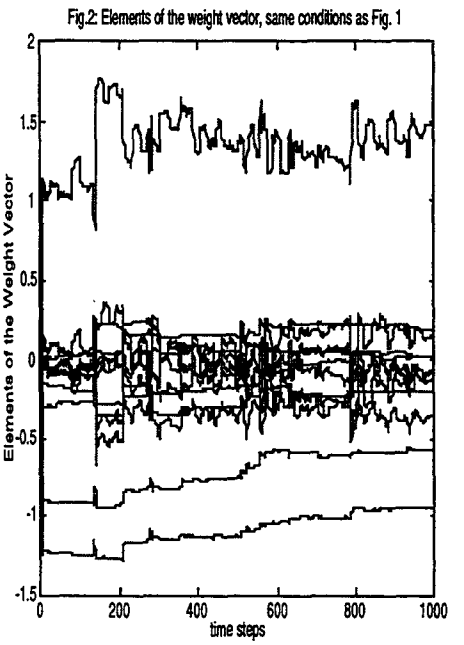

Figure 2: Elements of the weight vector of the identifier versus time, same conditions as Fig.1.

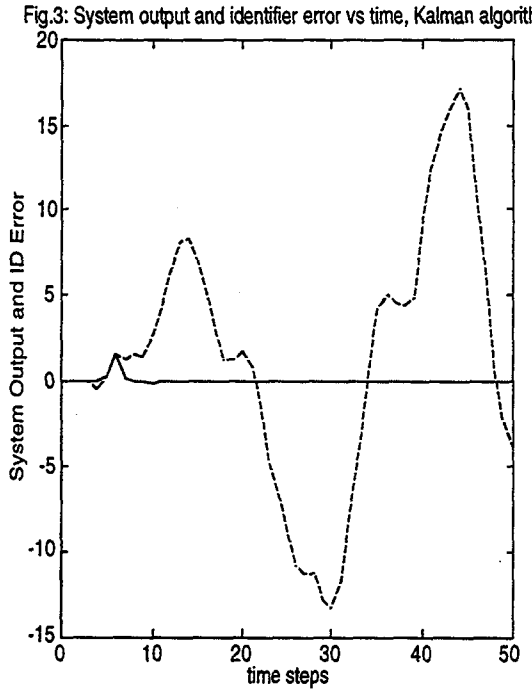

Figure 3: System output, y(k) (dashed line), and identifier output error, e(k) (solid line) versus time for the three mode example, information filter algorithm with N = 6, and alpha = 1e12. The initial weight vector is zero.

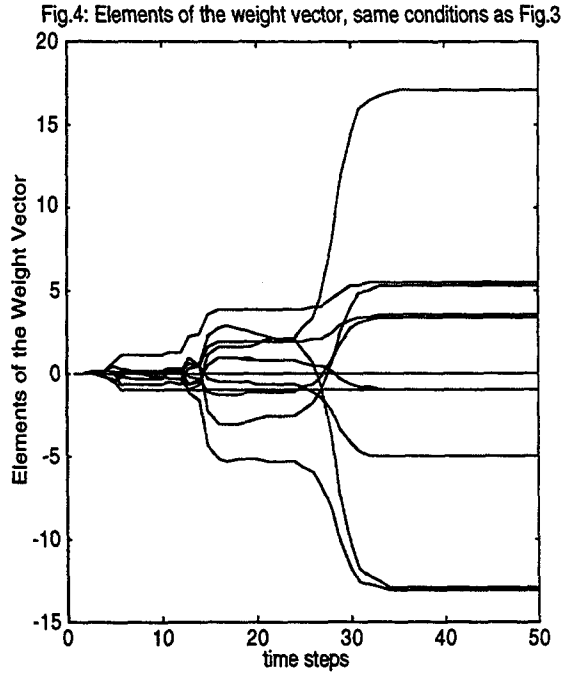

Figure 4: Elements of the weight vector of the identifier versus time, same conditions as Fig.3.

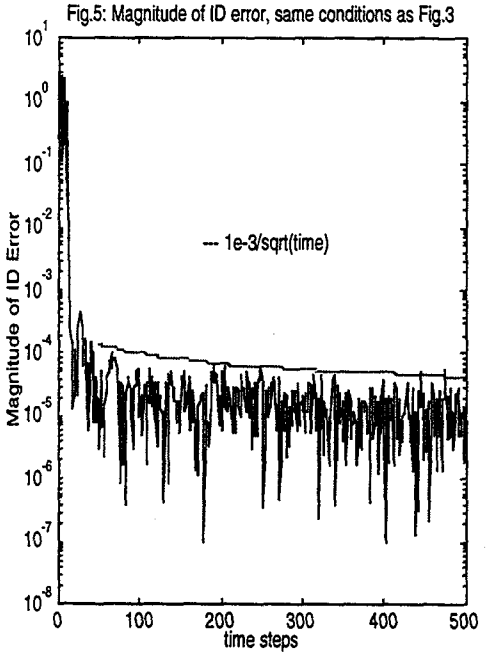

Figure 5: Semi-log plot of the magnitude of the identifier output error versus time, same conditions as Fig.3 except a longer duration simulation. For comparison, the figure also shows 1e-3 / sqrt(time) for time > 50.

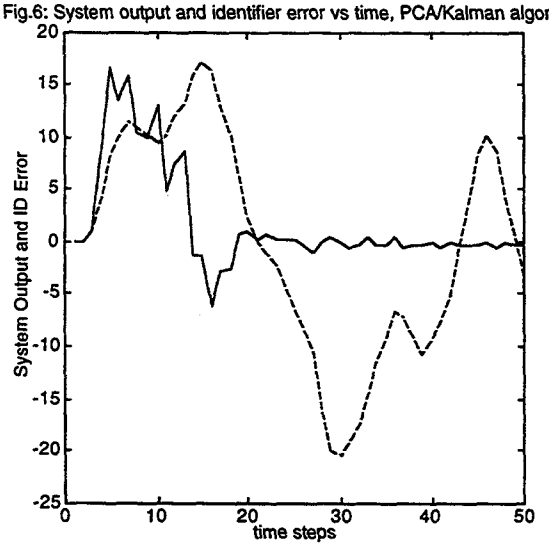

Figure 6: System output, y(k) (dashed line), and identifier output error, e(k) (solid line) versus time for the three mode example, using the PCA-generated regressor transformation followed by the information filter algorithm with N = 6, and alpha = 1 and in the presence of sensor noise with sigma = 0.1. The initial weight vector is zero.

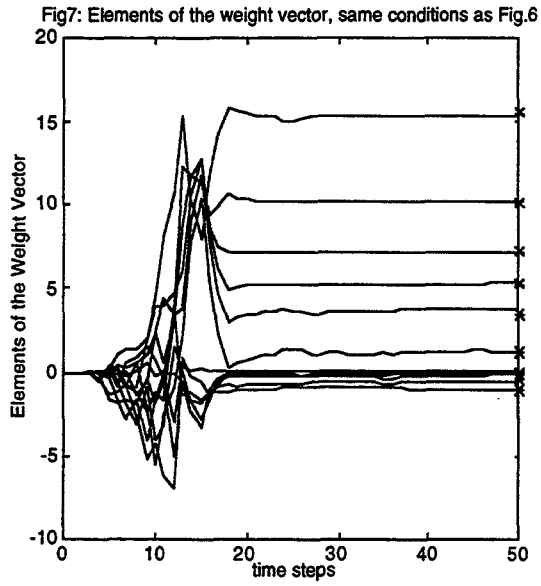

Figure 7: Elements of the weight vector of the identifier versus time, same conditions as Fig.6. For comparison, the stars show the converged values for W in the zero noise case; i.e. the exact values of the elements of W.

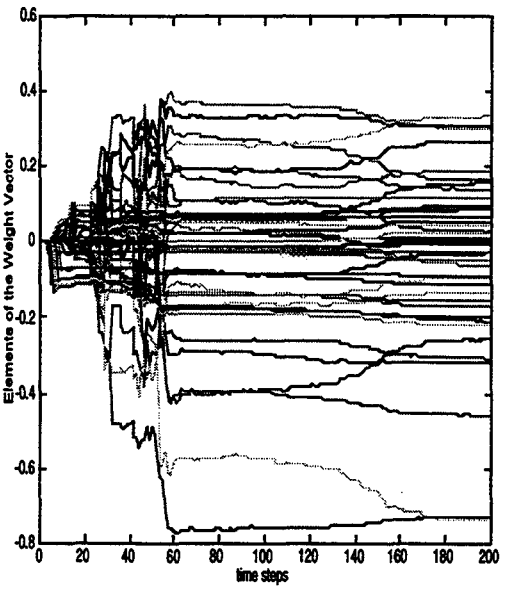

Figure9: Elements of the weight vector of the identifier versus time, same conditions as Fig.8.

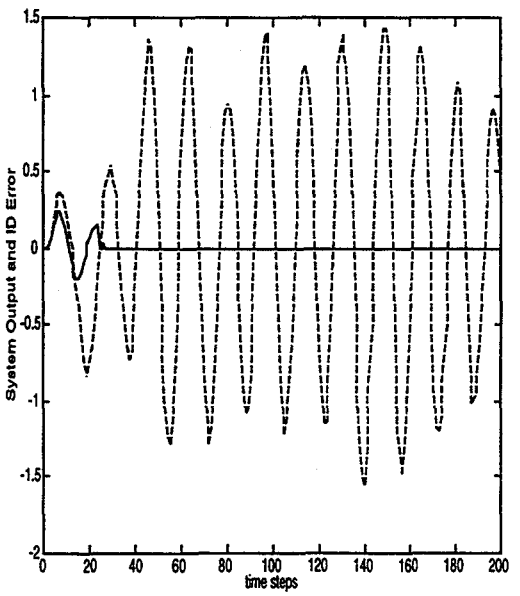

Figure 8: System output, y(k) (dashed line), and identifier output error, e(k) (solid line) versus time for the ten mode UltraLITE example, information filter algorithm with N = 20, and α = 1e12. The initial weight vector is zero.