EFFICIENT LINEAR MIMO ADAPTIVE INVERSE CONTROL

Gregory L. Plett [1]

University of Colorado at Colorado Springs,
Department of Electrical and Computer Engineering
1420 Austin Bluffs Parkway, P.O. Box 7150, Colorado Springs, CO
80933–7150 USA, `glp@eas.uccs.edu`

Abstract: Adaptive inverse control is an automatic control-system design method which "learns" over time how to control a particular dynamic system. This simple controller design approach can achieve very precise control of minimum- and nonminimum-phase systems, even with poor a priori knowledge of the plant dynamics. A problem with adaptive inverse control in its current state is that the learning algorithms for MIMO and nonlinear systems are very slow. This paper addresses some results concerning a new way to perform adaptive inverse control of linear MIMO systems which learns very quickly. An example is presented to demonstrate the dramatic speedup.

Keywords: Adaptive control, Disturbance rejection, Efficient algorithms, Linear control systems, MIMO, Recursive least squares, Stochastic control, Wiener filters

1. INTRODUCTION

Adaptive inverse control is an automatic control-system design method which learns how to control a specific plant (Plett, n.d.; Plett, 1998; Widrow *et al.*, 1998; Widrow and Plett, 1997; Widrow and Walach, 1996; Widrow and Plett, 1996*a*; Widrow and Plett, 1996*b*; Widrow and Plett, 1996*c*). Adaptive filtering methods are used throughout. Invisible to the user, a three-part process is used internally. First, an adaptive plant model $\widehat{P}$ learns the dynamics of the plant. Secondly, an adaptive feedforward controller C learns to control the dynamics of the plant. Thirdly, an adaptive feedback disturbance canceler X learns to cancel disturbances which affect the plant. These processes may proceed concurrently. A block diagram of an adaptive inverse control system is shown in Fig. 1.

Methods are known to quickly adapt controllers for linear SISO plants (Widrow and Walach, 1996), but known methods for adapting controllers for linear MIMO and nonlinear plants learn very slowly (Plett, n.d.; Plett, 1998; Widrow and Walach, 1996). This paper presents some results regarding a training method for linear MIMO adaptive inverse control that learns about as quickly as methods for linear SISO control.

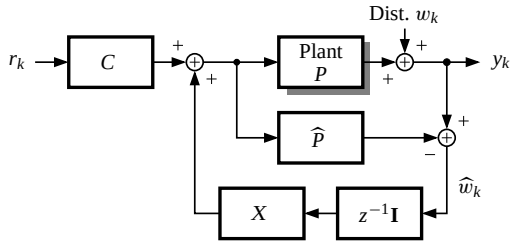

Fig. 1. Adaptive inverse control system block diagram.

2. ADAPTIVE DIGITAL FILTERING

Adaptive inverse control relies on adaptive digital filtering methods. Adaptive filters have an input, an output, and a "special input" called the desired response. The desired response d_k specifies the output we wish the filter to have. It is used to calculate an error signal e_k, which in turn is used to modify the internal parameters of the filter in such a way that the filter "learns" to

[1] Partially supported by a William Grigsby Sencenbaugh grant.

perform a certain function. Often, the trick to applying adaptive filtering to a specific application is finding a way to generate an appropriate d_k signal.

For the sake of the present discussion we limit ourselves to *finite impulse response* (FIR) filters. The filter output is computed as a weighted sum of its current and N previous inputs $y_k = \mathbf{W}X_k$, where the column-vector $X_k = [x_k^T,\ x_{k-1}^T \ldots x_{k-N}^T]^T$ and $\mathbf{W}$ is the *weight matrix* of the filter.

Linear SISO filters have a single input and a single output at each time instant; for $\mathbf{W}$ to represent a SISO system, x_k and y_k must be scalars. $\mathbf{W}$ is then a single row, and its components are the impulse response of the filter. Linear MIMO filters have (possibly) many inputs and outputs at each time instant so x_k and y_k are (column) vector signals. $\mathbf{W}$ then has many rows, and the individual impulse responses are interleaved in each row.

The weights in $\mathbf{W}$ may be adapted in a variety of ways in order that the output of the filter learns to closely match the desired output. For example, the weights may be updated by a gradient-descent optimization procedure such as $\Delta\mathbf{W} = 2\mu(d_k - y_k)X_k^T$, where μ is a small positive *learning constant.* This rule, commonly known as the matrix-LMS algorithm is well described in several textbooks (Haykin, 1996; Widrow and Stearns, 1985). Other, more sophisticated, algorithms, such as matrix-RLS (see the algorithm in Alg. 1) usually converge more quickly, and are very popular. For a good overview, see Glentis *et al.* (1999).

3. CONTROLLING LINEAR SISO PLANTS

We now consider adapting the feedforward controller C. (We shall restrict our development to apply only to stable plants. If the plant of interest is unstable, conventional feedback should be applied to stabilize it. Then the combination of the plant and its feedback stabilizer can be regarded as an equivalent stable plant.) The goal is to make the dynamics of the controlled system PC approximate the fixed filter M as closely as possible, where M is a user-specified *reference model*. The input reference signal r_k is filtered through M to create a desired response d_k for the plant output. The measured plant output is compared with the desired plant output to create a system error signal $e_k^{(sys)} = d_k - y_k$. We wish to adapt C to minimize the mean-squared system error.

We note that if the plant is minimum-phase, that is, has all of its poles and zeros inside the unit circle in the z-plane, then the inverse will be stable with all of its poles inside the unit circle. If the plant is nonminimum-phase, then some of the poles of the inverse will be outside the unit circle. According to the theory of two-sided z-transforms, the inverse will then either be unstable *or* noncausal. Since minimizing system error will not lead to an unstable solution—which would have unbounded system error—the algorithm will attempt to match a noncausal solution. This will result in very poor control unless the reference model has built-in latency. The longer the latency, the better we can approximate a delayed version of a noncausal inverse with a causal controller. A typical design latency is the transport delay of the plant.

Algorithm 1 Matrix-RLS algorithm.

$$\pi(k) = X(k)^T \Phi_{xx}^{-1}(k-1)$$
$$r(k) = \frac{1}{\lambda + \pi(k)X(k)}$$
$$K(k) = r(k)\pi(k)$$
$$\zeta(k) = d(k) - \mathbf{W}(k-1)X(k)$$
$$\mathbf{W}(k) = \mathbf{W}(k-1) + \zeta(k)K(k)$$
$$\Phi_{xx}^{-1}(k) = \frac{1}{\lambda}\mathrm{tri}\left\{\Phi_{xx}^{-1}(k-1) - \pi(k)^T K(k)\right\},$$

where the tri{} operator takes the upper or lower triangular part of a matrix and replicates it in the lower or upper part to preserve symmetry. λ is a forgetting constant, and is set slightly less than 1. $X(k)$ is the tap-delay-line input to the filter at time k, such that the filter output is $\mathbf{W}(k)X(k)$. $\Phi_{xx}^{-1}(k)$ is initialized to a diagonal matrix with large (order of 10,000) entries.

3.1 *Linear SISO Feedforward Control:*

The challenge when attempting to adapt C is in generating its desired response signal. We notice that the system desired response is available at the output of the plant, and not at the output of the adaptive controller. A variety of solutions are known when the plant is a linear SISO system (Widrow and Walach, 1996). One simple and very effective method takes advantage of the commutability of transfer functions of linear SISO systems. That is, $P(z)C(z) = C(z)P(z)$. The block diagram of Fig. 2(a) may then be used to adapt the controller. A random signal n_k is filtered by a digital copy [2] of the plant model $\widehat{P}(z)$ and then by the controller $C(z)$. It is also filtered by the reference model $M(z)$. Since the output of the reference model is the desired output of the cascade $C(z)\widehat{P}(z)$, it is used as the desired response signal for $C(z)$. A filter whose weights are a digital copy of $C(z)$ is used as the feedforward controller.

This method is unbiased by zero-mean disturbance, but may be affected by plant modeling errors. More sophisticated methods are available to adapt controllers and which are not affected by plant modeling errors (Widrow and Walach, 1996). If the RLS algorithm is used to adapt $C(z)$, convergence occurs

[2] The weights of the digital copy are identical to the weights of the adaptive plant model, although the input to both filters is different and hence the outputs of the two filters are different. Note that alternative adaptive methods such as MRAC by Åström and Wittenmark (1995) do not require a digital copy of a plant model. For a good discussion comparing and contrasting STR/MRAC with AIC, see Appendix C of Widrow and Walach (1996).

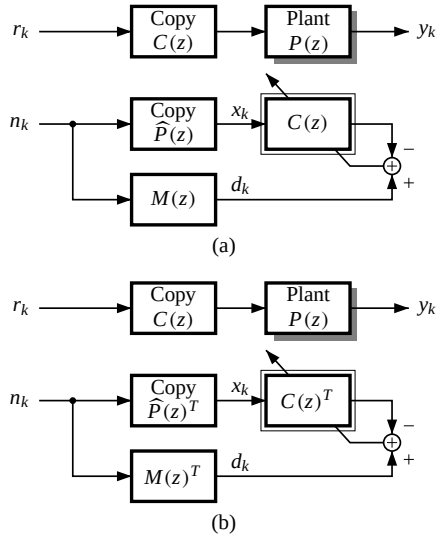

Fig. 2. Adapting a linear controller $C(z)$. (a) For a SISO linear plant; (b) For a MIMO linear plant. The plant modeling process has been omitted for clarity.

within twice as many iterations as there are taps in the controller filter C (Haykin, 1996).

3.2 *Linear SISO Disturbance Canceling*

The dynamic response of the system may now be controlled but plant disturbance, however, is not yet rejected. This can be accomplished via another special adaptive filter called the disturbance canceler X. In order to avoid bias in the adaptive plant model due to the disturbance, a special architecture must be used when adapting $\widehat{P}$ and X simultaneously. The solution is shown in Fig. 3. It can be shown that the plant model will adapt to the correct solution using this scheme (Plett, n.d.; Plett, 1998) so long as the disturbance is is zero-mean and uncorrelated with u_k.

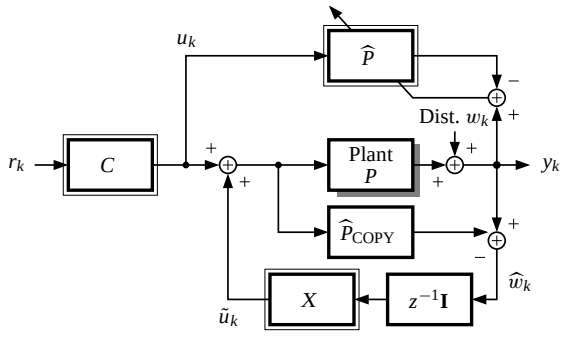

Fig. 3. Correct on-line adaptive plant modeling in conjunction with disturbance canceling for linear plants. The circuitry for adapting C has been omitted for clarity.

When considering the linear SISO case, we would like to adapt the disturbance canceler X such that $z^{-1}P(z)X(z) = -1$. This would entirely cancel the disturbance, but result in a non-causal X. We can still use this formula to adapt $X(z)$, as shown in Fig. 4(a), but by the adaptive method, $X(z)$ will adapt to the optimal causal solution. [3] As when adapting a controller, we take advantage of the fact that linear systems commute in order to generate an adaptation signal for X.

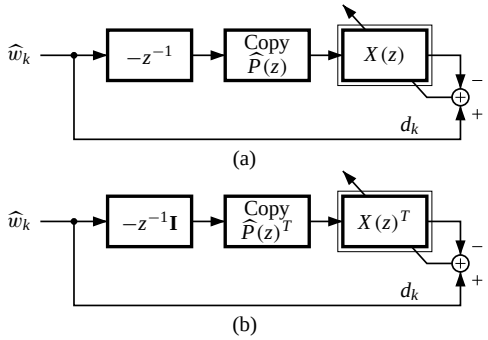

Fig. 4. Architectures for training disturbance canceler X; (a) for linear SISO systems; (b) for linear MIMO systems. The actual disturbance canceler is a digital copy of X.

4. THE NOVEL APPROACH TO LINEAR MIMO ADAPTIVE INVERSE CONTROL

4.1 *Feedforward Control*

While linear SISO systems have transfer functions, linear MIMO systems have transfer function matrices, denoted for example as $[P(z)]$. Matrix multiplication is not in general commutative; therefore, $[P(z)][C(z)] \neq [C(z)][P(z)]$ and we can not use the same simple method as with the SISO system. A number of ways have been proposed to adapt a controller for a linear MIMO system (Plett, n.d.; Plett, 1998; Widrow *et al.*, 1998; Widrow and Plett, 1997; Widrow and Walach, 1996). These methods tend to make inefficient use of the available data and are slow. Here, we present a very simple and fast method to adapt a linear MIMO controller (Plett, 2000). It uses the fact that $[P(z)][C(z)] = [C(z)]^T[P(z)]^T$. The block diagram in Fig. 2(b) may then be used to adapt the controller. The entire operation depends on being able to take the "transpose" of an adaptive filter representing a transfer function matrix. These filters are actually stored as impulse-response matrices, and the transpose operation is simply a re-organization of the components of the impulse-response matrices. The mechanics of taking a filter transpose are discussed in the next section. As with the SISO linear case, convergence using the RLS algorithm occurs within twice as many iterations as there are taps in the longest impulse response in the

[3] We note that this solution amounts to a predictor which predicts a future sample of the disturbance, cascaded with a plant inverse. Since optimal prediction is in general a nonlinear operation, the ideal disturbance canceler is a nonlinear adaptive filter even if the plant is linear!

MIMO controller filter C. This is an improvement of many orders of magnitude when compared with the other known cited methods.

4.2 *Transpose of Transfer Functions*

In order to train the controller, notice that we need the transposed filter $[\hat{P}(z)]^T$. The weight matrix for this filter is not the same as $\mathbf{W}_{\hat{P}}^T$. To find the correct weight matrix for an arbitrary transposed filter $[H(z)]^T$ consider first the weight matrix for some arbitrary filter $[H(z)]$. For the sake of example we will assume $[H(z)]$ has three inputs, two outputs and $N = 1$ so that the impulse response is two samples long. Then, the H-matrix has entries as shown in Fig. 5.

$[h_{11}]_0$	$[h_{12}]_0$	$[h_{13}]_0$	$[h_{11}]_1$	$[h_{12}]_1$	$[h_{13}]_1$
$[h_{21}]_0$	$[h_{22}]_0$	$[h_{23}]_0$	$[h_{21}]_1$	$[h_{22}]_1$	$[h_{23}]_1$

Fig. 5. Weight matrix for filter $[H(z)]$.

There are six impulse responses embedded in this matrix: h_{11}, h_{12}, h_{13}, h_{21}, h_{22} and h_{23}. When we take the z-transform and make the transfer-function matrix for this filter, we get

$$[H(z)] = \begin{bmatrix} H_{11}(z) & H_{12}(z) & H_{13}(z) \\ H_{21}(z) & H_{22}(z) & H_{23}(z) \end{bmatrix}$$

which has transpose

$$[H(z)]^T = \begin{bmatrix} H_{11}(z) & H_{21}(z) \\ H_{12}(z) & H_{22}(z) \\ H_{13}(z) & H_{23}(z) \end{bmatrix}.$$

So, the transpose of the weight filter has entries as shown in Fig. 6. It may be computed in Matlab, for example, with a sequence of matrix `reshape` and `permute` commands.

$[h_{11}]_0$	$[h_{21}]_0$	$[h_{11}]_1$	$[h_{21}]_1$
$[h_{12}]_0$	$[h_{22}]_0$	$[h_{12}]_1$	$[h_{22}]_1$
$[h_{13}]_0$	$[h_{23}]_0$	$[h_{13}]_1$	$[h_{23}]_1$

Fig. 6. Weight matrix for filter $[H(z)]^T$.

4.3 *Wiener Solution for Controller*

A property of linear adaptive filtering is that the solution to which the algorithm converges may be verified mathematically. This solution is known as the *Wiener* solution. There are three cases to consider when computing the Wiener solution for the controller weight matrix when the plant is linear MIMO: (1) the plant has more outputs than inputs; (2) the plant has fewer outputs than inputs; (3) the plant has an equal number of outputs and inputs. The details of the Wiener solution are in Plett (2000).

More outputs than inputs: If the plant has more outputs than inputs, then $\widehat{P}(z)^T \widehat{P}(z)$ is generally invertible. The controller converges to

$$C^{(opt)}(z) = \left[\widehat{P}(z)^T \widehat{P}(z)\right]^{-1} \widehat{P}(z)^T M(z),$$

which is the pseudo-inverse of $\widehat{P}(z)$ in cascade with the reference model. This is the desired solution.

More inputs than outputs: If the plant has more inputs than outputs, then $\widehat{P}(z)\widehat{P}(z)^T$ is generally invertible. One possible solution for $C(z)$ is

$$C^{(opt)}(z) = \widehat{P}(z)^T \left[\widehat{P}(z)\widehat{P}(z)^T\right]^{-1} M(z).$$

This is the minimum-norm solution. Generally, when there are more inputs than outputs there will be many solutions to $C(z)$ which all achieve zero error. The minimum-norm solution is one of these, and uses the least amount of control effort (in a mean-squared sense). It is not a guaranteed solution.

Equal number of inputs and outputs: If the plant has an equal number of inputs and outputs, and if the plant is invertible, then both solutions become

$$C^{(opt)}(z) = \widehat{P}(z)^{-1} M(z).$$

We note that all three Wiener solutions are based on the inverse of the plant model and not on the inverse of the plant. So, the efficacy of this method is dependent on an accurate plant model.

4.4 *Disturbance Canceling*

We can adapt a disturbance canceler for a linear MIMO system using the same equation as for the SISO case, and adopting the transpose method used when adapting C for the MIMO system. The block diagram of the adaptation method is shown in Fig. 4(b).

4.5 *Speed of Convergence*

There are two convergence issues to deal with. First, convergence of the plant model, and then convergence of the controller/disturbance canceler. The plant model needs to converge before either the controller or disturbance canceler can converge. Here we assume that matrix-RLS is used to adapt the two filters.

According to Haykin, RLS converges in two times as many iterations as there are taps in the FIR filter (Haykin, 1996). So, for example, if a matrix-FIR filter has five taps per sub-filter, convergence is achieved in about ten iterations. Similarly, convergence of the controller or disturbance canceler occurs in about twice as many iterations as there are taps in each filter. However, since the controller- and canceler adaptation process is done offline, it can be done

quickly in the background and entire system convergence occurs can occur in about twice as many iterations as there are taps in the plant-model FIR filter.

5. EXAMPLE

Aspects of flight control for a Boeing 747 aircraft were selected to demonstrate linear, MIMO control. The dynamics of the airplane have been approximated by a linear model around an equilibrium point. In the case at hand, the equilibrium "point" is level flight at 40,000 ft and a nominal forward speed of Mach 0.8 (774 ft/sec). For further detail regarding particulars of this example, see Plett (1998). Here, three cases are considered. All simulations consider the system to have two inputs (rudder angle and aileron angle). Simulations have one through three outputs (yaw-rate, roll-rate, bank-angle).

The discrete-time state-space model of the plant has

$$A_d = \begin{bmatrix} 0.8876 & -0.3081 & 0.0415 & 0.0198 \\ 0.2020 & 0.3973 & -0.0046 & 0.0024 \\ -1.2515 & 0.5106 & 0.7617 & -0.0139 \\ -0.3313 & 0.1510 & 0.4407 & 0.9976 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0.4806 & -0.0013 \\ -1.5809 & 0.3887 \\ 0.0599 & 4.8390 \\ 0.0390 & 1.2585 \end{bmatrix},$$

with a sampling period of 0.5 sec. The output matrix was chosen to be the first one, two, or three rows of

$$C_d = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

depending on whether the plant had one, two or three outputs, and $D_d = 0$. The reference input was filtered white noise, and disturbance was nonlinearly filtered white noise, added directly to the state. The reference model was a unit delay: $M(z) = z^{-1}\mathbf{I}$.

Tracking results for converged controllers (in the absence of disturbance) are presented in Fig. 7. Tracking is essentially perfect when there are more plant inputs than outputs, and is also very good when there are an equal number of plant inputs and outputs. When there are more plant outputs than inputs, it becomes impossible to get perfect tracking in general, and the controller adapts to find the solution which minimizes the mean-square output error.

Note: simulations in Plett (1998) considered two-input two-output control. The controller, trained with the BPTM method, converged in about 10^8 iterations. Learning curves for the present method and the three simulation cases are presented in Fig. 8. We see convergence in all cases within 200–400 iterations. Training time, to a similar level of steady-state error, is

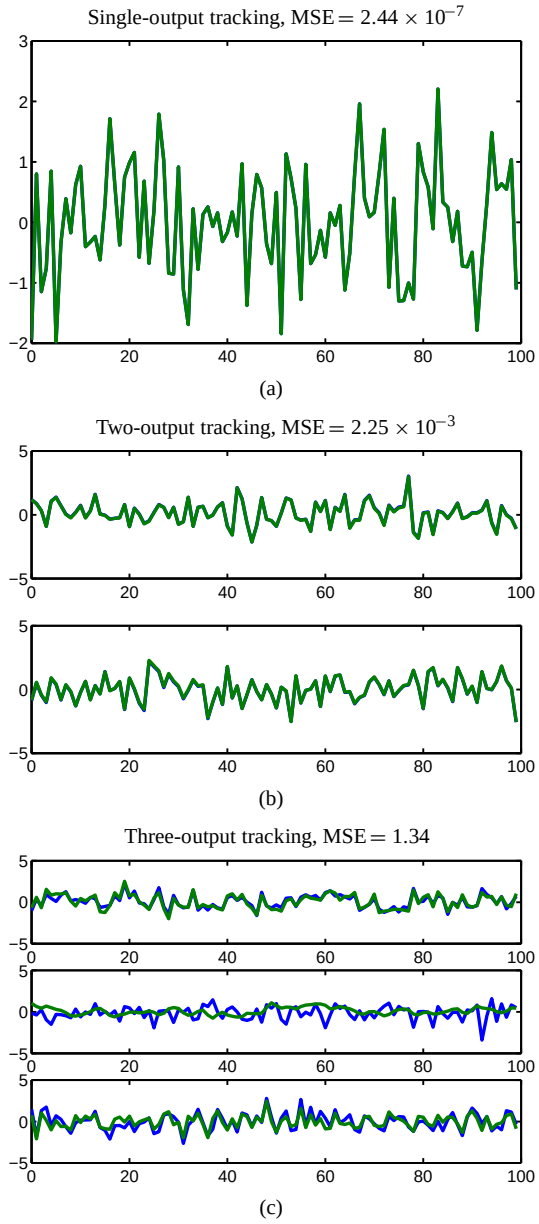

Fig. 7. Tracking performance for three trained controllers. The plant had one through three outputs in (a) through (c), respectively. The dark line is the desired plant output; the light line is the actual plant output.

improved by several orders of magnitude. As a point of reference, the plant model had 61 taps in each impulse response, and the controller had 71 taps in each impulse response.

Disturbance canceling was tested for the MIMO system, and results are plotted in Fig. 9. The figure shows the squared-norm of the system error plotted versus iteration. Disturbance was present at all times, and the (trained) disturbance canceler was turned "on" at time 1000. As can be seen, the disturbance canceler removes essentially all of the disturbance, resulting in near-perfect tracking even in the presence of disturbance.

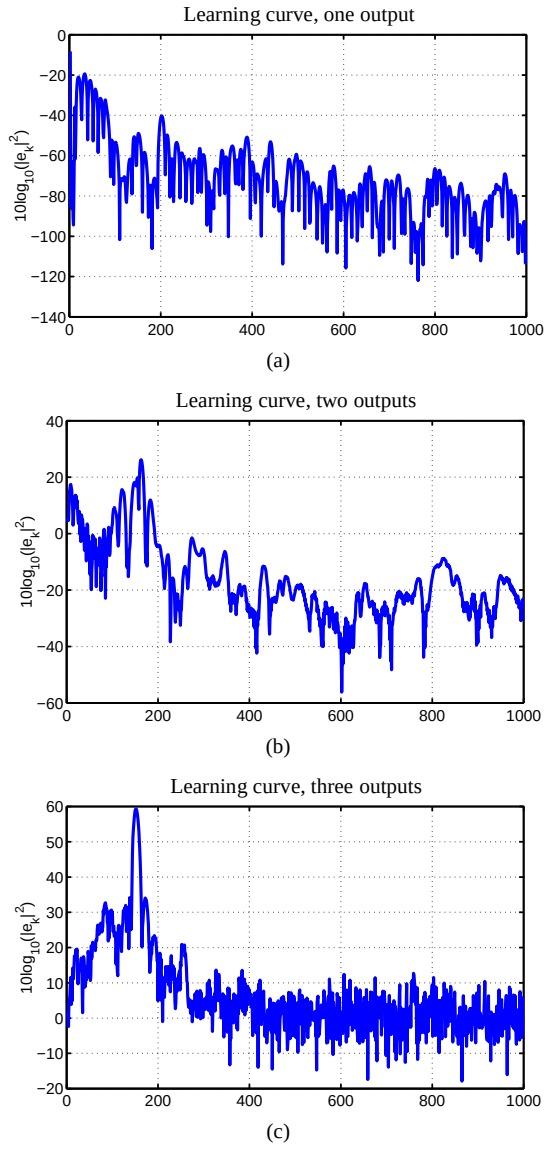

Fig. 8. Learning curves training the three controllers (in dB). The plant had one through three outputs in (a) through (c), respectively. The horizontal axis is the number of training iterations.

6. CONCLUSIONS

A new method is presented in this paper for adapting the controller and disturbance canceler in an adaptive-inverse-control system for linear MIMO plants. The method works very well and is very fast. Simulation results indicate orders-of-magnitude speed improvement over known methods. The method works for MIMO systems with equal or unequal numbers of inputs and outputs.

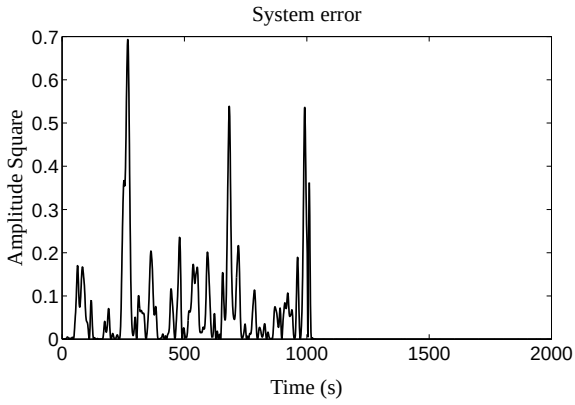

Fig. 9. Disturbance canceling for MIMO system.

REFERENCES

Åström, K. J. and B. Wittenmark (1995). *Adaptive Control*. second ed.. Addison-Wesley. Reading, MA.

Glentis, G., K Berberidis and Sergios Theodoridis (1999). Efficient least squares adaptive algorithms for FIR transversal filtering. *IEEE Signal Processing Magazine* **16**(4), 13–41.

Haykin, S. (1996). *Adaptive Filter Theory*. third ed.. Prentice Hall. Upper Saddle River, NJ.

Plett, G. L. (1998). Adaptive Inverse Control of Plants with Disturbances. PhD thesis. Stanford University. Stanford, CA 94305.

Plett, G. L. (2000). Some results concerning fast linear MIMO adaptive inverse control. Technical Report EAS_ECE_2000_11. ECE Department, University of Colorado at Colorado Springs. P.O. Box 7150, Colorado Springs, CO 80933–7150.

Plett, G. L. (n.d.). Adaptive inverse control of unmodeled stable SISO and MIMO linear systems. *International Journal of Adaptive Control and Signal Processing*. (in press).

Widrow, B. and E. Walach (1996). *Adaptive Inverse Control*. Prentice Hall P T R. Upper Saddle River, NJ.

Widrow, B. and G. L. Plett (1996*a*). Adaptive inverse control based on linear and nonlinear adaptive filtering. In: *Proceedings of International Workshop on Neural Networks for Identification, Control, Robotics and Signal/Image Processing*. (Venice, Italy: August 1996). pp. 30–38.

Widrow, B. and G. L. Plett (1996*b*). Adaptive inverse control based on linear and nonlinear adaptive filtering. In: *Proceedings of the World Congress on Neural Networks*. (San Diego, CA: September 1996). pp. 620–27.

Widrow, B. and G. L. Plett (1996*c*). 'Intelligent' adaptive inverse control. In: *Proceedings of IFAC*. (San Francisco, CA: July 1996). pp. 104–105.

Widrow, B. and G. L. Plett (1997). Nonlinear adaptive inverse control. In: *Proceedings of the 36th IEEE Conference on Decision and Control*. Vol. 2. (San Diego, CA: December 1997). pp. 1032–1037.

Widrow, B. and S. D. Stearns (1985). *Adaptive Signal Processing*. Prentice-Hall. Englewood Cliffs, NJ.

Widrow, B., G. L. Plett, E. Ferreira and M. Lamego (1998). Adaptive inverse control based on nonlinear adaptive filtering. In: *Proceedings of 5th IFAC Workshop on Algorithms and Architectures for Real-Time Control AARTC'98*. (Cancun, MX: April 1998). pp. 247–252. (invited paper).