복합시스템을 위한 간접분산학습제어
Indirect Decentralized Learning Control for the Multiple Systems

Soo Cheol Lee
Department of Industrial Engineering, Taegu University

ABSTRACT

The new field of learning control develops controllers that learn to improve their performance at executing a given task, based on experience performing this specific task. In a previous work[6], the authors presented a theory of indirect learning control based on use of indirect adaptive control concepts employing simultaneous identification and control. This paper develops improved indirect learning control algorithms, and studies the use of such controllers in decentralized systems. The original motivation of the learning control field was learning in robots doing repetitive tasks such as on an assembly line. This paper starts with decentralized discrete time systems, and progresses to the robot application, modeling the robot as a time varying linear system in the neighborhood of the nominal trajectory, and using the usual robot controllers that are decentralized, treating each link as if it is independent of any coupling with other links. The basic result of the paper is to show that stability of the indirect learning controllers for all subsystems when the coupling between subsystems is turned off, assures convergence to zero tracking error of the decentralized indirect learning control of the coupled system, provided that the sample time in the digital learning controller is sufficiently short.

Ⅰ. INTRODUCTION

When a control system is required to execute the same command repeatedly, the error in following the command will be repeated (except for random disturbances). It seems a bit primitive to produce the same errors every time the command is given. The new field of learning control refers to controllers that can learn from previous experience executing a command in order to improve their performance. They learn what command input should be given to the system in order to have the response be the desired response. They eliminate the deterministic errors of the control system in

executing the command, and they eliminate errors due to disturbances that repeat each time the command is given. Learning controllers aim to accomplish this with minimal knowledge of the system being controlled, and base their adjustments to the command on previous experience performing the command without relying on an a priori model of the system dynamics. There has been considerable research activity in this field in the last few years, some examples of which are given in the references [1-9].
Learning control has application to all tracking problems in which the command is given repeatedly, but the application that motivated the development of the field in the last few years is robots performing repeated tracking commands, for example on an assembly line. The question arises, what happens if a separate learning controller is used with each of the separate feedback controllers of the robot arm. Such an application represents use of a decentralized learning control. A serious issue is whether the dynamic interactions in the dynamics of the systems governed by the separate learning controllers could cause the learning processes to fail to converge. It is the purpose of this paper to extend the indirect learning control law of [4] to apply to decentralized control situations. In the process we will develop modified forms of the indirect learning control law with various desirable properties.

II. MATHEMATICAL FORMULATION

The s coupled subsystems can be written as one large state equation in an obvious manner

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + w(k) \\ y(k) &= C(k)x(k) \end{aligned} \tag{1}$$

Let the difference operator δ_r operating on any quantity represent the value of that quantity at repetition r minus the value at repetition r -1. Since w(k) repeats each repetition, and since in the learning control problem it is assumed that the initial condition is the same every repetition, we can rewrite (1) as

$$\delta_r \underline{y} = P\delta_r \underline{u} \tag{2}$$

$$\underline{y} = [y^T(1)\ y^T(2)\ \ldots Y^T(p)]^T$$

$$\underline{u} = [u^T(0)\ u^T(1)\ \ldots u^T(p-1)]^T$$

where

$$P = \begin{bmatrix} C(1)B(0) & 0 & \cdots & 0 \\ C(2)A(1)B(0) & C(2)B(1) & \cdots & 0 \\ C(p)(\Pi_{k=1}^{p-1}A(k))B(0) & C(p)(\Pi_{k=2}^{p-1}A(k))B(1) & \cdots & C(p)B(p-1) \end{bmatrix} \tag{3}$$

The product notation represents a matrix product going from larger arguments on the left to smaller arguments on the right.

By reordering the elements of the matrices in (2) to separate elements into those that apply to each subsystem and those that couple the subsystems one can write (2) *in the form*

$$\delta_r \underline{y}_i = P_{ii}\delta_r \underline{u}_i + \sum_{\substack{j=1 \\ j \neq 1}}^{s} P_{ij}\delta_r \underline{u}_j \tag{4}$$

Here the P_{ii} and P_{ij} are lower block triangular matrices for the ith subsystem. The first represents the pulse responses of the ith subsystem to its own inputs and the second gives the pulse responses of the ith system to inputs in other subsystems. This equation serves as the basic equation for the development of all of our decentralized learning control strategies.

For purposes of illustration, suppose that the original system was time invariant, contained two subsystems (s=2), and that the desired trajectory is three time steps long (p=3). Then equation (4) for system one is

$$\begin{bmatrix}\delta_r y_1(1)\\ \delta_r y_1(2)\\ \delta_r y_1(3)\end{bmatrix} = \begin{bmatrix} C_1B_1 & 0 & 0\\ C_1A_{11}B_1 & C_1B_1 & 0\\ C_1(A_{11}^2+A_{12}A_{21})B_1 & C_1A_{11}B_1 & C_1B_1\end{bmatrix}\begin{bmatrix}\delta_r u_1(0)\\ \delta_r u_1(1)\\ \delta_r u_1(2)\end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0\\ C_1A_{12}B_2 & C_1B_1 & 0\\ C_1(A_{11}A_{12}+A_{12}A_{22})B_2 & C_1A_{12}B_2 & 0\end{bmatrix}\begin{bmatrix}\delta_r u_2(0)\\ \delta_r u_2(1)\\ \delta_r u_2(2)\end{bmatrix} \tag{5}$$

Note that due to causality, the P_{ii}, and P_{ij} matrices are lower block triangular, and that in addition the matrices coupling the subsystems, $P_{ij}{}'$, have zero diagonal block elements.

We will assume that the number of output variables at each time step is the same as the number of input variables. Hence, the product C(k)B(k-1) is square, and we furthermore require that it be full rank. This is required for the existence of a solution. If there are more outputs than inputs in the original description of the problem, one must limit the number of output variables which one wishes to force to have zero tracking error. In this case, one has the option of choosing a different set of outputs each time step so that zero tracking error is obtained for all desired output variables at some but not at every time step. Note that making such changes from one time step to the next will create a time-varying system from a time-invariant one.

Ⅲ. THE DECENTRALIZED APPLICATION OF INDIRECT LEARNING CONTROL

The indirect learning control of reference [4], is designed to apply to time varying linear systems, and to apply to time invariant linear systems as a degenerate case. Equation (4) can be thought of as a system representation in modern control form

with the state vector being the history of the outputs for a repetition, with the identity matrix as the system matrix, and with the changes in the inputs from one repetition to the next as the control variables. Reference [4] shows how to apply indirect adaptive control in a centralized manner to such a modern control representation operating in the repetition domain. Here we consider various possible ways to apply indirect adaptive control ideas in a decentralized manner.
Since system i does not know what inputs are being used in other systems, in order to allow each system to accomplish its goal of learning, the first decentralized learning process considered here requires that at each repetition only one subsystem learns, and the remaining subsystems keep their learning control signals frozen. Hence, if at repetition r it is subsystem i's turn to learn, then equation (4) becomes

$$\delta_r \underline{y}_i = P_{ii} \delta_r \underline{u}_i \qquad (6)$$

Such input-output pairs obtained each time it is i's turn to learn, allow the decentralized learning controller for system i to estimate the matrix P_{ii} , call it $\hat{P}_{ii,r}$ Using this estimated matrix, the learning control law generates the change $\delta_r \underline{u}_i$ required to make a change $\delta_r \underline{y}_i$ that will cancel the error according to

$$\delta_r \underline{u}_i = \hat{P}_{ii,r}^{-1} (\underline{y}_i^* - \underline{y}_{i,r}) \qquad (7)$$

There are various choices for the estimation of this matrix, including the projection algorithm, the orthogonalized projection algorithm, and the recursive least squares algorithm. Ideally, each of these can be computed in real time from one time step to the next, so that at the end of a repetition the information is available for immediate use whenever the next repetition starts. An important freedom in the learning control problem is that there is no requirement that the computation be made in real time. There is no requirement that learning take place at every repetition, so that one can skip learning for a repetition while one waits for the needed computation to be completed. Note that one can make use of the lower block triangular nature of the estimated matrix in order to obtain the inverse in a recursive manner.
The centralized indirect learning control results in [4] guarantee zero tracking error without any requirement that the identified matrix $\hat{P}_{ii,r}$ converges to the true matrix. This result is analogous to standard results in adaptive control theory. We will not address such issues here. Instead we simply agree to introduce an independent $\delta_r \underline{u}_i$ if at some repetition, (7) produces a change in the leaning control input which is not independent of previous changes.
Here we consider the recursive least squares algorithm because it is relatively insensitive to noise, and because it can guarantee convergence in a finite number of steps when the data is noise-free and independent. The equations appropriate for (7) are given in [3]. Consider the computations made by the ith subsystem, and for the sake of simplicity of notation, we temporarily drop explicit indication of dependence on

i in the symbols used. Let $\hat{P}_{l,r}$ represent the column vector which is the transpose of the lth row of $\hat{P}_{ii,r}$, but with the zero elements to the right of the block diagonal deleted from the column vector. Let $\delta^{t}_{r}\underline{u}$ represent the quantity $\delta_r \underline{u}$ but with the elements deleted that are multiplied by these zero elements in the product $P_{ii}\delta_r \underline{u}_i$. And let $\delta^{l}_{r}\underline{y}$ represent the lth row of $\delta_r \underline{y}$. Then the recursive least squares update is

$$\hat{P}_{l,r} = \hat{P}_{l,r-1} + M_{l,r-2}\delta^{t}_{r}\underline{u}\left[\frac{\delta^{l}_{r}\underline{y} - (\delta^{t}_{r}\underline{u})^{T}\hat{P}_{l,r-1}}{1+(\delta^{t}_{r}\underline{u})^{T}M_{l,r-2}\delta^{t}_{r}\underline{u}}\right] \tag{8}$$

$$M_{l,r-1} = M_{l,r-2} - \frac{M_{l,r-2}\delta^{t}_{r}\underline{u}(\delta^{t}_{r}\underline{u})^{T}M_{l,r-2}}{1+(\delta^{t}_{r}\underline{u})^{T}M_{l,r-2}\delta^{t}_{r}\underline{u}} \quad ; \quad r \geq 2$$

The initial value $M_{l,0}$ is chosen as the identity matrix of the same dimension as the $\hat{P}_{l,r}$ corresponding $\hat{P}_{l,r}$. Note that this matrix need only be updated when the dimension of increases when l is increased, and the same $M_{l,r}$ can be used for all rows corresponding to the same time step in the multiple output case.

The decentralized indirect learning control algorithm based on the centralized indirect learning control algorithm of [4] can be summarized as follows. Only one subsystem learns during each repetition, while the other subsystems keep their learning control signals unaltered. As the repetitions progress, each subsystem gets its opportunity to learn in an order that is pre-chosen and known to each of the learning controllers. Then at the repetition for which the ith subsystem learns, the learning control law for subsystem i is: equation (8), together with equation (7) with a recursive computation of $\hat{P}^{-1}_{ii,r}$, and together with the requirement for independent changes of the learning control signal for this subsystem. In a later subsection we will study the convergence behavior of this decentralized learning control scheme. We will also develop a modified version of the algorithm requiring less computation, and producing faster convergence.

IV. NUMERICAL EXAMPLES

This paper presented various concepts for application of learning control in a decentralized manner, and showed that they can lead to guaranteed convergence to zero tacking error. These concepts still leave us with a number of choices for implementation. We can have each subsystem take turns learning, and we can choose how often they alternate. Or we can have the subsystems learn simultaneously, but do so in a wave, and in this case we can choose how fast to make the wave progress. In the identification process, we also have choices. One can use the recursive least squares method as was suggested above, or one can simply solve the simultaneous equations for the unknowns, which can be more reasonable when the number of unknowns is small, as is the case when learning in a wave. In this section we examine these options by studying several examples.

4.1. Subsystems Alternate Learning the Complete Trajectory

Figure 1 presents results when this decentralized learning process is used on the linearized model of the polar coordinate robot example, which was presented as example 2 in Section 3.5, with 10 time steps during the 1 second maneuver. In this figure and those that follow, subsystem 1 refers to control of radial displacement, measured in meters, subsystem 2 controls the rotation angle, given in radians. The top parts of this figure present repetition 1 which corresponds to using the feedback controller alone, and then repetitions 2 through 11 correspond to having subsystem 1 doing the learning. Without noise this number of repetitions is sufficient for system 1 to learn all elements of P_{11}. However, *in this example we use the usual recursive* least squares equation (8), which has a 1 in the denominator that is introduced to avoid the possibility of singularity. This means that the resulting identification is not exact in the noise free case except asymptotically. The initial condition for the recursive least squares is an a priori estimate of P_{ii} that is in error -- every element being 10% too high. The next two parts of the figure present the corresponding repetitions when subsystem 2 learns P_{22}, repetitions 12 through 21. Then the *subsystems alternate learning every repetition*, with subsystem 1 learning in repetition 22. Both linear plots and logarithmic plots are presented for these repetitions. By the 28th repetition the error is zero to the plotting accuracy of the linear plot.

Figure 2 simulates the same system, but the learning is alternated between subsystems starting from the first repetition, without separate repetitions allocated for each system to learn its own P_{ii}. The first repetition corresponds to feedback control only, and then in the first repetition subsystem 1 learns. The theory presented guarantees convergence for this case as well (in both cases a sufficiently small sample time must be used). On the linear plots, essentially zero tracking error is reached after 7 repetitions, which is much faster than the 28th repetition in the previous case. This difference would be even more extreme if there were more time steps in the trajectory. We conclude that it is best to start alternating from the first repetition..

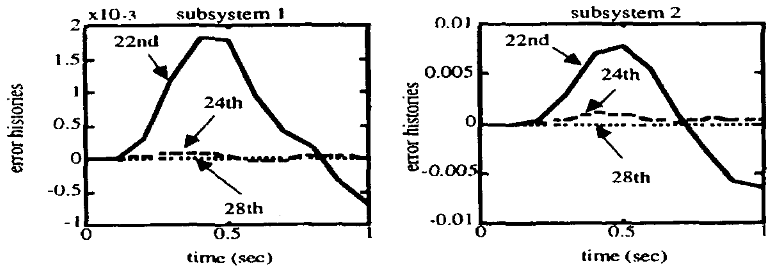

Fig. 1 Error histories for the linearized polar coordinate robot using alternate learning after initial identification of by each subsystem.

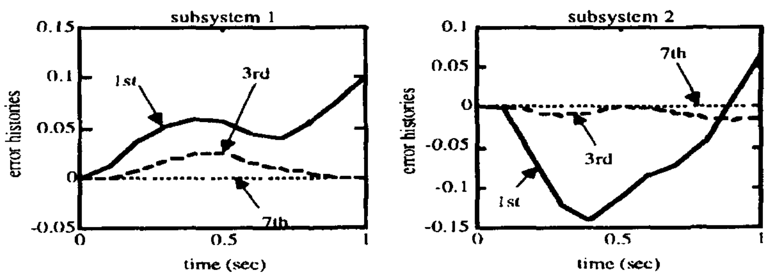

Fig. 2 Error histories for decentalized learning in the linearized polar coordinate model using alternate learning every repetition starting with repetition 2.

4.2. Subsystems Learn Progressive Time-Steps Simultaneously

In the decentralized learning method of the above examples, each subsystem eventually needs to identify all elements of its P_{ii} matrix. When the subsystems learn simultaneously, one time step at a time, the identification is limited to having each subsystem find the instantaneous value of its own input-output matrix product. Rather than use the recursive least squares approach of (8), here we compute this product as follows:

$$u_{i,r}(k) = u_{i,r-1}(k) + \delta_r u_i(k)$$

where

$$\delta_r u_i(k) = [E_r(C_i(k+1)B_i(k))]^{-1}.(y_i^*(k+1) - y_{i,r-1}(k+1)) \tag{9}$$

The estimate $E_{r+1}(C_i(k+1)B_i(k))$ of $C_i(k+1)B_i(k)$ of subsystem i is chosen as the latest value according to

$$E_{r+1}(C_i(k+1)B_i(k)) = \frac{\delta_r y_i(k+1)}{\delta_r u_i(k)} \tag{10}$$

Care must be taken to avoid singularity problems in performing this division, when the learning control signal approaches convergence. We will vary the number of repetitions used for learning at each time step. If the system were truly a linear time-varying discrete time system with no coupling between subsystems in the input and output matrices, one would prefer to average the set of numbers obtained from (10) for this time step, rather than use the latest value, in order to average the effects of noise in the data. However, noise may not be the issue. Other considerations suggest using the latest value. In the process of discretizing the linearized differential equations, coupling was introduced between the subsystems, which for small sample times is small but not zero. Also, the actual system is nonlinear, and the linearized differential equation model considered here to model the system is linearized about the desired trajectory. Hence, it is only as the system approaches the desired trajectory that the estimate of the $C_i(k+1)B_i(k)$ product approaches the true value.

Figure 3 presents results of learning in the same linearized polar coordinate model as above, with the same sample time. The first repetition is with feedback control only, and then the wave of learning starts, using two repetitions for each time step. After the wave finishes the final time step at repetition 21, a second wave of learning is performed for repetitions 22 through 41. The computations use noise free data computed from the time varying difference equation. Due to the effects mentioned above, the error at the end of the first wave is not zero, although the error is much improved over feedback alone. During the second wave, the error behind the wave is made very small, although the error in front of the wave is somewhat accentuated temporarily during the learning process. Figure 4 shows the corresponding results when four repetitions are used at each time step, and only one wave of learning is used, for the same total of 41 repetitions. During repetitions 1 through 21 in Fig. 3, one has somewhat better and more uniform error histories than in Fig. 4, but during the second wave of learning for repetitions 22 through 41 in Fig. 3 the errors do not decay monotonically and can be worse than in Fig. 4. So, the double wave of learning with two repetitions per time step has better initial behavior, but one pays some price later with worse transient behavior after the transients have decayed significantly.

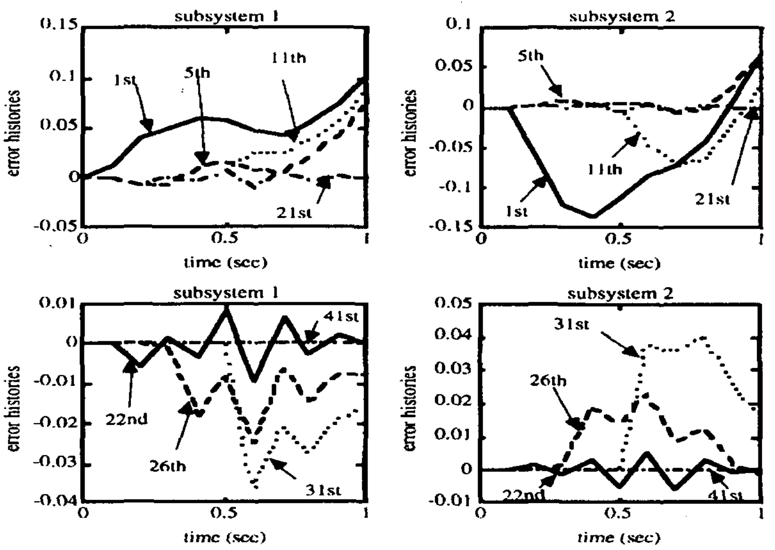

Fig. 3 Error histories for learning in a wave in the linearized polar coordinate model, progressing one time step every two repetitions, and performing two waves of learning.

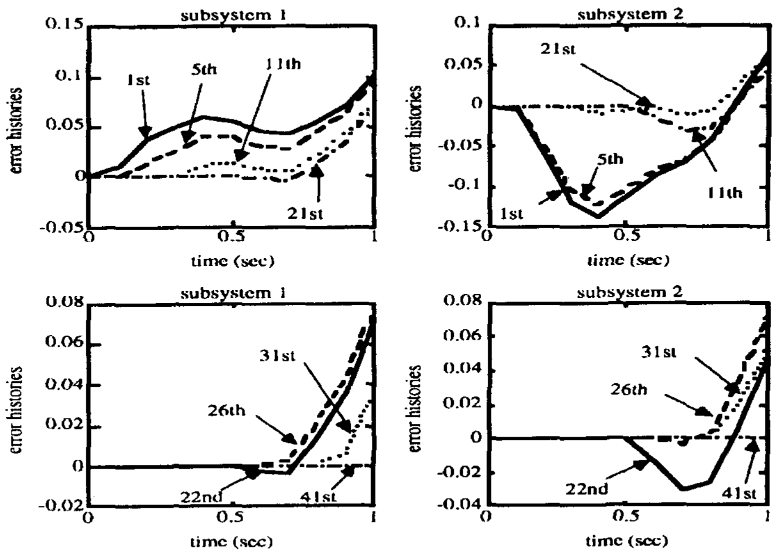

Fig. 4 Error histories for learning in a wave, progressing one time step every four repetitions, and performing one wave of learning.

The concept of learning in a wave was first introduced in Ref. [15] for centralized linear learning control as a method to improve the learning control transients. It was introduced here for a different reason, as a way to decouple the subsystems when the subsystems are learning simultaneously. It may have advantages in producing better transients as well. This may be true when one has only poor a priori knowledge of the system, but the fact that the learning transients are better in Figs. 2 than in 3 indicates that when one knows the P_{ii} matrices to within 10%, there is no need to learn in a wave for purposes of improving transients.

All of the above results used the time varying linearized difference equations for horizontal motion of the polar coordinate robot model. Figures 5 and 6 apply decentralized learning control to the nonlinear differential equation model in order to

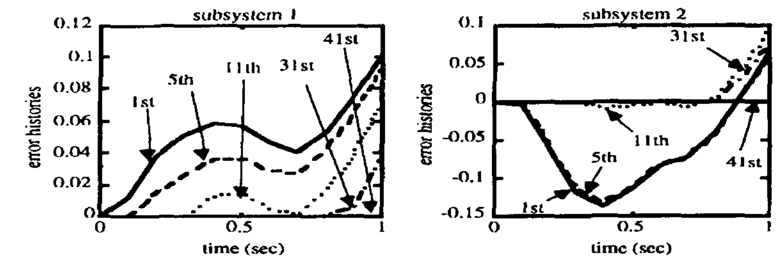

Fig. 5 Error histories for one wave of learning in the nonlinear polar robot model, using four repetitions for learning at each time step, with sample time T=0.1 sec.

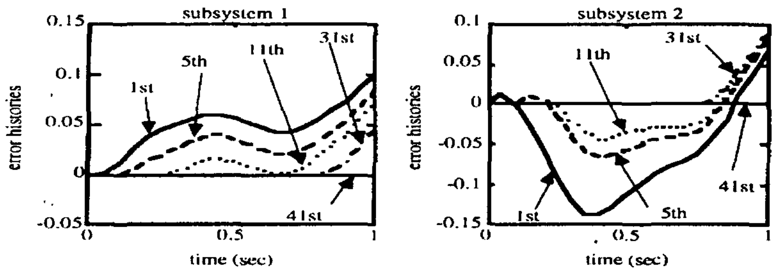

Fig. 6 Error histories for one wave of learning in the nonlinear polar robot model, using two repetitions for learning at each time step, with sample time T=0.05 sec.

see the effects of nonlinearities. Figure 5 repeats Fig. 4 for these nonlinear differential equations, i.e., it uses learning in a wave with four repetitions for each time step, before letting the wave progress a time step. As before, the wave of learning is complete after 41 repetitions. The error histories are similar to those for the linearized time varying model in Fig. 4, although for subsystem 2 the results are somewhat worse throughout for repetition 5, and somewhat worse at the end in repetition 11. In the nonlinear case, the multiple repetitions for each time step try to correct for not only the coupling in the input matrices introduced in the time discretization, but also for system nonlinearities. Figure 6 cuts the sample time in half to 0.05 seconds, which decreases both the coupling in the input matrices and the influence of nonlinearities during one time step. The number of repetitions per time step is decreased to two, so that the wave of learning is again finished at the end of 41 repetitions. Thus, we study the trade-off between decreasing these coupling effects by decreasing the number of time steps, versus decreasing these coupling effects by repeated repetitions at the same time step. Comparing Figs. 5 and 6, does not give a clear indication of which approach is best. For subsystem 2, using the smaller sample time results in a substantial improvement in performance at repetition 5, but a somewhat worse error in repetition 11.

V. CONCLUDING REMARKS

In this paper, two classes of methods were developed for decentralized indirect learning control based on different agreements between the subsystems as to when each subsystem learns. In the first, the subsystems agree to alternate the learning of the complete trajectory with the repetitions. The second algorithm has the appeal of learning in a wave progressing from the start of the p-step process and progressing to the end, with all subsystems learning simultaneously the same time step. Fewer parameters need to be identified when learning in a wave than in the alternate learning approach, and this distinction is even more extreme in the case of time-invariant systems. In [15], learning in a wave similar to this was used with the integral control based learning control, as one technique to have control over the size of the transients in the learning process. Numerical results not presented here indicate that learning in a wave is preferable to the alternate learning method when one has very poor a priori knowledge of the system, but the reverse is true if one has a reasonable system model. Examples also illustrate the trade-offs between how many repetitions are used for each time step when learning in a wave, how many waves of learning to use, and how small a sample time to use.

REFERENCES

[1] S. Arimoto, S. Kawamura, and F. Miyasaki, "Bettering Operation of Robots by Learning," Journal of Robotic Systems, Vol. 1, No. 2, 1984, pp. 123-140.

[2] R. H. Middleton, G. C. Goodwin, and R. W. Longman, "A Method for Improving the Dynamic Accu racy of a Robot Performing a Repetitive Task," International Journal of Robotics Research, Vol. 8, No. 5, October 1989, pp.67-74.

[3] M. Phan and R. W. Longman, "A Mathematical Theory of Learning Control for Linear Discrete Mul tivariable Systems," Proceedings of the AIAA/AAS Astrodynamics Conference, Min neapolis, Minnesota, August 1988, pp. 740-746

[4] M. Phan and R. W. Longman, "Indirect Learning Control with Guaranteed Stability," Proceedings of the 1989 Conference on Information Sciences and Systems, Johns Hopkins University, Balti more, MD, 1989, pp.125-131.

[5] M. Phan and R. W. Longman, "Learning System Identification," Mod eling and Simulation, Instrument Society of America, Vol. 20, Part 5, 1989, pp. 1857-1864.

[6] M. Phan, Y. Wei, L. G. Horta, J.-N. Juang, and R. W. Longman, "Learning Control for Slewing of a Flexible Panel," Dynamics and Control of Large Structures, Proceedings of the Seventh VPI&SU Symposium, L. Meirovitch, Editor, May 1989, pp. 39-52.

[7] M. Phan, J.-N. Juang, and R. W. Longman, "Recent Developments in Learning Control and System Identification for Robots and Structures," Dynamics of Flexible Structures in Space, Proceedings of the First International Conference, Cranfield, England, May 1990.

[8] C.-K. Chang, R. W. Longman, and M. Phan, "Techniques for Improving Transients in Learning Control Systems," Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Durango, Colorado, August 1991, Advances in the Astronautical Sciences.

[9] S. C. Lee, R. W. Longman, and M. Phan, "Linear Decentralized Learning Control," Paper No. 91-447, Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Durango, Colorado, August 1991, Advances in the Astronautical Sciences.