A PARALLEL NETWORK IMPLEMENTATION OF THE GENERALIZED SECANT LEARNING-ADAPTIVE CONTROLLER

Homayoon S.M. Beigi
T.J. Watson Research Center
International Business Machines
P.O. Box 704
Yorktown Heights, New York 10598

Abstract

When control systems are given a task to perform repeatedly, they will usually repeat the same errors in executing the command, except for some random noise effects. Learning controllers on the other hand can improve their performance at a given task with each repetition of the task. Learning control algorithms have used many different approaches to the solving this problem, such as analogues of integral control in the repetition domain to eliminate errors [1] and conversions of the adaptive control approaches to the learning control problem [2]. This paper presents a method which is a combination of a parallel network for solving the control and the generalized secant method of solving a set of linear equations for solving the parameter estimation problem.

1 Introduction

Many practical applications of control theory involve systems which are repeatedly asked to perform the same task. Examples include a large number of manufacturing problems as well as tracking problems for robots on an assembly line. Standard controller design methods often produce systems that do an imperfect job of executing a command and when these commands are repeated, the systems repeat the same errors everytime. It is perhaps a bit primitive to persist in repeating the same errors. In the last few years a theory of learning control has been developed in the literature, generating controllers that can learn from their previous experience at performing a specific task, see [3].

In this paper, we consider the learning control problem for discrete systems, a system of linear algebraic equations relating the change in the state of the system to the change in the control action from one repetition to another. Then we utilize a parallel network to generate the change in the control input from one repetition of the task to the next using the best of our knowledge of the system. After Application of this change in control to the actual dynamic system, a parameter estimation is done using an extension to the generalized secant update. [4]

2 Problem Formulation

Consider a general discrete-time linear time-variant or time-invariant system,

$$x^k(t+1) = A(t)x^k(t) + B(t)u^k(t) + w^k(t) \qquad (1)$$

$$\begin{aligned} y^k(t+1) &= C^k(t+1)x^k(t+1) \\ t &= 0,1,2,\ldots,p-1 \\ k &= 0,1,2,\ldots \end{aligned} \qquad (2)$$

where the state x is n-dimensional, the control u is m-dimensional, the output y is q-dimensional $(q > m)$, t is the time step in a p-step repetitive operation, and k is the repetition number. Also, A, B, C, and $w^k(t)$ are assumed to be unknown – otherwise one could determine in advance what control to use to minimize the tracking error. For simplicity, the dimension n of the system is assumed known, but generalization to just knowing an upper bound on n is easily considered.

In the learning control problem (as contrasted with the repetitive control problem in [1]) the system is assumed to always start from the same initial state in each repetition of the task. Matrix A includes any state or output feedback control present in the system and the symbol u^k is reserved for the signal added to the control for learning purposes. A time-variant model is considered because many applications such as in robotics and machining involve nonlinear dynamic systems which, when linearized, produce linear models with coefficients that vary with the time step and vary in the same manner each repetition. In such repetitive operations it is often the case that there will be disturbances $w^k(t)$ that repeat with each repetition of the task and the learning can be made to also correct for this source of errors in a natural way. One of the purposes of a feedback control is to handle any non-repetitive disturbances, and these will be ignored for purposes of designing the learning controller.

Based on the system equations 1 and 2, reference [4] gives the transition between two consecutive repetitions as,

$$\mathbf{y}^{k+1} - \mathbf{y}^k = P\,\underbrace{\left(\mathbf{u}^{k+1} - \mathbf{u}^k\right)}_{\mathbf{v}^k} \tag{3}$$

P is given in the appendix (equation 13) and,

$$\mathbf{y}^k = \left[\begin{array}{cccc} y^{k^T}(1) & y^{k^T}(2) & \cdots & y^{k^T}(p) \end{array}\right]^T \tag{4}$$

$$\mathbf{u}^k = \left[\begin{array}{cccc} u^{k^T}(0) & u^{k^T}(1) & \cdots & u^{k^T}(p-1) \end{array}\right]^T \tag{5}$$

Defining $\mathbf{e}^k \equiv \mathbf{y}^k - \mathbf{y}_d$ where $\mathbf{y}_d$ is the discrete desired trajectoy, the recursive generalied secant learning adaptive control and estimation is,

$$\mathbf{v}^k = -P^{k^\dagger}\mathbf{e}^k \tag{6}$$

$$P^{k+1} = P^k + \frac{\left(\mathbf{e}^{k+1} - \mathbf{e}^k - P^k\mathbf{v}^k\right)\mathbf{z}^{k^T}}{\mathbf{z}^{k^T}\mathbf{v}^k} \tag{7}$$

where $\mathbf{z}^k$ are secant projection vectors and are defined in [4].

The most computation intensive part of this algorithm is the evaluation of the pseudo-inverse in equation 6. Although, the nature of this computation is such that it could be done between two repetitions, it would be nice to be able to solve equation 6 in a more effective and quicker fashion. By nature, pseudo-inversion is a minimization problem such that the Frobenius norm of the error $\|P^k\mathbf{v}^k + \mathbf{e}^k\|$ is minimized. This error, E, could also be formulated in terms of a minimization problem in $\mathbf{v}^k$,

$$\frac{d\mathbf{v}^k}{d\xi} = -M(\xi)\nabla E(\mathbf{v}^k) \tag{8}$$

$$\nabla E(\mathbf{v}^k(\xi)) = P^{k^T}(P^k\mathbf{v}^k(\xi) + \mathbf{e}^k) \tag{9}$$

$$\mathbf{v}^k(0) = \mathbf{v}_0^k \tag{10}$$

This is a possible minimization formulation of the pseudo-inverse. If the weighting factor $M(\xi)$ is set to the identity matrix, the minimization problem reduces to a steepest descent problem. For convergence of the minimization problem, M should be chosen to be positive definite. Reference [5] shows a network which recursively goes through the calculations of equations 8- 10. Figure 1 (in the appendix), shows a parallel network which basically solves this minimization problem. However, since this is a parallel analog network, the calculations are done much more quickly.

3 Convergence

A proof of convergence of equations 6 and 7 is given in [3, 4]. This proof of convergence is independent of the solution for the change in the control vector. The change in the control input vector converges if M is picked to be positive definite. This proof could be found in any mathematical handbook or reference [5] Therefore, the two problems are almost decoupled and if both converge, the system will converge.

4 Simulations and Results

The learning control algorithm based on the generalized secant method was tested on two nonlinear dynamic systems. System 1 was a nonlinear mass- spring-dashpot system given by equation 11,

$$m\ddot{\alpha} + c\,(1 + \gamma|\alpha|)\,\dot{\alpha} + (k|\alpha|)\,\alpha = T \tag{11}$$

System 2 was a damped pendulum with the following differential equation:

$$ml^2\ddot{\alpha} + c\dot{\alpha} + mglsin\alpha = T \tag{12}$$

Figure 2 shows the demanding non-linear trajectory which was requested from both systems. The parameters of equation 11 were, $m = 0.1kgr$, $c = 0.1N/(m/s)$, $k = 0.1N/m$, and $\gamma = 0.2$. Figure 3 shows a plot of the sum of squares of errors as a function of the repetition number for this system. Figures 4 is a similar plot for the pendulum of equation 12 with $m = 1kgr$, $c = 1N/(m/s)$, and $l = 0.1m$.

Figure 2: Desired Trajectory

Figure 3: Squares of Errors (System 1)

The performances reflected through these figures when compared with those in [4] where the actual pseudo-inversion was done, are almost identical. This is due to the fact that the network simulation for solving the pseudo-inverse through the weighted steepest descent converged very quickly and to a high precision. This shows that once the pseudo-inversion is done through the use of a parallel network, almost no accuracy is sacrificed. The parallel network, however, will reduce the amount of computation (wait) between two consequent repetitions of a task.

Figure 4: Squares of Errors (System 2)

5 Conclusion

This paper has combined the generalized secant learning controller with a parallel scheme for evaluating the pseudo-inversion required in that learning control algorithm. This makes the waiting period between two consequent repetitions shorter. The convergence of the two methods is shown to be almost independent. The results of simulations and comparisons to results in [4] with the actual pseudo-inversion, support the fact that using this parallel network for obtaining the pseudo-inverse is very practical. However, there is one problem of practicality which is still outstanding and that is the fact that for each time step there should be a connection built in the network. However, if networks are developed for this type of a controller, a maximum size network can work for any number of time steps less than or equal to its maximum capacity.

References

[1] R.H. Middleton, G.C. Goodwin, and R.W. Longman, "A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task," The International

Journal of Robotics Research, Vol. 8, No. 5, October 1989, pp. 67-74.

[2] C. James Li, Homayoon S.M. Beigi, Shengyi Li, and Jiancheng Liang, "A Self-tuning Regulator with Learning Parameter Estimation," Robotics Research, Dynamic Systems and Control Vol. 26, The ASME Winter Annual Meeting, Dallas, TX, Nov. 25-30, 1990, pp. 1-6.

[3] Homayoon S.M. Beigi, "Neural Network Learning and Learning Control Through Optimization Techniques", Doctoral Thesis, Columbia University, 1991.

[4] Homayoon S.M. Beigi, C. James Li and R.W. Longman, "Learning Control Based on Generalized Secant Methos and Other Numerical Optimization Methods," Sensors, Controls, and Quality Issues in Manufacturing, ASME: Atlanta,PED-Vol.55, pp. 163-175, December 1991.

[5] Andrzej Cichocki and Rolf Unbehauen, "Neural Networks for Solving Systems of Linear Equations and Related Problems," IEEE Transactions on Circuits and Systems, Fundamental Theory and Applications, Vol. 39, No.2, pp. 124-138, Frbruary 1992.

Appendix

The system transition matrix:

$$P = \begin{bmatrix} C(1)B(0) & 0 & \cdots & 0 \\ C(2)A(1)B(0) & C(2)B(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C(p)A(p-1)\cdots A(1)B(0) & \cdots & \cdots & C(p)B(p-1) \end{bmatrix} \tag{13}$$

Figure 1: Parallel Network for Solving the Change in control v^k