DSC-Vol. 26, Robotics Research — 1990
ASME 1991

A SELF-TUNING REGULATOR WITH LEARNING PARAMETER ESTIMATION

C. James Li, Homayoon S. M. Beigi, Shengyi Li, and Jiancheng Liang
Department of Mechanical Engineering
Columbia Universtiy
New York, New York

Abstract

This paper presents a learning adaptive controller which improves the tracking performance of itself while performing repetitive tracks. The controller is a self-tuning regulator based on learning parameter estimation. Simulation results show that the tracking error is reduced near monotonically over the history of repeating a prescribed task by the proposed controller as applied to a pendulum . Experimentally, the controller was used to control the movement of a nonlinear piezoelectric actuator which is a part of the tool positioning system for a diamond turning lathe. Experimental results also show that the controller is able to reduce the tracking error through the repetition of the task.

I. Introduction

One of the major factors contributing to performance limitations of today's industrial automated machines/processes is the restricted capabilities of their control systems. A wide class of these controllers employ constant pre-defined gains and do not take into consideration the non-linear dynamics in these machines. The result is that these machines are not being utilized to their full potential in terms of their speed and precision. With a more sophisticated control strategy, it is possible to compensate for the complicated effects of non-linearities which have in the past been considered as mere disturbances in most systems.

Several advanced schemes have been proposed for an improved performance, which would generate control actions to compensate for the aforementioned non-linear dynamics. These include non-linear feedback control [1], feedforward control [2], resolved motion control [3,4], sliding mode control [5], repetitive control [6,7] and learning control [8]. In particular, adaptive controls have drawn a lot of attention in various applications [9-12]. One class of these adaptive controllers is the self-tuning regulator [9].

A self-tuning regulator consists of a parameter estimator and a controller. The parameter estimator estimates the parameters of an approximated model of the controlled system by utilizing a recursive estimation scheme. Based on the approximate model and the estimated parameters, the controller adjusts its actions to maintain its performance. Therefore, the performance of a self-tuning regulator depends greatly on that of the parameter estimator employed.

For example, the existing recursive least square (RLS) algorithms do not utilize their past experience in estimating repetitive parameters. The implication is that the system will keep making the same errors at corresponding times in the duration of each repetition.

Another limit of these estimators is that they have to keep the changes of estimations small between neighboring sample instants to maintain their immunity to noise and disturbances. Consequently, the estimators cannot respond to large changes of parameters quickly.

The objective of this paper is to present a learning adaptive control scheme based on a learning estimator which utilizes the information from past performances of a repetitive task to refine the estimate of the plant parameters. Consequently, the learning adaptive controller improves its performance throughout the repetitions. Since the adaptation of parameters at all sampling instants is made over the repetitions of the task, the estimator is free to follow the changes of parameters over time.

II. The Self-tuning Regulator

A self-turning regulator has the structure shown in figure 1. The recursive parameter estimator observes the input and output to and from the plant and provides estimates of the parameters of the model of the plant. These estimates are then used to compose control actions applied to the plant. References 12-14 provide a more detailed description of the self-turning regulator and parameter estimation.

The self-turning regulator has been applied to the control of nonlinear systems such as robots based on linear discrete uncoupled models of the plant. The rationale of this approach is that linearization of the nonlinear governing equations at each sampling instants would yield linear difference equations of time-varying parameters.

A second order example of such an equation is:

$$\alpha(t) + \theta_1(t)\,\alpha(t-1) + \theta_2(t)\,\alpha(t-2) = q^{-d}\,(\theta_3(t)\,u(t) + \theta_4(t)\,u(t-1)) + v(t) \quad (1)$$

where q^{-d} is the backward shift (or delay) operator such that, $q^{-d}\,\alpha(t) = \alpha(t-d)$; d is the delay, u(t) is the input at time step t, $\alpha(t)$ is the output at time step t, v(t) denotes the equation error and θ's are parameters.

Let's assume that d=1 and solve for $\alpha(t)$ from equation (1):

$$\alpha(t) = \underline{\theta}^{'}(t)\,\underline{\phi}(t) + v(t) \quad (2)$$

where,

$\underline{\theta}^{'}(t) = [\theta_1(t), \theta_2(t), \theta_3(t), \theta_4(t)]$ is the parameter vector at time t and

$\underline{\phi}^{'}(t) = [-\alpha(t-1), -\alpha(t-2), u(t-1), u(t-2)]$ is the linear regression

vector.

A self-tuning regulator models a given dynamic system with a difference equation in the form of (2) and uses a recursive estimator to estimate the parameter vector $\underline{\theta}(t)$ at each sampling instant and adjusts its control action accordingly. Therefore, the overall performance of the control system is very much dependent on how close these estimates are to the real system parameters.

One popular parameter estimator is the Recursive Least Squares (RLS) parameter estimator which minimizes the sum of squares of the equation errors v(t) over time. Therefore, at time step k, the function,

$$v_k(\theta) = \sum_{t=0}^{k} \lambda^{k-t} v^2(t) \qquad (3)$$

is minimized, where λ is a forgetting factor which is a measure of how fast the old data is forgotten. The range of λ is between 0 and 1 and as it becomes smaller the old data is forgotten more quickly. Using equation (2), the expression for v_k can be written:

$$v_k(\theta) = \sum_{t=0}^{k} \lambda^{k-t} [\alpha(t) - \underline{\theta}^{'}(t)\,\underline{\phi}(t)]^2 \qquad (4)$$

Equation (4) is then minimized with respect to $\underline{\theta}$ and a parameter, $\underline{\theta}$, vector is obtained for time step k.

To recursively find a $\underline{\theta}$ vector which would minimize (4), the following algorithm can be used:[13,14]

$$\underline{\theta}(k) = \underline{\theta}(k-1) + \underline{L}(k-1)\,[\,\alpha(k) - \underline{\theta}^{'}(k-1)\,\underline{\phi}(k)\,] \qquad (5a)$$

$$\underline{L}(k-1) = \frac{P(k-1)\,\underline{\phi}(k)}{\lambda + \underline{\phi}^{'}(k)\,P(k-1)\,\underline{\phi}(k)} \qquad (5b)$$

$$P(k) = \frac{[\,I - \underline{L}(k-1)\,\underline{\phi}^{'}(k)\,]\,P(k-1)}{\lambda} \qquad (5c)$$

where, $\underline{L}$ is an intermediate vector which is calculated each time to avoid matrix inversions and P is the covariance matrix which signifies the degree of confidence in the accuracy of the parameter estimates and is started off as a diagonal matrix with large diagonal elements to minimize the effect of the initial guess for the parameter vector.

The forgetting factor λ is used to give less weight to older data so the estimator can track the variation of parameters. Typically an initial estimate of the parameters has to be given to the estimator to start the recursion. Frequently, this estimate is different from the real value of the parameters. Therefore, a tracking error will be produced until the parameter estimator catches up which may never be achieved if the parameters are varying from one sampling instant to the next. Consequently, a persistent tracking error resulted from the lagging of the estimate of the parameters at all times is inevitable.

If the same task is executed repeatedly, the controller repeats without looking back at how it had performed previously. Therefore, the controller will make the same errors at corresponding times of repetitions.

The following proposition of a learning estimator provides a solution which will allow the use of the information acquired from previous repetitions to improve the estimate of the parameters over repetitions of a task. With better estimates of parameters the self-turning controller will be able to reduce tracking errors over the repetitions of a task.

III. Learning Recursive Least Squares Estimator (LRLS):

We will use Fig. 2 to illustrate the idea behind the proposed learning parameter estimator. Each dot in Fig. 2 represents one sampling instant. Each row of (p+1) dots represents a run of the task which is numbered from repetition 0 to repetition r.

Let us look at time step k of repetition r, and assume that there is a linear difference equation which adequately describes the dynamics of the plant.

$$\alpha^r(k) = \underline{\phi}^{r'}(k)\;\underline{\theta}^r(k) + v^r(k) \qquad (6)$$

where,

$\underline{\theta}^{r'}(k) = [\,\theta^r{}_1(k)\,,\,\theta^r{}_2(k)\,,\,\theta^r{}_3(k)\,,\,\theta^r{}_4(k),\,\ldots\ldots\,]$,

$\underline{\phi}^{r'}(k) = [\,-\alpha^r(k-1)\,,\,u^r(k-1)\,,\,-\alpha^r(k-2)\,,\,u^r(k-2),\,\ldots\ldots\,]$,

and $v^r(k)$ denotes the equation errors.

For r=0, 1, 2,....r, equation (6) gives a set of linear algebraic equations which may be written in the vector matrix form as

$$\underline{\alpha}^r(k) = \Phi^r(k)\underline{\theta}^r(k) + \underline{v}^r(k) \qquad (7)$$

where,

$\underline{v}^{r'}(k) = [v^0(k),\, v^1(k),\, v^2(k),\ldots.v^r(k)]$

$\underline{\alpha}^{r'}(k) = [\alpha^0(k),\, \alpha^1(k),\, \alpha^2(k),\ldots..\alpha^r(k)]$

and

$$\Phi^{r'}(k) = \left[\underline{\phi}^0(k), \underline{\phi}^1(k), \ldots\ldots, \underline{\phi}^r(k)\right]$$

The least square method is concerned with determining the estimate of $\underline{\theta}$, that is $\underline{\theta}$ defined as the value that minimizes the sum of the squares of the equation errors. Thus calculates $\underline{\theta}$ so that

$$J = [\,\underline{\alpha}^r(k) - \Phi^r(k)\underline{\theta}^r(k)]\,[\underline{\alpha}^r(k) - \Phi^r(k)\underline{\theta}^r(k)]$$

$$= \sum_{j=0}^{r} [\alpha^j(k) - \underline{\phi}^{j'}(k)\,\underline{\theta}^j(k)]^2 \qquad (8)$$

is minimized.

Let us differentiate the quadratic cost function eq. (8) to find the value of $\underline{\theta}^r(k)$ that minimize the function. Thus we have

$$\frac{\partial J^r_k(\underline{\theta}^r(k))}{\partial\, \underline{\theta}^r(k)} = -\Phi^{r'}(k)\,[\underline{\alpha}^r(k) - \Phi^r(k)\underline{\theta}^r(k)] = 0 \qquad (9)$$

and the least square estimate of $\underline{\theta}$ is

$$\hat{\underline{\theta}}^r(k) = [\Phi^{r'}(k)\Phi^r(k)]^{-1}\Phi^{r'}(k)\,\underline{\alpha}^r(k) \qquad (10)$$

The number of rows, r, of the matrix $\Phi^r(k)$ in (7) can be very large and is successively supplemented by new rows of data obtained by new repetitions. It is uneconomical to repeat in all repetitions of sampling instant k the calculation of parameters with all past data stored in the matrix $\Phi(k)$, namely, if the number of rows was increased by one only.

In order to reduce the number of numerical operations, we shall derive a recursive type of estimation, the following denotation will be used.

$$\underline{\alpha}^{r+1}(k) = \begin{bmatrix} \underline{\alpha}^r(k) \\ \alpha^{r+1}(k) \end{bmatrix} \qquad \Phi^{r+1}(k) = \begin{bmatrix} \Phi^r(k) \\ \underline{\phi}^{r+1}(k) \end{bmatrix} \qquad (11)$$

where $\underline{\alpha}^r(k)$ and $\Phi^r(k)$ are defined in eq. (7). The least square estimate of parameters is according to (10).

$$\hat{\underline{\theta}}^{r+1}(k) = [\Phi^{r+1'}(k)\Phi^{r+1}(k)]^{-1}\Phi^{r+1'}(k)\underline{\alpha}^{r+1}(k) \qquad (12)$$

$$= [\Phi^{r'}(k)\Phi^r(k) + \underline{\phi}^{r+1}(k)\underline{\phi}^{r+1'}(k)]^{-1}[\Phi^{r'}(k)\underline{\alpha}^r(k) + \underline{\phi}^{r+1}(k)\,\alpha^{r+1}(k)]$$

Similar to the derivation of equation (5), it is straightforward to write this estimate of parameters in the following formulas:

$$\hat{\underline{\theta}}^{r+1}(k) = \hat{\underline{\theta}}^r(k) + \underline{L}^r(k)[\alpha^{r+1}(k) - \hat{\underline{\theta}}^{r'}(k)\underline{\phi}^{r+1}(k)] \qquad (13a)$$

$$\underline{L}^r(k) = \frac{P^r(k)\,\underline{\phi}^{r+1}(k)}{1 + \underline{\phi}^{r+1'}(k)\,P^r(k)\,\underline{\phi}^{r+1}(k)} \qquad (13b)$$

$$P^{r+1}(k) = [\,I - \underline{L}^r(k)\;\underline{\phi}^{r+1'}(k)\,]\;P^r(k) \qquad (13c)$$

The above derivation can be easily extended to minimize eq. (14a) which includes γ as a forgetting factor to determine how fast old data from previous repetitions is forgotten.

$$J = \sum_{j=0}^{r+1} \gamma^{r-j}[\alpha^j(k) - \underline{\phi}^{j'}(k)\,\underline{\theta}^j(k)]^2 \qquad (14a)$$

$$\hat{\underline{\theta}}^{r+1}(k) = \hat{\underline{\theta}}^{r}(k) + \underline{L}^r(k)[\alpha^{r+1}(k) - \hat{\underline{\theta}}^{r'}(k)\underline{\phi}^{r+1}(k)] \qquad (14b)$$

$$\underline{L}^r(k) = \frac{P^r(k)\,\underline{\phi}^{r+1}(k)}{\gamma + \underline{\phi}^{r+1'}(k)\,P^r(k)\,\underline{\phi}^{r+1}(k)} \qquad (14c)$$

$$P^{r+1}(k) = \frac{[\,I - \underline{L}^r(k)\;\underline{\phi}^{r+1'}(k)\,]\;P^r(k)}{\gamma} \qquad (14d)$$

It is evident that the estimate corresponding to the (r+1)th repetition is equal to the estimate from the previous repetition corrected by a term proportional to $[\alpha^{r+1}(k) - \underline{\theta}^{r'}(k)\,\underline{\phi}^{r+1}(k)]$. The product $\underline{\theta}^{r'}(k)\,\underline{\phi}^{r+1}(k)$ may be considered as the prediction of the value $\alpha^{r+1}(k)$ based on the estimate of parameter $\underline{\theta}^r(k)$ and on the set of measurements $\phi^{r+1}(k)$. This predicted value equals to the value $\alpha^{r+1}(k)$ only if $\underline{\theta}^r(k) = \underline{\theta}(k)$ (the estimate equals to the true parameters value) and if the measurement noise is absent.

The factor γ may be selected in the interval $0 < \gamma \leq 1$. With $\gamma = 1$, all sampled data u^r, α^r, r = 0,1... are equally weighted. With $\alpha << 1$, a large weight is placed on recently acquired data by rapid weighting out of samples from previous repetitions.

For a repetitive task which takes p sampling intervals (p+1 sampling instants) to complete, one would need (p+1) linear difference equations to approximate the real equation of motions. The (p+1) sets of parameters of these difference equations can be updated independently by equation (14) from one repetition to the next. Since this updating does not have to be done on-line as long as it is done before the next repetition is started, the demand on the on-line computing power is much less than traditional adaptive controllers.

The details concerning the starting of the recursive updating is described now. Depending on the available computing power, one has two alternatives. If one has enough computing power to calçulate equation (5) in real time, one should start the first execution of the desired trajectory (r=0) using the adaptive controller to obtain the least possible tracking error at the first run. The parameter vector $\underline{\theta}$ and the matrix P yielded by equation (5) at every sampling instant should be recorded and be recursively updated using equation (14) repetition after repetition later. If real time computation of equation (5) is not feasible, a simpler controller can be employed to try out the execution for the first time. In the mean tine, measurements of inputs and outputs will be recorded. After completing the task for the first time, these measurement shall be used by equation (5) to compete the θ and P for every sampling instant. Then, they will be updated over repetitions by eq. (14).

IV. The Control Law:

Since the control law is not the emphasis of this research, the One-Step-Ahead control law was selected for its simplicity [9]. In general, the self-tuning regulator is very flexible with control laws. Virtually any technique (pole-placement, minimum variance, etc.) can be accommodated. The One-Step-Ahead control law is the application of an inverse model while an estimate to the plant parameters is known. For example, if the plant can be approximated by a second order linear time-variant difference equation such as:

$$\alpha^r(k) + \theta^{r-1}{}_1(k)\,\alpha^r(k-1) + \theta^{r-1}{}_2(k)\,\alpha^r(k-2) = \theta_3{}^{r-1}(k)\,u^r(k-1) + \theta^{r-1}{}_4(k)\,u^r(k-2) \qquad (15)$$

If $\alpha_d(k)$ denotes the desired position at time step k, then at time step k-1, the control,

$$u^r(k-1) = [\alpha_d(k) + \theta_1^{r-1}(k)\,\alpha^r(k-1) + \theta_2^{r-1}(k)\,\alpha^r(k-2) - \theta_4^{r-1}(k)u^r(k-2)] / \theta^{r-1}{}_3(k) \qquad (16)$$

should be applied.

V. Simulation and Experimental Results:

In both simulation and experimental studies, the controller described in fig. 3 were used. The controller uses the learning parameter estimator and one-step-ahead control law which are illustrated in Sec. III and IV.

5.1 The Simulation Studies

The learning self-tuning regulator was applied, through computer simulation, to the control of a pendulum. The nonlinear equation of motion of the pendulum (fig. 4) is a second order nonlinear ordinary differential equation:

$$m l^2 \ddot{\alpha} + c\,\dot{\alpha} + m\,g\,l \sin\alpha = T \qquad (17)$$

where mass: m = 1.0 kgr, link length : l = 0.1 m, and damping coefficient: c = 1.0 Ns/m.

The desired trajectory over 15 sampling intervals (p = 15) for the pendulum is shown in Fig. 5 which swings across two radians so that the nonlinearity of the system is significant. For the first time of executing the desired trajectory (r=0), a self-tuning controller utilizing the RLS estimator (Eq. 5) is used, so that the improvement made by learning over the self-tuning control will become obvious. Since the equation of motion is a 2nd order nonlinear ODE, we decided to use a 2nd order linear difference equation with time varying parameters (eq. 15) as our model for the controller.

While executing the task for the first time, the estimates of P's and $\underline{\theta}$'s at all sampling instants yielded by Eq. 5 are stored in memory. Then, the task is repeated a number of times using the one step ahead control law (Eq. 16) and the estimates of P's and $\underline{\theta}$'s are updated using Eq. 14 everytime a repetition is completed. Forgetting factors used in the parameter estimator were $\lambda = 0.98$ for Eq. 5 and $\gamma = 0.98$ for Eq. 14.

Figure 6 presents the tracking error versus time for repetitions 0, 1 and 5. At the fifth repetition, the tracking errors are almost zero except the first two sampling instants. Everytime the task is repeated, the squares of tracking errors at all sampling instants are added to form a sum. Figure 7 shows the sum of squares of errors decreases as the number of repetitions increases. These results show the effectiveness of the learning adaptive controller to gradually improve the performance of an adaptive controller.

The reason that tracking errors remain large at the first two time steps is because a PD controller was used for these two steps in all the repetitions. This is because the unavailability of data from the -1 and -2 time steps which are needed by both eq. (5) and (14).

5.2 Experimental Studies

In order to study the behavior of the proposed controller in the control of a real nonlinear plant. The controller was used to control the movement of a piezoelectric actuator which is part of a diamond turning lathe. In precision machining such as diamond turning, there are a lot of repetitive error sources such as spindle run-off error and guideway geometric error which would produce inaccuracies on the workpiece if left uncompensated for. Thus, a tool positioning system is called for.

The experimental setup is illustrated in Fig. 8. The tool holder, which is made out of a complete piece of steel using an electric discharge machine, has a construction of two parallel elastic plates which provides a small axial rigidity in comparison to other degrees of freedom. Therefore, the diamond cutting tool moves back and forth as the piezoelectric actuator expands and shrinks. This movement is measured by the inductive displacement sensor which is an integral part of the tool holder.

The piezoelectric actuator is made of PZT-La stack of fifty disks which has a dimension of 14×16×0.5mm. The actuator generates displacement through the mechanism of electrostrictive. This material has less hysteresis loss and better aging property than that of piezoelectric material. However, it has a very nonlinear characteristic between the displacement and the electrical potential applied to it. The parabolic characteristic curve of the actuator is shown in Fig. 9. The existing of the hysteresis nonlinearity is also obvious since the curve follows different paths while the voltage is increased and decreased.

In our experiments, the tool is commanded to follow a square wave with an amplitude of 1.1 μm and a period of 2.3435 second. This desired trajectory is shown in fig. 10. The sampling period is 0.15625. The first execution of the task was carried out by a PD controller. Then, the learning adaptive controller performed the following repetitions as described in section 4. At the beginning, we used a second order difference equation like Eq. 15 in our controller and the tracking error actually got larger and larger over the repetitions. Thus, we increase the order from two to three. This enabled us to obtain a satisfactory controller.

Fig.11 shows the tracking error versus time measured by the inductive position transducer at repetition 0, 1, 6 and 10. It is obvious that significant reduction of tracking errors was achieved by the proposed controller over the repetition of the desired trajectory. Figures 12 shows that the sum of squares of errors decreases as the number of repetitions increases.

VI. Conclusion and future development:

Simulation and experiment results show that the learning self-tuning regulator is a quick learning controller which reduces tracking errors of the two nonlinear systems over repetitions learning tasks. Also, the fact that the parameter updating part could be done off-line between repetitions, makes this control algorithm very feasible. In fact, compared with a PD (proportional plus derivative) controller, only three more additions and three more multiplications have to be done in real-time at each time step if a second order difference equation is used. However, more memory is needed to implement the controller. The amount of memory needed for implementing the controller in an n degree of freedom system is,

= 22 * # of time steps in each repetition * n.

This amount of memory, considering today's low cost memory chips, is feasible for most applications, using even a personal computer.

The simulation and experimental results have shown that the proposed learning self-tuning regulator can be applied to the control of a variety of nonlinear dynamic systems such as a pendulum and a piezo tool positioning systems, possible applications include the control of robots, precision machining, process controls and other manufacturing processes. Also, the notion of a learning parameter estimator could be applied to other parameter estimators than the RLS estimator. These are some paths in the authors' future research.

REFERENCES

1. A. K. Bejczy, T. J. Tarn, Y. L. Chen, "Computer Control of Robot Arms," Proceedings of first IEEE International Conference on Robotics and Automation, St. Louis, Missouri, March 1985.
2. J. J. Craig, Introduction to Robotics Mechanics and Control, Addison-Wesley, Reading, Massachusetts, 1986.
3. D. E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Transactions of Man, Machines, and Systems, Vol. MMS-10, No. 2, pp. 47-53, June 1969.
4. J. Y. S. Luh, M. W. Walker, and R. P. Paul, "Resolved Acceleration Control of Mechanical Manipulators," IEEE Transactions on Automatic Control, Vol. AC-25, No.3, June 1980.
5. J. E. Slotine, "Sliding Controller Design for Non-linear Systems," International Journal of Control, Vol. 40, No. 2, pp. 421-434, 1984.
6. R. H. Middleton, G. C. Goodwin, and R. W. Longman, "A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task," accepted for publication in The International Journal of Robotic Research.
7. M. C. Tsai, G. Anwar, M. Tomizuka, "Discrete Time Repetitive Control for Robots", Proceedings to IEEE Conference, 1988.
8. S. Kawamura, F. Miyazaki and S. Arimoto, "Applications of Learning Methods for Dynamic Controls of Robot Manipulators," Proceeding of the 24th IEEE CDC, Fort Lauderdale, Florida, Dec. 1985.
9. G. C. Goodwin and K. S. Sin, Adaptive Filtering Prediction and Control, Prentice-Hall, New Jersey, 1984.
10. S. Dubowsky, and D. T. DeForges, "The Application of Model Referenced Adaptive Control to Robot Manipulators," ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 101, pp. 193-200, Sep. 1979.
11. J. J. Craig, Adaptive Control of Mechanical Manipulators, Addison-Wesley Publishing Company, New York, 1987.
12. K. J. Åström, "Adaptive Feedback Control," Proceedings of the IEEE, Vol. 75, No. 2, February 1987.
13. K. J. Åström, B. Wittenmark, Computer Controlled Systems Theory and Design, Prentice- Hall, New Jersey, 1984.
14. L. Ljung and T. Sodestrom, Theory and Practice of Recursive Identification, The MIT Press, Cambridge, Massachusetts, 1986.

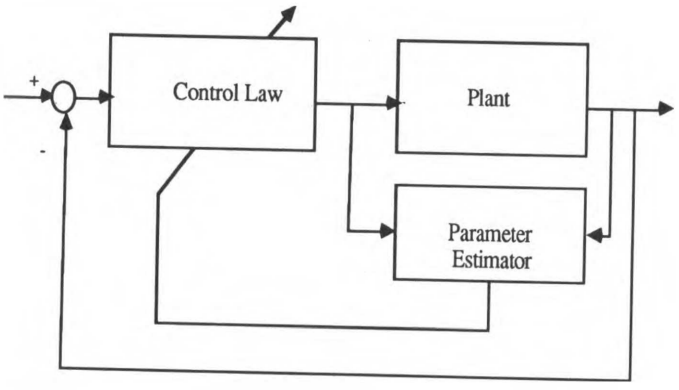

Figure 1 The Self-tuning Regulator

Figure 2 Sampling instants in each repetition of a task

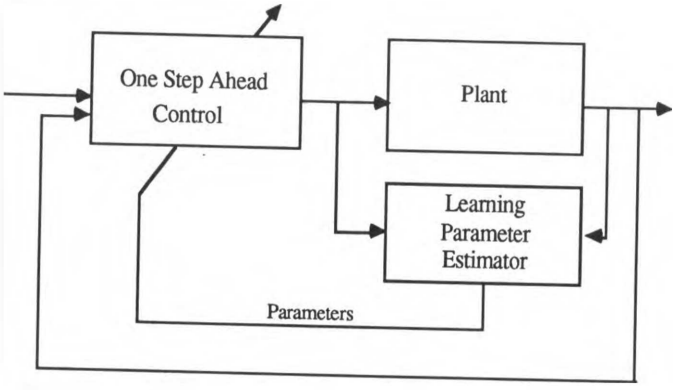

Figure 3 The Learning Self-tuning Regulator

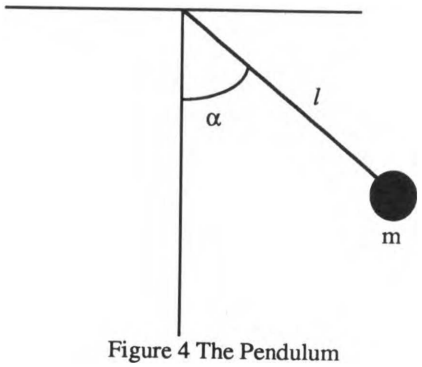

Figure 4 The Pendulum

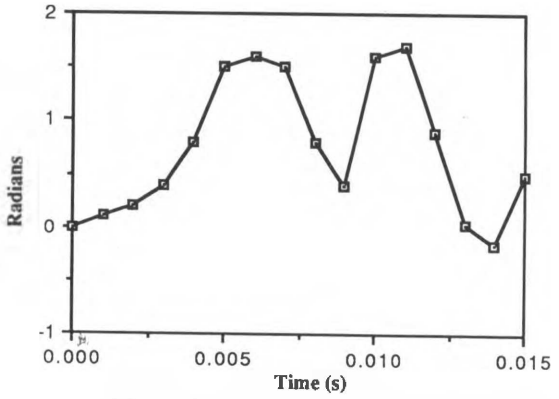

Figure 5 Desired trajectory for the Pendulum

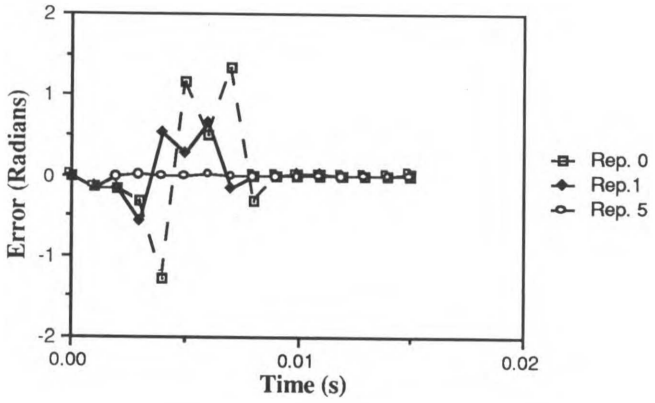

Figure 6 (Tracking errors of repetitions 0, 1 and 5)

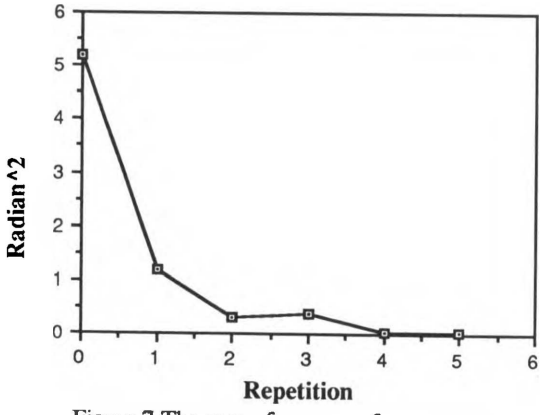

Figure 7 The sum of squares of errors over repetitions

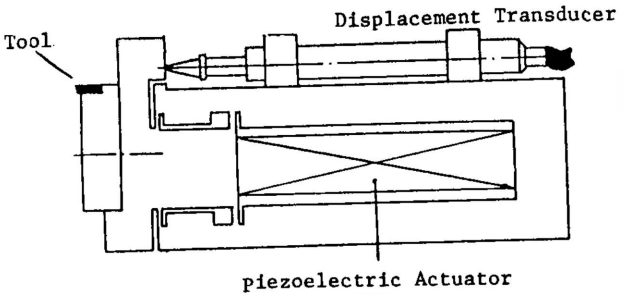

Fig. 8 The experimental setup of the diamond turning tool

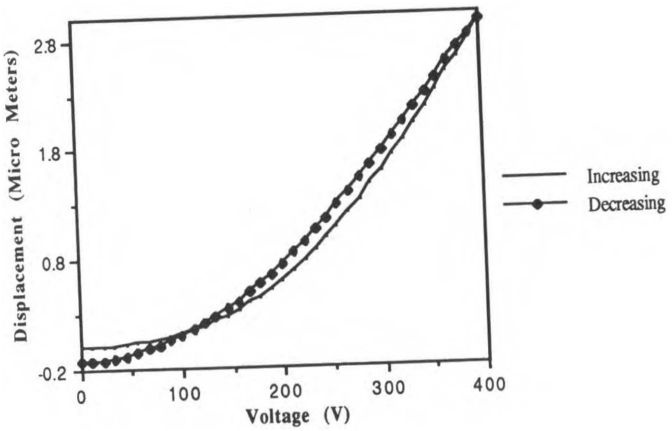

Figure 9 Measured Static Response of the Piezoelectric Tool posiitioning System

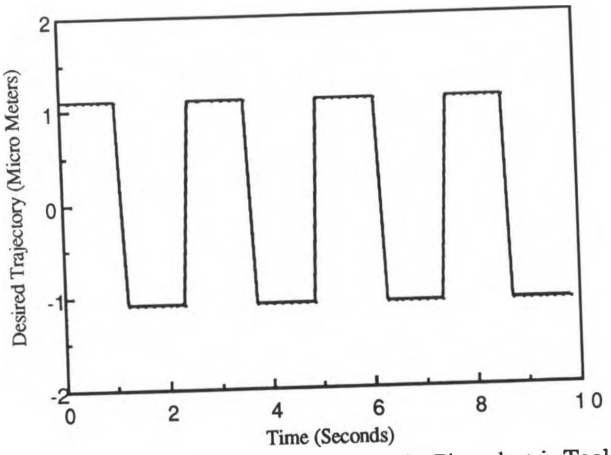

Figure 10 Desired Trajectory for the Piezoelectric Tool

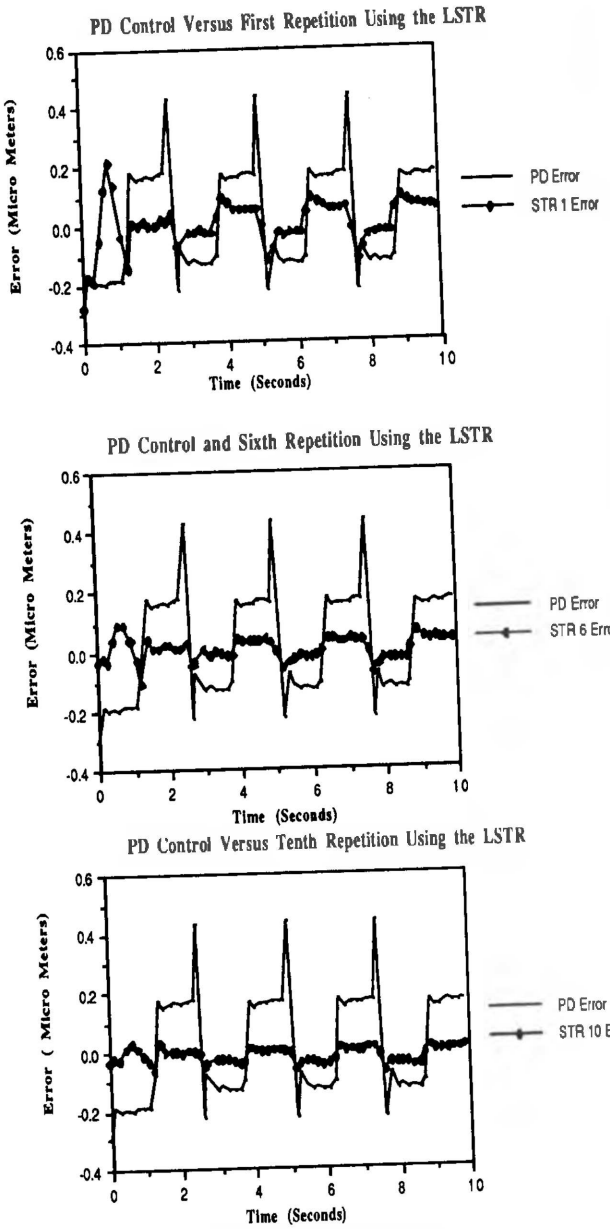

Figure 11 Tracking error of the Piezo Tool at repetitions 0, 1, 6 and 10

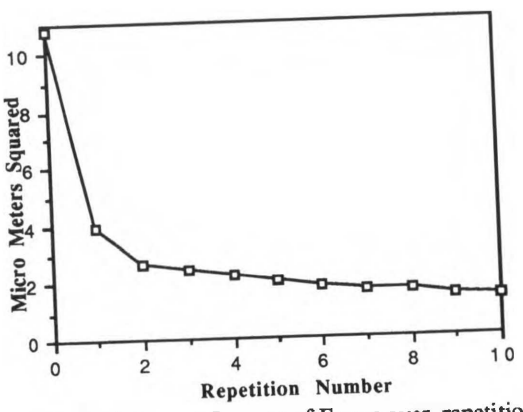

Figure 12 Sum of Squares of Errors over repetitions for Piezo Tool Control