Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems Yokohama, Japan July 26-30, 1993

On The Dynamics Of A Neural Network For Robot Trajectory Tracking

Peter C.Y. Chen† James K. Mills† Kenneth C. Smith†‡*

† Department of Mechanical Engineering, University of Toronto, 5 King's College Road, Toronto, Ontario, Canada M5S 1A4

‡ Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 1A4

* Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 1A4

Abstract

In this paper, the dynamic behavior of a three-layer feedforward neural network as a uncertainty compensator for robotic control is investigated. The investigation is conducted in the context of the robot trajectory tracking problem, where the neural network (with the error-backpropagation algorithm) is used as a uncertainty compensator in conjunction with the feedback linearization control (i.e. computed torque) and a PD control. Through computer simulation, it is verified that the dynamics of the neural network has a specific pattern when the learning rate is sufficiently small, and that such a specific pattern of weight variation in the neural network represents a sufficient condition for closed-loop system performance improvement.

1 Introduction

Traditionally, neural networks have been used to learn the structure of the input-output mappings of static and dynamical systems. It has been demonstrated that upon learning the structure of a given mapping, a neural network is able to generalize, and thus responding "optimally" to unfamiliar situations [7, 10, 11]. Relatively recently, there has been a trend within the robotics control literature to apply neural networks for the control of robotic systems. In many applications reported in the literature (e.g. [4, 5, 6, 8, 9, 12, 13, 15]) the process of neural network learning is conducted on-line (i.e. the dynamics of the neural network is embedded in closed-loop with the dynamics of the robotic system), yet there appears to be a lack of studies focusing on the dynamic behavior of the neural network (i.e. weight variation during learning and/or control) when the network is used in such context.

It is the premise of this work that investigation into the dynamic behavior of the neural network (when used in closed-loop with other system components) is required. This premise is motivated by the observation that, in closed-loop systems that include neural network dynamics, the dynamics of the control object (i.e. the system that a neural network is to learn to control) changes with the dynamics of the neural network (i.e. weight adjustment). Proper theoretical insight regarding the dynamical properties of the neural network and that of the overall closed-loop system must be given to support any scheme that utilizes neural networks for control. Furthermore, it must be demonstrated (and analytically verifiable) that the use of neural networks dynamically in closed-loop with other components of the system leads to not only a stable system, but also an improvement in closed-loop system performance.

In this paper, the dynamic behavior of a three-layer feedforward neural network as a uncertainty compensator for robotic control is investigated. The investigation is conducted in the context of the robot trajectory tracking problem, where the neural network (with the error-backpropagation algorithm) is

0-7803-0823-9/93/$3.00 (C) 1993 IEEE

used as a uncertainty compensator in conjunction with the feedback linearization control (i.e. computed torque) and a PD control. It is verified through computer simulation that the dynamics of the neural network has a specific pattern when the learning rate is sufficiently small, and that such a specific pattern of weight variation in the neural network constitutes a sufficient condition for closed-loop system performance improvement.

This paper is organized as follows. Section 2 formulates the problem. Sections 3 and 4 briefly review the approach and analysis originally presented in [2]. Section 5 describes the computer simulation conducted for the purpose of verifying the analytical results discussed in Section 4, and presents the results of the computer simulation that evidently confirm those analytical results. Section 6 concludes the paper.

2 Problem Formulation

In general, the dynamics of a unconstrained serial manipulator with n joints can be described by a set of nonlinear differential equations, compactly expressed in the form

$$M(q)\ddot{q} + h(q,\dot{q}) = \tau \tag{1}$$

where $q \in R^n, \dot{q} \in R^n$, and $\ddot{q} \in R^n$ are respectively the joint position, joint velocity, and joint acceleration vectors, $M(\cdot) \in R^{n\times n}$ is the inertia matrix, $h(\cdot) \in R^n$ is a vector containing the Coriolis and gravitation terms, and $\tau \in R^n$ is the input torque vector. With the nonlinear feedback control law [14]

$$\tau = \hat{M}(q)u + \hat{h}(q,\dot{q}) \tag{2}$$

where $\hat{M}(\cdot)$ and $\hat{h}(\cdot)$ are respectively the estimates of the actual $M(\cdot)$ and $h(\cdot)$, and

$$u = \ddot{q}^d + K_v\dot{e}_1 + K_p e_1 \tag{3}$$

where $e_1 = q^d - q$, $\dot{e}_1 = \dot{q}^d - \dot{q}$, and $q^d \in R^n, \dot{q}^d \in R^n$, and $\ddot{q}^d \in R^n$ are respectively the desired joint position, velocity, and acceleration vectors, $K_v \in R^{n\times n}$ and $K_p \in R^{n\times n}$ are diagonal gain matrices, the resulting closed-loop error dynamics becomes [14]

$$\dot{e} = Ae + B\eta(q,\dot{q},\ddot{q}) \tag{4}$$

where

$$e = \begin{bmatrix} e_1 \\ \dot{e}_1 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{5}$$

and $\eta(q,\dot{q},\ddot{q}) = (\hat{M}^{-1}M - I)\ddot{q} + \hat{M}^{-1}(h - \hat{h})$ is a nonlinear function (of $q, \dot{q}$, and $\ddot{q}$) containing unknown parameters in M and h, and is defined as the *uncertainty* associated with the robot trajectory tracking problem. Note that for brevity, the arguments for M, h, $\hat{M}$, and $\hat{h}$ have been omitted.

A signal v is introduced to compensate for the uncertainty η. Now let

$$u = \ddot{q}^d + K_v\dot{e}_1 + K_p e_1 + v. \tag{6}$$

Then the error dynamics of the resulting closed-loop system can be expressed as

$$\dot{e} = Ae + B\Delta v \tag{7}$$

where $\Delta v = \eta(q,\dot{q},\ddot{q}) - v$ is referred to as the *control error*. Thus the problem is to generate the appropriate signal v to eliminate the control error so that the desired error dynamics

$$\dot{e} = Ae \tag{8}$$

can be achieved. Figure 1 illustrates this idea of uncertainty compensation.

3 Approach[2]

From $\Delta v = \eta(q,\dot{q},\ddot{q}) - v$, it is observed that $\Delta v = 0$ implies that $v = \eta(q,\dot{q},\ddot{q})$. An ideal compensator is a function whose output v exactly equals that of the function $\eta(\cdot)$ so that the control error $\Delta v = \eta(\cdot) - v = 0$. Based on such a premise, the problem of designing the compensator can then be considered as a function approximation problem.

It is proposed that a three-layer feedforward neural network (Figure 2) be employed as the compensator. To approximate the function $\eta(q,\dot{q},\ddot{q})$, the neural network takes $q,\dot{q},\ddot{q}$ as its input, and produces an output v. Thus for an n-joint manipulator, the numbers of units in the input layer and the output layer are respectively $3n$ and n. The connection weights between the input layer and the hidden layer are represented by the matrix Θ, while that between the hidden layer and the output layer are represented by the matrix Γ. Suppose that the number of units in the hidden layer is m, then $\Theta \in R^{3n\times m}$ and $\Gamma \in R^{m\times n}$. A general weight matrix W can be defined as: $W = (\Theta,\Gamma)^T$. The output of the neural network v is bounded if a sigmoid function is used as the activation function for each of the output units.

Note that the control error

$$\Delta v = \eta - v \tag{9}$$

represents the difference between the output of the uncertainty function $\eta(\cdot)$ and the output of the neural network v. The objective of neural network learning is then to effectively adjust the weights of the neural network to minimize the controller error Δv. Let a cost function to be minimized be

$$J = \frac{1}{2}\Delta v^T \Delta v. \tag{10}$$

According to the error-backpropagation algorithm [7, 10, 11], the weights of the neural network are adjusted in proportion to the gradient $\frac{\partial \Delta v}{\partial W}$, i.e.

$$\dot{W} = -\lambda \frac{\partial J}{\partial W} = -\lambda \Delta v^T \frac{\partial \Delta v}{\partial W} \tag{11}$$

where λ is the learning rate. Since $\Delta v = v^d - v$ and $\frac{\partial v^d}{\partial W} = 0$, so (11) becomes

$$\dot{W} = \lambda \Delta v^T \frac{\partial v}{\partial W} \tag{12}$$

where the controller error Δv can be determined in real-time as (from (7))

$$\Delta v = \eta - v = \ddot{e}_1 + K_v \dot{e}_1 + K_p e_1. \tag{13}$$

Note that (13) contains the joint acceleration vector $\ddot{q}$, which, though not directly available from the robotic system output, can be estimated as

$$\ddot{q}(t) \approx \frac{\dot{q}(t + \Delta t) - \dot{q}(t)}{\Delta t} \tag{14}$$

where Δt is the sampling interval of the velocity information. The use of the network output error (i.e. the control error in this case) for neural network learning is similar to that in [12], although in a different context.

4 Neural Network Dynamics and Closed-loop System Performance[2]

The closed-loop dynamics of the system with the neural network learning on-line is described by

$$\begin{cases} \dot{e} = Ae + B\Delta v(q, \dot{q}, \ddot{q}, W) \\ \dot{W} = -\lambda \Delta v^T(q, \dot{q}, \ddot{q}, W) \frac{\partial \Delta v(q, \dot{q}, \ddot{q}, W)}{\partial W}. \end{cases} \tag{15}$$

It is generally accepted that, for the error backpropagation algorithm to function properly, the learning rate λ must be small [10]. Most empirical studies suggest that λ be limited under 0.1. With a small learning rate, the closed-loop system (15) can be considered as a two time-scale system with the robot having the fast dynamics while the neural network having the slow dynamics. As the learning rate λ approaches zero, the change of the state of the neural network (i.e. the change in the values of the connection weights) becomes infinitesimally small.

One direct consequence for having a small learning rate is that, because neural network learning is thus made slow, the learning process must be conducted in a repetitive manner (i.e. through a series of trials). In other words, the robot executes a task repetitively while the neural network learns on-line to generate the signal v to counteract the uncertainty η.

It has been proved in [2] that the trajectory tracking error e in (15) decreases as the number of trials increases. This proof is based in the following assumption about the dynamic behavior of the neural network.

Assumption: For the system (15) with $\lambda << 1$, the following approximation is valid:

$$\int_0^\xi \dot{W}(p+1, \sigma) d\sigma \approx \int_0^\xi \dot{W}(p, \sigma) d\sigma, \quad \forall \xi \in [0, T]. \tag{16}$$

where $W(p, \sigma)$ represents the value of the weight matrix W at the σ^{th} second of the p^{th} trial, and ξ is the time variable associated with one trial (i.e. if each trial lasts T seconds, then $0 \leq \xi \leq T$).

This assumption means that, for $\lambda << 1$, the difference between the weight change of any two *successive* trials is considered negligible. For example, as illustrated in Figure 3, let $\bar{W}$ be an element of W. Then

$$\Delta \bar{w}(p+1, \xi) = \int_0^\xi \dot{\bar{W}}(p+1, \sigma) d\sigma, \tag{17}$$

and

$$\Delta \bar{w}(p, \xi) = \int_0^\xi \dot{\bar{W}}(p, \sigma) d\sigma. \tag{18}$$

Although the difference between the initial and final values of $\bar{W}$ (i.e. $\bar{W}(p+1, 0) - \bar{W}(p, 0)$) of any trial could be significant, the difference between the *change* in $\bar{W}$ of any two successive trials can be considered negligible, i.e.

$$\Delta \bar{w}(p+1, \xi) \approx \Delta \bar{w}(p, \xi), \quad \forall \xi \in [0, T]. \tag{19}$$

Under this assumption, it has been proved that the trajectory tracking error decreases as the number of trials increases. The proof utililizes a cost function defined as

$$L(p) = \frac{1}{2} \int_0^T \Delta v^T(p, \xi) \Delta v(p, \xi) d\xi \tag{20}$$

and shows that, under the above assumption about the dynamic behavior of the neural network, $L(p)$ decreases as the number of trials p increases. This implies that the controller error Δv in (15) decreases (due to the error-backpropagation algorithm) as the number of trials p increases. Since a reduction in the control error directly translates to a reduction in the trajectory tracking error, it can be concluded that the trajectory tracking error decreases as the number of trials increases.

The physical relevance of the above assumption can be interpretated as follows. In the closed-loop system (15), with a small λ (i.e. $0 < \lambda << 1$), the change in the weights *per trial* can be expected to be small. Such a small change in the weights will not have significant effect on the state of the

robot. Therefore, between any two *successive* trials, the change in the state of the robot, and consequently the difference between the two amounts of weight change, can be considered negligible.

5 Simulations

Computer simulations have been conducted for the trajectory following problem to verify the assumption and to substantiate the theoretical results discussed in Section 4. A simple two-link planar robot (Figure 4) [3] was used in the simulations. It is assumed that the mass of each link can be represented as a point mass at the end of the link.

A control system (equivalent to that shown in Figure 1) was setup in the *EASY5*[1] environment (Figure 5). The component $RA01$ contains the dynamic equations of the planar robot, expressed in *EASY5*'s macro language [1]. This component was used to represent the "actual" robot. The same dynamic equations were also used in the nonlinear interface (i.e. feedback linearization), which was implemented in the component $RB01$ using *EASY5*'s macro language as well. Thus if the parameters (e.g. mass of a link) of the "actual" robot are known precisely (resulting in $\eta = 0$), then the "actual" robot ($RA01$), together with the nonlinear interface ($RB01$), become a linear second order system.

To see th effect of robot parameter uncertainty on the system behavior, discrepanicies between the masses of the links of the "actual" robot ($RA01$) and that used in the nonlinear interface ($RB01$) were introduced. Specifically, in the "actual" robot, the masses of the first and second links were set to be $5kg$ and $4kg$ respectively, but in the nonlinear interface, these masses were set to be $4.5kg$ for both links. Thus an error of 10% was introduced in the mass of the first link, and an error of 12.5% was introduced in the mass of the second link. As a result, $\eta \neq 0$.

A generic three-layer feedforward neural network was also developed using *EASY5*'s macro language. This neural network macro allows various parameters, such as the number of input units, the number of hidden units, the number output units, the learning rate, and the initial weights, etc., to be set for a specific simulation. The neural network used in this particular application (component $NN01$) has six input units, twenty-five hidden units, and two output units. The input to the neural network were $q \in R^{2\times 1}, \dot{q} \in R^{2\times 1}$, and $\ddot{q} \in R^{2\times 1}$, and was constructed by the FORTRAN component $FO66$ as a 6×1 vector. The signal Δv was constructed by the FORTRAN component $FO55$ according to (13). The output of the neural network was $v \in R^{2\times 1}$. The learning rate λ was set at 0.001. The activation function for the hidden units and the output units was set to be a hyperbolic tangent function with the limits of -1 and 1.

Another macro component $TG01$ was developed to generate the desired trajetcory, which was specified as

$$q^d = 0.1t - 0.1t^2 + 0.05e^{-2t}. \tag{21}$$

The length of the first link was set to be $0.7m$, and that of the second link $0.5m$. The gains in for the PD controller were

$$K_v = \begin{bmatrix} 15 & 0 \\ 0 & 40 \end{bmatrix}, \quad K_p = \begin{bmatrix} 80 & 0 \\ 0 & 150 \end{bmatrix}. \tag{22}$$

The initial state of the "actual" robot was set to be

$$q_1 = q_2 = 0.05 \ rad, \tag{23}$$

$$\dot{q}_1 = \dot{q}_2 = 0 \ rad/sec. \tag{24}$$

A series of simulation runs (i.e. trials) were carried out using the *EASY5* model as shown in Figure 5. The initial weights of the neural network for the first trial were set to arbitrary values in the order of 10^{-4}. Using the user interface of EASY5, the values of the weights at the end of each trial were saved in a text file, and were used as the initial state of the neural network for the next trial.

Figures 6 shows the dynamics of one connection weight ($W(1,10)$) during the 1^{st} trial (plot on the left) and the 2^{nd} (plot ont the right) trial. Figure 7 shows the dynamics of the same connection weight during the 40^{th} trial and the 41^{st} trial. It can be seen that, for one given trial, the final value and the initial value of the weight are different, but the *difference* of the weight change between two *successive* trials (i.e. 1^{st} and 2^{nd}, 40^{th} and 41^{st}) is negligible. Comparing these figures to Figure 3 (which illustrates the assumption that the difference of the weight change between any two successive trials is negligible), it is evident that the neural network does indeed possess the dynamic behavior as assumed.

Figure 8 shows the trajectory tracking error of the two joints in the 1^{st} trial (plot on the left) and the 41^{st} trial (plot on the right). It can be seen that the trajectory tracking error of both joints in the 41^{st} trial is noticieably smaller than that in the 1^{st} trial. This clearly confirms the theoretical conclusion that

[1]The *Boeing EASY5 Engineering Analysis System (EASY5)* is an interactive computer program for the modeling, analysis, and design of large complex dynamical systems defined by algebraic, differential, and/or difference equations. The software version used for the simulations in this work was *EASY5z, Version 2.0.920406.*

the trajectory tracking error decreases as the number of trials increases.

6 Conclusion

The dynamics behavior of a three-layer feedforward neural network for robotic uncertainty compensation has been investigated via computer simulation. It has been verified that when used in closed-loop with other control system components for robot trajectory tracking, the dynamic behavior of the neural network (effected by the error-backpropagation learning algorithm) follows a specific pattern when the learning rate is sufficiently small. It has also been verified that this specific dynamic pattern of the variation in the neural network weights represents a sufficient condition for guaranteeing improvement in the closed-loop system performance. These simulation results are significant because they confirm the theoretical analysis of [2], and thus validating a novel approach and methodology for the analysis of the closed-loop stability and performance of control systems that incoporate dynamic neural networks.

Acknowledgement

This work is financially supported by IRIS, NSERC, and OGS. The *EASY5x* software package and product related technical advices are provided by The Boeing Company.

References

[1] Boeing Computer Service, *EASY5 User Guide.* Seattle: The Boeing Company, 1988.

[2] Chen, Peter C.Y., James K. Mills, and Kenneth C. Smith, *Uncertainty Compensation in the Control of Robotic Systems: A Neural Network Approach.* Proceedings of the 1992 Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, September 13-16, 1992.

[3] Craig, J., *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley, 1986.

[4] Fukuda, T. and T. Shibata, Neuromorphic Control for Robotic Manipulators: Position, Force and Impact Control. *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, (Philadelphia, Pennsylvania), 310-315, 1990.

[5] Gu, Y. and J.W.M. Chan, On Design of Nonlinear Robotic Control System with Neural Networks. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, (Cambridge, Massachusetts), 200-205, 1989.

[6] Helferty, J.J. and S. Biswas, Neuromorphic Control as a Self-Tuning Regulator. *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, (Philadelphia, Pennsylvania), 506-511, 1990.

[7] Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation.* Addison-Wesley: Redwood City, California, 1991.

[8] Jamshidi, M., B. Horne, and N. Vadiee, A Neural Network-Based Controller for a Two-Link Robot. *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), 3256-3257, 1990.

[9] Karakasoglu, A. and M.K. Sundareshan, Decentralized Variable Structure Control of Robotic Manipulators: Neural Computational Algorithms. *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), 3258-3259, 1990.

[10] McClelland, J.L. and D.E. Rumelhart (eds), *Parallel Distributed Processing*, Vol. 1, Cambridge, Massachusetts: MIT Press, 1986.

[11] Muller, B. and J. Reinhardt, *Neural Networks: An Introduction.* Berlin: Springer-Verlag, 1990.

[12] Okuma, S. and A. Ishiguro, A Neural Network Compensator for Uncertainties of Robotic Manipulators. *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), 3303-3307, 1990.

[13] Pourboghrat, F., Neuromorphic Controllers. *Proceedings of the 28th IEEE Conference on Decision and Control*, (Tempa, Florida), 1748-1749, 1989.

[14] Spong, M.W. and M. Vidyasagar, *Robot Dynamics and Control.* New York: John Wiley & Sons, 1989.

[15] Yamamura, A.A., A. Sideris, C. Ji, and D. Psaltis, Neural Network Control of a Two-Link Manipulator. *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), 3303-3307, 1990.

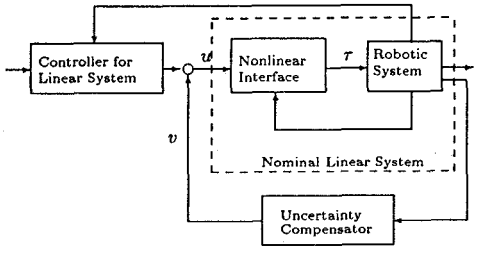

Figure 1 Uncertainty Compensation

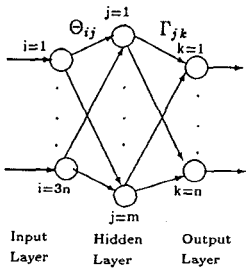

Figure 2 Neural Network

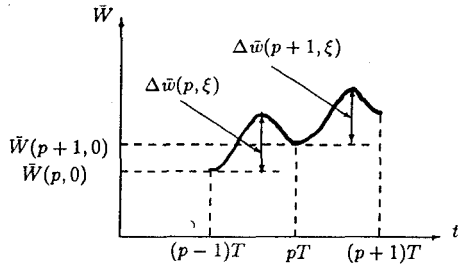

Figure 3 Weight and Weight Change

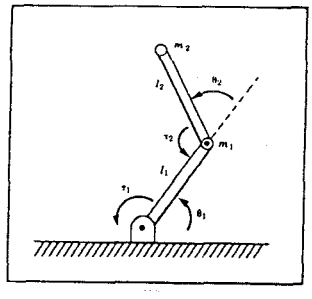

Figure 4 A Two-Link Planar Robot

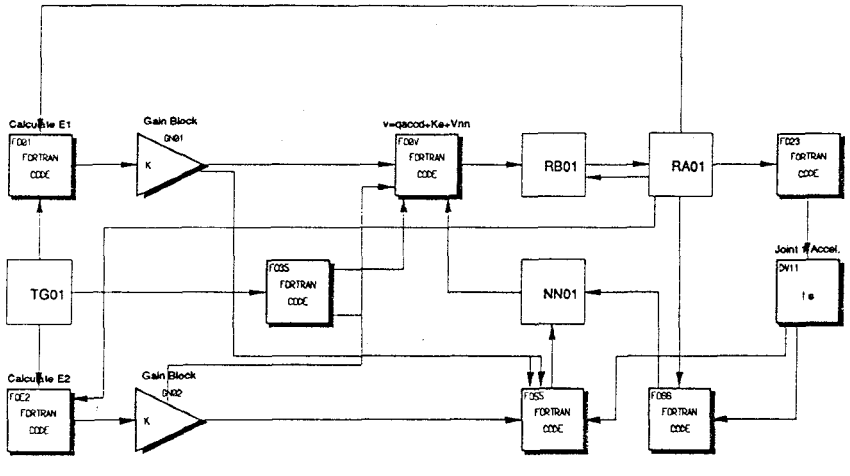

Figure 5 An EASY5 Schematic Model

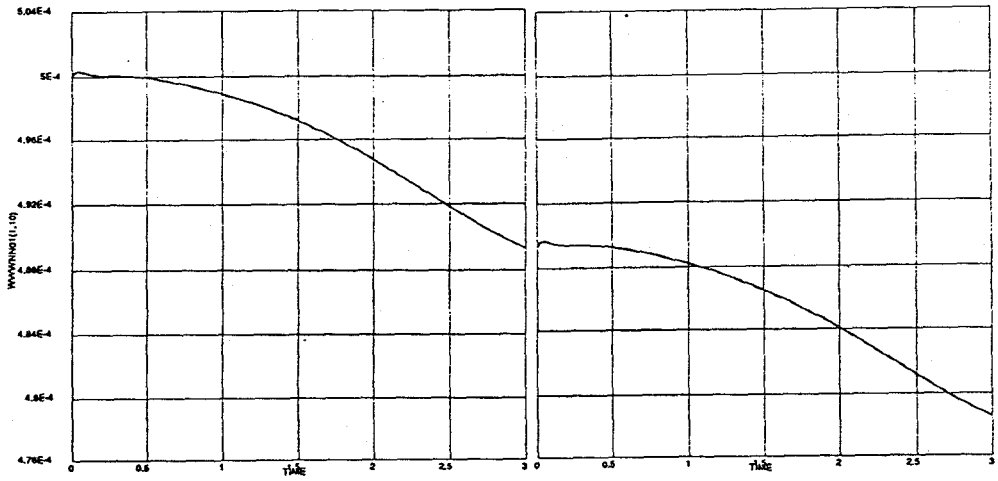

Figure 6 Weight Dynamics in 1st and 2nd Trial

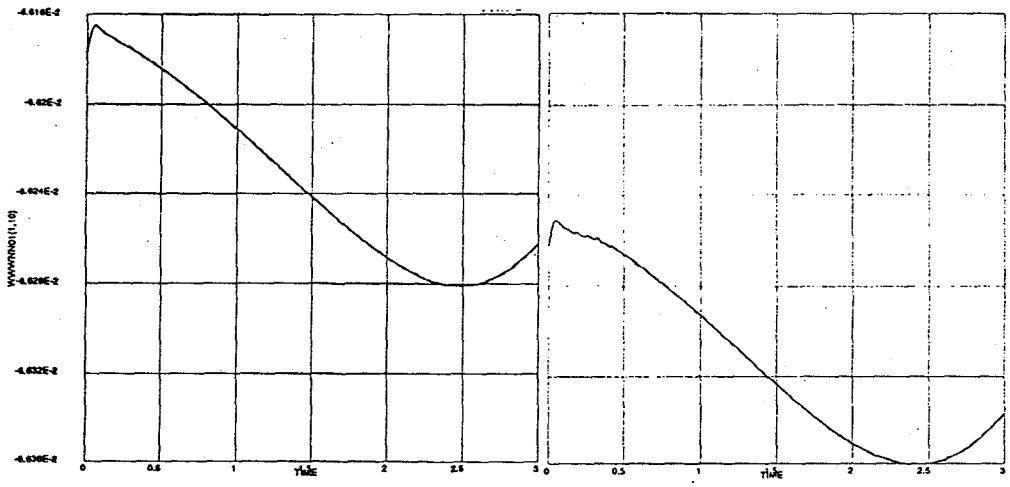

Figure 7 Weight Dynamics in 40th and 41st Trial

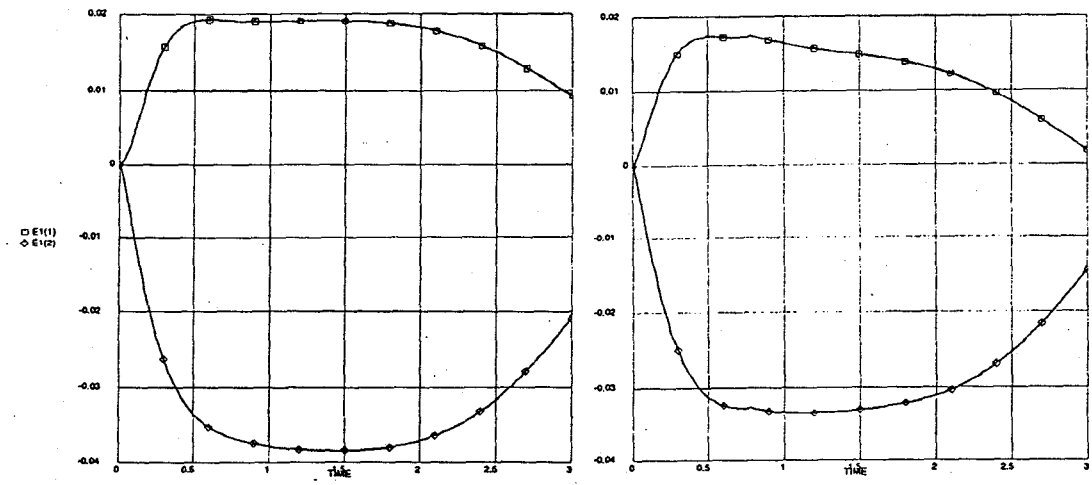

Figure 8 Tracking Error in 1st and 41st Trial (in Rad.)