MULTI-LAYER NEURAL NETWORKS FOR ROBOT CONTROL

Farzad Pourboghrat

Department of Electrical Engineering
Southern Illinois University
Carbondale, IL 62901-6603

Abstract

Two neural learning controller designs for manipulators are considered. The first design is based on a neural inverse-dynamics system. The second is the combination of the first one with a neural adaptive state feedback system. Both types of controllers enable the manipulator to perform any given task very well after a period of training, and to do other untrained tasks satisfactorily. The second design also enables the manipulator to compensate for unpredictable perturbations.

1. Introduction

The design of advanced control systems for robot manipulators has been a very active area of research in recent years. Inadequacy of current control strategies suggests that there is a need for a newer and faster control architecture which will account for both learning and control of robotic manipulators.

In classical systems theory, input-output descriptions are based on some assumed or predetermined mathematical structures, normally a set of linear differential equations. Replacement of these predetermined structures by learned associative memory mappings of stimulus-response leads to more general, normally non-linear, representations of the connections between inputs and outputs. This procedure can be implemented by neural networks [1]. The best example of a system with such an architecture is the human brain, which performs many complex functions superbly.

In the problem of motor control, obtaining an input function u(t) to generate a desired motion y(t) is directly related to finding the inverse-dynamics of the controlled system. Let the operator $\mathcal{G}$ denote the dynamics relation of the system, where $\mathcal{G}(u)=y$. Then the inverse-dynamics of the system is the operator $\mathcal{E}=\mathcal{G}^{-1}$ such that $\mathcal{E}(y)=u$. Knowing the inverse-dynamics relation $\mathcal{E}=\mathcal{G}^{-1}$, for a given desired motion trajectory y_d, the required input u_d can be found from $u_d=\mathcal{E}(y_d)$. This is because the motion corresponding to u_d is equal to $y=\mathcal{G}(u_d)=\mathcal{G}(\mathcal{E}(y_d))=\mathcal{G}(\mathcal{G}^{-1}(y_d))=y_d$.

It has been shown that multi-layer neural networks with sigmoidal functions are able to map any measurable function to another with an arbitrary degree of accuracy, provided that there are enough units in their hidden layers. Therefore, such networks can be used for approximating the model of the inverse of the dynamics of a system [2-10]. In this paper the development of neuromorphic learning controllers is considered. First, a recurrent neural network learning controller $\mathcal{C}$ is designed. The design has a neural inverse-dynamics block $\mathcal{E}$ and a PD-type feedback block $\mathcal{H}$. Next, the learning controller $\mathcal{C}$ is modified, where its PD-type feedback block is replaced by a neural adaptive state feedback block $\mathcal{H}$, which is to optimally compensate for unpredictable perturbations. The architectures of these learning controllers are similar to those in [10], which are inspired by the model of the cerebellum given by Kawato [5-6].

2. Robot Dynamics

The dynamics of a robot manipulator can be represented by an operator $\mathcal{G}$ which corresponds to a set of n coupled nonlinear differential equations, given by

$$M(q)\,q'' + N(q,q') + Q(q) = u \qquad (1a)$$

or $$\mathcal{G}(u) = q \qquad (1b)$$

where q, q', and q" are n-dimensional vectors of the positions, velocities, and accelerations of the joints, respectively, where "prime" denotes the time-derivative. M(q) is the nxn inertia matrix of the arm, which is symmetric and positive definite. N(q,q') is the n-dimensional vector of coriolis, centrifugal, and frictional forces. Q(q) is the n-dimensional vector representing the torques due to gravitational forces, and u is the n-dimensional vector of the generalized input torques applied to the robot.

3. Learning Controller Design

There are a variety of algorithms which can be used for multi-layer neural networks to learn the mapping between two patterns [1]. However, the state of the art learning algorithms are most effective when the input-output patterns are fixed. This condition, in general, is not satisfied when the objective error function is not identical to the error function at the neural network's output layer. To satisfy this condition we observe the following.

Lemma 1

Consider a stable system given by the operator $\mathcal{G}$, as in Figure 1, where its output q is desired to follow a reference function q_r. Let the high gain feedback block given by the linear operator $\mathcal{H}$ be such that the closed-loop $(I+\mathcal{G}\mathcal{H})^{-1}\mathcal{G}$ is stable and that $\|\mathcal{G}\mathcal{H}\| >> 1$. Then for bounded input v the output error $e=q_r-q$ is bounded and is given by $e=[(I+\mathcal{G}\mathcal{H})^{-1}\mathcal{G}](\delta v)\approx\mathcal{H}^{-1}(\delta v)$, where $\delta v=r-v$. Moreover, the feedback signal $\delta u=\mathcal{H}(e)\approx\delta v$.

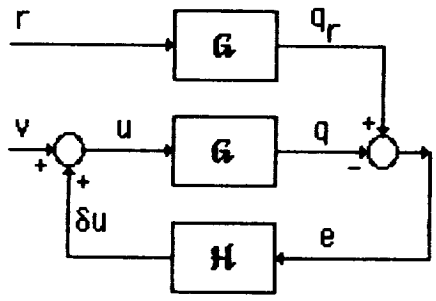

Figure 1

Proof

From Figure 1, by some block manipulation, it is easy to see that $e=[(I+\mathcal{G}\mathcal{H})^{-1}\mathcal{G}](\delta v)$, where $\delta v=r-v$. Now let v be bounded. Then, since r exists, δv is also bounded. But since the closed-loop system $(I+\mathcal{G}\mathcal{H})^{-1}\mathcal{G}$ is stable, the error signal $e=q_r-q$ is also bounded. Now since $\|\mathcal{G}\mathcal{H}\| >> 1$, we get $e\approx\mathcal{H}^{-1}(\delta v)$. On the other hand, since $\mathcal{H}$ is linear, we have $\delta u=[(I+\mathcal{G}\mathcal{H})^{-1}\mathcal{G}\mathcal{H}](\delta v)$. But again, since $\|\mathcal{G}\mathcal{H}\| >> 1$, it is easy to see that $\delta u=\mathcal{H}(e)\approx\delta v$. □

4. Neural Inverse-Dynamics Model for Learning Control

The learning controller $\mathfrak{C}$, shown in Figure 2, has only one neural network block $\mathfrak{E}$ to approximate the inverse-dynamics model. There is also a feedback block $\mathfrak{H}$, of the PD-type, which is used for both the neural learning and the error compensation, and is given by

$$\mathfrak{H}(e) = \delta u = K_p e + K_d e' . \tag{2}$$

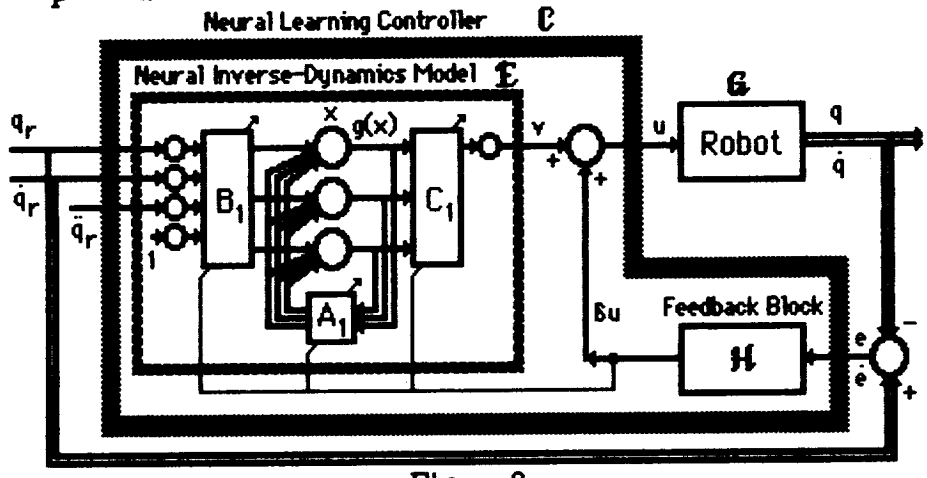

Figure 2

Network's Architecture

The neural block $\mathfrak{E}$ used here is essentially a recurrent multi-layer neural network. The input-output relation of the neural network $\mathfrak{E}$ is given by

$$\begin{aligned} x' &= A_1\, g(x) + B_1\, \theta \\ v &= C_1\, g(x) \end{aligned} \tag{3}$$

where $\theta = [q_r^T, q'_r{}^T, q''_r{}^T, 1]^T \in \Re^{3n+1}$, $x \in \Re^N$, and $v \in \Re^n$ are respectively the vectors of the network's input, states, and outputs. A_1, B_1, and C_1 are respectively the matrices of the network's state recurrence, input, and output connection weights, and g is the sigmoidal function given by g(x)=tanh(x). The unity input in vector θ is added to allow for the automatic adjustment of the bias term.

Network's Learning Rule

The learning algorithm used for the network is a modification of the delta rule [1], and is given by [11]

$$\begin{aligned} a'_{1,ij} &= \alpha_1\, \delta u^T\, \nabla g(x)\, C_1\, \eta_{1,ij} \\ b'_{1,ik} &= \beta_1\, \delta u^T\, \nabla g(x)\, C_1\, \zeta_{1,ik} \\ c'_{1,pj} &= \gamma_1\, \delta u_p\, g(x_j) \\ \eta'_{1,ij} &= A_1 \nabla g(x)\, \eta_{1,ij} + I_i\, g(x_j) \\ \zeta'_{1,ik} &= A_1 \nabla g(x)\, \zeta_{1,ik} + I_i\, \theta_k \end{aligned} \tag{4}$$

where I_i is the ith column of the identity matrix, δu is the feedback torque which is also the network's output error, $\nabla g(x)=\partial g(x)/\partial x$ is the Jacobian matrix, and α_1, β_1, and γ_1 are the learning rate constants which are small positive numbers. The initial values of matrices A_1, B_1, and C_1 are selected randomly between -0.2 and 0.2, and $\eta_{1,ij}(0)=\zeta_{1,ik}(0)=0$.

The objective of the learning controller $\mathbf{C}$ is to force the system's output error to zero through repeated trials of the desired task. During trials, when the reference input is repeatedly applied to the system, the system's output error is used to adjust the controller parameters, which are the connection weights of the neural network block $\mathbf{E}$. Therefore, the feedforward block $\mathbf{E}$ is modified in such a way to force the feedback torque to vanish, which indirectly decreases the robot's output error. When the error becomes small, learning has been accomplished and the neural network block $\mathbf{E}$ is said to have acquired the model of the inverse-dynamics of the robot. But for this, the corresponding learning algorithm must be convergent, or, the dynamics of the learning system must be asymptotically stable.

Result 1

Consider the robotic manipulator given by the operator $\mathbf{G}$, as in equation (1). Let the neural learning controller $\mathbf{C}$ given by equations (2) and (3) be applied to the system, as shown in Figure 2. Let the feedback block $\mathbf{H}$ be such that the closed-loop system $(I+\mathbf{GH})^{-1}\mathbf{G}$ is stable and that $\|\mathbf{GH}\|>>1$. Then the neural learning controller $\mathbf{C}$, together with the learning rule (4) is asymptotically stable. That is, the proposed learning controller forces the manipulator's trajectory q, q', to follow the desired trajectory q_r, q'_r, after a sufficiently long period of time.

Proof

Let $\eta_{1,ij}=\partial x/\partial a_{1,ij}$ and $\zeta_{1,ik}=\partial x/\partial b_{1,ik}$, then from equation (3), we get [11]

$$x = \int_0^t [A_1 g(x) + B_1 \theta]\, d\tau$$

$$\eta_{1,ij} = \int_0^t \partial/\partial a_{1,ij} [A_1 g(x) + B_1 \theta]\, d\tau = \int_0^t [A_1 \nabla g(x) \eta_{1,ij} + I_i\, g(x_j)]\, d\tau$$

$$\zeta_{1,ik} = \int_0^t \partial/\partial b_{1,ik} [A_1 g(x) + B_1 \theta]\, d\tau = \int_0^t [A_1 \nabla g(x) \zeta_{1,ik} + I_i\, \theta_k]\, d\tau.$$

Differentiating the above two relationships, we get

$$\eta'_{1,ij} = A_1 \nabla g(x)\, \eta_{1,ij} + I_i\, g(x_j) \qquad (5)$$

$$\zeta'_{1,ik} = A_1 \nabla g(x)\, \zeta_{1,ik} + I_i\, \theta_k.$$

Now, without loss of generality, we asume that there exists an input function r(t) to the manipulator such that $q_r=\mathbf{G}(r)$. Let a performance function for the learning process of the neural inverse-dynamics network be defined by

$$J_1(t) = 0.5\, [r(t)-v(t)]^T [r(t)-v(t)] = 0.5\, \delta r^2 \,. \qquad (6)$$

Since $J_1(t)$ is positive definite and monotonically increasing, for asymptotic stability, $J'_1(t)$

must be negative definite. But, the time derivative of $J_1(t)$ is given by

$$\begin{aligned} J'_1(t) &= \delta r^T \partial(\delta r)/\partial t \\ &= -\delta r^T [(\partial(\delta r)/\partial a_{1,ij})\, a'_{1,ij} + (\partial(\delta r)/\partial b_{1,ik})\, b'_{1,ik} + (\partial(\delta r)/\partial c_{1,pj})\, c'_{1,pj}] \\ &= -\delta r^T [C_1\, \nabla g(x)\, \eta_{1,ij}\, a'_{1,ij} + C_1\, \nabla g(x)\, \zeta_{1,ik}\, b'_{1,ik} + I_\rho\, g(x_j)\, c'_{1,pj}]. \end{aligned} \tag{7}$$

On the other hand, since $\|\partial(\delta r)/\partial(\delta u)\| \gg 1$, from Lemma 1 we have $\delta u = \delta r$. Therefore we have

$$J'_1(t) = -\delta u^T [C_1\, \nabla g(x)\, \eta_{1,ij}\, a'_{1,ij} + C_1\, \nabla g(x)\, \zeta_{1,ik}\, b'_{1,ik}] - \delta u_\rho\, g(x_j)\, c'_{1,pi}\,. \tag{8}$$

However, for $a'_{1,ij}$, $b'_{1,ik}$, and $c'_{1,pj}$ given by equation (4), we get

$$J'_1(t) = -\alpha_1 [\delta u^T C_1\, \nabla g(x)\, \eta_{1,ij}]^2 - \beta_1 [\delta u^T C_1\, \nabla g(x)\, \zeta_{1,ik}]^2 - \gamma_1 [\delta u_\rho\, g(x_j)]^2 \tag{9}$$

which is a negative definite scalar function, except when we have $\delta u = 0$ where the learning is complete. Therefore, from the second method of Liapunov, the learning controller $\mathfrak{C}$ with the weight adjustments given by equation (4), is asymptotically stable (i.e., it is convergent). That is, the connection weight matrices A_1, B_1, and C_1 in the neural inverse-dynamics block $\mathfrak{E}$ will be adjusted until $J_1(t)=0$, that is when $\delta u = u - v = 0$ or equivalently when $\delta r = r - v = 0$. However, since the feedback operator is linear, $\delta u = K_p e + K_d e' = 0$ implies that $e = e' = 0$, since e and e' are linearly independent. Therefore, $q = q_r$, and $q' = q'_r$ as time t approaches infinity (i.e., the manipulator's trajectory q, q' follow the desired trajectory q_r, q'_r). □

The neural network $\mathfrak{E}$ part of the controller $\mathfrak{C}$ is able to acquire the model of the inverse-dynamics of the manipulator after a sufficiently long period of training. After this, the robot with the inverse-dynamics block $\mathfrak{E}$ alone (i.e., without the error feedback block $\mathfrak{H}$), is able to perform the trained tasks very well. In addition, the robot is able to perform some new tasks satisfactorily. However, without the feedback block $\mathfrak{H}$, the robot is not quite able to compensate for unpredictable perturbations. It is easily seen, however, that leaving block $\mathfrak{H}$ in the controller loop after the period of training greatly improves the ability of the controller to compensate for perturbations. This is the motivation for the next design.

5. Neural Adaptive State Feedback Model for Learning Control

The learning controller $\mathfrak{C}$ in this section contains both a feedforward and a feedback neural network block. The feedback neural block $\mathfrak{H}$ in this design has substituted for the PD-type error feedback block, as in Figure 3.

The neural adaptive state feedback block $\mathfrak{H}$ is intended as an optimal state feedback controller, and contains two sub-networks. One is the dynamics identifier $\mathfrak{D}$, which realizes the dynamics model of the system's perturbation about the nominal operating point. The other is the state feedback $\mathfrak{F}$, which generates an optimal state feedback for disturbance compensation. The overall feedback network $\mathfrak{H}$ learns to generate the optimal state feedback

torques to eliminate perturbations. From the Linear Quadratic Control Theory, this network is equivalent to an optimal state feedback which continuously identifies the parameters of the perturbation dynamics of the manipulator, and from these, produces the optimum compensating torques.

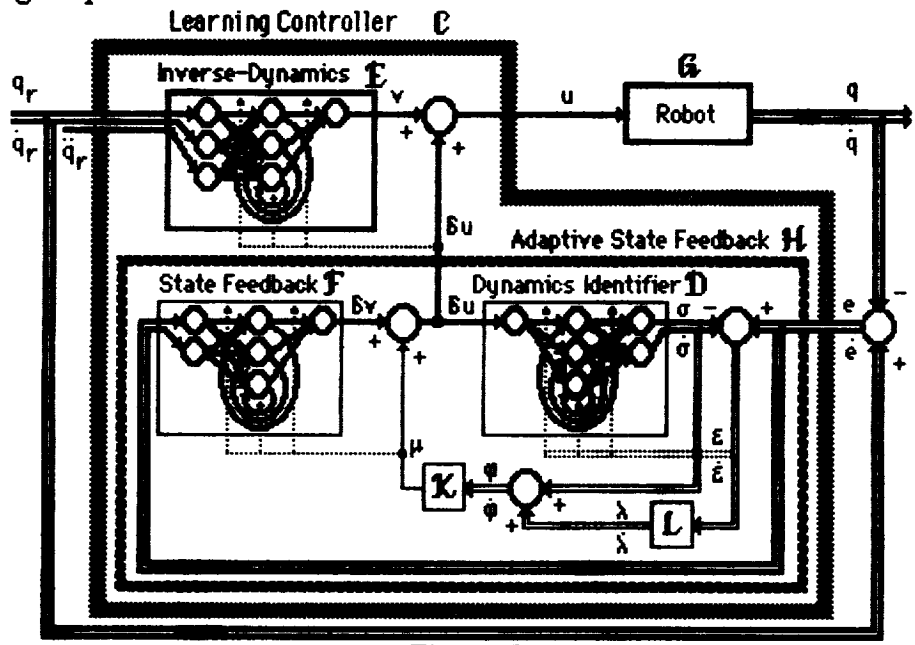

Figure 3

Feedback Network's Architecture

The input to neural block $\mathcal{H}$ is a 2n-vector of the angular position velocity errors e and e'. The outputs of the network are the n compensating torque signals δu.

The input-output relationship for the dynamics identifier network $\mathcal{D}$ is given by

$$y' = A_2\, g(y) + B_2\, v \tag{10}$$
$$\underline{\sigma} = C_2\, g(y)$$

where $v = [\delta u^T, 1]^T \in \mathbb{R}^{n+1}$, $y \in \mathbb{R}^M$, $\underline{\sigma}=[\sigma^T, \sigma'^T]^T \in \mathbb{R}^{2n}$ are respectively the input, state, and the output of the network. A_2, B_2, and C_2 are respectively the matrices of the network's state recurrence, input, and output connection weights.

The input-output relationship for the state feedback network $\mathcal{F}$ is given by

$$z' = A_3\, g(z) + B_3\, \underline{e} \tag{11}$$
$$\delta v = C_3\, g(z)$$

where $\underline{e}=[e^T, e'^T, 1]^T \in \mathbb{R}^{2n+1}$, $z \in \mathbb{R}^L$, $\delta v \in \mathbb{R}^n$ are respectively the input, state, and the output of the network, and A_3, B_3, C_3 are respectively the matrices of the network's state recurrence, input, and output connection weights. As shown in Figure 3, there are some internal feedback blocks $\mathcal{K}$ and $\mathcal{L}$ within the neural adaptive state feedback block $\mathcal{H}$ which are used primarily to provide a performance function for the networks' learning algorithm.

That is

$$\begin{aligned} \delta u &= \delta v + \mu \\ \mu &= \mathcal{K}(\varphi) = K_p\, \varphi + K_d\, \varphi' \\ \underline{\varphi} &= \underline{\sigma} + \underline{\lambda} \\ \underline{\lambda} &= \mathcal{L}(\underline{\varepsilon}) = L_p\, \varepsilon + L_d\, \varepsilon' \end{aligned} \tag{12}$$

where $\underline{\varphi}=[\varphi^T,\varphi'^T]^T$, $\underline{\sigma}=[\sigma^T,\sigma'^T]^T$, $\underline{\lambda}=[\lambda^T,\lambda'^T]^T$, $\underline{\varepsilon}=[\varepsilon^T,\varepsilon'^T]^T=[(e-\sigma)^T,(e'-\sigma')^T]^T$, $\mathcal{K}$ is a linear high gain feedback operator, and $\mathcal{L}$ is a linear feedback gain block.

In the feedback block $\mathcal{H}$, the neural dynamics identifier $\mathcal{D}$ approximates the input-output relationship of the dynamics of perturbations by forcing its outputs to follow the system errors e and e'. The neural state feedback $\mathcal{F}$, on the other hand, approximates the input-output relationship of an optimal state error feedback system by forcing its output to follow the input of the neural dynamics identifier $\mathcal{D}$. This, in effect, adjusts block $\mathcal{F}$ to approximate the inverse $\mathcal{D}^{-1}$ of the neural dynamics identifier $\mathcal{D}$.

Networks' Learning Rules

The learning algorithm used for the neural dynamics identifier network $\mathcal{D}$ is similar to that of inverse-dynamics network $\mathcal{E}$, i.e., the time derivative of the connection weight matrices A_2, B_2, and C_2 are given by [11]

$$\begin{aligned} a'_{2,ij} &= \alpha_2\, \underline{\varepsilon}^T \nabla g(y)\, C_2\, \eta_{2,ij} \\ b'_{2,ik} &= \beta_2\, \underline{\varepsilon}^T \nabla g(y)\, C_2\, \zeta_{2,ik} \\ c'_{2,pj} &= \gamma_2\, \underline{\varepsilon}_p\, g(y_j) \\ \eta'_{2,ij} &= A_2 \nabla g(y)\, \eta_{2,ij} + I_i\, g(y_j) \\ \zeta'_{2,ik} &= A_2 \nabla g(y)\, \zeta_{2,ik} + I_i\, v_k \end{aligned} \tag{13}$$

where $\underline{\varepsilon}=[\varepsilon^T,\varepsilon^T]^T$ is the network's output error, $\nabla g(y)=\partial g(y)/\partial y$ is the Jacobian matrix, and α_2, β_2, and γ_2 are small positive learning rate constants. The initial values of matrices A_2, B_2, and C_2 are selected randomly between -0.2 and 0.2, and $\eta_{2,ij}(0)=\zeta_{2,ik}(0)=0$.

The learning scheme for the neural state feedback network block $\mathcal{F}$ is similar; i.e., the time derivative of the connection weight matrices A_3, B_3, C_3, are given by

$$\begin{aligned} a'_{3,ij} &= \alpha_3\, \mu^T \nabla g(z)\, C_3\, \eta_{3,ij} \\ b'_{3,ik} &= \beta_3\, \mu^T \nabla g(z)\, C_3\, \zeta_{3,ik} \\ c'_{3,pj} &= \gamma_3\, \mu_p\, g(z_j) \\ \eta'_{3,ij} &= A_3\, \nabla g(z)\, \eta_{3,ij} + I_i\, g(z_j) \\ \zeta'_{3,ik} &= A_3\, \nabla g(z)\, \zeta_{3,ik} + I_i\, e_k \end{aligned} \tag{14}$$

where $\mu=\delta u-\delta v$ is the network's output error, $\nabla g(z)=\partial g(z)/\partial z$ is the Jacobian matrix, and α_3, β_3, and γ_3 are small positive learning rate constants. The initial values of matrices A_3, B_3, and C_3 are selected randomly between -0.2 and 0.2, and $\eta_{3,ij}(0)=\zeta_{3,ik}(0)=0$.

From Result 1, the inverse-dynamics neural network $\mathfrak{E}$ with its learning rule is able to realize the model of the inverse-dynamics $\mathfrak{G}^{-1}$ of the robot and to generate the required robot torque corresponding to the desired trajectory q_r and q'_r. From the Linear Quadratic Control Theory, in order to generate the compensating torque corresponding to the dynamics perturbations about the nominal trajectory of the robot, the adaptive state feedback neural network $\mathfrak{H}$ must identify the dynamics relation of the perturbations and correspondingly generate the optimal feedback according to some performance criterion. But for this the corresponding learning algorithm must be convergent (i.e., the dynamics of the learning system must be asymptotically stable).

Result 2

Consider the robotic manipulator given by the operator $\mathfrak{G}$, as in equation (1). Assume that the neural learning controller $\mathfrak{C}$, given by equations (3) and (10-12), is applied to the system, as shown in Figure 3. Let the feedback operator $\mathfrak{L}$ be a unity gain. Also let the high gain feedback block $\mathfrak{K}$ be such that the closed-loop system $(I+\mathfrak{G}\mathfrak{K})^{-1}\mathfrak{G}$ is stable and that $\|\mathfrak{G}\mathfrak{K}\| >> 1$. Then the neural learning controller $\mathfrak{C}$, together with the learning rules (4) and (13-14) is asymptotically stable. That is, the learning controller $\mathfrak{C}$ forces the manipulator's trajectory q and q' to follow the desired trajectory q_r and q'_r after a sufficiently long time.

Proof

From Result 1, since $\mathfrak{L}$ is a unity gain, $\mathfrak{K}$ is such that $(I+\mathfrak{G}\mathfrak{K})^{-1}\mathfrak{G}$ is stable, and $\|\mathfrak{G}\mathfrak{K}\| >> 1$, the learning process for the neural inverse-dynamics network $\mathfrak{E}$ is asymptotically stable.

Now let $\eta_{2,ij}=\partial y/\partial a_{2,ij}$, $\zeta_{2,ik}=\partial y/\partial b_{2,ik}$, $\eta_{3,ij}=\partial z/\partial a3,ij$, and $\zeta_{3,ij}=\partial z/\partial b3,ik$. Then, similar to the proof of Result 1, from the neural network's dynamics equations (10-11), we get

$$\eta'_{2,ij} = A_2\nabla g(y)\,\eta_{2,ij} + I_i\,g(y_j) \qquad (15)$$

$$\zeta'_{2,ik} = A_2\nabla g(y)\,\zeta_{2,ik} + I_i\,v_k.$$

$$\eta'_{3,ij} = A_3\nabla g(z)\,\eta_{3,ij} + I_i\,g(z_j)$$

$$\zeta'_{3,ik} = A_3\nabla g(z)\,\zeta_{3,ik} + I_i\,v_k.$$

Now considering the convergence of the feedback block $\mathfrak{H}$, let a performance function for the learning process of the dynamics identifier sub-network $\mathfrak{D}$ be defined by

$$J_2(t) = 0.5\,\underline{\varepsilon}(t)^T\underline{\varepsilon}(t) \qquad (16)$$

where $\underline{\varepsilon}=[\varepsilon^T,\varepsilon'^T]^T$ and $\varepsilon=e-\sigma$. Since $J_2(t)$ is positive definite and monotonically increasing, for asymptotic stability, $J'_2(t)$ must be negative definite. Using the chain rule, we get

$$J'_2(t) = \underline{\varepsilon}^T \partial(\underline{\varepsilon}')/\partial t \qquad (17)$$
$$= -\underline{\varepsilon}^T [C_2 \nabla g(y) \eta_{2,ij} a'_{2,ij} + C_2 \nabla g(y) \zeta_{2,ik} b'_{2,ik} + I_\rho g(y_j) c'_{2,pj}].$$

However, for the weight adjustments given by equation (13), we have

$$J'_2(t) = -\alpha_2 [\underline{\varepsilon}^T C_2 \nabla g(y) \eta_{2,ij}]^2 - \beta_2 [\underline{\varepsilon}^T C_2 \nabla g(y) \zeta_{2,ik}]^2 - \gamma_2 [\underline{\varepsilon}_\rho g(y_j)]^2 \qquad (18)$$

which is a negative definite scalar function, except when $\underline{\varepsilon}=0$ where learning is complete.

Similarly, let the learning performance function for neural block $\mathbf{F}$ be defined by

$$J_3(t) = 0.5\, \zeta(t)^T \zeta(t) \qquad (19)$$

where $\zeta = \delta r - \delta v$ is the output error of the network, $\delta r = r - v$, and r is such that $\mathbf{G}(r)=q_r$. Again, since $J_3(t)$ is positive definite and monotonically increasing, for asymptotic stability, $J'_3(t)$ must be negative definite. Similar to the earlier case, by the chain rule, we have

$$J'_3(t) = \zeta^T \partial(\zeta)/\partial t \qquad (20)$$
$$= -\zeta^T [C_3 \nabla g(z) \eta_{3,ij} a'_{3,ij} + C_3 \nabla g(z) \zeta_{3,ik} b'_{3,ik} + I_\rho g(z_j) c'_{3,pj}].$$

On the other hand, since $\|\mathbf{G}\| >> 1$, from Lemma 1 we have $\delta u = \delta r$ and hence $\mu = \zeta$. Therefore, for the weight adjustments given by equation (14), we get

$$J'_3(t) = -\alpha_3 [\mu^T C_3 \nabla g(z) \eta_{3,ij}]^2 - \beta_3 [\mu^T C_3 \nabla g(z) \zeta_{3,ik}]^2 - \gamma_3 [\mu_\rho g(z_j)]^2 \qquad (21)$$

which is a negative definite scalar function, except when $\mu=0$ where learning is complete. Therefore from the second method of Liapunov, this learning system is asymptotically stable. This means that the connection weights in the networks will be adjusted until $J_2(t)=0$ and $J_3(t)=0$, or equivalently $\varepsilon = e - \sigma = 0$, $\varepsilon' = e' - \sigma' = 0$, and $\mu = \delta u - \delta v = 0$. However, these imply that $e=e'=0$, and that the dynamics identifier sub-network $\mathbf{D}$ acquires the model of the dynamics of the perturbation system. Also, the optimal state feedback sub-network $\mathbf{F}$ becomes identical to the inverse $\mathbf{D}^{-1}$ of the dynamics identifier sub-network $\mathbf{D}$, which generates the compensating torque corresponding to the trajectory perturbation e and e'. Therefore, $q=q_r$ and $q'=q'_r$ as time t approaches infinity (i.e., the manipulator's trajectory q,q' follow the desired trajectory q_r, q'_r). □

6. Conclusion

In this paper, two neural learning controller designs have been considered. They mimic the functions of the cerebellum for the learning and control of voluntary movements and they have parallel processing capabilities which make them fast and adaptable. The designs have several promising attributes that make them very feasible solutions to current problems

in Robotics. Most importantly, such controllers are able to approximate the model of the inverse-dynamics of the robot, during the training period. This allows the robot to learn repetitive motions almost perfectly. But even above that, it can perform tasks that it has not been trained to do yet, and to perform them well. In addition, the second design has a good adaptation capability which allows the controller to compensate for unexpected disturbances.

Another advantage of these designs is that they do not require knowledge of the system parameters, and they are robust with respect to parameter variation and disturbances under a variety of tasks. Finally, the parallel processing property of these architectures makes them highly suitable for the integration of a multitude of sensory information into the motion controller networks.

7. References

[1] Rumelhart, D.E., McClelland, J.L., and the PDP Research Group, Parallel Distributed Processing, Volume 1, MIT Press, 1987.

[2] Raibert, M.H., "A Model for Sensorimotor Control and Learning," Biol. Cybern. 29, 1978.

[3] Eckmiller, R., "Neural Network Mechanisms for Generation and Learning of Motor Programs," ICNN, 1987.

[4] Psaltis, D., Sideris, A., and Yamamura, A., "Neural Controllers," 1st IEEE ICNN, 1987.

[5] Kawato, M., Uno, Y., Isobe, M. and Suzuki, R., "A Hierarchical Model for Voluntary Movement and Its Application to Robotics," ICNN, 1987.

[6] Kawato, M., Furukawa, K., and Suzuki, R., "A Hierarchical Neural Network Model for Control and Learning of Voluntary Movement," Biol. Cybern., 57, 1987.

[7] Pourboghrat, F., "Neuronal Controller for Robotic Manipulators," IASTED Int. Symp. Robotics Autom., 1988.

[8] Pourboghrat, F. and M.R. Sayeh, "Neural Network Learning Controller for Manipulators," INNS Conf. in Boston, MA, Sept. 1988.

[9] Pourboghrat, F. and M.R. Sayeh, "Neural Network Models for the Learning Control of Dynamical Systems with Application to Robotics," Springer-Verlag Lecture Notes in Control and Information Sciences (to appear). Also in Int. Conf. Advances in Comm. & Cont. Syst., Oct. 1988.

[10] Pourboghrat, F., "Neural Learning Controllers for Manipulators," IEEE Trans. Syst. Man Cybernetics (submitted).

[11] Pourboghrat, F., "A Learning Algorithm for Temporal Pattern Recognition," IEEE Trans. Syst. Man Cybernetics (submitted).