9 Computational Schemes and Neural Network Models for Formation and Control of Multijoint Arm Trajectory

Mitsuo Kawato

9.1 Introduction

In this chapter, we propose a computational model and two neural network models for trajectory formation and control of multijoint arm movement based on data from behavioral experiments and physiological knowledge.

First, in this section, we explain that the three problems, trajectory formation, inverse kinematics and inverse dynamics must be solved for control of voluntary movements. In the next section, it is shown that all the three problems are ill posed in the sense that the solutions to these problems are not unique. We propose two entirely different neural network models for resolving the ill-posedness of motor control.

In the first model, the minimum torque change criterion derived from data from human arm movement experiments is utilized to resolve the ill-posedness of arm trajectory formation. We propose a neural network model with a repetitive cascade structure which generates the minimum torque change trajectory. The model first acquires the forward dynamics model of a controlled object in the learning phase, then minimizes some energy to generate a trajectory by relaxation computation in the pattern generating phase.

In the latter half of this chapter, we will show that the feedback error learning scheme for motor learning that we proposed can be used to resolve the ill-posedness of motor control. This scheme is compared with alternative approaches such as direct inverse modeling or combination of forward and inverse modeling. We will show experiment data of feedback error learning for acquiring the inverse-dynamics model of a PUMA manipulator.

A computational model for voluntary movement is proposed (figure 9.1) which accounts for Marr's (1982) first level for understanding complex information-processing systems (i.e., computational theory).

Consider a thirsty person reaching for a glass of water on a table. The goal of the movement is to move the arm toward the glass which in turn will lead to reducing the thirst. First, one desirable trajectory in the task-oriented coordinates must be selected from an infinite number of possible trajectories which lead to the glass whose spatial coordinates are provided by the visual system (trajectory determination in figure 9.1). Second, the spatial coordinates of the desired trajectory must

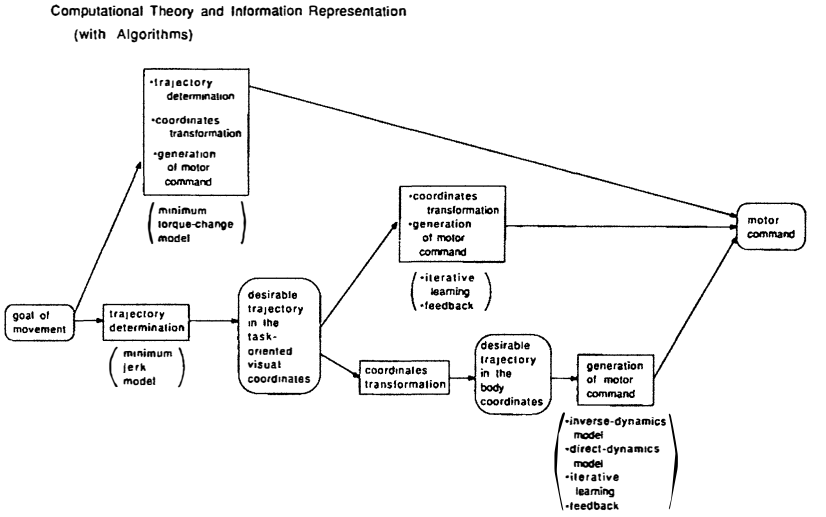


Figure 9.1
Computational theories and internal information representation in the brain for sensory-motor control of voluntary movement.

be reinterpreted in terms of a corresponding set of body coordinates, such as joint angles or muscle lengths (coordinates transformation in figure 9.1). Finally, motor commands, that is muscle torque, must be generated to coordinate the activity of many muscles so that the desired trajectory is realized (generation of motor command in figure 9.1). The second and the third problems are called the inverse kinematics problem and the inverse dynamics problem in robotics literature. Several lines of experimental evidence suggest that these three sets of information shown in figure 9.1, i.e., the desired trajectory in visual coordinates, the desired trajectory in body coordinates and the active torque, are internally represented in the brain (Kawato 1989).

However, it must be noted that here we do not adhere to the hypothesis of the step-by-step information processing shown on the bottom line in figure 9.1. Rather, our model indicates that there are other stages of information processing which can lead to the desired trajectory. In the middle line of figure 9.1, the motor command is obtained directly

from the desired trajectory represented in the task-oriented coordinates, that is, the two problems (coordinates transformation and generation of motor command) are simultaneously solved. Previously, we proposed that some areas of the sensory association cortex (areas 5 and 7) constitute the locus of this computation by an iterative learning algorithm (Kawato, Isobe, Maeda, and Suzuki 1988). That is, the motor command is not determined immediately, but in a stepwise, trial-and-error fashion over the course of a set of repetitions. In this motor learning situation, short-term memory of the time history of trajectory and torque are required. We successfully applied this learning scheme to trajectory control of the PUMA manipulator in the visual coordinates of two position sensor head cameras (Isobe, Kawato, and Suzuki 1988).

Further, in the uppermost line of figure 9.1, the motor command is calculated directly from the goal of movement, that is, the three problems (trajectory determination, coordinates transformation and generation of motor command) are simultaneously solved. The minimum torque change criterion and the repetitively structured cascade neural network model for trajectory formation, which I will propose in the first half of this chapter, correspond to the uppermost line of information processing in figure 9.1.

On the other hand, the feedback error learning scheme, which will be explained in the latter half of this chapter, provides a computational scheme for the final step of the bottom line in figure 9.1 (i.e. generation of motor command).

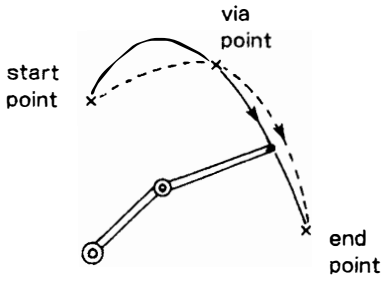
9.2 Ill-posed Problems in Sensory-motor Control

A problem is well posed when its solution exists, is unique and depends continuously on the initial data. Ill posed problems fail to satisfy one or more of these criteria. Most motor control problems are ill posed in the sense that the solution is not unique.

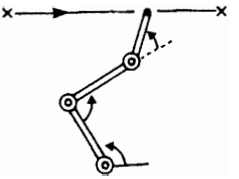
We list three ill-posed control problems in figure 9.2. First, consider the trajectory determination problem for human planar, two-joint arm movement, when the starting, intermediate and end points, as well as the movement time, are specified (figure 9.2, top). There is an infinite number of possible trajectories satisfying these conditions. Thus, the solution is not unique and the problem is ill posed.

The second ill-posed problem is the inverse kinematics problem in a redundant manipulator with excess degrees of freedom. For example, consider a three-degree-of-freedom manipulator in a plane (figure 9.2,

Trajectory Formation



Inverse Kinematics in Redundant Manipulator



Inverse Dynamics in Redundant Manipulator

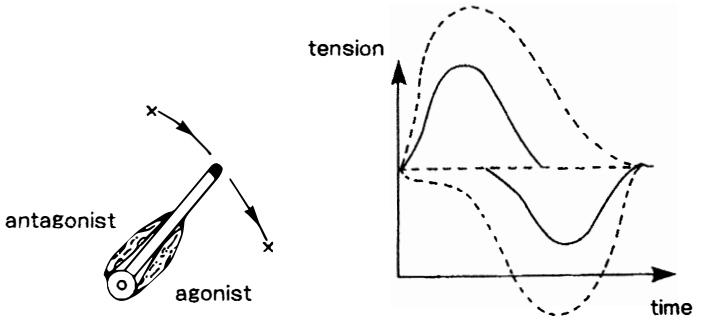


Figure 9.2
Three ill-posed problems in sensory-motor control.

middle). The inverse kinematics problem is to determine the three joint angles (three degrees of freedom) when the hand position in the Cartesian coordinates (two degrees of freedom) is given. Because of the redundancy, even when the time course of the hand position is strictly determined, the time course of the three joint angles cannot be determined uniquely. We note that human hands have excess degrees of freedom.

The third ill-posed motor control problem is the inverse dynamics problem in a manipulator with agonist and antagonist muscles (actuators). Consider a single joint manipulator with a pair of muscles (figure 9.2 bottom). The inverse dynamics problem is to determine time courses of tensions of agonist and antagonist muscles when the joint angle time course is determined. Even when the time course of the joint angle is specified, there are an infinite number of tension waveforms of the two muscles which realize the same joint angle time course, as indicated by solid and broken curves in figure 9.2 bottom.

To resolve the ill-posedness of these problems, we need to introduce some performance index other than the above conditions which specify movement pattern. We will propose such an objective function in the next section. This kind of objective function is still not a sufficient condition for resolving the ill-posedness. Computational schemes, algorithms and neural network models must be compatible with the objective function so that the objective function is naturally embedded into them. It is worthwhile to evaluate computational schemes and neural network models on the basis of whether they can cope with the ill-posedness inherent in a sensory-motor control.

9.3 Uno's Minimum Torque Change Model for Formation of Human Multijoint Arm Trajectory

Early studies of motor control concentrated on single-joint arm movements. Whereas, recently, several studies have been reported regarding the kinematic and dynamic aspects of multijoint arm movements. Morasso (1981) provided experimental data which suggests that the desired trajectory is first planned at the task-oriented (visual) coordinates. He measured human two-joint arm movements restricted to an horizontal plane, and found the following common invariant kinematic features. When a subject was instructed merely to move his hand from one visual target to another, his hand usually moved along a roughly straight path with a bell-shaped speed profile. Morasso also reported that, in contrast to the simple hand profile, the angular positions and velocity profiles of

the two joints (shoulder and elbow) were widely different according to the parts of the workspace in which movements were performed. Abend, Bizzi and Morasso (1982) investigated not only straight paths but also curved paths. In contrast to the point-to-point movements, when the subject was instructed to move his hand while avoiding an obstacle or along a self-generated curved path, the hand path appeared to be composed of a series of gently curved segments and the speed profile often had several peaks.

In order to account for these kinematic features, Flash and Hogan (1985) proposed a mathematical model, the *minimum jerk model*. They proposed that the trajectory, followed by the subject's arms, tended to minimize the following quadratic measure of performance: the integral of the square of the jerk (rate of change of acceleration) of the hand position (x, y), integrated over the entire movement.

$$C_J = \int_0^{t_f} \{(\frac{d^3x}{dt^3})^2 + (\frac{d^3y}{dt^3})^2\} dt \tag{9.1}$$

Flash and Hogan showed that the unique trajectory which yields the best performance was in good agreement with experimental data in one region of the workspace, that is, the region just in front of the body. Their analysis was based solely on the kinematics of movement, independent of the dynamics of the musculoskeletal system, and was successful only when formulated in terms of the motion of the hand in extracorporal space.

Based on the idea that the objective function must be related to the dynamics, Uno, Kawato and Suzuki (1987) proposed the following alternative quadratic measure of performance:

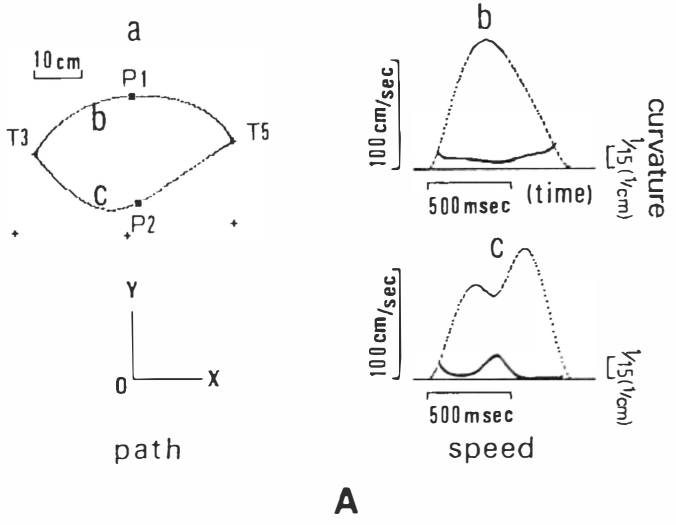
$$C_T = \int_0^{t_f} \sum_{i=1}^{n} (\frac{dT_i}{dt})^2 dt, \tag{9.2}$$

here, T_i is the torque fed to the i-th actuator out of n actuators. The objective function is the sum of the square of the rate of change of the torque, integrated over the entire movement. One can easily see that the two objective functions, C_J and C_T, are closely related. However, it must be emphasized that the objective function C_T critically depends on the dynamics of the musculoskeletal system. Due to this fact, it is much more difficult to determine the unique trajectory which minimizes C_T. Uno et al. (1987) overcame this difficulty by developing an iterative scheme, so that the unique trajectory and the associated motor command (torque) were obtained simultaneously. In this

work, locations of the end point and the intermediate point were given in the task-oriented Cartesian coordinates instead of the body coordinates. That is, the three above-mentioned problems—trajectory formation, coordinate transformation and generation of motor command, can be solved simultaneously using this algorithm. Mathematically, the iterative-learning scheme can be regarded as a Newton-like method in a function space.

Trajectories derived from the minimum torque change model are quite different from those of the minimum jerk model under the following behavioral situations: (i) excessive horizontal free movement between two targets; (ii) constrained and horizontal movement between two targets; (iii) vertical arm movement between two targets (see experimental data of Atkeson and Hollerbach 1985); (iv) free and horizontal movement via a point. Uno, Kawato, and Suzuki (1989) recently examined human arm trajectories under these conditions and found that the minimum torque change model reproduced these experimental data better than the minimum jerk model. Here, only the fourth case is explained. Figure 9.3 shows prediction by the minimum torque change model (A) and experimental data (B) of Uno, Kawato, and Suzuki (1989). For horizontal arm movements which had to travel between two targets passing through a specified point (a via-point), both the minimum jerk model and the minimum torque change model predicted curved hand paths with single-peaked or double-peaked speed profiles. It depended on the location of the via-point whether the hand speed profile had a single peak or two peaks. In both the models, if the via-point was located near to the line connecting the initial and the final targets, the hand speed profiles were single peaked; on the other hand, if the via-point was located further away from the line connecting two targets, highly curved movements were produced and the hand speed profiles were double-peaked. Furthermore, according as the curvature of hand path became larger, the valley in the double-peaked speed profile tended to be deeper. In this case, the peak in the path curvature was temporally associated with the valley in the speed profile (see figure 9.3A(**c**)).

However, when the via-point was located at a certain distance from the line connecting two targets, the two models predicted quite different trajectories. Consider two subcases, with identical start and end points, but with mirror-image via-points (see figure 9.3A(**a**)). That is, the start point $T3$ and the end point $T5$ are the same for these two subcases, but the two via-points $P1$ and $P2$ are located symmetrically with respect to the line connecting the common start and end points. Here, the via-point $P1$ is located further away from the body than the line $T3T5$ and



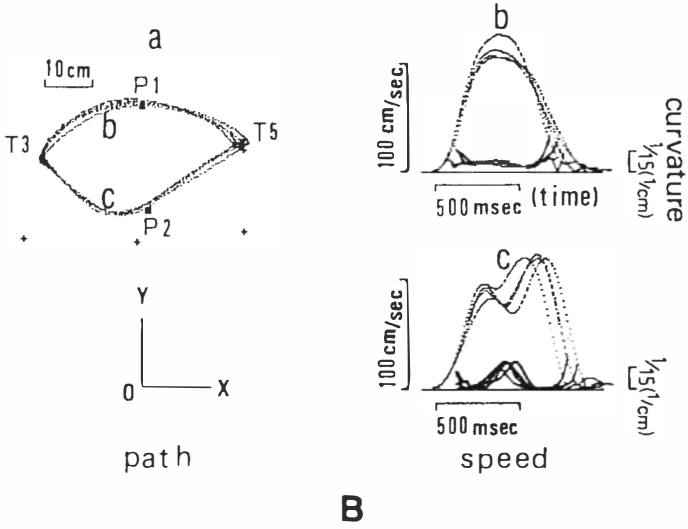


Figure 9.3
Free movements passing through a via-point, $P1$ or $P2$. $P1$ and $P2$ are located symmetrically with respect to the line connecting $T3$ and $T5$.

the via-point $P2$ is located nearer to the body than the line $T3T5$ as shown in figure 9.3A(**a**).

If one notices invariance of the criterion function C_J under translation, rotation and turning up, it is easy to see that the minimum jerk model predicts identical paths (with respect to turning up) and identical speed profiles for the two subcases. On the other hand, the minimum torque change model predicted two different shapes of trajectories corresponding to the two subcases; for the movement passing through the via-point $P1$, a convex curved path was formed and the associated speed profile had only one peak (figure 9.3A(**b**)); in contrast, for the movement passing through the via-point $P2$, a concave path was formed and the associated speed profile had two peaks (figure 9.3A(**c**)). Furthermore, the convex path ($T3 \rightarrow P1 \rightarrow T5$) and the concave path ($T3 \rightarrow P2 \rightarrow T5$) were not symmetric with respect to the line $T3T5$.

In short, the trajectory derived from the minimum jerk model is determined only by the geometric relation among the initial, final and intermediate points, whereas the trajectory derived from the minimum torque change model depends not only on the relation among these three points but also on the arm posture (in other words, the relative location of the shoulder for the three points).

Uno, Kawato, and Suzuki (1989) examined the same free movements passing through a via-point in human subjects. Specifying via-points $P1$ and $P2$ as shown in figure 9.3B, Uno, Kawato, and Suzuki (1989) measured via-point movements, $T3 \rightarrow P1 \rightarrow T5$ and $T3 \rightarrow P2 \rightarrow T5$. The convex path movements ($T3 \rightarrow P1 \rightarrow T5$) were quite different from the concave path movements ($T3 \rightarrow P2 \rightarrow T5$). In particular, the speed profiles of the former were single peaked as shown in figure 9.3B(**b**), while those of the latter were double peaked as shown in figure 9.3B(**c**). These experimental results were consistent with the predictions of the minimum torque-change model.

9.4 Repetitively Structured, Time-Invariant Cascade Neural Network Model for Vector Field of Dynamics

Since the dynamics of the human arm or the robotic manipulator is nonlinear, the task of finding a unique trajectory which minimizes C_T is a nonlinear optimization problem. The central nervous system does not seem to adopt the iterative algorithm which we proposed in Uno, Kawato, and Suzuki (1987). It was reported that some neural network models can solve difficult optimization problems such as the traveling

salesman problem or early visions by minimizing "energy" through the network dynamics. Previously, we proposed a neural network model which automatically generates a torque to minimize C_T without explicit handling of the cost function (Kawato, Uno, Isobe, and Suzuki 1988). This network can be regarded as one example of autonomous motor pattern generators such as a neural oscillator for rhythmic movements.

Here, I propose a repetitively structured, time-invariant, cascade neural network model (figure 9.4) for trajectory formation based on minimum torque change criterion, which is an extension of our previous neural network model. It can learn the vector field of the ordinary differential equation which describes the forward dynamics of the controlled object. The model consists of many identical four-layer network units which are connected in a cascade formation through a unit weight serial connection from the fourth layer of the k-th network unit to the first layer of the $k+1$-th network unit, and a unit weight connection bypath from the fourth layer of the k-th network unit to the fourth layer of the $k+1$-th network unit.

The network unit consists of four layers of neurons. The first layer represents the time course of the trajectory and the torque. The third layer represents the change of the trajectory within a unit of time, that is, the vector field multiplied by the unit of time. The second layer is expected to provide necessary nonlinear transformations for learning the vector field multiplied by the unit of time. The fourth layer and the output line on the right side represent the estimated time course of the trajectory.

We divide operations of this network into the learning phase (figure 9.4) and the pattern-generating phase (figure 9.5). In the learning phase, this network acquires an internal model of a vector field of forward dynamics of the controlled object between the first and the third layers using synaptic plasticity, while monitoring the actual trajectory from the controlled object as a teaching signal. In the pattern-generating phase, electrical coupling between neighboring neurons in the first layer is activated (figure 9.5). Then the network changes its state autonomously through feedforward synaptic connections and feedback propagation of error signals. The stable equilibrium state of the network corresponds to the minimum energy state and, hence, the network outputs the torque which realizes the minimum torque change trajectory.

For simplicity, let us consider a two-joint manipulator in a plane. The hand position is represented in the Cartesian coordinates (x, y). The forward kinematics and dynamics of the manipulator can be described

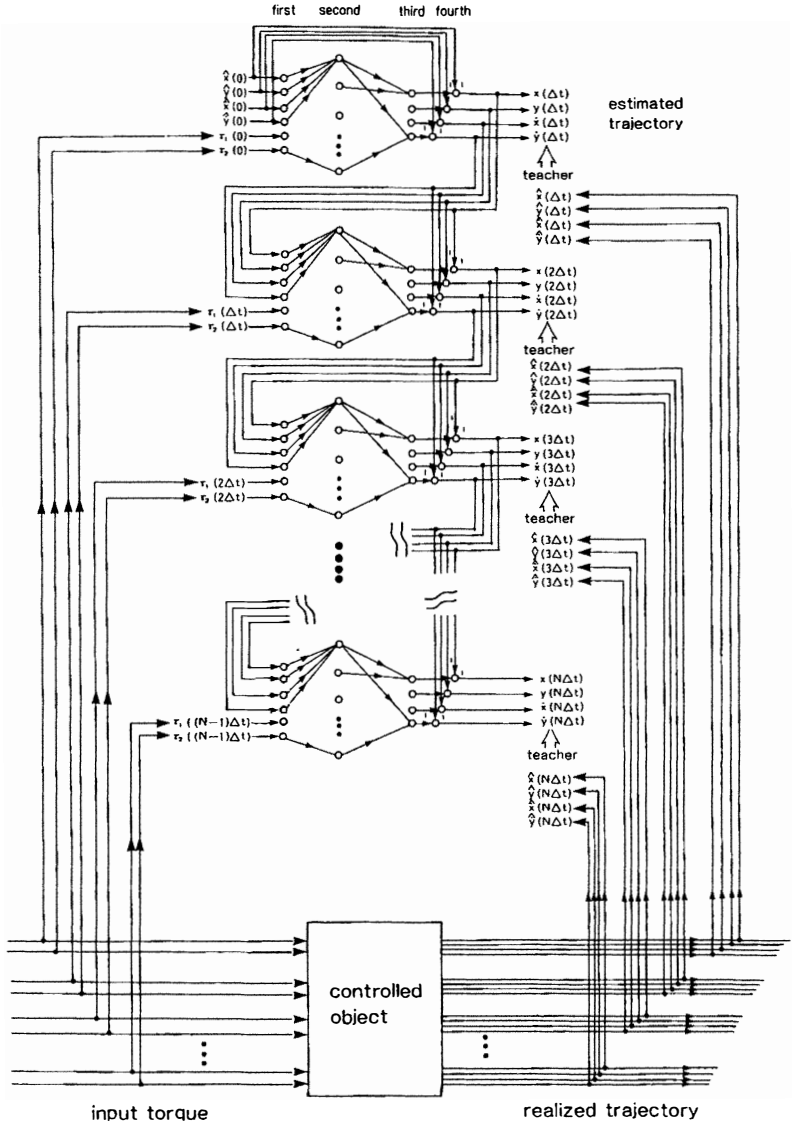


Figure 9.4
A repetitive neural network model learns and minimizes energy for generation of torque waveforms which realize the minimum torque change arm trajectory. Energy learning mode.

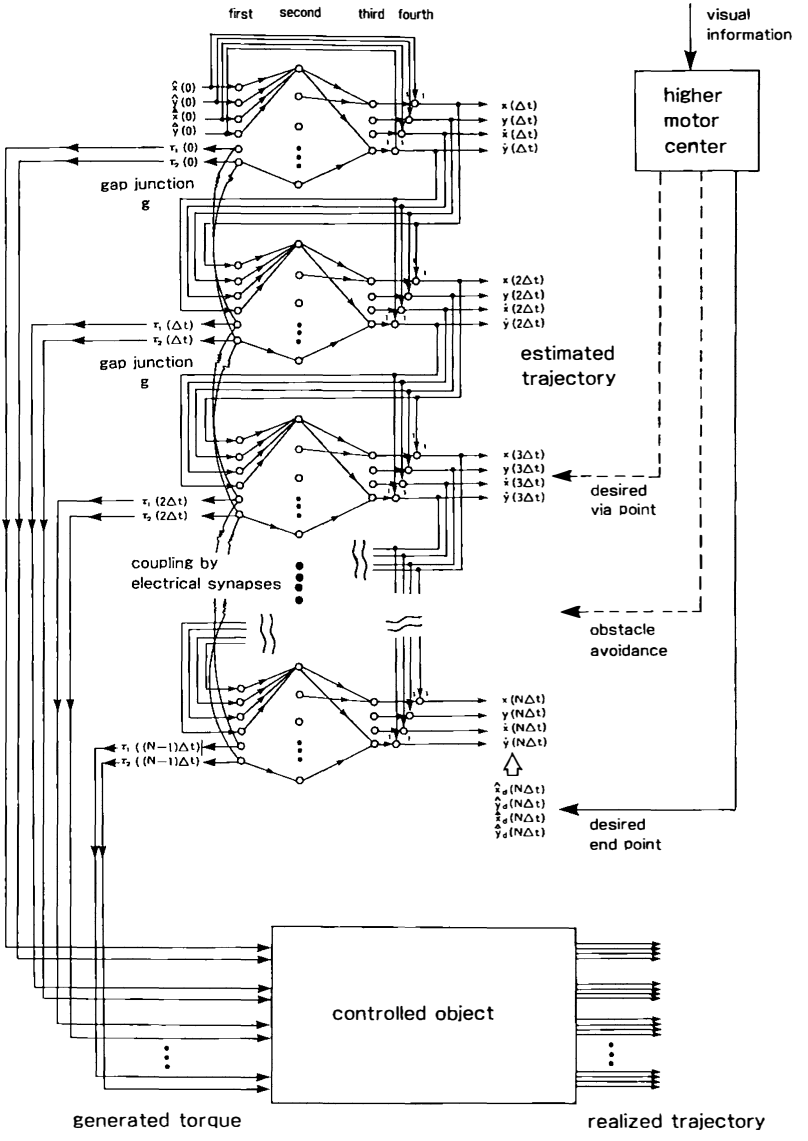


Figure 9.5
A repetitive neural network model learns and minimizes energy for generation of torque waveforms which realize the minimum torque change arm trajectory. Energy minimization mode.

by the following ordinary differential equations.

$$dx/dt = \dot{x} \tag{9.3}$$
$$dy/dt = \dot{y} \tag{9.4}$$
$$d\dot{x}/dt = f(x, y, \dot{x}, \dot{y}, \tau_1, \tau_2) \tag{9.5}$$
$$d\dot{y}/dt = g(x, y, \dot{x}, \dot{y}, \tau_1, \tau_2). \tag{9.6}$$

Here, $\tau_1(t)$ and $\tau_2(t)$ represent torque waveforms fed to the shoulder and the elbow. f and g are nonlinear functions which determine nonlinear dynamics and kinematics of the manipulator. The time t is digitized with the increment Δt. The k-th network unit corresponds to the k-th time point, $k\Delta t$. Through supervised learning, each network unit acquires a model of the vector field of the forward kinematics and dynamics equation, that is the right side of equations 9.3-9.6, multiplied by the time increment Δt. Since the vector field is time invariant, all the network units are identical. That is, the number of neurons and the synaptic weights are exactly the same for all units. Operation of each neuron is assumed as linear weighted summation of synaptic inputs and the sigmoid nonlinear transformation. The input-output transformation of the neurons in the first and the fourth-layers is assumed linear. The fourth-layer neurons in the k-th unit output the estimated trajectory, $x(k\Delta t), y(k\Delta t), \dot{x}(k\Delta t), \dot{y}(k\Delta t)$ at time $k\Delta t$, which are the summation of their two synaptic inputs, i.e., the outputs of the third layer in the k-th unit and outputs from the fourth layer neurons in the $(k-1)$st unit. The first four neurons in the first layer of the first unit represent the initial hand position $\hat{x}(0), \hat{y}(0), \hat{\dot{x}}(0), \hat{\dot{y}}(0)$. In short, the cascade formation of the network constitutes a hardware implementation of Euler's method of the numerical integration of the ordinary differential equations.

Because the model faithfully reproduces the time structure of the dynamical system, the following intrinsic properties of the flow of the dynamical system are embedded into the cascade structure of the model. (i) continuity of the solution of the dynamical system with respect to the time and the initial condition; (ii) group property of the flow ψ: $\psi(t+s;x) = \psi(t;\psi(s;x))$; (iii) causality: trajectory at some time does not depend on the torque input after that time.

In the learning phase (figure 9.4), the common input torque are fed to both the controlled object and the neural network model. The realized trajectory from the controlled object is used as a teaching signal to acquire the forward model of the manipulator between the first and the third layers of the network unit. The steepest descent method is

introduced for learning synaptic weights in the network unit, with the following output error function E for the output of the fourth layer.

$$E = \sum_{c} \sum_{k=1}^{N} \{\hat{X}_c(k\Delta t) - X_c(k\Delta t)\}^2. \tag{9.7}$$

Here, c represents the pair of input torque and the resulting trajectory. X denotes the vector $(x, y, \dot{x}, \dot{y})^T$ which represents the trajectory estimated by the neural network model. $\hat{X}$ represents the trajectory realized by the controlled object. The backpropagation learning algorithm (Rumelhart, Hinton, and Williams 1986) can be extended to this case as follows. The error signal $\delta^4_{i,k}$ of the i-th neuron in the fourth layer of the k-th network unit is the summation of the error between the realized trajectory and the estimated trajectory at time $k\Delta t$, the error signal of the i-th neuron in the fourth layer of the $(k+1)$-st network unit and the error signal of the i-th neuron in the first layer of the $(k+1)$-st network unit.

$$\begin{aligned} \delta^4_{i,k} &= \sum_{c} \{\hat{X}_{c,i}(k\Delta t) - X_{c,i}(k\Delta t)\} + \delta^4_{i,k+1} + \delta^1_{i,k+1} \\ & k = 1, 2, ..., N; i = 1, 2, 3, 4. \end{aligned} \tag{9.8}$$

The error signal of neurons in the third layer is the same as that of the fourth layer. The error signals of neurons in the first and the second layers are computed from those in the thirrd layer as in the usual back propagation learning rule. After the incremental changes of synaptic weights between the first and the second layers, and between the second and the third layers are calculated independently in every network unit, all changes for the corresponding synaptic weight are summed to be the net change. This procedure guarantees identity of all network units.

In the pattern-generating phase (figure 9.5), electrical couplings between neurons representing torques in the first layer of the neighboring units are activated while torque inputs to the first layer and the teaching signal to the fourth layer are suppressed. Instead, the central commands which specify the desired end point, the desired via point and the locations of obstacles to be avoided are given to the fourth layer from the higher motor center which receives the necessary visual information for the trajectory specification. Locations of the end point, the intermediate point and the obstacle are given in the task-oriented Cartesian coordinates to the fourth layer of network units.

The error signals for all neurons are calculated from equation 6.6 exactly the same as in the learning phase, while replacing the realized

trajectory as the teaching signal by the desired end point, the via-point, obstacles and so on, as the objective signal. The error signals are not used for synaptic modification. But, the error signals to the torque neurons in the first layer are used for dynamical state change of the model. That is, the network changes its state autonomously by feedforward synaptic connections and by error signals backpropagated through the cascade structure while obeying the following differential equation.

$$\begin{aligned} dm^1_{i,k}/ds &= \delta^1_{i,k} + g(m^1_{i,k+1} - 2m^1_{i,k} + m^1_{i,k-1}) \\ & k = 1, 2, ..., N; i = 5, 6. \end{aligned} \tag{9.9}$$

Here, s is the time of neuron state change and has nothing to do with the movement time t. m represents the membrane potential and the output signal of the neuron in the first layer, and g denotes the electrical conductance of the gap junction. We note that calculation of the error signal $\delta^1_{i,k}$ at every instant s in equation 6.9 requires both the feedforward calculation of every neuron state through the entire network with learned synaptic weights, and the feedback calculation of the error signal by equation 6.8 through the entire network.

This procedure is equivalent to impose the potential function which corresponds to the conditions of the movement. In the case of obstacle avoidance, we impose a quadratic potential function which is positive inside the location of the obstacle and zero outside the obstacle. In the case of the intermediate point and the end point, the quadratic potential is minimum at the site of these points. So, the potential of the obstacle is inverted in comparison with the potential of the end and intermediate points. However, the calculation of error signals is exactly the same in both cases.

It can be shown that the network dynamics has the following Liapunov function or "energy."

$$L = \{\hat{X}_d(N\Delta t) - X(N\Delta t)\}^2 + g\sum_{i=5}^{6}\sum_{k=1}^{N}(m^1_{i,k} - m^1_{i,k-1})^2 \tag{9.10}$$

The first term of the Liapunov function requires that the hand reaches the end point and the second term guarantees the minimum torque change criterion. The stable equilibrium point of the network dynamics corresponds to the minimum energy state, and, hence, the network outputs the torque, which realizes the minimum torque change trajectory. The conductance g is slowly decreased during relaxation of the state point to the equilibrium similarly to the temperature in "simulated annealing."

The performance of the proposed network model was examined in computer simulation experiments using a two-joint manipulator model of human arm as a controlled object (Maeda et al. 1989). Maeda et al. have successfully shown that the network produced a good approximate trajectory on minimum torque change criterion for the movements between pairs of target, movements where the intermediate point is specified, and movements where obstacles must be avoided. In the via-point movement case, we did not specify the time when the via-point must be passed. Instead, the network model autonomously finds the best time to pass the via-point on the minimum torque change criterion. In the obstacle avoidance case, we specified only the location of the obstacle in the task-oriented Cartesian coordinates. Because locations of end points, via-points and obstacles were given in the task-oriented Cartesian coordinates in this study, the neural network model solved the problem of inverse kinematics as well as the problems of trajectory formation and inverse dynamics.

The first term of the Liapunov function is the data constraint imposed by trajectory specification, and the second term is the smoothness constraint. Introduction of the second energy as electrical couplings resolves the ill-posedness of trajectory formation. The network learns the first term of energy as synaptic weights, and then minimizes the total energy.

This model has several conceptual similarities with the motor program subnetwork conjoined with the forward model subnetwork proposed by Jordan (1989). Jordan showed computer simulation of trajectory formation for a one-joint arm based on minimum jerk model. He noted that it is straightforward to implement the minimum torque change criterion in his model. Our model and Jordan's model both uses the forward model of the controlled object. The idea to resolve the ill-posedness of motor control by introducing smoothness constraint such as the minimum jerk criterion or the minimum torque change criterion is also common.

The essential difference of the two models is in the method of satisfying the smoothness constraint. In our model, relaxation of state vectors of the model (energy minimization) is utilized for attaining the smoothness constraint. Because of this, torque values at different time must be independently represented in the model, and hence time is represented spatially and the objective function is guaranteed by electrical coupling. On the other hand, in Jordan's model, time is represented as time in a recurrent network and the smoothness constraint is guaranteed by an error term based on comparing the activations of units at adjacent moments in time. In this sense, the smoothness constraint is embedded in synaptic weights of the subnetwork through learning.

One of the advantages of Jordan's model is that it can generate a trajectory in real time once it learns the forward model of the controlled object and the smoothness constraint. On the other hand, in our model, the relaxation time (typically hundreds of iteration) is required for trajectory formation. We recently found that multigrid, multiresolution (multi-time-scale) extension of our network can calculate the trajectory an order of magnitude faster than the original model. Research in this direction might be able to overcome the shortcoming of our model.

One advantage of our model is that it can generate any trajectory regardless of locations of the end point, the intermediate points and obstacles to be avoided once it learns the forward model of the controlled object. On the other hand, in Jordan's model, the motor learning subnetwork must learn many instances of trajectories with various locations of the end points, the intermediate points and obstacles so that it can generate them.

Jordan (1988, 1989) showed that his model can resolve the ill-posedness in the inverse kinematics, the inverse dynamics and the trajectory formation. We also note that the proposed repetitive network model can not only resolve the ill-posedness in the trajectory determination problem but can also resolve the ill-posedness in inverse kinematics and inverse dynamics problems for redundant manipulators (figure 9.2, middle and bottom). In these situations, the desired trajectory in the task-oriented coordinates must be fed to the neurons in the fourth layer of all units as the objective signal in the pattern-generating phase. If we introduce the electrical junctions between the neurons in the third layer representing change of the velocity instead of the first layer, the network can generate torque which realizes the minimum jerk trajectory.

Examination of human multijoint arm movement in the three-dimensional space while avoiding obstacles, and comparison of the experimental data with predictions of the present model are our future problems.

9.5 Hierarchical Neural Network Model for Control and Learning of Voluntary Movement

Ito (1970) proposed that the cerebrocerebellar communication loop is used as a reference model for the open-loop control of voluntary movement. Allen and Tsukahara (1974) proposed a comprehensive model which accounts for the functional roles of several brain regions in the control of voluntary movements. Gutkahara and Kawato (1982) proposed a

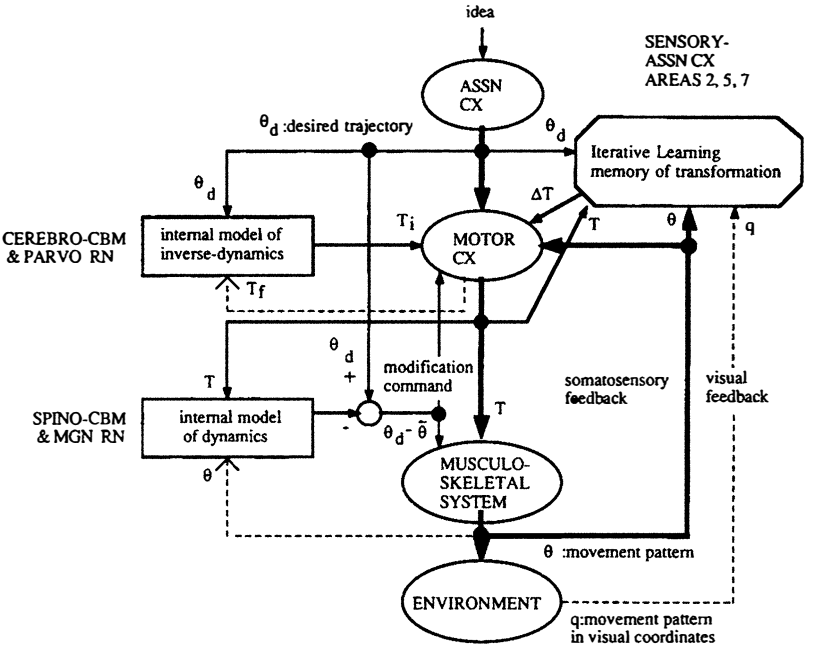


Figure 9.6
A hierarchical neural network model for control and learning of voluntary movement.

theoretical model of the cerebro-cerebello-rubral learning system based on recent experimental findings of the synaptic plasticity. Expanding these previous models and the adaptive filter model of the cerebellum (Fujita 1982), we proposed a neural network model for the control and learning of voluntary movement (Kawato, Furukawa, and Suzuki 1987).

In our model shown in figure 9.6, the association cortex sends the desired movement pattern θ_d, expressed in the body coordinates, to the motor cortex where the motor command T, i.e., the torque to be generated by muscles, is then somehow computed. The actual movement pattern θ is measured by proprioceptors and returned to the motor cortex via the transcortical loop. Then, feedback control can be performed utilizing error in the movement trajectory. However, both feedback delays and small gains limit the controllable speeds of motions.

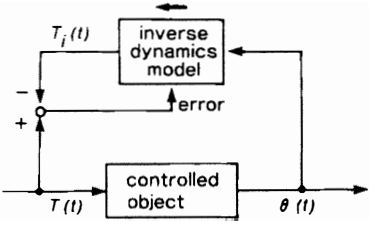
The cerebrocerebellum-parvocellular part of the red nucleus system receives synaptic inputs from wide areas of the cerebral cortex but does not receive peripheral sensory input. That is, it monitors both the desired trajectory and the motor command but it does not receive information about the actual movement. Within the cerebrocerebellum-parvocellular red nucleus system, an internal neural model of the inverse dynamics of the musculoskeletal system is acquired. The inverse-dynamics of the musculoskeletal system is defined as the nonlinear system whose input and output are inverted (trajectory is the input and motor command is the output). Once the inverse-dynamics model is acquired by motor learning, it can compute an appropriate motor command T_i directly from the desired trajectory θ_d.

9.6 Several Computational Approaches for Learning of Inverse Dynamics Model of a Controlled Object

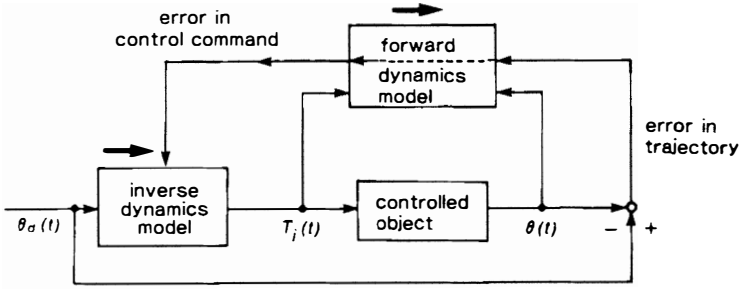
In this section we compare several computational approaches for learning of inverse-dynamics model of a controlled object. Figure 9.7 shows three different approaches to acquire the inverse dynamics model. In this figure, the block named "inverse dynamics model" is just a black box with its input and output. One can realize it by many kinds of neural network models such as the feedforward multilayer network (multilayer perceptron), the prepared nonlinearity network, the simple table lookup method, CMAC, the nearest neighborhood search with content addressable memories, etc. In this section, we do not mind what kind of neural network model constitutes the inverse-dynamics model. Our purpose is to compare properties of different computational approaches to learn the inverse dynamics model, which are somewhat independent of types of neural network model which actually constitutes the inverse dynamics model.

The simplest approach for acquiring the inverse dynamics model of a controlled object is shown in figure 9.7a. The manipulator receives the torque input $T(t)$ and outputs the resulting trajectory $\theta(t)$. The inverse dynamics model is set in the opposite input-output direction to that of the manipulator, as shown by the arrow. That is, it receives the trajectory as an input and outputs the torque $T_i(t)$. The error signal $s(t)$ is given as the difference between the real torque and the estimated torque: $s(t) = T(t) - T_i(t)$. This approach to acquire an inverse dynamics model is referred to as direct inverse modeling by Jordan (1988). Direct inverse modeling was used by Albus (1975), Miller

a. direct inverse modeling



b. forward and inverse modeling



c. feedback error learning

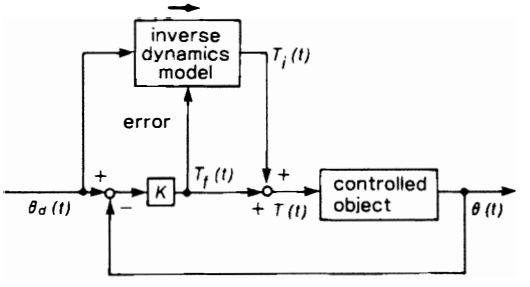


Figure 9.7
Three computational schemes for learning inverse- dynamics model of a controlled object. (a) Direct inverse modeling. (b) Forward and inverse modeling. (c) Feedback error learning scheme.

(1987), Miller, Glanz, and Kraft (1987), Kuperstein (1988), and Atkeson and Reinkensmeyer (1988). However, in Atkeson and Reinkensmeyer (1988), the content-addressable memories (CAM) are used as the inverse dynamics model, and hence the learning can be accomplished within one trial. In this sense, the error signal needs not be calculated in their case. Furthermore, the CAM has no direction built in, and hence it can use the same data for both the forward and inverse dynamics.

Figure 9.7b shows the scheme of combining a forward model and the inverse model, proposed by Jordan and Rumelhart (see Jordan 1988; Jordan and Rosenbaum 1988). First, the forward model of the controlled object is learned by monitoring the input $T_i(t)$ and the output $\theta(t)$ of the controlled object. Next, the desired trajectory $\theta_d(t)$ is fed to the inverse model to calculate the feedforward motor command $T_i(t)$. The resulting error in the trajectory $\theta_d(t) - \theta(t)$ is back propagated through the forward model to calculate the error in the motor command, which is then used as the error signal for training the inverse model.

Figure 9.7c shows the alternative computational approach which we proposed and termed, feedback error learning (Kawato, Furukawa, and Suzuki 1987). This block diagram includes the motor cortex (feedback gain K and summation of feedback and feedforward commands), the transcortical loop (negative feedback loop) and the cerebrocerebellum-parvocellular red nucleus system (inverse-dynamics model). The total torque $T(t)$ fed to an actuator of the manipulator is the sum of the feedback torque $T_f(t)$ and the feedforward torque $T_i(t)$, which is calculated by the inverse dynamics model. The inverse-dynamics model receives the desired trajectory θ_d represented in the body coordinates, such as the joint angles or muscle lengths, and monitors the feedback torque $T_f(t)$ as the error signal. It is expected that the feedback signal tends to zero as learning proceeds. This has been proved recently. We call this learning scheme feedback error learning, emphasizing the importance of using the feedback torque (motor command) as the error signal of the heterosynaptic learning.

We will compare the three computational schemes for acquiring the inverse model of the controlled object (see table 9.1 for reference).

The direct inverse modeling does not seem to be used in the central nervous system because of the following reasons. First, after the inverse dynamics model is acquired, large-scale connection changes must be carried out before it can be input from the desired trajectory instead of the actual trajectory, while preserving the minute one-to-one correspondence, so that it can be used in feedforward control. So, other supervising neural networks must determine when the connection change

Table 9.1
Comparisons of the three computational schemes for acquiring the internal inverse model of the controlled object.

	Direct Inverse Modeling	Forward and Inverse Modeling	Feedback Error Learning
learning & control	separated (biology) simultaneous (engineering)	control after forward learning	simultaneous
hardware requirement	memory copy or large scale connection change	forward model	feedback controller
dynamic change of controlled object	no (biology) yes (engineering)	first relearns forward model	yes
goal-directed learning	no	yes	yes
inverse kinematics	yes	yes	yes but requres sophisticated feedback controller
redundancy (many to one problem)	no	yes	yes

should be done are required. Because this method separates the learning and control modes, it is not easy to cope with the dynamics change of a controlled object. From an engineering point of view, memory copy from the learning hardware, which receives the realized trajectory, to the control hardware, which receives the desired trajectory, is more convenient and efficient, and was utilized by Miller (1987) and Miller, Glanz, and Kraft (1987). But memory copy seems unnatural in the biological system. If the memory copy scheme is utilized instead of the connection change, the direct inverse modeling approach can cope with the dynamic change of the controlled object in real time since learning and control can be performed simultaneously. In summary, the direct inverse modeling only requires "separated" learning and control with regard to biologically realistic structures. In practice, simultaneous control and training is easy to implement.

In engineering application, one drawback of the direct inverse modeling approach seems to be that it does not necessarily achieve a particular target trajectory $\theta_d(t)$, even when the training period is sufficiently long. In this sense, the learning is not "goal-directed" (Jordan and Rosenbaum 1988). Let us suppose that the control objective is to realize

some desired trajectory $\theta_d(t)$ by using the direct inverse modeling and the table lookup-type learning network. Initially, we need some fixed feedforward controller and feedback controller for generating training examples $T(t)$ and $\theta(t)$. If the control problem is difficult and/or the initial controller is poor, the realized trajectory $\theta(t)$ might be outside the generalization range (e.g., range of CMAC characteristic function) of the desired trajectory $\theta_d(t)$. Then, the memory content of the motor command at the address of the desired trajectory is not refreshed. Consequently, even if we apply the desired trajectory to the network many times, the control performance does not improve at all. The nearest neighborhood search utilized by Atkeson and Reinkensmeyer (1988) also suffered from the "stuck state" phenomena after several iterations for the particular target trajectory. The stuck state might be explained as follows. Let the pair (θ_{NN}, T_{NN}) be the nonempty memory content which is the nearest neighbor of the desired trajectory point θ_d. Because the dynamics of the controlled object depends on the past history of the trajectory and the effect of the feedback controller on the motor command, the actual trajectory point θ_a may be further from θ_d than θ_{NN}. Then, for this trial, only the memory content for θ_a is refreshed as the summation of T_{NN} and the feedback motor command. θ_a is not used to generate the next motor command, since θ_{NN} is closer. In the next trial exactly the same thing happens, and the memory content is not changed. It will be most interesting to see in the future whether combinations of multilayer networks, which are expected to have more generalization capability than table lookup methods, will exhibit goal directed learning properties with direct inverse modeling.

The forward and inverse modeling approach of Jordan and Rumelhart is goal directed because the error for learning is defined as the square of the difference between the desired trajectory and the realized trajectory.

The feedback error learning approach is also goal-directed even if we use a fine-grained table lookup learning network. The iterative performance of feedback error learning with a simple table lookup memory for a single desired trajectory presented repeatedly is similar to our Newton-like method in iterative learning control (Kawato, Isobe, Maeda, and Suzuki 1988). The feedforward motor command $T_{ff}^{k+1}(t)$ calculated by the inverse dynamics model in the $k+1$-st iteration of the single target trajectory is the summation of the feedforward torque $T_{ff}^{k}(t)$ and the feedback torque in the k-th iteration:

$$T_{ff}^{k+1}(t) = T_{ff}^{k}(t) + K_p\{\theta_d(t) - \theta^k(t)\} + K_v\{\dot{\theta}_d(t) - \dot{\theta}^k(t)\}, \qquad (9.11)$$

here, K_p and K_d are feedback gains of the proportional term and the differential term, respectively. The upper suffix k indicates the number of the iteration. Because the above equation for modification of the feedforward motor command does not contain the term which is proportional to the error in the acceleration of trajectory, we cannot mathematically prove the exponential convergence rate for the feedback error learning approach, which was guaranteed for the Newton-like method (Kawato, Isobe, Maeda and Suzuki 1988). However, it is intuitively clear that the above scheme converges albeit very slowly if the feedback gains are small enough.

The direct inverse modeling method can not cope with the second and the third ill-posed problems shown in figure 9.2, unless some specific mechanisms, such as specifying some joint angles or mechanical impedance, are introduced. Jordan (1988) explained this reason in the one to many inverse kinematics problem associated with motor control of redundant manipulators with excess degrees of freedom. The forward and inverse modeling approach can resolve the ill-posedness by learning some performance index as synaptic weights in the inverse model. The feedback controller selects one specific motor command even in the inverse dynamics and inverse kinematics problems for redundant manipulators. But the desired trajectory can not be exactly realized only by the feedback control. The feedback error learning approach can resolve the ill-posedness in the second and the third problems shown in figure 9.2 because of the above-mentioned good characteristics inherent in the feedback controller. We showed that the feedback error learning network can accurately realize the desired trajectory in the Cartesian coordinates for a three-link manipulator in a plane (Kano, Kawato, and Suzuki 1989). The selected joint angle time courses depended on the initial posture of the manipulator and the gains of the feedback controller.

The feedback error learning scheme does not require the teaching signal or the desired output for the neural network controller. Instead, the feedback torque is used as the error signal. The control and learning are performed simultaneously. Furthermore, backpropagation of the error signal through the controlled object or through a forward model of the controlled object is not necessary. The difficulty of the feedback error learning approach in its application to the inverse kinematics problem is that it requires a sophisticated feedback controller with such as the transpose of the Jacobian or the pseudoinverse of the Jacobian of the forward kinematics for generating the feedback motor command.

Above comparisons of different computational schemes and those shown in table 9.1 were made based only on sparse experimental data

and a small number of literatures within a very short period. So, we do not intend to draw any conclusions from the comparisons. What we would like to emphasize is that we have choices from several different computational approaches for learning an inverse model of the controlled object other than the problem regarding what kind of neural networks should be used for the inverse model itself, such as the simple table lookup method, CMAC, the nearest neighborhood search with content- addressable memories, etc. It will be very important to continue systematic comparisons of the different computational approaches in combination with different types of learning networks.

9.7 Manipulator Control Experiment by Feedback Error Learning

There are two possibilities regarding how the central nervous system computes nonlinear transformations required for an inverse dynamics model of a nonlinear controlled object. One is that they are computed by nonlinear information processing within the dendrites of neurons (Kawato et al. 1984; Kawato, Furukawa, and Suzuki 1987; Miyamoto et al. 1988). The other is that they are realized by neural circuits, and are acquired by motor learning (Kawato, Setoyama, and Suzuki 1988).

Examining the first possibility, we have successfully applied the feedback error learning neural network to trajectory control of an industrial robotic manipulator (Kawasaki-Unimate PUMA 260) with prepared nonlinear transformations derived from a dynamics equation of a manipulator-idealized mechanical model (Miyamoto et al. 1988). A simple training movement pattern lasting for 6s was elicited three hundred times. Both the error of the trajectory and the feedback torque decreased dramatically during thirty minutes of learning. Moreover, the effect of learning for faster and quite different movement patterns from the training pattern was noted; that is, the network showed the ability of performing learning generalization.

Regarding the second possibility, we succeeded in obtaining learning control of the robotic manipulator by an inverse dynamics model formed by a three-layer neural network shown in figure 9.8 (Kawato, Setoyama, and Suzuki 1988). In this network, nonlinear transformation was made only of a cascade of linear-weighted summation and sigmoid nonlinearity. That is, we did not use any a priori knowledge about the dynamical structure of the controlled object. In this particular task, we continued to use the torque command as the error signal.

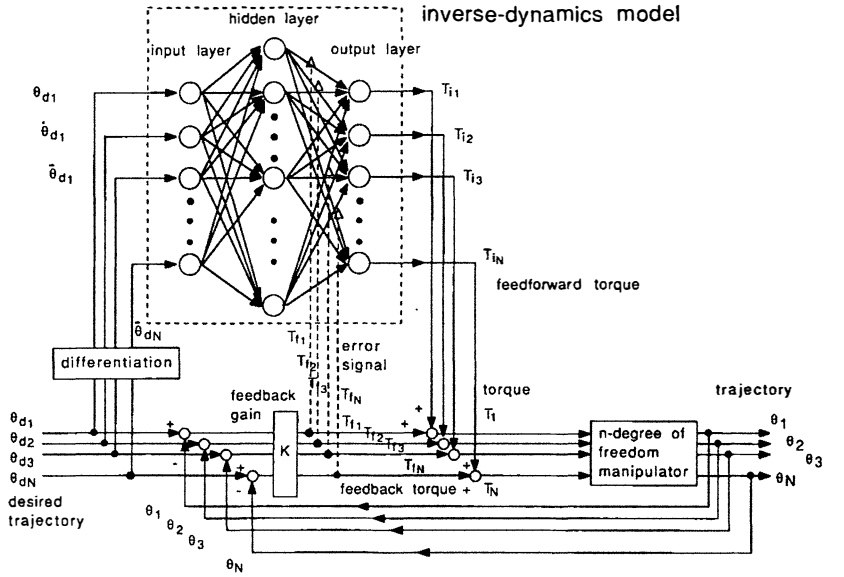


Figure 9.8
A feedback error learning neural network model. The inverse dynamics model is acquired in the three-layer neural network.

Neurons in the input layer represent desired joint angles, angular velocities, and angular accelerations. The input-output transformation function in the input layer is assumed linear. The neurons in the output layer represent the feedforward motor command for different actuators. Let $x_j^I, y_j^I, x_k^H, y_k^H, x_m^O, y_m^O, w_{jk}^{IH}, w_{km}^{HO}$ represent the membrane potential and the output of the j-th neuron in the input layer, membrane potential and the output of the k-th neuron in the hidden layer, the membrane potential and the output of the m-th neuron in the output layer, the synaptic weight from the j-th neuron in the input layer to the k-th neuron in the hidden layer, and the synaptic weight from the k-th neuron in the hidden layer to the m-th neuron in the output layer, respectively. The following equations hold for inputs and outputs of the neurons.

$$y_j^I = x_j^I \tag{9.12}$$

$$x_k^H = \sum_j w_{jk}^{IH} y_j^I \tag{9.13}$$

$$y_k^H = f(x_k^H) \tag{9.14}$$

$$x_m^O = \sum_k w_{km}^{HO} y_k^H \tag{9.15}$$

$$y_m^O = f(x_m^O) \tag{9.16}$$

$$T_{im} = y_m^O, \tag{9.17}$$

Here, f is the sigmoid function. The synaptic weight between the hidden layer and the output layer is modified according to the following heterosynaptic learning rule while using the feedback motor command T_{fm} for the m-th degree of freedom as the error signal.

$$\tau dw_{km}^{HO}/dt = y_k^H \dot{f}(x_m^O) T_{fm}, \tag{9.18}$$

here, τ is the time constant of the synaptic modification. The error signal of the k-th neuron in the hidden layer is calculated as the weighted summation of the responsible feedback motor commands by the synaptic weights:

$$\delta_k^H = \sum_m w_{km}^{HO} \dot{f}(x_m^O) T_{fm}. \tag{9.19}$$

The rate of change of the synaptic weight from the input layer to the hidden layer is again given by the product of the input to the hidden-layer neuron and the error signal.

$$\tau dw_{jk}^{IH}/dt = y_j^I \dot{f}(x_k^H) \delta_k^H. \tag{9.20}$$

The motor command fed to the m-th degree of freedom of the controlled object is the sum of the feedforward torque and the feedback torque.

$$T_m = T_{im} + T_{fm}. \tag{9.21}$$

In this study, the learning task proceeded smoothly and the network showed the capability of some degree of generalization as shown in figure 9.9 (Setoyama, Kawato, and Suzuki 1988). Figure 9.9 shows mean square errors of the three joint angles for the two training patterns represented by the two solid curves, and errors for a twofold faster test movement pattern shown by the one point chain curve, and a slower test pattern

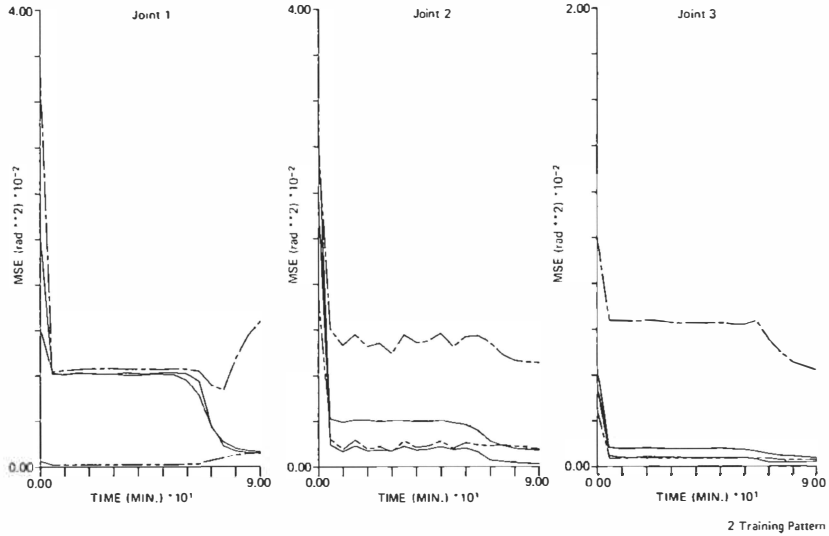


Figure 9.9
Change of the mean square error of the three joint angles during 90 minutes learning. Cited from Setoyama, Kawato, and Suzuki (1988).

shown by the two-points chain curve. The test movement patterns were not used in the training. The network performance for the test patterns was examined while fixing the synaptic weights at some time in the learning duration.

We recently obtained results that information representation of hidden units properly corresponds to the inertia force, the viscous friction force, Coulomb friction force, and the gravitational force. However, the Coriolis forces and the centripetal forces are widely represented in the many hidden units and it is not easy to analyze their representations. It is one of our future tasks. We now know that three-layer networks with sufficient number of hidden units with the steepest descent method combined with simulated annealing can learn arbitrary nonlinear functions. However, in the future, we must examine whether this statement has any practical implications in real world applications.

References

Abend, W., Bizzi, E. and Morasso, P. (1982). Human arm trajectory formation. *Brain* **105**:331–348.

Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control* 97:220–227.

Allen, G. I., and Tsukahara, N. (1974). Cerebrocerebellar communication systems. *Physiological Reviews* **54**:957–1006.

Atkeson, C. G., and Hollerbach, J. M. (1985). Kinematic features of unrestrained vertical arm movements. *Journal of Neuroscience* **5**:2318–2330.

Atkeson, C. G., and Reinkensmeyer, D. J. (1988). Using associative content-addressable memories to control robots. In *Proceedings of the IEEE Conference on Decision and Control,* 792–797. Austin, TX, Dec. 7–9.

Flash, T. and Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience* **5**:1688–1703.

Fujita, M. (1982). Adaptive filter model of the cerebellum. *Biological Cybernetics* **45**:195–206.

Isobe, M., Kawato, M., and Suzuki, S. (1988). Iterative learning control of industrial manipulator in joint-angular and visual coordinates. Japan IEICE Technical Report. **MBE87–134,** 241–248.

Ito, M. (1970). Neurophysiological aspects of the cerebellar motor control system, *International Journal of Neurology* **7**:162–176.

Jordan, M. I. (1988). Supervised learning and systems with excess degrees of freedom, COINS Technical Report 88-27, 1-41, Department of Computer Science and Information, University of Massachusetts, Amherst.

Jordan, M. I. (1989). Indeterminate motor skill learning problems. In M. Jeannerod ed., *Attention and performance XIII.* Hillsdale, NJ: Lawrence Erlbaum.

Jordan, M. I. and Rosenbaum, D. A. (1988). Action COINS Technical Report 88-26, 1-68, Department of Computer Science and Information, University of Massachusetts, Amherst.

Kano, M., Kawato, M., and Suzuki, R. (1989). Acquisition of inverse-kinematics and inverse-dynamics model of a redundant arm by the feedback error learning. Japan IEICE Technical Report **MBE88-171**, 91–96.

Kawato, M. (1989). Adaptation and learning in control of voluntary movement by the central nervous system. *Advanced Robotics* **3**(3):000–000.

Kawato, M., Furukawa, K., and Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement, *Biological Cybernetics* **57**:169–185.

Kawato, M., Hamaguchi, T., Murakami, F. and Tsukahara, N. (1984). Quantitative analysis of electrical properties of dendritic spines, *Biological Cybernetics* **50**:447–454.

Kawato, M., Isobe, M., Maeda, Y., and Suzuki, R. (1988). Coordinates transformation and learning control for visually-guided voluntary movement with iteration: A Newton-like method in a function space, *Biological Cybernetics* **59**:161–177.

Kawato, M., Isobe, M., and Suzuki, R. (1988). Hierarchical learning of voluntary movement by cerebellum and sensory association cortex. In M. A. Arbib and S. Amari eds., *Dynamic interaction in neural networks: Models and data,* 195–214, New York: Springer-Verlag.

Kawato, M., Setoyama, T. and Suzuki, R. (1988). Feedback error learning of movement by multi-layer neural network. In *Proceedings of the International Neural Networks Society First Annual Meeting,* 342.

Kawato, M., Uno, Y., Isobe, M., and Suzuki, R. (1988). A hierarchical neural network model for voluntary movement with application to robotics, *IEEE Control Systems Magazine* **8**:8–16.

Kuperstein, M. (1988). Neural model of adaptive hand-eye coordination for single postures. *Science* **239**:1308–1311.

Maeda, Y., Kawato, M., Uno, Y., and Suzuki, R. (1989) Trajectory formation for human multi-joint arm by a cascade neural network model. Japan IEICE Technical Report **MBE88-169**, 79–84.

Marr, D. (1982). *Vision.* New York: W. H. Freeman.

Miller, W. T. (1987). Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Journal of Robotics and Automation* **RA-3**:157–165.

Miller, W. T., Glanz, F. H., and Kraft, L. G. (1987). Application of a general learning algorithm to the control of robotic manipulators. *International Journal of Robotics Research* **6**:84–98.

Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R. (1988). Feedback-error-learning neural network for trajectory control of a robotic manipulator, *Neural Networks* **1**:251–265.

Morasso, P. (1981). Spatial control of arm movements. *Experimental Brain Research* **42**:223–227.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323**:533–536.

Setoyama, T., Kawato, M., and Suzuki, R. (1988). Manipulator control by inverse-dynamics model learned in multi-layer neural network. Japan IEICE Technical Report **MBE87-135**, 249–256.

Tsukahara, N. and Kawato, M. (1982) Dynamic and plastic properties of the brain stem neuronal networks as the possible neuronal basis of learning and memory. In S. Amari and M. A. Arbib eds., *Lecture Notes in Biomathematics.* Vol. 45, *Competition and Cooperation in Neural Nets,* 430–441. New York: Springer-Verlag.

Uno, Y., Kawato, M., and Suzuki, R. (1987). Formation of optimum trajectory in control of arm movement: minimum torque-change model. Japan IEICE Technical Report **MBE86-79**, 9–16.

Uno, Y., Kawato, M., and Suzuki, R. (1989). Formation and control of optimal trajectory in human multijoint arm movement: minimum torque-change model. *Biological Cybernetics* 61:89–101.