TR－A－0056

Trajectory Formation of Arm Movement by Cascade Neural Network Model Based on Minimum Torque-change Criterion

Mitsuo KAWATO, Yoshiharu MAEDA,*
Yoji UNO† and Ryoji SUZUKI†

**Osaka University* *†University of Tokyo*

1989. 7. 26.

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町乾谷 ☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408
Telex: 5452-516 ATR J

Trajectory Formation of Arm Movement by Cascade Neural Network Model Based on Minimum Torque-change Criterion

Mitsuo Kawato[1]
Cognitive Processes Department
ATR Auditory and Visual Perception Research Laboratories, Kyoto

Yoshiharu Maeda[2]
Department of Biophysical Engineering
Faculty of Engineering Science, Osaka University, Osaka

Yoji Uno and Ryoji Suzuki
Department of Mathematical Engineering and Information Physics
Faculty of Engineering, University of Tokyo, Tokyo

[1]to whom correspondence should be addressed: ATR Auditory and Visual Perception Research Laboratories, Sanpeidani, Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02, Japan, Telephone (Direct) 07749-5-1452

[2]present address: International Institute for Advanced Study of Social Information Science, Fujitsu Limited, Numazu, Japan

Abstract

We proposed that the trajectory followed by human subject arms tended to minimize the time integral of the square of the rate of change of torque (Uno, Kawato, Suzuki, 1987). This minimum torque-change model predicted and reproduced human multi-joint movement data quite well (Uno, Kawato, Suzuki, 1989). Here, we propose a neural network model for trajectory formation based on the minimum torque-change criterion. Basic ideas of information representation and algorithm are (i) spatial representation of time, (ii) learning of forward dynamics and kinematics model and (iii) relaxation computation based on the acquired model. Operations of this network are divided into the learning phase and the pattern-generating phase. In the learning phase, this network acquires a forward model of the multi-degrees-of-freedom controlled object while monitoring the actual trajectory as a teaching signal. In particular, it learns a vector field of the ordinary differential equation which describes the dynamics of the controlled object. Correspondingly, the network structure is a cascade of many identical network units, each of which approximates the vector field. In the pattern-generating phase, electrical coupling between neurons representing motor commands at neighboring times is activated to guarantee the minimum torque-change criterion. The network changes its state autonomously by forward calculation through the cascade structure, and by error back-propagation based on the acquired model. At the stable equilibrium state with minimum energy, the network outputs the torque which realizes the minimum torque-change trajectory. The model can resolve ill-posed inverse kinematics and inverse dynamics problems for redundant controlled objects as well as ill-posed trajectory formation problems. By computer simulation, we show that the model can produce a multi-joint arm trajectory while avoiding obstacles or passing through via-points.

1. Introduction

Based on the pioneering work by Saltzman (1979), we (Kawato, Furukawa, Suzuki, 1987) proposed a computational model for control of voluntary movement which accounts for Marr's (Marr, 1982) first level of understanding complex information-processing systems: i.e., computational theory. In this model, three sets of information are assumed to be internally represented in the brain, that is, a desirable trajectory in task-oriented coordinates, a desirable trajectory in body coordinates and a motor command. Computations which derive these three sets of information are called a trajectory formation problem, a coordinate transformation problem and a motor command generation problem, respectively. The second and the third problems are called the inverse kinematics problem and the inverse dynamics problem in robotics literature.

As explained in Kawato et al. (1987) and Kawato (1989), several lines of experimental evidence suggest that the three sets of information are internally represented in the brain. For example, Bizzi, Accornero, Chapple and Hogan (1984) reported experiment results which indicate that the desired trajectory is explicitly planned in the brain. When the forearm of a deafferented monkey was quickly forced to the target position early in the movement, the arm returned to some intermediate point between the initial and final target positions, then gradually approached the final position again. A trajectory which connects the above intermediate points for various times of perturbation can be regarded as the desired, planned trajectory.

In this paper, we propose a neural network model which solves the three problems (trajectory formation, coordinate transformation and generation of motor command) at the same time. An input to the network is a goal of movement, such as a desired end point, a desired via-point, and locations of obstacles to be avoided , which are expressed in task-oriented coordinates, and a movement time. The output from the network is a motor command. This network can be regarded as one example of endogenous motor pattern generators such as a neural oscillator for rhythmic movements. The network internally estimates the desired trajectory in the task-oriented coordinates as well as the desired trajectory in the body coordinates. However, they are not essential in feedforward motor control. Consequently, a computational scheme adopted in this neural network model obtains the motor command directly from the goal of movement. In our previous papers (Kawato et al., 1987; Kawato, Isobe, Maeda, Suzuki, 1988), we proposed step-by-step computational schemes, where the three computational problems are solved step by step, as well as the above direct computational scheme. The model proposed here is assumed to be used for very skilled movements, while the step-by-step computations are utilized for relatively difficult or less skilled movements. This point will be discussed later in connection with behavioral observations of motor learning in human arm movements.

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 Fig. 1..1. First, consider the trajectory determination problem for a planar, two-joint arm movement within a plane, when the starting point, the via-point and the end point, as well as the movement time, are specified (Fig. 1..1, top). There are 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-degrees-of-freedom manipulator in a plane (Fig. 1..1, 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 can not uniquely be determined. We note that human arms have excess degrees of freedom.

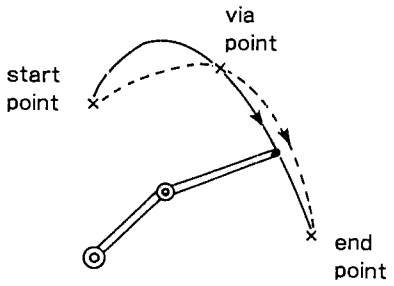
The third ill-posed motor control problem is the inverse dynamics problem in a manipulator with agonist and antagonist muscles (actuators). Consider a single joint manipulator with a pair of muscles (Fig. 1..1, bottom). The inverse dynamics problem is to determine the time courses of agonist and antagonist muscle tensions 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 Fig. 1..1, bottom.

Jordan (1988) clearly explained why the widely studied direct inverse modeling neural network approach (Albus, 1975; Miller, 1987; Miller, Glanz, Kraft, 1987; Kuperstein, 1988; Atkeson, Reinkensmeyer, 1988) is inadequate for resolving ill-posed inverse kinematics problems.

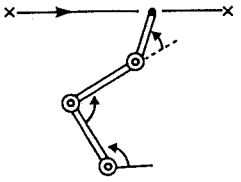
There are two different approaches which resolve these ill-posed problems. One approach is to utilize a feedback controller. The feedback controller selects one specific motor command in the inverse dynamics and inverse kinematics problems even for redundant manipulators. But the desired trajectory can not exactly be realized by feedback control alone. The feedback error learning approach proposed in Kawato et al. (1987) can resolve ill-posed inverse kinematics and inverse dynamics problems 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 expressed in Cartesian coordinates for a three-link manipulator within a plane (Kano, Kawato, Suzuki, 1989). The selected joint angle time courses depended on the initial posture of the manipulator and the gains of the feedback controller. The kinematic net proposed by Mussa-Ivaldi, Morasso and Zaccaria (1988) is essentially the feedback control approach and deals with the inverse kinematics (static) problem.

Another approach is to introduce a smoothness performance index. Here, we briefly explain two experimentally confirmed objective functions for voluntary movements. The most marked and beautiful experiment features of human multi-joint

Trajectory Formation



Inverse Kinematics in Redundant Manipulator



Inverse Dynamics in Redundant Manipulator

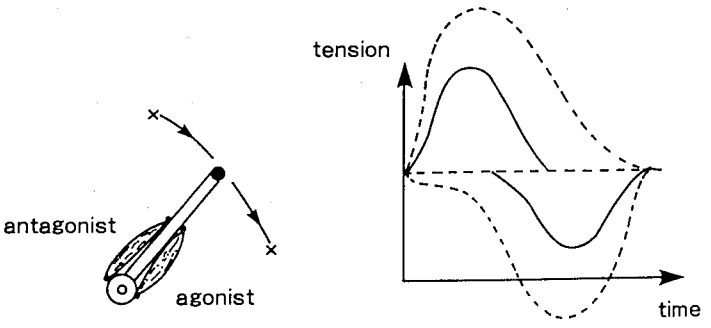


Figure 1..1: Three ill-posed problems in sensory-motor control.

arm movements between two points are roughly straight hand paths and bell-shaped hand tangential speed profiles (Morasso, 1981; Abend, Bizzi, Morasso, 1982; Atkeson, Hollerbach, 1985; Flash, Hogan, 1985; Uno, Kawato, Suzuki, 1989). In order to account for such kinematic features of human multi-joint arm movements, 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.

$$C_J = 1/2 \int_0^{t_f} \{(\frac{d^3X}{dt^3})^2 + (\frac{d^3Y}{dt^3})^2\} dt. \tag{1.1}$$

Here, (X, Y) are Cartesian coordinates of the hand, and t_f is the movement duration. Flash and Hogan showed that the unique trajectory which yielded the best performance was in good agreement with the experiment data on movement within 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 extracorporeal 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:

$$C_T = 1/2 \int_0^{t_f} \sum_{i=1}^{m} (\frac{d\tau^i}{dt})^2 dt, \tag{1.2}$$

here, τ^i is the torque fed to the ith of m 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.

For the movements between pairs of targets just in front of the body, predictions of both the models were in good agreement with the experiment data. However, the trajectories predicted by the minimum torque-change model were quite different from the minimum jerk model in four behavioral situations. In one situation, past experiment data support the minimum torque-change model (see below). The rest three of the situations were not examined in past experiments. Uno et al. (1989) in recent experiments examined human planar arm movement in three different situations and found that the minimum torque-change model predicted the real data better than the minimum jerk model.

First, when the starting point is an arm outstretched to the side and the end point is in front of the body, the path was curved in the minimum torque-change model, but always straight in the minimum jerk model. The hand paths of 16 human subjects were all curved.

Second, consider movements between two points while resisting a spring, one end of which is attached to the hand while the other is fixed. The minimum jerk

model always predicts a straight path regardless of the external forces. On the other hand, the minimum torque-change model predicted a curved trajectory and an asymmetrical speed profile for the movement with the spring. These predictions are in close agreement with experiment data.

Third, we examined vertical movements which are affected by gravity. The minimum jerk model always predicts a straight path between two points. On the other hand, the minimum torque-change model predicted curved paths for large up and down movements, roughly straight paths for small fore and aft movements. The speed profiles were bell-shaped for both movements. These predictions are in close agreement with experiment data of Atkeson and Hollerbach (1985).

Finally, the most compelling evidence is about a pair of via-point movements. Consider two subcases, with identical start and end points, but with dictated mirror-image via-points. If one notices invariance of the objective function C_J of the minimum jerk model under translation, rotation and rolling, it is easy to see that the minimum jerk model predicts identical paths with respect to rolling as well as identical speed profiles for the two subcases. On the other hand, the minimum torque-change model predicts two different trajectories. For the concave path, the speed profile has two peaks. However, for the convex path, the speed profile has only one peak. These predictions are in close agreement with the human data (Uno, Kawato, Suzuki, 1989).

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

The minimum jerk model formulated in the task-oriented coordinates can not resolve the ill-posed inverse kinematics and inverse dynamics problems for redundant manipulators. The feedback control approach can not resolve the ill-posed trajectory formation problem in spite of the early hypothesis of the *end point control* (see Bizzi et al., 1984). However, the minimum torque-change model can resolve all three ill-posed problems shown in Fig. 1..1 at the same time when the locations of the desired end point, desired via-points and obstacles are given in task-oriented coordinates.

If we adopt the minimum torque-change model as a computational scheme, it leads to two important conceptual assumptions. First, the brain needs to acquire, by training, an internal model of the dynamics and kinematics of the controlled object and continuously utilize it for trajectory formation. Second, the brain must solve the three problems simultaneously.

The purpose of this paper is to propose a neural network model which, based on the minimum torque-change criterion, coherently resolves all three ill-posed problems shown in Fig. 1..1. The minimum torque-change criterion is embedded as hardware (electrical synapses) in the neural network model. The neural network model first acquires a forward dynamics model of a controlled object by training and then cal-

culates the motor command by relaxation computation utilizing the learned forward dynamics model. In a sense, the network first learns the energy to be minimized, and then minimizes the learned energy.

The minimum torque-change model is (i) a computational model for the trajectory formation problem. The neural network model proposed in this paper provides understanding on the (ii) representation and algorithm level, and (iii) hardware level, for the same problem. We believe that the most important proposition in Marr (1982) is the need to connect understanding at three levels, rather than to stick to one of the three levels, even the computational level.

It is possible for two different neural network structures to acquire the forward dynamics and kinematics model of a controlled object. Previously, we proposed a four-layer model, which learns the flow of a dynamical system describing the controlled object (Maeda, Kawato, Uno, Suzuki, 1988; Kawato, Uno, Isobe, Suzuki, 1988; Uno, Kawato, Maeda, Suzuki, 1988). In this paper, we expand the previous model and propose a model which learns the vector field of the dynamical system instead of the flow. Accordingly, the network has a cascade structure made up of many repetitions of a network unit which approximates the vector field. We show computer simulations of trajectory formation while passing through via-points or avoiding obstacles. We will discuss possibilities of using the proposed neural network model for continuous speech recognition, as was first hypothesized in "motor theory of speech perception" (Liberman, Cooper, Shankweiler, Studdert-Kennedy, 1967).

2. Information representation and algorithms

In this section we explain information representation and basic algorithms of our neural network model.

Since the dynamics of the human arm or a robotic manipulator is nonlinear, finding the unique trajectory which minimizes C_T is a nonlinear optimization problem. Accordingly, it is much more difficult to determine the unique trajectory which minimizes C_T than to find the minimum jerk trajectory. Uno et al. (1987) overcame this difficulty by developing an iterative scheme, so that the unique trajectory and the associated motor command (torque) could be determined simultaneously. Mathematically, the iterative learning scheme can be regarded as a Newton-like method in a function space. The central nervous system does not seem to adopt this algorithm in trajectory formation.

It was reported that some neural network models can solve computationally difficult problems such as the traveling salesman problem (Hopfield and Tank, 1985) or early visions (Poggio, Torre, Koch, 1985; Koch, Marroquin, Yuille, 1986). These computational problems can be regarded as nonlinear optimization problems with some constraints. The neural network models solve these problems by minimizing specific cost functions (energy) through dynamical changes of their state variables. Because of the success of the minimum torque-change model, the problem of trajectory formation can also be regarded as a nonlinear optimization problem with a constraint given as the nonlinear dynamics of the controlled object.

Let us mathematically formulate the trajectory formation problem on the minimum torque-change criterion. First, forward dynamics and inverse dynamics problems are defined. θ denotes an n-dimensional vector which represents the body coordinates, such as joint angles or muscle lengths, of a controlled object. $\dot{\theta}$ represents the corresponding velocity vector. τ represents an m-dimensional vector of motor commands such as joint torques or muscle tensions. The state change of the controlled object is described by the following ordinary differential equations.

$$\begin{aligned} d\theta/dt &= \dot{\theta} \\ d\dot{\theta}/dt &= f(\theta, \dot{\theta}, \tau), \end{aligned} \tag{2.1}$$

here f is an n-dimensional nonlinear vector function. The forward dynamics problem is to find the body space trajectory $(\theta(t), \dot{\theta}(t))$ when the motor command $\tau(t)$ is given. Conversely, the inverse dynamics problem is to find the motor command $\tau(t)$ which realizes a given trajectory $(\theta(t), \dot{\theta}(t))$. The forward dynamics can be conveniently expressed by the following flow $\psi : R \times R^{2n} \times R^{m^R} \longrightarrow R^{2n}$ of the dynamical system 2.1:

$$(\theta(t), \dot{\theta}(t)) = \psi(t; (\theta(0), \dot{\theta}(0)); \tau(\cdot)), \tag{2.2}$$

here $\tau(\cdot)$ represents the time course of the motor command.

Second, the forward kinematics and inverse kinematics problems are defined. x denotes a k-dimensional vector representing the task-oriented coordinates of the controlled object, for example, the Cartesian coordinates of the hand position or the retinal coordinates of a grasped object. x is uniquely determined from θ according to the following nonlinear equation:

$$x = G(\theta), \tag{2.3}$$

here G is a k-dimensional nonlinear vector function. The forward kinematics is to determine x from θ based on the above equation. The inverse kinematics is to compute θ from x. As explained in Fig. 1..1, if $n > k$, then G^{-1} does not generally exist since there are many θs which lead to the same x.

Two kinds of constraints are imposed on the movement. The first constraint which must be rigorously satisfied is called a hard constraint. The second constraint which requires smoothness of the movement with the lower priority is called a soft or smoothness constraint. Hard constraints for the movement are given as follows:

$$x(0) = x_0 \tag{2.4}$$

$$\dot{x}(0) = v_0 \tag{2.5}$$

$$x(t_f) = x_f \tag{2.6}$$

$$\dot{x}(t_f) = v_f \tag{2.7}$$

$$x(t_{via}) = x_{via} \quad ; 0 < t_{via} < t_f \tag{2.8}$$

$$x(t) \notin \Omega \subset R^k \quad ; 0 \leq t \leq t_f. \tag{2.9}$$

The first two equations specify the initial conditions at the starting point. The third and fourth equations specify the terminal conditions at the end of the movement,

t_f. The fifth equation is the condition of passing through the via-point. The time t_{via} when the via-point must be passed may or may not be specified. The sixth equation requires avoidance of obstacles whose locations are given as a subset Ω in the task state space R^k. The first four conditions are always given, but the last two conditions are not always given. Please note that all of these conditions are given in task-oriented coordinates. However, for a redundant controlled object it is sometimes more convenient to express the fourth condition in body coordinates as we will discuss later.

Definition 1 *The trajectory formation problem based on the minimum torque change criterion is to find $\tau(t)$ for $0 \leq t \leq t_f$ which minimizes the objective function C_T 1.2 among those which satisfy the dynamics equation 2.1, the kinematics equation 2.3 and the movement conditions 2.6, 2.7, 2.8, 2.9.*

Basic ideas of information representation and an algorithm of our neural network model are (i) spatial representation of time, (ii) learning of the forward dynamics and kinematics model and (iii) relaxation computation based on the acquired model. These three ideas are schematically illustrated in a 5-layer neural network model of Fig. 2..1. The motor command $\tau(t)$, the body space trajectory $(\theta(t), \dot{\theta}(t))$, and the task space trajectory $(x(t), \dot{x}(t))$ at different times $j\Delta t$ with a fixed time step Δt are distributed throughout the network. So, for example, a single neuron is allocated to represent the ith component of the motor command $\tau_j^i = \tau^i(j\Delta t)$ at time $j\Delta t$. The movement time is divided into N steps ($t_f = N\Delta t$) and $t_{via} = L\Delta t$. In other words, time is represented spatially. Consequently, a delay line is required to transmit the spatially represented motor command time course to a controlled object.

In the learning phase, the neural network acquires a forward dynamics and kinematics model of the controlled object, within a multi-layer feedforward (MLFF) structure. The MLFF structure consists of an input layer (the first layer in Fig. 2..1) which represents the motor command, an intermediate layer (the third layer in Fig. 2..1) which represents the body space trajectory, and an output layer (the fifth layer in Fig. 2..1) which represents the task space trajectory, and other hidden layers (the second and fourth layers in Fig. 2..1). Between the motor command layer and the body space layer, an internal model of the forward dynamics ψ, eq. 2.2 is acquired by training. Similarly, between the body space layer and the task space layer, an internal model of the forward kinematics G, eq. 2.3 is acquired by training. The hidden layers provide necessary nonlinear transformations for approximating nonlinear forward dynamics and forward kinematics. The network as a whole provides a nonlinear mapping F from the motor command τ to the task space trajectory $(x, \dot{x})$. F is a composite function $G \cdot \psi$.

One of the most attractive features of the MLFF structure is that the network can calculate the partial derivative $\partial(x, \dot{x})/\partial\tau = \partial F(\tau)/\partial\tau$ in parallel using learned synaptic weights based on the error back-propagation algorithm (Rumelhart, Hinton, Williams, 1986; Werbos, 1988), once it acquires the mapping F. In the pattern generating phase, the network performs relaxation computation utilizing this partial

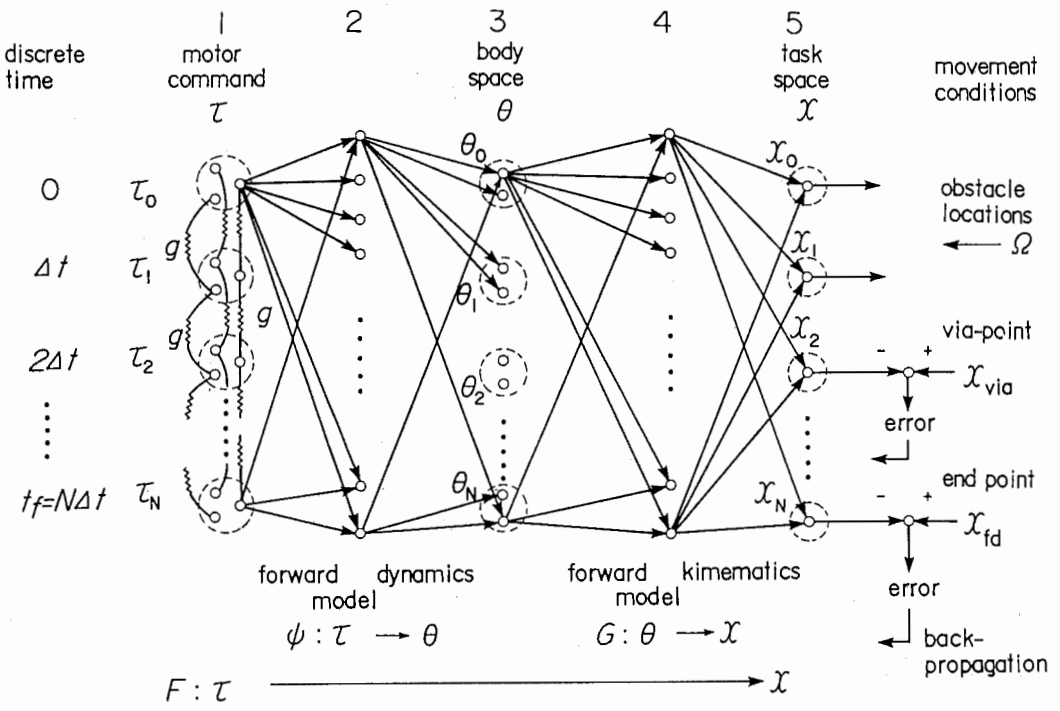


Figure 2..1: A schematic diagram of a 5-layer neural network model for illustration of basic ideas of information representations and algorithms. In this figure, τ_j, θ_j and x_j represent the motor command, the body space trajectory, and the task space trajectory at the jth time.

derivative. Biologically more plausible learning algorithms, such as the associative reward-penalty learning of Barto and Anandan (1985), could be used to attain the same task.

A total energy E of the network is defined as a summation of a hard constraint energy E_D and a smoothness constraint energy E_S multiplied by a regularization parameter λ.

$$E = E_D + \lambda E_S. \tag{2.10}$$

The smoothness constraint term is simply a discrete version of the minimum torque-change criterion 1.2:

$$E_S = 1/2 \sum_{j=1}^{N} \sum_{i=1}^{m} (\tau_{j-1}^i - \tau_j^i)^2 \tag{2.11}$$

The hard constraint energy is a summation of energy terms which correspond to the movement conditions 2.6, 2.7, 2.8, 2.9.

$$\begin{aligned}
E_D &= E_f + E_{vf} + E_{via} + \mu E_{obst} && (2.12)\\
E_f &= 1/2(x_f - x_N)^T(x_f - x_N) && (2.13)\\
E_{vf} &= 1/2(v_f - \dot{x}_N)^T(v_f - \dot{x}_N) && (2.14)\\
E_{via} &= 1/2(x_{via} - x_L)^T(x_{via} - x_L) && (2.15)\\
E_{obst} &= \sum_{j=1}^{N} H(x_j; \Omega). && (2.16)
\end{aligned}$$

Here, $(x_j, \dot{x}_j)$ is a task space trajectory at time $j\Delta t$ estimated by the neural network model from the learned mapping F and a time history of τ. If the time the via-point must be passed is not specified, the time $L\Delta t$ is chosen so that E_{via} is as its minimum (i.e. L is such that $\forall j, \quad \mid x_{via} - x_L \mid \leq \mid x_{via} - x_j \mid$). H is a non-negative function, which vanishes when $x_j \notin \Omega$. H is differentiable in Ω. μ is a positive, finite sharpening parameter. For fast relaxation, μ is first set =0 at the beginning of relaxation and slowly increased to 1 during relaxation.

Relaxation of the motor command $\tau(t)$ obeys the following equations. We introduce a relaxation time s which is independent of the movement time t. So, $\tau_j^i = \tau^i(j\Delta t)$, which represents the ith component of the motor command at the jth time, changes with the relaxation time s. Here, $i = 1, 2, \cdots, m; j = 0, 1, \cdots, N-1, N$.

$$\begin{aligned}
\tau_0 &= \tau_{init} && (2.17)\\
T \cdot d\tau_j^i/ds &= -\partial E/\partial \tau_j^i = -\partial E_D/\partial \tau_j^i - \lambda \partial E_S/\partial \tau_j^i\\
&= (x_f - x_N)^T \partial x_N/\partial \tau_j^i + (v_f - \dot{x}_N)^T \partial \dot{x}_N/\partial \tau_j^i + (x_{via} - x_L)^T \partial x_L/\partial \tau_j^i\\
&\quad -\mu \sum_{l=1}^{N} \partial H(x_l; \Omega)/\partial x_l \cdot \partial x_l/\partial \tau_j^i + \lambda(\tau_{j-1}^i - 2\tau_j^i + \tau_{j+1}^i) && (2.18)\\
&\quad j = 1, 2, \cdots, N-1\\
\tau_N^i &= \tau_{fin}, && (2.19)
\end{aligned}$$

here τ_{init} and τ_{fin} are necessary motor commands to hold the starting and final postures. If gravitational or muscle forces are compensated for beforehand, these two boundary conditions are zero. T is a time constant of change of τ.

The first four terms of the relaxation equation 2.18 contain the partial derivative $\partial(x, \dot{x})/\partial\tau = \partial F(\tau)/\partial\tau$. As mentioned earlier, they can be calculated by back-propagation methods. Actually, if we regard x_f, v_f and x_{via} as teaching signals to the estimated values of x_N, $\dot{x}_N$ and x_L, respectively, $(x_f - x_N)$, $(v_f - \dot{x}_N)$ and $(x_{via} - x_L)$ equal the corresponding error signals at the output layer (see Fig. 2..1). Then, it is straightforward to show that the first three terms of 2.18 are exactly the error signals at the input layer, which are calculated backward from the output layer. This also applies to the fourth term of 2.18 if we introduce an appropriate error signal for obstacle avoidance (see section 5.3 for detail). Consequently, we can rewrite equation 2.18 using error signals δ_j^i back propagated to the ith component motor command neuron at the jth time:

$$T \cdot d\tau_j^i(s)/ds = \delta_j^i(s) + \lambda(\tau_{j-1}^i(s) - 2\tau_j^i(s) + \tau_{j+1}^i(s)). \tag{2.20}$$

Here, we note that $\delta_j^i(s) = -\partial E_D/\partial\tau_j^i(s)$ and $\delta_j^i(s)$ is a nonlinear function of $\tau_j^i(s)$. If τ at some relaxation time s is determined, $(\theta, \dot{\theta})$ and x are determined by feedforward calculation of the network. Then the error signals at the output layers $(x_f - x_N(s))$, $(v_f - \dot{x}_N(s))$ and $(x_{via} - x_L(s))$ at that instant s are calculated. Finally, the error signals at the input layer $\delta_j^i(s)$ at time s is calculated by backward propagation of errors. Then the motor command changes its state according to the relaxation equation 2.20. The variables other than the motor commands (i.e. body space trajectory, task space trajectory, error signals at the output layer, error signals at the input layer) can be regarded as dependent variables of the motor commands, from the viewpoint of the dynamical system 2.20 and the relaxation time s. That is, they are instantaneously determined from the motor commands.

The second term of the relaxation equation 2.20 are implemented as electrical synapses or gap junctions between motor command neurons at the neighboring times (see electrical resistances in Fig. 2..1). Then we can identify the regularization parameter λ as electrical conductance g of gap junctions. g is slowly decreased to zero during relaxation. This is just like the "penalty method" in optimal control theory. In this operation, we treat the first term of eq. 2.10 as a hard constraint and the second term as a soft constraint. In other words, we seek the solution with the minimum torque change from those which strictly satisfy the movement conditions.

Because E is a Ljapunov function of the dynamical system 2.18, it always decreases. The stable equilibrium state of the neural network model corresponds to the (local) minimum "energy" state, and hence the network attains the motor command which satisfies the hard constraint ($E_D = 0$) and with a minimum E_S. The network first acquires E_D by training, and then minimizes the learned energy $E = E_D + \lambda E_S$. The smoothness energy E_S is embedded as a hardware structure of the network.

3. Repetitively structured, time invariant cascade neural network model for vector field of dynamics

Previously, we proposed a four-layer MLFF model, which first learns the flow of a dynamical system describing the controlled object, and then generates the minimum torque-change trajectory (Maeda, Kawato, Uno, Suzuki, 1988; Kawato, Uno, Isobe, Suzuki, 1988; Uno, Kawato, Maeda, Suzuki, 1988). By computer simulation, we showed that the proposed model can generate a fairly good minimum torque-change trajectory. This simple model does not *a priori* possess intrinsic properties of the flow such as (i) continuity of the solution of the dynamical system with respect to time and initial conditions, (ii) group property of the flow ψ: $\psi(t+s;x) = \psi(t;\psi(s;x))$, (iii) causality: trajectory at a given time does not depend on the torque input after that time. We found that the network can acquire the third property by only training from examples (Maeda et al., 1988). But the first two properties were not acquired. The most serious shortcoming of this simple model is its size. If the time step Δt is fixed and movement time becomes long, then the network size becomes enormous. Especially, the number of synaptic weights which must be learned grows in polynomial order of the movement time.

Hence, in this paper we propose a neural network model which inherently possesses the above three properties of the flow. The number of the synaptic weights to be learned in this model is constant regardless of the movement time. It is a repetitively structured, time invariant, cascade neural network model (Fig. 3..1). This is a natural extension of our previous model. It learns the vector field of the ordinary differential equation which describes the forward dynamics of the controlled object instead of the flow. We note that general mathematical arguments developed in the previous section all remain applicable to the cascade model, since the model still has a multi-layer feedforward structure regardless of its complicated appearance at first sight.

For simplicity, we explain the cascade model in the case of a single-degree-of-freedom controlled object without forward or inverse kinematics (see Fig. 3..1). Extension to the multi-degrees-of-freedom case with the ill-posed inverse dynamics and kinematics problems is straightforward, and will be given in the next section. Let us assume that the controlled object is described by the following equations.

$$d\theta/dt = \dot{\theta} \tag{3.1}$$

$$d\dot{\theta}/dt = h(\theta, \dot{\theta}, \tau), \tag{3.2}$$

here, only in this section, θ, $\dot{\theta}$, τ are all scalars, and h is a scalar function. The Euler's method of numerical integration of this dynamical equation can be written as follows.

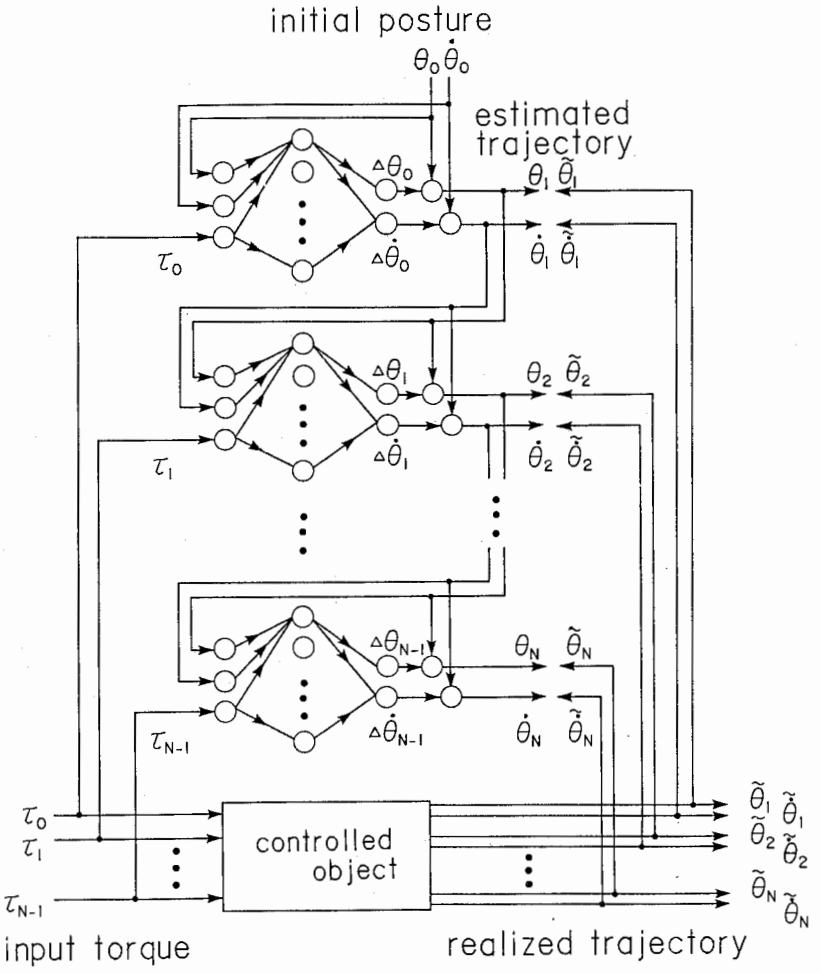
$$\theta((j+1)\Delta t) = \theta(j\Delta t) + \Delta t \cdot \dot{\theta}(j\Delta t) \tag{3.3}$$

$$\dot{\theta}((j+1)\Delta t) = \dot{\theta}(j\Delta t) + \Delta t \cdot h(\theta(j\Delta t), \dot{\theta}(j\Delta t), \tau(j\Delta t)). \tag{3.4}$$

The cascade structure of the neural network model shown in Fig. 3..1 is simply a direct hardware implementation of Euler's method above.

A

Energy Learning Phase



B

Energy Minimization Phase

— Trajectory Formation —

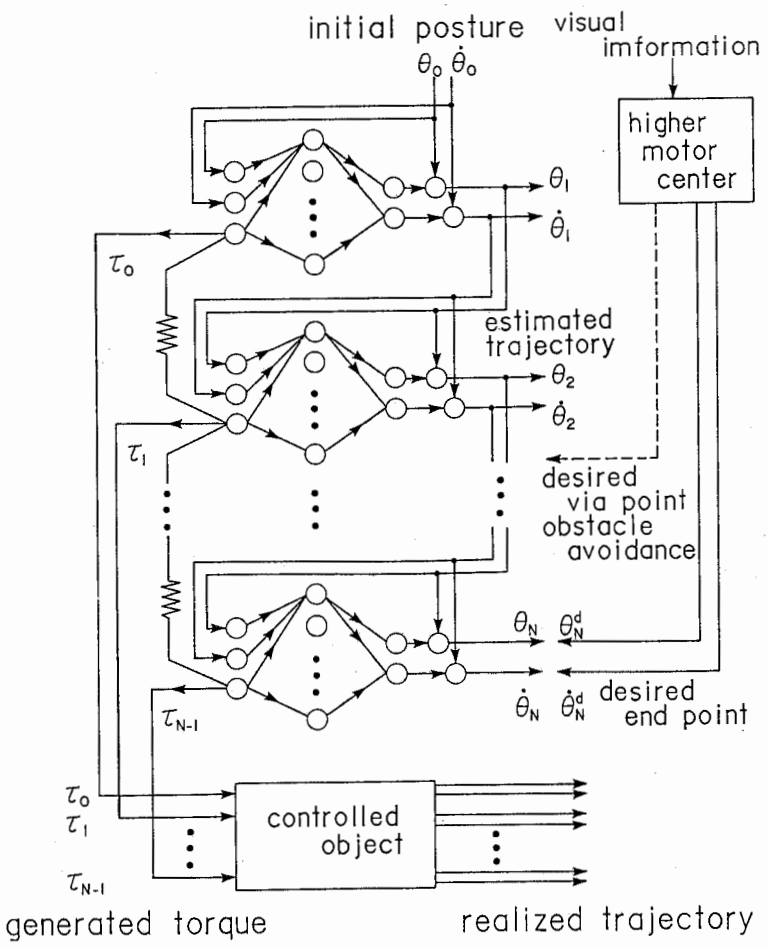


Figure 3..1: A repetitively structured cascade neural network model for trajectory formation based on the minimum torque-change criterion. For simplicity, a single degree of freedom case without the kinematics problem is shown. All four-layer network units are identical. Operations of the model are divided into (a) the learning phase, and (b) the pattern generating phase. In the learning phase, the network unit learns the vector field of the dynamical equation which describes the forward dynamics of the controlled object. In the pattern generating phase, the network relaxes its state to minimum energy equilibrium based on backward propagation of errors through the acquired forward model.

The model consists of many identical four-layer network units. The jth network unit corresponds to time $j\Delta t$. The network units are connected in a cascade formation. Operation of each neuron is assumed to be the 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. Let $u^i_{m,j}$ and $y^i_{m,j}$ denote an average membrane potential (i.e. weighted summation of all synaptic inputs) and an output (i.e. average firing frequency) of the ith neuron in the mth layer of the jth network unit. ϕ is the input-output transformation. $w^{m-1,m}_{k,i}$ denotes a synaptic weight from the kth neuron in the $m-1$th layer to the ith neuron in the mth layer. Then the following equations hold:

$$u^i_{m,j} = \sum_k w^{m-1,m}_{k,i} y^k_{m-1,j}, \tag{3.5}$$

$$y^i_{m,j} = \phi(u^i_{m,j}). \tag{3.6}$$

The cascade structure is formed by a unit-weight serial connection from the fourth layer of the jth network unit to the first layer of the $j+1$th network unit, and a unit-weight connection bypass from the fourth layer of the jth network unit to the fourth layer of the $j+1$th network unit:

$$y^i_{1,j+1} = y^i_{4,j}, \tag{3.7}$$

$$y^i_{4,j+1} = y^i_{4,j} + y^i_{3,j+1}. \tag{3.8}$$

The serial connections correspond to the second terms of the right sides of Euler's formula. The bypass connections correspond to the first terms of the right sides, and just convey the previous states to the next time step. Actually, eq. 3.8 corresponds to eqs.3.3 and 3.4. 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.

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, $\Delta t \cdot h(\theta(j\Delta t), \dot{\theta}(j\Delta t), \tau(j\Delta t))$. The second layer is expected to provide the 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. The fourth layer neurons in the jth unit output the estimated trajectory, $\theta(j\Delta t)$ and $\dot{\theta}(j\Delta t)$ at time $j\Delta t$, which are the summation of their two synaptic inputs, i.e., the outputs of the third layer in the jth unit and outputs from the fourth layer neurons in the $(j-1)$st unit (see eq. 3.8). The first two neurons in the first layer of the first unit represent the initial conditions $\theta(0)$ and $\dot{\theta}(0)$. Because the model faithfully reproduces the time structure of the dynamical system, the three intrinsic properties of the flow of the dynamical system, which are listed in the beginning of this section, are embedded into the cascade structure of the model.

Exactly the same as in the previous model, operations of this network are divided into the learning phase (Fig. 3..1a) and the pattern-generating phase (Fig. 3..1b).

In the learning phase (Fig. 3..1a), the common input torque are fed to both the controlled object and the neural network model. The realized trajectory $(\tilde{\theta}, \tilde{\dot{\theta}})$ from the controlled object is used as a teaching signal to acquire the forward dynamics model 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 error function E for the output of the 4th layer.

$$E = 1/2 \sum_{c} \sum_{j=1}^{N} [\{\tilde{\theta}_c(j\Delta t) - \theta_c(j\Delta t)\}^2 + \{\tilde{\dot{\theta}}_c(j\Delta t) - \dot{\theta}_c(j\Delta t)\}^2]. \tag{3.9}$$

Here, c represents the input torque and the resulting trajectory. The back-propagation learning algorithm (Rumelhart, Hinton, & Williams, 1986; Werbos, 1988) can be applied to this case as follows. Let $\delta^i_{m,j}$ denote an error signal of the ith neuron in the mth layer of the jth network unit. Then, $\delta^i_{m,j} = -\partial E / \partial u^i_{m,j}$. The error signal $\delta^i_{4,j}$ of the ith neuron in the 4th layer of the jth network unit is the summation of the error between the realized trajectory and the estimated trajectory at time $j\Delta t$, the error signal of the ith neuron in the 4th layer of the $(j+1)$st network unit and the error signal of the ith neuron in the 1st layer of the $(j+1)$st network unit:

$$\begin{aligned} \delta^i_{4,j} &= \sum_{c} \{\tilde{\Theta}^i_c(j\Delta t) - \Theta^i_c(j\Delta t)\} + \delta^i_{4,j+1} + \delta^i_{1,j+1}, \\ & j = 1, 2, ..., N; i = 1, 2, \end{aligned} \tag{3.10}$$

here $\Theta = (\theta, \dot{\theta})$. The error signal of neurons in the 3rd layer is the same as that of the 4th layer. The error signals of neurons in the 1st and the 2nd layers are computed from those in the 3rd layer as in the usual back-propagation learning rule:

$$\delta^i_{m,j} = \phi'(u^i_{m,j}) \sum_{k} w^{m,m+1}_{i,k} \delta^k_{m+1,j}. \tag{3.11}$$

After the incremental changes of synaptic weights 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 (Fig. 3..1b), electrical couplings between neurons representing torques in the 1st layer of the neighboring units are activated while torque inputs to the 1st layer and the teaching signal to the 4th 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 4th layer from the higher motor center which receives the necessary information for the trajectory specification. Locations of the end point, the intermediate point and the obstacle are given to the fourth-layer of the network units.

The error signals for all neurons are calculated from eq. 3.10 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. If we consider the simple case without via-point or obstacles, the backward propagation

of errors in the pattern generating phase obeys the following equation:

$$\begin{aligned} \delta^i_{4,N} &= \Theta^i_{d,N} - \Theta^i_N, \\ \delta^i_{4,j} &= \delta^i_{4,j+1} + \delta^i_{1,j+1}, \\ & \quad j = 1, 2, ..., N-1; i = 1, 2, \end{aligned} \tag{3.12}$$

here $\Theta_{d,N} = (\theta_{d,N}, \dot{\theta}_{d,N})$ represents desired terminal conditions. The error signals are not used for synaptic modification. However, 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 forward synaptic calculation and by back-propagation of error signals through the cascade structure while obeying the following differential equation.

$$\begin{aligned} T \cdot d\tau_j/ds &= \delta^3_{1,j} + g(\tau_{j+1} - 2\tau_j + \tau_{j-1}) \\ & \quad j = 0, 1, 2, ..., N-1. \end{aligned} \tag{3.13}$$

Here, s is the time of neuron state change and has nothing to do with the movement time t. g denotes the electrical conductance of the gap junction. We note that calculation of the error signal $\delta^3_{1,j}$ at every instant s in eq. 3.13 requires both the forward calculation of every neuron state through the entire network with learned synaptic weights, and the backward propagation of the error signal by eq. 3.12 through the entire network.

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

$$L = 1/2\{\theta_d(N\Delta t) - \theta(N\Delta t)\}^2 + 1/2\{\dot{\theta}_d(N\Delta t) - \dot{\theta}_d(N\Delta t)\}^2 + 1/2 \cdot g \sum_{j=0}^{N} (\tau_j - \tau_{j-1})^2 \tag{3.14}$$

Here we show the Ljapunov function in a simple case without the via-point or the obstacle avoidance conditions. The first two terms of the Ljapunov function require that the hand reaches the end point with a specified speed and the third 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. An appropriate delay line should be inserted between the cascade network and the controlled object in Fig. 3..1b. The conductance g is slowly decreased to 0, during relaxation of the state point to the equilibrium, similar to the temperature in "simulated annealing" (Kirkpatrick, Gellat, Vecci, 1983).

The first two terms of the Ljapunov function are the hard constraints imposed by trajectory specification, and the third term is the smoothness constraint. Introduction of the third energy as electrical couplings resolves the ill-posed trajectory formation. The network learns the first two terms of the energy as synaptic weights, and then minimizes the total energy.

Mathematically, the relaxation computation of the cascade network has a close relationship with the first-order gradient method which is a well known numerical

method in optimal control theory (Bryson and Ho, 1975). As mentioned before, the forward calculation through the cascade structure using eqs. 3.3 and 3.4 is interpreted as forward numerical integration of the control system equation 2.1. Similarly, the backward propagation of error is interpreted as backward integration of a part of an adjoint equation of the dynamics eq. 2.1. Equation 3.12 can be interpreted as backward Euler's numerical integration of a part of the adjoint equation (see Appendix). The proposed network is different from the control theory approach in that it does not utilize any co-state (in this case $\dot{\tau}$). Because of this, (i) the modification algorithm of the motor command is different, (ii) the neural network model uses fewer state variables, (iii) and it is several times faster. The most marked advantage of the neural network model is that we can impose constraints on the motor command by directly constraining states of the neurons which represent the motor command. Instead, if we use the co-state, these constraints must be treated as those on the state variables, which leads to further complication of the calculations. This is important, since when we extend the minimum torque-change model to the minimum muscle tension-change model, we must introduce positivity constraints on muscle tensions. Actually, in this case, if we use a standard sigmoid input-output transformation function for neurons representing motor commands, we do not need to introduce any extra constraint for positivity of muscle tension at all.

4. Cascade dynamics network in conjunction with forward kinematics model for a redundant controlled object

In this section we consider a multi-degree of freedom controlled object, which has redundancy at both the dynamics and kinematics levels. For trajectory formation of this controlled object, we propose a combination network model which consists of the cascade neural network shown in Fig. 3..1 as a forward dynamics model and MLFF network as a forward kinematics model. This extended model can simultaneously solve the three ill-posed problems illustrated in Fig. 1..1.

Let us return to the notations and equations in Section 2. The motor command has m control variables. The body coordinates have $n$degrees of freedom. The task-oriented coordinates have k dimensions. We consider the general case of $m \geq n \geq k$.

For acquiring the forward dynamics model described by eq. 2.1, we use the cascade network shown in Fig. 3..1. But in this case, the first layer of the network unit has $2n + m$ neurons, the third layer and the fourth layer have $2n$ neurons. The forward kinematics described by eq. 2.3 is separately acquired in an MLFF by training. The forward kinematics network has an input layer which represents the position of the body coordinates and an output layer which represents the position of the task-oriented coordinates. The number of neurons in the input layer and the output layer are n and k. Thus, the network learns the many-to-one mapping. Learning the forward dynamics model and the forward kinematics model can be done separately by providing them with appropriate inputs and teaching signals.

In the pattern generating phase, the forward kinematics model is attached to the output line of each network unit in the cascade neural network model. Thus, we

need the same number of duplicates of the forward kinematics model as the number of network units. Among the four movement conditions, only the terminal velocity condition is represented in body coordinates and hence directly given to the cascade dynamics network model. This is because a stop of the hand does not necessarily imply a stop of the arm for a redundant controlled object. The other three conditions regarding the end point location, the via-point and obstacle avoidance condition are all expressed in the task-oriented coordinates. Correspondingly, objective signals are given to the output layers of the forward kinematics model. The error signals are first calculated at the output layer of the forward kinematics model, then they are back-propagated through the forward kinematics model and subsequently through the cascade dynamics network. Based on this back-propagated error signal, relaxation computation is done exactly as described in the previous section.

We note that if $n > k$, we can not describe dynamical changes of states of the controlled object by only using the task-oriented coordinates. This is why the cascade network cannot acquire the composite dynamics and kinematics model if $n > k$.

The inverse dynamics and inverse kinematics problems for redundant control objects can not uniquely be solved by the direct inverse modeling approach (Albus, 1975; Miller, 1987; Miller, Glanz, Kraft, 1987; Kuperstein, 1988; Atkeson, Reinkensmeyer, 1988). We emphasize that the composite neural network model resolves these ill-posed problems based on the minimum torque-change criterion.

5. Simulation for a two-link arm model

5.1 Learning forward dynamics and kinematics model

The performance of the proposed network model was examined in computer simulation experiments of trajectory formation. A two-link robotic manipulator was used as a model of a human right arm (see Fig. 5..1). Links 1 and 2 correspond to the upper arm and the forearm, and joints 1 and 2 correspond to the shoulder and the elbow. Joint 1 (shoulder) is located at the origin of the $X-Y$ cartesian coordinates. The positive direction of the X coordinate is the right side direction of the body, and the positive direction of the Y coordinate is the front direction of the body (see Fig. 5..1). Physical parameters of the model manipulator are shown in Table 5..1. They were chosen based on experiment data and human arm geometry (see Uno et al., 1989). The hand position is represented as (X, Y) in meter.

Because the manipulator has two degrees of freedom within the plane, there is no redundancy at the kinematics level. Because the manipulator has only one actuator at each joint, there is no redundancy at the dynamics level. Consequently, in this simplest case, the forward kinematics and dynamics of the manipulator can be described by the following ordinary differential equations expressed in task-oriented coordinates.

$$\begin{aligned} dX/dt &= \dot{X} \\ dY/dt &= \dot{Y} \end{aligned}$$

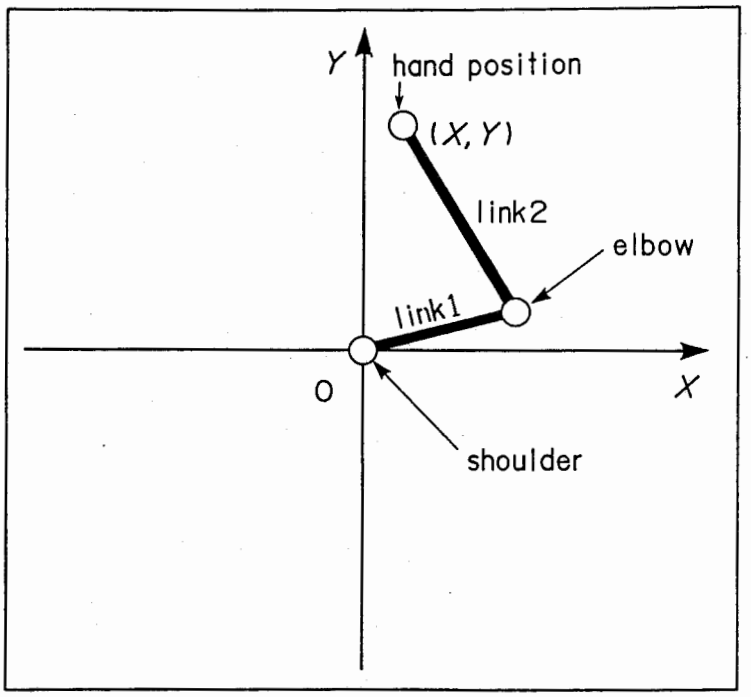


Figure 5..1: A two-joint robotic manipulator within a horizontal plane as a model of a human arm. See Table 5..1 for values of physical parameters of the model manipulator.

Parameter	Upper arm	Forearm
L_k(m)	0.25	0.35
S_k(m)	0.11	0.15
M_k(kg)	0.9	1.1
I_k(kgm^2)	0.065	0.100
B_k(kgm^2/s)	0.07	0.07

Table 5..1: Physical parameters of the model manipulator shown in Fig. 5..1.

$$
\begin{aligned}
d\dot{X}/dt &= p(X, Y, \dot{X}, \dot{Y}, \tau^1, \tau^2) \\
d\dot{Y}/dt &= q(X, Y, \dot{X}, \dot{Y}, \tau^1, \tau^2).
\end{aligned} \tag{5.1}
$$

Here, $\tau^1(t)$ and $\tau^2(t)$ represent torque waveforms fed to the shoulder and the elbow. p and q are nonlinear functions which determine the nonlinear dynamics and kinematics of the manipulator.

Correspondingly, we chose the simple cascade neural network model shown in Fig. 3..1 as a composite model of the forward dynamics and kinematics of the controlled object. In other words, the right side of eq. 5.1 is acquired in a three-layer network unit. The first layer of each network unit contains 6 neurons. 4 of them represent the task space trajectory $X, Y, \dot{X}, \dot{Y}$, and the other 2 represent torque waveforms τ^1, τ^2. The third layer of each network unit contains 4 neurons which represent changes of the task space trajectory within a unit of time (i.e. the right sides of eq. 5.1 multiplied by Δt). The fourth layer of each network unit contains 4 neurons, and they represent the task space trajectory $X, Y, \dot{X}, \dot{Y}$. There are 20 hidden neurons in the second layer of each network unit. Hence, there are 34 neurons in each network unit. 200 synaptic weights need to be learned to acquire combination of forward dynamics and kinematics.

The movement time was set at 500ms. 5ms was chosen for the time step Δt of the network. Thus, we prepared 100 network units. The total neural network model contains 3,400 neurons and 20,000 modifiable synaptic weights. However, only 200 synaptic weights are independent parameters to be acquired since each network unit is identical.

Two different learning schemes are possible to train the cascade network as the forward model. The first was described in Section 3, and uses back-propagation of trajectory errors all through the cascade structure. The second scheme is simpler than this. We detach each network unit from the cascade structure. The jth network unit receives the realized trajectory and the motor command at $(j-1)\Delta t$ as inputs, and receives the realized trajectory at $j\Delta t$ as a desired output or the teaching signal. Consequently, each network unit is trained to acquire only the vector field multiplied by Δt, and no back-propagation of trajectory errors is conducted.

Learning of the forward dynamics and kinematics model was conducted utilizing both schemes with 40 training trajectories whose durations were all 500ms. The starting point and the end point of the training trajectory were located within a circle with a center (0.2,0.2) and a radius 0.15. Each trajectory contains 100 sampling points. Therefore, the training set consists of 4,000 data. This is sufficiently large compared with the 200 independent synaptic weights, so we expect that the network generalizes well after sufficient learning. First, the training set was given to the network 1,000 times utilizing the second learning scheme. Next, the same training set was again given 1,000 times with the first learning method. The estimated trajectories by the network were almost identical to the teaching trajectories after learning. The network estimated fairly good trajectory even for an inexperienced torque input. Then we used the trained network for arm trajectory formation.

5.2 Trajectory between two points

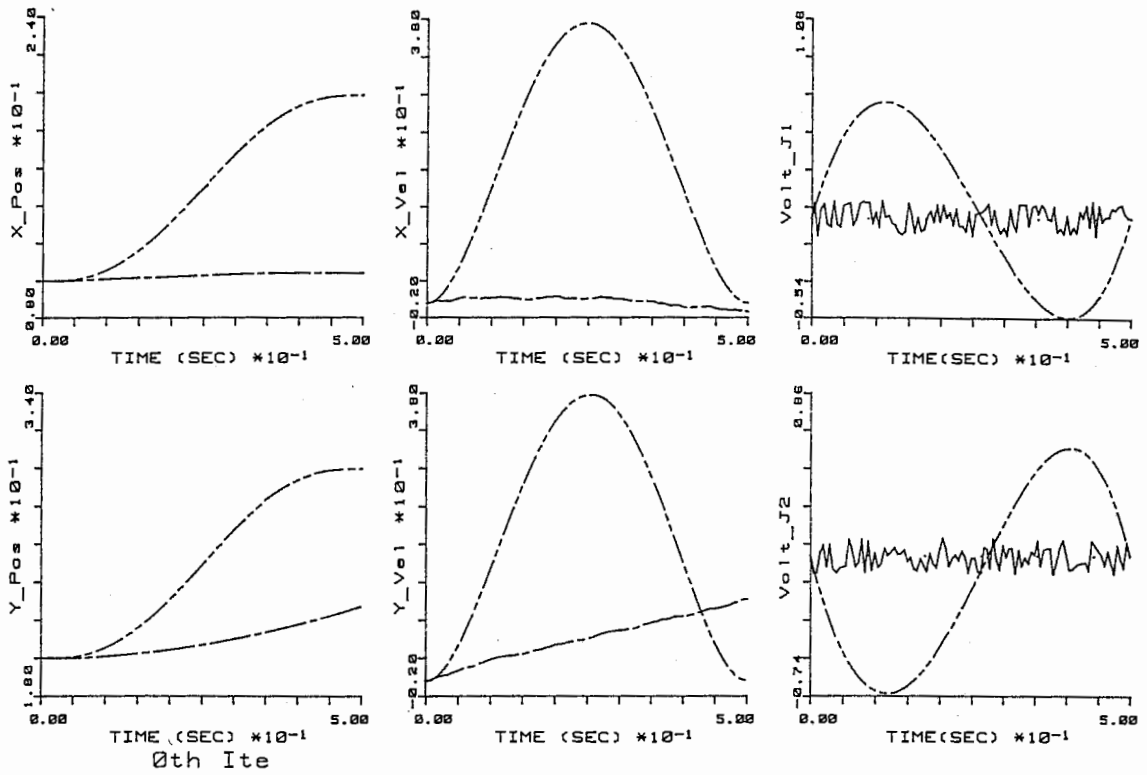
First, trajectory formation between two points was examined. The objective is to move the arm between the starting point (0.1,0.2) and the end point (0.2,0.3) in 500ms based on the minimum torque-change criterion.

Fig. 5..2a shows the initial condition of the network in relaxation computation for trajectory formation. The left column shows the hand trajectory X (above) and Y (below). The middle column shows the hand velocity time course $\dot{X}$ (above) and $\dot{Y}$ (below). The right column shows the torque waveforms fed to the shoulder τ^1 (above) and to the elbow τ^2 (below). The two-point chain curves show the exact trajectory and torque of the minimum torque-change model, which were obtained by the Newton-like method. The one-point chain curves show the trajectory estimated by the network (i.e. outputs from the 4th layer of the network units) and the states of the torque neurons (5th and 6th neurons in the 1st layer of network units). The states of 200 torque neurons were set at random initial values at the beginning of the relaxation. After 1,000 iterations of relaxation computation according to eq. 3.13, the state settles down to a stable equilibrium and generates an estimated trajectory and a torque shown in Fig. 5..2b. Solid curves in Fig. 5..2b show a trajectory realized by the model manipulator when fed the torque estimated by the network. The estimated trajectory reaches the end point at the given time and stopped properly. The generated torque waveforms were smooth enough and fairly close to the exact solution.

The realized trajectory did not reach the end point. There are consistent errors between the minimum torque-change trajectory (two-point chain curve), the estimated trajectory (one-point chain curve) and the realized trajectory (solid curve). We know that the cause of this error is incomplete learning of dynamics and kinematics, since the error is almost negligible in the simulation experiments where an accurate forward model is used (see section 5.4). Thus, we expect that the network can generate a more accurate trajectory and torque if we use more efficient learning algorithms such as the conjugate gradient method or the quasi-Newton method instead of the back-propagation (steepest descent) algorithm, and if we carefully select more independent training data.

Fig. 5..3 shows a bird's-eye view of another example of a hand path between the starting point (0.3,0.3) and the end point (0.15,0.15), and configurations of the arm every 125ms. The minimum torque-change hand path and the realized hand path are shown by the two-point chain curve and the solid curve, respectively. At this resolution, the minimum-torque change hand path and the one estimated by the neural network model can not be separately seen. Again, the error between the realized hand path and the minimum torque-change hand path was due to the incomplete learning of the forward model.

A


X_Pos *10-1
2.40
0.00
0.00
5.00
TIME (SEC) *10-1
X_Vel *10-1
3.80
-0.20
0.00
5.00
TIME (SEC) *10-1
Volt_J1
1.00
-0.54
0.00
5.00
TIME(SEC) *10-1
Y_Pos *10-1
3.40
1.80
0.00
5.00
TIME (SEC) *10-1
Y_Vel *10-1
3.80
-0.20
0.00
5.00
TIME (SEC) *10-1
Volt_J2
0.86
-0.71
0.00
5.00
TIME(SEC) *10-1
0th Ite

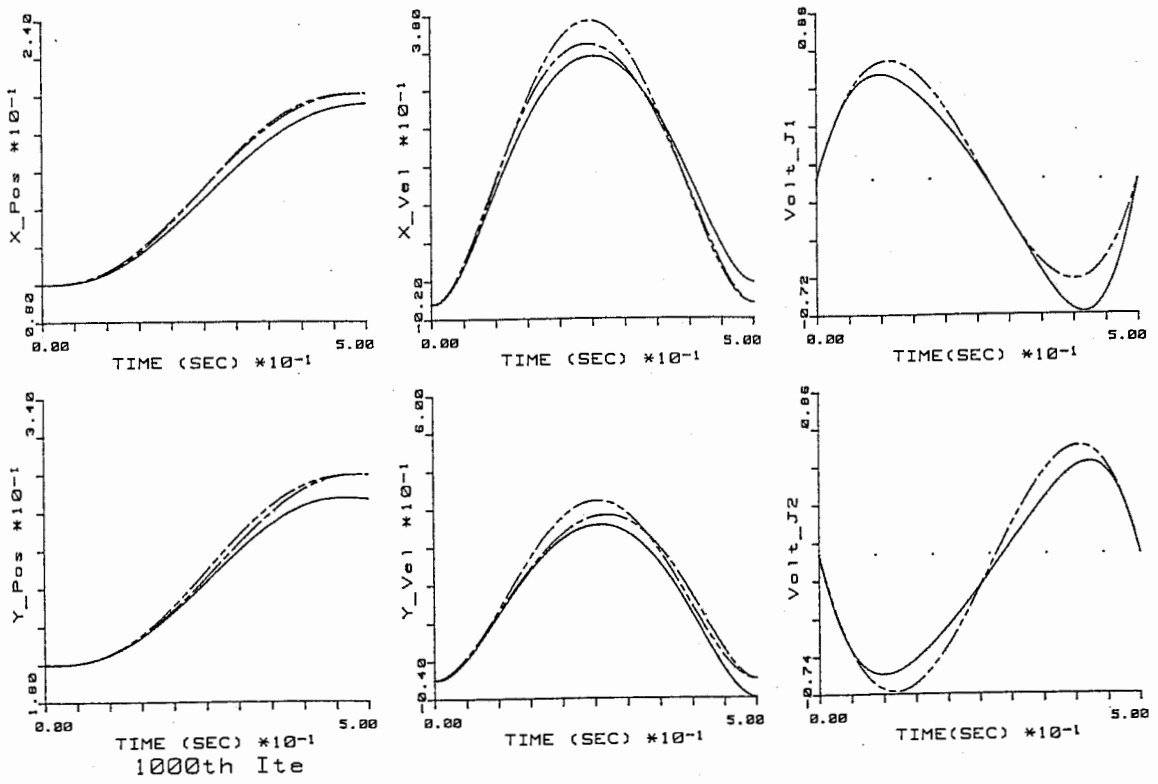


Figure 5..2: Relaxation of the neural network state during generation of a movement between two points. (a) shows the initial condition of the neural network model. (b) shows the stable equilibrium state of the network after 1,000 iterations and the corresponding realized trajectory by the model manipulator. The abscissa of all the figures shows the time in seconds. The left column shows the hand position trajectory $X(t)$ (above) and $Y(t)$ (below). The scale of the ordinate is in meters. The middle column shows the hand velocity history $\dot{X}(t)$ (above) and $\dot{Y}(t)$ (below). The scale of the ordinate is in meters per second. The right column shows the torque waveforms fed to the shoulder (above) and to the elbow (below). The scale of the ordinate is in volts. Two-point chain curves are exact trajectories and torque waveforms of the minimum torque-change model calculated by a Newton-like method. One-point chain curves in the left and middle columns are trajectories estimated by the network model. Solid curves in the right column represent torque waveforms generated by the network model. Solid curves in the left and middle columns in (b) show the realized trajectory by the model manipulator when the generated torque is fed to it. Fig. 5..4b and Fig. 5..6b will be presented in the same format as (b).

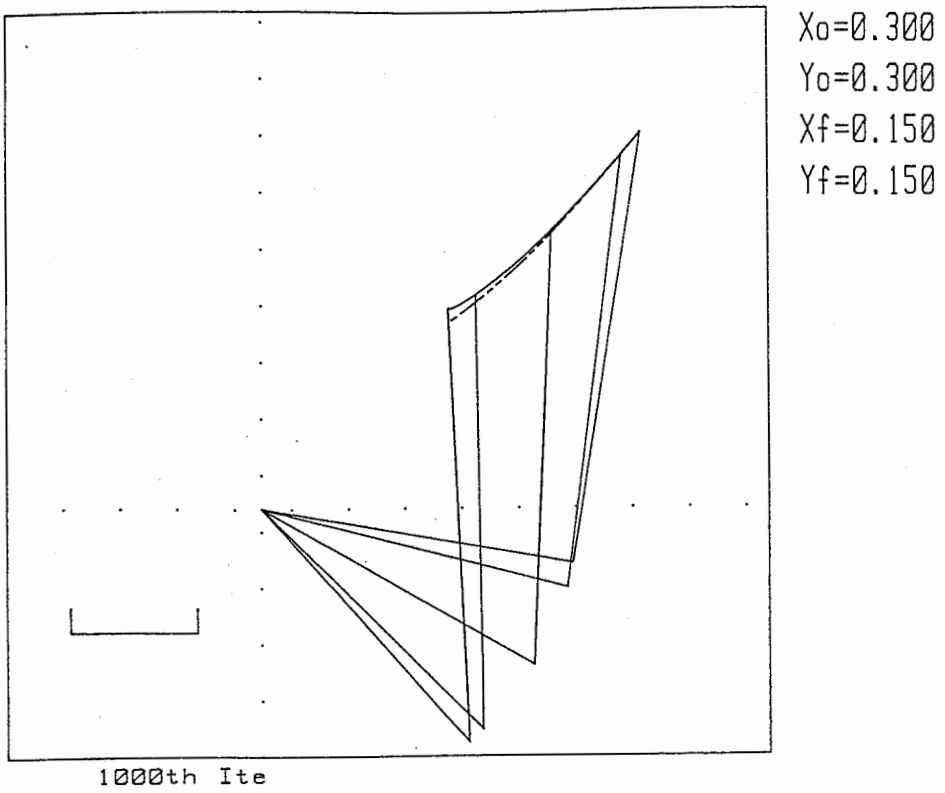
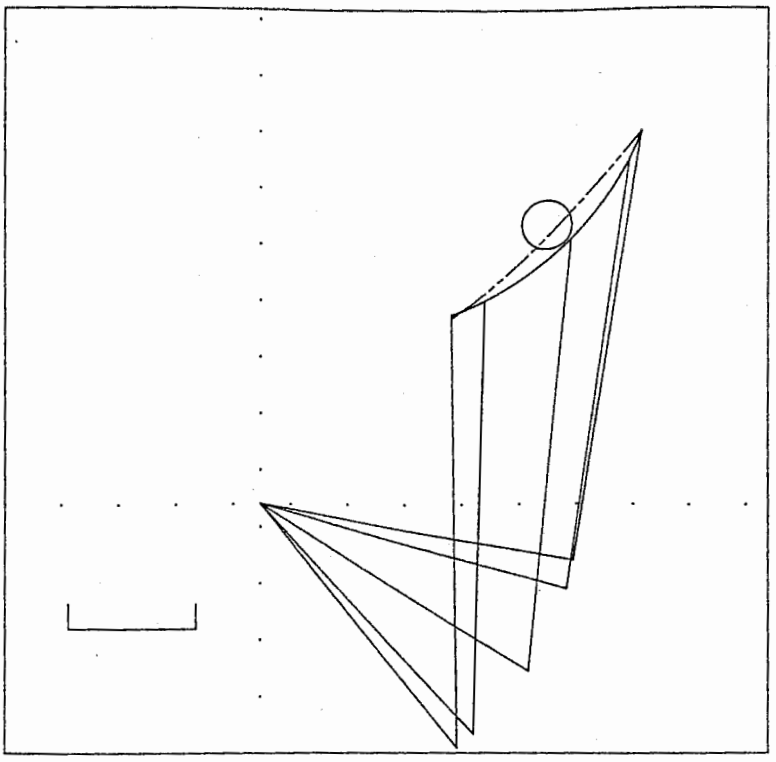


Figure 5..3: Bird's-eye view of a hand path between two points and 5 configurations of the model manipulator every 125ms. The scale in the figure is 0.1m. The solid curve is the realized hand path, and the two-point chain curve is the exact minimum torque-change hand path calculated by a Newton-like method. The number of relaxation computation iterations was 1,000.

A



Xo=0.300
Yo=0.300
Xf=0.150
Yf=0.150
Ox=0.225
Oy=0.225
Ro=0.020

5.3 Obstacle avoidance trajectory

We simulated trajectory formation by the neural network model while avoiding obstacles. First, a simple case with one obstacle which is a circle with center (C_X, C_Y) =(0.225,0.225) and a radius R=0.02m was examined (see Fig. 5..4a). The starting point and the end point were (0.3,0.3) and (0.15,0.15), which are the same as in Fig. 5..3.

Let (X_j, Y_j) denote the hand position at the jth time which is estimated by the neural network model. r_j represents a distance between the center of the circle and the estimated hand position at the jth time. The non-negative obstacle avoidance function in eq. 2.16 was chosen as follows:

$$\begin{aligned} H(X_j, Y_j; \Omega) &= (r_j - R)^2/2 \quad (r_j < R) \\ &= 0 \quad (r_j \geq R). \end{aligned} \tag{5.2}$$

With this obstacle avoidance potential, the corresponding obstacle avoidance error signals for the first and the second neurons in the fourth layer of the jth network unit can be calculated as follows:

$$\delta^1_{4,j} = -h(r_j)(C_X - X_j) \tag{5.3}$$

$$\delta^2_{4,j} = -h(r_j)(C_Y - Y_j) \tag{5.4}$$

$$\begin{aligned} h(r_j) &= (R - r_j) r_j^{-1} \quad (r_j < R) \\ &= 0 \quad (r_j \geq R). \end{aligned} \tag{5.5}$$

Note that the sign of the error signal is opposite that of the usual error signals associated with the desired end or via-points, since h is positive. Thus, the center of the obstacle acts like a repelling point when the hand position is within a circle. If the hand is located outside the obstacle, the obstacle does not exert any force on the arm at all.

Fig. 5..4a shows the hand path with the same format as in Fig. 5..3. Fig. 5..4b shows corresponding trajectories and torque waveforms. The two-point chain curves in Fig. 5..4a and b show hand paths, trajectories and torque waveforms for the minimum torque-change trajectory without the obstacle. One can see considerable deformation of the torque waveforms because of avoidance of even the small obstacle.

The neural network model can also generate a trajectory while avoiding multiple obstacles as shown in Fig. 5..5. The starting point is (0.1,0.2), and the end point is (0.2,0.3). Centers of the two circle obstacles with a radius 0.02m are (0.125,0.245) and (0.175,0.255). In this simulation, we used the exact equation shown in eq. 5.1 for forward calculation in the relaxation computation instead of using the network unit. Similarly, we used the transpose of the Jacobian of the right hand sides of eq. 5.1 with respect to $X, Y, \dot{X}, \dot{Y}, \tau^1, \tau^2$ during backward calculation of error signals instead of the usual back-propagation of the neural network model. Because we used the exact forward dynamics and kinematics model, the estimated trajectory and the realized trajectory coincided almost perfectly (see Fig. 5..5). We note that the realized and

B

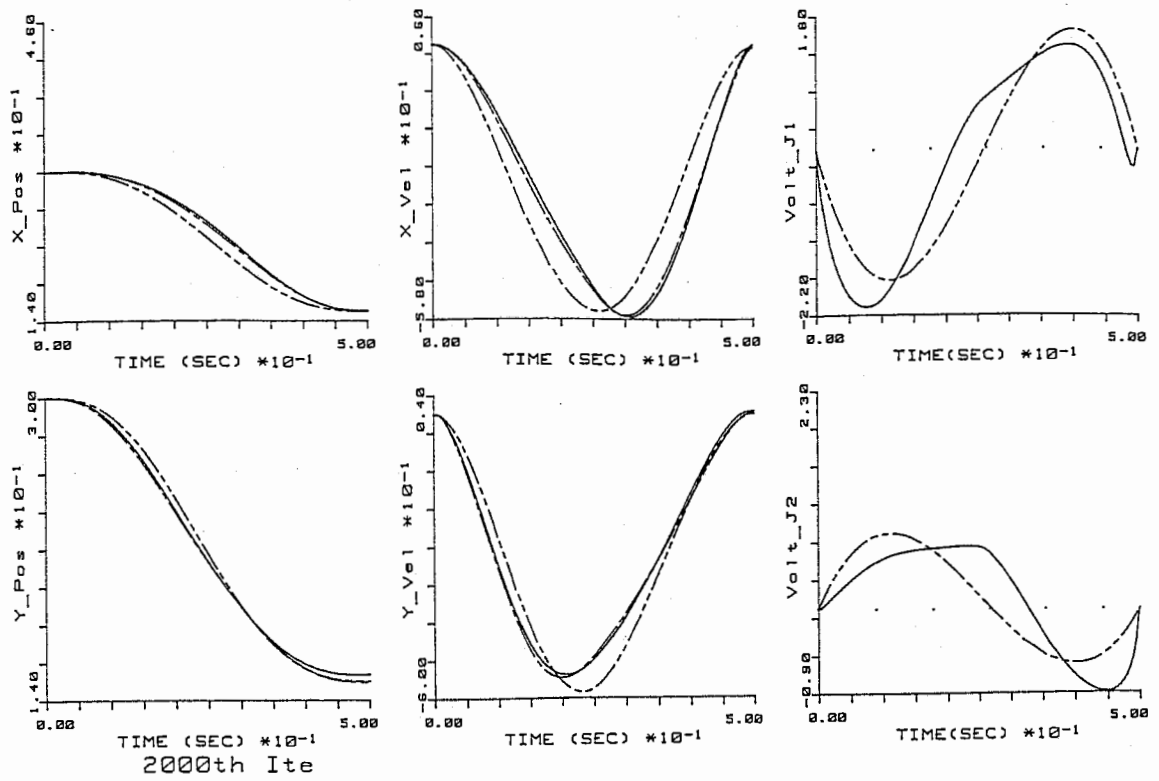


Figure 5..4: (a) Bird's-eye view of a hand path (solid curve) while avoiding a circle obstacle and 5 configurations of the model manipulator every 125ms. The format is the same as in Fig. 5..3. The two-point chain curve shows the minimum torque-change hand path when there is no obstacle. (b) Trajectories and torque waveforms for a movement avoiding an obstacle (solid curves). The format is the same as in Fig. 5..2b. The two-point chain curves show the minimum torque-change trajectory and the torque waveform when there is no obstacle. Note the considerable deformation of the torque waveforms due to the presence of the obstacle. The number of relaxation computation iterations was 2,000 in this experiment.

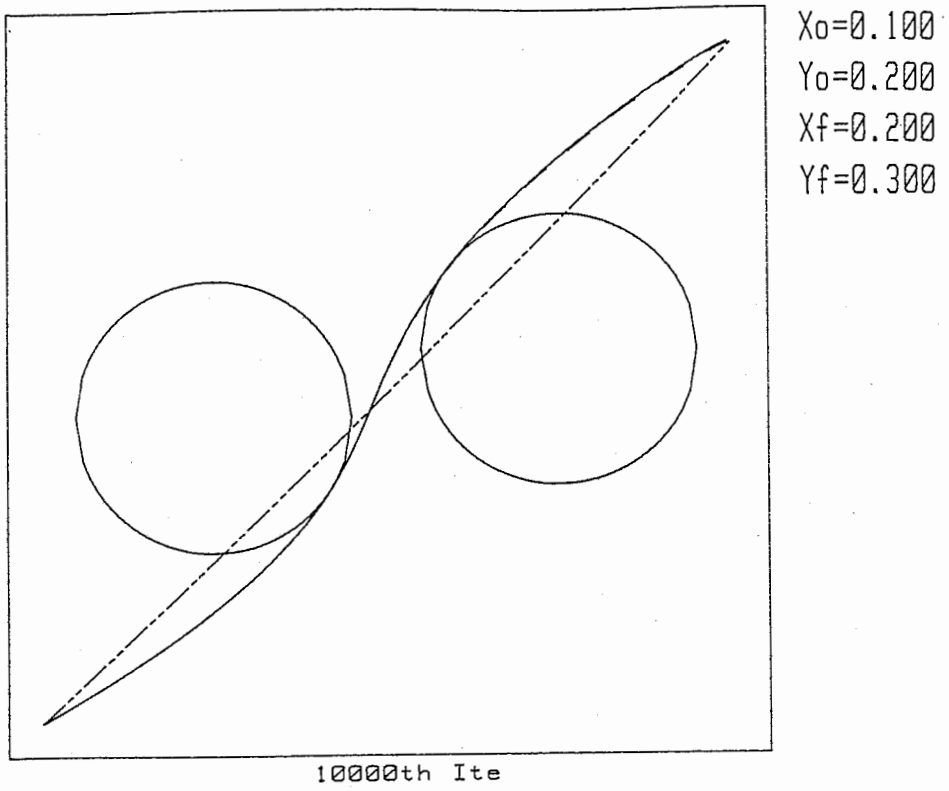


Figure 5..5: Bird's-eye view of a hand path which avoids two obstacles. A two-point chain curve shows the exact minimum torque-change hand path calculated by a Newton-like method when there is no obstacle. A one-point chain curve shows a hand path estimated by the network model. A solid curve shows a realized trajectory by the manipulator when the generated torque is fed to it. Because we used an accurate model of the forward dynamics in this simulation, these two curves almost coincided and can not be seen separately at this resolution. The number of relaxation computation iterations was 10,000 in this experiment.

estimated trajectories came in contact with the two obstacles because of the minimum torque-change criterion.

Because we imposed the obstacle avoidance potential only to the hand position, the links of the manipulator collided with the right obstacle in this case. We can extend the present model so that neither the links nor the hand collide with obstacles, as follows: First, we need to train a neural network model which estimates link locations from the hand position or from the joint angles. In the pattern generating phase, this extra forward kinematics model is attached to the output lines of the original cascade network. The error signals for obstacle avoidance are first calculated at the output side of this extra network, then they are back-propagated through it, and subsequently through the cascade network as before.

We note that the neural network model has multiple stable equilibrium in relaxation computation for avoidance of multiple obstacles (Fig. 5..5). Depending on the initial conditions, the network settled down to three different trajectories. The first is shown in Fig. 5..5. The second trajectory runs above the two obstacles, and the third one runs below. The forward dynamics and kinematics of the arm is nonlinear. The obstacle avoidance potential field is not convex. Consequently, the total energy of the network, eq. 2.10, is not convex, and the steepest descent method employed in this paper may be trapped to local minimum, as exemplified in this example. We do not think of this local minimum problem as a deficiency of our model. This kind of local minimum behavior is frequently seen in humans. Furthermore, we can avoid local minimum by adding noise to the relaxation equation as in the simulated annealing. However, we do not think this is necessary.

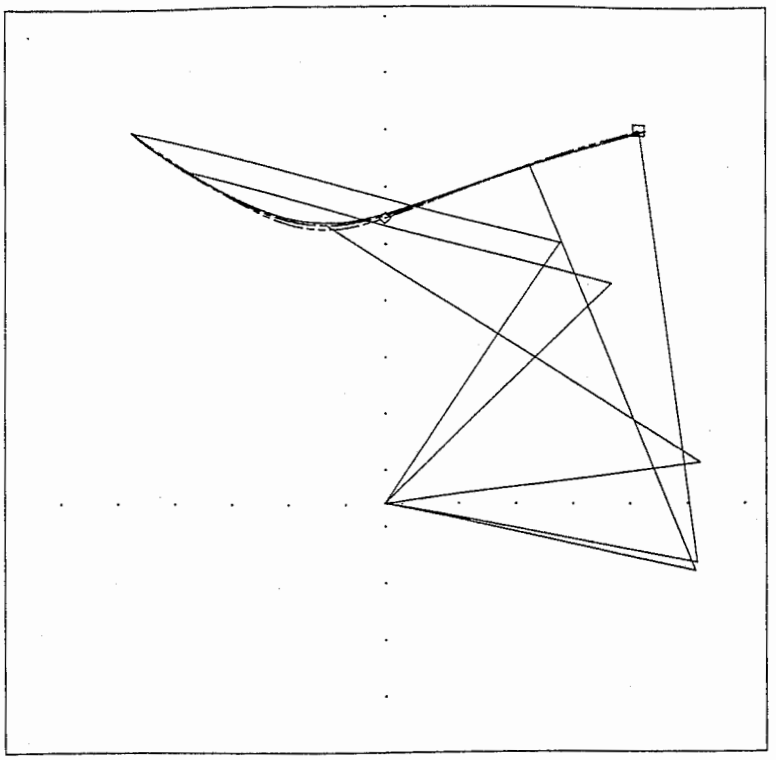
5.4 Via-point trajectory

The neural network generated a via-point trajectory which passes through the via-point (0,0.23) from the start point (-0.2,0.3) to the end point (0.2,0.3) (see Fig. 5..6a and b). Fig. 5..6a shows the hand paths and configurations of the manipulator. Fig. 5..6b shows trajectories and torque waveforms. Parallelograms in Fig. 5..6 show the location of the via-point. We used exact equations in forward and backward calculations instead of using the network unit. In this simulation we did not specify the time when the via-point must be passed. The neural network model autonomously finds the best time to pass the via-point on the minimum torque-change criterion using the energy of eq. 2.15. The estimated and realized trajectories were in fairly close agreement with the trajectory calculated by the Newton-like method. In particular, between these two methods the time required to pass the via-point differed by only 0.005s.

6. Discussion

We proposed a cascade neural network model for multi-joint arm trajectory formation, and showed that it can produce minimum torque-change motor commands and trajectories under several behavioral conditions. Examples shown include simple

A



Xo=-0.200
Yo=0.300
Xf=0.200
Yf=0.300
Xv=0.000
Yv=0.230
Vt= 58

B

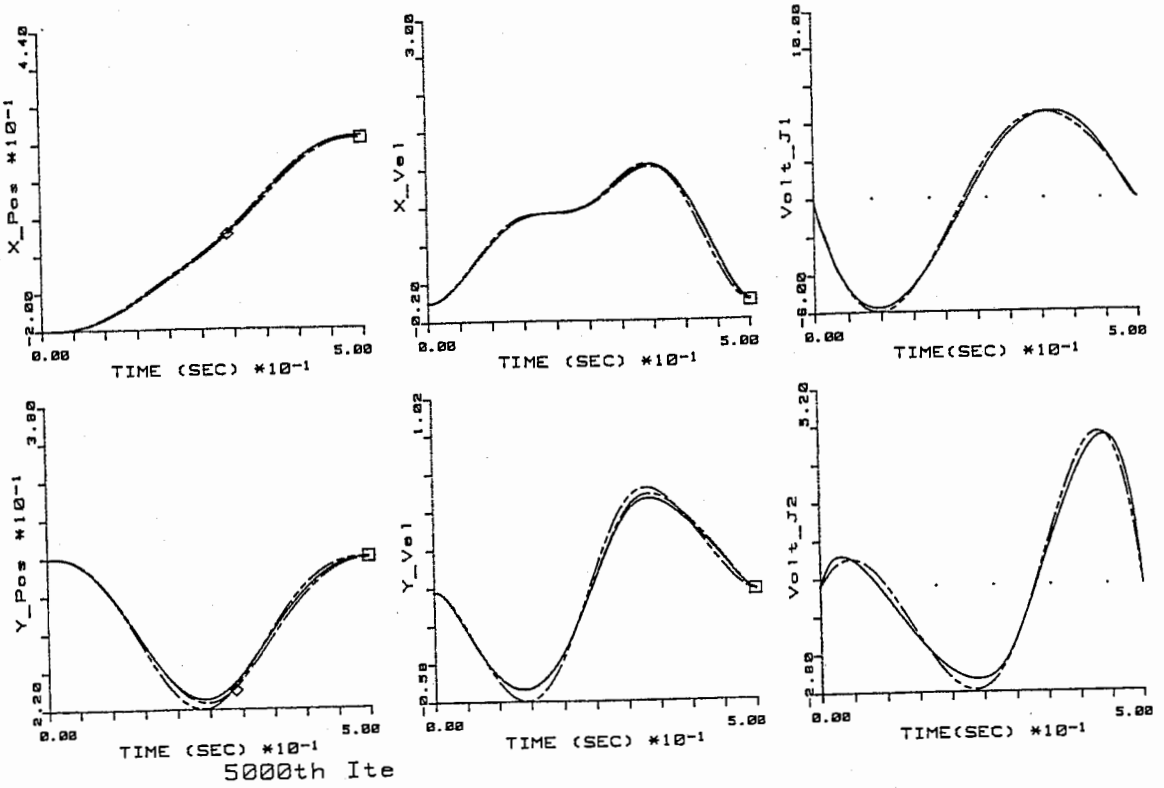


Figure 5..6: Hand paths, arm configurations (a), trajectories and torque waveforms (b) for a via-point movement. (a) is the same format as Fig. 5..3. (b) is the same format as Fig. 5..2b. The parallelogram in (a) shows the location of the via-point. Parallelograms in the left column of (b) show the corresponding X and Y coordinates of the via-point. The number of relaxation computation iterations was 5,000 in this experiment.

movements between two points, via-point movements, obstacle avoidance movements with a single obstacle or multiple obstacles. The neural network model first learns forward dynamics and kinematics of the controlled object. Then, based on the acquired model, it generates torques and trajectory by relaxation computations. The model predicts and reproduces human experiment data. The model can resolve ill-posed problems inherent in trajectory formation and control of a redundant controlled object (i.e. all three problems shown in Fig. 1..1 are ill-posed).

The trajectory formation problem has been actively studied by the nerual network modeling approach (Eckmiller, 1988; Bullock and Grossberg, 1988; Massone and Bizzi, 1989; Jordan, 1989).

The model proposed in this paper 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 the minimum jerk model. He noted that it is straightforward to implement the minimum torque-change criterion in his model. Our model and Jordan's both use the forward model of the controlled object. The idea of resolving the ill-posed motor control problems by introducing a smoothness constraint such as the minimum jerk criterion or minimum torque-change criterion is also common.

The essential difference between the two models is 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 times must be independently represented in the model, and hence time is represented spatially and the smoothness constraint is guaranteed by electrical coupling. Actually, if this were not necessary, our cascade network could be viewed as a spatially unrolled version of a recurrent network (see Fig. 5 of Rumelhart et al., 1986). 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 motor control subnetwork through learning.

One of the advantages of Jordan's model is that, once it learns the forward model of the controlled object and the smoothness constraint, it can generate a trajectory in real time. On the other hand, in our model, the relaxation time (typically hundreds of iterations) is required for trajectory formation. This may be a serious shortcoming of our network as a model of the brain because it is obvious that the motor control nerual network must calculate trajectory in real time (at least within a hundred milliseconds). There might be three ways to relax this shortcoming.

First, in computer simulation of relaxation computation, we discretized the relaxation time s. It is always difficult to estimate Δs in biological neural networks. When we deal with a continuous dynamical system without time delay which has a stable equilibrium point, we can arbitrarily speed up the convergence rate to the equilibrium by magnifying the vector field. Hence, the real problem is to estimate the extent of the time delay associated with chemical synaptic transmission and con-

duction along axons. One reason to expect that Δs in our neural network model is very small resides in electrical synapses. The interaction through gap junctions may not be associated with any significant delay, and convergence due to the smoothness constraint can be very fast. On the other hand, the forward and backward calculation through a vast number of cascaded network units inevitably induce considerable synaptic and conduction delays. Thus, whereas Δs may be large for the first term of eq. 3.13, it can be very small for the second term. We need to further investigate the system convergence rate in this complicated situation. Kitano et al. (unpublished observation) found that the iteration number of relaxation computation can be as small as 50 for movements between two points, if one considers the above point and uses the novel "virtual end point method" with a fixed relatively large g instead of the "simulated annealing" (decrease of g).

Second, we recently found that a multi-grid, multi-resolution (multi time-scale) extension of our network (Fig. 6..1) can calculate the trajectory an order of magnitude faster than the original model (Maeda et al., 1989; Maeda, 1989). Fig. 6..1 shows an example with three hierarchies. A model network within each hierarchy is just the cascade network shown in Fig. 3..1. At the higher hierarchy, the step size of the movement time is longer. Because of the smaller number of network units, the higher level network settles down to the equilibrium solution faster than the lower level. The higher level gives a rough initial solution to the lower level via a smoothing network, then the lower level calculates a more accurate solution. Besides the computation time, the multi-grid network may play an important role in trajectory planning with a long time scale. Although the number of synaptic weights to be learned in the cascade network is constant regardless of the movement time (see Section 3), the network size itself becomes larger linearly with the movement time. This might be a serious problem if we consider a long range movement planning (e.g. 10 second speech). The multi-grid network can resolve this size difficulty by introducing different smoothness constraints at different time scales which can fix the size of the lowest level cascade network.

Third, it is clear that the convergence time is very short if the initial condition for the relaxation computation is fairly close to the equilibrium solution. We can imagine that some associative content addressable memory (ACAM) network can store the converged solution of the cascade network, and then can instantaneously load it on the cascade network as the initial condition. One candidate of such ACAM is Jordan's recurrent network (1986), and this is the same usage of the recurrent network as a storage tool studied by Massone and Bizzi (1989). We are now collaborating Jordan to develop this aspect of both models.

One advantage of our model is that, once it learns the forward model of the controlled object, it can generate any trajectory regardless of locations of the end point, intermediate points and obstacles to be avoided. 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, intermediate points and obstacles so that it can generate them. This advantage of our model is especially important if the net-

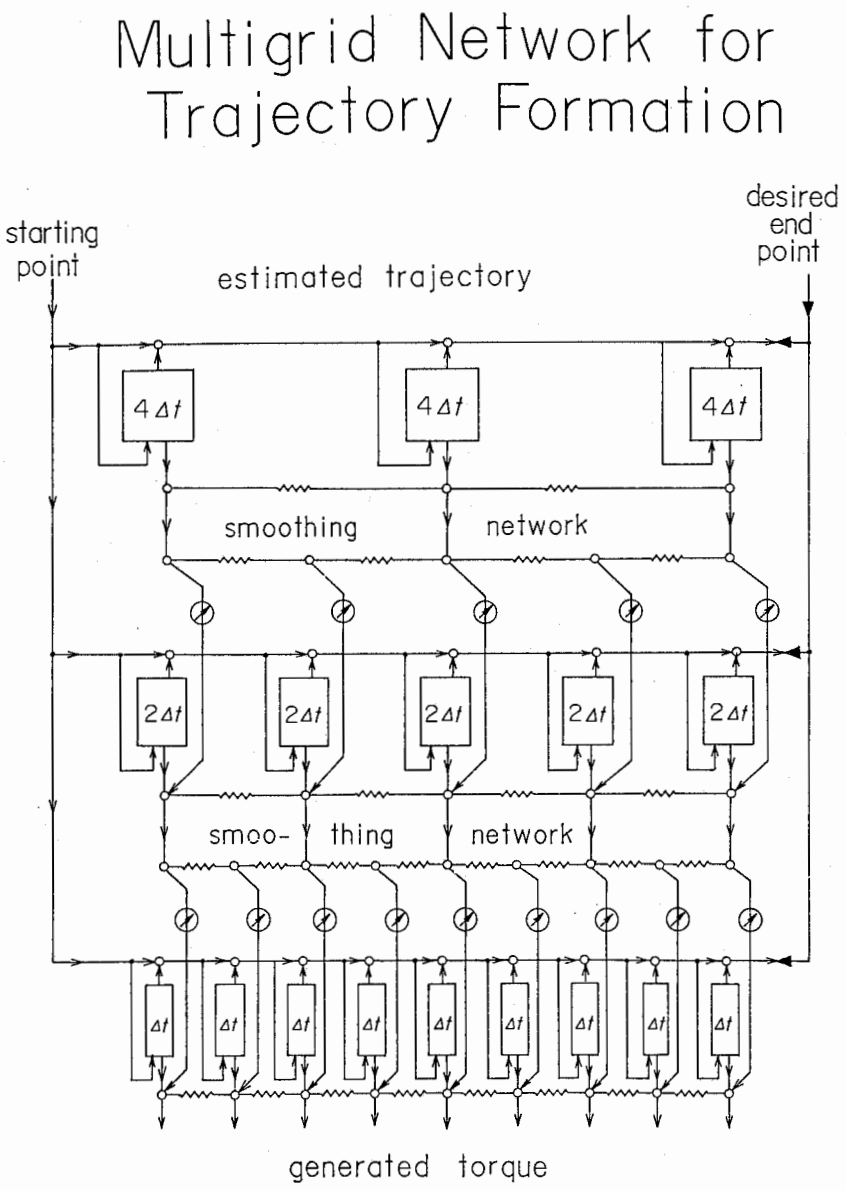


Figure 6..1: A multi-grid, multi-resolution neural network model for trajectory formation. A three level hierarchical structure is schematically shown. Each rectangle represents a three-layer network unit. The numeral in the rectangle shows the time step adopted in each hierarchy. The width of the rectangle schematically represents the length of the time step.

work is used for trajectory formation of an articulator. We believe that the minimum torque-change model established for arm movements can also be applied to articulator movements (speech), since the computational principle such as the minimum torque-change model must be independent of the controlled object, but inherent in the central nervous system itself. In speech synthesis, a long series of phonemes must be uttered continuously. From the trajectory formation standpoint, this implies that many via-points are specified for one continuous trajectory. Our cascade network can autonomously find the best time to pass through many via-points. This is one of the most attractive capabilities of our network. The cascade network can be applied not only to speech synthesis but also to continuous speech recognition, as first hypothesized in "motor theory of speech perception" (Liberman et al., 1967). The trajectory formation network can be used as an inverse system of trajectory formation, that is, as a continuous speech recognition system, by recurrently back-propagating an error between speech data and generated data. Application of the cascade network to speech synthesis and speech recognition is one of our important projects.

As mentioned earlier, there are two different approaches which resolve ill-posed motor control problems. One approach is to introduce a performance index. Another approach is to utilize a feedback controller. The feedback controller selects one specific motor command in the inverse dynamics and inverse kinematics problems even for redundant manipulators. The minimum jerk model formulated in the task-oriented coordinates can resolve ill-posed trajectory formation problems, but can not resolve ill-posed inverse kinematics and inverse dynamics problems for redundant manipulators. The feedback control approach can not resolve the ill-posed trajectory formation problem. Thus, a combination of these two approaches can resolve all three ill-posed problems. This is the step-by-step computational approach. This has been studied by many researchers including us (Hogan, 1984; Flash, 1987; Mussa-Ivaldi, Morasso, Zaccaria, 1988; Massone, Bizzi, 1989; Kano, Kawato, Suzuki, 1989).

We hypothesize that the cascade network (direct scheme) is used for very skilled movements, while step-by-step computation is utilized for relatively difficult or less skilled movements. That is, we suppose that the computational scheme adopted by the brain changes with motor learning. We have a few experimental data which seem to support this idea. First, in the human arm movement with the external spring force (Fig. 6 of Uno et al., 1989), subjects first tended to generate trajectories of various shapes at the beginning of the experiment when they were still not accustomed to the spring. After tens of repetitions, subjects began to consistently generate a curved hand path, which is the minimum torque-change trajectory (Uno, Kawato, Suzuki, 1989). Second, Uno et al. (unpublished observation) introduced nonlinear coordinates transformation between the hand position on a 2-dimensional position digitizer and the CRT coordinates where the end point, the start point and the hand position were displayed. Because of the nonlinear transformation, a straight line on the CRT corresponds to a curve on the digitizer, and vice versa. Subjects first generated roughly straight hand paths on the CRT. This is close to the minimum jerk trajectory in the visual task space (CRT coordinate). After several periods of

training, they tended to generate roughly straight hand paths on the digitizer (i.e. curved paths on the CRT), which are the minimum torque-change trajectories (Uno et al., unpublished observation). These experiment data could be explained if the step-by-step computation is taken over by direct computation with motor learning. In the first case, the forward dynamics model of an arm in combination with the spring must be relearned. In the second case, the forward kinematics model of an arm in combination with the imposed nonlinear transformation between the digitizer and the CRT must be relearned. Thus, the step-by-step computation seems to be temporarily utilized until the forward model is relearned. We do not think these observed changes of trajectories from the minimum jerk type to the minimum torque-change type with motor learning can be explained only by the combination of the minimum jerk trajectory planning and virtual trajectory control (Hogan, 1984; Flash, 1987) because this class of the step-by-step computational model has no capability of learning or adaptation with practice.

In section 5.2, we showed that the cascade neural network generated only fairly good torque waveforms on the minimum torque-change criterion. The estimated trajectory by the network was fairly close to the minimum torque-change trajectory and its end point is exactly the same as the desired end point (see Fig. 5..2b). However, the realized trajectory was considerably different from these two, and in particular the end point of the movement was different. The deviation of the end point is a serious problem. We can attain the desired end point even with an incomplete forward model by combining the original feedforward control with feedback control. We make use of the fact that the estimated trajectory reaches the desired end point. The estimated trajectory is used as a desired trajectory in the feedback control. Consequently, the torque fed to the controlled object is a summation of the feedforward torque generated by the neural network model and the feedback torque which is calculated in real time based on the error between the estimated desired trajectory by the network and the realized trajectory. We can infer two predictions from this composite feedforward and feedback control. First, if the learned forward model is inaccurate, the realized trajectory and the total torque fed to the controlled object would be quite jerky, especially around the end point near the movement end time. Second, conversely, as the forward model is improved, the role of the feedback control decreases, and hence the jerkiness of the movement is reduced. Schneider and Zernicke (1989) reported decrease of jerk cost during practice. This might be explained as improved control performance caused by an intensive learning of forward dynamics and kinematics of the arm for a special task. Some motor control schemes, such as the equilibrium trajectory approach (Hogan, 1984; Flash, 1987), do not support efficient movement refinement during practice. Finally, the composite feedforward and feedback control can cope with a sudden perturbation of movement whereas the pure feedforward control can not.

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

Acknowledgement

One of the author (M.K.) wishes to express his thanks to Drs. E. Yodogawa and K. Nakane of ATR Auditory and Visual Perception Research Laboratories for thier continuing encouragement.

References

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

[2] 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**, 270–277.

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

[4] Atkeson, C.G. and Reinkensmeyer, D.J.(1988). Using associative content-addressable memories to control robots. *Proc. of IEEE Conference on Decision and Control,* 792–797, Dec. 7-9, 1988, Austin, Texas.

[5] Barto, A.G. and Anandan, P.(1985). Pattern-recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-15**, 36—375.

[6] Bizzi, E., Accornero, N., Chapple, W. and Hogan, N.(1984). Posture control and trajectory formation during arm movement. *Journal of Neuroscience,* **4**, 2738–2744.

[7] Bryson, A.E. and Ho, Y.(1975). *Applied Optimal Control.* New York: Hemisphere.

[8] Bullock, D. and Grossberg, S.(1988). Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review*, **95**, 49–90.

[9] Eckmiller, R.(1988). Concept of a 4-joint machine with neural network control for the generation of two-dimensional trajectories, *Proc. of the First Annual INNS Meeting*, Boston (MA), 334.

[10] Flash, T.(1987). The control of hand equilibrium trajectories in multi-joint arm movements, *Biol. Cybern.*, **57**, 257–274.

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

[12] Hogan, N.(1984). An organizing principle for a class of voluntary movements. *Journal of Neuroscience*, **4**, 2745–2754.

[13] Hopfield, J.J. and Tank, T.W.(1985). "Neural" computation of decisions in optimization problems, *Biol. Cybern.,* **52**, 141–152.

[14] Jordan, M.I.(1986). Attractor dynamics and parallelism in a connectionist sequential machine, *Proc. 8th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Erlbaum, 531–546.

[15] Jordan, M.I.(1988). Supervised learning and systems with excess degrees of freedom, *COINS Technical Report 88-27*, 1–41.

[16] Jordan, M.I.(1989). Indeterminate motor skill learning problems. In M. Jeannerod (Ed.), *Attention and Performance, XIII.* Cambridge, MA: MIT Press. in press.

[17] 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, (in Japanese).

[18] 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.

[19] 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.

[20] 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.

[21] Kawato, M.(1989). Adaptation and learning in control of voluntary movement by the central nervous system, *Advanced Robotics.* **3**, No. 3.

[22] Kirkpatrick, S., Gellat, C.D., and Vecci, M.P.(1983). Optimization by simulated annealing, *Science*, **220**, 671–680.

[23] Koch, C., Marroquin, J. and Yuille, A.(1986). Analog "neuronal" networks in early vision, *Proc. Natl. Acad. Sci. USA,* **83**, 4263–4267.

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

[25] Liberman, A.M., Cooper, F.S., Shankweiler, D.P., and Studdert-Kennedy, M.(1967). Perception of the speech code. *Psychological Review*, **74**, 431–461.

[26] Maeda, Y., Kawato, M., Uno, Y. and Suzuki, R.(1988). Multi-layer neural network model which learns and generates human multi-joint arm trajectory. *Japan IEICE Technical Report,* **MBE87-133**, 233–240, (in Japanese).

[27] 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, (in Japanese).

[28] Maeda, Y.(1989). Multi-layer nerual network model which learns and controls human arm movement trajectory. *Osaka University Master's Thesis,* (in Japanese).

[29] Marr, D.(1982). *Vision.* New York: Freeman.

[30] Massone, L. and Bizzi, E.(1989). A neural network model for limb trajectory formation. *Biol. Cybern.* in press.

[31] 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.

[32] 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.

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

[34] Mussa-Ivaldi, F.A., Morasso, P. and Zaccaria, R. (1988). Kinematic networks - A distributed model for representing and regularizing motor redundancy, *Biol. Cybern.* **60**, 1–16.

[35] Poggio, T., Torre, V. and Koch, C.(1985). Computational vision and regularization theory, *Nature*, **317**, 314–319.

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

[37] Saltzman, E.L.(1979). Levels of sensorimotor representation. *J. Mathematical Psychology,* **20**, 91–163.

[38] Schneider, K., and Zernicke, R.F.(1989). Jerk-cost modulations during the practice of rapid arm movements. *Biol. Cybern.*, **60**, 221–230.

[39] 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, (in Japanese).

[40] Uno, Y., Kawato, M., Maeda, Y. and Suzuki, R.(1988). A neural network model for optimal control in human arm movement, *Proceedings of the 27th SICE Annual Conference*, **2**, 1053–1056.

[41] 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.

[42] Werbos, P.J.(1988). Generalization of backpropagation with application to a recurrent gas market model, *Neural Networks*, **1**, 339-356.

Appendix: Relationship of the cascade network with the first-order gradient algorithm

In this appendix, we use notations in Section 2. The problem stated in *Definition 1* can be solved by the following first-order gradient algorithm.

Let u denote the co-state $\dot{\tau}$. Then, the dynamics equation can be formally rewritten as follows, using a composite state variable $W = (\theta, \dot{\theta}, \tau)^T$:

$$dW/dt = A(\theta, \dot{\theta}, \tau, u), \tag{6.1}$$

here, W is a $(2n+m)$-dimensional vector, and A is a $(2n+m)$-dimensional nonlinear function, and is defined as $(\dot{\theta}, f(\theta, \dot{\theta}, \tau), u)^T$. The adjoint equation of this equation is given as:

$$d\Psi/dt = -(\partial A/\partial W)^T \Psi. \tag{6.2}$$

Ψ is also a $(2n+m)$-dimensional vector. The first n component, the second n component and the last m component of Ψ are denoted as Ψ_θ, $\Psi_{\dot{\theta}}$, and Ψ_τ, since they correspond to θ, $\dot{\theta}$, and τ. The first-order gradient algorithm, a numerical method in optimal control theory is described as follows (Bryson and Ho, 1975).
(step 1) Estimate an initial control $u_0(t)$.
(step 2) Integrate the system eq. 6.1 with the specified initial conditions $W(0)$ and the specified control variables history. Record $W(t)$.
(step 3) Backward integration of adjoint equation 6.2 to determine $\Psi(t)$, with the specified control $u(t)$, the trajectory $W(t)$ and the terminal condition $\Psi(t_f) = W_d(t_f) - W(t_f)$. Here, $W_d(t_f)$ are desirable terminal conditions for the trajectory.
(step 4) Modification of the control according to the steepest descent of the Hamiltonian:

$$\delta u(t) = \varepsilon(-\lambda u + \Psi_\tau(t)). \tag{6.3}$$

(step 5) Repeat steps 1 through 4, using an improved estimate of $u(t)$ until the error is admissible.

It is clear that the forward calculation through the cascade network is conducting the step 2. We will show that the backward propagation of errors through the cascade structure is partially equivalent to the step 3. The adjoint equation can be rewritten in components as follows.

$$d\Psi_\theta/dt = -(\partial f/\partial\theta)^T \Psi_{\dot{\theta}} \tag{6.4}$$

$$d\Psi_{\dot{\theta}}/dt = -\Psi_\theta - (\partial f/\partial\dot{\theta})^T \Psi_{\dot{\theta}} \tag{6.5}$$

$$d\Psi_\tau/dt = -(\partial f/\partial\tau)^T \Psi_{\dot{\theta}}. \tag{6.6}$$

It is not difficult to see that the backward numerical integration of the first two equations is equivalent to the error back-propagation through the cascade network based on eq. 3.12. This is because the partial derivatives $(\partial f/\partial\theta)$, $(\partial f/\partial\dot{\theta})$ and $(\partial f/\partial\tau)$ are calculated by back-propagation within each network unit. The reason that the network can calculate exact Ψ_θ and $\Psi_{\dot{\theta}}$ although it does not contain either the co-state u or the corresponding Ψ_τ is that the adjoint equations do not contain

Ψ_τ in the right side. Because the network does not contain the co-state u, it cannot use equation 6.3 and hence needs to introduce direct interactions between the motor commands. This is the box diffusion interaction (electrical couplings) between control variables.