



HOKKAIDO UNIVERSITY

Title	Design of Recurrent Neural Networks for Dynamical Robot Control
Author(s)	Kumagai, Toru
Degree Grantor	北海道大学
Degree Name	博士(工学)
Dissertation Number	甲第4148号
Issue Date	1997-03-25
DOI	https://doi.org/10.11501/3122306
Doc URL	https://hdl.handle.net/2115/51425
Type	doctoral thesis
File Information	000000307497.pdf



Design of
Recurrent Neural Networks
for Dynamical Robot Control

by
Toru Kumagai

①

Contents

Design of Recurrent Neural Networks for Dynamical Robot Control

by
Toru Kumagai

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Engineering
Hokkaido University

December 1996

Contents

Contents	i
List of Figures	iii
 Chapter 1	
Introduction	1
1.1 Potential of Neural Networks	1
1.2 Neural Network Control of Dynamical Robots	3
1.3 What is Described in This Thesis	3
References of Chapter 1	8
 Chapter 2	
Dividing Control for Dynamical Robot System	11
2.1 Introduction	11
2.2 Dividing Control	12
2.3 Dividing Control of an Inverted Pendulum	16
2.3.1 Conventional Learning Control	16
2.3.2 Inverted Pendulum System	16
2.3.3 Dividing of an Inverted Pendulum	19
2.3.4 Applying a Learning Control System	20
2.3.5 Experimental Results	24
2.3.6 Discussions	25
2.4 Control Problem of an Active Mass Damper	27
2.4.1 Active Mass Damper	28
2.4.2 Dividing of a Control System	30
2.4.3 Introducing Learning Controller and Reference Model	31
2.4.4 Results of Experiments	34
2.4.5 Discussions	40
2.5 Conclusions	42
References of Chapter 2	43
 Chapter 3	
Analytical Design of Recurrent Neural Networks	46
3.1 Introduction	46

3.2 Recurrent Neural Network	47
3.2.1 Neural Network Model	47
3.2.2 Dynamics	50
3.2.3 Designing Algorithm	55
3.2.4 Condition for Oscillating a Certain limit Cycle	58
3.2.5 Condition for Converging in a Certain Limit Cycle	60
3.3 Potential of the Network	61
3.3.1 Capabilities	61
3.3.2 Performance	68
3.4 Conclusions	68
References of Chapter 3	70

Chapter 4

Learning of Recurrent Networks Using Genetic Algorithm	72
4.1 Introduction	72
4.2 Genetic Algorithm for Recurrent Neural Networks	74
4.2.1 Recurrent Neural Networks	74
4.2.2 Coding of the Gene	74
4.2.3 Genetic Algorithm with Copy Operators	76
4.3 Application of GA with Copy Operators	80
4.3.1 Excitation of Dynamical Systems	80
4.3.2 Learning Limit Cycles in a Network	89
4.3.3 Structures Acquired by Copy Operators	97
4.3.4 Swinging up of an Inverted Pendulum	97
4.3.5 Application to Large Scale Problems	101
4.4 Conclusions	105
References of Chapter 4	106

Chapter 5

Summary	107
----------------------	------------

Acknowledgments	114
List of the Results	115

List of Figures

Fig. 2.2.1	Structure of the plant.....	15
Fig. 2.2.2	Structure of the plant (After decomposition)	15
Fig. 2.2.3	The block diagram of a controller designed by dividing control....	15
Fig. 2.3.1	An inverted pendulum system.....	18
Fig. 2.3.2	The block diagram of a experimental system	18
Fig. 2.3.3	The block diagram of a control system.....	19
Fig. 2.3.4	The learning control system using a neural network	22
Fig. 2.3.5	Responses of the pendulum.....	22
Fig. 2.3.6	Responses of the pendulum.....	26
Fig. 2.3.7	Responses of the inverted pendulum system	26
Fig. 2.4.1	An elastic arm with an active mass damper	29
Fig. 2.4.2	A second degree of freedom vibration system.....	30
Fig. 2.4.3	The block diagram of the controller designed by dividing control	31
Fig. 2.4.4	The learning controller using a neural network	34
Fig. 2.4.5	Responses of the elastic arm with AMD	35
Fig. 2.4.6	Responses of the elastic arm with AMD	36
Fig. 2.4.7	Responses of the experimental system	37
Fig. 2.4.8	Responses of the experimental system controlled by three networks	38
Fig. 2.4.9	Maximum amplitude of x_1	40
Fig. 2.4.10	Responses of the experimental system controlled by a linear quadratic controller	40
Fig. 2.4.11	Responses of the plant when k_1 fluctuates.....	41
Fig. 2.4.12	Responses of the plant when c_2 fluctuates	41
Fig. 3.2.1	The neuron model	49
Fig. 3.3.2	A neural network with two neurons.....	63
Fig. 3.3.3	The map of the initial state.....	65
Fig. 3.3.4	The map of the initial state.....	66
Fig. 3.3.5	Transition of limit cycles evoked by external input.....	67
Fig. 4.2.1	The coding of the gene.....	75
Fig. 4.2.2	The genetic algorithm with copy operator	78
Fig. 4.2.3	Genetic algorithm operators.....	79

Fig. 4.3.1	An excitation problem of a vibration system.....	81
Fig. 4.3.2	Evaluation values after training	84
Fig. 4.3.3	Response of a network.....	85
Fig. 4.3.4	Response of a network	85
Fig. 4.3.5	Role of each operator in evolution process	86
Fig. 4.3.6	Role of each operator in evolution process	86
Fig. 4.3.7	Response and network structure just after the start of learning	87
Fig. 4.3.8	Response and network structure in the beginning of leaning	87
Fig. 4.3.9	Response and network structure in the middle period	88
Fig. 4.3.10	Response and network structure at the end of learning	88
Fig. 4.3.11	An excitation of two dynamical systems	90
Fig. 4.3.12	A limit cycle acquired in a network.....	92
Fig. 4.3.13	A limit cycle acquired in a network.....	92
Fig. 4.3.14	Time progress of evaluation values through learning	93
Fig. 4.3.15	Evaluation Values at 400th generation	93
Fig. 4.3.16	Time course of evaluation values of a lineage when mutation and internal copy operator are applied.....	94
Fig. 4.3.17	Time course of evaluation values of a lineage when mutation and inter individual copy operators are applied.....	94
Fig. 4.3.18	Time course of evaluation values of a lineage when mutation and cross over operators are applied.....	95
Fig. 4.3.19	Time course of a evaluation value of a lineage when mutation is applied.....	95
Fig. 4.3.20	The network structure acquired by mutation	96
Fig. 4.3.21	The network structure acquired by mutation and internal-copy	96
Fig. 4.3.22	An inverted pendulum system.....	98
Fig. 4.3.23	Response of a pendulum	100
Fig. 4.3.24	A process of swinging up.....	100
Fig. 4.3.25	Response of a recurrent network.....	103
Fig. 4.3.26	Time course of evaluation values.....	103
Fig. 4.3.27	Response of a recurrent network.....	104

Chapter 1

Introduction

1.1 Potential of Neural Networks

This thesis describes about a design method of recurrent neural networks for dynamical robot control. Recurrent neural networks are excellent to discriminate, transform, memorize and recall temporal data. They would be useful to realize dynamical control of robots in the real world. However there is no powerful enough designing method for them.

Neural networks are a technology of parallel computation in which the computing elements are artificial neurons based on simple properties of biological neurons. They are roughly classified into recurrent neural networks which have some feedback connections and feedforward networks. Feedforward networks have been applied to many applications in comparison to recurrent neural networks, because feedforward networks have several universal learning algorithms such as backpropagation algorithm[1]. However feedforward neural networks can memorize only static mapping between its input and output, and can not deal with temporal data.

Contrary, recurrent neural networks that have feedback connections can represent dynamics. One feature of recurrent networks is its relaxation or

convergent dynamics where networks tend to settle in stationary points. This aspect of recurrent networks became popular through *Hopfield network*[2]. Hopfield network is a network which has symmetric weights, no self-connection and asynchronous update. It has dynamics and can memorize attractors. Hopfield network can solve optimization problems based on the energy function. Original Hopfield network consists of time discrete binary output neurons. It was extended to a discrete time continuous function model[3] and a continuous time continuous function model[4]. However, these networks can not memorize temporal sequences such as limit cycles.

Other important feature of recurrent networks is their non-convergent dynamics. After backpropagation algorithm was proposed, a layered network was extended to several kinds of recurrent networks for temporal data [1][5][6][7][8][9]. In recurrent network, a change of weights and external signals can influence future behavior of the networks. Learning algorithm that takes into account this effect has been proposed. For example, Real Time Recurrent Learning (RTRL) [9] and Backpropagation Through Time (BPTT) [1] were derived for fully connected recurrent networks. However, their learning convergence is suffering from local minimum problem more than feedforward networks, because these algorithms propagate an error through time, and the error gradient can decay or expand exponentially in time. Moreover convergence of learning depends critically on the choice of network topology, initial weights, and training samples.

The other stream of neural network studies is modeling of biological system. Many models of biological networks have been proposed and their dynamical characteristics were analyzed. For example models of central pattern generators[10][11], olfactory bulb[12][13][14], olfactory cortex[15][16] and respiratory pattern generator[17] have been studied. In contrast to engineering recurrent networks described in the above, biological neural networks have unsymmetrical structures and incorporate significant delays as a result of axonal

propagation. It is supposed that they have high computational capability beyond simple recurrent networks which were extended from layered neural networks. Therefore many kinds of biological neuron models have been studied[18][19][20]. Most of these are still too complex for constructing networks. One of the simplest biological inspired neuron is the leaky integrator model such as Caianiello-Nagumo neuron model. These models are simple enough for the engineering purpose and yet the dynamics of these neurons show high computational capability. However, networks consist of these models have not been analyzed enough.

1.2 Neural Network Control of Dynamical Robots

Reflecting lack of a designing method of recurrent neural networks, most contributions of neural networks for dynamical robot control have been based on layered neural networks. Kawato et al.[21][22][23] made use of feedback error learning for forward control and regulating problems. Jordan et al.[24][25] proposed the scheme of combining a forward model and an inverse model. In these algorithms, a layered neural network acquires inverse dynamics of a controlled object and is used as a regulator. Recurrent networks extended from layered neural networks have been applied to system identify[26] or control[27][28][29]. In these researches, recurrent networks play the role of feedforward model or inverse model again.

However, what is required in a controller is not only regulation, but also realization of a series of dynamical movement and an information processing of temporal data. The establishment of a design method of dynamic recurrent neural networks is one of the key to realize such kinds of dynamic controller.

1.3 What is Described in This Thesis

Based on the above consideration, this thesis describes a design method of recurrent networks for dynamical robot control. The contents of remaining chapters are summarized as follows.

In chapter 2, a new design method of a neural network controller, dividing control is described. Conventional learning methods are not available for a multi output plant, when the exact model of a plant is unknown. To solve this problem, dividing control is introduced. Dividing control makes possible learning control of a multi output system. The contents of this chapter are the following.

1. Dividing control is introduced.

- Dividing control is a control method to divide a plant to single input and single output sub-plants and apply a neural network controller or other control systems to each sub-plant.

2. Dividing control is applied to an inverted pendulum system.

- Through dividing control, an appropriate nonlinear controller is acquired in neural networks for an inverted pendulum without the exact model of a plant.
- It is shown that the neural network learning controller with reference model that is applied to each sub-plant has high robustness.

3. Dividing control is applied to an active mass damper.

- Through dividing control, an appropriate controller is acquired for an active mass damper.
- The controller acquired in a network satisfies non-linear restriction of an active mass damper. Performance of the acquired controller is superior to a quadratic linear optimal controller.

As so far, dividing control makes possible non-linear learning control of a multi output system. Dividing control extends the application domain of a neural network learning controller to multi output systems where plural sub systems interfere with each other.

However, it is sure that layered networks can not deal with temporal or

context sensitive data directly. Therefore in the remaining part of this thesis, designing methods of recurrent network are described.

In chapter 3, a time discrete binary output recurrent neural network model which consists of Caianiello-Nagumo neuron model is introduced. It is known that recurrent networks have great capability. However, for dynamic recurrent neural networks which consist of realistic neurons have not been analyzed enough because of mathematical difficulty. Here, the recurrent neural network which consists of simple realistic neuron, Caianiello-Nagumo neuron, is proposed. Its designing algorithm and dynamics are described. Also the convergence area of certain limit cycles is decided theoretically. The contents of this chapter are the following.

1. A discrete time binary recurrent neural network is introduced. The network consists of Caianiello-Nagumo neurons which have leaky integration term.

2. Dynamics of the network is analyzed and the following conditions and algorithms are derived.

- The condition for oscillating a certain limit cycle is derived. It is concluded that the proposed network tends to stay a limit cycle.
- The condition for memorizing a set of limit cycles in a network is derived. The algorithm that decides a set of parameters for memorizing certain limit cycles in a network is derived.
- The condition for convergence in a certain limit cycle is derived. The algorithm that decides initial states where the network converges to certain limit cycle is derived. The condition for evoking a transfer between limit cycles by external input is derived.

3. The performance of the proposed network is discussed.

- Several sets of limit cycles are stored analytically in the proposed network. It is demonstrated that a network classifies a couple of context sensitive limit cycles. The external input which evokes a transfer between

limit cycles is also designed by the proposed method.

- It is shown that the network can memorize only limited oscillation patterns when the period of limit cycles is more than 3. This drawback increases when the period or the number of limit cycles increases. It is concluded that to memorize a set of arbitrary limit cycles, the network requires hidden neurons, and a designer must decide behavior of hidden neurons.

As so far, it is shown that the proposed network has capability to deal with context sensitive dynamics. The theoretical designing algorithm is derived. It is also shown that the theoretical algorithm is not available for embedding of complex limit cycles. It is clear that networks require hidden neurons for complex limit cycles and no method is established to decide behavior of hidden neurons. Therefore, in next chapter, learning of recurrent networks using genetic algorithms is described.

In chapter 4, a genetic algorithm is introduced for learning of the recurrent network proposed in chapter 3. To accelerate performance of genetic algorithm, the *internal copy operator* and *inter-individual copy operator* are introduced. The effectiveness of the proposed algorithm is shown in some applications.

1. Internal and inter-individual copy operators are introduced.

- Internal copy operator is a genetic algorithm operator which copies a portion of gene to an other part. Inter-individual copy operator copies a portion of gene to the other part of other genes.

2. The proposed method is applied to an excitation problem of a vibration system.

- Learning of the recurrent network is accelerated by introducing copy operators. In the beginning of learning, mutation operator is dominant.

Next, copy operator is dominant. At last mutation operator is dominant. It is concluded that copy operator copies functional parts made by mutation operator.

3. The proposed method is applied to the excitation problem of a set of vibration system.

- The goal of this task is to excite two vibration systems which are connected to the recurrent network alternatively. This problem has severe local minimum area. Nevertheless, the quasi optimal solution is found by the proposed method. The network hardly can memorize two limit cycles without internal copy operator.

- The network acquires a peculiar structure by internal copy operator. It is supposed that this structure accelerates learning.

4. The proposed method is applied to a swing up problem.

- The goal of this task is swinging up a pendulum, and keep it on top position. The network acquires an appropriate control rule for these two behaviors.

5. The proposed method is applied to the large scale problem.

- The proposed method is applied to the learning problem of recurrent networks which consists of 128 neurons. The goal of task is to excite a vibration system driven by recurrent networks. After enough learning, the network acquires an appropriate control rule. All neurons work cooperatively.

As so far, it is shown that genetic algorithm with internal copy accelerates learning of recurrent networks for dynamical control tasks. Recurrent networks can acquire a set of limit cycles, a set of behavior and a peculiar structure through genetic algorithm learning.

In chapter 5, the results of this thesis are summarized.

References of Chapter 1

- [1]D.E.Rumelhart, G.E.Hinton and R.J.Williams, Learning internal representations by error propagation, Parallel Distributed Processing Volume 1, MIT Press,pp.318-362(1986).
- [2]J.Hopfield, Neural networks and physical systems with emergent collective computational properties, Proc. Natl. Acad. Sci. USA, vol.79,pp.2254-2558(1982).
- [3]R.M.Golden, The brain state in a box neural model is a gradient descent algorithm, J.Math. Psych., vol.30, pp.73-80(1986).
- [4]M.Cohen and S.Grossberg, Absolute stability of global pattern formation and parallel memory storage by competitive neural network, IEEE Trans. Sys. Man Cyber., SMC-13, pp.815-826(1983).
- [5]M.I.Jordan, Attractor dynamics and parallelism in a connectionist sequential machine, in Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Hillsdale, NJ:Erlbaum, pp.531-546(1986).
- [6]J.L.Elman, finding structure in time, Cognit. Sci., vol.14,pp.179-211(1990).
- [7]R.J.Williams, and D.Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Computat., vol.1,pp.270-280(1989).
- [8]B.A.Pearlmutter, Learning stat space trajectories in recurrent neural networks, Neural Computat., vol.1, pp.263-369(1989).
- [9]K.Doya, S.Yoshizawa, Adaptive neural oscillator using continuous-time back-propagation learning, Neural Netw., vol.2, pp.375-386(1989).
- [10]N.Kopell, Toward a theory of modelling central pattern generators, in Neural Control of Rhythmic Movement in Vertebrates(A.H.Cohen, S.Grillner, and

S.Rossignol, Eds.), New York:Wiley, pp.369-413(1987).

[11]W.O.Friesen, and G.S.Stent, Generation of a locomotory rhythm by a neural network with recurrent cyclic inhibition, *Biol. Cybern.*, vol.28,pp.27-40(1977).

[12]D.Kleinfeld, K.R.Delaney, M.S.Fee, J.A.Flores, D.W.Tank and A.Gelperin, Dynamics of propagating waves in the olfactory network of terrestrial mollusc: An electrical and optical study, *J.Neurophysiol.*, vol.72,pp.1402-1409(1994).

[13]Z.Li and J.J.Hopfield, Modeling the olfactory bulb and its neural oscillatory processing, *Biol. Cybern.*, vol.61, pp.379-392(1991).

[14]J.S.Kauer, Contributions of topography and parallel processing to odor coding in the vertebrate olfactory pathway, *Trends Neurosci.*, vol.14, pp.79-85(1991).

[15]W.J.Freeman, Simulation of chaotic EEG patterns with a dynamic model of the olfactory system, *Biol. Cybern.*, vol.56,pp.139-150(1987).

[16]J.Ambros-Ingerson, R.Granger, and G.Lynch, Simulation of paleocortex performs hierarchical clustering, *Science*, vol.247,pp.1344-1348(1990).

[17]S.M.Botros and E.N.Bruce, Neural network implementation of a three-phase model of respiratory rhythm generation. *Biol. Cybern.*, vol.63:pp.143-153(1993).

[18]AL.Hodgkin, AF.Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol(London)* vol.117,pp.500-544(1952).

[19]E.R.Caianiello and A.D.Luca, Decision Equation for Binary Systems. Application to Neuronal Behavior, *Kybernetik*, vol.3, pp.33-40(1966).

[20]B.Cartling, Response characteristics of a low-dimensional model neuron, *Neural Comput*, vol.8, pp.1643-52(1996).

[21]M.Kawato, K.Furukawa and R.Suzuki, A hierarchical neural network model

for control and learning of voluntary movement, *Biological Cybernetics*, vol.57, pp.169-185(1987).

[22]M.Kawato: Computational Schemes and Neural network Models for Formation and Control of Multi-joint Arm Trajectory, *Neural Network for Control*, The MIT Press(1990).

[23] M.Kawato et al: A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement. *Biol.Cybern.*57,169-185(1987).

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

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

[26]J.Liang, P.N.Nikiforuk, M.M.Gupta, Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks, *IEEE Trans. Autom. Control*, vol.40,pp.1266-70(1995).

[27]I.Kamwa, R.Grondin, V.K.Sood, C.Gagnon et al., Recurrent neural networks for phasor detection and adaptive identification in power system control and protection.*IEEE Trans. Instrum. Meas*, vol.45, pp.657-64(1996).

[28]K.Y.Chao-Chee Ku Lee, Diagonal recurrent neural networks for dynamic systems control, *IEEE Trans. Neural Netw.*, vol.6,pp.144-156(1995).

[29]K.P.Venugopa, A.S.Pandya, R.Sudhakar, A recurrent neural network controller and learning algorithm for the on-line learning control of autonomous underwater vehicles, *Neural Netw.*, vol.7,pp.833-46(1994).

Chapter 2

Dividing Control for Dynamical Robot System

2.1 Introduction

Several kinds of learning controllers using a layered neural network have been proposed. However, most of them can not be applied to complex problems such as a multi output problem (This nature of neural network learning controllers will be discussed in section 2.3.1 through the control problem of an inverted pendulum).

On the other hand, decomposition of a large problem is a general idea to solve a large problem in daily situations. This concept is also important from the engineering point of view. For example, modular architecture has been studied eagerly for neural network learning of complex functions[1][2][3]. Decomposition is also effective for analytical design of a dynamical control system. Decentralized and hierarchical control have been studied to simplify design of a linear control system[4][5]. Backstepping method, where a plant is divided into subsystems,

makes possible design of an adaptive control system without augmented errors[6][7].

In this chapter, dividing control is introduced to solve a multi output learning control problem[8][9]. The proposed design method utilizes the concept of decomposition. This method consists of two designing steps. In the first step, a complex plant is divided into single input single output sub-plants. In the second step, a learning controller is applied to each sub-plant.

This chapter is structured as follows. Dividing control is introduced in section 2.2. In section 2.3 and 2.4, dividing control is applied to the stabilization problem of an inverted problem and the control problem of an active mass damper. It is shown that a structure introduced by the dividing process makes possible learning control of a multi output system.

2.2 Dividing Control

In this section, dividing control is introduced through an application to a linear system. A linear system which has one input m outputs is described as follows.

$$\begin{aligned}\dot{\mathbf{x}}_i &= \mathbf{A}_i \mathbf{x}_i + \mathbf{B}_i u + \sum_{j=1, j \neq i}^m \mathbf{z}_{i,j} \mathbf{x}_j \\ y_i &= \mathbf{C} \mathbf{x}_i \quad (i = 1, 2, \dots, m)\end{aligned}\tag{2.2.1}$$

where u is the manipulated value, $\mathbf{x}_i$ is the state vector of sub-system i , y_i is output of sub-system i , $\mathbf{z}_{i,j}$ is the matrix that decides interference between system i and j . The above equation is rewritten in the following expression while assuming each subsystem is first order lag time system and $\mathbf{z}_{i,j}$ is negligible for simplicity.

$$\begin{aligned}m_i \dot{x}_i &= k_i x_i + u \quad (|m_i| < |m_j| \mid i < j) \\ y_i &= x_i \quad (i = 1, 2, \dots, m)\end{aligned}\tag{2.2.2}$$

The structure of equation (2.2.2) is shown in Figure 2.2.1. In this expression, the manipulated value u interferes with all sub systems. Equation (2.2.3) is derived from equation (2.2.2).

$$m_1 \dot{x}_1 = k_1 x_1 + u \quad y_1 = x_1 \quad (2.2.3)(a)$$

$$m_1 \dot{x}_i = k_i x_i + (m_{i-1} \dot{x}_{i-1} - k_{i-1} x_{i-1})$$

$$y_i = x_i \quad (i = 2, 3, \dots, m) \quad (2.2.3)(b)$$

In equation (2.2.3), the sub-system 1, equation (2.2.3)(a), is a one input one output system whose input is u . The sub-system i , equation (2.2.3)(b), is a one input one output system whose input is output of the system $(i-1)$. Now, each sub-system is independent each other except for connections between adjacent systems (Figure 2.2.2). Equation (2.2.5) is derived when the following linear feed back, equation (2.2.4), is applied to equation (2.2.3) (Figure 2.2.3).

$$u = -k_1(x_1 - x_r)$$

$$x_{(i-1)r} = k_i(x_i - x_{ir}) \quad (i = 2, 3, \dots, m-1)$$

$$x_{mr} = 0 \quad (2.2.4)$$

$$u = \sum_{i=1}^m \left(x_i \prod_{j=1}^m (-1) k_j \right) \quad (2.2.5)$$

Equation (2.2.6) is derived from equation (2.2.5) by change of variables.

$$u = \sum_{i=1}^m (K_i x_i) \quad (2.2.6)$$

It means that applying a linear feedback controller to each sub-system is equivalent to a linear state feedback applied to an original system, equation

(2.2.1). Then let's think to apply a controller using a layered neural network to equation (2.2.3), instead of a linear controller, equation (2.3.4). If the subsystem i is controlled with time constant faster enough than the sub-system $(i + 1)$, the whole system will be controlled by training controllers one by one, from sub-system 1 to sub-system m . Neural network controllers can cancel non linearity and interference between subsystems, which was neglected in equation (2.2.2), through learning.

Dividing control has two remained problems. First dividing control requires some information to decompose an original plant. There is no general way to decide how to divide. Second problem is how to design each control system. One solution is to design the controller from a priori information. This method is used in the 2.3 for the control problem of an active mass damper.

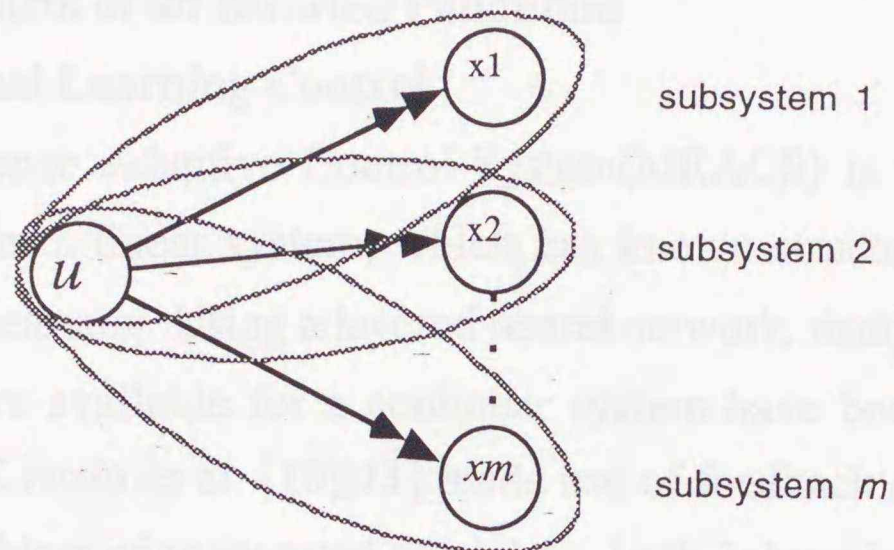


Figure 2.2.1 Structure of the plant

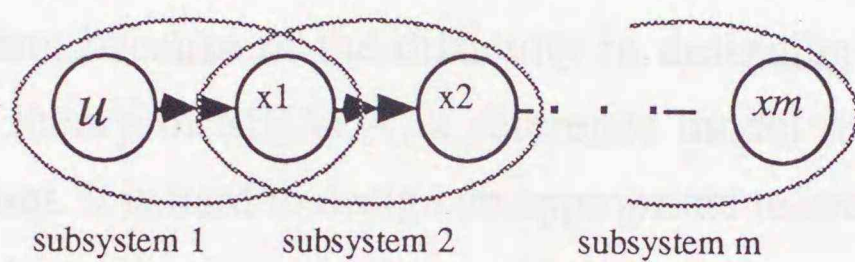


Figure 2.2.2 Structure of the plant (After decomposition)

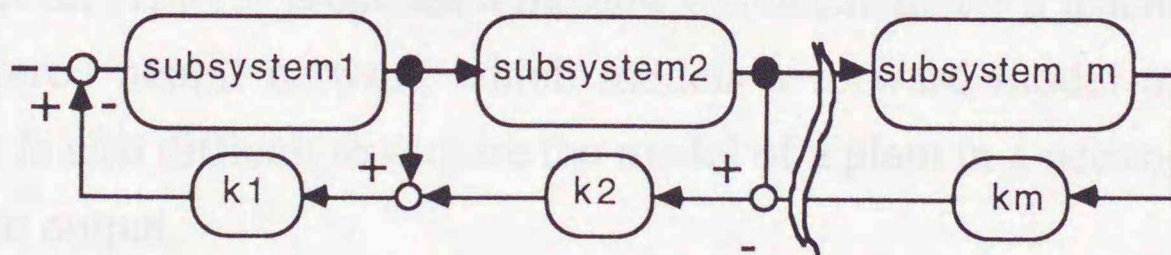


Figure 2.2.3 The block diagram of a controller designed by dividing control

2.3 Dividing Control of an Inverted Pendulum

2.3.1 Conventional Learning Control

Model Reference Adaptive Control System(MRACS) is noted as a good control method for a linear system, which has known structure and unknown parameters, with learning. Using a layered neural network, many learning control systems which are available for a nonlinear system have been proposed. For instance, Gomi, Kawato et al. [10][11] made use of feedback error learning for the regulating problem of an inverted pendulum. In their learning control system, a feed back controller acts as a reference model which is used to calculate a teaching signal.

It is difficult, however, to use these control methods for a multi output or higher order system, because of the difficulty in designing reference models. According to the theory of MRACA, a reference model must have the same structure with a plant. It is hard to design an appropriate reference model without parameters of a plant when a plant has complex structure, i.e., a multi input output or higher order structure. Furthermore it requires all state variables to calculate a teaching signal.

Jordan et al. [12][13] proposed a method which calculates a teaching signal using a layered neural network which models a forward model of a plant. However, it is also difficult to acquire the model of a plant in a network when a plant is multi output.

Here, dividing control is applied to the regulation problem of an inverted pendulum, which is a one-input two-output system. An inverted pendulum system is regulated in both simulation and experimental system. Moreover, it is shown that a reference model of order lower than a plant order is available.

2.3.2 Inverted Pendulum System

Figure 2.3.1 illustrates the regulating problem of an inverted pendulum. Input of the system is force applied to the cart. Output is the angle of a pendulum and the position of a cart. Following is a model of the experimental system of an inverted pendulum including the motor system that drives the cart.

$$M\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta + c\dot{x} = u \quad (2.3.1)$$

$$mL\ddot{x}\cos\theta + J\ddot{\theta} + p\dot{\theta} - mgL\sin\theta = 0 \quad (2.3.2)$$

Where θ is angle of the pendulum, p is viscosity coefficient of the pendulum, J is inertia moment of the pendulum, round the rotation axis, m is mass of the pendulum, L is length between center mass of the pendulum and the rotation axis, c is viscosity coefficient of the cart, M is sum of the mass of the pendulum and the cart, u is input which is given to the drive system of the cart. c and M include the viscosity and the inertial mass of the drive system of the cart.

Figure 2.3.2 is a block diagram of the experimental system. The velocity of the cart maintains a fixed relation to input of the driver. Manipulated signal F is integrated before being transmitted to the driver. Because of that the manipulated variable corresponds to the acceleration of the cart. Parameters of the system (2.3.1) and (2.3.2) are as follows.

$$\begin{aligned} m &: 0.015[\text{kg}] & L &: 0.12[\text{m}] \\ J &: 2.58 \times 10^{-4}[\text{kg} \cdot \text{m}^2] & p &: 0.62 \times 10^{-4}[\text{kg} \cdot \text{m}^2/\text{s}] \\ M &: 0.625[\text{kg}] & C &: 13.6[\text{kg/s}] \\ U(t) &: \int F(t) \cdot dt \end{aligned}$$

In the experimental system, θ and x are measured by encoder and $\dot{x}$ and $\dot{\theta}$ are determined by numerical differentiation. Moving average method is used

with previous three θ to compensate for lack of accuracy of the encoder. The sampling time, that will be noted by δt , is 20ms. Simulation of neural networks and control of the experimental system is performed by a personal computer (cpu=80386, fpu=80387, cpu clock=16MHz). In the simulation, an inverted pendulum is simulated by Runge-Kutta-Gill formulas with the above parameters for the experimental system.

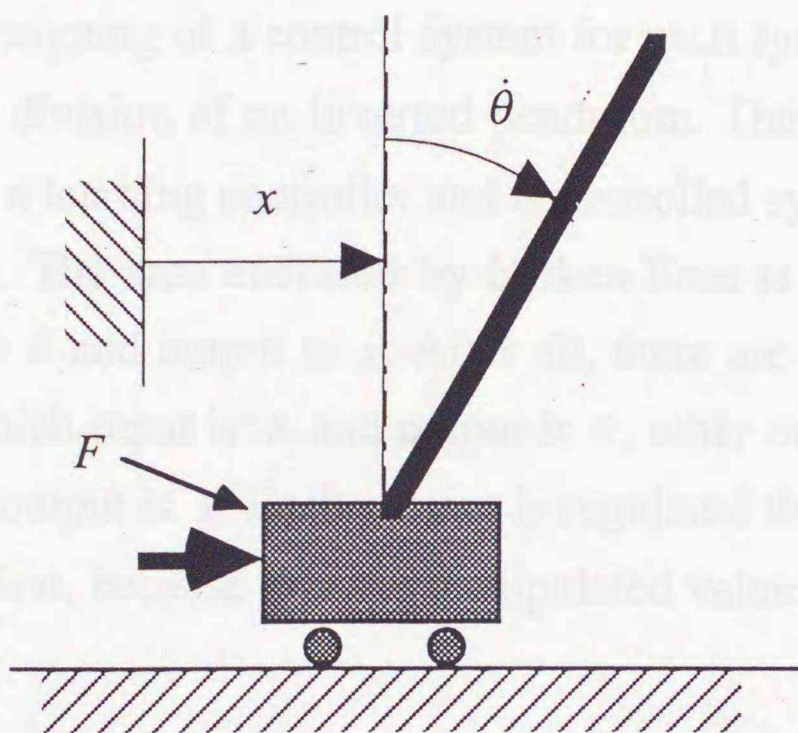


Figure2.3.1 An inverted pendulum system

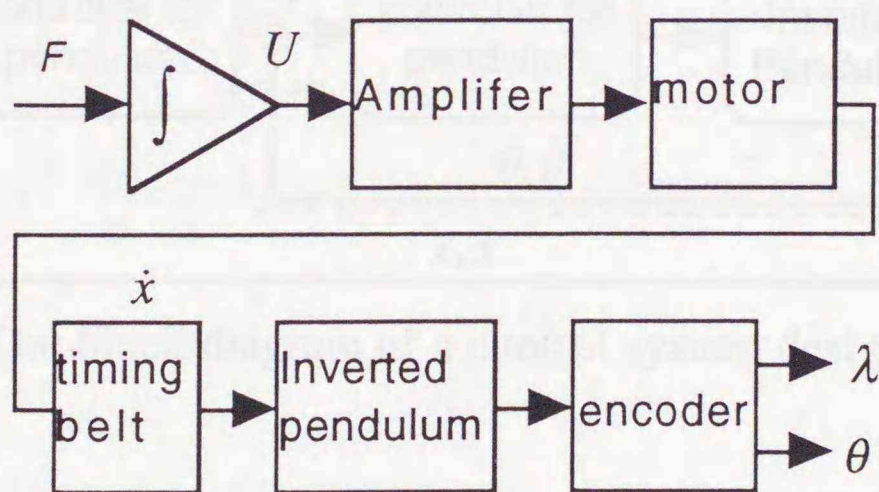


Figure2.3.2 The block diagram of an experimental system

2.3.3 Dividing of an Inverted Pendulum

An inverted pendulum is a one input two output system, i.e., the manipulated variable is F and the controlled variables are x and θ . It is difficult therefore to design a reference model for an inverted pendulum.

Lets' apply dividing control to an inverted pendulum. Dividing control consists of two design steps: dividing of the inverted pendulum into two one input one output systems and designing of a control system for each system. Figure 2.3.3 shows the method of division of an inverted pendulum. The area enclosed by broken lines contains a learning controller and a controlled system where input is F and output is θ . The area enclosed by broken lines is also a controlled system where input is θ and output is x . After all, there are two systems, one being a system for which input is F and output is θ , other one is a system for which input is θ and output is x . Each system is regulated through learning. θ must be regulated at first, because θ is the manipulated value when controlling x .

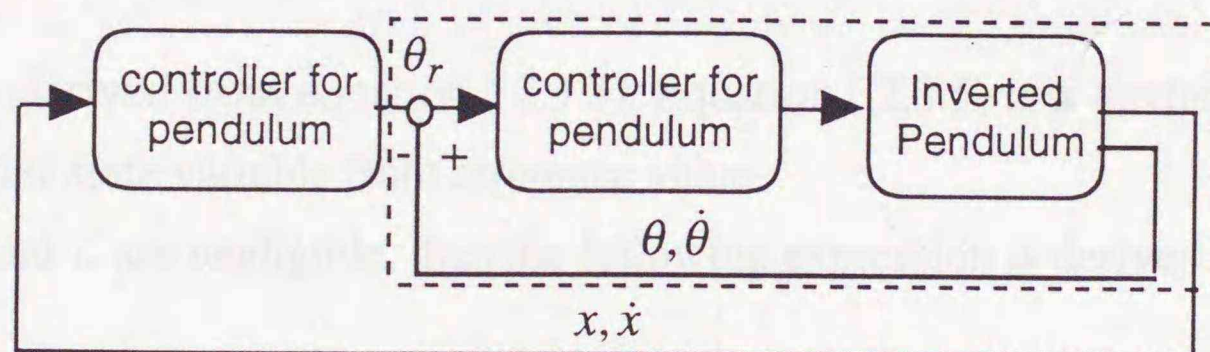


Figure2.3.3 The block diagram of a control system designed by dividing control

2.3.4 Applying Learning Control System

(a) Regulation of the Angle of a Pendulum

First, θ is regulated with a learning control system as shown in Figure 2.3.4. The system where input is F and output is x is a second order lag system if neglecting the reaction of the pendulum and delay of the cart response from manipulated signal. Therefore second order lag reference model is used.

$$\ddot{\theta}_M + k_{M1}\dot{\theta}_M + k_{M2}\theta_M = 0 \quad (2.3.3)$$

The learning controller consists a 3 layered neural network that contains 10 neuron in the hidden layer. Inputs of the network are θ , $\dot{\theta}$. Output of the network is u which is the manipulated signal determines F . Network is trained by backpropagation algorithm. The teaching signal used in learning is shown below.

$$T(t) = u(t) + \delta F \operatorname{sgn}(e(t + \delta t)) \quad (2.3.4)$$

$$e(t) = \ddot{\theta} - \ddot{\theta}_M \quad (2.3.5)$$

$\ddot{\theta}_M$ is derived from equation (2.3.3). Equation (2.3.5) is a deviation of the highest order state variable from reference value.

If p and c are negligible, then the following expression is derived.

$$J\ddot{\theta}^2 = -u(t)M + mgL\theta$$

and assuming that θ is constant then

$$\ddot{\theta} \propto u(t) \quad (2.3.6)$$

Therefore learning through teaching signal (2.3.4) always decreases error value (2.3.5). In fact, the angle of the pendulum is regulated by reference model (2.3.3), teaching signal (2.3.4) and error value (2.3.5).

It is usually difficult to get $\ddot{\theta}$ with accuracy, so let's consider use of the following first order lag reference model and teaching signal.

$$\dot{\theta}_M + k_{M3}\theta_M = 0 \quad (2.3.7)$$

$$T(t) = u(t) + \delta F \operatorname{sgn}(e(t + \delta t)) \quad (2.3.8)$$

$$e(t) = \dot{\theta} - \dot{\theta}_M \quad (2.3.9)$$

Although the teaching signal (2.3.8) is not always appropriate as a teaching signal, θ was regulated through learning (Figure 2.3.5). Moreover, the following zero order lag reference model and teaching signal are effective.

$$\theta_M = 0 \quad (2.3.10)$$

$$T(t) = u(t) + \delta F \operatorname{sgn}(e(t + \delta t)) \quad (2.3.11)$$

$$e(t) = \theta - \theta_M \quad (2.3.12)$$

After enough learning with (2.1.10)(2.1.11)(2.1.12), the angle of the pendulum is regulated.

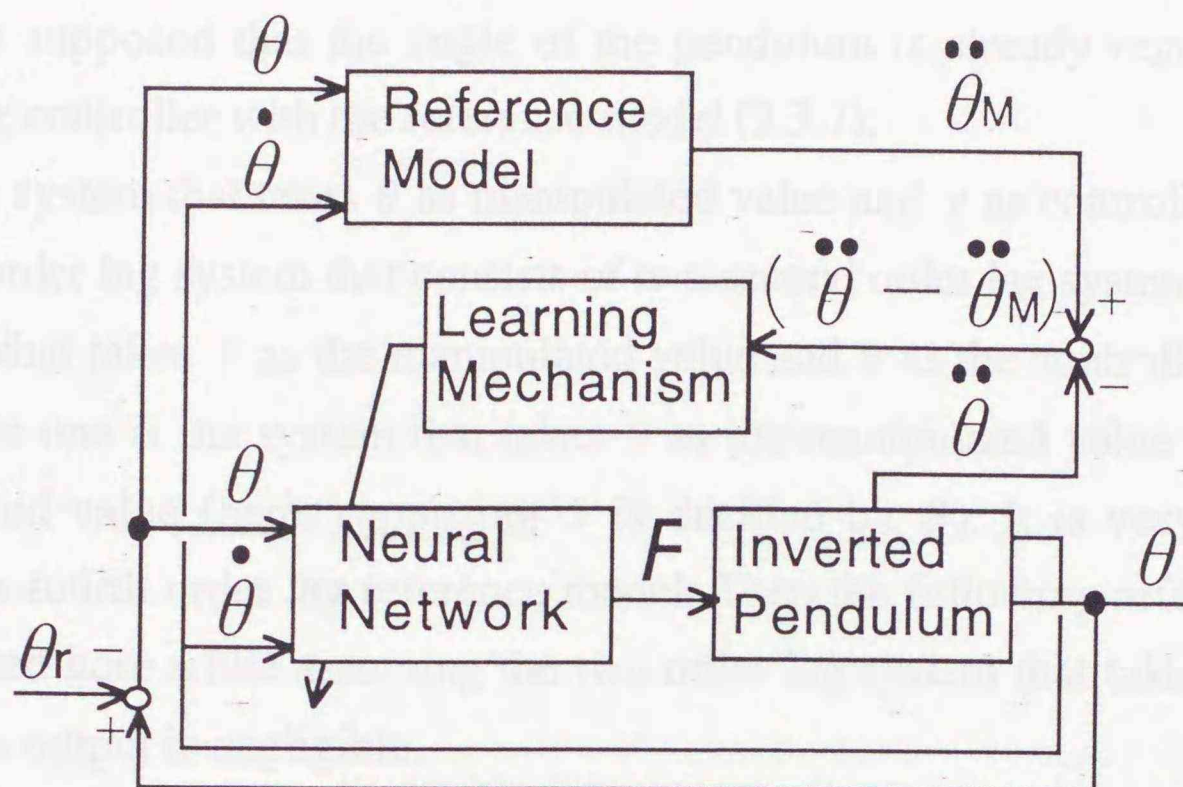


Figure2.3.4 The learning control system using a neural network

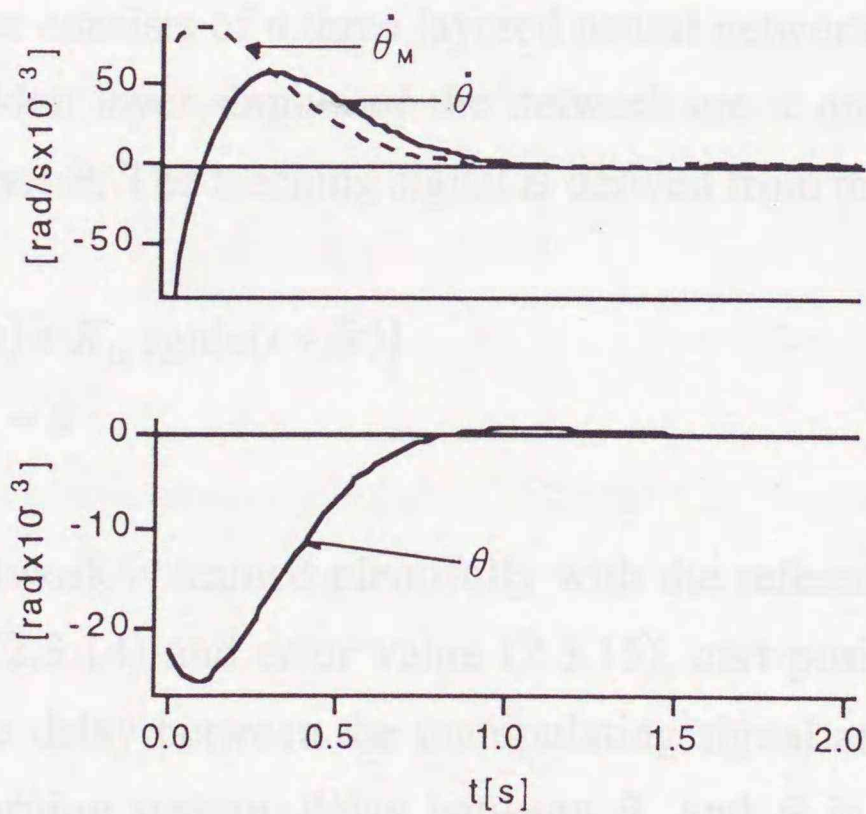


Figure2.3.5 Responses of the pendulum
(using a first order lag reference model)

(b) Control of the Cart Position

It is supposed that the angle of the pendulum is already regulated by the learning controller with the reference model (2.3.7).

The system that takes θ as manipulated value and x as controlled value is a fourth order lag system that consists of two second order lag systems. One is the system that takes F as the manipulated value and θ as the controlled value and the other one is the system that takes $\ddot{x}$ as the manipulated value and x as the controlled value (here, supposing $\ddot{x}$ is decided by θ). It is very difficult to design a fourth order lag reference model. Then the following reference model is adopted here while assuming the two order lag system that takes F as input and θ as output is negligible.

$$\ddot{x}_M + k_{Mx1}\dot{x}_M + k_{Mx2}x_M = 0 \quad (2.3.13)$$

The controller consists of a three layered neural network which contains 10 neurons in a hidden layer. Inputs of the network are x and $\dot{x}$. Output of the network determines θ . The teaching signal is derived from the following.

$$T(t) = u(t) + K_{lx} \operatorname{sgn}(e(t + \delta t)) \quad (2.3.14)$$

$$e(t) = \ddot{x}_M - \ddot{x} \quad (2.3.15)$$

When the network is trained plentifully with the reference model (2.3.13), teaching signal (2.3.14) and error value (2.3.15), cart position vibrates. It is caused by a large delay between the manipulating signal and the manipulated value. In this learning system, delay between θ_r and θ is larger than δt , so teaching signals that are made from $x(t + \delta t), \dot{x}(t + \delta t)$ are not appropriate.

Therefore the network is trained with the following process. 1) At first, measurement of plenty of data. 2) Then, training the neural network using data

selected randomly from measured data with back propagation. After sufficient training, cart position and angle of the pendulum were regulated.

2.3.5 Experimental Results

(a) Regulation of the Pendulum Angle

The angle of the inverted pendulum is regulated on the experimental system, using reference model (2.3.7), teaching signal (2.3.8) and error value (2.3.9). After sufficient training, the angle of the pendulum vibrates. This is caused by the delay between calculated value of the angle velocity with moving average method and the true value. In fact, in the simulation the angle of the pendulum also vibrates when using the angle velocity with moving average. It is observed that the angle velocity vibrates round a reference value. To reduce this vibration, we modify the teaching signal to the following.

$$T(t) = u(t) + \delta F \operatorname{sgn}(e(t + \delta t)) + K_D \dot{e}(t + \delta t) \quad (2.3.16)$$

Teaching signal (2.3.16) contains differentiation, so we modify equation (2.3.16) to discrete expression.

$$T(t) = u(t) + \delta F \operatorname{sgn}(e(t + \delta t)) + K_D (x(t + \delta t) - x(t)) \quad (2.3.17)$$

Training the network using equation (2.3.17) as teaching signal, the angle of the pendulum was regulated. Figure 2.3.6 shows the response of the pendulum when training 800 times with

$$\delta F = 0.1, K_D = 5.0.$$

(b) Regulation of the cart position

Next, the cart position is regulated on the experimental system using reference

model (2.3.13), teaching signal (2.3.14) and error value (2.3.15). The network is trained with data that are selected randomly from measured data. After adequate training, cart position and angle of the network were regulated. Figure 2.3.7 exhibits the response of the inverted pendulum, when training 400 times with $k_{Mx1} = 0.3$, $k_{Mx2} = 0.3$ and $K_{lx} = 0.03$.

2.3.6 Discussions

Dividing control makes possible designing of a reference model for a multi output plant. It is difficult to design a reference model for an inverted pendulum, without knowledge of structure and parameters of a plant. Contrary it is much easier to design a reference model for a sub-plant, i.e., two order last elements. Moreover neural network learning control has high robustness. A function which should be learned by each neural network is simplified and the risk of falling into local minimums is also reduced. Consequently an appropriate control rule was acquired. In theory, a three layered neural network can approximate any function with accuracy. For a complicated problem, however, the method of dividing is important.

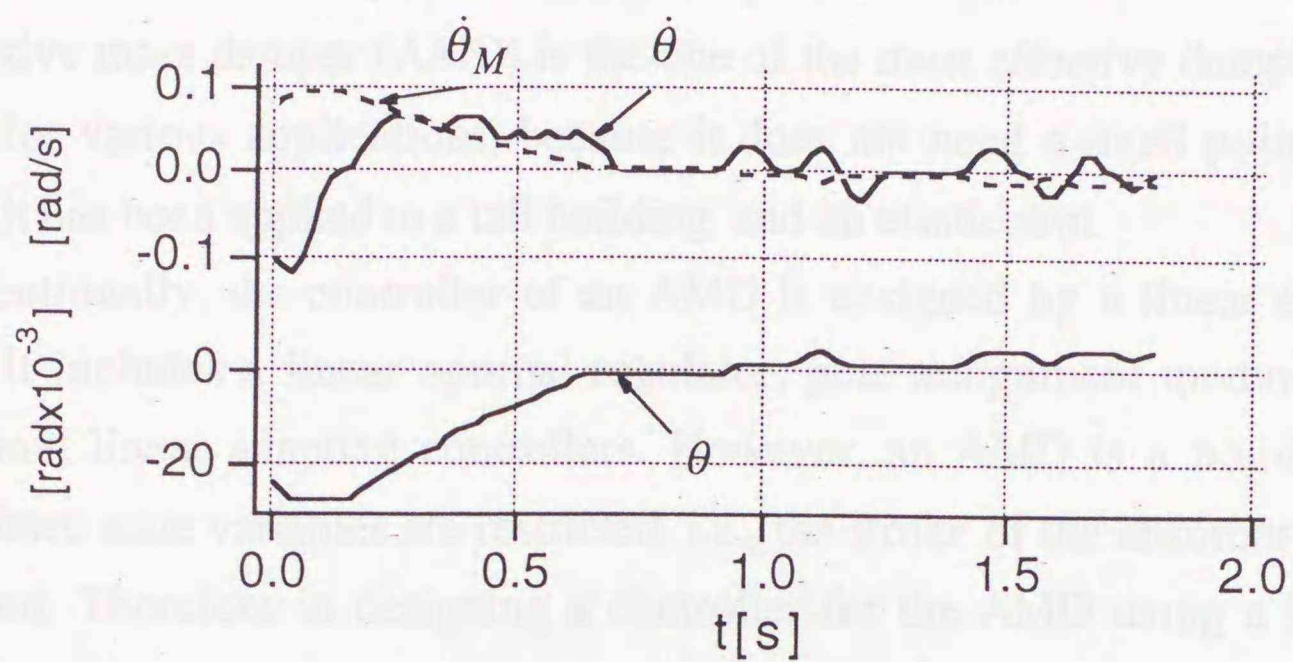


Figure 2.3.6 Responses of the pendulum

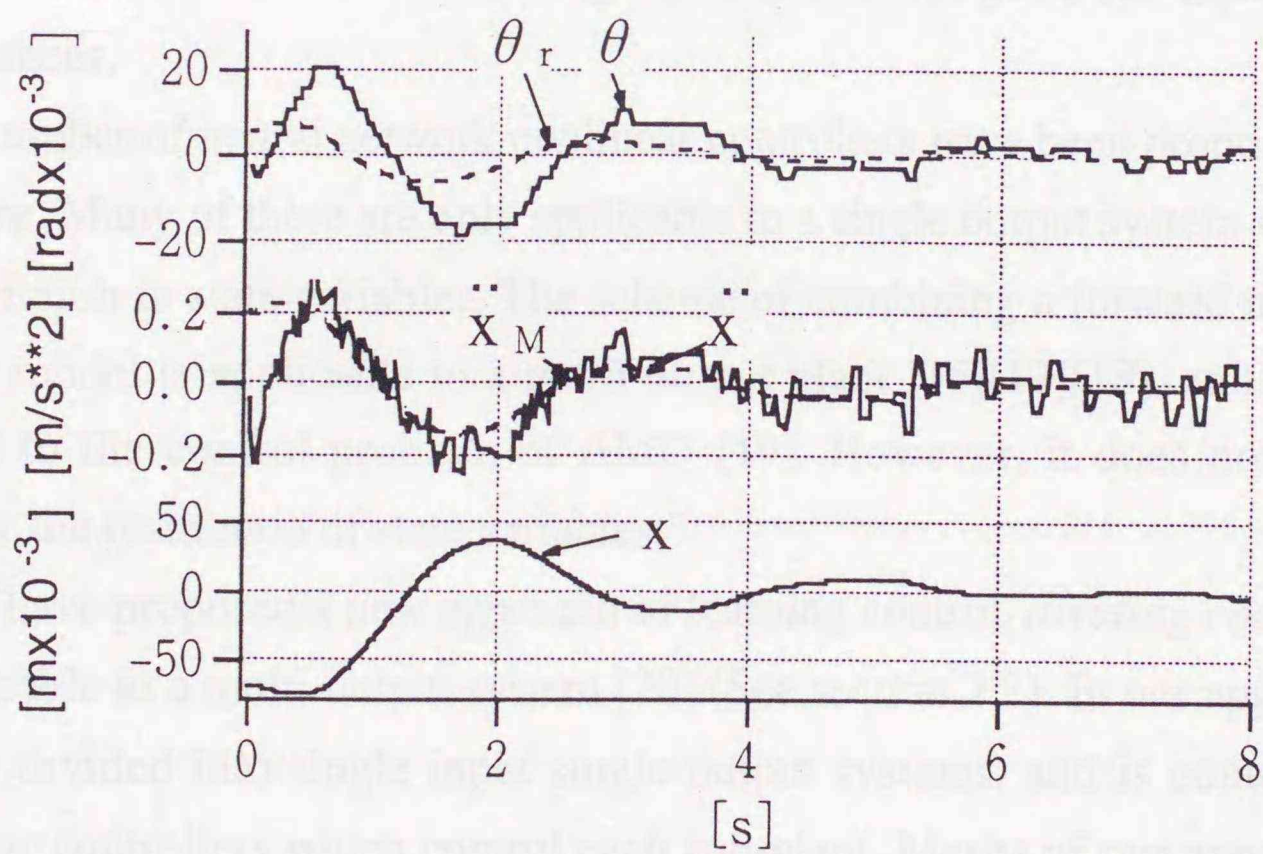


Figure 2.3.7 Responses of the inverted pendulum system

2.4 Control Problem of an Active Mass Damper.

An active mass damper (AMD) is the one of the most effective dampers. It is useful for various applications, because it does not need a fixed point. For example, it has been applied to a tall building and an elastic arm.

Conventionally, the controller of an AMD is designed by a linear theory [14][15]. It includes a linear optimal regulator, pole assignment method and conventional linear adaptive controllers. However, an AMD is a non-linear system whose state variables are restricted, i.e., the stroke of the absorber mass is restricted. Therefore in designing a controller for the AMD using a linear theory, it is necessary to restrict state variables by trial and error.

The algorithm of Ponrjagin and Dynamical Programming solves a nonlinear optimal control problem. However, when a plant is nonlinear, its solution is expressed as a function of time in general and is not good for control under disturbances.

A number of neural network nonlinear controllers have been proposed in the literature. Many of these are only applicable to a single output system which has no restriction in state variables. The scheme of combining a forward model and inverse model is applicable to a multi-output plant [16][17][18], and has been applied to the control problem of AMD [19]. However, it does not directly consider the restriction of state variables.

We have proposed a new approach of learning control, *dividing control*, that is applicable to a multi-output system [20] (See section 2.2). In our approach, a plant is divided into single input single output systems, and is controlled by nonlinear controllers which control each sub-plant. Merits of our approach are that it is applicable to a multi-output system and does not need a forward model in contrast with the scheme of combining a forward model and inverse model. We have applied it to a control problem of an inverted pendulum [7] (See section 2.3). Output of a sub-plant is controlled as it follows a reference model.

Therefore a controller can restrict some state variables by designing an appropriate reference model.

In this section, dividing control is applied to the control problem of an AMD which is applied to an elastic arm. In some experiments of this section, it is supposed that an AMD has no spring and damper between a main structure and an absorber mass. A neural network can adapt itself to the system which has spring and damper through learning as well.

2.4.1 Active Mass Damper

An elastic arm with AMD is illustrated in Figure 2.4.1. Assuming that an elastic arm is a single degree of freedom vibration system, the plant is modeled as a two degrees of freedom vibration system (Figure 2.4.2). Its dynamics are as follows.

$$m_1 \ddot{x}_1 = -(k_1 + k_2)x_1 + k_2 x_2 - (c_1 + c_2)\dot{x}_1 + c_2 \dot{x}_2 - u \quad (2.4.1)$$

$$m_2 \ddot{x}_2 = -k_2 x_2 - c_2 \dot{x}_2 + u \quad (2.4.2)$$

Where x_1 , x_2 and x_3 denote the displacement of the main structure, the displacement of the absorber mass and the relative displacement between the main structure and the absorber mass respectively. m_1 , m_2 , k_1 , k_2 , c_1 , c_2 and u denote mass of the main structure, mass of the damper mass, the spring constant of the main structure, the spring constant of the absorber, the damping coefficient of the main structure, the damping coefficient of the absorber and the force applied by the actuator respectively. x_0 corresponds to the angle of the axis of rotation.

If k_2 and c_2 are negligible, then equation (2.4.1) and (2.4.2) become the following.

$$m_1 \ddot{x}_1 = -k_1 x_1 - c_1 \dot{x}_1 - u \quad (2.4.3)$$

$$m_2 \ddot{x}_2 = u$$

(2.4.4)

Let's introduce $x_{3\max}$ and $u_{\max}$, which denote the maximum stroke of x_3 and the maximum value of u . The object of control is to absorb x_1 as fast as possible, while keeping $|x_3| < x_{3\max}$ and $|u| < u_{\max}$.

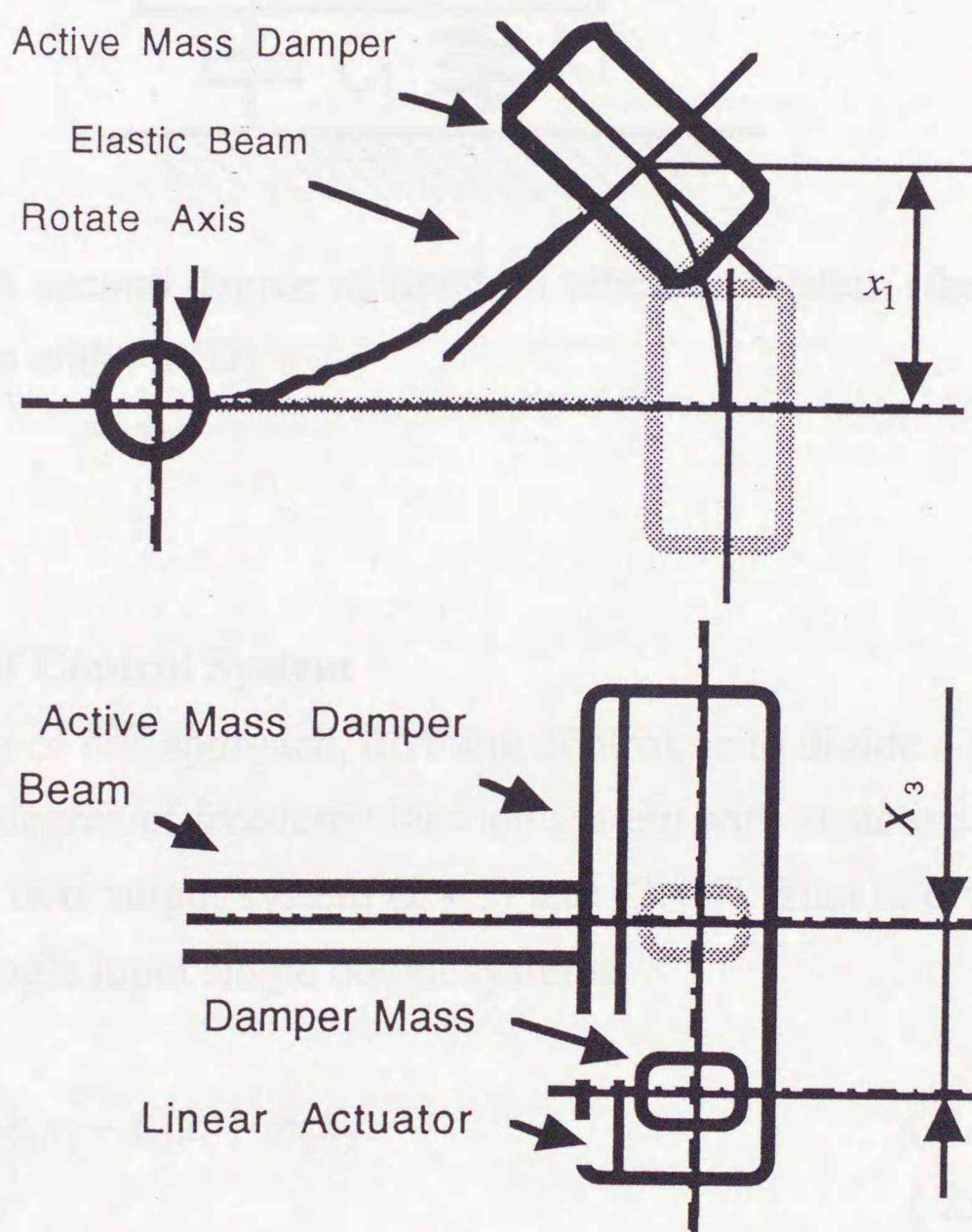


Figure 2.4.1 An elastic arm with an active mass damper

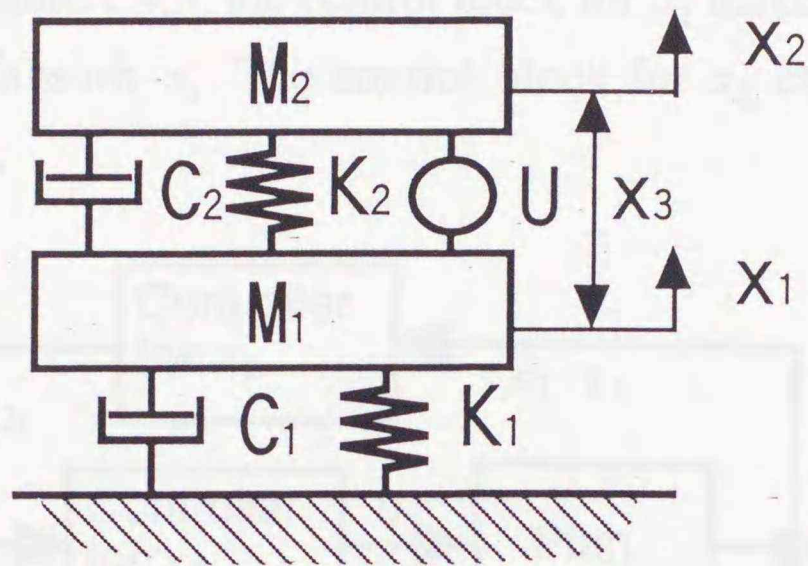


Figure 2.4.2 A second degree of freedom vibration system (the model of an elastic arm with AMD)

2.4.2 Dividing of Control System

The first step of our approach, dividing control, is to divide a plant into sub plants. A single degree of freedom vibration system with an active mass damper is a single input two-output system (2.4.3) and (2.4.4). That is divided into the following two single input single output systems.

$$m_1 \ddot{x}_1 = -k_1 x_1 - C_1 \dot{x}_1 - m_2 \ddot{x}_2 \quad (2.4.5)$$

$$m_2 \ddot{x}_2 = u \quad (2.4.6)$$

System (2.4.5) is regarded as a s.i.s.o. system whose input is $\ddot{x}_2$ and output is x_1 . System (2.4.6) is also regarded as a s.i.s.o. system whose input is u and output is x_2 . Each system is controllable.

Input of the system (2.4.5), $\ddot{x}_2$, is not directly manipulative. However, if it is possible to control x_2 with a sufficiently short time constant, $\ddot{x}_2$ is indirectly

manipulative.

Consequently, a controller which has a block diagram is designed as shown in Figure 2.4.3. In Figure 2.4.3, the control block for x_1 makes x_{2r} , a reference value of x_2 , which absorb x_1 . The control block for x_2 calculates u which makes x_2 follow x_{2r} .

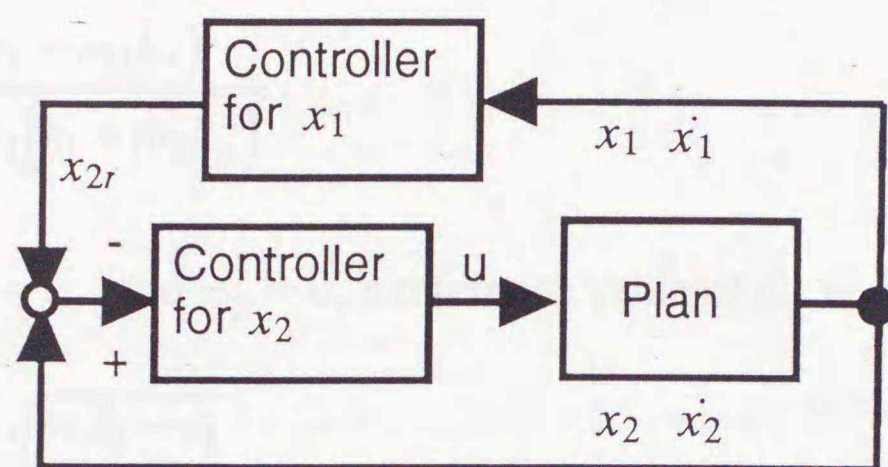


Figure 2.4.3 The block diagram of the controller designed by dividing control

2.4.3 Introducing Learning Controller and Reference Model

The second step of our approach is to design control blocks which control sub plants. There are several ways to design a controller for each sub plant. From the point of flexibility and adaptability, it should be a nonlinear learning controller. In previous section a nonlinear learning controller is applied to each sub plant of an inverted pendulum [7]. However, it is difficult to apply a learning controller to the control block for x_1 , because the output of the system (2.4.6) and input of the system (2.4.5) is different. Therefore we designed the control block for x_1 analytically, and a learning controller is applied to control block for x_2 .

(a) Design of Control Block for x_1 .

The control block for x_1 is derived analytically as follows.

Let us apply a linear feed back controller to the system (5), then the input of the system, $\ddot{x}_2$, is as following.

$$\ddot{x}_2 = k_p x_1 + k_d \dot{x}_1 \quad (2.4.7)$$

A damping ratio of x_1 , ζ , is derived by substituting equation (2.4.7) for equation (2.4.5).

$$\zeta = \frac{(c_1 + m_2 k_d)}{2\sqrt{m_1(k_1 + m_2 k_p)}} \quad (2.4.8)$$

Supposing $\zeta = \zeta_r$ and $k_p = 0$, a reference value of $\ddot{x}_2$ is obtained as follows.

$$k_d = \frac{2\zeta_r \sqrt{m_1 k_1 - c_1}}{m_2} \quad (2.4.9)$$

$$\ddot{x}_{2r} = k_d \dot{x}_1 \quad (2.4.10)$$

The reference value of x_2 and $\dot{x}_2$ are obtained from integration of equation (2.4.10), deciding a constant of integration as 0 to keep the center of vibration of x_3 to the displacement 0, and supposing (2.4.11).

$$x_1 \cong a(t) \sin \omega t, \quad \dot{x}_1 \cong a(t) \cos \omega t \quad (2.4.11)$$

$$\dot{x}_{2r} = -b(t)a \sin \omega t = k_d x_1 \quad (2.4.12)$$

$$x_{2r} = b(t) \cos \omega t = -\frac{k_d}{\omega^2} \dot{x}_1 \quad (2.4.13)$$

$$b(t) = k_d \frac{a(t)}{\omega} \quad (2.4.14)$$

where ω is a natural frequency of the main structure, which is measured practically, $a(t)$ is amplitude of the main structure which is calculated as below.

$$a(t) = \sqrt{x_1^2 + \frac{1}{\omega^2} \dot{x}_1^2} \quad (2.4.15)$$

Equation (2.4.13) and (2.4.14) are constrained by the restriction concerning x_3 and u as follows.

$$b(t) < \sqrt{x_{3\max}^2 - a(t)^2} \quad (2.4.16)$$

$$b(t) < \frac{u_{\max}}{\omega^2 m_2} \quad (2.4.17)$$

In the above, the reference value of x_2 and $\dot{x}_2$ is derived as the function of ζ_r .

(b) Design of Control Block for x_2

The control block for x_2 is designed with a learning controller using a layered neural network shown in Figure 2.4.4[7]. It consists of four parts: 1) a plant, 2) a neural network which acquires a control law by learning, 3) a part that makes a reference value of output of the plant, 4) a learning mechanism which makes a neural network learn a control rule using a deviation between a reference value and output of a plant.

Learning algorithm is shown in the following.

$$tc(t) = k_{tp}(t)(x_{2r} - x_2) + k_{td}(t)(\dot{x}_{2r} - \dot{x}_2) \quad (2.4.18)$$

$$back\ Pr\ opagation(NN(\mathbf{x}(t)), tc(t + \delta t)) \quad (2.4.19)$$

where backpropagation(a,b) denotes modification of a neural network with teaching signal using back propagation method. $\mathbf{x}$ denotes the input vector of a neural network. Usually $\mathbf{x}$ contains all state variables.

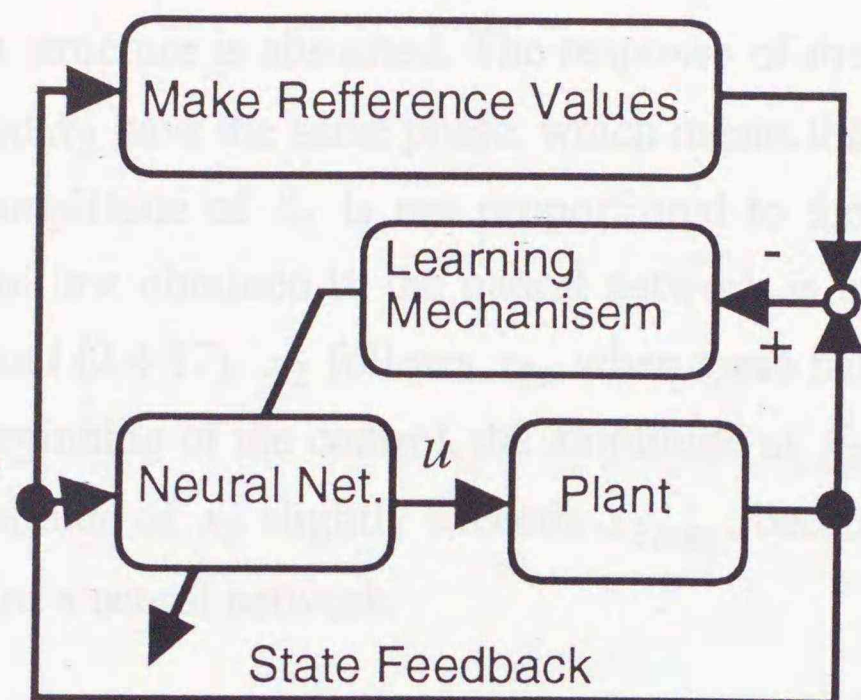


Figure 2.4.4 The learning controller using a neural network

2.4.4 Results of Experiments

The last step of our approach is a learning of a neural network. To demonstrate the effectiveness of the controller designed by our approach, experiments are conducted. Through the experiment, the following is used as plant parameters unless particular descriptions.

$$\begin{aligned}
 k_1 &: 70[\text{N/m}], m_1 : 0.70[\text{kg}], c_1 : 0.0[\text{Ns/m}] \\
 k_2 &: 0[\text{N/m}], m_2 : 0.043[\text{kg}], c_2 : 0.0[\text{Ns/m}], \\
 x_{3\max} &: 0.08[\text{m}], u_{\max} : 0.35[\text{N}]
 \end{aligned}
 \tag{2.4.20}$$

A neural network which is used in a learning controller is configured with four input cells, one hidden layer of 10 cells, and one output cell. A neural network takes x_1 , $\dot{x}_1$, x_3 and $\dot{x}_3$ as input.

(a) Experiment I: Simulation Environment

An experiment in the simulation environment is conducted. After adequate learning, the main structure is absorbed. The response of the plant is shown in Figure 2.4.5. $\dot{x}_1$ and $\ddot{x}_2$ have the same phase, which means that equation (2.4.10) is satisfied. The amplitude of $\ddot{x}_2$ is not proportional to the amplitude of $\dot{x}_1$, because the control law obtained in the neural network is nonlinear to satisfy equation (2.4.16) and (2.4.17). x_2 follows x_{2r} when these have the same phase. However, in the beginning of the control, the amplitude of x_2 exceeds that x_{2r} . Therefore the amplitude of x_3 slightly exceeds $x_{3\max}$, because of the shortage of learning ability of a neural network.

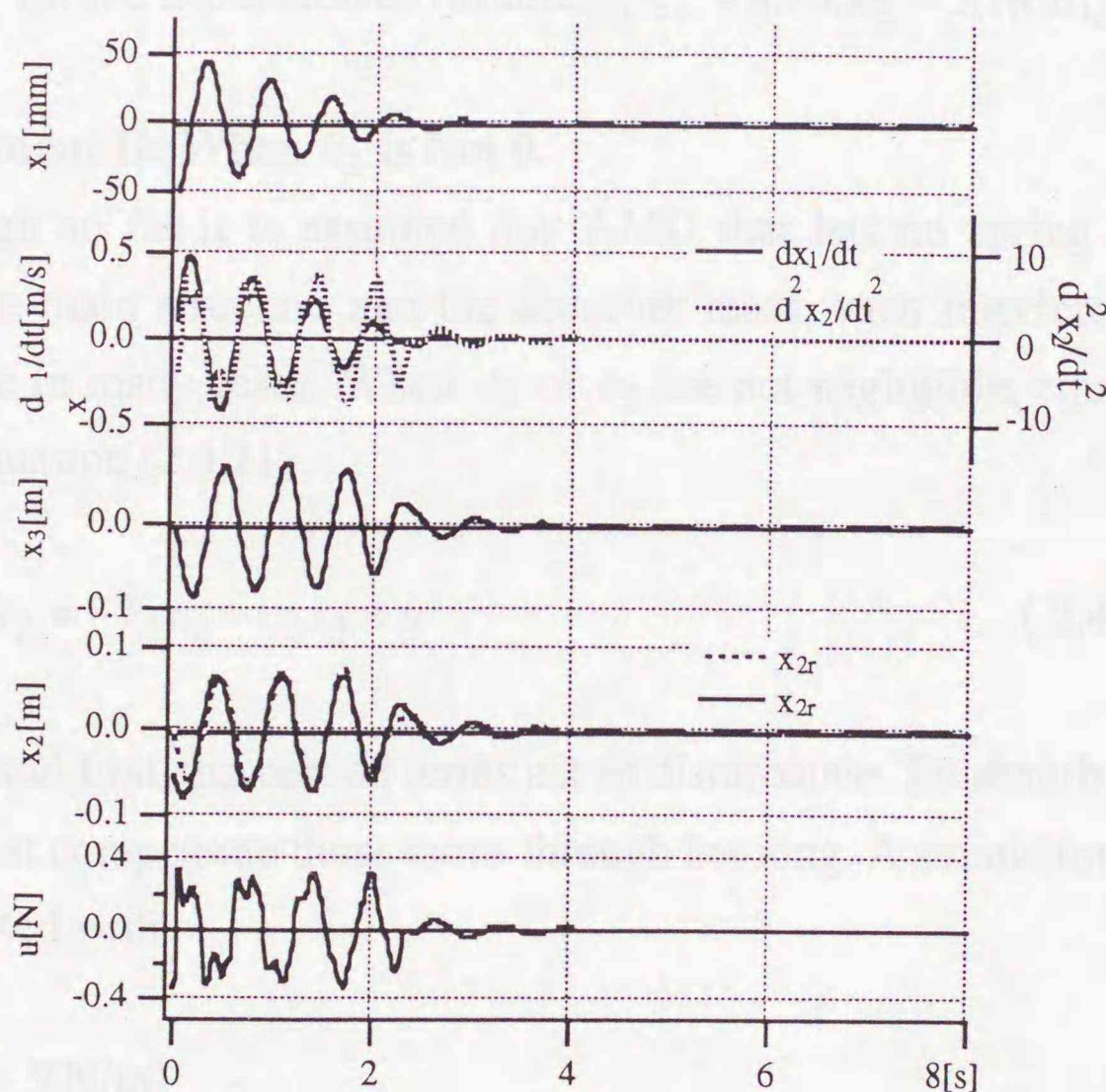


Figure 2.4.5 Responses of the elastic arm with AMD
(simulation environment) ($\zeta_r = 0.15$).

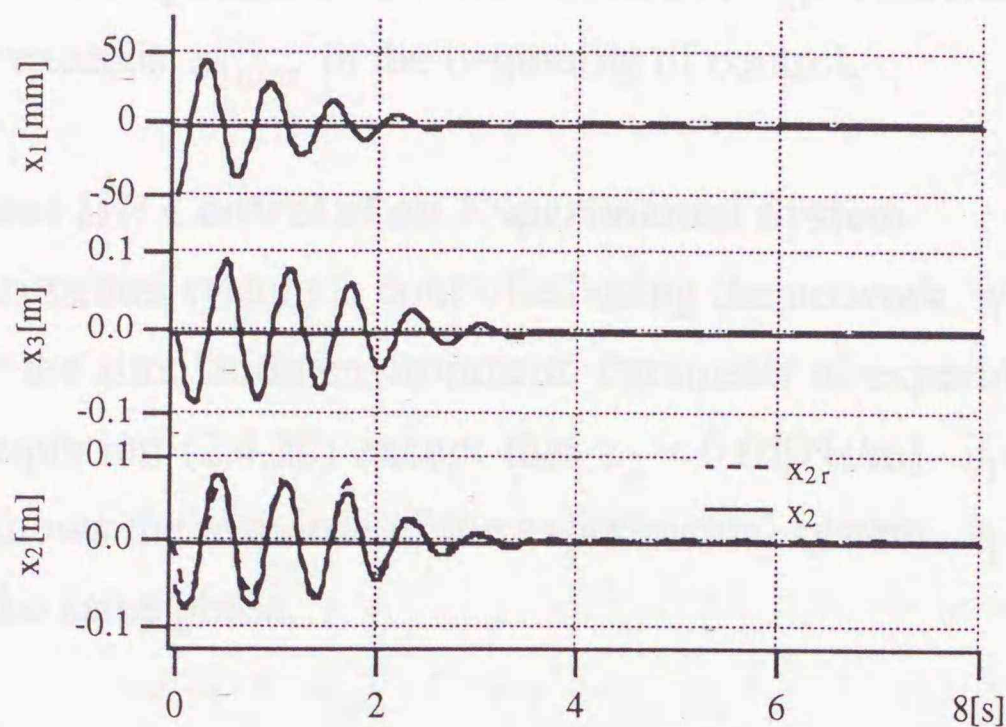


Figure 2.4.6 Responses of the elastic arm with AMD
(in the simulation environment, $\zeta_r = 0.15, k_2 = 5[\text{N/m}]$)

(b) Experiment II: When k_2 is Not 0.

Although so far it is assumed that AMD that has no spring and damper between the main structure and the absorber mass, such interference force is unavoidable in many cases. When c_2 or k_2 are not negligible, equation (2.4.6) becomes equation (2.4.21).

$$m_2 \ddot{x}_2 = -k_2 x_3 - c_2 \dot{x}_3 + u \quad (2.4.21)$$

Right hand first and second terms act as disturbance. To absorb x_1 , a neural network must compensate those terms through learning. A simulation experiment was conducted with

$$k_2 = 5[\text{N/m}]$$

After sufficient learning, the main structure is absorbed (Figure 2.4.6). Disturbance is compensated by the neural network. However, in the beginning

of the control, the amplitude of x_2 exceeds one of x_{2r} . Therefore the amplitude of x_3 slightly exceeds $x_{3\max}$ in the beginning of control.

(c) Experiment III: Control of an Experimental System

The experimental system is controlled using the network, which is obtained by learning in the simulation environment. Parameter of experimental system is the same as equation (2.4.20) except that $c_2 = 0.01[\text{Ns/m}]$, $c_1 = 0.043[\text{Ns/m}]$. Figure 2.4.7 shows the response of the experimental system. x_1 is absorbed. $\dot{x}_1$ and $\ddot{x}_2$ have the same phase.

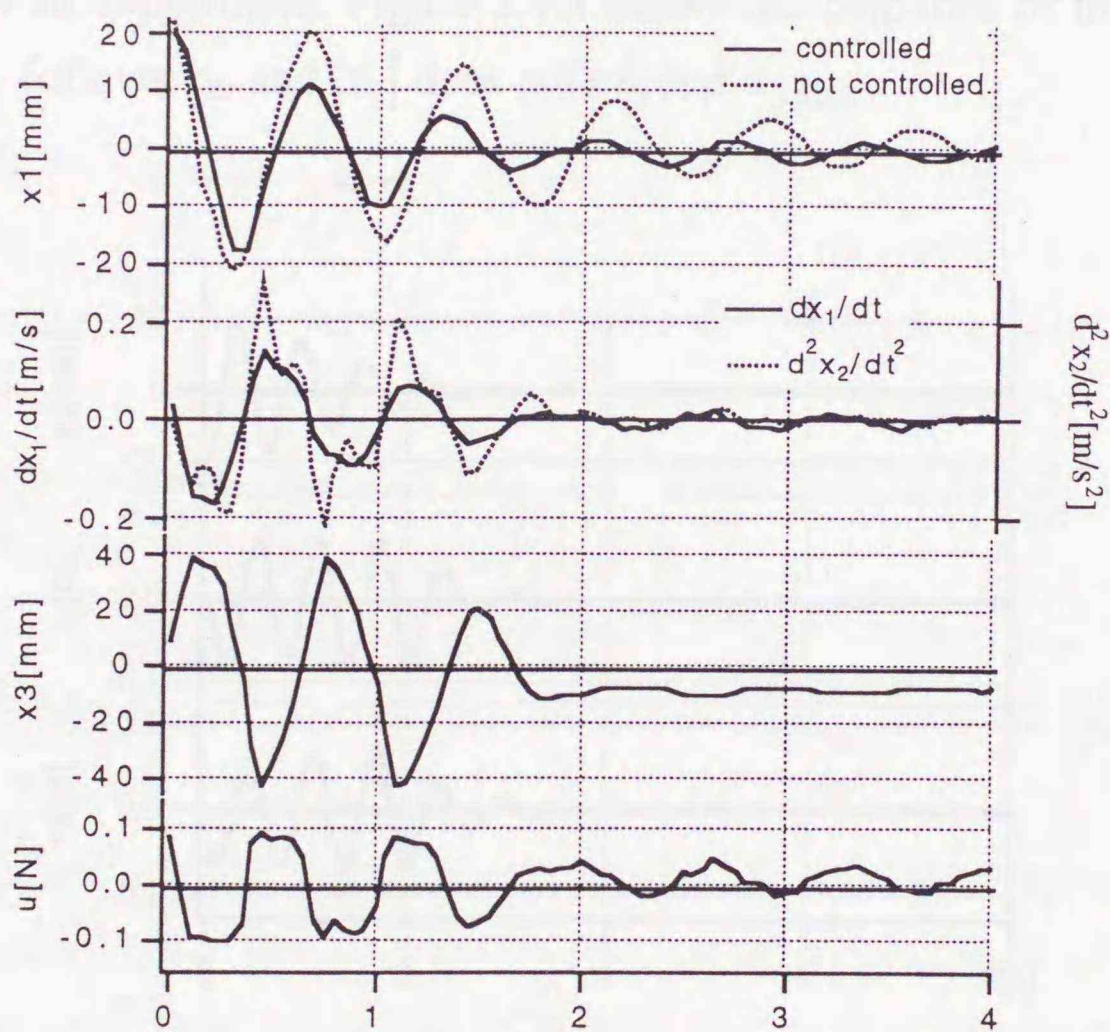


Figure 2.4.7 Responses of the experimental system
(Controlled by the neural network which is trained with $\zeta_r = 0.44$.)

(d) Experiment IV: Using Three Neural Networks.

In experiment I and II, learning error is not reduced enough because of a strong nonlinearity of a control law which must be learned by a neural network. Controller must acquire three different control rules: 1) in the beginning of control a controller adjusts the center of vibration and phase of x_2 to x_{2r} , 2) in the middle a controller maintains the restriction (2.4.16), 3) in the last controller damps x_1 with ζ_r . To solve this problem, first number of hidden layer cells of a neural network is increased to 100; however, performance of controller is not improved. Therefore authors apply a neural network to each part of the control and conduct an experiment. Figure 2.4.8 shows the response of the plant after learning. x_2 follows x_{2r} and $|x_3|$ does not exceed $x_{3\max}$.

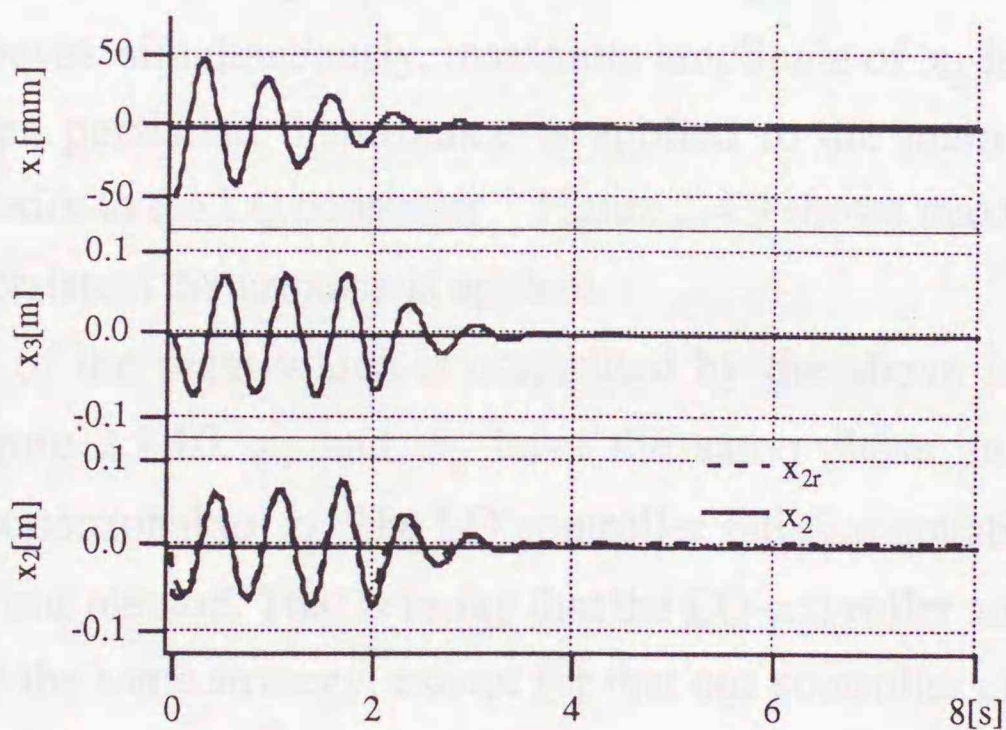


Figure 2.4.8 Responses of the elastic arm with AMD controlled by three networks.

(e) Comparison With a Linear Quadratic Controller.

The controller designed by our method is superior to LQ controller in the settling time and maximum amplitude of x_3 .

Table 1 shows settling time of x_1 and maximum amplitude of x_3 when a plant is controlled by a neural network which is obtained in experiment I and a LQ controller from initial states are $x_1 = 0.05[\text{m}]$, $x_3 = 0[\text{m}]$. Here, a cost function of the LQ controller is defined as follows.

$$J = \int (w_1 x_1^2 + w_2 x_2^2 + w_4 u^2) dt$$
$$w_1 = 100, w_2 = 4, w_4 = 0.9 \quad (2.4.22)$$

Our method achieves a shorter settling time with less amplitude of x_3 than the LQ controller. Increasing w_1 of (2.4.22), settling time of the LQ controller is reduced. However, simultaneously, maximum amplitude of x_3 increases.

Also, when persistent disturbance is applied to the main structure, our method is superior to the LQ controller. Figure 2.4.9 shows maximum amplitude of x_1 when persistent disturbance is applied.

Response of the plant which is controlled by the above LQ controller is shown in Figure 2.4.10. $\dot{x}_1$ and $\ddot{x}_2$ have the same phase and $\ddot{x}_2$ is almost completely proportional to $\dot{x}_1$. The LQ controller satisfies equation (2.4.10) that is satisfied by our method. That is to say that the LQ controller and our controller have basically the same strategy, except for that our controller changes the ratio between $\dot{x}_1$ and $\ddot{x}_2$ to keep a stroke of an absorber mass in a restricted range. In fact, our controller learning without restriction about x_3 and u behaves like an LQ controller.

This fact suggests that our controller has as much robustness as the LQ controller. The settling time and the maximum amplitude of x_3 vary as shown in Figure 2.4.11 and 2.4.12 by the fluctuation of k_1 and c_2 . They show that the

controller designed by our approach is as robust as the LQ controller.

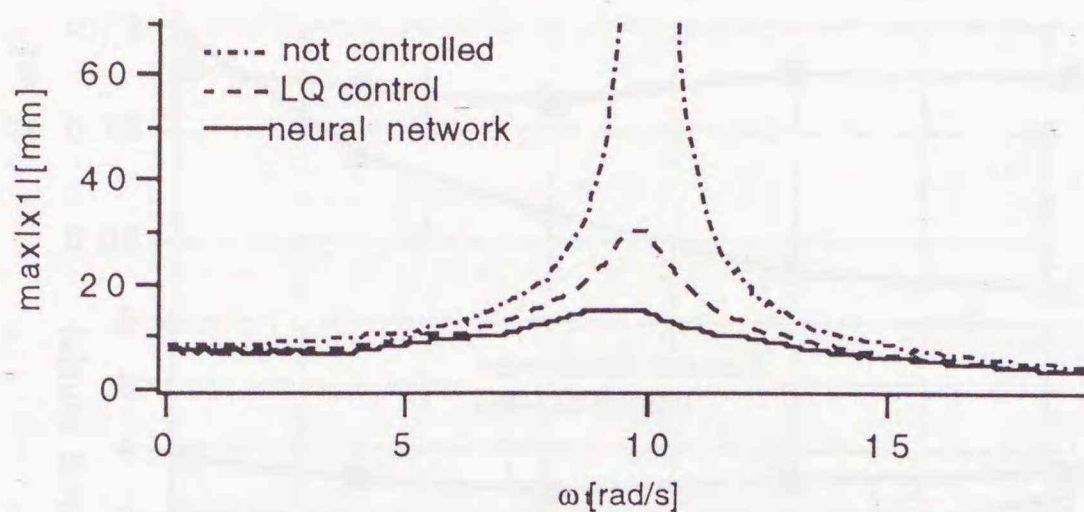


Figure 2.4.9 Maximum amplitude of x_1
(persistently disturbance is applied to the angle of the rotary axis)

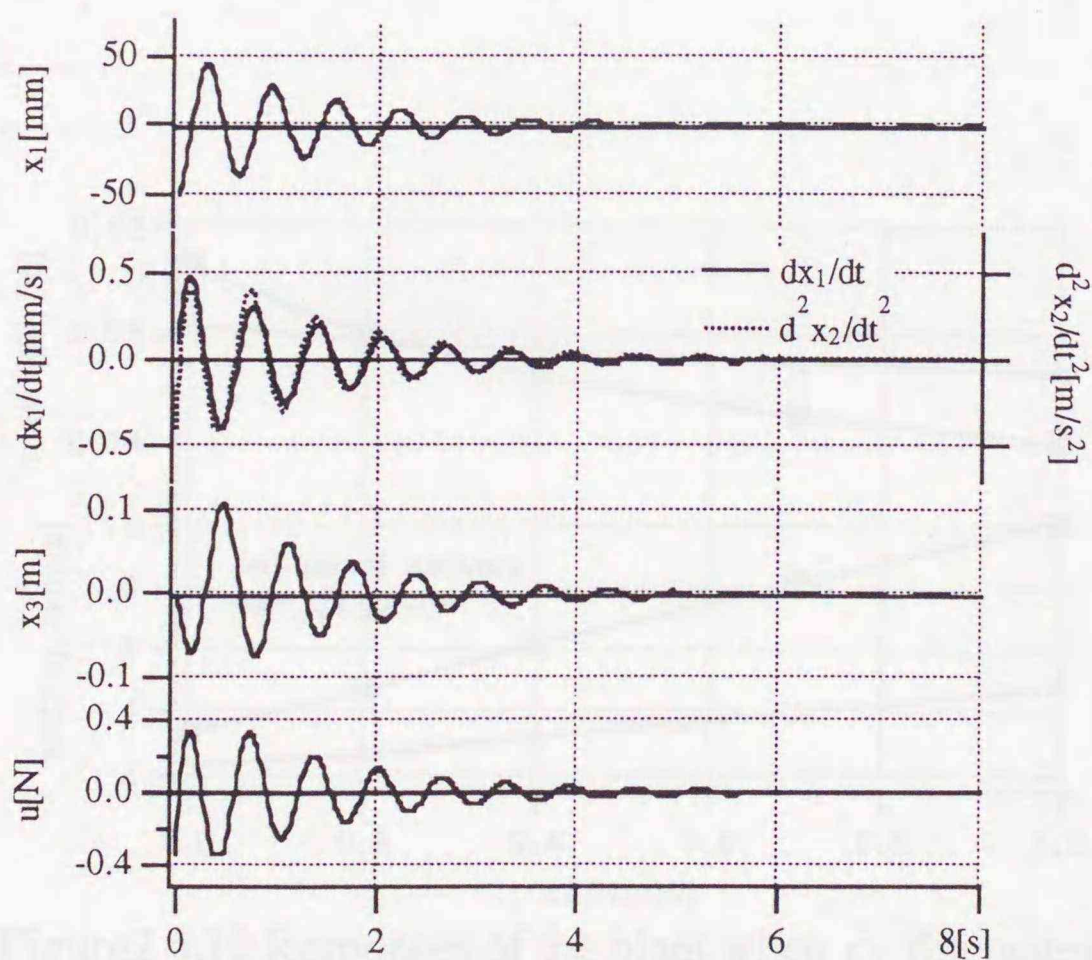


Figure 2.4.10 Responses of the the experimental system controlled by a linear quadratic controller

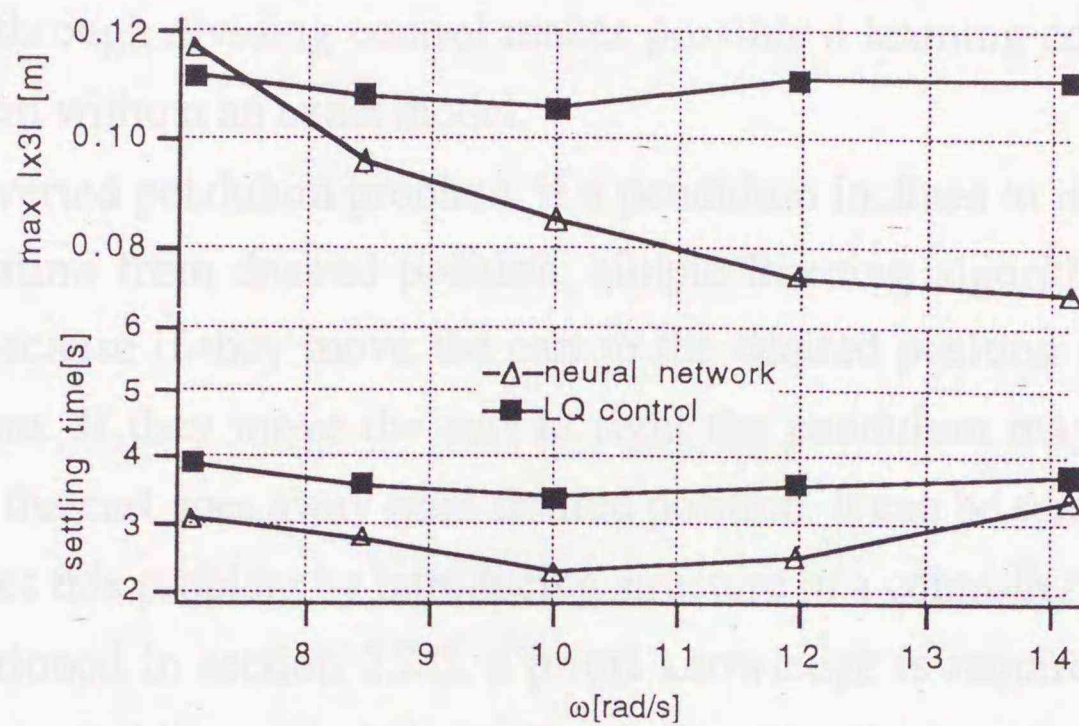


Figure 2.4.11 Responses of the plant when k_1 fluctuates

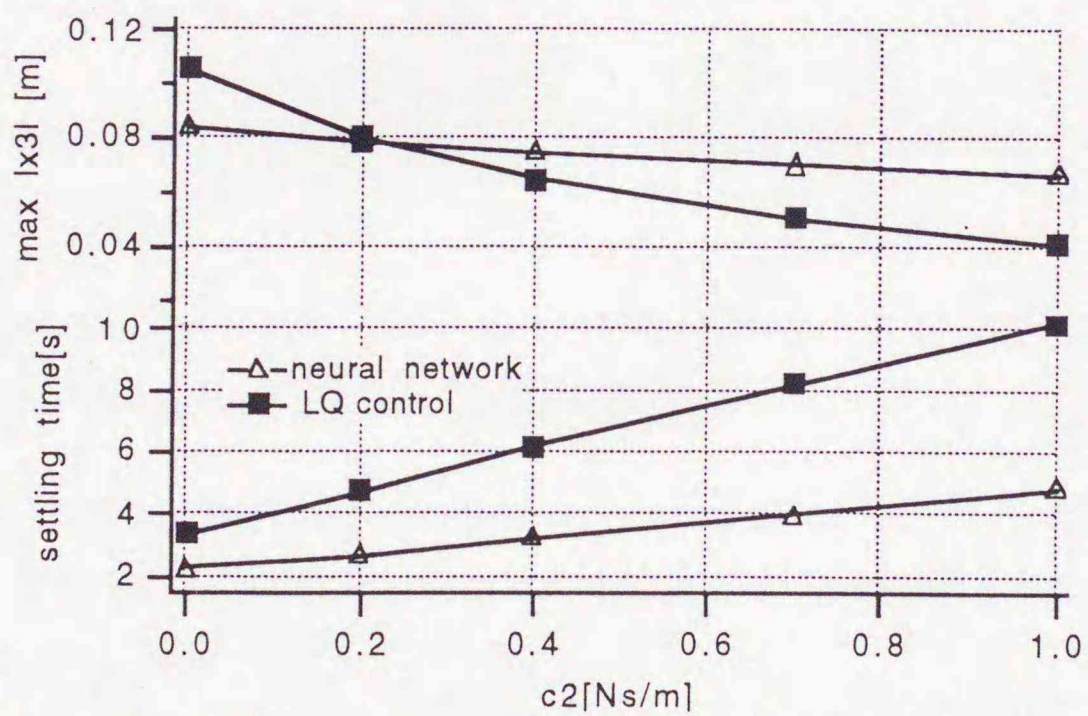


Figure 2.4.12 Responses of the plant when c_2 fluctuates

2.5 Conclusions

In this chapter, dividing control method is introduced. A structured controller introduced through dividing control makes possible a learning control of multi output system without an exact model.

In an inverted pendulum problem, if a pendulum inclines to right and a cart is right position from desired position, simple learning algorithm may be at deadlock, because if they move the cart to the desired position the pendulum inclines more. If they move the cart to right the pendulum may approach to upright, but the cart goes away from desired position. It can be said that dividing control solves this problem by introducing structure in a controller.

As mentioned in section 2.2.2, a priori knowledge is required to divide a plant. However dividing control require less information than most of conventional controller.

References of Chapter 2

- [1] M.I. Jordan and R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, *Neural Computation*, vol.6, pp.181-214(1994).
- [2] H. Gomi, M. Kawato, Recognition of manipulated objects by motor learning with modular architecture networks, *Neural Network*, vol.6, pp.485-497(1994).
- [3] M. Dorigo, ALECSYS and the Autonomouse: Learning to control a real robot by distributed classifier systems, *Machine Learning*, Vol.19, pp.290-240(1995).
- [4] W. Findeisen, M. Brys, M. Kalinowski, P. Tatjewski and A. Wozniak, on-line hierarchical control for steady state system, *IEEE Trans. Automat. Control*, AC-23-2, pp.189-209(1978)
- [5] M.G. Singh, *Dynamical hierarchical control*, North-Holland(1980).
- [6] I. Kanellakopoulos, P.V. Kokotovic and A.S. Morse, Systematic design of adaptive controllers for feedback linearizable systems, *IEEE Trans. Automat. Control*, AC-36, pp.1241-1253(1991)
- [7] Y. Miyasato, A new trend of adaptive control, *System, Control and Information*, vol.38, pp.477-484(1994).
- [8] T. Kumagai, et al: Control of an Active Mass Damper Using Layered Neural Networks, *Proceedings of the JSME ICAM*, pp.943-6(1993).
- [9] T. Kumagai et al: Learning Control of an Inverted Pendulum Using Neural Networks, *Trans. of the Japan Society of Mechanical Engineers*, 58-550, C(1992), pp.1781-7. (In Japanese)
- [10] M. Kawato: Computational Schemes and Neural network Models for Formation and Control of Multijoint Arm Trajectory, *Neural Network for Control*, The MIT Press(1990)

- [11] M.Kawato et al: A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement. *Biol.Cybern.*57,169-185(1987).
- [12]M.I.Jordan, Supervised learning and systems with excess degrees of freedom, COINS Technical Report, vol.88-27, pp.1-41, Department of Computer Science and Information, University of Massachusetts, Amherst(1988).
- [13]M.I.Jordan, D.A.Rosenbaum, Action COINS Technical Report vol.88-26, pp.1-68, Department of Computer Science and Information, University of Massachusetts, Amherst.
- [14]S. Lee, A.Sinha: Design of an active vibration absorber, *Journal of Sound and Vibration* 109(2), pp.347-352 (1986).
- [15]I.N.Jordanov, B.I.Cheshankov : Optimal design of linear and non-linear dynamic vibration absorbers, *Journal of Sound and Vibration* 123(1), pp.157-170 (1988).
- [16]D.H.Nguyen, B.Widrow: Neural Networks for Self-Learning Control Systems, *IEEE Control System Magazine*, April(1990).
- [17]M. I. Jordan, Supervised Learning and Systems with Excess Degrees of Freedom, COINS 88-27, Massachusetts Institute of Technology, Cambridge, MA, 1988.
- [18]M.Kawato: Computational Schemes and Neural network Models for Formation and Control of Multijoint Arm Trajectory, *Neural Network for Control*, The MIT Press(1990).
- [19]H.Nagato, K.Yoshida: Control of Active Dynamic Vibration Absorber by Neural Network, *Trans. of the Japan Society of Mechanical Engineers*, 58-550, C(1992), 1755.(in Japanese).

[20]T.Kumagai, R.Hashimoto, M.Wada: Control of an Active Mass Damper Using a Neural Network, Trans. of the Japan Society Mechanical Engineers, 59-564,C, pp.2305-2311(1993).(in Japanese)

Analytical Design of Recurrent Neural Networks

1. Introduction

As mentioned in section 1, there are recurrent neural networks which have been proposed for dynamic systems. However, neural networks with non-linear dynamics, namely oscillatory or chaotic dynamics, have received so far relatively little attention. The main reason for this is that the differential equations that are analyzed or controlled in this kind of networks is exceedingly difficult. As a matter of fact, in most cases, much of the work in this field is either empirically developed or it is too complex to be useful. For instance, no criterion analogous to the Lyapunov function is known for the convergence to stationary cycles.

On the other hand, several real neural networks have been modeled and their dynamical characteristics were analyzed [1][2][3][4][5][6][7][8]. These models exhibit various oscillation dynamics.

Chapter 3

Analytical Design of Recurrent Neural Networks

3.1 Introduction

As mentioned in section 1.1, a lot of artificial neural networks which have convergence dynamics or static characteristics have been proposed and studied. However, neural networks with non-convergent dynamics, namely oscillatory or chaotic dynamics, have received so far relatively little attention. The main reason for this widespread disaffection is that the analysis or control of this kind of networks is exceedingly difficult. As a matter of fact, in most cases, much of the relevant mathematics is either insufficiently developed or it is too complex to be useful. For instance, no criterion analogous to the Lyapunov function is known for the convergence to stationary cycles.

On the other hand, several real neural networks have been modeled and their dynamical characteristics were analyzed [1][2][3][4][5][6][7][8]. These models exhibit various oscillation dynamics.

Model of a single biological neuron has been also studied[9][10]. In these researches it has been shown that a single neuron has great computation power. However, dynamics of those models are too complex to construct networks. In fact, a network which consists of a few realistic neurons shows very complex behavior[11][12][13].

In this section, we introduce fully interconnected neural network which consists time discrete binary output leaky integrator model, i.e., Caianiello-Nagumo neuron model [14][15][16][17]. In the following section, an analytical designing method and convergence conditions are derived[18][19][20]. This network model can memorize a set of context sensitive limit cycles.

3.2 Recurrent Neural Network

3.2.1 Network Model

We assume that time is discrete and outputs of the neurons are binary $\{0,1\}$. The neurons building up the network incorporate a refractory mechanism and perform the integration of afferent signals. The structure of the neuron model and its function are illustrated in Figure 3.2.1. In more detail, each neuron receives two kinds of inputs: namely, inputs from other neurons and externally applied inputs. Incoming signals are weighted, integrated in a leaky way inside the connection paths, and then summed up. The result is compared to a threshold and the neuron output is set at 1 if the difference is positive and at 0 otherwise. The output of the neuron is fed back through a leaky integrator to produce a refractory period. In this way, the firing of a neuron will be influenced by the sequence of its previous inputs and its own outputs, up to the origin of time. By modeling the leaky integrators with first-order recursive digital filters, the influence of past signals will decay with time according to an exponential law. For the sake of simplicity in calculations, although it has no biological foundation, it is assumed that the decay parameter is the same for the refractory mechanism and the integration of the afferent signals. This assumption, however, is not

essential and can be easily removed: the effects of the difference between the two decay parameters can be incorporated in and attributed to an equivalent external bias varying with time.

All the units ($i=1\dots N$) in the network are synchronously updated as follows:

$$y_i(k+1) = f \left(\sum_{j=1}^M v_{ij} \sum_{r=0}^k T^r U_j(k-r) - \sum_{j=1}^N w_{ij} \sum_{r=0}^k T^r y_j(k-r) - h_i \right) \quad (3.2.1)$$

where $y_i(h)$ represents the output of the i -th ($i=1\dots N$) neuron at time h and $u_i(h)$ the j -th ($j=1\dots M$) external input at time h ; v_{ij} is the connection weight from the j -th externally applied input to the i -th neuron; w_{ij} is the connection weight from the j -th neuron to the i -th neuron; f is the step function: $f(x)=0$ for $x<0$ and $f(x)=1$ for $0\leq x$; h_i is the internal threshold and $T(0\leq T<1)$ is the decay parameter. The equation (3.2.1) can be equivalently written in the form:

$$\begin{aligned} x_i(k+1) &= T x_i(k) + \sum_{j=1}^M v_{ij} u_j(k) - \sum_{j=1}^N w_{ij} y_j(k) - \sum_{j=1}^N w_{ij} y_j(k) - h_i(1-T) \\ y_i(k) &= f(x_i(k)) \end{aligned} \quad (3.2.2)$$

Let's introduce the new variable $x_i(k)$ which represents the internal state of the neuron:

$$x_i(k+1) = \sum_{j=1}^M v_{ij} \sum_{r=0}^k T^r u_j(k-r) - \sum_{j=1}^N w_{ij} \sum_{r=0}^k T^r y_j(k-r) - h_i \quad (3.2.3)$$

The state of the network will be described at any time instant k by the vector $\mathbf{x}(k)$ of components $x_i(k)$ with $i=1\dots N$.

By setting $T=0$ and $M=1$, our neural network model coincides with the Hopfield model. When $N=1$ and $M=1$, that is when considering a single

neuron with a single stimulus, is equivalent to Caianiello-Nagumo neuron model.

3.2.2 Dynamics of Network

Suppose the network is stimulated by a set of M periodic inputs, the minimum common multiple of which is T ($T \geq 1$). $V_m \geq 0$, $u_m(t+T) = u_m(t) = 0$, $u_m(t) = 1$, $t \in [0, T)$. Let us order them in such a way that each row corresponds to an input and each column represents a time instant. For every time instant we find its position in the network.

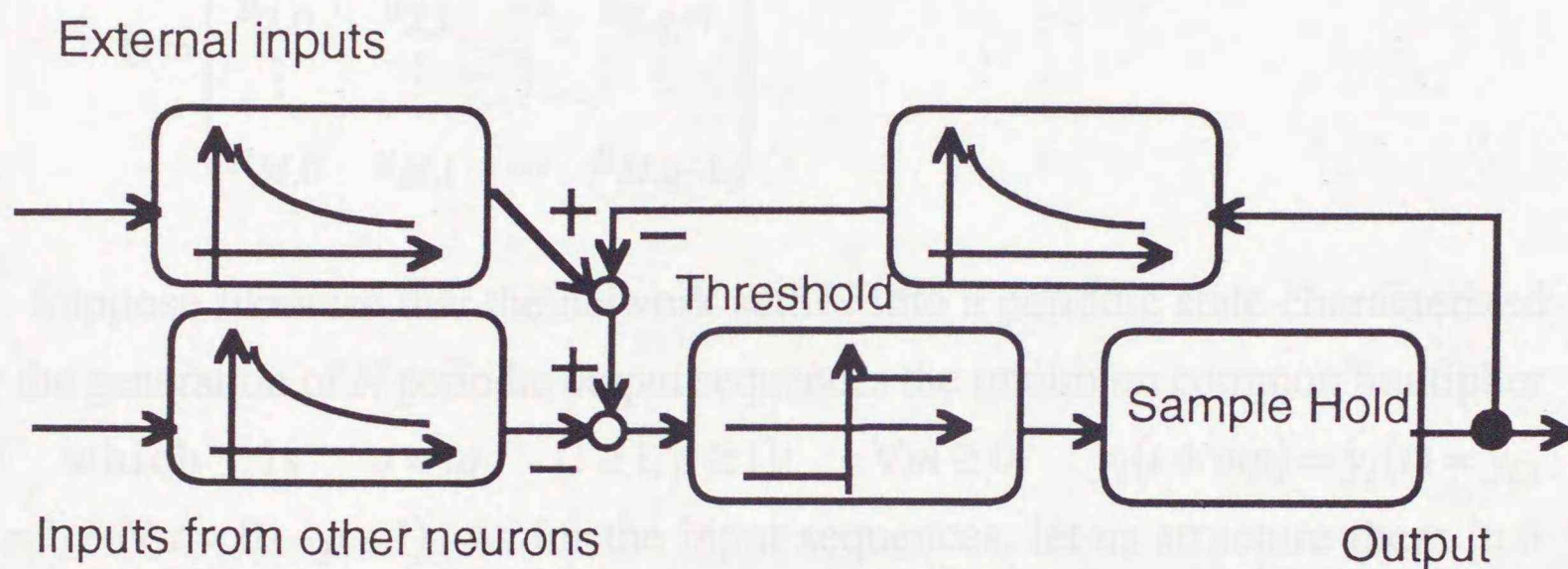


Figure 3.2.1 The neuron model

3.2.2 Dynamics of Network

Suppose the network is stimulated by a set of M periodic inputs, the minimum common multiple of which is q ($q \geq 1$): $\forall m \geq 0 \quad u_i(d + mq) = u_i(d) = u_{i,d}$ ($j = 1 \dots M, d = 0 \dots q-1$). Let us order them in a matrix $\mathbf{U}$ in such a way that each row corresponds to an input and each column represents a time instant. Successive time instants are lined up from left to right.

$$\mathbf{U} = \begin{pmatrix} u_{1,0} & u_{1,1} & \dots & u_{1,q-1} \\ u_{2,0} & u_{2,1} & \dots & u_{2,q-1} \\ \vdots & \vdots & \vdots & \vdots \\ u_{M,0} & u_{M,1} & \dots & u_{M,q-1} \end{pmatrix}$$

Suppose likewise that the network settles into a periodic state characterized by the generation of N periodic output sequences the minimum common multiplier of which is $p = lq$ ($l \geq 1, p \geq 1$): $\forall m \geq 0 \quad y_i(t + mp) = y_i(t) = y_{i,t}$ ($i = 1 \dots N, t = 0 \dots p-1$). As for the input sequences, let us structure them in a matrix $\mathbf{Y}$ where each row corresponds to a neuron and each column represents a time instant. Time instants are lined up in the two matrices.

$$\mathbf{Y} = \begin{pmatrix} y_{1,0} & y_{1,1} & \dots & y_{1,p-1} \\ y_{2,0} & y_{2,1} & \dots & y_{2,p-1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N,0} & y_{N,1} & \dots & y_{N,p-1} \end{pmatrix}$$

Matrices $\mathbf{U}$ and $\mathbf{Y}$ give thus a complete external macroscopic characterization of the network at any time.

The following fundamental result holds:

Theorem: There exists an initial internal state $\mathbf{x}(0)$ of the network such that

a given set of periodic input sequences $\mathbf{U}$ will produce the set of output sequences $\mathbf{Y}$, if and only if:

$$\forall i(i=1 \dots N):$$

$$\forall s(s=0 \dots q-1): \hat{L}_i^s \leq U_i^s - h_i < \tilde{L}_i^s$$

or

$$\forall s(s=0 \dots q-1): \tilde{L}_i^s < U_i^s - h_i \leq \hat{L}_i^s \quad (3.2.4)$$

where:

$$U_i^s = \frac{1}{1-T^q} (\mathbf{e}_i^N)^T \mathbf{Y} \mathbf{U} \mathbf{I}_s^q \mathbf{t}^q \quad (3.2.5)$$

$$L_i^{rq+s} = \frac{1}{1-T^p} (\mathbf{e}_i^N)^T \mathbf{W} \mathbf{Y} \mathbf{I}_{rq+s}^p \mathbf{t}^q \quad (3.2.6)$$

$$\tilde{L}_i^s = \begin{cases} \text{Min}_{r=0 \dots l-1} \{L_i^{rq+s} \mid y_{i,m+s} = 0\} & \text{if it exists} \\ +\infty & \text{otherwise} \end{cases} \quad (3.2.7)$$

$$\hat{L}_i^s = \begin{cases} \text{Max}_{r=0 \dots l-1} \{L_i^{rq+s} \mid y_{i,m+s} = 1\} & \text{if it exists} \\ -\infty & \text{otherwise} \end{cases} \quad (3.2.8)$$

and having introduced the following matrices:

$$(\mathbf{t}^d)^T = (T^{d-1} \quad \dots \quad T \quad 1)$$

$$\mathbf{v} = \begin{pmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,M} \\ v_{2,1} & v_{2,2} & \dots & v_{2,M} \\ \vdots & \vdots & \vdots & \vdots \\ v_{N,1} & v_{N,2} & \dots & v_{N,M} \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,N} \\ w_{2,1} & w_{2,2} & \dots & w_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{N,1} & w_{N,2} & \dots & w_{N,N} \end{pmatrix}$$

$\mathbf{e}_i^d$ is the i -th column of the identity matrix $\mathbf{I}^d$ of order d and $\mathbf{I}_c^d$ is obtained by rotating the columns of the identity matrix in the left direction c times.

Outline of the proof:

I] The necessary condition.

Suppose there is an initial internal state $\mathbf{x}(0)$ of the network such that under the periodic stimulation $\mathbf{U}$ the periodic output $\mathbf{Y}$ will be generated.

It is easy to show by mathematical induction that after $(rq + s + mp)$ time instants (where $m \geq 0, r = 0 \dots l-1$ and $s = 0 \dots q-1$) the internal state $x_i(rq + s + mp)$ of neuron i becomes:

$$x_i(rq + s + mp) = T^{mp} x_i(rq + s) + (1 - T^{mp}) (U_i^s - L_i^{rq+s} - h_i) \quad (3.2.9)$$

It can be looked at $x_i(rq + s + mp)$ as a numerical series of parameter m , which is monotonous and whose limit, for $m \rightarrow +\infty$, is:

$$U_i^s - L_i^{rq+s} - h_i \quad (3.2.10)$$

Two possibilities arise:

if $y_{i,m+s} = 1$ then $\forall m \geq 0 : 0 \leq x_i(rq + s + mp)$. Taking the limit, then:

$$0 \leq U_i^s - L_i^{rq+s} - h_i$$

if $y_{i,m+s} = 0$ then $\forall m \geq 0 : x_i(rq + s + mp) < 0$ which implies that

$$U_i^s - L_i^{rq+s} - h_i \leq 0$$

Assembling all these conditions and taking the extreme values over $r = 0 \dots l-1$, we obtain:

$$\forall i (i = 1 \dots N) \forall s (s = 0 \dots q-1) : \hat{L}_i^s \leq U_i^s - h_i \leq \tilde{L}_i^s \quad (3.2.11)$$

We prove now by reductio ad absurdum that, for a same neuron i , if the equality sign holds at one side of equation (3.2.11) for some $s = s_1$ then it can not hold at the other side of equation (3.2.11) for any s (including $s = s_1$). Suppose that for some neuron i there are two indices a and b (eventually coincident) such that the conditions $\hat{L}_i^a = U_i^a - h_i$ and $U_i^b - h_i = \tilde{L}_i^b$ are satisfied. Then there are two indices h and k ($h, k < 1$) such that $L_i^{kq+a} = \hat{L}_i^a$ and $L_i^{kq+b} = \tilde{L}_i^b$. Without loss of generality, let us assume that $kq + a > hq + b$. By hypothesis, the internal state of neuron i at time $hq + b$ is such that $x_i(hq + b) \geq 0$. It is possible to show that the internal state after $(k - h)q + a - b$ time instants becomes $x_i(kq + a) = T^{(k-h)q+a-b} x_i(hq + b)$. Since $Y_i(kq + a) = 0$, we gather that $0 > x_i(hq + b)$ which is incompatible with the condition $x_i(hq + b) \geq 0$. We have thus:

$$\forall i (i = 1 \dots N):$$

$$\forall s (s = 0 \dots q-1): \hat{L}_i^s \leq U_i^s - h_i < \tilde{L}_i^s$$

or

$$\forall s (s = 0 \dots q-1): \hat{L}_i^s < U_i^s - h_i \leq \tilde{L}_i^s$$

II] The sufficient condition.

Suppose at first that the input and output sequences $\mathbf{U}$ and $\mathbf{Y}$ are such that:

$$\hat{L}_i^s \leq U_i^s - h_i < \tilde{L}_i^s$$

for all neurons $i = 1 \dots N$ and for all indices $s = 0 \dots q-1$. We build the following set of internal states:

$$x_{i,rq+s} = U_i^s - L_i^{rq+s} - h_i \quad (3.2.12)$$

It is easy to check the following properties:

$$f(x_{i,rq+s}) = y_{i,rq+s}$$

$$Tx_{i,rq+s} + \sum_{j=1}^M v_{ij} a_j(k) - \sum_{j=1}^N w_{ij} y_j(k) - h_i(1-T) = x_{i,rq+s+l}$$

By setting thus the initial internal state of the network as follows:

$$x_i(0) = x_{i,0}$$

The network will generate the periodic output sequence $\mathbf{Y}$ when the periodic stimulus $\mathbf{U}$ is applied.

Suppose now that for some neurons $i \in \mathbf{I}^*$ there are indices $s \in S_i^*$ such that:

$$\hat{L}_i^s < U_i^s - h_i \leq \tilde{L}_i^s$$

and that for the remaining neurons we have:

$$\hat{L}_i^s \leq U_i^s - h_i < \tilde{L}_i^s$$

For the former neurons the following values are defined:

$$o_i = \min_{s=0 \dots q-1} \{U_i^s - h_i - \hat{L}_i^s \mid i \in \mathbf{I}^*\} \quad (3.2.13)$$

It is possible to show that the output sequence $\mathbf{Y}$ will be generated from the input sequence $\mathbf{U}$ when the initial state is set as follows:

$$\begin{aligned} u_i^0 - h_i - L_i^0 - o_i < x_i(0) < 0 \quad \text{for } i \in \mathbf{I}^* \\ x_i(0) = x_{i,0} \quad \text{for } i \notin \mathbf{I}^* \end{aligned}$$

where $x_{i,0}$ has been defined in (11).

This completes the proof @

This theorem illustrates the relation existing between the stationary dynamical behavior and the structural parameters of the network, namely the connection weights between neurons w_{ij} , the connection weights to the external inputs v_{ij} and the thresholds h_i . Incidentally, we consider the decay parameter T a preset constant and the size of the network - N and M - fixed.

Once a specific task is assigned to the network in terms of input-output mappings to be performed, the above theorem gives the conditions the structural parameters of the network must satisfy.

3.2.3 Designing Algorithm

Suppose we want to store E different couples of input-output periodic patterns in the network. We must then find a set of parameters satisfying the necessary and sufficient condition (4) for all the patterns:

$$\begin{aligned} \forall f(f=1 \cdots E), \forall i(i=1 \cdots N): \\ \forall s(s=0 \cdots q-1): \hat{L}_i^s(f) \leq U_i^s(f) - h_i < \tilde{L}_i^s(f) \\ \text{or} \\ \forall s(s=0 \cdots q-1): \hat{L}_i^s(f) < U_i^s(f) - h_i \leq \tilde{L}_i^s(f) \end{aligned} \quad (3.2.14)$$

where f is the index characterizing each one. Actually we also have the constraints $w_{ij} \geq 0$, which come from biological considerations. However, in this general approach we will neglect them. Nevertheless, they can be easily integrated in the procedure indicated in the following. The system of inequalities (3.4.1) can be recast as follows:

$$\forall f(f=1 \cdots E), \forall i(i=1 \cdots N), \forall r(r=0 \cdots l-1), \forall s(s=0 \cdots q-1):$$

$$(1 - y_{i,rq+s}(f)) \left(\sum_{j=1}^N c_j(f) w_{ij} - \sum_{j=1}^M c_j''(f) v_{i,j} - h_i \right) > 0 \quad (3.2.15)$$

with

$$c_i(f) = \frac{1}{1 - T^p} \sum_{k=0}^{p-1} T^k y_{j,rq+s-1-k}(f)$$

$$c_i''(f) = \frac{1}{1 - T^q} \sum_{k=0}^{q-1} T^k u_{j,s-1-k}(f)$$

whereby, imposing the strict inequality sign, we decide to discard the solutions for which the equality sign holds in some inequality of (3.2.14).

Hence the problem of storing specific patterns in the neural network is mathematically related to the problem of finding a solution of a system of linear inequalities. Using compact matrices notations, system (3.2.15) becomes:

$$\mathbf{A} \cdot \mathbf{x} > 0 \quad (3.2.16)$$

where $\mathbf{x}$ is the n -dimensional, with $n = N(N + M + 1)$, column vector representing the unknown quantities $w_{i,j}, v_{i,k}$ and $h_i (i = 1 \dots N, j = 1 \dots N, k = 1 \dots M)$ and $\mathbf{A}$ is the $m \times n$, with $m = NEp$, matrix of the corresponding coefficients. The vector inequality $\mathbf{a} > 0$ means that each component of $\mathbf{a}$ is non-negative. We call the set $\mathbf{S}$ of elements $\mathbf{x}$ satisfying the conditions (3.2.16) the solution set.

Each constraint in the condition (3.2.16) defines an open half space in the n -dimensional space. The solution set $\mathbf{S}$ is the cut set of all these half spaces. Since all the hyperplanes delimiting the half spaces contain the origin $\mathbf{o}$ (with $\mathbf{o} \notin \mathbf{S}$), the solution set $\mathbf{S}$ will be either empty or unbounded. Any n -uple of network parameters $w_{i,j}, v_{i,k}$ and h_i belonging to $\mathbf{S}$ will do for the purpose of storing the assigned sequences in the network. However, it is essential to guarantee a global robustness of the neural network and a reasonable insensitivity to

parameter tolerance. This implies that solutions lying far enough (and eventually roughly at the same distance) from each hyperplane should be preferred. This justifies our decision to discard the solutions lying on the border of the solution set of the original system (3.2.14).

The solutions of system (3.2.16), whose distance from any hyperplane defining S is at least b , are also solutions of the system:

$$\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b} \quad (3.2.17)$$

where $\mathbf{b}$ is the n -dimensional column vector of components b , and vice versa. If we call S_b the solution set of system (3.2.17), we have then: $S_b \subseteq S$. Moreover, it is easy to show that: $S_b = \emptyset \Leftrightarrow S = \emptyset$. Therefore we can use system (3.2.17) in lieu of system (3.2.16) with the advantage of the resulting network being more robust.

Many algorithms are known for searching a solution of a linear system of inequalities in the form (3.2.17). The relaxation algorithm, for instance, is a very simple algorithm which is known to converge to a solution in a finite number of steps, if a solution exists at all [21]. Starting from an arbitrary point $\mathbf{x}$, all the inequalities are checked in turn. Whenever an inequality is violated, the candidate solution is mirrored through the corresponding hyperplane. The procedure is iterated until all the constraints are satisfied. The algorithm can be summarized as follows:

Step 1: set $\mathbf{x}$ arbitrarily;

set found to true;

Step 2: for all inequalities ($j = 0 \cdots m$) of (3.2.17)

if $\mathbf{a}_j^T \cdot \mathbf{x} < b$ then

set found to false;

$$\mathbf{x} = \mathbf{x} - 2 \left(\left(\mathbf{a}_j^T \cdot \mathbf{x} + b \right) / \mathbf{a}_j^T \cdot \mathbf{a}_j \right) \mathbf{a}_j$$

Step 3: if found = true then stop

else repeat step 2;

It is worth noting that the system (3.2.17) -or (3.2.16)- can actually be divided into N sub-systems, one for each neuron i , independent from each other. Each sub-system consists of $m' = Ep$ inequalities with $n' = N + M + l$ variables. This subdivision leads to shorter computation times.

3.2.4 Condition for Oscillating a Certain Limit Cycle

Theorem 1 shows the condition of the network for oscillating the sets of limit cycles. However it does not show the initial state for oscillating a certain limit cycle. An initial state decides which limit cycle appears when network memorizes plural limit cycles. In the following, we derive a condition for producing the limit cycle $\mathbf{Y}$ from an initial state $\mathbf{x}(0)$.

Suppose that the simultaneous inequalities (3.2.3) is satisfied and network starts to oscillate the limit cycle $\mathbf{Y}$ from the initial state $\mathbf{x}(0)$, $\mathbf{x}(k)$ is derived from equation (3.2.1) as the below.

$$x_i(rq + s + mp) = T^{mp} x_i(rq + x) + (1 - T^{mp}) (U_i^s - L_i^{rq+s} - h_i) \quad (3.2.18)$$

if supposing $m \rightarrow \infty$ then

$$x_i(rq + s + mp) = U_i^s - L_i^{rq+s} - h_i \quad (3.2.19)$$

Equation (3.2.18) and (3.2.19) show that the network keeps producing the limit cycle $\mathbf{Y}$ if it generates one period of the sequence.

Suppose that the initial state is $\mathbf{x}(0)$ and the network generates the limit cycle $\mathbf{Y}$ from the pattern $\mathbf{y}(n)$, the state $\mathbf{x}(k)$ is the following.

$$\begin{aligned}
x_i(k) &= \begin{cases} x_i(0) & \text{if } k = 0 \\ T^k x_i(0) + (1 + T + \dots + T^{k-1}) h_i(T-1) \\ - \sum_{j=0}^{k-1} \mathbf{w}_i \mathbf{y}((j+n) \bmod p) T^{k-1-j} & \text{if } k \neq 0 \end{cases} \\
\mathbf{w}_i &= \{w_{i,1} \quad w_{i,2} \quad \dots \quad w_{i,N}\}^T \\
\mathbf{y}(k) &= \{y_1(k) \quad y_2(k) \quad \dots \quad y_N(k)\}^T \quad (3.2.20)
\end{aligned}$$

Consequently we get the following condition for starting to oscillate the limit cycle $\mathbf{Y}$ from an initial state $\mathbf{x}(0)$.

$$\begin{aligned}
&\exists n(0 \leq n < p): \\
&\quad \forall k(0 < k < p+1) \forall i(0 < i < N): \\
&\quad x_i(k)(2y_i((k+n) \bmod p) - 1) > 0 \\
&\quad \text{or} \\
&\quad x_i(k) = 0 \text{ and } 2y_i((k+n) \bmod p) > 0 \quad (3.2.21)
\end{aligned}$$

The solutions of the simultaneously inequalities (3.2.21) are the sets of initial states that oscillate the limit cycle $\mathbf{Y}$.

3.2.5 Condition for Converging in a Certain Limit Cycle

We consider the condition for convergence to the limit cycle $\mathbf{Y}$ after sequence $\mathbf{Z}$ of which length is g . Supposing that network produces the limit cycle $\mathbf{Y}$ after producing the sequence $\mathbf{Z}$, the state of network $\mathbf{x}(k)$ is the following.

$$x_i(k) = \begin{cases} x_i(0) & \text{if } k = 0 \\ T^k x_i(0) + (1 + T + \dots + T^{k-1}) h_i(T-1) - \sum_{j=0}^k \mathbf{w}_i^T \mathbf{z}(j) T^{k-1-j} & \text{if } 0 < k \leq g \\ T^k x_i(0) + (1 + T + \dots + T^{k-1}) h_i(T-1) - \sum_{j=0}^k \mathbf{w}_i^T \mathbf{z}(j) T^{k-1-j} - \sum_{j=g}^{k-1} \mathbf{w}_i^T \mathbf{y}((j+n) \bmod p) T^{k-1-j} & \text{if } g < k \end{cases} \quad (3.2.22)$$

$$\mathbf{Z} = \{z(0) \quad z(1) \quad \dots \quad z(g-1)\}$$

$$\mathbf{z}(k) = \{z_1(k) \quad z_2(k) \quad \dots \quad z_N(k)\}^T$$

The condition for convergence to the limit cycle $\mathbf{Y}$ through the sequence $\mathbf{Z}$ is the following.

$$\exists n(0 \leq n < p):$$

$$\exists(\mathbf{Z} \mid \text{a sequence of which length is } g(0 \leq g \leq g_{\max})):$$

$$\forall i(0 < i < N):$$

$$\forall k(0 < k < g-1):$$

$$x_i(k)(2z_i(k) - 1) > 0$$

or

$$(x_i(k) = 0 \text{ and } z_i(k) > 0)$$

$$\begin{aligned}
&\forall k(g < k < g + p + 1): \\
&\quad x_i(k)(2y_i((k+n) \bmod p) - 1) > 0 \\
&\quad \text{or} \\
&\quad (x_i(k) = 0 \text{ and } y_i((k+n) \bmod p) > 0)
\end{aligned}
\tag{3.2.23}$$

The solutions of the simultaneous inequalities (3.2.23) are the set of the initial states $\mathbf{x}(0)$ that converge to the limit cycle $\mathbf{Y}$ through the sequence $\mathbf{Z}$.

3.3 Potential of the Network

As shown in the previous section, the designing stage can be achieved in a straightforward way: once a realizable task has been assigned, it is possible to choose (and we know how) a set of structural parameters such that the desired dynamical behavior is achieved. In the following, we shall illustrate some capabilities of the network and its possible applications.

3.3.1 Capabilities

Example 1

The neural network shown in figure 3.3.1 ($T = 0.5$) comprises two neurons ($N = 2$) and has no externally applied inputs ($M = 0$). Two periodic sequences with period 4 have been stored in the network:

$$Y(1) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \text{ and } Y(2) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In terms of the designing algorithm shown above, this corresponds to solving 2 systems of 8 linear inequalities (some of which are redundant) with 3 variables each.

This example clearly points out the wealth of dynamic behavior, the

outstanding features of which are summarized below, of this network:

a) An output state can lead to different transitions inside a same periodic output sequence. This is the case of:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

in $Y(1)$ and of:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

in $Y(2)$. Such a bifurcation is impossible in a network without time delay, where the present output determines univocally the next output. It is important to keep in mind that in this network, the present output depends on the entire past history, which is recorded in the internal state variables. This kind of behavior was expected since it is a distinctive feature of neurons of the Caianiello-Nagumo type which make up the network.

b) Moreover, same output states can belong to more than one periodic output sequences simultaneously stored in the network. In the example, the outputs:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

both belong to the periodic output sequences $Y(1)$ and $Y(2)$. The generation of the sequences depends on the initial internal state. On the contrary, a single neuron of the Nagumo-Sato type (with no input) generates the same output pattern whatever the initial internal state is, once the neuron parameters are fixed [7-8].

c) Finally, let us consider the output patterns of the two neurons separately.

Those produced in the case of the periodic sequence $Y(2)$ can be generated by two independent neurons, that is by two neurons having no connection between each other, except a common clock for the synchronization. However, it has been proven that a single neuron can not generate the pattern $(1 \ 1 \ 0 \ 0)$ present in the bottom row of $Y(1)$. This is possible only in virtue of the interaction between neurons.

In order to check the operation of the network, we must run it, starting from several different initial internal states.

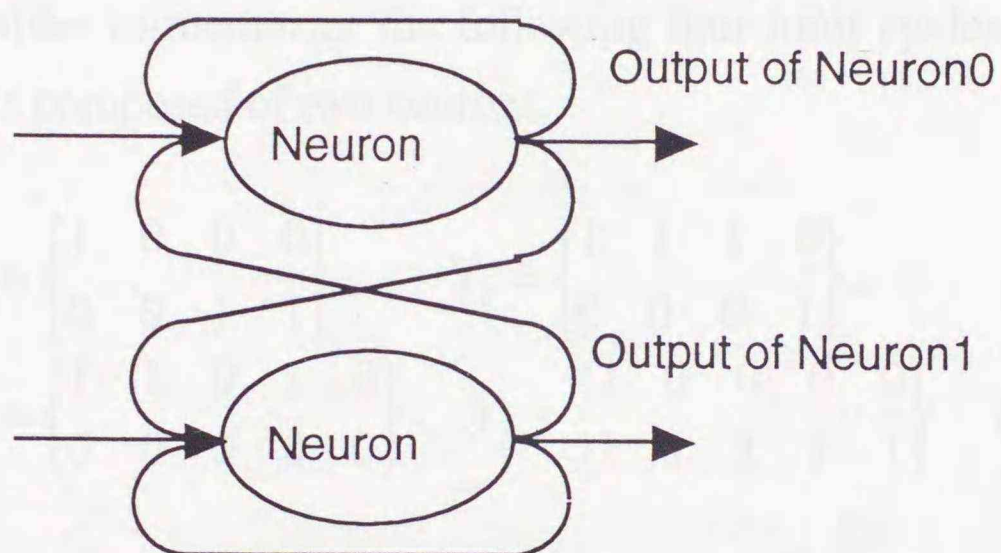


Figure 3.3.1 A neural network with two neurons

Example 2: memorize three limit cycles

Let's consider to memorize the following three limit cycles into a neural network which is composed of two neurons.

$$Y_a = \begin{Bmatrix} 0 & 0 \\ 0 & 1 \end{Bmatrix} \quad Y_b = \begin{Bmatrix} 0 & 1 \\ 0 & 1 \end{Bmatrix} \quad Y_c = \begin{Bmatrix} 1 & 1 \\ 0 & 1 \end{Bmatrix} \quad (3.3.1)$$

We get the following network parameters from the simultaneous inequalities (3.2.4).

$$\begin{array}{lll}
w_{11} = -0.3 & w_{12} = 0.6 & h_1 = -0.3 \\
w_{21} = 0.0 & w_{22} = 0.3 & h_2 = -0.3
\end{array} \quad (3.3.2)$$

The relationship between the initial state and the generated limit cycle is shown in figure 3.3.2, which is derived from the simultaneous inequalities equation (3.2.22) and (3.2.23). The network memorizes equation (3.3.1) and no more limit cycle.

Example 3: memorize 4 limit cycles

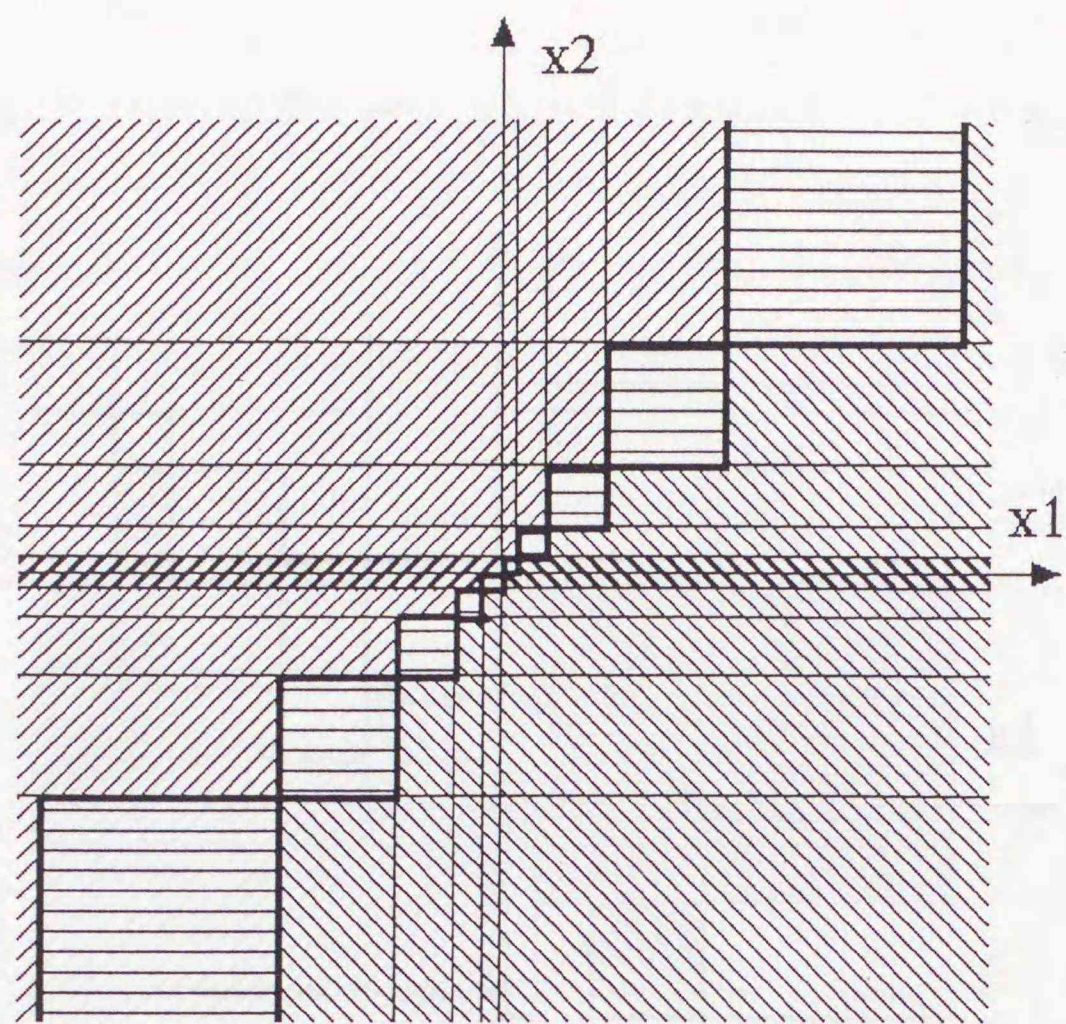
Let's consider to memorize the following four limit cycles into the neural network that is composed of two neurons.

$$\begin{array}{ll}
\mathbf{Y}_d = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{Bmatrix}, & \mathbf{Y}_e = \begin{Bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}, \\
\mathbf{Y}_f = \begin{Bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{Bmatrix}, & \mathbf{Y}_g = \begin{Bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{Bmatrix}
\end{array} \quad (3.3.3)$$

We get the following network parameters from the simultaneous inequalities (3.2.3).

$$\begin{array}{lll}
w_{11} = -7.5 & w_{12} = -9.00 & h_1 = 15.3 \\
w_{21} = 3.00 & w_{22} = 4.50 & h_2 = -6.30
\end{array} \quad (3.3.4)$$

The relationship between the initial state and the generated limit cycle is shown in Figure 3.3.3 which is derived from the simultaneous inequalities equation (3.2.22). The network memorizes equation (3.3.3) and no other limit cycle.



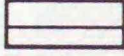
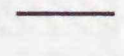
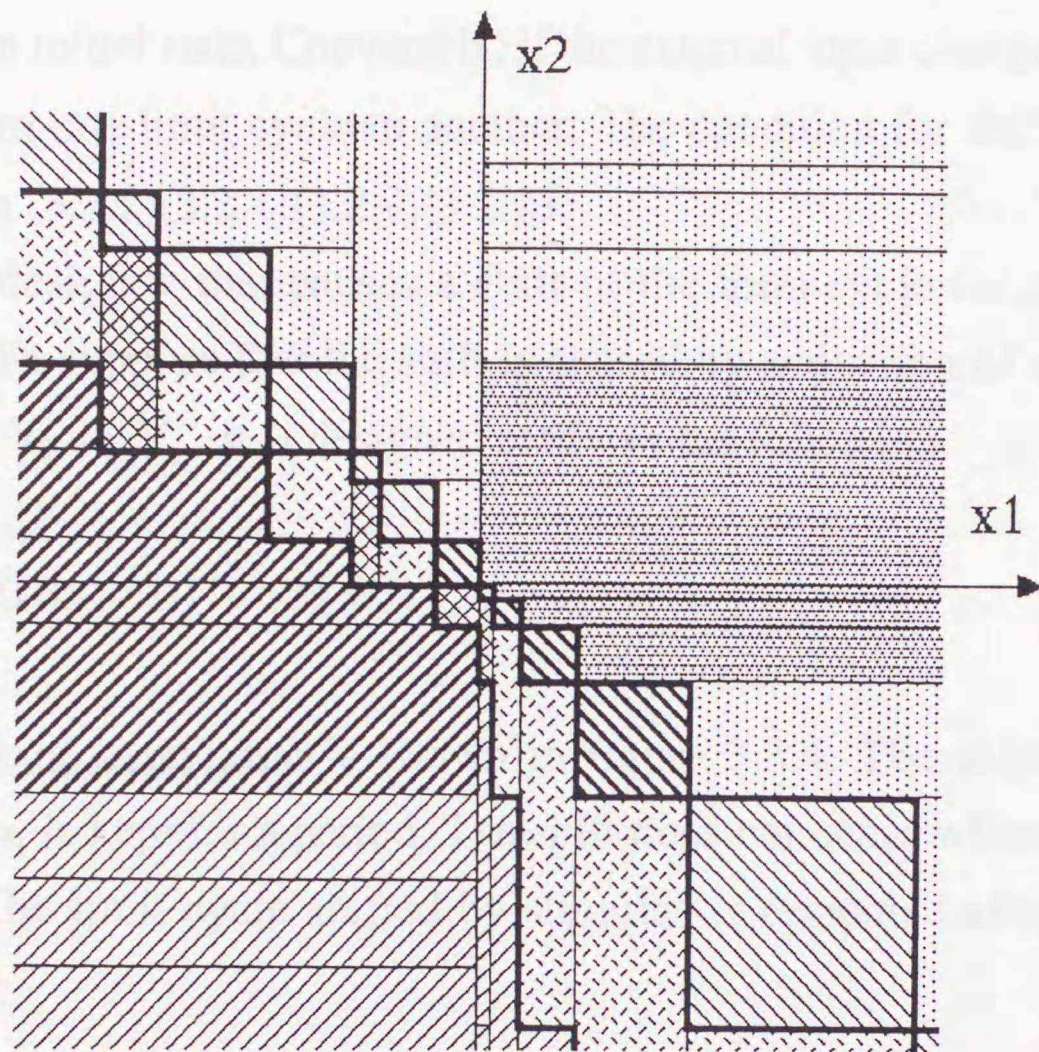
-  area enters the limit cycle Y_a after finite steps
-  area generates the limit cycle Y_a
-  area enters the limit cycle Y_b after finite steps
-  area generates the limit cycle Y_b
-  area enters the limit cycle Y_c after finite steps
-  area generates the limit cycle Y_c
-  boundary between limit cycles
-  boundary between some conditions

Figure 3.3.2 The map of the initial state



-  area enters the limit cycle Y_d after finite steps
-  area generates the limit cycle Y_d
-  area enters the limit cycle Y_e after finite steps
-  area generates the limit cycle Y_e
-  area enters the limit cycle Y_f after finite steps
-  area generates the limit cycle Y_f
-  area enters the limit cycle Y_g after finite steps
-  area generates the limit cycle Y_g
-  boundary between limit cycles
-  boundary between some conditiones

Figure 3.3.3 The map of the initial state

Example 4 : shifting between limit cycles by the external input

When the external input remains U , the network keeps the limit cycle that is decided by the initial state. Conversely, if the external input changes, the network may shift from one limit cycle to another. The condition for shifting is derived from equation (3.2.2)(3.2.4)(3.2.9)(3.2.23).

Here we design v that causes a shift of the limit cycle for the network of Example 1. We suppose that the shift is caused by activation of u_1 , from Y_a to Y_b and from Y_b to Y_c . v is designed easily as the following.

$$V = \{0.051 \ 0\}^T \quad (M=1) \quad (3.3.5)$$

The result of simulation is shown in figure 3.3.4. The shift occurs when activation of u_1 is kept for a period. The shift does not occur when u_1 is activated temporally. The limit cycle shifted by stimulus is remained after the stimulus goes out.

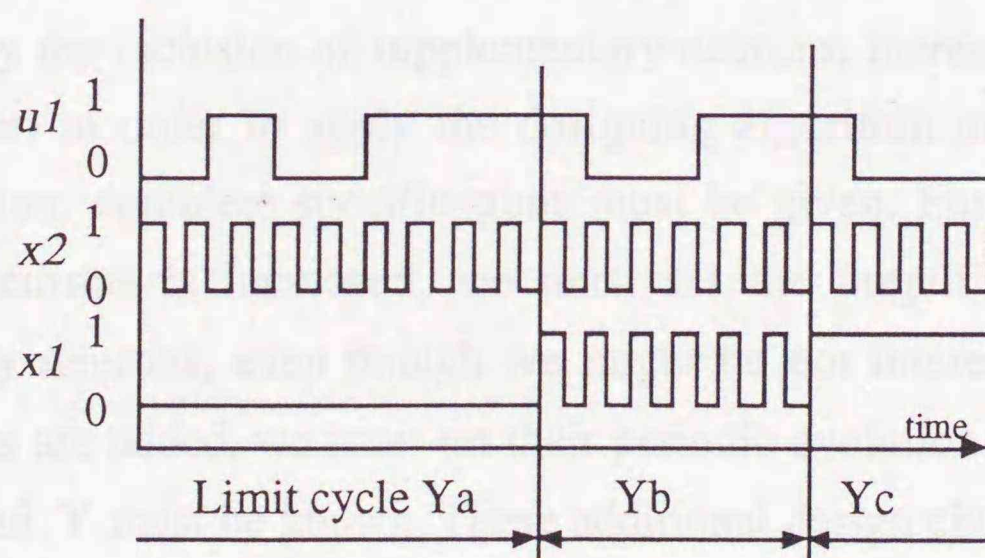


Figure 3.3.4 Transition of limit cycles evoked by external input

3.3.2 Performance

Now, let us try to quantify the performance of a two-neurons autonomous network like the one of Figure 3.3.1. We restrict our attention to the storage of couples of periodic output sequences.

We have tried to store alternatively other couples with period less or equal than $P_{\max} = 4$ and we have found that most of them are not storable. This is because the number of constraints largely exceeds the number of variables. However, if the maximum period $P_{\max}$ is reduced, then the ratio between the number of storable couples and the total number of couples drastically increases. Mathematically, to reduce $P_{\max}$ corresponds to reducing the maximum number of constraints while keeping the same number of variables. For $p = 2$, all the couples of periodic sequences can be stored. In this case the variables are $2 \times 3 = 6$ while the constraints are at most $2 \times 4 = 8$.

Given a maximum period $P_{\max}$ if we want to increase the number of storable couples, we must add external inputs, i.e., it is necessary to increase M . In mathematical terms, this means the addition of variables. Another solution is represented by the inclusion of supplementary neurons, increasing N . In both cases however, in order to apply the designing algorithm introduced in the previous section, complete specifications must be given. For instance, if the number of neurons is increased, we must set the output patterns of the supplementary neurons, even though we might be not interested in them. If external inputs are added, we must set their periodic evolution. In other words, matrices $\mathbf{U}$ and $\mathbf{Y}$ must be known. These additional design choices may not be easy *a priori*.

3.4 Conclusions

A simple time discrete binary output neural network model is introduced and its dynamics is analyzed. The neuron model includes features like the

refractory mechanism and the afferent signals integration phenomenon and is capable of complex oscillatory dynamics. Designing algorithm for storing specified mappings between input and output periodic sequences is derived. The condition for convergence in a certain limit cycle is derived. By studying a basic example in support of the theoretical results, we have shown that the network is capable of high dynamic performances. It is thought that model has great potentialities.

- [1] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [2] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [3] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [4] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [5] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [6] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [7] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [8] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [9] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [10] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).
- [11] J. J. Hopfield, "Neurons with graded response have collective properties that enable them to perform associative functions," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092 (1984).

References of Chapter 3

- [1]N.Kopell, Toward a theory of modelling central pattern generators, in Neural Control of Rhythmic Movement in Vertebrates(A.H.Cohen, S.Grillner, and S.Rossignol, Eds.), New York:Wiley, pp.369-413(1987).
- [2]W.O.Friesen, and R.A.Pearce, Mechanisms of intersegmental coordination in leech locomotion, Semin. Neurosci., vol.5,pp.41-47.
- [3]D.Kleinfeld, K.R.Delaney, M.S.Fee, J.A.Flores, D.W.Tank and A.Gelperin, Dynamics of propagating waves in the olfactory network of terrestrial mollusc: An electrical and optical study, J.Neurophysiol., vol.72,pp.1402-1409(1994).
- [4]Z.Li and J.J.Hopfield, Modeling the olfactory bulb and its neural oscillatory processing, Biol. Cybern., vol.61, pp.379-392.
- [5]J.S.Kauer, Contributions of topography and parallel processing to odor coding in the vertebrate olfactory pathway, Trends Neurosci., vol.14, pp.79-85(1991).
- [6]W.J.Freeman, Simulation of chaotic EEG patterns with a dynamic model of the olfactory system, Biol. Cybern., vol.56,pp.139-150(1987).
- [7]J.Ambros-Ingerson, R.Granger, and G.Lynch, Simulation of paleocortex performs hierarchical clustering, Science, vol.247,pp.1344-1348(1990).
- [8]S.M.Botros and E.N.Bruce, Neural network implementation of a three-phase model of respiratory rhythm generation. Biol. Cybern., vol.63:pp.143-153.
- [9]A.L.Hodgkin, A.F.Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J.Physiol.,vo.117,pp.500-544.
- [10]K.Pakdaman,J.F.Vibert, E.Boussard and N.Azmy, Single Neuron with Recurrent Excitation:Effect of the Transmission Delay, vol.9, pp.797-818(1996).
- [11]R.D.Hangartner, P.Cull, A ternary logic model for recurrent neuromime

networks with delay, Biological Cybernetics, Vol.73, pp.177-178(1995).

[12]Y.Kuramoto, Collective synchronization of pulse-coupled oscillators and excitable units, Physica D, vol.50,pp.15-30(1990)

[13]H.S.Strongz, R.E.Mirollo, and P.C.Matthews, Coupled nonlinear oscillators below the synchronization threshold -relaxation by generalized Landau damping, phy. Rev. Lett., vol.68,pp.2730-2733(1992).

[14]J.Nagumo and S.Sato, On a Response Characterisitic of a Mathematical Neuron Model, Kybernetik, Vol.10, pp.155-164, (1972).

[15]T.Yoshida, On Periodic Responses of a Mathematical Neuron model, Biological Cybernetics, Vol.52, pp.383-390, (1985).

[16]E.R.Caianiello and A.D.Luca, Decision Equation for Binary Systems. Application to Neuronal Behavior, Kybernetik, vol.3, pp.33-40(1966).

[17]Mitsuo WADA, Cyclic Sequences Generated by Neuron Model, Electronics and Communications in Japan, 65,2, pp.1-10(1983).

[18]T.Kumagai, R.Hashimoto, M.Wada, Learning of Limit Cycles in Discrete-Time Neural Network with Time Delay, Proceeding of the 3rd Intelligent System Symposium, JSME No.930-60, pp.121-124(1993).(in Japanese)

[19]S.Gardella, T.Kumagai, R.Hashimoto, M.Wada, On the dynamics and Applications of a DiscreteTime Binary Neural Network with Time Delay, Journal of Intelligent and Fuzzy Systems, Vol.2, pp.221-228(1994).

[20]T.Kumagai, R.Hashimoto, M.Wada, Learning of Limit Cycles in Discrete-Time Neural Network, Neurocomputing Journal, Vol.13, No.1, pp.1-10(1996).

[21]H.Harrer, J.A.Nossek and F.Zou, A Learning Algorithm for Time-Discrete Cellular Neural Networks, Proceedings of IJCNN'91, pp.717-722, Singapore, November (1991).

Chapter 4

Learning of Recurrent Neural Networks Using Genetic Algorithm

4.1 Introduction

Global search capability of genetic algorithm is attractive to researchers who tackle with recurrent neural networks, because a local minimum problem is the largest obstacle in a learning problem of recurrent networks. Many researchers have applied genetic algorithm to learning problems of neural networks[1][2][3]. However it is reported that genetic algorithm is often inefficient for network learning [4]. It is due to incomplete behavior of cross over operator. In network learning cross over operator often produces weak children, because neural networks can take various structure for the same function. Genetic algorithm without cross over operator becomes a simple multi point search algorithm.

Another problem peculiar to networks is that genetic algorithm can not copy between loci. Binary output networks, such as proposed in previous section,

must express a continuous value through population expression. In these situations, some neurons need to acquire the same or similar functions as others. It is assumed that continuous output network also needs to acquire similar function to describe a complex function. However genetic algorithm does not have any copy function between loci. Therefore networks must acquire the same connection or function repeatedly and independently.

In this section *internal copy* operator and *inter individual copy* operator are introduced to solve this problem. Internal copy operator is a genetic algorithm operator which copies a portion of a gene to another part of the same gene [5]. Inter individual operator copies a portion of a gene between individuals. These two operators are different from cross over operator in the following two points. First, cross over operator usually exchange half of gene. Contrastively proposed operators copy only a portion of a gene. Second, cross over operator can not copy between loci, but proposed operator copies between loci.

4.2 Genetic Algorithm for Recurrent Neural Networks

4.2.1 Recurrent Neural Network

Here, genetic algorithm is applied to a learning problem of the following time discrete binary output neural network model that is proposed in chapter 3[6][7].

$$\begin{aligned} x_i(k+1) &= Tx_i(k) + \sum_{j=1}^M v_{ij}u_j(k) - \sum_{j=1}^N w_{ij}y_j(k) - \sum_{j=1}^N w_{ij}y_j(k) - h_i(1-T) \\ y_i(k) &= f(x_i(k)) \end{aligned} \quad (4.2.1)$$

where $y_i(h)$ represents output of the i -th ($i=1\dots N$) neuron at time h and $u_i(h)$ the j -th ($j=1\dots M$) external input at time h ; v_{ij} is the connection weight from the j -th externally applied input to the i -th neuron; w_{ij} is the connection weight from the j -th neuron to the i -th neuron; f is the step function: $f(x)=0$ for $x<0$ and $f(x)=1$ for $0\leq x$; h_i is the internal threshold and $T(0\leq T<1, \text{ in this study } T=0.5)$ is the decay parameter.

4.2.2 Coding of the Gene

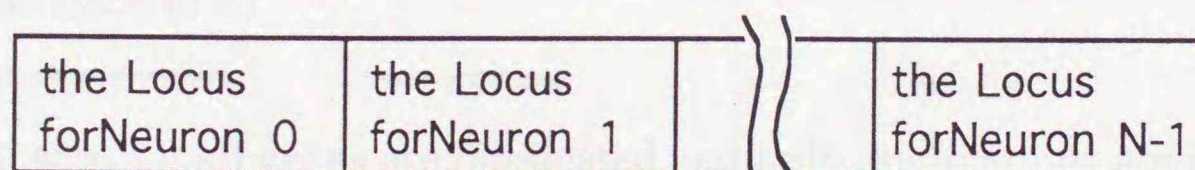
There are two ways to code neural networks in a gene. One way is to describe production rules of a neural network [3]. This method can describe effectively various network structure, however it can not describe actual values of each parameter. Therefore this method is not fit with recurrent networks because there is no universal learning method for recurrent networks except probabilistic search algorithm such as genetic algorithm. It is much time consuming to apply genetic algorithm to individuals produced by genetic algorithm.

Other way is a direct genotype to phenotype mapping where every parameter of a network is described in gene by binary or a real number. This study adopts this coding method. Figure 4.2.1 shows the adopted coding. Each locus has

weight and threshold values of a neuron. Each locus is located in a gene sequentially in order of logical locus number. In locus, each weight is described in order of logical neuron number, i.e., i in equation (4.2.1). In this coding, the same element of each locus describes a connection from the same neuron. For example the second element of each locus means always a weight value of connection from neuron 0.

It is known that direct genotype to phenotype mapping encounters problems of scalability[8]. The number of parameters necessary to encode a network increases exponentially with a number of neurons. Consequently, the search space also increases exponentially. It is expected that internal copy operator and inter individual copy operator solve this problem. They can prevent from relearning of the same function or architecture by utilizing genetic resources effectively. This aspect will be discussed in the following sections through experiments.

the gene coding



the locus of a individual neuron

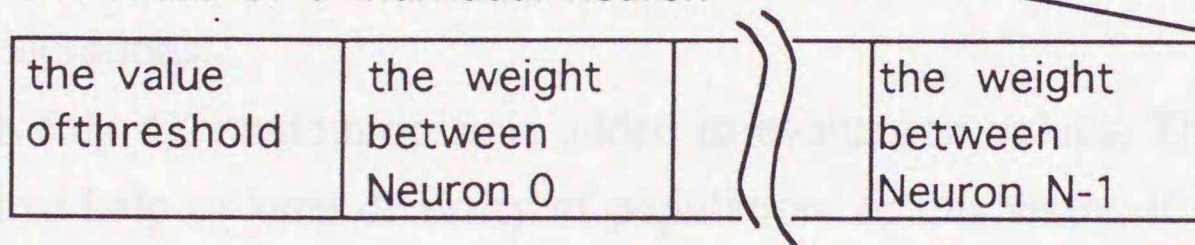


Figure 4.2.1 The coding of the gene

4.2.3 Genetic Algorithm with Copy Operators

The genetic algorithm composed in this study is illustrated in Figure 4.2.2. This is standard genetic algorithm except that internal and inter individual operators are introduced. The following is the detail of a procedure.

- 1) All genes in the population are initialized with zero.
- 2) All genes are Duplicated.
- 3) An operator is applied to each duplication. For each gene, an applied operator is chosen from a set of operators by equal probability.
- 4) An evaluation value is calculated for each gene with equation (4.2.2).

$$est = f()(1 + rnd(kn)) \quad (4.2.2)$$

where $f()$ is an evaluation function given by designer.

$rnd(k_n)$ is an equally random function whose range is $(-k_n, k_n)$.

- 5) Select half of genes for the next generation in order of evaluation.
- 6) Go back to 2)

At the step 2), all genes are duplicated and only duplications are modified in stage 3). If genetic operators produce weak descent, a present generation survived to next generations.

In the step 4), random noise is added to evaluation values. This noise is supposed to help to keep diversity of population. In this study, if there is no particular description, $k_n = 0.1$.

In the step 3) of the above algorithm, as a set of operator, we used some of the following four operators, the Mutation, Cross over, Internal Copy and Inter-Individual copy operators. These operators are illustrated in Figure 4.2.2.

(a) Mutation:

This operator modifies each element in a gene with probability m

according to the following equation (Figure 4.2.3 (a)):

$$\Delta\omega = rnd(k_\omega) \quad (4.2.3)$$

where $\Delta\omega$ is the quantity of modification of each parameter, and $rnd(k_\omega)$ is an equally random function whose range is $(-k_\omega, k_\omega)$.

(b) Cross Over:

This operator is a standard two-point cross over operator (Figure 4.2.3 (b)). However, in this study, only a duplication gene is modified. The concrete procedure is the following. First a parent gene is chosen from original gene by random number and it is divided into two parts which are same length by two points cutting. Then one of two parts is copied to the same locus of a duplication gene.

(c) Internal Copy:

This operator copies a consecutive block of loci to another portion of the same gene. A source and destination are chosen by random numbers. The length of a copied locus block is decided as follows.

$$locilengh = Rnd(NB) \quad (4.2.4)$$

or

$$locilengh = NB \quad (4.2.5)$$

where $Rnd(NB)$ is an equally random integer function which range is $[1, NB]$.

(d) Inter-Individual Copy:

This operator copies a consecutive block of loci as same as internal copy. However a source gene is chosen from the original genes.

Each operator will be denoted in the following section by 'M', 'C', 'IC' and 'II'. The combination of operators will be expressed by '+'. For example 'M+IC' means the combination of mutation and internal copy operator.

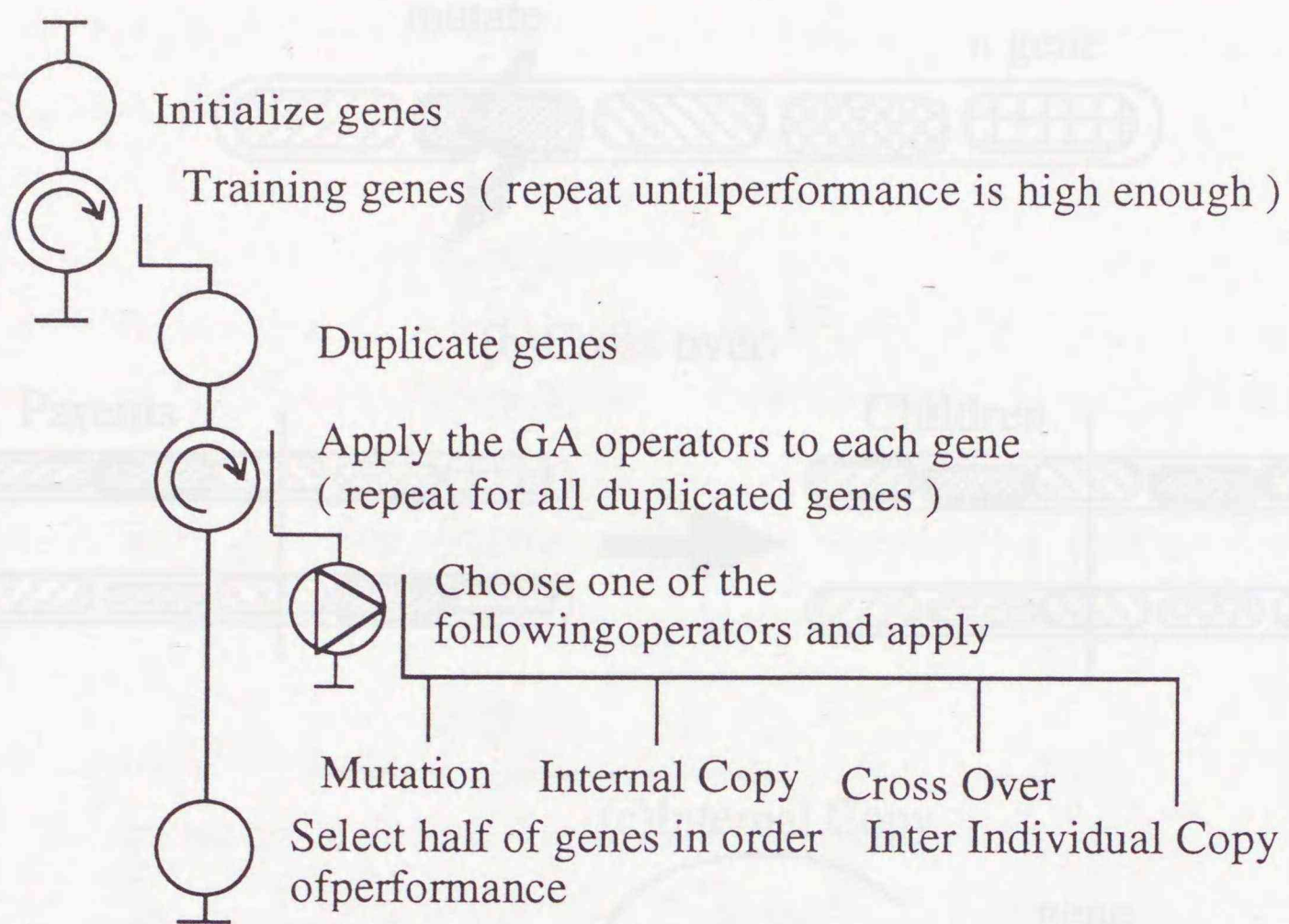


Figure 4.2.2 The genetic algorithm with copy operators

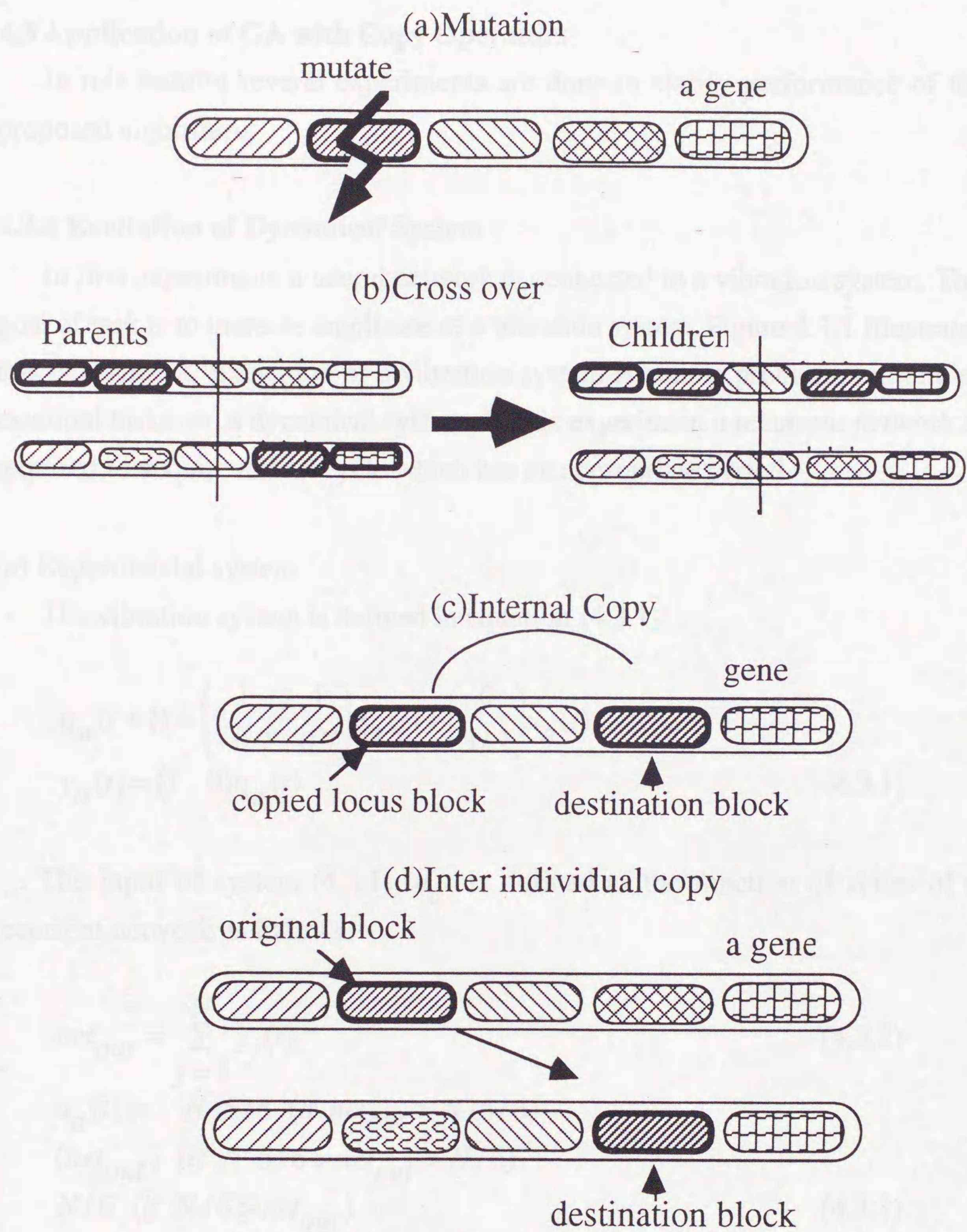


Figure 4.2.3 Genetic algorithm operators

4.3 Application of GA with Copy Operators

In this section several experiments are done to clarify performance of the proposed algorithm.

4.3.1 Excitation of Dynamical System

In first experiment, a neural network is connected to a vibration system. The goal of task is to increase amplitude of a vibration system. Figure 4.3.1 Illustrates this experiment. Excitation of a vibration system is simple problem, but one of essential tasks on a dynamical system. In this experiment a recurrent network is required to acquire a limit cycle which has an appropriate period.

(a) Experimental system

The vibration system is defined in equation (4.3.1),

$$\begin{aligned} \mathbf{q}_a(t+1) &= \begin{pmatrix} 1 & 1 \\ -0.9 & 0.7 \end{pmatrix} \mathbf{q}_a(t) + \begin{pmatrix} 0 \\ 3/2 \end{pmatrix} u_a(t) \\ y_a(t) &= (1 \ 0) \mathbf{q}_a(t) \end{aligned} \quad (4.3.1)$$

The input of system (4.3.1), u_a , is defined as the function of states of a recurrent network as follows.

$$net_{out} = \sum_{j=1}^N y_j(t) \quad (4.3.2)$$

$$\begin{aligned} u_a(t) &= \begin{cases} N \cdot 5/6 & (\text{if } net_{out} > N \cdot 5/6) \\ (net_{out}) & (\text{if } N \cdot 5/6 > net_{out} > N/6) \\ N/6 & (\text{if } N/6 > net_{out}) \end{cases} \end{aligned} \quad (4.3.3)$$

Note population coding is adopted here for the coding of u_a . Therefore the network can take lengthy expression (equation (4.3.2) (4.3.3)). The evaluation

function is shown in equation (4.3.4).

$$f_a = \max(y_a) - \min(y_a) \quad (4.3.4)$$

Equation (4.3.4) is a function of time development of equation (4.3.1). All networks are initialized with equation (4.3.5) before evaluate equation (4.3.4).

$$\mathbf{q} = \{0, 0\}^T \quad x_i = 0 \quad \{i | N/2 + 1 \leq i \leq N\} \quad (4.3.5)$$

The population size is 24. The parameters of the network are decided as follows:

$$NB=3, \quad N=12 \quad (4.3.6)$$

Length of a copy block is decided by Equation (4.2.5).

Recurrent network is trained with (M+C), (M) and (M+I).

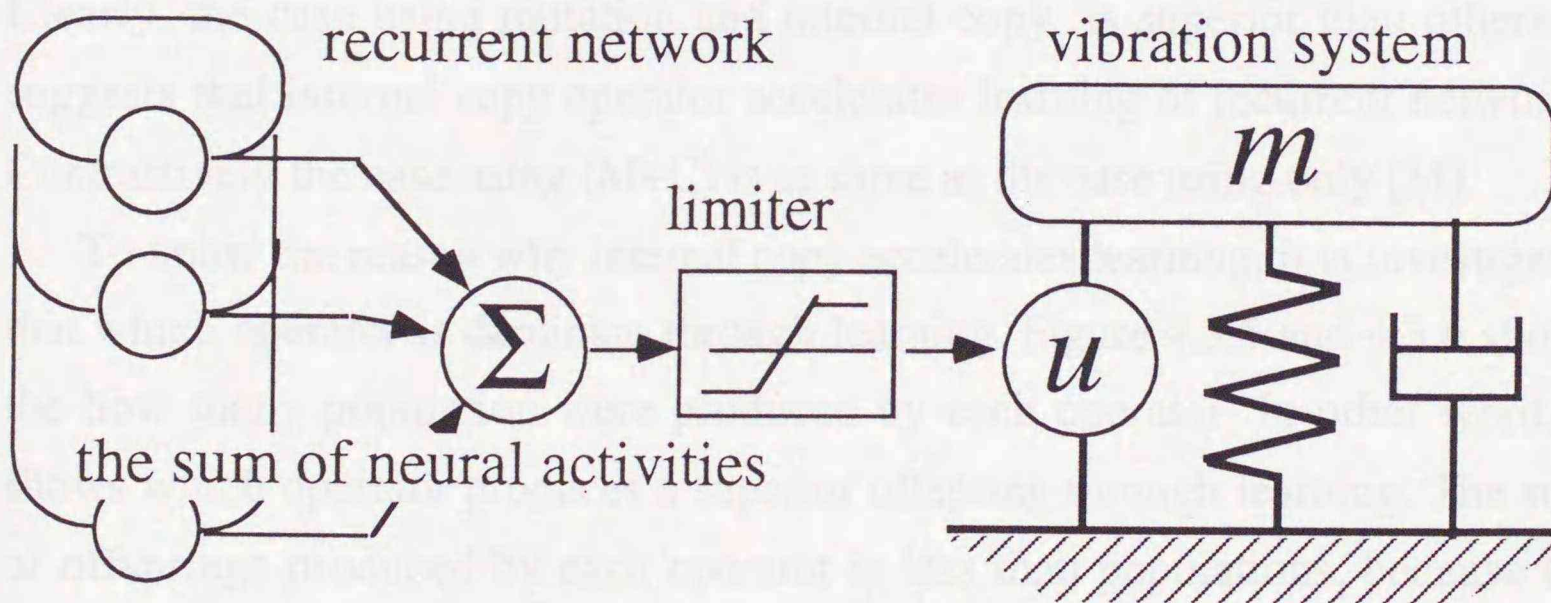


Figure 4.3.1 An excitation problem of a vibration system

(b) Results of experiments

After enough learning, the recurrent network acquires the limit cycles which excites the vibration system. Figure 4.3.3 shows the limit cycles acquired by a network through 200 generations learning with mutation and internal copy operators. The limit cycles acquired by the networks is similar to the optimal solution, bang-bang control: period of the limit cycle is as same as the eigenfrequency, and it forms rectangular wave. In stead of equation (4.3.2), equation (4.3.7) is also available as the way of population coding.

$$\begin{aligned} net_{out} &= net_a - net_b \\ net_a &= \sum_{i=0}^{N/2-1} y_i, \quad net_b = \sum_{i=N/2}^N y_i \end{aligned} \quad (4.3.7)$$

The recurrent networks acquired a limit cycle which similar to optimal solution again with the coding (4.3.7) (Figure 4.3.4).

Figure 4.3.2 shows the comparison of evaluation values at 200 generations. Average, maximum and minimum values through 5 experiments are plotted. Clearly, the case using mutation and internal copy is superior than others. It suggests that internal copy operator accelerates learning of recurrent networks. Contrastively the case using (M+C) is as same as the case using only (M).

To know the reason why internal copy accelerates learning, it is investigated that which operator is dominant through learning. Figure 4.3.5 and 4.3.6 shows the how many population were produced by each operator. In other word, it shows which operator produces a superior offspring through learning. The sum of offsprings produced by each operator is less than populations, because the genetic operations often made weak offsprings and previous generations survived. When (M+IC) is applied (Figure 4.3.5), mutation is dominant in the beginning of learning. Internal copy operator is comparatively active in the middle of learning. End of the learning, nothing works much. When (M+C) is applied

(Figure 4.3.6), mutation is dominant in the beginning of learning and cross over operator dominant in the end of learning.

When (M+IC) is applied, what acquired in network at each generation is shown in Figure 4.3.7, 4.3.8, 4.3.9 and 4.3.10. Left column of these figures shows a response of a network. Right column shows graphical representation of weight matrix of networks: where z axis means strength of weights, x and y axis means sources and destinations of connections. At first, almost all elements of weight matrix are zero. Responses of networks are flat. In the beginning of learning, some parts of weight matrix are modified by mutation and a part of network begins to oscillate. In the middle period, repetitive structure appears in weight matrix and networks oscillate entirely. End of learning, a weight matrix loses repetitive structure slightly, and responses of networks approximate the optimal solution. Those phenomena will be interpreted roughly as follows.

In the beginning of learning, mutation operator works hard and finds out a fragment of an oscillative structure. In the middle period, internal copy operator copies it to entire network. The duplication of a fragment makes repetitive structure on weight matrix. Then the network oscillates entirely. End of the learning, the network acquires quasi optimal solution by mutation. The above interpretation explains that why internal copy operator can accelerate the learning. In the middle period, networks seem to have redundancy. However, at the end period, quasi optimal solution is acquired by mutation.

On the other hand, when (M+C) is applied, after the middle period of learning, the population often lost diversity. This is one of the reason that why cross over operator produced so many offsprings. Because if all populations are same or similar, offsprings produced by cross over is not different from their parents. This inconvenient might be caused by a small population. A large population may be activate cross over operator. However large population takes much computation time.

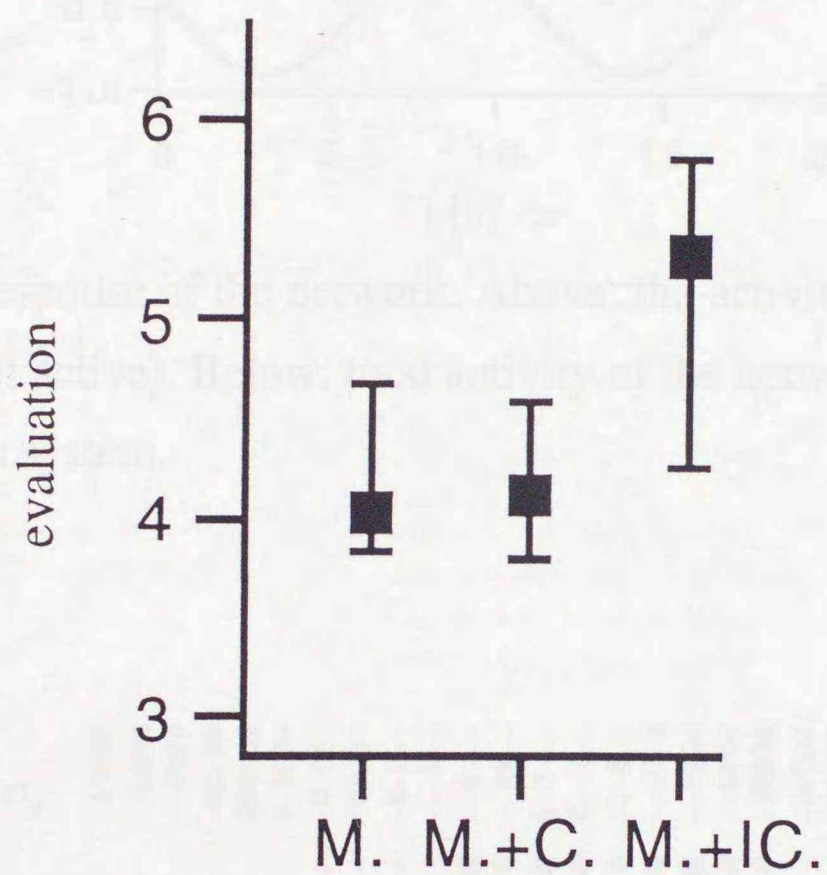


Figure 4.3.2 Evaluation values after training.

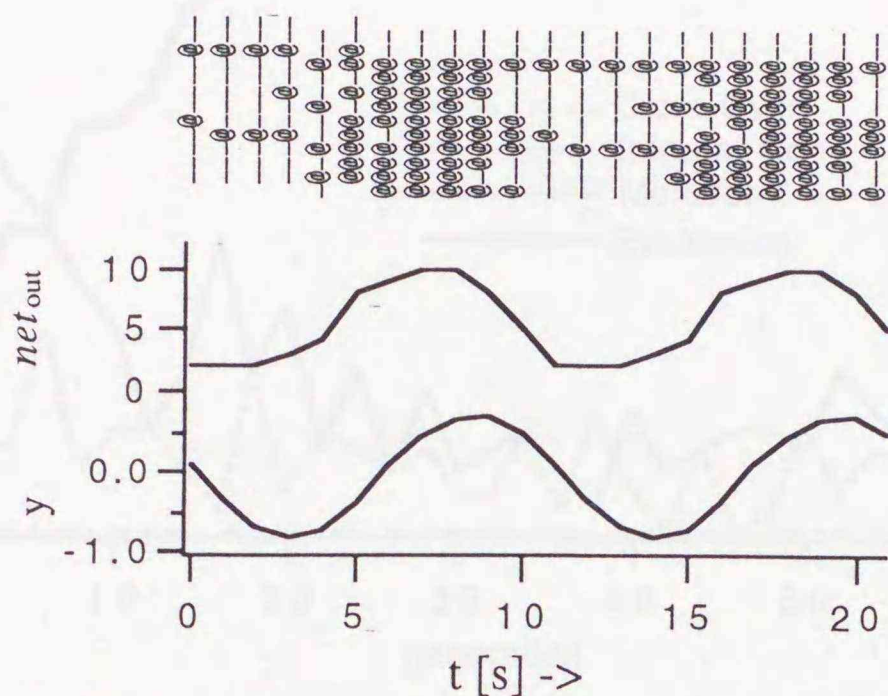


Figure 4.3.3 Response of the network. Above: the activity of each neuron (@ active, | not active). Below: total activity of the network and response of the vibration system.

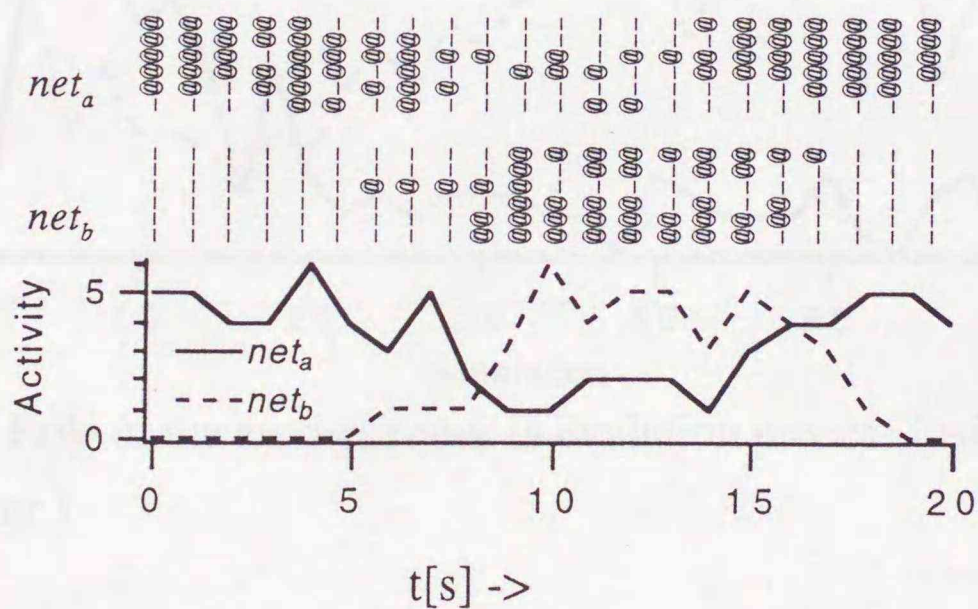


Figure 4.3.4 Response of the network. Above: the activity of each neuron (@ active, | not active). Below: total activity of the network

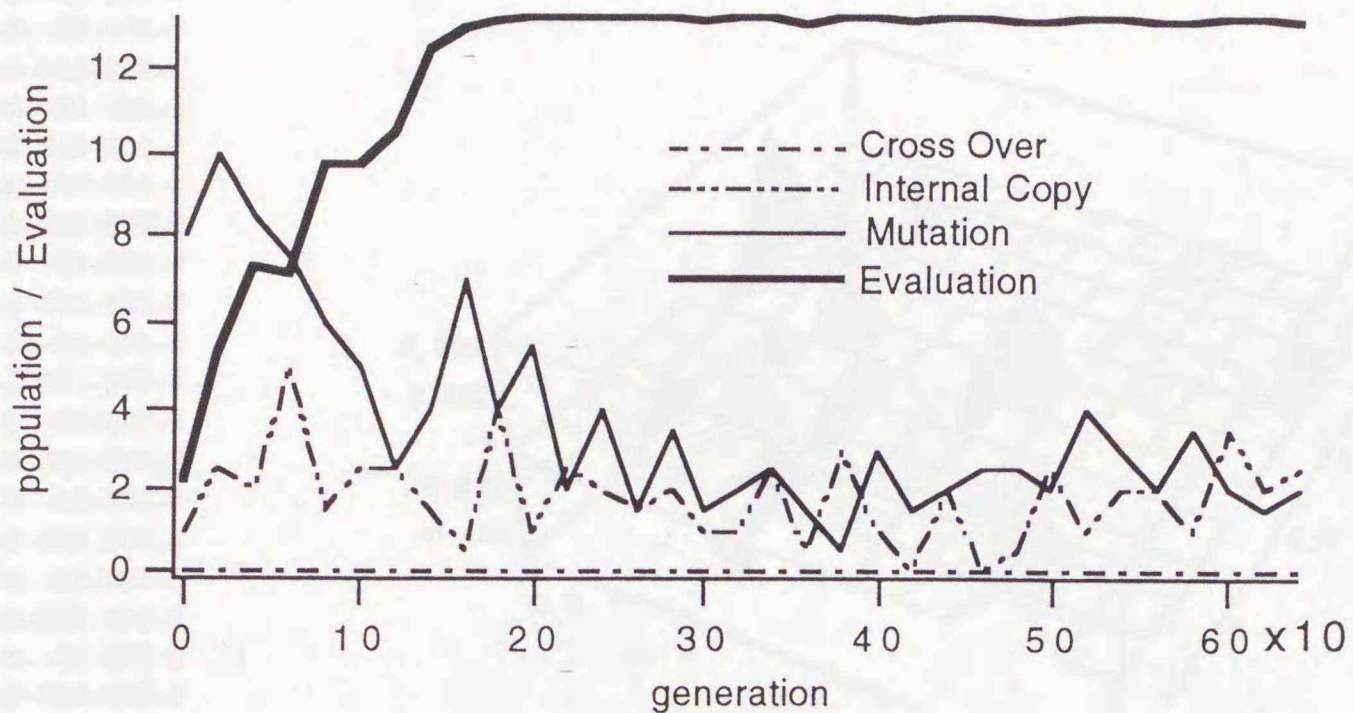


Figure 4.3.5 Role of the each operator in evolution process (using mutation and internal copy)

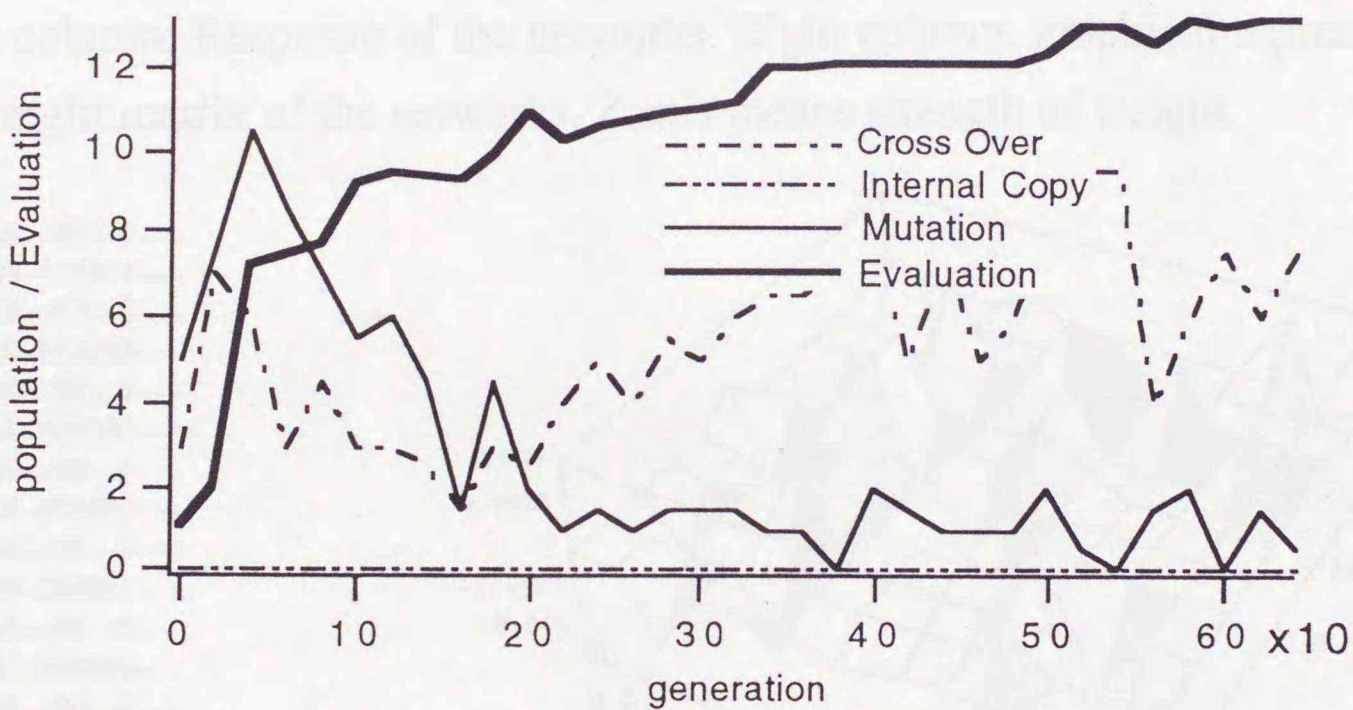


Figure 4.3.6 Role of the each operator in evolution process (using mutation and cross over)

-0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0
 -0-000-000-0
 -0-00-000-0

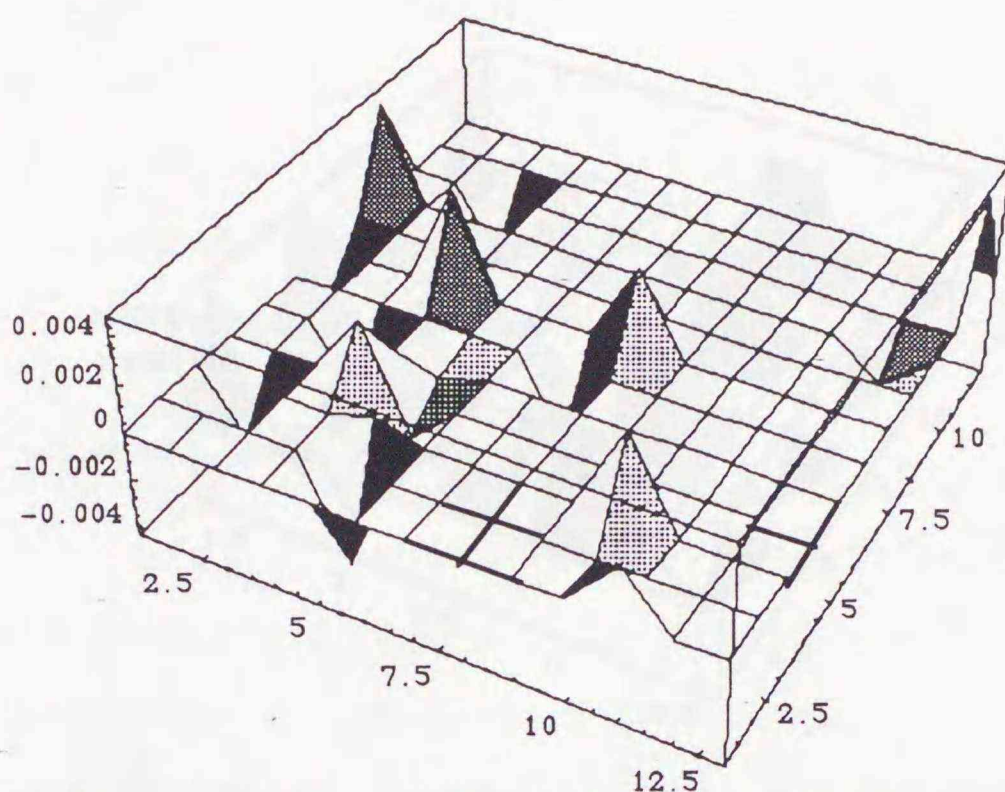


Figure 4.3.7 Response and network structure just after the start of learning.
 Left column: Response of the networks. Right column: graphical representation of weight matrix of the networks. Z axis means strength of weight.

00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---
 00-00-0---
 00-000000---

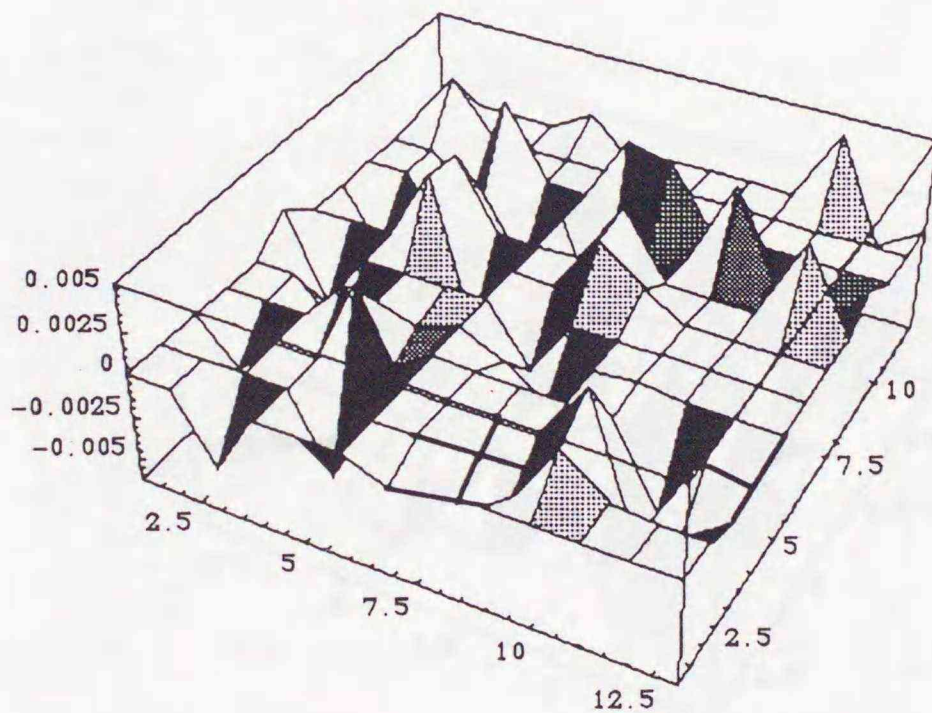


Figure 4.3.8 Response and network structure in the beginning of leaning.
 Left column: Response of the networks. Right column: graphical representation of weight matrix of the networks. Z axis means strength of weight.

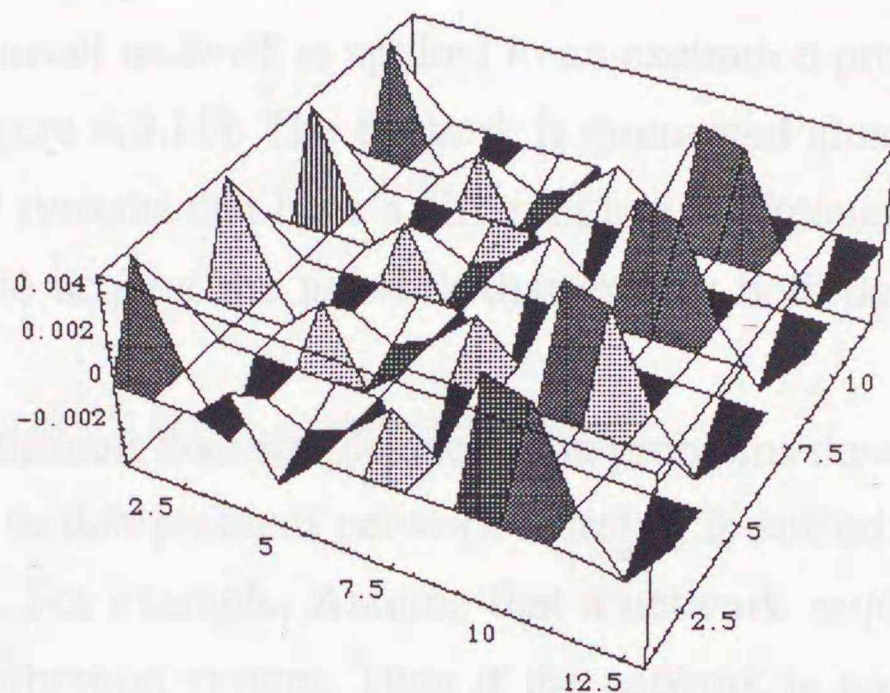
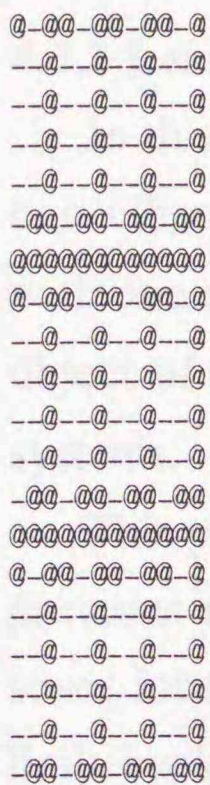


Figure 4.3.9 Response and network structure in the middle period.

Left column: Response of the networks. Right column: graphical representation of weight matrix of the networks. Z axis means strength of weight.

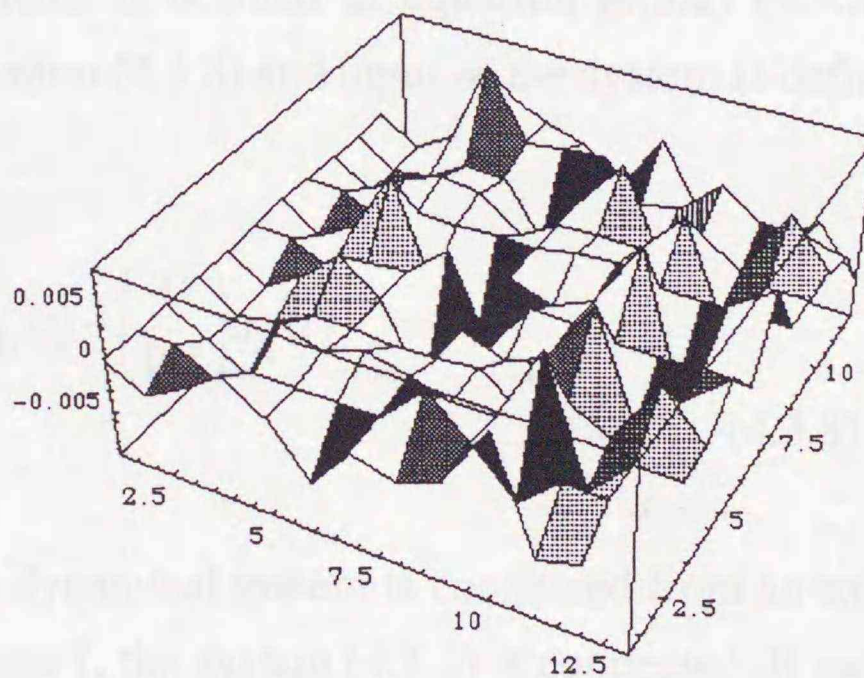
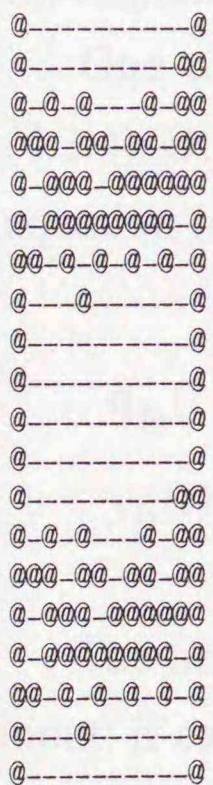


Figure 4.3.10 Response and network structure at the end of learning.

Left column: Response of the networks. Right column: graphical representation of weight matrix of the networks. Z axis means strength of weight.

4.3.2 Learning a Set of Limit Cycles in a Network.

In this section, the recurrent network is applied to an excitation problem of two vibration systems (Figure 4.3.11). The network is connected alternatively with one of two dynamical systems that have a different natural frequency. The object of this problem is to acquire the network that excites both dynamical systems.

This problem is more difficult than simple excitation problems described in previous section, because in this problem network learning is suffering from sever local minimum area. For example, Assume that a network acquires the limit cycles for a certain vibration system. Then if the network is trained for other vibration systems, it forgets something about first vibration system in the most of cases.

(a) Experimental system

One of two vibration systems is defined in equation (4.3.2) (4.3.3) and (4.3.4). Other is defined in equation (4.3.8) and input of the system is defined as equation (4.3.3) and (4.3.4).

$$\begin{aligned} \mathbf{q}_b(t+1) &= \begin{pmatrix} 1 & 1 \\ -0.3 & 0.7 \end{pmatrix} \mathbf{q}_b(t) + \begin{pmatrix} 0 \\ 0.5 \end{pmatrix} u_a(t) \\ y_b(t) &= (1 \ 0) \mathbf{q}_b(t) \end{aligned} \quad (4.3.8)$$

The network detects which dynamical system is connected from an external input: if external input has values 1, the system (4.3.2) is connected. If external input has value 0, the system (4.3.8) is connected. The evaluation function is equation (4.3.11).

$$f_c = \max(y_a) - \min(y_a) \quad (4.3.9)$$

$$f_b = \max(y_b) - \min(y_b) \quad (4.3.10)$$

$$f_d = f_c \cdot f_b \quad (4.3.11)$$

Learning conditions were as follows:

$$NB=3, \quad N=12 \quad (4.3.12)$$

Length of a copy block is decided by equation 4.2.5. Recurrent network is trained with four sets of operator: (M+C), (M), (M+IC) and (M+II). The population size of GA was 24.

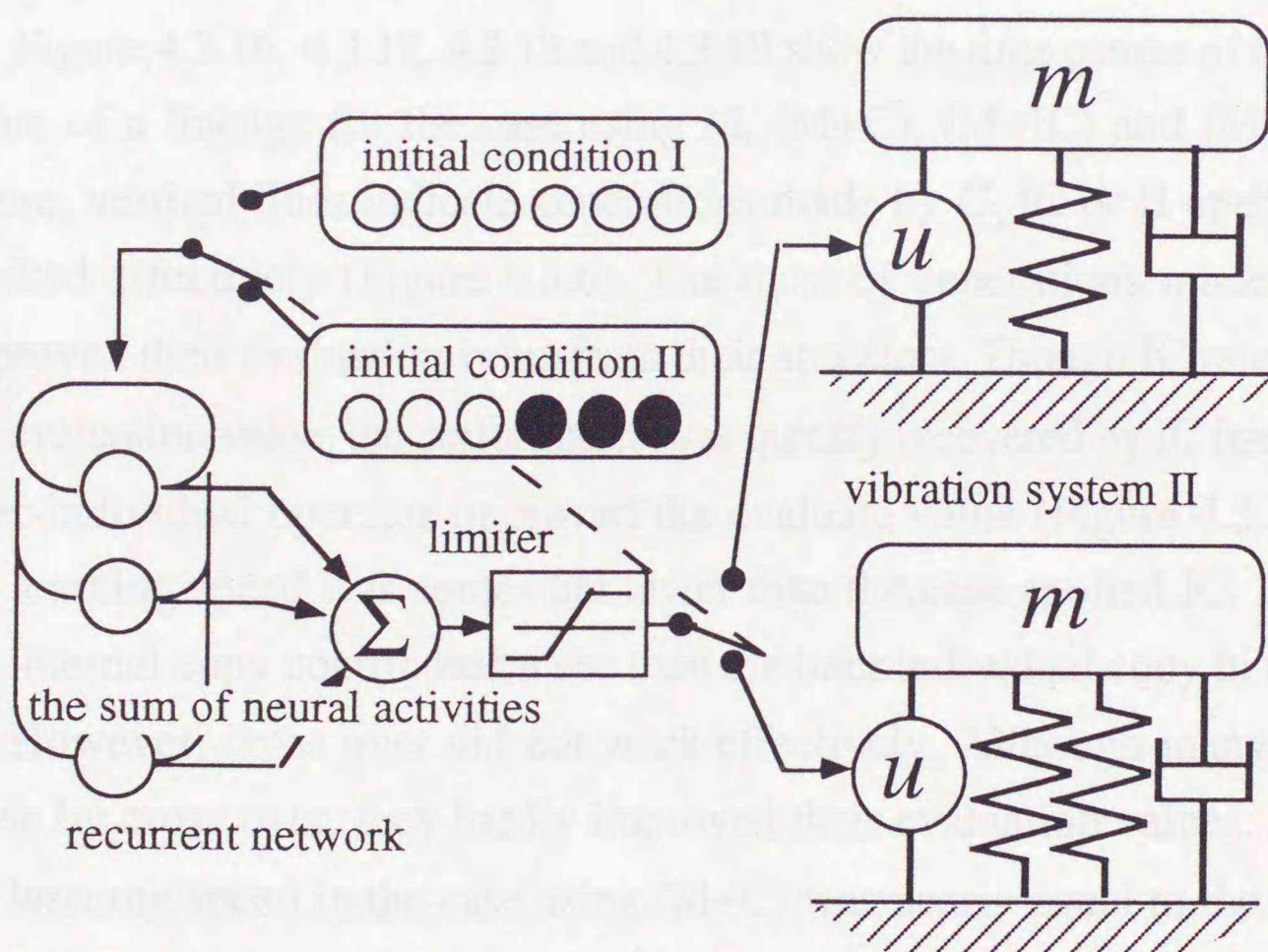


Figure 4.3.11 An excitation problem of two dynamical systems

(b) Results of experiments

The network acquired two limit cycles through training (Figure 4.3.12, 4.3.13). Each vibration system reacted actively. The acquired limit cycles were similar to the optimal solution.

We made experiments using M, (M+C), (M+IC) and (M+II), respectively, to compare each other. Figure 4.3.14 shows time course of evaluation values. Figure 4.3.15 shows estimation values at 500th generation for each case. The cases using (M+IC) and (M+II) are clearly superior the case using M and (M+C). In the cases using (M+IC) and (M+II), the recurrent network could acquire two limit cycles for two vibration systems. However, the network could not acquire two limit cycles in the other cases.

Figure 4.3.16, 4.3.17, 4.3.18 and 4.3.19 show the time course of the evaluation value of a lineage for the case using M, (M+C), (M+IC) and (M+II). In each figure, vertical lines indicate generations made by C, IC or II operator. The IC worked effectively (Figure 4.3.6). The most of generations made by IC were improved their evaluation value from their ancestors. Though IC rarely decreased the evaluation value, the performance was quickly recovered by IC itself. Likewise, inter-individual operator improved the evaluate value (Figure 4.3.7), although the learning speed was somewhat lower than the case applied IC. It means that the internal copy contributed more than the inter individual copy in this task.

However, cross over did not work effectively. Although many generations made by cross over, they hardly improved their evaluation values. For instance the learning speed in the case using (M+C) was nearly equal to the speed of the case using only M.

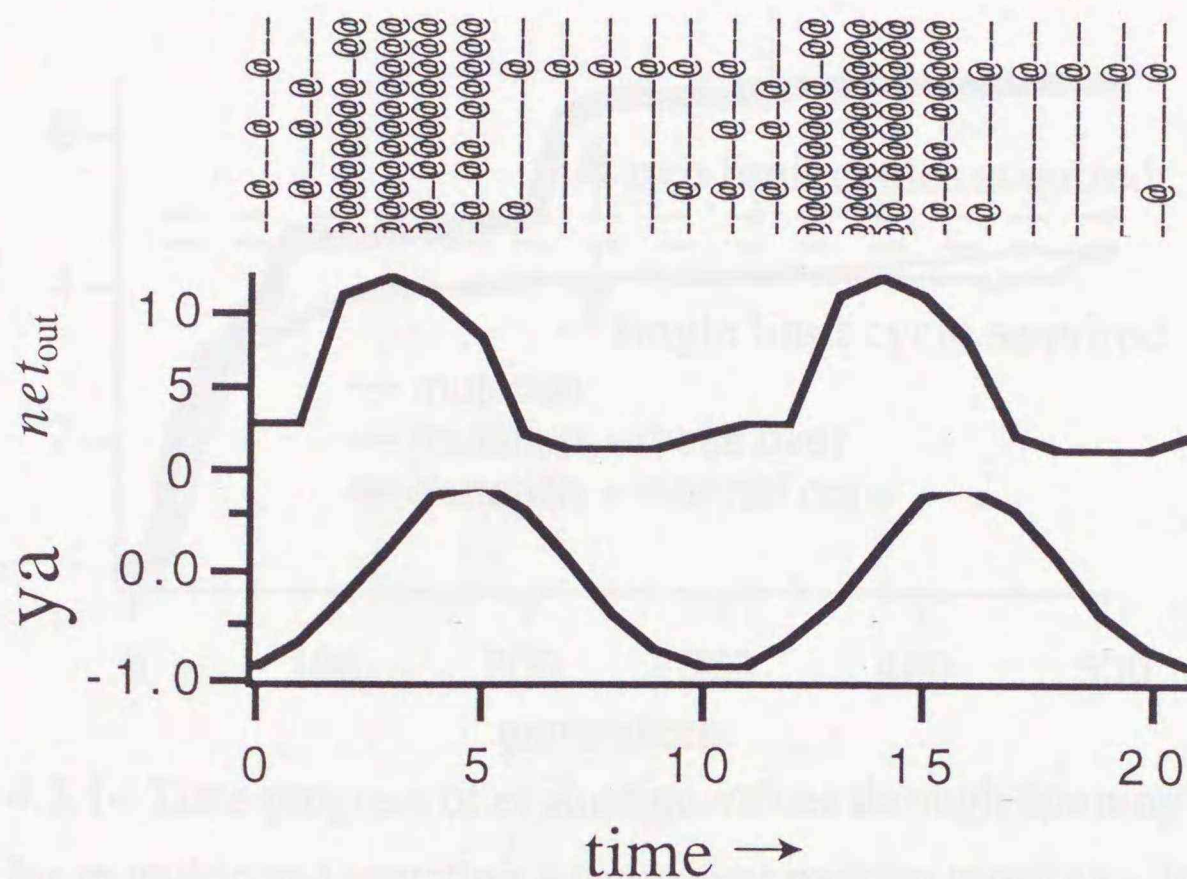


Figure 4.3.12 The limit cycle acquired in a network (for system 4.3.1).
 @ and | mean that each neuron is active and not active respectively at each time.

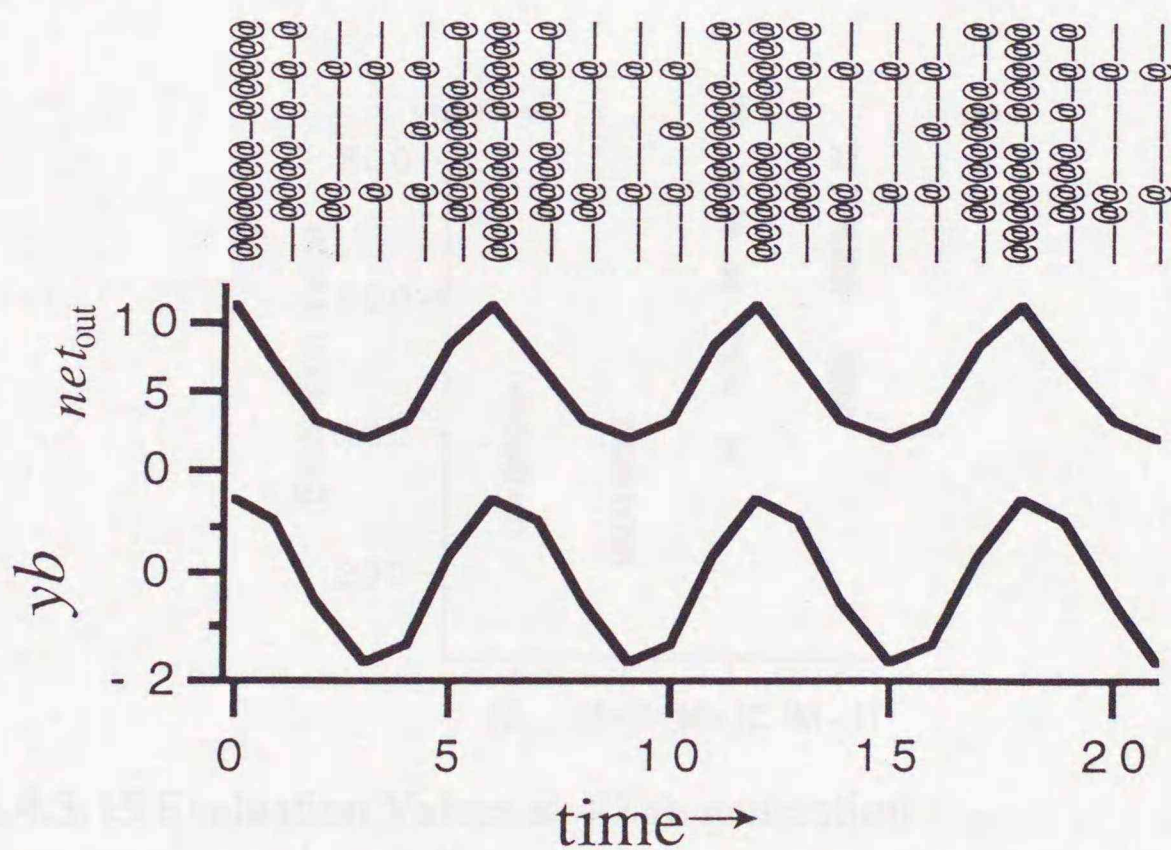


Figure 4.3.13 The limit cycle acquired in a network (for system 4.3.8).
 @ and | mean that each neuron is active and not active respectively at each time.

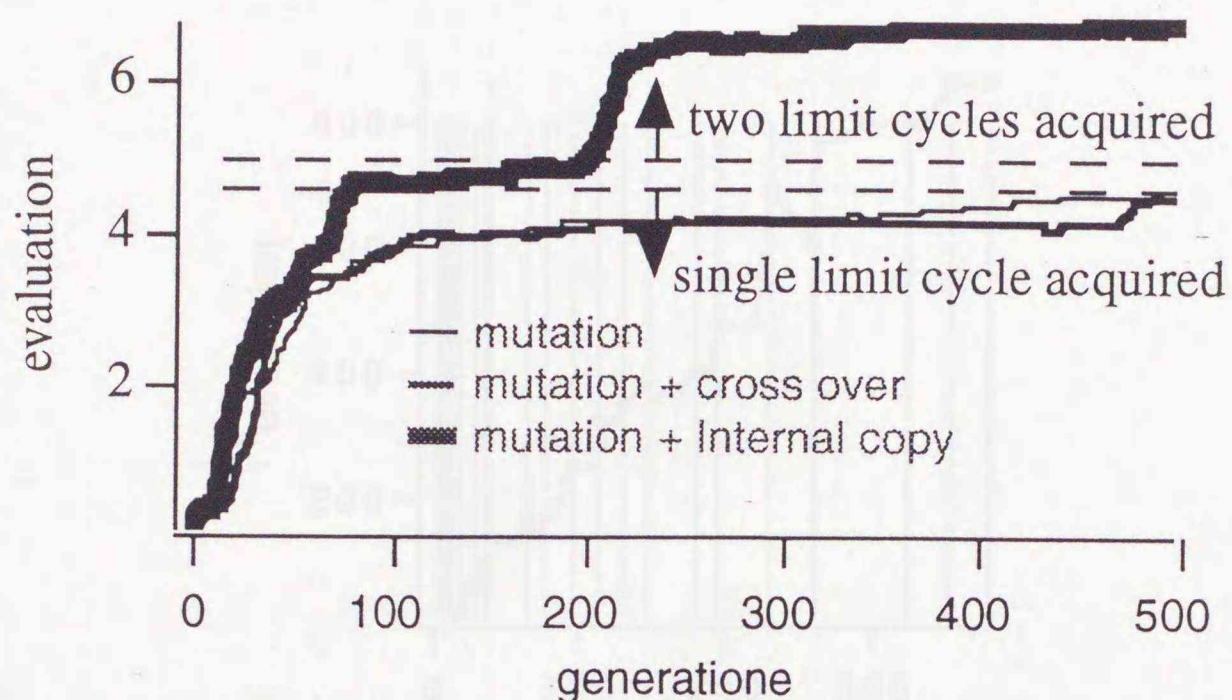


Figure 4.3.14 Time progress of evaluation values through learning.
(Lines for mutation and mutation + cross over overlap mostly each other.)

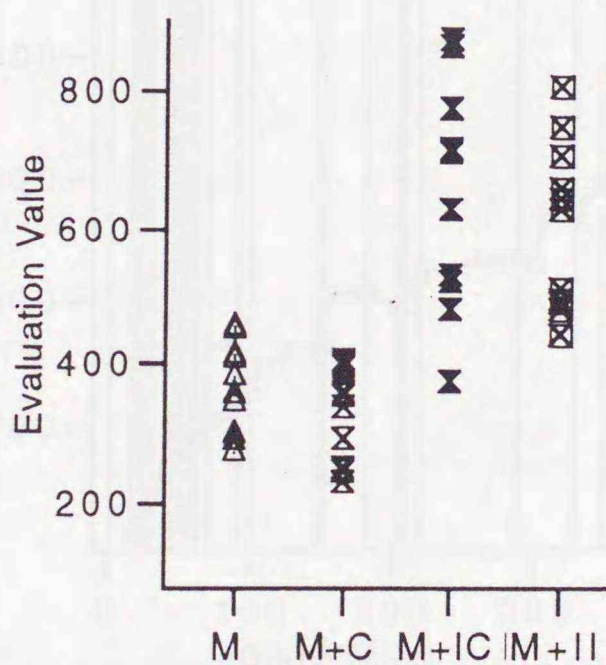


Figure 4.3.15 Evaluation Values at 400th generation
M: mutation. M+C: mutation and cross over .
M+IC: mutation and internal copy. M+II: mutation and inter-individual copy.

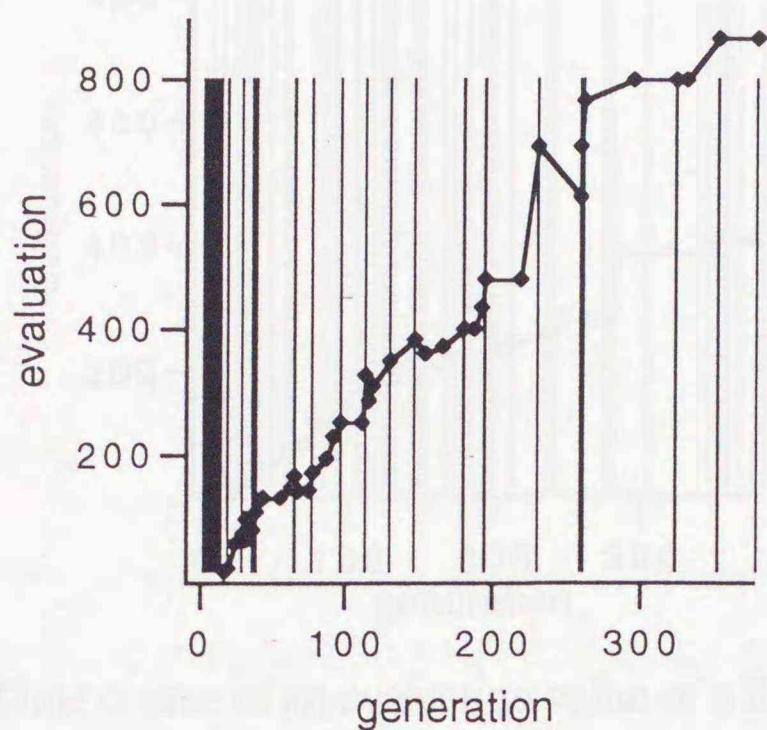


Figure 4.3.16 Time course of an evaluation value of a lineage when mutation and internal copy are applied. Vertical lines indicate generations generated by internal copy.

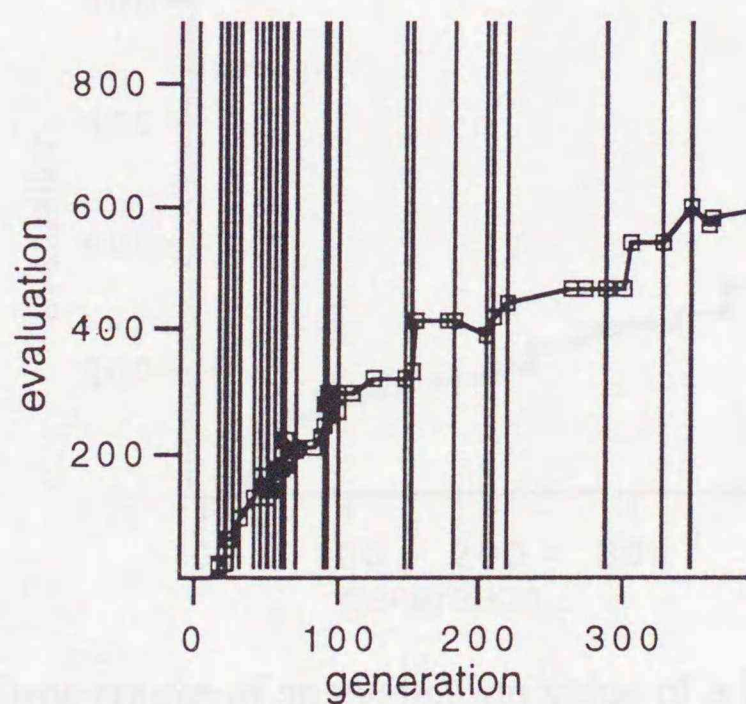


Figure 4.3.17 Time course of an evaluation value of a lineage when mutation and inter individual copy are applied. Vertical lines indicate generations generated by inter-individual copy.

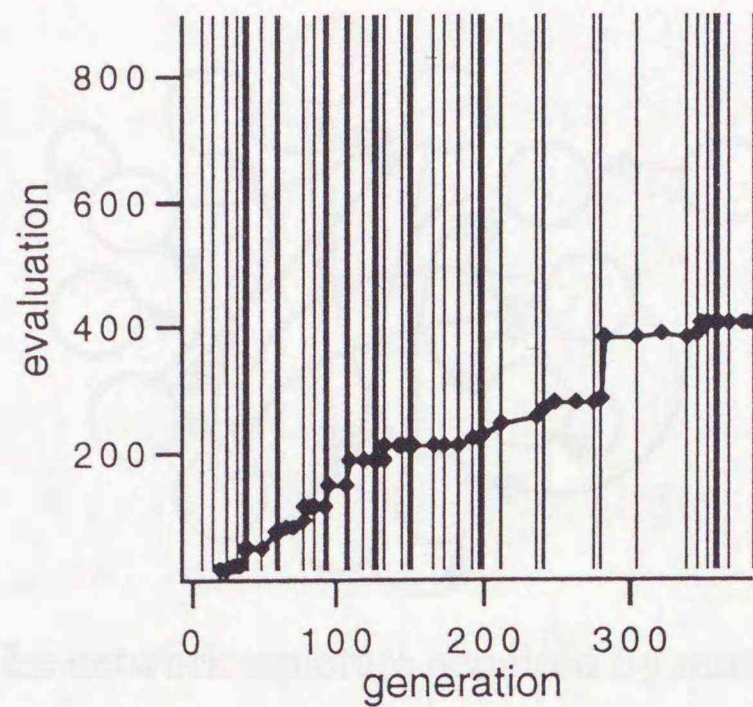


Figure 4.3.18 Time course of an evaluation value of a lineage when mutation and cross over are applied. Vertical lines indicate generations generated by cross over.

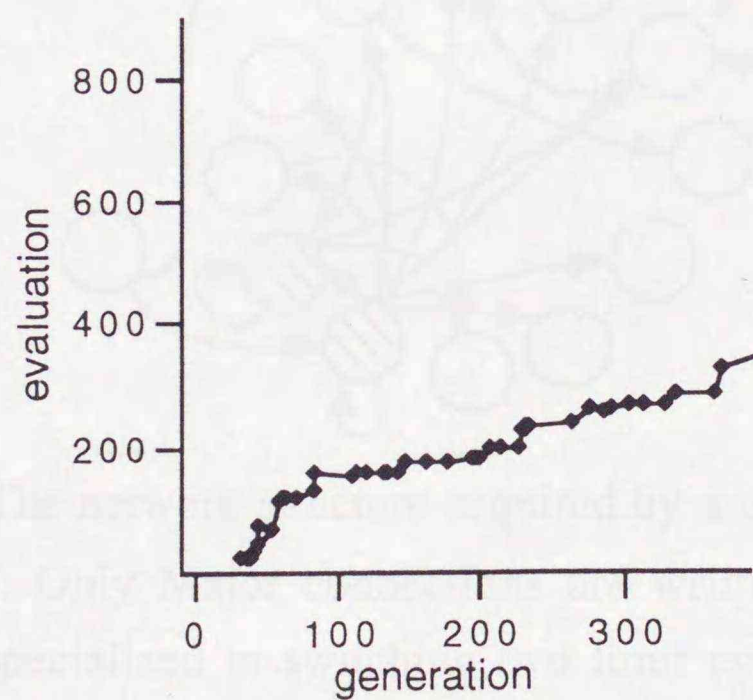


Figure 4.3.19 Time course of an evaluation value of a lineage when mutation is applied.

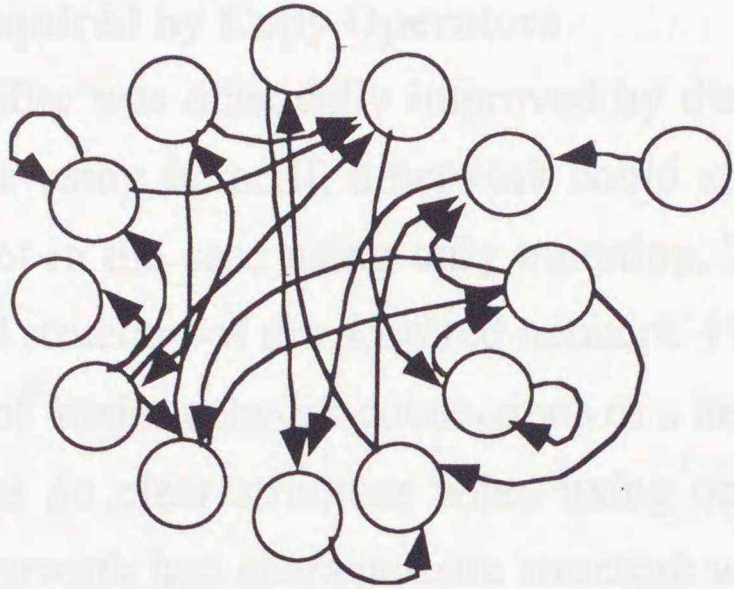


Figure 4.3.20 The network structure acquired by mutation operator. Only major connections are written. There is no apparent structure in network.

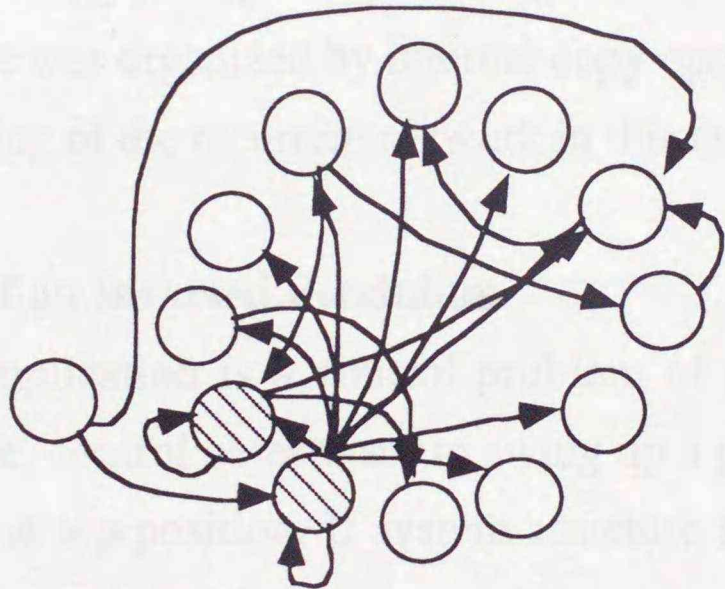


Figure 4.3.21 The network structure acquired by mutation and internal-copy operators. Only Major connections are written. A few neurons (hatched) are specialized in switching two limit cycles and keeping a network synchronization.

4.3.3 Structure Acquired by Copy Operators

The learning ability was drastically improved by the IC or II operator. For example, in the case using IC or II, a network could acquire two limit cycles although it could not in the case using only mutation. To know the reason of this, we investigated structure of the acquired network. Figure 4.3.20 and 4.3.21 shows the example of major network connections of a network after learning.

The network has no clear structure when using only M (Figure 4.3.20). Contrastively, the network had characteristic structure when it learned through (IC+M) (Figure 4.3.21). A few neurons had strong connections to the others. It means that neurons could work as the pace maker of the network. They also have major connections from the external input. That is useful to change the oscillation mode in corresponding to external signals.

In short, structure was organized by internal copy operator and that structure accelerates the learning of the recurrent network in this task.

4.3.4 Swinging up of an Inverted Pendulum

The following application is a control problem of an inverted pendulum (Figure 4.3.12). Here, control objects are to swing up a pendulum from bottom position and keep it at top position. If system structure and system parameters are available, this control problem is easy. However, it becomes a difficult problem when assuming that system structure is unknown and non-teacher is available. Here, only the angle of a pendulum is controlled. Namely cart position is ignored. And yet, it is difficult task because it includes two behaviors which have different characteristics: swinging up by excitation and stabilization at unstable point.

(a) Experimental system

The following are the equation of motion of an inverted pendulum system

and parameters which are used here.

$$M\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta + C\dot{x} = u \quad (4.3.13)$$

$$mL\ddot{x}\cos\theta + J\ddot{\theta} + p\dot{\theta} - mgL\sin\theta = 0 \quad (4.3.14)$$

$$m : 0.015[\text{kg}]$$

$$J : 2.58 \times 10^{-4}[\text{kg} \cdot \text{m}^2]$$

$$M : 0.625[\text{kg}]$$

$$L : 0.12[\text{m}]$$

$$p : 0.62 \times 10^{-4}[\text{kg} \cdot \text{m}^2/\text{s}]$$

$$C : 13.6[\text{Ns/m}]$$

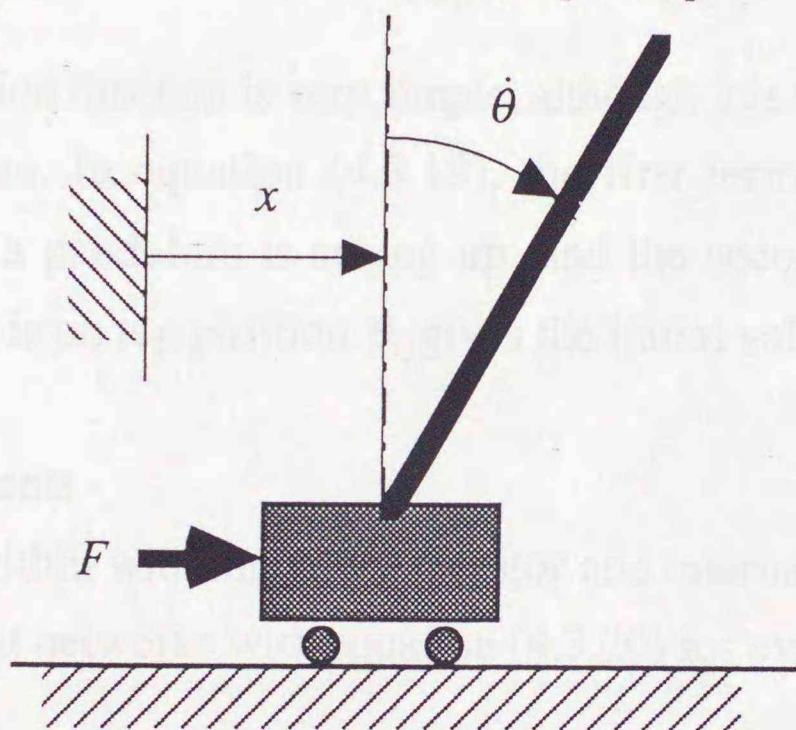


Figure 4.3.22 An inverted pendulum system

The manipulated value u is calculated as follows.

$$net_{out} = \sum_{j=1}^N y_j(t) \quad (4.3.15)$$

$$u(t) = \begin{cases} N \cdot 5/6 & (\text{if } net_{out} > N \cdot 5/6) \\ (net_{out}) & (\text{if } N \cdot 5/6 > net_{out} > N/6) \\ N/6 & (\text{if } N/6 > net_{out}) \end{cases} \quad (4.3.16)$$

Learning conditions were as follows:

$$NB=2, \quad N=12 \quad (4.3.17)$$

Instead of θ , $\sin(\theta)$ and $\cos(\theta)$ are given to a network as external input. The following are an evaluation function.

$$f_k = \begin{cases} (1 - |\theta|) & \text{if } |\theta| < 0.2 \\ 0 & \text{others} \end{cases} \quad (4.3.18)$$

$$f_{sw}(\theta_0) = (\pi - \min(|\theta|)) + \int f_k dt \quad \{\theta, t \mid \theta(0) = \theta_0, 0 \leq t < 10\} \quad (4.3.19)$$

The above evaluation function is very simple, although it is a little complicated by formula expression. In equation (4.3.19), the first term of the right part evaluates how much a pendulum is swung up, and the second term evaluates how long a pendulum is on top position. θ_0 gives the initial value of θ .

(b) Results of experiments

First genetic algorithm with mutation operator and internal copy operator is applied to the recurrent networks with equation (4.3.20) for evaluation.

$$f = f_{sw}(-0.10) + f_{sw}(0.10) \quad (4.3.20)$$

Through this evaluation function, the network stabilized the pendulum at top position after enough learning (Figure 4.3.23). Moreover, using the following evaluation function, genetic algorithm is applied to the network which already learn how to stabilize the pendulum at the top.

$$f = f_{sw}(\pi) + f_{sw}(-0.10) + f_{sw}(0.10) \quad (4.3.21)$$

Then the network acquired the way of swinging up and stabilization at the same time (Figure 4.3.24).

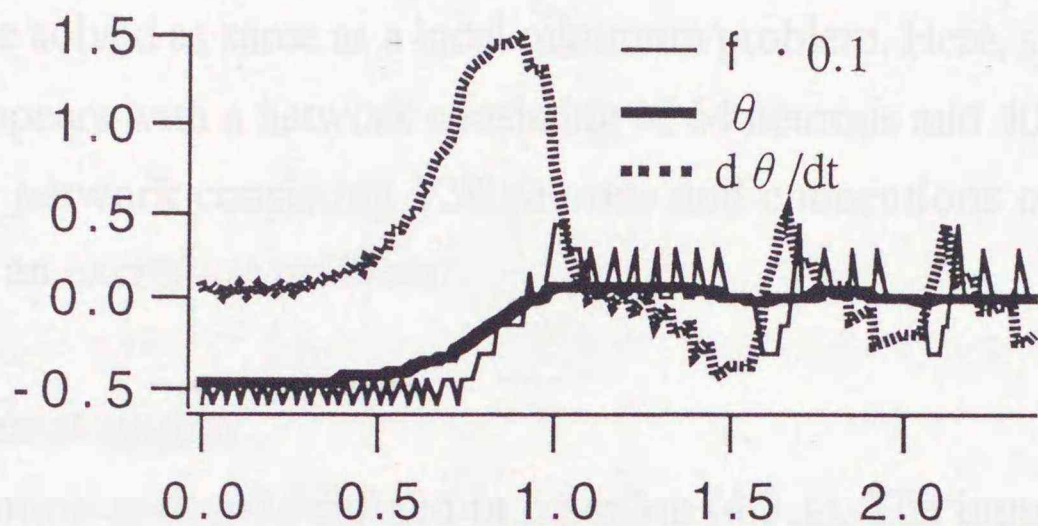


Figure 4.3.23 Response of a pendulum

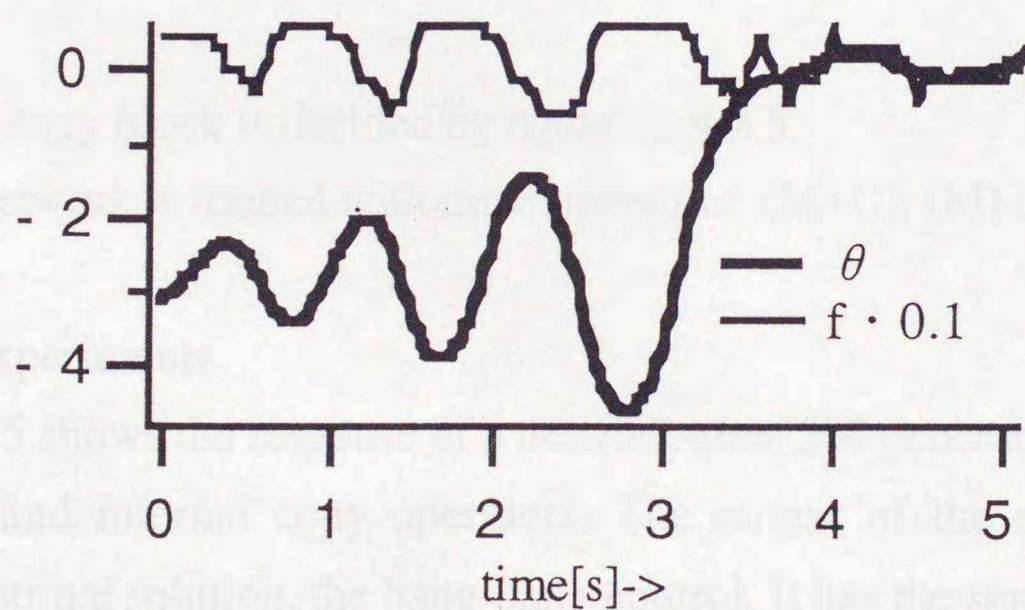


Figure 4.3.24 Process of swinging up

4.3.5 Application to Large Scale Problems

In this section, proposed genetic algorithm is applied to some large scale problems in comparison with previous sections. Scale problem is important problem to be solved as same as a local minimum problem. Here, simple excitation problem reappears with a network consisting of 64 neurons and 4096 connections. Moreover, a network consisting 128 neurons and connections over than 16000 is applied to an excitation problem.

(a) Experimental system

The vibration system is defined in equation (4.3.1). The input of the system (4.3.1) is defined in (4.3.2). The evaluation function was equation (4.3.4). The system is initialized with equation (4.3.5) before evaluation.

The population size of GA is 24. The parameters of the network were decided as follows:

$$NB=24, \quad N=64 \quad (4.3.22)$$

Length of a copy block is decided by equation 4.2.5.

Recurrent network is trained with three operators: (M+C), (M) and (M+IC).

(b) Results of experiments

Figure 4.3.25 shows the response of a network after 500 generations learning with mutation and internal copy operators. The output of the network was similar to the optimal solution, the bang-bang control. It has the same frequency as the natural frequency of the dynamical system.

We made experiments with M, (M+IC) and (M+II) operator, respectively. Figure 4.3.26 shows the time course of evaluation values. The learning was accelerated in the case using (M+IC) or (M+II) in comparison to the case that

used only M. We also made experiment using (M+C). However, there was no significant difference in the learning speed between the case using (M+C) and the case using only M.

We made experiments with networks which 128 neuron with the following conditions.

$$NB=48, \quad N=128 \quad (4.3.23)$$

Figure 4.3.27 shows the response of a network after 500 generation learning with mutation and internal copy operators. The output of a network approximates the optimal solution. In this experiment number of connections is 100 times than the experiment in the section 4.3.1. As so far it is shown that internal copy operator is available for a kind of large scale problem.

Figure 4.3.27 Response of internal network



Figure 4.3.28 Distribution of evaluation value

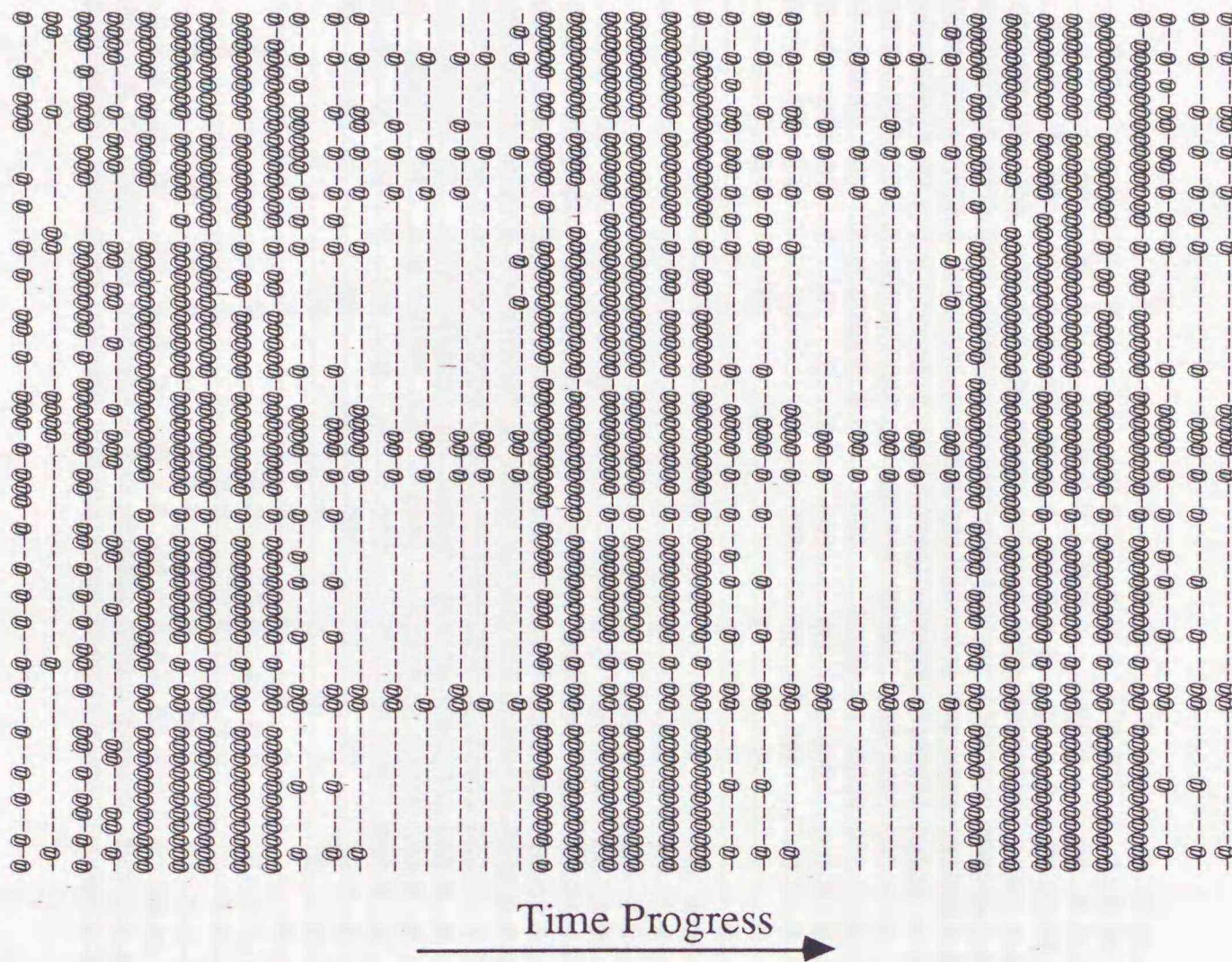


Figure 4.3.25 Response of a recurrent network

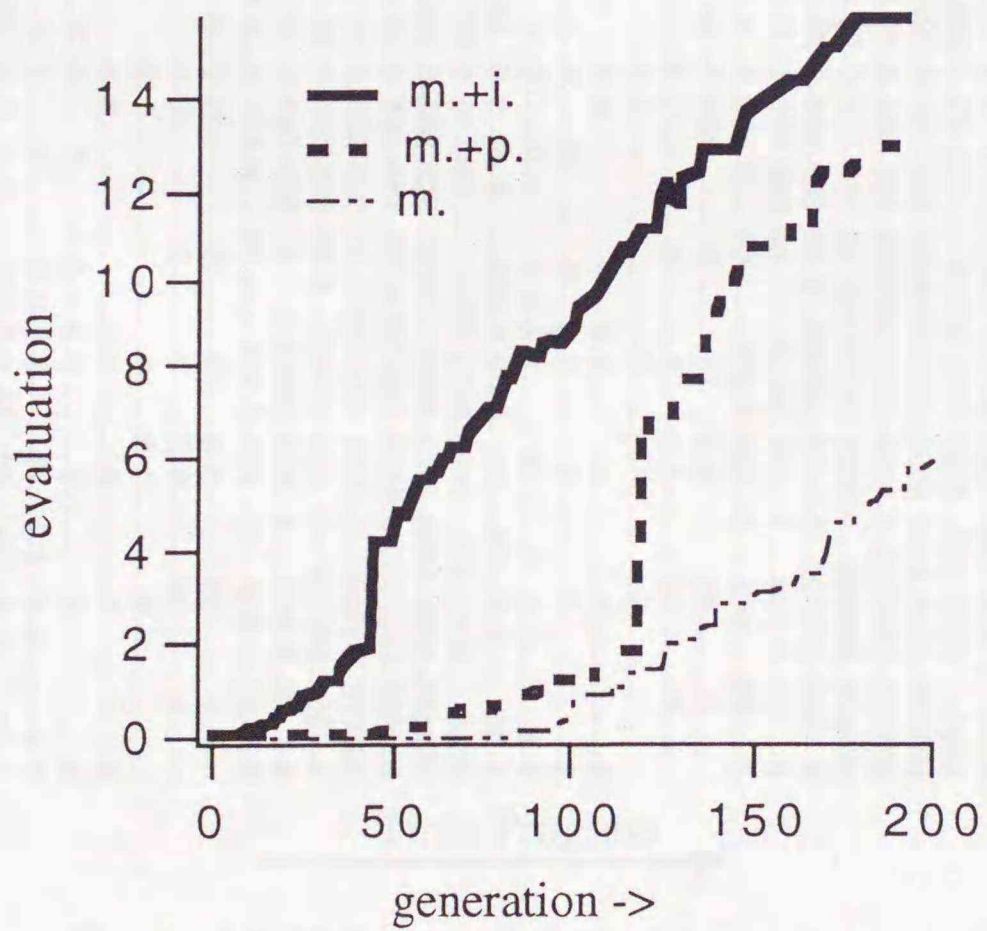


Figure 4.3.26 Time course of evaluation values

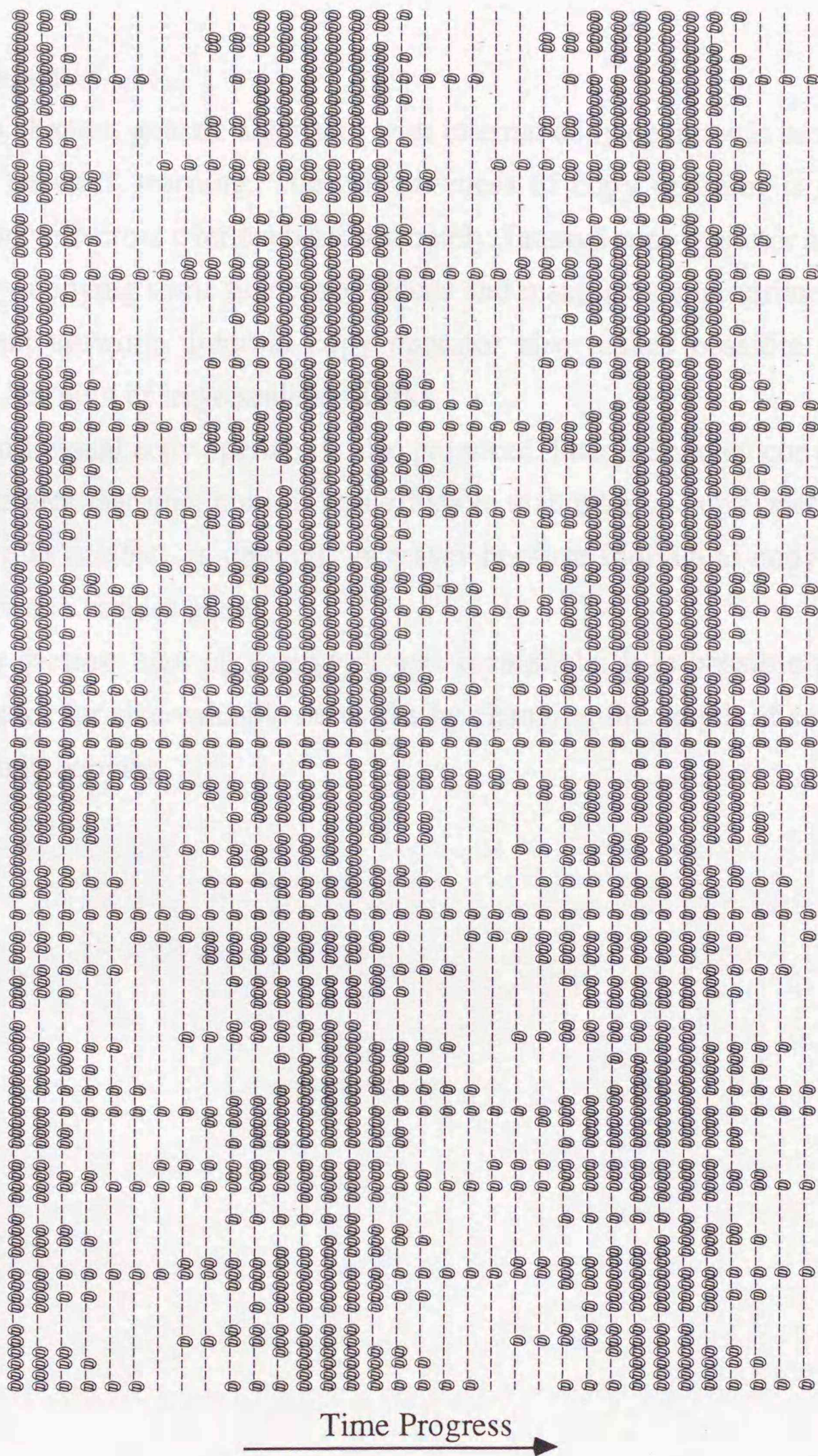


Figure 4.3.27 Response of a recurrent network

4.4 Conclusions

In this chapter, genetic algorithm with internal copy operator is proposed for recurrent network learning. The effectiveness of copy operator is shown in comparison with cross over operator concretely. Internal copy operator accelerates learning by copying some functional blocks and making an appropriate structure in recurrent network. Internal copy operator also makes possible a genetic algorithm learning of large scale networks.

Inter individual copy operator is also proposed. Inter individual copy operator also accelerates learning, but it is less effective than internal copy operator. This reason is that ineffective copying increases because individual copy operator copies between various genes.

In this section, size of a network was invaluable. It is possible to extend copy operator for size valuable networks by changing the length of source and destination in copying.

References of Chapter 4

- [1]A.P.Wieland, Evolving neural network controllers for unstable systems, IEEE IJCNN II pp.667-673(1990).
- [2]A.D.Beer and J.C.Gallagher, Evolving dynamical neural networks for adaptive behavior, Adaptive Behavior, vol.1 no.1, pp.91-122(1992).
- [3]R.K.Belew, J.McInerney, and N.N.Schraudolf, Evolving networks:Using the genetic algorithm with connectionist learning, Artificial Life II:Proceedings of the Workshop on Artificial Life, C.G.Langton, C.Taylor, J.D.Farmer, and S.Rasmussen, Eds. Addison-Wesley,pp.511-547(1992).
- [4]P.J.Angeline, G.M.Saunders, and J.B.Pollack, An Evolutionary Algorithm that Constructs Recurrent Neural Networks, IEEE Trans. on Neural Networks, Vol5., No.1, pp.54-65,1994
- [5]T.Kumagai, M.Wada, S.Mikami: Learning of Recurrent Neural Networks Through Genetic Algorithm with Internal Copy Operator, International Conference on Neural Networks, Special Sessions, pp.141-148(1996).
- [6]T.Kumagai, R.Hashimoto, M.Wada: Learning of Limit Cycles in Discrete-Time Neural Network, Neurocomputing Journal, Vol.13, No.1, pp.1-10(1996).
- [7]S.Gardella, T.Kumagai, R.Hashimoto and M.Wada, On the dynamics and applications of a discrete time binary neural network with time delay, Journal of Intelligent & Fuzzy Systems, Vol.2,No.6,pp.243-250,1994
- [8]H.Kitano, Designing neural networks using genetic algorithms with graph generation system, Complex Systems, vol.4,pp.461-476(1990).

Chapter 5

Summary

The aim of this thesis is to represent results of my works on design of recurrent neural networks. This thesis is summarized as follows.

The backgrounds of this study are described in chapter 1.

The outline of this thesis is also illustrated.

In chapter 2, dividing control is proposed to develop a design method of a neural network learning controller, because conventional learning control method can not be applied to a multi output plant when an exact model of a plant is unknown. In a multi output problem, generally, it is difficult to design an appropriate teaching signal since a manipulated variable influence many output values.

First dividing control is introduced through an application to a linear system.

If plant is written by a linear simultaneous equation, a plant is rewritten in the following form:

1. Only one system has an original manipulated variable as input.
2. The other systems have an output of another system as input.

Then conventional learning controller can be applied to each system. I named this process dividing of a plant and each divided system is called a sub-system. It is derived that if a plant is linear and linear feedback is applied to sub plants, then constructed controller is equivalent to a feedback controller applied to the original plant.

Dividing control is applied to a stabilization problem of an inverted pendulum. An inverted pendulum system is divided into two second order lag systems by the above procedure. A neural network learning controller with a reference model is applied to each system.

Dividing control is also applied to a control problem of an active mass damper. The control system is divided into two second order lag systems. In this case, two sub systems have very strong interference between two. Therefore one sub system is controlled by the nonlinear controller based on apriori knowledge. The other one is controlled by a neural network learning controller. After learning non-linear controller is acquired in a neural network. The acquired controller has robustness as same as a linear controller. Its performance is superior to quadratic optimal linear controller.

As so far, dividing control makes possible a non-linear learning control of multi output system. In other word, dividing control extends the application domain of learning controller to complex systems where plural systems interfere with each other. The function which is acquired by each network is simplified in comparison with the case where a plant is controlled by only one network. In theory neural network can map any function. However convergence of

backpropagation algorithm is not guaranteed. Therefore simplification of acquired function is very important.

In chapter 3, a time discrete binary output recurrent neural network is proposed, analyzed and demonstrated, because layered networks can not deal with temporal data.

First, it is mentioned that teaching algorithm has not been derived for almost networks consists of realistic neuron because of mathematical difficulty.

The authors notice Caianiello-Nagumo neuron model, a time discrete binary output simplest leaky integral neuron model, because it has brilliant computation ability in spite of its simplicity. For example, it can produce a pulse sequence with arbitrary firing rate by only one neuron. A fully connected recurrent network is constructed from Caianiello-Nagumo neuron model. The constructed neural network model is also extension of Hopfield network.

One of the most important characteristics of proposed networks is that it has broad tolerance to keep oscillation of a certain limit cycle or to convergence to a certain limit cycle, because a neuron consists threshold function, and output of a neuron is not change until internal state crosses a threshold value. This characteristic is very helpful for memorizing of limit cycles.

Based on this stability of networks, the condition for embedding a set of limit cycles is derived. From this result, the algorithm that decides the parameters of networks which memorize a set of limit cycles. Moreover the condition to converge to a certain limit cycles and the algorithm which decides state spaces in which the network converges to a certain limit cycle are derived.

Some set of short limit cycles is embedded in two neuron networks for demonstration. Then it is shown that network can memorize a set of context sensitive limit cycles. Moreover network which transfer between limit cycles according to external input is designed.

As so far the fully connected time discrete binary output network which consists of leaky integration neurons is proposed, analyzed and its analytical design method is derived. The theoretical designing algorithm is derived. It is also shown that the theoretical algorithm is not available for embedding of complex limit cycles. It is clear that networks require hidden neurons for complex limit cycles and no method is established to decide behavior of hidden neurons. Therefore, in chapter 4, learning of recurrent networks using genetic algorithms is described.

In chapter 4, the genetic algorithm with internal copy operator is proposed. There is no way to design the proposed recurrent networks, although it has excellent expression capability. Most difficult point of recurrent network learning is that search space is very large. Genetic algorithm has the advantage that it can search broad search space. Therefore genetic algorithm is applied to recurrent network learning.

First, it is described that the inefficiency of conventional genetic algorithm on neural network learning. That reason is that cross over operator does not work. Since neural networks can take various structure for the same function. Therefore the authors propose a genetic algorithm operator, internal copy operator, which copies a portion of gene to other part of the same gene to accelerate network learning.

Genetic algorithm with internal copy is applied to some learning problems. In the first experiment, the goal of the task is to excite a vibration system. Vibration system is driven by a total active neuron number of a recurrent neural network. Recurrent networks are trained with mutation operator, mutation and cross over operator, mutation and internal copy operator, respectively. After learning, recurrent networks acquire a quasi optimal solution, bang-bang control. Internal copy operator accelerates network learning. This experiment shows that

how internal copy operator works effectively. The authors investigate the relationship among generation, which operator produces children, weight matrix of networks and response of a network. The authors conclude the role of internal copy as follows. In the beginning of leaning, mutation operator works hard and finds out a fragment of oscillation structure. In the middle period, internal copy operator broadly copies it to whole network. The duplication of a fragment makes repetitive structure on weight matrix. The network oscillates entirely. End of the learning, the network acquires quasi optimal solution by mutation.

In next experiment, the goal is to excite a set of vibration system which is connected alternatively to recurrent networks. In this experiment, a recurrent neural network is required to acquire two limit cycles for two vibration systems. In this problem, network learning is suffering from local minimum problem. Assume that a network acquires the limit cycles for a certain vibration system. If the network is trained for other vibration systems, it must forget something about first vibration system, in the most of the case. In this experiment the authors introduce inter individual copy operator and compare with internal copy operator. Inter individual copy operator copy a portion of gene as same as internal copy operator, but it copies from another gene. The network is trained with mutation operator, mutation and cross over, mutation and internal copy, mutation and inter individual operator. In this experiment internal copy operator is superior other operator again. Especially the case using only mutation, mutation and cross over operator could not acquire two limit cycles in the almost case. This is because that internal copy operator makes network structure where some neurons have function as pace maker and many other neurons have strong connections between this neuron. Inter individual copy is superior cross over operator but inferior than internal copy operator.

In the next experiment, the goal of the task is to swing up the inverted pendulum from bottom position and keep it on upright position. Cart position is

ignored. In this experiment, a recurrent network must acquire two behaviors which have different characteristics, and must output consecutively. After enough learning, a network achieves the goal.

A large scale network is trained. First network with 64 neuron, more than 4000 parameters is trained for an excitation problem of a vibration system. Networks acquired quasi optimal solution easily. Here again internal copy operator is superior to cross over operator in learning speed. Inter individual operator is superior to cross over operator, but inferior than internal copy operator. Moreover networks with 128 neuron, more than 16000 parameters is trained for a stabilization problem of an inverted pendulum. Here, cart position is ignored.

As so far, internal copy operator accelerates genetic algorithm learning of recurrent networks in the following way. First it copies function block acquired by mutation operator to other part. Second it constructs structure and that makes possible acquiring of a set of different temporal sequences. Moreover, it is also effective for large scale problems.

However it is sure that internal copy operator is not universal, because what is required is depend on a task and what is acquired is depend on the coding of gene and what is copied by copy operator. Therefore it is important to modify coding and copy operator according to tasks. Most universal way is to code a way of copy in a gene.

Consequently, in this thesis, the dynamical control by dividing control, the dynamic recurrent neural network model and its designing method are described. Dividing control makes possible to control complex systems by layered networks. The proposed network model is theoretically analyzed and it is shown that the network has excellent capability. Proposed genetic algorithm is superior to simple genetic algorithm on learning speed, and it makes a training of large scale networks easy. This network and algorithm will be effective to realize dynamical

robot control in the real world.

Acknowledgments

This study would never have been accomplished without the help of many people. The author would like to particularly appreciate Professor Mitsuo Wada of Hokkaido University for his constant patient instruction and accurate advice on this work.

Dear thanks are also due to Professor Yukioji Sakata, Professor Akio Onuchi and Associate Professor Sadayoshi Mikami of Hokkaido University.

He especially thanks to Professor Kazuo Yamafuji of University of Electro-Communications for his constant encouragement.

He also thanks to Associate Professor Satoshi Aoki of University of Electro-Communications for his beneficial advice.

He wishes to express his gratitude to Professor Hiroshi Miyamoto, Associate Professor Hiroshi Yokoi, Associate Professor Kazu Suzuki and the other members of the research group of computer systems engineering of Hokkaido University for their beneficial discussions and suggestions.

He really thanks to Director General Michio Oishi of National Institute of Electroscience and Human Technology for his support of setting up the author's research environment.

He also really thanks to Head of Human Information Department, Shinya Saito of National Institute of Electroscience and Human Technology.

The author's thanks go to many staffs of National Institute of Electroscience and Human Technology, especially to Senior Researcher Myoichi Matsumoto, Senior Researcher Hiroshi Endo, Senior Researcher Atsuhiko Takahashi, Senior Researcher Akio Usami and Head of Behavior Control Laboratory, Terukazu Takada for their great assistance and encouragement.

Acknowledgments

This study would never have been accomplished without the help of many people. The author would like to particularly appreciate Professor Mitsuo Wada of Hokkaido University for his constant patient instruction and adequate advice on this work.

Deep thanks are also due to Professor Yukinori Kakazu, Professor Azuma Ohuchi and Associate Professor Sadayoshi Mikami of Hokkaido University.

He especially thanks to Professor Kazuo Yamafuji of University of Electro-Communications for his constant encouragement.

He also thanks to Associate Professor Satoshi Akita of University of Electro-Communications for his beneficial advice.

He wishes to express his gratitude to Professor Eiichi Miyamoto, Associate Professor Hiroshi Yokoi, Associate Professor Keiji Suzuki and the other members of the research group of complex systems engineering of Hokkaido University for their beneficial discussions and suggestions.

He really thanks to Director-General Michio Oishi of National Institute of Bioscience and Human-Technology for his support of setting up the author's research environment.

He also really thanks to Head of Human Informatics Department Shinya Saida of National Institute of Bioscience and Human-technology.

The author's thanks go to many staffs of National Institute of Bioscience and Human-technology, especially to Senior Researcher Ryoichi Hashimoto, Senior Researcher Hiroshi Endo, Senior Researcher Akihiko Takahashi, Senior Researcher Akio Utsugi and Head of Behavior Control Laboratory Tsunehiro Takeda for their great helpfulness and encouragement.

List of Results

Paper

- [1] T.Kumagai, S.Akita, M.Wada: Learning Control of an Inverted Pendulum Using Neural Networks, Trans. of the Japan Society of Mechanical Engineers, 58-550, C, pp.1781-7(1992).(in Japanese)
- [2] T.Kumagai, R.Hashimoto, M.Wada: Control of an Active Mass Damper Using a Neural Network, Trans. of the Japan Society Mechanical Engineers, 59-564, C, pp.2305-2311(1993).(in Japanese)
- [3] S.Gardella, R.Hashimoto, T.Kumagai, M.Wada: Periodic Responses of a Hysteresis Neuron Model, IEICE Trans. on information and Systems, Vol.E76-D, No.3, pp.368-376(1993).
- [4] S.Gardella, T.Kumagai, R.Hashimoto, M.Wada: On the dynamics and Applications of a DiscreteTime Binary Neural Network with Time Delay, Journal of Intelligent and Fuzzy Systems, Vol.2, pp.221-228(1994).
- [5] T.Kumagai, R.Hashimoto, M.Wada: Learning of Limit Cycles in Discrete-Time Neural Network, Neurocomputing Journal, Vol.13, No.1, pp.1-10(1996).
- [6] T.Kumagai, M.Wada, S.Mikami, R.Hashimoto: Learning Control Method for Robotic Dynamical System, Journal of Robotics and Mechatronics, Vol.9, No.1(1997.2, to appear).
- [7] T.Kumagai, M.Wada, S.Mikami, Learning of Recurrent Neural Network Using Genetic Algorithm with Internal Copy Operator, Transactions of the Institute of Electronics, Information and Communication Engineers D-II.(submitted)

International Conference

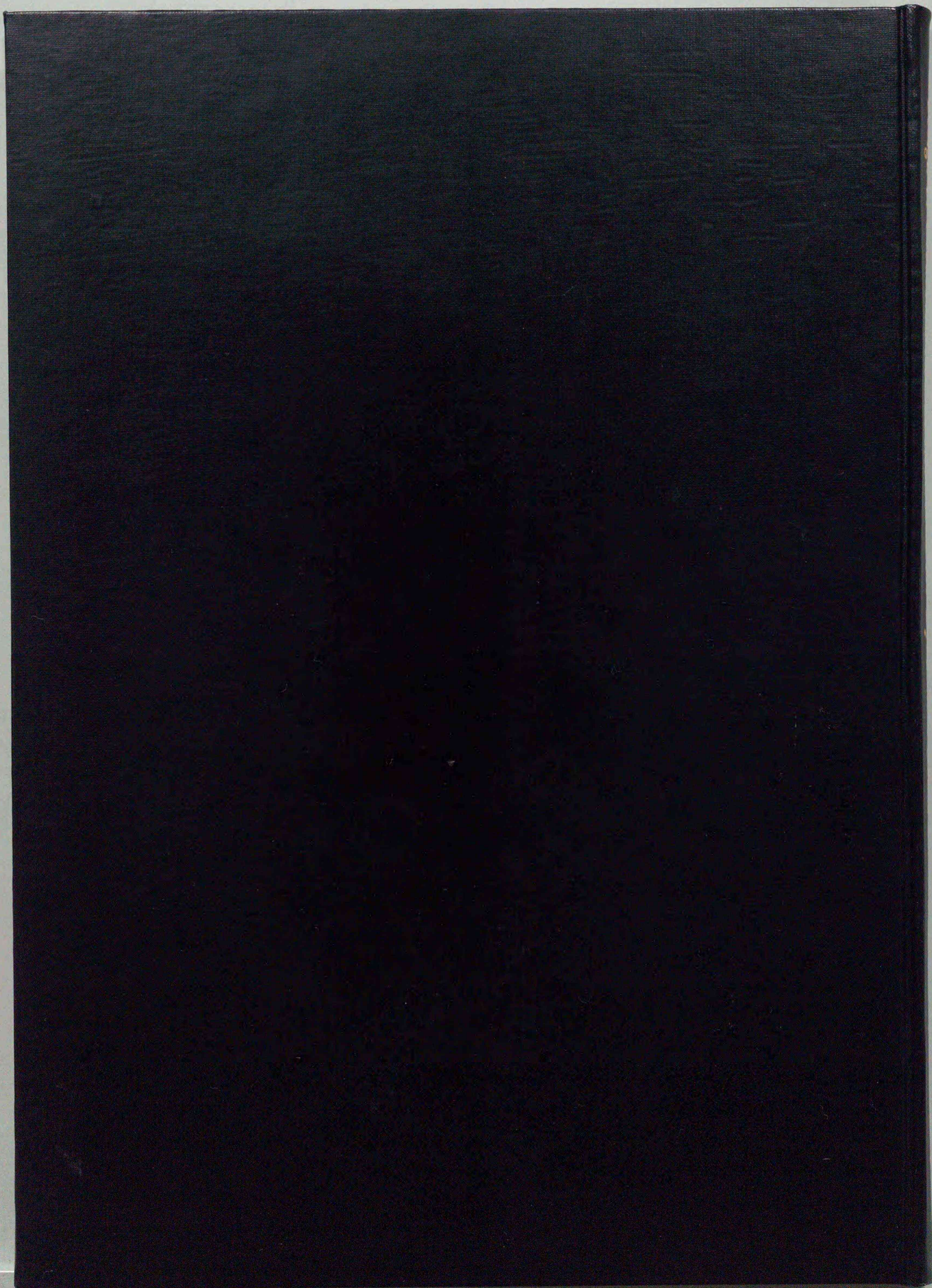
- [1] S.Gardella, T.Kumagai, R.Hashimoto and M.Wada: On the Dynamics and Potentialities of a Discrete-Time Binary Neural Network with Time Delay, Proc.2nd International Conference on Fuzzy Logic & Neural Networks, July,

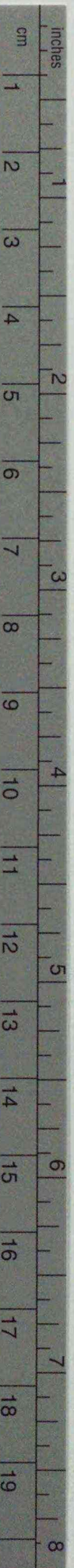
pp.493-499(1992).

[2]T.Kumagai, R.Hashimoto, M.Wada, M.Tanaka and Y.Yoshida: Control of an Active Mass Damper Using Layered Neural Networks, Proceedings of the 1993 JSME International Conference on Advanced Mechatronics, pp.943-8(1993).

[3]T.Kumagai, R.Hashimoto and M.Wada: Control of a Multi Degree of Freedom System Using Neural Networks, Proceedings of the Asian Control Conference Tokyo, pp.293-296(1994).

[4]T.Kumagai, M.Wada, S.Mikami: Learning of Recurrent Neural Networks Through Genetic Algorithm with Internal Copy Operator, International Conference on Neural Networks, Special Sessions, pp.141-148(1996).

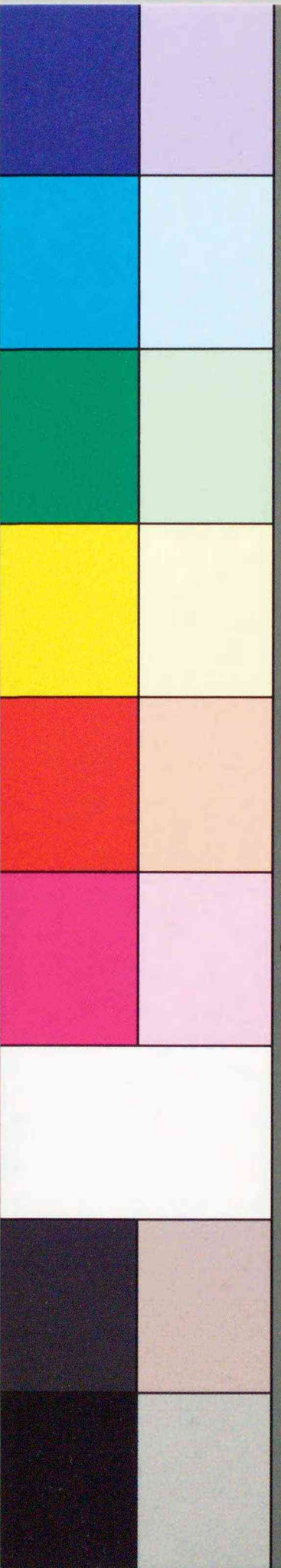




Kodak Color Control Patches

© Kodak, 2007 TM: Kodak

Blue Cyan Green Yellow Red Magenta White 3/Color Black



Kodak Gray Scale



© Kodak, 2007 TM: Kodak

A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19

