

1993

Adaptive neural network control of a flexible manipulator

Sun-Ti Wang
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Wang, Sun-Ti, "Adaptive neural network control of a flexible manipulator" (1993). *Master's Theses*. 596.
DOI: <https://doi.org/10.31979/etd.g8yr-52mq>
https://scholarworks.sjsu.edu/etd_theses/596

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.



University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 1353073

Adaptive neural network control of a flexible manipulator

Wang, Sun-Ti, M.S.

San Jose State University, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106



ADAPTIVE NEURAL NETWORK CONTROL OF A
FLEXIBLE MANIPULATOR

A THESIS

PRESENTED TO

THE DEPARTMENT OF GENERAL ENGINEERING

SAN JOSE STATE UNIVERSITY

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

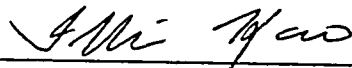
MASTER OF SCIENCE

By

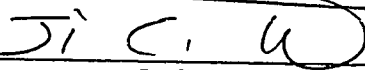
Sun-Ti Wang

May, 1993

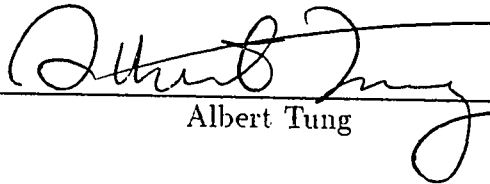
APPROVED FOR THE DEPARTMENT OF
GENERAL ENGINEERING



Imin Kao (Principal Adviser)

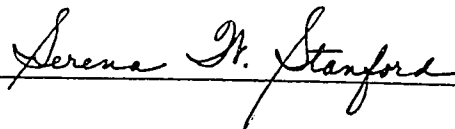


J. C. Wang



Albert Tung

APPROVED FOR THE UNIVERSITY



ABSTRACT

ADAPTIVE NEURAL NETWORK CONTROL OF A FLEXIBLE MANIPULATOR

by Sun-Ti Wang

A back-propagation neural network is applied to the end-position control of a one-link flexible manipulator with payload variations. A very difficult problem in controlling complex mechanical systems such as flexible beam robot is finding computationally acceptable methods to compensate for physical variations in the system. Adaptive control techniques have been proven to be useful only in stabilization of linear systems or some special non-linear systems; nevertheless, neural networks are capable of handling the computational complexity which sometimes can not be described in detailed mathematical terms. A deadbeat controller tuned by neural network for a single flexible arm robot is presented to demonstrate the capability of neural networks on the field of robotic control. In such an application, the neural network is employed to adjust parameters of the deadbeat controller as well as to identify changes of system dynamics when the flexible arm robot is manipulating various payload conditions.

Acknowledgements

I am grateful to my principal advisor, Dr. Imin Kao, for his helpful guidance throughout this study. I would also like to thank Dr. J. C. Wang and Dr. Albert Tung for their encouragement and academic advice.

Thanks are also due to my friends who encourage and support me. Very special thanks to my family for their unfailing support and love.

Contents

1	Introduction	1
1.1	General Introduction	1
1.2	Literature Review	2
1.3	Thesis Outline	4
2	Artificial Neural Networks	6
2.1	What Are Neural Networks?	6
2.2	Neural Network Construction	8
2.3	Back-Propagation Network	12
2.3.1	Learning Behavior	13
2.3.2	Simulation Examples	14
3	Neural Networks and Control	19
3.1	Control Systems and Neural Networks	19
3.2	Training of Neural Networks for Control	22
3.2.1	Direct Neural Controller	23
3.2.2	Neural Network Tuning Controller	25
4	Control of One-link Flexible Manipulator	27
4.1	Problem Definition	27
4.2	Mathematical Model	28
4.3	Deadbeat Control	31
5	Adaptive Neural Network Controller	36
5.1	System Performance vs. Payload Variation	36
5.2	Adaptive Neural Controller Design	38
5.3	Simulation Results	40
6	Conclusions and Future Work	47
	Bibliography	49
	Appendices	51

A Aspirin/MIGRANES	51
A.1 Aspirin Language and MIGRANES Interface	51
A.2 Example Files	53
A.2.1 Aspirin File	53
A.2.2 MIGRANES Command File	54
B Numerical Data	56
B.1 Experimental Data of Literatures	56
B.1.1 Data of Flexible Beam for First Vibration Mode	56
B.1.2 Data of Flexible Beam with Payload Variations	57
B.2 Simulation Data	57
C Symbols	59
C.1 Neural Network Symbols	59
C.2 Symbols of Flexible Beam Dynamics	60

List of Tables

B.1	Model parameters for first flexible mode	54
B.2	Frequency changes with respect to payload variations	55
B.3	Training data of the adaptive neural controller	56
B.4	Testing data of the adaptive neural controller	56
B.5	Simulation results of the zeros and poles of the deadbeat compensators	56

List of Figures

2.1	The schematic of bio-neuron and artificial neuron	7
2.2	An artificial neural network (ANN)	9
2.3	Constructure of single artificial neuron	10
2.4	Simulation of the sine function	15
2.5	A two-link planar robot arm	16
2.6	Kinematic training results with 300 millions iterations	17
2.7	Trajectory test of the two-link manipulator	17
2.8	Workspace of the two-link anipulator ($0^\circ \leq \theta_1 \leq 180^\circ, 0^\circ \leq \theta_2 \leq 180^\circ$)	18
3.1	Problem of obtaining training information	22
3.2	A direct neural control system	23
3.3	Copying training data from an existing controller	25
3.4	A neural network tuned PID control system	26
4.1	Schematic of a single flexible manipulator	29
4.2	Root locus of deadbeat control system	34
4.3	Response and control effort of deadbeat control system	35
5.1	Response of payloaded system with free-payload compensator	37
5.2	Deadbeat control for various payload	38
5.3	Configuration of adaptive neural network	39
5.4	Adaptive neural deadbeat control system	40
5.5	System response of the adaptive neural network controller with 3% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).	42
5.6	System response of the adaptive neural network controller with 13% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).	43
5.7	System response of the adaptive neural network controller with 23% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).	44
5.8	System response of the adaptive neural network controller with 28% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).	45

5.9 System response of the adaptive neural network controller with 38% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line). 46

Chapter 1

Introduction

1.1 General Introduction

An artificial neural network is a system with inputs and outputs, and is internally composed of simple processing elements which function in an analogous way to a biological neuron. These processing elements are interconnected by lines with weights, namely synapses in biological terms. In general, neural networks have many different types of algorithm in selecting the weights to achieve the desired input/output mapping. Depending on the choice of the algorithm, the learning process determines the type of a neural network. In control systems, this feature can be applied in different situations, specially in the presence of the operation uncertainty of the plant where the conventional methods tend to fail frequently under extreme conditions. Generally speaking, a controlled process with a certain neural network architecture can lead to a good solution for such systems.

A back-propagation neural network is introduced to complement the end-position control of a single link flexible arm robot for various payload conditions. Although many researchers have presented the modeling and control of flexible manipulators recently, few of them have discussed the payload variations that affect the overall performance of the control systems. The main objective of this thesis is to present an adaptive neural controller based on a deadbeat control scheme to achieve the flexible beam control with payload variations.

In the present work software developments have been taken into account. We utilize MATRIX_X¹ on UNIX environment for control systems analyses, and both Aspirin/MIGRAINES² on UNIX and NWorks Explorer³ on PC are used for back-propagation network studies.

1.2 Literature Review

Neural networks have been hailed as the greatest technological advance since the transistor. A new form of machine intelligence has been elevated to transcendental heights, and thousands of technical papers and magazine articles have been written on the subject. Many types of researchers have an interest in this subject, and each brings a piece to the puzzle in an attempt to assemble a detailed working model of the mind. Understanding neural networks with the material available today is not easy without a degree in mathematics, biology or computer science. In order to

¹MATRIX_X is a registered trademark of Integrated Systems Inc. California, USA.

²This software is the copyright of Russell Leighton and the MITRE Corporation.

³This software is the copyright of NeuralWare, Inc.

show some appreciation for the vast amount of science brought together by neural networks, Stanley and Bak [1988] introduced materials on the basics of neurobiology and cognitive science to bridge the gap so that more people can understand and use neural networks.

Today neural networks are actively explored in artificial intelligence, psychology, engineering, and physics. There seems little doubt that neural network technology will have a lasting impact on the engineering science. For control applications, Millers *et al.* [1990b] brought together the different strands of theory that are most relevant to understanding and extending the state of the art in the use of neural networks. Barto *et al.* [1983] showed how a system consisting of two neuronlike adaptive elements can solve a difficult learning control problem. Also, Narendra and Parthasarathy [1990] demonstrated that neural networks can be used effectively for the identification and control of nonlinear dynamical systems.

In robot control, the overall complexity of many robotic control problems and the ideal of a truly general robotic system have led to much discussion of the use of neural networks. Millers *et al.* [1990a] presented a learning control technique based on an extension of the CMAC (Cerebellar Model Arithmetic Computer) network proposed by Albus that a training scheme is used to adjust the weights in the network in order to form an approximate dynamic model of the robot in appropriate regions of the control space and simultaneously the network is used during each control cycle to predict the actuator drives required to follow a desired trajectory. In addition, Ichikawa and Sawa [1992] introduced a learning algorithm and capabilities of neural

networks whose outputs and inputs are directly connected to plants just like ordinary feedback controllers. They proposed such a control scheme which is equivalent to state estimation and capable of controlling a highly nonlinear plant that can never be stabilized using a constant-gain linear controller. Another application of neural networks does not serve as a direct controller, but adjust the parameters of a conventional controller. Cañete *et al.* applied this type of neural network to a nonlinear self-tuning tracking problem [IWANN 1991].

In current robot control practice, the problem of controlling flexible manipulators has been widely studied, because there are substantial potential advantages to using robot manipulators that are very lightweight and flexible – advantages in power, quickness, and performance, as well as in weight. Initial studies have been concentrated on the single flexible link. Although most of the previously proposed control strategies have been proved successful [Cannon and Schmitz 1984, Wang *et al.* 1989, Wang and Vidyasagar 1991], they are insensitive to modeling uncertainty, such as payload variations, while achieving good stability and high performance.

The purpose of this thesis is to utilize neural networks based upon a tuning controller scheme on the single beam control problem while payload conditions are varied.

1.3 Thesis Outline

In what follows, we first describe the features of neural networks and then briefly introduce the learning algorithm of a back-propagation neural network. The reason

for applying the artificial neural network techniques to control systems is the next subject. Secondly, the adaptive neural control architecture is presented by following an outline of modeling and deadbeat control strategy of a single flexible arm robot. Finally, we show the simulation results and conclude the work we have done.

Chapter 2

Artificial Neural Networks

2.1 What Are Neural Networks?

The term neural networks comes from their design, which is based on the neural structure of the brain; both have highly interconnected neurons. At the present time, biologists, psychologists, computer scientists, engineers, and mathematicians are working all over the world to learn more about neural networks. Much is already known about the topic, but as is the case in most new research areas, there are a lot more questions than there are answers [Miller III *et al.* 1990b].

Neural networks are sometimes called by other names: artificial neural systems, connectionist systems, neurocomputers, adaptive systems, parallel distributed processors, collective decision circuits, and natural intelligence [Stanley 1988]. A neural network is a massively parallel, dynamic system of highly interconnected interacting

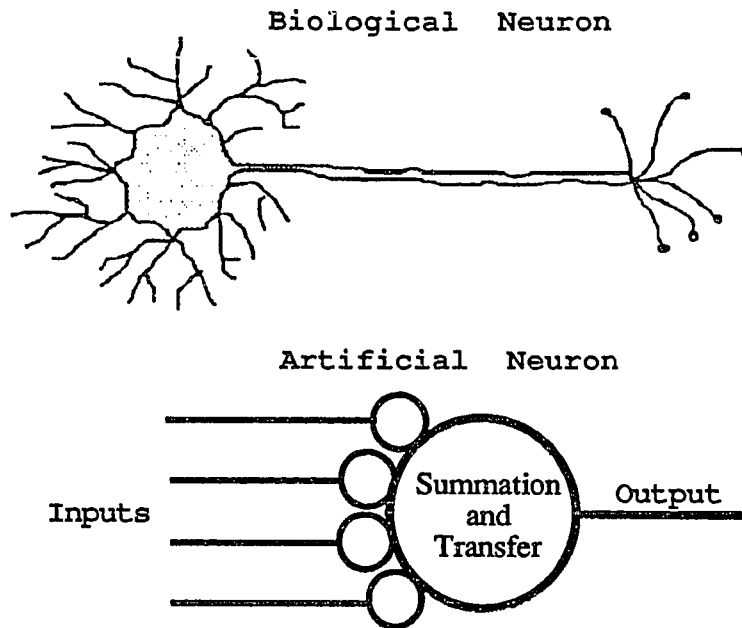


Figure 2.1: The schematic of bio-neuron and artificial neuron

parts based on neurobiological models. The behavior of the network depends heavily on connection details. The state of the network evolves continually with time. Networks are considered clever and intuitive because they learn by example rather than by following programmed rules. With experience, networks become sensitive to subtle relationships in the environment.

The goal of neural network designers is to mimic what the brain does best: associative reasoning, learning, and thought. The ability to learn is one of the distinguishing features of neural networks. They are not programmed the same way artificial intelligence systems are programmed. Information is stored as patterns, not as a series of information bits as in normal computers. Most neural networks are software simulations, with neurons that are only memory locations in a conventional computer.

An *artificial neural network* (ANN) is a model that simulates activities of neurons in a biological neural network (see Figure 2.1). Now we usually write a set of instructions for a computer so that the computer simulates the neurons and the connections between the neurons. We then try to see if these model systems behave like the biological systems. In spite of the enormous amount left to learn about neural networks, we can already build simple artificial neural systems that allow a computer to speak aloud, read in text from a scanner, and control robot arms.

2.2 Neural Network Construction

A neural network consists of layers of neurons which are connected to each other. A neuron is defined as a processing element which sums up the incoming signals and generates an output signal according to some predefined function. The details of how the neurons interconnect are important in building a neural network. Some of the neurons will be used to communicate with the outside world. There are input neurons which receive information from the environment and send it to the inner layers of neurons. The output neurons provide the network's response in the form of objects. All the others are hidden and are part of a large internal abstract pattern. An ANN is somehow like a black box with some magic inside that takes raw materials and gives out expected products. Figure 2.2 depicts the basic configuration of an artificial neural network.

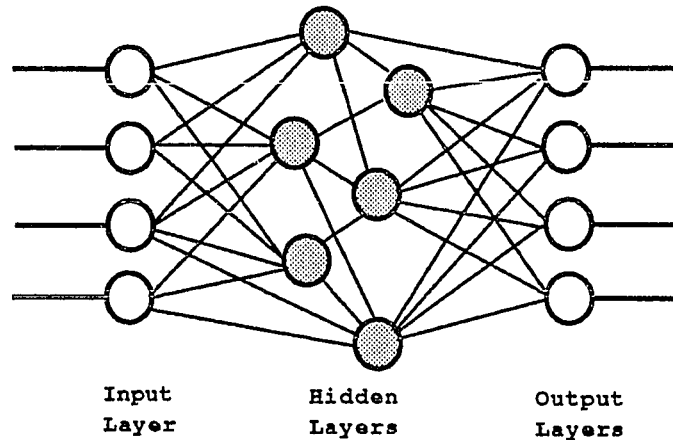


Figure 2.2: An artificial neural network (ANN)

A neural network can be described in terms of its individual neurons, the connections between them, and the activation and transfer functions. These terms are common to all neural networks. First, it is important to understand these concepts and then to explore more complex concepts which explain the differences in various neural networks.

A connection is a unique line of communication that goes from one sending neuron to one receiving neuron. On each connection at the input of a neuron, there is a weight, or connection strength, which is analogous to a synapse. The weight controls the strength of that incoming signal to the neuron. As shown in Figure 2.3, the weight of a particular connection is represented by w_{ij} , where i is the receiving neuron and j is the sending neuron. The i th neuron is represented by u_i where u stands for unit of neuron and i is its number.

The neural diagram looks like a simplified model of a biological network and the terminology is the same. This does not imply that the actual operations of a real

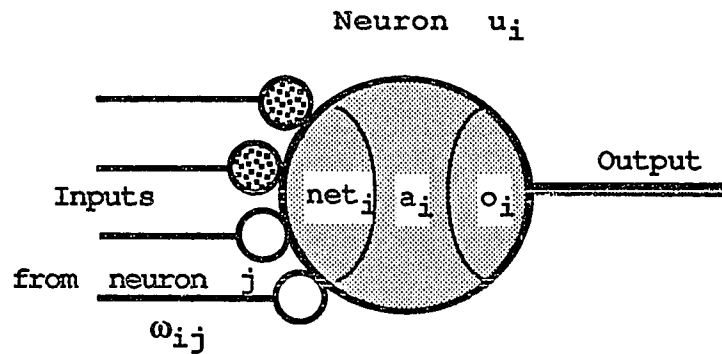


Figure 2.3: Constructure of single artificial neuron

neuron are being depicted. We are merely illustrating the artificial neuron used in a neural network. The neural connections in the brain are much more dense than the diagram shows, but we can know much about the basics of learning using this model.

The state of activation is a way to refer to the state of the neural network at a given time. At any point in time t , the neuron i adds up the weighted inputs to produce an activation value $a_i(t)$ whose value is the sum of the weighted inputs at that time. The activation is passed through an output, or transfer function f_i , which produces the actual output for that neuron for that time, $o_i(t)$. The activation function specifies what the neuron is to do with the signals after the weights have had their effect. Once inside the neuron, the weighted signals are summed to a net value. After summation, the net input of the neuron is combined with the previous state of the neuron to produce a new activation value.

After the weighted sum of the effective inputs has been computed, it is transformed by a typically non-linear function which is called *transfer function*. The transfer function defines how the neuron's activation value is to be output. It can be

linear, sigmoid, or threshold that acts to “transfer” the internally generated sum to a potential output value. Normally, but not always, the transfer function is fixed at the time a network is constructed, and sends out a signal determined by the activation value of the neuron.

The way in which the neurons are connected to each other has an enormous effect on the operation of the network. Specifying the connections determines the type of processing that will occur. No matter what the exact transfer function is, the neuron fires when it recognizes a particular value combination of incoming neural signals. In other words, the operation of a neuron is defined by laws (learning rules) for determining a match between the input vector, consisting of incoming signals, and a weight vector or internal parameter set.

The first and most important decision made for applications of a neural network is choosing the architecture. Neural network architecture refers to how the neurons are connected to each other and what kind of neurons they are. The training of a neural network depends upon what architecture is chosen. Also, the number of inputs and outputs, the number of layers, and the training laws of the network must be specified. When a network is well designed, its overall state after training will be such that new inputs will produce the desired response patterns.

A frequently asked question about neural network training is: How long should the network be trained? The answer is that training time is application specific. The training time duration of a network is based on the performance users required. The test phase is one way of determining how well the network has learned and how well

the network will perform. A major difference between training and testing is that in the test phase, the weights in the network are not updated. The network is trained by adjusting its connection weights on the training cases, and the test cases serve as a way of measuring network performance. During this test phase, the test cases are presented to the network and the network provides results. If the test cases are representative of data the network will see in the real world, we will have a good idea of how well the network will perform its desired task.

2.3 Back-Propagation Network

Currently the most widely known connectionist learning scheme of neural networks is the back-propagation network, a layered network consisting of an input layer of linear neurons, an output layer and at least one layer of nonlinear neurons. Back-propagation is a supervised learning method in which an error signal is fed back through the network altering weights as it goes, in order to prevent the same error from happening again. Weights are updated by means of a chosen learning rule to reach the desired output. For this particular algorithm, the vector of the output signals of the network is compared with an output objective vector, in such a way that the learning process $\frac{\partial E}{\partial w_{ji}} \approx 0$ is achieved. where E is the global error and w_{ji} are the weights.

2.3.1 Learning Behavior

The aim of the learning process is to minimize the global error E of the system by modifying the weights. Given the current set of weights ${}^s w_{ij}$, it is necessary to determine how to increment or decrement them in order to decrease the global error. Each time a particular data set (i, d) – input and desired output – is shown, the back-propagation algorithm modifies the weights to reduce that particular component of the overall error function E , which is defined proportional to the square of the Euclidean distance between the desired output, d , and the actual output, o , of the network for a particular input pattern as a standard form.

$$E = \frac{1}{2} \times (d_k - o_k)^2. \quad (2.1)$$

The actual error function is unimportant in understanding the mechanism of back-propagation. The critical parameter that is passed back through the layers is the error defined by

$${}^s \epsilon_j = -\frac{E}{{}^s I_j}, \quad (2.2)$$

where ${}^s I_j$ is the weighted summation of inputs to j th neuron in layer s . Later, it will be shown that this can be considered as a measure of the local error at j th processing element (PE) in s level.

$${}^s w_{ji} = l_{coef} \times {}^s \epsilon_j \times {}^{s-1} x_i. \quad (2.3)$$

where ${}^s w_{ji}$ is the amount of change to the weights joining i th PE in layer $(s - 1)$ to j th PE in layer s . l_{coef} is the learning coefficient or learning rate, ${}^s \epsilon_j$ is the local error at j th PE in level s , ${}^{s-1} x_i$ is the activation value of i th PE in level $(s - 1)$.

Activation is the state of a PE at a given time. Its value indicates the sum of the weighted inputs for the PE at the particular time.

$${}^s x_j = f[{}^s w_{ji} \times {}^{s-1} x_i] = f[{}^s I_j]. \quad (2.4)$$

With large learning rates, a network may go through large oscillations during learning. In fact, if the rates are too large, the network may never settle or converge. Smaller rates tend to be more stable. Nevertheless, a small learning coefficient can lead to very slow learning. A variation on the algorithm is introduced to resolve this dichotomy.

$${}^s w_{ji} = l_{coef} \times {}^s \epsilon_j \times {}^{s-1} x_i + momentum \times {}^s w_{ji}. \quad (2.5)$$

The momentum term is added so that the learning rate can be increased without causing oscillations. This acts as a low-pass filter on the delta weight (the amount of change to the weights) terms since general trends are reinforced, whereas oscillatory behavior cancels itself. As the learning coefficient tends to zero and the number of updates tends to infinity, the learning algorithm is guaranteed to find the optimized set of weights that gives the least mean square error of the total error function.

2.3.2 Simulation Examples

A couple of learning examples of back-propagation networks worked on UNIX using Aspirin/MIGRAINES (see Appendix A.1) are examined to demonstrate some basic ideas of neural networks' applications.

The first exercise to simulate the input-output mappings of function $\sin\theta$. A C code is written to calculate values of $\sin\theta$ with small θ increment, e.g. 5 degrees, as

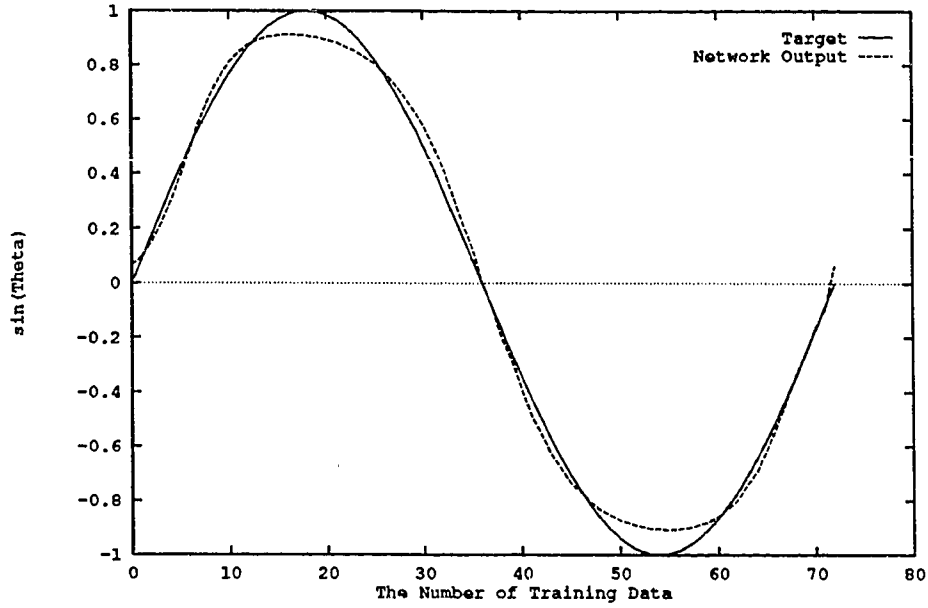


Figure 2.4: Simulation of the sine function

training data for the network. The neural network learns the input-output behavior of function $\sin\theta$ as shown in Figure 2.4. The theoretical data are plotted in a solid line and the neural network simulation results are in a dotted line. From the figure, it is obvious that the simulation results are fairly close to the exact values.

The second attempt is to have a neural network learn the kinematic of a two link planar robot, as shown in figure 2.5. The position of the end-effector of the robot can be expressed as a function of the link parameters, l_1 and l_2 , and the angles, θ_1 and θ_2 . The expression is in equation 2.6.

$$\begin{aligned}
 x &= l_1 \times \cos\theta_1 + l_2 \times \cos(\theta_1 + \theta_2), \\
 y &= l_1 \times \sin\theta_1 + l_2 \times \sin(\theta_1 + \theta_2).
 \end{aligned}
 \tag{2.6}$$

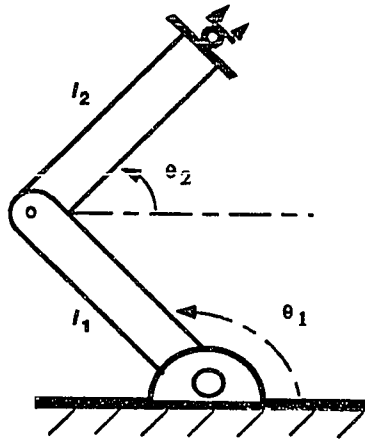


Figure 2.5: A two-link planar robot arm

The training information is derived by incrementing joint angle θ_1 by 9° from 0° to 180° and for each θ_1 increment the second joint angle θ_2 varies from 0° to 180° by 9° . Both training data and testing data can be generated through C codes according to the above equations.

Figures 2.6 - 2.8 illustrate the learning results of back-propagation neural networks. Despite the learning algorithm, some significant factors must be considered when the neural network is built. As mentioned earlier, learning coefficients will determine the connection convergence. Size of training data, as well as times of iterations, will have a significant impact on the learning results, and the transfer function will decide the range of the network outputs. In addition, the number of layers and processing elements per layer are also important decisions.¹ The way to choose proper values for these criteria is based upon user's experience. The more we know or learn about the

¹Most back-propagation networks will have one or two hidden layers, with the number of processing elements in the hidden layers usually falling somewhere between the total number of input and output processing elements. The more complex the relationship between the input data and the desired output, the more processing elements are normally required in the hidden layer.

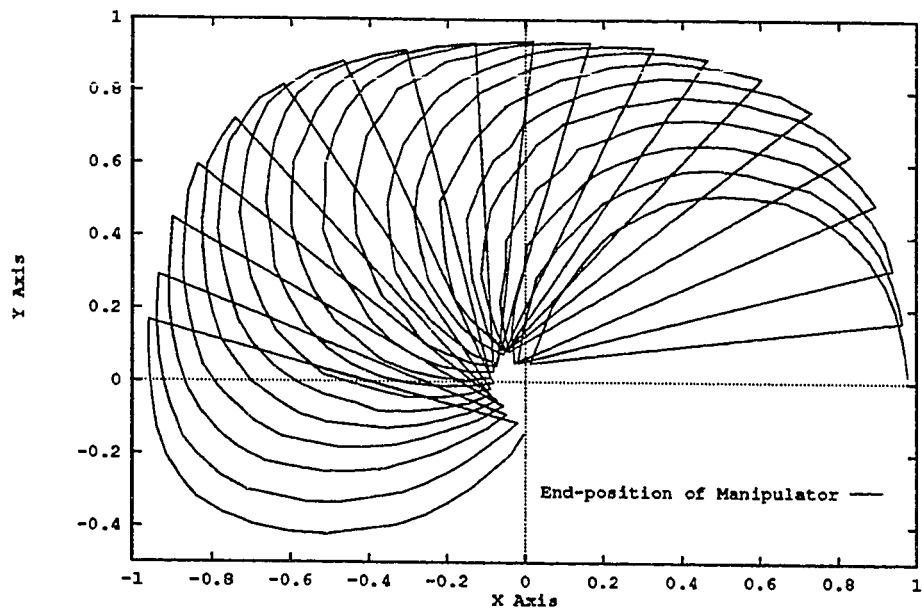


Figure 2.6: Kinematic training results with 300 millions iterations

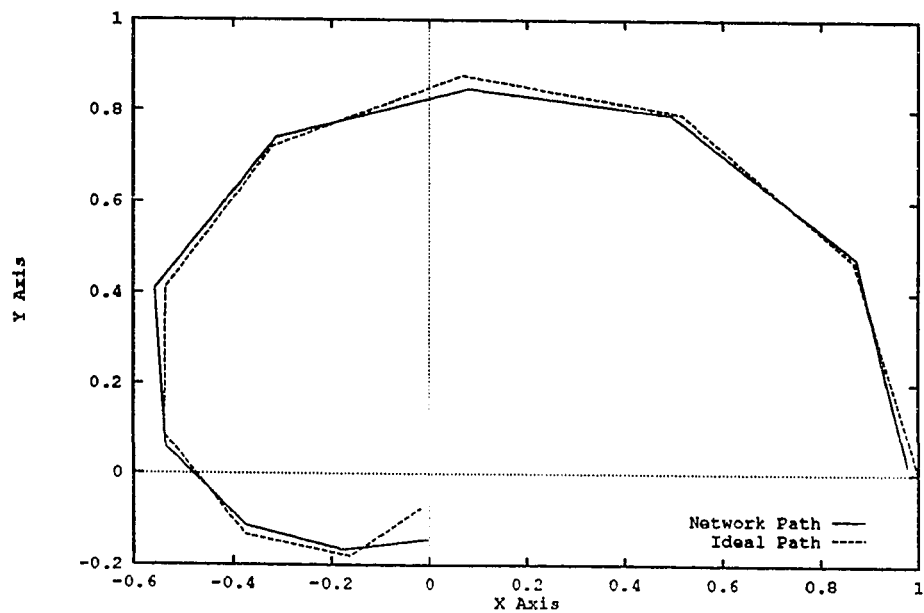


Figure 2.7: Trajectory test of the two-link manipulator

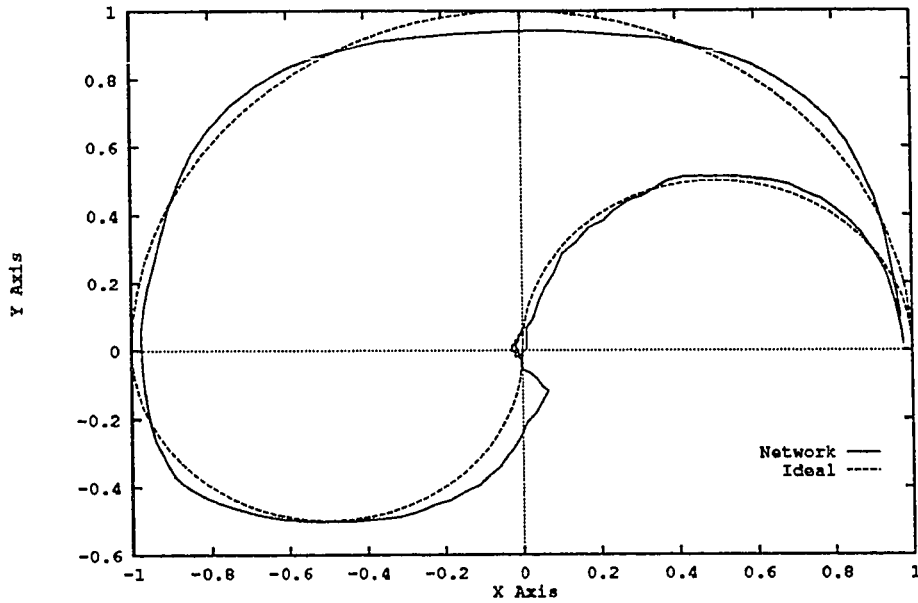


Figure 2.8: Workspace of the two-link manipulator ($0^\circ \leq \theta_1 \leq 180^\circ, 0^\circ \leq \theta_2 \leq 180^\circ$)

features of neural network the better we can utilize the network.

Although the relationships between inputs and outputs of the above examples could be expressed by equation 2.6, the beauty of ANNs is that their learning abilities are not necessarily described by mathematical expressions, which will be demonstrated in later sections.

Chapter 3

Neural Networks and Control

3.1 Control Systems and Neural Networks

A successful control system can respond to changing demands and standards, and should be trying to satisfy many requirements at once – not only keeping the plant output within some limits (and that is meant in the most general sense), but also doing so inexpensively, quickly and efficiently. Especially in robotics research, it has become increasingly understood that the environment in which robots work requires more intelligent control. Many of the assumptions made in control theories should be questioned if robots are to handle the tasks assigned. Systems that are subject to multiple purposes and abstract goals may not be adequately handled with current control methodologies. Learning control is a method of controlling a nonlinear system that may achieve the desired goals. Furthermore, the model of the system may be incompletely known. In addition, such systems may interact with unknown changing

environments and have complex behaviors and goals. ANNs may provide more than conventional control systems. They may be used to build a model of a little known system and to control it.

Neural networks offer certain advantages over traditional adaptive control algorithms and their implementations. Yet, in designing the concept of a learning control system the methods of conventional control should not be discarded. Methods and concepts that have been developed in control can be used in designing more complex learning controllers. For example, if our goal is system identification, a traditional method is to build a (structural or mathematical) model of the system and excite it with the same input as is provided to the actual system. The difference between the model and the system output is used to update the parameters of the model. We may study this technique using more extensive models, such as those attainable by using an ANN. This method of control should build on the powers and tools that the field of conventional control has shown already, but it must extend its tools beyond the rigor of mathematical formulations, so that it can work in a large range of domains, as mentioned above, and under more dynamic and more realistic conditions.

Expectations for applying the neural networks to controllers are associated with learning ability and versatile mapping capabilities from input to output. The versatile mapping capabilities should provide a means of controlling nonlinear plants which cannot be carried out well with traditional linear feedback controllers. The learning ability can reduce human effort in designing controllers and even suggests a potential for discovering better control schemes than presently known.

In most current robot arm position control techniques, the dynamic equations of the robot arm are linearized or assumptions are made to simplify the model into a linear one. In order to deal with nonlinearities and more complex robotic systems we need to consider the nonlinearities of the process. The nonlinear relationships may be modeled, yet it may be difficult, if not impossible, to design controllers for their processes. In addition, causes of optimal design difficulties do not necessarily lie in the system dynamics, but may also be in the specifications required. In practical situations, there are a variety of preferences or evaluations for control performances and most of them are not optimized theoretically. Only a very limited number of criteria can be optimized systematically.

In contrast, the tools and capabilities of conventional adaptive control should not be thrown away, but their limitations should also be recognized. At the root of the trouble, we believe, is the primary aim of control theory to describe plants with manipulable mathematical models, in order to analyze those models and to design controllers for them that can be proven to be stable. However, the plant and environment may be very complex and difficult to model, especially within the framework necessary to apply these techniques. A more general abstract control that can be applied to these complex problems may never be proven to be optimal or consistently convergent, just as there is no way in programming to prove a program of any size or competence to be correct.

A more complex and general control can be developed by not invariably requiring mathematical justification of our models, and opening far wider the doors of what is to

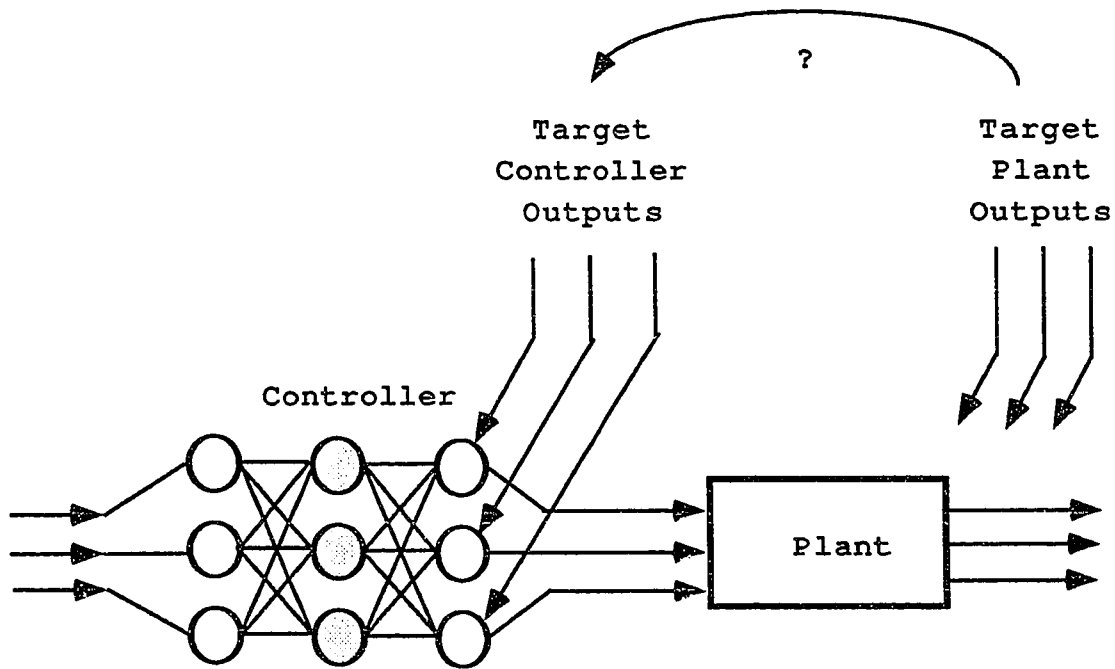


Figure 3.1: Problem of obtaining training information

be considered in control. Neural networks systems display many of the characteristics that are needed in the dramatic future of learning control applied to robotics.

3.2 Training of Neural Networks for Control

Learning control using neural networks somehow requires a training procedure in which desired, or target, outputs of the decision rule, model..., *etc.* are available for a sufficiently varied set of cases. Where does this training information come from? Figure 3.1 outlines the general problem of obtaining training information in neural network control: target plant outputs may be known, but not the control signals that produce them.

Suppose one wants a neural network to learn to control a plant that is too complex,

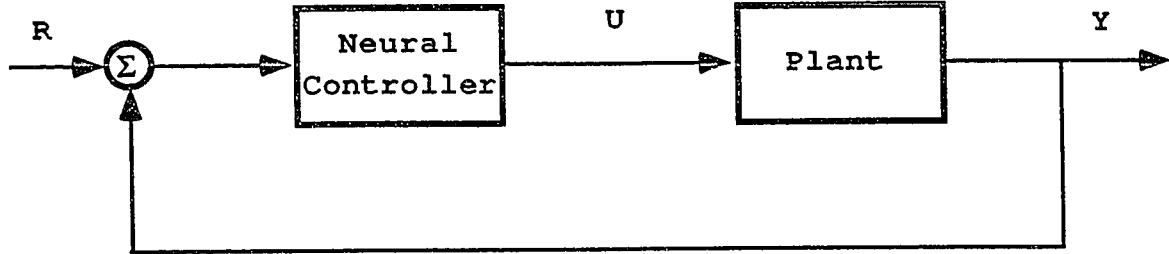


Figure 3.2: A direct neural control system

or about which too little is known, to permit the use of conventional techniques for designing controllers. In a typical control problem, one may have target plant outputs but not target network outputs which are the control signals. How can training signals be provided to the network unless some agency already knows a great deal about how to control the plant? Furthermore, the modeling uncertainty also gives rise to a difficult problem with regard to robustness. Learning by experiments on an actual plant could skip the modeling process and avoid the uncertainty.

3.2.1 Direct Neural Controller

Two types of application of neural network control system architectures are possible. The first type of network serves as a direct controller whose outputs and inputs are directly connected to plants just like ordinary feedback controllers (Figure 3.2). For this type of neural system, we have to overcome an important problem: who can teach the neural controller and how?

Although evaluation of the plant response is quite easy, it is very difficult to show the right answer at each time step of a transient process. If the evaluation can be made on an instantaneous basis, we can tell whether the network output of this moment

is appropriate or not. However, most dynamic systems do not guarantee the success of this local, instantaneous judgment. The evaluation should be made based on a global basis where the appropriateness of the control at each time is judged in a total response curve which spans the time from the initial to the settling. It is preferable to let the neural network learn so that a global and arbitrary evaluation of the total responses of the plant will be optimized and the output of the controller (network) will cancel the plant output error eventually.

In order to satisfy these needs, learning algorithms must be modified to accommodate the network learning procedure. Ichikawa and Sawa [1992] proposed a learning algorithm which is a kind of simulated evaluation process in which a group of networks gradually improves as a whole, by crossing over connection weights among them, or by mutational changes of the weights, according to fitness values assigned to each network by a global evaluation.

Copying an existing controller is another possible way to train the network controller. If there exists a controller capable of controlling the plant, then the information required to train a neural network can be obtained from this controller as shown in Figure 3.3. One might question the utility of this method on the grounds that if there already exists an effective controller, why would it be useful to have another one in the form of a network? Three answers are apparent: first, the existing controller may be a device that is impractical to use (such as a person); second, the adaptive network may be able to form an effective control rule on the basis of a representation of the system state that is easier to measure than the representation required by the

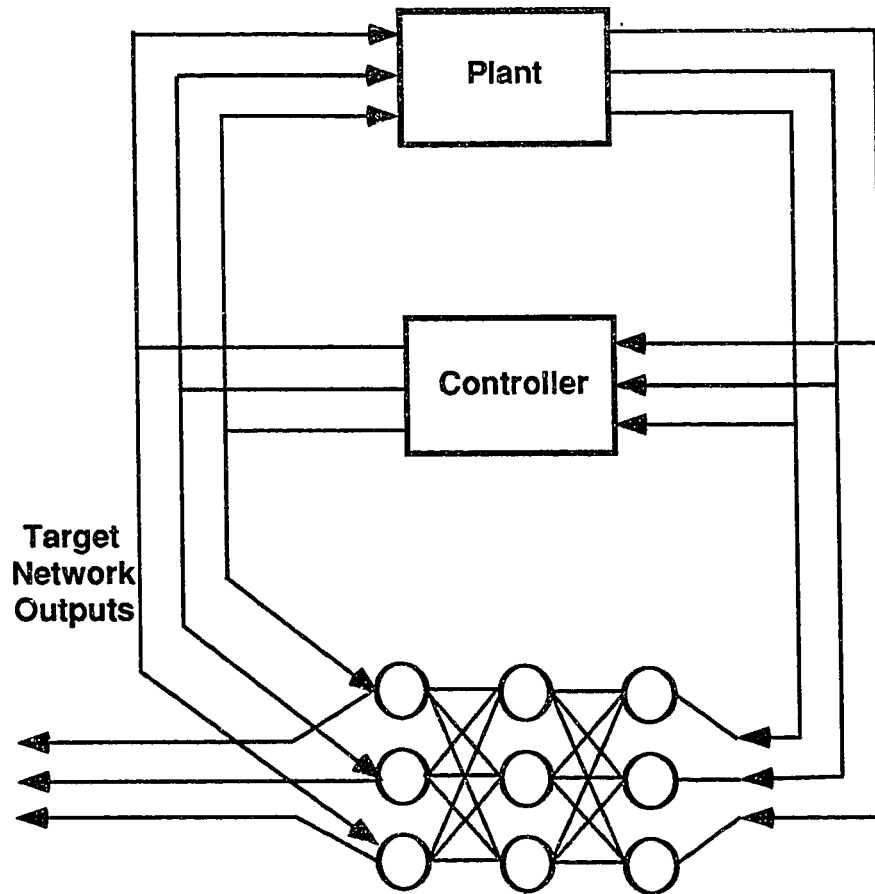


Figure 3.3: Copying training data from an existing controller

existing controller [Miller III *et al.* 1990b]; and third, the existing controller may not be able to respond well enough to changing task requirements which are not modeled.

3.2.2 Neural Network Tuning Controller

The second type of neural control system is the tuning controller coupled with a neural network to adjust the parameters of a conventional controller. This type of control architecture is capable of performing generalization for tracking input commands not previous learned and producing satisfactory responses when operation conditions

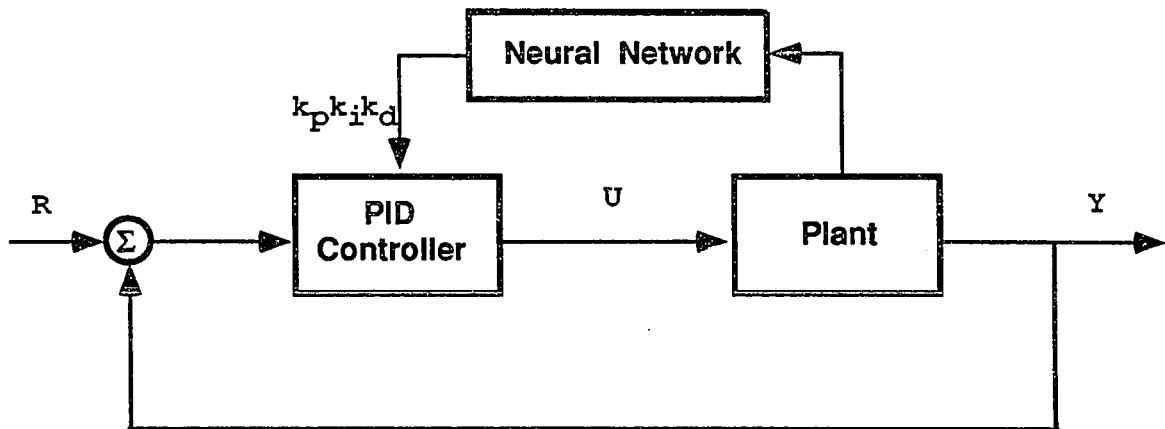


Figure 3.4: A neural network tuned PID control system

change. For example, a PID controller shown in Figure 3.4 can be tuned by adding a neural network to verify its parameters for a given input reference [IWANN 1991].

Although the neural network does not actually control the plant in terms of so-called adaptive neural control system, the idea is useful to help expand the functions of conventional controllers when facing uncertain conditions from the environment. In this thesis, we will address this kind of application on flexible beam control based on deadbeat control theory.

Chapter 4

Control of One-link Flexible Manipulator

4.1 Problem Definition

In recent years the problem of controlling flexible manipulators has been studied extensively for the purpose of performing tasks, such as autonomous satellite retrieval or repair and in orbit assembly of the space station.

Control of a rigid arm robot has two severe limitations: first, the inherent flexibility in the structure and drive trains of robots (as well as in the bases on which they are mounted) makes it impossible to achieve truly high precision; and second, members and drive trains of the robot have to be made very stiff, and must therefore be heavy, in order to get some degree of precision. These limitations typically limit the robot to slow speeds and/or require high levels of drive power. A flexible beam

manipulator is recommended to overcome these limitations.

As we shall see, the increase of velocity and decrease of mass of robotic systems become two important topics of current research in this field. As a consequence of this development, the manipulators undergo considerable elastic deformations. If the oscillations caused by elasticity of manipulators could be compensated by appropriate control concepts, it would be possible to build light weight robots suitable for space applications.

4.2 Mathematical Model

An accurate dynamic model for the flexible arm is required in order to control the tip position of the manipulator. A simplified analytic model is derived by Cannon and Schmitz [1984]. The arm is modeled as a continuous, pinned-free beam of length l whose moment of inertia about the root is I_b , with an additional lumped inertia I_h at the actuator end (hub). Displacement of any point P , along the beam at a distance x from the hub is given by the hub angle $\theta(t)$ and the elastic deflection $\omega(x, t)$ measured from the line ox , as shown in Figure 4.1.

For the beam, we use the Euler-Bernoulli model for which rotary inertia and shear deformation effects are ignored. so that the beam deflection $y(x, t)$ can be expressed as

$$y(x, t) = \omega(x, t) + x\theta(t). \quad (4.1)$$

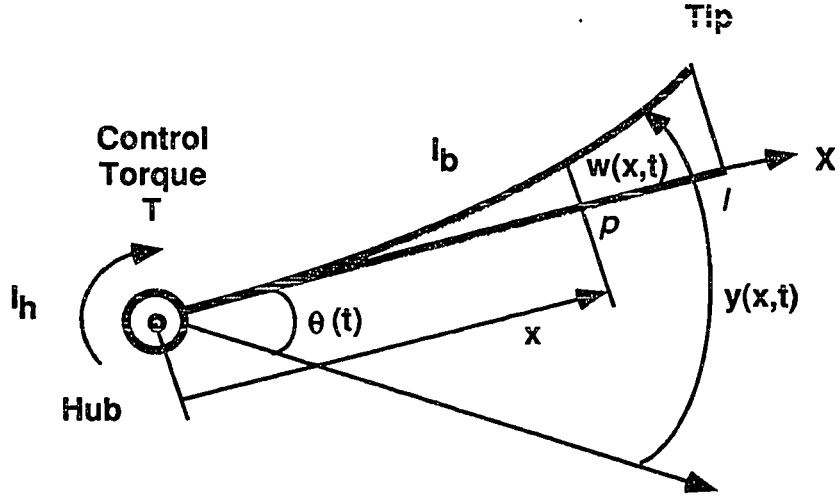


Figure 4.1: Schematic of a single flexible manipulator

Then the kinetic energy K and potential energy V are

$$\begin{aligned}
 K &= \frac{1}{2} [I_h \dot{\theta}^2 + \int_0^l (\frac{\delta y}{\delta t})^2 dm], \\
 V &= \frac{1}{2} [\int_0^l EI (\frac{\delta^2 y}{\delta x^2})^2 dx - T\theta].
 \end{aligned} \tag{4.2}$$

The assumed-modes approach is employed to derive the beam deflection:

$$y(x, t) = \sum_{i=0}^{\infty} \phi_i(x) q_i(x), \tag{4.3}$$

where i denotes various vibration modes, $\phi_i(x)$ are the mode shapes of the beam, and $q_i(x)$ are the corresponding generalized coordinates.

Now the Lagrangian, $L = K - V$, can be found as

$$2L = \sum_{i=0}^{\infty} (I_h + I_b) \dot{q}_i^2 - \sum_{i=0}^{\infty} (I_h + I_b) \omega_i^2 q_i^2 + T (q_0 + \sum_{i=0}^{\infty} \frac{d\phi_i}{dx}(0) q_i), \tag{4.4}$$

where q_0 describes the rigid body mode, ω are the pinned-free frequencies, and $\frac{d\phi_i}{dx}(0)$ are the actuator modal gains. The q_i 's are considered as the generalized coordinates

of the system so that one can apply the Euler-Lagrange equations. The equations of motion are, therefore,

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{q}_i}\right) - \frac{\delta L}{\delta q_i} = 0, \quad (4.5)$$

where $i = 0, 1, 2, \dots, n$. Substituting equation 4.4 into equation 4.5, we have

$$\ddot{q}_i = -\left(\frac{T}{I_h}\right) \int_0^l \phi_i x dm - q_i \omega_i^2 \left[1 + \frac{(\int_0^l \phi_i x dm)^2}{I_h}\right] - \sum_{j \neq i}^n \left(\frac{q_j \omega_j^2 \int_0^l \phi_j x dm \int_0^l \phi_i x dm}{I_h}\right). \quad (4.6)$$

The state space equations can now be derived in the form

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U}, \\ \mathbf{Y} &= \mathbf{C}\mathbf{X}. \end{aligned} \quad (4.7)$$

The matrices \mathbf{X} , \mathbf{A} , \mathbf{B} , and \mathbf{C} are expressed for the case $n = 3$, but the same pattern holds for all n :

$$\mathbf{X} = \begin{bmatrix} q_0 & \dot{q}_0 & q_2 & \dot{q}_2 & q_3 & \dot{q}_3 \end{bmatrix}^T, \quad (4.8)$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\omega_1^2 & -2\xi_1\omega_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\omega_2^2 & -2\xi_2\omega_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\omega_3^2 & -2\xi_3\omega_3 \end{bmatrix}. \quad (4.9)$$

$$\mathbf{B} = \frac{1}{I_h + I_b} \begin{bmatrix} 0 & 1 & 0 & \frac{d\phi_1}{dx}(0) & 0 & \frac{d\phi_2}{dx}(0) & 0 & \frac{d\phi_3}{dx}(0) \end{bmatrix}^T, \quad (4.10)$$

$$\mathbf{C} = \begin{bmatrix} l & 0 & \phi_1(l) & 0 & \phi_2(l) & 0 & \phi_3(l) & 0 \\ 0 & 1 & 0 & \frac{d\phi_1}{dx}(0) & 0 & \frac{d\phi_2}{dx}(0) & 0 & \frac{d\phi_3}{dx}(0) \end{bmatrix}. \quad (4.11)$$

The two rows of the measurement output matrix \mathbf{C} are the tip-sensor and the hub-rate-sensor measurement vectors. The numerical data is adopted from the experiments in [Cannon and Schmitz 1984] to identify the elements of above matrices (see Appendix B.1.1). Theoretically, the number of vibration modes is infinity, but practically a finite value is used due to the limitation of the bandwidths of the actuator and sensors.

4.3 Deadbeat Control

The first vibration mode of the flexible beam system is investigated through the rest of this thesis to examine the study objectives. Hence, the discrete manipulator dynamic of this fourth-order system in state space form can be represented as

$$\begin{aligned} \mathbf{X}_{k+1} &= \mathbf{P}\mathbf{X}_k + \mathbf{Q}\mathbf{U}_k. \\ \mathbf{Y}_k &= \mathbf{C}\mathbf{X}_k. \end{aligned} \quad (4.12)$$

Suppose that the initial condition is set to be zero. The system transfer function

can be obtained:

$$G(z) = \mathbf{C}(z\mathbf{I} - \mathbf{P})^{-1}\mathbf{Q}. \quad (4.13)$$

This transfer function is processed by `MATRIXX` and its general form for the fourth-order system is

$$G(z) = \frac{b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + b_4z^{-4}}{1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} + a_4z^{-4}}. \quad (4.14)$$

where a_1, a_2, \dots, b_3 , and b_4 are constant coefficients.

Before the control strategy is explored, it is better to determine whether the system is controllable. The system controllability matrix can be derived as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{Q} & \mathbf{PQ} & \mathbf{P}^2\mathbf{Q} & \mathbf{P}^3\mathbf{Q} \end{bmatrix}. \quad (4.15)$$

When the rank of matrix \mathbf{M} meets the order of the system, we can confirm that the system is controllable.

The easiest method of testing the response of the system is to introduce a step function which is accomplished by changing the input quickly to a new position and observing the reaction of the output. If the sampling time of this discrete system is long, however, a control that moves the state along as rapidly as possible might be feasible. A system with this kind of controller which beats the state to a dead stop in at most n steps, where n is the order of the system, is referred to as a *deadbeat* system.¹ Such a digital system has a remarkable property; that is, all of its poles are at $z = 0$. This response is ideal for many applications. In many practical situations, digital systems are designed to be deadbeat, if possible.

¹In some literature, a *deadbeat* system is also named *critical damping* system that the system output reaches the desired value as rapidly as possible and without overshooting [Baeck 1968].

If we employ deadbeat control strategy to our system where sampling time is 1 second, then a compensator would be derived:

$$D(z) = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2} + q_3 z^{-3} + q_4 z^{-4}}{1 - p_1 z^{-1} - p_2 z^{-2} - p_3 z^{-3} - p_4 z^{-4}} \quad (4.16)$$

where

$$q_0 = \frac{1}{b_1 + b_2 + b_3 + b_4},$$

$$q_1 = a_1 q_0,$$

$$q_2 = a_2 q_0,$$

$$q_3 = a_3 q_0,$$

$$q_4 = a_4 q_0,$$

$$p_1 = b_1 q_0,$$

$$p_2 = b_2 q_0,$$

$$p_3 = b_3 q_0,$$

$$p_4 = b_4 q_0,$$

and these p, q constant coefficients must satisfy

$$p_1 + p_2 + p_3 + p_4 = 1,$$

$$q_0 + q_1 + q_2 + q_3 + q_4 = U(4). \quad (4.17)$$

where $U(4)$ is the fourth step or final control torque applied to the control system.

Thus, the system close-loop transfer function will be

$$G_c(z) = \frac{p_1 z^3 + p_2 z^2 + p_3 z + p_4}{z^4}. \quad (4.18)$$

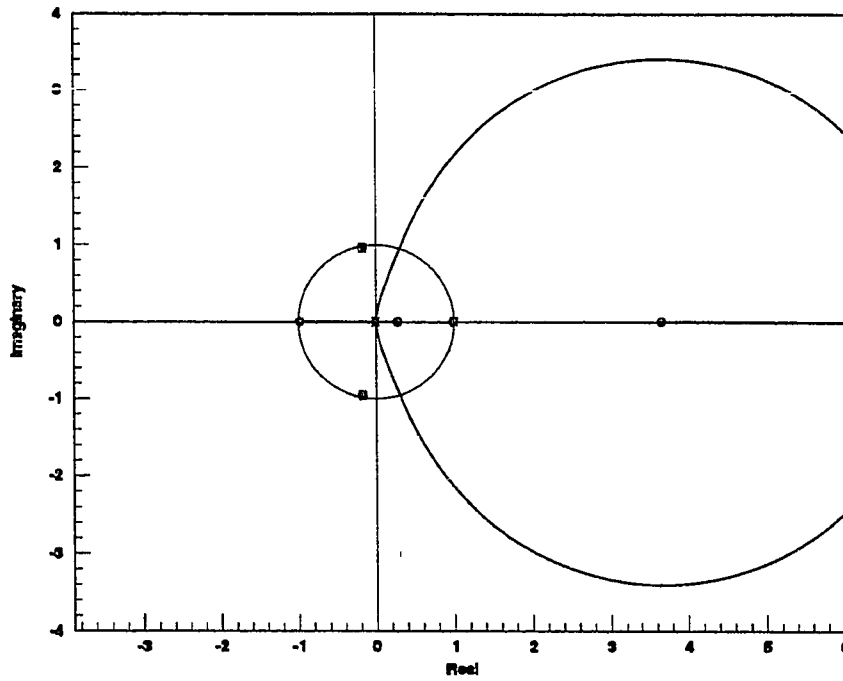


Figure 4.2: Root locus of deadbeat control system

From Figure 4.2, we can see that the deadbeat controller drives all system poles to the origin in z -plane. Figure 4.3 shows the deadbeat control effort and system response for the free payload flexible manipulator; the solid line indicates the step response and the dotted line represents the control efforts.

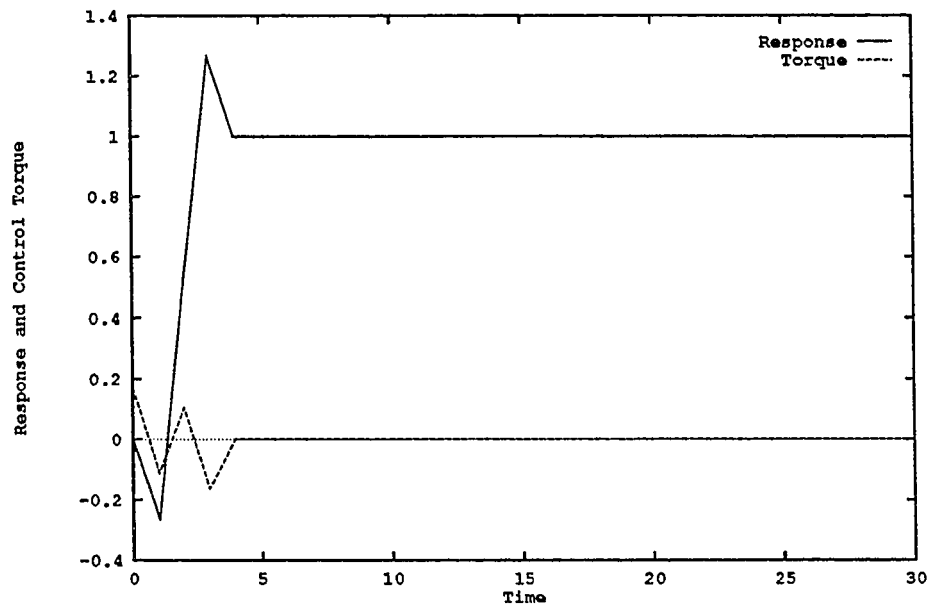


Figure 4.3: Response and control effort of deadbeat control system

Chapter 5

Adaptive Neural Network Controller

5.1 System Performance vs. Payload Variation

Many of today's robots are required to perform sequential tasks; for example, it may be required to pick up an unknown mass, and then move to a pre-specified location or along a pre-planned trajectory. Therefore, a good controller design should have the ability to suppress the undesired vibratory motion which is due to payload variation. For an arm with considerable rigidity, the undesired dynamics can be suppressed by using a high gain controller. Yet, for an arm with considerable flexibility or when high speed manipulation is required, the effectiveness of the designed compensator is sensitive to payload variation. As shown in Figure 5.1, the free payload compensator is not suitable for the system with variable, finite payload.

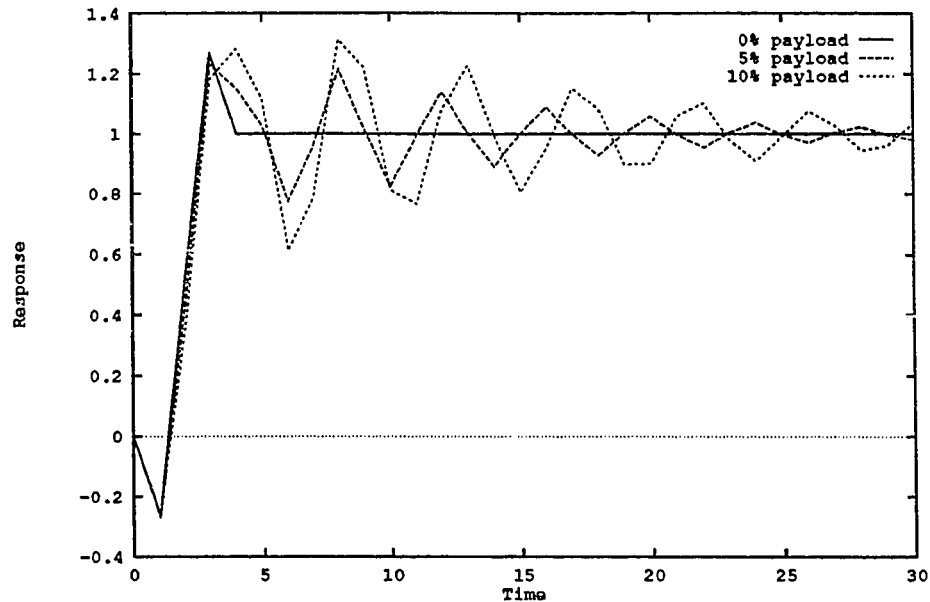


Figure 5.1: Response of payloaded system with free-payload compensator

Due to the parameter changes, such as moment of inertia and natural frequency caused by changes of the mass of payload,¹ system dynamics are no longer the same as those of the free-loaded system where system dynamics matrix \mathbf{P} and control input matrix \mathbf{Q} in the state space (see equation 4.12) are varied. Hence, a payload adaptation mechanism capable of taking care of both the system identification and compensator modification according to payload conditions is desired, in order to handle the payload variations.

¹Assume the damping ratio is constant because its changes are very small. Typically, damping ratio of the flexible beam ranges from 0.007 to 0.01 [Hastings and Book 1987].

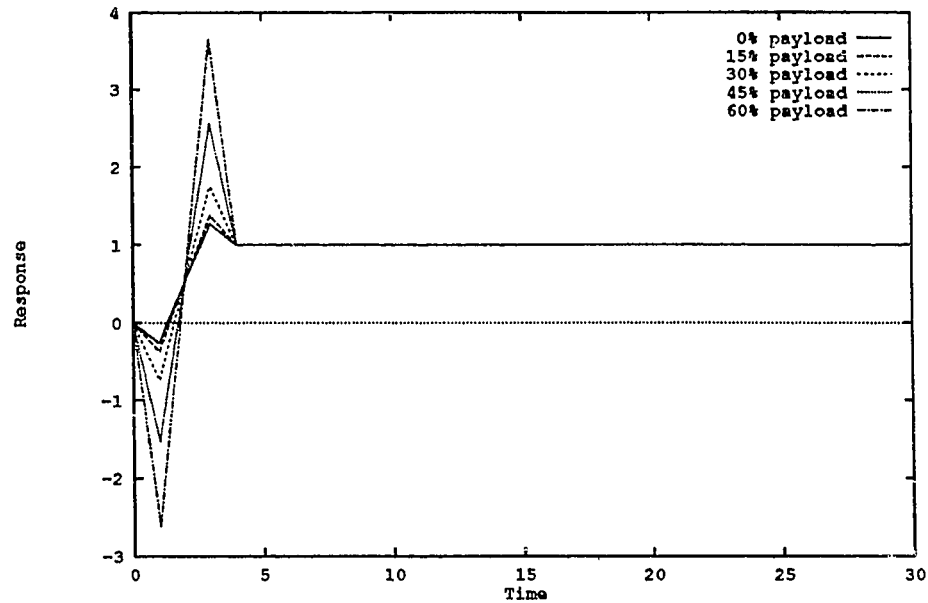


Figure 5.2: Deadbeat control for various payload

5.2 Adaptive Neural Controller Design

We now intend to adopt a back-propagation neural network to design a neural network controller that can adapt to the unpredicted variation in the payload using the approach depicted in Figure 3.4.

Because a deadbeat controller is to beat the system response to the desired value as rapidly as possible, it can result in a significant overshoot when the payload is too heavy, e.g., 60% of the beam mass or more. Such an undesirable overshoot is shown in Figure 5.2. A reasonable payload range, specified within 40% of the beam mass, will be considered in the following study.

If the flexible arm is equipped with a payload measure device, the system payload measurement will be linked to the neural network which performs two tasks mentioned

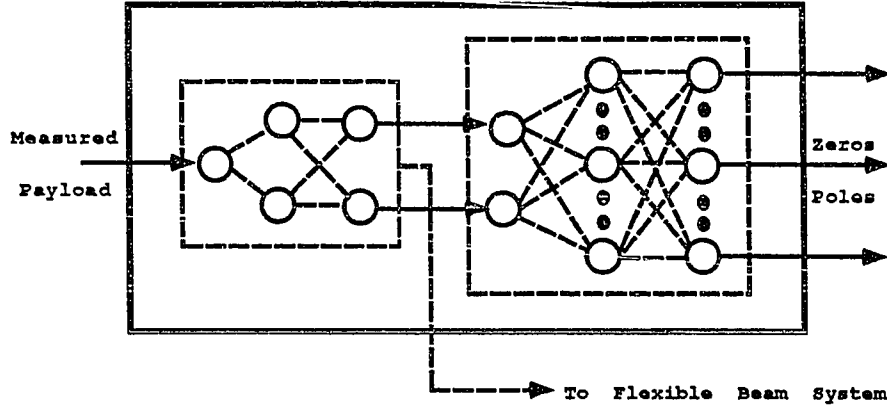


Figure 5.3: Configuration of adaptive neural network

in section 5.1. The neural network design, as shown in Figure 5.3, consists of two subnetworks. The first network takes payload measurement as its input and outputs moment of inertia and natural frequency for the next network's inputs as well as for system identification. For the second subnetwork, a problem arises: What would be the best choice for this network to learn? It is better to review the system dynamics before we answer this question.

In order to analyze the control system through the Lagrangian for our design purposes, the dynamic equations of the first mode vibrating arm can be formulated as:

$$\begin{aligned} \ddot{q}_0 &= b_0 u, \\ \ddot{q}_1 + 2\xi_1 \omega_1 \dot{q}_1 + \omega_1^2 q_1 &= b_1 u. \end{aligned} \quad (5.1)$$

Hence the characteristic equation of the fourth-order open-loop system in z domain can be written as:

$$(z - 1)^2(z^2 + cz + d) = 0. \quad (5.2)$$

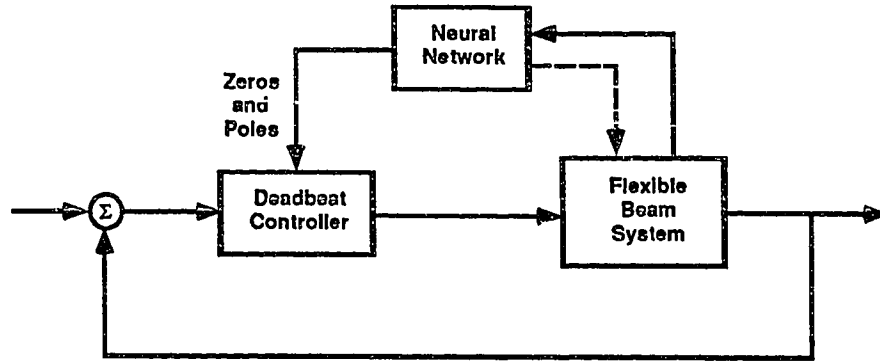


Figure 5.4: Adaptive neural deadbeat control system

And from the previous deadbeat controller design, we can find that the compensator has two zeros at $z = 1$. Obviously, these zero-pole cancellations are very important criteria for the deadbeat controller to accomplish system stability. According to this observation, the neural network is expected to handle the cancellation perfectly; otherwise the order of the system will be increased. Eventually, the second subnetwork is designed to learn the locations of zeros and poles of the deadbeat compensator so that the cancellation will be done fairly. It has been proven that the network can learn the pattern $z = 1$ perfectly via computer simulations. Therefore, the adaptive neural control system is constructed as Figure 5.4 which employs a back-propagation network to tune the deadbeat controller as well as to identify the flexible beam dynamics with respect to payload variations.

5.3 Simulation Results

Data analyzed in this simulation study is based on the experimental results derived in [Wang *et al.* 1989] and the ideal deadbeat compensator is obtained through

MATRIX analyses. We adopt nine sets of various payload data (see Appendix B.1.2) to train the neural network which learns the patterns of zero-pole locations of compensator with payload over beam mass from 0% to 40% increased by 5%.

After the back-propagation network has been trained, it is evaluated by randomly assigning payload conditions, e.g., 3%. Figures 5.5 to 5.9 demonstrate the capability of the designed neural network to handle the payload variation for the flexible arm robot. Within the specified payload range, it is clear that the adaptive neural network is capable of controlling the system behavior with payload up to 38% with stable response. As the payload increases, the response of the adaptive neural network controller takes longer to settle. Nevertheless, the adaptive neural controller is able to deliver adequate control responses.

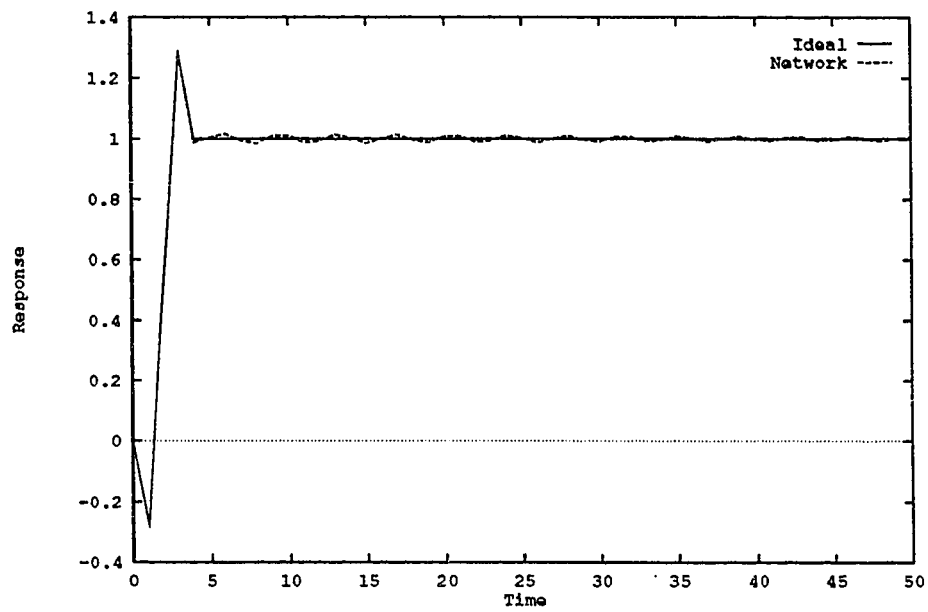


Figure 5.5: System response of the adaptive neural network controller with 3% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).

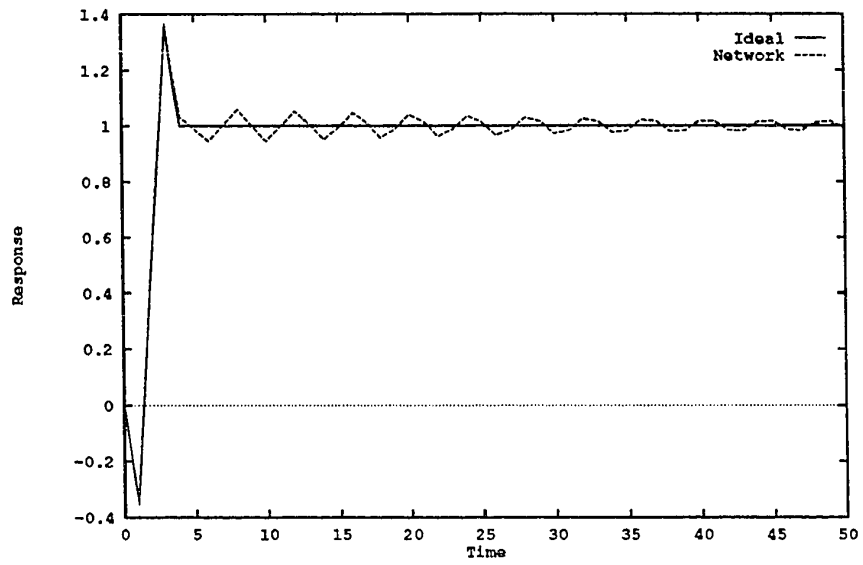


Figure 5.6: System response of the adaptive neural network controller with 13% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).

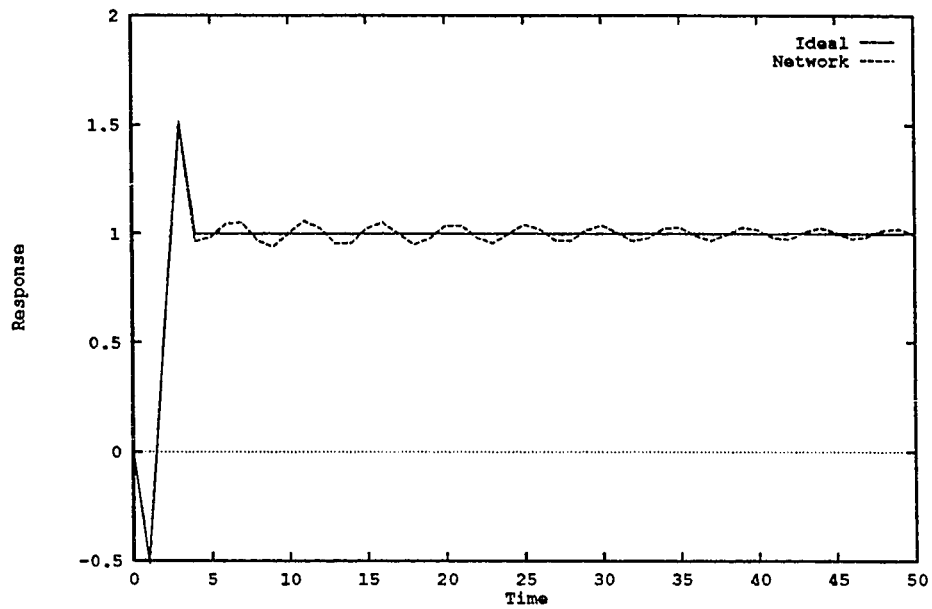


Figure 5.7: System response of the adaptive neural network controller with 23% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).

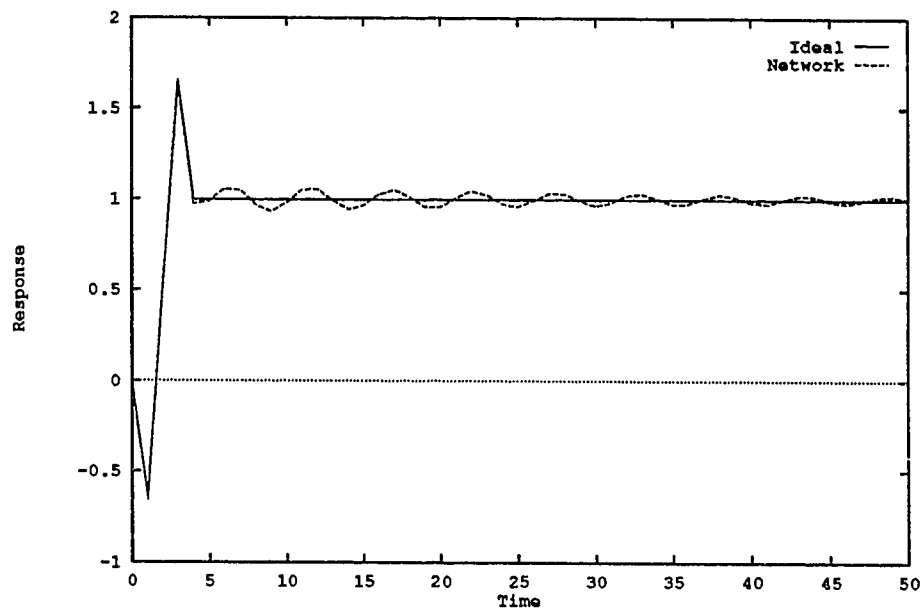


Figure 5.8: System response of the adaptive neural network controller with 28% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).

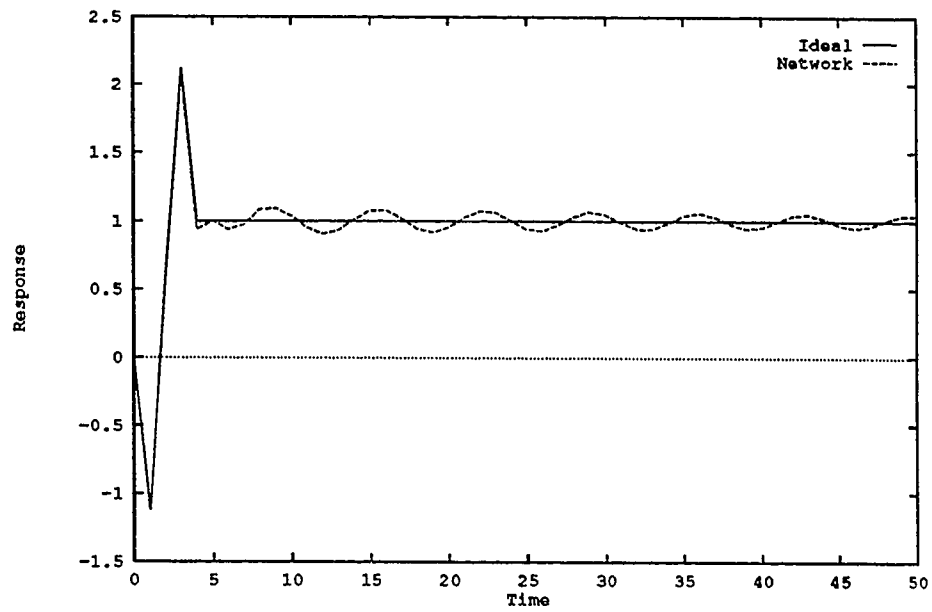


Figure 5.9: System response of the adaptive neural network controller with 38% payload. The plot compares the neural network simulation results (dotted line) with the ideal response (solid line).

Chapter 6

Conclusions and Future Work

It has been shown that a simple architecture in which a neural network takes over the conventional feedback controller has promising potential. One distinguishing feature about ANN is its self-tuning capability. Simulation results reveal the neural network's versatility to adapt to various payload conditions for a single flexible robot manipulator. The work presented here has been based on simulations; experimental verification should be performed in the future.

In terms of flexible beam control strategy for a single link, the use of more effective control schemes, such as a low-order compensator or a LQG controller instead of deadbeat controller, is suggested for future study with sufficient vibration modes to represent the system dynamic. Neural networks should be applied to these improved control systems based upon the work presented in this thesis.

In addition, the state-of-the-art neural networks are not just used to replace the

conventional adaptive controllers but to incorporate intelligence with robotic applications. In the future, we might expect robots to handle tasks and to reason or respond logically like human beings with intelligent neural network control. Challenges of neural network control are lying ahead and await further investigation and research.

Bibliography

- [1] Baeck, Henry S. 1968. *Practical Servomechanism Design*. McGraw-Hill.
- [2] Barto, Andrew G.; Sutton, Richard S.; and Anderson, Charles W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics* SMC-13(5).
- [3] Cannon, Robert H. and Schmitz, Jr. Eric 1984. Initial experiments on the end-point control of a flexible one-link robot. *Int. J. Robot. Res.* 3(3).
- [4] D'Azzo, John J. and Houpis, Constantine H. 1966. *Feedback Control System Analysis and Synthesis*. McGraw-Hill.
- [5] Franklin, Gene F.; Powell, J. David; and Workman, Michael L. 1990. *Digital Control of Dynamic Systems*. Addison-Wesley.
- [6] Hastings, G. G. and Book, W. J 1987. A linear dynamic model for flexible robotic manipulators. *IEEE Control System Magazine* 1(1).
- [7] Ichikawa, Yoshiaki and Sawa, Toshiyuki 1992. Neural network application for direct feedback controller. *IEEE Trans. on Neural Networks* 3(2).
- [8] IWANN, International Workshop 1991. *Artificial Neural Networks*. Springer-Verlag. Granada, Spain.
- [9] Kung, Sun-Yuan and Hwang, Jenq-Neng 1989. Neural network architectures for robotic applications. *IEEE Trans. on Robotics and Automation* 5(5).
- [10] Menq, Chia-Hsiang and Chen, Jian-Shiang 1988. Dynamic modeling and payload-adaptive control of a flexible manipulator. *Proc. IEEE Conf. on Robotics and Automation* 1.
- [11] Miller III, W. Thomas; Hewes, Robert P.; Glanz, Filson H.; and Kraft III, L. Gordon 1990a. Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. *IEEE Trans. on Robotics and Automation* 6(1).
- [12] Miller III, W. Thomas; Sutton, Richard S.; and Werbos, Paul J. 1990b. *Neural Networks for Control*. MIT Press.
- [13] Narendra, Kumpati S. and Parthasarathy, Kannan 1990. Identification and control of dynamical system using neural networks. *IEEE Trans. on Neural Networks* 1(1).
- [14] Phillips, Charles L. and Harbor, Royce D. 1988. *Feedback Control Systems*. Prentice Hall.

- [15] Stanley, Jeannette 1988. *Introduction to Neural Networks*. California Scientific Software.
- [16] Wang, David and Vidyasagar, M. 1991. Transfer functions for a single flexible link. *Int. J. Robot. Res.* 10(5).
- [17] Wang, Wen-Jien; Lu, Shui-Shong; and Hsu, Chen-Fa 1989. Experiments on the position control of a one-link flexible robot arm. *IEEE Trans. on Robotics and Automation* 5(3).

Appendix A

Aspirin/MIGRANES

A.1 Aspirin Language and MIGRANES Interface

The Aspirin/MIGRANES system is the copyright of Russell Leighton and the MITRE Corporation, and it is licenced free of charge for research and development purposes.

The Aspirin/MIGRANES system is intended to simplify and speed up the process of investigating neural network paradigms as well as to facilitate the efficient execution of large, non-trivial neural network systems.

Aspirin is a declarative language used to describe arbitrarily complex neural networks. The goal of Aspirin is to make simple (i.e., commonly used) network constructs easy to describe, yet allow any network topology to be specified. An Aspirin description of a network is used to generate a computer program to simulate that network. Aspirin encompasses a very large range of capabilities, enough to consistently describe

the majority of feed-forward network topologies and the related learning algorithms in an economical manner. When new cases arise, Aspirin enables a user to define new functionality by specifying replacement simulation code or by adding extensions to Aspirin's code generators.

MIGRAINES is an interface for evaluating and interacting with a neural network simulation. Utilities exist for moving quickly from an Aspirin description of a network directly to an executable program for simulating and evaluating that network. The interface MIGRAINES has been kept intentionally separate from Aspirin so that the limitations of MIGRAINES do not restrict the performance of Aspirin. However, in practice, Aspirin and MIGRAINES are used as a single, cohesive unit. This combination allows for simple specification and creation of efficient neural networks.

A.2 Example Files

The following two files illustrate an Aspirin program and how MIGRAINES is used to navigate through the network.

A.2.1 Aspirin File

Aspirin file of two link manipulator simulation:

```
DefineBlackBox Robot
{
    OutputLayer-> Out
    InputSize-> 2
    Components-> {
        PdpNode3 Hidden [10]
        {
            InputsFrom-> $INPUTS
        }
        PdpNode3 Out [2]
        {
            InputsFrom-> Hidden
        }
    }
}
```

A.2.2 MIGRAINES Command File

This command file illustrates some features of the MIGRAINES user interface.

```
# This program saves the hidden layer and output vectors
# for plotting
# This program assumes you have loaded a test.df file

# Keep sure we're in the right place
poproot

# Load the network
load Network.save

push Robot

# Get some values in the hidden nodes
cycle 1

# no header
pnoheader

echo      $<<<<$ Open pipe to file $>>>>$

push Robot:Hidden

# This opens a file and dumps the data...
popenNodes HiddenValues cat $>$ hidden\_values.pdat

pop

push Robot:Out
```

```
# This opens a file and dumps the data...  
echo "opens a file and dumps the data..."  
popenNodes OutputValues cat $>$ output\_values.pdat  
echo "Done."  
  
# There are 99 more patterns, after each pattern  
# all node value pipes are updated  
cycle 99  
  
echo      $<<<<$ Close pipes $>>>>$  
pclose HiddenValues  
pclose OutputValues  
quit
```

Appendix B

Numerical Data

B.1 Experimental Data of Literatures

B.1.1 Data of Flexible Beam for First Vibration Mode

Data of table B.1 is from [Cannon and Schmitz 1984], where beam length $l = 1$ m and the total moment of inertia $I_T = 0.44$ kgm^2 .

<i>Mode</i>	<i>Frequency</i> ω	<i>Modal Damping</i> ξ	<i>Actuator Gain</i> $\frac{d\phi_i}{dx}(0)$	<i>Tip-sensor Gain</i> $\phi_i(l)$
<i>Rigid Body</i>	0	0	1.0	1.12
<i>First Mode</i>	1.88	0.015	2.97	-1.1

Table B.1: Model parameters for first flexible mode

B.1.2 Data of Flexible Beam with Payload Variations

Data of table B.2 is from [Wang *et al.* 1989] which is a follow-up study of [Cannon and Schmitz 1984].

B.2 Simulation Data

Data of Table B.3 is used to train the adaptive neural network. The zeros and poles of compensators from Table B.4 are target outputs of the network for testing, and the actual network outputs are tabulated in Table B.5.

<i>Payload/Beam Mass</i>	<i>Frequency of First Mode (Hz)</i>
0%	1.75
30%	1.10
60%	0.60
90%	0.50

Table B.2: Frequency changes with respect to payload variations

%	$\omega(Hz)$	$I_T(kgm^2)$	$D(z)$ poles				$D(z)$ zeros		
0%	1.7500	0.4400	0.2867	1	-0.7749±j0.5688	1	1	-0.1734±j0.9585	
5%	1.6766	0.4546	0.3043	1	-0.7981±j0.5726	1	1	-0.1028±j0.9697	
10%	1.6083	0.4684	0.3215	1	-0.8217±j0.5758	1	1	-0.0364±j0.9755	
15%	1.5097	0.4909	0.3479	1	-0.8604±j0.5796	1	1	0.0599±j0.9758	
20%	1.3968	0.5108	0.3808	1	-0.9126±j0.5816	1	1	0.1697±j0.9645	
25%	1.2575	0.5393	0.4256	1	-0.9929±j0.5775	1	1	0.3026±j0.9335	
30%	1.1000	0.5720	0.4832	1	-1.1152±j0.5525	1	1	0.4463±j0.8766	
35%	0.9719	0.6005	0.5362	1	-1.2533±j0.4911	1	1	0.5557±j0.8139	
40%	0.8541	0.6276	0.5905	1	-1.4303±j0.3275	1	1	0.6486±j0.7443	

Table B.3: Training data of the adaptive neural controller

%	$\omega(Hz)$	$I_T(kgm^2)$	$D(z)$ poles				$D(z)$ zeros		
3%	1.7078	0.4488	0.2967	1	-0.7879±j0.5710	1	1	-0.1329±j0.9656	
13%	1.5600	0.4780	0.3342	1	-0.8400±j0.5778	1	1	0.0107±j0.9768	
23%	1.3249	0.5255	0.4033	1	-0.9515±j0.5808	1	1	0.2388±j0.9508	
28%	1.1801	0.5554	0.4530	1	-1.0479±j0.5692	1	1	0.3743±j0.9084	
38%	0.9083	0.6150	0.5648	1	-1.3413±j0.4266	1	1	0.6068±j0.7777	

Table B.4: Testing data of the adaptive neural controller

%	$D(z)$ poles				$D(z)$ zeros		
3%	0.2975	1	-0.7879±j0.5723	1	1	-0.1290±j0.9677	
13%	0.3312	1	-0.8378±j0.5772	1	1	0.0027±j0.9712	
23%	0.4046	1	-0.9530±j0.5770	1	1	0.2459±j0.9524	
28%	0.4539	1	-1.0469±j0.5692	1	1	0.3790±j0.9122	
38%	0.5681	1	-1.3576±j0.4009	1	1	0.6095±j0.7720	

Table B.5: Simulation results of the zeros and poles of the deadbeat compensators

Appendix C

Symbols

C.1 Neural Network Symbols

$a_i(t)$	Activation value of i -th neuron at time t .
E	Global error of a neural network.
${}^s e_j$	Local error at j -th PE in s layer.
f	Transfer function.
${}^s I_j$	Weighted summation of inputs to j -th neuron in layer s .
l_{coef}	Learning coefficient or learning rate.
$o_i(t)$	Output value of i -th neuron at time t .
w_{ji}	Weights between j -th neuron and i -th neuron.
${}^{s-1} x_i$	Activation value of i -th PE in level $(s - 1)$.

C.2 Symbols of Flexible Beam Dynamics

l	Beam length.
I_b	Moment of inertia of the beam about the root.
I_h	Moment of inertia at the hub.
I_T	Total moment of inertia, sum of I_b and I_h .
ω	Natural frequency.
ξ	Damping ratio.
$\theta(t)$	Hub angle.
$\omega(x, t)$	Elastic deflection.
$y(x, t)$	Beam deformation.
K	Kinetic energy.
V	Potential energy.
L	Lagrangian.
$q(x)$	Generalized coordinate.
$\phi(l)$	Mode shape of the beam.
$\frac{d\phi_i}{dx}(0)$	Actuator modal gain.