A Neural Network-based Controller for a Single-Link Flexible Manipulator Using the Inverse Dynamics Approach

Zhihong Su

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

August 2000

ABSTRACT

A Neural Network-based Controller for a Single-Link Flexible Manipulator Using the Inverse Dynamics Approach

Zhihong Su

This thesis presents an intelligent strategy for controlling the tip position of a flexible-link manipulator. Motivated by the well-known inverse dynamics control approach for rigid-link manipulators, two multi-layer feedforward neural networks are developed to learn the nonlinearities of the system dynamics. The re-defined output scheme is used by feeding back this output to guarantee the minimum phase behavior of the resulting closed-loop system. No *a priori* knowledge about the nonlinearities of the system is needed where the payload mass is also assumed to be unknown. The weights of the networks are adjusted using a modified on-line error backpropagation algorithm that is based on the propagation of the redefined output error, derivative of this error and the tip deflection of the manipulator. Numerical simulations as well as real-time controller implementation on an experimental setup are carried out. The results achieved by the proposed neural network-based controller are compared in simulations and experimentally with conventional PD-type and inverse dynamics controls to substantiate and demonstrate the advantages and the promising potentials of this scheme.

To my mother, and

In memory of my father and
my first Canadian friend, Mr. Owen Bond

ACKNOWLEDGMENTS

I would like to express my appreciation to all those who have directly and indirectly helped me with my study and research in Concordia.

My heartfelt gratitude first goes to my supervisor, Dr. Khorasani, who introduced me to the control systems area, and guided me through every stage of my study and research. His constant encouragement helped me confidently keep going on my way. Over the years, productive discussions and interactions with him have not only contributed to the understanding of my academic area, but also changed my mentality and my attitude towards what I do, which will be invaluable in challenging my future career and life. I am also very obliged to him for providing support to my study.

The early guidance to my research by Dr. Talebi is gratefully acknowledged. His dedicated help substantially contributed to my research fulfillment.

For the English reviews of my whole thesis, millions of thanks go to Dr. Barbara Deans. She made a considerable effort to improve my written English.

Many of my friends have helped to bring this thesis to life. I can only mention a few of them here. My appreciation goes to Sam, who helped me with the word-processor, LaTeX, to Qian Tao, who gave me a big help with my experimental work, to Sanjeev for proof-reading of a part of this thesis, and to my best friends, Quan and Qingyuan, who have always given me a spiritual support.

But last, and always, I am indebted to my mother, the rest of my family in China, and my canadian friend, Dorothy, for their continuing backup and encouragement during my study. Their ambitions on me have always been my source of strength and motivation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

The outline of the topics addressed in this introductory chapter is as follows. The motivation and objectives of this research, which deals with a flexible-link manipulator, are proposed in *Section 1.1.* The literature review including both theoretical contributions as well as practical applications to flexible manipulators is presented in two sections. *Section 1.2* is mainly concerned with typical conventional control strategies developed in this area in the literature, specifically, joint-based control approach, inverse dynamics control strategies, non-causal controllers, transmission zero assignment, singular perturbation theory and integral manifold strategy, and adaptive control schemes. Neural network-based control strategies are reviewed in *Section 1.3.* Finally, the contributions of this research and the organization of the thesis are presented in *Section 1.4.*

1.1 Motivation and Objectives

Modeling and control of flexible-link manipulators have received a lot of attention in the past few years. The interest in lightweight flexible-link manipulators has increased largely due to their potential benefits to space robotics and industrial applications.

There is an increasing demand imposed by a variety of applications due to higher productivity needs to have manipulators that can operate with higher speed, more precision, less energy consumption, and improved payload handling capabilities. These requirements translate into manipulators that have structural flexibility and are lightweight. For instance, a manipulator that is cleaning a delicate surface [1] needs to have significant structural flexibility so that errors in position control do not generate large forces that may damage the surface.

The space shuttle arm possesses a very long and slender geometric structure, which contributes to certain mechanical vibrations. In general, flexible arms need to be much lighter in weight compared to rigid robots in order to achieve a quick response with lower energy requirements, and at the same time ensure precise motion and output tracking.

Satisfying the above demanding requirements for flexible manipulators is a very challenging problem in this area. From the control point of view, the inherent structural flexibility in a manipulator makes it impossible to have an exact finite-dimensional model of the system due to the distributed nature of the dynamics of the arm. Secondly, since the sensor and the actuator (input and output signals) are non-colocated, taking the tip position as an output feedback control will result in a

non-minimum phase dynamic representation. In other words, the open-loop transfer function of the manipulator from joint torque to tip position will have zeros that are located in the right half s-plane. This property can impose limitations on the controller design as far as closed-loop stability margins are concerned. Thirdly, due to the infinite-dimensional characteristics of the arm, the control system is underactuated, which imposes severe limitations on what the control can achieve. And, finally, the flexible manipulator is characterized by a complicated set of nonlinear dynamical equations as well as inherent unmodeled dynamics and unstructured uncertainties, which add further limitations to the control capabilities.

To address the above challenges as far as the control design is concerned, the classical frameworks of neither adaptive nor non-adaptive control schemes may be adequate. If high performance requirements are desired, necessarily, an accurate dynamic model should be available to design an appropriate controller. However, as discussed above, an accurate model is not easy to obtain for a flexible-link manipulator. In attempts to find a solution to these problems, much research has been directed toward intelligent-based approaches.

An understanding of the biological mechanism of the human nervous system and its structure has been influential in the design of artificial neural networks and their wide applications. The abilities of these artificial neural networks in parallel processing, learning, nonlinear mapping and generalization have motivated extensive research and development in their use as a powerful tool for identification and control of dynamic systems. It would appear that artificial neural networks with their ability to learn complex mappings show promise in providing better solutions for tracking

control of flexible manipulators.

The objective of the present research is to study more advanced control techniques for a flexible-link manipulator in order to achieve a higher performance both in simulation and experimentation. Specifically, a novel neural network-based control configuration is developed for precise tip position tracking control of a single-link flexible manipulator. The dynamic features of neural networks combined with a simple conventional linear controller would allow the following design goals for this dynamic system to be attained,

- Good transient and steady-state tracking of desired motion trajectory,
- Suppression of unwanted vibrations,
- High-speed and precision manipulation relative to structural flexibility, and
- Good performance and stability robustness against unknown task condition variations.

1.2 Review of Conventional Control Schemes

Considerable research has been conducted on the subject of the modeling and control of flexible-link manipulators. Much of the research has involved nonlinear conventional control strategies which are well developed for rigid robots. A review of the research in which nonlinear conventional control strategies have been applied to a flexible-link manipulator follows.

Without taking into consideration the flexibility of the manipulator as feedback, a *joint-based strategy* was experimentally implemented by De Luca, *et al* [2] on

a two-link robot with a flexible forearm. The control algorithms were presented as a linear feedback plus a model-based feedforward term. The aim was to compensate nominally, nonlinearities and interactions of the dynamic model in order to improve the tracking accuracy. Cetinkunt and Book [1] applied adaptive model following control (AMFC) to a structurally flexible manipulator assuming that the Coriolis and centrifugal nonlinear forces were negligible. Only the joint variable feedback was taken into the control algorithm to avoid the closed-loop non-minimum phase problem. The performance limitations due to the joint variable as feedback was also discussed. The adaptation algorithm provided good tracking of joint variables against large nonlinear forces only by increasing the feedback gains. This resulted in very stiff joints and persistent structural vibrations. If joint position control was too stiff relative to the arm flexibility, it was not possible to provide well-damped dominant modes, no matter how large the velocity feedback gain was selected. As a result, vibrations of the structural flexibility took a longer time to die out so that the performance of the tip position was degraded, especially during transient. It should be noted that when only the joint variables were used as feedback, the dynamic nonlinearities were dominant as the speed of a manipulator motion increased. Since there was no direct control for the tip position, the structural flexibility in the response of the manipulator became very apparent.

Among the various control strategies developed for tracking control of a rigid manipulator, the *inverse dynamics control scheme* is by far the most common nonlinear scheme found in the literature. Because this control scheme is relatively straightforward, it has attracted a great deal of the research targeted towards

flexible-link manipulator control. The essential idea is to transform nonlinear system dynamics into a linear one, such that linear control approaches could be applied. However, the non-minimum phase characterics of a flexible-link manipulator has hindered this approach being applied to a direct tip position control, unless the output re-definition approach is utilized. Consequently, it might be possible to apply this "input-redefined output" linearization approach to the tip position tracking control for a class of structurally flexible dynamic systems.

Wang and Vidyasagar [3] developed a transfer function modeling of a single-link flexible manipulator through re-defining output as "reflected" tip position to achieve stable zero dynamics. According to this definition, the output variable was the rigid body deflection minus the elastic deflection, instead of the net tip position. However, the disadvantage of this scheme is that it might not be easy to relate the desired net tip position to the desired reflected tip position. It might also show a significant discrepancy between the net tip position and the reflected tip position in the following situations: (i) the speed of the flexible-link manipulator motion increases substantially or the joint makes a large rotation, which may excite more higher frequency flexible modes, (ii) the manipulator has a structurally higher flexibility. De Luca and Siciliano [4] fully investigated an input-output nonlinear inversion algorithm which ensured exact tracking of any smooth trajectory. Both the joint and an arbitrary point along the link were considered as output signals. In these studies the closed-loop dynamics were always stable when the output was chosen to be the joint angle. The "re-defined output" concept was proposed to allow the possibility for the tracking control of a range of angular outputs defined along

the link. The purpose was to stabilize the zero dynamics of the resulting input-output map. De Luca and Lanari [5] further considered a suitable range of input-output locations along the link, in which it was possible to achieve a minimum phase behavior based on a linear finite-dimensional model of the flexible beam. The other approach discussed for achieving a minimum phase closed-loop behavior kept the output fixed at the tip and let the actuation point vary along the link. This idea may be only of theoretical interest due to the physical difficulty of allocating an actuator along a lightweight flexible link. Madhavan and Singh [6] suggested a joint position added to a scaling of the tip elastic deformation as a "re-defined output". A feasible region of the output along the flexible link was studied to achieve the minimum phase property for a two-link manipulator. Talebi *et al.* [7] advanced the idea that an output re-definition scheme could be used without any *a priori* knowledge of the payload mass. Considering this, it might be possible to design controllers that would remain robust to payload variations. This more general output re-definition concept was developed and implemented on a multi-link flexible manipulator with the input-output linearization technique by Moallem *et al.* [8]. A major difficulty in implementing this type of approach would be its heavy dependence on the accuracy of the inverse model used in the controller, despite the presence of disturbances, unmodeled nonlinearities and uncertainties.

Consideration was even given to the application of *non-causal controllers* to solve the non-minimum phase problem when employing an inverse dynamics control scheme. Bayo and Moulin [9] computed a non-causal torque by integrating the end-point acceleration profile, which acted before the tip started moving and after the tip

stopped moving. Kwon and Book [10] decomposed the inverse dynamics of a flexible manipulator into a causal system, which was integrated forward in time, and an anti-causal system which was integrated backward in time. This was done in order to compute the required torque and the trajectory of flexible mode coordinates in the time domain for a desired tip position trajectory. The preceding two approaches seem to solve the non-minimum phase problem of flexible-link manipulators. However, the substantial amount of computation necessary and the requirements of the linear approximation model of the flexible-link manipulator significantly limit their application.

Influenced by the implementation of *transmission zero assignment* for linear systems by Patel and Misra [11], the principle was experimentally applied to a single-link flexible manipulator by Geniele *et al.* [12] to achieve tip position tracking control. The dynamic model was first linearized about an operating point. Then an inner stabilizing control loop incorporated a feedthrough term to assign the system's transmission zeros at desired locations in the complex plane, and a feedback term to move the system's poles to appropriate positions in the left half s-plane.

Siciliano and Book [13] applied the *singular perturbation theory* to control a lightweight flexible manipulator. The composite control design consisted of two parts: (i) tracking the slow subsystem output and, (ii) stabilizing the fast subsystem around the equilibrium trajectory corresponding to the slow subsystem under the effect of the slow control. The slow control was designed based on the well-established control schemes for rigid manipulators. The *integral manifold strategy* for the control of flexible-joint manipulators was proposed by Khorasani and Spong

[14], and Spong *et al.* [15]. The objective was to obtain a more accurate slow subsystem, which would then facilitate the implementation of a more accurate controller. Hashtrudi-Zaad and Khorasani [16] further developed these methodologies for an approximate model of the Canadarm (shuttle arm) by measuring the hub angle and the tip position to control the arm to an arbitrary degree of accuracy. Moallem *et al.* [17] used a similar approach for a multi-link flexible manipulator that was then verified experimentally for a single-link arm. These control approaches may also be viewed as belonging to a class of output re-definition strategies.

Researchers have also investigated adaptive control schemes because they provide a possibility to update estimation of uncertainties on-line. Due to parametric uncertainties, in most cases of payload variations, nonlinear adaptive control schemes have also been investigated for the flexible-link manipulator. The recursive least square (RLS) algorithm was used by Yurkovich and Pacheco [18] to adjust the parameters of a simple PID controller for a flexible-link manipulator, which was subject to varying, unknown payloads. Lucibello and Bellezza [19] studied a self-tuning adaptive scheme for a two-link flexible robot arm. A least square identification scheme estimated the unknown payload mass on-line. The result was used to tune the computation of a bounded solution for the inverse reduced system and the computation of the nominal torques. The unmodeled dynamics were not considered. A rather simple, finite-dimensional model of the robot was incorporated in the controller design. This was done to keep the algebra as simple as possible and to retain the relevant nonlinear behavior of the system. In their experiments Rokui and Khorasani [20] used an indirect adaptive control scheme for a single-link flexible

manipulator. The research was based on a discrete-time nonlinear model of the system derived by using the forward difference method. This type of control strategy assumed that the payload mass had to enter linearly into the dynamic model of the flexible-link manipulator. Based on this assumption in the newly obtained discrete-time model, a new regressor was proposed. The multi-output recursive-least-square (RLS) algorithm was used to identify the upper bounded unknown payload mass. Siciliano *et al.* [21], and Yuh [22] applied a model reference adaptive control (MRAC) approach to regulate the tip position tracking of a flexible-link manipulator. The former [21] designed a proportional plus derivative (PD) controller so that its gains were adaptively changed using the MRAC algorithm. On the other hand Yuh [22] implemented a discrete-time MRAC to a simplified dynamic model, which was a rigid manipulator plus the disturbance term (process noise). The simulation was carried out under two conditions i.e., a colocated sensor and actuator as well as a non-colocated tip sensor and actuator. For the non-colocated system, the tracking performance degraded with higher frequency modes when the power limitation of the actuator was considered. The MRAC type adaptive control scheme established that a tradeoff between the condition of persistent excitation and closed-loop system stability restrained the achievement of the exact tip position tracking.

From the above survey of typical conventional nonlinear control strategies, it is worth noting that the greatest obstacle for applications of the current modern nonlinear control techniques to flexible-link manipulators is the strong dependence of these techniques on accurate dynamic models. By estimating the parametric and

dynamic uncertainties, the adaptive control schemes reduce the dependence of accurate models to some degree. However the linearities in the parameters require the assumption of a fixed structure, despite the existence of nonlinear uncertainties. Furthermore, in order for the parameters estimates to converge to their true values, the reference trajectory must be "sufficiently rich". In other words, the reference trajectory must sufficiently excite the dynamic modes of the system such that the effects of various parameters can be distinguished. This raises potential problems and limitations for this type of schemes to be implemented on flexible-link manipulators because the unmodeled flexible modes may also be excited that could make the closed-loop system unstable.

1.3 Review of Neural Network-based Control Schemes

The advantages of using an artificial neural network as a viable method to reduce the dependence of a controller on limitations discussed above have been demonstrated in many systems and application areas. In particular, linearity in the parameters is not required and certainty equivalence principle is also not used. This resolves the above limitations for implementing a standard adaptive control scheme. Various configurations are proposed to "intelligently" control flexible-link manipulators. Essentially, in one configuration the information of the dynamic model is partially used, whereas in the other configuration almost no *a priori* knowledge of the system is assumed.

An adaptive type direct neural controller was proposed by Takahashi and Yamada [23], and experimentally tested on a single-link flexible arm for the tip

angular position control. A three layer neural network was trained on-line to learn the inverse dynamics of the model through minimizing a quadratic cost function that took into account both output error and control input. However, when taking the deflection modes directly as feedback into the neural controller in the forward path, no solution for the non-minimum phase problem was presented.

A multi-layer perceptron controller was constructed by Register *et al.* [24] to learn the inverse dynamics model of a single-link manipulator for the tip position control. The simulation model was based on a linearized state-space model that has neglected natural damping. The network was trained by an error-backpropagation algorithm. Better results were obtained through penalizing the hub velocity in a modified cost function. However this approach failed when the link measurement exceeded a certain critical length.

An alternative neural network-based controller was designed taking advantage of the well-established knowledge of the dynamic model. For instance, two multi-layer neural networks were configured by Lin and Yih [25] to cope with *a priori* knowledge of the corresponding model of a rigid manipulator for identification and control of a flexible-link manipulator. The objective was to try to use fewer neurons and a shorter learning time in an effort to reduce the tracking error. Weights of two networks were adjusted off-line after 40 training cycles through error-backpropragation algorithm, and later used as initial weights for on-line adaptation simulations.

Surdhar *et al.* [26] proposed two different neural network-based control structures for a flexible-link manipulator: (i) a radial basis function (RBF) and, (ii)

a multi-layered perceptron (MLP). The networks were both trained off-line using backpropagation algorithm in a supervised fashion and the training data was copied from an existing fuzzy PD controller. A discussion on the difference due to the generalization properties between the two kinds of networks was also presented, i.e., the RBF generalized the data locally and the MLP had global generalization. Results showed that the MLP network was less sensitive to the loading of the arm than the RBF network.

An error-backpropagation neural network was used in Zeman *et al.* [27] to model the inverse dynamics of a flexible-joint manipulator. Given the difficulties in determining the training signal and a "sufficiently rich/persistently exciting" training set, the network was initially trained off-line by performing a generalized learning to establish an accurate inverse dynamics model. Then the system was operated under specialized learning with a slower learning rate to compensate for variations in the plant dynamics.

The problem of how to obtain an error signal for training a neural controller has been an issue for researchers for a long time. Miyamoto *et al.* [28] dealt with this essential problem by developing the concept of *feedback-error-learning.* It was pointed out later by Gomi and Kawato [29] that "in supervised learning, the error for a neural controller adaptation should not be the trajectory-error, but the command-error". Using this method, the neural network model for a feedforward control acquires the inverse dynamics model of a control object. The benefit of this learning strategy is that the "target signal" for the output of the network, which is impossible to determine, is not required.

The feedback-error-learning was conceptually adopted by Newton and Xu [30] for control of a flexible manipulator. In [30], a recurrent three-layer perceptron neural network controller was designed for a space flexible manipulator. The small gain constants were used for implementing experimentally a low-pass filter to prevent excitation of the vibration modes. However, to bypass the non-minimum phase problem, no strain or tip measurements were used, instead only joint variables were used as feedback for the closed-loop control system. These experiments may be viewed as an extension of a rigid manipulator case.

Later, feedback-error-learning and re-defined output concepts were developed further by Talebi *et al.* [7] to design neural network-based controllers for flexible-link manipulators. Implementations were conducted in simulation for a two-link manipulator with a flexible forearm and experimentally for a single-link flexible manipulator. Four different architectures were proposed, and all of them were trained on-line to learn the inverse dynamics model of the manipulator to be represented by a neural network. In this way the tip position tracking control was achieved. The first two schemes were developed to learn the inverse dynamics of the flexible-link manipulator. Different information was taken from the system outputs and desired reference signals as inputs to the corresponding neural network. These two schemes needed conventional linear controllers in the forward path to help stabilize the closed-loop system and accomplish the control objectives. A third scheme was based on tracking the joint position, simultaneously penalizing the tip deflection by taking it into the cost function. The fourth scheme employed two neural networks which were applied on-line to learn the re-defined output and the

inverse dynamics controller respectively, without any *a priori* knowledge of the dynamics model. Promising results were obtained in comparison to those obtained from PD-type control schemes.

1.4 Synopsis and Contributions of the Thesis

The aim of the research presented in this thesis is to implement a proposed neural network-based control strategy on a single-link flexible manipulator both in simulation and experimentation. From the literature review, the advantages of utilizing the neural network-based schemes in motion control of a flexible manipulator have been established. Motivated from the work of Talebi *et al.* [7], the first two schemes in which partial knowledge of the model was incorporated in the design gave satisfactory results. Less favorable results were obtained with the other two schemes in which the assumption of some *a priori* knowledge of the dynamic model was diminished. Thus, it seems reasonable as well as preferable to take the available knowledge of the dynamic model as a basis in designing a neural network-based controller.

Bearing this fact in mind, in this research instead of treating the whole inverse dynamics model of a single-link flexible manipulator as a "black box", advantage was taken of the existing knowledge of the structure of the inverse dynamics model in designing the proposed neural network-based controller. It will be shown experimentally as well as in simulation results that when two neural networks are configured to couple with a conventional linear controller, more accurate tip position tracking of a single-link flexible manipulator may be achieved.

There are 6 chapters in this thesis. *Chapter 2* presents the mathematical

dynamic model of the experimental single-link flexible manipulator used for simulations. The representation of an existing physical model is derived following the recursive Lagrangian approach using the Euler-Bernoulli beam theory and the assumed modes method (Geniele [31]).

For the purpose of comparison with the neural network-based scheme, two conventional control schemes, namely the PD-type, and the inverse dynamics controllers are reviewed in *Chapter 3*. The mathematical models are developed and the design scenarios are reviewed briefly with respect to a specific single-link flexible manipulator. Special emphasis is devoted to employing neural networks in the motion control of a single-link flexible manipulator. Initially, some theoretical background is provided concerning the underlying principles of neural networks. For instance, the philosophy of neural networks and their mathematical basis are addressed, mainly emphasizing multi-layer perceptron architectures and the related training algorithm, namely error-backpropagation. Next the function approximation basis of neural networks is presented. Finally the proposed neural network-based control strategy developed in this research is presented.

Numerical simulations are outlined in *Chapter 4* based on the mathematical dynamic model of a single-link flexible manipulator derived in Chapter 2. The neural network configurations and their related design considerations are presented. Results and performances from simulations are discussed and compared with conventional control schemes.

In *Chapter 5* the experimental implementations of the proposed neural

network-based control strategy on a single-link flexible manipulator test-bed (Geniele [31]) are demonstrated. The hardware configurations and the software components of this real-time system are introduced. Considerations related to the experimental implementation of the proposed control scheme are also addressed. Finally, experimental results are presented, along with discussions and comparisons with those obtained from conventional control schemes.

The concluding remarks of this research, and proposals for the future work in neural network-based control for the flexible-link manipulator are provided in *Chapter 6.*

Chapter 2

Dynamic Model of a Single-Link Flexible Manipulator

In this chapter the mathematical model of an experimental single-link flexible manipulator is reviewed. The mathematical formulae will be utilized for numerical simulation studies. In *Section 2.1*, several approaches in the literature for modeling the flexible-link manipulator are discussed. The characteristics of a physical model for a planer single-link flexible manipulator is described in *Section 2.2*. In particular, the derivations for the dynamic model are given in *Section 2.3* by utilizing the energy-based Lagrangian formulation and the Euler-Bernoulli beam theory as well as the assumed modes method.

2.1 Introduction

An accurate dynamic model plays an important role in the design of an advanced model-based nonlinear controller, as well as for the numerical simulation purposes. The model should possess the most relevant properties of the system, should provide a clear understanding of the dynamic interaction and couplings, and finally should be useful for control design. The flexible-link manipulator is a highly nonlinear, distributed parameter system consisting of a rigid body motion with the ability to make elastic vibrations. With these considerations, as pointed out by Tokhi and Azad [32], several approaches have been used in deriving the dynamic model of flexible-link manipulators for obtaining the mapping between the force exerted on the joint and the positions, velocities and accelerations of the joint and the tip. These approaches include the Lagrangian formulation and assumed modes method, the Lagrangian formulation and finite element method, the Newton-Euler equations and assumed modes method, and the Hamilton's principle and finite element technique.

The Lagrangian formulation is simple and systematic. In the assumed modes method, the deflection of a flexible-link manipulator is represented as a sum of modes. Each mode is assumed to be a product of the distance along the length of the manipulator and a generalized coordinate which depends on time. An infinite number of modes are truncated in order to synthesize a finite-dimensional dynamic model for practical purposes. The work done by Book [33], Cannon and Schmitz [34], Wang and Vidyasagar [3], Cetinkunt and Yu [35], and Geniele [31] fell into this category. The finite element approach is conceptually similar to the assumed modes method except that the generalized coordinates are the displacement or slope at

specific points along the manipulator. Usoro *et al.* [36] investigated this approach in their research.

The Newton-Euler equations and assumed modes method is a direct and an efficient method for the numerical evaluation of the system dynamics. Newton's second law is used to balance the rate changes of linear and angular momentum with the applied forces. The basic principle is to divide the manipulator into a number of elements. Raksha and Goldenberg [37] developed this strategy for modeling a single-link flexible robot.

The major advantage of the Hamilton's principle and a finite element approach is that different material properties and boundary conditions like hubs, tip load and changes in cross-section can be handled in a simple manner. Bayo [38], and Tokhi *et al.* [39] used this concept in their work.

The necessity for computing the internal forces to arrive at the generalized forces presents difficulties when the Newton-Euler formulation is used. However, for control purposes, it is not necessary to calculate these internal forces. The Lagrangian formulation directly computes the generalized forces in a straightforward manner using the Lagrangian function. In this thesis, the Lagrangian formulation and assumed modes method as discussed in Geniele [31] for modeling a planar single-link flexible manipulator is adopted. The re-defined output concept of [28] and [6] has been used here to facilitate the design of the feedback controllers.

2.2 Characterestics of the Physical Arm

The schematic of a planar single-link flexible manipulator is shown in Figure 2.1. (X_0, Y_0) is an inertial coordinate frame, and (X_1, Y_1) is the coordinate assigned for a flexible link. θ, $v(x,t)$ and τ represent the hub position, the deflection along the arm, and the torque applied to the hub, respectively.

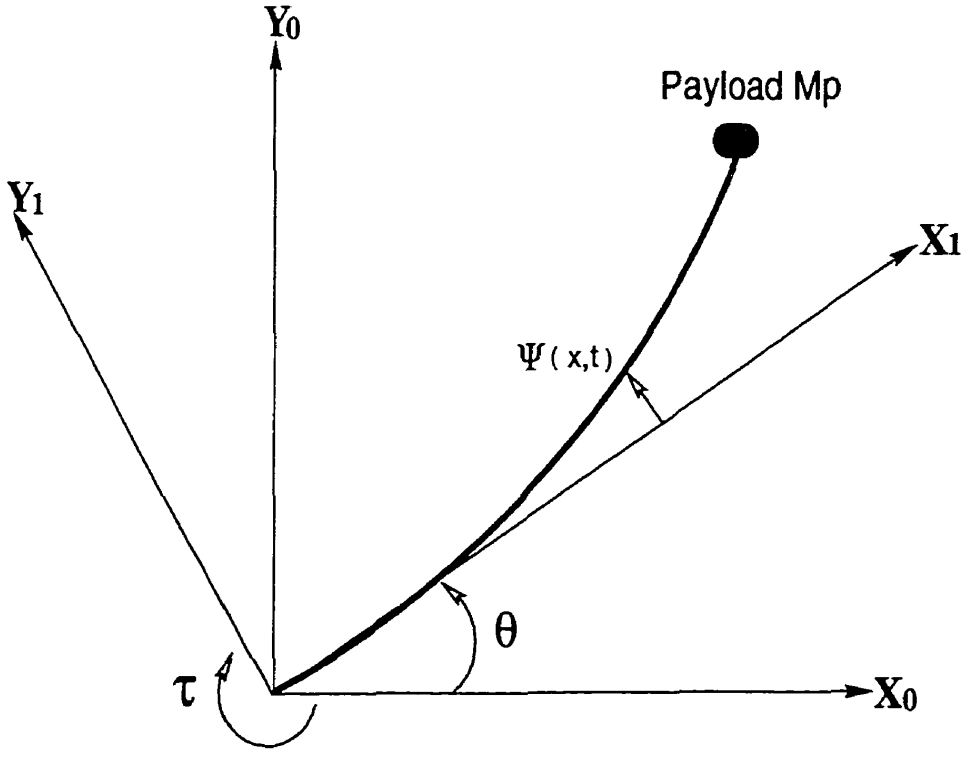

FIGURE 2.1. A Planar Single-Link Flexible Manipulator

The existing experimental single-link flexible manipulator is a 1.2-m-long, very flexible structure that can bend freely in the horizontal plane but not in the vertical plane. At one end, the arm is clamped on a rigid hub mounted directly on the vertical shaft of a DC motor. A torque applied by the DC motor rotates the

arm in a horizontal plane. The other end of the arm with payload mass attached is free. The beam of the manipulator consists of a central stainless steel tube with annular surface corrugations. Aluminum blocks are bolted to the tube and two thin parallel steel spring strips slide within slots cut into the blocks.

2.3 Dynamic Modeling

The dynamic model of the single-link flexible manipulator is derived here by first utilizing the Euler-Bernoulli beam theory to obtain a partial differential equation (PDE) with the corresponding boundary conditions representing the motion of the manipulator. Then, from the system energy point of view, the Lagrangian formulation approach along with the assumed modes method results in a state-space representation of the dynamics of the flexible manipulator.

In order to derive a mathematical model, the following assumptions have to be made,

- Rotary inertia and deformation of the beam due to shear forces may be neglected because the cantilever beam assumes that the cross-sectional area of the link is small in comparison with its length h,
- Beam inertia and flexibility are uniformly distributed over the link length, assuming that EI is a constant, and
- The motion of the flexible link is limited to small displacements.

2.3.1 Euler-Bernoulli Equation

By applying Euler-Bernoulli beam theory, the dynamic equation of motion of the flexible-link manipulator may be obtained. The distributed nature of the system is evident in the presence of a fourth-order partial differential equation (PDE),

$$\frac{EI}{\gamma}\frac{\partial^4\psi(x,t)}{\partial x^4}+\frac{\partial^2\psi(x,t)}{\partial t^2}=0 \tag{2.3.1}$$

where x is the normalized position along the link of length h, I is the link cross-sectional moment of inertia, E is the Young's modulus of the material, γ is the mass per unit length, and $\psi(x,t)$ is the deflection of a point located at a distance x along the beam.

With the cantilever beam assumption, the following boundary conditions are used to solve the above PDE,

$$\text{at} \quad x=0 \qquad \begin{cases} \psi=0 \\ \dfrac{d\psi}{dx}=0 \end{cases}$$

$$\text{at} \quad x=h \qquad \begin{cases} B=EI\dfrac{d^2\psi}{dx^2}=0 \\ S=EI\dfrac{d^3\psi}{dx^3}=M_p\dfrac{\partial^2\psi(h,t)}{\partial t^2} \end{cases}$$

where M_p is the payload mass, B and S represent bending moment and shear force, respectively.

The assumed form for the solution of (2.3.1) is arrived at by solving an infinite set of eigenvalues and corresponding eigenfunctions,

$$\psi(x,t) = \sum_{i=1}^{\infty} \phi_i(x) q_i(t) \tag{2.3.2}$$

where $q_i(t)$ is the generalized coordinate of the ith mode of the link, and $\phi_i(x)$ is the normalized, clamped-free eigenfunction of the ith mode [3], which is given by

$$\phi_i(x) = d_i\Big(\sin k_i x - \sinh k_i x - \frac{\sin k_i h + \sinh k_i h}{\cos k_i h + \cosh k_i h}\big(\cos k_i x - \cosh k_i x\big)\Big) \tag{2.3.3}$$

with k_i determined from the following transcendental frequency equation,

$$\frac{M_p k_i}{\gamma}\big(\sin k_i h \cosh k_i h - \cos k_i h \sinh k_i h\big) - \cosh k_i h \cos k_i h - 1 = 0 \tag{2.3.4}$$

with $\phi_i(x)$ solved by choosing d_i to satisfy $\int_0^h \phi_i^2(x)dx = 1$.

2.3.2 Lagrangian Approach

The energy-based Lagrangian formulation of the motion is oriented towards the analytical computation of the manipulator dynamics. It gives more insight into the sensitivity of the model to the changes in parameter values.

The mathematical dynamic model is developed to reveal the dynamic behavior of the system using the Lagrangian approach which is defined as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \qquad i = 1, \ldots, n \tag{2.3.5}$$

where

$$L = K - U \tag{2.3.6}$$

is the Lagrangian expressed as the difference between the kinetic energy K and the elastic potential energy U of the total system. τ_i is the generalized force at the joint i.

The mathematical equation (2.3.2) indicates that an exact solution to the Euler-Bernoulli partial derivative equation (PDE) requires an infinite number of modes. A truncated model is obtained using the assumed modes shape with a finite number of modes/eigenvalues to approximate the exact solution,

$$\psi(x,t) = \sum_{i=1}^{N} \phi_i(x) q_i(t) \tag{2.3.7}$$

As a result, a mathematical representation for the dynamic model of the single-link flexible manipulator may be derived (Geniele [31]) in the state-space form as follows,

$$\boldsymbol{M}(q) \begin{bmatrix} \ddot{\theta} \\ \\ \ddot{q} \end{bmatrix} + \boldsymbol{C}(q,\dot{q}) \begin{bmatrix} \dot{\theta} \\ \\ \dot{q} \end{bmatrix} + \boldsymbol{K}(\dot{\theta}) \begin{bmatrix} \theta \\ \\ q \end{bmatrix} + \begin{bmatrix} Fric(\dot{\theta}) \\ \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{u(t)} \\ \\ 0 \end{bmatrix} \tag{2.3.8}$$

with the re-defined output expressed as

$$y_a = \theta + \frac{\alpha}{h} \Phi_{1\times N} q_{N\times 1} \tag{2.3.9}$$

where

$$\begin{aligned} \Phi_{1\times N} &= \begin{bmatrix} \phi_1(h) & \phi_2(h) & \dots & \phi_N(h) \end{bmatrix} \\ q^T_{N\times 1} &= \begin{bmatrix} q_1 & q_2 & \dots & q_N \end{bmatrix} \end{aligned}$$

and q is the elastic variable, θ is the joint variable, h is the length of the link, and $u(t)$ is an input torque applied to the hub, α is a parameter which varies between

$[-1\ 1]$ with $\alpha = 1, 0, -1$ corresponding to the tip position, the joint angle, and the reflected tip position, respectively. For the re-defined output case, α is chosen between [0 1].

The matrix $M(q)$ is the positive-definite symmetric inertia matrix with the definition,

$$\boldsymbol{M}(q) = \begin{bmatrix} m_1 & M_2^T \\ M_2 & \boldsymbol{M_3} \end{bmatrix} \tag{2.3.10}$$

where

$$m_1 = I_h + \frac{\gamma h^3}{3} + M_p\left(h^2 + \left(\sum_{i=1}^{N}\phi_i(h)q_i\right)^2\right)$$

$$\begin{aligned} M_2^T \;=\; & \left[\gamma\int_0^h \phi_1 x dx + M_p h\phi_1(h) \quad \gamma\int_0^h \phi_2 x dx + M_p h\phi_2(h) \quad \ldots \right. \\ & \left. \gamma\int_0^h \phi_N x dx + M_p h\phi_N(h)\right] \end{aligned}$$

$$\boldsymbol{M_3} = \begin{bmatrix} \gamma + M_p\phi_1^2(h) & M_p\phi_1(h)\phi_2(h) & \ldots & M_p\phi_1(h)\phi_N(h) \\ M_p\phi_2(h)\phi_1(h) & \gamma + M_p\phi_2^2(h) & \ldots & M_p\phi_2(h)\phi_N(h) \\ \vdots & \vdots & \ddots & \vdots \\ M_p\phi_N(h)\phi_1(h) & M_p\phi_N(h)\phi_2(h) & \ldots & \gamma + M_p\phi_N^2(h) \end{bmatrix}$$

The matrix $C(q, \dot{q})$ which is the Coriolis and viscous damping matrix is defined as

$$\boldsymbol{C}(q, \dot{q}) = \begin{bmatrix} C_1 & 0 \\ 0 & \boldsymbol{C_2} \end{bmatrix} \tag{2.3.11}$$

where

$$C_1 = b + 2\gamma \sum_{i=1}^{N} q_i \dot{q}_i + M_p \sum_{k=1}^{N} \sum_{m=1}^{N} \phi_k \phi_m (\dot{q}_k q_m + q_k \dot{q}_m)$$

$$\boldsymbol{C_2} = \begin{bmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_N \end{bmatrix}$$

The term $K(\dot{\theta})$ may be defined as

$$\boldsymbol{K}(\dot{\theta}) = \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{K_1} \end{bmatrix} - \dot{\theta}^2 \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{K_2} \end{bmatrix} \tag{2.3.12}$$

where

$$\boldsymbol{K_1} = \gamma \begin{bmatrix} \omega_1^2 & 0 & \dots & 0 \\ 0 & \omega_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \omega_N^2 \end{bmatrix}$$

and

$$\boldsymbol{K_2} = \begin{bmatrix} \gamma + M_p \phi_1^2(h) & M_p \phi_1(h) \phi_2(h) & \dots & M_p \phi_1(h) \phi_N(h) \\ M_p \phi_2(h) \phi_1(h) & \gamma + M_p \phi_2^2(h) & \dots & M_p \phi_2(h) \phi_N(h) \\ \vdots & \vdots & \ddots & \vdots \\ M_p \phi_N(h) \phi_1(h) & M_p \phi_N(h) \phi_2(h) & \dots & \gamma + M_p \phi_N^2(h) \end{bmatrix}$$

and the combination of $\dot{\theta}^2 q K_2$ yields the vector of centrifugal forces.

The term $Fric(\dot{\theta})$ represents Coulomb friction at the hub, which is modeled as $C_{coul}sgn(\dot{\theta})$ with C_{coul} as a coefficient of the Coulomb friction.

The design of a controller for a flexible-link manipulator requires the knowledge of the parameters presented in the above dynamic model. The specific values for these parameters corresponding to our experimental single-link flexible manipulator may be found in Geniele [31]. They were experimentally obtained through on-line system identification techniques.

Chapter 3

Control Schemes

As proposed in Chapter 1, the major objectives for control design of a flexible-link manipulator are to have an accurate and fast tip position tracking, and simultaneously to ensure that arm deflections are well suppressed. In order to reach these goals, a novel *neural network-based control* scheme is proposed in this chapter. In *Sections 3.1* and *3.2* basic concepts and mathematical fundamentals of conventional control schemes are reviewed. These control schemes, namely *PD-type control* and *inverse dynamics control*, will be compared with the proposed neural network-based control. The design for the neural network-based control scheme is presented in *Section 3.3*, which focuses specifically on certain mathematical preliminary, theoretical foundations of function approximation, and control design considerations.

3.1 PD-type Control Schemes

Fundamentally, a PD-type (Proportional and Derivative) controller is designed to tune gains with a position and a velocity feedback, and to meet transient and steady-state specifications of a closed-loop system. Its mathematical representation is as follows,

$$u = K_d(\dot{y}_r - \dot{y}_a) - K_p(y_r - y_a) \tag{3.1.1}$$

where y_r is the tip position reference trajectory, y_a is the re-defined output, and K_d and K_p are gains to be tuned with respect to proportional and derivative terms, respectively.

From (3.1.1), an advantage of not requiring knowledge of either a model structure (2.3.8) or model parameters is noticeable. However, a PD-type control scheme shows its inadequacy in robustness, optimization, as well as handling non-linearities. These limitations will be made more clear later, when this control scheme is implemented on a single-link flexible manipulator, which is a highly nonlinear, and a complicated dynamic system.

3.2 Inverse Dynamics Control Schemes

3.2.1 Mathematical Fundamentals and Error Dynamics

The development of an inverse dynamics control law is approached by first defining an inverse of the inertia matrix (2.3.10) as

$$\boldsymbol{M}^{-1}(q) = \begin{bmatrix} m_{11} & M_{12}^T \\ M_{12} & M_{22} \end{bmatrix} \tag{3.2.1}$$

Dynamic equations of the single-link flexible manipulator (2.3.8) may now be re-arranged as follows,

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{q} \end{bmatrix} = \boldsymbol{M}^{-1}(q)\left(\begin{bmatrix} \boldsymbol{u}(t) \\ 0 \end{bmatrix} - \boldsymbol{C} \begin{bmatrix} \dot{\theta} \\ \dot{q} \end{bmatrix} - \boldsymbol{K} \begin{bmatrix} \theta \\ q \end{bmatrix} - \begin{bmatrix} Fric(\dot{\theta}) \\ 0 \end{bmatrix} \right) \tag{3.2.2}$$

In order to obtain the inverse dynamics of the system, a successive derivative of the re-defined output y_a (2.3.9) with respect to time t is taken, until an input $u(t)$ appears. In this case, two differentiations of y_a are required, such that

$$\ddot{y}_a = \ddot{\theta} + \frac{\alpha}{h}\Phi\ddot{q} \tag{3.2.3}$$

When combined with (3.2.2), (3.2.3) becomes

$$\begin{aligned} \ddot{y}_a \;=\; & \boldsymbol{u}(t)(m_{11} + \frac{\alpha}{h}\Phi M_{12}) - (m_{11} + \frac{\alpha}{h}\Phi M_{12})(C_1\dot{\theta} + Fric) \\ & - (M_{12}^T + \frac{\alpha}{h}\Phi M_{22})(C_2\dot{q} + K_1 q - K_2\dot{\theta}^2 q) \end{aligned} \tag{3.2.4}$$

By some mathematical manipulations of (3.2.4), the inverse dynamics control law may now be defined according to

$$\boldsymbol{u} = \boldsymbol{G}(q)\boldsymbol{V} + \boldsymbol{F}(\theta, \dot{\theta}, q, \dot{q}) \tag{3.2.5}$$

where V is a new input to be specified subsequently, and

$$\boldsymbol{G}(q) = \frac{1}{m_{11} + \frac{\alpha}{h}\Phi M_{12}}$$

$$\boldsymbol{F}(\theta, \dot{\theta}, q, \dot{q}) = (C_1\dot{\theta} + Fric) + \frac{\left(M_{12}^T + \frac{\alpha}{h}\Phi M_{22}\right)\left(C_2\dot{q} + qK_1 - q\dot{\theta}^2 K_2\right)}{m_{11} + \frac{\alpha}{h}\Phi M_{12}}$$

Substituting (3.2.5) in (3.2.4), the equivalent dynamic model becomes

$$\ddot{y}_a = \boldsymbol{V} \tag{3.2.6}$$

which is an input-output (re-defined output) linearized system with respect to the new input to the system. It should be noted that there are several ways for selecting $V(\cdot)$. In general, it may be chosen as

$$\boldsymbol{V} = \ddot{y}_r + K_d(\dot{y}_r - \dot{y}_a) + K_p(y_r - y_a) \tag{3.2.7}$$

Applying (3.2.7) to the input-output linearized representation of (3.2.6), the closed-loop error dynamics is now governed by

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \tag{3.2.8}$$

where the error e is defined as $e = y_r - y_a$. This error will be used subsequently for the weights tuning algorithm of the neural networks.

3.2.2 Design Strategies

As pointed out earlier, the inverse dynamics technique, also known as feedback linearization, has been successfully applied to rigid manipulators. This may be due to its straightforward design procedure. The essential idea of this approach is to

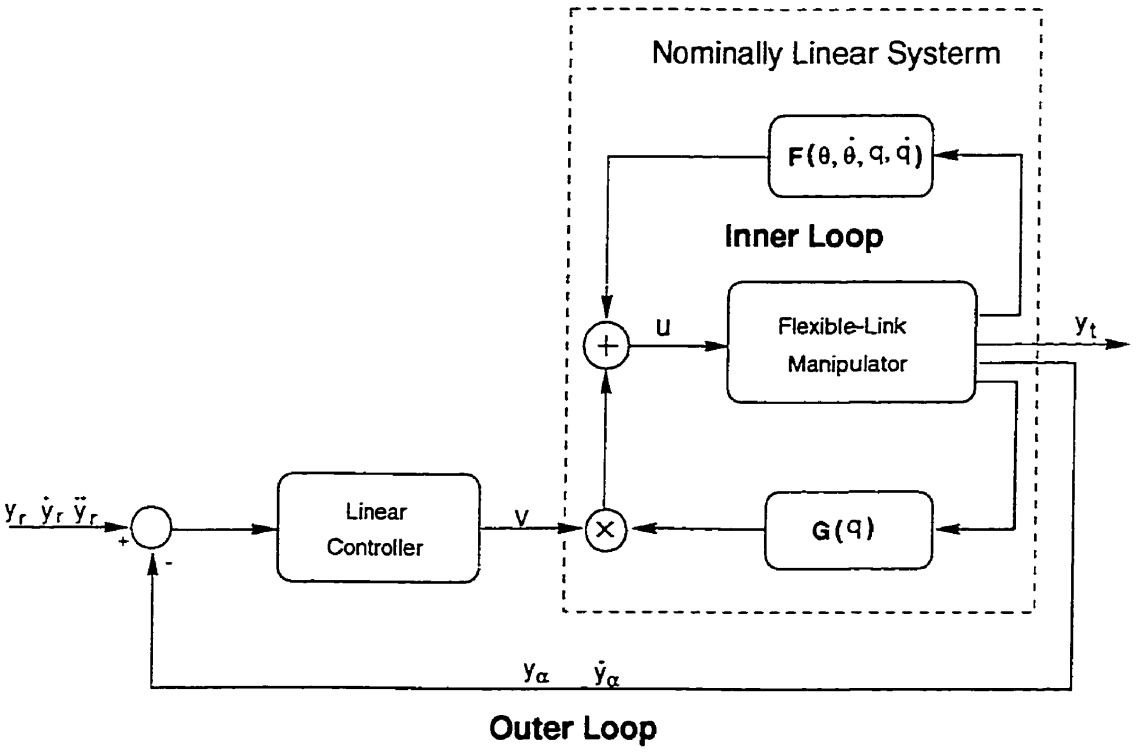

FIGURE 3.1. Inverse Dynamics Control Strategy for a Flexible-Link Manipulator: Inner Loop/Outer Loop Control Architecture

transform algebraically a nonlinear system dynamics into a linear one, so that linear control techniques can then be easily applied. This is to be realized by exact state-space coordinate transformations and feedback, rather than by a linear approximation of the dynamic model.

From the preceding mathematical development, a rather remarkable result is achieved, such that the "new" system (3.2.6) is both linear and decoupled. It implies that an input $V(\cdot)$ can be designed to control a linear system. Moreover, if it is assumed that $V(\cdot)$ is a function of only y_a and its derivatives, then $V(\cdot)$ will affect y_a independent of the link motions. In other words, the highly nonlinear and coupled dynamics of the manipulator are cancelled and replaced by a simple decoupled linear second-order system. This concept is depicted in Figure 3.1. However, the exact cancellation of nonlinearities leaves the door open to many questions on issues such as sensitivity and robustness.

An important issue which has to be addressed when an inverse dynamics scheme is applied to a flexible-link manipulator is the problem of stability. Generally, with an inverse dynamics controller, an original nth order system is divided into two parts. One part is observed from an output, and its derivative with a relative degree of "r". The other part which is not observed has an order of "$n - r$". Since the design of the controller is based on the observed part only, the stability of the whole system depends on the stability of the unobserved part which is also known as *internal dynamics*. A simplified version of the internal dynamics is referred to as *zero-dynamics*.

The zero-dynamics are dynamics left in the system when the input is chosen

in such a way that it forces the output to remain exactly at zero. If the zero-dynamics are stable, the closed-loop system using inverse dynamics may be designed to be stable. On the other hand, the system is internally unstable, and the inverse dynamics scheme can not be implemented for this type of system which is known as a *non-minimum phase.*

The tip position tracking control of a flexible-link manipulator has a non-minimum phase nature, due to the actuator-sensor non-colocated configuration. In other words, an open-loop transfer function from a joint torque to a tip position has zeros that are located in the right half s-plane. Therefore, a closed-loop system under non-colocated control is only conditionally stable. The re-defined output concept ([4], [5], [6]) is adopted in this research to alleviate this specific problem.

3.3 Neural Network-based Control Schemes

3.3.1 Introduction

An artificial neural network is essentially a network of suitably inter-connected non-linear elements of very simple forms that possesses an ability for learning and adaptation. It is characterized in principle by a network topology, a connection pattern, neural activation properties, train strategy, and an ability to process information.

Neural networks have emerged as very powerful tools for modeling nonlinear dynamic systems and for designing intelligent control systems. Increasingly sophisticated tasks required of flexible-link manipulators demand better control techniques to achieve higher system performance.

Next some mathematical preliminaries for neural networks are introduced. The capabilities of neural networks for nonlinear functional approximation are reviewed from a theoretical and a fundamental point of view. Finally, a neural network-based control configuration for the single-link flexible manipulator is proposed.

3.3.2 Mathematical Preliminaries

An artificial neural network is made up of densely parallel layers of simple nonlinear synaptic neurons/units. They are interconnected with varying weights that completely specify the behavior of the network once it has been trained. The neurons of a network are arranged in such a manner as to process information in a parallel manner and simultaneously. Each neuron sends activated signals to other neurons while its activation depends on signals received from other neurons to which it is connected. Suitable interconnection of these simple elements can yield powerful networks with an ability to learn, adapt and infer.

Multi-Layer Perceptrons

A multi-layer perceptron is a feedforward neural network consisting of a number of units (neurons). The perceptron is directly applicable as a nonlinear controller as a consequence of its ability to produce any arbitrary input-output mappings connected by weighted links. Typically units are organized in several layers, namely an input layer, one or more hidden layers, and an output layer. The input layer receives an external signal, and passes it via weighted connections to units in the first hidden

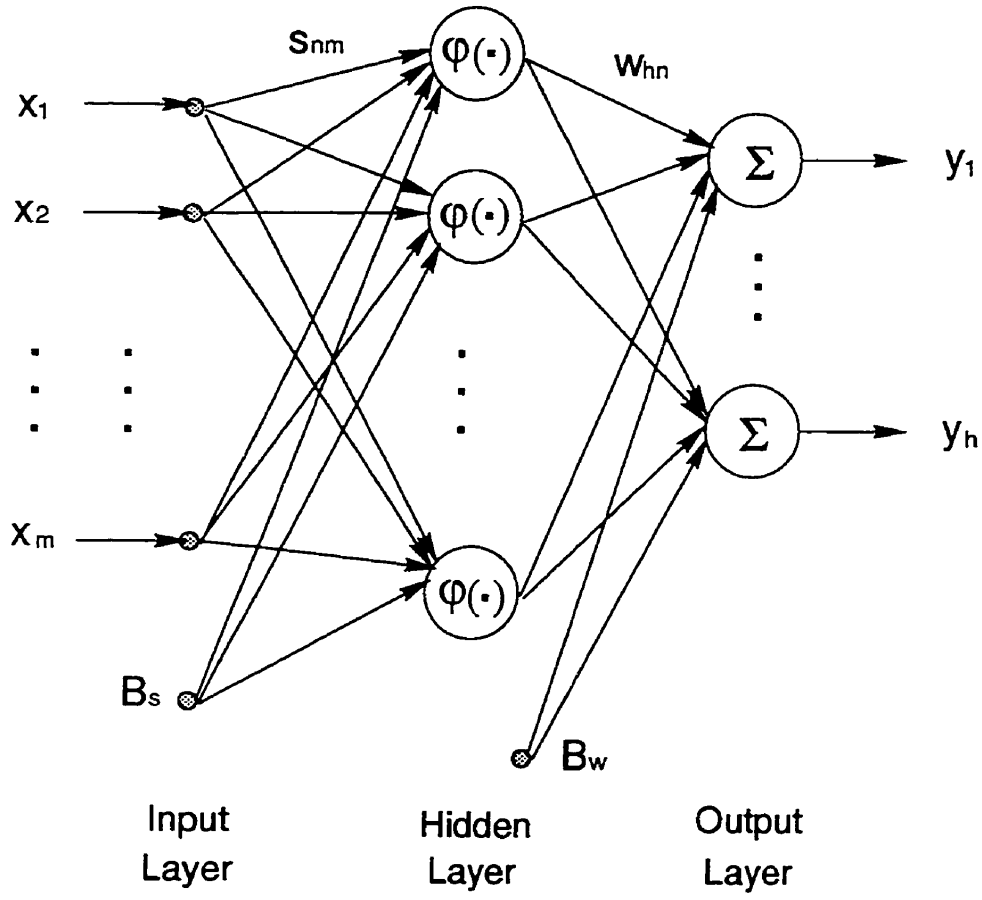

FIGURE 3.2. A Multi-Layer Perceptron with One Hidden Layer

layer. These units compute their activations and pass them to neurons in succeeding layers until the output layer is reached. Figure 3.2 illustrates a typical feedforward network with one hidden layer.

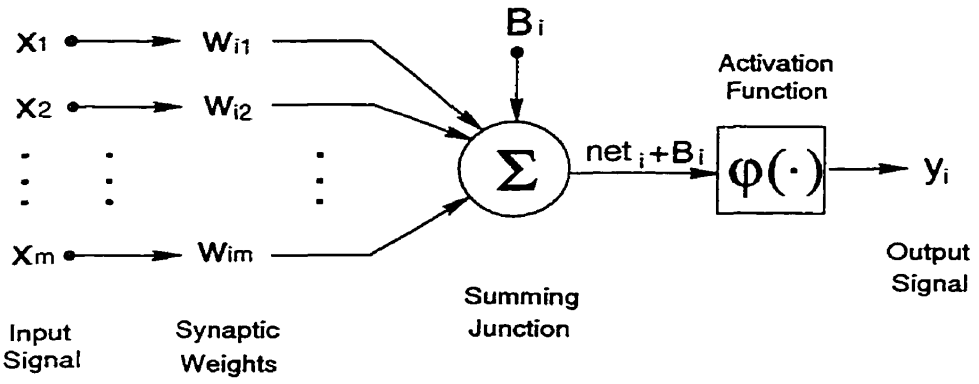

FIGURE 3.3. A Nonlinear Model of a Neuron

A nonlinear model of a neuron is shown in Figure 3.3. Each neuron i in a network is a simple processing unit that has a linear combiner/summing junction, whose output net_i is a weighted sum of its inputs x_j, that is

$$net_i = \sum_{j=1}^{m} w_{ij} x_j \tag{3.3.1}$$

An output y_i of a unit i is computed by passing the net_i input through a nonlinear activation function $\varphi(\cdot)$,

$$y_i = \varphi(net_i + B_i) \tag{3.3.2}$$

where w_{i1}, w_{i2}, . . ., w_{im} are synaptic weights of neuron i, and $\varphi(\cdot)$ is an activation function. B_i is the bias which has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively. It can be treated like a connection weight from an input.

The output y of a multi-layer perceptron network in Figure 3.2 is a vector with h components that are determined in terms of m components of an input vector x and n components of the hidden layer. Its mathematical representation may be expressed as

$$y_i = \sum_{j=1}^{n} \left[w_{ij} \varphi \Big(\sum_{k=1}^{m} s_{jk} x_k + B_{sj} \Big) + B_{wi} \right], \quad i = 1, \ldots, h \tag{3.3.3}$$

with s_{jk} denoting first-layer interconnection weights, and w_{ij} denoting second-layer interconnection weights. B_{wi} and B_{sj} represent biases of the output and hidden layers, respectively. A vector form of (3.3.3) is expressed as

$$y = W^T \varphi(S^T x) \tag{3.3.4}$$

Learning Algorithm

Learning is one of the most important features of a neural network. All knowledge in neural networks is encoded in interconnection weights, and the weights are determined during a learning process which is interpreted through an appropriate training algorithm. The network training is a basis for establishing a functional relationship between inputs and outputs of any neural network. A supervised learning and an unsupervised learning are two essential types of network training. In the course of a supervised learning phase, which is the most widely used for control, desired outputs of the network for every given input condition are specified. The network learns an appropriate functional relationship between them following repeated application of training sets of input-output pairs.

An *error back-propagation* algorithm or *generalized delta rule*, which is adopted in this research, belongs to the supervised learning type of network training. Weights of a multi-layer network are adapted iteratively by propagating some measurement of an error between a desired and an actual output of the network from its output back to its input. However, in neural network-based control applications, the relationship between an input and an output signal of a network is more complicated to specify and may be unknown. A typical way to alleviate this problem is to formulate a training process as a minimization of an energy function in weight space. In this research, the error signal for training the neural network-based controller is taken from a discrepancy between an output of the system and its desired reference trajectory. A *feedback-error-learning* ([28]) approach is employed based on this error.

The mathematical basis for an error back-propagation algorithm is an optimization technique known as a *gradient descent* method. According to this method there is an error function that defines a surface over weight space, and weights are modified in the opposite direction of gradient of this surface. Generally, three stages are involved in training a network through error back-propagation algorithm,

- the forward propagation of an input training pattern through the network,
- the calculation and backward propagation of associated errors, and
- the adjustment of weights to reduce the error.

An error function is defined as

$$E = \frac{1}{2} \sum_i \left(d_i - y_i \right)^2 \tag{3.3.5}$$

which is a continuous and a differentiable function. d_i is the desired output of unit i. A weight adjustment procedure is derived by computing a change in the error function with respect to a change in each weight. The above error function may be minimized by changing weights according to a recursive algorithm starting at output units and working back to the first hidden layer. Therefore, the weights update recursion is expressed as

$$w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t) \tag{3.3.6}$$

The weight change Δw_{ij} is calculated by a steepest descent method

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{3.3.7}$$

where the partial derivative is to be evaluated at current weight values using a chain rule for differentiation. In particular, for a linear activation function, an output is defined as $y_j = \sum_i w_{ij} x_i$. Then weight change is evaluated as

$$\Delta w_{ij} = \eta \delta_j x_i \tag{3.3.8}$$

with δ_j defined as an error signal $(d_j - y_j)$. For a nonlinear activation function, an output becomes $y_j = \varphi\Big(\sum_i w_{ij} x_i\Big)$, and the corresponding weight evaluation is

$$\Delta w_{ij} = \eta \delta_j x_i \varphi'(\cdot) \tag{3.3.9}$$

Successively, starting with an output layer, a gradient ∇E is calculated and is propagated back through network layers until an input layer is reached. The only difference between weight updating calculations in an output layer and those in hidden layers is that, for the output layer, an error depends on the difference

between the desired and its actual network output values, whereas for hidden layers, local errors are calculated on the basis of errors in a next (output) layer.

The learning rate η in (3.3.7) is a constant, and determines convergence speed of the weight adjustment step. Generally speaking, if η is small, a search path will closely approximate a gradient path, but the convergence will be very slow due to a large number of update steps needed to reach a local minima. If η is large, the convergence initially will be very fast, but the algorithm may eventually oscillate and thus not reach a minima.

Activation Function

Activation functions for hidden units are needed to introduce nonlinearity into the network. It is the nonlinearity that makes multi-layer networks very powerful. Sigmoidal functions such as logistic, *tanh* and Gaussian functions are the most common choices. In this research, a bipolar *tanh* function at a range of $[-1, 1]$ has been chosen as an activation function for all hidden units. The *tanh* function produces both positive and negative values, and tends to yield faster training because of its better numerical conditioning than functions that produce only positive values such as logistic. A mathematical representation of a *tanh* function may be expressed as

$$\varphi(net_i) = \frac{2}{1 + e^{-\lambda net_i}} - 1 \tag{3.3.10}$$

with a steepness parameter λ which is a slope of the activation function, and is often chosen as 1 or 2. A derivative of (3.3.10) is manipulated as

$$\varphi'(net_i) = \frac{2\lambda e^{-\lambda net_i}}{(1+e^{-\lambda net_i})^2} \tag{3.3.11}$$

It is noted that weights adjustment is proportional to a value of $\varphi'(net_i)$, when calculating the gradient ∇E of the overall error function $E(\cdot)$.

A linear activation function was chosen for output units, since the output signal in this research will contribute to a torque that is applied to the flexible-link manipulator. Generally, a torque value has no *a priori* bounded range, so that it is better to choose an unbounded activation function which is most often a linear function.

3.3.3 Function Approximation Fundamentals

Theoretical foundations for modeling and control of nonlinear dynamic systems with neural networks are based on the ability of networks of various types to approximate arbitrarily continuous nonlinear mappings. The mathematical basis for this ability is a function approximation theory, which deals with the problem of approximation and interpolation of a continuous function.

Mathematical results such as approximation and learning theories, have been utilized by some researchers. *Kolmogorov's superposition theorem* [40], a very basic theory refined later by others, proved that: for any continuous function $f(\cdot)$ of n real variables, there is a continuous function $g(\cdot)$ (of one variable) on [0, 1] into the real line such that

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} g\left[\sum_{p=1}^{n} \varphi_q(x_p)\right] \tag{3.3.12}$$

for all values of x_1, x_2, ..., and x_{2n+1} in [0, 1]. The function $g(\cdot)$ was specifically chosen as a continuous function of one variable, and $\varphi_q(\cdot)$ is a continuous, monotonically increasing function independent of $f(\cdot)$. The importance of *Kolmogorov's theorem* might not be recognized in its direct application for proving the universality of neural networks as function approximators, however it does point to the feasibility of using parallel and layered network structures for multivariate function mappings.

Poggio and Girosi [41] introduced a connection between artificial neural networks and approximation theory. They proposed *learning as approximation*, so that the problem of learning a mapping between an input and an output space was equivalent to a problem of estimating a system that transforms inputs into outputs. In the later case a set of examples of input-output pairs must have been given.

Rigorous mathematical proofs were provided for the universality of multilayer feedforward networks with continuous sigmoidal activations, as well as other general types of functions. Some attention has been paid to the *number* of hidden layers that a network should have. Funahashi [42] showed that any continuous mapping could be approximated by a network with at least one hidden layer using sigmoidal activation functions. He also stated that activation functions for an input and an output layer were linear in the sense of uniform topology. Two theorems and several corollaries were given by using an integral formula presented by Irie and Miyake [43] to substantiate statements in [42]. An essential concept which forms a mathematical basis for further analyzing approximation capabilities of the neural

networks was also stated.

Hornik *et al.* [44] employed the *Stone-Weierstrass theorem* to prove another important result, i.e. a standard multi-layer feedforward network with as few as one hidden layer using arbitrary squashing functions is capable of approximating any measurable function to any desired degree of accuracy, provided that sufficient hidden units are available.

The above results established multi-layer feedforward networks as a class of universal approximators. This implies that any failure from an approximation of a function mapping by a multi-layer network must arise from inadequate choice of parameters or an insufficient number of hidden nodes. The researchers [44] also showed that these networks could approximate not only an unknown function but also its derivatives.

Considering how many hidden layers are required for an approximation application, Chester [45] argued that, in practice, two layers of hidden units might give better interpolation capabilities with faster training. A problem with only one hidden layer, he commented, was that "the neurons therein interact with each other globally, making it difficult to improve an approximation at one point without worsening elsewhere." With more than one hidden layer, he continued, "approximations in different layers can be adjusted independently of each other, much as is done in the finite element method for solving partial differential equations or the spline technique for fitting curves."

It should be noted that a functional approximation capability of a neural network holds for all continuous functions over a compact set. Hence, a neural

network technique can be applied to approximations of nonlinear functions and parameters of these functions may not be linear.

The most important attribute of a multi-layer feedforward network is that it can learn a mapping of any complexity. Due to its universal approximating capability of nonlinear maps to any desired degree of accuracy, the network possesses a capacity to identify and hence is directly applicable to the control of nonlinear dynamic systems.

3.3.4 Proposed Neural Network-based Control Structure

The universal approximation properties of neural networks outlined earlier provide the motivation for trying to incorporate neural networks into a nonlinear control of a complicated dynamic system. It is expected that a controller comprised of neural networks would perform better than those normally performed by a conventional controller.

As Vemuri [40] pointed out, when a neural network plays a role in a control action, it could actually be utilized for generating input signals and using these signals to control its environment. Those neural networks which are used for actuator control functions are intentionally designed and trained in an environment in which the neural networks and the system to be controlled are placed in one closed-loop. In this case, using neural networks to control a dynamic system involves on-line training of the networks.

Depending on how and where neural networks are used in a control loop, many schemes have been investigated by researchers. Most practical applications

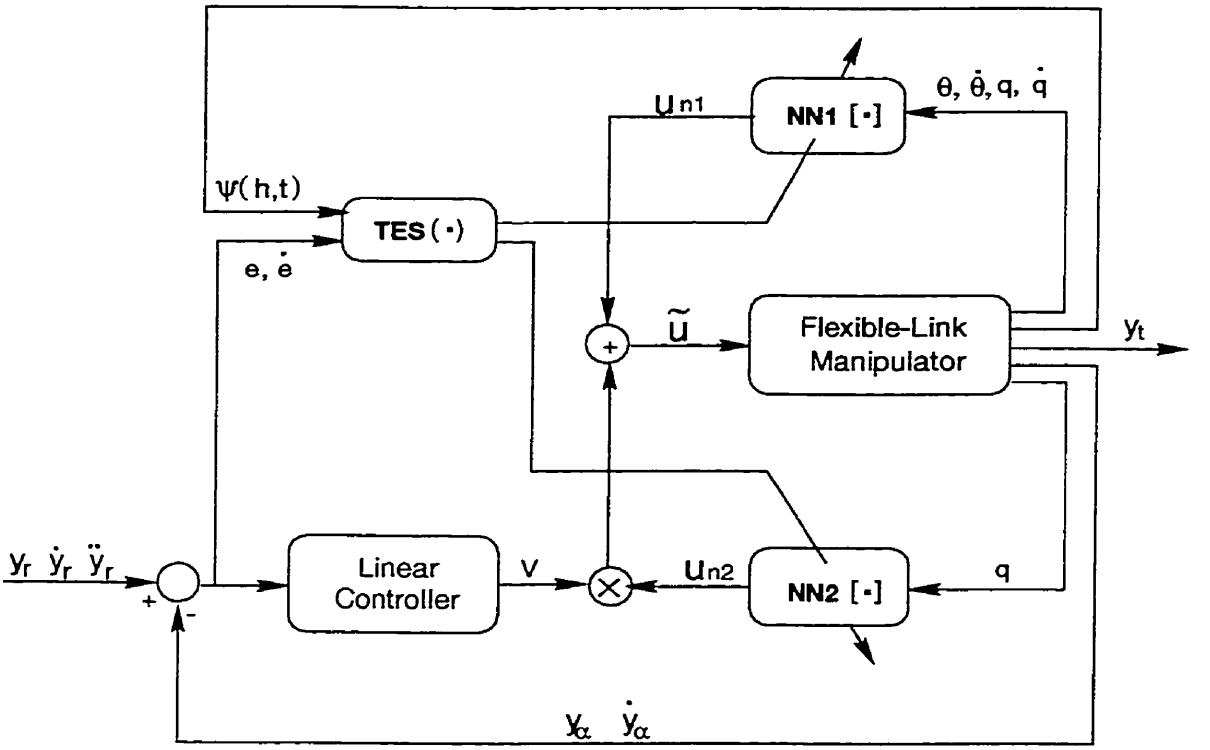

FIGURE 3.4. Neural Network-based Control Architecture for the Flexible-Link Manipulator

probably involve a mix of known and unknown dynamics, given that best results are obtained when conventional and neural network-based schemes are applied to known and unknown dynamics, respectively. A conventional nonlinear control scheme, such as the inverse dynamics control typically applied to rigid manipulators, is a well-established control framework that provides a basis for nonlinear control using neural networks. In this research, by taking advantage of this framework (depicted in Figure 3.1), two neural networks are proposed to learn nonlinear dynamics of a single-link flexible manipulator on-line. The trained networks will also work simultaneously with a conventional linear controller to generate a desired torque to the manipulator.

Control Configuration

The neural network-based controller proposed in this thesis is shown in Figure 3.4. The block $TES(\cdot)$ is introduced to the system for the two proposed networks $NN_1[\cdot]$ and $NN_2[\cdot]$ as a training signal/pattern generator. It is constructed using weighted output error, derivative of this error and the tip deflection. It is assumed that the nonlinear components of $F(\cdot)$ and $G(\cdot)$ in (3.2.5) are not known *a priori*. During the learning phase, two neural networks $NN_1[\cdot]$ and $NN_2[\cdot]$ are trained to functionally approximate the nonlinearities of the single-link flexible manipulator, represented by $F(\cdot)$ and $G(\cdot)$, respectively, by means of back-propagating the training signal (TES). Simultaneously, the output (u_{n2}) of network $NN_2[\cdot]$ is multiplied by the output (V) of the linear controller, and the result is added to the output (u_{n1}) of network $NN_1[\cdot]$. The combined total signal contributes to the control torque $(\bar{u})$ that is applied to the manipulator.

The neural network-based control may be presented mathematically as

$$\bar{u} = \boldsymbol{NN_2}[q]V + \boldsymbol{NN_1}[\theta, \dot{\theta}, q, \dot{q}] \tag{3.3.13}$$

with $NN_1[\cdot]$ and $NN_2[\cdot]$ as stated above approximating the nonlinearities $F(\cdot)$ and $G(\cdot)$ in (3.2.5), respectively, through a learning algorithm proposed in *Section 3.3.2*. The *a priori* assumptions here are as follows: $G(\cdot)$ is the function of the deflection variable (q) and $F(\cdot)$ is the function of the joint variable (θ), the deflection variable (q), and their derivatives ($\dot{\theta}$ and $\dot{q}$). Consequently, the network $NN_1[\cdot]$ receives $\theta, \dot{\theta}, q, \dot{q}$ as inputs and the network $NN_2[\cdot]$ receives only q as input. The linear controller $V(\cdot)$ defined in (3.2.7) facilitates an asymptotic stability of the error dynamics (3.2.8) of the closed-loop system during the learning process.

Cost Function and Derivatives

The learning process for an artificial neural network can be viewed as a process of searching in a multi-dimensional parameter space for a state that optimizes a pre-defined *criterion function* $J(\cdot)$, also called the *cost function*. To learn the characteristics of the dynamic model and to adapt to varying uncertainties, a neural controller must be given the values of its outputs. As discussed previously for a control system problem, it may not be easy to know *a priori* an appropriate *target* control signal and most of time it may not be available. This results in a major difficulty when one is designing a neural network-based controller by using a supervised learning scheme, as is the case in the standard back-propagation error learning algorithm.

However, a desired "target signal" for the output of the plant is always available. In the *feedback-error-learning* [28] scheme, the difference between an actual output of the plant, which in this research is the re-defined output y_a, and its desired output y_r is taken as an error to be used for the weights adaptation algorithm.

As previously stated the major objectives of the proposed controller are to achieve as small as possible a tip position tracking error, and at the same time, to minimize deflections of the flexible-link manipulator. In an attempt to achieve these objectives, the derivative of the output error ($\dot{e}$) and tip deflections ($\psi(h,t)$) may be added to the original cost function (3.3.5) for training purposes (as suggested by Talebi *et al.* [7]). Consequently, the modified cost function to be minimized by the neural networks is proposed as

$$J = \frac{1}{2}\left[e^T K_2 e + \dot{e}^T K_1 \dot{e} + \psi^T(h,t) K_0 \psi(h,t)\right] \tag{3.3.14}$$

where $e = y_r - y_a$ is the re-defined output error. Consequently, elastic vibrations may be directly controlled through K_0.

Applying the modified cost function (3.3.14) to the weights adjustment rule (3.3.7) results in

$$\Delta w = -\eta \left(\frac{\partial J}{\partial w}\right)^T \tag{3.3.15}$$

For the network NN_1:

$$\begin{aligned} \frac{\partial J}{\partial w_1} &= \frac{\partial J}{\partial e}\frac{\partial e}{\partial w_1} + \frac{\partial J}{\partial \dot{e}}\frac{\partial \dot{e}}{\partial w_1} + \frac{\partial J}{\partial \psi}\frac{\partial \psi}{\partial w_1} \\ &= e^T K_2 \frac{\partial e}{\partial w_1} + \dot{e}^T K_1 \frac{\partial \dot{e}}{\partial w_1} + \psi^T K_0 \frac{\partial \psi}{\partial w_1} \end{aligned} \tag{3.3.16}$$

where w_1 denotes the NN_1 weight vector. Since $e = y_r - y_a$ and $\dot{e} = \dot{y}_r - \dot{y}_a$, (3.3.16) becomes

$$\frac{\partial J}{\partial w_1} = -e^T K_2 \frac{\partial y_a}{\partial w_1} - \dot{e}^T K_1 \frac{\partial \dot{y}_a}{\partial w_1} + \psi^T K_0 \frac{\partial \psi}{\partial w_1} \tag{3.3.17}$$

and from Figure 3.4

$$\tilde{u} = V u_{n2} + u_{n1} \tag{3.3.18}$$

Furthermore,

$$\frac{\partial y_a}{\partial w_1} = \frac{\partial y_a}{\partial \tilde{u}} \frac{\partial \tilde{u}}{\partial w_1}, \quad \frac{\partial \dot{y}_a}{\partial w_1} = \frac{\partial \dot{y}_a}{\partial \tilde{u}} \frac{\partial \tilde{u}}{\partial w_1}, \quad \frac{\partial \psi}{\partial w_1} = \frac{\partial \psi}{\partial \tilde{u}} \frac{\partial \tilde{u}}{\partial w_1},$$

As suggested by Lightbody *et al.* [46], the following terms are computed using the sign of the gradients rather than their real values for training the neural networks:

$$\begin{aligned}
\frac{\partial y_a}{\partial \tilde{u}} \quad &\text{is approximated by} \quad sign(\frac{\partial y_a}{\partial \tilde{u}}) = 1 \\
\frac{\partial \dot{y}_a}{\partial \tilde{u}} \quad &\text{is approximated by} \quad sign(\frac{\partial \dot{y}_a}{\partial \tilde{u}}) = 1 \\
\frac{\partial \psi}{\partial \tilde{u}} \quad &\text{is approximated by} \quad sign(\frac{\partial \psi}{\partial \tilde{u}}) = -1
\end{aligned} \tag{3.3.19}$$

The final expression for (3.3.16) is presented as

$$\frac{\partial J}{\partial w_1} = -\Big(K_2 e^T + K_1 \dot{e}^T + K_0 \psi^T\Big) \frac{\partial \tilde{u}}{\partial w_1} \tag{3.3.20}$$

where

$$\frac{\partial \tilde{u}}{\partial w_1} = \frac{\partial (V u_{n2})}{\partial w_1} + \frac{\partial u_{n1}}{\partial w_1} = \frac{\partial u_{n1}}{\partial w_1} \tag{3.3.21}$$

Therefore, the weight adjustment for NN_1 becomes

$$\Delta w_1 = \eta\Big(K_2 e + K_1 \dot{e} + K_0 \psi\Big)\frac{\partial u_{n1}}{\partial w_1} \tag{3.3.22}$$

and $\dfrac{\partial u_{n1}}{\partial w_1}$ can be computed through the backpropagation algorithm.

For the network NN_2:

$$\begin{aligned}\frac{\partial J}{\partial w_2} &= \frac{\partial J}{\partial e}\frac{\partial e}{\partial w_2} + \frac{\partial J}{\partial \dot{e}}\frac{\partial \dot{e}}{\partial w_2} + \frac{\partial J}{\partial \psi}\frac{\partial \psi}{\partial w_2} \\ &= e^T K_2 \frac{\partial e}{\partial w_2} + \dot{e}^T K_1 \frac{\partial \dot{e}}{\partial w_2} + \psi^T K_0 \frac{\partial \psi}{\partial w_2}\end{aligned} \tag{3.3.23}$$

where w_2 denotes the NN_2 weight vector. Since $e = y_r - y_a$ and $\dot{e} = \dot{y}_r - \dot{y}_a$, (3.3.23) becomes

$$\frac{\partial J}{\partial w_2} = -e^T K_2 \frac{\partial y_a}{\partial w_2} - \dot{e}^T K_1 \frac{\partial \dot{y}_a}{\partial w_2} + \psi^T K_0 \frac{\partial \psi}{\partial w_2} \tag{3.3.24}$$

By using (3.3.18), we get

$$\frac{\partial y_a}{\partial w_2} = \frac{\partial y_a}{\partial \tilde{u}}\frac{\partial \tilde{u}}{\partial w_2}, \quad \frac{\partial \dot{y}_a}{\partial w_2} = \frac{\partial \dot{y}_a}{\partial \tilde{u}}\frac{\partial \tilde{u}}{\partial w_2}, \quad \frac{\partial \psi}{\partial w_2} = \frac{\partial \psi}{\partial \tilde{u}}\frac{\partial \tilde{u}}{\partial w_2},$$

Using (3.3.19), the final expression for (3.3.23) is presented as

$$\frac{\partial J}{\partial w_2} = -\Big(K_2 e^T + K_1 \dot{e}^T + K_0 \psi^T\Big)\frac{\partial \tilde{u}}{\partial w_2} \tag{3.3.25}$$

where

$$\begin{aligned}\frac{\partial \tilde{u}}{\partial w_2} &= \frac{\partial (V u_{n2})}{\partial w_2} + \frac{\partial u_{n1}}{\partial w_2} = \frac{\partial (V u_{n2})}{\partial w_2} \\ &= \frac{\partial u_{n2}}{\partial w_2} V + \frac{\partial V}{\partial w_2} u_{n2} = \frac{\partial u_{n2}}{\partial w_2} V\end{aligned} \tag{3.3.26}$$

Therefore, the weight adjustment for NN_2 becomes

$$\begin{aligned}\Delta w_2 &= \eta\Big(K_2 e + K_1 \dot{e} + K_0 \psi\Big)\frac{\partial u_{n2}}{\partial w_2} V \\ &= \underbrace{\eta V\Big(K_2 e + K_1 \dot{e} + K_0 \psi\Big)}_{(\bigstar)}\frac{\partial u_{n2}}{\partial w_2} \qquad (3.3.27)\end{aligned}$$

where $\dfrac{\partial u_{n2}}{\partial w_2}$ can be computed through the backpropagation algorithm, and the above term ($\bigstar$) is manipulated by using (3.2.7) as follows,

$$\begin{aligned}(\bigstar) &= V\Big(K_2 e + K_1 \dot{e} + K_0 \psi\Big) \\ &= \Big(\ddot{y}_r + K_p e + K_d \dot{e}\Big)\Big(K_2 e + K_1 \dot{e} + K_0 \psi\Big) \\ &= \ddot{y}_r K_2 e + \ddot{y}_r K_1 \dot{e} + \ddot{y}_r K_0 \psi \\ &\quad + K_p K_2 \underline{e^2} + K_p K_1 \underline{e\dot{e}} + K_p K_0 e\psi \\ &\quad + K_d K_2 \underline{e\dot{e}} + K_d K_1 \underline{\dot{e}^2} + K_d K_0 \dot{e}\psi \qquad (3.3.28)\end{aligned}$$

It now follows that for NN_2, the training signal is a nonlinear function of the error and its derivative (as can be seen from the underlined terms in (3.3.28)). For sake of simplicity, in the training process, we have approximated V by setting it equal to one, however have used its actual value in the design of the control law (3.3.18). In this way, the linear training signal may be used for network NN_2 which is the same as that of network NN_1 ([47]).

Chapter 4

Simulation Studies

In this chapter, numerical simulations are presented in which a neural network-based control strategy proposed in the previous chapter, as well as two conventional control schemes are implemented for a single-link flexible manipulator. In *Section 4.1* the objectives and goals of the simulations are stated. Control design considerations are addressed in *Section 4.2* both for the dynamic model and the neural networks. *Section 4.3* provides simulation for step and sinusoidal reference trajectories having different amplitudes. Quantitative comparisons between the proposed neural network-based controller and the conventional controllers are presented. The final comments on the simulation results are provided in *Section 4.4*.

4.1 Introduction

In order to evaluate the effectiveness and illustrate the potential of the control strategies developed in the previous chapter, the proposed neural network-based controller as well as a standard inverse dynamics controller and a PD-type controller are all applied to a single-link flexible manipulator for performing numerical simulations. Due to the function approximation capabilities of neural networks and their application potential an improvement of the system performance is expected, especially in comparison with conventional schemes. At the same time, an "intelligent" characteristic of neural networks should be clearly demonstrated.

Two neural networks, which are multi-layer perceptrons, are trained on-line together with a linear controller to generate an appropriate torque that drives the tip (y_t) of the flexible link to its desired trajectory (y_r) (schematically shown in Figure 3.4). This task is accomplished by minimizing the cost function $J(\cdot)$ (3.3.14) through adjusting weights of the networks using the steepest descent gradient approach.

All the simulations are performed using *Matlab* 5.3/*Simulink* 1.1.

4.2 Design Considerations

4.2.1 Dynamic Model Aspects

As stated in *Chapter 2*, a flexible manipulator has infinite number of modes distributed along its link. For purposes of numerical simulations, the assumed mode method ([33], [34]) is adopted in the following simulations. Moreover, it is well

known that the first few modes of the manipulator dominate the dynamic characteristics of the system. Accordingly, in this research only the first elastic mode is extracted. It is also assumed that the states chosen for implementation purposes, and their derivatives, are all measurable from the system.

Link parameters associated with the target single-link flexible manipulator are shown in Table 4.1. All parameters are originally taken from Geniele [31] who used a model validation technique.

Name	Value	Name	Value
h(length of link)	$1.2m$	$EI^{\dagger}$	$1.94Nm^2$
I_h(hub inertia)	$0.3kgm^2$	$b^{\ddagger}$	$1.29Nm/rad\,s^{-1}$
γ(mass per unit length)	$1.2kg/m$	ω_1^*	$3\,rad/s$
C_2(damping coefficient)	0.4	$C^{\star}_{coul}$	$4.74Nm$ for $\dot{\theta} > 0$
M_p(payload mass)	$30g$	C_{coul}	$4.77Nm$ for $\dot{\theta} < 0$

† Young's modulus and beam area moment of inertia,
‡ viscous friction at hub,
* resonance frequency of the vibration,
⋆ Coulomb friction coefficient.

Table 4.1. Numerical data for the experimental flexible-link manipulator

As pointed out in *Section 1.2*, the non-minimum phase characterics of a flexible-link manipulator restricts a direct use of the tip position feedback control. However, the re-defined output approach ([4], [6]) rectified this problem, by defining a new output as close as possible to the tip of the manipulator ([6]), such that the associated *zero dynamics* are asymptotically stable.

Details concerning the derivation of the *zero dynamics* of the single-link manipulator may be found in Talebi *et al.* [7]. In their study a critical value α^* of the re-defined output parameter was determined to guarantee stability of the *zero*

dynamics of the system. They stated that a value of $\alpha^* = 0.6$ was found to stabilize the *zero dynamics*, and at the same time to be robust to payload mass variations. For this reason, $\alpha = 0.6$ is also chosen for all simulations conducted in this chapter.

4.2.2 Neural Networks Aspects

Networks Configurations

In principle, one hidden layer is sufficient for any approximation ([40]). Chester [45] suggested that, in practice, two layers of hidden units might give better interpolation capabilities with faster training. In our simulations, a two-hidden multi-layer perceptron is chosen for the proposed two neural networks.

Network NN_1 is configured as $\mathcal{N}_{4,3,2,1}$ which implies that NN_1 has 4 input neurons, 3 first hidden neurons, 2 second hidden neurons, and 1 output neuron. NN_1 takes as inputs, joint position θ, derivative of joint position $\dot{\theta}$, elastic deflection variable q, and derivative of the elastic deflection variable $\dot{q}$.

The configuration of network NN_2 is $\mathcal{N}_{1,3,3,1}$ which allows NN_2 to have 1 input neuron, 3 first hidden layer neurons, 3 second hidden layer neurons, and 1 output neuron. NN_2 only receives elastic deflection variable q as an input.

Initialization Issues

The choice of initial weights will influence whether a network reaches a global /local minimum of the error and, if so, how quickly it converges ([48]). Moreover, the update of the weight between two units depends on the derivatives of the two activation functions. Therefore, it is important to avoid choosing initial weights which

will force either the activation outputs, or their derivatives to become very small.

If values for the initial weights are too large, initial input signals to each hidden or output unit will be likely to fall into a region known as the saturation region, where the derivative of the activation function has a very small value. If initial weights are too small, a network input to a hidden or output unit will be close to zero, which also causes extremely slow learning. In simulation, all weights are initialized to random values between $[-0.2\ 0.2]$. Within this region, values are allowed to be either positive or negative, so that final weights after training may also be of either sign. The randomness involved in the initialization prevents units from adopting similar functions and becoming redundant.

An *universal approximation* property of multi-layer perceptrons with most commonly-used hidden-layer activation functions does not hold if bias units are omitted. A bias has the effect of increasing or lowering the network input of an activation function, depending on whether it is positive or negative, respectively. It can be treated as a connection weight from an input called a *bias unit* with a constant value of one. A single bias unit is connected to every hidden or output unit that needs a bias value. Hence bias values can be learned in the same way as other weights. In the simulation, the initial value of biases is chosen as 0.6.

Learning Rate

The convergence speed of a back-propagation algorithm is directly related to a learning rate parameter η in (3.3.7). It determines what amount of calculated error sensitivity to a weight change will be used for a weight correction. The rationale for choosing a learning rate η is based on the knowledge of the shape of an error surface,

which is rarely available.

If η is too small, a search path will closely approximate a gradient path, but convergence will be very slow due to a large number of update steps needed to reach a local minima ([49]). On the other hand, if η is too large, convergence initially will be very fast, but the network may be unstable and overshoot a minimum in a weight space, or else a network may oscillate around a minimum without any further learning. A general rule might be to use the largest learning rate that produces a suitable convergence speed but does not cause oscillation. However, it should be pointed out that only small learning constants guarantee a true gradient descent. In this simulation, a fixed value of $\eta = 0.3$ is chosen as the learning rate for all networks training.

Other Considerations

In our simulation results, gains for the linear controller $V(\cdot)$ (3.2.7) are chosen as $K_p = 20$ and $K_d = 30$. The gains for the cost function $J(\cdot)$ (3.3.14) are chosen after some trail and error as $K_0 = 5$, $K_1 = 4$, and $K_2 = 4$.

4.3 Results and Analysis

4.3.1 Step References

A $y_r = 0.8\,rad$ step reference trajectory is applied to the proposed neural network-based controller, as well as the PD-type controller (with $K_p = 80$, $K_d = 80$) and the inverse dynamics controller (with $K_p = 30$, $K_d = 35$). The PD-type control results

in a very poor control of the tip deflection as can be seen from Figure 4.1(a)(c)(e), and also poor reaction to payload variations (see Figure 4.1(b)(d)(f)). Even after 15 *sec*, the tip position (Figure 4.1(a)) still fluctuates around 0.8 *rad*. When a 603 g payload mass is added to the tip, significant oscillations occur in the tip response (Figure 4.1(b)). The standard inverse dynamics control scheme (shown in Figure 4.2) results in an improvement over the PD-type control. The small oscillations in the transient response depicted in Figure 4.2(a) cause the tip response to reach the steady-state in 7 *sec*. Adding a 603 g payload mass causes a bigger overshoot and more oscillations as shown in Figure 4.2(b).

The proposed neural network-based controller demonstrates a significant improvement over the other two controllers as illustrated in Figure 4.3. The tip of the manipulator tracks the 0.8 *rad* step reference trajectory with almost no overshoot or undershoot (shown in Figure 4.3(a)). A 603 g payload mass added to the tip generates only a 12.5% overshoot at the tip. The payload variations experienced is clearly much better than the other two conventional control strategies. The comparative performances of the tip position tracking for the above three control strategies corresponding to the 0.8 *rad* step reference trajectory are summarized in Table 4.2.

When the magnitude of the step reference trajectory is increased to 1.5 *rad*, the transient behaviors of both PD-type control (Figure 4.4(a)(b)) and inverse dynamics control (Figure 4.4(c)(d)) are degraded. Whereas, the performance of the proposed neural network-based controller, as can be seen from Figure 4.4(e)(f), remains almost the same as that of the 0.8 *rad* trajectory (see Figure 4.3(a)(c)), except

Control Scheme	Payload (g)	Settling Time(s)	Overshoot (%)	Undershoot (%)	Reference Figure
PD type	30	after 15	–	21.3	4.1 (a)
	603	after 15	12.5	25.0	4.1 (b)
Inverse	30	6	–	12.5	4.2 (a)
Dynamics	603	6	15.0	6.3	4.2 (b)
Neural	30	4	2.5	1.0	4.3 (a)
Networks	603	6	12.0	3.5	4.3 (b)

Table 4.2. Comparative performances for the three controllers corresponding to a 0.8 *rad* step reference trajectory

that there is less than a 5% overshoot in the tip response (Figure 4.4(e)).

In the next set of simulations, starting with a 0.4 *rad* step reference trajectory, every 10 *sec* the magnitude of the trajectory is increased by 0.4 *rad* until 2.0 *rad* is reached. Applying this reference trajectory to the inverse dynamics controller (results shown in Figure 4.5) and the neural network-based control scheme (results shown in Figure 4.6), the *learning ability* of the neural networks is clearly seen. In the course of training, starting from the third window (20 *seconds*), the transient performance of the system is substantially improved. Specifically, the settling time is decreasing as shown in Figure 4.6(a). The tip oscillations are also suppressed, and the magnitude of the tip deflection during transient is reduced in consecutive windows as shown in Figure 4.6(b). When the inverse dynamics scheme is applied, we observe no improvement in the system performance. Figure 4.5 shows that the performance during the five windows are exactly the same, implying that there is no learning with this conventional control scheme as expected.

4.3.2 Sinusoidal References

Figures 4.7 through 4.11 further demonstrate clear improvements that may be achieved in tip position tracking control by using the proposed neural network-based controller as compared with the other two conventional control schemes corresponding to sinusoidal reference trajectories.

When a $0.8sin(t)$ reference trajectory is applied, there is some tracking error and transient overshoot in the tip position response with the PD-type control (see Figure 4.7(a)). Adding a $603\,g$ payload mass to the tip of the manipulator causes irregularities in the tip response (as depicted in Figure 4.7(b)). The performance results for the inverse dynamics control scheme (Figure 4.8) shows improvement over the PD-type control (Figure 4.7). The superiority of the neural network-based control scheme over the conventional strategies is demonstrated in Figure 4.9.

When increasing the magnitude of the sinusoidal reference trajectory to $1.0sin(t)$, the results are shown in Figure 4.10. These performances are obtained when the neural network-based control scheme is used under different payload conditions. When a payload is increased to $603\,g$, there is some tracking error at about $5\,sec$, however, after $8\,sec$ the tip position tends to track the reference trajectory again. The learning ability of neural networks is depicted in Figure 4.10(b). Note that the system becomes **unstable** when a $603\,g$ payload is used for $1.0sin(t)$ reference trajectory subject to the two conventional control strategies. Performance of the tip position and the arm deflection under the three control strategies for a $1.5sin(t)$ reference trajectory are shown in Figure 4.11. These results reinforce the advantage and superiority of the neural network-based control scheme when applied

to single-link flexible manipulators.

4.4 Summary

Numerical simulations are carried out to exhibit the potential and effectiveness of the proposed neural network-based controller over conventional control schemes in suppressing elastic vibrations to obtain satisfactory tip position tracking performance.

It has been found that even a well-tuned PD-type control always results in oscillations or tracking errors in the tip position response. In other words, the tip position tracking performance can not be only improved through PD-type control. Since the target model involves many nonlinear dynamics and components, such as Coriolis and centrifugal forces, Coulomb friction, and uncertainties in model parameters, a linear controller cannot properly compensate these nonlinearities with certainty. The simulation results for PD-type control clearly show this limitation for our highly nonlinear system.

Generally reasonable performances for the closed-loop system may be achieved at least in simulations by implementing inverse dynamics control (the typical conventional nonlinear control scheme) based on a *a priori* knowledge of the exact mathematical model. However, when operating conditions are changed, for instance the payload or the magnitude of the reference trajectory are varying, the degradation in the overall system performance is obtained. This is due to the dependence of the inverse dynamics control strategy on the knowledge of the exact dynamic model.

The proposed neural network-based control scheme that is motivated by the inverse dynamics control, clearly demonstrated more robustness to the above

variations as compared to the other two conventional control schemes. In particular, the transient behavior of the closed-loop system is improved significantly. Two networks are trained on-line, to approximate the nonlinearities of the given mathematical model which is assumed to be unknown *a priori*, and at the same time to generate the appropriate torque needed to drive the tip position of the manipulator follow a desired trajectory. The other advantage of the neural network-based control scheme is the usage of a lower control effort as illustrated in Figure 4.12(c), in particular, when compared to the case of a PD-type control as shown in Figure 4.12(a).

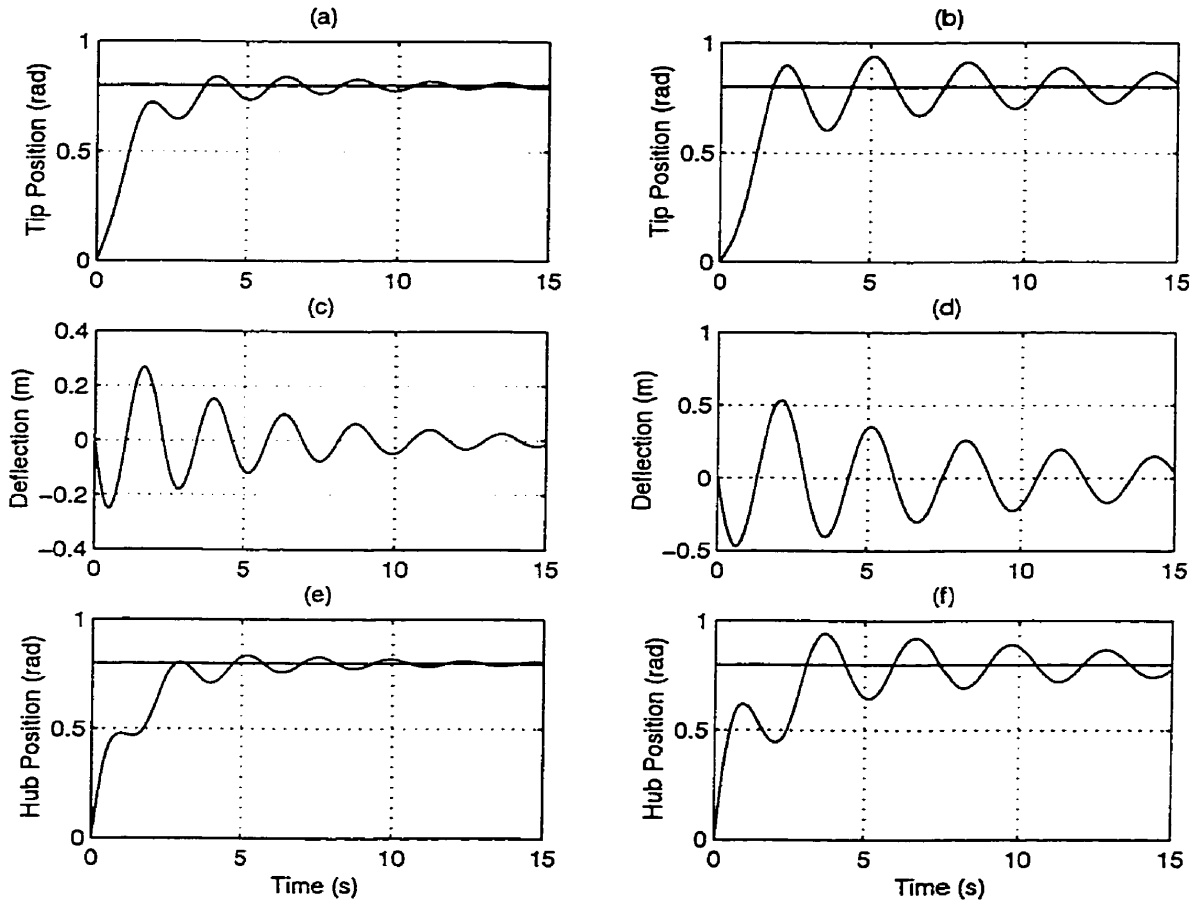

FIGURE 4.1. Simulation results for PD-type control with $0.8\,rad$ step reference trajectory. (a) (c) (e) with payload $M_p = 30g$, (b) (d) (f) with payload $M_p = 603g$

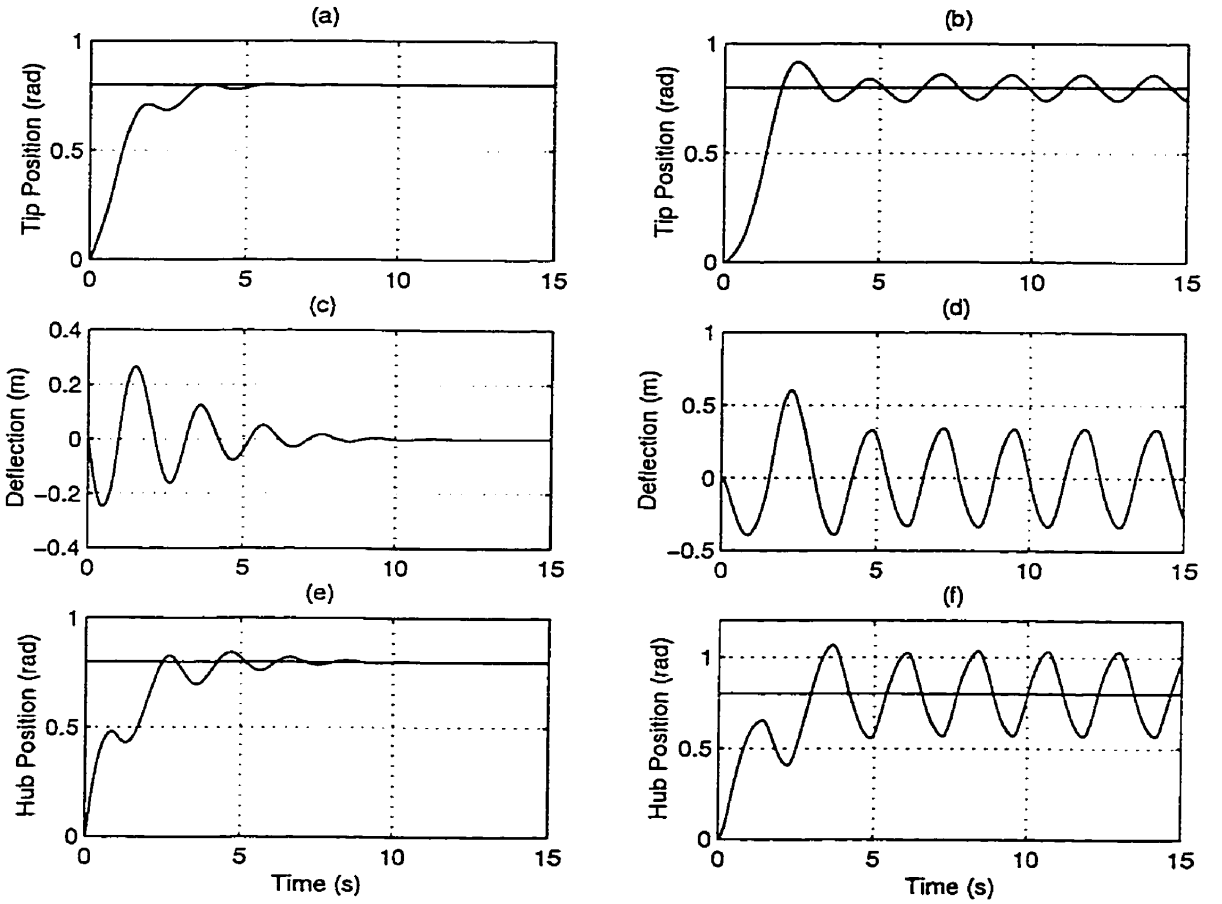

FIGURE 4.2. Simulation results for inverse dynamics control with $0.8\,rad$ step reference trajectory. (a) (c) (e) with payload $M_p = 30g$, (b) (d) (f) with payload $M_p = 603g$

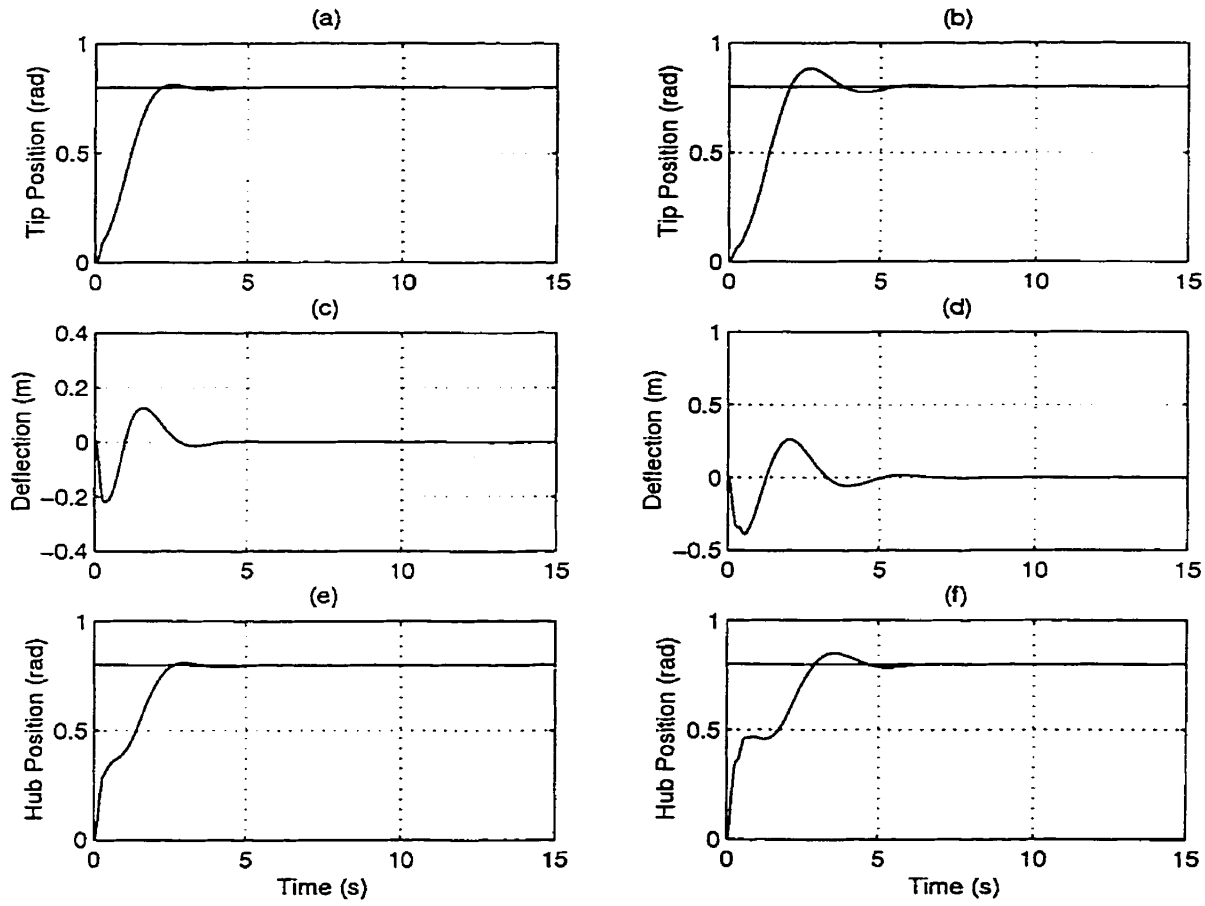

FIGURE 4.3. Simulation results for neural network-based control with 0.8 *rad* step reference trajectory. (a) (c) (e) with payload $M_p = 30g$, (b) (d) (f) with payload $M_p = 603g$

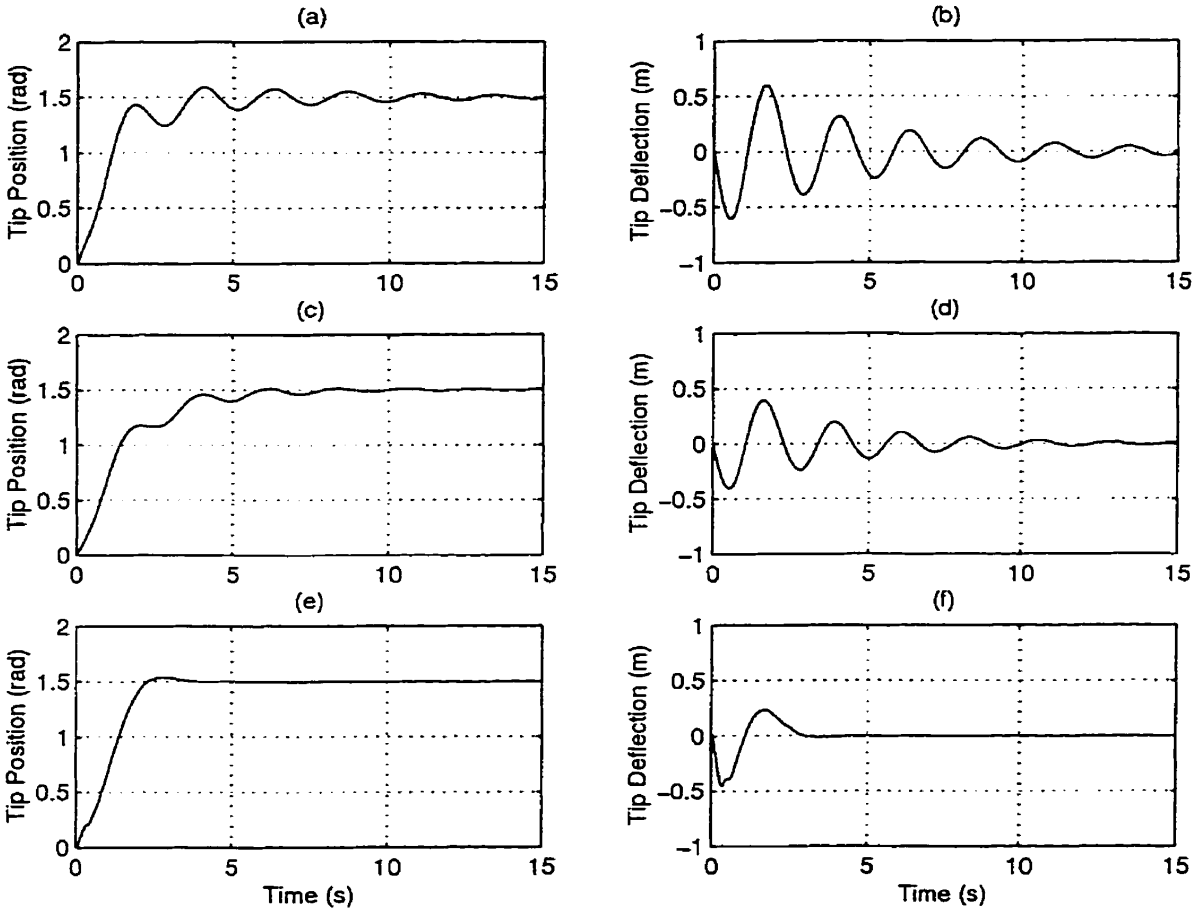

FIGURE 4.4. Simulation results for 1.5 *rad* step reference trajectory with payload $M_p = 30g$. (a) (b) PD-type control, (c) (d) inverse dynamics control, (e) (f) neural networks-based control

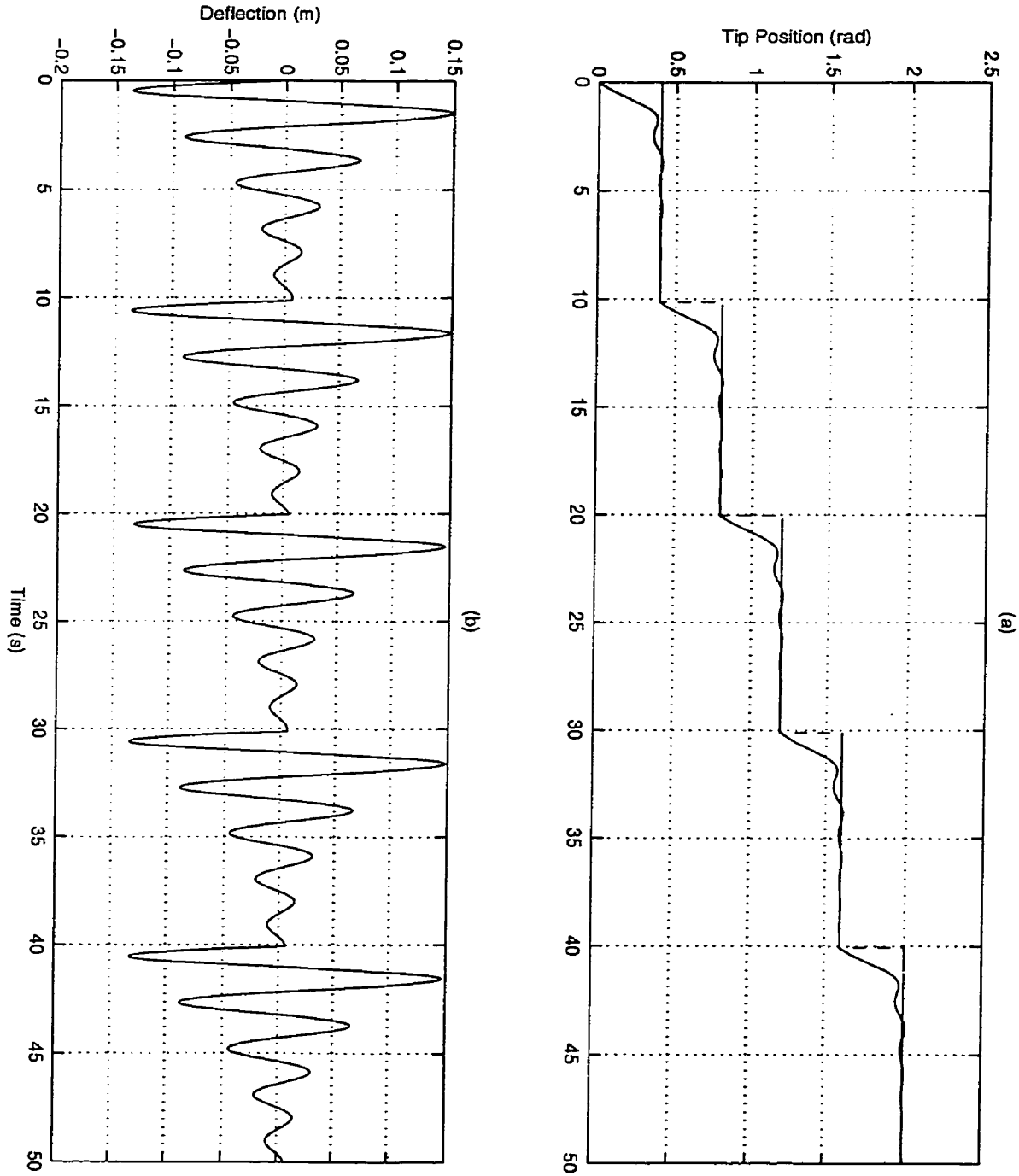

FIGURE 4.5. Simulation results for inverse dynamics control with payload $M_p = 30g$. Starting with $0.4\,rad$ step reference trajectory, every $10\,sec$, magnitude of the reference trajectory increases by $0.4\,rad$ until $2.0\,rad$ is reached.

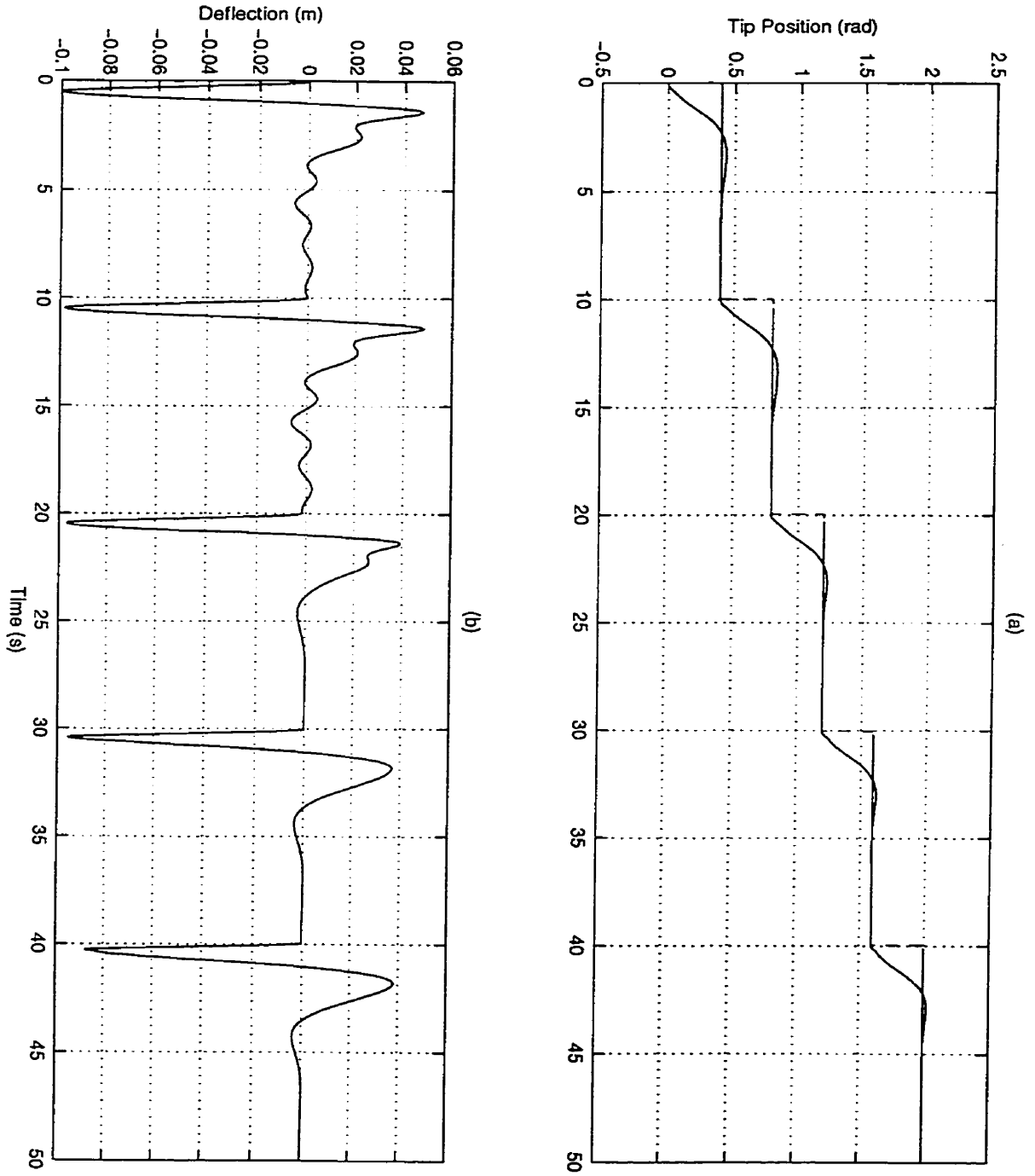

FIGURE 4.6. Simulation results for neural network-based control with payload $M_p = 30g$. Starting with $0.4\,rad$ step reference trajectory, every $10\,sec$, magnitude of the reference trajectory increases by $0.4\,rad$ until $2.0\,rad$ is reached.

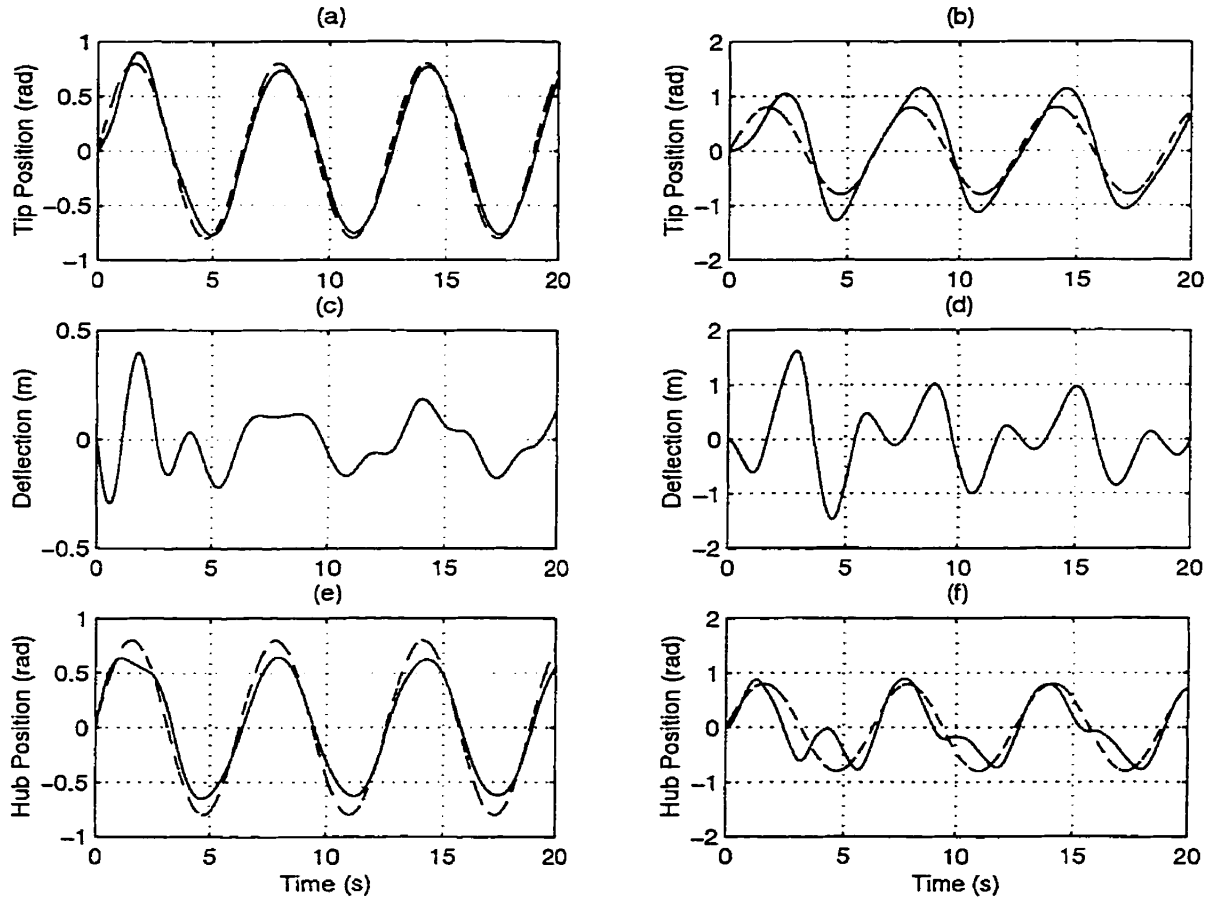

FIGURE 4.7. Simulation results for PD-type control with $0.8sin(t)$ reference trajectory (dashed line). (a) (c) (e) with payload $M_p = 30g$, (b) (d) (f) with payload $M_p = 603g$

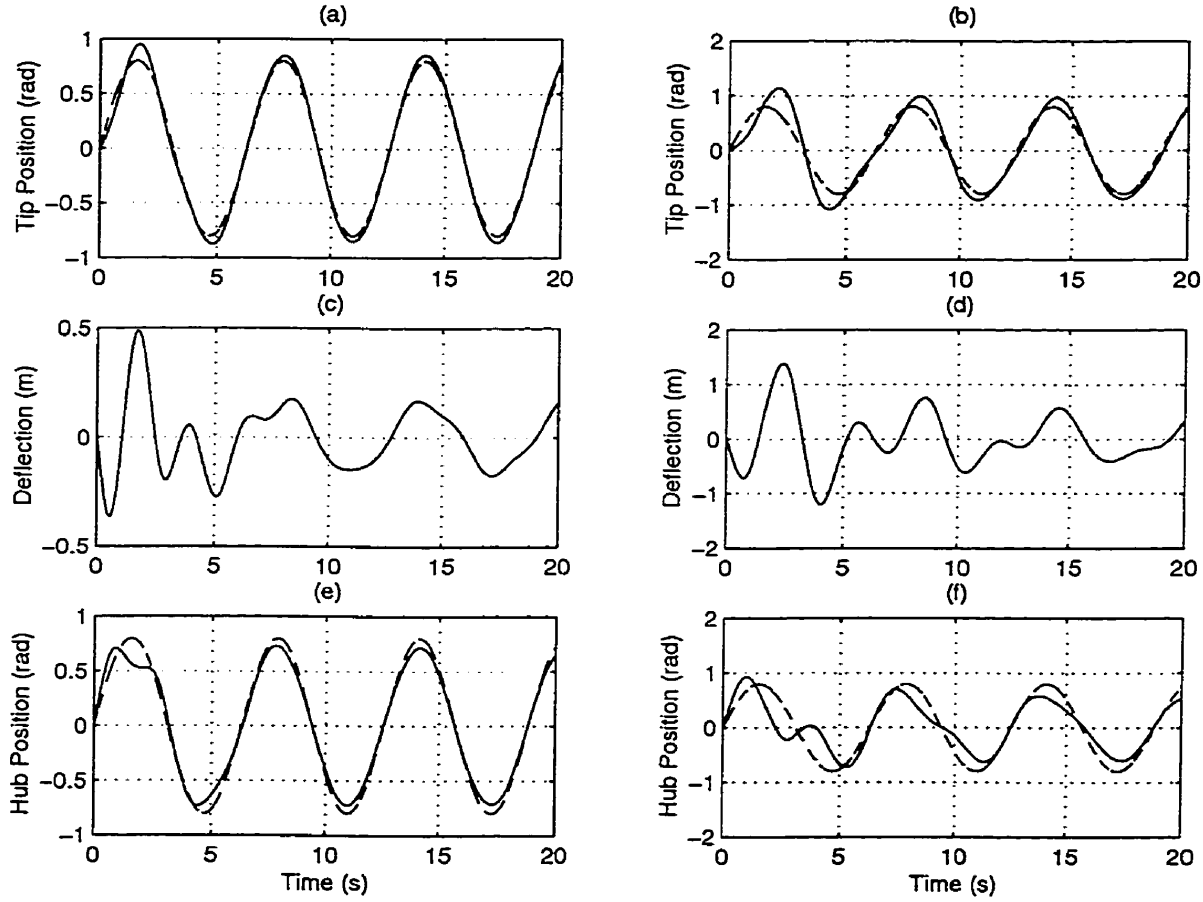

FIGURE 4.8. Simulation results for inverse dynamics control with $0.8sin(t)$ reference trajectory (dashed line). (a) (c) (e) with payload $M_p = 30g$, (b) (d) (f) with payload $M_p = 603g$

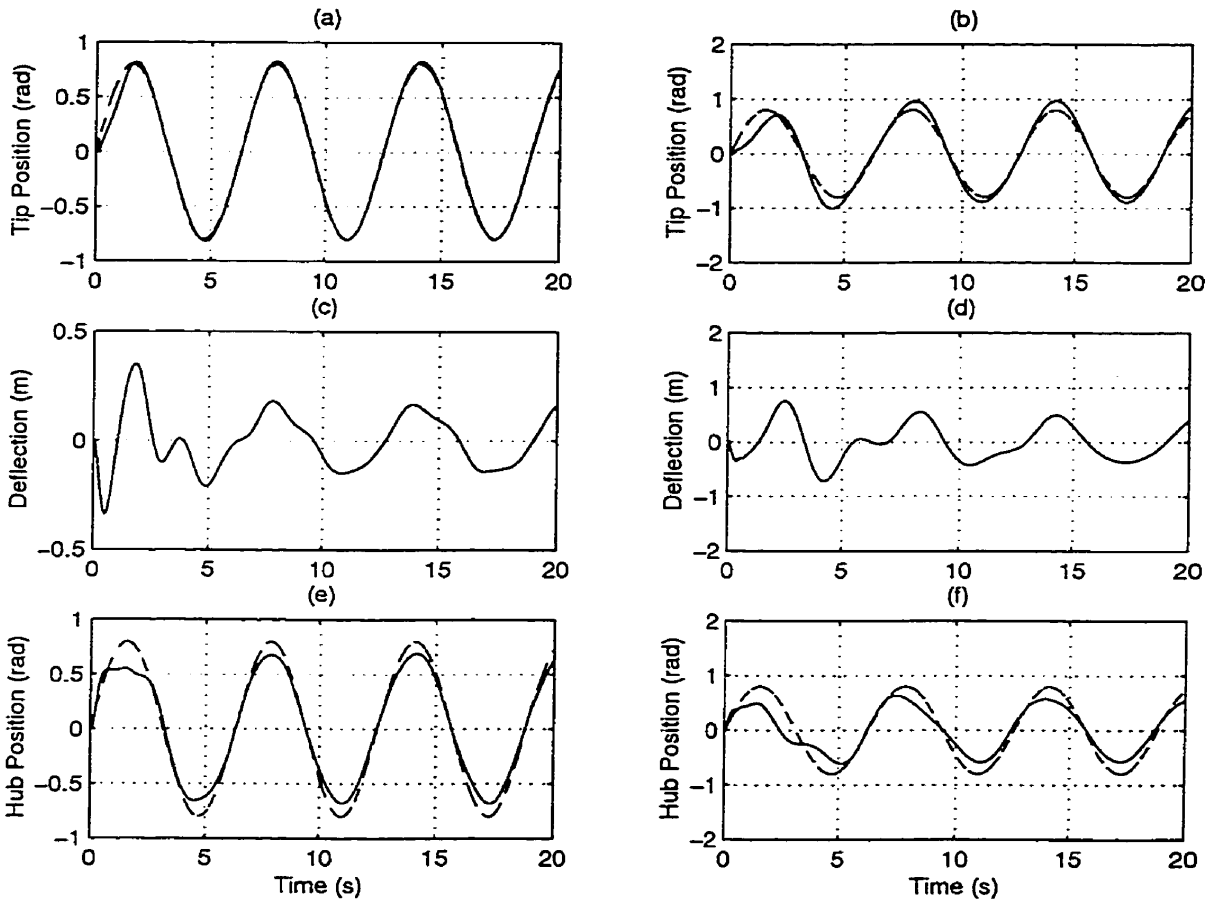

FIGURE 4.9. Simulation results for neural network-based control with $0.8sin(t)$ reference trajectory (dashed line). (a) (c) (e) with payload $M_p = 30g$, (b) (d) (f) with payload $M_p = 603g$

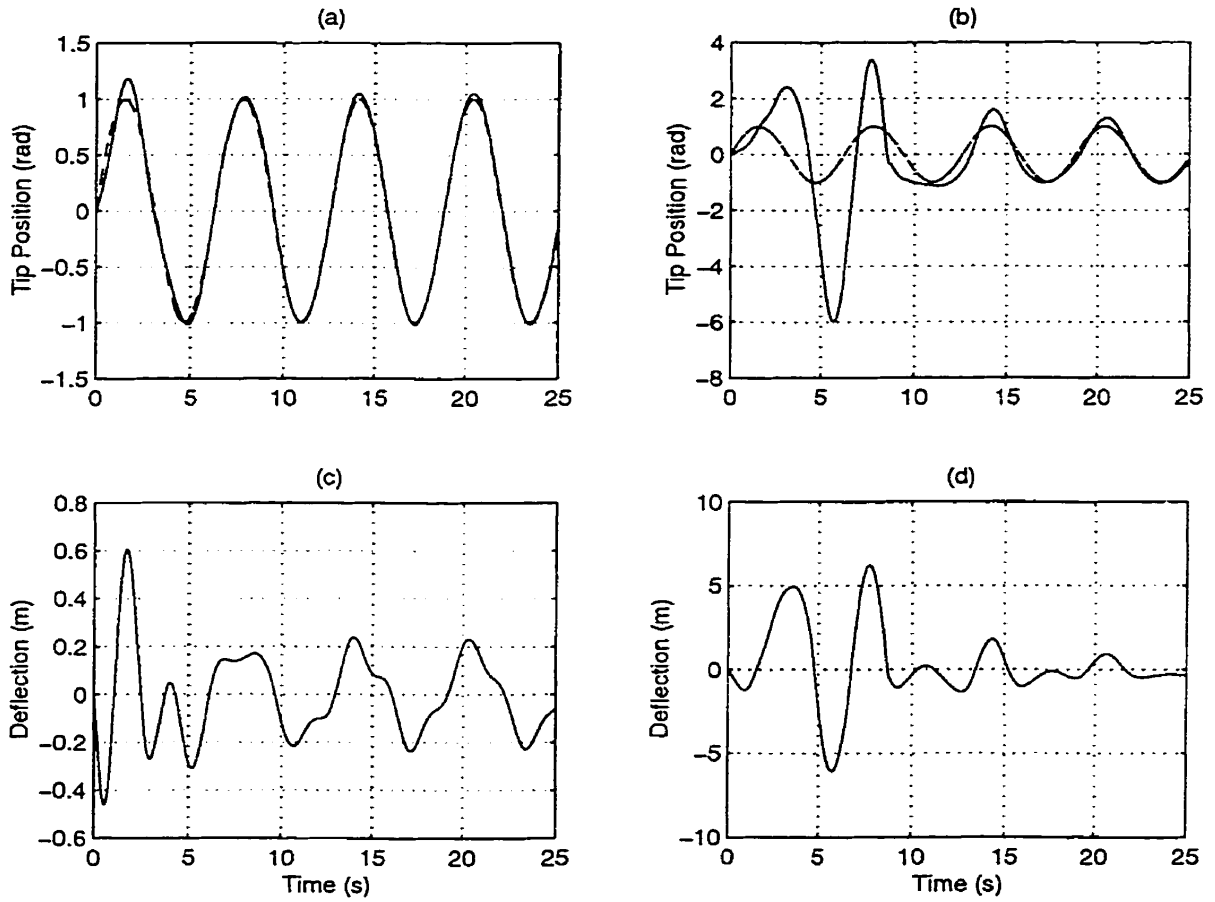

FIGURE 4.10. Simulation results for neural network-based control with $1.0sin(t)$ reference trajectory (dashed line). (a) (c) with payload $M_p = 30g$, (b) (d) with payload $M_p = 603g$

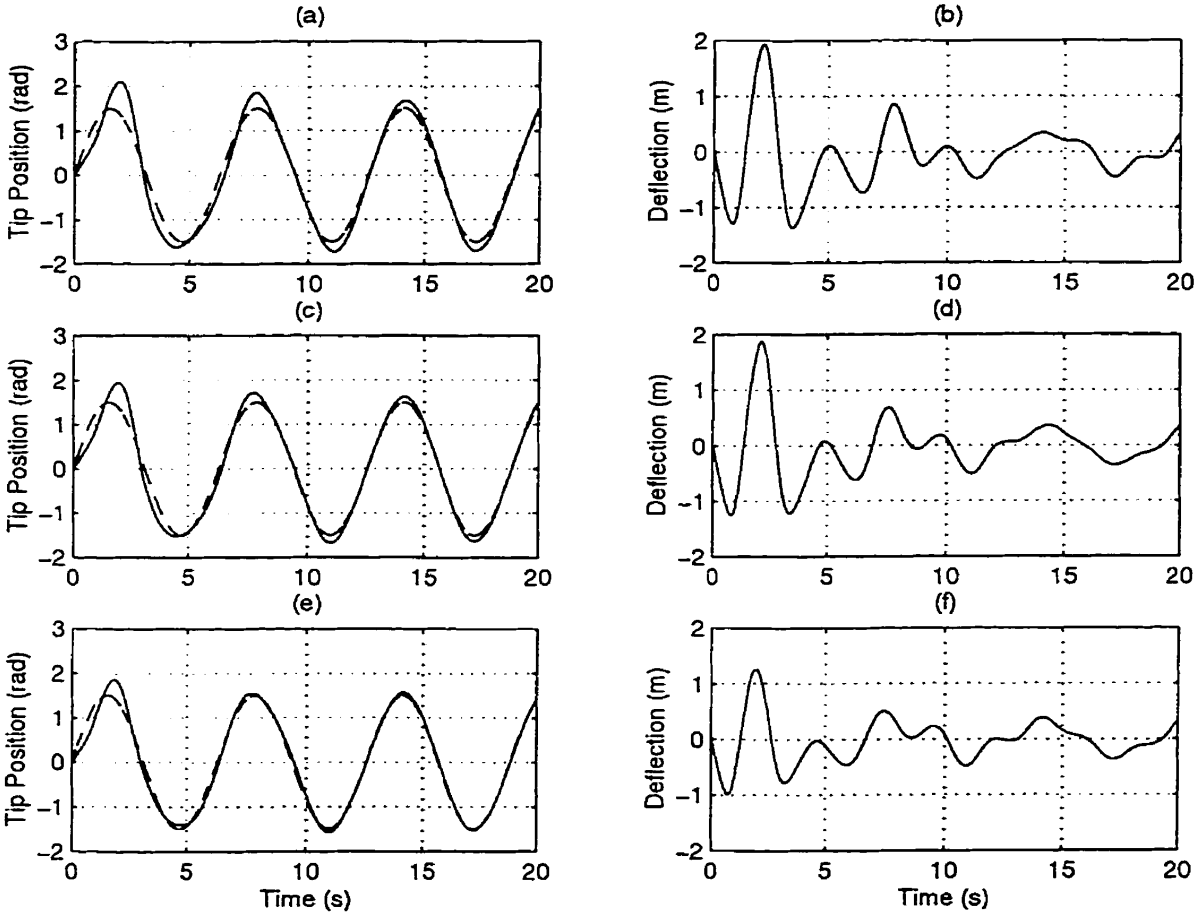

FIGURE 4.11. Simulation results for $1.5sin(t)$ reference trajectory (dashed line) with payload $M_p = 30g$. (a) (b) PD-type control, (c) (d) Inverse dynamics control, (e) (f) Neural network-based control

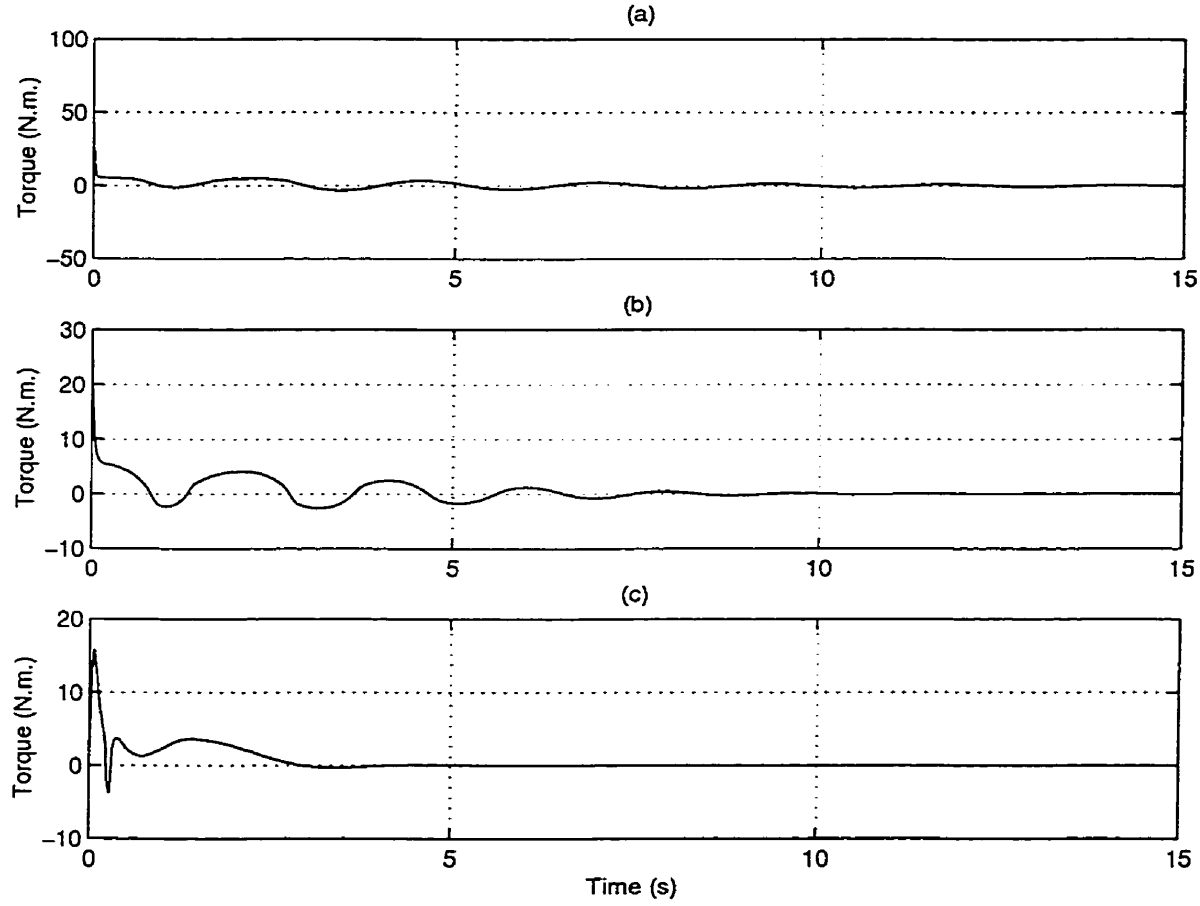

FIGURE 4.12. Simulation results for $0.8\,rad$ step reference trajectory with payload $M_p = 30g$. (a) PD-type control, (b) inverse dynamics control, (c) neural networks-based control

Chapter 5

Experimental Evaluations

In this chapter, the three control strategies discussed in previous chapters are digitally implemented on an experimental test-bed (Geniele [31]) of a single-link flexible manipulator. Specifically, the superiority of the proposed neural network-based controller is further verified and validated experimentally. In *Section 5.1*, the real-time system configuration consisting of the experimental test-bed, the sensors and actuators, the DSP board, etc. are described in some detail. The software development and its environment with respect to the experimental implementation is addressed in *Section 5.2*. Several aspects related to digitally implementing the proposed neural network-based controller are considered in *Section 5.3*. Experimental results, analysis, and discussions are provided in *Section 5.4*, followed by a summary in *Section 5.5*.

5.1 Hardware Configuration

The essential components of the experimental real-time system (see Figure 5.1) are as follows: a TMS320C30 system DSP board, a host PC, a single-link flexible manipulator and an interface board. The DSP board mounted in a 16-bit expansion slot within the host PC, implements the control algorithms using a TMS320C30 DSP chip that operates on a 33.3 MHz clock cycle and performs 16.7 million instructions per second. The 16-bit dual-channel A/D and D/A systems are mounted on the system board. The input channel includes sample/hold amplifiers, and both input and output channels are buffered through 4th order Sallen-Key lowpass filters. Input filters limit noise and provide anti-aliasing protection when the sampling rate is much higher than the filter cutoff frequency. Output filters provide a smoothing of the otherwise stepped D/A output signal. One channel A/D and D/A system is used in this experiment and clocked by a programmable *Am9513APC system timing controller* which is located on a separate interface board outside the host PC.

An expansion bus of the TMS320C30 DSP chip is connected to this separate interface board. Several components are integrated on the interface board, including a programmable timing controller and its circuitry, high current sources driving an infrared diode mounted at the tip of the flexible link, and circuitry decoding the hub position which is detected by an *optical incremental encoder.*

The host PC interfaces between the user and the executive TMS320C30 system board to send user commands to, and acquire data from, a dual-ported on-board RAM in real-time. Simultaneously, the execution of the whole closed-loop control system is monitored on the screen.

The single-link flexible manipulator is driven by a *DC servo motor* located at the bottom of the hub. It is a permanent magnet, brush type DC servo motor which generates a torque $\tau_m(t) = K_t i_a(t)$, where $K_t = 0.1175\, Nm/A$. Since the motor is a high-speed and relatively low-torque actuator, it is coupled to the hub through a *harmonic drive speed reducer* with a gear ratio of $50:1$ which decreases the speed of the motor, and provides a sufficient torque to accelerate the hub. The output torque has a range of $\pm 35.25\, Nm$. The optical incremental encoder is attached to the DC servo motor making the hub position (θ) digitally available. This digital signal is decoded through an integrated circuitry on the interface board for feeding back the hub position (θ) and its derivative ($\dot{\theta}$) to the closed-loop control system.

An Opto Diode OD-50L Super High-Power GaAIAs *infrared emitting diode* is mounted at the tip of the flexible link, providing the infrared light source for an *UDT camera*. This diode emits $880\, nm$ non-coherent infrared radiant energy with extra output power up to $600\, mW$ in peaks of optical output pulses. The *UDT camera* consists of a wide angle lens and a lateral-effect *photodiode detector* assembly. It is located on one side of the link positioned on the top of the hub, cooperating with the photodiode detector to determine the tip deflection (ψ). After being processed by a *signal conditioning amplifier*, this analog signal is sent to an A/D converter on the DSP board to make the signal digitally available for feeding back to the closed-loop control system. It should be noted that beyond a deflection range of $\pm 0.25\, m$, the diode cannot accurately be detected by the camera, even though it is still within the lens field of view of $\pm 0.5\, m$.

Other technical and functional details about the experimental test-bed and

its peripheral devices may be found in Geniele [31].

5.2 Software Components

The software developed for the TMS320C30 based control system consists of two parts, a DSP program and a host PC program. The DSP program is the essential program written in C language with in-line assembly language statements. It is responsible for handling timing controller, control algorithm and interrupt service routine. The interrupt service routine is synchronized through an external programmable timing controller mounted on the interface board. Once an external trigger signal from the timing controller initiates the A/D, the A/D starts to perform the conversion and outputs an end-of-convert signal. This end-of-convert signal is an input to the C30 chip as an interrupt request.

The PC program written in C language is a user interface program for the host PC which starts up the DSP program to interact with it and with the user. This program calls functions from the *High Level Language Interface Library* (provided by the TMS320C30 system board software package), which lessens the need for the user to learn hardware details of how to deal with the host interface mechanism to the TMS320C30. For instance, the library routines are used to download the TMS320C30 object code file onto DSP system board memory, to start execution of this code, and to pass data back and forth between the program running on the TMS320C30 and on the host PC.

A DSP program running on the system board and a program running on the host PC communicate with each other through a 64K words dual-ported on-board

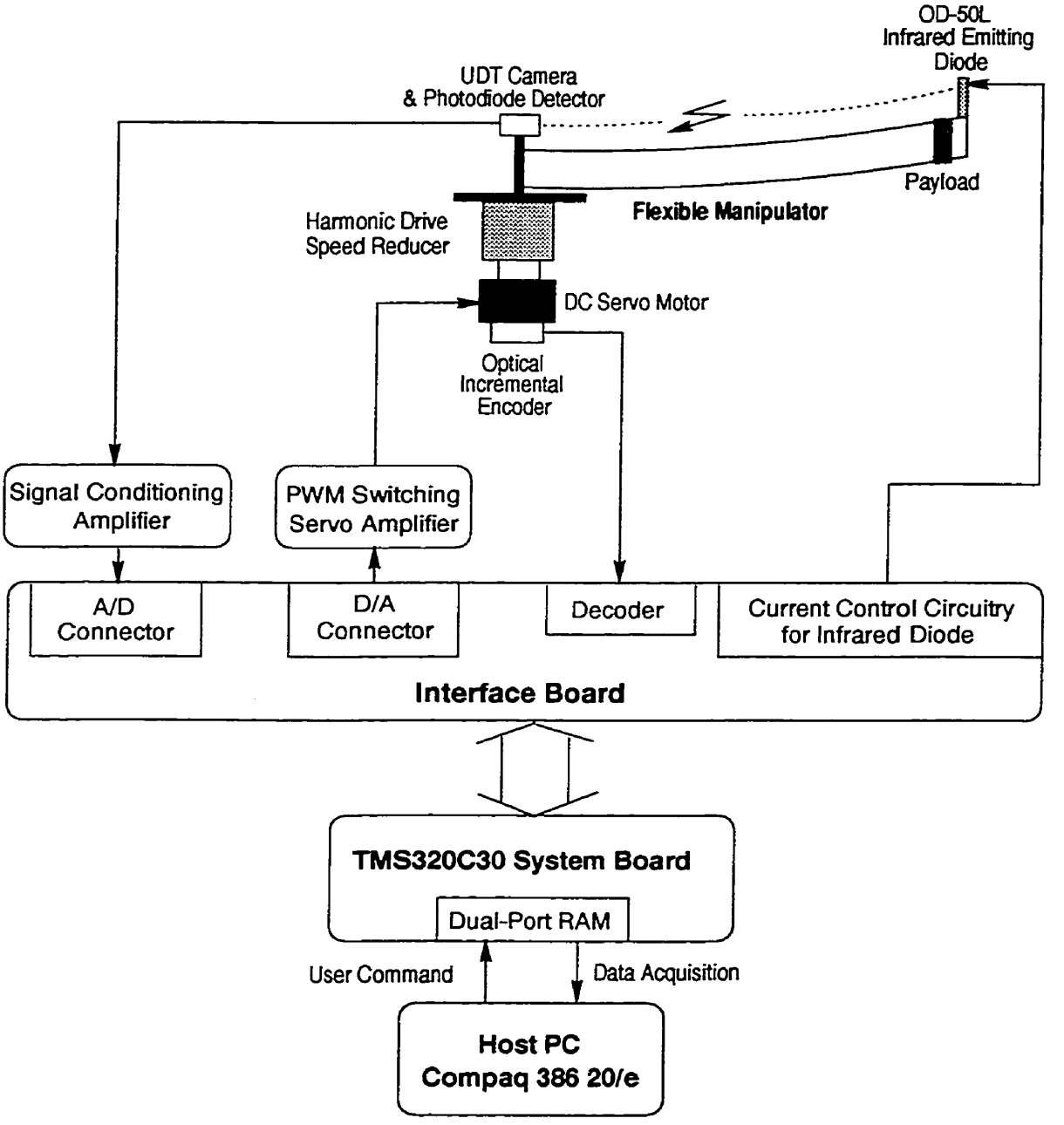

FIGURE 5.1. Configuration of the experimental test-bed for a single-link flexible manipulator

RAM. The two concurrent-running processors are controlled by using absolute-addressed communication flags. Specifically, PC-proceed-flags and DSP-proceed-flags are defined to guarantee that the data exchange properly between the DSP program and the PC program.

5.3 Digital Implementation of the Proposed Neural Network-based Controller

Using the previous hardware and software configuration, the proposed neural network-based control scheme is digitally implemented. During each sampling period $T_s = \frac{1}{200}$ *sec*, the TMS320C30 processor reads two signals, the joint position (θ) and the tip deflection (ψ) of the manipulator. The processor executes the control algorithm and generates a control signal (a voltage of $v(t)$) which is dispatched into the D/A converter port. The resulting analog control signal is sent to a pulse-width modulated (PWM) switching servo amplifier through the interface board. The amplifier serves as a transductance in which the input voltage $v(t)$ proportionally results in a output current $i_a(t)$, where $i_a(t) = K_a v(t)$ and $K_a = 2.0$. After being processed, this current signal $i_a(t)$ drives the DC servo motor which actuates the flexible-link manipulator through a harmonic drive speed reducer. At every sampling time, the control torque is updated according to the output signals of the physical plant and related desired signals in order to reach the steady-state.

5.3.1 Design Considerations

In all of the following experimental results, the parameter α in (2.3.9) for specifying the re-defined output is experimentally chosen as $\alpha = 0.4$ ([7]). This choice is obtained in order to have a minimum phase closed-loop system and to avoid transient oscillations that might exceed the sensor range.

With a PD-type control scheme, control gains are tuned as $K_p = 70$ and $K_d = 40$ for (3.1.1). For designing the inverse dynamics controller (Figure 3.1), it is assumed that the exact dynamic model (2.3.8) of the single-link flexible manipulator developed in *Section 2.3.2* is known precisely, so that the input-output (in this case, it is actually a *re-defined* output) linearization control strategy may be realized using (3.2.5). Two gains for the linearized closed-loop error dynamics (3.2.8) are chosen as $K_p = 35$ and $K_d = 35$.

The configuration of the proposed neural network-based controller is shown in Figure 3.4. Due to certain physical limitations in the experimental test-bed, such as the operational range of the actuator and the tip sensor limited range as described in *Section 5.1*, all weights of the two networks are initialized to zero to avoid possible early stage saturation of the sensor and the actuator. Due to the on-line constraints present for implementing these networks for the test-bed, only one hidden layer is selected for each network in contrast to the two hidden layer networks used in the simulation studies. Consequently, topologies of the two networks NN_1 and NN_2 are configured as $\mathcal{N}_{4,8,1}$ and $\mathcal{N}_{1,10,1}$, respectively. Other initialization issues related to the neural networks are as follows:

- Activation functions for all the hidden neurons are chosen as a bipolar *tanh*

function limited by $[-1,\ 1]$, and a linear activation function for the output neurons.

- The learning rate η for updating the weights of the two networks is chosen as $\eta = 0.8$.
- The two gains for the linear controller $V(\cdot)$ defined in (3.2.7) are chosen as $K_p = 0.6$ and $K_d = 0.6$.
- The values of $K_0 = 1.2$, $K_1 = 0.6$ and $K_2 = 0.6$ are assigned to the cost function defined in $J(\cdot)$ (3.3.14) which is to be minimized by the learning algorithm of the two networks.

5.3.2 Other Considerations

One flexible mode shape is used for the experimental measurements of the states of the manipulator. In other words, four states, namely joint position (θ) and its derivative ($\dot{\theta}$), first elastic mode (q) and its derivative ($\dot{q}$), are assumed to be fed back for the closed-loop control system. Since the above derivatives are not directly measurable from the test-bed, a first-order high-pass filter is designed to numerically approximate them from their respective positions. Considering the first mode of the flexible-link manipulator, the filter is expressed as follows ([47]),

$$G(z) = \frac{10s}{s+10}\bigg|_{s=\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}} = \frac{9.7561(1-z^{-1})}{1-0.9512z^{-1}}$$

Which is discretized by the bilinear transformation.

5.4 Results and Analysis

The performance results from the experimental implementation of the three controllers are shown in Figures 5.2 to 5.6.

For the PD-type control, applying a $0.2\,rad$ step reference trajectory results in an undershoot of 33.5%, an overshoot of 5%, a settling time of 6 *sec*, and slight oscillatory transients (see Figure 5.2(a)). By adding a $603\,g$ payload to the tip of the manipulator and using the same gains, an overshoot of 10%, a settling time of 9 *sec*, and a noticeable *steady-state error* with more oscillation during transient are obtained (see Figure 5.2(b)). When a $0.4\,rad$ step reference trajectory is applied to the system, the response becomes sluggish with undershoot and large oscillations, and initially a very large input torque is generated (refer to Figure 5.5(a)(c)(e)). In this case, when a $603\,g$ payload is added to the tip, the manipulator becomes **unstable.**

For the inverse dynamics control case, a $0.2\,rad$ step reference trajectory applied to the manipulator results in the tip response with some transient oscillations, a 33.5% undershoot, no overshoot, and a 4 *sec* settling time (shown in Figure 5.3(a)). Adding a $603\,g$ payload to the tip degrades the performance of the closed-loop system, i.e. little overshoot and longer oscillatory transient responses are obtained (refer to Figure 5.3(b)). When the magnitude of the step reference trajectory is increased to $0.4\,rad$, there is considerable undershoot and 8 *sec* of settling time (see Figure 5.5(b)(d)(f) solid line). With an additional payload, the transient behavior degrades even further (refer to Figure 5.5(b) dashed line).

For the proposed neural network-based control scheme, it can be seen from

Figure 5.4(a)(c)(e) that quite a satisfactory result for a $0.2\,rad$ step reference trajectory is achieved with $4\,sec$ of settling time, no overshoot and no steady-state error. When a $603\,g$ payload is added to the tip, the performance of the system remains practically unchanged (see Figure 5.4(b)(d)(f)). By increasing the magnitude of the step reference trajectory to $0.4\,rad$, the performance of the system (refer to Figure 5.6(a)(c)) remains almost the same as that shown in Figure 5.4(a)(c), with the exception that slightly bigger elastic deflections are present as compared with the results for a $0.2\,rad$ reference trajectory. By adding a $603\,g$ payload to the tip, we obtain a 10% overshoot (refer to Figure 5.6) which is within the standard acceptable performance specifications.

Control Scheme	**Ref.Traj.** (rad)	**Payload** (g)	**Settling Time**(s)	**Over-shoot**(%)	**Under-shoot**(%)	**Reference Figure**
PD-type	0.2	30	6	–	33.5	5.2(a)
		603	9	12.5	25	5.2(b)
	0.4	30	after10	25	50	5.5(a)
		603	–	–	–	unstable
Inverse Dynamics	0.2	30	4	–	35	5.3(a)
		603	6	5	25	5.3(b)
	0.4	30	8	–	10	5.5(b)*
		603	8	10	60	5.5(b)*
Neural Network-based	0.2	30	3.5	–	–	5.4(a)
		603	3	–	–	5.4(b)
	0.4	30	4	–	–	5.6(a)
		603	6	10	–	5.6(b)

$\star$ solid line, $*$ dashed line.

Table 5.1. Comparative experimental results for the three controllers with step reference trajectories

5.5 Summary

The proposed neural network-based control scheme was successfully implemented on the experimental test-bed in real-time. The performances achieved are substantially superior to those obtained from the PD-type controller and the conventional inverse dynamics controller. Quantitative comparison of these performances are shown in Table 5.1.

In summary, in addition to the satisfactory performance achieved (Figure 5.4(a)) with the proposed controller, robustness to the payload variations and different magnitudes of the reference trajectory are shown in Figures 5.4(b) and 5.6(a)(b). These achievements have validated the possibility of experimentally implementing the neural network-based controller for a flexible-link manipulator through on-line learning. The results also show that control based on the proposed learning scheme is feasible in real-time without a pre-training stage. Given the states of the process, and a training signal, a neural network may approximate the nonlinear functions of a given system. This can then provide the appropriate control effort to minimize the tip tracking error of the manipulator, and simultaneously suppress the arm deflections stemming from the elastic characteristics of the flexible-link manipulator.

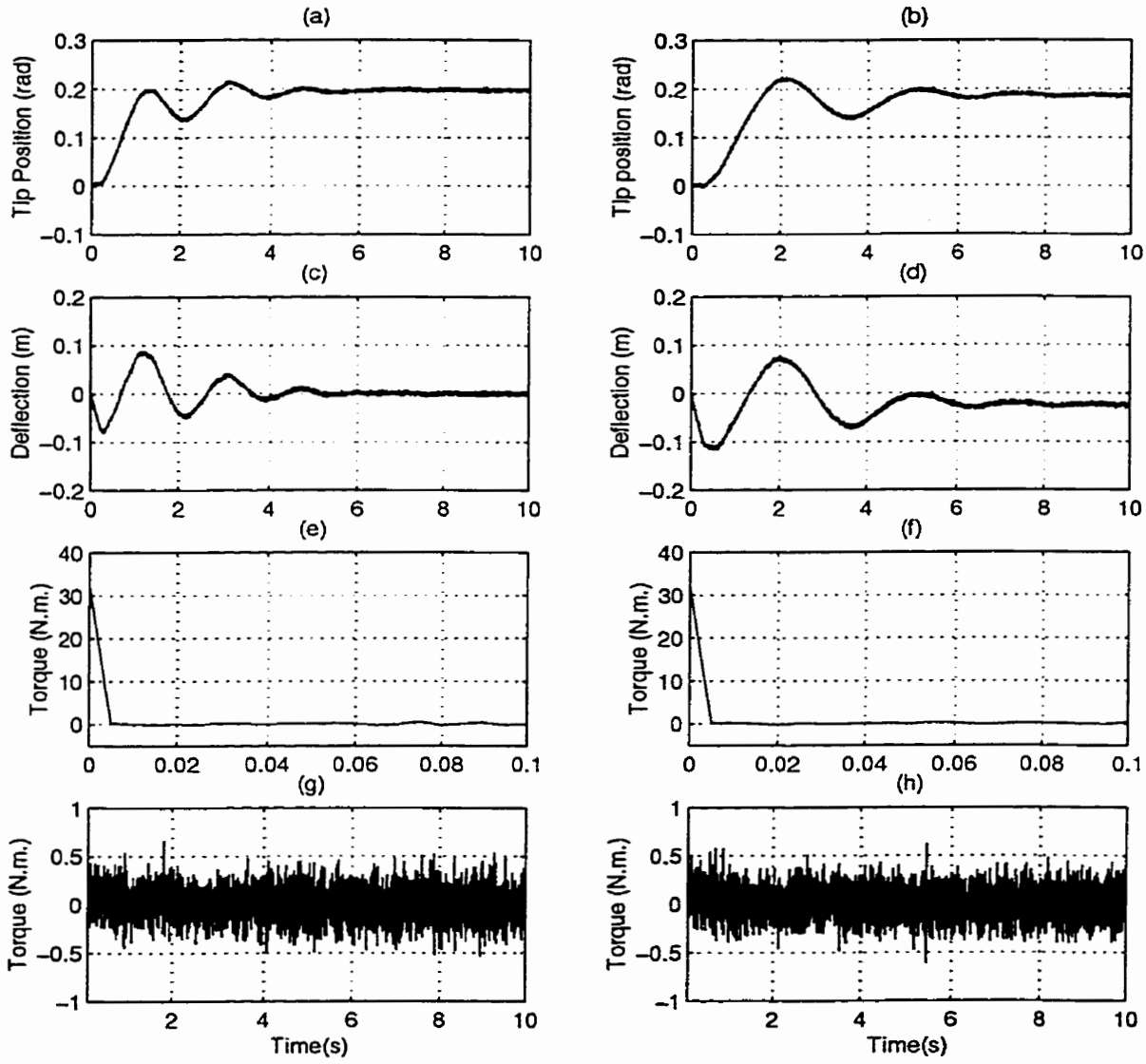

FIGURE 5.2. Experimental results for PD-type control with $0.2\,rad$ step reference trajectory. (a) (c) (e) (g) with payload $M_p = 30\,g$, (b) (d) (f) (h) with payload $M_p = 603\,g$. Note that in (e) and (f) the control is shown during the first 20 samples, and in (g) and (h) the control is shown for the remaining time

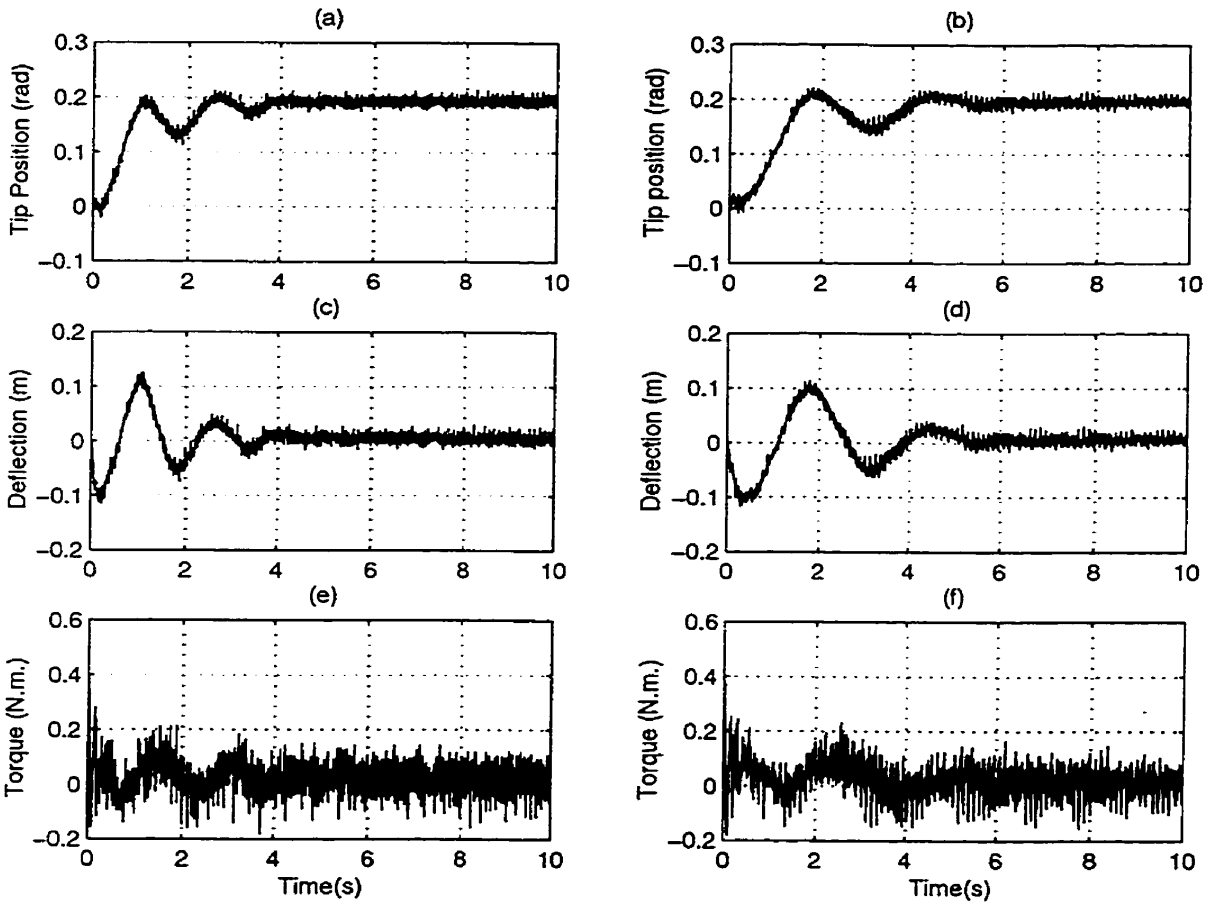

FIGURE 5.3. Experimental results for inverse dynamics control with $0.2\,rad$ step reference trajectory. (a) (c) (e) with payload $M_p = 30\,g$, (b) (d) (f) with payload $M_p = 603\,g$

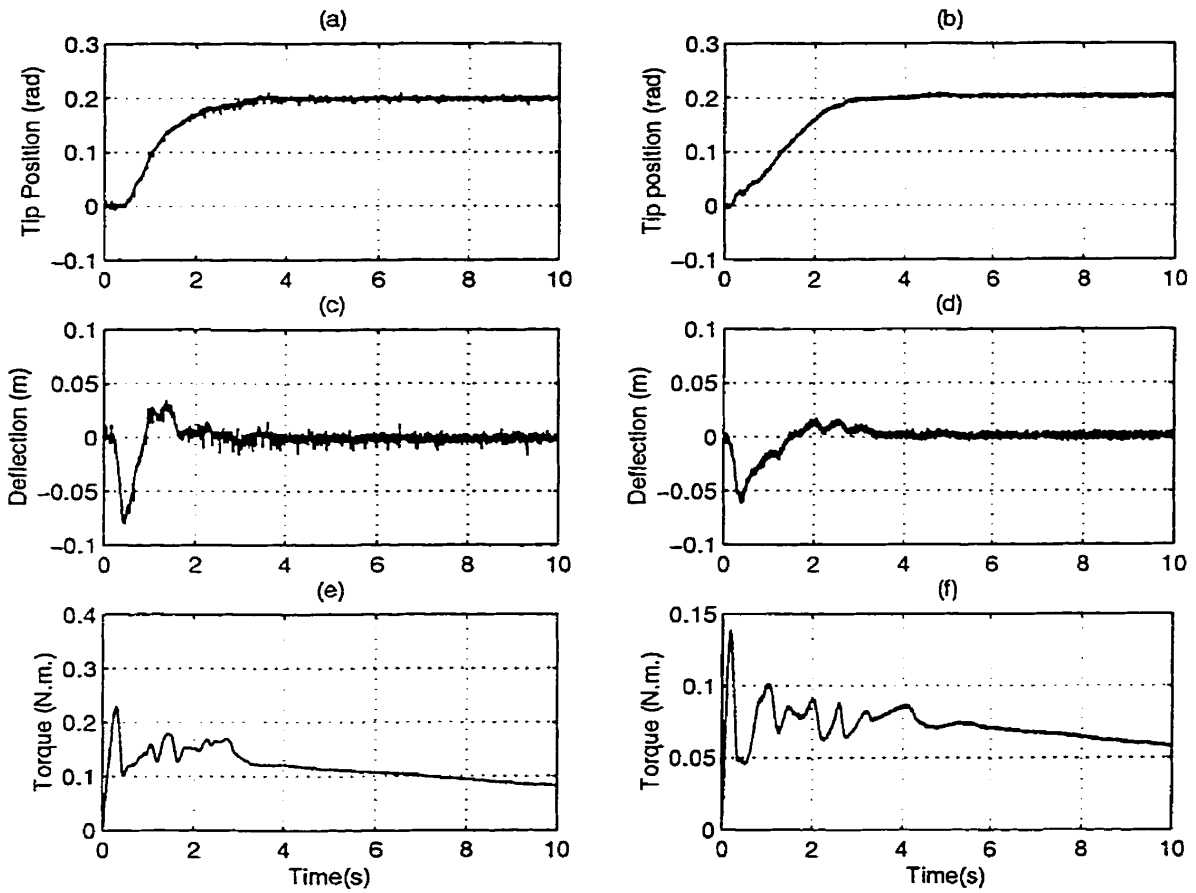

FIGURE 5.4. Experimental results for neural network-based control with $0.2\,rad$ step reference trajectory. (a) (c) (e) with payload $M_p = 30\,g$, (b) (d) (f) with payload $M_p = 603\,g$

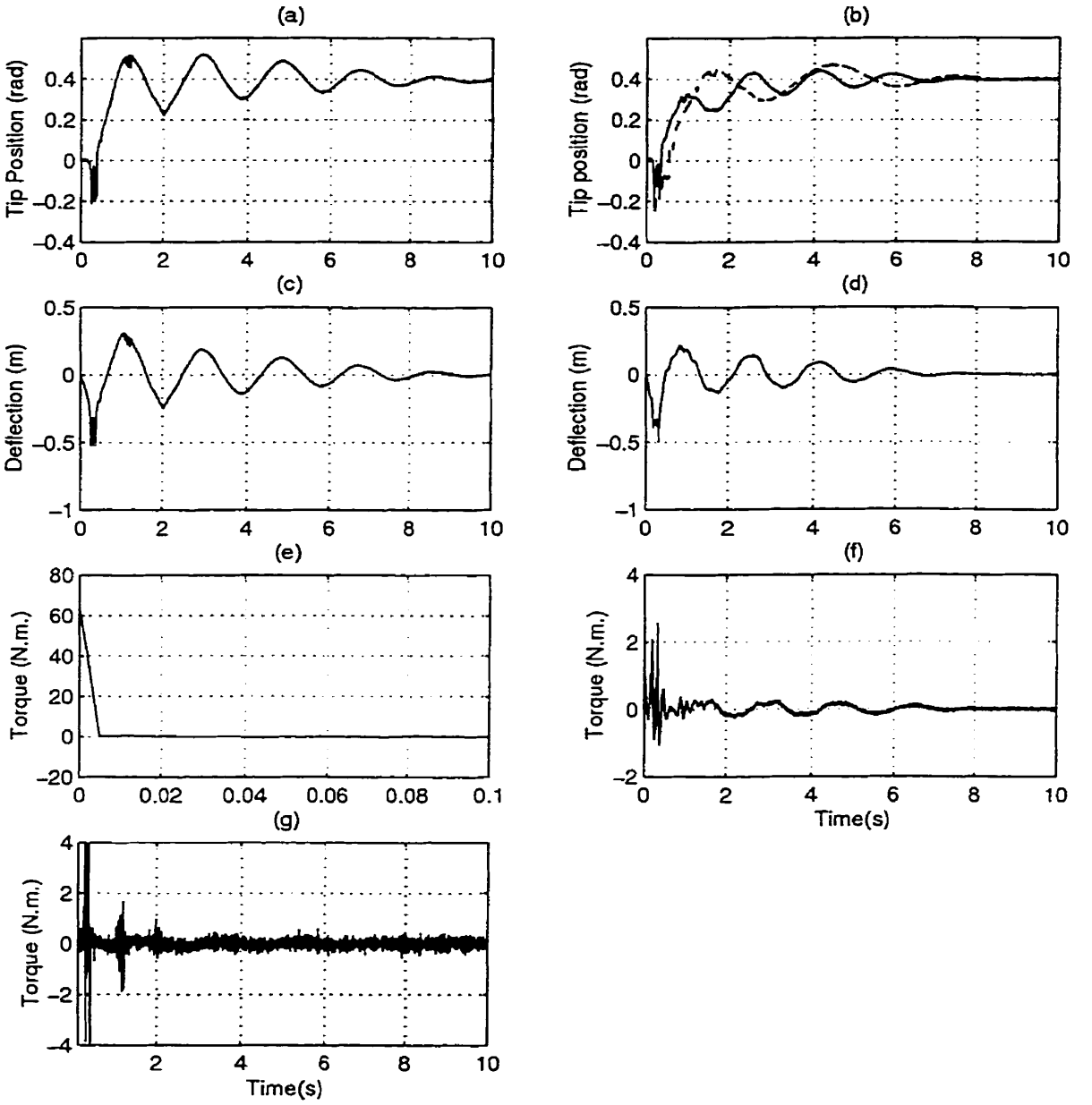

FIGURE 5.5. Experimental results with $0.4\,rad$ step reference trajectory. (a) (c) (e) PD-type control with payload $M_p = 30\,g$, (b)(solid line)(d) (f) inverse dynamics control with payload $M_p = 30\,g$, (b)(dashed line) with payload $M_p = 603\,g$. Note that in (e) the control is shown during the first 20 samples, and in (g) the control is shown for the remaining time

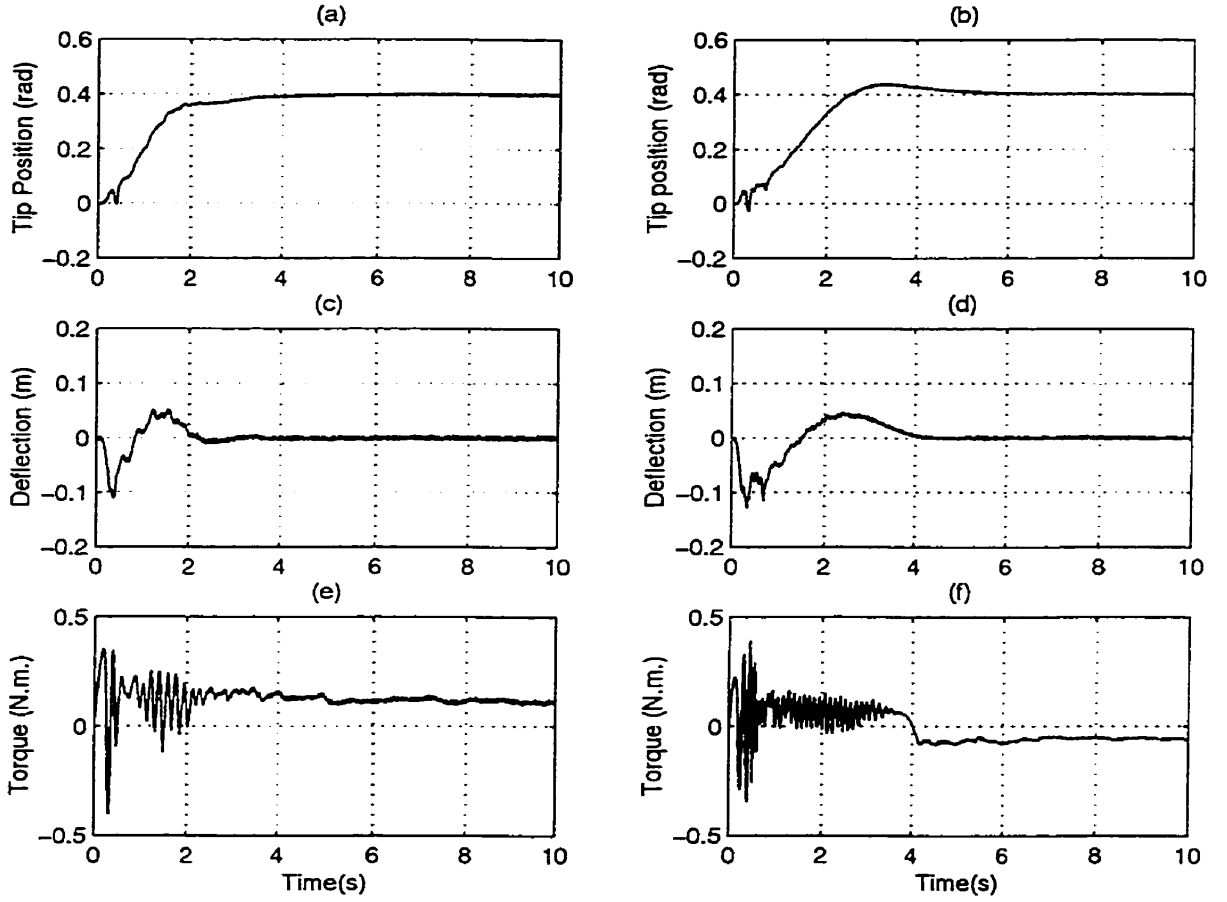

FIGURE 5.6. Experimental results for neural network-based control with $0.4\,rad$ step reference trajectory. (a) (c) (e) with payload $M_p = 30\,g$, (b) (d) (f) with payload $M_p = 603\,g$

Chapter 6

Concluding Remarks and Future Directions

6.1 Conclusions

The proposed neural network-based controller developed in this work, and which was motivated from the inverse dynamics structure using the re-defined output, was successfully tested both in a simulation environment as well as in a real-time implementation on a test-bed of a single-link flexible manipulator. The two neural networks were trained on-line and no off-line training was required to represent the nonlinearities of this highly nonlinear and complicated system. The proposed approach significantly improved the tip motion tracking precision of the manipulator through on-line learning. At the same time, the tip deflections were suppressed to the extent that was possible. These were the main objectives of this research.

To illustrate the superiority of the proposed intelligent control strategy, the

comparisons were conducted with two conventional control schemes, namely, PD-type control and nonlinear inverse dynamics control. As a result of the comparison the following conclusion were reached.

- Although the PD-type control is relatively simple, basic and linear, larger control gains have to be selected than those for inverse dynamics control. The control gains are even larger when compared with the neural network-based control scheme. This may be due to the fact that the only way the PD-type control tends to overcome the effects of the friction at the hub is by having large gains. The penalty one pays for higher gains is a larger torque of 30 $N.m.$ with the PD-type controller (Figure 5.2(e)), as compared with a torque generated by the proposed neural network-based scheme of only 0.23 $N.m.$ (see Figure 5.4(e)). This is usually an important issue and aspect to consider when a control system is going to be physically implemented, and torque capacity and energy consumption must be taken into account.

- When uncertainties and disturbances are present, for instance the payload on the tip is varying, or the amplitude of the input reference trajectory is changing, then the proposed neural network-based control scheme shows higher robustness to these variations than the other two conventional control strategies. This does validate and verify the advantage of the learning capabilities of the neural network-based scheme.

Although conventional control techniques may behave reasonably well within a certain operating region for this highly nonlinear system, in the experimental test-bed they showed an inability to handle uncertainties and disturbances. This is

especially true in the case of the PD-type control. The inverse dynamics controller may compensate to some extent for nonlinearities of the system at the expense of requiring an exact dynamic model of the system. In reality, the model may be complex with nonlinearities and the parameters that are often unknown or ill-defined. The parameters may also exhibit variations, or even may not appear linearly. However, the neural network-based approach shows promise in dealing with the above nonlinearities and uncertainties through weight adaptation and on-line learning which avoids the need for an extensive characterization and knowledge of the dynamics of the model being controlled.

Another conclusion which may be drawn from this research is that a certain knowledge of the dynamic model assisting the neural networks to achieve the control goal might be preferable to proceeding with a single neural network alone. It has been shown that, by incorporating the neural networks using the structure of a typical inverse dynamics approach, the limitations of the inverse dynamics scheme which completely depends on an accurate and exact dynamic model is reduced. In this work the neural networks are designed to cancel out the nonlinearities of the dynamic model. Accordingly, if the cancellation is sufficiently accurate, the system is no longer nonlinear. Therefore, the control design is simplified to designing a linear controller for the system.

In contrast to conventional control schemes, intelligent control is based on advanced computational techniques for reproducing "human" knowledge and experience. Thus, in intelligent control, the attention is moved away from the tedious task

of establishing an explicit, microscopic model of the controlled plant. The subsequent design of the corresponding controller is to emulate the cognitive mechanisms used by humans to make control decisions ([50]). In this research, it has been shown that the intelligent control is an effective means of enhancing conventional control schemes to produce more accurate, sophisticated, and cost-effective configurations of control to meet the ever-increasing demands of industry.

6.2 Future Directions

Control design for a flexible-link manipulator in order to operate in real-time with high precision, especially when the manipulator picks and places various payload masses, is an extremely important issue for industrial application. The payload masses concerned in this research are unknown but constant. An important related area for further research would be the investigation of the robustness of neural network-based control schemes with respect to a *time-varying* payload at the tip of a flexible-link manipulator. It would pose more challenges in the design of the control system.

It is well known that a flexible-link manipulator is a non-minimum phase dynamic system. Generally, researchers have overcome this problem in control design of this type of manipulators by defining a new output along the link. The aim is to end up with a minimum phase dynamics. But tradeoffs exist in locating this point along the link. Considering the work done by Talebi *et al.* [7] suggests that another important area for additional investigation would be achieving on-line minimum phase closed-loop system through neural network techniques based on the proposed

neural network-based control approach described in this thesis.

Bibliography

[1] S. Cetinkunt and W. J. Book, "Performance limitations of joint variable-feedback controllers due to manipulator structural flexibility," *IEEE Transactions on Robots and automation*, vol. 6, pp. 219–231, Apr. 1990.

[2] A. D. Luca, P. Lanari, P. Lucibello, S. Panzieri, and G. Ulivi, "Control experiments on a two-link robot with a flexible forearm," in *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), pp. 520–527, Dec. 1990.

[3] D. Wang and M. Vidyasagar, "Transfer function for a single flexible link," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1042–1047, 1989.

[4] A. D. Luca and B. Siciliano, "Trajectory control of a non-linear one-link flexible arm," *International Journal of Control*, vol. 50, no. 5, pp. 1699–1715, 1988.

[5] A. D. Luca and L. Lanari, "Achieving minimum phase behavior in a one-link flexible arm," in *Proceedings of International Symposium on intelligent Robotics*, (Bangalore, India), pp. 224–235, 1991.

[6] S. K. Madhaven and S. N. Singh, "Inverse trajectory control and zero dynamic sensitivity of an elastic manipulator," *International Journal of Robotics and Automation*, vol. 6, no. 4, pp. 179–191, 1991.

[7] H. A. Talebi, K. Khorasani, and R. V. Patel, "Neural network based control schemes for flexible-link manipulators: simulations and experiments," *Neural Networks*, vol. 11, no. 7, pp. 1357–1377, 1998.

[8] M. Moallem, R. V. Patel, and K. Khorasani, "An inverse dynamics control strategy for tip position tracking of flexible multi-link manipulators," *Journal of Robotic Systems*, vol. 14, no. 9, pp. 649–658, 1997.

[9] E. Bayo and H. Moulin, "An efficient computation of the inverse dynamics of flexible manipulators in the time domain," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 710–715, 1989.

[10] D. S. Kwon and W. J. Book, "An inverse dynamics method yielding flexible manipulator state trajectories," in *Proceedings of American Control Conference*, (San Diego, CA), pp. 186–193, 1990.

[11] R. V. Patel and P. Misra, "Transmission zero assignment in linear multivariable systems, part ii: The general case," in *Proceedings of American Control Conference*, (Chicago, IL), pp. 644–648, 1992.

[12] H. Geniele, R. V. Patel, and K. Khorasani, "End-point control of a flexible-link manipulator: Theory and experiments," *IEEE Transactions on Control Systems Technology*, vol. 5, pp. 556–570, Nov. 1997.

[13] B. Siciliano and W. J. Book, "A singular perturbation approach to control of lightweight manipulators," *The International Journal of Robotics Research*, vol. 7, pp. 79–90, Aug. 1988.

[14] K. Khorasani and M. W. Spong, "Invariant manifolds and their application to robot manipulators with flexible joints," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 978–983, 1985.

[15] M. W. Spong, K. Khorasani, and P. V. Kokotovic, "An integral manifold approach to the feedback control of flexible joint robots," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 291–300, Aug. 1987.

[16] K. Hashtrudi-Zaad and K. Khorasani, "Control of non-minimum phase singularly perturbed systems with application to flexible link manipulators," *International Journal of Control*, vol. 63, pp. 679–701, Mar. 1996.

[17] M. Moallem, K. Khorasani, and R. V. Patel, "An integral manifold approach for tip-position tracking of flexible multilink manipulator," *IEEE Transactions on Robotics and Automation*, vol. 13, pp. 1–15, Dec. 1997.

[18] S. Yurkovich and F. E. Pacheco, "On controller tuning for a flexible-link manipulator with varying payload," *Journal of Robotic Systems*, vol. 6, no. 3, pp. 233–254, 1989.

[19] P. Lucibello and F. Bellezza, "Nonlinear adaptive control of a two link flexible robot arm," in *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), pp. 2545–2550, Dec. 1985.

[20] M. R. Rokui and K. Khorasani, "Experimental results on discrete-time nonlinear adaptive tracking control of a flexible-link manipulator," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 30, pp. 151–164, Feb. 2000.

[21] B. Siciliano, B. S. Yuan, and W. J. Book, "Model reference adaptive control of a one link flexible arm," in *Proceedings of the 25th Conference on Decision and Control*, (Athens), Dec. 1986.

[22] J. Yuh, "Application of discrete-time model reference adaptive control to a flexible single-link robot," *Journal of Robotic Systems*, vol. 4, no. 5, pp. 621–630, 1987.

[23] K. Takahashi and I. Yamada, "Neural-network based learning control of flexible mechanism with application to a single-link flexible arm," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 116, pp. 792–795, 1994.

[24] A. Register, W. Book, and C. O. Alford, "Artificial neural network control of a nonominimum phase, single-flexible-link," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, (Minneapolis, MN), pp. 1935–1940, Apr. 1996.

[25] L. Lin and T. Yih, "Rigid model-based neural network control of flexible-link manipulators," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 595–601, Aug. 1996.

[26] J. S. Surdhar, A. S. White, and R. Gill, "Neural network control applied to a flexible link manipulator in a tip feedback sensor based configuration," in

Proceedings of the 20th International Conference on CAD/CAM Robotics and Factories of the Future, (London, UK), pp. 14–16, Aug. 1996.

[27] V. Zeman, R. V. Patel, and K. Khorasani, "Control of a flexible-joint robot using neural networks," *IEEE Transactions on Control Systems Technology*, vol. 5, pp. 453–462, July 1997.

[28] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," *Neural Networks*, vol. 1, pp. 251–265, 1988.

[29] H. Gomi and M. Kawato, "Neural network control for a closed-loop system using feedback-error-learning," *Neural Networks*, vol. 6, pp. 933–946, 1993.

[30] R. T. Newton and Y. Xu, "Neural network control of a space manipulator," *IEEE Control Systems Magazine*, vol. 12, pp. 14–22, 1993.

[31] H. Geniele, "Control of a flexible-link manipulator," Master's thesis, Concordia University, Montreal, Canada, Aug. 1994.

[32] M. O. Tokhi and A. K. M. Azad, "Modeling of a single-link flexible manipulator system: theoretical and practical investigations," *Robotica*, vol. 14, pp. 91–102, 1996.

[33] W. J. Book, "Recursive lagrangian dynamics of flexible manipulator arms," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 87–101, 1984.

[34] R. H. Cannon and E. Schmitz, "Initial experiments on the end-point control of a flexible one-link robot," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 62–75, 1984.

[35] S. Cetinkunt and W. Yu, "Closed-loop behavior of a feedback-controlled flexible arm: A comparative study," *The International Journal of Robotics Research*, vol. 10, pp. 263–275, June 1991.

[36] R. B. Usoro, r. Nadira, and S. S. Mahil, "A finite element/lagrange approach to modeling lightweight flexible manipulators," *Transactions of the ASME Journal of dynamic Systems, Measurement and Control*, vol. 108, pp. 198–205, 1986.

[37] F. Raksha and A. A. Goldenberg, "Dynamic modeling of a single-link flexible robot," in *Proceedings of IEEE International Conference on Robotics and Automation*, (San Francisco, CA), pp. 918–924, Apr. 1986.

[38] E. Bayo, "A finite-element approach to control the end-point motion of a single-link flexible robot," *Journal of Robotic Systems*, vol. 4, no. 1, pp. 63–75, 1987.

[39] M. O. Tokhi, M. A. Hossain, and A. K. M. Azad, "Sequential and parallel real-time simulation of a flexible manipulator system," *Robotica*, vol. 16, pp. 445–456, 1998.

[40] V. R. Vemuri, ed., *Artificial Neural Networks: Concepts and Control Applications*. IEEE Computer Society Press, 1992.

[41] T. Poggio and F. Girosi, "Networks for approximation and learning," *in Proceedings of the IEEE*, vol. 78, pp. 1481–1497, Sept. 1990.

[42] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[43] B. Irie and S. Miyake, "Capabilities of three-layered perceptrons," in *IEEE International Conference on Neural Networks*, vol. 1, pp. 641–648, 1988.

[44] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[45] D. L. Chester, "Why two hidden layers are better than one," in *Proceedings of IJCNN International Joint Conference on Neural Networks*, vol. 1, pp. 265–268, 1990.

[46] G. Lightbody, W. H. Wu, and G. W. Irwin, "Control application for feedforward networks," in *Neural Networks for Control* (T. W. Miller and et al, eds.), pp. 51–71, Cambridge, MA: MIT Press, 1990.

[47] H. A. Talebi, *Neural network-based control of Flexible-Link Manipulators.* PhD thesis, Concordia University, Montreal, Canada, 1997.

[48] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms and Applications.* Prentice-Hall Inc., 1994.

[49] M. H. Hassoun, *Fundamentals of Artificial Neural Networks.* MIT Press, 1995.

[50] R. E. King, ed., *Computational Intelligence in Control Engineering.* Marcel Dekker, Inc., 1999.

[51] W. J. Book, "Modeling, design, and control of flexible manipulator arms: A tutorial review," in *Proceedings of the 29th Conference on Decision and Control*, (Honolulu, Hawaii), pp. 500–506, Dec. 1990.

[52] M. Moallem, K. Khorasani, and R. V. Patel, "Inversion-based sliding control of a flexible-link manipulator," *International Journal of Control*, vol. 71, no. 3, pp. 477–490, 1998.

[53] S. S. Ge, T. H. Lee, and G. Zhu, "Improving regulation of a single-link flexible manipulator with strain feedback," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 179–185, 1998.

[54] R. Brogliato, D. Rey, A. Pastore, and J. Barnier, "Experimental comparison of nonlinear controllers for flexible joint manipulators," *International Journal of Robotics Research*, vol. 17, no. 3, pp. 260–281, 1998.

[55] D. Psaltis, A. Sideris, and A. A. Yamanura, "A multilayered neural network controller," in *Proceedings of the IEEE International Conference Neural Networks*, (San Diego, CA), pp. 500–506, June 1987.

[56] A. Yesildirek and F. L. Lewis, "Feedback linearization using neural networks," *Automatica*, vol. 31, no. 11, pp. 1659–1664, 1995.

[57] L. Behera, M. Gopal, and S. Chaudhury, "On adaptive trajectory tracking of a robot manipulator using inversion of its neural emulator," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1401–1414, Nov. 1996.

[58] R. T. Newton and Y. Xu, "Real-time implementation of neural network learning control of a flexible space manipulator," in *Artificial Neural Networks: Concepts and Control Application* (V. R. Vemuri, ed.), pp. 150–156, Los Alamitox, CA: IEEE Computer Society Press, 1996.

[59] M. M. Polycarpou and P. A. Ioannou, "Neural networks as on-line approximators of nonlinear systems," in *Proceedings of the 31th Conference on Decision and Control*, (Tucson, Arizona), pp. 7–12, Dec. 1992.

[60] A. I. Mistry, S. Chang, and S. S. Nair, "Indirect control of a class of nonlinear dynamic systems," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1015–1023, July 1996.

[61] O. Omidvar and D. L. Elliott, eds., *Neural Systems for Control.* Chestnut Hill, MA: ACADEMIC PRESS, 1997.

[62] C. T. Leondes, ed., *Industrial and Manufacturing Systems*, vol. 4 of *Neural Network Systems Techniques and Applications.* San Diego, CA: ACADEMIC PRESS, 1998.

[63] A. Isidori, *Nonlinear Control Systems.* Springer-Verlag London Limited, third ed., 1995.

[64] C. C. de Wit, B. Siciliano, and G. Bastin, eds., *Theory of Robot Control.* Springer-Verlag London Limited, 1996.

[65] M. Moallem, *Control and Design of Flexible-Link Manipulators.* PhD thesis, Concordia University, Montreal, Canada, Dec. 1996.

[66] Z. Su and K. Khorasani, "A neural networks controller for a single-llink flexible manipulator based on the inverse dyanmics structure," in *Proceedings of IJCNN International Joint Conference on Neural Networks*, vol. 5, (Como, Italy), pp. 311–316, July 2000.

[67] K. Warwick, G. W. Irwin, and K. J. Hunt, eds., *Neural networks for control and systems.* Herts, UK: Peter Peregrinus Ltd., 1992.

[68] P. K. Simpson, ed., *Neural Networks Applications.* Piscataway, NJ: IEEE Technical Activities Board, 1996.

[69] J.-J. E. Slotine and W. Li, eds., *Applied Nonlinear Control.* Englewood Ciffs, NJ: Prentice Hall, Inc., 1991.

[70] F. L. Lewis, C. T. Abdallah, and D. M. Dawson, eds., *Control of Robot Manipulators.* New York, NY: Macmillan Publishing Company, 1993.

[71] K. Ogata, ed., *Modern Control Engineering.* Upper Saddle River, NJ: Prentice Hall, Inc., third ed., 1997.

[72] J. M. Zurada, ed., *An Introduction to Artificial Neural Systems.* St. Paul: West Publishing, 1992.

[73] G. Gratzer, ed., *Math into LaTeX: An Introduction to LaTeXand AMS-LaTeX.* Birkhauser Boston, 1996.

[74] SPECTRUM Signal Processing Inc., *TMS320C30 SYSTEM BOARD: Technical Reference Manual*, Aug. 1990. Issue 1.01.

[75] SPECTRUM Signal Processing Inc., *TMS320C30 SYSTEM BOARD: User's Manual*, Aug. 1990. Issue 1.01.