INFORMATION TO USERS

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

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

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

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

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

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UNIVERSITY OF ALBERTA

Adaptive Fuzzy Network with Application to Neural Prosthetic Control: A Computer Simulation Study

BY

Feng Wang

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

DEPARTMENT OF BIOMEDICAL ENGINEERING

Edmonton, Alberta
Spring 1999

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Feng Wang

TITLE OF THESIS: **Adaptive Fuzzy Network with Application to Neural Prosthetic Control: A Computer Simulation Study**

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1999

(Signed) F. Wang

Feng Wang
8903 112 Street, Apt.3A
Edmonton, Alberta
Canada, T6G 2C5

Date: Jan. 5, 1996

"Whatsoever Things Are True"

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Adaptive Fuzzy Network with Application to Neural Prosthetic Control: A Computer Simulation Study** submitted by Feng Wang in partial fulfillment of the requirements for the degree of Master of Science.

B. Andrews
Dr. B.J. Andrews (Supervisor)

A.W. Lipsett
Dr. A.W. Lipsett (External)

R.E. Snyder
Dr. R.E. Snyder (Examiner)

Z.J. Koles
Dr. Z.J. Koles (Examiner)

D.J. Griffiths
Dr. D.J. Griffiths (Chair)

Date: 3 Jan 1996

This thesis is dedicated to my family

Abstract

In this thesis, the author developed an Adaptive Fuzzy Network (AFN) with supervised and reinforcement learning mechanisms. The AFN was applied to the adaptive finite state control of the computer simulated swing phase of paraplegic locomotion. The swinging leg was modeled as a compound pendulum with electrically stimulated quadriceps and a powered hip brace.

It was found that the supervised learning controller was able to mimic a previously optimized open-loop controller after approximately ten training trials and could use sensor feedback to cope with slight parameter variations which caused the open-loop controller to fail. The reinforcement learning controller was able to learn a similar optimal control strategy using only evaluative reinforcements and could re-adapt to new models with significant parameter variations within tens of trials after previously learned control strategy failed. Starting from no experience, the reinforcement learning controller required hundreds of trials to succeed. However, incorporating a priori knowledge greatly accelerated the learning process, which is important for clinical implementation.

Preface

This thesis is the outcome of two-year graduate study and research from 1993 to 1995 in the Department of Biomedical Engineering at the University of Alberta. Part of the research was carried out at the Rehabilitation Research Center of Glenrose Rehabilitation Hospital and at the Alberta Microelectric Center. The following is a brief outline of the thesis:

Chapter 1 is an introduction to biological motor control systems, the role of the spinal cord in motor control, Spinal Cord Injury (SCI), and the Functional Neuromuscular Stimulation (FNS) technique.

Chapter 2 is a literature review of adaptive nonlinear control in general and control strategies for FNS in particular. Based on this review, my initial hypothesis for this thesis was proposed.

Chapter 3 describes the methods, including the computer model of the swinging leg used in the computer simulation, and the adaptive fuzzy network. Section 3.1 describes the computer model of the FNS-induced motion of the swinging leg; Section 3.2 describes the adaptive fuzzy logic system, in which fuzzy set theory was introduced, and the Adaptive Fuzzy Network (AFN) was developed. The supervised learning algorithm for the AFN was formulated using the mean squares error as optimal object function and a gradient algorithm for updating internal parameters. A test of the AFN's ability to approximate a function was verified using a nonlinear function; Reinforcement learning is introduced, including the REINFORCE and the TD algorithms, as well as the relationship between Reinforcement Learning and Dynamic Programming. This is followed by the formulation of the specific reinforcement learning algorithm for the AFN.

Chapter 4 represents the results of computer simulations demonstrating the adaptive control of the FNS swinging leg. The simulations include supervised learning, reinforcement learning, and a combination of both techniques.

Chapter 5 is the discussion of results, and the conclusions.

Chapter 6 is suggestions for further work.

In the Appendixes, Details of swinging leg model are given.

The thesis was prepared with LaTeXtypesetting software and GNU Emacs editor on Sun workstations. [1]

Feng Wang

December, 1995

Edmonton, Alberta, Canada

[1]To meet the required thesis format of graduate school, LaTeXdocument style macro thesis.sty designed by Randal Peters (1992) and Fahiem Bacchus (1987) is used. Figures are either directly exported from MATLAB simulation software or scanned from references using HP ScanJet, and then processed by XFIG, XPAINT and XV, CorelDRAW software. Thanks to Free Software Foundation and other software developers for providing these excellent softwares.

Acknowledgements

I would like to thank the following people for their help with my graduate study at UofA and the preparation of this thesis:

Dr. Brian J. Andrews for supervising my graduate program and providing access to the resources of his research projects; Jeroen Bielen for providing his FNS swing leg model software; Maisie Goh for reminding me all those requirements of graduate school; All my colleagues in FNS research group: RonChing Dai, Adam Thrasher, Rahman Davoodi, Aleksandar Kostov, for the helpful discussions on FNS, muscle model, adaptive fuzzy logic control, machine learning; Adam Thrasher and Richard Williamson for proof-reading the thesis draft; The Internet community for many helpful and enjoyable discussions ranging from synchronization of data acquisition with video signal, to the relationship between traditional Chinese Taoism and the Oscar-winning movie "Forrest Gump".

The support of the Alberta Heritage Foundation for Medical Research (AHFMR) is acknowledged and the author was a recipient of the UofA graduate assistantship.

Last, but not the least, I would like to thank my family for being so supportive when I studied in Canada.

Contents

List of Figures

Abbreviations

AFN: Adaptive Fuzzy Network

BP: BackPropagation

FES: Functional Electrical Stimulation

FL,FLC: Fuzzy Logic, Fuzzy Logic Control/Controller,

FNS: Functional Neuromuscular Stimulation

FRO: Floor Reaction Orthosis

LMS: Least Means Squares

LS,RLS: Least Squares, Recursive Least Squares algorithms

MF: Membership Function

MIMO: Multiple Inputs Multiple Outputs

MISO: Multiple Inputs Single Output

MIT rule: a heuristic adaptive control rule proposed by researchers in Massachusetts Institute of Technology

NN,ANN: Neural network, Artificial Neural Network

PD,PID: Proportional-Integral-Derivative, Proportional-Derivative

RBF,RBFN: Radial Basis Function, Radial Basis Function Network

RL: Reinforcement Learning

SCI: Spinal Cord Injury, Spinal Injured

SISO: Single Input Single Output

SL: Supervise Learning

SOC: Self-Organization-Controller

SOM: Self-Organizing-Map, a kind of clustering algorithm.

TD: Temporal Difference

Chapter 1

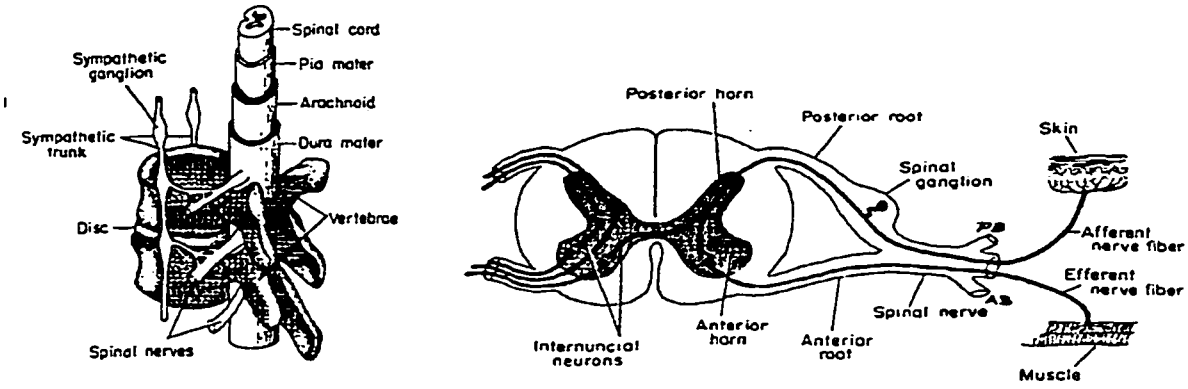
Introduction

1.1 Human Motor Control and the Spinal Cord

Walking is an important movement in human daily life. The biological motor control system is a sophisticated, hierarchical and distributed system [Ste80, Pro93, Ito84, BSB93, KFS87, Alb81, Bro75, BRM93]. The nervous system falls into two major divisions, the central and the peripheral. The central nervous system (CNS) includes the brain (cerebrum and cerebellum) and the spinal cord (Fig. 1.2B). The spinal cord is an extension of the brain and also consists of white matter that is bundles of nerve fibres, and grey matter which contains the nerve cell bodies and endings where the synapses take place (Fig.1.1B). The spinal cord lies within the spinal canal, which is formed posteriorly by the vertebral bodies and anteriorly by the neural arches of the vertebrae (Fig.1.1(A)). The vertebrae are divided into four groups–from top to bottom– cervical, thoracic, lumbar and sacral. Abbreviated names are commonly used, e.g., T4 refers to the fourth thoracic vertebra. Different spinal nerve groups control different muscle groups, as indicated in Fig. 1.2.

In this hierarchical neural motor control system, sensory signals and motor commands pass between sensory organs, muscles, spinal cord, cerebellum, sensory cortex, and motor cortex. The cerebral motor cortex (situated in the gyrus immediately anterior to the central fissure of Rolando) provides high-level motor planning and control. The cerebellum is the middle level motor learning and control center [Mar69, Alb81, Ito84] which performs sub-consciously. The spinal cord is the low level control center where many preset motor programs such as the stretch reflex and Central Pattern Generator exist.

The upper motor neurons arise in several different areas of the brain (but mostly in



(A) (B)

Figure 1.1: Sectional view of spinal vertebrae and cord. (A)Spinal cord lies inside three meninges and vertebral canal. (B)Diagram showing afferent and efferent fibers, interneurons, H-sharped area of gray matter, and reflex arc inside the cord. (from [Bro75])

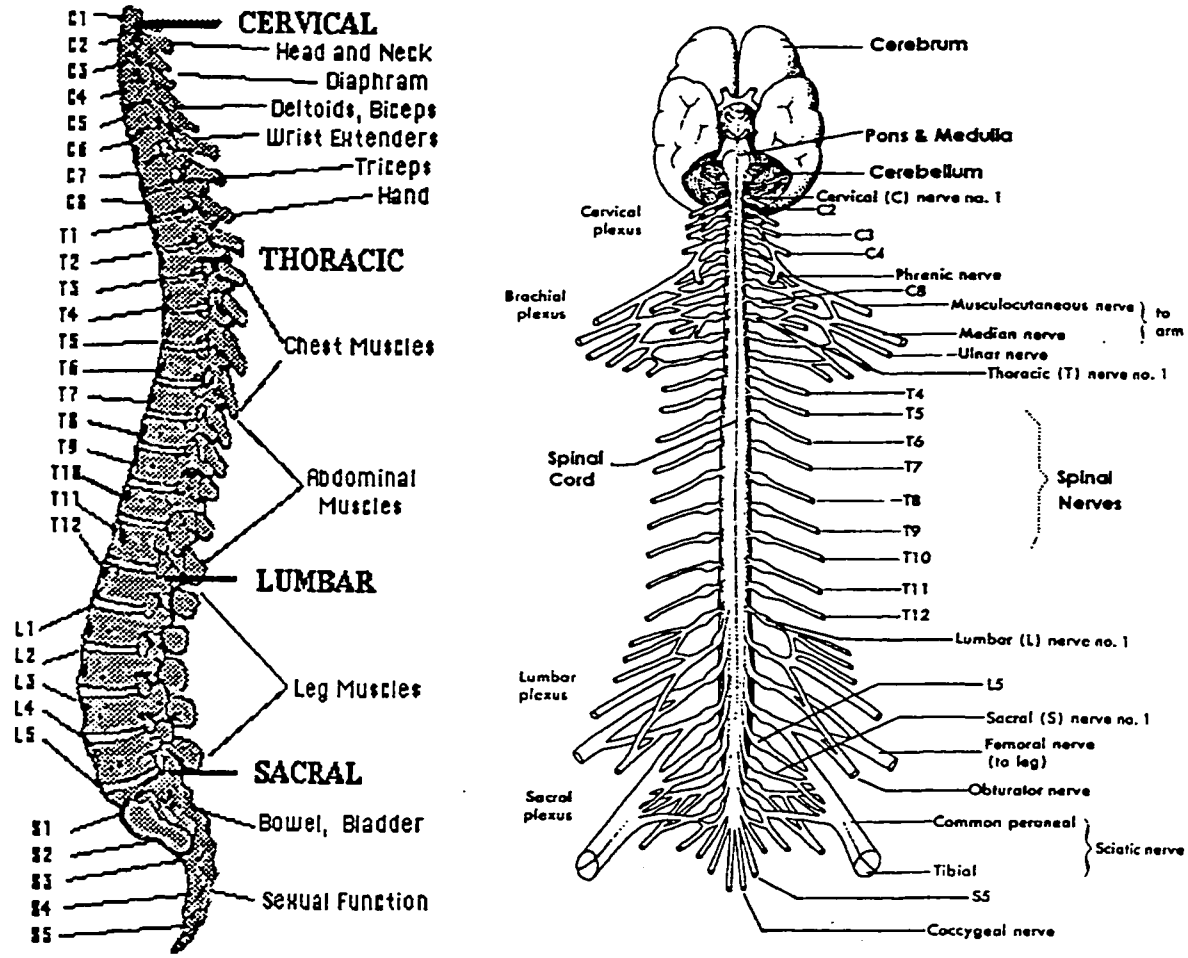
the motor cortex), pass down the descending tracts of the spinal cord, and cross over to the other side before exiting the spinal roots. Within the grey matter of the anterior horn, the upper motor neurons synapse with the lower motor neurons, as well as with a large number of inter carry many different types of sensory information like the familiar sensations of touch, temperature, pain and vibration, as well as "proprioceptive" signals used in feedback control of limbs. These signals include the positions of the joints and the tension in the muscles and ligaments.

Figure 1.3 shows descending motor tract and ascending sensory tract passing between motor cortex, sensory cortex, thalamus, spinal cord, muscles, and sensory nerves. Figure 1.1(B) shows the afferent, efferent neurons, and reflex arc inside spinal cord.

1.2 Spinal Cord Injury

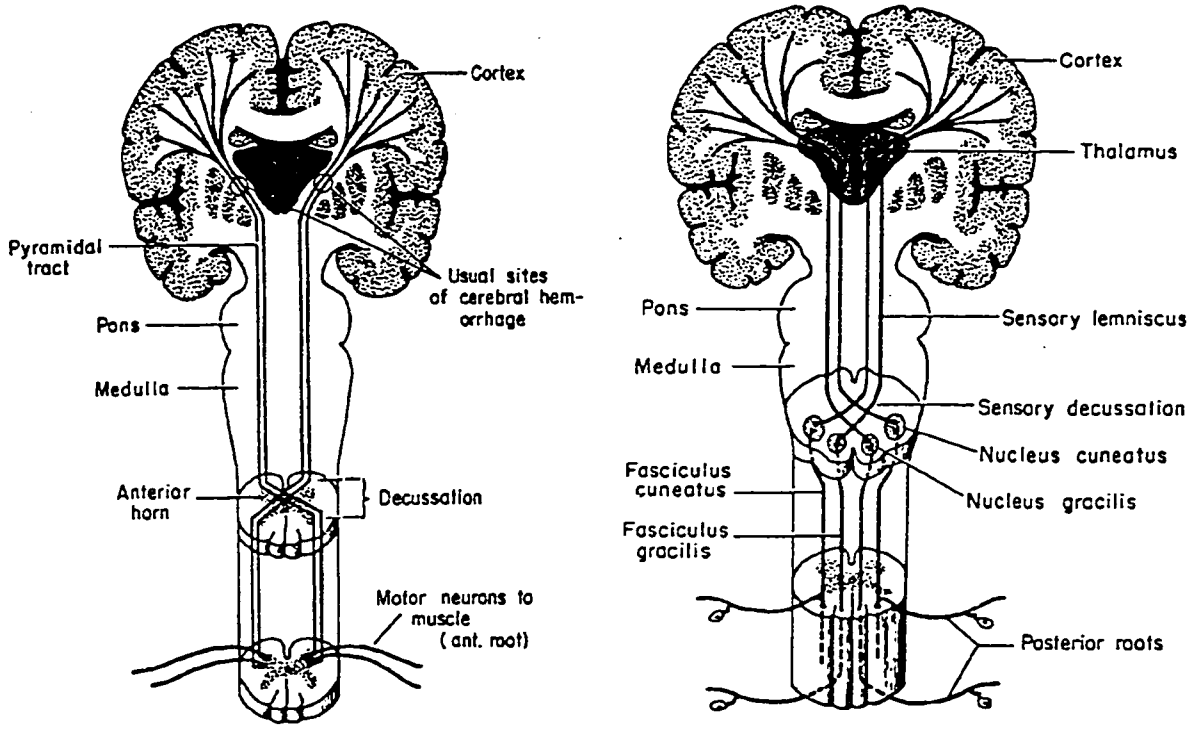
If the spinal cord is damaged by accident or disease, the result can be a wide variety of motor disabilities depending on the lesion level and the degree of injury. The abbreviation SCI (Spinal Cord Injury, or Spinal Cord Injured) refers to various spinal injuries or those people whose spinal cords are injured.

People with spinal injuries at the level of the cervical spine are 'quadriplegic', since both the arms and legs are affected. If the lesion is above C4, the diaphragm is also affected, making breathing difficult or impossible, and the chance of survival poor. At the lower



(A) (B)

Figure 1.2: Whole view of spinal vertebrae and cord. (A)Lateral view of spinal vertebrae, with nomenclature of four groups (GIF from [CPN94]). (B) Ventral view of cord, showing different spinal nerves controlling different muscle groups (scan from[Bro75]).



(A) (B)

Figure 1.3: Motor and sensory tracts in spinal cord. (A)Descending pyramidal motor tract from motor cortex to efferent neurons. (B)Sensory ascending tract from afferent neurons, through the sensory information pre-processing hub Thalamus, to the sensory cortex. (scan from[Bro75])

cervical levels, some arm and hand functions are preserved. If the spinal cord is damaged in the thoracic or lumbar regions, only the legs are affected and the individual is said to be "paraplegic". Paraplegics with lesions in the middle to low thoracic range (T5-T12) have full upper limb function and a degree of control of the trunk musculature (important for posture and gait), depending on the lesion level. It is these paraplegics that are suitable for the restoration of standing and walking with present electrical stimulation technologies. The spinal cord terminates at the S1 vertebral level, below this level there are only lower motor neurons. Injury at or below S1 level will thus result in damage of these lower motor neurons, leading to degeneration and denervation of muscle. Denervated muscle cannot be excited by electrical stimulation sufficient to produce functional movement.

According to the degree of injury, SCI can also be classified as "complete SCI" with a complete loss of sensory or motor functions below the level of lesion; or "incomplete SCI" with different degree of preserved sensory or motor functions below the level of lesion.

It is estimated that in U.S.A. alone, there are 238,000 SCI people and 11,000 new ones per year [Erg85] and most SCI are young males (Fig. 1.4). In addition, there are millions of partly or wholly immobilized individuals, including about 2.5 million affected by stroke and 1.8 million by head injuries [Kob94]. Rehabilitation technology now enables SCI individuals to live a normal life-span, and care costs exceed 1 million $US for each SCI individual. Advances have been made in various approaches to enhance locomotor recovery after SCI, including pharmacotherapy and electrical stimulation [BR94]. Spinal cord regeneration (partially or fully) may provide the ultimate cure for SCI, but for now it is a distant hope [Faw92]. On the other hand, currently available rehabilitation techniques such as wheelchairs, mechanical bracing, and electrical stimulation can already provide SCI individuals with limited mobility, physical and psychological rehabilitation. It is argued [BR94] that a combination of the various approaches will provide an optimal base for functional locomotor recovery after SCI.

1.3 Function Neuromuscular Stimulation

Although SCI individuals can not voluntarily control their muscles, the motor units (motor neuron with controlled muscles) are still intact, and only the supra-spinal connection to the brain is damaged. The FES (Functional Electrical Stimulation) technique can be used to

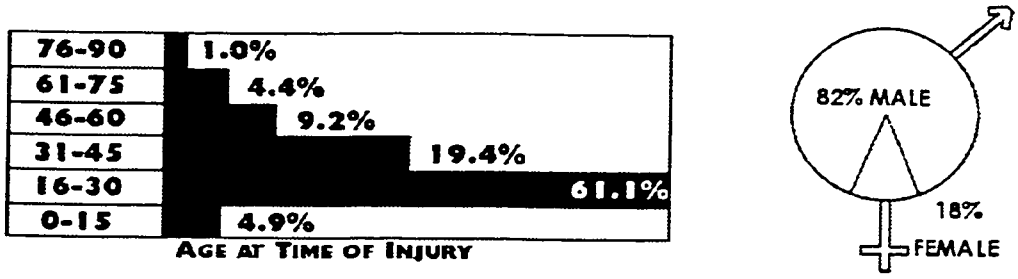


Figure 1.4: SCI statistical figure. Majority of SCI individuals are (A) young (61.1% is aged 16-30). and (B) males (82%). (GIF picture from [CPN94])

stimulate the neuromuscular system to assist locomotion in individuals with SCI. FES has been applied to restore movement. hearing. vision. cardiac pacemaking and bladder control [Loe89. HH88]. Here the term FNS (Functional Neuromuscular Stimulation) will be used to emphasize the application to the neuromuscular systems to assist functional movement.

FNS is based on the excitability of human nerves and muscle fibres [Ste80]. Under the influence of an external electrical field generated by cathode (negative) and anode (positive) electrodes (Fig. 1.5). the excitation of nerve and muscle is induced by ionic transport across cell tissue membranes. as described by the Huxley and Hodgkin Equations [Ste80]. Beneath the active (cathode) electrode. positive ions (mostly sodium and some potassium) are attracted while negative ions (principally chloride) are simultaneously repelled onto the nerve membrane. This results in the depolarization of the nerve membrane which has positive potential outside and negative potential inside at resting state. An anode or indifferent electrode is required to form a close circuit for the ionic current. When externally induced depolarization reaches a threshold, spontaneous depolarization occurs and an all-or-none "action potential" is generated in the nerve.

The forces generated by muscles are regulated by the firing rate of action potentials in each motor unit, and the recruitment of motor units within the motor neuron pool according to their cell body size. The "Size Principle" [Ste80] states that: if a stimulus is applied directly on the nerve fibre membrane, then the larger diameter fibre requires a higher threshold for initiating action potentials. This is because the membrane area is proportional to fibre diameter, thus a stronger stimulus is required to stimulate larger membrane area. In the normal intact neuromuscular system, the synaptic electric current that excites fibres is applied directly on membranes, and thus will follow the size principle.

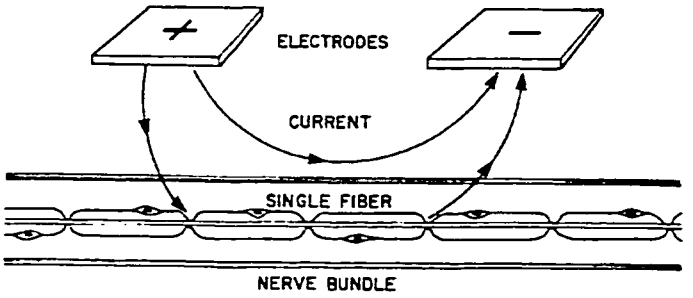


Figure 1.5: Nerve fiber under external electrical field. Negative electrode (active electrode) attracts positive ions from the membrane and causes depolarization (scan from [BBBW81]).

This will allow smaller fibres to be manipulated at lower force levels, while large fibres will be available to produce large forces if necessary by applying strong stimulus. However, if stimulus is not applied directly on the membranes, but applied externally between two distant electrodes, the threshold of external stimulus for initiating an action potentials will vary inversely with the fibre diameter. This is because the increased membrane area of the larger fibre will receive a larger amount of current from the external electrical field. The increase in received charge due to the larger fibre diameter outweighs the requirement for a larger current to initiate action potentials for larger fibre diameter due to the Size Principle. This results in an inverse relationship between fibre size and threshold of externally applied stimulus. This unnatural "inverse size principle" prevents the fine manipulation of FNS-induced muscle force, since larger motor units are excited first. Zhou et al. [ZBS87] proposed a strategy to overcome the inverse size principle in FNS by using supra-maximal amplitude high frequency blocking stimuli with normal FNS stimuli.

There are two basic methods to regulate muscle force by external stimulation: One is recruitment modulation which varies the number of activated motor units. Pulse width modulation is preferred to amplitude modulation because less charge is transferred per stimulus for any given force, which reduces risk of electrode corrosion and tissue damage. In addition, pulsewidth modulation is straightforward to implement electronically. The other method is frequency modulation in which the stimulus frequency is modulated to

change axonal firing rate. It was found [CLPC91] that a combination of pulse width and frequency modulation in FNS improved control performance, in particular a better transient performance than that obtained using pulsewidth modulation alone.

It is also possible to selectively excite different kinds of muscle fibres with different membrane time constants by selecting external stimulus duration and amplitude. From a simple RC membrane model, it is known that the chronaxie value (Fig.1.6) is proportional to membrane time constant. Larger time constant membranes require more charging ions to reach a threshold potential to trigger action potentials. Therefore, for a fixed stimulus amplitude, a certain stimulus duration will activate some muscle fibres, but not activate others with longer membrane time constants. The relationship between amplitude and duration of threshold stimulus for externally excited muscle is called "Lapicque strength-duration curve" [BBBW81], which reflects the excitability of the muscle (Fig. 1.6). This strength-duration curve has a shape similar to the hyperbolic function $y=1/(x-a)+b$, where a and b are constants. For real nerves and muscles, these parameters may be considered to be constants for short pulses [BBBW81]. The important concepts in this hyperbolic "strength-duration curve" are that stimuli of short duration require a non-linearly increasing intensity in order to excite the tissue and that stimuli of extremely short duration will not cause depolarization except with intensities impractically high for clinical purposes. Conversely, a point is reached with the weak stimulus of a low level current at which no response occurs regardless of the length of time the current is applied. The magnitude of current just sufficient to cause excitation of a particular fibre is called "rheobase". It has become customary to measure the length of time required for threshold stimulus at a amplitude which is twice that of the rheobase, and to report this value as "chronaxie". Different kinds of nerves and muscle fibres have different characteristic Lapicque curves, e.g., a nerve has a Lapicque curve close to the axes due to its high excitability, while denervated muscle fibre has a Lapicque curve further away from the axes due to its low excitability. For nerves and muscle fibres with the same membrane time constant, selective stimulation using stimulus duration and amplitude will not work. The size principle could be applied to recruit nerves with same membrane time constant but different axon size.

In FNS terminology, the "motor point" is an important concept. A motor point is not a particular anatomical structure or physiological organ like the motor unit, but is rather

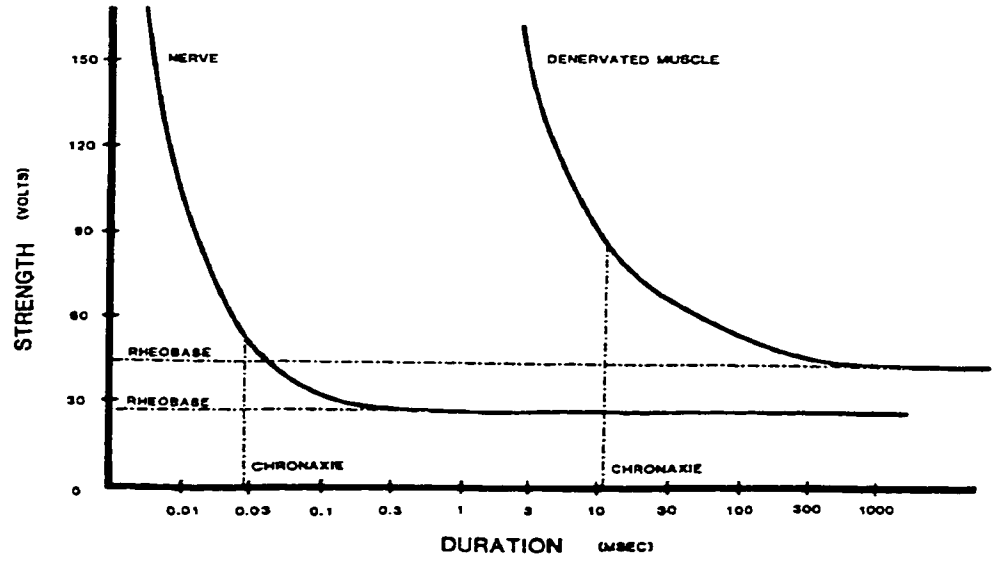


Figure 1.6: Strength-duration curve of threshold stimuli to nerves and muscles. The left curve represents highly excitable nerve, and the right curve describes the low excitable denervated muscle. Two important points also shown are, rheobase, the minimum amplitude to excite the tissue, and chronaxie, the duration of a stimulus at twice rheobasic amplitude must be applied to excite the tissue (scan from [BBBW81]).

a point on skin surface where an electrical stimulus can produce significant contraction of a particular muscle or muscle group. Usually it is a point on the skin where the motor nerves to a particular muscle pass superficially, or a point where an underlying muscle can be selectively stimulated. Various motor points maps and practical electrode placements have been described [BBBW81]. One important motor point is situated over the common peroneal nerve on the lower extremity where stimulation can illicit a flexor reflex that involves both the hip and knee joints. This is useful for initiating the swing phase of gait using surface electrodes [GHN+93].

Although the application of electricity to treat neuromuscular diseases has a centuries long history [BBBW81], Liberson's work [LHSD61] is widely considered to be the beginning of modern FNS technology. In this landmark work, Liberson and his colleagues applied electrical stimulation to the peroneal nerve to dorsiflex the foot in order to overcome foot drop during the swing phase of hemiplegic gait. The electrical stimulation was synchronized to the gait cycle using a heel-switch attached under subject's foot.

Around 1962, a group of researchers at Case Western Reserve University led by James B.

Reswick began extensive studies on the application of FNS technology to assist functional movements of the SCI. For more than 30 years, the Cleveland FNS research group, including researchers at CWRU and Cleveland Veterans Affairs Medical Center, has been the most active FNS research group [CKM+88, Chi92, Kob94].

In 1962, the first symposium of "Advances in external control of human extremities(ECHE)" was held in Yugoslavia, during which Dr. Rajko Tomovic, one of pioneering researchers in FNS and rehabilitation robotics, made introductory remarks on human extremities control. James Reswick, and Norbert Wiener were also invitees in that first symposium [Res90]. This international ECHE symposium has been held every three years since then and has documented progress in FNS [Pop90].

Recently, a commercial FNS system, Parastep© System, was developed and marketed by Sigmedics Inc., Northbrook, IL, USA [KAM+93]. It had been undergoing the first FDA-approved FNS clinical trials at 20 sites with over 100 subjects (mostly complete SCI subjects) since the late 1980's. The Parastep© system is a 6-channel, microprocessor-controlled device using surface electrodes in which the quadriceps and hip muscles were stimulated to extend the knees and hips for standing. The peroneal nerves were stimulated to illicit flexion reflexes for stepping. In this system, no bracing was used and an adapted walker was used to provide body weight support and for mounting the manual control switches. The functional purpose of the Parastep© system is upright mobility–including moving through doorways and up a one-step curb–as an alternative, but not a substitute, for a wheelchair. The participants in the clinical trials could purchase the system at cost ($7,000 - $10,000).

In April 1994, the Parastep© system, initially developed by D. Graupe and associates at the University of Illinois, became the first and the only FNS walking system to obtain FDA approval for unbraced ambulation by complete paraplegics [GK95]. The system is presently in use by over 300 people worldwide. This is a significant development since prior to this, FDA labelling requirements restricted FES/FNS only to the following uses: retarding or reversing atrophy; increasing muscle blood flow; muscle reeducation; maintaining or increasing range of motion; inhibiting spasticity; and preventing venous thrombosis in post-surgical patients. The new FDA approval for unbraced FNS ambulation means that FNS technology has finally achieved some clinically recognized "functional" benefits rather

than being a purely therapeutic modality.

Currently, the wheel-chair remains the most efficient and convenient way for paraplegics to achieve mobility, and its potential has been well exploited. However, FNS is a promising and challenging methodology which could greatly improve the quality of life through physical/psychological rehabilitation [GFAD93] and workplace mobility[KAM+93] for paraplegics provided some key technical problems are overcome. Many improvements are required before FNS can become as mature as say the cardiac pacemaker. Currently, there are approximately a dozen FNS research groups around the world, in the U.S.A., Canada, U.K., Japan, Netherlands, and Slovenia, and Yugoslavia. Some typical FNS research topics are: FNS control strategies [CKM+88, Chi92, Pro93]; computer simulation studies [KZ89, YZ90]; FNS electrode technology (surface, percutaneous and wireless implanted electrodes) [LZST91, SSLT92]; application of different sensors [CCNH86, And95], including strain-gauge flexible goniometers [KAP+95], force sensing resistors [ABPK89, And95], accelerometers [VFVB93, And95], natural sensory nerve signals (ENG) [PSJ+93], EMG signals[GK95], sensor signal telemetry [FC93, And95]; hybrid FNS systems incorporating active or passive mechanical braces [ABB+88, PTS89, KAM+93, DH90, Dur92]; multichannel computerized FNS systems [EWG+91, GP91, IVP94, BFK89], which usually consist of a portable stimulator based on microprocessor and an experimental control host based on personal computer or mini computer; methods of selective nerve stimulation [ZBS87, CLPC91] and therapeutic effects of FNS [SRZG93, GFAD93] to overcome the chronic secondary problems caused by reduced muscle activity after SCI, such as disuse atrophy of muscle, bone demineralization, impaired circulation leading to abnormal thrombus formation and decubitus ulcers, increased incidence of hypertension and heart disease, and a general decrease in cardiopulmonary function.

The future of FNS and other neural prostheses appears to be optimistic, according to G.E. Loeb [Loe89]: "Neural prosthetics today is at a level of development comparable to cardiology 25 years ago, when pacemakers were novel and primitive, ECG interpretation was largely subjective and empirical, and artificial hearts were a dream. The nervous system is certainly more complicated than the heart, but our technology is now vastly more sophisticated and more rapidly delivered. The next 25 years will be seminal for applied neurosciences."

Maybe, this new emerging technology should be coined as "neuroengineering", rather than the awkward "applied neurosciences".

Chapter 2

Literature Review

After decades of extensive research and clinical evaluation, FNS has been demonstrated to be feasible in assisting paraplegic subjects to stand and walk [LHSD61, TM66, BBBW81, HH88, Loe89, ABPK89, Pop90, YZ90, Chi92, KAM+93, BR94, KM94, GK95]. However, there are several limitations to the daily-use of FNS, including: the rapid onset of muscle fatigue induced by electrical stimulation, limited ability to modulate the force of muscular contraction, high energy consumption due to excessive effort of the upper body, limited selection of sensors, limited human-machine interface, and the difficulties in controlling highly non-linear, time-varying, and high degree-of-freedom neuromuscular systems. Poor control results in unnatural, jerky and inefficient FNS gaits in terms of metabolic energy consumption, speed and endurance [KSM+91, Mar91, Kob94, KM94]. Improved control is therefore essential to improve the practicality of future FNS systems.

2.1 General Review of Adaptive Nonlinear Control

The neuromuscular system is non-linear (e.g. muscle recruitment curve), time-varing (e.g. muscle fatigue and potentiation), and subject to external disturbances (e.g. load change in hand grasping control or the unpredictable voluntary movement of upper body in leg walking control). Before discussing FNS control strategies, it may be helpful to review adaptive nonlinear control strategies in general in order to get a broad view of the available tools.

If the plant is linear and time-invariant, then standard linear system feedback control (e.g. PID, LQR) methods may suffice [Veg90, Kir70]. If the plant is time-varying but

linear, then conventional linear adaptive controllers [AW95] such as the Self Tuning Regulator [HB81] using recursive parameter identification algorithms, or the Model Reference Adaptive Controller [NM80] using Lyapunov stability techniques, are applicable.

Although for non-linear and time-invariant systems, there are no well-established systematic theories similar to those in linear systems, there are some practical nonlinear control methods [SL91]. For example, gain scheduling [JJ93] can synthesize a piecewise controller using different feedback gains for different operating points. Sliding mode controllers have excellent robustness properties to nonlinear model parametric uncertainty [Slo84]. A nonlinear feedforward compensator can be designed to cancel nonlinearity. Various linearization methods could also be used, such as perturbation linearization about an operating point [KZ89, HLL91]; feedback linearization (or exact linearization) [Isi85]. Isidori [Isi85] argued that after 1980s, differential geometry had proven to be as successful to the study of nonlinear systems as the Laplace transform and complex functions theory were in the 1950s to the study of single input single output linear systems or linear algebra in the 1960s was to the study of multivariable linear systems.

However, in general, for non-linear and time-varying systems, it remains very difficult to synthesize stable adaptive nonlinear controllers that can perform satisfactorily. Conventional non-linear control and adaptive control have been combined to synthesize adaptive nonlinear controllers such as the adaptive sliding mode controller [SC86] or the self tuning regulator with nonlinear compensation [dKSG94]. Since 1990, significant progress has been made in the use of artificial neural networks (ANN) [Fra89, MSW90b, WS92, NP90, LN93, HSZG92, Son93, ZHD+94, SS94a], fuzzy logic controllers (FLC) [Lee90, Men95], and neuro-fuzzy systems [Men95, JS95, Bru93]. Although the early investigations into the application of ANN and FL control were mostly conducted on an ad hoc or trial and error basis, recent work has now established a rigorous theoretical foundation with systematic design methods.

2.1.1 Adaptive Nonlinear Control Using Neural Networks

It has been proven that ANNs are universal nonlinear function approximators [ZHD+94, Son93, HSZG92, SS94a, SS92b]. Theoretically, ANNs can be used as nonlinear system identifiers and controllers [NP90]. The important issue now is how to design learning

or adaptation rules that guarantee stability and convergence for adaptive nonlinear ANN controllers.

If there is a explicit teacher to provide the desired control signals, then supervised learning methods such as Least Mean Squares (LMS), Recursive Least Squares (RLS), or BackPropagation (BP) [Mam92] can be used to adjust parameters. However, in many real control problems, such explicit teaching signals are not available or very expensive to obtain. For example, in human motor learning tasks such as playing golf, a teacher can demonstrate the desired movement trajectory, but cannot directly show the desired motor commands or muscle activities to the student. Therefore, converting the trajectory tracking error to a motor control error is an essential and difficult task.

To obtain a motor control error from the trajectory error, the backpropagation algorithm could be used if the Jacobian matrix (partial derivatives of system output with respect to system inputs) of the controlled plant is known. However, in practice, the Jacobian matrix of a plant is usually unknown or time-varying. Therefore, learning or adaptive methods are required to learn the unknown Jacobian or adapt to a time-varying Jacobian.

Psaltis et al. [PSY87] proposed several learning schemes for neural controllers. His "generalized learning" scheme used different control inputs to the plant and observations of the actual plant outputs. An ANN used the actual plant trajectories as input and the control inputs to plant as desired ANN outputs. After the ANN learning phase was completed, it will have acquired the inverse dynamics of the plant. Then the ANN used the desired trajectory as input, and its output signals controlled the plant so as to follow the desired trajectory. The problem with this generalized learning scheme was that a large general operating range had to be learned by the ANN to avoid degrading its performance about any particular operating point. It is impossible to teach an ANN while it is in actual use, thus restricting the learning to be off-line. Another drawback to this scheme is that the plant may not be invertible, a condition which is not always satisfied in practice, for example, a redundantly controlled plant whose degrees of freedom of the motor commands exceed the degrees of freedom of the state variables, such as the primate limb. In such a redundantly controlled plant, many sets of motor commands correspond to a single movement, and a unique invertible relationship between the desired trajectory and motor commands does not exist. Even if the plant is invertible, the inverse control scheme may

not be acceptable, for example, the direct inverse of a non-minimum phase system is not stable. Psaltis "specialized learning" solved some of the problems of generalized learning by serially inserting the ANN controller between the desired trajectory and the plant. The ANN had the desired trajectory as input and learned the control signals required to keep the plant on the desired trajectory. This scheme was an example of on-line learning whilst actually performing a control task, and could learn within the operating region of interest. In order to derive the control error to train the ANN, the trajectory error was propagated back through the plant using an estimated Jacobian at particular operating points. This was called "backpropagation through the plant". However, the drawback was that the Jacobian could only be estimated by input perturbation or comparing changes with previous iterations. As reported in [MM92], any subtle distortions in the gradient estimation might cause the controller to fail, especially for highly nonlinear control tasks. It was suggested that a combination of generalized learning with specialized learning might obtain a better result.

Nguyen and Widrow [NW90] used two ANNs for nonlinear control. One ANN was first trained as an emulator to identify the plant's dynamics, using the plant's inputs as input data and the plant's actual trajectory as the ANN's teacher signals. After the emulator training was completed, another ANN was used as a controller to subsequently control the ANN emulator. Since the internal structure of the ANN emulator was known, the trajectory error of the emulator could be back-propagated through the emulator to obtain a motor control error for the ANN controller. After this two-phase learning was completed, the ANN controller could then be used alone to control the actual plant. This controller was successfully applied to the control of truck guidance system. However, this is an off-line learning scheme and can not be used as an on-line adaptive controller expected to cope with any time-varying plant dynamics. Another problem, as in Psaltis' generalized learning, was that the ANN emulator had to be trained about a wide range of operating points to truly identify the plant's dynamics. Otherwise, even if the ANN controller could control ANN emulator correctly, it may fail to control the real plant. Any practical ANN will have only a limited ability to approximate nonlinear functions. Therefore an ANN-based controller can not be expected to have a fine performance in any particular operating area if the training is done in other, perhaps irrelevant, operating area. Jacobs and Jordan [JJ93] have proposed

a modular ANN for learning piecewise control strategies such as non-linear gain scheduling control. They also demonstrated that the control performance of such modular ANN's was indeed superior to that of a single ANN controller.

Jordan et al. [JJ90] also proposed backpropagation through a forward model learning method, in which an ANN emulator and a ANN controller were used on-line. The ANN controller used the desired trajectory as input, and its output was sent to both the controlled plant and the ANN emulator. The plant's actual trajectory was sent to the ANN emulator as the teacher signal. The error between actual plant trajectory and the desired trajectory was propagated back through the ANN emulator to obtain a control error signal for the ANN controller. As the learning progressed, the ANN emulator acquired the plant's dynamics, whilst the ANN controller learned a control law that kept the plant on the desired trajectory. This forward-inverse model was an on-line learning scheme and could be focused on special operating points of interest. However, it was reported that [MM92] the controller's performance was sensitive to the accuracy of the forward model, depending on the degree of nonlinearity in the control task. In highly nonlinear situations, subtle distortions in the gradient predicted by an inaccurate forward model could cause the controller to fail. Therefore, the success of backpropagation through the forward model method may be highly task-dependent.

Cui and colleagues[CS93] used a simplified Jacobian matrix of the plant, consisting of only the signs of the partial derivatives in order to to calculate the control error from the trajectory error. This scheme only worked for a class of 'monotone-responded' plants whose Jacobians were either positive or negative. Yuh [Yuh94] also proposed a variant of the M.I.T. rule in which a constant approximation to the plant Jacobian was used to train a ANN controller for an underwater robotic vehicle. However in practice, many plant Jacobians change magnitudes and signs at different operating points and, for some complex systems, it is difficult to determine even the signs of the Jacobians. Therefore, the applicability of these learning controllers using simplified plant Jacobians is limited.

Based on neural circuits in the cerebellum, especially transcortical somatosensory feedback loop, Kawato [KFS87] proposed a controller using a neural network combined with a PID feedback controller. The PID output was used as the training error signal for the neural network. The PID controller actually played the role of a linear approximator of

the inverse dynamics of plant and converted the trajectory error into a motor control error. As learning progressed, the neural network minimized the output of the PID controller whilst increasing its own control output, and thus gradually acquired the inverse dynamics of plant. The control performance improved as the nonlinear dynamics were gradually compensated for by neural network, thereafter the linear PID controller only needed to correct a small residual tracking error. This ANN controller was successfully applied to robot control [KFS87, KUIS88], and also provided clues about the biological control of eye movements [KG92]. Later, based on the Lyapunov method, it was proven [Kaw90, NM93] that this feedback-error-learning rule was actually a stable adaptation rule. Therefore, this learning scheme is not only biologically plausible, but also theoretically sound.

Gomi and Kawato [GK90] extended the original Kawato model to two new models, the Inverse Dynamics Model Learning (IDML) and the Nonlinear Regulator Learning (NRL), and proved global stability using the Lyapunov technique. The difference between IDML and Kawato's original model is that IDML used the actual trajectory as input to the ANN, while original model used the desired trajectory as input to the ANN. An additional stochastic disturbance was also added to the control signal to satisfy the persistent excitation condition. A computer simulation showed that both the ANN parameter estimation and trajectory tracking were stable and convergent. As in Kawato's original model, the ANN in the IDML also gradually acquired the inverse dynamics of plant as learning progressed. The difference between the NRL and Kawato's original model is that NRL used trajectory error feedback, in addition to the desired trajectory, as inputs to the ANN. Hence the ANN used in NRL executed adaptive feedback control as well as adaptive nonlinear inverse dynamics compensation, whilst the ANN in the original model only acted as a feedforward nonlinear compensator. Since these two new models used the actual trajectory and/or the desired trajectory as inputs to the ANN's, they were called "closed-loop" feedback-error-learning, and the original model was called "open-loop" feedback-error-learning because only the desired trajectory was used as input to ANN. Although a PID feedback controller was used in all three models, it merely converted trajectory error into motor control error as a linear approximation of the inverse dynamics of the nonlinear plant. The linear PID controller itself was not able to control a nonlinear plant. That is why even though there was a closed-loop PID component in Kawato's original model, it was still called 'open-loop'

control, whilst "closed-loop" referred to a nonlinear feedback loop and not the linear PID feedback loop. However, the PID component did provide the major control signal during the initial learning phase and provided the training error signal for the ANN component for stable weight adjustment, thus guaranteeing global trajectory stability and convergence. The PID component also defined an inverse reference model that the complete dynamic system would ultimately follow after the training was converged.

The above three different feedback-error-learning models were used to explain different biological motor control tasks in four regions of cerebellum [GK90, KG91]. It was suggested that: Kawato's original "open-loop" feedforward controller was suitable for adaptive control of voluntary motor tasks such as hand trajectory following, during which the feedforward nonlinear controller could provide motor commands rapidly and without feedback delays. The "closed-loop" IDML model also suggested for posture control since the controlled plant was inherently unstable and nonlinear feedback control is desirable. Computer simulations [GK90] of the inverted pendulum, the simplest model of trunk and legs, suggested that the IDML scheme might be appropriate for posture control. The NRL "closed-loop" model was also shown to be appropriate for the control of locomotion, during which desired rhythmical movement pattern was time varying and usually high speed, thus feedforward control is necessary to excuse fast and smooth movement. Meanwhile, the controlled plant in locomotion was also inherently unstable, thus nonlinear feedback control was also essential. Therefore, adaptive control of locomotion required both nonlinear feedforward and nonlinear feedback control, which were both included in the NRL model.

Miller et al. [MHGK90] also proposed a similar combination of a neural network with a PID controller. The neural network used was the Albus' Cerebellar Model Arithmetic Computer (CMAC) [Alb81, MGK90] based on the neural structure of the cerebellum [Mar69, Alb81]. Again, the training error signal was the output of a PID module. In contrast to Kawato's on-line training method, a two-phase training method was adopted in Miller's approach. In the response (recall) phase, the input to the CMAC was the desired trajectory and the output of the CMAC was then combined with PID output. At the end of each control cycle, a training (update) phase was executed. The observed trajectory during the previous response phase was used as input to CMAC, and the actual control (sum of CMAC and PID outputs) during the previous response phase was used

as the desired control signal to update the CMAC parameters. Thus Kawato's approach involved a continuous on-line learning, whilst Miller's approach used a response-training cycle. Because of this intermittent response-training cycle, the whole system is not a continuous dynamic system, and Lyapunov stability analysis can not be applied to Miller's system. Although there is no theoretical guarantee, Miller's method is expected to yield similar results as Kawato's approach. This has been demonstrated by numerous successful robot control experiments conducted by Miller. The CMAC model was also applied to the control of a biped walking robot[Mil94]. Kraft and colleagues [KC90] compared this CMAC-based controller with two conventional adaptive controllers, namely the STR and the MRAC. They found that the CMAC approach had more tolerance of noise, performed well for both linear and nonlinear systems, and could be implemented more efficiently for large-scale systems. In Kraft's study, the desired trajectory rather than actual trajectory, was used as input to the CMAC during the training phase.

Nordgren [NM93] used the Lyapunov method, in a similar way to its use in the MRAC design, to derive a globally asymptotically stable (GAS) adaptation law for a single-layer neural network (adaptive linear combiner) controller combined with a PD feedback component. It was found that the GAS law shared similarities with Kawato's feedback-error-learning law, but had a faster convergence rate due to the use of an individual learning rate for each adjustable weight. Parameter convergence was not achieved if the persistent excitation condition of the input was not satisfied, although trajectory tracking convergence was guaranteed.

Sanner and Slotine [SS92b, SS92a, SS94a] have extensively studied the use of gaussian radial networks for nonlinear compensation and feedback linearization. Here, the controller was a combination of PD feedback, an adaptive gaussian network and a robust sliding component. A stable weight adjustment law for gaussian network was synthesized using Lyapunov theory. Deadzones were introduced into the network's adaptation mechanism and used together with the robust sliding controller to keep the plant state within a bounded nominal operating range for which the network was designed. When the plant state moved outside the nominal range due to the approximation error, the robust sliding controller would force the state back into the nominal range, whereupon the gaussian network was able to adaptively approximate the plant nonlinearity. As with the Nordgren's

model. trajectory tracking convergence was guaranteed. but parameter convergence relied conditionally on the persistent excitation condition in the feedback error signal. Because inputs to the ANN were feedback error signals that could not be selected. parameter convergence can not be guaranteed. However in Nordgren's model, the input to the ANN was the desired trajectory that could be selected to satisfy the persistent excitation condition that guaranteed convergence of the parameter estimation. Another difference is that in the Nordgren's model. the robust sliding component was not needed because the ANN was used as a feedforward compensator for the inverse dynamics. Thus inputs (the desired trajectory) to the ANN could never leave the bounded set in state space for which it was designed. Robustness and convergence of adaptation in Nordgren's model relied upon a persistent excitation of the inputs. whilst in Shanner's model. robustness relied on a sliding component. Levin and Narendra [LN93] also reported a simple example in which a ANN was used to adaptively learn the feedback linearization.

Zbikowski and Hunt et al. of NACT (Neural Adaptive Control Technology) research group presented an extensive review of neural adaptive control [ZHD+94]. A number of fundamental control and system theories were reviewed to provide a theoretical basis for the analysis and design of adaptive neural control. The ANN method was analyzed from the perspectives of system identification. function approximation. stability theory. nonlinear controls (including the differential geometric approach. i.e. feedback linearization and the sliding mode approach). and adaptive control theory.

Summary: In summary, it appears that one biologically plausible and theoretically solid motor control model may be the combination of CMAC model with the Kawato on-line training method. The cerebellum is widely believed to be control center for motor control learning and adaptation [Ito84]. The Marr-Albus cerebellum model (CMAC) is an adjustable pattern classifier based on the neural structure of the cerebellum and has been shown to be an universal nonlinear approximator. Unlike the backpropagation rule for multi-layer sigmoid neural network which may get stuck in a local minima, the CMAC's LMS-like adaptation rule is guaranteed to converge to a global minima, due to its quadratic error surface. In addition, the CMAC's learning rate, because of its local nature, is much faster than backpropagation [Mar69, Alb81, MGK90]. The CMAC is a static associative memory model and does not solve the issue of how to obtain the motor training signal. On the other hand, Kawato's

feedback-error-learning scheme is a dynamic motor learning and control process that is based on various somato-sensory feedback to cerebellum and is guaranteed, by Lyapunov theory, to be a globally asymptotically stable adaptation law for dynamic systems. Therefore, a combination of the static CMAC model with a dynamic feedback-error-learning law should be good adaptive learning motor control strategy. This is a good example of how engineering can learn from biological systems, as well as provide mathematical explanation of working mechanisms of biological systems.

Werbos of NSF and Pellionisz of NASA [WP92, PJW92] suggested a cerebellar neurocontroller project involving NFS, NASA and NIH to develop an adaptive and reliable flight control systems, based on the adaptive sensorimotor control systems in cerebellum, for an airplane which is able to reach earth orbit (NASP, National Aerospace Plane) and will be essential for large-scale economical human activities in the space. From the perspective of neurosciences, this multidisciplinary collaboration would be "leading us towards an exact link between well-defined mathematical operations and specific sites in the brain – something which, when consolidated, would truly constitute a Newtonian revolution in neuroscience"[WP92].

2.1.2 Reinforcement Learning as Direct Adaptive Optimal Control

Most current artificial learning systems are supervised learning systems which learn under the tutelage of a knowledgeable 'teacher' that is able to provide a set of required training input-output examples. Learning under these conditions is limited, particularly, when it is impossible to obtain this kind of rich training information. Reinforcement learning refers to learning without an explicit teacher, by actively interacting with the environment and receiving appropriate reinforcements (both reward and penalty). It is based on the common sense idea that if an action is followed by a satisfactory performance, then the tendency to reproduce that action is strengthened; alternatively if an action leads to an unsatisfactory performance, then that tendency is weakened, i.e. only satisfactory actions are reinforced. Reinforcement learning is a learning mechanism rooted in biological learning systems, while having a rigorous mathematical basis and a potential for engineering applications.

Reinforcement learning has its origin in the study of animal learning, in which qualitative models were used explain the learning behavior of animals and humans. The Rescorla-

Wagner model [RW72] of classical Pavlovian conditioning is one such model. Meanwhile, in the engineering field, Mendel and Fu [MF70], and Narendra [NT74] et al. proposed quantitative reinforcement learning algorithms for stochastic learning automata, statistical pattern recognition, stochastic approximation and optimization. During this early stage, most of the reinforcement learning models were "nonassociative", i.e. there were no inputs to the learning system except for reinforcement signals.

Later, "associative reinforcement learning" was emphasized, in which the learning systems try to associate input patterns with output signals according to reinforcement feedback. Klopf [Klo74] proposed a hypothesis that neurons implemented a strategy for attempting to maximize the frequency of occurrence of one type of input signal and minimize the frequency of occurrence of another. Accordingly, neurons could be conditioned in an operant or instrumental manner, where certain types of inputs acted as reward stimuli whilst others acted as punish stimuli. A neuron learns how to attain certain types of reward stimuli and avoided other punish stimuli by adjusting its synaptic weight according to the feedback consequences of its discharges.

Klopf's "greedy neuron" hypothesis inspired the well-known neuron-like ASE/ACE controller of Barto, Sutton, and Anderson [BSA83]. This single-layer linear-combinator-like neural controller consisted of two elements: an Adaptive Search Element (ASE), which was a direct computational realization of Klopf's neuron model, searched for optimal control outputs according to the input patterns. The output of the ASE was added with a random variable to determine a stochastic binary control signal. The stochastic control signal was biased by the deterministic ASE output, which gradually moved toward the optimal control as learning progressed; an Adaptive Critic Element (ACE) calculated a secondary internal reinforcement based on input patterns and feedback reinforcement from environment, and used it to condition weights in the ASE. The input space was encoded using the BOXES system proposed by Michie and Chambers [MC72]. To cope with delayed reinforcement, decaying eligibility traces were used in both the ASE and ACE, and a temporal credit assignment algorithm was also used in ACE. The eligibility trace in ASE was proposed by Klopf to represent the decaying strength of a pairing of a nonzero input signal with the firing of the neuron. The eligibility trace in ACE was similar to the "local demon" in Michie's BOXES system. The ACE used an Adaptive Heuristic Critic (AHC) [Sut84, Sut88] rule

to calculate the secondary internal reinforcement from delayed or infrequently-occurring reinforcements. The AHC was a temporal credit assignment algorithm similar to Holland's "bucket brigade" used in Genetic Algorithm "classifier system" [BGH89]. Both borrowed the same key idea from checker-playing program written by Samuel [Sam59] – in that the steps of a multi-step sequence should be evaluated and adjusted according to their immediate on near-immediate successors, rather than on the final outcome. ASE/ACE was successfully applied to the difficult task of learning the control of an unstable car-pole system with only a failure signal as reinforcement.

The ASE/ACE model marked the beginning of the new stage of associative reinforcement learning. It was mathematically formalized in [BAS5] as a pattern-recognizing stochastic learning automata, and was generalized to the Associative Reward Penalty (A_{R-P}) algorithm. In addition, some convergence properties were proven.

The BOXES scheme used in ASE/ACE for input space partitioning required careful arrangement to give fine discriminations among some critical states and coarse generalization among others, so as to achieve good learning control performance without excessive memory requirements. To cope with this drawback, Anderson [And89] used multi-layer neural networks as the ASE and ACE in order to approximate the nonlinear evaluation and control functions. Neural networks are universal function approximators and can be used to eliminate the requirement for manual partitioning of the input space since neural networks can learn by themselves to make the right partitioning. It was demonstrated that a multi-layer neural network was able to learn to balance the inverted pendulum, while a single-layer neural network failed without the BOXES input encoding. In the same simulation, it was also shown that ASE/ACE model learned much faster than Michie's BOXES. However, multi-layer neural networks required thousands of trials to succeed, while original ASE/ACE controller with BOXES input encoding succeeded in less than 100 trials. One reason for this hugh difference in learning rate could be that manual adjustment work in the BOXES encoding was now automated in multi-layer neural networks and certainly required extra learning trials. Another reason could be the well-known slow learning rate of multi-layer neural network using backpropagation. A third reason could be that the eligibility traces were not used in Anderson's multi-layer neural networks (only AHC was used). This would also slow down the learning rate in a delayed reinforcement learning

task.

After early mostly empirical development, some rigorous mathematical results have emerged for reinforcement learning since the late 1980s and early 1990s.

Sutton [Sut88] introduced a class of Temporal Difference (TD) methods to solve the multi-step prediction problem. The TD method formalized the previous poorly understood methods such as AHC, Holland's bucket brigade, and Samuel's checker player program. Convergence and optimality, for special cases, was proven (The proof was extended to the general case with stronger results by Dayan [DS94]). The TD method became one of the fundamental theorems of reinforcement learning which usually deal with delayed and infrequently-occurring reinforcement signals. TD method was further extended to vector predictions, and combined with backpropagation to implement in multi-layer neural networks with eligibility traces for every adjustable weights [Sut87, BS92]. The implementation in [Sut87, BS92] was more complete than that in Anderson's model [And89].

The TD method could be used for the multi-step prediction of reinforcement signals in reinforcement learning. However, it did not address the issue of how to train the controller using this more informative secondary reinforcement signals. Williams' REINFORCE algorithms [Wil87, Wil92] provided a general class of associative reinforcement learning algorithms for connectionist networks containing stochastic units allowing them to make weight adjustments along the gradient of the expected reinforcement. Interestingly, the REINFORCE algorithms perform the statistical gradient following without explicitly computing gradient estimates or even storing information in preparation for computing the such estimates. The REINFORCE algorithms could also be used with both immediate-reinforcement tasks and in limited forms of delayed-reinforcement tasks. When combined with TD methods, they could be applied to general delayed-reinforcement tasks. When applying REINFORCE algorithms to multiparameter stochastic units such as a Gaussian unit, one would obtain reinforcement learning controllers with real-valued continuous control outputs. Previously, reinforcement learning controllers (e.g. [BSA83, And89]) were limited to binary control outputs. It was pointed out that certain algorithms, studied elsewhere, were special cases of the REINFORCE algorithms, including Barto's associative reward-inaction A_{R-I} algorithm (a variant of the associative reward-penalty A_{R-P} algorithm) [BA85], Narendra's linear reward-inaction (L_{R-I}) stochastic learning automa-

ton [NT74]. Gullapalli's stochastic real-valued (SRV) unit [Gul90] for a real-valued controller. Since REINFORCE is a class of statistical gradient following algorithms, it could be integrated with backpropagation. Williams theorem is another important theorem for reinforcement learning.

Watkins [WD92] has proposed a kind of reinforcement learning called Q-learning. Q-learning uses a real value function Q to represent the performance evaluation of state-action pairs. Unlike evaluation functions in other common reinforcement learning models which only used state variables, the Q-value function was defined on state-action pairs. Therefore, Q-learning is a more complete learning model than other reinforcement learning models like the ASE/ACE, and is becoming the most prominent reinforcement learning algorithm.

Recently, Barto and Sutton et al. [BSW90] showed that the TD methods used in reinforcement learning (RL) for evaluation prediction and optimal policy learning were closely related to stochastic Dynamic Programming (DP) [Bel57]. TD methods solved sequential Markovian decision making tasks by using temporal differences similar to the idea of the "greedy policy" used in "Bellman Optimality Equation" of DP. Reinforcement learning is an on-line, Monte Carlo version of DP in the sense that the probability distributions usually required by DP are sampled rather than known in RL (i.e. try an action and see a particular consequence rather than knowing the probability distribution of the consequences). In general, RL differs from DP in the fundamental way that a model of the world's dynamics is not required. Therefore, RL is a kind of Real-Time Dynamical Programming (RTDP) [BBS93], by which an embedded system could improve its performance with experience. Bradtke [Bra93] designed a Linear Quadratic Regulator (LQR) using a reinforcement learning algorithm to demonstrate that DP-based reinforcement learning could be applied to traditional control problems with continuous state and action spaces.

Werbos proposed two types of reinforcement learning. One was based on the backpropagation of utility, including backpropagation through time [Wer90, Wer92b] to maximize a utility function; another type was the Adaptive Critic, or Approximate Dynamic Programming [Wer92a], in which a secondary utility function J was approximated by an adaptive critic network. Any control strategy that maximized the secondary utility function J in the short term would also maximize the sum of primary utility function U over all future times. The adaptive critic network thus performed a continuous and differentiable map-

ping between the control space and the evaluation space, and backpropagated the primary utility U into a secondary utility J through the critic network. In this sense, it might be called as backpropagation through critic.

As pointed out by Sutton, Barto, and Williams [SBW92], reinforcement learning is direct adaptive optimal control. RL is adaptive since it can learn the control law and adapt to a changing system; RL is optimal since it can approximate (statistically in Williams' REINFORCE or deterministically in Werbos' Adaptive Critic) the optimal control strategy using real-time Dynamic Programming; RL is direct since it does not need a system model to determine the optimal control. Most well-known adaptive control methods [AW95], including STR [HB81] and MRAC [NM80], are associated with tracking and regulation problems, in which prior knowledge of a reference trajectory or set points are known. But for many problems, the determination of a reference trajectory is an important part – if not the most important part – of the overall control system design. For example, trajectory planning is a key and difficult problem in robot navigation tasks. This kind of trajectory planning or formatting tasks could usually be solved by optimal control principles [KUIS88, Kaw90, KG92, LFC94, HLL91, LN93, YZ90]. However, these optimal control problems used to be solved in an off-line manner using tools such as Dynamic Programming, and usually required expensive computing power, sometimes at the level of supercomputer [LFC94, YZ90]. Now reinforcement learning provides an approach to do the optimal control in an on-line, direct, incremental manner, requiring less on-line computing power.

The most recent development in reinforcement learning has been the use of function approximation and generalization to scale up RL to large systems.

Thrun [Thr93] pointed out that inaccurate function approximation would lead to a systematic overestimation of reinforcement value function. When overestimation exceeded certain bounds, reinforcement learning was expected to fail. Simulations using some of the most popular function approximators, including multi-layer neural networks with linear, sigmoidal, or radial basis functions, supported the theoretical findings.

On the other hand, Singh and Yeh [SY93] presented a result which guaranteed that small errors in the approximation of value function cannot produce arbitrarily bad performance when actions were selected greedily. An upper bound on performance loss was derived. Williams and Baird [WB93] also gave similar but tighter performance bounds of reinforce-

ment learning based on imperfect value functions. These theoretical results provided a theoretical justification for using function approximators in reinforcement learning.

Successful computer simulations by Sutton [Sut95] using CMAC as function approximators supported the above theoretical results. It was argued that the use of sparse-coarse-coded function approximators such as CMAC, instead of using global functional approximators such as sigmoidal multi-layer networks, was the key for success. To avoid the "curse of dimensionality", an effective method was to ignore some dimensions in some partitioning, i.e. to use hyperplanar slices instead of grid boxes. Another method was "hashing" – a consistent random collapsing of a large state space into a much smaller set. Hashing reduced the memory requirements by large factors with little loss of performance. This was possible because high resolution was needed only in a small fraction of the state space. Hashing reduced the curse of dimensionality in the sense that memory requirements needed not be exponential in the number of dimensions, but needed merely match the real demands of the task.

There are some practical research topics such as using more efficient eligibility traces [SS96], or using more efficient exploration methods such as directed exploration instead of a undirected random search [Thr92]. Another very active research area is to incorporate fuzzy logic into reinforcement learning to provide a good jump start for reinforcement learning and shorten overall learning time (This will be reviewed in the next section). All these practical methods aim at speeding up reinforcement learning which is usually quite inefficient, compared to other learning schemes [Hin89]. This is certainly due to the lack of informative teaching signals, rather than the drawback of reinforcement learning method. When a knowledgeable teacher is available to provide explicit training patterns, then supervised learning should be used. When a knowledgeable teacher is not available and autonomous ability is needed, reinforcement learning provides the solution. In some situations, reinforcement learning is more stable and efficient than supervised learning [MM92].

Researchers have applied reinforcement learning to various real world problems, including path-finding for autonomous mobil robot [Thr93]; gait synthesis for biped robot [SZ92]; collision-free trajectory planning for multi-linked robot manipulator [TP92], in which CMAC-based controllers were trained using REINFORCE and TD algorithms; and

an assembly robot able to do peg-in-hole insertion [GFB94]. Houk, Barto and Berthier et al. [MSW90a, BSB93] recently proposed a controller based on biological Adjustable Pattern Generator (APG) model trained by reinforcement learning algorithm. The controller successfully controlled a simple two DOF simulated limb via reinforcement learning. One of the long term objectives of this research is to produce a competent controller for complex dynamic limbs. Sofge and White [SW92] applied the Werbos' Adaptive Critic methods to learning the optimal control of composite processing. The adaptive critic network was implemented with a modified CMAC which was differentiable in order to backpropagate the utility function U to secondary utility function J. This adaptive critic network was found to have rapid convergence due to local learning paradigms of CMAC. Vector reinforcement signals were also used to represent different optimal criteria. The reinforcement learning controller was able to solve the difficult (resisting to other methods) optimization control of composite materials used in NASP [WP92]. These fruitful applications of neural control led to a handbook of intelligent control [WS92].

Reinforcement learning is attracting increasing attention in computer science and control engineering because it can be used by autonomous systems to learn from their past experiences instead of from knowledgeable teachers, and it is attracting attention in computational neuroscience because it is consonant with biological principles.

2.1.3 Adaptive Nonlinear Control Using Fuzzy Logic Systems

The past few years have witnessed a rapid growth in the number and variety of applications of fuzzy logic. The applications range from consumer products such as cameras, washing machines, and microwave ovens to industrial process control, medical instrumentation, decision-support systems, and portfolio selection. Some giant companies such as Siemens [HP94] and General Electric [BBC+95] are very active in exploration of the industrial applications for fuzzy logic technologies.

Fuzzy logic was introduced by Dr. Lotfi Zadeh [YOTN87] at the University of California/Berkeley in the 1960's as a means to model the uncertainty of natural language. It is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth – truth values between "completely true" and "completely false". In June 1995, Zadeh was awarded the IEEE Medal of Honor – the highest honor given by the IEEE

– for "pioneering development of fuzzy logic and its many diverse applications."[Per95] This medal from IEEE put Dr. Zadeh on the same honorable rank as other IEEE medallists including Richard Bellman, Rudolph Kalman, and Karl Astrom, as well put fuzzy logic on the same important position as dynamic programming [Bel57], Kalman optimal filter theory, and linear adaptive control theory [AW95].

Since the landmark paper of Zadeh in 1965 [Zad65], fuzzy logic control has emerged as the most active and fruitful areas for research in the application of fuzzy set theory.

In 1974, Mamdani [Mam74] reported his seminal work of applying fuzzy logic ideas to design control systems. The controller was similar to a PD controller, using feedback error and change of error as controller inputs, but the implementation was in the form of production rules, rather than numerical equations. Therefore, rigorous mathematical equations modeling the controlled plant were not required.

In 1979, Procyk and Mamdani [PM79] further proposed fuzzy Self-Organizing Controller (SOC) which was a generalization of simple fuzzy logic controller. SOC incorporated Mamdani's basic PD-like fuzzy logic controller, but added another high level fuzzy logic rule modifier which measured the performance of the basic fuzzy logic controller. Control rules of basic fuzzy logic controller were modified according to its performance, based on the fuzzy rules of a high level modifier. The SOC was applied to a wide range of nonlinear and multivariable systems and its robustness was demonstrated.

Tanscheit et al. [TS88] successfully applied the SOC to the control of a highly nonlinear jointed robot arm. It was found that the SOC had a performance slightly superior to that of a PID controller when noise was present and the process had time varing coefficients as well as pronounced nonlinear characteristics.

The above mentioned PID-like fuzzy logic control is somewhat similar to conventional numerical control, although implemented in different ways. There is another type of fuzzy logic control in which heuristic control rules from experienced human operators are used. For large scale industrial processes such as the control of cement kiln, it is difficult or expensive to build a mathematical model. On the other hand, experienced human operators can successfully control such systems using heuristic rules based on their experiences which include a qualitative model of the influence of control variables on the system. Such experiences are usually expressed in vague terms such as: "if the coal-feed rate is increased, the

kiln drive load and the temperature in the smoke chamber will increase". Fuzzy logic is an excellent tool to transfer such human linguistic experiences into computer algorithms and to improve control performance. Larsen [Lar81] reported the application of such expert-system-like fuzzy logic controller to two rotary cement kiln control projects. This was one of the first successful test runs on a full scale industrial process in the late 1970s.

C.C. Lee [Lee90] provided a comprehensive survey on the development of fuzzy logic control before the 1990s.

Recently, fuzzy logic control has advanced very rapidly. One source for this rapid advance is the availability of inexpensive yet powerful microcomputers, which make it easy to incorporate fuzzy logic algorithms in embedded controllers. Compared with the conventional numerical controller design method, the fuzzy logic rule-based design approach could drastically reduce the development time and cost [BBC+95], with usually excellent control performance [CCM91, Sch92]. Another reason for the advance of fuzzy logic control is the establishment of rigorous mathematical properties such as universal function approximation and global stability. These theoretical advances parallel those in neural networks, and there is now a trend to combine both techniques to produce neuro-fuzzy systems. In March 1995 issue of "Proceedings of the IEEE", there was a special issue on fuzzy logic with engineering applications, including a tutorial of fuzzy logic systems from engineering perspective [Men95], and a comprehensive survey on neuro-fuzzy systems [JS95].

Based on the theories similar to those used in neural networks, it has been proven [WM92c, WM92b, Men95, Cas95, BH93] that fuzzy logic systems are universal function approximators. Therefore, fuzzy logic systems can be used as universal nonlinear system identifiers and controllers. This is the theoretical basis that explains why fuzzy logic controllers, if designed correctly, are able to achieve excellent control performance for nonlinear systems.

However, the design of fuzzy logic controllers remains a nontrivial task. This used to rely purely on human experiences, as the above-mentioned PID-like or expert-system-like fuzzy logic controllers. This is what fuzzy logic was originally designed for, i.e. to translate human linguistic experiences into mathematical sets that could be used in computer algorithms. There are many methods available from "knowledge engineering" [Cro88], especially on how to acquire knowledge from human experts. Even though knowledge ac-

quisition and knowledge representation in fuzzy logic systems are not exactly the same as those in traditional expert systems, there are still many similarities.

In some cases, however, human experiences may not be available, but huge amount of experimental data are available. In other cases, human experiences can be further clarified by extracting useful information hidden in the huge amount of data that are hard for human to analyse. In these situations, learning fuzzy logic systems are essential. Combination of neural network and fuzzy logic results in so called neuro-fuzzy systems [JS95] which are able to incorporate human experiences as well as learn from examples. For adaptive or learning fuzzy logic controllers, the central problem is, similar to neural network controllers, how to design learning algorithms. There are two learning tasks for fuzzy logic controllers, namely, structure learning which determines rule number and structure, and parameter learning which updates parameters of rules.

For parameter learning, backpropagation is widely adopted [WM92a, LL91, Jan92]. Jang [Jan93] proposed a combination of backpropagation and Least Squares Estimation to reduce the dimensions of the search space in the gradient backpropagation method, with the restriction to linear consequence fuzzy membership functions. Wang proposed several learning algorithms for both structure and parameter learning, including direct rule generation from input/output data pairs [WM92c], and orthogonal least squares [WM92b] to select significant fuzzy rules. C. Lin et al. [LL91] used a Self-Organizing-Map (SOM) [Koh88] to design fuzzy rules. Tani et al. [TST92] used Quinlan's ID3 algorithm [Qui86], a decision tree machine learning algorithm, to design fuzzy rules. Jang [Jan94] proposed the use of CART (classification and regression tree), another decision tree machine learning algorithm similar to ID3, for structure determination of fuzzy rules, and the parameter learning was carried out by the backpropagation or least squares algorithms. The use of a decision tree learning algorithm generated a tree partition of the input space by selecting most informative inputs, thus relieved the problem of the "curse of dimensionality" (number of rules increases exponentially with number of inputs) associated with the commonly used grid partitioning schemes.

These learning algorithms basically are static pattern classifier type learning algorithms which do not take into account the dynamics of controller and controlled plant.

Jang [Jan92] used temporal backpropagation (or backpropagation through time), a

training scheme proposed by Werbos [Wer90] for long term prediction and optimization, to train neuro-fuzzy controller for dynamical systems.

Chung et al. [CO93] proposed a model reference fuzzy logic control in which the reference model generated a desired trajectory, and a performance index based on the difference between the reference and plant output was minimized using gradient M.I.T. rule. This reference model replaced the high-level rule modifier in SOC [PM79] to provide the performance index. However, as with the neural network control situation, in which the Jacobian matrix of the plant was not known, trajectory error was converted into a control error in order to train the controller. Chung only proposed a simplified MIT learning algorithm for a class of simple SISO systems. Nie [NL93] also proposed a similar model reference approach for learning fuzzy logic control.

Stability of fuzzy logic control systems have also been studied. It was pointed out [HP94] that most fuzzy logic controllers for nonlinear 2nd order systems designed with a two-dimensional phase plane using tracking error and change of error are actually sliding mode controllers which are robust to parameter fluctuations and disturbances. Similar robust control design methods could also be applied to higher order systems. Jang [JS95] also pointed out that certain kinds of fuzzy logic controllers were actually nonlinear gain scheduling controllers that switched between several sets of feedback gains.

Langari et al. [LT90a] used direct Lyapunov method to determine sufficient conditions for global stability of a class of fuzzy logic control systems using feedback error and change of error as inputs. They also briefly reviewed the historical context of stability analysis of fuzzy logic control systems. However, in Langari's approach, the mathematical model of the plant must be known in order to design a stable fuzzy logic controller.

For unknown plants, Wang [Wan92, Wan93, Wan94b] proposed an adaptive fuzzy logic control, using Lyapunov synthesis approach similar to Sanner's approach for the adaptive gaussian network control discussed earlier. The controller consisted of adaptive nonlinear and supervisory components. A slightly different method from Sanner's approach to synthesize Lyapunov-stable adaptation law was taken, replacing the deadzone with assumed prior bounds on the maximum magnitudes of the adjustable output weights of the FLC, and projecting these weight estimates away from the bounds in order to maintain robust adaptation. The supervisory controller would only work when the system state hit the

boundary of the constraint set to force the state back within the constraint set where the adaptive component was able to approximate a nonlinear control law. The supervisory controller played the same role as sliding mode controller in Sanner's adaptive gaussian network controller. Wang further proposed a hierarchical system [Wan94a] for intelligent control using the dual roles of fuzzy logic, both as symbolic rule-based system and as numerical nonlinear approximator. Fuzzy logic systems could be implemented at all control levels, from high level planning, middle level supervision, to low level servo control. This unified the mathematical tools used for all levels.

Inspired by the work on adaptive gaussian network control of Sanner [SS92b], Su [SS94b] proposed a very similar stable adaptive fuzzy logic control using Lyapunov synthesis method. The controller comprised linear feedback, sliding mode (to guarantee global stability), and an adaptive fuzzy logic component (to approximate nonlinear control function).

It was shown [JS93, BH93] that under some minor restrictions, the functional behavior of radial basis function networks (RBFN, or gaussian function networks) and fuzzy logic systems are actually equivalent. This functional equivalence enabled many results from RBFN to be applied to fuzzy logic systems. For example, Sanner and Slotine [SS92b, SS92a, SS94a] have extensively studied gaussian networks for adaptive nonlinear control, including universal function approximation, adaptation law satisfying Lyaponuv stability, learning of gaussian function centers and norm weights, "curse of dimensionality" problem. Poggio [PG90] also has extensive studies about RBF, including its function approximation, backpropagation learning algorithm for adjusting output weights as well as function centers and norm weights. All these results coming from studies of RBFN could be directly or indirectly applied to fuzzy logic systems. Indeed, many researchers have already done so, proposing various Fuzzy Radial Basis Function Networks [NL93] or Fuzzy Membership Function Based Neural Networks [IK94]. Stable adaptive fuzzy logic control proposed by Su [SS94b] was directly inspired from Sanner's work on stable adaptive gaussian network control [SS92b]. Stable adaptive fuzzy logic control proposed by Wang [Wan93, Wan94b] was directly inspired from work on stable nonlinear neural network control [PI91], and was also similar to Sanner's adaptive gaussian network control. Orthogonal Least squares learning for determining fuzzy system structure proposed by Wang [WM92b] was also based

on the similar algorithm for RBFN [CCG91].

It appears that early work in fuzzy logic control systems using pure fuzzy logic, as reviewed in [Lee90], has been gradually replaced by very active research in adaptive neuro-fuzzy systems. The following are some excellent works in this new field: Bruske [Bru93] gave a broad and in-depth review on various neuro-fuzzy systems, and proposed a new Neural Fuzzy Decision System; Wang [WM92c, WM92b, Wan92, WM92a, Wan93, Wan94b, Wan94a] reported extensive and in-depth studies of neuro-fuzzy systems, including the advantages of using neuro-fuzzy systems, universal approximation, rule extraction from examples, backpropagation, orthogonal least squares learning for structure determination, stable adaptive control using Lyapunov synthesis, hierarchical systems. Part of Wang's work was reviewed in [Men95]; Jang [Jan92, Jan93, JS93, Jan94] proposed a flexible AFNIS (Adaptive Network Based Fuzzy Inference Systems) model which could deal with plants that could be represented in a piecewise-differentiable format, such as difference equations, fuzzy models, and neural networks. AFNIS and other neuro-fuzzy models were reviewed in [JS95], along with static pattern classifier type learning algorithms such as backpropagation(BP), Least Squares Estimation (LSE), and a combination of BP and LSE, as well as dynamic learning algorithms such as the direct mimicking of another working controller, inverse control, specialized learning, backpropagation through time, sliding mode control for feedback linearizable nonlinear systems, and nonlinear gain scheduling control. Jang was one of developers of MATLAB fuzzy logic toolbox [Mat95] which includes the AFNIS architecture.

The above adaptive fuzzy systems are mostly supervised learning systems. Recently, incorporating fuzzy logic with reinforcement learning has become an active research area. Fuzzy logic combined with expert knowledge could provide a good start point and reduce the learning time of the usually slow reinforcement learning process, while reinforcement learning could fine tune fuzzy logic systems without explicit teachers.

Lee and Berenji [LB89] were the first to apply reinforcement learning to fuzzy logic systems. Single-layer neural networks from Barto's model [BSA83] were used to train a fuzzy logic controller. The problem was that the fuzzy logic controller was coupled with another external neural network controller and could not operate independently as a stand-alone controller. In a new Approximate Reasoning based Intelligent Control (ERIC)

architecture [HB2b]. Berenji replaced the single layer neural networks with Anderson's model using multi-layer neural networks [And89]. The fuzzy logic controller was modified by adjustable parameters inserted between the fuzzifier, rules and the defuzzifier. These parameters could be viewed as a neural network tightly coupled with the fuzzy logic system. Another neural network served as an evaluation network to calculate internal reinforcement. The same reinforcement learning algorithms as that used in Anderson's model were also used to train two neural networks. As the learning progressed, the fuzzy logic controller was tuned using adjustable external parameters. The problem with the ARIC was still that the fuzzy logic controller was tightly coupled with an external neural network and was not a stand-alone system even after the learning was completed. Other problems were due to the drawbacks of Anderson's model, e.g. the control outputs were only binary signals, and not eligibility traces were used to cope with delayed reinforcement.

Berenji extended the ARIC to a GARIC (Generalized ARIC) architecture [Ber92], in which the fuzzy logic system functioned as an independent controller with adjustable internal parameters, instead of the external parameters used in the ARIC. This fuzzy controller was called an Action Selection Network (ASN). Another standard multi-layer neural network was used as an Action Evaluation Network (AEN) and backpropagation was used to adjust the internal parameters of these networks. A Stochastic Action Modifier similar to Gullapalli's SRV unit was used for real-valued continuous outputs. The fuzzy system in GARIC could work as a stand-alone controller once the learning was completed. The problem was that no eligibility traces were used, and there seemed to be a mistake in the formulated learning algorithm. The estimation of partial derivative of evaluation value with respect to the control action (equation (16) in [Ber92]) was not necessary in a reinforcement learning, since it was already statistically approximated by reinforcement learning itself, as shown by Williams [Wil87, Wil92]. Also, many fuzzy operators used in the GARIC scheme seemed to be unnecessarily complex. ARIC and GARIC were successfully applied to the computer simulated attitude control of the space shuttle [BLJ+93]. The adaptive fuzzy controller had similar fuel consumption efficiency as the conventional controller, with additional advantage of being able to automatically learn the control law and adapt to new requirements. In [Ber93], Berenji further replaced the multi-layer neural network used as AEN in GARIC with another fuzzy logic system. By using adaptive fuzzy systems and

incorporating prior knowledge in both AEN and ASN, learning speed was gained substantially, and a uniform integration of reinforcement learning and fuzzy logic was achieved.

Chen et al. [CLH92] proposed a self-learning fuzzy controller very similar Barto's ASE/ACE model [BSA83]. The difference was that the BOXES partitioning used for input encoding in the original ASE/ACE model was replaced with fuzzy partitioning. A simplified fuzzy controller with fuzzy singletons as outputs was used as the ASE controller. The Condition part of fuzzy system was fixed, whilst the output fuzzy singletons were adjusted by the reinforcement learning algorithm. This fuzzy controller was able to learn how to control the classic cart-pole system and a number of other unstable nonlinear plants. A similar system was also proposed by Boem et al. [BC95] for a collision avoidance goal-seeking navigation control system for use with a mobile robot. The drawback of this system was that the control action network and the performance evaluation network shared the same fuzzy input partitioning. Because the evaluation function and the action function may require different resolutions for the input space partitioning, it would be better to use two different fuzzy systems for the action and evaluation functions respectively, as Berenji did with the modified GARIC model.

Lin and Lee used two neural-network-based fuzzy logic controllers (NN-FLC) proposed in [LL91] to construct a reinforcement NN-FLC (RNN-FLC) [LL94]. One NN-FLC performed as a fuzzy controller, while the other one acted as a fuzzy performance predicator. A stochastic unit was used to generated continuous control outputs. The basic idea was similar to that in Berenji's GARIC system, only the neuro-fuzzy subsystems used for the action and evaluation functions were different. Another new feature of RNN-FLC was that it was able to modify its rule structure as well as the parameters of the membership functions.

Lee and Takagi [LT93] proposed the use of Genetic Algorithms for optimizing fuzzy systems. Genetic Algorithms [BGH89] apply the ideas from natural genetic evolution to optimization problems in the engineering field. Elements with a good performance will survive and become stronger, while elements with poor performance will gradually be eliminated. Genetic Algorithms could also be considered as a kind of reinforcement learning algorithms in the sense that good performer will be reinforced. However, unlike the commonly used reinforcement learning algorithms, the complex encoding/decoding schemes used in Ge-

netic Algorithms make it difficult to incorporate Genetic Algorithms with connectionist neural networks. Genetic algorithms have been most successfully used in rule-based Classifier Systems [BGH89] where a finite set of rules can be optimized. It is very natural to apply Genetic Algorithms to the optimization of fuzzy logic rule-based systems, for both structure and parameter optimizations [LT93, HM95].

Summary: Adaptive nonlinear control has been a challenging topic in control engineering. Emerging techniques from the developments in neural networks, fuzzy logic systems, Genetic Algorithms, and reinforcement learning provide new and promising approaches to the control of complex, non-linear, time-varying systems. Combining these adaptive nonlinear elements with control systems under the established principles of conventional adaptive nonlinear control theory is a promising approach.

2.2 Critical Review of FNS Control Systems

Having reviewed adaptive nonlinear control strategies in general, now control systems for Functional Neuromuscular Stimulation (FNS) will be reviewed in particular.

2.2.1 Hybrid FNS System

The hybrid assistive system was first suggested in 1972 by three leaders in the field of control, FNS and robotics: Tomovic, Vukobratovic and Vodovnik [TVV72]. Since then it has been demonstrated that hybrid systems have certain advantages over systems in which FNS or bracing technology is used separately. Mechanical braces can reduce the degrees of freedom by restricting the motion of anatomical joints or by allowing motion only in a certain axis and thus simplify the FNS control problem. Under certain conditions, braces can stabilize the biomechanical system and reduce the required muscle activity, and thus delay the onset of FNS-induced muscle fatigue. Bracing can compensate for insufficient muscle force due to denervation [YZ90]. External mechanical components can be used for mounting sensors and actuators. Braces with externally powered actuators can provide external torques to further reduce the requirement for muscle action and thus compensate for weak muscles. Walking aids such as crutches or rolling walkers are usually used to maintain balance during FNS walking.

Popovic and Tomovic et al. [PTS89] introduced a hybrid assistive system (HAS) with a

combination of FNS and Self-Fitting-Modular-Orthosis (SFMO). The SFMO was an active mechanical brace with powered actuators to provide external torques on joints. The main features of SFMO included: a partially active, unilateral and lightweight external skeleton: the self-adjusting of brace to the body through telescopic elements: a soft interface: a modular design allowing independent application to any of six joints: cybernetic actuator units able to control both stiffness (free-locked) and motion (flexion-extension) of joints. It was found in experimental work that energy expenditure in paraplegic walking with HAS was lower than those with FNS and SFMO used alone: The gait efficiency as indicated by walking speed was the highest when HAS was used: The HAS also was found to reduce the force load on the upper extremities. They asserted that HAS was a new step in the development of neuroprostheses for locomotion and manipulation.

Solomonow [KAM+93] reported a Louisiana State University reciprocating gait orthosis (LSU-RGO) powered with FNS to provide a simple form of paraplegic locomotion, including standing up, standing and balancing, simultaneous swing-phase and contralateral push-off of the legs. LSU-RGO was a passive HKAFO (hip-knee-ankle-foot orthosis). The hip joints were connected to each other with two cables which prevented simultaneous flexion of both hips and consequent collapse, and provided force transmission from one hip to the contralateral one to make reciprocal gaits. The reciprocal mechanism could be disengaged to allow simultaneous hip flexions for sitting. The RGO allows stable, upright balance at minimum metabolic energy cost, while FNS provides power for locomotion and releases the upper extremities and trunks muscles from bearing all the burden of movements, thereby reducing the energy expenditure per unit body mass. This hybrid system has been fitted to over 5000 patients worldwide on a custom-made basis, with a US$ 12,000 average cost for hardware and training. Efforts are being made to form a manufacturing facility and to obtain FDA approval for commercial distribution.

Andrews et al. [ABB+88, KAM+93] introduced a rigid ankle-foot brace able to provide stability, without FNS activation of muscles, for 'C' standing postures. Stability was maintained as long as ground reaction vector remained anterior to the knee joint axis. Only when ground reaction vector passed through or behind the knee axis, quadriceps were stimulated to extend the knee and prevent buckling. Since the floor reaction vectors are mainly oriented ahead of the knee joint during the stance phase of walking and standing,

especially after 'C' standing postures were properly adopted by FNS subjects, this Floor Reaction Orthosis (FRO) reduced the duty cycle of activation of the extensors to a few percent, thus reducing muscle fatigue during prolonged upright activity. This FRO could either be used alone for FNS standing and walking, or used as a part in a 2-part modular hybrid FNS system including another lightweight hip/trunk reciprocating Orthosis (RO) with powered hip joint actuators and hip linking mechanism. The hip/trunk brace provided better stability in standing and walking than using FRO alone, and computer-controlled hip joint actuators provided greater controllability on biomechanical system.

Durfee et al. [DH90, Dur92] have designed a Controlled-Brake Orthosis (CBO) for regulating FNS-aided gait. The CBO was a long-legged brace with controllable friction brakes at the hip and knee joints. FNS activated muscles were used as unregulated power, and controllable brakes were used to regulate this raw power to achieve an acceptable leg trajectory.

Summary: From review of above typical hybrid FNS systems, one important question emerges, i.e. how to design FNS systems with the best effect/effort ratio. It appeared that SFMO proposed by Popovic et al. was a complex system with many attractive features. However, a complex system is usually difficult to operate, either for the human operators, or for the controlling computers. LSU-RGO with FNS reported by Solomonow was a reliable hybrid system, mainly due to the stability provided by LSU-RGO. However, LSU-RGO was a long HKAFO which braced all six joints and feet, and thus could become a heavy burden in subject's daily living. Modular hybrid FNS system proposed by Andrews et al. modularized the LSU-RGO, and thus partially overcame some drawbacks of LSU-RGO such as encumbrance, while provided the similar stability. CBO proposed by Durfee et al. was quite novel in that the brace was used to brake the leg movement, rather than to provide stability or power for leg movement as in other passive or active braces. One possible drawback of CBO was that muscles had to be activated to large levels, even to their maximums, in order to provide sufficient raw power to be regulated by brakes. If there was no enough raw power, the brakes would not help. These typical hybrid FNS systems show that longer bracing will provide better stability, and power brace will provide better controllability. However, if improvement in effect is small compared to the effort required to operate the external elements, the final effect/effort will actually decrease. A modular design is necessary to provide the

best effect/effort ratios for different cases.

2.2.2 Hierarchical FNS Control System

Hierarchical and distributed control have been adopted by some FNS researchers [CKM+88, Chi92, ABPK89]. Here the control problem is decomposed into a set of sub-control tasks. Conceptually (and somewhat arbitrarily), the FNS controller can be divided into three levels: the command processor, the coordination level controller and the actuator level [CKM+88]. The command processor translates SCI subject's commands (e.g. sit, stand, walk) into desired biomechanical outputs. The coordination level controller decomposes the complex movement instructions into stimulation patterns of individual muscle groups and coordinates the actions of actuator level controllers. A the actuator level, controllers generate stimulation signals to individual groups of muscles to achieve desired motions. Similar hierarchical and distributed structures can also be found in the natural motion control in vertebrates [Ito84, HKB93, MSW90a] albus81,kawato87,barto93,berthier93,prochazka93, and it is a natural and effective way to deal with high DOF control problems in neuromuscular systems. There is, however, no very clear line between these three FNS control levels. One type of control technique, such as rule-based control, could be used in all levels. And in some simple FNS systems, there could be only one level controller which takes care of all control tasks. In FNS field, various control strategies have been proposed and are reviewed in the following sections. These include open-loop controllers, conventional closed-loop controllers (e.g. PID, STR, MRAC, LQR), to nonconventional closed-loop control strategies such as rule-based control, neural network control, and fuzzy logic control.

Open-loop FNS Controller

Open-loop FNS controllers usually can not maintain satisfactory performance all the time. However, their effect/effort ratios, defined as functional or therapeutic effect obtained by using FNS with respect to the effort required to operate the FNS system, are not low. This is partially due to the fact that open-loop controllers do not require feedback sensors and thus avoid the tedious procedure of sensor mounting, cabling and calibration. Another reason is that open-loop controllers do not need complex calculation, thus reduce the requirement for fast real-time computing power. Indeed, most FNS stimulators (e.g. those reported in

[KAM+93]) today operate without sensory feedback. The only command input is usually manual switches from the users.

McNeal et al. [MNMT89] used an open-loop quadriceps FNS stimulator to control the freely-swing knee angle to follow a specific trajectory. An iterative trial-an-error procedure is used to manually adjust the sequences and amplitudes of stimulations. In four paraplegic subjects, it was found that on average 12.6±2.9 iterations were required to approximate the required trajectory, with final error of 2.1±0.7 degrees. Repeated responses were extremely consistent: the average difference between successive trials was less than 1 degree in 81% of the trials: stimulation sequences achieved accurate matches of the desired trajectory on subsequent days when adjusted by a simple gain factor; stimulus modulation envelopes for all four subjects were similar in sharp (although varied in amplitude) indicating that the iterative process can be shortened by starting with an averaged modulation envelope. However, there was a progressive degradation in the response when the stimulation sequence was repeated every 3 seconds for 50 trials, demonstrating the limitation of the open-loop control.

In Cleveland, Ohio, U.S.A., Marsolais and Kobetic et al. have developed a portable and laboratory-based 48-channel CWAMC-CWRU system [CKM+88, Kob94, KM94] for FNS walking controlled by preprogrammed stimulation sequences. The stimulation patterns were derived from trial-and-error experimentation, starting with known muscle EMG patterns observed during normal gait, and fine-tuned stimulus timing and amplitude manually using qualitative visual observations and quantitative evaluation of FNS gait with a motion analysis system [KSM+91, Mar91]. The users manually operated a hand switch to initiate the preset stimulation sequences for different states of paraplegic gait, based on his own visual, auditory and somatosensory feedback. When the subject began to slow down due to muscle fatigue, another stimulation pattern with higher muscle activation level was activated to regain the original speed. This system demonstrated the feasibility of FNS paraplegic walking. However, it was found that automatic detection of incipient gait events such as heel strike would be necessary to provide for timely activation of muscles in order to reduce undesired delays, improve gait quality and reduce metabolic energy expenditure. Such gait event detection would also enable use of sensor information for next step (cycle-to-cycle) adjustment of trajectory based on experiences gained by experts with

preprogrammed stimulation. Researchers at Cleveland Center are now working toward a next-generation, closed-loop FNS walking system with cycle-to-cycle fuzzy logic controller to make adjustments at each succeeding step.

Yamaguchi and Zajac [YZ90] designed an open-loop FNS controller to restore unassisted natural gait in a 3-D, 8 DOF computer model of a paraplegic. They used a computationally intensive Dynamical Programming process with trial-and-error adjustment to find suboptimal stimulation patterns for at least seven muscle-groups in each leg. An ankle-foot orthosis was found to be especially useful, as it helped to stabilize the stance leg and simplified the task of controlling the foot during swing. Although this computer simulation did demonstrate the possibility for unsupported FNS paraplegic gait, it was only limited to undisturbed, level gait and did not take into account muscle fatigue.

In additional to the above empirical trial-and-error or dynamical programming procedure, there are other, more theoretical, methods utilizing neuromuscular models to design non-linear open-loop controllers.

Hausdorff and Durfee [HD91] designed an open-loop freely-swing knee joint position controller in able-bodied subjects using quadriceps and hamstrings. By inverting the non-linear static muscle recruitment curve, linear muscle dynamics, a non-linear feedforward compensator was formulated. The model parameters were identified by a series of off-line experiments. This compensator worked well for open-loop isometric control of torque for short periods, especially if the dependence of torque on position was accounted for. However, for non-isometric position control, its performance was poor, due to the inaccurate muscle dynamic model under nonisometric conditions. The authors suggested that closed-loop control might be essential for functional restoration of gait.

Veltink et al. [VCCeB92] also tested a similar open-loop nonlinear compensator in cat limb joint control experiments. The controller compensated for the nonlinear muscle recruitment curve and a three-factor (muscle activation dynamics, angle-torque relationship, and angular velocity-torque relationship) non-linear muscle dynamics. The performance of the compensator appeared to be strongly dependent on modeling errors and could be improved by combining with PID feedback controller.

In an attempt to automate the synthesis of a stimulus map for open-loop upper-extremity FNS prostheses, Kilgore and Peckham [KP93] developed a method they called

External Moment Grasp Synthesis Map (EMGSP), which described the interaction between the active moments generated by electrical stimulation, the passive joint moments, and the resulting total joint moment and angle. The EMGSP was used to automate the development of stimulus maps for three paralyzed subjects, and the EMGSP generated grasp had lower errors than the grasp patterns developed using a rule-based qualitative method. It was found that the time dependent recruitment variations, such as fatigue, were a primary source of errors.

Based on a neurobiological model of vertebrate motor control network, Abbas [AC92, AT93] designed an adaptive feedforward FNS controller, comprising a set of coupled unit pattern generators (PG, could be simplified to a look-up table) for generating a cyclic pattern of activity, and a pattern shaper (PS) to adaptively filter the PG signals using a heterosynaptic Hebbian-type learning algorithm. This feedforward controller is supplemented by a PD feedback controller at the training phase, and the PD controller output decreased as the training progressed and could be totally disabled after training was completed. This feedforward control supplemented by simple PID feedback control is similar to Kawato's "feedback-error-learning" method [KFS87] based on a cerebellum network, which had been applied to robot control systems. (The difference is that Abbas used error directly while Kawato used PID output as training signal). Abbas evaluated his adaptive feedforward FNS controller using a computer model and the quadriceps of SCI individuals during a tracking task. The control system was demonstrated to provide automated customization for a given musculoskeletal system as well as on-line adaptation to account for changes in musculoskeletal system, such as fatigue. However, this neural network controller is not a complete open-loop system, because it required feedback error in order to learn during the training phase. After training, the feedforward control could work independently, however, if on-line adaptation to parameter changes was required, then feedback error was also necessary. Sankai [San95] proposed a similar adaptive pattern generator FNS controller using recurrent neural network trained by Genetic Algorithm.

Summary: Open-loop FNS controllers are easy to implement and do not need extra sensor mounting and cabling. They are widely used in most current FNS systems. However, some forms of closed-loop control may improve control performance. For FNS controllers using preset stimulation patterns, sensory feedback would provide necessary gait event infor-

mation for better timing to trigger preset stimulation patterns in order to avoid unnecessary muscle activation. For nonlinear feedforward compensators, extra feedback control would correct the modelling inaccuracy. Finally, to cope with time-varying properties such as muscle fatigue and potentiation, and internal or external disturbances such as change of load, closed-loop control is necessary.

Numerical Closed-loop FNS Controller

The simplest numerical closed-loop controller is the standard PID (proportional-integral-derivative) controller. Since the early 1980s, Cleveland research group has studied the PID control for FNS in animal using single muscles, in human upper extremities and lower extremities [CKM+88, Chi92]. Usually a co-stimulation mapping function was used for controlling two antagonist muscles and compensating nonlinear muscle recruitment properties [Chi92]. It was found [CKM+88] that PID feedback controllers performed well in maintaining stance stability for short durations (5-10 minutes) in the presence of external disturbances, and feedback control of hip and trunk provided the subject with an increased ability to free his hands while standing. However, stable stance duration was limited by fatigue of muscles, which increased input-output deadbands and resulted in oscillations in some cases.

Recently, Franken and Veltink et al. [FV95] has proposed a cycle-to-cycle PID control strategy to cope with slowly-varying parameters such as muscle fatigue. PID controller was not designed to correct instantaneous trajectory tracking error, but rather to compensate for slowly-varying muscle properties by computing parameterized stimulations on the basis of gait in previous cycles. It was questioned if instantaneous control of system trajectory was possible or even essential for the lower extremity FNS system. This is certainly a good question, and it might suggest that conventional numerical tracking control might not be suitable for the complex FNS systems with poor controllability. Kobetic et al. [Kob94, KM94] recently proposed the modification of FNS stimulation based on the control performance of previous gait cycle.

A common method to deal with time-varying dynamics is adaptive control. Allin and Inbar [AI86] designed a third-order model reference adaptive controller for tracking control of upper limb. Their results suggested that the adaptive controller performed at least as

well as, and generally better than a third-order feedforward controller in terms of tracking error and disturbance resistance. The feedforward controller required a lengthy a priori off-line identification procedure each time it was used. In contrast, the adaptive controller could identify parameters on-line within a few seconds with a consequent improvement in performance. The benefits of MRAC included the ease of setting up controller and continuous adaptation, and hence its great practicality.

While results from Allin and Inbar were encouraging, results from Cleveland group using adaptive controllers were not very positive. Bernotas, Crago, and Chizeck [BCC87] designed a STR type adaptive controller using recursive least squares parameter estimation. The controller was evaluated in an animal muscle. It was found that the controller effectively tracked the required commands. However, the performance of this adaptive controller showed only a minor difference with that of a pole-zero fixed parameter controller. The problems with this adaptive controller included instability due to measurement noise and reduced effectiveness caused by the absence of a persistent excitation in constant commands and saturated controller outputs. These difficulties of adaptive controller, and the remarkable robustness of a simple fixed parameter controller, suggested that adaptive controllers might be best used in conjunction with a supervisory controller to guarantee safety. Lan, Crago and Chizeck [LCC91] further compared the performance of a similar second-order STR adaptive controller and two first-order fixed parameter controllers (one using pulse-width modulation, the other using both pulse-width and stimulus-period modulation). The comparison was carried out on an animal muscle preparation for different tasks. It was found that the simplest PW controller demonstrated robust control for all tasks; The PW/SP controller performed significantly better than PW under transition situations. However, the adaptive controller did not perform significantly better than the simple PW controller. It was argued that abrupt changes such as external loading transitions limited the performance of adaptive controller.

To cope with the nonlinearities of the neuromuscular system, perturbation linearization or nonlinear feedforward compensator could be used. Successful results using perturbation linearization were reported by Khang [KZ89] and He [HLL91] in computer simulations. This kind of perturbation methods are limited to standing postural control.

Lan [LCC90] proposed a general perturbation control strategy to decompose a nonlin-

ear system into a "nonlinear nominal system" compensated by a non-linear feedforward compensator and a "perturbed linear system" controlled by a linear feedback controller.

Veltink et al. [VCCeB92] compared the performances of a nonlinear feedforward compensator, a PID feedback controller, and a PID feedback with nonlinear feedforward compensator for joint angle tracking control in an animal muscle. It was found that the performance of using feedforward nonlinear compensator depended strongly on the accuracy of the model.

del Re et al. [dKSG94] designed a self-tuning PID controller with a nonlinear compensator to deal with nonlinear and time-varying muscle dynamics. Experimental results for the control of the freely swing knee joints of able bodied persons demonstrated the effectiveness of this controller. A good correct approximation for the nonlinear muscle recruitment characteristic curve was shown to be necessary for stable control. In addition, special treatments such as an adaptation switching mechanism and a linear forgetting algorithm were also important for reliable control.

Summary: Conventional numerical control, from simple PID, linear adaptive MRAC and STR, to complex self-tuning linear adaptive control with nonlinear compensator have been applied to FNS systems. PID controllers with nonlinear co-stimulation mapping provided robust control, but only for short periods. The results of linear adaptive controllers in FNS from two independent research groups were mixed, with both positive and no-so-positive conclusions. This suggests that neuromuscular systems are complex nonlinear systems, and that conventional linear adaptive controllers may not always work well in FNS control. The results of PID feedback control with feedforward compensator suggests that the accuracy of the nonlinear function approximation is important for improving control performance. If a nonlinear model is not accurate, the use of nonlinear compensator would not improve, and may even degrade, the performance of a simple linear PID controller. In conventional numerical control field, accurate nonlinear function approximation is difficult to obtain, and depends on the accuracy of both structural identification and parameter estimation. On the other hand, as reviewed in early sections, non-conventional neural networks and fuzzy logic systems have been proven to be universal nonlinear function approximators having excellent learning capacity. Thus, incorporating neural networks and fuzzy logic systems as adaptive nonlinear elements in numerical control systems is a promising approach for nonlinear

adaptive control in general, in particular for the control of FNS systems.

Lan, Feng and Crago [LFC94] proposed a feedforward artificial neural network with output recurrent loop and input time delay for FNS control of quadriplegic arm. The optimal movement and corresponding muscle activation patterns were derived from the equilibrium point hypothesis of natural single joint movements and numerically solved using a nonlinear programming software which required computation power at the supercomputer level. The ANN was able to learn the inverse dynamic relationship for the optimal movement and generalize the learned optimal control to a class of scaled movements, and had the advantage of high speed computation (due to its parallel structure) which was critical for real-time implementation. The trained ANN open-loop controller could control the nonlinear muscle/joint model sufficiently well so as to reproduce a range of scaled optimal movements. It was also shown that the ANN controller's ability to extrapolate was speed-limited and that the ANN-produced movements might deviate from the optimal movements, particularly at the termination of the movements, due to various sources of errors. They recommended an additional on-line adaptation algorithm or feedback controller to assist in refining muscle stimulation for practical applications, and suggested the combination of an open-loop ANN pattern generator and a closed-loop controller in a perturbation controller [LCC90].

Teixeira and Chizeck et al. [TJS+91, Chi92] used an ANN to model the inverse dynamics of a nonlinear neuromuscular system for use in a feedback linearizing controller. They successfully applied the method proposed by Loparo [LT90b] to the control of FNS using a computer muscle model.

Finite State Closed-loop FNS Controller

Conventional numerical control has been quite successful in process control and robotics. However, the difficulties in its application to FNS suggest that the control of biological systems may need a totally different approach. There is neurobiological evidence to suggest that numerical error driven servo-controllers do exist in biological systems, e.g. stretch reflex [Ste80] and posture control [HLL91]. Numerical control methods could be applied to the quantitative analysis of such feedback loops. However, there is other evidence to suggest that biological motor control systems adopt distributed, non-numerical, pattern-driven control strategies [TM66, Pro93, MSW90a, HKB93, BSB93, Alb81, Bee90, BRM93].

One difference between centralized and distributed control systems is that distributed systems allow "graceful degradation" [PJW92] of performance in the situations of hardware or software failures, while such failures could have fatal consequences in centralized systems. A "graceful degradation" would provide more realistic and valid safety guarantee than the classical safety techniques relying on high quality components. Under "graceful degradation", reliable control is guaranteed with low quality sensory information. Indeed, most sensory information in biological systems are not high quality, compared to those in artificial systems such as robots. In FNS systems, the situations are similar. For example, only a limited selection of sensors are practical for FNS systems [CCNH86, And95]. Even traditional goniometers, widely used in robots, are not practical for daily usage in FNS, considering the mechanical encumbrance and tedious calibration and cabling. Usually some convenient, compact, reliable, easy-to-install sensors such as pressure sensing resistors (FSR)[ABPK89, And95], radio goniometer [And95], inclinometer, accelerometer [VFVB93, VBK+93, And95] are used in FNS. These sensory signals are not directly related to the system dynamics modelled by differential equations using angular variables, but rather are coarse, sometimes nonlinear, information reflecting the current states of the systems. It is almost impossible to design conventional numerical controllers using this sensory information.

Finite state control is a feedback control strategy without using explicit dynamic models. Finite state controllers make control decisions based on the pattern-matching in sensory feedback, rather than calculating control signals from tracking errors based on dynamic models. Pattern-driven finite state control may seem too simple from control engineering point of view, yet, there are evidences of such motor control programs existing in spinal cord, cerebellum, and motor cortex of intact animals, such as as "Central Pattern Generator (CPG)" [Ste80, Pro93, Ito84] and "Adjustable Pattern Generator (APG)" [BSB93, HKB93, MSW90a] which are preset motor programs triggered by particular sensory patterns. It is interesting to note that gait analysis [VDO92] also reflected the finite state nature of human gaits, e.g. human gait is divided into two main phases (swing and stance) with eight main events (toe-off, acceleration, midswing, deceleration, heel strike, foot-flat, midstance, heel-off).

Based on a neurobiological model, Berthier et al. [BSB93] designed an Adjustable Pat-

tern Generator neural network controller which successfully controlled a simple two degrees of freedom simulated limb. Beer et al. [Bee90] also used a biologically-based motor pattern generator to successfully control an "artificial cockroach". Srinivasan et al. [SGW92] demonstrated that a sequential artificial neural network was capable of learning periodic movement trajectories with generalization and fault tolerance abilities and suggested its use as bipedal movement pattern generator in legged robots and rehabilitation engineering.

Donner [Don87] designed a finite state controller for a real legged robot. Albus [Alb81] demonstrated that his CMAC model could become a finite-state automaton, or production rule system, if the trajectory information is directly fed back to the controller. This "stimulus-response chaining" type CMAC's have been applied to the control of robot manipulators [Alb81, HLG90]. Since the mid 1980s, the "behaviour-based robot" has become an active research topic [Bro86]. Behavior-based robots differ from traditional robots in that there is a direct mapping from sensors to actuators for each task-oriented behavior, while there are many layers (perception, modelling,planning,task execution, motor control) from sensors to actuators in traditional robot. Complex behavior can be decomposed into simpler component behaviors and designed accordingly. Recently, Mahadevan et al. [MC92] used reinforcement learning algorithms for the automatic programming of behavior-based robots, and suggested that converting a complex task into a simpler set of special-purpose reactive sub-tasks would speed up learning. In fact, reinforcement learning is based on finite state models and provides a natural approach for developing adaptive learning finite state control. As pointed out by Sutton [SBW92], reinforcement learning is a direct adaptive optimal control that learns optimal control actions based on input state patterns without going through the traditional two-step trajectory planning/tracking scheme.

In FNS and rehabilitation robotics, Tomovic and McGhee [TM66] firstly proposed finite state approach for the control of bioengineering systems in the 1960s. Later, Artificial Reflex (AR) control was proposed by Tomovic [Tom84, PTS89], and further developed into Adaptive Reflex [TPT87] and Rule-Based Control [PTS89, ABPK89]. Actually, Liberson's first FNS system for foot-drop correction was also a finite state controller. Finite state controllers have been used in prolonged FNS standing with [ABPK89, KAM+93] and without foot bracing [MVBZ92]: standing up [MVB92]; reciprocal gaits [ABPK89, CKM+88]; and swing-through gaits [Hel92].

A recent and interesting development is the application of machine learning algorithms to the automatic design of finite state or rule based FNS controllers [KA88, ABPK89, KAM89, KA89, Hel92, HVR+93, Kos95, KAP+95]. The application of Quinlan's [Qui86] ID3 type inductive machine learning to FNS control was firstly reported by Kirkwood and Andrews et al. [KA88, ABPK89]. Since then, inductive learning algorithms have been used to automatically derive FNS control rules from working examples [KA89, Hel92, Kos95, KAP+95], to automatically detect normal gait events [KAM89], and to reconstruct muscle activation patterns from kinematic data in normal gaits [HVR+93]. A disadvantage of ID3 type inductive learning is that it is an off-line mode of learning. Kostov et al. compared inductive learning (IL) algorithm with another supervised learning algorithm, Adaptive Logic Network (ALN) for the automatic design of rule-based FES control. The results suggest that the IL learned faster than the ALN, while both performed the test rapidly. An advantage of the ALN over this IL was that ALN's could be incrementally trained with new data without losing previously collected knowledge. The advantages of the IL over the ALN were that the IL produced human comprehensible decision trees and that the relative importance of each sensory contribution could be quantified. The last feature is important to relax the "curse of dimensionality" problem when machine learning is to be scaled up to large systems.

To cope with uncertainty in the training data and noise in sensor signals, Heller [Hel92] incorporated fuzzy logic weighting into inductive learning algorithms. It was found that fuzzy weighting improved classification accuracy and allowed the induction of more robust rule-sets. Ng and Chizeck [NC93] used fuzzy logic to predict FNS gait events in the presence of sensor noise. The fuzzy detector was intended for on-line estimation of FNS gait phrases, as part of a closed-loop controller for improved paraplegic locomotion [Kob94, KM94]. Popovic [Pop93] proposed a neural network with special class of fuzzy logic elements called "preferential neurons" for automatic synthesis of finite state control rules. The preferential neurons were used to estimate the relevance of each of the sensory inputs to the recognition of patterns defined as finite states.

Neural networks could also be used for finite state control. Although neural networks can not be expressed as explicit IF/THEN rules, they can still be used to generate state-action mappings. Note that finite state control is not equal to rule-based control. Rather,

finite state is an abstract mathematical model, which can be realized by production rule systems, neural networks, look-up tables, and other mapping functions.

Kostov et al. [KSAT92, Kos95, KAP+95] has applied Adaptive Logic Net(ALN), a kind of neural network using boolean trees, for automatic generation of finite state FNS controllers. It was found that ALN was able to clone the control skills from skilled subjects or therapist, and more interestingly, able to provide an early prediction of stimulation events, up to two seconds in advance.

Heller et al. [HVR+93] used multi-layer neural network (NN) and inductive learning (IL) to reconstruct muscle activities during normal gaits. It was found that NN was able to reconstruct the continuous muscle activation patterns of both muscles from one network, whereas two separate rule sets were needed for the rule-based IL. It appears that NN's had better function approximation capacity than rule-based IL. However, the disadvantage of NN was that the control rules generated by NN were implicit within the network structure and weight parameters and not easily comprehended.

Summary: Finite state control is a flexible control strategy which does not need a explicit dynamic models of the controlled plants. Finite state control could be implemented using production rule systems, neural networks, or fuzzy logic systems. Control strategies could be adaptively learned from human experts or from training data using symbolic or connectionist machine learning techniques. Finite state control could be applied to all three level control form command level, coordination level, to low actuator level. Production rule systems or fuzzy logic systems have advantages of using explicit rules, thus, are easy to incorporate expert knowledge into control rule bases, and if rules are learned from examples, they are comprehensible. However, inductive learning algorithms used in rule-based systems are off-line, batch-mode learning algorithms that are not suitable for on-line adaptive control. Neural networks have the advantage of excellent function approximation ability and could be trained on-line and incrementally. The disadvantage of neural networks is the difficulty for human to understand the learned control rules, and to incorporate prior expert knowledge. A combination of neural network and fuzzy logic rule-based systems will take the advantages from both sides to overcome the disadvantages of both sides.

Command Level FNS Controller

Top level decisions (such as sit, stand, walk) are made by FNS user. The user's commands may be passed down to FNS controller and the information about current states of FNS systems should also be fed back to the FNS user. Thus there is a man-machine interface between user and FNS controller. A practical FES system should have a natural man-machine interface. Some criteria for the design of FNS man-machine interface have been proposed [MP90]:

(1) provide reliable logical command signals that the user can consistently produce. These commands are used to turn on or off the FNS system, select from a set of movement patterns, etc. The classification accuracy should be high. Some crucial commands should have 100% accurate rate.

(2) provide repeatable proportional command signals that do not drift or change over time. These commands are used to control values of continuous parameters like position, velocity, forces, etc.

(3) not interfere with other activities of the user.

(4) provide adequate communication rate.

(5) provide sensory feedback to user.

(6) easy for user to learn to control this new interface.

(7) provide a natural extension to the user's intact motor system.

(8) provide subconscious control, receiving little attention from the user.

In low level SCI (paraplegia), hand switches are usually used in FNS to restore locomotion. Hand switches have high reliability, which is important to user's safety in locomotion. Actually, hand switches meet almost all the above listed criteria except perhaps (8). However, in high level SCI (quadriplegic) who have impaired upper limb motor functions, even hand switches can't be used. Automatic command level FNS controller is more necessary in upper limb FNS systems for individuals with severe motor deficits. However, even for paraplegics able to use hand switches, to reduce the amount of mental attention to operate the FNS systems is still important.

Various command sources have been explored as automatic intention detectors, including shoulder movement [MP90], head movement, myoelectrical (EMG) signals [GK95], voice control, electrooculography (EOG) [JH90], direct neuronal recording from motor cor-

tex [HY92], sensorimotor EEG rhythm [WM94].

Physiologically, voluntary motor control signals originate in the motor cortex. Therefore, direct use of electrical signals from the brain as FNS command sources is a very attractive idea. Wolpaw et al. [WM94] have used multichannel EEG as brain-computer communication tools for severe movement disorders. Specific sensorimotor mu rhythm (8-12Hz) was determined by fast Fourier transform and used to control vertical and horizontal cursor movements on the computer screen. Of course, moving cursor around on a screen is much simpler than controlling FNS system. The brain itself is a complex system with billions of neurons, more complex than neuromuscular systems. Coupling these two complex systems with each other won't be a simple task.

Recently, Graupe [GK95] reported the use of ART neural networks in an FDA-approved FNS ambulation system in which above-lesion surface EMG signals were used as user command sources. The EMG was preprocessed using an autoregressive (AR) parametric model. The AR parameters were classified using an on-line unsupervised learning ART neural networks. The EMG signals from paralyzed muscles were used as muscle fatigue index to modify stimulation strength.

Del Boca et al. [BP94] reported a similar application of EMG signals to FNS control. EMG signal features were first extracted by a Fourier analysis, then clustered using a fuzzy c-means algorithm. Data from unsupervised learning technique were presented to a multilayer perceptron type neural network to produce stimulus control signals. A digital signal processor was used for real-time operation. A highly accurate discrimination rate was achieved.

Summary: Basically, command level controllers are pattern classifiers. Many bioelectrical signals, including EEG and EMG, are stochastic signals. Various statistical signal processing/pattern recognition algorithms [Mam92, WS85, ZW90], from conventional linear signal processing techniques such as the Kalman Filter to nonlinear neural networks and neuro-fuzzy systems, could be used for signal enhancement, feature extraction and clustering. Fourier transform, parametric modelling could be used for pre-processing and feature extraction. Unsupervised learning algorithms such as Kohonen's Self-Organizing-Map [Koh88] or above mentioned ART are particularly useful for automatic clustering and classifying.

2.3 Hypothesis and Objectives of the Thesis

2.3.1 Genesis

This thesis is inspired by the pioneering researches in the Bioengineering Unit at the University of Strathclyde, U.K., led by Dr. Brian Andrews. Researchers in Bioengineering Unit have applied artificial intelligence techniques to the control of neuromuscular prostheses for many years [KA88, ABPK89, KAM89, KA89, Hel92, HVR+93]. Due to my research interests in biomedical engineering and adaptive signal processing [ZW90], it has been an enjoyable experience for me to further this challenging and stimulating work under supervision of Dr. Andrews.

Kirkwood and Andrews et al. [KA88, ABPK89] firstly reported the application of Quinlan's ID3 type inductive machine learning to FNS control. Kirkwood [KAM89, KA89] further developed a new inductive learning software, DISCIPLE, based on the idea of partitioning attribute space by maximizing information gain. Heller [Hel92, HVR+93] improved Kirkwood's algorithm and incorporated fuzzy weighting into a new inductive learning software EMPIRIC running on IBM-compatible PC.

At the Research Center (then Rehabilitation Technology Department) of Glenrose Rehabilitation Hospital (Edmonton, Canada), a real-time experimental FNS system based on IBM PC running OS/2 operation systems was developed by Jerry Penner [Pen92]. A finite state machine software module was incorporated in the FNS experimental control software. The EMPIRIC inductive learning was to be embedded into the finite state machine, but has not yet been implemented. I took over this project in early 1993. Besides the many problems with the then immature OS/2, another big problem was that the inductive learning was an off-line batch learning algorithm. How to obtain on-line incremental learning became one of our objectives.

Meanwhile, in the Biomedical Engineering Department (then Applied Sciences in Medicine) at the University of Alberta, Jeroen Bielen [Bie93] developed a computer model to verify the feasibility of a hybrid FNS system with power hip brace.

My first attempt, to improve the inductive learning, was to fuzzify the rule base generated by EMPIRIC algorithm, similar to that used in [TST92]. Then, optimization algorithms, including those conventional optimization algorithms available on MATLAB

and Genetic Algorithms [BGH89], were used to adjust parameters of fuzzy membership functions[Ath93, LT93]. This idea was tested on Jeroen's computer model. However, it was found that the parameter space was too big because there were usually many parameters in a fuzzy rule base. Direct optimization in the parameter space was found to be too costly, and a gradient descent method, taking advantage of known structure of fuzzy systems to reduce search space, was found to be more efficient [NHW92].

Around the same time, the intelligent control field [WS92], using artificial neural networks and fuzzy logic systems for adaptive nonlinear control had grown very rapidly since the early '90s. I have been very interested in the research of Berenji at NASA, U.S.A. [LB89, Ber92, HB2b, BLJ+93, Ber93], in which reinforcement learning algorithms were combined with fuzzy logic to design adaptive learning controllers which could improve control performance by interaction with the environment.

It seemed to me, however, that Berenji's fuzzy systems were too complex. A simplified Fuzzy Neural Network (FNN) proposed by Shibata et al.[SFK+92] using differential Gaussian membership functions made it easier to incorporate various learning algorithms in contrast to traditional fuzzy logic systems using triangular membership functions.

Influenced by these ideas, the author developed a new Adaptive Fuzzy Network (AFN) incorporating reinforcement learning algorithms and supervised learning algorithms, and applied it to the control of FNS. Our preliminary result was reported in [WA94].

2.3.2 Initial Hypothesis

The finite state model provides a flexible framework for the control of complex neural prostheses. Neuro-fuzzy systems that combine artificial neural networks and fuzzy logic, can be used to implement finite state controllers which have the capacity to incorporate a priori knowledge from human experts, fine-tuning control rules, as well as learning from examples.

Reinforcement learning control is a direct adaptive optimal control strategy [SBW92] that can be applied to automate the design of finite state neural prosthetic controllers without knowledgeable teachers. Incorporating a priori knowledge into the fuzzy rule base will speed up reinforcement learning. Reinforcement learning provides fine-tuning of handcrafted rules from human experts or automatic customization of previously learned control

rules from similar models.

2.3.3 Thesis Objectives

The following objectives were set:

(1) Through literature review, assess the potential of neural networks, fuzzy systems, and symbolic rule-based systems for adaptive nonlinear control and finite state control, and the application to control of FNS neural prostheses.

(2) Develop an adaptive fuzzy network by incorporating fuzzy logic and neural network; formulate the supervised learning and reinforcement learning algorithms for the proposed adaptive network; identify the important issues for practical application, such as suitable network structure, learning rate, function approximation and efficient exploration in reinforcement learning.

(3) Demonstrate the feasibility of applying reinforcement learning to control of a neural prosthetic system in a computer model of the swinging leg. Determine the rate of learning and assess methods to speed up reinforcement learning by incorporating a priori knowledge.

(4) Using the computer model, compare the performance of supervised learning and reinforcement learning control against open loop control in the events of time varying changes due to muscle fatigue and significant changes in model parameters reflecting different SCI subjects.

Chapter 3

Methods

This chapter describes the methods used in the computer simulation study. Section 3.1 describes the computer model of a paralyzed human leg in the swing phase of gait driven by an active hybrid prosthesis. Section 3.2 describes the Adaptive Fuzzy Network incorporating supervised and reinforcement learning mechanisms.

3.1 Biomechanical Model of the Paralyzed Swinging Leg with a Hybrid Neural Prosthesis

3.1.1 A Modular Hybrid Prosthesis

Hybrid neural prostheses incorporating FNS and mechanical bracing have potential advantages over FNS or brace used alone. Modular systems can be customized for individual SCI persons for the best effect/effort ratio. The modular hybrid system modeled in this thesis for assisting paraplegic locomotion is shown in Fig.3.1. The Floor Reaction Orthosis (FRO) provides stability during prolonged standing or the stance phase of gait without FNS activation of muscles, provided the ground reaction vector remains anterior to the knee joint axis, and thereby reduce the muscle fatigue. FRO could also compensate for foot drop during the swing phase [YZ90]. The proposed powered hip brace (PHB) will supply the torque on the hip joint to initiate forward swing phase and also provides trunk stability. FNS is used to activate quadriceps to extend the knee during standing, the stance and the terminal swing phase of gait. Crutches or other walking aids may be used to provide balance and upper-body support.

In the FNS systems which use surface electrodes [ABB+88, ABPK89, KSAT92, Kos95,

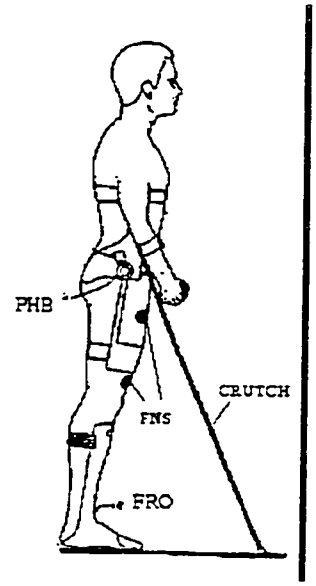


Figure 3.1: A modular hybrid prosthesis with quadriceps FNS, Floor Reaction Orthosis (FRO), and powered hip brace (PHB). Crutch provides upper body support.

KAP+95], the flexion reflex was commonly used to flex both the hip and knee joints to initiate a forward swing. The disadvantage of using the flexion reflex is poor controllability, including large variations in response, habituation and time delay [GHN+93]. An externally powered hip brace may improve the controllability of hip flexion. This actuator also has to cause sufficient knee flexion via the inertia of the lower leg to clear the foot during the swing phase. A certain amount of acceleration of the thigh is required, thus a minimum hip torque is necessary.

The feasibility of this powered hybrid system was suggested by Jeroen Bielen [Bie93] in a computer simulation study. It was found that for a person with 80kg body mass, the required hip torque was typically 25Nm for a duration of 0.25 seconds, resulting in 7W average power supply for a typical paraplegic reciprocal gait (2 seconds/cycle). One drawback of this system is that hip torque is used to flex the knee via inertial effect. It was found [Bie93] that increase of the damping in the knee drastically increased the minimum hip torque requirement. Using hamstrings muscle for knee flexion could reduce the power consumption of the hip actuator.

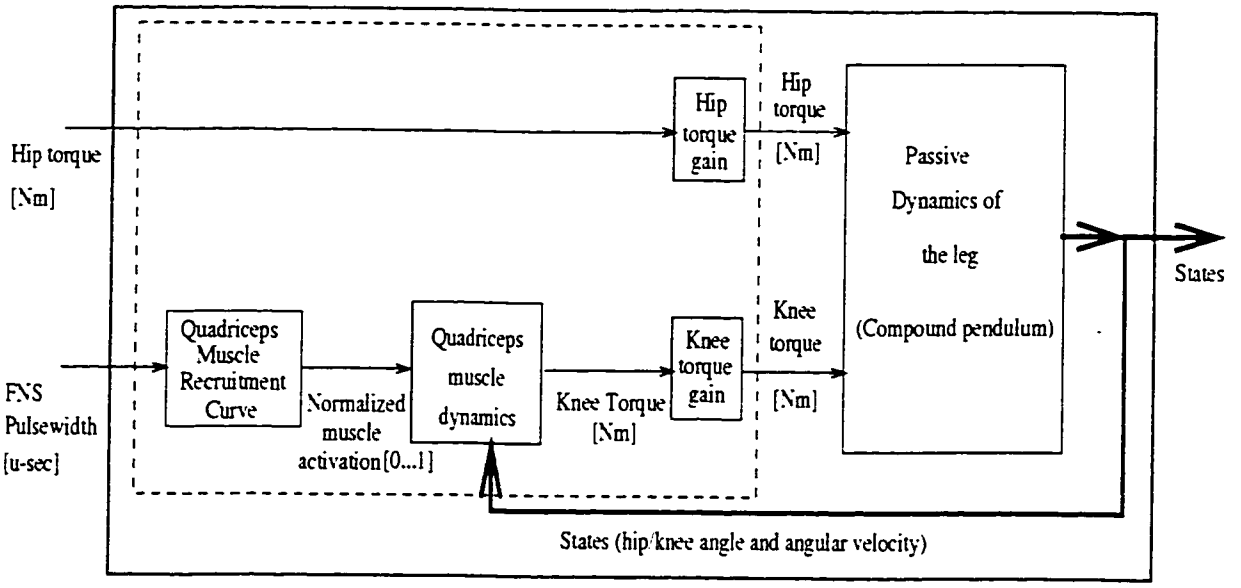


Figure 3.2: Block diagram of the swinging leg model, including a passive leg dynamics modeled as a compound pendulum, and a active muscle model (grouped by the dashed-line block).

3.1.2 Computer Model of the Swinging Leg

Only the swing phase of gait in the sagittal plane is investigated in this computer simulation. The swinging leg model can be divided into two major parts (Fig.3.2): One part is the passive leg dynamical system, representing the dynamic behavior of the leg modeled as a compound pendulum. The other part contains active components, including a nonlinear quadriceps muscle recruitment curve, three-factor (activation/angle/angular-velocity dependency) quadriceps muscle model [VCCeB92] with first order dynamics, and gain factors for both knee and hip torque. Two scalar gain factors, as a simple simulation of muscle fatigue and potentiation, are cascaded between the active torques and the passive leg dynamic system. The two control inputs are hip torque and quadriceps FNS pulsewidth, and the state outputs of the model are hip and knee angle and angular velocities. Hip torque from the powered hip brace is multiplied by a gain factor and then applied directly to the leg dynamic equation. The quadriceps FNS pulsewidth is transferred to a normalized activation by the nonlinear recruitment curve. The quadriceps muscle activation generates a knee torque which is multiplied by a gain factor and then applied to knee joint.

Three criteria for a successful swing phase are proposed: foot clearance, a certain hip

angle range (<=0.7 rad.) to get a reasonable and safe step-length, and knee extension (-0.1rad.< knee angle <0.1rad.) at the end of swing phase to prepare for body weight transfer.

The model was implemented using MATLAB software. The differential equations were solved numerically with a 2nd/3rd order Runger-Kutta method. Initial condition of the dynamical system was adopted from the GAITLAB [VDO92] data. For details of the model, see Appendix A.

In this swinging leg model, there are two control signals (hip torque and quadriceps FNS stimulation), four state signals (hip and knee angular signals), and three control objectives. With this model as testbed, various learning control strategies were investigated.

3.1.3 Optimization of an Open-loop Controller for the Swinging Leg

In a previous simulation study [Bie93], an open-loop controller was designed by optimizing parameterized control signals. The quadriceps stimulation was a simple rectangular wave with adjustable on/off timing. The hip torque was a exponential function with adjustable amplitude and on/off timing. The object function to be minimized was:

$$E(x) = \sum_{i=1}^{3} w_i \cdot (f_i(x) - t_i)^2 \tag{3.1}$$

where t_i was one of the target values of above-mentioned three control objectives, $f_i(x)$ was one of the actual control objectives obtained with control parameter set x, and w_i was a weighting factor to emphasize certain object parameters (foot clearance is more important than some deviation from the desired maximum hip angle). The SIMPLEX algorithm available in MATLAB was used to optimize the control parameters. One set of the optimized control signals for a model with 55kg body mass and 1.65m body height is shown in Fig.3.3 The swinging leg trajectory controlled by the optimized open-loop controller is shown in Fig.3.4.

As will be shown later, the performance of the open-loop controller will deteriorate or even totally fail under the situations of parameter variations simulating fatigue, potentiation, and external disturbance. Closed-loop control is should provide improved performance.

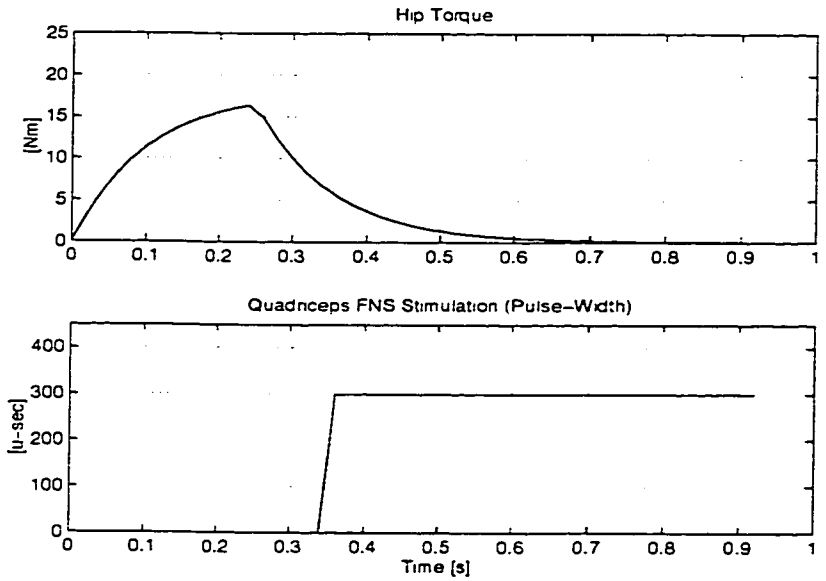


Figure 3.3: Optimized open-loop controller for the swinging leg. Two control signals are hip torque and quadriceps stimulation.

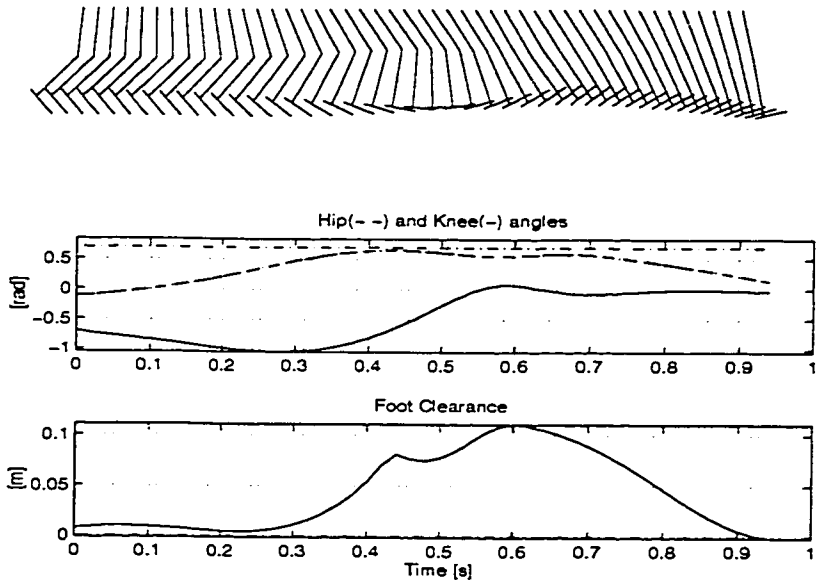


Figure 3.4: The swinging leg trajectory controlled by the optimized open-loop controller

3.2 The Adaptive Fuzzy Network (AFN)

In a narrow sense, fuzzy logic is an extension of Boolean logic. Fuzzy set theory is a theory which relates to classes of objects with unsharp boundaries in which membership is a matter of degree.

In a broader sense, fuzzy logic is a kind of so-called soft computing algorithm which includes neural network and fuzzy logic and their combination (neuro-fuzzy). Unlike traditional hard computing, soft computing is aimed at an accommodation with the pervasive imprecision of the real world. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution cost.

Artificial neural networks can be viewed as the hardware realization of fuzzy logic, just as the digital computer is the hardware realization of Boolean logic. Viewing from fuzzy logic side, such neuro-fuzzy systems play a particularly important role in the induction of rules from observations and the fine-tuning of the existing rules through training. The combination of neural network and fuzzy logic is not surprising if one realizes that fuzzy logic mimics the imprecise reasoning ability (software) of the human, while neural networks mimic the hardware of the human brain. The imprecise reasoning ability of human is the computing result of the human neural network. Indeed, Poggio [PG90] pointed out that Gaussian network, a functional equivalence [JS93] of fuzzy logic system, has similar "receptive field" to that of neurons in the brain.

In section 3.2.1, the author describes fuzzy logic in a narrow sense (fuzzy set) to introduce basic concept and terminology. Emphasis is put on the understanding of fuzzy set theory, rather than rigorous mathematical definition. In section 3.2.2, the author will introduce Adaptive Fuzzy Network (AFN), a broad sense fuzzy logic system which combines fuzzy logic and neural network. In section 3.2.3 and 3.2.4, supervised and reinforcement learning algorithms for AFN will be formulated.

3.2.1 Introduction to Fuzzy Set Theory

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth. It was meant to model the uncertainty of natural language.

Zadeh says that rather than regarding fuzzy theory as a single theory, we should regard the process of "fuzzification" as a methodology to generalize any specific theory from a crisp (discrete) to a continuous (fuzzy) form ("extension principle" [YOTN87, Lar81]). Thus recently researchers have also introduced "fuzzy calculus", "fuzzy differential equations", and so on [Lar81, YOTN87].

Fuzzy Subset

Just as there is a strong relationship between Boolean logic and the concept of a subset, there is a similar strong relationship between fuzzy logic and fuzzy subset theory.

In classical set theory, a subset U of a set S can be defined as a mapping from the elements of set S to the elements of the set 0, 1.

```
U: S --> {0, 1}
```

This mapping may be represented as a set of ordered pairs, with exactly one ordered pair present for each element of S. The first element of the ordered pair is an element of the set S, and the second element is an element of the set 0, 1. The value zero is used to represent non-membership, and the value one is used to represent membership.

Take a example. Let's talk about people and "tallness". In this case the set S (the universe of discourse) is the set of people. Let's define a Boolean subset TALL, which will answer the question "Is person x tall?". The Boolean subset TALL is defined as:

```
tall(x)={0,                    if height(x) <= 6ft.,
         1,                    if height(x) > 6ft. }
```

From this definition, we can get ordered pairs like:

```
{(4.0 0),(5.0  0) ,..., (6.0 1), (7.0 1)}
```

The truth or falsity of the statement

```
x is in U
```

is determined by finding the ordered pair whose first element is x. The statement is true if the second element of the ordered pair is 1, and the statement is false if it is 0. If Joe's

height is 7 ft., then the statement 'Joe is tall' is true since in the above TALL subset, we can find ordered pair (7.0 1).

Similarly, a fuzzy subset F of a set S can be defined as a set of ordered pairs, each with the first element from S, and the second element from the interval [0,1], with exactly one ordered pair present for each element of S. This defines a mapping between elements of the set S and values in the interval [0,1]. The value zero is used to represent none membership, and values in between are used to represent intermediate DEGREES OF MEMBERSHIP. The set S is referred to as the UNIVERSE OF DISCOURSE for the fuzzy subset F. Frequently, the mapping is described as a function, the MEMBERSHIP FUNCTION of F. The degree to which the statement

```
x is in F
```

is true is determined by finding the ordered pair whose first element is x. The DEGREE OF TRUTH of the statement is the second element of the ordered pair.

Let's define a fuzzy subset TALL, which will answer the question "to what degree is person x tall?". Zadeh describes TALL as a LINGUISTIC VARIABLE, which represents our cognitive category of "tallness". To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset TALL. The easiest way to do this is with a membership function based on the person's height.

```
tall(x)={0,                      if height(x) < 5ft.,
         (height(x)-5ft.)/2ft., if 5ft.<= height(x) <=7ft.,
         1,                      if height(x) > 7ft. }
```

Given this definition, here are some example order pairs:

Height	degree of tallness
3' 2"	0.00
5' 5"	0.21
5' 9"	0.38
5' 10"	0.42
6' 1"	0.54
7' 2"	1.00

Expressions like "A is X" can be interpreted as degrees of truth. If Joe's height is 5'9", then the degree of truth of statement "Joe is TALL" = 0.38.

The area under which membership function has nonzero value is called SUPPORT SET. If support set is only one single value, this fuzzy subset is called FUZZY SINGLETON. Fuzzy singleton only supports a single real value (SUPPORTING POINT). Its membership function is 1 in its supporting point and 0 otherwise.

Logic Operations

The standard definitions of logic operations in fuzzy logic are:

```
truth (NOT x)   = 1.0 - truth (x)
truth (x AND y) = minimum (truth(x), truth(y))
truth (x OR y)  = maximum (truth(x), truth(y))
```

The above AND, OR operations are calculated by minimum and maximum, respectively. There are other variations of the AND and OR operations, such as following product and sum:

```
truth (x AND y) = product (truth(x), truth(y))
truth (x OR y)  = sum     (truth(x), truth(y))
```

Note that if you place just the values zero and one into these definitions, you get the same truth tables as you would expect from conventional Boolean logic. This is known as the EXTENSION PRINCIPLE, which states that the classical results of Boolean logic are recovered from fuzzy logic operations when all fuzzy membership grades are restricted to the traditional set 0, 1. This effectively establishes fuzzy subsets and logic as a true generalization of classical set theory and logic. In fact, by this reasoning all crisp (traditional) subsets ARE fuzzy subsets of this very special type; and there is no conflict between fuzzy and crisp methods.

An example – assume the same definition of TALL as above, and in addition, assume that we have a fuzzy subset OLD defined by the membership function:

```
old (x) = { 0,                    if age(x) <18 yr.
          (age(x)-18yr.)/42yr., if 18 yr.<= age(x) <=60yr.
          1,                    if age(x) >60 yr. }
```

If Joe's age is 27 years, then the degree of truth of statement "Joe is OLD" = 0.21. Using the previous calculation about Joe's TALL, we can compute the degree of truth of the following statement:

```
a = Joe is TALL and Joe is OLD
```

using minimum as AND fuzzy logic operation, and get

```
truth( (0.38) AND (0.21) ) =  minimum (0.38, 0.21) = 0.21
```

Fuzzy Rule

Another basic concept in FL, which plays a central role in most of its applications, is that of a fuzzy if-then rule or, simply, fuzzy rule. The fuzzy rules are usually of a form similar to the following:

```
if x is LOW and y is HIGH then z is MEDIUM
```

where x and y are input variables, z is an output variable, LOW is a fuzzy subset defined on x, HIGH is a fuzzy subset defined on y, and MEDIUM is a fuzzy subset defined on z. The antecedent (the rule's premise) describes to what degree the rule applies, while the conclusion (the rule's consequent) assigns a fuzzy subset to each of one or more output variables. The set of rules in a fuzzy expert system is known as the fuzzy rulebase or knowledge base.

Fuzzy System

A fuzzy system is an expert system that uses a collection of fuzzy membership functions and fuzzy rulebase to reason about data. The general inference process proceeds in three or four steps. In the following, the inference process of the fuzzy system will be explained, along with a simple example used for intuitive understanding.

Assume that the variables x, y, and z all take on values in the interval [0,10], and that the following membership functions and rules are defined for all three variables:

```
low(t)  = 1 - ( t / 10 )
high(t) = t / 10
```

In addition to above membership functions, there is following rulebase:

```
    rule 1: if x is low and y is low then z is high
(or) rule 2: if x is high and y is high then z is low
```

Suppose the actual sampled inputs are x=3.2, y=3.1. The whole fuzzy reasoning proceeds as following:

1. FUZZIFICATION: The membership functions defined on the input variables are applied to the actual input values, to determine the degree of truth for each rule premise.

```
fuzzification of input variable x: low(x)=0.68, high(x)=0.32,
fuzzification of input variable y: low(y)=0.69, high(y)=0.31.
```

2. FUZZY RULE EVALUATION (INFERENCE): The truth value for the premise of each rule is computed by applying fuzzy logic operations to the rule's premise. The truth value for a rule's premise is referred to as its FIRING STRENGTH of the rule. If a rule's premise has a nonzero firing strength, then the rule is said to FIRE. The firing strength is applied to the conclusion part of each rule. The fuzzy operator in the rule's premise parts is usually AND. If AND is calculated by minimum operation, then it is called "MIN inference". If fuzzy AND operator is calculated by product operation, then it is called "PRODUCT inference".

In our simple example, if "PRODUCT inference" is adopted, then:

```
rule 1's  firing strength = low(x) AND low(y) = 0.68 x 0.69 = 0.47
rule 2's  firing strength = high(x) AND high(y) 0.32 x 0.31 = 0.0992
```

In the rulebase, each rule's consequent part associates one entire fuzzy subset to each output variable. However, if the fuzzy subset for the output variable is a fuzzy singleton, then single supporting value is associated.

```
rule 1's consequent part associates subset ''high'' to output z
rule 2's consequent part associates subset ''low'' to output z
```

Now, the rule premise's firing strength is used to clipped off (multiply) membership function associated with the output to assign a new fuzzy subset to each output variable. Assume "PRODUCT inference",then:

```
rule 1 assigns new subset (0.47''high'') to z
rule 2 assigns new subset (0.0992''low'')  to z
```

The membership functions of the new fuzzy subsets are calculated by multiplying the old membership functions with rule's firing strengths, as follows:

```
MF of subset (0.0992''low'') = 0.0992 x low(t) = 0.0992 (1 - ( t / 10 ))
MF of subset (0.47''high'')  = 0.47 x high(t)  = 0.47 ( t / 10 )
```

3. COMPOSITION: All of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. Composition solves the conflict between simultaneously firing rules. The logic relationship between fuzzy rules is OR, although usually not explicitly indicated in the fuzzy rule base. The fuzzy OR operator could be calculated by either maximum or sum operation, resulting in two COMPOSITION RULES: MAX composition and SUM composition. In MAX composition, the combined output fuzzy subset is constructed by taking the pointwise maximum over all of the fuzzy subsets assigned to the output variable by the inference rule. In SUM composition, the combined output fuzzy subset is constructed by taking the pointwise sum over all of the fuzzy subsets assigned to the output variable by the inference rule. Note that this can result in truth values greater than one! For this reason, SUM composition is only used when it will be followed by a defuzzification method, such as the CENTROID method, that does not have a problem with this odd case. Otherwise SUM composition can be combined with normalization and is therefore a general purpose method again.

Step 2 and 3 can be combined as one inference procedure. Term "MIN-MAX inference" means the combination of MAX composition and MIN inference. Term "PRODUCT-SUM inference" means the combination of SUM composition and PRODUCT inference. Term "PRODUCT-SUM-NORMALIZATION inference" means the combination of SUM-NORMALIZATION composition and PRODUCT inference. The use of term "inference" is quite confused. Sometimes it refers to only step 2 (rule evaluation), sometimes refers to both step 2 and step 3 (rule evaluation and composition). This confusion could be cleared by adding an adjective before inference, e.g. "PRODUCT inference" only means step 2 rule evaluation using PRODUCT, while "PRODUCT-SUM inference" means both

rule evaluation and composition. Another new term for this inference procedure is APPROXIMATION REASONING.

Enough for new terms now. Let's check our simple example for composition: fuzzy subsets (0.47"high") and (0.0992"low") assigned to the output z by rule 1 and rule 2 are combined to form a single new subset, [(0.47"high")OR(0.0992"low")], whose membership function is determined by combining the membership functions of subset (0.47"high") with subset (0.0992"low").

4. DEFUZZIFICATION: Usually it is necessary to convert the composed output fuzzy subset to a crisp number. There are many defuzzification methods [Lee90, JS95, Men95]. The most common technique is the CENTROID method. The crisp value of the output variable is computed by finding the variable value of the center of gravity of the membership function for the new composed fuzzy subset in step 3.

In our example, the composed output fuzzy subset [(0.47"high")OR(0.0992"low")] has a complex membership function. It is not difficult to calculate it by following above steps. But for simple illustration purpose, assume the composed output subset, defined on output variables z, has following triangular membership function:

```
MF(z) = {   z/10,              if 0 < z < 5
            1- ( z / 10 ),     if 5 < z < 10 }
```

Then the output of defuzzification is the centroid of the function MF(z) on z=[0 10], which is 5. This real value z=5 is the decision made by the fuzzy logic system in response to the input pattern (x=3.2, y=3.1). This concludes the introduction of fuzzy logic system with simple example.

A fuzzy logic system can be implemented by hardware or software. A widely adopted block diagram is shown in Fig.3.5.

As we can see in the above procedure, a fuzzy logic system projects a numerical input pattern into fuzzy domain by fuzzification, then reasons about these input projections in the fuzzy domain using fuzzy rulebase and fuzzy inference methods. Finally, the composed new fuzzy subset by composition procedure becomes the abstraction of the numerical input pattern in the fuzzy domain. This models what happens in human perception-abstraction process. The basic concept in the fuzzy system is linguistic variable, that is, a variable whose values are words rather than numbers. In effect, much of fuzzy logic may be viewed

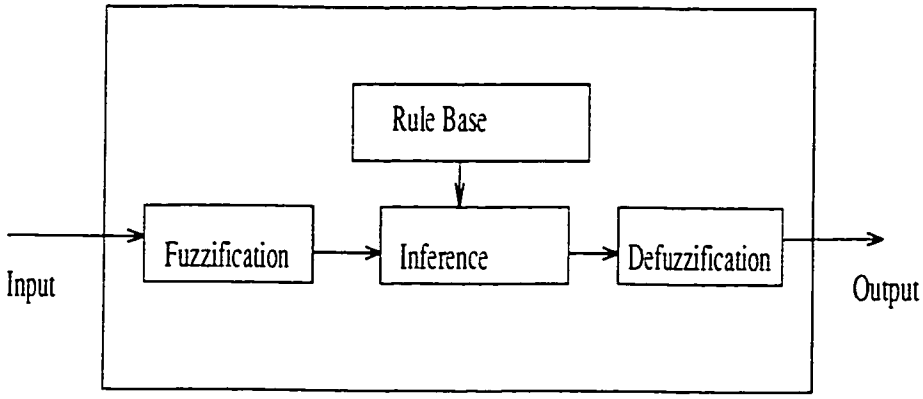


Figure 3.5: Fuzzy logic system with four basic components: rule base (including membership functions), and inference engine (input fuzzifier, fuzzy inference, and output defuzzifier).

as a methodology for computing with words rather than numbers. A word is an abstracted projection of a particular numerical input pattern in fuzzy domain (equal to human abstractive concept domain). Words are inherently less precise but more abstractive than numbers. Computing with words exploits the tolerance for imprecision, and abstraction for detail. As this fuzzy model is close to human reasoning method, it usually lowers the cost of solution when human knowledge is available. High practicality including easy to understand, simple to implement (with many microcontrollers, fuzzy VLSI chips, and fuzzy software development tools available), and inexpensive to develop [Wan93], have been major reasons for the early success of fuzzy logic in consumer products. As fuzzy logic is rapidly applied to more advanced systems, more theoretical results, under the framework of adaptive nonlinear control and neural network, have emerged [Wan93, ZHD+94, HP94, BBC+95, Men95, JS95].

3.2.2 The Adaptive Fuzzy Network (AFN)

We have introduced the basic elements of the fuzzy logic system. In the following subsections, the author will describe the Adaptive Fuzzy Network (AFN), an adaptive fuzzy logic system implemented as a layered neural network. This subsection will introduce the system structure. The learning mechanisms will be discussed in another subsection.

Rule Base of the AFN

The rule base of an AFN is shown as follows:

(SW_1)	IF	x_1 is λ_{11}	...	and	x_i is λ_{i1}	...	and	x_n is λ_{n1}	THEN	y is w_1
(SW_2)	IF	x_1 is λ_{12}	...	and	x_i is λ_{i2}	...	and	x_n is λ_{n2}	THEN	y is w_2
.....										
(SW_k)	IF	x_1 is λ_{1k}	...	and	x_i is λ_{ik}	...	and	x_n is λ_{nk}	THEN	y is w_k
.....										
(SW_m)	IF	x_1 is λ_{1m}	...	and	x_i is λ_{im}	...	and	x_n is λ_{nm}	THEN	y is w_m

Every rule has similar structure. The k-th rule has all inputs (x_i is i-th input, $i = 1...n$) in its antecedent part and only one output (y) in the consequent. Hence the AFN is a multiple inputs single output (MISO) system. Fuzzy linguistic variables (λ_{ik}, i=1...n) are defined using a generalized Gaussian membership function (Eq.3.2, Fig.3.6) for every input variable x_i ($i = 1...n$). The generalized Gaussian function is obtained by adding a slope/shape parameter a in the power term of the regular Gaussian function, as follows:

$$\Lambda_{ik}(x_i) = \frac{1}{\epsilon^{\left[\frac{(x_i - c_{ik})^2}{b_{ik}^2}\right]^{a_{ik}}}} \tag{3.2}$$

In actual applications, the fuzzy variables λ_{ik} could be linguistic terms like LARGE, MEDIUM, SMALL. Different rules can share the same linguistic variables. Therefore, the number of linguistic variables defined for one input x_i is not necessarily equal to the number of the rules (m). By optionally using the ignore keyword IG to replace one particular fuzzy variable λ_{ik} in the k-th rule, the corresponding input variable x_i will be ignored in the k-th rule and thus not be used in the reasoning procedure of that particular rule. This provides the flexible selection of different inputs for different rules. One whole rule can also be optionally disabled by selecting a software switch (SW_k, $k = 1...m$) in the rule base to test the effect of that particular rule. The output fuzzy linguistic variables assigned

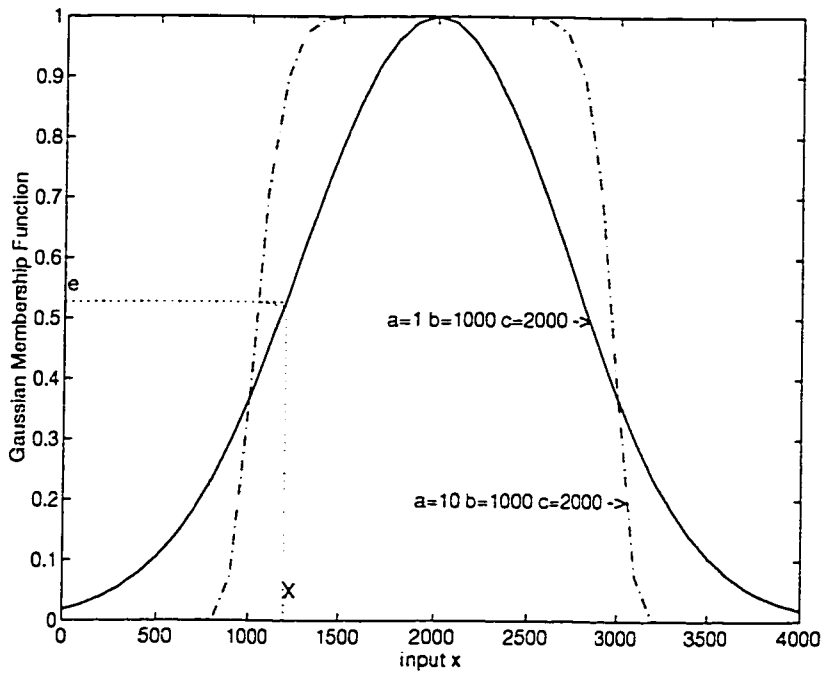


Figure 3.6: Generalized Gaussian membership function used in AFN (parameter c determines the center. b determines base width. parameter a determines slope/shape).See Eq. 3.2 for analytical form. Two Gaussian membership functions (dashed line and solid line) are shown here with two sets of parameters specified. The effect of slope/shape parameter 'a' is clearly shown. The mapping from $x- > \epsilon$ (dotted line) illustrates the fuzzification of a input x to the fitness e.

to the only output y are fuzzy singletons, i.e. a single value w_k $(k = 1...m)$. One fuzzy singleton is assigned for each rule, hence the number of output singletons is equal to the number of the rules.

Once the rule-base and membership functions are defined, the structure of the AFN is defined. However, the parameters of the Gaussian membership function can be adjusted. The inference engine of the AFN consists of three parts: fuzzifier, fuzzy inference, and composition. Since the output fuzzy variables are singletons, there is no need for defuzzification. An output scaling procedure is added after fuzzy inference.

(1) Input Fuzzification in the AFN

Fuzzification determines to what degree (fitness) a particular input x_i belongs to a fuzzy linguistic variable λ_{ik}. In the AFN, fuzzification is done by Eq.3.3. Also see Fig.3.6 for a

graphical illustration of the fuzzification.

$$\mu_{ik} = \Lambda_{ik}(x_i) = \frac{1}{\epsilon^{\left[\frac{(x_i - c_{ik})^2}{b_{ik}^2}\right]^{a_{ik}}}} \tag{3.3}$$

(2) Rule Evaluation in the AFN

In the AFN, only the fuzzy logic AND operation is used for rule evaluation. The fuzzy AND is calculated by the "product" operation. Thus, the k-th rule's firing strength α_k is the product of all fitness μ_{ik} in the rule's antecedent part, as in the following:

$$\alpha_k = \prod_{i=1}^{n} \mu_{ik} \tag{3.4}$$

(3) Output Composition in the AFN

The output of fuzzy inference is the composition of all rules' consequent parts. In the AFN, "SUM-NORMALIZATION" is used for composition. The composed output fuzzy subset is the normalized average of all output fuzzy variables w_k, weighted by each rule's firing strength α_k:

$$s = \frac{\sum_{k=1}^{m} w_k \cdot \alpha_k}{\sum_{k=1}^{m} \alpha_k} \tag{3.5}$$

In the AFN, output fuzzy variables are singletons, thus the result of the composition is a real value, rather than a fuzzy subset. Therefore, no defuzzification is needed in AFN.

(4) Output Squashing and Scaling

This layer scales the output of fuzzy inference subsystem into the domain range of output signal.

Squashing: Squash the output of fuzzy inference to [0 1] using nonlinear sigmoid function:

$$z = \frac{1}{1 + \epsilon^{-s/s_{sigmoid}}} \tag{3.6}$$

where $s_{sigmoid}$ determines the nonlinearity of the sigmoid function.

Scaling: Scale squashed output from [0 1] to domain range $[y_{center} - \frac{y_{range}}{2}, y_{center} + \frac{y_{range}}{2}]$:

$$y = (z - 0.5) \cdot y_{range} + y_{center} \tag{3.7}$$

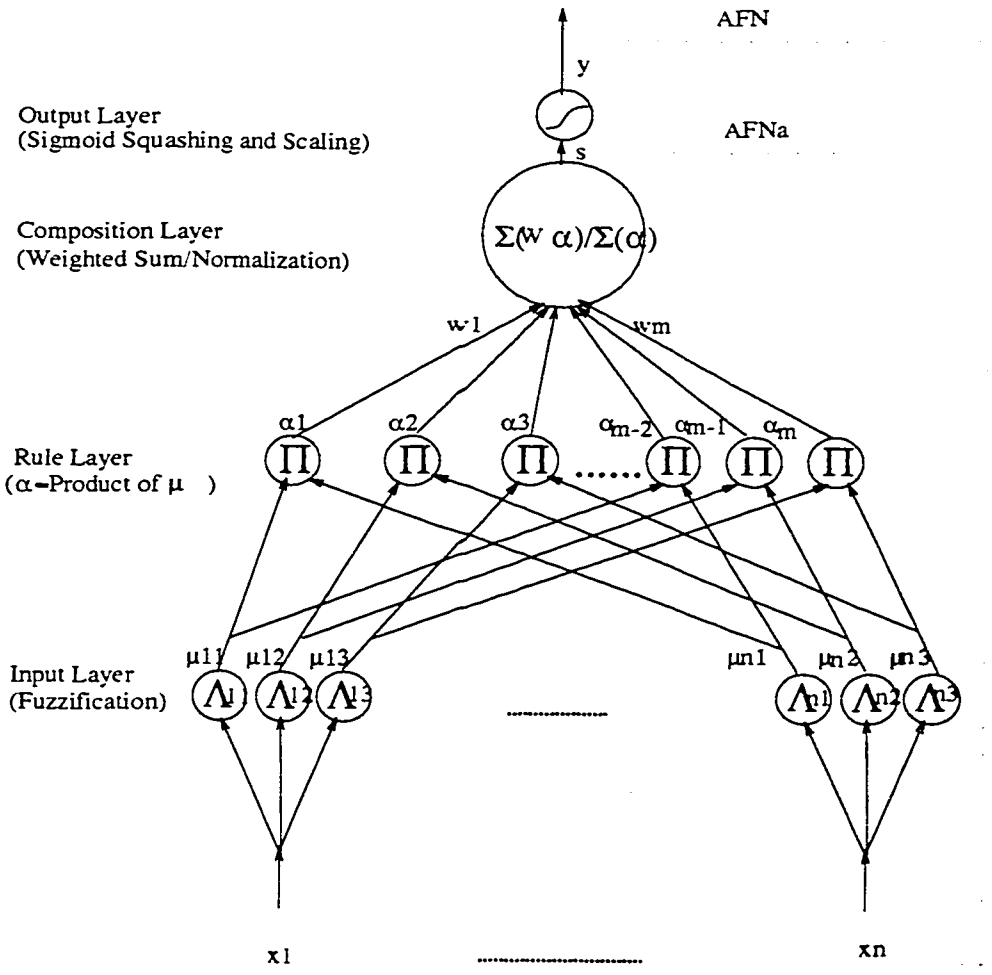


Figure 3.7: Topological structure of Adaptive Fuzzy Net (AFN). The AFNa was implemented as a three layer fuzzy inference system without output squashing/scaling layer.

This neuro-fuzzy system, called Adaptive Fuzzy Network (AFN), defines a mapping between input x_i and output y. Its transfer function is determined by all the adjustable parameters $a_{ik}, b_{ik}, c_{ik}, w_k$, fixed parameters $s_{sigmoid}, y_{range}, y_{center}$, and the rule base. The AFN system is an universal non-linear function approximator [WM92c, WM92b, Men95, Cas95, BH93]. The whole AFN network architecture is shown in Fig.3.7.

The AFN without output squashing/scaling layer is a simple fuzzy inference system, and is called AFNa for type-a AFN. The fuzzy inference sub-system (AFNa) used in the above AFN differs from commonly used fuzzy logic system as follows:

(*) use differentiable generalized gaussian function as membership function instead of triangular or trapeze membership function.

(*) use PRODUCT-SUM-NORMALIZATION fuzzy inference instead of MIN-MAX fuzzy inference, i.e. use product operation as fuzzy AND, and sum operation as fuzzy OR.

(*) use fuzzy singleton (scalar value) instead of fuzzy subsets for output variables.

(*) output fuzzy variables are singletons, the defuzzification in regular fuzzy system is not needed.

These modifications make the AFN a differentiable system, while preserving the main advantages of fuzzy logic, such as generalization (interpolation/extrapolation) and graceful degradation by using membership functions, especially in the antecedent part which partitions the input space. Differentiability of the AFN allows the use of a gradient algorithm to adjust the internal parameters.

The AFN was inspired by Shibata's Fuzzy Neural Net [SFK+92], with the modification of gaussian membership function to include adjustable slope/shape parameter, the use of SUM-NORMALIZATION composition instead of only SUM composition, and an additional output squashing/scaling layer.

The generalization of Gaussian function allows the adjustment of the shape of membership function in addition to the center and the width. The Generalized Gaussian function can approximate both triangular and trapezoid function, while regular Gaussian function can only approximate the triangular function (Fig.3.6). Using generalized Gaussian function can reduce the number of membership functions when trapeze-like functions are necessary.

The use of normalization in the composition procedure was suggested in the computer simulations. Originally, only sum operation was used for composition as in Shibata's FNN [SFK+92]. It was found in the simulation that the actual plant state might move outside the state space covered by the rule base. In such situations, the firing strengths α_k (k=1...m) of all rules were very small. If only sum operation was used (no normalization term $\frac{1}{\sum_{k=1}^{m} \alpha_k}$ in Eq.3.5), then the system output was very small, even zero. Worse yet, the learning system would not adapt its rule base to cover these out-of-range states. In the learning algorithms formulated later (Eq.3.16), it could be found that the learning rate was proportional to the firing strengths α_k. If no normalization ($\frac{1}{\sum_{k=1}^{m} \alpha_k}$) was used, small α_k would virtually stop the learning. If normalization was used, there were always some relative large firing strengths being enhanced by the normalization. Therefore, output would not become zero,

and learning would not stop. Normalization procedure played the role of enhancing filter, particularly when signals were all small.

The author also added the squashing/scaling procedures after fuzzy inference. In simulation study, this scaling procedure proved to be important in that it will limit the output to the bounded domain range. This is particularly important for stochastic reinforcement learning because it reduces the search space. Actually, in reinforcement learning controller using AFN, the stochastic search unit is inserted between the fuzzy inference sub-system (AFNa) and the output scaling layer (Fig.3.12).

In the next two subsections, the author will formulate the learning algorithms for AFN. Learning is usually the optimization of an object function J. By adjusting parameters w/a/b/c to achieve $J- > optimum$. Two types of learning algorithms, supervised and reinforcement learning, will be incorporated in the AFN.

3.2.3 Supervised Learning Algorithm for AFN

Supervised learning is the most commonly used learning algorithm. It requires a teacher to provide the desired signals. The learning controller learns to match the required signal. The difference between actual signal and desired signal is propagated back to the AFN to modify the internal parameters (Fig.3.8).

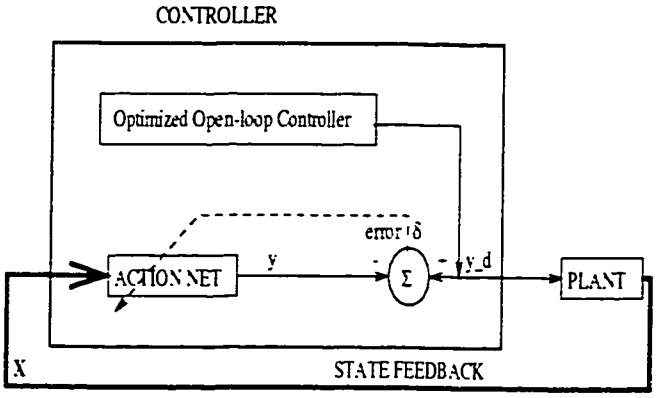


Figure 3.8: Diagram of Supervised Learning Controller. Optimized open-loop controller is used as teacher to provide desired control y_d. The ACTION NET (using AFN) maps the state feedback (X) to control y. Error between y_d and y is backpropagated to ACTION NET to modify the internal parameters of AFN.

Gradient Learning Rules

Formulated here is a supervised learning algorithm for the AFN.

In supervised learning, the object function is total squared error to be minimized:

$$J = 1/2 \cdot \sum_{k=1}^{N} (y_d(k) - y(k))^2 \tag{3.8}$$

where $y(k)_d$ is desired output, $y(k)$ is actual output of AFN, and N is number of sample points.

Parameters P (including parameters a/b/c of input membership functions and parameters w of output singletons) satisfy

$$\nabla_P J = 0 \tag{3.9}$$

is the optimal solution P_{opt}.

There is not a closed-form analytic solution of P_{opt} because of the non-linear sigmoid function and the gaussian function. Numerical methods are needed to solve P_{opt}. The

gradient algorithm is an optimization algorithm that adjusts the parameters P along the direction of gradient of J with respect to the parameter P in parameter space to minimize object function:

$$\Delta P = \tau \cdot \nabla_P J \tag{3.10}$$

where τ is a learning rate constant.

To calculate $\nabla_P J$, the chain rule for differentiation is applied:

$$\nabla_P J = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial s} \cdot \frac{\partial s}{\partial P} \tag{3.11}$$

Calculation of $\frac{\partial J}{\partial y}$:

$$\frac{\partial J}{\partial y} = \frac{1}{2} \cdot 2 \cdot (y_d(k) - y(k)) \cdot (-1) = -\delta \tag{3.12}$$

where $\delta = y_d(k) - y(k)$ is instantaneous error at time k.

In output squashing/scaling layer, there are no adjustable parameters. The error is backpropagated through this layer through multiplication by $\frac{\partial y}{\partial s}$. From eq(3.6.3.7):

$$\frac{\partial y}{\partial s} = \frac{y_{range} * (\frac{(y - y_{center})}{y_{range}} + .5) * (.5 - \frac{(y - y_{center})}{y_{range}})}{s_{sigmoid}} \tag{3.13}$$

In the composition layer, supposing parameters a,b,c of input membership functions are fixed, the Gaussian membership functions map the input space $[x_i | i = number\ of\ inputs]$ into the rule firing strength space $[\alpha_k | k = number\ of\ rules]$. This layer can therefore be viewed as a normalized linear combinator with rule firing strengths α_k as inputs.

$$s = \frac{\sum_{k=1}^{m} w_k \cdot \alpha_k}{\sum_{k=1}^{m} \alpha_k} \tag{3.14}$$

Hence:

$$\frac{\partial s}{\partial w_k} = \frac{\alpha_k}{\sum_{k=1}^{m} \alpha_k} \tag{3.15}$$

Since the object is to minimize J, the learning rate is multiplied by a negative factor (gradient descent), resulting in the following learning rule for the consequence part parameter w:

$$\Delta w_k = \tau_w \cdot \delta \cdot \frac{\partial y}{\partial s} \cdot \frac{\partial s}{\partial w_k} = \tau_w \cdot \delta \cdot \frac{\partial y}{\partial s} \cdot \frac{\alpha_k}{\sum_{k=1}^{m} \alpha_k} \tag{3.16}$$

where τ_w is learning rate; δ is error defined in eq.3.12 and $\frac{\partial y}{\partial s}$ is defined in eq.3.13

In the input fuzzification layer, the learning rules for the premise part parameters a.b.c are:

$$\Delta c_{ik} = \tau_c \cdot \delta \cdot \frac{\partial y}{\partial s} \cdot \frac{\partial s}{\partial c_{ik}} \tag{3.17}$$

$$\Delta b_{ik} = \tau_b \cdot \delta \cdot \frac{\partial y}{\partial s} \cdot \frac{\partial s}{\partial b_{ik}} \tag{3.18}$$

$$\Delta a_{ik} = \tau_a \cdot \delta \cdot \frac{\partial y}{\partial s} \cdot \frac{\partial y}{\partial a_{ik}} \tag{3.19}$$

where $\tau_{a/b/c}$ are learning rates. δ is error. $\frac{\partial y}{\partial s}$ is defined in eq.(3.13).

The formulas for $\frac{\partial s}{\partial a/b/c}$(eq.3.22 3.21 3.20) are:

$$\frac{\partial s}{\partial c_{ik}} = \frac{\partial s}{\partial \alpha_k} \cdot \frac{\partial \alpha_k}{\partial c_{ik}} = (w_k - s) \cdot \frac{\alpha_k}{\sum_{k=1}^{m} \alpha_k} \cdot 2 \cdot \frac{x_i - c_{ik}}{b_{ik}^2} \cdot a_{ik} \cdot ((\frac{x_i - c_{ik}}{b_{ik}})^2)^{(a_{ik}-1)} \tag{3.20}$$

$$\frac{\partial s}{\partial b_{ik}} = \frac{\partial s}{\partial \alpha_k} \cdot \frac{\partial \alpha_k}{\partial b_{ik}} = (w_k - s) \cdot \frac{\alpha_k}{\sum_{k=1}^{m} \alpha_k} \cdot 2 \cdot \frac{(x_i - c_{ik})^2}{b_{ik}^3} \cdot a_{ik} \cdot ((\frac{x_i - c_{ik}}{b_{ik}})^2)^{(a_{ik}-1)} \tag{3.21}$$

$$\frac{\partial s}{\partial a_{ik}} = \frac{\partial s}{\partial \alpha_k} \cdot \frac{\partial \alpha_k}{\partial a_{ik}} = (w_k - s) \cdot \frac{\alpha_k}{\sum_{k=1}^{m} \alpha_k} \cdot (-1) \cdot ((\frac{x_i - c_{ik}}{b_{ik}})^2)^{a_{ik}} \cdot ln((\frac{x_i - c_{ik}}{b_{ik}})^2) \tag{3.22}$$

Remarks on the Gradient Learning Rules

1. Equation (3.16) means the correction of output singleton w_k is proportional to the error δ and the normalized firing strength $\frac{\alpha_k}{\sum_{k=1}^{m} \alpha_k}$. This is an instantaneous steepest gradient algorithm similar to the Widrow's LMS algorithm for linear combinator [WS85, Mam92]. Statistically, this reduces the mean squares error (Eq.3.8). Normalization enhances the learning when all firing strengths are small, and also introduces a kind of competition learning mechanism into weight adjustment.

2. Adjustment of center parameter c_{ik}, Eq.(3.20), moves centers toward the data point x_i by a amount proportional to $(x_i - c_{ik})$. Statistically, function centers are clustered towards the majority of the data. However, the adjustment also depends on the error term δ in Eq.(3.17), which means the clustering also depends on the output error, not just purely depends on the input data statistical property like Kohonen's SOM[Koh88]. Hence adjustment of centers using gradient is similar to task-dependent clustering[PG90].

3. Adjustment of width parameter b_{ik} decreases the distance $\|x_i - c_{ik}\|$ in Eq.(3.21) between data point and the function center, subject to the output error δ in Eq.(3.18). Hence, it is similar to task-dependent dimensionality reduction[PG90].

4. Adjustment of slope/shape parameter a_{ik} increases the slope when data point x_i is within the receptive field of the function ($ln((\frac{x_i - c_{ik}}{b_{ik}})^2) < 0$ in Eq.3.22) to narrow the receptive field, and decreases the slope to widen the receptive when data point is outside the receptive field, subject to the output error term δ in Eq.(3.19).

5. Term $(w_k - s)$ in Eq.(3.22)(3.21)(3.20) introduces a kind of competition learning mechanism into the adjustment of parameters. Parameters in rules with above average outputs ($w_k - s > 0$) (s is average output, Eq.(3.5) are adjusted in one direction, while those in rules with below average outputs are adjusted in the opposite direction. This enhances the difference between above/below average rules. This enhancement in learning is a result of using normalization in the fuzzy inference.

More remarks about parameter adjustment in Gaussian type networks could be found in [PG90].

Layer-wise Learning Rates

The learning rate constants $\tau_w, \tau_a, \tau_b, \tau_c$ should be selected properly in order to speed up and assure the convergence of the gradient algorithm. The algorithm adjusts the parameters along the gradient of error surface (Eq.3.10). Since the shape of the error surface is determined by the $\frac{\partial J}{\partial P}$, it has different steep rates along different parameter axis. The step size should be proportional to the norm of the gradient to get the fastest convergence. Following simple example will demonstrate the importance of proper learning rate for convergence:

Consider a quadratic error surface $J = 1000 \cdot w_1^2 + w_2^2$. The minima is J=0 at point $(w_1 = 0, w_2 = 0)$. The partial derivatives are $\frac{\partial J}{\partial w_1} = 2000 \cdot w_1$ and $\frac{\partial J}{\partial w_2} = 2 \cdot w_2$. Suppose initially $(w_1 = 1, w_2 = 1)$. At this initial point, the parameter adjustments using gradient gives $\delta w_1 = \tau \cdot 2000$ and $\delta w_2 = \tau \cdot 2$. Learning rate τ should be smaller than 1/2000 so that $\delta w_1 < 1$ in order to guarantee the convergence from point (1,1) to minima (0,0). But with this learning rates, $\delta w_2 < 1/1000$, which is very slow. In a multi-layer neural network, there are lots of parameter w_i, and the $\frac{\partial J}{\partial w_i}$ will vary considerably, depending on the loca-

tion of parameters (output layer, hidden layer, input layer). But the learning rate should be small enough for ALL those w_i to guarantee the convergence and even stability of the gradient algorithm. This will slow down the learning, as demonstrated in the above only two parameters system. The solution is to use individual learning rate for every parameter w_i. An individualized learning rate with $\tau_i = \tau_0 \cdot \frac{\frac{\partial J}{\partial w_i}}{\sqrt{\sum_i (\frac{\partial J}{\partial w_i})^2}}$ would give a constant step size. With batch training and a suitable learning rate τ_0, this method would approach the minimum and then wander around in the vicinity of the minimum without converging. Another similar choice is $\tau_i = \tau_0 \cdot J \cdot \frac{\frac{\partial J}{\partial w_i}}{\sqrt{\sum_i (\frac{\partial J}{\partial w_i})^2}}$ which was used in Adaptive Backpropagation [PFA+94]. Other adaptive learning rate schemes, like Quickprop, RPROP, steepest descent with line-search, or any of the other more sophisticated methods in the numerical optimization literatures could be used. Basically, training of neural networks is an application of numerical optimization techniques. Other more efficient (thus more computationally complex) optimizations such as Marquardt algorithm [HM94], Least Squares [Jan93] could also be used. In above simple example, using individual learnings $\tau_1 = 1/2000$ and $\tau_2 = 1/2$ would get fast learning. It is not practical to assign individual learning rate for every parameter in the network. But it is possible to assign individual learning rates for different layers. Within the same layer, $\frac{\partial J}{\partial w_i}$ should be almost the same.

For the AFN, following layer-wise learning rates are formulated by using unit-analysis, i.e. the units of the learning rates should be invariant to the change of units of input and output signals.

For output singletons w_i, a factor $\frac{y_{sigmoida}}{y_{range}}$ is assigned to compensate for the term $\frac{y_{range}}{s_{sigmoid}}$ in $\frac{\partial y}{\partial s}$ (Eq.3.13). Another factor $\frac{1}{y_{range}}$ is to compensate for output error δ in Eq.(3.16) which has unit of y_{range}. A third factor $y_{sigmoida} \cdot 10$ was to scale the unit of δw_k to its domain range $y_{sigmoida}$. Combining all these factors leads to the following normalized learning rate for output singleton layer:

$$\hat{\tau}_w = \frac{\tau_w \cdot y_{sigmoida} \cdot (y_{sigmoida} \cdot 10)}{y_{range}^2} \tag{3.23}$$

Noted that the same factors are not combined or canceled for clearly showing the original factors.

Similar unit analysis can be carried out to obtain the following normalized learning

rates:

$$\hat{\tau}_c = \frac{\tau_c \cdot x_{range}^2 \cdot y_{sigmoida} \cdot (y_{sigmoida} \cdot 10)}{y_{range}^2} \tag{3.24}$$

$$\hat{\tau}_b = \frac{\tau_b \cdot x_{range}^2 \cdot y_{sigmoida} \cdot (y_{sigmoida} \cdot 10)}{y_{range}^2} \tag{3.25}$$

$$\hat{\tau}_a = \frac{\tau_a \cdot y_{sigmoida} \cdot (y_{sigmoida} \cdot 10)}{y_{range}^2} \tag{3.26}$$

Bruske[Bru93] also proposed layer-wise learning rates in a neural fuzzy system, but the approach was heuristic rather than based on the above unit-analysis.

Although the above gradient algorithm has been widely used in neural networks, its global convergence is not guaranteed due to nonlinearity. Backpropagation-like gradient algorithms are generalizations of Widrow's LMS algorithm which has been very successful in the adaptive linear filter/optimal control field. Widrow's LMS algorithm, the Recursive Least Square algorithms, and the Kalman Filter can all be related to the recursive approaches to the Wiener-Hoff optimal solution in linear stochastic systems. Analytical algorithm performances such as convergence, efficiency, biasness can be obtained for linear systems. According to the analytical result, we can also select suitable learning parameters. For example, learning rates should be selected to be smaller than maximal trace of input variables to guarantee the convergence. Although we can still use similar algorithms for the nonlinear systems, there are generally not analytical results to guarantee the performance of algorithm. The only 'universal' solutions to the non-linear system problem seems to be numerical computer simulations.

Test of AFN for Nonlinear Function Approximation

The AFN with gradient supervised learning algorithm was first tested in a nonlinear function approximation problem, as shown in the following figures. It is clear that AFN is able to approximate non-linear function. The learning rates in this example were $\tau_w = 1, \tau_a = 0.1, \tau_b = 0.1, \tau_c = 0.1$.

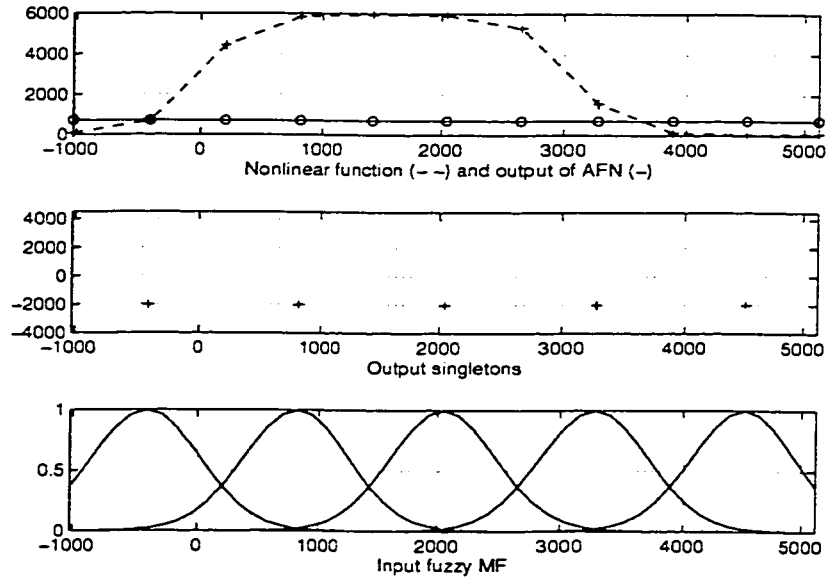


Figure 3.9: Non-linear function approximation (before training). Output singletons ('+' in the second row) were all initialized to -2000. The output of AFN (solid line in the first row) was near zero across the input range [-1000, 5000], while the required signal was the nonlinear function (dashed lines in the first row). '+' and 'o' symbols in the first and second rows were sampled data points. The third row was the input fuzzy membership functions.

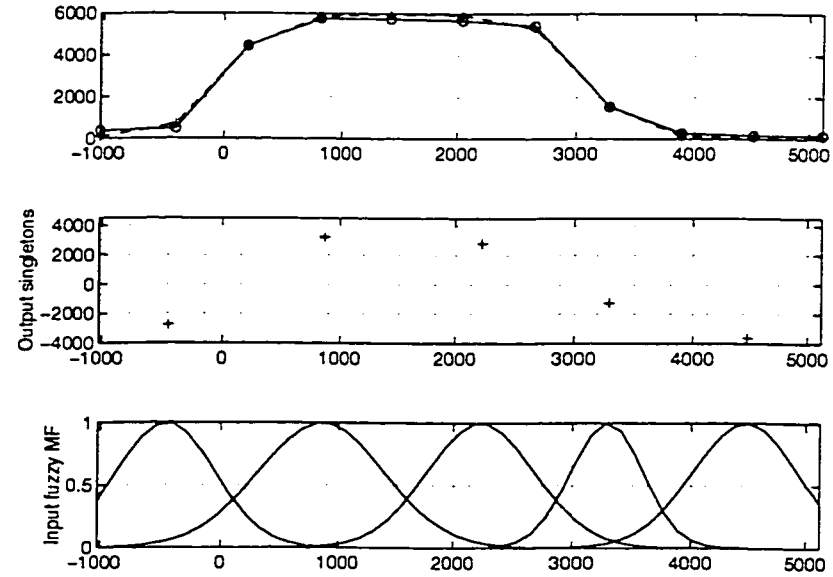


Figure 3.10: Non-linear function approximation (after training). The output fuzzy singletons (the second row) and the input fuzzy membership functions (the third row) were changed. As the result, the output of AFN (solid line in the first row) closely matched the required non-linear function (dashed lines in the first row).

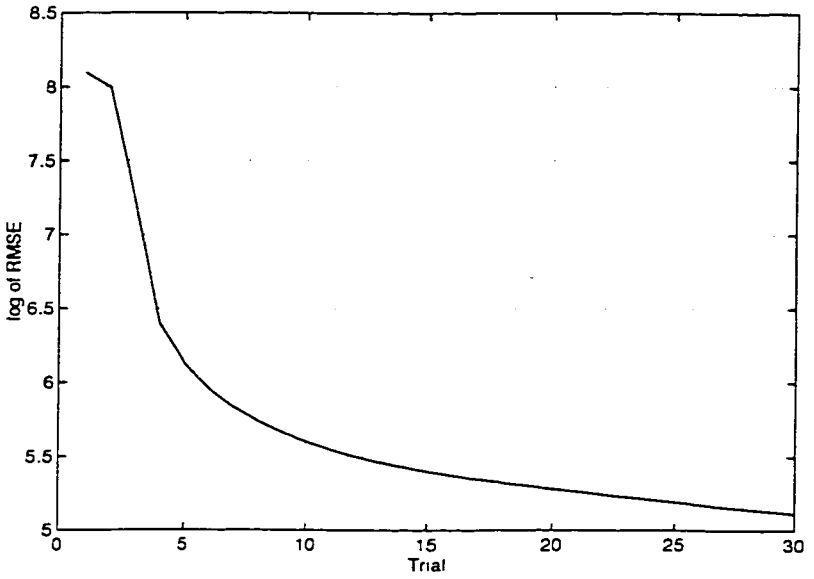


Figure 3.11: Learning curve of supervised-learning AFN for non-linear function approximation. (RMSE = Root Mean Squares Error)

3.2.4 Reinforcement Learning Algorithm for AFN

Reinforcement Learning is a new family of machine learning algorithms. It does not require explicit teacher signals. Only a binary failure signal is needed. In reinforcement learning, an agent and its training environment interact in the following manner: The agent receives a time-varying vector of inputs from the environment and sends a time-varying vector of outputs to the environment. In addition, it receives a time-varying scalar signal, called reinforcement, from the environment. The object of learning is for the network to try to maximize some function of this reinforcement signal, such as the expectation of its value on the upcoming time step or the expectation of some integral of its value over all future time. The computation of reinforcement by the environment can be anything appropriate for a particular problem and is assumed to be unknown to the learning system. In general, it is some function, stochastic or deterministic, of input patterns produced by the environment and output patterns received from the network. The basic idea of reinforcement learning is to establish a sensor-action mapping to maximize a performance index.

This reinforcement learning is contrasted to the supervised learning in which the network receives a time-varying vector signal indicating desired output from the environment, rather than the scalar reinforcement signal, and the learning object is for the network's output to match the desired output. The distinction is that the feedback provided to the network is instructive in supervised learning and evaluative in reinforcement learning.

Williams' REINFORCE Algorithm

There are several reinforcement learning algorithms, such as Barto's Adaptive Critic Element/Associative Search Element (ACE/ASE) [BSA83], Sutton's Adaptive Heuristic Critic (AHC) [Sut84], Watkins' Q learning algorithm [WD92], Williams' REINFORCE algorithm [Wil87] [Wil92], and Barto's Real-Time Dynamic Programming (RTDP) [BBS93]. Most reinforcement learning algorithms deal with discrete action problems. However, many control problems require continuous control signals. Gullapalli [Gul90] proposed the Stochastic Real-Valued (SRV) unit to learn functions with continuous outputs using a connectionist network. Williams' REINFORCE algorithm is a more general learning algorithm that can handle both discrete action and continuous action problems. In this thesis, Williams' REINFORCE algorithm is used.

The weight updating rule of REINFORCE algorithm is:

$$\Delta w = \alpha(r - b)e \tag{3.27}$$

where α is *learning rate factor*, r is reinforcement, b is *reinforcement baseline*, $e = \frac{\partial ln g}{\partial w}$ is *characteristic eligibility* of w, $g(\xi, w, x) = Pr[y = \xi | w, x]$ is the probability mass function (or probability density function for continuous-valued units) determining the value of y as a function of the weights w and inputs x.

The name REINFORCE is an acronym for "REward Increment = Nonnegative factor x Offset Reinforcement x Characteristic Eligibility", which describes the form of the algorithm.

The following theorem is the mathematical foundation of REINFORCE algorithm:

Theorem (Williams) : For any REINFORCE algorithm, the inner product of $E[\Delta W|W]$ and $\nabla_w E[r|W]$ is nonnegative. Furthermore, if $\alpha > 0$ for all individual weights, then this inner product is zero only when $\nabla_w E[r|W] = 0$. Also, if α is independent of individual weights, then $E[\Delta W|W] = \alpha \nabla_w E[r|W]$.

This theorem states that the average update vector in weight space ($E[\Delta W|W]$) lies in the direction for which reinforcement performance is increasing. A proof of this theorem can be found in ref.[Wil92]. Therefore, the REINFORCE algorithm is a stochastic gradient algorithm which climbs the gradient in stochastic sense.

While the mathematical theorem seems quite sophisticated, the intuitive understanding of the REINFORCE algorithm is very straight-forward. Take gaussian random search unit as example:

$$g(y, \mu, \sigma) = \frac{1}{(2\pi)^{1/2}} e^{-(y-\mu)^2/2\sigma^2} \tag{3.28}$$

The characteristic eligibility of μ is:

$$\frac{\partial ln g}{\partial \mu} = \frac{y - \mu}{\sigma^2} \tag{3.29}$$

And characteristic eligibility of σ is:

$$\frac{\partial ln g}{\partial \sigma} = \frac{(y - \mu)^2 - \sigma^2}{\sigma^3} \tag{3.30}$$

Hence, the parameter update rules are:

$$\Delta \mu = \alpha_\mu (r - b_\mu) \frac{y - \mu}{\sigma^2} \tag{3.31}$$

$$\Delta\sigma = \alpha_\sigma (r - b_\sigma) \frac{(y-\mu)^2 - \sigma^2}{\sigma^3} \tag{3.32}$$

This REINFORCE gaussian search unit behaves in the following way: If a randomly-sampled value y leads to a higher reinforcement than reinforcement baseline, ($r - b_\mu$ is positive), then the mean parameter μ is moved toward y (i.e. if $y > \mu$ then μ is increased, if $y < \mu$ then μ is decreased). Similarly, if a randomly-sampled value y leads to a lower reinforcement than reinforcement baseline ($r - b_\mu$ is negative), then μ is moved away from y (i.e. if $y > \mu$ then μ is decreased, if $y < \mu$ then μ is increased). In this way, the mean parameter μ is moved toward the maximum reinforcement point. The update of standard deviation parameter σ is as follow: If a randomly-sampled value y leads to a higher reinforcement than baseline, then σ will decrease if $|y - \mu| < \sigma$ and increase if $|y - u| > \sigma$. There is corresponding behavior in the opposite direction if the randomly-sampled value leads to a lower reinforcement than baseline. This search unit thus narrows the search around μ if a better point is found close to the mean or a worse point is found far from the mean; while broadening the search around μ if a worse point is found close to the mean or a better point is found far from the mean. This allows the convergence of the search when maximum point is found.

In Gullapalli's Stochastic Real Value (SRV) unit [Gul90], the mean value is updated in a similar way as in the REINFORCE algorithm, while the standard deviation is a monotonically-decreasing, nonnegative function of reinforcement signal, $\sigma(t) = s(r(t))$. This means the search scale is wide when the performance is unsatisfactory and a stochastic search will converge (standard deviation $\sigma \to 0$) after satisfactory performance is achieved.

According to this **Williams theorem**, there seems to be a possible problem in the reinforcement learning algorithm in Berenji's GARIC model [HB2b]. Berenji et al. used numerical estimation of the partial derivative $\frac{\partial r}{\partial y}$ along with the REINFORCE-type algorithm. But as analysed in the **Williams Theorem**, the REINFORCE algorithm itself already computes the gradient $\nabla_w E[r|W]$ stochastically. There is no need for estimating gradient from reinforcement r to output y if a REINFORCE-type algorithm is employed.

Temporal Difference (TD) Algorithm

In reinforcement learning or supervised learning tasks, if the evaluation or teacher signal is not available at every step but rather is given after multiple steps, the ordinary rein-

forcement learning or supervised learning algorithms can not be employed. Another case is when not only the immediate short-term payoff should be considered as in REINFORCE algorithm, but also the long-term cumulative payoff should be considered and is more important. In these situations, new learning mechanisms other than the simple REINFORCE algorithm or gradient algorithm should be incorporated. Sutton [Sut84] [Sut88] proposed the Temporal Differences (TD) algorithm for delayed multi-step prediction and long-term cumulative payoff prediction. While conventional prediction-learning algorithms assign credit using the difference between predictions and actual outcomes, the TD algorithms assign credit using the difference between temporally successive predictions.

Consider a multi-step prediction problem where prediction accuracy is not revealed at once but after multiple steps, although partial information relevant to the prediction accuracy could be revealed at each step. Examples of this multi-step prediction problem are: The outcome (win/loss/draw) of chess game is not revealed until the game is over, although partial information relevant to the win/loss/draw could be revealed at every move; The cumulative reinforcement return over a certain period is not revealed until the final step is finished, although reinforcement at every step is available. Let the states of such a multi-step prediction problem represented by a series of real-valued vectors, x_t. After a sequence of m states, $x_1, x_2, ..., x_m$, a real-valued scalar outcome z occurs. For each state-outcome sequence, $x_1, x_2, ..., x_m, z$, the learner produces a corresponding prediction sequence $P_1, P_2, ..., P_m$. Each P_t is an estimate of z based on its state x_t. Prediction P_t is a function of the state vectors x_t and a vector of modifiable weights w and can be explicitly denoted as $P(x_t, w)$. All learning algorithms are expressed as rules for updating the weight w. For every step, t, an increment to w, denoted as ΔW_t, is determined. After a complete sequence, w is changed by the sum of all the sequence's increments:

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w_t \tag{3.33}$$

As discussed in previous sections, the gradient-like supervised-learning algorithm for weight updating is:

$$\Delta w_t = \tau(z - P_t)\nabla_w P_t \tag{3.34}$$

where τ is learning rate, and $\nabla_w P_t$ is gradient of P_t with respect to w. For the special linear case where P_t is a linear function of x_t and w, i.e. $P_t = w^T x_t = \sum_i w(i)x_t(i)$, $\nabla_w P_t = x_t$ is

achieved. and equation (3.34) reduces to the well-known Widrow-Hoff LMS rule [WS85]:

$$\Delta w_t = \tau(z - w^T x_t)x_t \tag{3.35}$$

For the above supervised-learning approach. all the Δw_t in equation (3.33) depend on z. and hence can not be computed until the end of the sequence when final outcome. z. becomes available. Therefore. the weight updating equation (3.33) can not be calculated incrementally.

On the other hand. the TD learning algorithm computes the weight updating incrementally by replacing the error $z - P_t$ as a sum of changes in successive predictions as follow:

$$z - P_t = \sum_{k=t}^{m}(P_{k+1} - P_k) \tag{3.36}$$

where $P_{m+1} \stackrel{def}{=} z$ by definition.

Then. equations (3.33) and (3.34) can be combined and re-written as:

$$\begin{aligned} w &\leftarrow w + \sum_{t=1}^{m} \tau(z - P_t)\nabla_w P_t \\ &= w + \sum_{t=1}^{m} \tau \sum_{k=t}^{m}(P_{k+1} - P_k)\nabla_w P_t \\ &= w + \sum_{k=1}^{m} \tau \sum_{t=1}^{k}(P_{k+1} - P_k)\nabla_w P_t \\ &= w + \sum_{t=1}^{m} \tau(P_{t+1} - P_t)\sum_{k=1}^{t} \nabla_w P_k \end{aligned} \tag{3.37}$$

And the weight updating for each step t is:

$$\Delta w_t = \tau(P_{t+1} - P_t)\sum_{k=1}^{t} \nabla_w P_k \tag{3.38}$$

Unlike 3.34, this equation can be computed incrementally, since each Δw_t depends only on successive prediction P_{t+1}, P_t and the sum of all past gradient $\nabla_w P_t$, which are all available at time t+1. The algorithm given by (3.38) is referred to as TD(1) algorithm. It is clear that the TD(1) algorithm produces the same per-sequence weight update as the gradient supervised-learning algorithm, since equation (3.37) is simply the result of replacing $z - P_t$ with $P_{t+1} - P_t$ and re-arranging the sum operations. Although the computational procedures are not the same, as one is batch type and another is incremental type, the final result remains the same as far as the total weight changes are concerned.

The significance of TD(1) algorithms is that they use differences between successive predictions rather than the overall error between predictions and final outcome as in supervised-learning algorithm. A more general TD(λ) family of learning algorithms can be formulated by generalizing the TD(1) algorithm as follow:

$$\Delta w_t = \tau(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \tag{3.39}$$

where $0 \leq \lambda \leq 1$ is exponential decay factor. This assigns greater "eligibilities" for more recent predictions in the weight updating Δw_t. The value of the sum in (3.39) is called *eligibility trace* [BSA83] [Sut87] for weight w at time t:

$$\epsilon_t = \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \tag{3.40}$$

The eligibility trace combines both "structural credit assignment" (using gradient $\nabla_w P_k$) and "temporal credit assignment" (using weighted sum of past gradients) into one term.

The weight updating rule (3.39) can then be re-written as [Sut87] [BS92]:

$$\Delta w_t = \tau(P_{t+1} - P_t)\epsilon_t \tag{3.41}$$

This means: at each time step t, a TD error $(P_{t+1} - P_t)$ is combined with eligibility of the weight ϵ_t to determine the weight changes.

When $\lambda = 1$, TD(λ) reduces to TD(1) in which all the past predictions contribute equally to the weight updating. For $\lambda = 0$, equation (3.39) reduces to:

$$\Delta w_t = \tau(P_{t+1} - P_t)\nabla_w P_t \tag{3.42}$$

The learning at each step is driven only by two recent predictions and the recent gradient $\nabla_w P_t$.

There are eligibility weighting methods other than the exponential decay given above (e.g.[SS96]). An important advantage to the exponential form is that it can be computed recursively using only current information and previous e_t:

$$\begin{aligned} \epsilon_{t+1} &= \sum_{k=1}^{t+1} \lambda^{t+1-k} \nabla_w P_k \\ &= \nabla_w P_{t+1} + \sum_{k=1}^{t} \lambda^{t+1-k} \nabla_w P_k \\ &= \nabla_w P_{t+1} + \lambda \epsilon_t \end{aligned} \tag{3.43}$$

So far, TD algorithms have been presented as multi-step prediction methods. To apply TD to a reinforcement learning task, one only needs to define the outcome of multi-step prediction as discounted sum of future reinforcement (cumulative reinforcement) at time t:

$$z_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.44}$$

where $0 \leq \gamma \leq 1$ is the exponential *discount rate* determining the extent to which short-term and long-term reinforcements are concerned. For a perfectly accurate prediction:

$$P_t = z_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} = r_{t+1} + \gamma P_{t+1} \tag{3.45}$$

Therefore for inaccurate predictions, the mismatch or TD error is the difference between the two sides of the equation:

$$\epsilon_{t+1} = (r_{t+1} + \gamma P_{t+1} - P_t) \tag{3.46}$$

Replacing the TD error term $(P_{t+1} - P_t)$ in the pure TD algorithm (equation 3.39) with the above TD error for cumulative reinforcement, the following TD algorithm for cumulative reinforcement prediction is achieved:

$$\Delta w_t = \tau(r_{t+1} + \gamma P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \tag{3.47}$$

In the form of eligibility trace, this is:

$$\Delta w_t = \tau(r_{t+1} + \gamma P_{t+1} - P_t)\epsilon_t \tag{3.48}$$

where ϵ_t is eligibility trace defined in (3.40) and can be computed recursively as in equation (3.43).

The above TD algorithm predicts the delayed reinforcement or cumulative reinforcement by using the eligibility trace $\epsilon_t = (\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k)$ and the TD error $(r_{t+1} + \gamma P_{t+1} - P_t)$. The factor γ determines how the reinforcement is weighted according to its recency, while the factor λ determines how fast the eligibility decays according to its recency.

REINFORCE, TD and Dynamic Programming

Having introduced the REINFORCE algorithm for immediate reinforcement learning tasks and the TD algorithm for delayed (or cumulative) reinforcement prediction, REINFORCE

and TD can be combined to solve the delayed reinforcement learning tasks. These delayed reinforcement learning tasks can be viewed as sequential decision tasks in which an action selected at a given time will influence future actions and the final outcome, and both short-term and long-term consequences of decisions have to be considered. These sequential decision tasks can be formulated in terms of stochastic dynamical systems whose behaviours (could be probabilistic) unfold over time under a decision-maker's actions. The objective is to find an action strategy to maximize final outcome or cumulative long-term payoff over time. A discount factor γ can be used to weight payoffs according to their recency. The value of discount factor adjusts the degree to which long-term payoffs should be accounted for. The agent (or learning controller) uses a rule (or control strategy) called *policy* π to select actions depending on its *state* x. A policy's *return* is the weighted sum of the payoffs r if the policy π is used to select all the actions. When system is stochastic, the objective will be the *expected return*:

$$E_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] \tag{3.49}$$

where $0 \leq \gamma \leq 1$.

The *evaluation function* $V^\pi(x)$, a function of state x, is the expected return of the policy π starting from state x:

$$V^\pi(x) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | x_0 = x] \tag{3.50}$$

A policy maximizing the expected return for all possible initial states x is called an *optimal policy* π^*. Because the optimality of policies depends on the discount factor γ, it should be more precisely referred to as γ-optimal policies. In cases where payoffs are zero everywhere unless goal states (positive payoffs) or penalty states(negative payoffs) are reached, selecting actions to maximize expected return is equal to selecting actions to bring system to the goal states in the fewest steps, while avoiding penalty states.

If the accurate model of the decision task is available, Dynamic Programming (DP, [Bel57], [Kir70], [Ros83]) methods can be used to compute the evaluations of states, and to find an optimal policy. Having an accurate model of the decision task means knowing the payoff expectations, $R(x,a)$, and state-transition probabilities, $P_{xy}(a)$, for all states x and y and all actions a. Here, $R(x,a)$ and $P_{xy}(a)$ are defined as follow:

Letting x_t denote the system state at time step t, if the agent using policy π, then the

action it takes at step t is $a_t = \pi(x_t)$. The system state changes according to probability:

$$P_{xy}(a) == Prob[x_{t+1} = y | x_t = x, a_t = a] \tag{3.51}$$

And the payoff expectation

$$R(x_t, a_t) = E[r_{t+1} | x_t, a_t] \tag{3.52}$$

is the expectation of payoff received by the agent at time step t+1 if the action $a_t = \pi(x_t)$ is taken at time step t while system is in state x_t.

It is straightforward to show (e.g. [Ros83], [BSW90]) that evaluation function V^π (3.50) satisfies the following condition for each state x:

$$V^\pi(x) = R(x, \pi(x)) + \gamma \sum_{y \in X} P_{xy}(\pi(x)) V^\pi(y) \tag{3.53}$$

where X is the finite set of states x. Equation (3.53) is one of the principle equations of dynamic programming [Bel57], upon which methods for computing the evaluation function for a given policy are based.

Now we discuss how to find a optimal policy to maximize the evaluation. If the optimal evaluation function, V^*, is known, it is relatively easy to determine an optimal policy, π^*, by defining it to select an action that maximizes Eq. (3.53) for each state x. The following equation, known as *Bellman Optimality Equation*, is used to compute the optimal evaluation function:

$$V^*(x) = \max_{a \in A} [R(x, a) + \gamma \sum_{y \in X} P_{xy}(a) V^*(y)] \tag{3.54}$$

where A is the finite set of actions a. This means: *The only policies that are greedy with respect to their own evaluation value functions are optimal policies.*

The above stochastic dynamic programming method is only applicable when an accurate model of the system is known. In the absence of such a complete model, adaptive methods which can learn the underlying model have to be used. There are generally two approaches for adaptive methods: One method is *model-based* approach which constructs a model of decision task in the form of estimates of the state-transition probabilities and payoff expectations while interacting with the system. Once an accurate model is estimated, dynamic programming methods as mentioned above can be applied to find an optimal policy. The second approach is a *direct* approach which, instead of learning a model of

the decision task, adjusts the policy directly while receiving payoffs of performing various actions.

Reinforcement learning methods incorporating TD algorithm is a direct and incremental approach for learning the evaluation function $V^{\pi}(x)$ in equation (3.50) and for finding an optimal policy to maximize evaluation V^{*}.

In previous sections, we have already presented the TD algorithm (eq.3.47) for learning the cumulative reinforcement (eq.3.44). This TD procedure is closely related to the learning of evaluation function $V^{\pi}(x)$ in stochastic dynamic programming. Let us denote the estimation of evaluation function V as $\hat{V}$. From one principle DP equation (3.53), the true evaluation function V satisfies:

$$V^{\pi}(x_t) = R(x_t, \pi(x_t)) + \gamma \sum_{y \in X} P_{x_t y}(\pi(x_t)) V^{\pi}(y) \tag{3.55}$$

and the error between true value V and its estimation $\hat{V}$ is:

$$V^{\pi}(x_t) - \hat{V}_t(x_t) = [R(x_t, \pi(x_t)) + \gamma \sum_{y \in X} P_{x_t y}(\pi(x_t)) V^{\pi}(y)] - \hat{V}_t(x_t) \tag{3.56}$$

If it were possible to adjust the weights to reduce this error, the new estimation would gradually approximates the true evaluation function. However, the payoff expectations $R(x_t, \pi(x_t))$ and state-transition probabilities $P_{x_t y}(\pi(x_t))$ are unknown. To overcome this problem, instantaneous values and estimations available during on-line learning are used to replace these unknown quantities. First, substitute the payoff actually received at time t+1, which is r_{t+1}, for the expected value of this payoff, $R(x_t, \pi(x_t))$. Now, substitute the current evaluation estimation of the state actually reached, which is $\hat{V}_t(x_{t+1})$, for the expectation of the evaluation over all the reachable states, $\sum_{y \in X} P_{x_t y}(\pi(x_t)) V^{\pi}(y)$. Then, the error given by eq. (3.56) can be approximated by:

$$[r_{t+1} + \gamma \hat{V}_t(x_{t+1}] - \hat{V}_t(x_t) \tag{3.57}$$

which is the *TD error* itself as defined in equation (3.46). The different notations shouldn't cause confusion. The P in equation (3.46) is for general prediction, while $\hat{V}$ in equation (3.57) is for specific prediction of the evaluation function V.

Therefore, the TD procedure is simply a learning method using instantaneous values and estimations available during on-line learning to replace unknown expectations and

transition probabilities. For this reason, TD reinforcement learning can be called a Monte-Carlo simulation (stochastic sampling instead of a priori enumerating) of DP algorithm. It should be noted that the substitution of expectations with instantaneous values and estimations does not necessarily guarantee unbiased results.

Having obtained the TD method to approximate evaluation function, next is how to learn a optimal policy to maximize the evaluation. For correspondence to the evaluation function, the policy π is also called *action function* which maps state input x into action a ($a_t = \pi(x_t)$). The action function is, like an evaluation function, given by some parameterized model with adjustable weights w. By selecting specific values for the weights, a specific action function is determined. Therefore, the learning of an optimal action function becomes the updating of the weights of action function to maximize evaluation function. By combining the REINFORCE algorithm with the TD algorithm, we get the following weight updating rules for action function:

$$\Delta w_t = \tau(r_{t+1} + \gamma P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial ln g_k}{\partial w} \tag{3.58}$$

where $\frac{\partial ln g_k}{\partial w}$ is characteristic eligibility at time k, as defined in REINFORCE algorithm (eq. 3.27), and the sum of past characteristic eligibilities

$$\sum_{k=1}^{t} \lambda^{t-k} \frac{\partial ln g_k}{\partial w} \tag{3.59}$$

can be called the *characteristic eligibility trace* , which is a combination of *characteristic eligibility*(eq.3.27) and *eligibility trace*(eq.3.40). The characteristic eligibility trace (3.59) can be computed using recursive equation similar to Eq.(3.43).

This learning algorithm for an optimal action function is a generalized REINFORCE algorithm (eq. 3.27) which incorporates the TD(λ) procedure (eq.3.39). It uses TD error ($r_{t+1} + \gamma P_{t+1} - P_t$) as offset reinforcement (r-b), and uses a cumulative characteristic eligibility trace instead of one-step characteristic eligibility. If TD factor λ=0, it will reduce to the immediate REINFORCE algorithm(eq. 3.27). The reason for choosing TD error ($r_{t+1} + \gamma P_{t+1} - P_t$) as offset reinforcement (r-b) is as follow:

For cumulative reinforcement or evaluation function defined in eq. 3.44 and 3.50, the reinforcement term r in REINFORCE algorithm should be:

$$r_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} = r_{t+1} + \gamma P_{t+1} \tag{3.60}$$

If P_t is chosen as reinforcement baseline, the offset reinforcement (r-b) will be exactly TD error. From **Williams Theorem** for the REINFORCE algorithm, we know that the above generalized REINFORCE with the TD procedure will maximize the cumulative reinforcement or evaluation function V.

In addition to the viewpoint from REINFORCE theorem as stochastic gradient algorithm, TD error used in the learning of optimal action function can also be related to stochastic dynamic programming, just like TD error in the learning of evaluation function is related to stochastic DP. From another principle DP equation, Bellman Optimality Equation (3.54), we know that:

$$V^*(x) = \max_{a \in A}[R(x,a) + \gamma \sum_{y \in X} P_{xy}(a)V^*(y)] \tag{3.61}$$

Therefore, if we knew the payoff expectations $R(x_t, a)$, transition possibilities $P_{x_t y}$ for current state x_t and all possible next states y under all possible actions a, then we could use the current evaluation function estimation $\hat{V}_t$ to select the desired optimal action a_opt that maximizes the following term:

$$R(x_t, a) + \gamma \sum_{y \in X} P_{x_t y}(a)\hat{V}_t(y) \tag{3.62}$$

The weights of action function could then be updated by supervised-learning algorithm using the error between optimal action and actual output of action function $(a_{opt} - a_t)$. Unfortunately, payoff expectations and transition probabilities are not known, thus we can not get the desired optimal action a_{opt}. Like in the learning of evaluation function, available instantaneous values and estimations at time t can be used to replace those unknown quantities, i.e. r_{t+1} for $R(x_t, a)$, and $\hat{V}_t(x_{t+1})$ for $\sum_{y \in X} P_{x_t y}(a)\hat{V}_t(y)$. After these substitutions, the term (3.62) becomes:

$$r_{t+1} + \gamma \hat{V}_t x_{t+1} \tag{3.63}$$

which is actually the (cumulative) reinforcement return as result of action a_t in the state x_t (see 3.60). This is not necessarily the maximum reinforcement since the action a_t may not be the optimal action. To determine the performance of the action a_t, we need to compare the reinforcement return of this action with respect to average reinforcement return of all actions performed in this state. A reinforcement error factor can be formed by subtracting

the expected value (or estimation of expected value) of the reinforcement return for all actions performed in this state, from the actual reinforcement return obtained for the given action a_t. Using this reinforcement error factor, a learning rule can reward actions leading to better-than-average performance while penalize actions leading to worse-than-average performance.

Conveniently, there is an estimation for this expected (average) value of reinforcement return for all actions performed in that state x_t. This is the estimation of evaluation function at state x_t, i.e. $\hat{V}_t(x_t)$. To see why $\hat{V}_t(x_t)$ can serve this purpose, consider what happens to the TD error $[r_{t+1} + \gamma \hat{V}_t(x_{t+1}] - \hat{V}_t(x_t)$ (equation 3.57): As the TD algorithm (3.39) adjusts the weights to approximate the evaluation function, the TD error will tend to zero, therefore $\hat{V}_t(x_t)$ should approach the expected value of $r_{t+1} + \gamma \hat{V}_t(x_{t+1})$.

Consequently, the reinforcement error factor can be formed by subtracting this estimation of expected (average) reinforcement return $\hat{V}_t(x_t)$, from the actual reinforcement return obtained for the given action a_t, which is $r_{t+1} + \gamma \hat{V}_t(x_{t+1})$. The result is TD error:

$$[r_{t+1} + \gamma \hat{V}_t(x_{t+1})] - \hat{V}_t(x_t)$$

From above discussion, TD error serves as same error term in the learning of evaluation and learning of optimal action function, although it is interpreted differently. More detailed discussion about reinforcement learning and dynamic programming can be found in [BSW90].

Remarks on Eligibility Traces

There are two mechanisms, TD error $(r_{t+1} + \gamma P_{t+1} - P_t)$ (Eq.(3.46(3.57)) and eligibility traces, used in reinforcement learning to cope with delays in either reinforcement signal or control system, or both of them. Eligibility traces include one for evaluation function (Eq.(3.40)) and another characteristic eligibility trace (Eq.(3.59)) for action function.

For evaluation function learning, combining this two mechanisms leads to $TD(\lambda)$ error (Eq.(3.39)):

$$TD(\lambda) = (r_{t+1} + \gamma P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k$$

For action function learning, combining this two mechanisms and the characteristic eligi-

bility of REINFORCE leads to $TD_c(\lambda)$ error with characteristic eligibility (Eq.3.58):

$$TD_c(\lambda) = (r_{t+1} + \gamma P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial ln g_k}{\partial w}$$

It should be noted that if only reinforcement signal is delayed, then using TD error $(r_{t+1} + \gamma P_{t+1} - P_t)$ should already suffice. However, if the agent (learning controller) itself has a delay inside (from control input to trajectory output), then eligibility trace should also be used in action function learning.

A simple example will demonstrate the situation of delay in the action but not in the evaluation. Suppose a dynamical system has a time delay of 3 steps. When there is an input at time k, the system only receives this input at time (k+3) and changes its dynamics. If the agent emits a large positive action $a(k)$ at time k, and receive a large positive reinforcement $r(k+1)$ at time k+1, then using the immediate reinforcement leaning algorithm, the weight adjustment should be $\delta w = a(k) \cdot r(k+1)$, hence the parameter w should be changed in positive direction. But the problem is that the positive reinforcement $r(k+1)$ at time (k+1) has nothing to do with the action $a(k)$ at k, it is actually the result of action $a(k-2)$ at time (k-2). A simple solution could be the use of past action to pair with current reinforcement, i.e. $\delta w = a(k-2) \cdot r(k+1)$. This scheme needs the a priori knowledge of the delay.

A general approach is, as discussed above, the use of a characteristic eligibility trace (Eq.3.59) to hold the eligibility of previous action $a(k-2)$ up to the time step k. But the standard exponentially-decaying eligibility trace gives more eligibility to the most recent action $a(k), a(k-1)$ which have nothing to do with reinforcement at k $r(k+1)$. This is certainly a problem. Conceptually, the eligibility trace should have the same shape as the true temporal structure of the credit assignment. In this simple example case, the first two terms in the eligibility traces should be zero. The standard decaying trace reflects the crude idea that the more recently something has occurred the more credit it should be given. If one has better knowledge, then it is natural to build that into the shape of the trace. For example, Klopf's DR model used an inverted-U shaped eligibility kernel, roughly to match animal learning data on the effect of the inter-stimulus interval. Besides the difficulty to obtain these a priori knowledge, if a eligibility kernel other than exponentially decaying trace is used, the recursive algorithm (Eq.3.43) can not be used to calculate the eligibility trace. This recursive algorithm is important for on-line implementation.

In system identification/control field[Eyk74], there is also time-delay problems. There are several techniques to solve it. One approach is the use of off-line identification methods to identify the time-delay at first. Then use this delay in the further system identification. In reinforcement learning, there is no similar off-line identification method to identify the time-delay of the control system.

Another method used in system identification to cope with delay is the use of a high order system model to cover the possible time delay. For example, if time delay is 3 steps but not known in advance to the designer, a high 6-order system, $y(k) = a1 \cdot x(k-1) + a2 \cdot x(k-2) ... + a6 \cdot x(k-6)$, could be used to cover the unknown 3-step time delay. Parameters $a1-6$ could then be identified. It will turn out that parameters $a1, a2, a3$ are actually very small because there is no correlation between output y(k) and input x(k-1), x(k-2), x(k-3) due to the delay. So even if the exact time delay is unknown, it is still possible to use a high order system to cover the unknown time delay. In reinforcement learning, high-order means large decaying constant λ in Eq.(3.59)(3.40). Therefore, if there is a unknown time delay between system input and output, a reinforcement learning controller can use a 'high-order' system with large decaying constant λ to cover the possible time delay in control system.

Reinforcement Learning Algorithm for the AFN

The FNS swing leg control problem can be formulated in terms of a delayed reinforcement learning task. The controller receives time-varying vectors of input sensor signals (hip and knee angle information) from the swing leg model. The controller sends a vector of output control signals (hip torque and quadriceps stimulation) to the swing leg model based on the sensor inputs. Then the controller receives feedbacked reinforcements (e.g. toe collision, hip angle exceeding maximum value, etc) indicating whether the control is successful or not. These reinforcements are evaluative rather than instructive and are not available all the time.

The adaptive fuzzy logic controller for the FNS swing based on reinforcement learning consists of two major blocks (fig. 3.12):

One major block is the Evaluation Net (EN) which maps state feedback X to TD error $\hat{r}$. The function approximator used in Evaluation Net is an AFN (fig.3.7). The AFN receives state feedback X and outputs the prediction of evaluation function v. This prediction, v,

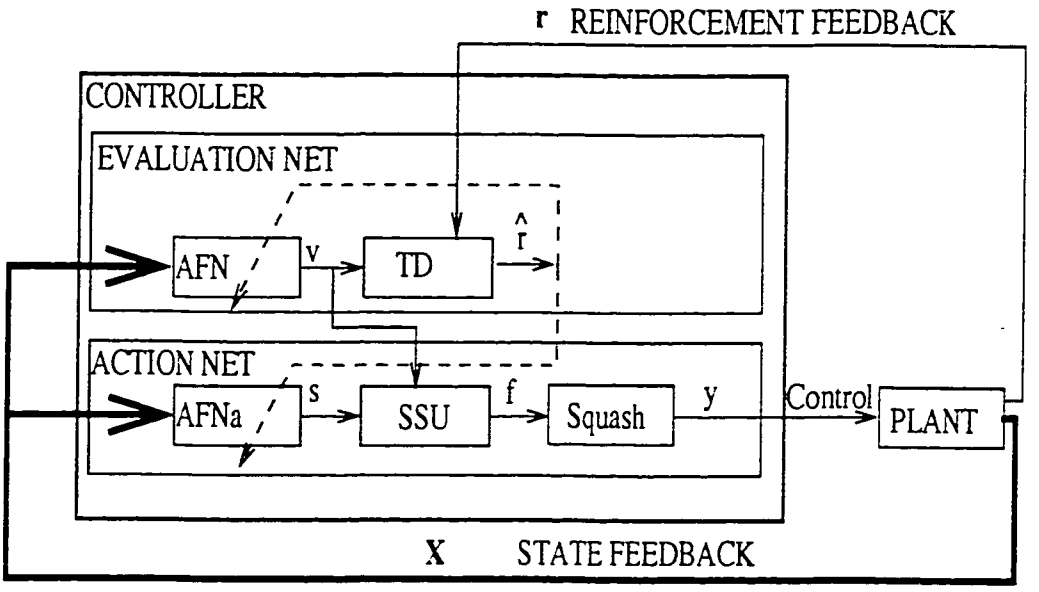


Figure 3.12: Reinforcement Learning Controller.

is combined with delayed reinforcement feedback, r, (also called *primary reinforcement*) to compute the TD error $\hat{r}$ (also called *secondary reinforcement*) according to equation 3.46 or 3.57. The TD error $\hat{r}$ is used by TD(λ) algorithm (eq. 3.47 or 3.48) to modify the weights of AFN (indicated in fig. 3.12 by dash-line). The detailed learning rules for different layer AFN parameters can be obtained by computing gradient $\nabla_w P_k$ in eq.(3.47) from equations (3.13,3.15,3.22,3.21,3.20). For following formulas, time index t is labeled as superscript and parameter index (i,k) is labeled as subscript.

For output composition layer parameters w_k, the weight update rule is:

$$\Delta w_k^t = \tau_w(r^{t+1} + \gamma v^{t+1} - v^t)\epsilon_{wk}^t \tag{3.64}$$

where τ_w is learning rate, ϵ_{wk}^t is eligibility trace (eq. 3.40) of weight w_k at time t:

$$\epsilon_{wk}^t = \sum_{l=1}^{t} \lambda^{t-l} \nabla_{w_k} v_l \tag{3.65}$$

ϵ_{wk}^t is computed recursively (3.43) as follow:

$$\begin{aligned} \epsilon_{wk}^{t+1} &= \lambda_w \epsilon_{wk}^t + \nabla_{w_k} v_{t+1} \\ &= \lambda_w \epsilon_{wk}^t + \frac{\partial v_{t+1}}{\partial s}\frac{\partial s}{\partial w_k} \end{aligned} \tag{3.66}$$

where λ_w is TD decaying factor, $\frac{\partial s}{\partial w_k}$ is computed as eq.3.15, and $\frac{\partial v_{t+1}}{\partial s}$ is computed as eq. 3.13 by using v_{t+1} to replace y.

Similarly, for the input fuzzification layer parameters a_{ik}, b_{ik}, c_{ik}, the weight update rules are:

$$\Delta a_{ik}^t = \tau_a(r^{t+1} + \gamma v^{t+1} - v^t)\epsilon_{aik}^t \tag{3.67}$$

$$\begin{aligned} \epsilon_{aik}^{t+1} &= \lambda_a \epsilon_{aik}^t + \nabla_{a_{ik}} v_{t+1} \\ &= \lambda_a \epsilon_{aik}^t + \frac{\partial v_{t+1}}{\partial s}\frac{\partial s}{\partial a_{ik}} \end{aligned} \tag{3.68}$$

$$\Delta b_{ik}^t = \tau_b(r^{t+1} + \gamma v^{t+1} - v^t)\epsilon_{bik}^t \tag{3.69}$$

$$\begin{aligned} \epsilon_{bik}^{t+1} &= \lambda_b \epsilon_{bik}^t + \nabla_{b_{ik}} v_{t+1} \\ &= \lambda_b \epsilon_{aik}^t + \frac{\partial v_{t+1}}{\partial s}\frac{\partial s}{\partial b_{ik}} \end{aligned} \tag{3.70}$$

$$\Delta c_{ik}^t = \tau_c(r^{t+1} + \gamma v^{t+1} - v^t)\epsilon_{cik}^t \tag{3.71}$$

$$\begin{aligned} \epsilon_{cik}^{t+1} &= \lambda_c \epsilon_{cik}^t + \nabla_{c_{ik}} v_{t+1} \\ &= \lambda_c \epsilon_{cik}^t + \frac{\partial v_{t+1}}{\partial s}\frac{\partial s}{\partial c_{ik}} \end{aligned} \tag{3.72}$$

where $\frac{\partial s}{\partial a_{ik}}$, $\frac{\partial s}{\partial b_{ik}}$, $\frac{\partial s}{\partial c_{ik}}$ are computed as eq. 3.22, 3.21,3.20.

The other major block in the adaptive controller is the Action Net (abbreviated as AN) which maps the state feedback X to control signal y. There are two Action Nets: one for hip torque control; another for quadriceps stimulation control. For simplicity, only one action network is shown in fig. 3.12. The function approximator used in the Action Net is a type-a AFN without an output squashing/scaling layer (AFNa, fig.3.7). The AFNa receives state feedback X and outputs recommended action, s. This recommended action, s, is feed-forwarded into Stochastic Search Unit (SSU) to produce stochastic action f. The action f is further squashed and scaled (equation 3.6, 3.7) to generate the final control signal y. The Action Net can be viewed as a normal AFN with a SSU inserted between the ANFa and output squashing/scaling layer. The Stochastic Search Unit (SSU), as its name suggests, serves as search mechanism by introducing controllable random noise into control signal. The output of SSU is a normally distributed random variable:

$$f \sim \Psi(s, \sigma) \tag{3.73}$$

where the mean value is the recommended action s from the AFNa, and the standard deviation σ is a monotonically decreasing, nonnegative function of evaluation prediction v

from EN. The control signal is therefore a random variable with mean value recommended by the AFNa and a standard deviation determined by reinforcement evaluation. The actual form of the standard deviation function, especially its scale and rate of decrease, should take the units and range of variation of the output variable into account. In this study, following function is adopted (figure 3.13):

$$\sigma = max(0., (1 - e^{v/\sigma_\tau})) \cdot \sigma_{max} \tag{3.74}$$

where σ_{max} determines the maximum value and σ_τ determines the decaying rate of exponential function. The random search scale is large when evaluation is large negative (high penalty) and small when evaluation is small negative (low penalty). When evaluation becomes positive (reward), the standard deviation becomes zero. Then, the random search will be terminated and the learning of the Action Net will also be terminated. The stochastic learning controller will converge into a deterministic controller. The choice of evaluation prediction v as the input to the deviation function is different to that in [HB2b] and is similar to that in [Gul90]. Berenji et al. [HB2b] chose TD error $\hat{r}$ as input to the standard deviation function which means the search is large when $\hat{r}$ is low (the last action is a bad move) and the search is small when $\hat{r}$ is high (the last action is a good move). But in the computer simulations, it was found that better results were achieved when the search scale was related to the current evaluation (v) instead of performance of last action ($\hat{r}$). When current evaluation is unsatisfactory (v is low), the search scale should be large to find possible good actions to escape the unsatisfactory situation. But the performance of last action could be pretty high ($\hat{r}$ is high) even if evaluation v is low. Therefore, random search scale should be linked to the evaluation v rather than TD error $\hat{r}$.

The learning algorithm for AFNa in Action Net is the REINFORCE+TD(λ) algorithm (eq. 3.58) using the TD error $\hat{r}$ from Evaluation Net (indicated in fig. 3.12 by dash-line). The detailed learning rules for different layer AFNa parameters can be obtained by computing the gradient $\frac{\partial ln g_k}{\partial w}$ in eq. (3.58) from equation (3.31)(3.13)(3.15)(3.22)(3.21)(3.20)

For output composition layer parameters w_k, the weight update rule is:

$$\Delta w_k^t = \tau_w (r^{t+1} + \gamma v^{t+1} - v^t) e_{wk}^t \tag{3.75}$$

where τ_w is learning rate, e_{wk}^t is characteristic eligibility trace (eq. 3.59) of weight w_k at

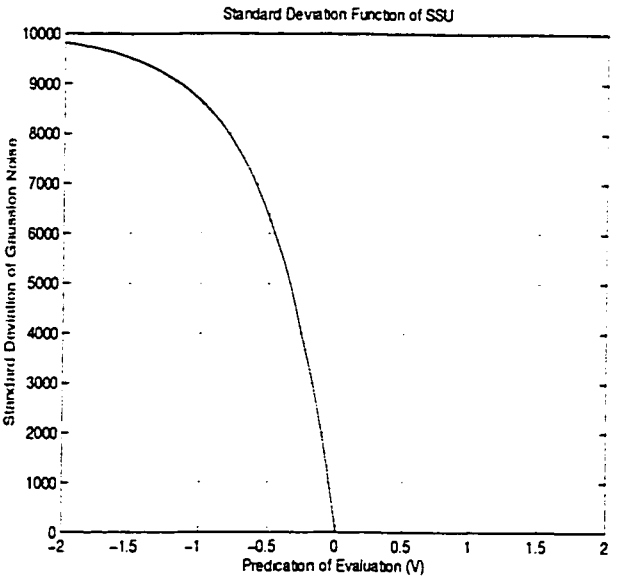


Figure 3.13: Standard deviation function of Stochastic Search Unit (SSU). See 3.74 for the expression. The parameters used are σ_{max}=10000, σ_r=0.5 .

time t:

$$\epsilon^t_{wk} = \sum_{l=1}^{t} \lambda^{t-l} \frac{\partial l n g_l}{\partial w_k} \tag{3.76}$$

ϵ^t_{wk} is computed recursively (3.43) as following:

$$\begin{aligned} \epsilon^{t+1}_{wk} &= \lambda_w \epsilon^t_{wk} + \frac{\partial l n g_{t+1}}{\partial w_k} \\ &= \lambda_w \epsilon^t_{wk} + \frac{\partial l n g_{t+1}}{\partial s} \frac{\partial s}{\partial w_k} \\ &= \lambda_w \epsilon^t_{wk} + \frac{f_{t+1} - s}{3 \cdot \sigma_{max}} \frac{\partial s}{\partial w_k} \end{aligned} \tag{3.77}$$

where λ_w is the TD decaying factor, $\frac{\partial s}{\partial w_k}$ is computed as eq.3.15, and $\frac{f_{t+1}-s}{3 \cdot \sigma_{max}}$ is a minor modification of the REINFORCE eq.(3.31) by using $(3 \cdot \sigma_{max})$ instead of (σ^2) in the denominator. The reason for the minor modification is if σ^2 is used, the weight change factor $\frac{f_{t+1}-s}{\sigma^2}$ could be quite large even when σ^2 is small. This is not desired since small σ means high evaluation value v (eq.3.74, fig.3.13) and good performance. The weight change should be small when the performance is already good. If $3 \cdot \sigma_{max}$ is used, the weight change factor $\frac{f_{t+1}-s}{3 \cdot \sigma_{max}}$ will be small when σ is small (because $f_{t+1} - s$ will be small while σ_{max} is constant) and weight change factor will be large when σ is large. This is desired. The weight change factor $\frac{f_{t+1}-s}{3 \cdot \sigma_{max}}$ can be called the *normalized noise*[Gul90] or *normalized perturbation*[HB2b] in the sense that it is gaussian noise $f_{t+1} - s$ normalized by its (maximum) standard de-

viation. Using constant 3 in the normalization denominator is due to the fact that the possibility of value $f_{t+1} - s$ lies within $[\pm 3 \cdot \sigma]$ is 95%.

Similarly for input fuzzification layer parameters a_{ik}, b_{ik}, c_{ik}, the weight update rules are:

$$\Delta a_{ik}^t = \tau_a (r^{t+1} + \gamma v^{t+1} - v^t) \epsilon_{aik}^t \tag{3.78}$$

$$\begin{aligned} \epsilon_{aik}^{t+1} &= \lambda_a \epsilon_{aik}^t + \frac{\partial ln g_{t+1}}{\partial a_{ik}} \\ &= \lambda_a \epsilon_{aik}^t + \frac{f_{t+1} - s}{3 \cdot \sigma_{max}} \frac{\partial s}{\partial a_{ik}} \end{aligned} \tag{3.79}$$

$$\Delta b_{ik}^t = \tau_b (r^{t+1} + \gamma v^{t+1} - v^t) \epsilon_{bik}^t \tag{3.80}$$

$$\begin{aligned} \epsilon_{bik}^{t+1} &= \lambda_b \epsilon_{bik}^t + \frac{\partial ln g_{t+1}}{\partial b_{ik}} \\ &= \lambda_b \epsilon_{aik}^t + \frac{f_{t+1} - s}{3 \cdot \sigma_{max}} \frac{\partial s}{\partial b_{ik}} \end{aligned} \tag{3.81}$$

$$\Delta c_{ik}^t = \tau_c (r^{t+1} + \gamma v^{t+1} - v^t) \epsilon_{cik}^t \tag{3.82}$$

$$\begin{aligned} \epsilon_{cik}^{t+1} &= \lambda_c \epsilon_{cik}^t + \frac{\partial ln g_{t+1}}{\partial c_{ik}} \\ &= \lambda_c \epsilon_{cik}^t + \frac{f_{t+1} - s}{3 \cdot \sigma_{max}} \frac{\partial s}{\partial c_{ik}} \end{aligned} \tag{3.83}$$

where $\frac{\partial s}{\partial a_{ik}}, \frac{\partial s}{\partial b_{ik}}, \frac{\partial s}{\partial c_{ik}}$ are computed as eq. 3.22, 3.21, 3.20.

From the above detailed learning rules for AFN parameters, we can further understand that reinforcement learning is a kind of stochastic optimization method which searches for the optimum by following gradient in stochastic sense. It can be used in a non-associative reinforcement learning task which directly adjusts the control parameters. More important is that it can be used with the Back Propagation algorithm in an associative reinforcement learning task. The REINFORCE error term ($\frac{f_{t+1} - s}{3 \cdot \sigma_{max}}$) is back-propagated for internal net parameter modification using a known gradient. This is not to be confused with the direct optimization search in internal net parameter space. The former takes the advantage of knowing net structure and propagating the optimization of control signal back to internal net (structural credit assignment) using gradient and, hence, is more efficient than direct searching in internal net parameter space without using the knowledge of known net structure. Usually, internal net parameter space is much larger than output control space. For

example, in our FNS swing model, only two controls are to be optimized, while the AFN may have hundreds of net parameters (a,b,c w parameters of membership function for each rule and each input). Therefore, a direct search in net parameter space will be much slower than the above combination of optimization in output control space using REINFORCE algorithm and back propagation in internal net. Direct optimization in internal net parameter space using standard optimization algorithms (such as simplex, Gaussian-Newton provided by MATLAB Optimization Toolbox) have been used. The result is exactly as discussed above.

From this analysis of combination of reinforcement learning and backpropagation-type gradient algorithm, it seems possible to use other non-linear optimization algorithms in control output space with Back Propagation algorithm in internal net space. The advantage of stochastic optimization like Reinforcement Learning is, that, it is more likely to escape from local minimums (due to its randomness) than the deterministic optimizations (especially deterministic gradient algorithms). Therefore, stochastic optimization is more likely to find a global minimum, with the possible cost of slow search speed. However, in the AFN situation, our expert knowledge may be good enough to pre-structure the AFN near the global minimum. Thus deterministic gradient or other heuristic search algorithms could be sufficient to converge to minimum (also global minimum), with possible advantage of fast convergence. This may be worth further study. Some basic requirements are: faster convergence than stochastic gradient following, computationally inexpensive(suitable for real time on-line implementation), incremental modification so that it can be combined with back propagation algorithm, ability to cope with non-differential case because the objective function J with respect to control output signal may be very complex, and is very likely non-continuous and non-differentiable. This may exclude some deterministic gradient-based algorithms like BFGS, DFP, Steepest Gradient, and Least Square, etc. Although the AFN is a differentiable net ($\frac{\partial y}{\partial P}$ exists thus BP gradient is applicable in net space), we do not know if $\frac{\partial r}{\partial y}$ exists. If $\frac{\partial r}{\partial y}$ does not exist, using deterministic gradient-based algorithms may cause a singular matrix problem. However, stochastic gradient-like algorithms such as reinforcement learning algorithms do not have this singular problem.

Chapter 4

Results

4.1 Supervised Learning Controller for FNS Swing

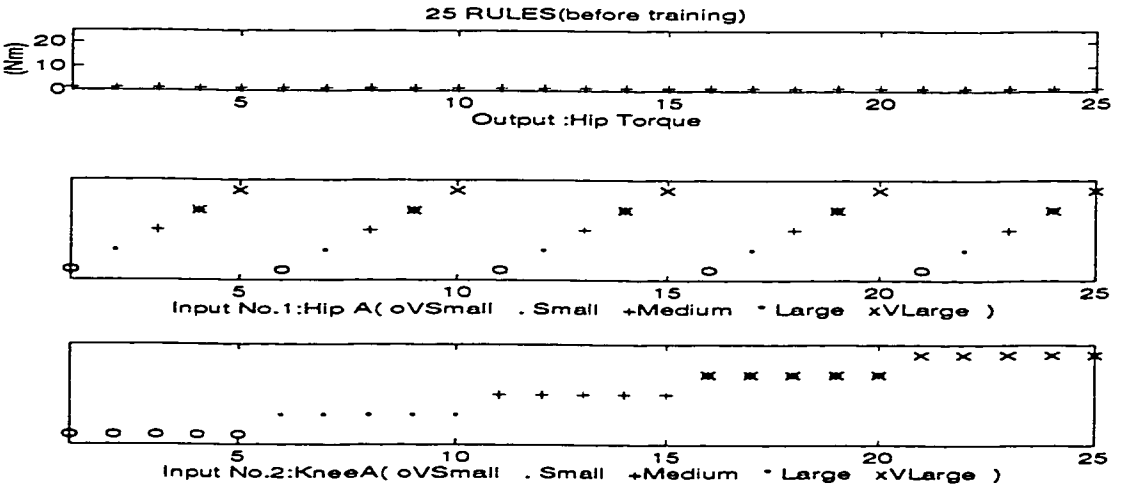
4.1.1 Simulation Parameters

Two separated supervised learning AFN controllers are used, one for hip torque control, another for quadriceps stimulation control. Previously optimized open-loop controller (see Fig.3.3) serves as teacher to provide the desired control signals.

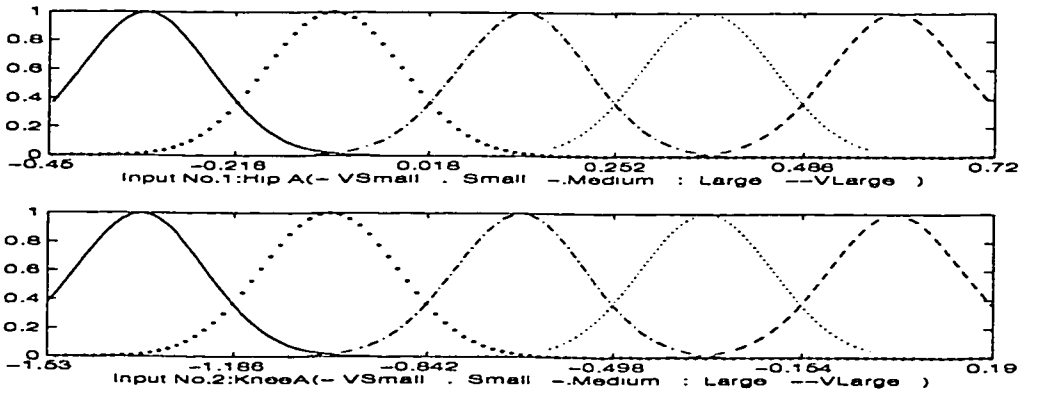
The controller inputs are hip angle and knee angle. The input range is [-0.45 rad, 0.72 rad] for hip angle, and [-1.53rad, 0.19rad] for knee angle. Although there are more variables, such as hip/knee angular velocity, available as controller inputs, it was found that using more inputs did not always help. More inputs means more dimensionality, and the "curse of dimensionality" will cause exponentially increasing rule numbers. For a given number of rules, low number of inputs will allow fine partition of state space. In [WA94], we used four input variables (hip/knee angle and angular velocity). But the result was not good, only foot-clearance was achieved in reinforcement learning after 200 training trials. Using only two inputs resulted in better results. The reason is explained as follows: The more inputs are used, the more information is available for the control purpose and thus the better the control performance could be. On the other hand, the more inputs are used, the larger the search space is. Therefore, there is a tradeoff between the search space and the control performance. If the number of inputs is too small, the controller will not have sufficient sensor information to control the system. But if the number of inputs is too large, the controller will not be able to find the optimal control strategy in the large search space. In the model studied here, it was found that using two inputs resulted in the best results.

Five fuzzy variables (VSmall, Small, Medium, Large, VLarge) are assigned to every input, resulting in a total of 25 (5x5) fuzzy control rules, and thus 25 output fuzzy singleton (one output fuzzy singleton per rule) for each controller. Output range is [0Nm, 24Nm] for the hip torque controller, and [0u-sec, 240u-sec] for the quadriceps stimulation controller. Output of controllers (output fuzzy singletons) are initialized to near zero (1.1Nm for hip torque, and 21 u-sec for quadriceps stimulation).

Membership functions of five input fuzzy variables, 25 output fuzzy singletons, and 25 control rules are shown in figure 4.1 and 4.2. The 25 fuzzy rules 'fuzzily' partition the two-dimensional input space (figure 4.3).



(A) Output fuzzy singletons, fuzzy control rules in graphic form.



(B) Membership functions of input fuzzy variables.

Rule No1: If hip-angle is VSmall and knee-angle is VSmall Then Hip Torque is 1.1 Nm

.....

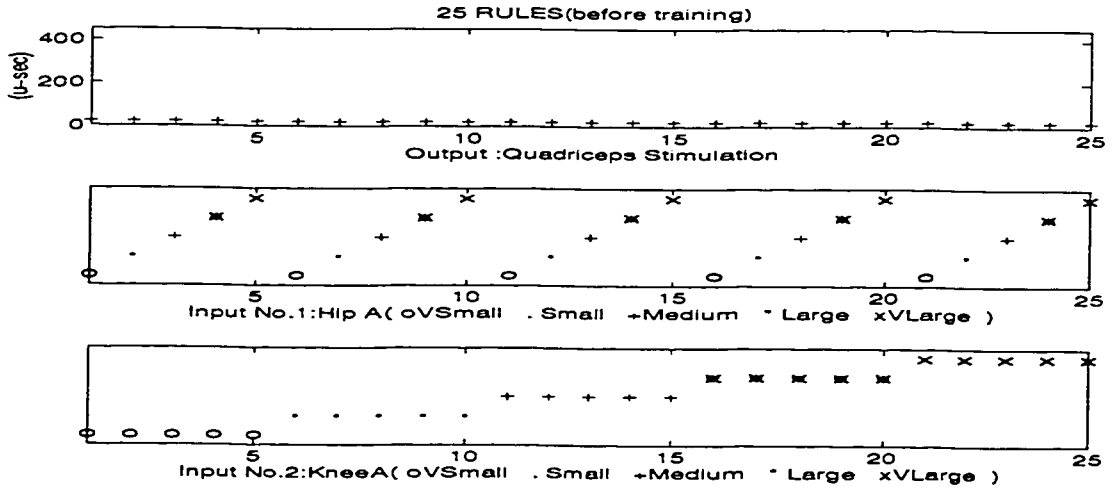
Rule No.15: If hip-angle is VLarge and knee-angle is Medium Then Hip Torque is 1.1 Nm

.....

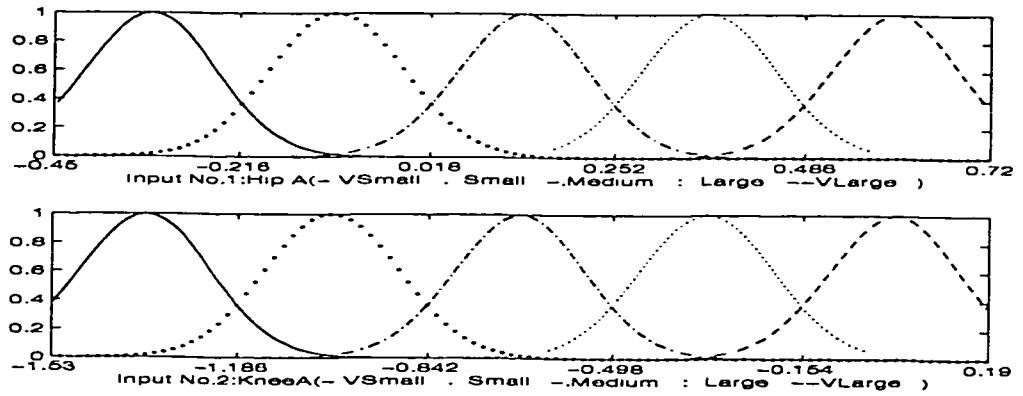
Rule No.25: If hip-angle is VLarge and knee-angle is VLarge Then Hip Torque is 1.1 Nm

(C) Fuzzy control rules in text form (other rules which are not shown here have similar rule form).

Figure 4.1: Fuzzy rule base for hip torque controller(before training).



(A) Output fuzzy singletons, fuzzy control rules in graphic form.



(B) Membership functions of input fuzzy variables.

Rule No1: If hip-angle is VSmall and knee-angle is VSmall Then Quadriceps Stimulation is 21 u-sec

.....

Rule No.15: If hip-angle is VLarge and knee-angle is Medium Then Quadriceps Stimulation is 21 u-sec

.....

Rule No.25: If hip-angle is VLarge and knee-angle is VLarge Then Quadriceps Stimulation is 21 u-sec

(C) Fuzzy control rules in text form (other rules which are not shown here have similar rule form).

Figure 4.2: Fuzzy rule base for quadriceps stimulation controller(before training).

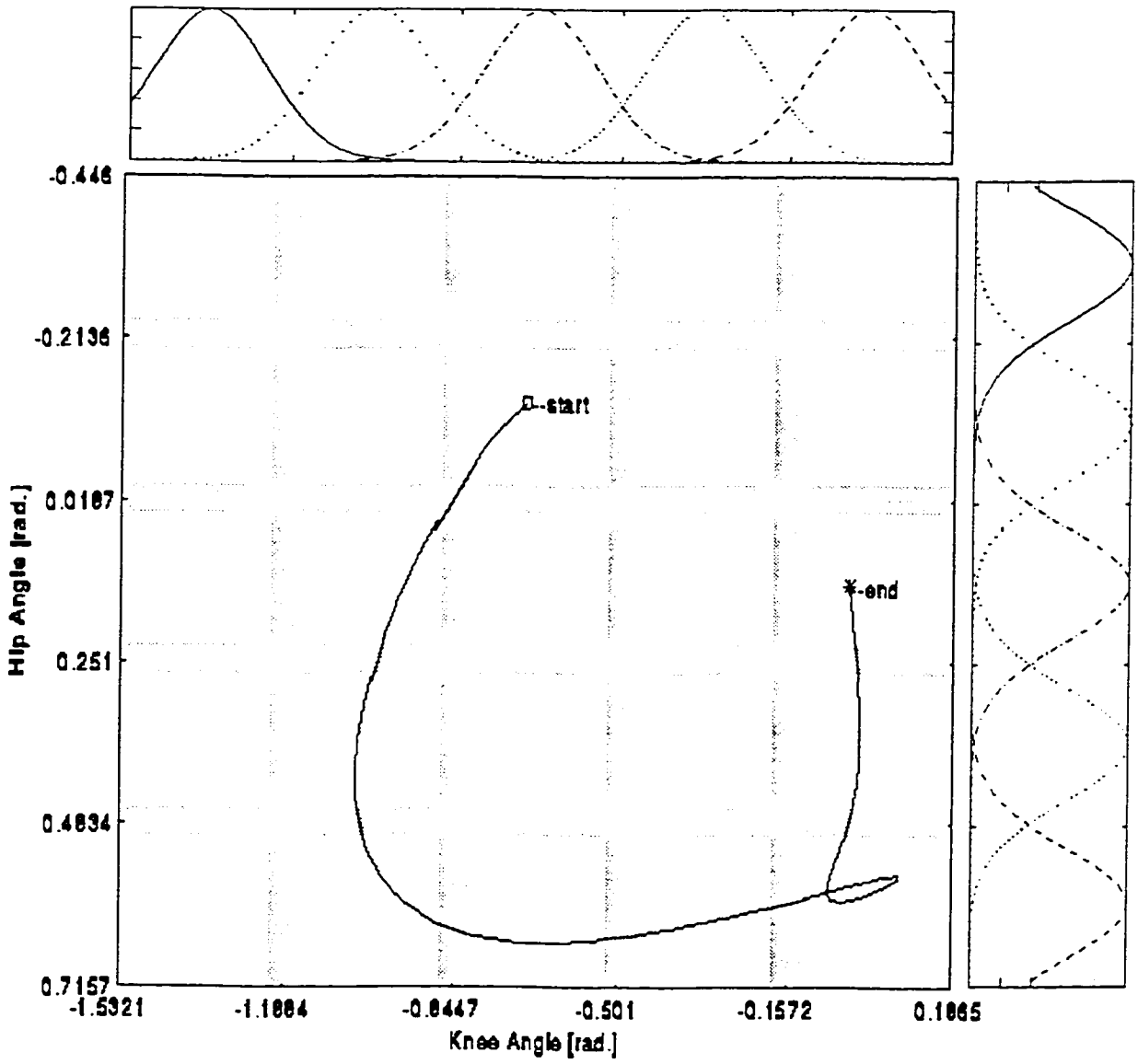


Figure 4.3: Fuzzy partitioning of two-dimensional input space. The membership functions of five input fuzzy variables are also shown aside. The grey area in the figure indicates the fuzzy area covered by more than one membership functions. The trajectory controlled by the optimal open-loop controller is also shown with start point(o) and end point(*).

4.1.2 Results

After 15 trials, the outputs of supervised learning AFN controller were almost the same as the optimized open-loop controller (figure 4.4). This learning controller can successfully control the leg to finish one swing phase (figure 4.5).

From the learning curves of the supervised learning controllers (figure 4.6), it learned very rapidly in the first 3 trials, then just further fine-tuned.

The fuzzy control rule bases after supervised learning are shown in figure 4.7 and 4.8. The input fuzzy membership function did not change significantly, as the learning rates ($\tau_{a/b/c}$) for theses parameters (a_{ij} b_{ij} c_{ij}) are quite small (0.01). But the output fuzzy singletons changed very significantly.

The three dimensional control surfaces synthesized from fuzzy control rules are shown in figure 4.9. The line trajectories on the surfaces are actual trajectories under control of the learning controller.

Figure 4.10 shows the firing pattern of control rules along the trajectory. From beginning to end of swing phase, fuzzy control rules fire sequentially as the trajectory visits the different portions of state space.

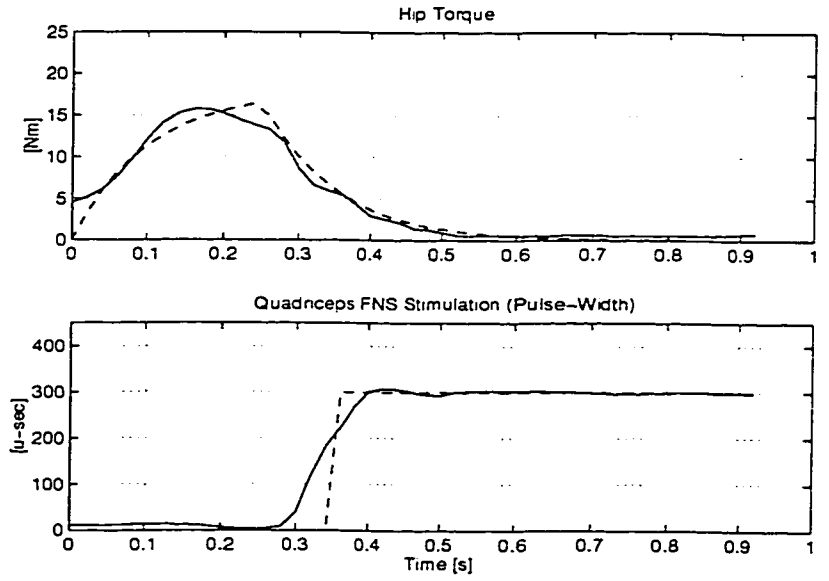


Figure 4.4: Controller outputs after supervised learning (The teacher signals are also shown with dashed lines).

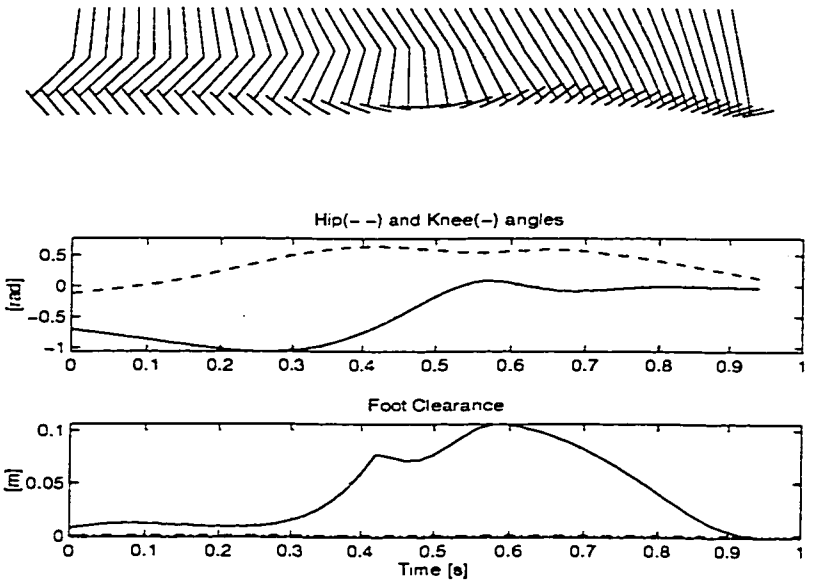


Figure 4.5: Trajectory of swing leg controlled by supervised learning controller.

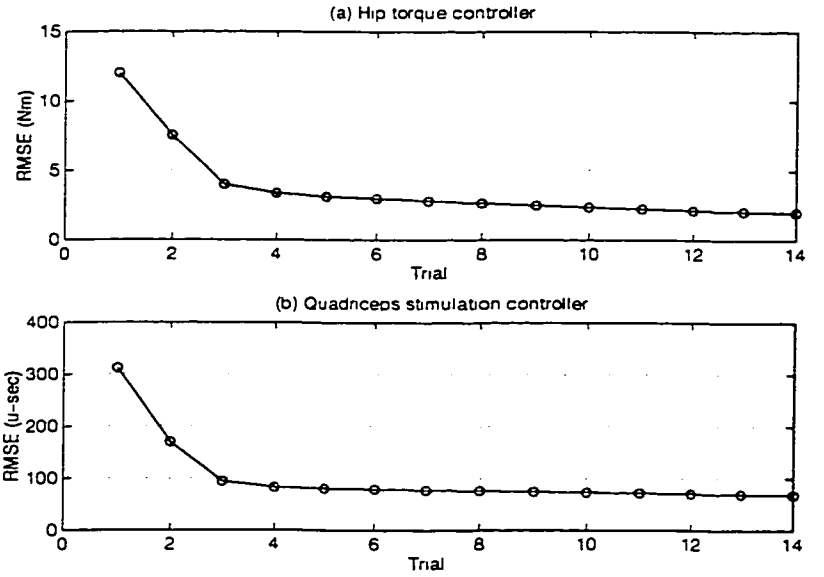
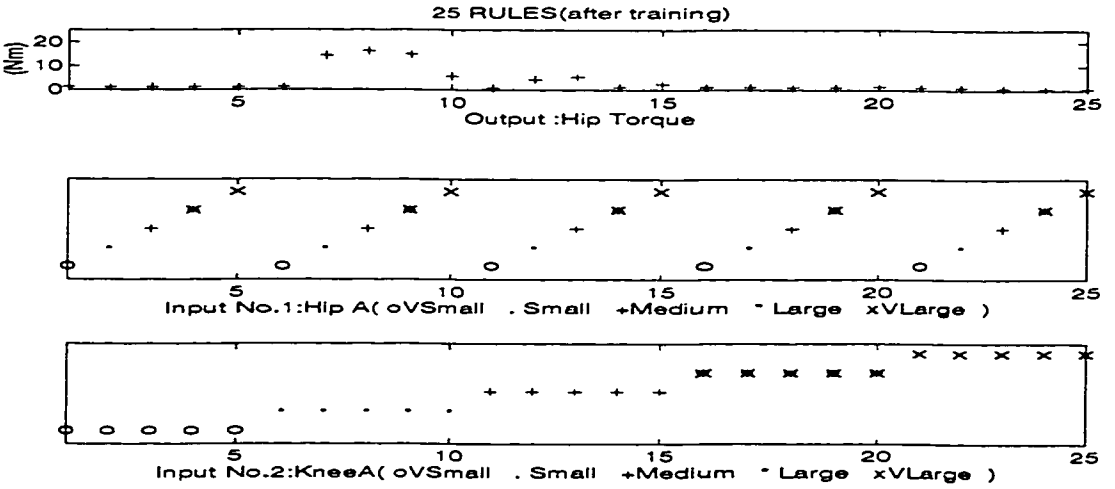
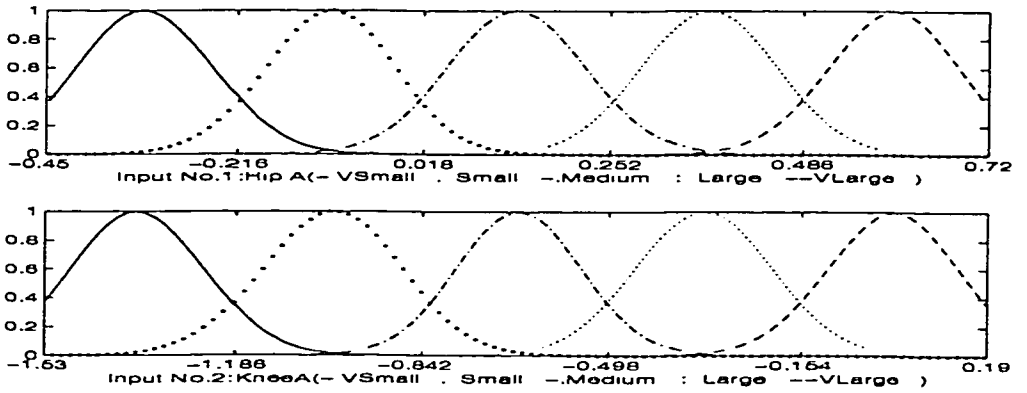


Figure 4.6: Learning curve of supervised learning controller.



(A) Output fuzzy singletons, fuzzy control rules in graphic form.



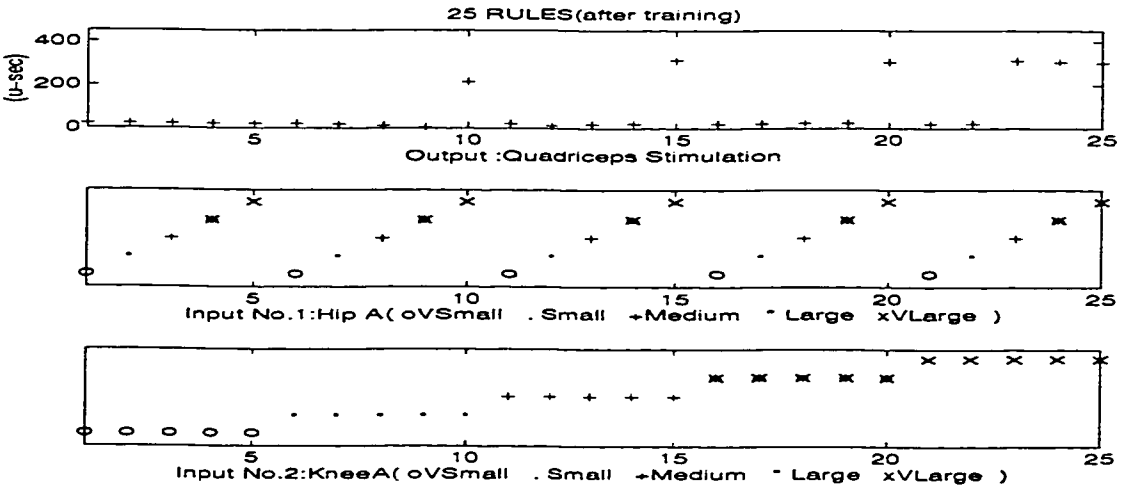
(B) Membership functions of input fuzzy variables.

Rule No.7: If HipAngle is Small and KneeAngle is Small Then Hip Torque is 14 Nm
Rule No.8: If HipAngle is Medium and KneeAngle is Small Then Hip Torque is 16 Nm
Rule No.9: If HipAngle is Large and KneeAngle is Small Then Hip Torque is 14 Nm
Rule No.10: If HipAngle is VLarge and KneeAngle is Small Then Hip Torque is 5 Nm
Rule No.12: If HipAngle is Small and KneeAngle is Medium Then Hip Torque is 4 Nm
Rule No.13: If HipAngle is Medium and KneeAngle is Medium Then Hip Torque is 5 Nm

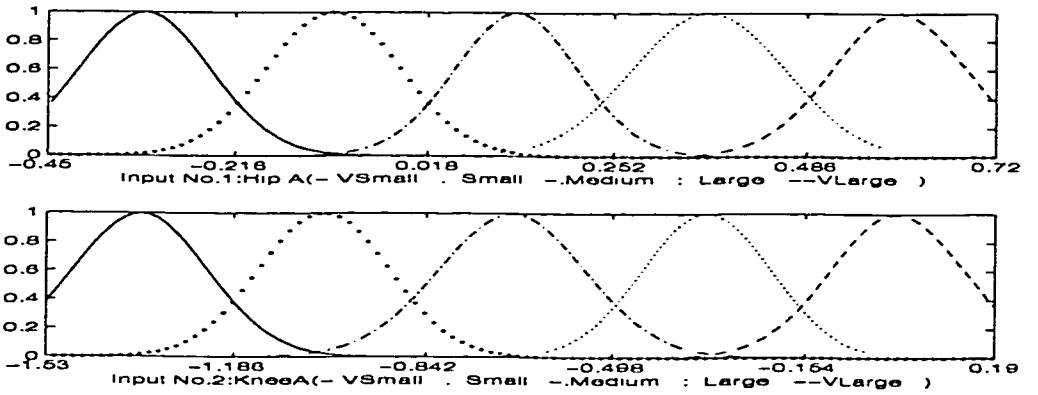
(C) Fuzzy control rules in text form

(other rules which are not shown here do not changed much in the learning and remain similar to their initialized form in Fig.4.1C).

Figure 4.7: Fuzzy rule base for hip torque controller(after supervised learning).



(A) Output fuzzy singletons, fuzzy control rules in graphic form.



(B) Membership functions of input fuzzy variables.

Rule No.10: If HipAngle is VLarge and KneeAngle is Small Then Quad Stim. is 210 u-sec.
Rule No.15: If HipAngle is VLarge and KneeAngle is Medium Then Quad Stim. is 320 u-sec.
Rule No.20: If HipAngle is VLarge and KneeAngle is Large Then Quad Stim. is 320 u-sec.
Rule No.23: If HipAngle is Medium and KneeAngle is VLarge Then Quad Stim. is 320 u-sec.
Rule No.24: If HipAngle is Large and KneeAngle is VLarge Then Quad Stim. is 310 u-sec.
Rule No.15: If HipAngle is VLarge and KneeAngle is VLarge Then Quad Stim. is 310 u-sec.

(C) Fuzzy control rules in text form

(other rules which are not shown here do not changed much in the learning and remain similar to their initialized form in Fig.4.2C).

Figure 4.8: Fuzzy rule base for quadriceps stimulation controller(after supervised learning).

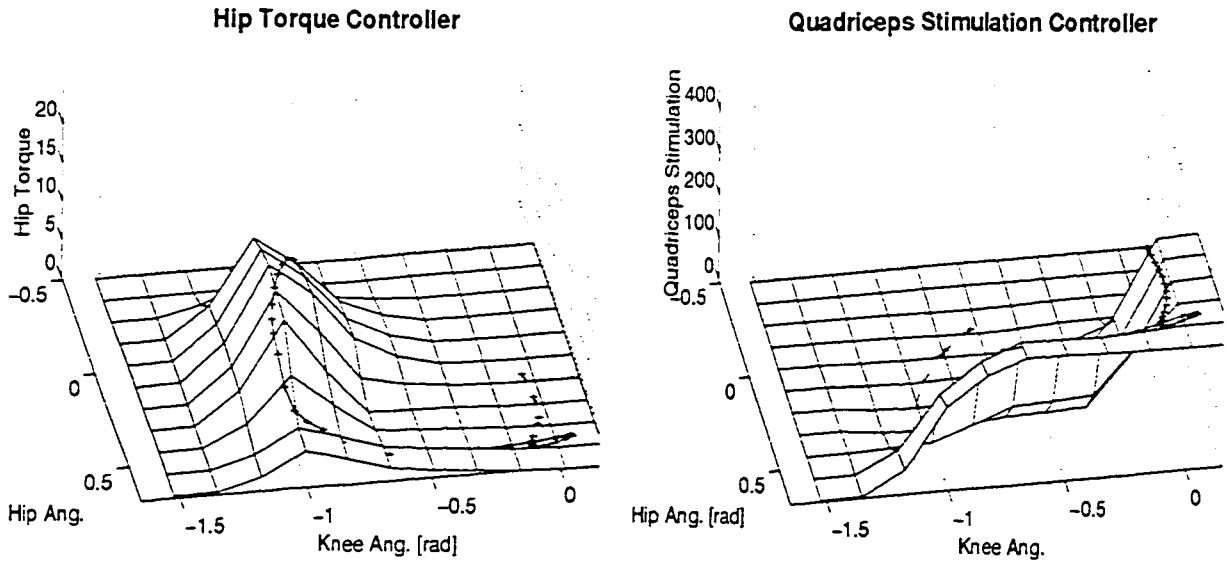


Figure 4.9: Control surface of fuzzy logic controller after supervised learning. The actual trajectory is also shown on the control surface. The '+' symbols indicate the sampled points.

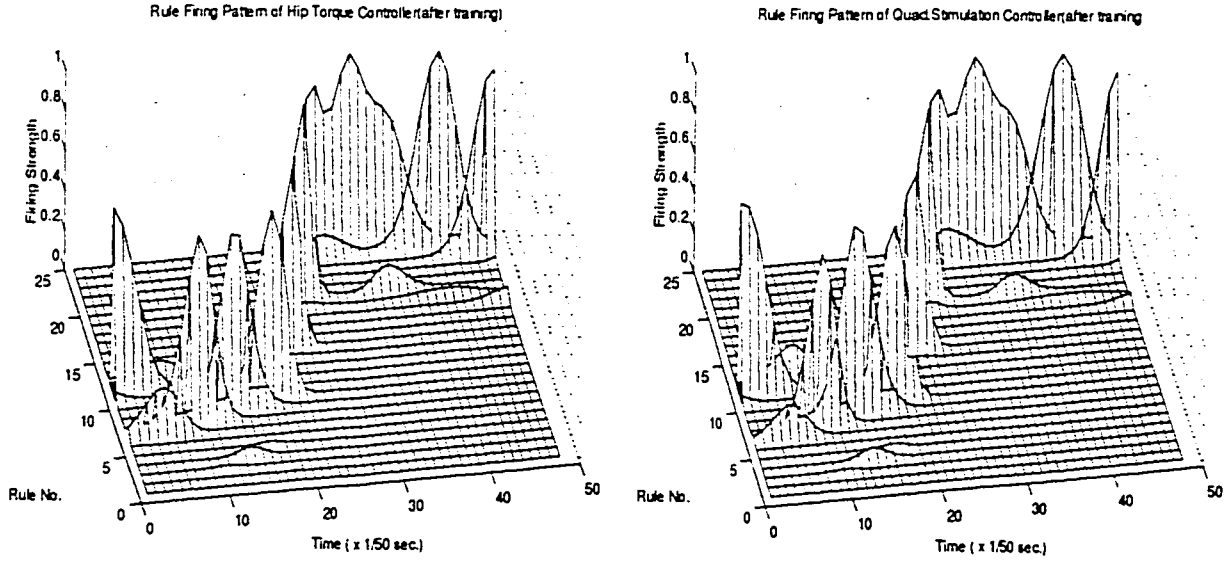


Figure 4.10: Firing pattern of fuzzy logic controllers.

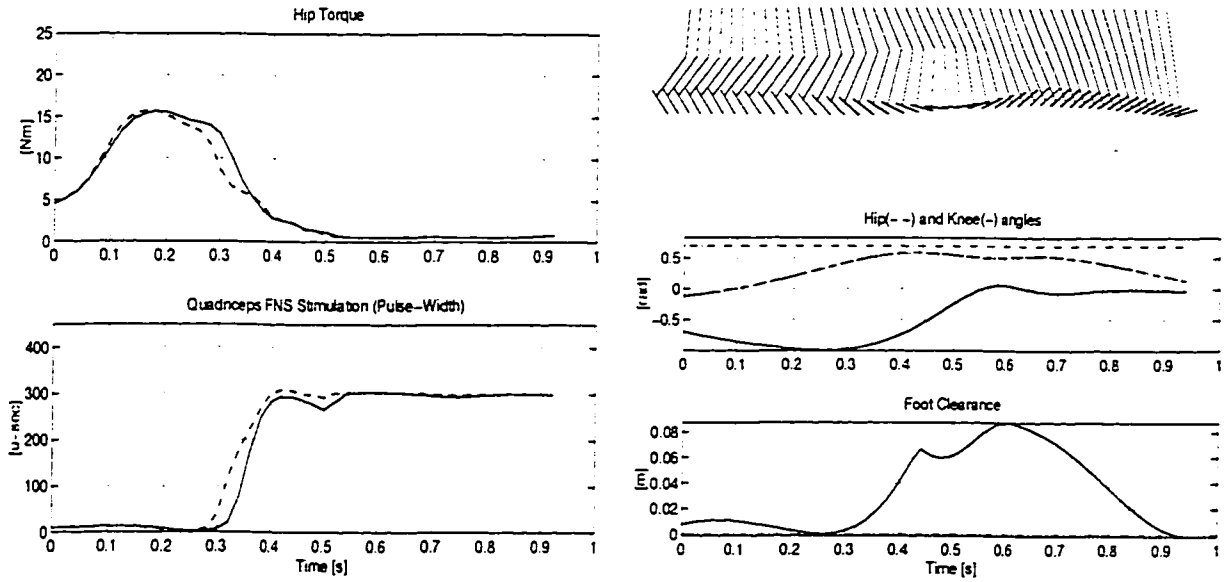
If the supervised learning controller can only mimic the optimized open-loop controller, then why use it? The reason is that the supervised learning controller is a closed-loop controller. Hopefully, it will be able to generalize the control rules learned from normal trajectories to other trajectories that open-loop controller does not teach it. Thus, it can handle the situation when there are parameter variations, while the open-loop controller is not able to cope, as described in the next section.

4.1.3 Adaptability to Parameter Variations

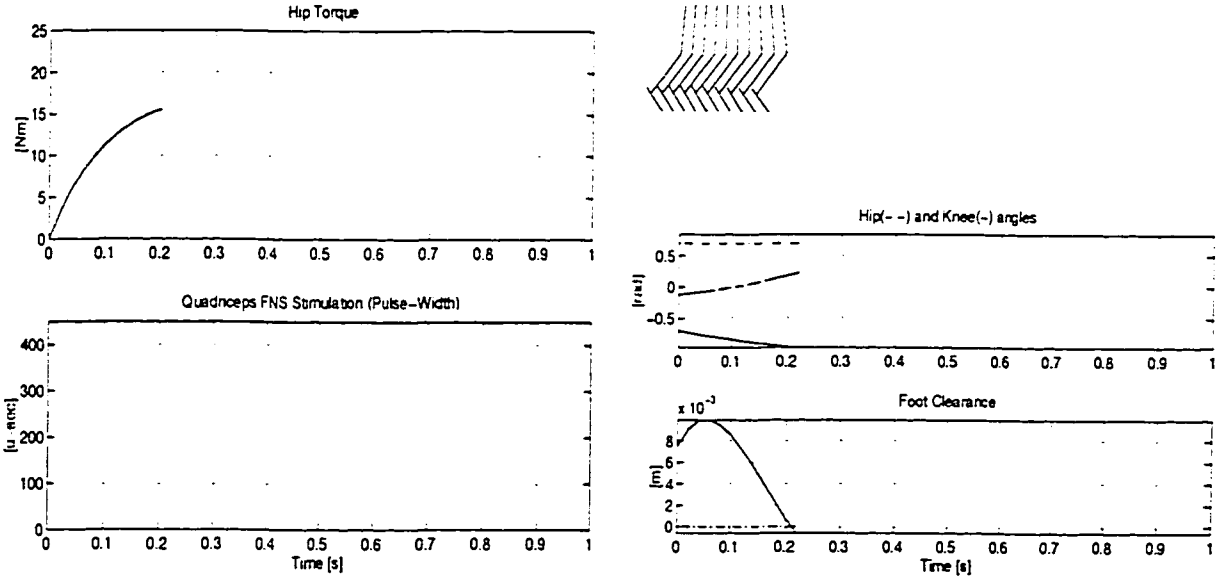
(1) Muscle Fatigue

To simulate the muscle fatigue which is a fundamental problem in FNS control system. we change hip torque gain from 1.00 to 0.86 (a change of 14quadriceps muscle gain remains 1.00. The supervised learning closed-loop controller used more hip torque to avoid toe collision in the initial swing phase and successfully finished swing phase (figure 4.11A).

In contrast, the open-loop controller failed (figure 4.11B) by using the same control as in the normal gain situation. The toe hits the ground due to insufficient hip torque caused by low gain.



(A) Closed-loop supervised learning controller succeeded (The control signals for normal gain are also shown with dashed lines).



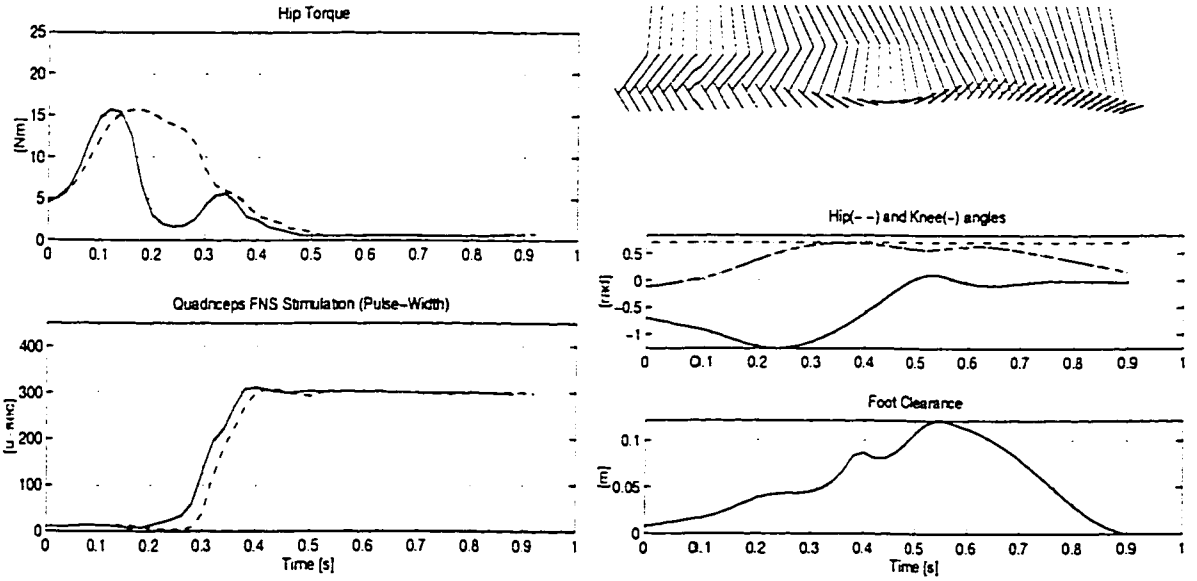
(B) Open-loop controller failed (The foot clearance=0 is indicated by dash-dot line).

Figure 4.11: Adaptability of supervised learning controller to low gain (muscle fatigue).

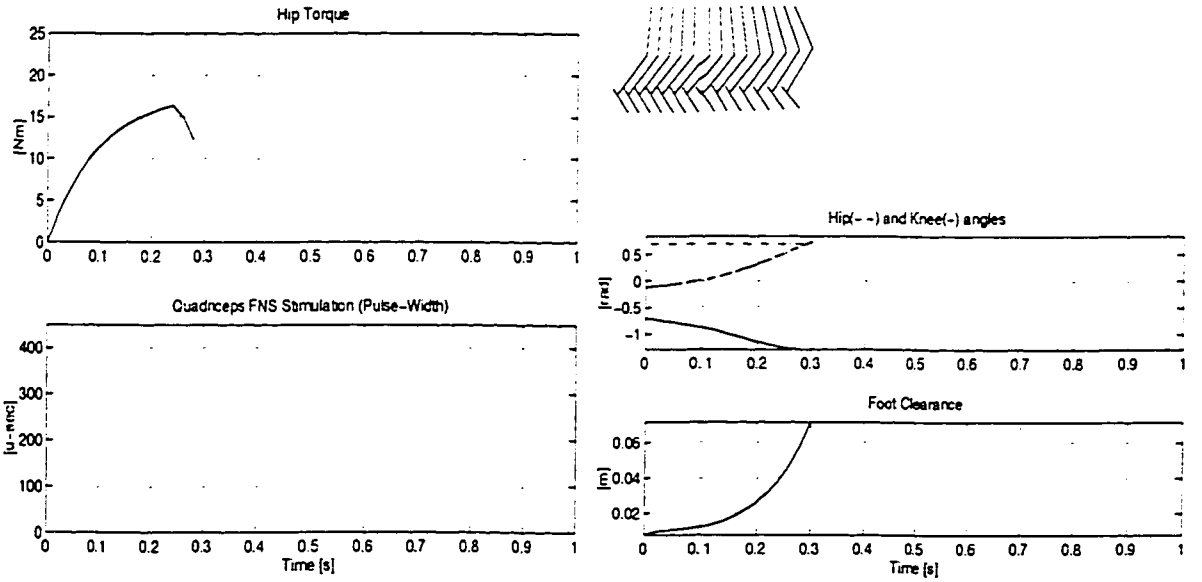
(2) Muscle Potentiation

To simulate the muscle potentiation. hip torque gain was changed to 1.70. and quadriceps muscle gain to 1.70. The supervised learning closed-loop controller used less hip torque to avoid exceeding maximum hip angle (figure 4.12(A)). The higher quadriceps muscle gain did not cause problems because of the saturation of the muscle recruitment curve.

Open-loop controller failed (figure 4.12(B)) by using the same control as in the normal situation. The hip angle exceeded the maximum angle due to high hip torque caused by high gain.



(A)Closed-loop supervised learning controller succeeded (The control signals for normal gain are also shown with dashed lines).



(B)Open-loop controller failed (The hip angle maximum (0.7) is indicated by dash-dot line).

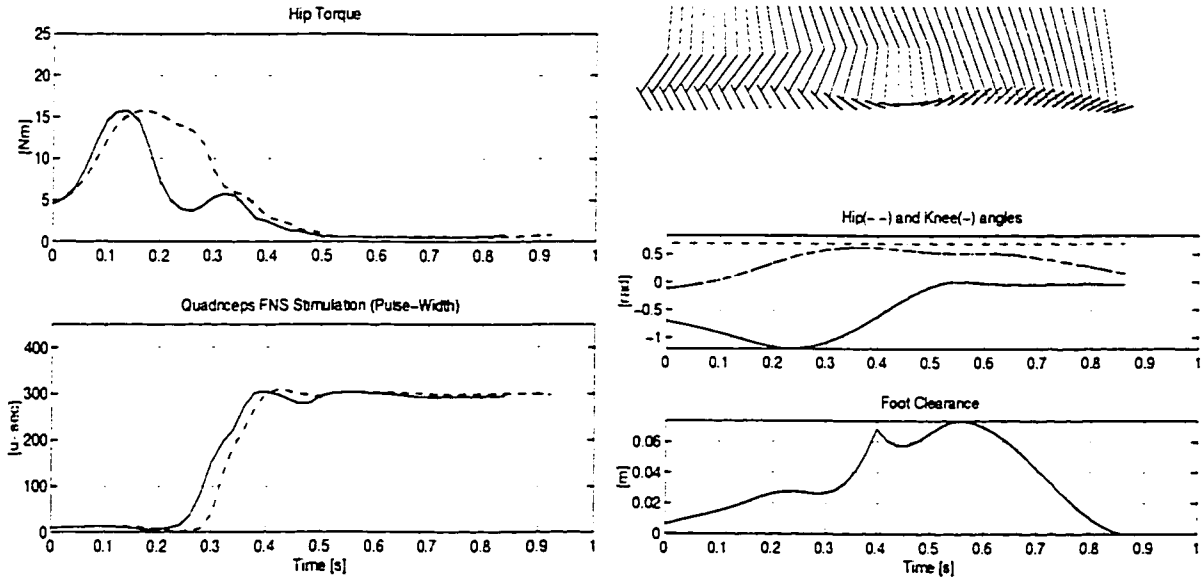
Figure 4.12: Adaptability of supervised learning controller to high gain (muscle potentiation).

(3) Light Body Mass

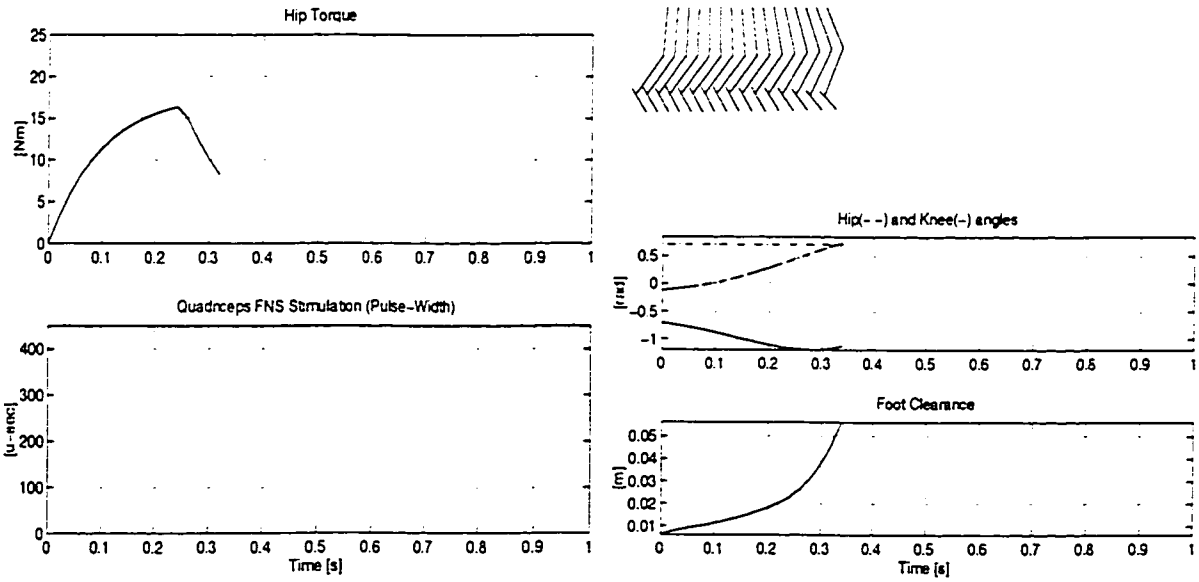
The body mass was changed to 45kg. and body height to 1.50m.

The supervised learning closed-loop controller used less hip torque to avoid exceeding maximum hip angle (figure 4.13(A)).

The open-loop controller failed (figure 4.13(B)). The hip angle exceeded the maximum hip angle (0.7 rad) due to smaller leg mass.



(A) Closed-loop supervised learning controller succeeded (The control signals for normal body mass is also shown with dashed lines).



(B) Open-loop controller failed (the hip maximum is indicated by dash-dot line).

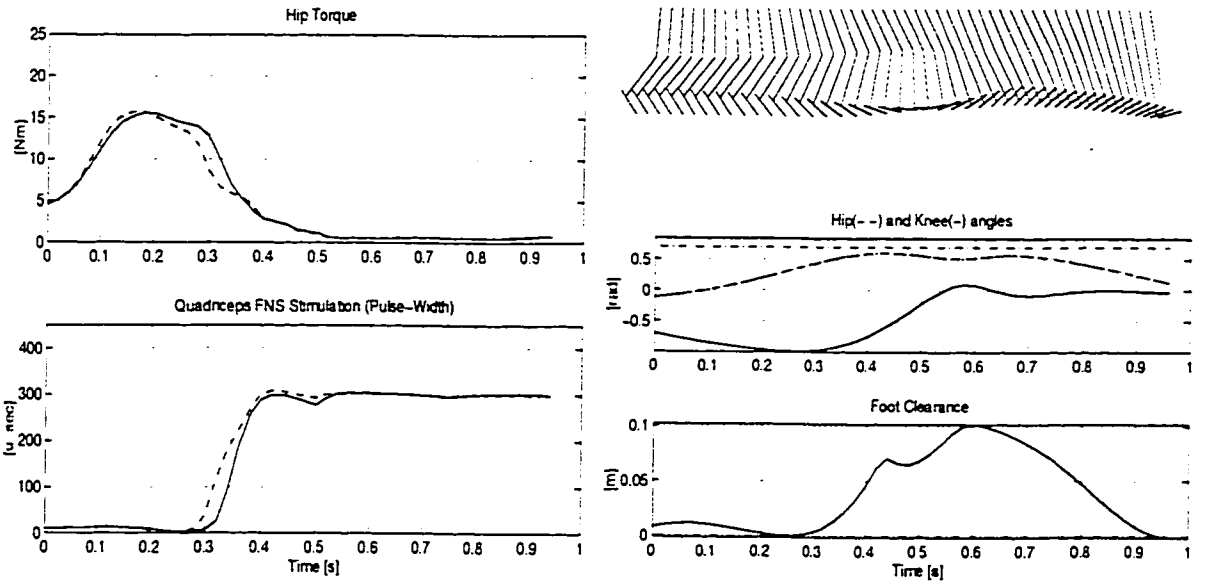
Figure 4.13: Adaptability of supervised learning controller to light body mass.

(4) Heavy Body Mass

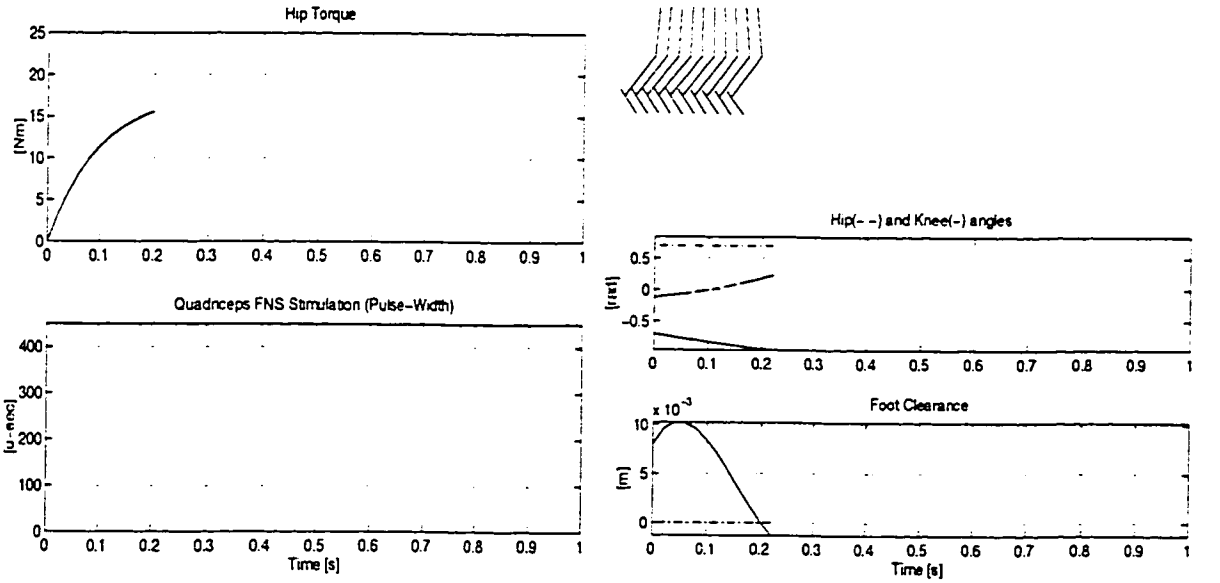
Body mass was changed to 60 kg. and body height to 1.70m.

The supervised learning controller used more hip torque to avoid the toe collision in the initial swing phase (figure 4.14(A)).

The open-loop controller failed by using the same control as in the normal situation (figure 4.14(B)). The toe hit the ground due to insufficient hip torque to flex the larger leg mass.



(A) Closed-loop supervised learning controller succeeded (The control signals for normal body mass is also shown with dashed lines).



(B) Open-loop controller failed (the foot clearance=0 is indicated by dash-dot line).

Figure 4.14: Adaptability of supervised learning controller to heavy body mass.

Conclusion: The supervised learning controller can rapidly learn the optimal control rules from the optimized open-loop controller in approximately 10 trials (Fig.4.6). Furthermore, by using the closed-loop sensor feedback, it can generalize learned control rules to the new models with different gain and body mass/height (e.g. Fig.4.11). Therefore, the closed-loop supervised learning controller is an improvement to the optimized open-loop controller.

However, when the situation is beyond the ability of closed-loop supervised learning controller, it can no longer handle it. For example, if gain change is too large, or new model's mass is very different from the original one, then the supervised learning controller will also fail (see section 4.3). In this case, it needs a new teacher to teach it new control rules. If there is no new teacher, this supervised learning controller will not work. In the next section, the reinforcement learning controller without teacher is able to handle this situation.

4.2 Reinforcement Learning Controller for FNS Swing

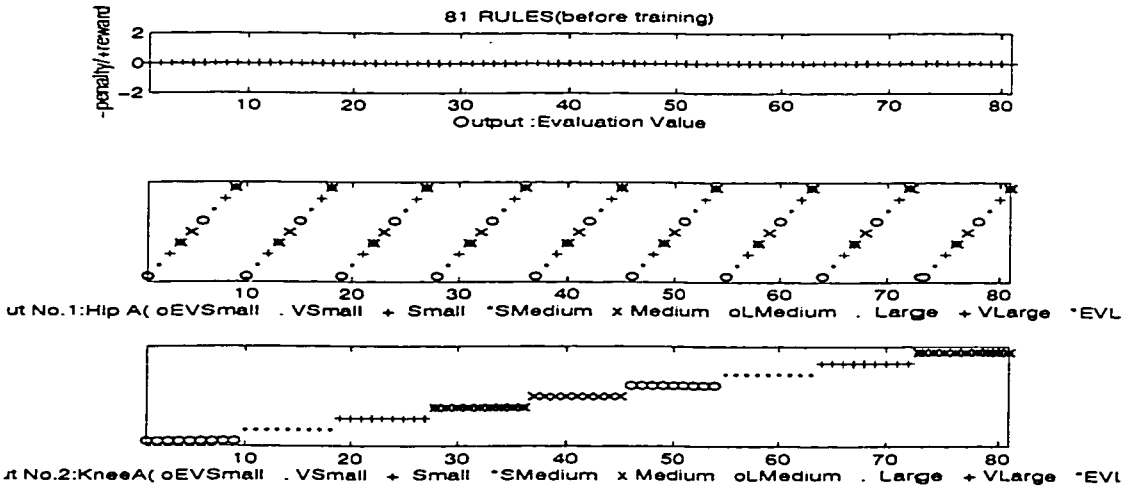
4.2.1 Simulation Parameters

Parameters for the hip torque controller and quadriceps stimulation controller are the same as in supervised learning, i.e. two inputs are hip angle and knee angle, five fuzzy variables for each input, and a total of 25 rules and output fuzzy singletons for each controller (figure 4.1 and 4.2). All controller outputs are initialized to near zero (Fig.4.2).

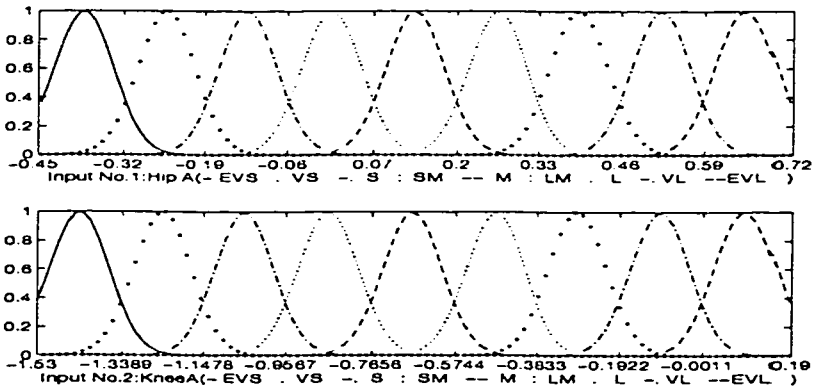
The TD decaying factors are λ=0.5 for hip torque controller, and a larger λ=0.9 for quadriceps stimulation controller, considering long delay of muscle response, and λ=0.5 for evaluation net.

The fuzzy rule base (membership functions of nine input fuzzy variables, 81 output fuzzy singletons, and fuzzy rules in graphic form) for the Evaluation Net is shown in figure 4.15. The outputs are initialized to zero. The Evaluation Net (EN) has 81 rules, i.e. 9 fuzzy variables for each input. Assigning more rules for the EN is due to the consideration that an accurate prediction of reinforcement is crucial for judging the right actions.

Figure 4.16 shows the fuzzy partitioning of input space by the evaluation net. The area bounded by line HA=0.7 and line FC=0 is the safe area (foot clearance larger than zero, and hip angle smaller than 0.7 rad), while any trajectory out of this area means failure (either toe collision, or hip angle exceeds maximum value, or knee angle is larger than 0.1 rad at the heel contact). The dark bar indicates the goal area (knee angle=[-.10rad 0.1rad]) in this two dimensional state space. The controller must learn to control the leg to 'travel', beginning from start point, through the state space without collisions (toe collision or maximum hip angle violation), and finally reach the goal area. This control problem is quite similar to the control of mobile robots for collision-free navigation or the control of a robot manipulator for collision-free path planning.



(A) Output fuzzy singletons and fuzzy rules.



(A) Membership functions of input fuzzy variables.

Figure 4.15: Fuzzy rule base of evaluation net (before reinforcement learning).

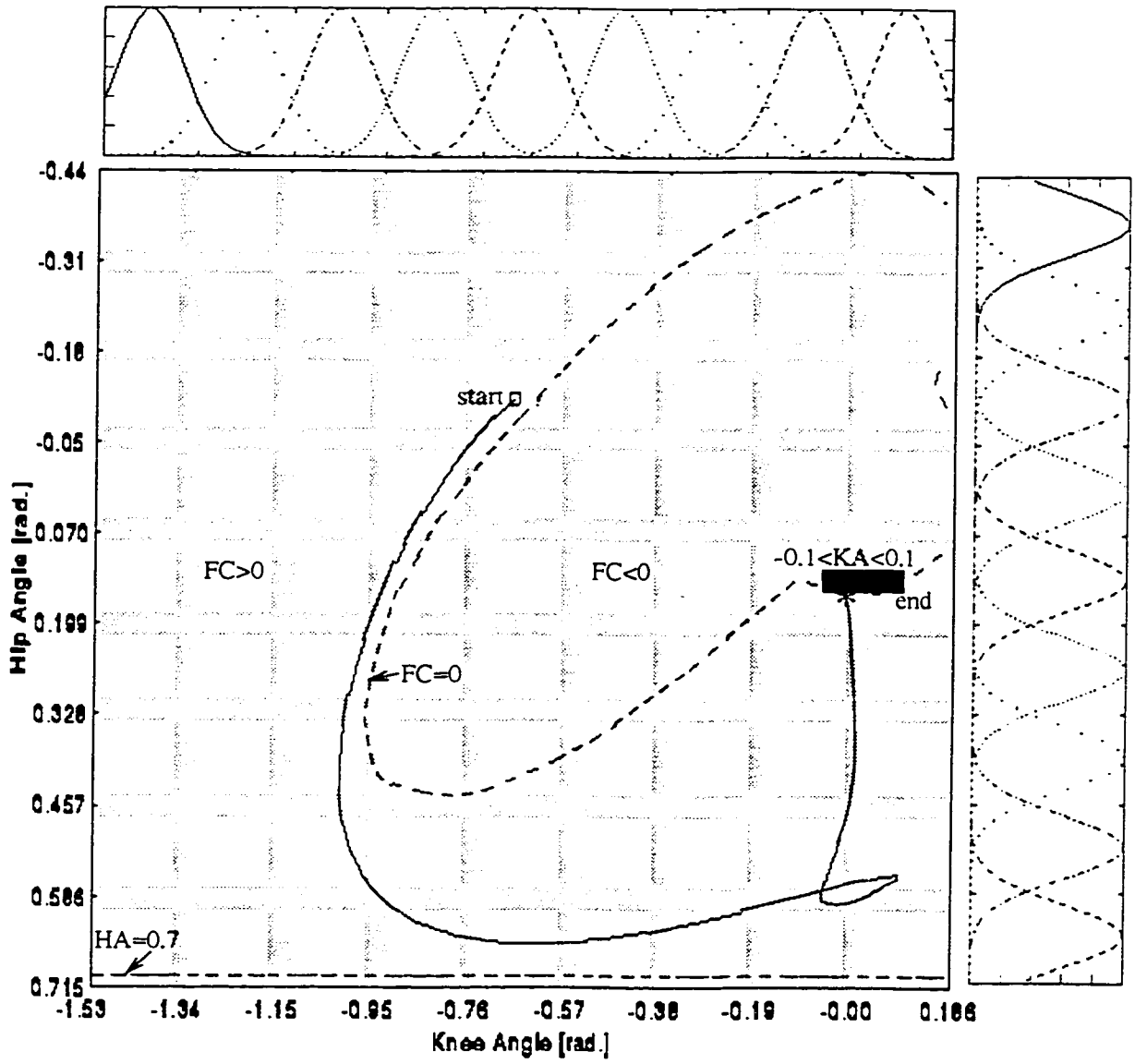


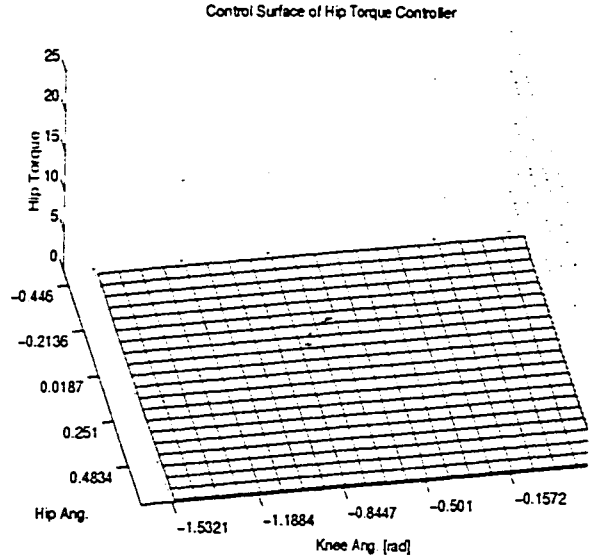
Figure 4.16: Fuzzy partitioning of input space by evaluation net. Nine fuzzy input variables are also shown aside. Trajectory controlled by optimal open-loop controller is shown with start point (o) and end point (*). Contour lines indicate foot-clearance=0 (FC=0) and hip angle maximum (HA=0.7 rad) are shown with dashed lines. The dark bar at the end of swing phase indicates the acceptable knee angle (-0.1rad< KA <0.1rad) at the end of swing phase (heel contact).

4.2.2 Results

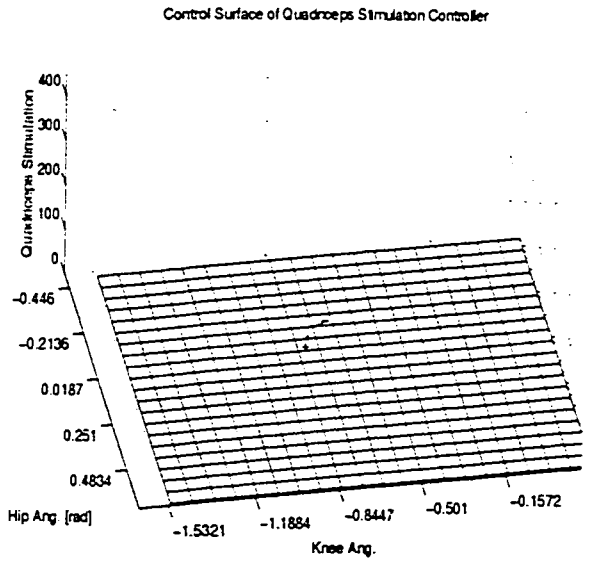
This section will show how the reinforcement learning controller learns from reinforcement feedback to avoid penalty and seek reward. A typical reinforcement learning process is shown in figures 4.17 to 4.21.

Stage 0: Initial state

Since both hip torque and quadriceps stimulation controllers are initialized to near zero (Fig.4.17A. Fig.4.17B), there is no sufficient hip torque to flex the hip and knee (figure.4.17(C)). After a few steps, the toe will hit the ground in the initial swing phase (figure.4.17(D)). The Evaluation Net is initialized to zero, so there is no prediction of reinforcement (figure.4.17(E)), and thus no stochastic search (figure.4.17(F)). In figures 4.17(A)(B)(E)(F), the trajectories are also shown on the 3D surfaces, with '+' symbols indicate the sampled points.

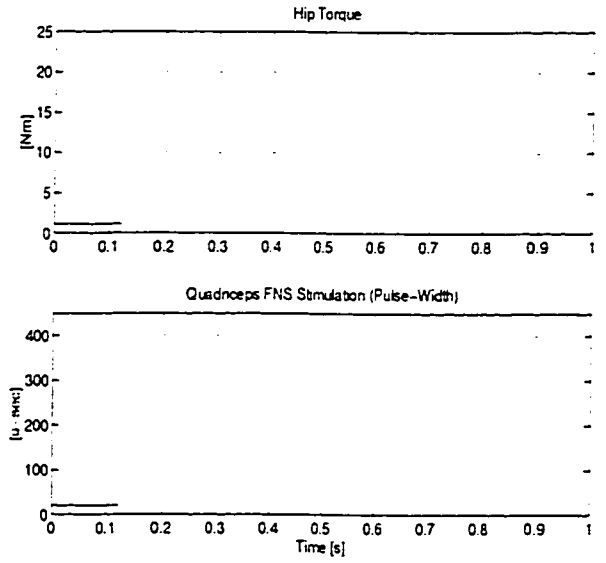


(A) 3D surface of hip torque controller

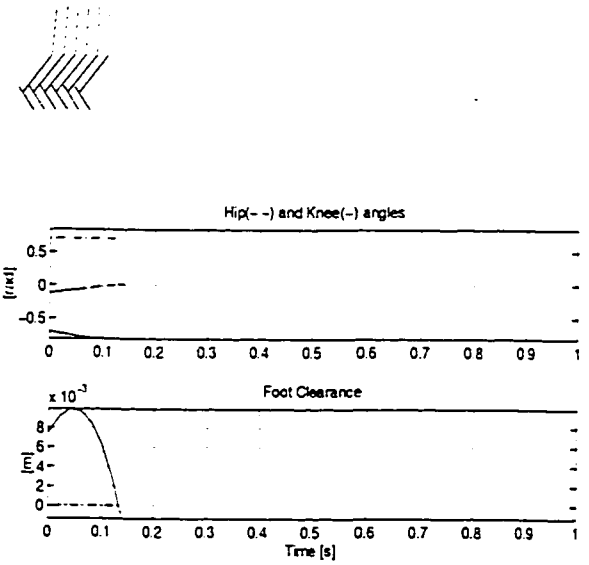


(B) 3D surface of quadriceps stimulation controller

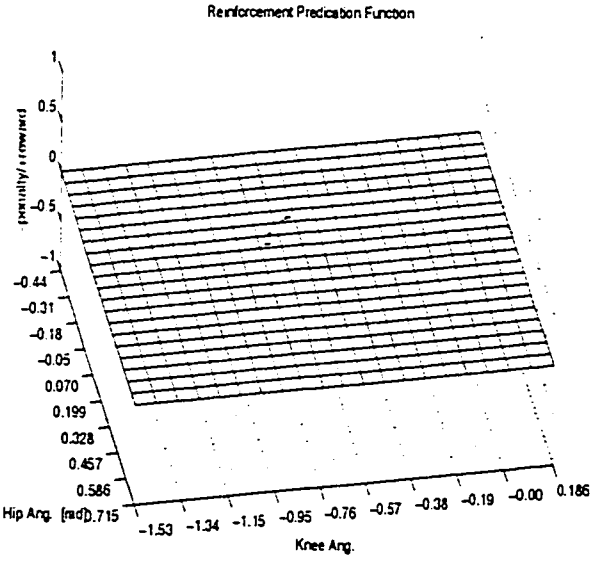
Figure 4.17: Reinforcement learning stage 0.



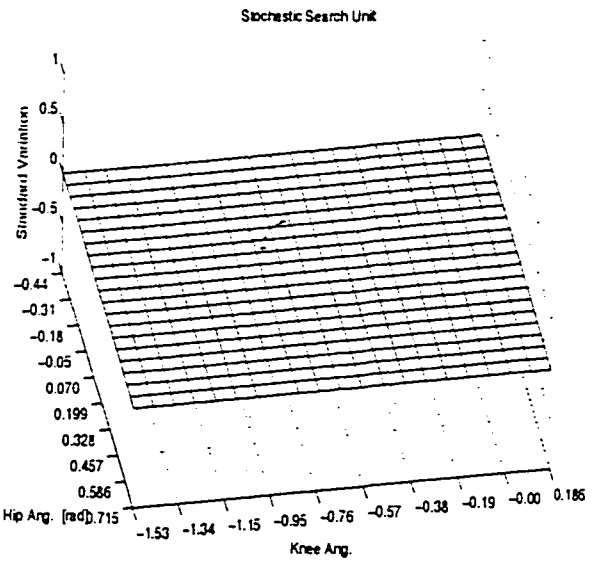
(C) Outputs of controllers



(D) Stick figures, angle trajectory, and foot clearance



(E) 3D surface of Evaluation Net



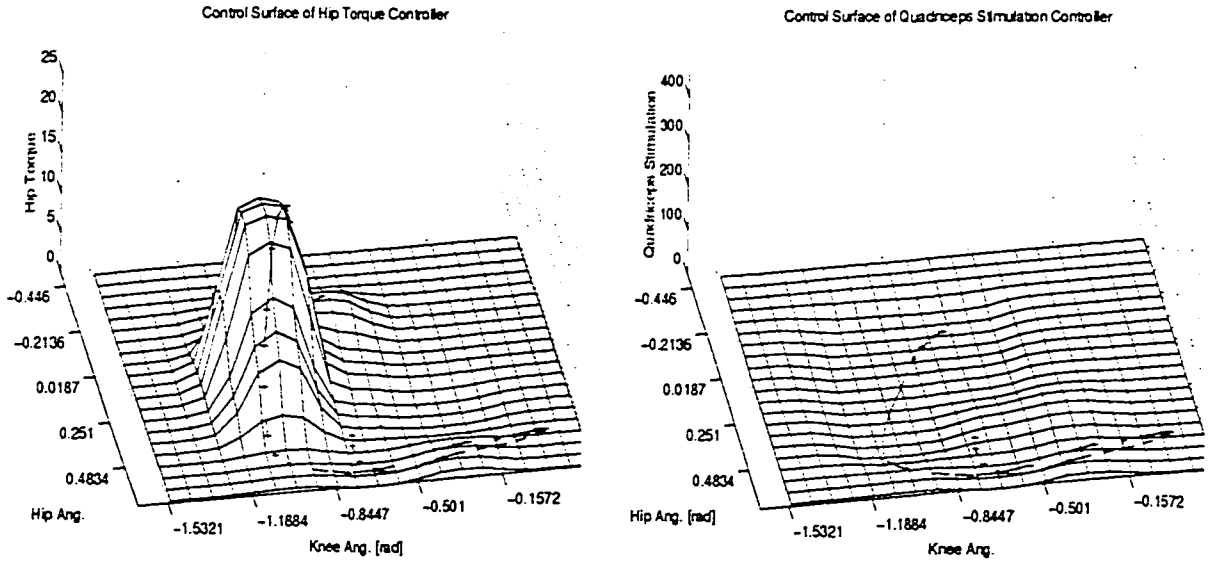
(F) 3D surface of Stochastic Search Unit

Figure 4.17(cont'd): Reinforcement learning stage 0.

Stage 1: Learn to avoid toe collision in the initial swing phase

A negative reinforcement penalty signal (-1) is fed back to the controller every time the toe hits the ground. Gradually, the Evaluation Net will build up the internal reinforcement prediction function to predict the toe collision area in the input state space (figure 4.18(E)). Stochastic Search Unit (figure 4.18(F)) will introduce the stochastic noise into the control system in collision areas for searching optimal control actions. Those control actions moving the leg from collision areas will be rewarded by the Evaluation Net, and those actions moving the leg toward collision areas will be punished by the Evaluation Net. Gradually, the action controllers will learn, under the prediction of Evaluation Net, to increase hip torque (figure 4.18(A)) and decrease quadriceps stimulation (figure 4.18(B)) in the initial swing phase.

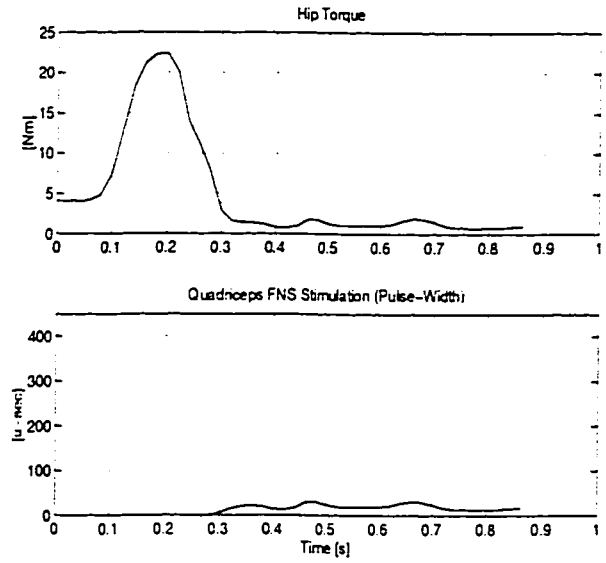
After around 40 trials, the controller successfully avoided toe collision (figure 4.18(D)) by activating hip torque and deactivating quadriceps stimulation in the initial swing (figure 4.18(C)).



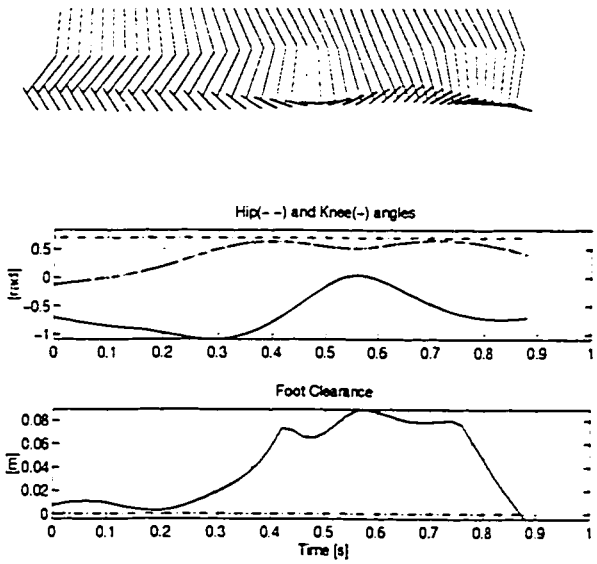
(A) 3D surface of hip torque controller

(B) 3D surface of quadriceps stimulation controller

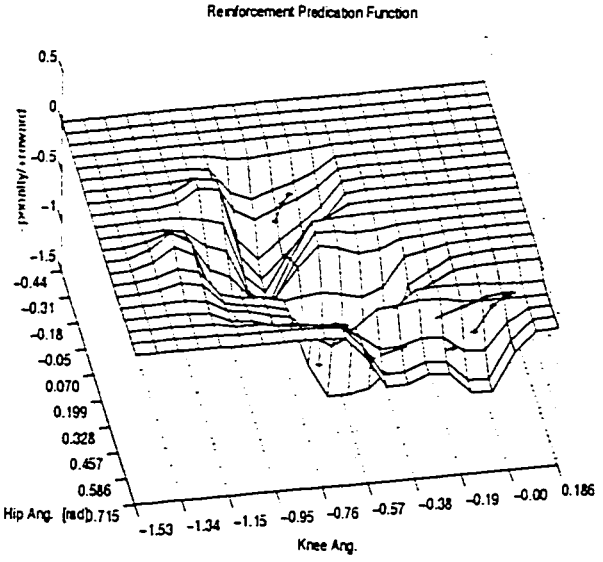
Figure 4.18: Reinforcement learning stage 1.



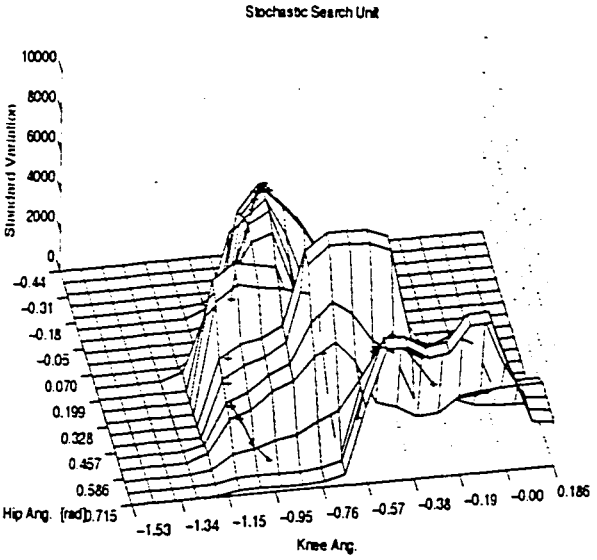
(C) Outputs of controllers



(D) Stick figures, angle trajectory, and foot clearance



(E) 3D surface of Evaluation Net

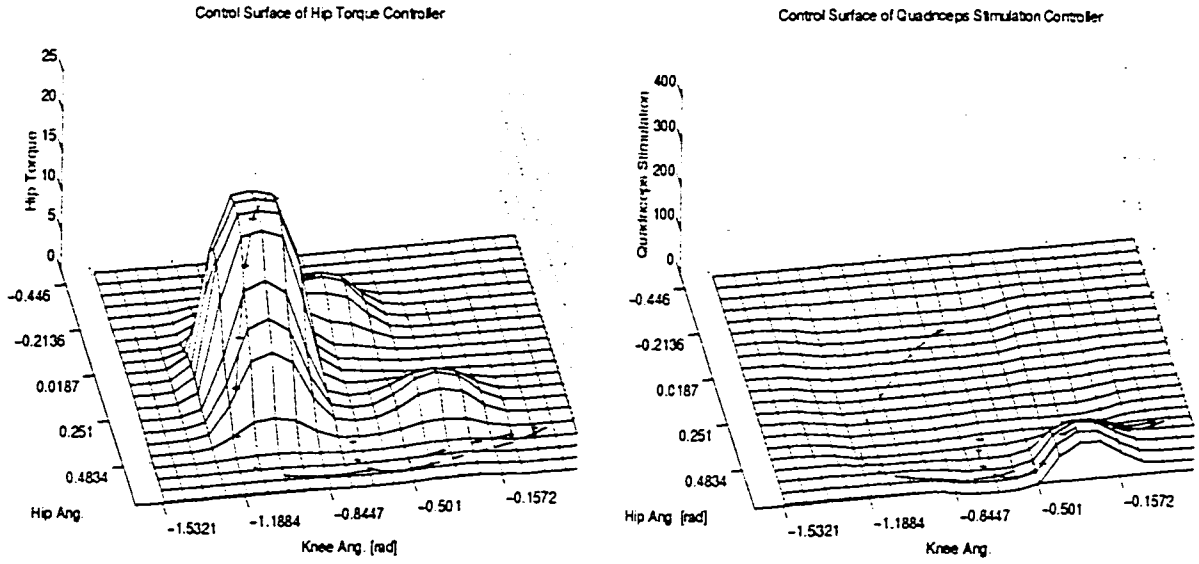


(F) 3D surface of Stochastic Search Unit

Figure 4.18(cont'd): Reinforcement learning stage 1.

Stage 2: Learn to avoid maximum hip angle

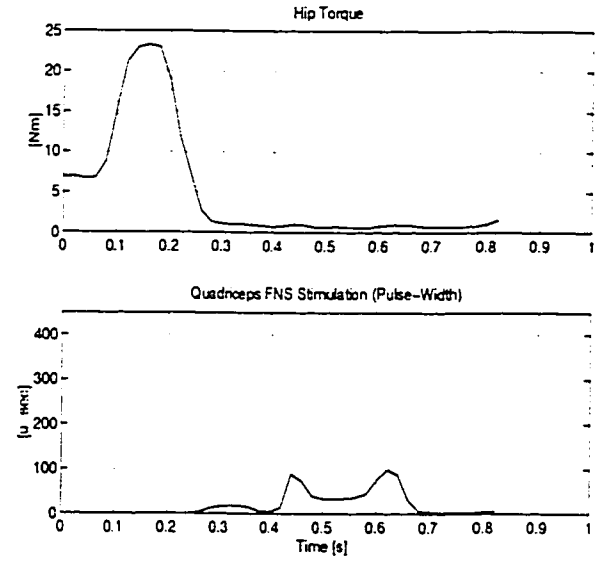
The hip torque is continuously increasing because it moves the leg away from the toe collision area. This will gradually lead to the maximum hip angle violation after hip torque is increased too much. Then a negative reinforcement penalty signal (-1) is fed back. Two negative peaks in the lower portion in figure 4.19(E) reflect the hip angle maximum violation situation. The Stochastic Search Unit (figure 4.19(F)) will introduce stochastic noise in the hip angle maximum areas for searching optimal control actions. The result is to decrease hip torque. Gradually, the hip torque will reach a balance point so that it is large enough to avoid toe collision, but not too large to avoid exceeding the maximum hip angle. Then, learning will proceed into another stage.



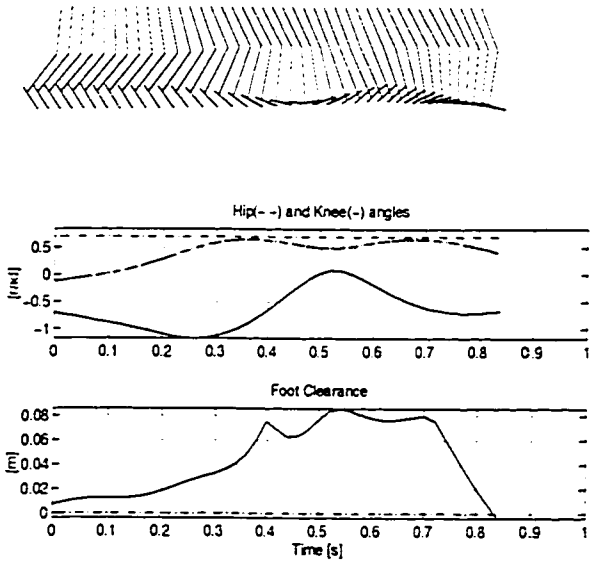
(A) 3D surface of hip torque controller

(B) 3D surface of quadriceps stimulation controller

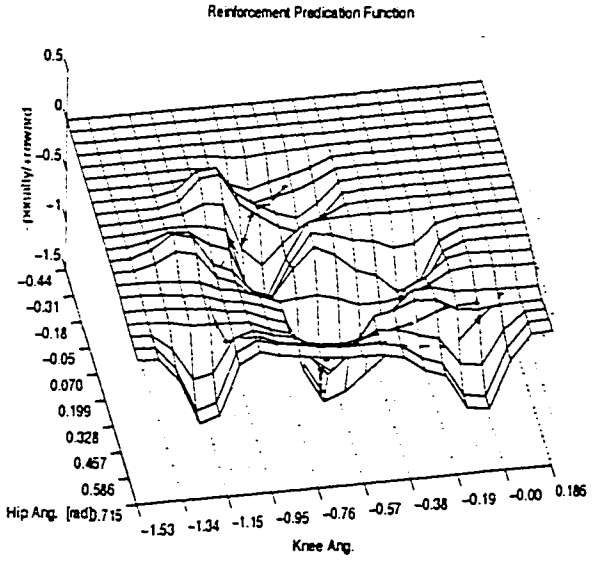
Figure 4.19: Reinforcement learning stage 2.



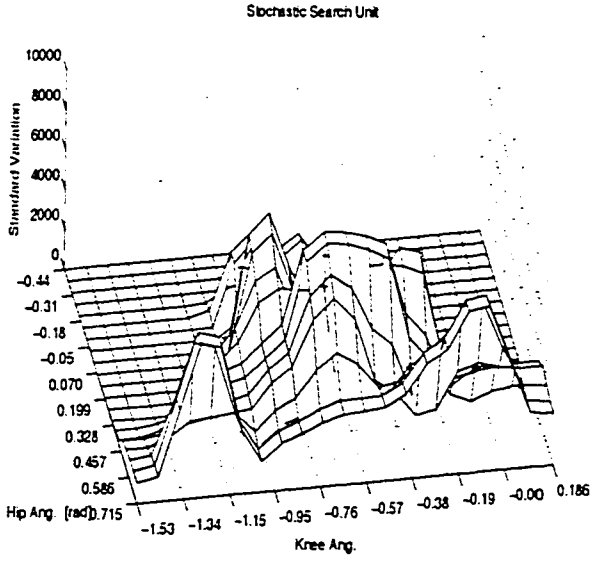
(C) Outputs of controllers



(D) Stick figures, angle trajectory, and foot clearance



(E) 3D surface of Evaluation Net

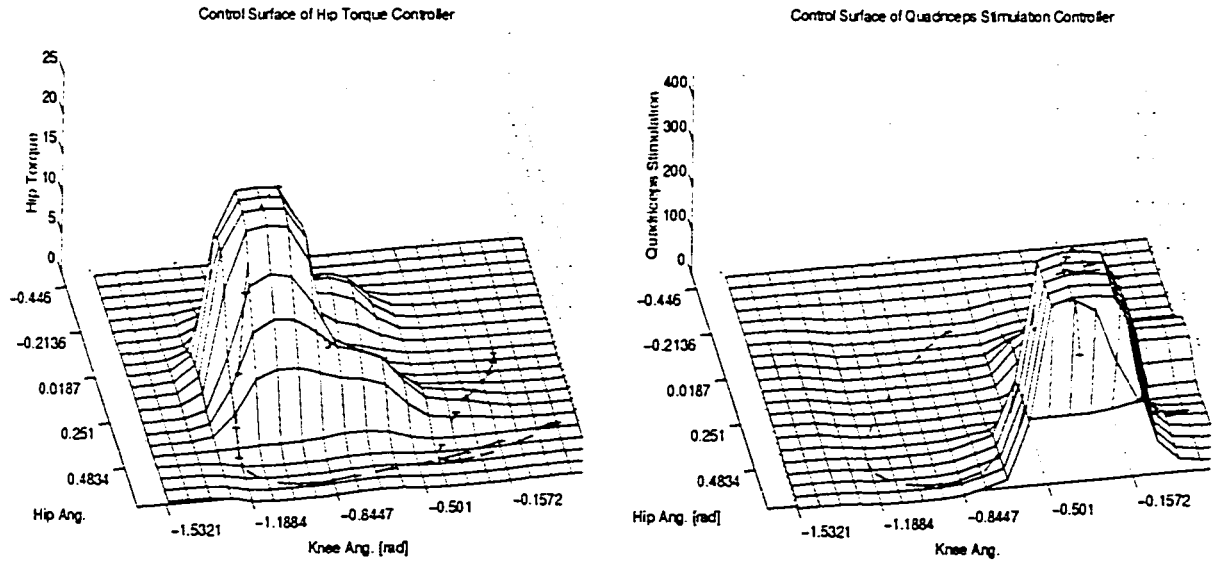


(F) 3D surface of Stochastic Search Unit

Figure 4.19(cont'd): Reinforcement learning stage 2.

Stage 3: Learn to extend knee at the end of swing phase

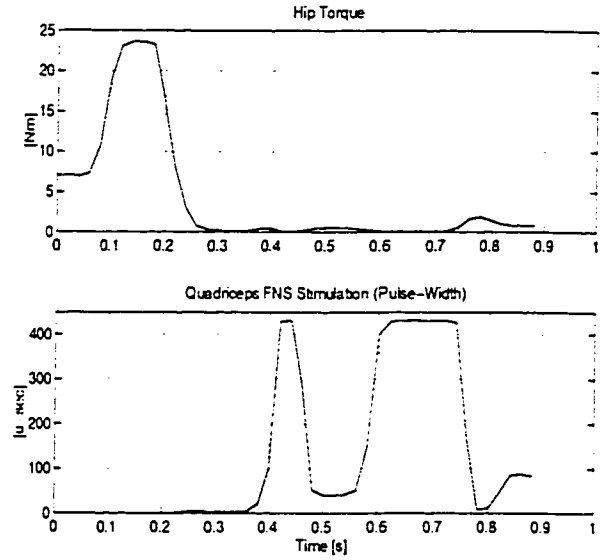
The hip torque increase has been limited, and the small hip torque (1.1Nm, initialization value) after 0.3 seconds has been further decreased to zero (figure 4.20(C)). No more hip angle maximum violations will happen. The controller can control the leg to avoid both toe collision and hip angle maximum violation in the initial swing phase and enter terminal swing phase. Since there is not enough quadriceps stimulation to extend the knee at terminal swing phase, knee angle will be larger than the acceptable value [-0.1rad 0.1rad]. A negative reinforcement penalty signal, proportional to the knee angle at heel contact, will be fed back. The negative areas in the low-right portion in figure 4.20 (E) shows this situation. Then, the quadriceps stimulation is improved (figure 4.20B) to decrease the negative penalty, thus decrease the knee angle at the heel contact. Since there is a long delay from electrical stimulation to muscle activation, this stage of learning takes nearly one hundred trials.



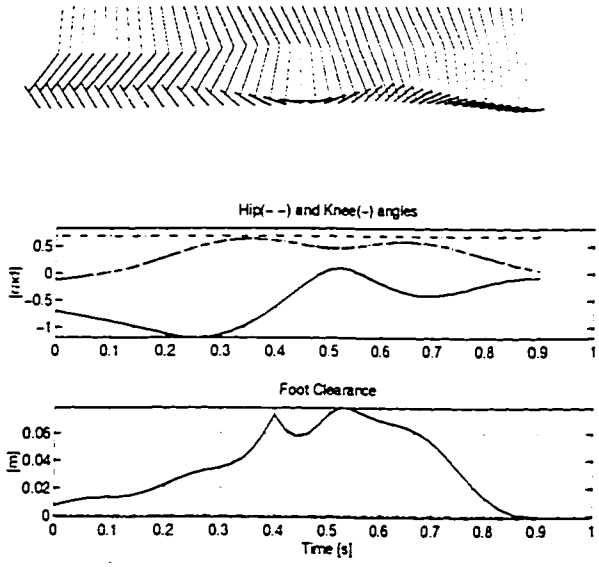
(A) 3D surface of hip torque controller

(B) 3D surface of quadriceps stimulation controller

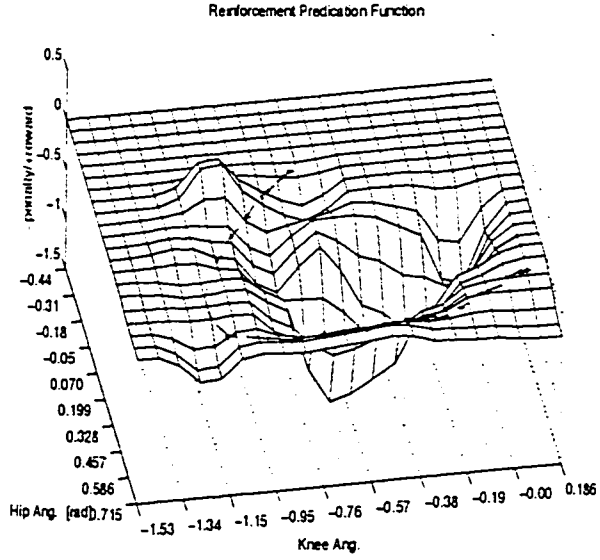
Figure 4.20: Reinforcement learning stage 3.



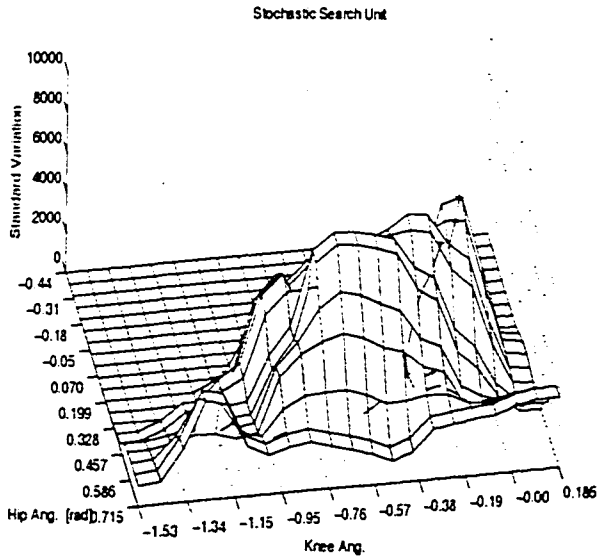
(C) Outputs of controllers



(D) Stick figures, angle trajectory, and foot clearance



(E) 3D surface of Evaluation Net

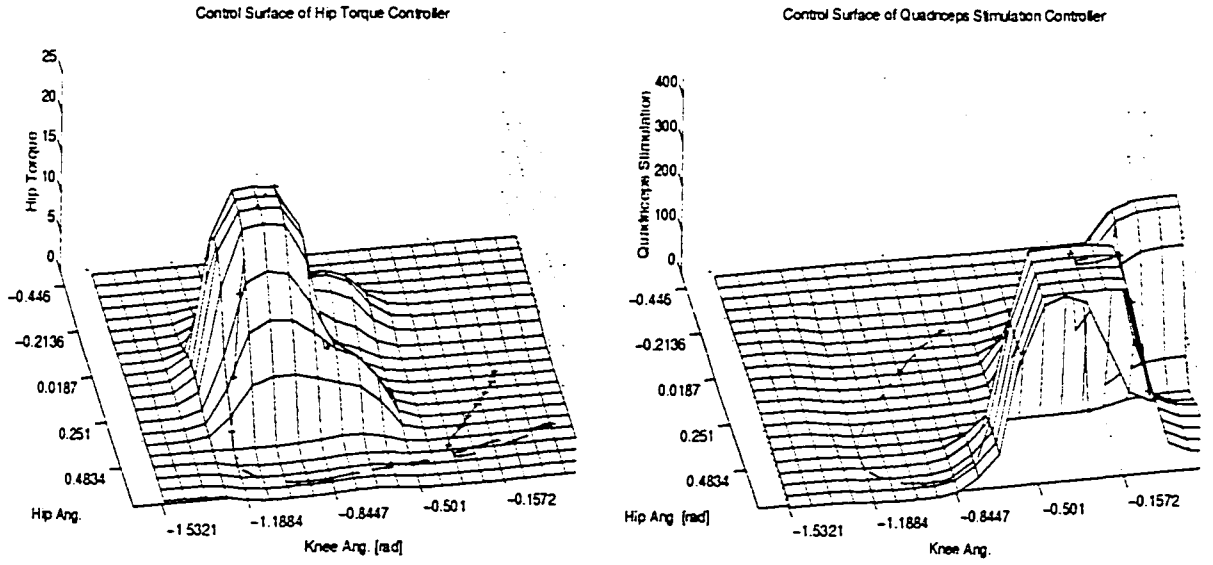


(F) 3D surface of Stochastic Search Unit

Figure 4.20(cont'd): Reinforcement learning stage 3.

Stage 4: Convergence of reinforcement learning

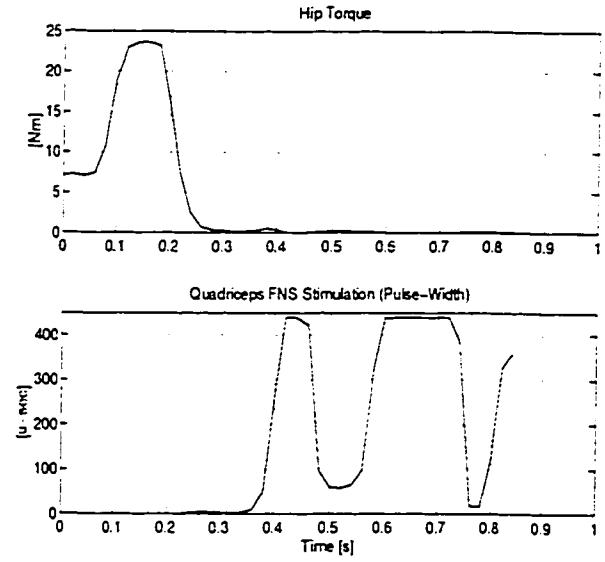
Finally, the quadriceps stimulation has been increased enough (figure 4.21(B)), and thus, knee angle at the heel contact has been reduced to acceptable range. Then, a positive reinforcement reward signal (+1) was fed back. The positive peak in the middle-right portion of figure(4.21E) indicates the final goal area for the swing leg. Now, optimal hip torque and quadriceps controllers have been learned (figure 4.21(A)(B)). After several successful trials, the positive reinforcement signal was propagated back along this collision-free trajectory. Thus, there was no more stochastic search along this trajectory (figure 4.21(F)). The stochastic controller then converged into a deterministic controller.



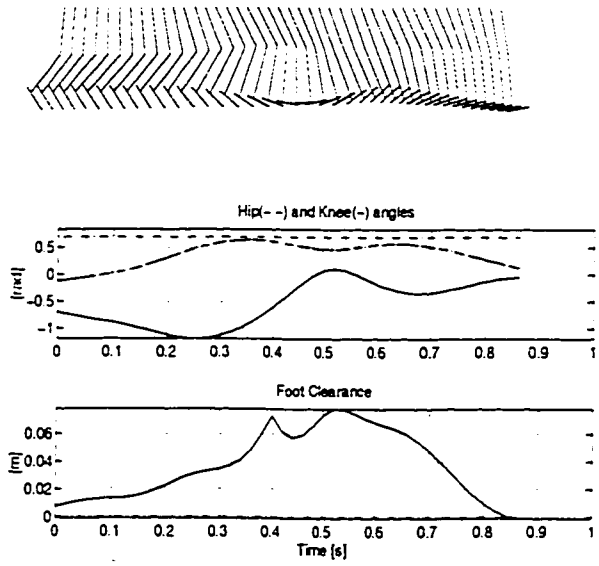
(A) 3D surface of hip torque controller

(B) 3D surface of quadriceps stimulation controller

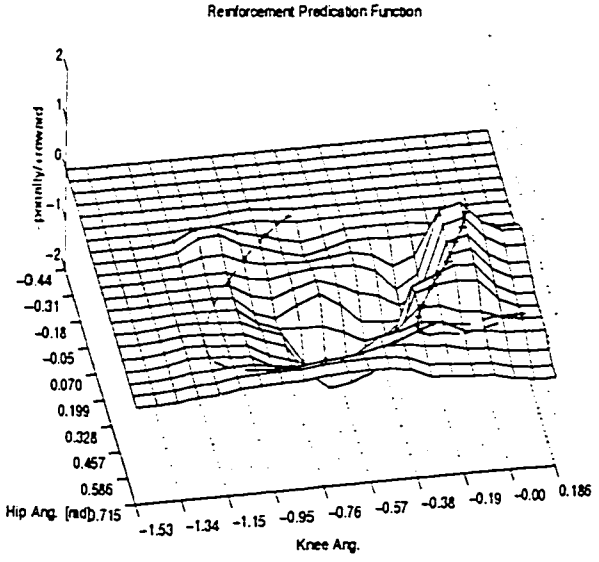
Figure 4.21: Reinforcement learning stage 4.



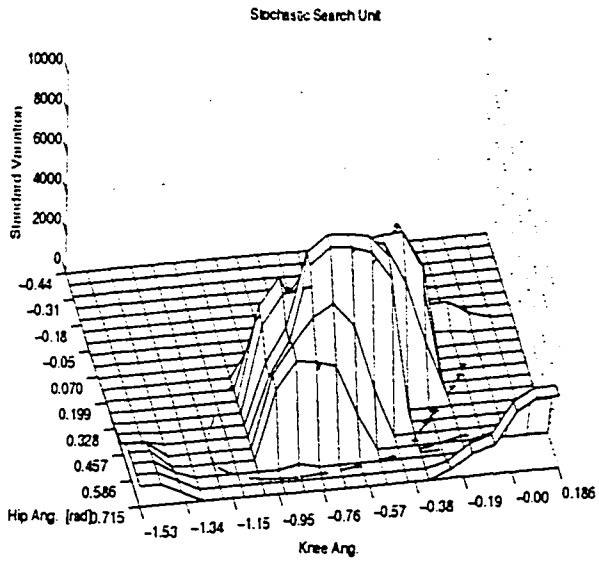
(C) Outputs of controllers



(D) Stick figures, angle trajectory, and foot clearance



(E) 3D surface of Evaluation Net



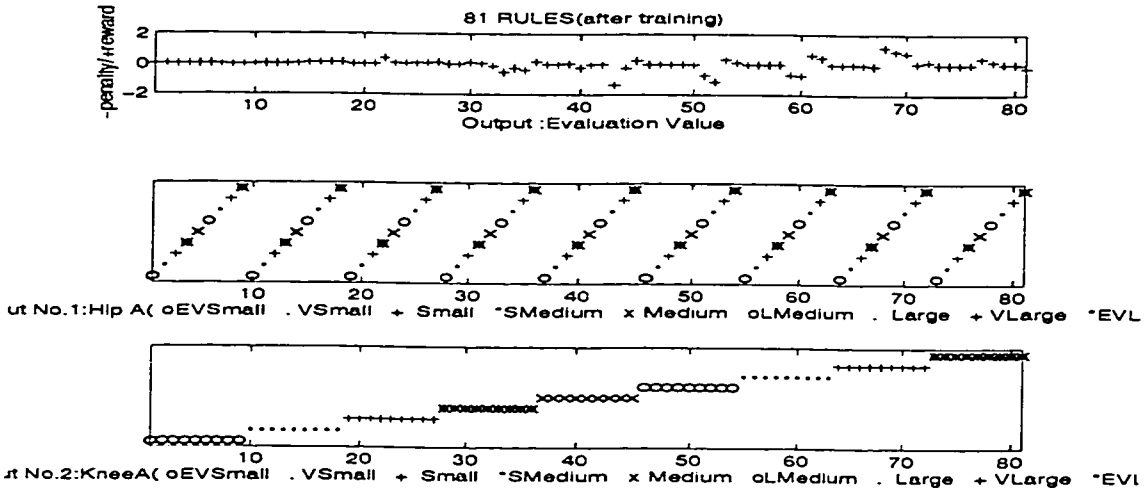
(F) 3D surface of Stochastic Search Unit

Figure 4.21(cont'd): Reinforcement learning stage 4.

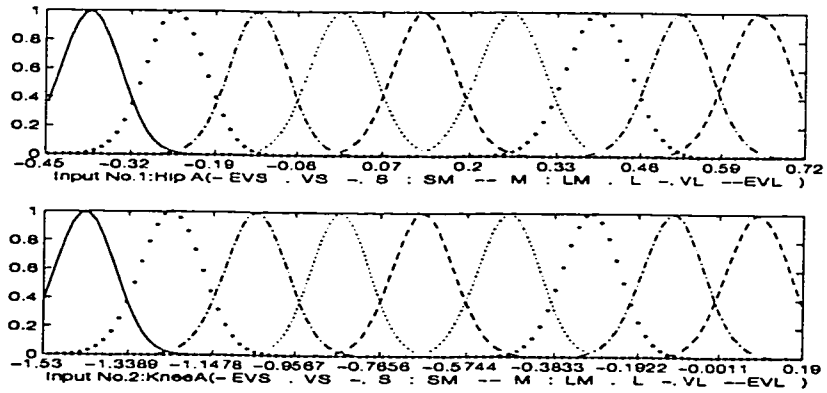
Membership functions of input fuzzy variables and output fuzzy singletons after reinforcement learning are shown in figure 4.22. 4.23.and4.24. for the Evaluation Net. hip torque controller. and quadriceps stimulation controllers. respectively.

Figure 4.25 is a contour plot of the learned Evaluation Net. Comparing this figure with figure 4.16. it is obvious that the Evaluation Net does predict the penalty and reward area quit accurately. It is accurate enough to guide the action controllers to find optimal control rules to avoid penalty and seek reward.

Conclusion: The reinforcement learning controller can learn. based on penalty/reward reinforcement signals. and without explicit teacher signals. to avoid toe collision at the initial swing phase. avoid maximum hip angle. and extend the knee at the terminal swing phase.

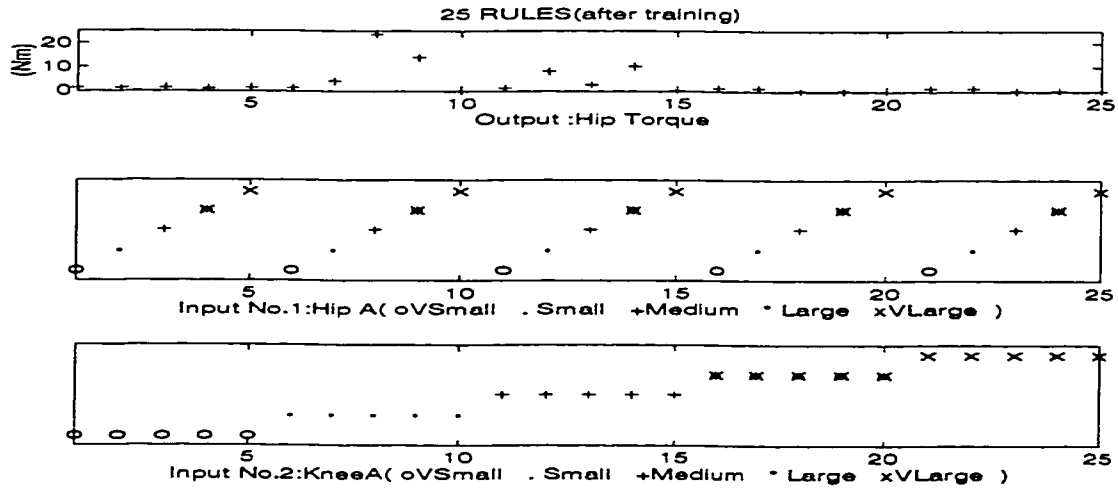


(A) Output fuzzy singletons, fuzzy rules in graphic form

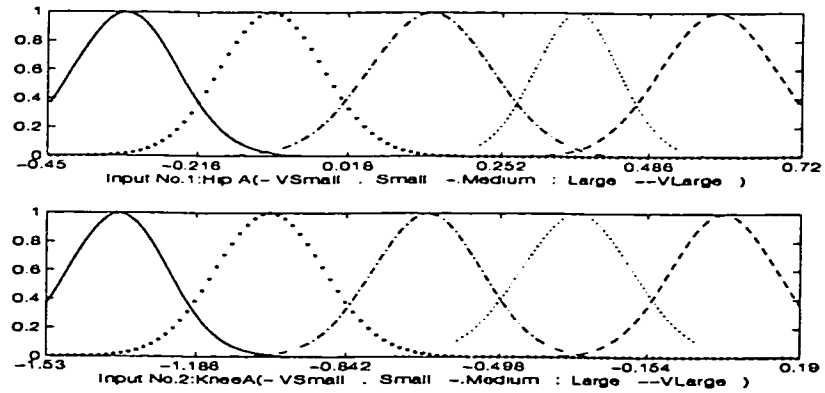


(B) Membership functions for input fuzzy variables

Figure 4.22: Fuzzy rule base of evaluation net(after reinforcement learning).

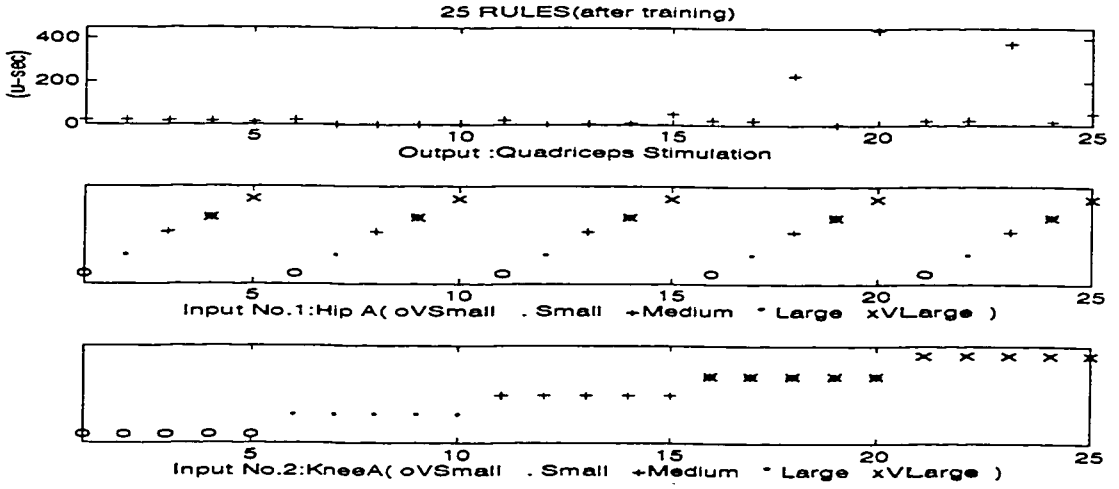


(A) Output fuzzy singletons. fuzzy control rules in graphic form

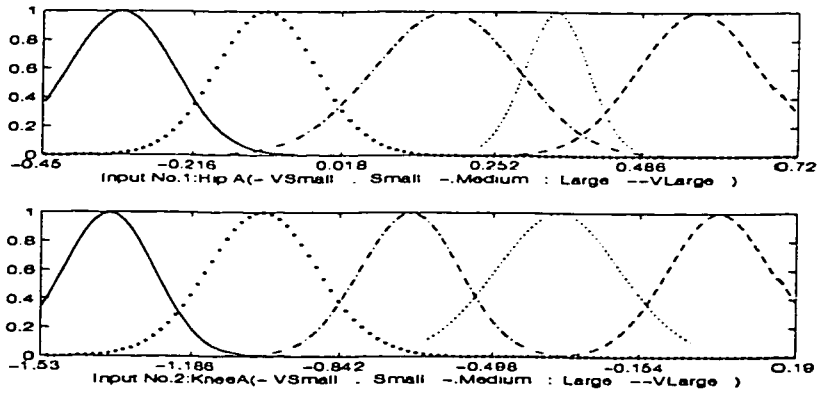


(A) Membership functions of input fuzzy variables

Figure 4.23: Fuzzy rule base of hip torque controller(after reinforcement learning).



(A) Output fuzzy singletons, fuzzy control rules in graphic form



(A) Membership functions of input fuzzy variables

Figure 4.24: Fuzzy rule base of quadriceps stimulation controller (after reinforcement learning).

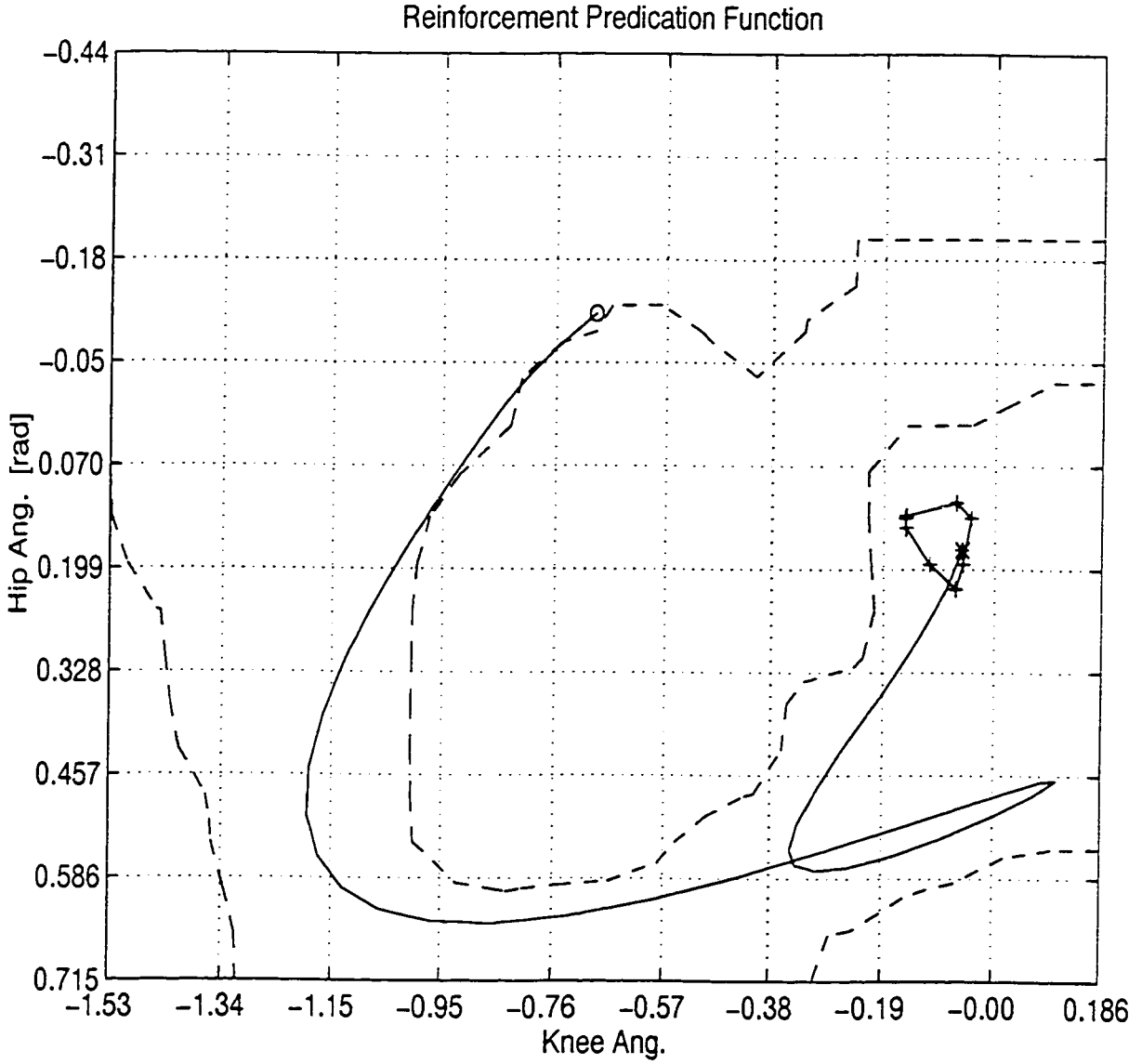


Figure 4.25: Penalty and reward area predicted by Evaluation Net. Dashed lines indicate where predicted reinforcement=0 (thus indicating the collision areas such as toe collision, hip angle maximum, and knee angle). Solid line with + signs indicate where predicted reinforcement > 0.9 (for reward). Grid lines dedicate fuzzy partitioning of input space. Trajectory controlled by reinforcement learning controller is also shown (solid line) with start point(o) and end point(*). For a three dimensional plot, see figure (4.21E). This predicated evaluation function is close to the actual evaluation function (Fig.4.16) .

4.2.3 Adaptability to Parameter Variations

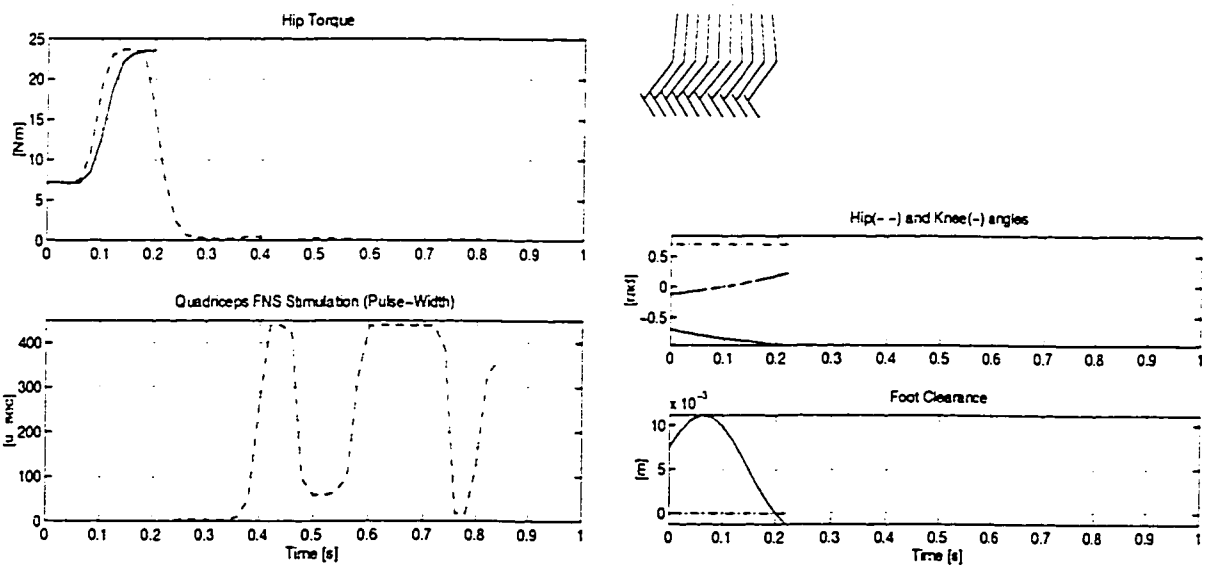
Similar to the closed-loop supervised learning controller, the reinforcement learning controller can also handle model parameter variations within a certain range by using the sensor feedback with learned control rules. The simulation results are very similar to figure 4.11 to 4.14. But more interesting, the reinforcement learning controller is able to re-adapt (or fine-tune) its learned control rules if these control rules fail, as described in the next section.

4.2.4 Using Previous Learning Experience in New Model

(1) Muscle Fatigue

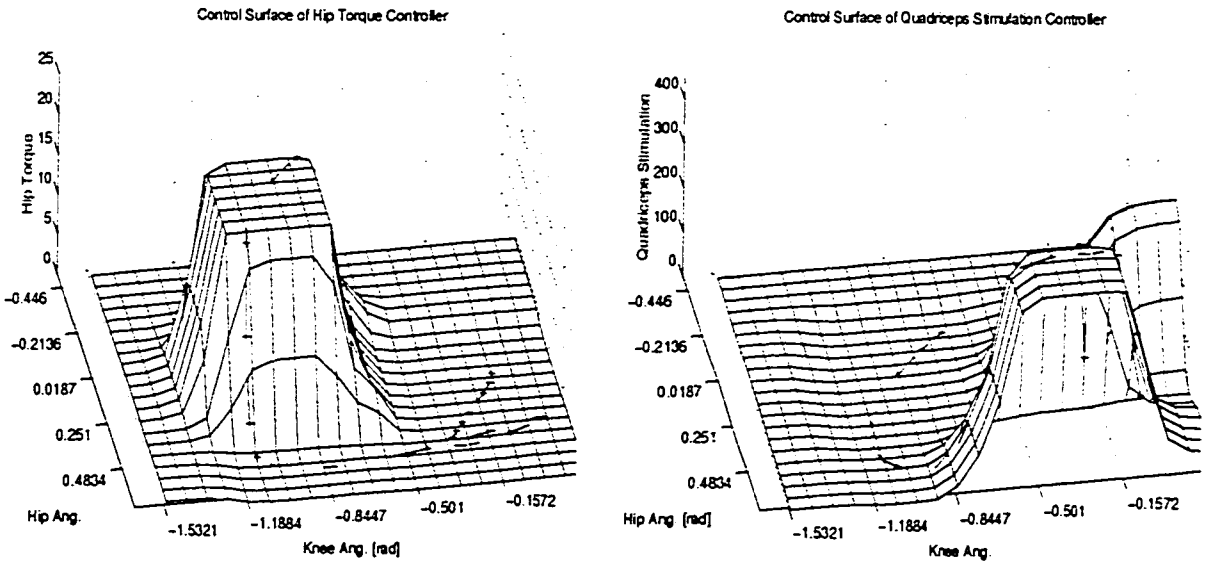
Hip torque gain was changed to 0.60, and quadriceps muscle gain remained 1.00.

Even with closed-loop sensor feedback, the controller failed. The toe hit the ground (figure 4.26A) due to very low gain. The reinforcement learning controller re-adapted by learning the new penalty area for the Evaluation Net, reactivating the Stochastic Search Unit in the new penalty area, and searching for new optimal control rules. This time, since there were previously learned control rules (figure 4.21(A)(B)) and the Evaluation Net (figure 4.21(E)) as the start point, learning took fewer trials than learning without experience. The old learned control rules only needed to be fine-tuned to fit the new situation. After 20 trials, a new controller with higher hip torque at the initial swing phase was learned (figure 4.26B) and converged to (stochastic search noise was zero along the trajectory). Under control of this new controller, successful swing is achieved (figure 4.26C).

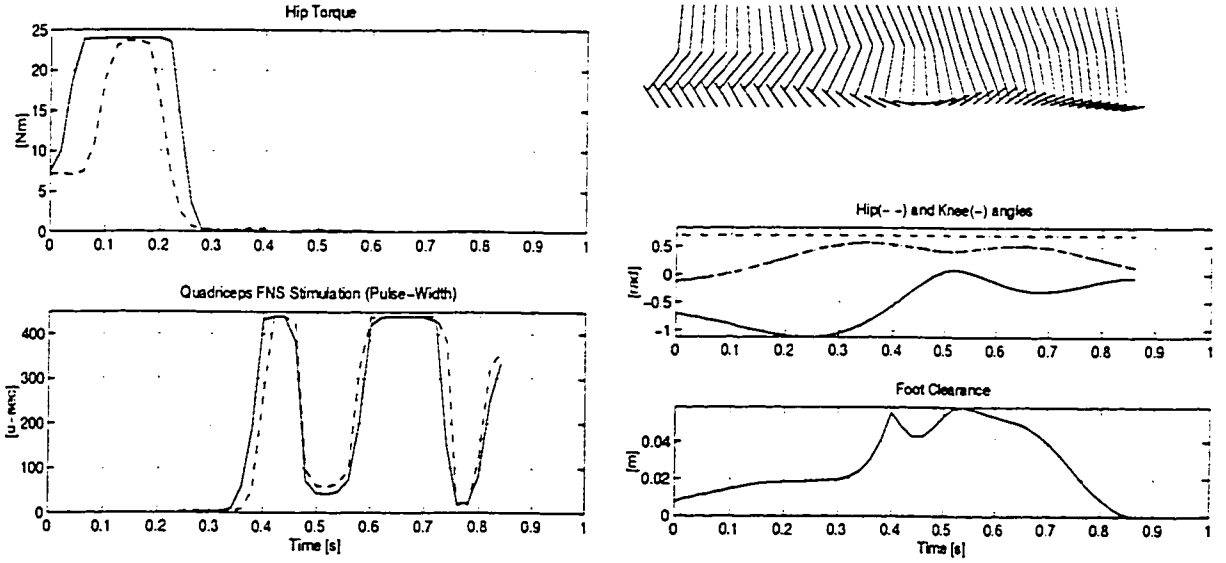


(A)Closed-loop reinforcement learning controller failed with learned control rules. The control signals for normal gain are also shown with dashed lines.

Figure 4.26: Fine-tuning of learned controller for very low gain (muscle fatigue).



(B) Reinforcement learning controller learned new control surfaces.

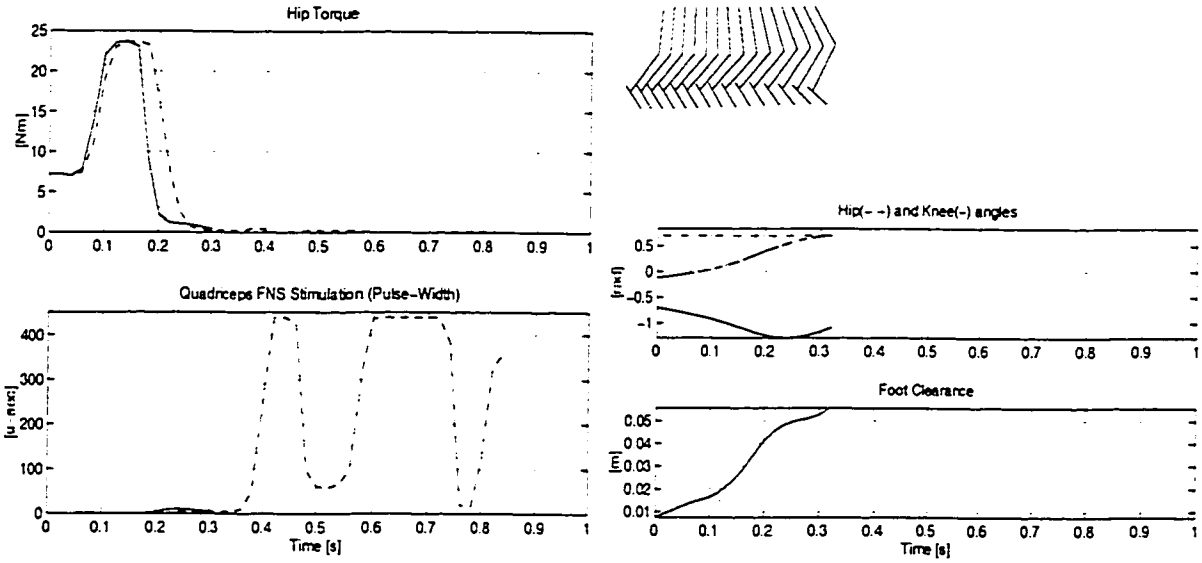


(C) Reinforcement learning controller succeeded with new control surfaces. The control signals for normal gain are also shown with dashed lines.

Figure 4.26(cont'd): Fine-tuning of learned controller for very low gain (muscle fatigue).

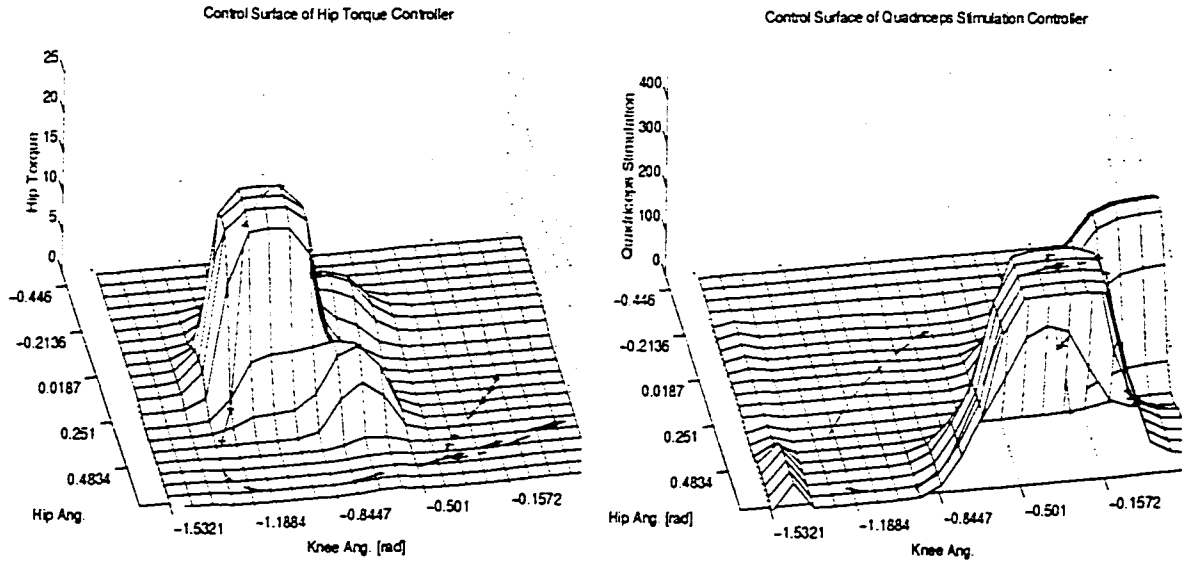
(2) Muscle Potentiation

Hip torque gain was changed to 1.30. and quadriceps muscle gain was changed to 1.30. The closed-loop controller failed with the old rules (figure 4.27A). It even attempted to use less hip torque, but still failed due to very high gain. The controller re-adapted (figure 4.27(B)), and succeeded with new control surfaces(figure 4.27(C)).

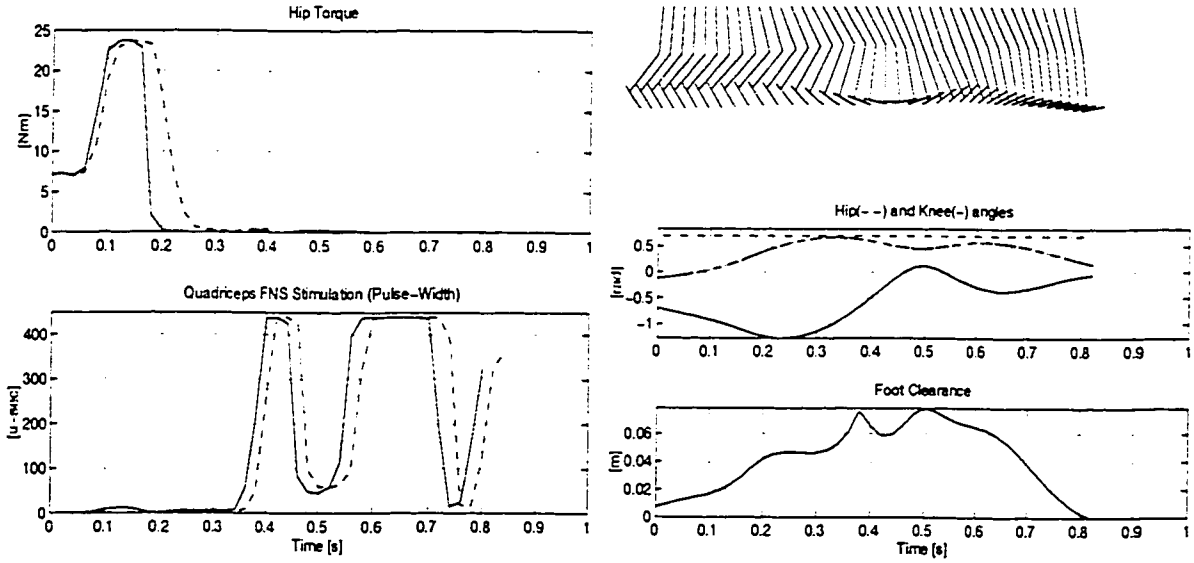


(A) Closed-loop reinforcement learning controller failed with learned control rules (The control signals for normal gain are also shown with dashed lines).

Figure 4.27: Fine tuning of learned controller for very high gain (muscle potentiation).



(B) Reinforcement learning controller learned new control surfaces.

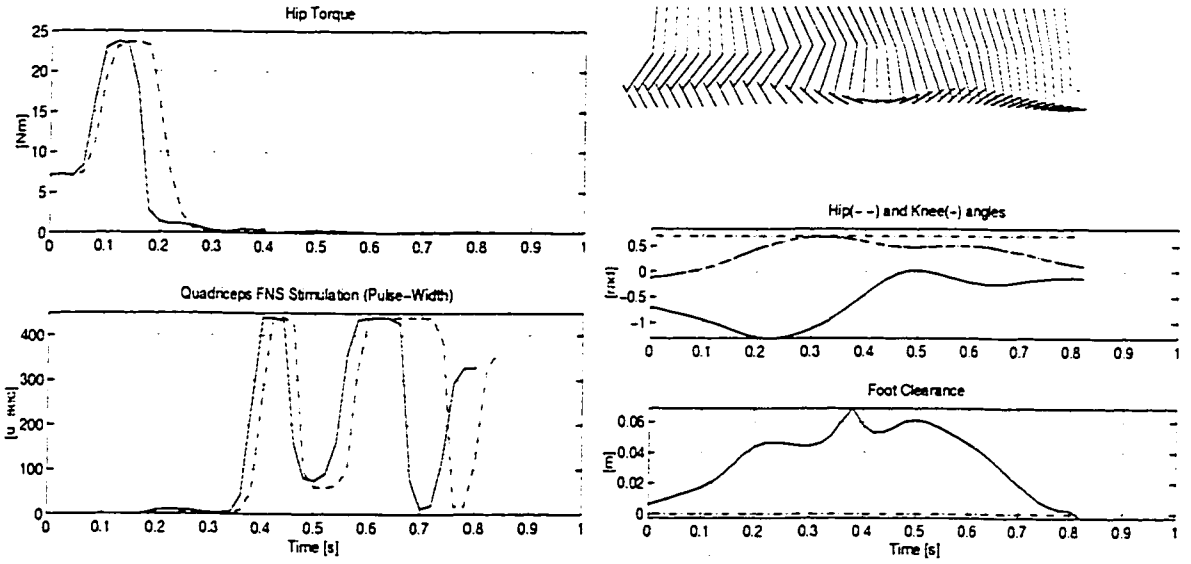


(C) Reinforcement learning controller succeeded with new control surfaces. The control signals for normal gain are also shown with dashed lines.

Figure 4.27(cont'd): Fine tuning of learned controller for very high gain (muscle potentiation).

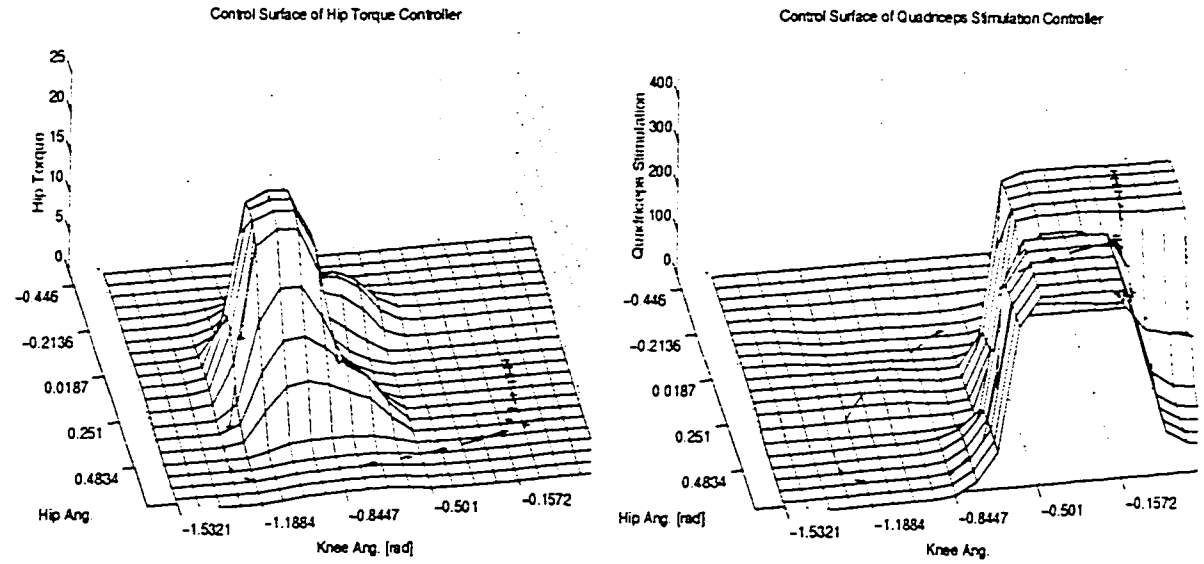
(3) Light body mass

Body mass was changed to 45kg. and body height to 1.50m. The controller failed with old control rules (figure 4.28A). It re-adapted (figure 4.28B) and succeeded with new control rules(figure 4.28(C)).

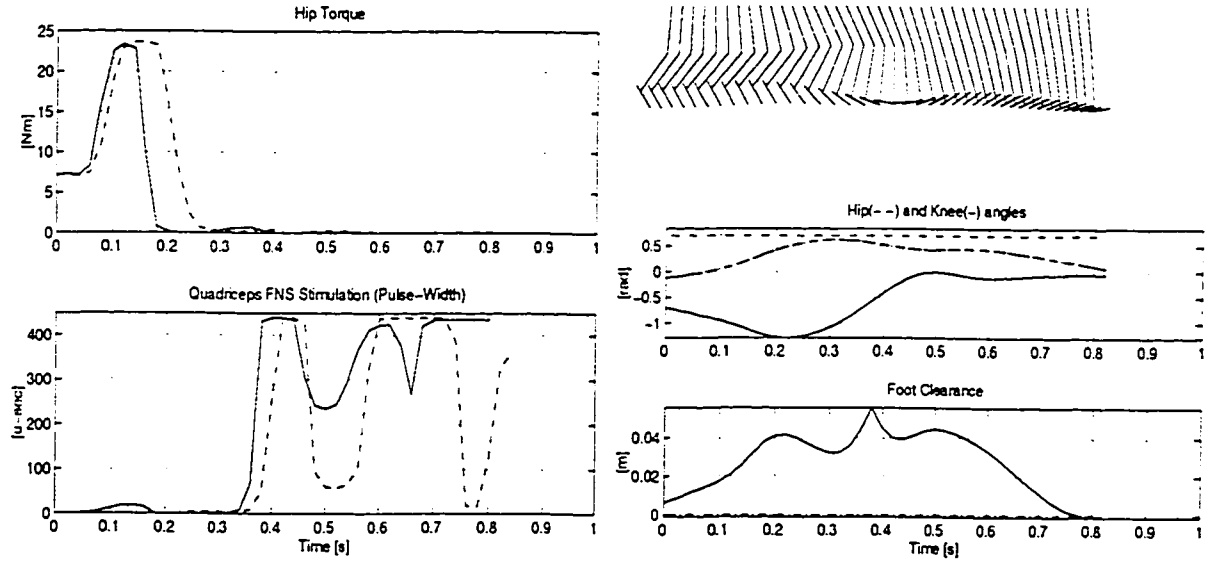


(A) Closed-loop reinforcement learning controller failed with learned control rules. The control signals for normal gain are also shown with dashed lines.

Figure 4.28: Fine-tuning of learned controller for very light body mass



(B) Reinforcement learning controller learned new control surfaces.

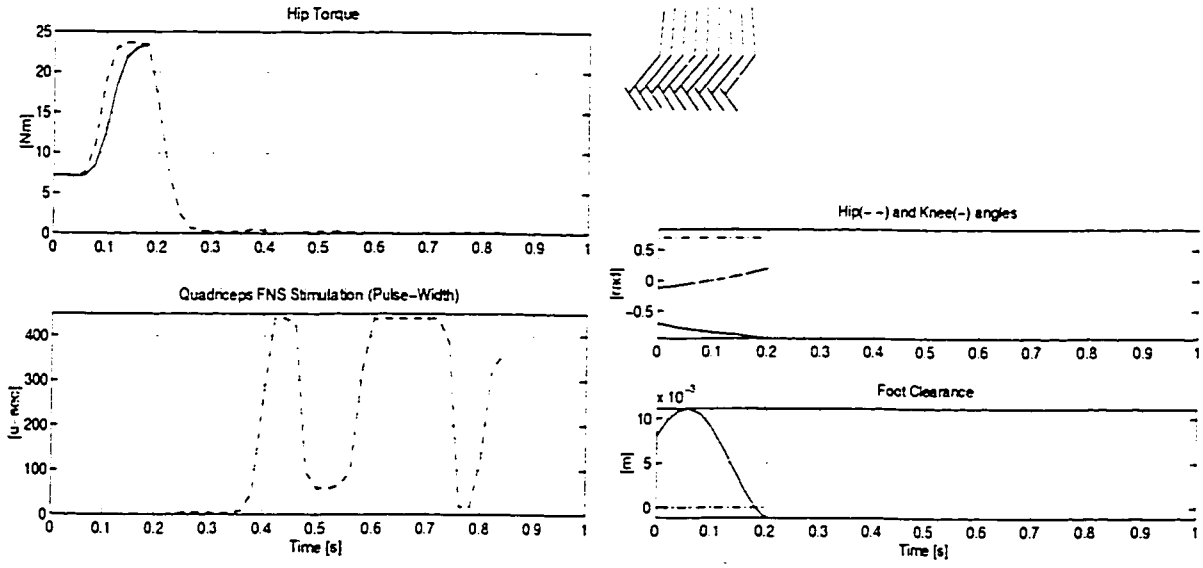


(C) Reinforcement learning controller succeeded with new control surfaces. The control signals for normal gain are also shown with dashed lines.

Figure 4.28(cont'd): Fine-tuning of learned controller for very light body mass

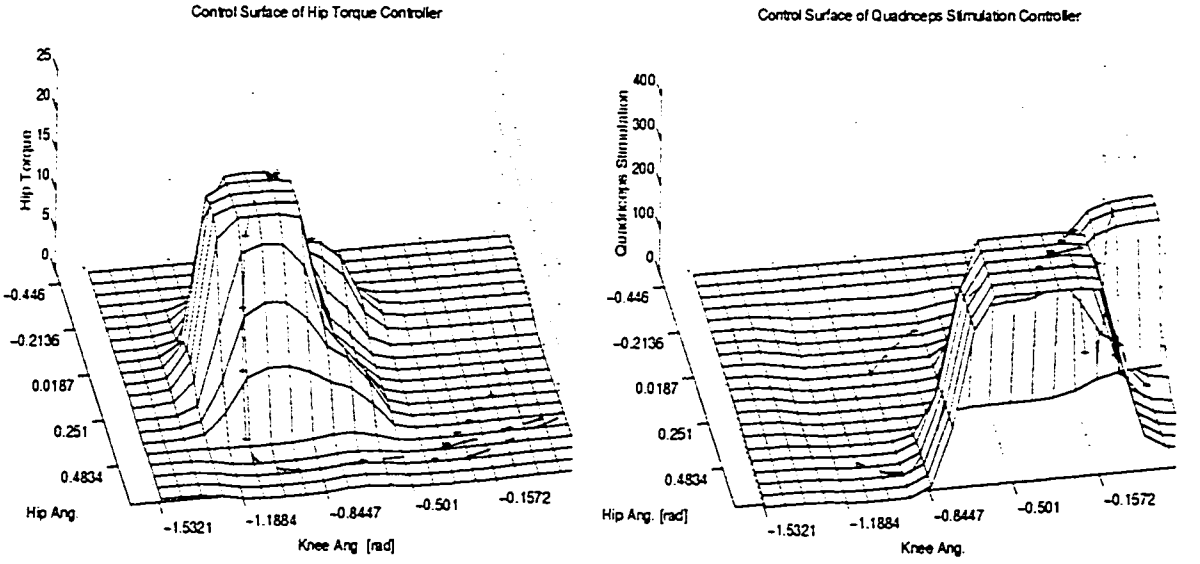
(4) Heavy Body Mass

Body mass was changed to 75kg. and body height 1.70m. The controller failed with learned control rules(figure 4.29A). It readapted (figure 4.29B). and succeeded with new control rules(figure 4.29C).

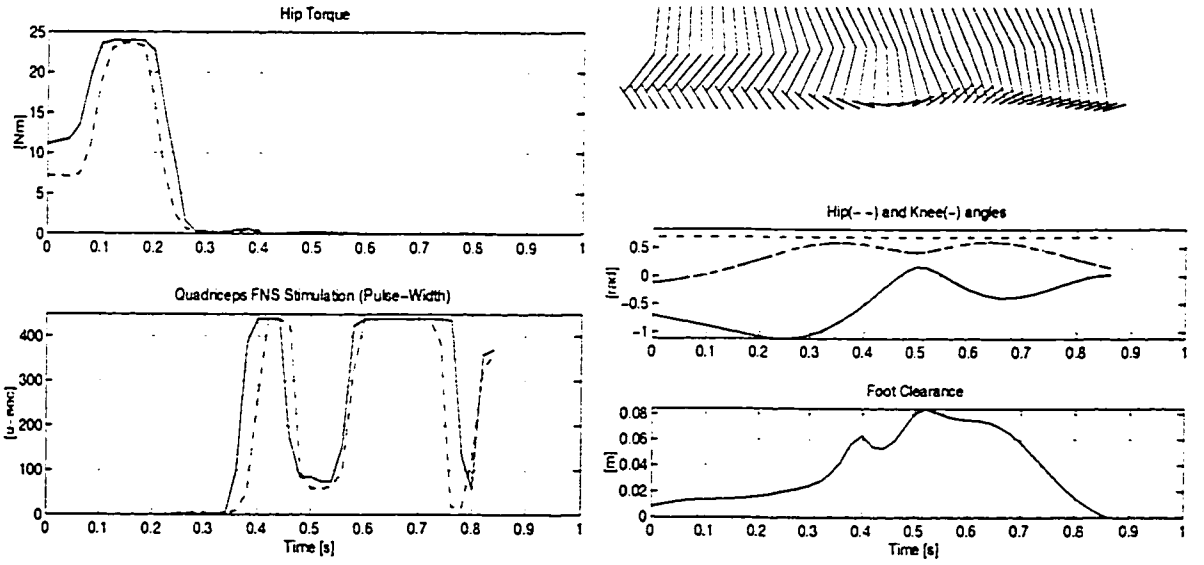


(A) Closed-loop reinforcement learning controller failed with learned control rules. The control signals for normal gain are also shown with dashed lines.

Figure 4.29: Fine-tuning of learned controller for very heavy body mass.



(B) Reinforcement learning controller learned new control surfaces.



(C) Reinforcement learning controller succeeded with new control surfaces. The control signals for normal gain is also shown with dashed lines.

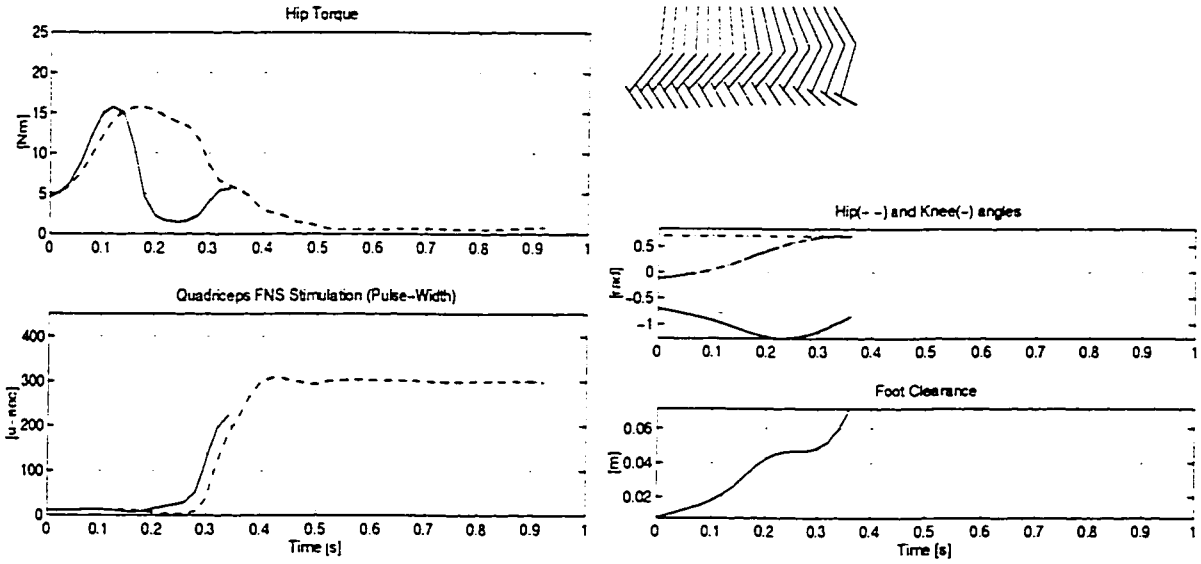
Figure 4.29(cont'd): Fine-tuning of learned controller for very heavy body mass.

4.3 Combining Supervised Learning with Reinforcement Learning

It is possible to combine supervised learning and reinforcement learning algorithms. Supervised learning is very fast to learn a general control rule base from a typical subject. Then reinforcement learning can be used to fine-tune (or customize) the general control rules for different subjects. If there are hand-crafted control rules available from human experts, then they can also be used as a starting point, and reinforcement learning can be used to fine tune the control rules for different subjects.

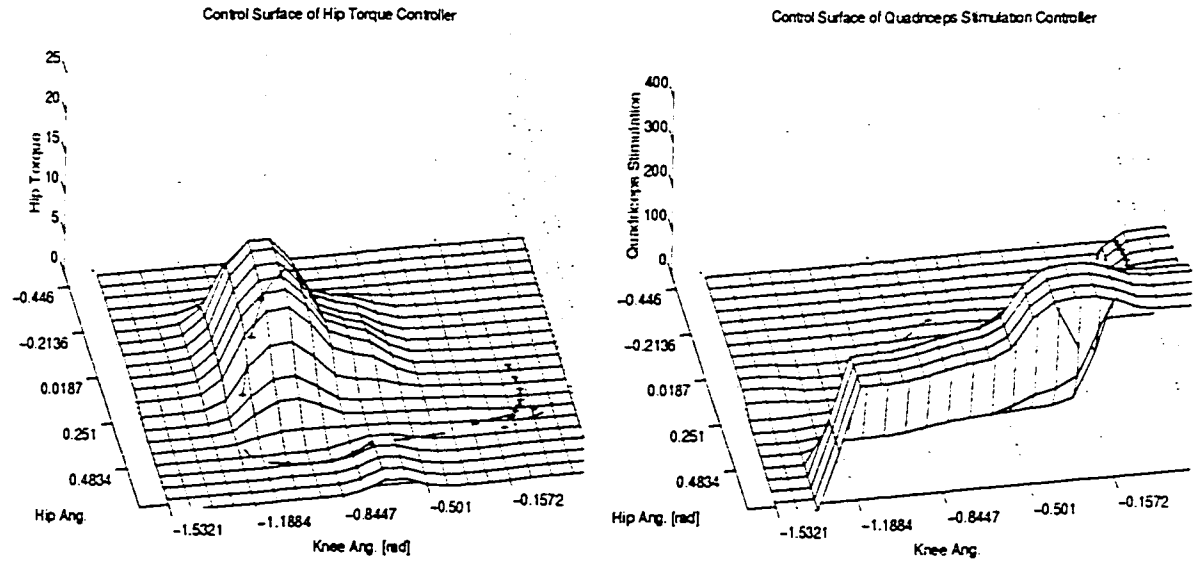
Figure 4.30 demonstrates this possibility. The control rules from supervised learning (figure 4.7 4.8 4.9 work fine with the original model. But for a new model with hip torque gain = 1.80 (a change of 80%), and quadriceps muscle gain = 1.80 (a change of 80%), the old learned controller will fail (figure 4.30A), even though this closed-loop controller did try to use less hip torque. The reinforcement learning algorithm is used to fine-tune the supervised-learned control rules and arrive at new control surfaces (figure 4.30B). With the new control rules, swing phase is successful (figure 4.30C). Note that the total learning takes approximately 30 trials, including less than 10 trials (Fig.4.6) for supervised learning to learn optimal control rules from the teacher for a general model, and 20 trials for reinforcement learning to fine-tune the old control rules for a new model.

Note that supervised learning did not generate evaluation functions, only action functions were learned. Yet it still reduced the training trials for reinforcement learning by placing control systems near the optimal points.

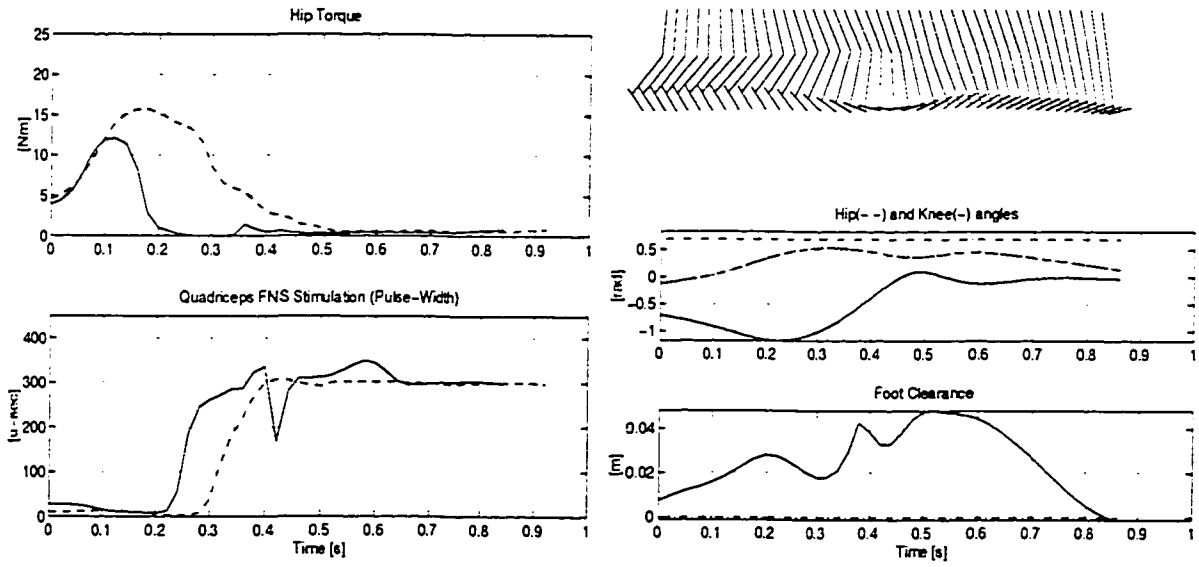


(A) Closed-loop supervised learning controller failed. The control signals for normal gain are also shown with dashed lines.

Figure 4.30: Fine-tuning of supervised learning control rules by reinforcement learning algorithm.



(B) Reinforcement learning controller learned new control surfaces.



(C) Learning controller succeeded with new control surfaces. The control signals for normal gain are also shown with dashed lines.

Figure 4.30(cont'd): Fine-tuning of supervised learning control rules by reinforcement learning algorithm.

Chapter 5

Discussion and Conclusions

Choosing control strategies is the first step in the design of controllers for Functional Neuromuscular Stimulation. Through extensive literature review, it is argued that finite state control is more suitable than conventional numerical control for Functional Neuromuscular Stimulation prostheses with many practical sensors (e.g. accelerometer, inclinometer, force sensing resistor) providing sensory signals not clearly related to the plant dynamics. Neuro-fuzzy systems that combine rule-based fuzzy systems with neural networks have advantages over individual systems used separately, and offer flexible frameworks to implement adaptive finite state control. Through literature review in Chapter 2, the thesis objective No.1 (section 2.3.3) has been achieved.

Based on the review of previous work, an Adaptive Fuzzy Network (AFN) was developed as a neuro-fuzzy system with supervised learning and reinforcement learning mechanisms. Generalized Gaussian function with three adjustable parameters (center, width, and slope) was used as differentiable membership functions in the input fuzzifier, and fuzzy singletons were used as the consequent parts of fuzzy rules. A product-sum-normalization inference was adopted. This normalization procedure was important in adapting fuzzy rules when plant state moves outside the state space covered by the fuzzy rule base. A squashing/scaling layer after fuzzy inference subsystem was useful to limit the control outputs to a bounded domain range, and reduce the search space for reinforcement learning.

A backpropagation-like gradient supervised learning algorithm was formulated for the differentiable AFN to modify the internal network parameters. It was observed that using individual learning rates for different parameters was essential for the gradient learning to converge quickly.The layered learning rates for AFN were derived using unit-analysis

method. The adaptive function approximation ability of the AFN was demonstrated in a nonlinear function test. The function approximation and generalization capacity of AFN was essential for scaling up learning controllers to real world problems, such as the control of neural prostheses.

For reinforcement learning, two AFN were employed, the Evaluation Network and Action Network respectively, and a Stochastic Search Unit was inserted between the fuzzy inference subsystem and the output squashing/scaling layer in the Action Network for active exploration. The relationships between Temporal Difference (TD(λ)) method, stochastic gradient following REINFORCE algorithm, and stochastic Dynamic Programming are clearly analyzed from different perspectives. A reinforcement learning algorithm integrating $TD(\lambda)$ method and stochastic gradient following REINFORCE was formulated for the AFN. The use of large decaying constant in the eligibility traces to cope with unknown time delay in control system is analyzed. It is also pointed out that using the evaluation of previous state instead of the evaluation of previous action [HB2b] to determine the stochastic search scope led to better exploration, and consequently resulted in faster overall convergence of reinforcement learning.

The thesis objective No.2 has been achieved by the development of AFN with supervised and reinforcement learning mechanisms.

The AFN was successfully applied to the finite state control of a hybrid neural prosthesis with power hip brace and quadriceps muscle stimulation. The plant was a swinging leg simulated by a compound pendulum and a three-factor (activation/angle/angular-velocity dependence) quadriceps muscle model with neural time-delay and nonlinear recruitment curve. The control inputs to the plant were hip torque supplied by power hip brace and functional electrical stimulation delivered to activate quadriceps muscle. The sensory feedback signals were hip angle and knee angle. The three optimal control objectives for successful swing phase were: foot clearance, a certain hip angle range, and knee extension at the end of swing phase. An open-loop controller was obtained by parameterizing control inputs and using SIMPLEX algorithm to optimize the control parameters to satisfy all the three swing phase objectives.

Through computer simulations, it was found that supervised learning AFN controller rapidly mimicked the optimal control strategy from the previously optimized open-loop

controller in approximately 10-20 training trials. Furthermore, the learning controller was able to generalize the mimicked control strategy to cope with slight parameter variations, while open-loop controller failed. This was attributed to the use of closed-loop sensor feedback and generalization capability of AFN, and demonstrated the advantages of closed-loop finite state control over open-loop control. Although the optimal control strategy was mimicked from open-loop controller, supervised learning controller was able to generalize between similar situations and actions.

Reinforcement learning AFN controller was able to synthesize a similar optimal control strategy in approximately 100-200 training trials. The controller learned the optimal control strategy through stochastic dynamic programming by using evaluative reinforcements from environment, rather than mimicking the instructive teaching signals through supervised learning from the instructive teacher. The reinforcements were delayed evaluative signals, i.e. scalar signals indicating whether swing was successful or failure according to three swing phase objectives. Reinforcement learning controller was also able to generalize the learned control strategy to cope with slight parameter variations.

The convergence to a optimal control strategy in reinforcement learning was slower than that in supervised learning by approximately an order of magnitude in this particular case. The difference in the convergence rate was due to the different quality of teaching signals, which was instructive and immediate in supervised learning, but only evaluative and delayed in reinforcement learning. When instructive teaching signals are available, supervised learning should be used. But reinforcement learning provides a solution when such knowledgeable teachers are not available.

Furthermore, it was demonstrated that using a control rule base previously learned (via supervised or reinforcement learning) from one model as starting point, reinforcement learning controller could learn a new optimal control strategy for another new model with different parameters in approximately tens of training trials (mostly in 10-20 trials). Incorporating a priori knowledge significantly (mostly by an order of magnitude) reduced the training trials for reinforcement learning controller. This combined reinforcement learning with a priori knowledge or supervised learning experience method is, thus, proposed for further clinical trial.

Thus, thesis objectives No.3 and No.4 have been achieved through computer simulation

study. Supervised learning and reinforcement learning could be used with previous learning experience from similar models or hand-crafted rules from human experts. Adaptive Fuzzy Network (AFN) provides an unified structure to incorporate all these complementary elements. To the author's knowledge, this is the first demonstration of feasibility of such techniques in neural prosthetic control.

Chapter 6

Suggestion for Future Work

Current work could be extended by further algorithmic development and by applying simulation results to real-time systems.

Further algorithmic development could include structure determination and utilizing more informative reinforcement signals.

Adaptive Fuzzy Network could be enhanced by incorporating inductive machine learning (e.g. EMPIRIC) for structure determination [TST92, Jan94]. Hyperplanar partitioning of the attribute space by decision trees could relax "curse of dimensionality", a problem commonly associated with fuzzy systems using grid partitioning. This topic could also be further developed into fuzzy system identification technique, like the conventional system identification [Eyk74] in numerical control field. Recently, ID3 has been extended to the continuous ID3 (CID3) [CL92] which performs the partitioning of attribute space using continuous hyperplanes instead of those perpendicular to the attribute axes. CID3 has been used for optimal architecture generation of neural networks. One weakness of ID3 type algorithms is that they are supervised learning algorithms requiring instructive training examples. The Genetic Algorithms [BGH89] belong to reinforcement learning family using performance index for optimization and have been used for both structure and parameter optimization of fuzzy systems [LT93, HM95]. The problem with Genetic Algorithms is that they are difficult to be incorporated with connectionist neural network due to their complex encoding/decoding schemes and not utilizing known network structure to reduce search space. Other unsupervised learning or clustering algorithms such as Kohonen's SOM [Koh88] could also be used for fuzzy rule extraction.

Utilizing more informative reinforcement signals could speed up reinforcement learn-

ing. One possible informative reinforcement signals are reference trajectories. Reference trajectory following algorithm has been used in [Yuh94, CS93, CO93] for training of neural networks or fuzzy system. In conventional adaptive control, there is also Model Reference Adaptive Control (MRAC). At early stage, it is just simple MIT heuristic rule (known as the MIT-rule in adaptive control literature). In adaptive fuzzy logic control, there are similar Model Reference Adaptive Fuzzy Logic Controller (MRAFLC), as reviewed in literature review chapter. But MRAC and MRAFLC are rather complex algorithm using Lyapunov method. MIT heuristic rule is simple, yet, could have very good results in some particular situations if designed properly using expert knowledge. Some preliminary simulations done in the swinging leg model (not reported in this thesis) showed very encouraging result. The heuristic MIT rule used following δu as controller training error:

$$\delta u = k1 \cdot (y_d - y) + k2 \cdot (v_d - v) \tag{6.1}$$

where y_d, v_d were desired angle and angular velocity, y, v were actual angle and angular velocity, and $k1$,$k2$ were two constants properly decided by considering the ratios between control signals and trajectory signals. Hip and knee joint controllers only took care of the hip and knee joint, respectively. The learning controller using this training error learned all the three swing phase objectives. This result suggested that simple heuristic MIT rule (Eq.6.1) worked quite well, at least for the swing leg model. Actually Eq.6.1 is just the output of a simple PD controller. If we combined the outputs of this PD controller and AFN controller, we actually get the Kawato's feedback-error-learning system [KFS87, KUIS88, Kaw90, GK90, KG91, KG92], as reviewed in literature review chapter.

Simulation results in this thesis could be applied to real-time systems for prolonged standing control and swing phase control.

A simple application is prolonged paraplegic standing using Floor Reaction Orthosis(FRO) and quadriceps stimulation. As reviewed early, Modular hybrid proposed by Andrews et al. [ABB+88, ABPK89, KAM+93] using FRO to provide stability, without FNS activation of muscles, for 'C' standing postures. Stability is maintained so long as the ground reaction vector remains anterior to the knee joint axis. When ground reaction vector pass through or behind the knee axis, quadriceps muscles are stimulated to prevent knee buckling. The knee buckling controller is a simple on/off finite state controller [ABB+88, ABPK89, MVBZ92]. One problem with this on/off controller is that a strong

quadriceps stimulation is usually delivered to activate a hyperextension of knee joint. While this assures the safety, it could be harmful to the knee joint if hyperextension is activated frequently. Furthermore, from two-dimensional phase-plane analysis of knee-buckling and recovery process, it is shown that at the end of recovery phase, the knee angular velocity is not near zero. The angular velocity is abruptly forced to the zero when knee joint reaches its limited position. This may cause some impact on the knee joint. An ideal buckling recovery should be a gentle process with angular velocity gradually decreases to zero as knee angle approaches its limit position. This could be automatically learned by a reinforcement learning controller. Two optimal objectives in this knee-buckling control are: knee angle remains smaller than a certain threshold, and knee angular velocity is near zero when knee angle reaches its limit position. When knee angle exceeds the threshold, the on/off controller is triggered to assure the subject's safety, and a penalty reinforcement signal is sent to reinforcement learning controller. Gradually, reinforcement learning controller should be able to learn to increase quadriceps stimulation to prevent knee buckle. Another reinforcement signal is proportional to the knee angular velocity at the end of recovery phase. Reinforcement learning controller should be able to decrease quadriceps stimulation in advance to prevent high angular velocity at the end of recovery phase. Then, we can get a controller with better performance than simple on/off controller, which could prevent knee buckling and high recovery angular velocity. It is also possible to use supervised learning first to learn the control surface from simple on/off controller, and then using reinforcement learning controller to fine-tune the learned rules. It is also straightforward to hand-craft the on/off control rules directly. Goniometers, force sensing resistors [ABB+88, ABPK89], and accelerometers [VFVB93, VBK+93] could be used as feedback sensors for knee stability detection. But goniometers should be used at least in training phase to detect the knee angle and angular velocity. After training, goniometer could be unmounted.

Another real-time application could be the parameterized swing phase control, like that proposed by Franken and Veltink et al.[FV95]. Franken's parameterized swing used the similar swing phase objectives as our simulated swing. The difference is that they used hip flexor for hip flexion and hamstrings for knee flexion, while only powered hip brace was used in our simulated swinging leg to flex both hip and knee. Since the power hip brace modeled in our computer simulation is not available yet, Franken's approach is more

realistic. Another alternative is to use flexion reflex [GHN+93] for hip and knee flexion. A similar stand-swing supporting frame with adjustable bicycle saddle was also set up in the Research Center at Glenrose Hospital. A reinforcement learning controller for swing phase control could be tested on this experimental setup. It should be emphasized that although joint angular signals were used in our computer simulations, other sensors such as accelerometers and inclinometers could also be used in the finite state control. No rigorous dynamic model was required for using this finite state controller. Finite state control based on pattern recognition requires richness in sensor information in order to make the right actions, while conventional numerical control based on dynamic model requires accuracy in sensor information in order to calculate the control signals from tracking error to robustly drive the system dynamics. In biological motor systems, information richness is easily guaranteed, but accuracy is not very high. In FNS systems, information richness is not easily guaranteed, considering the practical limitation on number of sensors mounted on human subjects. However, accuracy is more difficult to obtain, considering the limited choice of practical sensors and difficulties in sensor mounting, cabling and calibration. It is recommended that joint angular signals (from goniometers) could be used in the training phase as reference trajectories to train the reinforcement learning controller. Reference joint angular trajectories could be adopted from GAITLAB data [VDO92]. Signals form practical sensors, including accelerometers and inclinometers, are used as actual state feedback to the controller, while signals from goniometers are used only as informative reinforcement signals. Simple MIT rule Eq.6.1 could be used to train the reinforcement learning controller. Once training is completed, mechanical goniometers that are not very convenient and enduring can be unmounted. Only practical sensors remain to provide state feedback for the finite state controller. Recent reports [VFVB93, VBK+93, And95] suggested that accelerometers could provide rich information for gait event discrimination.

A real-time PC-based experimental control system and a mechanical supporting frame had been developed in Research Center at Glenrose. Real-time performance of the finite state learning controllers should be evaluated in the laboratory using human SCI subjects.

Appendix A

Dynamic Equations of the Swinging Leg Model

The biomechanical model of the swinging leg used in this study was adopted from [Bie93]. It was improved in this study by adding a nonlinear muscle recruitment curve. Also, hip torque is assumed to be controllable at any moment, rather than a parameterized waveform function. Two scalar gain factors were added for simulating muscle fatigue and potential.

The passive dynamics of the leg was modeled as a freely swinging compound pendulum with two degrees of freedom (hip and knee joints in the sagittal plane) and adequate damping and elasticity in the joints (Fig.A.1). The ankle joint is assumed to be fixed at 90 degrees by the floor reaction orthosis. No standing leg is considered. The hip was in a fixed position, because simulations showed that the influence of one inch hip motion on the dynamics of the leg was negligible, yet including hip motion demanded extra differential equations.

The compound pendulum has certain masses and inertia according to the limb mass of the person. There is a linear damping to account for friction in the joints, and a global elasticity representing muscles and ligaments. The global elasticities of ligaments are described with exponential functions. The global elasticity of the muscles is simplified into a linear relation. The torque on the joints due to the elasticity is:

$$M_{E_i} = k_1 e^{-k_2(\theta_1 - \theta_i)} - k_3 e^{-k_4(\theta_i + \theta_2)} + k_5(\theta_i - \theta_3)$$

i=1,2 (knee, hip) $\theta_{1,2,3}$ = offset constants $k_{1...5}$ = shape/slope factors

The complete equation (A.1) , describing the dynamics behavior of the whole leg can

be written as:

$$\ddot{o}(t) = J^{-1}(M - G\dot{o}(t) - D\dot{o}(t) - E - g) \tag{A.1}$$

where:

$$o(t) = Joint\ angle\ vector = \begin{bmatrix} o_h(t) \\ o_k(t) \end{bmatrix} = \begin{bmatrix} \text{Hip Angle} \\ \text{Knee Angle} \end{bmatrix}$$

$$J = Inertia\ matrix = \begin{bmatrix} jm_{11} & jm_{12} \\ jm_{21} & jm_{22} \end{bmatrix}$$

$$M = Input\ torque\ vector = \begin{bmatrix} M_h \\ M_k \end{bmatrix} = \begin{bmatrix} \text{Hip Torque provided by powered brace} \\ \text{Knee Torque provided by quadriceps} \end{bmatrix}$$

$$G = Coriolis\ and\ centripetal\ force\ matrix = \begin{bmatrix} gyr_{11} & gyr_{12} \\ gyr_{21} & gyr_{22} \end{bmatrix}$$

$$D = Damping\ vector = \begin{bmatrix} D_h \\ D_k \end{bmatrix} = \begin{bmatrix} \text{Damping in hip joint} \\ \text{Damping in knee joint} \end{bmatrix}$$

$$E = Elasticityvector = \begin{bmatrix} E_h \\ E_k \end{bmatrix} = \begin{bmatrix} \text{Torque caused by elasticity around hip joint} \\ \text{Torque caused by elasticity around knee joint} \end{bmatrix}$$

$$g = Gravitationalforcevector = \begin{bmatrix} g_h \\ g_k \end{bmatrix} = \begin{bmatrix} \text{Gravitational force in hip joint} \\ \text{Gravitational force in knee joint} \end{bmatrix}$$

For more details of the dynamic equations, please consult the technical report [Bie93].

The initial conditions of the dynamic system are adopted from GAITLAB data: Hip angular velocity is 1 rad./sec, knee angular velocity is -1.7 rad./sec, hip angle is -.12 rad., and knee angle is -0.7 rad. body mass=55m, and body height=1.65kg.

The active quadriceps muscle is modeled as a three factor nonlinear dynamic model with time delay and nonlinear recruitment curve (Fig.A.2). The three-factor muscle model [VCCeB92] includes activation dependency , angle dependency, and angular velocity dependency. Input is FNS pulsewidth, which is transferred into normalized activation Sq by nonlinear recruitment curve. The recruitment curve has three typical regions: deadzone, high-slope, and saturation [DM89], and is modeled by a sigmoid function.

The following MATLAB program is used to calculate biomechanical parameters from the body mass m0 and body height l0. See Fig.(A.1) and Eq.(A.1) for labels. The parameters of quadriceps muscles are also given in what follows.

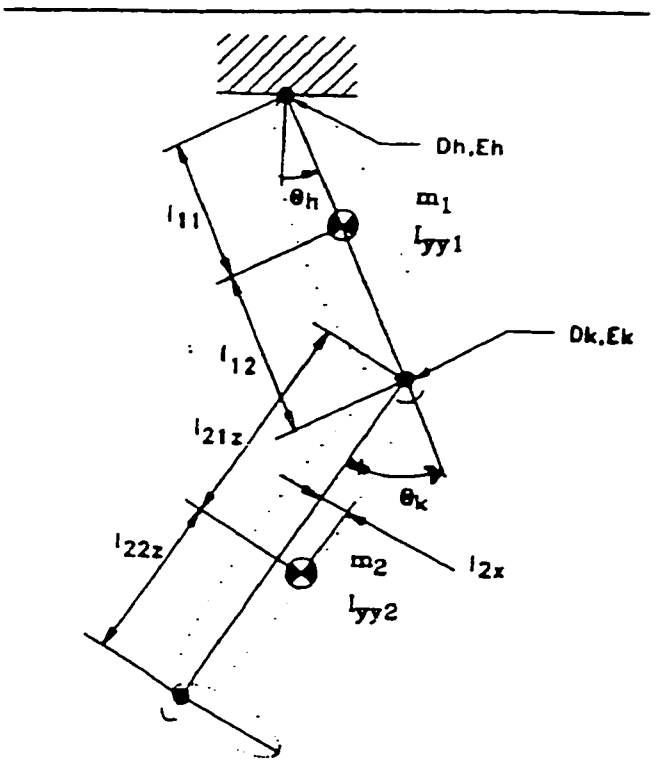


Figure A.1: The leg is modeled as a compound pendulum with mass, damping and elasticities on the joints.

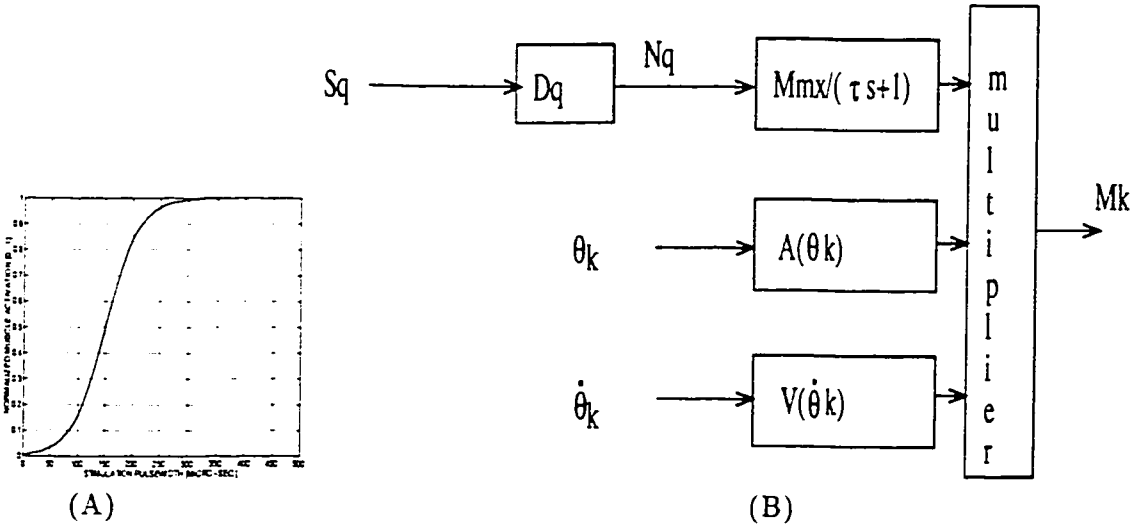


Figure A.2: Quadriceps muscle model: (A) Nonlinear sigmoid recruitment curve, the input is FNS pulsewidth, the output is normalized muscle activation; (B) Three factor muscle dynamic model. Sq is the input normalized muscle activation; Dq is the time delay; Mmx is the maximum torque; Mk is the output torque applied on knee joint. $Mmx/(\tau s + 1)$ is a first-order linear activation dynamics; $A(\theta_k)$ is the angle dependency; $V(\dot{\theta}_k)$ is the angular velocity dependency.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Approximates model constants for normal person
% Input: Body mass=m0 [kg], height=l0 [m]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mass and inertia parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% upper/lower leg mass (kg) , gravity (m/s^2):
 m1 = .1*m0; m2 = .061*m0;  g = 9.81;
% Geometry (m), Inertia (kgm^2):
 l11 = -.433*.245*l0; l12 = .567*.245*l0;
 l21z = -.606*.246*l0; l22z = .136*l0; l2x = -.009*l0;
 hlx=-.25*.152*l0-.01; flx=.75*.152*l0+.01;
 Iyy1=m1*(.323*(l12-l11))^2;Iyy2=m2*(.416*.246*l0)^2;
% Elasticity parameters (no scaling formulas known):
 k1h = 8.7; k2h = 1.3; k3h = 2.6; k4h = 5.8; k5h = .0;
 ph1h = 1.92; ph2h = -.52; ph3h = 0;
%  !!! no valid data on k5h and ph3h !!
 k1k = 10.5; k2k = 11.8; k3k =  3.1; k4k = 5.9; k5k = 3.0;
 ph1k = .1; ph2k = -1.92; ph3k = -1.5;
% Damping
 Damph=1.06; Dampk=.16;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quadriceps muscles parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Maximum torque, depending on the body mass
 Mmx=(m0-55)*0.10+15;
% Parameters for angle dependency
 phimx=-.87; k6=1.5; k7=.2; phioff=-1.3;
% Parameters for angular velocity dependency
 velmx = -12; k8 = .35;
% Time constants for the muscle activation dynamics
 tau3on=.15; tau3of=.05;
% neural delay constant [second]
 dtime=.06;
% quadriceps muscle recruitment curve: parameters for sigmoid function
 quad_sa=30; quad_sc=150;quad_yr=1;quad_yc=0.5;
```

Bibliography

[ABB+88] B.J. Andrews, R.H. Baxendale, R. Barnett, G.F. Phillips, T. Yamazaki, J.P. Paul, and P.A. Freeman. Hybrid FES orthosis incorporating closed loop control and sensory feedback. *Journal of Biomedical Engineering*, 10(2):189–195, April 1988.

[ABPK89] B. J. Andrews, R. W. Barnett, G. F. Phillips, and C. A. Kirkwood. Rule-based control of a hybrid FES orthosis for assisting paraplegic locomotion. *Automedica*, 1989.

[AC92] J.J. Abbas and H.J. Chizeck. Adaptive feedforward control of cyclic movements using artificial neural networks. In *Proc. of IEEE-INNS Int. Joint. Conf. on Neural Networks*, pages II832–II837, Baltimore, 1992.

[AI86] J. Allin and G.F. Inbar. Fns control schemes for the upper limb. *IEEE Trans. on Biomedical Engineering*, 33(9):818–828, September 1986.

[Alb81] James S. Albus. *Brains, Behavior, and Robotics*. BYTE Publications, Peterborough, NH, USA, 1981.

[And89] C.W. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, pages 31–37, April 1989.

[And95] B.J. Andrews. On the use of sensors for FES control. In *Proc. of 5th Vienna Int. Workshop onOB Functional Electrostimulation*, pages 263–266, Vienna, Austria, 1995.

[AT93] J.J. Abbas and R.J. Triolo. Experimental evaluation of an adaptive feedforward controller for use in functional neuromuscular stimulation systems. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*. pages 1326–1327. 1993.

[Ath93] A. Athalye. On designing a fuzzy control system using an optimization algorithm. *Fuzzy Sets and Systems*. 56:281–290. 1993.

[AW95] K.J. Astrom and B. Wittenmark. *Adaptive control*. Addison-Wesley. Reading. MA. USA. 2nd edition. 1995.

[BA85] A.G. Barto and P. Anandan. Pattern-recognizing stochastic learning automata. *IEEE Trans. on Systems. Man, and Cybernetics*. 15:360–375. 1985.

[BBBW81] L.A. Benton. L.L. Baker. B.B. Bowman. and R.L. Waters. *Functional Electrical Stimulation-A practical clinic guide*. Rancho Los Amigos Hospital. Downey. California. 1981.

[BBC+95] R.P. Bonissone. V. Badami. K.H. Chiang. P.S. Khedkar. and K.W. Industrial applications of fuzzy logic at General Electric. *Proc. of IEEE*. 83(3):450–465. March 1995.

[BBS93] A. G. Barto. S. J. Bradtke. and S. P. Singh. Learning to act using real-time dynamic programming. *AI Journal*. 1993.

[BC95] H.R. Beom and H.S. Cho. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *IEEE Trans. on Systems. Man. and Cybernetics*. 25(3):464–477. March 1995.

[BCC87] L.A. Bernotas. P.E. Crago. and H. Chizeck. Adaptive control of electrically stimulated muscle. *IEEE Trans. on Biomedical Engineering*. 34(2):140–147. February 1987.

[Bee90] R. D. Beer. *Intelligence as adaptive behavior: an experiment in computational neuroethology*. Academic Press. Boston. 1990. (A computer simulation software of a simplified cockroach based on Beer's book is available from ftp://oak.oakland.edu/SimTel/msdos/neurlnet/nerves.zip).

[Bel57] Richard Bellman. *Dynamic programming.* Princeton University Press, Princeton, N.J., 1957.

[Ber92] H.R. Berenji. An architecture for designing fuzzy controllers using neural networks. *Int. J. of Approximation Reasoning*, 6(2):267–292, February 1992.

[Ber93] H.R. Berenji. Adaptive fuzzy control with reinforcement learning. In *Proc. of American Control Conference*, pages 1840–1844, San Francisco, CA, 1993.

[BFK89] G. Borges, K. Ferguson, and R. Kobetic. Development and operation of portable and laboratory electrical stimulation systems for walking in paraplegic subjects. *IEEE Trans. on Biomedical Engineering*, 36(7):798–800, July 1989.

[BGH89] L.B. Booker, D.E. Goldberg, and J.H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[BH93] J.J. Buckley and Y. Hayashi. Numerical relationships between neural networks, continuous functions, and fuzzy systems. *Fuzzy Sets and Systems*, 60:1–8, 1993.

[Bie93] Jeroen Bielen. The feasibility of a powered brace for paraplegic gait-preliminary results of computer simulation study. Technical report, Department of Applied Sciences in Medicine, University of Alberta, Edmonton, Canada, April 1993.

[BLJ+93] H.R. Berenji, R.N. Lea, Y. Jani, P. Khedkar, A. Malkani, and J. Hoblit. Space shuttle attitude control by reinforcement learning and fuzzy logic. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, volume 2, pages 1396–1401, 1993.

[BP94] A. Del Boca and D.C. Park. Inductive cloning of a state model for normal gait. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, pages 918–919, 1994.

[BR94] H. Barbeau and S. Rossignol. Enhancement of locomotor recovery following spinal cord injury. *Current Opinion in Neurology*, 7:517–524, 1994.

[Bra93] S. J. Bradtke. Reinforcement learning applied to linear quadratic regulation. In *Advances in Neural Information Processing Systems (NIPS)*. San Mateo, CA, 1993. Morgan Kaufmann. (ftp://ftp.gmd.de/Learning/rl/bradtke.nips5.ps.Z).

[BRM93] R.D. Beer, R.E. Ritzmann, and T. McKenna. *Biological neural networks in invertebrate neuroethology and robotics*. Academic Press, Boston, 1993.

[Bro75] S. M. Brooks. *Basic science and the human body-anatomy and physiology*. Mosby, Saint Louis, 1975.

[Bro86] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Robot. Autom.*, 2:14–23, 1986.

[Bru93] J. Bruske. Neural fuzzy decision systems. Diploma thesis, Fachbereich Informatik, Univ. Kaiserslautern, Germany, 1993. (available from ftp://ag_vp_file_server.informatik.uni-kl.de/Public/Neural_Networks/Reports/Bruske.Diploma-thesis).

[BS92] Allen Bonde and R. Sutton. Nonlinear TD/backprop pseudo c-code. Technical report, GTE Laboratories Inc., April 1992. (available from ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/td-backprop-psuedo-code).

[BSA83] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics*, 13:834–846, 1983.

[BSB93] N.E. Berthier, S.P. Singh, and A.G. Barto. Distributed representation of limb motor programs in arrays of adjustable pattern generators. *Journal of cognitive neuroscience*, 5(1):56–78, Wint 1993.

[BSW90] A. G. Barto, R.S. Sutton, and C.J.C.H. Watkins. Learning and sequential decision making. In M. Gabriel and J.W. Moore, editors, *Learning and computational neuroscience*, pages 539–602. MIT Press, 1990. (PS file available from ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/barto-sutton-watkins-90.ps).

[Cas95] J.L. Castro. Fuzzy logic controllers are universal approximators. *IEEE Trans. on Systems, Man, and Cybernetics*, 25(4):629–635, April 1995.

[CCG91] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithms for radial basis function networks. *IEEE Trans. on Neural Networks*, 2(2):302–309, March 1991.

[CCM91] S. Chiu, S. Chand, and D. Moore. Fuzzy logic for control of roll and moment for a flexible wing aircraft. *IEEECSM*, pages 42–48, June 1991.

[CCNH86] P.E. Crago, H.J. Chizeck, M.R. Neuman, and F.T. Hambrecht. Sensors for use with functional neuromuscular stimulation. *IEEE Trans. on Biomedical Engineering*, 33(1):256–267, 1986.

[Chi92] Howard J. Chizeck. Adaptive and nonlinear control methods for neural prostheses. In Richard B. Stein, P. Hunter Peckham, and Dejan B. Popovic, editors, *Neural prostheses: replacing motor function after disease or disability*, pages 298–328, New York, 1992. Oxford University Press.

[CKM+88] H. J. Chizeck, R. Kobetic, E.B. Marsolais, J.J. Abbas, I.H. Donner, and E. Simon. Control of functional neuromuscular stimulation systems for standing and locomotion in paraplegics. *Proc. of IEEE*, 76:1155–1165, 1988.

[CL92] K.J. Cios and N. Lan. a machine learning method for generation of a neural network architecture: A continuous ID3 algorithm. *IEEE Trans. on Neural Networks*, 3(2):280–291, March 1992.

[CLH92] Y.Y. Chen, K.Z. Lin, and S.T. Hsu. A self-learning fuzzy controller. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pages 189–196, San Diego, USA, 1992.

[CLPC91] H.J. Chizeck, N. Lan, L.S. Palmieri, and P.E. Crago. Feedback control of electrically stimulated muscle using simultaneous pulse width and stimulus period modulation. *IEEE Trans. on Biomedical Engineering*, 38(12):1224–1234, December 1991.

[CO93] B. Chung and J. Oh. Control of dynamic systems using fuzzy learning algorithm. *Fuzzy Sets and Systems*, pages 1–14. 1993.

[CPN94] CPN. *Cure Paralysis Now World Wide Web (WWW) Page*. InfoWest, http://www.infowest.com/cpn/index.html. 1994.

[Cro88] T.B. Cross. *Knowledge Engineering*. Brady, New York, USA, 1988.

[CS93] X. Cui and K.G. Shin. Direct control and coordination using neural network. *IEEE Trans. on Systems, Man, and Cybernetics*. 23(3):686–697. 1993.

[DH90] William K. Durfee and J.M. Hausdorff. Regulating knee joint position by combining electrical stimulation with a controllable friction brake. *Annals of Biomedical Engineering*. 18(6):575–96. 1990.

[dKSG94] L. delRe, F. Kraus, J. Schultheiss, and H. Gerber. Self-tuning PID controller for lower limb FES with nonlinear compensation. In *Proc. of American Control Conference*, pages 2015–2019, 1994.

[DM89] W.K. Durfee and K.E. MacLean. Methods for estimating isometric recruitment curve of electrically stimulated muscle. *IEEE Trans. on Biomedical Engineering*. 36(7):654–667. July 1989.

[Don87] Marc D. Donner. *Real-time control of walking*. Birkhauser, Boston, 1987.

[DS94] P. Dayan and T.J. Sejnowski. Td(λ) converges with probability 1. *Machine Learning*. 14(3):295–301, 1994.

[Dur92] William K. Durfee. Design of a controlled-brake orthosis for regulating FES-aided gait. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, pages 1337–1338, 1992.

[Erg85] Z. Ergas. Spinal cord injury in the United States: a statistical update. *Central Nervous System Trauma*, 2:19–32, 1985.

[EWG+91] T. Eilin, M. Wild, H. Gerber, E. Stussi, and S. Hacisalihzade. An eight-channel computer controlled stimulator for functional electrical stimulation. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 13, pages part 4: 1802–1804, 1991.

[Eyk74] P. Eykhoff. *System identification : parameter and state estimation.* Wiley-Interscience, London, 1974.

[Faw92] J.W. Fawcett. Spinal cord repair: future directions. *Paraplegia*, 30(2):83–85, February 1992.

[FC93] Z.P. Fang and T.J. Crish. Sensor signal telemetry for functional electrical stimulation. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 15, pages part3: 1244–1245, 1993.

[Fra89] J.A. Franklin. Historical perspective and state of the art in connectionist learning control. In *Proc. of IEEE Conf. on Decision and Control*, volume 28, pages 1730–1736, 1989.

[FV95] Henry M. Franken and P.H. Veltink. Cycle-to-cycle control of the swing phase of paraplegic gait induced by surface electrical stimulation. *Medical & Biological Engineering & Computing*, 33:440–451, 1995.

[GFAD93] M.H. Granat, A.C.B. Ferguson, B.J. Andrews, and M. Delargy. The role of functional electrical stimulation in the rehabilitation of patients with incomplete spinal cord injury—observed benefits during gait studies. *Paraplegia*, 31:207–215, 1993.

[GFB94] V. Gullapalli, J. Franklin, and H. Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine*, pages 13–24, February 1994.

[GHN+93] M.H. Granat, B.W. Heller, D.J. Nicol, R.H. Baxendale, and B.J. Andrews. Improving limb flexion in FES gait using the flexion withdrawal response for the spinal cord injured person. *Journal of Biomedical Engineering*, 15:51–56, January 1993.

[GK90] H. Gomi and M. Kawato. Learning control for a closed loop system using feedback-error-learning. In *Proc. of IEEE Conf. on Decision and Control* volume 29, pages 3289–3294, 1990.

[GK95] D. Graupe and H. Kordylewski. Artificial neural network control of FES in paraplegics for patient responsive ambulation. *IEEE Trans. on Biomedical Engineering*. 42(7):699–707. July 1995.

[GP91] M. Gauthier and A. Prochazka. Microprocessor control strategy for functional electrical stimulation to improve hemiparetic gait. *Annals of Biomedical Engineering*. 19(5):627. 1991.

[Gul90] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*. 3:671–692. 1990.

[HB81] C.J. Harris and S.A. Billings. *Self-Tuning and adaptive control: theory and applications*. volume 15 of *IEE control engineering series*. Peter Peregrinus Ltd., Stevenage UK and New York. 1981.

[HB2b] P. Khedkar H.R. Berenji. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. on Neural Networks*. 3(5):724–740. September 1992b.

[HD91] J.M. Hausdorff and W.K. Durfee. Open-loop position control of the knee joint using electrical stimulation of the quadriceps and hamstrings. *Medical & Biological Engineering & Computing*. 29:269–280. 1991.

[Hel92] B. W. Heller. *The production and control of Functional Electrical Stimulation*. PhD thesis. Bioengineering Unit, Univ. of Strathclyde. 1992.

[HH88] W.J. Heetderks and F.T. Harmbrecht. Applied neural control in the 1990s. *Proc. of IEEE*, 76:1115–1121. 1988.

[Hin89] G.E. Hinton. Connectionist learning procedurea. *Artificial Intelligence*. 40:185–234. 1989.

[HKB93] J.C. Houk, J. Keifer, and A.G. Barto. Distributed motor commands in the limb premotor network. *Trends in Neurosciences*. 16(1):27–33, January 1993.

[HLG90] D.A. Handelman, S.H. Lane, and J.J. Gelfand. Integrating neural networks and knowledge-based systems for intelligent robotic control. *IEEE Control Systems Magazine*, pages 77–87, April 1990.

[HLL91] J. He, W.S. Levine, and G.E. Loeb. Feedback gains for correcting small perturbations to standing posture. *IEEE Trans. on Automatic Control*, 36:322–332, 1991.

[HM94] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. on Neural Networks*, 5(6):989–993, November 1994.

[HM95] A. Homaifar and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controllers using Genetic Algorithms. *IEEE Trans. on Fuzzy Systems*, 3(2):129–139, May 1995.

[HP94] H. Hellendoorn and R. Palm. Fuzzy system technologies at Siemens R & D. *Fuzzy Sets and Systems*, 63:245–269, 1994.

[HSZG92] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop. Neural networks for control systems-a survey. *Automatica*, 28(6):1083–1112, 1992.

[HVR+93] B.W. Heller, P.H. Veltink, N.J. Rijkhoff, W.L. Rutten, and B.J. Andrews. Reconstructing muscle activation during normal walking: a comparison of symbolic and connectionist machine learning techniques. *Biological Cybernetics*, 69(4):327–335, 1993.

[HY92] H.Takahashi and Y.Handa. Predication of the intended movements by analysis of neural activity in the monkey motor areas-studies on possible sources of FES control commands. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 14, pages 1461–1462, 1992.

[IK94] I.H.Suh and T.W. Kim. Fuzzy membership function based neural networks with applications to the visual servoing of robot manipulators. *IEEE Trans. on Fuzzy Systems*, 2(3):203–220, August 1994.

[Isi85] A. Isidori. *Nonlinear control systems: an introduction.* Springer-Verlag, New York, 1985.

[Ito84] Masao Ito. *The cerebellum and neural control.* Raven Press, New York, 1984.

[IVP94] M. Ilic, D. Vasiljevic, and D. Popovic. A programmable electric stimulator for FES system. *IEEE Trans. on Rehabilitation Engineering*, 2(4):234–240, December 1994.

[Jan92] J.-S. R. Jang. Self-learning fuzzy controllers based on temporal back propagation. *IEEETNN*, 3(5):714–723, September 1992.

[Jan93] J.-S. Roger Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 23:665–685, 1993.

[Jan94] J.-S. Roger Jang. Structure determination in fuzzy modeling: a fuzzy CART approach. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, 1994.

[JH90] J.R.LaCourse and F.C.C. Hludik. An eye movement communication-cotrol system for the disabled. *IEEE Trans. on Biomedical Engineering*, 37(12):1215–1220, December 1990.

[JJ90] M.I. Jordan and R.A. Jacobs. Learning to control an unstable system with forward modelling. In D.S. Tourztzky, editor, *Advances in Neural Information Processing Systems (NIPS)*, pages 324–331, San Mateo, CA, 1990. Morgan Kaufmann.

[JJ93] R.A. Jacobs and M.I. Jordan. Learning piecewise control strategies in a modular neural network architecture. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(2):337–345, 1993.

[JS93] J.-S. R. Jang and C.T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEETNN*, 4(1):156–158, January 1993.

[JS95] J.R. Jang and C.T. Sun. Neuro-fuzzy modeling and control. *Proc. of IEEE*, 83(3):378–421, March 1995.

[KA88] C.A. Kirkwood and B.J. Andrews. Rule-based control for FES using firmware transitional logic. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 10, pages 1562–1563, 1988.

[KA89] C.A. Kirkwood and B.J. Andrews. Finite state control of FES systems: application of AI inductive learning techniques. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 11, pages 1020–1021, 1989.

[KAM89] C.A. Kirkwood, B.J. Andrews, and P. Mowforth. Automatic detection of gait events: a case study using inductive learning techniques. *Journal of Biomedical Engineering*, 11:511–516, November 1989.

[KAM+93] C. Kantor, B.J. Andrews, E.B. Marsolais, M. Solomonow, R.D. Lew, and K.T. Ragnarsson. Report on a conference on motor prostheses for workplace mobility of paraplegic patients in north america. *Paraplegia*, 31(7):439–56, July 1993.

[KAP+95] A. Kostov, B.J. Andrews, D.B. Popovic, R.B. Stein, and W.W. Armstrong. Machine learning in control of functional electrical stimulation systems. *IEEE Trans. on Biomedical Engineering*, 42(6):541–551, June 1995.

[Kaw90] M. Kawato. The feedback-error-learning neural network for supervised motor learning. In R. Eckmiller, editor, *Advanced neural computers: Proc. of int. symp. on neural network for sensory and motor systems*, pages 365–372, Amsterdam, 1990. Elsevier.

[KC90] L.G. Kraft and D.P. Campagna. A comparison between CMAC neural network and two traditional adaptive control systems. *IEEE Control Systems Magazine*, pages 36–43, April 1990.

[KFS87] M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169–185, 1987.

[KG91] M. Kawato and H. Gomi. Model of four regions of the cerebellum. In *Proc. of IEEE-INNS Int. Joint. Conf. on Neural Networks*, pages 410–419, Singapore, 1991.

[KG92] M. Kawato and H. Gomi. The cerebellum and VOR/OKR learning models. *Trends in Neurosciences*, 15(11):445–453, 1992.

[Kir70] Donald E. Kirk. *Optimal control theory: an introduction.* Prentice-Hall, Englewood Cliffs, N.J.,, 1970.

[Klo74] A.H. Klopf. Brain function and adaptive systems – a heterostatic theory. *IEEE Trans. on Systems, Man, and Cybernetics*, 1974.

[KM94] R. Kobetic and E.B. Marsolais. Synthesis of paraplegic gait with multichannel functional neuromuscular stimulation. *IEEE Trans. on Rehabilitation Engineering*, 2(2):66–79, June 1994.

[Kob94] Rudi Kobetic. Advancing step by step. *IEEE Spectrum*, pages 27–31, October 1994.

[Koh88] T. Kohonen. *Self-organization and associative memory.* Springer-Verlag, berlin, 2nd edition, 1988.

[Kos95] Aleksandar Kostov. *Machine learning techniques for the control of FES-assisted locomotion after spinal cord injury.* PhD thesis, Division of Neuroscience, University of Alberta, Edmonton, Alberta, Canada, Spring 1995.

[KP93] K.L. Kilgore and P.H. Peckham. Grasp synthesis for upper-extremity FNS. *Medical & Biological Engineering & Computing*, 21:604–614, 1993.

[KSAT92] Aleksander Kostov, R.B. Stein, W.W. Armstrong, and M. Thomas. Evaluation of Adaptive Logic Networks for control of walking in paralyzed patients. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, pages 1332–1333, 1992.

[KSM+91] R. Kobetic, P. Samame, P. Miller, G. Borges, and E.B.Marsolais. Analysis of paraplegic gait induced by functional neuromuscular stimulation. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 13, pages 2009–2010, October 1991.

[KUIS88] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki. Hierarchical neural network model for voluntary movement with application to robotics. *IEEE Control Systems Magazine*, pages 8–15, April 1988.

[KZ89] G. Khang and F.E. Zajac. Paraplegic standing controlled by Functional Neuromuscular Stimulation: computer model and control-system design, computer simulation studies. *IEEE Trans. on Biomedical Engineering*, 36(9):873–894, September 1989.

[Lar81] P.M. Larsen. Industrial applications of fuzzy logic control. In E.H. Mamdani and B.R. Gaines, editors, *Fuzzy reasoning and its applications*, pages 335–342. London, 1981. Academic Press.

[LB89] C.C. Lee and H.R. Berenji. Intelligent controller based on approximation reasoning and reinforcement learning. In *Proc. of IEEE Int. Symposium on Intelligent Control*, pages 200–205, Albany, USA, 1989.

[LCC90] N. Lan, P.E. Crago, and H.J. Chizeck. A perturbation control strategy for FNS motor prostheses. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 12, pages 2327–2328, 1990.

[LCC91] N. Lan, P.E. Crago, and H.J. Chizeck. Feedback control methods for task regulation by electrical stimulation of muscles. *IEEE Trans. on Biomedical Engineering*, 38(12):1213–1223, December 1991.

[Lee90] C.C. Lee. Fuzzy logic control systems: fuzzy logic controller. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2):404–435, 1990.

[LFC94] N. Lan, H.Q. Feng, and E. Crago. Neural network generation of muscle stimulation for control of arm movements. *IEEE Trans. on Rehabilitation Engineering*, 2(4):213–223, December 1994.

[LHSD61] W.T. Liberson, H.J. Holmquest, D. Scott, and M. Dow. Functional electrotherapy, stimulation of the peroneal nerve synchronized with the swing phase of the gait of hemiplegic patients. *Arch. Phys. Med. Rehabil.*, 42:101–105, 1961.

[LL91] C. Lin and C.S.G. Lee. Neural-network-based fuzzy logic control and decision system. *IEEE Trans. on Compyters*, 40(12), December 1991.

[LL94] C. Lin and C.S.G. Lee. Reinforcement structure/parameter learning for neural-network-basd fuzzy logic control systems. *IEEE Trans. on Fuzzy Systems*, 2(1):46–63, February 1994.

[LN93] A.U. Levin and K.S. Narendra. Control of nonlinear dynamical systems using neural networks: controllability and stabilization. *IEEE Trans. on Neural Networks*, 4(2):192–206, March 1993.

[Loe89] G.E. Loeb. Neural prosthetic interfaces with the nervous system. *Trends in Neurosciens*, 12:195–201, 1989.

[LT90a] G. Langari and M. Tomizuka. Stability of fuzzy linguistic control systems. In *IEEECDC*, pages 2185–2190, Honolulu, USA, December 1990.

[LT90b] K. Loparo and E. Teixeira. A new approach for adaptive control of a nonlinear system using neural networks. In *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 38–40, 1990.

[LT93] M.A. Lee and H. Takagi. Integrating design stages of fuzzy systems using genetic algorithm. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pages 612–617, 1993.

[LZST91] G.E. Loeb, C.J. Zamin, J.H. Schulman, and P.R. Troyk. Injectable microstimulator for functional electrical stimulation. *Medical & Biological Engineering & Computing*, 29:NS13–NS19, 1991.

[Mam74] E.H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Proc. IEE*, 121:1585–1588, 1974.

[Mam92] R.J. Mammone. *Computational methods of signal recovery and recognition.* John Wiley and Sons, New York, 1992.

[Mar69] D. Marr. A theory of cerebellar cortex. *J. Physiology*, 202:437–470, 1969.

[Mar91] E.B. Marsolais. Stability of the hip and trunk in paraplegic electrically augmented gait. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 13, pages 2007–2008, October 1991.

[Mat95] Mathworks. *MATLAB Fuzzy Logic Toolbox*. Mathworks, http://www.mathworks.com/fuzzytbx.html, 1995. (this is a WWW page).

[MC72] D. Michie and R.A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence 2*, Edinburgh, 1972. Oliver and Boyd.

[MC92] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(20-3):311–365, June 1992.

[Men95] J.M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proc. of IEEE*, 83(3):345–377, March 1995.

[MF70] J.M. Mendel and K.S. Fu. *Adaptive, learning and pattern recognization systems*. Academic Press, New York, 1970.

[MGK90] W.T. Miller, F.H. Glanz, and L.G. Kraft. CMAC: an associative neural network alternative to backpropagation. *Proc. of IEEE*, 78(10):1561–1567, October 1990.

[MHGK90] W.T. Miller, R.P. Hewes, F.H. Glanz, and L.G. Kraft. Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. *IEEE Trans. Robotics and Automation*, 6(1):1–9, February 1990.

[Mil94] W.T. Miller. Real-time neural network control of a biped walking robot. *IEEE Control Systems Magazine*, pages 41–48, February 1994.

[MM92] K.L. Markey and M.C. Mozer. Comparison of reinforcement algorithms on discrete functions: learnability, time complexity, and scaling. In *Proc. of IEEE-INNS Int. Joint. Conf. on Neural Networks*, pages I853–I859, Baltimore, 1992.

[MNMT89] D. R. McNeal, R.J. Nakai, P. Meadows, and W. Tu. Open-loop control of the freely-swinging paralyzed leg. *IEEE Trans. on Biomedical Engineering*, 36(9):895–905, September 1989.

[MP90] M.W.Johnson and P.H.Peckham. Evaluation of shoulder movement as a command control source. *IEEE Trans. on Biomedical Engineering*, 37(9):189–195, September 1990.

[MSW90a] W. Thomas Miller, Richard S. Sutton, and Paul J. Werbos. An adaptive sensorimotor network inspired by the anatomy and physiology of the cerebellum. In J.C. Houk, S.P. Singh, C. Fisher, and A.G. Barto, editors, *Neural Networks for Control*, pages 301–348, Cambridge, MA, US, 1990. MIT Press.

[MSW90b] W.T. Miller, R.S. Sutton, and P.J. Werbos. *Neural networks for control*. MIT Press, MA, USA, 1990.

[MVB92] A. J. Mulder, P.H. Veltink, and H.B. Boom. On/off control in FES-induced standing up: a model study and experiments. *Medical & Biological Engineering & Computing*, 30:205–212, March 1992.

[MVBZ92] A.J. Mulder, P.H. Veltink, H.B.K Boom, and G. Zilvold. low-level finite state control of knee joint in paraplegic standing. *Journal of Biomedical Engineering*, 14(1):3–8, January 1992.

[NC93] S. K. Ng and H. J. Chizeck. A fuzzy logic gait event detector for FES paraplegic gait. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, pages 1238–1239, San Diego, 1993.

[NHW92] H. Nomura, I. Hayashi, and N. Wakami. A learning method of fuzzy inference rules by descent method. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pages 203–210, 1992.

[NL93] J. Nie and D.A. Linkens. Learning control using fuzzified self-organizing radial basis function network. *IEEE Trans. on Fuzzy Systems*, 1(4):280–287, November 1993.

[NM80] Kumpati S. Narendra and Richard V. Monopoli. *Applications of adaptive control.* International Workshop on Applications of Adaptive Control (1979 : Yale University). Academic Press, New York, 1980.

[NM93] R.E. Nordgren and P. Meckl. An analytical comparison of a neural network and a model-based adaptive controller. *IEEE Trans. on Neural Networks*, 4(4):685–694, July 1993.

[NP90] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4–27, March 1990.

[NT74] K.S. Narendra and M.A.L. Thatachar. Learning automata – a survey. *IEEE Trans. on Systems, Man, and Cybernetics*, 4:323–334, 1974.

[NW90] D.H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, pages 18–23, April 1990.

[Pen92] Jerry Penner. Functional electrical stimulator software maintenance manual. Technical report, Rehab. Tech. Dept., Glenrose Rehabilitation Hostital, Edmonton, Canada, December 1992.

[Per95] T.S Perry. Lotfi A. Zadeh. *IEEE Spectrum*, pages 32–35, June 1995.

[PFA+94] A.G. Parlos, B. Fernandez, A.F. Atiya, J. Muthusami, and W.K. Tsai. An accelerated learning algorithm for multilayer perceptron networks. *IEEE Trans. on Neural Networks*, 5(3):493–497, May 1994.

[PG90] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. of IEEE*, 78(9):1481–1497, September 1990.

[PI91] M.M. Polycarpou and P.A. Ioannou. Identification and control nonlinear systems using neural network models: design and stability analysis. Technical Report EE-Report 91-09-01, USC, 1991.

[PJW92] A.J. Pellionisz, C.C. Jorgensen, and P.J. Werbos. Cerebellar neurocontroller project, for aerospace application. In *Proc. of IEEE-INNS Int. Joint. Conf. on Neural Networks*, pages III379–III384, Baltimore, 1992.

[PM79] T.J Procyk and E.H. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, 1979.

[Pop90] D. Popovic. *Advances in external control of human extremities X*. NAUKA, Belgrade, 1990.

[Pop93] D.B. Popovic. Finite state model of locomotion for functional electrical stimulation. *Progress in Brain Research*, 97:397–407, 1993.

[Pro93] Arthur Prochazka. Comparison of natural and artificial control of movement. *IEEE Trans. on Rehabilitation Engineering*, 1:7–17, 1993.

[PSJ+93] D. Popovic, R.B. Stein, K. Jovanovic, R.C. Dai, A. Kostov, and W.W. Armstrong. Sensory nerve recording for closed-loop control to restore motor functions. *IEEE Trans. on Biomedical Engineering*, 40(10):1024–1031, 1993.

[PSY87] D. Psaltis, A. Sideris, and A. Yamamura. Neural controllers. *Proc. of IEEE Int. Conf. Neural Networks (San Diego)*, 4:551–557, 1987.

[PTS89] D. Popovic, R. Tomovic, and L. Schwirtlich. Hybrid assistive system–the motor neuroprosthesis. *IEEE Trans. on Biomedical Engineering*, 36(7):729–737, 1989.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:129–151, 1986.

[Res90] James Reswick. Moon over dubrovnik–a tale of worldwide impact on persons with disabilities. In Dejan Popovic, editor, *Advances in external control of human extremities X*, pages 1–7, Belgrade, 1990. NAUKA.

[Ros83] S. Ross. *Introduction to stochastic dynamic programming*. Academic Press, New York, 1983.

[RW72] R.A. Rescorla and A.R. Wagner. A theory of pavlovian conditioning: variations in the effctiveness of reinforcement and nonreinforcement. In A.H. Black and W.R. Prokasy, editors, *Classical Conditioning II: Current research and theory*. New York, 1972. Appleton-Century-Crofts.

[Sam59] A.L. Samuel. Some studies in machine learning using the game of checker. *IBM J. Res. Develop.*, 3:210–229, 1959.

[San95] Y. Sankai. A FES controller with artificial CPG for biological systems. In *Proc. of 5th Vienna Int. Workshop onOB Functional Electrostimulation*, pages 267–270. Vienna, Austria, 1995.

[SBW92] R.S. Sutton, A.G. Barto, and R.J. Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, pages 19–22, April 1992.

[SC86] J.E. Slotine and J.A. Coetsee. Adaptive sliding controller synthesis for nonlinear systems. *Int. J. Control*, 43(6):1631–1651, 1986.

[Sch92] A.L. Schwartz. Comments on "Fuzzy logic for control of roll and moment for a flexible wing aircraft". *IEEECSM*, pages 61–63, February 1992. (with author's reply from S. Chiu).

[SFK+92] T. Shibata, T. Fukuda, K. Kosuge, F. Arai, M. Tokita, and T. Mitsuoka. Hierarchical intelligent control for robotic motion by using fuzzy, artificial intelligence, and neural network. In *Proc. of IEEE-INNS Int. Joint. Conf. on Neural Networks*, pages I269–I274, Baltimore, 1992.

[SGW92] S. Srinivasan, R.E. Gander, and H.C. Wood. A movement pattern generator model using artificial neural networks. *IEEE Trans. on Biomedical Engineering*, 39(7):716–722, July 1992.

[SL91] J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[Slo84] J.E. Slotine. Sliding controller design for non-linear systems. *Int. J. Control*, 40(2):421–434, 1984.

[Son93] E.D. Sontag. Some topics in neural networks and control. Technical Report LS93-02, Dept. of Mathematics, Rutgers Univ., Rutgers, NJ, USA, 1993. (ftp://siemens.com/pub/learning/TechReports/Sontag9302.ps.Z).

[SRZG93] L.M. Schutte, M.M. Rodgers, F.E. Zajac, and R.M. Glaser. Improving the efficacy of electrical stimulation-induced leg cycle ergometry. *IEEE Trans. on Rehabilitation Engineering*, 1(2), June 1993.

[SS92a] R.M. Sanner and J.E. Slotine. Enhanced algorithms for adaptive control using radial gaussian networks. In *Proc. of American Control Conference*, pages 2618–2622, 1992.

[SS92b] R.M. Sanner and J.E. Slotine. Gaussian networks for direct adaptive control. *IEEE Trans. on Neural Networks*, 3(6):837–863, November 1992.

[SS94a] R. Sanner and J.E. Slotine. Function approximation, neural networks, and adaptive nonlinear control. In *Proc. 1994 IEEE Conf. on Control Application, Glasgow, UK*, volume 2, pages 1225–1232, 1994.

[SS94b] C.Y. Su and Y. Stepanenko. Adaptive control of a class of nonlinear systems with fuzzy logic. *IEEE Trans. on Fuzzy Systems*, 2(4):285–294, November 1994.

[SS96] S.P. Singh and R.S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, pages 1–37, 1996.

[SSLT92] P. Strojnik, S. Schulman, G. Loeb, and P. Troyk. Multichannel FES system with distributed microstimulator. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 14, pages part 4: 1352–1353, 1992.

[Ste80] R.B. Stein. *Nerve and Muscle-membranes,cells,and systems.* Plenum Press, New York and London, 1980.

[Sut84] R.S. Sutton. *Temporal credit assignment in reinforcement learning.* PhD thesis, University of Massachusetts, 1984.

[Sut87] Richard S. Sutton. Implementation details of the TD(lambda) procedure for the case of vector predication and backpropagation. Technical Report TN87-509.1, GTE laboratories Inc., May 1987. (available by ftp from ftp.cs.umass.edu/pub/anw/pub/sutton/sutton-89.ps.gz).

[Sut88] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Sut95] R. Sutton. Generalization in reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, San Mateo, CA, 1995. Morgan Kaufmann. PS file available from ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/sutton-inprep.ps.

[SW92] D.A. Sofge and D.AA. White. Applied learning optimal control for manufacturing. In D.A. White and D.A. Sofge, editors, *Handbook of intelligent control: neural, fuzzy, and adaptive approaches*, pages 259–282, New York, 1992. Van Nostrand Reinhold.

[SY93] S. P. Singh and R.C Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning Journal (In Press)*, 1993. available from ftp://ftp.cs.umass.edu/pub/anw/pub/yee/approx-rl-loss.ps.Z.

[SZ92] A.W. Salatian and Y.F. Zheng. gait synthesis for a biped robot climbing sloping surface using neural networks. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 2601–2611, Nice, France, 1992.

[Thr92] S.B. Thrun. The role of exploration in learning control. In D.A. White and D.A. Sofge, editors, *Handbook of intelligent control: neural, fuzzy, and adaptive approaches*, pages 527–559, New York, 1992. Van Nostrand Reinhold.

[Thr93] S. B. Thrun. Exploration and model building in mobile robot domains. In *Proc. of IEEE Int. Conf. on Neural Networks*, San Francisco, CA, 1993. available from ftp://ftp.gmd.de/Learning/rl/thrun.robots-icnn93ps.Z.

[TJS+91] E. Teixeira, G. Jayaraman, G. Shue, P. Crago, K. Loparo, and H. Chizeck. Feedback control of nonlinear multiplicative systems using neural networks: an application to electrically stimulated muscle. In *Proc. IEEE Int. Conf. on Systems Engineering*, pages 218–220, Fairborn, USA, 1991.

[TM66] R. Tomovic and R.B. McGhee. A finite state approach to the synthesis of bioengineering control systems. *IEEE Trans. on Human Factors in Electronics*, HFE 7(2):65–69, 1966.

[Tom84] R. Tomovic. Control of assistive systems by external reflex arcs. In *Advances in External Control of Human Extremities*, volume VIII, pages 7–21, Belgrade, 1984. Yugoslav Committee for ETAN.

[TP92] Chen K. Tham and Richard W. Prager. Reinforcement learning for multi-linked manipulator control. Technical Report CUED/F-INFENG/TR104, Cambridge University Engineering Department, Cambridge, UK, 1992. available from ftp://svr-ftp.eng.cam.ac.uk/pub/reports/tham_tr104.ps.Z.

[TPT87] R. Tomovic, D. Popovic, and D. Tepavec. Adaptive reflex control of assistive systems. In *Advances in External Control of Human Extremities*, volume IX, pages 207–214, Belgrade, 1987. Yugoslav Committee for ETAN.

[TS88] R. Tanscheit and E.M. Scharf. Experiments with the use of a rule-based self-organising controller for robotics applications. *Fuzzy Sets and Systems*, 26:195–214, 1988.

[TST92] T. Tani, M. Sakoda, and K. Tanaka. Fuzzy modeling by ID3 algorithm and its application to predication to prediction of heater outlet temperature. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pages 923–930, 1992.

[TVV72] R. Tomovic, M. Vukobratovic, and L. Vodovnik. Hybrid actuators for orthotic systems: hybrid assistive systems. In *Advances in External Control of Human Extremities IV*, pages 231–238. Yugoslav Committee for ETAN, 1972.

[VBK+93] P.H. Veltink, H.B.J. Bussmann, F. Koelma, H.M. Franken, L.J. Wim, and R.C. van Lummel. Feasibility of posture and movement detection by accelerometry. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 15, pages 1230–1231, 1993.

[VCCeB92] P.H. Veltink, H.J. Chizeck, P.E. Crago, and A. el Bialy. Nonlinear joint angle control for artificially stimulated muscle. *IEEE Trans. on Biomedical Engineering*, 39(4):368–380, April 1992.

[VDO92] Christopher L. Vaughan, Brian L. Davis, and Jeremy C. O'Connor. *Dynamics of human gait.* human Kinetics Publishers, Champaignm Ill., 1992.

[Veg90] John Van De Vegte. *Feedback control systems.* Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1990.

[VFVB93] P.H. Veltink, H.M. Franken, A.W. Verboon, and H.B.K. Boom. Detection of knee instability using accelerometers. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 15, pages 1232–1233, 1993.

[WA94] F. Wang and B.J. Andrews. Adaptive fuzzy logic controller for FES–computer simulation study. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 16, pages 406–407, 1994.

[Wan92] L.X. Wang. Fuzzy systems as nonlinear dynamic system identifiers. In *Proc. of IEEE Conf. on Decision and Control*, pages 897–902, 1992.

[Wan93] L.X. Wang. Stable adaptive fuzzy control of nonlinear systems. *IEEE Trans. on Fuzzy Systems*, 1(2):146–155, May 1993.

[Wan94a] L.X. Wang. A mathematical formulation of hierarchical systems using fuzzy logic systems. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pages 183–188, Orlando, USA, 1994.

[Wan94b] L.X. Wang. A supervisory controller for fuzzy control systems that guarantees stability. *IEEE Trans. on Automatic Control*, 39(9):1845–1847, September 1994.

[WB93] R.J. Williams and L.M. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, College of Computer Science, Northeastern University, Boston, USA, 1993. available from ftp://ftp.gmd.de/Learning/rl/williams.perf-bound.ps.Z.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q learning. *Machine Learning*, 8:279–292, 1992.

[Wer90] P.J. Werbos. Backpropagation through time: what it is and how to do it. *Proc. of IEEE*, 3, October 1990.

[Wer92a] P.J. Werbos. Approximate dynamic programming for real-time control and neural modelling. In D.A. White and D.A. Sofge, editors, *Handbook of intelligent control: neural, fuzzy, and adaptive approaches*, pages 493–525, New York, 1992. Van Nostrand Reinhold.

[Wer92b] P.J. Werbos. Neurocontrol and supervised learning: An overview and evaluation. In D.A. White and D.A. Sofge, editors, *Handbook of intelligent control: neural, fuzzy, and adaptive approaches*, pages 65–89, New York, 1992. Van Nostrand Reinhold.

[Wil87] Ronald J. Williams. A class of gradient-estimating algorithms for reinforcement learning in neural networks. In *IEEE International Conference on Neural Networks*, volume II, pages II601–II608, 1987.

[Wil92] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[WM92a] L.X. Wang and J.M. Mendel. Backpropagation fuzzy system as nonlinear dynamic system identifiers. In *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pages 1409–1418, San Diego, USA, 1992.

[WM92b] L.X. Wang and J.M. Mendel. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Trans. on Neural Networks*, 3(5):807–814, September 1992.

[WM92c] L.X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(6):1414–1427, 1992.

[WM94] J.R. Wolpaw and D.J. McFarland. Multichannel EEG-based brain-computer communication. *Electroencephalography & Clinical Neurophysiology*, 90(6):444–449, 1994.

[WP92] P.J. Werbos and A.J. Pellionisz. Neurocontrol and neurobiology. In *Proc. of IEEE-INNS Int. Joint. Conf. on Neural Networks*, pages III373–III378, Baltimore, 1992.

[WS85] B. Widrow and S.D. Stearns. *Adaptive signal processing.* Prentice-Hall, Englewood Cliffs, N.J., 1985.

[WS92] D.A. White and D.A. Sofge. *Handbook of intelligent control: neural, fuzzy, and adaptive approaches.* Van Nostrand Reinhold, New York, 1992.

[YOTN87] R.R. Yager, S. Ovchinnikov, R.M. Tong, and H.T. Nguyen. *Fuzzy sets and applications: selected papers by L.A. Zadeh.* John Wiley & Sons, New York, 1987.

[Yuh94] J. Yuh. Learning control for underwater robotic vehicles. *IEEE Control Systems Magazine*, pages 39–46, April 1994.

[YZ90] G.T. Yamaguchi and F.E. Zajac. Restoring unassisted natural gait to paraplegics via functional neuromuscular stimulation: a computer simulation study. *IEEE Trans. on Biomedical Engineering*, 37(9):886–902, September 1990.

[Zad65] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[ZBS87] B.H. Zhou, B. Baratta, and M. Solomonow. Manipulation of muscle force with firing rate and recruitment control strategies. *IEEE Trans. on Biomedical Engineering*, 34:128–140, 1987.

[ZHD+94] R. Zbikowski, K.J. Hunt, A. Dzielinski, R.M. Smith, and P.J. Gawthrop. A review of advances in neural adaptive control systems. Technical Report NACT

TP-1, Daimler-Benz AG and University of Glasgow, U.K. and Germany, 1994. (PS file available from ftp://ftp.mech.gla.ac.uk/nact/nact_tp1.ps).

[ZW90] Z.S. Zhang and F. Wang. Estimation of single trial Evoked Potentials using parametric modelling. In *Proc. of Ann. Int. Conf. of IEEE/EMBS*, volume 12, pages 906–907, 1990.