Learning Control of Bipedal Dynamic Walking Robots with Neural Networks

by

Jianjuen Hu

M.S., Electrical Engineering & Computer Science (1996)
M.S., Mechanical Engineering (1996)
Massachusetts Institute of Technology

Submitted to the Department of Electrical Engineering and Computer Science
In partial fulfillment of the requirements for the degree of

ELECTRICAL ENGINEER

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August, 1998

Signature of Author _____
Department of Electrical Engineering and Computer Science
August 31, 1998

Certified by _____
Gill A. Pratt
Assistant Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Learning Control of Bipedal Dynamic Walking Robots with Neural Networks

by
Jianjuen Hu

Submitted to the Department of Electrical Engineering and Computer Science
On August 30, 1998, in partial fulfillment of the
Requirement for the degree of
Electrical Engineer

Abstract

Stability and robustness are two important performance requirements for a dynamic walking robot. Learning and adaptation can improve stability and robustness. This thesis explores such an adaptation capability through the use of neural networks.

Three neural network models (BP, CMAC and RBF networks) are studied. The RBF network is chosen as best, despite its weakness at covering high dimensional input spaces. To overcome this problem, a self-organizing scheme of data clustering is explored. This system is applied successfully in a biped walking robot system with a supervised learning mode.

Generalized Virtual Model Control (GVMC) is also proposed in this thesis, which is inspired by a bio-mechanical model of locomotion, and is an extension of ordinary Virtual Model Control. Instead of adding virtual impedance components to the biped skeletal system in virtual Cartesian space, GVMC uses adaptation to approximately reconstruct the dynamics of the biped.

The effectiveness of these approaches is proved both theoretically and experimentally (in simulation).

Thesis Supervisor: Gill A. Pratt
Title: Assistant Professor of Electrical Engineering and Computer Science

Acknowledgments

First of all, I would like to express my deep appreciation to my thesis supervisor, Professor Gill Pratt for his kind encouragement, support and expert guidance through the course of this research. His gentle style of advising helped me in the challenging research area of biped locomotion control.

I am very grateful to all the members of the leg lab for their kind help during the years. It has been a great experience to stay with such a group of people who have a broad spectrum of talents in robotics, mechanical design, computer, and electronics. Particularly, I want to thank Jerry Pratt for his unselfish help and support through my research. The simulation of biped locomotion learning control was based on his previous simulation code (with the virtual model control scheme). He offered me a lot kind help in solving my puzzles in the creature library. I also wish to thank Bruce Deffenbaugh for his friendship and encouragement when I was exploring in the darkness. We spent many night hours sitting together and talking about many interesting things. His wisdom and endless stories made me feel fresh before I ran out of gas in my research during the night. Thanks to Dr. Hugh Herr for his help in setting up external force experiment in my simulation, which is an important part in this research. Also I appreciate the help of Professor J.J. Slotine for the interesting and constructive discussions on neural networks between us.

I can never forget to thank Marilyn Pierce, my academic counselor of the EE graduate program. Her kindness and advising in my course selection were also important for me to pursue my ideal goal in EECS department of MIT.

Finally I want to thank my parents, my wife, and my son. Their spiritual support and sacrifice made me be strong enough to accomplish my study in America and overcome all the adversity I had in my life. They gave me endless power in my work. I can always feel their support although they are far away from me.

This thesis is dedicated to my parents
for their love and support

Contents

List of Figures

Chapter 1

Introduction

There are two main research focuses in legged locomotion: biological research and engineering research.

In the past, robotics researchers utilized the results of external macro behavior characteristics measured by biologists and built walking robots such that the desired behavior (for example, gaits, efficiency etc.) could be achieved. Today, the advances in biology can provide us some more useful intrinsic models of locomotion with present available technologies. It is time to pay attention to the intrinsic microscopic models of locomotion and exploit them for engineering use. For example, in motor control of biped animals, some neural control models may be useful in today's engineering research.

This thesis emphasizes the dynamics of a bipedal robot, including the essential mechanism, the structure its dynamics, dynamic control, stability, robustness and adaptation. Before going to the specific sections, a review of the research in the area of bipedal leg locomotion is helpful to better understand the approaches in this study. Besides, some important concepts need to be explained clearly.

1.1 Review of studies in bipedal leg locomotion

Significant advances have occurred over the past two decades in issues related to mechanical (structure and actuators) design of legged robots, coordination and the control of legs during locomotion. There have been several driving forces. First, exploring the biological mechanism of bipedal locomotion has a strong scientific significance. Second, potential applications have been a driven force. In particular occasions, legged machines are required for transportation or system mobility where wheeled machines are limited, such as rough, irregular terrain. Third, computer technologies make on-board computer control of leg locomotion possible today. To replicate human beings (humanoid robots) or an animal becomes a very exciting research area (Raibert 1986).

1.1.1 Biologically motivated studies of leg locomotion.

Animal locomotion has been studied by neurobiologists, zoologists, orthopedists, bioengineers and physiologists (not to mention anatomists, sports scientists, dancers and acrobaticians etc.). The early studies concentrated on macro behavior, such as energetics and gaits (McMahon 1984, Alexander 1990). Bipeds (in vertebral category) have relatively simpler macro gaits by moving their two lower limbs either in phase (hopping) or in alternate phase (walking and running), which are combinations of double support phase, single support phase and swing phase. In the studies of bipeds, attention was particularly paid at walking/running speed, stride frequency and oxygen consumption, efficiency etc.

About two decades ago, researchers made significant progress in understanding the features of neural control of animal locomotion. In the seminal work of Grillner (1976), a spinalized cat was shown to have adequate reflexive control to walk relatively on a treadmill. Walking patterns are also generated in such a cat (although altered significantly in magnitude and speed), when all afferent inputs are cut. This showed that the timing mechanism of the neural circuitry can free-

run, thus mandating the existence of dynamic internal state and feedback with spinal circuits. So far, the biological researchers have proved that both reactive control and central pattern generators exist in vertebrates (spinal cord), and the feedback loop that generates the motor patterns is closed both by internal state variables and by the environment. This implies that the lower level dynamic control is in charge by the neural circuits in the spinal cord (not in the brain), but the neural circuits (in spinal cord) are also affected by the inputs from higher-level control systems.

Researchers in biology propose motor control models for bipedal leg locomotion. With a proper model, one can predict a human walker's or runner's performance in the sense of energetics and dynamics. McMahon (1984) proposed a model of the bipedal leg, which was used to predict a runner's performance on a compliant track accurately. His running model of the leg is composed of a parallel spring and dash-pot, which was approximately a macro model for the leg muscles. Morecki's general rheolegical model of isolated muscle (Morecki 1987) captures the key characteristics of limb muscles, and can be used for dynamics analysis of bipedal walker and runner as well as the possible application of bipedal robot design. In Morecki's model, spring and dash-pot are connected in series, like a series elastic link, and the dash-pot (muscle viscosity) is dependent on the stimulation. In a passive state, a muscle, when unstimulated, and passively stretched under static/dynamic conditions behaves like a non-linear spring. In an active state, a muscle behaves as a control force source with an elastic component in series.

In the study of locomotory control, elucidation of locomotor behavior supplies us with an important clue as to how the motor pattern is generated through the interplay of motor and sensory systems. It has been generally shown that rhythmic motor patterns are coordinated by neural circuits referred to as central pattern generators (Grillner 1976, 1985), and the controlled musculo-skeletal system (body) faithfully responds to the commands of the nervous system in master-slave manner, but is influenced by functional and environmental constraints (Taga 1991). This has inspired theoretical studies of motor pattern generators in isolated or distributed neural networks in the absence of sensory feedback (Miller & Scott 1977). How sensory feedback interacts with the central pattern generator so as to adapt locomotory system to the environment is still an open question.

In biology area, there are a few groups that are trying to execute motory adaptation by means of blending sensory inputs from the periphery with ongoing centrally patterned activity (Chiel & Beer 1992, Parkins 1997, Taga 1998). On the other hand, based on the finding in neuro-physiological studies that a hierarchical structure is presented in locomotion system (Grillner 1975), some researchers initiated their explorations of motor system adaptation with cerebellum functions. It has been suggested that the whole aspects of cerebellar function could be explained in terms of adaptive learning control (Yuasa & Ito 1990). The concern of the cerebellum seems to be the generalized integration and graceful smoothing of behaviors, which is heavily dependent on sensory input below the level of consciousness. Ito (1990) proposed a control system model which views the cerebellum facilitating adaptive control of all kinds of functions – from reflexes to voluntary movements to mental activity. His adaptive control mechanism would emerge from the self-organizing capability postulated by Marr (1969), Albus (1971) and Fujita (1982).

1.1.2 Research progress of bipedal walking in engineering aspects.

While researchers in biology were investigating how motor control and motor pattern generators work in the leg locomotion biologically, engineers in robotics field were induldged in their

studies on legged machines for travelling on rough terrain where existing wheeled vehicles can not go. Basically, their aims were disigning and analysing biped locomotion from the viewpoint of torque control and actuation as well as system implementation. The first walking machine, an eight-legged kinematically walking machine, was built in 1961 (Morrison 1968). But the first bipedal walking robot was born in 1974. It was a hydraulic biped that could walk with quasi-dynamic gait (Kato 1974).

One of the pioneers of bipedal robotics in the early years is Vokobratovic of former Yugoslavia. Vukobrutovic introduced the concept of "Zero Moment Point Control" in the 1960s (Vokobratovic 1973), which is widely used recently by the engineers for controlling their bipedal robots. In 1974, the Wabot (Kato 1974), a hydraulically powered biped, (one of the most photographed walking robots), was built by Kato's group in Waseda University, Japan. Wabot has two legs and many other anthropomorphic features such as arms, head-mounted visual sensors and voice communications. It was statically stable at all time by keeping its center of mass above one of its large feet. A modified zero moment control approach was utilized. Since then, there are several bipedal robots that have been built successfully with proper control. In 1983, a group at Osaka University built a biped and developed a method called hierarchical control structure, which allowed dynamically stable walking (Miyazaki & Arimoto 1983). Another series of bipeds for dynamic walking experiments was built at University of Tokyo. One of these, named BIPER-4, has seven electrically powered joints: a hip roll, ankle pitch and knee joint in each leg and hip pitch joint connecting the two legs (Miura & Shimoyana 1984). Dynamically stable walking was also achieved for the BIPER-3, which had un-powered ankle joints and no knee joints. The feet do not contribute to pitch stability but allow the angle of the leg from the ground plane to be measured. In 1991, Kajita & Tani presented their experimental results of zero moment control with their biped "Meltran II", which had 40 cm height and 4 kg weight with electric DC motor actuators (Kajita & Tani 1995). Two bipedal walking robots, Spring Turkey (in 1995) and Flamingo (in 1997) were built in MIT Leg Lab. Both robots can walk dynamically stable with "Virtual Model Control" (Pratt, J. 1996). Both had electric series-elastic actuators (Pratt, G. 1995) and the latter had an on board computer control system. Recently, a humanoid bipedal robot built by Honda Motor Company in Japan (Hirai, Hirose et al 1998, Ozawa 1995) has integrated many existing technologies in control and navigation, such as, zero moment control, vision guidance etc. The Honda robot can walk up/down stairs stably and can also perform some simple task with its hands.

To summarize, there are three important issues in the bipedal walking robot study: control, gaits and actuation. Control has been regarded as the most crucial aspect and has received considerable attention in the past comparing with the other two issues. Nearly all the walking machines constructed so far were built for control studies. Many aspects of control are still under active study. This is particularly true for control of statically unstable locomotion. As for the gait study, the results from biological research such as motor pattern generator theory etc have not been applied in the bipedal walking robots although there are a few preliminary investigations in quadrupeds and hexapods (Todd 1985, Raibert 1986). In bipedal robots, only global discrete gaits are used successfully. In the existing bipedal robots, both hydraulically and electrically powered bipedal actuators have been well explored thus far. Elastic components are widely recognized as necessary components for a biped. The series-elastic actuator (Pratt, G. 1995) developed in MIT Leg Lab is one of the success achieved in bipedal robots controlled electrically.

1.1.3 Existing issues in leg locomotion

The following issues have drawn researchers' attentions:

1) Adaptation with central pattern generators;
2) Mixing CPG with feedback sensory information;
3) Intelligently dealing with complex tasks;
4) Hierarchical learning and control;
5) Cerebellum function in dynamic control of locomotion;
6) Robot actuation technique;
7) Robot structure design;
8) Stability and gait adjustment;
9) Robot system robustness;
10) Vision integration and autonomous navigation.

1.2 Dynamic control of bipedal walking robots

Since a biped mechanical system usually has high order and nonlinear, complex dynamics, it is a very hard task to design controllers for the joint actuators with full consideration of the entire biped system dynamics. Besides, from a mechanical point of view, a biped robot is inherently unstable because the center of mass extends beyond the base of support most of the time during walking.

For such a complex system, usually the control engineers' dilemma is how to make a suitable trade-off between the simplification of the system model and control precision. In walking machine design, there are two different approaches to controller design in current use. One approach is to design effective controllers based on some approximation of the dynamics of the bipedal mechanical system (McGeer 1990). Another approach, (called model-free control design in the thesis), aims to make use of the control engineers' experience, intuition and learning techniques (Pratt 1996, Murakami 1995, Miller 1994, Hu 1998).

One of the common features of the control approaches developed by researchers is that controllers of bipeds were designed based on approximations of the bipedal mechanical system. In the early days of 1970s, the simplest model used for the study of some of the characteristics of human walking is the inverted pendulum model. More complex models with more degrees of freedom were used mainly after 1980 for a more complete study of human walking (or other biped animals) as well as for the actual construction of biped robotic systems. For example, Hemami et al used the inverted pendulum model to investigate the biped stability in 1977 (Golliday & Hemami 1977). For control design, Golliday and Hemami used state feedback to decouple the high-order system of a biped into independent low-order subsystems. Miyazaki and Arimoto (1980) used a singular perturbation technique and showed that bipedal locomotion can be divided into two modes: a fast mode and a slow mode, thus simplifing the controller design. Furusho and Masubuchi (1987) derived a reduced order model as a dominant subsystem that approximates the original high-order model very well by applying local feedback control to each joint of a biped robot. Raibert (1986) used symmetry to analyze his hopping robots controlled by his "three part control". Miura and Shimoyama (1984) linearized the biped dynamics and designed stable controllers by means of linear feedback. Kajita and Tani (1995) developed their 6 d.o.f. bipedal robot "Meltran II" using a "Linear Inverted Pendulum Mode" successfully. A research group of Honda Motor Company designed their control system for a humanoid bipedal robot using zero moment force control in dealing with the complex dynamics (Ozawa 1995). In

the above research, the dynamics of the biped robots were simplified so as to utilize the existing modern control theory in the controller design.

On the other hand, the model-free control approach does not incorporate the complex intrinsic dynamics of the biped robot system in the controller design. Basically, this type of approach avoids dealing with the complex dynamics directly and, instead, it tries to incorporate the control engineer's intuition, experience and knowledge into the control system design. Pratt (1996, 1997) developed virtual model control, by which one can achieve good control capability for biped walking robots through appropriate choice of virtual components in virtual space. In addition, other researchers have made good progresses in control of biped robots by means of learning techniques such as fuzzy logic and neural network control (Miller 1994, Murakami 1995).

In summary, because of the complexity of the system dynamics in walking robots, the stability and robustness analyses are very difficult to accomplish. In this study, I took the virtual model control approach and proposed a new control mechanism, adaptive virtual model control, which can enhance the robustness of the intuition based virtual model control for a walking robot, especially for a bipedal walking robot. The adaptation mechanism designed in the virtual space is described in the following sections. Some simulation results are also presented.

1.3 System performance index for biped control systems

Finding a proper performance index function is an important step for learning control scheme. Stability is an important performance index for bio-mechanical systems like legged robots. In general, the performance measurements of walking robots are much different from the typical notions of performance for manipulators, such as command following and disturbance rejection. The overall performance of a walking robot is usually defined in terms of biological similarity, efficiency, locomotion, smoothness, and robustness to the environments.

Robustness for a walking robot implies that its stability should be maintained when encountering unexpected external disturbances and complex environments. There are two types of stability for a legged robot. First, the stability of a legged robot requires the stability of internal structure dynamics in each individual phase under external disturbances, which means that, in a stance phase, the robot can maintain a proper posture. Second, the robot stability also requires gait stability (or step to step stability). It means that the robot has persistent smooth movements in complex environments, which can be achieved by means of appropriate gait adjustment. In this study, the adaptive VMC is utilized to meet the first requirement for stability. The second requirement should be achieved through an adaptive gait control approach and it is not the focus of this study.

1.4 Learning and robustness enhancement in bipedal robots

The performance of current legged robots remains far below even simplest counterparts in the biological world. This has led to a search by scientists for biologically motivated approaches to the design and control of legged robots in order to improve their performance and robustness. Generally there are several key factors that affect the performance of a bipedal robot, such as system stability, robustness, adaptation and learning. In control engineering, the above factors are tied to the dynamics, but for a bipedal walking robot, one can refer those factors in different levels, like dynamics level, skill level and task planning level etc. In this research, only the dynamics level is addressed

Since a bipedal walking robot has not only structure dynamics, but also neural dynamics (or gait dynamics, motor dynamics), one needs to explore a broader sense of definitions for stability, robustness, adaptation and learning. The followings are some preliminary considerations in the study.

Stability: Stability for a walking robot means that the robot can maintain desired walking gaits consistently without falling down.

Robustness: Robustness is the capability that a robot can tolerate the disturbances and maintain the nominal motions.

Adaptation: A robot can make adjustments whenever encountering the variations of environments.

Learning: In dynamics level, learning means gaining knowledge from the interactions with the system environments and intending to improve the robot performance. Learning can be in supervised mode or in unsupervised mode.

So far, the learning approaches used for bipedal walking robots are neural networks, fuzzy logic and reinforcement learning. In dynamics level, neural networks based learning becomes more mathematically mature and convenient to conduct dynamics reconstruction or identification and further adaptation.

1.5 Outline of thesis

This thesis is organized as follows:

Chapter 1 briefly reviews the research on leg locomotion in the past decades.

Chapter 2 describes the biologically motivated models of locomotion.

Chapter 3 presents extensions of virtual model control and the adaptive virtual model control.

Chapter 4 introduces three different types of neural network models (like BP, CMAC, RBF neural networks) and the corresponding learning algorithms. The comparisons on their learning capabilities are also illustrated.

Chapter 5 describes the framework of radial basis function (RBF) neural network control of a bipedal walking robot. A self-organized RBF neural network is developed.

Chapter 6 presents the neural network based adaptive dynamic control design of a bipedal walking robot. Both the theory and the results are elucidated and discussed.

Chapter 7 summarizes the research of this thesis and proposes future research directions.

Chapter 2

Bio-mechanical Models of Legged Locomotion

Biological researchers study the fundamental basis in neuroscience for animal leg locomotion. The concept of hierarchical neural control was proposed and proved experimentally. Today it is known that the central pattern generators in spinal cord can drive the animal legs walk mechanically and the high level control of cerebellum make it walk better. Many models of neural pattern generator circuits were proposed to describe the motor control in animal locomotion. On the other hand, robotics engineers used the available technologies (computer, power electronics, motors and materials) and tried to build legged robots such that the robots behaved like animals. Good progress has been made by engineers in robot design, actuators, global discrete gait and dynamic control etc. However, the researches in biology science and robotics engineering were moving in parallel for a long time. Only a few researchers tried to apply the results of biology area into the robotics research, such as in hexapod case (Chiel & Beer et al 1992). Yet, there is little application of biological findings to biped locomotion.

Establishing a bridge between the biological research and robotics engineering and utilizing the results from biology science to help engineers better understand the bipeds locomotion may lead to better robot performance. Legged robots, especially bipeds perform worse than their biological counterparts. Why shouldn't we learn from the animals? Making use of the results in biology will help robotics researcher improve the performance of their robots, particularly for bipedal robots. The biological model inspired control approach may lead to significant performance improvement in the future comparing with the traditional control approaches described in the previous chapter. This chapter provides a survey on bio-mechanical model based control approaches.

2.1 Central pattern generator model

It has been a mystery for a long time how pattern generators work in the motor control of animal locomotion. Grillner's demonstration (1976) that imposed rhythmic movements of the legs of spinal cats can entrain central rhythms over a wide range of frequency gave us a crucial clue for the hierarchical sensory motor control. In a system of legged locomotion, the high level nervous system and spinal cord rhythmic nervous system control the musculo-skeletal system in a hierarchical manner. The high level nervous system such as cerebellum is in charge of the intelligence part of locomotion, e.g., adaptation, learning etc. But the body below the head (i.e. lower limbs) should move with proper rhythms (patterns) under the pattern generators' driving and the local sensory feedback as well as impedance control. As a very complex and elegant locomotory system, human leg locomotion was studied starting with the central pattern generator about ten years ago. Several neural network circuits for the rhythmic oscillators were proposed for the bipedal locomotion (Matsuoka 1985, Taga 1991).

2.1.1 Matuoka's model

Among many models of motor pattern generator, one of the simplest neural network models (circuits), which generate oscillating activity consists of two tonically excited neurons, with the adaptation (to neuron itself) or self-inhibition effect, linked reciprocally via inhibitory

connections. This model was originally developed by Brown (1914) to account for the alternating activation of flexor and extensor muscles of a cat's limbs during walking. Based on this, Matsuoka (1985) did systematic investigation of a series of pattern generator models with different number of neurons. Matsuoka's model can be mathematically represented by the following continuous differential equations:

$$\tau \cdot \dot{u}_1 = -u_1 - w \cdot y_2 - \beta v_1 + u_0 \tag{2-1}$$

$$\tau \cdot \dot{u}_2 = -u_2 - w \cdot y_1 - \beta v_2 + u_0 \tag{2-2}$$

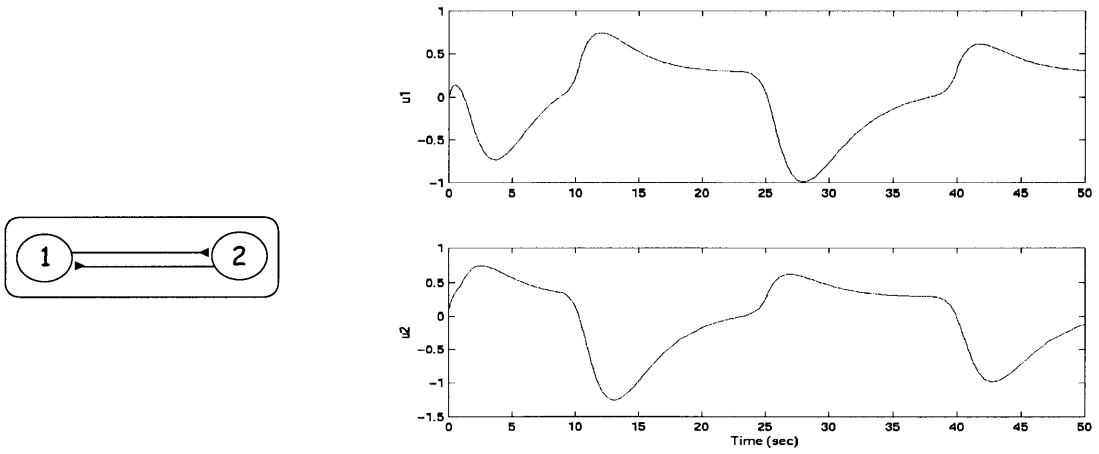
$$\tau' \cdot \dot{v}_1 = -v_1 - y_1 \tag{2-3}$$

$$\tau' \cdot \dot{v}_2 = -v_{21} - y_2 \tag{2-4}$$

where $y_i = f(u_i)$, (2-5)

$$f(x) = \max(0, x), \; i = 1,2. \tag{2-6}$$

In the above model, u_i is the state of the ith neuron. y_i is the output of the ith neuron. v_i is the co-state of the ith neuron or a variable representing the degree of adaptation or self-inhibition effect of the ith neuron. u_0 is an external input. w is a connecting weight; and τ and τ' are time constants. Matsuoka explored oscillator network by linearization and using linear system theory. It has been found that the time constants τ and τ' change the frequency of oscillation and the external input term u_0 changes the amplitude of the system output. Figure 2-1 shows the typical responses of a two-neuron central pattern generator.



(a) Diagram of neural oscillator (b) Responses of neural oscillator

Figure 2-1: Responses of a two-neuron pattern generator.

2.1.2 Taga's model

Taga (1991, 1995) extended Matsuoka's model for the purpose of his complex motor control by adding new excitation inputs to the oscillator circuit input. His neural rhyth generator is expressed by the following differential equations:

$$\tau_i \cdot \dot{u}_i = -u_i + \sum_{i,j=1}^{N} w_{ij} y_j - \beta v_i + u_0 + q_i \tag{2-7}$$

$$\tau_i' \cdot \dot{v}_i = -v_i + y_i \tag{2-8}$$

$$y_i = f(u_i), \quad i = 1,2,\cdots,N. \tag{2-9}$$

where q_i is a sensory feedback from the musculo-skeletal system. It can be the kinematic or postural sensory information, or some sort of the dynamics information.
Observation: q_i can be treated as a high-level nervous system command for system adaptation.

2.2 A bio-mechanical model of locomotion

There are two crucial types of dynamics in human locomotion: musculo-skeletal dynamics and neural dynamics of central pattern generator (CPG). The former is on the dynamics of the musculo-skeletal system in response to a given set of neural inputs (Taga 1995). The latter is to reveal the neural dynamics, i.e. how the basic rhythm of movement is controlled by rhythm-generating networks (CPG) in the nervous system (Grillner 1985, Taga 1991, Chiel et al 1992). It is challenging and very important to develop a principle which links the above dynamics (musculo-skeletal dynamics and neural dynamics). Taga developed an integrative principle of linkage between neural and musculo-skeletal dynamics and proposed that 'global entrainment' between the neural and musculo-skeletal systems generates stable and flexible locomotion in an unpredictable environment. By their computer simulations, Taga demonstrated that the locomotion of a biped with a simple musculo-skeletal system emerged as a limit cycle which was stable against mechanical perturbation and environmental changes. Based on their work, they proposed an integrated biomechanical model of locomotion including dynamics of the human musculo-skeletal system, neural dynamics of CPG and sensory mechanisms. Using this model, a stable gait was achieved.

In this section, Taga's biomechanical model is described and analyzed briefly. Then the significance of this model to the study of biped robots is presented.

2.2.1 Bio-mechanical model of human locomotion

In his simulation study of human locomotion, Taga used the model (Figure 2-2) to reproduce stable human-like walking. Figure 2-2 shows the basic structure of the proposed model of human locomotion for the human body including trunk and limbs. In this model, there are two dynamical systems: a musculo-skeletal system and a neural rhythm generator composed of neural oscillators (or called central pattern generators, CPGs). The musculo-skeletal system is controlled by the output signals from the motor command generator module, which combines the outputs from the neural rhythm generator and the internal impedance controller. The output signals from the neural system induce body movements by activating muscles in the musculo-skeletal system, which also interacts with the environments. The current state of the musculo-skeletal system and the environment is detected and observed by a sensory system and sent to the neural rhythm generator and the internal impedance controller. The higher center of the neural system (such as cerebellar and cerebral neural sub-systems) regulates the activity level of the neural rhythm generator by sending a nonspecific input.

A distributed motor control scheme is suggested in this locomotion model. The basic strategy for locomotion control is to establish a stable limit cycle (gait state cycle) using global

entrainment (Taga 1995) between the neural and musculo-skeletal systems since both systems have an oscillatory characteristic. A gait state cycle is described in Figure 2-3.

The mechanical conditions of the limbs change drastically within a gait cycle. This implies that the control process must also change according to the phase (state) of the gait. It is assumed that a gait can be represented as a sequence of so called 'global states' and that both the generation of motor commands and sensing processing are modulated by the global states, so that dynamic linking between the neural rhythm generator and musculo-skeletal system is established and maintained. The global states also control the coupling among the neural oscillators, which enables generation of a complex pattern of activity with appropriate timing. The main functions of two parallel control modules in this model are the following. The neural rhythm generator controls rhythmic movements and the internal impedance control aims to adding mechanical impedance components (e.g. springs and dampers), which maintains the upper body in an upright posture and prevents limbs in the stance phase from collapsing.

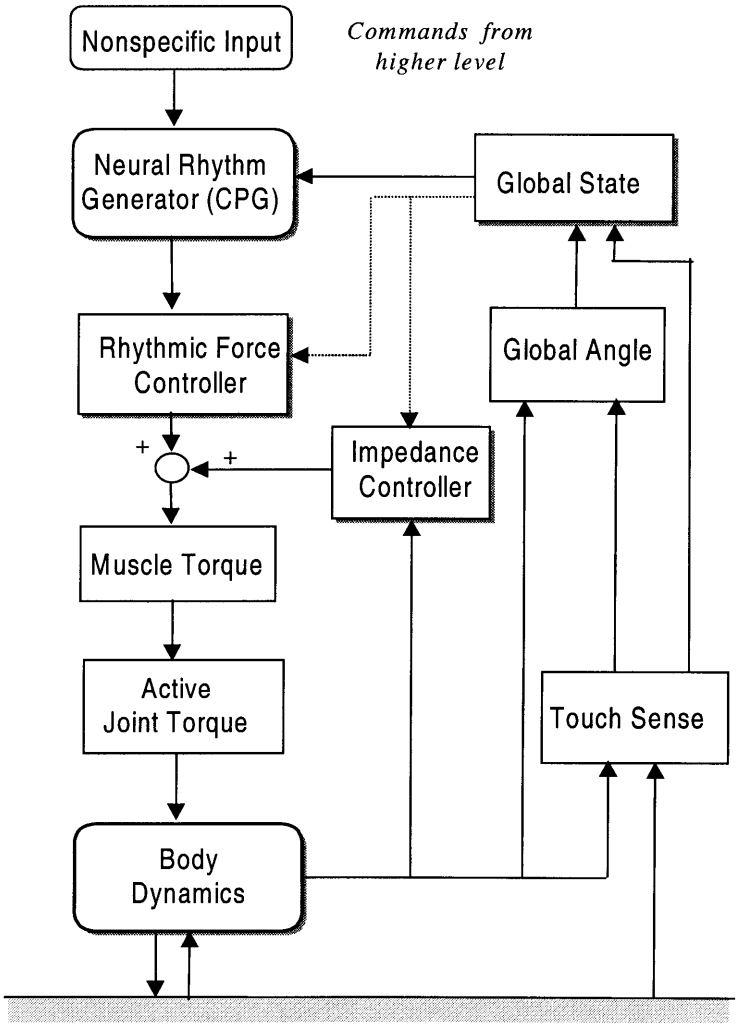


Figure 2-2: Taga's neuro-mechanical model of human locomotion

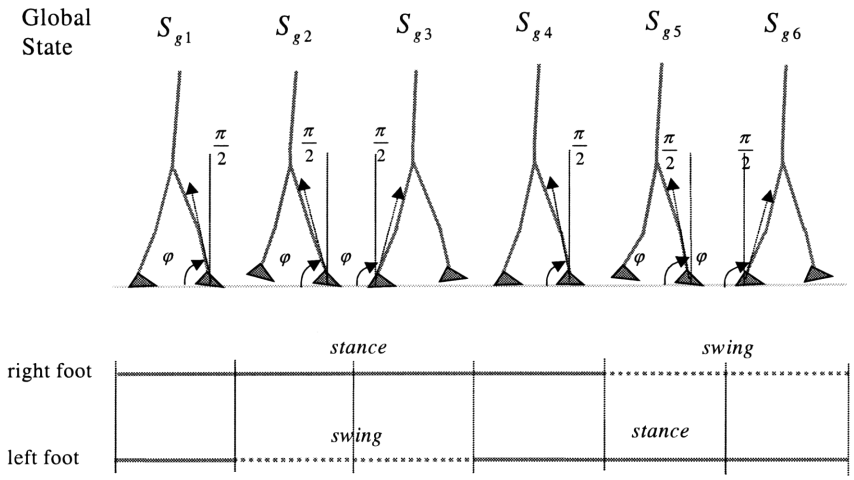


Figure 2-3: Diagram of the global gait states in a cycle.

2.2.2 Implementation of the distributed human locomotion model in simulation

The above biped locomotion model was used in the computer simulation to reproduce human-like walking. During the simulation study, the body is represented by eight segments (refer to Figure 2-4b), which are two complete three-segment lower limbs (thigh, shank, and foot) and two segments in the upper body (trunk, pelvis). Each foot is represented by a triangle of appropriate proportions. The interaction between the foot and the ground is modeled as springs and dampers acting at the heels and toes. The body model is confined to movement in the saggital plane.

There are seven joints in the body: six in the lower limbs (hip, knee, and ankle for each limb) and one in upper body (trunk). Correspondingly, the neural rhythm generator consists of seven neural oscillators (refer to Figure 2-4a), with one neural oscillator for the trunk and a pair of three neural oscillators for the limbs, each of which induces the action of specific muscles. There are twenty muscles in total in the body. Therefore, the implementation of locomotory (motor) control is realized in a decentralized way, i.e. twenty muscle equations plus seven neural oscillator dynamic equations are simulated with distributed dynamics.

Each neural oscillator controls muscle around a single joint and it receives sensory signals from the adjacent segments. The active torques at the joints are generated by twenty muscles. The muscle model is simply assumed as a pure force source which is proportional to the output of the neural system (i.e. motor command generation module in Figure 2-2). The motor command generation module is simply realized by summing the signals from impedance controller and the signals from neural oscillator. The passive torques exerted on joints are expressed by small viscous torques at each joint and large visco-elastic torques that limits the range of joint flexion and extension.

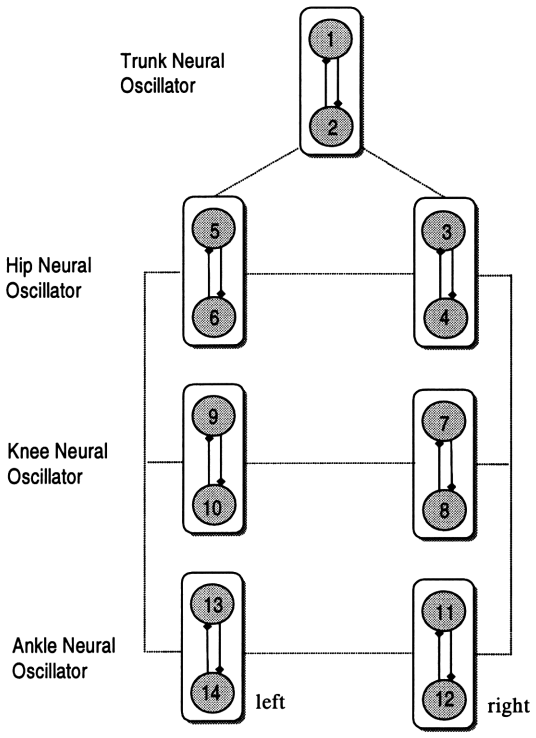


Figure 2-4a: Neural oscillator distribution

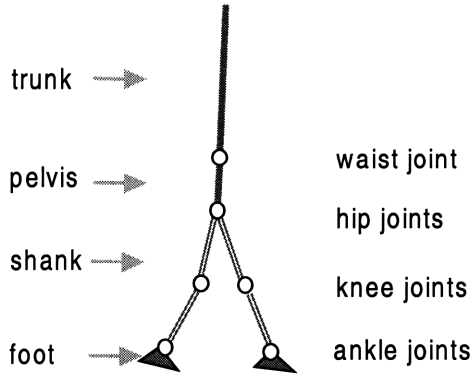


Figure 2-4b: Diagram of the simulated human model

2.2.3 Significance of this model in robotics sense

Understanding and digitally reproducing the biped human locomotion can provide fundamental support and useful insights for a biped robot design and control. Taga's human locomotion model described above has integrated nearly all the main features of human walking discovered by biologists so far, and it demonstrated how a distributed neural oscillator based model could contribute in human leg locomotion systematically. It presented a way to reproduce leg locomotion with proper impedance control and CPG control as well as local CPG adaptation. This can at least help us to better understand the gait control and adaptive adjustment of gait in dynamic walking robot study. The significance of this model can be summarized as:

1) Hierarchical control structure is useful for future intelligent control system.
2) Internal impedance control can be used not only to interact with environments, but also to sustain the skeletal structure from collapsing during stance phase.
3) An adaptation mechanism has been embedded in the CPG (in a neural dynamics level). The dynamics of the neural oscillators are in the format of recurrent neural networks and have adaptation factors in them. Therefore the robustness for cyclic oscillation pattern is guaranteed.
4) Stable walking is achieved by combination of the CPG control and impedance control.
5) Entrainment between neural oscillators and mechanical system.

2.3 Control and Adaptation in human leg locomotion modelling

It is possible to apply Taga's human locomotion control model into a bipedal walking robot system in the future. Particularly, this model has the potential of being extended for the purpose of adaptation properties, which can be done in hierarchical fashion. Figure 2-5 presents an extended hierarchical control scheme of human locomotion. There are three control levels in this scheme: dynamics control level, motor intervention level and motor planing level. In dynamics control level, there are mainly three types of components, namely CPGs, sensory neurons and motor neurons. The CPG generates the motor patterns and the sensory neurons receive information from environment and the muscula-skeletal structure. Then they supply important signals to the motor neurons that produce muscle forces by combining the motor patterns and structure dynamics (impedance). This level has direct interface with the muscula-skeletal system and cerebellum module as well as the cerebral cortex.

The adaptation in the dynamics control level can be realized by adjusting CPG neural dynamics, motor force generator in motor neurons and the impedance parameters. In the motor intervention level, the main component is cerebellum, which interacts with cerebral cortex and the lower level components (i.e. CPG and motor neurons). The adaptation is realized by direct motor adjustment command to motor neurons and input command to the CPGs. The third level, motor planning level contains cerebral cortex (motor cortex and visual system). The adaptation can be implemented through commanding CPG and cerebellum from the top level. For example, by using visual feedback, an anticipation locomotion adaptive control can be achieved. As we know, blind walking can only succeed with simple environments.

Figure 2-6 is a simplified neural network control system paradigm for leg locomotion control. It has simpler system structure and can be implemented through neural networks in different levels.

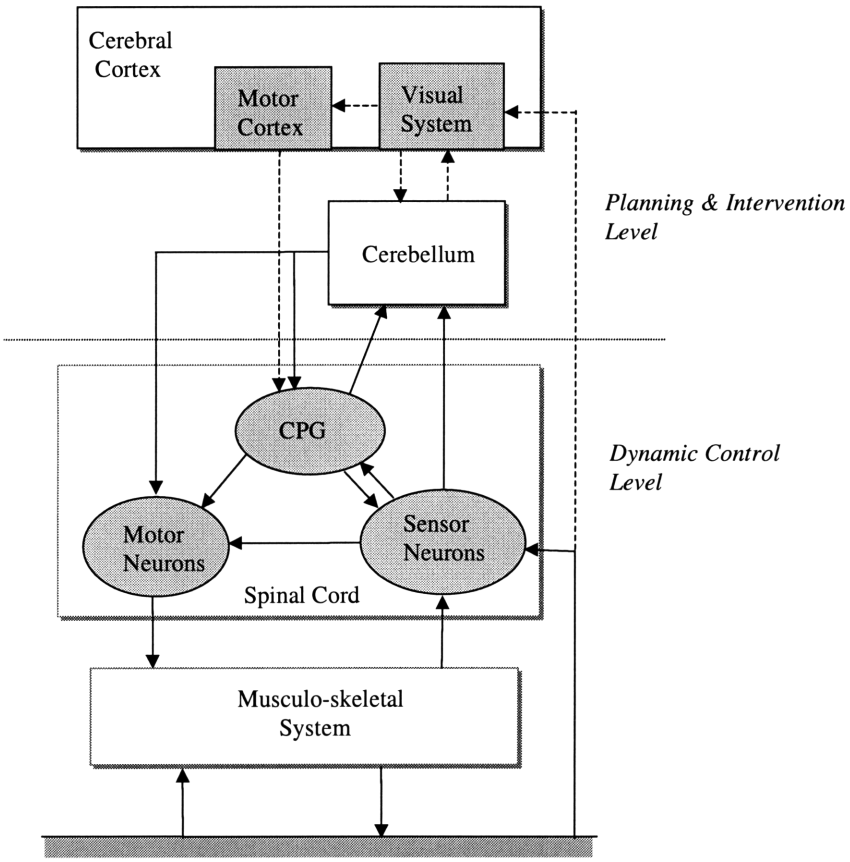


Figure 2-5: System structure for adaptation in human locomotion control. Three key components govern the motion control and adaptation of locomotion: spinal cord, cerebellum and cerebral cortex. The adaptation is conducted in a hierarchical framework, i.e. adaptation in planning and intervention level and adaptation in dynamic control level.

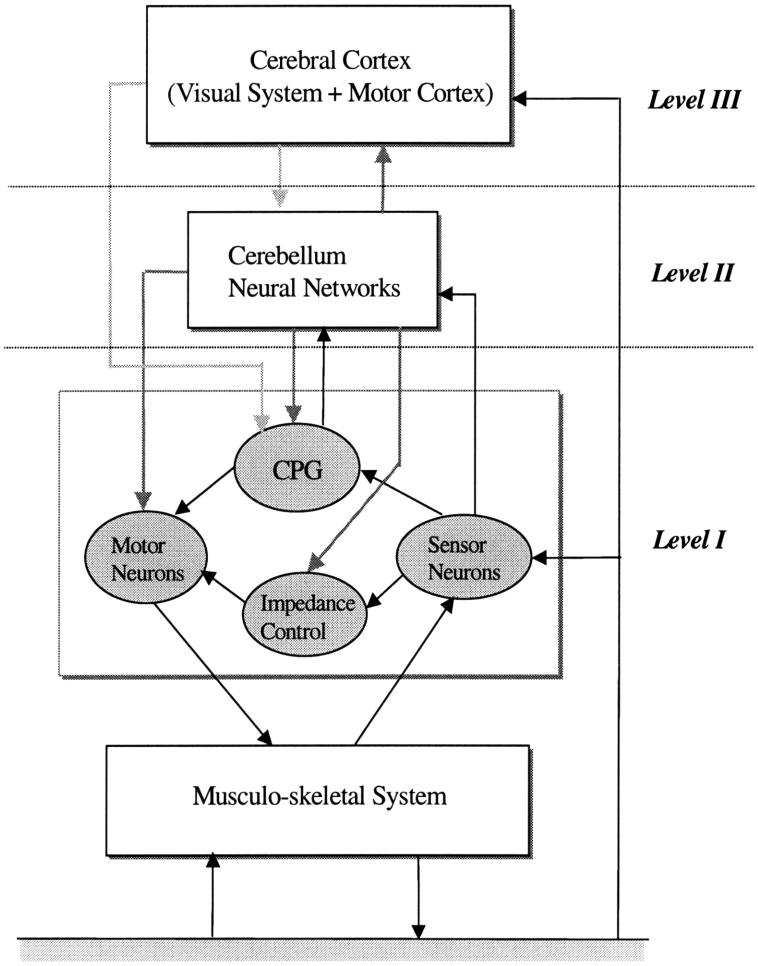


Figure 2-6: A neural network control system paradigm for leg locomotion. Three level adaptive control can be performed in a top-down hierarchical frame with appropriate neural network models.

Chapter 3

Generalized VMC of a Bipedal Robot: A Control Approach Inspired by a Bio-mechanical Locomotion Model

Research on biped locomotion has been making slow progress, mainly because of control difficulty in pursuing stable biped locomotion as compared with multi-legged locomotion. The major problem associated with the analysis and control of biped locomotion systems is the high-order, highly coupled nonlinear complex dynamics due to the multiple degree-of-freedom which is inherent to the structure and the multiple state walking gait. The complexity makes the synthesis of a control law difficult to be accomplished by the direct application of modern control theory. Several control approaches have been proposed for the bipedal walking robots, which is reviewed in section 1.2. Among those control approaches, virtual model control (VMC) has many advantages in robot control design because of the simplicity and flexibility for modifications. The purpose of this chapter is to extend the VMC to a greater mathematics framework, i.e. a generalized virtual model control (GVMC) framework, and further to find a way of designing robust control.

3.1 Virtual Model Control

Dynamically walking bipedal robots are difficult to control for several reasons. They are non-linear, passively unstable, under-actuated, and exhibit varying dynamics depending on which feet contact the ground. Because of these difficulties, textbook control solutions typically are difficult to apply. Instead, physical intuition is often used as a tool to develop a controller.

Virtual Model Control (Pratt 1996, 1997, 1998, Chew 1998) is one such technique. Virtual components are attached between parts of the physical structure of the robot and between the robot and the environment. Torque is applied to the joints of the robot so as to make the robot behave as if the virtual components are present. A finite state machine monitors the robot's configuration and discretely modulates the virtual to physical transformation and the parameters of the virtual components.

Figure 3-1 shows a diagram of one set of virtual components that can be used to control a planar bipedal walking robot. These components were used in the control of our 4-DOF walking robot Spring Turkey (Pratt 1996). Virtual spring-damper components are attached to the robot in three axes (Z, X, θ), and provide height, pitch, and forward velocity control. The "dogtrack bunny" indicates that a spring-damper mechanism in the X direction is pulled along at the desired velocity. Due to the constraint of an un-actuated ankle in this robot, the X axis spring-damper mechanism is attached only when the robot is in its double support phase of walking.

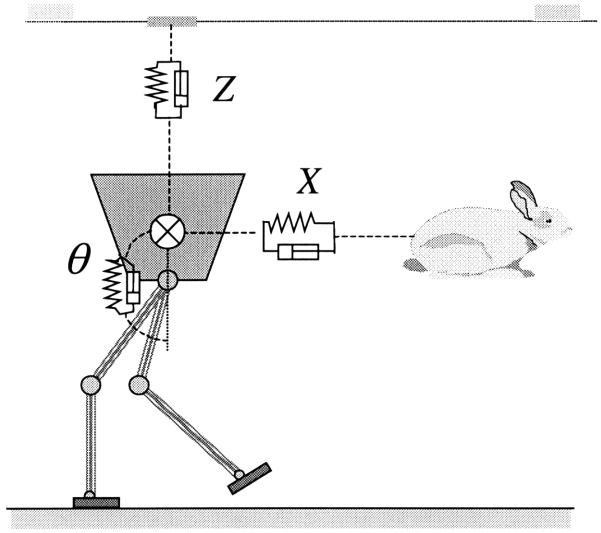


Figure 3-1: One implementation of Virtual Model Control applied to a seven-link bipedal walking robot.

There are three steps to implementing a Virtual Model Controller:

1) Design the controller by choosing virtual components and their attachment points.
2) Design the finite state machine or other method of virtual component modulation.
3) Determine the virtual to physical transformation.

Figure 3-2 shows a state machine that was used in the control of our bipedal robots "Spring Turkey" and, later, "Spring Flamingo" (which had actuated ankles). The virtual to physical transformation is based on the robot's Jacobian and some additional constraints (Pratt 1996).

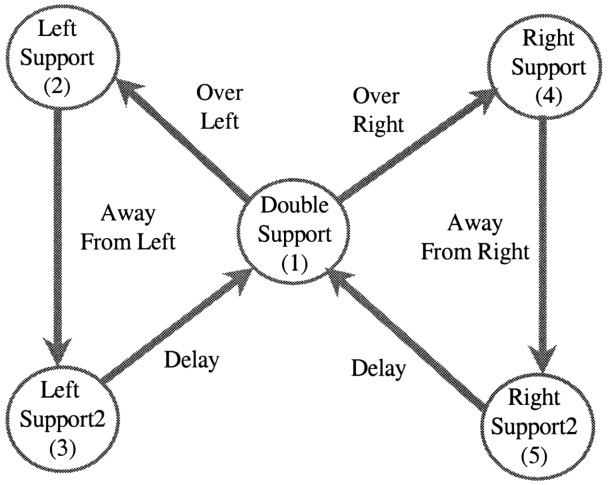


Figure 3-2: Diagram of the global state machine for a bipedal walking robot.

3.2 Generalized virtual model control (GVMC)

In the study of dynamic biped locomotion, many researchers have been trying to discover walking characteristics by measuring the human motions during locomotion in a experimental way and most recently to explore the intrinsic bio-mechanical models of locomotion, including adaptation of CPG and muscle control mechanism theoretically. On the other hand, bipedal leg locomotion was also investigated by many engineering oriented groups with target in finding the control law of biped walking and in deriving the control algorithm of biped locomotion machines. The performance of current legged robots, however, remains far below even simplest counterparts in the biological world. This has led to a search for biologically motivated approaches to the design and control of legged robots in order to improve their performance and robustness.

To enhance the robustness and other control performance in the original VMC, one needs to consider the unmodelled (high-order) dynamics in the real mechanical structure and the imperfect situation of the actuator dynamics as well as the pattern generator dynamics based on the VMC framework. In order to accomplish the above, we extend the concept of VMC into a generalized VMC (GVMC) framework in virtual dynamics space (or briefly virtual space). The GVMC approach was inspired by a bio-mechanical leg locomotion model proposed in Taga's paper (Taga 1995). Basing on of the GVMC frameworks, robust control design procedure is developed in this study. Then the implementation process of this approach is also described. At the end, the effectiveness of the GVMC is shown by the simulation results.

3.2.1 Robot control with inspiration of a biological human locomotion model

As described in section 2.2 and section 2.3, a bio-mechanical human locomotion model developed by Taga consists of neural dynamics of CPG, musculo-skeletal dynamics and motor control module. The musculo-skeletal system is controlled by the outputs from the motor

command generation module, which combines the output signals from the neural rhythm generator and the internal impedance controller. Consequently the system control can be considered as two parallel control modules: neural rhythm generator (CPG) driven control and impedance control. The former controls rhythmic movements and propels the system to move with desired motion patterns, and the latter maintains the upper body in an appropriate upright posture in stance phase.

By means of the global state variable, the musculo-skeletal dynamics and neural dynamics are integrated together, and thus an system causality is formed. This means that the neural dynamics (of CPG) provides part of the driving force and impedance control provides the other part, and the information of the entire skeletal system dynamics is fed back in terms of the global state variable. This process is actually a modulation processing.

The strategy proposed in Taga's human locomotion simulation study can be potentially utilized in robot control system. In this locomotion model, the motor control scheme is in a distributed control fashion. The basic strategy for locomotion control is to establish a stable limit cycle (global gait state cycle) using global entrainment variable (Taga 1991, 1995) between the neural and musculo-skeletal systems since both systems have an oscillatory characteristics.

There are several features of Taga's locomotion model that can be used in locomotion control for a walking robot:

1) In dynamics level, locomotion control is composed of impedance control and CPG motor control. The impedance control provides the support for the skeletal structure in the gravity field, and the CPG signals modulated by a global motor modulator yield the forces to propel the bipedal walking robot in desired patterns. The modulation is achieved by means of the global state variable S_{gi} $(i=1,2,...,6)$.
2) The global state, like a regular state machine is used as a modulator in the locomotion control.
3) Adaptation can be achieved through CPG, impedance parameter modification and the motor modulator as well as the higher level commands and interventions.
4) The CPG driven forces are generated in a distributed fashion, then they are integrated into joint torque signals (in centralized fashion) for joint actuation. Therefore, the locomotion can be implemented in both decentralized (distributed) way and centralized way.
5) The strategy can be duplicated in robot control because the bio-mechanical model is greatly simplified in a biological sense.

From Taga's bio-mechanical model, it can be suggested that three components are crucial for appropriate biped locomotion control system. They are: a) CPG (distributed or centralized), b) impedance control, c) motor modulation with global states (gaits). According to the above insights, a more general sense of virtual model control approach is proposed in the following section.

3.2.2 Generalized VMC inspired by the bio-mechanical model of locomotion

As an extension of the original VMC (Pratt 1996), the generalized VMC can be defined as follows. In the case of locomotion control of dynamic walking robots, a virtual dynamic framework can be assumed and be used for the purpose of designing control system with more simplification and convenience or more accuracy in model similarity in terms of biological mechanism. The control is achieved in a well defined virtual dynamic frame (virtual space) and consequently the force commands are obtained in virtual space. Then the virtual force commands are transformed (or modulated) into actual joint torque commands in the physical dynamics space so that the real robot joints can be actuated.

In a generalized VMC, two parts of dynamics control are always essential: structure dynamics and gait dynamics. In structure dynamics, a proper internal impedance control is needed so as to hold the robot in an appropriate upright posture and to prevent its limbs in a stance phase from collapsing. However, to make the robot walk in a desired way, gait dynamics need to be shaped properly. An additional force command with a modulator is expected to propel the robot walking in the desired pace.

Based on the above bio-mechanical locomotion model, a general VMC (type I) can be proposed as follows. Using the virtual musculo-skeletal structure similar to Taga's locomotion model, determine a proper distributed CPG and the corresponding impedance control. Then use the global state to modulate the motor generator so that the physical joint torque can be obtained.

Figure 3-3 describes a distributed dynamics based virtual model control approach. In this control system diagram, the CPG neural networks are distributed in the legs and the body, which represents the neural dynamics of the spinal cord of animals. The CPG networks are modulated by the global states (global kinematic feedback from skeletal system), and their outputs are sent to the motor control generation module so as to generate the active motor forces. Meanwhile the impedance control is based on the distributed muscle dynamics model. By combining the outputs from the impedance control module and the motor control generation module, the distributed virtual force commands are obtained, then by the transformation module, the actual joint toque commands are determined. This system has kinematic feedback (to the CPG) and the dynamic feedback (to the impedance control).

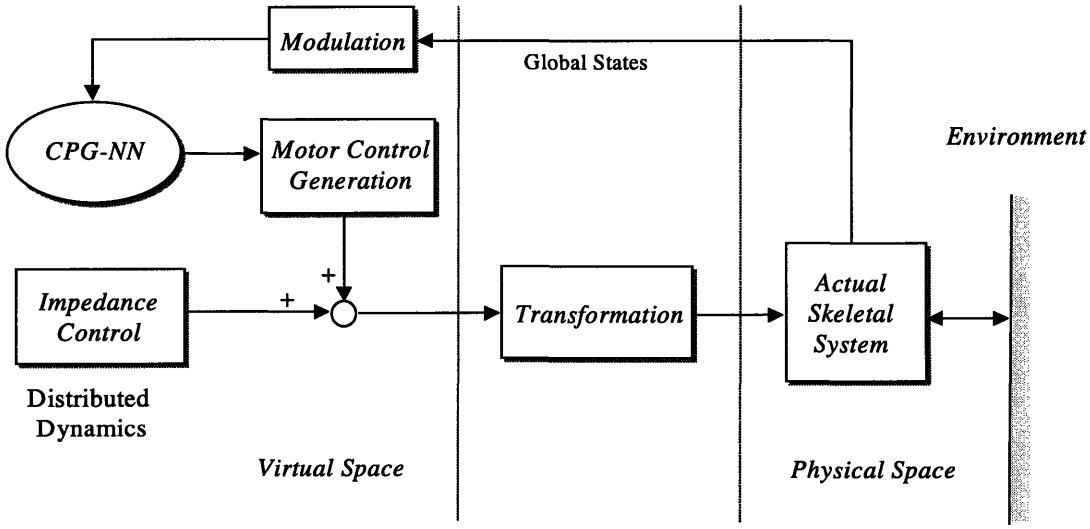


Figure 3-3: Virtual model control with distributed neural dynamic control (Type I model).

3.2.3 Generalized VMC and Centralization

A successful biped walking robot require proper structure dynamics and gait dynamics. To achieve these, an impedance control and a pattern driven or modulated control are essential. There are a number of ways to do this in robotics (which may be different from the biological findings). GVMC type I (Figure 3-3) is one approach. The advantage is that it has the structure similarity of the biological counterpart and more freedom (space) for adjustment and adaptation.

But the disadvantage is that there are too many parameters to be determined and the principle of tuning them and designing them is unclear so far. In other words, close form optimization of those parameters is extremely hard to pursue. In engineering, further simplification is quite necessary. A centralized approach is a right choice because engineers can avoid dealing with parameter redundancy.

To enhance the system robustness and accomplish stable walking, we need to consider using a centralized control approach for suitable biped robot control. There are three requirements to be met: 1) finding the control which can hold the skeletal system without collapsing; 2) interacting with the environments and feeding back the information to the control system and then tuning the control parameters automatically; 3) establishing global walking gait patterns (the cyclic global states). To do so, we extend the control concept of VMC in the following way. Namely, we imagine a proximate virtual dynamics model for the skeletal system of real biped robot in virtual space, and then design the robust control to achieve the above two requirements (1 & 2) in the virtual space.

Inspired by the above bio-mechanical model of locomotion, a centralized control model (VMC type II) for biped robot system is proposed as in Figure 3-4.

The considerations are:

1) For simplicity, the above bio-mechanical model should be modified into a centralized format for practical implementation of a biped robot.
2) In the neural dynamic control system, both neural oscillator and impedance control can be combined into one control module for convenience.
3) Robustness features have to be added to the control module.
4) CPG is used to modulate the motor command generation in a global (macro) way instead of a dynamic way.
5) By using one control module (robust impedance control), the robustness over changing environments (changing terrain) can be well covered by the control.

Generalized Virtual Model Control (Type II)

Figure 3-4 shows the diagram of generalized VMC type II. In this paradigm, the neural oscillators and impedance control are based on the centralized model in virtual space. The neural oscillators take feedback form skeletal system in form of global states, and command the macro CPG (global gait state machine), which modulates the virtual force commands into the actual joint torque commands for actuating the physical system.

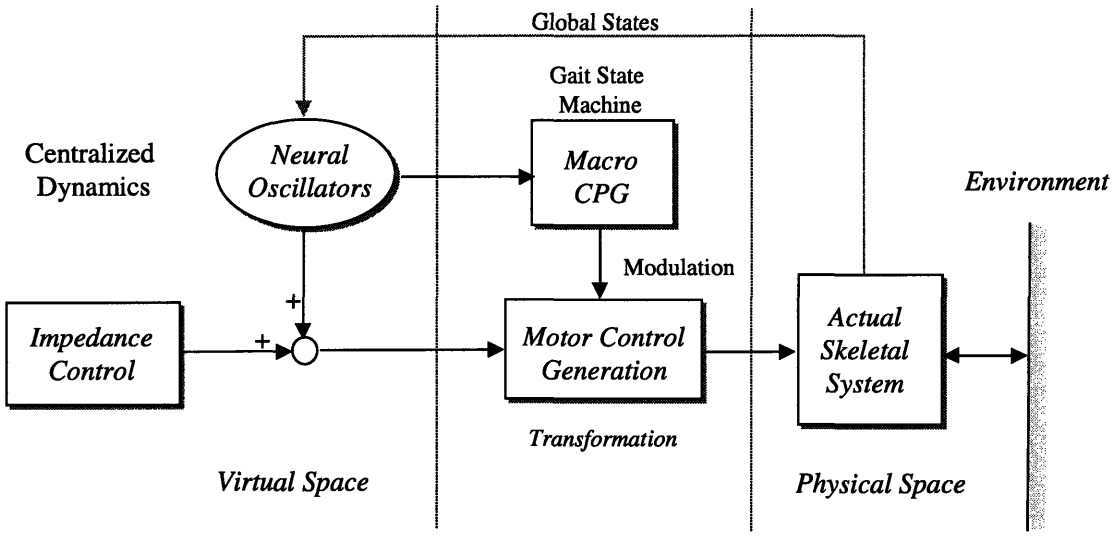


Figure3-4: Virtual model control with centralized dynamics (Type II model).

Generalized VMC (Type III)

Figure 3-5 shows a diagram of a generalized VMC, type III model.

This approach is similar to the centralized control approach type II in Figure 3-4, except that the CPG neural dynamics is directly combined with the impedance control in the virtual dynamics space. The macro CPG (state machine) of VMC in type III is exactly the same to that in type II. But the CPG-NN on top of the macro CPG takes feedback from skeletal system by means of the global states. The controller with virtual components (i.e. springs, dampers etc) can be treated as impedance controller. However, there is a difference in concept technically, that is in the force transformation and Cartesian virtual space control. The virtual impedance control in VMC is conducted in a virtual Cartesian space and the control signals (or force commands output from the controller) are to be transformed into actuator torque commands. The force transformation module functions in the same way as a motor control generation module in the centralized control approach type II with the modulation of the gait state machine. This leads to a lot of advantages in control design. One only needs to choose appropriate virtual components with proper parameters in the virtual Cartesian space to accomplish the VMC design. The key point is the same as the impedance controller in the bio-mechanical model of locomotion (Figure 2-2). The virtual components can be imagined as if one attached some physical impedance components, such as springs and dampers, to the biped robot structure, so that the robot skeletal structure is prevented from collapsing during the stance phase of walking.

It has been shown that VMC can control the biped robot walk stably with success (Pratt 1997, 1998). The issues left over for VMC are: 1) It needs an automatic or intelligent mechanism of tuning the parameters for the chosen virtual components; 2) The capability of robustness is under exploration and is to be enhanced.

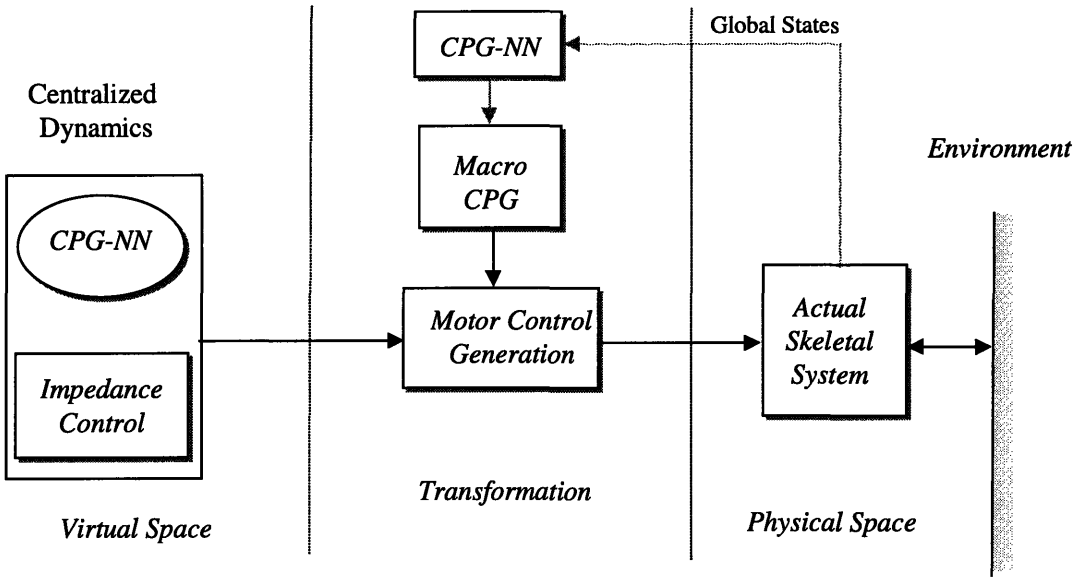


Figure 3-5: Virtual model control with centralized dynamics and global gait.

Generalized VMC (type IV)
Virtual inverted pendulum dynamics based control. In this paradigm, the system structure is the same as type III. But the virtual dynamics is specified in the formulation of an inverted pendulum frame. It is worthwhile to explore this framework and make use of the existing research results, such as Golloday and Hemami's work (1977) and Kajita and Tani's work (1995).

Force Transformation with Gait Modulation v.s. Adaptation
This force transformation module in GVMC is a key point to distinguish the types of GVMC described above. It is used to map the force commands from the virtual space to the physical actuator space. The gait modulation should be incorporated in the transformation. This module functions as the motor control generation in the centralized control approaches type II, type III and type IV. With the gait modulation, we need to derive the mapping transfer relation for each state (or global state) of the gait state machine. One method of doing so is to use the energy conservation (i.e. the Jacobian equation) and the dynamics constraints during the corresponding gait state. But the quality of this approach is a subject to be investigated in the future.

3.3 Robust sliding mode control of a biped robot

3.3.1 Linear virtual dynamics

Automatic tuning of a controller requires a suitable framework or a dynamic model that can reflect the physical interaction between the environment and the system itself. In this section, our robust control design is based on the framework of a virtual dynamics model. Figure 3-6 shows the diagram of a virtual dynamics model based control design (VMC type III). In this diagram,

there is a virtual dynamics space and a physical dynamics space. By utilizing an observation module, the necessary information is collected from the physical space and formulated into the properly selected virtual axes of the virtual space, such that the virtual dynamics of the biped robot can be reconstructed. The virtual control is designed based on the reconstructed virtual dynamics model. The outputs of the controller are the generalized virtual forces that are transformed into the physical torque commands for the actuators by means of the dynamics transformations. The transformations are different in different states, for instance, the single-support states and the double-support state. In this paradigm, the blocks within a dashed-point square is actually fixed in our implementation, but it leaves some space for further enhancement in gait dynamics.

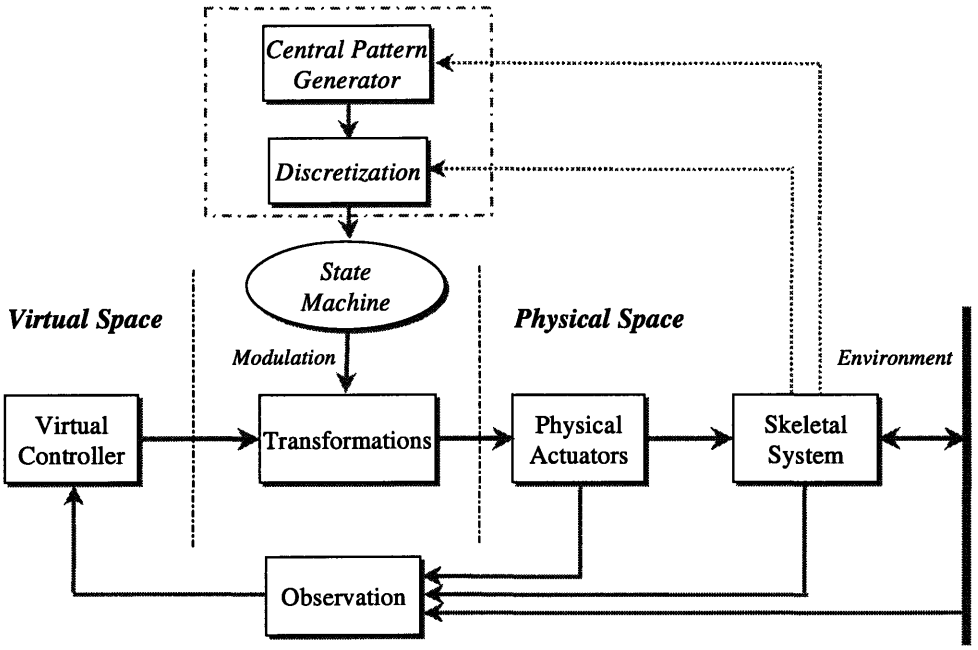


Figure 3-6: A virtual model control implementation paradigm with centralized dynamics (Type III).

In general, the control law of a dynamic system can be formulated as,

$$u = -K_d(\dot{\tilde{x}} + \lambda\tilde{x}) + \Delta u_c \qquad (3\text{-}1)$$

where the control is composed of a linear feedback control part plus a control action correction term Δu_c, $\lambda, K_d > 0$, and $\tilde{x}$ is system tracking error. Here we call Δu_c a learning control term which will be updated on line in an adaptive control system. How to determine Δu_c is the focus of this section. Our approach is to utilize the information observed from physical space and compute Δu_c based on the reconstructed virtual model in virtual space.

The formulation of the virtual dynamics is based on the concept of linearization of dynamics, which says that any nonlinear dynamics equation can be linearized into a locally linear dynamics equation around an operating point (state) and globally the dynamics can be considered as time-

varying linear dynamics. Therefore, in our design, we use the form of a time-varying linear virtual dynamics model and put the error model (unmodelled dynamics) into an error bound for a robustness mechanism to tolerate. For simplicity, in this design, a second order virtual system model is utilized. It is expected that the controller designed in virtual space should be able to take care of the unmodelled dynamics. We choose the adaptive sliding control approach with dead zone to handle this problem.

In our design, the dynamics of the biped legged robot is formulated in z, x, and θ axis of the virtual space. The general form of the virtual model in (z, x, and θ axis) can be written as,

$$a_1\ddot{x} + a_2\dot{x} + a_3x + a_4 + f_x(x,\dot{x},t,...) + d_x = u_x \tag{3-2}$$

$$b_1\ddot{z} + b_2\dot{z} + b_3z + b_4 + f_z(z,\dot{z},t,...) + d_z = u_z \tag{3-3}$$

$$c_1\ddot{\theta} + c_2\dot{\theta} + c_3\theta + c_4 + f_\theta(\theta,\dot{\theta},t,...) + d_\theta = u_\theta \tag{3-4}$$

where x, z, θ are the state variables, u_x, u_z, u_θ are the control commands and $f_x(x,\dot{x},t,...)$, $f_z(z,\dot{z},t,...)$, $f_\theta(\theta,\dot{\theta},t,...)$ are the unmodelled dynamics terms which are unknown functions of the state variable x, $\dot{x}$, z, $\dot{z}$, θ, $\dot{\theta}$ time t, and the variables, d_x, d_z, d_θ are the disturbance terms. The linear crossover terms are not included here in the above equations, but in a general case, they should be present.

3.3.2 Adaptive control design

Using the above virtual dynamics formulation and the framework of virtual dynamics model based control (VMC type III of Figure 3-6), the adaptive controller can be designed in the virtual space by means of adaptive sliding control theory (Slotine 1991). Since the linear dynamics of z, x, and θ axis are in a similar formulation, the general dynamics (3-5) of only one axis is described in the following section.

$$a_1\ddot{x} + a_2\dot{x} + a_3x + a_4 + f(x,\dot{x},t,...) + d = u \tag{3-5}$$

Define the switching variable *s(t)* as,

$$s(t) = \dot{\tilde{x}} + \lambda\tilde{x} = \dot{x} - \dot{x}_r \tag{3-6}$$

where $\tilde{x} = x - x_d$, x_d is the desired trajectory, λ is a strictly positive gain (except $\lambda = 0$ for x axis dynamics).

Note that $\dot{x}_r$ can be computed from the state $(x, \dot{x})$ and the desired trajectory x_d,

$$\dot{x}_r = \dot{x} - s(t) = \dot{x}_d - \lambda(x - x_d) \tag{3-7}$$

According to adaptive control theory (Slotine 1991), we can derive the following control law,

$$u = Y\hat{a} - K_D s \tag{3-8}$$

Choose the adaptation law as,

$$\dot{\hat{a}} = -\Gamma Y^T S_\Delta \tag{3-9}$$

$$S_\Delta = s - \Phi sat(\frac{s}{\Phi}) \tag{3-10}$$

$$sat(x) = \begin{cases} x & |x| < 1 \\ \text{sgn}(x) & else \end{cases} \tag{3-11}$$

where $\Phi = \frac{D}{K_D}$, $|d| + |f(x, \dot{x}, t, ...)| \le D$, D is the upper bound of the disturbance and the unmodelled dynamics. $\hat{a}$ is estimation of the parameter vector a. $\Gamma = diag\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$, ($\gamma_i > 0$) is the adaptation gain matrix.

$$Y = [\ddot{x}_r \quad \dot{x} \quad x \quad 1] \tag{3-12}$$

$$a = [a_1 \quad a_2 \quad a_3 \quad a_4]^T \tag{3-13}$$

By the above control law and adaptation law, it can be guaranteed that the positive semidefinite Lyapunov function candidate

$$V = \frac{a_1}{2} S_\Delta^2 + \frac{1}{2} \tilde{a}^T \Gamma^{-1} \tilde{a} \tag{3-14}$$

where $\tilde{a} = \hat{a} - a$ has a negative semidefinite time derivative. Therefore

$$\dot{V} \le -K_D S_\Delta^2 \tag{3-15}$$

From the above result, we can prove uniform global stability of the system ($S_\Delta \to 0$, $t \to \infty$) by Barbalat's Lemma (Slotine 1991).

The above adaptive control design can be used to design the adaptive control for *x, z, 6* axis. Thus the corresponding controls are the force commands (f_x, f_z, f_θ) generated in the virtual space. Then following the steps in section 3.1, the actuator torques can be obtained by forward dynamics transformations. Referring to the general form of control law (3-1), in this case $\Delta u_c = Y\hat{a}$.

The result in equation (3-15) presents good behavior of asymptotically global stability outside of the sliding boundary layer assuming the given continuous dynamics as in (3-5). In fact, in our application, the equivalent virtual dynamics in the *X*-axis is not a continuous function in terms of the alternate states during bipedal walking. So the performance in *X*-axis dynamics is not guaranteed. This is addressed in the conclusions.

It is worthwhile to mention that the above adaptive control scheme has a nice property, namely robustness, which is achieved by means of boundary layer tolerance. The model error and disturbance are all formulated into a pre-estimated error bound. Then the boundary layer thickness is determined based on it. This implies that by adding better identification mechanisms to the above dynamics framework (Figure 3-6), the performance could be improved further. For example, we can incorporate some nonlinear identification schemes (such as neural networks) to the adaptive control system. Combining a nonlinear identification model such as radial base

function neural networks and the above linear time varying model in (3-5), we can derive the following virtual dynamics equations:

$$a_1\ddot{x}+a_2\dot{x}+a_3x+a_4+\sum_{i=1}^{N}C_ig_i(\bullet)+\varepsilon+d=u \tag{3-16}$$

In (3-16), we have a mixed model with linear and nonlinear part where ε is the model error, $g_i(\bullet)$ is the nonlinear base functions and d is the disturbance term. By using this formulation and proper identification techniques, such as neural networks, the model error can be reduced. In this case (not discussed here), the error bound is smaller and the tracking performance further improved.

3.4 Parameter Adaptation of Linear Virtual Model Controller

When a linear virtual model control approach is assumed, the adaptation is basically done for parameters K_d (rate control) and K_p (position control). Physically, the parameters of virtual spring components and damping components are adapted. A system description is the same as described in section 3.3. To derive the adaptation principle, it is assumed that the same nominal virtual dynamics models as equation (3-5) are used. Similarly to section 3.3, consider only one general axis in the virtual Cartesian space. Define the same switching variable $s(t)$ as equation (3-6). If the Lyapunov function is chosen as (3-14), then the control law and the adaptation law will be in the same form as equations (3-8) and (3-9).
In order to introduce parameter adaptation to the linear virtual components, let

$$K_d=K_d^*+\Delta K_d \tag{3-17}$$

$$K_p=K_p^*+\Delta K_p \tag{3-18}$$

where K_d^* and K_p^* are the optimal values for the virtual components.
Define vectors $\hat{\boldsymbol{a}}$ and Y as follows,

$$\hat{\boldsymbol{a}}=\begin{bmatrix}\hat{a}_1 & \hat{a}_2 & \hat{a}_3 & \hat{a}_4 & \Delta\hat{K}_d & \Delta\hat{K}_p\end{bmatrix}^T \tag{3-19}$$

$$\boldsymbol{Y}=\begin{bmatrix}\ddot{x}_r & \dot{x} & x & 1 & \dot{\tilde{x}} & \tilde{x}\end{bmatrix} \tag{3-20}$$

Then the control law becomes,

$$u=\boldsymbol{Y}\cdot\hat{\boldsymbol{a}}-K_ds=\boldsymbol{Y}\cdot\hat{\boldsymbol{a}}-[(K_d^*+\Delta K_d)\dot{\tilde{x}}+(K_p^*+\Delta K_p)\tilde{x}]$$

$$\Rightarrow\ u=\boldsymbol{Y}_0\cdot\hat{\boldsymbol{a}}_0-[K_d^*\dot{\tilde{x}}+K_p^*\tilde{x}] \tag{3-21}$$

where $\lambda=K_p/K_d$ and

$$Y_0 = [\ddot{x}_r \quad \dot{x} \quad x \quad 1] \tag{3-22}$$

$$\hat{a}_0 = [\hat{a}_1 \quad \hat{a}_{21} \quad \hat{a}_3 \quad \hat{a}_4]^T \tag{3-23}$$

From (3-21), it demonstrates that parameter adaptation of the linear virtual components can be achieved by introducing new variables as (3-17) and (3-18).

3.5 Simulations and analysis

A planar bipedal walking robot was created in simulation. The simulated biped has a mass of approximately 8.0 kg and stands 0.80 m tall. Both virtual model control (Pratt 1996) and Adaptive Virtual Model Control were applied to the biped. During the simulations, external force disturbances were exerted on the biped in different directions to test the control robustness. We observed that the adaptive virtual model controller improved the system's robustness. When an impulse external force was exerted on the robot, the robot was able to maintain stable walking and recovered its continuous motion. Also, the simulations showed that the biped with AVMC could better maintain the desired height of center of mass (CM), the desired body pitch as well as smooth motion in the x-axis. The simulated biped with AVMC can walk indefinitely.

Figure 3-7b shows the simulation results with adaptive VMC, the dynamics of force signals f_x, f_z, f_θ, and forward velocity in x, and actual position in z, θ (i.e. height, pitch) in virtual space. Figure 3-7a shows the planar bipedal robot controlled by the VMC scheme. In Figure 3-7a & 7b, the responses are robot height (Z), pitch (Theta, θ), forward velocity (X-dot), as well as virtual force commands generated by a controller, such as f_z (f-z), f_x (f-x) and f_θ (f-t). Comparing the results under VMC and AVMC, we can see that AVMC can improve the dynamic tracking performance of height and pitch, but it can't help much in forward velocity control because the controller can only function in the double support state. In single support states, because of the dynamics constraints, the controller of X-axis is disconnected by the dynamics transformations. This implies that the virtual dynamics in X-axis is not a continuous function. Therefore the adaptive control can not really achieve a desired performance in the forward speed control (of X-axis), This could be further improved by a gait control scheme. Figure 3-8 shows the parameter identification of the virtual linear dynamics model by the adaptation mechanism.

In the test of robustness with external disturbances, we did an external force impact test in our simulation. Figure 3-9 shows the simulation responses of a bipedal walking robot experiencing an external force impact (10 Newtons) in the z-direction. Figure 3-10 shows the stick plot diagram of this walking profile with a force impact. From the above results, it has been shown that improved robustness can be achieved by means of the adaptive VMC scheme. The robustness of the biped with changing terrain will be tested in our future research.

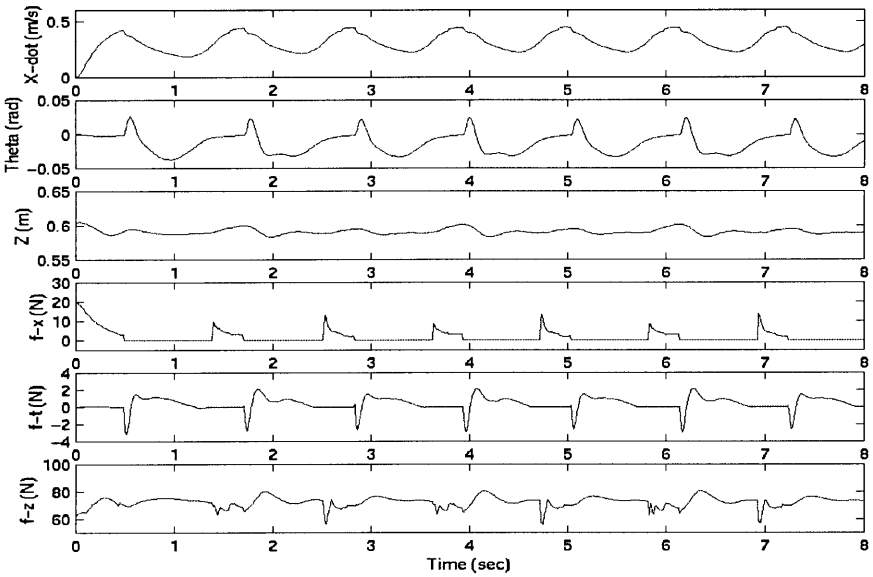


Figure 3-7a: Simulation results of a bipedal robot with VMC. X-dot, Theta, and Z are responses of the forward velocity, pitch angle and robot body height respectively. f-x, f-t, f-z are the virtual force commands generated by the VMC.

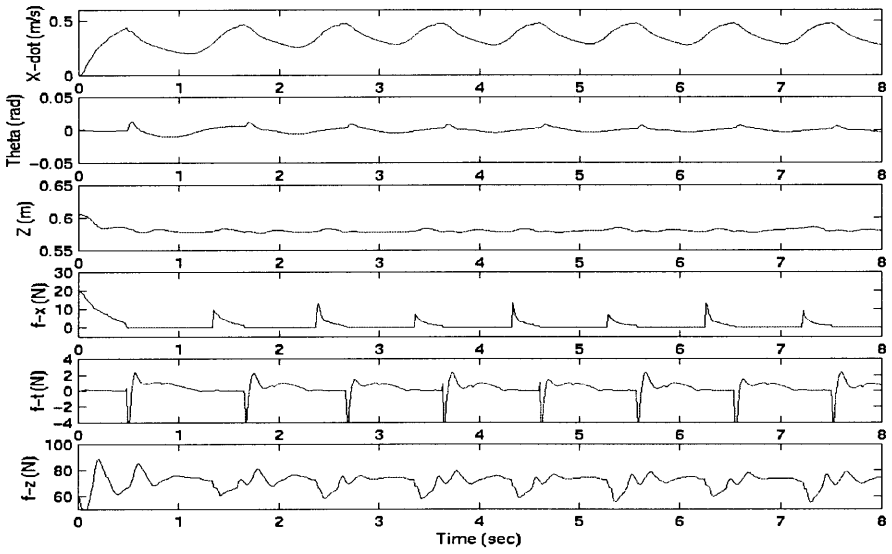


Figure 3-7b: Simulation results of a bipedal robot with Adaptive VMC. X-dot, Theta, and Z are responses of the forward velocity, pitch angle and robot body height respectively, f-x, f-t, f-z are the virtual force commands generated by the Adaptive VMC.

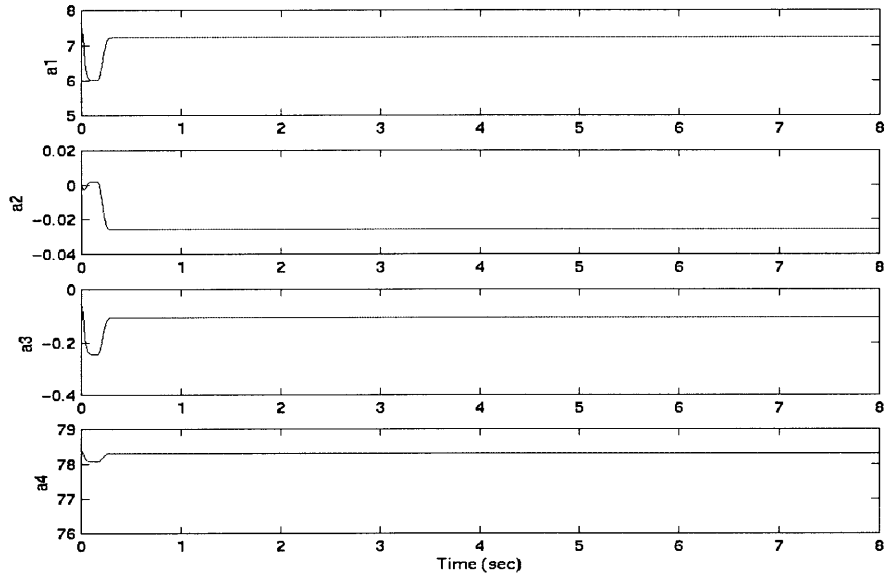


Figure 3-8: Parameter identification of the linear virtual dynamics model in z-axis.

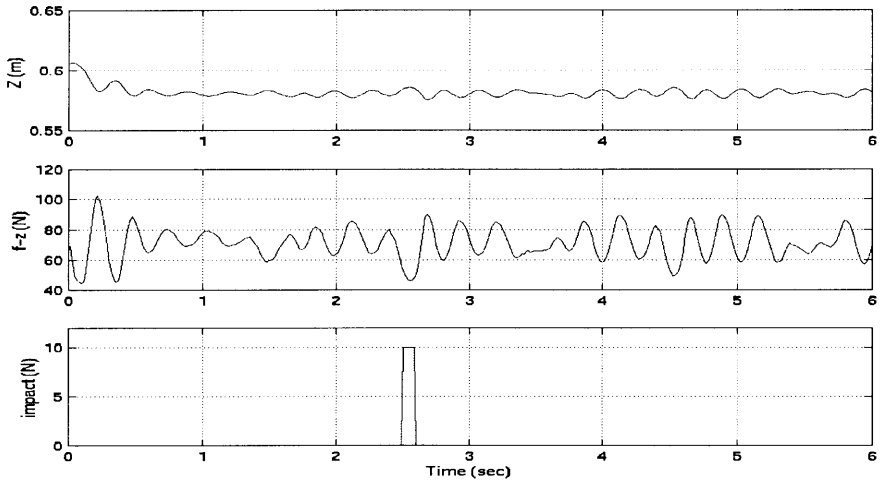


Figure 3-9: Simulation results of a bipedal walking robot with external impact (10 Newtons) in z direction. Z and f-z are the robot height and the virtual force command in z axis respectively.

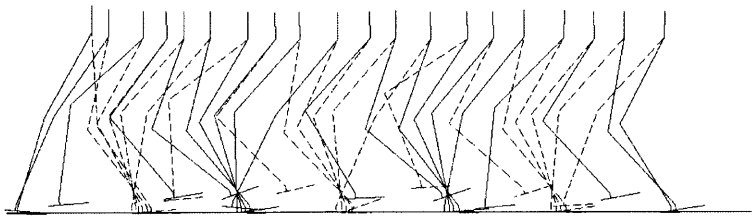


Figure 3-10: Stickplot of a bipedal walking robot experiencing external force impact (10 Newtons) in z direction.

Chapter 4

Neural Networks and Learning Control

Artificial neural networks can be used as a representation framework for modelling nonlinear dynamical systems. It is also possible to incorporate these nonlinear models within nonlinear feedback control structures (or model based control structures). Three different types of neural networks and their corresponding training algorithms are described in this chapter. These neural network models are suitable for modelling and control of nonlinear dynamical systems, which is demonstrated in the following chapters.

4.1 Radial basis function (RBF) neural networks

Recently, there has been a great deal of interest in Radial Basis Functions (RBF) Neural Networks within the engineering community. RBFs are a traditional and powerful technique for function interpolation and approximation in multidimensional space (Haykin 1994). A generalized form of RBF neural networks has found wide applications in areas such as, image processing, signal processing, control engineering, etc. Due to the properties of rapid training, network structure simplicity and function locality, the RBF networks have been given many diverse names, such as Gaussian Potential Functions (Lee & Rhee, 1991), Localized Receptive Fields (Moody & Darken, 1988), Regularization Networks (Poggio & Girosi, 1989), Locally Tuned Processing Units (Moody & Darken, 1989), and Gaussian Neural Networks (Sanner & Slotine, 1991).
One of the important advantages in RBF neural networks is the generality. It has been proven (Girosi & Poggio, 1990, Hartman & Keeler, 1991) that they are universal approximators, that is, given a network with enough hidden layer neurons, they can approximate any continuous function with arbitrary accuracy. This property is also shared by other feedforward multi-layer perceptrons (addressed in next section).

4.1.1 Neuron model

Similarly to the standard artificial neuron model, a radial basis function neuron (Figure 4-1) has multiple inputs, one output, a threshold vector θ (called the center of the base function), and nonlinear neuron state function $h(\bullet)$, which is defined as a radial function, such as a Gaussian function.

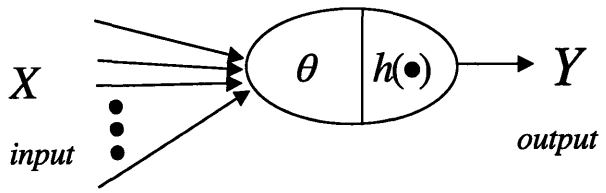


Figure 4-1: Neuron model for RBF network.

$$h(\vec{x}) = \exp\left(\frac{-\left\|\vec{x}-\vec{\theta}\right\|_2}{\sigma^2}\right) \qquad (4\text{-}1)$$

where σ is the width or standard deviation of the Gaussian basis function.

4.1.2 Radial basis function neural networks structure

Figure 4-2 shows the structure of a radial basis neural network and its supervised learning (or training) scheme. This is a three layer neural network structure. The first layer is the input field for measuring the locality. The second layer net is a hidden layer, where the Gaussian radial basis functions are distributed in the input field with different centers (threshold vectors). The third layer net is an output layer (usually $v = 0$). The training scheme in the RBF neural networks is in a supervised learning mode traditionally. In this framework shown here the network is supposed to learn the given data patterns from an unknown system. The training algorithms used in this research are recursive least mean square (RLMS) approach and general least mean square (LMS) approach. Comparing the plant output with the neural networks predication, the difference (or error) vector can be obtained and then fed into the RLMS module (or LMS module) to compute the weight increment, hence the weights are updated.

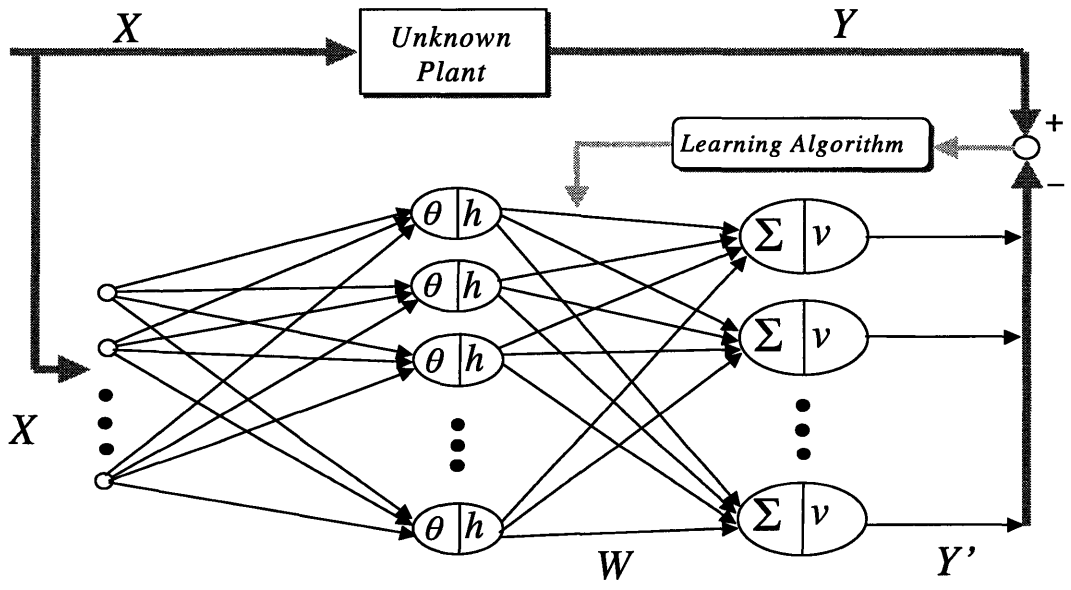


Figure 4-2: Radial basis neural network with supervised learning.

4.1.3 Approximation theory and learning algorithms

Considering n-to-1 mapping as described below in Figure 4-3. Assume that the threshold in output layer is zero.

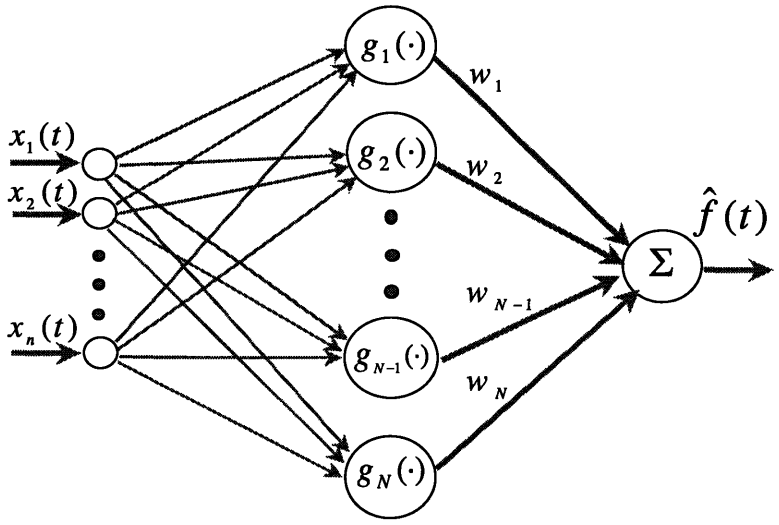


Figure 4-3: General n-to-1 mapping with RBF network

By using this network structure, one can easily extend to a n-to-m mapping, i.e. multiple input multiple output cases. Since the output layer is a linear function of $\{g_i\}$, multiple outputs can share the same hidden layer and one only needs to add linear components in the network.

A. Approximation theory (Poggio & Girosi 1989, Hartman & Keeler 1990).
   1) The RBF neural network is a universal approximator in that it can approximate arbitrarily well any multivariate continuous function on a compact subset of R^p, given a sufficiently large number of hidden neurons.
   2) Since the approximation scheme is linear in the unknown coefficients, it follows that the RBF network has the best approximation property. This means that given an unknown nonlinear function *f*, there always exists a choice of coefficients that approximates *f* better than all other possible choices.
   3) The solution computed by the regulation network (Poggio and Girosi, 1989) is optimal. Optimality here means that the regularization network minimizes a function that measures how much the solution deviates from its true value as represented by the training data.

B. Off-line supervised learning algorithm.

In RBF neural networks, output y is a linear combination of the radial basis functions. If the centers $\{\mu_i, \sigma_i\}$ are known to us, the learning process can be reduced to a linear system problem for determining the linear weights w_i, $(i = 1,2,\cdots,N)$.

To train the neural networks off-line, we can formulate our task as a least mean square (LMS) optimization process. Given sample data sets $\{\underline{x}^i, y^i\}$, $(i = 1,2,\cdots,S)$, obtain weights $\{w_1, w_2, \cdots, w_n\}$, which will minimize the approximation error index function (or cost function) *J*,

$$J = \tfrac{1}{2}\sum_{i=1}^{S}[y^i - \hat{y}^i]^2 \qquad (4\text{-}2)$$

where $\hat{y}^i = \sum_{j=1}^{N} w_j \exp(\frac{-\left\| \underline{x}^i - \underline{\mu}_j \right\|^2}{\sigma_j^2})$ (4-3)

is the output prediction of the network for the corresponding input vector $\underline{x}^i$. Here $\underline{\mu}_j$ is a distributed center.

Define $\boldsymbol{H} = \{H_{ji}\}_{N \times S}$, (4-4)

and $H_{ji} = \exp(\frac{-\left\| \underline{x}^i - \underline{\mu}_j \right\|^2}{\sigma_j^2})$, (4-5)

$$\boldsymbol{W} = [w_1, w_2, \cdots, w_N]^T, \tag{4-6}$$

$$\boldsymbol{Y} = [y^1, y^2, \cdots, y^S]^T. \tag{4-7}$$

Applying the LMS optimization method (described in Appendix A), we can obtain the optimal solution for the weights $\boldsymbol{W}^*$ in a least mean square sense.

$$\boldsymbol{W}^* = \boldsymbol{H}^{\#} \cdot \boldsymbol{Y} \tag{4-8}$$

where $\boldsymbol{H}^{\#} = [\boldsymbol{H}\boldsymbol{H}^T]^{-1}\boldsymbol{H}$ is the pseudo-inverse of matrix $\boldsymbol{H}$.

C. On-line supervised learning algorithm RLMS.

The weights $\boldsymbol{W}$ can be tuned recursively. In step k, given the kth sample data set $\{\underline{x}[k], y[k]\}$, the on-line recursive learning algorithm can be implemented as follows,

$$\Delta \boldsymbol{W}[k] = \boldsymbol{R}[k]\boldsymbol{h}[k]^T \delta[k] \tag{4-9}$$

$$\delta[k] = y[k] - \boldsymbol{h}[k]^T \boldsymbol{W}[k-1] \tag{4-10}$$

$$\boldsymbol{R}[k] = \boldsymbol{R}[k-1] - \frac{\boldsymbol{R}[k-1]\boldsymbol{h}[k]\boldsymbol{h}[k]^T \boldsymbol{R}[k-1]}{1 + \boldsymbol{h}[k]^T \boldsymbol{R}[k-1]\boldsymbol{h}[k]} \tag{4-11}$$

$$\boldsymbol{h}[k] = \begin{bmatrix} h_1(\underline{x}[k]) \\ h_2(\underline{x}[k]) \\ \vdots \\ h_N(\underline{x}[k]) \end{bmatrix}_{N \times 1} \tag{4-12}$$

$$\boldsymbol{W}[k] = \boldsymbol{W}[k-1] + \Delta \boldsymbol{W}[k] \tag{4-13}$$

Initial values: $\boldsymbol{R}[0] = g \cdot \boldsymbol{I}_{N \times N}$, where g can be chosen as a big positive number, e.g. 10^6. Centers $\{\underline{\mu}_i\}$ can be selected evenly.

D. On-line supervisory learning algorithm (the Delta rule).

$$\Delta \boldsymbol{W}[k] = -\boldsymbol{\Gamma} \cdot \boldsymbol{h}[k] \cdot \delta[k] \tag{4-14}$$

$$\boldsymbol{\Gamma} = diag\{\gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_N\} \tag{4-15}$$

where $1 > \gamma_i > 0$, $i = 1,2,\cdots,N$ is the learning rate.

4.2 Multi-layer Perceptron and Back Propagation (BP) Algorithm

The multi-layer perceptron trained by the back propagation learning algorithm is one of the most widely used and very important neural networks models. This model stimulated the interest in neural networks in the mid-1980s (Rumelhart, Hinton & William 1986). It can be shown that it is a universal approximator, which means that is can approximate any continuous real valued functions to any given degree of accuracy (Funahashi 1989, Necht-Neilson 1989). Generally, multi-layer perceptrons with back propagation are trained in a supervised learning paradigm.

4.2.1 Neuron model in multi-layer perceptron

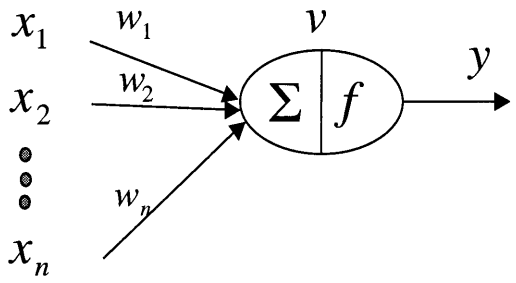


Figure 4-4: Neuron model for multi-layer feedforward networks

$$v = \sum_{i=1}^{n} x_i w_i \tag{4-16}$$

$$y = f(v) \tag{4-17}$$

where $f(v)$ should be nonlinear differentiable function. It can be selected as a sigmoidal function, or called *S*-shape function. Usually,

$$f(v) = \frac{1}{1+e^{-\lambda v}}, \ (\lambda > 0) \tag{4-18}$$

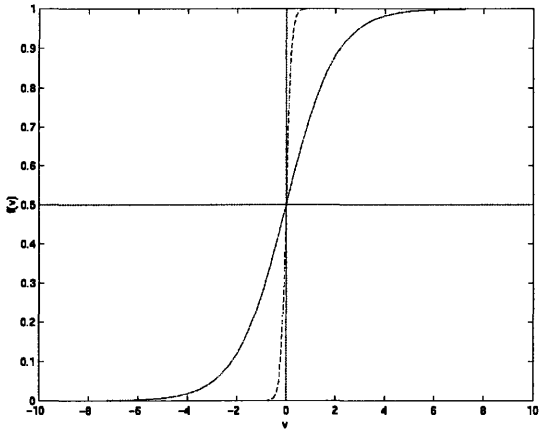


Figure 4-5: *S*-shape nonlinear function. (a) figure with solid line $\lambda = 1$; (b) figure with dash line, $\lambda = 10$.

4.2.2 Multi-layer feed forward perceptron structure.

Figure 4-6 shows the general structure of the multi-layer feed forward neural networks. In this structure, all the signals are feed forward, and the neurons of two adjacent layers are fully connected with each other in forward direction. To have learning capability, this neural network must have nonlinear function at each internal neuron. It has been proved that a multi-layer sigmoidal neural network can represent an arbitrary (measurable) function to an arbitrary accuracy on a compact data set (Funahashi 1989).

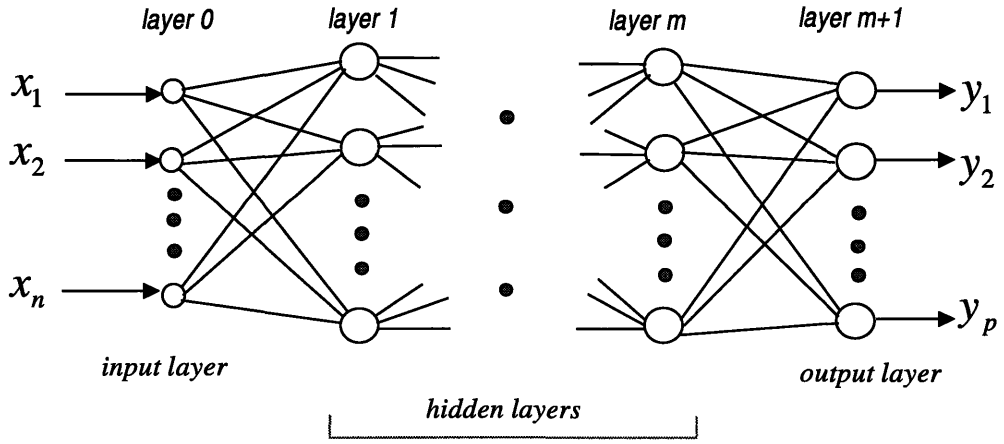


Figure 4-6: General structure of multi-layer feedforward neural networks

4.2.3 Back propagation (BP) learning algorithm.

This is a supervised learning algorithm. The BP algorithm is derived in Appendix B. It can be summarized as follows (for a N layer neural network),

$$\Delta W_{ji}^{m} = \eta \delta_i^m i_i^m \tag{4-19}$$

When $m = N$ (in the output layer),

$$\delta_j^N = (t_j - O_j^N) f'(v_j^N) \tag{4-20}$$

When $1 \le m \le N-1$ (in the hidden layers),

$$\delta_j^m = \sum_k \delta_k^{m+1} W_{kj}^{m+1} f'(v_j^m) \tag{4-21}$$

where m is a layer number.

The signal flow in this network is in a manner of parallel-distributed information processing. At first, the input signals go through the network forward, and then the network output is computed. Comparing the network output with the desired output from the sample data set, the training error of the network is obtained. Second, the error message propagates from the output layer backward and go through the hidden layers down to the input layer. Then the errors at all the neurons can be calculated. Finally, using the weight updating formula to compute the new weights in the network. In this algorithm, in order to update the weights, the error message has to propagate backward through the multi-layer network, thus the name of this training algorithm, Back Propagation Algorithm.

4.2.4 Applications of multi-layer feed forward neural networks with BP algorithm.

There are many successful application examples of BP algorithm, for instance, applications in dynamic control, pattern recognition, data clustering, signal processing etc. Particularly, in dynamic control, the BP network is widely used to learn the inverse dynamics or the dynamic data mapping with off-line unsupervised learning scheme. Literatures can be found in (Wu 1994, Narendra & Parthasarathy 1990).

4.3 CMAC Neural Networks

The CMAC network, abbreviated for cerebellar model articulation controller, or cerebellar model arithmetic computer, was developed by Albus (Albus 1975, 1979). It is one of the neural network models that have drawn a lot of attentions and been widely applied in control areas. Fast training process, good function approximation and hardware representation are the important characteristics of this model. Many successful applications can be found in the literatures (Miller 1989, 1990, Handelman 1989, Kraft & Campagna 1989, An 1994, Xu 1994).

4.3.1 CMAC structure

In Figure 4-7, S is the input state space, the sub-vectors s_i in it are input vectors of dimension n. A is an N dimensional memory. Each s in S is mapped to C locations in A. In general, the theoretical size (number of memory locations) of A is unpractically large, while the memory requirement for a typical control problem is much smaller. For this reason, the large memory A is randomly mapped into a smaller memory A', but there are still C locations in A' which corresponds to each point in S. The values stored at these locations are summed to produce the CMAC output $f(S)$. As a whole, CMAC acts as an arbitrary function $f(\cdot)$ such that,

$$u = f(S) \tag{4-22}$$

where S is the input to CMAC and u is the output of CMAC. A CMAC module has a number of characteristics. Because of its input mapping, it has local input generalisation, that is, similar inputs will produce similar outputs. A large CMAC network can be used and trained in practical time (on-line). Due to the learning rule used in CMAC, it has a unique global minimum in a least mean square optimization sense. Finally, CMAC can learn a wide variety of functions.

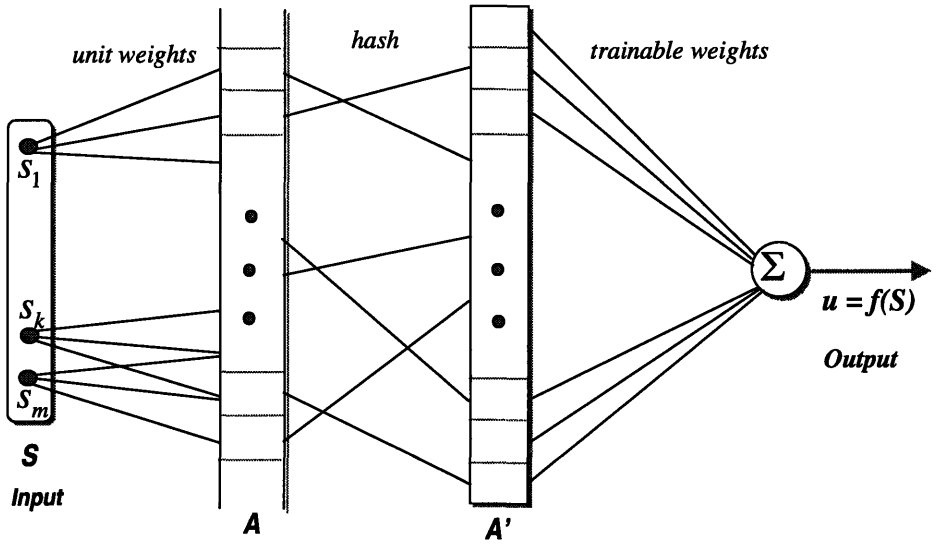


Figure 4-7: General diagram of CMAC neural network

4.3.2 Learning algorithm

The training of CMAC can be conducted as follows:

1) Assume that $f(\cdot)$ is a function for CMAC to learn. Then $u = f(\cdot)$ is the desired CMAC output;
2) Get an input-output training pair $u = f(s)$. Compute the current value (prediction) $\hat{u} = f(s)$;

3) If $|u-\hat{u}| \leq \xi$, where ξ is an acceptable error-margin $(\xi > 0)$, then do nothing. If $|u-\hat{u}| > \xi$, then add to every connection weight that contributed to u the quantity

$$\Delta W = \alpha \cdot \frac{(u-\hat{u})}{|A^*|} \tag{4-23}$$

where $|A^*|$ = the number of weights from A' that contributed to u; α is the learning rate, $(0 < \alpha < 1)$.

4) Select another input point, repeat the above procedure until all the input points in the sample data sets have been processed.

4.3.3 Applications in control

There are two schemes basically for using CMAC in control. The first scheme has the configuration depicted in Figure 4-8 (Albus 1975), which is actually a standard supervised learning control paradigm (refer to Figure 4-11 of section 4.5). This scheme is for learning a controller from an expert such as a human or other available controller.

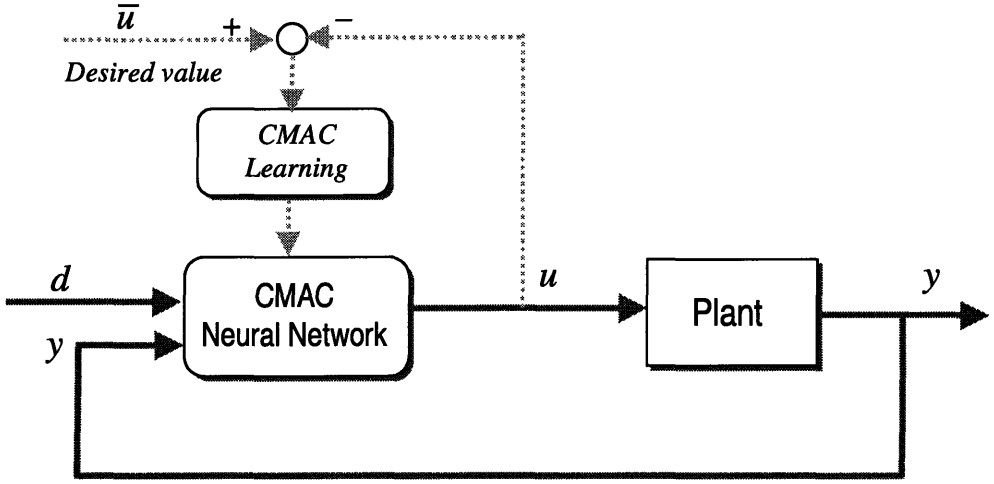


Figure 4-8: CMAC control paradigm with supervised learning.

Figure 4-8 shows that, in the control system, both the command and feedback signals are used as inputs to the CMAC controller. The output of the controller is fed directly to the plant. The desired output of the neural network controller has to be supplied, thus it is in supervised learning mode. The training of the controller is based on the error between the desired and actual CMAC controller output. Two stages are needed to make the system work. The first stage is training the neural controller. When CMAC receives the command and feedback signals, it produces an output. This output is compared with the desired output. If there are differences between them, then the weights are adjusted to eliminate the differences. On completion of this stage, CMAC network has learnt how to produce a suitable output to control the plant according to the given command and the measured feedback signals. The second stage is control in action. CMAC network can work well when the required control is close to that with which it has been

trained. Both stages are completed without the need to analyze the dynamics of the plant and to solve complex equations. However, in the training stage, this scheme requires the desired plant inputs to be known.

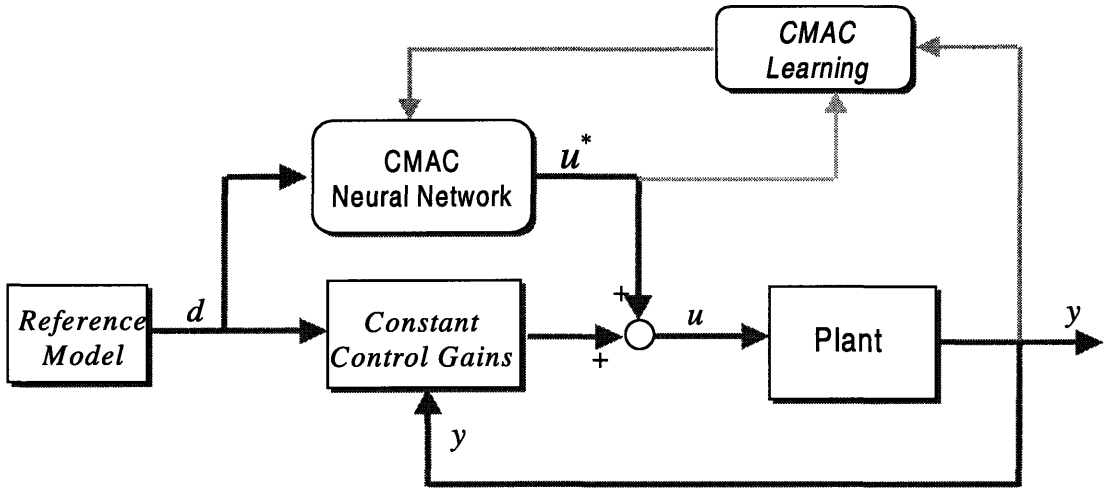


Figure 4-9: CMAC control paradigm with unsupervised learning.

The second control scheme is illustrated in Figure 4-9 (Miller, 1987,1989). In this scheme, the reference output block produces a desired output at each control cycle. The desired output is sent to the CMAC module that provides a signal to supplement the control signal from a fixed gain conventional error feedback controller. At the end of each control cycle, a training step is executed. The observed plant output during the previous control cycle is used as input to the CMAC module. The difference between the computed plant input u^* and the actual input u is used to compute the weight adjustment. As CMAC is trained continuously following successive control cycles, the CMAC function forms an approximation of the plant inverse dynamics over particular regions of the input space. If the future desired outputs are in regions similar to previous observed outputs, the CMAC output will be similar to the actual plant input required. As a result, the output errors will be small and CMAC will take over from the fixed-gain conventional controller (the original teacher).

From the above description, Scheme 1 is a purely closed-loop control system, because besides the command variables, the feedback variables are used as inputs to the CMAC module to be encoded so that any variations in the plant output can cause variations in the input it receives. In scheme 1, the adjustment of weights is based on the error between the desired controller output $\overline{u}$ and the actual controller output u, rather than the error between desired plant output and actual plant output. As already mentioned, this requires the designer to assign the desired controller output and will cause problems because usually only the desired plant output is known to the designer. The training in scheme 1 can be considered to be the identification of a proper feedback controller. In scheme 2, the CMAC module is used for learning an inverse dynamics with the assistance of a conventional fixed-gain feedback controller. After training, the CAMC network will become the principal controller. In this scheme, control and learning proceed at the same time. The disadvantage of this scheme is that it requires a fixed-gain controller to be designed for the plant in advance.

4.4 Survey on learning process

Among the many interesting properties of a neural network, learning from its environment is the most significant ability of a neural network. A neural network can improve its performance through learning. Such a learning process takes place by means of adjusting its synaptic weights and thresholds. Ideally, the neural network becomes more knowledgeable about its environment after each iteration in the learning process.
In the context of neural networks, learning is generally defined as follows:

Learning is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place.

There are several different types of learning processes that have been studied over the years. In this section, a brief survey on the learning processes is addressed as follows.

4.4.1 Delta Learning Rule

This is actually a kind of error-correction learning. Assume the neurons in the output layer of a neural network are linear. Let $y_k^d(n)$ denote the desired response or target response for neuron k at time n, and the actual response of this neuron be denoted by $y_k(n)$. The error signal is $e_k(n) = y_k^d(n) - y_k(n)$. The ultimate purpose of the Delta learning rule is to minimize an error signal based cost function,

$$J = \frac{1}{2}\sum_{n=1}^{N}\{\sum_{k} e_k^2(n)\} \tag{4-24}$$

An approximate solution to the above optimization problem can be obtained in a least mean square sense. According to the delta learning rule, J can be minimized with respect to the synaptic weights of the network. The adjustment of the synaptic weight w_{kj} at time n is given by,

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{4-25}$$

where η is a positive constant that determines the rate of learning and $x_j(n)$ is the input from the previous layer. In other words, the adjustment made to a synaptic weight is proportional to the product of the error signal and the input signal through the synapse. A unique minimum can be achieved by this learning rule for the neural networks that have a linear output layer.

4.4.2 Supervised Learning

The essential feature of supervised learning is the availability of an external teacher or supervisor. In conceptual terms, we may think of the teacher as having knowledge of the environment that is represented by a set of input-output data or examples. The environment is, however, unknown to the neural network of interest. Suppose now that the teacher and the neural network are both exposed to a training vector (or example) drawn from the environment. By virtue of built-in knowledge, the teacher is able to provide the neural network with a desired or

target response for that training vector. Indeed, the desired response represents the optimum action to be performed by the neural network. The network parameters are adjusted under the combined influence between the training vector and the error signal, i.e. the difference between the actual response of the network and the desired response.

This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network emulate the teacher in such a way that the emulation is optimized in a statistical sense. In other words, knowledge of the environment available to the teacher is transferred to the neural network as fully as possible. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment thereafter completely by itself. Generally, the supervised learning is virtually formulated in a error-correction fashion, which is described in the above section. The least-mean-square (LMS) algorithm and the back-propagation (BP) algorithm are two widely applied supervised learning algorithms.

4.4.3 Unsupervised Learning

In unsupervised learning, or self-organizing learning, there is no external teacher to oversee the learning process. In other words, there are no specific data sample sets or examples of function available to be learned by the network. Rather, provision is made for a task-independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input and thereby create new classes automatically (Becker, 1991). Hebbian learning and competitive learning are two types of unsupervised learning approaches, which have neurobiological basis (Haykin, 1994).

Unsupervised learning is a good choice for dealing with unknown or partially unknown system dynamics. In reality, combining the supervised learning and unsupervised learning can help to utilize the available knowledge or experiences gained so far and explore the further unknown aspects. One successful example in control engineering is model reference adaptive control. In model reference adaptive control system, a reference model works as a teacher and the adaptation mechanism adjusts the control system gains such that the entire system (controller + plant) behaves as the reference model and the Lyapunov stability is guaranteed in the gain tuning process.

4.4.4 Reinforcement Learning

Reinforcement learning is the on-line learning of an input-output mapping through a process of trial and error designed to maximize a scalar performance index function called a reinforcement signal. The basic reinforcement learning has its original roots in experimental studies of animal learning in psychology. But in Sutton's reinforcement learning paradigm, the definition of reinforcement learning has been stated as follows,

If an action taken by a learning system is followed by a satisfactory state of affairs, then the tendency of the system to produce that particular action is strengthened or reinforced. Otherwise, the tendency of the system to produce that action is weakened.

One of well-developed reinforcement learning framework is Adaptive Heuristic Critic (Barto et al 1983, Sutton 1984). Its theory has several artificial intelligence components and is closely linked with optimal control and dynamic programming techniques in the implementation. The

existing dynamic programming techniques provide the learning process an efficient and powerful mechanism for sequential-decision making and global optimization of the index function.

Reinforcement learning can be viewed as an unsupervised learning if one considers the index function as an evaluative feedback component imbedded in the system. Or it can be viewed as a supervised learning when one treats the index function as an evaluative teacher. But this approach can deal with both statistically stationary and non-stationary systems.

4.5 Adaptive learning with neural networks

As we noticed, the spatiotemporal nature of learning is exemplified by many of the learning tasks (e.g., control and identification) discussed in the previous chapters. Moreover, animals ranging from insects to humans have an inherent capacity to represent the temporal structure of experience. Such a representation makes it possible for an animal to adapt its behavior to the temporal structure of an event in its behavioral space (Gallistel 1990). Temporal representation of experience and adaptation are the key in natural learning.

It is known that both adaptation and learning can be achieved by means of neural networks. There are different definitions of adaptation in control engineering, cognitive science and artificial intelligence. Generally, adaptation means that making appropriate adjustments to the networks such that the overall system can achieve an optimal behavior under unknown and unexpected situations occurred in the environment. It is often assumed that a nominal network can make the system work properly during the nominal situations. However, learning can be classified mainly as supervised learning and unsupervised learning based on the presence of teacher.

When a neural network operates in a stationary environment (i.e., one with statistical characteristics that do not change with time), the essential statistics of the environment can in theory be learned by the network under supervision of a teacher. In particular, the synaptic weights of the network can be computed by having the network undergo a training session with a set of data that is representative of the environment. Once the training process is completed, the network should capture the underlying statistical structure of the environment. Thus the supervised learning is done. In this case, adaptation is not necessary.

Frequently, however, the environment of interest is non-stationary, which means that the statistical parameters of the information-bearing signals generated by the environment vary with time. In the situation of this kind, the traditional methods of supervised learning may prove to be inadequate, because the supervisor available may not have the sufficient knowledge to capture the features of the time varying environment. To overcome this shortcoming, the network has to be able continually to adapt its free parameters to variations in the incoming signals in a real-time fashion. Thus an adaptive system or an unsupervised learning system is required.

In a time varying or an uncertain environment, a nominal network or any available supervisor may not be able to make the system work in a satisfactory way. In most cases of this kind, at least the criterion or the index function for judging the system behaviors should be found and a general principle of operation like optimization is also available. Based on the chosen criterion or index, an adaptive mechanism or unsupervised learning method can be developed. Yet, there is no such a unique theory existing for the development because the nature of the environment is not unique. The nature of uncertainty in the environment decides the features of the possible adaptation approaches that can be applied.

Among many successful examples, there are two types of adaptation or unsupervised learning, or so called adaptive learning. One is Lyapunov stability theory based learning principle. The other is reinforcement learning method, which is similar to human learning

activity. The former has a lot application in dynamic control systems. Other approaches, such as neurobiological mechanism based self-organizing learning approaches, have been applied successfully in pattern recognition and dynamic systems etc, where the Hebbian learning rule and competitive learning rule are used in the learning process. Important examples of this types of neural networks are Kohonen's *Self-organizing Maps* (Kohonen 1997) and Grossberg's *Adaptive Resonance Theory (ART)* (Carpenter & Grossberg 1988). For practical applications, combining the supervised learning and adaptive learning is strongly suggested. Model reference adaptation, which combines a supervisor (reference model) and an adaptation module, is not only a good dynamic control approach, but also a good neural network adaptive learning approach. Many successful application examples are formulated in this framework. In this study, Lyapunov stability based adaptive learning is focused.

4.6 Learning control of an inverted pendulum model

The inverted pendulum model has been widely used as a benchmark for verifying the feasibility of many learning control and nonlinear controllers. That is because an inverted pendulum is essentially a nonlinear and unstable system. A bipedal walking robot can be simplified as an inverted pendulum based dynamic walking, which was reported in many literatures (Golliday & Hemani 1977). A dynamic model of an inverted pendulum may be helpful to analyze the gait stability with respect to a stance length although the biped in single support phase is different from an inverted pendulum pivoted on a cart.

The following neural network learning control paradigms are standard frameworks that can be applied similarly in walking robot control systems.

4.6.1 Inverted pendulum model

As derived in Appendix C, the dynamic model can be formulated as follows,

$$(M+m)\ddot{x} + ml\cos\theta\cdot\ddot{\theta} - ml\sin\theta\cdot\dot{\theta}^2 = f \tag{4-26}$$

$$ml^2\ddot{\theta} + ml\cos\theta\cdot\ddot{x} - mgl\sin\theta = \tau \tag{4-27}$$

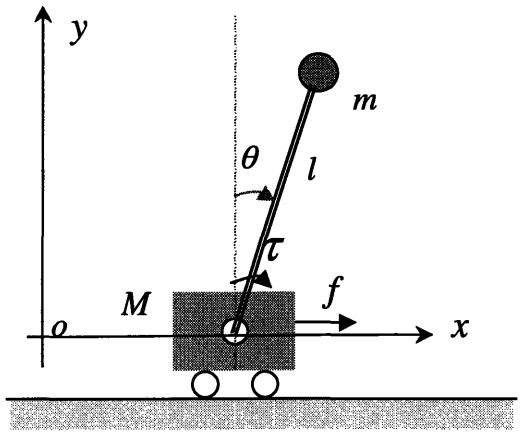


Figure 4-10: Diagram of an inverted pendulum.

4.6.2 Supervised learning control of the inverted pendulum

In the study of supervised learning control, a linear controller is designed by means of the linearization of the given nonlinear inverted pendulum model. Then the inverted pendulum controlled with linear controller is simulated. In order to train the neural networks, the linear controller is utilized as a teacher or supervisor. The supervised learning control scheme is shown in Figure 4-11. Two neural networks, CMAC and RBF neural networks, have been tested. Because of long training time is required, the multi-layer feedforward neural network with back-propagation algorithm is not applied in this inverted pendulum control.

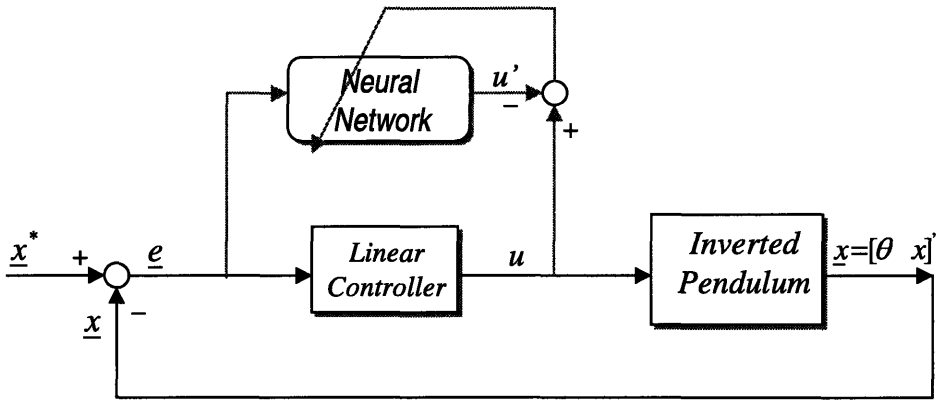


Figure 4-11: Neural network control paradigm with supervised learning

4.6.3 Unsupervised control of the inverted pendulum

For unsupervised learning control, an adaptation scheme is developed, which will be described in details in chapter 6. Both CMAC and RBF neural networks can be used for unsupervised learning control of the given inverted pendulum model, where a linearized inverted pendulum model is derived and the neural networks are used to take care of the nonlinear dynamics part. Figure 4-12 shows the unsupervised learning control scheme.

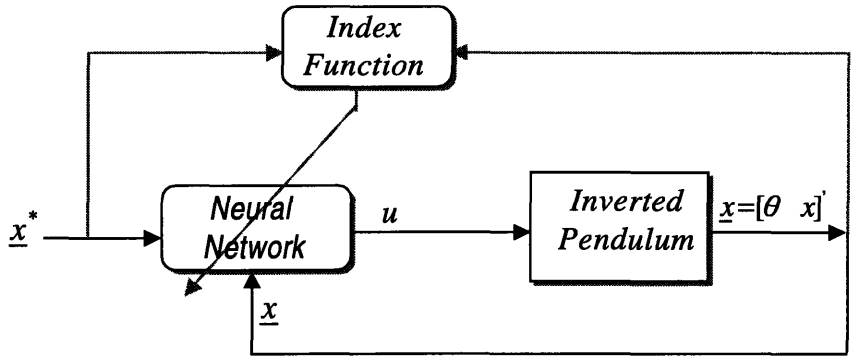
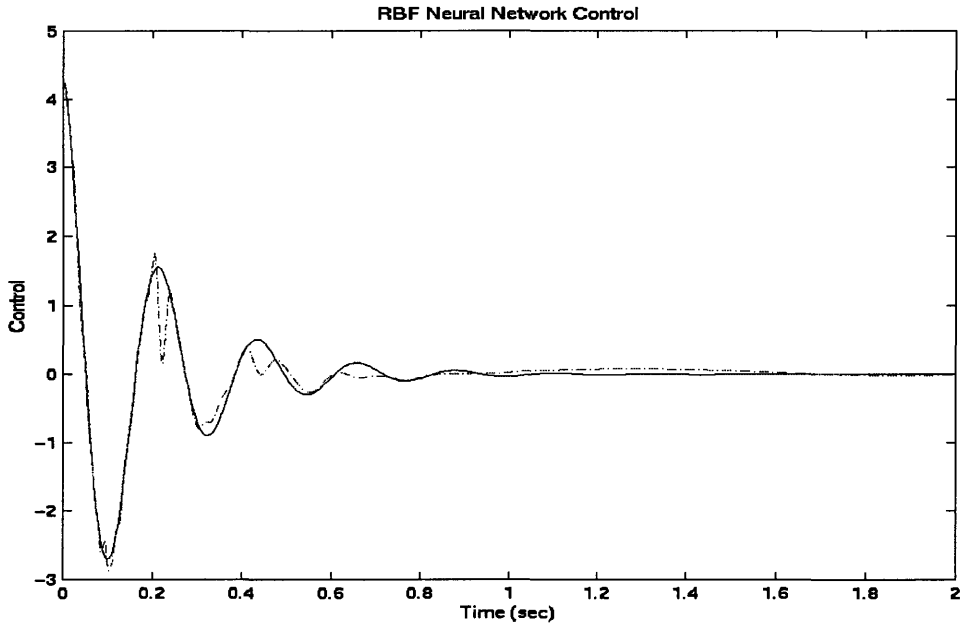


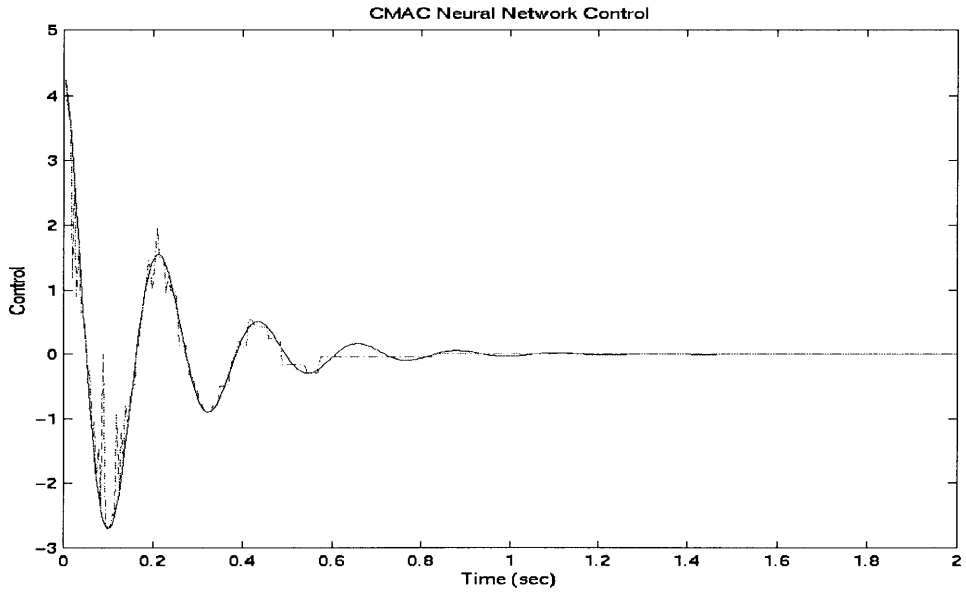
Figure 4-12: Neural network control paradigm with unsupervised learning.

4.6.4 Dynamic simulation of learning control and analysis

Using the model described in section 4.6.1, the simulation has been conducted successfully by means of leg lab creature library. The parameters of the inverted pendulum are *m=0.1 (kg), M=1.0 (kg),* and *l=0.5 (m).* Figure 4-13 shows responses of the RBF neural network controller and the CMAC neural network controller, which are trained with a linear controller (the supervisor). The RBF network has 100 hidden neurons and its training process takes about 1000 iterations. The CMAC network has 2 k hidden neuron cells and the training only takes about 200 iterations. Both neural networks worked well for the control of the given inverted pendulum model.



(a) RBF neural network control



(b) CMAC neural network control

Figure 4-13: Neural network control for an inverted pendulum model with supervised learning. (a) RBF neural network control. The solid line is linear control output and the dashed line is output from the RBF neural network. (b) CMAC neural network control. The solid line is output of a linear controller and the dashed line is output from CMAC neural network.

4.7 Comparison of BP, CMAC and RBF neural networks

Multilayer feedforward neural network with back propagation learning, CMAC neural network and RBF neural network are three neural network models that have been applied in control systems the most successfully so far. To investigate their capability in learning is very important for our potential applications of learning control in legged locomotion. There are several factors to be considered when one chooses a suitable neural network model for his applications, such as, training time, global optimization in learning, high dimension input field learning, error tolerance and adaptation capability. In the following, those characteristics of networks are evaluated in details.

4.7.1 Comparison based on XOR function learning

Table 4-1 shows the 2-D XOR function. The neural network learning setup is described as follows:

1). BP network: 3 layers, 2 hidden neurons, random initial values.
2). CMAC network: C=2, input variable dimension =3, 25 neuron cells used. zero initial values.
3). RBF network: 3 hidden neurons with centers (0,0), (1,1), (0.5,0.5), zero initial values.

Table 4-1: 2_D XOR function

Input x_1	Input x_2	Output y
0	0	0
0	1	1
1	0	1
1	1	0

Table 4-2 shows the training time comparison for three neural network models (BP network, RBF network and CMAC network). The cost function threshold used in training is 0.0002.

Table 4-2: Training time comparison

Type of neural network:	Number of iteration:	Network size:
BP neural network	25833	3 neurons
CMAC neural network	8	25 neurons
RBF neural network	2	3 neurons

4.7.2 Summary of comparison

Training time

Training time is a very important performance index for choosing neural networks. In some applications, on-line training is required. Thus the training speed becomes very crucial in those circumstances. In general, among three types of neural networks, BP, CMAC, RBF neural networks, back-propagation algorithm trained multi-layer neural networks require vast amount of training time (refer to Table 4-2). Comparably, the learning process of BP networks is the slowest. The learning process of RBF neural networks is in the same order of magnitude with CMAC neural networks. But in general, the CMAC neural network is slightly faster than the RBF neural networks in training with respect to the same application. The major reason is that the both CMAC and RBF neural networks have the spatial locality, but the BP networks don't. Therefore, the BP networks need to make adjustments to all the synaptic weights whenever a new data set is used in training.

Structure size

Time and space are the two related factors in the world, particularly in neural network design. People always try to make a good trade off between them. In neural network design, the network that has fast learning capability, usually has a big spatial structure. Comparing the three neural networks, BP, CMAC, RBF neural networks, the BP networks have the smallest structure size, RBF neural networks have a very big structure size, and CMAC neural networks have the largest spatial structure size (refer to the XOR function learning in 4.7.1).

Global optimization

Achieving the global minimum is a key in the learning process of a neural network. Failure in this implies that the network is not well trained and thus can not represent the plant that supplies the sample data sets. CMAC and RBF neural networks can guarantee the global minimum during the learning process. But the BP networks can reach local minimum in training. Therefore it is very difficult to apply the BP networks in a complex dynamic system.

Error tolerance

As long as the neural networks are well trained, they usually have certain capability of error tolerance, which can be proved theoretically by means of interpolating techniques. In general, CMAC networks have relatively worse error tolerance because of the square wave basis functions are used in the inherent network structure,

Adaptation capability

Since both RBF and CMAC neural networks have one hidden layer with nonlinear neurons and a linear output neuron layer, they have the potential to be added with certain adaptation capability. This will be described in details in chapter 6. However, for the BP networks, the task becomes extremely hard because of low converging speed and risk in local minimum trap.

The overall comparison is summarized in the table below:

Table 4-3: Property comparison of three networks.

Properties	*RBF*	*CMAC*	*BP*
Training	fast	fast	very slow
Computation	fast	fast	fastest
Network Size	big	biggest	small
Approximation	accurate	accurate	not always
Adaptation	yes	yes	not easy

Chapter 5

Neural Network Control of a Bipedal Walking Robot

In the previous chapter, three different types of neural networks have been investigated. As we discovered, RBF neural networks and CMAC neural networks are better than multi-layer neural networks trained by BP algorithm for a complex dynamic model. So in the application to a bipedal walking robot control, RBF neural networks and CMAC neural networks are two good candidates for us.

Space is one fundamental issue of the learning process, and time is the other. For real time application, better spatial representation obtained by neural networks (like RBF network, CMAC network) is needed. We can sacrifice the memory space in order to gain the advantages in learning speed and global optimization over the BP learning algorithm. In this chapter, dealing with the high dimension applications and dynamics learning with neural networks are the focus. Since the RBF neural networks and CMAC neural networks share a lot of characteristics, they are similar in the applications.

5.1 Clustering algorithm for RBF neural networks

As we know, both CMAC and RBF neural networks share many properties: fast training, global minimization and the locality. However, the complexity and difficulty has been shifted to the requirement of huge memory space in case of high dimension applications. The locations of the RBF (Gaussian) and CMAC (square shape) basis functions should be distributed in the entire input vector field according to the approximation theory. Suppose we have N inputs and we choose M even distributed Gaussian basis functions for every input variable, then the total number of required centers is N^M, which is an astronomical number. Obviously, selecting the centers of basis functions in this way is not practical at all. A good way to do it is to use data clustering algorithms or hashing functions (Miller 1989). The underlining assumption is that, in a real application, the system dynamics is usually constrained in certain subspace of the input variable space, we only need to choose a subset of points such that they cover the entire dynamic space of the system. Data clustering algorithm and hashing are two efficient ways to achieve the goal. The hashing function approach is an alternative to the clustering method and is not included in this study.

5.1.1 Advantage of clustering

As with the RBF neural networks, a very large training set may create an unnecessary heavy computational load on the system. But some form of clustering is desirable to replace all training vectors in a cluster with a single vector. Once a cluster of input training vectors is identified, its central vector is found by averaging all vectors in the cluster. This central vector then represents for the training vectors in the cluster.

In high dimensional input vector spaces, the volume of the space rises exponentially with the dimensionality. If a cluster center is defined where no input vectors are nearby, the output of that hidden layer neuron will be essentially zero for all input vectors. Equation (4-9) and (4-14) show

that no training will occur in this case. Thus, clustering is most useful where there is an excess of training data forming dense clusters.

5.1.2 Clustering algorithms

All of the basis function networks mentioned in this chapter may benefit by clustering the training vectors when there is a large amount of training data. In a dynamic control system or in signal processing applications, the amount of training data can be virtually unlimited. But it is believed that most data patterns are generally fixed. So if we can identify those data patterns by clusters and only use the central points of the clusters as the centers of basis functions, then the size (memory need) of the network and the computation load of the network are reduced.

Clustering algorithm I: Suppose there are N sample data sets and the maximum radius of clusters is R (or called threshold of clustering). The algorithm is summarized as follows:

a) Take a point as a center of first cluster.

b) For each unclassified point, compute the distances between the point and all the centers of clusters. If all of the distances are greater than the radius R, take the point as a center of new class. Otherwise, the point will be assigned to the cluster whose center is closest to the point.

c) The algorithm is terminated when all points are classified.

This algorithm has limitations, such as that the result is dependent on the chosen maximum radius and the order of pattern vectors to be processed. However, it constitutes a quick, simple way to establish some rough properties of a given set of data.

Clustering algorithm II: Maximum-distance algorithm (Wu 1994). This is a simple heuristic procedure based on Euclidean distance. Rather than forming new clusters during the classification process, this algorithm consists of two phases. The first phase determines all possible clusters and the second phase performs classification. The procedure of the algorithm is described as follows.

a) Take the first point as the first cluster center, and denote it by z_1.

b) Find the sample that is the farthest from z_1, and take it as the center of the second cluster z_2. Take the distance between z_1 and z_2 as the standard distance D_s.

c) Compute the distance from each remaining point to cluster centers z_1, z_2, ..., find pointswith the minimum distance to these cluster centers. Then find the largest of the minimum distances for all these points. If the ratio of this maxima over standard distance D_s is greater than a threshold T, the corresponding point is taken as another cluster center. This process terminates when no new cluster center can be found.

d) Assign each sample point to the nearest cluster as represented by their centers.

Clustering algorithm III: (K-means clustering algorithm) (Wu, 1994)

a) Choose K initial cluster centers $z_1, z_2, ..., z_k$.

b) Distribute the samples among the K clusters using minimum distance criterion.

c) Take the centroid of each cluster as a new center of that cluster.

d) Compare the new centers with the previous ones. If the change is below a threshold, terminate the algorithm, and go back to b) otherwise.

In the above two intuitive algorithms (I & II), cluster centers are determined randomly because the order of pattern processing and the threshold are all chosen randomly. Considering the fact that randomly chosen cluster centers are usually not exactly the right ones, then the clusters obtained usually cannot reflect the nature of the pattern data. K-mean algorithm dynamics updates cluster centers for each iteration as new samples have been assigned to clusters, until the cluster centers reach a stable state.

It is also worth to note that by the cluster center updating procedure, the center moves to the dense part of the cluster so that the intra-cluster distance is minimized. If the number of clusters is correct, the algorithm will eventually converge to a correct solution. But the behavior of the K-mean algorithm is influenced by the number of clusters specified, and the choice of initial cluster centers. An improved approach, adaptive K-means can be used in this regard (Moody and Darken, 1989).

5.2 Self-organizing RBF neural networks

With a proper clustering algorithm, one can design the RBF neural network structure and thus train the neural network efficiently in high dimension applications. The trained network will behave well with the inputs in the sample data sets. In fact, generalization and adaptation becomes very important in practical applications. Since the data sets were obtained from the given system within a period of time. The sample data can probably represent the major behavior of the system, but not the complete system behaviors. So, if a network trained with the collected sample data sets can do well in real applications in normal cases (similar situations as the sampling time), the trained network is considered having proper capability of generalization. If not, then on-line training is an option for developing the generalization capability.
There are three steps for training a network:

1) Supervised off-line training: set up the structure of network, train the network with the sample data sets available.
2) Supervised on-line training: develop network generalization capability.
3) Unsupervised learning for environment adaptation and robustness enhancement.

A self-organizing neural network is an ideal network that has been pre-trained with the sample data sets and behaves well in the normal situations, and has the potential in adaptation and structure modification. RBF neural network can be designed into such a format because of its locality property. In a real time application, if new data which does not belong to the training data sets shows up, it is easy to modify the network structure such that the global behavior being improved. Adaptation is used to deal with the disturbances and environment changes.
Figure 5-1 is a picture of a self-organizing RBF neural network paradigm.

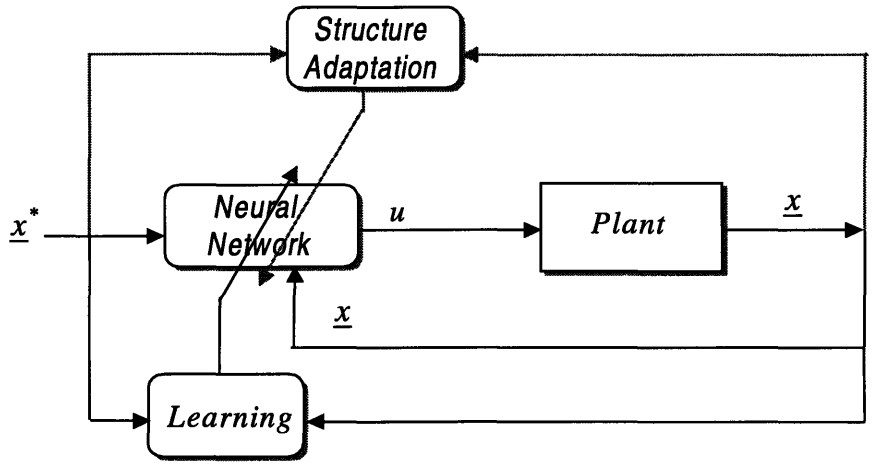


Figure 5-1: Self-organizing RBF neural network structure.

Structure modification: detect the irregular dynamics by observing the neuron outputs, $g_i(.)$, when $0 < g_i(.) \leq \beta$ (threshold, $0 < \beta \leq 1$), $\forall\, i \in [1,...,N]$, then a new center needs to be added.

Adaptation (unsupervised learning): take a performance measurement index to supervise the neural network and do the weights adjustments when necessary. A Lyapunov stability theory based adaptation scheme is developed in next chapter.

5.3 Application to dynamic bipedal walking control

For the bipedal walking control, the sample data sets are collected from a virtual model control module which can propel the biped walking successfully. There are two RBF neural networks used to control the leg locomotion. The dual neural network model training approach is actually based on the belief that the coupling between the left leg joints and the right leg joints is week. It is appropriate to use two separate neural networks for identifying the extrinsic dynamics with virtual model control. Figure 5-2 shows the general structure of RBF neural network locomotion control with supervised learning, which is used for both left leg joint control and right leg joint control. In the simulation of RBF neural network control of a bipedal walking robot, the input variables are modified slightly. Considering the decoupling requirement, the global state variable (of the state machine) is used as an input signal for the neural networks of left leg and right leg. Therefore each neural network control model has seven inputs and three outputs.

Inputs: State machine state, hip angular position, hip angular velocity, knee angular position, knee angular velocity, ankle angular position, ankle angular velocity.

Outputs: hip joint torque, knee joint torque and ankle joint torque.

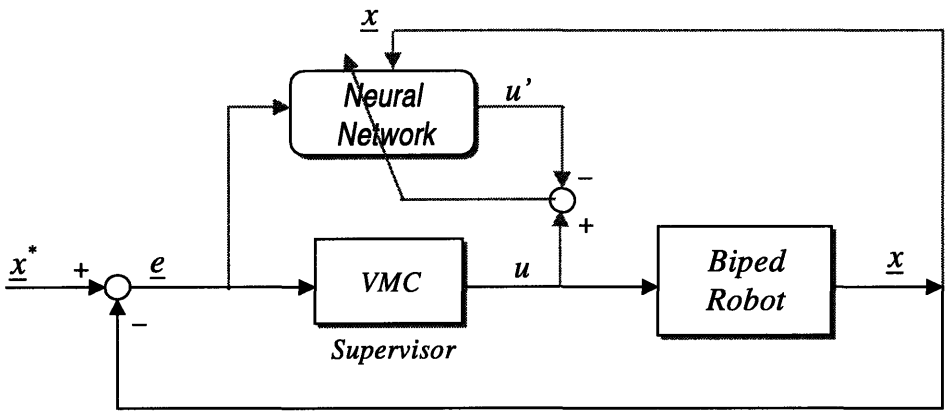
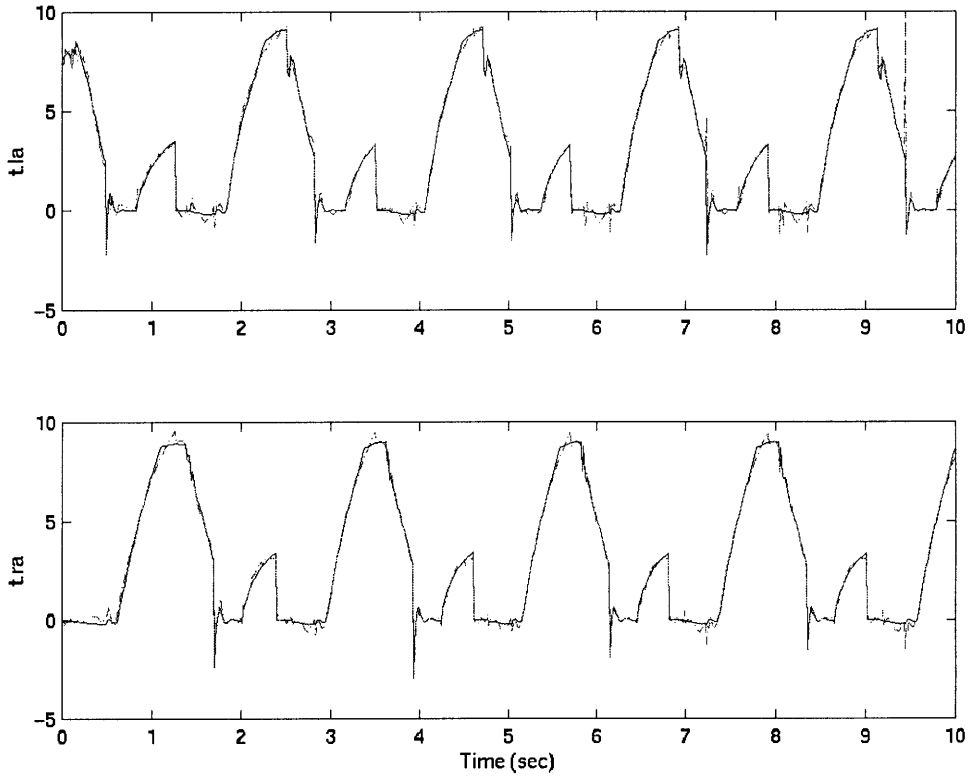


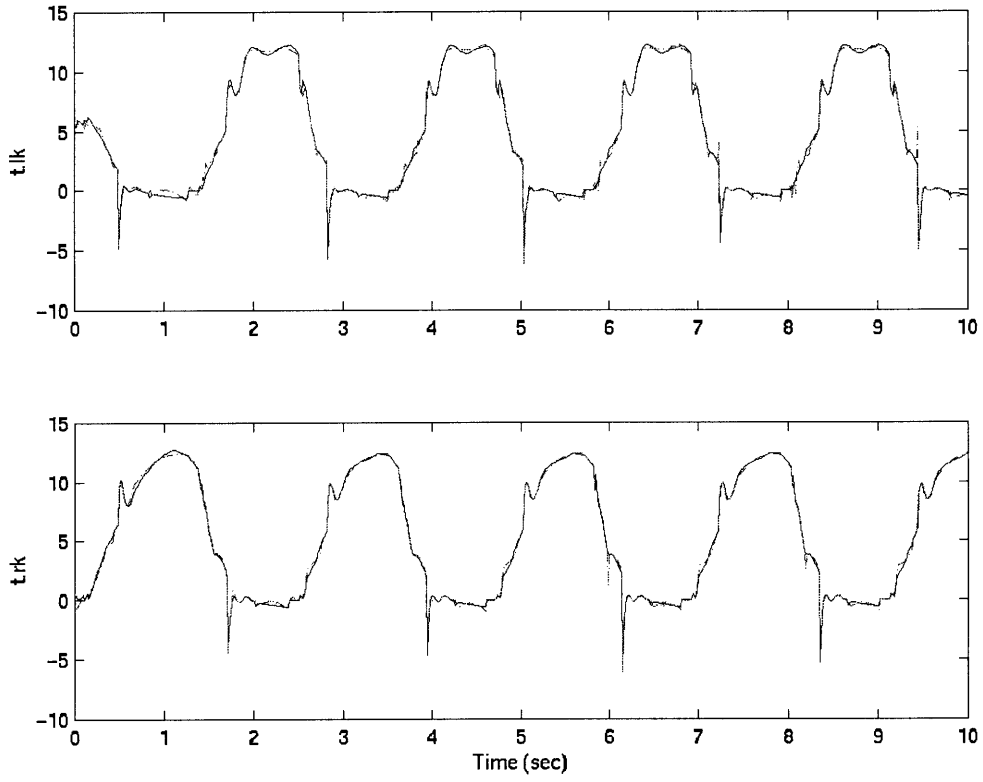
Figure 5-2: RBF neural network locomotion control with supervised learning.

5.4 Simulation and analysis

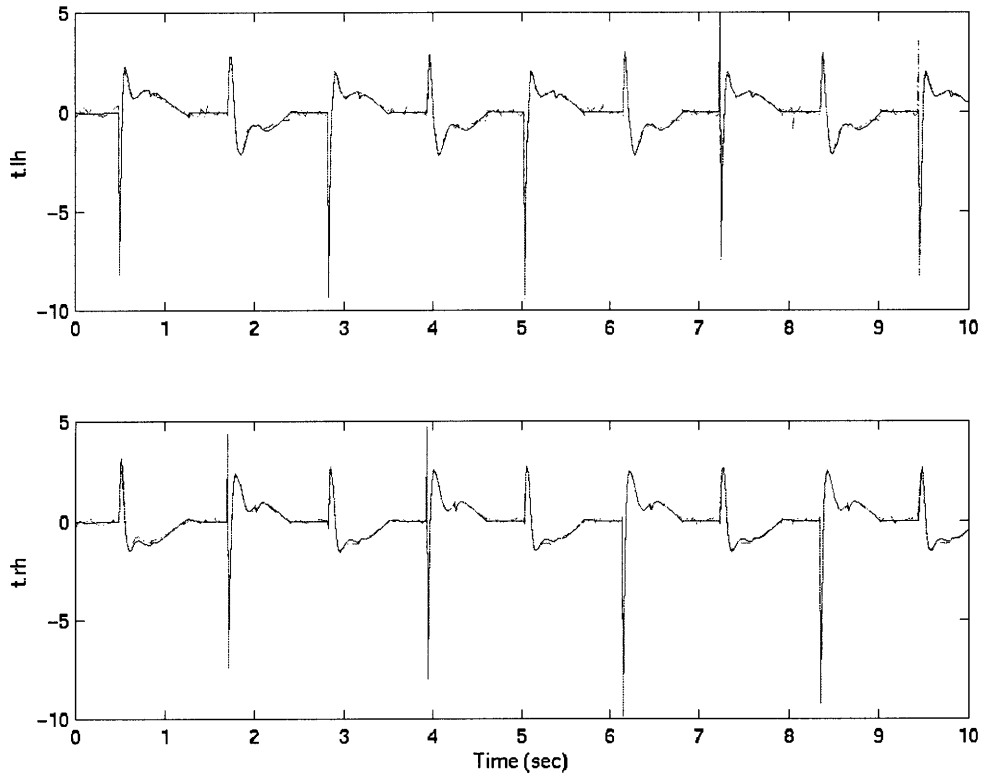
The following are the simulation results of RBF neural network control of a bipedal walking robot with supervised learning. The training data sets are shown in Appendix D. By using the training data sets, the two RBF neural networks were trained off-line. Then their capabilities of data prediction were tested by setting the networks as observers and using VMC (the supervisor) as an actual controller in the simulation. The comparison of VMC outputs and the network outputs are shown in Figure 5-3. One can clearly see that the prediction is accurate. However, because of the limited number of data sample points were used for training the networks, the neural network control could not do well. When untrained data points appeared in operation, the network could not make proper adjustments, then the overall control performance was not as good as the case with VMC. But the on-line training neural network control can control the walking robot as well as the VMC, which demonstrated that the on-line training algorithm and the network structure self-organizing algorithm are both correct. Figure 5-4 shows the data outputs from neural networks and the corresponding VMC. They are very close dynamically. Figure 5-5 is the stick diagram of the on-line simulation with neural network control.



(a) Prediction of ankle joint torque

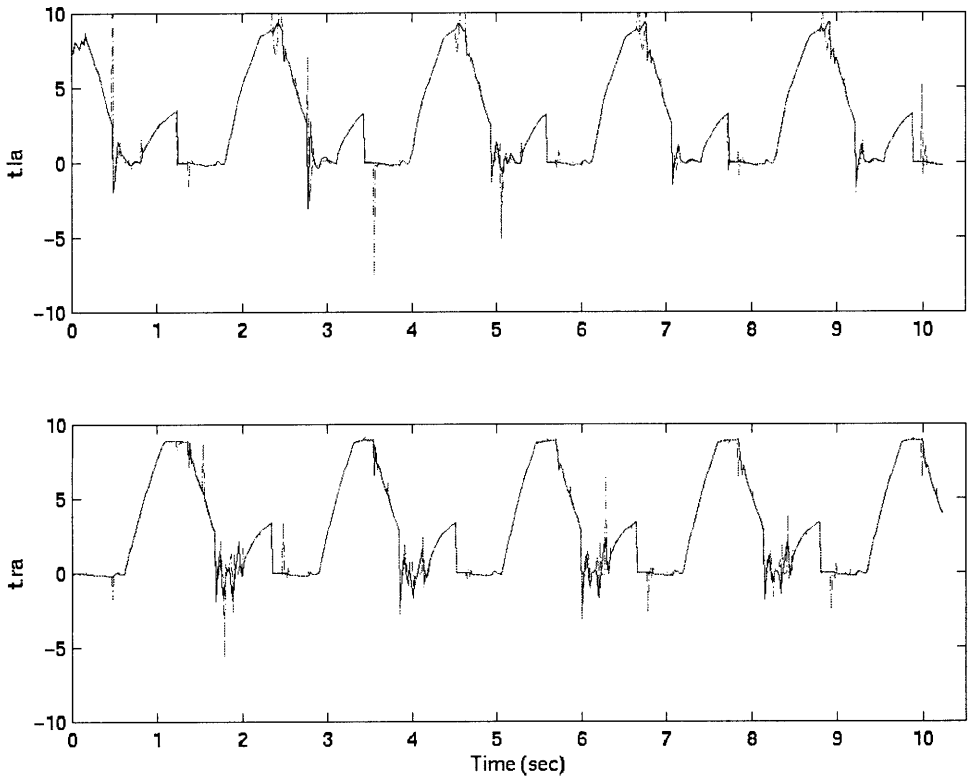


(b) Prediction of knee joint torque

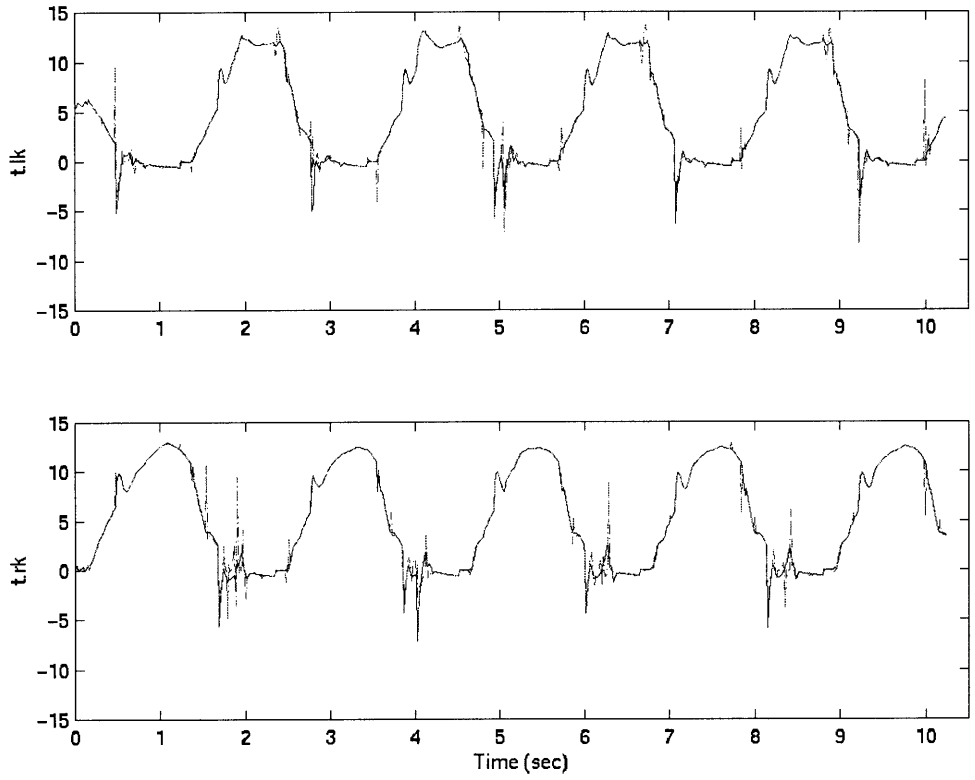


(c) Prediction of hip joint torque

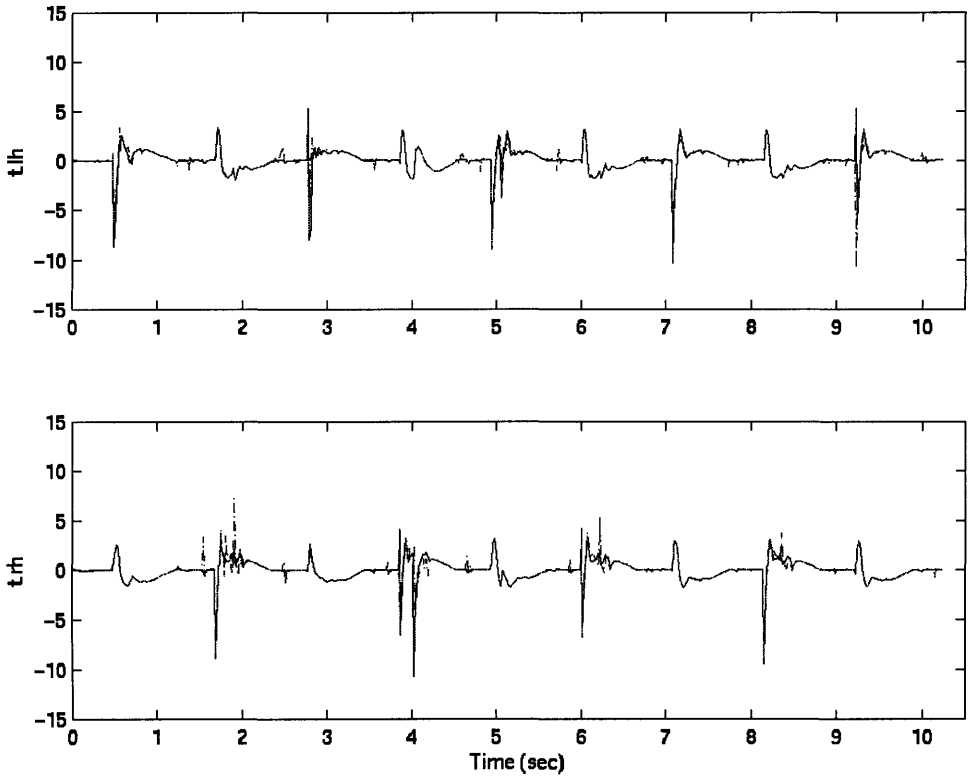
Figure 5-3: Dynamic prediction with neural network control trained off-line. Solid line figures are signals of VMC, and dashed line figures are signals from RBF neural networks.



(a) Ankle torque responses



(b) Knee torque responses



(c) Hip torque responses

Figure 5-4: Dynamic responses of neural network control with on-line learning. Solid line figures are signals of VMC, and dashed line figures are signals from RBF neural networks.

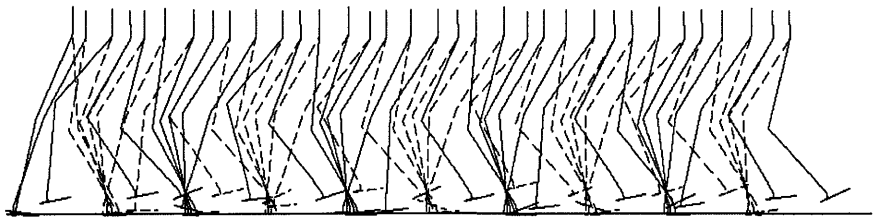


Figure 5-5: Stick diagram of a walking robot with neural network control (on-line learning)

Chapter 6

Adaptive Learning Control with RBF Neural Networks

In this chapter, the neural networks based adaptive learning mechanisms are studied and they are applied to enhance the robustness of virtual model control (VMC), which has been described in chapter 3. Using the framework of VMC (Type III), a neural network adaptive control (NNAC) approach is proposed, which combines and extends some recent insights in VMC, neural networks and adaptive system theory. The NNAC assumes a linear dynamics structure and uses a radial basis function neural network to identify the unmodelled dynamics. The RBF network can be parameterized in a Lyapunov stability based adaptation framework (Sanner & Slotine, 1991), and it is organized by means of the clustering method for the high dimension case. The residual unmodelled dynamics and external disturbances are accommodated by a dead zone robustness adjustment mechanism.

6.1 Dynamics representation in virtual space and neural network identification

Neural networks, such as RBF and CMAC neural networks, can be used to learn the deterministic nonlinear dynamics and the uncertain dynamics of a system and its environment variations. For the deterministic nonlinear dynamics identification, a supervised learning algorithm can provide a good solution. But for the uncertainty of dynamics in a system and the varying environments, adaptive learning is required. Frequently the traditional adaptation approaches can not do the job well, where the nonlinear and uncertain dynamics appears in the system operations. In such a case, neural network based adaptive learning is a good choice. In this study, only RBF neural network has been considered. The CMAC neural network can be applied similarly.

In order to have a learning system working efficiently, it is important to represent (or formulate) the system dynamics in a proper way and to assign suitable computation load to the adaptation learning mechanism. As described in chapter 3, virtual model control (VMC) is a feasible and convenient approach for biped locomotion control. So, in this study, the goal is to develop a neural network based adaptive learning control approach for the bipedal walking robot, such that the system performance (such as robustness and stability) can be further improved. In the following dynamics formulation, the virtual dynamics concept (described in section 3.2) is assumed.

To enhance the adaptation capability and robustness to the control system, we ought to include the interactive information between the robot and its environment. Two types of information can be available to our controller design, namely, the kinetic information of the center of robot mass and measured information from the physical joints. Any mal-functions of a biped robot is developed during a dynamic process of its physical joints. So, the information from the physical joints can be effectively used to aid the controller design.

The strategy described in Figure 3-6 can utilize the observed information and the measured information to achieve a robust controller design in virtual space, in which the real dynamics is imagined and decomposed into a Cartesian dynamic system.

In the virtual Cartesian space, the virtual dynamics can be considered as linear dynamics plus some nonlinear dynamics including physical joint dynamics coupling and the state of the finite state machine etc. The linear dynamics is based on the observed information in the virtual space, such as position, velocity of the center of mass. Generally, the linear dynamics (in one axis X, it is the same for Z, and θ axis) is expressed as

$$a_0\ddot{x} + a_1\dot{x} + a_2 x + a_3 = u \tag{6-1}$$

To capture the complex nonlinear dynamics of the real biped, a three-layer RBF neural network can be used.

$$f(\underline{X}) = f_0(\dot{x}, x, \dot{z}, z, \dot{\theta}, \theta) = f(\dot{q}_a, q_a, \dot{q}_k, q_k, \dot{q}_h, q_h, s) = \sum_{i=1}^{N} C_i g_i(\underline{X}, \xi_i) \tag{6-2}$$

$$\underline{X} = [\dot{q}_a, q_a, \dot{q}_k, q_k, \dot{q}_h, q_h, s] \tag{6-3}$$

$$g_i = e^{\frac{-\pi\|\underline{X}-\xi_i\|_2}{\sigma}} \tag{6-4}$$

where g_i is the nonlinear radial basis function of node i, ξ_i is the basis function center of node i, and $\underline{X}$ is the input vector, in which q_a, q_k, q_h indicate the angular position of the ankle joints, the knee joints and the hip joints respectively, and s means state variable of the state machine.

The desired dynamics can be specified based on the walking pattern and required performance. In a simple case, we can specify the desired dynamics in virtual space as,

$$\theta_d = 0,\ Z_d = c_1,\ \dot{X}_d = c_2 \tag{6-5}$$

where c_1 and c_2 are constants. In the above case, the desired trajectory was assumed as constant body height, zero pitch angle and constant forward velocity. In real world, animals don't use this type of desired trajectory, instead the desired trajectories of animals are determined basing on effort minimization and comfort maximization, which are hard to quantify and thus are very difficult to be formulated in a training process.

In summary, the virtual dynamics is represented as,

$$a_0\ddot{x} + a_1\dot{x} + a_2 x + a_3 + \sum_{i=1}^{N} C_i g_i(\underline{X}, \xi_i) + d + \varepsilon = u, \tag{6-6}$$

where d is the disturbance and ε is truncation error of the approximation with the finite neural network to the actual nonlinear dynamics. In equation (6-6), the unmodelled nonlinear dynamics is represented by means of a RBF neural network. When the coefficients of the network are determined by means of the adaptive learning algorithm, the dynamics will be identified consequently and will be used by the controller automatically. Thus the system performance is improved.

6.2 Adaptive learning control with RBF neural networks

Constructing the RBF neural network effectively so as to achieve good approximation of the target nonlinear dynamics is crucial for implementing this RBF neural network control. One strategy (Sanner & Slotine 1992) is to define a nominal sub-space D_n in which the RBF neural network is designed with even distributed basis functions. The adaptive control is a combination of linear control, sliding control and neural network control. The sliding control only works outside the sub-space D_n and it always drive the dynamics into D_n. Within D_n the neural network control takes over from the sliding control.

In our approach, we use a global dynamics formulation with RBF neural network (refer to equation (6-6)) and a self-organizing structure. In the control system, since we have seven input variables to the RBF neural network, high dimension becomes a serious barrier in implementing the RBF neural network. However from our observations, we found out that the robot dynamics is always continuous and confined in a sub-space of the dynamics, such as the working space. Therefore, we can use a clustering algorithm to determine the nominal centers for basis functions in the global space. To take into account the irregular cases, a self-organizing auxiliary network can be added based on the on-line detection of the dynamics states (refer to section 5.2). In this approach, the adaptive RBF neural network control is designed globally in the entire working space.

From (6-6) we can represent the dynamics as

$$a_0\ddot{x} + f_\Delta(\dot{x}, x, ...) + d + \varepsilon = u \tag{6-7}$$

$$f_\Delta(\dot{x}, x, ...) = a_1\dot{x} + a_2 x + a_3 + \sum_{i=1}^{N} C_i g_i(\underline{X}, \xi_i) \tag{6-8}$$

Choosing the switching (or sliding) surface as,

$$s = \dot{\tilde{x}} + \lambda\tilde{x}, \tag{6-9}$$

where $\tilde{x} = x - x_d$; $\lambda > 0$; x is the state variable in one virtual axis (X, Z, θ); x_d is the desired trajectory in one virtual axis.

According to the adaptive control developed in (Sanner & Slotine 1991, 1992), the control law can be designed in the following format,

$$u = u_{ad} + u_{pd} = Y\hat{a} - K_d s, \tag{6-10}$$

where $$Y = [\ddot{x}_r, \dot{x}, x, 1, g_1, g_2, ..., g_N], \tag{6-11}$$

$$a = [a_0, a_1, a_2, a_3, C_1, C_2, ..., C_N]^T, \tag{6-12}$$

$$\dot{x}_r = \dot{x}_d + \lambda(x - x_d). \tag{6-13}$$

Here $\hat{a}$ is the estimate of vector a. The control is composed of $u_{ad} = Y\hat{a}$, and $u_{pd} = -K_d \cdot s$, $K_d > 0$. Actually $u_{pd} = -K_d(\dot{\tilde{x}} + \lambda\tilde{x})$ is in the same form of a PD controller and an *Spring-damper* impedance controller like what was used in VMC system (Pratt 1996). Besides, u_{ad} is

the learning control part in this system since $\hat{a}$ will be updated on-line based on the system dynamics and interaction with the environments.

When the approximation error ε is small and the disturbance is ignorable, the adaptation (unsupervised learning) algorithm can be derived as,

Adaptation law: $$\dot{\hat{a}} = -\Gamma Y^T s, \tag{6-14}$$

which satisfy

$$\dot{V} = -K_d s^2 < 0. \tag{6-15}$$

The Lyapunov function candidate is

$$V = \tfrac{1}{2} a_0 s^2 + \tfrac{1}{2} \tilde{a} \Gamma^{-1} \tilde{a} \tag{6-16}$$

$$\text{where } \Gamma = diag\{\gamma_0 \quad \gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \delta_1 \quad \delta_2 \quad \ldots \quad \delta_N\}, \tag{6-17}$$

$$\gamma_i > 0,\ \delta_j > 0,\ (i = 0,1,2,3,\ j = 1,2,\ldots,N\ ),$$

$$\tilde{a} = \hat{a} - a. \tag{6-18}$$

Therefore the asymptotic and uniform global stability of the system can be achieved. (Slotine & Li, 1991)

6.3 Robustness with dead zone

Robustness is one of the system performance measurements. In reality, there are always some disturbances coupled into the system dynamics, which can affect the system behavior. Besides, for the dynamics identification with a finite size of neural network, there exists certain network approximation error with respect to the actual dynamics. Those effects should be considered for the system robustness enhancement.

Assuming the truncation error of the finite network approximation ε and the system disturbance d can be bounded as $|\varepsilon| + |d| \le E$. Then to tolerate the above bounded error term, a boundary layer with thickness Φ can be introduced. Consequently a new sliding variable with dead zone s_Δ is defined as follows,

$$s_\Delta = s - \Phi sat(\frac{s}{\Phi}) \tag{6-19}$$

$$sat(x) = \begin{cases} x, & |x| \le 1 \\ \text{sgn}(x), & else \end{cases} \tag{6-20}$$

By using the Lyapunov function candidate,

$$V = \tfrac{1}{2} a_0 s_\Delta^2 + \tfrac{1}{2} \tilde{a} \Gamma^{-1} \tilde{a} \tag{6-21}$$

where $\Gamma = diag\{\gamma_0 \quad \gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \delta_1 \quad \delta_2 \quad \ldots \quad \delta_N\}$, $\gamma_i > 0$, $\delta_j > 0$, $(i = 0,1,2,3,$ $j = 1,2,\ldots,N)$, $\tilde{a} = \hat{a} - a$.

Combining the equations (6-7)~(6-13), (6-19)~(6-21), and using the following adaptation law, we can derive,

$$\dot{V} = -K_d s_\Delta^2 < 0 \tag{6-22}$$

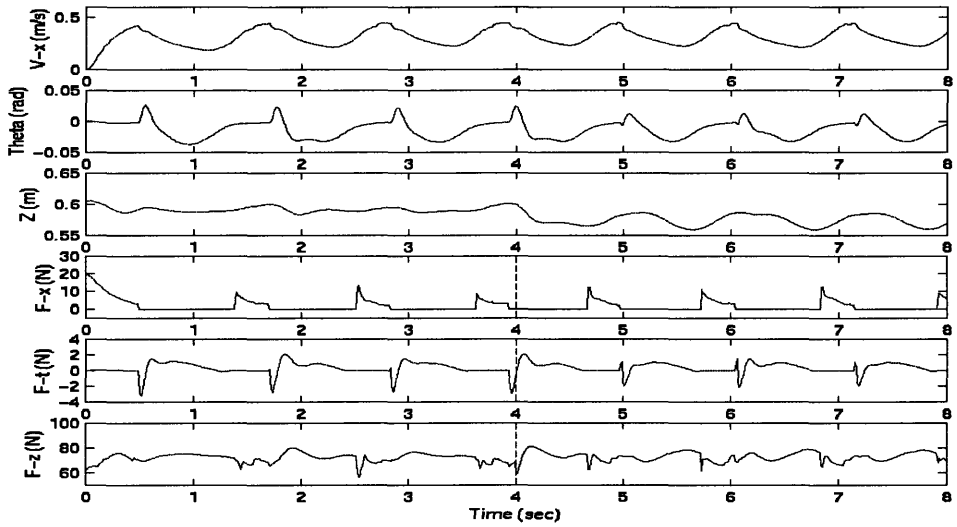
Adaptation law: $\dot{\hat{a}} = -\Gamma Y^T s_\Delta$. (6-23)

Therefore, the asymptotic and uniform global stability of this system can be achieved ($s_\Delta \to 0$, $t \to \infty$) by Barbalat's Lemma (Slotine & Li 1991). This also implies that the system trajectory error will be confined in the chosen boundary layer. Hence the robustness is enhanced.

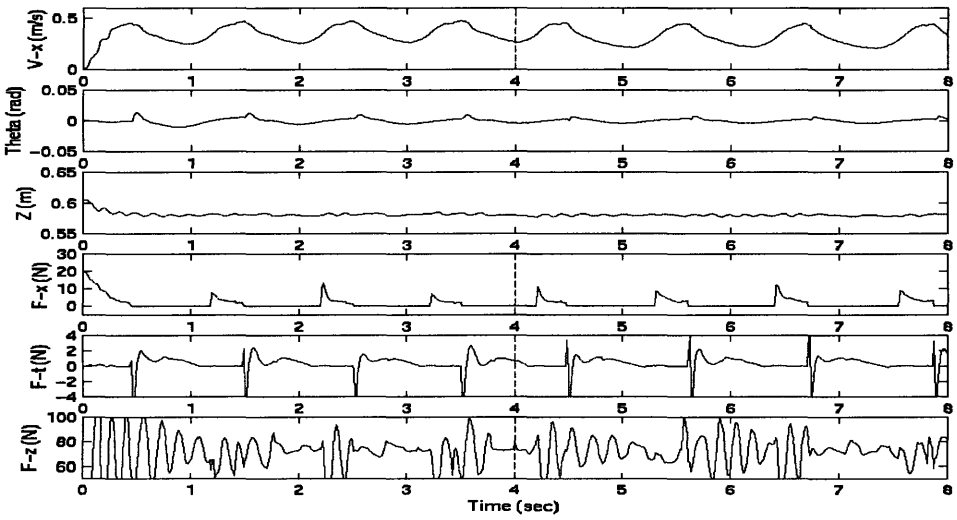
6.4 Simulations and analysis

A dynamic simulation was designed based on the "Creature Library" and implemented by Computer C. The motion of biped robot was confined in a sagittal plane and was controlled to walk on the flat ground with some mild rough terrain on it. It walked stably and robustly in MIT Leg Lab in the simulation. This robot model has height 0.8 meters, leg length 0.5 m, foot length 0.1 m, and body weight 8 kg. There are six joint actuators for hip, knee and ankle in each leg. There are two RBF neural networks, which were implemented in parallel and used to capture the nonlinear structure dynamics of the left leg and right leg during walking. The size of each network is N=200 in our simulation. The simulation has shown that the adaptive control algorithm with RBF neural network worked correctly and the system robustness was enhanced through the simulation tests.

Figure 6-1 shows that the dynamics of the robot when it encountered disturbances. Figure 6-2 shows the stick plot of stable walking.



(a)



(b)

Figure 6-1: Dynamic response of the bipedal walking robot simulation. (a) Virtual Model Control is used in the upper plots. (b) Adaptive Virtual Model Control is used in the lower plots. From top to bottom, graphs are forward velocity, pitch, height, virtual horizontal force, virtual torque about the hip, and virtual vertical force. At t=4 seconds, the stiffness of the ground was significantly reduced.

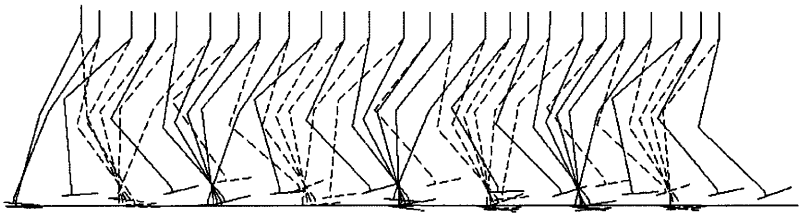


Figure 6-2: Stick diagram of the bipedal walking robot simulation data from Figure 6-1b when controlled by the Adaptive Virtual Model Controller with RBF Neural Networks.

Chapter 7

Conclusions and Future Work

7.1 Conclusions and discussions

7.1.1 Conclusions

1) Neural networks. Three neural network models have been studied and analyzed. It is suggested that RBF neural networks and CMAC neural networks are suitable for applications in bipedal locomotion control because of the fast learning and locality features. The on-line learning algorithms and adaptive learning algorithms have been developed successfully.

2) Generalized virtual model control. Based on Taga's bio-mechanical neural control model, several different types of virtual control schemes are proposed. Among them, a centralized global gait driven virtual model control has been chosen for further developing robust and adaptive controller because of the structure simplicity.

3) Self-organizing neural networks. A self-organizing scheme has been developed for adjusting the neural network structure when untrained data appears in the learning process.

4) Robust control of biped walking robots. A nonlinear sliding mode based adaptive virtual model control approach has been proposed to enhance the robustness of the control system for a bipedal walking robot. When adaptation is added to the virtual components, the controller responds to time varying parameters and external disturbances. It also adapts to unmodelled dynamics, resulting in more accurate height and pitch trajectory tracking.

5) Adaptive learning control of bipedal robots. The robustness of a bipedal walking algorithm can be enhanced by the use of a general virtual model based adaptive learning control approach with neural networks. The learning and adaptation was implemented with a Radial Basis Function Neural Network Adaptive Controller. With the adaptive learning controller, a simulated bipedal walking robot maintained pitch and height tracking after the ground stiffness decreased significantly.

7.1.2 Discussions

1) Although robustness was enhanced in terms of height and pitch tracking, the adaptive controller does not guarantee the long-term stability of the walking gait. However, by ensuring more robust, consistent behavior of the height and pitch during each of the walking phases and at each of the state machine transitions, the robustness of the algorithm as a whole can be increased.
2) As is understood in bio-mechanical models (in Chapter 3), bipedal locomotion is composed of skeletal (structure) dynamics and gait dynamics. This thesis only deals with the former part, i.e. the structure dynamics. This is why the forward speed control can not be further improved significantly where a fixed gait state machine is used. To enhance the overall performance, gait adaptation should be considered in the future.

3) Without vision feedback, blind walking can only success under simple uncertain environments. There are some limits on blind walking control and adaptation. Robustness has been tested only with disturbances and texture changing in this study.

7.2 Future work

1) Because of the limitations with a fixed gait state machine, CPG driven neuro-mechanical locomotion control approaches (VMC type I, II) have been proposed. The CPG neural model used in this study has a good adaptation and stability in neuro-dynamics and thus in walking gaits. So it is suggested to investigate CPG control approaches in the future and explore the intrinsic mechanism and causality in biped locomotion control.

2) Adaptation is an important issue, which has been never studied completely so far. Under the paradigm in Figure 2-5 and Figure 2-6, adaptation components can be added in three different levels, such as, dynamics control level, motor intervention level and motor planning level. In fact, animals (including human being) have different adaptation or learning skills in different levels. In our study of locomotion control, it is wise to start with the above three levels of adaptation in our further research.

3) 3-D robot control study is something with the more complexity and of course it is the ultimate goal in the biped locomotion control. From 2-D control to 3-D is not as simple as an easy extension. There is something unknown to be explored further. So the 3-D locomotion control should be treated as a serious challenging task in the research.

4) The success of simulation in biped walking can be extended into studying the malfunction of human being in leg locomotion (by simulation). A direct application is in orthotics or prosthetics study.

Appendix A

Least Mean Square Algorithm

Given a system as shown in Figure 4-2. Assume the system model,

$$\hat{y}[n] = \sum_{i=1}^{N} c_i x_i[n], \tag{A-1}$$

$$\underline{x}[n] = [x_1[n], x_2[n], \cdots, x_N[n]]^T \tag{A-2}$$

$$\begin{cases} y[1] = c_1 x_1[1] + c_2 x_2[1] + \cdots + c_N x_N[1] \\ y[2] = c_1 x_1[2] + c_2 x_2[2] + \cdots + c_N x_N[2] \\ \qquad\qquad \cdots\cdots \\ y[S] = c_1 x_1[S] + c_2 x_2[S] + \cdots + c_N x_N[S] \end{cases} \tag{A-3}$$

This becomes an over-determined equation when $S>N$.
Least Mean Square (LMS) solution for the above system:
Define the cost function as,

$$J = \tfrac{1}{2} \sum_{n=1}^{S} [y[n] - \hat{y}[n]]^2 \tag{A-4}$$

find $\{c_i\}$, such that

$$\hat{y}[n] = \sum_{i=1}^{N} c_i^{*} x_i[n] \tag{A-5}$$

minimizes the above cost function J.
Express equation (A-3) into a matrix form

$$\boldsymbol{Y} = \boldsymbol{H}^T \cdot \boldsymbol{W} \tag{A-6}$$

where $\boldsymbol{H} = [\boldsymbol{X}^1, \boldsymbol{X}^2, \cdots, \boldsymbol{X}^S]_{N \times S}$,

$$\boldsymbol{X}^i = [x_1[i], x_2[i], \cdots, x_N[i]]^T, \quad i = 1,2,\cdots,S$$

$$\boldsymbol{Y} = [y[1], y[2], \cdots, y[n]]^T,$$

$$\boldsymbol{W} = [c_1, c_2, \cdots, c_N]^T$$

The optimal solution to (A-4) and (A-5) is

$$\boldsymbol{W}^* = (\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}\boldsymbol{Y} = \boldsymbol{H}^{\#}\boldsymbol{Y} \tag{A-7}$$

where $\boldsymbol{H}^{\#} = (\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}$ is the pseudo-inverse of matrix $\boldsymbol{H}$.

Appendix B

Derivation of Back Propagation Algorithm for Multi-layer Feed forward Neural Networks

Figure B-1 is a general diagram for a multi-layer feed forward neural network. Suppose that the network has *N+1* layers (including the input layer). Figure B-1 shows notations and labels.

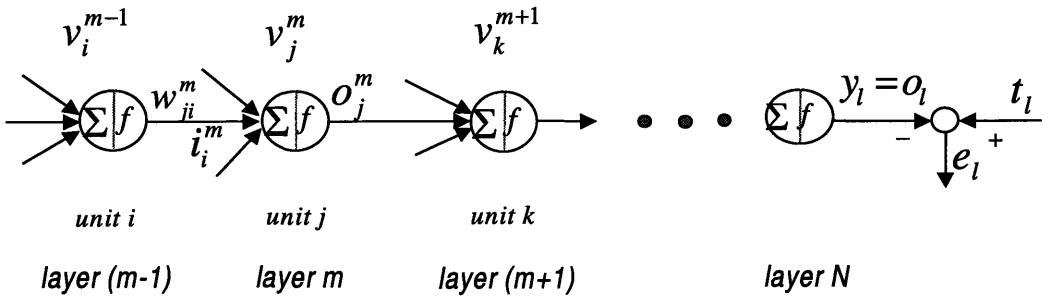


Figure B-1: General feedforward multi-layer neural network.

Notations:

$v_j^m \Rightarrow$ state of unit *j* in layer *m*;

$i_i^m \Rightarrow$ input to a unit in layer *m* from unit *i* of layer *m-1*;

$O_j^m \Rightarrow$ output from unit *j* in layer *m*;

$W_{ji}^m \Rightarrow$ weight of the connection from unit *i* (layer *m-1*) to unit *j* in layer *m*;

$$v_j^m = \sum_{i=1} W_{ji}^m \cdot i_i^m \tag{B-1}$$

$$O_j^m = f(v_j^m) = i_j^{m+1} \tag{B-2}$$

$$i_i^m = O_i^{m-1} \tag{B-3}$$

$$f(v) = \frac{1}{1+e^{-v}} \tag{B-4}$$

$$f'(v) = \frac{d}{dv} f(v) = (1 - f(v)) \cdot f(v) \tag{B-5}$$

Problem statement:
Given a set of training samples, obtain a learning rule for connecting weights in all the layers so that the total error E can be decreased in the steepest direction for such sample presentation.

$$E = \tfrac{1}{2}\sum_{l \in S_w} (t_l - O_l^N)^2 \tag{B-6}$$

where $S_w = \{1,2,\cdots,S\}$, S is the total number of samples.

Find ΔW_{ji}^m: (Assume $\exists f'(v)$)
Recall the steepest descent LMS rule,

$$\Delta W = -\rho \nabla_w E \tag{B-7}$$

$$\begin{aligned} \Delta W_{ji}^m &= -\eta \frac{\partial E}{\partial W_{ji}^m} = -\eta \frac{\partial E}{\partial v_j^m} \cdot \frac{\partial v_j^m}{\partial W_{ji}^m} \\ &= -\eta \frac{\partial E}{\partial v_j^m} \cdot i_i^m \end{aligned} \tag{B-8}$$

Define $-\delta_j^m = \dfrac{\partial E}{\partial v_j^m} = \dfrac{\partial E}{\partial O_j^m} \cdot \dfrac{\partial O_j^m}{\partial v_j^m}$

$$= \frac{\partial E}{\partial O_j^m} \cdot f'(v_j^m)$$

$$\Rightarrow \quad \Delta W_{ji}^m = \eta \delta_j^m i_i^m \tag{B-9}$$

Computing δ_j^m:
1) For the final layer (output layer, m=N):

$$\begin{aligned} -\delta_j^N &= \frac{\partial}{\partial O_j^N}[\tfrac{1}{2}\sum_l (t_l - O_l^N)^2] f'(v_j^N) \\ &= -(t_j - O_j^N) f'(v_j^N) \end{aligned} \tag{B-10}$$

$$\Rightarrow \Delta W_{ji}^N = \eta (t_j - O_j^N) f'(v_j^N) \cdot i_i^N = \eta \cdot \delta_j^N \cdot i_i^N$$

$$\delta_j^N = (t_j - O_j^N) f'(v_j^N) \qquad \text{(Widrow-Hopf Rule)} \tag{B-11}$$

2) For the hidden layers ($1 \le m \le N-1$):

$$v_k^{m+1} = \sum_j W_{kj}^{m+1} \cdot O_j^m \tag{B-12}$$

$$\begin{aligned} -\delta_j^m &= \frac{\partial E}{\partial v_j^m} = \frac{\partial E}{\partial O_j^m} \cdot \frac{\partial O_j^m}{\partial v_j^m} \\ &= \sum_k \frac{\partial E}{\partial v_k^{m+1}} \cdot \frac{\partial v_k^{m+1}}{\partial O_j^m} \cdot f'(v_j^m) \end{aligned}$$

$$\Rightarrow \delta_j^m = \sum_k \delta_k^{m+1} \cdot W_{kj}^{m+1} \cdot f'(v_j^m) \tag{B-13}$$

where $\frac{\partial E}{\partial v_k^{m+1}} = -\delta_k^{m+1}$ (B-14)

and $\frac{\partial v_k^{m+1}}{\partial O_j^m} = W_{kj}^{m+1}$. (B-15)

3) Summary:

Back propagation algorithm (steepest descent) is summarized as follows,

$$\Delta W_{ji}^m = \eta \delta_i^m i_i^m \tag{B-16}$$

When $m = N$ (in the output layer),

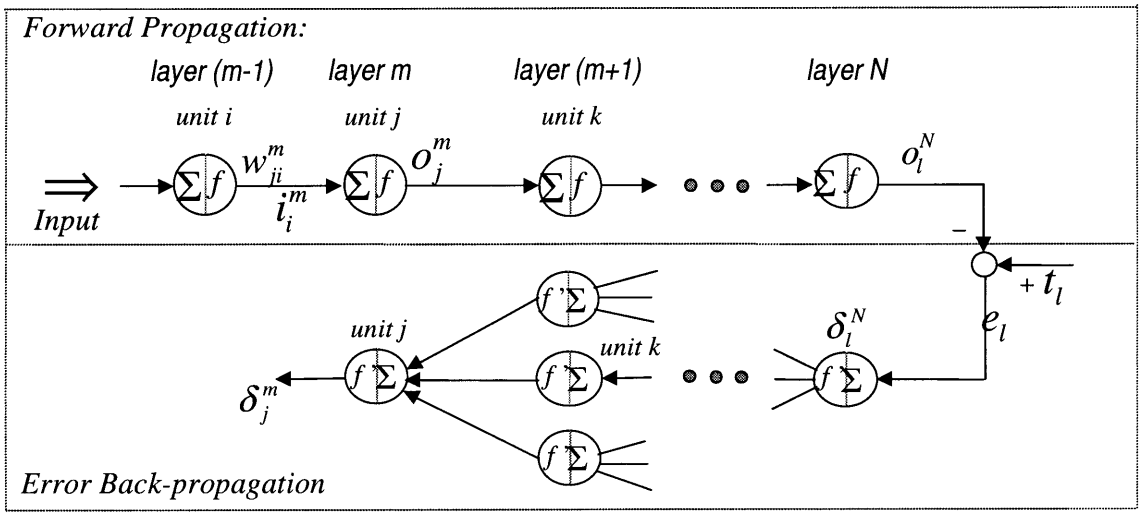
$$\delta_j^N = (t_j - O_j^N) f'(v_j^N) \tag{B-17}$$

When $1 \le m \le N-1$ (in the hidden layers),

$$\delta_j^m = \sum_k \delta_k^{m+1} W_{kj}^{m+1} f'(v_j^m) \tag{B-18}$$

where m is a layer number.

Figure B-2 shows the signal path of this back propagation algorithm.



Parallel Distributed Data Processing

FigureB-2: Signal flow in back-propagation learning algorithm.

The signal flow in this network is in a manner of parallel-distributed information processing. At first, the input signals go through the network forward, and then the network output is computed. Comparing the network output with the desired output from the sample data set, the training error of the network is obtained. Then in next step, the error message propagates from the output layer backward and go through the hidden layers down to the input layer. Then the errors at all the neurons can be calculated. Finally, using the weight updating formula (B-16) to compute the new weights in the network. In this learning algorithm, the error message has to propagate backward through the multi-layer network, thus the weights are updated consequently. Therefore, this learning algorithm was named as *Back Propagation Algorithm*, which is very powerful in small size of multi-layer feed forward neural networks.

Appendix C

Dynamic model of an inverted pendulum

For a model as shown in Figure 4-10, assume the pendulum rod is massless. Its dynamic model can be derived by means of Lagrangian approach.

Chosen the following generalized variables, $\xi_1 = x$, $\xi_2 = \theta$. Then the corresponding generalized force are $\Xi_1 = f$ and $\Xi_2 = \tau$.

Kinetic co-energy T^* is

$$\begin{aligned} T^* &= \tfrac{1}{2}M\dot{x}^2 + \tfrac{1}{2}mv_c^2 \\ &= \tfrac{1}{2}M\dot{x}^2 + \tfrac{1}{2}m[\dot{x}^2 + (l\dot{\theta})^2 + 2\dot{x}(l\dot{\theta})\cos\theta] \\ &= \tfrac{1}{2}(M+m)\dot{x} + \tfrac{1}{2}ml\dot{\theta}^2 + ml\cos\theta\cdot\dot{x}\dot{\theta} \end{aligned} \tag{C-1}$$

Potential energy V is

$$V = mgl\cos\theta \tag{C-2}$$

Therefore the Lagrangian L equals

$$L = T^* - V = \tfrac{1}{2}(M+m)\dot{x} + \tfrac{1}{2}ml\dot{\theta}^2 + ml\cos\theta\cdot\dot{x}\dot{\theta} - mgl\cos\theta \tag{C-3}$$

So the Lagrangian equations are

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = f \tag{C-4}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = \tau \tag{C-5}$$

$$\Rightarrow \frac{\partial L}{\partial \dot{x}} = (M+m)\dot{x} + ml\cos\theta\cdot\dot{\theta} \tag{C-6}$$

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta} + ml\cos\theta\cdot\dot{x} \tag{C-7}$$

Finally, from the above Lagrangian equations, we get the dynamics equations of an inverted pendulum,

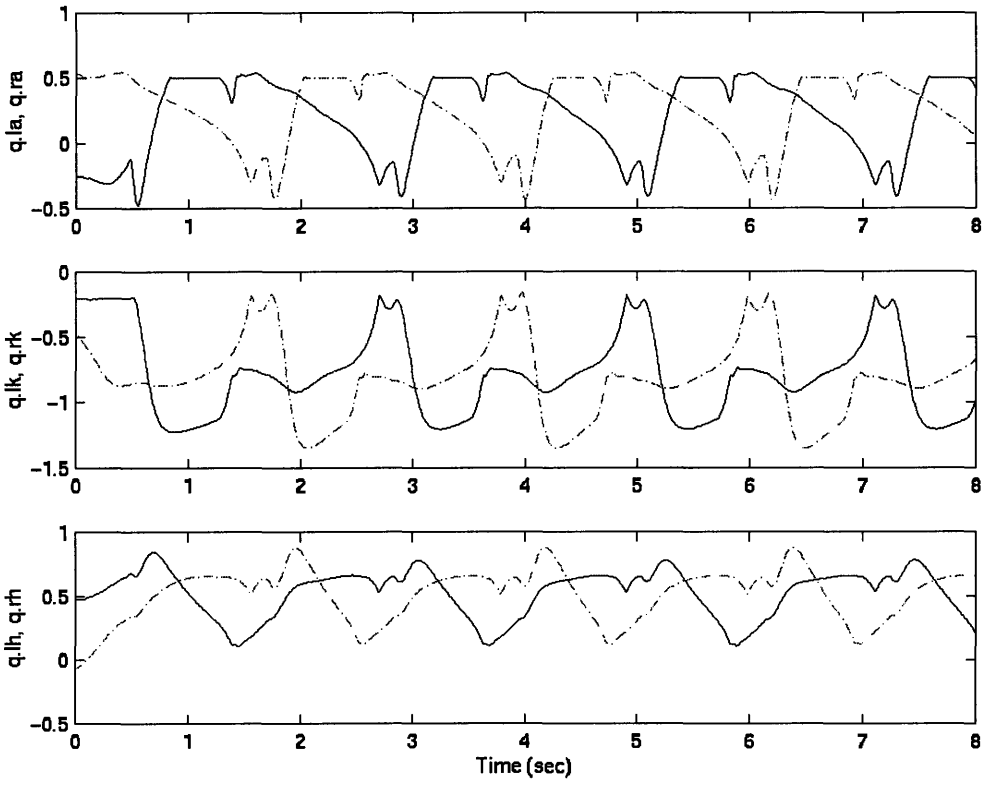
$$(M+m)\ddot{x} + ml\cos\theta\cdot\ddot{\theta} - ml\sin\theta\cdot\dot{\theta}^2 = f \tag{C-8}$$

$$ml^2\ddot{\theta} + ml\cos\theta\cdot\ddot{x} - mgl\sin\theta = \tau \tag{C-9}$$

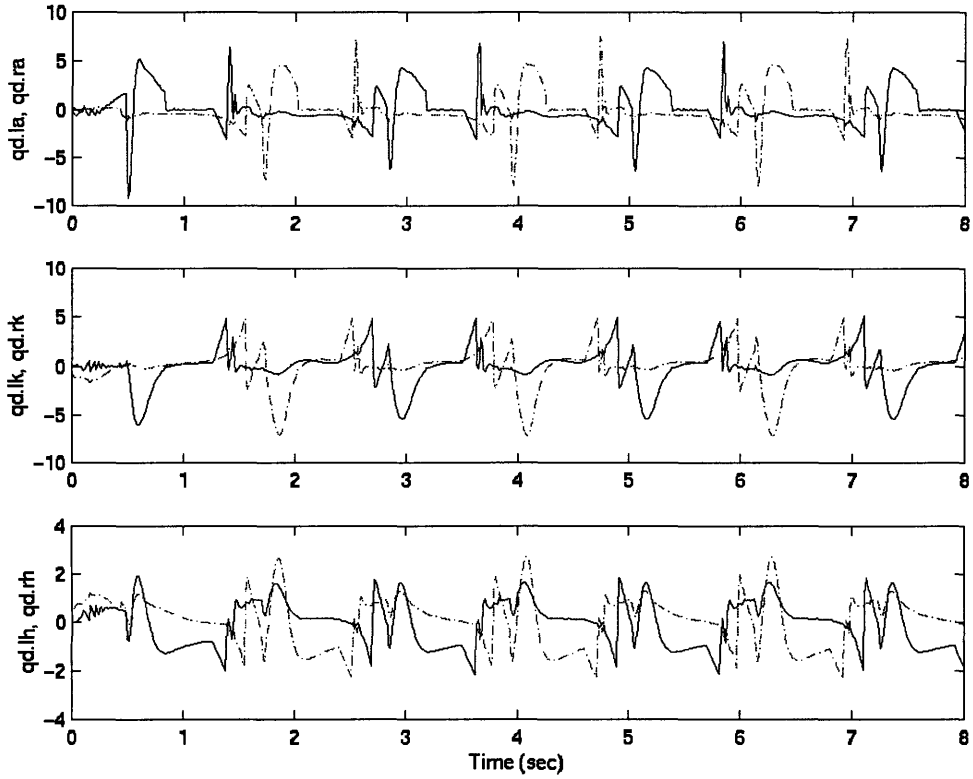
Appendix D

Input and output data for training the RBF neural network controller

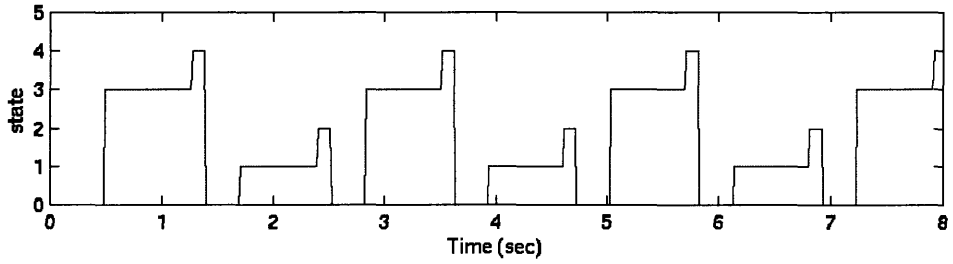
There are two RBF neural networks are used for the locomotion control of left leg joints and right leg joints. The inputs for each network are state machine variable, three joint positions and three corresponding joint velocities. The data is shown in the following figures (Figure D-1 and Figure D-2).



(a) Joint angular position signals

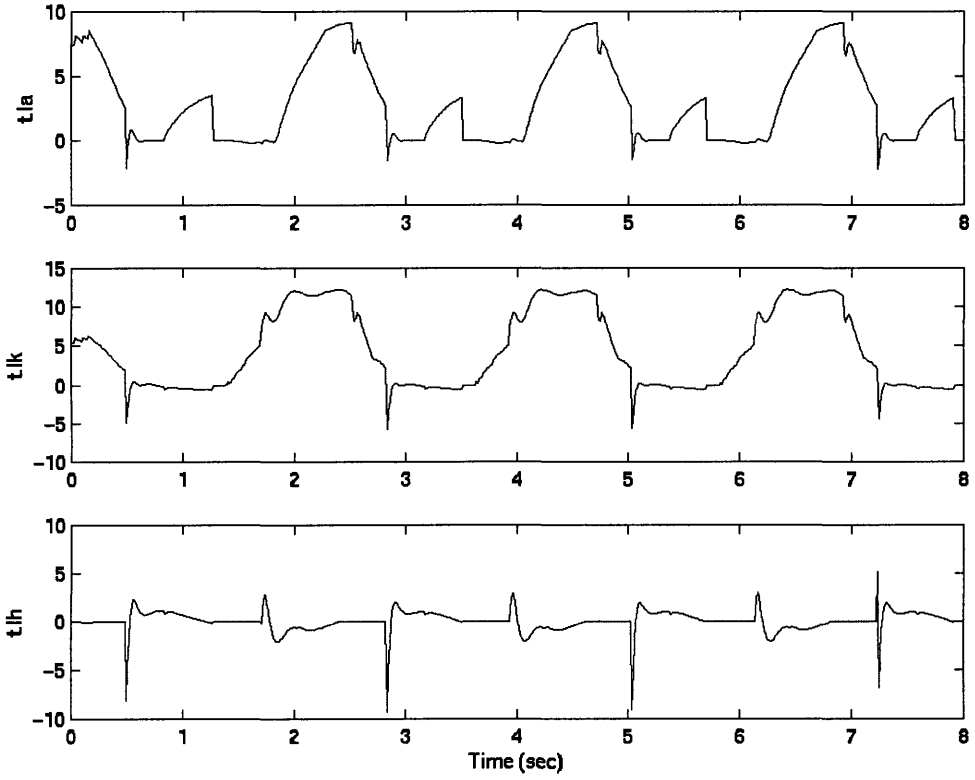


(b) Joint angular velocity signals

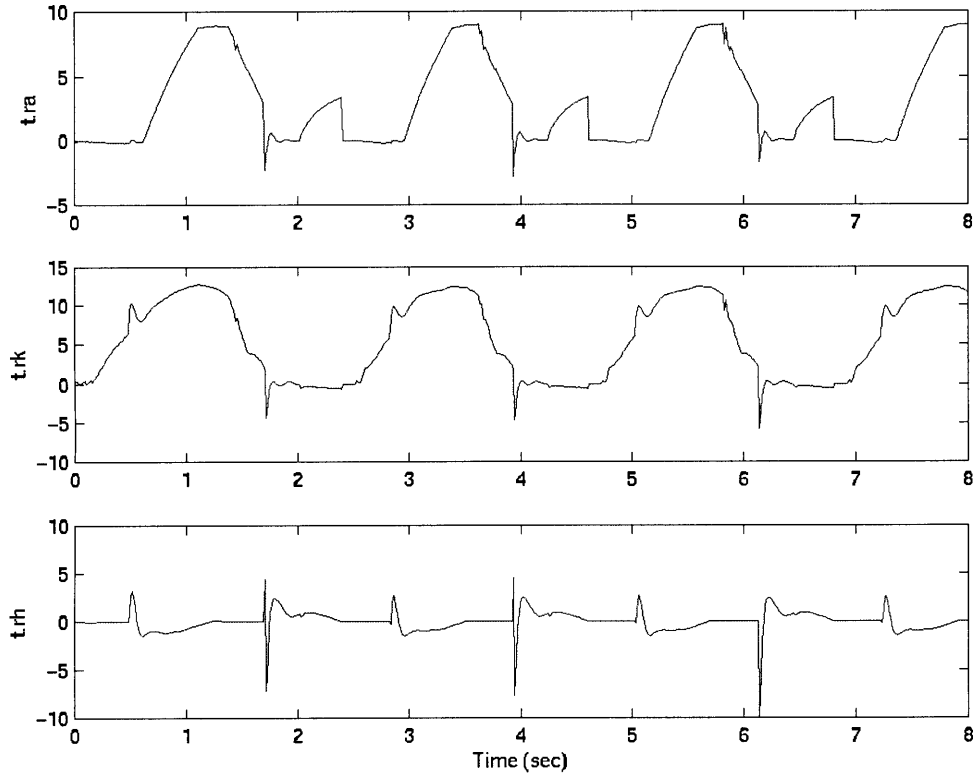


(c) Global state variable

Figure D-1: Input signals for off-line training of RBF neural networks. In (a) and (b), solid line figures are signals of the left leg, and dashed line figures are signals of the right leg.



(a) Output signals (torques of left leg joints) for off-line training.



(b) Output signals (torques of right leg joints) for off-line training.

Figure D-2: Output signals of the VMC module for off-line training of neural networks.

References:

[1] Albus, J. (1971), A theory of cerebellar function, *Math. Bioscience*, 10: 25-61.

[2] Albus, J.S., (1975a), A new approach to manipulator control: cerebellar model articulation control (CMAC), *Transaction on ASME, Journal of Dynamics Syst., Meas. and Contr.*, 97, 220-227.

[3] Albus, J.S., (1975b), Data storage in the cerebellar model articulation controller (CMAC), *Transaction on ASME, Journal of Dynamics Syst., Meas. and Contr.*, 97, 228-233.

[4] Albus, J.S., (1979), A model of the brain for robot control, *Byte*, 54-95.

[5] Alexander, R.M., (1990), Three uses for springs in legged locomotion, *International Journal of Robotics Research*, Vol.9, No.2.

[6] An, P.E., Brown, M., Harris, C.J., Lawrence, A.J., Moore, C.J., (1994), Associative memory neural networks: adaptive modelling theory, software implementations and graphical user, *Engineering Applied Artificial Intelligence*, 7(1), 1-21.

[7] Barto, A.G., Sutton, R.S. and Anderson, C.W. (1983), Neuron-like elements that can solve difficult learning control problems, IEEE Trans. on Systems, Man, and Cybernetics, 13(5): 835-846.

[8] Becker, S. (1991), Unsupervised learning procedures for neural networks, *International Journal of Neural Systems*, 2, 17-33.

[9] Carpenter, G.A. and Grossberg, S. (1988), The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, 77-88.

[10] Chew, C.M. (1998), Blind walking of a planar biped on sloped terrain, Master Thesis of Mechanical Engineering Department, MIT, Cambridge, MA.

[11] Chiel, H.J., Beer, R.D., Quinn, R.D. & Espenschied, K.S. (1992), Robustness of a Distributed Neural Network Controller for Locomotion in a Hexapod Robot, *IEEE Trans. on Robotics and Automation*, Vol.8, No.3.

[12] Dunn, E. and Howe, R. (1996), Foot placement and velocity control in smooth bipedal walking, *IEEE International Conference on Robotics and Automation*, Minneapolis.

[13] Fujita, M., (1982), Adaptive filter model of the cerebellum, *Biological Cybernetics*, 45: 195-206.

[14] Funahashi, K., (1989), On the approximate realization of continuous mappings by neural networks, *Neural Networks*, 2, 183-192.

[15] Furusho, J. and Masubuchi, M. (1987), A theoretically motivated reduced order model for the control of dynamic biped locomotion, *ASME Journal of Dynamic Systems, Measurement, and Control*, 109, 155-163.

[16] Golliday, C.L. and Hemami, H. (1977), An approach to analyzing biped locomotion dynamics and designing robot locomotion control, *IEEE Trans. on Automatic Control*, AC-22-6, 963-972.

[17] Grillner, S. (1976), Some Aspects on the Descending Control of the Spinal Circuits Generating Locomotor Movements, in Herman, R.N., Grillner, S., Stein, P.S. & Stuart, D.G. eds. 'Neural Control of Locomotion', Plenum Press, New York, NY.

[18] Grillner, S. (1985), Neurobiological bases of rhythmic motor acts in vertebrates, *Science* 228: 143-149.

[19] Handelman, D.A., Lane, S.H., and Gelfand, J.J. (1989), Integrating neural networks and knowledge based systems for robotic control, *IEEE International Conference on Robotics and Automation*, 3, 1454-1460.

[20] Hartman, E.J., Keeler, J.D. and Kowalski, J.M. (1990), Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation* 2(2): 210-215.

[21] Haykin, S. (1994), Neural Networks: A Comprehensive Foundation, Prentice-Hall, Inc.

[22] Hecht-Nielsen, R., (1989), Neurocomputing, Addison-Wesley, Reading, MA.

[23] Hu, J., Pratt, J., Chew, C.M., Herr, H. and Pratt, G., (1998), Adaptive Virtual Model Control of a Bipedal Walking Robot, *IEEE Symposium on Intelligence in Automation and Robotics*, Washington D.C.

[24] Hu, J., Pratt, J., and Pratt, G., (1998), Adaptive Dynamic Control of a Bipedal Walking Robot with Radial Basis Function Neural Networks, *IEEE International Conference on Intelligent Robots and Systems*, Victor, Canada.

[25] Ito, M. (1990), A new physiological concept on cerebellum, *Rev. Neurol* (Paris) 146: 564-569.

[26] Kajita, S. and Tani, K. (1995), Experimental study of biped dynamic walking in the linear inverted pendulum mode, *IEEE International Conference on Robotics and Automation.*

[27] Kato, T. et al, (1974), Information-power machine with senses and limbs, In *1st CIM-IFToMM Symp. on Theory & Practice of Robots & Manipulators*, Spring-Verlag.

[28] Kraft, L.K., and Campagna, D.P. (1989), A comparison of CMAC neural network and traditional adaptive control systems, American Control Conference, Pittsburgh, PA, 1, 884-889.

[29] Kohonen, T., (1997), Self-organizing Maps, Springer.

[30] Lee, S. and Kil, R.M. (1991), A Gaussian potential function network with hierarchically self-organizing learning, *Neural Networks*, 4(2): 207-224.

[31] Marr, D. (1969), A theory of cerebellar cortex, *Journal of Physiology* (London) 202: 437-470.

[32] Matsuoka, K., (1985), Sustained Oscillators Generated by Mutually Inhibiting Neurons with Adaptation, *Biological Cybernetics*, 52: 367-376.

[33] McGeer, T. (1990), Passive Dynamic Walking, *International Journal of Robotics Research*, Vol. 9, No.2.

[34] McMahon, T.A., (1984), Mechanics of Locomotion, *International Journal of Robotics Research*, Vol. 3, No. 2.

[35] Miller, S. and Scott, P.D. (1977), The spinal locomotor generator, *Exp. Brain Res.* 30:387-403.

[36] Miller III, W.T. (1989), Real time application of neural networks for sensor based control of robots with vision, *IEEE Trans. on System Man, and Cybernetics*, 19(4), 825-831.

[37] Miller III, W.T., Henes, R.P, Glanz, F.H. and Kraft III, L.G. (1990), Real time dynamic control of an industrial manipulator using a neural network based learning controller, *IEEE Trans. on Robotics and Automation*, 6(10, 1-9.

[38] Miller, W.T. III, (1994), Real-time neural network control of a biped walking robot, *IEEE Control System Magazine*, February.

[39] Hirai, K., Hirose, M. et al, (1998), The development of Honda Humanoid Robot, *IEEE International Conference on Robotics and Automation*, Belgium.

[40] Miura, H. and Shimoyama, I., (1984), Dynamic walk of a biped, *International Journal of Robotics Research*, Vol. 3, No. 2.

[41] Miyazaki, F., and Arimoto, S. (1980), A control theoretic study on dynamical biped locomotion, *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 102, 223-239.

[42] Miyazaki, F., and Arimoto, S. (1983), A hierarchical control for biped robots, *IEEE International Conference on Robotics and Automation*, 299-306.

[43] Moody, J. and Darken, C. (1988), Learning with localized receptive fields, *Proceeding of the 1988 Connectionist Models Summer School*, eds. Touretzky, D., Hinton, G. and Sejnowski, T., 133-143, San Mateo, CA: Morgan Kaufmann.

[44] Moody, J. and Darken, C.J. (1989), Fast learning networks of locally-tuned processing units, *Neural Computation*, 1(2): 281-294.

[45] Morecki, A. (1987), Biomechanics of Engineering: Modelling, Simulation, Control, Spring-Verlag, Wien-New York.

[46] Morrison, R.A. (1968), Iron mule train, In Cornell Aeronautical Lab./ISTVS Off-road Mobility Research Symposium, Washington, DC.

[47] Murakami, S., Ymammoto, E. and Fujimoto, K., (1995), Fuzzy control of dynamic biped walking robot, *IEEE international Conference on Fuzzy Systems*.

[48] Narendra, K.S. and Annaswamy, A.M. (1989), Stable Adaptive Systems, Prentice Hall.

[49] Narendra & Parthasarathy (1990), Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, No.1, 4-27.

[50] Ozawa, N., (1995), Locomotion Control System for Mobile Robot, United States Patent #5,402,050, Honda Motor Company.

[51] Parkins, E.J., (1997), Cerebellum and Cerebrum in Adaptive Control and Cognition, *Biological Cybernetics*, 77: 78-87.

[52] Poggio, T., and Girosi, F., (1989), A Theory of Networks for Approximation and Learning, Artificial Intelligence Lab. Memo No.1140, MIT, Cambridge, MA.

[53] Pratt, G., Matthew M.W., (1995), Series Elastic Actuators, *IEEE International Conference on Intelligent Robots and Systems*, Pittsburgh.

[54] Pratt, J., Torres, A., Dilworth, P., Pratt, G., (1996), Virtual Model Control, *IEEE International Conference on Intelligent Robots and Systems*, Osaka, Japan.

[55] Pratt, J., Dilworth, P., Pratt, G., (1997), Virtual Model Control of a Bipedal Walking Robot, *IEEE International Conference on Robotics and Automation*, Albuquerque, NM.

[56] Pratt, J and Pratt G., (1998), Intuitive control of a planar biped walking robot, *IEEE International Conference on Robotics and Automation*, Belgium.

[57] Raibert, M.H., (1986), Legged Robots That Balance, Cambridge, Mass: MIT Press.

[58] Rumellhart, D.E., Hinton, G.E. and Williams R.J. (1986), Learning internal representations by back-propagating errors, *Nature* (London), 323: 533-536.

[59] Sanner, R.M and Slotine, J.E. (1991), Stable Adaptive Control and Recursive Identification Using Radial Basis Gaussian Networks, *IEEE International Conference on Decision and Control*, Brighton, England.

[60] Sanner, R.M and Slotine, J.E. (1992), Gaussian Networks for Direct Adaptive Control, *IEEE Transaction on Neural Networks*, Vol. 3, No.6.

[61] Slotine, J.J. and Li, W.P., (1991), Applied Nonlinear Control, Prentice Hall.

[62] Sutton, R.S. (1994), Temporal credit assignment in reinforcement learning, Ph.D. Dissertation, Univ. of Massachusetts, Amherst, MA.

[63] Taga, G., Yamaguchi, Y. and Shimizu, H., (1991), Self-organized control of biped locomotion by neural oscillators in unpredictable environments, *Biological Cybernetics*, 65: 147-159.

[64] Taga, G., (1995), A model of the Neuro-musculo-skeletal System for Human Locomotion: I. Emergence of Basic Gait, *Biological Cybernetics*, 73: 97-111.

[65] Taga, G. (1998), A model of the neural musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance, *Biological Cybernetics*, 78: 9-17.

[66] Todd, D.J. (1985), Walking Machines: An Introduction to Legged Robots, Kogan Page Ltd., London.

[67] Vukobratovic, M. (1973), Legged locomotion robots: mathematical models, control algorithms, and realizations, In *5th IFAC Symposium on Automatic Control in Space*, Genoa.

[68] Vukobratovic, M., Borovac, B., Surla, D. and Stokic, D. (1990), Biped Locomotion: Dynamics, Stability, Control and Application, Berlin-Heidelberg, Germany: Springer-Verlag.

[69] Winter, D.A., (1990), Biomechanics and Motor Control of Human Movement, John Wiley & Sons, Inc.

[70] Warwick, K., Irwin, G.W., and Hunt, K.J. (1992), Neural networks for control and systems, Peter Peregrinus Ltd., London.

[71] Wu, J.K. (1994), Neural Networks and Simulation Methods, Marcel Dekker, Inc., New York.

[72] Xu, L., Jiang, J.P. and Zhu, J. (1994), Supervised learning control of a nonlinear polymerization reactor using the CAMC neural network for knowledge storage, *Proceeding of IEE, Part D*, 141(1), 33-38.

[73] Yuasa, H. Ito, M., (1990), Coordination of many oscillators and generation of locomotory patterns, *Biological Cybernetics*, 63: 177-184.