COMPUTATIONAL MODELING OF CEREBELLAR MODEL ARTICULATION CONTROLLER

ALIREZA JALALI

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

COMPUTATIONAL MODELING OF CEREBELLAR MODEL ARTICULATION CONTROLLER

ALIREZA JALALI

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Alireza Jalali**

Registration/Matric No: **WHA070021**

Name of Degree: **PhD**

Title of ~~Project Paper/Research Report/Dissertation~~/Thesis ("this Work"):

Computational Modeling of Cerebellar Model Articulation Controller

Field of Study: **Artificial Intelligence**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;

(2) This Work is original;

(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;

(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;

(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;

(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name :
Designation:

ABSTRACT

The cerebellum has major role in the human motor control to coordinate the motion. The cerebellar model articulation controller is a computational model of the human cerebellum. This research is towards the study of cerebellar model articulation controller (CMAC) and its application to non-linear systems. This model of the CMAC is developed to explore its potential for predictive control of movement.

The main limitation of Cerebellar Model Articulation Controller is memory size in application for non-linear systems. The size of memory which used by CMAC depends on input space dimension and input signal quantification step. Therefore, the efficient utilization of the CMAC memory is a crucial issue. Our main aim is to develop an optimal CMAC model which decrease memory size and increase the learning accuracy. To solve the memory size problem of CMAC a model namely Hierarchically Clustered Fuzzy Cerebellar Model Articulation Controller (HCFCMAC) is proposed. The performance of the proposed model is simulate and tested to control robotic arm. The presented simulation results show that proposed model is able to obtain a minimal modelling error and increase the learning accuracy.

This study is an examination of the HCFCMAC in biped robot control. It addresses simulations of the cerebellum to control robot swing leg. The proposed method includes a new concept of footstep planning strategy based on the Semi Online Fuzzy Q-learning concept for biped robot control in dynamic environments.

The main advantages of proposed approach are that, the computing time is very short and the footstep planning for both predictable and unpredictable obstacle in dynamic environment is operational. It will allow the controller to increase the strength. Another main contribution is on obstacle avoidance strategy for robot in dynamic environment. In this research the mathematical model of kinematics and dynamic of biped robot are described. Our approach is on gait pattern planning and control strategy for biped robot

stepping over dynamic obstacles. The high–level control used to predict the motion of the robot and the low-level control applied to compute the trajectory of swing leg with operation of HCFCMAC.

ABSTRAK (MALAY)

Otak kecil '*cerebellum'* mempunyai peranan utama dalam kawalan motor manusia untuk mengawal pergerakan. Pengawal Artikulasi Model Cerebellar adalah model komputer bagi otak kecil manusia. Dalam penyelidikan ini kita memberi tumpuan kepada kajian pengawal artikulasi model cerebellar (CMAC) dan aplikasinya pada sistem bukan linear. Model CMAC Ini dibangunkan untuk meneroka potensinya bagi kawalan ramalan dan membolehkan siasatan proses ramalan yang berkaitan dengan otak kecil dalam kawalan pergerakan.

Pengawal Artikulasi Model Cerebellar mempunyai had utama kepada saiz memori dalam aplikasi untuk sistem bukan linear. Memori yang digunakan oleh CMAC bergantung kepada dimensi ruang input dan langkah kuantifikasi isyarat input. Oleh itu, kecekapan penggunaan memori CMAC adalah isu penting. Matlamat kami adalah untuk mencari CMAC optimum yang membolehkan pengurangan dalam saiz memori dan masa pengkomputeran. Untuk menyelesaikan masalah saiz memori kami membentangkan model seni bina CMAC iaitu Pengawal Artikulasi Model Cerebellar Kabur Berkelompok Hierarki (HCFCMAC). Prestasi rangkaian yang dicadangkan diuji untuk mengawal lengan robot. Keputusan simulasi yang dibentangkan menunjukkan bahawa model kami boleh mendapatkan satu ralat model yang minimum.

Kajian ini merupakan pemeriksaan model HCFCMAC dalam kawalan robot. Ia menangani simulasi otak kecil untuk mengawal ayunan kaki robot dalam persekitaran dengan halangan, dalam sistem bukan linear. Kaedah ini termasuk strategi perancangan jejak langkah yang berdasarkan konsep Q-pembelajaran Kabur Semi Online untuk mengawal robot berkaki dua dalam persekitaran yang dinamik. Keberkesanan kaedah penyelesaian masalah utama dalam penyelidikan teknologi robot kawalan adalah juga

merupakan tumpuan utama. Halangan dinamik yang boleh dan tidak boleh diramalkan yang dihadapi dalam sistem dibincangkan.

Dalam kajian ini juga kami membentangkan satu konsep baru daripada strategi perancangan jejak langkah yang berdasarkan kepada konsep Q-pembelajaran untuk robot dalam persekitaran dinamik. Kelebihan utama pendekatan kami adalah tentang masa pengkomputeran yang sangat pendek dan perancangan jejak langkah itu beroperasi untuk kedua-dua halangan dinamik yang boleh dan tidak boleh diramalkan, membolehkan sistem kawalan dalam meningkatkan kekuatan. Satu lagi sumbangan utama ialah tentang strategi untuk mengelakkan halangan bagi robot dalam persekitaran yang dinamik. Strategi kawalan bagi pergerakan robot berkaki dua perlu dibahagikan terutamanya kepada dua kategori. Dalam kajian ini kita memberi tumpuan kepada model mekanikal kinematik dan dinamik bagi model berkaki dua. Di sini kita mengkaji model pergerakan bagi robot lima-link berkaki dua. Pendekatan kami adalah mengenai perancangan corak gaya berjalan dan strategi kawalan bagi robot berkaki dua melangkah halangan dinamik. Ia boleh dibuat untuk berfungsi secara berasingan untuk kawalan tahap tinggi dan kawalan tahap rendah. Kami memberi tumpuan kepada kawalan peringkat tinggi untuk meramalkan pergerakan robot. Untuk kawalan dalam talian masa pengkomputeran bagi proses pembelajaran adalah satu parameter kritikal. Untuk mengurangkan masa pengiraan, peringkat pembelajaran bagi perancangan langkah kaki digunakan untuk mereka bentuk strategi kawalan peringkat tinggi. Kita mengkaji kawalan peringkat rendah untuk mengira trajektori ayunan kaki dari output HCFCMAC.

Acknowledgements

I would like to begin with my sincere gratitude to the current Dean of the faculty, former Deans and the staff members for providing the support and assistance during this research. I would also like to express my gratitude to the former Deputy Dean Assoc. Prof. Dr. Diljit Singh for his invaluable advice and encouragement.

Most importantly and dearest to my heart, I am indebted to my supervisor Prof. Dr. Roziati Zainuddin and co-supervisor Dr. Woo Chaw Seng for their invaluable support, encouragement, guidance, corrections, reviews and motivations from initial stage to the final thesis compilation to ensure the success of this study.

I am also taking this opportunity to thank both the past and present members of the Artificial Intelligence Laboratory University of Malaya for their generous support, and the individuals who have contributed their voice for this research.

Finally I like to thank my beloved wife Marzieh Yaeghoobi, She was always there cheering me up and stood by me through the good times and bad as well all of my family members, who have made enumerable sacrifices to allow me to pursue my research goals.

May the Almighty God richly bless all of you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AAAI	- Association for the Advancement of Artificial Intelligence
AB	- Ascending Branch
AI	- Artificial Intelligence
BCI	- Brain Computer Interface
CF	- Climbing Fibers
CMAC	- Cerebellar Model Articulation Controller
CNS	- Computational Neuro Science
DHP	- Denavit-Hartenberg Parameters
DOF	- Degree Of Freedom
EBL	- Explanation-Based Learning
EDA	- Estimating Distributions Algorithm
EDAQ	- Estimating Distributions Algorithm Q-learning
FCMAC	- Fuzzy Cerebellar Model Articulation Controller
FQL	- Fuzzy Q-Learning
FRL	- Fuzzy Reinforcement Learning
GC	- Golgi Cells
HCFCMAC	- Hierarchically Clustered Fuzzy Cerebellar Model Articulation Controller
LTD	- Long-Term Depression
LTP	- Long-Term Potentiation
MDP	- Markovian Decision making Processes
MIMO CMAC	- Multi Input Multi Output (MIMO) CMAC state space
MRI	- Magnetic Resonance Image
NN	- Neural Network
PC	- Purkinje Cells
PF	- Parallel Fibers
RL	- Reinforcement Learning
SIMO-CMAC	- Single Output Single Input CMAC
SOFQL	- Semi On-line Fuzzy Q-learning
SOQL	- Semi On-line Q-learning
TD	- Temporal-Difference

CHAPTER 1

GENERAL INTRODUCTION

1.0 INTRODUCTION

1.1 Research Inspiration and Background

The motivation behind this research and research background are narrated in the following sections on Intelligence control, the cerebellum and problems in its simulation.

1.1.1 Intelligent Control

Now a day in modern society, the devices around us are independent on human to give them instruction on how they function. This is especially true of control of complex dynamic systems. However, the design and implementation of these control systems is much difficult, as complex input-output relationships resulting from the interaction between a process and its environment are often not readily solvable by traditional control methods. Classical control gives appreciable control on linear, non-time-varying, single-input and single-output systems, but is not desirable for the control of non-linear and time-varying systems (Nitin Mathur, 2004).

In recent times, Artificial Intelligence is tending to be profitable in making simpler the actual effects of these complex systems through permitting the gadgets to learn their own control features. By using intelligent systems, the machines are qualified of making human-like decision on their function without having human designer to provide solution for every problem. Intelligent control is that control algorithm is developed by certain characters of intelligent biological system like human being. The intelligent control uses several Artificial Intelligence computing approaches, such as neural network, machine learning, evolutionary computation, Bayesian probability, fuzzy

control and genetic algorithm. The algorithm and structure of a typical fuzzy control has been completely discused on many book and papers such as (Pawel, 2007). Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection and crossover (Liguo Huo, 2009). The resulting offspring populations are successively selected, recombined and altered to form iteratively better solution to the problem (Goldberg, 1989; Mitchell, 1996). Some examples of using GA in robotic control are, Davidor (1991) proposed a technique to apply GAs to the problem of robot trajectory generation in environment free of obstacles. Nearchou (1996, 1998) used GAs to solve the inverse kinematics problem in environments with obstacles. Simulation of human capabilities is one of the most challenging topics of intelligent control.

Simulations of human capabilities such as to the ability to walk happen as a result of connections between the nervous system and the musculoskeletal. In order to simulate these movements with the biped robot several aspects of human neurology must be integrated with biomechanics. The CMAC model is widely used because it features impressive speeds and learning abilities to accomplish this integration. The CMAC is a computational algorithm based upon activity through the cerebellum of the brain. The application was first used to research robotic arm movement. The CMAC algorithm began as the Cerebellar Model Arithmetic Computer, as a method of providing local generalization of the state space based upon how the human brain responds to stimulus (Lin & Wright, 2010). Robots and other automated devices are now capable of performing high level humanistic actions such as making decisions without a solution for each problem fact which could be encountered. The CMAC model has been revised to include enhancements such as a neuro fuzzy system (FCMAC). The fuzzy CMAC is more precise and highly capable to control the robot (Mohajeri et al., 2009).

1.1.2 The Cerebellum

Human motion occurs through a process of communication between the brain, the nervous system and the muscles. Simple movements such as playing a piano require a relay of messages from the cerebellar portion of brain to the fingers. The cerebellum plays a major role in human motor control, muscle control movements, and the senses. The most intricate functions perform associative mappings between the input sensory information and the cerebellar output required for the production of temporal-dependent precise behaviors (Kandll et al., 2000). To accomplish movement, over half of the neurons in the brain are located in the cerebellum. The cerebellum functions as a comparator with respect to its role in muscle control. A sample of the motor command from the cerebral cortex to the skeletal muscles is relayed to the cerebellar cortex for evaluation. Once the motor act begins, the cerebellar cortex begins to receive input through spinocerebellar tracts from the proprioceptors in muscles, tendons, and joints involved in the movement. In this way, the cerebellum compares the actual performance of a given movement with the original 'intent' of the brain (Computational NeuroScience Meeting, 2007). In addition to motor and sensory functions, the cerebellum also participates in cognitive processes such as learning a language.

The cerebellum is positioned in the lower back of the brain beneath the cortex's temporal and occipital lobes. It is comprised of two lateral hemispheres, which are responsible for neurotransmittal signals associated with movement, balance, posture, and sensorial perception (Kinser, 2013). Fissures divide the cortex into three lobes:

1) Flocculonodular,

2) Anterior

3) Posterior

Sensory information is transmitted in and out of the cerebellum through neuron cells at extremely high speeds. The cerebellar fiber peduncles transport signals into and out of

the brain. Three peduncles, which are the superior, middle and inferior consist of afferent and efferent fibers from different parts of the brain.

1.1.2.1 Inputs

Millions of neurons in the cerebellum process data and relay it to areas of the brain that manage motor control (Bailey, 2013). Mossy fibers carry input motor control information. The granule cell layer is a point of intake for mossy fiber inputs which come from the cerebral cortex, see figure 1.1.

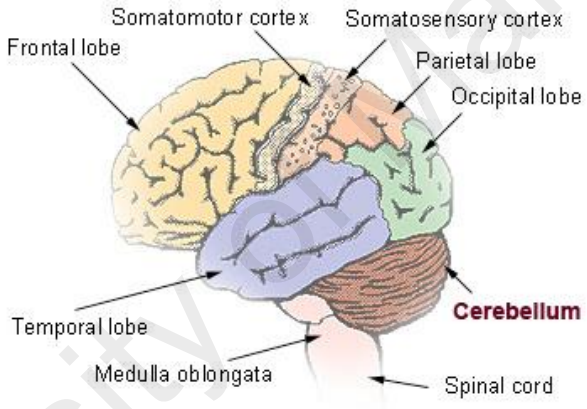


Figure 1.1: Cerebellum and Brain Facts (Adopted from Fine EJ, 2002)

Inputs are also received by the fastigial and interposed nuclei, and the dentate nucleus, which is the most massive of the cerebellar nuclei. All of the cerebellar nuclei receive input from the medulla's inferior olive. Afferent pathways include the anterior spinocerebellar and tectocerebellar tracts. Efferent pathways include cerebellorubral, dentatothalamic, and fastigioreticular fibers which emerge from cerebellar nuclei; the cerebellorubral fibers from the globose and emboliform nuclei, the dentatothalamic fibers from the dentate nucleus, and the fastigioreticular fibers from the fastigial nucleus (CNS, 2007).

1.1.2.2 Outputs

All outputs from the cerebellum depart from the cerebellar deep nuclei. The cerebellum is comprised of three divisions to produce the vermis and two hemispheres. Outputs to the red nucleus manage motor movement on the ipsilateral side of the body. At the point of motion, the cerebellum compares the inputs of the movement received by the cerebellar cortex. The lower surface of the cerebellum manages signals that travel to the nervous system, motor, and muscle tissue. The Purkinje cells integrate input connections. The interrelation of the cells within the cerebellum work integrally to process input and output signals (Marr, 1969). We can see the schematic of cerebellum in figure 1.2.

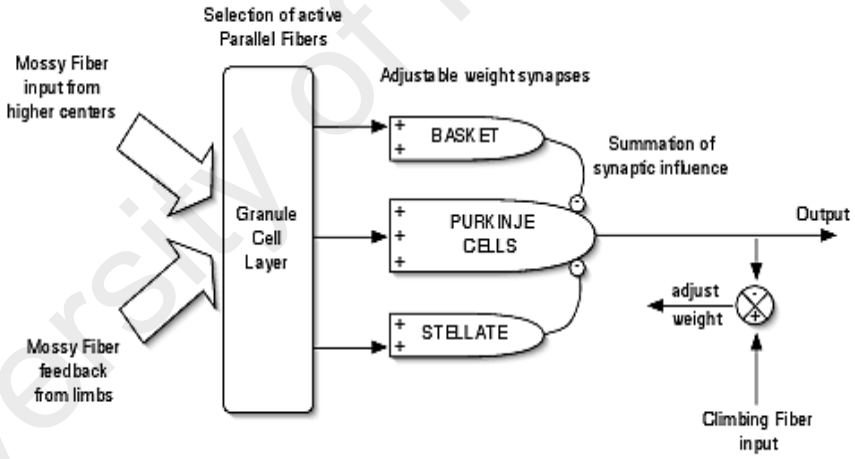


Figure 1.2: Schematic Diagram of the Cerebellum (Adopted from J. Albus, 1989)

As the dysfunction of the human cerebellum disrupts motor processes in human motion, defects in the Cerebellar Model Articulation Controller (CMAC) model for simulated robotic motion disturb the motor processes.

1.1.2.3 Mossy Fibers and Purkinje Cells

Human motion originates by excitable signal inputs and outputs from the cerebellum. Climbing, Mossy, and Parallel fibers integrate as a network to process signals in the cerebral portion of the brain. According to Marrs and Albus, 1989, the climbing fibers are teaching signals. The signals are translated inputs by way of mossy fiber axons which also send signals through the pontocerebellar pathway, and even the spinal cord.

The mossy fibers transport much of the human sensory input signals to the cerebellum and are comprised of a sophisticated network of neurons. Two of the three input sources for the cerebellum are the mossy fibers from the spinocerebellar pathways and from the pons. The mossy fibers are comprised of many rosettes with small branches that extend through the granule cell layer. The rosettes are excitatory and eventually enter the Purkinje cells which are comprised of dendritic spines.

The Golgi cell is the largest cell in the cerebellar cortex and aids the mossy fibers in trafficking of signals. The Golgi cells in the granular layer are defined as inhibitory, and transform the mossy fibers.

The Purkinje cells host two electrophysiological spikes: simple and complex. Climbing fiber in the Purkinje cells can cause significant activating reactions. Simple spikes occur rather quickly compared to the complex which transpires at a rate of around 3 Hz. All of these activities are important, along with the mossy fiber inputs, to the functioning of the cerebellum. Constant trafficking of signals takes place between the mossy fibers and the granule cells. The granule cells of mossy fibers can be seen, microscopically, grabbing the terminals. Once the axons reach the molecular layer, they are considered as parallel fibers.

The parallel cells synapse and excite the Purkinje cells producing spikes. The integration of thousands of parallel fibers and the Purkinje cells provoke responses from the brain from synaptic contact. This activity continues to progress to the Purkinje cells,

where the parallel fibers enter the dendrites. This occurs by synaptic traffic through the mossy fiber rosettes containing sensory signals that travel indirectly to the Purkinje cells. Figure 1.3 shows the circuitry model of cerebellum.

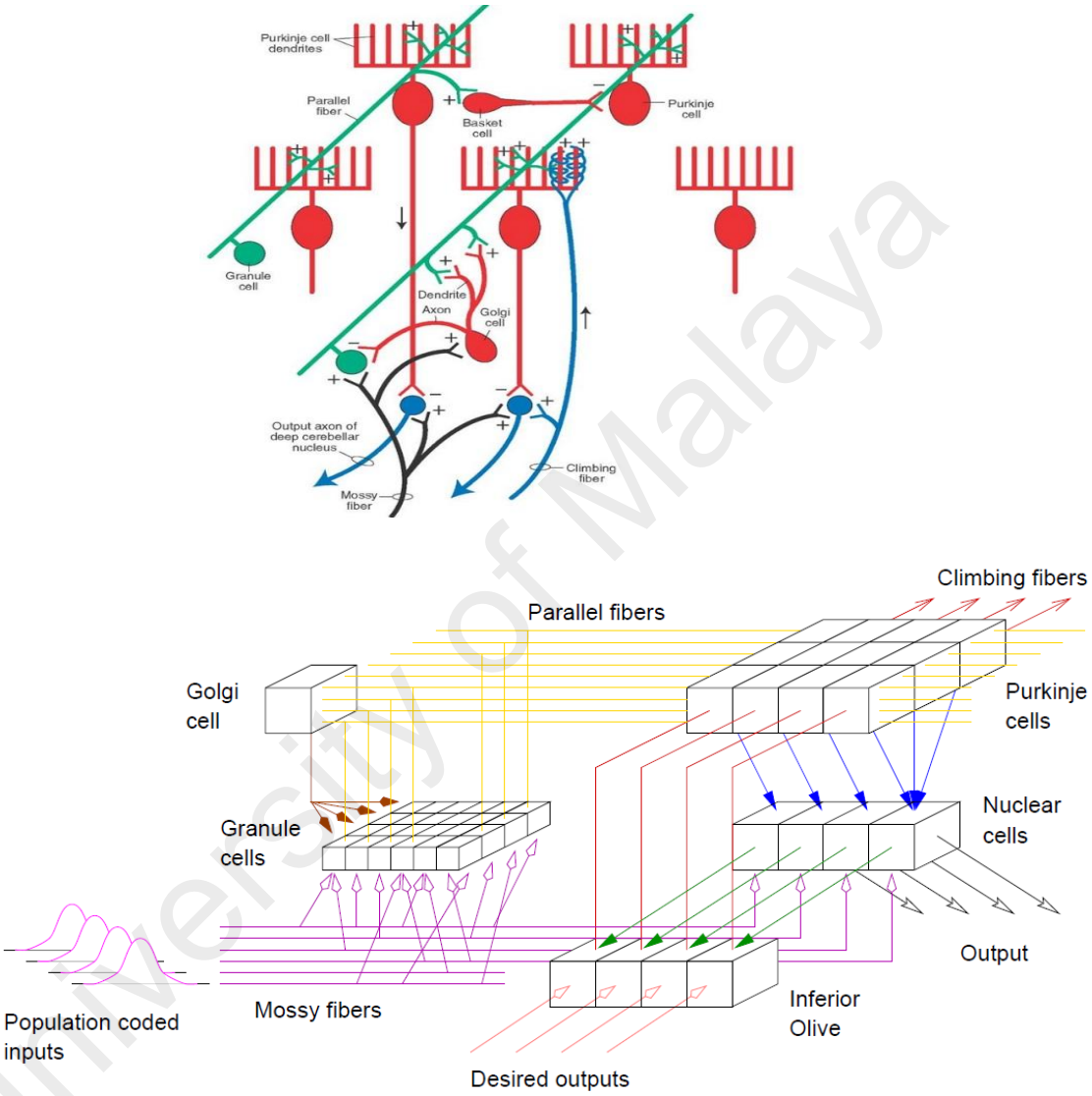


Figure 1.3: The circuitry model of cerebellum (Adapted from Schweighofer, 1995)

1.1.2.4 Cerebellar Cortex

The cerebral cortex of the brain is significant in simulations of human movements in robotics. The cerebellar cortex, through the cerebellar and brainstem nuclei, can direct

corrective action both at the cortical source through ascending pathways, and the spinal cord level through descending pathways (CNS, 2007).

According to a review of the model by Jason Carr, the developers of the cerebellar model obtained a robot that performs automatic learning by extracting the input layer functionalities of the brain cortex (Carr, 2012). According to Carr's research, the robotic arm initially performed with too much power. Thus, the movements required excessive power usage. The neurons in the brain transmit signals at speeds above 200 mph (Walker, 2002), a phenomenon that cannot be simulated with rechargeable batteries.

The solution for motor functions in robotics was to integrate traits from the cerebellum: To resolve the challenge, the researchers implemented a new cerebellar spiking model that can adapt to corrections and store their sensorial effects (CNS, 2007). Apart from that, it records motor commands to predict the action or movement to be executed. From here the researchers then expanded upon the automatic learning. The synergy between the cerebellum and the automatic control system enables robot's adaptability to changing conditions to make interact with humans. The biologically-inspired architectures in this model combine the error training approach with predictive adaptive control. Brain damage to the cerebellum has been known to cause a reduction in acceleration, imbalance in motion, an increased overshoot, and swerving in joint movements. Many have speculated that significant damage to the cerebellum will result in paralysis; however this is not true. It is at this point that technology has advanced sufficiently to allow integration of the brain and it's applications to be applied to robotics. The applications developed for simulations have been applied also to other disciplines. Breakthroughs in animated medical devices and flight equipment, among many others, are only a few results of successful programming languages modeled after the design and logic of the human cerebellum.

1.1.2.5 Cerebellum Role

One of the cerebellum functions is to control the motor cortex activity appearing in other brain parts, for example the basal ganglia, brain stem and spinal cord. The role of cerebellum is very important for any handiness, coordination and time arranging of almost entirely whole body plan, particularly top speed schedule (Albus, 1971). Signals which are generated out of the cerebellum modulate the amount of passage, beginning and ending changes, and just monitor the timing from a lot of tasks of coordinated sequences of transfer. Along on the vestibular group, in addition it may also help to make certain marked stability (Arthur c. Guyton, 1972). It could switch "clumsy" change instructions originating included in the motor cortex to efficient, fluid actions. Figure 1.4 shows the anatomical structure of cerebellum and its connections with the cortex.

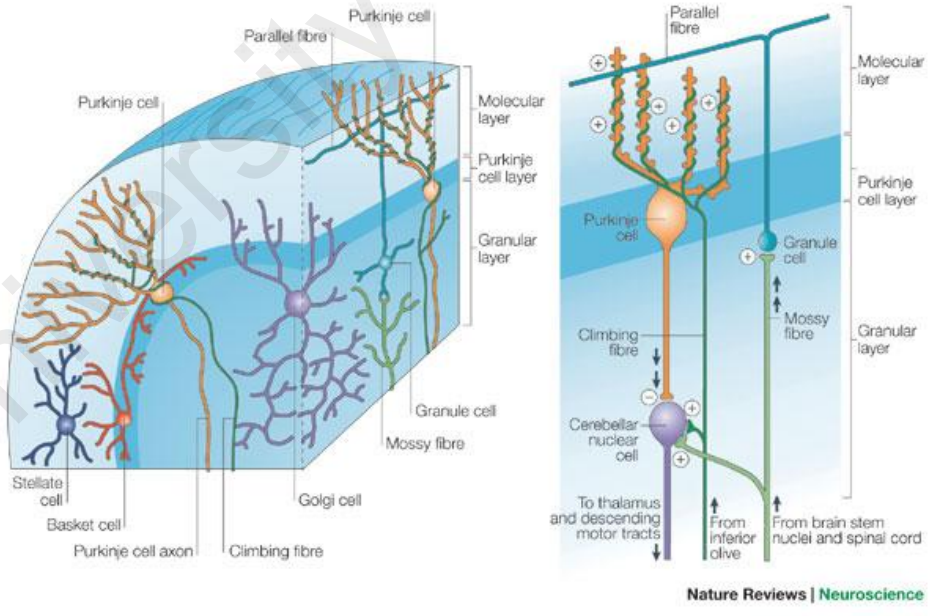


Figure 1.4: The anatomical structure of cerebellum (Adapted from Miall et al., 1993)

Another special cerebellum task is that it behaves as a sensory predictor. Sensory predictions that happen to be featured in coming of the usual delayed signals may be

used to manage motor systems (Miall et al., 1993). Sensory predictions can be significant for other tasks removed from motor control (Miall & Wolpert, 1996). There is certainly a group of evidence to support that the cerebellum can involve the majority of different functions.

1.1.2.6 Cerebellar Model

The cerebellar model consisted of 12 PC (Purkinje cells) neurons with 8560 PF (parallel fibers), 12 AB (ascending branch) and 12 CF (climbing fibers) inputs adapted from the model neurons described in detail by Nelson and Paulin (Nelson et al., 1995). All units were adaptive threshold spiking neurons. Some of the PF inputs were excitatory, representing GC (Golgi cells) activity that sampled from the state variables on the mossy fibers (MF). The receptive field of each GC was chosen as a radial basis function. The receptive fields overlapped and covered the input space uniformly. The remaining PF inputs were inhibitory, representing stellate cells present in the molecular layer. Other inhibitory interneurons were not modeled. Each Purkinje cells received one ascending branch input representing the summed input of many granule cells carrying information about the particular input variable to be learned. Correlations between the AB input and PF inputs caused learning at the PF-PC synapse resulting in feedforward predictive during the trial. After the trial ended, the throw result was evaluated and a binary error signal returned on the CF input for feedback error correction to the PC assigned to learn the release time. Trial was forgotten, if the CF indicates an error.

1.1.2.7 Cerebellar learning

In 1969 David Marr published his revolutionary theory of cerebellar cortex (Marr 1969), combining cerebellar physiology and anatomy with the machine learning methods of his day. Marr's foremost prediction was that PF synapses onto PCs would

undergo Hebbian facilitation when presynaptic PF activity was coincident with postsynaptic PC depolarization induced by CF input. Subsequent physiological experiments showed that simultaneous stimulation of PF and CF inputs causes LTD (long-term depression) but not LTP (long-term potentiation). Alternative Marr-like models therefore used LTD to confer learning capability to cerebellar cortex; LTD was widely regarded as the memory element for cerebellar motor learning (Garrido et al., 2012).

The cerebellar module has PCs, every single action as a quasi-feature detector in response to different MF input vectors. The GC to PC projection via the PFs will not be a completely connected group. When the IR states showed the robot to get centered, one of the PCs will respond more keenly comparing to another. Because of the binary nature of the GC amenable fields, the signal via a PF will either be zero or the weight through the PF-PC synapse. For this reason a PC will present between zero and six of its PFs active at any time. This combined with the PC-GC synapses results in a total of connections in cerebellar module, with no multiplication operations needed for a promote pass of the system. Whenever the cerebellar module doesn't generate a motor command deemed enough, a correction must happen to assist the module in becoming more competent.

The AB inputs may induce postsynaptic activity in the PC dendrite, that, when correlated with local PF input, leads to learning or LTP at the PC-PF synapse. Recent anatomical studies of the AB pathway indicate that AB-PC synapses are morphologically distinct from the PF-PC synapses. PF inputs adjacent to AB inputs are well situated to modulate or gate the AB response, creating ideal conditions for Hebbian-like facilitation or LTP (Garrido et al., 2012). In same behavior the cerebellar circuitry can function as the associative memory, by learning patterns of sensory and motor inputs presented by the mossy fiber pathway as they are projected onto the AB

and PF inputs. This is very complementary to the LTD result since the CF pathway is now free to assume other roles, including the representation of error signals. By assigning the AB and PFs to feedforward state prediction and CF input to feedback error correction or change of states. This learning hypothesis tested with *in **vivo*** experiments (Assad et al., 2001).

1.1.3 Neural Network

An artificial neural network is an interconnected group of natural or artificial neurons that uses a mathematical or computation model for information processing based on a connection approach to computation. A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

(1) Knowledge is acquired by the network through a learning process.

(2) Inter-neuron connection strengths known as synaptic weights are used to store the knowledge (Wen Chen, 2002).

The basic neuron model includes inputs, weights, a summation, an activation function, and an output as shown in figure1.5 (Frederick, 2005).

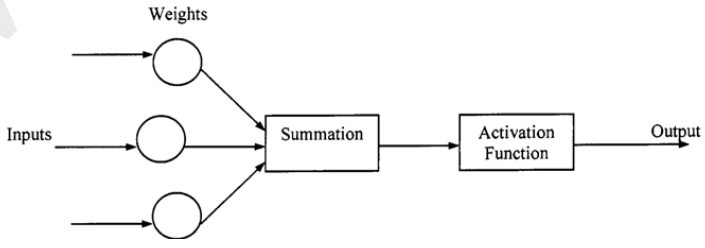


Figure1.5: Neuron Model (Adapted from Frederick, 2005)

The inputs can come from other neurons or external inputs and are multiplied by adjustable weights corresponding to biological synapses. The weights are determined by using a training algorithm. The weighted inputs are summed, and an activation function

determines the output of the neuron. The output of the neuron varies between zero and one. Neural network controllers have been successfully used in manipulator control, path planning, contract force control and grasping, multiple robot coordination and mobile robot autonomous navigation (Manish Kumar, 2004). Zeman (1997) has used neural network to control a robot having flexible joints, which is not applicable for the adaptive control. As well, they used neural network controller for robot manipulators in position control, force control and parallel-link mechanisms without requiring limiting conditions of linearity of parameters. Using neural network in presence of completely unknown manipulator nonlinearity, Kim et al. (2000) have proposed a neural adaptive learning approach for controlling robotic manipulators. Yang (2003) has proposed a neural dynamics based approach for real-time motion planning and collision avoidance for mobile robots in a dynamic environment. Kiguchi et al (2003) have used neural network as grasping force planner to emulate the grasping behaviour that humans show while manipulating an object. Manish Kumar (2004) proposed a strategy based on artificial neural networks to learn and optimize fuzzy logic rule base and membership function parameters to control multiple industrial robots working cooperatively in a fully automated manufacturing work cell. Neural networks are useful in the control of nonlinear multi-variable systems, are capable of learning from a training set, and parallel processing is inherent. However, a common criticism, particularly in robotics, includes the difficulty of extracting the knowledge base contained in the net, predicting results for cases outside the training set, and the convergence and training time.

A specific type of neural network, the Cerebellar Model Articulation Controller (CMAC), proposed by Albus (1975), has been successfully used to solve many complex and diverse tasks, ranging from autonomously flying aircraft to mobile robot. CMAC takes real-valued vectors and produces real-valued output vectors, can learn locally and generalize, can learn nonlinear function, has a relatively short training time, requires a

small number of computations per training iteration, and can be implemented in simple software and hardware (F. H. Glanz et al., 1991). However, besides its attractive advantages, the disadvantage is that to implement large and effective software neural networks, much processing and storage resources need to be committed.

Using hashing function is sometimes suggested to reduce memory size; however, the CMAC with hashing works well when data for only a small portion of input space needs to be stored.

1.1.4 Reinforcement Learning

Reinforcement learning is a major subset of machine learning. It is a computational approach to learning whereby an agent explores a complex and uncertain environment, perceives its current state, and takes actions that will eventually lead to a specific goal. The environment, in return, provides a reward reflecting the outcome of each action with respect to finding the optimal path to the goal. Reinforcement learning algorithms attempt to find a policy for maximizing a cumulative reward for the agent over the course of the learning process.

Figure 1.6 illustrates a reinforcement learning diagram that shows the relationship among the agent, state, action and reward in the environment.

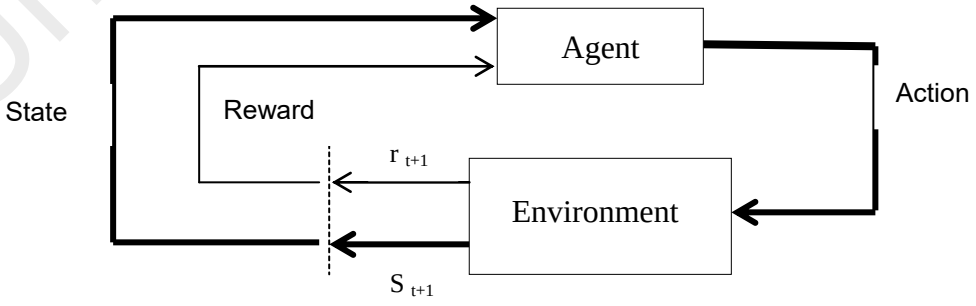


Figure1.6: Reinforcement learning diagram

Reinforcement learning mainly consists of three main threads: a) thread involves learning by trial and error, b) thread does not involve learning for the most part, but involves the problem of optimal control and its solution using value functions and dynamic programming and c) thread concerns temporal difference learning.

In reinforcement learning, an agent interacts with an environment for which neither the probability of transitioning to any state nor expected immediate reward is known. As a result, dynamic programming methods are not directly applicable for the reason that on one hand, the value iteration update cannot be computed, on the other hand, since computing the optimal policy requires knowing the probability of transitioning to any state and expected immediate reward, it is no longer sufficient to learn the state value function. However, by using temporal difference methods which synthesize dynamic programming with some other methods, the agent can learn the optimal action value function. Each time an agent in state takes an action, the reward it receives and the state to which it transitions can be sued to estimate the role of probability of transitioning and expected immediate reward in the update (Shimon A.W., 2007).

Q-learning is a popular temporal difference method, because of its simple computations per time step and also because it has been proven to converge to a global optimum. A value function is used to estimate how good it is for an agent to be in a given state. The goodness measure comes from the future rewards that can be expected, which depends on what actions are taken. Value functions are defined with respect to particular policies. Several papers focus on the application of Q-learning approach ranging from the control of robot and aircraft, game theory, and traffic signal control. El-Tantawy (2010) applied Q-learning in the optimal control of coordinated traffic signal. He developed a Q-learning based acyclic signal control system that uses a variable phasing sequence, which can minimize the vehicle delay. Therefore, except its advantages, a major problem with Q-learning is its inability to handle large state spaces.

With larger state spaces, longer training times are required since multiple visits of each state action pair are required for the agent to learn. Lookup tables are used to store Q-values with one cell for each state and action. Large state spaces also require impractically large amounts of memory.

1.2 Focus and Scope

This research is mainly focused on the development of the new cerebellar model articulation controller for function approximation and solving the memory size problem by presentation of a new architecture named Hierarchically Clustered Fuzzy CMAC (HCFCMAC). The model which is developed in this research tested to control simulated robotic arm.

Secondly, we proposed new concept of path planning and control strategy for biped robot to step over an obstacle in dynamic environment. For this purpose the new model of the step length planning strategy developed by semi online fuzzy Q-learning is proposed and experimentally tested to control simulated biped robot. Both of our models only tested in simulation environment.

1.3 Research Problems

1.3.1 Issues on Cerebellar Model Articulation Controller

Actually the important problem in developing a CMAC is the memory size and computing time. A small sized CMAC is able to have better information of the training data but at a diminished output resolution; while a large sized CMAC produces more accurate outputs. Some problem cause increase memory allocation size in CMAC. It is therefore judicious to devise an efficient memory allocation scheme for the CMAC network to assign more memory cells to the input regions that require higher output

resolution. So, it has two advantages as: (1) enhance the CMAC memory utilization, (2) provide more accurate outputs with a reasonable degree of data generalization. The memory allocation process in a CMAC-based system refers to the construction of the CMAC computing structure (i.e. the quantization functions) to define the output resolution with respect to the different regions of the input space.

1.3.2 Issues on Path Planning and Q-learning

The design of a path planning for biped robots in both indoor and outdoor environments is more difficult than for wheeled robots because it must take into account the abilities of biped robots to step over obstacles. The design of control strategy has a very challenging problem when dealing with the stepping over dynamic obstacle.

In real environment, neither the velocity of the obstacle is constant nor can it be predicted. Therefore, in control robot movement the most important problem which should be solved is to predict the next step and condition of obstacle. However, most of the previous works that have been adopted to solve this problem is focussed on footstep location planning concerning the goal position within the static environment or dynamic environment changes in predictable ways and it should be described in a time function. Q-learning algorithm has been developed to learn crucial parameters of robot motion.

1.4 Research Main Aim and Objectives

The main aim of this research is to develop the new cerebellar model articulation controller (CMAC) for improvement in the memory size problem and implement its application to control robotic arm, as well as to present a new concept of a footstep planning strategy for biped robot control in dynamic environment. To achieve these aims the following more specific objectives are set:

1) To develop the new model of cerebellar model articulation controller (CMAC) to reduce memory size, this model called Hierarchically Clustered Fuzzy CMAC.
2) To apply the proposed model to control dynamic systems such as robotic arm.
3) To investigate and improve the foot step planning approach based on Semi Online Fuzzy Q-learning algorithm.
4) To evaluate the combination of proposed CMAC model and foot step planning in control of biped robot step over an obstacle in dynamic environment.

1.5 Fundamental Research Questions

The following research questions are sought and act as guidance to conduct this research at various stages:

Q1. What are the issues that deter the development of cerebellar model articulation controller?

Q2. What are the major features of cerebellar model articulation controller?

Q3. How do the inputs effects on memory size in CMAC?

Q4. In what ways the proposed CMAC model can solve the memory size problem and minimal modeling error?

Q5. How is possible to control complex system such as robotic arm with the proposed model?

Q6. How can footstep planning be developed based on Semi Online Fuzzy Q-learning?

Q7. Which type of control strategy is suitable for biped robot to step over an obstacle?

Q8. How evaluate the proposed model of CMAC able to predict swinging trajectory?

1.6 Research Contributions

The developed CMAC in this research is able to solve the crucial problem on memory size. Due to the capability of this model, it can be used in complex control systems such as medical robot which needs more accuracy and sharp response.

In this research the proposed model is named as Hierarchically Clustered Fuzzy CMAC (HCFCMAC). HCFCMAC employs hierarchical clustering for the non-uniform quantization of the input space to recognize important input and allocate more memory to the areas. The effectiveness of this model is evaluated by comparison controlling robotic arm with this model and Fuzzy CMAC.

Another contribution of this thesis research is in representing a concept of footstep planning strategy that is based on Semi Online Fuzzy Q-learning. The learning agent should choose action pair of step length and duration time, as well as the corresponding maximum step height. After the training phase, the robot is able to adapt the step length, step duration time, and step height simultaneously by using a suitable system for the step over obstacles within specified ranges. The bipedal locomotion during the single-support-leg phase is studied as a tree-like topology. On the other hand, control strategy for biped robot stepping over obstacle is considered. We used two types of control strategy, namely the high level controller which is based on the Semi Online Fuzzy Q-learning algorithm which includes the footstep planning, foot trajectory and joint angle profile generation; and the low-level control based on the developed HCFCMAC, which allows both to learn the predicted swinging trajectory and to control the tracking of

these desired trajectories. The simulation results show that proposed approach is operational in the case of a robot without feet moving in a sagittal plane.

1.7 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 covers the literature review about cerebellum model articulation controller, Q-learning, robot control history, dynamic environment and reinforcement learning. This chapter also discusses about human.

Chapter 3 describes the neurophysiological aspects of the human cerebellum and the development process of the human brain. The CMAC structure optimization for function approximation problem is also studied. How the structural parameters influence the required memory size and approximation quality is discovered, and the parameters optimization algorithm is developed. Q-learning approach and its application is also focussed and discussed in this chapter.

Chapter 4 deals with controller design for solving memory problem. This chapter outlines the basic principles of the CMAC neural network, followed by a brief important review and describes in detail the proposed Hierarchically Clustered Fuzzy CMAC architecture. The proof of learning convergence of the HCFCMAC network is established by applying this model to control robotic arm. The proposed CMAC learning algorithm and mathematical model of 3-DOF robotic arm are obtained in this chapter.

Chapter 5 discusses on the development of footstep planning for biped robot in dynamic environment, based on Semi Online Fuzzy Q-learning algorithm. The control strategy for biped robot stepping over dynamic obstacle is investigated.

Chapter 6 outlines the mathematical model development of the five-link biped robot in the sagittal plane. The methodology for developing the equations of motion based on Lagrangian formulation is presented in detail. The high-level controller and low-level

controller are designed respectively. This chapter also contains the simulation result of developed CMAC to predict trajectory for swinging leg and the highlights of findings.

Chapter 7 finally concludes the overall research conducted; and further research direction is suggested.

CHAPTER 2

TIMELINE OF ROBOTICS CONTROL

2.0 Timeline of Robotics Control

2.1 Timeline of Development of Bio Robot

In 1921, science fiction writer Karel Capek of the Malé Vatonovice presented the world with the play Rostrum's Universal Robotsand with the first known occurrence of the word 'robot' in world history (Legacy, 2011). The play was debuted in Prague, Czechoslovakia, where the root term 'robota' was translated to mean 'forced labor' (Farlex, 2013). In Capek's production, and in his own perception, a 'robot' represented a slave, an artificial person, or at best, a silent partner - a source of output without the ability or desire to give intellectual input. Capek's perception was typical of 1920s Czechoslovakia, when hardcore industrialism, factory work, and war permeated the minds of Slavic societies. In addition, many years before Karel Capek in 18^{th} and 19^{th} century there are some examples of human crafted. For example, Marry Shelly published "Frankenstein or The Modern Prometheus" in 1818 and of course there have always been the searches for explaining what intelligent behavior is even dating back to the old Greeks.

In the next decade, depictions of a robot had evolved from a factory worker to the monster, as portrayed by Boris Karloff in a 1935 film (Karloff, 1935). During the same year, mathematician Alan Turing produced 'the Universal Turing Machine', which is considered one of the world's first official computers. By 1945, he was convinced by his studies that computable operations could mimic mental functions performed by the human brain. In 1950, Turing published Computing Machinery and Intelligence, which addressed the computability of his neural networks (Hodges, 2007). According to a biography written by Andrew Hodges, Turing's works had shown the stunning power of the computable in mechanizing expert human procedures and judgments.

From 1941 onwards, Turing had also discussed the mechanization of chess-playing and other 'intelligent activities' (Hodges, 2007). Alan Turing's contributions to computer programming inspired subsequent research and the development of Artificial Intelligence.

From the 1930s to the 1950s theoretical reasoning had matured to the dimension of mathematical logic. Until the Age of Computers began in the 1940's, explorations of artificial intelligence were few in number and not much in depth. Astute atheist, John McCarthy who is considered the 'Father of Artificial Intelligence', contributed breakthrough programming languages and computer timesharing. McCarthy's LISP collection of programming languages provided the foundation for modern programming languages; including if-then-else statements (John M., 2012). The *LISP 1.5 Programmer's Manual* was published in 1965. Since then, development of machine Artificial Intelligence information resources and research has been mutually inclusive to the rate of growth of technological advancement.

By 1950, robots had begun to lose their notorious, monstrous images and began to take on attributes of a friend or companion. Isaac Asimov's *Robbie,* from his book entitled *I, Robot*, interacted with humans in a social setting (Asimov, 1950). A comparison of old and new technologies, such as analog voice communications systems and Controller technologies reflect the significance of enhancements that have been made in the past few years. Asimov's science fiction creations were reproduced in 2004, by Jeff Vintar and Akiva Goldsman in the film *I, Robot.* In the 2004 version, *Robbie* evolved to *Sonny,* a robot accused of murdering a scientist.

Today, the progression of robotics through the research of Artificial Intelligence is remarkable. The development of the computer has transformed the theories of great thinkers to virtual reality. In a sense, Capek's robots were indeed a precursor to

cerebellar modeling in theoretical fields of study, particularly in studies of robotics applications in Artificial Intelligence.

Bio-inspired robotics involves studying biological systems as well as identifying and analyzing the mechanisms that may solve a problem in the engineering field. Artificial intelligence (AI) is technology in computer science that involves studying intelligent machines and software. A general definition of artificial intelligence is 'the study of *the synthesis and analysis of computational agents that act intelligently'* (Poole & Mackworth, 2010). Technology and systems solutions are a bit easier to assess because contributions may be assessed by output. Because the research is designed to provide a predetermined output, its success can be measured by the provisions results reports and data**.** An intelligent system perceives its environment and takes actions which can maximize its chances of success. A prime ability of humans, intelligence—the sapience of *Homo sapiens*—can be precisely described that it can be simulated by a machine. The development of computer hardware and applications pave the way to greater capacities in research. Medically-inclined researchers have produced valuable tools for the physically handicapped to include motorized wheelchairs with many substitute functions and artificial limbs that replace arms and legs. This chapter covers main aspects on human thought and behavior processes, artificial intelligence, brain-computer interface, cerebellum, cerebellar model articulation controller, Q-learning theory, reinforcement learning and dynamic environments as in the following sections.

2.2 Human Thought and Behavior Processes

Arbib (1964) encouraged research of the relationship between the brain and machines to the benefit of artificial intelligence studies. His studies were based upon action and the perception of action by schema theory and neural networks. Arbib's research is on the evolution of brain mechanisms for human language, pursuing the

Mirror System Hypothesis in linking language to the properties of the mirror system for grasping with neurons active for both the execution and observation of actions, as to why human brains can acquire sign language as readily as speech (Arbib, 1964).

As a mirror neuron fires both when an animal acts and when it observes the same action executed by another, the neuron mirrors the behavior of the other, as if the observer was itself acting. Work on a range of simulations of the brains and behaviors of frogs (Rana computatrix), monkeys and humansis paralleled by work in biologically-inspired robots. Recent work includes detailed modeling of hand control, mirror neurons and sequencing as part of a program to determine ”What the Macaque Brain Tells the Human Mind” (Arbib, 2010). The Mirror System Hypothesis for the evolution of the language-ready brain evokes a path for evolution of brain mechanisms beyond the mirror system for grasping, with new processes supporting simple and complex imitation, gesture, pantomime and finally protosign and protospeech. “Language readiness” (biological) is distinguished from “having language” (cultural) with the evolution of the language-ready brain and language involving several stages: grasping; mirror system for grasping; simple imitation system for grasping, shared with the common ancestor of human and chimpanzee (which allows imitation of complex “object oriented” sequences but only as the result of extensive practice); complex imitation system for grasping (which allows rapid imitation even of complex sequences, under appropriate conditions); protosign, succeeding the fixed repertoire of primate vocalizations to produce an open repertoire for communication; protospeech, the open-ended production and perception of sequences of vocal gestures, a process of cultural evolution in *Homo sapiens* providing full human languages. Protosign formed scaffolding for protospeech and they interacted with each other to support the evolution of brain and body that made *Homo sapiens* “language-ready”. This progression causes

the end of the simple model to fully explore the lessons of Simulation of Animal Behavior for computational neuroscience and biologically-inspired robotics.

Human thought processes which control movement and behavior can be researched from different social and philosophical viewpoints. Brain Computer Interface programming requires theoretical insight on what drives human thought and stimulates movement. For the purposes of this study, cognitive considerations of human thought processes are a part of the Reinforcement Learning focus. Cognitive perceptions of human thought and consequently human actions include the thought processes which create tendencies, likes and dislikes.

If the human thought processes can be defined and classified, then human actions can be understood through these thought processes and thus, specifically predicted and even evoked. Some researcher presented a model of the human consciousness that consists of the Id, the ego, and the super ego which drive human behavior. Human behavior is based upon processes of the conscious and subconscious mind. His classification of the unconscious includes elements of thought that we are not directly aware of, where the ego constantly strives to keep the Id content. It is here that external events obtain power to affect the thinking and subsequent actions of the individual.

They insisted that the human mind is divided in parts which possess unique functions. *"The ego is not sharply separated from the id; its lower portion merges into it. But the repressed merges into the id and is merely a part of it. The repressed is only cut off sharply from the ego by the resistances of repression; it can communicate with the ego through the id". His theory acknowledges the capabilities of humans to resist or repress unfavorable thoughts or urges. According to (Freud, 1932) the repressed thoughts or urges are still present, constantly seeking to be expressed. If this is true, it indicates that most human thoughts and movements are voluntary and deliberate.*

Many theologians refute Freud's claim that the three divisions of the human consciousness exist and interact on such a level. Research has been conducted to disprove the existence or the dominance of the Id in Freud's model. However, Freud's theory, along with other assumptions regarding human motivation, has been continuously used to develop strategies in other disciplines that require compliance. A deficiency in quality decision making can result in poor design, accidents due to negligence, or bad judgment. Regardless of the level of technology, the quality of human decisions made by people such as the pilots, designers, and technicians affect the quality of the experience.

The motive for such strategies is performance, and, productivity. Skinner paralleled Freud's theory in his perspective of conscious versus unconscious control of behavior and of selection by consequences (Overskeid, 2007).

According to Allen Newell's Unified Theories of Cognition, the definition of intelligence is 'the degree to which a system approximates a knowledge-level system. The next level makes decisions based upon both knowledge and rationality. According to William Cohen (1994), the basic insight upon which the Explanation-based technique is founded is that the EBL (Explanation-Based Learning) can be considered learning control knowledge to prove a theorem. By providing a richer representation for such control knowledge, more general rules can be learned: in particular, by providing looping constructs, rules which generalize number can be expressed; and by providing conditional branches, rules learned from different training examples can be combined (William Cohen, 1994).

2.3 Artificial Intelligence

In the modern period, which began approximately in the late 70s, valuable contributions were made to the study of Artificial Intelligence by studies such as the

MYCIN experiments of Stanford University and the MECHO system. The MYCIN system contributed to the development of immunizations for infectious bacteriological diseases; and the MECHO, solutions for Newtonian machines (Winograd, 1971). Until the emergence of computer, most typical AI projects were not possible. From the 60's to the 70's, the Romantic period was filled with studies of programming languages. Terry Winograd published Understanding Natural Language in 1972. Winograd developed the SHRDLU language program at MIT from 1968 to 1970. Considering the importance of the cerebellum in motor control, it is not surprising to find that there are numerous neural pathways connecting it with the cerebral cortex, brainstem nuclei, spinal cord proprioceptive tracts, and the vestibular system. The information-conducting fibers to and from the cerebellum are passed through the superior, middle and inferior cerebellar peduncles (Jordan et al., 2007).

Other important studies focus on methods for reusing requirements in different ways. Reusability in computer science and software engineering refers to the tendency that a part of source code can be reused to add new functionalities with either slight or no modification. Reusable modules and classes can decrease implementation time, increase the possibility that prior testing and use has eradicated bugs and localizes code modifications when a change in implementation is needed. In one study about reusability of software the authors described software reuse to be the only practical approach that can produce the productivity increase and the quality that the industries need. The advantages of reusability are better when the abstraction level is raised and not only through requirement reusability, but also through designs and specifications reusability. There are several approaches to requirements reusability, but the most successful method of requirements reusability should address the three major approaches: text processing, knowledge management and process improvement (Cybulsky and Reed, 2000).

In 1979, the Association for the Advancement of Artificial Intelligence (AAAI) was founded to increase scientific understanding, encourage research and responsible use, and increase public awareness of artificial intelligence (AAAI, 2013). Headquartered in California, USA, the organization addressed the needs of global scientific communities. Since its inception, major accomplishments have been made through computer science with the aid of computer software and other resources such as the Internet.

The history of robots was researched back to Czechoslovakia in the early 1900's, to the production of *Rostrum's Universal Robots* by Kapek (1921). The history progresses through the invention of the first computer and computer languages by Turing and McCarthy (Hodges, 2007). A brief tour of the evolution of technology is rich with historical developments of robotics, the CMAC simulation of the human cerebellum to Q-Learning and greedy policies, to arm and leg motion in dynamic environments, to real time and semi online Reinforcement Learning. Today, robotics is much more than theatrical monsters. Scientists have discovered the power and potential of wireless animation and thus have expanded their research to Brain Computer Interfaces and mechanisms that control objects as small as a hearing aid or as large as an aircraft carrier.

The requirements for engineering generate corresponding specifications, which are detailed lists of needs to meet the design requirements. In the requirement elicitation process, requirements for a new system are retrieved for the project and used to create a requirements specification document. The quality and safety of the specifications is mutually inclusive of the sufficiency of the requirements. To be of any value, once defined, the specifications must be adhered to during requirements development and management. "For projects that include complex electronics, the requirements decomposition is higher than the device level. When requirements documents exist for complex developments, it may still be less-specific than desired. In particular, the

document may be lacking in complete interface specifications, requirements imposed by technology constraints, and testability requirements". During this stage of development, errors may occur due to flaws in the initial requirements or deficient corresponding specifications.

The human brain has been integrated with computer programming software to achieve the necessary integration mental human processes and controlled automation. According to this research, a co-integrated system can be represented in an error correction structure which incorporates changes and variables. This error correction structure provides the framework for testing of co-integrated systems. The structure resembles the error correction paradigm of the human brain that is presented by (Wisse et al., 2005).

2.4 Cerebellar Model Articulation Controller (CMAC) and Q-Learning Theory

The interest in the processes of the human brain stems from efforts to simulate in electronic devices, particularly the biped robot. The robotic simulations of human movement are extracted from the model of the human brain, particularly the cerebellum. The structure of the cerebellum is mimicked to accomplish controlled motion by robots such as walking, running, and tracking. Researchers work on the structure of the cerebellum and its function. Bailey (2013) discussed the anatomy of the brain and its function in processing signals significant to motor functions. Q-Learning logic is created by simulating signal responses from the human brain. According to Bailey in Latin, the word cerebellum means little brain. The cerebellum is the area of the hindbrain that controls motor movement coordination, balance, equilibrium and muscle tone. The cerebellum is comprised of white matter and a thin, outer layer of densely folded gray matter. The folded outer layer of the cerebellum (cerebellar cortex) has smaller and more compact folds than those of the cerebral cortex. The cerebellum

contains hundreds of millions of neurons for processing data and relays information between body muscles and the cerebral cortex that are involved in motor control.

In 1971, Electrical Engineer, James Albus published his theory of the cerebellar function, 3 years after David Marr published his theory of the cerebellar cortex (Albus, 1971). From here, Albus continued his research and provided the CMAC neural net computer. Prior research of cerebellum structure and its functions has led to new findings about its cells' function and composition, an internal model of connection between neurons, as well as the learning structure of the human brain. New research has uncovered other functions of the cerebellum in addition to motor movement. Biological links to cognitive and human speech processes have also been found. In the human conscientiousness Id, the regulation of emotions such as pleasure and fear are associated with signal transmittal in the cerebellum.

Together, Albus and Marr produced a theoretical model of the cerebellum that is used today in progressive robotics research and neurophysiology. Even medical researchers have benefitted from the accomplishments by using the animated processes to create replacement limbs and automated devices to aid the handicapped. The approach used to create a biped robot simulation is based upon Albus' CMAC model.

2.5 Reinforcement Learning Theory

Reinforcement learning has grown from methods to train animals to teaching robots to perform actions through control sensors and rewards. Researchers have acquired to solicit desired responses of movement; independent actions performed in dynamic environments. A general observation of optimal policy manipulation is introduced in "Neural Optimal Control of Robotic Manipulators with CMAC Networks" (Lin et al., 2005).

The basic idea of reinforcement learning control is shown in figure 2.1. The key point is that the learning system receives a teaching signal that only evaluates the "goodness" of the current state. It does not provide the correct "desired" output of the controller. Barto (1995a) note that reinforcement learning involves a conflict between *exploitation* and *exploration*.

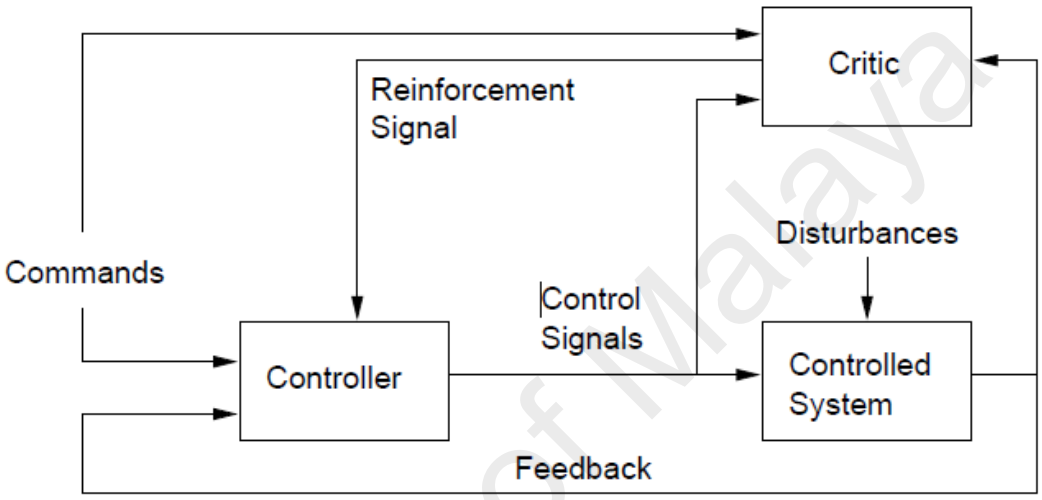


Figure 2.1: Basic reinforcement control scheme (Adapted from Barto, 1995b).

The inverse kinematics of a robotic manipulator is nonlinear and may not have a closed form solution. Iterative and geometric solutions are not sufficient if the manipulator is complex.

Q-learning is an off-policy Temporal-Difference (TD) control algorithm and it is believed to be one of the most important breakthroughs in reinforcement learning. Q-learning is an off-policy control algorithm, whereby it is able to learn the optimal policy while following a slightly randomized policy that ensures exploration. Furthermore, when said that Q-learning is a Temporal-Difference control algorithm, it mean that it approximates its current estimate based on previous learned estimates (Dayan, 1992). This process is also called bootstrapping. Q-learning has one more advantage in that it can be applied online. Online learning occurs when the agent learns by moving about

the real environment and observing results. In contrast, in offline learning the agent learns by making use of typically large sets of training examples. Once the training session is over, an offline learning agent no longer changes its approximation to the target function. In other words, post-training queries to the system have no effect on the system itself, and thus the same query to the system will always produce the same result. By using an offline learning method it betting that this set of training examples is all needs to handle any situation the system may encounter in the future. The main disadvantage with offline learning is that it is generally unable to provide good local approximations to the target function.

The most important characteristic of online learning is that soon after the prediction is made, the true label of the instance is discovered. This information can then be used to refine the prediction hypothesis used by the algorithm. Online learning has several advantages. For starters, online learning tends to require less memory resources than offline learning, since offline learning often processes training examples in batches, which can be very numerous. Furthermore, an offline learning system will not be able to adapt to changes of the environment unless a new training session takes place. For instance, the system may be in charge of controlling a robot. Eventually, some of the parts in the robot may wear out and the actuators response may be slightly different. An online learning system may be able to automatically adapt to this change while an offline learning system may not.

The ultimate goal of a Q-learning agent is to come up with a control policy. This policy consists of a group of state-action pairs. In its most basic form, this policy has a state-action pair for each possible state. The action associated with each state represents the “best” action that can be taken from that state. By “best” action we mean the action that is expected to provide the largest reward in the long run. In a more sophisticated form, this policy has only some examples of state-action pairs and the other pairs are

inferred from these examples. Once Q-learning generates a Q-function, it is very simple to come up with its associated control policy. All that needs to be done is to analyze each state in the Q-function and select the action that maximizes the expected reward. Berenji (1994) published work was accomplished through fuzzy Q-Learning in a dynamic environment. His writing addressed fuzzy reinforcement learning (FRL) by fuzzy logic rules. The reinforcement learning method for this study used the Semi Online Fuzzy Q-Learning method. *According to Newell et al. (1994) the problem space consists of a problem solving strategy that is divided to four parts:*

(1) *States Knowledge Level*

(2) *Lists operators of change; Symbol Level*

(3) *Limitations of the operators*

(4) *Control knowledge*

The amount of memory needed to carry out the commands remains the issue; however approaches to solving the space and memory problems have had good results. The knowledge is extracted from a data base of logical functions based upon the desired outcome. The knowledge level has little association to the implementation of the policy. The control knowledge is the most important factor of the group.

According to Miller, Glanz, & Kraft (1990) the CMAC may serve as an alternative method to back propagation. As technological capabilities increase, the platform for critical research of motor control in several areas of artificial intelligence is enhanced to benefit society in many ways. Military aircraft with remote capabilities can engage in combat without a human pilot. Wheelchairs for the elderly and the handicapped can be upgraded to enhance safety and capabilities.

Much of the research for the Semi Online Reinforcement Learning was retrieved from Carreras et al. (2003) "Semi Online Neural Q-Learning for Real-time Robot". An

interesting portion of their study is the result of a robot underwater following a target. The advantages of this approach are:

a) Brief computing time

b) Footstep planning operational for predictable and unpredictable dynamical obstacles.

Watanabe (2009) introduced a study of Q-Learning and parallel structure. The study supported fuzzy multi-grain configured modular fuzzy models that improved the performance of the robotic movement. Fuzzy Q-learning is one of the promising approaches for implementation of reinforcement learning function owing to its high ability of model representation. However, in applying fuzzy Q-learning to actual application, the number of iterations for learning also becomes huge as well as almost all Q-learning applications (Watanabe, 2009).

Righetti & Schaal (2012) produced testing for optimal distribution of contact forces for inverse dynamics. Three tests for normality that can be used for multiple regressions are listed as:

- D'Agostino-Pearson Test
- Shapiro-Wilk Test
- Kolmogorov-Smirnov Test

In these types of test the robot is controlled by task space using *n* number of tasks with a resolved acceleration scheme and hierarchies through null space projection. When the robot moves, the motion of the base of the robot whose acceleration is computed uses a PD controller and a posture control around desired joint positions. When one leg is unloaded, the position of the foot is also controlled (Righetti & Schaal, 2012).

The research of Teddy, Lai & Quek (2007) was adapted to develop the basis of the control scheme. Their research proposed the Hierarchically Clustered Adaptive

Quantization model as an upgrade from the CMAC. The layered cell activations in the original CMAC network contribute to two significant computational objectives:

(1) A smoothing of the computed output

(2) An activation of similar or highly correlated computing cells in the input–output (I/O) associative space

These modeling principles are similarly conserved in the single-layered model of the CMAC network through the introduction of a neighborhood-based computational process (Teddy et al., 2007). Much of the empirical results of this study are based upon Teddy, Lai, and Quek's method using the CMAC model for inputs and outputs.

Ron Larson and Betsy Farber's (2003) "Elementary Statistics: Picturing the World" provides tremendous insight and guidance for the quantitative analysis of the data sample. The Q-Learning algorithms are analyzed using hypothesis testing and correlation. The study consists of bivariate and multivariate co-integration. Scatter plots present correlation between the variables. A more precise way to measure the type and strength of a linear correlation between two variables is to calculate the correlation coefficient (Larson & Farber, 2003).

Becket and Chang (1967) made a great contribution to swing leg trajectory theory by expanding the research of related pendulum swing motion which was first considered a century earlier. The study is based upon early theories of pendulum motion and ballistics. The pendulum hypothesis was founded upon rejection of joint torque occurrences during swing motion of the leg. Recent studies have been conducted to test the opposite of Becket and Chang's findings, particularly regarding the ballistic model. In a project by He et al. (2009) a trajectory optimization method was presented to realize the desirable punch trajectory under multiple constraints on velocity, acceleration and jerk of the servomotor.

Kim and his colleague in 2010 produced a study with valuable examples of a matrix function in regard to the turning and the walking speed of the biped robot. According to their study specifically in fast walking, the touching of the ground of the swaying foot before a normal landing provides more impact to the body than in slow walking. Thus, the swaying foot needs be sufficiently high, which can be implemented by adding a related term in the cost function (Kim et al., 2010).

McGeer introduced the concept of the passive walker (McGeer, 1990). His studies reflected the works of Becket and Chang (1967) with regards to pendulum and ballistic theory. The study provided an example of a two legged walker's procession down a slope using the pendulum theory of motion. The walker was expanded to include bending knees and a stable limit cycle. Kuo, at the University of Michigan researched the walker in 3+D, in which he added a degree of freedom and eliminated the stable limit cycle. The dynamics and balance of the legged robot may engage in fuzzy control. In this instance, the design of robots with more than two legs may be designed from the cognitive models of animals, which naturally travel through unpredictable terrains.

A study by Suzuki & Takahashi (2011) predicted the coordinates for the robot's steps, taking into consideration the obstacle settings and successfully plotting a planned avoidance path. The angles of the ankles and joints and the size and shape of the feet are of tremendous importance in the design. The feet are ideally constructed of sprung sole plates with a weight that is compatible to the height and weight of the robot. The degree of freedom chosen for the joints of the biped robot for this study is six. Not all footstep planning must be based upon obstacle avoidance. Robots that are to be used indoors usually must be configured to avoid becoming trapped in a dead end.

Nakayatna and Watanabe (2009) suggested using piezoelectric ceramic sensors to improve the walking motion for a biped robot. The ceramics are used in many electronic products because of the properties. Researchers have agreed that some significance

exists in the properties of the material used for the sensors. The properties of the ceramic material are ideal for sensors as well as control mechanisms for other robotic devices. Currently in the literature there is a wealth of studies that focus on the improvement of requirements specifications through a variety of methods. Requirements specification is the foundation for the whole software development process. It is essential that requirements are of quality and satisfy user needs.

Regarding the reusability aspects, Krueger (1992) produced a study that describes various approaches to reuse. He evaluated the effectiveness of reuse techniques in terms of cognitive distance. He determined that the most effective technique was automation of the abstractions in a reuse technique to an executable implementation.

2.6 Dynamic Environments

The dynamic environment changes as the robot navigates through the terrain. As opposed to static environments, the robot must be capable of adaptation and contend with obstacles. Control laws may be instated to address the changes across the environment.

The basis for optimization policies in dynamic environments is to solicit an appropriate response to changes from the robot. Collision avoidance, traditionally considered a high level planning problem, is effectively distributed between different levels of control, permitting real-time robot operations to occur in a dynamic environment. This method has been extended to moving obstacles by using a time-varying artificial potential field (Khatib, 1986). By simply walking through an environment, a robot may experience less problems adapting to changes than a robot that is running. When a robot must spend a significant amount of time supported by a single foot, the contact that foot makes with the ground is very important for stability, requiring that the robot properly assesses foot placement to maintain steady balance

during locomotion (Chestnutt, 2007). However, the dynamic environment may change constantly, randomly, and frequently, making it difficult for even a walking robot to navigate successfully through the entire course. For this reason, population-based searches (It is learning algorithms that solve the theory problems) must make room for variation. Some experiments draw from similar previous research to obtain solutions. The solutions must be applicable to the state of the robot after it has encountered the change.

2.7 Summary

Robots and devices are capable of making control such as human decisions without having a solution for each problem fact which could be encountered. Currently in CMAC literature there are a number of studies that focus on the improvement of design specifications through a variety of planning methods. Requirements specifications represent the foundation for the computer interface development process. Researchers predict that, despite problems associated with dynamic environments and Reinforcement Learning, robots will eventually be able to outperform humans in a variety of platforms if the problems can be resolved. Such challenging research is sought after.

CHAPTER 3

Structure of Cerebellar Model Articulation Controller (CMAC) and Q-Learning

3.0 Structure of Cerebellar Model Articulation Controller (CMAC) and Q-Learning

3.1 Introduction

A thorough simulation of the human brain must include a simulation of its neurons. An artificial neural network is designed to accomplish this task, where the journey to optimization of a learning rule begins. In 1943, McCulloch and Pitts published A Logical Calculus of the Ideas Immanent in Nervous Activity, one of the first known models of the neuron. The model neuron was extremely basic, lacking learning capabilities. In 1958, Frank Rosenblatt expanded upon McCulloch and Pitts' model of a neuron with the 'perceptron', which had more dimension than the neuron such as pattern recognition. The perceptron neuron passed through association processing units and responded to excitatory and inhibitory signals. In 1969, Minsky and Papert reviewed the single layer perceptron but were not impressed with the pattern recognition features. However, the learning aspect of McCulloch and Pitts' neural network quickly expanded to a wide scope of learning strategies such as CMAC. The back-propagation rule was introduced by Hinton and Williams (1986) and developed by Rumelhart et al. (1986). The concepts of forward pass and backward pass through the perceptron, along with back propagation of error, gave the neuron the trainability to output efficiently. The human biological neuron structure is comprised of three types of neurons: unipolar, bipolar, and multipolar according to the number of processes. Thus, we have three types of artificial neural networks:

- Perception – Single Layered Network
- Multi-layer Feed Forward Network
- Recursive Network

For the purposes of locomotion in robotics, the neurons are separated by function, in which the sensory neurons address perception and motor coordination.

Today, neural networks can be used in most applications and do not require a stated problem. However, a neural network is defined as a 'computer-based decision-making system for complex data sets comprising processor nodes interconnected by weight, simulating a biologic nervous system' (Farlex, 2012). Dr. James Albus created the CMAC in 1975. According to Albus:

"I developed a new type of neural net computer: the CMAC. It has several significant advantages over other neural nets. One is that CMAC learns non-linear functions orders of magnitude faster than other neural net algorithms such as back propagation".

CMAC is also much more efficient in execution, so that even on 1975 era computers it was able to compute in milliseconds (Albus, 1975). The schematic of CMAC that was developed by Albus is shown in figure 3.1.

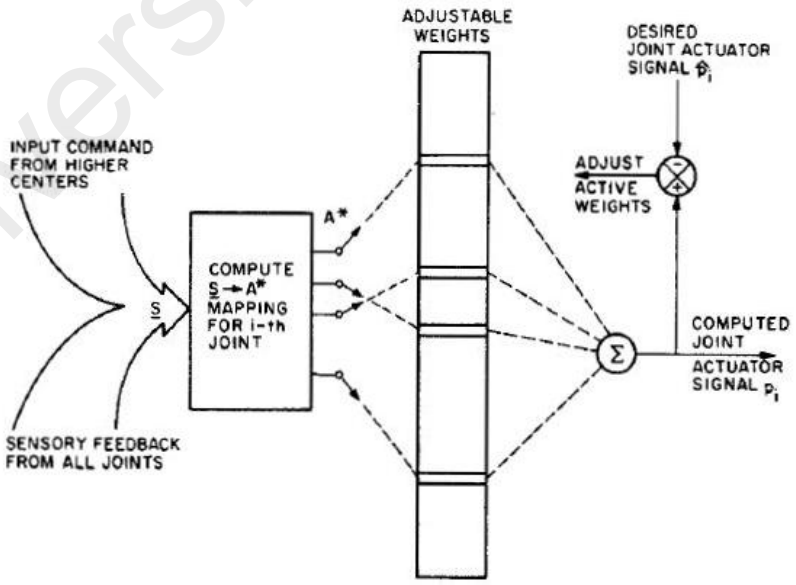


Figure 3.1: James Albus CMAC System Block (Adapted from Albus J.S, 1975)

The modern artificial neural network is not a computer processor; but rather its design is a replica of the applicable microprocessor. Thomas Miller (1994) conducted a study of the CMAC model for real time control of a biped robot. According to Thomas: A hardware CMAC neural network design was developed which provides sub millisecond response and training times for typical CMAC neural networks with tens of 16 bit inputs, 1-8 16 bit outputs, and 512K 16 bit weights. These specifications are suitable for many real-time control problems. CMAC neural network can solve control problems in both dynamic and static environments, as a memory bank, the network stores vector responses to vector inputs.

3.2 Cerebellar Model Articulation Controller as a Model of the Human Cerebellum

The CMAC model is used to explore potential for predictive control and an investigation of predictive processes relevant to the cerebellum in the control of movement (see figure 3.2). As the central mechanism to control the movements of the biped robot, everything that scientists need the robot to accomplish must come from accurate mathematical functions of the CMAC controller. Criticism of the CMAC model generally lies in the memory capacity of the model. A compromise must be reached between the quality of learning and the amount of memory available.

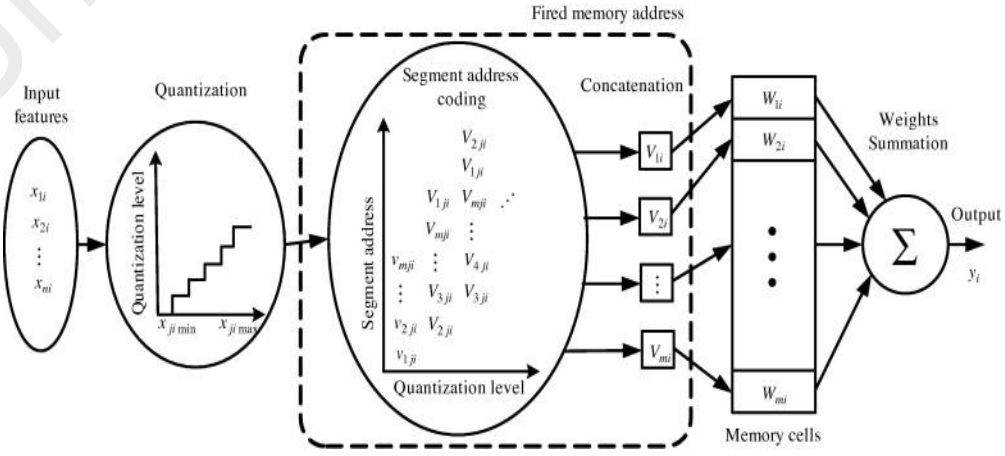


Figure 3.2: CMAC Neural Network (Adapted from Albus J.S, 1989)

Figure 3.2 shows the computational model of the CMAC neural network (Albus, 1979). The CMAC neural network is fundamentally implemented as a multi-dimensional memory array, whereby an input vector acts as the address decoder to access the respective memory cells containing the weight parameters to compute the corresponding output. The cerebellar model articulation controller learns the correct output response to each input vector by modifying the contents of the selected memory locations.

The CMAC neural network has been commended for its compatibility with both hardware and software (Chen & Ou, 2011). The model inputs enter into the quantization stage before coding. Concatenation occurs after which the input translates to memory, where the sum of the weighted connections yields an output.

The learning process implemented in the CMAC neural network is based upon error correction. For every input vector, the difference between the CMAC output and the known desired output is computed and the weight values of the chosen memory cells in the network are modified correspondingly.

The CMAC acts as a static associative memory which models the non linear mapping between the input and the outputs of the human cerebellum. The input transfer to cerebellum through mossy fiber and output comes from purkinje cell. The enormous mesh of granule cell encoders in the cerebellum corresponds to an association layer that produces a scanty and extended representation of the mossy fiber inputs. The synaptic connection between the parallel fibers and the dendrites of the purkinje cells forms an array of adjustable synaptic weights that motivate the grid-like CMAC computing structure. In the human cerebellum, these adjustable synaptic weights are linearly integrated by the purkinje cells to form the cerebellar output.

3.2.1 Important Basics of CMAC

The principle of CMAC is to map a difficult to solve, usually low-dimensional problem into a space of much higher dimension. In this respect, a linear solution is searched, while training a simple one-layer, feed-forward perception. The CMAC can be considered as an associative memory, which performs two subsequent mappings. The first one, that is a non-linear mapping, projects an input space point *X* into a binary association vector *a*. The association vectors always have *N* active elements, which mean that *N* bits of an association vector are ones and the others are zeros. As the value of *N* affects the generalization property of the CMAC, it is often called generalization parameter. There is a one-to-one mapping between the individual input data and the association vectors. Bits are in the association vector parallel to *a* binary basis function with a finite support of *N* quantization intervals. This means that a bit will be active if the input value is within the support of the corresponding basis function which support is often called as the receptive field of the basis function.

The second mapping calculates the output of the network as a scalar product of the association vector *a* and the weight vector *w:*

$$U = a(X)^T W \tag{3.1}$$

The weights of CMAC are updated by using below equation:

$$W_{t_i} = W_{t_{i-1}} + \frac{\beta e}{N} \tag{3.2}$$

W_{t_i} and $W_{t_{i-1}}$ are respectively the weights after and before the training at each sample time, β is learning rate which is included in [0, 1], e is the error between the desired output and the computed output of the CMAC.

Cerebellar Model Articulation Controller might possibly be viewed as a lookup table. Every state variable is quantized as well as the state space is split into discrete states. A vector of quantized input values specifies a discrete state and it is use to

generate addresses for retrieving information coming from the memory due to this state. Figure 3.3 illustrates the partition scheme of the two discrete state variables (Goodwin et al., 2001).

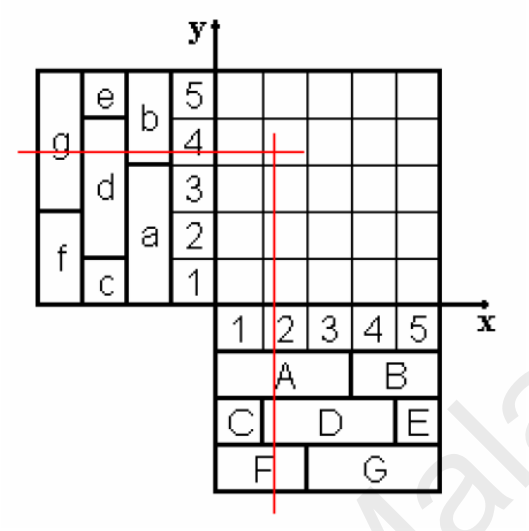


Figure 3.3: Block division of the CMAC in the case of two input variables (Adapted from Goodwin et al., 2001)

Each state variable is quantized into two blocks. In this example, variable V_1 is divided into *A* and *B*, and V_2 is divided with *a* and *b*. Notation A_a, A_b, B_a and B_b regions are known as hypercubes. Hypercubes may be obtained by shifting each block a small interval. In particular, possible shifted regions are *C*, *D* and *E* in the second row for V_1 and *c*, *d* and *e* in the second column for V_2; then we have new hypercubes C_c, C_d, C_e, D_c, D_d, D_e. E_c, E_d and E_e,With this particular type of decomposition, we are able to suppose that there exists N_h layers of hypercubes, and N_e will be the quantity of elements within a complete block. Hypercubes in the p_{th} layer are defined by the p_{th} row and p_{th} column. Each state present by N_h different hypercubes. In the CMAC each hypercube corresponds to a physical memory element. Memory elements distributively store information for a discrete state (Jalali et al., 2012). The data retrieval process of the CMAC for a memory size of N_h and an amount of stored data Y_i can be expressed as:

$$Y_i = a_i w = \sum_{j=1}^{N_h} a_{ij}\, w_j \tag{3.3}$$

where w is the column vector of the memory contents and a_i is a memory element selection row vector that has N_e ones. Figure3.4 shows the block diagram of the CMAC (C. M. Lin, 2004).

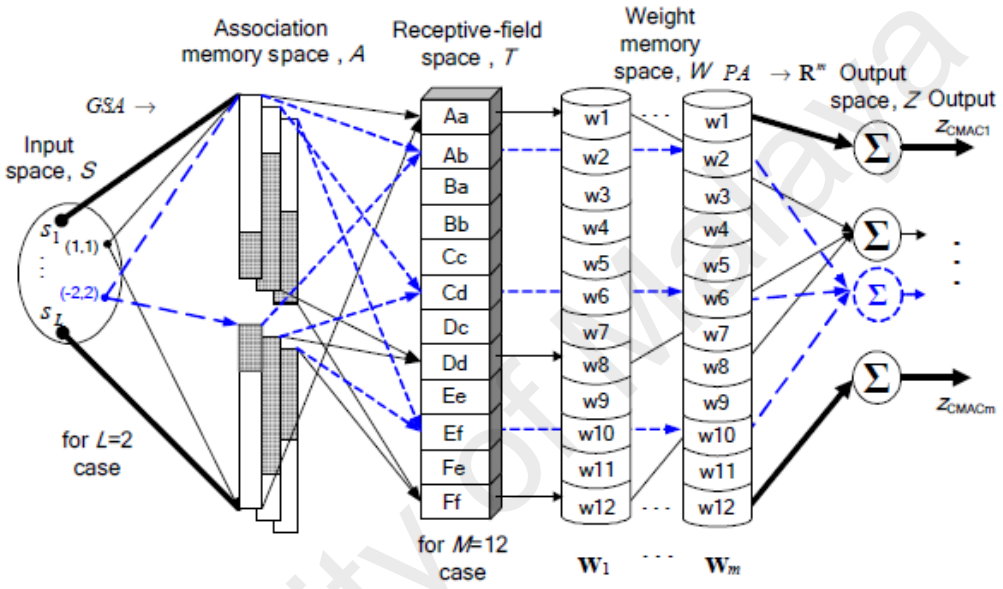


Figure 3.4: Mapping and signal propagation for a basic CMAC (Adapted from C. M. Lin, 2004)

For example, for the (v_1,v_2)=(1,1) case, the hypercubes are A_a, D_d and E_e, and the memory element selection vector is [1 0 0 0 0 0 0 1 1 0 0 0] for the 12 hypercubes *[*A_a A_b B_a B_b C_c C_d D_c D_d E_e E_f F_e F_f *]*. In that algorithm, as mentioned above, the intended response is to be provided to learn the weights of the CMAC. For multi-dimensional input state space cases, the required memory size will grow. In fact it quickly becomes unpractical for CMAC applications. To accomplish this goal, a fuzzy Cerebellar Model Articulation Controller is proposed.

3.3 Q-learning

Adding the learning rule is simple and can modify until the desired output is generated. The hidden layers of the multilayer perceptron were unaddressed. It is here that rules begin to surface to remedy the multilayer problem:

Q-Learning Transition Rule:

Q (state, action) = R (state, action) + gamma * Max [Q (next state, all actions)] (3.4)

There are numerous ways to present commands for Q-Learning and thus motion, in computer software programming. Reinforcement Learning (RL) is achieved in robotics by interaction with an environment versus trial and error approaches (Woergoetter & Porr, 2008). The method is chosen by whether or not the learning will occur in real time with applications such as Semi Online Fuzzy Q–Learning. Currently Reinforcement Learning methods include off-line, semi online, online, and fast online learning.

State signals with the Markov property rely upon the state and action to obtain the response (Sutton & Barto, 1998). Semi Online Q-Learning methods are more efficient programming with important regular updates of the parameters. Estimating distributions algorithm Q-learning (EDAQ) offers support to the space more efficiently than the original Q-Learning function. Optimization by estimating distributions algorithm (EDAs) is just one method to increase the reward for learning. Since its inception, Q-learning has been expanded for use in several disciplines including robotics. Early strategies of Reinforcement Learning were hindered by too many options. Modern scientists have revised the early applications of it for Reinforcement Learning to allow for a versatile platform across computer languages.

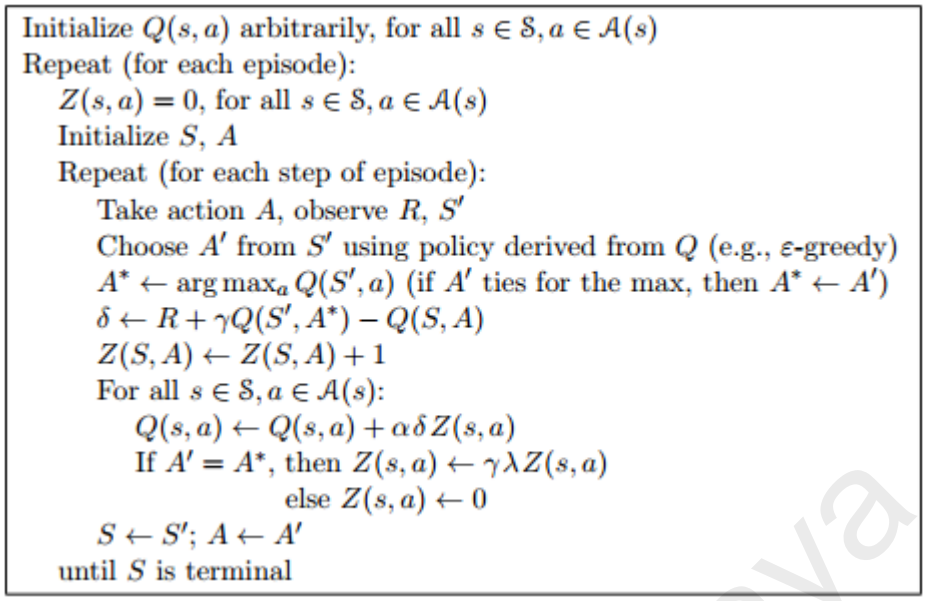


Figure 3.5: Watkins Q (λ) Learning form (Adapted from Sutton & Barto, 1998)

Recent languages have been revised to reduce the number of options in the dynamic reinforcement environment to save space and time. Originally, Q-Learning only acknowledged the first reward and *I* step renditions of it. Most of the revisions have been applied to the state and action spaces. Watkins (1989) decided that a combination of Q(λ) and TD(λ) would increase its efficiency as shown in figure 3.5 (Sutton & Barto, 1998). 'Lazy learning' occurs when the Q-values are not considered at all when they are not required to perform the function.

The value of Q (*s, a*) is estimated by measurement of temporal differences, even without a defined model environment (Poole & Mackworth, 2010). The max variable represents the highest attainable reward in the next state. A learning rate between 0 and 1 indicates that the Q-values are not updated (Dayan, 1992). Higher values mean that learning can occur in brief periods of time. The discount factor determines the importance of future rewards. A factor of 0 will make the agent (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a

long-term high reward. If the discount factor meets or exceeds 1, the action values may diverge.

$$P_{ss'}^{a} = P_r\{S_{t+1} = S' | S_t = S, a_t = a\} \quad (3.5)$$

The probability *P* of the next step is represented as *s'*. Next, the reward *R* is calculated according to the same variables. Probability is replaced by Expectation *E*.

$$R_{ss'}^{a} = E\{r_{t+1} | S_t = S, a_t = a\ , S_{t+1} = S'\} \quad (3.6)$$

Q-learning optimization can be accomplished by estimating distributions algorithm (EDAs) (Hu, 2006). The working spaces for the biped robot are multidimensional; thus Q-learning is used to construct the statistical models.

Formulating the biped gait synthesis as a constrained multi-objective optimization problem provides a dynamically stable and low cost biped gait generated by EDAs with Q-learning (EDAQ). EDAQs estimate probability distributions derived from objective functions optimized to generate searching points in the highly coupled and dimensional working space of biped robots (Hu & Sun, 2006).

Watkins' Q-Learning (1989, 1992) is broadened by the fuzzy inference system instead of a generalized algorithm. Carrersas, Ridao, & El-Fakdi (2003) solved the generalization problem by a Semi Online Neural Q-learning algorithm using Q-learning as the learning technique using previous and current samples in real-time.

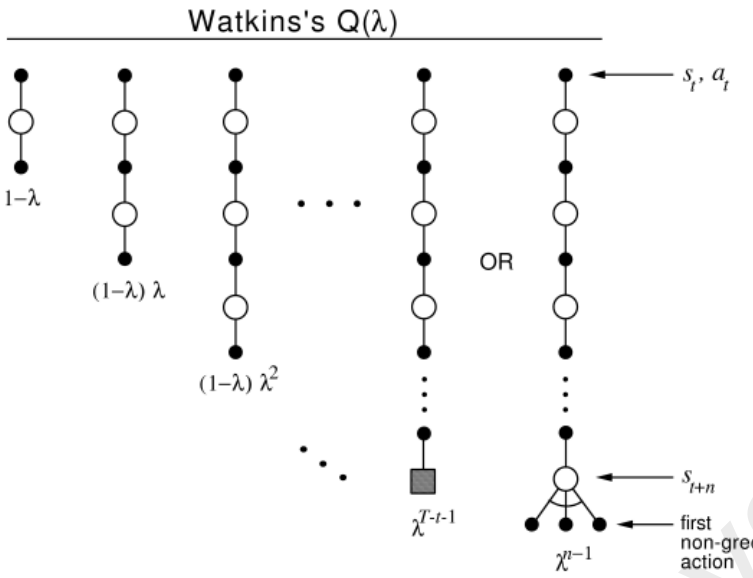


Figure 3.6: The backup diagram for Watkins' Q (λ) (Adapted from Sutton & Barto, 1998)

There are several versions of Q-learning which can use fuzzy logic and implement credit assignments (Gu & Hu, 2004). The Q-Learning Reinforcement Learning algorithm was formulated by Watkins in 1989. The algorithm is based upon one-step dynamic Markovian decision making processes (MDP). The process is comprised of state and action sets which can be seen in the formula as variables ***S*** and ***a*** for state space and action.

The eligibility traces of Reinforcement Learning record occurrences of events from a forward or a backward view. Watkins' model has a zero value for its eligibility trace after each non-greedy action and initializes arbitrarily. Typically, a greedy policy can be used to choose an action a_t. Greedy algorithms provide estimation of the rate of approximation. The first greedy action can be seen in Watkins' model which is shown in figure 3.6 (Sutton & Barto, 1998).

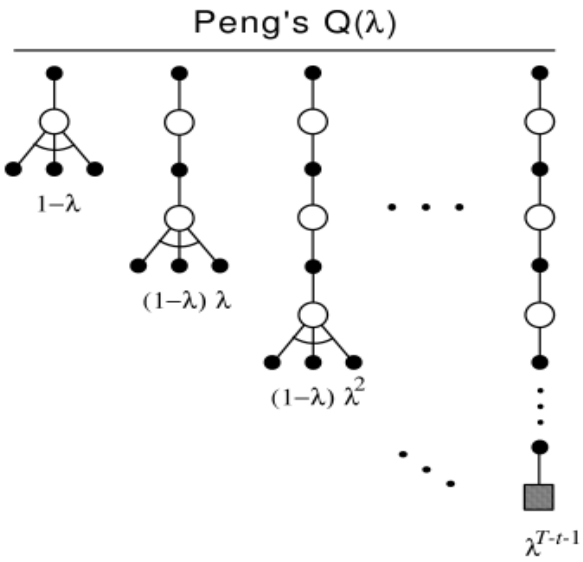


Figure 3.7: The backup diagram for Peng's Q (λ) (Adapted from Sutton & Barto, 1998)

Several variations of Watkins' model are made in Peng's functions for Q-Learning. One important variation is that Peng's model displays no differences in value for greedy and non-greedy actions as shown in figure 3.7 (Sutton & Barto, 1998). Historically, Peng's Q Model is less desirable because it is harder to work through and does not always produce optimization. However, a combination of Peng's and Watkins' models (PW) produces a powerful model which is now used in combination to solve reinforcement problems using either property for the state space (Hu & Sun, 2006).

The last approach is, single agent Naïve Q-Learning method which consists of Watkins' Q-Learning model without the zero eligibility. The Bayesian logic incorporates the Naïve-Bayes algorithm to solve learning problems (Watkins, 1989).

As in the real world, future rewards are perceived less highly in value than immediate rewards. Q-Learning algorithms produce optimal control schemes with delayed reward. The algorithm provides movement, actions in a stated dynamic environment that the biped robot can perform. A challenge in RL is the trade-off between exploration and exploitation. To obtain a large portion of a reward, a RL agent must prefer actions that it has tried in the past and found effective. To discover the

actions, it must try new actions. The agent has to *exploit* what it knows in order to obtain reward, but it also has to *explore* in order to make better action selections in the future. The proposed Q-Learning algorithm will introduce and discus on chapter 5, this model called as semi online fuzzy Q-learning and used to footstep planning approaches step over an obstacle.

3.4 Summary

CMAC neural network is used in many real-time control systems, pattern recognition and signal processing problems. Besides its charming features, the main difficulty of the CMAC network in realistic on-line applications is related to the required memory size. The literature reviews show that an optimal structure carrying out a minimal modelling error can be achieved. Therefore, the choice of an optimal structure allows decreasing the memory size as well reduces the computing time.

The investigations on Q-learning algorithm show a real interest of this method because: (1) computing time is very short. (2) This approach is valid for both dynamic and static obstacles. (3) The Q-learning method for footstep planning is operational for both predictable and unpredictable dynamical environment permitting the control to intensify the robustness.

CHAPTER 4

Design of Hierarchically Clustered Fuzzy CMAC and Adaptive Control of Robotic Arm

4.0 Design of Hierarchically Clustered Fuzzy CMAC and Adaptive Control of Robotic Arm

4.1 Introduction

This model of fuzzy cerebellar is able to perform dynamic state estimation and prediction. The system is able to predict next states by using information from motor commands. Cerebellar output can then be used for several purposes: to adaptively filter future input, to improve detection of novel or unexpected events, to modulate motor outputs, or to provide feedback for motor learning. Our goal is to build a Hierarchically Clustered Fuzzy model of cerebellum that inspired by its circuitry and function. To develop this model first a fuzzy cerebellar model articulation controller is developed.

The cerebellum model incorporates two types of learning which are potentially supported by the cerebellar cortex. Purkinje cells (PC) have long been known to receive inputs from two sources: Parallel fibers (PF) and Climbing fibers.

Activation Parallel fibers (PF) followed by Climbing fibers (CF) input have been shown to cause long-term depression (LTD) at the PF-PC synapse, when climbing and parallel fibers are conjunctively activated LTD is quickly evoked and actually been implicated especially in kinds of motor learning. Granule cell axons make multiple synapses onto their overlying PCs as they ascend through the PC layer to the molecular layer. This Ascending Branch (AB) input induces postsynaptic activity that causes facilitation when coupled with local PF input (Assad, 2001); AB-PF correlations lead to *supervised* learning including long-term potentiating (LTP) at the PC-PF synapse. This allows the learning in cerebellar cortex to be much more flexible through a combination of long-term potentiating and long-term depression. The proposed model combines PF LTP for feedforward state prediction with CF LTD for feedback error correction. Given

a mechanical system with particular dynamics and motor command inputs, three functions are useful to learn: (1) through state space for predicting trajectories, (2) to learn the "good" region of state space in which an action decision (3) to simulate motor commands to change trajectory to increase or optimize its intersection with the "good" regions of the state space. The cerebellum has been implicated in all three functions (Zhou et al., 2001).

The objective of the Hierarchically Clustered Fuzzy CMAC (HCFCMAC) neural network is to develop a memory allocation procedure to improve the storage efficiency, in addition to relieve the generalization-accuracy of the FCMAC network. The proposed network is inspired by the neurophysiology of the human brain. In addition, the learning process of the proposed HCFCMAC network always converges when the learning rate is within a theoretical range.

According to Teddy, Lai, & Quek (2007): "The two major drawbacks associated with the uniform quantization scheme of the CMAC network are (1) a constant output resolution associated with the entire input space, and (2) the generalization–accuracy dilemma" . The accuracy of learning is reduced in juxtaposition to memory input and output capacities. The primary obstacle 'in utilizing the CMAC is finding a suitable way to break up the state space into tiles. If the application is performed incorrectly, states with different preferences can be forced to learn together when confined to a single tile. The result is lagging or deterrence to learning a successful policy (Lin & Wright, 2010).

The largest limitation of the Cerebellum Model Articulation Controller is related to memory size in applications for nonlinear systems. In some instances, all of the CMAC memory cells are not used in the process. The memory used by CMAC depends on input space dimension and input signal quantification step (Sabourin, Yu, & Madani et al., 2012).

4.2 Design of Fuzzy cerebellum model articulation controller

A fuzzy Cerebellar Model Articulation Controller neural network combined with a two dimensional CMAC neural network is considered. The structure of a single output FCMAC was proposed in figure 4.1 (Wu, 2006).

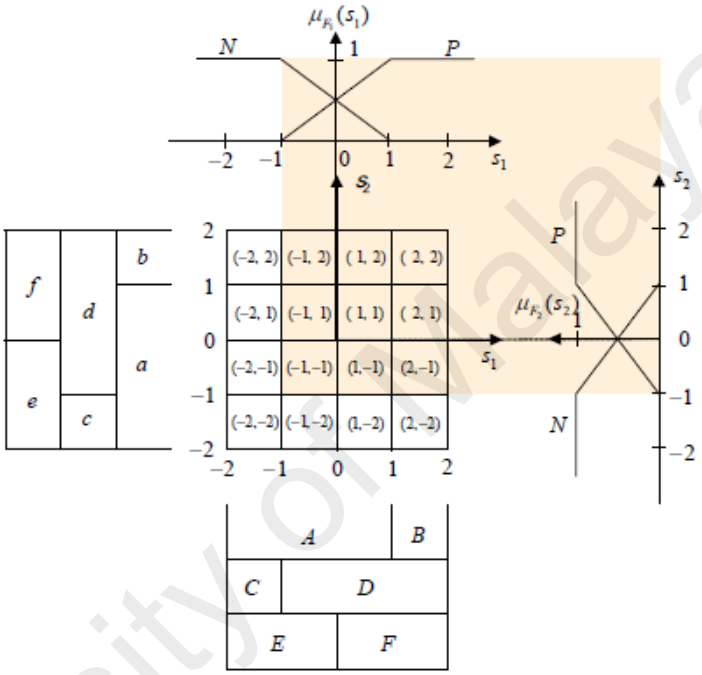


Figure 4.1: Sketch of CMAC integrated with fuzzy sets (Adapted from Wu, 2006)

The fuzzy rules, as usual, are composed of the following structure.

$$\text{IF } \sigma = B_1^{i_1} \text{ and } \dot{\sigma} = B_2^{i_2}, \text{ THEN } y = a_{i_1 i_2} w$$

where i_j= 1,2,…,p_j, j=1 and 2 and $a_{i_1 i_2}$is the memory element selection row vector due to the conventional CMAC, and w is the weighting column vector of the CMAC. $\dot{\sigma}$ is the rate variable of σ. The linguistic variable σ is defined by:

$$\sigma = \sum_{i=1}^{n} x_i \qquad (4.1)$$

Where x_i is the state variable and i=1, 2 … n, It therefore have the rate of σ; which is designated as $\dot{\sigma}$; Using this transformation, easily can deal with the high dimensional CMAC neural network controller. Furthermore, for notation and analytical simplicity, the input variable of the fuzzy CMAC neural network as s= $[\sigma_1, \sigma_2]$ where $\sigma_1 = \sigma$ and $\sigma_2 = \dot{\sigma}$ .Now the following fuzzy CMAC IF-part inferred function from the fuzzy rules defines as:

$$\varphi_{i_1 i_2}(\boldsymbol{s}) = \prod_{j=1}^{2} \mu_{B_j^{i_j}}(\sigma_j) \Big/ \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} \left(\prod_{j=1}^{2} \mu_{B_j^{i_j}}(\sigma_j) \right) \tag{4.2}$$

Where the $\mu_{B_j^{i_j}}$ () is the membership function of fuzzy set $B_j^{i_j}$. Therefore, the output of the fuzzy CMAC obtained as:

$$y = \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} \boldsymbol{a}_{i_1 i_2} \boldsymbol{w} \left(\prod_{j=1}^{2} \mu_{B_j^{i_i}}(\sigma_j) \right) \Big/ \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} \left(\prod_{j=1}^{2} \mu_{B_j^{i_i}}(\sigma_j) \right) \tag{4.3}$$

$$= \varphi(\mathbf{s}) \boldsymbol{A} \boldsymbol{w}$$

Where A is a matrix constituted from the memory element selection vector $\boldsymbol{a}_{i_1 i_2}$, and $\varphi(\mathbf{s})$ is a row vector whose proper dimension depends on the number of fuzzy rules. The important advantages of the proposed FCMAC are: (I) the vector $\boldsymbol{a}_{i_1 i_2}$ easily determine even for multi-dimensional structures for the CMAC; and (II) the computational load can be considerably reduced in real-time operation. In the case of association of fuzzy layers, assume: n is the number of input variables. Each node at

Fuzzified Layer corresponds to variables which are expressed by membership functions $\mu_{B_j^{i_j}}$.

There are *m* quantization's (membership functions) for each input. The number of the nodes in this layer is n^m. Fuzzified layer performs the fuzzification of input variables and it matches both sensor layer of CMAC and fuzzifier of fuzzy logic controller. Fuzzy association layer connects fuzzified layer and accomplishes the matching of precondition of fuzzy logic rule.

There is not difference between any types of Fuzzy neural network that is chosen, diverse result chooser consequent in different model of fuzzy networks. According to (J. Jang, 1993 and C. Lin, 2006) TSK-type fuzzy model is superior in accuracy of learning and the network size compare to another type of fuzzy network. Here we used TSK-type fuzzy model indicate the weight vector of CMAC.

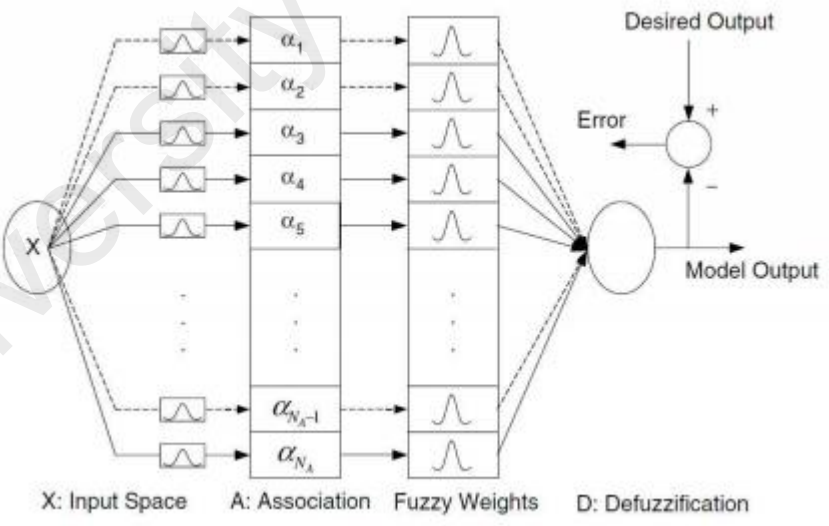


Figure 4.2: The architecture of Fuzzy CMAC (Adapted C-J Lin, 2009)

The FCMAC like basic CMAC that also approaches a nonlinear function $y = f(x)$ which has two mapping $S(x)$ and $P(\propto)$. $S(x)$ point every *x* in the input vector onto an merge space $\propto = S(x) \in A$ that is N_l contains nonzero elements. The Gaussian

function utilized as linear parametric equation and the receptive field functions of the input variance as the TSK-form output.

A one-dimension Gaussian function equation is as follow:

$$\mu(x) = e^{-(x-m/\sigma)^2} \tag{4.4}$$

Wherein x is particular input vector, σ indicate the variance and m present the corresponding centre.

If N_D dimensional problem formulate, the equation for GF will be as follows:

$$\alpha_j = \prod_{i=1}^{N_D} e^{-(x_i-m_{ij}/\sigma_{ij})^2} \tag{4.5}$$

Where Π is the product operation, α_j present the j^{th} element of the association space and N_D is number of the acceptable field function for every input stage.

The $P(\propto)$ calculate output y by planning association space onto a space of tuneable receptive field function. Here, utilize by centroid of area the fuzzy output can defuzzified onto a numeric output y. So, the genuine output y can formulate as below:

$$y = \sum_{j=1}^{N_L} \alpha_j (a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i) \Big/ \sum_{j=1}^{N_L} \alpha_j \tag{4.6}$$

The j^{th} element of output cell is introduced as below:

$$a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i \tag{4.7}$$

Whereby, a_{0j} and a_{ij} explain by numeric value, N_D represent the number of input dimension, N_L is the number of hypercube cells. According to above equation hierarchically cluster learning algorithm is proposed to determine the structure and parameters.

All input variables x_i $(i = 1 \dots n)$ have m quantization. t is the time to present association from an input vector $X = [x_1 \dots x_n]$ to an output $y(t)$. Thus the memory space is $t \times m^n$. The number of memory has to increase exponentially with the number of input variables. An important problem facing FCMAC applications is how to supply with this memory detonation problem.

4.3 The Hierarchically Clustered Fuzzy Cerebellum Model Articulation Controller Algorithm

4.3.1 HCFCMAC Architecture

The primary function of the uniform quantization outline in the CMAC neural network results is a stable output resolution all over the whole input space. Due, to the meaning of dynamic and characteristics of the learning data, specific part of input has more significant data compare to the rests. Thus, inspire by brain neurophysiological studies, novel fuzzy CMAC structure named the Hierarchically Clustered Fuzzy CMAC (HCFCMAC) suggested as an alternative to realize non uniform quantization. HCFCMAC effectively designates the number of valid memory cells on the basis of the information contents of the input/output mapping of the data. In the HCFCMAC network, memory productivity is obtained by allocating more memory cells to the input regions where quick instability of the output values are observed, and less memory cells to regions with comparatively unchanged output values.

Therefore, a finer quantization level is achieved for the regions of the input space that contains more information. This procedure is shown in figures 4.3. Figure 4.3(a) describe the analogous uniformly quantized CMAC memory structure, and figure 4.3(b) shows the HCFCMAC for the similar target output surface.

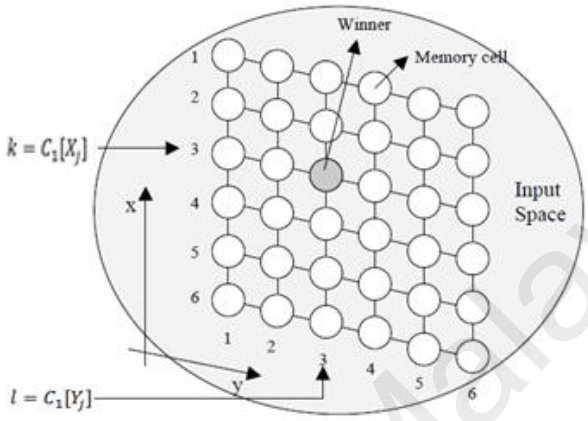


Figure 4.3(a): Two dimension CMAC memory structure

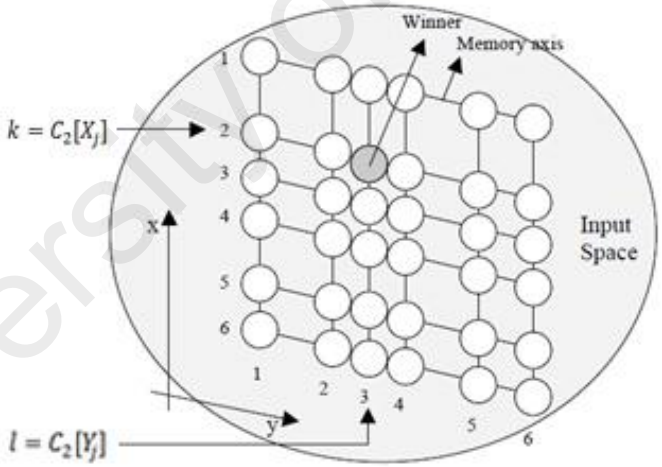


Figure 4.3(b): Two dimension HCFCMAC memory structure

Figure 4.3: Comparison of CMAC and HCFCMAC memory structure for a specific output surface

Because the size of the CMAC network is an exponential function of the size of the inputs, larger command functions can become extremely heavy. To address these concerns, methods have been developed to quantize. Quantization will occur by either increasing resolution or using quantization step sizes. In such uniform quantization, the

solution to the memory problem creates input generalization problems. The CMAC model with quantization of the same size undervalues the variance of the target function (Yeh, 2002). The association cell of CMAC activated by the input vector addresses memories to determine the output. To increase the control accuracy, the selected quantification step must be as small as possible. The characteristics of the training data determine the small percentage of the entire set of CMAC memory cells that are used. A small-sized CMAC is able to better generalize the characteristics of the training data but at a lower output resolution, while a large-sized CMAC produces more accurate outputs at the expense of data generalization (Teddy et al., 2007).

In non-uniform quantization methods, the quantization error is configured along with the input. Quantization methods for cerebellar models are designed with memory issues. The Hierarchically Clustered Fuzzy CMAC (HCFCMAC) is a model of CMAC architecture that is developed to add memory to the input space. The strength of the model lies in its learning convergence.

In HCFCMAC, the available memory cells are allocated on the basis of the characteristic of the training data that is determined by the data repartition in the input space and also the difference of the output value.

In HCFCMAC, there are two levels in the operation of the network:

(1) Network initialization phase, and

(2) Network learning phase.

Determine the quantization function at each input dimensions is the target of the first phase, while the second stage is to learn the memory content of the hierarchically clustered fuzzy CMAC network.

The hierarchical clustering technique is occupied at every dimension to recognize the optimal quantization decision function in each of the input dimensions. A quantization cluster is explained as the range of a memory quantization level in a particular input

dimension. Beginning with the initial set of clusters in a specific input dimension, two clusters with the smallest combining cost are merged in each repetition of the hierarchical clustering process until the number of quantization clusters is equal to the number of memory space in the input dimension (see figure 4.4).

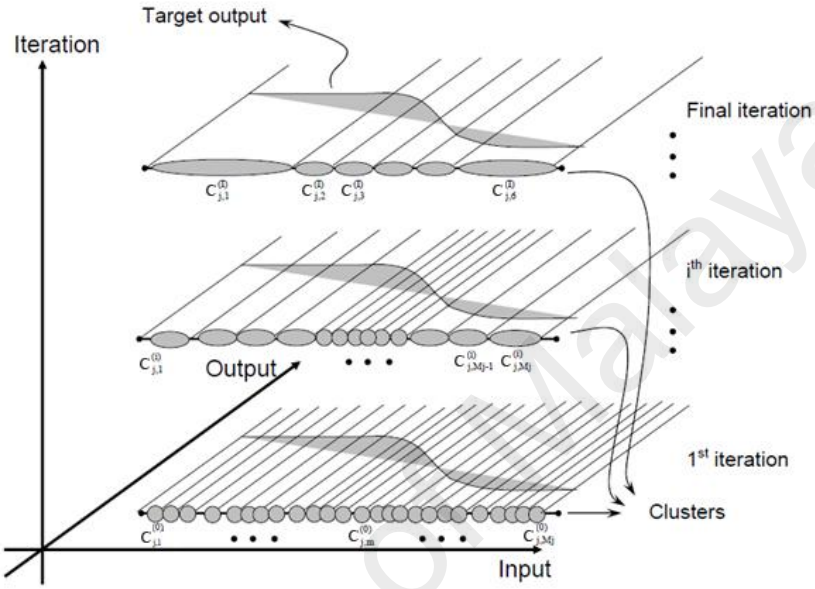


Figure 4.4: illustration of HCFCMAC quantization process for input dimension *j* where $\widehat{M}$=6

Where j is the total number of input dimension. Presume, the training dataset that is used for training the HCFCMAC is as follow:

$$L = \{(X_1, \hat{Y}_1), (X_2, \hat{Y}_2), \dots, (X_S, \hat{Y}_S)\} \tag{4.8}$$

The s^{th} input vector explained by $X_s = [x_{s,1}, x_{s,2}, \dots, x_{s,J}]^P$, and $\hat{Y}_s$ define the expected output of HCFCMAC. The total number of available memory space in each dimensions denote by $\widehat{M}$ and $C_j^i = \{C_{j,1}^i, C_{j,2}^i, \dots, C_{j,m}^i, \dots, C_{j,M_J}^i\}$ shows a group of M_j clusters in the j^{th} input dimension on the i^{th} iteration of hierarchical clustering proceeding.

The HCFCMAC memory allocation procedure is considered as follows:

Phase 1: data preparation

For every input dimension $j \in \{1 \dots J\}$, the input training samples $x_{s,j}$'s are $\{x_{1,j}, x_{2,j}, \dots, x_{s,J}\}$ where $x_{s,j} \leq x_{s+1,j}$

Phase 2: Characterize the initial set of quantization clusters

Here $C_{j,m}^{i}$define the m^{th}cluster in the j^{th} input dimension on the i^{th} iteration of hierarchical clustering process. $d_{c_{j,m}^{i}}$ is the number of data points.

The quantization cluster $C_{j,m}^{i}$ is describe as the set of data pairs,

$$C_{j,m}^{i} = \{(x_{j,m-1}^{'i}, y_{j,m-1}^{i}),\ (x_{j,m-2}^{'i}, y_{j,m-2}^{i}), \dots, \left(x_{j,m-d_{c_{j,m}}}^{'i}, y_{j,m-d_{c_{j,m}}}^{i}\right)\}$$

The initial set of clusters C_{j}^{0} is taken from the input training data points. *One* cluster in the j^{th} dimension organizes by each *different input* $x_{s,j}$ from the training set. Training data are combined together based on the same input value as follow:

$$C_{j,m}^{0} = \left\{(x_{j,m-1}^{'0}, y_{j,m-1}^{'0}), (x_{j,m-2}^{'0}, y_{j,m-2}^{'0}), \dots, \left(x_{j,m-d_{c_{j,m}}}^{'0}, y_{j,m-d_{c_{j,m}}}^{'0}\right)\right\} \text{ where}$$

$$x_{j,m-1}^{'0} = x_{j,m-2}^{'0} = \dots = x_{j,m-d_{c_{j,m}}}^{'0} = x_{j,m}^{'0} \tag{4.9}$$

The quantisation cluster $C_{j,m}^{i}$ is specified by characteristic value $V_{j,m}^{i}$ and its centroid $R_{j,m}^{(i)}$, $V_{j,m}^{i}$ of a quantisation cluster $C_{j,m}^{i}$ is consider as mean of the output value $y_{j,m-k}^{'i}$ and is defined by below equation:

$$V_{j,m}^{i} = \frac{\sum_k y_{j,m-k}^{'i}}{d_{C_{j,m}}} \qquad k \in \{1 \dots d_{C_{j,m}}\} \tag{4.10}$$

The centroid of the quantisation cluster is described as follow:

$$R_{j,m}^{i} = \frac{\max(x_{j,m-k}^{'i}) - \min(x_{j,m-k}^{'i})}{2} \qquad k \in \{1 \dots d_{C_{j,m}}\} \tag{4.11}$$

Accordingly the initial set of clusters is described as follow:

$$C_{j}^{0} = \left\{C_{j,1}^{0}, C_{j,2}^{0}, \dots, C_{j,M_J}^{0}\right\} \text{where } V_{j,m}^{0} = \frac{\sum_k y_{j,m-k}^{'0}}{d_{C_{j,m}}} \quad k \in \left\{1 \dots d_{C_{j,m}}\right\} \text{and } R_{j,m}^{0} = x_{j,m}^{'0}$$

Since, $x'^{0}_{j,m-1} = x'^{0}_{j,m-2} = \cdots = x'^{0}_{j,m-d_{c_{j,m}}} = x'^{0}_{j,m}$

Phase 3: merge similar clusters

The clusters in the initial set of quantization clusters (C_j^0) is repetition combined till the number of quantization clusters in the same dimension M_j become equal to the number of available memory cells which defined in advanced. The *cost function* is basis of cluster merging. The weighted compound of distance between the characteristic value and the centroid of two neighbour clusters are the main variable of merging cost function and is defined as follow equation:

$$\mathcal{F}\left(C^i_{j,m_1}, C^i_{j,m_2}\right) = \varphi_1\left(\left\|V^i_{j,m_1} - V^i_{j,m_2}\right\|\right) + \varphi_2\left(\left\|R^i_{j,m_1} - R^i_{j,m_2}\right\|\right) \quad (4.12)$$

$$\varphi_1 + \varphi_2 = 1 \quad (4.13)$$

Where, $m_1,\ m_2 \in \{1 \dots M_j\}$ and $m_2 = m_1 + 1$ at the j^{th} iteration. φ_1 and φ_2 are user defined. The parameters φ_1 and φ_2 weight the significance of the measured differences in the characteristic value (output) and the quantization point (input) size as the full of merging two neighbour clusters. A system with slow changing input and fast changing output should need a larger φ_1 than φ_2 and vice versa.

In every repetition, the two neighbour clusters with the minimum merging cost are merged as below equation:

$$C^{i+1}_{j,\acute{m}} = C^i_{j,\tilde{m}_1} \cup C^i_{j,\tilde{m}_2} \quad (4.14)$$

$$\mathcal{F}\left(C^i_{j,\tilde{m}_1}, C^i_{j,\tilde{m}_2}\right) = \min \mathcal{F}\left(C^i_{j,m_1}, C^i_{j,m_2}\right) \text{ where}$$

$$m_1, m_2 \in \{1 \dots M_j\} \text{ and } m_2 = m_1 + 1$$

So, $V^{i+1}_{j,\acute{m}}$ and $R^{i+1}_{j,\acute{m}}$ of $C^{i+1}_{j,\acute{m}}$ are computed using by equation (4.10), (4.11). Actually this procedure adopted from brain neuron connections, non active neurons are omitted and their functions add by the active (winning) neurons. In HCFCMAC, to reduce data abundance, similar clusters are combined (merged) and indicate by

expanded clusters. HCFCMAC assigned more memory cube (memory cell) to the regions with higher degrees of output variation.

Phase 4: Create the quantization decision function.

In the end of hierarchical clustering a group of $\widehat{M}$ quantization clusters is obtained process for each j^{th} input dimension. The last cluster iteration for input j is denoted by I. Therefore, the final set of quantization cluster is described as:

$C_j^I = \{C_{j,1}^I, C_{j,2}^I, \dots, C_{j,\widehat{M}}^i\}$. The quantization decision function in j^{th} input dimension $C_j[.]$ is specific from C_j^I, as defined by follow equation:

$$C_j[.] \rightarrow \{R_{j,1}^I, R_{j,2}^I, \dots, R_{j,\widehat{M}}^i\} \tag{4.15}$$

Whereby, $C_j[.]$ defined the quantization mapping function, $R_{j,m}^I$ is the centroid of the m^{th} $q$uantization cluster of the j^{th} input dimendion after merging process. $\widehat{M}$ is the number of predefined memory cells in each dimension. The quantization decision points derived for each input dimension following form the memory axes of the HCFCMAC.

4.3.2 The HCFCMAC learning process

The HCFCMAC obtain a proper output to an input by modifying the contents of the chose memory spaces. Here I denoted the maximum number of training iterations. The HCFCMAC learning process is divided as below:

1) Specific the winner neuron for input X at the i^{th} iteration. where i $\in \{1 \dots I\}$. For every input, index $\bar{X}$ of the selected neuron is calculated through the quantization mapping function $C[.]$. For input $X = [x_1, x_2, \dots, x_j, \dots, x_J]^P$ the winner neuron $\bar{X}$ is computed as follow:

$$\bar{X} = C[X] = [C_1[x_1], C_2[x_2], \dots, C_j[x_j], \dots, C_J[x_J]]^P \tag{4.16}$$

Whereby, $\bar{X}$ specific the quantized input X and J is the number of dimension.

2) Compute the output

In the learning sequence there are four parameters $m_{ij}, \sigma_{ij}, w_j^m \; and \; w_j^\sigma$ which requires tune. For learning algorithm of HCFCMAC the supervised method used to modify the parameters. This parameter gives suitable value to present the memory content.

Minimize the cost function E is main goal in single output model, which is described as below:

$$E = \frac{1}{2}\,(y^d(t) - y(t))^2 \tag{4.17}$$

The desire output indicated by $y^d(t)$ also real output described by $y(t)$ at t time. The following equations are used for updating the fuzzy weight.

$$w_j^m(t+1) = w_j^m(t) + \Delta w_j^m \tag{4.18}$$

$$w_j^\sigma(t+1) = w_j^\sigma(t) + \Delta w_j^\sigma \tag{4.19}$$

w_j^m denoted the mean of weights and the variance of the weights specified by w_j^σ. where j defined the j^{th} weight cell for $j = 1,2, \dots, N_L$

$$\Delta w_j^m = L_r . E_r . \frac{\partial y}{\partial w_j^m} = L_r . E_r . \frac{\alpha_j w_j^\sigma}{\sum_{j=1}^{N_L} \alpha_j w_j^\sigma} \tag{4.20}$$

$$\Delta w_j^\sigma = L_r . E_r . \frac{\partial y}{\partial w_j^\sigma} = L_r . E_r . \frac{\alpha_j w_j^m \sum_{j=1}^{N_L} \alpha_j w_j^\sigma - \alpha_j \sum_{j=1}^{N_L} \alpha_j w_j^\sigma w_j^m}{(\sum_{j=1}^{N_L} \alpha_j w_j^\sigma)} \tag{4.21}$$

Where L_r denotes the learning rate, $L_r \rightarrow [0,1]$ and the learning error E_r is described as the error between the expected output and actual output. The learning rate is given in equation below:

$$E_r = y^d(t) - y(t) \tag{4.22}$$

Therefore, the receptive field functions are updated as the below equation:

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij} \tag{4.23}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta\sigma_{ij} \tag{4.24}$$

Where, j denoted the j^{th} weight cell for $j = 1,2,\dots,N_L$ and i indicated the i^{th} input dimension for $i = 1,2,\dots,n$. σ_{ij} indicated the variance and m_{ij} denoted the mean of the receptive field functions. So the parameters are updated as follow:

$$\Delta m_{ij} = L_r . E_r . \frac{\partial y}{\partial \alpha_j} . \frac{\partial \alpha_j}{\partial m_{ij}} \tag{4.25}$$

$$= L_r . E_r . \frac{w_j^\sigma w_j^m \sum_{j=1}^{N_L} \alpha_j w_j^\sigma - w_j^\sigma \sum_{j=1}^{N_L} \alpha_j w_j^\sigma w_j^m}{(\sum_{j=1}^{N_L} \alpha_j w_j^\sigma)} . \alpha_j . \frac{2(x_i - m_{ij})}{\sigma_{ij}^2}$$

$$\Delta\sigma_{ij} = L_r . E_r . \frac{\partial y}{\partial \alpha_j} . \frac{\partial \alpha_j}{\partial \sigma_{ij}}$$

$$= L_r . E_r . \frac{w_j^\sigma w_j^m \sum_{j=1}^{N_L} \alpha_j w_j^\sigma - w_j^\sigma \sum_{j=1}^{N_L} \alpha_j w_j^\sigma w_j^m}{(\sum_{j=1}^{N_L} \alpha_j w_j^\sigma)^2} . \alpha_j . \frac{2(x_i - m_{ij})^2}{\sigma_{ij}^3}$$

The output of the network to the input X also describe as follow:

$$Y = Z_{\bar{X}} \tag{4.26}$$

Where, Y denoted the HCFCMAC output for input X and Z denoted the hypercube memory array.

3) Update the HCFCMAC memory

The equation for update memory is as follow:

$$Z_{\bar{X}}^{i+1} = Z_{\bar{X}}^{i} + \Delta Z_{\bar{X}}^{i} \tag{4.27}$$

$$\Delta Z_{\bar{X}}^{i} = \tau E_r^i \tag{4.28}$$

Where τ denoted the learning constant.

A neighbourhood based activation of the cells is occupied to smoothen the output during the testing phase of the HCFCMAC. The HCFCMAC output during the testing stage with given input $X_s = [x_{s,1}, x_{s,2}, \dots, x_{s,J}]^P$ is derived as following steps:

Phase 1) Specify the area of activation

The HCFCMAC output is described as the mean of the memory values of the activated cube in the vicinity of X_s. For network input X_s the neighbourhood is specify by a neighbourhood constant N, that defines the comparative size of the neighbourhood. The activation neighbourhood, for input X_s is defined as follows:

$$LB_{s,j} = x_{s,j} - 0.5\,.\,N.D_{s,j} \tag{4.29}$$

$$RB_{s,j} = x_{s,j} + 0.5\,.\,N.D_{s,j} \tag{4.30}$$

Where $j \in \{1, 2, \dots, J\}$ denotes the j^{th} input dimension; $D_{s,j}$ defines the input domain for j^{th} dimension; N is the neighbourhood constant; $LB_{s,j}$ define the left boundary of the neighbourhood and $RB_{s,j}$ is the right boundary. As a result, the memory cells within the neighbourhood for the input X_s make the group of activated computing cells. The accuracy of HCFCMAC output is affected by the size of neighbourhood means more accurate output computation is derived by a smaller neighbourhood size. So, for a compact dataset a smaller neighbourhood size is suitable and larger neighbourhood size appropriate for a dataset which is sparse in the input space as this increment the generalization capability of the HCFCMAC network.

Phase2) The HCFCMAC output

The HCFCMAC output is defined as follow:

$$Y_s = \frac{\sum_{a \in A_S} M_a}{n_{A_S}} \tag{4.31}$$

Where Y_s is the HCFCMAC output with the input stimulus X_s; A_s defines the set of indexes of the activate neighbourhood cells corresponding to input X_s; n_{A_S} is the cardinality of A_s and M_a define the memory value of the activated cell with index a.

4.4 Effect of Input Dimension on Required Memory Size

In a CMAC network, there are essentially two factors ruling the function approximation quality. The first one, called "quantization step" (Δq), allows to map a continuous signal into a separate signal. Actually, Δq corresponds to the relative displacement between cells of the two sequential layers as well. The second parameter, called "generalization parameter" N_l corresponds to the number of layers. These two factors allow to define the size of each cell S_c , the number of cells on the first layer N_c^0 and the total number of used memory size (number of cells) N_c .

Then, we can calculate the size of each cell S_c, the number of cells on the first layer N_c^0 and the total number of used memory size (number of cells) N_c according to the following expressions:

$$S_c = \Delta q N_l \tag{4.32}$$

$$N_c^0 = S_{min} - S_{max}/S_c \tag{4.33}$$

$$N_c = (N_c^0)^2 + (N_c^0 + 1)^2(N_l - 1) \tag{4.34}$$

Now, the effect of input dimension of CMAC on required memory size is considered. Sin function and FSIN function which are expressed in equations (4.35) and (4.36) are used as examples to compare the required memory size of one dimension and two dimensions CMAC during its training.

Sin function: [0,1]→[-1, 1]

$$y = Sin\, x \tag{4.35}$$

FSIN function: $[0,1]^2 \rightarrow [0,1]$

$$FSIN\ (x_1, x_2) = sin^2(2\pi x_1 + 1) sin^2(2\pi x_2 - 1) \tag{4.36}$$

The number of layers defined as N_l and quantization step Δq choose randomly for each of the functions. A training set including 100 random values selected in the corresponding dimensional space, has been constructed. When the CMAC totally trained, the modelling error called mean squared error is carried out that is computed from the equation (4.37).

$$E_{square} = \sqrt{\frac{\sum_{i=1}^{100}(y_i^d - y_i)^2}{\sum_{i=1}^{100}(y_i^d)^2}} \tag{4.37}$$

Where y_i^d represent the desired output and y_i is the actually learned output value.

Figure 4.5 and figure 4.6 show that the CMAC network can approximate both of the two functions well. For one-input sin function the used memory size is only 434 when the mean squared error reached 0.58% (listed in Table 4.1), while two-input FSIN function requires 10 times greater memory size than it for sin function, but E_{square} only approximates 5.9%. If the input dimension is greater than two, the needed memory size will be enormous.

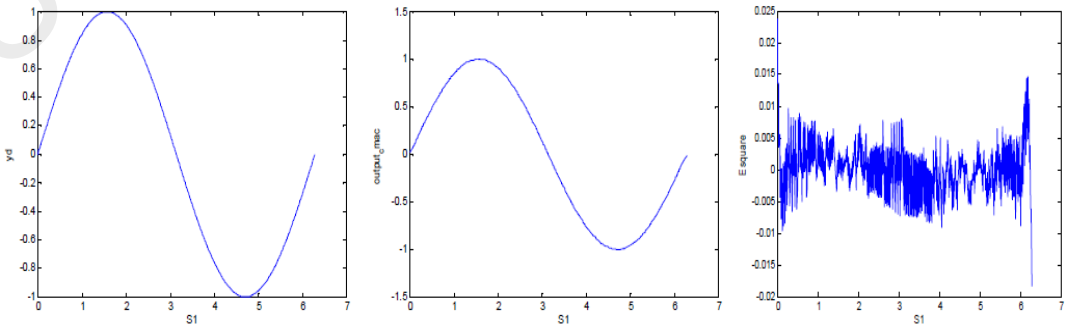


Figure 4.5: SIN function, CMAC based approximated SIN function, and mean square error

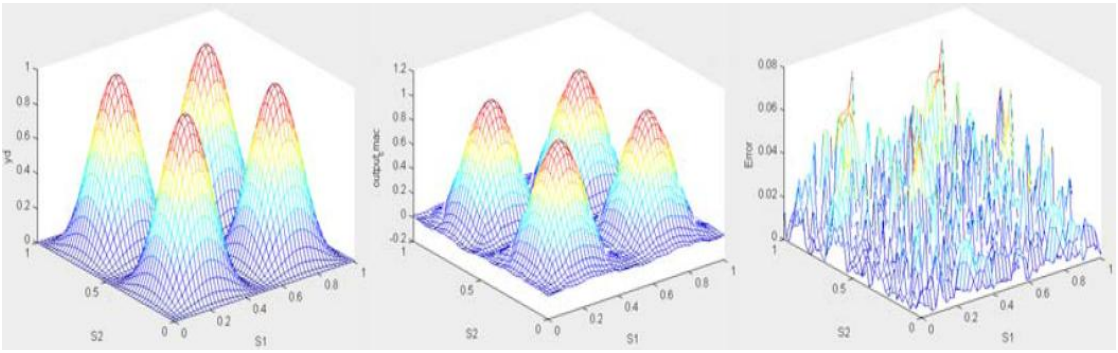


Figure 4.6: FSIN function, CMAC based approximated FSIN function, and mean aquare error

Table 4.1: Comparing of required memory size for one dimension input and two dimension input CMAC

Function	Δq	N_l	E_{square}	N_c
SIN	0.0148	15	0.58%	434
FSIN	0.00247	41	5.9%	4940

In this table, the second column, called "quantization step" Δq, allows to map a continuous signal into a discrete signal. In fact, Δq corresponds to the relative displacement between cells of the two consecutive layers as well. The third parameter, called "generalization parameter" N_l corresponds to the number of layers. The total number of used memory size (number of cells) shows by N_c and E_{square} represent mean squared error.

4.5 Simulation Examples

To consider the learning effect of HCFCMAC, obtain the following non-linear sample to study the learning consequences of CMAC, FCMAC as well as proposed HCFCMAC.

A) One-dimensional non-linear sample

Use the following function that is expressed in equation (4.38) respectively as sample to compare different type of CMAC.

$$y(x) = \sin x + \cos x \qquad -\pi \leq x \leq \pi \tag{4.38}$$

In the training process, total absolute error (TAE) and root mean square error (RMSE) are used to show the model learning accuracy and speed.

$$TAE = \sum_{i=1}^{n} \left| Y_i^d - Y_i \right| \tag{4.39}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(Y_i^d - Y_i \right)^2} \tag{4.40}$$

Where, n denoted the total iteration number, Y_i^d denoted desire output value of i iteration and Y_i is the real learned output data. CMAC, FCMAC and HCFCMAC trained within 20 iterations.

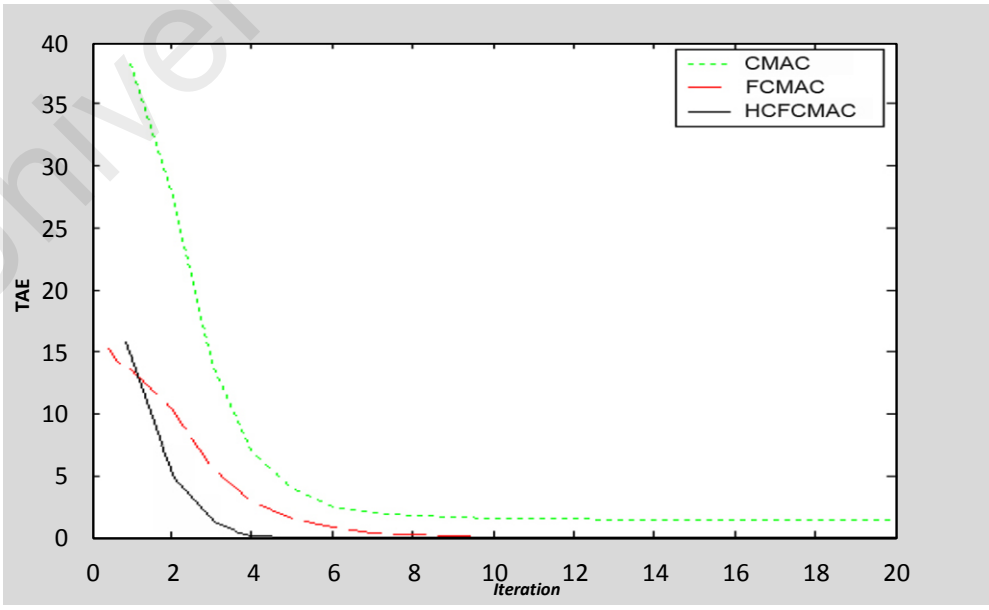


Figure 4.7: One-dimensional total absolute error (TAE)

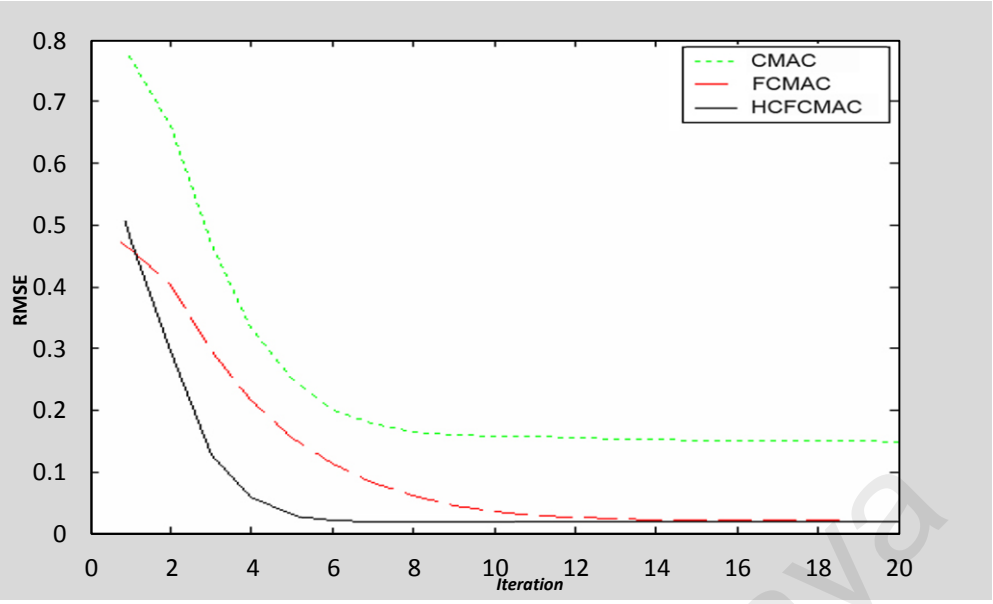


Figure 4.8: One-dimensional root mean square error (RMSE)

B) Two-dimensional non-linear sample

For this example presume the following SIN function:

$$y(x_1, x_2) = \sin(x_1, x_2) \quad -1 \leq x_1, x_2 \leq 1 \tag{4.41}$$

TAE and RMSE are calculated same with equations (4.39) and (4.40). The following figure shows descend of the errors for CMAC, FCMAC and HCFCMAC.

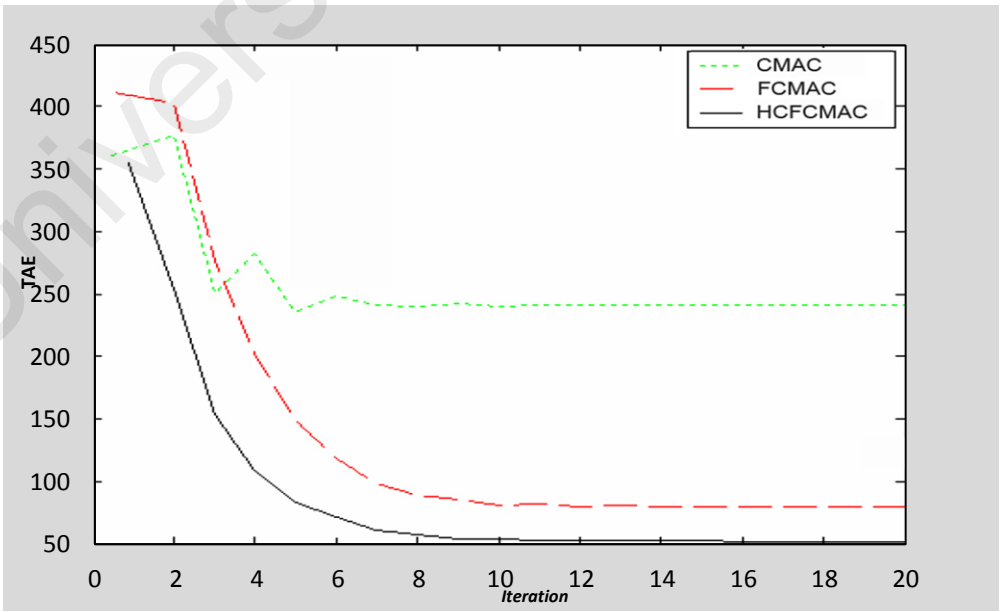


Figure 4.9: Two dimensional total absolute error (TAE)

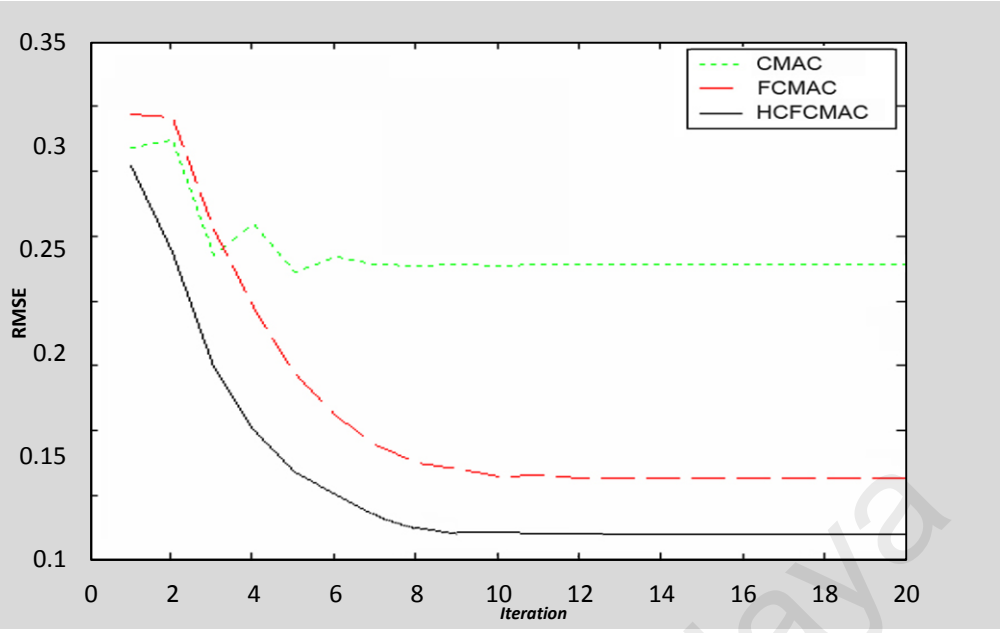


Figure 4.10: Two dimensional root mean square error (RMSE)

Above figures (4.7) ~ (4.10), shows the learning process of one or two dimensional non-linear sample with various CMAC model. There is a huge difference in the error modelling speed. However, HCFCMAC has highest converge speed and learning accuracy among all models and basic CMAC has lowest converge speed and learning accuracy. As well, these figures shows, as the increase of dimension, the computation mounts grow sharply that cause the gaps in accuracy and learning speed even more distinct.

4.6 Robotic Arm controller based on HCFCMAC

4.6.1 Conventional inverse kinematics

For a kinematic manipulator, the joint variable used to compute the end effector position function that is expressed as the following equation:

$$\dot{X} = \mathrm{J}(\theta)\dot{\theta} \tag{4.42}$$

Where, $\dot{X}$ is the end effector velocity vector, $\dot{\theta}$ is the joint velocity vector and $\mathrm{J}(\theta)$ is the Jacobian matrix. Our target (control robot arm) concern is to solve the inverse kinematic as below equation:

$$\dot{\theta} = \mathrm{J}^{-1}(\theta)\dot{X} \tag{4.43}$$

To solve the equation (4.43), it is used pseudo inverse of the Jacobian which minimizes the norm of joint velocity. So, the equation can becomes as follow:

$$\dot{\theta} = \mathrm{J}^{+}(\theta)\dot{X} + (1\_\mathrm{J}^{+}\mathrm{J})\dot{\mathrm{q}}_{\mathrm{a}} \tag{4.44}$$

Where; $\dot{\mathrm{q}}_{\mathrm{a}}$ defines the vector of arbitrary joint velocity planned in the null space of J. It may be resolved the redundancy by finding $\dot{\mathrm{q}}_{\mathrm{a}}$. As we can see, above formulation need a lot of calculation to solving the equations and it is not useful for real time control of robotic arm. Thus, to reduce the computation time and memory usage in this example of real time control (Robotic arm), employing of HCFCMAC algorithm is proposed. The results will show the fast learning and rapid computation of this model. After applying the proposed model to control robotic arm the memory size of HCFCMAC is compared with FCMAC.

4.6.2 Learning process

Network learning is established on observed training input vector X_s, the desired output $Y^d(t)$ and the network output(t) . Actually the learning process described in 4.3.2, which is divided to 3 main steps, 1) Specific the winner neuron for input X at the i^{th} iteration , 2) Retrieve the output and 3) Update the memory.

4.6.3 Solving inverse kinematic problems

The inverse kinematic application is showed in figure (4.11). From the figure we can see that:

$$\dot{X}^{t+1} = \dot{X_d}^{t+1} + K_e \tag{4.45}$$

Actually the main part of diagram is the HCFCMAC box which is solved the inverse kinematic problem.

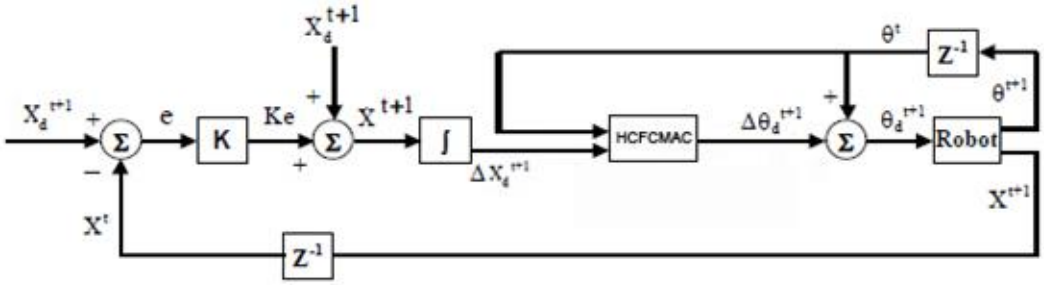


Figure 4.11: Block diagram of HCFCMAC for solving the kinematic problems

Where, (θ^t) defines the inputs which are the current manipulator confirmation at time (t) and ΔX_d^{t+1} the desired position increment at time $(t+1)$. The joint angles are showed as output. Here HCFCMAC used to generate the initial solution $\Delta\theta^{t+1}$, the difference between initial solution and the desired solution is used to update the weights of the HCFCMAC. For optimization and reduce the error and minimize the joint angle changes the following equation is used.

$$\frac{\partial U}{\partial \Delta \dot{e}} = -\alpha e^T J + \beta \Delta \dot{e}^T \tag{4.46}$$

Where, the position error vector is defined as e, α *and* β are the weight constant and J is the Jacobian matrix. Thus, we can have:

$$\Delta \dot{e}_{k+1=} \Delta \dot{e}_k - \frac{\partial U}{\partial \Delta \dot{e}} \tag{4.47}$$

Where, k is the number of iteration.

4.6.4 Simulation

During this control example, a 5 link planer manipulator is considered. Each link is connected with revolute joints as shown in figure 4.12 and figure 4.13 (Craig, 1986). The Denavit Hartenberg notation of the manipulator transpires as in Table 4.2.

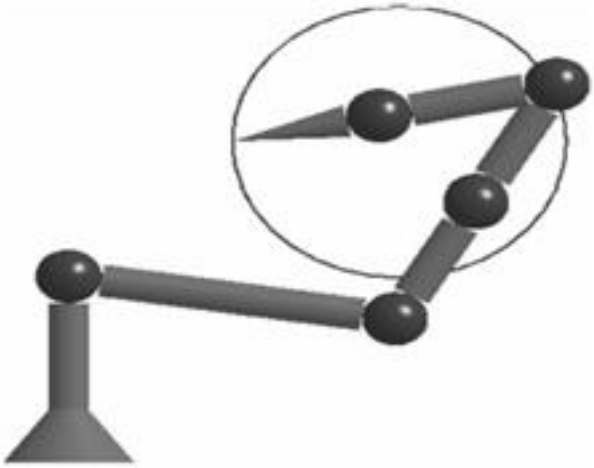


Figure 4.12: Initial configuration of the Robot arm (Adapted from Craig, 1986).

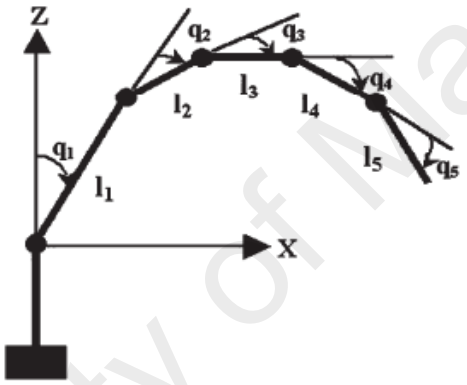


Figure 4.13: Planar Manipulator diagram (Adapted from Craig, 1986).

The kinematic equations of the robot arm control can be calculated as:

$$\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} l_1 S_1 + l_2 S_{12} + l_3 S_{123} + l_4 S_{1234} + l_5 S_{12345} \\ l_1 C_1 + l_2 C_{12} + l_3 C_{123} + l_4 C_{1234} + l_5 C_{12345} \end{bmatrix} \tag{4.48}$$

Table 4.2: Denavit-Hartenberg parameters

Link	a_i	α_i	d_i	q_i
0	0	$-\pi/2$	0	*0*
1	l_1	0	0	$q_1 - \pi/2$
2	l_2	0	0	q_2
3	l_3	0	0	q_3
4	l_4	0	0	q_4
5	l_5	0	0	q_5

where: l_1; l_2; l_3; l_4 and l_5 is the link-length, respectively q_1; q_2; q_3; q_4 and q_5 is each joint's angle, respectively $S_1 = \sin(q_1)$, $S_{12} = \sin(q_1+q_2)$, $S_{123} = \sin(q_1+q_2+q_3)$, $S_{1234} = \sin(q_1 + q_1 + q_3 + q_4)$, $S_{12345} = \sin(q_1 + q_2 + q_3 + q_4 + q_5)$, $C_1 = \cos(q_1)$, $C_{12} = \cos(q_1 + q_2)$, $C_{123} = \cos(q_1 + q_2 + q_3)$, $C_{1234} = \cos(q_1 + q_2 + q_3 + q_4)$, and $C_{12345} = \cos(q_1 + q_2 + q_3 + q_4 + q_5)$. The equations between the end-effector and the joint angular velocity showed as below:

$$\dot{x} = \mathbf{J}(q)\dot{q} \tag{4.49}$$

Where

$$\mathbf{J}(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial X}{\partial q_1} & \frac{\partial X}{\partial q_2} & \frac{\partial X}{\partial q_3} & \frac{\partial X}{\partial q_4} & \frac{\partial X}{\partial q_5} \\ \frac{\partial Y}{\partial q_1} & \frac{\partial Y}{\partial q_2} & \frac{\partial Y}{\partial q_3} & \frac{\partial Y}{\partial q_4} & \frac{\partial Y}{\partial q_5} \end{bmatrix} \tag{4.50}$$

And

$$\frac{\partial X}{\partial q_1} = l_1C_1 + l_2C_{12} + l_3C_{123} + l_4C_{1234} + l_5C_{12345}, \tag{4.51}$$

$$\frac{\partial X}{\partial q_2} = l_2C_{12} + l_3C_{123} + l_4C_{1234} + l_5C_{12345},$$

$$\frac{\partial X}{\partial q_3} = l_3C_{123} + l_4C_{1234} + l_5C_{12345},$$

$$\frac{\partial X}{\partial q_4} = l_4C_{1234} + l_5C_{12345}, \qquad \frac{\partial X}{\partial q_5} = l_5C_{12345}$$

$$\frac{\partial Y}{\partial q_1} = -(l_1S_1 + l_2S_{12} + l_3S_{123} + l_4S_{1234} + l_5S_{12345}), \tag{4.52}$$

$$\frac{\partial Y}{\partial q_2} = -(l_2S_{12} + l_3S_{123} + l_4S_{1234} + l_5S_{12345}),$$

$$\frac{\partial Y}{\partial q_3} = -(l_3 S_{123} + l_4 S_{1234} + l_5 S_{12345}),$$

$$\frac{\partial Y}{\partial q_4} = -(l_4 S_{1234} + l_5 S_{12345}), \qquad \frac{\partial Y}{\partial q_5} = -l_5 S_{12345},$$

In our simulation the end effector trajectory is centered at (0.47, 0.56) and the desired angular velocity w is 0.478 π rad/s. Also, l_1 is equal 0.5 m, l_2 *and* l_3 are 0.25m and fourh and fifth link are 0.27m.

$$\begin{bmatrix} X_d(t) \\ Y_d(t) \end{bmatrix} = \begin{bmatrix} 0.47 - 0.25\ \cos(wt) \\ 0.23 + 0.25 \sin(wt) \end{bmatrix} \tag{4.53}$$

The cycle time is 4.5 seconds. In our simulations system, the norm error is defined as:

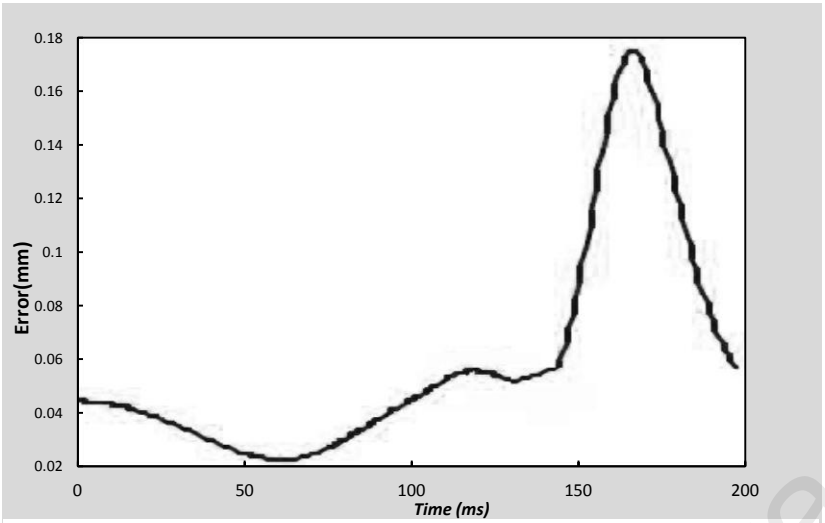
$$E_{norm}(t) = \sqrt{[X_d(t) - X(t)]^2 + [Y_d(t) - Y(t)]^2} \tag{4.54}$$

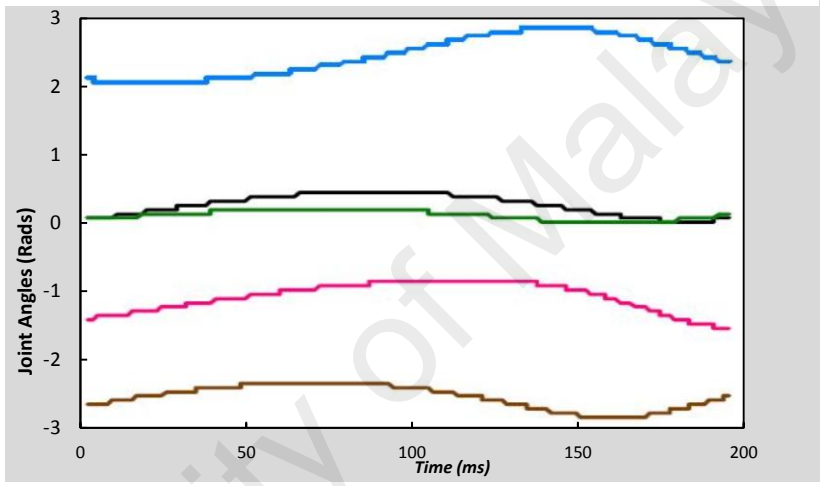
The average error is defined as:

$$E_{av} = \frac{\sum_{n=1}^{2\pi/(w\Delta t)} E_{norm}(n\Delta t)}{2\pi/(w\Delta t)} \tag{4.55}$$

4.6.5 Simulation Results

a) First of all the results of manipulator control by CMAC showed, these results observed after 100 iterations. It shows the norm error is become stable with maximum norm error on 0.18 mm. the number of memory is 10^4 as showed in table 4.2





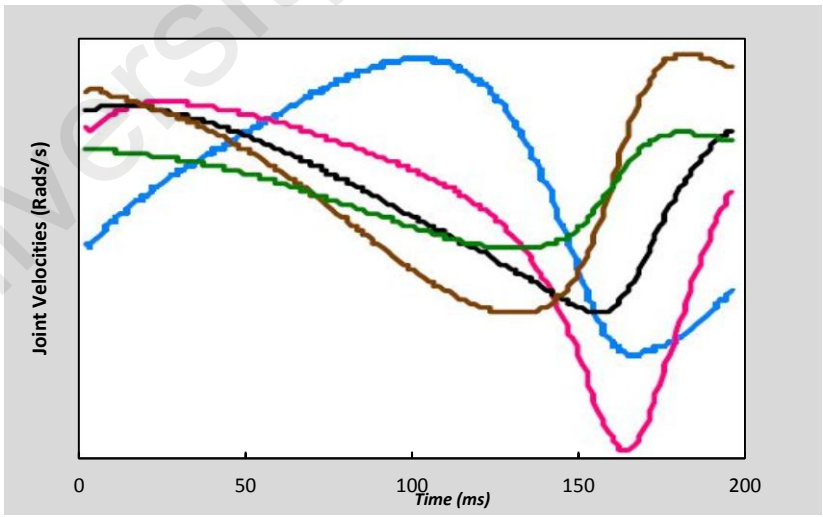


Figure 4.14: The norm error, joint angles and joint velocity after 100 iteration, using CMAC

b) Now HCFCMAC is applied and have it trained step by step, the maximum norm error is about 0.3 mm after 1000 trainings. The network learning constant is 0.1. Thus the HCFCMAC can really reduce the training time and to check the stability of the system it tested with 3000 training step.

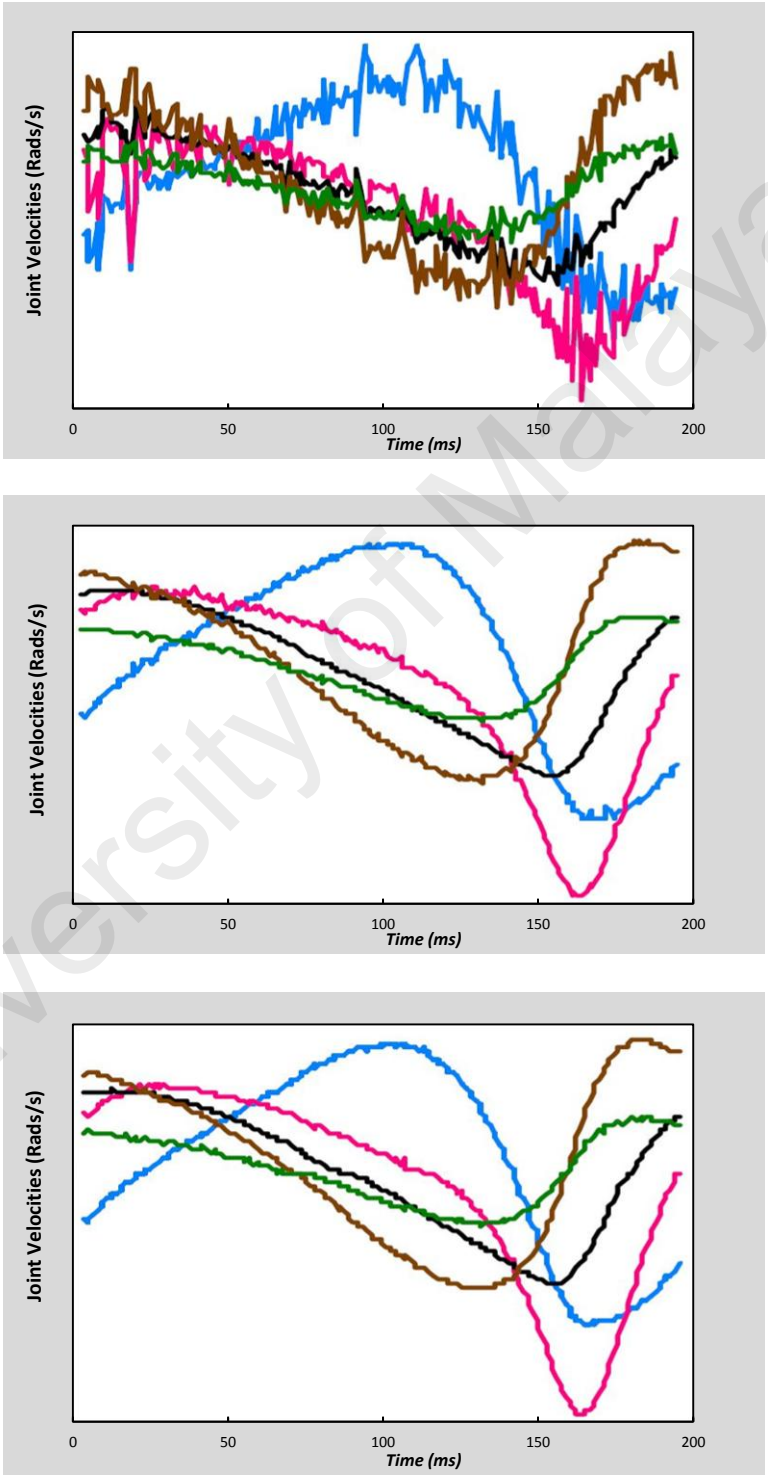


Figure 4.15: The joint velocity using HCFCMAC, number of cycles=100, 1000, and 3000

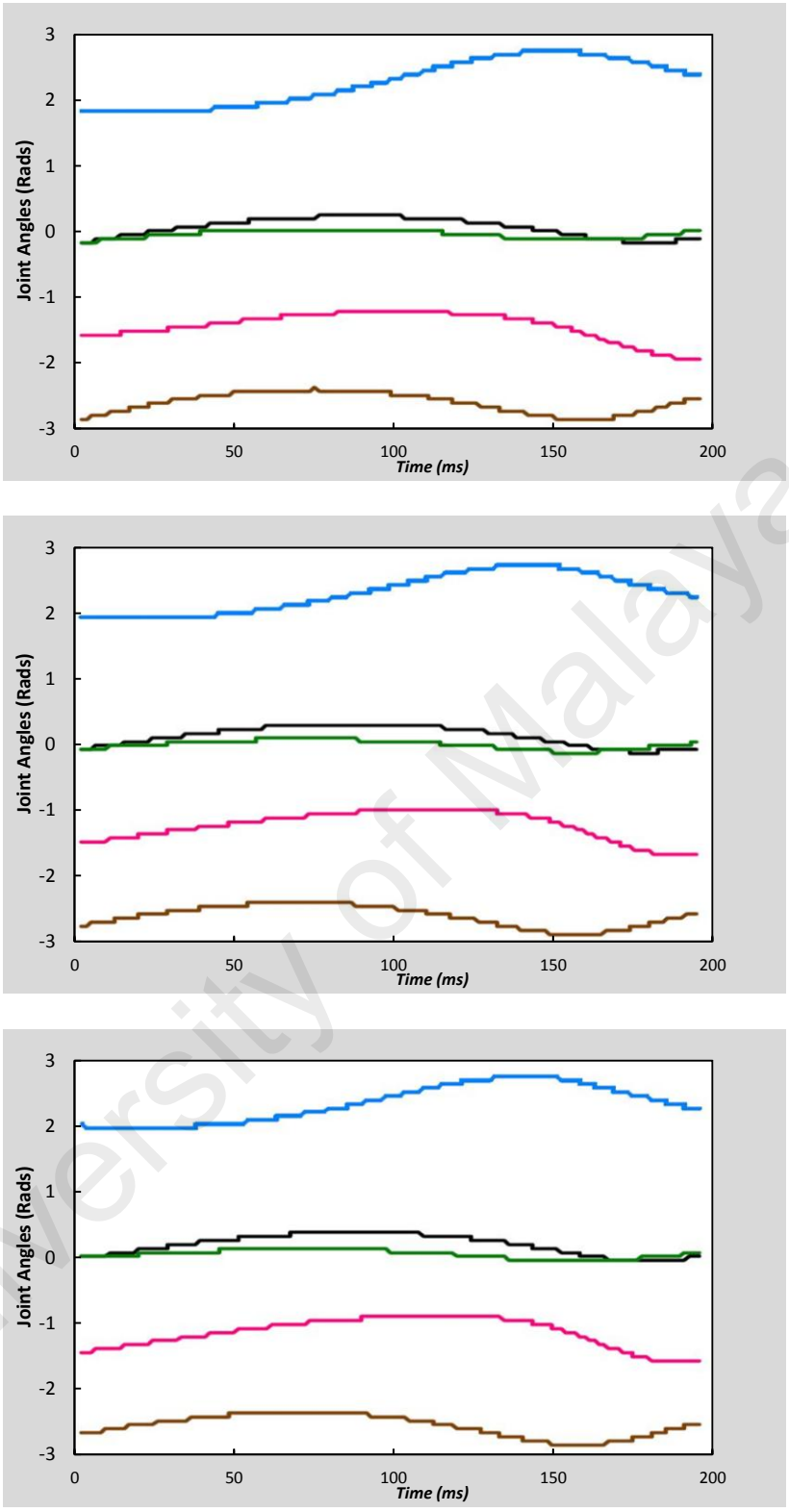


Figure 4.16: The joint angles using HCFCMAC, number of cycles=100, 1000 and 3000

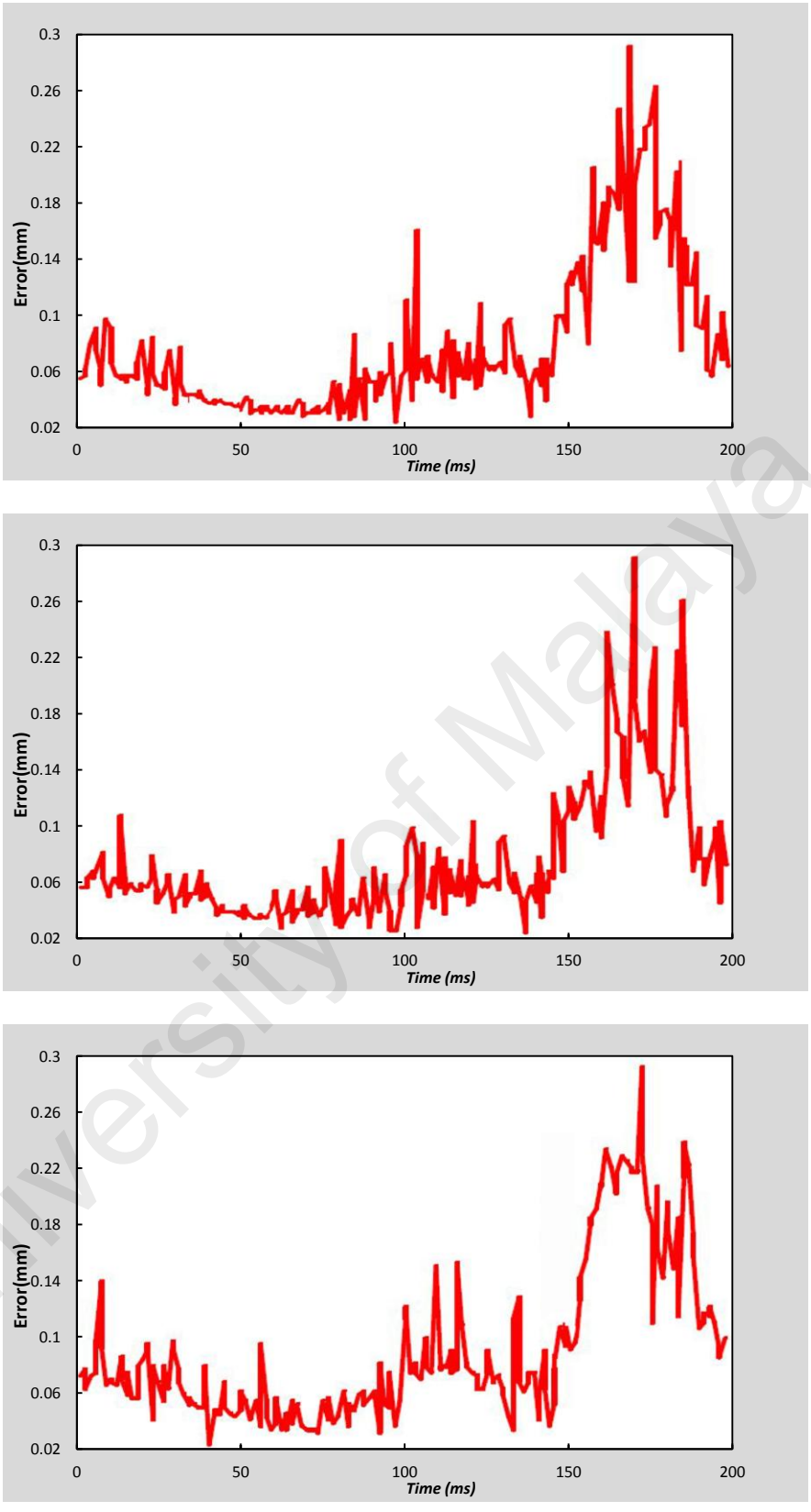


Figure 4.17: The norm error using HCFCMAC, number of cycles=100, 1000 and 3000

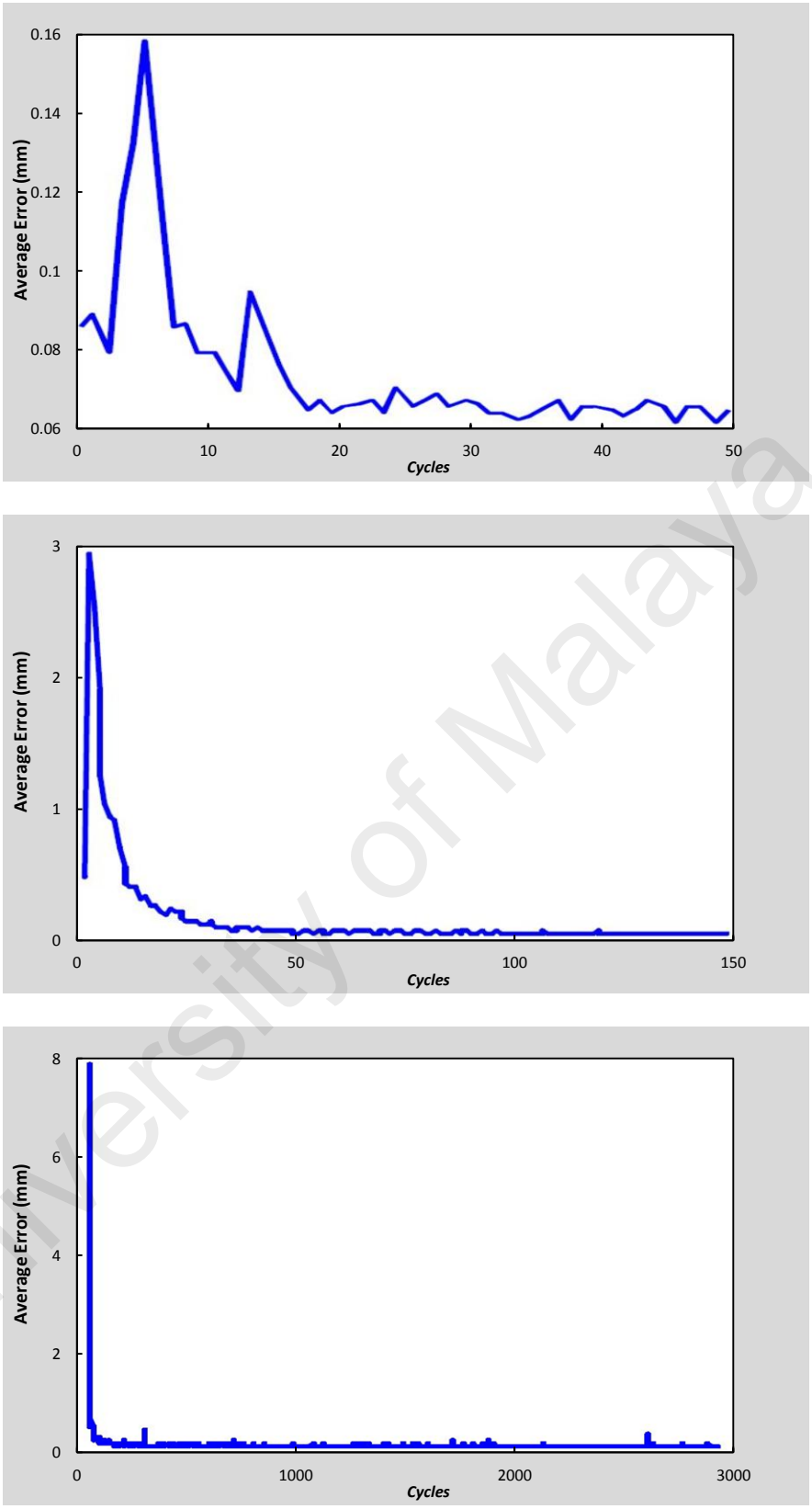


Figure 4.18: The average error using HCFCMAC, number of cycles= 3000

From the figures see the norm error after 3000 training is same as 1000 training and the average error is mostly less than 0.1mm. Above figures showed the HCFCMAC

converges more quickly compared to conventional CMAC. The above figures showed the computing time to control robotic arm based on HCFCMAC is reduced and learning accuracy is increased.

c) The table 4.2 layouts the performance of HCFCMAC compare to CMAC on memory size allocation. The system was trained within 500 training cycles. The neighborhood constant is equal 0.2 and learning constant is equal 0.1. To determine the performance of each model the memory network size is varied. The relation between the trained network cells and network size is defines as Cell Occupancy Rate (COR). The system performance is measured by average of joint angle (AJA) and average of joint velocity (AJV). The procedure of memory allocation of HCFCMAC is described on section 4.3. The simulation robotic arm has 5 joints (5 DOF), thus, the input dimension is equal to 5.

Table 4.3: comparison results for the various CMAC using robot arm controller

Networks	**Memory size** (5 input dimension)	**Total cells**	**Train Time**	**Trained cells**	**AJA** (Rads)	**AJV** (Rads/s)	**COR**
CMAC	7^5	16,807	24297	644	1.7272	1.3131	3.83%
	8^5	32,768	55245	816	1.7016	1.4071	2.49%
	10^5	**100,000**	**77531**	**1520**	**1.7594**	**1.2068**	**1.52%**
HCFCMAC	**5^5**	**3,125**	**14234**	**365**	**1.690**	**0.1591**	**11.68%**
	6^5	7.776	25796	534	1.7278	1.2079	6.86%
	10^5	100,000	89256	2140	1.7316	1.4054	2.14%

According to above table, the CMAC take an optimal efficiency with a memory size of 10 cells per dimension and the HCFCMAC based controller has better performance compare to basic CMAC model. An optimal HCFCMAC performance achieved with a memory size of 5 memory cells per dimension. The HCFCMAC allocate available

memory cells to the input regions with high utilization function this cause more cells allocate to the important input regions which consist more information. In addition, the smaller network size in HCFCMAC fewer cells are allocated to region with little or no training data thus improve the generalization ability. The required training time is reduced by memory allocation scheme of HCFCMAC (i.e. 14234 ms). As well table showed the efficiency of proposed HCFCMAC memory allocation observe the higher COR Value compare to another model.

4.7 Summary

This chapter present a human brain inspired CMAC model which is named the Hierarchically Clustered Fuzzy Cerebellum Model Articulation Controller (HCFCMAC) network. Encouraged by study of the functions of the human cerebellum, in addition to the neurobiological mechanisms of the human brain, the HCFCMAC operates as a memory allotment scheme. The HCFCMAC contains a self-organizing input space module and a hierarchical network to facilitate the fuzzification process. There is an important increase in the computational intricacy of the model which approved by simulation results. Hierarchically clustered technique applied to FCMAC after operating input fuzzification to allocate more memory cells to the input domains which contain further training signals this cause to reduce the memory size. The function of the HCFCMAC was evaluated in two non-linear example as well as control robotic arm.

The proposed model has not yet been optimized for learning conditions or speed which is beyond the scope, and was applied to a relatively simple system. The real power or weakness of the model may only be realized if the model is efficiently scaled up to higher degree of freedom (DOF) problems, mobile robots and robust robots. Different models of CMAC have difference in the error descent speed. As it showed in

the results the HCFCMAC converges more quickly compared to conventional CMAC. It has high accuracy in the learning and reduces the memory size. For further investigation of HCFCMAC performance this proposed model will be used as a controller to compute and predict the trajectories of the swing leg in biped robot on chapter 6.

CHAPTER 5

Footstep Planning and Semi Online Fuzzy Q-learning

5.0 Footstep Planning based on Semi Online Fuzzy Q-learning

5.1 Introduction

A significant part of the success of robotic movement capabilities is the ability to take steps in an environment with obstacles. Between heuristics and fuzzy logic, the optimal footstep planning strategy can be achieved depending upon the weight and height of the biped robot. The planned motion of the biped robot can be classified as low level or high level which requires a more complex programming language. Walk plan motions that have similar identical sequences may still be different in other aspects. In order to accomplish a simulation in robotic movement, the method must include consideration for footstep planning, obstacle avoidance, and trajectory. The processes for footstep motion are not the same as for the robot arm swing. Most methods use human joint trajectories by regression or by dynamic primitives.

For this study, the robot will take steps by footstep planning based upon the Semi-Online Fuzzy Q-Learning algorithm. Each algorithm represents a robot function for one degree of freedom and is suitable for online learning. The SOFQL is used as the reinforcement learning method which will also address the generalization issues associated with footstep planning. The SOFQL allows continuous states and actions that also improve the speed of convergence. This is accomplished by using the latest learning data. Below is the simple model of biped robot made by Kuffner when planning the footstep strategy for the biped robot:

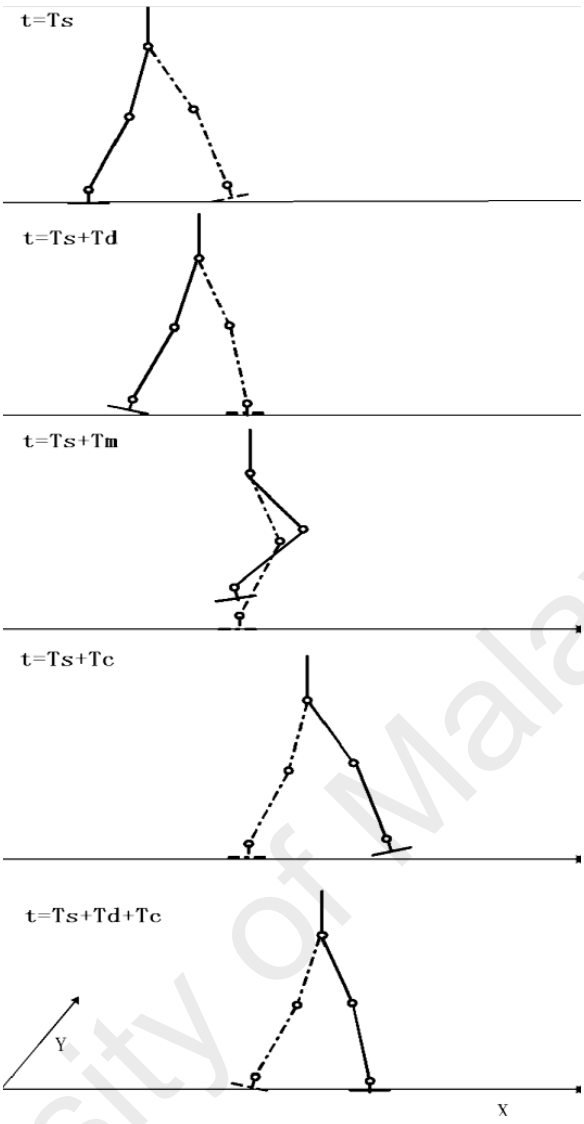


Figure 5.1: Kuffner Walking Cycles, (adapted from Tang, 2002)

In this way, the variables for walking cycle are as follow:

T_c = *time period for one walking cycle*

T_d = *interval of double support phase*

T_s = *single support phase*

T_m = *highest point*

Presume right leg is the swing leg. Walking cycle variants:

$t = T_s$ *to* $t = T_s + T_d$

$t = T_s + T_d$ *to* $T_s + T_c$

$t = T_s + T_m$

Kuffner's biped robot foot placement navigation assumptions:

1. The environment floor is flat and contains non-moving obstacles of disclosed position and height.
2. A discrete set of feasible, statically-stable foot-step place positions and associated stepping motions are pre-calculated.
3. The floor surface is prepared for predetermined placement of the feet rather than the use of an obstacle surface (Kuffner et al., 2001)

When the foot is reground, this is considered a type of collision.

5.2 Design of Semi On-line Fuzzy Q-Learning algorithm

5.2.1 Q-learning Algorithm

Actually, in artificial machine control learning field resources are so poor and insufficient to utilize supervised learning algorithms. As well as there is even no precise information about the data on which learning should be done. Besides there might be no previous sample data set available, where some unsupervised learning approaches depend on. Reinforcement Learning is developed to solve the above problem (Alireza Ferdowsizadeh Naeeni, 2004).

Reinforcement learning is the problem faced by an agent that should learn behaviour by way of trial and error interactions with a non-linear environment. Strategies for solving reinforcement learning problems are divided in two main categories. One is to search in the space of behaviours in order to find one that performs well in the environment. The second one is to use statistical techniques and dynamic programming methods to estimate the utility of taking action in states. The Q-learning algorithms are based on the second strategy. The advantages of using Q-learning algorithm are the possibility of executing new behaviours without any previous phase such as "on site

manual tuning" or "off line learning. Another important typical is that it is an off policy algorithm. The optimal state-action mapping is learning freely of the policy being followed.

In Q-learning modelled as a Markov Decision Process it observes discrete states of the world $s_t \in S$ and can execute discrete actions $a_t \in A$. At each discrete time step, the agent observes state s_t, takes action a_t, observes new state s_{t+1}, and receives immediate reward γ_t. Transitions are probabilistic, that is s_{t+1} and γ_t are drawn from stationary probability distributions.

The diagram of Q-Learning algorithm is shown in figure 5.2:

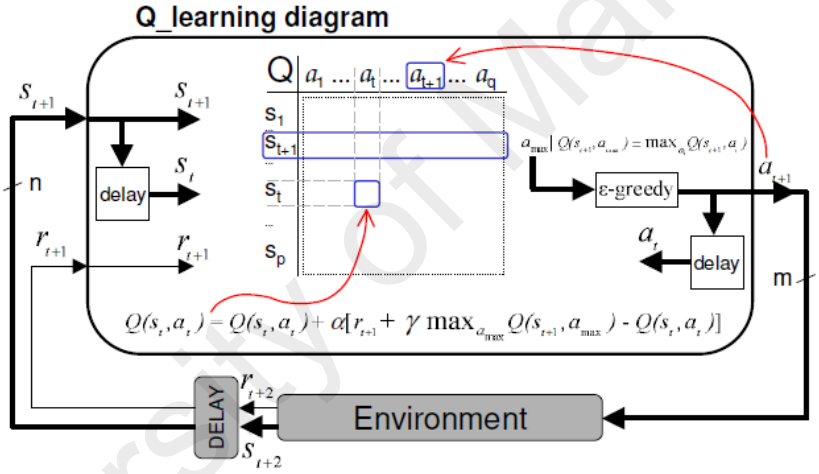


Figure 5.2: Diagram of Q-Learning algorithm (Adapted from Carreras et al., 2003)

The first version of Q-Learning is based on the temporal differences of order 0, while only considering the following step. The estimated value of the new state $V_t(s_{t+1})$, that is defined by:

$$V_t(s_{t+1}) = \max_{b \in a_{t+1}} Q(s_{t+1}, b) \tag{5.1}$$

The update corresponds to the equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \beta[r_t + \gamma V_t(s_{t+1}) - Q(s_t, a_t)] \tag{5.2}$$

Parameter γ can be chosen in [0,1] . If γ is close to 0, the agent will tend to consider only the immediate reward *r*. If γ is closer to 1, the agent will consider the future rewards with greater weight. β is the learning rate. The update corresponds to the centre of the old and the new rewards, weighted by β. The equation 5.2 can update as follow:

$$Q(s_t, a_t) = r_t + \gamma . \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \quad (5.3)$$

5.2.2 Design of reinforcement signal

The reward function *r* is the reward or punishment action, which appraises the contribution of every action for the agent to achieve the goal. The reinforcement signal *r* can be designed in several ways according to the given problems, but normally classified into two main branches (Dayan and Watkins, 1992).

- Shortest path problems

In this case, the reward function is equal to -1 in all cases if the present state is different from the desired state, in order to force the agent to reach this state as quickly as possible.

- Avoiding problems

The reinforcement signal is set to be 0 in all states, except in case of failure, where it takes the value -1.

For example: in the robot soccer game, a behavioural variety is encouraged with the following reinforcement signal: if the team scores $r = 1$, if the opponent scores $r = -1$, in other cases $r = 0$.

5.2.3 Choose of policy

After convergence of Q-Learning, to every state, the optimal policy is performed while choosing the action that maximizes the Q function:

$$a = \text{argMaxQ(x, b)} \quad where\ b \in A_x \tag{5.4}$$

This policy is called greedy. But, the too fast choice of the action which has the biggest Q value will drive to local minima. Thus, it is necessary to insure that all actions were tested sufficiently in order to get a useful evaluation of Q. It is the so called the phase of exploration in opposition to the exploitation one. At each state, the agent must choose between an action for whose expected reward is supposed to be good quality, or an action whose quality can be less good but for which application could drive it in promising zones. Obviously, explore all the possible actions with all the states costs abounded times.

Here, we list famous methods of exploration that are used (Sutton, 1998):

- Pseudo-stochastic Method

The action with the best value has a probability *P* to be chosen. Otherwise, an action is chosen randomly among all possible actions in the given state.

- Pseudo-exhaustive Method

The action with the best value has a probability P to be chosen. Otherwise, one takes the action the least lately chosen in the given state.

- Boltzmann Distribution

The action *a* is chosen with the probability:

$$P(a|s) = \frac{\exp(\frac{1}{T}Q(s,a))}{\sum_b \exp(\frac{1}{T}Q(s,a))} \tag{5.5}$$

T is comparable to "temperature" in simulated annealing. This parameter decreases in the time.

5.2.4 Fuzzy Q-Learning Algorithm

In the Q-Learning method, it deals with discrete cases and the whole state space can be enumerated and stored in memory, this method is either unpractical in case of large state-action spaces because the Q-values are stored in a look-up table. For continuous state space, Glorenec et al (1997), proposed to use fuzzy logic where both actions and Q function may be represented by Takagi-Sugeno FIS (C. Sabourin, 2007). In which there is only one conclusion for each rule, the Fuzzy Q-Learning (FQL) approach admit several actions per rule.

The Fuzzy Q-Learning (FQL) approach admits several actions per rule. Therefore, the learning agent has to find the best conclusion for each rule. To simplify, for each rule, we suppose that the learning system can choose one action among the total J actions. We call $a[i,j]$ the j^{th} possible action in rule i and $q[i,j]$ is its corresponding q-value. So, the FQL is built with competing actions for each rule:

If state is S_i, then choose $a[i,1]$, with $q = q[i,1]$

or choose $a[i,2]$, with $q = q[i,2]$

… …

or choose $a[i,J]$, with $q = q[i,J]$

The learning agent has to find the best conclusion for each rule, for example, the action with the maximum q-value. For every state S_i , the final action $A(s)$ is chosen through two levels of computation: in first level, local action $a[i,j]$ in each fired rule is determined, and in the second level global action is calculated among all the local actions. In our approach, ε-greedy algorithm is used to elect the local action in each

activated rule. The action with the best evaluation value has a probabilityε to be chosen, otherwise, an action is chosen randomly among all possible actions in the given state.

a) After the fuzzification of the perceived state S_i , the rule values $\alpha_i(s)s$ are computing using the following equation:

$$\alpha_i(s) = \mu_1^i \mu_2^i \dots \mu_J^i \tag{5.6}$$

b) The final action *A*(*s*) is computed through two levels of computation: in the first level, local action in each activated rule is determined, and in the second level global action is calculated as a combination of all the local actions:

$$A(s) = \sum_{i=1}^{N} \alpha_i(s).a[i,j] \tag{5.7}$$

$$Q(s) = \sum_{i=1}^{N} \alpha_i(s).q[i,j] \tag{5.8}$$

c) After the application on the environment of the new action given by *A*(*s*), the temporal difference error may be computed as:

$$\Delta Q = \beta[r + \gamma V_{max}(s') - Q(S, A(S))] \tag{5.9}$$

Where, r is the discount factor and β is the learning rate. $V_{max}(s')$ defined the maximum Q-value for the activated role.

d) Finally, for each activated rule, the corresponding elementary quality $\Delta q[i, j]$ of the Q matrix is updated as:

$$\Delta q[i,j] = \Delta Q.\alpha_i(s) \tag{5.10}$$

5.2.5 Semi Online Fuzzy Q-Learning

Neural Networks have a high capability; however they suffer from the interference problem whereby it occurs when learning in one region of the input space causes defeat of learning in other zones (Weaver et al., 1998). To solve this problem, the Semi Online Fuzzy Q-learning uses a database of learning samples. The principal purpose of the database is to include a characteristic set of visited learning samples that are repetitively used to update the algorithm. The advantage of the database can sort as:

a) The stability of the learning process and its convergence even in trouble problems.
b) The Q functions are regularly updated with samples of the whole visited state-action space, which are one of the conditions of the original Q-learning algorithm.
c) The updating of the network Q-learning is done with all the samples of the database, hence, the convergence is obtained with less repetition. Therefore the semi online mentioned to the fact which the current and previous samples are used in the learning (Carreras et al., 2003).

A result of the database is the speed up of the learning. The initial state define as s_t, the new state is s_{t+1}, the action a_t and the reward r_{t+1}. The learning samples are added to the database throughout the learning progression. The new samples replace older samples. The substitution is founded on the distance between vectors s_t, a_t and r_{t+1} of both samples. The sample is removed from the database if this distance is less than a density parameter *t* for any old sample.

The proposed algorithm can be divided in four different stages; this algorithm is shown in figure 5.3.

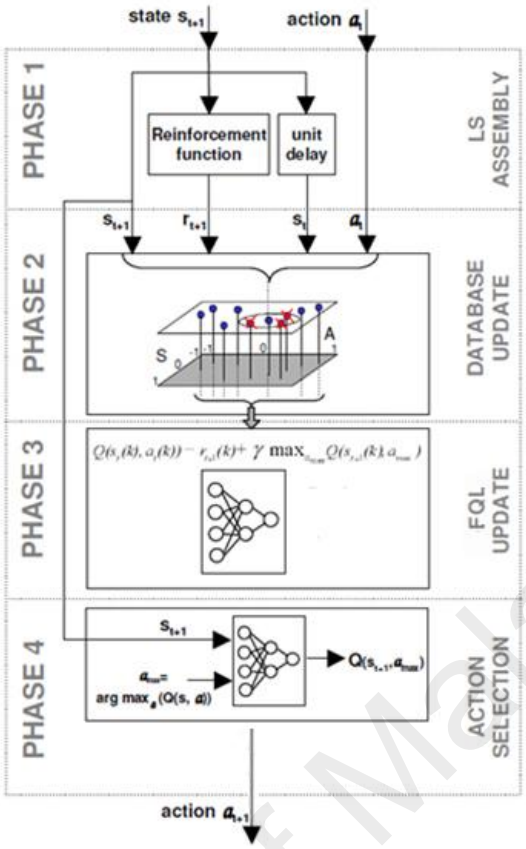


Figure 5.3: Phases of the Semi Online fuzzy Q-learning algorithm

In the first phase, the current learning samples are assembled, and then in the second phase, the database is updated with the new learning sample. According to, old samples which are similar to the new one will be removed. The third phase consists of updating the weights of the neural network based on fuzzy algorithm. If the Semi Online Fuzzy Q-learning algorithm is used in a non-linear real time system, this stage is carried out in an individual step. The final phase included selecting a new action r_{t+1}.

5.3 Path Planning

5.3.1 Definition of the Path Planning

In complicated environment distributing with obstacles "x", the robot, starting from initial position "*I*", finds its optimal way to reach the goal position "*G*" and away from

crashing with the obstacles at the same time. The distribution of obstacles "x" in the environment is shown in example figure 5.4. The robot can move toward four directions, but one step at a time, which can be expressed by four actions: step forward, step backward, step right and step left. Although this environment is relatively simple, it is representative for illustrating the robot's path planning problem.

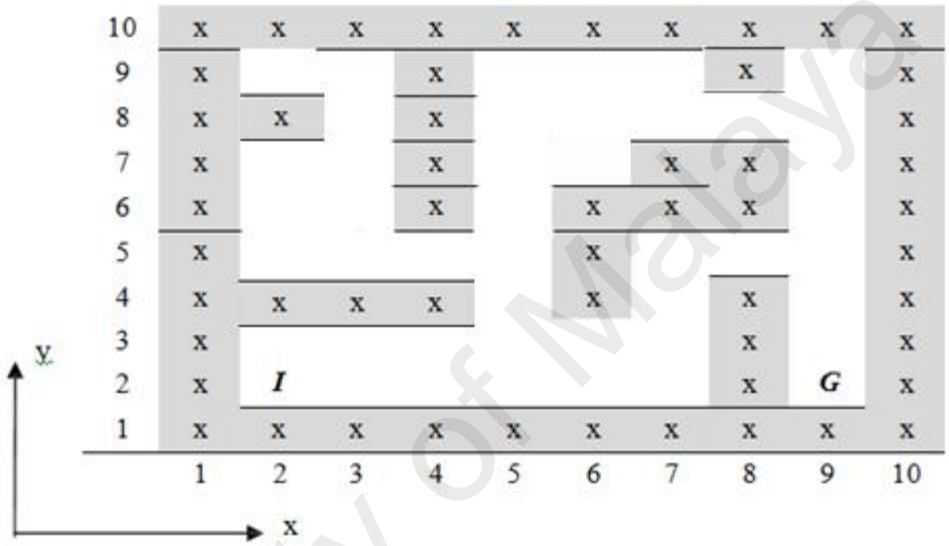


Figure 5.4: Example distribution of obstacles in the environment

The planning of the path of the biped robot is contingent upon factors such as the surface material and whether or not the robot will be carrying a load. Differences in these factors affect the friction and the ability of the robot to avoid obstacles. An optimal policy is performed while choosing the action that maximizes the fuzzy Q function: The too fast choice of the action which has the biggest Q value will drive to local minima. Thus, it is necessary to insure that all actions were tested sufficiently in order to get a useful evaluation of Q. It is the so called the phase of exploration in opposition to the exploitation one.

At each state, the agent must choose between an action for whose expected reward is supposed to be good quality, or an action whose quality can be less good but for which application could drive it in promising zones.

5.3.2 Simulation of path planning

Q-Learning method is usable to solve the path planning problem. The reward function r is actually the reward or punishment function, which evaluate the contribution of each action for the agent to achieve the goal. The reinforcement signal r can be configured in several ways based upon the given problems, but normally can be classified into two big parts (Watkins, 1992). The reinforcement signal r is equals to -1, when the robot facing the obstacle; in other hand r *is* set to be 1 when the robot reaches the goal location, as the robot in other positions, the rewards equals to $R(x, y)$, which can be expressed as followed equations. Representinv the rewards is inverse proportion to the distance from current position to the goal position.

$$r = \begin{cases} -1, & crash\ with\ equation \\ 1, & arrive\ at\ the\ goal\ position \\ R(x,y) & in\ other\ positions \end{cases} \tag{5.11}$$

In which:

$$R(x,y) = -\sqrt{(x - x_G)^2 + (y - y_G)^2} \tag{5.12}$$

The update of $Q(s_t\ ,\ a_t)$ matrix is according to equation (5.3).Parameter γcan be chosen in $[0, 1]$. If γ is close to 0, the agent will tend to consider only the immediate reward r. If γ is closer to 1, the agent will consider the future rewards with greater weight. β is the learning rate. The update corresponds to the barycenter of the old and the new

rewards, weighted by β. The actions which could be chosen by the robot is $= [a_1, a_2, a_3, a_4]$, standing for step forward, step backward, step right and step left relatively.

For the choice of policy, "Pseudo-stochastic Method" and "Pseudo-exhaustive Method" is applied respectively. For the Pseudo-stochastic method, Probability P is equal to the best action. The simulation result of the supposed robot's path planning problem is shown in figure 5.5(a) and figure 5.5(b).

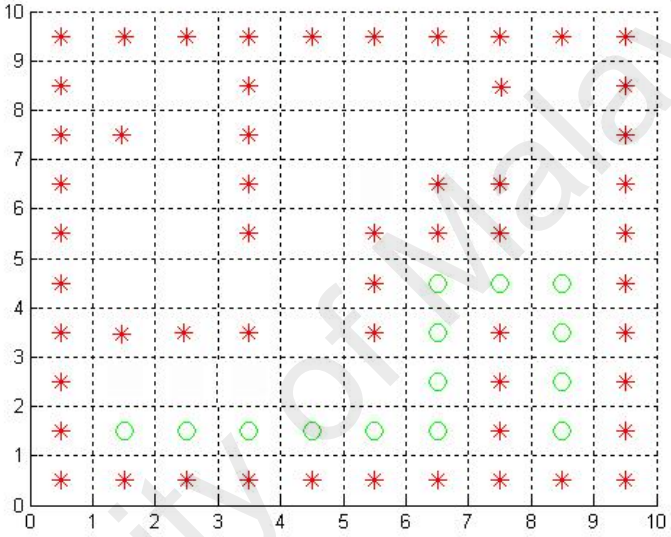


Figure 5.5(a): Chosen Pseudo-stochastic Method as policy for path planning

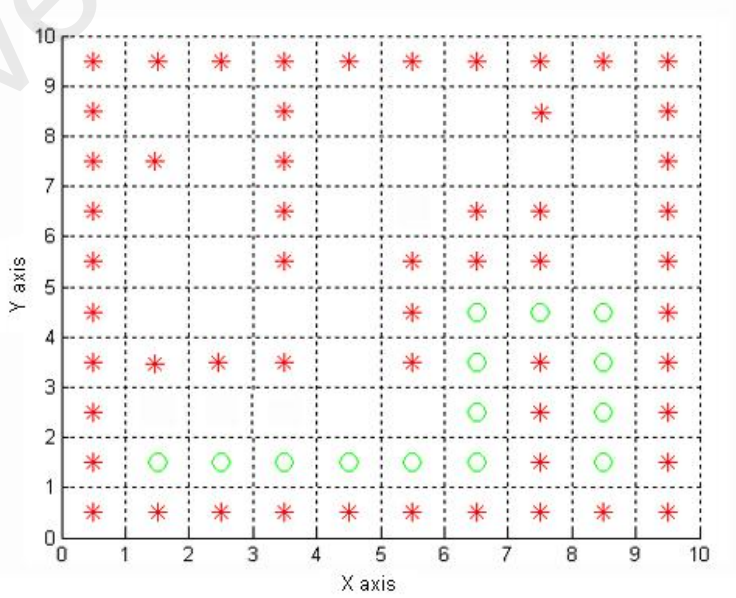


Figure 5.5(b): Chosen Pseudo-exhaustive Method as policy for path planning

The reinforcement signal is set to be 0 in all states, except in case of failure, where it takes the value -1. In the above figure, the Red Cross represents the position of obstacle, the green dots stands for the path of robot moving from initial state to goal position. After learning sufficiently (choose the episode of Q-Learning as 3000, and the learning steps of each episode is 1000), calculate the sum of reinforcement signal for each episode according to equation (5.13). The results are shown in figure 5.6(a) and 5.6(b) for both "Pseudo-stochastic Method" and "Pseudo-exhaustive Method" chosen to be its policy.

$$Sum\_r = \sum r_i \tag{5.13}$$

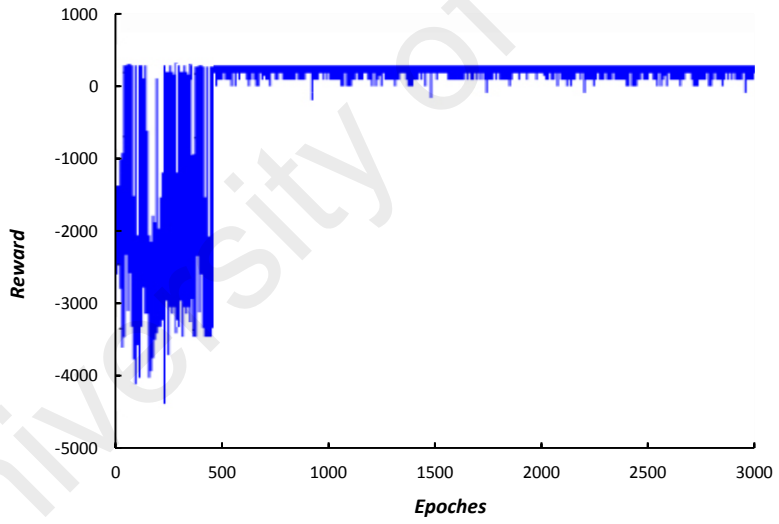


Figure 5.6 (a): The sum of reinforcement signal for each learning episode (stochastic)

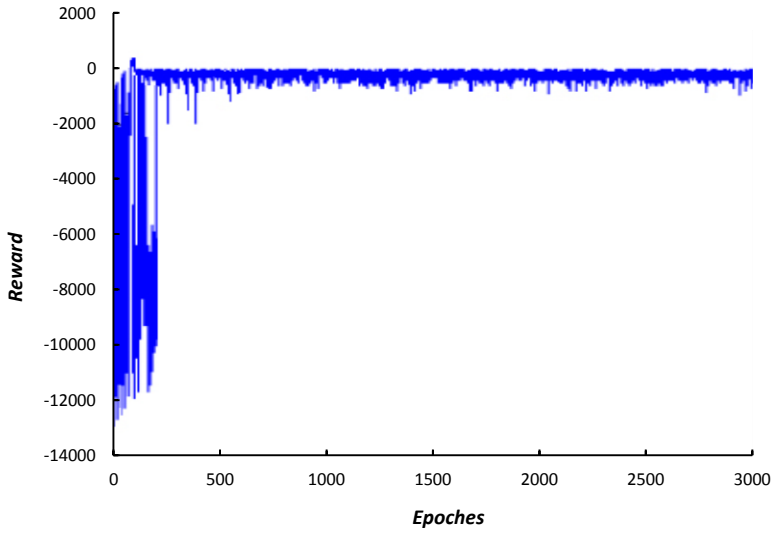


Figure 5.6 (b): The sum of reinforcement signal for each learning episode (exhaustive)

Based on the reinforcement signal designed, the reward is minus for every step, and therefore, the sum of the rewards of each episode is less than zero. Adopting the two policies respectively for the robot to choose the following action, both of the two cases can achieve the optimal path successfully. However, comparing with the learning time, the sum of reinforcement signal is convergence after 518 with Pseudo-stochastic method, while using Pseudo-exhaustive method, the sum of reward signals convergence after 358 learning episodes. This results show experimentally that the Pseudo-exhaustive Method is better than the Pseudo-stochastic Method considering the learning efficiency.

5.4 Swing Leg Trajectories

In order for human beings to walk, the brain must produce a plan for one of the legs to rise and then to reground the foot. The leg that the brain chooses to initiate the action is the swing leg. The leg that remains stationary is fixed (see figure 5.7).

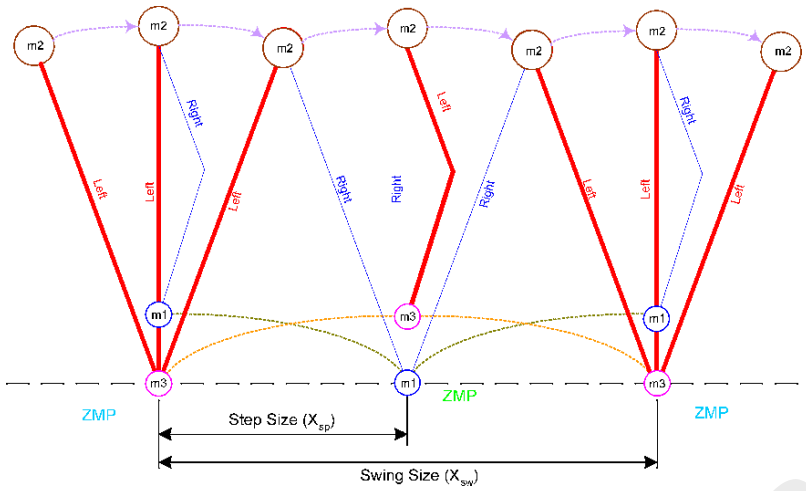


Figure 5.7: Swing leg and support leg in swing leg motion (Adapted from Wang, 2011)

The swing leg motion was first associated with the principal of a pendulum swing and research was performed on the theory as early as the 1830s. Trajectory planning and control for industrial robots equate joint trajectories that take minimum traveling time of motion between two points. Each joint trajectory is expressed by a quadratic polynomial function, thus the constraint on the joint jerk is considered in the optimization problem (Xiapong and Wang, 2011). The slope of the angle is also of great consideration because it also determines the amount of energy that must be exerted to accomplish the desired gait. The influence of the slope angle is similar to the step length. A steeper slope provides more energy input and thus the resultant gait is faster, an effect similar to decreasing the step length (Wisse et al., 2005).

The signals that are processed in the controller will seek the plan which uses the least amount of energy. Therefore, the greatest concerns for accomplishing a successful gait pattern in a robotic study are the amount of energy required for the dynamics as well as stability, overcoming obstacles and step length. The metabolic trade off of energy loss lies within the size of the steps, with the broad step requiring more energy than the narrow.

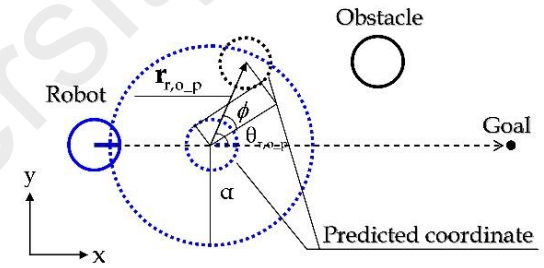
The human back step is also referred to as ‘swing leg retraction’ or ‘ground speed matching’ which researchers believe impacts stability, speed, and energy lost. “In human walking, the swing leg moves backward just prior to ground contact; for example, the relative angle between the thighs decreases. The hypothesis is that the swing leg retraction will positively affect the gait stability. Similar effects have been reported in passive dynamic walking models and running models. The gait of a simple forward progression of the legs requires a scheme independent of that for arm movement or for robots using limbs other than a leg. The conditions surrounding robot carrying a load are different than for walking, running, or wheeling. We assume for the purposes of this study that it is not necessary for the robot to take steps backward or to carry a load.

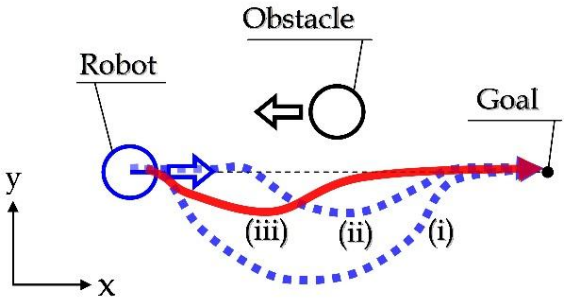
5.5 Control Strategy for Obstacle Avoidance

Walking in an environment with obstacles compounds the challenge of footstep planning. Accurate placement of the foot is needed to maintain balance and accomplish navigation. In Kuffner’s walking cycles, the environment had obstacles which were non-moving with known coordinates. This aspect is especially important for robots that are to be used indoors. Successful robot control schemes reduce the probability of the robot falling or colliding. Timing the avoidance of a fall or collision depends upon how fast the robot is moving. An obstacle avoidance strategy for robots in dynamic environment must be developed, which is accomplished in the way of the footstep planning method. As legged robots gain the abilities to walk and balance on more than just flat, obstacle-free floors, they grow closer to fulfilling the potential of legged locomotion shown by biological systems. To truly fulfill this potential these robots must successfully traverse complicated, rough terrain, requiring the robots to step onto or

over various features of the environment (Chestnutt et al, 2007). The control strategy in our model for obstacle avoidance is accomplished based upon the semi online fuzzy q-learning algorithm.

When robots are used indoors, the possibility of falling, colliding, or becoming trapped in a dead end must be reduced. A study by Suzuki & Takahashi (2011) predicted the coordinates for the robot's steps, taking into consideration the obstacle settings and successfully plotting a planned avoidance path (Figure 5.8). Relative velocity is used with relative position to calculate a position vector for the predetermined obstacle. The greatest concern in passive dynamic walking will be the size of the steps for the planning. The main drawback of this approach is it is operational only in the case of predictable dynamical environments (Chestnutt, 2005). Also, in order to limit the computational time, the path planning should be calculated with no more than 15 steps (Kuffner, 2001).





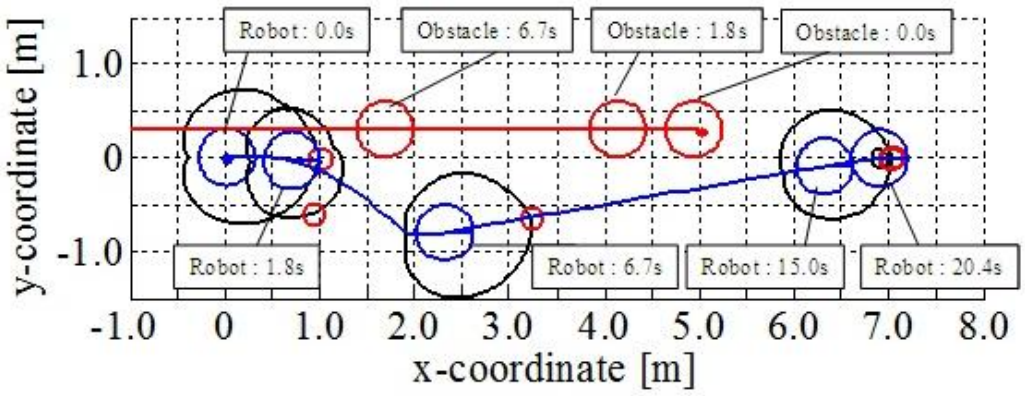


Figure 5.8: Obstacle setting in robot trajectory movement (Adapted from Suzuki & Takahashi 2011)

The footstep planning is a feasible method to solve the problem of the path planning for biped robots. The control strategy which solves the step over obstacle problem is based upon Kuffner's foot step planning approach.

Kuffner's strategy focuses on the construction of a search tree from a discrete set of footstep locations that correspond to chosen stepping motion trajectories.

The planner maintains a priority queue of search nodes containing a footprint placement configuration and a heuristic cost value. The search terminates when the next node falls within the predefined goal position, and the planning path back to the root node is returned.

5.6 Step leg planning for biped robot

The robot moves in an unknown environment, when the obstacle is static, where it is called "static environment". While when the obstacle is also moving, it is considered to be "dynamic environment". Furthermore, in dynamic environment, the obstacle can move with predictable velocity or with random velocity. The trajectory planning problem is based upon the optimization of some parameter or some objective function.

The common optimal criteria are minimum execution time, minimum energy or minimum jerk (He et al., 2009).

The gait of the robot is determined to be static or dynamic by the number of sequences involved in its stability. For this study, the environment is assumed to be dynamic. The importance of the step length is similar to the slope angle. A steeper slope provides more energy input and thus the resultant gait is faster, an effect similar to decreasing the step length (Wisse et al., 2005). The walking gait contains four parameters:

1. Initial – the robot is stationary and has not begun any movement
2. The first stride
3. Maximum height
4. Gait – the robot repeats the motions

The optimum traveling time for the robot corresponds to the minimization of a set of time intervals. This technique leads to the maximization of the operation speed. The number of robot joints must be considered simultaneously (Saramago et al., 1998).

Although the robot has the ability to adjust its foot step, there are two possibilities in which the robot may crash with the obstacle. One occurs when the step length is not correctly adapted according to the position of the dynamic obstacle. In this case, the swing leg touches directly the obstacle during a double support phase. The other case corresponds to the situation when the obstacle collides with the stance leg during the single or double support phase.

This section aim is to design a control strategy allowing biped robot to adjust automatically the step length in order to make the robot avoid dynamical obstacle using step over strategy.

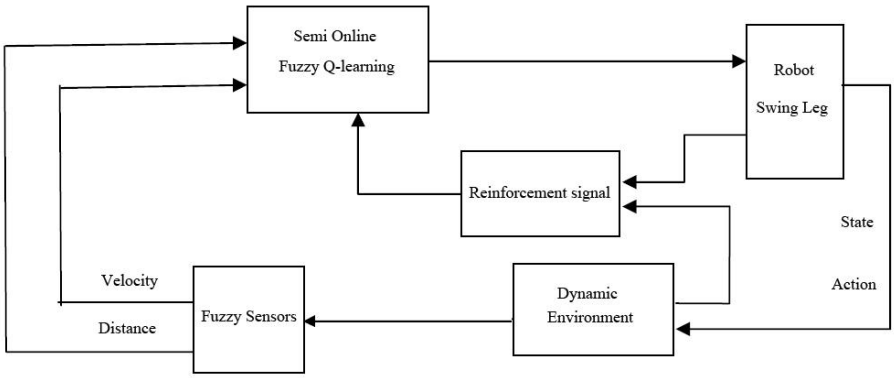
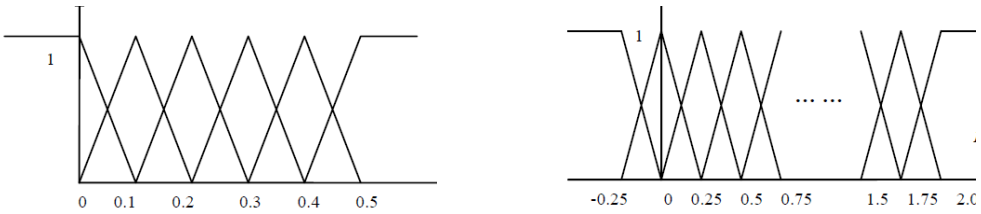


Figure 5.9: Strategy of foot step planning

The step-length adjusting system based on semi online fuzzy Q-learning (SOFQL) approach divided into four parts (figure 5.9), Dynamic environment, fuzzy sensors, concern the SOFQL algorithm to compute the length of the step and finally the reinforcement signal. In dynamic environment both of the robot and obstacle move in sagittal plan but in opposite directions. We considered the walking of the biped robot like a succession of both single and instantaneous double support phases. The biped robot may adjust the length of its step but we consider that the duration of the step is always equal to one second.

5.6.1 Fuzzy sensors

Each input needs the fuzzification. In our approach, we used two inputs to perform a correct footstep planning. These inputs are the distance between the robot and the obstacle d_o and the velocity of the obstacle v_o. Thus, distance between front foot and the edge of obstacle, is corresponding as d_o and calculated by the distance that is cover by 1second of time. d_o and v_o are updated at each double support phase. The fuzzification carried out by using 6 triangular membership function for d_o and 6 triangular membership function for v_o. Figure 5.10 shows the membership function of the obstacle velocity and distance respectively.



(a) Membership function for obstacle velocity v_o **(b)** Membership function for distance d_o

Figure 5.10: Membership function for obstacle velocity and distance

5.7 Step length based on Semi On-line Fuzzy Q-learning

This algorithm uses a set of fuzzy rule that are 66 (membership function, 6 for v_o and 11 for d_o) and for each role possible outputs can be define as [0.1, 0.2,0.3, 0.4, 0.5]m. The velocity of obstacle is chosen randomly. Actually this outputs are corresponds to the length of step, means the algorithm must choose one output at each step time among these outputs for each activated role. During one episode, the step length of the robot is computed using the SOFQL algorithm described in section 5.2. Consequently, the biped robot moves step by step towards the obstacle during the episode. The episode is finished if the robot steps over the obstacle or if the robot crashes into the obstacle. In order that the agent converges towards an optimal solution, the number of episode must be sufficient. The discount factor and learning rate parameter are equal to 0.8 and 0.1 respectively. These parameters have been chosen empirically after several trials in order to assure a good convergence of SOFQL algorithm. The probability P_ε is equal to 0.1, which means that the random exploration is privileged during the learning phase.

5.7.1 Reinforcement Signal

In our case, the learning agent must find a succession of action allowing the biped robot to step over an obstacle. But here the obstacle is a dynamic object which moves towards

the biped robot. Thus, the reinforcement information has to take into account the velocity of the moving obstacle. The position of the foot just before the stepping over is very important too. Based on these considerations, we design reinforcement into two parts.

Firstly, if $x_r < x_o$, where x_r is the position of robot and x_o gives position of the obstacle:

- r = 0, if robot is still far from obstacle
- r = 1, if robot position is appropriate to step over obstacle at the next step
- r = −1, if robot is too close to the obstacle

In this step we can find r as the followed equation:

$$r = \begin{cases} 0, & if(x_r \leq (x_r - 1.2v_o\Delta t)) \\ 1, & if\big(x_r > (x_r - 1.2v_o\Delta t)\big) \\ & and\ if\big(x_r \leq (x_r - 1.1v_o\Delta t)\big) \\ -1, & if\big(x_r > (x_r - 1.1v_o\Delta t)\big) \end{cases} \quad (5.14)$$

Where, $v_o\Delta t$ presents the distance covering by obstacle during the time Δt . Δt is always equal to*1s*. x_r and x_o are updated after each action.

Secondly, if $x_r \geq x_o$:

- r = −2, if robot crashes with the obstacle in next time,
- r = 2, if robot steps over the obstacle in next time.

In this last case, r is given by equation (5.15):

$$r = \begin{cases} -2, & if\big(x_r > (x_r - L_o)\big) \\ & and\ \big(x_o \geq (x_r + L_o)\big) \\ 2, & else \end{cases} \quad (5.15)$$

In which, L_o is represent the size of the obstacle.

5.7.2 Step duration time planning

The step duration time is defined as the durable time of one walking period and it can calculated as the time from leaving to landing on the ground of swing leg. Actually to increase the success rate, the step length and step duration time planned at the same time. The learning system must choose one action pair between the number of possible action pair (N_L) which this action defined as (L_S^j , T_S^j). L_S^j defined the step length and T_S^j defined the step duration time in one walking period. We choose pseudo-exhaustive method as the searching policy, the action pair with best evaluation value has a probability $P(a/x)$ to be chosen. The output of the Fuzzy system is to be the local action pair of the activated rule. After implement the next action pair, the Q matrix value $Q(x, L_s, T_s)$ at present time is calculated according to the present state and reinforcement signal. After the application the new action pair the TDE (Temporal Difference Error) is computed as below:

$$\Delta Q = \beta[r + \gamma V_{max}(s') - Q(S, A(S))] \quad (5.16)$$

This equation described in section 5.2.

The information of step length is very important to design the reinforcement signal, because the impact of next step is necessary besides the step duration time. For example if there is no enough time for avoiding in next step or any crash happening into the obstacle. So, the reinforcement signal is designed as below:

A) If $x'_r < x'_o$, it means the chosen present action pair lead the robot faraway from obstacle in the next step,

- if $x'_r \leq (x'_o - n.T_s.v_o)$, $r = 0$, the robot still far away from moving obstacle in the next step.
- if $x'_r > (x'_o - n.T_s.v_o)\ nad\ x'_r \leq (x'_o - m.T_s.v_o)$, the robot can step over the obstacle which means the present state is suitable

- if $x'_r > (x'_o - m.T_s.v_o)$, $r = -1$, the robot is very close to the obstacle and it will leads to crash in next step

B) If $x'_r \geq x'_o$, it means the robot is in the state of stepping over. The chosen action pair is determines if crash happens.

- if $x'_r > ((x_o + L_o))\ and\ x'_r > ((x_o - L_S))$, $r = 2$, the robot can step over the obstacle successfully.
- Otherwise, the robot crashes into obstacle.

In the above description, x_r represent the present position and the following position of the biped robot described by x'_r. x_o and x'_o are the corresponding position of obstacle to the above two state of biped robot. Both of m and n are constants, which have physical meaning of evaluating if the present action pair (L_S^j, T_S^j) and present position is appropriate for executing the next action pair in order to avoid crash. According to the above both step duration time and step length that are chosen in present time will influence on the present and future state.

5.7.3 Maximum step height planning

The vertical distance from highest point of the swing trajectory to the ground is as the maximum step height. So, in unknown environment the height of obstacle is not predictable, thus it is important to planning the step height for avoiding any crash. The robot can step over obstacle only with adjusting step length and step duration time. During the action of stepping over, when the swing leg of biped robot swings in the air, meanwhile the obstacle moves into the interval area of two legs. As the trajectory of the swing leg is usually an arch, one has to ensure that at the same time, every point in this arch is not superposition with the obstacle. As a result, adjusting the maximum step height according to the obstacle height is necessary for biped robot to avoid collision (Kim, Kim, & Lee, 2011). The robot only needs modify the step height for the last step

and maintenance the same maximum step height before stepping over. In order to make the stepping over safer, learning of the step height is usually done in advance.

In dynamic environment, the obstacle keeps on walking during swing leg uplift period. There are mainly two cases including: At the present time, the abscissa of the trajectory point equals to any point of the obstacle abscissa, the step height has to be modified. While there is difference between these two abscissas, the swing leg can continue the previous designed trajectory.

The trajectory is specified by the starting and landing points and step height of swing leg. The maximum step height is developed based on semi online fuzzy Q-learning which can be describe as follow: L_o defined the step length, distance between robot and obstacle is d_o and the input of fuzzy sensors is v_o which is the obstacle velocity.

The fuzzification of L_o is carried out by using 11 triangular membership functions. The fuzzy sensors for distance and obstacle velocity defined in previous section.

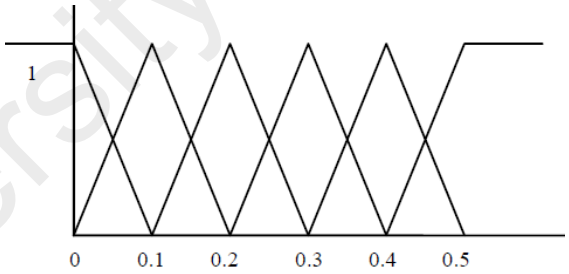


Figure 5.11: membership function of step length

Step length and obstacle velocity are updated before the action of stepping over. There is no need for the biped robot to adjust the step height (h_m) every step, it can holding the same height before stepping over. Therefore, the reinforcement signal r_h doesn't have to consider the influence of present action to the future state, but only take into database the impact of chosen action h_m to present state. The reinforcement signal describe as below:

If the abscissa of the trajectory point $P_x(t)$ superposition to any point of the obstacle abscissa $x_o(t)$,

· $r_h = 1$, if vertical coordinate of this trajectory point $P_y(t)$ is bigger than the height of obstacle h_o ,

· $r_h = -1$, if vertical coordinate of this trajectory point $P_y(t)$ is smaller than the height of obstacle h_o , the collision will be occurs,

if there is difference between the abscissa of the trajectory point $P_x(t)$ and the obstacle abscissa $x_o(t)$ at each time. There is no need for the swing leg to modify the step height in this case, because the swing leg will never collide with the obstacle.

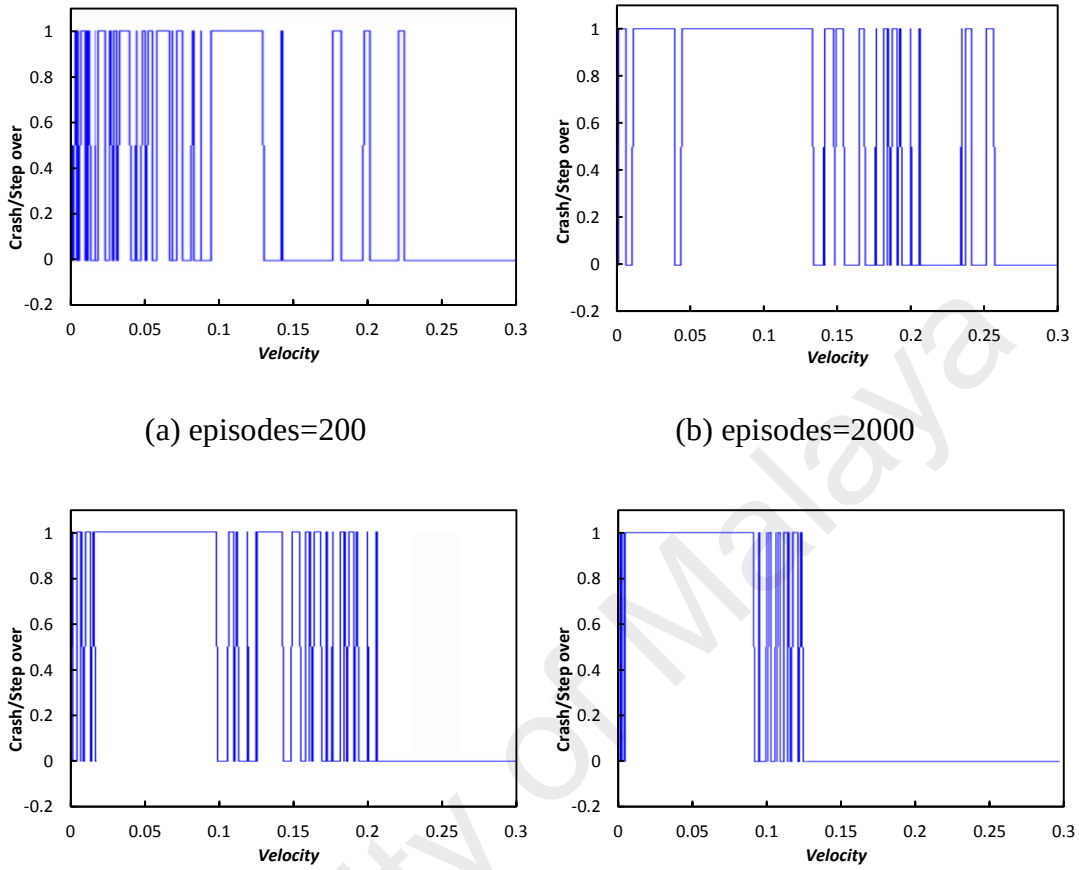
· $r_h = 0$

5.8 Simulation and results

In this sub section, the simulation and results presented link to foot step planning which consists of step length, duration time and maximum height planning. This simulation consist the convergence of proposed algorithm according the number of episodes during the learning phase as well as the example of the foot step planning. It must be noticed that our goal is to design a control strategy allowing giving a path planning in dynamical environment for biped robot, but we do not take into account the dynamic of the biped robot. We consider only discrete information allowing to compute the landing position of the foot. In addition, the simulation considers only flat obstacles.

a) The target of learning agent during learning phase is to find the best roles in order to make the biped robot step over an obstacle. The Q matrix is trained for different number of episodes. The footstep planning approaches tested for 1000

velocity sample covering uniformly the input range (0, 0.3) *m/s*, and in this simulation the size of obstacle is equal 0.2.



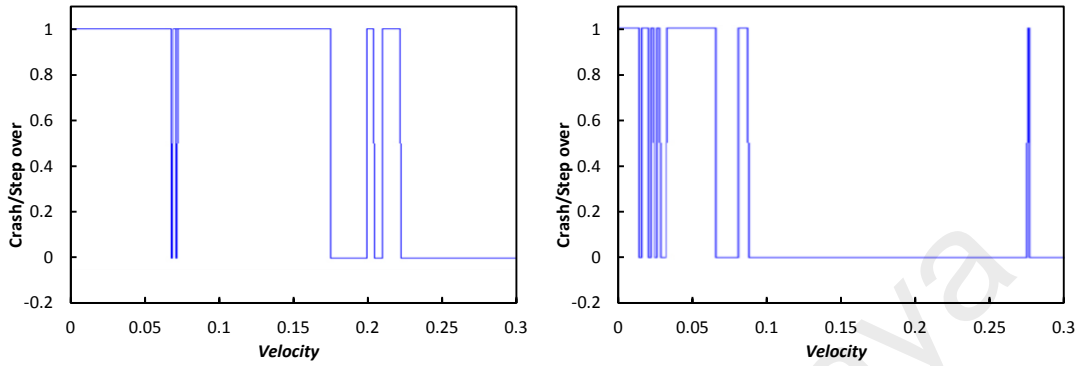
(a) episodes=200 (b) episodes=2000

(c) episodes=5000 (d) episodes=10000

Figure 5.12: Success Rate according to number of episodes

When the robot successfully steps over the obstacle the result is equal 1 and if any crashes occur the result will be 0. The relation between the number of successes and the total of trials is defined as success rate. The above figure showed the success rate in figure 5.12 (a) is equal 27.1%, figure 5.12 (b) is equal 41.8%, figure 5.12 (c) is equal 49.2% and figure 5.12 (d) is equal 31%. It must be pointed out, more learning agent needs for converging when the number of episode increases. This output figures showed whether the velocity of the obstacle is greater than the threshold approximately around 1.1m, no solution exists.

b) In this step, we analysis the success rate when the size of obstacle (L_o) is changed the results shows for obstacle size equals to 0.15 and 0.35m.



(a) obstacle size=0.15m (b) obstacle size=0.35m

Figure 5.13: Success Rate according to the size of obstacle

The above results showed for obstacle size (L_o) 0.15m the success rate is 67.4% and for $L_o = 0.35m$ the success rate is 25.7%. The success rate will be weak if the size of obstacle is large. So, after the learning phase we know the limitation of the proposed algorithm.

c) One of most important features of our model is good operational when the moving is unpredictable. Figure 5.14 showed the foot step sequence when the obstacle moves with the sum of a constant value that equals to 0.12 *m/s* and random velocity value included in [0…0.35] *m/s*. the size of obstacle is equal to 0.13m. The proposed model tested for around 1000 episodes.Black rectangle indicates the obstacle. Blue and red spots indicate the two positions of the right and left foot for each step.

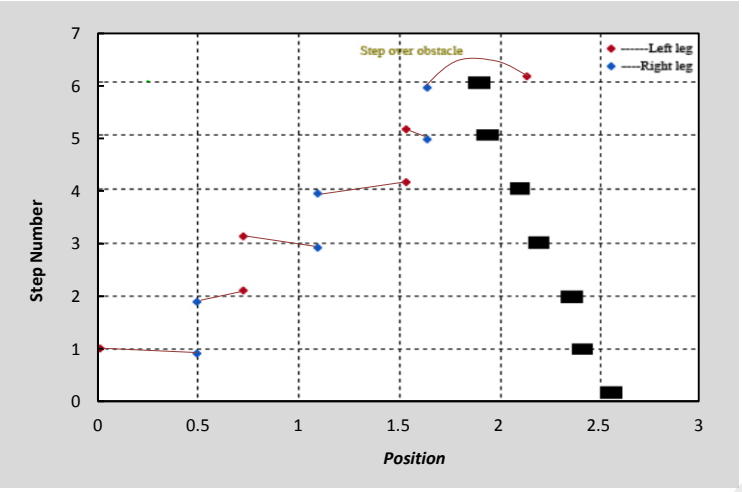


Figure 5.14: Footstep planning with random velocity

Figure 5.14 showed the step length deserves to point out that when the biped robot is close to the obstacle, then the length of the step decrease in order to prepare the stepping over. Finally, the last step allows avoiding obstacle without collision. Approximately the success rate in the same situation is equal to 81%. The control strategy of this algorithm allows the length of the step adapt automatically corresponding the obstacle velocity, this point cause increase the robustness.

D) As the output defuzzificated by the interface systems, the practical output $[L_s, T_s, H_m]$ is a group of real number. The obstacle velocity is change randomly. The step length is [0.15 0.25 0.35 0.45 0.55]m. On the assumption that the original maximum step height h_m is 0.06m, the robot keeps the same step height before stepping over. h_m can be chosen within [0.06 0.08 0.1 0.125 0.15 0.175]m. The presumable step duration time is [0.1 0.2 0.3 0.4 0.5]s. The following table shows the maximum height, length step position and time step for random velocity.

Table 5.1: Result of Maximum height, time step and length step for 6 sample step, when obstacle move with random velocity

Number of steps	1	2	3	4	5	6
Max Height	0.06	0.06	0.06	0.06	0.06	0.15
Time step	0.25	0.23	0.4	0.35	0.23	0.23
Length step	0.45	0.47	0.475	0.48	0.37	0.485

From the above results it can be seen, the last step allows the biped robot to avoid random velocity moving obstacle without collision. The output step length, step duration time and maximum step height are real numbers. It has been emphasized that the robot modifies the step length and step duration time action pair (L_S^j, T_S^j) for six steps, but the maximum step height is only adjusted from 0.06m to 0.15m in the last step. The Q matrix corresponding to the step height has only to been trained for the last step.

The above results are on the assumption of the initial distance (d_o) between the robot and obstacle equals to 2.0m. This initial distance determines the starting time for the biped robot training the Q matrix. In fact, when the obstacle is moving with constant velocity, the robot can modify the footstep fewer steps in advance. If the initial distance d_o is set to be 1.0m, the biped robot can avoid collision only by correcting 3 steps (refer to table 5.2).

Table 5.2: Result of Maximum height, time step and length step for 3 sample step, when obstacle move with constant velocity

Number of steps	1	2	3
Max Height	0.06	0.06	0.06
Time step	0.28	0.12	0.22
Length step	0.53	0.38	0.55

The above result showed, when the initial distance between robot and obstacle decrease, less steps of modification is enough for stepping over the obstacle successfully.

5.9 Summary

We have presented a footstep planning approach, allowing the biped robot to step over dynamic obstacles by adjusting the step length, step duration time and maximum step height. Our footstep planning strategy is based on a Semi On-line Fuzzy Q-Learning concept. The step length and step duration times are considered to be an action pair. The learning system needs to choose one action pair in all the possible action pairs for each activated rule. If the number of episodes is big enough during the learning phase, the Q matrix can be trained completely and the algorithm converges towards an optimal compromise. In addition, the study of the result gives information about the maximum obstacle velocity according to the size of the obstacle. The proposed footstep planning is operational for both constant and unpredictable motion of the obstacle. The learning system makes use of the obstacle velocity and chosen action pair training Q matrix.

The investigations in this chapter show a real interest of this approach because: (i) the computing time is very short by using database in second phase of proposed algorithm. (ii) This footstep planning approach is valid for both static and dynamic obstacles. (iii) The footstep planning is operational for both predictable and unpredictable dynamical environment allowing the control to increase the robustness.

In this chapter also the Semi Online Fuzzy Q-learning algorithm proposed, which attempts to solve the learning problem with a learning samples database. The advantage of this model is the important reduction of the iterations required to converge, compared to the classic Q-learning algorithm. The algorithm is also able to learn in real time the

optimal state-action policy. The most important drawback in the robotics task was the correct observation of the state. Our proposed algorithm helps to solve the various crucial problems.

CHAPTER 6

Method of preventing obstacle for biped robot in unstructured environment

6.0 Method of Preventing Obstacle for Biped Robot in Unstructured Environment

6.1 Introduction

Biped walk learning for robots is a difficult process because many areas of the implementation are unclear to researchers (Mercili & Veloso, 2010).The strategy for control of robot movement can mainly divided in two categories: the first one is modeled on kinematic and dynamic of biped robot and the second one is based on computing technology, such as neural network, fuzzy logic, genetic algorithm and the learning machine. Besides the first strategy, these require knowing the internal specification of biped robot incisively, which requires measurement of the velocity, joint angels, acceleration and the evaluation contact pressure in between feet and ground. The control method based upon mathematical model normally needs large computation and it leads to complication in on-line control. But in second model no need to know the mechanical structure of robot. Besides, the off-line and on-line situation can be done in learning process (Sabourin, 2008).

For the design of control strategy permitting the robot to move in dynamical environment, noted in this strategy some complicated activity will occur. It includes solving the number of problems such as dynamic stability control or desired joint trajectory tracking that is called low-level control and furthermore the path or gait planning which is called high-level control.

The low-level control taking advantage of feedback format depends on proprioceptive information or reactive control and the high-level control uses a predictive technique (planning). Many researchers involve the control of autonomous robot (Hackel, 2007), but only a few studies concern the high-level control working

with path planning and the obstacle avoidances. In this chapter, the Semi On-line Fuzzy Q-learning algorithm used to solve the problems of biped robot stepping over the obstacle that is moving with unselected velocity. After the learning process maximum height of every step, step length and duration time are recorded. Desired foot trajectory of swing leg could be achieved after interpolating the starting and ending point and maximum step height.

The joint angle profiles are able to be calculated depending on the relationship between pitch angels by using inverse kinematics. The footstep planning and foot trajectory with inverse kinematic for generating the joint angle profiles establish the high-level control. Joint trajectory and control the tracking of desired trajectories involve the low-level control. It may be broken down to two parts: a hierarchically clustered fuzzy cerebellar model articulation controller (HCFCMAC) for approximating the joint profile for swing leg and customization of the pitch angle of the trunk.

This chapter contains: introduction to the low and high level control design strategy, development of trajectory for swing leg and joint angels, whereby robot simulate and test the proposed high-level control strategy, introduce the low-level control containing description of HCFCMAC for estimating the joint profile and pitch angle customization and the last section consists of the simulation results.

6.2 Five-link Biped Robot Motion Equations

The mathematical modeling for movement of humanoid robot is always a complicated issue, because of the many degrees of freedom in linkage system. Typically, mathematical modeling of biped robot includes two stages: first, dynamics modeling and the second one, kinematic modeling. Identify and analyses the action

relationship between each part of body is the goal of kinematic modeling. The dynamic behaviour of a robot locomotion system is detailed in term of the time rate of modifying of the linkage configuration in relation to the joint torque. Besides, dynamic of biped robot motion can be categorized in two research classes: forward dynamic and inverse dynamic problems.

The first mathematical model of biped robot was introduced by Chow and Jacobson (1972) as an inverted pendulum. This pendulum was used for control of the postural stability of upper body for robot motion. Hemami and his colleague (1977,1980) improved this system into a massive inverted pendulum with the base joint fixed, to be able to study the behaviour of body in standing position. These models are developed and researcher can find mathematical model for five and seven link robots. Tzafestas,S. et al. (1996) applied five link biped robot model to study the forward walking motion in sagittal environment. O. Haavisto (2004) developed seven degree of freedom dynamic model to explain the dynamic of system in all position. The joint angles are described by five coordinations and the position of the centre of mass defined by the rest two coordinations. We will study the motion of biped robot in unstructured plan.

6.2.1 The kinematics Model of the Five-link Biped Robot

Biped robot locomotion system is a very complex dynamic system from the aspect of control system complexity and mechanical structure complexity. To study this system we choose the five-linked biped structure for illustrating our control technique. In this study we only consider the movement on sagittal plan. The particular sagittal plane is defined by the vertical axis and the direction on motion. The trunk, head and arm which are known as upper body of biped robot is considered as a rigid inverted pendulum. The upper part is coupled to the legs with a pair of rotational joints. Every leg is made up of

two rigid links. The rotational joint is connected to all links together. The ground condition is assumed non-slip. At any time quick only one foot has a position contact with the land.

One of the best advantages of this model is that it has sufficiently degree of freedom to keep equations of movement into a manageable level, even thought still having adequate degrees of freedom to describe the walking locomotion which includes the effect between the end of swing leg and walking surface (Tzafestas, 1996). A three segment kinematic chain with pinot joint is shown in figure 6.1. Figure 6.2 shows the structure of 5-linked robot and muscle diagram.

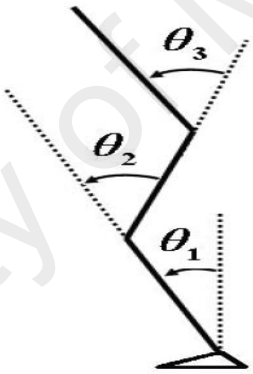


Figure 6.1: Body segment parameters and body configuration angle conventions:

$\theta_{ankle} = \theta_1$, $\theta_{knee} = \theta_2$, $\theta_{hip} = \theta_3$.

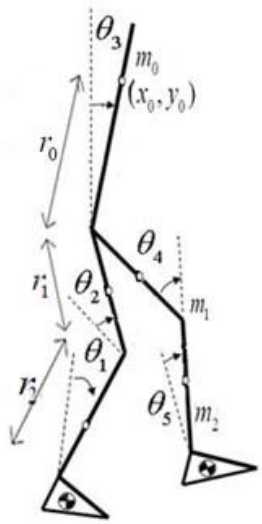


Figure 6.2: The 5-linked biped robot model

Refering to figure 6.2, the mass centre of trunk is (x_0, y_0). $\boldsymbol{\theta_i}$ (i=1…5) corresponds to angle of link ***i*** with dependence upon the vertical direction. $\boldsymbol{m_0}$ is the mass of torso. $\boldsymbol{r_0}$ represents the distance from the mass centre to the hip joint. $\boldsymbol{r_1}$ is the distance from (x_0 , y_0) to the hip joint of two thighs and the $\boldsymbol{r_2}$ refers to the distance from ankle to the mass centre. Most likely, the length of two thighs is $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ is the length of the shank, respectively, $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ are their masses. According to tree structure each link turns into a node and each joint becomes an edge of tree (Kyong-Sok Chang, 2000). In this topology each branch has a serial number: the shank is known as link A, and the thigh is called link B. Link C is the upper body and the thigh and shank of stance leg respective link D and E.

Based on the kinematic relationship between links, (x_a, y_a) is the position of the stance feet, and the end of swing leg is (x_e, y_e) which is free. These positions could be derived from equation (6.1) and (6.2).

$$x_a = x_0 - r_0 . \sin\theta_3 - l_1 . \sin\theta_2 - l_2 \sin\theta_1$$
$$y_a = y_0 - r_0 . \cos\theta_3 - l_1 . \cos\theta_2 - l_2 \cos\theta_1 \quad (6.1)$$

$$x_e = x_0 - r_0 . \sin\theta_3 + l_1 . \sin\theta_4 + l_2 \sin\theta_5$$
$$y_e = y_0 - r_0 . \cos\theta_3 + l_1 . \cos\theta_4 + l_2 \cos\theta_5 \quad (6.2)$$

Now we can make the velocity of end of swing leg **V**:

$$V = \begin{bmatrix} v_{ex} \\ v_{ey} \end{bmatrix} = \begin{bmatrix} -r_0 \cos\theta_3 \\ r_0 \sin\theta_3 \end{bmatrix} . \dot{\theta}_3 + \begin{bmatrix} l_1 \cos\theta_4 \\ l_1 \sin\theta_4 \end{bmatrix} . \dot{\theta}_4 + \begin{bmatrix} l_2 \cos\theta_5 \\ l_2 \sin\theta_5 \end{bmatrix} . \dot{\theta}_5 \quad (6.3)$$

The mass centre of each link ***i*** can be represented as follow:

Link A:

$$x_A = x_0 - r_0 . \sin\theta_3 - l_1 . \sin\theta_2 - r_2 \sin\theta_1$$
$$y_A = y_0 - r_0 . \cos\theta_3 - l_1 . \cos\theta_2 - r_2 \cos\theta_1 \quad (6.4)$$

Link B:

$$x_B = x_0 - r_0 . \sin\theta_3 - r_1 . \sin\theta_2$$
$$y_B = y_0 - r_0 . \cos\theta_3 - r_1 . \cos\theta_2 \quad (6.5)$$

Link C:

$$x_C = x_0$$
$$y_C = y_0 \quad (6.6)$$

Link D:

$$x_D = x_0 - r_0 . \sin\theta_3 + r_1 . \sin\theta_4 \quad (6.7)$$

$$y_D = y_0 - r_0 . \cos\theta_3 + r_1 . \cos\theta_4$$

Link E:

$$x_E = x_0 - r_0 . \sin\theta_3 + l_1 . \sin\theta_4 + r_2 \sin\theta_5$$
$$y_E = y_0 - r_0 . \cos\theta_3 - l_1 . \cos\theta_4 - r_2 \cos\theta_5 \quad (6.8)$$

The coordinate of mass centre of biped robot model found by using (6.4) ~ (6.8) to the following equation:

$$cm_x = \frac{m_0 x_C + m_1 x_B + m_1 x_D + m_2 x_A + m_2 x_E}{m_0 + 2m_1 + 2m_2} \quad (6.9)$$

$$cm_y = \frac{m_0 y_C + m_1 y_B + m_1 y_D + m_2 y_A + m_2 y_E}{m_0 + 2m_1 + 2m_2}$$

The velocity of mass centre for each link is derived by using the equation (6.4) ~ (6.8):

$$V_A = \begin{bmatrix} v_{Ax} \\ v_{Ay} \end{bmatrix} = \begin{bmatrix} -r_0 \cos\theta_3 \\ r_0 \sin\theta_3 \end{bmatrix} . \dot{\theta}_3 + \begin{bmatrix} -l_1 \cos\theta_2 \\ l_1 \sin\theta_2 \end{bmatrix} . \dot{\theta}_2 + \begin{bmatrix} -r_2 \cos\theta_1 \\ r_2 \sin\theta_1 \end{bmatrix} . \dot{\theta}_1 \quad (6.10)$$

$$V_B = \begin{bmatrix} v_{Bx} \\ v_{By} \end{bmatrix} = \begin{bmatrix} -r_0 \cos\theta_3 \\ r_0 \sin\theta_3 \end{bmatrix} . \dot{\theta}_3 + \begin{bmatrix} -r_1 \cos\theta_2 \\ r_1 \sin\theta_2 \end{bmatrix} . \dot{\theta}_2 \quad (6.11)$$

$$V_C = \begin{bmatrix} v_{Cx} \\ v_{Cy} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6.12)$$

$$V_D = \begin{bmatrix} v_{Dx} \\ v_{Dy} \end{bmatrix} = \begin{bmatrix} -r_0 \cos\theta_3 \\ r_0 \sin\theta_3 \end{bmatrix} . \dot{\theta}_3 + \begin{bmatrix} r_1 \cos\theta_4 \\ r_1 \sin\theta_4 \end{bmatrix} . \dot{\theta}_4 \quad (6.13)$$

$$V_E = \begin{bmatrix} v_{Ex} \\ v_{Ey} \end{bmatrix} = \begin{bmatrix} -r_0 \cos\theta_3 \\ r_0 \sin\theta_3 \end{bmatrix} . \dot{\theta}_3 + \begin{bmatrix} l_1 \cos\theta_4 \\ l_1 \sin\theta_4 \end{bmatrix} . \dot{\theta}_4 + \begin{bmatrix} r_2 \cos\theta_5 \\ r_2 \sin\theta_5 \end{bmatrix} . \dot{\theta}_5 \qquad (6.14)$$

6.2.2 The Dynamic Model of 5-link Biped Motion Robot

The dynamic of robotic mechanisms has two major formulations: dynamic formulation for the joint space (Walker, 1982) and the operation space (Oussarna Khatib, 1987). The dynamic of joints is identified by joint space formulation. For the study of biped robot motion in our problem, we use the joint space dynamic formulation.

By applying the Lagrangian formulation the forward dynamic model of biped robot movement can be developed. The advantage of this formulation is that only the kinetic and potential energies of the system are required and all the actual workless forces are automatically removed.

6.2.2.1 Dynamic Model of Biped Robot

The motion of biped robot in sagittal plan is a smooth forward motion. In this phase the supporting leg is carrying all the weight of body and the swing leg swinging in the air in front direction. The contacting point between ground and the assistant foot is (x_a , y_a), and is valid during this step:

$$\dot{x}_a = \dot{y}_a = 0 \qquad (6.15)$$

As the system contains 5 links, it has five degrees of freedom. The complimenting five coordinates are chosen according to figure 6.1:

$$\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T \tag{6.16}$$

Also the model has five movements:

$$T = [T_{\theta_1}, T_{\theta_2}, T_{\theta_3}, T_{\theta_4}, T_{\theta_5}]^T \tag{6.17}$$

The Lagrangian formulation of the 5-linked system is provided by diversity between kinetic and possibility energies (Chung, 2000):

$$L = K - P \tag{6.18}$$

L is equal Lagrange multiplier, ***K*** is the kinetic energy that is the sum of all five links and ***P*** is the entire potential energies which include each link:

$$K = \sum_{i=1}^{5} k_i \tag{6.19}$$

$$P = \sum_{i=1}^{5} P_i \tag{6.20}$$

The motion equation is in the form as:

$$\boldsymbol{T_i} = \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} \tag{6.21}$$

Where, ***T*** is the sum of torques during turning motion, according to equations (6.19) ~ (6.21), the Lagrangian equation can be rearranged in the form:

$$T_i = \frac{\partial\left(\frac{\partial\left(\sum_{i=1}^{5} K_i - \sum_{i=1}^{5} P_i\right)}{\partial \dot{\theta}_i}\right)}{\theta t} - \frac{\partial\left(\sum_{i=1}^{5} K_i - \sum_{i=1}^{5} P_i\right)}{\partial \theta_i} \tag{6.22}$$

The kinetic energy consists of two parts: the rectilinear motion and rotary motion around the centre mass. The potential energy and kinetic energy for each link are shown as below equations:

$$K_i = \frac{1}{2} m_i \left(v_{ix}^2 + v_{iy}^2\right) + \frac{1}{2} I_i \dot{\theta}_i^2 \qquad i = 1,2 \ldots 5 \tag{6.23}$$

$$P_i = m_i \mathrm{g}. y_i \tag{6.24}$$

Here, $\boldsymbol{g}$ is the gravitational acceleration. Therefore, the kinetic and potential energy for each link are calculated as follows:

Link A:

$$K_A = \frac{1}{2}(m_2 r_2^2 + I_A).\dot{\theta}_1^2 + \frac{1}{2} m_2 I_1^2.\dot{\theta}_2^2 + \frac{1}{2} m_2 r_0^2.\dot{\theta}_3^2 + m_2[r_2 l_1 \cos(\theta_1 - \theta_2).\dot{\theta}_1\dot{\theta}_2 + r_0 l_1 \cos(\theta_2 - \theta_3).\dot{\theta}_2\dot{\theta}_3 + r_0 r_2 \cos(\theta_1 - \theta_3).\dot{\theta}_1\dot{\theta}_3] \tag{6.25}$$

$$P_A = m_2 \mathrm{g}(y_0 - r_0 \cos\theta_3 - l_1 \cos\theta_2 - r_2 \cos\theta_1) \tag{6.26}$$

Link B:

$$K_B = \frac{1}{2}(m_1 r_1^2 + I_B).\dot{\theta}_2^2 + \frac{1}{2} m_1 r_0^2.\dot{\theta}_3^2 + m_1\big[r_0 r_1 cos(\theta_2 - \theta_3).\dot{\theta}_2\dot{\theta}_3 \tag{6.27}$$

$$P_B = m_1 \mathrm{g}(y_0 - r_0 \cos\theta_3 - r_1 \cos\theta_2) \tag{6.28}$$

Link C:

$$K_C = \frac{1}{2} I_C \dot{\theta}_3^2 \tag{6.29}$$

$$P_C = m_0 \mathrm{g}\ .\ y_0 \tag{6.30}$$

Link D:

$$K_D = \frac{1}{2}(m_1 r_1^2 + I_D).\dot{\theta}_4^2 + \frac{1}{2}m_1 r_0^2.\dot{\theta}_3^2 - m_1[r_0 r_1 . \cos(\theta_3 - \theta_4) . \dot{\theta}_3 \dot{\theta}_4 \quad (6.31)$$

$$P_D = m_1 \mathrm{g}.(y_0 - r_0 . \cos\theta_3 - r_1 . \cos\theta_4) \quad (6.32)$$

Link E:

$$K_E = \frac{1}{2}(m_2 r_2^2 + I_E).\dot{\theta}_5^2 + \frac{1}{2}m_2 r_0^2.\dot{\theta}_3^2 + \frac{1}{2}m_2 l_1^2.\dot{\theta}_4^2 + m_2[r_2 l_1 \cos(\theta_4 - \theta_5) . \dot{\theta}_4 \dot{\theta}_5$$
$$- r_0 l_1 \cos(\theta_3 + \theta_4) . \dot{\theta}_4 \dot{\theta}_3 - r_0 r_2 \cos(\theta_3 + \theta_5) . \dot{\theta}_3 \dot{\theta}_5] \quad (6.33)$$

$$P_E = m_2 \mathrm{g} .(y_0 - r_0 \cos\theta_3 - l_1 \cos\theta_4 - r_2 \cos\theta_5) \quad (6.34)$$

The following standard form of the dynamic equation of biped robot in dynamic environment found by replacing equation (6.25) ~ (6.34) to equation (6.22),

$$\mathbf{A}(\boldsymbol{\theta})\ddot{\theta} + \mathbf{B}(\boldsymbol{\theta})\dot{\theta}^2 + \mathbf{C}(\boldsymbol{\theta}) = \mathbf{T} \quad (6.35)$$

As we simulate the five-link robot motion and it consists five degrees of freedom there must exist five different equations, **T** is the actuated moments due to equation (6.17). $\mathbf{A}(\boldsymbol{\theta})$ is the internal matrix and each term is formulated as follow:

$$a_{11} = m_2 r_2^2 + I_A$$

$$a_{12} = m_2 r_2 l_1 \cos(\theta_1 - \theta_2)$$

$$a_{13} = m_2 r_0 r_1 \cos(\theta_1 - \theta_3) \quad (6.36)$$

$$a_{14} = 0$$

$$a_{15} = 0$$

$$a_{21} = m_2 r_2 l_1 \cos(\theta_1 - \theta_2)$$

$$a_{22} = m_2 l_1^2 + m_1 r_1^2 + I_B$$

$$a_{23} = (m_2 r_0 l_1 + m_1 r_0 r_1)\cos(\theta_2 - \theta_3) \tag{6.37}$$

$$a_{24} = 0$$

$$a_{25} = 0$$

$$a_{31} = m_2 r_2 r_0 \cos(\theta_1 - \theta_3)$$

$$a_{32} = (m_2 r_0 l_1 + m_1 r_0 r_1)\cos(\theta_2 - \theta_3)$$

$$a_{33} = 2m_2 r_0^2 + 2m_1 r_0^2 + I_C \tag{6.38}$$

$$a_{34} = -m_1 r_0 r_1 \cos(\theta_3 - \theta_4) - m_2 l_1 r_0 \cos(\theta_3 + \theta_4)$$

$$a_{35} = -m_2 r_2 r_0 \cos(\theta_3 + \theta_5)$$

$$a_{41} = 0$$

$$a_{42} = 0$$

$$a_{43} = -m_1 r_0 r_1 \cos(\theta_3 - \theta_4) - m_2 l_1 r_0 \cos(\theta_3 + \theta_4) \tag{6.39}$$

$$a_{44} = m_2 r_1^2 + m_2 l_1^2 + I_D$$

$$a_{45} = m_2 r_2 l_1 \cos(\theta_4 - \theta_5)$$

$$
\begin{aligned}
a_{51} &= 0 \\
a_{52} &= 0 \\
a_{53} &= -m_2 r_2 r_0 \cos(\theta_3 + \theta_5) \\
a_{54} &= m_2 r_2 l_1 \cos(\theta_4 - \theta_5) \\
a_{55} &= m_2 r_2^2 + I_E
\end{aligned}
\tag{6.40}
$$

From the previous expression it can be seen, $\mathbf{A}(\boldsymbol{\theta})$ is a symmetric array. $\mathbf{B}(\boldsymbol{\theta}) \in \mathcal{R}^{5\times5}$ is a matrix which is the quadratic term of $\dot{\theta}$. So, the expression of each term is:

$$
\begin{aligned}
b_{11} &= 0 \\
b_{12} &= m_2 r_2 l_1 \sin(\theta_1 - \theta_2) \\
b_{13} &= m_2 r_2 r_0 \sin(\theta_1 - \theta_3) \\
b_{14} &= 0 \\
b_{15} &= 0
\end{aligned}
\tag{6.41}
$$

$$
\begin{aligned}
b_{21} &= -m_2 r_2 l_1 \sin(\theta_1 - \theta_2) \\
b_{22} &= 0 \\
b_{23} &= (m_2 r_0 l_1 + m_1 r_1 r_0) \sin(\theta_2 - \theta_3) \\
b_{24} &= 0 \\
b_{25} &= 0
\end{aligned}
\tag{6.42}
$$

$$b_{31} = -m_2 r_2 r_0 \sin(\theta_1 - \theta_3)$$

$$b_{32} = -(m_2 r_0 l_1 + m_1 r_1 r_0) \sin(\theta_2 - \theta_3)$$

$$b_{33} = 0 \tag{6.43}$$

$$b_{34} = m_2 r_0 l_1 \sin(\theta_3 + \theta_4) - m_1 r_0 r_1 \sin(\theta_3 - \theta_4)$$

$$b_{35} = m_2 r_0 r_2 \sin(\theta_3 + \theta_5)$$

$$b_{41} = 0$$

$$b_{42} = 0$$

$$b_{43} = m_2 r_0 l_1 \sin(\theta_3 + \theta_4) + m_1 r_0 r_1 \sin(\theta_3 - \theta_4) \tag{6.44}$$

$$b_{44} = 0$$

$$b_{45} = m_2 l_1 r_2 \sin(\theta_4 - \theta_5)$$

$$b_{51} = 0$$

$$b_{52} = 0$$

$$b_{53} = m_2 r_0 r_2 \sin(\theta_3 + \theta_5) \tag{6.45}$$

$$b_{54} = -m_2 l_1 r_2 \sin(\theta_4 - \theta_5)$$

$$b_{55} = 0$$

$\mathbf{C}(\boldsymbol{\theta}) \in \mathcal{R}^{5\times 1}$ is a vector, that is the expression of θ_i:

$$c_1 = m_2 g r_2 \sin\theta_1$$

$$c_2 = (m_2 g l_1 + m_1 g r_1) \sin\theta_2$$

$$c_3 = (2m_2 g r_0 + 2m_1 g r_0) \sin\theta_3 \qquad (6.46)$$

$$c_4 = (m_2 g l_1 + m_1 g r_1) \sin\theta_4$$

$$c_5 = m_2 g r_2 \sin\theta_5$$

6.2.2.2 Alteration of the Dynamic Model

The above derivative process utilizes θ_i the angles with respect to the vertical direction. Refering to figure 6.3, for the control of motion; we use the relative angle $[\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T$ between two connected links.

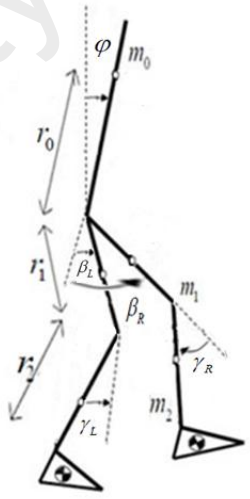


Figure 6.3: Biped model with relative angles

Instead of absolute angle θ_i of each link, the relative angle between links is used. The relationship between absolute and relative angles is as below:

$$\begin{cases} \gamma_L = \theta_1 - \theta_2 \\ \beta_L = \varphi - \theta_2 \\ \varphi = \theta_3 \\ \beta_R = \varphi + \theta_4 \\ \gamma_R = \theta_4 - \theta_5 \end{cases} \tag{6.47}$$

So, the variables θ_i can be expressed as:

$$\begin{cases} \theta_1 = \gamma_L + \varphi - \beta_L \\ \theta_2 = \varphi - \beta_L \\ \theta_3 = \varphi \\ \theta_4 = \beta_R - \varphi \\ \theta_5 = \beta_R - \varphi - \gamma_R \end{cases} \tag{6.48}$$

We assume the actual driving torques of the joints is $\mathbf{M} = [M_{L1}, M_{R1}, M_{\varphi}, M_{R2}, M_{L2}]^T$

The relative angles defined as follows refering to figure 6.4:

$$\mathbf{q} = [q_1, q_2, q_3, q_4, q_5]^T = [\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T \tag{6.49}$$

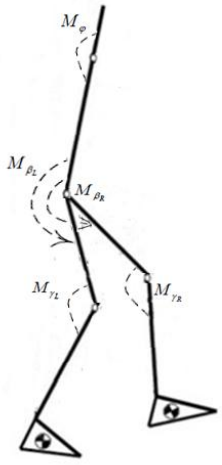


Figure 6.4: Biped model coordinates with applied torques

Thus, from the previous relation we have:

$$M_{q_j} = \sum_{i=1}^{5} T_{\theta_i} \frac{\partial_{\theta_i}}{\partial_{q_j}} \qquad j = (1,2,\dots,5) \tag{6.50}$$

Also we can find relationship between M_{q_j} and T_{θ_i}:

$$\begin{cases} M_{q_1} = T_{\theta_1} \\ M_{q_2} = -T_{\theta_1} - T_{\theta_2} \\ M_{q_3} = T_{\theta_1} + T_{\theta_2} + T_{\theta_3} - T_{\theta_4} - T_{\theta_5} \\ M_{q_4} = T_{\theta_4} + T_{\theta_5} \\ M_{q_5} = -T_{\theta_5} \end{cases} \tag{6.51}$$

The biped model transferred as below by using above equations:

$$A_{11}\ddot{\theta}_1 + A_{12}\ddot{\theta}_2 + A_{13}\ddot{\theta}_3 + A_{14}\ddot{\theta}_4 + A_{15}\ddot{\theta}_5 + h_{q1} + G_{q1} = M_{q1} \tag{6.52}$$

In which

$$\begin{aligned} A_{1j} &= a_{1j} \qquad j = 1,2,\dots,5 \\ h_{q1} &= b_1 \\ G_{q1} &= c_1 \end{aligned} \tag{6.53}$$

$$A_{21}\ddot{\theta}_1 + A_{22}\ddot{\theta}_2 + A_{23}\ddot{\theta}_3 + A_{24}\ddot{\theta}_4 + A_{25}\ddot{\theta}_5 + h_{q2} + G_{q2} = M_{q2} \tag{6.54}$$

Whereby

$$\begin{aligned} A_{2j} &= -a_{1j} - a_{2j} \qquad j = 1,2,\dots,5 \\ h_{q2} &= b_1 \\ G_{q2} &= c_1 \end{aligned} \tag{6.55}$$

$$A_{31}\ddot{\theta}_1 + A_{32}\ddot{\theta}_2 + A_{33}\ddot{\theta}_3 + A_{34}\ddot{\theta}_4 + A_{35}\ddot{\theta}_5 + h_{q3} + G_{q3} = M_{q3} \tag{6.56}$$

In which

$$\begin{aligned} A_{3j} &= a_{1j} + a_{2j} + a_{3j} - a_{4j} - a_{5j} \qquad j = 1,2,\dots,5 \\ h_{q3} &= b_1 + b_2 + b_3 - b_4 - b_5 \\ G_{q3} &= c_1 + c_2 + c_3 - c_4 - c_5 \end{aligned} \tag{6.57}$$

$$A_{41}\ddot{\theta}_1 + A_{42}\ddot{\theta}_2 + A_{43}\ddot{\theta}_3 + A_{44}\ddot{\theta}_4 + A_{45}\ddot{\theta}_5 + h_{q4} + G_{q4} = M_{q4} \tag{6.58}$$

Whereby

$$\begin{aligned} A_{4j} &= a_{4j} + a_{5j} \qquad j = 1,2,\dots,5 \\ h_{q4} &= b_4 + b_5 \\ G_{q4} &= c_4 + c_5 \end{aligned} \tag{6.59}$$

$$A_{51}\ddot{\theta}_1 + A_{52}\ddot{\theta}_2 + A_{53}\ddot{\theta}_3 + A_{54}\ddot{\theta}_4 + A_{55}\ddot{\theta}_5 + h_{q5} + G_{q5} = M_{q5} \tag{6.60}$$

In which

$$\begin{aligned} A_{5j} &= -a_{5j} \qquad j = 1,2,\dots,5 \\ h_{q5} &= -b_5 \\ G_{q5} &= -c_5 \end{aligned} \tag{6.61}$$

Again, using the same relations, the dynamic model of biped robot can be transformed for the control purpose as:

$$\mathbf{D_q}(q)\ddot{q} + \mathbf{h_q}(q,\dot{q})\dot{q}^2 + \mathbf{G_q}(q) = \mathbf{M_q} \tag{6.62}$$

Where

$$\begin{cases} D_q(i,1) = A_{i1} & i = 1,2,\dots,5 \\ D_q(i,2) = -A_{i1} - A_{i2} \\ D_q(i,3) = A_{i1} + A_{i2} + A_{i3} - A_{i4} - A_{i5} \\ D_q(i,4) = A_{i4} + A_{i5} \\ D_q(i,5) = -A_{i5} \end{cases} \tag{6.63}$$

$h_q\dot{q} = [h_{q_1}, h_{q_2}, h_{q_3}, h_{q_4}, h_{q_5}]^T$

$G_q = [G_{q_1}, G_{q_2}, G_{q_3}, G_{q_4}, G_{q_5}]^T$

$\mathbf{D_q}(q)$ is the 5x5 definite inertia matrix, $\mathbf{h_q}(q,\dot{q})$ is the 5x1 vector of centripetal torques, $\mathbf{M_q}$ is the 5×1vector of control torque placed at each joint. We can change each formulated term as follow by applying equation (6.53), (6.55), (6.57), (6.59) and (6.61) into equation (6.63):

For $\mathbf{D_q}(q)\ddot{q} = \mathbf{D_q}(i,j) \qquad (i = 1,2,\dots,5 \;\; j = 1,2,\dots,5)$

$$\begin{cases} D_q(1,1) = m_2 r_2^2 + I_A \\ D_q(1,2) = -(m_2 r_2^2 + I_A) - m_2 r_2 l_1 \cos(\theta_1 - \theta_2) \\ D_q(1,3) = m_2 r_2^2 + I_A + m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + m_2 r_2 r_0 \cos(\theta_1 - \theta_3) \\ D_q(1,4) = 0 \\ D_q(1,5) = 0 \end{cases} \tag{6.64}$$

$$\begin{cases} D_q(2,1) = -(m_2 r_2^2 + I_A) - m_2 r_2 l_1 \cos(\theta_1 - \theta_2) = D_q(1,2) \\ D_q(2,2) = 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + m_2 r_2^2 + I_A + m_2 l_1^2 + m_1 r_1^2 + I_B \\ D_q(2,3) = -(m_2 r_2^2 + I_A) - 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) - (m_2 l_1^2 + m_1 r_1^2 + I_B) \\ \qquad -m_2 r_2 r_0 \cos(\theta_1 - \theta_3) - (m_2 r_0 l_1 + m_1 r_0 r_1)\cos(\theta_2 - \theta_3) \\ D_q(2,4) = 0 \\ D_q(2,5) = 0 \end{cases} \tag{6.65}$$

$$\begin{cases} D_q(3,1) = m_2 r_2^2 + I_A + m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + m_2 r_2 r_0 \cos(\theta_1 - \theta_3) = D_q(1,3) \\ D_q(3,2) = -(m_2 r_2^2 + I_A) - 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) - (m_2 l_1^2 + m_1 r_1^2 + I_B) \\ \qquad -m_2 r_2 r_0 \cos(\theta_1 - \theta_3) - (m_2 r_0 l_1 + m_1 r_0 r_1)\cos(\theta_2 - \theta_3) = D_q(2,3) \\ D_q(3,3) = m_2 r_2^2 + I_A + 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + 2m_2 r_2 r_0 \cos(\theta_1 - \theta_3) + (m_2 l_1^2 + \\ \qquad m_1 r_1^2 + I_B) + 2\,(m_2 r_0 l_1 + m_1 r_0 r_1)\cos(\theta_2 - \theta_3) + (2m_2 r_0^2 + 2m_1 r_0^2 + I_C) \\ \qquad +2m_2 r_2 r_0 \cos(\theta_3 + \theta_5) + 2[m_1 r_1 r_0 \cos(\theta_3 - \theta_4) + m_2 l_1 r_0 \cos(\theta_3 + \theta_4)] \\ \qquad +m_1 r_1^2 + m_2 l_1^2 + I_D + 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E \\ D_q(3,4) = -m_1 r_1 r_0 \cos(\theta_3 - \theta_4) - m_2 l_1 r_0 \cos(\theta_3 + \theta_4) - m_2 r_2 r_0 \cos(\theta_3 + \theta_5) \\ \qquad -(m_1 r_1^2 + m_2 l_1^2 + I_D) - 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) - (m_2 r_2^2 + I_E) \\ D_q(3,5) = m_2 r_2 r_0 \cos(\theta_3 + \theta_5) + m_2 l_1 r_2 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E \end{cases} \tag{6.66}$$

$$\begin{cases} D_q(4,1) = 0 = D_q(1,4) \\ D_q(4,2) = 0 = D_q(2,4) \\ D_q(4,3) = -m_1 r_1 r_0 \cos(\theta_3 - \theta_4) - m_2 l_1 r_0 \cos(\theta_3 + \theta_4) - m_2 r_2 r_0 \cos(\theta_3 + \theta_5) \\ \qquad -(m_1 r_1^2 + m_2 l_1^2 + I_D) - 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) - (m_2 r_2^2 + I_E) = D_q(3,4) \\ D_q(4,4) = m_1 r_1^2 + m_2 l_1^2 + I_D + 2m_2 l_1 r_2 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E \\ D_q(4,5) = -m_2 l_1 r_2 \cos(\theta_4 - \theta_5) - (m_2 r_2^2 + I_E) \end{cases} \quad (6.67)$$

$$\begin{cases} D_q(5,1) = 0 = D_q(1,5) \\ D_q(5,2) = 0 = D_q(2,5) \\ D_q(5,3) = m_2 r_2 r_0 \cos(\theta_3 + \theta_5) + m_2 l_1 r_2 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E = D_q(3,5) \\ D_q(5,4) = -m_2 l_1 r_2 \cos(\theta_4 - \theta_5) - (m_2 r_2^2 + I_E) = D_q(4,5) \\ D_q(5,5) = -m_2 r_2^2 - I_E \end{cases} \quad (6.68)$$

Then, put the equation (6.68) (the relationship between relative angle and absolute angle) into equation (6.64)~(6.68), we could get inertia matrix $\mathbf{D_q}(q)$ as pursue:

$$\begin{cases} D_q(1,1) = m_2 r_2^2 + I_A \\ D_q(1,2) = -(m_2 r_2^2 + I_A) - m_2 r_2 l_1 \cos(\gamma_L) \\ D_q(1,3) = m_2 r_2^2 + I_A + m_2 r_2 l_1 \cos(\gamma_L) + m_2 r_2 r_0 \cos(\gamma_L - \beta_L) \\ D_q(1,4) = 0 \\ D_q(1,5) = 0 \end{cases} \quad (6.69)$$

$$\begin{cases} D_q(2,1) = -(m_2 r_2^2 + I_A) - m_2 r_2 l_1 \cos(\gamma_L) = D_q(1,2) \\ D_q(2,2) = -(m_2 r_2^2 + I_A) + m_2 l_1^2 + m_1 r_1^2 + I_B \\ D_q(2,3) = -(m_2 r_2^2 + I_A) - 2m_2 r_2 l_1 \cos(\gamma_L) - (m_2 l_1^2 + m_1 r_1^2 + I_B) \\ \qquad -m_2 r_2 r_0 \cos(\gamma_L - \beta_L) - (m_2 r_0 l_1 + m_1 r_0 r_1) \cos(-\beta_L) \\ D_q(2,4) = 0 \\ D_q(2,5) = 0 \end{cases} \quad (6.70)$$

$$
\begin{cases}
D_q(3,1) = m_2 r_2^2 + I_A + m_2 r_2 l_1 \cos(\gamma_L) + m_2 r_2 r_0 \cos(\gamma_L - \beta_L) = D_q(1,3) \\
D_q(3,2) = -(m_2 r_2^2 + I_A) - 2m_2 r_2 l_1 \cos(\gamma_L) - (m_2 l_1^2 + m_1 r_1^2 + I_B) \\
\qquad -m_2 r_2 r_0 \cos(\gamma_L - \beta_L) - (m_2 r_0 l_1 + m_1 r_0 r_1)\cos(-\beta_L) = D_q(2,3) \\
D_q(3,3) = m_2 r_2^2 + I_A + 2m_2 r_2 l_1 \cos(\gamma_L) + 2m_2 r_2 r_0 \cos(\gamma_L - \beta_L) + (m_2 l_1^2 + m_1 r_1^2 \\
\qquad +I_B) + 2\,(m_2 r_0 l_1 + m_1 r_0 r_1)\cos(-\beta_L) + (2m_2 r_0^2 + 2m_1 r_0^2 + I_C) \\
\qquad +2m_2 r_2 r_0 \cos(\beta_R - \gamma_R) + 2[m_1 r_1 r_0 \cos(2\varphi - \beta_R) + m_2 l_1 r_0 \cos(\beta_R)] \\
\qquad +m_1 r_1^2 + m_2 l_1^2 + I_D + 2m_2 r_2 l_1 \cos(\gamma_R) + m_2 r_2^2 + I_E \\
D_q(3,4) = -m_1 r_1 r_0 \cos(2\varphi - \beta_R) - m_2 l_1 r_0 \cos(\beta_R) - m_2 r_2 r_0 \cos(\beta_R - \gamma_R) \\
\qquad -(m_1 r_1^2 + m_2 l_1^2 + I_D) - 2m_2 r_2 l_1 \cos(\gamma_R) - (m_2 r_2^2 + I_E) \\
D_q(3,5) = m_2 r_2 r_0 \cos(\beta_R - \gamma_R) + m_2 l_1 r_2 \cos(\gamma_R) + m_2 r_2^2 + I_E
\end{cases} \tag{6.71}
$$

$$
\begin{cases}
D_q(4,1) = 0 = D_q(1,4) \\
D_q(4,2) = 0 = D_q(2,4) \\
D_q(4,3) = -m_1 r_1 r_0 \cos(2\varphi - \beta_R) - m_2 l_1 r_0 \cos(\beta_R) - m_2 r_2 r_0 \cos(\beta_R - \gamma_R) \\
\qquad -(m_1 r_1^2 + m_2 l_1^2 + I_D) - 2m_2 r_2 l_1 \cos(\gamma_R) - (m_2 r_2^2 + I_E) = D_q(3,4) \\
D_q(4,4) = m_1 r_1^2 + m_2 l_1^2 + I_D + 2m_2 l_1 r_2 \cos(\gamma_R) + m_2 r_2^2 + I_E \\
D_q(4,5) = -m_2 l_1 r_2 \cos(\gamma_R) - (m_2 r_2^2 + I_E)
\end{cases} \tag{6.72}
$$

$$
\begin{cases}
D_q(5,1) = 0 = D_q(1,5) \\
D_q(5,2) = 0 = D_q(2,5) \\
D_q(5,3) = m_2 r_2 r_0 \cos(\beta_R - \gamma_R) + m_2 l_1 r_2 \cos(\gamma_R) + m_2 r_2^2 + I_E = D_q(3,5) \\
D_q(5,4) = -m_2 l_1 r_2 \cos(\gamma_R) - (m_2 r_2^2 + I_E) = D_q(4,5) \\
D_q(5,5) = -m_2 r_2^2 - I_E
\end{cases} \tag{6.73}
$$

In the same way, we can obtain the coriolis and centripetal torques $\mathbf{h_q}(q, \dot{q})$ and also the gravitational torque $\mathbf{G_q}(q)$ in the following form: ($i = 1,2,\dots,5 \;\; j = 1,2,\dots,5$)

$$\begin{cases} h_q(1,1) = 0 \\ h_q(1,2) = -m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(1,3) = m_2 r_2 l_1 \sin(\gamma_L) + m_2 r_2 r_0 \sin(\gamma_L - \beta_L) \\ h_q(1,4) = 0 \\ h_q(1,5) = 0 \end{cases} \tag{6.74}$$

$$\begin{cases} h_q(2,1) = m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(2,2) = 0 \\ h_q(2,3) = -m_2 r_2 r_0 \sin(\gamma_L - \beta_L) - (m_2 r_0 l_1 + m_1 r_0 r_1) \sin(-\beta_L) \\ h_q(2,4) = 0 \\ h_q(2,5) = 0 \end{cases} \tag{6.75}$$

$$\begin{cases} h_q(3,1) = m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(3,2) = 2m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(3,3) = -2m_2 l_1 r_0 \sin(\beta_R) - 2m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\ h_q(3,4) = 0 \\ h_q(3,5) = 0 \end{cases} \tag{6.76}$$

$$\begin{cases} h_q(4,1) = 0 \\ h_q(4,2) = 0 \\ h_q(4,3) = m_2 l_1 r_0 \sin(\beta_R) + m_2 r_0 r_2 \sin(\beta_R - \gamma_R) + m_1 r_0 r_1 \sin(2\varphi - \beta_R) \\ h_q(4,4) = 0 \\ h_q(4,5) = -m_2 r_2 l_1 \sin(\gamma_R) \end{cases} \tag{6.77}$$

$$\begin{cases} h_q(5,1) = 0 \\ h_q(5,2) = 0 \\ h_q(5,3) = -m_2 r_2 r_0 \sin(\beta_R - \gamma_R) - m_2 r_2 l_1 \sin(\gamma_R) \\ h_q(5,4) = m_2 r_2 l_1 \sin(\gamma_R) \\ h_q(5,5) = 0 \end{cases} \tag{6.78}$$

$$\begin{cases} G_q(1) = m_2 g r_2 \sin(\gamma_L + \varphi - \beta_L) \\ G_q(2) = -m_2 g r_2 \sin(\gamma_L + \varphi - \beta_L) - (m_2 g l_1 + m_1 g r_1) \sin(\varphi - \beta_L) \\ G_q(3) = m_2 g r_2 \sin(\gamma_L + \varphi - \beta_L) + (m_2 g l_1 + m_1 g r_1) \sin(\varphi - \beta_L) \\ \quad +(2m_2 g r_0 + 2m_1 g r_0) \sin(\varphi) - (m_2 g l_1 + m_1 g r_1) \sin(\beta_R - \varphi) \\ \quad -m_2 g r_2 \sin(\beta_R - \varphi - \gamma_R) \\ G_q(4) = (m_1 g r_1 + m_2 g l_1) \sin(\beta_R - \varphi) + m_2 g r_2 \sin(\beta_R - \varphi - \gamma_R) \\ G_q(5) = m_2 g r_2 \sin(\beta_R - \varphi - \gamma_R) \end{cases} \quad (6.79)$$

6.2.3 Summary

In this part, the methodology for the derivation of equations of movement to explain the motion of a biped robot walking on a plain smooth area showed. We developed a 5-link kinematic model of the biped robot which has effectively free degree of freedom to keep equation in achievable level. The dynamic model is created through the phase of single support foot, determined by Lagrangian equation. In this resulted equation, the joint angles are definite angles. As a result, the dynamic equations adjusted looking at relative angles between each link for control purpose. The proposed model has less complexity, It happens because of two reasons: first, the structures of both legs of robot are similar and the second one, the entire kinetic energy and potential energy have uncomplicated expressions.

6.3 Control Strategy

Our supposed control strategy is based on low-level control and high-level control of biped robot stepping over dynamic obstacle and can design separately. The objectives of control strategy are the tracking of desired joint angles in low-level control and to plan the footstep and joint angle for a few steps in high-level control.

6.3.1 High-level Controller Structure

The target of high-level control is to provide the path and joint angle planning for robot in dynamic environments. Because of this goal the control method needs to use a predictive approach in line with an on-line optimization of learning process. So our model is based on Semi On-line Fuzzy Q-learning (SOFQL) theory that is proposed in chapter 5.

The high-level control methods can be divided into the following parts:

- The first step is to compute the position of the foot and obstacle, and the velocity of obstacle which is used for simulating the dynamic environment. We suppose the walking of biped robot is a succession of single support phases.
- The second part involves a fuzzification of inputs of states.
- The third step involves the Semi On-line Fuzzy Q-learning algorithm which must choose one output for each activated rules; we use two Fuzzy Q-learning algorithms, one for learning step length and time, the other for maximum step height.
- This part gives the reinforcement signals. The reinforcement signal informs the learning agent about the quality of selected action.
- The fifth part contains the issued point such as starting point, target point and maximum step height to get the trajectories.
- The sixth and final part release the joint profiles based on the foot trajectory by inverse kinematics.

6.3.1.1 Development of Trajectory for Swing Leg

Actually the SOFQL is developed to train the step length, maximum height and duration time of every step which find out the trajectory of swing leg. Additionally the

starting point x_{min} and landing point x_{max} of foot which are decided by length of step (refer to 6.80), and the maximum step height h_{max} that is chosen as the peak of trajectory will shape the trajectory. The time duration of step only identifies the frequency of locomotion. The swing trajectory is shown in figure 6.5,

$$L_s = x_{max} - x_{min} \tag{6.80}$$

The swing trajectory can be obtained with starting point, landing and peak point, by interpolation algorithm. We chose cubic spline interpolation in our method because of its fewer amounts of calculation and better accuracy.

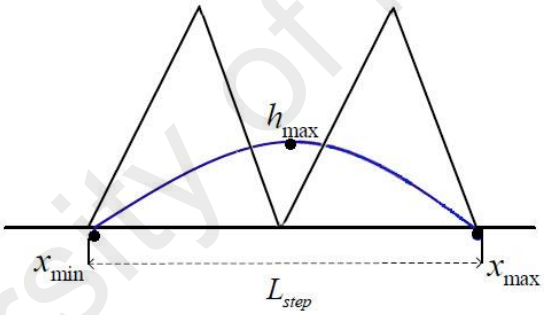


Figure 6.5: Swing trajectory during on step

In cubic spline algorithm, for $n+1$ given points, every point on the curve is the corresponding value of cubic polynomial, which can be expressed as:

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \tag{6.81}$$

By limiting the derivative coefficient of every cubic polynomial at the smashing point, the coefficients of the polynomial may be figured out by the following equations.

(6.82)

$$\begin{cases} a_i = S_i(x_i) = y_i \\ b_i = S_i'(x_i) = -\dfrac{h_i}{6} - \dfrac{h_i}{3} z_i + \dfrac{y_{i+1} - y_i}{h_i} \\ c_i = \dfrac{1}{2} S_i''(x_i) = \dfrac{z_i}{2} \\ d_i = \dfrac{1}{6} S_i'''(x_i) = \dfrac{z_{i+1} - z_i}{6h_i} \end{cases}$$

In which, $h_i = x_{i+1} - x_i$ by using the idea of multiplication, we shall get:

$$\begin{cases} z_i = 0 & if \ \ i = 0, n \\ z_i = (v_i - h_i z_{i+1})/u_i \end{cases} \tag{6.83}$$

Through Gaussian elimination, we can get:

$$if \ \ i = 1 \ \ \begin{cases} u_1 = 2(h_0 - h_1) \\ v_1 = 6(b_1 - b_0) \end{cases} \tag{6.84}$$

$$else \ \begin{cases} u_i = 2(h_{i-1} + h_i) - h_{i-1}^2/u_{i-1} \\ v_i = 6(b_i - b_{i-1}) - h_{i-1} v_{i-1}/u_{i-1} \end{cases} \tag{6.85}$$

The coefficients of every cubic polynomial can be calculated by replacement equation (6.83), (6.84) and (6.85) into equation (6.82). After that, new equation is used to interpolate the starting point, landing point and vertex; a smooth swing trajectory can be achieved.

6.3.1.2 Generation of Joint Angle

6.3.1.2.1 Inverse Kinematic

There are two main approaches to develop inverse kinematical formula of humanoid robot, algebraic method and geometry method. The geometry method is generally applied on robot with simple structure, like as robot arm movement in two dimension

plan. The algebraic method is more suitable for the robot that moves in three dimensions or with more joints and degrees of freedom (Chung, 2000). So, the way by which derivative the inverse kinematic equation depends on the structure of robot. In this thesis, we simulate the five link model to explain the robot structure and its movement in sagittal plane. Therefore, the geometry method is selected to develop the inverse kinematic equation of biped robot.

The walking period of robot contains single support leg phase and stance phase. During the single support leg phase because of the half of support leg, function is preventing the swing leg from touching the ground before stance, and has little effect on the walking movement of robot. Hence, we did not present the knee joint of support leg in the inverse kinematical equation.

The concept of developing the inverse kinematics by geometry method is to calculate the locomotion of every joint, given the segment position of robot relative to the reference coordinate. So, the inverse kinematics of biped robot could be described as that the desired position of swing leg is known, compute the joint angle to satisfy its location posture. In the gait planning of biped robot, usually the support leg is assumed immovable.

To describe the problem, we redefine equation in the angle of thigh of support leg with respect to the vertical direction as θ_1. The angle of thigh of swing leg regarding the vertical direction shown as θ_2 and θ_3 represent for the angle of shank of swing leg. The length of thigh and shank for both legs is described as $l_1$and l_2 singly.

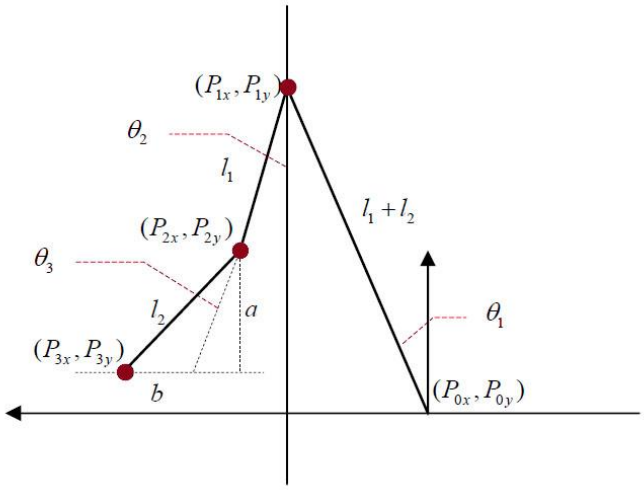


Figure 6.6: Geometrical relationship between stance leg and swing leg

The geometrical relationship between stance leg and swing leg of humanoid robot is illustrated in figure 6.6. Generally, in the stepping phase of robot, the stance leg does not bend. After footstep planning approach which is developed in previous chapter, the angle between stance foot and vertical line θ_1 and its position (P_{0x}, P_{0y}) are taken as primary condition. The coordinate of any point (P_{3x}, P_{3y}) on the swing trajectory can be calculated by cubic spline interpolating.

The position of hip joint (P_{1x}, P_{1y}) and position of knee joint for swing leg (P_{2x}, P_{2y}) can be shown as:

$$\begin{cases} P_{1x} = (l_1 + l_2)\sin\theta_1 \\ P_{1y} = (l_1 + l_2)\cos\theta_1 \end{cases} \tag{6.86}$$

$$\begin{cases} P_{2x} = l_1 \sin\theta_2 + P_{1x} \\ P_{2y} = P_{1y} - l_1 \cos\theta_2 \end{cases} \tag{6.87}$$

Derive from geometric method, we can find:

$$a = P_{2y} - P_{3y} = (P_{1y} - l_1\cos\theta_2) - P_{3y} = (l_1 + l_2)\cos\theta_1 - l_1\cos\theta_2 - P_{3y} \tag{6.88}$$

$$b = P_{3x} - P_{2x} = P_{3x} - (l_1 \sin\theta_2 + P_{1x}) = P_{3x} - (l_1 + l_2)\sin\theta_1 - l_1 \sin\theta_2 \quad (6.89)$$

According to relation of sides of right-angled triangle we have:

$$[(l_1 + l_2)\cos\theta_1 - l_1 \cos\theta_2 - P_{3y}]^2 + [P_{3x} - (l_1 + l_2)\sin\theta_1 - l_1 \sin\theta_2]^2 = 1 \quad (6.90)$$

The angle between thigh of swing leg and direction θ_2 can be calculated with follow equation based on the above equations,

$$\theta_2 = atg\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right) \mp atg\left(\frac{A}{B}\right) \quad (6.91)$$

In which:

$$\begin{cases} A = 2l_1[(l_1 + l_2)\sin\theta_1 - P_{3x}] \\ B = 2l_1[P_{3y} - (l_1 + l_2)\cos\theta_1] \\ C = (l_1 + l_2)\left[2P_{3x}\sin\theta_1 + 2P_{3y}\cos\theta_1 - (l_1 + l_2)\right] + (l_2^2 - l_1^2 - P_{3x}^2 - P_{3y}^2) \end{cases} \quad (6.92)$$

In this equation, If C=0 that means:

$$\theta_1 = atg\left(\frac{\sqrt{A_1^2 + B_1^2 + C_1^2}}{C_1}\right) \mp atg\left(\frac{A_1}{B_1}\right) \quad (6.93)$$

So,

$$\begin{cases} A_1 = 2P_{3x}(l_1 + l_2) \\ B_1 = 2P_{3y}(l_1 + l_2) \\ C_1 = 2l_1^2 + 2l_1 l_2 + P_{3x}^2 + P_{3y}^2) \end{cases} \quad (6.94)$$

In this case $\theta_2$must be calculated as:

$$\theta_2 = atg\left(\frac{-B}{A}\right) \quad (6.95)$$

Based on the geometric relation and with respect to the vertical direction θ_3 in the similar way, we can find:

$$l_2 \sin(\theta_2 + \theta_3) = b \tag{6.96}$$

By expanssion the above equation:

$$(l_2 \cos\theta_2)\sin\theta_3 + (l_2 \sin\theta_2)\cos\theta_3 = P_{3x} - (l_1 + l_2)\sin\theta_1 - l_1 \sin\theta_2 \tag{6.97}$$

Thus, θ_3 can be calculated as below:

$$\theta_3 = atg\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \mp atg\left(\frac{F}{G}\right) \tag{6.98}$$

In which:

$$\begin{cases} F = l_2 \cos\theta_2 \\ G = l_2 \sin\theta_2 \\ H = P_{3x} - (l_1 + l_2)\sin\theta_1 - l_1 \sin\theta_2 \end{cases} \tag{6.99}$$

When, $\theta_2 = a\sin\left[\left(\frac{P_{3x} - (l_1 + l_2)\sin\theta_1}{l_1}\right)\right]$ and $H = 0$. In this case θ_3 can be calculated as:

$$\theta_3 = atg\left(\frac{-G}{F}\right) \tag{6.100}$$

Hip joint angle θ_2 and knee joint angle θ_3 are computed by using equations (6.91) and (6.98). when $C \neq 0$ and $H \neq 0$, we have four combination of θ_2 and θ_3 as below (this equation number is 6.101):

$$[\theta_2, \theta_3] = \begin{bmatrix} atg\left(\frac{A}{B}\right) - atg\left(\frac{\sqrt{A^2+B^2-C^2}}{C}\right) & atg\left(\frac{F}{G}\right) - atg\left(\frac{\sqrt{F^2+G^2-H^2}}{H}\right) \\ atg\left(\frac{A}{B}\right) - atg\left(\frac{\sqrt{A^2+B^2-C^2}}{C}\right) & atg\left(\frac{F}{G}\right) + atg\left(\frac{\sqrt{F^2+G^2-H^2}}{H}\right) \\ atg\left(\frac{A}{B}\right) + atg\left(\frac{\sqrt{A^2+B^2-C^2}}{C}\right) & atg\left(\frac{F}{G}\right) - atg\left(\frac{\sqrt{F^2+G^2-H^2}}{H}\right) \\ atg\left(\frac{A}{B}\right) + atg\left(\frac{\sqrt{A^2+B^2-C^2}}{C}\right) & atg\left(\frac{F}{G}\right) + atg\left(\frac{\sqrt{F^2+G^2-H^2}}{H}\right) \end{bmatrix}$$

In order to keep the normal walking of biped robot, the knee joint cannot bend backward during both single-support phase and stance phase. With this ristriction of knee joint, Hip joint angle θ_2 and knee joint angle θ_3 can combine as follows:

$$[\theta_2, \theta_3] = [atg\left(\frac{A}{B}\right) - atg\left(\frac{\sqrt{A^2+B^2-C^2}}{C}\right), atg\left(\frac{F}{G}\right) - atg\left(\frac{\sqrt{F^2+G^2-H^2}}{H}\right) \quad (6.102)$$

Thus, during the single-support-leg phase of biped robot, according to the reference position of two feet, the moving trajectory of hip joint and knee joint of swing leg can be calculated.

6.3.1.2.2 Simulation and Result

In this section, we will test our model of generation of joint angles for swing leg based on the given foot step. First, using cubic spline to originate the swing leg trajectory, then, the joint angles of swing leg are calculated by the deduced inverse kinematics of biped robot.

Taking for a certain step, the step length of biped robot L_s as 0.49m, the maximum step height h_{Max} as 0.12m found on the proposed approach and after identifying on the suggested method, the simulation results of coordinates of swing foot movement with hip joint angle of support leg θ_1 is denoted in figure 6.7. Both hip and knee joint angles for swing leg (θ_2, θ_3) varied by the hip joint angle of support leg θ_1 are shown in figure 6.8.

Figure 6.9 shows the simulation results of foot trajectory of swing leg, in which, left leg represents the swinging leg, right leg presents the support leg and black dot is the joint. We suppose that the maximum step height appears in the middle of step length, because the inverse kinematics based on the model of coordinate origin setting vertical to hip joint, the joint angle profile and foot trajectory include both negative and positive number; in figure 6.9 the foot trajectory is symmetric to the coordinate origin.

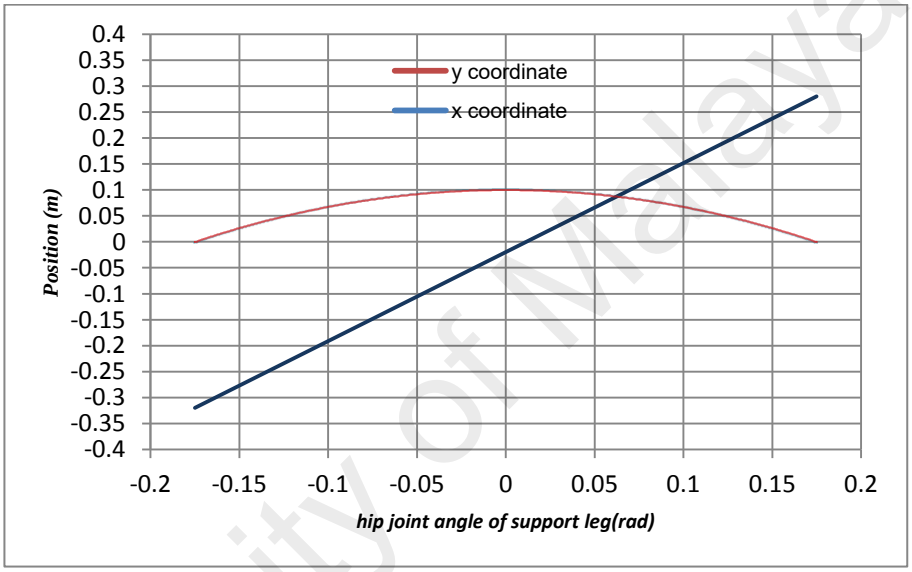


Figure 6.7: coordinates of swing foot varied by the hip joint angle of support leg θ_1

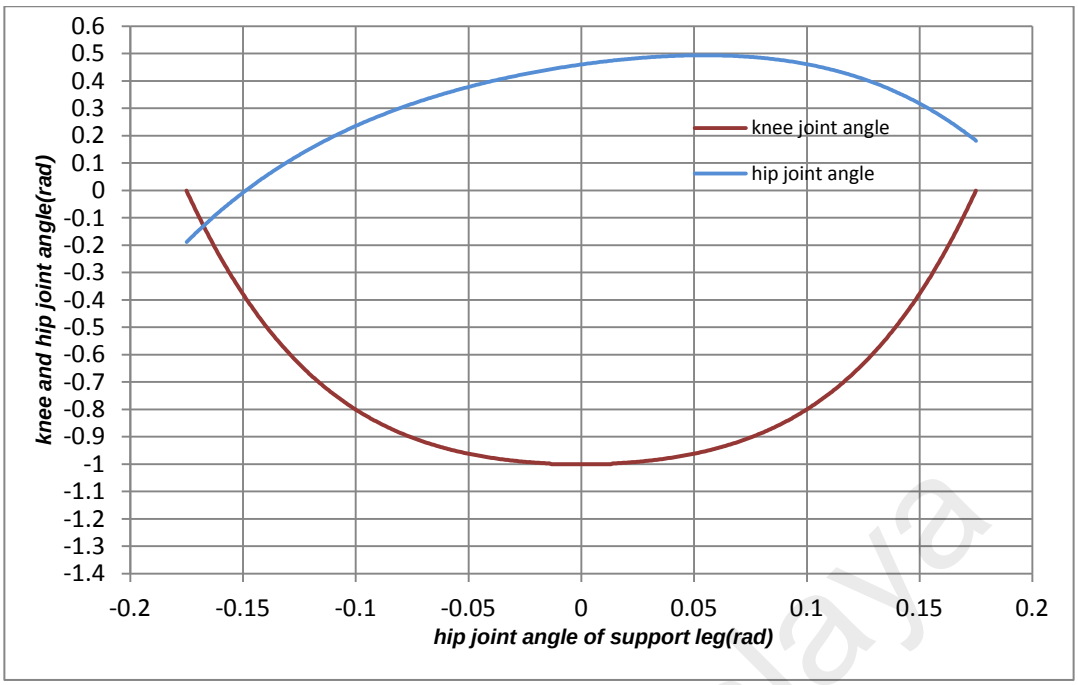


Figure 6.8: joint angles of swing leg varied by the hip joint angle of support leg θ_1

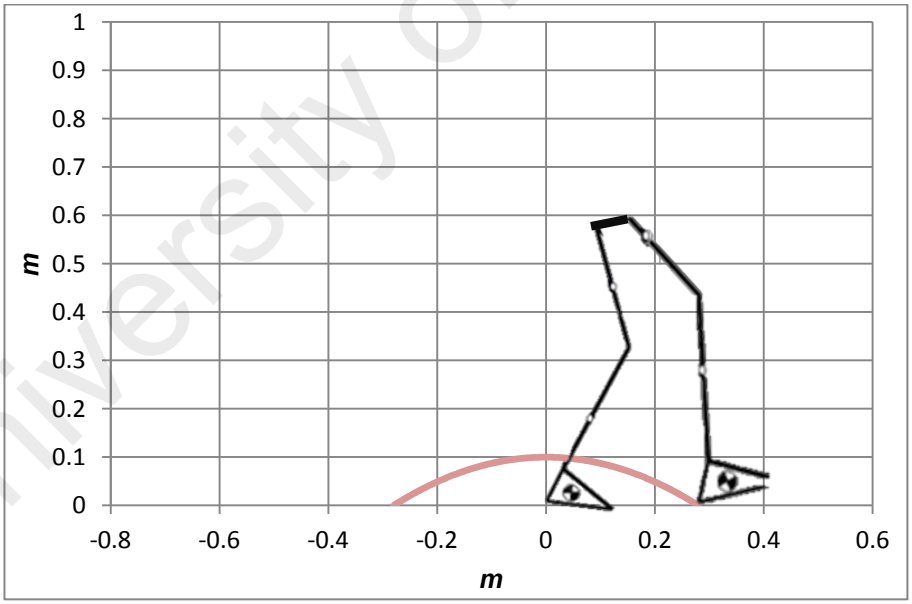
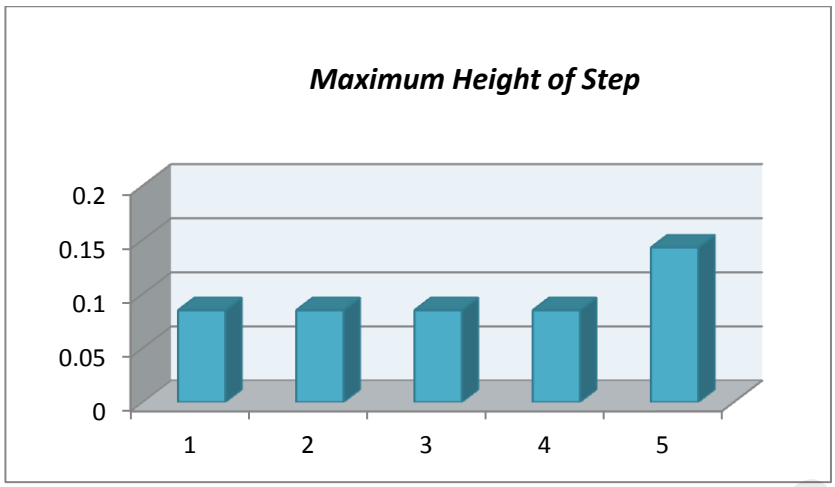


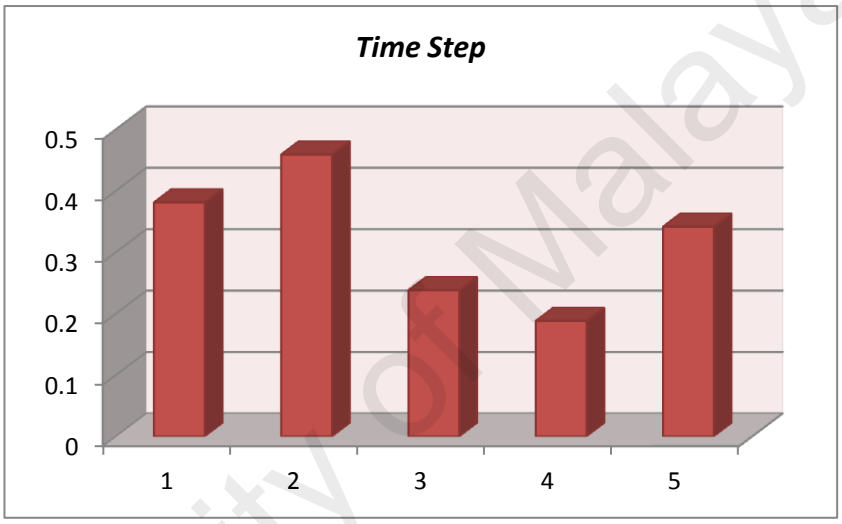
Figure 6.9: foot trajectory of swinging leg

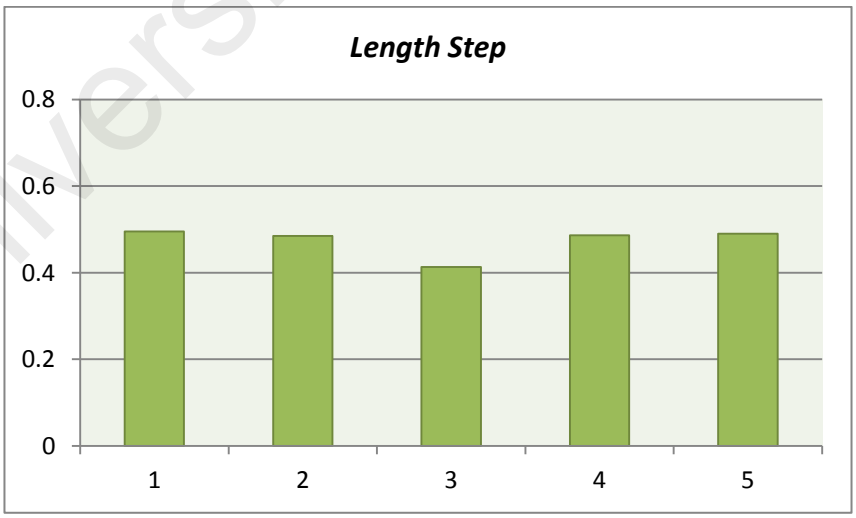
6.3.1.3 Simulation Results and Analysis

Based on the high-level control approach, simulation is done, given the example that is described in chapter 5. We suppose that the velocity of the obstacle is dynamic and may change between 0 to 0.4*m/s*, and the robot can only modify duration of every step within (0.1 0.45)*s*, the maximum height of step can be chosen among [0.08 0.1 0.125 0.15 0.175]*m*, and its step length within [0.15 0.25 0.35 0.45 0.55]*m*, according to some limitation.

Footstep is designed based on Semi Online Fuzzy Q-learning approach. We choose learning rate parameter $\beta = 0.1$ to learn step length, step duration time and maximum step height relating individually to each, discount factor $\gamma = 0.8$. Also we suppose to choose the episode of the training phase as 1000 times, which make secure the Q matrix is trained entirely with all the possible obstacle velocity. The learning results are presented in figure 6.10 and the numeric values are listed in table 6.1.







Figures 6.10: Footstep planning results

Table 6.1: Footstep planning numeric results

Step	1	2	3	4	5
Length	0.495	0.485	0.413	0.486	0.490
Time	0.381	0.459	0.238	0.188	0.3415
Maximum Height	0.085	0.085	0.085	0.085	0.1429

The designated point such as starting point, landing point and maximum step height is applied in cubic spline interpolation to get the swinging trajectory. With inverse kinematics, the curves of hip and knee angle of swing leg are calculated according to the geometrical relationship between pitch angles. As the walking movement of biped robot is its left leg and right leg alternating process, the left and right leg will be the swinging leg by turns. We assume that the footstep planning starts with the left leg of biped robot as its swinging leg. Figure 6.11~6.14 represent the hip joint profile and knee joint profile for both legs varied by step duration time respectively. The result of foot trajectory varied by step duration time is shown in figure 6.15.

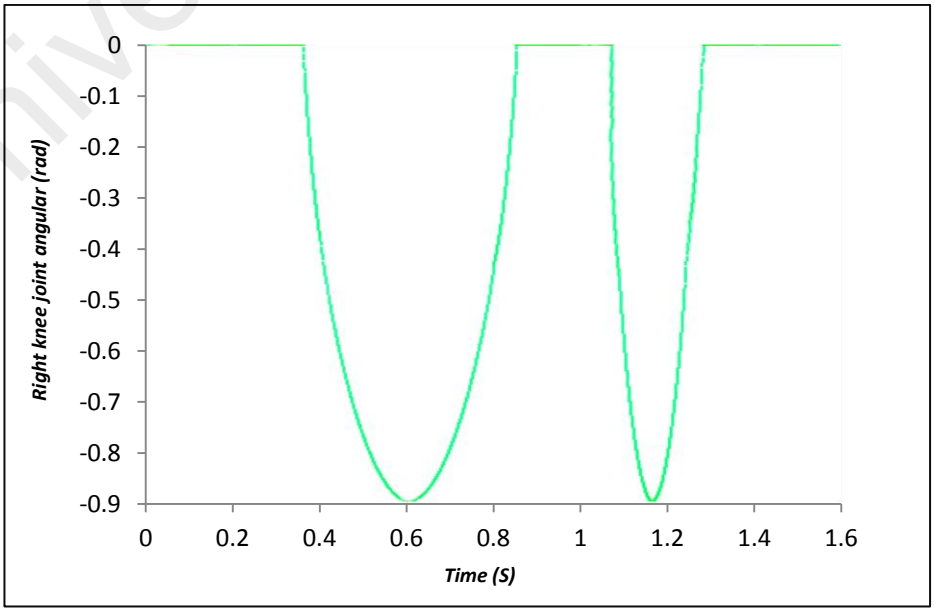


Figure 6.11: joint angle profile for right knee

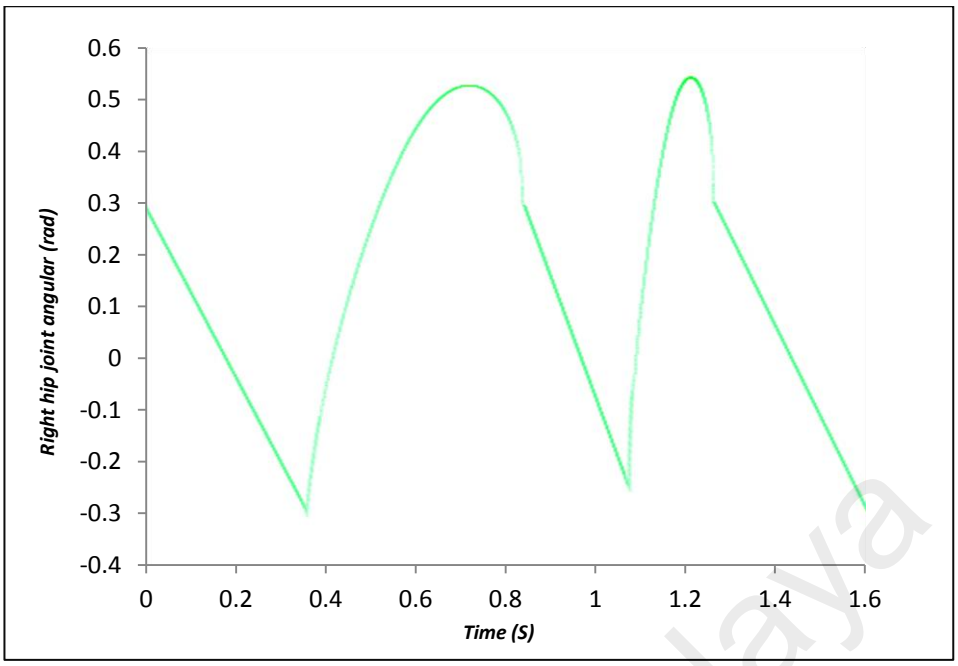


Figure 6.12: joint angle profile for right hip

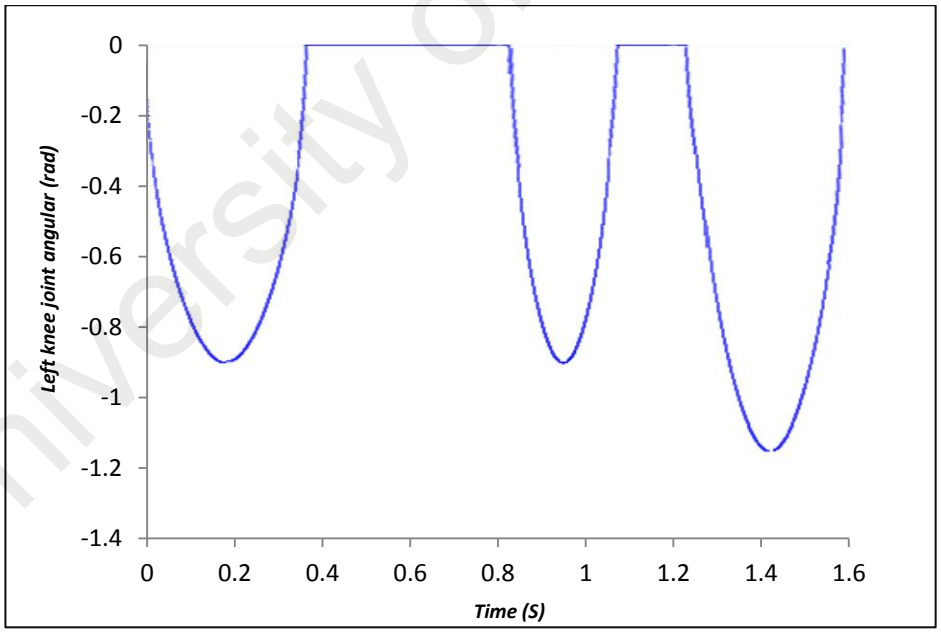


Figure 6.13: joint angle profile for left knee

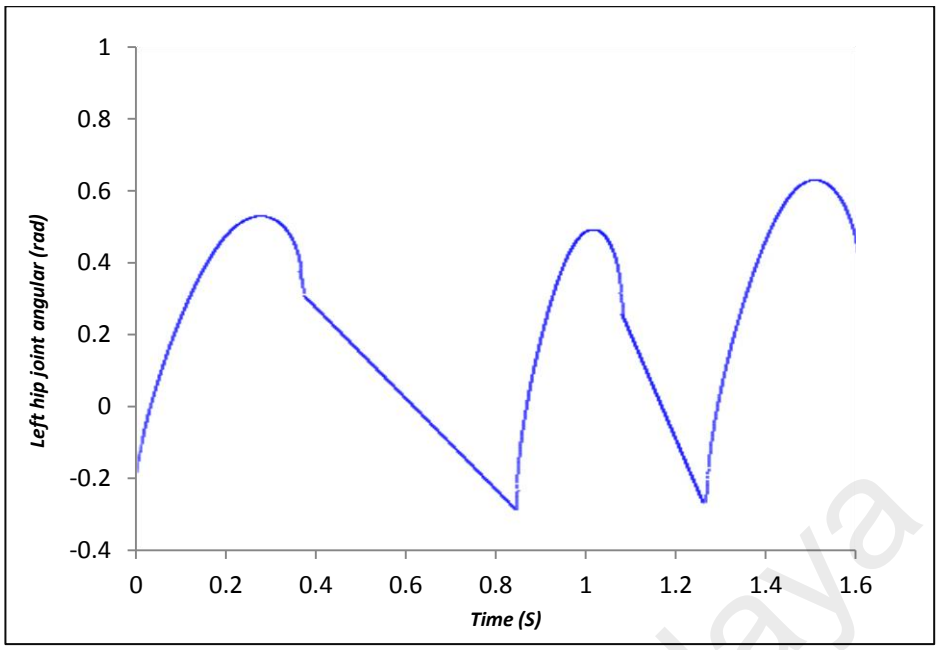


Figure 6.14: joint angle profile for left hip

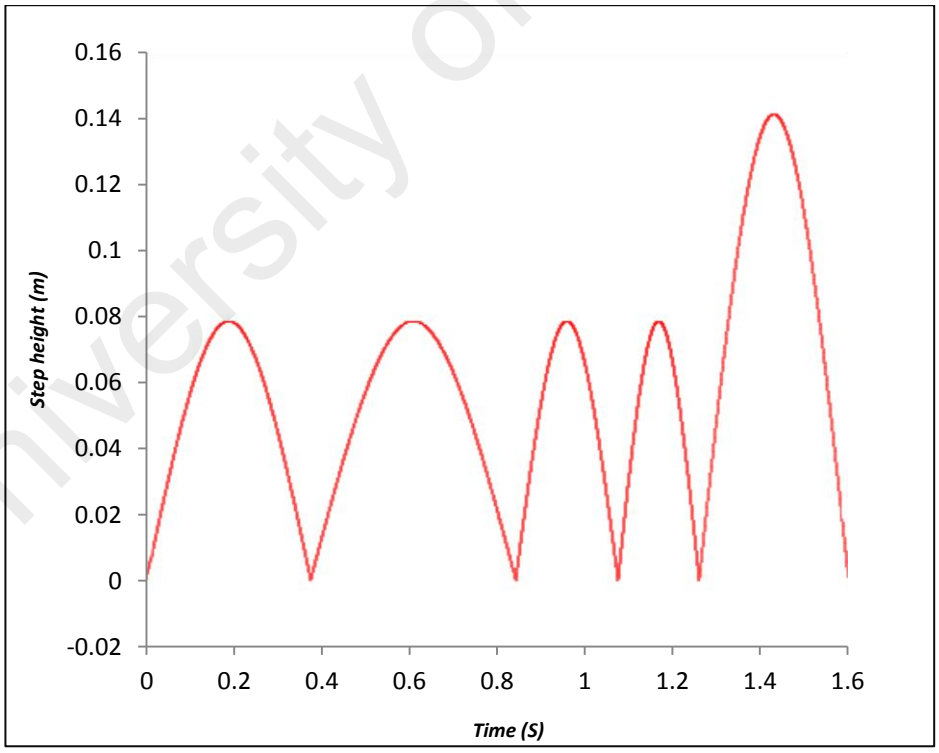


Figure 6.15: Foot trajectory of swing leg varied by step duration time

The simulation results show that the proposed high-level control strategy is achievable for that on one hand, the robot could step over dynamic obstacle successfully, on the other hand, swinging trajectory is smooth and there is no uniqueness value for hip and knee swinging angle.

6.3.2 Low-level Controller Design

The low-level control allows to learn the predicted swinging trajectory and to control the tracking of these desired trajectories, which can be decomposed into three parts (Sabourin, 2005):

- The first one allows the regulation of the average velocity from adjustment of the pitch angle of the trunk.
- The second one is used to compute the trajectories of the swing leg from several outputs of CMAC.
- The third one is composed by some control method in order to ensure the tracking of the reference trajectory at the level of each joint.

The Hierarchically Clustered Fuzzy CMAC is applied to learn the predicted joint angle profile that is gained from high-level control. Regarding the coordinate of swinging foot (P_{3x}, P_{3y}) as the two inputs, two CMAC are utilised for training hip joint angle and knee joint angle separately. The hip angle of support leg θ_1 is chosen within $(\theta_{1min}, \theta_{1max})$ that is changing in one step period. With this chosen θ_1 we can compute the horizontal coordinate of P_{3x} and by cubic spline can calculate the corresponding vertical coordinate P_{3y}.

The weights of HCFCMAC are updated based on the difference between the outputs of HCFCMAC and hip or knee joint angle of swinging leg that are calculated from inverse kinematics. Figure 6.16 shows the result of swinging leg joint angle approximation by our proposed HCFCMAC model in which the blue curve stands for

desired joint angle profile, red one is the hip joint angle approximation and the green curve represents the output of HCFCMAC approximation knee joint angle.

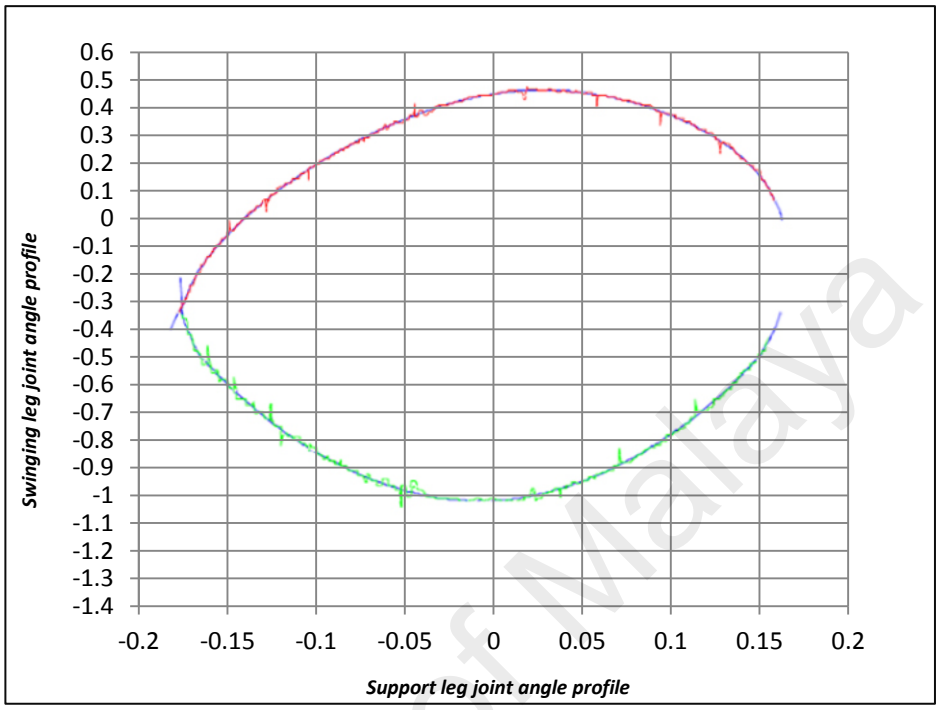


Figure 6.16: Swinging leg joint angle approximations with HCFCMAC

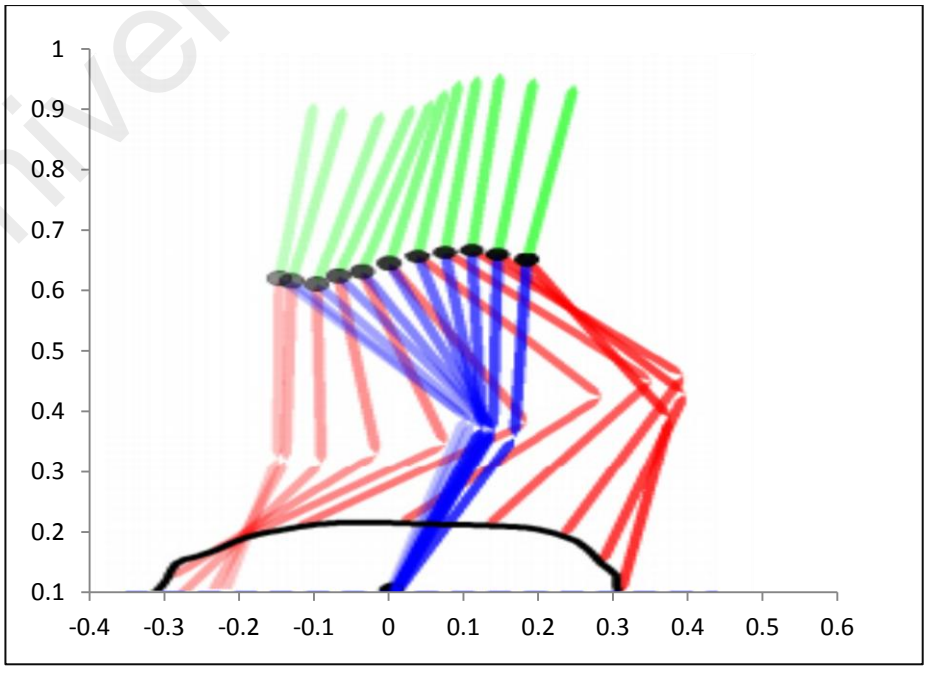


Figure 6.17: Swinging leg trajectory approximations with HCFCMAC

Figure 6.17 gives the pictured simulation result of swinging leg trajectory approximation with CMAC during one step period.

We use these proposed CMAC model to approximate the joint angle profile and hip angle profile, as the walking locomotion of the biped robot is the right leg and left leg substitute sequence. Suppose the stepping motion starts with left leg, the next figures show the simulation results of the output of HCFCMACs approximating the joint angle profile.

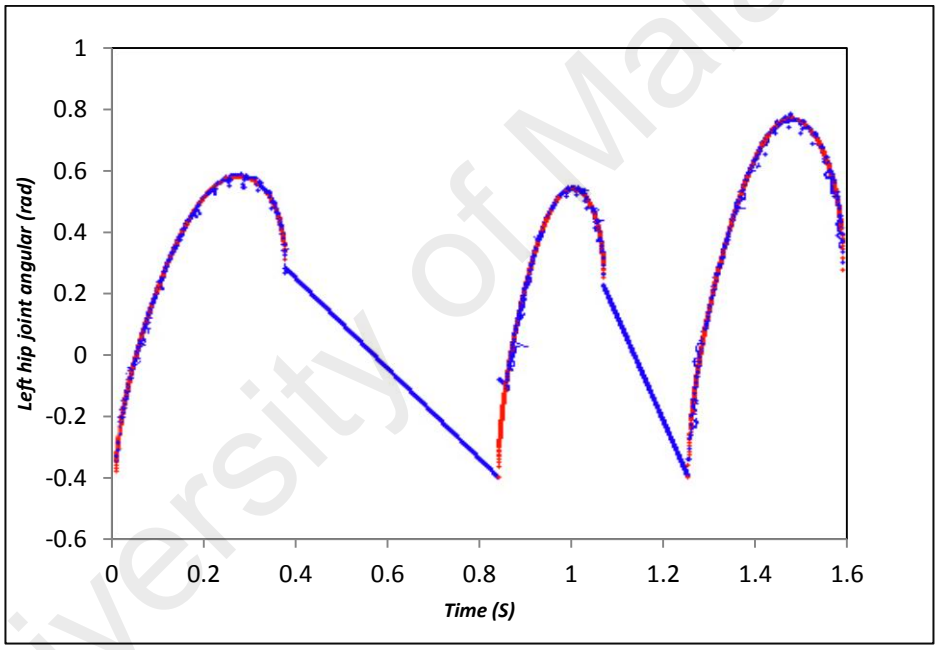


Figure 6.18: HCFCMAC approximation of left hip joint angle changing with step duration time

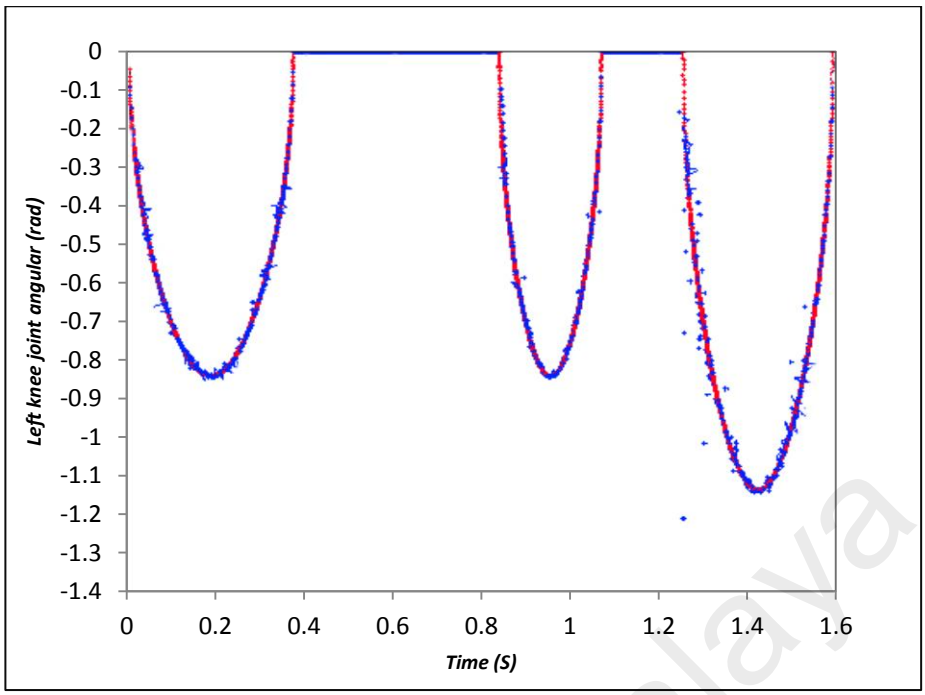


Figure 6.19: HCFCMAC approximation of left knee joint angle changing with step duration time

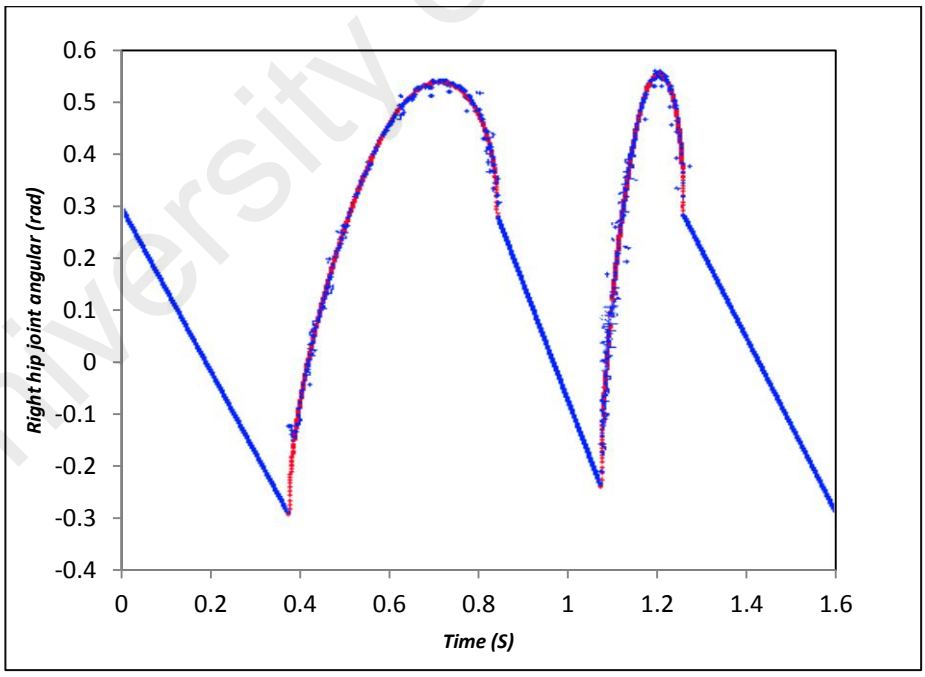


Figure 6.20: HCFCMAC approximation of right hip joint angle changing with step duration time

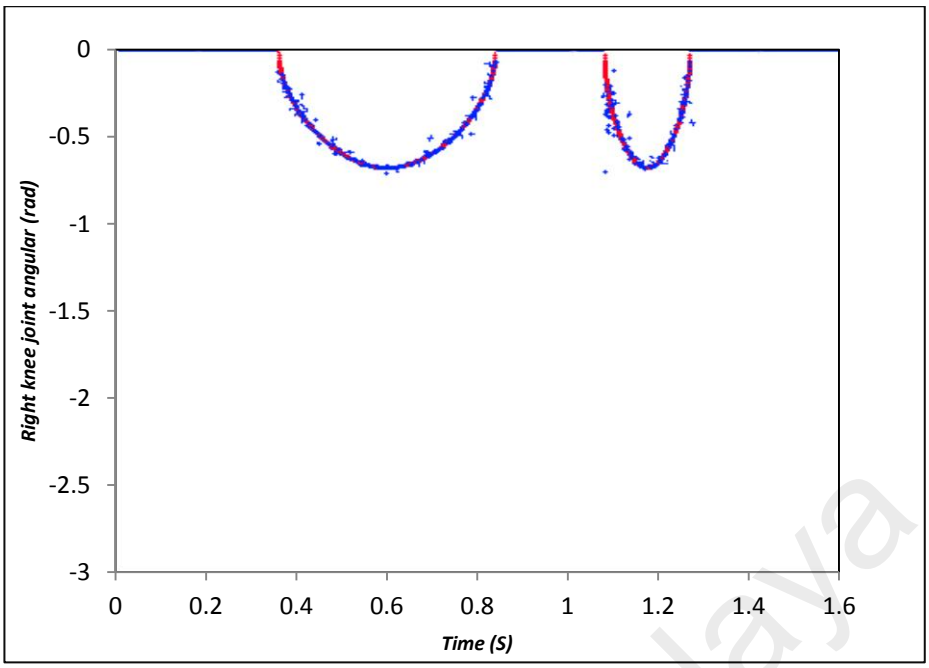


Figure 6.21: HCFCMAC approximation of right knee joint angle changing with step duration time

6.4 Control Strategy Analysis and Simulation Results

In this part, the proposed high level and low level control strategy is tested by simulation example where the biped robot and the obstacle moving oppositely. The biped robot advancement is the third task. It is constituted by the following phases: preswing, initial swing, midswing and terminal swing. So, the phases of the gait cycle are detailed as following

- Initial contact (IC)
- Loading response (LR)
- Midstance (MSt)
- Terminal stance (TSt)
- Preswing (PSw)
- Initial swing (ISw)
- Midswing (MSw)

• Terminal swing (TSw)

It is clear that the stance event has five phases and the swing event has three phases. The preswing phase adapts the biped robot for going ahead, so it is included in the swing event.

The virtual dynamic model of biped robot is simulated with Matlab software. The length of obstacle is equal $0.2m$. For high level control the inputs of footstep planning are the velocity of the obstacle v_o and the distance between the robot and the obstacled_o. The velocity of the obstacle is computed from the distance covered during one step. d_o represent the distance between the front foot and the first side of the obstacle. These inputs are updated at each double support phase. The fuzzification carried out by using 3 and 5 triangular membership function therefore the number of rule is equal 15.

The footstep planning is designed according the algorithm which proposed in chapter 5. During the learning phase the velocity is chosen randomly with Gaussian probability $\sigma = 0.02$ around the average value 0.2 *m/s*. the initial distance between the robot and obstacle is equal $1.5m$. The obstacle length is equal $0.1m$ and the obstacle height is equal $0.3m$.

The five reference gaits that are characterized by parameters of footstep planning given in table 6.2.

Table 6.2: Reference Gaits

Gait	**Gait$_1$**	**Gait$_2$**	**Gait$_3$**	**Gait$_4$**	**Gait$_5$**
Step Length (L_s)	0.50	0.480	0.61	0.496	0.482
Duration Time(T_s)	0.65	0.65	0.58	0.56	0.48
Maximum Height(h_m)	0.32	0.38	0.52	0.57	0.62

The Q matrix is trained for 300 episodes. Figure 6.22 shows the sum of the computing value $\Delta Q(t)$for each episode according to the number of episode.

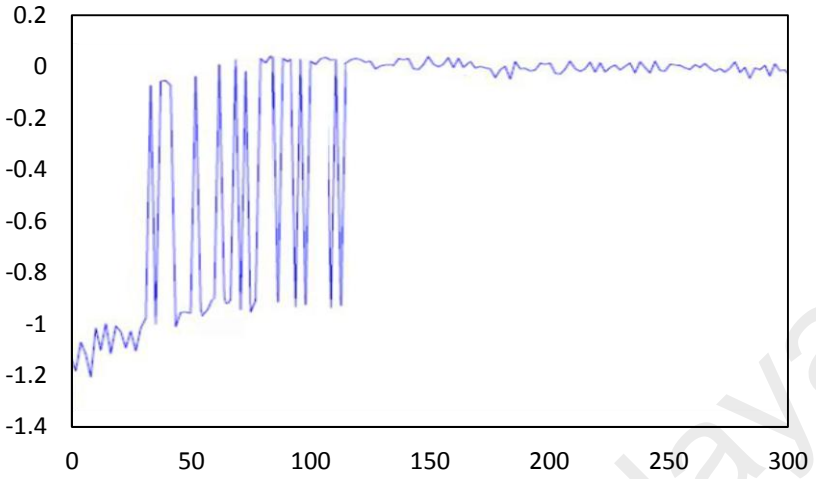


Figure 6.22: Sum of the computing value for each episode

It must be noticed that this value, which depends directly on the reinforcement signal, converges toward 0 quickly within 100 episodes. It is possible to obtain the best rules after the learning phase. Table 6.3 gives the obtained best rules in the case of the presented example.

Table 6.3: generate best rules after learning phase

d_o/v_o	**Small**	**Medium**	**Big**
Very Small	$Gait_3$	$Gait_3$	$Gait_3$
Small	$Gait_5$	$Gait_4$	$Gait_1$
Medium	$Gait_5$	$Gait_2$	$Gait_1$
Big	$Gait_3$	$Gait_3$	$Gait_3$
Very Big	$Gait_3$	$Gait_5$	$Gait_5$

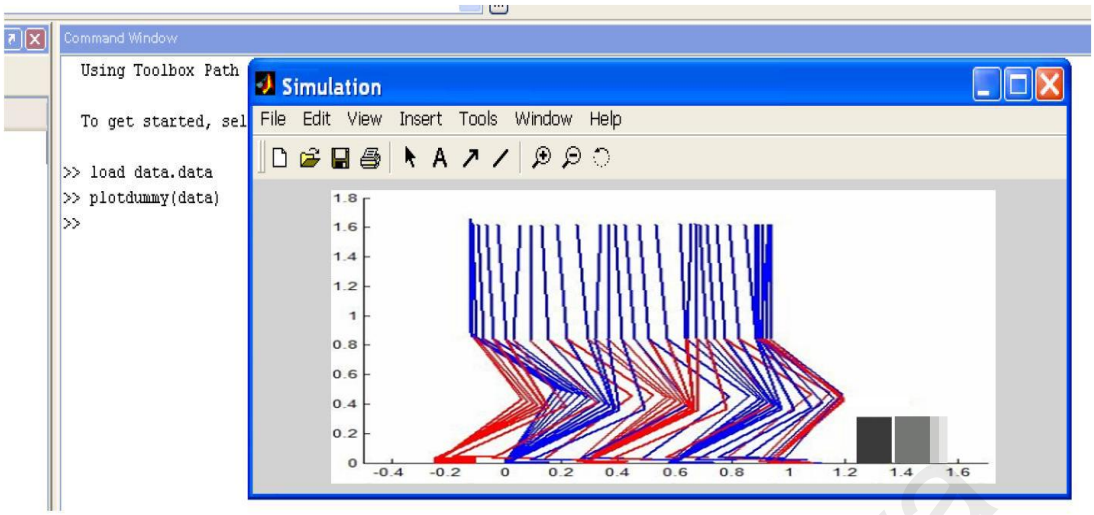


Figure 6.23: Stick Plot of walking motion sequence when the robot is walking toward a dynamic obstacle

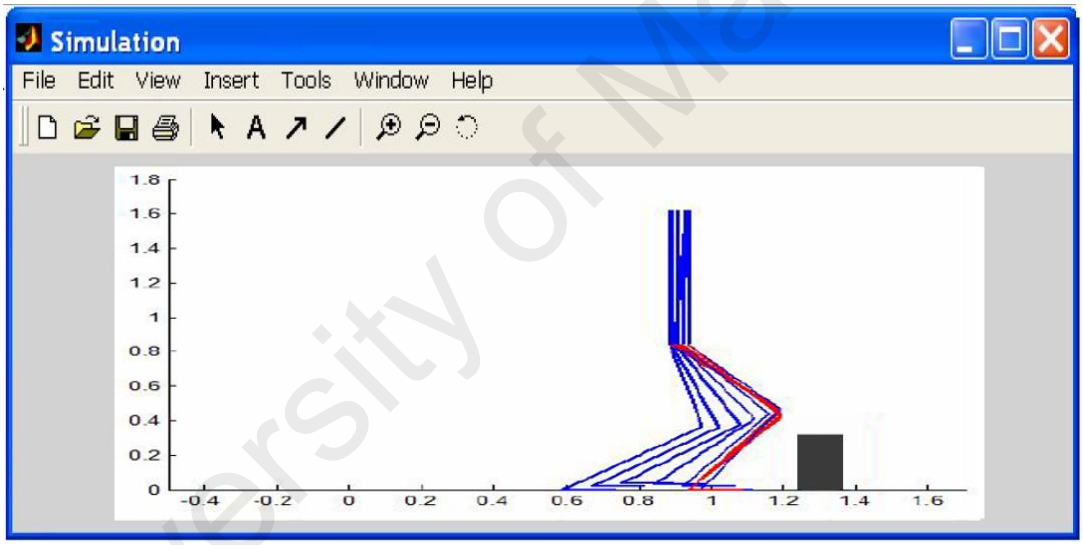


Figure 6.24: Stick Plot of Last gait before obstacle

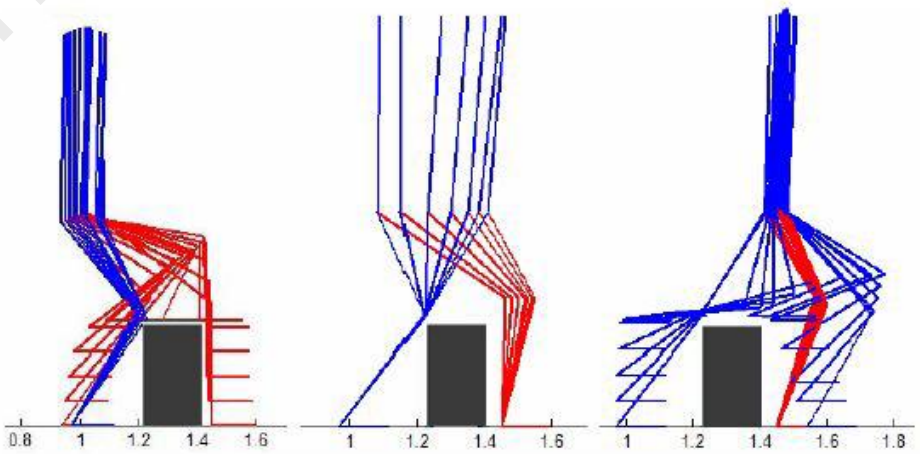


Figure 6.25: Three phases of stepping over obstacle

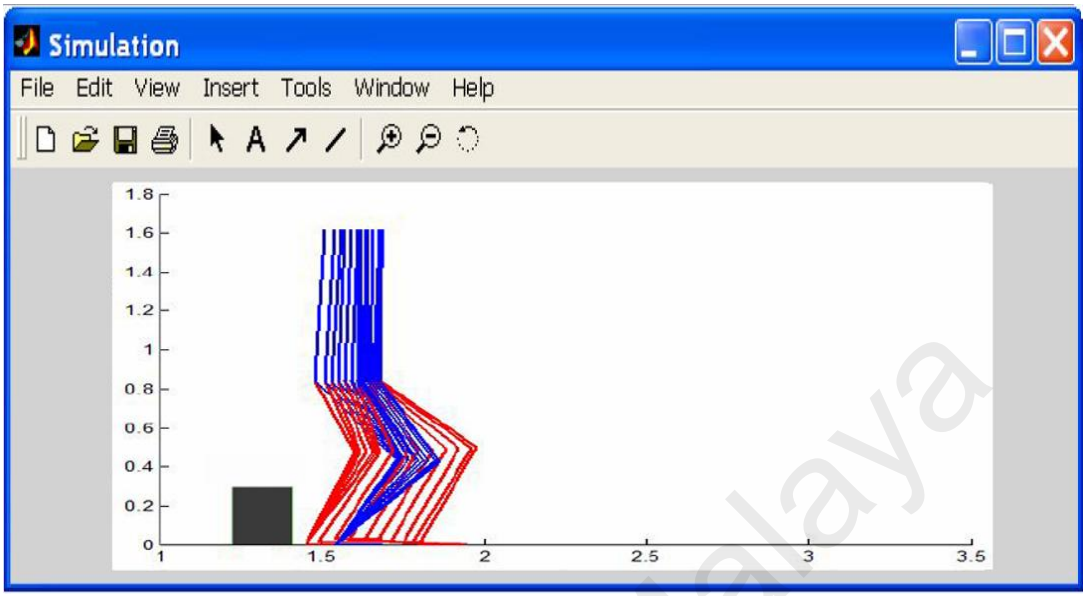


Figure 6.26: Short gait after stepping over obstacle

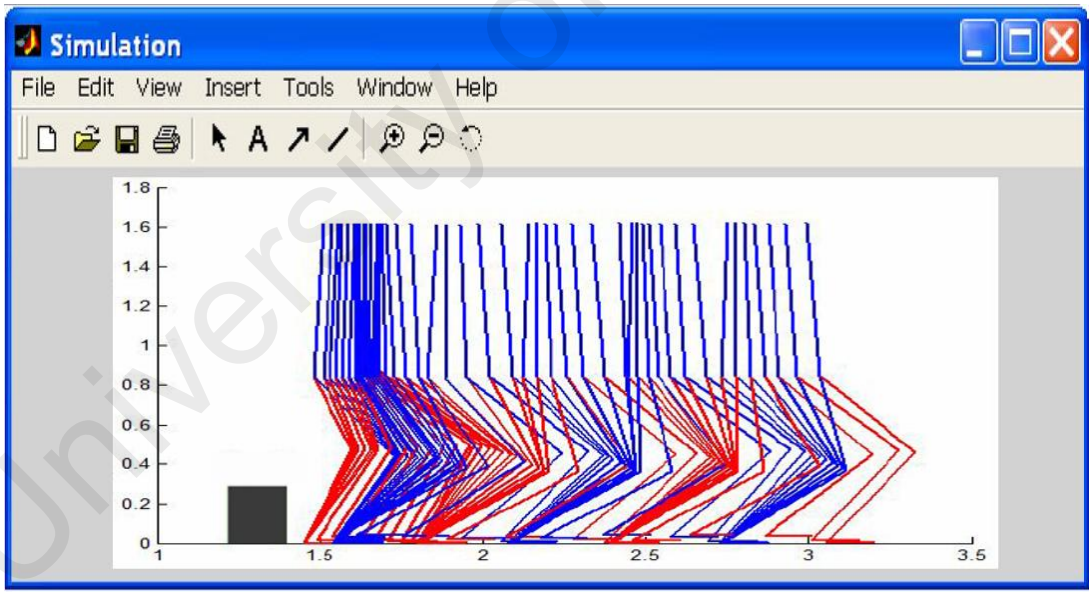


Figure 6.27: Normal walking motion sequence after stepping over obstacle

Figure 6.23 shows several walking gait before obstacle detecting. The obstacle is moving towards the biped robot. The last step is shorter than the other steps because an obstacle is detected which is shows in figure 6.24. Three phases of stepping over the obstacle are shown in figure 6.25. The tip of the swinging leg and the knee would not

touch the edge of obstacle otherwise the crash will be happen. Figure 6.26 and 6.27 shows the biped robot continues walking after passing the Obstacle. It should be noted that after stepping over the obstacle, the legs are next to each other and it should step a short gait before it can resume its normal walking.

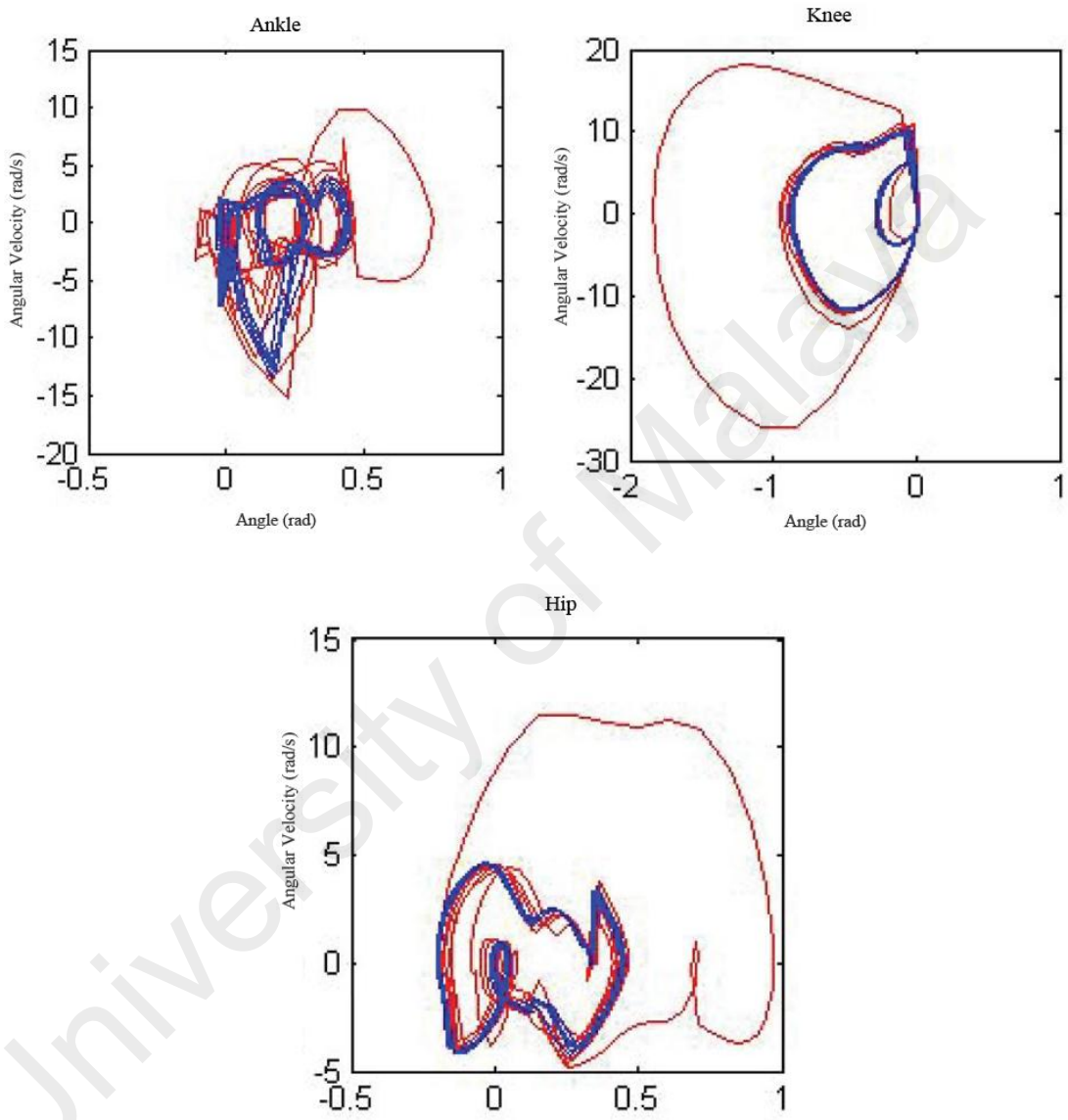


Figure 6.28: Corresponding phase plane plots of ankle, knee and hip in comparison with normal walking pattern (Blue: Normal Walking, Red: Disturbed Walking)

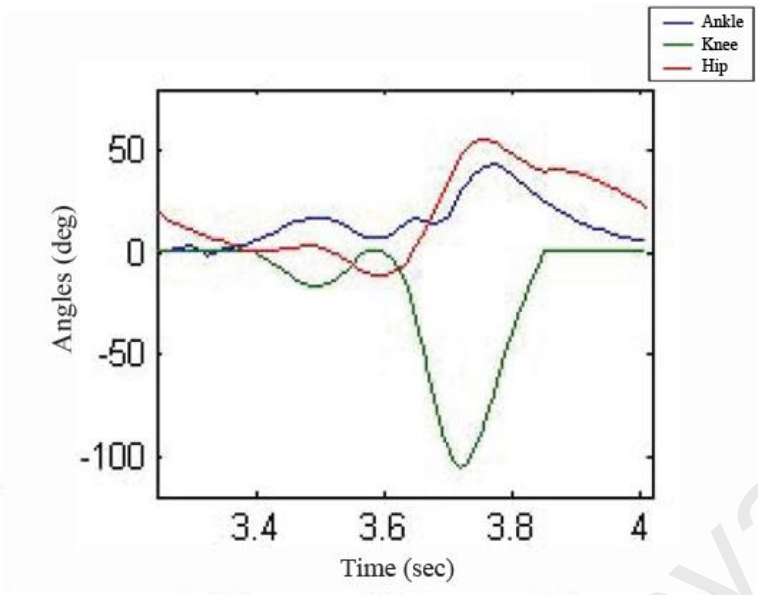
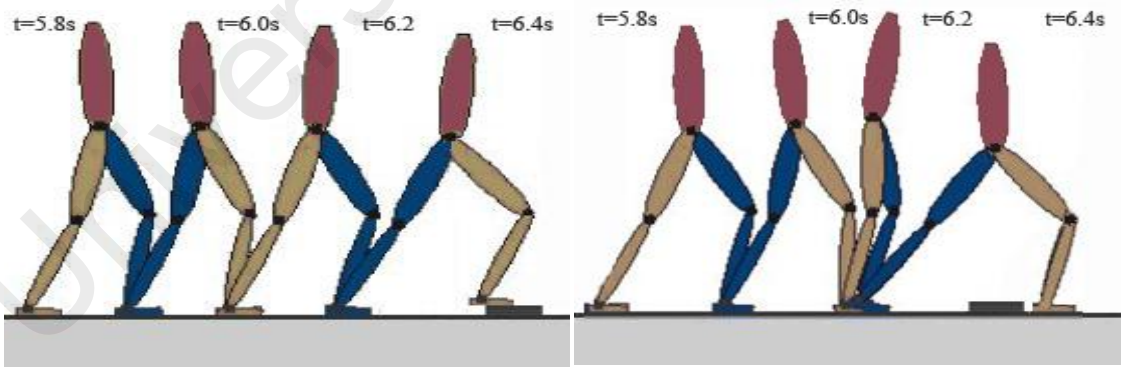


Figure 6.29: Joints trajectories during voluntary perturbation

Figure 6.28 shows phase plane for ankle, knee and hip of biped robot stepping over an obstacle comparison with normal walking pattern which the blue lines shows the normal walking pattern and the red line shows the walking step over an obstacle. Figure 6.29 shows the joint trajectories within a period of time.



(a) Without step length adaption **(b)** With step length adaption

Figure 6.30: Snapshot of comparative study of footstep planning results

This simulation is done on the base of Adams software with Matlab controller. Consequently, these rules are used in order to design the gait pattern based on

Hierarchically Clustered Fuzzy CMAC (HCFCMAC). Figure 6.30 shows the comparative study of the footstep planning results.

Figure 6.30 (a) concerns a control strategy without step length adaptation, and the figure 6.30 (b) shows the biped robot which uses the proposed control strategy. It is can be seen very easily that the robot without adaptation crash into obstacle when $t = 6.4s$, but when the length of the step is adjusted by using the Semi online Fuzzy Q-learning footstep planning, the robot is able to avoid the obstacle.

6.5 Summary

In this chapter, a control strategy for biped robot walking in dynamic environment is proposed, in which the robot and obstacle move with opposite direction in the sagittal plane. The inspiration of our approach is that we design the high-level and the low-level control separately. The high level control is composed of three parts:

- footstep planning
- foot trajectory generating
- joint angle generating.

The footstep planning is derived based on a semi online fuzzy Q-learning concept. The step length, step duration time and maximum step height, which in fact, determines the landing point and shape of foot trajectory, shall be obtained after the learning phase. Cubic spline interpolation has been applied to interpolate the designated point such as starting point, landing point and maximum step height to get the foot trajectory of swinging leg. With inverse kinematics, the curve of hip and knee angle of swing leg has been calculated according to the geometrical relationship between pitch angles.

The low-level control allows to approximate the joint trajectories and to control the tracking of these desired trajectories. The Hierarchically Clustered Fuzzy CMAC has been applied to approximate the desired joint angles.

The presented results show that our approach is operational in the case of a robot without feet moving in a sagittal plane and must consider. The proposed approach does not require the knowledge of probability transitions from a state to another for robot and has fast learning property. However, it is very important to point out that the robot can't step over the dynamic obstacle in all cases in our approach. The success rate is related to the size of obstacle. The longer the obstacle is the smaller the success rate. Because of the landing position of swing leg, there are also some conditions that no matter how the maximum height of last step is adjusted, the robot will always crash with the obstacle. The smaller the obstacle height is the bigger success rate the robot can step over.

CHAPTER 7

Conclusion and Implication of Future Direction

7.0 Conclusion and Implication of Future Direction

7.1 Conclusion

Designing autonomous intelligent control systems for real-world problems is a daunting task. The complex input-output relationships resulting from the interaction between a process and its environment are often not readily solvable by traditional mathematical methods. A growing amount of research is being performed in designing control systems which develop their own solution by utilizing methods borrowed from intelligent control.

CMAC is a neural network with local generalization abilities and can achieve convergence rapidly compared with other neural networks. Because of the advantages of simplified and effective training properties, fast learning convergence and digital hardware implementation, CMAC neural network is widely applied in the complex dynamic systems, such as robots. Usually the required memory size is dramatically increased with its input dimension. However, the memory size and computation time are important for the real-time control of these systems.

As a very important branch of reinforcement learning, Q-learning approach has been widely used in dynamic systems, because of its simple computations per time step and also because it has been proven to converge to a global optimum. Besides its advantage, a major problem with Q-learning is its inability to handle large state spaces. With larger state spaces, longer training times are required since multiple visits of each state action pair are required for the agent to learn and large state spaces also require impractically large amounts of memory. Therefore, we pay attention to Semi On-line Fuzzy Q-learning approach. It is developed to solve the problem of biped robot stepping over obstacles in dynamic environment.

We develop our research and obtain overall conclusion around the following points:

1) The main drawback of high-dimension input CMAC for its application on the real-time dynamic system is that the required memory size increases exponentially with the input dimension. Indeed the required memory size depends on its structure parameters. Our goal is to find an optimal CMAC structure for a given problem. The presented simulation results show that an optimal structure carrying a minimal modelling error could be achieved. Consequently, the choice of an optimal structure allows on one hand decreasing the memory size and on the other hand the computing time. Considering these two factors, the algorithm developed for optimizing the CMAC structure.

2) This research presents novel brain-inspired cerebellar model articulation controller architecture namely the Hierarchically Clustered Fuzzy Cerebellar Model Articulation Controller (HCFCMAC) network. Inspired by the physiology of cerebellum, the HCFCMAC network operates as an information-driven memory allocation scheme. The developed HCFCMAC stretches from the basic Fuzzy cerebellar model articulation controller by employing a hierarchical clustering technique to selectively allocate more memory cells to the input regions which include more training signal information as returned by the variations in the target output. The performance of the HCFCMAC was evaluated by control robotic arm. This modeling effort is a proof-of-concept demonstration of the utility of combining the two types of learning inspired by the cerebellar circuitry. According the simulation results in chapter 4 Hierarchically Clustered Fuzzy CMAC (HCFCMAC) has high accuracy in the learning and output.
3) In this research achievement, the footstep planning approach for biped robot proposed which could solve the problems of steeping over the obstacle. Furthermore, in order to avoid the high computational complexity, the planning has to be limited

within a few steps. We developed the footstep planning based on the semi online fuzzy Q-learning algorithm. The approach is divided into two separate semi online fuzzy Q-learning algorithm design processes, one is for the step length and step duration time action pair, and another is for the maximum step height. These three parameters determine the foot trajectory profile of swing leg. The main interests of this investigation are: (a) the computing time is very short (b) this approach is valid for both static and dynamic obstacles. (c) The footstep planning is operational for both predictable and unpredictable dynamical environment.

4) To study the locomotion of biped robot in the sagittal plan, one has to first develop the mathematical model of the robot. The kinematics equations are derived based on the five-link model. An advantage of this five-link model is that it has sufficiently appropriate number (not too many) of degrees of freedom to keep the equations of motion to a manageable level, while still having large enough degrees of freedom to adequately describe the walking motion. The dynamics equations during the single-support-leg phase are deduced from the Lagrangian formulation.

5) Based on the idea that the high-level control and low-level control can be designed separately, the control strategy for biped robot stepping over dynamic obstacle proposed. The high-level control consists of footstep planning which is based on semi online fuzzy Q-learning algorithm, foot trajectory generation and joint angle generation. The low-level control is based on HCFCMAC, allowing both to learn the predicted swinging trajectory and to control the tracking of these desired trajectories. The simulation results show that our approach is operational in the case of a robot without feet moving in a sagittal plane.

From the research findings the objectives are achieved. The 1st objective is to develop the new model of cerebellar model articulation controller (CMAC). After studying and

investigating on cerebellum connection and its function, a model of CMAC developed, which is named hierarchically clustered fuzzy CMAC and the results used to control robot arm that are discussed in chapter 4. Hence the 2rd objective to apply proposed model of CMAC to control robotic arm is met.

The 3th objective is to present a new concept of a footstep planning strategy that is based on Semi On-line Fuzzy Q-learning concept. The new method of Q-learning introduced which called Semi On-line Fuzzy Q-learning. One of the important advantages of this system is reducing computing time compared to other model of Q-learning. The proposed system is simulated in dynamic environment for control robot motion and avoiding the obstacle. The 4th objective is on obstacle avoidance strategy in dynamic environment based on mathematical model of biped robot. the mathematical model of five-link biped robot is computed and our proposed CMAC model and semi online fuzzy Q-learning algorithm used to control of biped robot for two types of controlling, high-level and low-level control. According to the results the function of HCFCMAC to predict the trajectory in the time of robot movement is achieved as one of the main advantages of artificial cerebellum is its capability to predict the next step of machine reaction.

7.2 Implication of Future Direction

The platform should set up and made several achievements about the CMAC and footstep planning for biped robot in dynamic environment; however, there is still insufficiency which needs further study. The real utmost power of the proposed model may only be realized if the model is efficiently scaled up to higher degree of freedom (DOF) problems, mobile robots and robust robots. In the footstep planning approach, the step length, step duration time and maximum step height have to be adjusted according to the velocity of obstacle, however the frequently change of these parameters

will influence the stability of biped walking robot. In the future, we will try to increase the robustness of our control strategy.

The HCFCMAC model can be used in the development of human blood pressure and heart attack prediction system for patient treatment as well as in the biomedical engineering domain such as surgical robot control and also can develop this model to be used in artificial limb to recover motion control in handicap people.

REFERENCES

REFERENCES

AAAI. (2013). Welcome to the Association for the Advancement of Artificial Intelligence. *AAAI Official Website*. 2013. Retrieved from http://www.aaai.org/home.html

Aggarwal, R. Sharan, A. and R Balasubramanian. (1997). Trajectory Control of Robotic Manipulators Using the Optimal Control Method. *CCECE 97 Canadian Conference on Electrical and Computer Engineering Innovation Voyage of Discovery Conference Proceedings*, 575-592.

AI. (2011). The History of Artificial Intelligence. *Oracle ThinkQuest*. Retrieved from http://library.thinkquest.org/2705/history.html

Albus, J. S. (1975). A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC). *In: Trans. ASME, Series G. Journal of Dynamic Systems, Measurement and Control*, 220-227.

Albus, J.S. (1971). A theory of cerebellar function. *Mathematical Bioscience*. 10: 25–61.

Albus, J.S. (1975). Data storage in the cerebellar model articulation controller (CMAC), Trans. ASME J. Dyn. Syst. Meas. Contr. (September), 228–233.

Albus, J. S. (1989). Marr and Albus theories of the cerebellum: Two early models of associative memory. *Thirty-Fourth IEEE Computer Society International Conference*, 577-582.

Alireza Ferdowsizadeh Naeeni, (2004). Advanced multi-agent fuzzy reinforcement learning, Master Thesis.

Aleksander Kołcz., Nigel M. Allinson. (1999). Basis function models of the CMAC network, *Journal of Neural Networks*, 107–126

Arthur, C Guyton. (1972). Structure and Function of the Nervous System. *Saunders WB Company.*

Arbib, Michael A. (2010), From Mirror Writing to Mirror Neurons, *Lecture Notes in Computer Science.* Volume 6226, 1-12

Arbib, M. (1964). Brains, Machines and Mathematics, *University of South California Neuroscience*. (1964). McGraw Hill. Retrieved from http://www.usc.edu/programs/neuroscience/faculty/profile.php?fid=16

Assad C (2001). A hypothesis for a novel learning mechanism in cerebellar cortex. *Journal of. Autonomous Robots*. 11: 285-290.

Asimov, I. (1950). "*I, Robot*". Gnome Press: New York. 1950.

Atkeson, C. Schaal, S. (1995). Memory-Based Neural Networks for Robot Learning. *Neurocomputing*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.7086

Bailón, W P et al. (2010). Mechanical energy optimization in trajectory planning for six DOF robot manipulators based on eighth-degree polynomial functions and a genetic algorithm. *Electrical Engineering Computing Science and Automatic Control CCE 2010 7th International Conference, 446-45.*

Bailey, R. (2013). *"Anatomy of the Brain – Cerebellum"* About.com. Biology. Retrieved from http://biology.about.com/od/anatomy/p/cerebellum.htm

Barto, A. G. (1995a). Reinforcement learning. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, The MIT Press, Cambridge, Massachusetts, 804–809.

Barto, A. G. (1995b). Reinforcement learning in motor control. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, The MIT Press, Cambridge, Massachusetts, 809–813

Becket, R. Chang, K. (1967). An Evolution of the kinematic gait by minimum energy. J Biomech, vol. 1

Berenji, H R. (1994). Fuzzy Q-learning: a new approach for fuzzy dynamic programming. *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, 486-491.

Borenstein, J, and Y Koren. (1990). Real-time obstacle avoidance for fast mobile robots in cluttered environments. *Proceedings IEEE International Conference on Robotics and Automation*, 572-577.

Carreras, M. Ridao, P. El-Fakdi, A. (2003). Semi-Online Neural-Q learning for Real-time Robot Learning. *Institute of Informatics and Applications; University of Girona*. Girona, Spain,

Carr, J. (2012). *"Artificial Cerebellum in Robotics Developed"*. Wired Cosmos. Retrieved from http://wiredcosmos.com/2012/07/03/artificial-cerebellum-in-robotics-developed/

Carreras, M. Ridao, P. El-Fakdi (2003). Semi-Online Neural-Q learning for Real-time Robot Learning. *IEEE RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS*; 1, 662-667.

Castro, Dotan Di, and Shie Mannor. (2010). Adaptive bases for Q-learning. *Decision and Control*. CDC 49th IEEE Conference on, 4587-4593.

Chen, M. Fu, Y. Ou, K. Chen, K. (2011).,Development of Navigation Schemes for Grouped Mobile Robots Leading to Biomimetic Applications,. *Department of Mechanical Engineering, National Cheng-Kung University, Taiwan.*

Chestnutt, Joel. (2007). Navigation Planning for Legged Robots. *Work* 2007, 1-37.

Chien-Kuo Li., Ching-Tsan Chiang. (2004). Neural Networks Composed of Single-variable CMACs, *2004 IEEE International Conference on Systems, Man and Cybernetics*, 3482-3487.

Chung Ying Amy Chan, (2000). Dynamic modeling, control and simulation of a planar five-link bipedal walking system,*Master thesis*, The University of Manitoba.

Chung, Yun-Kung Chung Yun-Kung, and Shing-Chung Hu Shing-Chung Hu. (2004). "An Alternative CMAC Trained with Genetic Algorithms to Solve an Instable Control Process." *2004 IEEE International Conference on Robotics and Biomimetics*.

Choi, C. (2007). ,Forecast: Sex and Marriage with Robots by 2050., *Khedo Live Science*, 12 Oct 2007. Retrieved from http://khedo.wordpress.com/2007/10/17/forecast-sex-and-marriage-with-robots-by-2050/

Chow, Mo-Yuen Chow Mo-Yuen, and A Menozzi. (1994). A self-organized CMAC controller., *Proceedings of IEEE International Conference on Industrial Technology ICIT 94* 1994, 68-72

C.-J. Lin, Y.J. Xu, The design of TSK-type fuzzy controllers using a new hybrid learning approach, *Int. J. Adaptive Control Signal Process*. 20 (1) (2006) 1–25.

C.-J. Lin, C.-Y. Lee, A novell parametric fuzzy CMAC network and its application , *Applied soft computing*. 9 (2009) 775-785.

C.M. Lin, and Y.-F. Peng, (2004). ,Adaptive CMAC-Based Supervisory Control for Uncertain Nonlinear Systems. *IEEE Trans. Systems, Man,and Cybernetics*, vol. 34, no. 2, 1248-1260.

CNS. (2007). *"*The Cerebellum as a Comparator*". CNS Clinic-Jordan*. Retrieved from http://www.humanneurophysiology.com/cerebellum.htm

Cobano, J A, J Estremera, and P Gonzalez de Santos. (2009). Accurate tracking of legged robots on natural terrain. *Autonomous Robots* 28.2, 231-244.

Craig JJ (1986).Introduction to Robotics.

Criveller, L. (2001). BCIs and mobile robots for neurological rehabilitation. *Department of Information Engineering.* University of Padova, Italy.

C. Sabourin, K. Madani and O. Bruneau. (2007) "Autonomous Biped Gait Pattern Based on Fuzzy-CMAC Neural Networks", *Integrated Computer-Aided Engineering (ICAE),* vol. Vol.14,. pp 1–14.

C.Sabourin, K.Madani, YU Weiwei, YAN Jie, (2008). Obstacle Avoidance Strategy for Biped Robot Based on Fuzzy Q-Learning Approach, *International Conference on Automatic Control & Robotics, Portugal*, 2008.09.

C.S. (2013). The Cerebellum (Motor Systems) Part 3. *What-When-How. Crankshift Publishing.* 2013. Retrieved from http://what-when-how.com/neuroscience/the-cerebellum-motor-systems-part-3/

C. Watkins, P. Dayan. (1992). Q-learning. *Journal of Machine Learning*, vol 8, 279–292.

Cybulsky J, Reed K. (2000). Requirements classification and reuse: crossing domains boundaries. *6th international conference on software reuse. Lecture Notes in Computer Science. Springer, Vienna*, 190–210.

Dave. (1972). Organization of SHRDLU. *Cognitive Psychology 3 No 1*. Academic Press. Retrieved from http://www.cs.cf.ac.uk/Dave/AI1/ailect15shrdlu4.gif

Dayan, P. and Watkins. C. (1992). The convergence of TD (λ) for general λ. *Machine Learning*, **8**, 341-362.

D.E.Goldberg. (1989). Genetic algorithms in search, optimization and machine learning, Boston, *MA, Kluwer Academic Publishers.*

El-Tantawy., S. Abdulhai., B. (2010). An agent-based learning towards decentralized and coordinated traffic signal control. *IEEE Int. Conference. On Intelligent Transportation Systems*, 665-670.

Farlex. (2013). Robota: Definition. [*Czech, from robota, drudgery; see orbh- in Indo-European roots*.] *The Free Online Dictionary. 2013*. Retrieved from http://www.thefreedictionary.com/robot

Fine EJ, Ionita CC, Lohr L., (2002). The History of the Development of the Cerebellar Examination. *Semin Neurol* 22 (4): 375–84.

Frederick G.H., (2005). Neural network control of a parallel hybrid-electric propulsion system for a small unmanned aerial vehicle, *PhD Thesis*, University of California.

Freud, S. (1932). The Ego and the ID. *W. W. Norton and Company*. The Standard Edition. *1990*. ISBN 978-0393001426

F.H.Glanz., T.W.Miller., and L.G.Kraft. (1991). *An overview of the CMAC neural network, IEEE Conf. on Neural Networks for Ocean Engineering*, 301-308

Fujiwara, Y, and S Adachi. (2004). Control system design for obstacle avoidance system using model predictive control. *IEEE International Conference on Control Applications*, 1591-1596.

García-Morales, V et al. (2009). Locomotion and gait disorders. *Revista de Neurologia* 48 Suppl 1: 71-78.

Garrido, Silvia, and Vidal (2012) Researchers Develop an Artificial Cerebellum than Enables Robotic Human-like Object Handling. *CanalUGR*. 2012. Retrieved from http://canal.ugr.es/information-and-communication-technologies/item/58501-researchers-develop-an-artificial-cerebellum-than-enables-robotic-human-like-object-handling

Glorennec, P Y. (1994). Fuzzy Q-learning and dynamical fuzzy Q-learning. *IEEE 3rd International Fuzzy Systems Conference*, 474–479.

Goodwin GC, Graebe SF, Salgado ME (2001). Control system design. (Prentice-Hall).

Grabianowski, B. (2013). How Brain- Computer Interfaces Work. *Computers: How Stuff Works*. 1998-2013. Retrieved from http://computer.howstuffworks.com/brain-computer-interface.htm

Gu, Dongbing Gu Dongbing, and Huosheng Hu Huosheng Hu. (2004). Accuracy based fuzzy Q-learning for robot behaviours, *IEEE International Conference on Fuzzy Systems IEEE Cat No04CH37542*, 1455-1460.

Hackel. M.Humanoid Robots: Human-like Machines. ITech Education and Publishing, Vienna, Austria, June 2007.

Hahn-Ming Lee., Chih-Ming Chen., and Yung-Feng Lu. (2003). A Self-Organizing HCMAC Neural-Network Classifier, *IEEE Transactions on Neural Networks*, Vol. 14, No. 1. 15-27.

Harris, G F, and J Wertsch. (1994). Procedures for gait analysis. *Arch Phys Med Rehabil* 75.2, 216-225.

Hauser, K et al. (2008). Motion Planning for Legged Robots on Varied Terrain. *The International Journal of Robotics Research* 27.11-12, 1325-1349.

He, Luo, Kong, and Du. (2009). *Trajectory Planning, Optimization and Control of a Hybrid Mechanical Press Chinese Academy of Sciences.*

Hodges, A. (2007). "*Alan Turing*" Stanford Encyclopedia of Philosophy. 2007. Retrieved from http://plato.stanford.edu/entries/turing/#BuiBra

H.Hemami, and R.L., (1977). Postural and gait stability of a planar five link biped by simulation, *IEEE Transactions on Automatic Control*, 1977, Vol.22, No.3, 452-458.

H.Hememi, and P.C.Camana, (1977). Nonlinear feedback in simple locomotion systems, *IEEE Transactions on Automatic Control*, 855-860.

H.Hemami and C.Jr.Golliday, (1977). The inverted pendulum and biped stability, *Journal of Mathematical Bioscience*, Vol.34, 95-110.

H.Hemami, C.S.Robinson and A.Z.Ceranowicz, (1980). Stability of planar biped models by simultaneous pole assignment and decoupling, International. Journal of Systems Science, Vol.11, No.1, 65-75.

Hinton, E. Rumelhart, D. Williams, R. (1986). Learning representations by back-propagating errors. *Nature* 323.6088, 533-536.

Huang, H. Yan, J. (2002). A fast smooth walking pattern generator for biped robots *Biped Robots, Prof. Armando Carlos Pina Filho* ISBN: 978-953-307-216-6, InTech, DOI: 10.5772/14875.

Hu, Lingyun Hu Lingyun, Changjiu Zhou Changjiu Zhou, and Zengqi Sun Zengqi Sun. (2006). Estimating Probability Distribution with Q-learning for Biped Gait Generation and Optimization. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 362-367

Hwang, Chih-Lyang, Chin-Yan Shih, and Ching-Chang Wong. (2007). "Distributed active-vision network-space approach for trajectory tracking and obstacle avoidance of a car-like mobile robot." *IEEE International Conference on Fuzzy Systems Vols 14*. Institute of Electrical and Electronics Engineers Inc, 159-164

Jalali A., Roziati Zainuddin and Pegah M. (2012). Control of a manipulator kinematics robotic arm on fuzzy cerebellar model articulation controller. *International Journal of Physical Sciences* Vol. 7(23), 2944-2951.

Jiang, Hong et al. (2011). Realization of the adaptive excitation control method with biped walking robot simulation system. *International Conference on Machine Learning and Cybernetics*, 1296-1301

Jiang, Zhao-Hui Jiang Zhao-Hui. (2004). End-effector robust trajectory tracking control for flexible robot manipulators. *IEEE International Conference on Systems Man and Cybernetics IEEE Cat No04CH37583*, 4394-4399.

Jianmin, Liu Jianmin Liu et al. (2010). The design on CMAC Neural network based on FPGA. *Computational Problem-solving ICCP 2010 International Conference*, 446-449.

John McCarthy. (2012). Soylent Communications. Retrieved from http://www.nndb.com/people/006/000030913/

Jordan W. H., Hall D. G., Young J. K., Hyten M. J. (2007). Practical rat neuropathology. *Journal of Histotechnol 30*, 115–120.

J. S. Albus, Marr and Albus theories of the cerebellum: Two early models of associative memory, *Proceedings of IEEE Compcon*, 1989.

J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Trans*.

Syst. Man Cyber. 23 (1993) 665–685.

Kajita, S, et al. (2003). Biped walking pattern generation by using preview control of zero-moment point. *IEEE International Conference on Robotics and Automation Cat No03CH37422* 2.2, 1620-1626.

Kandel, E R, J H Schwartz, and T M Jessell. (2000). *Principles of Neural Science", 4th edition*. McGraw-Hill, 2000.

Kanniah, J. Lwin, Z., Kumar, D., Fatt, N. (2004) A'ZMP' management scheme for trajectory control of biped robots using a three mass model. *2nd International Conference on Autonomous Robots and Agents* December 13-15, 2004 Palmerston North, New ZealandKinser,

Khatib, O. (1986). *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research* 5.1, 90-98.

Kiguchi, K., Watanabe, K., and etc.,(2003). A humanlike grasping force planner for object manipulation by robot manipulators, *Cybernetics and Systems*, Vol.34(8), pp.645-662.

Kim, Y.H., Lewis, F.L. and Dawson, D.M., (2000). Intelligent optimal control of robotic manipulators using neural networks, *Automatica*, Vol.36, pp.1355-1364.

Kim, Kim, Kim, & Kim. (2010). Implementation of a Wireless Haptic Controller for Humanoid Robot Walking Advances in Haptics. ISBN: 978-953-307-093-3, InTech, DOI: 10.5772-8690

Kim, J-H, J-Y Kim, and Jun-Ho Oh. (2011). Adaptive walking pattern generation and balance control of the passenger-carrying biped robot, HUBO FX-1, for variable passenger weights. *Autonomous Robots*, 427-443.

Kim, TKT, EKE Kim, and J-WKJ-W Kim. (2008). Development of a humanoid walking command system using a wireless haptic controller. *International Conference on Control, Automation and Systems*, 1178-1183.

Kim, J-J, Y-J Kim, and Ju-Jang Lee. (2011). A machine learning approach to falling detection and avoidance for biped robots. *SICE Annual Conference*, 562-567.

Kinser, A. (2013). Brain Structures and their Functions. *Serendip*. 2013. Haverford College. Retrieved from http://serendip.brynmawr.edu/bb/kinser/Structure1.html#cerebellum

Kolcz, A, and Nigel M Allinson. (1999). Basis function models of the CMAC network. *Neural Networks*, 107-126.

Krueger, C. W. (1992). Application-specific object management architectures. Ph.D. dissertation, Carnegie-Mellon Univ.

Kuffner, J. Nishiwaki, K. Kagami, S. Inaba, M. Inoue, H. (2001). Footstep Planning among Obstacles for Biped Robots *IEEERSJ International Conference on Intelligent Robots and Systems Cat No01CH37180* 2001, 500-505

Kunimatsu, S et al. (2008). preview control for biped walking pattern generation. *SICE Annual Conference*, 1916-1919.

Kyong-Sok Chang, (2000). Efficient algorithms for articulated branching mechanisms: dynamic modeling, control and simulation, PH.D.thesis.

Larson, Ron., Farber, Betsy. (2003). ElementaryStatistics. *208 Pages, Published 2003 by Prentice Hall*

Legacy. (2011). *Karel Kapek – Beyond the Robots Legends & Legacy; Legacy.com*. Retrieved from http://www.legacy.com/ns/news-story.aspx?id=213

Liguo, Huo. (2009). Robotic joint-motion optimization of functionally-redundant tasks for joint-limits and singularity avoidance, PhD thesis, 2009, 04.

Lim, Heonyoung Lim Heonyoung et al. (2008).Nonlinear Model Predictive Controller Design with Obstacle Avoidance for a Mobile Robot*IEEEASME International Conference on Mechtronic and Embedded Systems and Applications*, 494-499.

Lim, Hun-ok Lim Hun-ok, Y Kaneshima, and A Takanishi. (2010). Online walking pattern generation for biped walking. *Control Automation and Systems ICCAS 2010 International* Conference, 3111-3116.

Lin, Xu-Mei Lin Xu-Mei., Tao Mei Tao Mei. and Hui-Jing Wang Hui-Jing Wang. (2005). Neural optimal control of robotic manipulators with CMAC networks. *IEEE International Conference Mechatronics and Automation*, 1391-1396.

Lin, S. Wright, R. (2010). Evolutionary Tile Coding: An Automated State Abstraction Algorithm for Reinforcement Learning. *Air Force Research Laboratory Information Directorate*. Rome, NY. 2010.

Liu, HLH et al. (2003). Robot manipulator controller based on fuzzy neural and CMAC network. *Proceedings of the International Conference on Machine Learning and Cybernetics IEEE Cat No03EX693*, 525-529

Marr, D. (1969). A theory of cerebellar cortex. *Journal of Physiology of London*, vol. 202, 437–470.

Manish Kumar, (2004), Sensor fusion and intelligent control of autonomous cooperating robots, *PhD. Thesis*.

McClelland, J. Rumelhart, D. Hinton, G. (1986). The Appeal of Parallel Distributed Processing. *Ed. D E Rumelhart & J L McClelland Processing*, 3-44.

McKay, B.E., Engbers, J.D.T., Mehaffey, W.H., Gordon, G., Molineux, M.L., Bains, J. and Turner, R.W. (2007). Climbing fiber discharge regulates cerebellar functions by controlling the intrinsic characteristics of Purkinje cell output. *J. Neurophysiology* 97, 2590-2604.

Mericli, C. Veloso, M. (2010). Biped Walk Learning through Playback and Corrective Demonstration. *Proceeding of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 194-205.

Miall, RC., Weir, DJ., Wolpert, DM & Stein, JF. (1993). Is the cerebellum a Smith Predictor? *Journal of Motor Behaviour*. 25, 203-216.

Miall, RC., Wolpert, DM. (1996). Forward models for physiological motor control. *Journal of Neural Network* 9: 1265-1279.

Miller, T. Glanz, F., Kraft, L. (1990). CMAC is an associative neural network alternative to back propagation. *Mendeley: Computer and Information Science*. Volume 78, Issue 10, IEEE,1561-1567.

Ming-Feng Yeh., and Kuang-Chiung Chang. (2006). A Self-Organizing CMAC Network With Gray Credit Assignment, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 36, No.3, 623-635.

Miller, W T Iii. (1994). Real-time neural network control of a biped walking robot. *Control Systems IEEE*, 41-48.

Miller, W. Glanz, F. Kraft, L. (1987). Application of a general learning algorithm to the control of robotic manipulators. *CiteSeer*. 1987. Retrieved from http://citeseerx.ist.psu.edu/showciting?cid=236206

M.Mitchell, (1996). An introduction to genetic algorithms, *Cambridge, MA, MIT Press*, 1996.

Mohajeri, K et al. (2009). Fuzzy CMAC structures. *IEEE International Conference on Fuzzy Systems.* 2126-2131

Morisawa, M et al. (2001). A walking pattern generation for biped robot with parallel mechanism by considering contact force. *IECON01 27th Annual Conference of the IEEE Industrial Electronics Society Cat No37243* 2001, 2184-2189.

Murphy, Susan A. (2005). A Generalization Error for Q-Learning. *Journal of machine learning research JMLR* 6.3, 1073-1097.

M.W.Walker, D.E.Orin, (1982). Efficient dynamic computer simulation of robotic mechanisms, Transactions of ASME *Journal of Dynamic Systems, Measurement, and Control. Vol104(3),* 205-211.

Nakayatna, H, and K Watanabe. (2009). Improved walking motion for a biped robot using piezoelectric ceramic sensors. *ICCASSICE* 2009. 1218-1221.

Nearchou, A.C. and Aspragathos, N.A. (1996). Application of genetic algorithms to point-to-point motion of redundant manipulators, *Mechanism and Machine Theory, vol 31, no. 3,* 261-270.

Nearchou, A.C. (1998). Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism & Machine Theory,* vol. 33, no. 3, 273-292

Nelson M E, Paulin M G (1995). Neural simulations of adaptive reafference suppression in theelasmobranch electrosensory system, *J. Comp. Physiol*. A 177: 723-736.

Newell, A. (1994). Unified Theories of Cognition. *Harvard University Press*, 1994.

Nicolas-Alonso, L. Gomez-Gil, J. (2012). Brain Computer Interfaces, a Review. *National Center for Biotechnology Information, U.S. National Library of Medicine.*

Nitin Mathur, Design of intelligent adaptive control using immune based algorithm, *Master Thesis*, 2004

Olli Haavisto and Heikki Hyotyniemi, (2004). Simulation tool of a biped walking robot model, *Helsinki University of Technology Control Engineering Laboratory, Report.*

O. Haavisto. (2004). Master's thesis, *Helsinki University of Technology*.

Oommen, B J, S Iyengar, and N Andrade. (1988). On using stochastic automata for trajectory planning of robot manipulators in noisy workspaces. *The Fourth Conference on Artificial Intelligence Applications*, 135-148.

Oyama, T, and Y Uno. (2004). Estimation of a human planned trajectory from a measured trajectory. *IEEE International Conference on Systems Man and Cybernetics. IEEE Cat No04CH37. 583, 728-734.*

Oussarna Khatib. (1987). A unified approach to motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation* , Vol3(1), 43-53.

Overskeid Geir. (2007). Looking for skinner and finding Freud. *American Journal of psychologist. Vol 62, No 6,* 590-595.

Pawel, M.P. (2007). *Fuzzy anticipatory learning classifier system for mobile robot navigation*, Master Thesis, University of Alberta, 2007

Peck, M. (2011). Talk to the Hand. *Popsci*. Retrieved from http://www.popsci.com/technology/article/2011-02/talk-hand-new-interface-bionic-limbs

Percival, C. (1977). Semi classical theory of Bound States. *Advances in Chemical Physics*. Ed. *I Prigogine & Stuart A Rice. John Wiley & Sons, Inc.*, 1977. 1.

Phuong, N T et al. (2008). An optimal control method for biped robot with stable walking gait. *Humanoids 8th IEEERAS International Conference on Humanoid Robots*, 211-218.

Ping, Guoxiang Ping Guoxiang et al. (2009). Real time obstacle avoidance for redundant robot. *International Conference on Mechatronics and Automation*, 27-238

Poole, D. Mackworth, A. (2010). Q-learning Artificial Intelligence. *Foundations of Computational Agents*. Cambridge University Press.

Righetti, L. and Schaal, S. (2012). Quadratic programming for inverse dynamics with optimal distribution of contact forces, *IEEE-RAS International Conference on Humanoid Robots*, 538-543

Rossini PM., Noris Ferilli MA., Ferreri F. (2012). Cortical plasticity and brain computer interface. *Europian Journal of Physic Rehabil Med* 48, 307–312.

Sabourin, C. Bruneau, O. (2005). CMAC Neural Network "Robustness of the dynamic walk of a biped robot subjected to disturbing external forces by using CMAC neural networks"**.** *Robots and Autonomous Systems*.. Retrieved from http://www.sciencedirect.com/science/article/pii/S09218890050002

Sabourin, C. Yu, W. Madani, K. (2012). ,Gait Pattern based on CMAC Neural Network for Robotic Applications, *Springer Science and Business Media.* New York, 2012.

Saramago, S F P, and V Steffen Jr. (1998). Optimization of the Trajectory Planning of Robot Manipulators taking into account the Dynamics of the System. *Mechanism and Machine Theory*, 883-894.

Saleh, M, and G Murdoch. (1985). In Defense of Gait Analysis: Observation and measurement in gait assessment. *The Journal of bone and joint surgery British volume* 67.2, 237-241.

Schweighofer N, Doya K, Lay F. (2001). Unsupervised learning of granule cell sparse codes enhances cerebellar adaptive control. *Journal of. Neuroscience*. 103(1), 35–50.

Shimmyo, S, and K Ohnishi. (2011). Nested preview control by utilizing virtual plane for biped walking pattern generation including COG up-down motion. *36th Annual Conference on IEEE Industrial Electronics Society*, 1571-1576.

Shimon A.W., (2007), Adaptive representations for reinforcement learning, *PhD Thesis*, University of Texas.

Shuangman, Xu, and Zhang Ying. (2011). Study on trajectory control for six-DOF joint robot. *International Conference on Electronics Communications and Control ICECC*, 3958-3961

Sniezyński, B et al. (2010). Combining Rule Induction and Reinforcement Learning: An Agent-based Vehicle Routing. *Ninth International Conference on Machine Learning and Applications*, 851-856.

Sutton, R S, and A G Barto. (1998). Reinforcement Learning: An Introduction. Ed. Press-Bradford Books. *IEEE Transactions on Neural Networks*, 322p, 172-179.

Suzuki, T. Takahashi, M. (2011). Obstacle Avoidance for Autonomous Mobile Robots Based on Position Prediction Using Fuzzy Inference, *InTech* ISBN 978-953-307-389-7, Published: September 9, 2011

S.Tzafestas, M.Raibert and C.Tzafestas, (1996). Robust sliding-mode control applied to a 5-link biped robot, *Journal of Intelligent and Robotic Systems*, Vol.15, 67-133.

S. Weaver, L. Baird, and M. Polycarpou. (1998). An Analytical Framework for Local Feedforward Networks. *IEEE Transactions on Neural Networks*, 9(3). 473-482.

Tang, Zhe Tang Zhe, Changjiu Zhou Changjiu Zhou, and Zenqi Sun Zenqi Sun. (2002). Trajectory planning for smooth transition of a biped robot. *7th International Workshop on Advanced Motion Control Proceedings*, 2455-2460.

Teddy, S., Lai, M. & Quek, C. (2007). Hierarchically Clustered Adaptive Quantization CMAC and Its Learning Convergence. *In Proceedings of IEEE Transactions on Neural Networks*, 1658-1682.

Uchiyama, Masaru, and Zhao Hui Jiang. (1991). Compensability of End-Effector Position Errors for Flexible Robot Manipulators. *American Control Conference*, 87-115.

Uchiyama, N et al. (2009). Sound source tracking and obstacle avoidance for a mobile robot. *IEEE Conference on Emerging Technologies Factory Automation*, 1057-1064.

Walker, Richard. (2002). *"Secret Worlds: Brain."* DK Children.

Wang, Fei et al. (2010). Walking gait generation using linear inverted pendulum model for biped robot with heterogeneous legs. *Control and Decision Conference CCDC Chinese*, 1326-1329.

Watanabe, T. (2009). Implementation of fuzzy Q-learning based on modular fuzzy model and parallel structured learning. *IEEE International Conference on Systems, Man and Cybernetics*, 1338-1344.

Weaver, L.C. and Krenz , N. (1998). Changes in the morphology of sympathetic preganglionic neurons parallel the development automatic dysreflexia after spinal cord injury in rat, *Neurosci. Let., 243*, 61-64

Wei, Lixin Wei Lixin, Hongrui Wang Hongrui Wang, and Ying Li Ying Li. (2004). Robust trajectory control for robot. *Fifth World Congress on Intelligent Control and Automation IEEE Cat No04EX788* 2004. 4927-4929.

Wen, Chen., (2002). *Recurrent neural networks applied to robotic motion control*, Master Thesis.

William W. Cohen. (1994). Incremental abductive EBL. *Machine Learning* 15(1), 5-24.

Winograd. (1971). *"SHRDLU"* MIT AI Technical Report 235" Stanford Education. Retrieved from http://hci.stanford.edu/~winograd/shrdlu/

Wisse, M, C G Atkeson, and D K Kloimwieder. (2005). Swing leg retraction helps biped walking stability. *5th IEEERAS International Conference on Humanoid Robots,* 295-300.

Woergoetter, F. Porr, B. (2008). "Reinforcement Learning". *University of Goettingen, Germany*. Retrieved from http://www.scholarpedia.org/article/Reinforcement_learning

Wu, T. Wang, L. Chang, F., Tsai, P. (2006). Multivariable Adaptive Fuzzy CMAC Control for a Class of Nonlinear Systems. *American Control Conference,* 11-13

Xiaolong, Dai Xiaolong Dai et al. (1998). Optimal design and optimal algorithm of fuzzy CMAC. *Proceedings of the 37th SICE Annual Conference International Session Papers.* 797-802

Xiaopong, Guan, and Wang Jidong. (2011). Trajectory planning theory and method of industrial robot. *3rd International Conference on Computer Research and Development*, 340-343.

Xu, Xiang-Rong Xu Xiang-Rong, Won-Jee Chung Won-Jee Chung, and Young-Hyu Choi Young-Hyu Choi. (2000). A method for trajectory planning of robot manipulators in Cartesian space. *Proceedings 1999 IEEE,* 1220-1225.

Yang, S.X. and Meng, M.Q.-H. (2003), Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach, *IEEE Trans. on Neural Networks,* Vol.14(6), pp.1541-1552.

Yeh, Ming-Feng Yeh Ming-Feng, and Hung-Ching Lu Hung-Ching Lu. (2002). On-line adaptive quantization input space in CMAC neural network. *IEEE International Conference on Systems Man and Cybernetics*. 457-483.

Yu, Weiwei et al. (2011). CMAC STRUCTURE OPTIMIZATION WITH Q-LEARNING. *International Conference on Neural Computation Theory and Applications NCTA*. 283-288.

Zeman, V., Patel, R.V. and Khorasani, K (1997), Control of a flexible joint robot using neural networks, *IEEE Trans. On Control Systems Technology,* Vol.5, pp. 453-462.

Zhou SM, Xu LD (2001). A new type of recurrent fuzzy neural network for modeling dynamic systems. *Journal of. Knowledge-Based Systems*. 14, 243-251.

Zong, Guanghua Zong Guanghua, Luhua Deng Luhua Deng, and Wei Wang Wei Wang. (2006). A Method for Robustness Improvement of Robot Obstacle Avoidance Algorithm. *IEEE International Conference on Robotics and Biomimetics*. 115-119.

List of Publication

1) **Alireza Jalali***, Roziati Zainuddin and Pegah Mohammadi. (2012). Control of a manipulator kinematics robotic arm on fuzzy cerebellar model articulation controller. International Journal of Physical Sciences Vol. 7(23), pp. 2944-2951, (*ISI-Cited Publication*)

2) Saeed Aghabozorgi, Teh Ying-Wah, Hamid Jalab, Mohammad Amin Shayegan, and **Alireza Jalali**, A Hybrid Algorithm for Clustering of Time-series Data based on Affinity Search Technique, The Scientific World Journal, 2014 (*ISI-Cited Publication*)

3) **Alireza Jalali***, Woo Chaw Seng, saeed aghabozorgi, (2014). Computational model of cerebellum acting as adaptive controller for biped robot. International Journal of computer technology and applications (Accepted)

4) **Alireza Jalali**, Roziati Zainuddin, Woo Chaw Seng, Pegah Mohammadi,"An Artificial Cerebellum to Control Motion", Malaysian Science and Technology Congress (MSTC 2010). Nov 2010.

5) **Alireza Jalali**, Roziati Zainuddin, Woo Chaw Seng. "Development of an artificial model of cerebellum for learning robot motion control", 2th International Conference on Signals, Systems and Automation, Gujarat, India. Jan 2011.

6) Pegah Mohammadi, Nazean Jomhari, **Alireza Jalali**,. "Application of Artificial Neural Network in compressive strength prediction of Lightweight Concrete with various percentage of scoria instead of sand ", Malaysian Science and Technology Congress (MSTC 2010). Nov 2010.