

Thèse

Présentée pour l'obtention des titres de
DOCTEUR DE L'UNIVERSITÉ PARIS-EST
DOCTOR OF PHILOSOPHY

Spécialité: Sciences Informatiques

par **Weiwei YU**

Contribution à l'étude et à la mise en œuvre de stratégies adaptatives de commandes intelligentes : application au contrôle de systèmes dynamiques complexes

Soutenue publiquement le 23 Février 2011 devant la commission d'examen composée de

Prof.	Yuanying	QIU	Rapporteur / Xidian University
Prof.	Tianshi	LI	Rapporteur / Xi'an Jiaotong University
Prof.	Xiansheng	QIN	Examineur / Northwestern Polytechnical University
Dr.	Christophe	SABOURIN	Examineur / University PARIS-EST Créteil
Prof.	Jie	YAN	Codirecteur de thèse / Northwestern Polytechnical University
Prof.	Kurosh	MADANI	Codirecteur de thèse / University PARIS-EST Créteil

Thesis

Presented to obtain the title of
DOCTOR OF UNIVERSITY PARIS-EST
DOCTOR OF PHILOSOPHY

Specialization: Computer Sciences

by **YU Weiwei**

**Contribution to study and implementation of intelligent adaptive control
strategies: application to control of complex dynamic systems**

Defended on 23 February 2011 in presence of commission composed by

Prof.	Yuanying	QIU	Reviewer / Xidian University
Prof.	Tianshi	LI	Reviewer / Xi'an Jiaotong University
Prof.	Xiansheng	QIN	Examiner / Northwestern Polytechnical University
Dr.	Christophe	SABOURIN	Examiner / University PARIS-EST Créteil
Prof.	Jie	YAN	PhD Co-Supervisor / Northwestern Polytechnical University
Prof.	Kurosh	MADANI	PhD Co-Supervisor / University PARIS-EST Créteil

Acknowledgements

I have benefited from the advice, friendship, and support of quite a number of people throughout my years at Paris Est University and Northwestern Polytechnical University, and my work on this thesis. First and foremost, my advisor Professor Kurosh Madani's guidance and encouragement as my co-advisor provided me with a solid advance control theory background and inspiration, even though we speak different language. I also thank him for his hard working which help with very trivial staffs, like registration and so on. My co-advisor Professor YAN Jie has provided invaluable intellectual and financial support, without which this thesis may never have reached completion. I must thank Christophe Sabourin for his comments and assistance with my research work. Without him, I can not find the direction of my research. His patient and big efforts help me a lot with the completion of my thesis. I would also like to thank Prof. Wang Runxiao and Prof. Qin Xiansheng, for they give me enough space and good environment which support me finishing my thesis.

I own a great deal to the friendship, intelligence, and patience of my companions and colleagues both within Paris Est University and Northwestern Polytechnical University: Zhong Dudu, Lu Cunkan, Chang Xiaofei, Yan Binbin, Zhang Juanjuan, Bai Xuxu, Mei Jinna, Sofian, Arash and Dalel. In particular, many thanks to Ivan and Nadia for their company during my stay in France. With them, I spent quite interesting and wonderful time in Lieusaint.

I would like to thank my family for their unlimited patience and persistence. I will finally be able to speak to them without answering the question "Are you done yet?"

Last but not least, I thank my husband, Chang Jian, for his endless patience, understanding, and encouragement. I look forward to spending more time with him.

Contents

Acknowledgements.....	1
Contents.....	2
Abstract.....	5
Chapter 1 Introduction.....	7
1.1 Intelligent control.....	7
1.2 Neural network.....	10
1.3 Reinforcement learning.....	12
1.4 Research summary and dissertation overview.....	14
Chapter 2 CMAC Neural network.....	16
2.1 Introduction.....	16
2.2 CMAC structure.....	17
2.2.1 One dimension CMAC structure.....	18
2.2.2 High dimension CMAC structure.....	18
2.3 Parameters influence of CMAC ANN for function approximation.....	19
2.3.1 Structure parameters of CMAC ANN.....	19
2.3.2 Impact of input dimension on required memory size.....	20
2.3.3 CMAC training experimental protocol.....	21
2.3.4 Structural parameters' influence.....	23
2.4 Structure optimization.....	26
2.5 Summary.....	28
Chapter 3 CMAC based controller design for generic hypersonic vehicle.....	30
3.1 Introduction.....	30
3.2 Mathematic model of GHV.....	31
3.2.1 Overview the mathematic models of GHV.....	31
3.2.2 Nonlinear longitudinal equations of GHV.....	33
3.3 CMAC based controller.....	42
3.3.1 CMAC based controller design.....	43
3.3.2 Simulation results and analysis.....	44
3.4 Summary.....	46
Chapter 4 Fuzzy Q-learning.....	48
4.1 Introduction.....	48
4.2 Q-learning Approach for Path Planning.....	49
4.2.1 Q-Learning Algorithm.....	50
4.2.2 Path Planning.....	52
4.3 Fuzzy Q-learning algorithm.....	54
4.3.1 Fuzzy inference system.....	54
4.3.2 Fuzzy Q-learning algorithm.....	56
4.4 Step length planning for biped robot.....	57
4.4.1 Virtual dynamical environment.....	58
4.4.2 Fuzzy sensors.....	59
4.4.3 FQL-based step length.....	60

4.4.4 Reinforcement signal	60
4.4.5 Simulation results and analysis	61
4.5 Step duration time planning for biped robot	65
4.5.1 FQL-based step duration time planning	65
4.5.2 Simulation results and analysis	66
4.6 Maximum step height planning	68
4.6.1 FQL-based maximum step height planning	69
4.6.2 Simulation results and analysis	71
4.7 Summary	74
Chapter 5 Obstacle avoidance strategy for biped robot in dynamic environment.....	76
5.1 Introduction	76
5.2 Motion equations for five-link biped robot	78
5.2.1 The kinematics model of the five-link biped robot	79
5.2.2 The dynamic model of the five-link biped robot.....	82
5.2.3 Summary	91
5.3 Control strategy	92
5.3.1 High-level controller design.....	92
5.3.2 Low-level controller design	101
5.4 Simulation results and analysis	104
5.5 Summary	107
Chapter 6 Conclusion and Future Work.....	108
6.1 Conclusion	108
6.2 Future work.....	110
Appendix Simulation results of CMAC based controller for generic hypersonic vehicle	112
Reference	115
Publications.....	124

Abstract

As the fast pace of technology development in modern society, the devices surround us are especially independent on humans to give them instruction on how they should function. This is especially true of control of complex dynamic systems which operate extremely automatic functions and perform quite intricate tasks, from robots to aircrafts. Now days, artificial intelligence control is prosperous to simplify the control of these complex dynamic systems by allowing the devices to learn their own control functions. Through the use of intelligent systems, the devices and machines are capable of making human-like decisions on their own without having a human designer provide solution for every problem that could be encountered. This dissertation is focus on, firstly, the study of CMAC neural network and its application on hypersonic vehicle and biped robot, secondly, the study of Q-learning approach and its application on the biped robot's footstep planning problem, and the main work can be described in the following aspects:

1. The main limitation of the CMAC (Cerebellar Model Articulation Controller) network in realistic applications for biped robot is related to the required memory size. It is pertinent to remind that the memory used by CMAC depends firstly on the input signal quantification step and secondly on the input space dimension. For real CMAC based control applications, on the one hand, in order to increase the accuracy of the control the chosen quantification step must be as small as possible; on the other hand, generally the input space dimension is greater than two. In order to overcome the problem relating the memory size, how both the generalization and step quantization parameters may influence the CMAC's approximation quality has been discussed. Our goal is to find an optimal CMAC structure for a given problem. The presented simulation results show that an optimal or sub-optimal structure carrying out a minimal modeling error could be achieved. The choice of an optimal structure allows decreasing the memory size and reducing the computing time as well.

2. Flight control design for airbreathing hypersonic vehicles is a very challenging task due to its sensitivity to the changes of flight condition and the difficulty in measuring and estimating the aerodynamic characteristics of the vehicle. All of these are due to the integrated engine-airframe configuration. This configuration results in significant coupling between the structure, propulsion system and vehicle aerodynamics. An implementation of CMAC neural network for altitude and velocity tracking control of the longitudinal model of an airbreathing hypersonic vehicle which has an integrated airframe-propulsion system configuration has been proposed. Simulation results demonstrate the effectiveness of CMAC neural network control design in tracking altitude and velocity commands.

3. This paper presents a new concept of a footstep planning strategy, which is based on an improved fuzzy Q-learning concept, for biped robot in dynamical environment. For each rule, the learning agent has to choose action pair of step length and duration time, and the corresponding maximum step height as well. After the training phase, the biped robot is able to adapt the step length, step duration time and maximum step height at the same time only using a Fuzzy Inference

System, in order to step over obstacles with random velocity and height within certain range. In comparison with other previous works, the investigations show a real interest of this approach because: (1) Computing time is very short. After the learning phase, the footstep planning is based only on a FIS. (2) The footstep planning is operational for both predictable and unpredictable dynamical obstacles allowing the control system increase the robustness.

4. In contrasts with previous studies where the path planning is given by taking into account the feasibility of the joint trajectories, the originality of our approach on gait pattern planning and control strategy for biped robot stepping over dynamic obstacles, is that we consider it should be possible to design separately the high-level control and the low-level control. The goal of the high-level control is to anticipate the moving of the robot by using an exteroceptive perception of the environment. The computing time of this learning process is a crucial parameter for on-line control. Consequently, in order to decrease the computing time, the learning stage of proposed footstep planning which takes into account only the dynamic of the environment, is carried out to design the high-level control strategy. The structure of the low-level control including the gait pattern can be decomposed into three parts: (1) The first part is used to compute the trajectories of the swing leg from several outputs of CMAC neural networks and a Fuzzy Inference System. (2) The second one allows the regulation of the average velocity from a modification of the pitch angle of the trunk. (3) The third part is composed of four PD controllers in order to ensure the tracking of the reference trajectories at the level of each joint. The simulation results show the effectiveness of the proposed approach in the case of a flat dynamic obstacle.

Keywords Biped robot; Cerebellar model arithmetic computer neural network; Fuzzy Q-learning algorithm; Obstacle avoidance; Footstep planning;

Chapter 1 Introduction

1.1 Intelligent control

As the fast pace of technology development in modern society, the devices surround us are especially independent on humans to give them instruction on how they should function. This is especially true of control of complex dynamic systems which operate extremely automatic functions and perform quite intricate tasks, from consumer electronics, to factory equipment, to extraterrestrial probes, to aerospace vehicles and mobile robots. Nevertheless, the design and implementation of these control systems is considerably tanglesome, 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, 2005).

Now days, artificial intelligence and statistical methods are prosperous to simplify the control of these complex dynamic systems by allowing the devices to learn their own control functions. Through the use of intelligent systems, the devices and machines are capable of making human-like decisions on their own without having a human designer provide solution for every problem that could be encountered. Intelligent control is the discipline in which control algorithms are developed by emulating certain characteristics of intelligent biological systems such as human beings. The intelligent control uses various artificial intelligence computing approaches, such as neural networks, machine learning, evolutionary computation, Bayesian probability, fuzzy control and genetic algorithms.

Evolutionary computation

In the field of intelligent control, a main branch is in the area of evolutionary computation. This branch is developed based on the approach to “evolve” a solution to a given problem over successive generations by using methods observed in nature such as Darwinian natural selection, immune systems, swarm intelligence, self organizations, and further combinations of these techniques. In effect, these methods could be seen as performing a guided stochastic search when viewed from a classical artificial intelligence perspective. The benefits of applying evolutionary computation strategies on dynamic system is on one hand, the inherent massive parallelism that exists inside the many stages of evolutionary computing algorithms, and on the other hand,

allowing one to generate desirable intelligent systems with minimal human intervention (Pawel Maksymilian Pytlak, 2007).

Bayesian probability

Bayesian probability is a statistical data fusion algorithm based on Bayes's theorem of conditional or a posteriori probability to estimate an n-dimensional state vector X , after the observation or measurement denoted by Z has been made. It makes use of a priori knowledge about the observation space to make inference about the quantity of interest in that space. The probabilistic information contained in Z about X is described by a probability density function $p(Z|X)$, known as likelihood function, which is an objective function based on observation. If the information about the state X is made available independently before any observation is made, then likelihood function can be improved to provide more accurate results. Such a priori information about X can be encapsulated as the prior probability $P(X = x)$ and is regarded as subjective because it is not based on observed data (Manish Kumar, 2008). Bayesian probability theorem provides the posterior conditional distribution of $X = x$, given $Z = z$, as

$$p(X = x|Z = z) = \frac{p(Z = z|X = x)P(X = x)}{\int p(Z = z|X = x)P(X = x)dx} = \frac{p(Z = z|X = x)P(X = x)}{p(Z = z)} \quad (1.1)$$

One of the main disadvantages of Bayesian models, is the computational complexity of the posterior distribution. The use of conjugate priors requires little computational effort, and their simple parametrization is often a nice alternative to more complex models. In such a situation, the prior-likelihood pair forms a conjugate family, where the prior and posterior distribution have the same form (Aurelie Labbe, 2005).

Fuzzy control

Fuzzy control is one of the control techniques that pertain to the realization of intelligent control systems. Fuzzy control is derived from the fuzzy logic and fuzzy set theory introduced by L.A.Zadeh (1965). Fuzzy logic is a departure from the classical two-valued sets and logic that uses "soft" linguistic system variables and a continuous range of truth values in the interval [0,1]. Formally, fuzzy logic was a structured, model-free estimator that approximated a function through linguistic input/output associations. The typical structure of a fuzzy controller is presented in figure 1.1. Firstly, a real-world value is passed through a fuzzification stage where its belongingness to different fuzzy membership functions is assessed. Next, the fuzzified value is used to perform a composition of the different rules present in the fuzzy rule base by means of an inference engine. Finally, the composite function is defuzzified to give a resulting crisp output

which is then utilized by given application (Pawel M.P., 2007). The advantages of fuzzy logic control system are: it is applicable for the system where it is not possible to form a model of the process; it is able to handle continuous ranges without overcomplicated mathematical formulations and it is also capable to gracefully handle noisy inputs that normally exist in real-world systems without causing erratic behavior.

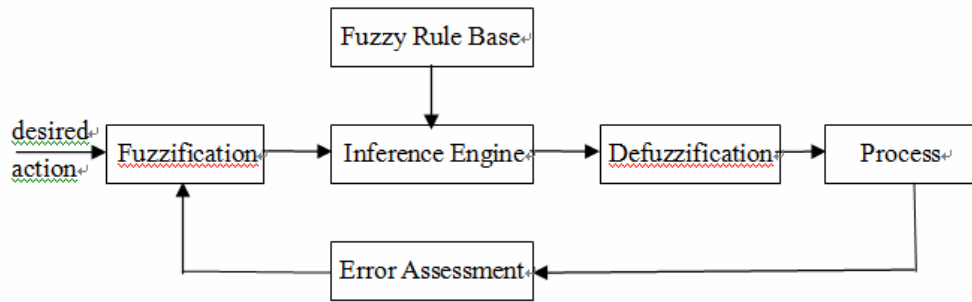


Fig. 1.1 Structure of a typical fuzzy controller (Pawel M.P., 2007)

Genetic Algorithms

Genetic Algorithms developed by Holland (1975) as a means to solve optimization problems, are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection and crossover (Liguo Huo, 2009). They accomplish the task through successive populations of candidate solution. During each iteration of the algorithm, GAs apply genetic operators modeled from nature. The resulting offspring populations are successively selected, recombined and altered to form an iteratively better solution to the problem (D.E.Goldberg, 1989, M.Mitchell, 1996). The scheme of basic genetic algorithm is represented in figure 1.2. 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 and point out that the advantage of using Genetic algorithm is allowing additional constraints to be easily specified, and working the variables represented as digital values that is more suitable for computer controlled dynamic system.

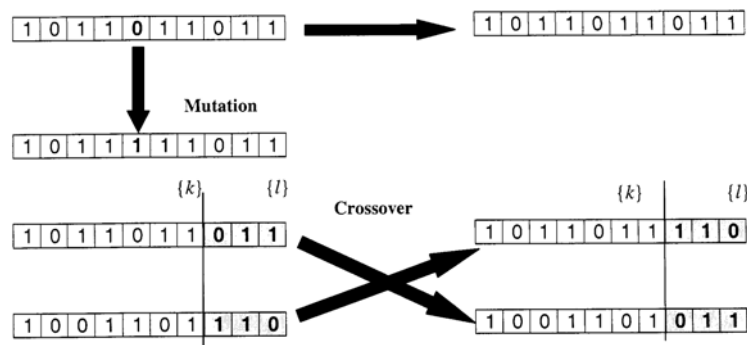


Fig. 1.2 Schematic representation of basic genetic algorithm operations (Manish Kumar, 2004)

1.2 Neural network

A neural network, sometimes called 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 connectionistic 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 Fig.1-3. 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 most common activation functions are linear, binary, sigmoid, hyperbolic tangent, or perceptrons (R.E.King, 1996). The output of the neuron varies between zero and one. The later more complex models are developed based on this simple neuron model.

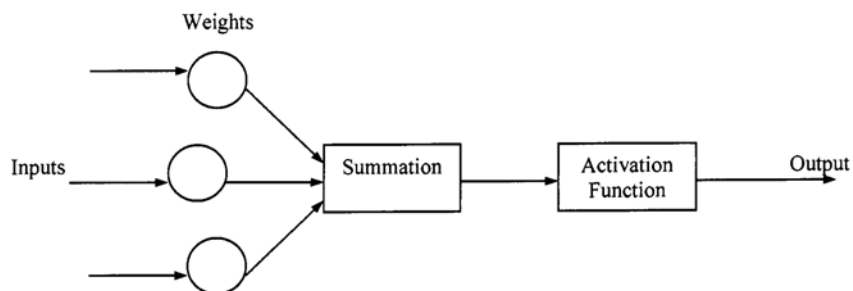


Fig. 1.3 Neuron Model (Frederick G.H., 2005)

The tasks to which artificial neural networks are applied tend to fall within the following broad categories: (1) Function approximation, or regression analysis, including time series prediction and modeling. (2) Classification including pattern and sequence recognition, novelty detection and sequential decision making. (3) Data processing, including filtering, clustering, blind signal separation and compression.

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. Lewis (1998) has 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 H-J-B optimization scheme and 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 behavior 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 plants, are capable of learning from a training set, and parallel processing is inherent. However, a common criticism, particularly in robotics, include 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, 1991). 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. Nonbinary function is suggested to reduce CMAC memory size, however in practical applications with several inputs, memory size dramatic increase. Chien-Kuo Li (2004) propose a neural network composed of single variable CMACs that need much smaller memory space compared to the conventional CMAC. Note that curse of dimensionality does not apply to single input CMAC.

It is a problem of the multidimensional case. Hahn-Ming Lee (2003) introduce to convert a multi-input CMAC to several CAMC with a lower number of inputs reduce the memory space. It can be shown that with this method, the memory size increasing with inputs is linear rather than exponentially. Nevertheless, CMAC in this structure must have nonbinary differentiable input functions, otherwise training weights is not possible except for the out put layer. Aleksander Kolcz (1999) use non-uniform structure for CMAC. CMAC will reflect the distribution of training data more actually, thus avoiding the extra cost of memory size or poor learning performance. Similarly, in Hahn-Ming Lee (2003) method, it adaptively determines quantization of each input dimension based on the various distributions of training data sets. In the approach of Ming-Feng Yeh (2006), self-organizing CMAC has been implemented by incorporating the structure of CMAC into the Kohonen layer of the self-organizing map.

1.3 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. The faster the cumulative reward reaches its maximum, the faster the agent learns the optimal path to the goal (Hani Al-Dayaa, 2006). Figure 1.4 illustrates a reinforcement learning diagram that shows the relationship among the agent, state, action and reward in the environment.

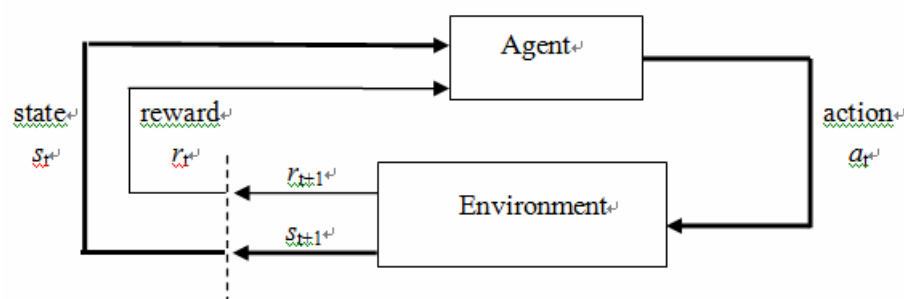


Fig. 1.4 Reinforcement learning diagram

Reinforcement learning mainly consists of three main threads (R.Sutton and A.Barto,1998):

- The first thread involves learning by trial and error and started in the psychology of animal learning, which is called trial-and-error learning.
- The second 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.

- The third thread concerns temporal difference learning, which are distinctive in being driven by the difference between temporally consecutive estimates of the same quantity.

Temporal difference learning plays a significant role in the reinforcement learning. In the reinforcement learning, an agent interacts with an environment for which neither the probability of transitioning to any state nor expected immediate reward are 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 (Sutton, 1988) which synthesize dynamic programming with Monte Carlo 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 used to estimate the role of probability of transitioning and expected immediate reward in the update (Shimon A.W., 2007).

Q-learning (Watkins and Dayan, 1992) 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. Q-value represents the expected return as a result of taking one action in one state, plus the value of the optimal policy thereafter (Krista Falkner, 2006).

A large amount of papers focus the application of Q-learning approach ranging from the control of robot and aircraft, game theory, traffic signal control, and . El-Tantawy S. (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. Valasek, J. et. al (2008) present an improved adaptive-Q learning control methodology for the problem of unmanned air vehicle morphing control. Vlachogiannis, J.G (2004). applied the Q-learning algorithm to the IEEE 14 busbar and to the IEEE 136 busbar system for constrained reactive power control, and the results demonstrate the advantages and flexibility. Many applications are in the field of robots. Ru-Bo Zhang (2004) combined fuzzy logic and reinforcement learning to improve the learning speed of the formation behavior of the robot. Meng Joo Er (2006) proposes a dynamic fuzzy Q-learning (DFQL) controller to define the hip

motion trajectory. A salient feature of the proposed approach is that the DFQL controller can self generate fuzzy rules without a priori knowledge and it is capable of dealing with dynamic systems. Suzuki, H (2008) considers a landing control of an acrobat robot by taking steps to avoid falling-down by applying Q-Learning with function approximation.

However, except 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. Lookup tables are used to store Q-values with one cell for each state and action. Therefore, large state spaces also require impractically large amounts of memory.

1.4 Research summary and dissertation overview

Based on the analysis of previous problems, the research is done around the following two categories:

- The study of CMAC neural network for function approximation and its application on the control complex dynamic systems, such as generic hypersonic vehicle and biped robot.

- The study of Q-learning approach, and its application on footstep planning of biped robot in dynamic environment, around this point, the lauched research include biped robot's mathematic model development, footstep planning approach based on fuzzy Q-learning algorithm and control strategy design for biped robot stepping over dynamic obstacle.

More specific, the main work of this thesis including:

(1) CMAC structure optimization for function approximation

Overview the CMAC neural network's structure of one input and two input case respectively. Compare the required memory size in these two cases and discuss the influence of the CMAC structure parameters to the approximation quality and training time. The examples of CMAC based function approximation are inducted and optimal CMAC structure for a given problem is investigated.

(2) Application of CMAC on generic hypersonic vehicle control system

Due to the strong coupling between the aerodynamics, the airframe, and the propulsion system, the mathematical model of GHV is firstly overviewed. CMAC neural network is applied on the velocity and height controller for nonlinear longitudinal model. Simulation is done on both GHV flight in the nominal case and with pitch moment uncertainty case.

(3) Footstep planning for biped robot based on fuzzy Q-learning algorithm

Q-learning algorithm is introduced and is applied on path planning example. Using fuzzy

logic where both actions and Q function may be represented by Takagi-Sugeno FIS, Fuzzy Q-learning approach is derived. Fuzzy Q-learning approach is developed to learn step length, step duration time and step maximum height in the case of robot and obstacle moving opposite in the sagittal plane. The simulation is done in the case of obstacle moving with average velocity and un-predict velocity.

(4) Found of biped robot mathematical model

The 5-link biped structure is selected for developing the mathematical model of biped robot in the sagittal plane. The bipedal locomotion during the single-support-leg phase is studied as a tree-like topology. The kinematical model of biped robot is derived based on this tree structure. The dynamic model is developed by Lagrangian formulation for its single-support-leg phase.

(5) Control strategy for biped robot stepping over dynamic obstacle

Design the high-level control and low-level control separately. The high-level controller is based on the Fuzzy Q-learning algorithm, which includes the footstep planning, foot trajectory generation and joint angle profile generation. The low-level controller The low-level control allows both to generate joint trajectories and to control the tracking of these desired trajectories, is based on CMAC neural network.

The organization of the dissertation is as follows:

The remaining chapters in this thesis are organized as follows. Chapter 2 study the CMAC structure optimization for function approximation problem. How the structural parameters influence the required memory size and approximation quality is discovered, and the parameters optimization algorithm is developed. Chapter 3 deals with longitudinal controller design for generic hypersonic vehicle. CMAC neural network is used to tracking throttle setting and elevator deflection, therefore effect the change of both flight velocity and height. Chapter 4 focuses on Q-learning approach, and its application. Develop footstep planning for biped robot in dynamic environment, based on Fuzzy Q-learning algorithm. The control strategy for biped robot stepping over dynamic obstacle is investigated in Chapter 5. At the beginning of this chapter, outline 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. The final conclusion and the future work recommendation are presented in Chapter 6.

Chapter 2 CMAC Neural network

2.1 Introduction

The CMAC (Cerebellar Model Articulation Controller) is a neural network based model proposed by Albus inspired from the studies on the human cerebellum (J.S. Albus, 1975(a)(b)). CMAC is a neural network with local generalization abilities. This means that only a small number of weights are necessary to compute the output of this neural network. Consequently, the main interest is the reduction of training and computing times compared with other neural networks (W.T. Miller, 1990). Because of the advantages of simple and effective training properties and fast learning convergence, CMAC neural network has been used in many real-time control systems, pattern recognition and signal processing problems successfully (A.L.Kun, 2000; C.Sabourin, 2005; O.G.Rudenko, 2003; Ming-Feng Yeh., 2007). In addition, a digital hardware implementation is possible for the CMAC neural network (T. Miller, 1990).

During the last three decades, a number of theoretical aspects have been developed overcoming a number of CMAC's primary limitations. For example, in original CMAC, designed by Albus, the responses of excited cells are given by binary basis functions. A number of works have proposed using continuous functions. In paper (Eldracher, M, 1994), authors gave a comparative study between both binary basis function and Gaussians continuous function when CMAC is used for function approximation. Szabó and Horváth (2002) used a modified training rule to increase the CMAC's generalization capability. Chow and Menozzi have studied a self-organizing version of the CMAC (1994). Sabourin et al proposed a new approach making possible to take advantage from both local and global generalization capacities with Fuzzy-CMAC neural networks (2007). The fuzzy-CMAC architecture is based on a merger of outputs of several Single-Input/Single-Output CMAC neural nets. It allows as well decreasing the memory's size as increasing the generalization abilities compared with a multi-input CMAC.

In fact, besides its attractive features, the main drawback of CMAC network in realistic applications is related to the required memory size. The needed memory size depends on the input signal quantification step and the input space size. For real CMAC control applications, on the one hand, in order to increase the accuracy of the control, the chosen quantification step must be as small as possible; on the other hand, in real world applications the input space dimension is greater than two. Therefore, in high-dimensional input cases, its application becomes impractical. In order to overcome the problem relating to the size of the memory, a hashing function is generally used. But in this case, because the number of memory's weights is smaller than the size

of the virtual addressing memory, some collisions can occur. In this chapter, we discuss how both the generalization and step quantization parameters may influence the CMAC approximation quality. Our goal is to find an optimal CMAC structure for a given problem. In this way, it is possible to decrease the memory size according to the desired performance of the CMAC neural network without increase the complexity of CMAC structure.

This chapter is organized as follows. In Section 2.2, the structure of CMAC neural network which includes one dimension and high dimension are introduced respectively. Section 2.3 presents the CMAC structure parameters which determine the memory size and how to calculate the memory size. The examples of CMAC based function approximation are inducted and optimal CMAC structure for a given problem is investigated. In section 2.4, a self-optimizing algorithm for the structure of CMAC has been developed and summary is concluded in Section 2.5.

2.2 CMAC structure

CMAC belongs to the family of feed-forward networks with a single linear trainable layer. 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 network can be considered as an associative memory, which performs two subsequent mappings. The first one, which 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 discrete input data and the association vectors. Every bit in the association vector corresponds 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 :

$$Y = A(X)^T W \quad (2.1)$$

The weights W of CMAC are updated by using equation:

$$W_{t_i} = W_{t_{i-1}} + \frac{\beta e}{N} \quad (2.2)$$

$W_{t_{i-1}}$ and W_{t_i} are respectively the weights before and after 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.

2.2.1 One dimension CMAC structure

Limited to the scalar case, none of the quantization intervals produced by the N scalar quantizers should be the same, that is, at any input point, the quantization intervals to which x belongs in the individual quantization layers overlap, but no two cells belonging to different layers can be exactly the same.

In the following discussion we will consider the case of uniform CMAC quantization, where $\{x_i\}_1^j$ define all uniform partition of input domain into $j-1$ equal length unit intervals. Whereas the individual CMAC quantizers have cells of length $N \times \Delta q$. As the input space is given by an interval of finite length, each quantizer will have a finite range without loss of generality. In the uniform case the staggered arrangement of CMAC quantizers is particularly clear. Given one of the quantizers, the remaining $N-1$ quantizers can be obtained by translating the original quantization-cell by $\Delta q, 2\Delta q, \dots, (N-1) \times \Delta q$. Furthermore, starting with any point of the input $x \in X$, increments or decrements of Δx by Δq at a time will cause exactly one of the quantizers to change its output for $x + \Delta x$ with respect to the values produced for x . Hence, for $\Delta x \geq N \times \Delta q$, the points x and $x + \Delta x$ will share no quantization cells.

2.2.2 High dimension CMAC structure

The quantization described for the one dimensional case is performed in a similar manner for each component of the input vector when input dimension $D \geq 2$. Each of the N vector quantizer of the network now consists of D individual scalar-quantizer components, one per each input dimension. This design guarantees that the D dimensional quantization cells are not the same and, at the same time, that any two cells belonging to different quantization layers will differ additionally in the sense that their D projections on the coordinate axes will be the same (irrespective of whether the cells overlap or not.). Figure 2.1 shows a simplified organization of the receptive fields for two input signals $X = [x_1, x_2]$. Its structure includes a set of $N_c = 41$ binary sensors (receptive fields) regularly distributed on $N_l = 3$ layers. The receptive fields of these detectors cover the totality of the input signals but each field corresponds to a limited range of inputs. On each layer, the receptive fields are shifted to a quantification step $\Delta q = [\Delta q_1, \Delta q_2]$.

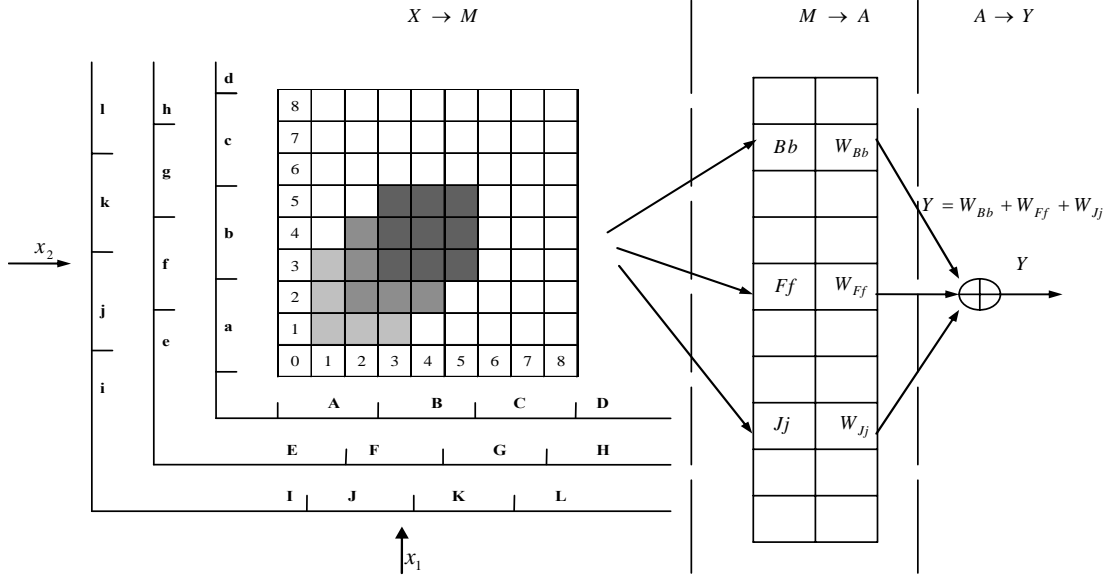


Fig.2.1 Bloc-diagram showing example of 2D CMAC ANN with three layers.

2.3 Parameters influence of CMAC ANN for function approximation

2.3.1 Structure parameters of CMAC ANN

As it has been mentioned previously, to approximate the proposed functions using neural networks, some parameters specific to the selected neural model, must be adjusted in order to define the appropriated structure of the used neural network. Also as it has been emphasized, there are essentially two factors ruling the function approximation quality in a CMAC network. The first one, 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 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 . The used memory size during the ANN training can be calculated by the expressions if the full active sensor vector $[a_0, a_1, \dots, a_m]$ has been calculated.

In the following discussion we will consider the case of uniform CMAC quantization. For a given output component, layer N_j and input x_i we interpret the first sensor as covering the interval $(-\infty, x_{\min} + O_j)$ or $(-\infty, x_{\min} + S_c)$ depending on whether the offset is nonzero. S_c represents the receptive field width (size of the cell). O_j represents the offset of each layer. In the case of uniform CMAC quantization, the offset equals to $O_j = \Delta q N_j$ ($N_j = \{0, 1, \dots, N_l\}$) for the j^{th} layer. Hence, if the i^{th} component of x , x_i satisfies $x_i < x_{\min}$, it is covered by the first sensor field. Similarly, the last sensor is interpreted as covering out to $+\infty$, so if $x_i > x_{\max}$, it is covered by the last sensor. We

then have two cases:

If offset O_j is not zero, the subintervals are then organized as follows:

$$\begin{aligned} & \left[x_{\min}, x_{\min} + O_j \right], \left[x_{\min} + O_j, x_{\min} + O_j + S_c \right], \left[x_{\min} + O_j + S_c, x_{\min} + O_j + 2S_c \right], \dots, \\ & \left[x_{\min} + O_j + (m-1)S_c, x_{\min} + O_j + mS_c \right], \text{ giving } m+1 \text{ sensors.} \end{aligned}$$

If offset O_j is zero, the subintervals are then organized as follows:

$$\begin{aligned} & \left[x_{\min}, x_{\min} + S_c \right], \left[x_{\min} + S_c, x_{\min} + 2S_c \right], \left[x_{\min} + 2S_c, x_{\min} + 3S_c \right], \dots, \left[x_{\min} + (m-1)S_c, x_{\min} + mS_c \right], \\ & \text{giving } m \text{ sensors.} \end{aligned}$$

So for a given component x_i , first we need to find which sensor is active on each level. Fixing the level, the determined active sensor will be stored in address a_i . The active sensor can be determined by using $ceil()$ and $floor()$ calculations according to the algorithm below:

if $O_j > 0$

$$a_i = \begin{cases} 0 & \text{if } x_i < x_{\min} + O_j \\ ceil(x_i - x_{\min} - O_j) / S_c & \end{cases} \quad (2.3)$$

else

$$a_i = floor((x_i - x_{\min}) / S_c)$$

Once the full active sensor vector $[a_0, a_1 \dots a_m]$ has been calculated, we then 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:

$$S_c = \Delta q N_l \quad (2.4)$$

$$N_c^0 = ceil(S_{\min} - S_{\max}) / S_c \quad (2.5)$$

$$N_c = (N_c^0)^2 + (N_c^0 + 1)^2 (N_l - 1) \quad (2.6)$$

2.3.2 Impact of input dimension on required memory size

In this section, we illustrate the impact of input dimension of CMAC network on required memory size. Use sin function and FSIN function which are expressed in equation (2.3) and (2.4) respectively as examples to compare the required memory size of one dimension and two dimension CMAC neural network during its training.

sin function: $[0,1] \longrightarrow [0,1]$

$$y = \sin x \quad (2.7)$$

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

$$FSIN(x_1, x_2) = \sin^2(2\pi x_1) \sin^2(2\pi x_2) \quad (2.8)$$

We choose the number of layers N_l and quantization step Δq randomly for each of the functions. Figure 2.2 and Figure 2.3 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 0.59% (listed in Table 2.1), while two-input FSIN function requires 10 times greater memory size than it for sin function, but E_{square} only approximates 5.81%. If the input dimension is greater than two, the needed memory size is tremendous.

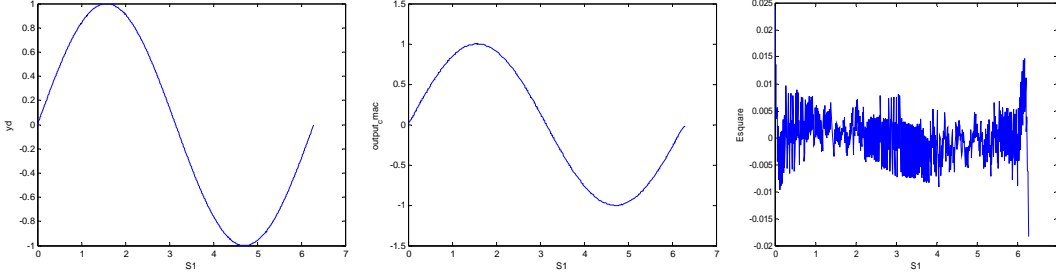


Fig. 2.2 SIN function, CMAC based approximated FSIN function, and mean absolute error

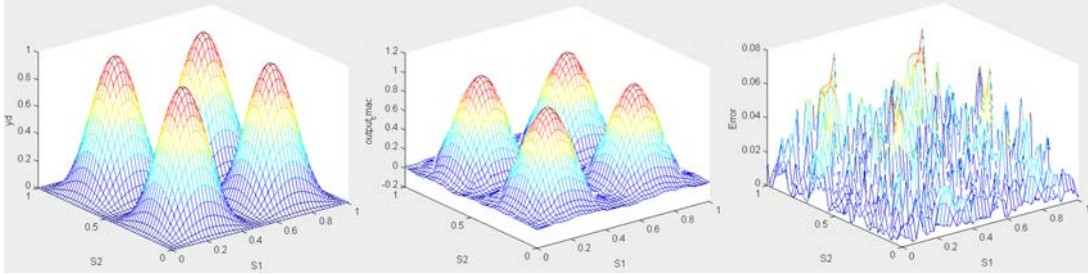


Fig. 2.3 FSIN function, CMAC based approximated FSIN function, and mean absolute error

Table 2.1 Comparing of required memory size for one dimension input and two dimension input CMAC

Function	Δq	N_l	E_{square}	N_C
sin	0.015	15	0.59%	434
FSIN	0.0025	41	5.81%	4940

2.3.3 CMAC training experimental protocol

Three functions are approximated in order to test the CMAC neural network's approximation ability. These 2-D functions are defined by equations (2.8), (2.9) and (2.10) and known as FSIN FCOS and GAUSS respectively. $x_1, x_2 \in [0,1]$ represents the two input signals for FSIN and FCOS functions, while $x_1, x_2 \in [-5,5]$ is the input of the considered 2-D input space for GUASS function.

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

$$FCOS(x_1, x_2) = \frac{1 + \cos(2\pi((2x_1 - 1)^2 + (2x_2 - 1)^2))}{2 \exp(0.25((2x_1 - 1)^2 + (2x_2 - 1)^2))} \quad (2.9)$$

GAUSS function: $[-5,5]^2 \longrightarrow [0,0.2]$

$$\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}((x_1 - \mu_1)^2 + (x_2 - \mu_2)^2)\right) \quad (2.10)$$

For each of the aforementioned functions, a training set including 100×100 random values selected in the corresponding two-dimensional space, has been constructed. Weights of CMAC are updated using equation (2.2). When CMAC is totally trained, three modeling errors are carried out:

- mean absolute error, defined by expression (2.11),
- mean squared error, computed from the equation (2.12) and
- maximum absolute error, obtained from equation (2.13).

$$E_{mean} = \frac{\sum_{i=1}^{100} |y_i^d - y_i|}{\sum_{i=1}^{100} |y_i^d|} \quad (2.11)$$

$$E_{square} = \sqrt{\frac{\sum_{i=1}^{100} (y_i^d - y_i)^2}{\sum_{i=1}^{100} (y_i^d)^2}} \quad (2.12)$$

$$E_{max} = \frac{\max_{i \in \{1, \dots, 100\}} |y_i^d - y_i|}{\max_{i \in \{1, \dots, 100\}} |y_i^d|} \quad (2.13)$$

where y_i^d represents the desired output values and y_i is the actually learned output value.

Figure 2.4 and Figure 2.5 show the shape of the above considered functions, their approximated functions as well as the mean absolute error for each function. To illustrate the CMAC's approximation abilities with the chosen values of the concerned parameters, the output of CMAC is computed using equation (2.1) and the error by equation (2.11) quantifying the dissimilarity between the desired output y_i^d and CMAC issued one y_i .

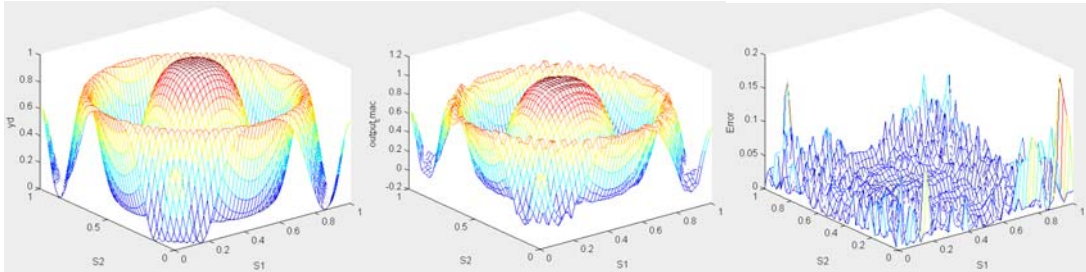


Fig. 2.4 FCOS function, CMAC based approximated FCOS function, and mean absolute error

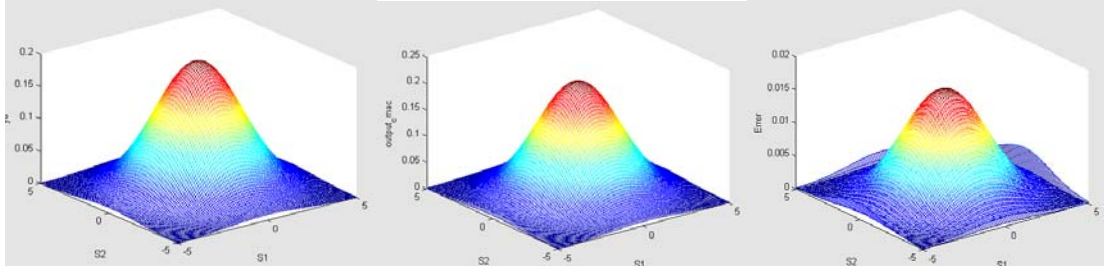


Fig. 2.5 FCOS function, CMAC based approximated FCOS function, and mean absolute error

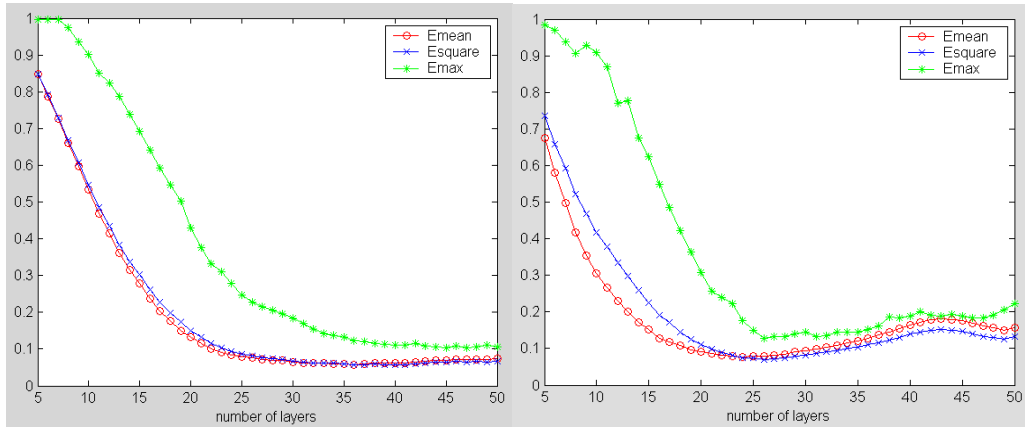
Table 2.2 gives modeling errors examples for the approximations of FSIN, FCOS and GAUSS functions. For the above reported function approximation examples, the quantization step has been set to 0.0075. Concerning the numbers of layer, it has been adjusted to 16 for FSIN, 13 for FCOS and 20 for GAUSS respectively. For these three examples, the chosen structure of CMAC remains arbitrarily meaning that we do not know if this considered structure is or is not optimal. We notice that for all the three functions, the mean absolute error E_{mean} , the mean squared error E_{square} are not very far from each other. However, the maximum absolute error remains quite significant, especially for the approximated FCOS function.

Table 2.2 Approximation performances for FSIN, FCOS and GAUSS function

Function	Δq	N_l	E_{mean}	E_{square}	E_{max}
FSIN	0.0075	16	7.10%	6.48%	11.3%
FCOS	0.0075	13	8.83%	10.0%	26.8%
GAUSS	0.0075	20	7.76%	8.125%	13.1%

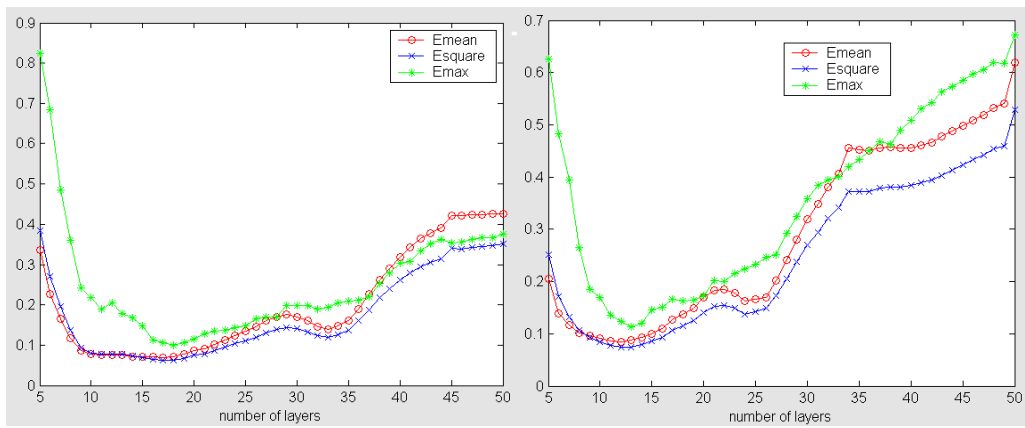
2.3.4 Structural parameters' influence

In this section, by experimental enquiry, we try to show the relation between the structural parameters of CMAC neural network, the quality of the approximation and the required memory size for a given function. For the above described FSIN, FCOS and GAUSS functions, simulations for four several step quantization Δq are carried out, when the number of layers increases from 5 to 50 for both FSIN and FCOS functions, and from 5 to 450 for GAUSS function. In each simulation, the mean absolute error E_{mean} , mean squared error E_{square} and maximum absolute error E_{max} are calculated by using (2.11), (2.12) and (2.13) respectively. The overview of the obtained results is shown in Figure 2.6, 2.7 and 2.8.



(a) $\Delta q = 0.0025$

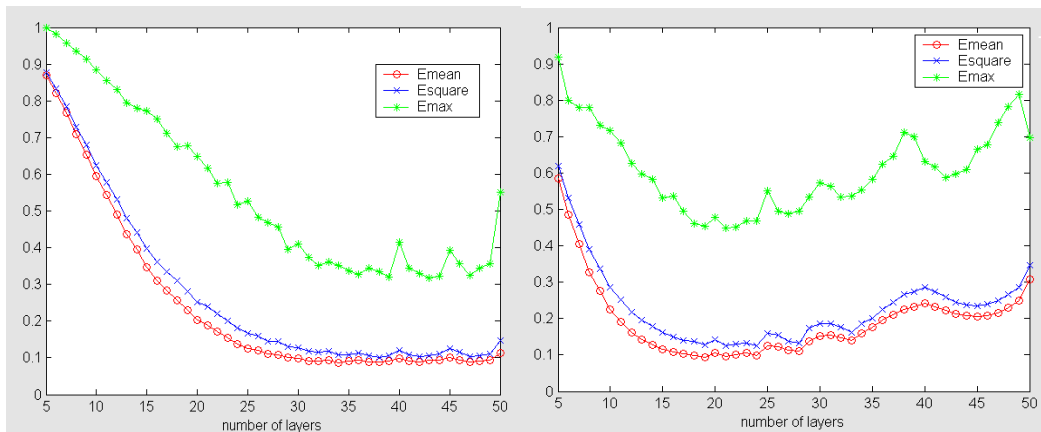
(b) $\Delta q = 0.005$



(c) $\Delta q = 0.0075$

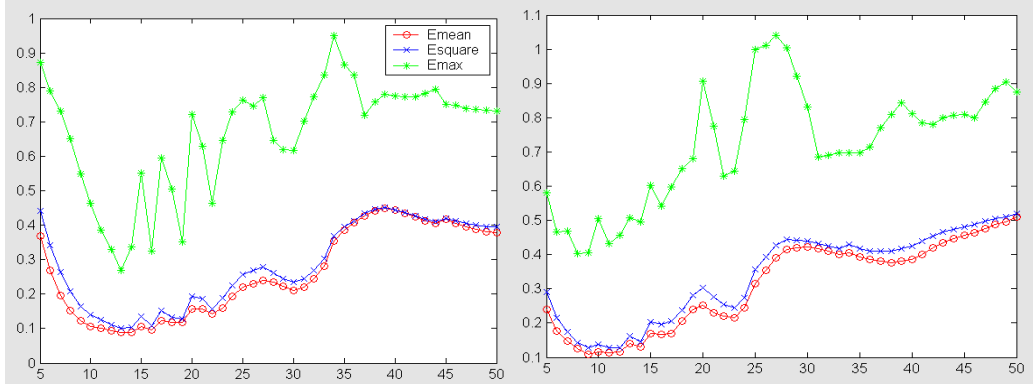
(d) $\Delta q = 0.01$

Fig. 2.6 Mean absolute error, mean squared error and maximum absolute error according to the number of layers for FSIN function when the step quantization are equal to 0.0025, 0.005, 0.0075, 0.01 respectively.



(a) $\Delta q = 0.0025$

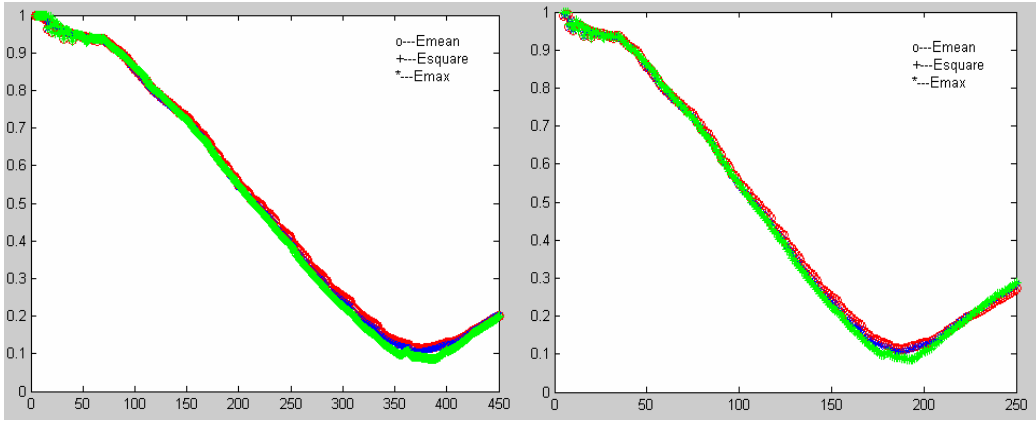
(b) $\Delta q = 0.005$



(c) $\Delta q = 0.0075$

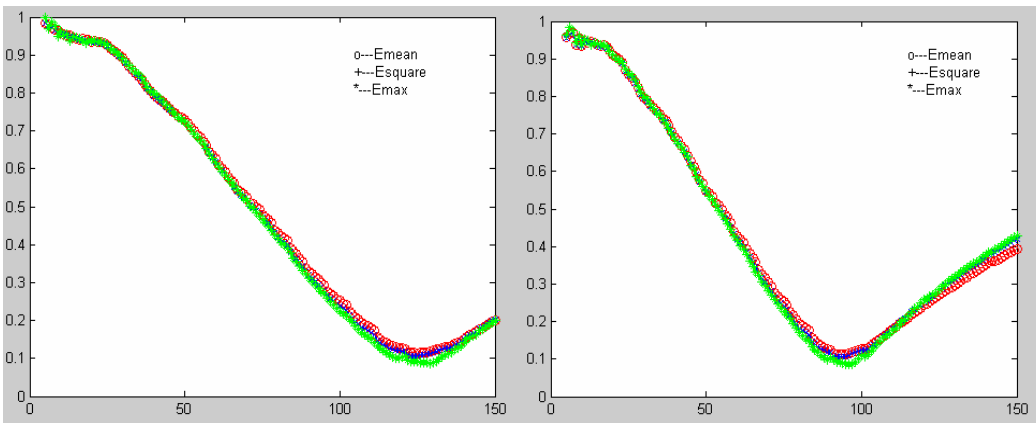
(d) $\Delta q = 0.01$

Fig. 2.7 Mean absolute error, mean squared error and maximum absolute error according to the number of layers for FCOS function when the step quantization are equal to 0.0025, 0.005, 0.0075, 0.01 respectively.



(a) $\Delta q = 0.0025$

(b) $\Delta q = 0.005$



(c) $\Delta q = 0.0075$

(d) $\Delta q = 0.01$

Fig. 2.8 Mean absolute error, mean squared error and maximum absolute error according to the number of layers for GUASS function when the step quantization are equal to 0.0025, 0.005, 0.0075, 0.01 respectively.

It must be noticed that the modeling error depends on the quantization step Δq on the one hand, and the number of layers N_l on the other hand. As the curve trends of the mean absolute error E_{mean} , mean squared error E_{square} and maximum absolute error E_{max} are same, we only take E_{square} as an example. Table 2.3, 2.4 and 2.5 give the optimal structure chosen on the base of

E_{square} for FSIN, FCOS and GAUSS respectively.

Table 2.3 CMAC structure with minimum mean squared error for FSIN function

Δq	N_l	E_{square}	S_C	N_C
0.0025	41	5.81%	0.1225	4940
0.005	26	6.93%	0.13	2089
0.0075	18	6.21%	0.135	1441
0.01	12	6.46%	0.12	1181

Table 2.4 CMAC structure with minimum mean squared error for FCOS function

Δq	N_l	E_{square}	S_C	N_C
0.0025	34	5.81%	0.1225	4940
0.005	20	6.93%	0.13	2089
0.0075	13	6.21%	0.135	1441
0.01	9	6.46%	0.12	1181

Table 2.5 CMAC structure with minimum mean squared error for GAUSS function

Δq	N_l	E_{square}	S_C	N_C
0.0025	115	7.80%	0.2875	7345
0.005	58	8.35%	0.29	3697
0.0075	39	8.84%	0.2925	2481
0.01	29	8.11%	0.29	1841

It can be seen from the above table, the mean squared error E_{square} for the FSIN function is equal to 5.81% and 6.21% in the case where $\Delta q = 0.0025$ and $\Delta q = 0.0075$ respectively. These chosen results show that the approximation abilities of the CMAC are similar in these two cases. However, in the points of view of computing time and memory size, the results are not similar. In fact, the computing time for $\Delta q = 0.0025$ is estimated 2.5 times longer than for $\Delta q = 0.0075$. And the memory size required for $\Delta q = 0.0025$ is 3.5 times greater than when $\Delta q = 0.0075$.

2.4 Structure optimization

Based on the results of previous section, it is clear to point out that depending on the structure of CMAC neural network, the approximately minimum modeling error can be gained with relatively small cost of required memory size and computing time. This remark is very important, because the required memory size and computing time are the key factors that determine if CMAC could be used in the real-time control system. For example, the CPU of robot or aircraft has to do many tasks meanwhile, therefore the useable memory size is quite limited and the control system has to response to the input signals immediately.

Thus, we developed the self-optimizing algorithm to adjust the number of layers and quantization step, in order to obtain the optimized CMAC's structure. Inspired from the reinforcement learning algorithm, a three-dimension matrix \mathbf{Q} is constructed for installing the

“reinforcement signal”, in which each dimension of the matrix \mathbf{Q} represents number of layers, quantization step and reinforcement respectively. The expression of reinforcement signal r refers to (2.14), which stands for evaluating the modeling error and required memory size of CMAC.

$$r = \frac{\gamma \cdot N_c}{100} + 1000 \cdot (1 - \gamma) \cdot E_{mean} \quad (2.14)$$

Where γ stands for the discount factor of the approximation quality and memory size. To balance the order of magnitude of these two factors, the coefficient 100 and 1000 are selected. In this manner, the modeling error and memory size are taken into consideration at the same time.

The logic of the self-optimizing algorithm is given below:

Choosing N_l from minimum to maximum number
 Choosing Δq from minimum to maximum value
 Based on CMAC training, calculating N_c and E_{square}
 For every combination of N_l and Δq , calculating the reinforcement signal r according to (3.14)
 Install r corresponding with N_l and Δq in a three-dimension matrix
 Finding the minimum value of r , lookup the matrix \mathbf{Q} for the matching value of N_l and Δq

We also take the FSIN and FCOS functional approximation and examples to test the developed algorithm. The discount factor γ is chosen to be 0.5 in this example, which means that the approximation quality and memory size are considered to be the same importance. It is deserved to mention that the traversal of the above two factors can be done off-line, and the on-line calculation only includes the lookup table process. Figure 2.9 and 2.10 show the obtained value of three-dimension matrix \mathbf{Q} for FSIN and FCOS case respectively. The optimized structural parameters of CMAC neural network with minimum modeling error and memory size for both functions are listed in Table 2.6.

Table 2.6 Optimized structural parameters of CMAC with minimal mean squared error for approximating FSIN and FCOS functions

Function	N_l	Δq	N_c	E_{square}
FSIN	15	0.0069	1794	5.53%
FCOS	20	0.0053	2399	7.65%

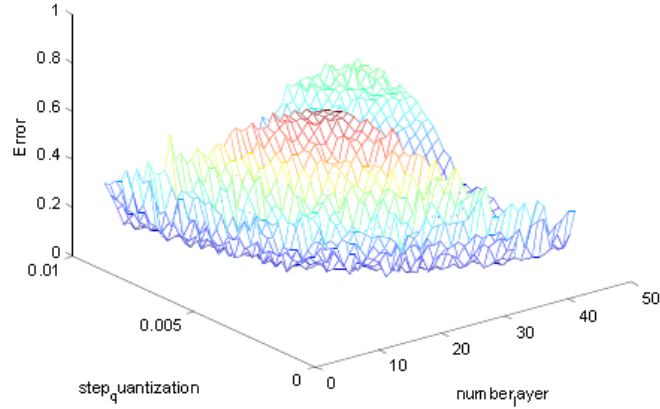


Fig. 2.9 The value of three-dimension matrix Q for approximating FSIN function

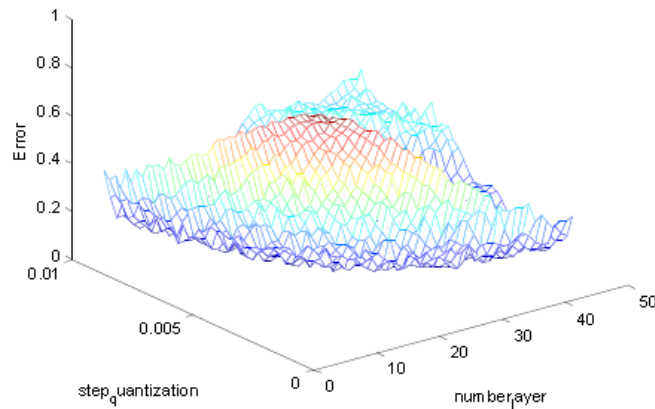


Fig. 2.10 The value of three-dimension matrix Q for approximating FCOS function

2.5 Summary

Because of the advantages of simple and effective training properties and fast learning convergence, CMAC neural network has been used in many real-time control systems pattern recognition and signal processing problems successfully. Besides its attractive features, the main drawback of the CMAC network in realistic on-line applications is related to the required memory size. For example, for the use of robot control system or aircraft control system, because the CPU has to handle an awful lot of information on-line at the same time, the left memory size is quite limited. In this paper, we have shown how both generalization and step quantization parameters influence the approximation qualities of CMAC neural networks. The presented simulation results show that an optimal structure carrying out a minimal modeling 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. Inspired from the reinforcement learning, we have developed the self-optimizing algorithm for the structure of CMAC, in which the modeling error and memory size of the network are considered at the same time. The results

indicate the effectiveness of this approach, and the approximately minimum modeling error can be achieved with relatively small number of required memory size without increase the structural complexity.

In the future, we will focus on the application of self-optimizing structural CMAC in the real-time control system.

Chapter 3 CMAC based controller design for generic hypersonic vehicle

3.1 Introduction

Hypersonic flight is flight through the atmosphere at speeds above about Mach 5.5, a speed where disassociation of air begins to become significant and high heat loads exist.

Feasible atmospheric hypersonic flight using air-breathing propulsion will provide routine and affordable space access and high speed civilian transportation. Technologies that enable cost effective vehicle systems for use in space launching, orbiting and maneuvering will also translate easily to improvements in military interceptors, tactical and strategic reconnaissance, as well as in high-speed and orbital transport activities (Andrew Clark et al. 2006).

Since the first Hypersonic Research Engine (HRE) program proposed by NASA began in 1964. The technologies around the generic hypersonic vehicle have been studied for nearly half a century. However, none of the flight tests was conducted successfully before 2004, in spite of the tremendous amount of effort and cost. Among them, the most famous and still expanding one is the Hyper-X program, which is started by NASA in 1996. This program is focused on the development and flight testing of one full-scale demonstrator vehicle and small-scale, whose serial number is X-43A, X-43B, X-43C, X-43D et al. X-43A is a 12 foot long hydrogen powered experimental vehicle with a five-foot wingspan. It has been used in three scramjet powered and un-powered flight tests at Mach 7 and Mach 10. Its flights were first successfully conducted in 2004. X-51 is another hypersonic flight which was tested successfully in recent years. The X-51 WaveRider program is run as a cooperative effort of the United States Air Force, DARPA, NASA, Boeing and Pratt & Whitney Rocketdyne. It is an unmanned scramjet demonstration aircraft for hypersonic (Mach 6, approximately 4,000 miles per hour (6,400 km/h) at altitude) flight testing. It successfully completed its first free-flight on 26 May 2010 and also achieved the longest duration flight at speeds over Mach 5.

In hypersonic air-breathing flight vehicles with airframe-integrated scramjet engines, the airframe, propulsion system and structural dynamics are highly interactive. The primary lift is generated by the body itself (Chavez, F.R., 1999; Curran, E.T., 2001). The engine airframe integration causes significant coupling between the propulsion system and vehicle aerodynamics (Bertin, J.J., 1992; Walton, J.T., 1989). Considering the longitudinal dynamics, the flow pressure acting on the forebody generates a nose-up pitching moment while the external nozzle flow

produces additional lift and nose down pitching moment. Similarly, the aerodynamics affects the propulsion system in several ways. The capture and compression of the flow through the inlet is determined by the properties of the bow shock wave under the vehicle forebody, which are determined by the angle of attack, the dynamic pressure, and the free stream characteristics. The thrust and the pressure at the engine inlet are affected by pitching control surface deflections. At the same time, the fuel flow rate as well as the diffuser area ratio changes has an effect on the vehicle pitch rate (Andrew Clark et al. 2006).

Because these coupling are significant, explicit characterization of flight dynamics and designing effective controller are highly challenging for this class of vehicles. Most of the previous controller design is based on the LTI model. Bolender and Doman (2005) have developed a control oriented nonlinear physics based model of the longitudinal dynamics of generic hypersonic vehicle. However, the approximated LTI model structure will lose some useful information of the system. There are few papers focus on the controller of nonlinear elastic model of GHV.

This chapter is organized as follows: In section 3.2, we will develop the elastic longitudinal dynamic model of generic hypersonic vehicle, which includes overview the linear mathematic model, nonlinear longitudinal model and develop the elastic longitudinal dynamics considering the elastic characteristics of the body. In section 3.3, we design the longitudinal controller based on CMAC neural network. One experiment is done to evaluate the performance of proposed controller to step response, and another experiment is done in the case of parameter uncertainty in section 3.4.

3.2 Mathematic model of GHV

3.2.1 Overview the mathematic models of GHV

Due to the strong coupling between the aerodynamics, the airframe, and the propulsion system, the dynamic characteristic of generic hypersonic flight makes the modeling of GHV very challenging. A mounts of literature discuss the rigid-body model of GHV for the winged-cone accelerator configuration. Most of them only focus on the longitudinal motion of the GHV dynamic models. However, the main differences between several different formulations of the dynamic model are weather linearization or not, and the method used for the assumption of the aerodynamics, mass, environment and propulsion properties (Bars Fidan, 2003).

Gregory et al. (1994) first study the hypersonic vehicle as the linear model, which can be expressed as:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \quad (3.1)$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}) \quad (3.2)$$

where \mathbf{x} , \mathbf{u} and \mathbf{y} denote the state, the input and the output of GHV system, respectively. Typically, \mathbf{x} is a five dimension vector, which contains information about velocity V , angle of attack α , pitch rate q , pitch attitude θ and altitude h .

$$\mathbf{x} = [V, \alpha, q, \theta, h]^T \quad (3.3)$$

The input \mathbf{u} is a two dimensional vector whose entries are adjustable control coefficients, such as elevon angle and fuel equivalence ratio, that determines the aerodynamic and propulsive power supplied to the system or fuel and air flow rates in the integrated engine.

$$\mathbf{u} = [\delta_e, \eta_f]^T \quad (3.4)$$

The output \mathbf{y} can be as same as the state \mathbf{x} , measurable entries of \mathbf{x} , or some measurements giving information about \mathbf{x} . It carries information about the altitude and the velocity of the GHV. \mathbf{A} and \mathbf{B} are system matrices which depend on the flight. Numerical values of \mathbf{A} and \mathbf{B} for certain flight conditions are derived from a six degree of freedom nonlinear rigid-body simulation of the conical accelerator vehicle.

Later, Gregory et al. (1994) included the turbulence as atmospheric affects in the linear model, and the motion equation is generalized as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{E}_x \mathbf{d} \quad (3.5)$$

where \mathbf{d} is a two dimension vector which stands for the atmospheric disturbances. And \mathbf{E}_x is a 5×2 matrix depending on the flight conditions.

In the above model, the forebody is axisymmetric and conical, the nozzle section is a cone frustum, and the engine modules are placed cylindrically all around the body. Furthermore, the model is developed on the assumption that the body is rigid the coupling effects among aerodynamics, propulsion, and structural dynamics of the vehicle are negligible. Later, Bushcek and Calise (1997) take the coupling effects into consideration. They are formulated as uncertainties and incorporated into the model as:

$$\dot{\mathbf{x}} = (\mathbf{A} + \Delta_A)\mathbf{x} + (\mathbf{B} + \Delta_B)\mathbf{u} \quad (3.6)$$

in which, Δ_A and Δ_B are additive uncertainties respectively. The impact of the propulsive perturbations on the pitching moment is focused among the coupling effects and this effect is addressed as parametric uncertainty in C_{m_α} , which can be calculated by $C_{m_\alpha} = \frac{\partial C_m}{\partial \alpha}$, where C_m is pitch moment coefficient and α is the angle of attack.

A drawback of the linear approaches above is that the capability of the model to represent the dynamics and the coupling effects realistically is limited. One way to eliminate this drawback is

introducing nonlinear models that carry more information instead of linear ones. C.Marrison (1998) and Q.Wang (2000) utilized the nonlinear longitudinal model for a specific flight condition in their design of GHV robust control system. In this model the coupling effects and other possible variations from the nominal system are also taken into consideration. The most widely used approach is taking a set of random variables to represent these effects and variations in order to design the robustness controller (Stengel, R.F., 1994; Zhou, K., 1998). The general nonlinear dynamic model of GHV can be described by:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}) \quad (3.7)$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{v}) \quad (3.8)$$

where \mathbf{f} and \mathbf{h} are smooth nonlinear functions, and \mathbf{v} is assumed to be constant throughout and individual process. The state \mathbf{x} and the input \mathbf{u} is defined by:

$$\mathbf{x} = [V, \gamma, h, \alpha, q]^T \quad (3.9)$$

$$\mathbf{u} = [\delta_t, \delta_e]^T \quad (3.10)$$

where \mathbf{x} includes the information of velocity V , flight-path angle γ , altitude h , angle of attack α , and pitch rate q . δ_t is the throttle setting, and δ_e represents the pitch control surface deflection. In the following section, we will given the expressions of nonlinear longitudinal equations of GHV.

3.2.2 Nonlinear longitudinal equations of GHV

3.2.2.1 Nonlinear longitudinal model of GHV

In order to be utilized in the simulation, the motion equations of generic hypersonic vehicle accounts for the time varying center of mass, the center of gravity and the moments of inertia. However, the total mass of the vehicle, its c.g. location and the products of inertia vary as fuel is consumed. Therefore, in the following equations of motion, it is assumed that:

- the GHV mass model is a rigid vehicle structure
- c.g. only moves along the body x-axis as the fuel is consumed.
- fuel slosh is negligible
- and the products of inertia are negligible.

The GHV nonlinear longitudinal equations of motion flying at orbital altitudes must include both an inverse-square-law gravitational model and the centripetal acceleration for the non-rotating earth. GHV in the inertial space is under the join forces of gravity, aerodynamic force and thrust. Its motion corresponds to the Newton's second law and meets the following vector

equations:

$$m\mathbf{a}_i = \mathbf{F} \quad (3.11)$$

In which, m is the mass of GHV, \mathbf{a}_i is the absolute acceleration, \mathbf{F} is the join forces action on the vehicle.

The research object in this paper is establish the mathematic model of generic hypersonic vehicle relative to the earth, which means considering the motion in the geocentric coordinate system S_e . Because of the angular velocity of earth $\boldsymbol{\omega}_e$, the geocentric coordinate system is not an inertial frame. Therefore, the absolute acceleration of GHV \mathbf{a}_i equals to the sum of relative acceleration \mathbf{a}_k , transport acceleration \mathbf{a}_e and Coriolis acceleration \mathbf{a}_c :

$$\mathbf{a}_i = \mathbf{a}_k + \mathbf{a}_e + \mathbf{a}_c \quad (3.12)$$

in which

$$\mathbf{a}_k = \frac{d\mathbf{V}}{dt} \quad (3.13)$$

$$\mathbf{a}_e = \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \mathbf{r}) \quad (3.14)$$

$$\mathbf{a}_c = 2\boldsymbol{\omega}_e \times \mathbf{V} \quad (3.15)$$

In the above equation, \mathbf{V} is the velocity of mass center of GHV with respect to the earth, known as the relative velocity.

Join forces \mathbf{F} include gravitational attraction $m\mathbf{g}$, aerodynamic force \mathbf{R} and thrust \mathbf{T} :

$$\mathbf{F} = m\mathbf{g} + \mathbf{R} + \mathbf{T} \quad (3.16)$$

where, the acceleration of earth gravity \mathbf{g} is

$$\mathbf{g} = -\left(\frac{\mu}{r^3}\right)\mathbf{r} \quad (3.17)$$

and $\mu = 3.986005 \times 10^{14} \text{ m}^3/\text{s}^2$ is the earth gravitational constant.

Substituting equation (3.12)~(3.17) to equation (3.11), one can obtain the kinematical equation of GHV relative to the earth.

$$\frac{d\mathbf{V}}{dt} = -\frac{\mu}{r^3}\mathbf{r} + \frac{\mathbf{T} + \mathbf{A}}{m} - 2\boldsymbol{\omega}_e \times \mathbf{V} - \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \mathbf{r}) \quad (3.18)$$

Equation (3.18) is established on the basis of geocentric coordinate system S_e , considering the rotation of S_e in the inertial frame S_i . If we choose another movable coordinate system S_m as the reference, which has the angular velocity $\boldsymbol{\omega}_{me}$ relative to S_e , we can deduce the component of equation (3.18) in this movable frame S_m ,

$$\left(\frac{d\mathbf{V}}{dt}\right)_m = \frac{d(\mathbf{V})_m}{dt} + (\boldsymbol{\omega}_{me})_m^\times (\mathbf{V})_m \quad (3.19)$$

The motion equation of GHV in the matrix form can be obtained based on equation (3.18) and (3.19).

$$\begin{aligned} \frac{d(\mathbf{V})_m}{dt} = & -(\boldsymbol{\omega}_{me})_m^\times (\mathbf{V})_m - \frac{\mu}{r^3} (\mathbf{r})_m + \frac{(\mathbf{T})_m + (\mathbf{A})_m}{m} \\ & - 2(\boldsymbol{\omega}_e)_m^\times (\mathbf{V})_m - (\boldsymbol{\omega}_e)_m^\times (\boldsymbol{\omega}_e)_m^\times (\mathbf{r})_m \end{aligned} \quad (3.20)$$

According to the engineering experience, the flight-path axis system S_k is chosen to be the frame of reference S_m , the prime reference of thrust \mathbf{T} is selected to be the aircraft-carried axis system S_b co-currently, and the prime reference of aerodynamic force \mathbf{A} is the airflow coordinate system S_a .

In the following part, we will give the transformation expression of different coordinate mentioned above but without reduction.

Suppose the position of vehicle is (r, λ, ϕ_c) , which is global coordinate. r is the distance between earth center and vehicle, while λ is longitude and ϕ_c is latitude. The transformation matrix from earth frame to the plumb axis system is:

$$\mathbf{L}_{ve} = \mathbf{L}_y\left(-\phi_c - \frac{\pi}{2}\right) \mathbf{L}_z(\lambda) = \begin{bmatrix} -\sin \phi_c \cos \lambda & -\sin \phi_c \sin \lambda & \cos \phi_c \\ -\sin \lambda & \cos \lambda & 0 \\ -\cos \phi_c \cos \lambda & -\cos \phi_c \sin \lambda & -\sin \phi_c \end{bmatrix} \quad (3.21)$$

The aircraft-carried axis system S_b and the plumb axis system S_v can be adopted to determine the attitude of the vehicle relative to the ground. Therefore, the transformation matrix from S_v to S_b can be expressed by three Eulerian angles: yaw angle ψ , pitch angle ϑ and roll angle ϕ .

$$\begin{aligned} \mathbf{L}_{bv} = & \mathbf{L}_x(\phi) \mathbf{L}_y(\vartheta) \mathbf{L}_z(\psi) \\ = & \begin{bmatrix} \cos \vartheta \cos \psi & \cos \vartheta \sin \psi & -\sin \vartheta \\ \cos \psi \sin \phi \sin \vartheta - \sin \psi \cos \phi & \sin \phi \sin \vartheta \sin \psi + \cos \phi \cos \psi & \cos \vartheta \sin \phi \\ \cos \phi \sin \vartheta \cos \psi + \sin \phi \sin \psi & \sin \vartheta \sin \psi \cos \phi - \cos \psi \sin \phi & \cos \phi \cos \vartheta \end{bmatrix} \end{aligned} \quad (3.22)$$

Airflow coordinate system S_a is connected with aircraft-carried axis system S_b by angle of attack α , and sideslip angle β , the transformation matrix from S_b to S_a is:

$$\mathbf{L}_{vb} = \mathbf{L}_z(\alpha) \mathbf{L}_y(\beta) = \begin{bmatrix} \cos \beta \cos \alpha & \sin \alpha & -\sin \beta \cos \alpha \\ -\cos \beta \sin \alpha & \cos \alpha & \sin \beta \sin \alpha \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \quad (3.23)$$

Through flight-path angle γ and flight path angle χ , affiliate the flight-path axis system to the plumb axis system. The transformation matrix can be written as:

$$\mathbf{L}_{kv} = \mathbf{L}_y(\gamma)\mathbf{L}_\chi(\chi) = \begin{bmatrix} \cos\gamma\cos\chi & \cos\gamma\sin\chi & -\sin\gamma \\ -\sin\chi & \cos\chi & 0 \\ \sin\gamma\cos\chi & \sin\gamma\sin\chi & \cos\gamma \end{bmatrix} \quad (3.24)$$

Using the transformation matrix of different coordinate, one can deduce the deformed formation of equation (3.20)

$$\begin{aligned} \frac{d(\mathbf{V})_k}{dt} = & -(\boldsymbol{\omega}_{ke})_k^\times (\mathbf{V})_k - \frac{\mu}{r^3}(\mathbf{r})_k + \frac{\mathbf{L}_{kv}\mathbf{L}_{vb}[(\mathbf{T})_b + \mathbf{L}_{ba}(\mathbf{A})_a]}{m} \\ & -2(\boldsymbol{\omega}_e)_k^\times (\mathbf{V})_k - (\boldsymbol{\omega}_e)_k^\times (\boldsymbol{\omega}_e)_k^\times (\mathbf{r})_k \end{aligned} \quad (3.25)$$

With respect to the earth, the angular velocity $\boldsymbol{\omega}_{ke}$ of track coordinate S_k is

$$\boldsymbol{\omega}_{ke} = \boldsymbol{\omega}_{ve} + \boldsymbol{\omega}_{kv} \quad (3.26)$$

in which, $\boldsymbol{\omega}_{ve} = \dot{\lambda}\mathbf{k}_e - \dot{\phi}_c\mathbf{j}_v$ is the angular velocity of plumb axis system S_v relative to the earth, while $\boldsymbol{\omega}_{kv} = \dot{\chi}\mathbf{k}_v + \dot{\gamma}\mathbf{j}_k$ is the angular velocity of flight-path axis system with respect to S_v .

Again, using the transformation relation (3.21) and (3.24), one can get the vector expression of $\boldsymbol{\omega}_{ke}$ in flight-path axis system:

$$\begin{aligned} (\boldsymbol{\omega}_{ke})_k &= \mathbf{L}_{kv}\mathbf{L}_{ve} \begin{bmatrix} 0 \\ 0 \\ \dot{\lambda} \end{bmatrix} + \mathbf{L}_{kv} \begin{bmatrix} 0 \\ -\dot{\phi}_c \\ \dot{\chi} \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\gamma} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \dot{\lambda}\cos\phi_c\cos\gamma\cos\chi - \dot{\phi}_c\cos\gamma\sin\chi + \dot{\lambda}\sin\phi_c\sin\gamma - \dot{\chi}\sin\gamma \\ -\dot{\lambda}\cos\phi_c\sin\chi - \dot{\phi}_c\cos\chi + \dot{\gamma} \\ \dot{\lambda}\cos\phi_c\sin\gamma\cos\chi - \dot{\phi}_c\sin\gamma\sin\chi - \dot{\lambda}\sin\phi_c\cos\gamma + \dot{\chi}\cos\gamma \end{bmatrix} \end{aligned} \quad (3.27)$$

The relation of longitude and latitude is :

$$\begin{cases} \dot{\phi}_c = \frac{V\cos\gamma\cos\chi}{r} \\ \dot{\lambda} = \frac{V\cos\gamma\sin\chi}{r\cos\phi_c} \end{cases} \quad (3.28)$$

$\dot{\lambda}$ and $\dot{\phi}_c$ in expression (3.27) can be eliminated by equation (3.28).

The component of velocity vector \mathbf{V} of generic hypersonic vehicle in the flight-path axis system $[0 \ 0 \ \boldsymbol{\omega}_e]^T$ is:

$$(\mathbf{V})_k = [V \ 0 \ 0]^T \quad (3.29)$$

And, the component of radius vector r of GHV in S_k is:

$$(\mathbf{r})_k = \mathbf{L}_{kv}(\mathbf{r})_v = [r\sin\gamma \ 0 \ -r\cos\gamma]^T \quad (3.30)$$

The component of earth rotational velocity $\boldsymbol{\omega}_e$ in geocentric coordinate system S_e is

$[0 \ 0 \ \omega_e]^T$, therefore, it is not difficult to obtain its component in S_k using the transformation matrix:

$$(\omega_e)_k = \mathbf{L}_{kv} \mathbf{L}_{ve} (\omega_e)_e = \omega_e \begin{bmatrix} \cos \phi_c \cos \gamma \cos \chi + \sin \phi_c \sin \gamma \\ -\cos \phi_c \sin \gamma \\ \cos \phi_c \sin \gamma \cos \chi - \sin \phi_c \cos \gamma \end{bmatrix} \quad (3.31)$$

By the same way, the thrust T has to be deformed in the same coordinates system S_k :

$$\begin{bmatrix} F_{xk} \\ F_{yk} \\ F_{zk} \end{bmatrix} = \mathbf{L}_{kv} \mathbf{L}_{vb} [(T)_b + L_{ba} (A)_a] \quad (3.32)$$

$$= \begin{bmatrix} -D + T \cos \alpha \cos \beta \\ (C + T \cos \alpha \sin \beta) \cos \gamma_v + (L + T \sin \alpha) \sin \gamma_v \\ (C + T \cos \alpha \sin \beta) \sin \gamma_v - (L + T \sin \alpha) \cos \gamma_v \end{bmatrix}$$

in which, L is the lift, D is the drag, C is the lateral force and γ_v is the velocity roll angle.

Substitute equation (3.26)~(3.32) into expression (3.25), the kinetic equation of GHV in flight-path axis system S_k is available.

$$\begin{bmatrix} dV/dt \\ V \cos \gamma (d\chi/dt) \\ -V (d\gamma/dt) \end{bmatrix} = \frac{V^2}{r} \begin{bmatrix} 0 \\ \tan \phi_c \cos^2 \gamma \sin \chi \\ -\cos \gamma \end{bmatrix} + \frac{\mu}{r^2} \begin{bmatrix} -\sin \gamma \\ 0 \\ \cos \gamma \end{bmatrix} + \frac{1}{m} \begin{bmatrix} F_{xk} \\ F_{yk} \\ F_{zk} \end{bmatrix} \quad (3.33)$$

$$+ 2\omega_e V \begin{bmatrix} 0 \\ -\cos \phi_c \sin \gamma \cos \chi + \sin \phi_c \cos \gamma \\ -\cos \phi_c \sin \chi \end{bmatrix}$$

$$+ \omega_e^2 r \begin{bmatrix} \cos \phi_c (-\sin \phi_c \cos \gamma \cos \chi + \cos \phi_c \sin \gamma) \\ \sin \phi_c \cos \phi_c \sin \chi \\ \cos \phi_c (-\sin \phi_c \sin \gamma \cos \chi - \cos \phi_c \cos \gamma) \end{bmatrix}$$

Combined with the relation expression between distance to the geocentre and relative velocity:

$$\dot{r} = V \sin \gamma \quad (3.34)$$

Equation (3.28) (3.33) and (3.34) make up the kinematic equations of generic hypersonic vehicle in the flight-path axis system. In the process of mathematic modeling for GHV, the influence of earth rotation and its radius of curvature to the motion of GHV is taken into consideration.

Aim at the considered objective generic hypersonic model, we care about the control system design for its horizon flight, hence only the motion of GHV in the longitudinal plane is taken into account. If the lateral aerodynamic force and moment is not considered, the thruster force is assumed to along the aircraft-carried axis, the influence of longitude and latitude can be negligible,

and the flight altitude h , the geocentric distance r have the same rate of change, we are interested in the flight altitude h during the flight control, combined with the condition $\chi = \phi_c = \beta = \gamma_v = 0$, the kinematic equations of generic hypersonic vehicle, equations (3.28) and (3.33) degenerated as:

$$\dot{\phi} = \frac{V \cos \gamma}{r} \quad (3.35)$$

$$\dot{h} = V \sin \gamma \quad (3.36)$$

Supplement with the kinematic equations and dynamic equations of GHV rotating around its center of mass, the equation set of generic hypersonic vehicle flying in the longitudinal plane can be obtained (Matthew Kuipers, 2007). The state space equation for the longitudinal dynamics are governed by the set of differential equations for velocity V , flight-path angle γ , altitude h , angle of attack α and pitch rate q , and can be expressed as:

$$\dot{V} = \frac{T \cos \alpha_r - D}{m} - \frac{\mu \sin \gamma}{r^2} \quad (3.37)$$

$$\dot{\gamma} = \frac{L + T \sin \alpha_r}{mV} - \frac{(\mu - V^2 r) \cos \gamma}{Vr^2} \quad (3.38)$$

$$\dot{h} = V \sin \gamma \quad (3.39)$$

$$\dot{\alpha} = q - \dot{\gamma} \quad (3.40)$$

$$\dot{q} = \frac{M_{yy}}{I_{yy}} \quad (3.41)$$

Equation (3.37) is simply Newton's law treating the vehicle as a point mass and resolving the components of force onto the velocity vector. The second term in the expression includes a varying gravitational force because of the inverse square law. Equation (3.38), which is the rotational kinematics equation, is calculated by the resolving the force components onto the upwardly pointing normal to the velocity vector. The second term of this equation comprises the centripetal force along with the varying gravitational field. Equation (3.40), which also expresses the rotational kinematics, is obtained from the geometric interpretation of α along with the elastic deflections and taking the time derivative. Equation (3.41) represents the translational kinematics equation for altitude. Euler's law governs the rotational dynamics, and hence the pitch dynamics (D.Clark, 2006)

In the following part, we will introduce how to model the lift L , the drag D , the thrust T , the pitching moment M , and the radius r_e from the center of earth in detail.

The total drag coefficient is obtained as

$$C_D = C_{Da} + C_{D,de} + C_{D,da} + C_{D,dr} \quad (3.42)$$

in which, $C_D = f(M, \alpha, \delta_e, \delta_r, \delta_c)$, C_{D_e} is the drag increment coefficient for basic vehicle, $C_{D,da}$ and $C_{D,de}$ represent for drag increment coefficient of the right elevon and left elevon respectively, $C_{D,dr}$ is the drag increment coefficient for rudder.

Thus, the drag force can be calculated as:

$$D = \bar{q} S_{ref} C_D \quad (3.43)$$

where, S_{ref} is the reference area, in our case it is theoretical wing area. \bar{q} is the dynamic pressure,

which can be expressed as $\bar{q} = \frac{1}{2} \rho V^2$, ρ is the air density.

The total lift coefficient can be obtained as:

$$C_L = C_{L_a} + C_{L,de} + C_{L,da} + C_{L,dc} \quad (3.44)$$

In the above equation, $C_L = f(M, \alpha, \delta_a, \delta_e, \delta_c)$, similarly with drag coefficient, C_{L_a} is the total lift coefficient for basic vehicle, $C_{L,da}$ and $C_{L,de}$ stands for the lift increment coefficient for right and left elevon respectively, $C_{L,dr}$ is the lift increment coefficient for rudder.

Therefore, the lift force is given by:

$$L = \bar{q} S_{ref} C_L \quad (3.45)$$

The total pitching moment coefficient is obtained by:

$$C_m = C_{m_a} + C_{m,de} + C_{m,da} + C_{m,dr} + C_{m,dc} + C_{mq} \left(\frac{qc}{2V} \right) \quad (3.46)$$

in which, $\left(\frac{qc}{2V} \right)$ is used to compute the non-dimensional pitch rate.

If the pitching moment about c.g. is required, then we have:

$$M_{yy} = M_{arc} - x_{cg} Z \quad (3.47)$$

Where x_{cg} is the longitudinal distance from momentum reference to vehicle c.g. Z-axis force can be computed as:

$$Z = -D \sin \alpha - L \cos \alpha \quad (3.48)$$

The pitching moment relative to the moment reference center is given by:

$$M_{yy} = \bar{q} c S_{ref} C_m \quad (3.49)$$

Note that each non-dimensional coefficient is a function of vehicle flight pitch moment M , angle of attack α , pitch control surface deflection δ_e and throttle setting δ_T . Their dependence

on both aerodynamic and propulsion conditions is a consequence of the aero-propulsion coupling characteristic of generic hypersonic vehicle.

3.2.2.2 Minimum required thrust

To calculate the minimum required thrust is very important for studying the steady state level flight of generic hypersonic vehicle. For steady state flight there is no acceleration and no change in altitude. During the level of accelerated flight, the wing must provide enough thrust to balance the drag force. Based on this point, the minimum required thrust can be computed by the drag polar at different flight Mach number. The minimum required thrust varies due to the variation in the parasitic drag plus the induced drag (Shahriar Keshmiri, 2007).

The total drag can be given by:

$$C_D = C_{D0} + kC_L^2 \quad (3.50)$$

where $k = \frac{1}{\pi e R_A}$, C_{D0} is the initial value of total drag coefficient. k and C_{D0} are two important parameters to calculate the minimum thrust for steady state flight at different Mach numbers.

Therefore, the minimum required thrust is approximated as:

$$T_{\min} \approx 2W \cdot \sqrt{C_{D0} \cdot k} \quad (3.51)$$

in which W is the vehicle weight.

3.2.2.3 Nonlinear longitudinal elastic model of GHV

Bolender and Doman (2005) developed the elastic model of a generic hypersonic vehicle, which includes the structural modes into the rigid body equations of motion. Andrew Clark et al. (2006) focus the interaction of the fuselage deformation with the aerodynamic and propulsion performance of vehicle. In their elastic model, the structural deformation manifests itself mainly through an effective change in the angle of attack and a perturbation in the control surface effectiveness thereby interacting with the aerodynamics and propulsion.

Different from the traditional treatment of aircraft elastic modes, which consider the elastic modes and the rigid body modes separately during aeroelastic instability analyses, their elastic model thinks of the maneuvering could excite structural modes which in turn can affect aerodynamic and propulsion performance of the vehicle due to the tightly airframe-engine integration. To develop the nonlinear longitudinal elastic model of generic hypersonic vehicle, they start by considering the fuselage as a long, slender, flexible beam. They derived a second order transfer function from the elevon normal force $P(s)$ to the body deflection angle $\theta(x,s)$ in terms of the elastic characteristics of the body (Andrew Clark, 2006):

$$\theta_i(x, s) = \frac{\phi_i(x_p) \frac{d\phi_i(x)}{dx}}{M_i(s^2 + 2\zeta_i\omega_i s + \omega_i^2)} P(s) \quad (3.52)$$

The normal force produce by the elevon P is obtained from the computational fluid dynamics data. The model data, natural frequency ω_i , and the mode shape $\phi_i(x)$ can be obtained from the finite element model of the GHV.

The two aeroelastic effects considered in their model are changes in angle of attack and elevon deflection angle. Due to the body deflection at the elevon captures the effective of vibrations on the effectiveness of the control surface, and the significant effects that the nose-induced shock wave has on the propulsion performance. The body deflection angle at the nose $\theta_i(0, t)$, is viewed as a change in the angle of attack $\Delta\alpha(s)$, and $\theta_i(x_p, t)$ at the tail, is regarded as a change in the elevon deflection $\Delta\delta_e(s)$. This can be expressed as:

$$\alpha(s) = \alpha_r(s) - \Delta\alpha(s) \quad (3.53)$$

$$\delta_e(s) = \delta_{e,r}(s) + \Delta\delta_e(s) \quad (3.54)$$

where, subscript r represents a rigid body quantity. The perturbations can be defined in terms of the rigid body elevon deflection angle in order to calculate the bounds on change in the elevon deflection $\Delta\delta_e(s)$ and change in the angle of attack $\Delta\alpha(s)$. Without derivation process, we give the dynamical relationship between $\Delta\delta_e(s)$ and elevon deflection $\delta_e(s)$ directly.

$$\Delta\delta_e(s) = \frac{T_t}{1+T_t} \delta_{e,r} \quad (3.55)$$

$$\frac{T_t}{1+T_t} = \sum_{i \in I} \frac{\phi_i(x_p) \frac{d\phi_i(x_p)}{dx} \frac{\partial P}{\partial \delta_e}}{M_i(s^2 + 2\zeta_i\omega_i s + \omega_i^2)} \quad (3.56)$$

Similarly, we represent the dynamical relationship between $\Delta\alpha(s)$ and $\delta_{e,r}(s)$ directly:

$$\Delta\alpha(s) = T_n(s)(\delta_{e,r} + \Delta\delta_e) \quad (3.57)$$

$$T_n(s) = \sum_{i \in I} \frac{\phi_i(x_p) \frac{d\phi_i(0)}{dx} \frac{\partial P}{\partial \delta_e}}{M_i(s^2 + 2\zeta_i\omega_i s + \omega_i^2)} \quad (3.58)$$

Bounds on change of the elevon deflection $\Delta\delta_e(s)$ and change of the angle of attack $\Delta\alpha(s)$ can be estimated by equation (3.59) and (3.60) respectively:

$$|\Delta\delta_e| = \left\| \frac{T_t}{1+T_t} \right\|_{\infty} \bar{\delta}_{e,r} \quad (3.59)$$

$$|\Delta\alpha| = \left\| T_n \left(1 + \frac{T_t}{1+T_t} \right) \right\|_{\infty} \bar{\delta}_{e,r} \quad (3.60)$$

in which $\|T\|_{\infty} = \sup_{\omega} |T(j\omega)|$, and $\bar{\delta}_{e,r}$ is the maximum rigid body elevon deflection.

Based on the above elastic model, take the effects of vehicle elasticity on the aerodynamics, thrust as well as control into account. Assume that the elevon deflection is affected by vehicle elasticity because of the aerodynamic force. The longitudinal equations of motion for GHV are further developed into a set of differential equations:

$$\dot{V} = \frac{T \cos \alpha - D}{m} - \frac{\mu \sin \gamma}{r^2} \quad (3.61)$$

$$\dot{\gamma} = \frac{L + T \sin \alpha}{mV} - \frac{(\mu - V^2 r) \cos \gamma}{Vr^2} \quad (3.62)$$

$$\dot{h} = V \sin \gamma \quad (3.63)$$

$$\dot{\alpha} = q - \dot{\gamma} + \sum \dot{\theta}_i(0,t) \quad (3.64)$$

$$\dot{q} = \frac{M_{yy}}{I_{yy}} \quad (3.65)$$

where, the lift L , the drag D , the thrust T , the pitching moment M , and the radius r_e from the center of earth is calculated as the same way as in sub-section 3.2.2.1. Compare the above set of differential equations with the equations (3.37)~(3.41), it is deserved to note that in equation (3.64), the time-derivation of angle of attack is assumed to depend on the nose deflection rate due to vehicle elasticity which is computed using the normal modes extracted from a finite element model.

3.3 CMAC based controller

Ying Huo (2007) proposed an adaptive fault-tolerant flight control scheme where the trajectory optimization is combined with the certainty equivalence principle to achieve the optimal path with occurrence of failures. The nonlinear equations of motions are approximated by a set of LTI models and system matrices are identified on-line with adaptive algorithms to estimate the aerodynamics after failures. The trajectory optimal control problem was then formulated to modify the trajectory commands which can be reached with the achievable control authorities after the failures.

However, the approximated LTI model structure will lose some useful information of the system, especially the highly nonlinear terms in aerodynamics. Furthermore, the approximated

linear programming introduced larger deviations from the nonlinear system, and may significantly interiorize the performance of the control system. Moreover, the system stability is not guaranteed which may lead to some severe problems. For these reasons, in recent years, a lot of works focus on the new nonlinear adaptive controller with proper mission modifications to achieve more advantaged fault tolerance capabilities for generic hypersonic vehicle.

3.3.1 CMAC based controller design

In the most widely used controller design cases, the problem is described as the commands are step-velocity and step-height signals, control the throttle setting and elevator deflection of hypersonic vehicle to track the commands, the tracking error at steady state can be guaranteed to converge to inside of a small residue set whose size can be determined.

The flight control system for hypersonic cruise vehicle based on CMAC is described in figure 3.1. In the scheme, V_d is the desired velocity of the hypersonic vehicle, and h_d is the desired height. V and h indicate the actual flight velocity and height of generic hypersonic vehicle respectively. The control inputs are throttle setting δ_t and elevator deflection δ_e .

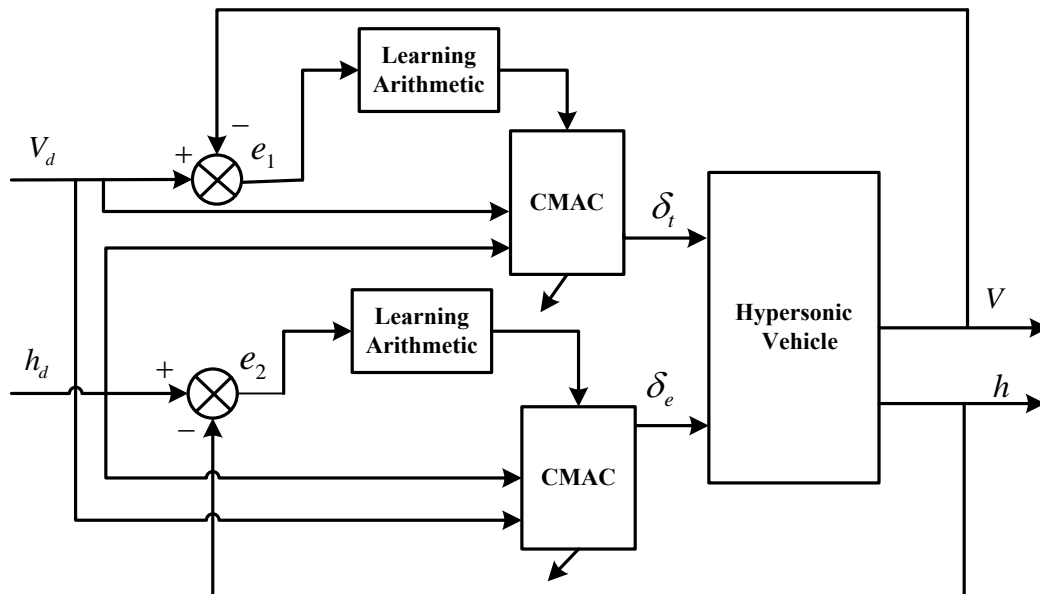


Fig. 3.1 Scheme of flight control system

On one hand, considering the coupling between flight velocity and height, and the fact that each of the throttle setting δ_t and elevator deflection δ_e will effect the change of both flight velocity V and height h . On the other hand, it is hard to know the exact explicit expression of relationship between the throttle setting, elevator deflection and flight velocity and height take all the aerodynamics into account. Therefore, CMAC neural network is chosen to design the

controller of generic hypersonic vehicle. Two two-dimension CMAC neural networks are defined separately in the universe of each input. Two inputs are flight velocity V and flight height h , and the output of the network is δ_t and δ_e respectively for each of the CMAC.

3.3.2 Simulation results and analysis

The control system design in above section is implemented using the nonlinear longitudinal dynamics of the hypersonic cruise vehicle which is developed by (Andrew D.Clark et al.,2006). The first experiment is conducted for the nominal case, where the vehicle's mathematical model and its dynamics are identical. The velocity V and altitude h are made to follow desired step commands $V_d = 3700m/s$ and $h_d = 37km$ respectively. Fig. 3.2 shows the actual outputs and the control commands to a commanded change of velocity and altitude. Fig. 3.3 shows the tracking errors of velocity and height. It can be seen that in the nominal case CMAC controller works well in order to achieve the reference commands. The tracking errors are less than 0.1% of the stable value.

The next experiment is done in the case of parameter uncertainty. As it is known, during the wind tunnel tests, accuracy of the measurements vary depending on the quality of the test and on which aerodynamic coefficient is being generated. Typically, the measurement of the pitching moment coefficient C_m is associated with some degree of uncertainty, as much as 50% on the error bound. In this simulation, the true value of C_m has been assumed, i.e.,

$$C_m = 70\% \times C_{m,0} \quad (3.66)$$

The simulation results are shown in Fig. 3.4 and Fig. 3.5. It is shown that the tracking performance requirements have been achieved. For more simulation results refers to Appendix A. The simulation results demonstrate that the CMAC controller is robust to parameter uncertainty.

From these experiments, it can be seen that the CMAC control is robust to modeling errors and uncertainties. Moreover, the CMAC control has the potential to cope with unexpected changes in the system. These uncertainties include battle damage and control surface actuation failures that will affect the systems' aerodynamic behavior and dynamic.

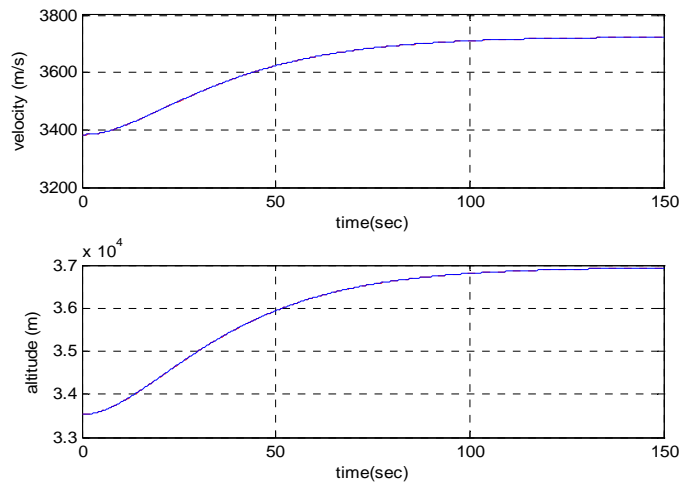


Fig. 3.2 Simulation results for the nominal case

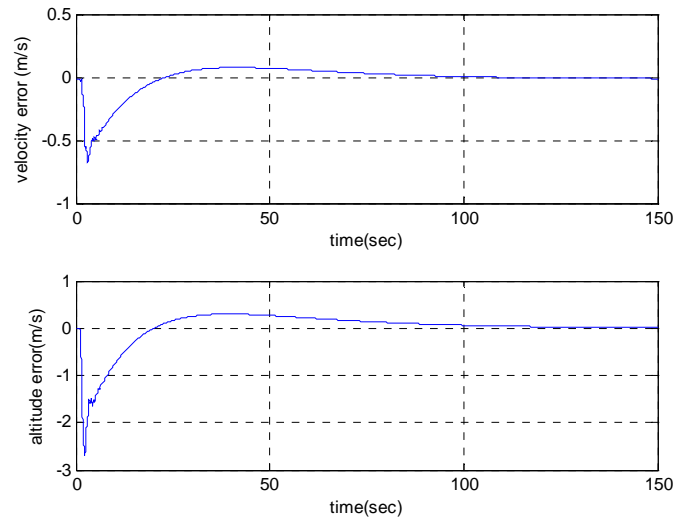


Fig. 3.3 Tracking error for nominal case

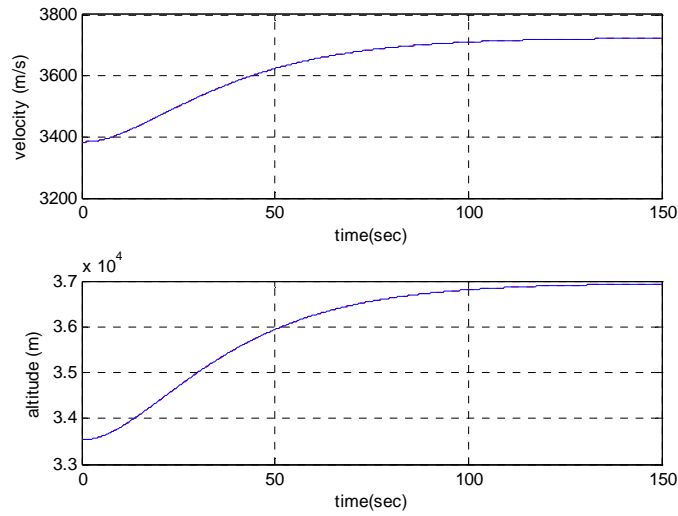


Fig. 3.4 Simulation for pitching moment uncertainty

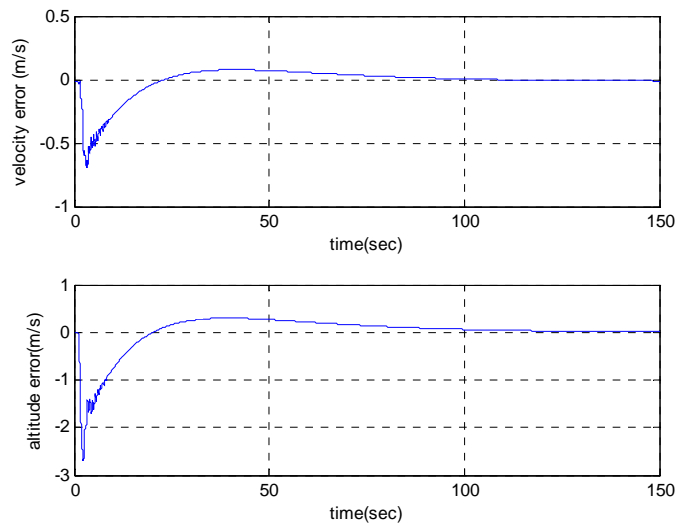


Fig. 3.5 Tracking error for pitching moment uncertainty

3.4 Summary

Flight control design for hypersonic vehicles is a very challenging task due to its sensitivity to the changes of flight condition and the difficulty in measuring and estimating the aerodynamic characteristics of the vehicle. All of these are due to the integrated engine-airframe configuration. This configuration results in significant coupling between the structure, propulsion system and vehicle aerodynamics.

Different from the other neural network based controller for generic hypersonic vehicle, in

which neural networks are used as the approximation models of the unknown dynamics of the aircraft. Our approach uses two-dimension CMAC neural networks to mapping the relationship of throttle setting and flight height and velocity, and the relation between the elevator deflection and flight height and velocity. The only assumptions made about the relation between the input flight height and velocity, and output throttle setting and elevator deflection are the unknown continuous nonlinearities, and a sufficient condition for controllability is satisfied. Simulation studies were conducted for trimmed cruise conditions of $33.6km$ and $March\ 3390m/s$ where the responses of the vehicle to a step change in altitude and airspeed were evaluated. The simulation results demonstrated that the performance requirements of the generic hypersonic vehicle were met.

Chapter 4 Fuzzy Q-learning

4.1 Introduction

Comparing with other ordinary mobile robots, the biped robots have more flexible mechanical system and can handle more complex environment. Actually, its abilities to step over static or dynamic obstacles allow the biped robots to cross uneven terrain where ordinary wheeled robots can fail. Although there are a lot of scientific papers in the field of biped and humanoid robots (M.Hackel, 2007; A.Carlos, 2007), only a few of these researches concern the path planning for biped robots (Y.Ayza, 2007; J.Chestnutt, 2004; K.Sabe, 2004). In fact, the design of a path planning for biped robots into both indoor and outdoor environment is more difficult than for wheeled robots because it must take into account the abilities of biped robots to step over obstacles. Consequently, path planning with obstacle avoidance strategy like wheeled robots is not sufficient.

The design of control strategy is a very challenging task when dealing with the stepping over dynamic obstacle problem. Kuffner et al. propose a foot step planning approach which focuses on how to build a search tree from a discrete set of feasible footstep locations corresponding to candidate stable 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. This approach has been validated on the robot H6 (J.J. Kuffner, 2001) and H7 (J.J. Kuffner, 2003). Later, this strategy has been extended for the robot Honda ASIMO (J. Chestnutt, 2005). Although the footstep planning proposed by Kuffner et al. seems an interesting way to solve the problem of the path planning for biped robots, the main drawback is this approach is operational only in the case of predictable dynamical environments (J. Chestnutt, 2005), moreover, in order to limit the computational time, the path planning should be calculated with the limitation of 15 steps (J.J. Kuffner, 2001).

In fact, in real environment, neither the velocity of the obstacle is constant nor can be predicted. However, most of the previous works that have been adopted to solve this problem focus 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 (Y. Ayza, 2007; J. Chestnutt, 2004). This chapter presents a new concept of a footstep planning for biped robots in dynamical environments. Our approach is based on Fuzzy Q-learning (FQL) algorithm. The FQL, proposed by Glorennec et al. (C. Watkins, 1992; R.S. Sutton, 1998), is an

extension of the traditional Q-learning concept (P.Y.Glorennec,1997,2000;L.Jouffe,1998), allowing to handle the continuous nature of the state-action. In this case, both actions and Q function may be represented by Takagi-Sugeno Fuzzy Inference System (TS-FIS). In the problem of humanoid robot stepping over dynamic obstacle in the sagittal plan, because of the height and unpredictable velocity of obstacle, the step length, duration time and maximum step height of robot are key factors which influence successful stepping over. Fuzzy Q-Learning algorithm has been developed to learn these crucial parameters. After a training phase, our footstep planning strategy is able to adapt the step length, step duration time and maximum step height of the biped robot only using Fuzzy Inference System.

This chapter is organized as follows. In Section 4.2, Q-learning algorithm is presented, and is applied on a path planning example. Section 4.3 Fuzzy Inference System is introduced to describe the states and actions of Q function, then Fuzzy Q-learning concept is presented. Section 4.4 is to develop the planning algorithm of step length. Based on this result, Section 4.5 presents the step duration time planning approach, and continue with these results, Section 4.6 introduce the algorithm of maximum step height planning based on Fuzzy Q-learning. Conclusions and further developments are finally set out in Section 4.7.

4.2 Q-learning Approach for Path Planning

There exist situations in Machine Learning field which the provided resources are so poor and inadequate to utilize supervised learning algorithms. In some other cases there is even no precise information about the data on which learning should be done. Moreover there might be no former sample data set available, where some unsupervised learning approaches depend on. Reinforcement Learning (RL) is developed to solve the above problem. RL is a way of learning behaviors for agents by interacting with their environment without any explicit teacher. It has its roots from psychology (Alireza Ferdowsizadeh Naeni, 2004).

Reinforcement learning is the problem faced by an agent that must learn behavior through trial and error interactions with a dynamic environment. It has been developed in the kind of discrete cases and assumes that the entire state space can be enumerated and stored in memory—an assumption to which conventional search algorithms are not tied (Richard S, 2004; Teknomo, 2005). There are two main strategies for solving reinforcement learning problems. One is to search in the space of behaviors in order to find one that performs well in the environment. The work in genetic algorithms and genetic programming are based on this strategy. The other is to use statistical techniques and dynamic programming methods to estimate the utility of taking action in states of the world. The Q learning algorithms which we work on is based on the second

strategy. Q learning is one of those for which the theory is most advanced and for which proofs of convergence exist. It does not require the knowledge of probability transitions from a state to another and is model-free (Leslie Pack Kaelbling; Mark Humphrys, 2006).

4.2.1 Q-Learning Algorithm

4.2.1.1 Presentation of Q-Learning

The Q-Learning, proposed by Watkins (Watkins C, 1989), is perhaps the most popular branch of reinforcement learning without models, by reason of its simplicity. The agent exists within a world that can be modeled as a Markov Decision Process. It observes discrete states of the world $x_t \in X$ and can execute discrete actions $a_t \in A$. Each discrete time step, the agent observes state x_t , take action a_t , observes new state x_{t+1} , and receives immediate reward r_t . Transitions are probabilistic, that is, x_{t+1} and r_t are drawn from stationary probability distributions.

The first version of Q-Learning is based on the temporal differences of order 0, while only considering the following step. The agent observes the present state x_t and executes an action a_t according to the evaluation of the return that it makes at this stage. It updates its evaluation of the value of the action while taking in account, 1) the immediate reinforcement r_t and 2) the estimated value of the new state $V_t(x_{t+1})$, that is defined by:

$$V_t(x_{t+1}) = \max_{b \in A_{t+1}} Q(x_{t+1}, b) \quad (4.1)$$

The update corresponds to the equation:

$$Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \beta[r_t + \gamma V_t(x_{t+1}) - Q(x_t, a_t)] \quad (4.2)$$

Where, 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 barycenter of the old and the new rewards, weighted by β . If there is enough learning, equation (4.2) could be written in the following form:

$$Q'(x_t, a_t) = (1 - \beta)Q(x_t, a_t) + \beta[r_t + \gamma V_t(x_{t+1})] \quad (4.3)$$

A reinforcement different from 0 can sometimes mean the end of a period, for example a mobile robot can receive the reinforcement $r = -1$ when it enters in collision with an obstacle. In this case, there is not a following state and the agent restarts a new sequence of training. The updating equating is:

$$Q'(x_t, a_t) = (1 - \beta)Q(x_t, a_t) + \beta r_t \quad (4.4)$$

4.2.1.2 Design of Reinforcement Signal

The reward function r is actually the reward or punishment function, which evaluate 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 can be classified into two big families (C. 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.

For example, the robot moves to the goal position, it chose the optimal path to reach the goal. In this case, the reinforcement signal could be -1 in all the states which the robot is not in the goal position.

- 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 case of inverted pendulum, where a failure occurs when the pole falls or the cart hits the end of the track, the value of the reinforcement signal is -1. Another case is the robot soccer game, a behavioral diversity is encouraged with the following reinforcement signal: if the team scores $r = 1$, if the opponent scores $r = -1$, in other cases $r = 0$.

4.2.1.3 Choice of Policy

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

$$a = \arg \max_{b \in A_x} Q(x, b) \quad (4.5)$$

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 an 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. Literatures (Caironi P., 1994; Meuleau N., 1996; Wyatt J., 1997) focus on how to find a compromise to solve this dilemma. Here, we listed three methods of exploration that are widely used (R.S. Sutton, 1998).

- Peseudo-stochastic Method

The action with the best value has a probability P to be chosen. Otherwise, and 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:

$$P(a|x) = \frac{\exp(\frac{1}{T}Q(x,a))}{\sum_b \exp(\frac{1}{T}Q(x,b))} \quad (4.6)$$

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

4.2.2 Path Planning

4.2.2.1 Definition of the Path Planning Problem

In complex environment distributing with obstacles “x”, the robot, starting from initial position “ I ”, finds its optimal way to reach the goal position “ G ” and avoid crashing with the obstacles at the same time. The distribution of obstacles “x” in the environment is shown in Figure 4.1, where “ I ” is the starting position, “ G ” is goal position. 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 and is enough to explain the Q-Learning algorithm based application.

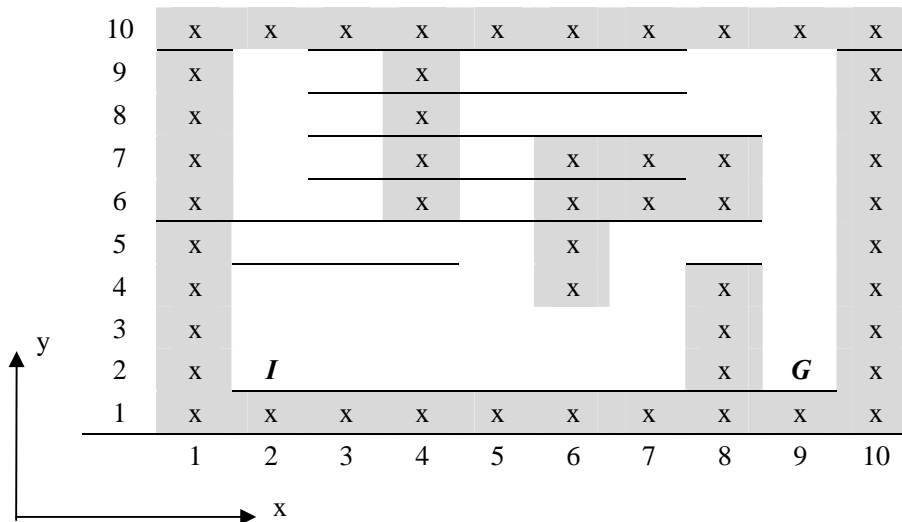


Fig.4.1 The distribution of obstacles in the environment

4.2.2.2 Simulation and results analysis

Q-Learning approach is applied to solve this path planning problem

Design of the reinforcement signal adopts the “avoiding problem” which represented in

Section 4.2.1. The reinforcement signal r equals to -1, when the robot encounters the obstacle, and r set to be 1 when the robot reaches the goal position, as the robot in other positions, the rewards equals to $R(x, y)$, which can be expressed as:

$$r = \begin{cases} -1 & \text{crash with obstacle} \\ 1 & \text{arrive at goal position} \\ R(x, y) & \text{in other positions} \end{cases} \quad (4.7)$$

In which,

$$R(x, y) = -\sqrt{(x-x_G)^2 + (y-y_G)^2} \quad (4.8)$$

represents the rewards is inverse proportion to the distance from current position to the goal position. The closer the robot from goal position, the bigger the reinforcement signal is. It will ensure that on one hand the robot can reach the destination as quickly as possible, on the other hand, it can avoid crash.

The update of $Q(x, a_i)$ matrix is according to equation (4.4), where parameter γ can be chosen in $[0, 1]$. If γ is close to 0, the robot will tend to consider only the immediate reward r for the current action. If γ is closer to 1, the agent will consider the future rewards with greater weight. We hope that the robot will forecast impact of current action to the future states, thus, γ is given to be 0.9. Learning rate β is set to be 0.1. The actions which could be chosen by the robot is $\mathbf{a} = [a_1, a_2, a_3, a_4]$, standing for step forward, step backward, step right and step left respectively.

For the choice of policy, ‘‘Pseudo-stochastic Method’’ and ‘‘Pseudo-exhaustive Method’’ is applied respectively. The simulation result of the above robot’s path planning problem is shown in Figure 4.2.

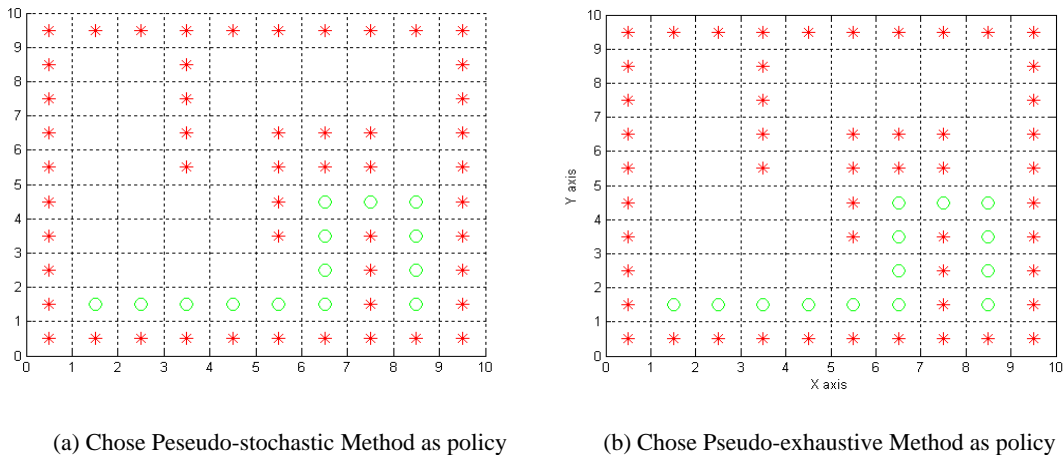
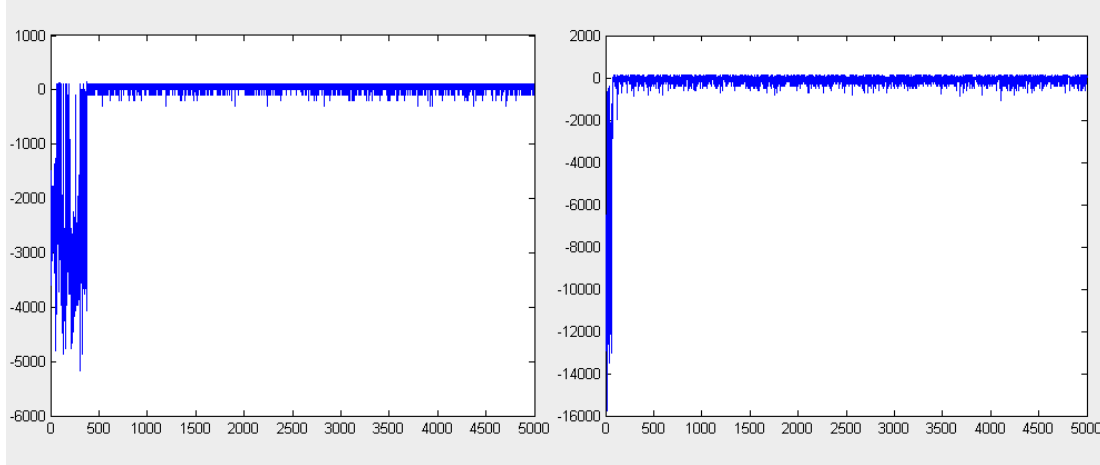


Fig.4.2 Results of path planning

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 5000, and the learning steps of each episode is 1000), calculate the sum of reinforcement signal for each episode according to equation (4.9). The results are shown in figure 4.3 for both “Peseudo-stochastic Method” and “Peseudo-exhaustive Method” chosen to be its policy.

$$Sum_r = \sum r_i \quad (4.10)$$



(a) Chose Peseudo-stochastic Method as policy (b) Chose Peseudo-exhaustive Method as policy

Fig.4.3 The sum of reinforcement signal for each learning episode

Based on the reinforcement signal designed in equation (4.7), the reward is minus for every step, therefore, the sum of the rewards of each episode is less than zero. Adopted the two policy 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 408 with Peseudo-stochastic Method, while using Peseudo-exhaustive Method the sum of reward signal convergence after 218 learning episodes. This results show experimentally that the Peseudo-exhaustive Method is better than the Peseudo-stochastic Method considering the learning efficiency. It accord with the conclusion of Caironi (1994).

4.3 Fuzzy Q-learning algorithm

4.3.1 Fuzzy inference system

Fuzzy inference systems (FIS) permit to model most of the continuous functions from a n dimension space to \mathfrak{R} . Unlike mathematical or “black-box” models, the natural language of FIS is used for the representation, which produces *if...then* rules. The order-0 Takagi-Sugeno FIS is the most useful FIS for modeling or control real systems. The TS FIS can be described by a set of fuzzy rules as follows:

$$\text{if } x_1 = A_1^j$$

$$\dots \dots$$

$$\text{and } x_i = A_i^j$$

$$\text{then } y_k = f_k(x_1, \dots, x_{N_i}) \quad (4.11)$$

in which x_i ($i=1 \dots N_i$) are the inputs of the FIS with N_i the dimension of the input space. A_i^j ($j=1 \dots N_j$) are linguistic terms, representative a fuzzy sets, numerically defined by memberships functions distributed in the universe of discourse for each input x_i . Each rule y_k of the output is a linear combination of input variables $y_k = f_k(x_1, \dots, x_{N_i})$ (f_k is a linear function of x_i , $k=1 \dots N_k$ and $i=1 \dots N_i$). The structure of TS-FIS is shown in figure 4.4 (C.Sabourin, 2007).

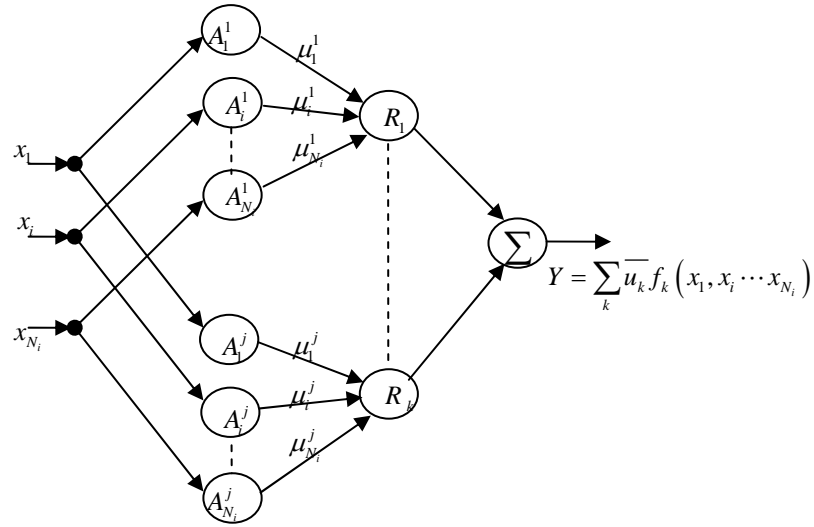


Fig.4.4 The structure of TS-FIS

The calculation of one output of TS-FIS is decomposed into three stages, including:

(1) For each condition " x_i is A_i^j ", it is necessary to compute μ_i^j which is the numerical value of the x_i input signal in the fuzzy set A_i^j . This stage is called "fuzzification".

(2) In order to determine each u_k ($k=1 \dots N_k$), apply the rule base. u_k is computed according to the following equation:

$$u_k = \mu_1^j \mu_2^j \dots \mu_{N_i}^j \quad (4.12)$$

(3) The numerical value \bar{u}_k is carried out by using the weighted average and each rule's output u_k . This stage is called defuzzification phase.

$$Y = \sum_k \bar{u}_k y_k \quad (4.13)$$

with \bar{u}_k is calculated by the following equation:

$$\bar{u}_k = u_k / \sum_{k=1}^{N_k} u_k \quad (4.14)$$

In the case of zero order Takagi-Sugeno, the output of each rule is singleton. Consequently,

$y_k = f_k(x_1, \dots, x_{N_k})$ is a constant value independent of the inputs x_i for each k ($k = 1 \dots N_k$) rules.

4.3.2 Fuzzy Q-learning algorithm

In the original Q-Learning method, it deals with discrete cases and assumes that the whole state space can be enumerated and stored in memory, because the Q values are stored in a look-up table, but this method is either unpractical in case of large state-action spaces, or impossible with continuous state space. 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. Unlike the TS-FIS In which there is only one conclusion for each rule, the Fuzzy Q-Learning (FQL) approach admit 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 FIS is build with competing actions for each rule:

$$\begin{aligned}
 &\text{If state is } S_i, \text{ then choose } a[i, 1], \text{ with } q = q[i, 1] \\
 &\qquad \qquad \qquad \text{or choose } a[i, 2], \text{ with } q = q[i, 2] \\
 &\qquad \qquad \qquad \dots \dots \\
 &\qquad \qquad \qquad \text{or choose } a[i, J], \text{ with } q = q[i, J]
 \end{aligned} \tag{4.15}$$

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 probability ε to be chosen, otherwise, an action is chosen randomly among all possible actions in the given state.

After the execution of the next computed action, the agent may update the Q value on the base of reinforcement signal. The algorithm of the FQL may be decomposed into four stages:

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

$$\alpha_i(s) = \mu_1^i \mu_2^i \dots \mu_j^i \tag{4.16}$$

- The final action $A(s)$ is computed through two levels of computation: in the first level, local action in each activated rule is determined by using EEP, and in the second level global action is calculated as a combination of all the local actions. Equation (4.17) and (4.18) give respectively the computation of the global action $A(s)$ and the corresponding $Q(S)$ value

according to the truth value $\alpha_i(s)$:

$$A(s) = \sum_{i=1}^N \alpha_i(s) \cdot a[i, j] \quad (4.17)$$

$$Q(s) = \sum_{i=1}^N \alpha_i(s) \cdot q[i, j] \quad (4.18)$$

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

$$\Delta Q = \beta[r + \gamma V_{\max}(s') - Q(s, A(s))] \quad (4.19)$$

where, same as Q-Learning algorithm, γ is a discount factor and β is a learning rate, $V_{\max}(s')$ is the maximum Q value for the activated rule at the next step time:

$$V_{\max}(s') = \sum_{i=1}^N \alpha_i(s') \cdot q[i, \max[i]] \quad (4.20)$$

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

$$\Delta q[i, j] = \Delta Q \cdot \alpha_i(s) \quad (4.21)$$

4.4 Step length planning for biped robot

The robot moves in an unknown environment, when the obstacle is static, 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. How to design the footstep planning strategy for biped robot is always the burning issues for researchers. And considering that there are seldom references concerning about the problem of obstacle moving with random velocity. The dynamic environment that we will discuss here includes both of the above two situations. In this section, our 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.

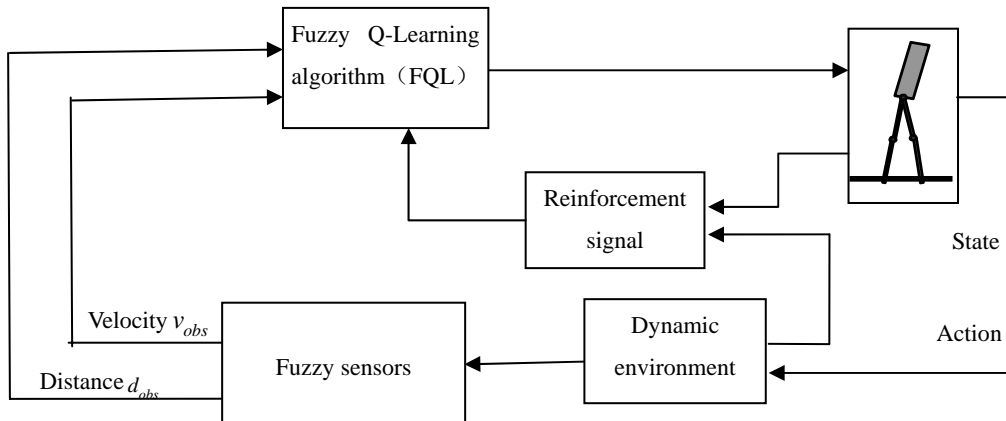


Fig 4.5 Foot step planning strategy

The step-length adjusting system based on FQL approach may be divided into four parts, Figure 4.5 show the outline of this strategy.

- First part allows simulating dynamical environment into which the robot moves.
- Second part is fuzzy sensors making it possible to compute the continuous inputs of the state.
- The third concerns the FQL algorithm allowing to compute the length of the step.
- And the fourth part gives the reinforcement signal.

In the following sub-sections, we will introduce these four parts in detail.

4.4.1 Virtual dynamical environment

Figure 4.6 shows a stick diagram of the walking sequence when the biped robot steps over an obstacle. Both of the robot and obstacle move in sagittal plan but in opposite directions. We consider 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.

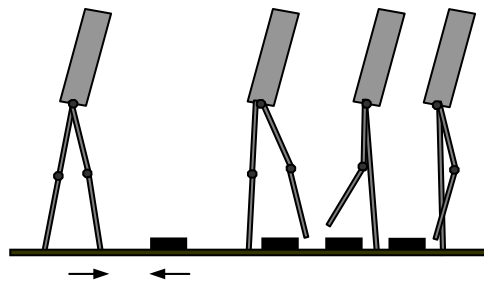


Fig.4.6 Virtual dynamical environment

The time period of a single walking cycle is divided into two major phases in the course of

dynamic walking (M.Yagi): (1) single support phase, while one leg is on the ground, the other leg is in the swinging motion. (2) double support phase, after the swinging foot reaching the ground, both of the two feet are stepping on the ground. 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, refers to figure 5.7(a). 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 (see figure 4.7(b))

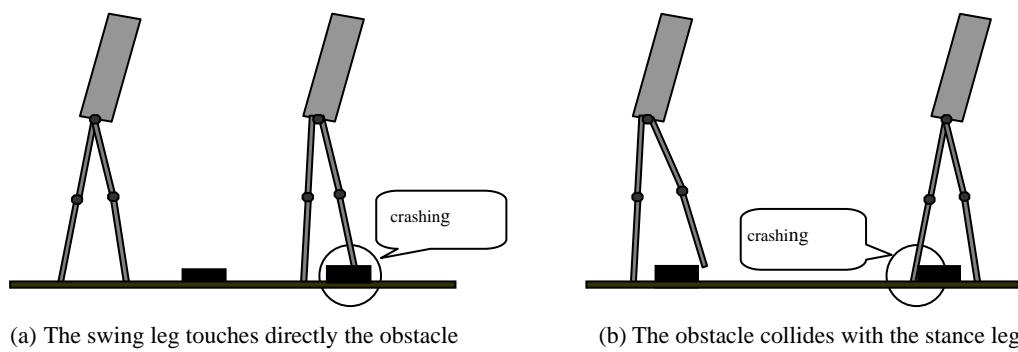


Fig. 4.7 Biped robot crashes into obstacle

4.4.2 Fuzzy sensors

The design of our footstep planning is based on both Takagi-Sugeno FIS and Q-Learning strategies. Consequently, it is necessary to use a fuzzification for each input. In the proposed approach, we use two inputs in order to perform a correct footstep planning. These inputs are the distance between the robot and the obstacle d_{obs} and the velocity of the obstacle v_{obs} . d_{obs} and v_{obs} are updated at each double support phase. d_{obs} corresponds to the distance between the front foot and the first side of the obstacle. v_{obs} is computed from the distance covered during 1s. The fuzzification of d_{obs} and v_{obs} is carried out by using respectively 6 and 11 triangular membership functions. Figure 4.8(a) and 4.8(b) gives the membership functions of the obstacle velocity and distance respectively.

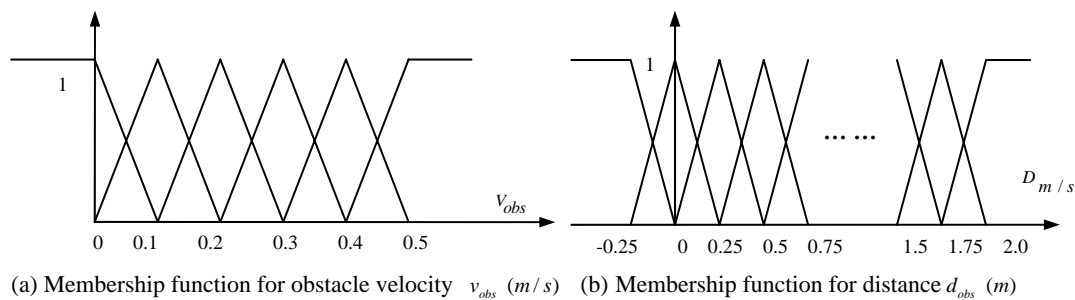


Fig.4.8 Membership functions used for the input space

4.4.3 FQL-based step length

The FQL algorithm uses a set of fuzzy rules such as equation (4.16). For the proposed problem, the number of the rules is 66 (6 and 11 membership functions for velocity and distance of the obstacle respectively). For each rules, we define 5 possible outputs which are $[0.1, 0.2, 0.3, 0.4, 0.5]m$. In fact, these outputs correspond to the length of the step. Consequently, at each step time, the goal of the Fuzzy Q-Learning algorithm is to choose one output among five possible outputs for each activated rules. It must be pointed out that the chosen output is included with a discrete set, but the real output is a real number due to the fuzzication. During the simulation, the size of the obstacle is constant but the velocity of the obstacle may be modified. At each episode, initialization of some parameters is necessary. The initial distance between the biped robot and the obstacle is always equal to $2.5m$. The velocity of the obstacle is chosen randomly from the interval $[0, 0.4]m/s$. During one episode, the step length of the robot is computed using the FQL algorithm described in section 4.3. 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 influence of this parameter will be studied in the next section. The discount factor γ and the learning rate parameter β are equal to 0.8 and 0.1 respectively. This parameters have been chosen empirically after several trials in order to assure a good convergence of FQL algorithm. The probability P_ϵ is equal to 0.1, which means that the random exploration is privileged during the learning phase.

4.4.4 Reinforcement signal

The reinforcement signal provides the information in terms of reward or punishment. Consequently, the reinforcement signal informs the learning agent about the quality of the chosen action. 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. In addition, the position of the foot just before the stepping over is very important too. On the base of these considerations, we designed reinforcement into two parts.

Firstly, if $x_{rob} < x_{obs}$, where x_{rob} and x_{obs} give respectively position of the robot and 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 first case, r is computed with the following equation:

$$r = \begin{cases} 0 & \text{if } (x_{rob} \leq (x_{rob} - 1.2v_{obs}\Delta t)) \\ 1 & \text{if } (x_{rob} > (x_{rob} - 1.2v_{obs}\Delta t)) \\ & \text{and } (x_{rob} \leq (x_{rob} - 1.1v_{obs}\Delta t)) \\ -1 & \text{if } (x_{rob} > (x_{rob} - 1.1v_{obs}\Delta t)) \end{cases} \quad (4.22)$$

x_{rob} and x_{obs} are updated after each action. $v_{rob}\Delta t$ represents the distance covering by obstacle during the time Δt . As the duration of the step is always equal to $1s$, Δt is always equal to $1s$.

Secondly, if $x_{rob} \geq x_{obs}$:

- $r = -2$, 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 (5,23):

$$r = \begin{cases} -2 & \text{if } (x_{rob} > (x_{rob} - L_{obs})) \\ & \text{and } (x_{obs} \geq (x_{rob} + L_{obs})) \\ 2 & \text{else} \end{cases} \quad (4.23)$$

where L_{obs} is the size of the obstacle.

4.4.5 Simulation results and analysis

In this sub-section, we present the main results related to the step length planning based on FQL approach by using MATLAB software. It must be noticed that our goal is to design a control strategy allowing to give 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, we consider only flat obstacles in the following simulations.

Firstly, we present results about the convergence of the algorithm according to the number of episodes during the learning phase. Secondly, we study the influence of the size of the obstacle. Finally, we show two examples of the foot step planning.

4.4.5.1 Convergence of the algorithm

During the learning phase, the goal of the learning agent is to find the best rules in order to make the biped robot step over the obstacle. On the base of the previous description, we trained the Q matrix for different number of episodes. After a sufficient training, we test the footstep planning approach for 1000 velocity samples covering uniformly the input range $[0,0.4]m/s$. For each simulation, the size of the obstacle is equal to $0.2m$. Figure 4.9(a), 4.9(b) and 4.9(c) show respectively results for 100, 1000 and 10000 episodes.

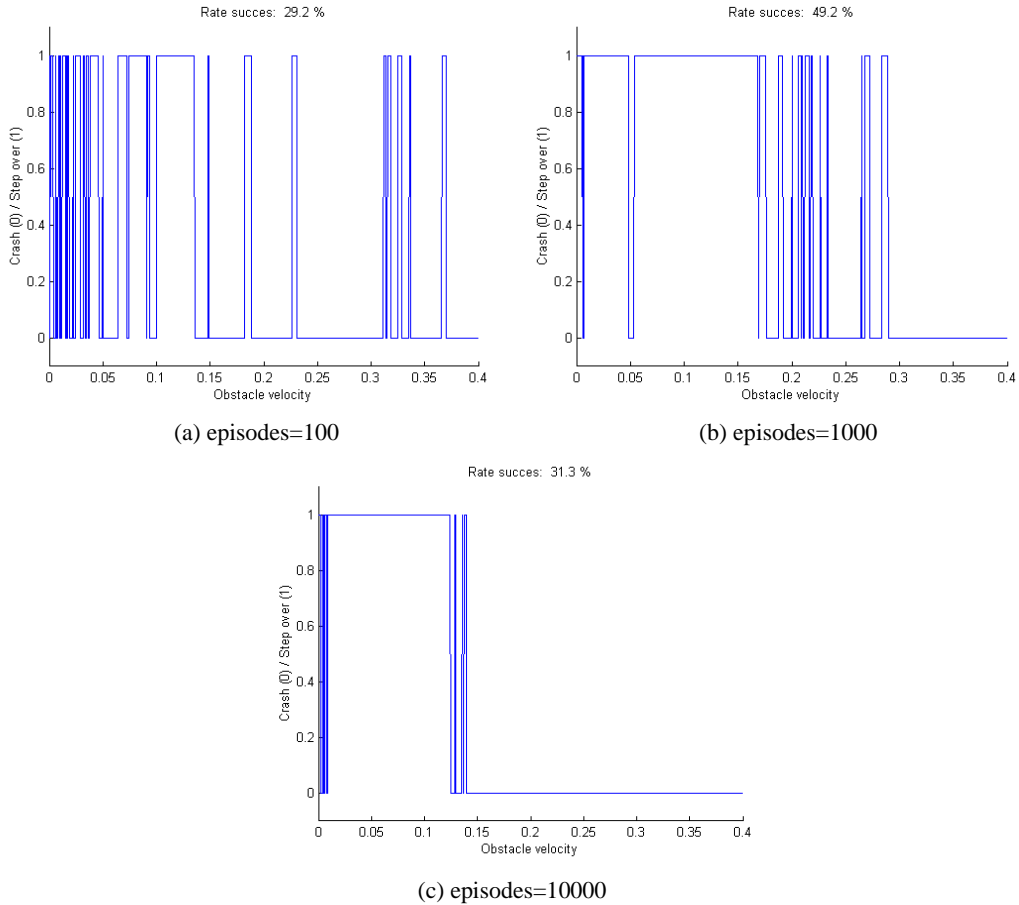


Fig.4.9 Success rate according to the number of episodes

When the robot can step over the obstacle successfully, the results is 1, otherwise it is 0. The success rate corresponds to the ratio between the number of successes and the totality of trials. It must be pointed out that more the number of episode increases, more learning the agent needs for converging towards an optimal compromise. This solution implies that whether the velocity of the obstacle is greater than a threshold ($0.12m$ approximately), no solution exists. Consequently, this approach is interesting because it is possible to determine a limit of v_{obs} which can not be exceed.

4.4.5.2 Influence of the obstacle size

In the previous simulation, the size of the obstacle always equals to $0.2m$. In this sub-section, we study the success rate according to the size of the obstacle L_{obs} . Figure 4.10(a), (b) and (c) show respectively for L_{obs} which equals to $0.1m$, $0.2m$ and $0.3m$, the repartition of both success and fail over the input range v_{obs} . Table 4.1 concludes the results about success rate for four sizes of obstacle. It must be pointed out that more the size of obstacle is large, more the success rate is weak. This is interesting information about the abilities to the FQL algorithm to solve the proposed problem. Consequently, after the learning phase it is possible to give information about the limitation of the proposed approach.

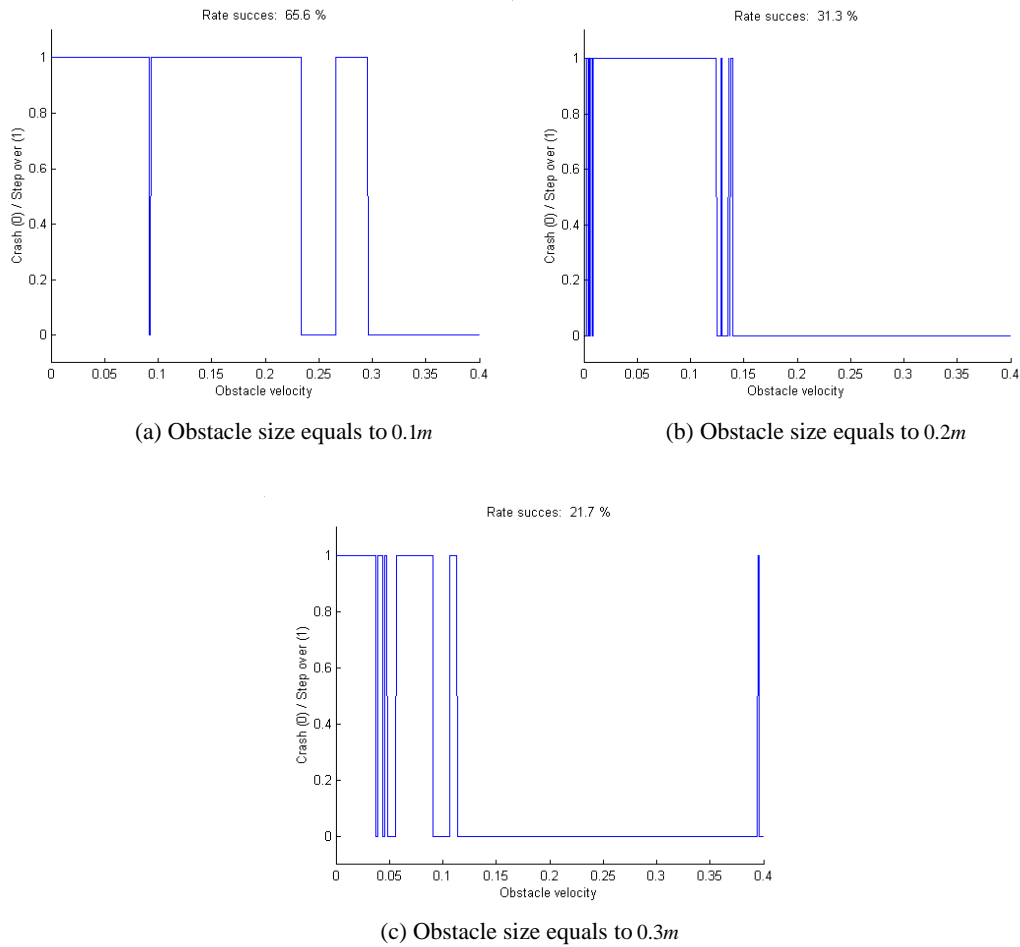


Fig. 4.10 Success rate according to the size of obstacle (10000 episodes)

Table 4.1 Success rate according to obstacle size

Size (m)	0.1	0.2	0.3	0.4
Constant v_{obs}	65.6	31.3	21.7	4.8

4.4.5.3 Foot step planning examples

Figure 4.11 shows a view of a foot step sequence when the robot steps over an obstacle of $0.2m$ length, which is moving with constant velocity. Rectangle indicates the obstacle and spots indicate the two positions of the left and right foot for each step. Table 4.2 gives the step length for each step. It 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.

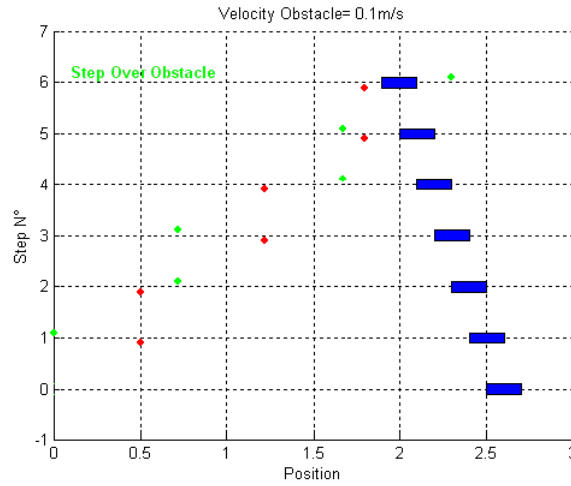


Fig.4.11 Successful footstep planning when $v_{obs} = 0.1m/s$ and $L_{obs} = 0.2m$

Table 4.2 Length of the step L_{step} when $v_{obs} = 0.1m/s$

Step	1	2	3	4	5	6
L_{step}	0.50	0.22	0.50	0.45	0.13	0.50

However, one of the most interests of our approach is also operational when the moving is unpredictable. Figure 4.12 shows the footstep sequence when the obstacle moves with a random velocity. The obstacle velocity is carried out by the sum of a constant value which equals to $0.1m/s$ and a random value included in $[0 \dots 0.3]m/s$. Table 4.3 gives v_{obs} and L_{step} for each step. The size of the obstacle is equal to $0.1m$. It must be pointed out that the control strategy allows the robot to adapt the length of the step automatically according to the obstacle velocity, thanks to FQL algorithm. For 1000 trials realized in the same conditions, the success rate is equal to 85% approximately. This is very interesting because our strategy allows the control to increase the robustness.

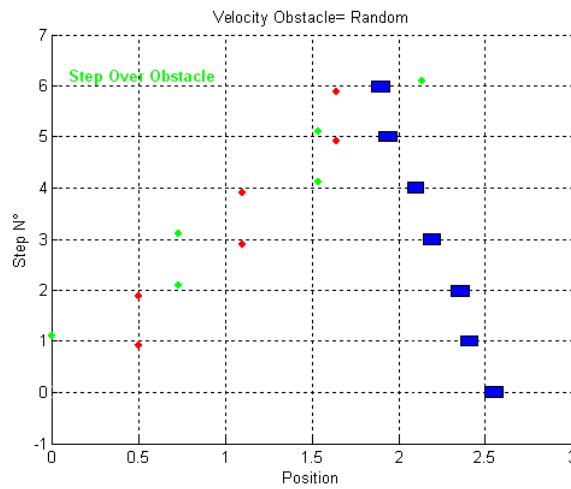


Fig.4.12 Successful footstep planning when v_{obs} is random $L_{obs} = 0.1m$

Table 4.2 Length of the step L_{step} when $v_{obs} = 0.1m/s$

Step	1	2	3	4	5	6
v_{obs}	0.14	0.05	0.16	0.10	0.16	0.04
L_{step}	0.50	0.23	0.37	0.44	0.10	0.50

4.5 Step duration time planning for biped robot

In the last section, we only adjust the step length for the biped robot to step over the dynamic obstacle. However, when the obstacle size is bigger, the success rate which is concluded in sub-section 4.4.5.2 is not desirable. In this section, we introduce the “step duration time”, which can be defined as the lasting time of one walking period for the biped robot. As it is on assumption that the double support phase is very short, the step duration time can be calculated as the time period of from leaving to landing on the ground of swing leg. The step duration time and step length will be planned at the same time, in order to increase the success rate of stepping over.

4.5.1 FQL-based step duration time planning

The step duration time planning approach is developed based on the Fuzzy Q-Learning algorithm introduced in section 4.3. During a stride circle, the robot needs to adjust the foot landing point, and modify the step time at the same time. Considering the step length L_{step}^j and step duration time T_{step}^j in one walking period as an “action pair”, for each fired rule, the learning system has to choose one action pair in all possible actions pairs (L_{step}^j, T_{step}^j) , (the number of the possible actions pairs equals to N_l). We choose Pseudo-exhaustive Method as the searching policy here. The action pair with best evaluation value has a probability $P(a/x)$ to be chosen. The output of the Fuzzy Inference System is to be the local action pair of the activated rule. After implement of the next action pair $(L_{step}^{j+1}, T_{step}^{j+1})$, the Q matrix value at present time $Q(x, L_{step}, T_{step})$ is calculated according to the reinforcement signal and present state. After the application of the new action pair (L_{step}^j, T_{step}^j) , the temporal difference error is computed as equation (4.19). The structure of this approach is described in figure 5.13.

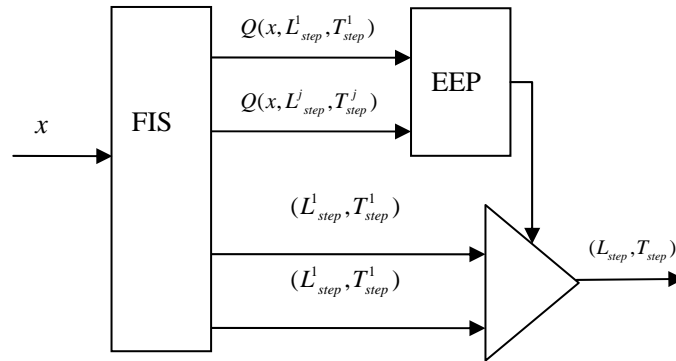


Fig.4.13 FQL-based step duration time planning

For design of the reinforcement signal, the biped robot not only considers the impact of step duration time at present time, but also the influence to the next step (for example, if the robot will crash into the obstacle or there is not enough time for avoiding in next step). Furthermore, the reinforcement signal has to integrate the information of step length. Take into account the above factors, the reinforcement signal can be designed as:

When $x'_{rob} < x'_{obs}$, which means that the chosen present action pair (L^j_{step}, T^j_{step}) will leads the robot far away from obstacle in the next step. The robot is not in the state of stepping over.

- if $x'_{rob} \leq (x'_{obs} - n \cdot T_{step} \cdot v_{obs})$, the robot is still far away from the moving obstacle in next step, r equals to 0.
- if $(x'_{rob} > (x'_{obs} - n \cdot T_{step} \cdot v_{obs})) \& (x'_{rob} \leq (x'_{obs} - m \cdot T_{step} \cdot v_{obs}))$, the present state of robot is appropriate that makes the robot can step over in next step.
- if $x'_{rob} > (x'_{obs} - m \cdot T_{step} \cdot v_{obs})$, the robot is too close to the obstacle and it will leads to crash in the next step, r equals to -1.

When $x'_{rob} \geq x'_{obs}$, which means that the robot is just in the state of stepping over. The chosen action pair (L^j_{step}, T^j_{step}) determines if crash happens directly.

- if $(x'_{rob} > (x_{obs} + l_{obs})) \& (x'_{obs} > (x_{rob} - L_{step}))$, the robot can step over the obstacle successfully, r equals to 2.
- otherwise, the robot crashes into obstacle.

In the above description, l_{obs} is the length of obstacle, x_{rob} and x'_{rob} represent respectively the present position and the following position of the biped robot. x_{obs} and x'_{obs} are respectively the corresponding position of obstacle to the above two states for robot. Both of m and n are constants, which have physical meaning of evaluating if the present action pair (L^j_{step}, T^j_{step}) and present position is appropriate for executing the next action pair in order to avoid collision. According to the above contrive of reinforcement signal, both of the step length and step duration time chosen at present time will influence not only the present state, but also the state in the future.

4.5.2 Simulation results and analysis

As the previous section, Takagi-Sugeno FIS is used to fuzzificate the input signal d_{obs} (distance from robot to obstacle) and obstacle velocity v_{obs} . Their membership function is shown in figure 4.8(a) and 4.8(b). But, we chose the 5 triangular membership functions for each of them, so the number of rules of fuzzification is 25. For Q-Learning, the number of episode is assumed to be 1000. The initial distance between robot and obstacle is around $1.25m$, which can be expressed as $(1.25 + \Delta\delta)m$, $\Delta\delta$ is a small difference. Suppose the obstacle length is $0.1m$, and its velocity can change randomly within $[0 \ 0.4]m/s$. As for each activated rule, define five possible step length $[0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5]m$, and five possible step duration time

[0.1 0.2 0.3 0.4 0.5]s. It must be noticed that the combination of step length and step time is randomly. So the number of action pair is 25.

In the example of obstacle moving with constant velocity $v_{obs} = 0.15m/s$, figure 4.14 shows the footstep of biped robot when it adjust the step length and step duration time at the same time. It can be seen from the figure, the robot has to adjust the step length and step time for three steps. And before the modulating, the robot can keep on walking with the same step length $L_{step} = 0.48m$ and step time $T_{step} = 0.213s$. When the obstacle velocity is random, figure 4.15 represent the footstep and step time planning results. In this case, the biped robot has to adjust the step length for four steps, but the step time has to be modified for 6 steps. The divide of step length to step time is velocity. In fact, in this approach, the stepping over obstacle is done by planning landing point of swing leg and its velocity.

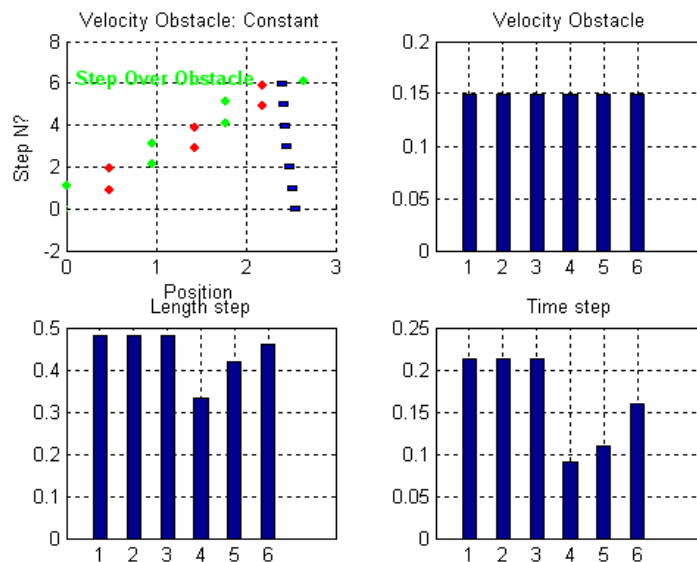


Fig.4.14 Step length and step duration time planning for biped robot stepping over obstacle with constant velocity $v_{obs} = 0.15m/s$.

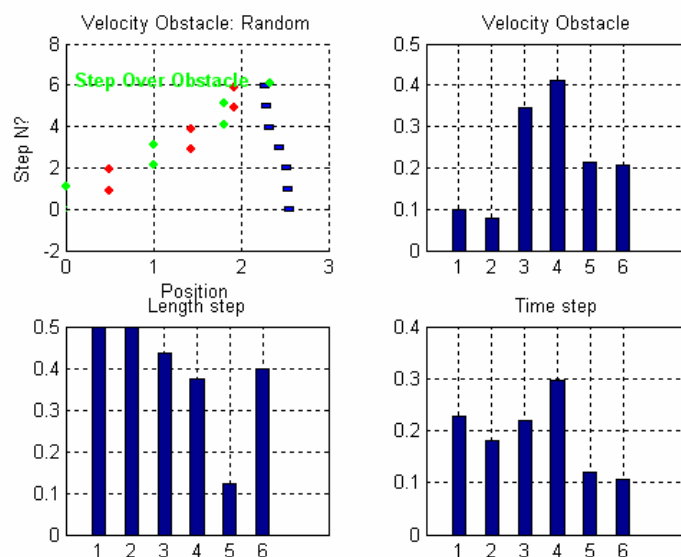


Fig.4.15 Step length and step duration time planning for biped robot stepping over obstacle with random velocity

Simulation is done when the obstacle length l_{obs} equals to $0.1m$, $0.2m$ and $0.3m$ respectively, and compute the success rate in these three cases. As the same, 1 stands for stepping over successfully and 0 represents failure. Simulation results are shown in figure 4.16(a), (b) and (c). While the obstacle length is $0.2m$, we can easily figure out that the biped robot can step over all the obstacles whose velocity is smaller than $0.32m/s$. Comparing this result with 4.10(b), the robot can only avoid crashing when the obstacle moves slower than $0.15m/s$. We can get the conclusion that the success rate increases obviously, when adjusting the step length and step duration time (velocity) at the same time.

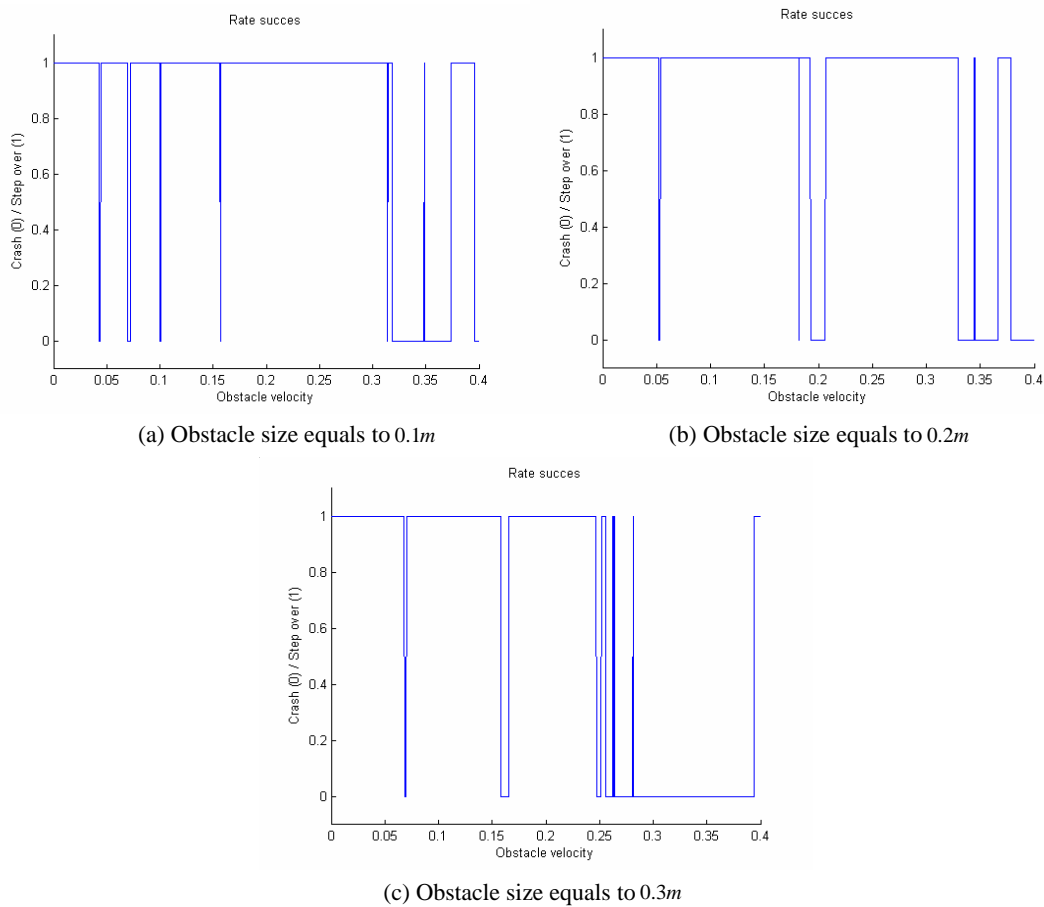


Fig. 4.16 Success rate according to the size of obstacle (10000 episodes)

4.6 Maximum step height planning

The “maximum step height” is defined as the vertical distance from highest point of the swing trajectory to the ground. In the previous discussion, the biped robot is supposed to keep one step height during the walking. However, in the unknown environment, the height of obstacle is unpredictable. Through the following example, we will illustrate why it is necessary to planning

the step height at the same time. Continue with the problem we described in the last section, the obstacle moves randomly with velocity including in $[0 \ 0.4]m/s$. If only adjusting the step length and step duration time, from the point of footstep planning, the robot can step over the obstacle successfully, refers to figure 4.17. But considering the trajectory of swing leg, the swing leg crashes with the obstacle in the air, see figure 4.18 (the red stick stands for the stance leg, the green stick represents for the swing leg, and the blue arch is the swing leg trajectory).

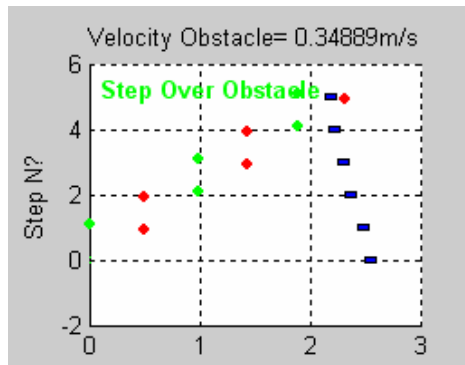


Fig.4.17 Footstep planning result

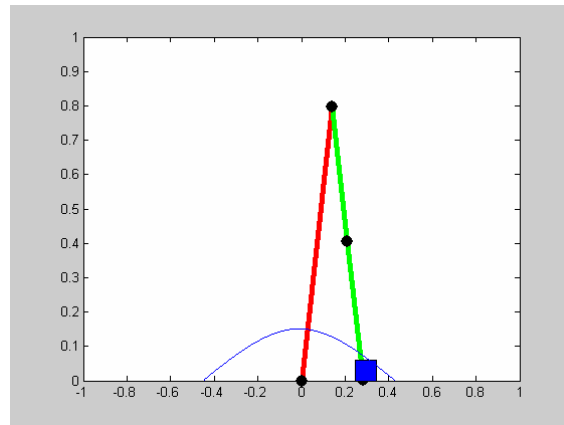


Fig.4.18 Swing leg trajectory during stepping over

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 (see figure 4.19). 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. Consequently, adjusting the maximum step height according to the obstacle height is necessary for biped robot to avoid collision. In the actual problem, the obstacle height is not alternatively. Thus, the biped 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.

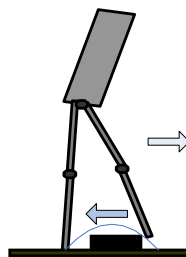


Fig. 4.19 The trajectory point of swing leg is higher than the obstacle at the same time.

4.6.1 FQL-based maximum step height planning

As previous description, on one hand most of the trajectories of swing leg are calculated by interpolation of the planned characteristic point on the track. And in order to keep the biped robot stable walking, the swing trajectory is usually a smooth arch. On the other hand, 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. Therefore, in order to step over the obstacle successfully, the biped robot has to modify the step length, step duration time and maximum step height at the same time.

The trajectory arch is determined by the starting point, landing point and step height of the swing leg. The difference between the landing point and starting point is just the step length, which has already been planned in section 5.4. The maximum step height is designed separately, based on Fuzzy Q-Learning algorithm. The FQL-based maximum step height planning strategy can be described as follows: Regarding the step length L_{step} , distance between obstacle and robot d_{obs} , together with the obstacle velocity v_{obs} as the input signals of the fuzzy sensors, but they are designed separately. The action pair (L_{step}^j, T_{step}^j) is learned by FQL algorithm I firstly. The chosen action pair and obstacle velocity at present time allow the FQL algorithm II to compute the maximum step height. Therefore the learning can be divided into two stages, the FQL-based maximum step height planning strategy is shown in figure 4.20.

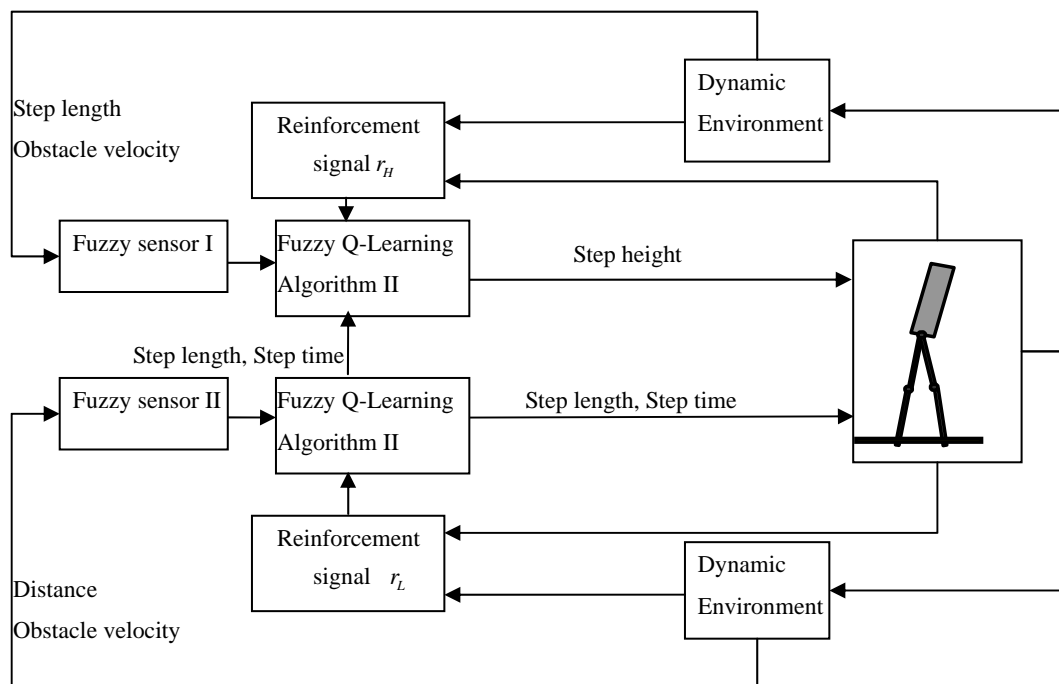


Fig.4.20 FQL-based maximum step height planning strategy

The fuzzification of step length L_{step} is carried out by using 11 triangular membership functions, refers to figure 4.21. We continue to use the fuzzy sensors for distance d_{obs} and obstacle velocity v_{obs} designed in section 4.4.3.

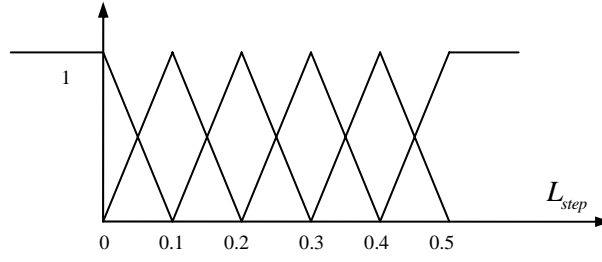


Fig.4.21 Membership function of step length L_{step}

Step length L_{step} and obstacle velocity v_{obs} are updated before the action of stepping over. Following the same idea, the reinforcement signal for learning step length and step time r_L and reinforcement signal for the FQL-based step height planning have to be designed separately. It is pointed out that in order to keep itself reach good stability, there is no need for the biped robot to adjust the step height H_{max} every step, it can holding the same H_{max} before stepping over. Therefore, the reinforcement signal r_H does not have to consider the influence of present action to the future state, but only take into account the impact of chosen action H_{max} to present state. The reinforcement signal r_H can be described as follows:

Firstly, if the abscissa of the trajectory point $P_x(t)$ superposition to any point of the obstacle abscissa $x_{obs}(t)$,

- $r_H = 1$, if vertical coordinate of this trajectory point $P_y(t)$ is bigger than the height of obstacle h_{obs} ,
- $r_H = -1$, if vertical coordinate of this trajectory point $P_y(t)$ is smaller than the height of obstacle h_{obs} , collision occurs,

Secondly, if there is difference between the abscissa of the trajectory point $P_x(t)$ and the obstacle abscissa $x_{obs}(t)$ 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.

- $r_H = 0$

4.6.2 Simulation results and analysis

In the design of Fuzzy Q-Learning algorithm II designed for planning maximum step length, the learning rate β equals to 0.8 and the discount factor γ is taken to be 0.1. The episode of learning is 5000 in order to make sufficient learning. The goal of learning agent is to search the optimal rule leads the biped robot to step over obstacle whose height is within certain range. Suppose that the obstacle velocity can be changed within $[0 \ 0.4]m/s$, but randomly. Five possible step length is $[0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5]m$ and the presumable step duration time is $[0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5]s$. On the assumption that the original maximum step height H_{max} is

0.08m , the robot keeps the same step height before stepping over. H_{\max} can be chosen within [0.08 0.1 0.125 0.15 0.175 0.2]m .

Step length L_{step} and step duration time T_{step} are designed according to section 4.5. The FQL-based maximum height planning is based on 4.6.1. As the output is defuzzicated by the Fuzzy Inference System, the practical output $[L_{step}, T_{step}, H_{\max}]$ is a group of real number. Figure 4.22 gives the simulation results when the obstacle moving with $v_{obs} = 0.15m/s$, while figure 4.23 shows the foot step sequence in the random velocity case. As the previous, rectangle indicates the obstacle and the spots indicate the two positions of the feet for every step.

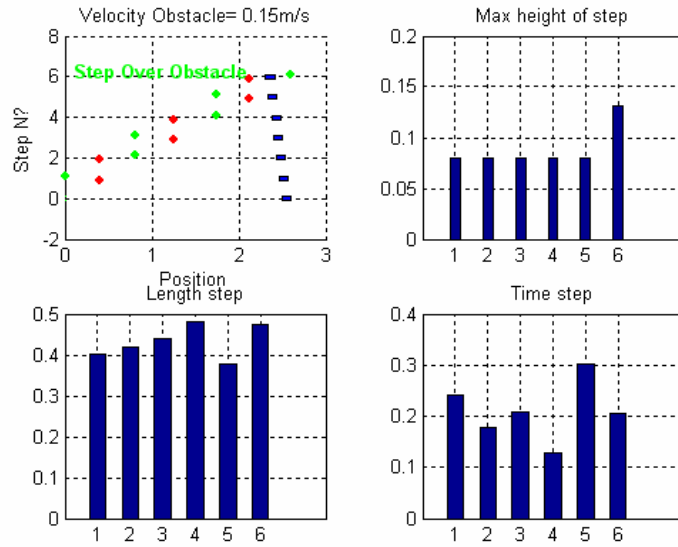


Fig.4.22 Results of step length, step duration time and maximum step height, when obstacle moving with constant velocity, $d_{int} = 2.5m$

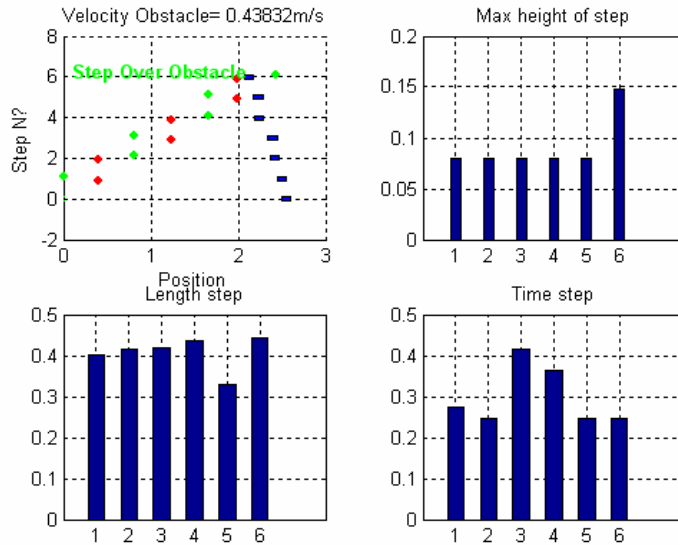


Fig.4.23 Results of step length, step duration time and maximum step height, when obstacle moving with random velocity, $d_{int} = 2.5m$

It can be seen from the above results, the last step allows the biped robot to avoid constant

velocity or 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 $[L_{step}^j \ T_{step}^j]$ for six steps, but the maximum step height is only adjusted from $0.08m$ to $0.148m$ in the last step. The Q matrix corresponding to the step height has only to be trained for the last step, therefore, the training time of the FQL-based footstep planning decreases dramatically.

The above results are on the assumption of the initial distance d_{obs} between the robot and obstacle equals to $1.5m$. 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_{obs} is set to be $1.5m$, the biped robot can avoid collision only by correcting 4 steps (see figure 4.24), while in the case of initial distance is $0.1m$, three steps of modification is enough for it stepping over successfully (refers to figure 4.25).

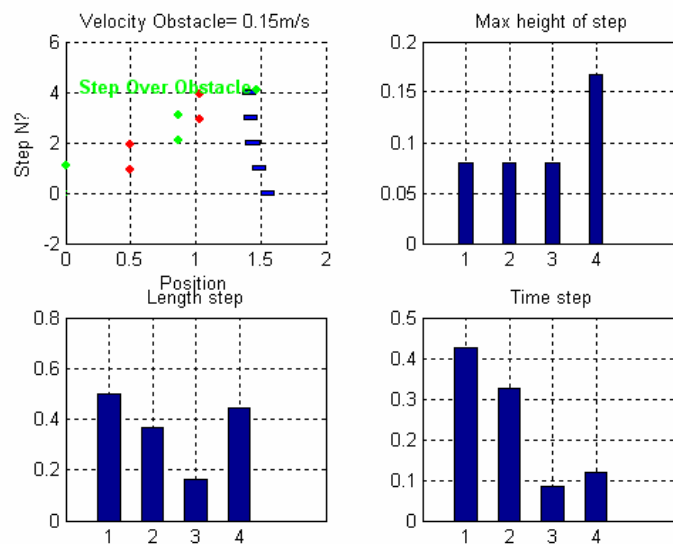


Fig.4.24 Results of step length, step duration time and maximum step height, when obstacle moving with constant velocity, $d_{int} = 1.5m$

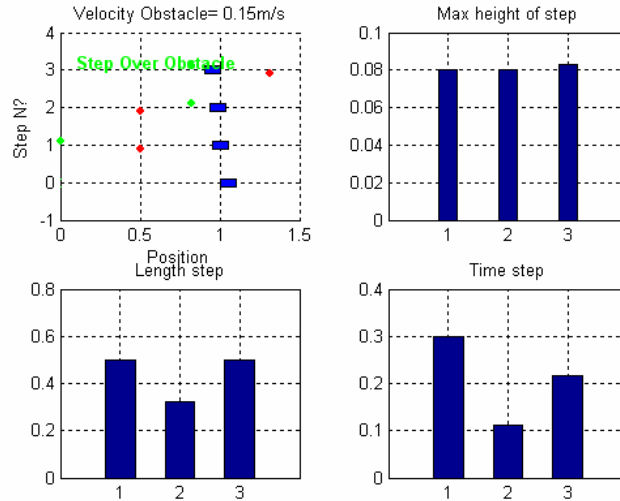


Fig.4.25 Results of step length, step duration time and maximum step height, when obstacle moving with constant velocity, $d_{int} = 1.0m$

4.7 Summary

Firstly, we have presented a footstep planning approach allowing the biped robot to step over dynamic obstacles by adjusting the step length and step duration time. Our footstep planning strategy is based on a fuzzy Q-Learning concept. The step length and step duration time 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 predictable and unpredictable motion of the obstacle.

However, on one hand, in unknown environment, the height of obstacle is unknown. On the other hand the dynamic obstacle keeps on moving while the swing leg uplift. Therefore, considering the swing leg will not crash into obstacle in the air, the maximum step height has to be modified at the same time. A FQL-based step height planning approach has been developed separately. The learning system makes use of the obstacle velocity and chosen action pair training Q matrix. The maximum step height is trained only for the last step, thus, the training time of Q matrix decreases dramatically. The simulation results show that the biped robot only needs to adjust the step height during the last step, and it can keep on walking with the same step height before stepping over.

The investigations in this chapter show a real interest of this approach because:(1) The computing time is very short. After the learning phase, the footstep planning is based only on two

FIS (one for action pair of step length and step duration time, another for maximum step height).
(2) This footstep planning approach is valid for both static and dynamic obstacles. (3) The footstep planning is operational for both predictable and unpredictable dynamical environment allowing the control to increase the robustness.

Chapter 5 Obstacle avoidance strategy for biped robot in dynamic environment

5.1 Introduction

The control strategy for the locomotion of biped robot can mainly be divided into two categories. One is based on the mechanical model of kinematics and dynamics of biped model, this means that one has to know the internal structural parameters of biped robot precisely, which needs nicety measurement of the joint angles, velocity and acceleration and the estimation of contact force between the feet and ground. Besides, the control strategy based on precise mathematical model usually requires mass computation, and it causes the implement difficulty in on-line control (K.Loffler et al., 2003). Therefore, predigest mathematical model is often used and the on-line joint trajectory computation grounds on the ZMP stability rules (Q. Huang, 2001; S. Kajita, 2003; M.Vukobratovic, 2004). The second category of control strategy mainly adopts the computing technology (A.L.Kun, 2000), such as fuzzy logic (C.Zhou, 2000), neural network genetic algorithm (F. Yamasaki, 2002), and the learning machine (A.Brnbrahim, 1997) (J. Nakanishi, 2004). The advantage of this control strategy is that one does not have to know the nicety mechanical structure of biped robot. Furthermore, the learning process can be done in both on-line and off-line situations. This point is very important, because the learning capacity can improve the autonomous of biped robot (C.Sabourin, 2008).

The design of a control strategy allowing the biped robots to move in autonomous manner within dynamical environment is a challenging task. It involves solving a number of problems concerning, on the one hand, the desired joint trajectories tracking or dynamic stability control, which is called the low-level control; and on the other hand, the gait planning or path planning, which is defined as the high-level control. Generally, the low-level control using feedback structure is based on proprioceptive and/or exteroceptive information (reactive control). And the high-level control needs to use a predictive approach (planning). The goal of this high-level control is to anticipate the moving of the robot by using an exteroceptive perception of the environment. But the effectiveness of this approach is due to optimization and/or learning process. Today, although there are a large number of research works concerning the control of biped robot (M.Hackel, 2007), however, only a few studies involve the high-level control dealing with the path planning and the obstacle avoidances. Consequently, it seems necessary to design complementary approaches in order to increase the biped robots' autonomy. It is pertinent to

emphasize that the specificity of biped robots is their ability to choose the landing point of their feet.

To design the joint angle profiles that describe human-like locomotion of the biped is a challenging problem. A well-structured approach of designing the joint angle profiles that ties the resulting gait patterns with the physically coherent parameters is desirable. Hurmuzlu (1993a) developed a systematic approach that can be followed to formulate objective functions. Such objective functions were cast in terms of step length, progression speed, maximum clearance of the swing leg, and the support knee bias that could be used to prescribe the gait of a planar five-link bipedal robot during the single support phase. The objective functions are modified by a constraint function that keeps the mechanical energy as a constant. A major challenge of using this systematic approach to obtain the joint angle profiles is solving a set of equations combined with differential and algebraic equations. Which are from the constraint function. There is no general way to solve this combination of differential and algebraic equations.

In this chapter, we still use the developed Fuzzy Q-Learning algorithm to solve the problem of biped robot stepping over the obstacle which is moving with random velocity. Step length, duration time and maximum height of every step have been recorded after learning process. After interpolating the starting point, ending point and maximum step height with cubic spline interpolation, desired foot trajectory of swing leg can be obtained. Then, with inverse kinematics, the joint angle profiles can be calculated according to the geometrical relationship between pitch angles. The footstep planning, cubic spline interpolation for generating the foot trajectory, together with inverse kinematics for generating the joint angle profiles constitutes the high-level control.

The low-level control allows both to generate joint trajectories and to control the tracking of these desired trajectories. It can be decomposed into three parts: several CMAC neural networks for approximating the joint profiles for swing leg, modification of the pitch angle of the trunk for adjusting the average velocity, and PD control for tracking the desired joint angle.

This chapter is organized as follows: Section 6.2 introduces the high-level control design strategy. Generation of trajectory for swing leg and generation of joint angle are presented in each sub-section. In last sub-section of section 6.2, we test the proposed high-level control strategy by a simulation example. The low-level control is introduced in Section 6.3, which includes the description of CMAC neural networks for approximating the joint profiles and pitch angle modification. The simulations result of our control strategy is given in last section 6.4.

5.2 Motion equations for five-link biped robot

Because of the complexity of linkage systems with many degrees of freedom, constructing the mathematical modeling for the locomotion of biped robot is always a challenging problem. In general, the biped locomotion consists of two stages of mathematical modeling: kinematics modeling and dynamics modeling. The goal of kinematics modeling is to determine and analyze the motion relationship between each part of the body, given the locomotion of each joint. The dynamic behavior of a biped locomotion system is described in terms of the time rate of change of the linkage configuration in relation to the joint torques. Furthermore, in the study of the dynamics of biped robot locomotion, it can be classified into two research categories: one approach is the forward dynamic problem; the other is inverse dynamic problem. For the forward dynamic problem, the moments applied to the system serve as the system inputs and the solution found is the system kinematics. In the inverse dynamic problem, the kinematics data is used as the system inputs to find the forces and moments applied to the system. The purpose of solving the inverse dynamic problem in locomotion is to obtain information on the joint moments and the reaction forces at the joints of robot lower extremities (Siegler et al. 1982).

As for the biped robot walking motion, abundant degrees of freedom may be involved, it is difficult to handle mathematically. Therefore, it is critical to select mathematical model having sufficiently few degrees of freedom to keep the equations of motion to manageable level, and yet having enough degrees of freedom to adequately describe the motion (Chung Ying Amy Chan, 2000).

The mathematical model of biped robot is first proposed by Chow and Jacobson (1972) as an inverted pendulum. They considered the upper body was modeled as a single link inverted pendulum with the prescribed base point moved only in the vertical direction. The use of this inverted pendulum is for the control of the postural stability of upper body for the biped locomotion. Hemami and his colleagues (1977,1980) developed this model into a massive inverted pendulum with the base joint fixed to the supporting ground in order to study the behavior of a body in standing position. To model the upper body of biped robot during gait, Wu et al. (1996, 1998) continued to develop the general single link inverted pendulum. Their mathematical model was developed with a base excited inverted pendulum could be used to predict major features of the upper body dynamics and to synthesize the mechanisms of walking. However, in the following study, it is pointed out that the single inverted pendulum models are too simple to accurately describe a complete locomotion (Wu et al. 1998). Multi-link planar models were then investigated to study biped locomotion, because they can solve the problems that are caused by the single inverted pendulum model.

In the study of Onyshko and Winter (1980), a model of a seven-link biped was used. They claimed that no constraints concerning the trajectories of any segment had been assumed. The results showed that a normal walking cycle can be achieved and the typical gait patterns could also be achieved with minor modification. Furusho and Masubuchi (1986, 1987) developed the biped structure modeled as a five-link, two-dimensional walking robot with a torso and knees but no ankles. The walking surface can be defined as a sequence of points connected with straight lines. This five-link model became very popular in the later application in the biped control. Hurmuzlu (1993a, 1993b) based on the five-link model, developed a systematic approach that can be followed to formulate objective functions to prescribe the gait during the single support phase. Tzafestas, S. et al. (1996) utilized this five-link biped model to study the forward walking motion in the sagittal plane. Regarding the interaction between the biped and the ground is modeled using external forces acting on each leg tip when the leg touches the ground. O. Haavisto (2004) developed seven degrees of freedom dynamic model to describe the dynamics of the system in all situations. Two coordinates fix the position of the center of mass of the torso, and the rest five coordinates describe the joint angles.

In this section, we will study the locomotion of biped robot in the sagittal plan.

5.2.1 The kinematics model of the five-link biped robot

From the aspect of mechanical-structural complexity and control system complexity, the biped robot locomotion system is an extremely complex dynamic system. To study this system and its motion requires certain simplifications. The 5-link biped structure developed by Furusho and Masubuchi (1986, 1987) is selected for illustrating the developed techniques. These authors have provided rather complete model data in their articles. Only the motion on the sagittal plane is considered in this study, thus the biped robot is considered as a planar model. The sagittal plane is defined by the vertical axis and the direction of locomotion. The planar biped model is considered to consist of five rigid links with five degrees of rotational freedom. The upper body of the planar biped model, which includes the head, arms and trunk, is considered as a massive rigid inverted pendulum. The swing motion of the arms and the motion between the thorax and pelvis are ignored. The upper body is connected to the two legs with two rotational joints. Each leg consists of two massive rigid links as a thigh and a shank. All links are connected with each other by rotational joints. The feet are considered to be massless, therefore, the dynamic structures of the feet are neglected (Chung Ying Amy Chan, 2000). The ground condition is assumed to be rigid and non-slip. At any time instant only one foot has a point contact with the ground.

The advantage of this five-link model is that it has sufficiently few degrees of freedom to

keep the equations of motion to a manageable level, while still having enough degrees of freedom to adequately describe the walking motion that includes the impact between the free end of the swing leg and the walking surface (Tzafestas, 1996). Figure 5.1 gives the configuration of the five-link model.

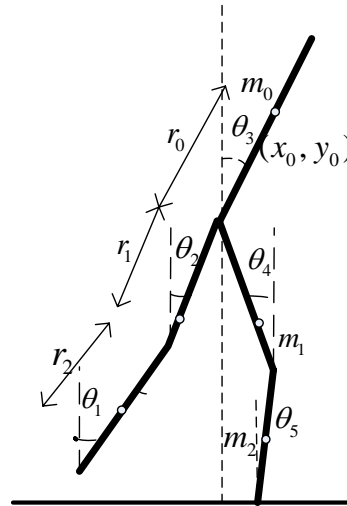


Fig.5.1 The five-link biped robot model

Refer to figure 5.1, θ_i ($i=1\cdots 5$) corresponds to the angle of link i with respect to the vertical direction. (x_0, y_0) is the mass center of the trunk.. The mass of the torso is m_0 . r_0 represents the distance from the mass center (x_0, y_0) to the hip joint. Suppose that the two legs of biped are same constructed, and it is accord to the general case. The distance from the mass center to the hip joint of two thighs is the same r_1 , and the distance from the mass center to the ankle is the same r_2 . Likely, both of the length of the two thigh is l_1 and the length of the shank is l_2 . Their masses are represented to be m_1 and m_2 respectively.

The bipedal locomotion during the single-support-leg phase can be studied as a tree-like topology. In its corresponding tree structure, each link becomes a node and each joint becomes an edge of the tree (Kyong-Sok Chang, 2000). Each branch of the tree topology is given a serial number: starting from the swing leg, the shank is called link A and the thigh is numbered as link B. The upper body is link C. And the link D and E stands for the thigh and shank of stance leg respectively. The rotational joints that connect each link are considered to be frictionless and are driven by independent motors.

According to the kinematics relationship between links shown in Figure 5.1, the position and velocity of the end for the swing leg can be defined. The position of the stance foot (x_e, y_e) and the free end of the swing leg (x_a, y_a) can be derived as equation (5.1) and equation (5.2) respectively:

$$\begin{aligned}x_a &= x_0 - r_0 \cdot \sin \theta_3 - l_1 \cdot \sin \theta_2 - l_2 \cdot \sin \theta_1 \\y_a &= y_0 - r_0 \cdot \cos \theta_3 - l_1 \cdot \cos \theta_2 - l_2 \cdot \cos \theta_1\end{aligned}\quad (5.1)$$

$$\begin{aligned}x_e &= x_0 - r_0 \cdot \sin \theta_3 + l_1 \cdot \sin \theta_4 + l_2 \cdot \sin \theta_5 \\y_e &= y_0 - r_0 \cdot \cos \theta_3 - l_1 \cdot \cos \theta_4 - l_2 \cdot \cos \theta_5\end{aligned}\quad (5.2)$$

Application of derivate of equation (5,2), we can get the velocity of end of swing foot \mathbf{V} :

$$\mathbf{v} = \begin{bmatrix} v_{ex} \\ v_{ey} \end{bmatrix} = \begin{bmatrix} -r_0 \cos \theta_3 \\ r_0 \sin \theta_3 \end{bmatrix} \cdot \dot{\theta}_3 + \begin{bmatrix} l_1 \cos \theta_4 \\ l_1 \sin \theta_4 \end{bmatrix} \cdot \dot{\theta}_4 + \begin{bmatrix} l_2 \cos \theta_5 \\ l_2 \sin \theta_5 \end{bmatrix} \cdot \dot{\theta}_5 \quad (5.3)$$

According to the mass center of trunk (x_0, y_0) , the mass center of each link i can be represented as follows:

Link A:

$$\begin{aligned}x_A &= x_0 - r_0 \cdot \sin \theta_3 - l_1 \cdot \sin \theta_2 - r_2 \cdot \sin \theta_1 \\y_A &= y_0 - r_0 \cdot \cos \theta_3 - l_1 \cdot \cos \theta_2 - r_2 \cdot \cos \theta_1\end{aligned}\quad (5.4)$$

Link B:

$$\begin{aligned}x_B &= x_0 - r_0 \cdot \sin \theta_3 - r_1 \cdot \sin \theta_2 \\y_B &= y_0 - r_0 \cdot \cos \theta_3 - r_1 \cdot \cos \theta_2\end{aligned}\quad (5.5)$$

Link C:

$$\begin{aligned}x_C &= x_0 \\y_C &= y_0\end{aligned}\quad (5.6)$$

Link D:

$$\begin{aligned}x_D &= x_0 - r_0 \cdot \sin \theta_3 + r_1 \cdot \sin \theta_4 \\y_D &= y_0 - r_0 \cdot \cos \theta_3 + r_1 \cdot \cos \theta_4\end{aligned}\quad (5.7)$$

Link E:

$$\begin{aligned}x_E &= x_0 - r_0 \cdot \sin \theta_3 + l_1 \cdot \sin \theta_4 + r_2 \cdot \sin \theta_5 \\y_E &= y_0 - r_0 \cdot \cos \theta_3 - l_1 \cdot \cos \theta_4 - r_2 \cdot \cos \theta_5\end{aligned}\quad (5.8)$$

Appling equation (5.4)~(5.8) to the following equation, we can get the coordinate of mass center of bipedal model:

$$\begin{aligned}cm_x &= \frac{m_0 x_C + m_1 x_B + m_1 x_D + m_2 x_A + m_2 x_E}{m_0 + 2m_1 + 2m_2} \\cm_y &= \frac{m_0 y_C + m_1 y_B + m_1 y_D + m_2 y_A + m_2 y_E}{m_0 + 2m_1 + 2m_2}\end{aligned}\quad (5.9)$$

Using the same method, derive the equation (2.4)~(2.8) by time, the velocity of mass center for each link can be calculated:

$$\mathbf{v}_A = \begin{bmatrix} v_{Ax} \\ v_{Ay} \end{bmatrix} = \begin{bmatrix} -r_0 \cdot \cos \theta_3 \\ r_0 \cdot \sin \theta_3 \end{bmatrix} \cdot \dot{\theta}_3 + \begin{bmatrix} -l_1 \cdot \cos \theta_2 \\ l_1 \cdot \sin \theta_2 \end{bmatrix} \cdot \dot{\theta}_2 + \begin{bmatrix} -r_2 \cdot \cos \theta_1 \\ r_2 \cdot \sin \theta_1 \end{bmatrix} \cdot \dot{\theta}_1 \quad (5.10)$$

$$\mathbf{v}_B = \begin{bmatrix} v_{Bx} \\ v_{By} \end{bmatrix} = \begin{bmatrix} -r_0 \cdot \cos \theta_3 \\ r_0 \cdot \sin \theta_3 \end{bmatrix} \cdot \dot{\theta}_3 + \begin{bmatrix} -r_1 \cdot \cos \theta_2 \\ r_1 \cdot \sin \theta_2 \end{bmatrix} \cdot \dot{\theta}_2 \quad (5.11)$$

$$\mathbf{v}_C = \begin{bmatrix} v_{Cx} \\ v_{Cy} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (5.12)$$

$$\mathbf{v}_D = \begin{bmatrix} v_{Dx} \\ v_{Dy} \end{bmatrix} = \begin{bmatrix} -r_0 \cdot \cos \theta_3 \\ r_0 \cdot \sin \theta_3 \end{bmatrix} \cdot \dot{\theta}_3 + \begin{bmatrix} r_1 \cdot \cos \theta_4 \\ r_1 \cdot \sin \theta_4 \end{bmatrix} \cdot \dot{\theta}_4 \quad (5.13)$$

$$\mathbf{v}_E = \begin{bmatrix} v_{Ex} \\ v_{Ey} \end{bmatrix} = \begin{bmatrix} -r_0 \cdot \cos \theta_3 \\ r_0 \cdot \sin \theta_3 \end{bmatrix} \cdot \dot{\theta}_3 + \begin{bmatrix} l_1 \cdot \cos \theta_4 \\ l_1 \cdot \sin \theta_4 \end{bmatrix} \cdot \dot{\theta}_4 + \begin{bmatrix} r_2 \cdot \cos \theta_5 \\ r_2 \cdot \sin \theta_5 \end{bmatrix} \cdot \dot{\theta}_5 \quad (5.14)$$

5.2.2 The dynamic model of the five-link biped robot

Two main formulations represent the dynamics of robotic mechanisms: the joint space dynamics formulation (M.W.Walker, 1982) and the operational space dynamics formulation (Oussarna Khatib,1987). Although these two different formulations ultimately describe the dynamics of the same robotic mechanism, each emphasizes different aspects of robot dynamics. The joint space dynamics formulation describes the dynamics of joints and the most common schemes for root dynamic formulation is a newer approach describing the dynamics of the tip of a manipulator with its task specification defined directly in its workspace. For the study of biped locomotion in our problem, we continue to use the joint space dynamics formulation.

The forward dynamic model of the bipedal locomotion system can be developed by utilizing the Lagrangian formulation or the Newton-Euler formulation. Lagrangian formulation has the advantage that only the kinetic and potential energies of the system are required to be computed and all the workless forces and constraint forces can be automatically eliminated. Thus, Lagrangian formulation has been employed to develop the equations of motion describing the dynamics of the bipedal locomotion.

5.2.2.1 Dynamic model of biped robot

The locomotion of biped model in the sigttal plan is a continuous forward motion, during this phase, the biped robot has the support leg in contact with the walking surface carrying all the weight of the body and the swing leg swinging in the air in the forward walking directions. It is supposed that the friction of the ground is sufficiently large, so that there is no slippage at the support end with the walking surface. To derive the dynamic equations of the single support phase, we utilize the mathematical model of the support leg attached to the ground. The contacting point between the support foot and ground (x_a, y_a) is constant, and is valid during this phase:

$$\dot{x}_a = \dot{y}_a = 0 \quad (5.15)$$

As the system can move freely in the $x-y$ plan and contains five links, it has five degrees of freedom. The corresponding five coordinates are selected according to figure 5.1:

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T \quad (5.16)$$

The model is actuated with five moments, refers to figure 5.2:

$$\mathbf{T} = [T_{\theta_1}, T_{\theta_2}, T_{\theta_3}, T_{\theta_4}, T_{\theta_5}]^T \quad (5.17)$$

During the phase of single-support-foot, the derivation of the equations of motion for the open kinematics chain used to describe the motion of the biped robot follows the standard procedure of Lagrangian formulation (Murray et al., 1994; Chung Ying Amy Chan, 2000).

The Lagrangian formulation of the five-link system is given by the difference between kinetic and potential energies:

$$L = K - P \quad (5.18)$$

In the above equation L is the Lagrange multiplier. K is the overall kinetic energy of the system which is the sum of all the five links, and P is the overall potential energy which includes each link:

$$K = \sum_{i=1}^5 K_i \quad (5.19)$$

$$P = \sum_{i=1}^5 P_i \quad (5.20)$$

The Lagrangian equation of motion is in the form as:

$$T_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{\theta}_i} \right) - \frac{\partial L}{\partial \theta_i} \quad (5.21)$$

Where, T is the sum of torques during the rotary motion. To substitute the lagrange multiplier with equation (5.19), (5.20) and (2.21), the Lagrangian equation of motion can be rearranged in the form:

$$T_i = \frac{\partial}{\partial t} \left(\frac{\partial (\sum_{i=1}^5 K_i - \sum_{i=1}^5 P_i)}{\partial \dot{\theta}_i} \right) - \frac{\partial (\sum_{i=1}^5 K_i - \sum_{i=1}^5 P_i)}{\partial \theta_i} \quad (5.22)$$

The kinetic energy includes two parts: the rectilinear motion and rotary motion around the center mass. The kinetic energy and potential energy for each link can be calculated as the following equations:

$$K_i = \frac{1}{2} m_i (v_{ix}^2 + v_{iy}^2) + \frac{1}{2} I_i \dot{\theta}_i^2 \quad i = 1, 2 \dots 5 \quad (5.23)$$

$$P_i = m_i g \cdot y_i \quad (5.24)$$

In which, g is the gravitational acceleration. Thus, the kinetic energy and potential energy for each link are formulated as follows:

Link A:

$$K_A = \frac{1}{2}(m_2 r_2^2 + I_A) \cdot \dot{\theta}_1^2 + \frac{1}{2} m_2 l_1^2 \cdot \dot{\theta}_2^2 + \frac{1}{2} m_2 r_0^2 \cdot \dot{\theta}_3^2 + m_2 [r_2 l_1 \cos(\theta_1 - \theta_2) \cdot \dot{\theta}_1 \dot{\theta}_2 + r_0 l_1 \cos(\theta_2 - \theta_3) \cdot \dot{\theta}_2 \dot{\theta}_3 + r_0 r_2 \cos(\theta_1 - \theta_3) \cdot \dot{\theta}_1 \dot{\theta}_3] \quad (5.25)$$

$$P_A = m_2 g \cdot (y_0 - r_0 \cdot \cos \theta_3 - l_1 \cdot \cos \theta_2 - r_2 \cdot \cos \theta_1) \quad (5.26)$$

Link B:

$$K_B = \frac{1}{2}(m_1 r_1^2 + I_B) \cdot \dot{\theta}_2^2 + \frac{1}{2} m_1 r_0^2 \cdot \dot{\theta}_3^2 + m_1 [r_0 r_1 \cdot \cos(\theta_2 - \theta_3) \cdot \dot{\theta}_2 \dot{\theta}_3] \quad (5.27)$$

$$P_B = m_1 g \cdot (y_0 - r_0 \cdot \cos \theta_3 - r_1 \cdot \cos \theta_2) \quad (5.28)$$

Link C:

$$K_C = \frac{1}{2} I_C \dot{\theta}_3^2 \quad (5.29)$$

$$P_C = m_0 g \cdot y_0 \quad (5.30)$$

Link D:

$$K_D = \frac{1}{2}(m_1 r_1^2 + I_D) \cdot \dot{\theta}_4^2 + \frac{1}{2} m_1 r_0^2 \cdot \dot{\theta}_5^2 - m_1 [r_0 r_1 \cdot \cos(\theta_3 - \theta_4) \cdot \dot{\theta}_3 \dot{\theta}_4] \quad (5.31)$$

$$P_D = m_1 g \cdot (y_0 - r_0 \cdot \cos \theta_3 - r_1 \cdot \cos \theta_4) \quad (5.32)$$

Link E:

$$K_E = \frac{1}{2}(m_2 r_2^2 + I_E) \cdot \dot{\theta}_5^2 + \frac{1}{2} m_2 r_0^2 \cdot \dot{\theta}_3^2 + \frac{1}{2} m_2 l_1^2 \cdot \dot{\theta}_4^2 + m_2 [r_2 l_1 \cos(\theta_4 - \theta_5) \cdot \dot{\theta}_4 \dot{\theta}_5 - r_0 l_1 \cos(\theta_3 + \theta_4) \cdot \dot{\theta}_3 \dot{\theta}_4 - r_0 r_2 \cos(\theta_3 + \theta_5) \cdot \dot{\theta}_3 \dot{\theta}_5] \quad (5.33)$$

$$P_E = m_2 g \cdot (y_0 - r_0 \cdot \cos \theta_3 - l_1 \cdot \cos \theta_4 - r_2 \cdot \cos \theta_5) \quad (5.34)$$

Substituting equation (5.25)~(5.34) to equation (5.22), we derive the dynamics equation of biped robot in the sigttal plan in the following standard form:

$$\mathbf{A}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{B}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}^2 + \mathbf{C}(\boldsymbol{\theta}) = \mathbf{T} \quad (5.35)$$

As the considered biped model is a five freedom system, there exit also five partial differential equations. \mathbf{T} is the actuated moments given by equation (5.17). $\mathbf{A}(\boldsymbol{\theta}) \in \mathfrak{R}^{5 \times 5}$ is the inertial matrix and each term is formulated as follows:

$$\begin{aligned} a_{11} &= m_2 r_2^2 + I_A \\ a_{12} &= m_2 r_2 l_1 \cos(\theta_1 - \theta_2) \\ a_{13} &= m_2 r_0 r_2 \cos(\theta_1 - \theta_3) \\ a_{14} &= 0 \\ a_{15} &= 0 \end{aligned} \quad (5.36)$$

$$\begin{aligned}
a_{21} &= m_2 r_2 l_1 \cos(\theta_1 - \theta_2) \\
a_{22} &= m_2 l_1^2 + m_1 r_1^2 + I_B \\
a_{23} &= (m_2 r_0 l_1 + m_1 r_0 r_1) \cos(\theta_2 - \theta_3) \\
a_{24} &= 0 \\
a_{25} &= 0
\end{aligned} \tag{5.37}$$

$$\begin{aligned}
a_{31} &= m_2 r_0 r_2 \cos(\theta_1 - \theta_3) \\
a_{32} &= (m_2 r_0 l_1 + m_1 r_0 r_1) \cos(\theta_2 - \theta_3) \\
a_{33} &= 2m_2 r_0^2 + 2m_1 r_0^2 + I_C \\
a_{34} &= -m_1 r_0 r \cos(\theta_3 - \theta_4) - m_2 r_0 l_1 \cos(\theta_3 + \theta_4) \\
a_{35} &= -m_2 r_0 r_2 \cos(\theta_3 + \theta_5)
\end{aligned} \tag{5.38}$$

$$\begin{aligned}
a_{41} &= 0 \\
a_{42} &= 0 \\
a_{43} &= -m_1 r_0 r \cos(\theta_3 - \theta_4) - m_2 r_0 l_1 \cos(\theta_3 + \theta_4) \\
a_{44} &= m_1 r_1^2 + m_2 l_1^2 + I_D \\
a_{45} &= m_2 r_2 l_1 \cos(\theta_4 - \theta_5)
\end{aligned} \tag{5.39}$$

$$\begin{aligned}
a_{51} &= 0 \\
a_{52} &= 0 \\
a_{53} &= -m_2 r_0 r_2 \cos(\theta_3 + \theta_5) \\
a_{54} &= m_2 r_2 l_1 \cos(\theta_4 - \theta_5) \\
a_{55} &= m_2 r_2^2 + I_E
\end{aligned} \tag{5.40}$$

It can be seen from the above expressions, $\mathbf{A}(\boldsymbol{\theta})$ is a symmetric matrix. $\mathbf{B}(\boldsymbol{\theta}) \in \mathfrak{R}^{5 \times 5}$, which is the quadratic term of $\dot{\boldsymbol{\theta}}$, is a matrix. The expression of each term is:

$$\begin{aligned}
b_{11} &= 0 \\
b_{12} &= m_2 r_2 l_1 \sin(\theta_1 - \theta_2) \\
b_{13} &= m_2 r_0 r_2 \sin(\theta_1 - \theta_3) \\
b_{14} &= 0 \\
b_{15} &= 0
\end{aligned} \tag{5.41}$$

$$\begin{aligned}
b_{21} &= -m_2 r_2 l_1 \sin(\theta_1 - \theta_2) \\
b_{22} &= 0 \\
b_{23} &= (m_2 r_0 l_1 + m_1 r_0 r_1) \sin(\theta_2 - \theta_3) \\
b_{24} &= 0 \\
b_{25} &= 0
\end{aligned} \tag{5.42}$$

$$\begin{aligned}
b_{31} &= -m_2 r_0 r_2 \sin(\theta_1 - \theta_3) \\
b_{32} &= -(m_2 r_0 l_1 + m_1 r_0 r_1) \sin(\theta_2 - \theta_3) \\
b_{33} &= 0 \\
b_{34} &= m_2 r_0 l_1 \sin(\theta_3 + \theta_4) - m_1 r_0 r_1 \sin(\theta_3 - \theta_4) \\
b_{35} &= m_2 r_0 r_2 \sin(\theta_3 + \theta_5)
\end{aligned} \tag{5.43}$$

$$\begin{aligned}
b_{41} &= 0 \\
b_{42} &= 0 \\
b_{43} &= m_2 r_0 l_1 \sin(\theta_3 + \theta_4) + m_1 r_0 r_1 \sin(\theta_3 - \theta_4) \\
b_{44} &= 0 \\
b_{45} &= m_2 r_2 l_1 \sin(\theta_4 - \theta_5)
\end{aligned} \tag{5.44}$$

$$\begin{aligned}
b_{51} &= 0 \\
b_{52} &= 0 \\
b_{53} &= m_2 r_0 r_2 \sin(\theta_3 + \theta_5) \\
b_{54} &= -m_2 r_2 l_1 \sin(\theta_4 - \theta_5) \\
b_{55} &= 0
\end{aligned} \tag{5.45}$$

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

$$\begin{aligned}
c_1 &= m_2 g r_2 \sin \theta_1 \\
c_2 &= (m_2 g l_1 + m_1 g r_1) \sin \theta_2 \\
c_3 &= (2m_2 g r_0 + 2m_1 g r_0) \sin \theta_3 \\
c_4 &= (m_1 g r_1 + m_2 g l_1) \sin \theta_4 \\
c_5 &= m_2 g r_2 \sin \theta_5
\end{aligned} \tag{5.46}$$

5.2.2.2 Transformation of the dynamic model

The above derivative process utilizes the angles θ_i with respect to the vertical direction. However, for the general control purpose, we usually use the relative angle $[\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T$ between the two connected links, whose definition refers to figure 5.2.

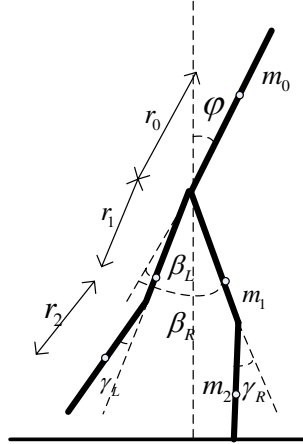


Fig.5.2 Biped model with relative angle

The relative angles $[\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T$ between links are used instead of the absolute angle θ_i of each link. The relationship between the relative angles and the absolute angles is as follows:

$$\begin{cases} \gamma_L = \theta_1 - \theta_2 \\ \beta_L = \varphi - \theta_2 \\ \varphi = \theta_3 \\ \beta_R = \varphi + \theta_4 \\ \gamma_R = \theta_4 - \theta_5 \end{cases} \quad (5.47)$$

Thus, the variables θ_i ($i = 1, 2, \dots, 5$) can be expressed in terms of q_i ($i = 1, 2, \dots, 5$) as:

$$\begin{cases} \theta_1 = \gamma_L + \varphi - \beta_L \\ \theta_2 = \varphi - \beta_L \\ \theta_3 = \varphi \\ \theta_4 = \beta_R - \varphi \\ \theta_5 = \beta_R - \varphi - \gamma_R \end{cases} \quad (5.48)$$

To derive the relationship between moments \mathbf{T} and the actual driving torques of the joints $\mathbf{M} = [M_{L1}, M_{R1}, M_\varphi, M_{L2}, M_{R2}]^T$, referring to figure 5.3, we define the relative angles $[\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T$ as:

$$\mathbf{q} = [q_1, q_2, q_3, q_4, q_5]^T = [\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T \quad (5.49)$$

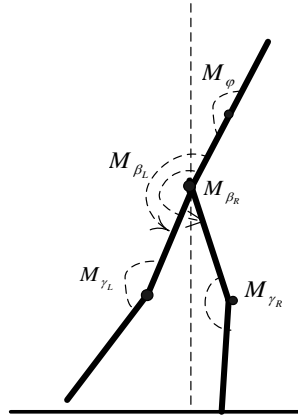


Fig.5.3 Biped model coordinates with applied torques.

Therefore, from the relation (Tzafestas, 1996) :

$$M_{q_j} = \sum_{i=1}^5 T_{\theta_i} \frac{\partial \theta_i}{\partial q_j} \quad (j = 1, 2, \dots, 5) \quad (5.50)$$

we can get the relationship between the general torque M_{q_j} and T_{θ_i}

$$\begin{cases} M_{q_1} = T_{\theta_1} \\ M_{q_2} = -T_{\theta_1} - T_{\theta_2} \\ M_{q_3} = T_{\theta_1} + T_{\theta_2} + T_{\theta_3} - T_{\theta_4} - T_{\theta_5} \\ M_{q_4} = T_{\theta_4} + T_{\theta_5} \\ M_{q_5} = -T_{\theta_5} \end{cases} \quad (5.51)$$

Using the above equations, the biped model is transformed as follows:

$$A_{11}\ddot{\theta}_1 + A_{12}\ddot{\theta}_2 + A_{13}\ddot{\theta}_3 + A_{14}\ddot{\theta}_4 + A_{15}\ddot{\theta}_5 + h_{q_1} + G_{q_1} = M_{q_1} \quad (5.52)$$

in which

$$\begin{aligned} A_{1j} &= a_{1j} & j &= 1, 2, \dots, 5 \\ h_{q_1} &= b_1 \\ G_{q_1} &= c_1 \end{aligned} \quad (5.53)$$

$$A_{21}\ddot{\theta}_1 + A_{22}\ddot{\theta}_2 + A_{23}\ddot{\theta}_3 + A_{24}\ddot{\theta}_4 + A_{25}\ddot{\theta}_5 + h_{q_2} + G_{q_2} = M_{q_2} \quad (5.54)$$

in which

$$\begin{aligned} A_{2j} &= -a_{1j} - a_{2j} & j &= 1, 2, \dots, 5 \\ h_{q_2} &= -b_1 - b_2 \\ G_{q_2} &= -c_1 - c_2 \end{aligned} \quad (5.55)$$

$$A_{31}\ddot{\theta}_1 + A_{32}\ddot{\theta}_2 + A_{33}\ddot{\theta}_3 + A_{34}\ddot{\theta}_4 + A_{35}\ddot{\theta}_5 + h_{q_3} + G_{q_3} = M_{q_3} \quad (5.56)$$

in which

$$\begin{aligned} A_{3j} &= a_{1j} + a_{2j} + a_{3j} - a_{4j} - a_{5j} & j &= 1, 2, \dots, 5 \\ h_{q_3} &= b_1 + b_2 + b_3 - b_4 - b_5 \\ G_{q_3} &= c_1 + c_2 + c_3 - c_4 - c_5 \end{aligned} \quad (5.57)$$

$$A_{41}\ddot{\theta}_1 + A_{42}\ddot{\theta}_2 + A_{43}\ddot{\theta}_3 + A_{44}\ddot{\theta}_4 + A_{45}\ddot{\theta}_5 + h_{q_4} + G_{q_4} = M_{q_4} \quad (5.58)$$

in which

$$\begin{aligned} A_{4j} &= a_{4j} + a_{5j} & j &= 1, 2, \dots, 5 \\ h_{q_4} &= b_4 + b_5 \\ G_{q_4} &= c_4 + c_5 \end{aligned} \quad (5.59)$$

$$A_{51}\ddot{\theta}_1 + A_{52}\ddot{\theta}_2 + A_{53}\ddot{\theta}_3 + A_{54}\ddot{\theta}_4 + A_{55}\ddot{\theta}_5 + h_{q_5} + G_{q_5} = M_{q_5} \quad (5.60)$$

in which

$$\begin{aligned} A_{5j} &= -a_{5j} & j &= 1, 2, \dots, 5 \\ h_{q_5} &= -b_5 \\ G_{q_5} &= -c_5 \end{aligned} \quad (5.61)$$

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

$$\mathbf{D}_q(q)\ddot{q} + \mathbf{h}_q(q, \dot{q})\dot{q}^2 + \mathbf{G}_q(q) = \mathbf{M}_q \quad (5.62)$$

Where

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

$$\begin{aligned}
h_q \dot{q} &= [h_{q_1}, h_{q_2}, h_{q_3}, h_{q_4}, h_{q_5}]^T \\
G_q &= [G_{q_1}, G_{q_2}, G_{q_3}, G_{q_4}, G_{q_5}]^T
\end{aligned} \tag{5.63}$$

$\mathbf{D}_q(q)$ is the 5×5 positive definite inertia matrix, $\mathbf{h}_q(q, \dot{q})$ is the 5×1 vector of centripetal and Coriolis torques, $\mathbf{G}_q(q)$ is the 5×1 vector representing gravitational torques, and \mathbf{M}_q is the 5×1 vector of control torque applied at each joint. Inserting equation (5.53), (5.55), (5.57), (5.59) and (5.61) into equation (5.63), one can get each formulated term as follows:

For $\mathbf{D}_q(q) = D_q(i, j) \quad (i = 1, 2, \dots, 5 \quad j = 1, 2, \dots, 5)$

$$\begin{cases}
D_q(1,1) = m_2 r_2^2 + I_A \\
D_q(1,2) = -(m_2 r_2^2 + I_A) - m_2 r_2 l_1 \cos(\theta_1 - \theta_2) \\
D_q(1,3) = m_2 r_2^2 + I_A + m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + m_2 r_0 r_2 \cos(\theta_1 - \theta_3) \\
D_q(1,4) = 0 \\
D_q(1,5) = 0
\end{cases} \tag{5.64}$$

$$\begin{cases}
D_q(2,1) = D_q(1,2) \\
D_q(2,2) = 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + m_2 r_2^2 + I_A + m_2 l_1^2 + m_1 r_1^2 + I_B \\
D_q(2,3) = -(m_2 r_2^2 + I_A) - 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) - (m_2 l_1^2 + m_1 r_1^2 + I_B) \\
\quad - m_2 r_0 r_2 \cos(\theta_1 - \theta_3) - (m_2 r_0 l_1 + m_1 r_0 r_1) \cos(\theta_2 - \theta_3) \\
D_q(2,4) = 0 \\
D_q(2,5) = 0
\end{cases} \tag{5.65}$$

$$\begin{cases}
D_q(3,1) = D_q(1,3) \\
D_q(3,2) = D_q(2,3) \\
D_q(3,3) = m_2 r_2^2 + I_A + 2m_2 r_2 l_1 \cos(\theta_1 - \theta_2) + 2m_2 r_0 r_2 \cos(\theta_1 - \theta_3) + (m_2 l_1^2 + m_1 r_1^2 + I_B) \\
\quad + 2(m_2 r_0 l_1 + m_1 r_0 r_1) \cos(\theta_2 - \theta_3) + (2m_2 r_0^2 + 2m_1 r_0^2 + I_C) + 2m_2 r_0 r_2 \cos(\theta_3 + \theta_5) \\
\quad + 2[m_1 r_0 r \cos(\theta_3 - \theta_4) + m_2 r_0 l_1 \cos(\theta_3 + \theta_4)] + m_1 r_1^2 + m_2 l_1^2 + I_D \\
\quad + 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E \\
D_q(3,4) = -m_1 r_0 r \cos(\theta_3 - \theta_4) - m_2 r_0 l_1 \cos(\theta_3 + \theta_4) - m_2 r_0 r_2 \cos(\theta_3 + \theta_5) - (m_1 r_1^2 + m_2 l_1^2 + I_D) \\
\quad - 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) - (m_2 r_2^2 + I_E) \\
D_q(3,5) = m_2 r_0 r_2 \cos(\theta_3 + \theta_5) + m_2 r_2 l_1 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E
\end{cases} \tag{5.66}$$

$$\begin{cases}
D_q(4,1) = D_q(1,4) \\
D_q(4,2) = D_q(2,4) \\
D_q(4,3) = D_q(3,4) \\
D_q(4,4) = m_1 r_1^2 + m_2 l_1^2 + I_D + 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E \\
D_q(4,5) = -m_2 r_2 l_1 \cos(\theta_4 - \theta_5) - (m_2 r_2^2 + I_E)
\end{cases} \tag{5.67}$$

$$\begin{cases} D_q(5,1) = D_q(1,5) \\ D_q(5,2) = D_q(2,5) \\ D_q(5,3) = D_q(3,5) \\ D_q(5,4) = D_q(4,5) \\ D_q(5,5) = -m_2 r_2^2 - I_E \end{cases} \quad (5.68)$$

Then, put the relationship between absolute angle and relative angle (equation (5.48)) into expressions (5.64)~(5.68), we can get the inertia matrix $\mathbf{D}_q(q)$ concerning the relative angles

$[\gamma_L, \beta_L, \varphi, \beta_R, \gamma_R]^T$ as:

$$\begin{cases} D_q(1,1) = m_2 r_2^2 + I_A \\ D_q(1,2) = -(m_2 r_2^2 + I_A) - m_2 r_2 l_1 \cos(\gamma_L) \\ D_q(1,3) = m_2 r_2^2 + I_A + m_2 r_2 l_1 \cos(\gamma_L) + m_2 r_0 r_2 \cos(\gamma_L - \beta_L) \\ D_q(1,4) = 0 \\ D_q(1,5) = 0 \end{cases} \quad (5.69)$$

$$\begin{cases} D_q(2,1) = D_q(1,2) \\ D_q(2,2) = -(m_2 r_2^2 + I_A) + m_2 l_1^2 + m_1 r_1^2 + I_B \\ D_q(2,3) = -(m_2 r_2^2 + I_A) - 2m_2 r_2 l_1 \cos(\gamma_L) - (m_2 l_1^2 + m_1 r_1^2 + I_B) \\ \quad - m_2 r_0 r_2 \cos(\gamma_L - \beta_L) - (m_2 r_0 l_1 + m_1 r_0 r_1) \cos(-\beta_L) \\ D_q(2,4) = 0 \\ D_q(2,5) = 0 \end{cases} \quad (5.70)$$

$$\begin{cases} D_q(3,1) = D_q(1,3) \\ D_q(3,2) = D_q(2,3) \\ D_q(3,3) = m_2 r_2^2 + I_A + 2m_2 r_2 l_1 \cos(\gamma_L) + 2m_2 r_0 r_2 \cos(\gamma_L - \beta_L) + (m_2 l_1^2 + m_1 r_1^2 + I_B) \\ \quad + 2(m_2 r_0 l_1 + m_1 r_0 r_1) \cos(-\beta_L) + (2m_2 r_0^2 + 2m_1 r_0^2 + I_C) + 2m_2 r_0 r_2 \cos(\beta_R - \gamma_R) \\ \quad + 2[m_1 r_0 r \cos(2\varphi - \beta_R) + m_2 r_0 l_1 \cos(\beta_R)] + m_1 r_1^2 + m_2 l_1^2 + I_D \\ \quad + 2m_2 r_2 l_1 \cos(\theta_4 - \theta_5) + m_2 r_2^2 + I_E \\ D_q(3,4) = -m_1 r_0 r \cos(2\varphi - \beta_R) - m_2 r_0 l_1 \cos(\beta_R) - m_2 r_0 r_2 \cos(\beta_R - \gamma_R) - (m_1 r_1^2 + m_2 l_1^2 + I_D) \\ \quad - 2m_2 r_2 l_1 \cos(\gamma_R) - (m_2 r_2^2 + I_E) \\ D_q(3,5) = m_2 r_0 r_2 \cos(\beta_R - \gamma_R) + m_2 r_2 l_1 \cos(\gamma_R) + m_2 r_2^2 + I_E \end{cases} \quad (5.71)$$

$$\begin{cases} D_q(4,1) = D_q(1,4) \\ D_q(4,2) = D_q(2,4) \\ D_q(4,3) = D_q(3,4) \\ D_q(4,4) = m_1 r_1^2 + m_2 l_1^2 + I_D + 2m_2 r_2 l_1 \cos(\gamma_R) + m_2 r_2^2 + I_E \\ D_q(4,5) = -m_2 r_2 l_1 \cos(\gamma_R) - (m_2 r_2^2 + I_E) \end{cases} \quad (5.72)$$

$$\begin{cases} D_q(5,1) = D_q(1,5) \\ D_q(5,2) = D_q(2,5) \\ D_q(5,3) = D_q(3,5) \\ D_q(5,4) = D_q(4,5) \\ D_q(5,5) = -m_2 r_2^2 - I_E \end{cases} \quad (5.73)$$

Similarly, we can derive the centripetal and Coriolis torques $\mathbf{h}_q(q, \dot{q})$ and the gravitational torques $\mathbf{G}_q(q)$ in the following from: ($i = 1, 2, \dots, 5$ $j = 1, 2, \dots, 5$)

$$\begin{cases} h_q(1,1) = 0 \\ h_q(1,2) = -m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(1,3) = m_2 r_2 l_1 \sin(\gamma_L) + m_2 r_0 r_2 \sin(\gamma_L - \beta_L) \\ h_q(1,4) = 0 \\ h_q(1,5) = 0 \end{cases} \quad (5.74)$$

$$\begin{cases} h_q(2,1) = m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(2,2) = 0 \\ h_q(2,3) = -m_2 r_0 r_2 \sin(\gamma_L - \beta_L) - (m_2 r_0 l_1 + m_1 r_0 r_1) \sin(-\beta_L) \\ h_q(2,4) = 0 \\ h_q(2,5) = 0 \end{cases} \quad (5.75)$$

$$\begin{cases} h_q(3,1) = m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(3,2) = 2m_2 r_2 l_1 \sin(\gamma_L) \\ h_q(3,3) = -2m_2 r_0 l_1 \sin(\beta_R) - 2m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\ h_q(3,4) = 0 \\ h_q(3,5) = 0 \end{cases} \quad (5.76)$$

$$\begin{cases} h_q(4,1) = 0 \\ h_q(4,2) = 0 \\ h_q(4,3) = m_2 r_0 l_1 \sin(\beta_R) + m_1 r_0 r_1 \sin(2\varphi - \beta_R) + m_2 r_0 r_2 \sin(\beta_R - \gamma_R) \\ h_q(4,4) = 0 \\ h_q(4,5) = -m_2 r_2 l_1 \sin(\gamma_R) \end{cases} \quad (5.77)$$

$$\begin{cases} h_q(5,1) = 0 \\ h_q(5,2) = 0 \\ h_q(5,3) = -m_2 r_0 r_2 \sin(\beta_R - \gamma_R) - m_2 r_2 l_1 \sin(\gamma_R) \\ h_q(5,4) = m_2 r_2 l_1 \sin(\gamma_R) \\ h_q(5,5) = 0 \end{cases} \quad (5.78)$$

$$\begin{cases} G_q(1) = m_2 g r_2 \sin(\gamma_L + \varphi - \beta_L) \\ G_q(2) = -m_2 g r_2 \sin(\gamma_L + \varphi - \beta_L) - (m_2 g l_1 + m_1 g r_1) \sin(\varphi - \beta_L) \\ G_q(3) = m_2 g r_2 \sin(\gamma_L + \varphi - \beta_L) + (m_2 g l_1 + m_1 g r_1) \sin(\varphi - \beta_L) + (2m_2 g r_0 + 2m_1 g r_0) \sin \varphi \\ \quad - (m_1 g r_1 + m_2 g l_1) \sin(\beta_R - \varphi) - m_2 g r_2 \sin(\beta_R - \varphi - \gamma_R) \\ G_q(4) = (m_1 g r_1 + m_2 g l_1) \sin(\beta_R - \varphi) + m_2 g r_2 \sin(\beta_R - \varphi - \gamma_R) \\ G_q(5) = m_2 g r_2 \sin(\beta_R - \varphi - \gamma_R) \end{cases} \quad (5.79)$$

5.2.3 Summary

In this section, we presented the methodology for the derivation of equations of motion to describe the locomotion of a biped robot walking on a flat horizontal surface. A five-link

kinematic model of the biped robot has been developed which has sufficiently few degrees of freedom to keep the equations of motion to a manageable level, while having enough degrees of freedom to approximately describe the locomotion. Based on Lagrangian equation, the dynamic model is derived during the phase of single-support-foot. In the developed equations, the input joint angles are absolute angles. Therefore, we transformed the dynamics equation considering the relative angles between each link for control purpose. Comparing with the derivation process and the expressions of dynamic equations for biped robot, which is developed by Chung Ying Amy Chan (2000) or by Olli Haavisto (2004), our model is relatively less complexity. It is because on the one hand, we suppose that the structures of the both legs of biped robot are same, and on the other hand, we choose the mass center of the trunk of biped robot as the starting point to derive the position and velocity of the mass center of each link, so the total kinetic energy and potential energy have much simple expressions.

5.3 Control strategy

Our proposed control strategy for biped robot stepping over dynamic obstacle is based on the assumption that the high-level controller and low-level controller can be designed separately. The objective of high-level controller is to plan the footstep and joint angle for a few steps and the low-level controller allows the biped to control the tracking of desired joint angles.

5.3.1 High-level controller design

Because the goal of the high-level control is to give the path and joint angle planning for biped robot in dynamic environment, the strategic control needs to use a predictive approach based on an on-line optimization of learning process. Our approach is based on FQL concept, which is developed in Chapter 4. The structure of the high-level control can be divided into the following six parts:

- The first part is to compute the position of the foot, the position of the obstacle and obstacle velocity by discrete information for simulating the dynamic environment. In this model we suppose the walking of biped robot is a succession of single support phases, and double support phase is instantaneous.
- The second part involves a fuzzification of inputs of state..
- The third part concerns the Fuzzy Q-learning algorithm which must choose one output for each activated rules, We use two Fuzzy Q-learning algorithm, one for learning step length and step duration time, and other for maximum step height.

- The fourth part gives the reinforcement signal. The reinforcement signal provides the information in terms of reward or punishment. Consequently, the reinforcement signal informs the learning agent about the quality of chosen actions.

- The fifth part interpolates the designated point such as starting point, landing point and maximum step height to get the swinging trajectory.

- The sixth part generates the joint profiles according to the foot trajectory by inverse kinematics.

The scheme of high-level control is presented in figure 5.4.

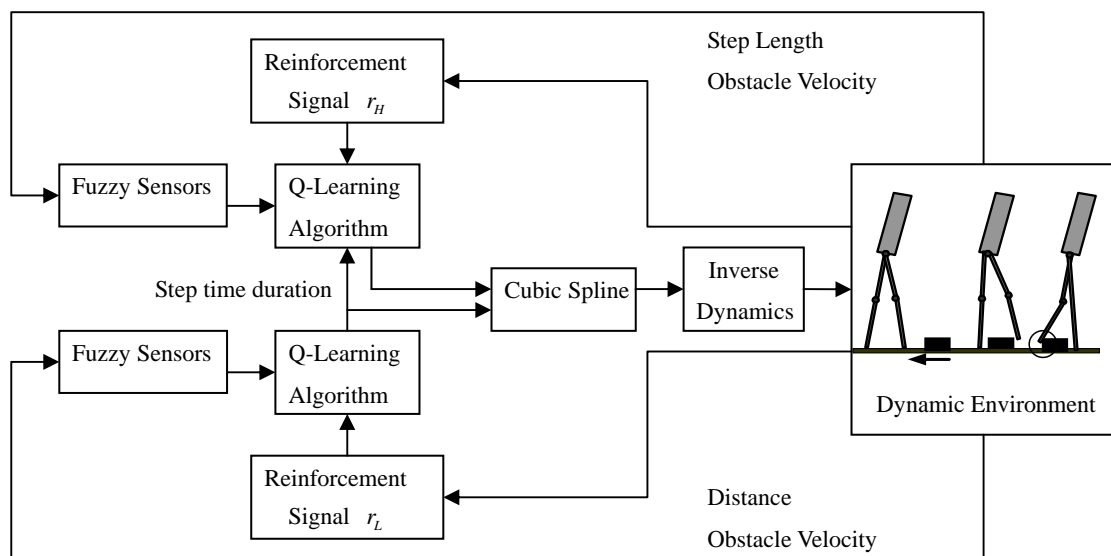


Fig.5.4 Scheme of high-level control for biped robot in dynamic environment

5.3.1.1 Generation of trajectory for swing leg

In the previous section, Fuzzy Q-Learning is developed to train the key parameters: step length, duration time and maximum height of every step which determine the trajectory of swing leg of humanoid robot. In fact, the starting point x_{\min} and landing point x_{\max} of foot which are decided by length of step (refers to equation (5.80)), and the maximum step height h_{\max} which decides the peak of the trajectory will shape the trajectory of swing leg. The time duration of step only determines the frequency of movement. The swing trajectory is shown in figure 5.5.

$$L_{step} = x_{\max} - x_{\min} \quad (5.80)$$

So with starting point, landing point and vertex, the swing trajectory can be obtained by interpolation algorithm. In this section, cubic spline interpolation is chosen in our approach because of its comparatively less amount of calculation property and better precision.

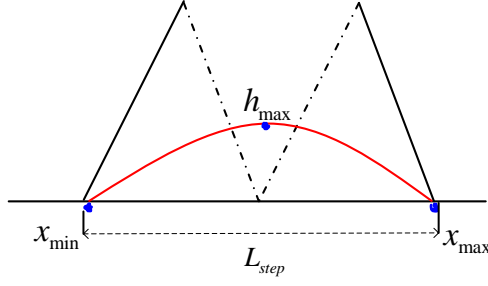


Fig. 5.5 Swing trajectory during on step

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:

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \quad (5.81)$$

To ensure the unique result, restriction is added into the cubic spline. By limiting the derivative coefficient of every cubic polynomial at the breaking point, the coefficients of the polynomial can be figured out by the following equations.

$$\begin{cases} a_i = S_i(x_i) = y_i \\ b_i = S'_i(x_i) = -\frac{h_i}{6} z_i + \frac{y_{i+1} - y_i}{h_i} \\ c_i = \frac{1}{2} S''_i(x_i) = \frac{z_i}{2} \\ d_i = \frac{1}{6} S'''_i(x_i) = \frac{z_{i+1} - z_i}{6h_i} \end{cases} \quad (5.82)$$

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

$$\begin{cases} z_i = 0 & \text{if } i = 0, n \\ z_i = (v_i - h_i z_{i+1}) / u_i \end{cases} \quad (5.83)$$

Through Gaussian elimination, one can get v_i and u_i expressions:

$$\text{if } i = 1 \quad \begin{cases} u_1 = 2(h_0 - h_1) \\ v_1 = 6(b_1 - b_0) \end{cases} \quad (5.84)$$

$$\text{else} \quad \begin{cases} u_i = 2(h_{i-1} + h_i) - h_{i-1}^2 / u_{i-1} \\ v_i = 6(b_i - b_{i-1}) - h_{i-1} v_{i-1} / u_{i-1} \end{cases} \quad (5.85)$$

Substitution equation (5.83), (5.84) and (5.85) into equation (5.82), the coefficients of every cubic polynomial can be calculated. Then, equation (5.82) is used to interpolate the characteristic points: starting point, landing point and vertex, a smooth swing trajectory can be achieved.

5.3.1.2 Generation of joint angle

Inverse kinematics

There are mainly two approaches by which develop the inverse kinematical equation of robot: geometry method and algebraic method. The geometry method is generally applied on the robot with relatively simpler structure, such as the robot arm moving in two-dimension plan or with parallel joints of fewer freedoms. While, for the robots which move in the three-dimension plan or with more joints, the algebraic method is more appropriate. However, when there exists coupling, multiple solutions and singularity in the kinematics equations, it is hard to develop the inverse kinematics. Therefore, the approach by which derive the inverse kinematical equations depends on the structure of robot. In the problem of biped robot stepping over obstacle, we using a five-link model to represent the structure of robot as explain in Chapter 1, and its motion is restricted in the sigttal plane. Thus, the geometry method is utilized to develop the inverse kinematical equations of biped robot.

One walking period of biped robot includes single-support-leg phase and stance phase. However, the stance phase is considered to be instantaneous. During the single-support-leg phase , because the calf of support leg function as preventing the swing leg from touching the ground before stance, and has little effect on the walking motion of biped robot. Consequently, we did not introduce the knee joint of support leg in the inverse kinematical equation of biped robot.

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

To describe the problem more clearly, we redefine the angle of thigh of support leg with respect to the vertical direction as θ_1 . θ_2 corresponds to the angle of thigh of swing leg concerning the vertical direction, and θ_3 represents the angle of shank of swing leg relatives to the vertical direction. The length of thigh and shank for both legs is described as l_1 and l_2 respectively.

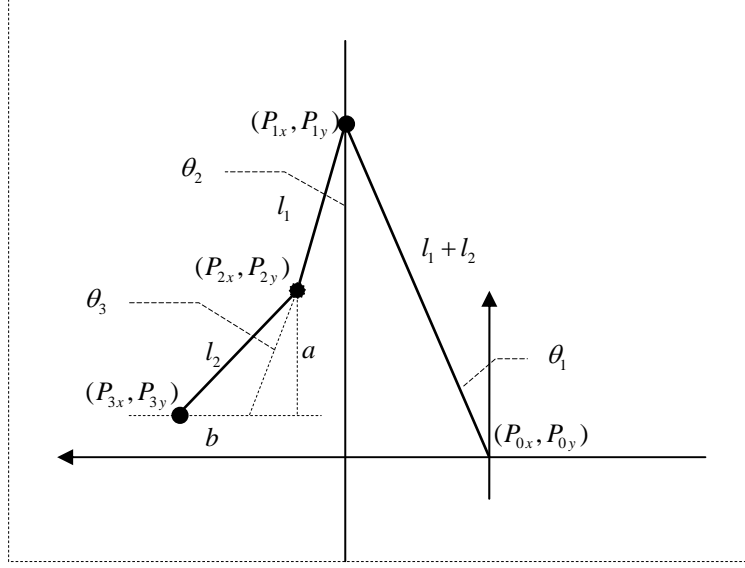


Fig. 5.6 Geometrical relationship between stance leg and swing leg

The geometrical relationship between stance leg and swing leg of humanoid robot is described in figure 5.6. Generally, in the stepping phase of robot, the stance leg does not bend. After footstep planning approach which is developed in Chapter 4, the angle between stance foot and vertical line θ_1 and its position (P_{0x}, P_{0y}) are recorded as the initial condition. The coordinate of any point (P_{3x}, P_{3y}) on the swing trajectory curve can also be calculated by cubic spline interpolating as described in section 5.3.1.1.

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

$$\begin{cases} P_{1x} = (l_1 + l_2) \sin \theta_1 \\ P_{1y} = (l_1 + l_2) \cos \theta_1 \end{cases} \quad (5.86)$$

$$\begin{cases} P_{2x} = l_1 \sin \theta_2 + P_{1x} \\ P_{2y} = P_{1y} - l_1 \cos \theta_2 \end{cases} \quad (5.87)$$

Derive from the geometric relationship, we can get:

$$a = P_{2y} - P_{3y} = (P_{1y} - l_1 \cos \theta_2) - P_{3y} = (l_1 + l_2) \cos \theta_1 - l_1 \cos \theta_2 - P_{3y} \quad (5.88)$$

$$b = P_{3x} - P_{2x} = P_{3x} - (l_1 \sin \theta_2 + P_{1x}) = P_{3x} - (l_1 + l_2) \sin \theta_1 - l_1 \sin \theta_2 \quad (5.89)$$

Based on the relation of sides of right-angled triangle:

$$[(l_1 + l_2) \cos \theta_1 - l_1 \cos \theta_2 - P_{3y}]^2 + [P_{3x} - (l_1 + l_2) \sin \theta_1 - l_1 \sin \theta_2]^2 = 1 \quad (5.90)$$

Solving above triangle equation, the angle between thigh of swing leg and vertical direction θ_2 can be calculated with:

$$\theta_2 = \text{atg}\left(\frac{A}{B}\right) \mp \text{atg}\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right) \quad (5.91)$$

in which:

$$A = 2l_1 \cdot [(l_1 + l_2) \sin \theta_1 - P_{3x}] \quad (5.92)$$

$$B = 2l_1 \cdot [P_{3y} - (l_1 + l_2) \cos \theta_1] \quad (5.93)$$

$$C = (l_1 + l_2) \cdot [2P_{3x} \cdot \sin \theta_1 + 2P_{3y} \cdot \cos \theta_1 - (l_1 + l_2)] + (l_2^2 - l_1^2 - P_{3x}^2 - P_{3y}^2) \quad (5.94)$$

In equation (5.94), If $C = 0$, which means when

$$\theta_1 = \text{atg}\left(\frac{A_1}{B_1}\right) \mp \text{atg}\left(\frac{\sqrt{A_1^2 + B_1^2 + C_1^2}}{C_1}\right) \quad (5.95)$$

where

$$A_1 = 2P_{3x}(l_1 + l_2) \quad (5.96)$$

$$B_1 = 2P_{3y}(l_1 + l_2) \quad (5.97)$$

$$C_1 = 2l_1^2 + 2l_1 \cdot l_2 + P_{3x}^2 + P_{3y}^2 \quad (5.98)$$

θ_2 is a singularity value. In this case, θ_2 should be calculated as

$$\theta_2 = \text{atg}\left(\frac{-B}{A}\right) \quad (5.99)$$

Deriving the angle of shank of swing leg with respect to the vertical direction θ_3 in the similar way, based on the geometric relation, we can get:

$$l_2 \sin(\theta_2 + \theta_3) = b \quad (5.100)$$

Expansion the above expression, basing on trigonometric function,

$$(l_2 \cos \theta_2) \sin \theta_3 + (l_2 \sin \theta_2) \cos \theta_3 = P_{3x} - (l_1 + l_2) \sin \theta_1 - l_1 \sin \theta_2 \quad (5.101)$$

The above equation is concerning with θ_2 , thus, θ_3 can be calculated as:

$$\theta_3 = \text{atg}\left(\frac{F}{G}\right) \mp \text{atg}\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \quad (5.102)$$

in which

$$F = l_2 \cos \theta_2 \quad (5.103)$$

$$G = l_2 \sin \theta_2 \quad (5.104)$$

$$H = P_{3x} - (l_1 + l_2) \sin \theta_1 - l_1 \sin \theta_2 \quad (5.105)$$

When $\theta_2 = \text{atg}\left[\left(\frac{P_{3x} - (l_1 + l_2) \sin \theta_1}{l_1}\right)\right]$, $H = 0$. In this case θ_3 is a singularity value, it should

be calculated as:

$$\theta_3 = \text{atg}\left(\frac{-G}{F}\right) \quad (5.106)$$

Hip joint angle θ_2 and knee joint angle θ_3 are computed according to equation (5.91) and

(5.102), hence, when $C \neq 0$ and $H \neq 0$, there are four combination of θ_2 and θ_3 :

$$[\theta_2, \theta_3] = \begin{bmatrix} \operatorname{atg}\left(\frac{A}{B}\right) - \operatorname{atg}\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right) & \operatorname{atg}\left(\frac{F}{G}\right) - \operatorname{atg}\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \\ \operatorname{atg}\left(\frac{A}{B}\right) - \operatorname{atg}\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right) & \operatorname{atg}\left(\frac{F}{G}\right) + \operatorname{atg}\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \\ \operatorname{atg}\left(\frac{A}{B}\right) + \operatorname{atg}\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right) & \operatorname{atg}\left(\frac{F}{G}\right) - \operatorname{atg}\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \\ \operatorname{atg}\left(\frac{A}{B}\right) + \operatorname{atg}\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right) & \operatorname{atg}\left(\frac{F}{G}\right) + \operatorname{atg}\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \end{bmatrix} \quad (5.107)$$

However, in order to keep the normal walking, the knee joint of biped robot can not bend backward during both the single-support phase and stance phase. With this restriction of knee joint, the angle of thigh θ_2 and angle of shank θ_3 with respect to the vertical direction for swing leg has single combination:

$$[\theta_2, \theta_3] = \left[\operatorname{atg}\left(\frac{A}{B}\right) - \operatorname{atg}\left(\frac{\sqrt{A^2 + B^2 - C^2}}{C}\right), \operatorname{atg}\left(\frac{F}{G}\right) - \operatorname{atg}\left(\frac{\sqrt{F^2 + G^2 - H^2}}{H}\right) \right] \quad (5.108)$$

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.

Simulation

In this sub-section, we will test our approach of generation of joint angle 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.

Suppose that for a specific step, the step length of biped robot L_{step} is $0.4826m$, the maximum step height h_{max} is $0.1m$, found on the proposed approach, the simulation results of coordinates of swing foot change with hip joint angle of support leg θ_1 is denoted in figure 5.7. and both hip and knee joint angle for swing leg (θ_2, θ_3) varied by the hip joint angle of support leg θ_1 is shown in figure 5.8.

The simulation results of foot trajectory of swing leg is given in figure 5.9, in which red stick represents the support leg, green stick stands for the swinging leg, and black dot is the joint. It has to be pointed out that as we derive 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. Because we assume that the maximum step height appears in the middle of step length, in figure 5.9 the foot trajectory is symmetric to the coordinate origin

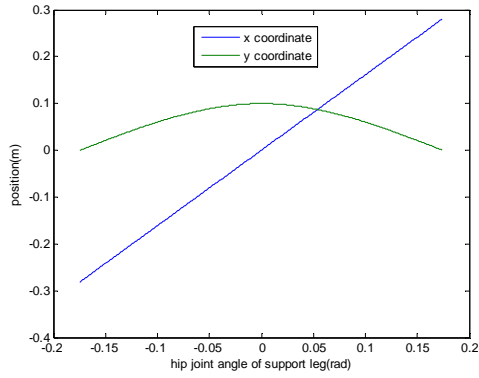


Fig.5.7 Coordinates of swing foot varied by the hip joint angle of support leg θ_1

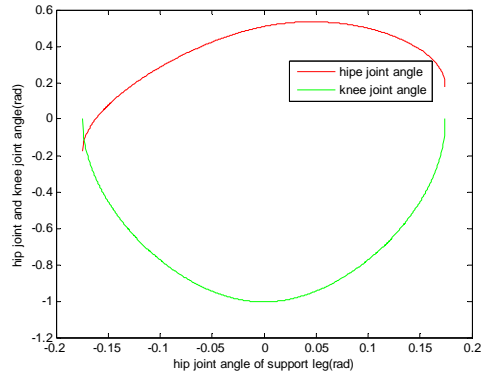


Fig.5.8 Joint angles of swing leg varied by the hip joint angle of support leg θ_1

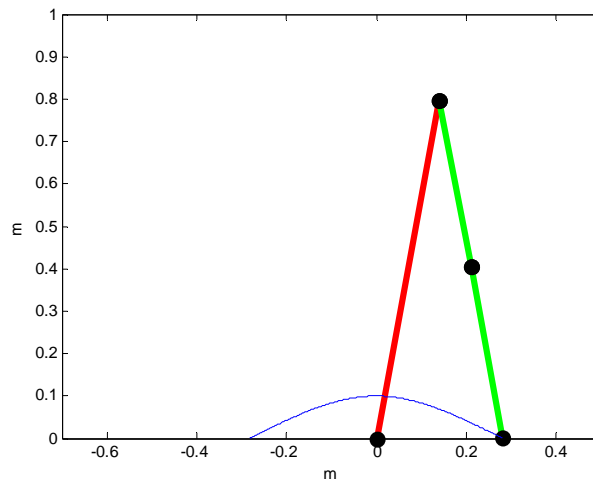


Fig. 5.9 Foot trajectory of swinging leg

5.3.1.3 Simulation results and analysis

Based on the previous developed high-level control approach, simulation is done given the example which is described in Chapter 5. We suppose that the velocity of the obstacle can be changed randomly between 0 and $0.4m/s$, while the robot can only modify duration of every step within $(0.1 \ 0.5)s$, the maximum height of step can be chosen among $[0.1 \ 0.125 \ 0.15 \ 0.175 \ 0.2]m$, and its step-length within $[0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5]m$, according to its mechanical limitation.

Footstep is designed based on fuzzy Q-learning approach. We choose discount factor $\gamma = 0.8$ and learning rate parameter $\beta = 0.1$ to learn step length, step duration time and maximum step height respective. Choosing the episode of the training phase as 10000 times, which ensures the Q matrix is trained completely with all the possible obstacle velocity. The learning results are presented in figure 5.10 and the numeric values are listed in table 5.1.

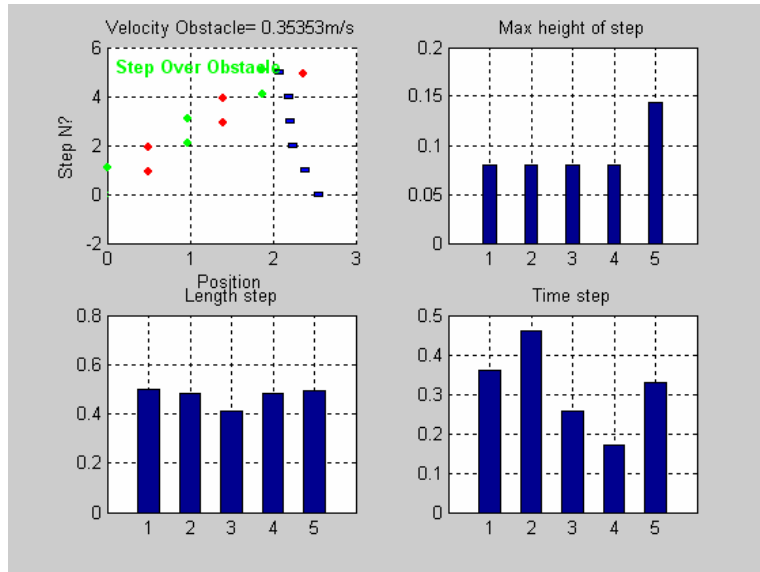


Fig. 5.10 Footstep planning results

Table 5.1 Footstep planning numeric results

Step	1	2	3	4	5
Length	0.5000	0.4802	0.4131	0.4807	0.4919
Time	0.3722	0.4646	0.2317	0.1840	0.3391
Hmax	0.08	0.08	0.08	0.08	0.1439

Cubic spline interpolation has been applied to interpolate the designated point such as starting point, landing point and maximum step height to get the swinging trajectory. With inverse kinematics, the curve of hip and knee angle of swing leg has been calculated according to the geometrical relationship between pitch angles. As the walking motion of biped robot is its left leg and right leg alternating process, the left and right leg will be the swinging leg by turns. Assume that the footstep planning starts with the left leg of biped robot as its swinging leg. Figure 5.11~5.14 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 5.15.

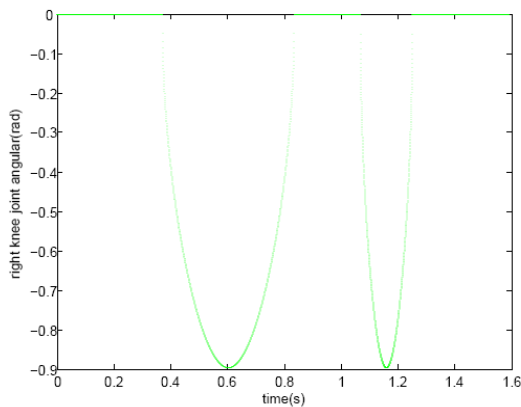


Fig.5.11 Joint angle profile for right knee

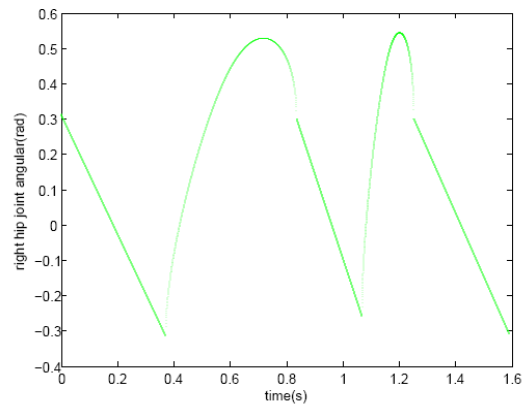


Fig.5.12 Joint angle profile for right hip

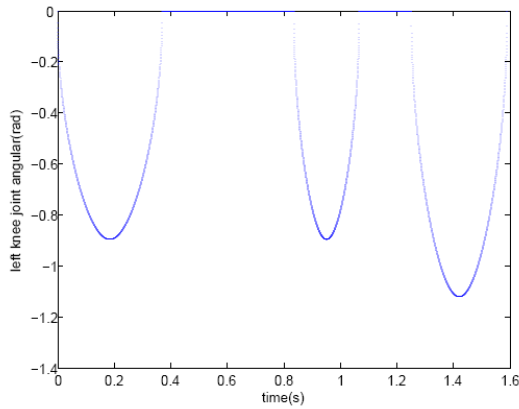


Fig.5.13 Joint angle profile for left knee

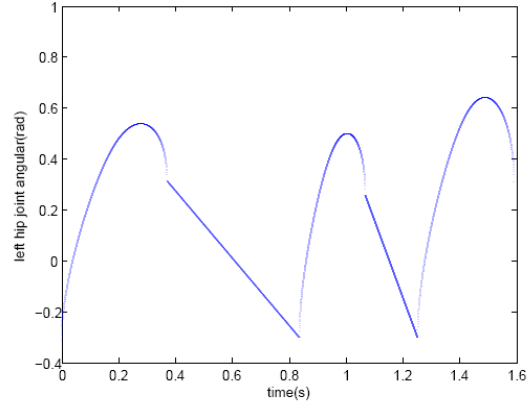


Fig.5.14 Joint angle profile for left hip

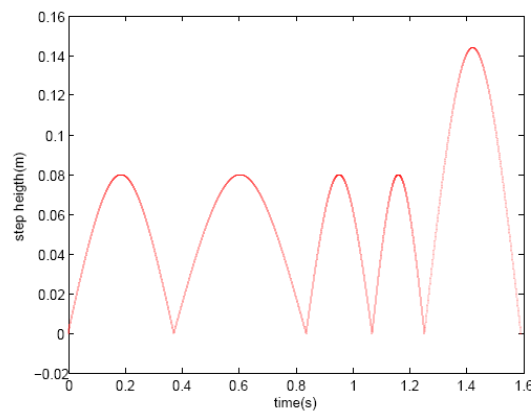


Fig.5.15 Foot trajectory of swing leg varied by step duration time

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

5.3.2 Low-level controller design

The low-level control allows both to learn the predicted swinging trajectory and to control the tracking of these desired trajectories, which can be decomposed into three parts (C.Sabourin and K.Madani, 2007, 2008):

- The first is used to compute the trajectories of the swing leg from several outputs of CMAC neural networks and a Fuzzy Inference System.
- The second one allows the regulation of the average velocity from a modification of the pitch angle of the trunk.
- The third is composed by four PD control in order to ensure the tracking the reference trajectory at the level of each joint.

Two dimension CMAC neural network is applied to learn the predicted joint angle profile

which are gained from high-level control. Regarding the coordinate of swinging foot (P_{3x}, P_{3y}) as the two inputs, two CMAC neural networks are utilized for training hip joint angle and knee joint angle separately. The hip angle of support leg θ_1 is chosen randomly within $(\theta_{1min}, \theta_{1max})$, which is the changing range of hip joint angle within one step period. The horizontal coordinate of P_{3x} can be computed with this chosen θ_1 and the corresponding vertical coordinate P_{3y} is calculated by cubic spline. The weights of CMAC are updated based on the difference between the output of CMAC and hip joint angle (or knee joint angle) of swinging leg calculated from inverse kinematics. Figure 5.16 shows the results of swinging leg joint angle approximation with CMAC, in which blue curve stands for the desired joint angle profile, red one is the hip joint angle approximation, and green curve represents the output of CMAC approximating knee joint angle.

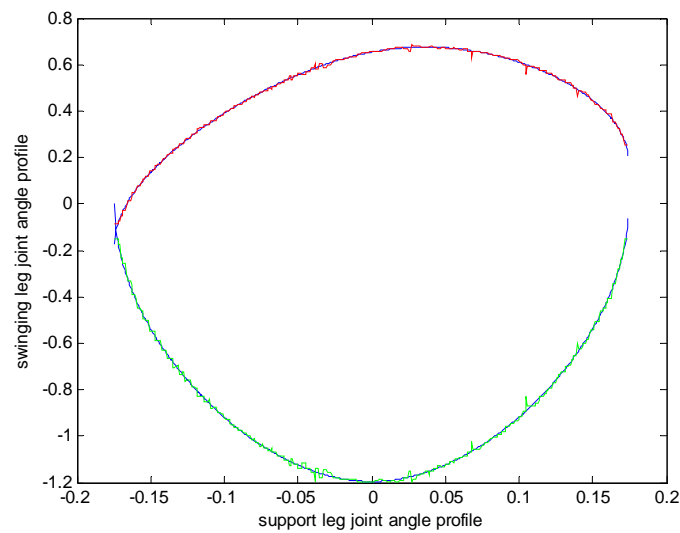


Fig. 5.16 Swinging leg joint angle approximation with CMAC

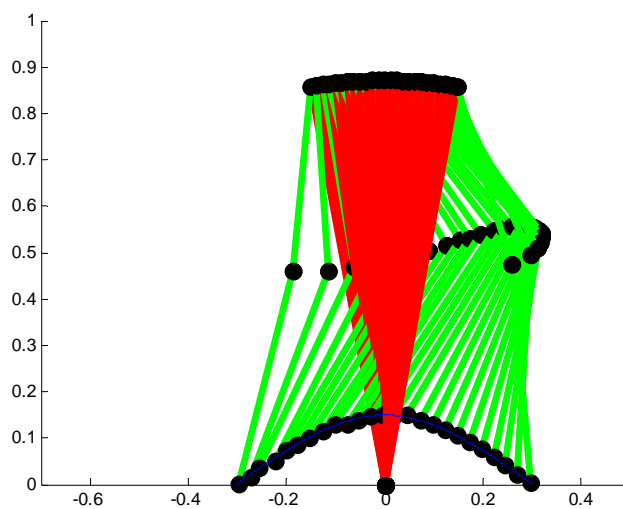


Fig. 5.17 Swinging leg trajectory approximation with CMAC

Figure 5.17 gives the visualized simulation results of swinging leg trajectory approximation

with CMAC during one step period. Similar with the previous example, green link represents the swinging leg, red link stands for the support leg and black dots are the joints.

As the walking motion of biped robot is the right leg and left leg alternates sequence, we use these trained CMAC neural networks to approximate the joint angle profiles for examples described in sub-section 5.3.1.2 (refers to figure 5.17). Suppose the stepping motion starts with the left leg of biped robot, figure 5.18~5.21 give the simulation results of the output of CMACs approximating the joint angle profiles.

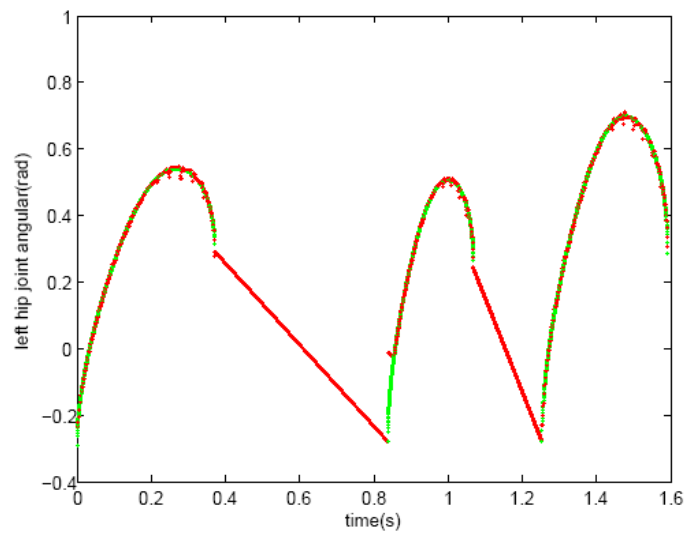


Fig. 5.18 CMAC approximation of left hip joint angle changing with step duration time

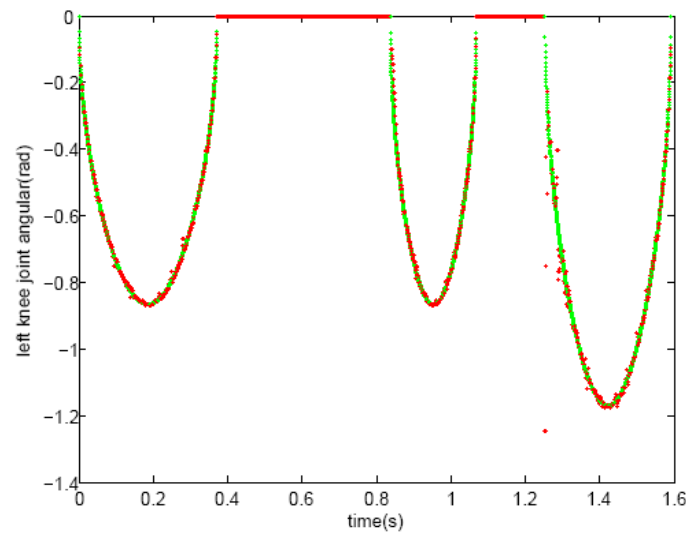


Fig. 5.19 CMAC approximation of left knee joint angle changing with step duration time

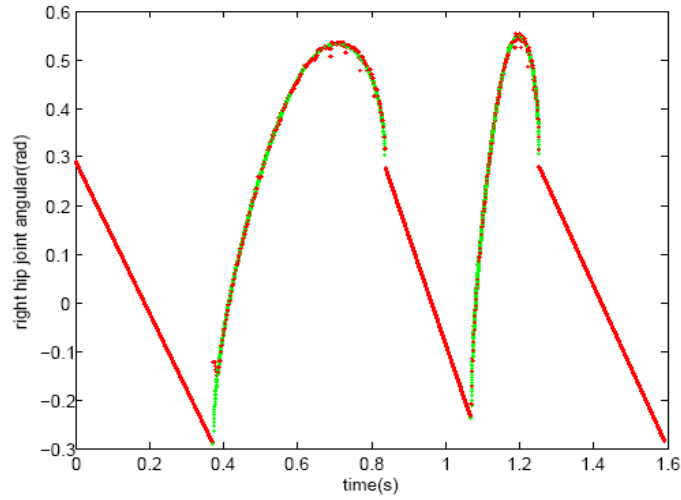


Fig. 5.20 CMAC approximation of right hip joint angle changing with step duration time

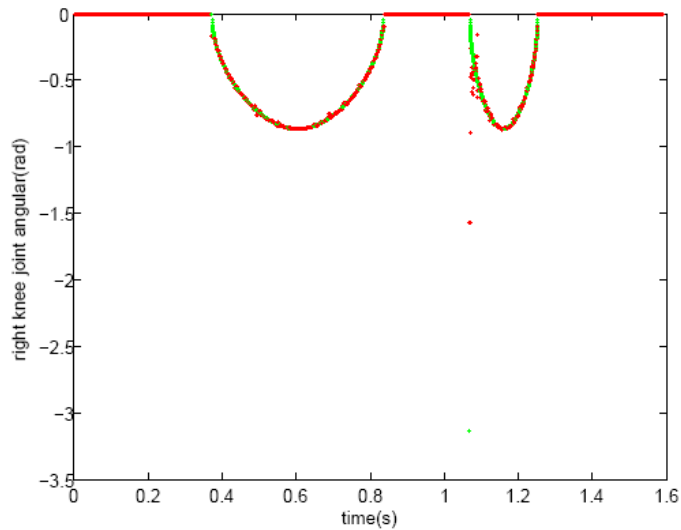


Fig. 5.21 CMAC approximation of right knee joint angle changing with step duration time

5.4 Simulation results and analysis

In this section, our high-level control and low-level strategy is tested by a simulation example where the biped robot and the dynamic obstacle moving oppositely in the sagittal plan. The virtual three-dimension dynamic model of biped robot is set up with Admas software, the controller is designed by Matlab. The data is transferred through Adams/Control connector, module and the simulation is done by united Admas 3D model with Matlab controller.

The velocity of the obstacle is $0.3m/s$, and the length of the obstacle equals to $0.1m$. Regard distance between the robot and the obstacle d_{obs} and the velocity of the obstacle v_{obs} as two inputs of footstep planning for high-level control. d_{obs} represents the distance between the

front foot and the first side of the obstacle and v_{obs} is computed from the distance covered during one step. These two input signals are updated at each double support phase.

Table 5.2 Reference gaits

<i>Gait</i>	$L_{step}(m)$	$T_{step}(s)$	$V_M(m/s)$
<i>Gait</i> ¹	0.24	0.6	0.4
<i>Gait</i> ²	0.3	0.6	0.5
<i>Gait</i> ³	0.34	0.567	0.6
<i>Gait</i> ⁴	0.38	0.543	0.7
<i>Gait</i> ⁵	0.43	0.537	0.8

The fuzzification of v_{obs} and d_{obs} is carried out by using 3 and 5 triangular membership functions respectively. Therefore, the number of the rules is 3 multiplies 5 and equals to 15. For each rule, we define 5 reference gaits which are characterized by parameters given in table 5.2.

The footstep planning is designed based on our previous idea. The Q matrix is trained for 150 episodes. During the learning phase, at each episode, the velocity is chosen in random manner with Gaussian probability $\sigma=0.02$ around the median value $0.3m/s$. The initial distance between the robot and the obstacle is around $2.5m$. Figure 5.22 shows the sum of the computing value $\Delta Q(t)$ for each episode according to the number of episode.

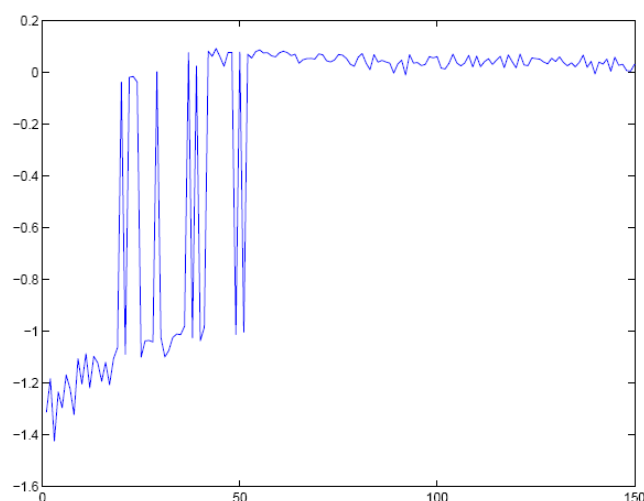


Fig. 5.22 Sum of the computing value $\Delta Q(t)$ for each episode

It must be noticed that this value, which depends directly on the reinforcement signal, converges toward 0 quickly within 60 episodes. It is possible to obtain the best rules after the learning phase. Table 5.3 gives the obtained best rules in the case of the presented example.

Table 5.3 Obtained best rules after learning stage

D_{obs}/V_{obs}	Small	Medium model	Big
V 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
V Big	Gait ³	Gait ⁵	Gait ⁵

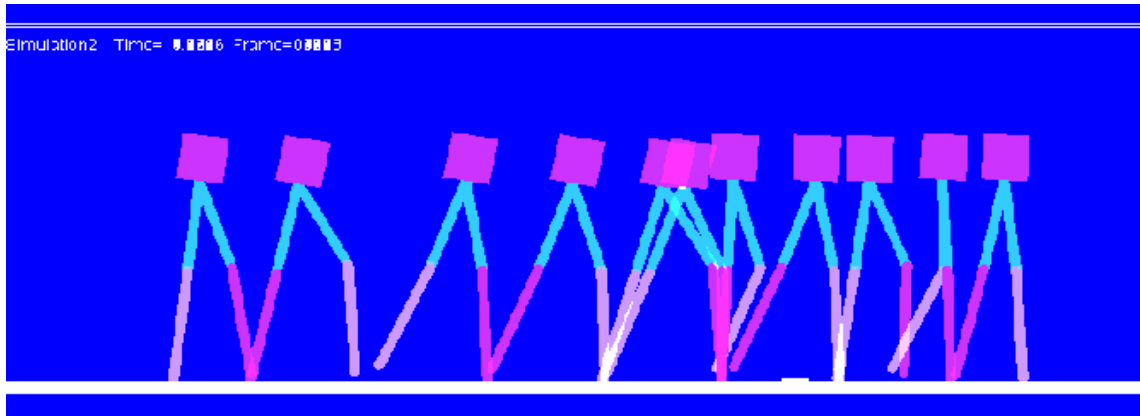


Fig. 5.23 Walking motion sequence when the robot is walking toward a dynamic obstacle

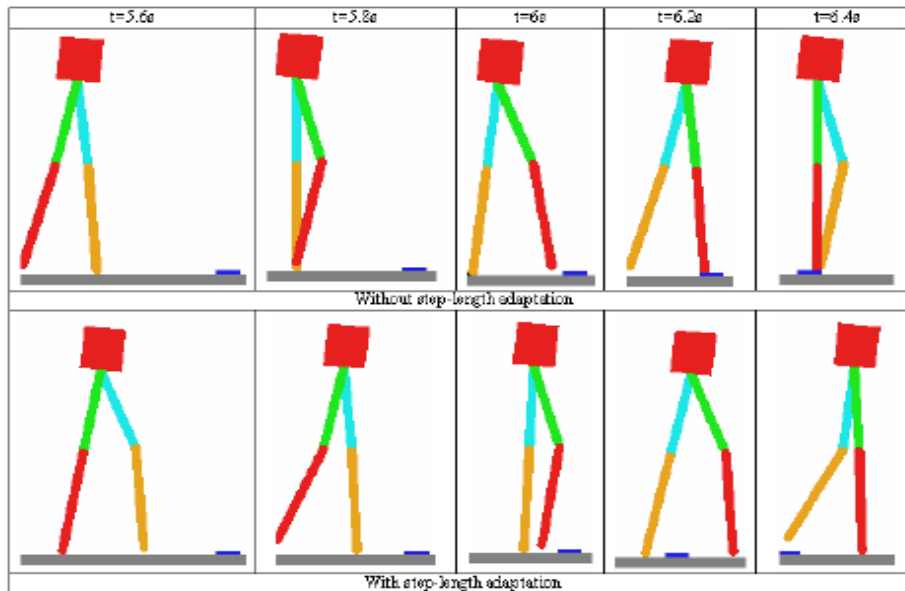


Fig. 5.24 Comparative study of footstep planning results

Consequently, we use these best rules in order to design the gait pattern based on Fuzzy CMAC neural networks. The simulation is done on the base of Adams with Matlab software. Figure 5.23 shows the walking sequence when the robot is walking towards an obstacle in the case of the proposed example. Figure 5.24 shows the comparative study of the footstep planning results.

The top figure concerns a control strategy without step length adaptation, and the bottom figure shows the biped robot which uses the proposed control strategy. Without adaptation, the length of the step is quasi-constant and equals to $0.33m$. It can be seen very easily that the robot crash into obstacle when $t = 6.2s$ in the above figure, but when the length of the step is adjusted by using the FQL footstep planning, the robot is able to avoid the obstacle.

5.5 Summary

In this chapter, we have proposed a control strategy for biped robot walking in dynamic environment, in which the robot and obstacle move with opposite direction in the sagittal plane. The originality 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 and joint angle generating. The footstep planning is derived based on a 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, can 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. CMAC neural networks have 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 where we consider a flat obstacle. Our 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 not step over the dynamic obstacle in all cases. The success rate is related to the size of obstacle. The longer the obstacle is the smaller success rate is. 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 6 Conclusion and Future Work

6.1 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 simple and effective training properties, fast learning convergence and digital hardware implementation, CMAC neural network is widely applied in the complex dynamic systems, such as robot and some flight vehicles. Usually the required memory size is dramatically increased with its input dimension. However, the memory size of CPU and computation time are quit important for the real-time control of these systems, as the CPU has to complete mass of tasks at the same time. Therefore, we concentrate on the structure optimization of CMAC neural network and its application on generic hypersonic vehicle and biped robot.

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 fuzzy Q-learning approach. It is been developed to solve the problem of biped robot stepping over obstacle in dynamic environment.

Just with the above research background, we develop our research and get some conclusion around the following points:

(1) The main drawback of high-dimension input CMAC neural network 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. We discuss how both the generalization and step quantization parameters may influence the CMAC approximation quality. 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 modeling 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, we developed the algorithm for optimizing the CMAC structure.

(2) In hypersonic air-breathing flight vehicles with airframe-integrated scramjet engines, the airframe, propulsion system and structural dynamics are highly interactive. The engine airframe integration causes significant coupling between the propulsion system and vehicle aerodynamics. Due to the strong coupling, explicit flight dynamics and designing effective controller are highly challenging for this class of vehicles. In this paper, we have developed the nonlinear longitudinal dynamic equations of GHV considering the elastic characteristics of the body. Different with the original neural network controller design, we propose two two-dimension CMAC neural networks to mapping the relationship of throttle setting and flight height and velocity, and the relation between the elevator deflection and flight height and velocity. The only assumptions made about the relation between the input (flight height and velocity), and output (throttle setting and elevator deflection) are continuous, and a sufficient condition for controllability is satisfied. The simulation is done to evaluate its performance on step change response, and the results demonstrate that the performance requirements of the GHV are met.

(3) In the presented research achievement, the footstep approach for biped robot in dynamic environment can only solve the problem of the obstacle moving with average velocity and predict velocity. Moreover, in order to avoid the high computational complexity, the planning has to be limited within a few steps. In this paper, we develop the footstep planning based on the fuzzy Q-learning algorithm. The approach is divided into two separate fuzzy Q-learning algorithm design process, 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 simulation results show that the learning approach convergence very quickly after only a few episodes. And after the learning phase, the biped robot can step over the unpredicted dynamic obstacle by adjusting the landing point and maximum step height. The main interests of this investigation are as following: (a) the computing time is very short, after the learning phase, the footstep planning is only based on two FIS. (b) This approach is valid for both static and dynamic obstacles. (c) The footstep planning is operational for both predictable and unpredictable

dynamical environment allowing the control to increase the robustness.

(4) To study the locomotion of biped robot in the sigttal plan, one has to first develop the mathematical model of the robot. The kinematics equations are derived based on the five-link model. The advantage of this five-link model is that it has sufficiently few degrees of freedom to keep the equations of motion to a manageable level, while still having 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. Our mathematical model is relatively less complexity. It is because on the one hand, we suppose that the structures of both legs are same which is correspond to the original case, and on the other hand, we choose the mass center of the trunk of biped robot as the starting point to derive the position and velocity of the mass center of each link.

(5) Base on the idea of the high-level control and low-level control can be designed separately, we propose the control strategy for biped robot stepping over dynamic obstacle. The high-level control consists of footstep planning which is found on fuzzy Q-learning algorithm, foot trajectory generation and joint angle generation. The low-level control based on the CMAC neural network, allows 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 and where we consider a flat obstacle. Our approach does not require the knowledge of probability transitions from a state to another for robot and has fast learning property.

6.2 Future work

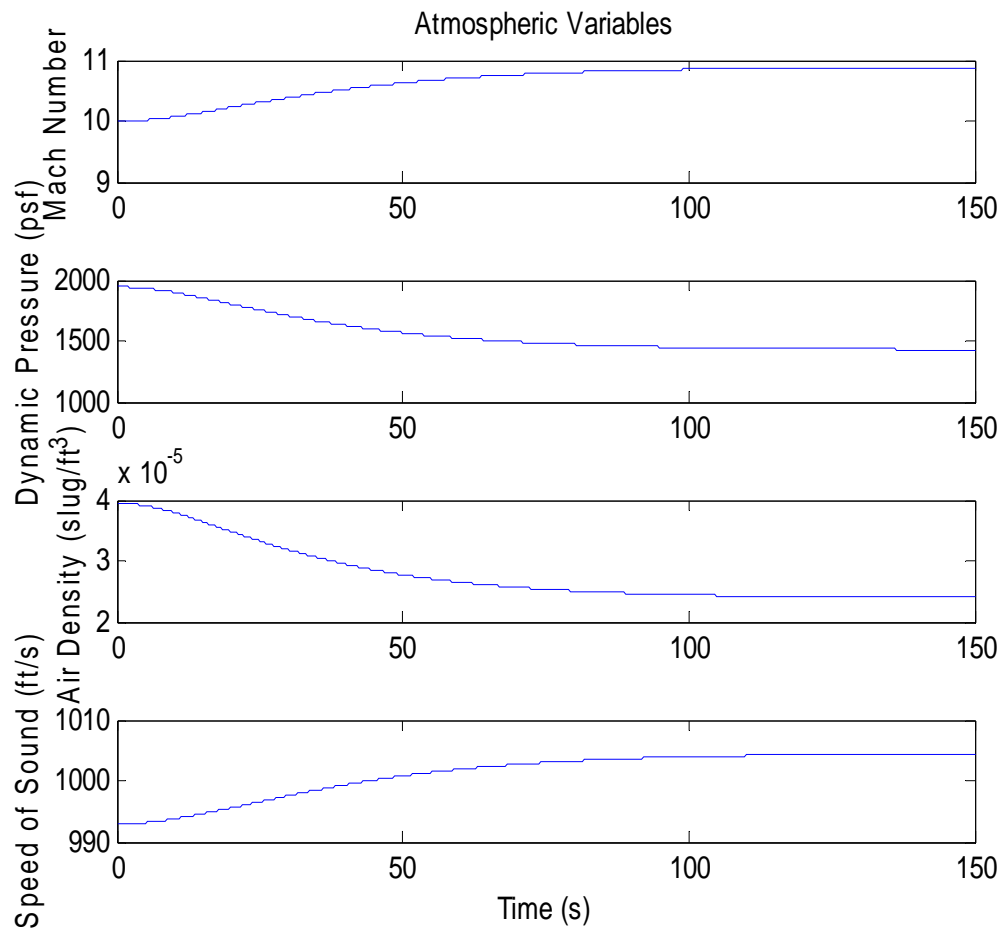
As the previous description, we have made some achievement about the footstep planning for biped robot in dynamic environment, and structure optimization for CMAC neural network in this paper. However, there are sill some insufficient which need further study.

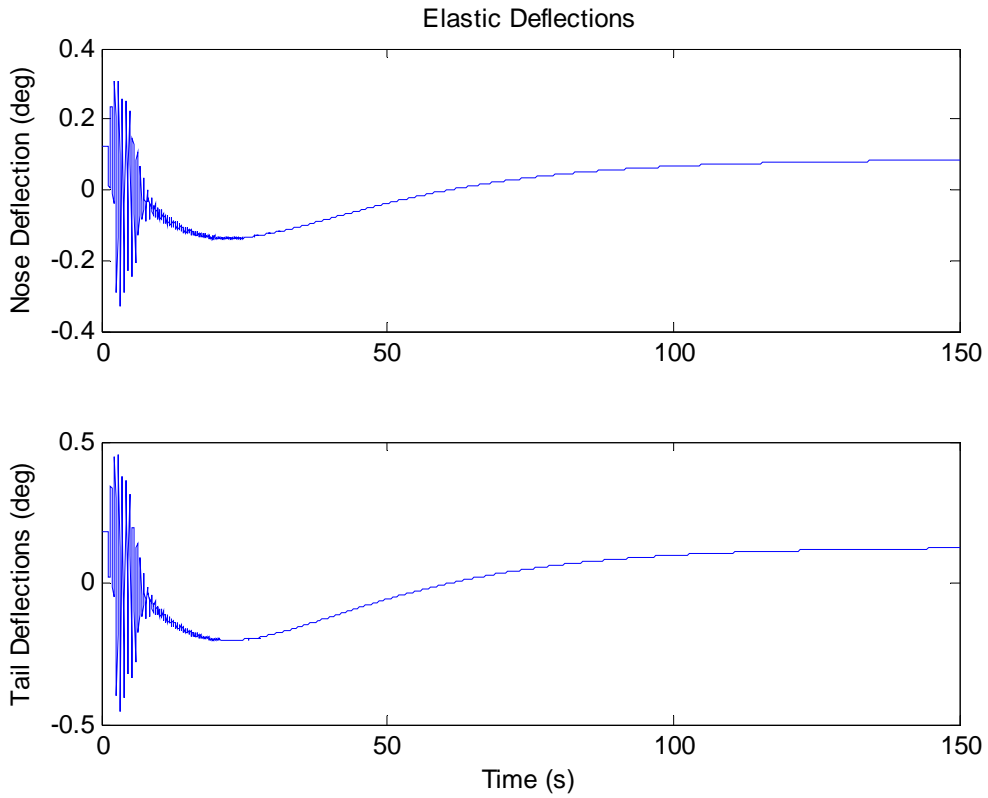
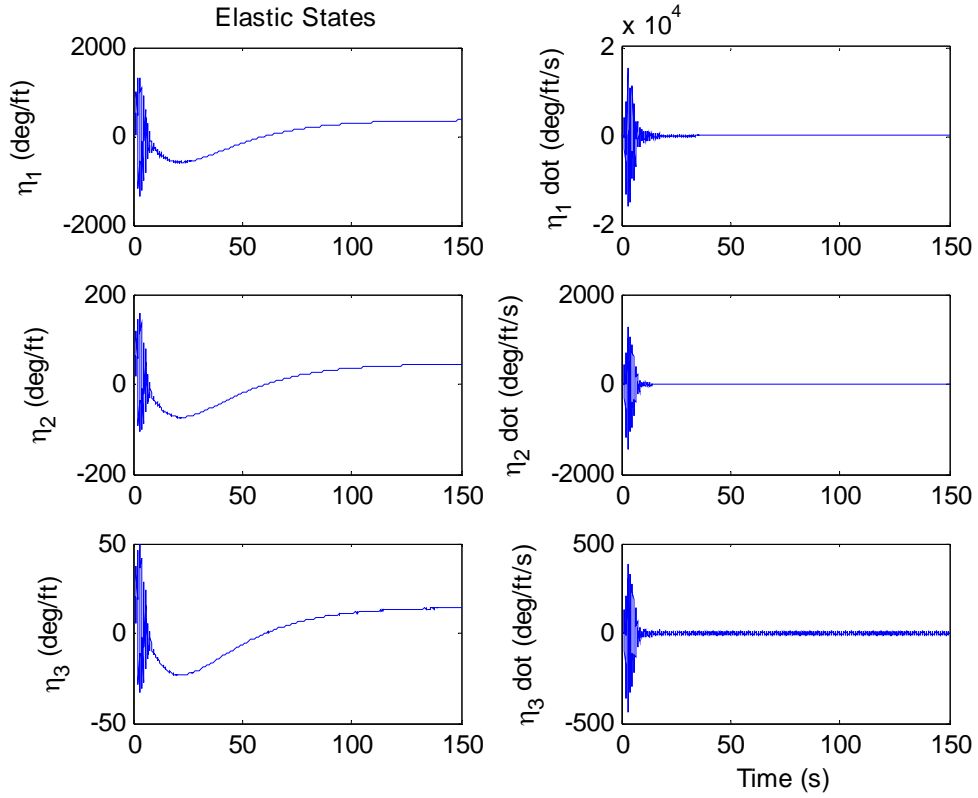
(1) In Chapter 2, we have present that the optimal structure can be achieved by adjusting the structure parameters. Through search of all the possible structure parameters, the quantization step and layer are chosen according to the reinforcement signal which contains the information about the required memory size and approximation error. In the future, we will focus on using the learning algorithm which will choose the structure parameters with no need of the ergodic search.

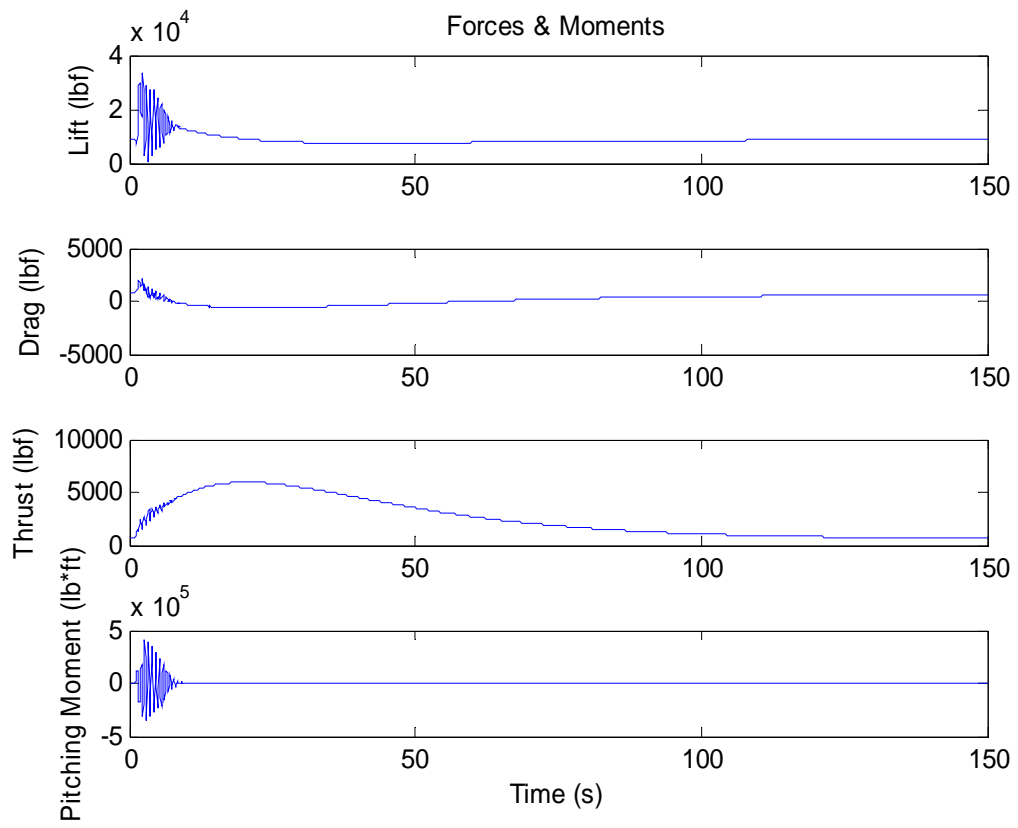
(2) In the footstep planning approach, the step length, step duration time and maximum step

height has to be adjusted according to the velocity of obstacle, however the frequently change of these parameters will influence the stability of bipedal walking robot. In the future, we will try to increase the robustness of our control strategy.

Appendix Simulation results of CMAC based controller for generic hypersonic vehicle







Reference

- A.Brnbrahim and J. Franklin, Bipeddynamic walking using reinforcement learning, *Robotics and Autonomous Systems*, 1997, Vol. 22, pp.283-302.
- A. Carlos, P. Filho. *Humanoid Robots: New Developments*. I-Tech Education and Publishing, Vienna, Austria, June 2007.
- Alekasnda Rodic and Dusko Katic, Trajectory prediction and path planning of intelligent autonomous biped robot-learning and decision making through perception and spatial reasoning, 9th Symposium on Neural Network Applications in Electrical Engineering, Serbia, 2008, 25-27.
- Aleksander Kołcz, Nigel M. Allinson, Basis function models of the CMAC network, *Neural Networks* 12 (1999) 107 - 126
- A. L. Kun, T. Miller. The design process of the unified walking controller for the UNH biped. *Proc.IEEE Conf. on Humanoid Robots*, 2000.
- Alireza Ferdowsizadeh Naeeni, *Advanced multi-agent fuzzy reinforcement learning*, Master Thesis, 2004.
- Andrew D.Clark, Maj Dean Mirmirani, Chivey Wu and Sangburn Choi, An aero-propulsion intergrated elastic model of a generic air-breathing hypersonic vehicle, *AIAA 2006-6560*.
- Aurelie Labbe, *Multiple testing using the posterior probability of half-space: application to gene expression data*, PhD Thesis, 2005.
- Bars Fidan, Maj Mirmirani, and Petros A. Ioannou, Flight dynamics and control of air-breathing hypersonic vehicles: review and new directions. *AIAA 2003-7801*.
- Bertin, J. J., Periaux, J., and Ballmann, J., editors, *Advances in Hypersonics*, Birkhäuser, Boston, MA, 1992.
- Bolender, M.A., and Doman, D.B., A non-linear model for the longitudinal dynamics of a hypersonic air-breathing vehicle, *AIAA Guidance, Navigation, and Control Conference*, *AIAA 2005-6255*, San Francisco, 2005.
- Bushcek,H. and Calise,A.J., Uncertainty modeling and fixed-order controller design for a gypersonic vehicle model, *AIAA Journal of Guidance, Control and Dynamics*, Vol.20(1), 1997, pp.42-48.
- Caironi P., Dorigo M., *Training Q-agents*, Tech. Rep. IRIDIA, n 94/14, Universit Libre de Bruxelles, 1994
- Chavez, F. R. and Schmidt, D. K., Uncertainty modeling for multivariable-control robustness analysis of elastic high-speed vehicles, *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 22(1), 1999, pp. 87-95.
- Chenglong FU, Mei SHUAI et al., Parametric walking patterns and optimum atlases for underactuated biped robot, *IEEE Intel. Conf. on Intelligent Robots and Systems*, 2006, 342-347.
- Chan-Mo Kim Kwang-Ho Choi Cho, Y.B. , *Hardware design of CMAC neural network for control applications*, *IEEE Int. Conf. on Neural Networks*, 2003.
- Chien-Kuo Li, Ching-Tsan Chiang, *Neural Networks Composed of Single-variable CMACs*, 2004 *IEEE International Conference on Systems, Man and Cybernetics* 3482-3487.

Chung Ying Amy Chan, Dynamic modeling, control and simulation of a planar five-link bipedal walking system, Master thesis, The University of Manitoba, 2000.

C.K.Chow and D.H.Jacobson, Further studies of human locomotion: postural stability and control, *Mathematical Biosciences*, 1972, Vol.15: pp.93-108.

Curran, E. T., Scramjet Engines: The first forty years, *AIAA Journal of Propulsion and Power*, Vol. 17(6), Nov. 2001, pp.1138-1148.

C. Sabourin, O. Bruneau. Robustness of the dynamic walk of a biped robot subjected to disturbing external forces by using CMAC neural networks. *Robotics and Autonomous Systems*, 2005, Vol.23, pp 81-99.

C. Sabourin, K. Madani and O. Bruneau "Autonomous Biped Gait Pattern Based on Fuzzy-CMAC Neural Networks", *Integrated Computer-Aided Engineering (ICAE)*, vol. Vol.14, n° N°1, pp. pp 1–14, 2007.

C.Sabourin, K.Madani, YU Weiwei, YAN Jie, Obstacle Avoidance Strategy for Biped Robot Based on Fuzzy Q-Learning Approach, *International Conference on Automatic Control & Robotics*, Portugal, 2008.09.

C.Sabourin, K.Madani, and O.Bruneau, Towards adaptive control strategy for biped robots, *Humanoid Robots: Human-like Machines*, I-Tech Education and Publishing, Austria, 2007, Chapter No.9, pp.191-120.

C.Sabourin, O.Bruneau, and J-G Fontaine, Pragmatic rules for real-time control of the dynamic walking of an under-actuated biped robot, *IEEE Int. Conf. on Robotics and Automation*, 2004, pp.4216-4221.

C.Sabourin, O.Bruneau, and G.Buche, Experimental validation of a robust control strategy for the robot RABBIT, *IEEE Int. Conf. on Robotics and Automation*, 2005, pp.2404-2409.

C.Sabourin, O.Bruneau, and Gabriel Buche, Control strategy for the robust dynamic walk of a biped robot, *The internal Journal of Robotics Research*, 2006, Vol.25, No.9, pp.1-17.

C. Watkins, P. Dayan. Q-learning. *Machine Learning*, 8, 1992, 279–292.

C.Zhou and Q.Meng, Reinforcement learning with fuzzy evaluative feedback for a biped robot, *Proc. IEEE Conf. on Robotics and Automation*, 2000, pp.3829-3834.

Dasgupta, A.Y. and Y. Nakamura, Making feasible walking motion of humanoid robots from human motion capture data, *Proc. IEEE Int. Conf. Robotics and Automation*, 1999, pp.1044-1049

Davidor, Y., *Genetic algorithms and robotics - A heuristic strategy for optimization*, World Scientific Publishing Co. Pte. Ltd., pp. 164, 1991.

D.E.Goldberg, *Genetic algorithms in search, optimization and machine learning*, Boston, MA, Kluwer Academic Publishers, 1989.

Eldracher, M., Staller, A., and Pompl, R. (1994). Function Approximation with Continuous Valued Activation Functions in CMAC. *Forschungsberichte Kunstliche Intelligenz FKI-199-94*, Institut fur Informatik, Technische Universitat Munchen.

El-Tantawy, S.; Abdulhai, B., An agent-based learning towards decentralized and coordinated traffic signal control. *IEEE Int. Cof. On Intelligent Transportation Systems*, 2010, pp.665-670.

Evrin Taskiran, Metin Yilmaz and etc., Trajectory generation with natural ZMP references for the biped walking

- robot SURALP, IEEE Int. Conf. on Robotics and Automation, Alaska, USA, 2010, 4237-4242.
- F.H.Glanz, T.W.Miller, and L.G.Kraft, An overview of the CMAC neural network, IEEE Conf. on Neural Networks for Ocean Engineering, 1991.
- Frank Kirchner, Sebastian Bartsch and Jose DeGea, Experiments on embodied cognition: A bio-inspired approach for robust biped locomotion, Humanoid Robots, New Developments, 2003, pp.487-504.
- Frederick G.H., Neural network control of a parallel hybrid-electric propulsion system for a small unmanned aerial vehicle, PhD Thesis, University of California, 2005.
- Fujimoto, Y. and A. Kawamura, Simulation of an autonomous biped walking robot including environmental force interaction, IEEE Robotics and Automation Magazine, 1998(6): pp.33-42.
- Fujimoto, Y. and A. Kawamura, Attitude control experiments of biped walking robot based on environmental force interaction, Workshop on Advanced Motion Control, 1998, pp.70-75.
- F. Yamasaki, K. Endo, H. Kitano and etc., Acquisition of humanoid walking motion using genetic algorithm, Considering characteristics of servo modules, Proc. IEEE Conf. on Robotics and Automation, 2002, pp.3123-3128.
- Gabor Horvath. Kernel CMAC: an Efficient Neural Network for Classification and Regression. Acta Polytechnica Hungarica, 2006, Vol.3.No.1, pp.5—20.
- Gregory, I.M., Chowdhry, R.S., Mcminn, J.D. and Shaughnessy, J.D., Hypersonic vehicle model and control law development using H_∞ and μ -synthesis, NASA TM-4562, 1994. synthesis, " NASA TM-4562, 1994.
- Hahn-Ming Lee, Chih-Ming Chen, and Yung-Feng Lu, A Self-Organizing HCMAC Neural-Network Classifier, IEEE Transactions on Neural Networks, Vol. 14, No. 1, 2003
- Hani Al-Dayaa, Theorization, implementation, system architecture and analysis of fast reinforcement learning techniques, with application to autonomous agents, PhD Thesis, University of Massachusetts Lowell, 2006
- H.Hemami, and R.L., Postural and gait stability of a planar five link biped by simulation, IEEE Transactions on Automatic Control, 1977, Vol.22, No.3: pp.452-458.
- H.Hemami, and P.C.Camara, Nonlinear feedback in simple locomotion systems, IEEE Transactions on Automatic Control, 1977, pp.855-860.
- H.Hemami and C.Jr.Golliday, The inverted pendulum and biped stability, Mathematical Bioscience, 1977, Vol.34: pp.95-110.
- H.Hemami, C.S.Robinson and A.Z.Ceranowicz, Stability of planar biped models by simultaneous pole assignment and decoupling, Int. J. Systems Sci., 1980, Vol.11, No.1: pp.65-75.
- Hurmuzlu Y., Dynamics of bipedal gait: part I-objective functions and the contact event of a planar five-link biped, Journal of Applied Mechanics, 1993, Vol.60, pp.331-336
- Hurmuzlu Y., Dynamics of bipedal gait: part II-stability analysis of a planar five-link biped, Journal of Applied Mechanics, 1993, Vol.60, pp.30-36.
- J. Chestnutt, J. J. Kuffner. A Tiered Planning Strategy for Biped Navigation. Int. Conf. on Humanoid Robots (Humanoids'04), Santa Monica, California, 2004.

- J. Chestnutt, M. Lau, G. Cheung, J.J. Kuffner, J. Hodgins, T.Kanade. Footstep Planning for the Honda Asimo Humanoid. Proceedings of IEEE Int. Conf. on Robotics Automation (ICRA), 2005, pp. 629-634
- Jens-Steffen Gutmann, Masaki Fukuchi and Masahiro Fujita, A floor and obstacle height map for 3D navigation of a humanoid robot, IEEE Int. Conf. on Robotics and Automation, 2005.
- J.Furusho and M.Masubuchi, Control of a dynamical biped locomotion system for steady walking, ASME, J. of Dynamic Systems, Measurement, and Control, 1986(6), Vol.108: pp.111-118.
- J.Furusho, M.Masubuchi, A theoretically motivated reduced order model for the control of dynamic biped locomotion, ASME, J.of Dynamic Systems, Measurement, and Control, 1987, Vol.109,pp.155-163.
- J.H.Holland, Adaptation in natural and artificial systems, Ann Arbor, University of Michigan Press, 1975.
- Jinsu Liu and Manuela Veloso, Online ZMP sampling search for biped walking planning, IEEE Intel. Conf. on Intelligent Robots and Systems, 2008, pp.22-26
- J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, H. Inoue. Footstep Planning Among Obstacles for Biped Robots. Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2001, pp.500–505.
- J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, H. Inoue. Online Footstep Planning for Humanoid Robots. Proceedings of IEEE/RSJ Int. Conf. on Robotics and Automation (ICRA), 2003, pp.932–937
- J. Nakanishi, J. Morimoto, G. Endo and etc., Learning from demonstration and adaptation of biped locomotion. Robotics and Automation Systems, 2004, Vol.47, No.2-3, pp.79-91.
- J.S. Albus. Data storage in the cerebellar model articulation controller (CMAC). Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control, September 1975, pp 228–233,.
- J.S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control, September 1975, pp 220–227.
- Kemalettin Erbatır, Akihiro Okazaki, Keisuke Obiya and etc., A study on the zero moment point measurement for biped walking robots, International Conference on AMC 2002, pp.431-436.
- K.Hirai, Current and future perspective of Honda humanoid robot, IEEE Int. Conf. on Intelligent Robots and Systems, 1997, pp.500-508.
- K.Hirai, M.Hirose, Y.Haikawa and T.Takenaka, The development of Honda humanoid robot, IEEE Int. Conf. on Robotics and Automation, 1998, pp.1321-1326.
- Kiguchi, K., Watanabe, K., and etc., A humanlike grasping force planner for object manipulation by robot manipulators, Cybernetics and Systems, Vol.34(8), 2003, pp.645-662.
- Kim, Y.H., Lewis, F.L. and Dawson, D.M., Intelligent optimal control of robotic manipulators using neural networks, Automatica, Vol.36(9), 2000, pp.1355-1364.
- K.Loffler, M.Gienger and F.Pfeiffer, Sensor and control design of a dynamically stable biped robot, Proc. IEEE Conf. on Robotics and Automation, 2003, pp. 484-490.
- Koichi Nishiwaki, Satoshi Kagami, James J. Kuffner and etc., Online humanoid walking control system and a

moving goal tracking experiment, IEEE Int. Conf. on Robotics and Automation, Taiwan, 2003.

Krista Falkner, Multiple Q-learning agents for bi-directional headway control of a high frequency streetcar line, Master Thesis, University of Toronto, 2006.

K. Sabe, M. Fukuchi, J. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision. Int. Conf. on Robotics Automation. 2004, pp.592–597.

Kyong-Sok Chang, Efficient algorithms for articulated branching mechanisms: dynamic modeling, control and simulation, PH.D.thesis, 2000.

Leslie Pack Kaelbling and Michael L. Littman, Reinforcement Learning: A Survey, <http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/rl-survey.html>.

Lewis, F.L., Neural network control of robot manipulators, IEEE Intelligent Systems, Vol.11, 1996, pp.64-75.

L. Jouffe. Fuzzy inference system learning by reinforcement methods. IEEE Trans. on SMC, Part C, August 1998, Vol. 28 (3).

Liguo Huo, Robotic joint-motion optimization of functionally-redundant tasks for joint-limits and singularity avoidance, PhD thesis, 2009, 04.

Macna,C.J., D Eleuterio and etc., CMAC adaptive control of flexible-joint robots using backstepping with tuning functions, IEEE Conf. on Robotics and Automation, 2004, pp.2679-2686.

Manish Kumar, Sensor fusion and intelligent control of autonomous cooperating robots, PhD. Thesis, 2004.

Maouche A.R. and Attari M., CMAC based adaptive control of a flexible link manipulator, Industrial Electronics, 2009, pp.1480-1485.

Marrison, C. I. and Stengel, R. F., Design of robust control systems for a hypersonic aircraft," AIAA Journal of Guidance, Control, and Dynamics, Vol. 21(1), 1998, pp. 58-63.

Mark Humphrys, "Action Selection methods using Reinforcement Learning" <http://www.compapp.dcu.ie/~humphrys/PhD/index.html>

Meng Joo Er; Yi Zhou; Chiang-Ju Chien; Gait Synthesis Self-generation by Dynamic Fuzzy Q-Learning Control of Humanoid Robots, IEEE Int. Conf. on Systems, Man and Cybernetics, 2006, pp.4231-4236.

Meuleau N., Le dilemme exploration/exploitation dans les systemes d'apprentissage par renforcement, thèse de l'Université de Caen, 1996.

M. Hackel. Humanoid Robots: Human-like Machines. ITech Education and Publishing, Vienna, Austria, June 2007 .

M.Hirose, Y.Haikawa, T.Takenaka and etc., Development of humanoid robot ASIMO, IEEE Int.Conf. on Intelligent Robots and Systems, Workshop2, 2001.

Ming-Feng Yeh Hung-Ching Lu, On-line adaptive quantization input space in CMAC neural network, Systems, Man and Cybernetics, 2002

Ming-Feng Yeh and Kuang-Chiung Chang, A Self-Organizing CMAC Network With Gray Credit Assignment, IEEE Transactions on Systems, Man, and Cybernetics, Vol 36, No.3, 2006, pp.623-635.

- Mingyang Chen, Control design and robustness measurement for biped locomotion, University of Missouri-Columbia, PH.D. thesis, 1996.
- Ming-Feng Yeh. Single-input CMAC control system. *Neurocomputing*, Volume 70, Issues 16-18, October 2007, pp 2638-2644
- M.Mitchell, An introduction to genetic algorithms, Cambridge, MA, MIT Press, 1996.
- Mohsen M.Dalvand and Majid M.moghadam, Stair climber smart mobile robot, *Autonomous Robots*, 2006, 3-14.
- Mo-Yuen Chow A Menozzi. A self-organized CMAC controller,. In *Proceedings of the IEEE International Conference on Industrial Technology*, 1994, pp: 68-72
- M.Vukobratovic and B.Borovac, Zero moment point-thirty five years of its live, *International Journal of Humanoid Robotics*, 2004, Vol.1, No.1, pp.157-173.
- M.W.Walker, D.E.Orin, Efficient dynamic computer simulation of robotic mechanisms, *Transactions of ASME Journal of Dynamic Systems, Measurement, and Control*. Vol104(3):pp205-211, 1982.
- M.Yagi and V.Lumelsky, Biped robot locomotion in scenes with unknown obstacles, *IEEE Int. Conf. on Robotics and Automation*, Michigan, 1999.
- Nearchou, A.C. and Aspragathos, N.A., Application of genetic algorithms to point-to-point motion of redundant manipulators, *Mechanism and Machine Theory*, v. 31, n 3, pp. 261-270, April, 1996
- Nearchou, A.C., Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm *Mechanism & Machine Theory*, v. 33, n. 3, pp. 273-292, April 1998.
- Nitin Mathur, Design of intelligent adaptive control using immune based algorithm, Master Thesis, 2004
- N.Kanehira, T.Kawasaki, S.Oita and etc., Design and experiment of advanced leg module for humanoid robot (HRP-2) development, *Pro. Int. Con. on Intelligent Robots and Systems*, 2002, pp.234-239.
- O.G.Rudenko, A.A.Bessonov.. CMAC Neural Network and Its Use in Problems of Identification and Control of Nonlinear Dynamic Objects. *Cybernetics and Systems Analysis*, Vol 41, N° 5, pp 1060-0396
- O. Haavisto.. Master's thesis, Helsinki University of Technology, 2004.
- Olli Haavisto and Heikki Hyotyniemi, Simulation tool of a biped walking robot model, Helsinki University of Technology Control Engineering Laboratory, Report, 2004.
- Onyshko,S., and Winter, D.A., A mathematical model for the dynamics of human locomotion, *Journal of Biomechanics*, 1980, Vlo.13, pp.361-368.
- Oussarna Khatib. A unified approach to motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, 1987, Vol3(1), pp43-53.
- Park, J.H. and Y.K. Rhee, ZMP Trajectory generation for reduced trunk motions of biped robots, *Proc.IEEE/RSJ Int.Conf. Intelligent Robots and Systems*, 1998(10): pp.90-95.
- Park, J.H and H.C. Cho, An on-line trajectory modifier for the base link of biped robots to enhance locomotion stability, *Proc. 2000 IEEE Int. Conf. on Robotics and Automation*, 2000, pp.3353-3358.
- Pawel M.P., Fuzzy anticipatory learning classifier system for mobile robot navigation, Master Thesis, University of

Alberta, 2007

Philipp Michel, Joel Chestnutt, James Kuffner and etc., Vision-guided humanoid footstep planning for dynamic environments, Int. Conf. on Humanoid Robots (Humanoids'05), 2005.

Philipp Michel, Joel Chesnutt, Satoshi Kagami, and etc., Online environment reconstruction for biped navigation, IEEE Int. Conf. on Robotics and Automation, 2006.

Prasad Kulkarni, Dip Goswami, Prithwijit Guha and Ashish Dutta, Path planning for a statically stable biped robot using PRM and reinforcement learning, Journal of Intelligent Robot System, 2006(47), pp.197-214.

P.Y. Glorennec, L. Jouffe. Fuzzy Q-Learning Proc. of FUZZ-IEEE'97, Barcelona, 1997.

P. Y. Glorennec. Reinforcement Learning: an Overview. European Symposium on Intelligent Techniques (ESIT), 2000,17–35.

Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, K. Tanie. Planning walking patterns for a biped robot. IEEE Transactions on Robotics and Automation, 2001, Vol.17, N3, pp. 280-289.

Q.Wu, Lyapunov stability analysis of a class of base-excited inverted pendulums with application to bipedal locomotion systems, University of Manitoba, Ph.D. Thesis, 1996.

Q.Wu, Thornton-Trump, A.B., and etc., Lyapunov stability control of inverted pendulums with general base point motion, International Journal of Non-Linear Mechanics, 1998, Vol.33, No.5: pp.801-818.

R.E.King, Computational intelligence in control engineering, New York, Marcel Dekker, Inc., 1999.

Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction" <http://www.cs.ualberta.ca/~sutton/book/ebook/the-book.html>

R.S. Sutton, A.G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.

Shan,J., Fumio Nagashima, Neural locomotion controller design and implementation for humanoid robot HOAP-1, The 20th Annual Conference of the Robotics Society of Japan, 2002.

Shaughnessy, J.D., Pinckney, S.Z., Mcminn, J.D., Cruz, C.I., and Kelley, M.L., Hypersonic vehicle simulation model: winged-cone configuration, NASA TM-102610, 1990.

Shahriar Keshmiri, Richard Colgren and Maj Dean Mirmirani, Six Dof nonlinear equations of motion for a generic hypersonic vehicle, AIAA 2007-6626.

Shimon A.W., Adaptive representations for reinforcement learning, PhD Thesis, University of Texas at Austin, 2007.

Siegler,S., Seliktar,R., and Human,W., Simulation of human gait with the aid of a simple mechanical model, Journal of Biomechanics, 1982, Vol.15(6), pp.415-452.

S. Kajita, F. Kanehiro, K. Kaneko and etc., Biped walking pattern generation by using preview control of Zero-Moment Point, Proc. IEEE Conf. on Robotics and Automation, 2003, pp.1620-1626.

S.Kagami, K.Nishiwaki, J.J.Kuffner and etc., Vision-based 2.5D terrain modeling for humanoid locomotion, IEEE Int. Conf. on Robotics and Automation, 2003.

Sorao, K., T.Murakami and K.Ohnishi, A unified approach to ZMP and gravity center control in biped dynamic

stable walking, Proc. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, 1997, CD-Rom.

Stengel, R. F., Optimal Control and Estimation, Dover, New York, NY, 1994.

S.Tzafestas, M.Raibert and C.Tzafestas, Robust sliding-mode control applied to a 5-link biped robot, Journal of Intelligent and Robotic Systems, Vol.15: pp.67-133, 1996.

Sutton, R. S., and Barto, A. G. (1998). Reinforcement Learning: An Introduction. Cambridge, Massachussets: MIT Press.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. Machine Learning, 3:9–44

Suzuki, H.; Yamakita, M.; Hirano, S.; Luo, Z.W., Falling-down avoidance control for acrobat robot by Q-Learning with Function Approximation, SICE Annual Conference, 2008, pp.863-841.

Szabó, T. and Horváth, G) CMAC Neural Network with Improved Generalization Capability for System Modelling. Proc. of the IEEE Conference on Instrumentation and Measurement, 2002,. Vol. II, pp.1603-1608.

Teknomo, Kardi, 2005, Q-Learning by Examples.

<http://people.revoledu.com/kardi/tutorial/Reinforcement/Learning/index.html>

T.Ishida, A small biped entertainment robot SDR-4X II, 2003 IEEE Internation Symposium on Computational Intelligence in Robotics and Automation, 2003, pp.1046-1051.

T. Miller, B. A. Box, E. C. Whitney, J.M. Glynn. Design and Implementation of a High Speed CMAC Neural Network. Proceedings of the conference on Advances in neural information processing systems, 1990, pp 1022 – 1027.

Valasek, J.; Doebbler, J.; Tandale, M.D.; Meade, A.J., Improved Adaptive–Reinforcement Learning Control for Morphing Unmanned Air Vehicles, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, 2008, pp.1024-1020.

Vlachogiannis, J.G.; Hatziargyriou, N.D. Reinforcement learning for reactive power control, IEEE Trans. on Power Systems, 2004, pp.1317-1325.

Wallner , Elmar Well and Klaus, Attitude control of a reentry vehicle with internal dynamics, AIAA Guidance, Navigation, and Control conference and Exhibit, 2002-4647

Walton, J. T., Performance Sensitivity of Hypersonic Vehicles to Change Angle of Attack and Dynamic Pressure, AIAA Paper89-2463, 1989.

Wang, Q. and Stengel, R. F., \Robust nonlinear control of a hypersonic aircraft," AIAA Journal of Guidance, Control, and Dynamics, Vol. 23(4), 2000, pp. 577-585.

Wen Chen, Recurrent neural networks applied to robotic motion control, Master Thesis, 2002

WERUAGA Luis ; MORALES Juan; VERDU Rafael; Active training on the cmac nonlinear adaptive system, European Signal Processing Conference, September 6-10, 2004, Vienna, Austria

Watkins C., Learning from delayed rewards, PhD Thesis, University of Cambridge, England, 1989.

W. T. Miller, F. H. Glanz, L. G. Kraft. CMAC: An associative neural network alternative to backpropagation. Proceedings of the IEEE, Special Issue on Neural Networks, 1990, vol.78,N°10, pp 1561-1567.

- Wyatt J., Exploration and inference in learning from reinforcement, PhD, University of Edinburgh, 1997.
- Yang, S.X. and Meng, M.Q.-H., Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach, IEEE Trans. on Neural Networks, Vol.14(6), 2003, pp.1541-1552.
- Y. Ayza, K. Munawar, M. B. Malik, A. Konno and M. Uchiyama. A Human-Like Approach to Footstep Planning. Humanoid Robots, I-Tech Education and Publishing, Vienna, Austria, June 2007, pp.296–314
- Ying Huo and Petros A.Ioannou, Robust adaptive fault-tolerant control with trajectory optimization, 45th AIAA Aerospace Sciences Meeting and Exhibit, Nevada, AIAA 2007-155.
- Zeman, V., Patel, R.V. and Khorasani, K., Control of a flexible joint robot using neural networks, IEEE Trans. on Control Systems Technology, Vol.5, 1997, pp. 453-462.
- Zhou, K. and Doyle, J. C., Essentials of Robust Control , Prentice-Hall, Upper Saddle River, NJ, 1998.

Publications

1. YU Weiwei, C.Sabourin, K.Madani, YAN Jie, Design of Footstep Planning Controller for Humanoid Robot in Dynamic Environment, 2008 IEEE International Symposium on Knowledge Acquisition and Modeling, China, Wuhan, 2008.12; (EI index)
2. YU Weiwei, C.Sabourin, K.Madani, YAN Jie, CMAC structure parameters and its structure optimization, Journal of Northwestern Polytechnical University, 2008.12; (EI index)
3. C.Sabourin, YU Weiwei, K.Madani, YAN Jie, Obstacle Avoidance Strategy for Biped Robot Based on Fuzzy Q-Learning Approach, International Conference on Automatic Control & Robotics, Portugal, 2008.09; (SCI index,EI index)
4. Zhong Dudu, Yu Weiwei, Zhang Kai, Lin Yi and Yan Jie, Design and Test of DMD Dynamic IR Scene Simulator, Journal of System Simulation, 2008.10; (EI index)
5. Chang Xiaofei, Yu Weiwei, Yan Jie, Controlling lunar lander in powered descending phase, 2009IEEE Vehicle Power and Propulsion Conference, 2009.09; (EI index)
6. YU Weiwei, C.Sabourin, K.Madani, Self-optimizing of structure parameters for CMAC neural network, 2008 IEEE International Symposium on Knowledge Acquisition and Modeling, 2010.11;
7. YU Weiwei, C.Sabourin, K.Madani, YAN Jie, CMAC Neural network Structure Optimization for Function Approximation, Neural Networks and Artificial Intelligence International Conference, Belarus, Minsk, 2008.05;
8. C.Sabourin, YU Weiwei, K.Madani, YAN Jie, Foot Step Planning for Biped Robot Based on Fuzzy Q-Learning Approach, International Conference on Informatics in Control, Automation & Robotics, Portugal, 2008.05.
9. Yan Binbin, Lu Cunkan, Yu Weiwei, Yan Jie, Fuzzy CMAC control design for an air-breathing hypersonic cruise vehicle, 4th IEEE Conference on Industrial Electronics and Applications, 2009, pp298~301. (EI Index)