THE UNIVERSITY OF QUEENSLAND

Undergraduate Thesis

Bachelor of Engineering (Mechatronic)

Feed-forward Control Algorithm based on Fuzzy Logic

Christopher Myatt

The University of Queensland

The School of Information Technology and Electrical Engineering

October 2004

Christopher Myatt
University of Queensland
October 2004

Professor Paul Bailes,
Acting Head of School,
School of Information Technology and Electrical Engineering,
University of Queensland,
St Lucia QLD 4072.

Dear Professor Bailes,

In accordance with the requirements of the degree of Bachelor of Engineering (Mechatronic), I present the following thesis entitled

"Feed-forward Control Algorithm based on Fuzzy Logic"

This thesis project was conducted under the supervision of Dr Gordon Wyeth.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

Christopher Myatt

ACKNOWLEDGEMENTS

I would sincerely like to thank all the people involved in making this thesis possible. This thesis could not have been completed without their contributions and support.

- My family for their unconditional support of my work throughout my university time.
- Dr Gordon Wyeth for his guidance and encouragement of this research topic.
- Damien Kee for his time and invaluable knowledge of all things GuRoo.
- Fellow undergraduate thesis students, for helping me keep up the social aspects of life, and always providing a laugh.

ABSTRACT

The purpose of this thesis is to develop a self-supervised, learning, feed-forward control algorithm. The motivation to develop such an algorithm is the University of Queensland's GuRoo robot. The GuRoo has significant errors in the control of the motors, due to such forces as gravity, backlash of gear trains, and less than accurate inverse dynamics of the joints. A Fuzzy Associative Memory (FAM) block learns a compensation signal that augments the input signal to the conventional control loop. The control algorithm is designed to minimise the error for cyclic motions, where the error can be predicted at each point in the phase of the cycle.

The FAM algorithm is based on Fuzzy Logic, which involves fuzzification of input values to a fuzzy set using a membership function. The Associative Memory system then uses the fuzzy set as an index to a lookup table, which contains the appropriate compensation signal. This compensation signal is learnt based on the position error in the system, and is stored in the lookup table.

The algorithm was tested on a second-order cart model, showing a significant improvement in trajectory tracking of the cart over a conventional controller. The shape of the membership function within the Fuzzy Logic system, and the learning rate were found to be the largest factors influencing the behaviour of the system. Preliminary development of the algorithm on the GuRoo simulator has shown promising results. Compensation for single joints has produced significant trajectory tracking improvement. Multi-joint compensation has been complicated by simultaneous learning of the joints which has introduced instabilities to the system. This area will require further development before the algorithm can be completely utilised by the GuRoo robot.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

The aim of this thesis is to develop a control algorithm that is able to accurately control the motion of dynamic systems, in particular, robotic arms, without precise knowledge of the inverse dynamics of the system. In the past, the inverse dynamics of a system have been approximated to control the system. This technique may provide favourable results in some applications, however there will always be some error if the inverse dynamics have not been correctly approximated. This thesis provides a new control approach that eliminates the error in an approximated control system.

The developed algorithm is based on the Trajectory Error Learning (TEL) model. The Trajectory Error Learning model learns by eliminating the trajectory error in a control system. The form of the error must be cyclic so that the model may provide a compensation signal corresponding to each degree throughout the cycle.

The algorithm was developed and tested on a simple second order model, proving the concept. The algorithm was then applied to the University of Queensland's GuRoo simulator. Testing on the GuRoo simulator showed that the algorithm gave significant trajectory tracking improvement.

1.1. THE PROBLEM

The conventional control approach of Proportional Integral Derivative (PID) control is the best way to control a system, given the inverse dynamics are known, or may be determined otherwise. For complex systems, such as robot arms, it may be difficult or even impossible to determine the inverse dynamics of the system.

By using an approximated controller, modelled on the approximate inverse dynamics of the system, the control system may contain some degree of error. This may be in the form of an error in phase, amplitude, or a combination of these. As an example, consider the position plot of a second order dynamic system shown in Figure 1.1 below. The position of the actuator is controlled using a suboptimal Proportional Derivative (PD) controller. The red line depicts the desired position of the actuator, and the blue line depicts the actual position of the actuator.

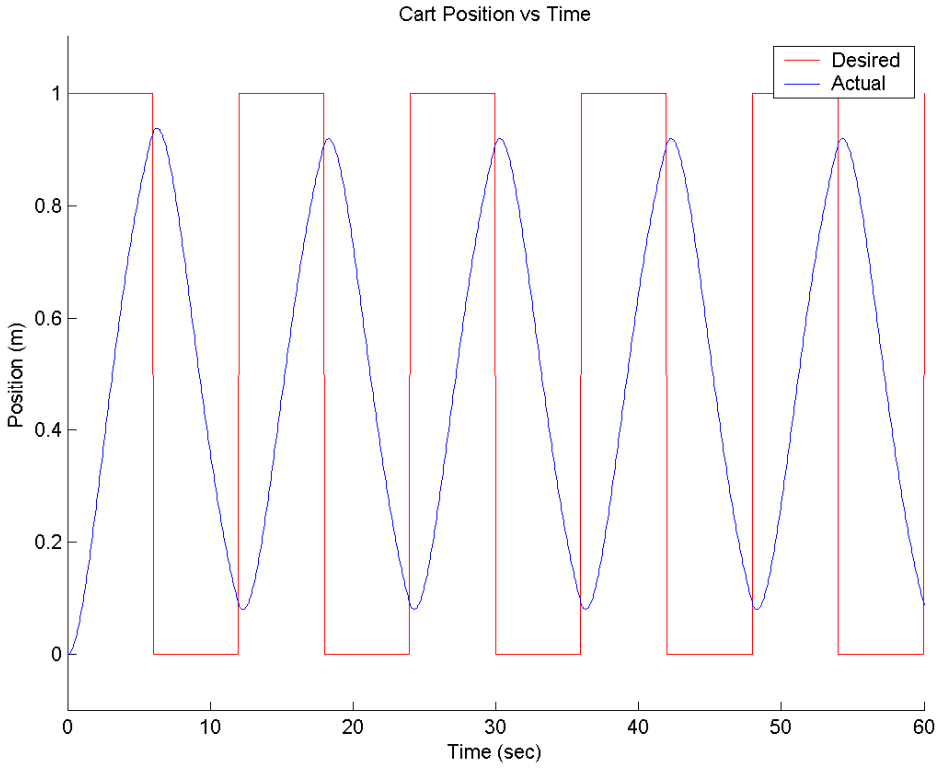

Figure 1.1 – Desired and actual position versus time

The difference between the desired and actual position of the actuator is an example of the error that may be generated by a control system using an approximation of the inverse dynamics, and hence, an approximated control system. In this example, there is a significant error in both the amplitude, phase and waveform of the signal. The aim of this thesis is to develop a control algorithm that will minimise the error to zero over a period of time.

1.2. THE APPROACH

An algorithm based on Fuzzy Logic provides the basis for the implementation of the Trajectory Error Learning model. The system also takes advantage of an Associative Memory system, allowing the system to become adaptive. In the past, Fuzzy Logic has been used as an alternative approach to conventional PID control theory. In this application, Fuzzy Logic will be used in conjunction with an existing PID control loop.

The architecture of the control system flow has been developed previously by Kee and Wyeth *[Kee, Wyeth, 2003]*. The architecture of the control system contains a feed-forward component, which contains the Fuzzy Associative Memory (FAM) system. This component provides a compensation signal to the desired signal, to minimise the error in the system. The compensation signal is driven by the error in the system, and the phase of the desired position. Figure 1.2 shows the architecture of the control system.

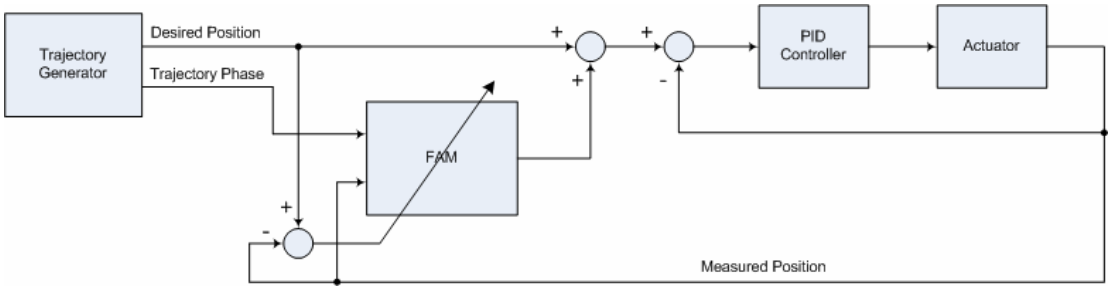

Figure 1.2 – Architecture of control system

1.3. THE GUROO

The University of Queensland has a research project concerned with the development of a humanoid robot, called the GuRoo. The GuRoo is a 1.2 metre tall, 40 kilogram, 23 degree of freedom android robot, shown below in Figure 1.3. The GuRoo is part of an international competition known as RoboCup.

The ultimate goal of the RoboCup project is by 2050, to develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer *[The RoboCup Federation, 1998-2003]*. To achieve this goal, a team of robots must be built that can perform agile athletic tasks such as running, kicking, and tackling. These are long-term goals which will only be met by first creating a robot with a strong, dynamic gait. To achieve this, the control of the joints must be very precise. The GuRoo is able to walk short distances with little trouble, albeit with some trajectory error in its joints.

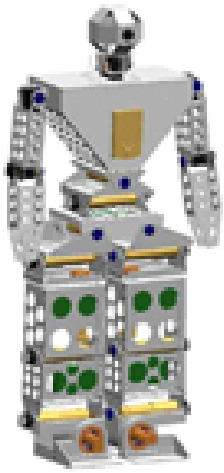

Figure 1.3 – University of Queensland's Robot, the GuRoo *[Pike, 2003]*

1.4. GOALS

The fundamental aim of this thesis is to determine if a Fuzzy Associative Memory system is a viable model to minimise the error in the control of dynamic systems, in particular, the GuRoo robot. Further to this aim, it is hoped that the Fuzzy Associative Memory system may provide an alternative to the previously developed Cerebellar Model Articulation Controller (CMAC) system *[Kee, 2003]*.

Specifically, the aims of the Fuzzy Associative Memory system are:

- Provide a simple control structure that may be used to more accurately control dynamic systems.
- Use a small amount of computer memory and processor power, such that implementation on a microcontroller is viable.
- Be applied to the GuRoo to give the robot an ability to be more human-like. That is, allow the robot to 'learn' how to control its limbs in any given situation such as in a new physical environment.

1.5. THESIS OUTLINE

The following chapters document the research and development of this thesis.

Chapter 2 describes the Trajectory Error Learning model and the previous research completed on the Cerebellar Model Articulation Controller network. This chapter outlines the Fuzzy Associative Memory system and other studies in this field. Further explanations of Fuzzy Logic and the adaptations of this technique are detailed in this chapter also.

Chapter 3 describes the integration of the FAM algorithm into the TEL model.

Chapter 4 documents the development and testing of algorithm on the single dimension, second order cart model. The chapter goes through the process of developing the algorithm, testing the algorithm, and porting the algorithm from the Matlab platform, to the C platform. This chapter also delivers the fixed-point arithmetic implementation of the FAM algorithm.

Chapter 5 documents the development and testing of the FAM algorithm on the GuRoo simulator. The chapter covers a series of experiments performed, with explanations of the results.

Chapter 6 provides a conclusion to the work and outlines the direction of any further work on this topic.

2. LITERATURE REVIEW

The problem of mobility is one of the largest problems faced when creating a humanoid robot *[Kee, 2003]*. Creating a robot that can walk on two legs with a regular human-like gait is an extremely difficult problem. Actuation of the joints is one of the most significant factors of this problem. A human has many muscles that generate large torques to move each of its limbs, controlled by senses, such as touch, sight, hearing, balance and neural information. To command a robot to perform such behaviours as walking and running, the control of its artificial limbs must be very precise.

2.1. TRAJECTORY ERROR LEARNING

The Trajectory Error Learning technique was described by Collins in 2002. Trajectory Error Learning uses the trajectory error generated to train a learning module. A system implementing Trajectory Error Learning should theoretically give a response that will maintain zero trajectory error *[Collins, 2002]*. The purpose of Collins' implementation of the Trajectory Error Learning technique was to reduce the error caused in a controlled system by sensory delay. Although this is a problem in many control systems, the effects of sensory delay in the GuRoo are insignificant when compared to the errors caused by approximated inverse dynamics and mechanical issues such backlash of the gear trains.

2.2. CEREBELLAR MODEL ARTICULATION CONTROLLER

The learning module used by Collins was the Cerebellar Model Articulation Controller neural network. Collins showed that the CMAC network could learn a compensation signal for the TEL model. This work was motivated by a sensory delay problem. Kee furthered this work to use Trajectory Error Learning, based on a CMAC neural network to assist a conventional Proportional Integral (PI) controller with trajectory tracking for a robot joint. Kee showed that the CMAC neural network, based on the TEL model could minimise the trajectory tracking error in a robot joint. *[Collins, 2002] [Kee, 2003]*

The CMAC neural network is an alternative to the back propagation-trained analog neural network. The CMAC has advantages including local generalisation, functional representation, output superposition, and fast practical hardware realisation. The CMAC may be described in three main stages: the input space, the mechanism of generalisation, and the output stage. The input space consists of a quantised set of all allowable inputs. The mechanism of generalisation may be viewed as a set of lookup tables, known as Association Units (AUs). Each input combination is mapped to an address for each AU. At the output stage, the weights from each of the AUs are summed to produce a real output. *[Miller; Glanz; Kraft, 1990] [Kee, Wyeth, 2003]*

The system has the ability to learn by adjusting the weights within the AUs according to an error signal during a learning phase. The CMAC system implemented on the GuRoo was trained using a training rule based on the Least Means Square training rule. This training rule is shown in the equation below. *[Kee, 2003] [Widrow, Stearns, 1985]*

$$w_{new} \leftarrow w_{old} + \frac{\alpha}{n}(\varepsilon) \qquad (1)$$

where: w_{new} = new weight value w_{old} = old weight value
α = learning rate n = number of AUs
ε = error

2.3. FUZZY LOGIC CONTROL SYSTEMS

Fuzzy control systems are an alternative control method to conventional PID control. Fuzzy control has many advantages, including that an accurate analytical model is not required, human experience may be applied, provides greater robustness and fault-tolerance *[Si; Zhang; Tang, 1999]*. While PID controllers provide a precise mathematical output for a given input, fuzzy logic works on a set of `IF-THEN` rules, integrated by membership functions. Fuzzy systems work in three stages. In the first stage, the inputs are 'fuzzified' using membership functions. This converts the numerical inputs into fuzzy states. The fuzzy rules are then applied to the 'fuzzified' inputs, producing a new set of 'fuzzified' variables. These variables represent the output or solution variable. Finally, the new fuzzy set is 'defuzzified', to produce a single output variable to the problem. Membership functions are used to determine the fuzzy set to which a value belongs and the degree of membership in that set *[Self, 1990]*.

By combining a neural network with a fuzzy logic controller, the ability for a control system to 'learn' becomes a reality. The integration of a neural network gives the fuzzy rules the ability to change with time. For the control system to internally generate the fuzzy rules, an appropriate neural network scheme is required. Fuzzy Associative Memory is a control system which combines a Self Organised Feature Mapping (SOFM) neural network methodology with a fuzzy logic control system structure *[Si; Zhang; Tang, 1999]*. Figure 2.1 shows the Fuzzy Associative Memory Model.

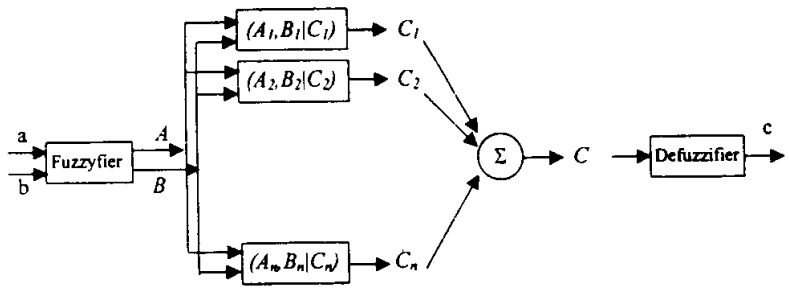

Figure 2.1 – The Fuzzy Associative Memory model *[Si; Zhang; Tang, 1999]*

2.4. EXPLANATION OF FUZZY LOGIC

The Fuzzy Associative Memory system is based on Fuzzy Logic. Therefore, to understand how the FAM works, it is important to understand how Fuzzy Logic works. As declared above, a Fuzzy Logic system works in three stages, fuzzification of input variables using membership functions, application of fuzzy rules, and defuzzification of fuzzy variables to a real output variable. Of these three layers, the fuzzification and defuzzification layers are implemented directly in the FAM system; however the application of the fuzzy rules is not. The rules form part of the Associative Memory layer of the FAM. The rules will appear to evolve over time, according to the error in the system.

The fuzzification layer is facilitated by using a 'membership function'. Membership functions come in a variety of shapes, and choosing the shape of the membership is a vital part of the design process for a Fuzzy Logic system. A common shape for a membership function is a simple triangle. Figure 2.2 shows an example of a simple triangular shaped membership function.

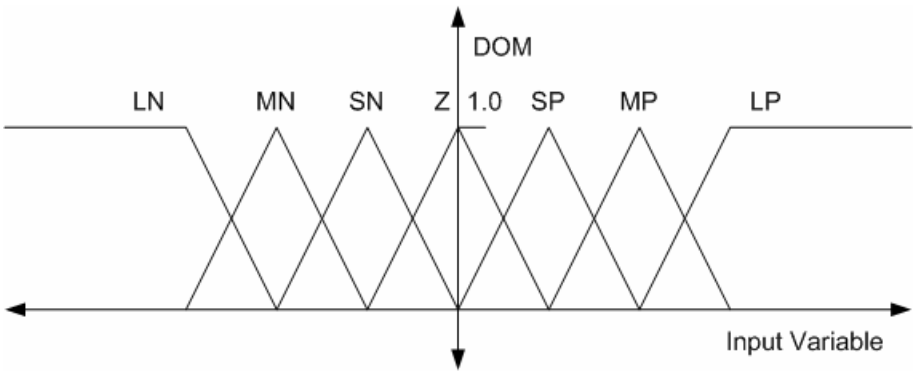

Figure 2.2 – Triangular shaped membership function

The membership function is made up of a series of fuzzy variables. In Figure 3.2 there are seven fuzzy variables. These variables are abbreviated as follows:

LN: Large Negative

MN: Medium Negative

SN: Small Negative

Z: Zero

SP: Small Positive

MP: Medium Positive

LP: Large Positive

It is important to remember that the fuzzy variable names are used only as an example, and in a fuzzy system, there may be an arbitrary number of fuzzy variables for each input variable depending on the design of the system. The fuzzy variables represent a state of the input variable. The Degree of Membership (DOM) gives a measure of the amount the input variable corresponds to the given fuzzy variable. The set of numbers given by the membership function is called a fuzzy set. Because of the nature of the triangular shaped membership function, no more than two fuzzy variables can have a non-zero DOM in a fuzzy set. For example, Figure 2.3 shows the fuzzification of the input value = 1.2. The corresponding fuzzy set for that input variable would be {0, 0, 0, 0, 0.8, 0.2, 0}.

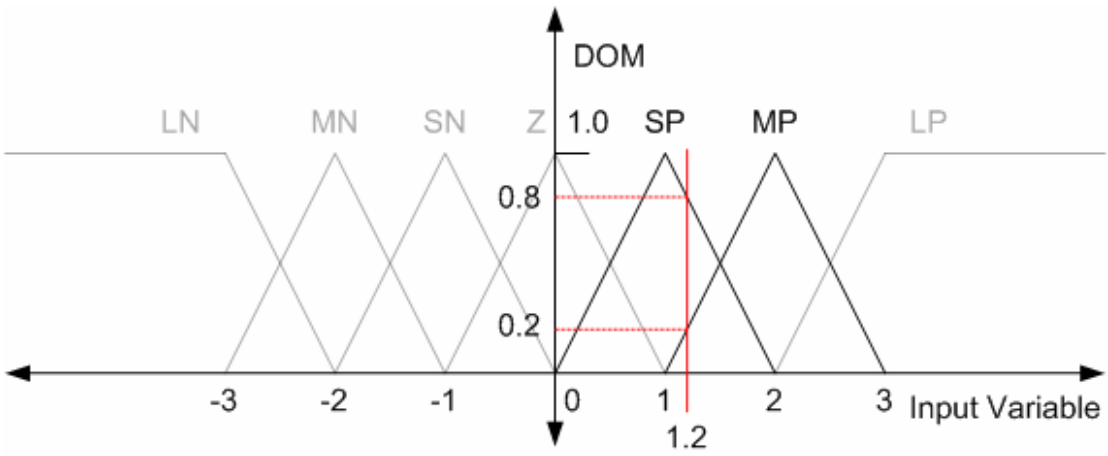

Figure 2.3 – Fuzzification of input value = 1.2

In any Fuzzy Logic system, all input variables are fuzzified to form their corresponding fuzzy sets. The next step is the application of the fuzzy rules. In a classic Fuzzy Logic system, the rules will be made up of a set of `IF THEN` statements. For example, with a two input system, `IF Input01 is LN AND Input02 is MN THEN Output01 is MP`.

Defuzzification is concerned with generating a real output parameter from the outputs of the fuzzy rules. There are many methods of defuzzification, including:

- Singleton method
- Centre of gravity method
- Mean of maximum method
- Weighted sum method

This thesis is primarily concerned with the weighted sum method, which will now be explained further. Take for instance a two input variable system with a triangular shaped membership function. As explained above, no more than two fuzzy variables in each fuzzy set can have a non-zero DOM. This would mean that at most, there would be four applicable fuzzy rules, giving four output fuzzy variables. The weighted sum defuzzification method involves multiplying the DOM of the corresponding fuzzy variables of a rule with the output fuzzy variable value. This process would be repeated for each fuzzy rule, and the results summed to give a real output parameter.

For example, with the names of the fuzzy variables abbreviated as above, let a two input Fuzzy Logic system have the fuzzy sets of:

Input01 = {0.0, 0.0, 0.0, 0.0, 0.4, 0.6, 0.0}, and

Input02 = {0.0, 0.0, 0.0, 0.0, 0.0, 0.7, 0.3}, with fuzzy variable names of

{LN, MN, SN, Z, SP, MP, LP}.

Let the only applicable fuzzy rules be:

```
IF Input01 is SP AND Input02 is MP THEN Output01 is MP,
IF Input01 is MP AND Input02 is MP THEN Output01 is MP,
IF Input01 is MP AND Input02 is LP THEN Output01 is LP.
```

Also, let the output fuzzy variables have the arbitrary values of:

{-60.0, -40.0, -20.0, 0.0, 20.0, 40.0, 60.0}.

The weighted sum value of the real output parameter would be:

$$\begin{aligned}\text{Output01} &= (0.4 \times 0.7 \times 40.0) + (0.6 \times 0.7 \times 40.0) + (0.6 \times 0.3 \times 60.0) \\ &= 38.8\end{aligned}$$

2.5. ASSOCIATIVE MEMORY

The Associative Memory system has two purposes. The first purpose is to implement the Fuzzy Logic rules. A complete set of generic fuzzy rules are implemented by the Associative Memory system. This implementation gives the system the ability to evolve the behaviour of the rules while the system is running. To fulfil a complete set of generic fuzzy rules, there must be a complete set of output fuzzy variables. This means that if a two input fuzzy system is fuzzified into five fuzzy variables each, there would be five times five, which equals twenty-five fuzzy output variables. This may be seen easily by looking at the system from a multi-dimensional perspective. The number of dimensions of the output fuzzy variable set is equal to the number of input variables. So a two input fuzzy system would have a two dimensional fuzzy output variable set. Figure 2.4 shows this in a graphical manner.

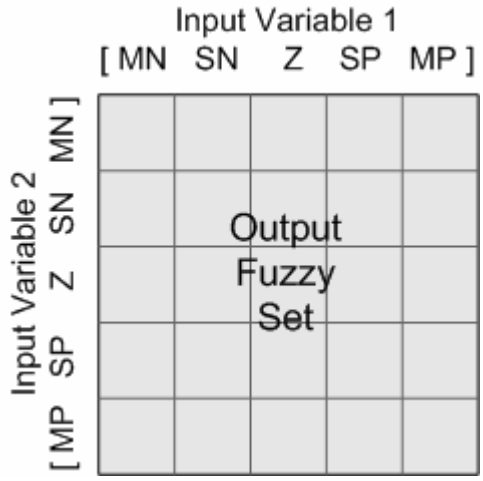

Figure 2.4 – Graphical description of input and output fuzzy sets

The second purpose of the Associative Memory is to give the system a way of altering the output fuzzy variable values while the system is running. This way, the output of the system may seemingly 'evolve' as the system iterates throughout each cycle. The output fuzzy variable set is given the name of the 'lookup table'.

When the system is started, the lookup table is initially empty, that is, full of zeros. Throughout each iteration, the lookup table is then updated using a learning rule. The learning rule is directly proportional to the error in the system. The effect of the error in the system is governed by a constant known as the *learning rate*. The learning rule based on equation (1) above, is described by the following equation:

$$w_{new} = w_{old} + \alpha \times \varepsilon \qquad (2)$$

where: w_{new} = new weight value w_{old} = old weight value

α = learning rate ε = error

Only the appropriate table entries are updated on each control loop. The table entries that will be updated are decided according to which input fuzzy variables are non-zero. For example, consider the two input fuzzy sets Input01 = {0.0, 0.0, 0.6, 0.4, 0.0} and Input02 = {0.0, 0.2, 0.8, 0.0, 0.0}. Figure 2.5 shows the four entries in the lookup table that would be updated according to the learning rule.

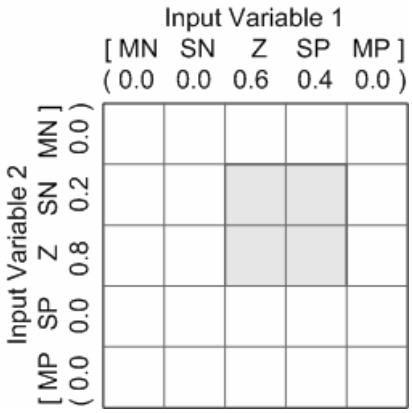

Figure 2.5 – Corresponding lookup table entries to be updated

2.6. DESIGN OF A FAM

A Fuzzy Associative Memory system is a combination of a Fuzzy Logic control system and an Associative Memory system. When designing a FAM, there are some important design parameters which must be carefully chosen to ensure stability, and to provide acceptable learning times. A list of design parameters is described below:

- Regarding the shape of membership functions;
  - Generally the shape is of triangular form, however may be otherwise,
  - If triangular, the width and the number of triangles, otherwise, the shape of each of the fuzzy variables in the membership function,
  - The limits of the membership functions, that is, the input range of the membership function where input values are distinguishable.
- Learning rate for the learning rule – a larger rate will cause the system to respond faster, however may increase overshoot, and cause stability problems.

The shape of the membership function has further implications in that the larger the number of fuzzy variables in a system, the larger the lookup table, and hence the more memory used by the control algorithm, and the more time required iterating throughout the elements in the `for-loops`, making the algorithm more processor intensive.

2.7. COMPARISON OF FAM TO CMAC

FAM and CMAC have some similarities as well as some distinguishing features.

FAM and CMAC are both local generalisers, meaning that similar inputs will provide similar outputs. Both FAM and CMAC make use of a lookup table structure. This gives both control systems the ability to be easily implemented in the digital realm, and provides quick response times. FAM and CMAC can be applied to large networks because of the small number of calculations per input *[Miller; Glanz; Kraft, 1990]*.

CMAC uses more memory than FAM, because the input-output characteristics of CMAC are continuous, while those of FAM are discrete. CMAC must be trained offline, whereas FAM may calculate the outputs online.

The most distinguishing advantage of FAM over CMAC is the ability for a FAM controller to learn online. This means that when a physical situation changes, the robot may learn how to behave in the new environment, rather than having to be turned off and taught offline.

3. OVERVIEW OF DESIGN: FAM

Following the research on FAM and CMAC, the approach to be taken with the thesis will be to implement a FAM control system according to the TEL technique.

3.1. TRAJECTORY ERROR LEARNING

The Trajectory Error Learning provides the basic model for the Fuzzy Associative Memory system. Trajectory Error Learning, as its name suggests, involves minimising the trajectory error in the system. Trajectory Error Learning is applicable to systems that have a cyclic motion. The cyclic motion of the system will cause a cyclic error. This cyclic error may be minimised by augmenting the desired trajectory according to a compensation signal. The compensation signal is unique for each known position throughout the cyclic motion. With each iteration of motion, the error becomes smaller, affecting the compensation signal less. This pattern continues until the error is minimised, and the change in the compensation signal is insignificant. This evolution of the compensation signal over time may be seen as the *learning time* of the system.

As has already been introduced, the architecture of the control system involves a feed-forward component, in which the Fuzzy Associative Memory will reside. Figure 3.1 shows this architecture.

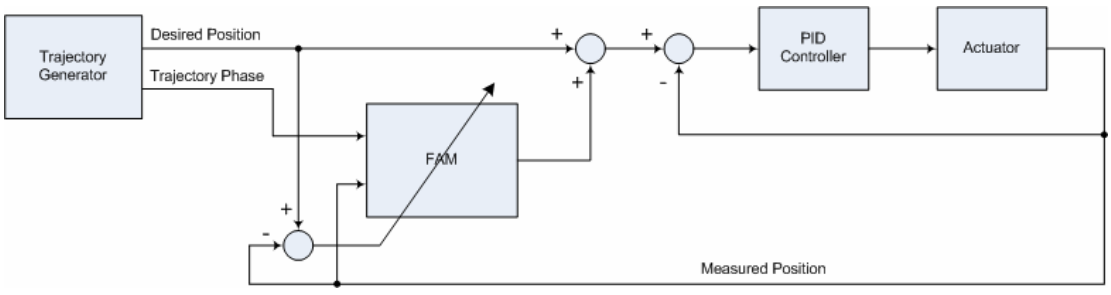

Figure 3.1 – Architecture of control system

The Trajectory Generator block provides the system with the desired position of the actuator, and the phase of the cycle for the desired position. The control system may be defined by a conventional PID controller approximating the inverse dynamics of the dynamic system. The Fuzzy Associative Memory system uses the inputs of the desired and actual position to derive the error in the system, while the trajectory phase provides the FAM with the necessary information regarding the current stage of the trajectory. The FAM provides a compensation signal that is added to the desired position signal, to compensate for the error in the controller, and with the aim to achieve exactly the desired position at the output.

In this thesis, a study of the triangular shaped membership function will be performed. A learning rule based on the Least Means Square training rule will be used, see equation (2) above.

4. CART MODEL DEVELOPMENT

In order to develop and test the Fuzzy Associative Memory control algorithm, a single plane-of-motion, second order cart model was developed in Matlab. This section presents the second order cart model, and the subsequent experimental results.

4.1. MATLAB/SIMULINK IMPLEMENTATION

A second order cart model was developed and implemented in Matlab. The free body diagram shown in Figure 4.1 shows the external and damping forces on the cart.

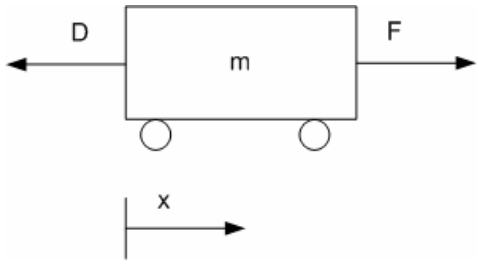

Figure 4.1 – Free body diagram of cart

4.1.1. Cart Transfer Functions

The following equations describe the mathematics and explain the derivation of the transfer function of the cart model.

$$F = m\ddot{x} + D\dot{x}$$
$$\Rightarrow F(s) = \left(ms^2 + Ds\right)X(s)$$
$$\Rightarrow \frac{X(s)}{F(s)} = \frac{1}{ms^2 + Ds}$$
$$\Rightarrow G(s) = \frac{1/m}{s^2 + D/m\, s}$$

With a cart mass of ten kilograms and a drag force of two kilograms per second, the plant transfer function is as follows.

$$G(s) = \frac{0.1}{s^2 + 0.2s}$$

To provide an adequate amount of error in the system, a rough controller was developed using Matlab's SISO tool. A suboptimal PD controller was developed, with transfer function shown below.

$$G_C(s) = 2 \times (2s + 1)$$

The controlled system produces a step response as shown in Figure 4.2. It can be seen that the controller is quite slow, with rise time about six seconds, and settle time approximately ten seconds.

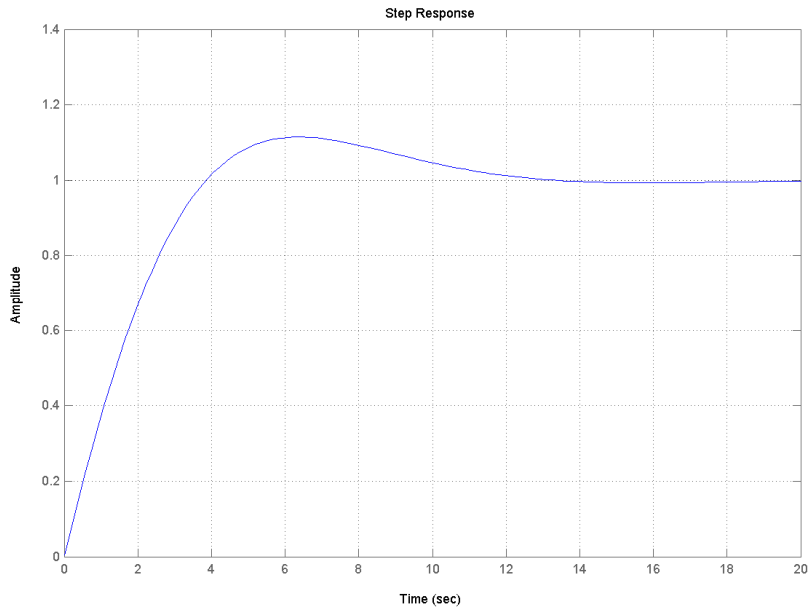

Figure 4.2 – Step response of controlled system

4.1.2. Simulink Model

Figure 4.3 shows an abbreviated summary of the model developed in Matlab. The trajectory generator is a sine wave, and the trajectory phase is generated by a saw-tooth signal.

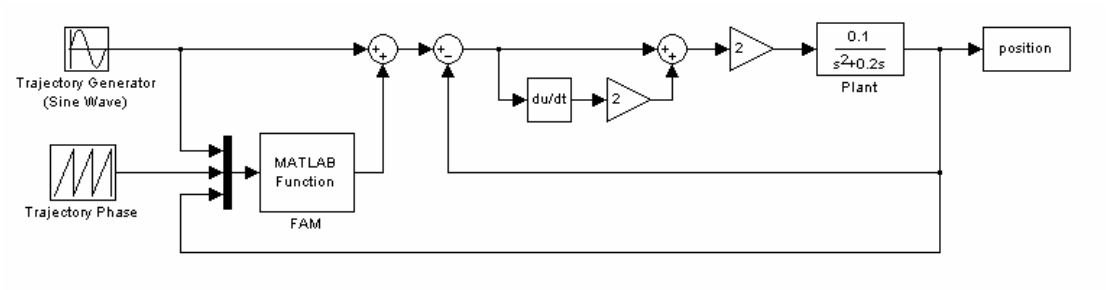

Figure 4.3 – Simulink model

The Fuzzy Associative Memory system is called via a Matlab callback function. The FAM algorithm is implemented in this Matlab function. The inputs to the FAM are the actual cart position, desired cart position and the phase. The actual cart position and phase signals are fuzzified, while the desired cart position is used in conjunction with the actual cart position to calculate the error in position.

4.1.3. Matlab Data Structures

The FAM algorithm utilizes Matlab's data structures. The input variable data structures are made up of four members:

1. An identification string for the input variable,
2. A scalar value describing the half width of the triangles in the membership function,
3. An array containing the centres of the triangles in the membership function,
4. An array containing the degree of membership of each of the fuzzy variables to the input value.

4.1.4. *Implementation of the Fuzzification Layer*

The fuzzification layer of the algorithm is broken into two main sections. The higher level section deals with iterating through each fuzzy variable. The lower level section calculates the corresponding degree of membership of the input value to a fuzzy variable according to the membership function. The calculation of the degree of membership for each fuzzy variable requires finding the corresponding value on the triangle. To find the value on the triangle, consider the triangle in Figure 4.4, with the centre of the triangle, c, and input variable value, u. There are three different conditions; $u < c$, $u = c$, and $u > c$.

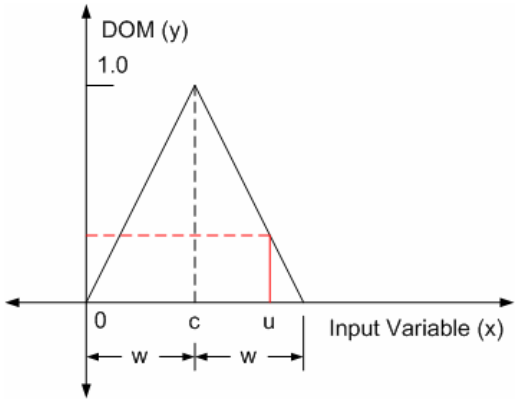

Figure 4.4 – Section of triangular membership function

For $u = c$, the degree of membership is equal to one. For $u < c$ and $u > c$, the gradient of each line is expressed as $m = \pm\frac{1}{w}$. A transformation to the origin may be performed by finding the absolute difference between the input value and the centre of the triangle. The degree of membership may then be found by using the equation for a linear function, $y = mx$. Figure 4.5 shows the transformed triangle.

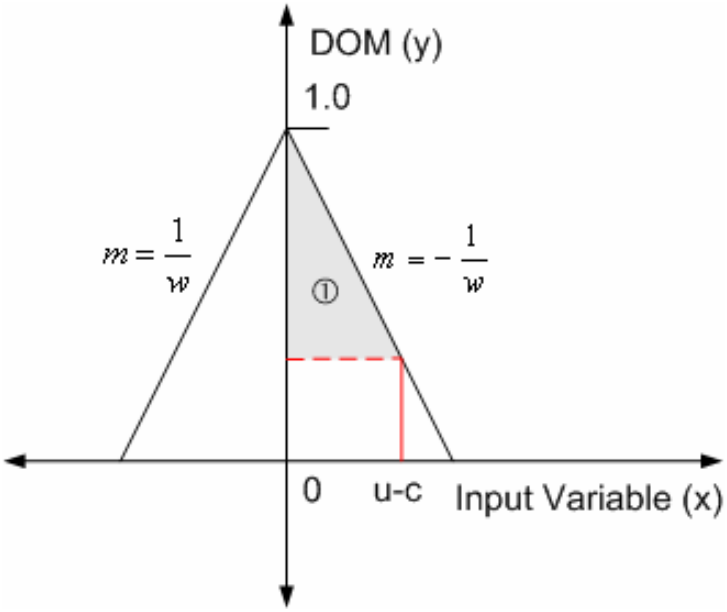

Figure 4.5 – Transformed triangle

The height of the inner triangle ① is given by $y = mx = \frac{u-c}{w}$. To calculate the degree of membership, this value is subtracted from one, giving the following equations for the degree of membership:

$$DOM = \begin{cases} 1 - \frac{u-c}{w} & \text{if } u > c, \text{or} \\ 1 - \frac{c-u}{w} & \text{if } u < c \end{cases} \qquad (3)$$

The shape of the membership function for the two extreme fuzzy variables requires a special case. Figure 4.6 shows the shape of the membership function for the most negative and most positive fuzzy variables.

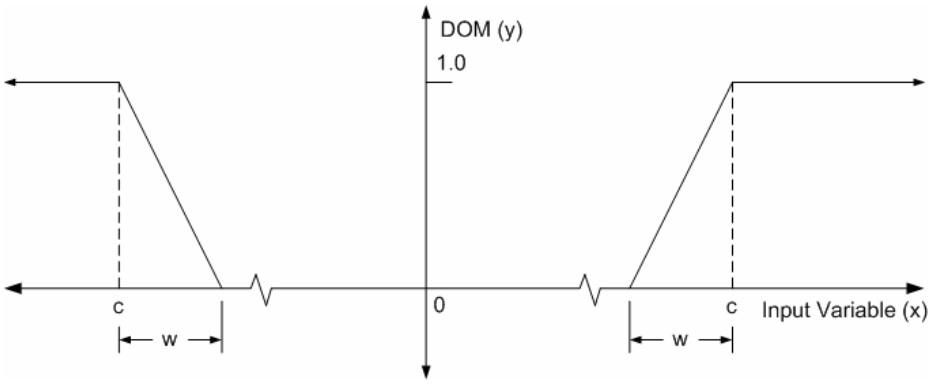

Figure 4.6 – Shape of membership function at extremes

The following Matlab code calculates the degree of membership to the most negative and most positive fuzzy variables. The functions are based on the function presented above. If the input value is less than or equal to, or, greater than or equal to the centre point, respectively, a value of one is returned. In other cases, the degree of membership is calculated according to the equation described above.

The higher level section of the fuzzification layer of the algorithm deals with iterating through each fuzzy variable. For input values, the extreme fuzzy variables are fuzzified using the special case as described, while the central fuzzy variables are fuzzified according to equation (3). Fuzzification of the phase value requires a further special case. The extreme fuzzy variables are combined to facilitate the 'wrap around' effect required.

4.1.5. Fuzzification of the Phase Variable

The phase variable allows each point in the cyclic motion to be uniquely identified. This allows the FAM algorithm to provide a unique compensation signal for each point in the motion. The fuzzification of the phase input variable requires careful consideration. As Figure 4.7 shows, the phase signal is a saw-tooth shaped waveform, looping instantaneously from the maximum value to the minimum value at the end of each cycle.

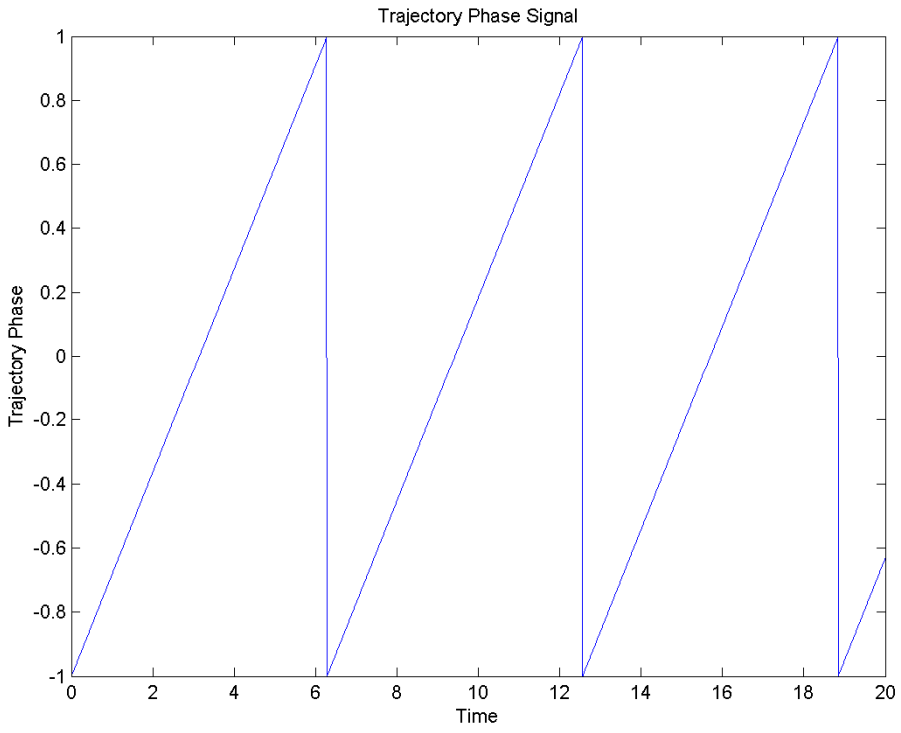

Figure 4.7 – Phase signal

When fuzzifying the phase signal to fuzzy variables, it can be seen that the large positive fuzzy variable would be related to the large negative fuzzy variable. The two extreme fuzzy variables should be represented by a single fuzzy variable. The shape of the membership function must effectively 'wrap around'. It may be visualised by considering the membership function being described in a circle dimension rather than a planar dimension.

4.1.6. Calculation of the Compensation Value

The compensation value may be calculated by performing matrix multiplication of the arrays containing the degree of membership for each input variable. The resulting matrix is then dot multiplied with the lookup table. The summation of the elements in the resulting matrix gives the compensation value. As an example, consider the following system. Let the fuzzy set for the input variable be $\{0 \quad 0.5 \quad 0.5 \quad 0 \quad 0\}$, and the fuzzy set for the phase variable be $\{0 \quad 0 \quad 0.25 \quad 0.75 \quad 0\}$. The matrix multiplication of these two arrays would be:

$$\begin{bmatrix} 0.00 \\ 0.50 \\ 0.50 \\ 0.00 \\ 0.00 \end{bmatrix} \times \begin{bmatrix} 0.00 & 0.00 & 0.25 & 0.75 & 0.00 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.125 & 0.375 & 0 \\ 0 & 0 & 0.125 & 0.375 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let the lookup table be $\begin{bmatrix} 4.8 & 1.0 & 1.3 & 1.5 & 2.5 \\ 2.5 & 1.0 & 1.4 & 1.6 & 0.1 \\ -2.5 & -1.6 & -1.4 & -1.0 & -4.7 \\ -0.2 & -1.4 & -1.4 & -1.0 & -2.3 \\ 2.5 & 1.0 & 1.4 & 1.0 & 0.1 \end{bmatrix}$.

Then, the dot multiplication would be:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.125 & 0.375 & 0 \\ 0 & 0 & 0.125 & 0.375 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 4.8 & 1.0 & 1.3 & 1.5 & 2.5 \\ 2.5 & 1.0 & 1.4 & 1.6 & 0.1 \\ -2.5 & -1.6 & -1.4 & -1.0 & -4.7 \\ -0.2 & -1.4 & -1.4 & -1.0 & -2.3 \\ 2.5 & 1.0 & 1.4 & 1.0 & 0.1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.175 & 0.600 & 0 \\ 0 & 0 & -0.175 & 0.375 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The compensation value is the sum of the elements in the resulting matrix, that is,

$correction = 0.175 + (-0.175) + 0.600 + 0.375 = 0.975$.

These commands may be completed in one line of Matlab code, as follows.

```
correction = sum(sum( (phase_mem.dom' * x_mem.dom) .*
                          RULE_TABLE ));
```

The update for the lookup table may be performed in a similar way to calculating the compensation value. The elements of the fuzzy sets which are non-zero are found using the Matlab function `find`. The binary arrays are then matrix multiplied and scaled according to the error and learning rate. This matrix is then added to the lookup table. For example, consider the parameters defined above, let the error be 0.8 and the learning rate be 0.05. The update matrix would be:

$$\begin{bmatrix}0\\1\\1\\0\\0\end{bmatrix}\times\begin{bmatrix}0 & 0 & 1 & 1 & 0.00\end{bmatrix}\times 0.8\times 0.05=\begin{bmatrix}0 & 0 & 0 & 0 & 0\\0 & 0 & 0.04 & 0.04 & 0\\0 & 0 & 0.04 & 0.04 & 0\\0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0\end{bmatrix}$$

The lookup table after this iteration of the control loop would then be:

$$\begin{bmatrix}4.8 & 1.0 & 1.3 & 1.5 & 2.5\\2.5 & 1.0 & 1.4 & 1.6 & 0.1\\-2.5 & -1.6 & -1.4 & -1.0 & -4.7\\-0.2 & -1.4 & -1.4 & -1.0 & -2.3\\2.5 & 1.0 & 1.4 & 1.0 & 0.1\end{bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0 & 0\\0 & 0 & 0.04 & 0.04 & 0\\0 & 0 & 0.04 & 0.04 & 0\\0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}4.8 & 1.0 & 1.3 & 1.5 & 2.5\\2.5 & 1.0 & 1.44 & 1.64 & 0.1\\-2.5 & -1.6 & -1.44 & -1.04 & -4.7\\-0.2 & -1.4 & -1.4 & -1.0 & -2.3\\2.5 & 1.0 & 1.4 & 1.0 & 0.1\end{bmatrix}$$

4.1.7. Stability

The stability of the FAM compensated control system depends on the shape of the membership function. If a disturbance occurs in the system, the membership function should be able to differ between large and small disturbances. If the membership function is incorrectly shaped, that is, covers an insufficient range of input values, the same compensation signal may be given for significantly different sized disturbances, causing instabilities. A large disturbance may cause the lookup table values to become significantly large, while a smaller disturbance may be compensated by the same signal. A large learning rate will amplify this problem. In general, a wide membership function with many triangles will provide the greatest stability. This added stability however will come at the cost of more computer memory and more processor computations per control loop.

4.1.8. Testing and Results of the Matlab Model

The Matlab model was extensively tested, with many different input waveforms, and many different FAM parameters. The first test was to use a step input, and compare the results of the FAM compensated system with the results from the stand alone PD controlled system. The learning rate and the number of triangles in the membership functions were the two parameters that were adjusted.

The main purpose of experiments with the step input is to have a basis for finding out the effects of adjusting the learning rate and the shape of the membership function. The model was run for a period of 40 seconds. To ensure the phase signal had no effect on the system, the period of the phase signal was set to 500 seconds.

It is important to remember that when the number of triangles in the membership function changes, the width of the triangles also changes. The membership function always covered the same input values for consistent testing. This means that the centre of the end triangles should always be at the same input value. For this test, the value was chosen to be 1.5. Figure 4.8 shows this graphically. With a step input, because the motion is not cyclic, the phase is redundant.

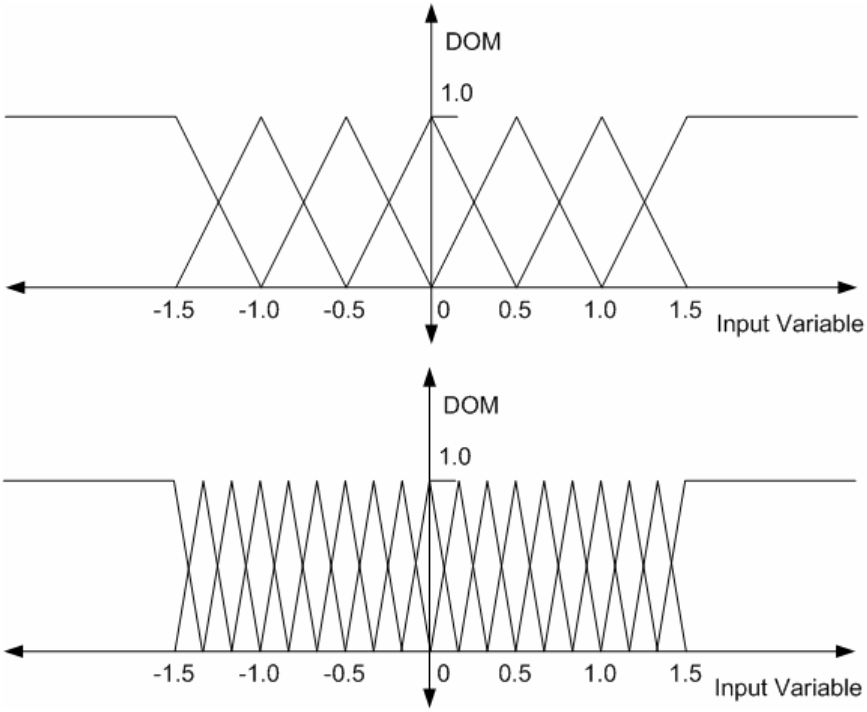

Figure 4.8 – Alternate shape of membership functions

Step Input Experiment – Variations in membership function

The first set of experiments was completed with a fixed learning rate of 0.01. The number of triangles was varied from 5, 11, 21, 51, 101, and 201. Figure 4.9 through Figure 4.14 shown below display the response of the cart in each of the different circumstances. The red line is the desired step response of the cart. The blue line shows the response of the uncompensated PD control loop, while the green line depicts the response of the FAM compensated controller.

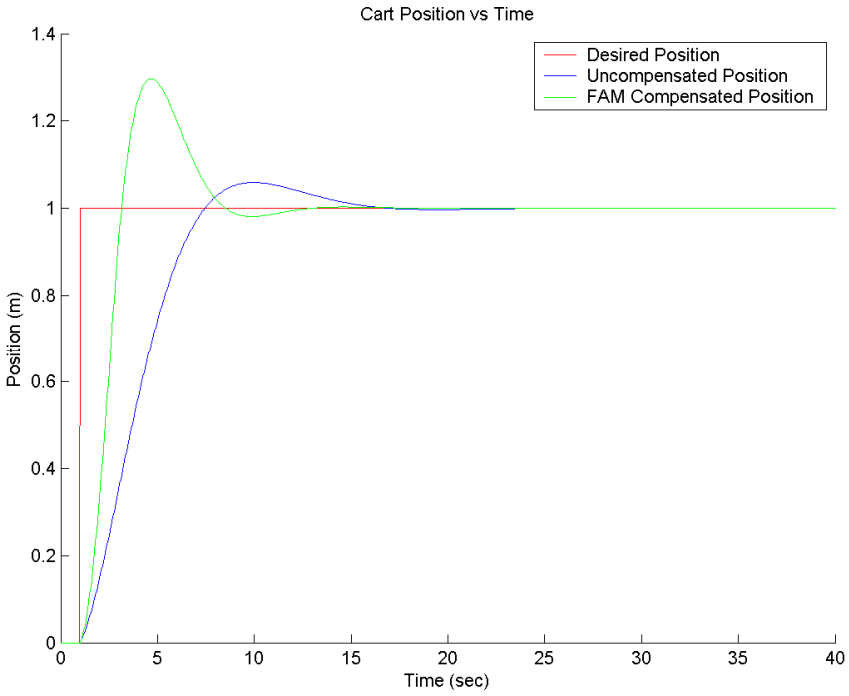

Figure 4.9 – Step response, 5 triangles in membership function

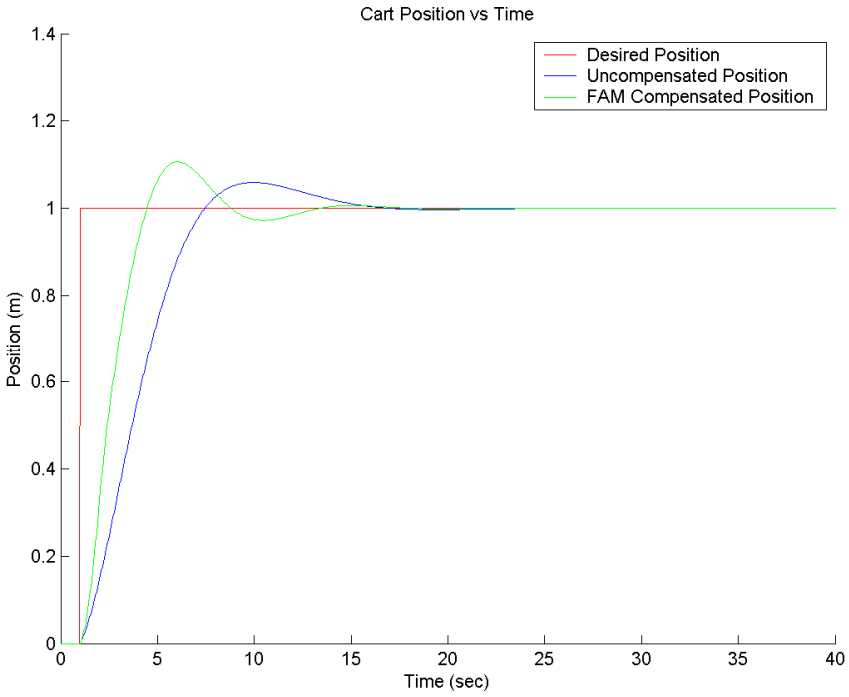

Figure 4.10 – Step response, 11 triangles in membership function

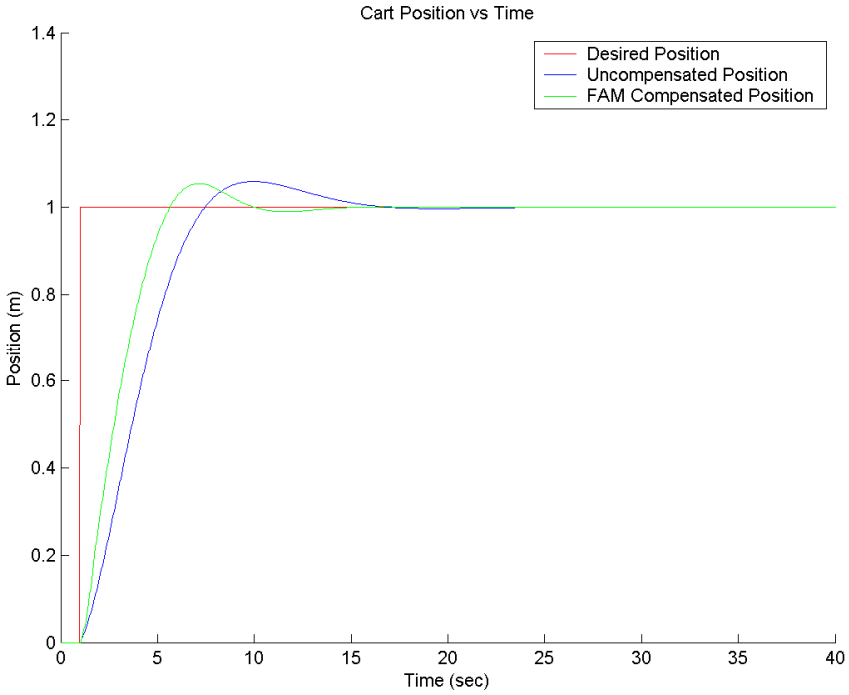

Figure 4.11 – Step response, 21 triangles in membership function

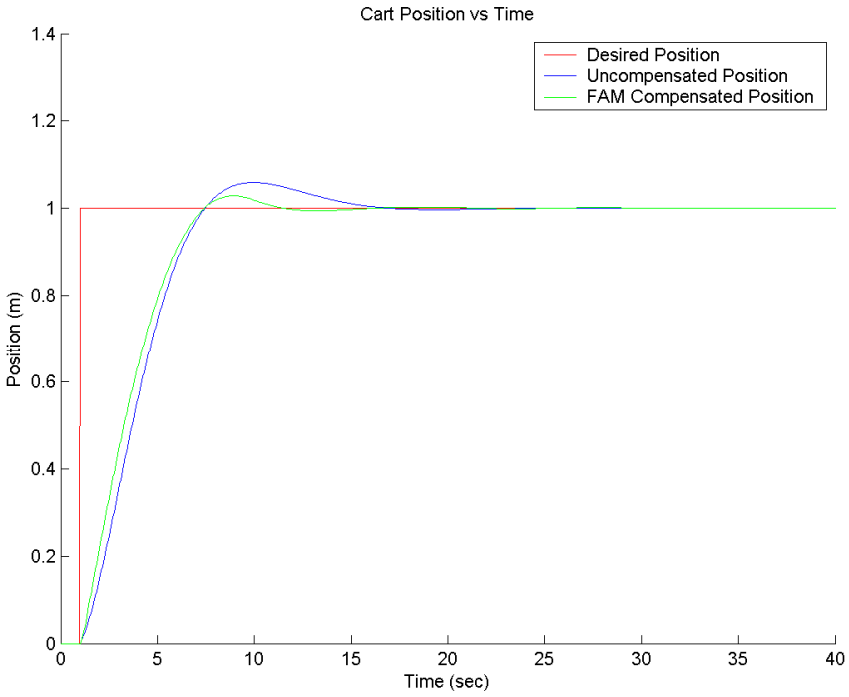

Figure 4.12 – Step response, 51 triangles in membership function

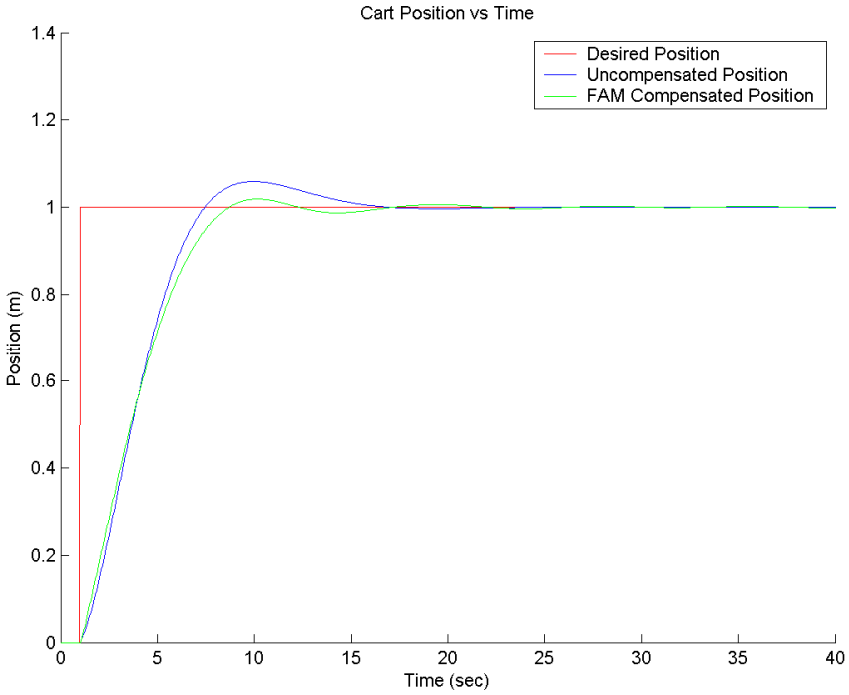

Figure 4.13 – Step response, 101 triangles in membership function

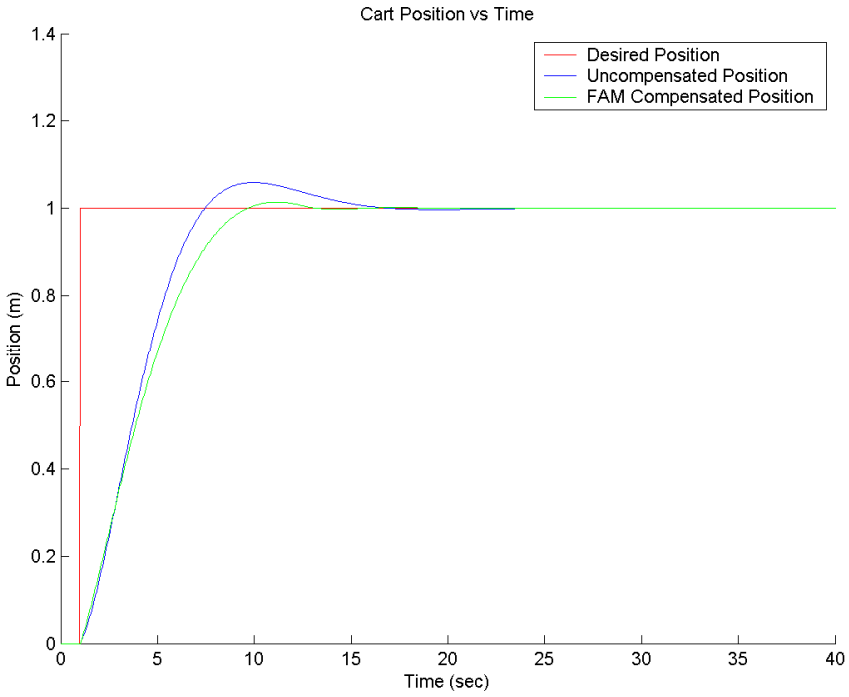

Figure 4.14 – Step response, 201 triangles in membership function

The experiment showed that a small number of triangles in the membership function gave a faster response, at the compromise of larger overshoot. As the number of triangles becomes larger, the overshoot is reduced, at the cost of a quicker response. Table 4.1 shows the response time and percent overshoot for each of the tests.

Table 4.1 – Response time and percent overshoot results

Number triangles	Response Time (sec)	Percent Overshoot
5	7.5	30%
11	7.7	10%
21	7.6	5%
51	6.5	3%
101	7.5	2%
201	8.2	1.5%

Step Input Experiment – Variations in learning rate

The second experiment was to observe the effect of adjusting the learning rate. Figure 4.15 through Figure 4.20 below, show a series of step responses with 21 triangles in the membership, and learning rates of 0.01, 0.025, 0.5, 0.1, 0.25 and 0.5 respectively.

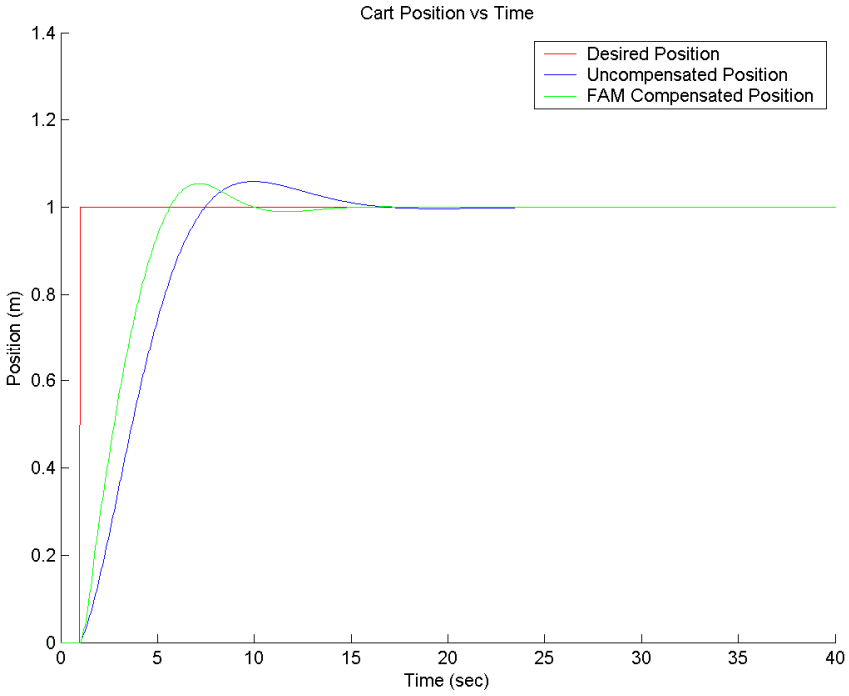

Figure 4.15 – Step response, learning rate: 0.01

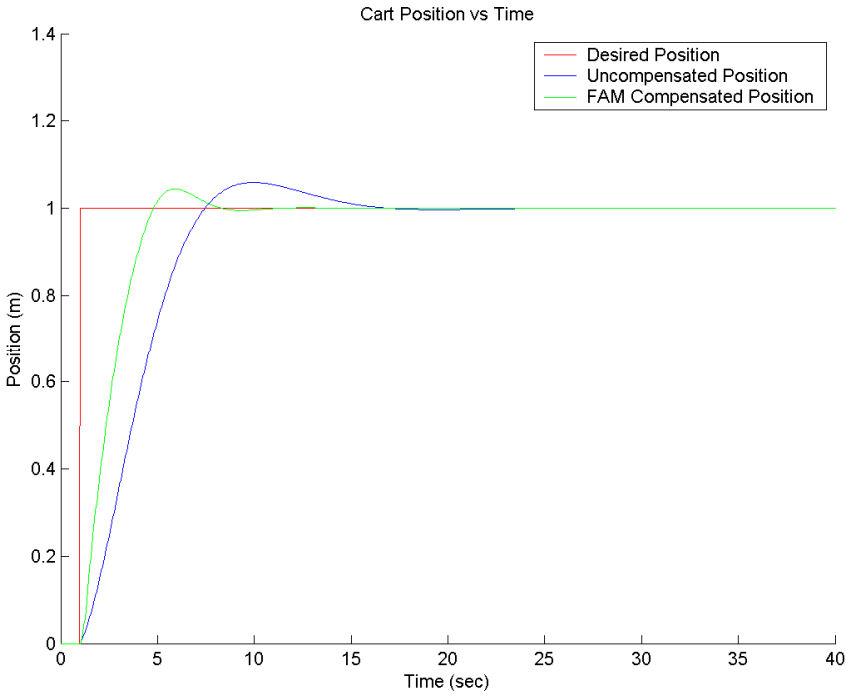

Figure 4.16 – Step response, learning rate: 0.025

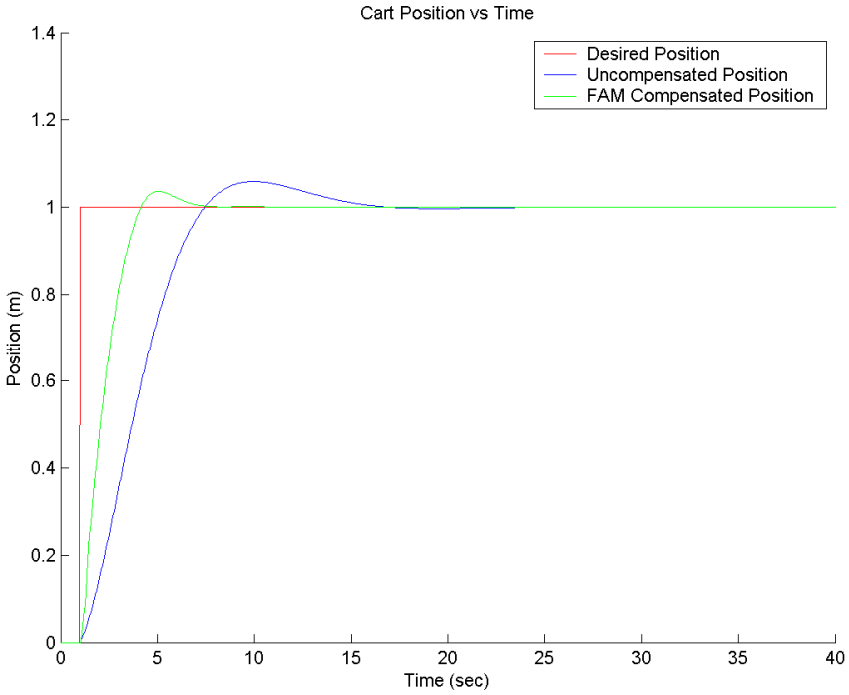

Figure 4.17 – Step response, learning rate: 0.05

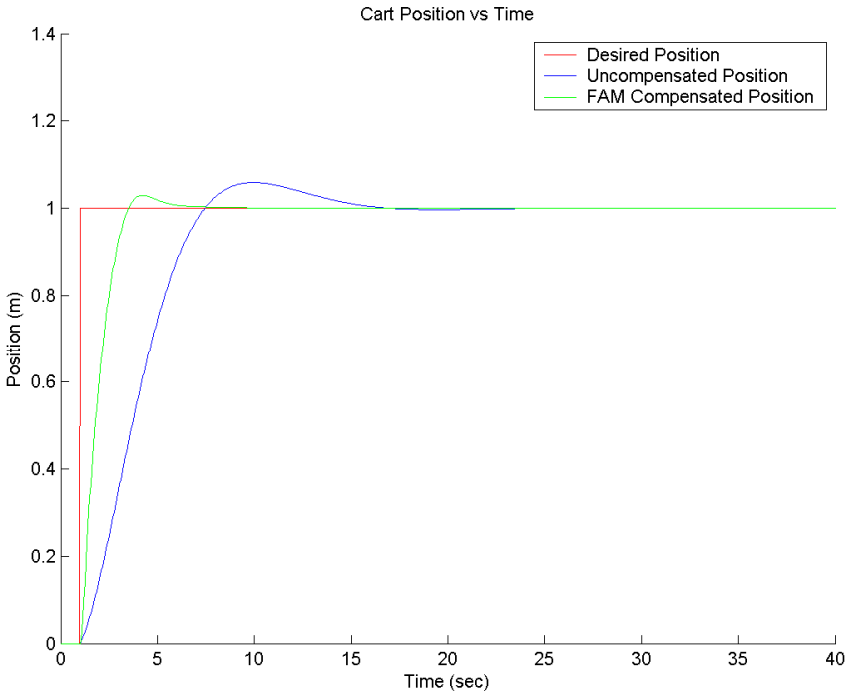

Figure 4.18 – Step response, learning rate: 0.1

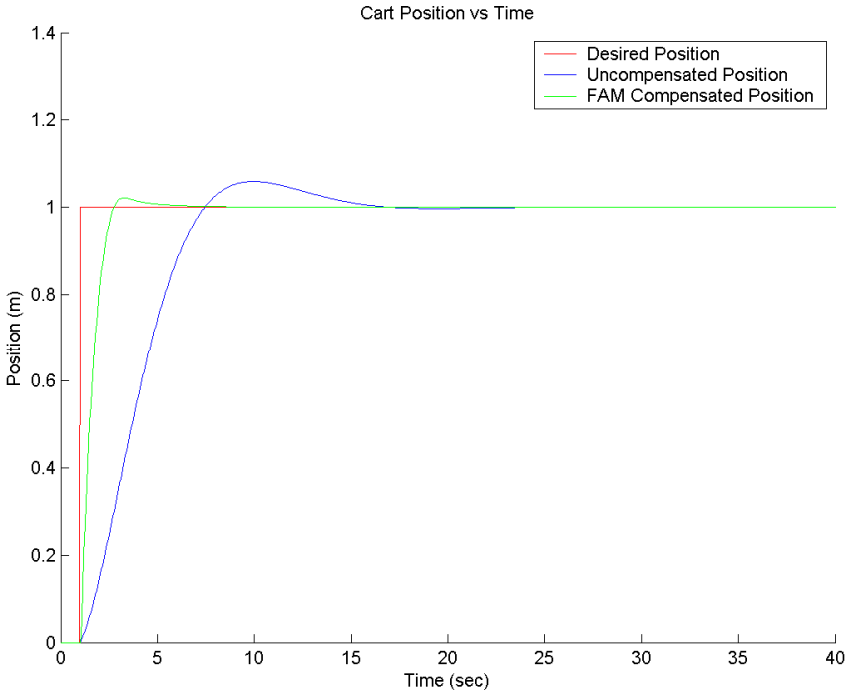

Figure 4.19 – Step response, learning rate: 0.25

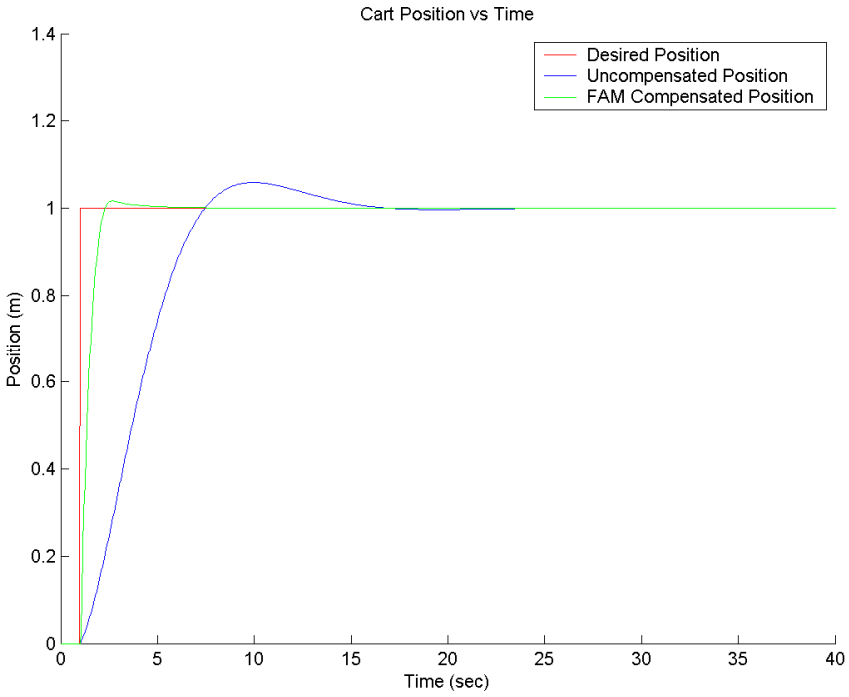

Figure 4.20 – Step response, learning rate: 0.5

This experiment showed that an increase in the learning rate improves the response time of the control system without increasing overshoot. Table 4.2 shows the response time and percent overshoot for each of the tests.

Table 4.2 – Response time and percent overshoot results

Learning Rate	Response Time (sec)	Percent Overshoot
0.01	7.6	5%
0.025	4.3	4.5%
0.05	3.7	4%
0.1	3.1	3%
0.25	2.4	2%
0.5	2.0	1.5%

Cart Experiment – Variations of input waveforms

A series of experiments was completed with different shaped waveforms, including sinusoidal, square, triangular, and trapezoidal waveforms. Figure 4.21 through Figure 4.24 below, show the desired, uncompensated PD controlled, and FAM compensated position of the cart for each of the tested waveforms. The FAM parameters used in each of the experiments is shown in Table 4.3.

Table 4.3 – Parameters for waveforms experiment

FAM Parameter	**Value**
Number of triangles in membership function	21
Learning rate	0.05

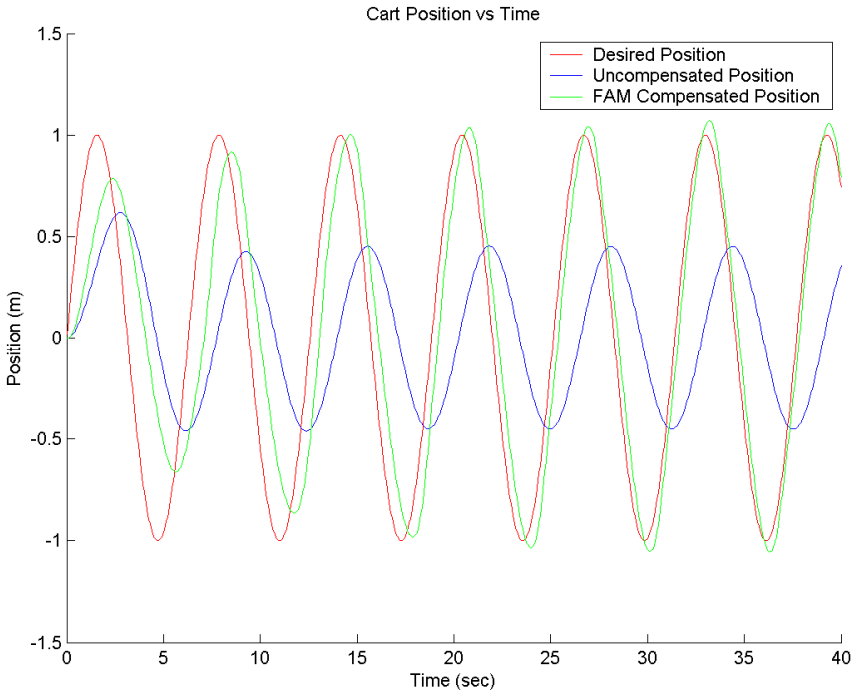

Figure 4.21 – Cart position: sinusoidal input

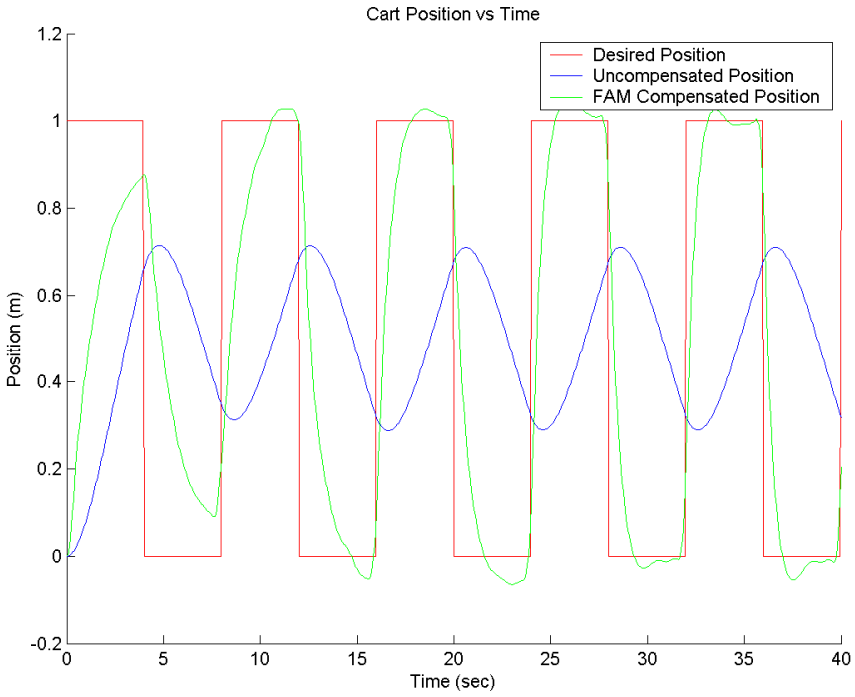

Figure 4.22 – Cart position: square wave input

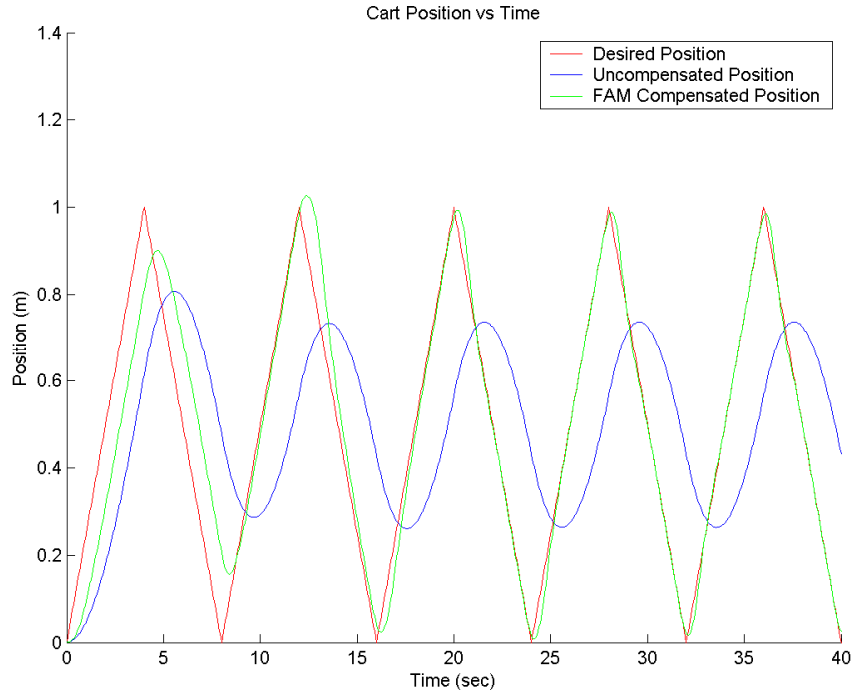

Figure 4.23 – Cart position: triangular input

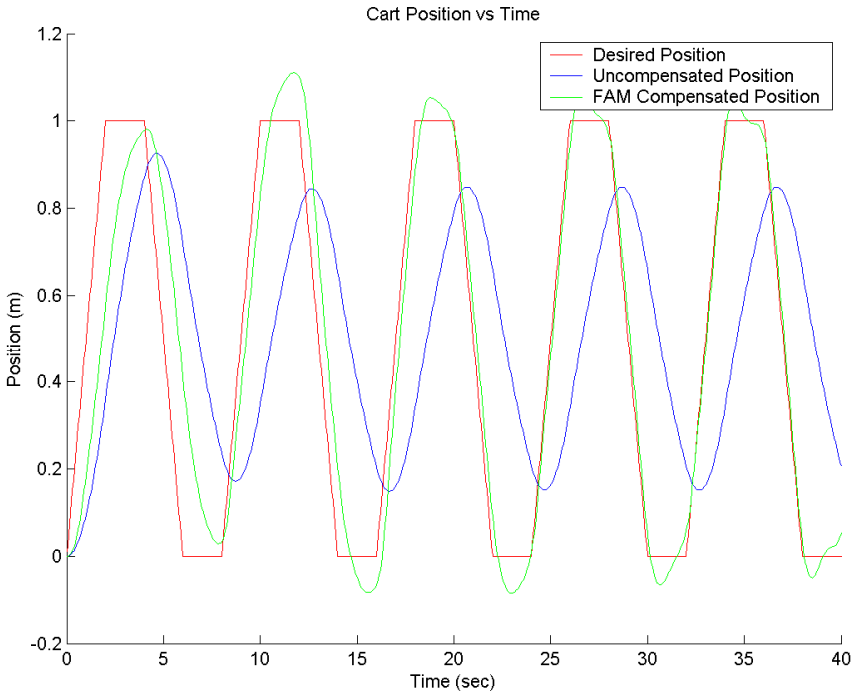

Figure 4.24 – Cart position: trapezoidal input

As can be seen, the algorithm is able to significantly improve trajectory tracking for each of these waveforms. Careful inspection of Figure 4.22 and Figure 4.24 reveals the predictive nature of the FAM algorithm. Immediately before the error becomes large within a cycle, the FAM position under-compensates in an effort to minimise the large error.

4.2. PORTING THE MODEL TO C

The model was ported to C for further implementation on other platforms. The model dynamics were converted to difference equations for implementation in C.

4.2.1. C Implementation of Cart Model

The control diagram below shows the states of the model.

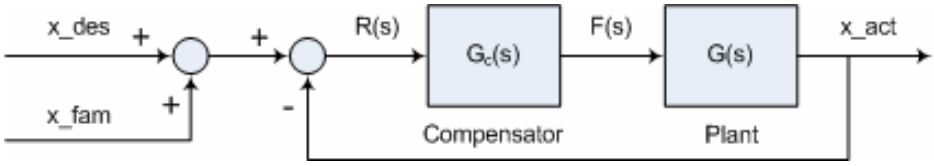

Figure 4.25 – Control diagram of cart model

From Figure 4.25, the time-domain signal to the compensator is defined by the equation below.

$$r(t) = \left(x\_des + x\_fam\right) - x\_act$$

The derivation of the time domain equation for the compensator is shown in the equation below.

$$\begin{aligned} & G_C(s) = 2 \times \left(2s + 1\right) \\ & \Rightarrow F(s) = G_C(s) R(s) \\ & \Rightarrow f(t) = 2 \times \left(r(t) + 2 \times \dot{r}(t)\right) \end{aligned}$$

Using a difference equation, the derivative of $r(t)$ is defined as:

$$\dot{r}(t) = \frac{r(t) - r(t-1)}{STEP\_SIZE}$$

From the equations of motion, the position of the cart is derived from the force, as shown in the equation below.

$$
\begin{aligned}
& F = m\ddot{x} + D\dot{x} \\
& \Rightarrow \ddot{x}(t) = \frac{f(t) - D\dot{x}(t-1)}{m} \\
& \dot{x}(t) = \ddot{x}(t-1) \times STEP\_SIZE + \dot{x}(t-1) \\
& x(t) = \dot{x}(t-1) \times STEP\_SIZE + x(t-1)
\end{aligned}
$$

These equations were implemented in a `for-loop` that iterated through time, according to the simulation time and the step size. The model was written in a separate C module to the FAM algorithm, to increase portability.

4.2.2. C Implementation of FAM algorithm

The Fuzzy Associative Memory algorithm was ported directly from Matlab M code to a C module, with little more than syntactical changes. The same implementation of data structures was used, while the matrix multiplication and dot multiplication was manually calculated, implemented in a double `for-loop`.

4.2.3. Testing and Results of the C Model

The C model produced the desired results, with a similar improvement in trajectory tracking to the Matlab model. The graph in Figure 4.26 shows the comparison between the results from the Matlab model and the results from the C model for a square shaped waveform. The difference between the C model and the Matlab model may be caused by the approximations in the implementation of the plant and compensator in the time domain. The behaviour of the C model also changes with the step size. The chosen step size of 0.01 seconds gives reasonable accuracy, and gives a very short simulation time.

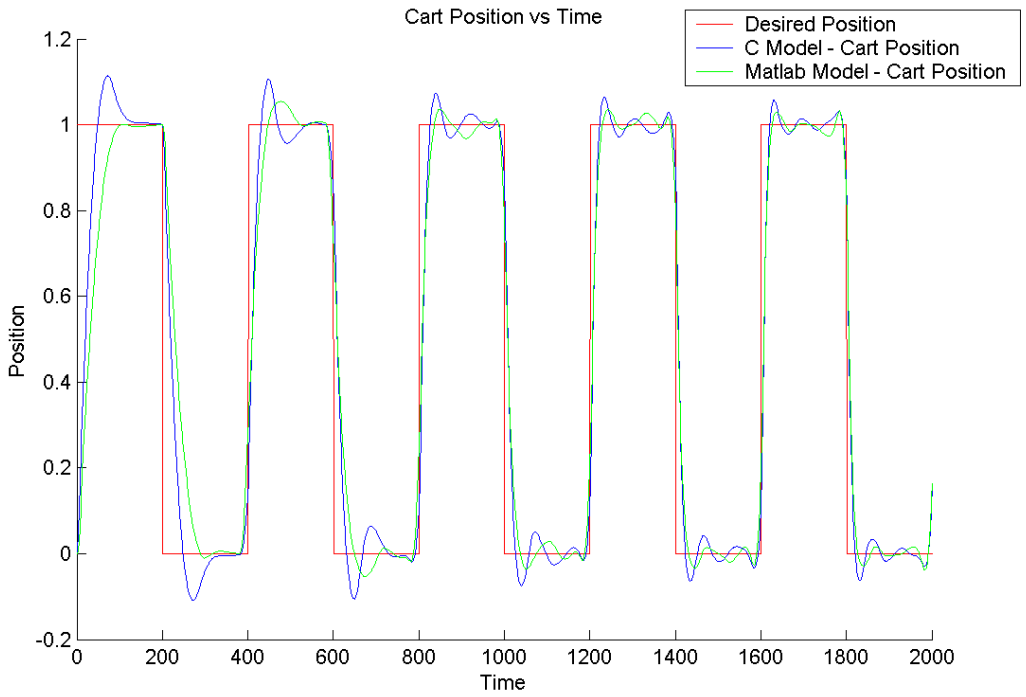

Figure 4.26 – Cart position of C model versus cart position of Matlab model

4.3. FIXED-POINT IMPLEMENTATION

To make the system portable to a number of operating environments, the FAM algorithm had to be converted from floating-point arithmetic, to a fixed-point arithmetic implementation. This change would remove the requirement of having a floating-point capable processor, and remove the reliance of the `math` library, from the Windows API. By using a fixed-point arithmetic implementation, the operating time for the algorithm is expected to decrease, giving better overall performance.

4.3.1. Changes to FAM algorithm

To convert the system to fixed-point arithmetic, all variables in the algorithm were converted to base two. This included the number and width of the triangles in the membership function, the learning rate, the degree of membership of an input value to a fuzzy variable, and the input and output values of the algorithm. The number of triangles is required to be an odd number, so the number of triangles is a number of base two, plus one. The learning rate is defined as an integer, with the error in the system right-shifted by that integer value, which is equivalent to dividing by two to the power of that number.

The largest modifications to the FAM algorithm occurred in fuzzification stage of the algorithm, and in the calculation of the compensation signal. With the fixed-point arithmetic, the height of the triangles in the membership function changes from a fractional value in between and inclusive of zero and one, to an integer value between zero and a number of the power of two. A larger number provides greater resolution, and hence greater accuracy of the algorithm, however at the risk of overflow. Initially, a triangle height of 256 was chosen, which gave overflow. A triangle height of 64 gave adequate accuracy without overflow.

As shown above, the calculation of the compensation signal involves multiplying the degree of memberships of the separate input values. When using fixed-point arithmetic, the integer values may be multiplied, and then multiplied by the corresponding value in the lookup table. It is of utmost importance that this number does not overflow, as the compensation signal would become incorrect, and the algorithm would fail. This number is then right-shifted by twice the height of the triangles.

4.3.2. Preventing Overflow in the Lookup Table

It is important also to limit the values in the lookup table to prevent overflow. Although this is also a problem with the floating-point arithmetic implementation, it is more apparent in the fixed-point arithmetic implementation as the numbers are larger, and are more likely to overflow. If overflow occurs within the lookup table, the compensation signal is corrupted. To prevent overflow, the integers within the lookup table are limited to upper and lower bounds after they are updated.

4.3.3. Output Limiting of Compensation Signal

For the C model to be more realistic, output limiting was implemented on the compensation signal. Because of the nature of the FAM system, unachievable input signals may be produced by the controller in an attempt improve trajectory tracking. In a physical model, this may involve providing a larger than allowable signal to an actuator, possibly causing physical damage to the actuator. This is implemented by limiting the output value from the FAM algorithm to upper and lower bounds.

4.3.4. Error plots

A plot of the error in position gives an alternative view of the improvement in trajectory tracking. Figure 4.27 shows the error for the cart tracking a sinusoidal path. The error plot shows the convergence of the error to zero, with a waveform that is symmetric about the zero axis. Error plots will be used to analyse the performance of the FAM algorithm throughout the following sections of this thesis.

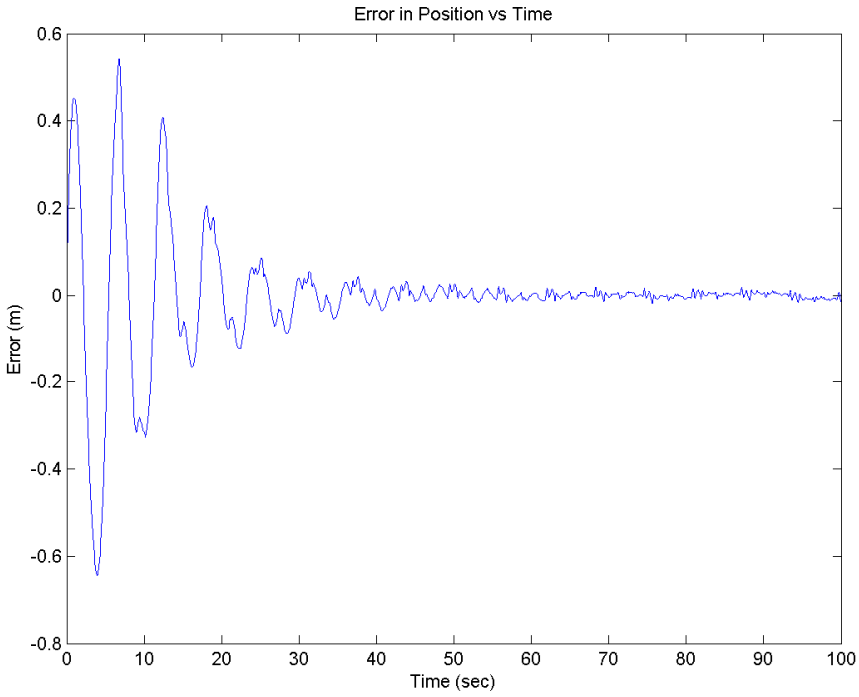

Figure 4.27 – Error in position versus time

4.3.5. Testing and Results of Fixed-Point Implementation

The accuracy of the fixed-point arithmetic implementation was tested, using a series of different shaped waveforms in comparison against the floating-point implementation. The effect of output limiting the compensation signal was also tested.

Fixed-Point Arithmetic Experiment – Comparison to Floating-Point

To check the accuracy of the fixed-point arithmetic implementation, a series of waveforms were tested for the floating and fixed-point arithmetic implementations, and the results compared. The number of triangles in the membership function in both models was 17, while the learning rate was 0.0625. Figure 4.28 through Figure 4.30 below, show the difference in position between the floating and fixed-point arithmetic implementations for a series of waveforms.

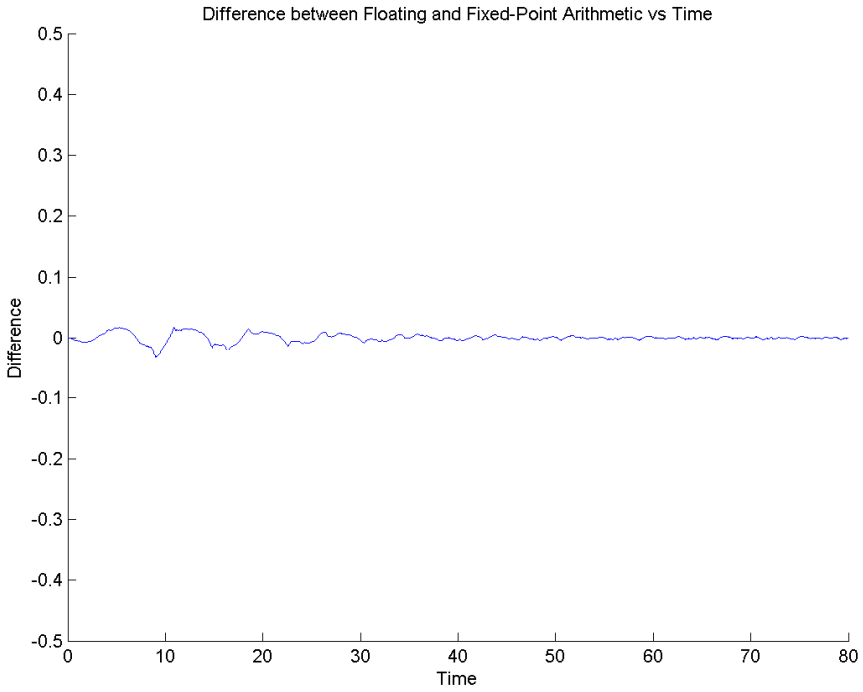

Figure 4.28 – Floating versus fixed-point arithmetic – sinusoidal wave

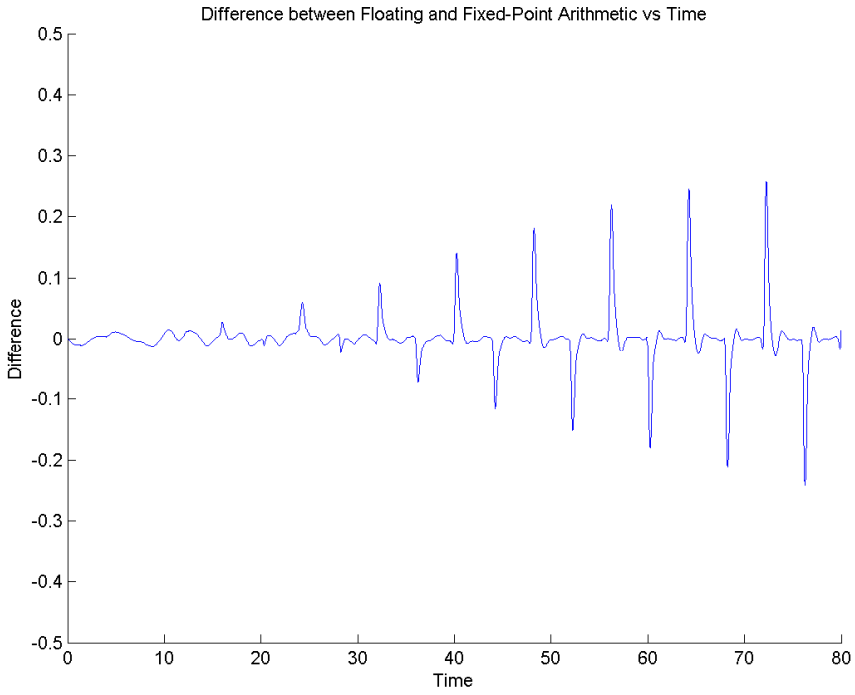

Figure 4.29 – Floating versus fixed-point arithmetic – square wave

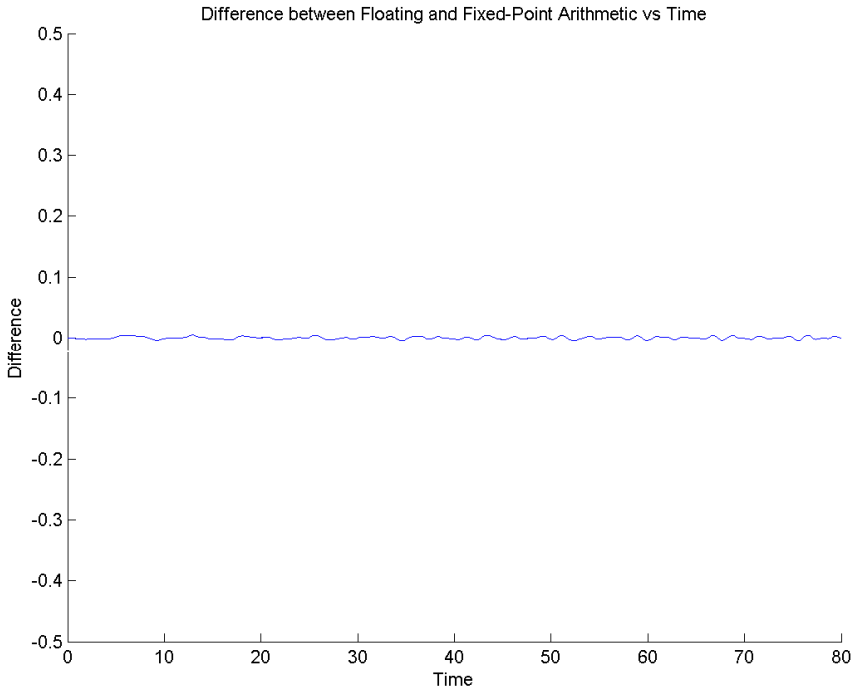

Figure 4.30 – Floating versus fixed-point arithmetic – triangular wave

The results show that the fixed-point arithmetic implementation is accurate when compared to the floating-point arithmetic implementation. A significant difference is revealed when a square input is used, caused by rounding errors in the large compensation signal.

Fixed-Point Arithmetic Experiment –Limiting Output Values

A series of experiments were performed to analyse the effect of limiting the compensation signal. The model and FAM parameters used in the experiment are shown in Table 4.4. Figure 4.31 through Figure 4.34 below, show a comparison of the experiment performed with and without limiting the compensation signal respectively.

Table 4.4 – Parameters for output limiting experiment

FAM Parameter	Value
Number of triangles in membership function	17
Learning rate	0.0625
Height of triangles	32
Lookup table value limit	±131072
Output limit on compensation signal	±2

As the figures below show, output limiting the compensation signal introduces a steady-state error to the system.

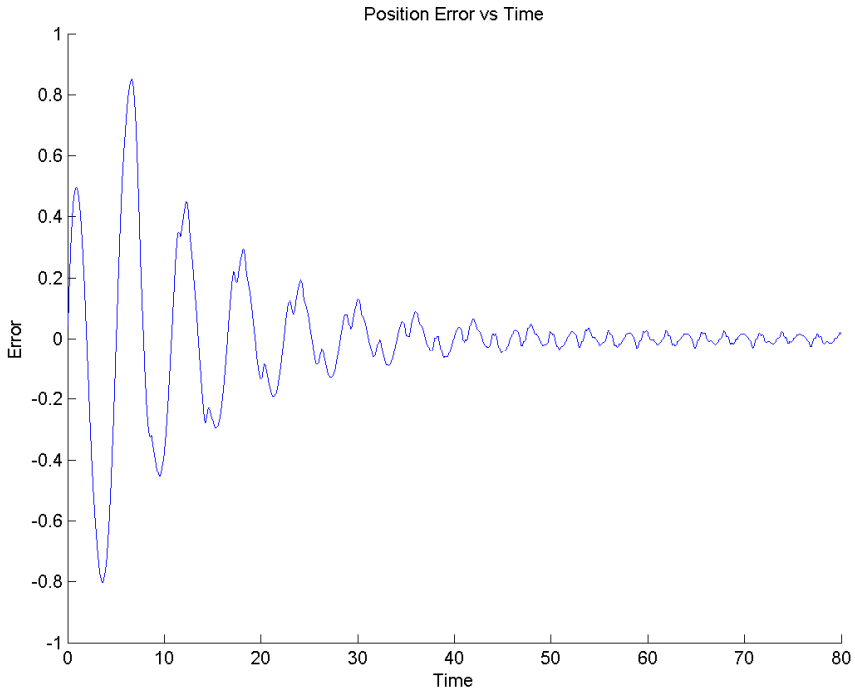

Figure 4.31 – Sine wave period six seconds

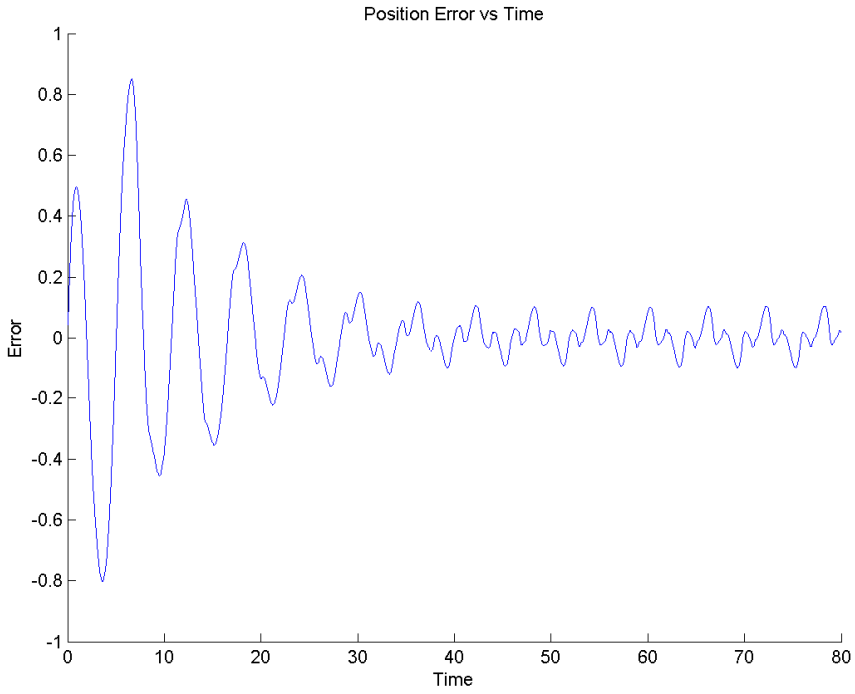

Figure 4.32 – Compensation limiting – Sine wave period six seconds

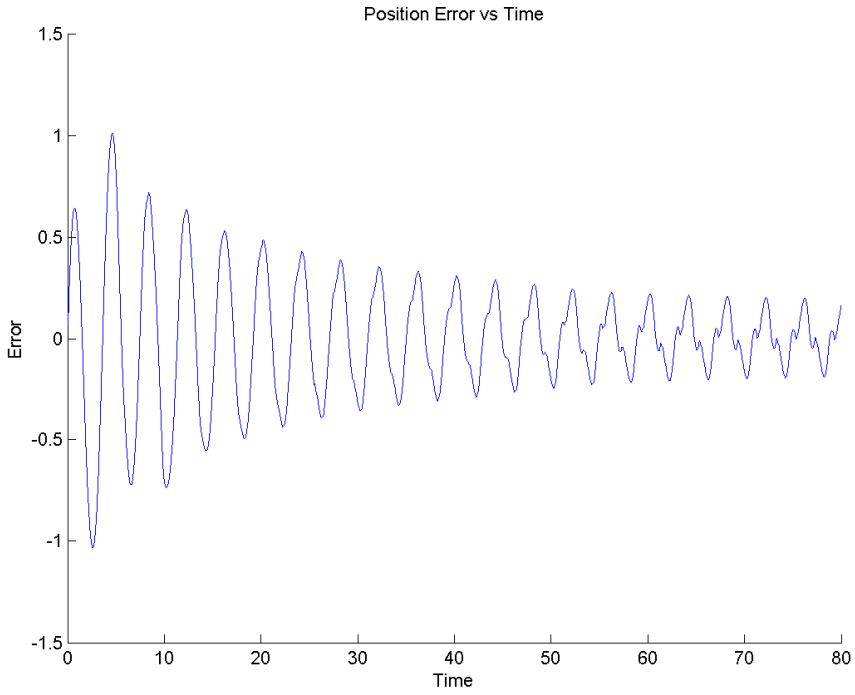

Figure 4.33 – Sine wave period four seconds

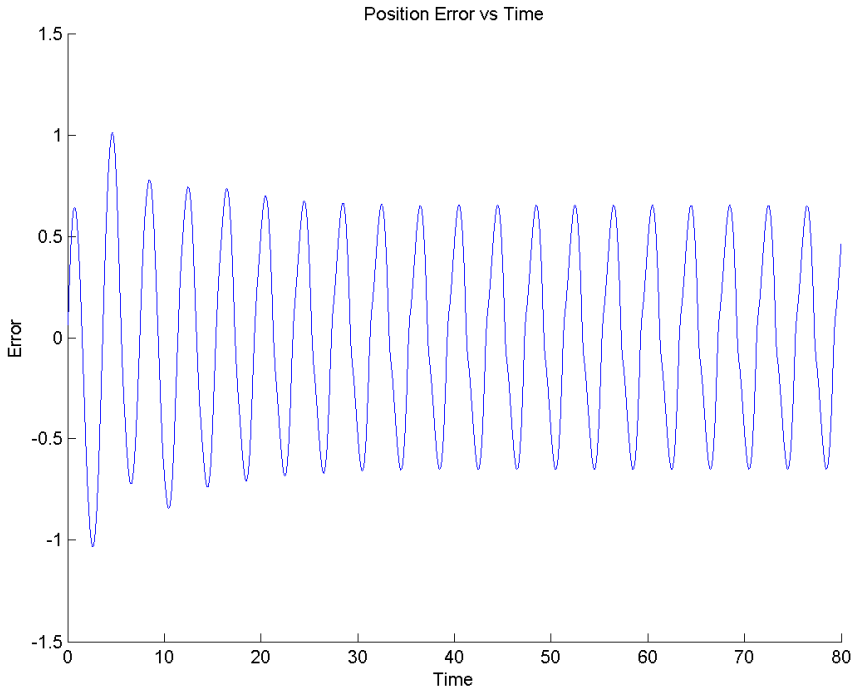

Figure 4.34 – Compensation limiting – Sine wave period four seconds

4.4. FINDINGS FROM CART MODEL

The cart model provided a suitable platform to develop and test the FAM algorithm, and showed that significant trajectory tracking improvement could be achieved. The cart model showed that the FAM algorithm could improve trajectory tracking for a variety of cyclic waveforms, from sinusoidal to square shaped waveforms. This indicates that the FAM algorithm is a robust algorithm, and it was expected that the algorithm would significantly improve trajectory tracking on the GuRoo robot.

The cart model gave a good feel for the behaviour of the algorithm, and the effects of adjusting the design parameters. It was found that decreasing the number of triangles in the membership function gave a faster and less stable response, while increasing the learning rate had a similar effect. The correct combination of these two critical design parameters would give the desired behaviour of reasonable response time with good stability. The cost of increasing the number of triangles in the membership function was in both computer memory and processor time. Although these problems are insignificant when used on a desktop workstation, the problems may become quite significant should the algorithm be implemented on microcontroller type systems with limited memory and processing power.

5. THE GUROO SIMULATOR

The GuRoo robot provides a real world project to test the capabilities of the FAM algorithm. The GuRoo robot is a complex dynamic system, with many degrees of freedom and less than perfect control loops. The GuRoo simulator, based on the Dynamechs package by McMillian *[McMillian, 1995]*, is a high fidelity dynamic simulation of the GuRoo robot. The motor characteristics of the Direct Current (DC) motors are modelled, including stiction, armature resistance and damping co-efficient *[Kee, 2003]*. The GuRoo simulator provides the platform to test the FAM algorithm. This chapter describes the application of the FAM algorithm to the GuRoo simulator and the results of the experiments performed on the simulator.

5.1. EXISTING JOINT CONTROL

The GuRoo has a finely tuned PI control loop in velocity, controlling each of the DC motors. This control loop corresponds to PD control in position. The Proportional and Integral constants were determined using a genetic algorithm, with a fitness function minimising trajectory error and maximising joint smoothness *[Roberts et al, 2003]*.

5.2. THE CROUCHING BEHAVIOUR

One of the simplest tasks the GuRoo can perform is the crouching behaviour. In this behaviour, only the pitch joints of the hip, knee and ankle are actuated, with position error of the order of one degree. Figure 5.1 shows the error in position for each of the described joints for a typical crouch over a period of five iterations. The crouching behaviour parameters are shown in Table 5.1.

Table 5.1 – Crouch behaviour parameters

Behaviour Parameter	Value
Crouch cycle time	4.0 seconds
Maximum pitch ankle joint angle	16 degrees
Maximum knee joint angle	35 degrees
Maximum pitch hip joint angle	22 degrees

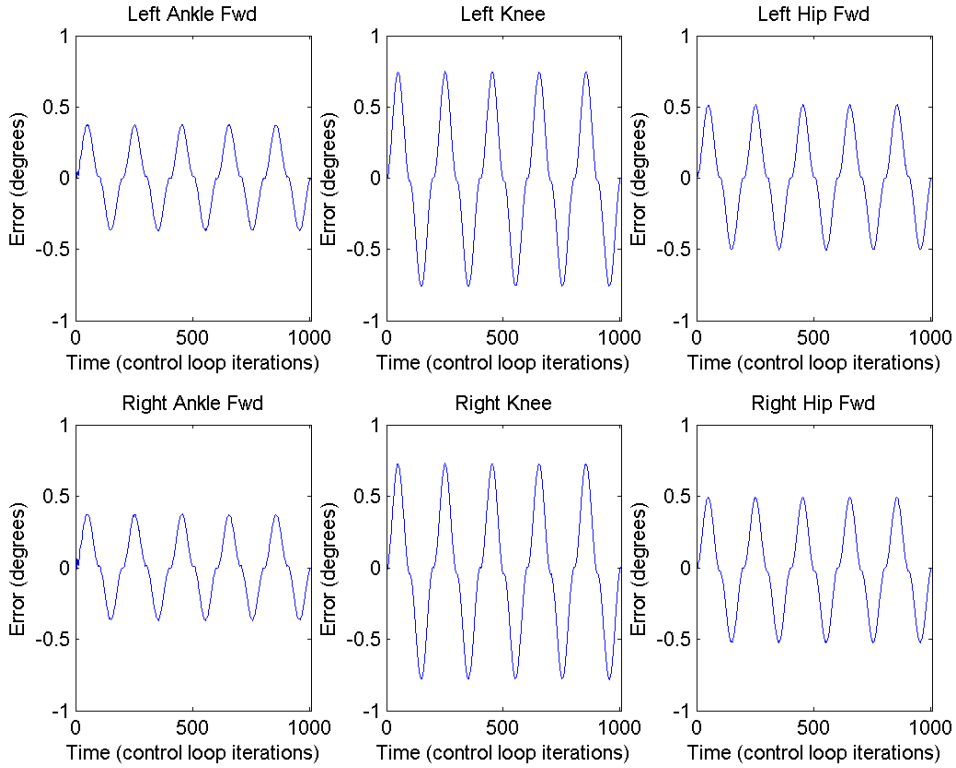

Figure 5.1 – Error in position for a crouching behaviour

The error plots show that even with a finely tuned PI control loop on each joint, a position error of the order of one degree exists for the crouching motion. A position error of such magnitude may be large enough to cause the robot to become unstable and over-balance.

5.3. FAM IMPLEMENTATION

The fixed-point arithmetic FAM module was ported directly from the cart model to the GuRoo simulator. The FAM algorithm was used to augment the desired velocity signal of the control loop. The input variables to the FAM algorithm are pre-scaled before they are fuzzified. Pre-scaling allows the real input values to match in the input space of the FAM algorithm. This pre-scaling allows for increased resolution when the input variables are fuzzified. The phase signal is scaled and quantised to give a value between -512 and +512.

5.4. EXPERIMENTS ON GUROO SIMULATOR

A series of experiments were performed on the GuRoo simulator. Many different inputs to the FAM algorithm were tested, with varying results. The inputs tested were:

- actual velocity,
- error in velocity,
- actual position,
- and error in position.

5.4.1. Fuzzification of Actual Velocity

The initial approach was to use fuzzify on the actual velocity of the joints. This was the intuitive approach, as the FAM algorithm provides a compensation signal to the desired position of the joints.

Actual Velocity Experiment – Single axis compensation

Single axis compensation was applied to the left knee joint for the crouching behaviour, with a significant improvement in the trajectory tracking of the left knee, and minor improvements in the trajectory tracking of the left ankle and the left hip joints as well. The FAM algorithm parameters that were used are shown in Table 5.2. The resulting trajectory error plot is shown in Figure 5.2.

Table 5.2 – FAM algorithm parameters

FAM Parameter	**Value**
Number of triangles in membership function	17
Learning rate	0.0625
Height of triangles	256
Centre of end triangles for input membership function	-65536, +65536
Lookup table value limit	±131072
Pre-scaler value for inputs	512
Output limit on compensation signal	±10

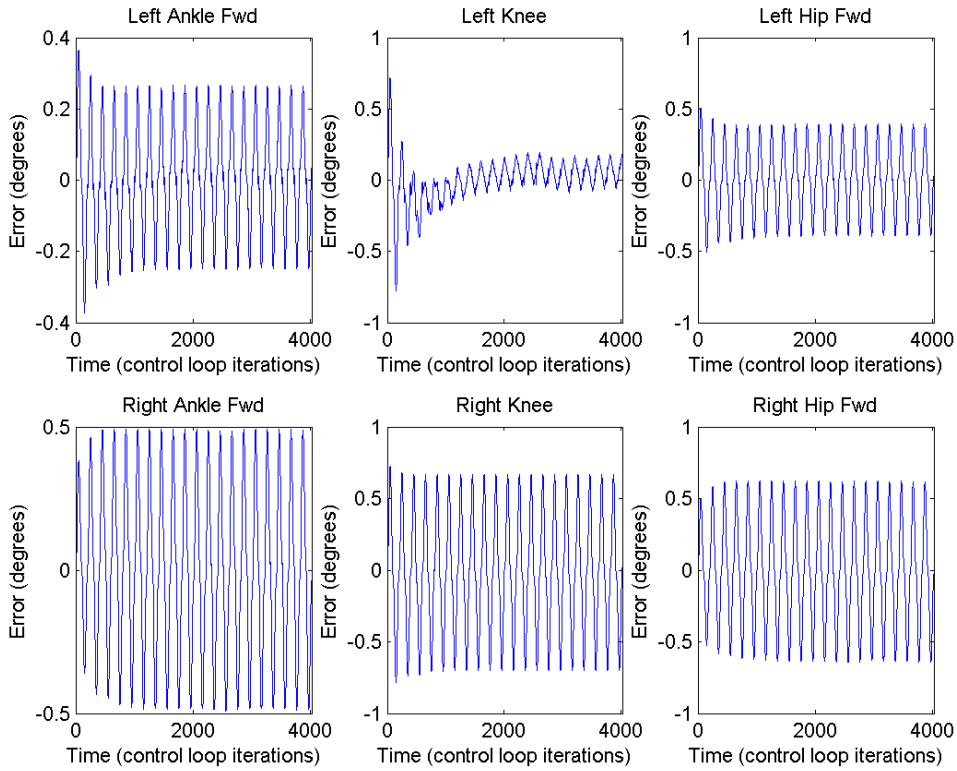

Figure 5.2 – Error in position, fuzzification on actual velocity – single-axis

As the error plot shows, the trajectory tracking has been significantly improved. A trajectory tracking improvement of at least 20% has been achieved. The learning time is about three crouching iterations.

Actual Velocity Experiment – Multiple axis compensation

The FAM algorithm was then expanded to compensate for all six joints in the crouching behaviour. Each joint has its own lookup table, and corresponding compensation signal, derived from the error in its own joint. Each joint used the same FAM parameters as shown above in Table 5.2. Figure 5.3 shows the results of FAM compensation for each of the joints in the crouching behaviour.

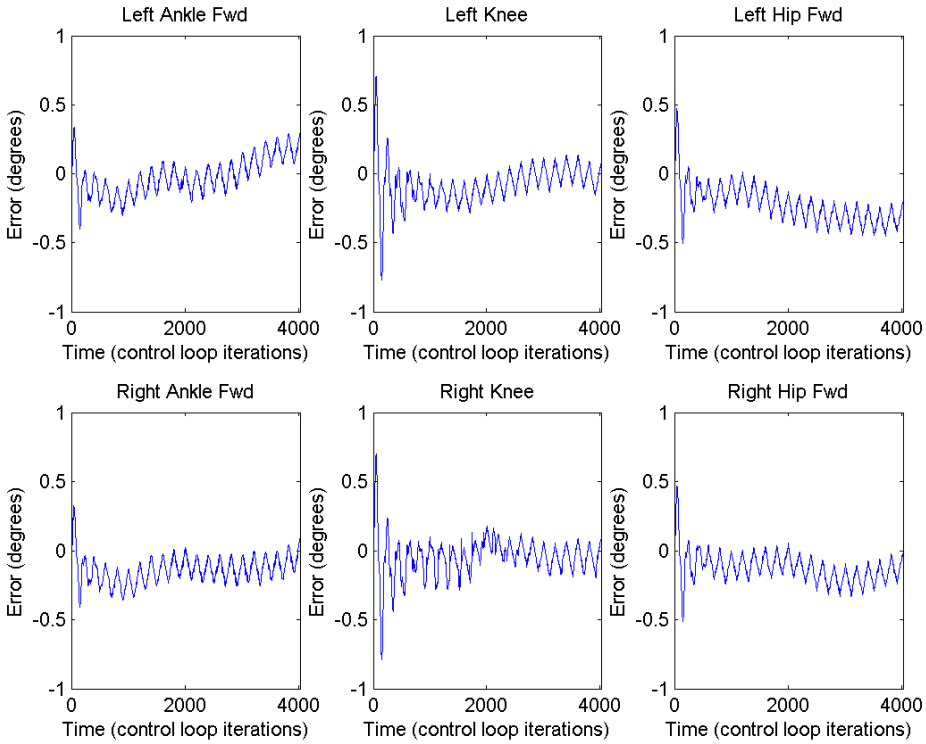

Figure 5.3 – Error in position, fuzzification on actual velocity – multi-axis

Figure 5.3 shows an improvement in the trajectory tracking of each of the joints, however, the error plots are not as smooth as the error plots from the cart model. The learning time of the algorithm is fast, taking only a single iteration of the crouch behaviour to minimise the error.

5.4.2. Fuzzification of Velocity Error

An alternative approach to using the actual velocity as an input to the FAM algorithm is to fuzzify the error in velocity.

Velocity Error Experiment – Multiple axis compensation

Figure 5.4 shows the results of fuzzifying the error in velocity, for multiple axis compensation for the crouching behaviour. The same FAM parameters as shown in Table 5.2 were again used in this experiment.

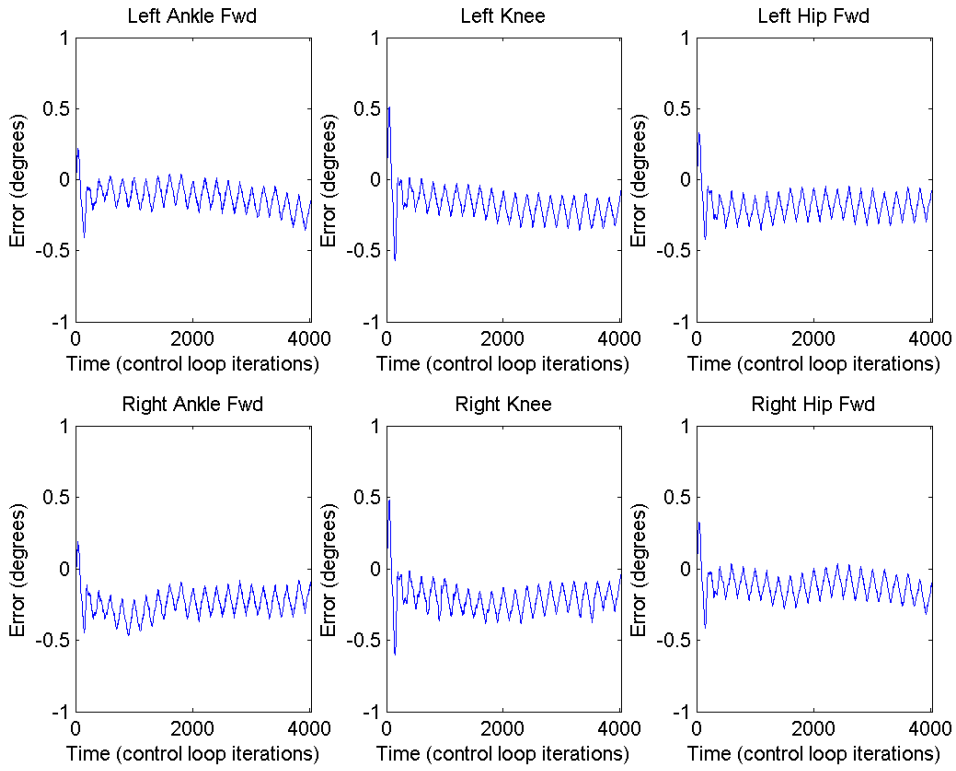

Figure 5.4 – Error in position, fuzzification on velocity error – multi-axis

This approach gave a slight improvement in the smoothness and symmetry of the trajectory tracking error plots; however the shape of the error plots is still far from the error plots obtained from the cart model.

5.4.3. *Fuzzification of Actual Position*

Yet another alternative approach tested was to use the actual position as an input to the FAM algorithm.

Actual Position Experiment – Multiple axis compensation

The approach of fuzzifying the actual position was tested using the crouching behaviour. The parameters described in Table 5.2 were used. Figure 5.5 shows an example of the unstable error plots.

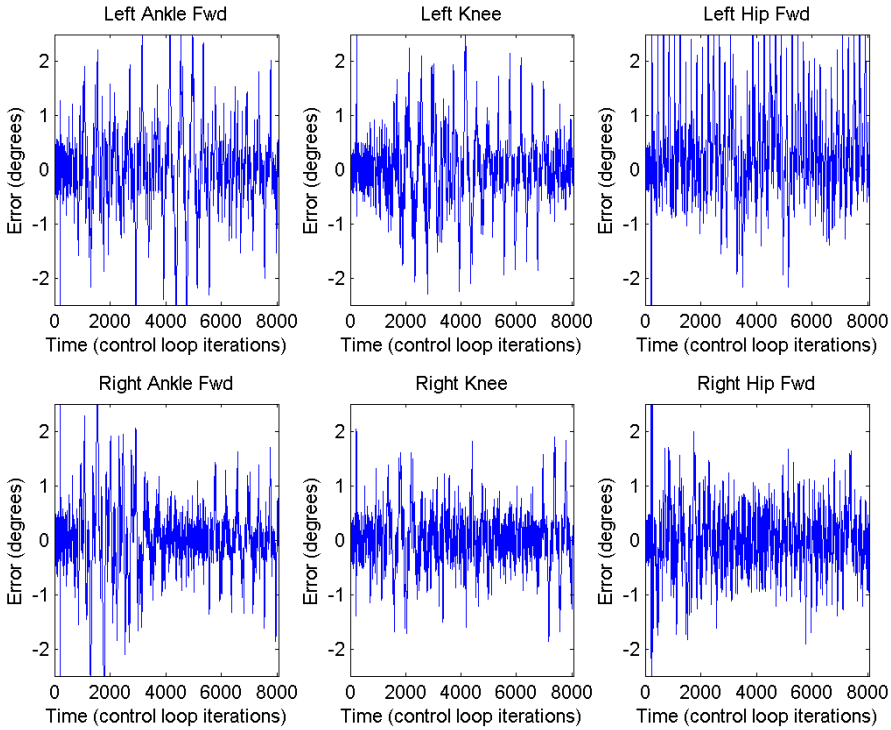

Figure 5.5 – Error in position, fuzzification on actual position – multi-axis

The error plot shows that this approach has yielded no stable trajectory tracking improvement.

5.4.4. Fuzzification of Position Error

The final and most successful approach was to use the position error as an input to the FAM algorithm. This approach yielded the quickest and most stable trajectory tracking improvement.

Position Error Experiment – Multiple axis compensation test 1

Figure 5.6 shows the best results for this approach, with the FAM parameters shown in Table 5.3 and Table 5.4. The FAM algorithm was used to compensate the knee and hip joints only.

Table 5.3 – FAM algorithm parameters

FAM Parameter	Value
Number of triangles in membership function	33
Learning rate	0.015625
Height of triangles	256
Centre of end triangles for input membership function	-65536, +65536
Lookup table value limit	±131072
Output limit on compensation signal	±10

Table 5.4 – FAM algorithm pre-scaler values

GuRoo Joint	Pre-scaler Value
Left and right ankle forward joint	200
Left and right knee joint	110
Left and right hip forward joint	250

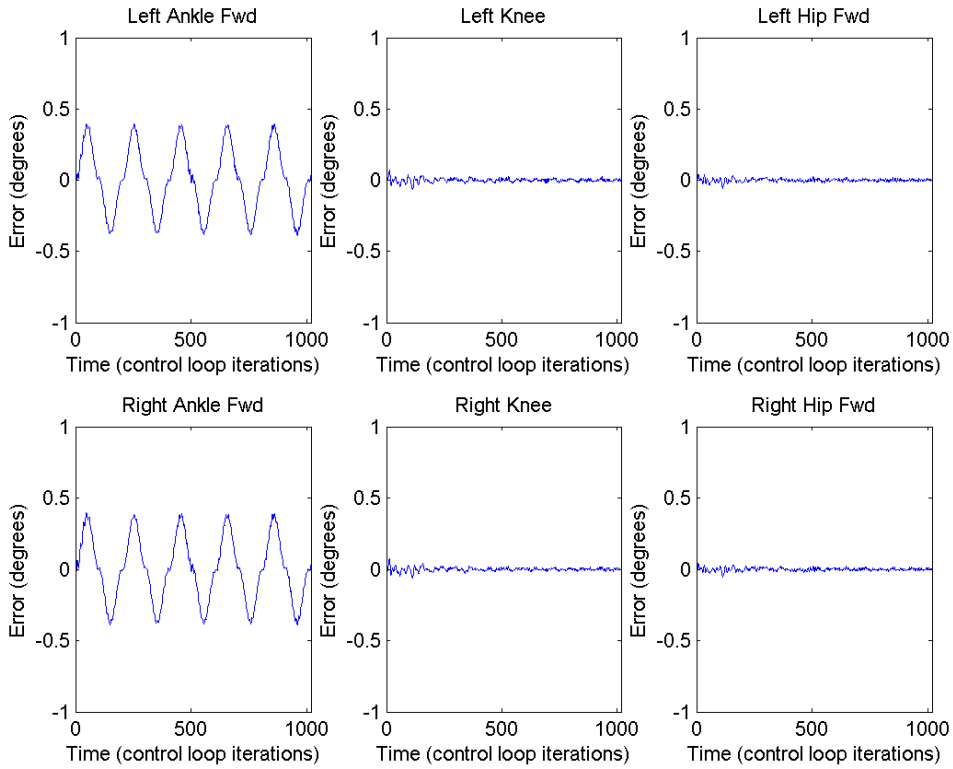

Figure 5.6 – Error in position, fuzzification on position error – multi-axis

The error plot shows a significant improvement in the trajectory tracking of the joints. The error is reduced to a minimum almost immediately. The error has been reduced to the extent that an improvement of almost 100% has been made.

Position Error Experiment – Multiple axis compensation test 2

Compensation of the ankle joint has exposed further complications to the system. With each joint in the leg learning, the effect of altering the motion of one joint may affect the motion of another joint. Hence, if all joints are learning simultaneously, the system may become unstable. This phenomenon is known as co-evolution. Extending the test above to compensate for the ankle joints exposes this problem. Figure 5.7 shows the error plot for that test.

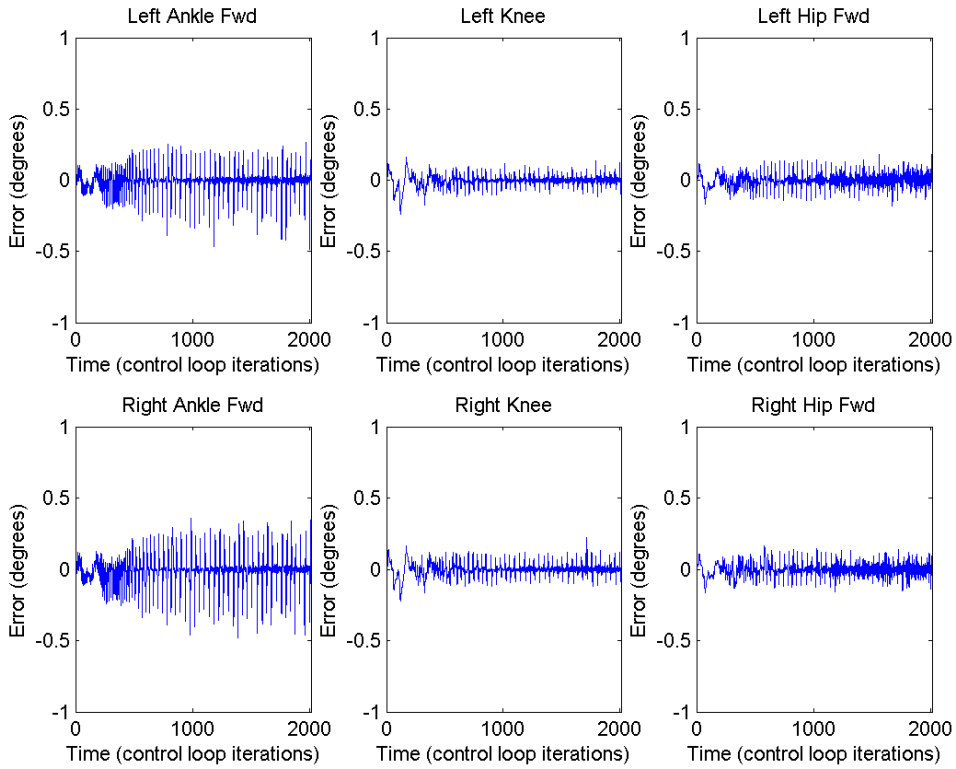

Figure 5.7 – Error in position, fuzzification on position error – multi-axis

The plot exposes an unstable trend in the error. This experiment shows that compensation for all joints has been affected by the problem of co-evolution.

5.5. SURFACE PLOT OF LOOKUP TABLES

A surface plot of the lookup table gives a visualisation of the compensation signal. This is helpful when fine tuning the FAM algorithm, as it shows the use of the lookup table, particularly, if the lookup table is becoming saturated. If all of the elements in the lookup table are non-zero, it may indicate that the shape of the membership needs to be adjusted, particularly the limits. Non-zero elements at the limits of the input variable axis give little room for disturbances, and because these fuzzy variables are more generalised, it can cause significant instabilities. Figure 5.8 through Figure 5.10 show the series of lookup tables generated in Position Error Experiment – Multiple axis compensation test 2 for the left leg only.

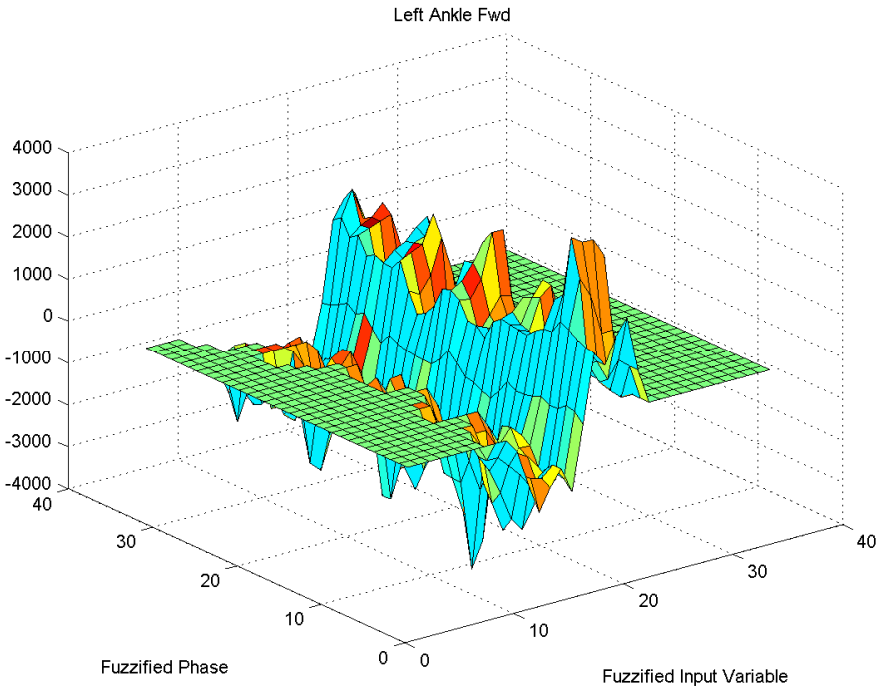

Figure 5.8 – Lookup table for left ankle forward joint

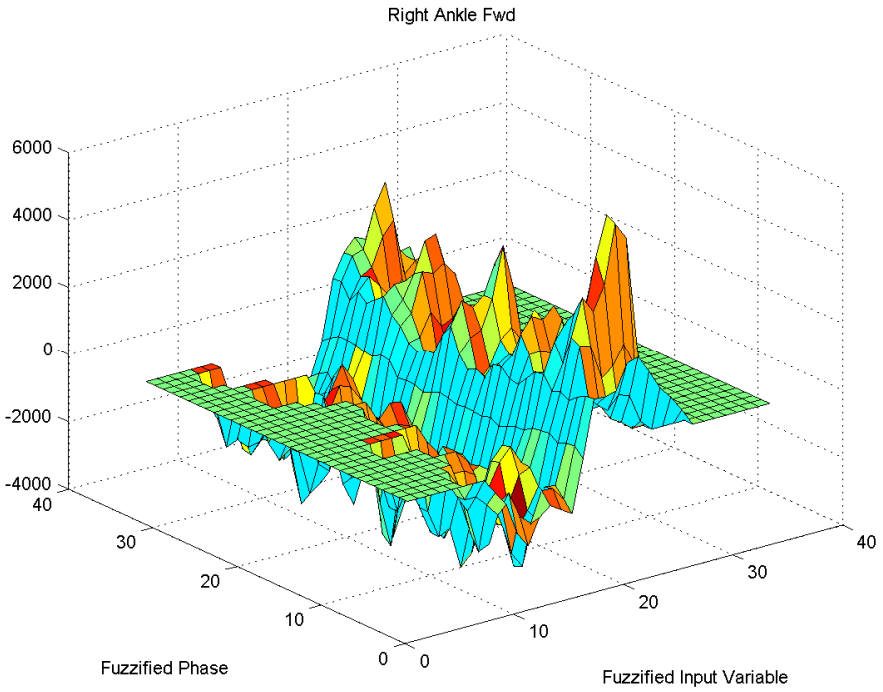

Figure 5.9 – Lookup table for right ankle forward joint

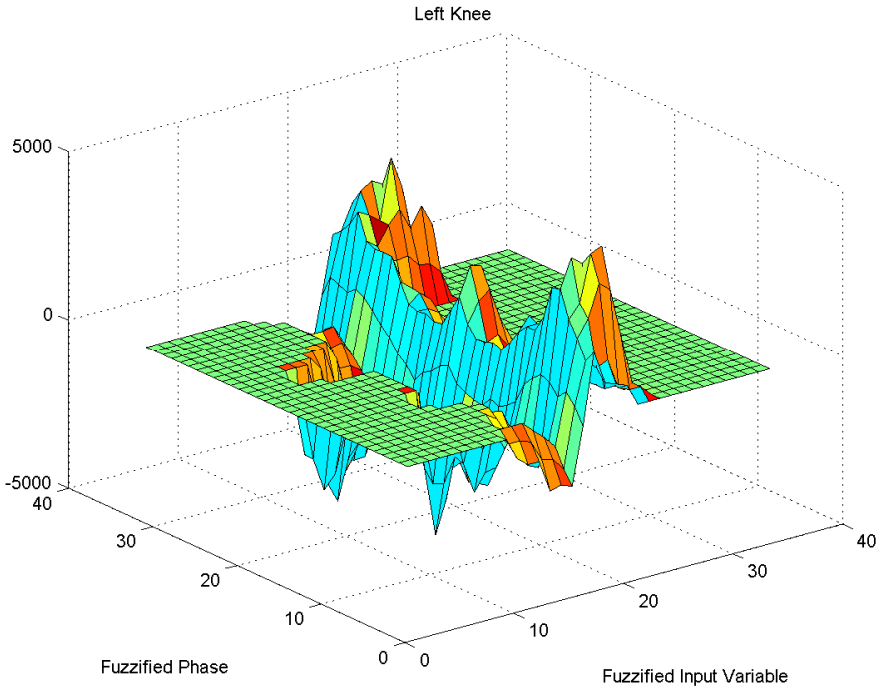

Figure 5.10 – Lookup table for left knee joint

Figure 5.8 through Figure 5.10 show the appropriate use of the lookup table. The shape of the compensation signal in the input variable plane follows a roughly sinusoidal shape. This is the expected shape, as it matches the uncompensated error in the joint.

The values in the lookup table will become saturated and incorrect if the tuning parameters of the FAM are incorrect. An example of an incorrectly used lookup table is shown in Figure 5.11. Some of the values have saturated to the limits, and the end fuzzy input variables are non-zero, indicating that the range of input values the membership function covers is too narrow.

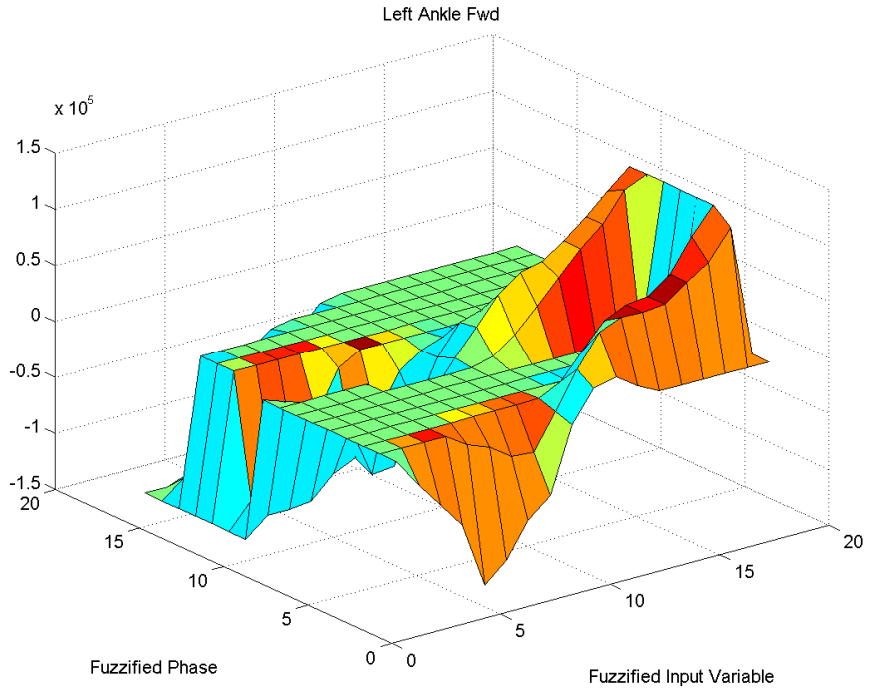

Figure 5.11 – Incorrectly populated lookup table

6. CONCLUSIONS

6.1. CONCLUSION

In many instances, conventional feedback control is insufficient for controlling dynamic systems with many degrees of freedom and complex dynamic models. The Trajectory Error Learning technique is a proven concept for improving trajectory tracking using a feed-forward control system. The feed-forward control system may be best utilised by implementing a neural network to learn the compensation signal. The Fuzzy Associative Memory system is a successful feed-forward, predictive, self-learning, self-supervised control approach to this problem.

The FAM algorithm was successfully developed and tested on single plane-of-motion, second order cart model in Matlab. The cart model showed that in an ideal model the trajectory tracking error can be minimised to zero using the FAM algorithm. The learning rate and the shape of the membership function were revealed to be the two major tuning parameters of the FAM algorithm. An increase in the learning rate resulted in a faster response of the system, at the cost of larger overshoot. Decreasing the number of triangles in the membership function quickened the response time without increasing overshoot, however at the cost of stability.

Fixed-point arithmetic implementation in C revealed that the values in the lookup table had to be limited to prevent overflow and subsequent corruption of the compensation signal. Output limiting the compensation signal revealed that a steady-state error may be introduced, if the error in the PD control loop is large.

The implementation of this control system on the GuRoo robot has shown that a significant improvement in trajectory tracking error can be made using this algorithm. Fuzzification of the position error gave the best results, with an improvement in error of almost 100% within one iteration of the crouching behaviour.

The FAM algorithm has out-performed the previous work using the CMAC system, in both computer resources including memory and processing power, and the control criteria of learning time.

6.2. FURTHER WORK

The Trajectory Error Learning model is a generic model, and may be applied to any control problem where trajectory tracking error needs to be minimised. The FAM algorithm may be implemented on alternative control problems other than the motor position control problem.

Further study of the application of the FAM algorithm to the GuRoo robot is required. The algorithm was unable to be tested on the real robot; instead this research has been limited to the GuRoo simulator. A set of real results would confirm that this algorithm is a viable solution to the problem of improving trajectory tracking error in complex dynamic systems.

A further study of the implementation of this algorithm to the joint controller boards *[Hall, 2004]* is also required. Implementation on the joint controller boards would minimise the communication signals between the boards and the central processing unit of the GuRoo. Implementation of the algorithm on the joint controller boards would require a thorough understanding of the capabilities of the joint controller boards, including processing power, memory availability, control loop rate, and algorithm completion time.

The co-evolutionary problem requires further study. This problem occurs because multiple joints are being trained simultaneously, attempting to minimise the error in each joint. When each joint is compensated, it not only affects the tracking of its joint, but due to the multi-degree of freedom in the system, it also affects the tracking of each of the other joints. This process occurs in each of the joints, all affecting each other simultaneously.

One approach to this problem is to extend the number of dimensions of the lookup table, such that a unique compensation signal is given for each position of all of the motors in a system. This approach however would use more memory according to the size of the lookup table, and be more processor intensive, having to iterate through the larger lookup table.

BIBLIOGRAPHY

[Collins, 2002] *Cerebellar Modelling Techniques for Mobile Robot Control in a Delayed Sensory Environment*, David Timothy Collins, University of Queensland, 2002.

[Hall, 2004] Joint Controller Development for a Humanoid Robot, Simon Hall, University of Queensland, 2004.

[Kee, 2003] *Confirmation of PhD Candidature*, Damien Kee, University of Queensland, 2003.

[Kee, Wyeth, 2003] *Cerebellar Joint Compensation for a Humanoid Robot*, Damien Kee and Gordon Wyeth, University of Queensland, 2003.

[Low, 2003] *Active Balance for a Humanoid Robot*, Toby Daniel Low, University of Queensland, 2003.

[Marshall, 2002] *Active Balance Control for a Humanoid Robot*, Ian Joseph Marshall, University of Queensland, 2002.

[McMillian, 1995] *Computational Dynamics for Robotic Systems on land and Underwater*, S. McMillian, Ohio State University, 1995.

[Miller; Glanz; Kraft, 1990] *CMAC: An Associative Neural Network Alternative to Backpropagation*, W. Thomas Miller, III, Filson H. Glanz, L. Gordon Kraft, III, Proceedings of the IEEE, Vol. 78, No. 10, 1999.

[Pike, 2003] *Gait Generation for a Humanoid Robot*, Tim Pike, University of Queensland, 2003.

[Self, 1990] *Designing with Fuzzy Logic*, Kevin Self, Spectrum, IEEE, Vol.27, Iss.11, Nov 1990.

[Si; Zhang; Tang, 1999] *Modified Fuzzy Associative Memory Scheme Using Genetic Algorithm*, Jie Si, Naiyao Zhang, and Rilun Tang, Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, Vol.3, Iss., 1999.

[The RoboCup Federation, 1998-2003] *What is RoboCup*, The RoboCup Federation, 1998-2003. http://www.robocup.org/02.html

[Widrow, Stearns, 1985] *Adaptive Signal Processing*, B. Widrow and S. D. Stearns, Prentice-Hall, 1985.

APPENDIX A – FAM.C MODULE

```
/*************************************
 *
 * fam.c
 * Christopher Myatt
 * Created 2 August 2004
 * Adaptive Feed-forward control algorithm:
 * Fuzzy Associative Memory
 * Last updated: 26 October 2004
 *
*************************************/


/* ****** */
/* macros */
/*        */

#define MAX(A,B) ((A) > (B) ? (A) : (B))
#define CEIL(A) ((A) > 0 ? 1 : 0)


/* ******************** */
/* fam tuning parameters */
/*                      */

/* membership function made up of 2^(NUM_TRI) + 1 triangles */
#define NUM_TRIANGLES_PWR_TWO 4

/* inverse of learning rate, closer to 1, larger rate */
#define LEARNING_RATE_PWR_TWO 4

/* (soft) bounds of the inputs ie. -2^(LIMIT) < input < 2^(LIMIT) */
#define X_LIMIT_PWR_TWO 16
#define PHASE_LIMIT_PWR_TWO 9

/* the maximum degree of membership (the height of the triangles) */
#define DOM_MAX_PWR_TWO 5


/* *****************/
/* fam definitions */
/*                 */

#define DOM_MAX (1<<DOM_MAX_PWR_TWO)

#define NUM_TRIANGLES ((1<<NUM_TRIANGLES_PWR_TWO)+1)
#define HALF_NUM_TRIANGLES (1<<(NUM_TRIANGLES_PWR_TWO-1))

#define X_TRIANGLE_WIDTH_PWR_TWO (X_LIMIT_PWR_TWO-NUM_TRIANGLES_PWR_TWO+1)
#define PHASE_TRIANGLE_WIDTH_PWR_TWO (PHASE_LIMIT_PWR_TWO-NUM_TRIANGLES_PWR_TWO+1)

#define X_TRIANGLE_WIDTH (1<<X_TRIANGLE_WIDTH_PWR_TWO)
#define PHASE_TRIANGLE_WIDTH (1<<PHASE_TRIANGLE_WIDTH_PWR_TWO)

#define LOOKUP_TABLE_MAX (1<<(X_LIMIT_PWR_TWO+1))
#define LOOKUP_TABLE_MIN (-(1<<(X_LIMIT_PWR_TWO+1)))


/* *************** */
/* global variables */
/*                 */

int RuleTable[NUM_TRIANGLES-1][NUM_TRIANGLES];
```

```
struct X_MEM {
  int center[NUM_TRIANGLES];
  int dom[NUM_TRIANGLES];
} xMem;

struct PHASE_MEM {
  int center[NUM_TRIANGLES];
  int dom[NUM_TRIANGLES-1];
} phaseMem;


/* ********** */
/* prototypes */
/*            */

void fuzzifyX(int u);
void fuzzifyPhase(int u);
int leftall(int u, int w, int c);
int triangle(int u, int w, int c);
int rightall(int u, int w, int c);

/* **************
/* initFuzzySys()
/*
/* initialise the shape of the membership
/* functions and the lookup table.
/* ************** */
void initFuzzySys() {

  int i, j;

  /* calculate the center of each of the membership functions */
  for (i=0; i<NUM_TRIANGLES; i++) {
    xMem.center[i] = (i - HALF_NUM_TRIANGLES) * X_TRIANGLE_WIDTH;
    phaseMem.center[i] = (i - HALF_NUM_TRIANGLES) *
                         PHASE_TRIANGLE_WIDTH;
  }

  /* initialise the rule table to zeros */
  for (i=0; i<NUM_TRIANGLES-1; i++) {
    for (j=0; j<NUM_TRIANGLES; j++) {

      RuleTable[i][j] = 0;
    }
  }
}


/* ************************************
/* int fam(int xDes, int phase, int xAct)
/*
/* the main function, calculates a corrections signal
/* for a desired input, actual input, and phase signal.
/* ************************************ */
int fam(int xDes, int phase, int xAct) {

  int correction, error;
  int i, j;

  /* fuzzify the phase and the input variable */
  fuzzifyX(xAct);
  fuzzifyPhase(phase);

  correction = 0;
  error = xDes - xAct;


  /* the main loop, calculates the correction by iterating
     through the rule table, and adjusts the rule table
     according to the error */
```

```
  for (i=0; i<NUM_TRIANGLES-1; i++) {
    for (j=0; j<NUM_TRIANGLES; j++) {

      correction = correction + ( (phaseMem.dom[i] * xMem.dom[j] * RuleTable[i][j])
                                  >> (DOM_MAX_PWR_TWO*2) );
      RuleTable[i][j] = RuleTable[i][j] + ( (CEIL(phaseMem.dom[i]) * CEIL(xMem.dom[j])
                                          * error) >> LEARNING_RATE_PWR_TWO );

      /* limit overflow in the lookup table */
      if (RuleTable[i][j] > LOOKUP_TABLE_MAX) RuleTable[i][j] = LOOKUP_TABLE_MAX;
      else if (RuleTable[i][j] < LOOKUP_TABLE_MIN) RuleTable[i][j] = LOOKUP_TABLE_MIN;
    }
  }
  return correction;
}

/* ********************
/* void fuzzifyX(int u)
/*
/* fuzzify the input variable.
/* ******************** */
void fuzzifyX(int u) {

  int i;

  xMem.dom[0] = leftall(u, X_TRIANGLE_WIDTH_PWR_TWO, xMem.center[0]);

  for (i=1; i<NUM_TRIANGLES-1; i++) {

    xMem.dom[i] = triangle(u, X_TRIANGLE_WIDTH_PWR_TWO, xMem.center[i]);
  }

  xMem.dom[i] = rightall(u, X_TRIANGLE_WIDTH_PWR_TWO, xMem.center[i]);
}

/* ************************
/* void fuzzifyPhase(int u)
/*
/* fuzzify the phase - wrap around implemented.
/* ************************ */
void fuzzifyPhase(int u) {

  int i;

  /* special case: wrap around of fuzzy variables */
  phaseMem.dom[0] = triangle(u, PHASE_TRIANGLE_WIDTH_PWR_TWO, phaseMem.center[0]) +
        triangle(u, PHASE_TRIANGLE_WIDTH_PWR_TWO, phaseMem.center[NUM_TRIANGLES-1]);

  for (i=1; i<NUM_TRIANGLES-1; i++) {

    phaseMem.dom[i] = triangle(u, PHASE_TRIANGLE_WIDTH_PWR_TWO, phaseMem.center[i]);
  }
}

/* ********************************
/* int leftall(int u, int w, int c)
/*
/* calculate degree of membership for a left-most fuzzy variable.
/* ******************************** */
int leftall(int u, int w, int c) {

  if (u <= c) return DOM_MAX;
  else {
    if (w >= DOM_MAX_PWR_TWO)
      return MAX(0, (DOM_MAX - ((u-c) >> (w-DOM_MAX_PWR_TWO))));
    else
      return MAX(0, (DOM_MAX - ((u-c) << (DOM_MAX_PWR_TWO-w))));
  }
}
```

```
/* ********************************
/* int triangle(int u, int w, int c)
/*
/* calculate degree of membership for a standard fuzzy variable.
/* ******************************** */
int triangle(int u, int w, int c) {

  if (u == c) return DOM_MAX;
  else if (u > c) {
    if (w >= DOM_MAX_PWR_TWO)
      return MAX(0, (DOM_MAX - ((u-c) >> (w-DOM_MAX_PWR_TWO))));
    else
      return MAX(0, (DOM_MAX - ((u-c) << (DOM_MAX_PWR_TWO-w))));
  }
  else {
    if (w >= DOM_MAX_PWR_TWO)
      return MAX(0, (DOM_MAX - ((c-u) >> (w-DOM_MAX_PWR_TWO))));
    else
      return MAX(0, (DOM_MAX - ((c-u) << (DOM_MAX_PWR_TWO-w))));
  }
}

/* ********************************
/* int rightall(int u, int w, int c)
/*
/* calculate degree of membership for a right-most fuzzy variable.
/* ******************************** */
int rightall(int u, int w, int c) {

  if (u >= c) return DOM_MAX;
  else {
    if (w >= DOM_MAX_PWR_TWO)
      return MAX(0, (DOM_MAX - ((c-u) >> (w-DOM_MAX_PWR_TWO))));
    else
      return MAX(0, (DOM_MAX - ((c-u) << (DOM_MAX_PWR_TWO-w))));
  }
}
```