COPYRIGHT AND CITATION CONSIDERATIONS FOR THIS THESIS/ DISSERTATION

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

- **NonCommercial** — You may not use the material for commercial purposes.

- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

**How to cite this thesis**

TRAJECTORY PLANNING AND CONTROL FOR AUTONOMOUS ROBOTS

by

JOHANNES GERHARDUS BENADÉ

A Thesis submitted as partial fulfilment of the requirements for the degree

MASTER OF ENGINEERING

MECHANICAL ENGINEERING

FACULTY OF ENGINEERING

at the

RAND AFRIKAANS UNIVERSITY

SUPERVISOR : Dr A L NEL

November 1993
Abstract

The research reported in this thesis describes the design and simulation of a neural controller for a three-degree-of-freedom robot leg for use as an hexapod leg. Biological systems are considered as a motivation to develop the neural control system for hexapod walking on a horizontal surface.

Backpropagation training of multilayer perceptrons and a combination of heterogeneous neurons are used to implement several pattern generators with different behaviours. The artificial neurons are simulated and connected together with the pattern generators to form a complete control system. Previous work [48] shows the performance of a two-degree-of-freedom leg controller - this type of controller however cannot compensate for surface irregularities. The control system for the three-degree-of-freedom leg is then further extended to compensate for surface irregularities that cannot be traversed by the two-degree-of-freedom leg.
Acknowledgements

I would like to take this opportunity to extend my thanks to everyone whose help aided me directly, and especially indirectly, to complete this work. My supervisor André Nel for allowing me to learn more than what is contained in these pages. My colleagues, Rudolf and Jan for their ideas and hours of planning. And especially my father for his unending support and interest.
# Table of Contents

Abstract  
Acknowledgements  
List of Figures  

## Chapter 1 INTRODUCTION

1.1 Machine intelligence in Mobile Robots .............................................. 2  
1.2 Research Hypothesis ........................................................................... 3  
1.3 Overview of Report ........................................................................... 4  

## Chapter 2 NEUROPHYSIOLOGY

2.1 Introduction .................................................................................. 5  
2.2 Building Blocks of Biological Neural Networks .................................. 5  
2.3 Sensory and Motor Neurons ............................................................. 6  
2.4 A Proposed Artificial Neuron ............................................................ 8  
2.5 The Neuron Model ........................................................................... 9  
2.6 Neural Network ............................................................................ 11  
   2.6.1 The Pacemaker Model .............................................................. 11  
   2.6.2 Sensory and Motor Neurons ..................................................... 12  
   2.6.3 Compound Synapses ............................................................... 13  
2.7 Summary ....................................................................................... 14  

## Chapter 3 WALKING BEHAVIOUR OF INSECTS

3.1 Introduction .................................................................................. 15  
3.2 Parameters of walking behaviour ................................................... 16  
3.3 Pattern of Leg Recovery .................................................................. 17  
3.4 Parameters Insects Regulate During Walking .................................. 17  
3.5 The Behavioural Effects of Interference with Walking .................. 18  
   3.5.1 Terrain .................................................................................... 18  
   3.5.2 Load ...................................................................................... 19  
   3.5.3 Obstruction ............................................................................ 19  
   3.5.4 Amputation ............................................................................ 20
Chapter 4 AUTONOMOUS NETWORKS

4.1 Introduction .......................................................... 24
4.2 Neural Basis of Locomotion in Biological Systems ................. 24
4.2.1 The Biological Step Cycle ...................................... 25
4.3 Proposed Leg Controller .............................................. 26
4.4 Leg Controller Model for a Three Degree of Freedom Leg ......... 29
4.5 Overview .................................................................. 32

Chapter 5 KINEMATICS OF MANIPULATION ROBOTS

5.1 Introduction .............................................................. 33
5.2 Degrees of freedom of a solid ....................................... 33
5.3 Description of mechanical systems .................................. 33
5.3.1 Borrel's Description ................................................ 34
5.3.2 Association between a graph and a mechanism .............. 35
5.3.3 Definition of a chain ............................................... 35
5.4 Coordinate Transformations .......................................... 36
5.4.1 Introduction ........................................................ 36
5.4.2 Homogeneous Transformations ................................ 36
5.4.3 Homogeneous Transformation matrices ....................... 38
5.4.4 Solving the Kinematic Equations ............................... 40
5.5 Summary .................................................................. 42

Chapter 6 IMPLEMENTATION

6.1 Introduction .............................................................. 43
6.2 Calculation of the Kinematic Equations ............................. 43
6.3 The Inverse Kinematic Problem ..................................... 48
# List of Figures

| Figure 2.1 | A schematic representation of the brain as a system of sequential stations - input (S), associative central processing (A) and motor output (M). | 7 |
| Figure 2.2 | A schematic drawing of a typical neuron. | 7 |
| Figure 2.3 | A schematic diagram of a McCulloch-Pitts neuron. | 8 |
| Figure 2.4 | The model neuron. | 10 |
| Figure 2.5 | Schematic illustration of a compound synapse. | 14 |
| Figure 4.1 | Pearson's model. | 25 |
| Figure 4.2 | Leg Controller Circuit proposed by Beer. | 26 |
| Figure 4.3 | Proposed leg controller for a three-degree-of-freedom robot leg. | 28 |
| Figure 5.1 | The six DOF of a solid: three translational \((T_1, T_2, T_3)\) and three rotational \((R_1, R_2, R_3)\). | 33 |
| Figure 5.2 | Control of an AMS using a computer. | 34 |
| Figure 5.3 | An example of graph-mechanism association. | 35 |
| Figure 5.4 | Length \(a\) and twist angle \(\alpha\) of a link. | 36 |
| Figure 5.5 | Link parameters \(\theta, d, a\) and \(\alpha\). | 37 |
| Figure 5.6 | Denavit-Hartenberg kinematic parameters for a prismatic joint. | 38 |
| Figure 6.1 | Schematic representation of the robot leg. | 43 |
| Figure 6.2 | Transformations between coordinate systems. | 46 |
| Figure 6.3 | Extreme values of the robot leg. | 48 |
| Figure 6.4 | Parameters of the robot leg. | 50 |
| Figure 6.5 | Using a connectionist network for adaptive prediction. | 52 |
| Figure 6.6 | Modified system for adaptive prediction. | 53 |
| Figure 6.7 | Motor patterns when stepping onto an object, a) motor 2 and b) motor 3. | 56 |
| Figure 6.8 | Motor pattern when stepping in a hole, a) motor 2 and b) motor 3. | 56 |
| Figure 6.9 | Modifications to proposed model for uneven terrain. | 57 |
| Figure 7.1 | Stance phase angles: \(\theta, \phi\) and \(\psi\). | 60 |
| Figure 7.2 | Schematic representation of the stance phase, a) 3D view, b) top view, c) front view, and d) side view. | 60 |
| Figure 7.3 | Swing phase angles: \(\theta, \phi\) and \(\psi\). | 61 |
Figure 7.4  Schematic representation of the swing phase. a) 3D view, b) top view, c) front view, and d) side view. ................................................................. 61
Figure 7.5  Simulation of a complete step ................................................................. 62
Figure 7.6  Possible Output Values from the Signal Processing Neuron. a) Stance Phase and b) Swing Phase. ................................................................. 63
Figure 7.7  PG Output for Motor 1 - Stance Phase. ....................................................... 64
Figure 7.8  PG Output for Motor 2 - Stance Phase. ....................................................... 64
Figure 7.9  PG Output for Motor 3 - Stance Phase. ....................................................... 64
Figure 7.10  PG Output for Motor 1 - Swing Phase. ..................................................... 65
Figure 7.11  PG Output For Motor 2 - Swing Phase. .................................................... 65
Figure 7.12  PG Output for Motor 3 - Swing Phase. ..................................................... 65
Figure 7.13  Trajectory planned by the pattern generators. ........................................... 66
Figure 7.14  PG output after connection to the proposed controller. ................................ 67
Figure 7.15  Schematic representation of the stance phase after the connection to the proposed controller. a) 3D view, b) top view, c) front view and d) side view. ................................................................. 68
Figure 7.16  Schematic representation of the swing phase after the connection to the proposed controller. a) 3D view, b) top view, c) front view and d) side view. ................................................................. 68
Figure 7.17  Adapted curves against ideal curves for Motor 2. ....................................... 70
Figure 7.18  Adapted curves against ideal curves for Motor 3. ....................................... 70
Figure 7.19  Adapted curve plotted against ideal curve for stepping on an object 23mm above the horizontal level. ......................................................... 71
Figure 7.20  Three-dimensional view of robot leg when stepping on an object 23mm above the horizontal level. ................................................................. 71
CHAPTER ONE

Introduction

The fact is, that civilization requires slaves. The Greeks were right there. Unless there are slaves to do the ugly, horrible, uninteresting work, culture and contemplation become almost impossible. Human slavery is wrong, insecure and demoralizing. On mechanical slavery, on the slavery of the machine, the future of the world depends.

OSCAR WILDE

Besides the sheer thrill of creating machines that can actually run, there are two serious reasons for exploring legged machines. One reason is mobility: There is an increasing need for vehicles that can travel on difficult terrain, where existing vehicles cannot go. Wheels excel on prepared surfaces such as rails and roads, but are poor at negotiating terrain that is soft or uneven. Because of these limitations only about half the earth's landmass is accessible to existing wheeled and tracked vehicles, whereas a much greater area can be reached by animals on foot. It should be possible to build legged vehicles that can go to places that at present only animals can reach.

One reason legs provide better mobility in rough terrain is that they can use isolated footholds that optimize support and traction, whereas a wheel requires a continuous path of support. Consequently, a legged system is free to choose among the best footholds in the reachable terrain whereas a wheel is forced to negotiate the terrain as it comes.

Another advantage of legs is that they provide an active suspension that decouples the path of the body from the paths of the feet. The payload is free to travel smoothly irrespective of pronounced variations in terrain. A legged system can also step over obstacles. The performance of legged vehicles can, to a greater extent, be independent of the macro or micro level of roughness of the ground.

The construction of legged vehicles depends on progress in several areas of science and engineering. Legged vehicles will need systems that control joint motions, the sequence use of legs, monitor and manipulate balance, generate motions to use known footholds, sense the terrain to find good footholds, and calculate negotiable foothold sequences.
A second reason for exploring legged machines is to understand how humans and animals use their legs for locomotion. Animals demonstrate a greater mobility and agility. They use their legs to move quickly and reliably through forest, swamp, marsh, and jungle, and from tree to tree. They move with great speed and efficiency.

Despite the skill we apply in using our own legs for locomotion, we are still at a primitive stage in understanding the principles that underlie walking and running. A question that needs to be asked is what control mechanisms do animals use?

1.1 Machine Intelligence in Mobile Robots

In most mobile robot systems developed so far, a common feature has been the recognition for the need of a hierarchical structure of intelligence and therefore hierarchical decomposition of the task (of locomotion) and the model of the perceived world. The higher level defines tasks or subgoals for the lower levels and monitor their status. As we go up in the hierarchy, on a time scale, there is a decrease in the frequency of updating sensory information, and on the length scale, the views considered is larger and contains with fewer details. At each successive level down the hierarchy, there is a decrease in generality and scope of the search and greater resolution. Four levels may be identified in the hierarchy - the planner or the route layout module, the navigator or the path selection module, the pilot or the guidance module and the controller.

**Planner** - The route layout is done at the highest level. It involves planning a route for the vehicle considering the general characteristics of the robot's ability to adapt to different terrains and conquer various obstacles. At this level, the basic element of locomotion, leg or wheel, is not of concern. In a completely autonomous system, it may be the only level that interacts with a human operator.

**Navigator** - The navigator is primarily concerned with path selection. It uses the terrain preview data and information from other sensors to chart a "best" course for several vehicle body lengths to comprehend the subgoal command from the upper level. Obstacle avoidance is an integral part of such a process. It in turn defines subgoals for the pilot level to meet the requirements of the selected path.

**Pilot** - The pilot plans a sequence of elementary acts of motion in space and time to generate a path between the goals prescribed by the path selection level. The pilot may tolerate minor deviations from the prescribed path to avoid small obstacles. The pilot has short term memory and its perception of the world is confined to one or two body lengths. In wheeled vehicles or robots this level may be virtually absent where the motion is almost fully constrained or specified by the path selection level. A legged locomotion system, however, is free to select footholds in space and time. That is, when and where it "samples" the terrain. Thus optimizations of gait parameters, optimal foothold selection, dynamic
balancing and attitude control of the robot body are tasks that must be performed at this level. The choice of legs as elements of locomotion introduces a new level of intelligence and a new degree of complexity into motion planning for the body.

**Controller** - The controller is the lowest level and is the only level which interacts directly with the actuators. It represents the "spinal" level associated with control of individual joints in biological systems and involves real-time servo control loops and sensory feedback at the actuator level. In legged locomotion systems, this also entails leg trajectory planning, and actuation, and incorporates "cerebellar" intelligence to a certain extent.

The four hierarchical levels are all necessary when developing a complete autonomous robot. However, one has to start with substantial research in each of these areas in order to produce a robust robot. At the stage where research in each of these areas has reached a point of "satisfaction" these levels could then be connected.

The primary aim of this project is to do research in the area of the controller. We aim to develop a robot leg with three degrees of freedom that is controlled by a neural controller. Intelligence is an essential part of the controlling system and must therefore be included in the proposed controller.

Only a leg that can walk on a flat surface is considered. Modifications to the controller to compensate for irregularities in terrain will be made at a stage when the controller has reached a point where control of the leg is satisfactory.

The design process had two stages: the kinematics of a three-degree-of-freedom leg was determined and simulated. The second stage involved the development of a controlling system for the leg and consisted of computer simulations in which different neural models, leg controllers, locomotion controllers and pattern generators were designed and tested.

1.2 Research Hypothesis

In this research, the research hypothesis that will be evaluated can be stated as follow:

*That the use of neural networks and pattern generators, to construct an intelligent controller, will enable the control of a three-degree-of-freedom robot leg, for walking on a horizontal surface by an hexapod.*
1.3 Overview of Report

The chapters, however, are presented in such a way that the reader is made familiar with all the terms and biological concepts used in the construction of the leg controller before commencing with a detail discussion of the results of the project.

Chapter 2 discusses neurophysiology. A brief overview of the basic building blocks for biological neural networks are given. The significant intermediate between biological neurons and those used in artificial networks is considered. The artificial neuron which forms the basis of the neural controller is explained as well as the variations to this model which is used to implement the different types of neurons used in the proposed controller.

When designing artificial controllers, based on biological systems, a basic knowledge of the biological systems is necessary. Chapter 3 deals with the walking behaviour in insects. The parameters that influence walking behaviour are discussed as well as the parameters insects regulate during walking. Another interesting topic discussed is the effects interference of walking has on the behaviour of insects. Sense organs and the effects of their destruction are also discussed.

Chapter 4 deals with autonomous networks. The neural basis of locomotion is discussed as well as the artificial neural leg controller developed by other researchers. The proposed controller is also discussed.

Chapter 5 deals with the kinematics of manipulation robots. Coordinate transformations are discussed under the topics homogeneous transformations, homogeneous transformation matrices and the solving of the kinematic equations.

Chapter 6 deals with the implementation of all the theory discussed in the preceding chapters.

Chapter 7 deals with the simulation results of the various parts of the model as well as the results of the complete connected controller simulation.

Chapter 8 gives a discussion of the project as well as suggesting future work in the area of neural network controlled legged locomotion.
2.1 Introduction

It is obvious that the human brain is superior to a digital computer at many tasks. A good example is the processing of visual information: a one-year-old baby recognises objects and faces more quickly than even the most advanced AI system running on the fastest supercomputer. Another example, is if a person were to toss a ball at you. In all likelihood you could catch the ball, even if you were a bit startled. Tracking the state of dynamic systems, such as the path of a ball thrown at you, is a nontrivial task for a computer [1]. Yet as a human, you can do this quite easily. You estimate the speed, trajectory, the weight (whether to duck or to catch it), and so on. The most important of all is that you do it in real time.

The brain has many other features that would be desirable in artificial systems.

- It is robust and fault tolerant. Nerve cells in the brain die everyday without affecting its overall performance significantly.
- It is flexible. It can easily adjust to a new environment by learning.
- It can deal with information that is fuzzy, probabilistic, noisy or inconsistent.
- It is highly parallel.
- It is small, compact and dissipates very little power.

In contrast the computer outperforms the brain only in tasks based on simple repetitive arithmetic!

2.2 Building Blocks of Biological Neural Networks

The basic elements of the brain are, neurons and synapses. The total amount of neurons (nerve cells) in the brain is of the order of $10^{11}$. There are a fairly large variety of neurons in the human nervous system; viz. variations are found in size, in structure and in function.
Amit et al [2] subscribe to a coarse division of the nervous system into three parts - input, central processing and output, as is expressed in the drawing in Figure 2.1. This is a division into neural systems, each rather complex in structure, with different functions and computational roles.

A basic neuron consists of a cell body or soma where the cell nucleus is located (see Figure 2.2). Tree-like networks of nerve fibre, called dendrites, are connected to the cell body. From the cell body a single long fibre, called the axon, extends, which eventually branches, or arborizes, into strands and substrands. At the end of these are the transmitting ends of the synaptic junctions, or synapses, to other neurons. The receiving ends of these junctions on other cells can be found both on the dendrites and on the cell bodies themselves. The axon of a typical neuron makes a few thousand synapses with other neurons depending on its location in the cortex. A most significant anatomical fact is that each neuron receives $10^4$ synaptic inputs from other neurons.

The transmission of a signal from one cell to another is a complex chemical process in which specific transmitter substances are released from the transmitting side of the junction. The effect is to lower or raise the electrical potential inside the body of the receiving cell. If this potential reaches a threshold, a pulse or action potential of fixed strength and duration is transmitted down the axon. It is then said that the cell has "fired". The pulse branches out through the axonal arborization to synaptic junctions and other cells. After firing, the cell has to wait for a duration, called the refractory period, before it can fire again.

### 2.3 Sensory and Motor Neurons

Two classes of nerve cells serve to connect the activity of neurons to the state of the body in which they are embedded: sensory neurons and motor neurons. Sensory neurons transduce physical properties of an animal or human's environment into electrical/chemical signals. The membranes of these nerve cells contain active channels whose configuration is effected by various physical properties, such as light intensity or force.

The membrane potential of a sensory neuron changes as these channels open or close in the presence of the appropriate sensory stimulus. Motor neurons transform electrical signals into some form of action involving the human body or an animal's body. Action-producing organs are called effectors. The transformation from electrical activity to action potentials is accomplished by synapses whose neurotransmitters trigger specific changes in the configuration of the associated effector organs.
Figure 2.1 A schematic representation of the brain as a system of sequential stations - input (S), associative central processing (A) and motor output (M).

Figure 2.2 A schematic drawing of a typical neuron.
2.4 A Proposed Artificial Neuron

A simple model of a neuron as a binary threshold unit was proposed by McCulloch and Pitts [3]. The model neuron computes a weighted sum of its inputs from other units and outputs a one or a zero according to whether this sum is above or below a certain threshold:

\[ n_i(t+1) = \theta \left( \sum_j w_{ij} n_j(t) - \mu_i \right) \]  (2.1)

(See Figure 2.3) Here \( n_i \) is either 1 or 0, and represents the state of neuron \( i \) as firing or not firing respectively. Time \( T \) is taken as discrete, with one time unit elapsing per processing step. \( \theta(x) \) is the unit step function, or Heaviside function:

\[ \theta(x) = \begin{cases} 1 & \text{if } x \geq 0; \\ 0 & \text{otherwise.} \end{cases} \]  (2.2)

The weight \( w_{ij} \) represents the strength of the synapse connecting neuron \( j \) to neuron \( i \). It can be positive or negative corresponding to an excitatory or inhibitory synapse respectively. If there is no synapse between \( i \) and \( j \) it is zero. The cell specific parameter \( \mu_i \) is the threshold value for unit \( i \). The weighted sum of inputs must reach or exceed the threshold for the neuron to fire.

This McCulloch-Pitts (M-P) device forms the basis of the neurons simulated in artificial neural networks (ANN) such as the multi-layer perceptron (MLP).

In biological neurons the post-synaptic potential due to pre-synaptic activation actually extends over a finite time and is graded. The biological neuron, in other words, acts like a current to frequency converter rather than a simple logical element [4].

---

**Neurophysiology**
2.5 The Neuron Model

A significant intermediate exists between biological neurons and those used in artificial networks. In other words the artificial neural networks are homogeneous while biological neural networks are heterogeneous. That is, the biological nervous system consists of many different types of nerve cells, each performing a specific function or range of functions. It also has the ability to temporally sum the inputs.

The model neuron used as basic building block of all the neurons used in the system is shown schematically in Figure 2.4. It represents the firing frequency of a biological cell as a voltage. This output voltage is a nonlinear function of its input potential. The neuron transfer function is a saturating linear threshold function with an initial jump in discontinuity as shown in Figure 2.4.

The transfer function, also known as the input/output function, is characterized by three parameters:

- The threshold voltage at which the neuron begins to fire.
- The minimum firing frequency. In other words the minimum output voltage of the neuron.
- The gain of the neuron.

Mathematically, if \( \xi_i \) is the input to the transfer function of the \( i \)-th neuron and \( F(V)_i \) is the output of that specific neuron, then

\[
F(V)_i = \begin{cases} 
\frac{(V_i - \chi_i) \cdot K_i}{1} & \text{if } \xi_i < 1 \\
1 & \text{if } \xi_i \geq 1 
\end{cases}
\]

(2.3)

where

- \( V_i \) - is the membrane potential
- \( K_i \) - is the gain and
- \( \chi_i \) - is the threshold voltage,

all of the \( i \)-th neuron.

These model neurons are connected by means of weighted synapses through which the firing of one neuron can cause currents to flow through the membrane of another cell. The output of the model neuron can be seen as a voltage rather than a firing frequency and therefore the weights can be thought of as resistors [5]. The passive RC properties of the network are used to simulate the properties of the cell membrane of the biological neuron [4].

One of the most significant differences between biological neurons and those typically used in ANN is the rich intrinsic dynamics of the biological neuron. Nerve cells contain a wide variety of active conductances which
gives them complex time-dependant responses to input and spontaneous activity. It is not possible to simulate these intrinsic properties, but to capture some of it, an intrinsic current is added to the model. These intrinsic currents may simply be static functions of membrane potential, or they can be described by their own differential equations.

The membrane potential of each model neuron may be described by the following differential equation:

\[ C_i \frac{dV_i(t)}{dt} = \sum_j S_{ji} F_j(V_{ji}) + \sum_k I_{ki}(V_{ki}, t) - V_i G_i \]  

(2.4)

where

- \( C_i \) is the membrane capacitance of neuron \( i \).
- \( V_i \) is the membrane voltage of neuron \( i \).
- \( S_{ji} \) is the synaptic weight between the \( j \)-th neuron and the \( i \)-th neuron.
- \( F_j \) is the output frequency of the \( j \)-th neuron.
- \( I_{ki} \) is the \( k \)-th intrinsic current of the neuron.
- \( G_i \) is the membrane conductance of the \( i \)-th neuron.

Networks of such neurons are represented by coupled sets of differential equations.

*Figure 2.4 The model neuron.*

---

**Neurophysiology**
2.6 Neural Network

The brain or neural network used to control the leg, which will be discussed in Chapter 4, is made up of several neurons which are derived from four basic types of neurons.

- **The Command neuron** - This neuron is the building block that forms the basis of all the other neurons in the network.
- **The Pacemaker neuron** - This neuron is capable of endogenously producing rhythmic bursting.
- **Motor Neuron** - The state of a given effector is a function of the firing frequency of its associated motor neuron.
- **Sensory neuron** - This type of neuron is modelled with intrinsic currents whose magnitude is a function of the intensity of a sensory input.

The model proposed by Beer et al [6] is of intermediate complexity between the M-P model and actual biological neurons. This model ignores the details of action potential generation and most complexities of synaptic and dendritic actions. Some input/output or transfer characteristics of the biological neuron are however employed. This model is most similar to Hopfield's continuous deterministic model, which includes the passive RC type characteristics of the cell membrane.

2.6.1 The Pacemaker Model

Evidence presented over the last two decades overwhelmingly supports the principle that the central nervous system does not require feedback to the higher cortex from sense organs in order to generate properly sequenced, rhythmic movement during repetitive behaviours such as locomotion.

Delcomyn [7] established that the central nervous system is intrinsically capable of providing the proper timing of muscle activation without requiring sensor feedback. That is, a single pacemaker neuron or a network of neurons, often called a neural "oscillator" or central pattern generator, is thought to be able to produce a repetitive rhythmic output. This output directly or indirectly drives the muscles used in the rhythmic behaviour in the proper sequence and the proper temporal relationships. The emphasis here is thus on a system that is automatic and independent of necessary, sensory feedback, although such feedback may modulate the intrinsic pattern.

The operation of these neurons is much like that of a unijunction transistor relaxation oscillator in that the bursting frequency is a function of the RC time constant of the charging and discharging capacitor [4].
Central pattern generators can be divided into two categories: those employing pacemaker cells and those employing network oscillators. Recall that a pacemaker cell is one that is capable of endogenously producing rhythmic bursts solely by virtue of their own intrinsic dynamics. Network oscillators, on the other hand, are networks of neurons which generate rhythmic patterns due to the synaptic interactions between their component neurons, none of which is capable of rhythmic activity in isolation.

The main advantage of using a pacemaker neuron instead of a network oscillator is that the behaviour of the pacemaker can easily be modified. This can easily be done by simply controlling the synaptic current inputs to the neuron. The frequency of the rhythmic bursting may, in other words, be controlled from some minimum value to a maximum value by gradually varying the synaptic current input. The oscillator can be reset simply by injecting a current pulse at the correct time.

A pacemaker cell exhibits the following characteristics:
- when it is sufficiently hyper polarized, it is silent.
- when it is sufficiently depolarized, it fires continuously.
- between these two extremes, it rhythmically produces a series of relatively fixed-duration bursts. The length of the interval between bursts is a continuous function of the injected synaptic current.
- a transient depolarization, which causes the cell to fire between bursts, can reset the bursting rhythm.
- A transient hyper polarization which prematurely terminates a burst can also reset the bursting rhythm.

These characteristics are produced in the model pacemaker by the control of two intrinsic currents, \( I_H \) and \( I_L \). \( I_H \) (the High current) is a depolarizing current which tends to pull the membrane potential above the threshold. \( I_L \) (the Low current) is a hyper polarizing current which tends to pull the membrane potential below the threshold.

These currents are manipulated, rather than being governed by differential equations, by the following rules:
- \( I_H \) is triggered whenever the cell membrane potential goes above the threshold value or \( I_L \) terminates. It remains active for a fixed period of time which determines the length of the burst.
- \( I_L \) is triggered whenever \( I_H \) terminates. It then remains active for a time period which is proportional to the steady state membrane potential.
- Only one of these currents may be active at any given time.

2.6.2 Sensory and Motor Neurons

Sensory and motor neurons serve as informational interfaces between the nervous system and the body in which it is embedded. These neurons have intrinsic currents whose magnitude is a function of the intensity of the physical stimulus to which its receptor is sensitive. The sensory neurons may be classified as follows:

---

Neurophysiology
Chemoreceptors: are sensory sensors and may be found in the antennae and mouth and are sensitive to the strength of the odour field at their location.

Mechanoreceptors: are receptive to some mechanical stimulus. The following mechanoreceptor neurons can be found:

- In the mouth: These are sensitive to the presence of food directly below the mouth.
- In the antennae: These are sensitive to the deflection of the tip of the antennae.
- In the legs: These neurons respond to the angular position of the leg with respect to the body.

Energy Sensors: are sensitive to the amount of energy that an insect possesses.

Most neuron outputs are normalised to the range [0, 1]. In the case of a sensor neuron, the output function is graded and not normalised to the range [0, 1]. This way an output is obtained which is proportional to the magnitude of the related stimulus.

The state of a given effector, such as the flexor or extensor muscle of the leg for example, is a function of the firing frequency of the motor neuron which controls it. To simplify the mechanical construction later on, the motor neurons are binary and normalised to the range [0, 1]. In this way a muscle is either contracting or relaxing at a maximum rate, with no intermediate state.

2.6.3 Compound Synapses

It might be necessary, sometimes, to allow one model neuron to modify the effect that another neuron has on a third. This can be done via compound synapse, in which one neuron synapses modifies the connection between two others.

There are two classes of compound synapses: **gaited synapses** and **modulated synapses**. A gaited synapse is one which can be switched on or off, or inverted, by currents delivered from other synapses or the controlling neuron. A modulated synapse is one whose gain can be continuously adjusted by other synaptic currents. Figure 2.5 illustrates a compound synapse schematically.

Both classes, gaited and modulated, of compound synapses involve modifications to the term in Equation 2.4 which represents the total synaptic current, namely:

\[ S_j F_j (V_j) \] (2.5)

For gaited synapses, this synaptic current takes the following form:
where $I_s$ represents the compound synaptic current, $I_e$ represents the standard synaptic current term, $U$ represents the ungaited state of synapses (either 1 for on, or 0 for off) and $I_g$ is the synaptic current from the gating synapse. This current from the gating synapse is, like a standard synapse, is equal to the product of the firing frequency of the gating neuron and the strength of the gating connection. The gating synaptic currents of the proper magnitude and sign can interact with $U$ to turn on, turn off or invert a gaited connection. In modulated synapse' this term takes the following form:

$$I_s = (U + I_g) I_s$$

(2.6)

where $I_s$ is the standard synaptic current term and $I_g$ is the synaptic current from the modulatory synapse as in the case with the gaited synapse. The essential idea is that a modulatory synapse which is excitatory can increase the gain of the modulated synapse by an amount proportional to the modulatory synaptic current, whereas a modulated synapses which is inhibitory can decrease the gain of the modulated synapse in a similar manner.

2.7 Summary

This chapter demonstrates the basic building blocks that can be used to construct an artificial neural network that is a simulation of some brain functions.
CHAPTER THREE
Walking Behaviour of Insects

In the beginning
GOD created......

3.1 Introduction

The scientific analysis of walking began more than 100 years ago as the result of a dispute about whether or not all four feet of a galloping horse are ever off the ground simultaneously [10]. Various techniques were used to try and find the solution and during this time more and more questions were asked about the walking behaviour of animals, insects etc.

Throughout this century there have been numerous studies of the nervous control of walking in amphibians and arthropods. Much of the early work on these animals was done at a time when the dominant concept in neurophysiology was that reflexes were essential for the generation of the stepping rhythm. Substantial evidence, recently, has accumulated demonstrating that this notion is incorrect and that the basic rhythm is generated within the central nervous system and is modified by sensory input [10].

To begin with, it is appropriate to define the basic mechanical requirements of the insect or any other self-supporting walking system. The legs have to perform four functions when an upright animal, which holds its body clear of the ground, walks on a horizontal surface. They must support the body, while holding it clear of small ground obstacles that would impede its progress. The legs must also stabilize the animal both along and across the body axis so that it does not fold or collapse. To achieve movement in the required direction the legs must provide leverage using those legs in contact with the ground. The legs must, finally, lift themselves from the ground in recovery strokes which place each leg in a favourable position to exert the next power stroke.

The study of walking behaviour in arthropods concentrates on the power and recovery stroke rather than stability and support. The reason why coordination is not mentioned as a fundamental requirement is because its importance appears to depend upon how many legs are available, their separation and orientation with respect to each other, and the load they have to support. As the number of legs is reduced the considerations of support...
begin to dominate the strategy of movement but where its constraint is relaxed the coordination of recovery strokes can be relaxed into a hazy alternating pattern [11]. The coordinated recovery and body stabilization relative to the surface become a vital consideration when the number of legs is reduced to six and the full body weight must be supported at all orientations of the walk surface to the gravity vector. If the number of legs is reduced further, then parts of the walk can no longer be considered to be in static equilibrium and it becomes necessary to provide a mechanism for dynamically stabilizing the body.

The insect, when walking upright, is always in static equilibrium and the leg system is designed to maintain the body in space with a suitable orientation relative to the substrate. The substrate is grasped by the sticky pads and claws of the tarsus and the legs provide the forces which counteract gravity and propel the animal in the required direction. In addition, the small number of legs requires that recovery strokes be organized in such a way that at least three legs, with a specific configuration, maintain contact with the substrate to hold the body in a suitable walking orientation.

3.2 Parameters of walking behaviour

One step cycle involves the relative movement of the segments of the leg and the movement of the leg as a whole unit. The process is a cyclic or repetitive one so that the pattern is apparent at all levels.

The cycle of movement in one leg is a function of the exact structure and orientation of the joints. Most walking behaviour can be resolved into a recovery, protraction, or swing phase in which the leg is lifted clear of the ground and swings forward to place itself in contact with the ground.

This forward position relative to the body, at the moment of touchdown, is referred to as the anterior extreme position (AEP) of the leg [12]. The leg may sometimes pause in this position before starting its rearward movement [13]. After such a pause the legs move to the rear in the support, retraction, or stance phase until it reaches the posterior extreme position (PEP) where it may pause again. At this stage the tarsal pads and claws are detached from the surface in preparation for lifting the leg in the next recovery stroke.

The step cycle for an individual leg thus consists of two phases: the stance phase, the part of a step from grasping the substrate at the AEP to the beginning of releasing the substrate at the PEP, and the swing phase, the part of the step from lifting the leg at the PEP to touching the substrate at the AEP. The propulsive force for progression is developed during the stance phase.

A common feature of the step cycle in most animals is that the duration of the swing phase remains comparatively constant as walking speed varies. Changes in speed of progressing are produced accordingly by
changes in the time it takes for the legs to be retracted during the stance phase [10].

In different insects the orientation and action of the leg vary considerably. In ants, flies and many insect nymphs the body is held clear of the walk surface and the plane containing the coxa, femur and tibia is approximately perpendicular to the walk surface. In the larger insects the abdomen is often dragged behind the legs. In the cockroach and beetles the plane of action may be rotated until it is almost parallel to the walk surface and most of the leg musculature contributes directly to the propulsive stroke [11].

3.3 Pattern of Leg Recovery

The gaits commonly seen in amphibians and arthropods resemble the walking and trotting gaits of cats. In the cockroach, which has six legs, the stepping of adjacent legs alternates during moderate-speed and high-speed walking. In other words the stepping of the three legs rear right, middle left and right front alternates with the stepping of left rear, right middle and left front. At all times, the two legs on any side of the insect strictly alternates. The animal is therefore always supported by at least three legs, and for that reason the gait is referred to as the tripod or triangle gait. As the walking speed decreases, the gait is described as a back to front sequence for the three legs on each side of the animal. The same is true for most other animals [10]. This is clearly seen in many-legged animals such as millipedes and centipedes, where "waves" of stepping appear from back to front.

3.4 Parameters Insects Regulate During Walking

As a general rule, an adult insect walking straight on a smooth, horizontal surface adopts the so-called alternating triangle gait. In this pattern of leg movements [11], [14], the front and rear legs on one side of the body, and the opposite middle leg, move forward and backward more or less as a unit, alternating their movements with those of the other triangle of legs. In most insects, this gait is used at all but the slowest speeds of progression. The first question that needs to be considered is, over which parameters of leg movements do insects have active control? It is these regulated parameters that the insect will be able to adjust when it encounters a problem as it walks.

Since the alternating triangle gait is defined by the time each leg begins its forward movement (protraction) relative to when other legs begin theirs, it follows that in order for an insect to use this gait over a variety of different walking speeds, it must actively regulate the timing of leg movements compared with one another. This is because each leg is physically capable of moving independently of all the others. The insect regulate many other parameters of walking, some as a direct consequence of the regulation timing, but others independently.

Thus, such readily measurable features of movements as where the tarsi are placed relative to the body [15], the angular displacement of the legs [16], and the time during which each leg is in contact with the walking surface [17] are all constant or vary predictably with the animal's speed of progression. In addition, such factors
as the height at which an insect keeps its body from the walking surface [18], are kept constant. Other factors, for example the angles through which leg segments move relative to other segments or the body, are probably regulated as well, but the factors listed above are those that have been examined so far, and attention will be restricted to them.

3.5 The Behavioural Effects of Interference with Walking

Knowing what parameters of walking can be regulated, it can now be considered how these parameters change when the insect encounters some "difficulty" or other. A number of experiments have been done in which the walking was interfered with, either by forcing the animal to walk over a nonsmooth or nonlevel surface (which "interferes" with normal walking by forcing the animal to adapt its leg movements to the surface) or by physically interfering with the movements of one or more legs. In most of these experiments, the "interference" is one that the insect may reasonably be expected to encounter in nature. The behaviour effects of these experiments tell how the animal adapts to these interferences.

This is discussed in terms of the results of four types of experiments:

- varying the terrain on which the insect walks,
- varying the load (weight) the insect must carry,
- physically obstructing the movements of one or more legs, and
- amputating one or more legs.

Each of these experiments presents a situation that the insect may encounter in nature. Natural walking surfaces are rarely smooth and level and the load an insect's legs must carry increases as the insect grows or when the insect lifts or carries an object. Further may twigs or other projections obstruct leg movement and legs may be lost because of accidents or attacks by predators.

3.5.1 Terrain

The experiment of varying the terrain over which the insect walk has been done in two different ways

- the slope of the surface on which the animal is walking is varied between horizontal and vertical and
- by presenting steps, ditches, barriers or any other surface irregularities to the insect.

How are the parameters of walking affected by forcing the insect to walk over irregular terrain? It is obvious and clear that an insect cannot flow over a barrier, or into and out of a ditch, but the strong tendency of an insect to maintain a fixed distance between its body and the walking surface is illustrated by some insects, such as the stick insect walking up or down a step or over a barrier or ditch. The body of this insect is long and flexible enough that each thoracic segment can bend somewhat relatively to the rest of the body, as the animal progresses over the obstacle the body tends to conform to it, bending up or down, segment by segment, over a step and dipping down into a ditch or arching over a barrier [18].

Walking Behaviour of Insects

Page 18
The legs as well as the body are affected by walking over varied terrain. Cruse [15] showed that stick insects walking over a ditch used the positions of anterior tarsi to guide the placement of the posterior ones.

Another question is how do the legs move when an insect is walking up or down an incline or vertical surface? This question has been examined in two different studies. In cockroaches walking up or down a 45° slope [17], the legs spend more time in contact with the slope than they would if the walking surface were horizontal. There is also a slight tendency for the front and middle legs to step forward a little earlier relative to the legs behind them than they normally would. The stick insect uses a different approach when walking up a vertical surface [16]. All the legs tend to be put down further forward on the surface relative to the body than normal, while they are still lifted from their normal positions. They move thus over a greater angle while in contact with the walking surface than they normally would and spend more time in contact with that surface as well.

3.5.2 Load

If the walking surface becomes nonhorizontal, the load on a walking insect increases. It has been shown by several researchers that at least some changes in leg movements that occur when an insect walks on an inclined surface do so as a direct consequence of the changed load conditions. A stick insect dragging a weight moves its legs farther forward than normal when walking [20].

Unloading the legs of a stick insect somewhat by suspending the animal over a surface of mercury does not affect the pattern of leg movement used by the insect and a little effect on the placement and points of lifting of the tarsi during walking, but it does tend to cause a faster walk [21]. Graham and Cruse [21] suggested that the speed of progression is determined in part by the resistance that the insect encounters to the forward movement of its body.

When the load on a free standing insect is varied, the effect of regulating the body height is also clearly seen. This has relatively little effect on the distance between the insect body and the surface.

3.5.3 Obstruction

Encountering a small obstacle poses no significant difficulty or problem for an insect. For example, Pearson and Franklin [19] reported that in locusts challenged with a barrier, one leg often struck the barrier as the insect started to climb over it. The response of the leg was additional elevation, which had the result of lifting the leg over the obstacle. When completely obstructing a leg so that it cannot complete its intended movement has interesting and dramatic effects on a walking insect. Experiments have been done on the stick insect and the effects of leg obstruction varies considerably. The most important factor was the position of the leg relatively to the body when it encountered the barrier.
The main response of the obstructed leg was an increase in the duration of its attempted forward movement. The earlier the leg encountered the obstacle in its forward movement, the longer it prolonged its attempts to move forward past the barrier. As long as the barrier was encountered after at least half the leg's normal forward movement had been completed, the leg continued to step once with each step of the remaining legs. However, more posterior positions often the barrier often caused the obstructed leg to continue its attempts to swing forward, while the remaining legs took several steps.

Adjacent legs were also affected by an obstacle. The result is an adjustment in the time and place from which the legs to the anterior of the obstructed leg were lifted from the walking surface. If the leg was prevented from moving very far forward by an obstruction early in the step cycle, the adjacent legs were positioned more posteriorly. If the obstruction was at the end of the forward swing phase, the adjacent legs were shifted to a much lesser extent.

3.5.4 Amputation
The loss of limbs is also a condition that might befall an insect in nature. The effect of this event has been investigated experimentally innumerable times during the last century. The removal of a leg has the most dramatic effect on an insect's walking behaviour. This is especially true if the amputated legs are the two middle ones. This situation presumably because of the loss of support on one side of the body if the insect were to continue to use the alternating triangle gait. The effect of such a double middle-leg amputation is a strong shift in gait from alternating triangle, where the rear and front legs move in synchrony, to a gait in which they alternate their movements. The intact legs tend to adjust their positions on the walking surfaces to compensate for the missing ones.

3.6 Sense Organs and the Effects of their Destruction
The preceding discussion focused on the behaviour responses of insects when they are challenged with irregularities in their normal walking pattern such as rough terrain, an increased load, an obstruction and if one or more legs are missing. To understand how an animal adapts to these irregularities, it is necessary to know which specific sense organs inform the animal that a normal movement may not be appropriate or even possible. Further it must be understood how the neural signals from these sense organs is used by the insect's central nervous system to adjust leg movements to the new circumstances. Physiological experiments address these kinds of questions [22].

3.6.1 Mechanoreceptors of the Legs
If one considers the variety of receptor types in, on and around and insect leg, there are only a few whose activities have any effect on walking. These are the campaniform sensilla, chordotonal organs, hair plates and
single hair sensilla [23]. The identity of these has not been established yet. There are, however, several physiological approaches that have been used to investigate the possible role of these sense organs.

3.6.2 Leg Reflexes and Walking
A number of reflexes have been described that are evoked by stimulation of one or more sense organs in a leg [24], [25]. The role of sense organs in walking have been studied by looking at the reflexes in restrained and dissected animals. These studies revealed very little that could be applied to a walking insect, because a reflex in animals, insects included, may be suppressed, modulated or changed in sign when an animal switches from one walking behaviour to another. Thus, any reflex effect shown to be present in a restrained preparation may be absent or have a different effect in a walking insect [22].

3.6.3 The Effects of Sense Organ Destruction
Destruction of individual sense organs in single legs has only minimal effects on leg movements. The effects that have been observed are often largely confined to the leg without any significant influence on the timing of steps in adjacent legs [22].

Results of behavioural experiments in which walking was interfered with, suggested that sensory input from the moving legs played an important role in the coordination of leg movements. This view seems a bit odd with the lack of any significant effects of the destruction of individual sense organs in the legs. The following two points must be kept in mind when interpreting the ablation experiments. At first, more than one sense organ may serve a particular function. The destruction of only one may therefore have very little effects because others can continue to carry out most of the function the ablated organ normally serves. It is, secondly, possible that centrally originating timing takes over in the absence of a specific sense organ that might ordinarily be providing timing cues. The loss of sense organs could happen because of the loss of a leg or damage to a leg by a predator or any other natural causes.

3.7 The Effect of Incorrect Sensory Input on Walking
What happen when a sense organ is stimulated at times during stepping when it is normally silent and made silent when it is normally active or if it is stimulated in such a way that it is active continuously rather than in timing with leg movements?

3.7.1 Inappropriately Timed Sensory Input
Experiments in which the femoral chordotonal organ has been induced to give an incorrect signal, has been done by carefully cutting the receptor apodeme of the organ from its anchor point in the dorsal.
When an insect walks slowly, the leg with the crossed receptor is used entirely normal. If the animal starts to move faster (which is accompanied by greater flexion and extension movements of the legs) the tibia of the operated leg suddenly extend and held up and out from the body in what is known as the "salute" position. This salute may be maintained for many steps of the other legs, but if the insect would slow down, the extended tibia may slowly relax and resume normal stepping. For the time the leg is held in the salute position, the remaining legs adopt the gait of an animal with one leg missing [27].

An explanation of this behaviour is that the outputs of the chordotonal organ normally signals that the tibia of a leg has reached a particular state of flexion and that it should therefore extend. During slow walking, the chordotonal organ is in approximately the same state of tension in the operated leg as it is in a normal leg, since the operated leg bends a little, it can still be used normally. As the leg is moved more vigorously, it is extended far enough that the repositioned chordotonal organ is strongly exited. The chordotonal organ in turn excites the tibial extensor muscles that would normally relieve the tension on the organ. In this case, tibial extension only serves to excite the chordotonal organ more strongly and the leg becomes locked into an extended position until the extensor muscle relaxes a bit.

It has also been shown that electrical stimulation of campaniform sensilla near the end of a step during walking in locusts will induce reflex excitation of leg extensor motor neurons, and after a slight delay, of flexor motor neurons as well [28]. Stimulation at other times had no effect.

3.7.2 Continuous Sensory Stimulation
When the chordotonal organ is permanently stretched it has an effect similar to that of crossing the receptor apodeme, except that the leg tends to be held up permanently, not just when the insect is walking briskly [29].

3.8 What Is the Role of Sense Organs in Walking?
The approximate timing of the two parts of each step, the initiation of the swing phase and the initiation of the stance phase, is established by a pattern generator, in common with other kinds of two-phase rhythmic movements. Sensory input that is superimposed on this centrally timed pattern is necessary to determine the proper time at which each leg is to start each swing and stance. This function can be served by input from one or more of four types of sense organs:
- the femoral chordotonal organ,
- campaniform sensilla,
- hair plates and
- perhaps the hair sensilla.
Without input from only one specific organ, such as a hair plate or chordotonal organ, the leg may move differently than it does normally [30], [31], as discussed above, but it is still able to contribute effectively to walking. The complete absence of input from all sense organs [32] does not destroy the ability of the leg to maintain its back and forth rhythm. This suggests that although sensory input is necessary to ensure the proper fine tuning of movements, without sensory input the leg can still be made to step at close to the proper time by the action of the central rhythm-generating network alone.

3.9 Conclusion

In conclusion insect walking is unusual among basic research topics in that there is a potential benefit in its study - the application of the specific coordinating mechanisms used by insects to the development of robotic, multilegged vehicles. The mechanisms used by insects to adapt to uncertainties of terrain is generally very useful in the developing process of a control system for a multilegged machine or robot.

In the following chapter the construction of a central control system to be used to control an autonomous robot is described.
4.1 Introduction

To design a legged machine or robot, one has to look at the simple organisms that have been roaming the earth for years with very simple but effective control mechanisms. The earth can be seen as a complex and often hostile environment. The control mechanisms used by these organisms are based on a system of distributed controllers, which are mainly responsible for the speed and robustness of the insect's locomotion controller.

The mathematical computations involved in the control of a legged robot are very complex. If one would use classical control methods to solve this type of control problem in real time, a very large computing system would be necessary.

4.2 Neural Basis of Locomotion in Biological Systems

Motion control in the lower class animals is highly decentralized. The high levels goals are formed in the brain, which then transmits these goals to the different control centres. There exists thus in biological systems a hierarchy of command levels. This is also applicable to the lower leg controllers, where different levels of control can be distinguished. Pearson's model [10] of a motion controller for a cockroach suggests that there are different levels of control as depicted in Figure 4.1.

At the centre of this model is a central pattern generator whose bursts of activity excite motor neurons responsible for the swing phase and exhibit those responsible for the stance phase. The pattern generator (flexor burst generator) is itself fed by higher brain centres. On the lowest level different sensory inputs are used to modulate the muscle activity generated by the pattern generator.
4.2.1 The Biological Step Cycle

As noted in Chapter 3 the basic rhythm for stepping pattern in each leg is generated within the central nervous system. In other words there exists within the lower levels of the nervous system networks of nerve cells that can generate the rhythmic sequence of electrical activity in flexors and extensors [10].

The existence of a central rhythm-generation for each leg does not mean that sensory input is unimportant in the patterning of motor activity. Sensory input, in fact, is essential if the animal is to be able to adapt its stepping movements promptly to compensate for irregularities in the terrain on which it is walking. For example, the stepping of the hind legs of a cat that has undergone either spinal or cerebral transection adapts to match the speed of a treadmill for a wide range of treadmill speeds [10].

"What happens in the lower control system of an insect during walking?"

The most basic components of walking are the rhythmic movements of each individual leg. These movements consist of a swing phase, in which the foot is up and the leg is swinging forward, and a stance phase, in which the foot is on the ground and the leg swinging back, propelling the body forward.

During walking, the transition from stance to swing is triggered when position-sense organs' signal that the leg has reached its extreme posterior position, and when campaniform sensilla reports a decrease in leg load, and when certain coordinating influences from other legs are present [10] (see Figure 4.1). Activation of the receptors that detect forces in the cuticle (the hard external skeleton) of the leg during the stance phase inhibits the flexor bursting generating system of interneurons and thereby preventing the switching from stance to swing.

---

**Figure 4.1 Pearson's model.**
As the leg is extended during the stance phase the stress in the cuticle diminishes, since the animals load or weight is carried by the other legs to a greater extent. This movement eliminates the inhibitory influence of the cuticle stress receptors from the bursting generator, with this enabling the system to become active and initiate the swing phase [10]. Two conditions seem thus to be necessary for the swing phase to be initiated, viz. the hip joint must be extended and the extensor muscles must be unloaded. Both conditions are fulfilled near the end of the stance phase, before the swing phase can be initiated.

The switching from swing to stance is also triggered by sensory input. A group of hair receptors is excited by flexion movements during the swing phase. The sensory inputs from these receptors inhibit the flexor motor neurons and the flexor bursting system. The latter effect terminates the flexor burst (and therefore the swing phase) and enables the leg to initiate the stance phase. The initiation of stance is also facilitated by an excitatory connection from the hair receptors to the motor neurons of the extensor muscle. The inhibitory pathway from the hair receptors to the flexor burst-generator ensures that the leg position at the end of the swing phase remains constant despite the position from which the step was initiated.

4.3 Proposed Leg Controller

Beer constructed an artificial leg controller based on the locomotion controller of the Pearson model, as shown in Figure 4.2. There is one command neuron, the leg controller neuron (LC), for the entire leg controller and a copy of the remainder of this circuit exist for each leg. The basic rhythmic movements of each leg are centrally generated by the pattern generator. Pearson’s unknown pattern generator circuitry has been replaced by a single pacemaker neuron (P). Each leg is controlled by three motor neurons. The stance and swing motor

![Figure 4.2 Leg Controller Circuit proposed by Beer [48].](image-url)
neurons determine the force with which the leg is swung forward or backwards respectively. Such forces will translate the body forward or backwards respectively depending on whether the foot is down or on the ground. The foot motor neuron controls the position of the foot. In Beer's specific application it is either up or down.

The foot neuron is normally active and the stance neuron is excited by the command neuron LC. This puts the foot down and pushes the leg back and so producing a stance phase. This state is periodically interrupted by a burst from the pacemaker neuron \( P \). This burst inhibits the foot and stance neurons and excites the swing motor neuron, lifting the foot and swinging the leg forward, in other words producing a swing phase. When this burst terminates, another stance phase begins. The rhythmic bursting in the pacemaker, \( P \), thus results in a basic swing/stance cycle that is required for walking.

The force applied during each stance phase, and the time between bursts in the pacemaker, depend upon the level of excitation supplied by the command neuron \( LC \). This basic design is based on the flexor-burst generator model found in the Pearson model.

To synchronise or properly time the transitions between the swing and the stance phases a need exists for sensory feedback information about the position of the legs. This information is provided by the back angle sensor (BAS) and the forward angle sensor (FAS).

These sensors serve to reinforce and fine-tune the centrally generated stepping rhythm. When a leg is all the way forward, the FAS encourages \( P \) to terminate its current burst by inhibiting it, thus initiating a stance phase. This sensor plays the same role as the hair receptors in Pearson's model, limiting the extent of a swing and enabling the switch to a stance phase. The BAS encourages \( P \) to initiate a swing by exciting it when a leg is all the way back. This sensor is related to the cuticle stress receptors in Pearson's model. The relation is however not a direct one.

The cuticle stress receptors are load sensors while the backward angle sensor is a position sensor. This simple approach seems to work satisfactorily even when no direct load feedback is provided. The leg controller also does not employ the load compensation reflex involving the cuticle receptors which means that the device will not be able to traverse uneven terrain successfully.

The RC characteristics of the neural model\(^1\) cause delays at the end of each swing before the next phase begins. This produced an unwanted 'jerky' walk. To smooth out this effect, inhibitory connections also exist from the

\(^1\) Chapter 2, *Neurophysiology*, paragraph 2.5, pp 8.

\[ \text{Autonomous Networks} \]
Central Pattern Control Level

Other control levels

Signal/Command to/from another hierarchy level.

a) Connections of the Central Pattern Controller

Central Pattern Control Level
Motion Pattern Control Level
Segment Control Level

Neurons:
LC - Locomotion Controller
BA - Backward Angle
FA - Forward Angle
ST - Stance
SW - Swing
P - Pace maker
FD - Foot Down

PG - Pattern Generator
SP - Signal Processing Neuron
ST - Stance Phase
SW - Swing Phase

b) Hierarchy levels of Control

Figure 4.3 Proposed leg controller for a three-degree-of-freedom robot leg.
FAS neuron to the swing neuron and the excitatory connections from the FAS to the stance and the foot neurons. This smooths the transition from a swing to a stance phase by giving a slight impulse in the direction of travel to begin a stance phase whenever the leg is all the way forward.

4.4 Leg Controller Model for a Three Degree of Freedom Leg

As stated earlier, in Chapter 1, we aim with this project to develop a neural controller that is able to control a robot leg with three degrees of freedom. The controller designed by Beer [6] needed some modification to perform this type of control. The hexapod developed by Beer, could only perform a two degree of freedom motion, and could cope, in theory, only with small irregularities in the terrain since no contact sensors are incorporated into the system. A biologically based three degree of freedom leg, with contact sensors, would overcome these limitations easily.

It has already been discussed that a normal step in walking has two different states that can be distinguished; viz. a stance and a swing phase. The coordination of the stance and swing phases is essential to the control activity that is arranged by the leg controller hierarchy. This control hierarchy must configure the three joints, i.e. the torso, hip and knee joints to ensure that the robot body moves in a straight horizontal line.

The complexity of the controller is increased by the requirement for coordination among the three motors. The need for a hierarchical type of controller is therefore essential. The three distinct levels of control can be identified in the proposed model, viz.

- central pattern control, the high level control of the leg as in the model developed by Beer,
- motion pattern control, the level where a pattern generator or neural network predicts the next state of the leg, and
- segment control, the control of the motors with a sensory feedback system.

Figure 4.3 shows a schematic representation of the connecting of the different hierarchy levels of control with each other.

The central pattern control level is similar to the controller developed by Beer except that this level of control activates lower control levels which is necessary to produce leg movement. Central pattern control is activated by the brain which cause the LC neuron to fire. The pacemaker (P) and stance (ST) neurons are then excited by the LC neuron. The excitation of the stance neuron causes the stance phase to occur. As in the model of Beer and Pearson the pacemaker fires periodically, interrupting the stance phase, and exciting the swing (SW) neuron which produces a swing phase. When the pacemaker's burst terminates, the stance phase commences, and in doing so, the cycle is completed. The rhythmic bursting of the pacemaker results in the basic stance/swing cycle required for walking.

---

Autonomous Networks
To properly synchronise the swing and the stance phases as in biological systems, the leg controller needs some feedback information about the exact position of the three segments of the leg. Another factor that needs to be known is the position of the foot either in the air or on the ground. To produce this information, two different sensor neurons are used to provide the necessary information to the central pattern controller; viz. the angle ($S$) and pressure ($F$) sensors. The angle sensors are used to locate the leg position. If the leg is at the end of the swing phase, the forward angle neuron is influenced by the shoulder sensor ($S$). This causes the forward angle neuron to excite the stance neuron and inhibit the backward angle neuron. When the leg is at the end of the stance phase, the shoulder sensor ($S$) influences the back angle neuron, which then excites the swing neuron and inhibits the forward angle neuron. These two factors, the leg position and the intrinsic properties of the pacemaker neuron, influence the duration of the pacemaker burst.

While the foot is on the ground, the pressure sensors determine the load carried by the foot and activate the foot down (FD) motor neuron. This ensures that the foot stays on the ground, during the stance phase, if the load on the leg is above a certain threshold. The FD sensor also inhibits the SW neuron preventing it from switching to a swing phase. This sensor corresponds to the hair receptors in the model of Pearson in Figure 4.1.

When the upper brainstem of a cat is transected, the animal can still stand but does not walk or run spontaneously. However, continuous electrical stimulation through electrodes in the cuneiform nucleus causes the cat to continue with a walking pattern [10], [33]. The same experiment has been done on the cockroach and the mantis [30]. The general conclusion of these kinds of experiments is that within the central nervous system there exists a hierarchically ordered series of connections that can initiate and control a programmed series of movements. In other words, some form of pattern generator exists in the lower level of leg controlling [10]. We decided to use this information to include a similar pattern generating system in our leg controller.

A few pattern generators were incorporated into the controller and were implemented in the form of homogeneous neural networks. When the stance or the swing neuron is activated, control is passed to a lower level of control, namely the motion pattern control. The motion pattern controller consists of a signal processing neuron and a neural pattern generator. It was necessary to teach the pattern generators a basic behaviour of walking. The teaching of the pattern generators will be discussed in depth in Chapter 6. The incorporation of pattern generators is biologically feasible as described in the previous paragraph.

Excitation of the swing or stance neuron causes that neuron to fire, and activate the applicable signal processing neuron. The signal processing neuron integrates the incoming signal from the stance or swing neuron. In hardware notation, the process of integration is performed by the RC-network of the cell membrane. If the swing or stance neurons fire, the conductance is low, ensuring that the cell membrane integrates sufficiently.
the swing or stance phase is completed and their signals terminated, the conductance is increased and this results in the hyperpolarizing of the cell. This change in conductance is biological consistent with its occurrence in biological neurons with gaited or active channels [34].

After the signal is integrated, it is passed to the pattern generators. The pattern generators then generate the different motor signals for the motors controlling the different leg segments, in order to produce the swing or stance phase.

The third level of control is the segment control and ensures the fluent motion of the different leg segments. To explain this control level considers the following. If a person would be asked to draw a straight horizontal line on a black board with his or her eyes closed. At first this seems to be difficult but most people have no difficulty in doing this accurately. The problem faced is with the eyes closed, how does one guide the hand, holding the chalk, to the right spatial locations. The problem is solved in a subconscious matter. Special sensory systems allow this kind of motor control although we are not aware of their internal operation. In other words biological systems have kinaesthetic and vestibular sensors that monitor segment position and loading. Most of the information generated by these sensors is used only at the local level, while some information is made available to other elements for processing at higher levels. Information from the sensors is used in the lower levels for momentary control. To achieve this control, a strict negative feedback closed loop controller is needed.

The time constant of the segment, in biological segment control, is much smaller than that of the neural controller. This may result in instability or oscillations at high input frequencies. Most neural controllers, especially on lower levels, have so-called Renshaw Cells [35]. At high frequencies, the Renshaw cell reduces the system gain by an inhibitory action on the segment controller. Biological systems therefore have a type of lower motor control. This motor control is implemented in our dc motor controller in the form of a phase-lead/phase-lag controller.

The duration of the walking cycle varies inversely with frequency [36]. In other words, higher step frequencies excite the $LC$ neuron to a higher firing rate. The result is greater excitation of the stance and pacemaker neurons. Thus the stance neuron is continuously activated and deactivated by the $LC$ and pacemaker neuron. This is due to the inherent characteristics of the control system. By implementing modulated synapse on the synapse between the forward angle sensor neuron and pacemaker neuron, the pacemaker neuron is deactivated during the stance at higher frequencies. A modulated synapse is therefore also applied to the synapse between the stance and swing neuron and the applicable signal processing neuron. This result is a faster throughput of the pattern.
4.5 Overview

A biologically feasible leg controller is constructed by looking at the control processes found in animals and insects. This information made it possible to construct a basic controller for a leg with three degrees of freedom. A basic controller but, however, the system becomes more complex if more degrees of freedom would be added to the system. If the robot should be expected to walk over uneven terrain another degree of complexity would be added.

The problem of walking in uneven terrain will be discussed in Chapter 6 as well as the necessary modifications that need to be made to this controller.
CHAPTER FIVE

Kinematics of Manipulation Robots

5.1 Introduction

Understanding how motions in general are produced enables us to design machines and other practical devices that move as we desire. The study of the relationship between the motion of a body and the causes for this motion is called dynamics. Dynamics is that branch of mechanics that deals with the motion of bodies under the action of forces. Dynamics has two distinct components - kinematics, which is the study of motion without reference to the forces which cause the motion, and kinetics, which relates the action of forces on bodies to their resulting motions. The factors that dictate the physical make-up of an articulated robot, both from a theoretical and practical point of view, are considered in this chapter.

5.2 Degrees of freedom of a solid

Consider a solid placed in real three-dimensional space (Figure 5.1). This solid possesses six degrees of freedom (DOF), three for translation and three for rotation. If an orthonormal trihedral is considered, which is centred on the centre of gravity of the solid, the six DOF can be expressed by the three axes of the trihedral so they perform a translation. The other three DOF can be 'used' to perform change in orientation, ie. when producing rotations with reference to the three axes of the trihedral.

5.3 Description of mechanical systems

To achieve computerized control of an articulated mechanical system (AMS), it is necessary to start with a model of such a system (see Figure 5.2).

![Figure 5.1](https://via.placeholder.com/150)

*Figure 5.1 The six DOF of a solid: three translational (T₁, T₂, T₃) and three rotational (R₁, R₂, R₃).*

Kinematics
The description is usually described in terms of characteristic variables which are specific to the system and include DOF, lengths, mass, inertias, etc. The number and nature of these parameters vary and depend on whether the model is dynamic, kinematic or geometric. Together these variables constitute the AMS. The aim of this description is two-fold:

- To be able to use the description for the software models.
- To be able to use this description to compare manipulators in order to classify them.

There exists a number of description methods such as the geometrical representation that uses standard schematic geometrical representations:

- the Roth-Pieper [37] description assumes that the joint between two segments has only one DOF,
- Khalil's [38] description has been deliberately constructed to allow the use of a computer, adapted to the generation of dynamic models of articulated systems,
- the aims of the Renaud-Zabala [39] description are similar to those of Khalil's. This description is symbolic and it is not possible to give numerical values to the variables concerned and
- Borrel's [40] description.

The latter form the basis of all the analysis that will follow and will be discussed in further detail.

5.3.1 Borrel's Description

Borrel adhered to a linguistic description that is appropriate to any segment of an articulated mechanism in which each component is activated, relative to its neighbours, by rotational or translational movement. For example, one of the components, \( C_0 \) (the base), can be placed in an orthogonal set of coordinates \( (R_0) \).

---

**Kinematics**
5.3.2 Association between a graph and a mechanism

Borrel associated a directed graph with a mechanism. In this, the nodes are the intermediate coordinates between $R_n$ and $R_0$, and the connecting arcs represent transfer from one coordinate set to another. (See Figure 5.3) Only two types of coordinate axes are used:
- rotation of a segment about one of the axes of the preceding segment.
- translation of a segment along one of the axes of the preceding segment.

5.3.3 Definition of a chain

A chain can be defined as the part of a mechanism corresponding to a graph for which all the nodes except the first and the last are nondivergent. A chain is a succession of coordinate transformations associated with the arcs of the graph. Then

$$HK = P_1P_2 \ldots P_n \ldots P_n$$

is the chain described by $n$ transformations.

A robot is essentially a functional machine. Since the DOF of one manipulator is not always equal to that of another, it is necessary to classify robots in the form of a hierarchy, in terms of the DOF of the manipulator, the arm and the end effector. The end effector is that part of a robot which physically performs the task and, the longer the arm, the greater the effective movement at the extremity. The execution of a task is thus dependent on the DOF. The DOF and their description could be a principal factor in the development of computer-aided simulation and modelling of AMS.
5.4 Coordinate Transformations

5.4.1 Introduction

The discussion in this section will be concerned with one of the basic problems in manipulator motion synthesis, namely computing of the manipulator position in the work space, when given a vector of joint coordinates. The main objective of this analysis is to generate a transformation matrix between an external coordinate set and the manipulator end-effector. The methods that are looked into in order to maintain such a transformation matrix is that of Denavit and Hartenberg [41], which is also used by Paul [42] and Vukobratovic [43], and the general method of homogeneous coordinate transformation described and used by Coiffet [44].

5.4.2 Homogeneous Transformations

A serial link manipulator consists of a sequence of mechanical links connected together by actuated joints. Such a structure forms a kinematic chain and may be analyzed by methods developed by Denavit and Hartenberg [41]. The results of this analysis are the matrix equations expressing manipulator end-effector Cartesian position and orientation in terms of the joint coordinates.

For an n-degree-of-freedom manipulator, there will be n links and n joints. Consider a simple open kinematic chain with n links. Each link is characterized by two dimensions: the common normal distance $a_n$ (along the common normal between axes of joint $n$ and $n+1$), and the twist angle $\alpha_n$ between these axes in the plain perpendicular to $a_n$. It is customary to call $a_n$ the "length" and $\alpha_n$ the "twist" of the link (see Figure 5.4).

Generally, two links are connected at each joint axis. Each joint axes has two normals to it $a_n$ and $a_n$ (see Figure 5.5). The relative position of two such connected links is given by $d_n$, the distance between the normals along the axis of joint $n$, and $\theta_n$ the angle between the normals measured in a plane normal to the axis. $d_n$ and $\theta_n$ is called the "distance" and the "angle" between the links, respectively.

![Figure 5.4 Length a and twist angle $\alpha$ of a link.](image-url)
In order to describe the relationship between links, coordinate frames will be assigned to each link. Revolute joints will be considered first in which $\Theta_n$ is the joint variable. The origin of the coordinate frame of link $n$ is set to be at the intersection of the common normal between joints $n$ and $n+1$. In the case of intersecting joint axes, the origin is set to be at the point of intersection of the joint axes. If the axes are parallel, the origin is chosen to make the joint distance zero for the next link whose coordinate origin is defined. The axes of the link coordinate system $O_{n}X_{n}Y_{n}Z_{n}$ are to be selected in the following way. The $z$ axis of system $n$ should coincide with the axis of joint $n+1$, about which rotation $\Theta_{n+1}$ is performed. The $x$ axis will be aligned with any common normal which exists (usually the normal between axes of joint $n$ and $n+1$) and is directed from joint $n$ to joint $n+1$. In the case of intersecting joint axes, the $x$ axis is chosen to be parallel or antiparallel to the vector cross product $Z_{n+1} \times Z_n$. The $y_n$ axis satisfies $X_n \times Y_n = Z_n$. It must be noticed that this condition is also satisfied for the $x$ axis directed along the normal between joints $n$ and $n+1$.

The joint coordinate $\Theta_n$ for a revolute joint is now defined as the angle between axis $x_{n+1}$ and $x_n$ (Figure 5.5). It is zero when these axis are parallel and have the same direction. The twist angle $\alpha_n$ is measured from axis $z_n$ to $z_{n+1}$ i.e. as a rotation about $x_n$ axis.

Let us now consider prismatic joints (Figure 5.6) where the distance $d_n$ is the joint variable. The direction of the joint axis is the direction in which the joint moves. Although the direction of the axis is defined, its position in space is not defined. In the case of a prismatic joint the length $a_n$ has no meaning and is set to zero. The origin of coordinate system corresponding to the sliding joint $n$ is chosen to coincide with the next defined link origin. The $z$ axis of the prismatic link is aligned with the axis of joint $n+1$.

---

**Figure 5.5** Link parameters $\Theta$, $d$, $a$ and $\alpha$
The \( x_n \) axis is chosen to be parallel or antiparallel to vector cross product of the direction of the prismatic joint \( z_n \). The zero position, with \( d_n = 0 \), will be defined when \( x_{n-1} \) and \( x_n \) intersect. For a prismatic joint the angle \( \theta_n \) between axis \( x_{n-1} \) and \( x_n \) is fixed and represents a kinematic parameter together with the twist angle \( \alpha_n \).

The origin of the reference or base coordinate system is set to coincide with the origin of the first link. If it is desired to define a different reference coordinate system then the relationship between the reference and base coordinate systems can be described by a fixed homogeneous transformation [45].

5.4.3 Homogeneous transformation matrices

Having assigned coordinate frames to all the links according to the preceding scheme, the relationship between successive links \( n-1 \) and \( n \) can be now establish in the form of transformation matrices between link coordinate systems and distance vectors between their origins. The homogeneous matrices involve both the information in the form of 4x4 matrices.

Denote \( A_n \) the homogeneous transformation relating between the coordinate system of link \( n \) and the coordinate frame of \( n-1 \). It can be represented in the form
\[
A = \begin{bmatrix}
A_{11} & A_{12} & A_{13} & P_1 \\
A_{21} & A_{22} & A_{23} & P_2 \\
A_{31} & A_{32} & A_{33} & P_3 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(5.2)

where \( A \) is the matrix which transforms vectors from system \( n \) into system \( n-1 \), and \( P \) is the distance vector between origins of systems \( n-1 \) and \( n \), expressed with respect to system \( n-1 \). Since \( T \) describes rotation between these systems it is sometimes referred to as the rotation matrix or the orientation matrix.

The transformation of system \( n-1 \) into system \( n \) can be described by the following set of rotations and translations:

- rotation about \( z_{n-1} \) by an angle \( \theta_n \),
- translation along \( z_{n-1} \) by a distance \( d_n \),
- translation along rotated \( x_{n-1} = x_n \) by a distance \( a_n \) and
- rotation about \( x_n \) by the twist angle \( \alpha_n \).

This sequence of rotations and translations can be presented as a product of the following homogeneous transformation matrices

\[
T_{i-1}^{-1} = \begin{bmatrix}
\cos \theta_n & -\sin \theta_n & 0 & 0 \\
\sin \theta_n & \cos \theta_n & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & a_n \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d_n \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \alpha_n & \sin \alpha_n & 0 \\
0 & \sin \alpha_n & \cos \alpha_n & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(5.3)

This relation yields the homogeneous transformation matrix between two successive coordinate systems \( n \) and \( n-1 \) for a revolute joint:

\[
T_{i-1}^{-1} = \begin{bmatrix}
\cos \theta_n & -\sin \theta_n & \cos \alpha_n & \sin \alpha_n & a_n & \cos \theta_n \\
\sin \theta_n & \cos \theta_n & -\sin \alpha_n & \cos \alpha_n & d_n & 0 \\
0 & \sin \alpha_n & \cos \alpha_n & 0 & 0 & 1
\end{bmatrix}
\]  

(5.4)

For a prismatic joint it becomes:

\[
T_{i-1}^{-1} = \begin{bmatrix}
\cos \theta_n & -\sin \theta_n & \sin \alpha_n & 0 \\
\sin \theta_n & \cos \theta_n & -\cos \alpha_n & 0 \\
0 & \sin \alpha_n & \cos \alpha_n & d_n \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(5.5)
Having obtained the homogeneous transformation matrices between successive coordinate frames, $T_i, i=1,\ldots,n$, one can easily obtain the homogeneous transformation between any two systems $j$ and $k, k\leq j$, from

$$T_j^k = T_{k+1}^{-1} \cdots T_{j+1}^{-1}, \quad 0 \leq k \leq j \leq n. \quad (5.6)$$

It gives information on the rotation between systems $j$ and $k$ and the distance vector between their origins. For $k=0$ and $j=n$ this matrix gives the Cartesian coordinates of the manipulator tip and information on hand orientation.

$$T_n^0 = \begin{bmatrix} x \\ A_n^0 \\ y \\ z \\ 0 \ 0 \ 0 \ 1 \end{bmatrix} \quad (5.7)$$

with respect to the base reference coordinate frame.

### 5.4.4 Solving the Kinematic Equations

Once coordinate frames have been assigned to a manipulator it is possible to obtain the Cartesian position and orientation of the manipulator end-effector when given the joint coordinates. Assume for the discussion that a manipulator with six degrees-of-freedom are used.

The description of the end of the manipulator, link coordinate frame 6, with respect to the link coordinate frame $n-1$ is given by $U_n$ where

$$U_n = A_n \cdot A_{n-1} \cdot A_{n-2} \cdots A_6. \quad (5.8)$$

The end of the manipulator with respect to the base, known as $T_6$, is given by $U_6$:

$$T_6 = U_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6. \quad (5.9)$$

If the manipulator is related to a reference coordinate frame by a transformation $Z$ and has a tool attached to its end described by $E$, the description of the tool with respect to the reference coordinate system described by $X$ as follows (5.8):

$$X = Z \cdot T_6 \cdot E. \quad (5.10)$$

In order to control the manipulator the inverse problem must be looked at, that is, given $X$ in (5.10), what are the corresponding joint coordinates?
\( T_6 \) may be obtained first as
\[
T_6 = Z^{-1} \cdot X \cdot E^{-1}
\] (5.11)

and the traditional approach is then to solve the matrix equation
\[
T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6
\] (5.12)

where \( T_6 \) is given numeric values. If \( T_6 \) is known, the required values of \( \theta_n, \theta_p, \theta_o, \theta_p, \) and \( \theta_o \) can be obtained by simultaneously solving the equations \( U_1, U_2, \ldots, U_6 \). This approach is quite difficult for the following reasons:
- the equations are transcendental,
- both the Sine and Cosine will be needed in order to determine the angles uniquely and accurately,
- the manipulator exhibits more than one solution for a given position and
- there are twelve equations and six unknowns.

Six other matrix equations, however, can be obtained by successively premultiplying (5.11) by the \( A \) matrix inverses:

\[
A_1^{-1} \cdot T_6 = U_2
\] (5.13)
\[
A_2^{-1} \cdot A_1^{-1} \cdot T_6 = U_3
\] (5.14)
\[
A_3^{-1} \cdot A_2^{-1} \cdot A_1^{-1} \cdot T_6 = U_4
\] (5.15)
\[
A_4^{-1} \cdot A_3^{-1} \cdot A_2^{-1} \cdot A_1^{-1} \cdot T_6 = U_5
\] (5.16)
\[
A_5^{-1} \cdot A_4^{-1} \cdot A_3^{-1} \cdot A_2^{-1} \cdot A_1^{-1} \cdot T_6 = U_5
\] (5.17)

The matrix elements on the left sides of these equations are functions of the elements of \( T_6 \) and of the first \( n-1 \) joint variables. The matrix elements on the right hand sides are either zero, constants or functions of the \( n \)th to 6th joint variables. As matrix equality implies element by element equality we obtain 12 equations from each matrix equation. Equating elements of these matrix equations frequently results in equations yielding joint variables explicitly.
5.5 Summary

This chapter demonstrates the main mathematical relations and definitions that describe the manipulator end-effector position and orientation. The approach used, uses homogeneous transformations with Denavit-Hartenberg kinematic notation of open kinematic chains. The link coordinate frames are, from a kinematic point of view, centred and orientated arbitrarily with respect to the joint centers and axes. The inverse kinematic problem is also considered: when given the position and orientation of the manipulator end-effector the joint coordinates can be obtained by analytic methods.
CHAPTER SIX
Implementation

6.1 Introduction
To implement the hexapod leg it was decided to start from the mathematical trajectory description and implement all the necessary neural pathways as each level was completed. The kinematic equations for the leg was developed as a start to the implementation of the hexapod motion controller. Different parameters for each of the segments of the leg, were then determined. A mathematical simulation of the kinematic equations then followed to obtain the different angles of the three joints, i.e. the torso, hip and knee as a function of leg position. These values were then used to construct pattern generators for the stance and swing phase. These pattern generators were then implemented by multilayer perceptrons composed of artificial neural networks.

The trained networks were then used to obtain the pattern necessary for walking on an horizontal surface. These networks were then further extended to obtain the pattern necessary if the robot were to step on an object or step in a hole in the ground. The different networks and controllers were then connected to each other in order to simulate a leg with three degrees of freedom.

6.2 Calculation of the Kinematic Equations
The aim of this analysis is to compute the necessary kinematic equations to control the leg shown in Figure 6.1. This is done by defining coordinate systems, for each of the leg segments, that coincides with the conditions outlined in Chapter 5.

Figure 6.1 Schematic representation of the robot leg.
The transformation matrices from one coordinate system to another are the following:

Rotation about the X-axis by an angle $\theta$:

$$\text{Rot (X,$\theta$)} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (6.1)

Rotation about the Y-axis by an angle $\theta$:

$$\text{Rot (Y,$\theta$)} = \begin{bmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (6.2)

Rotation about the Z-axis by an angle $\theta$:

$$\text{Rot (Z,$\theta$)} = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (6.3)

Translation along the X-axis by distance $a$:

$$\text{Trans (X,$a$)} = \begin{bmatrix}
1 & 0 & 0 & a \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (6.4)

Translation along the Y-axis by distance $a$:

$$\text{Trans (Y,$a$)} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & a \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (6.5)

Translation along the Z-axis by distance $a$:

$$\text{Trans (Z,$a$)} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & a \\
0 & 0 & 0 & 1
\end{bmatrix}$$  \hspace{1cm} (6.6)

\textbf{Implementation}
These six transformations form the basis of the analysis that will follow. The derivation of the transformations is given in Appendix A1.1.

The robot leg consists of three segments and has three degrees of freedom, viz. the first segment nearest to the robot body that rotates by an angle \( \theta \), the second segment which hinges on the first segment by an angle \( \phi \) and the third segment which hinges on the second segment by an angle \( \psi \). In order to determine the position of the foot with respect to the robot body, it is necessary to assign transformation matrices to each of the segments.

The transformation matrix for the first joint is found by transferring the coordinate system \( X_0Y_0Z_0 \) to \( X_1Y_1Z_1 \). See Figure 6.2a. This means a rotation, \( \theta \), around the \( Z_0 \)-axis, a translation, \( d_1 \), along the \( Z_0 \)-axis, a translation, \( a_1 \), along the \( X_0 \)-axis and a rotation of \( a_1 \) around the \( X_0 \)-axis. In other words by multiplying Equations (6.3) \((\theta=\theta), (6.4) (a = d_1), (6.4) (a = a_1) \) and \((6.1) (\theta = a_1 = \pi/2)\). The result is the following:

\[
T_1^0 = \begin{bmatrix}
\cos \theta & 0 & \sin \theta & a_1 \cdot \cos \theta \\
\sin \theta & 0 & \cos \theta & a_1 \cdot \sin \theta \\
0 & 1 & 0 & d_1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(6.7)

The coordinate system \( X_1Y_1Z_1 \) needs to be transferred further to the \( X_2Y_2Z_2 \) coordinate system. See Figure 6.2b. This is done by a rotation, \( \phi \), around the \( Z_1 \)-axis and a translation, \( a_2 \), along the \( X_1 \)-axis. The result is obtained by multiplying Equations (6.3) \((\theta = \phi)\) and (6.4) \((a = a_2)\) and is given by

\[
T_2^1 = \begin{bmatrix}
\cos \phi & -\sin \phi & 0 & a_2 \cdot \cos \phi \\
\sin \phi & \cos \phi & 0 & a_2 \cdot \sin \phi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(6.8)

The transformation from the second coordinate system, \( X_2Y_2Z_2 \) to the third coordinate system, \( X_3Y_3Z_3 \) also requires both a rotation and a translation (Figure 6.2c). Rotation, \( \psi \), around the \( Z_2 \)-axis and a translation, \( a_3 \), along the \( X_2 \)-axis is given mathematically by multiplying Equations (6.3) \((\theta = \psi)\) and (6.4) \((a = a_3)\). The result is the following

\[
T_3^2 = \begin{bmatrix}
\cos \psi & -\sin \psi & 0 & a_3 \cdot \cos \psi \\
\sin \psi & \cos \psi & 0 & a_3 \cdot \sin \psi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(6.9)

The same results could have been obtained by substituting the values for \( a_\alpha, \theta, \phi, \psi, a_\alpha, a_\phi, a_\psi \) and \( d_\alpha \) into Equation (4.4) for each segment.

---

**Implementation**
Figure 6.2 Transformations between coordinate systems.
Having obtained the homogenous transformation matrices between the successive coordinate frames the homogenous transformation matrix between systems 0 and 3 can be obtained by using Equation 4.6. In other words by multiplying Equations (6.7), (6.8) and (6.9) successively. The elements of this matrix can be represented as follow:

\[
A_3^0 = \begin{bmatrix}
A_{11} & A_{12} & A_{13} & P_x \\
A_{21} & A_{22} & A_{23} & P_y \\
A_{31} & A_{32} & A_{33} & P_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(6.10)

where

\[A_{11} = \cos \theta \cos (\phi + \psi) \]  

(6.11)

\[A_{12} = -\cos \theta \sin (\phi + \psi) \]  

(6.12)

\[A_{13} = \sin \theta \]  

(6.13)

\[P_x = \cos \theta \left( a_1 + a_2 \cos \phi + a_3 \cos (\phi + \psi) \right) \]  

(6.14)

\[A_{21} = \sin \theta \cos (\phi + \psi) \]  

(6.15)

\[A_{22} = -\sin \theta \sin (\phi + \psi) \]  

(6.16)

\[A_{23} = \cos \theta \]  

(6.17)

\[P_y = \sin \theta \left( a_1 + a_2 \cos \phi + a_3 \cos (\phi + \psi) \right) \]  

(6.18)

\[A_{31} = \sin (\phi + \psi) \]  

(6.19)

\[A_{32} = \cos (\phi + \psi) \]  

(6.20)

\[A_{33} = 0 \]  

(6.21)

\[P_z = d_1 + a_2 \sin \phi + a_3 \sin (\phi + \psi) \]  

(6.22)

The transformation matrix, Equation (6.10), shows the transformation matrix that is necessary to obtain the position and orientation of the robot foot with respect to the robot body.

---

**Implementation**
6.3 The Inverse Kinematic Problem

If the joint coordinates are given the position and orientation of the robot foot can be obtained by using Equation (6.10). In other words the position and orientation of the robot foot with respect to the robot body's coordinate frame can now be calculated. For example assume the angles has the following values, $\theta = 90^\circ$ and $\psi = \phi = \theta^\circ$. This gives the maximum extension of the robot leg as shown in Figure 6.3.

![Figure 6.3 Extreme values of the robot leg.](image)

In order to control the robot foot, we are interested in the inverse kinematic problem, that is, given $X$ in Equation (6.10) what are the corresponding joint coordinates? Since we are dealing with a three degree-of-freedom robot leg only the position $P_x, P_y, P_z$ in Equation (6.10) is known. The joint coordinate values $\theta, \phi$ and $\psi$ must be determined by the following equations:

\[
\begin{align*}
(T_2^1)^{-1} \cdot (T_1^0)^{-1} \cdot T_3^0 &= T_3^1 \quad (6.23) \\
(T_1^0)^{-1} \cdot T_2^0 &= T_1^1 \quad (6.24)
\end{align*}
\]

The elements of matrices on the left hand sides of the above equations depend on the elements of $T_2^0= T_1^0$ matrix and the joint coordinates $\theta$ and $\phi$, while the matrices on the right hand sides depend on the remaining joint coordinate $\psi$. The analytical solution for the joint coordinates is obtained in a useful form by equating the elements of matrices on the left and right-hand sides of these equations. Computing Equation (6.23) we find:

\[
\begin{bmatrix}
\cos \theta & \sin \theta & 0 & -a_1 \\
0 & 0 & 1 & 0 \\
-\cos \theta & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
A_{11} & A_{12} & A_{13} & P_x \\
A_{21} & A_{22} & A_{23} & P_y \\
A_{31} & A_{32} & A_{33} & P_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos(\phi+\psi) & -\sin(\phi+\psi) & 0 & a_2 \cos(\phi+\psi) + a_2 \cos \phi \\
\sin(\phi+\psi) & \cos(\phi+\psi) & 0 & a_2 \sin(\phi+\psi) + a_2 \sin \phi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
= 
\begin{bmatrix}
\cos \theta & \sin \theta & 0 & -a_1 \\
0 & 0 & 1 & 0 \\
-\cos \theta & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos(\phi+\psi) & -\sin(\phi+\psi) & 0 & a_2 \cos(\phi+\psi) + a_2 \cos \phi \\
\sin(\phi+\psi) & \cos(\phi+\psi) & 0 & a_2 \sin(\phi+\psi) + a_2 \sin \phi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Now, we are looking for the constant elements on the right-hand side of Equation (6.25) to correspond to a function of the unknown angle on the left-hand side. Equating elements [3,4] on both sides we find

\[ P_x \sin \theta - P_y \cos \theta = 0 \]  

(6.26)
yielding

\[ \theta = \arctan \left( \frac{P_y}{P_x} \right) \]  

(6.27)

The next joint coordinate should be obtained by premultiplying the next transformation i.e. Equation (6.24). In this case the equation would not yield the solution for \( \phi \), since the axis of the next two joints are parallel.

The elements [1,4] and [2,4] on the left and right-hand sides of Equation (6.25) will be equated. That is

\[ P_x \cos \theta + P_y \sin \theta - a_1 = a_2 \cos(\phi) + a_3 \cos(\phi + \psi) \]  

(6.28)

\[ P_x - d_1 = a_2 \sin \phi + a_3 \sin(\phi + \psi) \]  

(6.29)

Since the angle \( \theta \) has already been determined the left hand sides of the above equations are known. By squaring both equations and then adding we obtain

\[ \cos \psi = \frac{(P_x \cos \theta + P_y \sin \theta - a_1)^2 + (P_x - d_1)^2 - a_2^2 - a_3^2}{2a_2a_3} \]  

(6.30)

This corresponds to the cosine theorem. The \( \sin \psi \) variable is obtained from

\[ \sin \psi = \pm \sqrt{1 - \cos^2 \psi} \]  

(6.31)

The signs + and - correspond to the elbow's up and down configuration. The angle \( \psi \) is given by

\[ \psi = \arctan \left( \frac{\sin \psi}{\cos \psi} \right) \]  

(6.32)

Angle \( \phi \) is now obtained from Equation (6.26) as

\[ \sin \phi = \frac{(P_x - d_1)(a_2 + a_3 \cos \psi) + (a_2 \sin \psi)(a_1 - P_x \sin \theta - P_y \sin \theta)}{a_2^2 + a_3^2 + 2a_2a_3 \cos \psi} \]  

(6.33)

\[ \cos \phi = \frac{(P_x \sin \theta + P_y \sin \theta - a_1)(a_2 + a_3 \cos \psi) + (a_2 \sin \psi)(P_x - d_1)}{a_2^2 + a_3^2 + 2a_2a_3 \cos \psi} \]  

(6.34)

---

**Implementation**
\[
\phi = \arctan \left( \frac{(P_z - d_1)(a_2 + a_3 \cos \psi) + (a_2 \sin \psi)(a_1 - P_z \sin \theta - P_y \sin \theta)}{(P_z \sin \theta + P_y \sin \theta - a_1)(a_2 + a_3 \cos \psi) + (a_2 \sin \psi)(P_z - d_1)} \right)
\] (6.35)

It is now possible to construct a mathematical simulation for the robot leg, with the equations known for each of the three angles, \(\theta\), \(\phi\) and \(\psi\).

### 6.4 Parameters of the Robot Leg

The different parameters for the robot leg is shown in Figure 6.4, \(a_1 = 50\text{mm}\), \(a_2 = 150\text{mm}\), \(a_3 = 300\text{mm}\) and \(d_1 = 175\text{mm}\). As a result of this configuration the angle \(\gamma\) that is formed between segment 3 and the ground is 69° and will therefore, not cause the leg to slip away from the body during the stance phase on a nonslippery surface.

### 6.5 Mathematical Simulation

A short program was written in Mathematica Version 2.2 to simulate the three phases of a step namely:
- the stance phase
- the first part of the return swing phase and

![Figure 6.4 Parameters of the robot leg.](image)
the second part of the return phase swing.

The two parts of the return swing is a result of the programming method. (The program written in Mathematica is shown in Appendix A1.2.1)

The body of the robot is held fixed while the leg is moved, during the simulation. The length of a step is taken as 400mm and the leg is said to move in a straight line (in fact it means the foot is on the ground and the body is propelled forward in a straight line), during the stance phase, 250mm from the body. During the swing phase the leg lifted upwards, extended to 290mm and placed on the ground again.

6.5.1 Stance Phase

The stance phase was simulated first and the initial and final conditions are as follow:
- \( P_x \text{ Begin} = 250.00 \) \( P_x \text{ End} = 250.00 \)
- \( P_y \text{ Begin} = 200.00 \) \( P_y \text{ End} = -200.00 \)
- \( P_z \text{ Begin} = 0.00 \) \( P_z \text{ End} = 0.00 \)

Equations (6.27) and (6.30-6.35) were then used to calculate the three different angles \( \theta \), \( \phi \) and \( \psi \). The result is shown in Figure 7.1.

6.5.2 The Swing Phase

The initial and final conditions for the first part of the swing phase are:
- \( P_x \text{ Begin} = 250.00 \) \( P_x \text{ End} = 275.00 \)
- \( P_y \text{ Begin} = -200.00 \) \( P_y \text{ End} = 0.00 \)
- \( P_z \text{ Begin} = 0.00 \) \( P_z \text{ End} = 100.00 \)

During the first part of the swing phase the leg is simultaneously lifted upwards, to a maximum of 100mm, and extended a further 25mm from the body. For the second part of the swing the initial and final conditions are:
- \( P_x \text{ Begin} = 275.00 \) \( P_x \text{ End} = 250.00 \)
- \( P_y \text{ Begin} = 0.00 \) \( P_y \text{ End} = 200.00 \)
- \( P_z \text{ Begin} = 100.00 \) \( P_z \text{ End} = 0.00 \)

Again Equations (6.27) and (6.30-6.35) were used to calculate the three different angles \( \theta \), \( \phi \) and \( \psi \). The result is shown in Figure 7.3.
6.6 Construction of the Pattern Generators

Artificial Neural networks are rough models of the biological basis of the mental process their name implies. Because of their substantial parallelism, they can process information and carry out solutions almost simultaneously. They learn by being shown examples and the expected results. Or, they form their own associations without being prompted and rewarded. A homogenous neural network was used in this project to implement the pattern generators. These pattern generators are in fact adaptive predictors.

6.6.1 Adaptive Prediction

The basic idea of training a network to be a predictor is illustrated in Figure 6.5. Assume that the lines across the top of the figure carry signals whose values one wishes to predict. For simplicity, also assume that these same signals provide the information from which the predictions will be made. For training purposes, the input to the network consists of delayed values of the signals, and the target output consists of the current values of the signals. The network therefore tries to match the current signals values by adjusting a function of their past values. Then when the input to the network consists of past and present values of the signals, the output of the network is a prediction of the values the signals will have in the future.

Assuming each delay unit shown in Figure 6.5 delays its input for $\tau$ time steps, and that the network requires a negligible amount of time to compute its output from its input. The trained network then provides estimates for the values of the signals $\tau$ steps in the future. This approach to adaptive prediction rests on the assumption of a parameterized class of models of the functional relationship between the current and past values of the signals and their later values, or equivalently between earlier values of the signals and their current values.

![Figure 6.5 Using a connectionist network for adaptive prediction.](image-url)
6.6.2 Process of Training Neural Networks

Neural networks learn from experience and not by programming. In supervised neural networks they successively work through a set of examples; each example consist of inputs (information used to make decisions) and patterns (decisions, predictions of conditions or situations). The program used to train the different networks is Brainmaker. Brainmaker attempts to compute the correct output for each example by using the inputs given to calculate the output. It then compares the computed output to the given output. If the calculated output is not equal to the given output, Brainmaker then tries to correct the network by changing the strength of the internal connections between the neurons. This back propagation method [46] continues until the computed output matches the given output according to a certain preset threshold.

6.6.3 Implementation: Training Neural Nets

A total of six neural networks were trained to be used as pattern generators. The model of adaptive prediction outlined in paragraph 6.6.1 was modified for this specific application and the modification is shown in Figure 6.6. Each joint of the robot leg has its own pattern generator for each phase, viz. the stance and swing phase. In Chapter 4, paragraph 4.4 the passing of the command to the signal processing neuron is described. The signal processing neuron is excited by either the stance or swing neuron. When excited, the signal processing neuron integrates the incoming signals and passes the value to the pattern generator (see Figure 4.3).

The lines entering the system in Figure 6.6 carries the input values to both the system and the neural network we wish to train. In other words these input values are the values received from the signal processing neuron. Exiting the system are the values displayed in Figures 7.1 and 7.3. As explained before an output is also essential for a network to be trained. The outputs displayed in the above mentioned figures are used as the output values to train the networks.

![Modified system for adaptive prediction](image)

**Figure 6.6 Modified system for adaptive prediction.**
It should be noted that in Figures 7.1 and 7.3 there are a total of 180 output facts and there are also a total of 180 inputs necessary for these outputs to be generated. At first all the data were used to train the networks, then only 90, then 60, then 45 and eventually 30 of a total of 180 training facts were used. The error made by the network remained the same each time (the calculation of the error will be explained later). Appendix A1.2.2. shows the program written and used to generate the training file for Brainmaker.

6.6.4 Layout of Pattern Generators
The objective of the pattern generators is a trajectory planning task. After the networks are taught to follow a certain trajectory they must be able to plan a trajectory according to the one they were trained with. In other words the pattern generators or networks have to plan its trajectory as close as possible to the training pattern. In order to obtain this precision, it is necessary to fine tune the networks for the smallest error possible. This means a long process of trial and error to find the correct number of inputs, number of neurons in the hidden layer and the optimum training and testing tolerances. (The latter are values used by Brainmaker to fine tune the network). After a substantial amount of time, the optimum networks were obtained and consisted of five input neurons, ten neurons in a single hidden layer with one output neuron. The five inputs are a result of memory that is built into the system and consists of the previous values of the inputs to the network.

6.6.5 Error Calculation
It is expected that the pattern generator will make a small error in its trajectory planning. In order to fine tune the networks, it is necessary to calculate this error. The simulation and network outputs were compared to compute the error, and is calculated by using the following equation:

\[
\text{Error} = \sum_{i=0}^{\infty} (N_{oi} - S_{oi})^2
\]

where \(N_{oi}\) = Network output value
\(S_{oi}\) = Simulation output value

6.6.6 Brainmaker Output
Brainmaker creates a file containing the connection matrix of the trained network. In order to use this file a program is written that uses the output from the signal processing neuron as input to the network and then generates the information necessary to control the three motors to the next point on the planned trajectory. A detailed outline of the program is listed in Appendix A1.2.4. Working through this program listing, the process of generating the control values for the motors, using the input values, will become clear.
6.7 Connection to the Central Pattern Controller

After constructing all the pattern generators they were connected to the central pattern controller. A shift register is used to retain the previous input values for the pattern generators. The connection and implementation of these pattern generators is shown schematically in Figure 4.3. The results are shown and explained in the next chapter.

6.8 Irregularities in Terrain - Stance Phase

After the control of the robot leg walking on a horizontal surface was deemed satisfactory, it was decided to look at the problem of walking over uneven surfaces. Only the stance phase will be considered at this point.

What happens when an insect steps on an object or in a hole in the ground? The insects walking control system must be able to adapt its stepping pattern movements promptly to compensate for irregularities in the terrain on which it is walking.

The mathematical simulation was run again but the values for \( P_x \) were varied between -50mm and 50mm in steps of 5mm. In other words the simulation was run as if the robot leg would be stepping in a hole or onto an object. The distance \( P_x \) was held constant at 100mm. It was found that only the hip and knee angles, \( \phi \) and \( \psi \), changed while the torso angles, \( \theta \), stayed constant. If the results are plotted it can be clearly seen that there exists some similarity between the different plotted lines. See Figures 6.7 and 6.8.

The similarity exists in that all the motor patterns are the same as the pattern when the robot leg is walking on a horizontal surface. In other words for the robot leg controller to compensate for stepping on objects or in holes, the controller must make some adjustment to the basic curve. (With the basic curve is meant the curve obtained when walking on an horizontal surface).

Looking closely at Figures 6.7 and 6.8 one can see that all the curves have the characteristic of an offset and gain applied to the basic curve. This property is then used to derive an equation to enable the controller to adapt to the irregularity in terrain.

The equation derived is the following:

\[
\theta_\Delta = \frac{(\theta_\Delta - \theta_{\text{min}} + \text{Gain} \times \theta_{\text{min}}) \theta_{\text{min}}}{\text{Gain} \times \theta_{\text{min}}} \quad (6.38)
\]

where

- \( \theta_\Delta \) = Desired angle when foot is at position \( \Delta \) from basic position.
- \( \theta_\Delta \) = Basic angle if foot would be on an horizontal surface.
- \( \theta_{\text{min}} \) = Minimum angle of basic curve.
- \( \theta_{\text{min}} \) = Minimum angle of curve when foot is at position \( \Delta \).
From Equation (6.38) all the values are constant except for $\theta_\alpha$ which represents the values of the basic curve. In other words if the constants in Equation (6.38) are known, the new curve can be derived.

At the end of the swing phase the robot foot will activate the force and forward angle sensor neurons. At this point the angles of motor 2 and motor 3 are known and the foot position can be determined. It would therefore be known that the foot is at a position $\Delta$ above or below the horizontal surface. If the robot foot is not on the horizontal surface the controller simply needs to make some adjustment to plan the new trajectory.

**Implementation**
The next step was to test this technique. The minimum angle values of all the curves in Figures 6.7 and 6.8 were calculated as well as the gain and offset for each curve. A neural network was trained with the motor angles 2 and 3, at positions Δ, as inputs. The gain and minimum values for each curve, when the foot is at position Δ, is used as outputs to the neural network. (See Figure 6.6).

Therefore if we know the angle values of motor 2 and motor 3 at the beginning of the stance phase these values are then sent to the neural network which will provide the gain and minimum values necessary to be used with equation (6.38) to derive the new trajectories that motor 2 and motor 3 must follow.

This pattern generator would be connected the proposed controller, discussed in Chapter 4, as shown schematically in Figure 6.9.

Assume that the robot leg has reached the end of the swing phase, at this point the FA neuron and the pressure sensor F are activated. The obstacle processing neuron, O, is then activated by both the FA neuron and the pressure sensor. On activation the obstacle neuron inhibits the SPST neuron to prevent it from firing. The obstacle neuron also calculates whether the foot is at a position Δ above or below the horizontal surface. Another input to this neuron is the angle values of motor 2 and motor 3, from sensors S₂ and S₃, at the end of the swing phase.

Figure 6.9 Modifications to the proposed model for uneven terrain.
The obstacle processing neuron then uses these signals as input values to a pattern generator. The pattern generator generates the gain and offset necessary to change the basic trajectory. The output of the pattern generator is then forwarded to the obstacle trajectory neuron \(OT\). This neuron then excites the \(SP_{st}\) neuron, cancelling the inhibition from the \(O\) neuron and causing the \(SP_{st}\) neuron to fire. The signal received from the \(PG_{st}\) is then changed by the \(OT\) neuron, using Equation (6.38) and the signals received from the \(O\) neuron. The output signals of the \(OT\) neuron are then the correct values to control motor 2 and motor 3.

All the results of the implementation discussed in this chapter are given in Chapter 7 with the necessary explanations and comments regarding the accuracy achieved by the controller.
CHAPTER SEVEN

Project Results

7.1 Introduction

This chapter deals with all the results obtained while implementing the preceding model of the various parts. This chapter will be ordered in the following way:
- Kinematic Simulation,
- Pattern Generators for walking on Horizontal Surfaces and
- Pattern Generators for Surface Irregularities.

7.2 Kinematic Simulation

As explained in paragraph 6.5, this kinematic simulation consisted of three phases namely:
- the stance phase,
- the first part of the swing phase and
- the second part of the swing phase.

7.2.1 Stance Phase

Equations 6.26 to 6.35 were used to obtain the different angles of the motors during the stance phase. Figure 7.1 shows the change of the angles for each motor. The curve of motor 1 shows that the speed of the leg changes from slow at the beginning of the stance phase, to fast in the middle and then slow again at the end of the stance phase.

Figure 7.2a shows the robot leg in three dimensions. It should be noted again that the body of the robot was held fixed during the simulation. Figures 7.2b, 7.2c and 7.2d shows the top, front and side views of the robot leg.

The curves in Figure 7.1 are the basic curves referred to in paragraph 6.8.

7.2.2 Swing Phase - First and Second Part

To obtain the different angles for the swing phase the same equations used for the stance phase were used with different parameters. The changes of the angles for each motor are shown in Figure 7.3.
Figure 7.1 Stance phase angles: $\theta$, $\phi$ and $\psi$.

Figure 7.2 Schematic representation of the stance phase. a) 3D view, b) top view, c) front view, and d) side view.
Figure 7.3 Swing phase angles: $\theta$, $\phi$ and $\psi$.

Figure 7.4 Schematic representation of the swing phase. a) 3D view, b) top view, c) front view, and d) side view.
In Figures 7.4 to 7.4d it can be clearly seen how the foot is lifted off the ground at the beginning of the swing phase, swung through the air and put down on the ground at the end of the swing phase. Again, the curves in Figure 7.4 are referred to as the basic swing curves.

The complete simulation step cycle, from stance phase through to the swing phase, is shown in Figure 7.5.

7.3 Pattern Generators for walking on Horizontal Surfaces

As explained in paragraph 6.6, inputs and outputs are necessary to train a neural net. The inputs to the neural net are the integrated values received from the signal processing neuron, the full range of possible values are shown in Figure 7.6a, for the stance phase, and in Figure 7.6b, for the swing phase. The output pattern of the network we desire is the patterns shown in Figures 7.1 and 7.3.

At first it was tried to use one pattern generator to predict the curves of all three motors for both the stance and swing phases. The data was not well enough defined and the networks refused to train. A pattern generator for each motor was then constructed, for the swing and stance phases, this was more suitable for Brainmaker and the networks trained very well.
7.3.1 Stance Phase

After the pattern generators were constructed, its output were compared to that of the simulations output. The outputs are compared by using Equation 6.37.

Figure 7.7 shows the output of the simulation, for motor 1, plotted against the output of the pattern generator for the stance phase. The error made by the pattern generator is in the order of 1.79%. Looking closely at the graph it can be seen that error is in fact very small and therefore acceptable.

Figure 7.8 shows the stance phase of motor 2. Again the output of the simulation is plotted against the output of the pattern generator. During this phase the error made by the pattern generator is in the order of 0.31%.

Figure 7.9 shows the output of the simulation and the pattern generator for motor 3 during the stance phase. The error made by the pattern generator is in the order of 0.575%.

Looking closely at these figures for the stance phase, it can be seen that the starting positions of the pattern generator output differ a little from that of the simulation. For motor 1 the starting angle is a little smaller than the simulation output. Looking at Figure 7.10, the output of the pattern generator for the swing phase, it can be seen that the ending position angle of the swing phase is also a little smaller than the simulations output. This will cause the transition, from swing to stance, to occur without sudden changes to motor positions.

Project Results
Figure 7.7 PG Output for Motor 1 - Stance Phase.

Figure 7.8 PG Output for Motor 2 - Stance Phase.

Figure 7.9 PG Output for Motor 3 - Stance Phase.
Figure 7.10 PG Output for Motor 1 - Swing Phase.

Figure 7.11 PG Output for Motor 2 - Swing Phase.

Figure 7.12 PG Output for Motor 3 - Swing Phase.
7.3.2 Swing Phase

Figure 7.10 shows the output of the simulation compared to the output of the pattern generator for motor 1. The error made by the pattern generator is in the order of 0.33% and is acceptable.

In Figure 7.11 the difference between the simulation output and the pattern generator output are shown. The pattern generator made an error in the order of 1.36%. It has to be kept in mind that this is the swing phase and the only concern is that the starting and ending positions of this motor should be as near as possible to that of the simulation. During the rest of the phase the pattern may differ in small variations from the simulations output. With this in mind the error made is acceptable.

Figure 7.12 shows the simulation output and pattern generator output plotted against each other. At first the error was made by the pattern generator was in the order of 3.1%. All the networks were trained with the same memory size. The smaller the network memory size, the less the computing time. The error of 3.1% made by this pattern generator was not acceptable. Changes were made to the pattern generator to try and reduce the error to a more acceptable number. Eventually the memory of this pattern generator was changed from 5 previous inputs to 10 previous inputs and the error reduced to 0.11%.

7.3.3 Complete Step Cycle

The pattern generators plan the trajectory of the robot leg by using previous information of the position of the robot leg. Figure 7.13 shows the complete cycle planned by the pattern generators.

![Figure 7.13 Trajectory planned by the pattern generators.](context)
The transition from stance to swing phase and swing to stance phase is not very smooth. There appears to be a jump from the one to the other. This is due to the fact that the starting and ending positions planned by the pattern generators for each motor differ a little bit.

7.3.4 Connection to the Proposed Controller
After the construction of the various pattern generators, they were connected to the proposed controller. Figure 7.14 shows the result of the planning done by the pattern generators during a step cycle. This compares very well to the simulation shown in Figure 7.5. The error made by the pattern generators are the same as the errors given in the preceding paragraphs.

Figure 7.14 shows the unstable transition from stance to swing between the various motors. This is simply caused by the fact that the angles given by the pattern generator at the end of a phase differ from the angles given at the beginning of a phase.

Figures 7.15 and 7.16 shows the robot leg in three dimensions during a step cycle after the pattern generators were connected to the proposed controller. These figures are not entirely the same as those in Figures 7.2 and 7.4 of the simulation.

![Figure 7.14 PG output after connection to the proposed controller.](image-url)
Figure 7.15  Schematic representation of the stance phase after the connection to the proposed controller. a) 3D view, b) top view, c) front view and d) side view.

Figure 7.16  Schematic representation of the swing phase after the connection to the proposed controller. a) 3D view, b) top view, c) front view and d) side view.
7.4 Pattern Generators for Irregularities in Walking Terrain

Paragraph 6.8 explained the theory about the changes needed to control the robot leg when stepping onto an object or in a hole. The inputs to the network are the motor angles 2 and 3, at the end of the swing phase, at positions $\Delta$ from the horizontal level. The outputs are the offset and gain values for each curve (see Figure 6.6). The gain values are calculated with the aid of Equation (6.38).

Two pattern generators were constructed; one for motor 2 and one for motor 3. The following results were obtained after the pattern generators were trained and tested.

<table>
<thead>
<tr>
<th>Position $\Delta$, from basic curve</th>
<th>Error (%) Motor 2 angle $\phi$</th>
<th>Error (%) Motor 3 angle $\psi$</th>
<th>Curve Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>-50</td>
<td>8.038</td>
<td>2.12</td>
<td>1</td>
</tr>
<tr>
<td>-40</td>
<td>2.385</td>
<td>2.95</td>
<td>2</td>
</tr>
<tr>
<td>-25</td>
<td>1.615</td>
<td>2.85</td>
<td>3</td>
</tr>
<tr>
<td>-5</td>
<td>0.975</td>
<td>3.13</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2.00</td>
<td>2.56</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>2.44</td>
<td>1.56</td>
<td>6</td>
</tr>
<tr>
<td>40</td>
<td>6.84</td>
<td>4.49</td>
<td>7</td>
</tr>
<tr>
<td>50</td>
<td>8.46</td>
<td>9.6</td>
<td>8</td>
</tr>
</tbody>
</table>

Figures 7.17 and 7.18 shows the adapted curves plotted against the simulation curves. It can be seen clearly from these figures that the factor responsible for the variation in error is the minimum value given by the pattern generator. The minimum values given by the pattern generator differ from -0.003 to 0.003 radians from the ideal values and therefore cause the increasing error.

To verify if these results are acceptable a test was run for $P$, at 23mm above the horizontal level. The outputs of the pattern generator are then used to calculate the curves for the two motors at this foot position. The results are shown in Figure 7.19. This figure shows that the adapted curve is virtually identical to the ideal curve given by the simulation. Figure 7.20 shows a three-dimensional view of the robot leg for the test. This figure compares very well to Figure 7.2.
Figure 7.17 Adapted curves against ideal curves for Motor 2.

Figure 7.18 Adapted curves against ideal curves for Motor 3.
Figure 7.19 Adapted curve plotted against ideal curve for stepping on an object 23mm above the horizontal level.

Figure 7.20 Three-dimensional view of robot leg when stepping on an object 23mm above the horizontal level.
CHAPTER EIGHT

Conclusion

I hear and I forget.
I see and I remember.
I do and I understand.
- Confucius

8.1 Discussion

During this research backpropagation training of multilayer perceptrons and a combination of heterogeneous neurons have been used to implement several pattern generators with different behaviours in order for an autonomous robot to be able to develop a trajectory for walking on an uneven solid surface.

The control of walking in humans and animals is a sophisticated and complex process. The method developed here was to consider a biological system to develop a neural control system for walking of a hexapod which is inherently biologically feasible. In this research, by investigating the behaviour of the cat, cockroach and praying mantis the basis of walking on horizontal surfaces has been extended for an autonomous agent from what was achieved by earlier researchers [48].

Several types of artificial neurons were simulated, each with its own function, and connected, to form a complete control system. The control system consisted of different hierarchical levels as in physical systems. The controller was then connected to the different pattern generators to control the motion of the robot leg.

The results presented in Chapter 7 show that a biologically feasible controller is a workable solution to hexapod locomotion on irregular surfaces. This system controlled a three degree-of-freedom robot leg for walking on a horizontal surface with satisfactory results. The higher level task is divided into workable subcommands at each command level within the complete hierarchy. Communication between the different levels of the control system is limited to essential neural quantities. The research hypothesis stated in Chapter 1 can thus be positively asserted. This control system can further be extended to include functions other than those of the basic leg control problem. A demonstrated example is that of negotiating an irregular terrain.
8.2 Future Work

The next stage in a project such as this would be to connect six of these controllers to a central locomotion initiation unit to investigate the existence of a stable gait for such a complex neural controller. Once a stable gait is maintained, the next step would be to build a physical hexapod robot. When the hexapod can walk autonomously on a horizontal surface, the next stage would then be to simulate and test the robot when walking over uneven surfaces. After a certain level of competence has been reached, problems such as walking over soft surfaces, inclined surfaces and slippery surfaces should be investigated. Further research possibilities include extensions of this basic motion controller design technique to alternative insect motion capabilities such as climbing, gliding and swimming.


Bibliography


---

Bibliography
Appendix

A1.1 Determination of Transformation Matrices

A1.1.1 Rotation of Axes in 3D-Space

Suppose a rectangular $xyz$-coordinate system is rotated around its $z$-axis counterclockwise (looking down the positive $z$-axis) through an angle $\theta$ (Figure A1.1). If we introduce unit vectors $u_1$, $u_2$, and $u_3$ along the positive $x$, $y$ and $z$ axes and unit vectors $u'_1$, $u'_2$, and $u'_3$ along the positive $x'$, $y'$ and $z'$ axes, we can regard the rotation as a change from the old basis $B = \{u_1, u_2, u_3\}$ to the new basis $B' = \{u'_1, u'_2, u'_3\}$. It should be evident that

$$[u_1]_B = \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} \quad \text{and} \quad [u_2]_B = \begin{bmatrix} -\sin \theta \\ \cos \theta \\ 0 \end{bmatrix}$$

(A1.1)

Moreover, since $u'_3$ extends 1 unit up the positive $z$-axis

$$[u_3]_B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

(A1.2)

The transition matrix from $B'$ to $B$ is

$$P = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(A1.3)

and the transition matrix from $B$ to $B'$ is

$$P = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(A1.4)
Thus the old coordinates \((x,y,z)\) of a point \(Q\) are related to its new coordinates \((x',y',z')\) by

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix}
= \begin{bmatrix}
\cos \theta & \sin \theta & 0 \\
-\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]  \hspace{1cm} (A1.5)

In other words, the rotation about the \(z\)-axis is given by:

\[
\text{Rot}(Z, \theta) = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (A1.6)

The determination of the other rotation matrices around the \(x\) and \(y\) axis follows in the same manner.
A1.2 Software listings - Implementation

A1.2.1 Mathematica Kinematic Simulation

**INITIALISATION**
d1=175.0; a1=50.0; a2=150.0; a3=300.0;
q=Table[0, {3}];
q1=Table[0, {3}];
Pos=Table[0, {180}, {4}, {3}];

*File for output values*
thetafile = OpenWrite["stance"];
iter=0;

**Starting and Final Conditions**
PxBegin=250.0;
PyBegin=200.0;
PzBegin=0.0;

PxEnd=250.0;
PyEnd=-200.0;
PzEnd=0.0;

Py=PyBegin;
Px=PxBegin;
Pz=PzBegin;

**Setting of Intervals**
steps=180;
dPx=(PxEnd-PxBegin)/steps;
dPy=(PyEnd-PyBegin);
dPz=(PzEnd-PzBegin)/steps;

**Starting of Simulation**
Do[iter++;

*Calculation of θ1*
If[Unequal[Px,0], q[[1]]=ArcTan[Py/Px], q[[1]]=Pi/2];

*Calculation of Cos θ, and Sin θ*
C3=((Cos[q[[1]]]Px+Sin[q[[1]]]Py-a1)^2+(Pz-d1)^2-a2^2-a3^2)/(2 a2 a3);
If[Abs[C3]>1, Print["Error -> Cos[theta3]>1"], S3=Sqrt[1-C3^2]];

*Calculation of θ3*
q[[3]]=ArcTan[Abs[S3]/Abs[C3]];
Determination of the quadrant for $\Theta_1$

\[
\begin{align*}
&\text{If}[C3 < 0.0 \&\& S3 < 0.0, q[3] = q[3] - \pi, \\
&\quad \text{If}[C3 > 0.0 \&\& S3 < 0.0, q[3] = -q[3]], \\
&\quad \text{If}[C3 < 0.0 \&\& S3 > 0.0, q[3] = \pi - q[3], t = 0]];
\end{align*}
\]

Calculation of $\sin \theta_1$ and $\cos \theta_1$

\[
S2 = \frac{(\cos[q[3]] a3 + a2)(Pz - dl) - \sin[q[3]] a3 (\cos[q[1]] Px + \sin[q[1]] Py - a1)}{(a2^2 + a3^2 + 2 a2 a3 \cos[q[3]])};
\]

\[
C2 = \frac{(\cos[q[3]] a3 + a2)(\cos[q[1]] Px + \sin[q[1]] Py - a1) + \sin[q[3]] a3 (Pz - dl)}{(a2^2 + a3^2 + 2 a2 a3 \cos[q[3]])};
\]

Calculation of $\theta_1$

\[
q[2] = \arctan[\text{Abs}[S2]/\text{Abs}[C2]]; \]

Determination of the quadrant for $\Theta_2$

\[
\begin{align*}
&\text{If}[C2 < 0.0 \&\& S2 < 0.0, q[2] = q[2] - \pi, \\
&\quad \text{If}[C2 > 0.0 \&\& S2 < 0.0, q[2] = -q[2]], \\
&\quad \text{If}[C2 < 0.0 \&\& S2 > 0.0, q[2] = -\pi + q[2], t = 0]];
\end{align*}
\]

Preparing of Data to be written to File

\[
\begin{align*}
&\text{If}[\text{Less}[q[1], 0.000008]\&\&\text{Greater}[q[1], -0.000008], q[1] = 0, q[1]], \\
&q[2] = N[q[2], 4], \\
&q[3] = N[(\pi + q[3]), 4];
\end{align*}
\]

Actual writing to File

\[
\text{Write}[\text{thetafile}, q[1], \text{OutputForm}[" ", q[2], \text{OutputForm}[" ", q[3]]];
\]

Add Interval

\[
dPystep = N[dPy, \text{Abs} [\cos[\pi(iter)/180] - \cos[\pi(iter-1)/180]]/2, 4];
\]

(This is a Cosine function because we want the leg to start moving slow, accelerate and then slowing down at the end of the stance or swing - Motor 1, See FIgure 7.2)

\[
Px += dPx; \\
Py += dPystep; \\
Pz += dPz;
\]

Calculation of the Various positions of the leg segments

Hinge point at body

\[
\begin{align*}
x & = 0; \\
y & = 0; \\
z & = dl;
\end{align*}
\]

Endpoint Position of First Segment

\[
\begin{align*}
x & = \cos[q[1]](a1); \\
y & = \sin[q[1]](a1); \\
z & = dl;
\end{align*}
\]
Endpoint Position of Second Segment
\[x_2 = \cos(q_1)(a_1 + a_2 \cos(q_2)); \]
\[y_2 = \sin(q_1)(a_1 + a_2 \cos(q_2)); \]
\[z_2 = d_1 + (a_2 \sin(q_2)); \]

Endpoint Position of Third Segment
\[x_3 = \cos(q_1)(a_1 + a_2 \cos(q_2) + a_3 \cos(q_2 + q_3)); \]
\[y_3 = \sin(q_1)(a_1 + a_2 \cos(q_2) + a_3 \cos(q_2 + q_3)); \]
\[z_3 = d_1 + (a_2 \sin(q_2) + a_3 \sin(q_2 + q_3)); \]

Preparing for Graphics - to draw lines between Endpoint Positions
Pos[[i]] = \{\{x_0, y_0, z_0\}, \{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}, \{x_3, y_3, z_3\}\};

End of Simulation
Close[thetafile]; File written to must be closed

Graphics
line1 = Table[0, \{20\}];
Do[line1[[i]] = Line[Pos[[i*9]]], \{i, 1, 20\}];
po1 = Show[Graphics3D[line1, PlotRange -> All]];

Show[po1, ViewPoint -> \{00, 0, 100\}];
Show[po1, ViewPoint -> \{00, 100, 0\}];
Show[po1, ViewPoint -> \{100, 00, 0\}];
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#define Pi 3.141592654

int geheue, i, aantal, NITEMS = 180, n, p=2, a, legno, mtrno;
char mtr[50];
char *infile = "stancer", read[50], write[50], test[50], tcmp[50], outfilc[50];
float thl[200], th2[200], th3[200], thltcmp[200], th2temp[200], th3temp[200];
float thlmax, th2max, th3max, th1min, th2min, th3min, maxposth2, maxposth3;
float lut[180];

void main (void)
{
    FILE *inputfile, *outputfile, *testfile;
    clrscr();

    printf("Output filename =>");
    scanf("%s", &outfile);
    printf("Motor number =>");
    scanf("%d", &mtrno);
    printf("Geheue =>");
    scanf("%d", &geheue);
    printf("Simulate ==>");
    scanf("%d", &mtrno);
    printf("%s", infile);
    sprintf (read, "%s", infile);
    sprintf (write, "%s.def", outfile);
    sprintf (test, "%s.tst", outfile);
    sprintf (tcmp, "%s.tcmp", outfile);
    sprintf (outfilc, "%s.def", outfile);

    /******************Read Simulation File into Memory **********************/

    if ((inputfile = fopen(read, "rt")) == NULL)
    {
        fprintf(stderr, "Cannot open input file.
");
        exit(0);
    }
    thlmax = th2max = th3max = 0;
    th1min = th2min = th3min = 100;
    for(i = 0; i <= NITEMS; i++)
    {
        printf("Reading file %d\r", i);
        fscanf(inputfile, "%f %f %f", &th1temp[i], &th2temp[i], &th3temp[i]);
    }
}
if( fabs(thltemp[i])>=thlmax ) thlmax =fabs(thltemp[i]);
if( th2temp[i]>=th2max ) th2max = th2temp[i];
if( th3temp[i]>=th3max ) th3max = th3temp[i];
if( th1temp[i]<=th1min ) th1min = th1temp[i];
if( th2temp[i]<=th2min ) th2min = th2temp[i];
if( th3temp[i]<=th3min ) th3min = th3temp[i];
}

maxposth2 = th2max - th2min;
maxposth3 = th3max - th3min;

printf("\nmax = %.4f %.4f %.4f\t", thlmax, th2max, th3max);
printf("min = %.4f %.4f %.4f\t", thlmin, th2min, th3min);

fclose(inputfile);
printf(\"\n\"");

/**************************************************************************/

for(i = 0; i<=NITEMS; i++)
{
    // Scaling values for Brainmaker between 4 and -4
    th1[i] = (th1temp[i])/th1max) * 4;    // Scale to one and then to 4
    th2[i] = (th2temp[i])/(th2min)/maxposth2 * 4;
    th3[i] = (th3temp[i])/(th3min)/maxposth3 * 4;

    // Make LUT for NN
    lut[i] = cos(\(\text{Pi}i\)/180);
}

/**************************************************************************/ Write desired definition file for Brainmaker***************************/

if ((outputfile = fopen(write, "wt")) == NULL)
{
    fprintf(stderr, "Cannot open output file.\n");
    exit(0);
}

aantal = geheue * 1;
// aantal = 3;

sprintf(temp, "input number 1 %d", aantal);

fprintf(outputfile, temp);
switch (mtno) {
    case 1: fprintf(outputfile, "\n\noutput number 1 \n\n", th1[i]); break;
    case 2: fprintf(outputfile, "\n\noutput number 1 \n\n", th2[i]); break;
    case 3: fprintf(outputfile, "\n\noutput number 1 \n\n", th3[i]); break;
    case 4: fprintf(outputfile, "\n\noutput number 1 3\n\n", th1[i], th2[i], th3[i]); break;
}

fprintf(outputfile, "scale input minimum\n");
for (i = 0; i<aantal; i++) {
    fprintf(outputfile, "-6 ");
}
fprintf(outputfile, "unscale input maximum\n");
for (i = 0; i<aantal;i++) {
    fprintf(outputfile, "6 ");
}

fprintf(outputfile, "unscale output minimum\n");
fprintf(outputfile, "4\n");

fprintf(outputfile, "scale output maximum\n");
fprintf(outputfile, "4 \n");

fprintf(outputfile, "nfacts\n");
for (i=0; i<=NITEMS; i+=p) {
    printf("Writing .def file %d\r", i);
    for(n=0; n<aantal; n++) {
        a = (i-n);
        if (a<0) fprintf(outputfile,"%.4f ",lut[180+a]);
        else fprintf(outputfile,"%.4f ",lut[a]);
    }
    switch (mtno) {
    case 1: fprintf(outputfile, "\n%.4f \n", th1[i]); break;
    case 2: fprintf(outputfile, "\n%.4f \n", th2[i]); break;
    case 3: fprintf(outputfile, "\n%.4f \n", th3[i]); break;
    case 4: fprintf(outputfile, "\n%.4f %.4f %.4f \n", th1[i], th2[i], th3[i]); break;
    }
}
fclose (outputfile);
printf("\n");
A1.2.3 Sample *.net File generated for Brainmaker

This file is a sample file for the swing phase of motor 1.

input number 1 5
output number 1 1
hidden 10

filename trainfacts swing1.def
filename testfacts swing 1.tst
checkpoint 8 CHECKPT.NET

learnrate 1.0000 50 1.0000 75 1.0000 90 1.0000
learnlayer 1.0000 1.0000
traintol 0.0010 0.0010 0.8000 100
testtol 0.0010
maxruns 99999
smoothing 0.9000 0.9000
testruns 0

function hidden1 sigmoid 0.0000 1.0000 0.0000 1
function output sigmoid 0.0000 1.0000 0.0000 1

display histogram hidden1 17 0
display histogram output 17 40
scale input minimum
-6 -6 -6 -6
scale input maximum
6 6 6 6
scale output minimum
-4
scale output maximum
4

statistics 5942482 4814570 65302
weights 3 1 5 10 1
5.4954 -5.6080 7.8036 2.3170 -3.9208 -3.1500
0.4554 -7.9126 -0.3366 -7.1044 6.5608 4.2536
-6.8964 -6.8964 -5.9776 -2.9934 -3.1350 -7.9998
7.9994 7.9994 7.9994 7.9994 7.9994 7.9994
4.7704 1.3510 3.4054 7.9994 -2.4860 1.3510 0.9508 -4.0850 3.4086 -0.0282 0.0384

Appendix Page A10
A1.2.4 Pattern Generator Program

This program reads the *.net generated by Brainmaker and generate the trajectory for a specific motor.

```
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <string.h>
#include <process.h>
#include <math.h>

#define MaxLayers 8

#define WeightVector(layer, neuron) (Weights[layer][neuron])
#define InitializeMemory(_i) ;

/************************ prototypes for all functions *************************/

void main(const void);
void ProcessInputs(void);
void GetNetworkSize(void);
void AllocateNetwork(void);
void DeallocateNetwork(void);
void RunNetwork(void);
int FindKeyword( FILE *f, int *output, int *max );
int ReadMinimumOrMaximumKeyword( FILE *f, int *max );
void VectorTransform( float *v1, int len1, int layer, float *v2, int len2 );
float VectorDotProduct( float *vl, int len, float *v2 );
float myexp( float n );
void TransferFunction( float *v, int len );
void ReadMinMax(void);
void DefaultMinMax(void);
void ReadWeights(void);
int ReadFact( FILE *f, float *v, int length, float *min, float *max );
void WriteFact( FILE *f, float *v, int len, float *min, float *max );
void FreeWeights( int layer, int neurons );
int AllocateWeights( int layer, int neurons, int connections );
void ExitProgram( char *errormessage, int errorcode );

float **Weights[MaxLayers-1];
char *WeightsFile[80] = "Mtl.net",
    MinMaxFile[80] = "Mtl.net",
    InputFile[80] = "Mt.mat",
    OutputFile[80] = "Mtl.mat";
float *Vectors[MaxLayers];
int NumberOfNeurons[MaxLayers];
int NumberOfLayers = 3, Thresholds = 1;
float *Min[2], *Max[2];
```
void main ( void )
{
    ProcessInputs();        // File Management
    GetNetworkSize();        // Get Network Size
    AllocateNetwork();       // Memory Management
    ReadMinMax();            // I/O Routine
    ReadWeights();           // I/O Routine
    RunNetwork();            // Processing Network
    DeallocateNetwork();     // Memory Management

    ExitProgram( "Done.", 0 );        // Process Control
}

/******************** Network definition ********************/

void ProcessInputs( )
{

    printf( "Reading weights from %s,\n", WeightsFile );
    printf( "reading inputs from %s,\n", InputFile    );
    printf( "writing outputs to %s,\n", OutputFile  );
    printf( "reading minmax from %s,\n", MinMaxFile );
}

/******************** Get Network Size ********************/

void GetNetworkSize
{
    FILE *f; int i; char keyword[128];

    f = fopen( WeightsFile, "r" );
    do fscanf( f, "%s", keyword );
        while( strncmp( keyword, "weights", 7 ) );

    fscanf( f, "%d %d", &NumberOfLayers, &Thresholds );

    for( i=0; i<NumberOfLayers; i++ )
        fscanf( f, "%d ", &NumberOfNeurons[i] );

    fclose( f );

    printf( "Network size: %d thresholds,\n", Thresholds );
    printf( "%d layers, layer sizes:\n", NumberOfLayers );
        for( i=0; i<NumberOfLayers; i++ )
            printf( "%d", NumberOfNeurons[i] );
    printf( "\n" );
}
/** Memory management

void AllocateNetwork()
{
    int i, j;

    for( i=0; i<NumberOfLayers; i++ )
    {
        Vectors[i] = (float *)malloc( (NumberOfNeurons[i] + Thresholds)*sizeof(float));
    }

    Min[0] = (float *)malloc( NumberOfNeurons[0] * sizeof(float));
    Max[0] = (float *)malloc( NumberOfNeurons[0] * sizeof(float));
    Min[1] = (float *)malloc( NumberOfNeurons[NumberOfLayers-1] * sizeof(float));
    Max[1] = (float *)malloc( NumberOfNeurons[NumberOfLayers-1] * sizeof(float));

    for( i=0; i<NumberOfLayers-1; i++ )
    {
        if( !AllocateWeights( i, NumberOfNeurons[i+1], NumberOfNeurons[i] + Thresholds ) )
        {
            ExitProgram( "Out of memory!", 1 );
        }
    }

}

void DeallocateNetwork()
{
    int i, j;

    for( i=0; i<NumberOfLayers-1; i++ )
    {
        FreeWeights( i, NumberOfNeurons[i+1] );
    }

    for( i=0; i<2; i++ )
    {
        free( Min[i] );
        free( Max[i] );
    }

    for( i=0; i<NumberOfLayers; i++ )
    {
        free( Vectors[i] );
    }
}

int AllocateWeights( int layer, int neurons, int connections )
{
    int i;

    Weights[layer] = (float **)malloc( neurons * sizeof(float) );
}
if( !Weights[layer][i] ) return 0;
for( i=0; i<neurons; i++ )
{
    Weights[layer][i] = (float *)malloc( connections * sizeof(float) );
    if( !Weights[layer][i] ) return 0;
}
return 1;

void FreeWeights( int layer, int neurons )
{
    int i;
    for( i=0; i<neurons; i++ )
    {
        free( Weights[layer][i] );
    }
    free( Weights[layer] );
}

/*************************** Neural network operation ***************************/

void RunNetwork()
{
    FILE *in, *out; int i;
in = fopen( InputFile, "r" );
out = fopen( OutputFile, "w" );
while( ReadFact( in, Vectors[0], NumberOfNeurons[0], Min[0], Max[0] ) )
{
    for( i=0; i<NumberOfLayers-1; i++ )
    {
        VectorTransform( Vectors[i], NumberOfNeurons[i] + Thresholds, i, Vectors[i+1],
NumberOfNeurons[i+1] );
        TransferFunction( Vectors[i+1], NumberOfNeurons[i+1] );
    }
    WriteFact( out, Vectors[NumberOfLayers-1],
NumberOfNeurons[NumberOfLayers-1], Min[1], Max[1] );
}
fclose( in );
fclose( out );
}

void VectorTransform( float *v1, int len1, int layer, float *v2, int len2 )
{
    int i; float *w;
v1[len1-1] = 1.0;  //Value of threshold neuron is set to 1

Appendix
for( i=0; i<len2; i++ )
{
    w = WeightVector( layer, i );
    *v2++ = VectorDotProduct( v1, len1, w );
}

float VectorDotProduct( float *v1, int len, float *v2 )
{
    float sum;
    sum = 0.0;
    while( len-- )
    {
        sum += *v1++ * *v2++;
    }
    return sum;
}

/* sigmoid transfer function definition, hard-coded. */
#define GAIN 1.0
#define HIGH 1.0
#define LOW 0.0
#define CENTER 0.0

float myexp( float n )
{
    if( n < -10.0 ) n = -10.0;
    if( n > 10.0 ) n = 10.0;
    return exp(n);
}

void TransferFunction( float *v, int len )
{
    while( len-- )
    {
        *v = (HIGH - LOW) / ( 1 + myexp( GAIN * ( CENTER - *v ) ) ) + LOW;
        if( *v < LOW ) *v = LOW; if( *v > HIGH ) *v = HIGH;
        v++;
    }
}

/**************************** I/O routines ***************************/

void ReadMinMax()
{
    int i;
}
for( i=0; i<NumberOfNeurons[0]; i++ )
{
    Min[0][i] = -6.0; Max[0][i] = 6.0;
}
for( i=0; i<NumberOfNeurons[NumberOfLayers-1]; i++ )
{
    Min[1][i] = -4.0; Max[1][i] = 4.0;
}

void ReadWeights()
{
    FILE *f; int i, j, k, dummy; double n; float *w; char keyword[128];

    f = fopen( WeightsFile, "r" );
    if( !f ) ExitProgram( "Cannot open weight file.", 1 );
do fscanf( f, "%s", keyword );
while( strcmp( keyword, "weights", 7 ) );

    fscanf( f, "%d %d ", &dummy, &dummy );
for( i=0; i<NumberOfLayers; i++ )
{
    fscanf( f, "%d ", &dummy );
}
for( i=0; i<NumberOfLayers-1; i++ )
{
    for( j=0; j<NumberOfNeurons[i+1]; j++ )
    {
        w = WeightVector( i, j );
        for( k=0; k<NumberOfNeurons[i]+Thresholds; k++ )
        {
            fscanf( f, "%lf ", &n );
            w[k] = (float)n;
        }
    }
}
fclose( f );

int ReadFact( FILE *f, float *v, int length, float *min, float *max )
{
    int i; double temp;
    for( i=0; i<length; i++ )
    {
        if( feof( f ) ) return 0;
        fscanf( f, "%lf ", &temp );
        v[i] = ( temp - min[i] ) / ( max[i] - min[i] );
    }
}

Appendix
void WriteFact( FILE *f, float *v, int len, float *min, float *max )
{
    int i; double temp;
    for( i=0; i<len; i++ )
    {
        temp = v[i] * ( max[i] - min[i] ) + min[i];
        fprintf( f, "%.5f", temp );
    }
    fprintf( f, "\n" );
}

/****************************** Process ******************************/

void ExitProgram( char *errormessage, int errorcode )
{
    printf( "%s\n", errormessage );
    exit( errorcode );
}