USE OF A DELTA ROBOT
AS
A WALKING MACHINE

A thesis
submitted in partial fulfilment
of the requirements for the degree
of
Master of Engineering
in the
University of Canterbury

by

EDWARD WONG TING PING

1998

Acknowledgements

I would like to express my sincere thanks to Associate Professor Reg Dunlop, my project supervisor, for his guidance, encouragement and patience throughout the duration of this project.

Special thanks go to the workshop staff, particularly Mr Andy Cree who spent many hours setting up the DSP controller and building an interface circuit board.

A special thanks to my parents whom have given me their love and support.

Summary

A 3 degree of freedom (dof) parallel Delta robot was built in the laboratory of the Mechanical Engineering Department, University of Canterbury. It possesses the similar characteristics and features as Clavel's Delta 4 robot, which is well known for pick and place applications. Due to its parallel actuated mechanisms, this type of robot, so far, has been claimed as the fastest robot in the world. However, the Canterbury Delta robot in the laboratory suffered jerky motions when travelling along a prescribed continuous path. This was due to the use of 3 single axis step motor controllers (donated) rather than a single multi-axes control system. In order to improve the performance of the robot, the 3 existing control systems were replaced by a single chip DSP controller (TMS320F240). Under control of this powerful controller, the robot is able to perform point-to-point motion and continuous path motion under an open loop control mode.

In order to use the Delta robot as a walking machine, a tripod foot was successfully developed and attached to the travelling platform of the delta robot. The result was a practical walking machine with 3 dof called Delta walker or Delta walking machine. It is based on parallel mechanisms and has a maximum allowable step length of 120mm. The step length and walking space of the Delta walking machine were studied and stimulated through a static forces analysis in Matlab[1]. It was found that the step length was constrained by the torque limit of the harmonic gear drives rather than the torque output by the stepping motors.

An off-line optimal continuous path planning method was developed in Matlab for real time control at the joint level. The step walking path is approximated by a set of location nodes selected on the desired path. The motion control of the machine is provided by trajectory interpolation at the joint level. The pulse rates and the direction values are generated and sent to the DSP through an RS232 serial communication port.

[1] Chapter 4. section 4.8, page 33.

Table of Contents

Chapter 4

Chapter 5

Trajectory modelling

Chapter 6

Hardware / software synergy on the motion control

List of figures

List of Tables

Page

Chapter 1 Introduction

1.0 General overview

This chapter begins with a discussion of the significant problems with the existing IFX Compumotor controller system used on the Canterbury Delta robot. The properties of the new control system are covered in section 1.2 followed by a general review of the overall physical dimensions and the structural layout of the Delta walking machine (Delta Walker). The gait of the walking machine and the objectives of the research are presented in section 1.3 and 1.4 respectively. In the final section an outline of the thesis is presented to conclude this chapter.

1.1 Motion controller system

A powerful controller is an essential and important part of the motion control system. Basically, it provides the intelligence to cause the manipulator or motor to perform in the manner specified by the user. The Canterbury Delta robot was built with donated step motors and HD65 Digiplan drives. Such a drive system is not ideal for such an application but did permit the robot to be built and static calibration research undertaken (Lintott and Dunlop 1997).

The existing set of motor control systems for the Delta robot use a dedicated IFX indexer to control each axis of movement. A single RS232 port is employed to link up the three IFX cards which are daisy-chained as shown in figure 1.0, to allow simultaneous control of three axes to coordinate motion control. However, serial linked indexers are slow in communicating with each other and need a stop point between each straight line movement to ensure coordinated motion. As a result, this controller system can only perform point to

point motion profiles, the production of circles, arcs and curves is choppy because of the numerous stops. Apart from that, the indexer is designed for trapezoidal motion profiles. This profile involves instantaneous changes in acceleration, which produces large jerks and vibrations during the transition stages. The jerk needs to be minimized because it causes vibration, mechanical joint wear and poor target accuracy. In conclusion, an unsuitable controller was used for the current robot control system. It is not suitable for the simultaneous control of multiple axes that is needed to create continuous movement along complex paths. Therefore, the replacement of the existing controller system is essential if the Delta walking machine is to perform a complex walking path.

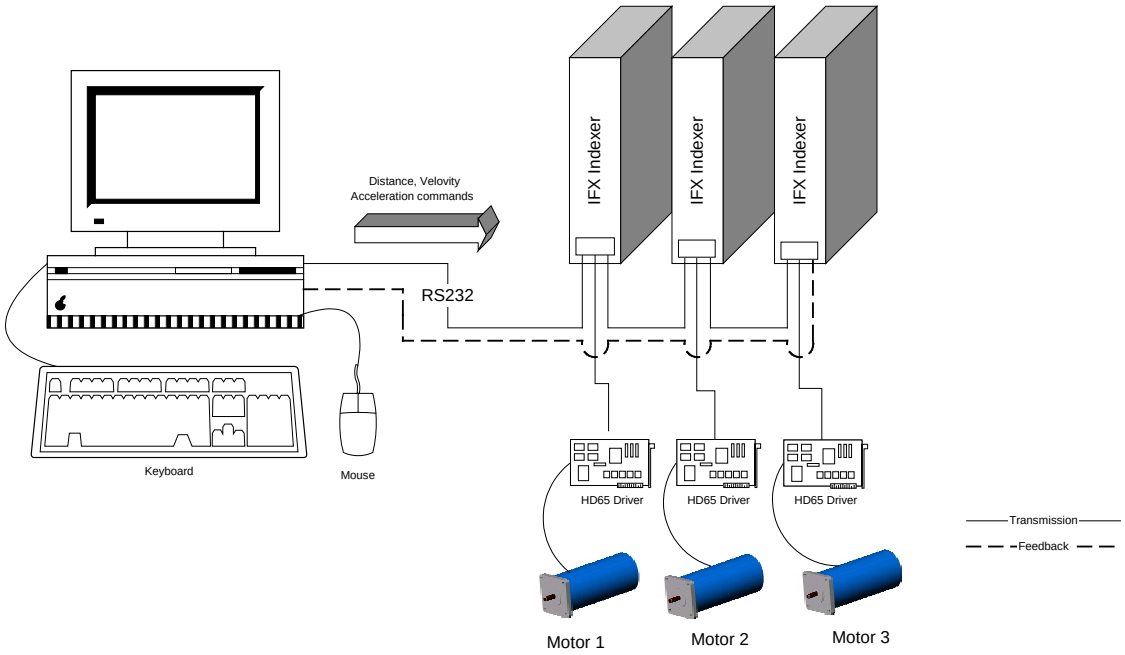


Figure 1.0. The existing control system layout.

To enable the Delta walking machine to perform a continuous movement profile, a versatile and powerful DSP (TMS320F240) is introduced into the control system. The TMS320F240 digital signal processor (DSP) is a single chip controller with multiple capabilities. It includes the following features:

1. 20 MIPS DSP core for real time processing.
2. Integrated with a motor control event manager, which supports up to 12 pulse width modulation (PWM) outputs with features that includes three

timers, nine comparators, dead-band generation logic and a state-space vector PWM generator.

3. SPI and SCI for serial communication.
4. 28 bidirectional I/O pins.
5. Watchdog timer.
6. Integrated dual 10 bit A/D converters.
7. 128k words SRAM data/program memory (external).

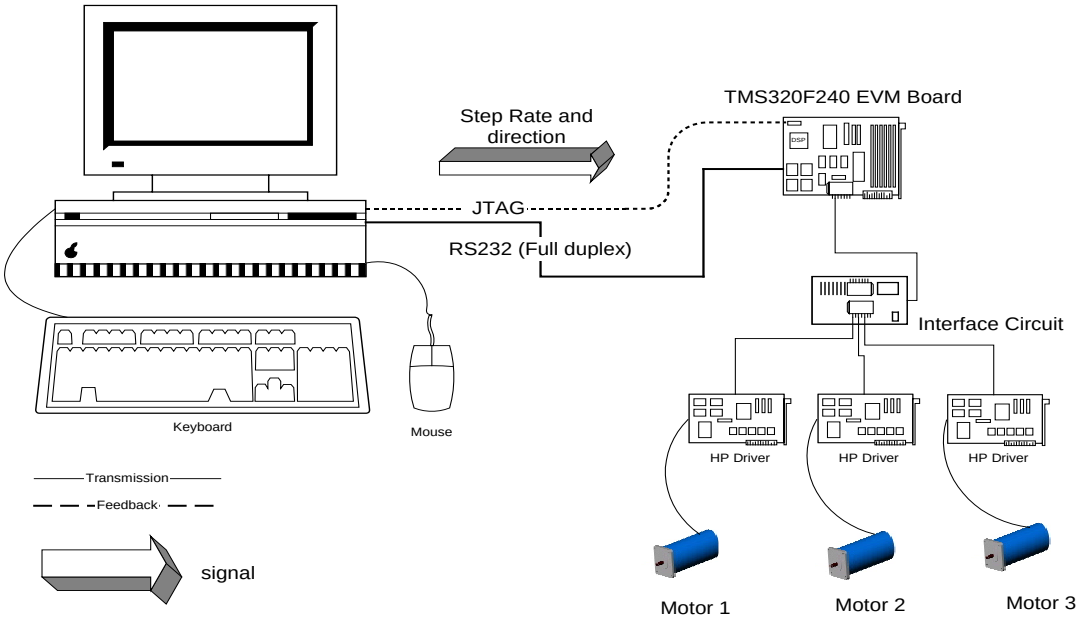


Figure 1.1. The new control system layout.

1.2 Walking Machine description

The Delta walking machine in the laboratory is approximately 1.7 meters long, 1.7 meters wide and 1.6 meters height as illustrated in figure 1.2. It consists of three major parts which are the 3 supporting legs, a 3 dof parallel Delta robot and a tripod end-effector (foot). The tripod end-effector is attached beneath the travelling plate of the parallel robot. Furthermore, the three identical supporting legs are bolted onto the base platform of the parallel robot in opposite orientation with the end-effector. As a result, the foot and the legs are basically in perspective and oriented in contraposition. This configuration allows

the robot to attain a larger walking space, and minimizes the danger of collision between legs and foot during walking.

Each parallel kinematic chain is actuated by a 2 kW hybrid stepper motor that is rigidly mounted onto the base platform. Each motor is limited to 6.5 Nm but is capable of supplying maximum torque up to 13.0 Nm. In addition a harmonic gear drive, with a reduction ratio of 80 to 1, is employed on each chain so that the robot's payload capacity is up to 370kg. In fact, the weight of the whole robot is only about 350 kg. The robot is designed to be able to lift itself up, and walk slowly in any direction.

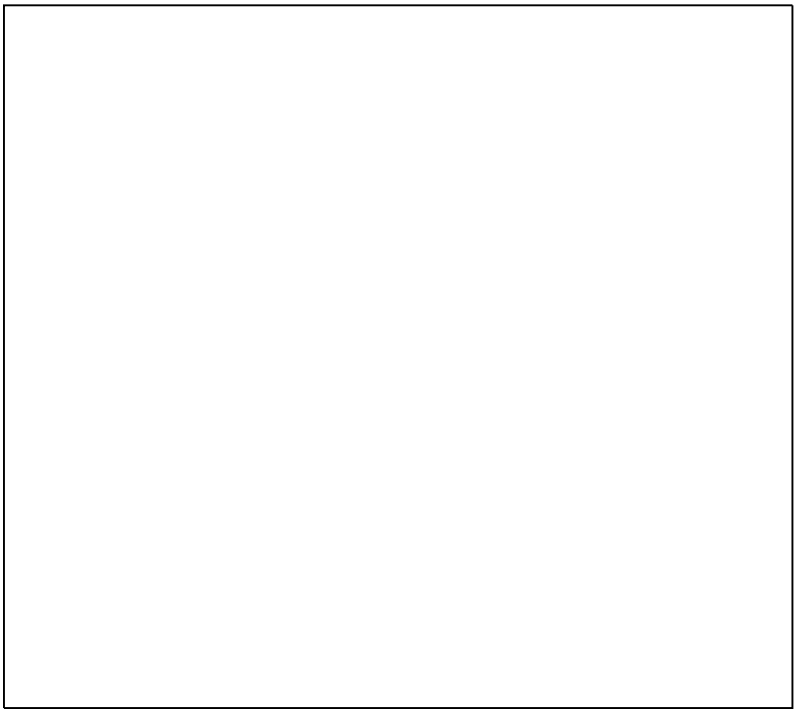


Figure 1.2. The walking robot with three degrees of freedom.

1.3 Locomotion of the walking machine

Strictly speaking, the Delta walking machine can be categorized as a six legged walking machine because it possesses a tripod foot on the travelling platform and three fixed supporting legs on the base platform. Step walking is achieved simply by moving one of the platforms while the other platform is on the ground. Bearing in mind that those two

platforms always stay parallel to each other while walking. The gait of the machine is similar to the parawalker [23]. The omnidirectional walking sequences of the walking machine can be divided in four phases, and depicted in figure 1.3.

Phase (1) : The tripod end-effector is ready to make a touch down.

Phase (2) : An adequate torque is delivered by the step motors to lift the supporting legs of the base platform off the ground.

Phase (3) : The base platform is shifted across to a destination then lowered to place the three supporting legs at a new location on the ground.

Phase (4) : The tripod end effector is shifted to a new position.

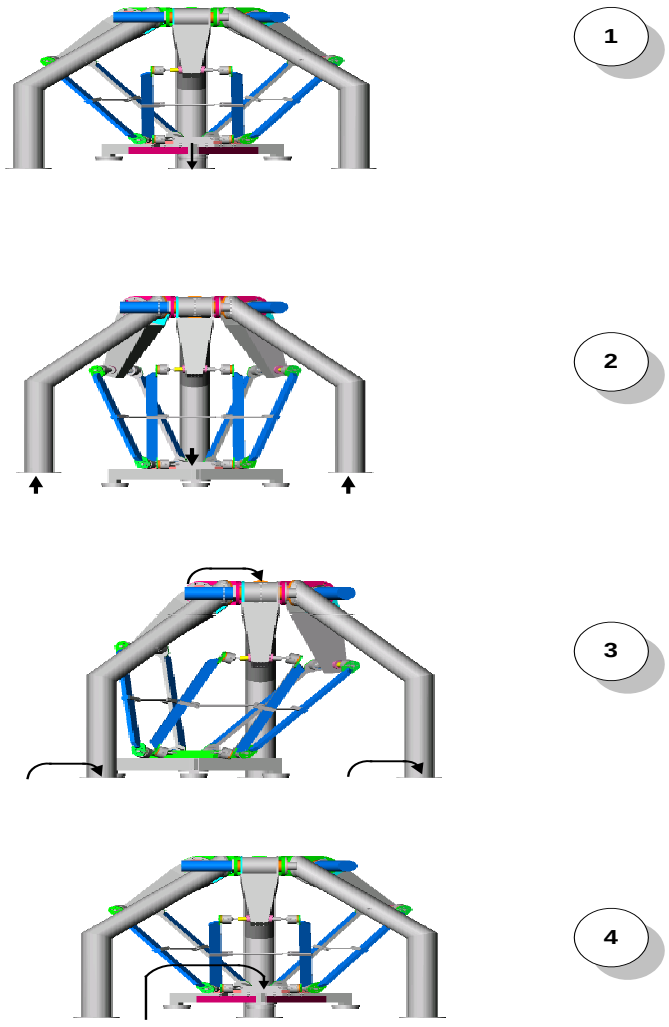


Figure 1.3. Gait of the walking machine.

1.4 The goals of the project

In order to develop a new style of walking machine based on the Delta robot, the following objectives are proposed:

1) Use a DSP (TMS240F240) to control the three actuators in synchronism.
2) Use of continuous paths in trajectory planning.
3) Design of an end-effector to support the omnidirectional step walking.

1.5 Thesis Outline

The thesis is organized into 8 chapters and 4 appendices. Chapter 2 reviews the advantages and disadvantages of the Delta robot, the current research on the parallel mechanisms and some industrial applications, followed by a description of the major components of a Delta robot and its kinematics. In chapter 3, a brief review of the design philosophy of walking robots and the problems associated with legged robots are presented. At the end of the chapter, two examples of walking robots with parallel mechanisms are highlighted. Then in chapter 4, force analysis is applied to determine the maximum step length and the walking space of the Delta walking machine. In chapter 5, the mathematical modelling of the trajectory profile is developed, and an off-line optimal continuous path planning method at the joint level is presented and then illustrated with examples. The hardware is described and software synergy of the motion control system is given in chapter 6. An example is presented to illustrate the synchronization of control on three axes during the real time implementation. In chapter 7, the problems that are associated with step walking are highlighted and some solutions are given. In the concluding chapter, the results achieved are discussed and the areas for further research are identified.

Chapter 2 Literature review

2.0 Overview

In this chapter, the advantages and disadvantages of a robot with a parallel mechanisms are described. A review of different parallel mechanisms and their industrial application is presented. In section 2.4, the basic components of the Canterbury Delta robot are described. The chapter closes with the development of a method of evaluating the kinematics solution of the Delta robot.

2.1 Introduction

Delta robot 4 is a 4 D.O.F parallel manipulator which was originally invented by Clavel [37] in 1988. It was designed to serve in the electronic, food and pharmaceutical industries which required a high level of hygiene and reliabable standards for of the products. Figure 2.1 shows the characteristics of the parallel manipulator.

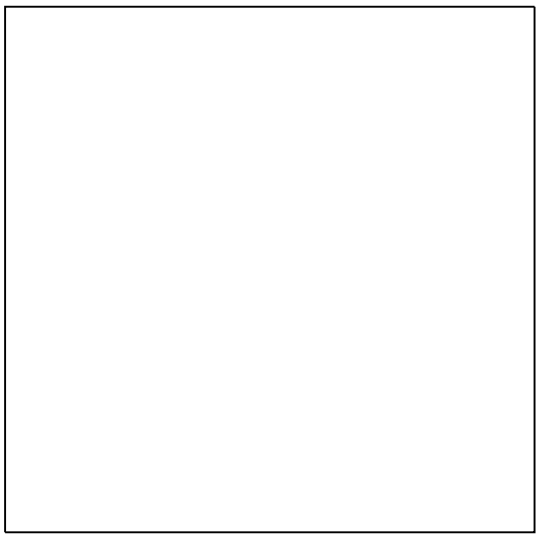


Figure 2.1. The "Delta" robot of Clavel.

The Delta robot constitutes technological innovation in the robot industry and has attracted much research interest since it was introduced to the industry for pick and place operations. The simple reason for this is that the robot offers the following advantages when compared with standard serial robots:

1. Inverse kinematics and direct kinematics can be solved easily.
2. The position and orientation of the moving platform are uncoupled.
3. Very high accelerations are possible due to the light weight of the moving parts.

Furthermore, the robot is a closed link structure as shown in figure 2.1. This structure allows the actuators to be fixed on the base whereas the traditional serial robots is usually actuated at each joint along the serial linkage. For a serial robot, each axis has to drive not only the weight of the driven link but also the load of the servomotors for the following links. This results in moving heavy masses and a low dynamic motion response especially on a big machine. The links of the parallel robot do not need to carry the load of actuators so the parallel links can be built as lightweight structures. Therefore, the delta robot provides multiple advantages when compared with serial robots. Those advantages are easy construction, higher stiffness of the structure, higher speed operation and better load to weight ratios.

On the other hand, Clavel pointed out that the delta robot suffers the following disadvantage.

1. The working volume of the robot is constrainted by its mechanical construction.
2. Singularity configurations define the workspace.

2.2 Further developments from the Delta Robot

In recent years, parallel manipulators have been studied extensively and several parallel mechanisms have been developed by numerous researchers [34,38,44,56] in response to different industrial applications. The new features of those manipulators are illustrated in the following diagrams.

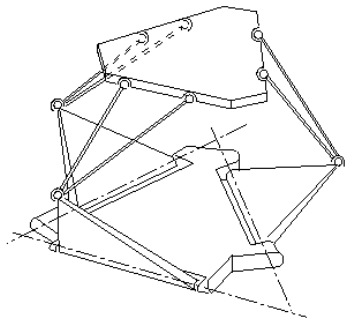


Figure 2.2. A structure proposed by Lambert.

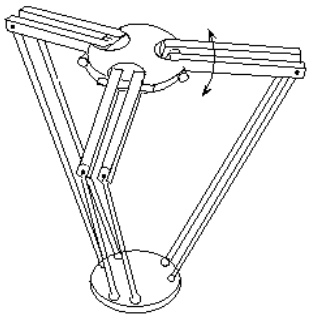


Figure 2.3. The "Hexa" robot of Pierrot (1991).

Error! No topic specified.

Figure 2.4. The Kiwibot from Dunlop (1992).

These robots are built with a combination of spherical joints and revolution joints on linkages between the base platform and moving platform thus forming closed chain loops to the manipulator. Although the mechanical structure of the kinematic chains of many parallel robots are similar, they usually have different mobilities. Lambert's parallel robot and Kiwibot both have 3 dof where as the Hexa robot has 6 dof. Due to their different motions, Lambert's parallel robot and Kiwibot are classified as spherical robots while the Hexa robot is clasified as a Cartesian robot.

2.3 Industrial Applications

The 3 dof parallel delta robot has gained its reputation in the industrial application due to its many advantages. As a result, the parallel robot's application field has been extended beyond the production line and it is accepted widely in the agricultural [44], and medical fields [45]. The following diagrams illustrate the application of parallel robots in different fields.

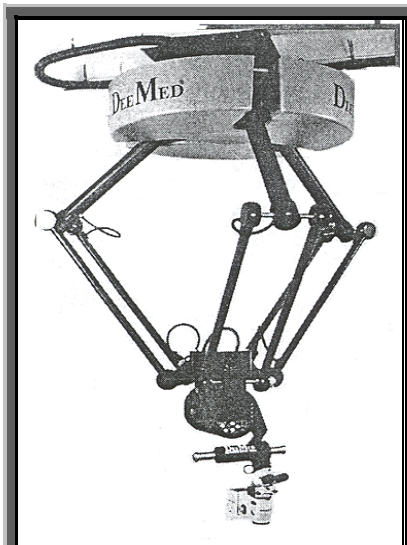


Figure 2.5. The Surgiscope, from Elekta. It is used for helping surgeons navigate through the brain during surgery.

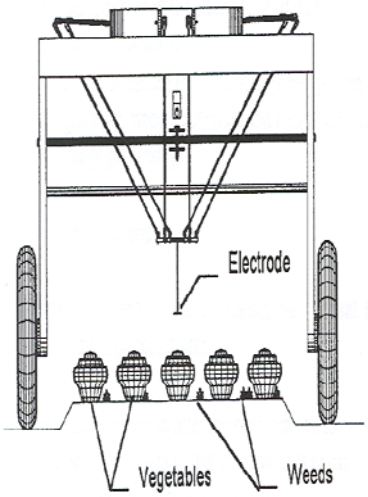


Figure 2.6. Mobile manipulator "Patchwork". This is a non-chemical weed controller designed for eliminating the weed by electrocution.

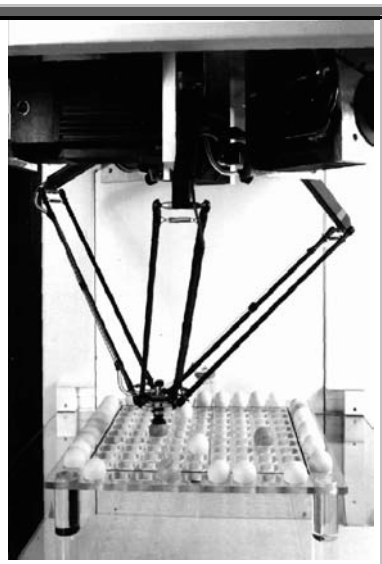


Figure 2.7. A Delta robot for "pick and place" operations.

2.4 Delta Robot Description

The Canterbury Delta Robot was built about the same time as the Delta 4 Robot but on a much larger scale. The robot was constructed with a base plate, a travelling plate and three identical kinematic chains. Each chain is made up of two parts, an upper arm and a lower parallelogram. Each upper arm is attached to the base as a revolution joint and actuated by a 2kW STEBON step motor (SDT1103-150-120). The stepper motors are securely mounted onto the fixed base plate and move the arm via an 80 :1 harmonic drive. The combined motion of the three actuators regulates the translational motions of the moving plate. The parallelogram is linked between the robot arms and the travelling plate by spherical joints as shown in figure 2.8. Thus, the movement of the travelling plate always stays parallel with the base platform within the primary working volume.

Error! No topic specified.

Figure 2.8. The Delta robot with 3 DOF.

2.5 Kinematic Solutions

The direct kinematic solution given here takes the same form as that of Clavel [37], and Pierrot and Fournier [43]. The inverse solution is formulated from the intersection of the locus of point B_i as it moves on a sphere about C_i and a circle about A_i (i = 1,2,3; refer to figure 2.9). The solution bears a similarity to that of Pierrot, Fournier, and Dauchez except it is in a closed form solution.

2.5.1 Direct Kinematic Solution

Since the top and bottom links (through B_i and C_i) of a parallelogram linkage are always parallel to the revolution axis (through A_i) of the connected driver on the base, the plate of the moving platform is always parallel to the base plate within the primary working volume. The orientation of the moving plate is always the same as the base so line OA_i is always parallel to the line PC_i.

Step 1) The linkage A_i, B_i, and C_i is displaced a distance along the direction C_i P so that A_i move to A'_i a distance r from A_i along the line A_iO. B_i moves to a known position B'_i and C_i moves to P.

Step 2) Construct 3 spheres of radius m centred on the B_i.

Step 3) Solve for the lower intersection point P of the 3 spheres

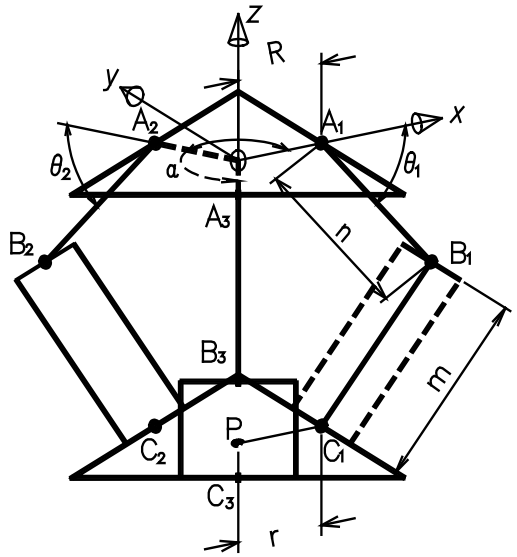


Figure 2.9. Kinematic model of the DELTA robot (from Lintott & Dunlop [56]).

2.5.2 Inverse Kinematic Solution

Given the R_{xyz} coordinates of point P (refer to figure 2.10), the effector angles may be calculated. The method is described as follows:

Step 1) Draw a sphere of radius m centred on P.

Step 2) An arm of length n centred on A_i traces on arc that intersect the sphere at 2 points : θ_{i1} & θ_{i2}.

Step 3) $\theta_i = \min(\theta_{i1}, \theta_{i2})$ to keep the arm out of the working volume.

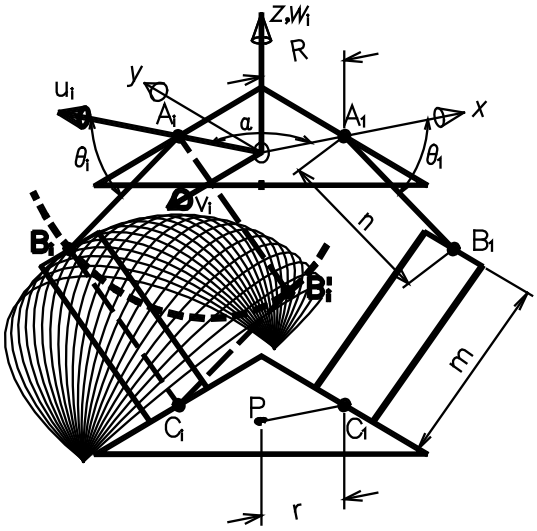


Figure 2.10. Kinematic model for inverse solution (from Lintott & Dunlop [56]).

Chapter 3 Walking robot overview

3.0 Overview

In this chapter, we describe the advantages of the legged walking robot and the type of robot that was developed. In sections 3.1 and 3.2, we discuss the design philosophy and problems of developing a legged robot. At the end of the chapter, two examples of six legged robots with parallel mechanisms are briefly described.

3.1 Introduction

Interest in walking machines has increased especially in walking robots with legged locomotion. With the ability to implement complex numerical control, the legged locomotion machines have potential not only to traverse irregular and unstructured terrain, but also to negotiate and step over obstacles. In addition, walking robots have the ability to move over and to carry out tasks in the hazardous environments which are not suitable for humans such as nuclear power plants, active volcanic zones and etc.

In recent years, several prototype of walking robots have been proposed with different leg mechanisms. The number of legs on a robot can be varied from one-legged hopping machines developed at MIT [57], right up to eight legged robot developed at Portsmouth [19]. However the most common and widely studied walking robots are biped, quadruped, and hexapod type robots. Out of these, the six-legged walking robots (hexapod type robots) are more suitable for walking over rough terrain because they are able to maintain their posture stability with at least three out of six legs touching the ground. The other types of the robots tend to be unstable and have difficulty in controlling the position of their center of mass, especially when the controller is over extended or fails.

3.2 Design philosophy

For designing a walking machine, the most crucial decision is to determine the number of legs for the robot. There is a tradeoff between motion control of the legs and the stability of a walking robot. As the number of legs increases, the control system becomes more sophisticated but not the static stability. According to Fukuda [12], the optimal number of legs for a walking machine is six. This is because the robot with six legs can provide two support tripods for static stability, and also use more than three legs to achieve the static balance of the center of mass while walking over irregular terrain.

Most of the legged robots are developed from biological model. The design and control of the legged machines are directly based on studies of insect walking. These evolutionary approaches are appropriate because the insects have passed the hard test in environments unfriendly to survival. For instance, Weidemann [16,17] designed a walking machine based on the stick insect (Carausius Morosus). Nelson [18] also developed a cockroach-like hexapod robot with 24 degrees of freedom.

3.3 Problems on legged robots

The majority of the animal-like legged robots require more than three legs in order to achieve a static stability. Each leg normally possesses 3 degrees of freedom to provide a better walking performance and higher locomotion capacity in a rough environment. This configuration results in many degrees of freedom being required for building a walking robot with six legs. For example, a six-legged walking robot has at least 18 dof (one central body with 6 legs and each leg has 3 dof). Because of this, the control system requires a dedicated motion process for each of the 18 axes and a master process to supervise the 18 local processes as well as perform other higher level tasks such as gait pattern, velocity profile of the body, stability and obstacle detection. This leads to quite complex control algorithms and control structures. These processes may be run on a single powerful processor or may be distributed amongst several processors. Thus, the penalty of having a robot with large number of dof is that the robot invariably moves

slowly, is heavily dependent on human operators, and may be unreliable [30]. If a leg mechanism with 2 dof is used, the foothold regions of the legs are limited. As a result the walking machine may lose its terrain adaptability and mobility in a rough environment.

3.4 Six-legged walking robot with parallel mechaniam

Ideally, the walking machine should be lightweight with only a few dof. Parallel robots such as Stewart platforms, S/P hybrid platforms and “Hexa” robots can meet the requirements. Ota [23] has developed two types of 6 DOF walking robots with different parallel mechanism. The Parawalker I and Parawalker II are shown in figures 3.1 and 3.2 respectively. Both robots can walk in any direction and have a high terrain adaptability.

Figure 3.1. A walking machine with Stewart Platform

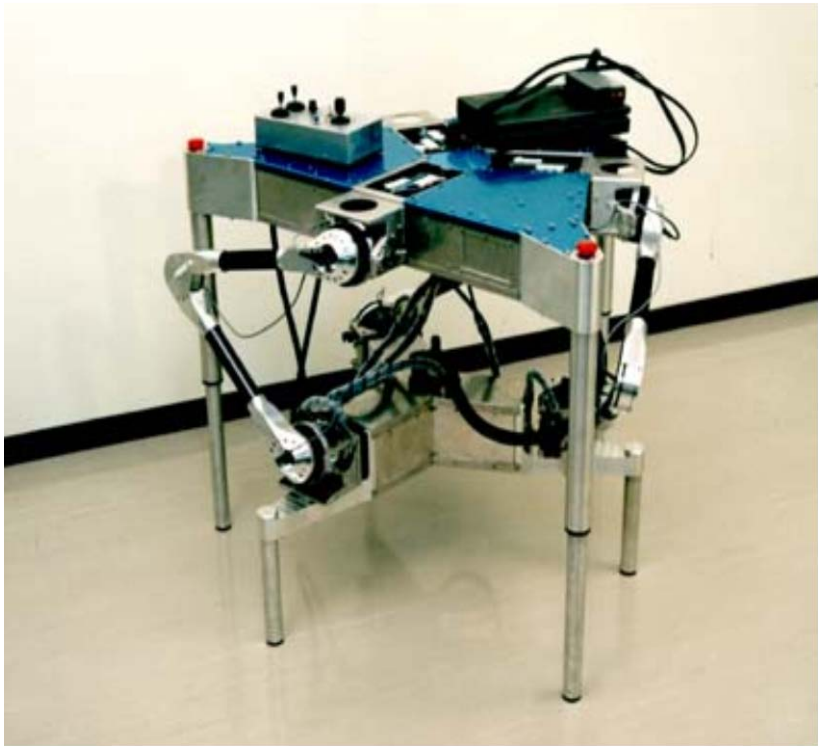


Figure 3.2. A walking machine with S/P Hybrid Platform

Chapter 4 Step Walking Analysis

4.0 Overview

The factors that affect the stability and mobility of the Delta walking machine are discussed in this chapter. The formulation of the static force analysis of the Delta walking machine are presented in section 4.4, then a software algorithm for determining the Delta walking machine's foothold region is developed. The algorithm implementation is described in section 4.6 and the simulation results are presented in section 4.7. Finally, a tripod foot is designed based on the foothold region analysis of the Delta walking machine.

4.1 Introduction

The fundamental problems associated with a walking robot are its mobility and its stability. Thus, robot stability and mobility are both discussed in this chapter. Without both, it is unrealistic to transform the Delta robot into a walking machine. For static stability, the number of legs in contact with the ground should be always more than three, and the center of gravity of the robot should remain within the polygonal area of grounded contact. These two factors are the primary guidelines for the foot design. Furthermore, the mobility of the Delta walking machine is dependent on the workspace of the parallel manipulator and the available output torque from each motor and gearbox. These two parameters can be determined by using a static force analysis. Since step walking is performed at low acceleration and low speed, the static force analysis is adequate for determining the Delta walking machine‘s foothold region i.e. where the robot can safely put its legs down and perform a step.

4.2 Modelling Assumptions

The mathematical model for the static force analysis uses the same assumptions as Pierrot [43], they are:

1. The rotational inertias of the forearms are neglected.
2. The mass of the lower arms are separated into two portions and located at the two ends of the parallelogram.
3. The friction effects on the ball joints are neglected.
4. The output torque from the motor being transmitted to the gear drive is 100%.

4.3 Static model of the Delta walking machine

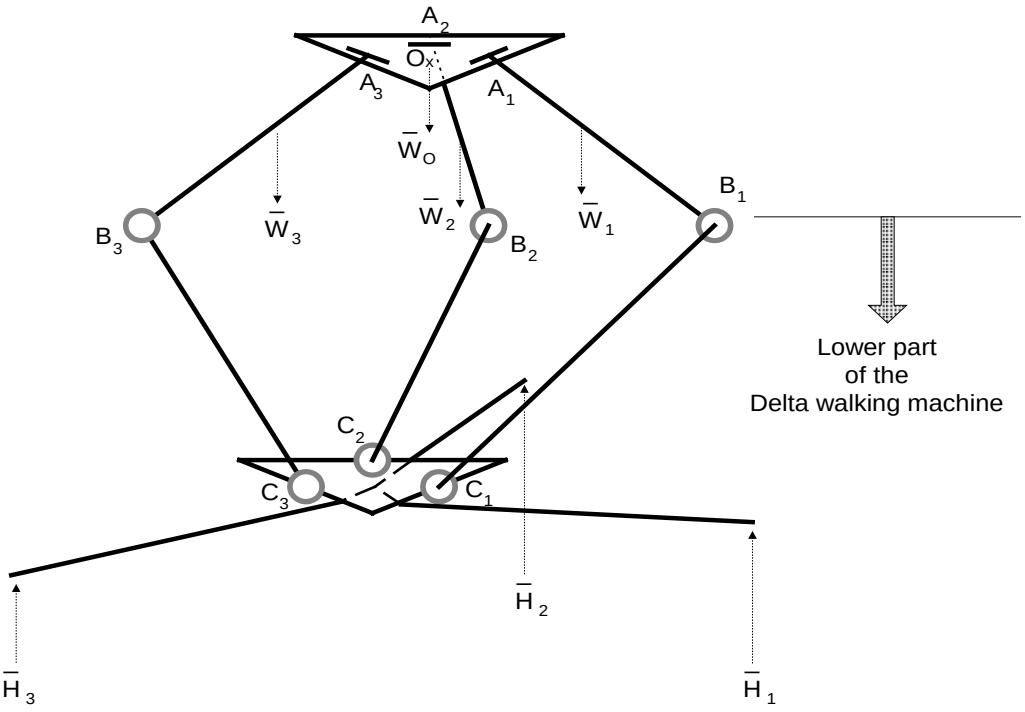


Figure 4.1. The forces acting on the Delta walking machine.

Figure 4.1 shows that the end-effector (the foot) is held on the ground allowing the base platform to move to a particular position. The mechanism of the robot is reversed so the

base platform interchanges with the travelling platform. Indeed, this configuration causes the parallel robot to lose its advantages of having the actuators on the base. The torque on each axis not only needs to support the weight of the upper body (220kg) but also the weight of the actuators (30kg). Therefore, when the base platform moves to a particular location, the stepping motors need to provide the minimum torque required for the static stability in that position. If the stepping motors fail to supply the required torque, the robot will be out of balance and fall to the ground as it nears the position. For the static model as shown in figure 4.1, the weight of the lower part of the Delta walking machine (i.e. 3 parallelogram arms, the travelling plate and the foot) is not taken into account since it is stationary and sitting on the floor. Therefore, the required torque needs to support only the weight of upper part of the body (e.g. 250 kg). Furthermore, each parallelogram arm is replaced by a single rod in order to develop a simple static model.

In addition, the required torque ($\tau_{a,\ i=1,2,3}$) must not exceed the constraint of the continuous torque capacity of the harmonic gear drives. The harmonic gear drive is allowed to handle a maximum torque of 560 Nm even though more torque is available from the stepping motors. If the harmonic gears are overloaded, the fatigue life of the gear drive will be shortened (e.g. 2 revolutions at 1100 Nm before failure). To avoid this, the foothold regions of the walking machine should be maintained within the maximum allowable torque capacity of the gear drive. Hence, the maximum torque from each control axis needed to support walking is restricted to 560 Nm.

4.4 Formulation of the static equation of equilibrium

Given that the lower three arms are fixed in a position, an imaginary cutting plane is made at the three spherical joints B_i in order to solve the joint reaction forces ($F_{B_i\ \ i=1,2,3}$) on the spherical ball joint as shown in figure 4.2. There are three unknown reaction components ($F_{B_i}(x),\ F_{B_i}(y),\ F_{B_i}(z)$) on each spherical joint which makes a total of 9 unknown reaction components. Since the structure is in equilibrium, the sum of these forces vector should

equal to the weight of upper part of the body. This single vector equation leads to three scalar algebraic equations. Solving these equations simultaneously yields the magnitudes of the force (K_{B_i} $_{i=1,2,3}$) on each spherical joint. Then the force reaction on each spherical joint is determined by multiplying the magnitudes of the force with a unit vector that pointed from C to B i.e. $\vec{F}_{B_i} = K_{B_i} * \vec{CB}_i$. The required torque from each control axis can be evaluated by summing the moment about the revolution joint through A_i.

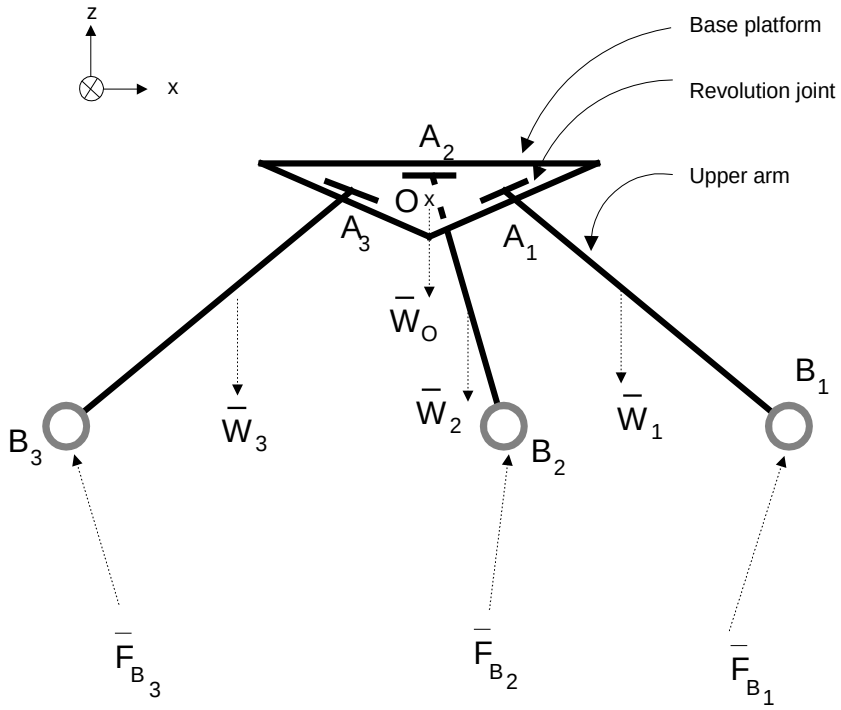


Figure 4.2. The free body diagram on the upper part of the Delta walking machine.

The force on the each spherical joint is expressed as:

$$\vec{F}_{B_i} = K_{B_i} * \vec{CB}_i \qquad (4.1)$$

Where $\vec{F}_{B_i}$ is joint reaction force vectors on each spherical joint. $\vec{CB}_i$ is a unit vector pointed from C to B, and K_{B_i} is magnitude of joint reaction force at point B.

The sum of the force vectors on the upper part of the Delta walking machine (refer to figure 4.2) equals a null vector because the structure is in equilibrium. Thus,

$$-\sum_{i=1}^{3}\vec{F}_{B_i}+\sum_{i=1}^{3}\vec{W}_i+\vec{W}_o=0 \qquad (4.2)$$

Where W_i is the weight of the upper arm, i = 1, 2, 3.

W_o is the center of gravity of the upper part of the Delta walking machine.

F_{B_i} is joint reaction force vectors on each spherical joint.

Substituting equation (4.1) into (4.2) yields:

$$\vec{W}_o+\vec{W}_1+\vec{W}_2+\vec{W}_3=K_{B_1}*\vec{CB}_1+K_{B_2}*\vec{CB}_2+K_{B_3}*\vec{CB}_3$$

Rearrange the above equation in a matrix form, thus

$$[W]=[k]\,[CB] \qquad (4.3)$$

Where

$$\left[W\right]=\vec{W}_o+\vec{W}_1+\vec{W}_2+\vec{W}_3=\begin{bmatrix}W_x\\W_y\\W_z\end{bmatrix}=\begin{bmatrix}0\\0\\W\end{bmatrix} \quad [k]=\begin{bmatrix}K_{B_1} & K_{B_1} & K_{B_1}\end{bmatrix}$$

and $$[BC]=\begin{bmatrix}CB_{1X} & CB_{1Y} & CB_{1Z}\\CB_{2X} & CB_{2Y} & CB_{2Z}\\CB_{3X} & CB_{3Y} & CB_{3Z}\end{bmatrix}$$

x, y and z represents the coordinate axes

The magnitudes of the reaction forces on each spherical joint are solved as follows:

$$[k]=[W]\,[CB]^{-1} \qquad (4.4)$$

Then the magnitudes of the reaction forces [k] and the unit vector $\vec{CB}_i$ are substituted into equation 4.1 to calculate the force reaction on each spherical joint.

Force balance on the Upper Arm

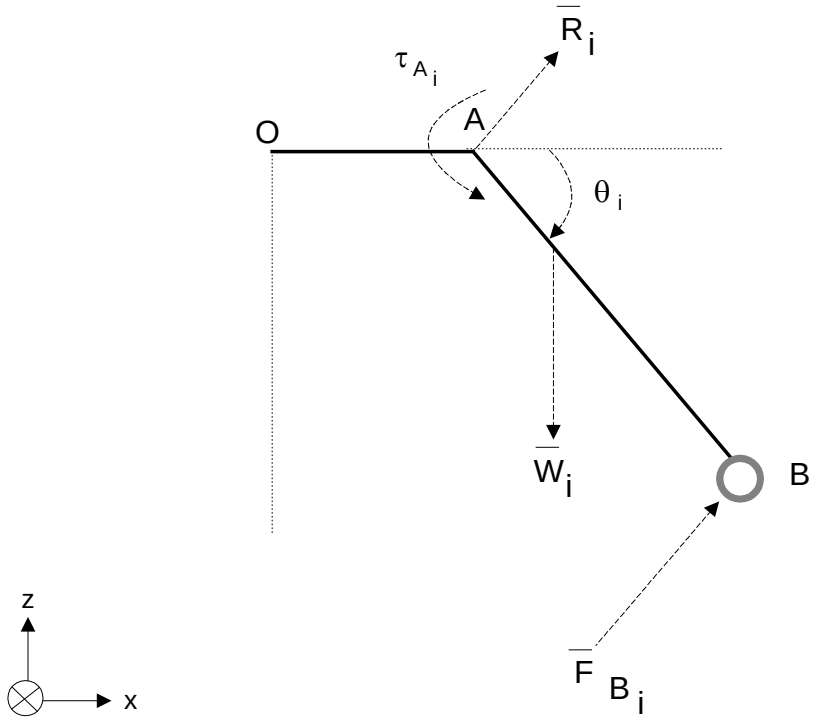


Figure 4.3. The free body diagram on the upper arm.

The moment about the y_i axis through point A is dependent on the weight of the upper arm and the spherical joint reaction force vector. Since the joint reaction forces have been found, the required torque on each axis (τ_{A_i}) can be calculated from the equation as follows:

$$\vec{\tau}_{A_i} = [\vec{F}_{B_i} \text{ x } \vec{AB}_i] + [\vec{W}_i \text{ x } \mu_i * \vec{AB}_i] \qquad (4.4)$$

Where μ_i is a constant e.g $\mu_i = 0.63$. (refer to Appendix B)

$\vec{F}_{B_i}$ is joint reaction force vectors on each spherical joint.

AB_i is the position vector pointed from A_i to B_i.

W_i is the weight of the upper arm.

Forces balance on travelling plate

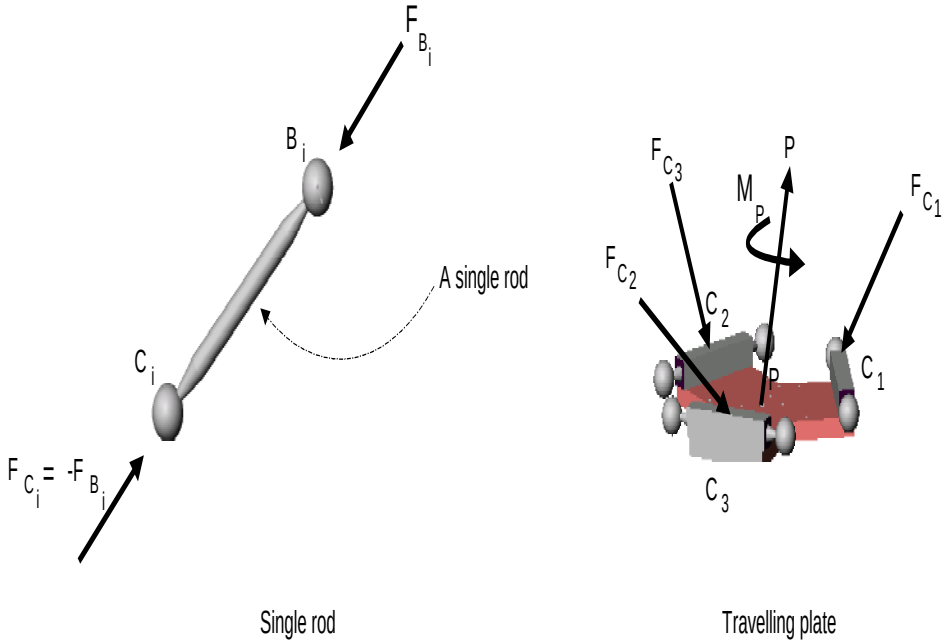


Figure 4.4. The free body diagram on the travelling plate and a single rod.

The sum of the all the forces on the travelling platform should equal to zero. Thus

$$\sum_{i=1}^{3} \vec{F}_{C_i} + \vec{P} = 0 \qquad (4.5)$$

Where P is the reaction force on the center on travelling plate.

F_{Ci} is joint reaction force vectors on each spherical joint C.

The moment (M_p) at the center of the travelling platform is

$$\vec{M}_p = \sum_{i=1}^{3} [\vec{F}_{C_i} \times \vec{PC}_i] \qquad (4.6)$$

Where PC_i is a position vector pointed from P to C_i.

F_{Ci} is joint reaction force vector on spherical joint C_i.

Forces balance on the foot

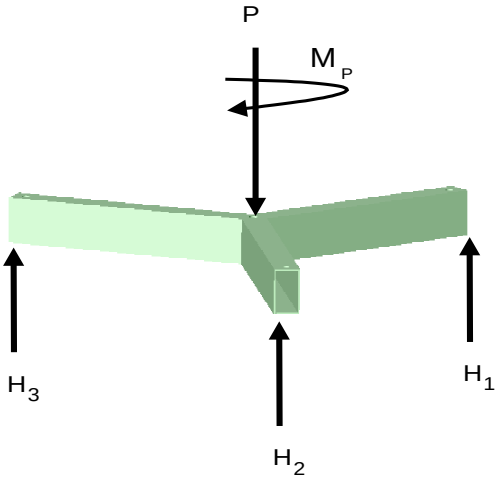


Figure 4.5. The free body diagram on a tripod foot.

The sum of the any force vector on a tripod foot is equal to zero. Thus

$$\sum_{i=1}^{3} \vec{H}_i + \vec{P} = 0 \qquad (4.7)$$

Where P is the reaction force on the center on a tripod foot.

H_i is the reaction force vector on footpad (i = 1, 2, 3).

The moment (M_p) at the center of a tripod foot is

$$\vec{M}_p = \sum_{i=1}^{3} \left[\vec{H}_i \times \vec{L}_i \right] \qquad (4.8)$$

Where L_i is the length of the rectangular tube on the foot.

H_i is the reaction force vector on each footpad.

4.5 Singularity

Singularities of the Delta robot are of the type where the mechanism gains extra transitory mobility so that the travelling plate (or nacelle) has the capability of movement without the movement of the actuators. Since the Delta walking machine is developed from the Canterbury Delta robot, the singularities of the Delta robot as showed in figure 4.6 need to be considered in the evaluation of the foothold regions in order to avoid the singularities during each step operation. When the walking machine is in a singular position, it will lose maneuverability and tip over unexpectedly since the position of the center of mass is approximately located at the center of base platform well above the foot. This can easily happen for cases (c) and (d). To avoid these two cases, the step movements need to be planned in advance to ensure that the walking machine is operated in safe regions well away from the singularities. The singularities shown in cases (a) and (b) can be simply avoided if all the step operations are performed below the home configuration ($z \leq$-1007 mm).

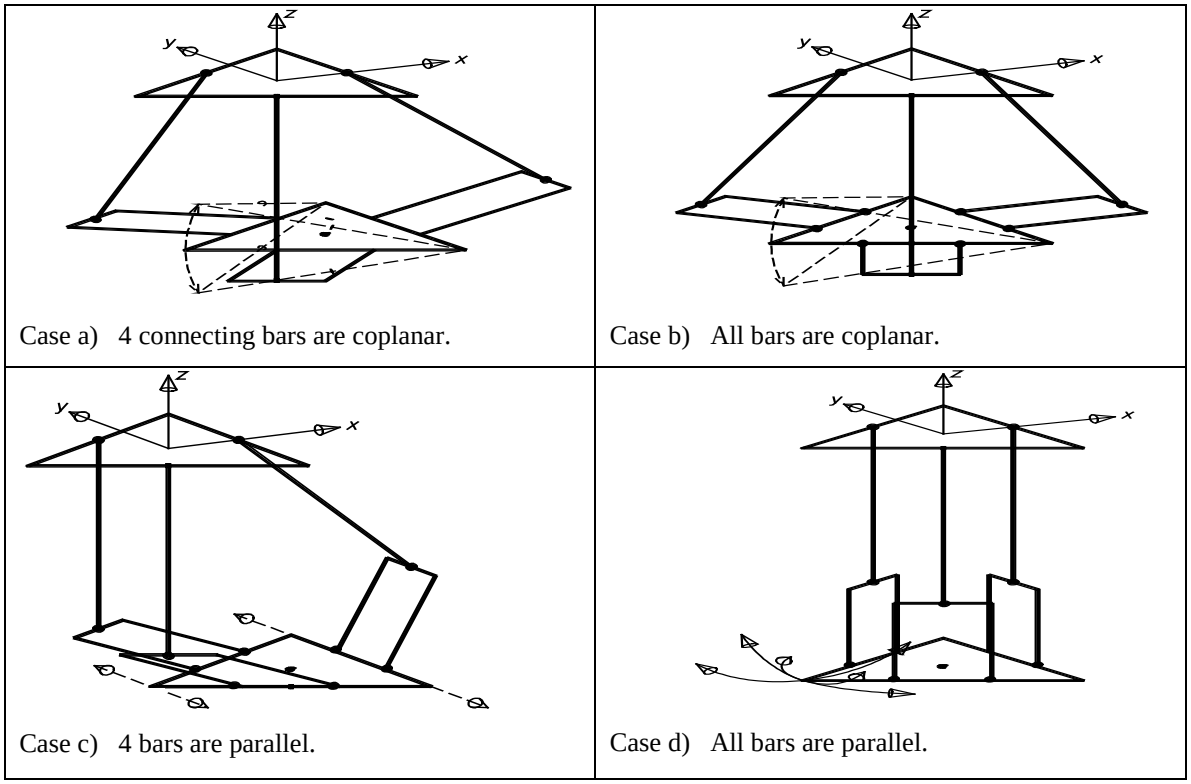


Figure 4.6. The types of Delta robot singularity.

4.6 Implemented algorithm

In order to evaluate the foothold regions of the Delta walking machine Walking Machine, an algorithm has been developed in the Matlab to search for the boundary limits of the regions. The procedure used to obtain the foothold regions for the worst case condition (i.e. when the Delta walking machine needs to move its base platform to a position) is illustrated in figure 4.8.

The Delta robot has a relatively small working envelope which is further reduced by using a range about the z-axis of -1100 mm to –1480 mm in order to reduce the required computation. The procedure starts by fixing the z position (z = -1480 mm) equal to z_{start}. Then a radial search is conducted in each z plane as shown in figure 4.7. The value of r_o is incremented or decremented for a particular angle value, and the required torque is evaluated at each coordinate point. When the required torque reaches the maximum allowable torque capacity of the harmonic gear drive (i.e. τ = 560Nm) or one of the joint angles ($\theta_{\{1,2,3\}}$) reaches 90 °, then the Cartesian coordinate of that point is recorded as a boundary for the step. The step direction angle value (ε) is then incremented by an amount ($\Delta\varphi$) and a new search starts from the last value of r_o at the boundary for the last angle value. Once the ε value reaches the value of $\pi/3$ (using $\pi/3$ instead of 2π because of the robot symmetry), the z position is incremented by Δz and a new z plane is created. The procedure is again executed to search for the boundary limit in the new z plane.

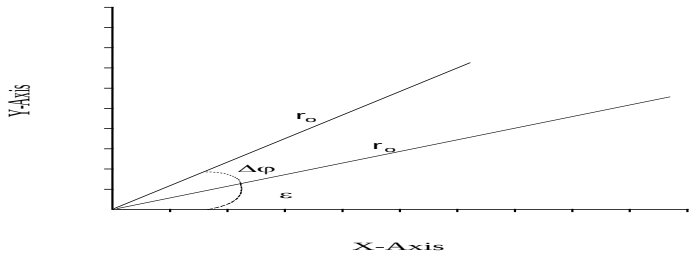


Figure 4.7. Boundary searching on the x-y plane.

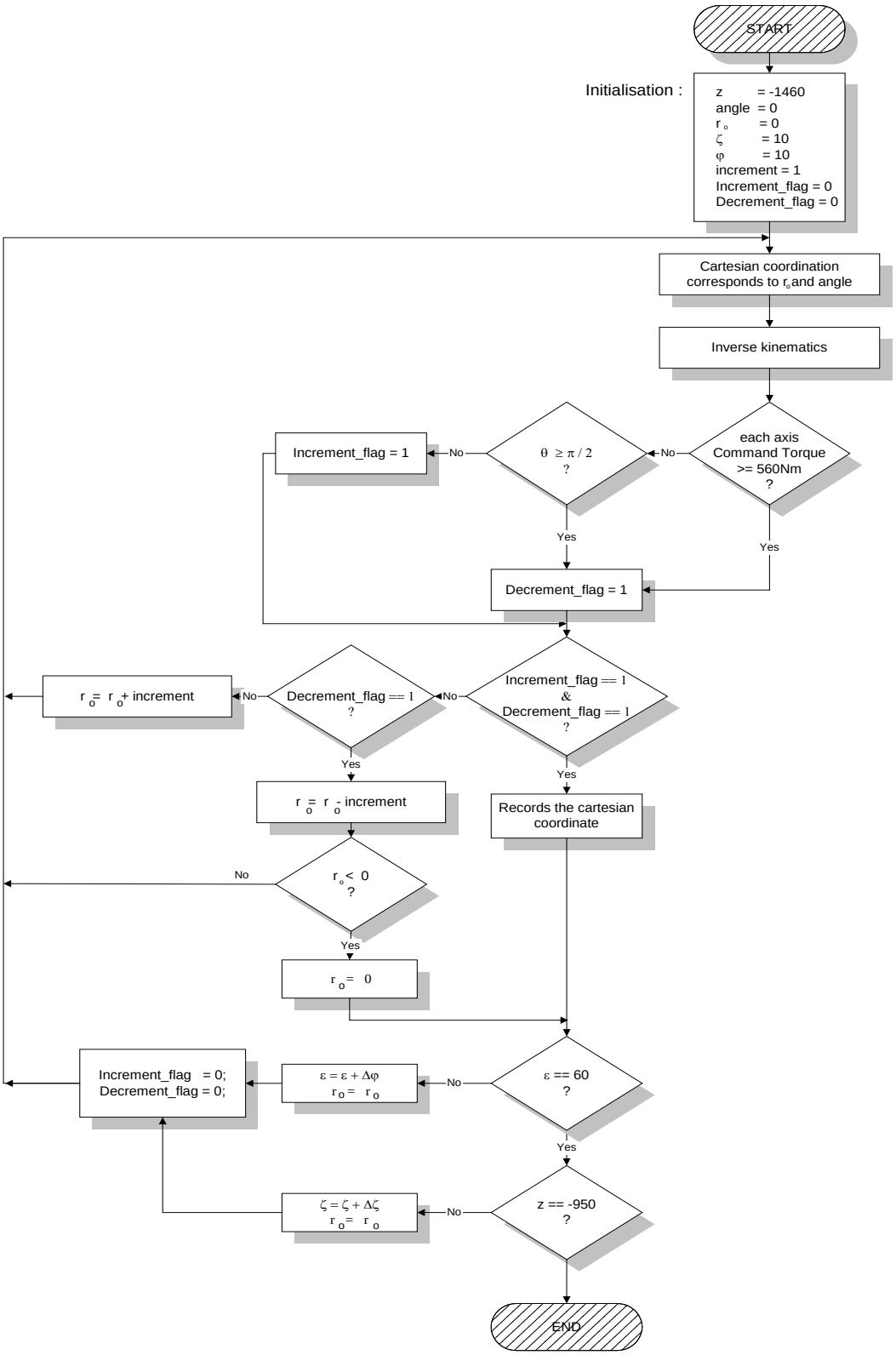


Figure 4.8. The flow diagram of the radial search.

4.7 Results

The foothold regions of the Delta walking machine are illustrated in figures 4.9. The boundary contour starts with a triangular cross sectional shape at the height of –1480mm, it becomes a circular shape when the height level reaches –1445m. After that particular circular contour, the contour becomes a triangular cross sectional shape again. When the height level is about –1355mm, the contour starts to take on a triangle shape. Finally, the contour ends with a single point at the height value of –1250mm.

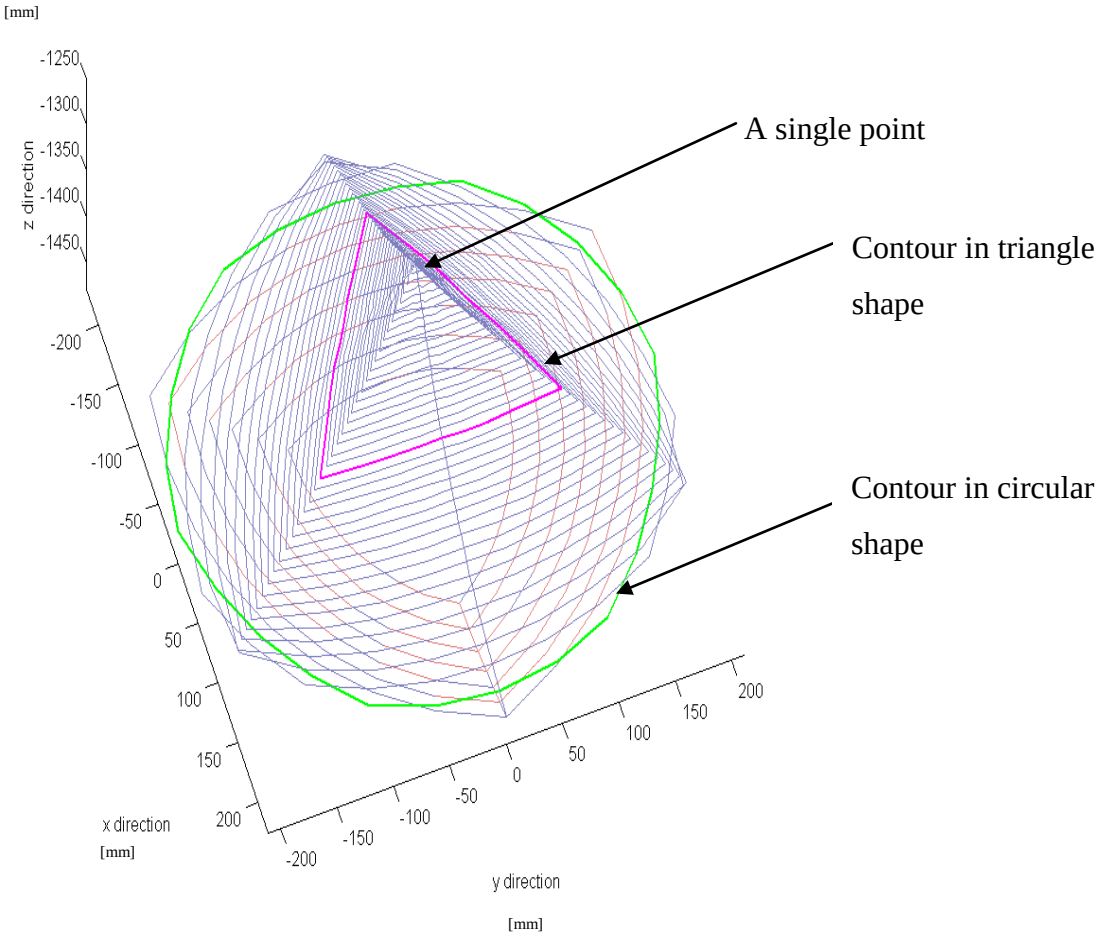


Figure 4.9. The foothold regions of the Delta walking machine (3D view).

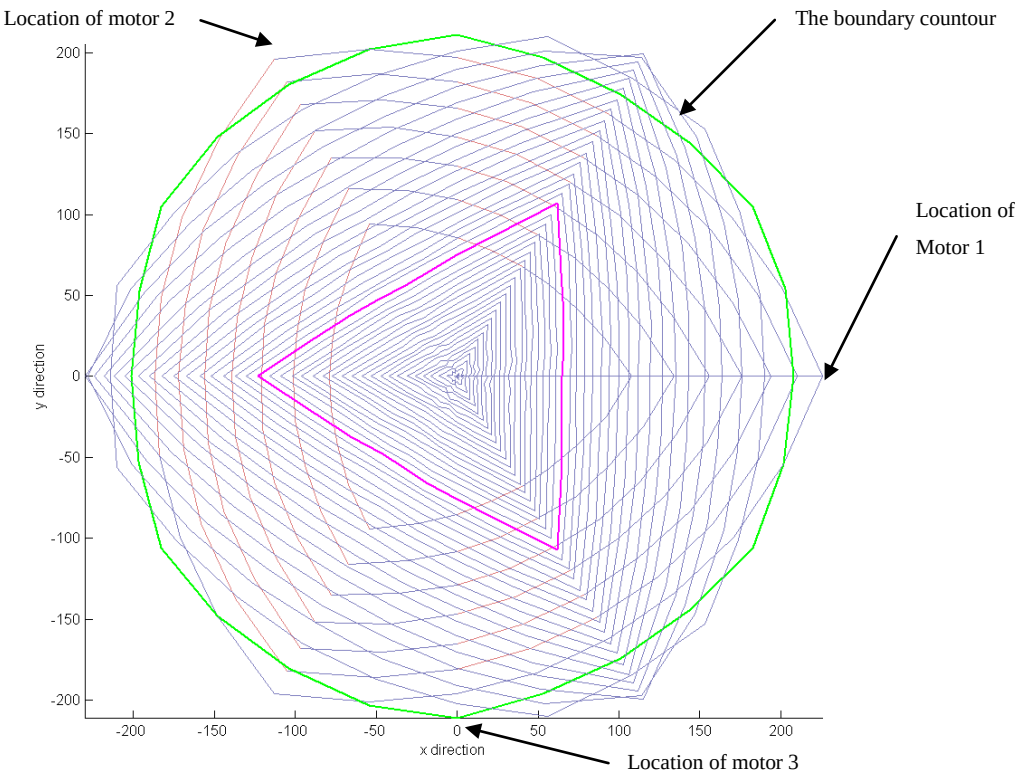


Figure 4.10. The foothold regions of the Delta walking machine (top view).

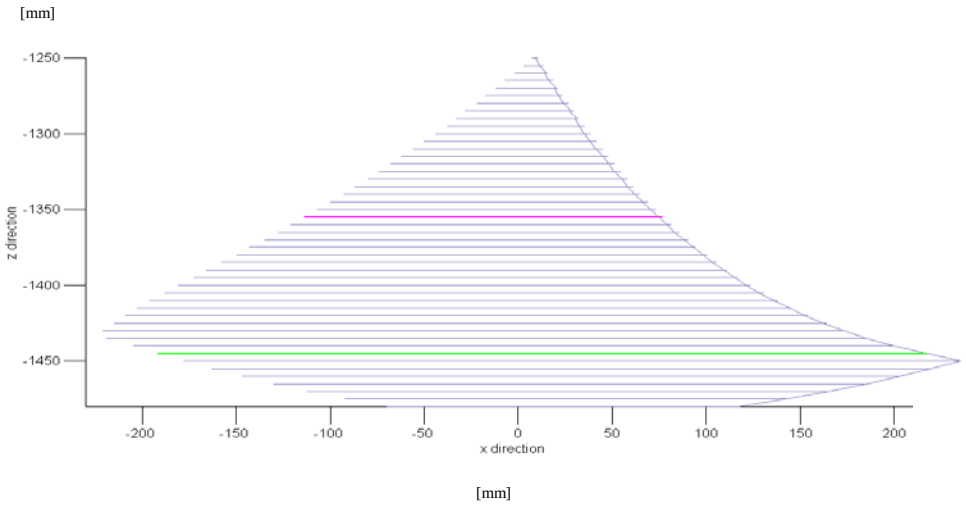


Figure 4.11. The foothold regions of the Delta walking machine in front view.

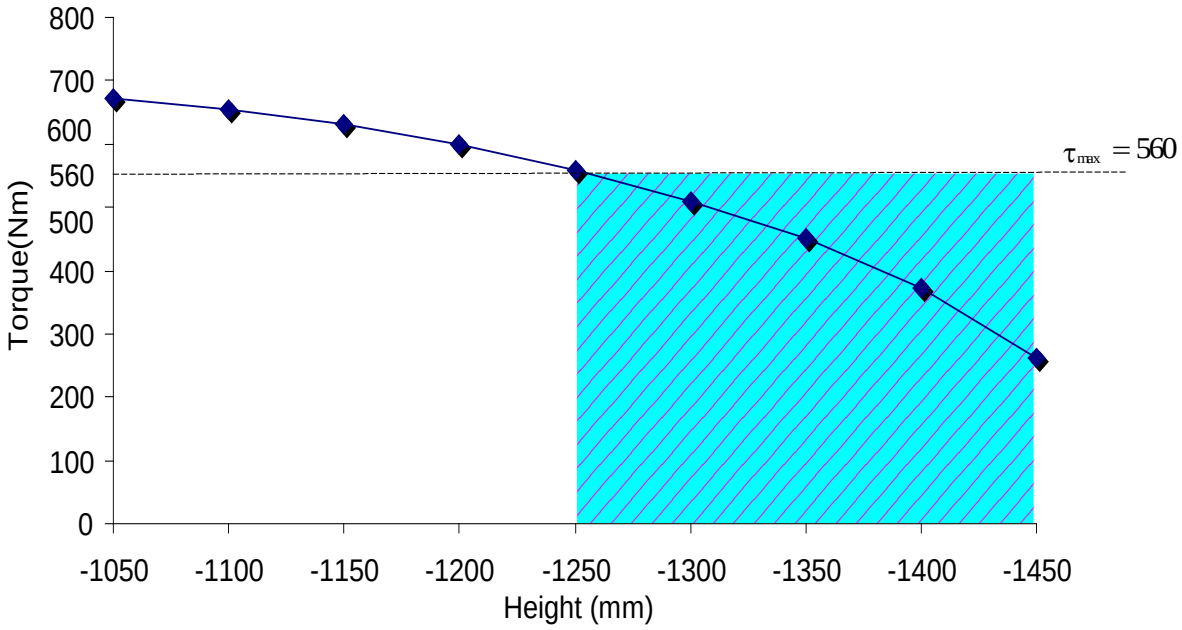


Figure 4.12. The required torque of supporting the weight of base platform.

Figure 4.12 shows that the required torque for each control axis to maintain the static equilibrium of the Delta walking machine is directly related to the distance between the two platforms. As the distance between the two platforms increases, less torque are required to attain the static equilibrium (and vice verse as the distance decreases). The results indicate that to support the weight of the upper part of the Delta walking machine without overloading the harmonic gear drive, the distance between the two platforms about the z-axis should be at least –1250mm apart. The operating range for the walking sequence of phase (2)[1] (e.g. the supporting legs are lifted off the floor by moving the base platform vertically upward to a position along the z axis) therefore is restricted to the hatched area.

[1] Refer to chapter 1, section 1.3 for detail.

4.8 Step Analysis

The mobility of the Delta walking machine is constrained by the step length that is performed in the walking sequence of phase (3)[2], and by the maximum continuous torque capacity of the harmonic gear drive. The static force analysis shows that the Delta walking machine is capable of an omnidirectional step of 120mm without over stressing the harmonic gear drives. The maximum step length is only attainable provided the phase (3) mode is performed within a cylindrical envelope of 120mm from –1395mm to –1460 mm below the base platform as shown in figure 4.13 (a) and (b).

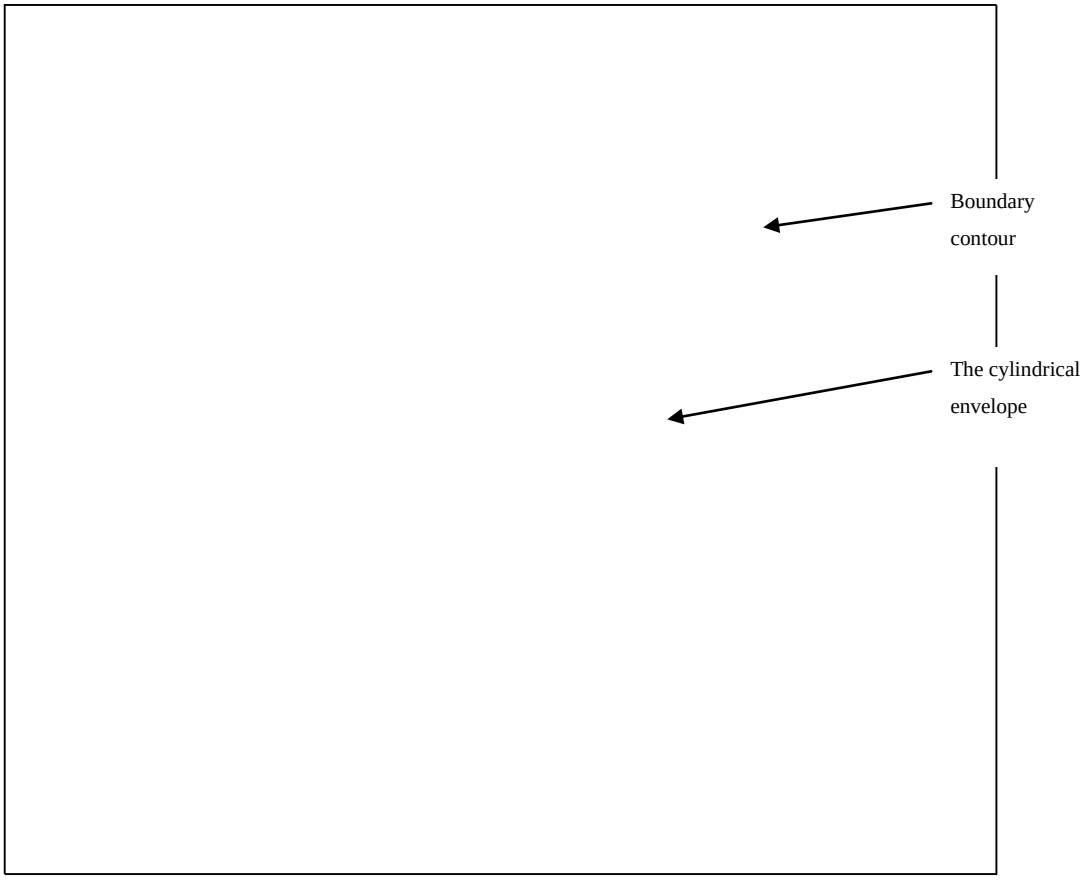


Figure 4.13(a). The omnidirectional step walking within the cylindrical envelope in top view.

[2] Refer to chapter 1, section 1.3 for detail.

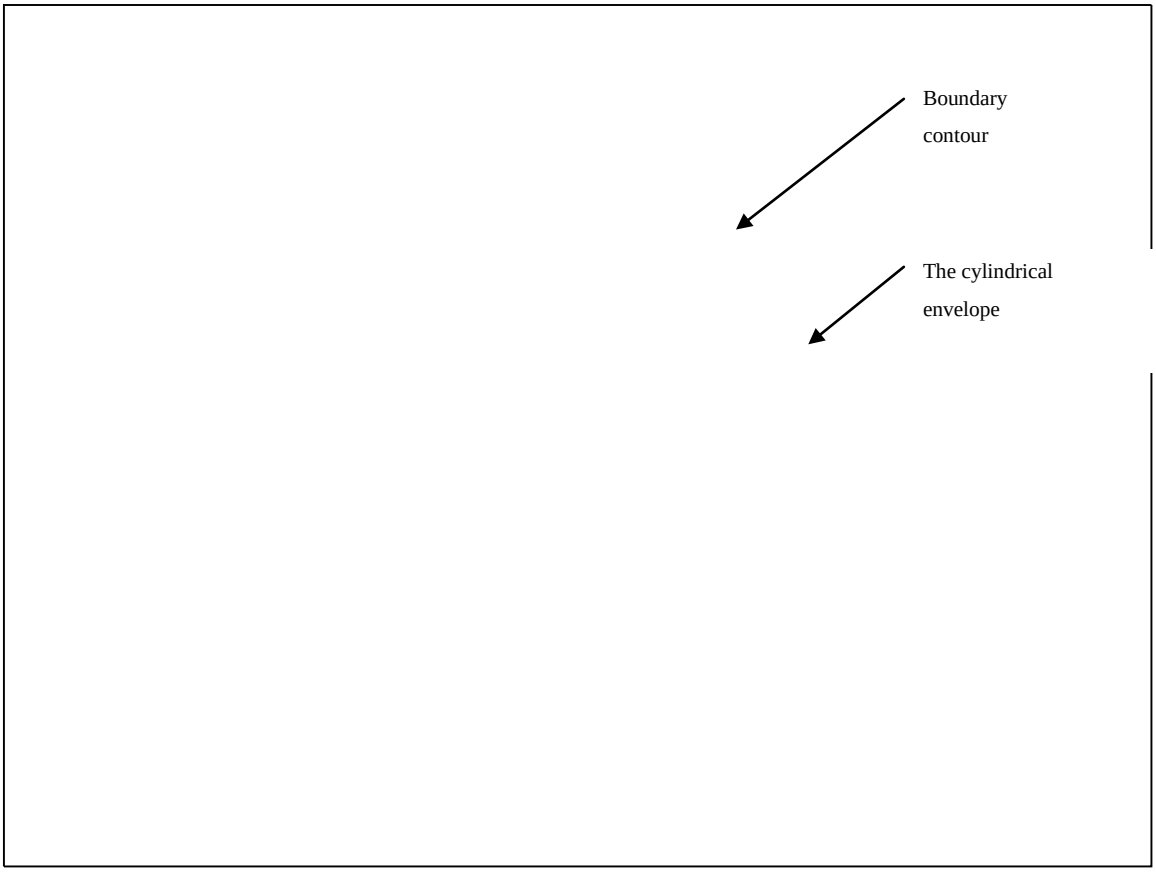


Figure 4.13(b). The omnidirectional step walking within the cylindrical envelope in isometric view.

Since the maximum step length has been verified, the motions for each walking sequence can be predefined. This ensure that the step cycles are executed in zones where the actuators can provide enough torque to maintain the walking machine in a static equilibrium state and every step operation is well away from the singularities. The Cartesian positions determined are shown in figure 4.14.

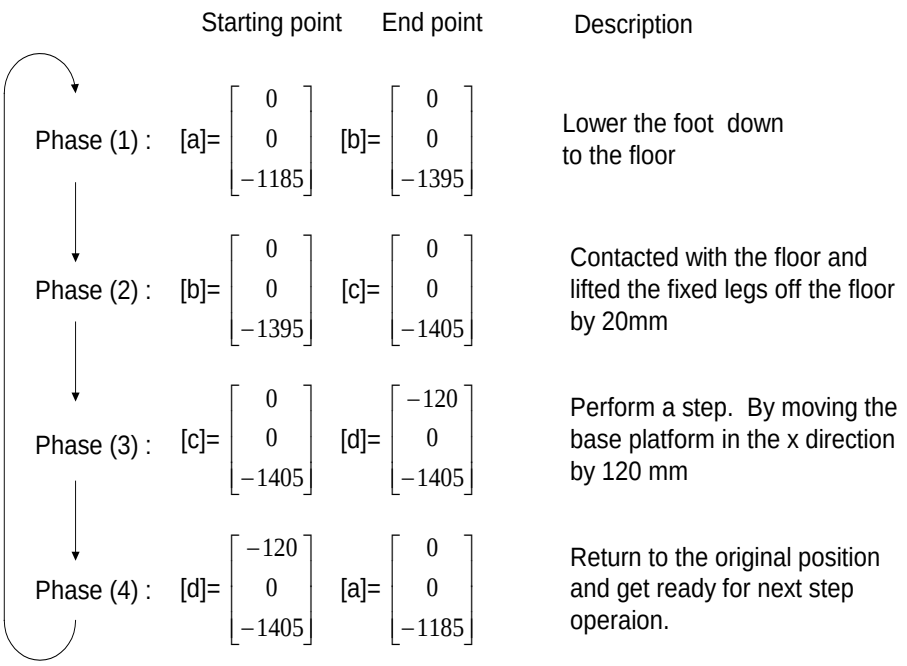


Figure 4.14(a). The determined Cartesian position for the walking sequence

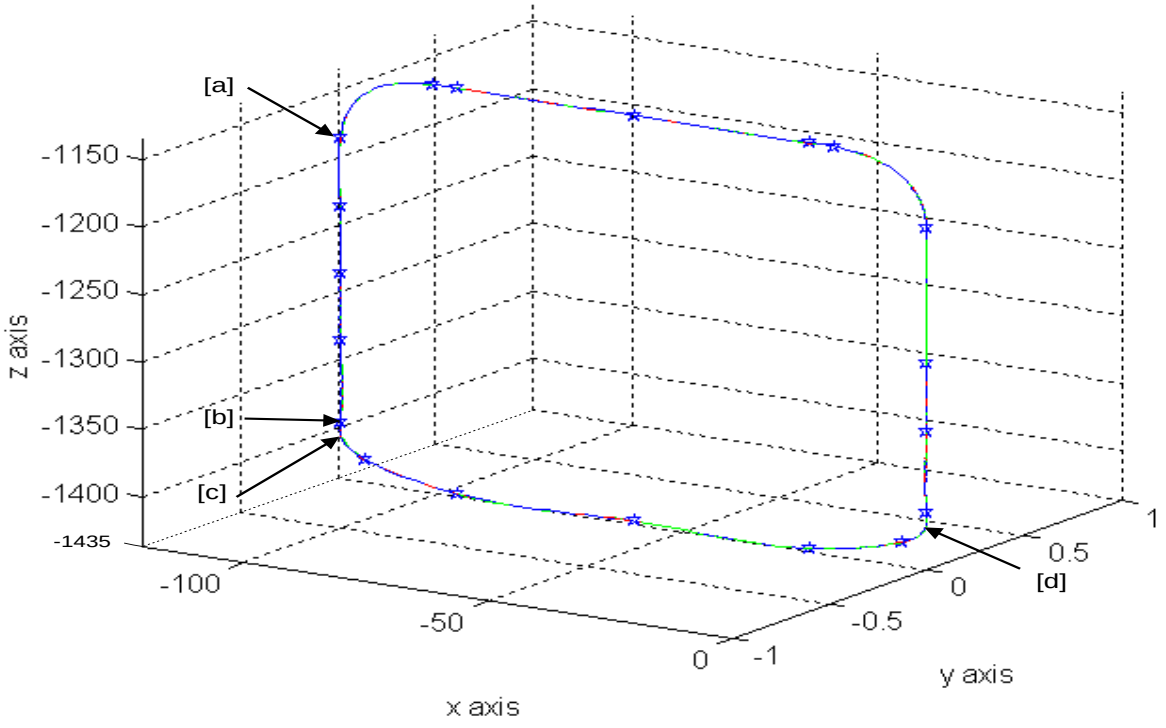


Figure 4.14(b). The determined Cartesian position for the walking sequence

4.9 Foot Design

The primary function of the end effector (foot) is to support the weight of the walking machine which is about 250kg, and to prevent the machine from tipping over while performing the phase (3) mode operation. Hence, the structure of the end-effector has to be strong and rigid for this application. The first issue that has to be considered is the physical dimension of the foot, followed then by the material selection for the foot. The height of the foot can seriously affect the walking gait of the walking machine if an inappropriate height is used. In fact, the perpendicular distance between the base reference on the base platform and the floor is about 1555mm, so to keep phase (2) and (3) operations within the cylinder envelope, a foot with height of 160mm is required. Also, the polygonal area of the foot determines the robot's static balance. The polygonal area should always be greater than the cross section area of the cylinder envelope. To satisfy this requirement, a tripod foot with length of 500mm from its orthocenter was developed. It consists of three Rectangular Aluminum alloy tubes (6063T) and three anti-vibration machine mounts as shown in figure 4.15. Aluminum alloy extrusions were used for their lightweight. The purpose of the machine mounts is to prevent the foot from slipping and to absorb the noise or the shock loading during the "touch down". In addition, another important aspect that has to be considered by the designer is the level of the tripod foot. It has to be parallel with the travelling platform so that the foot will touch down and all the legs will lift off the ground simultaneously. If the foot is not planted evenly, unsteady lifting may result.

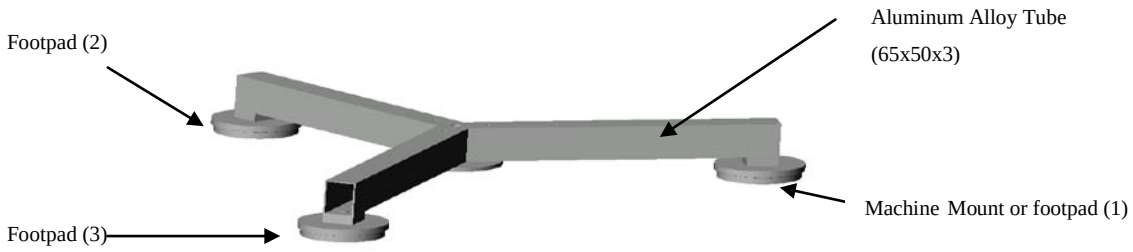


Figure 4.15. The proposed tripod foot.

Chapter 5 Trajectory modelling

5.0 Overview

In this chapter, an off-line optimal continuous path planning method is presented and an algorithm is developed in Matlab for real time joint trajectory control at the joint level. The step walking path is approximated by a set of location nodes selected on the desired path. The motion control of the walking machine is provided by a spline trajectory interpolation at the joint level.

5.1 Introduction

The Delta walking machine is programmed to perform onmidirectional walking in a structured environment, such as in the laboratory, without any external guide track. Off-line optimal trajectory planning has been implemented to trace the motion of the tripod foot (end effector) so that it is able to repetitively follow a planned step path while the Delta walking machine moves toward a particular destination.

5.2 Manipulator Path Control Method

Generally, there are two alternative approaches to achieve the motion on the closed chain kinematics manipulator. One is direct control at the foot level or end-effector level. The other is control at the joint level. The first approach requires evaluating the inverse kinematics at each sampling period in order to obtain joint incremental displacement commands from the end-effector incremental displacement commands. This results in high computational loads. The alternative joint control requires that a number of nodes, known as waypoints or knots, be determined along the desired cartesian path. These knots are mapped into joint space variables via the inverse kinematics transformation. Once the joint coordinations corresponding to the path

waypoint are obtained, an analytic interpolation curve (e.g. 4-3-4 trajectory) is fitted through those coordinations. The position and velocity commands for a joint actuator can be directly evaluated from the interpolated joint trajectories. The time scale can be distorted on all 3 axes if the acceleration or velocity demands exceed the capabilities of any axis drive.

The joint variable control method has been chosen for the foot trajectory generation of the Delta walking machine for the following reasons: First, the trajectory can be planned directly in terms of the controlled variable (such as joint position, velocity and acceleration) which results in easy trajectory planning and fast mathematical calculation. Secondly the computational load is dramatically reduced because inverse kinematics are only applied on those selected waypoints. However, the disadvantage of using this trajectory profile is that the robot dynamics have been ignored so the resultant trajectories do not take full advantage of the robot‘s capabilities [3].

5.3 Spline Method for Path Planning

The most popular type of technique for smoothing a joint trajectory is using polynomial splines. Thus, the 4-3-4-trajectory generator, widely used for interpolation, is chosen for determining the walking path of the Delta walking machine. A fourth order polynomial is introduced to the beginning and end segments in order to attain a zero joint velocity and zero joint acceleration for lifting off at the starting point and setting down at the end point. In the middle segments, a 3^{rd} order algebraic spline consisting of piecewise-continuous 3^{rd} order polynomials is used because of its simplicity and continuity on joint accelerations. Also the cubic polynomial trajectories are smooth and have a small angular motor displacement overshoot of no more that 10 degrees. As a result, the spline generator maintains the continuity of the joint positions, velocities and accelerations, and limits the jerks to an acceptable level at the minimum travelling time. The drawback of this method is that an update on the nodes requires the entire trajectory be recomputed.

To have a prescribed path with a large number of waypoints, the spline fit for the trajectory requires that all spline coefficients be computed before the trajectory is

executed. Because of the complexity of the spline calculations, the off-line trajectory planning scheme is more practical. Nevertheless, the off-line method allows the waypoints to be specified more closely in time, thus increasing the accuracy of the trajectory and allowing a shorter sampling period to be used. All the step data for the walking application can be precalculated and stored in the ram of the PC, and then sent to the controller (DSP) via serial communication at run time.

5.4 Inverse Kinematics and Direct Kinematics Solution

Solutions of inverse kinematics and direction kinematics of the Delta robot have been presented by several researchers [37-43]. Here, the inverse kinematics and direct kinematics solution are derived following the method used by Lintott and Dunlop [56].

5.5 Trajectory modelling on each axis

Since the joint control mode is operated in open loop mode, there is no information being fedback or being passed to other axes. Thus trajectory modelling is crucial for the whole motion control system. During the step movement, the movement of the tripod foot (attached under the travelling platform) on the Delta walking machine depends purely on the coordinated movement of the 3 actuators on the base platform. Therefore, to command the tripod foot to move along a desired path, a joint space trajectory profile is required for governing the motor motion on each axis.

Although the 3-actuator arms and linkages form a closed loop geometry, the joint trajectory for each axis is interpolated independently. Thus, the motion on each axis is governed by its individual trajectory profile, and the synchronization of the three axes is ensured by modelling the three independent trajectories with respect to an identical travelling time history. This control scheme provides a simple algorithm and allows each axis to be driven with its own set of parameters such as speed, acceleration and jerk. In the trajectory modelling, the three individual joint space trajectories are used to determine the required number of increments between each sample time for each control

axis. Software has been written in Matlab to support the modelling system (trajectory generator) for the foot and its procedure is illustrated in figure 5.1.

Moreover, the velocity, acceleration and jerk limits are imposed on the joint trajectory to ensure that the stepping motors, DSP and its interface circuit board are all operating within their limits. Therefore, pulse losses on the interface circuit board and stalling of the stepping motors can be avoided during operation. Three assumptions are inherent in the Matlab modelling:

1. The backlash on the harmonic gear drive is neglected and assumed to be zero.
2. All the joints are free from deformation.
3. Links are rigid.

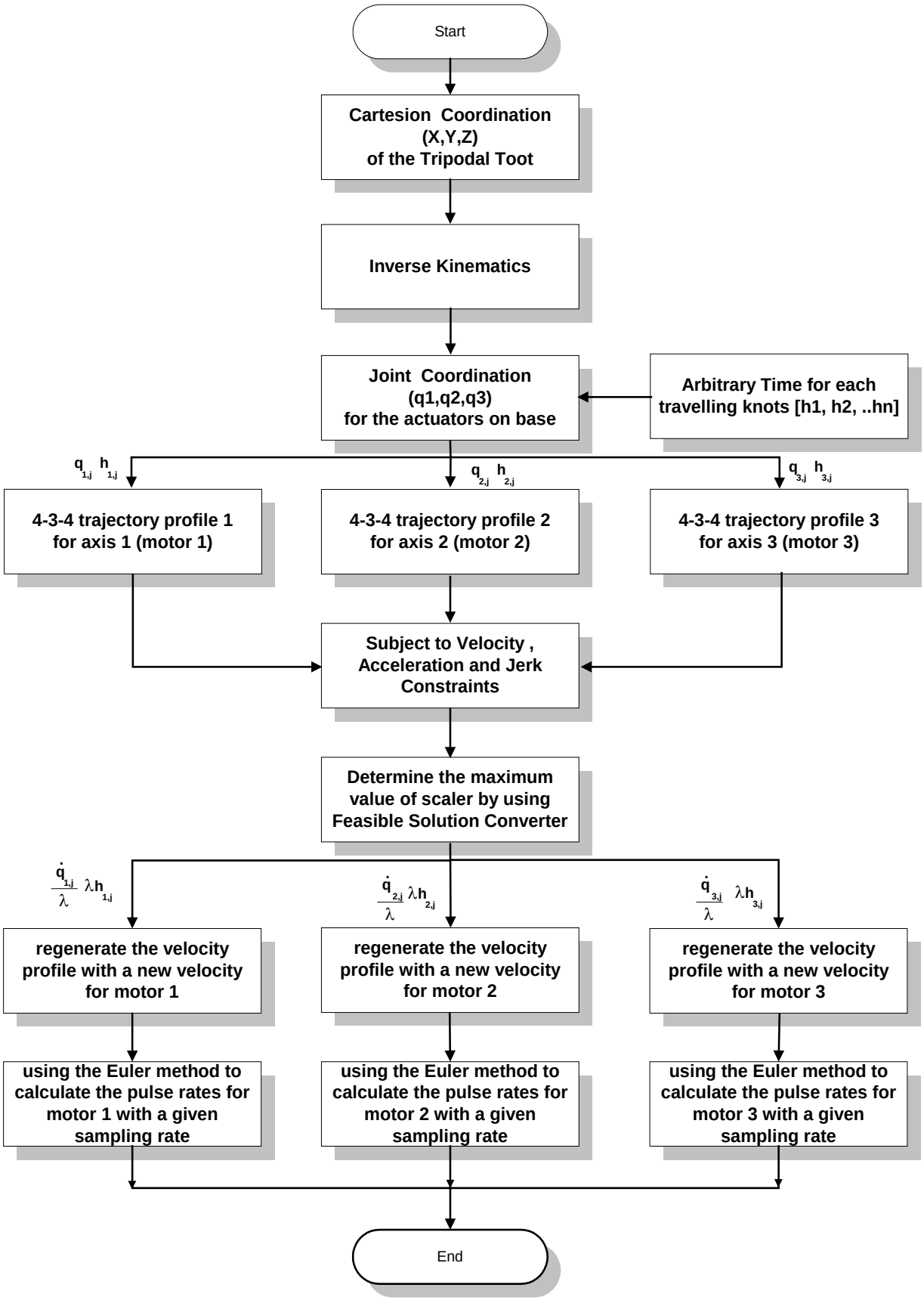


Figure 5.1. Trajectory generator with three separated trajectory profiles.

5.6 4-3-4 trajectory generation for each joint

The 4-3-4 trajectory profile is used to generate the joint trajectory for each stepping motor in order to ensure that each motor starts smoothly at the beginning and also ends with a smooth stop. In the trajectory generation, a separate polynomial is defined for each segment of the path. The polynomial function (f_k) of each segment must be continuous in displacement, velocity and acceleration. These conditions can be met by equating the end of the segment with the beginning of the next segment i.e. $f_k(t_k) = f_{k+1}(0)$, $f'_k(t_n) = f'_{k+1}(0)$ and $f''_k(t_n) = f''_{k+1}(0)$. Thus to develop a trajectory through n joint coordinate requires n-1 polynomials: a fourth-order polynomial, n-3 third-order polynomials, and a final fourth-order polynomial.

Let the task be assigned by a sequence of n cartesian poses $\mathbf{P}_1$, $\mathbf{P}_2$, ….., $\mathbf{P}_n$ to be assumed by the tripod foot at specified time instant T_1, T_2, ……, T_n. By using the inverse kinematics transformation, these data are transformed into joint configurations $q_{g,1}$, $q_{g,2}$, ……,$q_{g,n}$ where $g \in [1\ 2\ 3]$ which represents motor 1,2 and 3 respectively, and n is an integer which represents number of joint coordinates corresponding to the n waypoints at times T_1, T_2, ……, T_n.

In the following section, motor 1 is used as an example to demonstrate the development of a joint trajectory through n points with n-1 polynomials. The joint coordinates, q_1, q_2, ……,q_n, as shown in figure 5.2, represents the joint coordinates that the stepping motor 1 needs to trace at times T_1, T_2,……T_n. The time interval for travelling from one coordinate to another are represented as t_1, t_2, ……,t_{n-1} i.e. $t_k = T_{k+1}-T_k$, k=1,2,…….,n-1. The velocity and acceleration at the beginning and at the end of the trajectory are set to zero (i.e. $q'_1 = q'_n = q''_1 = q''_n = 0$).

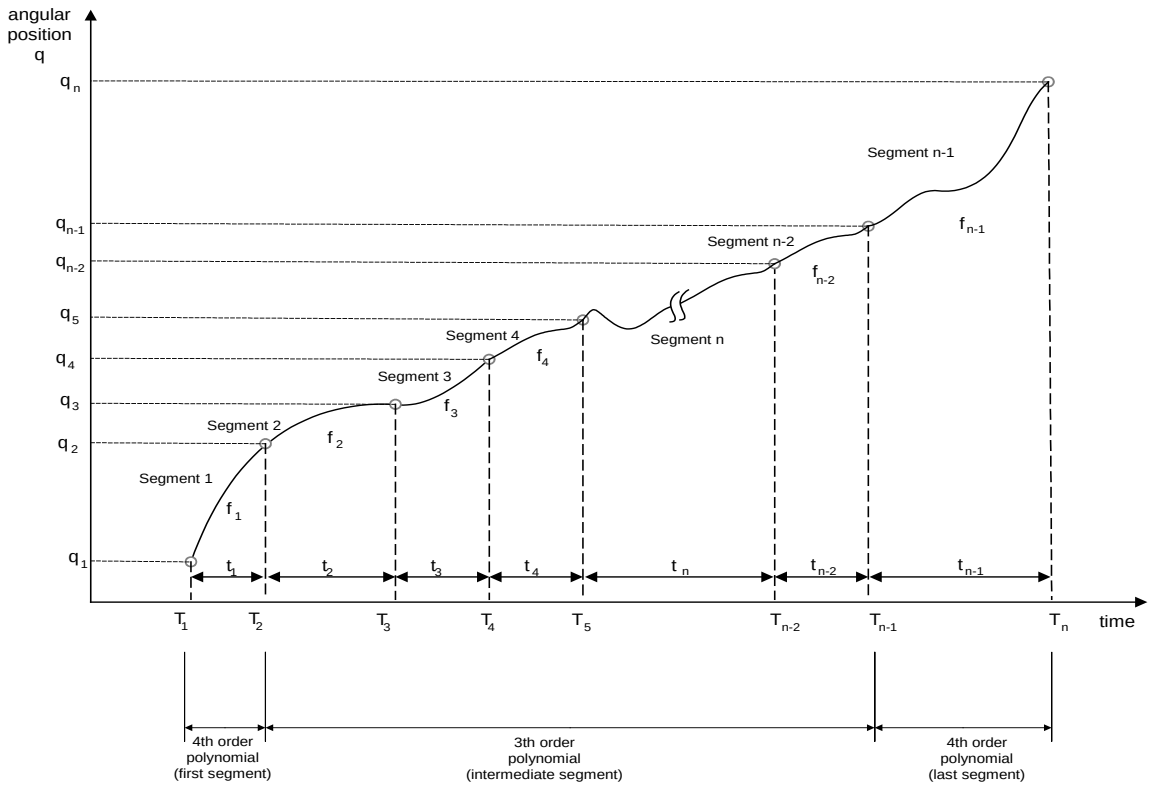


Figure 5.2. The joint coordinates for motor 1 to attain at the specified times.

5.6.1 At the First segment (q_1 to q_2)

For the first trajectory segment (between the initial position and lift off), the governing equation is a fourth order polynomial of the form:

$$f_1(t) = a_1(t)^4 + b_1(t)^3 + c_1(t)^2 + d_1(t) + e_1 \qquad (5.1)$$

Where $t \in [0, t_1]$ and the boundary conditions :

1) $f_1(0) = e_1 = q_1$ (5.2)

2) $f'_1(0) = d_1 = q'_1 = 0$ (5.3)

3) $f''_1(0) = 2c_1 = q''_1 = 0$ (5.4)

4) $f_1(t_1) = q_1 + b_1(t_1)^3 + a_1(t_1)^4 = f_2(0) = q_2$ (5.5)

5) $f'_1(t_1) = 3b_1(t_1)^2 + 4a_1(t_1)^3 = f'_2(0) = q'_2$ (5.6)

6) $f''_1(t_1) = 6b_1(t_1) + 12a_1(t_1)^2 = f''_2(0) = q''_2$ (5.7)

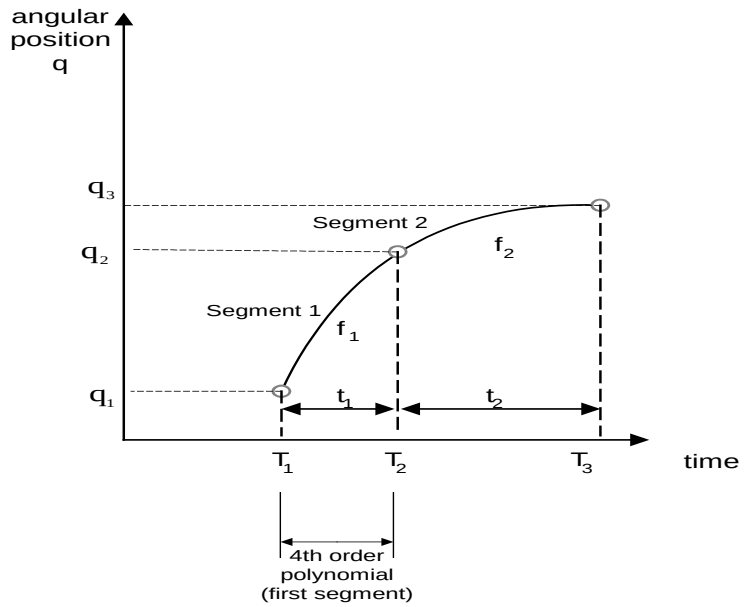


Figure 5.3. The joint coordinates for segment1.

By solving Equations (5.5) and (5.6) for b_1 and a_1, yields:

$$a_1 = \frac{3}{t_1^4}(q_1 - q_2) + \frac{1}{t_1^3}q'_2 \tag{5.8}$$

$$b_1 = \frac{4}{t_1^3}(q_2 - q_1) - \frac{1}{t_1^2}q'_2 \tag{5.9}$$

The fourth order polynomial becomes:

$$f_1(t) = [\frac{3}{t_1^4}(q_1 - q_2) + \frac{1}{t_1^3}q'_2](t)^4 + [\frac{4}{t_1^3}(q_2 - q_1) - \frac{1}{t_1^2}q'_2](t)^3 + q_1 \tag{5.10}$$

5.6.2 At Intermediate segment (q_2 to q_{n-1})

The equation for the spline segment between two intermediate points q_k and q_{k+1} ($2 < k < n-2$) of n-point trajectory consisting of n-1 spline segment can be written of the form:

$$f_k(t) = a_k(t)^3 + b_k(t)^2 + c_k(t) + d_k \tag{5.11}$$

Where $f_k(t)$ represents the position of the joint as a function of time and the limit time interval corresponding to each intermediate segment running from zero to t_k or $t = [0, t_k]$.

The boundaries conditions :

$$1)\ f_k(0) = d_k = q_k \tag{5.12}$$

$$2)\ f'_k(0) = c_k = q'_k \tag{5.13}$$

$$3)\ f''_k(0) = 2b_k = q''_k \tag{5.14}$$

$$4)\ f_k(t_k) = d_k + c_k t_k + b_k (t_k)^2 + a_k (t_k)^3 = f_{k+1}(0) = q_{k+1} \tag{5.15}$$

$$5)\ f'_k(t_k) = c_k + 2b_k t_k + 3a_k (t_k)^2 = f'_{k+1}(0) = q'_{k+1} \tag{5.16}$$

$$6)\ f''_k(t_k) = 2b_k + 6a_k t_k = f''_{k+1}(0) = q''_{k+1} \tag{5.17}$$

Substituting equations (5.12) and (5.13) into equations (5.15) and (5.16), and solving equations (5.15) and (5.16) for a_k and b_k yields:

$$a_k = \frac{2}{t_k^3}(q_k - q_{k+1}) + \frac{1}{t_k^2} q'_{k+1} + \frac{1}{t_k^2} q'_k \tag{5.18}$$

$$b_k = \frac{3}{t_k^2}(q_{k+1} - q_k) - \frac{1}{t_k} q'_{k+1} - \frac{2}{t_k} q'_k \tag{5.19}$$

From equations (5.12), (5.13), (5.18) and (5.19) the coefficient $\{a_k, b_k, c_k, d_k\}$ of the two intermediate points (refer as knots) can be solved with the boundary conditions consisting of position and velocity at each segment endpoint. Thus

$$\begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix} = \begin{bmatrix} \frac{2}{t_k^3} & \frac{-2}{t_k^3} & \frac{1}{t_k^2} & \frac{1}{t_k^2} \\ \frac{-3}{t_k^2} & \frac{3}{t_k^2} & \frac{-2}{t_k} & \frac{-1}{t_k} \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} q_k \\ q_{k+1} \\ q'_k \\ q'_{k+1} \end{bmatrix} \tag{5.20}$$

where $2 \le k \le n-3$, and t_k is the time interval in the k^{th} segment of cubic spline.

Equation (5.20) show that the coefficients can be determined only when the values of q'_{k+1}, q'_k and t_k are given. Those unknown parameter in each spline segment can be determined by imposing continuity of acceleration at the knot, equation (5.17). For example, given three waypoints, q_2, q_3 and q_4 connecting them with time intervals $[0,t_2]$ and $[0,t_3]$ as shown in figure 5.4.

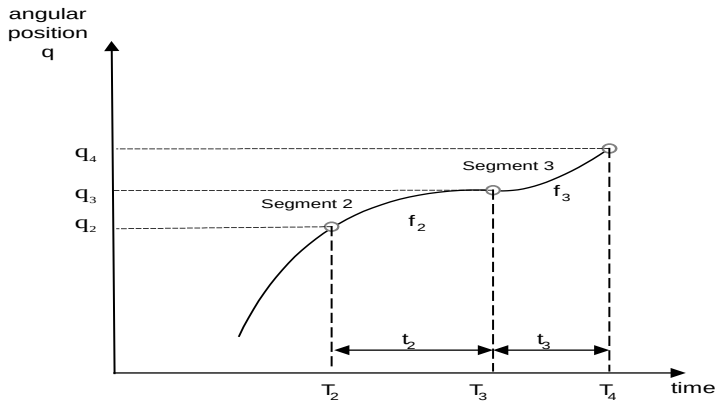


Figure 5.4. The joint coordinates for intermediate segment.

The acceleration at the end of the segment 2 can be determined as :

$$f''_2(t_2) = 6a_2 t_2 + 2b_2$$
$$= \frac{6}{t_2}\left[\frac{2(q_2 - q_3)}{t_2} + q'_2 + q'_3\right] + \frac{2}{t_2}\left[\frac{3(q_3 - q_2)}{t_2} - 2q'_2 - q'_3\right] \quad (5.21)$$

also, the acceleration at the beginning of the segment $f_3(0)$ evaluated as :

$$f''_3(0) = 2b_3$$
$$= \frac{2}{t_3}\left[\frac{3(q_4 - q_3)}{t_3} - 2q'_3 - q'_4\right] \quad (5.22)$$

Enforcing continuity of acceleration between the neighbouring segments leads to:

$$(t_3)q'_2 + 2(t_2 + t_3)q'_3 + (t_2)q'_4 = \frac{3}{t_2 t_3}\left[t_2^2(q_4 - q_3) + t_3^2(q_3 - q_2)\right] \quad (5.23)$$

Similarly, equation (5.23) can be applied to the intermediate segment 2 through n-2.

$$(t_{k+1})q'_k + 2(t_k + t_{k+1})q'_{k+1} + (t_k)q'_{k+2} = \frac{3}{t_{k+1}t_k}\left[t_k^2(q_{k+2} - q_{k+1}) + t_{k+1}^2(q_{k+1} - q_k)\right] \quad (5.24)$$

Therefore, the entire sequence can also be presented in matrix form as follows :

$$\begin{bmatrix} t_3 & 2(t_3+t_2) & t_2 & 0 & 0 \\ 0 & t_4 & 2(t_3+t_4) & t_3 & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & t_{k+1} & 2(t_{k+1}+t_k) & t_k \end{bmatrix} * \begin{bmatrix} q'_2 \\ q'_3 \\ . \\ . \\ . \\ q'_{k+2} \end{bmatrix} = \begin{bmatrix} \frac{3}{t_2 t_3}\left[t_2^2(q_4 - q_3) + t_3^2(q_3 - q_2)\right] \\ \frac{3}{t_4 t_3}\left[t_3^2(q_5 - q_4) + t_4^2(q_4 - q_3)\right] \\ . \\ . \\ . \\ \frac{3}{t_k t_{k+1}}\left[t_k^2(q_{k+2} - q_{k+1}) + t_{k+1}^2(q_{k+1} - q_k)\right] \end{bmatrix} \quad (5.25)$$

where $k \in [2, n-3]$

Equation (5.25) is rewritten as [m]*[q'] = [w], where [m] is an (n-4) x (n-2) matrix, [q'] is an (n-2) x 1 matrix, and [w] is an (n-4) x 1 matrix.

5.6.3 At the last segment

Lastly, for the last trajectory segment (between the final position and set down), the governing equation is also a fourth order polynomial of the form :

$$f_{n-1}(t) = a_{n-1}(t)^4 + b_{n-1}(t)^3 + c_{n-1}(t)^2 + d_{n-1}(t) + e_{n-1} \quad (5.26)$$

Where $t \in [0, t_{n-1}]$. The boundary conditions of equation (5.26):

1) $f_{n-1}(0) = e_{n-1} = q_{n-1}$ (5.27)

2) $f'_{n-1}(0) = d_{n-1} = q'_{n-1}$ (5.28)

3) $f''_{n-1}(0) = 2c_{n-1} = q''_{n-1}$ (5.29)

4) $f_{n-1}(t_{n-1}) = q_n = q_{n-1} + q'_{n-1}t_{n-1} + c_{n-1}(t_{n-1})^2 + b_{n-1}(t_{n-1})^3 + a_{n-1}(t_{n-1})^4$ (5.30)

5) $f'_{n-1}(t_{n-1}) = 0 = q'_{n-1} + 2c_{n-1}t_{n-1} + 3b_{n-1}(t_{n-1})^2 + 4a_{n-1}(t_{n-1})^3$ (5.31)

6) $f''_{n-1}(t_{n-1}) = 0 = 2c_{n-1} + 6b_{n-1}t_{n-1} + 12a_{n-1}(t_{n-1})^2$ (5.32)

Solving Equations (5.21), (5.22) and (5.23) for c_{n-1} , b_{n-1} and a_{n-1}, yields :

$$c_{n-1} = \frac{6}{t_{n-1}^2}(q_n - q_{n-1}) - \frac{3}{t_{n-1}} q'_{n-1} \tag{5.33}$$

$$b_{n-1} = \frac{8}{t_{n-1}^3}(q_{n-1} - q_n) + \frac{3}{t_{n-1}^2} q'_{n-1} \tag{5.34}$$

$$a_{n-1} = \frac{3}{t_{n-1}^4}(q_n - q_{n-1}) - \frac{1}{t_{n-1}^3} q'_{n-1} \tag{5.35}$$

To have continuity for the position, velocity and acceleration over the entire trajectory (the 1st segment, intermediate segments and last segment), equate the acceleration at the end of the each segment with the acceleration at the beginning of the next segment i.e. match coordinate value $f_{n-1}(t_n) = q_n$ and also the next two derivatives (i.e. $f'_{n-1}(t_n) = f'_n(0)$ and $f''_{n-1}(t_n) = f''_n(0)$).

The results of the equating of accelerations ($f''_{n-1}(t_n) = q''_n$) are shown as follows:

The acceleration at the end of the first segment ($f_1(t)$) and t = [0,t_1]):

$$f''_1(t_1) = \frac{6}{t_1^2}\left[4(q_2 - q_1) - t_1 q'_2\right] + \frac{12}{t_1^2}\left[3(q_1 - q_2) + t_1 q'_2\right] \tag{5.36}$$

Similarly, the acceleration at the beginning of the intermediate segment ($f_2(t)$) and t = [0,t_2]):

$$f''_2(0) = \frac{2}{t_2^2}\left[3(q_3 - q_2) - 2t_2 q'_2 - t_2 q'_3\right] \tag{5.37}$$

Equating equation (5.36) with (5.37) yields:

$$\left(\frac{2}{t_2} + \frac{3}{t_1}\right) q'_2 + \frac{1}{t_2} q'_3 = \frac{3}{t_2^2}(q_3 - q_2) + \frac{6}{t_1^2}(q_2 - q_1) \tag{5.38}$$

Therefore, with the same operation, the relationship between the last two segment can be derived from equation (5.38) as :

$$\left(\frac{2}{t_{n-2}} + \frac{3}{t_{n-1}}\right) q'_{n-1} + \frac{1}{t_{n-2}} q'_{n-2} = \frac{3}{t_{n-2}^2}(q_{n-1} - q_{n-2}) + \frac{6}{t_{n-1}^2}(q_n - q_{n-1}) \tag{5.39}$$

By combining the equation (5.38), (5.25) and (5.39), the whole trajectory can be expressed in a matrix form as follows :

$$\begin{bmatrix} \frac{2}{t_2}+\frac{3}{t_1} & \frac{1}{t_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_3 & 2(t_3+t_2) & t_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & & \bullet & & & . & & \\ . & & & & \bullet & & . & & \\ . & & & & & t_{n-2} & 2(t_{n-3}+t_{n-2}) & t_{n-3} & \\ 0 & & & & & & \frac{1}{t_{n-2}} & \frac{2}{t_{n-2}}+\frac{3}{t_{n-1}} & \end{bmatrix} * \begin{bmatrix} q'_2 \\ q'_3 \\ \bullet \\ \bullet \\ q'_{n-2} \\ q'_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{3}{t_2^2}(q_3-q_2)+\frac{6}{t_1^2}(q_2-q_1) \\ \frac{3}{t_3 t_2}\left[t_2^2(q_4-q_3)+t_3^2(q_3-q_2)\right] \\ \bullet \\ \bullet \\ \frac{3}{t_{n-2}t_{n-3}}\left[t_{n-3}^2(q_{n-1}-q_{n-2})+t_{n-2}^2(q_{n-2}-q_{n-3})\right] \\ \frac{3}{t_{n-2}^2}(q_{n-1}-q_{n-2})+\frac{6}{t_{n-1}^2}(q_n-q_{n-1}) \end{bmatrix}$$

(5.40)

The above equation can be simplified as [A]*[X] = [B], where [A] is a matrix of size (n-2)x(n-2), [X] is (n-2)x1 and [B] is (n-2)x1. The velocities of each segment ($[q'_n]$) can be evaluated by solving the equation i.e. $[X] = [A]^{-1}[B]$.

5.7 Time Sub-Optimal control

After segmentation, the maximum joint velocity, maximum joint acceleration and maximum jerk of each axis must be checked to see whether they exceed the constraints. The next goal is to find the maximum joint velocity, joint acceleration and jerk for each spline segment. The following section provides the formulations for determining the maximum joint velocity, joint acceleration and jerk values at each spline segment.

<u>For the First segment or last segment (by replacing 1 with n-1)</u>

Maximum Jerk:

The jerk equation for this segment is a linear function, it can be obtained from equation (5.1) as follows:

$$f'''_{1,max} = 24a_1 t + 6b_1 \tag{5.41}$$

where a_1, b_1 evaluated from Eqn (5.8)and (5.9), and $t = 0$ (or $t = t_1$)

The maximum jerk value exists either at the beginning segment ($t = 0$) or the end of segment ($t = t_1$) i.e. $|f'''_{1,\ max}| = 24a_1 t_1 + 6b_1$ (or $|f'''_{1,\ max}| = |6b_1|$) .

Maximum Acceleration:

First set equation (5.41) to zero and then calculate t_{max} where maximum acceleration occurs. Substituting the t_{max} into the following equation to find the maximum acceleration.

$$f''_{1,max} = 12a_1 t^2 + 6b_1 t + 2c_1 \tag{5.42}$$

We know a_1, b_1 c_1 and $t = t_{max}$

Maximum Velocity:

Set equation (5.42) to zero and then solve the quadratic function for $t \geq 0$ to get $t = t_{vel\_max}$ so that:

$$f'_{1,max} = 4a_1 t^3 + 3b_1 t^2 + 2c_1 t + d_1 \tag{5.43}$$

where a_1, b_1 are evaluated from the spline function, $c_1 = d_1 = 0$

Similarly, the maximum velocity, acceleration and jerk of the last segment can also be evaluated using the same approach.

For the intermediate segments (cubic polynomial)

Maximum Jerk:

Firstly, calculate the acceleration at the beginning and the end of each segment in the intermediate section by using the following equation.

$$f'''_{k\_max} = 6a_k \tag{5.44}$$

where $2 < k < n-3$, $n \in R$, a_k is determined from Eqn (5.20).

Maximum Acceleration:

Since the acceleration is a linear function, the maximum acceleration exists either at $t = 0$ or $t = t_k$. Substitute the time variables into the following equation to find the maximum acceleration.

$$f''_{k\_max}(t) = 6a_k t + 2b_k \tag{5.45}$$

Maximum Velocity:

The velocity constraints in an intermediate segment can be determined by setting equation (5.45) to zero first, then find $t_{k,,vel\_max}$ by setting equation (5.45) to zero.

There are three cases available by setting equation (5.45) to zero.

Case (1) $f''_k(t=0) > 0$ and $f''_k(t=t_k) < 0$

$t_{k\_vel\_max} = \dfrac{-b_k}{3a_k}$, substitute $t_{k\_vel\_max}$ into $f'_{k\_max}(t) = 3a_k t^2 + 2b_k t + c_k$

thus

$$f'_{k\_max} = c_k - \frac{b_k^2}{3a_k} \tag{5.46}$$

Where a_k, b_k and c_k are determined from Eqn (5.20).

Case (2) $f''_k(t=0) < 0$ and $f''_k(t=t_k) > 0$

Using the same principles and equations as case 1)

Case (3a) $f''_k(t=0) > 0$ and $f''_k(t=t_k) > 0$

(3b) $f''_k(t=0) < 0$ and $f''_k(t=t_k) < 0$

In either case

$$f'_{k\_max} = f'_k(t=0) \quad \text{or} \quad f'_{k\_max} = f'_k(t=t_k) \tag{5.47}$$

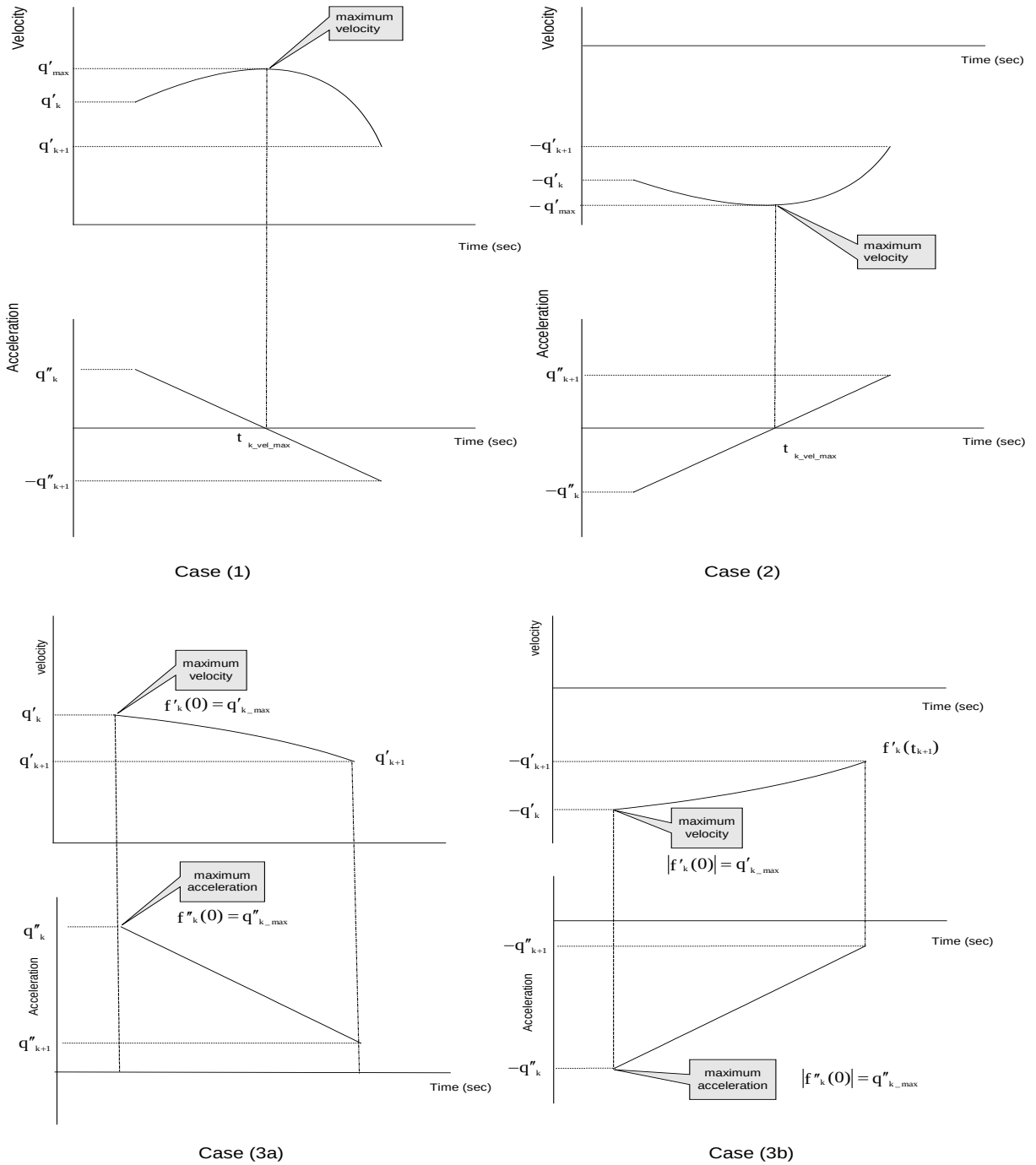


Figure 5.5. The maximum velocity and acceleration at a time interval.

5.8 Existence of A Feasible Solution

To obtain a smooth path that satisfies all the constraints, two methods can be used. One is a 'feasible solution converter (FSC)' proposed by Luh [11]. The other is a local time scaling technique. The joint angular velocities, accelerations and jerks values on each travelling knot are compared with the maximum allowable limits. If one of the joints exceeded the constraint values, the feasible solution converter simply expands the entire time intervals $\{t_1, t_2, \ldots\ldots, t_n\}$ to bring the unsatisfied velocities, accelerations and jerks down to the constrained values. The converter (FSC) operated as follows:

$$\lambda_1 = \max_{j=1..n-1}\left[\max\left|f'_{j,max}\right| / \text{Velocity\_Constraints}_j\right]$$

$$\lambda_2 = \max_{j=1..n-1}\left[\max\left|f''_{j,max}\right| / \text{Acceleration\_Constraints}_j\right]$$

$$\lambda_3 = \max_{j=1..n-1}\left[\max f'''_{j,max} / \text{Jerk\_Constraints}_j\right]$$

$$\lambda = \max\left(1, \lambda_1, \lambda_2^{1/2}, \lambda_3^{1/3}\right)$$

Step 1) Calculate λ_1, λ_2, and λ_3.

Step 2) Decide λ.

Step 3) Replace the time intervals $\{t_1, t_2, \ldots\ldots, t_n\}$ with $\{\lambda t_1, \lambda t_2, \ldots\ldots, \lambda t_n\}$.

Step 4) Replace the angular velocity $\{q'_1, q'_2 \ldots., q'_{n-1}\}$ with $\{q'_1/\lambda, q'_2/\lambda \ldots., q'_{n-1}/\lambda\}$ if $\lambda \neq 1$.

If the moving speed of the Delta walking machine is desired to be as high as possible, a locally time scaling method can be used. This method only expands the time history of the travelling knot that exceeded the constrained values but the coefficients of all the polynomials need to be recomputed when a travelling knot along the trajectory is modified. This can be very time consuming. Since the step operation of the Delta walking machine is performed with low speed, the use the FSC method is acceptable.

5.9 An illustrative example

A program has been written in Matlab to implement the method of the path generation presented in the proceeding section. An example is used to show the difference between the motion profile without the constraints and with the constraints. For simplicity, the Cartesian positions, as shown in table 5.1, were converted into joint coordination by using inverse kinematics and a sequence of arbitrary times was assigned for each of the knots as shown in table 5.2.

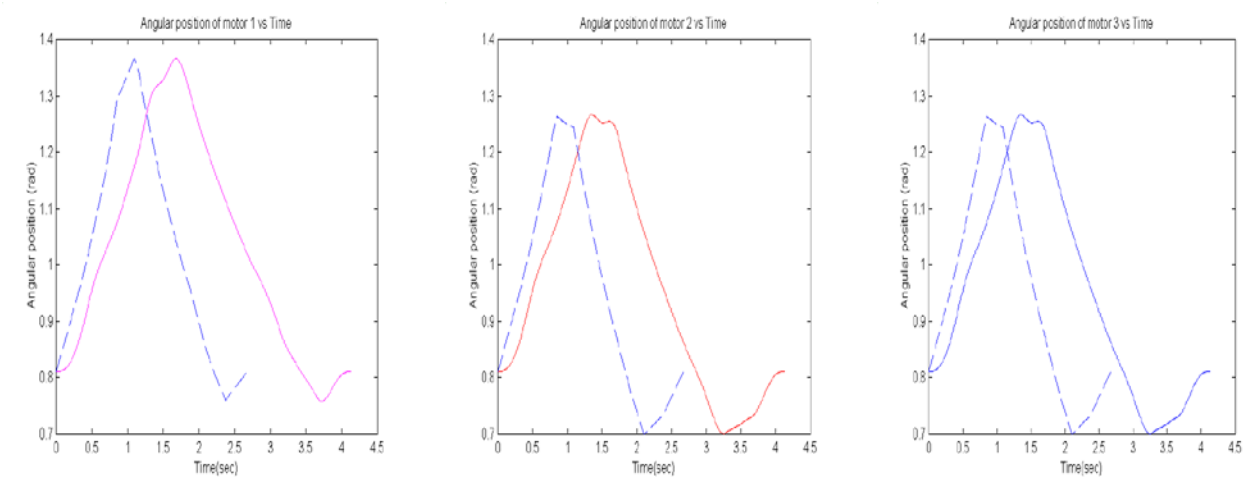
Number of knots	X	Y	Z
1	0	0	-1185
2	0	0	-1285
3	0	0	-1335
4	0	0	-1395
5	5	0	-1420
6	24	0	-1435
7	60	0	-1435
8	96	0	-1435
9	115	0	-1420
10	120	0	-1395
11	120	0	-1335
12	120	0	-1285
13	120	0	-1235
14	120	0	-1185
15	101	0	-1135
16	96	0	-1135
17	60	0	-1135
18	24	0	-1135
19	19	0	-1135
20	0	0	-1185

Table 5.1. The Cartesian coordination of the path.

Number of knots	Arbitrary time base (sec)	Angular position for axis 1(rad)	Angular position for axis 2 (rad)	Angular position for axis 3 (rad)
1	0.00	0.811	0.811	0.811
2	0.33	0.964	0.964	0.964
3	0.17	1.050	1.050	1.050
4	0.20	1.172	1.172	1.172
5	0.08	1.228	1.234	1.234
6	0.06	1.253	1.284	1.284
7	0.12	1.225	1.304	1.304
8	0.12	1.199	1.329	1.329
9	0.06	1.146	1.300	1.300
10	0.08	1.081	1.241	1.241
11	0.20	0.956	1.117	1.117
12	0.17	0.866	1.030	1.030
13	0.17	0.784	0.952	0.952
14	0.17	0.707	0.880	0.880
15	0.17	0.649	0.800	0.800
16	0.02	0.653	0.797	0.797
17	0.12	0.685	0.774	0.774
18	0.12	0.719	0.754	0.754
19	0.02	0.723	0.751	0.751
20	0.30	0.811	0.811	0.811

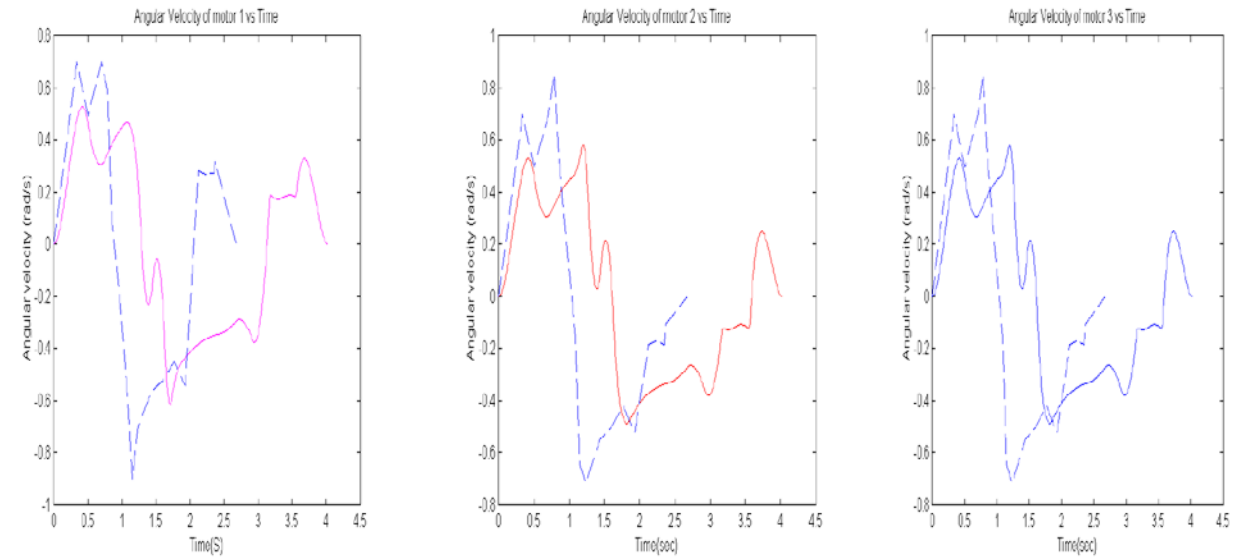
Table 5.2. The Joint coordination and time bases for each axis.

The admissible joint angular velocities, angular accelerations and jerks for each axis were set to 1 rad/s, 25 rad/s^2 and 200 rad/s^3 respectively. Figure 5.5 and 5.6 illustrate the joint coordination and angular velocities for three stepping motors respectively. Figure 5.6 clearly shows that the motion time for the case without constraints (dashed line) was 2.76sec whereas with constraints (solid line), the motion time was 4.02sec.



(a) (b) (c)

Figure 5.6. The joint coordination for stepping motor 1,2 and 3



(a) (b) (c)

Figure 5.7. The angular velocities for stepping motor 1,2 and 3

5.10 Pulse rate generation

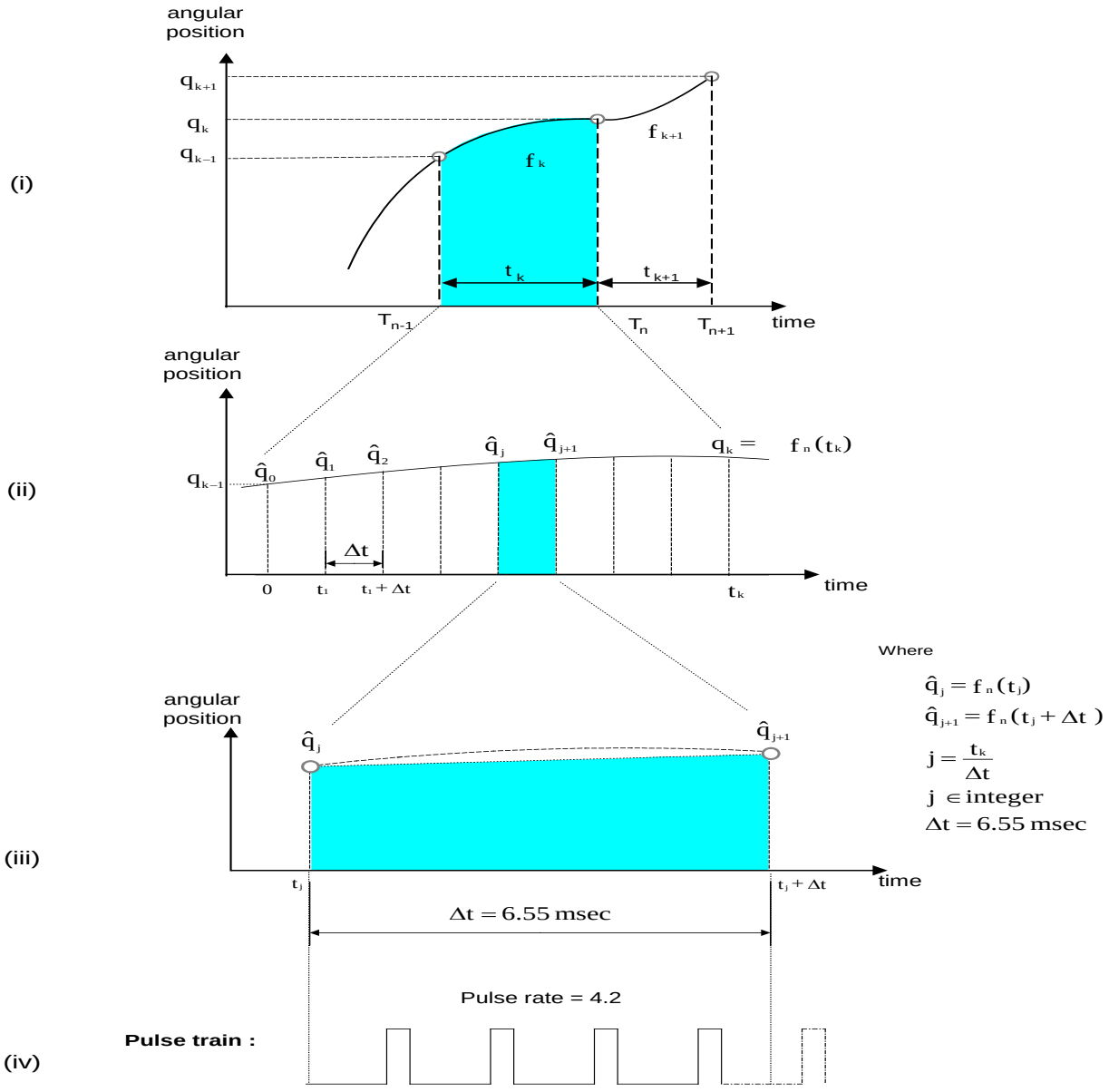


Figure 5.8. The procedure for generating the pulse rate.

Since the polynomial coefficients at each segment are determined and the polynomial function for each separate segment is a continuous function, the polynomial function (f_k) over the time interval [0, t_k] can be computed to obtain the number of step command pulses for a specified sampling period (e.g. 6.55ms). The time interval between the knots (q_k to q_{k+1}) is divided into j subintervals of width 6.55ms (c.f. figure 5.8 (ii)) and then the corresponding value ($\hat{q}_j$) of the angular positions on each subinterval are

evaluated as $f_k(\ j*\Delta t\ )$, where j is an integer and Δt is 6.55ms. Thus, the number of step command at each sampling time is evaluated as:

$$\text{Number of pulses} = f_k(t_j + \Delta t) - f_k(t_j) \tag{5.48}$$

Where t_j is $j*\Delta t$ and j is an integer, Δt is controller sampling time and

f_k is a polynomial function .

The evaluated pulse rates are floating point variables which are slow to send from the PC to the DSP via the RS232 serial communication port. In addition, the performance of the fixed point DSP is degraded if the floating point variables are used. Thus, the floating point pulse rate (number of steps at Δt) is rounded down to the nearest integer and the remainder of the truncated value (β) is added to the number of steps calculated for the next sampling time as illustrated in figure 5.9. This first stage of interpolation is run in Matlab.

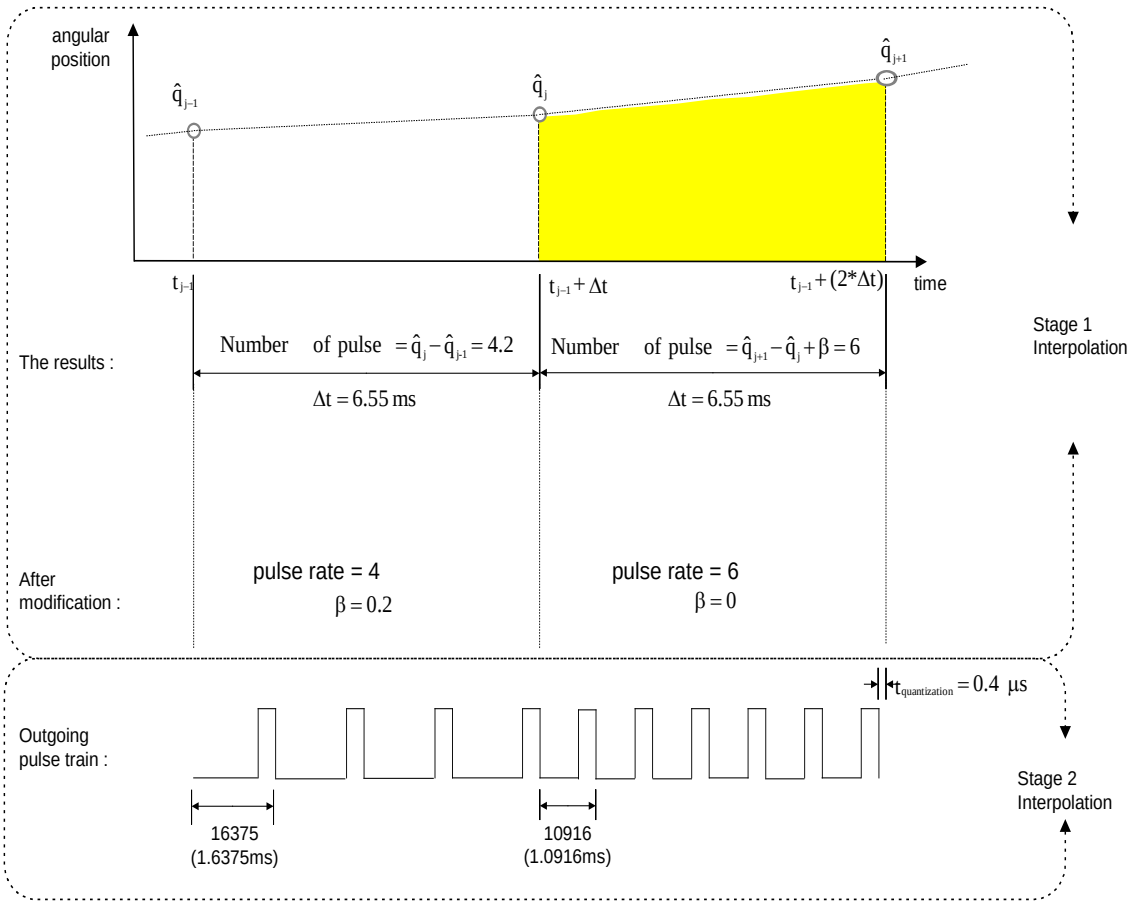


Figure 5.9. Numerical calculation of number of pulses over a sampling period

The pulse rate calculated by the first stage interpolation is transferred over to the DSP via the RS232 serial communication port, and then the 16 bits fixed point DSP performs a second stage of interpolation. Since the step commands over the sampling period (6.55msec) are spaced uniformly, the time interval for a step command can be approximated as

$$\text{Number of ticks for a step command} = \frac{65500}{\text{Number of step command}} \tag{5.49}$$

Where 65500 is a constant which corresponds to the number of ticks in a sampling period (the system clock is set to 10 MHz or 0.1μs per tick so a controller sampling time of 6.55 ms requires 65500 timer ticks).

However, when the remainder of the division in equation (5.49) is nonzero, a quantization error occurs. The number of ticks for a step command is always rounded down to the nearest integer value (e.g. $65500 \div 32 = 2046.8 \approx 2046$). This leads to an uneven spacing of the pulses and a small discontinuity in velocity at the end of each sampling period as shown in figure 5.10. The worse case at 32 steps in 6.55ms produces a delay of 2.8 μs (see Table 5.3) or about 1.4% of a step i.e. negligible.

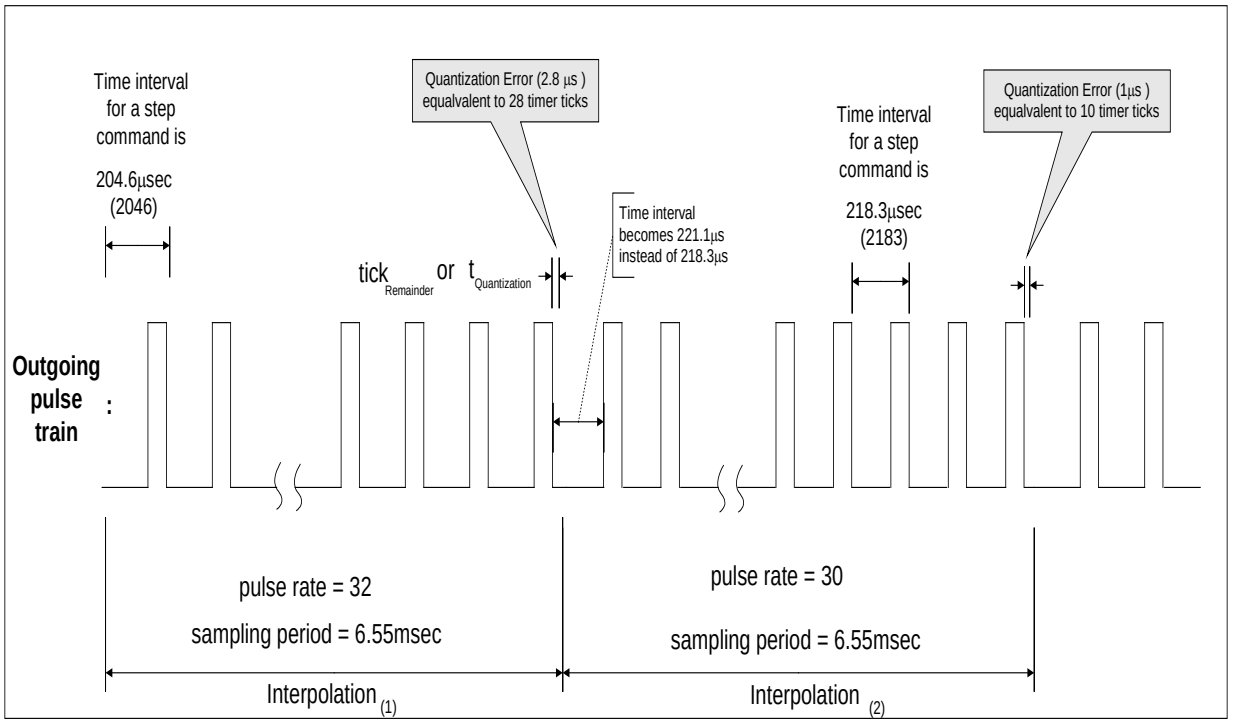


Figure 5.10. Quantization error over a sampling period

Pulse rates	No. of ticks for a step command(T1PER)	Round(T1PER)	$tick_{remainder}$	Quantization Error $t_{quantization}$ [sec]	Error in % ΔE
1	65500	65500	0	0	0
2	32750	32750	0	0	0
3	21833.33	21833	1	1.00E-07	0
4	16375	16375	0	0	0
5	13100	13100	0	0	0
6	10916.67	10916	4	4.00E-07	0
7	9357.14	9357	1	1.00E-07	0
8	8187.50	8187	4	4.00E-07	0
9	7277.78	7277	7	7.00E-07	0.10
10	6550	6550	0	0	0.00
11	5954.55	5954	6	6.00E-07	0.10
12	5458.33	5458	4	4.00E-07	0.07
13	5038.46	5038	6	6.00E-07	0.12
14	4678.57	4678	8	8.00E-07	0.17
15	4366.67	4366	10	1.00E-06	0.23
16	4093.75	4093	12	1.20E-06	0.29
17	3852.94	3852	16	1.60E-06	0.42
18	3638.89	3638	16	1.60E-06	0.44
19	3447.37	3447	7	7.00E-07	0.20
20	3275	3275	0	0	0
21	3119.05	3119	1	1.00E-07	0
22	2977.27	2977	6	6.00E-07	0.2
23	2847.83	2847	19	1.90E-06	0.7
24	2729.17	2729	4	4.00E-07	0.1
25	2620	2620	0	0	0
26	2519.23	2519	6	6.00E-07	0.2
27	2425.93	2425	25	2.50E-06	1.0
28	2339.29	2339	8	8.00E-07	0.3
29	2258.62	2258	18	1.80E-06	0.8
30	2183.33	2183	10	1.00E-06	0.5
31	2112.90	2112	28	2.80E-06	1.3
32	2046.88	2046	28	2.80E-06	1.4
33	1984.85	1984	28	2.80E-06	1.4
34	1926.47	1926	16	1.60E-06	0.8
35	1871.43	1871	15	1.50E-06	0.8
36	1819.44	1819	16	1.60E-06	0.9
37	1770.27	1770	10	1.00E-06	0.6
38	1723.68	1723	26	2.60E-06	1.3
39	1679.49	1679	19	1.90E-06	1.1
40	1637.50	1637	20	2.00E-06	1.2

The worse case condition

$$t_{quantization} = tick_{remainder} \times 0.1\mu sec \qquad \Delta E = \frac{t_{quantization}}{Round(T1PER) * 0.1\mu s} \times 100$$

Table 5.3. The quantization error for each pulse rate.

In order to achieve velocity continuity over the whole trajectory and to improve the interpolation time accuracy, the remaining time value ($t_{Quantization}$, or $tick_{remainder}$ {number of ticks}) due to quantization error is added to the sampling time interval of the next

interpolation (i.e. $\Delta t_2 = 6.55x10^{-3} + t_{Quantization}$). The time interval for a step command over the next sampling period becomes

$$\text{Number of ticks for a step command} = \frac{65500 + \text{tick}_{\text{remainder}}}{\text{Number of step command}} \qquad (5.50)$$

For this particular example (c.f. figure 5.10), the sampling time interval for the interpolation$_{(2)}$(i.e. pulse rate with 30 steps/sec) is adjusted to 65528 (65500+28 = 65528). The time interval for a step command over this sampling period is 2184 ($65528 \div 30 = 2184.267 \approx 2184$) and the quatization error for the interpolation$_{(2)}$ is 8 timer ticks ($65528 - [2184 * 30] = 8$). It can be seen that without the error correction, the quantization error in interpolation$_{(2)}$ is 10 timer ticks whereas with correction, the error is reduced to 8 timer ticks and the quantisation error of the interpolation$_{(1)}$ is eliminated. The quantization error correction presented may require a longer CPU interpolation time, hence there is a trade-off between the interpolation time and the relative accuracy between time precision.

Chapter 6 Hardware / software synergy of the motion control

6.0 Overview

This chapter examines on the basic features of the DSP (TMS320F240) that were used to support the three axes motion control. A detailed example is given to demonstrate how this single chip controller is configured to attain synchronisation of the three control axes.

6.1 Introduction

The motion of the 3 dof parallel mechanism is highly influenced by the coordination of the motion on the three control axes. Therefore, to perform a smooth motion along the prescribed walking path, the stepping motors on each axis ought to run interactively and simultaneously under all conditions. If one of the motors fails to synchronize with the others, an incorrect and rough walking path results. However, this undesirable motion can be minimised with a reliable and powerful DSP, provided the robot is working within the limited workspace and an adequate torque is supplied by each motor. It is clear that the software algorithms and DSP hardware architecture play important roles in the synchronised motion control of the three axes. Therefore, both hardware and software must work interactively in order to synchronize the three motors under the open-loop control system.

6.2 DSP Hardware Consideration

With a TMS320F240 EVM board as shown in figure 6.1, three on-chip peripherals can be employed to support the synchronised motion of the three stepping motors. These peripherals include:

1. The event manager module provides three independent general purpose timers and three compare registers for producing three independent outgoing pulses. As a result, each axis can be controlled by an individual clock to generate its own time base.
2. Digital I/O module controls the rotation direction and detects the limit switches or emergency switch signal.
3. Serial communication interface (SCI) module supports data reception and transmission between the PC and DSP.

Figure 6.2 illustrates the inter-relationship between each peripheral and the general layout of the control system.

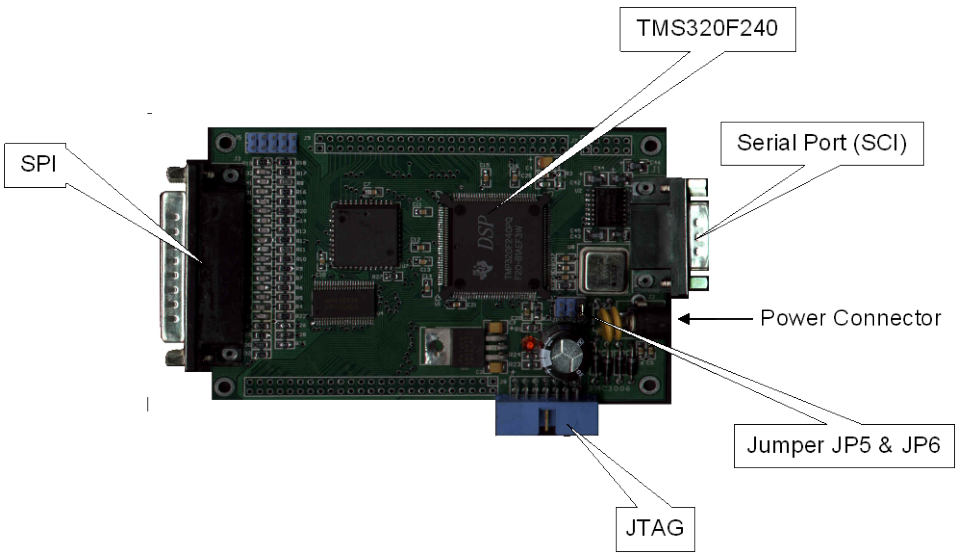


Figure 6.1 A TMS320F240 Evaluation Module from Softronics.

Error! No topic specified.

Figure 6.2. The basic modules required in DSP for synchronisation control.

6.3 Software Consideration

6.3.1 Software Control Description

The DSP receives the step rate (8 bit 2's complement: –128 to 127) and direction signal (positive step rate means clockwise rotation and vice verse) from the PC via a serial communication port (RS232), and saves the data into RAM for later use. When the ram buffer is half filled or half of the data has been transmitted successfully, the DSP interrupts are enabled to generate pulses and direction signals to control each stepping motor according to pulse rates that have been saved in the DSP ram.

In the DSP, the pulse generation and the direction outputs for a single step motor can be simply handled by a single periodic interrupt. Consequently, three independent period interrupts TPINT1, TPINT2 and TPINT3 are used to synthesize the step commands for motors 1, 2 and 3 respectively. For the first interrupt, TPINT1 corresponds to the Event Manager Interrupt Group A (EVIGA) that is a member of Core Interrupt INT2. When the TPINT1 is triggered, it sends a request signal to INT2 to make a demand for an interrupt service routine (ISR) execution. There is only one interrupt source within group A in use so the specified interrupt service routine (SISR) corresponding to the TPINT1 is performed immediately. However, for interrupts TPINT2 and TPINT3, both use EVIGB group B that is a member of Core Interrupt INT3. In this case, there are two interrupt sources within group B so the ISR of INT3 needs to read the content of the interrupt vector register B (IVRB) in order to determine the interrupt source that initiated the request, and hence to then perform the task required for the interrupt source. Occasionally, it is possible that both core interrupts are pending at the same time and since interrupt INT2 has priority over INT3, INT2 will be serviced first, followed by interrupt INT3. The interrupt hierarchy on CPU and its peripheral interrupts are detailed in appendix A.

Two interrupt groups and three time bases are used to achieve synchronised control of the 3 actuators on the Delta walking machine. A bug in the DSP microcode (TMS320F240 version 1.1) can be activated when two or more individual interrupts are enabled within an EVI Group (A, B or C). The bug is activated when that higher-priority interrupt within a group is triggered in the same cycle as the EVI vector register (EVIVRx) is read for a previously received lower-priority interrupt in the same group. This causes the higher-

priority pending flag in EVI Flag register (EVIFR) to be cleared in error. When this incident happens (during the interrupt operation), the higher priority interrupt is serviced using the vector address of the previously received lower-priority interrupt. The result is that the SISR of the higher-priority interrupt is not serviced and the SISR of the lower-priority interrupt is serviced twice (or more if more than two interrupts are used).

The bug is activated because of a one-cycle delay between a flag being set and the EVIVRx register being updated with a new vector address. This one cycle happens when a higher-priority event occurs in the same cycle as the vector read. This hardware problem cannot be avoided by software when three or more interrupts are enabled in one group as there is no way to detect which higher-priority interrupt flag bit gets cleared in error. Since TI provides a software workaround for two interrupts in a single EVI group, motors 2 and 3 are allocated to group B of INT3 for the step motor control. The software workaround procedure is:

1. Read the vector register and save.
2. Read the flag register and save.
3. Branch to the ISR corresponding to the vector saved in step 1. Branch table must check for lower-priority vector first.
4. In lower-priority ISR, check if the flag bit (saved in step 2) is set. (In normal operation, the flag bit corresponding to the vector read is cleared). If set, skip lower-priority ISR and branch to higher-priority ISR. If cleared, service lower priority as normal.
5. In higher-priority ISR, no checking of the flag is necessary. Either its vector was read correctly and the code branched here directly, –OR– its flag was cleared in error as a result of the bug, and the lower-priority ISR code caught the error and branched here.

6.3.2 An example of the System operation

An example is given in the following section to show how a period interrupt TPINT1 with GP timer 1 is used to achieve variable step rates with fix period motion control on stepping motor 1.

In the implementation, if the system clock of the CPU is operated at 10 MHz and the input clock prescaler is set to 1, then the ticks per second is set at 10^7. A Controller sampling time of 6.55 msec requires 65500 timer ticks. Thus, for producing 5 pulses per sampling time with pulse widths of 1 μsec, 5 interrupts need to be performed in 6.55 msec so the period register (T1PER) and the compare register (T1CMP) are set to 13100 ($\frac{65500}{5} = 13100$) and 13090 ($[13100 - 13090] * 0.1\mu\text{sec} = 1\mu\text{sec}$) respectively. The basic function of the period register is to handle the timer period for the occurrence of the next interrupt and to generate an interrupt request to the CPU for performing an SISR. The timer period value for the occurrence of the next interrupt is dependent on the pulse rates. It is determined as:

$$\text{Period register value (T1PER)} = \frac{65500}{\text{Pulse rate}} \qquad (6.1)$$

Where Period register value is an integer to be loaded into the register
Pulse rate = number of pulses (or step command) for each controller sampling time (6.55 msec).

The calculated period value (T1PER) is loaded into the period register which is constantly compared to the value of the free running timer counter 1 (T1CNT) that is set to count up from 0 continuously. When the value of the counter matches the value in the period register, two events take place. First, the free running timer counter is set to zero and restarted. Second, the period interrupt flag is set to high which sends a request for the execution of the timer 1 SISR.

Similarly, the Compare register (T1CMP) is used to control the width of the pulse by switching the polarity of the compare/pulse output from active low to active high. In this particular application, the width of the pulse (1 μsec) is produced by 10 cycles of the system clock (0.1 μsec). To obtain a pulse with a specified width, the compare value in the compare register can be determined as:

$$\text{T1CMP} = \text{T1PER} - 10 \qquad (6.2)$$

The compare register is loaded with the calculated compare value; it is also constantly compared with the value in free running timer counter 1 (TICNT). When the values match,

the polarity of the compare/pulse output pin is switched from active low to active high. It switches back to normal (switching from active high to active low) when the timer counter is reset to zero.

Upon receiving a period interrupt, an SISR is executed: three parameters are evaluated; the timer period for the next interrupt, the motor's rotation direction and the timer value for switching the polarity of the compare/Pulse output from low to high. In order to execute the interrupt in minimum time, a look up table is created in which the period value corresponding to each pulse rate is precalculated and stored in the DSP ram. Hence, for a given pulse rate, the period value to be loaded into the T1PER register can be directly mapped from the look up table without evaluating the equation (6.1) during the interrupt. The period register in the DSP is a double buffered with a shadowed register which means that every new period value will be temporarily stored in a shadow register before being reloaded into a working (active) register when T1CNT is set to zero. Therefore, the period value for the current working period register is always determined at the last interrupt. The advantage of this feature is that it allows the period value of the following period to be changed on the fly. However, this is not the case for the compare register since it is can be initialised to load the newly loaded compare value directly into the working register. Therefore equation (6.2) is evaluated during every interrupt. The newly calculated compare value is immediately loaded to the active compare register for the current duty cycle to use. If the compare value in the compare register is greater than the period value in the period register, a zero pulse is produced in the current duty cycle.

The motor step direction is also being determined in the SISR. When a positive value of pulse rate is given, I/OPB1 (pin 1 on the I/O port B) is set low by setting the bit0 in Register PBDATDIR to 1, whereas if a negative pulse rate is received the bit is set to 0. As a result, the output signal is either high or low. If the output signal is high, the HD65 step motor driver will drive the subsequent motor steps in a clockwise direction and vice verse if the output signal is low.

For example, to obtain the pulse and direction signal as shown in figure 6.3, the related control registers need to be set as follows:

	Section 1	Section 2
Steps per 6.55ms	-4	+5
T1PER	16374	13100
T1CMPR	16364	13090
PBDATDIR (bits 3)	0 (anticlockwise)	1 (clockwise)

Error! No topic specified.

Figure 6.3. Motor is operated in fixed period, variant step profile (eg 5 steps, in 0.00655sec).

To reduce the complication, the serial communication between the PC and DSP is directly handled by a polling routine in the main controller program of DSP. When the SCIRXD pin on the CPU receives a character, the character is shifted to a receive buffer register (SCIRXBUF) and the flag bit RXRDY (in register SCIRXST, bit 6) is set to high indicating that a character is ready to be read. Every time a character is ready, a background routine in the main program interprets the character. If the character is identified as a command status sequence, a function corresponding to the command status is executed, but if it is a data sequence, the character is saved to a defined location in the DSP RAM.

Three interrupts are used in conjunction with the polling systems which allow the CPU to receive the buffer sequence or command sequence consecutively from the PC without time constraints, and to generate various step command pulses to maneuver the motors during each controller sampling interval. The whole operation system can be depicted as in figure 6.4.

Error! No topic specified.

Figure 6.4. The interrupts and polling operation system in DSP.

6.3.3 Software structure for the DSP

The main controller program is written in ANSI C language so that it can be called from any C program, and also provide well-structured, readable software. However, the interrupt service routine (ISR) is written in assembly language in order to service the interrupt with less execution time. Figure 6.5 shows the structure of the SM (stepping Motor) control software for the TMS320F240 DSP.

Error! No topic specified.

Figure 6.5. TMS320F240 SM control program Structure

At the highest level the software consists of initialisation routines and run routines. Upon completion of the necessary initialisation, the background task starts. The background is simply an infinite loop, the functions in the background can be executed only when required or as conditions are fulfilled. In the ISR processing, timer 1 ISR, timer 2 ISR and timer 3 ISR are being executed in timer 1 period interrupt, timer 2 period interrupt and timer 3 period interrupt, respectively. Details of the interrupt operation can be found in appendix A (or see the example in the previous section).

Table 6.1 summarises the processing requirement for each period interrupt in a single chip DSP, TMS320F240. It is obvious that TPINT3 needs a longer execution time compared to TPINT1 and TPINT2. This is due to the microcode bug on multiple interrupt groups that can only be avoided by using a software workaround. Because of this more assembly code is introduced into TPINT3 for a software workaround.

Interrupt group	Time ($x10^{-6}$ sec)
1) TPINT1 (Timer 1 period interrupt)	11.2
2) TPINT2 (Timer 2 period interrupt)	11.7
3) TPINT3 (Timer 3 period interrupt)	12.5
Total time required	35.4

Table 6.1. The interrupts servicing time.

6.3.4 DSP initialisation routine

In order to use a single chip DSP as a multi-axis motion controller, the on chip peripherals and CPU speed need to be configured in order to cope with the sophisticated task. The initial step is set jumper JP5 between the pin numbers 2&3. This causes the V_{ccp} pin voltages to be equal to V_{cc} (5 V), allowing the watchdog timer module to be disabled by software. The next module that needs to be configured is the PLL module. The PLL clock module provides all the necessary clock signals for the F240 such as CPUCLK (CPU Clock) and SYSCLK (System Clock). In this application, the speed of the CPUCLK and SYSCLK need to be run at 40 MHz and 20 MHz respectively. Otherwise, the system is unable to support the high data-transfer rates (eg 11.5 kbyte/sec) between PC and DSP. Therefore, to yield the CPUCLK and SYSCLK with 40 MHz and 20 MHz respectively, the PLL module is configured as follows:

1. Setting CKCR1 (bits 4:7) to 0110b so that 20 MHz crystal oscillator from the EVM is used as a clock in frequency.
2. CKCR1 (bit 3) is set to 1 to select the PLL frequency divided by 2.
3. CKCR1 (bits 0:2) is set to 0x4h.
4. Set the register CKCR0 (bit 0) to 1, in which the system clock frequency is defined as half of the CPU clock

The digital I/O ports module is designed as a pin sharing system. Each of the pins is multiplexed with another function on the TMS320F240. However, the pins for pulse output and I/O functions need to be configured for sole use. Otherwise, the input or output signal are generated internally and the pins are not released solely for motion control signals. For this multi-axis application, I/O port A and I/O port B are configured as input and I/O port C set as outputs. The function of the configured pins is laid out in table 6.2.

Error! No topic specified.

Table 6.2. The pins layout on the DSP

Once the CPUCLK frequency has been determined for the PLL module and the output pins have been configured for the specified functions, three GP Timers in the Event Manager Module can be configured to produce the motor step pulse signals. To create three individual step signals from three different time bases, the control registers in the Event manager such as GPTCON, TxPER, TxCNT, TxCMPR and TxCON (where x $\in$ 1 2 3) need to be set appropriately. Since the CPU is running at 40 MHz, it is easy to obtain a clock base 10 MHz signal by setting control register (TxCON bits(8:10)) to 0x010. The three timers are programmed with the following features:

1. Generating a continuous up count with an asymmetrical pulse output.
2. Stop immediately on emulation suspend.
3. Internal clock with input pre-scaler of 4.
4. Reload the shadow compare register into working compare register immediately.
5. Pulse output Active High

Since the system clock (SYSCLK) is set to 20MHz, the baud rate of SCI can be set up to 115200 by writing 0x14h to register SCILBAUD. Furthermore, the data stream of the SCI can be set to 8 character bits, 1 stop bit and no parity, by writing 0x13h to the SCI communication control register (SCICCR).

6.4 Opto interface Card

An opto interface card was developed to interface between the HD65 stepper motor drivers and the DSP. It is equipped with optocouplers which act as coupling devices between the DSP and the drive system which requires 12 Volts. Consequently, the optocouplers create an insulation barrier to protect the DSP from electrical damage.

In addition, the interface card has incorporated with Dual Precision Monostable Multivibrators (PC74HCT244P) which increase the pulse width so that the terminal on the HD65 driver is able to pick up the signal and cause the motor to step. The width of the pulse should be at least 4 μs.

Error! No topic specified.

Figure 6.6. The Opto interface card

6.5 Control Structure

The DSP is controlled by means of a graphic user interface (GUI) running under Matlab5.3 system on a standard PC. The user graphic interfaces, as shown in figures 6.6 and 6.7, provide a user-friendly environment for monitoring the state of the DSP (TMS320F240) in real time via the RS232 serial port running approximately at 115200 byte/sec (8.7 μsec/per byte). All the functionality in the figures is programmed in function format so that the functions that are associated with the pushbutton are executed immediately as the user clicks on a pushbutton with the mouse.

When the DSP has been initialised and is ready for communication, a user can simply follow these procedures to control the Delta walking machine:

Step 1 : Click on “A”, to enable the Comm Port 2 handle.

Step 2 : Click on “B”, to reset and reinitialise all the variables in the DSP.

Step 3 : Click on “C” to choose a file to restore the data from the pulse rate generation. If not necessary, a default file DSPDATA.txt will be used in step 4.

Step 4 : Option a) If a user wants to perform a continuous path motion control, open a text file named waypoint.txt then insert all the desired cartesian points. Save it and then click on “D” to generate all the pulse rates for this application.

Option b) If a user wants to perform a single or a series of walking steps, open a text file named walking.txt and insert all the step lengths and its directions into the file. Save and click on “E” to generate all the pulse rates for this application.

Step 5 : Click on “F” to initiate the home configuration motion.

Step 6 : Click on “G” then followed by “H”. To execute the continuous path operation mode on the end-effector. If the user wants to quit the execution, click on “I”.

Step 7 : To start with a new execution, go to step 3.

Step 8 : To terminate the program, click on “J”.

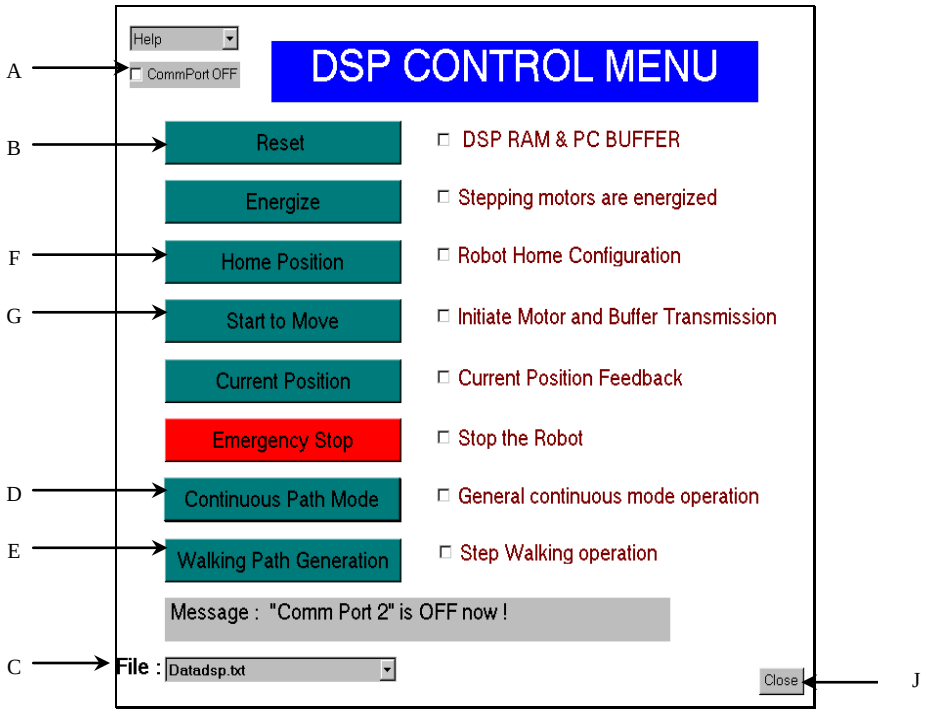


Figure 6.7. The DSP's main control panel.

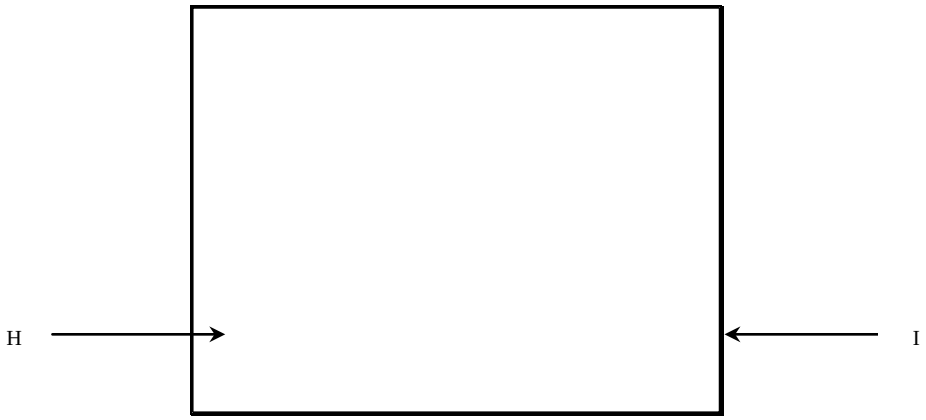


Figure 6.8. The DSP's auxiliary control panel

Chapter 7 Walking Experiment and Discussion

The Delta walking machine was intended to perform all four of the phases of the walking sequence as outlined in chapter 4. However, in the first run, a failure occurred in the phase (2) operation mode. The walking machine was not able to lift itself from the ground steadily (simultaneously with all 3 arms). The foot always made a touch down with one of 3 footpads first then followed by the others after a short time. As a result, one leg was lifted high in the air while the others remained on the same spot. This kind of performance is unacceptable so an investigation was carried out. Initially, the investigation started at the DSP and the interface circuit board. Neither were missing pulses nor was there abnormal behavior observed or detected in the pulse trains. Finally the cause of unsteady lifting was found to be mainly due to geometric errors on the kinematics chain of the Delta walking machine.

Two possible geometric errors arise from the upper arm as illustrated in figure 7.1. One is the ball joints at the end of the tapered arm determine the line B_b and as a result line B_b may not be parallel with the line A_a. Another possible error is that the point $\hat{B}$, midway between the 2 spherical ball joints, is not on the centerline C_L. Nevertheless, the easiest way of correcting the geometric error is adjusting the height of the foot rather than modifying the kinematics chains. Hence, the height on footpads 1 and 2 were raised with 11mm and 9mm respectively.

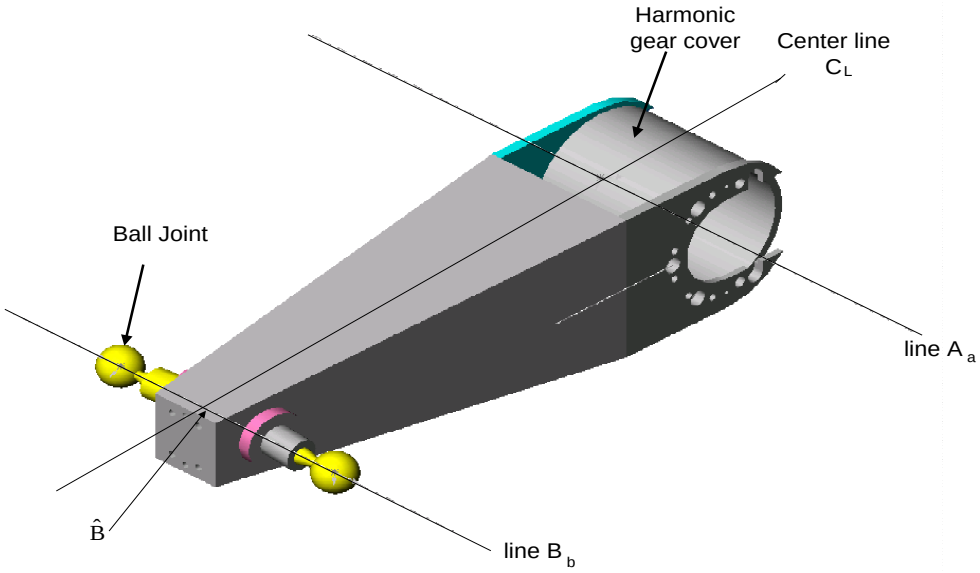


Figure 7.1. Upper arm with ball joints attached

After the correction, the Delta walking machine was able to lift itself steadily. However, another failure mode was encountered this time, the failure happened at the phase (3) operation mode. The whole body dropped to the ground instantaneously while the base platform started to shift across with a small step. It was obvious that the cause of the failure was due to the inadequate torque supply from the stepping motors.

Since the lifting force of the arms is based upon the maximum torque output of the gear drives, this gave a maximum lifting force of about 981N or 100kg weight when the arm was horizontal. A test was carried out on each control axis to ensure that the motors were capable of supplying the required torque output to support the maximum lifting force. In the test, the end of the upper arm was loaded with a payload of 100 kg as shown in figure 7.2. The arm was then stepped upwards until its joint angle (θ) reached zero. The test results showed that only motor 2 (on control axis 2) was able to maintain the static equilibrium against the external load when the joint angle was zero. Motors 1 and 3 both failed to carry the weight any further than a joint angle is about 47 °.

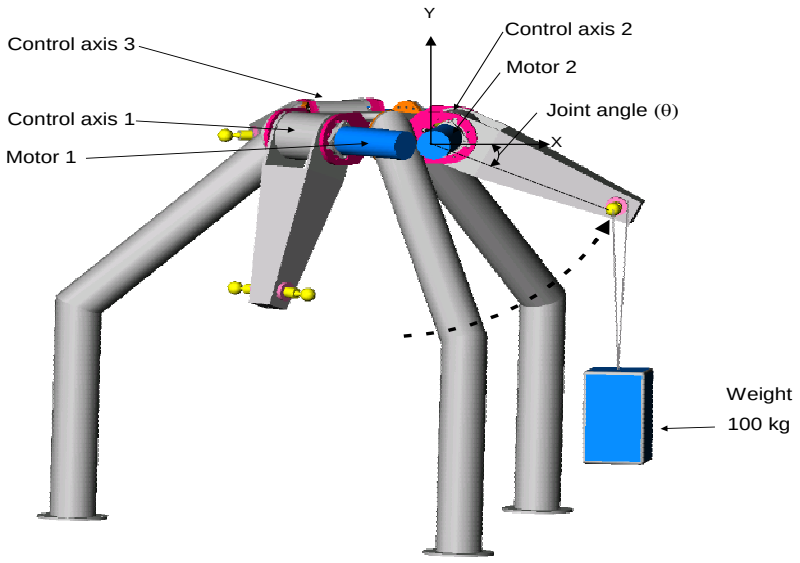


Figure 7.2. Testing the torque capability of motor 2

The inadequate torque supply can be caused by the HD65 stepper motor drivers, the stepping motors, or the harmonic gear drives. A series of tests were conducted to determine what was wrong. First the HD65 stepper motor drivers were examined. If the insufficient torque supply was due to a driver deficiency then using a new driver with the same set up would correct the problem immediately. However, this did not solve the problem. Next, the phase current on each stepping motor was measured to ensure that the power supplies delivered the required current to support the application. The measured phase currents on each motor were in the range of 6.55 amp to 6.88 amp. These measurements and a series of tests confirmed that the drives were in good condition.

The winding connection on each stepping motor was checked and then changed from the parallel to the serial connection. This doubled the phase Ampere-turns and the problem was solved. At the end, each upper arm was capable of carrying a 100kg payload and stayed in static equilibrium with the arm horizontal. The same walking sequence was repeated and the machine managed to perform a step of 120mm although the walking

motion during phase (3) mode was clumsy. During the maneuver, one of the legs started to lean and then hit the floor after travelling a certain distance. As a result, the canted leg was being dragged across the floor. The cause of this problem was the distortion of the travelling platform as shown in figure 7.3. To prevent the travelling platform from twisting, the chassis of the foot was redesigned with additional side reinforcements as shown in figure 7.4. Finally, the Delta walking machine was able to perform an onmidirectional step walking according to the designed walking path.

A film was produced for demonstration purpose as shown in figure 7.5. The Delta walking machine performed two steps walking in the y direction and then two steps backwards to the original position. The film on the CD at the back of the thesis can be run under the program called QuickTime 3.0 that is free to downloaded from the website (http:\\www.apple.com).

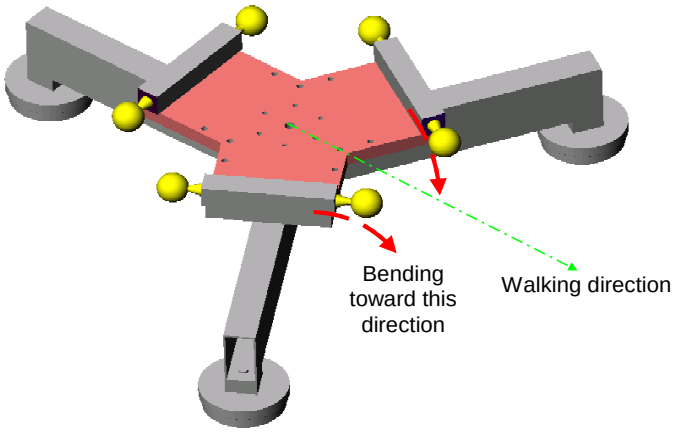


Figure 7.3. The distortion on the travelling platform

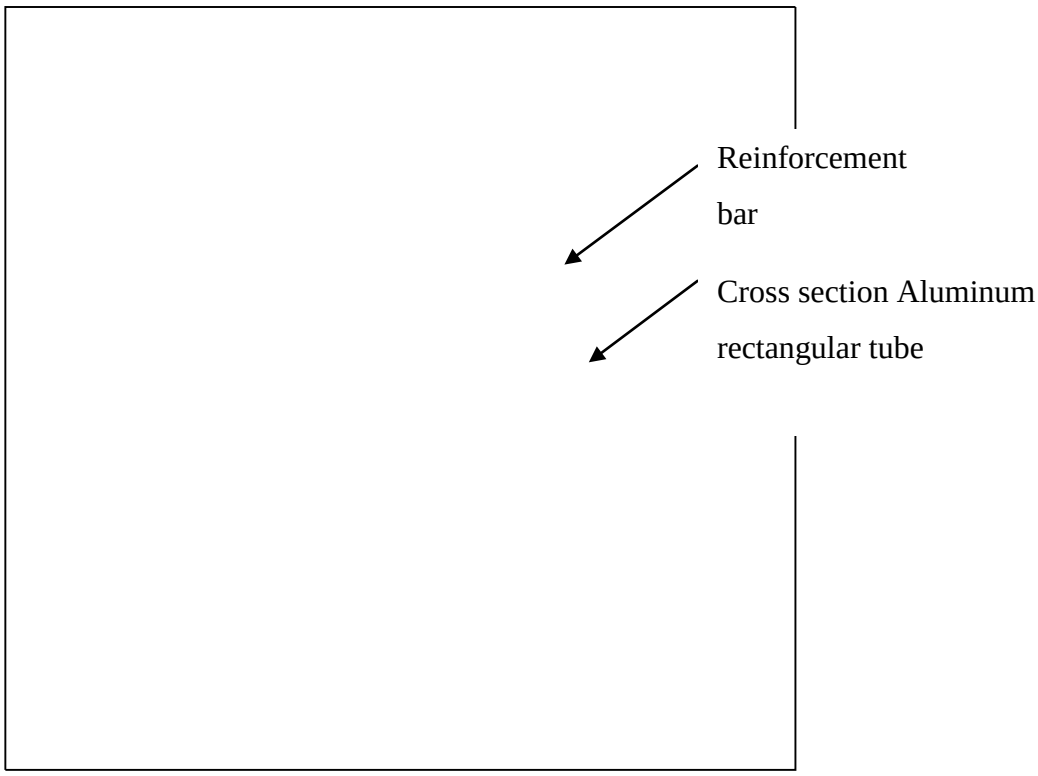


Figure 7.4. The redesigned foot

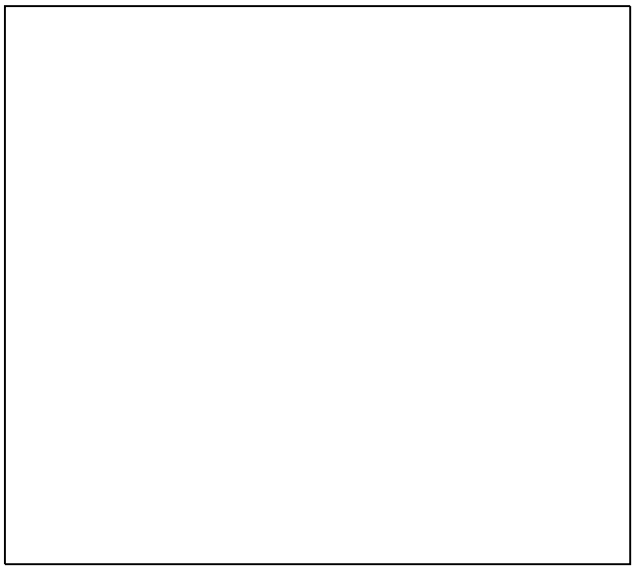


Figure 7.5. One of the walking examples in the movie film.

A video of the walking process may be viewed on the enclosed CD.

Chapter 8 Conclusion and Recommendations

The Delta walking machine was successfully developed. While walking in the laboratory, it is able to perform an omnidirectional step with a maximum length of 120mm on a hard level surface. In addition, the performance of the Delta robot on which the Delta walking machine is based, has been dramatically improved. The Delta robot is still able to perform point-to-point motion as before but now also performs continuous path motion under the open loop step motor control system. This continuous path control was made possible by using a DSP controller based on the TMS320F240. While the DSP receives the step rates and direction signals from the main computer via a serial communication link, it also generates three outgoing pulses and three direction control signals to move the three control axes in synchronism by generating three different time bases on the DSP. The timers are forced to run in synchronisation by the common system clock.

In addition, a GUI control panel was developed to provide a user-friendly software control structure between a user and the DSP. As a result, the user can directly control the state of the DSP from the PC via an RS232 serial port.

The performance of the Delta walking machine can be further improved with extendable feet and legs. With this, the machine would be capable of performing a step while walking in a rough terrain rather than in a well-structured laboratory environment. Furthermore, if the foot is specially designed to include the functionality of arms and legs such as limb mechanisms, the machine should be able to perform tasks at one place and then move on to a new location for another task.

To improve the dynamic performance of the Delta robot/ delta-walking machine from the current static approach, the step motors could be replaced with DC or brushless DC motors that can be operated in a closed loop mode. Possibly feedback around the step motors could also achieve similar performance. In all cases, operating the motors in a

controlled current mode would protect the harmonic drives by limiting the maximum torque output, while still deliver maximum acceleration.

Reference

1. L. W. TSAI, “Robot Analysis”, John Wiley & Sons, 1998.
2. Stuart R.Ball, “Debugging Embedded Microprocessor Systems”, Newnes, 1998.
3. Simon Dan, and C.Isik, “Optimal Trigonometric Robot Joint Trajectories”, Robotica, Vol. 9, page 379-386, 1991.
4. Lothar Rossol and Kenneth A. Stoddard, “Guiding the user to continuous-path control (part I)”, The Industrial Robot, March, page 18-22, 1990.
5. Lothar Rossol and Kenneth A. Stoddard, “Guiding the user to continuous-path control (part II)”, The Industrial Robot, June, page 75-78, 1990.
6. Fu S K, R. C. Gonzalez and Lee G. S.C, “Robotics: Control, Sensing, Vision, and Intelligence”, McGraw-Hill Book Company, 1987.
7. C. C. Cook and C. Y. Ho, “ The application of spline functions to Trajectory generation for computer controlled manipulators”, Digital Systems for Industrial Automation, Vol. 1, Number 4, 1982.
8. R. H. Taylor, “ Planning and Execution of Straight Line Manipulator Trajectories”, IBM J.RES. DELEVOP. Vol.23, No.4, July, 1979.
9. Lorenzo Sciavicco and Bruno Sililiano, “ Modelling and Control of Robot Manipulators”, McGraw-Hill Book Company, 1992.
10. S. J. Lym and K. F. Ehmann, “ A Fast Interpolation Algorithm for Temporally Nonequidistant Waypoints”, Journal of Engineering of Industry. Vol.115, pp 501-507, Nov, 1993.
11. C. S. Lin, P. R. Chang and J. Y. S. Luh, ‘Formulation and optimisation of cubic polynomial joint trajectories for industrial robots, IEEE, transactions on automatic control, VOL. AC-28, No.12, pp1066-1074, Dec, 1983.
12. T. Fukuda, Y. Adachi and Haruo Hoshino, “Posture control of 6-leg walking robot”, IEEE International Conference on Robotics and Automation, page 1006-1011, 1995.

13. Hironori Adachi, Noriho Koyachi and Eji Nakano, " Mechanism and control of a quadruped walking robot", IEEE control system magazine, page 14-19, Oct 1998.
14. Mohamed ABID, Adel Changuel and Ahmed Jerraya, " A hardware /software codesign case study: Design of a robot arm controller", IEEE, page 599.
15. W.C Flannigan, G.M. Nelson and R. D. Quinn, "Locomotion controller for a crab-like robot", IEEE International Conference on Robotics and Automation, page 125-156, May 1998.
16. H. J. Weidemann, F. Pfeiffer and J. Eltze, " A design Concept for legged robot derived from the walking Stick Insert", Proceeding of the 1993 International Conference on Intelligent Robots and Systems, IROS93, July 26-30, 1993, Japan.
17. H. J. Weidemann, F. Pfeiffer and J. Eltze, "Leg design based on biological principles", Proceedings of the 1993 IEEE Conference on Robotics and Automations, May 2-7, 1993 Alanta, Georgia, USA.
18. G.M. Nelson and R. D. Quinn, "Posture control of a Cockroach-like robot", IEEE International Conference on Robotics and Automation, page 157-162, May 1998.
19. S Galt and BL Luk, "Evolutionary design and development techniques for an 8-legged robot", IEE Genetic algorithms in engineering system: Innovations and Applications, 2-4 Sep 1997.
20. S Galt, DS Cooke, AA Collie and BL Luk, "A Tele-operated Semi-intelligent climbing robot for nuclear Applications", IEEE, 4th Annual Conference Mechatronics and Machine Vision in practice, page 118-123, 1997.
21. Randall D.Beer, Roger D. Quinn, Hillel J. Chiel and Roy E. Ritzmann, "Biologically Inspired Approaches to Robotics", Communications of The ACM, page 31-38, Vol 40, No.3, March 1997.
22. Yusuke OTA, "Research on a six-legged walking robot with parallel mechanism", proceeding of the 1998 IEEE/RSJ international conference on intelligent robots and systems, Victoria, B.C, Canada, page 241-248, Oct 1998.

23. Yusuke OTA, “Development of a six-legged walking robot with parallel mechanism”, Technical report, Tokyo Institute of Technology 1998.
24. Haruo Hoshino, “Cooperative motion control of a six-legged walking robot equipped with a manipulator”, ICAR 97, Monterey, CA, July 7-9, 1997.
25. Andre Preumont,” An investigation of the kinematic control of a six-legged walking robot”, Advances in robot kinematics and computationed geometry, 171-178, Kluwer Academic Publishers, Netherlands, 1994.
26. Enric Celaya and J.M Porta, “ A control structure for the locomotion of a legged robot on difficult terrain”, IEEE robotics and automation magazine, page 43-51, 1998.
27. Schmuker U., Schneider A. & Ihme T., “six-legged robot for operations”, 1996 IEEE proceeding of EUROBOT’96.
28. J. C. Grieco, M. Prieto, M. Armada & Pablo Gonzales de Santos, “ A six-legged climbing robot for high payloads”, proceeding of the 1998 IEEE international conference on control applications, Trieste, Italy, 1-4, Sep 1998.
29. Laurent Berthod, “A new controller for the fastest industrial robot”, Industrial Robot, Vol 23, No.6, page 30-35, 1996.
30. G.S. Virk, “Industrial mobile robot: the fucture”, Industrial Robot, Vol 24, No.2, page 102-105, 1997.
31. Noriho Koyachi, Tatsu Arai and Hironori Adachi, “Hexapod with integrated limb mechanism of leg and arm”, IEEE International Conference on Robotics and Automation, pp 1952-1957, 1995.
32. Anibal t. de Almedia and Oussama Khatib, “ Autonomous Robotic Systems”, Springer1998.
33. I.j. Cox and G. T. Wilfong, “Autonomous Robot Vehicles”, Springer-Verlag, 1990.
34. Jones T.P; “Kinematics, dynamics and design of a spherical positioning robot for satellite tracking and other applications”, PhD Thesis, Chapter 3 & 4, School of Mechanical Engineering, UOC, 1996.
35. J. W Jeon; “A generalised approach for acceleration & deceleration of CNC Machine Tools”, page 1283-1288, IEEE, 1995.
36. J. W Jeon; “An efficient acceleration & deceleration for fast motion of Industrial Robot”, page 1336-1341, IEEE, 1996.

37. R. Clavel; “DELTA, a fast robot with Parallel geometry”, pg 91-100, April, Proc. Int. Symposium on Industrial Robot, 1988.
38. F. Sternheim; “Tridimensional computer simulation of a parallel robot. Results for the DELTA4 Machine”, page 333-340, April, Proc. Int. Symposium on Industrial Robot, 1988.
39. F. Pierrot, P.Daucher and A. Fournier, “Hexa: a fast six-DOF fully parallel robot”, IEEE ICAR 91, Vol 2, page 1158-1163.
40. A.Coudourey, R. Clavel and C.W. Burckhardt, “Control algorithm and controller for the Direct Drive Delta Robot”, IFAC robot control, Vienna, Austra, 1991.
41. Karol Miller, “ Experimental Verification of Modelling of Delta Robot Dynamics by Direct Application of Hamilton’s Principle”, IEEE International Conference on Robotics and Automation, page 532-537, 1995.
42. Miller K., "Model based control of DELTA direct drive parallel robot," 26th International Symposium on Industrial Robots, October 1995, Singapore, page. 491-495.
43. F. Pierrot, P.Daucher and J.M Galmiche, “ High speed control of a parallel robot”, IEEE International Conference on Intelligent Robots and Systems, page 949-954, 1995.
44. Pierre Thompson, Gilles Rabatel and France Pierrot, “Performance Comparison of Various Control Strategies for a Mobile Manipulator”, IEEE, International Conference on Intelligent Robotics and systems 95, page 473-479.
45. Stephen J. Mraz, “Streamlining tomorrow’s operating room”, Machine Design, page 44-50, Nov 9, 1995.
46. Russell H. Taylor, “ Planning and Execution of straight line manipulator Trajectories”, IBM J.RES. Develop, Vol 23, No.4, July, 1979.
47. S. Meshkat, “Advanced DSP motion control systems (Part III : Powerful Interpolations), PCIM, pp 40-42, July, 1996.
48. J. D. Decotignie, “Distributed path and speed control in Machine-Tool axis motion”, IECON’91, page 772-777.
49. E. Debourse, “Emergence of DSP’s in Machine Tool axes Control Systems : Application of the Distributed Interpolation Concepts”, IEEE Industrial Automation Systems, pp63-68, Feb 1987.

50. L. Feng, Y. Koren and J. Borenstein, “Cross-coupling Motion Controller for Mobile Robots”, IEEE control systems, pp35-43, Dec 1993.
51. Y. Koren and O. Masory, “ Reference-Pulse Circlar Interpolators for CNC systems”, ASME Journal of Engineering for Industry, Vol 103, pp131-136, Feb 1981.
52. Y. Koren and O. Masory, “ Reference-Word Circlar Interpolators for CNC systems”, ASME Journal of Engineering for Industry, Vol 104, pp400-405, Nov 1982.
53. D Hine, L. Burbridge, “ A microcomputer algorithm for open loop step motor control, Trans Inst MC, Vol 1, No.4, Oct-Dec 1979.
54. B. Benhabib, A.A. Goldenberg and R. G. Fenton, “Optimal continuous path planning for seven degrees of freedom robots”, ASME Journal of Engineering for Industry, Vol 108, pp213-218, Aug 1986.
55. Shafaat Ahmed Bazaz and Bertrand Tondu, “On-line Computing of a robotic manipulator joint trajectory with velocity and acceleration constraints”, Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning, Marina del, CA, Aug 1997.
56. G R Dunlop and A. B. Lintott, “A 3 axis palletising robot which can walk”, IPENZ Conference, 1992.
57. M. H. Raibert, H. B. Brown and M. Chepponis, “Experiments in balance with a 3D one-legged hopping machine”, International Journal of Robotics Research, page75-92, Mar 1984.

Appendix A TMS320F240 Interrupt concepts

When an interrupt signal in the DSP (TMS320F240) is recognized and acknowledged, the CPU (Central Unit Processor) on the DSP board ought to suspend what it is currently doing in the main program and branch to a special subroutine, called an interrupt service routine (ISR), to perform a specific task. Generally, the ISR incorporates a context save, general interrupt service routine (GISR) or specific interrupt service routine (SISR) for a specific task, and a context restore.

During ISR execution, a context save is always executed first, followed by the particular task in GISR or SISR. When the specific task has run to completion, a context restore is carried out and then a return instruction is included at the end ISR to terminate the ISR. This instruction leads the CPU to return to the main program where it left off due to interrupt, and to continue with the proceeding program sequence.

TMS320F240 Interrupt Architecture

The TMS320F240 DSP is capable of implementing up to 32 CPU interrupts. The interrupts supported by the F240 can be divided into four groups: maskable external interrupts (INT1-6), maskable on-chips peripheral interrupts, nonmaskable external interrupts (NMI, RS), and software interrupts (INT8-16, INT20-31). Generally, interrupts are serviced on priority basis. For instant, when more than one maskable external interrupts are triggered such as INT2 and INT3, INT 2 is always serviced first, followed by INT3. This happens because INT2 has higher priority ranking than INT2 and the interrupts are always serviced according to their priority ranking. The F240 device also supports on-chip peripheral interrupts by sharing the maskable CPU interrupts (INT 1-6) with the multiple maskable peripheral interrupts as illustrated on the figure A1.1. This arrangement allows the multiple peripheral interrupts to be isolated from the CPU interrupts, and to efficiently set up their own priority ranking in response to different applications.

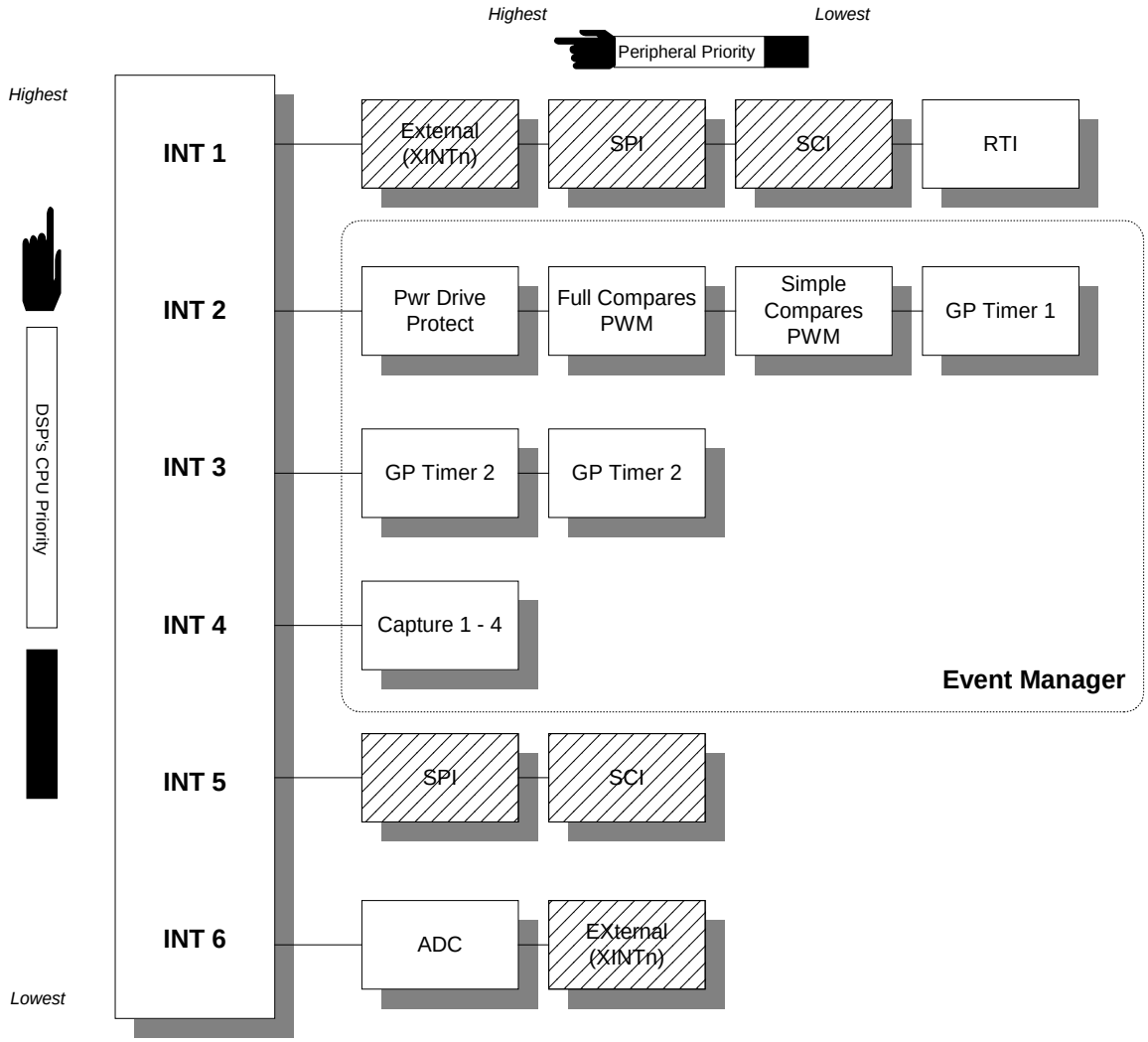


Figure A1.1 On-chip peripheral interrupt priority within F240 Interrupts Hierarchy.

Appendix B Estimation of the Centre of gravity of an Upper arm

The following section shows the procedure of estimating the centre of mass the upper arm (v) along the centre line e-e.

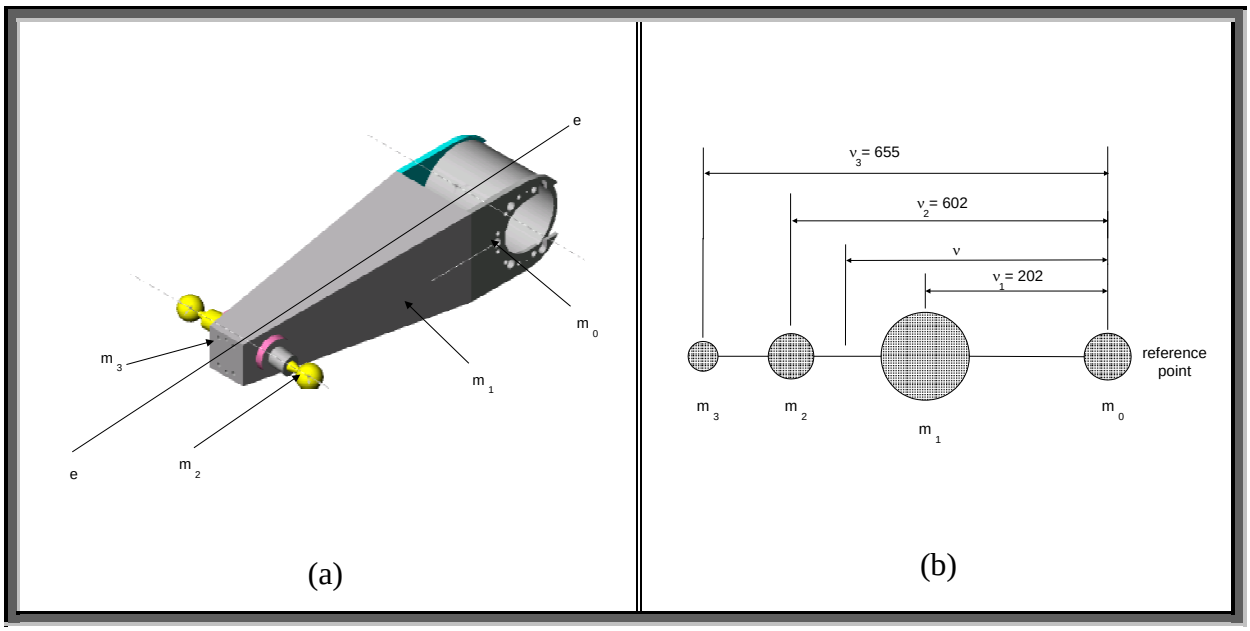


Figure B.1. The mass distribution on the upper arm

We know that $m_1 = 8.4$ kg

$m_2 = 2.35$ kg

$m_3 = 0.4$ kg

So the total weight of the arm : $m_{total} = m_1 + 2*m_2 + m_3 + m_0 = 13.75$kg

Since the middle section of the upper arm is a frustum of a pyramid, its centre of gravity ($a_{frustum}$) can be calculated with the equation as follows:

$$a_{frustum} = \frac{h(A_1 + 2\sqrt{A_2 * A_1} + 3A_2)}{4(A_1 + \sqrt{A_2 * A_1} + A_2)} \qquad \text{(B1.1)}$$

Where $A_1 = 0.0103\ m^2$ (the area of the base surface).

$A_2 = 0.0369\ m^2$ (the area of the top surface).

$h = 0.505$m (the height of the frustum).

Therefore $a_{frustum} = 0.202$ m.

The centre of gravity of the upper arm can be evaluated with the equation as follows:

$$\nu = \frac{\nu_1 * m_1 + \nu_2 * m_2 + \nu_3 * m_3}{\text{Total weight of the upper arm}} \qquad \text{(B1.2)}$$

Where ν_1 is the distance between the centre of gravity of the middle section of the upper arms (m_1) with respect to the original point.

$\nu_1 = a_{frustum} = 0.202$ m.

ν_2 is the distance between the centre of gravity of the two ball joints (m_2) with respect to the original point.

ν_3 is the distance between the centre of gravity of the end cap (m_3) with respect to the original point.

Thus

$$\nu = \frac{0.202 * 8.4 + 0.602 * 5.35 + 0.655 * 0.41}{13.75}$$

$\nu = 0.377$ m (from the reference point)

The constant for equation (4.4) in chapter 4 is evaluated as :

$$\mu = \frac{\text{center\_of\_mass}}{\text{Length\_of\_upper\_arm}} = \frac{0.377}{0.6} = 0.63$$

Appendix C Design of the tripodal foot

The weight of the whole walking machine is about 300kg. As the base platform moves in a particular direction, the centre of gravity will move in that direction as well. Hence, the worst loading condition will be experienced by the foot is the centre of the mass of the machine is moved to one tip of the foot as shown in figure c1.1.

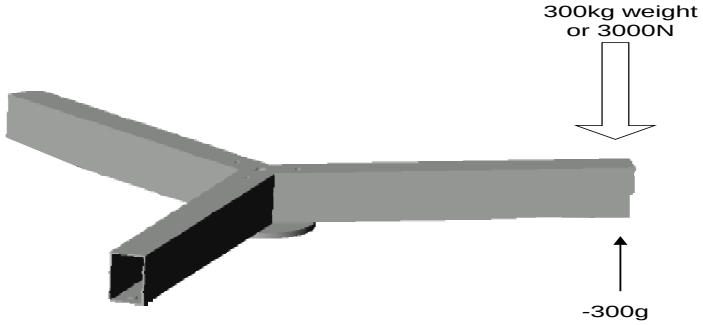


Figure c1.1. The worst loading condition

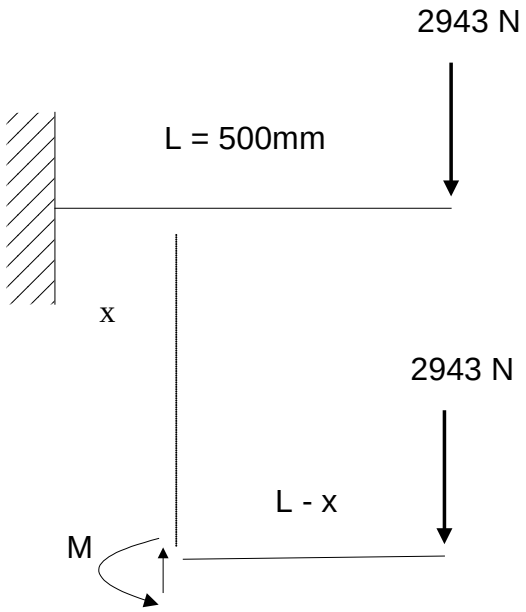


Figure c1.2. The loading model for the foot

The maximum moment occurs at x = 0

$$\sum M = 0: \quad M+2943(0.5-x) = 0$$

$$\therefore \ M = 1471.5 \text{ Nm}$$

The deflection (d) of the rectangular aluminium tube(6063T6, 100x50x3) due to the loading (2943N), can be evaluated with the equation as follows:

$$d = -\frac{PL^3}{3EI} \qquad \text{(C.1)}$$

Where P is the loading on the tube. (N)

L is the length of the tube. (m)

E is modulus of elasticity of the materials. (GN/m^2)

I is moment of inertia. (m^4)

As $I = bh^3/12;$

$= 1.121x10^{-6}\ m^4$

$E = 68.3GPa$ and $L = 0.5$ m

Thus

$$d = -\frac{350*9.81(0.5)^3}{3*1.12110^{-6}*68.3*10^9}$$

$$d = 1.86\,\text{mm}.$$

Meanwhile, the maximum bending stress on the tip of the tube:

$$\sigma = \frac{M*y}{I} = \frac{1471.5*0.05}{1.12*10^{-6}}$$

$$= 76.57 \text{ MPa}.$$

The maximum bending stress bending stress (76.57MPa) is well below the yield strength (215MPa) of the aluminium. Therefore, it is safe for walking operations.

Appendix D Software listing for using TMS320F240 as a multi axis motion controller

This appendix contains the software to implement a synchronisation control on three step motors through a TMS320F240 DSP. The files below are used for controlling the three step motors through three different periodic interrupts (T1PER, T2PER, and T3PER).

File	Description
1) TYPEDFS.H	Header file – data type definitions
2) C240.H	Header file – F240 register definitions
3) Vectors.asm	Interrupt Vector
4) DSPRUN.C	Supervisory program.
5) DSPTABLE.C	A look up table for the timer period value.
6) DSPINT.asm	Threes periodic Interrupt ISR - To generate step command pulses and their step direction.
7) DSPRUN.CMD	Linker command file
8) Rtsnew.lib	Library

All the programs in this appendix are enclosed in the CD.