JOINT CONTROL FOR AN AUTONOMOUS HUMANOID ROBOT

Emanuel Zelniker 33697659

19th October 2001

Emanuel Zelniker
947 Moggill Road
Kenmore QLD 4069

4 October 2001.

Professor Simon Kaplan
Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072

Dear Professor Kaplan,

In accordance with the requirements of the degree of Bachelor of Engineering (Honours) in the division of Electrical Engineering, I present the following thesis entitled “Joint Control for an Autonomous Humanoid Robot.” This thesis project was completed under the supervision of Dr Gordon Wyeth.

I declare that all the work submitted in this thesis is my own, except where acknowledged by references and footnotes. This work, to the best of my knowledge, has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

Emanuel Zelniker.

Acknowledgements

Gordon Wyeth.

All the *GuRoo* team members: Bartek Babel, Jarad Stirzaker, Timothy Cartwright, Nathaniel Brewer, Shane Hosking, Mark Wagstaff, Damien Kee, Andrew Blower, David Prasser, Andrew Smith.

The rest of the squad in the robotics lab.

Abstract

This thesis describes the preliminary design, development and research of a joint control system for an autonomous bipedal humanoid robot. Motors controlling the various movements of the robot will track pre-specified joint trajectories and move to certain positions to achieve a good walking pattern. A combination of low-level control, results from simulation and results obtained from the actual motors will be presented. The latter objective is highly dependent on whether the electro-mechanical design will be built in time.

Firstly, a SIMULINK model is developed to model each joint in the lower body and torso. This model is used to estimate the **P**roportional **I**ntegral (PI) control actions on each joint. The values obtained from the SIMULINK model will then be substituted into the DYNAMECHS simulator and tested for various movements to see whether each joint is reaching its desired position and whether the control action is improving steady state error.

The DYNAMECHS simulator is the mobile robot's test-bed in which all testing of the robot takes place. It is a software package with library files that enable complete dynamic simulation of robots with limbs or "branches". The library files are based on recursive algorithms with an integration step size of 200 microseconds.

Using the proportional integral values from the SIMULINK model, it was possible to reduce the position error of every joint and make sure that each joint is reaching its desired position. The upper body uses RC servo-motor joints with built in control, so they will not require any additional control actions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

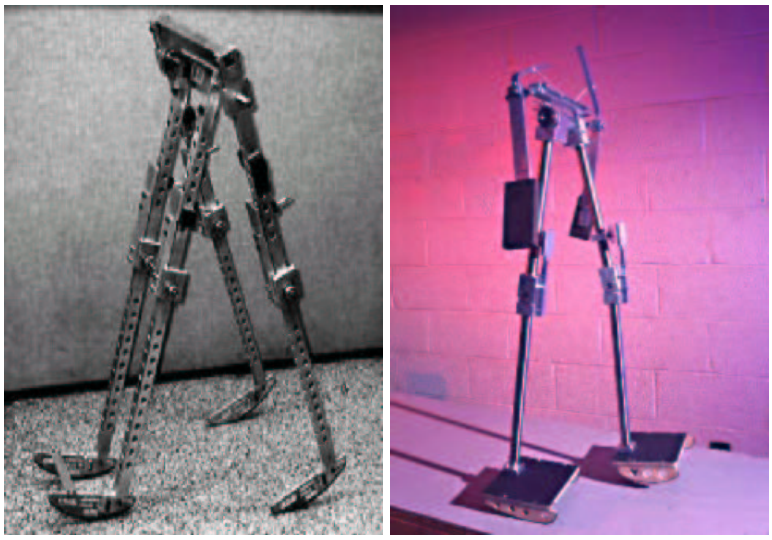
1.1 Introduction

This thesis describes the design, development and research of a joint control system for an autonomous bipedal humanoid robot. Motors controlling the various movements of the robot will track pre-specified joint trajectories and move to certain positions to achieve a good walking pattern. A combination of low level control, results from simulation and results obtained from hardware will be presented.

1.2 History of Bipedal Robots

Research into bipedal walkers has been escalating in the past decade. Biped walkers started from passive walking. Passive walkers rely completely on their natural dynamics and gravity in order to move. McGeer[12] has demonstrated that a system can walk downhill, relying only on its mechanical design with no controllers, sensors or actuators. The design is based around the transformation of the ideal wagon wheel to one where the rim is removed and what's left is the spokes only. These spokes can be considered as a pair of legs modelled as a point mass acting at a distance from the hip joint (or the centre of the rimless wheel).

Unlike the ideal wheel, which is assumed to roll smoothly in 2-dimensions and maintain a certain speed with no loss in energy, the spokes of the rimless wheel will collide with the ground plastically, slowing the system down to conserve momentum about the point of impact. The movements are uneven and impulsive. Figure 1.1a shows McGeer's passive dynamic walker.



(a) McGeer's biped.

(b) Steven Collins armed passive dynamic walker.

Figure 1.1: Some previous passive dynamic walkers.

McGeer improved his 2D model, which had a very inelegant gait pattern, by creating a 3D walker which can move side to side, rocking as a pendulum laterally to achieve balance, much like humans do.

The addition of knees was another innovation to enable foot clearance of the swinging leg. This 2D model had pin-jointed knees with stops to prevent the shank from bending irregularly with respect to the thigh. McGeer referred to this as the prevention of hyperextension. The collision of the shank with the stop of the swinging leg towards the end of its swing phase was modelled to be plastic and the knee remained locked until the next foot strike. Figure 1.1b shows a more recent passive dynamic walker with arms and knees.

It should be noted that despite the simplicity of the above model, it will **only** work when subjected to gentle downhill slopes, thereby limiting its usability.

Since McGeer's design, there has been much research into bipedal locomotion and humanoid robots, some of which are shown in figure 1.2

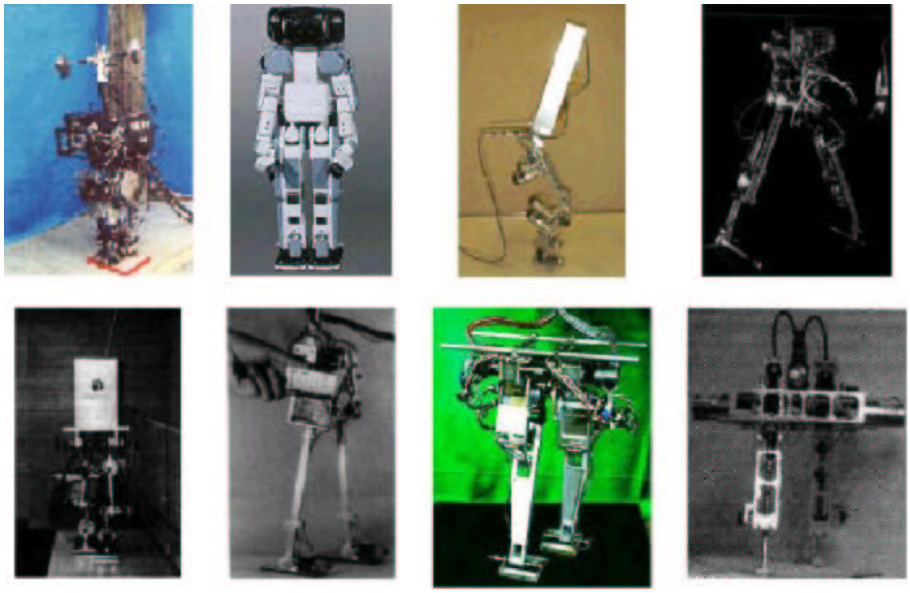


Figure 1.2: Some bipedal robots.
From left to right: WL-10RV1 from Waseda, P2 from Honda, Toddler from UNH, the Moscow State University Biped, SD-2 from Clemson and Ohio State, Biper from University of Tokyo, Meltran II from Mechanical Engineering Lab in Tsukuba, and Timmy from Harvard. Taken from [18].

1.3 What is *RoboCup*?

The robot world cup initiative (*RoboCup*) is a world wide organisation with the aim of promoting education and research. It chooses to use the game of soccer to achieve this aim by organising a world wide competition in which teams from many universities and companies come together and play their robots against each other. It is just as interesting to watch these robots play as watching a real soccer team, but at the same time, it poses a real challenge to all the competitors (humans, that is).

In order for a team of robots to play the game of soccer, a range of technologies must be integrated together including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real time reasoning, robotics and sensor fusion.

At the moment, *RoboCup* has the simulation league, small robot league, the humanoid league which is in its demonstration phase (competition starts from 2002) and many more. All this will ensure the *RoboCup* objective of pushing the state of the art and reach the dream project that:

> **by the mid 21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent world cup champions[19].**

1.4 The *GuRoo* Project

Humans build and shape their environment in a manner that is both suitable and comfortable for themselves, therefore it would make sense for robots to take on the form of a humanoid, so that they can be incorporated more naturally into our society.

The *GoRoo*[1] project is the University of Queensland Robotics laboratory aim to design an autonomous humanoid robot. The robot is meant to play soccer as part of the humanoid league *RoboCup* division. However, at this early stage of development, it is expected that the robot will balance, crouch and walk autonomously in a straight line. Eventually, the robot will be able to turn and stand from a horizontal position with its "face" to the ground. It will also be required to cope with external disturbances such as making contact with a ball and other players, so it will be able to play soccer.

The aim is to make the robot about 1.2 meters tall, weigh a total of about 30 kilograms and have 23 degrees of freedom.

[1]The *GuRoo* stands for Grossly Under-funded Roo. The amount of money injected into this project is nothing compared to the millions of dollars that other companies have spent on their humanoid projects, such as Honda or Sony, hence the term Grossly Under-funded. The Roo is meant to comply with the University of Queensland Robotics laboratory tradition of naming its robot projects, among which are the *RoboRoos*, *the ViperRoos* and the *CrocaRoos*.

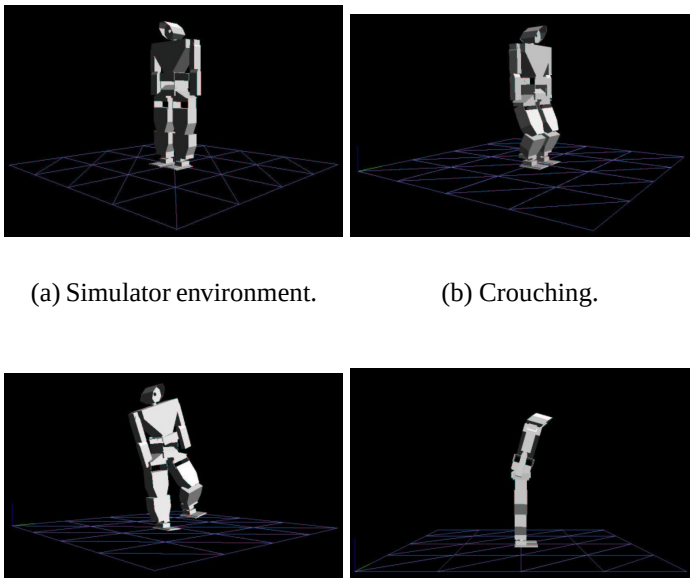
In order to realise this project, the *GuRoo* project was divided into several subcatagories, each of which was undertaken by 1 or 2 people. The subcatagories are: Mechanical design, actuation and sensors, power systems, joint controllers, internal network, vision hardware and software, joint control software and balancing software or development of a "good" gait. The robot requires a vision sensor to detect objects in the soccer field and must be able to detect objects such as the boundaries of a the field, the ball, the goals and other players from its own team and the opposition. The vision system is local and is very similar to the system currently used by the *ViperRoos*[23].

The challenge in designing a controller for the *GuRoo* is to determine what data is needed, how to measure it and how to use the data to meet the desired specification of the hardware. In other words, different subsystems must be integrated in a hierarchical control logic that enables the robot to function as a whole system to be able to walk and play soccer.

The mobile robot test-bed is a of the form of a dynamic simulator in which all testing of the robot takes place. The simulator was developed in the *DynaMechs* project[14], a software package with library files that enable complete dynamic simulations of robots with limbs or "branches", that is, a star topology. It is possible to use the package to simulate mobile robots and robots with a fixed base, in different environments, ranging from rough terrain to underwater conditions to different gravity fields. It also enables the user to simulate the various aspects of the motors and drives that control the robots movements and the internal CAN network. The library files are based on recursive algorithms with an integration step size of 500 microseconds. The graphical display is in an OpenGL environment and is updated every 5 milliseconds of simulated time[25].

1.5 Importance of Joint Control to the *GuRoo*

Figure 1.4a shows the degrees of freedom in each joint of the *GuRoo*. Fifteen of these will be high power joints that will require control. These are the three joints at the waist



(a) Simulator environment.

(b) Crouching.

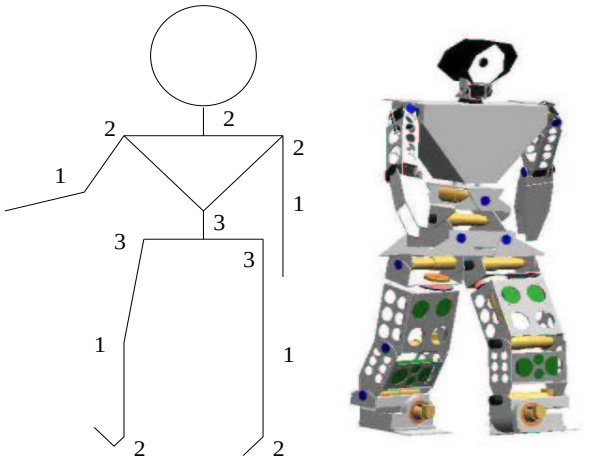
(c) Balancing on the right foot.

(d) Leaning forwards.

Figure 1.3: The DYNAMECHS simulator in an OpenGL environment.

and hips, the joints at the knees and the two joints at the ankles. Controlling these is important because in order for the *GuRoo* to walk properly based on pre-specified joint trajectories, there must be feedback employed so that the actual position of each joint in time can be compared to the desired pre-specified joint position at that point in time. If there is a discrepancy between the actual joint position and the desired joint position, compensation must be provided in order to drive the discrepancy or error between the input and the output to zero.

As was stated above, the *GuRoo* will move based on pre-specified joint trajectories. These will be in the form of position and velocity commands sent by the central controller every 2 milliseconds. In other words, the position and velocity commands from the central controller will be updated every 2 milliseconds. The joint feedback will occur locally on the corresponding controller board that its motor is connected to. The actual position of the joint is fed back locally and is compared to the most recent command from the central controller. Figure 1.5 shows this diagrammatically.



(a) Skeletal model of GuRoo.

(b) CAD model of the GuRoo

Figure 1.4: *GuRoo's* degrees of freedom.

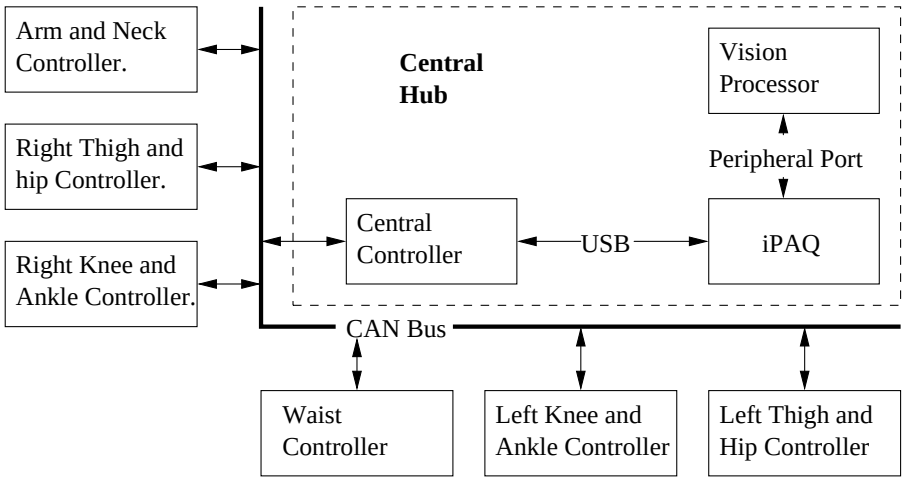


Figure 1.5: Block diagram of the distributed control system. Taken from [25].

1.6 Outcomes of Research

The need for a joint control system for bipedal robots was described in the previous sections. The specific aim of the proposed research is stated as follows:

1. To develop a prototype model of a joint controller system for the autonomous robot, *GuRoo*, for successful tracking of pre-specified individual and multi-axis joint tra-

jectories and meet some of the requirements for the *RoboCup* contest in the humanoid league division.

2. To develop a mathematical model and simulate the control loops of the system. Specifically, to develop a PI controller with feed-forward cancellation to achieve accurate trajectory following.

3. To test the PI controller for each joint in the lower body and torso in the DYNAMECHS simulator for various movements and look at the position error of each link.

1.7 Thesis Overview

Chapter 1 presented a discussion of the problem statement, motivation behind the research and the research goals.

Chapter 2 is a literature review outlining the existing knowledge in the area of bipedal control and an introduction to motor control theory.

Chapter 3 describes the specifications of the report.

Chapter 4 describes the control model for the *GuRoo* and the model used to simulate the different joints in the lower body in SIMULINK.

Chapter 5 will deal with the software for implementing the control loops in the DYNAMECHS simulator.

Chapter 6 deals with the testing and results of the control loops simulation in SIMULINK and the DYNAMECHS software.

Chapter 7 concludes the thesis with some recommendations and future work.

Chapter 2

Literature Review

2.1 Introduction

This chapter deals with the work done by other researchers who have investigated control methods for mobile bipedal robotic systems. Bipedal locomotion can be categorised into two groups: Passive dynamic walkers which have already been discussed and active walkers.

2.2 Active Walking

Active walking can be split into two groups:

- Dynamic actuation. This includes “intelligent” fuzzy rule based control, neural networks and adaptive control, and
- An entirely power based system which relies on pre-specified trajectory tracking. It is the power based system that employed to make the *GuRoo* walk at this early stage of development.

2.2.1 Dynamic Actuation—Adaptive Control

The control of plants whose dynamics vary and are constantly subjected to the uncertainty of the outside world is a problem which bears a lot of theoretical and practical importance.

If we consider a robotic arm for instance, using a computed-torque approach, which is a method used to cancel not only the offset due to gravity, but also the coriolis and centrifugal force, friction and the inertia tensor of a dynamic manipulator, there would be a need to know an estimate of all these parameters, which means that the computed-torque method will suffer from uncertainties in the estimates of the manipulator parameters above. The adaptive control technique is a more robust method which can make a controller be less sensitive to noise and uncertainty [20]. An adaptive controller will continuously change because the parameters of the system are a changing function with time.

An example of an adaptive controller is the **M**odel **R**eference **A**daptive **S**ystem (MRAS) shown in figure 2.1. The performance of the desired system (Reference model) is modelled using a mathematical model simulation. The plant outputs are compared with the simulated model outputs to generate an error signal. This error signal is used to modify the input to the plant and/or to update or change its parameters with time. In other words, the model forces the model output and the system output to zero for a robotic manipulator operating in an unknown environment.

The problems of using an adaptive control scheme as in figure 2.1 on robotic manipulators is due to the fact that parameters such as inertia and the offset due to gravity change rapidly with time as the manipulator moves with respect to the reference model's time constants. This means that the error signal will need to be updated quicker and the adaptation will need to occur quicker as well, however, with the rapid advances in microprocessors, adaptive control techniques are having more and more success in the field of robotics.

There are several methods of adaptive control. Dubowsky and Desforges [5] derived an adaptive control system that minimises a quadratic function of the error signal in figure

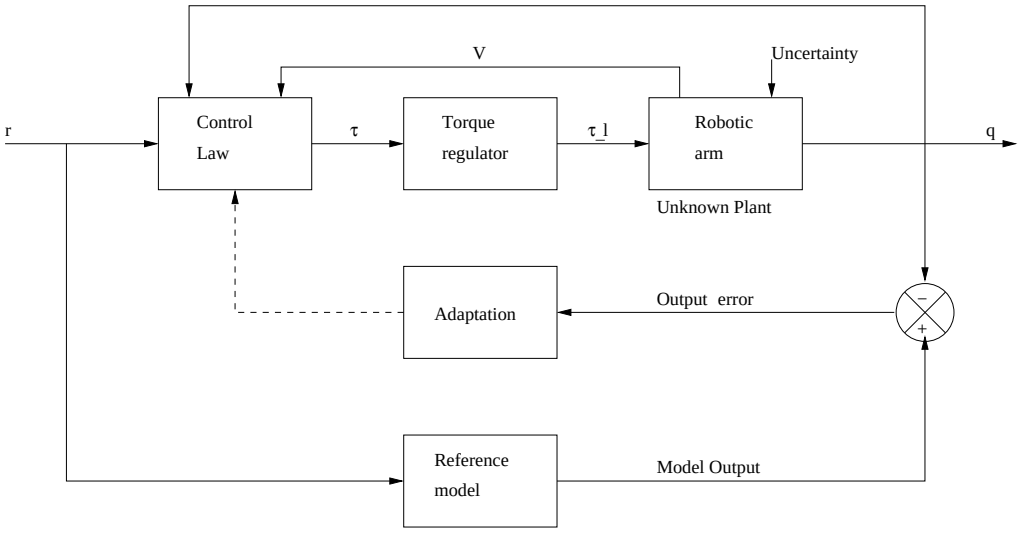


Figure 2.1: Model Reference adaptive System of a robotic manipulator.

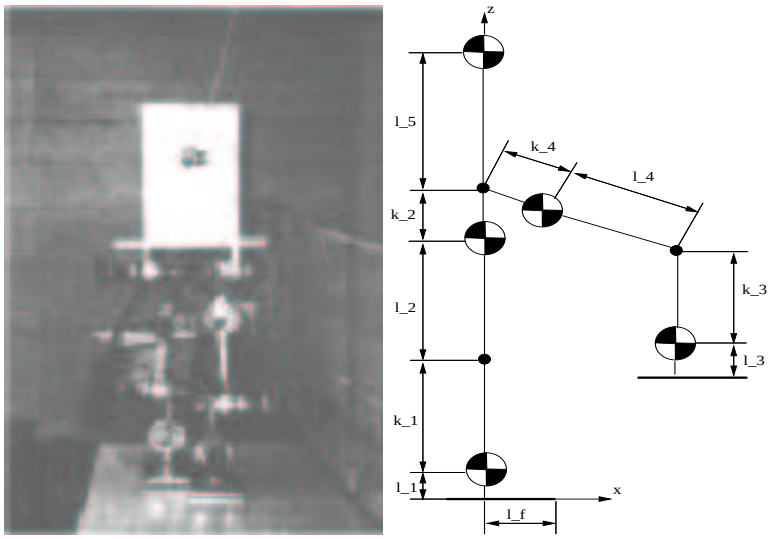
2.1. Arimoto and Takegaki [1] applied an adaptive control system based on local parameter optimisation to a robot described by a linear time varying motor derived from a linearisation around the desired trajectory. Nicolo and Katende [15] use an approach where a separability property of Newtonian systems allows the construction of an adaptive control scheme with a partially linear controller.

2.2.2 Power Based Systems

Power based systems rely on tracking pre-specified joint trajectories. The advantage of such systems is that they are relatively easy to control because a “good” gait pattern is already pre-determined. The disadvantage is that when these systems are exposed to different environments, they will not operate robustly as the pre-specified trajectories may not be compatible with the surrounding conditions.

A power based system will generally involve a central controller that will send and update position commands or velocity profiles to the plant (or plants) and each plant will have a local controller performing some control law depending on the application. The controller can be a position controller, a proportional plus velocity controller, a proportional plus integral plus derivative controller or a proportional derivative plus gravity controller.

An example of a power based robot is SD-2 by Golden and Zheng[8]. Their motivation was to develop a biped robot that can move in industrial areas such as nuclear power plants designed for humans. Their design concentrated on the kinematic behaviour of human stair climbing and the phases involved. They decomposed one period of stair climbing into nine rudimentary phases. They are based on making sure that the Centre of Gravity (COG) remains on top of the weight bearing foot during the single support phase of the gait. Figure 2.2a shows a photo of SD-2. The robot has 8 DOF; two in each hip, one in each knee and one in each ankle.



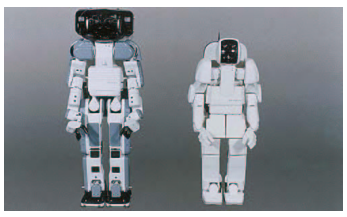
(a) SD-2. (b) Model of SD-2.

Figure 2.2: SD-2 from Clemson Ohio State.

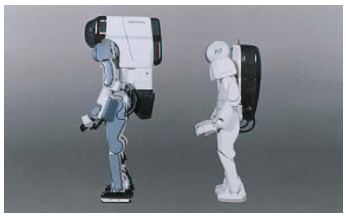
After selecting the desired location of the COG during each stage of the gait, it is now a matter of performing the inverse kinematics in order to work out all the joint angles during each stage. Once this is done for all phases of the gait, the joint positions are placed into the control software and empirically tested. As each position was tested, adjustments were made to the original data in order to compensate for minor inaccuracies of the model. If an adaptive control scheme was applied to the problem, this adjustment stage could of been avoided.

2.3 Living and Breathing Bipeds

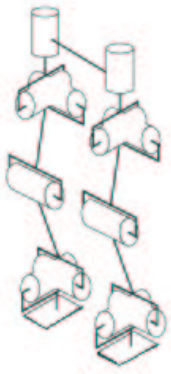
Out of all the previous attempts at building and controlling bipedal robots, the Honda P2 and P3 biped robots shown in figure 2.3a and 2.3b are some of the very few that closely resemble a human structure and performance. Honda began its research towards humanoid robots in 1986. At the time, knowledge on legged locomotion was sparse and Honda started by studying bipedal walking mechanisms. Their aim was to develop a robot able to coexist and collaborate with humans and to perform tasks that humans cannot[9, 7].



(a) P2 is 1820 mm tall and weighs 210 kg. P3 is a more compact version.



(b) Side view of P2 and P3. P3 is some 22 cm shorter and 80 kg lighter.



(c) Schematic view of P2's DOFs.

Figure 2.3: P2 in comparison to P3.

In 1996, Honda announced the development of a humanoid robot with 2 arms and legs called P2. P2 could balance, walk on level ground, climb stairs, turn and push objects. It did this by following a combination of a pre-defined path for the Zero Moment Position (ZMP) and a dynamic balance control similar to that of humans. The balance control was done by comparing the desired ZMP to the centre of actual total ground reaction force (C_ATGRF). The desired ZMP is a point on the ground about which the desired inertia forces have a net cancellation. Ideally, the ZMP and the C_ATGRF will be co-incident and the robot will have balance, otherwise the robot will experience a tipping moment as

follows

$$M_T = (desired\,ZMP - (C\_ATGRF)) \times Vertical\,Component\,of\,Net\,Inertia\,force \quad (2.1)$$

The tipping moment is cancelled via "ground reaction force control" and the updating of the desired ZMP in time is done by "model ZMP control". Ground reaction force control positions the feet on the ground appropriately to control the position of the C-ATGRF sensed by a 6-axis force sensor. Model ZMP controls the desired ZMP of the robot to prevent the occurrence of tipping moments as per equation 2.1.

Figure 2.3c is a schematic view of the lower body of the P2 humanoid robot. The model is of particular interest because it closely corresponds to the *GuRoo* physical model in the lower body. It has the same number of degrees of freedom in each leg (see figure 1.4) and uses similar sensors to control for balance as the *GuRoo* will be using.

2.4 Motor Control

This section presents an explanation of basic motor control theory. The next three sub-sections will deal with open and closed loop systems, the crucial elements of a control system and the different types of control actions.

2.4.1 Open and Closed Loop Systems

Open loop systems are systems where the output bears no effect on the control action. The output is not measured nor fed back to be compared to the input. As a result, open loop systems will not perform well in the presence of external or internal disturbances and are limited to situations where the relationship between the input and the output is well known, that is, the accuracy of the system depends on calibration[17].

An example of an open loop system is a motor which drives an ink cartridge horizontally across a piece of paper in a bubble jet printer for text to be printed. Once a line of text

has been printed onto the paper, the motor moves back across the page for the next line of text to be printed. If a fault such as a paper jam occurs, the motor will continue to print text across the page regardless of the paper jam. This is why when the paper jam is removed, there is usually a smudge of ink on the paper.

closed loop systems are feedback systems, where the output has an effect on the control actions. The output is measured and fed back to be compared to the input. These systems can operate in the presence of disturbances and can provide accurate control to a given plant, whereas doing so is not possible in open loop systems, however open loop systems should be used in situations when the inputs are known in advance and there are no disturbances[17].

An example of a closed loop system is a missile which can be controlled by generating torques via deflection panels on the missile's body. These torque commands are generated by a computer guidance system that maintains the trajectory of the missile in the air.

2.4.2 The Crucial Elements of a control System

Control systems provide an output response for a given input. Figure 2.4 shows a basic closed loop control system. The function of a control system is to maintain the value of $Y(s)$ as closely as possible to $R(s)$ and to correct any error and uncertainty as quickly as possible. The output is fed back and is compared to the input $R(s)$. This generates an error signal which drives the plant $G(s)$. When the input equals the output, the error signal is zero and the process reaches an end.

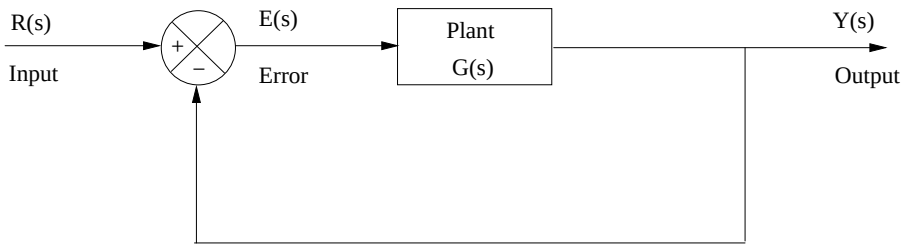


Figure 2.4: Basic closed loop control system.

Figure 2.4 shows the crucial elements of a control system: a controller operating on the input; a sensor that measures the output signal and feeds it back to the input; and a comparison block that measures the error and performs the control action.

2.4.3 Types of Control Actions

This section will discuss some common control actions to meet transient and steady state specifications. Compensation is performed for two main reasons, to get rid of disturbances and to get rid of uncertainties.

2.4.3.1 Proportional Plus Integral Action

A **P**roportional **I**ntegral (PI, figure 2.5) compensator is an example of a lag compensator. The integral path accumulates the error during the transient response, so the energy build up in the integral path will increase at a decreasing rate. Therefore, towards the end of the transient response, the integral path can be used to supply enough energy to drive the plant to its required state, to meet a required steady state error or to eliminate it completely.

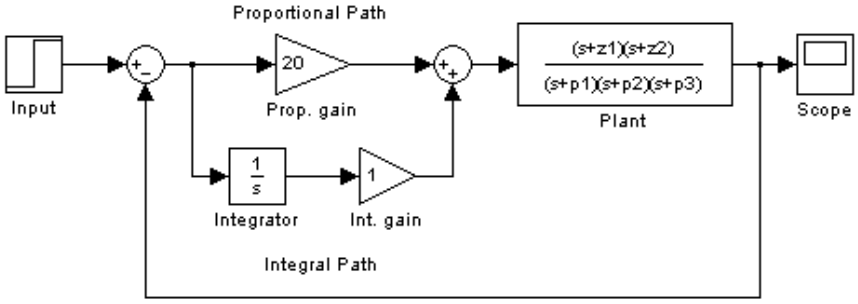


Figure 2.5: Proportional Integral action.

Such a compensator might be used to overcome stiction in a gearbox which transmits power from a motor to a load for instance. If there was just Proportional compensation, the signal towards the end of the transient response might not be strong enough to overcome the stiction in the gears. But with the integral component, enough energy can be put into the plant to overcome the stiction in the gears.

A PI compensator is a low pass filter, attenuating the high frequency components of the signal. The transfer function is $K_p\left(1+\frac{1}{Ts}\right)$. The zero is positioned at $-\frac{1}{T}$ and a pole is placed at the origin. This increases the order of the system (it can be seen that at zero frequency, there will be infinite gain) and thus improves the steady state response. It is possible to use a root locus technique to select K_p and T so that the transient response exhibits small to no overshoot for a step input [6].

2.4.3.2 Proportional Plus Derivative Action

A **P**roportional **D**erivative (PD, figure 2.6) compensator is an example of a lead compensator. Initially, the error between the input and the output is large, so the derivative of the error will be large. This will result in a large signal driving the plant at the start, speeding up the transient response. As the error between the input and the output diminishes, so will the derivative of the error, and the derivative path will become less significant towards the end of the transient response.

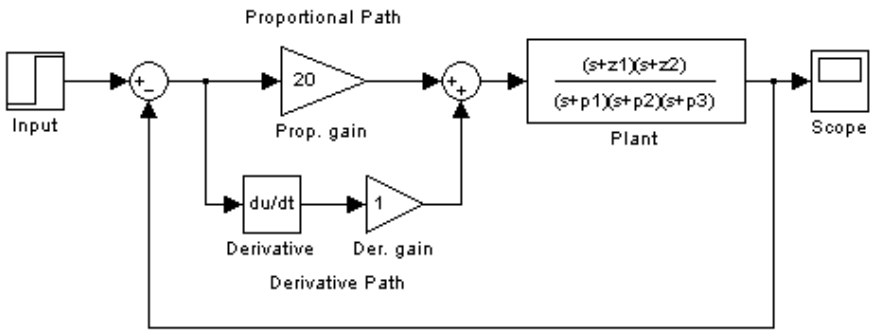


Figure 2.6: Proportional Derivative action.

A PD compensator is a high pass filter having a transfer function $K_p(1+Ts)$. K_p needs to be selected in order to meet steady state requirements and by positioning the zero appropriately, it is possible to speed up the original system. It should be noted that differentiation is a noisy process and may introduce large unwanted signals or saturation [16].

2.4.3.3 Proportional Plus Integral Plus Derivative Action

As the name suggests, figure 2.7 shows a **P**roportional plus **I**ntegral plus **D**erivative (PID) compensator, which is a combination of PI and PD control to improve both the steady state error and transient response separately. A PID compensator has the advantage of all three actions. The transfer function is $K_p\left(1+\frac{1}{T_i s}+T_d s\right)$ and this is an example of a lead-lag filter.

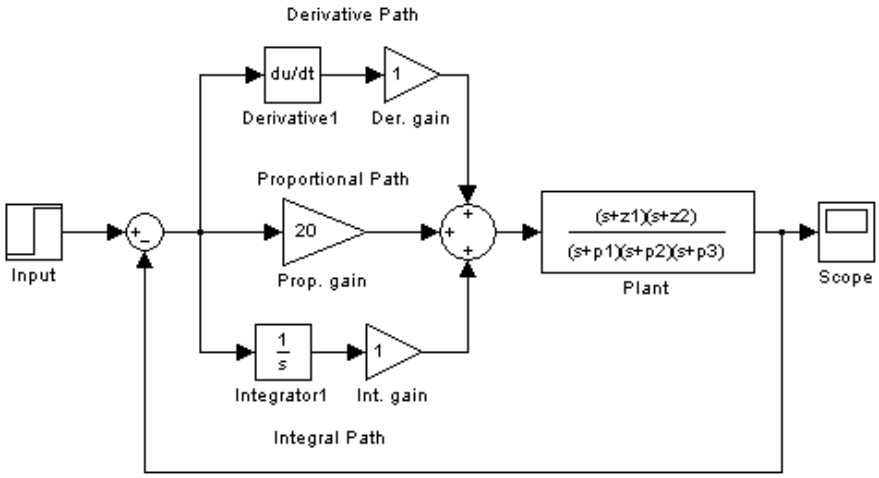


Figure 2.7: Proportional plus integral plus derivative action.

2.4.3.4 Feedforward Cancellation

Feedback control is an error-based process, so by definition there must be an error signal (see figure 2.4) for the system to be doing anything. Feedforward control takes an alternative approach. A model of the system is designed and the inverse dynamics are solved for the input. If the plant dynamics are modelled as

$$g(t) = R(r(t), g(0)) \tag{2.2}$$

then

$$r(t) = R^{-1}(g_d(t)) \tag{2.3}$$

This is the inverse dynamics of the plant, given the desired state of the system, $g_d(t)$. This has the potential for accurate control, but the model of the plant must be sufficiently accurate, otherwise, the deviation from the target reference may be a rapidly increasing function with time.

The following figure illustrates how feedforward control can be used with PI compensation. It is possible to obtain an output from this system which tracks the input with only a time delay.

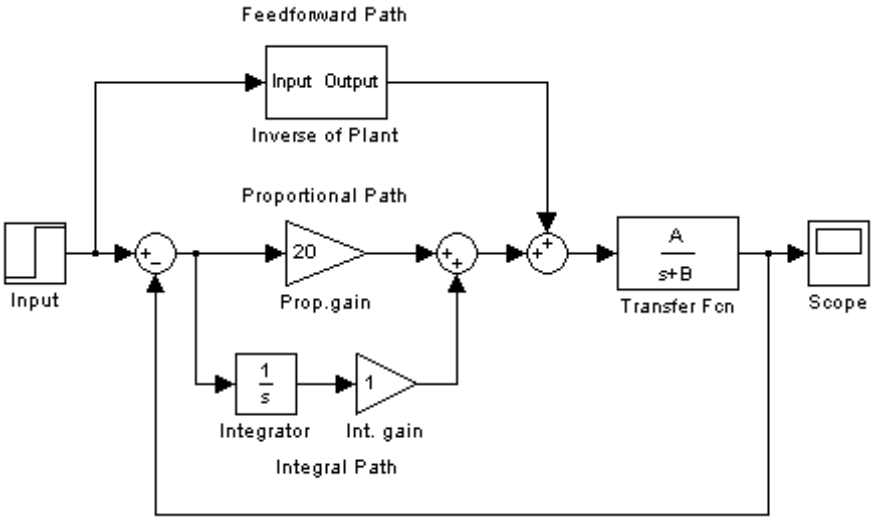


Figure 2.8: feedforward with PI compensation.

Chapter 3

Problem Specification

3.1 Introduction

This chapter deals with the specifications of the thesis, what the aim is and how it is going to be achieved. Firstly, a SIMULINK model will be developed to model each joint in the lower body and torso. This model will be used to estimate the proportional and integral constants to be used in the PI compensator. These values will then be substituted in the DYNAMECHS simulator [14] and tested for various motions to see whether each joint is reaching its desired position and whether the PI compensator is improving steady state error. Lastly, and this depends on whether the electro-mechanical design will be built, the compensator will be tested on the actual physical motors so that the results from simulation can be compared to the actual response of the motors.

3.2 Simulink Model

The SIMULINK model is an approximations designed to estimate the PI compensator for each joint in the lower body. By all rights, the loading on each joint in the lower body will change with time as the robot moves. The simulink model does not take this into account, but rather, it models the load on every joint as a static point mass acting at its

estimated radius of gyration from the joint at the worst case angle. The worst case angle is the angle at which the load is maximum for every joint—because of the offset due to gravity—taking into account the limits to which each joint can move.

Appendix A shows a CAD model of the *GuRoo* and table 3.1 below lists the static mass (M_s) on each joint, the radius of gyration (r_k) that the mass is acting from the joint and the worst case angle (θ_{wc}) for each joint. The static mass on each joint is calculated by adding up all the individual masses of the links and motors that the joint will move. For example, the knee joint (joints numbers 5 and 6) will move the lower leg ($1.72\,kg$), the frame that will move the foot plus the ankle roll motor ($1.48\,kg$) and the foot ($1.17\,kg$), giving a total of $4.37\,kg$. Similar calculations can be performed for all other joints and the results are listed in table 3.1. The table in appendix A lists the individual masses of each link.

It can be seen that the loadings and the radii of gyration are the same for some joints due to the natural symmetry of a humanoid configuration. For instance, the ankle motors responsible for the roll and pitch of the foot on each leg (4 motors) will have the same loading and radius of gyration. When these motors move the entire body sideways or forwards, the worst case angle will be seven degrees. However, when these motors are responsible for moving the foot, the worst case angle will be ninety degrees.

The knee joint on each leg (2 motors) is responsible for moving the lower leg to provide for clearance for the entire leg to swing forward, and the worst case angle will be ninety degrees. Using the knee joint to move most of the body mass above it is not a realistic situation and will therefore not be considered.

The joint responsible for twisting each leg (2 motors) will play a part in moving the robot forward. When the leg is swinging in the air, the joint will be moving (or twisting) the leg, but when the leg is acting as the weight bearing member (while the other leg is coming forward), the joint will be moving the upper body above it. Both situations should be considered, however since the joint is moving mass in a plane that is parallel to the ground, there will be no offset due to gravity and no worst case angle.

The hip joint responsible for moving each leg backwards and forwards (2 motors) will move the leg during its swing phase and move the upper body when that leg is the weight bearing member. When the joint is moving the leg, the worst case angle is ninety degrees, but when the joint is moving the upper body, the worst case angle is fifteen degrees.

The hip joint responsible for moving each leg sideways (2 motors) will move the leg during its swing phase and will lean the upper body sideways when that leg is the weight bearing member. When the joint is moving the leg, the worst case angle is ninety degrees, but when the joint is leaning the upper body sideways, the worst case angle is fifteen degrees.

The torso has 3 degrees of freedom: front to back; side to side; and twist. However, the joints responsible for each degree of freedom are on top of each other. First comes the front to back motion, then the sideways motion and on top of that, the twisting motion (see appendix A). The front to back and sideways motions (2 motors) will move approximately the same amount of mass as they are very close to each other, and will be responsible for moving the upper body from front to back and sideways respectively. The worst case angle for both joints is twenty degrees.

The joint responsible for twisting the upper body (1 motor) will move mass in a plane that is parallel to the ground, so there will be no offset due to gravity and no worst case angle. Also, because of the symmetry of the torso and upper body, the effective point mass will lie along the axis of rotation of the twist joint and the effective radius of gyration will be zero, so the only loading on the torso twist motor joint will be the rotor inertia of the motor itself.

The next chapter is devoted entirely on the SIMULINK model for each joint in the lower body and will elaborate on the material discussed in this section.

3.3 DYNAMECHS Simulator

As was mentioned in chapter 1, the DYNAMECHS simulator is the *GuRoo's* test bed in which all testing of the robot takes place. Once the proportional and integral constants for each joint are found in the SIMULINK model, they will be entered into the DYNAMECHS simulator and trialled for various motions such as crouching, balancing on the right foot and leaning forward with the upper body. It is possible to extract from the simulator information such as the torques on individual joints in time, the currents in the armature of their respective motors, the total current drawn by all motors at any point in time and the positional error at any point in time. It is important to make sure that the current in the armature of any motor doesn't exceed four amperes and the total current doesn't exceed 20 amperes. The voltage input into a motor must not exceed forty volts. This is due to the physical limitations of the motor and the power systems design [3]. The DC motor specifications for the joints of the lower body and torso are listed in the data-sheet in appendix E. Some of these characteristics will be used to model the motors in chapter 4. Chapter 5 will deal with the DYNAMECHS simulator control software in more detail.

3.4 RC Servo-motors

Unlike the DC motors which will require external control, the RC servo-motors will not require any control. These motors come with their own controller chip built in, and therefore, the RC servo-motors can be considered as a black box which will receive a position input from the central controller every two milliseconds (see figure 1.5) and the control law will be performed internally within the motor.

3.5 Comparison to Electro-mechanical Design

The proportional integral constants from the SIMULINK model will also need to be tested on the actual motors so that results from simulation can be compared to real results from

the motors. This is highly dependent on whether the electro-mechanical design will be built in time. The mechanical design and selection of actuators is not in the scope of this thesis. Refer to [24] and [11] for details on the electro-mechanical design.

Joint No.	Description	M_s (kg)		r_k (m)		$\theta_{wc}(^\circ)$	
		Moving link	Moving body	Moving link	Moving body	Moving link	Moving body
1,2,3,4	Ankle roll and pitch motors	1.167	37.3	0.07	0.618	90	7
5,6	knees	4.348	-	0.172	-	90	-
7,8	leg twist	7.004	30.9	0.07	0.15	-	-
9,10	leg pitch	8.425	29.5	0.21	0.2	90	15
11,12	leg side to side	9.935	28.5	0.25	0.2	90	15
13,14	torso pitch and side to side	-	14.24	-	0.25	-	20
15	torso twist	-	11.4	-	0	-	-

Table 3.1: Static mass on each joint, radius of gyration and worst case angle.

Chapter 4

SIMULINK Model

4.1 Introduction

This chapter will discuss the local control model for each joint in the lower body and torso. The model treats the load on every joint as a static point mass acting at the estimated radius of gyration (r_k) from the joint at the worst case angle (see table 3.1), which is defined as the angle at which the load is maximum for every joint. The worst case angle arises from the need to include gravity in the model and will not be considered for joints that move mass in a plane that is parallel to the ground. This chapter will start talking about the actuator dynamics which will lead to the design of the plant model for each joint. A discussion will be presented on the offset due to gravity. This will be followed by the control model for each joint, the PI control action and the feedforward model. Finally, an explanation of how to obtain the duty cycle from the model will be presented.

4.2 Plant—Actuator Dynamics

The permanent magnet DC motor in figure 4.1 (or PMDC motor) is based on 2 equations [**?**],

$$\tau_m = K_t i_a \tag{4.1}$$

where:

- τ_m = motor torque (Nm).
- K_t = motor torque constant (Nm/A)
- i_a = Armature current (A)

As the motor spins, it generates a back EMF voltage proportional to the motor speed,

$$V_b = K_b\dot{\theta}_m, \tag{4.2}$$

where K_b is the back EMF constant. By performing KVL for the circuit in figure 4.1,

$$L_a\frac{di_a}{dt} + R_a i_a = E_a - V_b. \tag{4.3}$$

If we assume that the armature inductance, L_a is much smaller than the armature resistance (which is typical of a DC motor),

$$i_a = \frac{E_a - K_b\dot{\theta}_m}{R_a} \tag{4.4}$$

and when this is substituted into equation 4.1,

$$\tau_m = \frac{K_t}{R_a}\left(E_a - K_b\dot{\theta}_m\right) \tag{4.5}$$

$$from\,which \qquad E_a = \frac{R_a}{K_t}\tau_m + K_b\dot{\theta}_m \tag{4.6}$$

When the motor is connected in series with a gear-box and load (a robotic link) as in figure 4.2,

we can write the equation of motion for the motor as,

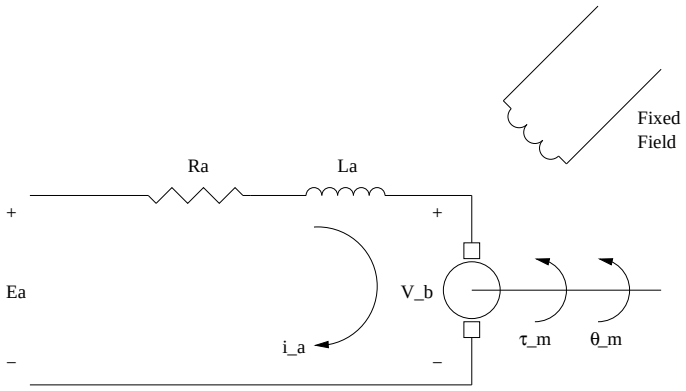


Figure 4.1: DC motor schematic.

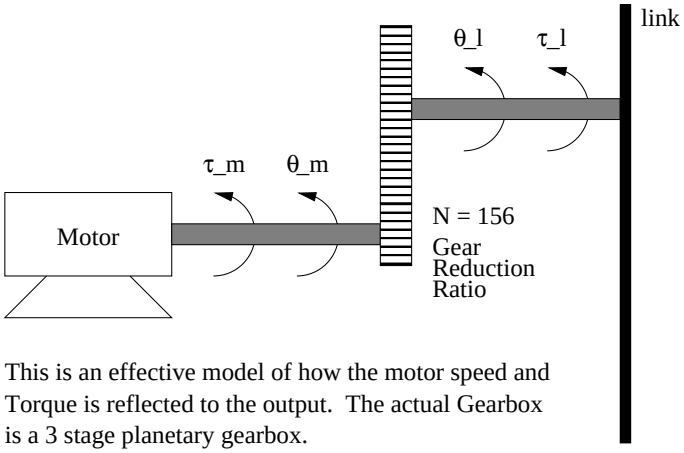


Figure 4.2: Gear-box and load on motor.

$$J\ddot{\theta}_m + D\dot{\theta}_m = \tau_m - \frac{\tau_l}{N} \tag{4.7}$$

$$J\ddot{\theta}_m + D\dot{\theta}_m = K_t i_a - \frac{\tau_l}{N} \tag{4.8}$$

where J is the combined inertia of the armature and the load reflected to the armature and D is the combined viscous damping of the armature and load reflected to the armature,

$$J = J_a + J_{load}\left(\frac{1}{N}\right)^2 \quad ; \quad D = D_a + D_{load}\left(\frac{1}{N}\right)^2. \tag{4.9}$$

Re-writing equation 4.3 and 4.7 in the Laplace domain,

$$(L_a s + R_a) I_a(s) = E_a(s) - K_b s\theta_m(s) \quad (4.10)$$

$$\left(Js^2 + Ds\right)\theta_m(s) = K_t I_a(s) - \frac{T_l(s)}{N}. \quad (4.11)$$

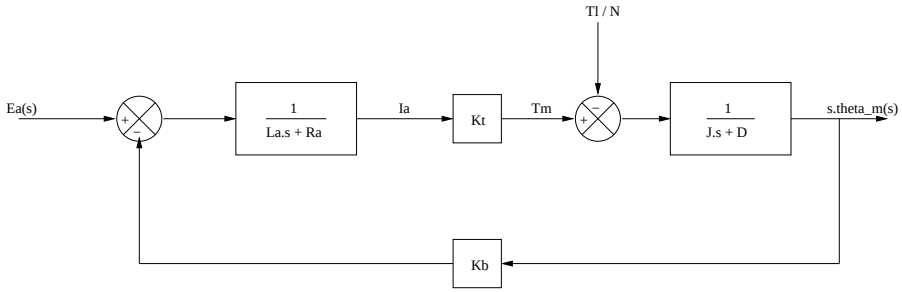


Figure 4.3: Plant model.

The transfer function in figure 4.3 between the input voltage, $E_a(s)$ and the motor speed $s\theta_m(s)$ is,

$$\frac{s\theta_m(s)}{E_a(s)} = \frac{K_t}{(L_a s + R_a)(Js + D) + K_b K_t} = \frac{K_t}{JL_a s^2 + (DL_a + R_a J)s + (R_a D + K_b K_t)} \quad (4.12)$$

and the transfer function between the offset torque, T_l/N, and the motor speed for a reference input of zero (or $E_a(s) = 0$)

$$\frac{s\theta_m(s)}{T_l(s)} = -\frac{(L_a s + R_a)}{N\left[(L_a s + R_a)(Js + D) + K_b K_t\right]} \quad (4.13)$$

The derivation for this transfer function can be found in appendix B.

Appendix D shows the SIMULINK model that is used to model the plant for every joint. The only difference between each model is the number used for J, the mass on each joint in the offset block, the radius of gyration in the offset block and the worst case angle in the offset block.

4.2.1 Offset due to Gravity

The above model of the plant (figure 4.3) is divided into two transfer functions: voltage to torque and torque to current. This is due to the offset from gravity, which will be in the form of a torque that must be reflected to the armature side and subtracted from the motor torque before evaluating the motor speed.

To illustrate this mathematically, consider figure 4.4 below which shows a point mass swinging at the radius of gyration, r_k from a pivoting point.

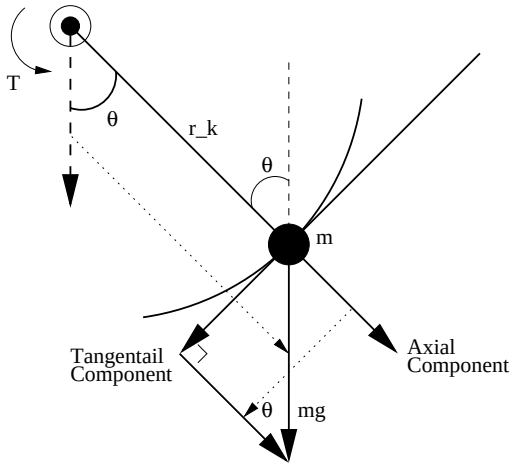


Figure 4.4: Swinging point mass.

This could be a simple model of a human leg. If we write the torque differential equation for the system, we get

$$T(t) = J\frac{d^2\theta}{dt^2} + D\frac{d\theta}{dt} + mgr_k \sin\theta. \tag{4.14}$$

The presence of the term $\sin\theta$ makes equation 4.14 non-linear. It is possible to linearise the above equation about the worst case angle, θ_{worst_case} or θ_{wc}, using a Taylor series expansion, which for a function $f(\theta)$ is

$$f(\theta) = f(\theta_{wc}) + \frac{df}{d\theta}|_{\theta=\theta_{wc}} \cdot \frac{(\theta-\theta_{wc})}{1!} + \frac{d^2f}{d\theta^2}|_{\theta=\theta_{wc}} \cdot \frac{(\theta-\theta_{wc})^2}{2!} + \ldots \tag{4.15}$$

By considering small perturbations of θ about θ_{wc}, it is possible to simplify equation 4.15 by considering only the first two terms on the *RHS* and this will linearise the change in

$f(\theta)$ from θ_{wc}. Considering only the first two terms in equation 4.15, we get

$$f(\theta) - f(\theta_{wc}) = \frac{df}{d\theta}|_{\theta=\theta_{wc}} \cdot (\theta - \theta_{wc}) \tag{4.16}$$

$$\delta f(\theta) = k|_{\theta=\theta_{wc}} \delta\theta. \tag{4.17}$$

This is a linear relationship where k is the slope at the worst case angle.

If we use equation 4.16 on the non-linear term in equation 4.14, where $f(\theta) = \sin\theta$ and $f(\theta_{wc}) = \sin\theta_{wc}$, we get

$$f(\theta) - f(\theta_{wc}) = \frac{df}{d\theta}|_{\theta=\theta_{wc}} \cdot (\theta - \theta_{wc}) \tag{4.18}$$

$$\sin\theta - \sin\theta_{wc} = \cos\theta_{wc} \cdot (\theta - \theta_{wc}) \tag{4.19}$$

$$or \quad \delta f(\theta) = \cos\theta_{wc}\delta\theta \tag{4.20}$$

$$and \quad \sin\theta = \sin\theta_{wc} + \cos\theta_{wc}\delta\theta \tag{4.21}$$

Also, $\frac{d^2\theta}{dt^2} = \frac{d^2\delta\theta}{dt^2}$and $\frac{d\theta}{dt} = \frac{d\delta\theta}{dt}$. If we substitute equation 4.21 into equation 4.14, then

$$T(t) = J\frac{d^2\delta\theta}{dt^2} + D\frac{d\delta\theta}{dt} + mgr_k(\sin\theta_{wc} + \cos\theta_{wc}\delta\theta) \tag{4.22}$$

which in the Laplace domain is

$$T(s) = Js^2\delta\theta + Ds\delta\theta + mgr_k\sin\theta_{wc} + mgr_k\cos\theta_{wc}\delta\theta. \tag{4.23}$$

By substituting equation 4.23 into equation 4.6, we get

$$\frac{R_aJ}{K_t}s^2\delta\theta_m + \frac{R_aD}{K_t}s\delta\theta_m + \frac{R_a}{K_t}\underline{mgr_k\sin\theta_{wc}} + \frac{R_a}{K_t}mgr_k\cos\theta_{wc}\delta\theta_m + K_bs\delta\theta_m = E_a(s) \tag{4.24}$$

The underlined term in equation 4.24 is a constant which in not in terms of $\delta\theta_m$ and represents the offset due to gravity. It is for this reason that the model in figure 4.3 is chosen to represent the plant for each joint.

4.2.2 First Order Approximation

The transfer function of equation 4.12 is a second order, type zero system. It includes both the electrical pole and the mechanical pole. The electrical pole is always much faster than the mechanical pole. Or in other words, the time constant of the electrical pole is much smaller than the time constant of the mechanical pole. The open loop electrical pole will lie much farther to the left of the negative real axis in the s-plane, and the transfer function in equation 4.12 can be simplified to a first order system, where only the dominant mechanical pole exists.

$$assume\, electrical\, time\, constant\, \frac{L_a}{R_a} \ll mechanical\, time\, constant\, \frac{J}{D} \tag{4.25}$$

$$then \quad \frac{s\theta_m(s)}{E_a(s)} = \frac{\frac{K_t}{R_a}}{Js + D + \frac{K_b K_t}{R_a}} \tag{4.26}$$

$$= \frac{\frac{K_t}{JR_a}}{s + \frac{1}{J}\left(D + \frac{K_b K_t}{R_a}\right)} \tag{4.27}$$

In the time domain, the differential equation will be,

$$J\ddot{\theta}_m + \left(D + \frac{K_b K_t}{R_a}\right)\dot{\theta}_m = \frac{K_t}{R_a}E_a(t) - \frac{\tau_l}{N} \tag{4.28}$$

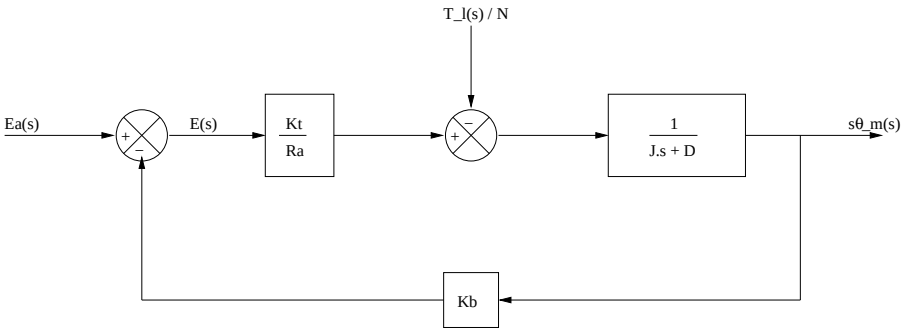


Figure 4.5: First order Approximation

4.3 Control Model

The plant model discussed above will be responsible for moving a link in the lower body and waist depending on what joint it is (foot, knee etc.). It is important to make sure that each joint reaches its desired position based on the velocity commands sent to it from the central controller. This is done using a PI compensator with feedforward cancellation as follows,

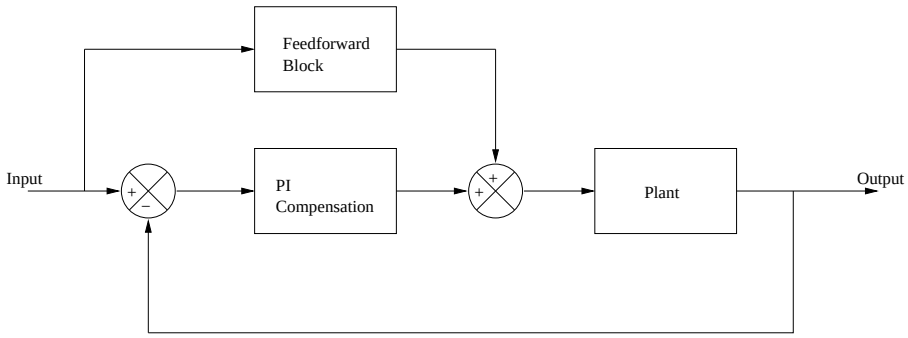


Figure 4.6: Local control model for each joint.

4.3.1 PI Compensator

The PI compensator was discussed in chapter 2. It is used to improve or eliminate steady state error by adding a pole at the origin and thus increasing the system type. The output of a PI compensator is the summation of the error and the integral of the error as follows,

$$u(t) = K_p e(t) + \frac{K_p}{T}\int_0^t e(t)dt \tag{4.29}$$

where K_p is the proportional gain and $\frac{K_p}{T}$ is the integral gain. In the Laplace domain equation 4.29 becomes

$$U(s) = K_p E(s) + \frac{K_p}{T}\frac{E(s)}{s} \tag{4.30}$$

from which

$$\frac{U(s)}{E(s)} = K_p\left(1+\frac{1}{Ts}\right) = K_p\left(\frac{Ts+1}{Ts}\right) = K_p\left(\frac{s+\frac{1}{T}}{s}\right) \tag{4.31}$$

which is the transfer function in chapter 2. By considering the first order approximation of the plant, it is possible to use a root locus technique to position the zero of the PI compensator sufficiently close to the PI pole at the origin so that the order of the system is increased while the root locus will remain unaffected (see figure 4.7).

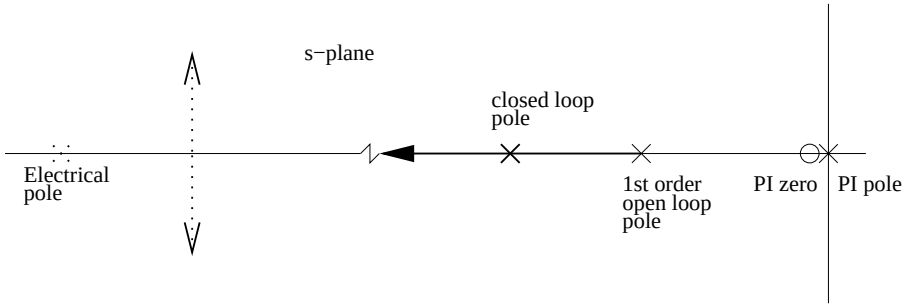


Figure 4.7: Root locus diagram.

Since the zero is positioned at $-\frac{1}{T}$, this will produce the value of T for each joint. The first order mechanical pole will move to the left of the negative real axis for different values of K_p. K_p will be the length from the first order open loop pole to the closed loop pole, or

$$K_p = \frac{\Pi\, pole\, length}{\Pi\, zero\, length}. \tag{4.32}$$

According to the first order approximation, K_p can move the first order pole to infinity, but due to the electrical pole (which is considered in figure 4.3), the maximum value that K_p can take in order for the response of the system to still be over-damped is the length from the first order pole to the breakaway point between the electrical and mechanical poles. Beyond that, the closed loop poles enter the complex plane and the response becomes under-damped with overshoot.

Having obtained the value for K_p and T, the integral gain will be $\frac{K_p}{T}$ and the PI compensator is complete.

4.3.2 Feedforward Path

The feedforward path models the inverse of the plant. The plant from equation 4.12 has the form

$$\frac{s\theta_m(s)}{E_a(s)} = \frac{D}{As^2+Bs+C} \tag{4.33}$$

where $D = K_t$, $A = JL_a$, $B = DL_a + R_aJ$ and $C = R_aD + K_bK_t$. The inverse of this will be of higher order in the numerator than in the denominator and unfortunately, SIMULINK can not handle non-causal systems. It is possible however, to use derivative blocks and create a sub-model of the feedforward path as follows. If

$$\frac{Y(s)}{X(s)} = \frac{As^2+Bs+C}{D},$$

where $X(s)$ is the input to the feedforward block and $Y(s)$ is the output of the feedforward block, then

$$DY(s) = X(s)\left[As^2+Bs+C\right] \tag{4.34}$$

$$DY(s) = As^2X(s) + BsX(s) + CX(s) \tag{4.35}$$

$$Y(s) = \frac{A}{D}s^2X(s) + \frac{B}{D}sX(s) + \frac{C}{D}X(s) \tag{4.36}$$

which in SIMULINK can be realised as per figure 4.8.

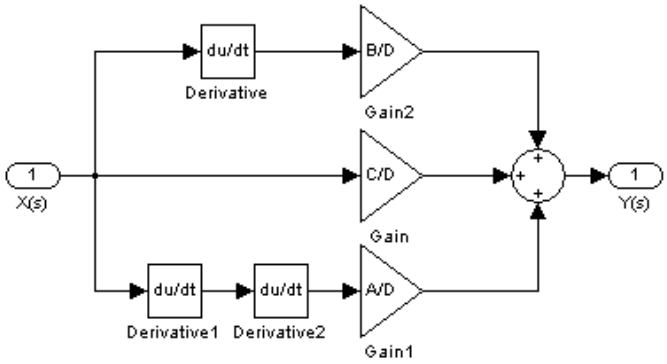


Figure 4.8: Feedforward block.

Appendix D shows the SIMULINK control model that is used for every joint. The number in the PI compensator will be different for each joint. Also, for different joints, the values of A and B will be different in the feedforward path.

4.3.3 Input Saturation

The input into the plant is limited to 32 volts because this is the nominal input voltage into the motor (plant) from the data-sheet in appendix E.

4.4 Duty Cycle

The duty cycle determines what percentage of the PWM period will deliver input into the plant, or $\frac{t_{on}}{\tau}$ from figure 4.9. The remainder of the period will deliver zero input. Therefore, by adjusting the duty cycle, the average input per period can be controlled. In the SIMULINK model, the voltage into the plant is divided by the nominal input into the motor to obtain the duty cycle. Appendix C shows how the duty cycle is obtained from the plant.

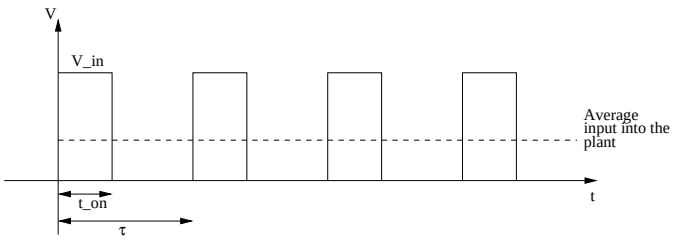


Figure 4.9: Duty cycle.

Chapter 5

DYNAMECHS Simulator

5.1 Introduction

This chapter will discuss the motor control software for each joint in the DYNAMECHS simulator. The central controller (see figure 1.5) will send velocity commands to the five DC joint controller boards [22] via the CAN bus. there will be two interrupt routines on these boards. One interrupt happens every two milliseconds upon the arrival of joint trajectory from the central controller into the CAN mailbox [2]. There is also an interrupt which occurs every 500 microseconds upon the arrival of an actual encoder count reading of a joint to perform the local control routine. The control routine will then compare the desired velocity and the encoder readings to generate a PWM value based on PI compensation to make sure that the link gets to a desired position. For the RC servo-motors located on one controller board [4], there is only one interrupt routine upon the arrival of a CAN packet. This is a position command from the central controller. The PWM duty cycle controls the position of the RC servo-motors, and the control law is performed internally within the motor package. The data sheet for the RC servo-motors in not available and can not be included in appendix E. For more information on the DYNAMECHS software structure, refer to [21].

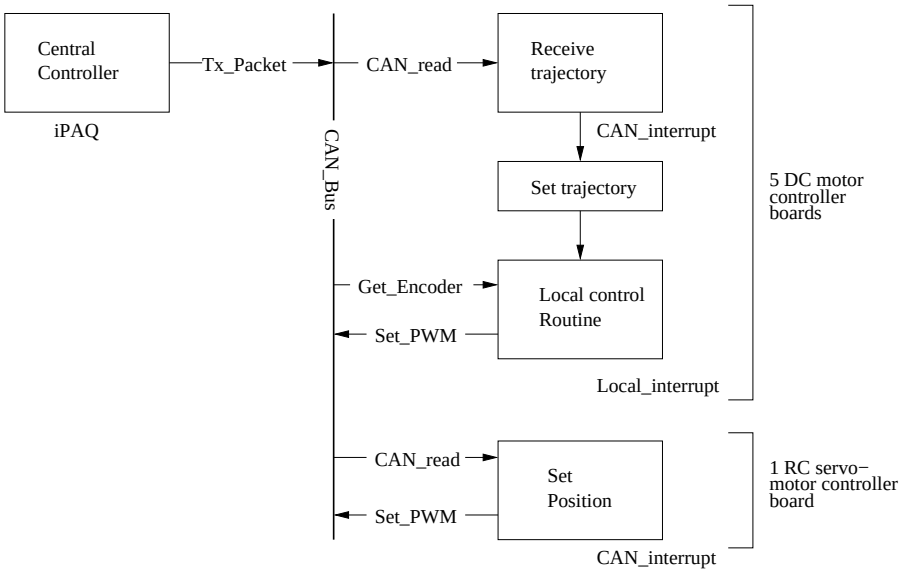


Figure 5.1: Software Structure.

5.2 Control Model

The control model was introduced in chapter 4. Figure 4.6 shows the control model that is implemented in the control software for every joint in the lower body and torso. The software is required to return the same results as the model in figure 4.6.

Appendix D contains the software for the PI control and the proportional control routine for the RC servo-motors.

The flow chart in figure 5.2 shows the control loop diagrammatically. The calculations that the control loop performs for the DC motors and RC servo-motors are in the following section. These calculations are used to obtain the proportional and integral errors in each of the high power joints (DC motors) and proportional errors for the RC servo-motors (because of the lack of data on the RC servo-motors, it was decided to perform proportional position feedback control on the RC servo-motors). From these errors, the system determines the PWM duty cycle to input to each plant.

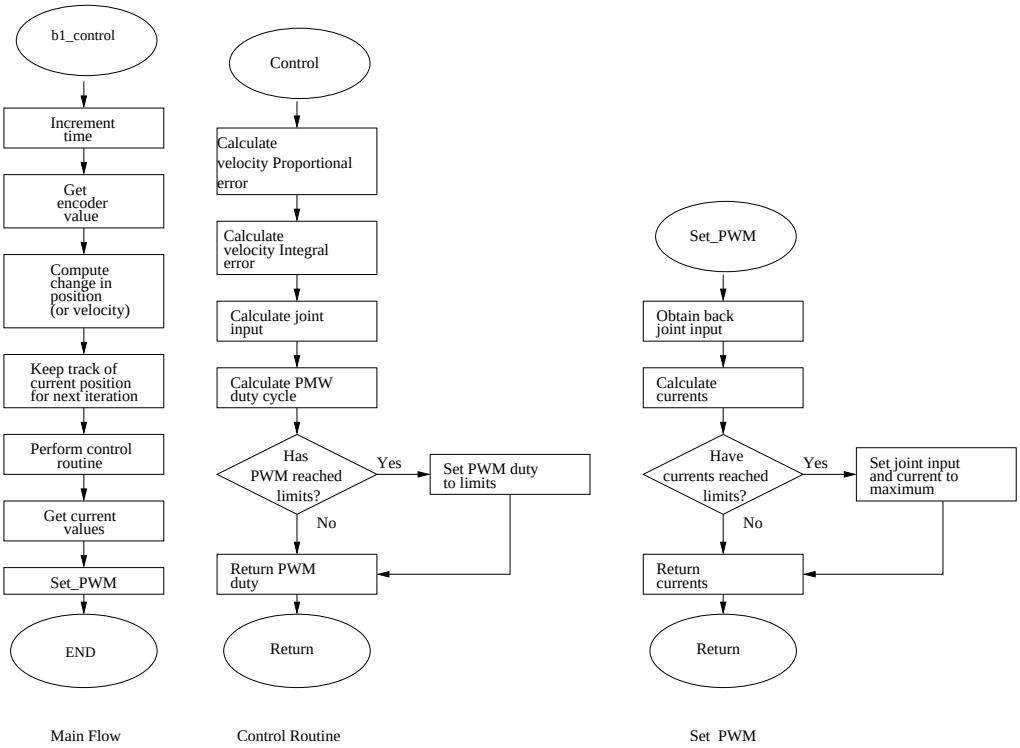


Figure 5.2: Control loop for DC motors.

5.3 Calculations in Software

The calculations used in software in appendix D are now looked at more closely. The velocity proportional errors for each joint is calculated as follows

$$Proportional\,error = desired\ joint\,velocity - joint\,velocity$$

The velocity integral error for each joint is given by

$$Integral\,error = \sum Proportional\,error \times step\,time$$

The joint input to each joint is determined by equation 5.1 [6],

$$(Prop.error + (Int.error \times Int\,gain) \times Prop.gain + (desired\ joint\,velocity \times Feed\,fwd\,gain) \tag{5.1}$$

The PWM duty value can then be calculated as follows

$$PWM\,duty = joint\,input / MOTOR\,VOLTAGE$$

where MOTOR VOLTAGE is set to 40 volts in software.

Current is calculated using the the following equation,

$$\frac{joint\,input - back\,EMF}{Armature\,resistance}.$$

For the RC servo-motors, the joint input is calculated using proportional compensation law as follows,

$$joint\,input = SERVO\_P \times (desired\ joint\,pos - joint\,pos).$$

Chapter 6

Testing, Results and Discussion

This chapter will discuss what testing was performed with the SIMULINK model for each joint and with the DYNAMECHS simulator. Three different methods will be presented, each of which differ in where the zero of the PI compensator is positioned on the s-plane. The first method looks at positioning the PI zero very close to the PI pole at the origin. The second method looks at positioning the zero further away from the PI pole at the origin but still far to the right of the mechanical pole. The third method investigates positioning the PI zero to the left of the mechanical pole. The motor characteristics used to model the plant are in appendix E. For all joints and methods, the feedforward gain is always set to 1.

6.1 Method 1—PI Zero very close to PI pole

This method involves positioning the PI compensator pole at the origin and selecting a PI zero very close to the PI pole at -0.0002. Because the zero is so close to the pole, this in effect will perform proportional compensation. The value used for K_p is the maximum value to still obtain an over-damped response (discussed in section 4.3.1). A MATLAB script was written to evaluate the inertia due to the load on each joint at the radius of gyration reflected to the armature (the mass on each joint and the radius of gyration is

in table A.1 and 3.1 respectively), the total inertia J on each joint, the transfer function for each joint (equation 4.12), the first order approximation for each joint (equation 4.27) and the breakaway point between the electrical and mechanical poles, given the mass on each joint and the radius of gyration.

Algorithm 1 MATLAB Script.

```
function output = PI_constants(m, r)
%

L = 0.3e-3;
N = 156;
Ja = 6.52e-6;
D = 0.0000157;
R = 1.71;
Kt = 0.0445;
Kb = 0.0444;

% The following works out the effective inertia on a motor
Jload = (m * (r)^2)/N^2
J = Ja + Jload

num = Kt
den = [L*J (L*D + R*J) (R*D + Kt*Kb)];
g = tf(num, den)
gzpk = zpk(g)
rlocus(g)            % to get the breakaway point.

num1 = Kt/R;
den1 = [J (D + Kt*Kb/R)];
g1 = tf(num1, den1)
g1zpk = zpk(g1)
```

As an example, consider the ankle roll and pitch joints. When these joints move the entire body (37.3 kg) by 7°(worst case angle) at the radius of gyration (0.618 m), the script in algorithm 1 returns $5.85 \times 10^{-4} kgm^2$ for the inertia due to the load on the ankle joints ($\frac{37.3 \times 0.618^2}{156^2}$), the total inertia is $5.9 \times 10^{-4} kgm^2$ ($5.85 \times 10^{-4} kgm^2 + 6.52 \times 10^{-6} kgm^2$), the transfer function is $\frac{0.0445}{1.77 \times 10^{-7} s^2 + 0.001 s + 0.002}$ (using equation 4.12 and the appropriate values), the first order approximation is $\frac{43.96}{s + 1.979}$ (using equation 4.27 and the appropriate

values), the mechanical pole is at -1.979 and the electrical pole is at -5698 (and it can be seen that the pole in the first order approximation is where the mechanical pole is located). The breakaway point between the electrical and mechanical poles will be at -2850. From this information, it is possible to obtain the maximum value of K_p for an over-damped response, which will be 2848 (or $2850-1.979$) and since the zero is at -0.0002 (which is $-\frac{1}{T}$), T will equal 5000 and K_I will be 0.57 ($=\frac{2848}{5000}$). The compensator is therefore $2848\left(\frac{s+0.0002}{s}\right)$.

When the ankle roll and pitch joints move just the foot (1.167 kg) by 90°(worst case angle) at the radius of gyration (0.07 m), the script in algorithm 1 returns $2.3\times10^{-7}\,kgm^2$ for the inertia due to the load on the ankle joints ($\frac{1.167\times0.07^2}{156^2}$), the total inertia is $6.8\times10^{-6}\,kgm^2$ ($2.3\times10^{-7}\,kgm^2+6.52\times10^{-6}\,kgm^2$), the transfer function is $\frac{0.0445}{2.03\times10^{-7}s^2+1.2\times10^{-5}s+0.002}$ (using equation 4.12 and the appropriate values), the first order approximation is $\frac{3852.5}{s+173.4}$ (using equation 4.27 and the appropriate values), the mechanical pole is at -178.9 and the electrical pole is at -5523. The breakaway point between the electrical and mechanical poles will be at -2851. From this information, it is possible to obtain the maximum value of K_p for an over-damped response, which will be 2677.6 (or $2851-173.4$) and since the zero is at -0.0002 (which is $-\frac{1}{T}$), T will equal 5000 and K_I will be 0.54 ($\frac{2677.6}{5000}$). The compensator is therefore $2677.6\left(\frac{s+0.0002}{s}\right)$.

For these joints, moving the upper body (37.3 kg) will be a more severe movement than just moving the foot, so the PI compensator for moving the upper body is considered. The three hip joints on each leg responsible for the roll, pitch and sideways motion of the leg will also move either the leg in the respective orientation or the upper body. For the hip joint responsible for twisting the leg, the more severe movement is moving the upper body, for the hip joint responsible for moving the leg backwards and forwards, the more severe movement is moving the leg to 90°(worst case angle) and for the hip joint responsible for moving the leg side to side, the more severe movement is moving the leg to 90°.

Performing the above procedure on every joint, it is possible to obtain the PI compensator for each joint. The results are summarised in table 6.1.

Joint	$J_{load}\,(kgm^2)$	$J\,(kgm^2)$	Transfer fnct.	1st order app.	Elec. pole	Mech. pole	Breakaway point	K_p	K_I
Ankle roll and pitch moving body	5.85×10^{-4}	5.9×10^{-4}	$\frac{0.0445}{1.77\times10^{-7}s^2+0.001s+0.002}$	$\frac{43.96}{s+1.979}$	-5698	-1.979	-2850	2848	0.57
Ankle roll and pitch moving foot	2.3×10^{-7}	6.8×10^{-6}	$\frac{0.0445}{2.03\times10^{-9}s^2+1.2\times10^{-5}s+0.002}$	$\frac{3852.5}{s+173.4}$	-5523	-178.9	-2851	2677.6	0.54
Knee	5.3×10^{-6}	1.2×10^{-5}	$\frac{0.0445}{3.54\times10^{-9}s^2+2.02\times10^{-5}s+0.002}$	$\frac{2201.4}{s+99.07}$	-5600	-100.8	-2850	2750.93	0.55
Right and left hip pitch moving leg	1.52×10^{-5}	2.2×10^{-5}	$\frac{0.0445}{6.54\times10^{-9}s^2+3.7\times10^{-5}s+0.002}$	$\frac{1194.44}{s+53.75}$	-5646	-54.26	-2850	2796.3	0.56
Right and left hip pitch moving body	4.8×10^{-5}	5.5×10^{-5}	$\frac{0.0445}{1.65\times10^{-8}s^2+9.41\times10^{-5}s+0.002}$	$\frac{473.1}{s+21.29}$	-5679	-21.37	-2850	2828.7	0.57
Right and left hip twist moving leg	1.41×10^{-6}	7.93×10^{-6}	$\frac{0.0445}{2.4\times10^{-9}s^2+1.4\times10^{-5}s+0.002}$	$\frac{3281.54}{s+147.7}$	-5550	-151.7	-2851	2703.3	0.54
Right and left hip twist moving body	2.85×10^{-5}	3.5×10^{-5}	$\frac{0.0445}{1.05\times10^{-9}s^2+6\times10^{-5}s+0.002}$	$\frac{741.6}{s+33.38}$	-5667	-33.57	-2850	2816.6	0.56
Right and left hip side to side moving leg	2.5×10^{-5}	3.2×10^{-5}	$\frac{0.0445}{9.6\times10^{-9}s^2+5.5\times10^{-5}s+0.002}$	$\frac{812.34}{s+36.56}$	-5664	-36.8	-2850	2813.4	0.56
Right and left hip side to side moving body	4.7×10^{-5}	5.3×10^{-5}	$\frac{0.0445}{1.6\times10^{-8}s^2+9.1\times10^{-5}s+0.002}$	$\frac{487.7}{s+21.95}$	-5678	-22.03	-2850	2828.1	0.57
Torso pitch and side to side	3.7×10^{-5}	4.3×10^{-5}	$\frac{0.0445}{1.3\times10^{-8}s^2+7.4\times10^{-5}s+0.002}$	$\frac{603.9}{s+27.18}$	-5673	-27.31	-2850	2822.82	0.56
Torso twist	0	6.52×10^{-6}	$\frac{0.0445}{1.96\times10^{-9}s^2+1.12\times10^{-5}s+0.002}$	$\frac{3991.32}{s+179.6}$	-5517	-185.6	-2851	2671.4	0.53

Table 6.1: Results for method 1. The PI zero is at -0.0002.

The PI compensator for each joint was then used in the SIMULINK model. As an example, for the ankle roll and pitch joints moving the upper body, the PI compensator was entered into the PI block (see appendix C), the feedforward block was set up with the appropriate constants, $\frac{1.77\times10^{-7}}{0.0445}$ for $\frac{A}{D}$ (see figure 4.8), $\frac{0.001}{0.0445}$ for $\frac{B}{D}$ and $\frac{0.002}{0.0445}$ for $\frac{C}{D}$. Also, the total inertia J (5.9×10^{-4}) was entered into the torque/speed block of the plant. The value for the damping D in the torque/speed block was obtained by having no load on the joint and no offset due to gravity. It was then made sure that the no load speed from appendix E was obtained at the motor shaft for the nominal input of 32 volts from the data-sheet with

no feedback. This value turned out to be $0.0000157\,\frac{Nms}{rad}$. The model was then given a step input. The result for this joint and the rest of the joints is summarised in tables 6.2 and 6.3 below.

Joint	Response	Settling time (T_s, sec.)
Ankle roll and pitch moving body	overdamped	1.75 (motor goes backwards)
Ankle roll and pitch moving foot	overdamped	0.01 (motor goes forwards with a steady state error of 0.9)
Knee	overdamped	0.05 (motor goes backwards)
Right and left hip pitch moving leg	overdamped	0.1 (motor goes backwards)
Right and left hip pitch moving body	overdamped	0.2 (motor goes backwards)
Right and left hip twist moving leg	overdamped	0.02 (motor goes forwards with a steady state error of 0.88)
Right and left hip twist moving body	overdamped	0.1 (motor goes forwards with a steady state error of 0.875)
Right and left hip side to side moving leg	overdamped	0.1 (motor goes backwards)
Right and left hip side to side moving body	overdamped	0.2 (motor goes backwards)
Torso pitch and side to side	overdamped	0.15 (motor goes backwards)
Torso twist	overdamped	0.02 (motor goes forwards with a steady state error of 0.875)

Table 6.2: Uncompensated response.

The same PI constants in table 6.1 were then used in the DYNAMECHS simulator for crouching, balancing on the right foot and leaning forward (see figure 1.3). Appendix D

Joint	Response	Settling time (T_s, sec.)	%OS	s.s.e	Final voltage (V)	Duty cycle
Ankle roll and pitch moving body	overdamped	0.16	-	0.005	13.8	0.431
Ankle roll and pitch moving foot	underdamped	0.003	2.5	0.002	7.2	0.217
Knee	underdamped	0.005	1.2	0.005	9	0.275
Right and left hip pitch moving leg	underdamped	0.007	0.5	0.004	11.2	0.35
Right and left hip pitch moving body	underdamped	0.02	0.4	0.004	10.7	0.33
Right and left hip twist moving leg	underdamped	0.003	2.4	0.002	7	0.23
Right and left hip twist moving body	underdamped	0.01	0.6	0.002	7	0.22
Right and left hip side to side moving leg	underdamped	0.01	0.7	0.005	13	0.41
Right and left hip side to side moving body	underdamped	0.015	0.4	0.004	10.5	0.33
Torso pitch and side to side	underdamped	0.012	0.53	0.004	10.1	0.312
Torso twist	underdamped	0.003	2.9	0.003	7	0.22

Table 6.3: Method 1 PI compensator.

contains the central controller code that generates all the movements that the robot will be tested on. The movements are periodic with a 5 second period. For crouching, it will

take 2.5 seconds to crouch down and 2.5 seconds to go back up again. For balancing on the right foot, it will take 2.5 seconds to lean to the side on the right foot and 2.5 seconds to get back to an upright position. For leaning forward, it will take 2.5 seconds to lean forward and 2.5 seconds get back to an upright position. Firstly, the uncompensated response was obtained. This was done by setting the proportional gains to 1 and setting the feedforward and integral gains to zero. When the simulation was set to crouch, balance on the right foot or lean forward, the model could not perform any of the movements and always seemed to fall down on its back, as shown in figure 6.1.

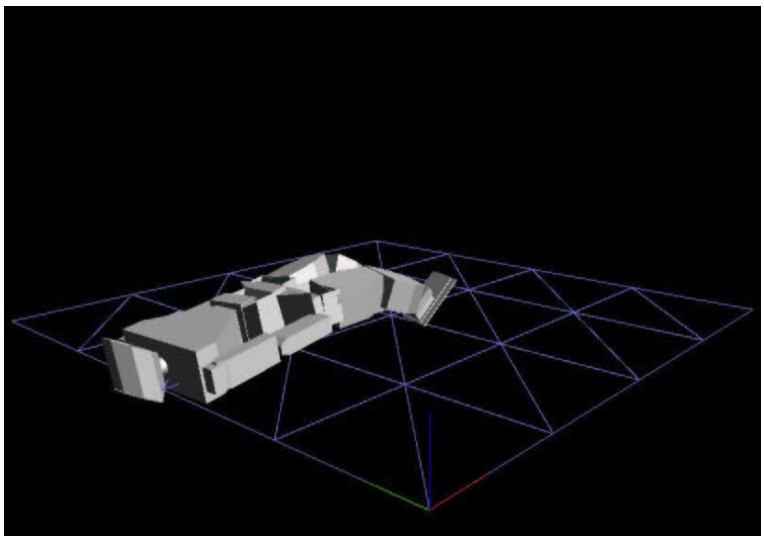


Figure 6.1: Performance of *GuRoo* with no compensation.

The reason that the model always fell on its back is due to the ankle pitch joints, knee joints and hip pitch joints, which made it easier for the robot fall backwards when there was no compensation.

For crouching, it is desirable to see information on the hip pitch, knee and ankle pitch joints which will be similar for both legs, so only the right leg information is presented. The results are shown in figure 6.2 and 6.3.

As can be seen from figures 6.2 and 6.3, the hip pitch (or hip fwd) joints are well within the current limits and the position error is very small (almost zero). The knee joints saturate to 4A for about 2 seconds each time the robot tries to stand up from its crouching

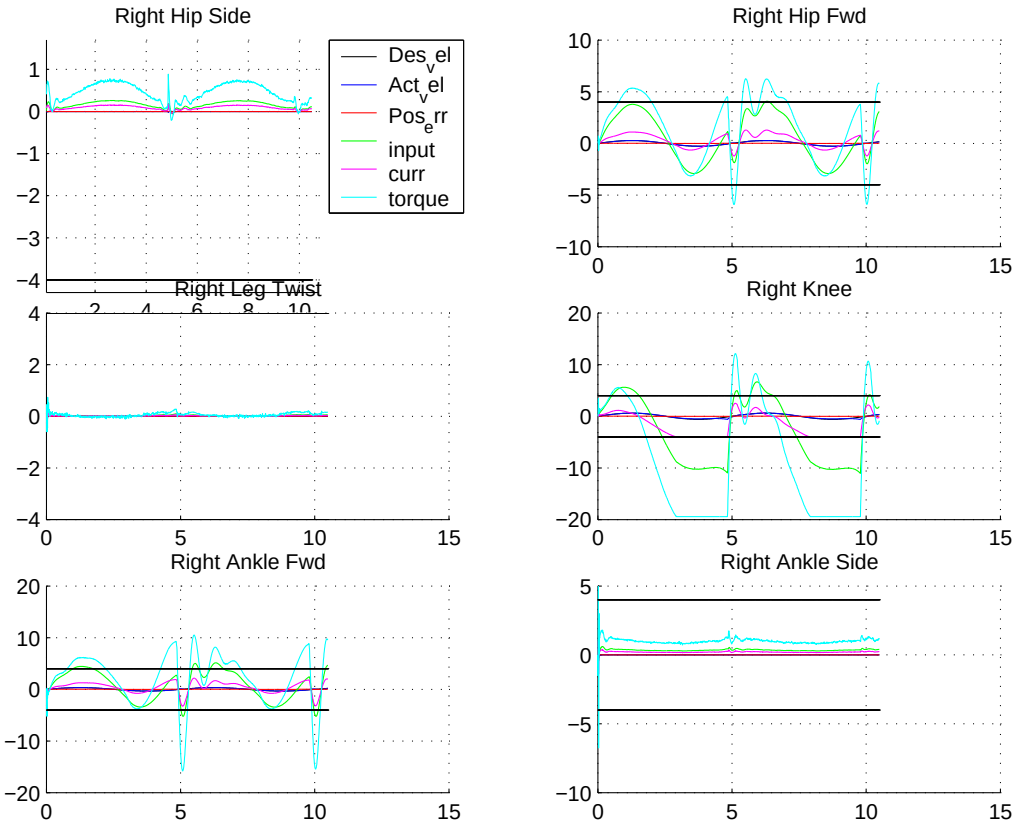


Figure 6.2: Method 1 DYNAMECHS results for crouching. Right leg data.

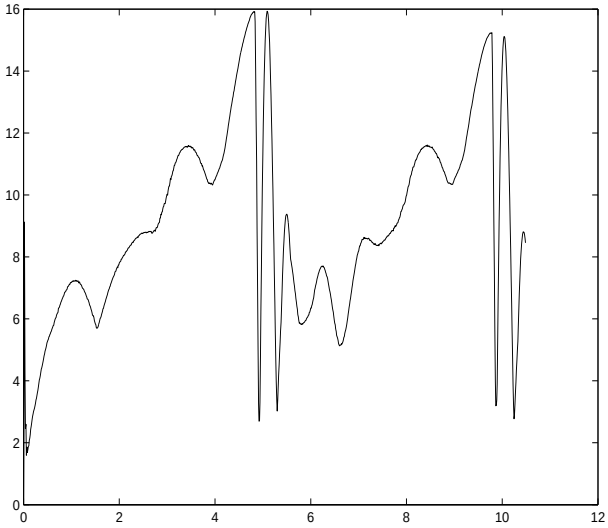


Figure 6.3: Method 1 DYNAMECHS results for crouching. Net current.

position and there is no position error. The ankle pitch (or ankle fwd) joints seem to remain within the current limits with no position error. The net currents stay below 20A

as required.

For balancing on the right foot, it is desirable to see information on the right hip side to side joint. The results are shown in figure 6.4 and 6.5.

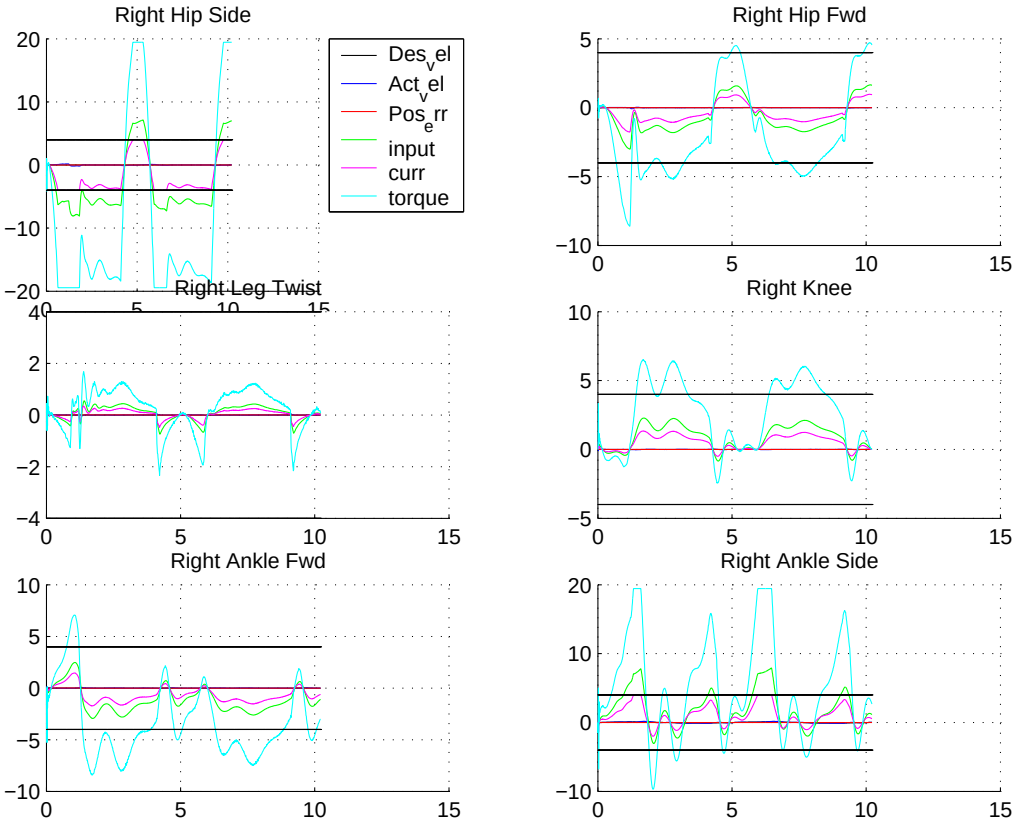


Figure 6.4: Method 1 DYNAMECHS results for balancing on the right foot. Right leg data.

As can be seen from figure 6.4, the hip side to side joint stays within the current limits with no error in position, but there is saturation for about one second as the robot starts to lean to the side. The net current exceeds 20 A for about 1.2 seconds as the robot starts to lean to the side, but is generally below 20A.

For leaning forward, it is desirable to see information on the torso pitch joint. The results are shown in figure 6.6 and 6.7.

As can be seen from figure 6.6, the torso pitch (or torso fwd) joint stays within the current limits with no error in position. The net current stays well within the 20A limit, as

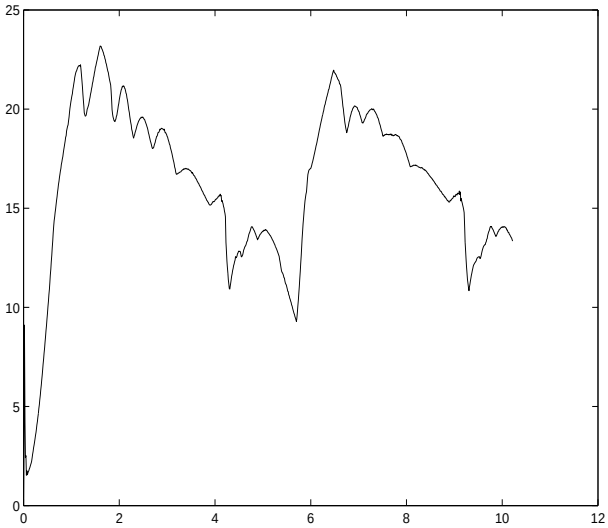


Figure 6.5: Method 1 DYNAMECHS results for balancing on the right foot. Net current.

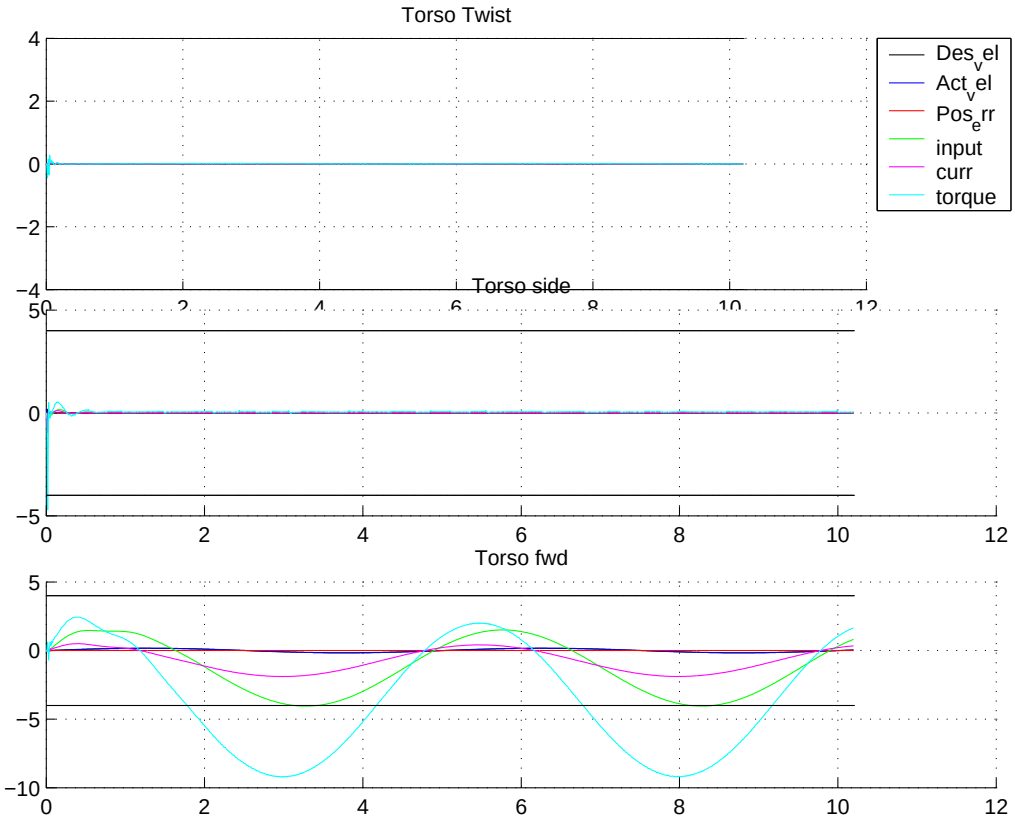


Figure 6.6: Method 1 DYNAMECHS results for leaning forward. Torso data.

required.

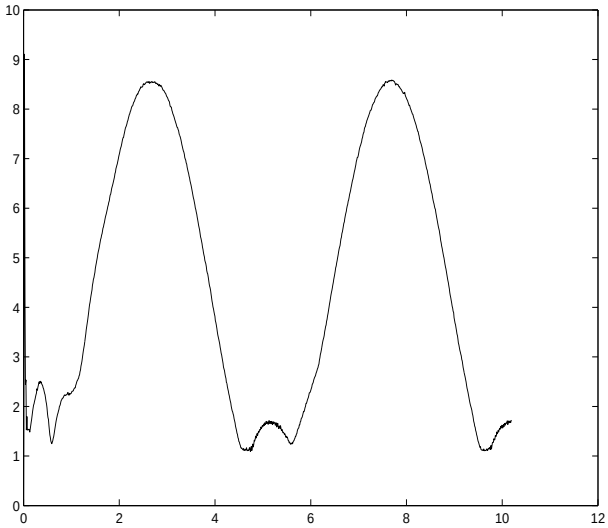


Figure 6.7: Method 1 DYNAMECHS results for leaning forward. Net currents.

6.2 Method 2—PI Zero Further away from PI pole

This method involves positioning the PI pole at the origin and selecting a PI zero further away from the pole at -2, however for the case when the ankle roll and pitch joints on each leg move the upper body (37.3 kg) to the worst case angle (7°), because the mechanical pole is at approximately -2 in the transfer function for this case, the zero is positioned at -0.2. The value used for K_p is again the maximum value to obtain an over-damped response, which is the distance between the breakaway point between the electrical and mechanical poles and the open loop mechanical pole.

As an example, consider the ankle roll and pitch joints. When these joints move the entire body (37.3 kg) by 7°(worst case angle) at the radius of gyration (0.618 m), the script in algorithm 1 returns the same values as before, but because the zero is now at -0.2 (which is $-\frac{1}{T}$), T will equal 5 and K_I will be 569.6 ($=\frac{2848}{5}$). The compensator is therefore $2848\left(\frac{s+0.2}{s}\right)$.

When the ankle roll and pitch joints move just the foot (1.167 kg) by 90°(worst case angle) at the radius of gyration (0.07 m), the script in algorithm 1 returns the same values as before, but because the zero is now at -2 (which is $-\frac{1}{T}$), T will equal 0.5 and K_I will be 5355.2 ($=\frac{2677.6}{0.5}$). The compensator is therefore $2677.6\left(\frac{s+2}{s}\right)$.

Performing this procedure on every joint, it is possible to obtain the PI compensator for each joint. The results are summarised in table 6.4.

Joint	$J_{load}\,(kgm^2)$	$J\,(kgm^2)$	Transfer fnct.	1st order app.	Elec. pole	Mech. pole	Breakaway point	K_p	K_I
Ankle roll and pitch moving body	5.85×10^{-4}	5.9×10^{-4}	$\frac{0.0445}{1.77\times10^{-7}s^2+0.001s+0.002}$	$\frac{43.96}{s+1.979}$	-5698	-1.979	-2850	2848	569.6
Ankle roll and pitch moving foot	2.3×10^{-7}	6.8×10^{-6}	$\frac{0.0445}{2.03\times10^{-9}s^2+1.2\times10^{-5}s+0.002}$	$\frac{3852.5}{s+173.4}$	-5523	-178.9	-2851	2677.6	5355.2
Knee	5.3×10^{-6}	1.2×10^{-5}	$\frac{0.0445}{3.54\times10^{-9}s^2+2.02\times10^{-5}s+0.002}$	$\frac{2201.4}{s+99.07}$	-5600	-100.8	-2850	2750.93	5501.86
Right and left hip pitch moving leg	1.52×10^{-5}	2.2×10^{-5}	$\frac{0.0445}{6.54\times10^{-9}s^2+3.7\times10^{-5}s+0.002}$	$\frac{1194.44}{s+53.75}$	-5646	-54.26	-2850	2796.3	5592.5
Right and left hip pitch moving body	4.8×10^{-5}	5.5×10^{-5}	$\frac{0.0445}{1.65\times10^{-8}s^2+9.41\times10^{-5}s+0.002}$	$\frac{473.1}{s+21.29}$	-5679	-21.37	-2850	2828.7	5657.42
Right and left hip twist moving leg	1.41×10^{-6}	7.93×10^{-6}	$\frac{0.0445}{2.4\times10^{-9}s^2+1.4\times10^{-5}s+0.002}$	$\frac{3281.54}{s+147.7}$	-5550	-151.7	-2851	2703.3	5406.6
Right and left hip twist moving body	2.85×10^{-5}	3.5×10^{-5}	$\frac{0.0445}{1.05\times10^{-9}s^2+6\times10^{-5}s+0.002}$	$\frac{741.6}{s+33.38}$	-5667	-33.57	-2850	2816.6	5633.24
Right and left hip side to side moving leg	2.5×10^{-5}	3.2×10^{-5}	$\frac{0.0445}{9.6\times10^{-9}s^2+5.5\times10^{-5}s+0.002}$	$\frac{812.34}{s+36.56}$	-5664	-36.8	-2850	2813.4	5626.9
Right and left hip side to side moving body	4.7×10^{-5}	5.3×10^{-5}	$\frac{0.0445}{1.6\times10^{-8}s^2+9.1\times10^{-5}s+0.002}$	$\frac{487.7}{s+21.95}$	-5678	-22.03	-2850	2828.1	5656.1
Torso pitch and side to side	3.7×10^{-5}	4.3×10^{-5}	$\frac{0.0445}{1.3\times10^{-8}s^2+7.4\times10^{-5}s+0.002}$	$\frac{603.9}{s+27.18}$	-5673	-27.31	-2850	2822.82	5645.4
Torso twist	0	6.52×10^{-6}	$\frac{0.0445}{1.96\times10^{-9}s^2+1.12\times10^{-5}s+0.002}$	$\frac{3991.32}{s+179.6}$	-5517	-185.6	-2851	2671.4	5342.8

Table 6.4: Results for method 2. The PI zero is at -0.2 for the ankle roll and pitch joints moving the body and -2 for all other cases.

The PI compensator for each joint was then used in the SIMULINK model. The model for each joint was set up the same way as for method 1 but with the different constants in the PI block. The model was given a step input. The results for all the joints are summarised in table 6.5. The uncompensated response is in table 6.2.

The same PI constants for method 2 were then used in the DYNAMECHS simulator for crouching, balancing on the right foot and leaning forward (see figure 1.3) as for method

Joint	Response	Settling time (T_s, sec.)	%OS	s.s.e	Final voltage (V)	Duty cycle
Ankle roll and pitch moving body	underdamped	0.165	1	0	13.8	0.432
Ankle roll and pitch moving foot	underdamped	0.5	2.5	0	7	0.22
Knee	underdamped	0.005	1.5	0	9	0.29
Right and left hip pitch moving leg	underdamped	0.5	1	0	11.2	0.36
Right and left hip pitch moving body	underdamped	1	1.3	0	10.7	0.33
Right and left hip twist moving leg	underdamped	0.6	2.4	0	7	0.22
Right and left hip twist moving body	underdamped	0.65	0.5	0	7	0.219
Right and left hip side to side moving leg	underdamped	0.012	0.55	0	13	0.4
Right and left hip side to side moving body	underdamped	0.9	0.3	0	10.6	0.332
Torso pitch and side to side	underdamped	0.6	0.45	0	10	0.31
Torso twist	underdamped	0.2	3	0	7	0.22

Table 6.5: Method 2 PI compensator.

1. For crouching, it is desirable to see information on the hip pitch, knee and ankle pitch joints which will be similar for both legs, so only the right leg information is presented. The results are shown in figure 6.8

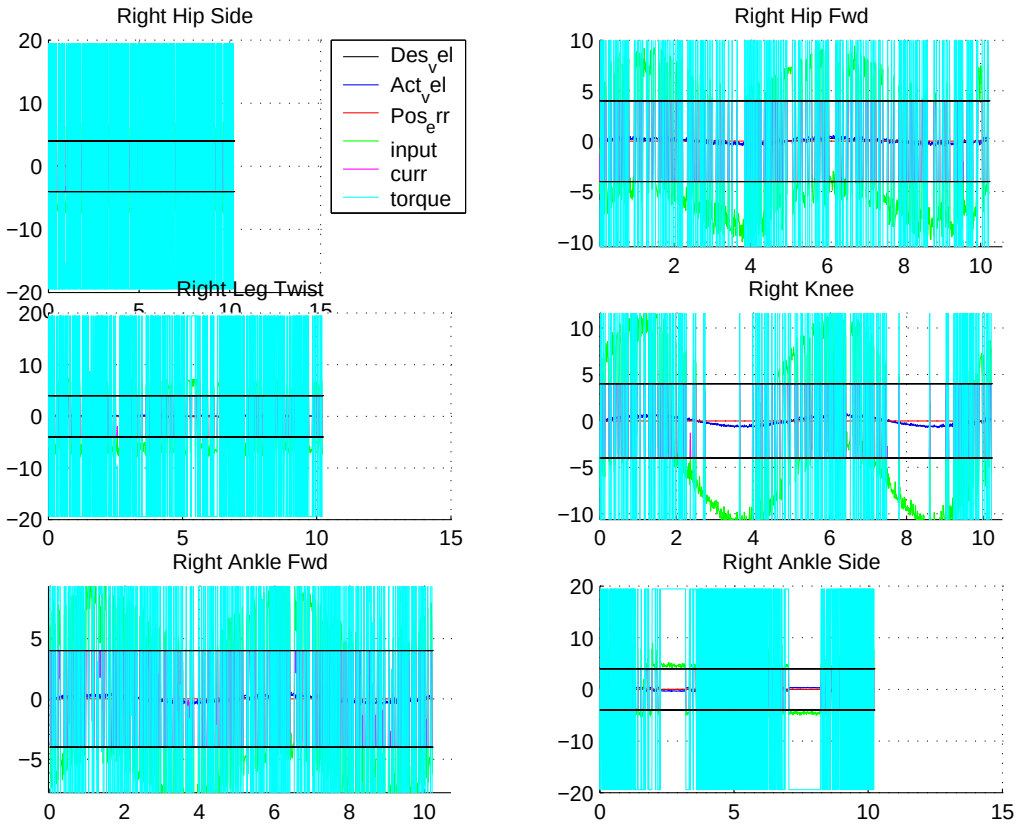


Figure 6.8: Method 2 DYNAMECHS results for crouching. Right leg data.

As can be seen from figure 6.8, the hip pitch (or hip fwd) joints are not within the current limits. The torques oscillate consistently and the current saturates to the limits, but the position error is very small (almost zero). The knee joints saturate to 4A consistently and there is no position error. The ankle pitch (or ankle fwd) joints also saturate to the current limits with no position error. The net current exceeds the 20A limit.

For balancing on the right foot, it is desirable to see information on the right hip side to side joint. The results are shown in figure 6.9.

As can be seen from figure 6.9, the hip side to side joint stays within the current limit for most of the time with no error in position, but the model is displaying saturation for small amounts of time consistently. The net current exceeds the 20A limit.

For leaning forward, it is desirable to see information on the torso pitch joint. The results are shown in figure 6.10.

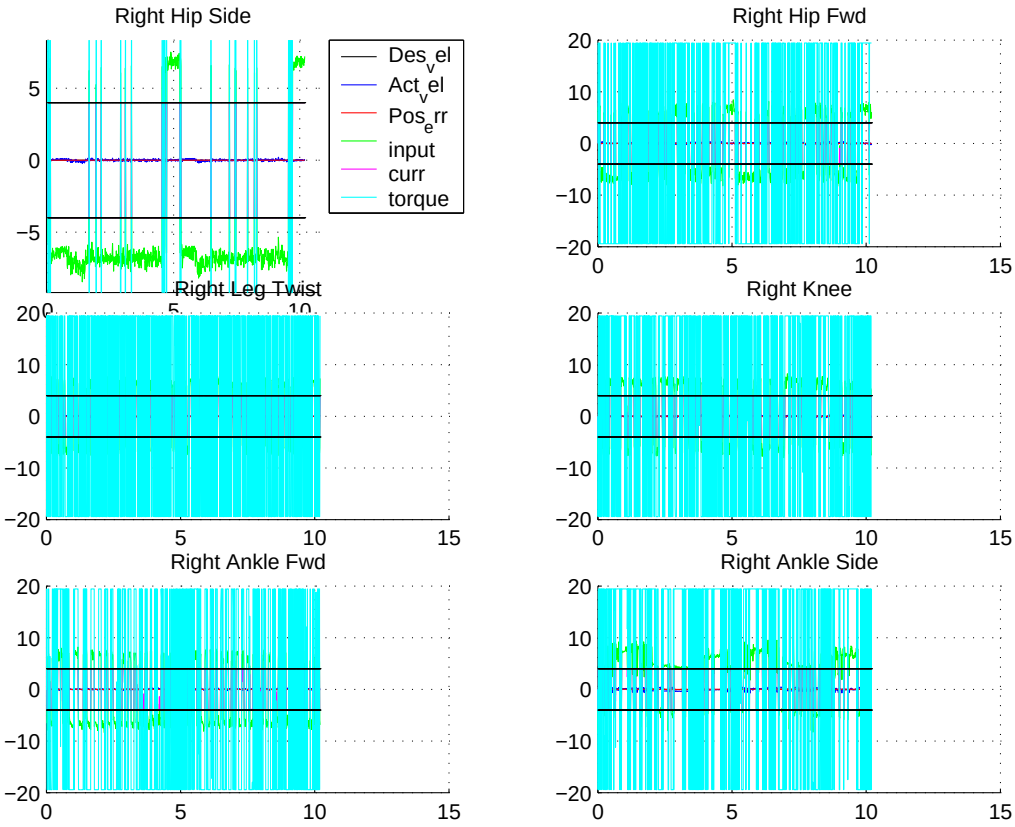


Figure 6.9: Method 2 DYNAMECHS results for balancing on the right foot. Right leg data.

As can be seen from figure 6.10, the torso pitch (or torso fwd) joint stays within the current limit with no error in position, but the current saturates to the limit consistently for short periods of time. The net current again exceeds the 20A current limit.

6.3 Method 3—Putting the PI Zero to the Left of the Mechanical Pole

The first two methods looked at obtaining an over-damped response. This method looks at obtaining the fastest transient response. This method utilises the first order approximation of the plant for each joint and involves positioning the PI pole at the origin such that the distance between the PI pole and the open loop mechanical pole is the same as the

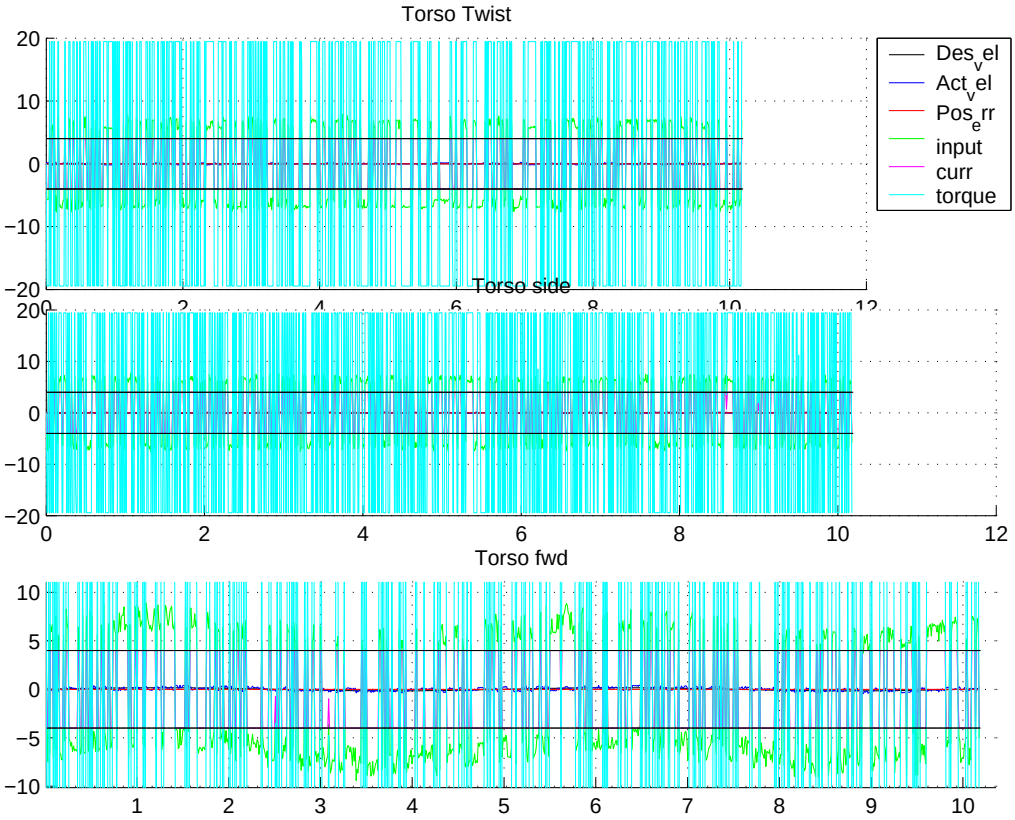


Figure 6.10: Method 2 DYNAMECHS results for leaning forward. Torso data.

distance between the open loop mechanical pole and the PI zero. The root locus for each plant will have the general form shown in figure 6.11.

It can be seen that the closed loop poles enter the complex plane. To obtain the fastest transient response, choose a radial line of 45°as in figure 6.11 which corresponds to a damping ratio, ζ, of $\frac{\sqrt{2}}{2}$ or about 0.707. This line intersects the root locus as shown in figure 6.11 and by using equation 4.32, it is possible to obtain K_p for each joint. The position of the zero is at $-\frac{1}{T}$, so T can be obtained and this together with K_p will return the integral constant $K_I = \frac{K_p}{T}$.

As an example, consider the ankle roll and pitch joints once again. When these joints move the entire body (37.3 kg) by 7°(worst case angle) at the radius of gyration (0.618 m), the script in algorithm 1 returns the same values as before (see section 6.1), but now the zero is at -3.958 (2×1.979) and the root locus is in figure 6.12a with the 45°radial

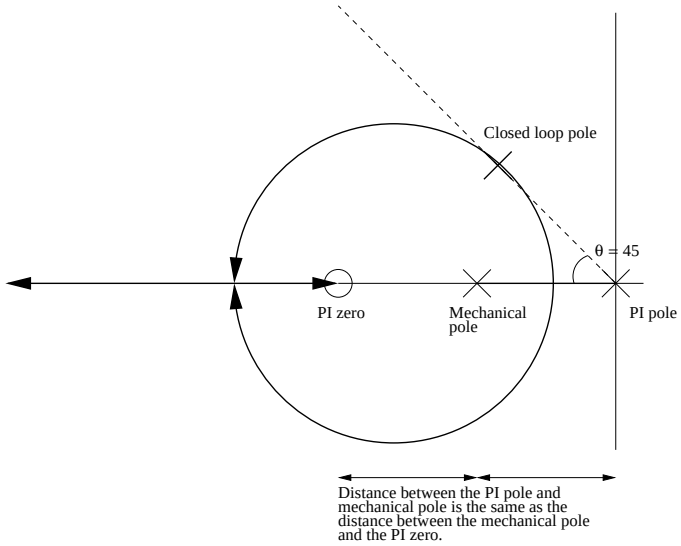
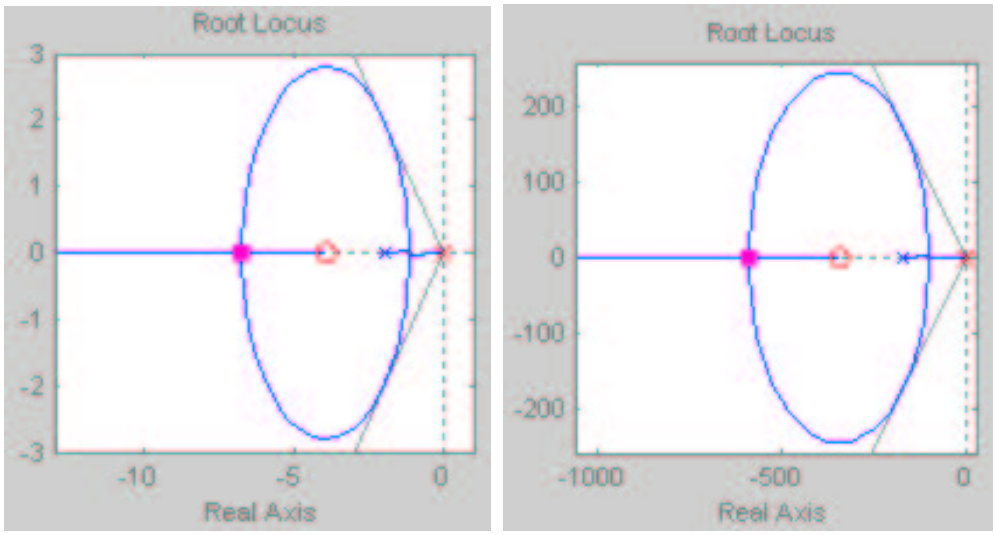


Figure 6.11: General form of root locus for method 3.

line superimposed on the s-plane.



(a) Move body. (b) Move foot.

Figure 6.12: Root locus for ankle roll and pitch joints moving the body.

Using equation 4.32, K_p is calculated to be 1.923, and since the zero is at -3.958 (which is $-\frac{1}{T}$), T will be 0.253 and K_I will be 7.6 ($= \frac{1.923}{0.253}$). The compensator is therefore $1.923\left(\frac{s+3.958}{s}\right)$.

When the ankle roll and pitch joints move just the foot (1.167 kg) by 90°(worst case angle) at the radius of gyration (0.07 m), the script in algorithm 1 returns the same values as before (see section 6.1), but now the zero is at -346.8 ($= 2 \times 173.4$) and the root locus is shown in figure 6.12. Equation 4.32 will return K_p to be 157.03, and since the zero is at -346.8 (which is $-\frac{1}{T}$), T will be 0.0029 and K_I will be 54148.04 ($= \frac{157.03}{0.0029}$). The compensator is therefore $157.03\left(\frac{s+346.8}{s}\right)$.

Performing this procedure on every joint, it is possible to obtain the PI compensator for every joint. The results are summarised in table 6.6.

Joint	$J_{load}\,(kgm^2)$	$J\,(kgm^2)$	Transfer fnct.	1st order app.	Elec. pole	Mech. pole	Breakaway point	K_p	K_I
Ankle roll and pitch moving body	5.85×10^{-4}	5.9×10^{-4}	$\frac{0.0445}{1.77\times10^{-7}s^2+0.001s+0.002}$	$\frac{43.96}{s+1.979}$	−5698	−1.979	−2850	1.923	7.6
Ankle roll and pitch moving foot	2.3×10^{-7}	6.8×10^{-6}	$\frac{0.0445}{2.03\times10^{-9}s^2+1.2\times10^{-5}s+0.002}$	$\frac{3852.5}{s+173.4}$	−5523	−178.9	−2851	157.03	54148.04
Knee	5.3×10^{-6}	1.2×10^{-5}	$\frac{0.0445}{3.54\times10^{-9}s^2+2.02\times10^{-5}s+0.002}$	$\frac{2201.4}{s+99.07}$	−5600	−100.8	−2850	98.93	19590.51
Right and left hip pitch moving leg	1.52×10^{-5}	2.2×10^{-5}	$\frac{0.0445}{6.54\times10^{-9}s^2+3.7\times10^{-5}s+0.002}$	$\frac{1194.44}{s+53.75}$	−5646	−54.26	−2850	53.3	5728.93
Right and left hip pitch moving body	4.8×10^{-5}	5.5×10^{-5}	$\frac{0.0445}{1.65\times10^{-8}s^2+9.41\times10^{-5}s+0.002}$	$\frac{473.1}{s+21.29}$	−5679	−21.37	−2850	21.11	898.3
Right and left hip twist moving leg	1.41×10^{-6}	7.93×10^{-6}	$\frac{0.0445}{2.4\times10^{-9}s^2+1.4\times10^{-5}s+0.002}$	$\frac{3281.54}{s+147.7}$	−5550	−151.7	−2851	148.3	43617.1
Right and left hip twist moving body	2.85×10^{-5}	3.5×10^{-5}	$\frac{0.0445}{1.05\times10^{-9}s^2+6\times10^{-5}s+0.002}$	$\frac{741.6}{s+33.38}$	−5667	−33.57	−2850	33.82	2270.01
Right and left hip side to side moving leg	2.5×10^{-5}	3.2×10^{-5}	$\frac{0.0445}{9.6\times10^{-9}s^2+5.5\times10^{-5}s+0.002}$	$\frac{812.34}{s+36.56}$	−5664	−36.8	−2850	36.44	2602.92
Right and left hip side to side moving body	4.7×10^{-5}	5.3×10^{-5}	$\frac{0.0445}{1.6\times10^{-8}s^2+9.1\times10^{-5}s+0.002}$	$\frac{487.7}{s+21.95}$	−5678	−22.03	−2850	21.75	945.65
Torso pitch and side to side	3.7×10^{-5}	4.3×10^{-5}	$\frac{0.0445}{1.3\times10^{-8}s^2+7.4\times10^{-5}s+0.002}$	$\frac{603.9}{s+27.18}$	−5673	−27.31	−2850	30.44	1654.14
Torso twist	0	6.52×10^{-6}	$\frac{0.0445}{1.96\times10^{-9}s^2+1.12\times10^{-5}s+0.002}$	$\frac{3991.32}{s+179.6}$	−5517	−185.6	−2851	170.53	60903.4

Table 6.6: Results for method 3. The PI zero is the same distance away from the open loop mechanical pole as the open loop mechanical pole is away from the PI pole.

The PI compensator for each joint was then used in the SIMULINK model. The model for each joint was set up the same way as for method 1 and 2 but with the different constants in the PI block. The model was given a step input. The results for all the joints are summarised in table 6.7.

The same PI constants for method 3 were then used in the DYNAMECHS simulator for crouching, balancing on the right foot and leaning forward (see figure 1.3) as for method 1. The model kept falling backwards with the calculated PI constants for the ankle roll and pitch joints moving the body. This is because the calculated proportional gain was not high enough so instead of using the method 3 constants for the ankle roll and pitch joints, the method 1 constants were used only for the ankle roll and pitch joints. For crouching, it is desirable to see information on the hip pitch, knee and ankle pitch joints which will be similar for both legs, so only the right leg information is presented. The results are shown in figure 6.13.

As can be seen from figure 6.13, the hip pitch (or hip fwd) joints are not within the current limits. The torques oscillate consistently and the current saturates to its limits, but the position error is very small (almost zero). The knee joints saturate to 4A consistently and there is no position error. The ankle pitch (or ankle fwd) joints stay within the current limits with no position error. The net current exceeds the 20A limit.

For balancing on the right foot, it is desirable to see information on the right hip side to side joint. The results are shown in figure in figure 6.14.

As can be seen from 6.14, the hip side to side joint saturates the current limits for most of the time with no error in position. The net current exceeds the 20A limit.

For leaning forward, it is desirable to see information on the torso pitch joint. The results are shown in figure 6.15.

As can be seen from figure 6.15, the torso pitch (or torso fwd) joint saturates to the current limits with no error in position. The net current again exceeds the 20A current limit.

Joint	Response	Settling time (T_s, sec.)	%OS	s.s.e	Final voltage (V)	Duty cycle
Ankle roll and pitch moving body	underdamped	3.3	0.6	0	13.81	0.431
Ankle roll and pitch moving foot	underdamped	0.003	1.7	0	7.2	0.23
Knee	underdamped	0.005	1.7	0	8.85	0.28
Right and left hip pitch moving leg	underdamped	0.03	8	0	11.3	0.353
Right and left hip pitch moving body	underdamped	0.094	4.6	0	10.7	0.33
Right and left hip twist moving leg	underdamped	0.011	22.6	0	7	0.22
Right and left hip twist moving body	underdamped	0.047	7	0	7	0.22
Right and left hip side to side moving leg	underdamped	0.04	3	0	13	0.41
Right and left hip side to side moving body	underdamped	0.09	5	0	10.6	0.33
Torso pitch and side to side	underdamped	0.06	4.6	0	9.8	0.31
Torso twist	underdamped	0.005	2.5	0	7	0.22

Table 6.7: Method 3 PI compensator.

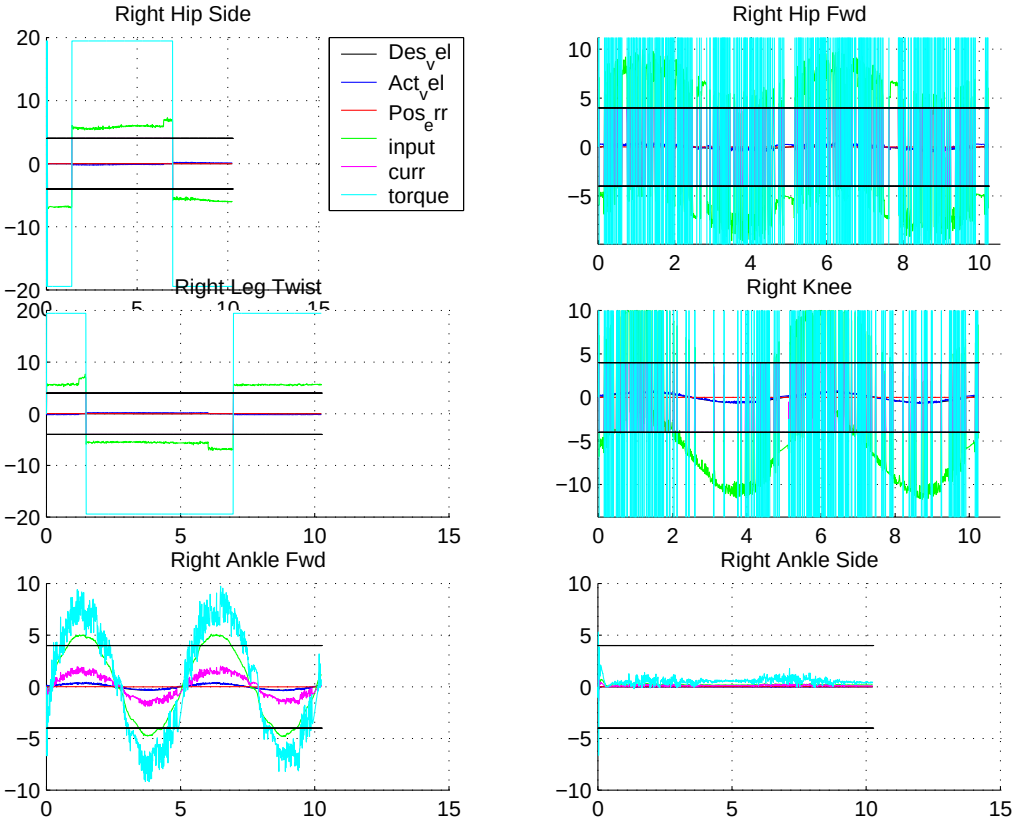


Figure 6.13: Method 3 DYNAMECHS results for crouching. Right leg data.

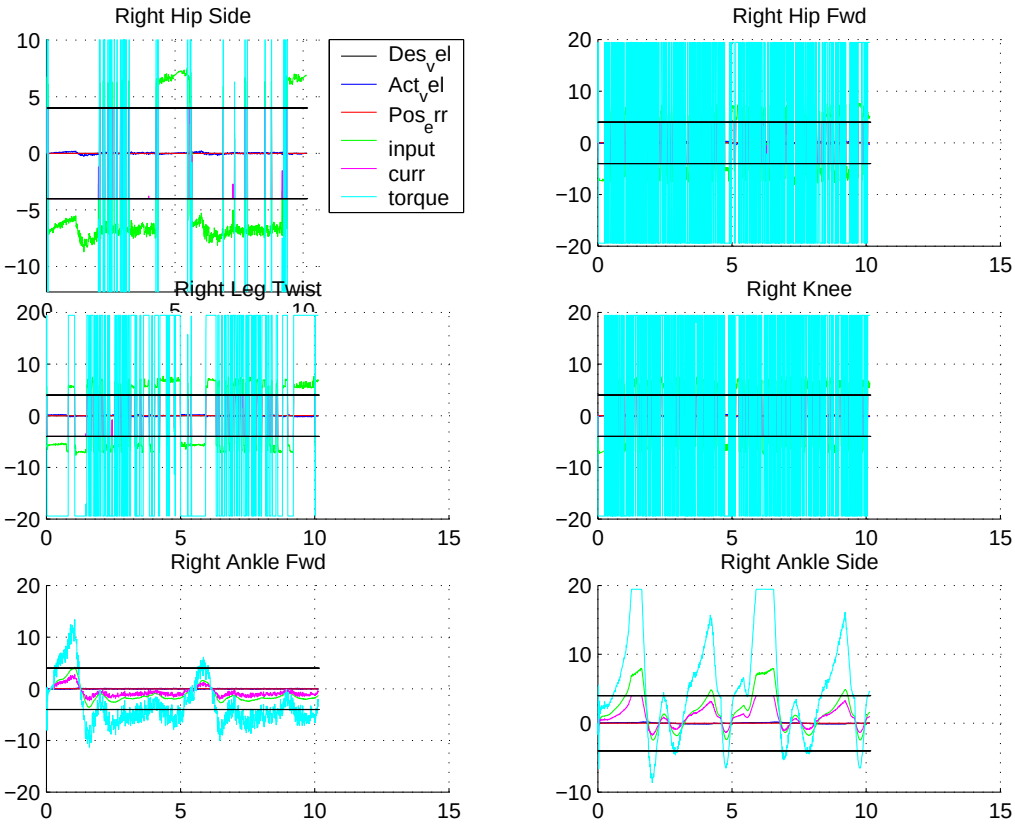


Figure 6.14: Method 3 DYNAMECHS results for balancing on the right foot. Right leg data.

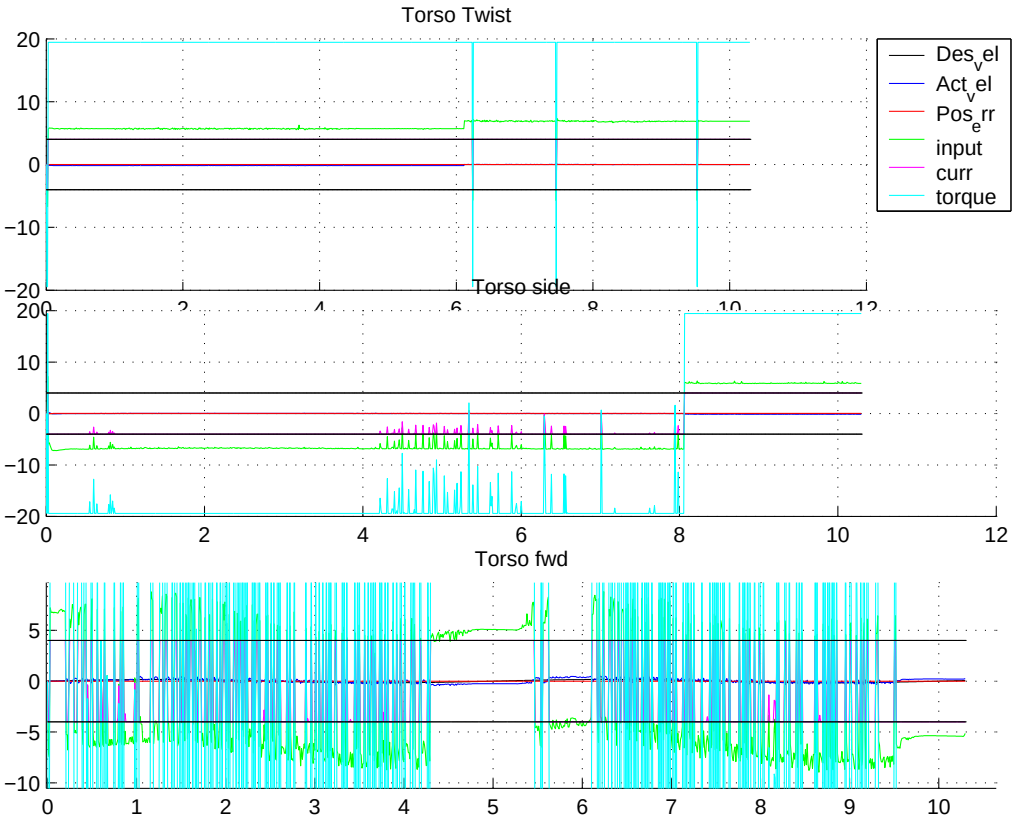


Figure 6.15: Method 3 DYNAMECHS results for leaning forward. Torso data.

Chapter 7

Conclusions and Future Work

7.1 Outcomes

This thesis achieved the following:

- A model was created to model a DC motor or plant controlling each joint in the lower body and torso of the *GuRoo* humanoid robot. The humanoid model was simplified by treating the load as a static point mass at the worst case angle on each joint acting at the estimated radius of gyration from the joint (based on the dimensions of the robot), with the offset due to gravity being taken into account.
- The uncompensated response of each joint was looked at in SIMULINK and a comparison was made to the compensated response. The compensation was a proportional plus integral (PI) controller with feedforward cancellation. Three methods of PI compensation were investigated. The first method looked at positioning the PI zero very close to the PI pole at the origin. The second method looked at positioning the zero further away from the PI pole at the origin but still far to the right of the open loop mechanical pole of the system. The third method investigated positioning the PI zero to the left of the open loop mechanical pole.

- From the SIMULINK model, the proportional and integral constants were obtained for each joint in each method and these constants were tested in the DYNAMECHS simulator for various motions of the robot, specifically, crouching, balancing on the right foot and leaning the upper body forward with the torso pitch joint. Firstly, the uncompensated response was looked at in the DYNAMECHS simulator for all motions. The model did not perform any of the movements and always fell backwards as in figure 6.1. The compensated response was then tested in the simulator for each joint using all three methods. Each method returned almost no position error for a step input, but only method 1 kept the currents in the armature below the hard limit of 4 amperes. Because the electro-mechanical design was not constructed in time, it was not possible to test any of the control loops in hardware, and therefore, no comparison can be made between the simulation results and results of the actual response of the robot. This will have to be done next year, when the robot will be built.

7.2 Future Work

This section will discuss some future control projects and things to do with the *GuRoo* robot.

7.2.1 Modelling the Damping of the DC Rotor

The value for the damping of the armature and load reflected to the armature was obtained by having no load on the joint and no offset due to gravity in the SIMULINK model. It was then made certain that the no load speed from appendix E was obtained at the motor shaft for the nominal input from the data-sheet with no feedback in the model. When the actual motors become available, it will be possible to obtain the actual damping of the DC rotor experimentally. This can be done by having no load on the motor and let it accelerate to its no load speed. At this point, ideally, the back EMF will equal the

input into the motor and no current will flow in the armature. However, in practice, there will damping which will result in a small amount of current still flowing in the armature at the no load speed. This current creates the losses due to friction according to the equation $\tau_{friction} = K_t I_{small}$, where I_{small} is the current at the no load speed that generates the frictional torque. A good model for friction is to say that the frictional torque is proportional to the speed of the motor, and the constant of proportionality will then be the damping, or $\tau_{friction} = D_{armature}\omega$. If ω is the no load speed, then the viscous damping of the armature can be obtained experimentally. I_{small} can be measured with an oscilloscope and the rest of the values can be obtained from the data-sheet in appendix E. $D_{armature}$ will then be $\frac{\tau_{friction}}{\omega}$.

7.2.2 PI tuning

The software to perform the PI control loops was written for the DYNAMECHS simulator. This software must be tested on the actual motors and tuned to achieve optimum performance in the environment that the robot will operate in. PI tuning is the process of adjusting the proportional and integral gains in the control loops by trial and error, or in other words, using rules of thumb [6]. This is achieved by adjusting the values in equation 5.1 and observing the performance of each joint in the robot.

PI tuning is possible because it is known what the response of each motor controlling each joint should be, that is, a response that has minimum overshoot and accurate trajectory following so that each joint would reach a designated position from the central controller in order for the entire robot to be able to walk.

When the electro-mechanical design is built, the PI gains obtained form the SIMULINK model must be tested. The response of each joint must then be observed. Then through a process of PI tuning, modifications will need to be made. The control loops must then be recompiled onto the hardware and tested again. This will keep going until a control law is achieved that produces a desired response. From past experience of other teams such as the 1998 *RoboRoos* project [6], the process of PI tuning can be very extensive and often

after a laborious session of PI tuning, no improvements are made to the response of the system and the default values used for the gains are as good as any.

7.2.3 Soft Current Limiting

The current limit of 4A that was used in testing is a hard limit. There must be a mechanism to enable for limiting the joint input and therefore, create a soft current limiter. This would mean that once the current reaches a value close to the current limit of 4A, the PWM duty cycle needs to be reduced. For instance, once the current reaches 3.5A, say, the PWM duty cycle will be reduced by a certain amount, thereby reducing the joint input into the plant and therefore reducing the current in the armature of each motor. The soft current limiter will need to be incorporated into the main flow of the control loop in figure 5.2 as in figure 7.1.

7.2.4 DYNAMECHS Simulator

The DYNAMECHS simulator will need to be upgraded to a model that takes into account the sensors that will be used for the robot [24] and a way to model external disturbances on the robot. There needs to be a mechanism of modelling external disturbances on the robot in the simulator to see their effects on the control loops, the movements of the robot and the reaction of the robot to these disturbances. This can be done by means of a point and click method, where the mouse can be position on the desired point of the robot in the OpenGL environment (see figure 1.3) and by clicking on that point, a disturbance can be placed on the robot and the response of the model can be observed. This will be a very useful feature in the simulator because it will enable to model the robot making contact with a soccer ball by clicking around the feet area at certain points in the gait or making contact with other humanoid robots in the soccer field by clicking around the shoulder area for a side contact or the chest area for a frontal collision between two humanoid robots.

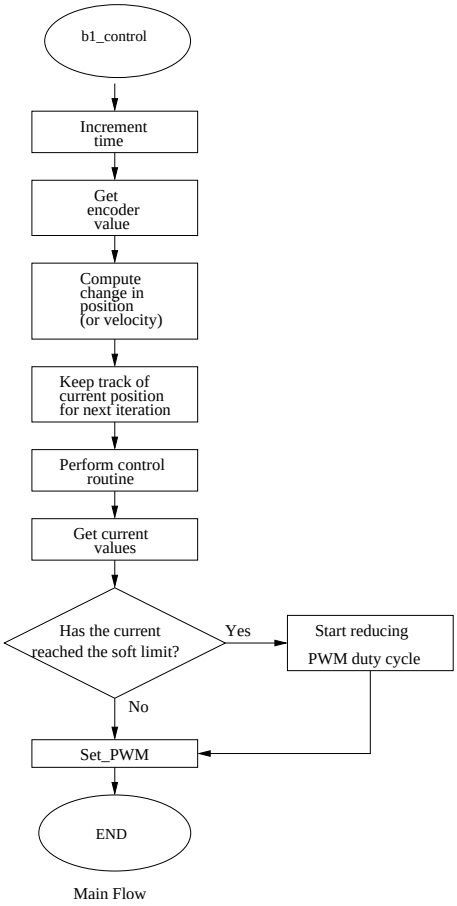


Figure 7.1: Addition of a soft current limiter.

The simulator also needs to take into account the noise introduced from the various sensors that will be used in the robot. The robot will eventually use duel axis gyroscopes which measure angular rates, accelerometers which measure straight line acceleration, pressure sensors on the feet and limit switches to constrain the movements of each joint to within a certain range [24], as well as the encoders that tell the position of the joint. These sensors add on noise to the response of each joint, which should be taken into account in the DYNAMECHS simulator as this could make the difference between a stable and an unstable system.

7.2.5 Closing the loop—Adaptive Control

As was discussed in chapter 2, open loop walking is a power based method used to control the movements of the *GuRoo*. A central controller sends position commands or velocity profiles to each joint and the control on each joint is performed locally, where the encoder counts of the joint are compared to the most recent command from the central controller so that a PWM frequency can be calculated to drive each joint based on a control law, in this case a PI controller with feed forward cancellation.

When the robot will be walking, it will be constantly changing its posture as it is moving and the loadings on each motor controlling each joint will be changing with time. In other words, the inertia due to the load will be a function of time and the offset due to gravity will also be a function of time. This means that the transfer function in equation 4.12 and 4.27 (first order approximation of the plant) will be changing with time because they are a function of inertia, and therefore the compensation block will need to change as well to cope with the changing parameters of the plant (or motor). This implies the use of an adaptive control scheme to create a more robust controller that will continuously modify the input to the plant.

Bibliography

[1] Arimoto, S. and M. Takegaki (1981), *An adaptive method for trajectory control of manipulators,* IFAC 8th triennial World Congress, Kyoto, August.

[2] B. Bebel, *Design and Implementation of a USB-to-CAN Bridge for the GuRoo Project*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[3] N. Brewer, *Power Systems for a Humanoid*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[4] T. Cartwright, *Design and Implementation of small scale joint controllers for a humanoid Robot*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[5] Dubowsky, S. and D. T. Desforges (1979), *Robotic manipulator control systems with invariant dynamic characteristics*, Proceedings of the 5th world congress on Machines and Mechanisms (IFIOMM), Montreal, July.

[6] M. Ford, *Operating System and Motor Control Software for a Soccer Playing Robot*, undergraduate thesis, Univ. of Queensland, Dept. of Computer Science and Electrical Engineering, 1998.

[7] Honda Technology Specifications, `http://www.honda.co.jp/tech/other/spec1.html`, (current July 2, 2001).

[8] J.A. Golden and Y.F. Zheng. Gait Synthesis for the SD-2 Biped Robot to Climb Stairs. *International Journal of Robotics and Automation*, v5, No. 4, pages 149-159, 1990.

[9] K. Hirai, M. Hirose, Y. Haikawa and Takenaka. The Development of Honda Humanoid Robot. *IEEE Conference on Robotics and Automation*. v2, pages 1321-1326, 1998.

[10] S. Horn, *Thecodont—The Biped Walker*, undergraduate thesis, Univ. of Queensland, Dept. of Electrical and Computer Engineering, 1999.

[11] D. Kee, *Drive System Selection for a Humanoid*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[12] T. McGeer, Passive Dynamic Walking. *International Journal of Robotics and Research*, v9, No. 2, pages 62-82, 1990.

[13] P. J. McKerro, *Introduction to Robotics*, Addison Wesley, Sydney, 1991.

[14] S. McMillan, Computational Dynamic for Robotic Systems on Land and Under Water, PhD Thesis, Ohio State University, Dept. of Electrical Engineering, 1995.

[15] Nicolo. F and J. Katende (1983), *A robust MRAC for Industrial Robots*, 2nd IASTED International Symposium on Robotics and Automation, Lugano.

[16] N. S. Nise, *Control Systems Engineering*, John Wiley & Sons, New York, 2000.

[17] K. Ogata, *Modern Control Engineering*, Prentice-Hall International, New Jersey, 1997.

[18] A. S. Parseghian, *Control of a Simulated, Three-dimensional Bipedal Robot to initiate Walking, Continue Walking, Rock Side to Side and Balance*, Masters thesis, Department of Electrical Engineering and Computer Science, 1998.

[19] The Robot World Cup Initiative, "*RoboCup:* Objective", Robot Soccer `http://www.robocup.org/overview/22.html`, (current July 2, 2001).

[20] R. J. Schilling, *Fundamentals of Robotics, Analysis and Control*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[21] A. Smith, *Simulator Adaption and Gait Pattern Creation for a Humanoid Robot*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[22] J. Stirzaker, *Design of DC Motor Controllers for a Humanoid Robot*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[23] ViperRoos-Hardware Design, `http://www.csee.uq.edu.au/~chang/ViperRoos/index.html`, (current Aug. 5, 2001).

[24] M. Wagstaff, *Mechanical Design and Internal Sensors for a Humanoid Robot*, undergraduate thesis, Univ. of Queensland, Dept. of Information Technology and Electrical Engineering, 2001.

[25] G. Wyeth, Design of an Autonomous Humanoid Robot, accepted by the *Australian Conference on Robotics and Automation*, Sydney, 2001.

Appendix A

Schematic of the *GuRoo*

Name of Link	Mass (*kg*)
Head	0.516
Neck	0.47
Torso (including neck pan motor and both shoulder rotation motors)	8.545
Shoulder up and down joint	0.206
Upper arm (including elbow motor)	0.381
Lower arm	0.328
Waist 1 (including torso twist motor)	1.984
Waist 2 (including torso front to back motor)	1.491
Hip	3.793
Abduction (including upper leg front to back motor)	1.51
flexion (just the frame)	0.871
Upper leg (including leg twist and knee motor)	3.193
Lower leg (including ankle pitch motor)	1.718
Ankle (frame plus ankle roll motor)	1.476
Foot	1.167

Table A.1: Individual Masses of each link used for testing.

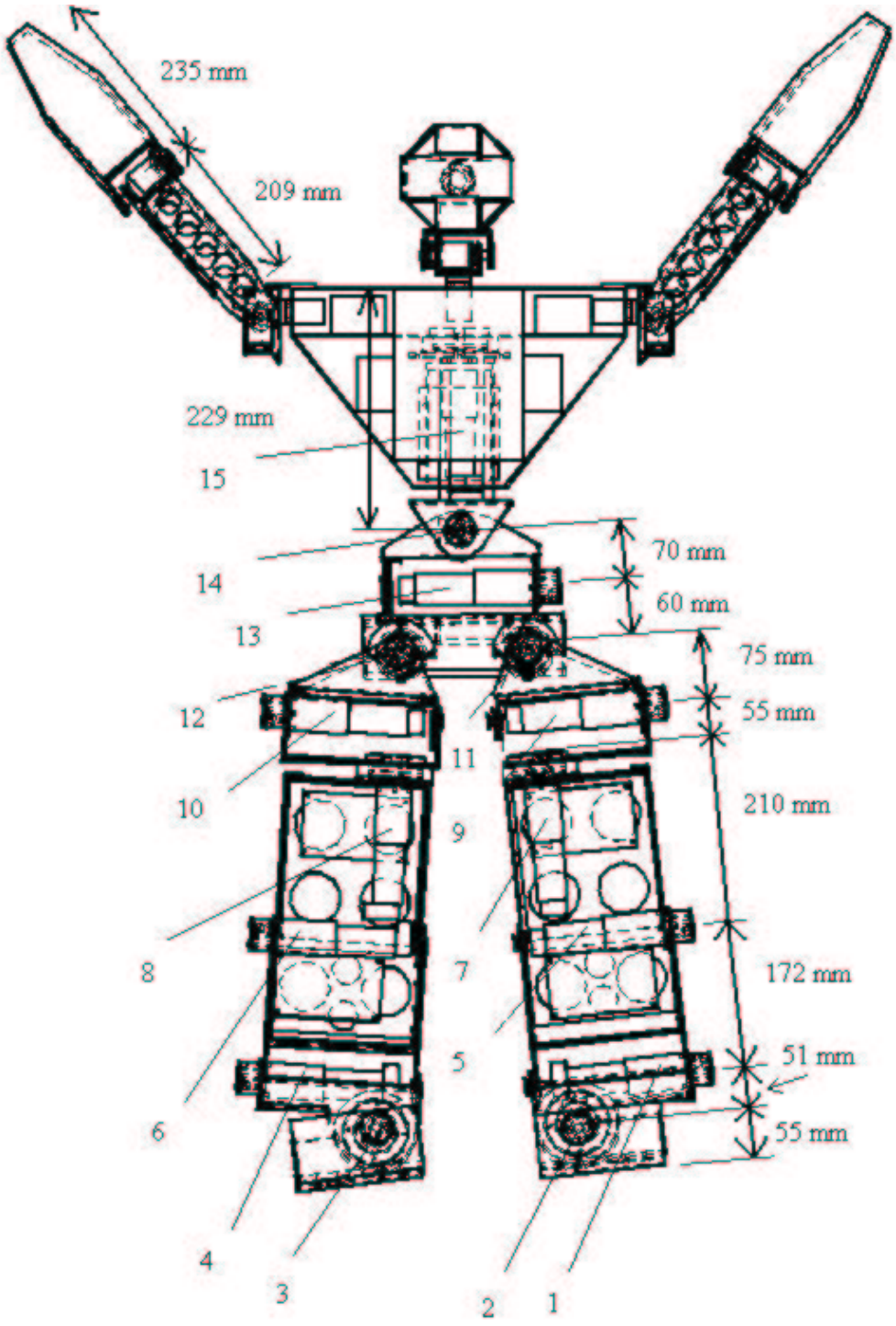


Figure A.1: Joint numbers of the lower body joints and torso and the lengths used in the SIMULINK model. Taken from [24].

Appendix B

Derivation of Offset Transfer Function

In section 4.2, the transfer function between the offset torque, $\frac{\tau_l}{N}$ and the motor speed was stated for an input reference of zero in figure 4.3. It is derived here. Let

$$\frac{C(s)}{X(s)} = \frac{1}{Js+D} \Rightarrow C(s) = \frac{X(s)}{Js+D} \tag{B.1}$$

$$X(s) = \frac{E(s)\cdot K_t}{L_a s+R_a} - \frac{T_l(s)}{N} \tag{B.2}$$

The error will be $-C(s)$ for a reference input of zero, so

$$X(s) = \frac{-C(s)K_bK_t}{L_a s+R_a} - \frac{T_l(s)}{N} \tag{B.3}$$

$$= \frac{-X(s)K_bK_t}{(L_a s+R_a)(Js+D)} - \frac{T_l(s)}{N} \tag{B.4}$$

$$X(s) + \frac{X(s)K_bK_t}{(L_a s+R_a)(Js+D)} = -\frac{T_l(s)}{N} \tag{B.5}$$

$$X(s)\left[1 + \frac{K_bK_t}{(L_a s+R_a)(Js+D)}\right] = -\frac{T_l(s)}{N} \tag{B.6}$$

$$X(s) = \frac{-\frac{T_l(s)}{N}}{1+\frac{K_bK_t}{(L_a s+R_a)(Js+D)}} \tag{B.7}$$

When equation B.7 is substituted into equation B.1,

$$\begin{aligned}
\frac{-NC(s)\left[1+\frac{K_bK_t}{(L_as+R_a)(Js+D)}\right]}{T_l(s)} &= \frac{1}{Js+D} \\
-\frac{NC(s)}{T_l(s)} &= \frac{1}{Js+D}\times\frac{1}{\left[1+\frac{K_bK_t}{(L_as+R_a)(Js+D)}\right]} \\
&= \frac{1}{(Js+D)\left(1+\frac{K_bK_t}{(L_as+R_a)(Js+D)}\right)} \\
&= \frac{1}{(Js+D)\left(\frac{(L_as+R_a)(Js+D)+K_bK_t}{(L_as+R_a)(Js+D)}\right)} \\
&= \frac{1}{\frac{(L_as+R_a)(Js+D)+K_bK_t}{(L_as+R_a)}} \\
&= \frac{(L_as+R_a)}{(L_as+R_a)(Js+D)+K_bK_t} \\
\frac{C(s)}{T_l(s)} &= \frac{-\frac{1}{N}(L_as+R_a)}{(L_as+R_a)(Js+D)+K_bK_t}
\end{aligned}$$

Appendix C

SIMULINK Model

Here is the SIMULINK model used for every joint. The example here is for the ankle roll and pitch joints moving the upper body.

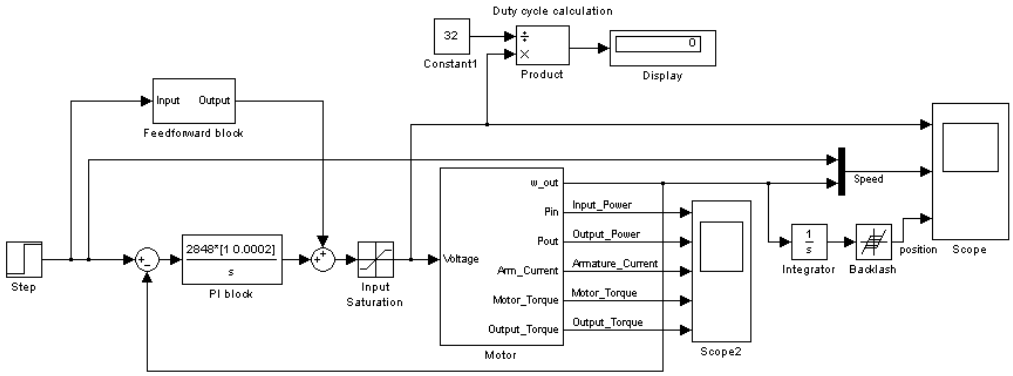


Figure C.1: The control loop for each joint. This one is for the ankle roll and pitch joints moving the upper body. For different joints, the numbers are simply different.

The plant or motor block in figure C.1 is shown in figure C.2.

The feedforward block in figure C.1 is shown in figure C.3.

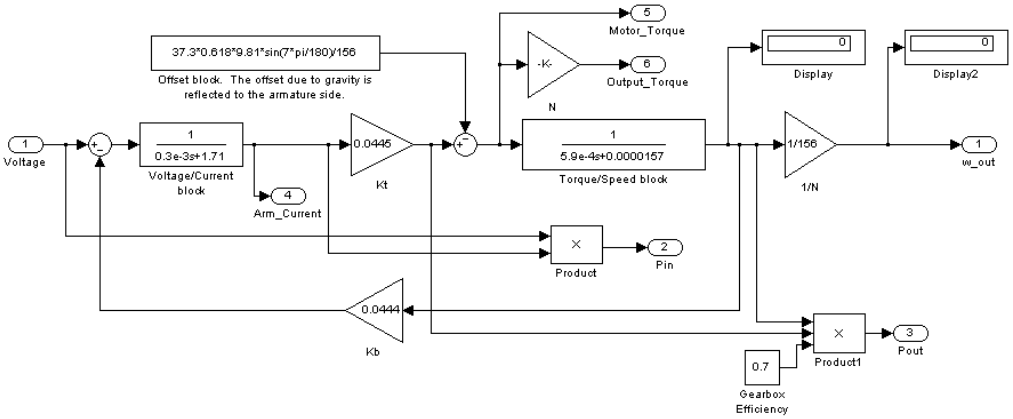


Figure C.2: The plant or motor block for the ankle roll and pitch joints moving the upper body. The numbers are different for different joints.

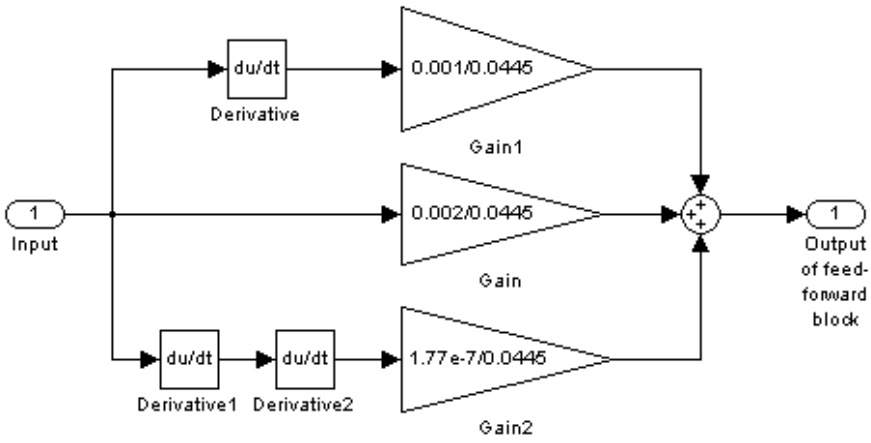


Figure C.3: The feedforward block for the ankle roll and pitch joints. Again, the numbers are different for different joints.

Appendix D

Software

Program D.1: DC motor control code

```
/*****************************************************************************
 * Copyright 2001, Gordon Wyeth
 *****************************************************************************
 *      File: board1.c
 *    Author: Gordon Wyeth
 *   Project: Humanoid 0.1
 *   Created: 26 March 2001
 *   Summary:
 *****************************************************************************/

#include "board1.h"
#include "angles.h"
#include "jointnum.h"
#include "control.h"
#include "humanoid.hpp"
#include "can.h"
#include <stdio.h>

// These are defined here to make logging data easier

static float desired_joint_vel[NUMBER_MOTORS];
static float I_err[NUMBER_MOTORS];

void b1_control(void)
{
    int i;
    float joint_pos;
    float current;
    float pwm_duty;
    float delta_pos;

    static int   motor_num = 0;
    static float time;
    static float request[NUMBER_MOTORS]; // Float so it can be passed to CAN check
    static float old_joint_pos[NUMBER_MOTORS];

    time += (float)(IDT);

    // Run each BOARD_DELAY seconds

    if (time >= BOARD_DELAY) {

            // Update from CAN

```

```
45              for (i = 1; i <= NUMBER_MOTOR_BOARDS; i++) {
46                      CAN_Check(i, &desired_joint_vel[(i - 1)*MOTORS_PER_BOARD]);
47              }
48
49              for (i = REQUEST_BOX_1; i <= (NUMBER_MOTOR_BOARDS + REQUEST_BOX_1); i++) {
50                      CAN_Check(i, &request[(i - REQUEST_BOX_1)*MOTORS_PER_BOARD]);
51              }
52
53              // For data logging purposes
54
55              for (i = 0; i < NUMBER_MOTORS; i ++) {
56
57                      if (desired_joint_vel[RIGHT_HIP_SIDE] > 0.0)
58                              time = time;
59
60                      // recieve encoder value
61                      joint_pos = Read_Enc(i);
62
63                      // Compute change in position
64                      delta_pos = (float)((joint_pos - old_joint_pos[i]) /
                            BOARD_DELAY);
65
66                      // Keep track of current position for next time
67                      old_joint_pos[i] = joint_pos;
68
69                      // Compute Control Law
70                      pwm_duty = Compute_Law(i, desired_joint_vel[i], delta_pos);
71
72               // Read current
73                      current = Read_Curr(i, pwm_duty);
74
75                      // Send of current if there has been a request for it
76                      if (request[i] == 1) {      //THIS ONE SHOULD BE A TRUE
77                              Request_Send(IPAQ_BOX, &current);
78                              request[i] = FALSE;
79                      }
80
81                      // SET PWM
82                  Set_PWM(i, pwm_duty);
83
84
85                  }
86                  time -= BOARD_DELAY;
87      }
88  }
89
90  //*************************************************************************
91  //
92  // Routines
93  //
94  //*************************************************************************
95
96  float Compute_Law(int i, float desired_joint_vel, float joint_vel)
97  {
98          float P_GAIN[NUMBER_MOTORS], I_GAIN[NUMBER_MOTORS], F_GAIN[NUMBER_MOTORS];
99
100         // Grab control values
101         init_Gains(P_GAIN, I_GAIN, F_GAIN);
102
103         // Calculate velocity proportional error
104         float P_err = desired_joint_vel - joint_vel;
105
106         // Calculate velocity integral error
107         I_err[i] += (float)(P_err * IDT);
108
109         //*****************
110         //
111         // control
112         //
113         //*****************
114
115         float joint_input = ((P_err+(I_err[i]*I_GAIN[i]))*P_GAIN[i]) + (F_GAIN[i]*
                desired_joint_vel);
```

```
116
117         float pwm_duty = (float)(joint_input/MOTOR_VOLTAGE);
118
119         // Make sure that the PWM frequency is not exceeded
120         if (pwm_duty  < -1.0)
121                 pwm_duty = -1.0;
122         else if (pwm_duty > 1.0)
123                 pwm_duty =  1.0;
124
125         // Check for voltage saturation, not needed on real boards.  On real boards,
126         // Will also need to check that the encoder register has not overflown and
127         // check if the motor is jammed or stuck.
128
129         return(pwm_duty);
130 }
131
132 // An initialisation function
133 void initialise_motor_board(void) {
134
135         extern float * log_desired_joint_vel;
136         extern float * log_I_err;
137         log_desired_joint_vel = desired_joint_vel;
138         log_I_err = I_err;
139 }
```

Program D.2: RC servo-motor control and current calculations code

```
1  /*****************************************************************************
2   * Copyright 2001, Gordon Wyeth
3   *****************************************************************************
4   *      File: humanoid.cpp
5   *    Author: Gordon Wyeth
6   *   Project: Humanoid 1.00
7   *   Created: 6 June 2001
8   *   Summary:
9   *****************************************************************************/
10
11 // This routine simulates the proportional control on the RC servo-motors
12
13 void servo_control(float desired_joint_pos[NUMBER_SERVOS])
14 {
15         float joint_pos[1];
16         float joint_vel[1];
17
18         for (int i = 0; i < NUMBER_SERVOS; i++) {
19
20                 int k = (NUMBER_MOTORS + i);
21                 int j = Joint_Conversion(k);
22
23                 // Get position
24                 robot_link[j]->getState(joint_pos, joint_vel);
25
26                 // Initialise the position of the head pitch joint to 90 degrees
27                 if (j == SIM_HEAD)
28                         joint_pos[0] = joint_pos[0] - PI/2.0;
29
30                 // Perform proportional compensation on servo_motors
31                 joint_input[k] = (float)(SERVO_P * (desired_joint_pos[i] - joint_pos
                       [0]));
32         }
33 }
34
35 // Calculates Current
36 float curr_calc(int i, float joint_input, float * motor_speed)
37 {
38         float joint_pos[1];
39         float joint_vel[1];
40
41     int j = Joint_Conversion(i);
42         robot_link[j]->getState(joint_pos, joint_vel);
43
44         // Using equation 4.4 to calculate the armature currents
45         motor_speed[0] = joint_vel[0] * 156.0 * 60.0 / (2.0 * PI);
```

```
46      float current = (joint_input - motor_speed[0] / 215.0) / 1.71;
47
48          // This line is used for data logging
49          log_joint_vel[i] = joint_vel[0];
50
51          return current;
52  }
```

Program D.3: Central Controller for movement generation

```
1   /*****************************************************************************
2    * Copyright 2001, Gordon Wyeth
3    *****************************************************************************
4    *      File: central.c
5    *    Author: Gordon Wyeth
6    *   Project: Humanoid 0.1
7    *   Created: 26 March 2001
8    *   Summary:
9    *****************************************************************************/
10
11  #include "central.h"
12  #include "can.h"
13  #include "jointnum.h"
14  #include "humanoid.hpp"
15
16  #include <math.h>
17  #include <stdio.h>
18
19  #define DOWN_CYCLE_TIME 2.0
20  #define DOWN_TIME (DOWN_CYCLE_TIME / 2.0)
21  #define UP_CYCLE_TIME 2.0
22  #define UP_TIME (UP_CYCLE_TIME / 2.0)
23  #define HIP_ANGLE    (25.0 * (PI / 180.0)) // For crouching
24  #define KNEE_ANGLE   (55.0 * (PI / 180.0)) // For crouching
25  #define ANKLE_ANGLE (30.0 * (PI / 180.0)) // For crouching
26  #define HIP_SIDE_ANGLE    (15.0 * (PI / 180.0))
27  #define ANKLE_SIDE_ANGLE (8.0 * (PI / 180.0))
28  #define RIGHT_UPPER_ARM_ANGLE (30.0 * (PI / 180.0))
29
30  void central_control (void)
31  {
32      static float time;
33      static float count_time;
34      float velocity;
35      float desired_joint_vel[NUMBER_MOTORS + NUMBER_SERVOS];
36
37      // Initialising the array is not needed as array is not static
38      for (int i = 0; i < (NUMBER_MOTORS + NUMBER_SERVOS); i++) {
39                  desired_joint_vel[i] = 0.0;
40      }
41
42      time += (float)(IDT);
43      count_time += (float)(IDT);
44
45      if (count_time >= CENTRAL_SPEED) {
46
47      // The motions are periodic with a period of 5 seconds.
48      #define PERIOD1 5.0
49
50      velocity = (float)(.5 * (2.0*PI/PERIOD1) * sin(2.0*PI*time/PERIOD1));
51
52      #define BALANCE
53
54      // This defines what motion to perform, in this case a crouch.
55      #ifdef CROUCH
56
57                  /* Crouching motion */
58                  desired_joint_vel[LEFT_HIP_FWD]   = (float)(HIP_ANGLE * velocity);
59                  desired_joint_vel[RIGHT_HIP_FWD] = (float)(HIP_ANGLE * velocity);
60                  desired_joint_vel[LEFT_KNEE]   = (float)(KNEE_ANGLE * velocity);
61                  desired_joint_vel[RIGHT_KNEE] = (float)(KNEE_ANGLE * velocity);
62                  desired_joint_vel[LEFT_ANKLE_FWD] = (float)(ANKLE_ANGLE * velocity);
63                  desired_joint_vel[RIGHT_ANKLE_FWD] = (float)(ANKLE_ANGLE * velocity);
64      #endif
```

```
65
66      #ifdef BALANCE
67
68                      /* Balancing motion */
69              desired_joint_vel[RIGHT_ANKLE_SIDE] = (float)(ANKLE_SIDE_ANGLE * velocity
                        );
70          desired_joint_vel[LEFT_ANKLE_SIDE] = (float)(ANKLE_SIDE_ANGLE * velocity);
71          desired_joint_vel[TORSO_SIDE] = (float)(HIP_SIDE_ANGLE * velocity);
72          desired_joint_vel[LEFT_HIP_SIDE] = (float)(-HIP_SIDE_ANGLE * velocity);
73              desired_joint_vel[LEFT_KNEE] = (float)(KNEE_ANGLE * velocity);
74              desired_joint_vel[LEFT_HIP_FWD] = (float)(HIP_ANGLE * velocity);
75                  desired_joint_vel[LEFT_ANKLE_FWD] = (float)(ANKLE_ANGLE * velocity);
76      #endif
77
78      #ifdef LEAN_FWD
79
80          /* Leaning forward */
81              desired_joint_vel[TORSO_FWD] = (float)(velocity * 15.0 * PI/180.0);
82      #endif
83
84      // Send data to b1_control which simulates all 5 DC controller boards
85      for (int i = 1; i < IPAQ_BOX; i++) {
86                      int motor_num = (i-1) * MOTORS_PER_BOARD;
87                      Request_Send(i, &desired_joint_vel[motor_num]);
88      }
89      count_time -= count_time;
90     }
91  }
```

Appendix E

Maxon DC motor Data sheet

The DC motor used in the Maxon data-sheet is the RE36 32 volt motor. Also refer to `www.maxonmotor.com` for more information.

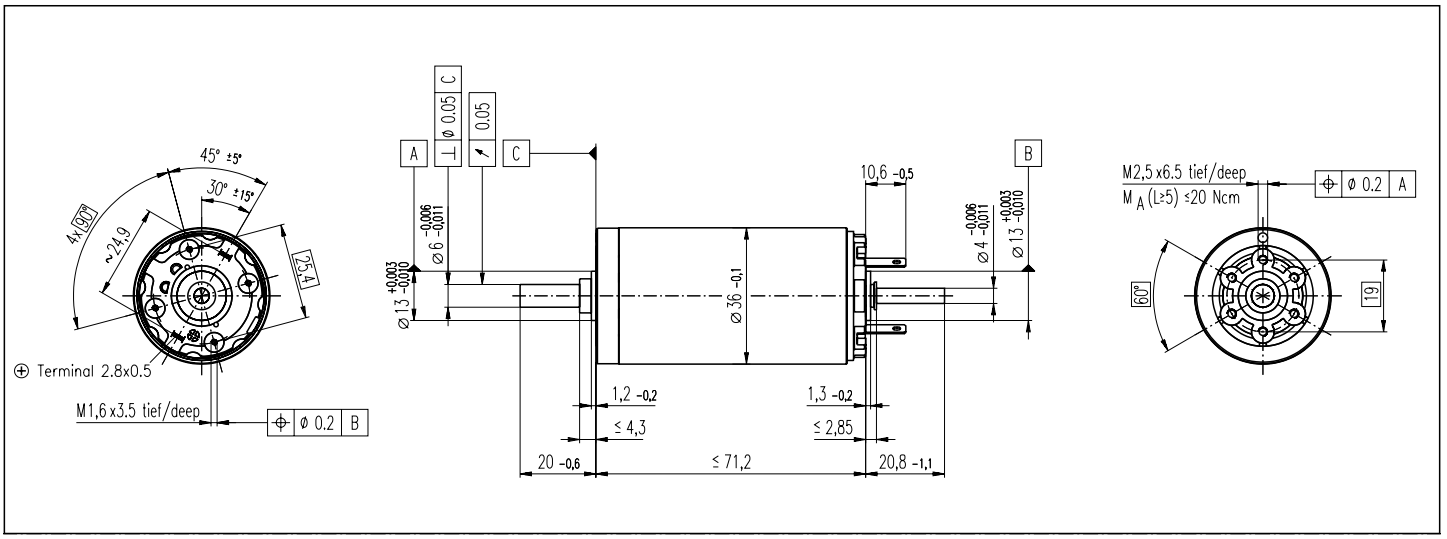
The values that are used from the data-sheet are summarised in table E.1.

Description	Value
Nominal voltage	32 V
No load speed	711.05 $rads^{-1}$
Terminal resistance	1.71Ω
Torque constant	0.0445 $\frac{N}{A}$
Speed constant	0.0444 $\frac{Vs}{rad}$
Rotor inertia	6.52×10^{-6} kgm^2
Terminal inductance	0.3 mH
Gearbox efficiency	0.7
Gear ratio	156

Table E.1: Values used from data-sheet.

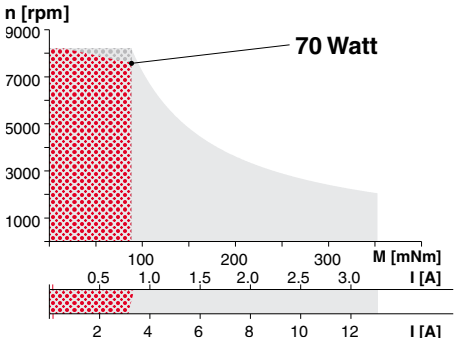
RE 36

∅36 mm, Graphite Brushes, 70 Watt



Motor Data:			Order Number													
			118797	118798	118799	118800	118801	118802	118803	118804	118805	118806	118807	118808	118809	118810
1	Assigned power rating	W	70	70	70	70	70	70	70	70	70	70	70	70	70	70
2	Nominal voltage	Volt	18.0	24.0	32.0	42.0	42.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0
3	No load speed	rpm	6410	6210	6790	7020	6340	6420	5220	4320	3450	2830	2280	1780	1420	1180
4	Stall torque	mNm	730	783	832	865	786	785	627	504	403	326	258	198	158	127
5	Speed/torque gradient	rpm/mNm	8.96	8.05	8.27	8.19	8.14	8.25	8.41	8.65	8.67	8.80	8.96	9.17	9.21	9.51
6	No load current	mA	147	105	89	70	61	55	42	33	25	20	15	12	9	7
7	Starting current	A	27.8	21.5	18.7	15.3	12.6	11.1	7.22	4.80	3.06	2.04	1.30	0.784	0.501	0.334
8	Terminal resistance	Ohm	0.647	1.11	1.71	2.75	3.35	4.32	6.65	10.00	15.7	23.5	36.8	61.3	95.8	144
9	Max. permissible speed	rpm	8200	8200	8200	8200	8200	8200	8200	8200	8200	8200	8200	8200	8200	8200
10	Max. continuous current	A	3.14	2.44	1.99	1.59	1.44	1.27	1.03	0.847	0.679	0.556	0.445	0.346	0.277	0.226
11	Max. continuous torque	mNm	82.4	88.8	88.5	89.8	90.4	90.1	89.8	89.0	89.2	88.8	88.1	87.3	87.2	85.8
12	Max. power output at nominal voltage	W	119	125	146	157	129	131	84.9	56.4	36.0	23.9	15.2	9.09	5.78	3.82
13	Max. efficiency	%	84	85	86	86	86	86	85	84	82	81	79	77	75	72
14	Torque constant	mNm/A	26.3	36.4	44.5	56.6	62.6	70.7	86.9	105	131	160	198	253	315	380
15	Speed constant	rpm/V	364	263	215	169	152	135	110	90.9	72.7	59.8	48.2	37.8	30.3	25.1
16	Mechanical time constant	ms	6	6	6	6	6	6	6	6	6	6	6	6	6	6
17	Rotor inertia	gcm²	62.0	67.7	65.2	65.4	65.6	64.6	63.3	61.5	61.3	60.3	59.2	57.8	57.5	55.7
18	Terminal inductance	mH	0.10	0.20	0.30	0.49	0.60	0.76	1.15	1.68	2.62	3.87	5.96	9.70	15.10	21.90
19	Thermal resistance housing-ambient	K/W	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4
20	Thermal resistance rotor-housing	K/W	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
21	Thermal time constant winding	s	39	43	41	41	41	41	40	39	39	38	37	36	36	35

Operating Range



Comments

Details on page 36

Recommended operating range

Continuous operation
In observation of above listed thermal resistances (lines 19 and 20) the maximum permissible rotor temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

118804 Motor with high resistance winding

118797 Motor with low resistance winding

Stock program
Standard program
Special program (on request!)

- Axial play 0.05 - 0.15 mm
- Max. **ball bearing** loads
  axial (dynamic)
  not preloaded 5.6 N
  preloaded 2.4 N
  radial (5 mm from flange) 28 N
  Press-fit force (static) 110 N
  same as above, shaft supported 1200 N
- Radial play **ball bearings** 0.025 mm
- Ambient temperature range -20/+100°C
- Max. rotor temperature +125°C
- Number of commutator segments 13
- Weight of motor 350 g
- Values listed in the table are nominal. For applicable tolerances (see page 33) and additional details please request our computer printout.

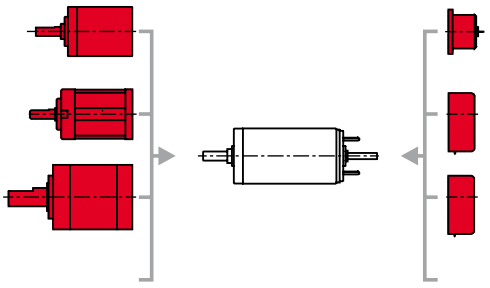
⚠ Tolerances may vary from the standard specification.

maxon Modular System

Planetary Gearhead
∅32 mm
0.75-4.5 Nm
Details page 161

Planetary Gearhead
∅32 mm
0.4-2 Nm
Details page 163

Planetary Gearhead
∅42 mm
3-15 Nm
Details page 165



DC Tacho
∅22 mm
0.52 V
Details page 172

Digital Encoder HP HEDS 5540
500 CTP, 3 channels
Details page 176

Digital Encoder HP HEDL 5540
500 CTP, 3 channels
Details page 178