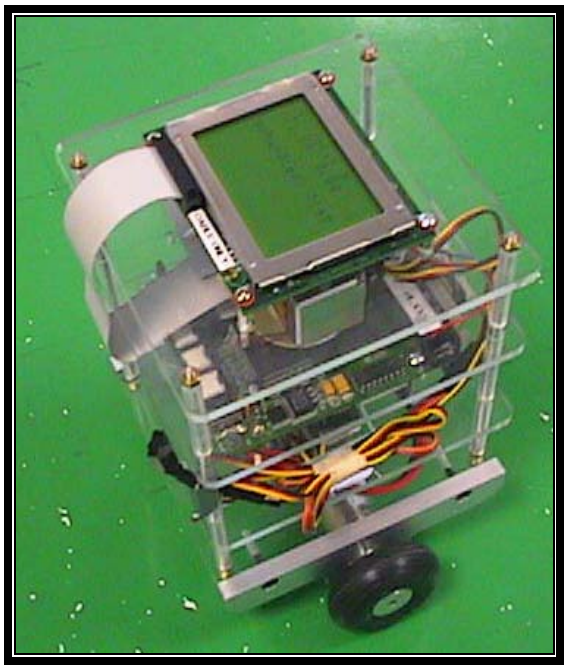
The University of Western Australia

School of Mechanical Engineering

Final Year Thesis 2003

Balancing a Two-Wheeled Autonomous Robot



Rich Chi Ooi

Supervisors : A/Prof. Thomas Bräunl

Prof. Jie Pan

Synopsis

The uniqueness of the inverted pendulum system has drawn interest from many researches due to the unstable nature of the system. The idea of a mobile inverted pendulum robot has surfaced in recent years and has attracted interest from control system researchers worldwide.

In collaboration with the Centre of Intelligent Information Processing System (CIIPS), a two-wheeled differential drive mobile robot based on the inverted pendulum model is built as a platform to investigate the use of a Kalman filter for sensor fusion. As the robot is mechanically unstable, it becomes necessary to explore the possibilities of implementing a control system to keep the system in equilibrium.

This thesis examines the suitability and evaluates the performance of a Linear Quadratic Regulator (LQR) and a Pole-placement controller in balancing the system. The LQR controller uses several weighting matrix to obtain the appropriate control force to be applied to the system while the Pole placement requires the poles of the system to be placed to guarantee stability. As the robot will be moving about on a surface, a PID controller is implemented to control the trajectory of the robot.

Letter of transmittal

Rich Chi Ooi
60 Monash Avenue
Nedlands,WA 6008

3rd November 2003

Executive Dean
Faculty of Engineering and Mathematical Sciences
University of Western Australia
CRAWLEY WA 6009

Dear Sir,

Final Year Thesis

It is with great honour that I submit this thesis entitled *"Balancing a Two-Wheeled Autonomous Robot"* to the University of Western Australia as a requirement for a Degree in Mechatronics Engineering.

Yours sincerely,

_____

Rich Chi Ooi

Acknowledgements

Firstly, I would like express my gratitude to the Faculty of Engineering, Computing & Mathematics for providing me the scholarship to study at the university, The School of Mechanical Engineering, for accepting my transferal, accrediting my past work and providing an outlet for my interests.

The successful completion of this project would not have been possible without the guidance by A/Prof Thomas Bräunl, Prof. Jie Pan and Dr. Roshun Paurobally. Many thanks to Mr. Joshua Pettit (always bursting with ideas and suggestions for me to try out for my project), whom I very much consider as my mentor since the day I got to, know him. Also thanks are due to the members of sourceforge.net autopilot development group, mechanical and electronics workshop of the EE department of UWA for their assistance, advice and support throughout this project.

Thanks to my parents, Frankie Ooi & Sally Chee, for their unconditional love and support, I would not have made it this far without their sacrifices. Not forgetting my true friends, Min Min Chin, Erawati Santoso, Simon Chan, Melissa Seto and Germaine Teo, who have always been supportive of my work and lending a hand at times of need.

Finally, a word or appreciation goes out to all my friends and colleagues at the University for showing interest in my work.

Executive Summary

This thesis discusses the processes developed and considerations involved in balancing a two-wheeled autonomous robot based on the inverted pendulum model. The work was conducted in collaboration with the Centre for Intelligent Information Processing Systems (CIIPS) and the School of Mechanical Engineering.

The balancing robot platform proved to be an excellent test bed for sensor fusion using the Kalman filter as the methodology is unprecedented in the CIIPS mobile robot lab. An indirect Kalman filter configuration combining a piezo rate gyroscope sensor and an inclinometer is implemented to obtain an accurate estimate of the tilt angle and its derivative.

The instability of inverted pendulum systems has always been an excellent test bed for control theory experimentation. Therefore it is also the aim of this thesis to investigate the suitability and examine the performance of linear control systems like the Linear Quadratic Regulator and Pole-placement controller in stabilising the system.

This thesis follows the structure of the author's approach in achieving the final goal of a two-wheeled autonomous balancing robot. The first and second chapter provides an introduction to the existing technology available in balancing such system. Chapter three of this thesis gives the reader an overview on the construction of the robot. The theory and design of a Kalman filter is presented in the fourth chapter followed by the dynamic model of the system given in chapter 5. Control systems designed for the balancing robot is designed and simulated before applying the gains result in the real robot for real time simulation, this is presented in chapter 6 and 7. Chapter 8 summarises the whole project and provides and outlook on the future of the project.

Nomenclature

The following are the list of variables used in this thesis:

x - process state vector.

u - piecewise continuous deterministic control input functions

F - system dynamics matrix.

B - deterministic input matrix.

G - noise input matrix.

z - discrete time measurement process.

H - measurement matrix.

v - discrete time white Gaussian noise.

θ *– Angle*

$\dot{\theta}$ *– Angular velocity*

β *– Gyroscope bias*

γ_r *– Real velocity*

γ_m *– Gyroscope measured velocity*

w - Process noise

τ_m *- Motor torque (Nm).*

τ_e *- Applied torque (Nm).*

R – Nominal terminal resistance (Ohms).

L – Rotor inductance (H).

k_f *– Frictional constant (Nms/rad).*

k_m *– Torque constant (Nm/A).*

K_e *– Back emf constant(Vs/rad).*

θ *- Angular position of shaft (rad).*

ω *- Angular velocity of shaft (rad/s).*

α *- Angular acceleration of shaft (rad/s^2).*

V_a *– Applied terminal voltage (V).*

V_e *– Back emf voltage (V).*

i – Current through armature (A).

I_R *– Rotor inertia (kgm^2).*

τ_m *- Motor torque (Nm).*

M_w *– mass of the wheel connected to both sides of the robot.*

M_p *– mass of the robot's chassis.*

I_w *– moment of inertia of the wheels.*

I_p *– moment of inertia of the robot's chassis.*

H_L, H_R, P_L, P_R *– reaction forces between the wheel and chassis.*

l – distance between the centres of the wheel and the robot's centre of gravity.

C_L, C_R *– applied torque from the motors to the wheels.*

H_{fL}, H_{fR} *– friction forces between the ground and the wheels.*

θ_w *- rotation angle of the wheels.*

θ_P *- rotation angle of the chassis.*

k_m *– Torque constant ($N \cdot m/A$)*

k_e *– Back emf constant ($V \cdot s/rad$)*

R - Nominal terminal resistance (Ω)

V_a *– Applied terminal voltage (V)*

$\dot{\theta}_w$ *– Angular velocity of wheel (rad/s)*

R(t) – Output.

K_p *– Proportional gain.*

K_i *– Integral gain.*

K_d *– Derivative gain.*

e(t) – Error function

Table of Content

1 Introduction

The research on balancing robot has gained momentum over the last decade in a number of robotics laboratories around the world. This is due to the inherent unstable dynamics of the system. Such robots are characterised by the ability to balance on its two wheels and spin on the spot. This additional manoeuvrability allows easy navigation on various terrains, turn sharp corners and traverse small steps or curbs. These capabilities have the potential to solve a number of challenges in industry and society. For example, a motorised wheelchair utilising this technology would give the operator greater manoeuvrability and thus access to places most able-bodied people take for granted. Small carts built utilising this technology allows humans to travel short distances in a small area or factories as opposed to using cars or buggies which is more polluting.

In collaboration with the Centre of Intelligent Information Processing System (CIIPS), a balancing robot is built as a platform to investigate the use of a Kalman filter for sensor fusion. The Kalman filter approach to sensor fusion is unprecedented in the CIIPS mobile robot laboratory. This would be a new avenue to explore the filter for future potential applications of the Kalman filter.

Apart from the above, this thesis will delve into the suitability and performance of linear state space controllers namely the Linear Quadratic Regulator (LQR) and a Pole-placement controller in balancing the system. The robot utilises a Proportional-Integral-Derivative (PID) controlled differential steering method for trajectory control. A gyroscope and inclinometer is used to measure the tilt of the robot and the encoders on the motors to measure the wheel's rotation.

2 Literature Review

Conducting literature review prior to undertaking research projects is critical as this will provide the researcher with much needed information on the technology available and methodologies used by other research counterparts around the world on the topic. This chapter provides a condensed summary of literature reviews on key topics related to balancing a two-wheeled autonomous robot.

2.1 Balancing Robots

The inverted pendulum problem is not uncommon in the field of control engineering. The uniqueness and wide application of technology derived from this unstable system has drawn interest from many researches and robotics enthusiasts around the world. In recent years, researchers have applied the idea of a mobile inverted pendulum model to various problems like designing walking gaits for humanoid robots, robotic wheelchairs and personal transport systems.

Researchers at the Industrial Electronics Laboratory at the Swiss Federal Institute of Technology have built a scaled down prototype of a Digital Signal Processor controlled two–wheeled vehicle based on the inverted pendulum with weights attached to the system to simulate a human driver. A linear state space controller utilising sensory information from a gyroscope and motor encoders is used to stabilise this system (Grasser et al. 2002).

Figure 2.1: JOE (http://leiwww.efpl.ch/joe)

A similar and commercially available system, 'SEGWAY HT' has been invented by Dean Kamen, who holds more than 150 U.S. and foreign patents related to medical devices, climate control systems, and helicopter design. . The 'SEGWAY HT' is able to balance a human standing on its platform while the user traverses the terrain with it. This innovation uses five gyroscopes and a collection of other tilt sensors to keep itself upright. Only three gyroscopes are needed for the whole system, the additional sensors are included as a safety precaution.

Figure 2.2: SEGWAY.
(www.segway.com)

The uniqueness of these systems has drawn interest from robot enthusiasts. For example, Nbot (figure 2.4), a two-wheeled balancing robot similar to JOE built by David .P Anderson, this robot uses a commercially available inertial sensor and position information from motor encoder to balance the system. Steven Hassenplug has successfully constructed a balancing robot called Legway (figure 2.4) using the LEGO Mindstorms robotics kit. Two Electro-Optical Proximity Detector (EOPD) sensors is used to provide the tilt angle of the robot to the controller which is programmed in BrickOS, a C/C++ like programming language specifically for LEGO Mindstorms.

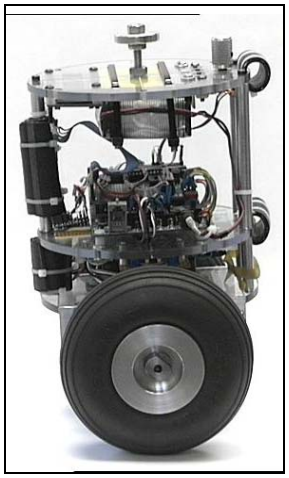


Figure 2.3 & 2.4 : Nbot [left] Legway [right]
(http://www.geology.smu.edu/~dpa-www/robo/nbot/)
(http://perso.freelug.org/legway/LegWay.html)

The paper 'Cooperative Behaviour of a Wheeled Inverted Pendulum for Object Transportation' presented by Shiroma et al. in 1996 shows the interaction of forces between objects and the robot by taking into account the stability effects due to these forces. This research highlights the possibility of cooperative transportation between two similar robots and between a robot and a human.

The rapid increase of the aged population in countries like Japan has prompted researchers to develop robotic wheelchairs to assist the infirm to move around, (Takahashi et al. 2000). The control system for an inverted pendulum is applied when the wheelchair manoeuvres a small step or road curbs.

On a higher level, Sugihara et al. (2002) modelled the walking motion of a human as an inverted pendulum in designing a real time motion generation method of a humanoid robot that controls the centre of gravity by indirect manipulation of the Zero Moment Point (ZMP). The real time response of the method provides humanoid robots with high mobility.

2.2 *Kalman Filter*

In 1960 R.E Kalman published a paper titled 'A New Approach to Linear Filtering and Prediction Problems' (Kalman, 1960). Kalman's research intended to overcome the limitations of the 'Weiner-Hopf' filter in solving problems of statistical nature which seriously curtails its practical usefulness. The process described within came to be known as Kalman filtering.

The Kalman filter is a set of mathematical equations that provides an efficient computational solution of the least squared method. The filter is very powerful as it supports estimations of past, present and even future states, and it can do so even when the precise nature of the modelled system is unknown.

There have been a number of texts written about Kalman filtering since Kalman's original paper. One of the complications in understanding the filter methodology is that there seems to be a lack of standard notation for the filter equations. It is possible that the notations used by Kalman are too complex. Therefore, successive authors have

simplified notation and expressed filter equations in different ways. This result in none of the books used identical notation and authors of academic papers which uses the Kalman filter usually follows the notation used by author's reference.

As the subject of Kalman filtering was relatively new at that period of time, the lack of continuity in these books and the different notation used in some chapters have hindered the understanding of the material presented. Books by Gelb (1974) and Maybeck (1979) provided a comprehensive explanation on Kalman filtering in a very practical way. Although both books provided practical aspects of implementing the Kalman filter, Maybeck's complete work which consists of three volumes are superior and more modern than Gelb's as it covers a broader range of topics in more depth. Maybeck's book is aimed at engineers and could serve as a reference on the topic of stochastic estimation.

A comprehensive description of the Kalman filter can be found in Chapter 4. This chapter adopts the notation used by Maybeck in his book. Kalman Filtering allows a best estimate of a system's future state despite inaccuracy in its measurements and unpredictable changes in that system.

2.3 Sensor Fusion Using Kalman Filter

The accuracy and reliability of information regarding its operating environment for mobile robots is critical as these systems are usually unmanned. These requirements call for highly accurate sensors which are very expensive. Sensor fusion technology, where signal from several sensors are combined to provide an accurate estimate is the most widely used solution.

The Kalman filter is used in a number of multisensor systems when it is necessary to combine dynamic low-level redundant data in real time. The filter uses the statistical characteristics of a measurement model to recursively determine estimates for fused data that are optimal in a statistical sense. The recursive nature of the filter makes it appropriate for use in systems without large data storage capabilities.

In 1993, Barshan & Durrant-Whyte published a paper on Inertial Navigation Systems for Mobile Robots. This pioneering research formed the building block for many other research applying sensor fusion technologies for mobile robot applications. The paper evaluated the performance of several sensors used in Inertial Navigation Systems and provided an outline for developing an Extended Kalman Filter for such systems.

The wide applicability of sensor fusion technology has inspired the use in numerous configurations. Borenstein & Feng (1996) developed 'Gyrodometry', which uses the Kalman filter for combining data from gyroscopes and odometry in mobile robots. This method effectively reduces odometry error which usually occurs when wheels of the robot slips on slippery surfaces. In a similar development, Komoriya & Oyama (1994) utilises the Kalman filter to combine velocity information from the Optical Fibre Gyroscope with the position information obtained from the motor encoders for an optimal trajectory control of mobile robots.

2.4 Control Systems

Control system development is vital to guarantee the success in balancing the robot, while there is abundance of control strategies that can be applied to stabilise the robot, the main aim is to control the system cheaply and effectively without sacrificing the robustness and reliability of the controller. The difference in balance control algorithm implemented depends mostly on how the system is modelled and how the tilt information is obtained. Nevertheless, a common approach separates the balancing and trajectory control of the mobile inverted pendulum.

The control strategies for such system can be divided into two distinct sections, namely a linear control model or a nonlinear controller model. Linear control methods often linearise the dynamics about a certain operating point. This method is usually sufficient in balancing the system. A nonlinear controller uses the unscathed dynamics model of the system in designing a controller. Although these controllers would provide a more robust system, the complexity and implementation difficulties of these methods results in most researchers utilising the linear controller approach.

A literature review found that nonlinear controllers are mostly implemented in solving the balance control problem of a simple pendulum on a cart model or a rotary inverted pendulum. Tarek et al. (1994) developed a Fuzzy Logic controller for balancing an inverted pendulum on a cart. This approach is based on approximate reasoning and knowledge based control. Williams & Matsuoka (1991) used the inverted pendulum model to demonstrate the ability of Neural Networks controller in controlling nonlinear unstable systems.

While simulation results proved that the system can be balanced with both controllers, there is no evidence of implementation of these ideas to verify their findings. Doskocz, Shtessel & Katsinis (1998) implemented a multi-input, multi-output sliding mode controller for a pick and place robotic arm modelled as an inverted pendulum. The researches mentioned above are predominantly purpose built and non-mobile. One of the reasons for that is nonlinear controllers usually requires high computational power.

The linear controllers are more popular among researcher designing similar balancing robots like JOE (Grasser et al, 2002). Linear state space controllers like the Pole-placement controller and the Linear Quadratic Regulators (LQR) are the two most popular control system implemented. The implementation of these controllers can be seen in papers published by Nakajima et al. (1997), Shiroma et al. (1996), Takahashi et al. (2000) and Grasser et al (2002).

In the research titled 'Comparative Study of Control Methods of Single – Rotational Inverted Pendulum' conducted by Xu & Duan (1997) showed that the LQR controller fared better than the pole placement controller in balancing an inverted pendulum mounted on a rotation arm. This is because the LQR controller offers an optimal control over the system's input by taking the states of the system and the control input into account. The arbitrary placement of control poles for Pole-placement controllers might cause the poles to be placed too far into the left-hand plane and cause the system susceptible to disturbances.

3 The Balancing Robot System

The balancing robot system is built as part of the thesis requirement for Kalman filter experimentation and to test out the performance of linear state space controllers in balancing an unstable system. The design of the system is kept as simple as possible but not compromising the aim of the project.

3.1 Robot Chassis

The robot's chassis design is based on a stack of 130mm x 130mm Perspex plates with the components placed in between the spacings of each plate. This simple design enables quick installation of component and the height of the robot can be increased or lowered when desired.

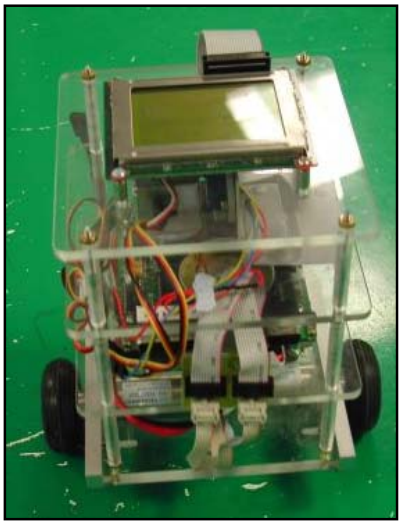


Figure 3.1: The Balancing Robot

The drive train of the robot resembles a two wheeled differential drive robot, but the balancing robot balances the load with its wheels instead dragging the weight around on a pivot in a regular differential drive robot.

The motors are fitted into the socket like motor mounts machined from aluminium. This provides a stronger grip on the motors apart from the three screw holes on the motor to avoid misalignment of the motors. The motor mounts are designed to have a maximum tilt angle of 30 degrees when the wheels are affixed; this is due to the limited measuring range of the sensors.

The Perspex plates are stacked 40mm apart from each other with acrylic tubes and four steel shafts with threading at the ends put through these tubes. The whole structure is held together with nuts both ends of the shaft. Tail-wheels from remote controlled planes bought from a local hobby shop is used as the wheels of the robot. As the original hub of the wheels are made from plastic, new aluminium hubs has to be machined in order for the wheels to fit onto the shaft.

3.2 Controller

A Mark 4 Eyebot controller running on Robios version 5.2 is used as the 'brain' of the balancing robot system. The controller consists of a powerful 32-Bit microcontroller running at 33MHz, there is 512k ROM and 2048k RAM onboard. This allows relatively large and computing intensive programs to be executed quite easily. The programming language used in this controller is C and Assembly language.

Figure 3.2: The Eyebot controller. (http://robotics.uwa.edu.au/eyebot)

3.3 Actuators

The actuation to balance the robot is via two Faulhaber DC motors made by Faulhaber Gmbh, Germany. Each motor has a gear reduction of 54.2:1 and a torque constant of 6.9203 x 10^{-4} kg-m/A. This type motor is widely used by the UWA Mobile Robot Lab.

3.4 Sensors

A HITEC GY-130 digital rate gyroscope and a SEIKA N3 digital inclinometer are installed on the balancing robot system to measure the tilt angle of the robot as well as the angular velocity.

The gyroscope returns the instantaneous angular velocity by requiring a 50% control sent to it via a Timing Processing Unit (TPU) channel and an altered signal is read back by another TPU channel. The instantaneous angular velocity is obtained by measuring the difference between the altered signal and the 50% control signal.

Figure 3.3: Hitec GY-130 Gyroscope. (www.hitecrcd.com)

The inclinometer is a capacitive, liquid based sensor that generates a pulse with modulated output relating to the absolute inclination of the system. The output of the inclinometer is quite linear with the measured angle but unfortunately this sensor has an effective measuring range of only ± 30 degrees.

Figure 3.4: Seika N3 Inclinometer.(www.seika.de)

Besides the gyroscope and the inclinometer, the absolute encoders built into the motors are used to provide information on the robot's movement as well as the speed at which it is travelling. These encoders are capable of providing 3495 counts per shaft revolution.

4 Sensor Fusion & Kalman Filtering

This chapter provides the details on the experiments conducted on the sensors used in this project. Problems related to the sensors were discovered, which leads to the implementation of the Kalman filter.

4.1 Sensor Processing

Both the gyroscope and inclinometer are tested using the servo test platform, similar to the one used for the tracked robot project, (C.M. Smith, 2002). Under this set-up, a test program is written to rotate the sensors through a number of set points. The actual angle of inclination can be obtained based upon the control signal sent to the servo. From there, a comparison is done between the actual angle of the servo rotation and the angle recorded by the sensors. This test platform ensures that the sensors manipulation is as exact as possible.

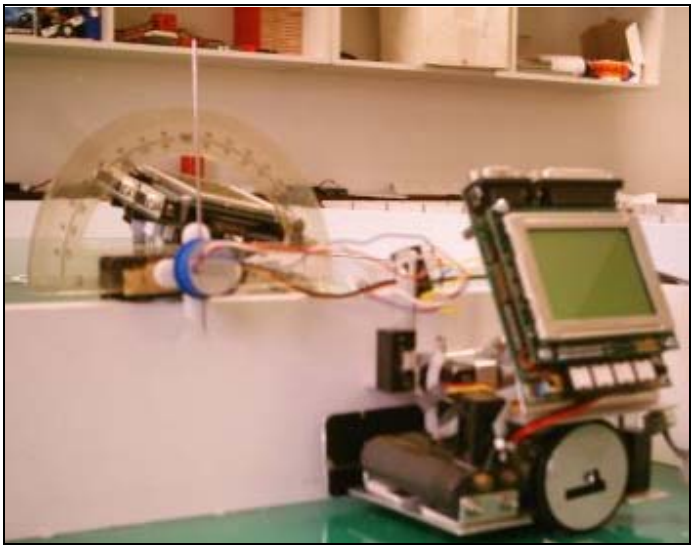


Figure 4.1: Servo test platform.

The digital rate gyroscope installed could only provide a measure of the instantaneous angular change. Furthermore, the gyroscope has a rest average -value (value of the gyroscope when it is not moving) which has to be offset at every measurement to get an accurate velocity measurement. It is found that the gyroscope's rest average value drifts with time. This will introduce significant errors in the velocity and angular measurements.

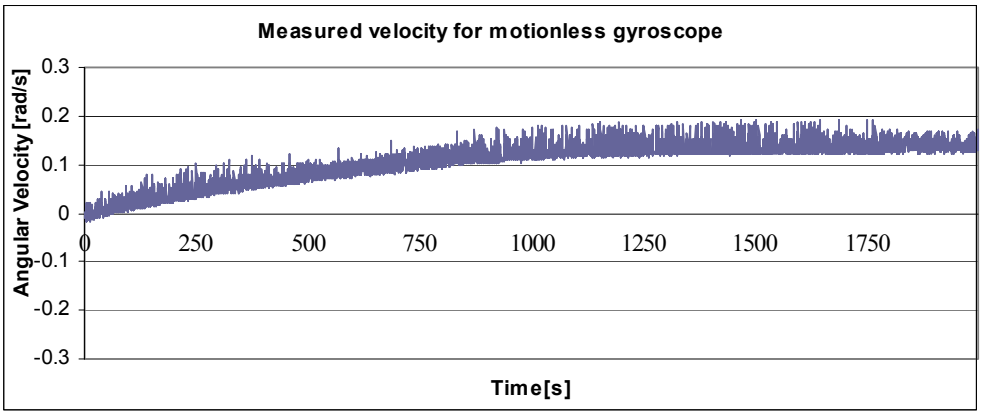


Figure 4.2: Gyroscope velocity value drift.

The liquid based digital inclinometer installed on the robot gives a good measure of absolute tilt angle. The inclinometer data is processed almost the same way as the gyroscope. Several problems observed while testing the inclinometer are the rest value of the inclinometer changes every time when it initialised and the output signal tends to exhibits a significant amount of noise.

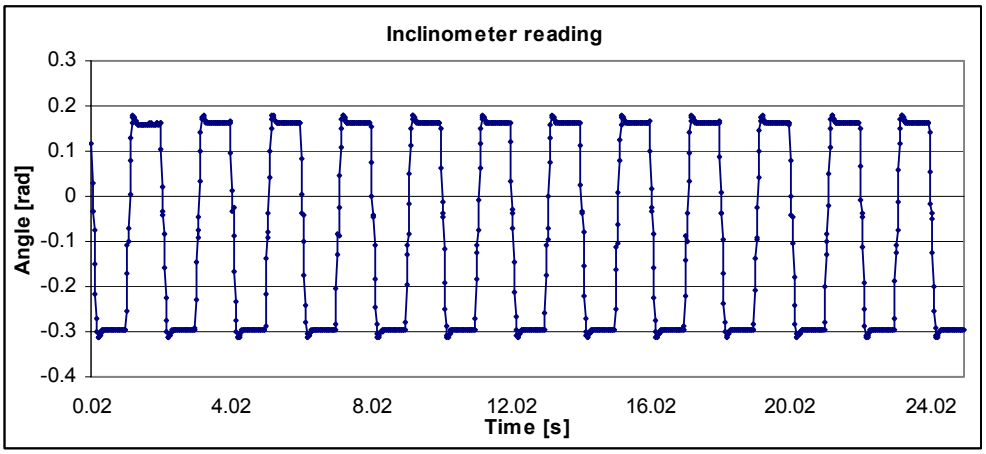


Figure 4.3: Noisy inclinometer reading.

4.2 The Need for Sensor Fusion

Initial tests performed above on the gyroscope and the inclinometer showed that the use of either one of the sensors is unable to provide sufficient and more importantly reliable information in order to balance the robot.

The gyroscope provides a measure of instantaneous angular change but it produces a significant drift when the gyroscope is operating. This may be due to the operating temperature or inherent characteristics of the gyroscope itself. On the other hand, the inclinometer provides an absolute measure of inclination, but the output signal is often corrupted with noise.

To overcome these problems, a signal-level sensor fusion technique using the Kalman filter is proposed. Signal-level fusion refers to the combination of signals of a group of sensors with the objective of providing a signal that is usually of the same form as the original signals but of greater quality. In this case, the inclinometer is used to eliminate the drift from the gyroscope signal via the Kalman filter. As a result, an accurate estimate of the angle and its derivative term is obtained.

4.3 The Kalman Filter

The Kalman filter is a set of mathematical equation for optimal recursive data processing algorithm that provides the solution of the least squares method. It incorporates all information that can be provided and processes all available measurements, regardless of their precision to estimate the current value of the variables of interest with the use of a knowledge of the system and measurement dynamics, the statistical description of the system noises, measurement errors, uncertainty in the model dynamics, and any available information about the initial condition of the variables of interest.

Under the assumptions that the model for system of interest is linear and the noise values are not correlated in time, the Kalman filter is optimal effectively to any criterion that makes sense.

The Kalman filter does not require all previous data to be kept in storage and reprocessed every time a new measurement is taken. With this behaviour, the Kalman filter is be implemented as a computer program in a small microcontroller despite the usual implication that a filter is a connection of electrical networks in a box.

4.4 The Discrete Kalman Filter Algorithm

4.4.1 State Equation

The Kalman filter can be applied to estimate the states when the system of interest is adequately modelled in the form of a linear stochastic differential equation. In other words, the Kalman filter can be utilised to estimate the states of a system when the system of interest can be modelled in state space form. Equation (4.1) represents the state process of the system in white noise notation.

$$\dot{x} = F(t)x(t) + B(t)u(t) + G(t)w(t) \tag{4.1}$$

4.4.2 Measurement Equation

Measurements for the system is taken at discrete time points, these measurements are included in the measurement vector and can be modelled by the relation,

$$z(t_i) = H(t_i)x(t_i) + v(t_i) \tag{4.2}$$

Equation (4.2) states that the measurements are dependent on the state of the system and are related by the measurement matrix with an addition of noise into the measurements. The measurements for the system are often obtained at equally spaced time, but this is not obligatory.

4.4.3 System and Measurement Noise

It is assumed that the noise vector *w(t)* and *v(t)* are white Gaussian noise with the following statistics:

$$E\{w(t)\} = 0 \tag{4.3}$$

$$E\{w(t)w(t')^T\} = Q(t)\partial(t - t') \tag{4.4}$$

and

$$E\{v(t_i)v^T(t_i)\} = \begin{cases} R(t_i) & ,t_i = t_j \\ 0 & ,t_i \neq t_j \end{cases} \tag{4.5}$$

$R(t_i)$ is a positive definite matrix, which means that all components of the measurement vector are corrupted with noise, and there is no linear combination of these components that would be noise free. Since the distributions of $w(t)$ and $v(t)$ are assumed Gaussian, this is equivalent to assuming that they are uncorrelated with each other.

4.4.4 Initial Conditions

The state differential equation (4.1) is propagated from the initial condition $x(t_o)$ and for any particular operation of the real system, the initial state assumes a specific value $x(t_o)$. However, because this value may not be known precisely in advance, it would be modelled as a random vector that is normally distributed. Thus, the description of $x(t_o)$ is completely specified by the mean $\hat{x}_o$ and covariance P_o.

$$E\{x(t_o)\} = \hat{x}_o \tag{4.6}$$

$$E\{[x(t_o) - \hat{x}_o][x(t_o) - \hat{x}_o]^T\} = P_o \tag{4.7}$$

P_o is a symmetric and positive semi-definite matrix. This matrix provides the expected value of the difference between the true state and the estimated state. The diagonal elements of this matrix provide the variance of each state variable from its true value.

4.4.5 Kalman Filter Equations

The Kalman filter estimates a process by using a form of feedback control. The states of the process is estimated by the filter at a certain point and then obtains a feedback of noisy measurements. Therefore, the equations for the Kalman filter fall into two groups, the time update equations and measurement update equations.

The current state and error covariance estimates propagated by the time update equations to obtain the a priori estimates for the next time step. On the other hand, the measurement update equations are responsible for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update

equations are modeled as the predictor equations, while the measurements update equations taken as the corrector equations. This concept is illustrated in figure (4.4).

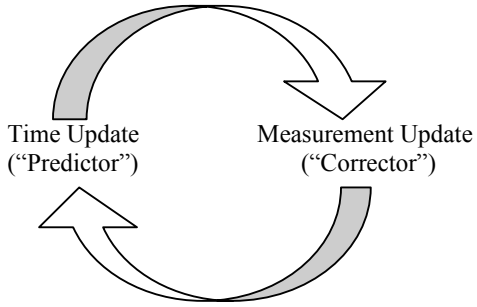


Figure 4.4: Kalman Filter's "Predictor-Corrector" structure.

Considering two measurement times, t_{i-1} and t_i, and time propagation estimates from the point just after the measurement at time t_{i-1} has been incorporated into the estimate, to the point after the measurement at time t_i is incorporated. This is represented by time t_{i-1}^{+} to time t_i^{+}. The optimal state estimate is propagated from measurement time t_{i-1} to measurement time t_i by the relations.

$$\hat{x}(t_i^{-}) = F\hat{x}(t_{i-1}^{+}) + Bu(t_{i-1}^{+}) \tag{4.8}$$

$$P(t_i^{-}) = FP(t_{i-1}^{+})F^T + Q \tag{4.9}$$

Equation (4.9) defines the conditional covariance matrix of the error is predicting the state *x.* As the state measurement z_i becomes available at time t_i, the estimate is updated by defining the Kalman filter gain $k(t_i)$ and employing it in both the mean and covariance relations.

$$k(t_i) = P(t_i^{-})H^T(t_i)[H(t_i)P(t_i^{-})H^T(t_i) + R(t_i)]^{-1} \tag{4.10}$$

$$\hat{x}(t_i^{+}) = \hat{x}(t_i^{-}) + k(t_i)[z_i - H(t_i)\hat{x}(t_i^{-})] \tag{4.11}$$

$$P(t_i^{+}) = P(t_i^{-}) - k(t_i)H(t_i)P(t_i^{-}) \tag{4.12}$$

The error estimate, that is the errors committed by the estimator $\hat{x}(t_i^-)$ for a particular measurement is given by,

$$E = H(t_i)P(t_i^-)H^T(t_i) + R(t_i)\,. \tag{4.13}$$

Alternatively, equation (4.10) can be rewritten as

$$k(t_i) = P(t_i^+)H^T(t_i)E^{-1}(t_i^-) \tag{4.14}$$

The difference between the true measurement z_i and its best prediction before it is actually taken is known as the measurement residual $r(t_i)$. This is defined by the following equation

$$r(t_i) = z_i - H(t_i)\hat{x}(t_i^-)\,. \tag{4.15}$$

The residual is then passed through an optimal weighing matrix $k(t_i)$ to generate the correction term to be added to $\hat{x}(t_i^-)$ to obtain $\hat{x}(t_i^+)$.

4.5 The Extended Kalman Filter

The Extended Kalman Filter provides a method applying the Kalman Filter technique to a nonlinear problem by linearising the estimation around the current estimate using partial derivatives.

4.5.1 State Equation

For the Extended Kalman Filter, the process are governed by the non-linear stochastic difference equations

$$\dot{x} = f(x) + w \tag{4.16}$$

where *x* is a vector of the system states and *f(x)* is a nonlinear function of those states.

4.5.2 Measurement Equation

The measurement equation for the Extended Kalman Filter are considered a non-linear function of the states according to

$$z = h(x) + v \tag{4.17}$$

4.5.3 System and Measurement Noise

The system and measurement noise v and w for the Extended Kalman Filter are modelled as a random process with zero mean, which is similar to the system and measurement noise statistics modelled for a normal Kalman filter.

4.5.4 Kalman Filter Equations

In order to apply the continuous Ricatti equation, the non-linear system and measurement equations are linearised with a first order approximation using the Jacobian matrix.

- F is the Jacobian matrix of partial derivatives of f with respect to x, that is

$$F = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}} \tag{4.18}$$

- H is the Jacobian matrix of partial derivatives of h with respect to x, that is

$$H = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}} \tag{4.19}$$

Now the optimal state estimate propagation from measurement time t_{i-1} to measurement time t_i can be represented with the following equations

$$\hat{x}(t_i^-) = f(x(t_{i-1}^+)) \tag{4.20}$$

$$P(t_i^-) = FP(t_{i-1}^+)F^T + Q \tag{4.21}$$

As before, when the state measurement z_i becomes available at time t_i, the estimate is updated by the following equations

$$k(t_i) = P(t_i^-)H^T(t_i)[H(t_i)P(t_i^-)H^T(t_i) + R(t_i)]^{-1} \tag{4.22}$$

$$\hat{x}(t_i^+) = \hat{x}(t_i^-) + k(t_i)[z_i - H(\hat{x}(t_i^-))] \tag{4.23}$$

$$P(t_i^+) = P(t_i^-) - k(t_i)H(t_i)P(t_i^-) \tag{4.24}$$

The linearization of the process will be valid and will produce excellent results if the estimate of the states is good. In cases where the state estimates are not good, the filter will diverge quickly and produce a very poor estimate.

4.6 Filter Tuning for Performance

The filter's performance could vary greatly if the parameters are not properly adjusted. Therefore, this section intends to provide some insight into the characteristics of the filters parameters. The adjustable parameters are:

1. The initial covariance matrix.
2. The state estimate vector.
3. The Q matrix and its corresponding random noise vector **w**.
4. The R matrix and its corresponding random noise vector **v**.

4.6.1 Initial Error Covariance Matrix

The expected variance of the error of the state estimate for the corresponding parameter is represented by the diagonal elements of this matrix. For the filter to work correctly, the values in the covariance matrix must be defined such that the difference between the

initial state and the initial state estimate fall in the range that is allowable according to the covariance matrix.

The diagonal element of the initial covariance matrix needs to be large enough to meet the aforementioned requirement. If the initial state estimate is quite accurate, then the covariance matrix needs only allow for a small error. The state estimate will take a longer time to converge as the covariance matrix gets larger. A different P_o matrix will yield a different magnitude transient characteristic, but its duration will be the same and the steady state conditions are unaffected.

4.6.2 Initial State Estimator

An initial estimate of the system must be available for the filter loop to be executed. There is no general requirement for the initial estimate to be accurate, provided that the initial covariance matrix values are sufficiently large for the filter to work correctly.

4.6.3 Q Matrix

The Q matrix represents the covariance of the system error vector w. It is assumed that the elements of w are uncorrelated; therefore the value of every element of Q which does not lie on the diagonal is zero.

Increasing Q would indicate either stronger noises driving the dynamics or increased uncertainty in the adequacy of the model itself to depict the true dynamics accurately. This will increase both the rate of growth of the $P(t)$ elements or eigenvalues between measurement times and their steady state values. As a result, the filter gains will generally increase and the measurements are weighted heavily, this is reasonable since increased Q dictates that we should put less confidence in the output of the filter's own dynamics model.

4.6.4 R Matrix

The R matrix represents the covariance of the measurement error matrix, v. This matrix indicates how large the measurement errors are expected to be. Increased R would indicate that the measurement are subjected to a stronger corruptive noise and so should be weighted less by the filter.

The filters steady state operation is quickly reached if the eigenvalues of Q are large compared to the eigenvalues of R (Q/R ratio is high). This is due to the uncertainty involved in the state propagation is large compared to the accuracy of the measurement, so the new state estimate is heavily dependent upon the new measurement and not closely related to prior estimates.

4.7 *Kalman Filter Design for Single Dimensional INS*

In this thesis, a Kalman filter is designed to provide an estimate of tilt angle and its derivative for a single dimensional Inertial Navigation System (INS). This filter will attempt to use the data from an inclinometer to eliminate the measurement drift from the gyroscope signal via the Kalman filter. In the process, corruptive noises in the inclinometer reading will also be minimised.

An *indirect feedback* configuration will be used for the single dimensional Inertial Navigation System. This configuration is illustrated in the figure below,

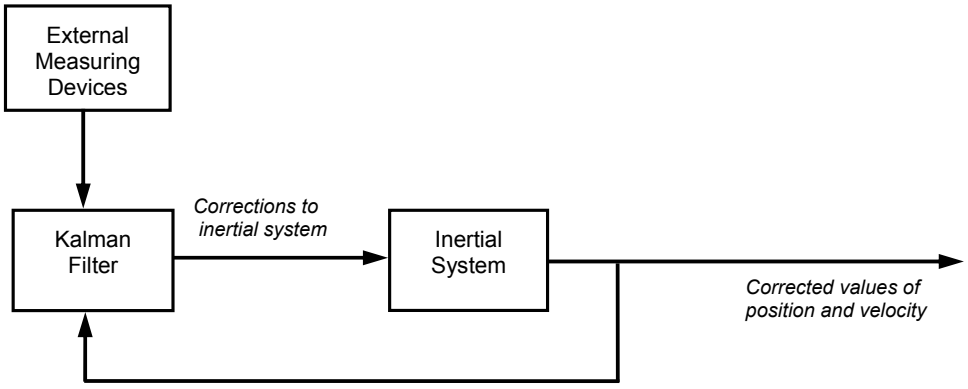


Figure 4.5: Indirect feedback Kalman filter [Maybeck, 79].

The filter compares the data from the sensors and the inertial system and uses this result to estimate the errors in the system. By feeding back these error estimates to the INS to correct it, the inertial errors are not allowed to go unchecked, this way the adequacy of the model is enhanced.

4.7.1 Process Model

For the single dimensional INS, a two state Kalman filter is implemented to track the tilt angle of the balancing robot and the gyroscope bias. The state equation is quite simple as we are only interested in the process where the gyroscope bias is removed from the readings. The process is depicted in the figure below,

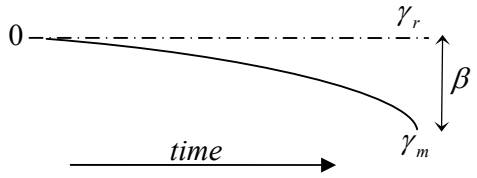


Figure 4.6: Process model depicting gyroscope drift.

The gyroscope measured angle will deviate from its zero set point as time increases. The process model written in the form of equation (4.1),

$$\begin{bmatrix}\dot{\theta}\\ \dot{\beta}\end{bmatrix}=\begin{bmatrix}0 & -1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \beta\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\gamma_m+\begin{bmatrix}1\\ 0\end{bmatrix}w \tag{4.25}$$

The derivative of the gyroscope bias is zero because it is only a scalar value. By using the principle of superposition, the process noise can be included into the gyroscope velocity measurement. Thus the new process model is now,

$$\begin{bmatrix}\dot{\theta}\\ \dot{\beta}\end{bmatrix}=\begin{bmatrix}0 & -1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \beta\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\gamma_m \tag{4.26}$$

The system model developed in this section is used to implement the Kalman filter for a single dimensional INS. The experimental result of this implementation could be found in chapter 7.

5 System Modelling

The dynamics of the robot has to be described by a mathematical model in order to facilitate the development of an efficient control system for the balancing robot. In this chapter, the equation of motion for a two-wheeled inverted pendulum and linear model for a DC motor is derived in detail.

5.1 Linear model of a direct current (DC) motor

The robot is powered by two Faulhaber DC motors. In this section the state space model of the DC motor is derived. This model is then used in the dynamic model of the balancing robot to provide a relationship between the input voltage to the motors and the control torque needed to balance the robot.

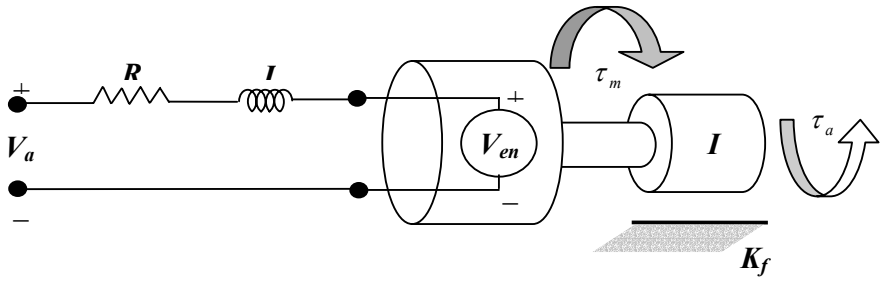


Figure 5.1: Diagram of a DC motor

Figure 5.1 exemplifies an effective linear model for a direct current motor. When a voltage is applied to the terminals of the motor, a current ***i,*** is generated in the motor armature. The motor produces a torque t_m, which is proportional to the current. This relationship can be expressed as.

$$\tau_m = k_m i \tag{5.1}$$

A resistor-inductor pair in series with a voltage, V_{emf} , can be used to model the electrical circuit of the motor. This *back electromotive force voltage* is produced because the coils of the motor are moving through a magnetic field. The voltage produced can be approximated as a linear function of shaft velocity, which can be written as

$$V_e = k_e \omega \tag{5.2}$$

At this point, a linear differential equation for the DC motor's electrical circuit can be written by using *Kirchoff's Voltage Law*, the law states that the sum of all voltages in the circuit must equal to zero. For the DC motor, this can be written as

$$V_a - Ri - L\frac{di}{dt} = 0 \qquad (5.3)$$

In deriving the equation of motion for the motor, the friction on the shaft of the motor is approximated as a linear function of the shaft velocity. The approximation that the friction on the shaft of the motor k_f, is a linear function of the shaft velocity is made. Newton's Law of Motion states that the sum of all torques produced on the shaft is linearly related to the acceleration of the shaft by the inertial load of armature. $\boldsymbol{I_R}$. The preceding statement can be written as

$$\sum M = \tau_m - k_f\omega - \tau_a = I_R\omega \qquad (5.4)$$

Substituting equation (5.1) and (5.2) into equations (5.3) and (5.4), and rearranging in terms of the time derivatives, leads to the following two fundamental equations which governs the motion of the motor. Both

$$\frac{di}{dt} = \frac{R}{L}i + \frac{k_e}{L}\omega + \frac{V_a}{L} \qquad (5.5)$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}i + \frac{-k_e}{I_R}\omega - \frac{\tau_a}{I_R} \qquad (5.6)$$

Both equations are linear functions of current and velocity and they include the first order time derivatives. A simplified DC motor model is sufficient for the balancing robot case. For that reason, the motor inductance and motor friction is considered negligible and is approximated as zero. Hence, [5] and [6] can be approximated as

$$i = -\frac{k_e}{R}\omega + \frac{1}{R}V_a \qquad (5.7)$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} i - \frac{\tau_a}{I_R} \tag{5.8}$$

By substituting equation (5.7) into equation (5.8), an approximation for the DC motor which is only a function of the current motor speed, applied voltage and applied torque is obtainable.

$$\frac{d\omega}{dt} = -\frac{k_m k_e}{I_R R}\omega + \frac{1}{I_R R} V_a - \frac{\tau_a}{I_R} \tag{5.9}$$

Since the motor inductance is neglected, the current through the windings is not considered in the equation of motion of the motor. The current will then reach a constant state immediately as compared to the velocity of the shaft, which takes time to speed up from some initial speed to a final speed after a change in the input voltage.

The motor's dynamics can be represented with a state space model, it is a system of first order differential equations with parameters position, θ, and velocity, ω, that uniquely represents the its operation. The inputs to the motor is then the applied voltage and applied torque.

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-k_m k_e}{I_R R} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{k_m}{I_R R} & \frac{-1}{I_R} \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix} \tag{5.10}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix} \tag{5.11}$$

5.2 *Dynamic model for a two wheeled inverted pendulum*

The two-wheeled inverted pendulum, albeit more complex in system dynamics, has similar behaviour with a pendulum on a cart. The pendulum and wheel dynamics are analysed separately at the beginning, but this will eventually lead to two equations of motion which completely describes the behaviour of the balancing robot.

As the robot's behaviour can be influenced by disturbances as well as the torque from the motor, the mathematical model will have to accommodate for such forces. Firstly the equations of motion associated with the left and right wheels are obtained. The following figure shows the free body diagram for both wheels. Since the equation for the left and right wheels are completely analogous, only the equation for the right wheel is given.

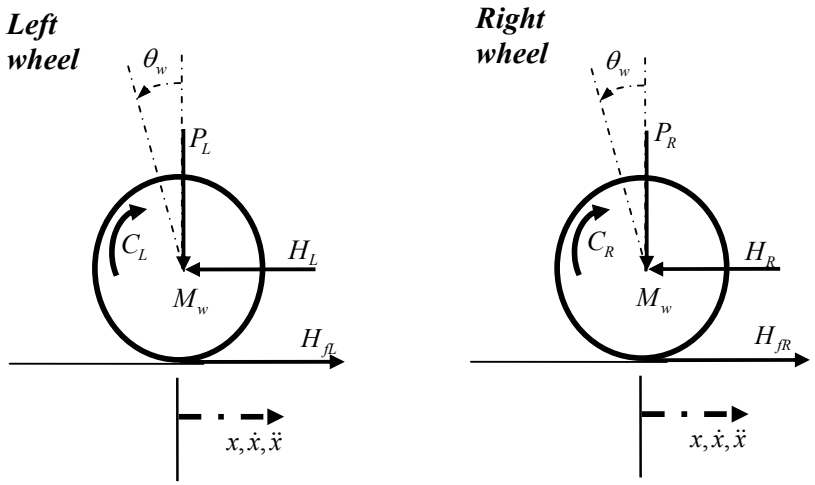


Figure 5.2: Free body diagram of the wheels

Using Newton's law of motion, the sum of forces on the horizontal *x* direction is

$$\sum F_x = Ma$$

$$M_w \ddot{x} = H_{fR} - H_R \qquad (5.12)$$

Sum of forces around the centre of the wheel gives

$$\sum M_o = I\alpha$$
$$I_w\ddot{\theta}_w = C_R - H_{fR}r \tag{5.13}$$

From DC motor dynamics, the motor torque can be expressed as,

$$\tau_m = I_R\frac{d\omega}{dt} + \tau_a \tag{5.14}$$

By rearranging the equation and substituting the parameters from the DC motor derivation section, the output torque to the wheels is attained

$$C = I_R\frac{d\omega}{dt} = \frac{-k_m k_e}{R}\dot{\theta}_w + \frac{k_m}{R}V_a \tag{5.15}$$

Therefore, equation (5.13) becomes

$$I_w\ddot{\theta}_w = \frac{-k_m k_e}{R}\dot{\theta}_w + \frac{k_m}{R}V_a - H_{fR}r \tag{5.16}$$

Thus,

$$H_{fR} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w}{r}\ddot{\theta}_w \tag{5.17}$$

Equation (5.15) is substituted into (5.12) to get the equation for the left and right wheels

For the left wheel

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w}{r}\ddot{\theta}_w - H_L \tag{5.18}$$

For the right wheel

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w}{r}\ddot{\theta}_w - H_R \qquad (5.19)$$

Because the linear motion is acting on the centre of the wheel, the angular rotation can be transformed into linear motion by simple transformation,

$$\ddot{\theta}_w r = \ddot{x} \quad \Rightarrow \ddot{\theta}_w = \frac{\ddot{x}}{r}$$
$$\dot{\theta}_w r = \dot{x} \quad \Rightarrow \dot{\theta}_w = \frac{\ddot{x}}{r}$$

By the linear transformation, equation (5.18) and (5.19) becomes:

For the left wheel,

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr^2}\dot{x} + \frac{k_m}{Rr}V_a - \frac{I_w}{r^2}\ddot{x} - H_L \qquad (5.20)$$

For the right wheel,

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr^2}\dot{x} + \frac{k_m}{Rr}V_a - \frac{I_w}{r^2}\ddot{x} - H_R \qquad (5.21)$$

Adding equation (5.20) and (5.21) together yields,

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{-2k_m k_e}{Rr^2}\dot{x} + \frac{2k_m}{Rr}V_a - \left(H_L + H_R\right) \qquad (5.22)$$

The robot's chassis can be modelled as an inverted pendulum, figure 5.3 shows the free body diagram of the chassis.

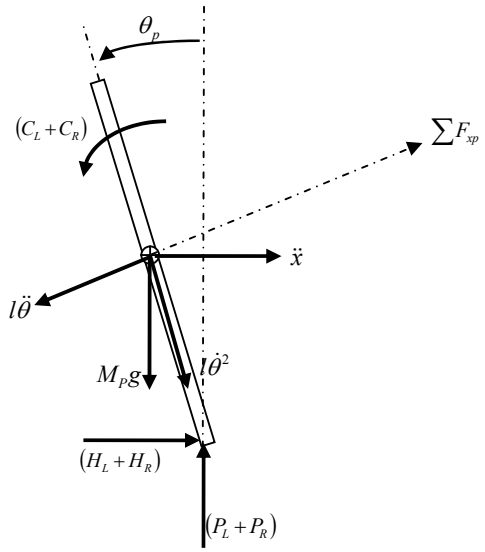


Figure 5.3: Free body diagram of the chassis

Again, by using Newton's law of motion, the sum of forces in the horizontal direction,

$$\sum F_x = M_p \ddot{x}$$
$$(H_L + H_R) - M_p l\ddot{\theta}_p \cos\theta_p + M_p l\dot{\theta}^2{}_p \sin\theta_p = M_p \ddot{x} \quad (5.23)$$

thus,

$$(H_L + H_R) = M_p \ddot{x} + M_p l\ddot{\theta}_p \cos\theta_p - M_p l\dot{\theta}^2{}_p \sin\theta_p \quad (5.24)$$

The sum of forces perpendicular to the pendulum,

$$\sum F_{xp} = M_p \ddot{x} \cos\theta_p$$
$$(H_L + H_R)\cos\theta_p + (P_L + P_R)\sin\theta_p - M_p g \sin\theta_p - M_p l\ddot{\theta}_p = M_p \ddot{x} \cos\theta_p \quad (5.25)$$

Sum of moments around the centre of mass of pendulum,

$$\sum M_o = I\alpha$$
$$-(H_L + H_R)l\cos\theta_p - (P_L + P_R)l\sin\theta_p - (C_L + C_R) = I_p\ddot{\theta}_p \quad (5.26)$$

The torque applied on the pendulum from the motor as defined in equation (5.15) and after linear transformation,

$$C_L + C_R = \frac{-2k_m k_e}{R}\frac{\dot{x}}{r} + \frac{2k_m}{R}V_a$$

Substituting this into equation (5.26) gives,

$$-(H_L + H_R)l\cos\theta_p - (P_L + P_R)l\sin\theta_p - \left(\frac{-2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a\right) = I_p\ddot{\theta}_p$$

thus,

$$-(H_L + H_R)l\cos\theta_p - (P_L + P_R)l\sin\theta_p = I_p\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a \quad (5.27)$$

Multiply equation (5.25) by $-l$,

$$\left[-(H_L + H_R)l\cos\theta_p - (P_L + P_R)l\sin\theta_p\right] + M_p gl\sin\theta_p + M_p l^2\ddot{\theta}_p = -M_p l\ddot{x}\cos\theta_p \quad (5.28)$$

Substitute equation (5.27) in equation (5.28),

$$I_p\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a + M_p gl\sin\theta_p + M_p l^2\ddot{\theta}_p = -M_p l\ddot{x}\cos\theta_p \quad (5.29)$$

To eliminate $(H_L + H_R)$ from the motor dynamics, equation (5.24) is substituted into equation (5.22),

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{-2k_m k_e}{Rr^2}\dot{x} + \frac{2k_m}{Rr}V_a - M_p\ddot{x} - M_p l\ddot{\theta}_p\cos\theta_p + M_p l\dot{\theta}^2{}_p\sin\theta_p \quad (5.30)$$

Rearranging equations (5.29) and (5.30) gives the non-linear equations of motion of the system,

$$\left(I_p + M_p l^2\right)\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a + M_p g l \sin\theta_p = -M_p l \ddot{x}\cos\theta_p \tag{5.31}$$

$$\frac{2k_m}{Rr}V_a = \left(2M_w + \frac{2I_w}{r^2} + M_p\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\dot{x} + M_p l\ddot{\theta}_p \cos\theta_p - M_p l\dot{\theta}^2{}_p \sin\theta_p \tag{5.32}$$

The above two equations can be linearised by assuming $\theta_p = \pi + \phi$, where ϕ represents a small angle from the vertical upward direction. This simplification was used to enable a linear model to be obtained so linear state space controllers could be implemented.

Therefore,

$$\cos\theta_p = -1, \quad \sin\theta_p = -\phi \quad \text{and} \left(\frac{d\theta_p}{dt}\right)^2 = 0 .$$

The linearised equation of motion is,

$$\left(I_p + M_p l^2\right)\ddot{\phi} - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a - M_p g l \phi = M_p l \ddot{x} \tag{5.33}$$

$$\frac{2k_m}{Rr}V_a = \left(2M_w + \frac{2I_w}{r^2} + M_p\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\dot{x} - M_p l\ddot{\phi} \tag{5.34}$$

In order to get the state space representation of the system, equations (5.33) and (5.34) are rearranged,

$$\ddot{\phi} = \frac{M_p l}{\left(I_p + M_p l^2\right)}\ddot{x} + \frac{2k_m k_e}{Rr\left(I_p + M_p l^2\right)}\dot{x} - \frac{2k_m}{R\left(I_p + M_p l^2\right)}V_a + \frac{M_p g l}{\left(I_p + M_p l^2\right)}\phi \tag{5.35}$$

$$\ddot{x} = \frac{2k_m}{Rr\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}V_a - \frac{2k_m k_e}{Rr^2\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}\dot{x} + \frac{M_p l}{\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}\ddot{\phi} \tag{5.36}$$

By substituting equation (5.35) into equation (5.34), substituting equation (5.36) into equation (5.33) and after a series of algebraic manipulation the state space equation for the system is obtained.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{2k_m k_e\left(M_p lr - I_p - M_p l^2\right)}{Rr^2\alpha} & \dfrac{{M_p}^2 g l^2}{\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{2k_m k_e\left(r\beta - M_p l\right)}{Rr^2\alpha} & \dfrac{M_p g l\beta}{\alpha} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{2k_m\left(I_p + M_p l^2 - M_p lr\right)}{Rr\alpha} \\ 0 \\ \dfrac{2k_m\left(M_p l - r\beta\right)}{Rr\alpha} \end{bmatrix} V_a \quad (5.37)$$

where,

$$\beta = \left(2M_w + \frac{2I_w}{r^2} + M_p\right) \qquad \alpha = \left[I_p\beta + 2M_p l^2\left(M_w + \frac{I_w}{r^2}\right)\right]$$

In the model above, it is assumed that the wheels of the vehicle will always stay in contact with the ground and that there is no slip at the wheels. Cornering forces are also considered negligible.

6 Control System Design

The balancing robot system described in chapter 3 is an excellent test bed for control theory because it exhibits non-linear and unstable system dynamics. Control objectives for these systems are always challenging as the full state of the system is not often fully measured.

Therefore, in this chapter, it is the author's aim to show how the system can be controlled using linear state space controllers. The controllers are designed utilising the dynamics model developed for the balancing robot in the previous chapter.

6.1 Control Problem

Because the system is inherently unstable, an impulse input applied to the open loop system will cause the tilt angle and position of the robot to rise unboundedly. This results in the robot falling over as the tilting range of the robot is limited to 0.52 radians on each side. Figure 6.1 shows the simulation when an impulse input is applied to the uncontrolled system.

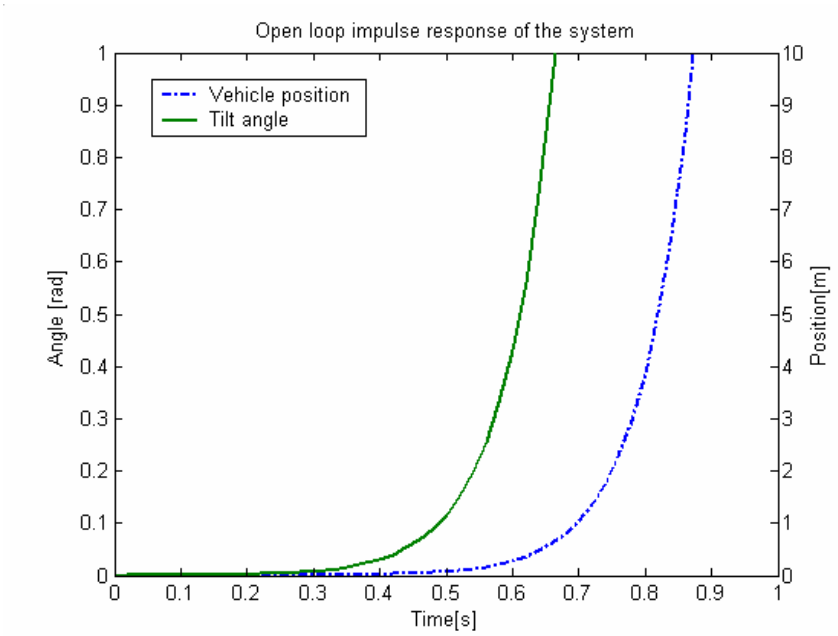


Figure 6.1: Open loop impulse response of the system

Plotting the Pole-Zero Map of the system verifies that the system is unstable as there is a pole on the right-hand plane of the plot. Ideally, all poles should be on the left-hand plane of the plot for the system to be stable.

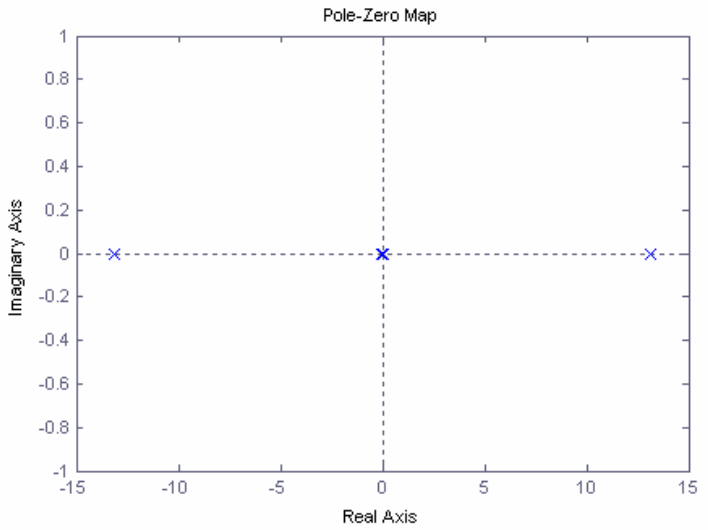


Figure 6.2: Pole-Zero map of the open loop system

The poles of the system are located at 0, -0.0078, 13.1183, -13.1202.

6.2 *Linear Quadratic Regulator (LQR)*

The Linear Quadratic Regulator (LQR) control is a modern state-space technique for designing optimal dynamic regulators. It refers to a linear system and a quadratic performance index according to

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (6.1)$$

$$x(0) = x_0 \qquad (6.2)$$

The performance of a LQR system can be represented by an integral performance index (equation 6.3). It enables a trade off between regulation performance and control effort via the performance index when the initial state x_0 is given.

$$J = \frac{1}{2}\int_0^\infty [x'(t)Qx(t) + u'(t)Ru(t)]dt \tag{6.3}$$

The control law for the LQR is specified as

$$u = -R^{-1}B'\overline{P}x \tag{6.4}$$

where $\overline{P} = \overline{P}' \geq 0$ solves the following algebraic Ricatti equation

$$0 = PA + A'P - PBR^{-1}B'P + Q \tag{6.5}$$

The gain vector $K = R^{-1}B'\overline{P}$ determines the amount of control fed back into the system. The matrix R and Q, will balance the relative importance of the control input and state in the cost function (J) being optimized with a condition that the elements in both Q and R matrices are positive values. The size of Q matrix depends on the size of the system's state matrix and R matrix is dependent on the number of control input to the system. The block diagram for the LQR controller is shown below,

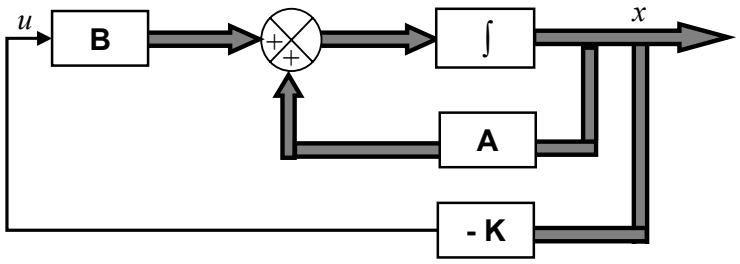


Figure 6.3: LQR control block diagram.

From Chapter 5, the state space model for the balancing robot is given as

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.0097 & 11.1594 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.0293 & 172.1160 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0815 \\ 0 \\ 0.2456 \end{bmatrix} V_a \tag{6.6}$$

For a linear control system to be implemented, the system has to be controllable. This requires that the rank of the *n* x *n* controllability matrix $C = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$ is n or $|C| \neq 0$.

Using MATLAB, the algebraic Ricatti equation is solved and the control gain K is evaluated for different values of Q and R weighting matrices. The response of the system is simulated as well.

The Q matrix assumes the form of

$$Q = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix}$$

where the values of a, b, c, d are the weightings for the respective states $x, \dot{x}, \phi, \dot{\phi}$ while the weighting matrix R is a scalar value as there is only one control input to the system. The values in the Q matrix are adjusted according to the required response of the system, a higher value of the weightings indicates the importance of the states compared to others.

The main aim of the control system is to make all the states of the system converge to zero at the shortest time possible. It will be impossible to achieve that goal as an infinite control force is non-existent; therefore some compromise on system response has to be made. A high weighting for R indicates less motor control is used to balance the system, this results in a low gain value for linear position, x, and linear velocity, $\dot{x}$ which causes the robot to continue moving in order to balance the system. This is illustrated in Figure 6.4.

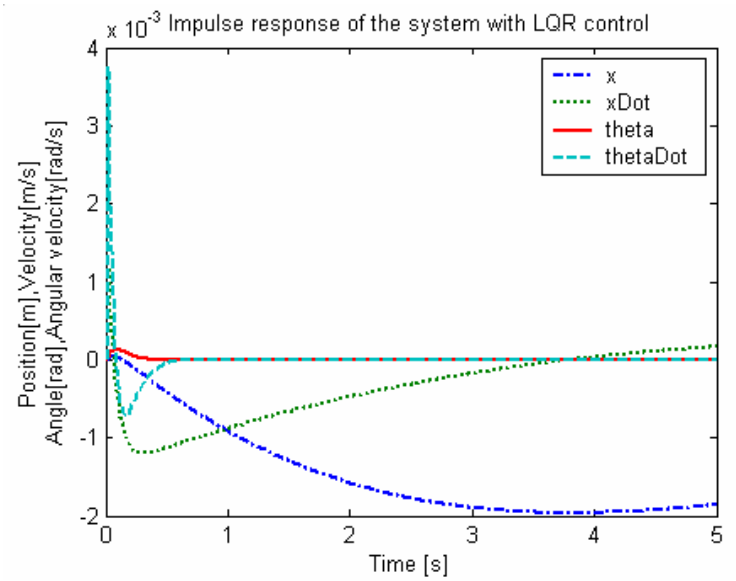


Figure 6.4: Response of the system with a high R weighting

Figure 6.4 shows that the increase of weighting for state *x* results in a high gain which reduces the settling time of the robot's position. Unfortunately, the motors will not be able to match the desired response because the required torque exceeds the maximum torque of the motor.

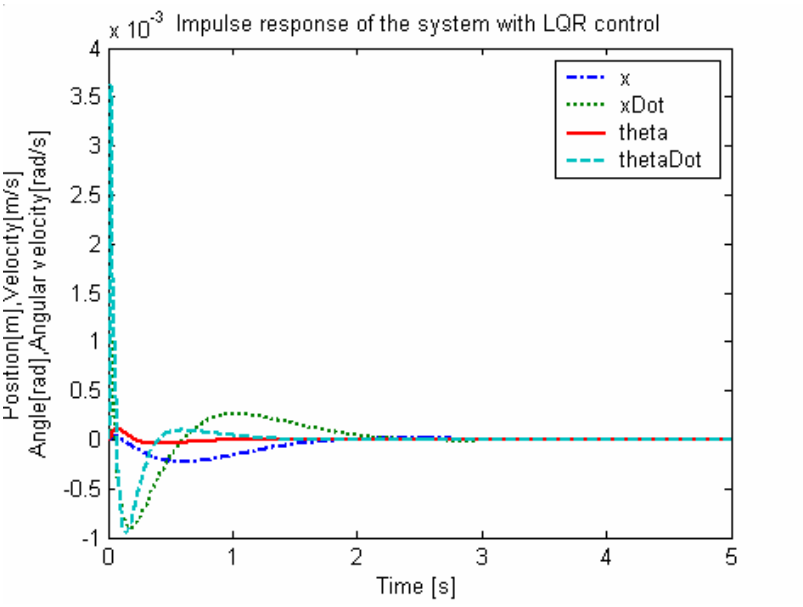


Figure 6.5: Response of the system with a high position weighting

Figure 6.5 show that the increase of weighting for tilt angle results in a high gain for the states, this will compromise the settling time and response of other states.

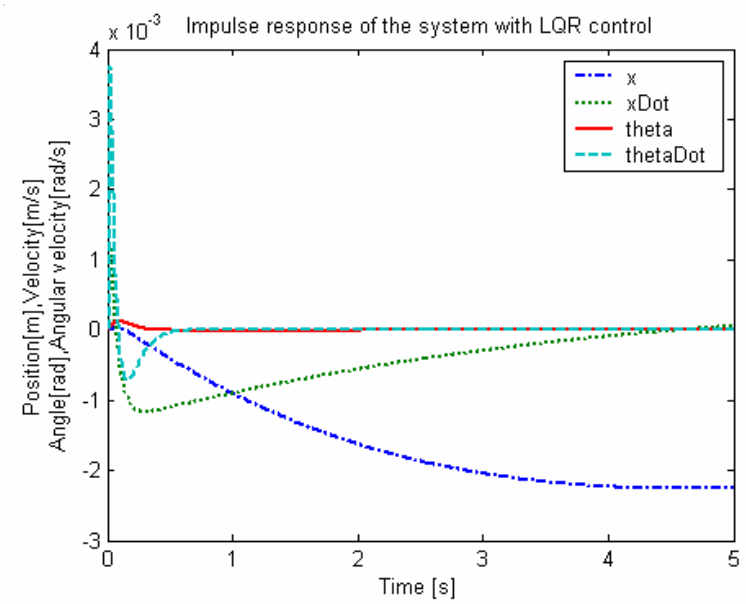


Figure 6.6: Response of the system with a high tilt angle gain

Because the elements of matrix Q and R are determined arbitrarily, several simulation and implementation trial are conducted to obtain the desired response for the balancing robot system. The most acceptable response for the system is shown in Figure 6.7.The weightings for the angular velocity and linear velocity is left unchanged as the response of the states are dependent on the other two states. Poles of the system are now placed at ***-13.1410 + 0.7342i, -13.1410 + 0.7342i, -1.0818 + 1.0768i, -1.0818 - 1.0768i.***

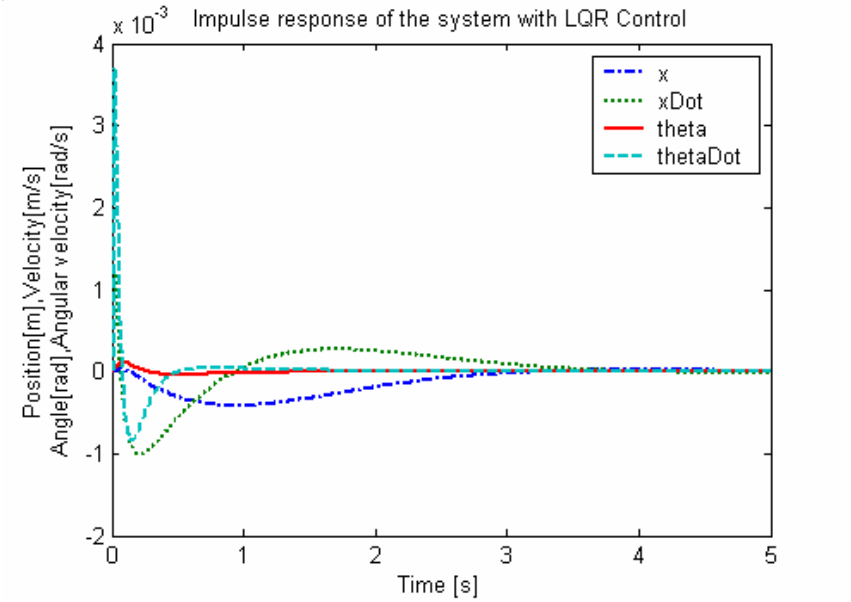


Figure 6.7: Response of the system with appropriate weighting

6.3 Pole Placement Control

The position of poles defines the stability of a system. According to linear control theory (Dorf & Bishop, 2001), the poles of the system can be arbitrarily placed in the complex plane if the Controllability matrix is of full rank. This matrix is defined by,

$$C = [B \quad AB \quad A^2B \quad A^3B] \tag{6.7}$$

The control law for the Pole-placement controller is given as

$$u = -Kx \tag{6.8}$$

where u is the control voltage, x is the state parameters and K is the state feedback gain matrix. Figure 6.8 shows the block diagram for the Pole-placement controller.

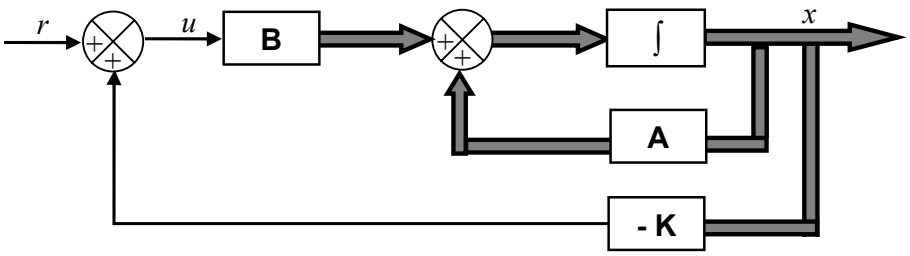


Figure 6.8: Block diagram for Pole-placement controller.

Pole-placement control gives designers the option of relocating all the closed loop poles of the system. This is in contrast with the classical design using Bode plots and frequency response methods, whereby the designer can only hope to achieve a pair of complex conjugate poles that are dominant. The theory of Pole-placement control might look trivial, but because all other poles and zeroes may fall anywhere, meeting the design specification becomes a matter of trial and error. With the freedom of choice rendered by state feedback comes the responsibility of selecting the poles judiciously.

The poles of the system have to be placed carefully as there are obvious costs that are associated with shifting pole locations. As a result, several simulation trials using MATLAB have to be performed to attain the best pole location that gives the desired response while not straining the control input.

Stability of the system can be guaranteed as long as all the poles of the system are in the left-hand plane of the pole-zero map, but the question is where in the Left-Hand Plane (LHP) should it be placed. If a fast response is desired, the poles can be placed further away from the imaginary axis. The further the poles are placed from the imaginary axis, the system will require a faster or a stronger actuator to perform the task. This is illustrated in Figure 6.9, the control input required for this pole configuration peaks at 0.9 N.

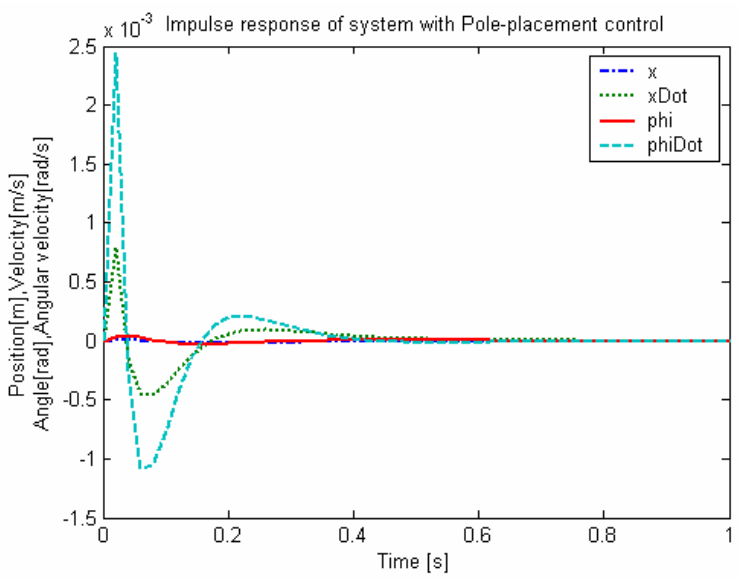


Figure 6.9: System response with poles placed too far into the LHP.

The transient response of a system is the response of a system which decays as time increases. Because the purpose of control systems is to provide a desired response, the transient response of control systems must be adjusted until it is satisfactory. In Pole-placement controllers, the transient response requirement of the system can be achieved by placing a pair of complex conjugate poles in the system.

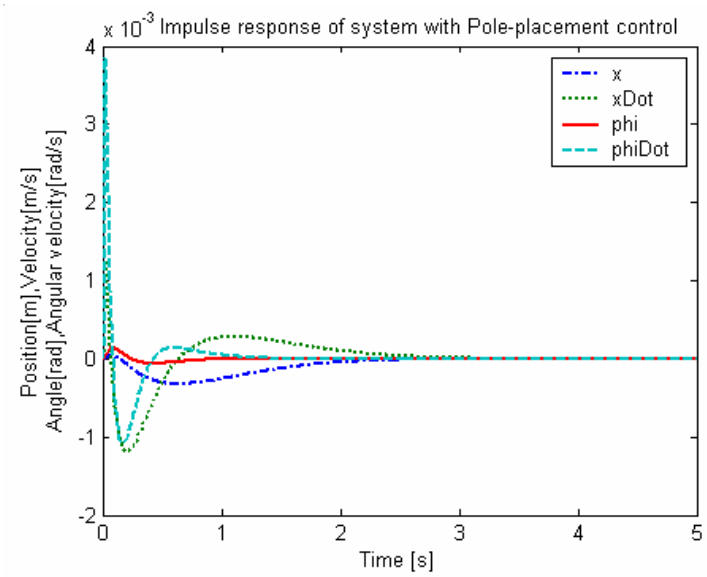


Figure 6.10: System response with Pole-placement control.

Figure 6.10 shows the result of the simulation with the poles of the system placed at $-2 \pm 1i, -10 \pm 5i$. The combination of complex poles gives a good transient response to the system and poles further away from the imaginary axis gives a fast response.

6.4 *Trajectory Control*

Differential drive robots have proven to be one of the least complicated locomotion systems. The differential scheme consists of two wheels on a common axis, each wheel driving independently. Such an arrangement gives the robot the ability to drive straight, turn in place and to move in an arc. The common assumption that differential drive robots can be driven straight all the time by applying equal amount of power to the motors is flawed. These robots will never drive straight without a proper control system because of errors like different wheel size and friction; these errors will cause the robot to deviate from its intended course.

The balancing robot uses a PID controller to overcome this problem. PID stands for Proportional-Integral-Derivative, it is a basic filter mechanism used to control some output based upon the aggregate function of factors. The PID control algorithm is used for the control of almost all loops in the process industries, and is also the basis for many advanced control algorithms and strategies.

The equation for the PID controller is given as

$$R(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \frac{de(t)}{dt} \qquad (6.9)$$

The proportional term, $K_p \cdot e(t)$ is an output value based on the difference between the set point and actual. The integral term, $K_i \cdot \int_0^t e(t)dt$ is the sum of all errors over time, this term allows the controller to advance the output such that all the errors are eventually cancelled out. The derivative term, $K_d \cdot \frac{de(t)}{dt}$ is based on the first derivative of the proportional error, it checks if the error is getting smaller as time goes by.

In order for control loops to work properly, the PID loop must be properly tuned. There are numerous ways to tune the PID controller each tuning method depends on how the PID controller is designed. For the Differential drive system, the PID controller used a set of tuning rules which is based on the Ziegler-Nichols method.

1. Select typical operating setting for desired speed, turn off integral and derivative part, and then increase K_p to max or until oscillation occurs.
2. If system oscillates, divide K_p by 2.
3. Increase K_d and observe behaviour when changing desired speed by about 5% and choose a value of K_d that gives a fast damped response.
4. Slowly increase K_i until oscillation starts. Then divide K_i by 2 or 3.

The encoder reading difference is passed through the PID loop to obtain the required feedback to be added to the motor controller. The implementation is successful if one wheel is jammed, the other will stop as well. The control block diagram for the PID control is illustrated in Figure 6.11.

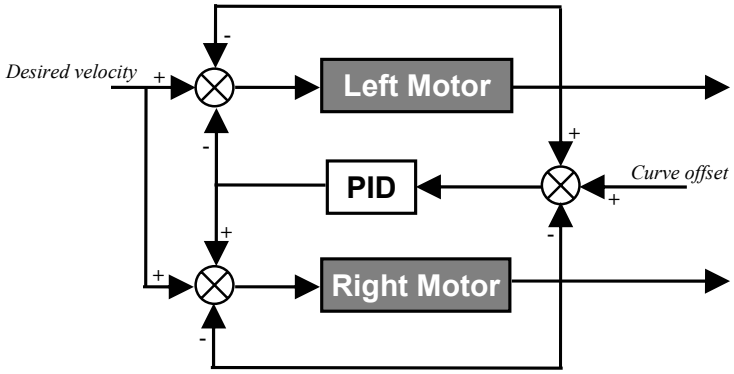


Figure 6.11: PID control scheme for differential drive. (Braunl, 2003)

7 Performance Evaluation

This section discusses the performance of the system following the implementation and testing completed in the duration of the project. The filters and control system are programmed in C language, an EYEBOT controller was used as the processor for the balancing robot system. The functional block diagram representing the information flow between the working components of the robot is shown below.

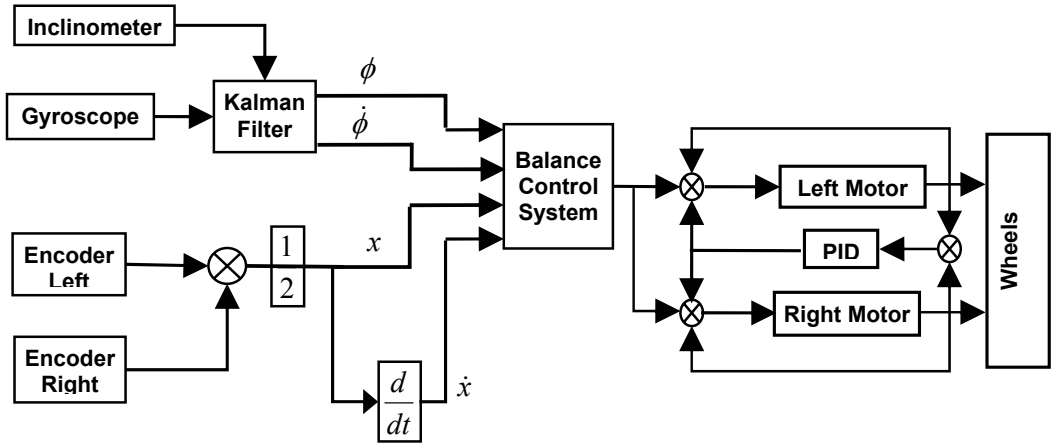


Figure 7.1: Functional block diagram of the balancing robot.

7.1 Kalman filtering

The servo test platform (Figure 4.1) described in chapter 4 was used for Kalman filter experimentation, both sensors are rotated at angles between ± 0.5 radians. There are several tasks that need to be completed before the Kalman filter implementation is achieved. Firstly, raw readings of the gyroscope have to be scaled to rad/s and be on the same scale with the inclinometer readings. As unit conversion will cause inaccuracy in measurements, the raw inclinometer readings are differentiated and compared the gyroscope raw velocity readings to obtain the correct scale factor.

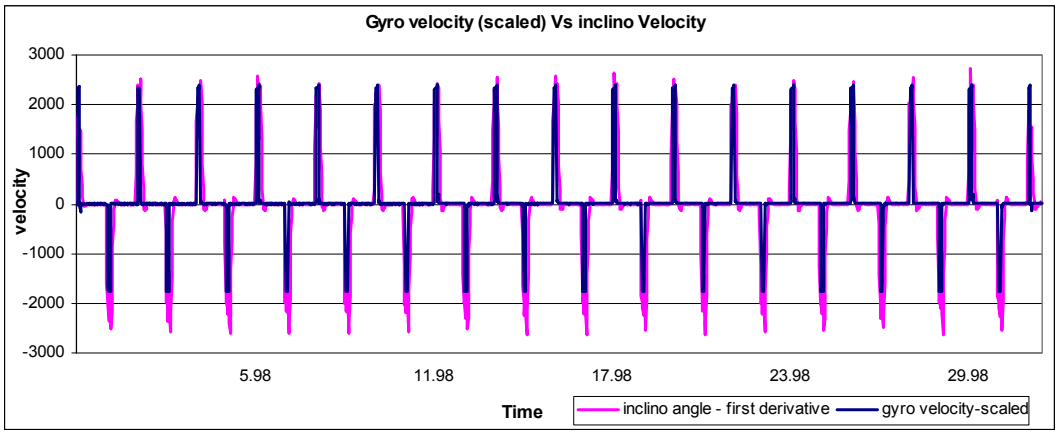


Figure 7.2: Scaled gyroscope velocity.

Figure 7.2 shows the first derivative of the inclinometer readings compared to the biased gyroscope reading with the scale factor of 0.02798. The gyroscope was unable to obtain the correct velocity reading when rotated in the negative direction, further investigation shows that this condition is due to the manufacture of the sensor and has been regarded as a systematic error in this experiment.

The Kalman filter is calibrated through the process covariance noise matrix Q and the measurement covariance noise matrix R. The forms of the matrices are given as:

$$Q = \begin{bmatrix} q\_inc & 0 \\ 0 & q\_gyro \end{bmatrix} \qquad R = \begin{bmatrix} r\_meas \end{bmatrix}$$

Values of *q_inc* and *q_gyro* are set depending on how much faith one puts on the particular sensor. A lower value indicates the particular sensor is trusted more compared to the others. The value of the R matrix indicates the amount of noise expected from the measurement, a high value means the measurement is highly corrupted with noise. The calibration of the Kalman filter is usually by some complex statistical method which requires a lot of computation. In this thesis, the values are set by trial and error, a good understanding of the process is required to quickly determine the appropriate values for the Q and R matrix.

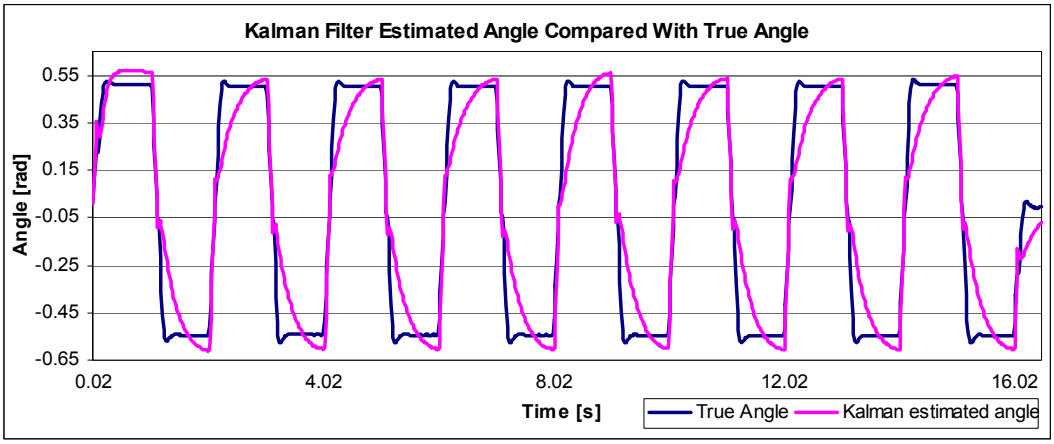


Figure 7.3: Comparison between estimated angle and true angle.

When the filter is not tuned properly, it will generate a sawtooth shaped graph as illustrated in Figure 7.3. A sawtooth output typically indicates that the sensors need calibration and the filter is only producing an estimate that is smaller than the actual angle. Another possibility is that the integration of the gyroscope reading is not propagated in real time or either the integration is propagated too fast or too slow.

The appropriate values for the Q and R matrix are found after numerous trials were performed using the servo test platform. The filter is propagated every 20ms as compared to 10ms as before which causes the sawtooth waveform output. The result of the implementation is shown below.

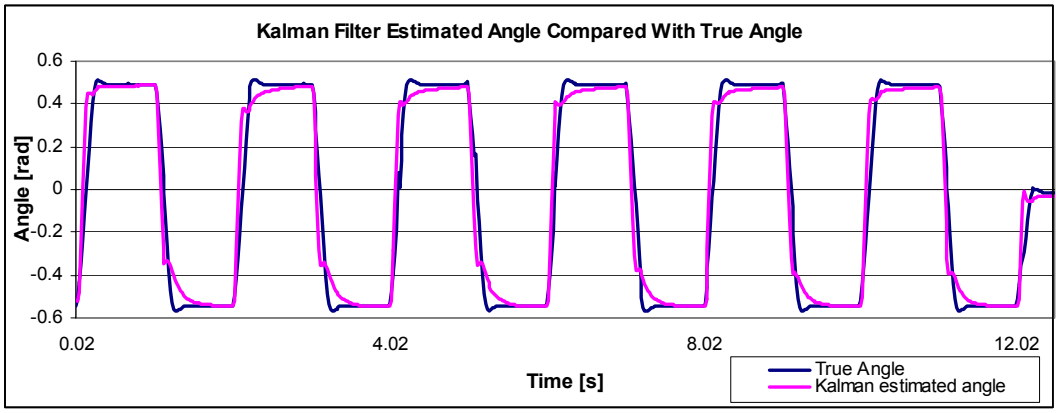


Figure 7.4: Comparison between estimated angle and true angle.

The Kalman filtered signal and raw gyroscope signal measuring the velocity of a motionless gyroscope is shown in Figure 7.5. The Kalman filter signal oscillates when it is first initialised but it immediately settles down to its steady state operation (this is

removed from the earlier plots for clarity), from that point onwards the filter is able to track instantaneous angle change with minimal error. On the other hand, the raw gyroscope signal drifted from the original reading, giving a ~ 0.18 rad/s signal when the actual reading is 0 rad/s.

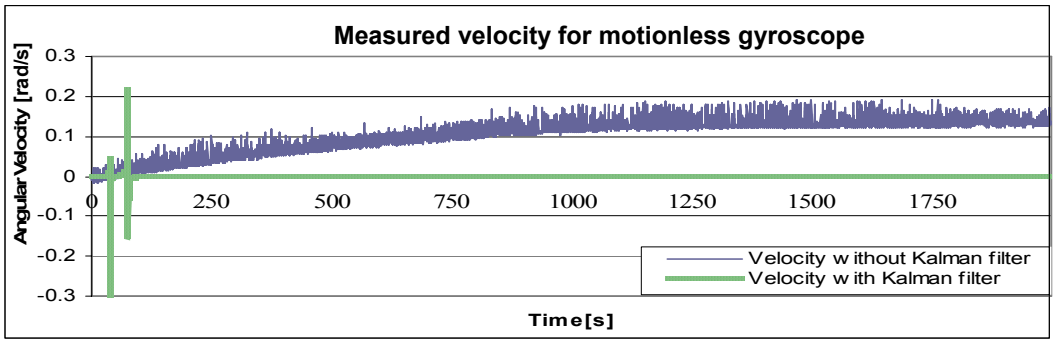


Figure 7.5: Comparison between filtered and unfiltered velocity measurement.

The Kalman filtering algorithm is initialised as a background process running at every 20 milliseconds to facilitate multitasking of the EYEBOT controller so that the processor could perform other required routines. Figure 7.6 illustrate the estimated angle when the sensors are placed on the balancing robot, a wide range of tilt angle were used to test the robustness and accuracy of the filter. The graphs have to be separated because the filter is working so well that there is almost no difference detected when the graphs are plotted together.

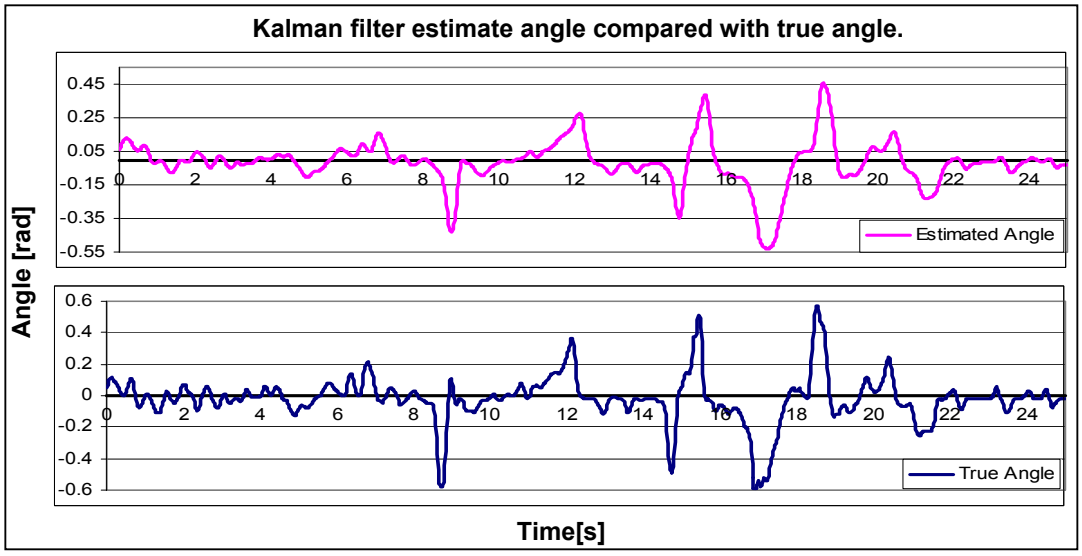


Figure 7.6: Kalman filter estimated angle for instantaneous angle change.

7.2 Balance Control

The implementation of the balance control algorithm was based on the detailed theory and simulation discussed in chapter 6. Controllers' gains were obtained from simulation results and applied to the system. Since the dynamic model of a system can never be accurate, the gains found usually acts as a foundation for further fine tuning of the controller to achieve the desired response. Figure 7.7 to 7.9 shows the graphs obtained from real-time experimentation with the Linear Quadratic Regulator.

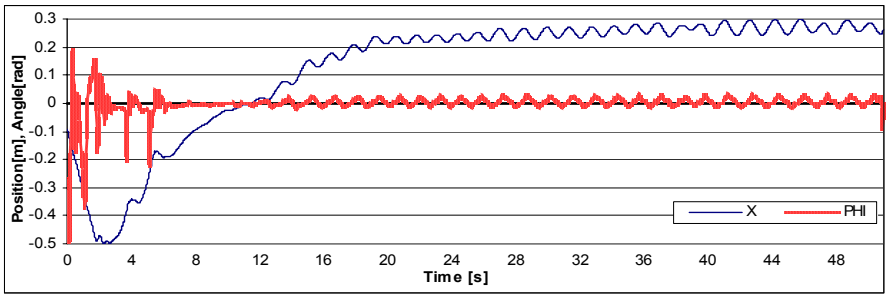


Figure 7.7: Real-time experimentation with LQR control [Position, Angle].

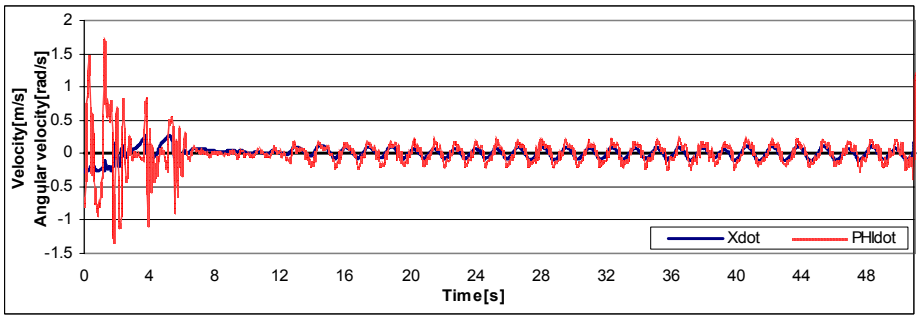


Figure 7.8: Real-time experimentation with LQR control [Velocity, Angular velocity].

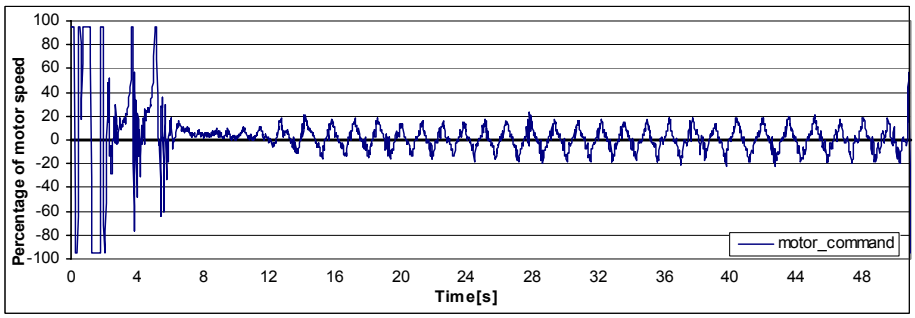


Figure 7.9: Real-time experimentation with LQR control [Motor command].

Though the system's state response is quite similar to the simulated response (Figure 6.7) with both graphs showing a settling time of about 4 seconds and the states converge to zero. It is expected that the actual system will not be as smooth as the simulation due to assumptions about unknown parameters made at the modelling stage.

At initialisation, the Kalman filter requires a couple of seconds to settle to its steady state operation. During this phase, the filter is unable to provide a correct estimate of angle and angular velocity to the system, thus making the robot oscillate erratically. But soon after initialisation, the Kalman filter stabilises and was able to track the angle and its derivative correctly and the robot behaves normally.

With the current weighting arrangement of the LQR controller (tilt angle is weighted higher than position), the system will balance the robot initially at the expense of the position, but this error should be compensated once the LQR controller is able to balance the robot. Figure 7.7 showed that the position of the robot, x moved 0.5 meters from the zero position when trying to balance the robot. However, instead to settling back to its initial position, the robot settles at 0.2 meters away from the desired position. After several testing trials, odometry error is identified as the cause of the position difference. Reason for this is that when the robot is first initialised, the incorrect estimate of angle from the Kalman filter causes the motor to spin. This action directly affects the encoder reading which affects the position measurement.

The performance of the Pole – placement controller in balancing the robot is tested in the same way as the LQR controller. Figures 7.10 to 7.12 shows the graphs obtained from real-time experimentation with Pole – Placement control.

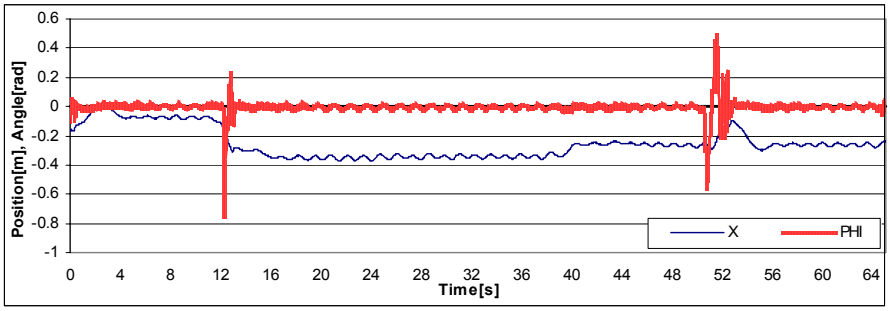


Figure 7.10: Real-time experimentation with Pole-Placement control [Position, Angle].

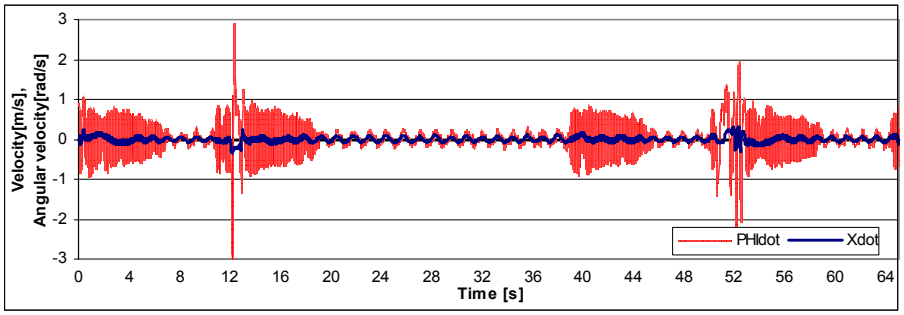


Figure 7.11: Real-time experimentation with Pole-Placement control [Velocity, Angular velocity].

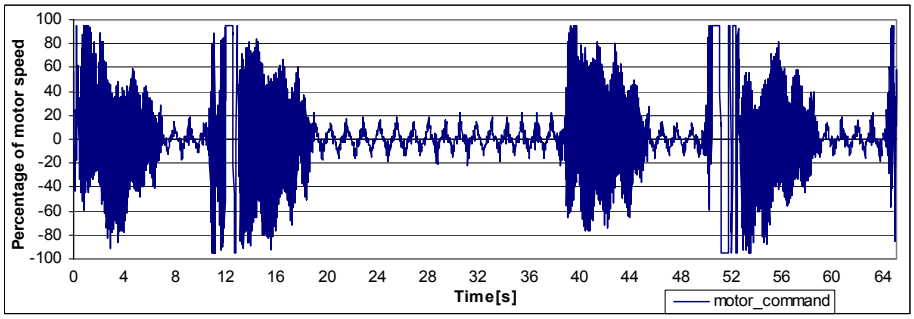


Figure 7.12: Real-time experimentation with Pole-Placement control [Motor command].

Performance of the system with Pole-Placement control is quite similar to the LQR control as most of the observations in the LQR control are also noticed here. Although the Pole – Placement controller is able to balance the system, the robustness is fairly inferior to the LQR controller. The robot keeps falling over at times since with the Pole –placement controller all states are treated equally causing the controller not knowing which states to stabilise first. The falling over of the robot is noticed when the angular velocity and motor command are highest. The robot jerks erratically when it is picked back up after falling over. At this time someone has to help the robot maintain its balance by holding the robot up for a while before letting it balance by itself again.

The effects of the assumption that the system can be linearised about its vertical operating point have started to show undesired results. While it is easier to develop a linear control system for such system, the importance of inherent nonlinearities in the system should not be neglected. Linear control systems may be stabilising the system at an operating point, it would not be able to guarantee global stability. For example, it is impossible to initialise the robot at an arbitrary angle and expect it to adjust itself to its vertical position with the current control scheme nor it would be possible for the robot

to reduce its oscillation while balancing on a specified point. Therefore, a nonlinear controller would be suggested as a future improvement to the current control system.

7.3 Trajectory Control

The amalgamation of the balance control algorithm and differential drive control algorithm results in a high manoeuvrability robot. With this in place, a driving program using a NOKIA TV infrared remote control and an IR remote controller decoder attached to EYEBOT is used to control the movement of the balancing robot. The driving program was designed to respond to a variety of remote control key codes. The following diagram shows the block diagram of the control system with the IR remote control.

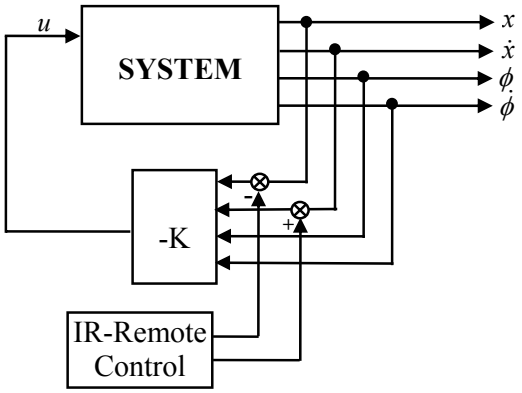


Figure 7.13: Balancing control system with IR remote control.

In order to make the robot move, the gain for the position is set to zero and the desired distance offset is added to the velocity term, this is done so that control system will continue to balance the robot while allowing the robot to move around. The direction and distance of the movement is controlled by the sign of the small offset added to the velocity term. The PID control is useful at this stage as it will ensure that the position error between the wheels are minimised so that the robot will move in a straight path. Turning the robot is achieved by applying an offset in the positive direction to the left wheel and an offset in the negative direction right wheel for turning right. To turn left, an offset in the negative direction is applied to the left wheel and an offset in the positive direction is applied to the right wheel. This will make the wheels move in the opposite directions while turning, thus making the robot turn on the spot.

Figure 7.14: The Nokia IR remote control.

The figure above shows the remote control used for the balancing robot. Only 6 buttons are used, they are,

- 2 – Move Forward.
- 8 – Move Backward.
- 5 – Stop.
- 4 – Turn Left.
- 6 – Turn Right.
- Standby – Disable remote control.

Implementation of trajectory control under remote control was verified through observation. The robot was able to maintain its balance while executing movements like drive forward, backwards and turn on a spot. However, the robot was unable to maintain its position when stopped due to the oscillation observed during balance control testing.

The PID controller was added as an odometry error control measure for the balancing robot. This might be sufficient for current state of trajectory implementation. But in order to achieve Non-Holonomic trajectory control, a better design of the PID controller using analytical methods is recommended. It is possible that the dynamics of the trajectory control be included in the system dynamics to facilitate the design of a single controller which is able to stabilise the robot as well as control its trajectory.

8 Conclusion

8.1 Project review

This project was successful in achieving its aims to balance a two-wheeled autonomous robot based on the inverted pendulum model. Two control strategies have been implemented to address the problem of balance control for the system. The gain matrices obtained from simulation for the LQR and pole-placement controller is implemented in the EYEBOT for real-time controller experimentation and both controllers showed promising results in balancing the robot. During testing, the robot is able to maintain its vertical position by slightly adjusting its wheels.

The Kalman filter has been successfully implemented. The gyroscope drift was effectively eliminated allowing for an accurate estimate of the tilt angle and its derivative for the robot.

While the stability of the system can be accomplished through the implementation of Pole-placement control, the LQR controller offers an optimal control over the system's input via the weighting matrix *R*. The arbitrary placement of control poles might cause the poles to be placed too far into the left-hand plane and cause the system susceptible to disturbances.

The Zeigler-Nichols tuning rules used for the PID controller are simple and intuitive as they require little process knowledge and can be applied with modest effort. However, for an accurate and robust operation, the trajectory control dynamics would be modelled to enable a suitable controller be designed.

More research is needed to investigate the effects of linearising the dynamics of the system mode to improve the stability and robustness of the robot. An attempt to control the system using nonlinear methods is highly recommended for future research. That way, oscillatory movements of the robot while balancing can be eliminated, thus accurate trajectory control and waypoint navigation can be implemented.

8.2 Recommendations for future work

This thesis provides the base for future research on Kalman filter applications and control systems development in the CIIPS mobile robot lab. More research should be conducted to exploit the Kalman filter technology and its application for other projects that require sensor fusion technology.

The linear control system developed in this thesis proved to be able to balance the robot under minimal disturbance. But the robustness of the system is not fully tested and is in question. More experiment needs to be performed to evaluate the robustness of the system and fine tuning of the control algorithm is required for better performance. Future research on implementing non-linear controllers is strongly recommended for the balancing robot system as it will improve the robustness of the system.

The trajectory control of the robot was delayed mainly because the robot was unable to stay at a fixed spot. When the problem is solved, the present trajectory control system can be expanded to enable waypoints to be set for the robot to follow instead of the limited movement restricted with the remote controller.

Bibliography

Anderson, D.P, (7th August 2003) nbot, a two wheel balancing robot [Online], Available from: http://www.geology.smu.edu/~dpa-www/robo/nbot/.) [28.09.2003]

Astrom, K & T. Hagglund. 1995 'PID Controllers: Theory, Design and Tuning'*Instument Society of America.* United States of America.
Barshan, Billur & Hugh F. Durrant-Whyte. 1995 'Inertial Navigation Systems for Mobile Robots'*, IEEE Transactions on Robotics and Automation*, vol 11, no. 3.

Borenstein, J & L. Feng. 1996, 'Gyrodometry: A New Method for Combining Data from Gyros and Odometry in Mobile Robots', *IEEE International Conference on Robotics and Automation,* Minneapolis, Apr 22-28, 1996, pp. 432-428.

Del Gobbo, Diego, Napolitano, Marcello, Parviz Famouri & Mario Innocenti. 2001 'Experimental Application of Extended Kalman Filtering for Sensor Validation'*, IEEE Transactions on Control Systems Technology*. vol. 9, no. 2.

Dorf, Richard & Robert H. Bishop. 2001 'Modern Control Systems', Prentice-Hall, United States of America.

Doskocz, Edward, Yuri Shtessel & Constantine Katsinis, 1998 'MIMO Sliding Mode Control of a robotic "Pick and Place" System Modeled as an Inverted Pendulum', *Proceedings of the Thirteenth Southestern Symposium on System Theory,* pp. 379-383.

Grasser, Felix, Alonso D'Arrigo, Silvio Colombi & Alfred C. Rufer. 2002 'JOE: A Mobile, Inverted Pendulum'*, IEEE Transactions on Industrial Electronics*, Vol 49.

Ha, Yunsu & Shin'ichi Yuta. 1996 'Trajectory Control for Navigating of Self-Contained Mobile Inverse Pendulum*', Robotics and Automated Systems*, v 17, no 1-2, pp 65-80.

Komoriya, K. & Oyama, E., 1994 'Position estimation of a mobile robot using optical fiber gyroscope (OFG)', *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 'Advanced Robotic Systems and the Real World'*, vol 1, pp. 143 -149.

Kalman R.E, 1960 'A New Approach to Linear Filtering and Prediction Problems', *Transactions of the ASME – Journal of Basic Engineering, 82(Series D), pp. 35-45.*

Lahdhiri, T., Carnal, C.L. & A.T. Alouani, 1994 'Cart-Pendulum Balancing Problem Using Fuzzy Logic Control', *Proceedings of the 1994 IEEE Southeastcon on 'Creative Technology Transfer – A Global Affair',* pp. 393 – 397.

Locatelli, Arturo. 2000, 'Optimal Control: An introduction', Birkhäuser Verlag, Switzerland.

Luo, Ren. C & Michael G. Kay (ed), 1995 'Multisensor Integration and Fusion for Intelligent Machines and Systems', *Ablex Publications, Norwood, N.J.*

Maybeck, Peter. 1979, 'Stochastic Models, Estimation, and Control Volume 1', Academic Press, United States of America.

Nakajima, Ryo, Takashi Tsubouchi, Shin'ichi Yuta & Eiji Koyanagi. 1997 'A Development of a New Mechanism of an Autonomous Unicycle', *IEEE International Conference on Intelligent Robots and Systems*, v 2, pp 906-912.

Ozaki, Hiroaki, Takashiro Ohgushi, Tetsuji Shimogawa & Chang-Jun Lin. 2001 'Position and Orientation of a Wheeled Inverted Pendulum', *JSME International Journal, series C: Mechanical Systems, Machine Elements and Manufacturing*, v 44, n 1, pp 188-195.

Shiroma,N, Matsumoto, O., Kajita, S. & Tani, K, 1996 'Cooperative Behaviour of a Wheeled Inverted Pendulum for Object Transportation', *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems '96, IROS 96*, Volume: 2 , 4-8 Nov. 1996 Page(s): 396 -401 vol.

Smith, C.M., 2002 'Vision Support system for Tracked Vehicle', BE thesis, University of Western Australia.

Sutherland, Alistair & Thomas Bräunl, 2002 'An Experimental Platform for Researching Robot Balance', Department of Electrical & Electronic Engineering. University of Western Australia.

Sugihara, T, T Nakamura & Hirochika I., 2002 'Realtime Humanoid Motion Generation Through ZMP Manipulation based on Inverted Pendulum Control',*Proceedings of the 2002 IEEE International Conference on Robotics & Automation,*Washington D.C.

Takahashi, Y., Ishikawa, N. & Hagiwara, T, 2001 'Inverse pendulum controlled two wheel drive system', *Proceedings of the 40th SICE Annual Conference, International Session Papers SICE 2001*, pp. 112 -115.

Welch, Greg & Gary bishop, 2002 'An Introduction to Kalman Filtering', Department of Computer Science, University of North Carolina at Chapel Hill. Chapel Hill, NC. United States of America.

Williams V. & K. Matsuoka, 1991 ' Learning to Balance the Inverted Pendulum Using Neural Networks', *IEEE International Joint Conference on Neural Networks,* vol 1, pp. 214-219.

Xu, Ke & Xu-Dong Duan, 2002 'Comparative Study of Control Methods of Single-Rotational Inverted Pendulum', *Proceedings of the First International Conference of Machine learning and Cybernetics*, Beijing.

Zarchan, Paul & Howard Musoff. 2000 'Fundamentals of Kalman Filtering', American Institute of Aeronautics and Astronautics, United States of America.

Appendix I: Simulation Codes

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% open_loop_impulse_responseMD.m
% Simulation of open loop impulse response of Balancing robot
% This model includes the motor dynamics
% Author: Rich Chi Ooi (0248566)
% 12.08.2003
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variable initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 g=9.81;             %Gravity(m/s^2)
 r=0.051;             %Radius of wheel(m)
 Mw=0.03;             %Mass of wheel(kg)
 Mp=1.13;             %Mass of body(kg)
 Iw=0.000039;         %Inertia of the wheel(kg*m^2)
 Ip=0.0041;           %Inertia of the body(kg*m^2)
 l=0.07;              %Length to the body's centre of mass(m)

 %Motor's variables
  Km = 0.006123;    %Motor torque constant (Nm/A)
  Ke = 0.006087;    %Back EMF constant (Vs/rad)
  R = 3;          %Nominal Terminal Resistance (Ohm)

% Va = voltage applied to motors for controlling the pendulum

%%%%%%%%%%%%%%
% System Matrices
%%%%%%%%%%%%%%

%pre-calculated to simplyfy the matrix
%Denominator for the A and B matrices
beta = (2*Mw+(2*Iw/r^2)+Mp);
alpha = (Ip*beta + 2*Mp*l^2*(Mw + Iw/r^2));

A = [0                 1                            0              0;
    0     (2*Km*Ke*(Mp*l*r-Ip-Mp*l^2))/(R*r^2*alpha)     (Mp^2*g*l^2)/alpha    0;
    0                 0                            0              1;
    0     (2*Km*Ke*(r*beta - Mp*l))/(R*r^2*alpha)       (Mp*g*l*beta)/alpha   0]

 B = [            0;
     (2*Km*(Ip + Mp*l^2 - Mp*l*r))/(R*r*alpha);
              0;
     (2*Km*(Mp*l-r*beta)/(R*r*alpha))]

 C = [1 0 0 0;
     0 0 1 0]

 D = [0;
     0]

%TRANSFER FUNCTION FORM OF THE STATE SPACE MODEL
disp('Transfer Function of the system')
[num,den] = ss2tf(A,B,C,D)
```

```
%OBTAINING THE IMPULSE RESPONSE OF THE SYSTEM
T = 0:0.02:10;
U = zeros(size(T));
U(1) = 1; %input voltage
[Y,X] = lsim(A,B,C,D,U,T);
plot(T,Y);

title('Open loop impulse response of the system')
ylabel('Position[m], Angle[rad]')
xlabel('Time[s]')
legend('Vehicle position','Tilt angle')
axis([0 2 0 100])



%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LQR_controlMD.m
% Simulation of Balancing Robot with LQR control
% This model includes the motor dynamics
% Author: Rich Chi Ooi (0248566)
% 12.08.2003
%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% Variable initialization
%%%%%%%%%%%%

 g=9.81;            %Gravity(m/s^2)
 r=0.051;           %Radius of wheel(m)
 Mw=0.03;           %Mass of wheel(kg)
 Mp= 1.13;          %Mass of body(kg)
 Iw=0.000039;       %Inertia of the wheel(kg*m^2)
 Ip=0.0041;         %Inertia of the body(kg*m^2)
 l=0.07;            %Length to the body's centre of mass(m)

 %Motor's variables
  Km = 0.006123;    %Motor torque constant (Nm/A)
  Ke = 0.006087;    %Back EMF constant (Vs/rad)
  R = 3;            %Nominal Terminal Resistance (Ohm)

% Va = voltage applied to motors for controlling the pendulum

%%%%%%%%%%%%
% System Matrices
%%%%%%%%%%%%

%pre-calculated to simplyfy the matrix
%Denominator for the A and B matrices
beta = (2*Mw+(2*Iw/r^2)+Mp);
alpha = (Ip*beta + 2*Mp*l^2*(Mw + Iw/r^2));

A = [0                  1                           0           0;
     0    (2*Km*Ke*(Mp*l*r-Ip-Mp*l^2))/(R*r^2*alpha)    (Mp^2*g*l^2)/alpha    0;
     0                  0                           0           1;
     0    (2*Km*Ke*(r*beta - Mp*l))/(R*r^2*alpha)       (Mp*g*l*beta)/alpha   0]

 B = [          0;
       (2*Km*(Ip + Mp*l^2 - Mp*l*r))/(R*r*alpha);
                0;
       (2*Km*(Mp*l-r*beta)/(R*r*alpha))]

 C = [1 0 0 0;
```

```
      0 0 1 0]

D = [0;
     0]

%Obtaining the eigenvalues of the system matrix
disp('The eigenvalues of the system matrix A')
disp('A positive value will indicate an unstable system')
p = eig(A)

%%%%%%%%%%%%%%
%LQR control design
%%%%%%%%%%%%%%
disp('Designing the optimal controller')
disp('Q = C"*C is a 4 x 4 weighting matrix for the outputs')
disp('Q could well be Identity matrix with size same with system matrix A, as long as it is positive
definite')
disp('R is a 1 x 1 weighting matrix for the input')

%x is the weighting for the cart position
%y is the weighting for the pendulum position

x = 1000; y =5000;

Q = [x 0 0 0;
    0 1 0 0;
    0 0 y 0;
    0 0 0 1];

 R =1;

BRinverse = B*inv(R)*B';
P = are(A,BRinverse,Q);

%Feedback Gain
disp('Feedback Gains for the system')
K = inv(R)*B'*P

%%%%%%%%%%%%%%
%Simulate the system
%%%%%%%%%%%%%%
%Simulation time step
T=0:0.02:5;

%Impulse response input
U=zeros(size(T));
U(1)= 1;

%System matrices with feedback
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];

%Obtaining the States and the output response
[Y,X]=lsim(Ac,Bc,Cc,Dc,U,T);

%Obtaining the torque needed to control the system
[n m] = size (X);
for i = 1:n
   UU(i) = -K*X(i,:)';
end
```

```
new_poles = eig(Ac)
figure,
%plot the states
title(' Impulse response of the plant with LQR control')
subplot(1,1,1), plot(T,[X(:,1) X(:,2) X(:,3) X(:,4)]), xlabel('Time [s]'),
ylabel('Position[m],Velocity[m/s],Angle[rad],Angular velocity[rad/s]')
legend('x','xDot','phi','phiDot')

%plot the outputs
%subplot(3,1,2), plot(T,[Y(:,1) Y(:,2)]), xlabel('Time [s]'), ylabel('Outputs')
%legend('x','theta')
%plot the control input
%subplot(2,1,2), plot(T,UU), xlabel('Time [s]'), ylabel('Control u')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pole_placementMD.m
% Simulation of Balancing Robot with pole placement control
% This model includes the motor dynamics
% Author: Rich Chi Ooi (0248566)
% 04.09.2003
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% Variable initialization
%%%%%%%%%%%%

 g=9.81;            %Gravity(m/s^2)
 r=0.051;           %Radius of wheel(m)
 Mw=0.03;           %Mass of wheel(kg)
 Mp= 1.13;          %Mass of body(kg)
 Iw=0.000039;       %Inertia of the wheel(kg*m^2)
 Ip=0.0041;         %Inertia of the body(kg*m^2)
 l=0.07;            %Length to the body's centre of mass(m)

 %Motor's variables
  Km = 0.006123;    %Motor torque constant (Nm/A)
  Ke = 0.0069203;   %Back EMF constant (Vs/rad)
  R = 3;            %Nominal Terminal Resistance (Ohm)

% Va = voltage applied to motors for controlling the pendulum

%%%%%%%%%%
% System Matrices
%%%%%%%%%%

%pre-calculated to simplyfy the matrix
%Denominator for the A and B matrices
beta = (2*Mw+(2*Iw/r^2)+Mp);
alpha = (Ip*beta + 2*Mp*l^2*(Mw + Iw/r^2));

A = [0                  1                               0             0;
    0     (2*Km*Ke*(Mp*l*r-Ip-Mp*l^2))/(R*r^2*alpha)     (Mp^2*g*l^2)/alpha    0;
    0                   0                               0             1;
    0     (2*Km*Ke*(r*beta - Mp*l))/(R*r^2*alpha)       (Mp*g*l*beta)/alpha   0]

 B = [             0;
     (2*Km*(Ip + Mp*l^2 - Mp*l*r))/(R*r*alpha);
                   0;
     (2*Km*(Mp*l-r*beta)/(R*r*alpha))]

 C = [1 0 0 0;
     0 0 1 0]
```

```
 D = [0;
      0]

%Obtaining the eigenvalues of the system matrix
disp('The eigenvalues of the system matrix A')
disp('A positive value will indicate an unstable system')
p = eig(A)

%%%%%%%%%%%%%%%%%%
%Pole Placement control design
%%%%%%%%%%%%%%%%%%

disp('The system matrix have to be full rank for pole placement control')
rank_=rank(ctrb(A,B))
P = [-10 -11 -20 -25]
K = place(A,B,P)
%%%%%%%%%%%%%%%%
%Simulate the system
%%%%%%%%%%%%%%%%

%Simulation time step
T=0:0.02:5;

%Impulse response input
U=zeros(size(T));
U(1)= 1;

%System matrices with feedback
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];

%Obtaining the States and the output response
[Y,X]=lsim(Ac,Bc,Cc,Dc,U,T);

%Obtaining the torque needed to control the system
[n m] = size (X);
for i = 1:n
   UU(i) = -K*X(i,:)';
end

figure,
%plot the states
title(' Impulse response of the plant with Pole Placement control')
plot(1), plot(T,[X(:,1) X(:,2) X(:,3) X(:,4)]), xlabel('Time [s]'),
ylabel('Position[m],Velocity[m/s],Angle[rad],Angular velocity[rad/s]')
legend('x','xDot','phi','phiDot')

% %plot the outputs
% subplot(3,1,2), plot(T,[Y(:,1) Y(:,2)]), xlabel('Time [s]'), ylabel('Outputs')
% legend('x','theta')
 %plot the control input
 plot(2), plot(T,UU), xlabel('Time [s]'), ylabel('Control u')
```

Appendix II: Contents of the CD

The attached CD at the back of the Thesis contains the following:

Folder: Balancing Videos

- Contains clips of video documentation of testing and presentation.

Folder: Documents

- Contains PDF and Microsoft Word files of the thesis, presentation slides, seminar papers and Word document of progress reports.

Folder: Robot simulation with motor dynamics

- Contains Matlab codes for system simulation with motor dynamics.

Folder: Robot simulation without motor dynamics

- Contains Matlab codes for system simulation without motor dynamics.

Folder: Source Codes

- Contains all source codes developed in this project.