Table of Contents

List of figures

Chapter 1

1. Introduction

1.1 Ballbot as an example of inverted pendulum

Ballbot is an example of control of an inverted pendulum which is naturally an unstable system. Especially designed as an unstable system, the fundamental part of a ballbot is a reliable control. The desire of this project is to build tall and narrow mobile robot that do not tip over and that leads to development of balancing mobile robot like the ballbot.

A ballbot generally has a body that balances on top of a single spherical wheel (ball). It forms an under actuated system, *i.e.*, there are more degrees of freedom (DOF) than there are independent control inputs. The ball is directly controlled using actuators, whereas the body has no direct control. The body is kept upright about its unstable equilibrium point by controlling the ball, much like the control of an inverted pendulum. This leads to limited but perpetual position displacements of the ballbot. The counter-intuitive aspect of the ballbot motion is that in order to move forward, the body has to lean forward and in order to lean forward, the ball must roll backwards. All these characteristics make planning to achieve desired motions for the ballbot a challenging task.

1.2 Motivation

Man has long dreamed of robotic personal aides, created to perform his every task. The idea of automation appeals not only to those who perhaps are too lazy to complete the task itself, but to those dreaming of increasing productivity or efficiency within their own lives. This ideal unfortunately does not lie within reach of current technology, but research has been conducted into individual requirements of such an aide.

Many existing robots involve the use of three or four wheels, to establish a large enough wheel-base. This wide base makes it difficult for these mobile robots to move in cluttered human environment. These robots have many other limitations like it cannot roll in any direction and

cannot perform immediate motion in every direction. These mobile robots are not suitable for changing human environment. To overcome this kind of problem Omni-directional mobile robots were proposed. A ballbot is a special type of Omni directional mobile robots. Without changing the direction of wheels, it can move in any arbitrary direction. It has a single ball as a core of its propulsion system. This ballbot tilt spontaneously in any direction and rotate about its own axis. A ballbot performs its motion in a very unique way. The concept of ballbot is very much similar to that of inverted pendulum.

The system ballbot is a dynamically stable robot balancing and driving on a single ball. It is composed of three main parts: the ball, the propulsion system and the body. The propulsive subsystem consists of three Omni wheels driven by motors with gear heads. Each motor has an encoder attached, measuring the position and speed of the motor axis.

The most important sensor is the inertial measurement unit **(IMU)** which measures all three Euler angles and the angular rates of the body.

1.3 Scope and Objectives

The main purpose of this project is the modelling and control of the Ballbot. The goal is to design a suitable controller that stabilize the unstable system. By designing a suitable controller, the ballbot is able to balance itself with a tilt angle of **10 degrees** and is able to track the given suitable set points. There are two expected outcomes of this ballbot, first one is balancing and the other one is tracking.

1.4 System Description

The ballbot is shown on following page. Our system ballbot consist of body, ball and propulsion system. The propulsion system consists of 3 actuators (Omni wheels). In this ballbot we used single disk omni-wheels because it provides single contact point between ball and itself which is required in our case. These Omni wheels are driven by DC geared motors. Each motor is attached to an encoder. The ball is the core element of the ballbot, it has to bear all the arising forces and

mechanical wear caused by rough contact surfaces. The ball arrestor is used to push the ball towards omni-wheels to increase the contact force. The body of the Ballbot includes battery, IMU (inertial measurement unit) and a microprocessor (myRIO). IMU is used to measure the Euler angles and angular rates of body and encoder is used to measure the position of ball and speed of motors. The Ballbot balance itself by measuring its body tilt angle and transfer this data to microprocessor (myRIO) to control the omni-wheels and to calculate the appropriate torque required to keep the Ballbot stable.

Figure 1.1 Portrait of Ballbot

1.5 Structure of report

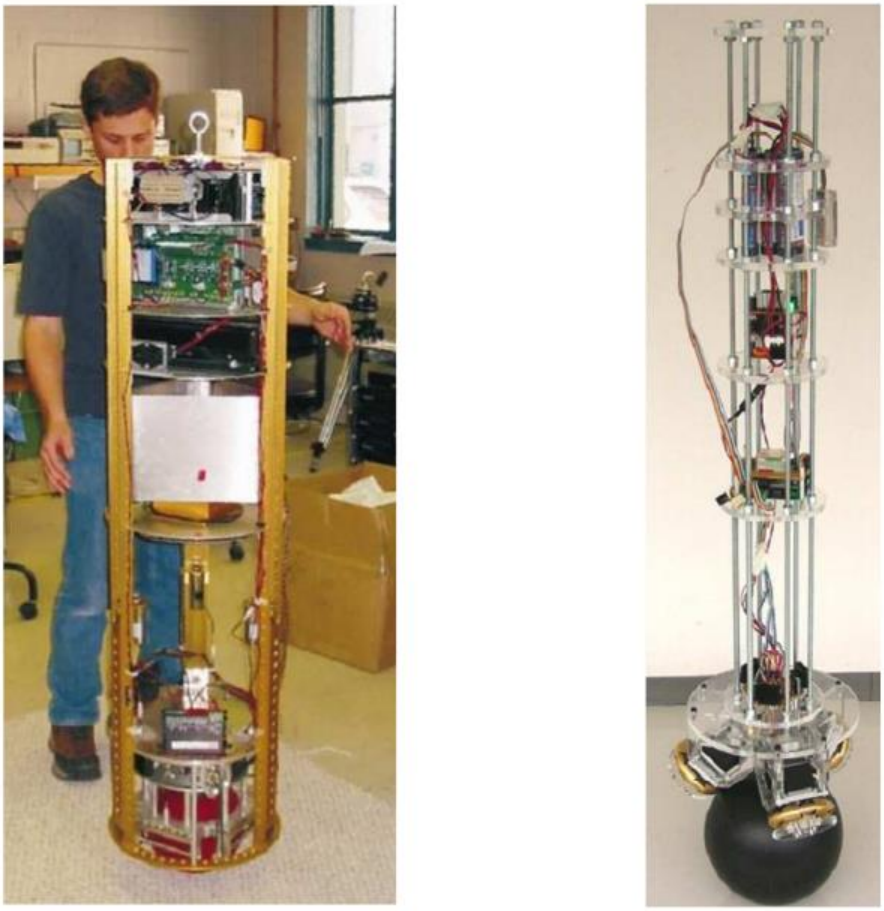
The report is divided into chapters. In chapter#1 a basic introduction of ballbot including the motivation, goal is given. In chapter#2 mechanical design and mechanical assembly of the ballbot is discussed. In chapter#3 the electronic circuitry of the ballbot is discussed. In chapter#4 three-dimensional model of the ballbot is derived and a controller for the ballbot is designed, which stabilizes the system, such that ballbot is able to track given positions set points. In chapter#5 the Simulink model and the filters that can be used are described. In chapter#6 the performance analysis is shown. In chapter#7 the details of implementation process are described. Finally, conclusions and recommendations can be found in chapter#8.

1.6 History of Ballbot

- Inverted pendulum is a system that is stable under specific conditions. It is inherently unstable but a few conditions can be matched to obtain stability.
- An **inverted pendulum** is a pendulum that has its center of mass above its pivot point. It is often implemented with the pivot point mounted on a cart that can move horizontally.
- Inverted pendulum is a standard problem in Control system and is implemented in areas of precision control and robotics. The concept is applied for high precision robotic arms, launching of a rocket, control of a Vertical Take-Off and Land (VTOL) aircraft, etc. Due to the advent of humanoid robot.
- An inverted pendulum is inherently unstable, and must be actively balanced in order to remain upright; this can be done either by
    1. Applying a torque at the pivot point,
    2. By moving the pivot point horizontally as part of a feedback system,
    3. By changing the rate of rotation of a mass mounted on the pendulum on an axis parallel to the pivot axis and thereby generating a net torque on the pendulum,
    4. By oscillating the pivot point vertically.

1.7 Background of Ballbot

A couple of ball-bots have been developed previously, with an intimate success. The first such robot was constructed and tested at Carnegie Mellon University (CMU) in 2006, and is documented by Lauwers et al. (2006). This robot, was given the name ‘Ballbot’ which has been, within the context of this report, taken to the name of all such robots. The second robot, developed at Tohoku Gakuin University (TGU) in 2007, is similar, but uses a more complex driving mechanism, and presents a more elegant solution. Photos of these robots can be seen in fig_1.2.



(a) CMU's Ballbot (b) TGU's Ballbot

Figure 1.2 Previously developed Ballbots

Both of these ball-bots are capable of balancing and point-to-point movement, with evidence of further development on the CMU ballbot including a retracting stand (Mampetta, 2006) and placing an arm on top of the ballbot (Schearer, 2006). Additionally, since the commencement of a project, a Ballbot constructed using tthe Lego NXT Mindstorms has been produced by Yamamoto (2009).

Chapter 2

2. Mechanical Design

Design considerations are an important factor when compiling a design because they facilitate an informed decision on the limitations, operating conditions and capabilities of the final product. This includes commercial aspects that may be undertaken if the product was to become mass produced or if an alternate material or component usage was to be implemented to reduce the potential manufacturing costs.

Many design considerations have to be kept in mind before finalizing the mechanical aspects of the system. There are certain limits to the design such as the weight of the body, the efficiency, safety features and various forms of control. Keeping these limitations in mind, we came up with our own customized design, which is described below.

2.1 Material Selection

The body can be constructed out of wood, plastic, metal, even heavy- duty cardboard or foam. But which to choose?

That all depends on the design of the robot. Before selecting an appropriate material, the major factors that need to be considered are:

- Durability of the material
- It’s Strength
- Maintainability
- Availability
- Energy efficiency
- Operating capability
- How big and heavy it is
- And what you intend to use it for.

The material also impacts on the size and contribute to the total weight of the final design. A heavier material such a metal is typically stronger then lighter materials such as plastics but it requires more energy to move. The size of the design impacts on its ability to navigate around, through or over obstacles as well as its transportability.

The materials we chose for our robot is aluminum, and a couple of metals for supporting bars and motor brackets. Aluminum is used for making the supporting plates to place the electronic circuitry and the motor brackets. Aluminum has advantage over other materials as Aluminum has low **density** and therefore low weight, high strength, superior **malleability**, easy machining, excellent **corrosion resistance** and good **thermal** and electrical **conductivity** are amongst aluminum's most important properties.

2.2 Mechanical Assembly

The mechanical assembly consists of a basketball on which the system balances. The actuation of the ball is then realized with three Omni wheels, which are driven by electrical motors combined with planetary gears. Using Omni wheels in ball-bots has got two advantages: on the one hand Omni wheels enable yaw rotations of the system and on the other the weight of the body acts as contact force between the wheels and the ball. The above mention parts, i.e. the motors with the planetary drives, are mounted on a circular ground plate, which also houses the suspension of the system. All relevant electronic components for the control task are positioned. Amongst others these include the IMU, which measures the accelerations and rotational velocities of the system.

2.3 Omni Wheels

Omni wheels, are wheels with small discs around the circumference which are perpendicular to the turning direction. The effect is that the wheel can be driven with full force, but will also slide laterally with great ease. These wheels are often employed in holonomic drive systems.

A platform employing three omni-wheels in a triangular configuration is generally called Kiwi Drive. They are often used in small autonomous robots in intelligent robot research in the academia.

The wheels provide traction and locomotion for the robot as well as support for the robot chassis. The diameter of the wheel should be chosen to best reflect the torque requirements of the machine. The wheel used will impact on traction and the smoothness of the motion experienced by the inertial sensors and sensitive components onboard. The number of wheels to be used in a design should depend on requirements of the capabilities.

Figure 1.1 single disk omni-wheel

Each of the three wheels will be independently controlled in either direction by a motor. The 60mm multi roller single disk omni-wheel is a robust, durable that provides easy 360° movement, with rotational and sideways maneuverability. It includes 10 rubber roller, the rollers are mounted along its circumstance and the rubber rollers to avoid slip.

2.4 Motors

Motors are an essential component for providing mobility. Without motors to move the machine, it would prove to be an expensive paperweight. Several types of motor are available including Alternating Current (AC) or Direct Current (DC) types. Variances of these motors include the

brushless, brush, servo and permanent magnet motors. Another consideration for the motor is its Revolutions per Minute (RPM). As most motors are rated between 2000-7000 RPM, some form of gearing is required to reduce the rate experienced at the wheel.

Motor we selected are DC geared motors. A DC motor is a mechanically commutated electric motor powered from direct current (DC). The stator is stationary in space. The current in the rotor is switched by the commutators.

DC motors have a rotating armature winding (winding in which a voltage is induced) but nonrotating armature magnetic field and a static field winding (winding that produce the main magnetic flux) or permanent magnet. Different connections of the field and armature winding provide different inherent speed/torque regulation characteristics. The speed of a DC motor can be controlled by changing the voltage applied to the armature or by changing the field current.

Figure 2.2 DC geared motor

This gear motor was used with an integrated encoder. The integrated quadrature encoder that provides a resolution of 64 counts per revolution of the motor shaft, which corresponds to 1856 counts per revolution of the gearbox's output shaft. Using three motors will also allow the torque to be more effectively applied to the Ball, thus reducing the need for complex calculations for frictional and rotational losses.

2.5 Ball Selection

Two types of Balls can be used for the ballbot depending upon the requirement of the application.

- Sports balls such as bowling balls
- A customaries structure, which consists of an inner form stable core and an outer friction and damping providing layer.

Both of these types are used in ballbot systems.

Regarding the price, sports balls have an advantage as they are mass-produced and therefore they are not costly, e.g. basketballs can be purchased quite easily in a convenient range.

In contrast to that, custom made balls such as used in Rezero are related to higher costs due to material and work effort. The Load capacity is high for both structures, especially for the custom made ball assuming appropriate mechanical design.

In terms of variability of the diameter the custom made design structure is beneficial as it can be adapted to our needs. Sports balls are fixed regarding their diameter as they are mass-produced.

Another relevant aspect is that the ball has sufficient friction on the ground. Sports balls made of rubber as well as the custom made structure with a proper outer layer provides this property.

We chose the basketball due to its easy availability, low price and negotiable properties.

Figure 3.3 Basketball

Chapter 3

3. Electronics and Circuitry

In this chapter, all the electronics that have been used in the project including Microcontrollers, Sensors and Power circuitry and the mechanical design of our project are discussed in detail.

3.1 Torque Controller

The ESCON Module 50/5 is a small-sized, powerful 4-quadrant PWM servo controller for the highly efficient control of permanent magnet-activated brushed DC motors or brushless EC motors up to approximately 250 Watts.

The featured operating modes – speed control (closed loop), speed control (open loop), and current control – meet the highest requirements. This Module is designed being commanded by an analog set value and features extensive analog and digital I/O functionality.

The miniaturized OEM plug-in module can be seamlessly integrated in complex customer applications.

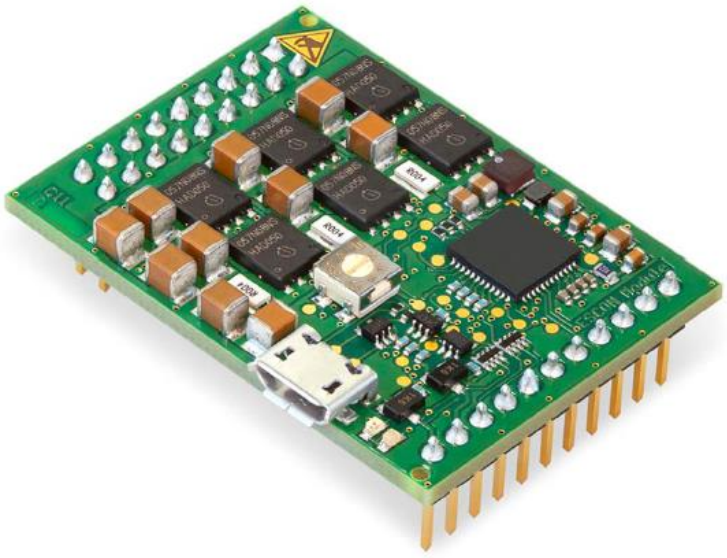


Figure 4.1 torque controller

Power Supply Requirements:

Output voltage	+VCC 10…50 VDC
Absolute output voltage	min. 8 VDC; max. 56 VDC
Output current	Depending on load • Continuous max. 5 A • Short-time (acceleration, <20 s) max. 15 A

Specifications:

Nominal operating voltage +VCC	10…50 VDC
Output voltage (max.)	0.98 x +VCC
Output current I_{cont}/I_{max} (<20 s)	5 A/15 A
Max. speed of DC motor	Limited by max. Permissible speed (motor) and max. output voltage (controller)
Sampling rate of PI current controller	53.6 kHz
Sampling rate of PI speed controller	5.36 kHz
Encoder supply voltage	+5 VDC (IL ≤70 mA)
Motor Connections	+ Motor, − Motor

3.2 Inertial Measurement Unit

An inertial measurement unit, or IMU, is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes. The IMU is the main component of inertial navigation systems used in aircraft, spacecraft, watercraft, and guided missiles among others. In this capacity, the data collected from the IMU's sensors allows a computer to track a craft's position, using a method known as dead reckoning.

3.2.1 Working of an IMU

An inertial measurement unit works by detecting the current rate of acceleration using one or more accelerometers, and detects changes in rotational attributes like pitch, roll and yaw using one or more gyroscopes. And some also include a magnetometer, mostly to assist calibrate against orientation drift.

Inertial navigation systems contain IMUs which have angular and linear accelerometers (for changes in position); some IMUs include a gyroscopic element (for maintaining an absolute angular reference).

Angular accelerometers measure how the vehicle is rotating in space. Generally, there is at least one sensor for each of the three axes: pitch (nose up and down), yaw (nose left and right) and roll (clockwise or counter-clockwise from the cockpit).

Linear accelerometers measure non-gravitational accelerations of the vehicle. Since it can move in three axes (up & down, left & right, forward & back), there is a linear accelerometer for each axis.

A computer continually calculates the vehicle's current position. First, for each of the six degrees of freedom (x, y, z and β_x, β_y and β_z), it integrates over time the sensed acceleration, together with an estimate of gravity, to calculate the current velocity. Then it integrates the velocity to calculate the current position.

3.2.2 Construction of an IMU

The term IMU is widely used to refer to a box containing three accelerometers and three gyroscopes. The accelerometers are placed such that their measuring axes are orthogonal to each other. They measure inertial acceleration, also known as G-forces. Three gyroscopes are placed in a similar orthogonal pattern, measuring rotational position in reference to an arbitrarily chosen coordinate system.

Recently, more and more manufacturers also include three magnetometers in IMUs. This allows better performance for dynamic orientation calculation in Attitude and heading reference systems which base on IMUs.

3.3 GY521

The IMU we are using is GY521. It contains both a 3-Axis Gyroscope and a 3-Axis accelerometer allowing measurements of both independently, but all based around the same axes, thus eliminating the problems of cross-axis errors when using separate devices. It is a commonly used chip that combines a MEMS gyroscope and a MEMS accelerometer and uses a standard I2C bus for data transmission.

The Specifications of GY521 are:

- Accelerometer ranges: ±2, ±4, ±8, ±16g
- Gyroscope ranges: ± 250, 500, 1000, 2000 °/s
- Voltage range: 3.3V - 5V (the module includes a low drop-out voltage regulator)

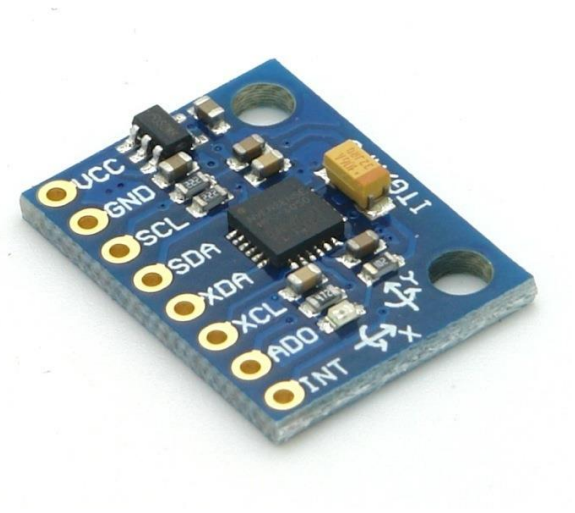


Figure 5.2 Gyroscope

3.4 Microcontroller

The purpose of a microcontroller is to complete computations and process data by executing instructions or programming. They contain all the necessary components expected of a computer system in a single board.

Ideally, a microcontroller will maintain a fast response with minimal power consumption and low susceptibility to interference. The microcontroller should contain all the devices necessary for operational requirements such as timers, ADC, DAC, and PWM outputs. This reduces the need for additional peripherals and hardware to be incorporated which would only increase the complexity required and the power requirements needed. The microcontroller should also be reprogrammable to allow improved operation and upgrades over it intended lifespan and beyond.

3.4.1 NI myRIO Controller

In this project, we used NI myRIO microcontroller. MyRIO is a real-time embedded evaluation board made by National Instruments. The National Instruments myRIO-1900 is a portable reconfigurable I/O (RIO) device. It is used to develop applications that utilize its onboard FPGA and microprocessor. It requires LabVIEW. We programmed the myRIO-1900 with LabVIEW. This Wi-Fi-enabled version allows for fast and easy integration into remote embedded applications. With its onboard devices, seamless software experience, and library of courseware and tutorials, the myRIO-1900 provides an affordable tool that helped us to complete our projects in one semester.

Figure 6.3 NI myRIO

Specifications:

- Xilinx Z-7010 processor 667 MHz (ARM Cortex A9 x2 cores 28 nm process NEON SIMD, VFPv3 Vector Float)
- Memory: NV: 256 MB, DDR3 512MB, 533 MHz, 16 bits
- FPGA type same as processor
- Wireless: IEEE 802.11 b, g, n ISM 2.4 GHz 20 MHz
- USB 2.0 Hi-Speed
- Breakout Board support
- 2 ports of 16 Digital I/O lines
- 3 axis accelerometer
- Max power consumption: 14 W
- Typical idle: 2.6 W
- LED's

3.4.2 Hardware Overview

The NI myRIO-1900 provides analog input (AI), analog output (AO), digital input and output (DIO), audio, and power output in a compact embedded device. The NI myRIO-1900 connects to a host computer over USB and wireless 802.11b, g, n.

Analog Input Channels

The NI myRIO-1900 has analog input channels on myRIO Expansion Port (MXP) connectors A and B, Mini System Port (MSP) connector C, and a stereo audio input connector. The analog inputs are multiplexed to a single analog-to-digital converter (ADC) that samples all channels.

Analog Output Channels

The NI myRIO-1900 has analog output channels on myRIO Expansion Port (MXP) connectors, Mini System Port (MSP) connector, and a stereo audio output connector. Each analog output channel has a dedicated digital-to-analog converter (DAC), so they can all update simultaneously. The DACs for the analog output channels are controlled by two serial communication buses from the FPGA.

Accelerometer

The NI myRIO-1900 contains a three-axis accelerometer. The accelerometer samples each axis continuously and updates a readable register with the result.

Number of axes..... 3

Range.....±8g
Resolution.....12bits

Samplerate.....800S/s
Noise..... 3.9 mg_{rms} typical at 25 °C

DIO Lines

The NI myRIO-1900 has 3.3 V general-purpose DIO lines on the MXP and MSP connectors. MXP connectors A and B have 16 DIO lines per connector. On the MXP connectors, each DIO line from 0 to 13 has a 40 kΩ pullup resistor to 3.3 V, and DIO lines 14 and 15 have 2.2 kΩ pullup resistors to 3.3 V.

UART Lines

The NI myRIO-1900 has one UART receive input line and one UART transmit output line on each MXP connector. The UART lines are electrically identical to DIO lines 0 to 13 on the MXP connectors.

Power Requirements

NI myRIO-1900 requires a power supply connected to the power connector.

Power supply voltage range.....6-16VDC

Maximum power consumption.....14W

Typical idle power consumption.....2.6 W

3.5 Software for NI myRIO

3.5.1 NI LabVIEW Support for NI myRIO

With NI LabVIEW software, you can take full advantage of both the processor and FPGA on NI myRIO. The easiest way to access the latest myRIO software is the LabVIEW 2015 myRIO Software Bundle Web-Based Installer. You can use this bundle to install all the required and optional software for programming myRIO. Earlier versions of the myRIO software are the LabVIEW 2014 myRIO Software Bundle and the LabVIEW 2013 myRIO Software Bundle.

The required software for programming myRIO includes:

- LabVIEW
- LabVIEW Real-Time Module
- LabVIEW myRIO Toolkit

LabVIEW is a graphical programming environment that students can use to quickly develop applications that scale across multiple platforms and operating systems. The power of LabVIEW is in its ability to interface with thousands of devices and instruments using hundreds of built-in function libraries to help you accelerate development time and quickly acquire, analyze, and present data.

Unlike text-based programming languages such as C, Java, C++, and Visual Basic, LabVIEW uses icons instead of lines of text to create applications. Due to this key difference, execution control is handled by a set of rules for data flow rather than sequentially. Wires connecting the nodes (coding elements) and VIs on the block diagram determine code execution order. In summary, LabVIEW VIs are graphical, driven by dataflow and event-based programming, and are multitarget and

multiplatform capable. They also have object-oriented flexibility and multithreading and parallelism features. LabVIEW VIs can be deployed to real-time and FPGA targets.

3.6 Why we preferred myRIO over Arduino:

- The FPGA on the myRIO allows you run really high-speed data acquisition / filtering / logic / digital IO on the order of 40 MHz clock rates - this is much faster than you could perform with the Arduino.
- The myRIO runs a linux-based real-time operating system - this means you can run your LabVIEW code on it as well as it having other utilities to allow you to configure it - you can also use multithreading to have some parallel operations on the myRIO (I think?) whereas to achieve this on the Arduino is much harder (e.g. no multithreading / use of interrupts)
- The myRIO has more I/O options - analogue/digital IO, RS232/SPI/I2C etc.
- The myRIO is more powerful - it has a faster processor / more memory etc. (you can do things like plug in a USB webcam and do vision acquisition/processing).
- I think the resolution/DAQ for the analogue inputs/outputs on the myRIO is better/faster than the Arduino
- The myRIO has built-in WiFi - for the Arduino you need either than Ethernet/WiFi shield or to use one of the Ethernet/WiFi Arduinos
- Using LabVIEW, you can more easily debug the code than you can with Arduino - Arduino is very much debugged by trial and error whereas with LabVIEW and the interactive mode, you can probe and debug more easily.
- The myRIO also has a built-in accelerometer - a good start for an IMU!

3.7 Power Source

A power source is essential for providing energy to a machine. Without energy, a machine would simply not operate. If an insufficient power source was implemented, a machine would not

function correctly nor would it perform adequately. Power sources may be AC through electrical main supplies or generators. They may also be DC from devices such as batteries. Batteries are rated by voltage output and ampere hours (AH). AH refers to the current that they can supply per hour.

Power supplies will be an essential component in allowing the robot to maintain autonomous operation. Ideally, the best practice is to minimize the total power consumption during this design phase so a smaller power system may be implemented. This is an important factor as power systems are typically the heaviest element of a robot or device.

12V and 24V power supplies are the most common power supplies of the systems available. The battery proved to be a significant expenditure of this projects budget.

Ideally the chosen power supply will also have a lower internal resistance which will allow the maximum power delivery to the motors through higher current capability. We used COSEL PAA600F-24 power supply. Specifications of this power supply are as follows.

Specifications:

MAX OUTPUT WATTAGE[W]	648
DC OUTPUT	24V 27A
Input Voltage	DC120 – 340
Output voltage	24V
Output current	27A
START-UP TIME[ms]	500max (ACIN 85V, Io=100%)
HOLD-UP TIME[ms]	20typ (ACIN 100V, Io=100%)
OPERATING INDICATION	LED (Green)

3.8 Safety Features

Safety features are an essential element of any design. They assure the safety of people involved with the machine, the machine itself and the world around it. Reducing the risk to people should always be the first objective in all designs. If potential hazards can be identified during the design

and testing phases, then safety features targeting those particular hazards can be developed to reduce the risks.

The main potential hazards created by a ballbot are those associated with it falling. The physical weight of the robot itself has been kept to a minimum to reduce the impact force and applied pressure upon contact. Electrical shock hazards remain a low risk due to the voltages used in the design so residual current detectors are not required.

Chapter 4

4. 3D System Model

Representing a system as a planer model in 2D space has many limitations.

To overcome those limitations, we here represent the system in full three-dimensional geometry and takes the parameter into account. All coupling effect that were neglected in the 2D model are considered here and hence no conversion is required. In this model assumptions are made, coordinates are defined and equation of motion are derived.

4.1 3D Model description

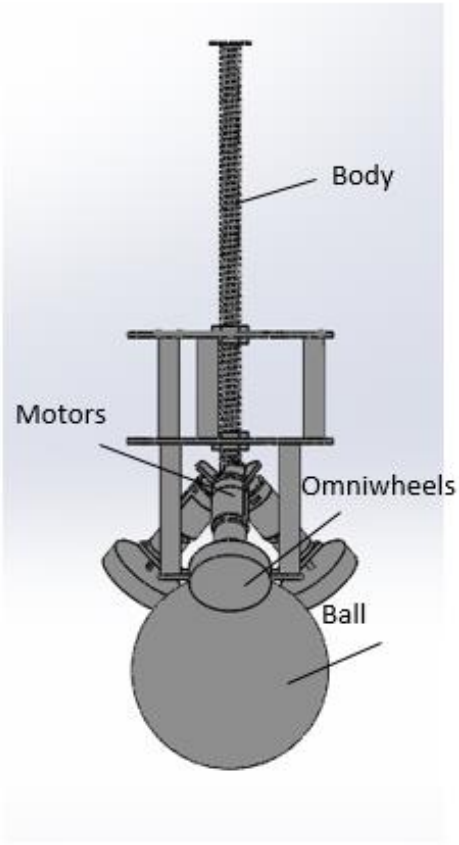


Figure 7.1 Ballbot 3D Model

The 3D model is shown in fig. This model consists of five solid bodies

- 1 ball
- 3 omni-wheels
- 1 body with 3 motors with gear head.

The ball is modeled as a hollow sphere, the body of the robot is modeled as a solid cuboid and the Omni-wheels are modeled as solid disks. The ball can rotate about any axis parallel to ground but not about the vertical axis. The ball has always single contact point with the ground. Three omni-wheels link the upper system to the ball in three points at any time. These wheels are fixed to the body with a dc geared motor. The omni-wheels can solely rotate about the motor axis in reference to the body coordinate system. The linkage between the wheels and the ball is holonomic and the wheel can only apply forces to the ball in the tangential direction of its rotation in the body reference system.

This model has five degree of freedom (DOF)

- 2 DOF for the position of the ball
- 3 DOF for the rotation of the body

4.2 Inputs and Outputs

This system has several inputs and outputs and its said to be a MIMO (multiple inputs and multiple outputs) system.

The torques generated by the motors (T1, T2, T3) are the **inputs** of the system and these torques are transmitted by the gear head. The sensors measure all the states directly and each output represents the states separately.

4.3 Assumption

In order to simplify the geometry of 3D model system, assumptions are except the assumption of the independent vertical planes. Because the rotation of the ball about vertical axis is not modeled, the ball must be in contact with the ground. In short, the assumptions were:

- **Rigid bodies**

  The system consists of 3 rigid bodies. Ball, body and 3 omni-wheels. These bodies are considered rigid because we ignore the deformation of the bodies and also deformation of the ground is neglected.

- **Frictionless**

  Besides static friction, all other types of friction, like rolling and kinetic friction, are assumed to be negligible. Non-continuous friction is hard to represent analytically and would create equations much more difficult than without friction and also accurate measurement of the friction is not possible.

- **No slip**

  The slippage between the ball and the floor and between the ball and the Omni-wheels is assumed to be negligible. This assumption convicts that:

1. In order to avoid the slippage between ball and Omni-wheels, the applied motor torques is restricted to suitable range.
2. The static friction must be high enough because if static friction is small then slippage will occur.
3. The ground must be rough enough to avoid the slipping of ball and also ball is always in contact with the ground to avoid jumping.

- **No deformation**

  The deformation appears at the contact points of the ball. In order to keep difficulty at a controllable level, this has to be neglected.

- **Negligible time delay**

  It is assumed that the time delay between the measurements of the sensors and the control of the actuators is negligible.

- **Horizontal surface**

  The ground on which the Ballbot moves is assumed to be horizontal. So, there is no potential energy of the ball. Vertical movement of the ball has to be neglected. Finally, this model is designed to move primarily on at surfaces without steep inclination.

4.4 Coordinates

The coordinate frames need to be defined in order to find the equations of motion. The coordinates are defined depending upon the degrees of freedom of the model. In this case we have 5 DOF.

- 2 DOF for the translation of ball
- 3 DOF for the rotation of the body

So the minimal coordinate vector is represented as

$$\vec{q} = [\alpha_x \; \alpha_y \; \beta_x \; \beta_y \; \beta_z]^T \qquad (2.1)$$

Where $\alpha_x \; \alpha_y$ denotes the translation of ball in x and y direction and $\beta_x \; \beta_y \; \beta_z$ denotes orientation of body in x, y and z direction.

Inertial frame I: first frame is inertial frame and it is represented by I.

First Frame K: By rotating the inertial frame around z-axis j_Z^I with angle β_z generates another coordinate frame represented as L.

Second Frame L: Now by rotating the coordinate frame K around y-axis j_y^K with angle β_y generates another coordinate frame represented as L.

Third and fixed frame M: Now by rotating the coordinate frame L around x-axis j_x^L with angle β_x generates fixed coordinate frame represented as M.

The coordinate frames are represented in Euler form. And in order to avoid the singularities, the system is operated in the range of tilt angles ±10°. The sequence of transformation of these coordinate frames is represented as **I →K → L→ M**

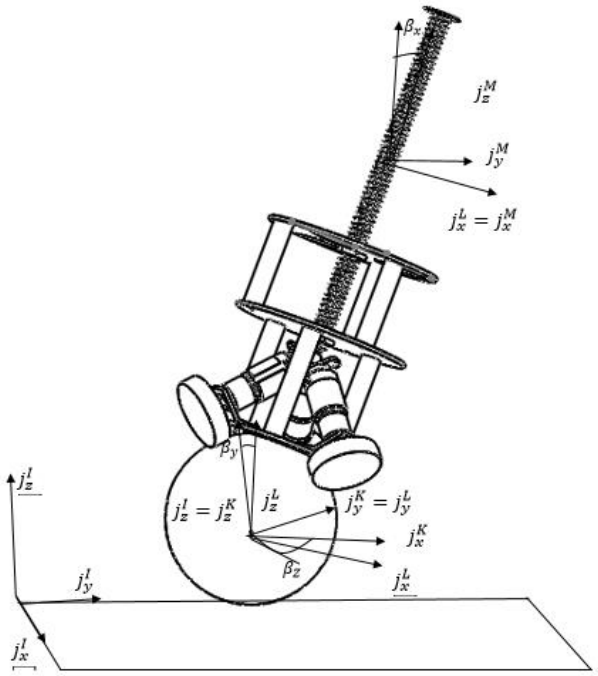


Figure 8.2 coordinates of system

The angular velocities of the body can be denoted as:

Angular velocity of Ball $\overrightarrow{\Omega_S}$:

In the coordinate system K it can be represented as:

$$K\overrightarrow{\Omega_S} = \begin{bmatrix} \dot{\alpha_x} \\ \dot{\alpha_y} \\ 0 \end{bmatrix} \quad (2.2)$$

Where $\dot{\alpha_x}$ and $\dot{\alpha_y}$ does not represent the change in orientation of the ball but represent the angular velocity of ball S $K\overrightarrow{\Omega_S}$.Error! Reference source not found.

Angular speed of an omni-wheel ω_{Wi}:

Angular speed of the omni-wheel in the coordinate frame M in direction of motor axis can be represented as

$$ {}_M\omega_{W1} = \dot{\gamma}_1 \qquad {}_M\omega_{W2} = \dot{\gamma}_2 \qquad {}_M\omega_{W3} = \dot{\gamma}_3 \tag{2.3}$$

Angular velocity of a body $\overrightarrow{\Omega_B}$:

The angular velocity is expressed by variables β_x , β_y and β_z the time variation needs to be converted with Jacobian matrix. J. so, in coordinate system M it can be parameterize as:

$$ {}_M\overrightarrow{\Omega_B} = \mathrm{J}.\dot{\vec{\beta}} = \begin{bmatrix} \dot{\theta}_x - \sin\theta_y \,.\, \dot{\theta}_z \\ \cos\theta \,.\, \dot{\theta}_y + \cos\theta_y \,.\, \sin\theta_x \,.\, \dot{\theta}_z \\ -\sin\theta_x \,.\, \dot{\theta}_y + \cos\theta_x \,.\, \cos\theta_y \,.\, \dot{\theta}_z \end{bmatrix} \tag{2.4}$$

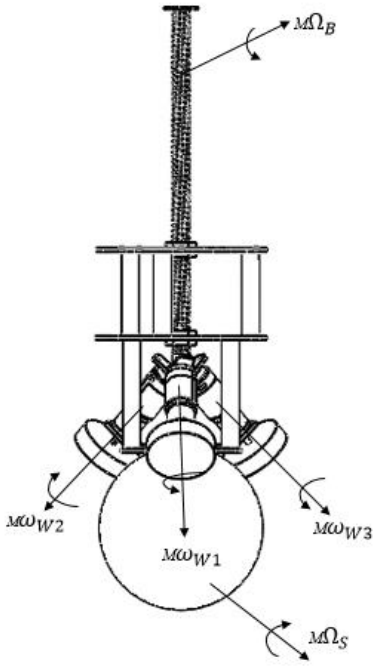


Figure 9.3 Angular velocities of system

4.5 Binding Equations:

This system has five DOF as described above. In order to express all the geometrical variables in terms of primary representation of the system, all the geometric needs have to be formulated.

In order to derive the equations of motion all coordinates must be expressed as a function of minimal coordinates. This can be primarily represented as α_x, α_y, β_x, β_y and β_z. It is important to note that while β_x, β_y, and β_z are spatial orientation angles, α_x and α_y are the integral of the parameterization of the angular velocity of the ball. α_x and α_y do not represent the orientation of the ball but the rolled angles in the respective direction.

Absolute Rotation of the omni-wheels:

The omni-wheels are attached to the body of the robot, so only the rotational motion of the omni-wheels has to be calculated. And only the angular rate is relevant for the model because the omni-wheels are fixed to the body of the robot.

First of all, absolute rotation of omni-wheels are calculated. The vectors are represented as:

$\overrightarrow{M'W_1}$: Vector from intersection of motors $M'$to the center of wheel 1.

$\overrightarrow{M'W_2}$: Vector from intersection of motors $M'$to the center of wheel 2.

$\overrightarrow{M'W_3}$: Vector from intersection of motors $M'$to the center of wheel 3.

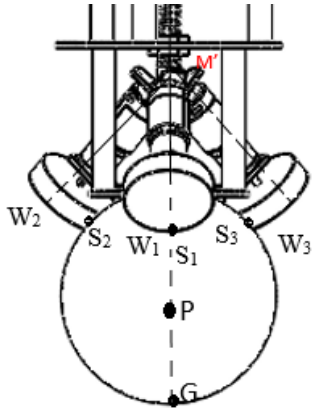


Figure 10.4 Geometric points for binding equations

Now the absolute angular speed of omni-wheels about motor axis in the body reference frame M can now be written as:

$$ {}_M\vec{\Omega}_{W_i} = {}_M\omega_{Wi} + \frac{{}_M\overrightarrow{M'W_i}}{\left\| {}_M\overrightarrow{M'W_i} \right\|} \cdot {}_M\vec{\Omega}_B \qquad \text{for i=1, 2, 3} \qquad (2.5) $$

Dependency on the Rotation of omni-wheels:

To calculate the tangential speed of the ball, first define the vectors from the center of the ball to the contact points with the omni-wheels. The vectors in the reference frame M are represented as:

$ {}_M\overrightarrow{PS_1}$: vector from the center of the ball P to the point S1 where omni-wheel is in contact with ball.
$ {}_M\overrightarrow{PS_2}$: vector from the center of the ball P to the point S2 where omni-wheel is in contact with ball.
$ {}_M\overrightarrow{PS_3}$: vector from the center of the ball P to the point S3 where omni-wheel is in contact with ball.
The directions of the tangential velocity of the rotation of the omni-wheels as unit vectors $M\overrightarrow{d_1}$, $M\overrightarrow{d_2}$and $M\overrightarrow{d_3}$.
The angular velocity of the ball relative to the body in the reference system M can be written as:

$$ {}_M\vec{\omega}_S = R_{MK} \cdot {}_K\overrightarrow{\Omega_S} - {}_M\overrightarrow{\Omega_B} \qquad (2.6) $$

Where $R_{MK}$is a vector from reference frame K to reference frame M.
By taking into consideration the no slip assumption, it follows that the speed on the surface of ball in the direction of omni-wheel has to be same speed as that of the tangential speed of omni-wheel. This can be expressed for each omni-wheel as:

$$ \left({}_M\vec{\omega}_S \times {}_M\overrightarrow{PS_i}\right) \cdot {}_M\overrightarrow{d_i} = {}_M\omega_{Wi} \cdot r_W \qquad \text{For i=1, 2, 3} \qquad (2.7) $$

In this equation, the r_W denotes the radius of omni-wheel. Now solving this equation for each omni-wheels *w1, w2* and *w3*, it can be seen that results will depend on these $\beta_x, \beta_y, \beta_z, \dot{\beta}_x, \dot{\beta}_y, \dot{\beta}_z, \dot{\alpha}_x, \dot{\alpha}_y$.

Now, ${}_M\vec{\Omega}_{W_1}$, ${}_M\vec{\Omega}_{W_2}$ and ${}_M\vec{\Omega}_{W_3}$ can be calculated. The solution for ${}_M\vec{\Omega}_{W_2}$ is given as an example.

Translation of the ball:

Translational speed of the ballbot must be known for the future calculations. Integrating the translational speed of the ball provides information for the exact location of the ballbot relative to the inertial reference frame *I*.

The vector from the ground surface G to the center of the ball P is represented in inertial frame as ${}_I\vec{GP}$. By knowing this vector, the speed vector of the center of the ball P can be expressed as:

$$ {}_I\vec{\dot{r}}P = {}_I\Omega_S \times {}_I\vec{GP} \qquad (2.8)$$

4.6 Parameters:Error! Reference source not found.

Before the equations of motion can be derived, some parameters need to be calculated.

Most of the parameters used in the 3D model are calculated and defined. But some additional parameters have to be calculated for the 3D model. Those additional parameters are the inertia tensors (i.e. the matrix that contains the moments of inertia and the inertia products about the three coordinate axes) of the ball and the body.

Parameters are shown in a Table 2.1**: parameters of the system**.

description	variable	values
Mass of the ball	m_S	0.625kg
Mass of the wheel and body	m_{BW}	6.71kg
Gear ratio	d_{gear}	17:1
Radius of the ball	r_S	0.120m

Radius of the omni-wheel	r_W	0.05m
Radius of the body	r_B	0.126m
Inertia of the ball	I_S	0.003606375kgm^2
Inertia of the body in x direction	$I_{B,x}$	1.4127814138kgm^2
Inertia of the body in y direction	$I_{B,y}$	1.412781311 kgm^2
Inertia of the body in z direction	$I_{B,z}$	0.053598646 kgm^2
Angle of omni-wheels contact point	θ	47
Angle of omni-wheels directions	$\varphi_1\varphi_2\varphi_3$	0,120,240
Height of the COG	l	0.22634m
Gravitational acceleration	g	9.81m/s^2

Table 2.1: parameters of the system

Calculation of moment of inertia:

It can be seen from table that inertia of the ball and body are expressed as I_S and I_B respectively. These inertias are the tensors and are given at the center of the ball. Here we consider the reference frame because in reference frame the principle moment of inertia become zero. These can be expressed as

$$ {}_I I_S = {}_K I_S = \begin{bmatrix} I_S & 0 & 0 \\ 0 & I_S & 0 \\ 0 & 0 & I_S \end{bmatrix} \quad (2.9) $$

$$ {}_M I_B = \begin{bmatrix} I_B & 0 & 0 \\ 0 & I_B & 0 \\ 0 & 0 & I_B \end{bmatrix} \quad (2.10) $$

In the body fixed coordinate system M, the moment of inertia for a motor with an attached omni-wheelI_W, is a scalar and is given specially for the rotation about the motor axis. In order to consider

the rotation of the motors and omni-wheels about all axes in the inertial reference frame I, their moment of inertia is simply added to the moment of inertia for the body. This simplifies the calculation for the energies of the omni-wheels, because now only the rotational energy of the omni-wheels needs to be calculated and the translational energy of the omni-wheels is not taken into account with the calculation of the translational energy of the body. This results in a simplified description for the system energies (see 4.7.1 Energies of system Energies of System).

However, with this simplification, the gyroscopic effects especially at high angular speed are not considered.

Of course, it is necessary to take into consideration the gear ratio between motor and omni-wheel. This Equation describes the relation between the moment of inertia of the motor I_M and moment of inertia of an omni-wheel I_{OW}.

$$I_W = I_{OW} + I_M \cdot i_{gear}^2 \tag{2.11}$$

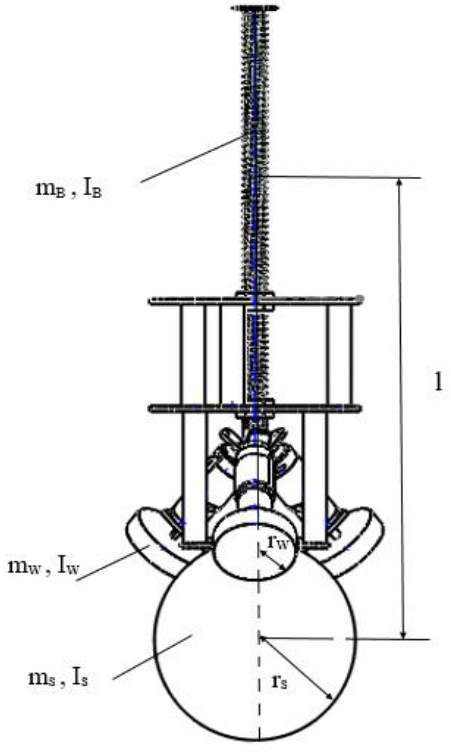


Figure 11.5 Geometric parameters of 3D system

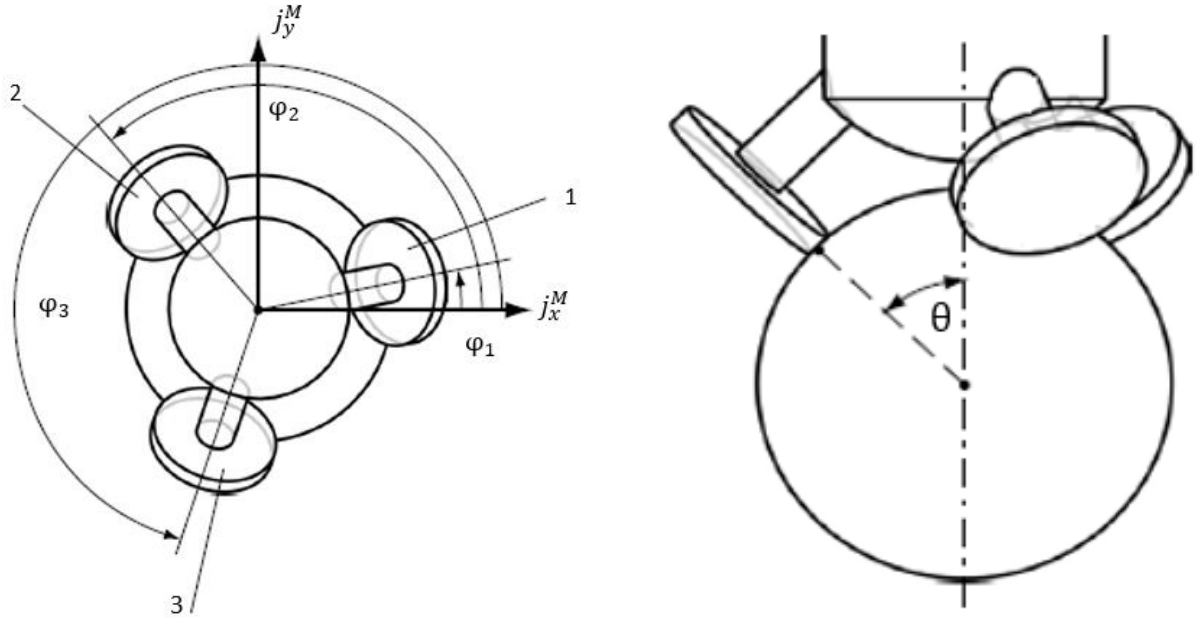


Figure 12.6 angles of motor configuration

4.7 Dynamics

Here we derive the dynamic equations of motion for the analysis, simulation and the design of the suitable controller. The Lagrangian method was selected as the common approach for that kind of problems so the minimal coordinate vector is represented as

$$\vec{q} = [\alpha_x\ \alpha_y\ \beta_x\ \beta_y\ \beta_z]^T \qquad (2.12)$$

4.7.1 Energies of system

For each body the kinetic (translational, rotational) and potential energies have to be calculated. The plane with zero potential energy is selected to coincide the center of the ball.

For ball

Ball kinetic energy consist of both translational and rotational energies. The potential energy of the ball is zero, because it is assumed that the ball only moves over horizontal surfaces:

$$V_S = 0$$

Kinetic is expressed as:

$$T_S = \underbrace{\frac{1}{2} \cdot m_S \cdot {}_I\dot{\vec{r}}_P^T \cdot \dot{\vec{r}}_P}_{Translation} + \underbrace{\frac{1}{2} \cdot {}_k\vec{\Omega}_S \cdot {}_kI_S \cdot {}_k\vec{\Omega}_S^T}_{\text{Rotation}} \; {}_I \qquad (2.13)$$

In this equation, the m_S denotes the mass of the ball.

The first part in this equation represents the translational part and second part represents the rotational part.

For body

As explained earlier in section 4.6 Parameters that in m_B and I_B the motors and omni-wheels masses are also included. For body coupling term is also included. It can be written as:

$$T_B = \underbrace{\frac{1}{2} \cdot m_B \cdot {}_I\vec{r}_P^T \cdot {}_I\vec{r}_P}_{\boldsymbol{Translation}} + \underbrace{m_B \cdot (R_{MI} \cdot {}_I\vec{r}_P) \cdot ({}_M\Omega_B \times {}_M\vec{r}_{PSM})}_{\boldsymbol{Coupling}}$$

$$+ \underbrace{\frac{1}{2} \cdot {}_M\vec{\Omega}_B^T \cdot {}_M I_B \cdot {}_M\vec{\Omega}_B}_{\boldsymbol{Rotation}} \qquad (2.14)$$

In this equation, the m_B denotes the mass of the body and ${}_M\vec{r}_{PSB}$ denotes the vector from center of the ball P to the center of the gravity of body SM in the reference system M. The point P is

chosen as the reference point at the center of the ball but it's not the center of the mass in comparison to calculation of the ball.

The potential energy of the body is defined as:

$$V_B = -m_B . \vec{G} . R_{MI} . {}_M\vec{r}_{PSM} \qquad \text{where} \quad \vec{G} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \qquad (2.15)$$

For omni-wheels

The kinetic energy of omni-wheels depends upon only on the rotational part. As a result of the simplification made in section 4.6 Parameters:, only the rotational energies have to be considered.

$$T_{w_i} = \frac{1}{2} . {}_M\Omega_{w_i}^2 . {}_M I_{w_i} \qquad \text{for} \quad i = 1, 2, 3 \qquad (2.16)$$

4.7.2 Non-Potential forces $\vec{f}_{NP}$

Non-potential forces are actually those forces that are applied externally on the system. In this case, the external torques applied on the system are non-potential forces. The system is actuated by the omni-wheels and motors, so the external torques are the torques of the motors τ_1, τ_2, τ_3. which transfer a torque to the omni-wheels.

They act directly on $M\omega_{W1}$, $M\omega_{W1}$, $M\omega_{W1}$. A polynomial separation has to be done because $\dot{\gamma}_1$, $\dot{\gamma}_2$, $\dot{\gamma}_3$ are not the part of the minimal coordinate vector.

$$ {}_M\omega_{Wi} = J_{\tau i} . \begin{bmatrix} \dot{\alpha}_x \\ \dot{\alpha}_y \\ \dot{\beta}_x \\ \dot{\beta}_y \\ \dot{\beta}_z \end{bmatrix} \qquad \text{for} \quad i = 1, 2, 3 \qquad (2.17)$$

There are also the counter torques acting on the body. There action on body is opposite to the direction of torquesτ_1, τ_2, τ_3.the counter torques are denoted as $\vec{\tau}_{C,1}, \vec{\tau}_{C,2}, \vec{\tau}_{C,3}$ and they can be expressed as

$$\vec{\tau}_{C,1} = \frac{{}_M\overrightarrow{MW_i}}{\left\|{}_M\overrightarrow{MW_i}\right\|} . (-\tau_i) \qquad \text{for} \qquad i = 1, 2, 3 \qquad (2.18)$$

The Jacobian matrix associated with the counter torques is derived in (2.4) and here is denoted as $\underline{J}_{\tau C}$.Hence, the non-potential forces f_{NP}can be written as

$$f_{NP} = \underline{J}_{\tau 1}^T . \tau_1 + \underline{J}_{\tau 2}^T . \tau_2 + \underline{J}_{\tau 3}^T . \tau_3 + \underline{J}_{\tau C}^T . (\vec{\tau}_{C,1}+\vec{\tau}_{C,2}+\vec{\tau}_{C,3}) \qquad (2.19)$$

4.8 Equations of motion

4.8.1 Lagrangian Approach

The equation of motion is solved using the lagrangian approach. The lagrangian approach can be applied as follows:

- Find the kinetic energy (T) and potential energies (V) of all the rigid bodies and expressed them using minimal coordinates.
- Express all external (or non-potential) torques (denoted as$\vec{f}_{NP}$) as functions of the minimal coordinates.
- Now, define the lagrangian as $L(q,\dot{q}) = T - V$ (q is defined as vector of minimal coordinate)
- Now calculate the Euler lagrangian equation as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vec{q}}}\right) - \left(\frac{\partial L}{\partial q}\right) = \vec{f}_{NP} \qquad (2.20)$$

The lagrangian equation of motion cane be defined as:

4.8.2 Lagrangian equations of motion

The lagrangian equation of motion cane be defined as:

$$\mathrm{L}\,(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathrm{T} - \mathrm{V} \tag{2.21}$$

$$T = T_S + T_B + T_{W1} + T_{W2} + T_{W3} \tag{2.22}$$

$$V = V_B \tag{2.23}$$

$$\mathrm{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T_S + T_B + T_{W1} + T_{W2} + T_{W3} - V_B \tag{2.24}$$

The Euler lagrangian equation can be solved by this equation

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vec{q}}}\right) - \left(\frac{\partial L}{\partial q}\right) = \vec{f}_{NP} \tag{2.25}$$

It can also be expressed in matrix notation form as:

$$\mathrm{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathrm{C}\,(\boldsymbol{q}, \dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \mathrm{G}(\boldsymbol{q}) = \vec{f}_{NP} \tag{2.26}$$

Because of the complexity of these matrices these are not shown here. At this point it is important to note that these calculations could not be done directly in a software package like Mathematica. Instead, the calculations had to be separated into multiple, smaller computations and simplified after each step.

4.9 System analysis

4.9.1 Nonlinear system:

The equation of motion that we have derived in section 2.6.3 are non-linear ordinary differential equations. They consist of thousands of terms and cannot be overviewed by any means.

At first, the 3d models also measured the rotation of the ball about the vertical axis and used different minimal coordinates. So, its analysis is limited to a linearized system because the size of this complex system is very large.

4.9.2 Linearized system

For balancing the ballbot, the equations of motion are only needed around the position of the ballbot to stands upright. Therefore, a linearized model is proposed, which is used to model the dynamics of the ballbot. First, the state vector $\vec{x}$ and input vector $\vec{u}$ and the output vector $\vec{y}$ need to be defined.

The process of normalization is left out, because all states, inputs and outputs have the same order of magnitude.

$$\dot{\vec{x}} = \mathrm{A}.\vec{x} + \mathrm{B}.\vec{u} \qquad \vec{y} = \mathrm{C}.\vec{x} + \mathrm{D}.\vec{u} \qquad (2.27)$$

$$\vec{x} = \left[\alpha_x\ \dot{\alpha}_x\ \alpha_y\ \dot{\alpha}_y\ \beta_x\ \dot{\beta}_x\ \beta_y\ \dot{\beta}_y\ \beta_z\ \dot{\beta}_z\right]^T \qquad (2.28)$$

And define the input as

$$\vec{u} = \begin{bmatrix} \mathrm{u}_1 \\ \mathrm{u}_2 \\ \mathrm{u}_3 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \qquad \vec{y} = \vec{x} \qquad (2.29)$$

The system output shows that output is equal to states because all the states are measured with sensors.

Linearization at zero

The system will be linearized around the unstable equilibrium, where all state variables and inputs are zero.

At first ballbot is controlled to keep the body upright and position constant. A ballbot has an unstable equilibrium and it can tilt in any direction that's why linearization at zero is required.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 66.15953177404623 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 66.15953479725454 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -34.25807548540608 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -34.25807705085397 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.30)$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 7.792242049446661 & -3.896121024723329 & -3.896121024723329 \\ 0 & 0 & 0 \\ 0 & 6.748279875625707 & -12.379086677896355 \\ 0 & 0 & 0 \\ -3.4137308966058617 & -3.4137308966058617 & -3.4137308966058617 \\ 0 & 0 & 0 \\ -13.430804433242663 & 6.715402216621331 & 6.715402216621331 \\ 0 & 0 & 0 \\ 0 & -11.631417992124659 & 14.54710677200806 \end{bmatrix} \quad (2.31)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.32)$$

$$D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.33)$$

Where C is the 10x10 identity matrix and D is 10x3 zero matrix.

The controllability and observability of this state space model can be checked in Matlab and it's completely observable and controllable and they have full rank.

Equation no (2.30) & (2.31) shows matrices A and B for linearized 3D system at zero and it can be split into three independent planar systems. The yz- and xz-plane are marked as red and green on matrices A and B and these are identical planes. And the xy-plane is marked as blue. The slight numerical difference in A for the yz- and xz-systems originates in the difference of their respective moment of inertia.

By observing the matrix B of the system it can be seen that motor effect on each system is not same. For example, input u_1 has an effect on the yz system while u_2 and u_3 has a smaller and opposite effect on that system. Similarly, the u_1 has no effect on the xz system.

4.10 Development of a 3D model based controller

In this section, a controller will be designed, based on the derived dynamical 3D model. The controller will control the 3D orientation of the body and the position of the ballbot, which includes both station keeping and tracking of given set points.

A model based, LQR (linear quadratic state feedback controller) is designed that controls the orientation of the body and the position of the ballbot simultaneously. In order to apply LQR, full state feedback is used. In many systems, it is very difficult to measure all the states.

4.10.1 Controller Design

For the 3D model a linear quadratic state feedback controller (LQR) is designed. In order to apply LQR, full state feedback is used. In many systems, it is hard to measure all states and therefore a model based observer has to be used. If all states are available, the solution is

$$u(t) = -K * x(t) \qquad \text{where} \qquad K = R^{-1} B$$

Since all states are estimated by the sensors in the system discussed, the application of an LQR controller is straightforward. The weighting matrices Q and Rare experimentally tuned (view section 4.10.2 Tuning on the Real System). For given Q and R matrices, the controller gains K can be derived using the MATLAB lqr command. Since the LQR controller is designed for a set point equal to zero, an additional factor to the set point is needed, in order to achieve a correct tracking behavior.

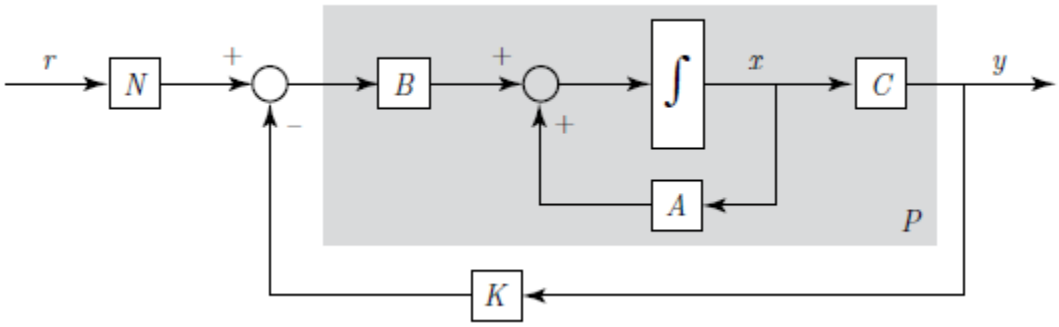


Figure 13.7 lock schema of the LQR controller with the linear model P

4.10.2 Tuning on the Real System

For linearized system where all operating points are at zero. The Q and R matrices which were chosen initially worked well for simulation but can't be implemented on the system because of the noise on gyroscope is amplified by the controller. Due to noise the system starts to shiver.

Therefore, the controller gain is minimized and R is increased. The final weighting matrices Q and R for this model are given below.

Q = diag (100, 50,100, 50, 40, 20, 20, 10, 20, 10) (2.35)

R = diag (100, 100, 100) (2.36)

The values for Q and R is experimentally tuned. R is chosen in such a way that input of the motors must remain within limits.

For β_z and $\dot{\beta}_z$ the values for Q are chosen in such a way that they do not influence the resulting gains for other dynamics and the ball does not slip bout vertical axis.

The controller gain K can be derived using MATLAB lqr command. The K matrix for position control is given as:

K =

$$\begin{bmatrix} 36.3008 & 11.7637 & -4.6874 & -1.5462 & -0.3100 & -0.2974 & 0.3650 & 0.5803 & -0.1368 & -0.1735 \\ -21.2440 & -6.8628 & 15.2996 & 4.9905 & -0.5464 & -0.6143 & -0.1828 & -0.2903 & 0.1315 & 0.2697 \\ -11.7554 & -3.8546 & -24.8157 & -8.1378 & -0.0727 & 0.0210 & -0.1827 & -0.2912 & -0.4050 & -0.6185 \end{bmatrix} \quad (2.37)$$

In order to improve the performance of the controller, a finite impulse response (FIR) low-pass filter of fourth order with a cutoff frequency of approximately 15Hz (view section **5.3.2** Low **Pass filter**is applied on the controller output. Therefore, the gains can be increased, while the system remains stable. By applying the linear controller all the ten states converge as shown in fig_4.7

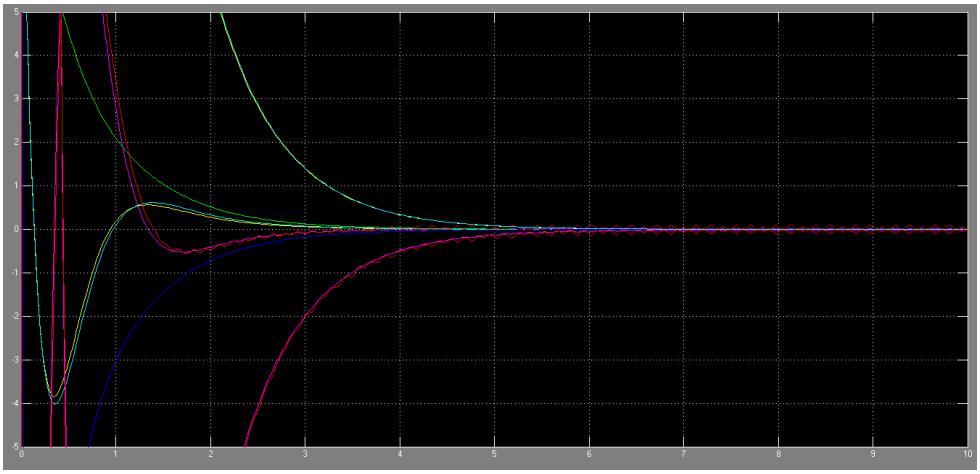


Figure 14.8 all ten converging states of the system

4.11 Design of a nonlinear controller

Because the system has strong nonlinearities so it's better to design a nonlinear controller. In this section, we will design a suitable nonlinear controller for the 3D model.

4.11.1 Concept

For this nonlinear controller, first of all the linear controller is designed and then 2nd step is to design a linear controller at different operating points other than zero. And then for the current state the controller gains are inserted.

The nonlinear controller is established for the velocity controller, because most of the input generating projects use this type of controller. These states $\dot{\alpha}_x, \dot{\alpha}_y, \dot{\beta}_x, \beta_y, \dot{\beta}_y, \dot{\beta}_Z, u_1, u_2$ and $u_3$caused the nonlinearities in the system while the states β_y β_y $\dot{\alpha}_x, \dot{\alpha}_y$are having the strongest influence on the system. To design a full nonlinear controller, the 24 entries of the velocity state feedback controller matrix K have to be inserted in these 10 dimensions.

4.11.2 Implementation

Now implementing this controller on the system model. First of all, we examine the nonlinearities in the system resulting from the velocity of the ballbot.

The primary reason that the linear controller is not suitable for real systems is its inability to control β_z at high speed**s, meaning over _2 m/s.** The ballbot at this speed starts oscillating about z axis that leads to instability of the system and we need to avoid this instability. The comparison between the performances of the linear and nonlinear controllers is shown in fig_4.8 below. The plot on the left side of the fig. shows the behavior of linear controller. The set points are same for both the controllers. Set points for the velocity in y direction are steep points from 0 to 2 m/s and flat ramp forβ_z.Additionally disturbances are also given for the α_y. The plot for the nonlinear controller is

given to the right side of the fig.. It can be shown that this controller ensures the system stability based on linearization points for the$\dot{\alpha}_x, \dot{\alpha}_y$. The controller that is shown here the same is implemented on the system and this controller works perfectly.

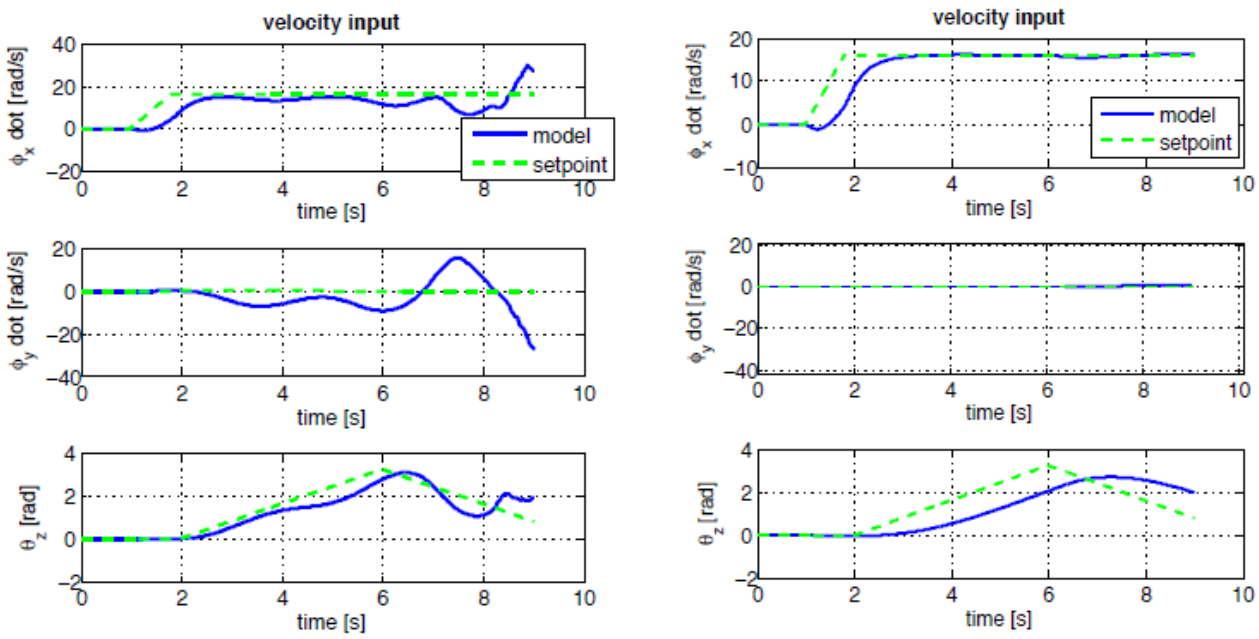


Figure 15.9 Performance of the linear (left) and the nonlinear controller (right) for the same set points

Chapter 5

5. Simulation of a 3D model

In the previous chapter a complete 3D dynamical model was derived and linearized around the position of the ballbot to stands upright. In this chapter, the simulations with this 3D model is done to analyze the nonlinear 3D model. Furthermore, the accuracy of the linearized model is investigated.

5.1 Implementation in Simulink

The nonlinear equations of the 3D model are applied in Simulink as in fig_5.1. The testing of the controller is also done using the Simulink. Therefore, the simulation should be appropriate for the reality. The model is implemented as S-function, Because of the size of the equations of motion, which is improved to make a real-time simulation possible. Whenever possible all other functions are implemented as MATLAB-functions. The similarity between the MATLAB-functions in the simulation and the C-functions in the real system is the basic advantage of this method. Therefore, changes made in the simulation are transferred to the ballbot fast and with a low risk of relocation errors.

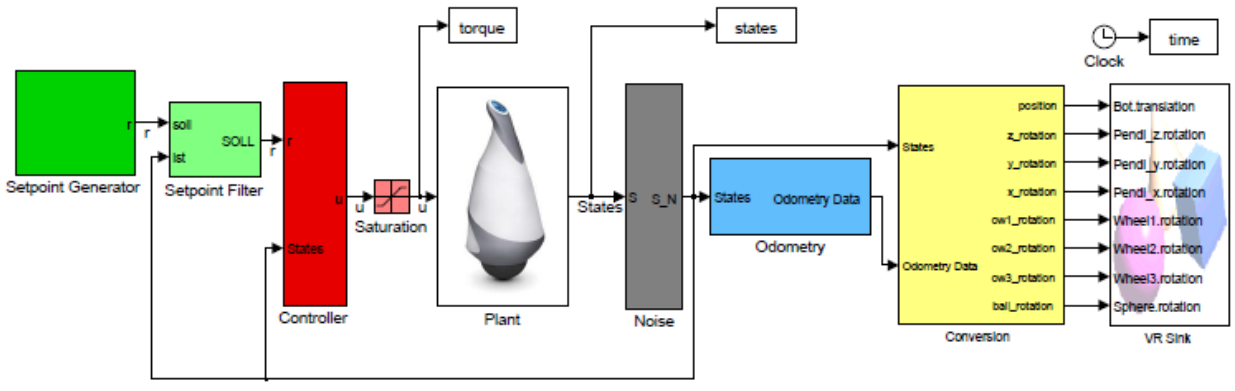


Figure 16.1 Block diagram of 3D in Simulink

5.2 Error Model

The effect of noise that acts on the system and the delay occurring in the system can be proposed by error model of the system. They are simulated in Simulink.

5.2.1 Noise

The only noise that is present in the system is the sensor noise from the gyroscope (IMU) and from the angle estimation. By rotating the IMU constantly around each axis separately the IMU noise is measured. To estimate the sensor noise, assumptions are made such that rotation contains no high frequency oscillations, so by subtracting the rotation speed from the sensor noise the magnitude of the noise is estimated. The mag measured in the test is shown in fig_5.3. This data is then added to the angle rate as in fig_5.2.

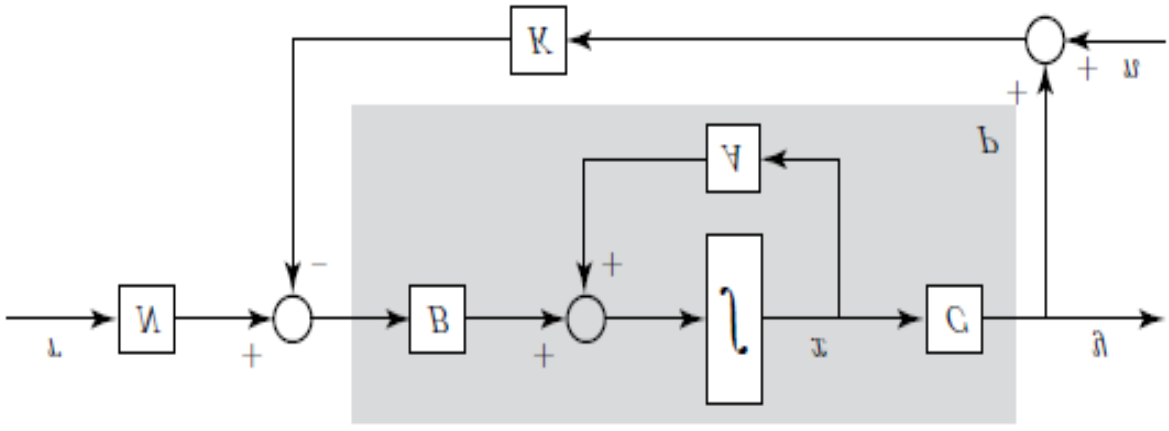


Figure 17.2 sensor noise n added in LQR feedback control schema

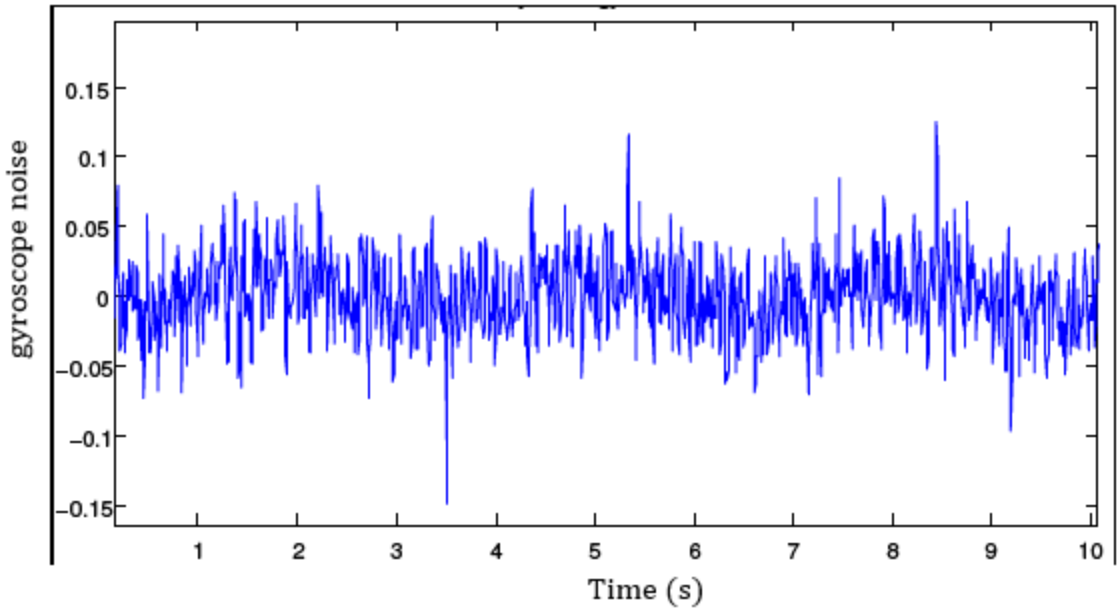


Figure 18.3 showing the noise effect on the data measured from the IMU gyroscopes

5.2.2 Delay

As already discussed that noise in the system is due to sensor noise and delay. A delay of 0.01s is added to the plant. Delay of 0.007 is produced by sampling rate of only 160 Hz. This delay in the plant signifies the delay in the real system.

5.3 Filtration

Sensor integration is the process of combining multiple sensor responses together into a single, more reliable output signal. In the sensor integration of an accelerometer and a gyroscope, the resultant data provides a much more reliable tilt angle value. This is achieved as the system has now compensated for gyroscope drift that would otherwise provide imprecise indications. The majority of sensors have a degree of unreliability that becomes inherent over time as well as introduced noise which can be overcome by these filtration processes.

In this section, different methods of filtration are explained that are used to solve the problem of the system noise. Previously it has been explained that noise is the main reason why the controller cannot be modified in the desired way. As discussed earlier the noise is limiting the controller gain and the performance of the controller. To overcome this noise in the system different solutions are tried.

5.3.1 Offset

Because of the friction, the motors do not start turning unless they get more than a specified amount of torque as input. The current needed to start the motors turning is experimentally determined and amounts to approximately 80 mA, what corresponds to input torque of 42 mNm. An offset with that threshold was tested on the ballbot without improvement and is therefore not used.

5.3.2 Low Pass filter

A low pass filter is designed to filter the noise. A low pass filter is applied at the output of the controller because the noise on the gyroscope values makes the output of the controller very noisy that causes shivering of the system and it can also be observed from the fig_5.3. This low-pass filter is designed with the Filter Design Toolbox of Matlab, which uses either an Infinite Impulse Response (IIR) filter or a Finite Impulse Response (FIR) filter as design method. It is chosen to design a FIR filter, because it can better address specific frequencies. It’s a fourth order finite impulse response filter (FIR) with cutoff frequency of approximately 15 Hz.

5.4 Verification of the 3D model

In this section, the simulation of a 3D model is verified. The responses of real system and simulation of 3D model to a position step can be shown in fig_5.6. The set point step on α_x from 4 rad to -8 rad corresponds to a position difference of 1.5 m. the result of the step responses fulfill the expectation. Because the real system is not able to stand upright perfectly so there will be deviation between the measured data and simulated data. The controller we used is totally model based controller and its performance is very good and close to the expectation. Therefore the model is a realistic representation of the system.

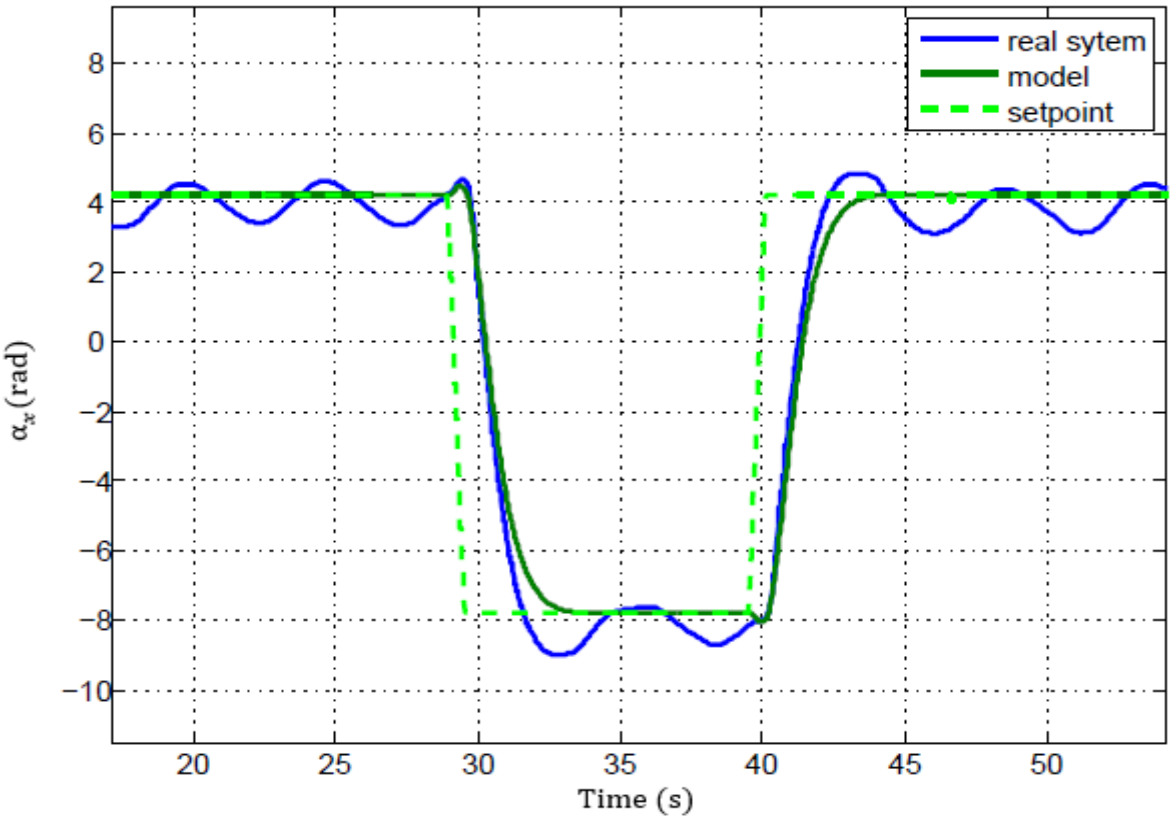


Figure 19.4 System and model response on a position step

As we know that the main difference between the 2D model and 3D model is the coupling effect that is considered in 3D model. Fig_5.7 shows the coupling effects occurring in the 3d model so moving in x direction and then turning about z axis does not affect the y direction.

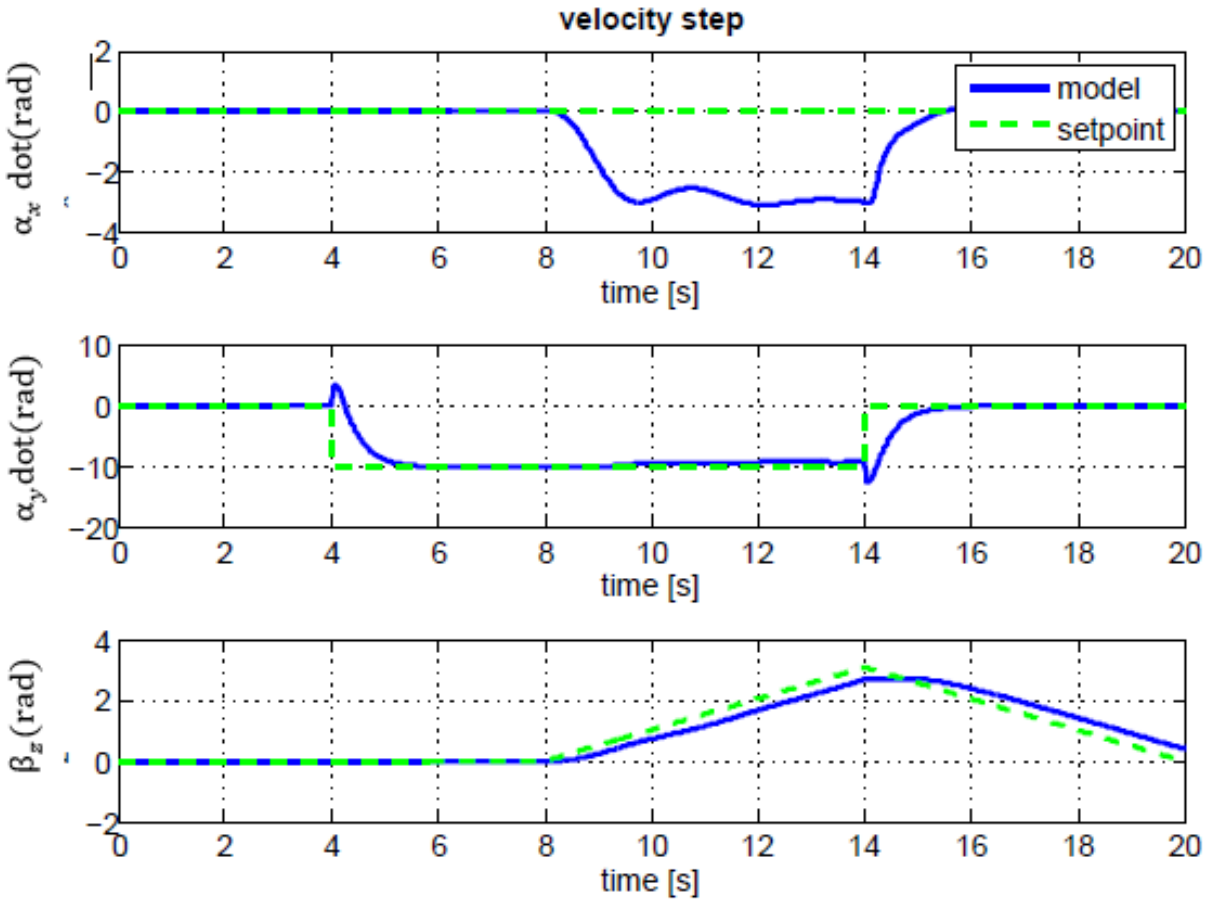


Figure 20.5The Response of the model on a velocity step in x direction (center) and a ramp on the rotation Around the z-axis (bottom).

Chapter 6

6. Performance Analysis

In this chapter, the performance of controller is analyzed and Tests are performed on different layers. The performance is slightly better on carpets because of the more damping and contact friction than on floors.

The motion of the ballbot can be observed while balancing on a single point. For performing this kind of task, we used a controller allows stronger gains and low pass filtration at the output. The ballbot can deviate slightly from its initial point by standing on a single point for a period of time.

The speed test can also be performed on the system. The maximum speed that is achieved during the test can also be plotted. The rotation speed of ball is represented in rad/sec then it's converted into the m/s by multiplying with radius of ball.

The maximum tilt angle that can be reached in this ballbot is 12^{o}. This tilt angle is bigger and this corresponds to the performance of the controller. The tilt angle relate directly with the acceleration of the ballbot.

In this section of performance analysis, all the test were done using the linear controller for the 3D model. By using the nonlinear controller higher performance can be achieved as described in section no 4.11 Design of a non-linear controller.

Chapter 7

7. Implementation

7.1 Set point filter

Set point filter is applied at the inputs of the system. To confirm that the set points chosen for the controller are suitable or not, the inputs are filtered by some set point filter. This filter is used to filter the input set points. This filter makes a ramp for each input set points and limits its maximum value as shown in fig_7.1.

This filter has two parameters to be adjusted:

- Maximum inclination of the ramp
- Maximum allowed input

The filter works the way, that if the set point difference is higher than the maximum allowable difference, the allowable difference is added to the last set point in the direction of the required set point. This is done every time before the set points are sent to the controller.
The ramp inclination value is selected at a point at which the torque of the motors reach their maximum value. The velocity of the system is also limited to a value at which the linear controller stabilizes the system. (See section 4.11 Design of a non-linear controller).

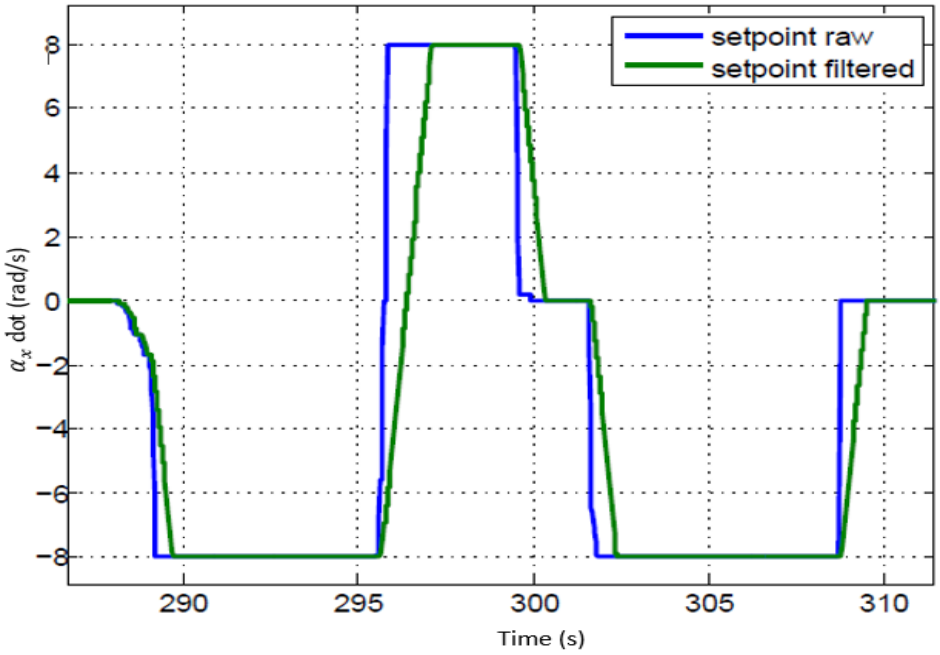


Figure 21.1 filtered values of set points after set point filter

7.2 Programming

The controller that we have designed in section 4.10.1 Controller Design is now implemented on the microcontroller. We used myRIO controller for our system.

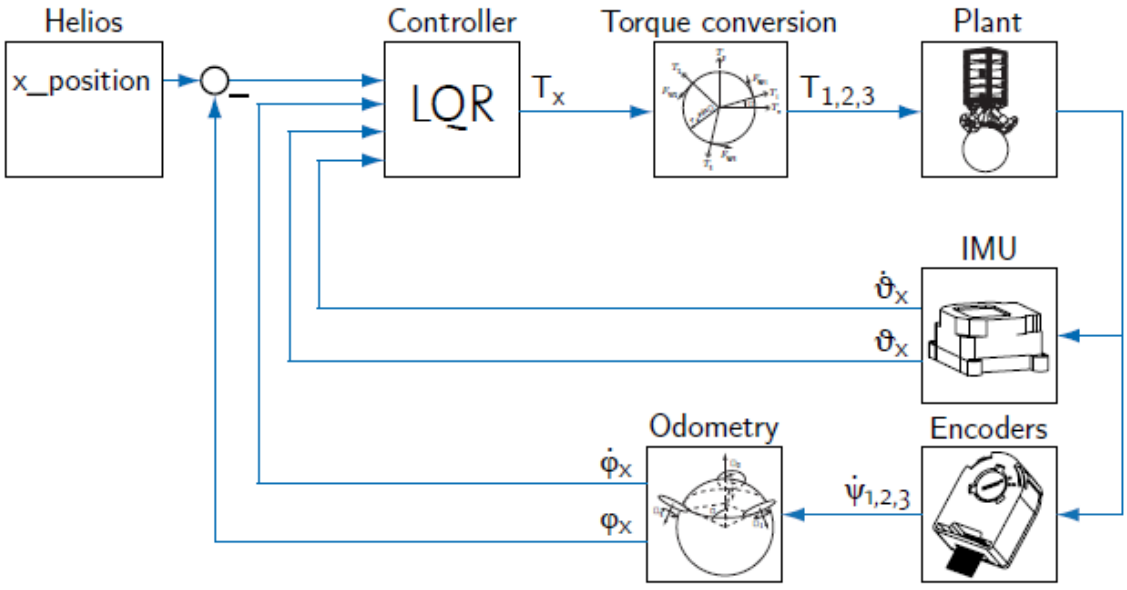


Figure 22.2 implemented control system

An overview of control systems that we implemented for the linear controller is shown in fig_7.2. The IMU and the encoder of the motors measure the states of the plant. In order to match the states from the controller the latter calculations have to be converted with the odometry. These states are then processed by the controller and to deliver the appropriate set point for the motor controllers these states also converted in the torque conversion.

Integration for $\alpha_{x,}$ $\alpha_{y \text{ and}}$ α_{z} is realized as numerical integration in trapezoidal form.

Chapter 8

8. Estimation of Ball-bot

8.1 Need of the Observer Design?

Initially we implemented the LQR controller on the Ball-bot. The response of the system become jerky and system starts to tremble. The jerky response of the system is due to the noise in the IMU sensor. Noise is further implemented due to the system controller. Our system behaves in jerky manner. To make the system response smooth and gentle, we design observer for the Ball-bot. In our case all states of system are assessable so we are free to implement Full state feedback law.

8.2 Controllability

If system input u(t) transfer any initial state $x(t_0)$ to a final state in a finite interval of time then we can say that our system is controllable.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$

The controllability can be checked by the simple equation.

$$\text{rank}[\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \ \ldots \ \mathbf{A}^{n-1}\mathbf{B}] = n$$

This equation is true for the **A** of n×n and **B** of n×1. For the multiple input system, **B** should be of n×m. For a single input and output system controllability can be checked by following equation.

$$\mathbf{P}_c = [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \ \ldots \ \mathbf{A}^{n-1}\mathbf{B}]$$

Find the determinant of P_c if the determinant is non-zero then system is controllable. Otherwise system will not be observable.

If states are not completely controllable and these states are inherently stable then our system will call stabilizable. In Ball-bot all states are controllable, so we have a completely controllable system.

8.3 Observability

If the initial states of the system x(0) can be found by the observation history in a finite time and there is control signal u(t) then system is called observable.

Consider a single input and output signal:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad \text{and} \quad y = \mathbf{C}\mathbf{x},$$

Where C is of 1×n and states x are of n×1. To check a system is observable or not we use the following observability matrix.

$$\mathbf{P}_o = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

Calculate the determinant of P_o if determinant have some non-zero value then system must be observable.

Like in controllability, if some states of the system are not completely observable. But these states are inherently stable then we call it as a detectable system.

8.4 Full State Feedback Law

State variable design problem is divided into two parts

1. Full state feedback design

2. Observer design

The first step is to check that all states are assessible and behave independently. In other words, system should be controllable.

The whole feedback control law based on the following equation.

$$u = -\mathbf{K}\mathbf{x}.$$

Determine the K is the objective of Full state feedback law. K is the gain matrix of the full state feedback law.

If a system is given as in its simplified form.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u,$$

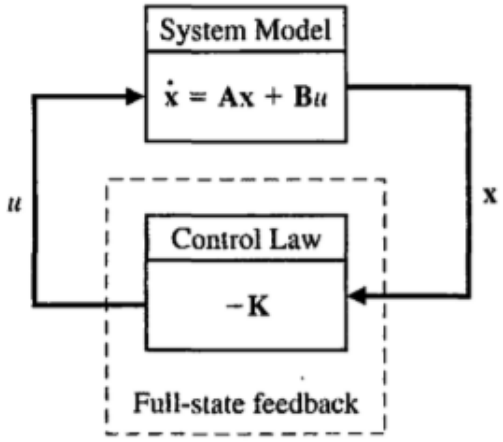
Then closed loop system can be shown as given below

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}.$$

The characteristics equation associated with that equation will be given as below.

$$\det(\lambda\mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K})) = 0.$$

If the poles of the system lie in the left half plane then the closed loop is stable. To meet the stability of system one condition must be meet and that is the controllability of the system.



State feedback gain matric can also be found by using these formulas

$$\mathbf{K} = [0 \quad 0 \quad \dots \quad 0 \quad 1]\mathbf{P}_c^{-1}q(\mathbf{A}),$$

$$q(\mathbf{A}) = \mathbf{A}^n + \alpha_{n-1}\mathbf{A}^{n-1} + \cdots \alpha_1\mathbf{A} + \alpha_0\mathbf{I},$$

8.5 Observer Design

When you have checked that your system is observable then you have to design your observer. If the we have a single input and output system as given below

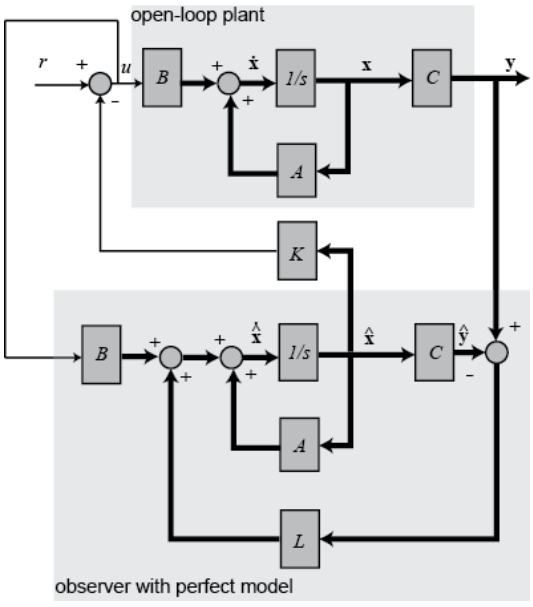
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x}$$

Then the observer for the system is given as

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}u + \mathbf{L}(y - \mathbf{C}\hat{\mathbf{x}})$$

Where $\hat{x}$ shows the estimated states and L show the gain matrix of observer which we have to design. Observer has two input u(t) and y and have one output $\hat{x}$.

Simple design of observer with a system is shown in the diagram.



In the figure, we can see that same system inputs are given two the system and the observer. The goal of the observer is to provide an estimate for the states. Initially we give a state x(t) and observer try to converge $\hat{x}$ to these states in an infinite time and try to make the error is equal to zero.

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t).$$

If we define the observer in terms of error that means by passing the time the error should be converge to zero. Observer in the form of error can be shown as.

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{x} + \mathbf{B}u - \mathbf{A}\hat{\mathbf{x}} - \mathbf{B}u - \mathbf{L}(y - \mathbf{C}\hat{\mathbf{x}})$$

$$\dot{\mathbf{e}}(t) = (\mathbf{A} - \mathbf{LC})\mathbf{e}(t).$$

The characteristics equation to find the L is given as

$$\det(\lambda\mathbf{I} - (\mathbf{A} - \mathbf{LC})) = 0$$

The roots of the equation must lie in the left half of the plane. There is only one condition to implement that equation is that all the states should be observable.

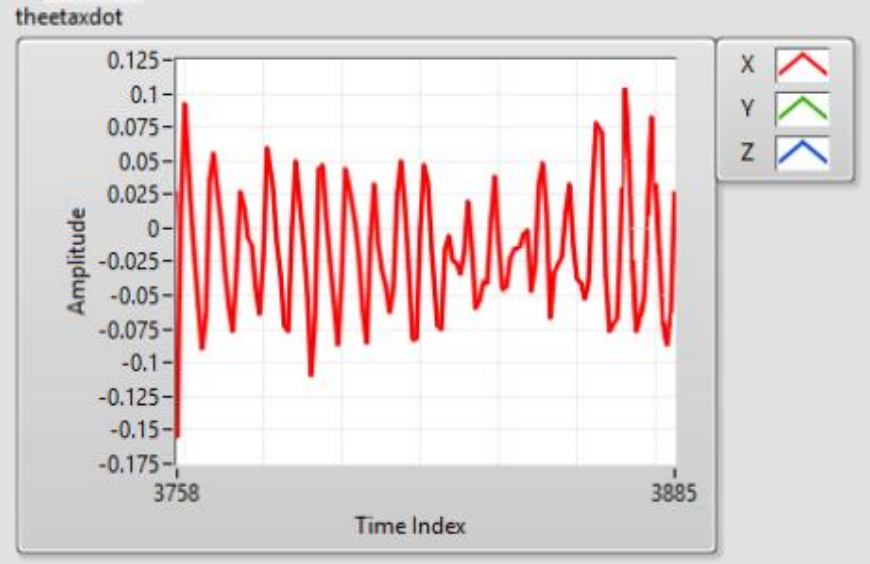
```
L=[0.1254   0       0       0       0    0    -0.5739 0       0              0;
   0        0.7257  0       0       0    0    0       -0.1993 0              0;
   0        0       0.1561  0       0    0    0       0       -0.5361        0;
   0        0       0       0.7258  0    0    0       0       0       -0.1994;
   0        0       0       0       0    0    0       0       0              0;
   0        0       0       0       0    0    0       0       0              0;
   -0.1722  0       0       0       0    0    0.7883  0       0              0;
   0       -0.9967  0       0       0    0    0       0.2738  0              0;
   0        0      -0.2144  0       0    0    0       0       0.04           0;
   0        0       0      -0.9968  0    0    0       0       0         0.2738];
```

The observer gains calculated after implementing the complete system model of the ballbot is given above.

8.6 Need for an estimator?

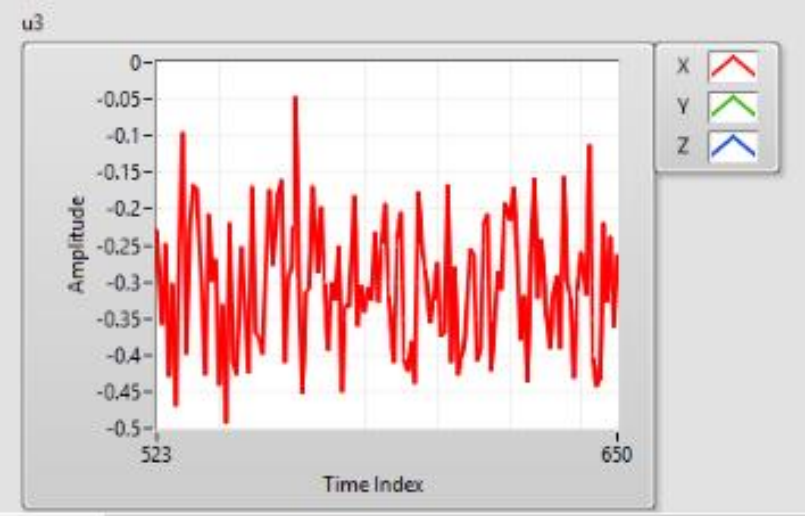
As we progressed in our project, apart from the mechanical issues faced, another issue arose related to the measurements of sensors i.e. gyroscope and encoders.
No matter how good the code we formed, the sensor value, especially the values taken from gyroscope which includes angular displacement and angular acceleration, were not consistent and the generated ripples were creating a huge issue as the whole control model is based on these sensor values.

The graphs of the gyro values without a proper estimator are shown as:



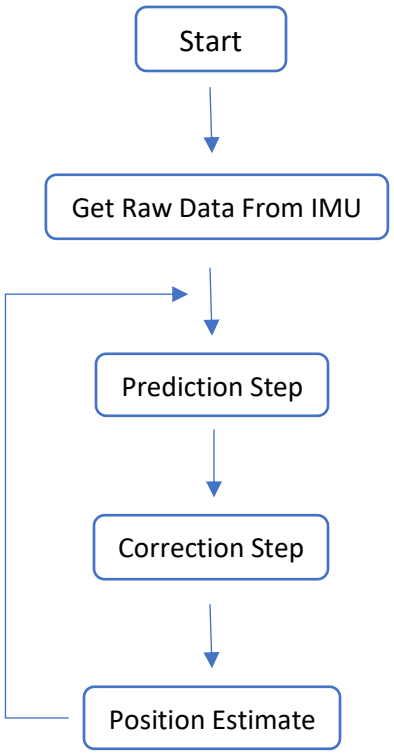
In the above given graph is generated while the Ballbot was stationary. As can be seen in the above given graph, the ripples generating are unacceptable where as in ideal state, the graph should be a straight horizontal line at 0.

The outputs generated, in the form of system torques from system model, using direct values from the gyroscope are:



Similarly one of the torques generated as output is shown in the graph above, and as like angular velocity, this output is generated while keeping the robot stationary and vertical straight upward and in ideal condition the torque output should be a horizontal line at 0 with no or minor ripples. Although simple filter can be used to filter out the ripples formed in the sensor values but an estimator based on Kalman Filter provides an optimum solution for the problem faced.

The flow chart for the proposed solution is given as:



8.7 Kalman Filter

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation.

A Kalman filter is an optimal estimator – i.e. infers parameters of interest from indirect, inaccurate and uncertain observations. It is recursive so that new measurements can be processed as they arrive.

The process of finding the "best estimate" from noisy data amounts to "filtering out" the noise. However, a Kalman filter also doesn't just clean up the data measurements, but also projects these measurements onto the state estimate

8.7.1 Popularity of Kalman Filter

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) solution of the least-squares method. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The list is long when we start discussing the positive aspects of such a filter but a few are listed as:

• Good results in practice due to optimality and structure.

• Convenient form for online real time processing.

• Easy to formulate and implement given a basic understanding.

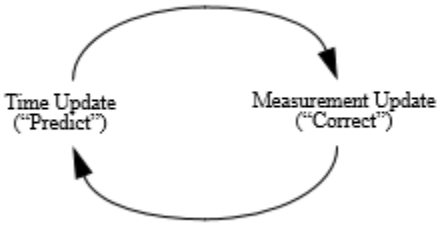
• Measurement equations need not be inverted.

Moving forward with the discussion of a Kalman Filter, we learnt and tried two types of Kalman filter to generate the optimum results required. Those two types are explained briefly later in this chapter.

8.7.2 Discrete Time Linear Kalman Filter

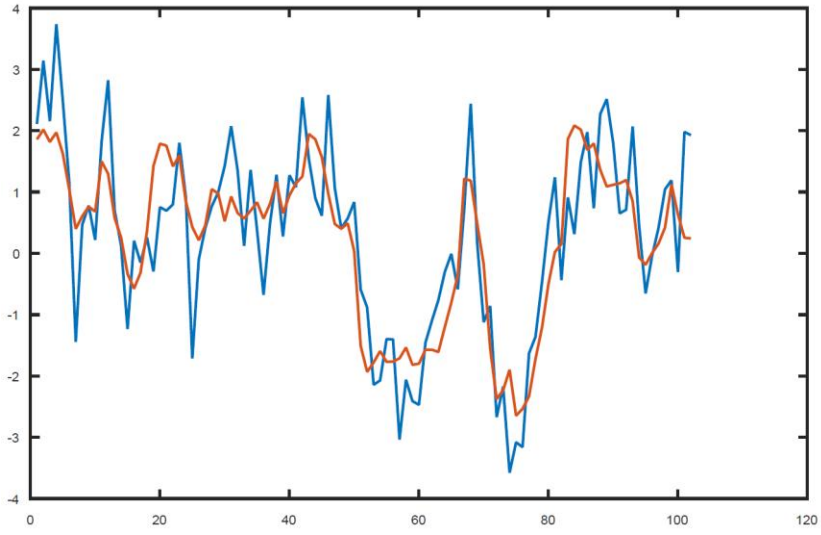
The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

The time update equations can also be thought of as predictor equations, while the measurement



update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems as shown below.

A sampled image showing the optimal results of a Discrete Time linear Kalman filter is shown as



Here, the blue line represents the measured values while the red line represents the output as given by Kalman Filter when measures values are processed through it.

8.7.2.1 Derivation

We begin the derivation of the discrete-time Kalman filter assuming that both the model and measurements are available in discrete-time form. Suppose that the initial condition of a state x0 is unknown, in addition suppose that the discrete time model and measurements are corrupted by

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \Gamma_k \mathbf{u}_k + \Upsilon_k \mathbf{w}_k$$
$$\tilde{\mathbf{y}}_k = H_k \mathbf{x}_k + \mathbf{v}_k$$

noise. The model for this case is given by:

where v_k and w_k are assumed to be zero-mean Gaussian white-noise processes, which means that the errors are not correlated forward or backward in time so that

$$E\left\{\mathbf{v}_k \mathbf{v}_j^T\right\} = \begin{cases} 0 & k \neq j \\ R_k & k = j \end{cases}$$

and

$$E\left\{\mathbf{w}_k \mathbf{w}_j^T\right\} = \begin{cases} 0 & k \neq j \\ Q_k & k = j \end{cases}$$

now rather than discussing the complete derivation of Discrete time Linear Kalman Filter, we'll focus on explaining how such a Kalman Filter is implemented, the whole process of implementing the filter can be done in five simple steps which are explained in further discussion.

8.7.2.2 Model generation

Starting with the 1st step, the model for input as well as output is generated using the equations:

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \Gamma_k \mathbf{u}_k + \Upsilon_k \mathbf{w}_k, \quad \mathbf{w}_k \sim N(\mathbf{0}, Q_k)$$
$$\tilde{\mathbf{y}}_k = H_k \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k \sim N(\mathbf{0}, R_k)$$

Here 'x_k' represents the input at discrete time 'k' and sigma represents the system matrix A, gamma represents the system matrix B while H_k represents the system matrix C.

8.7.2.3 Initializing

After the generation of input and output equations, the filter is initialized by providing different states at discrete time k=0, the states include input at time k=0 as well as error covariances P_0

$$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0$$
$$P_0 = E\left\{\tilde{\mathbf{x}}(t_0)\tilde{\mathbf{x}}^T(t_0)\right\}$$

Note that at this point both the prior error covariance P^- as well as posteriori error covariance P^+ are same and equals to P_0 as given above.

8.7.2,4 Gain Calculation

After initializing the input and error covariance, gain for the estimator is calculated. The matrix K is chosen to be the gain or blending factor that minimizes the posteriori error covariance. Now here one thing should be kept in mind that estimator gain is a different thing as that of system gain and will only be used while estimator is working. It should not be confused with the system gain.

The Gain matrix K is given as,

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$$

8.7.2.5 Updation

In this step, the system values are updated by comparing them with the measured values at discrete time k. The gain K of the estimator is used to minimize the error in the estimated value and the measured value. The updation equations for the input state x and posteriori error

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k[\tilde{\mathbf{y}}_k - H_k\hat{\mathbf{x}}_k^-]$$
$$P_k^+ = [I - K_k H_k]P_k^-$$

covariance P^+ are:

So until now, we see that error covariance P_k holds a lot significance in the Kalman filter. The error covariance is calculated using the Discrete Riccati Equation which is given as:

$$P_{k+1} = \Phi_k P_k \Phi_k^T - \Phi_k K_k H_k P_k \Phi_k^T + \Upsilon_k Q_k \Upsilon_k^T$$

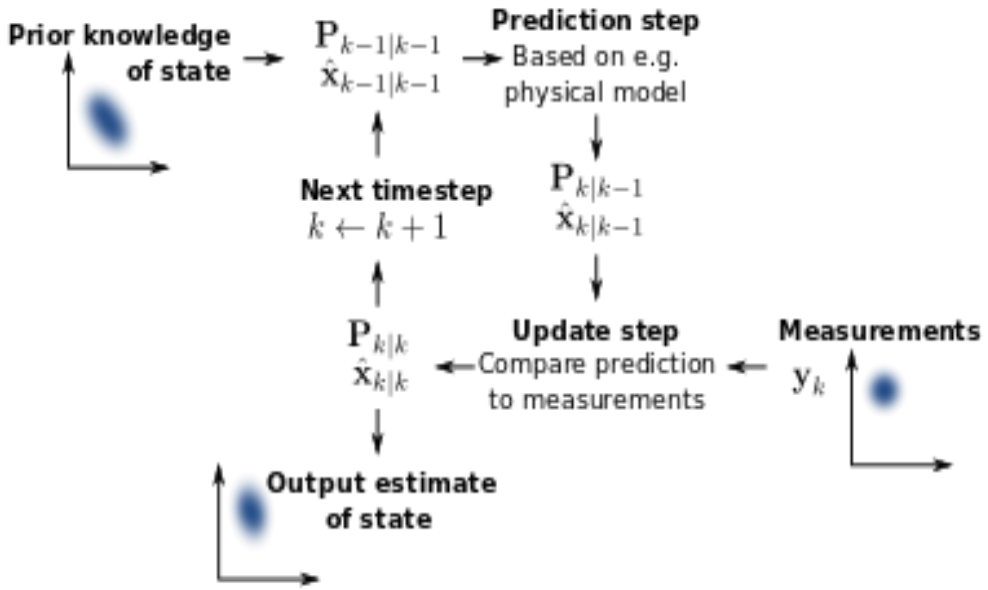
8.7.2.6 Propagation

After the process of updation, the 2nd main step of Discrete Time Linear Kalman Filter is that of propagation. In this step, states are generated following the trend of the system rather than depending on faulty input values. In this way, somewhat true states can be forwarded to control system rather than providing the system with faulty measured values.

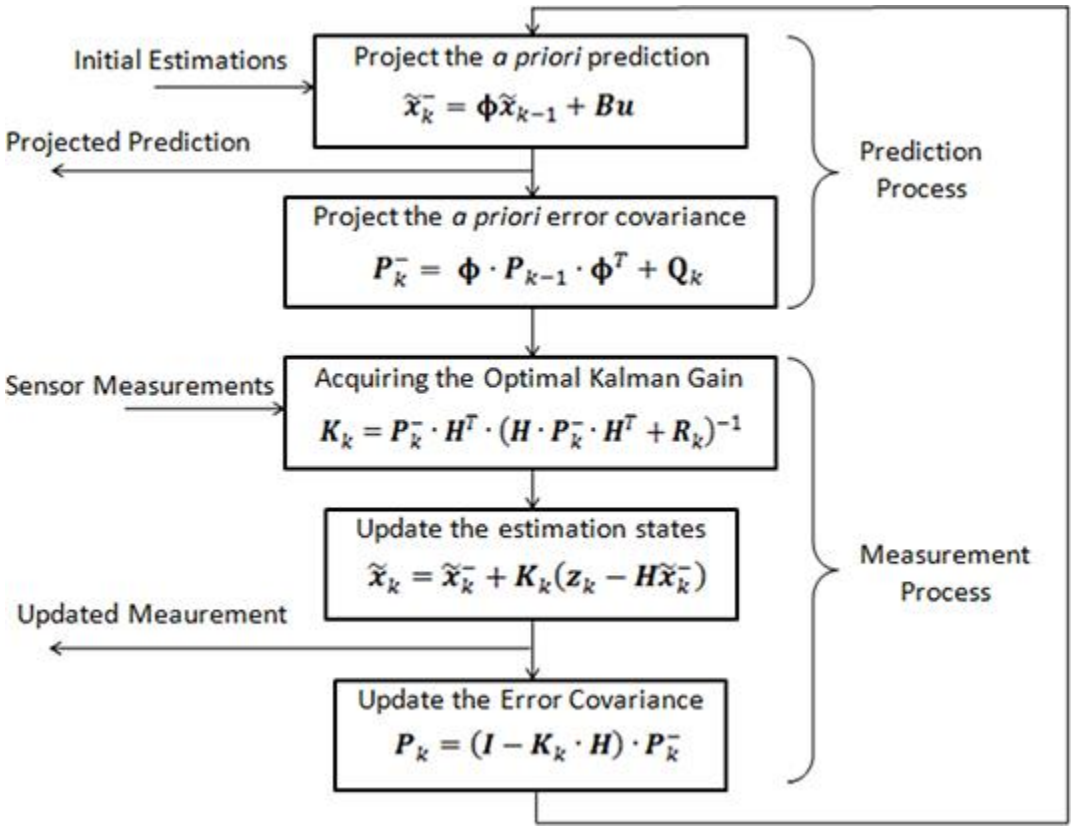
The propagated states are given by:

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k^+ + \Gamma_k \mathbf{u}_k$$
$$P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + \Upsilon_k Q_k \Upsilon_k^T$$

This process of propagation and updation keeps on going as discussed earlier while discussing the basic algorithm. The algorithm, in simple words, works in a way that it propagates the values to the control model based on the last updated trend of the system states. Then after few defined number of propagated values, the system states are compared with the propagated or estimated states and then the system is updated with this state.



The block diagram for the algorithm of Kalman Filter is:



8.7.3 Extended Kalman Filter

In estimation theory, the extended Kalman filter (EKF) is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. In the case of well-defined transition models, the EKF has been considered the effective standard in the theory of nonlinear state estimation, navigation systems and GPS.

8.7.3.1 Why extended Kalman Filter?

The assumptions of linearity for both the measurement and state transition are essential for the correctness of the Kalman filter. The observation that any linear transformation of a Gaussian random variable yields another Gaussian is important in the derivation of the Kalman filter algorithm, so what happens if state transitions and measurements are no longer linear?

Unfortunately, state transitions and measurements are rarely linear in practice.

A large class of estimation problems involve nonlinear models. For several reasons state estimation for non-linear systems is considerably more difficult and admits a wider variety of solutions than the linear problem. A vast majority of nonlinear models are given in continuous-time.

The extended Kalman filter, though not precisely “optimum,” has been successfully applied to many nonlinear systems over the past many years. The fundamental concept of this filter involves the notion that the true state is sufficiently close to the estimated state. Therefore, the error dynamics can be represented fairly accurately by a linearized first-order Taylor series expansion.

The common nonlinear truth model with continuous-time measurements is given as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + G(t)\,\mathbf{w}(t)$$
$$\tilde{\mathbf{y}}(t) = \mathbf{h}(\mathbf{x}(t), t) + \mathbf{v}(t)$$

where f(x(t), u(t), t) and h(x(t), t) are assumed to be continuously differentiable, and w(t) and v(t) follow exactly as in Discrete Time Linear Kalman Filter.

The problem with this non-linear model is that a Gaussian input does not necessarily produce a Gaussian output (unlike the linear case).

Still the basic flow of algorithm is same as that of Linear Kalman Filter and the equations along with the steps are shown in table below:

Model	$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + G(t)\,\mathbf{w}(t),\ \mathbf{w}(t) \sim N(\mathbf{0}, Q(t))$ $\tilde{\mathbf{y}}(t) = \mathbf{h}(\mathbf{x}(t), t) + \mathbf{v}(t),\ \mathbf{v}(t) \sim N(\mathbf{0}, R(t))$
Initialize	$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0$ $P_0 = E\left\{\tilde{\mathbf{x}}(t_0)\,\tilde{\mathbf{x}}^T(t_0)\right\}$
Gain	$K(t) = P(t)\,H^T(\hat{\mathbf{x}}(t), t)\,R^{-1}(t)$
Covariance	$\dot{P}(t) = F(\hat{\mathbf{x}}(t), t)\,P(t) + P(t)\,F^T(\hat{\mathbf{x}}(t), t)$ $-P(t)\,H^T(\hat{\mathbf{x}}(t), t)\,R^{-1}(t)H(\hat{\mathbf{x}}(t), t)\,P(t) + G(t)\,Q(t)\,G^T(t)$ $F(\hat{\mathbf{x}}(t), t) \equiv \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right\|_{\hat{\mathbf{x}}(t)}, \quad H(\hat{\mathbf{x}}(t), t) \equiv \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right\|_{\hat{\mathbf{x}}(t)}$
Estimate	$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t) + K(t)[\tilde{\mathbf{y}}(t) - \mathbf{h}(\hat{\mathbf{x}}(t), t)]$

As far as theoretically discussed, Extended Kalman Filter is a great estimator but there is a constraint. We are using linearized model of our system and as in linearized model, system matrices A,B,C and D are constant or fixed since we have linearized the system on the stable most point, while Extended Kalman Filter demands the continuously changing real time system model and real time LQR gain calculation which requires a lot of computational power. If we implement such a system on our microcontroller on real time, it will put a lot of computational load and hence cannot be implemented at this level.

Both Discrete Time Linear Kalman Filter as well as Extended Kalman Filter were discussed and studied thoroughly. Simulation has been done with some low level examples to get a know-how of these estimators but unfortunately due to shortage of time, we are not able to implement these estimators on our system.

Chapter 9

9. Conclusions

The goal of this project is the modelling and control of the ballbot. The main goal is to balance the ballbot and to track of the ballbot on a given set points.

To achieve this goal, we developed a controller for a highly agile and maneuverable ballbot. Advancing the system model, we proposed a full three-dimensional model and this model is verified. A model based, LQR (linear quadratic state feedback controller) is designed that controls the orientation of the body and the position of the ballbot simultaneously and shown to be able to stabilize the system for complex trajectories. Various experiments demonstrate the unique capabilities of the ballbot stabilized by the 3d controller. A nonlinear controller is also proposed in order to expand the performance of the ballbot prototype.

Due to the fact that the controller mostly reduces the errors in pitch and roll angles and only slightly reduces the errors in pitch and roll angular rates, so only small variations in pitch and roll angles can be controlled by the ballbot. However, to be able to control the position of the ballbot, such that the ballbot is able to move around ant to track the given set points, the BBR must be able to cope with large variations in pitch and roll angles. Therefore, due to noisy sensor data, position control is not possible as long as the magnitude of the gains for the pitch and roll angular rates is limited by noise in sensor data.

Based on the performance of the system with the developed model based LQR controller, it can be concluded that the developed model has proved to be a sufficient representation of reality for the design of a controller.

Finally, it can be concluded that the system is robust to large variations in model parameters with the developed controller. Based on the 3D model, the system will most likely remain stable when new features will be added to the robot in the future.

9.1 Problems Experienced

The difficulties experienced throughout the project include theory, hardware, software and programming related problems. The main theoretical difficulty encountered was associated with understanding the mathematics behind calculating the controller gains. This was necessary for implementing the stable system through the programming

Programming of the microcontroller proved to be a difficult challenge as well. Completing the hardware initialization, PID controller, Kalman filter and other sub-routines within the programming caused excessive time consumption for the project. This was further compounded as the real-time demonstration phase could not be adequately completed.

9.2 Recommendations and Future work

To improve the performance of the ballbot in future, several recommendations can be made.
First of all, to improve the performance of the system and to further reduce the noise of the sensor data, it is recommended to design and implement a more advanced and effective filter to filter the sensor data, for example a Kalman filter.
Secondly, to increase the grip between the omni-wheels and the ball, it is recommended to use brackets around the ball, so that slippage will be less and using brackets makes it possible to allow larger tilt angles for higher performance.
Thirdly, to improve the attenuation of system vibrations, it is recommended to prevent them from being measured by the gyroscope.
Furthermore, the implementation of position control is left as future work. After a successful implementation of position control, the controller, can also be implemented to increase the performance of the robot.

Bibliography

- L. Guzzella. Analysis and Synthesis of Single-Input Single-Output Control Systems. vdf Hochschulverlag AG, Z• urich, Switzerland, 2nd edition, 2009.
- L. Guzzella. Analysis and Synthesis of MIMO Control Systems. Lecture notes, ETH Zurich, Switzerland, 2010.
- T. Holzhuter. Zustandsregelung. Germany, 2010.
- J. Hussy. Signalprocessing for a ballbot. Bachelor thesis, ETH Z• urich, Switzerland, TBP.
- J. Fong, S. Uppill. Design and Build a Ballbot. Report, The University of Adelaide,
- Australia, 2009.
- S. Leutenegger. Unmanned Aircraft Design, Modeling and Control. Lecture notes, ETH Zurich, Switzerland, 2010.
- M. Kumagai, T. Ochiai. Development of a Robot Balancing on a Ball. Paper, Tohoku Gakuin University, Japan, 2008.
- N.G.M. Rademakers. Control of a Tailless Fighter using Gain-Scheduling. Traineeship report, Eindhoven University of Technology, Netherlands, 2004.
- S. Wigert S. Schuller. Regelung eines auf einem Ball balancierenden Roboters. Bachelor thesis, Zurich University of Applied Sciences, Switzerland, 2010.
- T. B. Lauwers, G. A. Kantor, R. L. Hollis. A Dynamically Stable Single-Wheeled Mobile Robot with Inverse Mouse-Ball Drive. Paper, Carnegie Mellon University, USA, 2006.
- Team Ballbot. Rezero {dynamisch stabil auf einer Kugel. Report, Swiss Federal Institute of Technology, Switzerland, 2010.
- U. Nagarajan, A. Mampetta, G. A. Kantor, R. L. Hollis. State Transition, Balancing, Station Keeping, and Yaw Control for a Dynamically Stable Single Spherical Wheel Mobile Robot. Paper, Carnegie Mellon University, USA, 2009.
- Eth Bachelor Thesis Modeling and Control of a Ballbot Spring Term 2010
- Modeling and Control of a Ball-Balancing Robot Internship & Master thesis at ALTEN Mechatronics July, 2014 by university of twenty

Odometry

Parameters

- **Radius of the ball**

```
rK;
```

- **Radius of the Omniwheel**

```
rW;
```

- **Radius of the body**

```
rA;
```

- **Distance between center of the ball an center of gravity of the body**

```
l;
```

- **Angle of the motors**

```
alpha = 45 °;
beta1 = 0 °;
beta2 = 120 °;
beta3 = 240 °;
```

- **Rotation speed vector of the ball in the Lisa reference frame L**

```
L_OmegaK[t] = {phix'[t], phiy'[t], phiz'[t]};
```

- **Rotation speed of the omniwheel about the motor axis in the body reference frame A (scalar, encoder value)**

```
A_omegaW1[t] = psi1'[t];
A_omegaW2[t] = psi2'[t];
A_omegaW3[t] = psi3'[t];
```

Coordinates and rotation matrices (I↔L↔A)

- **Rotation matrices I↔L↔A**

Rotations around z, y and x Axes

```
Rz = {{Cos[thetaz[t]], -Sin[thetaz[t]], 0},
   {Sin[thetaz[t]], Cos[thetaz[t]], 0}, {0, 0, 1}};
Ry = {{Cos[thetay[t]], 0, Sin[thetay[t]]}, {0, 1, 0},
   {-Sin[thetay[t]], 0, Cos[thetay[t]]}};
Rx = {{1, 0, 0}, {0, Cos[thetax[t]], -Sin[thetax[t]]},
   {0, Sin[thetax[t]], Cos[thetax[t]]}};
```

Rotation Matrix (I to L and L to I)

```
RIL = Rz;
RLI = Transpose[RIL];
```

Rotation Matrix (L to A and A to L)

```
RLA = Ry.Rx;
RAL = Transpose[RLA];
```

Rotation Matrix (I to A and A to I)

```
RIA = RIL.RLA;
RAI = Transpose[RIA];
```

- **Ball**

Rotation of the ball in I

```
OmegaK[t] = RIL.L_OmegaK[t];
```

- **Jacob**

Jacobian Matrix

```
J = {{1, 0, -Sin[thetay[t]]}, {0, Cos[thetax[t]], Sin[thetax[t]] * Cos[thetay[t]]},
   {0, -Sin[thetax[t]], Cos[thetax[t]] * Cos[thetay[t]]}};
```

Time variation of Tait-Bryan angles

```
ThetaDot[t] = {thetax'[t], thetay'[t], thetaz'[t]};
```

Rotation Vector of the Aufbau (in A)

```
A_OmegaA[t] = J.ThetaDot[t];
```

Rotation of the ball

Vector from center of the ball (P) to the contact point with the omniwheels (K1, K2, K3) in A

```
A_rPK1 = {rK * Sin[alpha] * Cos[beta1], rK * Sin[alpha] * Sin[beta1], rK * Cos[alpha]};
A_rPK2 = {rK * Sin[alpha] * Cos[beta2], rK * Sin[alpha] * Sin[beta2], rK * Cos[alpha]};
A_rPK3 = {rK * Sin[alpha] * Cos[beta3], rK * Sin[alpha] * Sin[beta3], rK * Cos[alpha]};
```

Direction of the tangential speed of the rotation of the omniwheels in A

```
A_d1 = {-Sin[beta1], Cos[beta1], 0};
A_d2 = {-Sin[beta2], Cos[beta2], 0};
A_d3 = {-Sin[beta3], Cos[beta3], 0};
```

Test : Absolute value has to be 1! (ok)

```
Norm[A_d1];
Norm[A_d2];
Norm[A_d3];
```

Rotation speed of the ball relative to the body in A

```
A_omegaK = {A_omegaKx, A_omegaKy, A_omegaKz};
```

Speed on the surface of the ball (in omniwheel direction) has to be the same speed as the tangetial speed of the omniwheel

```
E1 = Cross[A_omegaK, A_rPK1].A_d1 == A_omegaW1[t] * rW;
E2 = Cross[A_omegaK, A_rPK2].A_d2 == A_omegaW2[t] * rW;
E3 = Cross[A_omegaK, A_rPK3].A_d3 == A_omegaW3[t] * rW;

sol = Solve[{E1, E2, E3}, {A_omegaKx, A_omegaKy, A_omegaKz}];

A_omegaK = {A_omegaKx /. sol[[1]], A_omegaKy /. sol[[1]], A_omegaKz /. sol[[1]]};
```

Rotation speed of the ball in A

```
A_OmegaK = A_omegaK + A_OmegaA[t];
```

Coordinate transformation from A to I System

```
L_OmegaK[t] = FullSimplify[RLA.A_OmegaK];
```

Solution

```
phix'[t] = L_OmegaK[t][[1]]
```

$$\frac{1}{3\,rK}\left(\sqrt{6}\,rW\,Sin[thetax[t]]\,Sin[thetay[t]]\,(-psi2'[t]+psi3'[t])+\sqrt{2}\,rW\,Cos[thetax[t]]\,Sin[thetay[t]]\,(psi1'[t]+psi2'[t]+psi3'[t])+Cos[thetay[t]]\left(\sqrt{2}\,rW\,(-2\,psi1'[t]+psi2'[t]+psi3'[t])+3\,rK\,thetax'[t]\right)\right)$$

```
phiy'[t] = L_OmegaK[t][[2]]
```

$$\frac{1}{3\,rK}\left(\sqrt{6}\,rW\,Cos[thetax[t]]\,(-psi2'[t]+psi3'[t])-\sqrt{2}\,rW\,Sin[thetax[t]]\,(psi1'[t]+psi2'[t]+psi3'[t])+3\,rK\,thetay'[t]\right)$$

```
phiz'[t] = L_OmegaK[t][[3]]
```

$$\frac{1}{3\,rK}\left(\sqrt{2}\,rW\,(Cos[thetax[t]]\,Cos[thetay[t]]+2\,Sin[thetay[t]])\,psi1'[t]+\sqrt{2}\,rW\left(\sqrt{3}\,Cos[thetay[t]]\,Sin[thetax[t]]\,(-psi2'[t]+psi3'[t])+Cos[thetax[t]]\,Cos[thetay[t]]\,(psi2'[t]+psi3'[t])-Sin[thetay[t]]\,(psi2'[t]+psi3'[t])\right)+3\,rK\,(-Sin[thetay[t]]\,thetax'[t]+thetaz'[t])\right)$$

Translation of the ball

Vector from the ground to center of the ball (P)

```
rBP = {0, 0, rK};
```

Substitution

```
OmegaK_given[t] = RIL.{phix_given'[t], phiy_given'[t], phiz_given'[t]};
```

Speed vector of the center of the ball (P) in I

Lightweight solution with use of solutions above

```
rPdot_light = FullSimplify[Cross[OmegaK_given[t], rBP]]
```

```
{rK (Sin[thetaz[t]] phix_given'[t] + Cos[thetaz[t]] phiy_given'[t]),
 -rK Cos[thetaz[t]] phix_given'[t] + rK Sin[thetaz[t]] phiy_given'[t], 0}
```

Explicit solution

```
rPdot = FullSimplify[Cross[OmegaK[t], rBP]]
```

$$\left\{\frac{1}{3}\left(\sqrt{2}\ rW \sin[\theta ty[t]] \sin[\theta tz[t]] \left(\sqrt{3}\ \sin[\theta tx[t]] (-\psi 2'[t] + \psi 3'[t]) + \cos[\theta tx[t]] (\psi 1'[t] + \psi 2'[t] + \psi 3'[t])\right) + \cos[\theta ty[t]] \sin[\theta tz[t]] \left(\sqrt{2}\ rW (-2\psi 1'[t] + \psi 2'[t] + \psi 3'[t]) + 3\, rK\, \theta tx'[t]\right) + \cos[\theta tz[t]] \left(\sqrt{6}\ rW \cos[\theta tx[t]] (-\psi 2'[t] + \psi 3'[t]) - \sqrt{2}\ rW \sin[\theta tx[t]] (\psi 1'[t] + \psi 2'[t] + \psi 3'[t]) + 3\, rK\, \theta ty'[t]\right)\right),$$

$$\frac{1}{3}\left(\sqrt{2}\ rW \left((\cos[\theta tz[t]] (2\cos[\theta ty[t]] - \cos[\theta tx[t]] \sin[\theta ty[t]]) - \sin[\theta tx[t]] \sin[\theta tz[t]]) \psi 1'[t] - \left(\cos[\theta tz[t]] \left(\cos[\theta ty[t]] + \left(\cos[\theta tx[t]] - \sqrt{3}\ \sin[\theta tx[t]]\right) \sin[\theta ty[t]]\right) + \left(\sqrt{3}\ \cos[\theta tx[t]] + \sin[\theta tx[t]]\right) \sin[\theta tz[t]]\right) \psi 2'[t] - \left(\cos[\theta ty[t]] \cos[\theta tz[t]] + \cos[\theta tz[t]] \left(\cos[\theta tx[t]] + \sqrt{3}\ \sin[\theta tx[t]]\right) \sin[\theta ty[t]] + \left(-\sqrt{3}\ \cos[\theta tx[t]] + \sin[\theta tx[t]]\right) \sin[\theta tz[t]]\right) \psi 3'[t]\right) - 3\, rK \cos[\theta ty[t]] \cos[\theta tz[t]] \theta tx'[t] + 3\, rK \sin[\theta tz[t]] \theta ty'[t]\right), 0\right\}$$