Optimal state estimation for a power line inspection robot

Divij Soobhug

A thesis presented for the degree of
Master of Science in Engineering

Department of Electrical Engineering
University of Cape Town
South Africa
13/07/2016

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signed by candidate

Divij Soobhug

Abstract

Following a paper published by E. Boje[1], this thesis discusses the design and off-line testing of different types of Kalman filters to estimate the attitude, position and velocity of a robotic platform moving along a power line. The nature of this problem limits the use of magnetometers. Magnetic field interference from the steel pylons and steel cored conductors will affect the local magnetic field. Moreover, high frequency signals from on-board power electronic drives and induced magnetic fields due to ferromagnetic components of the robot along with aliasing, quantization effects and a low signal to noise ratio make notch filtering at 50 Hz impractical. Thus, a GPS/IMU filter solution, which uses the power line curvature and horizontal direction in measurements, to constrain the robot to the line was designed. Different types of filters were implemented; The Extended Kalman filter (EKF), the Unscented Kalman filter (UKF) and the Error State Kalman filter (ErKF). Measurements were recorded and the filters were tested offline.

While all the filters tracked properly, it was found that the EKF was better in computational speed completing an iteration in 87 μs, the ErKF was second best with an average time of 120 μs for one iteration and the UKF was last with an average time of 1040 μs for one iteration. Errors between the true state and estimated state for the simulation were quantified using root mean square values (RMS). The RMS values were almost the same for the EKF and ErKF with the error for the x position at 0.81 m and z position at 0.038 m. The UKF produced RMS errors of 0.79 m for x position and 0.11 m for z position. It can be seen that the UKF is slightly better for the x position but is much worse for the z position. Overall, the GPS measurement RMS values used were 4 m and 20 m for the horizontal and vertical positions respectively. Thus, the filters brought a big improvement. However, the recommended filter is the EKF as is produced comparable or better results as compared to other filters and expends the least computational effort.

A state estimator was also developed for a J.Patel's PLIR project [2], where a brachiating version of a power line robot was modeled. The brachiation mechanism was approximated to a double pendulum and kinematics based Kalman filter was designed. Simulations of EKF and UKF were made. The EKF is still recommended as its estimates are closer to the true values and its computation time is about five times faster.

Acknowledgements

I would first like to thank my supervisor Professor Edward Boje for his guidance throughout the whole project.

Next, thanks goes to Arnold Pretorius, a PhD student at UCT, who gave useful insight during the project. I would also like to thank Javaad Patel, a UCT MSc student, who helped with the brachiating robot.

Finally, I would like to thank all those who contributed towards my moral support, especially my parents, my fiancee Chaya, and spiritual master Sri Sri Ravi Shankar.

Contents

List of Figures

List of Tables

Nomenclature

List Of Abbreviations

$3D$ Three Dimensional

CEP Circular Error Probability

DCM Direction Cosine Matrix

DOP Dilution Of Precision

$DRMS$ Distance Root Mean Square

EKF Extended Kalman Filter

$ErKF$ Error State Kalman Filter

$ESKOM$ Electricity Supply Commission

GPS Global Positioning System

$HDOP$ Horizontal Dilution Of Precision

IMU Inertial Measurement Unit

JPL Jet Propulsion Laboratory

LSE Least Squares Estimation

$PDOP$ Positional Dilution Of Precision

$UERE$ User Range Equivalent Error

UKF Unscented Kalman Filter

$UKZN$ University Of KwaZulu-Natal

$VDOP$ Vertical Dilution Of Precision

List Of Symbols

α_u	Scaling factor used in UKF.
$\bar{p}$	Four element quaternion vector.
$\bar{q}$	Four element quaternion vector.
β_u	Value that gives knowledge of the type of noise distribution in the UKF.
$\mathbf{0}$	Zero matrix.
$\boldsymbol{\chi}$	Vector with sigma points.
$\boldsymbol{\delta\hat{x}}$	Estimated error state vector.
$\boldsymbol{\delta b_\omega}$	Error state vector of angular velocity bias.
$\boldsymbol{\delta b_a}$	Error state vector of linear acceleration bias.
$\boldsymbol{\delta q}$	Quaternion vector error state.
$\boldsymbol{\delta s}$	Position error state vector.
$\boldsymbol{\delta v}$	Velocity error state vector.
$\boldsymbol{\delta x}$	Error state vector.
$\boldsymbol{\eta_\omega}$	Vector representing white noise in the angular velocity measurements.
$\boldsymbol{\eta_a}$	Vector representing white noise in the linear acceleration measurements.
$\boldsymbol{\eta_{b\omega}}$	Vector representing bias noise of velocity measurements.
$\boldsymbol{\eta_{ba}}$	Vector representing bias noise of acceleration measurements.
$\boldsymbol{\eta}$	Constant representing weighs assigned to sigma points in the UKF.
$\boldsymbol{\hat{e}}$	Unit vector.

$\boldsymbol{\Lambda}(\bar{q})$ Matrix used to convert post multiplication by a quaternion vector to premultiplication by a quaternion.

$\boldsymbol{\omega_m}$ Measured Angular velocity vector.

$\boldsymbol{\omega}$ Angular velocity vector (True angular velocity).

$\boldsymbol{\phi}$ Discrete state transition matrix.

$\boldsymbol{\Psi}$ Measurement vector obtained by applying sigma points in the measurement equation.

$\boldsymbol{a_m}$ Measured linear acceleration vector.

$\boldsymbol{A}$ State matrix of the state-space equation.

$\boldsymbol{a}$ Acceleration vector.

$\boldsymbol{b_\omega}$ Bias vector of the angular velocity measurements.

$\boldsymbol{b_a}$ Bias vector of the linear acceleration measurements.

$\boldsymbol{B}$ Input matrix of the state-space equation.

$\boldsymbol{C_B}$ Matrix representing the Coriolis effects acting on the brachiating robot.

$\boldsymbol{C}$ Output matrix of the state-space equation.

$\boldsymbol{D}$ Constraint matrix.

$\boldsymbol{d}$ Constraint measurement vector.

$\boldsymbol{e}$ Error vector.

$\boldsymbol{F_B}$ Matrix representing the frictional forces acting on the brachiating robot

$\boldsymbol{G_B}$ Matrix representing the gravitational forces acting on the brachiating robot.

$\boldsymbol{g}$ Gravity vector.

$\boldsymbol{I}$	Identity matrix.
$\boldsymbol{i}$	Imaginary Number.
$\boldsymbol{j}$	Imaginary Number.
$\boldsymbol{K}$	Kalman gain matrix.
$\boldsymbol{k}$	Imaginary Number.
$\boldsymbol{M_B}$	Matrix representing the inertia of the brachiating robot.
$\boldsymbol{o'}$	Vector produced after vector $\boldsymbol{o}$ has been rotated.
$\boldsymbol{o}$	Vector which undergoes quaternion rotation.
$\boldsymbol{P}$	Error covariance matrix.
$\boldsymbol{p}$	Vector containing the first three elements of the quaternion $\bar{p}$.
$\boldsymbol{P}_0$	Initial value of the error covariance matrix.
$\boldsymbol{Q_c}$	Process noise covariance matrix for a continuous state-space system
$\boldsymbol{Q_d}$	Process noise covariance matrix for a discrete state-space system
$\boldsymbol{q_g}$	Generalized Lagrangian coordinates.
$\boldsymbol{Q}$	Process noise covariance matrix.
$\boldsymbol{q}$	Vector containing the first three elements of the quaternion $\bar{q}$.
$\boldsymbol{Q}_{gps}$	Noise covariance of GPS position measurements.
$\boldsymbol{R}$	Measurement noise covariance matrix.
$\boldsymbol{R}(\bar{q})$	Rotation matrix.
$\boldsymbol{s}$	Vector representing linear displacement of the robot.
$\boldsymbol{u}(t)$	Input to state equation.
$\boldsymbol{v}$	Vector representing linear velocity of the robot.

$\boldsymbol{v}(t)$	Measurement noise.
$\boldsymbol{w}(t)$	Process noise.
$\boldsymbol{x}$	State vector containing the true values of the states.
$\boldsymbol{x}_0$	Initial value of the state vector.
$\boldsymbol{y}$	Output vector formed from measurements.
$\boldsymbol{z}$	Measurement vector.
$\delta\bar{q}$	Quaternion error state vector
$\hat{\bar{q}}$	Estimated quaternion.
$\hat{\boldsymbol{x}}$	Vector representing the estimated states.
$\hat{\boldsymbol{x}}_0$	Initial value of the state vector estimate.
$\hat{\boldsymbol{y}}$	Estimated measurement vector obtained from state estimates.
κ_u	Scaling factor used in UKF.
Ł	Dimension of state vector $x\boldsymbol{x}$.
λ	Scaling factor used in UKF.
λ_L	Latitude.
ω_x	Angular velocity about the x axis.
ω_y	Angular velocity about the y axis.
ω_z	Angular velocity about the z axis.
ϕ	Yaw angle of the brachiating robot.
ϕ_L	Longitude.
ψ	Yaw angle of the brachiating robot.
σ_{uere}	GPS user equivalent range error.

σ_x	GPS positional error in the east direction.
σ_y	GPS positional error in the north direction.
σ_z	GPS positional error vertically up.
θ	Pitch angle of the brachiating robot.
θ_1	Angle that the first arm of the brachiating robot makes with the vertical axis
θ_2	Angle between the first and second arms of the brachiating robot.
φ	Angle denoting the amount by which a unit vector $\hat{\boldsymbol{e}}$ is rotated.
ζ	Catenary constant of the power line.
a_x	Acceleration of the robot along the reference x axis.
a_y	Acceleration of the robot along the reference y axis.
a_z	Acceleration of the robot along the reference z axis.
H	Horizontal Tension acting on power line.
L	Span of the power line.
p_1	First element of the quaternion $\bar{p}$.
p_2	Second element of the quaternion $\bar{p}$.
p_3	Third element of the quaternion $\bar{p}$.
p_4	Fourth element of the quaternion $\bar{p}$.
q_1	First element of the quaternion $\bar{q}$.
q_2	Second element of the quaternion $\bar{q}$.
q_3	Third element of the quaternion $\bar{q}$.
q_4	Fourth element of the quaternion $\bar{q}$.

S	Sag of the power line.
T	Sampling time of the system.
v	Velocity along the power line.
v_x	Velocity along the reference x axis.
v_y	Velocity along the reference y axis.
v_z	Velocity along the reference z axis.
w	Weight per unit length of power line.
x	Position along the reference x axis.
y	Position along the reference y axis.
z	Position along the reference z axis.

Chapter 1

Introduction

Well maintained power lines are one of the cornerstones to provide sustainable electricity supply to customers; hence regular power line inspections need to be conducted by the electricity providers. These inspections are currently being performed manually at ESKOM. However this method is both costly and time consuming. Thus, as a more cost effective and time efficient solution, the use of power line inspection robots was proposed [10].

This dissertation's focus is on the state estimation strategy required by a robotic platform to manoeuvre along power lines. A GPS and IMU Kalman filter solution will be used to track the position and attitude of the robot in motion. Unfortunately, due to the interference created by the power line magnetic field, the use of a magnetometer for this application is not suitable. Magnetic field interference from the steel pylons and steel cored conductors will affect the local magnetic field. Additionally high frequency electromagnetic signals from on-board power electronic drives and induced magnetic fields due to ferromagnetic components of the robot along with aliasing, quantization effects and a low signal to noise ratio making notch filtering at 50 Hz impractical [1]. Thus, extra information such as the sag parameter of the power line and the power line direction will be used to compensate for the above [1].

Depending on the type of robot used, different system models may be needed. For instance, the brachiating robot [2], consists of a rolling motion (when moving along the line) and a swinging motion (when swinging between line segments and past obstacles such as spacers, dampers and suspension clamps). Hence, different strategies will be required to track the robot's joint angles during each type of motion.

1.1 Literature Review

1.1.1 Optimal Estimation

Developed by Gauss in the early 1800s, least squares estimation (LSE) is one of the earliest forms of optimal approximation [11]. It is based on the minimization of mean squared errors that many modern estimation techniques use.

In the LSE, a measurement vector, $\boldsymbol{z}$ ($\in R^n$) with error $\boldsymbol{e}$ ($\in R^n$) is expressed as a linear combination of states $\boldsymbol{x}$ ($\in R^m$) and a matrix $\boldsymbol{M}$ ($\in$ of $R^{n\times m}$), equation (1.1). The error square ($\boldsymbol{ee^T}$) is then minimized by equating its first derivative, with respect to $\boldsymbol{x}$, to zero. The error can also be weighted (using matrix $\boldsymbol{Q}$, which is positive definite and usually contains inverse variance values of the error) to provide an unbiased estimate. This results in equation (1.2) [11].

$$\boldsymbol{Mx} = \boldsymbol{z} + \boldsymbol{e} \tag{1.1}$$

$$\boldsymbol{x} = (\boldsymbol{M^T Q M})^{-1}\boldsymbol{M^T Q z} \tag{1.2}$$

As shown in [11], equation (1.2) can be transformed into a recursive algorithm (recursive linear least squares filter) as measurements are obtained at every point in time.

In the 1940's Wiener proposed a weighting function approach to minimize the mean square error, and this is considered to be the starting point of modern optimal filtering theory. While Weiner's filter is a steady-state version of the Kalman-Bucy filter for stationary, linear continuous time processes. Kalman developed a filter which considered noise as a discrete process and was posed in the form of a state-space equation [8].

The Kalman filter is the best unbiased linear estimator that reduces the errors introduced in the states of a system, under a set of conditions (i.e. all errors follow a Gaussian distribution). The Kalman filter is an elegant solution to the state estimation problem, due to the fact that it is mathematically well formulated and makes use of the well understood state-space model. This simplifies the treatment of noisy signals when present in the system.

Algorithms, such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) have been developed to deal with nonlinear systems. These are used in many tracking applications involving GPS/IMU fusion, especially in car and marine vessel navigation systems. The high frequency IMU measurements are used in a dead reckoning approach when GPS signals are not available. States are corrected as soon as the low frequency GPS signal is obtained. Such a filter is also called the multi-rate Kalman filter. Kalman filtering is discussed in greater detail in Chapter 2.

A similar similar scheme as in GPS/IMU solutions used in land vehicle navigation can be used [3]. This is shown in figure 1.1.

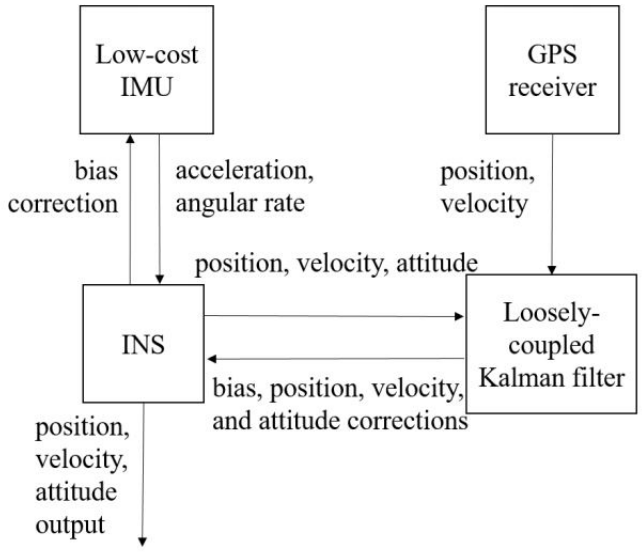

Figure 1.1: GPS/IMU filtering scheme [3].

The GPS, accelerometer and gyroscope measurements are fused using a loosely coupled Kalman filter. The IMU measurements are used to form the state matrix and the GPS measurements are used to form the output matrix. Also, biases are estimated and used to correct the IMU measurements. The above approach can be used and restrictions applied to the robot to lie on the power line, similar to restrictions applied to land vehicles to stay on a predefined track [3].

1.1.2 Attitude Representations

A convenient method of attitude representation may decrease the complexity of the system model equations. Traditionally, Euler angles have been used for attitude representation, but have been superseded in various applications by quaternions.

The attitude of a system is characterized by 3 angles; namely roll, pitch and yaw. Converting from the inertial frame to body frame requires the rotation of the x, y and z axes by the roll, pitch and yaw angles, respectively, as shown in figure 1.2. Thus, the sequence of pitch, roll and yaw rotation characterize Euler angles.

Other attitude representations such as the Rodrigues parameters, the Cayley Klein parameters or modified parameters, have been developed, using Euler angles as a base, to suit various applications [12].

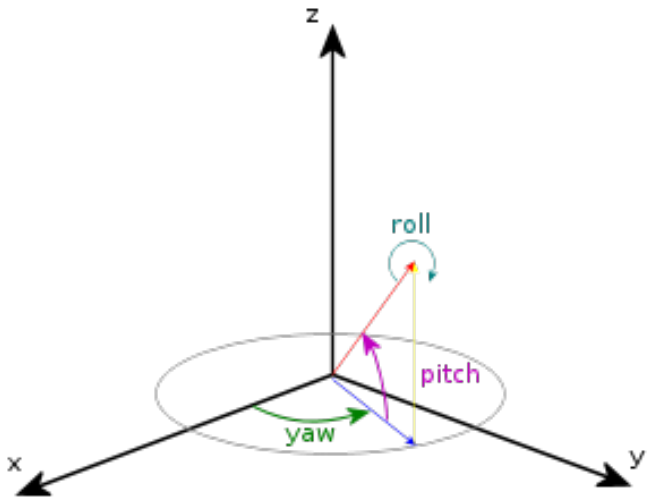

Figure 1.2: Roll Pitch and Yaw angles [4]

.

While being easy to use and visualize, Euler angles come with a major disadvantage. Singularities can be encountered, especially when the body makes full 3D rotations. This effect is also commonly known as gimbal-lock. In the case of a brachiating robot, it is something that must be carefully considered as two full rotations are required when swinging to another line segment.

Therefore, quaternions can be used as an alternative in the estimation algorithm.

The quaternion, first proposed by Sir William Rowan Hamilton, is a four element vector representing 3D attitude. It is convenient to group the quaternion $\bar{q}$ into a 3 element vector $\boldsymbol{q}$ and a scalar part q_4. There are two notations of the quaternion commonly used, namely, the JPL (Jet Propulsion Laboratory) and the Hamiltonian convention. The JPL convention uses the scalar part of the quaternion as its fourth element while the Hamiltonian convention uses it as the first element [13].

$$\bar{q} = \begin{bmatrix} \boldsymbol{q} \\ q_4 \end{bmatrix} \text{ JPL convention} \qquad \bar{q} = \begin{bmatrix} q_4 \\ \boldsymbol{q} \end{bmatrix} \text{ Hamiltonian convention} \tag{1.3}$$

Throughout this dissertation the JPL notation will be used to represent quaternions. Hence, a quaternion rotation can be described by equation (1.4), where the Kronecker symbol denotes quaternion multiplication and $\boldsymbol{o}$ is a vector. Also, the inverse of a quaternion is simply its conjugate (i.e. changing the sign of its vector part) [14].

$$\boldsymbol{o}' = \bar{q} \otimes \boldsymbol{o} \otimes \bar{q}^{-1} \tag{1.4}$$

The quaternion can also be defined using imaginary numbers $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ (equation (1.5)).

$$\bar{q} = q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} + q_4 \tag{1.5}$$

These imaginary numbers satisfy equation (1.6).

$$\begin{aligned} \boldsymbol{i}^2 = -1 \quad \boldsymbol{j}^2 = -1 \quad \boldsymbol{k}^2 = -1 \\ -\boldsymbol{ij} = \boldsymbol{ji} = \boldsymbol{k} \quad -\boldsymbol{jk} = \boldsymbol{kj} = \boldsymbol{i} \quad -\boldsymbol{ki} = \boldsymbol{ik} = \boldsymbol{j} \end{aligned} \tag{1.6}$$

Multiplying two quaternions requires one to perform the operation in equation (1.7).

$$\bar{q} \otimes \bar{p} = (q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} + q_4)(p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k} + p_4) \tag{1.7}$$

The above can be simplified, and can be written in vector form in equation (1.8) [14]. It is seen that the quaternion multiplication operation is different from matrix multiplication.

$$\bar{q} \otimes \bar{p} = \begin{bmatrix} q_4\boldsymbol{p} + p_4\boldsymbol{q} - \boldsymbol{p} \times \boldsymbol{p} \\ q_4 p_4 - \boldsymbol{q}^T\boldsymbol{p} \end{bmatrix} \tag{1.8}$$

Quaternion rotations can be visualized as rotating the reference axes about a vector rather than rotating each axis individually by a pitch roll and yaw angle as shown in figure 1.3. Equation (1.9) shows the quaternion required to rotate the axes by an angle φ about the normal vector $\hat{\boldsymbol{e}}$. Therefore, the vector component of the quaternion contains information on the unit vector used to rotate the axes and the scalar part contains the angle of rotation.

$$\bar{q} = \begin{bmatrix} \boldsymbol{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{e}} \sin(\varphi/2) \\ \cos(\varphi/2) \end{bmatrix} \tag{1.9}$$

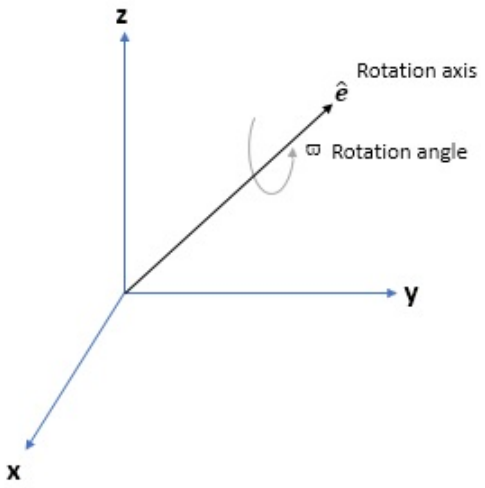

Figure 1.3: Quaternion rotation

Quaternions can be converted to Euler angles and vice versa by using Euler's direction cosine matrix (DCM) formula (see Appendix A). The DCM is the matrix used to rotate a vector from one frame to the other.

Throughout this project, the inertial frame will be assumed to be an Earth fixed frame which is stationary during the robot's motion. In the inertial frame, the y axis is assumed to lie along the true north, x axis along the east and z axis vertically up. Also, a convenient reference frame will be defined so that some equations can be simplified. In this reference frame, the x axis lies in the direction of the power line, the z axis is vertically up and y axis perpendicular to the line, with the origin starting at the first pylon position, figure 1.4. (i.e The reference frame is the inertial frame rotated about the z axis).

Figure 1.4: Reference frame with x axis along the line and z axis up [1].

1.1.3 Global Positioning System

The GPS has been developed by the US military in the 1970's and can provide a user with the current position and speed [15]. This system is now available to civilians and is used in a myriad of applications, although there are restrictions imposed on public use (such as a 18 km altitude and 515 m/s cap)[15]. The GPS consists of three segments: the space segment, the control segment and the user segment. The space segment further consists of 32 GPS satellites orbiting the Earth to ensure that there are enough satellites in view at any point on the Earth's surface. The control segment refers to the worldwide network of GPS tracking stations and the user segment would be the GPS receiver on the user end [15].

The data received from each satellite is called the almanac which contains

information such as the satellite's position, velocity and time. From this information, the distance between the satellite and receiver can be computed. Usually, a minimum of four satellites are used to get the receiver's position by a trilateration process. This process works by using the intersection of satellite loci to produce a point or two points where the receiver must lie as in figure 1.5 (i.e. the intersection of at least three spheres is needed to obtain the point/points where the receiver could be at). Finally, a fourth satellite is required to synchronize time and obtain the final GPS receiver position [16][15].

Figure 1.5: GPS trilateration [5].

The set of measurements x, y, z and t which are obtained from satellites are called pseudorange measurements. The uncertainty in the pseudorange measurements can be due to atmospheric effects, multipath effects or ephemeris, and clock bias [16]. As the GPS signal passes through the atmosphere it gets refracted, therefore causing delays. Multipath errors occur when signals are reflected over surfaces before reaching the receiver while ephemeris errors are caused by imperfections in the satellite dynamical model.

GPS accuracy is determined by the position of the satellites and the number of satellites in view. This accuracy is usually given in the form of DOP (Dilution of Precision) values. With satellites further apart, higher accuracies are obtained [16]. Similarly, with more satellites, a greater resolution is obtained. Horizontal dilution of precision (HDOP) and positional dilution of presicion (PDOP) are described in [16] by equation (1.10) and equation (1.11), where σ_{uere} is the user equivalent range error, and σ_x, σ_y and σ_z are

the errors in east, north and up directions respectively.

$$HDOP = \frac{\sqrt{\sigma_x^2 + \sigma_y^2}}{\sigma_{uere}} \tag{1.10}$$

$$PDOP = \frac{\sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}}{\sigma_{uere}} = \sqrt{HDOP^2 + \frac{\sigma_z^2}{\sigma_{uere}^2}} \tag{1.11}$$

To further increase the reliability of the GPS signal, differential GPS can be used. The differential GPS uses another receiver which can be a ground based station to compensate for the delay created by the Earth's atmosphere.

When using the GPS along with other sensors such as IMUs and magnetometers, either a tight coupling or a loose coupling approach can be used to fuse them. The difference between the two is that the tightly coupled approach uses the pseudorange data directly, while the loosely coupled system uses the position and velocity readings obtained from the receiver as measurements [17]. The advantage of tight coupling is that individual satellite measurements can still contribute to the Kalman filter, in case there are not enough satellites to produce position and velocity estimates. Also, the state vector will need to be augmented to accomodate for the clock states. Thus, despite being less accurate, the loosely coupled systems are often preferred due to their simplicity [8].

1.1.4 Power Line Inspection Robot

A power line inspection robot, designed at UKZN, has been successfully tested in the field and is shown in figure 1.6 [18]. The robot rolls along the power line by means of wheels, and consists of a platform with all the on-board electronics.

The brachiating robot (figure 1.7) is a prototype that has been developed at the University of Cape Town. It is an under actuated system as it requires less actuator torque to get around obstacles because of swinging. The dynamics of brachiation, required for modelling, has been described in [2].

The brachiating robot can be modeled as a double pendulum acting in a vertical plane through the power line. Using Lagrangian dynamics, an expression for the torque (equation (1.12)) was derived, [2] where $\boldsymbol{M_B}$,$\boldsymbol{C_B}$,$\boldsymbol{G_B}$,$\boldsymbol{F_B}$

Figure 1.6: UKZN's power line robot [6].

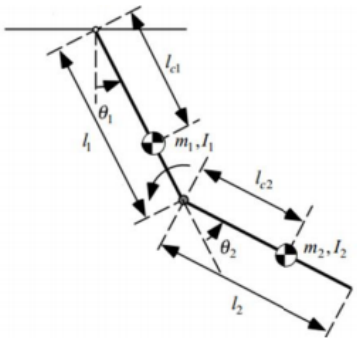

Figure 1.7: The brachiating robot [2].

and $\boldsymbol{\tau}$ represent the Inertia, Coriolis effects, gravity, friction and torque. The robot was assumed to be a double pendulum rotating in a plane and whose generalized coordinates q_g was chosen to be a vector formed from angles θ_1 and θ_2.

$$\boldsymbol{M_B(q_g)\ddot{q}_g + C_B(q_g, \dot{q}_g)\dot{q}_g + G_B(q_g) + F_B = B\tau} \quad \text{where} \quad \boldsymbol{q_g} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{1.12}$$

1.2 Summary

This chapter introduces the problem, which essentially entails estimating the position, velocity and attitude of a power line inspection robot. A literature review follows, which includes some of the information needed to produce a solution to the problem. Four main sections are covered, which are as follows:

- An introduction to optimal state estimation, which describes the background of the subject. This includes the least square estimation and Kalman filtering techniques.
- Spatial attitude representations in terms of Euler angles and quaternions.
- The basics of the global positioning systems (GPS).
- The power line inspection robot, where two types of robots are described.

In the next chapter, the different types of Kalman filters, which form an essential part of this project, will be described in more detail.

Chapter 2

The Kalman Filter

2.1 Linear Kalman Filter

Kalman filtering is fundamental to modern optimal state estimation. The Kalman filter is a linear unbiased least squares optimal estimator of the state vector of a linear dynamic system. Properly designing one, requires a well defined state-space system model, a measurement model, and knowledge of the magnitude of noise signals affecting the state variables and measurements.

The Kalman filter assumes that all state variables are random with a mean value and an uncertainty given by its variance. The state variables can be correlated or uncorrelated. In the case of uncorrelated state variables, a state variable gives no information about other state variables and is therefore not very useful. However, in the case of correlated state variables, one state can be used to predict the value of other states. For example, velocity is used in estimating position. Correlation in Kalman filtering is captured by the error covariance matrix $\boldsymbol{P}$, where the off diagonal elements give the amount of correlation between states and the diagonal elements give the variance of the states. The error covariance matrix is expressed in equation (2.1) and is a measure of the error between the estimated states and the real states, where $\boldsymbol{x}$ is the true value and $\hat{\boldsymbol{x}}$ is the estimated value.

$$\boldsymbol{P} = E < (\boldsymbol{x} - \hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}})^T > \text{ where } E < . > \text{ is the expectation operation.} \tag{2.1}$$

Once a system has been modeled (equation (2.2), where $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ represent uncorrelated noise terms), it is possible to use the previous estimated states to predict what the next states should be. This is achieved using equation (2.6). The new covariance due to state propagation is given by the term $\boldsymbol{APA}^T$, of equation (2.38) where $\boldsymbol{A}$ is the state matrix. However,

there might be other external factors (uncertainties), acting on the system that need to be taken into account during this prediction step. This is modeled by using a process noise covariance matrix $\boldsymbol{Q}$ (equation (2.3)). The full error covariance is then propagated using equation (2.38) where both of the above covariances are added.

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{w}_t \\ \boldsymbol{y}_t &= \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}_t \end{aligned} \tag{2.2}$$

$$\boldsymbol{Q} = E < \boldsymbol{w}_k \boldsymbol{w}_k^T > \quad \text{and} \quad \boldsymbol{R} = E < \boldsymbol{v}_k \boldsymbol{v}_k^T > \tag{2.3}$$

The state estimates can then be refined using measurements. Measurements are related to the states by a measurement matrix $\boldsymbol{C}$. Also, the uncertainties in these measurements are captured by the measurement covariance matrix $\boldsymbol{R}$ (equation (2.3)). An estimate of the measurement can be found by applying the state estimates to the measurement equation ($\boldsymbol{C}\hat{\boldsymbol{x}}$) which will have a covariance given by $\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^T$. The above measurement estimate and actual measurement can be blended, using a blending factor, such that the mean square estimation error is minimized. Minimizing the state error covariance $\boldsymbol{P}$, results in the Ricatti difference equation (equation (2.4)) [11]. The update and propagation steps for the error covariance in table 2.1 result in the Ricatti equation when combined. The optimal blending factor, or Kalman gain, for which the Ricatti equation is minimized is given by equation (2.8). The states are then corrected using the Kalman gain and measurement residual (equation (2.9)).

$$\boldsymbol{P}_{t+1} = \boldsymbol{A}_t \boldsymbol{P}_t \boldsymbol{A}_t^T - \boldsymbol{A}_t \boldsymbol{P}_t \boldsymbol{C}_t^T (\boldsymbol{C}_t \boldsymbol{P}_t \boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1} \boldsymbol{C}_t \boldsymbol{P}_t \boldsymbol{A}_t^T + \boldsymbol{G}_t \boldsymbol{Q}_t \boldsymbol{G}_t^T \tag{2.4}$$

In Kalman filtering, it is assumed that all noises are white with zero mean. If in addition noises are Gaussian, the Kalman filter is the best filter even without the restriction of requiring a linear filter.

During the initialization step, the initial error state covariance matrix P_0, must be chosen to be equal to $E < (\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)^T >$. When an accurate value of the initial state $\hat{\boldsymbol{x}}_0$ is known, the value of $\boldsymbol{P}_0$ needs to be

Linear Kalman Filter		
Initialization	$\hat{\boldsymbol{x}}_0 \quad \text{and} \quad \boldsymbol{P}_0$	(2.5)
Propagation	$\hat{\boldsymbol{x}}_{t+1\|t} = \boldsymbol{A}_t\hat{\boldsymbol{x}}_{t\|t} + \boldsymbol{B}_t\boldsymbol{u}_t$	(2.6)
	$\boldsymbol{P}_{t+1\|t} = \boldsymbol{A}_t\boldsymbol{P}_{t\|t}\boldsymbol{A}_t^T + \boldsymbol{G}_t\boldsymbol{Q}\boldsymbol{G}_t^T$	(2.7)
Update	$\boldsymbol{K}_{t+1} = \boldsymbol{P}_{t+1\|t}\boldsymbol{C}_t^T(\boldsymbol{C}_t\boldsymbol{P}_{t+1\|t}\boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1}$	(2.8)
	$\hat{\boldsymbol{x}}_{t+1\|t+1} = \hat{\boldsymbol{x}}_{t+1\|t} + \boldsymbol{K}_{t+1}(\boldsymbol{y}_{t+1} - \boldsymbol{C}\hat{\boldsymbol{x}}_{t+1\|t})$	(2.9)
	$\boldsymbol{P}_{t+1\|t+1} = (\boldsymbol{I} - \boldsymbol{K}_{t+1}\boldsymbol{C}_t)\boldsymbol{P}_{t+1\|t}$	(2.10)

Table 2.1: Linear Kalman Filter algorithm [8]

very small. On the other hand, if $\hat{\boldsymbol{x}}_0$ is uncertain, then $\boldsymbol{P}_0$ must be high. If the initial state was known with high accuracy but the initial covariance is chosen to be too high, the filter will take more time than necessary to converge. To sum it up, $\boldsymbol{P}_0$ must be chosen to reflect the accuracy with which $\hat{\boldsymbol{x}}_0$ is known.

The Kalman filter is an optimal estimator in linear cases. However, most real world systems are nonlinear. Thus, different algorithms, such as the EKF (Extended Kalman Filter) and UKF (Unscented Kalman Filter), have been devised to deal with nonlinearities. The EKF uses local derivatives (Jacobians) while the UKF uses sigma points and the unscented transform to propagate the system variances [8].

2.2 The Extended Kalman Filter

The extended Kalman filter linearizes the system around the current estimates by using partial derivatives (or Jacobians). Assuming additive noise, the state-space model is given in equation (2.11).

$$\begin{aligned} \boldsymbol{x}_{t+1} &= f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w}_t \\ \boldsymbol{y}_t &= h(\boldsymbol{x}_t) + \boldsymbol{v}_t \end{aligned} \tag{2.11}$$

The variables of a discrete nonlinear differential equation are separated into a nominal value and an error (equation (2.12)) where the latest state estimate is used as the nominal value. The differential equation can then be re-written using a Taylor series expansion (equation (2.13))

$$\boldsymbol{x}_t = \hat{\boldsymbol{x}}_t + \boldsymbol{\delta x}_t \tag{2.12}$$

$$\begin{aligned} f(\boldsymbol{x}_t, \boldsymbol{u}_t) &\approx f(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_t) + \frac{df}{dx}_{|\hat{\boldsymbol{x}}} \boldsymbol{\delta x} + \frac{1}{2!}\frac{d^2 f}{dx^2}_{|\hat{\boldsymbol{x}}} \boldsymbol{\delta x}^2 + ... \\ h(\boldsymbol{x}_t) &\approx h(\hat{\boldsymbol{x}}_t) + \frac{dh}{dx}_{|\hat{\boldsymbol{x}}} \boldsymbol{\delta x} + \frac{1}{2!}\frac{d^2 h}{dx^2}_{|\hat{\boldsymbol{x}}} \boldsymbol{\delta x}^2 + ... \end{aligned} \tag{2.13}$$

By substituting equation (2.12) and equation (2.13) into equation (2.11) and rearranging, a linearized differential equation in terms of error states is obtained. Equation (2.14) is linearized up to first order terms assuming that δx and/or the higher order terms are small enough to form a smooth function, $f(\boldsymbol{x})$.[8]

$$\begin{aligned} \delta\hat{\boldsymbol{x}}_{t+1} &= \left.\frac{df}{dx}\right|_{\hat{\boldsymbol{x}}} \boldsymbol{\delta}\hat{\boldsymbol{x}}_t + \boldsymbol{w}_t \\ \boldsymbol{y}_t - h(\hat{\boldsymbol{x}}_t) &= \left.\frac{dh}{dx}\right|_{\hat{\boldsymbol{x}}} \boldsymbol{\delta}\hat{\boldsymbol{x}}_t + \boldsymbol{v}_t \end{aligned} \tag{2.14}$$

The above dynamics and measurement equations are linearized and can be applied in the Kalman filter algorithm. However, it is to be noted that the estimated states are incremental quantities or error states. These need to be added to the nominal state, according to equation (2.15), to obtain the total estimate.

$$\hat{\boldsymbol{x}}_{t+1|t+1} = \hat{\boldsymbol{x}}_{t+1|t} + \boldsymbol{\delta}\hat{\boldsymbol{x}}_{t+1|t+1} \tag{2.15}$$

Matrices A and C of the Kalman filter algorithm given by table 2.2 are formed from Jacobian matrices as shown in equation (2.22).

$$\boldsymbol{A} = \left.\frac{df}{dx}\right|_{\hat{\boldsymbol{x}}} \quad \text{and} \quad \boldsymbol{C} = \left.\frac{dh}{dx}\right|_{\hat{\boldsymbol{x}}} \tag{2.22}$$

Such a filter is sometimes referred to as the indirect Kalman filter or the error state Kalman filter [8].

Extended Kalman Filter		
Initialization	$\hat{\boldsymbol{x}}_0 \quad \text{and} \quad \boldsymbol{P}_0$	(2.16)
Propagation	$\hat{\boldsymbol{x}}_{t+1\|t} = f(\boldsymbol{x_t}, \boldsymbol{u_t})$	(2.17)
	$\boldsymbol{P}_{t+1\|t} = \boldsymbol{A}_t \boldsymbol{P}_{t\|t} \boldsymbol{A}_t^T + \boldsymbol{G}_t \boldsymbol{Q} \boldsymbol{G}_t^T$	(2.18)
Update	$\boldsymbol{K}_{t+1} = \boldsymbol{P}_{t+1\|t} \boldsymbol{C}_t^T (\boldsymbol{C}_t \boldsymbol{P}_{t+1\|t} \boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1}$	(2.19)
	$\hat{\boldsymbol{x}}_{t+1\|t+1} = \hat{\boldsymbol{x}}_{t+1\|t} + \boldsymbol{K}_{t+1}(\boldsymbol{y}_{t+1} - \boldsymbol{C}\hat{\boldsymbol{x}}_{t+1\|t})$	(2.20)
	$\boldsymbol{P}_{t+1\|t+1} = (\boldsymbol{I} - \boldsymbol{K}_{t+1}\boldsymbol{C}_t)\boldsymbol{P}_{t+1\|t}$	(2.21)

Table 2.2: Extended Kalman Filter algorithm [8]

It is often more convenient to estimate total states rather than error states. Using equation (2.14),an updated equation of the error state Kalman filter can be derived as follows (equation(2.23)).

$$\delta\hat{\boldsymbol{x}}_{t+1|t+1} = \delta\hat{\boldsymbol{x}}_{t+1|t} + \boldsymbol{K}_{t+1}(y_t - h(\hat{\boldsymbol{x}}_{t+1|t}) - \boldsymbol{C}_t\delta\hat{x}_{t+1|t}) \tag{2.23}$$

Applying equation (2.15) to the above and using the fact that the predicted error state estimate $\delta\hat{x}_{t+1|t}$ is zero [19], an equivalent equation for the total state estimate can be formed (equation(2.24))

$$\hat{\boldsymbol{x}}_{t+1|t+1} = \hat{\boldsymbol{x}}_{t+1|t} + \boldsymbol{K}_{t+1}(\boldsymbol{y}_t - \boldsymbol{C}_t\hat{\boldsymbol{x}}_{t+1|t}) \tag{2.24}$$

The above is the standard extended Kalman filter. Both of the approaches described above lead to the same results [19]. However, the ErKF estimates the error in the states and the EKF estimates the full state. As a result, the same problem requires different state and measurement equations for each filter although the end result is the same. The filtering method that can lead to simpler equations can be chosen to solve a particular problem.

2.3 The Unscented Kalman Filter

Another method of dealing with nonlinearities is the Unscented Kalman filter. Here, a deterministic sampling approach is used. The state distribution is represented by sample points which are calculated taking the state covariance into account. These sample points, called sigma points, are then

propagated through the nonlinear equations to obtain a new set of sample points. Each of these new sample points are given a weight from which the new mean with its corresponding covariance is calculated. This method originates from the Unscented Transform, which is commonly used to estimate the statistics of a random variable subject to nonlinearities [20].

When constructing sigma points the following constraints apply; the original mean and covariance values must be obtained if reconstructed from the sigma points.

Sigma points are formed from points around the a priori state estimate. The spread of samples from the mean can be varied using scaling factors, but is generally kept at one sigma. Also, to estimate the covariance and mean, sigma points need to be weighed using appropriate factors as described below in equation (2.25) [8].

$$\begin{aligned} \lambda &= \alpha_u^2(L + \kappa_u) - L \\ \boldsymbol{\eta_0^m} &= \frac{\lambda}{L + \lambda} \\ \boldsymbol{\eta_0^c} &= \frac{\lambda}{L + \lambda} + 1 - \alpha_u^2 + \beta_u \\ \boldsymbol{\eta_i^m} &= \boldsymbol{\eta_i^c} = \frac{1}{2(L + \lambda)} \end{aligned} \tag{2.25}$$

In the above; L is the dimension of the state vector, λ is a scaling factor computed from the combination of other scaling factors giving the spread of the sigma points about the mean value, α_u gives the spread of the sigma points about the mean value, κ_u is a secondary scaling parameter which usually set to zero [20], β_u gives knowledge of the type of distributon ($\beta = 2$ for Gaussaian distributions) and $\boldsymbol{\eta}$ represents weights for the sigma points.

Increasing either α or κ will increase the spread of the sigma points from the mean value.

The mean is set as the first sigma point. Then, two sigma points are chosen for each dimension, creating $2L + 1$ sigma points. The sigma points, other than the mean, are chosen so that they are symmetrically distributed about the mean value, such that if all sigma points are added, the original mean value is obtained. The state covariance for sigma points is weighted using the $(L + \lambda)$ term. Cholesky decomposition can then be used to find the

square root of the covariance matrix (i.e. finding the standard deviation), which is then added to and subtracted from the mean to form two sets of sigma points (equation (2.26)) [20].

$$\boldsymbol{\chi}_{k-1} = \begin{bmatrix} \hat{\boldsymbol{x}}_{k-1} & \hat{\boldsymbol{x}}_{k-1} + \sqrt{(L+\lambda)\boldsymbol{P}_{k-1}} & \hat{\boldsymbol{x}}_{k-1} - \sqrt{(L+\lambda)\boldsymbol{P}_{k-1}} \end{bmatrix} \quad (2.26)$$

The sigma points, $\boldsymbol{\chi}$, are then propagated through the nonlinear state equation, forming the next set of sigma points (equation (2.27)).

$$\boldsymbol{\chi}^i_{k|k-1} = f(\boldsymbol{\chi}^i_{k-1}, \boldsymbol{u}_{k-1}), \quad i = 0, 1, ..., 2L \quad (2.27)$$

The newly found sigma points are then summed using the appropriate weight for the sigma point, to find an updated mean value.

$$\hat{\boldsymbol{x}}_{k|k-1} = \sum_{i=0}^{2L} \boldsymbol{\chi}^i_{k|k-1} \boldsymbol{\eta}^{\boldsymbol{m}}_{\boldsymbol{i}} \quad (2.28)$$

Propagating through the nonlinear equation also changes the covariance of the state variables. Using the standard covariance definition the propagated error covariance is estimated for each sigma point. These covariances are blended by summing them using the appropriate weighting factor. External influences on the states during this propagation step is captured by the process noise covariance matrix $\boldsymbol{Q}$ (equation (2.29)) [8].

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{Q}_{k-1} + \sum_{i=0}^{2L} \boldsymbol{\eta}^{\boldsymbol{c}}_{\boldsymbol{i}} (\boldsymbol{\chi}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{\chi}^i_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1})^T \quad (2.29)$$

The above equations (2.26) to (2.29) are similar to the update step of the Kalman filter.

The sigma points are then projected through the nonlinear measurement equations as shown in equation (3.122).

$$\boldsymbol{\Psi}^i_{k|k-1} = h(\boldsymbol{\chi}^i_{k-1}, \boldsymbol{u}_{k-1}), \quad i = 0, 1, ..., 2L \quad (2.30)$$

An estimate of the measurements based on the states is then computed using the weights and $\boldsymbol{\Psi}$ values obtained from the above. This is shown in equation (2.31).

$$\hat{\boldsymbol{y}}_{k|k-1} = \sum_{i=0}^{2L} \boldsymbol{\Psi}^{i}_{k|k-1} \boldsymbol{\eta}^{\boldsymbol{m}}_{\boldsymbol{i}} \tag{2.31}$$

The Kalman filter works by correcting the propagated states by using measurements. The difference between the actual measurements and the measurements estimated from the state (i.e. the residual) is related to the error in the states. The Kalman gain which is the optimal gain is used to blend the residual with the state variables, while weighting which of the state model or measurement is more accurate, based on their covariances. In the process, the error covariance is minimized.

Computing the Kalman gain in the UKF requires the computation of the cross covariance between the state and measurement (equation (2.32)) and the measurement covariance (equation (2.33)) [8]. Weighted sums are used in the computation of both the cross covariance and measurement covariance. Also, errors due to measurement sensors are captured by measurement covariance matrix R.

$$\boldsymbol{P}^{yy}_{k} = \boldsymbol{R}_k + \sum_{i=0}^{2L} \boldsymbol{\eta}^{\boldsymbol{c}}_{\boldsymbol{i}} (\boldsymbol{\Psi}^{i}_{k|k-1} - \hat{\boldsymbol{y}}_{k|k-1})(\boldsymbol{\Psi}^{i}_{k|k-1} - \hat{\boldsymbol{y}}_{k|k-1})^T \tag{2.32}$$

$$\boldsymbol{P}^{xy}_{k} = \sum_{i=0}^{2L} \boldsymbol{\eta}^{\boldsymbol{c}}_{\boldsymbol{i}} (\boldsymbol{\chi}^{i}_{k|k-1} - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{\Psi}^{i}_{k|k-1} - \hat{\boldsymbol{y}}_{k|k-1})^T \tag{2.33}$$

The Kalman gain is then calculated based on the covariances as shown in equation (2.34) [20]. This is equivalent to the way the Kalman gain is calculated in the linear Kalman filter, although different equations are used.

$$\boldsymbol{K}_k = \boldsymbol{P}^{xy}_{k} (\boldsymbol{P}^{yy}_{k})^{-1} \tag{2.34}$$

The remaining propagation equations are similar to the linear Kalman filter as shown below.

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1}) \tag{2.35}$$

$$\boldsymbol{P}_k = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{P}_k^{yy} \boldsymbol{K}_k^T \tag{2.36}$$

Equations (3.122) to (2.36) form the update stage of the Kalman filter. The recursive cycle can now be repeated by going back to equation (2.26). Table 2.3 summarizes the UKF algorithm.

2.4 Summary

In this chapter, various Kalman filtering techniques are reviewed. First, the most basic Kalman filter, the linear Kalman filter, is described. The linear Kalman filter is then adapted to suit nonlinear problems by the use of partial derivatives (Jacobians), which results in the extended Kalman filter. The extended Kalman filter is described in both total and error state forms. Finally, the unscented Kalman filter, which linearizes the system using a deterministic sampling approach, is described.

The next chapter will deal with system modeling and the application of Kalman filtering for state estimation.

Linear Kalman Filter	
Initialization	$\hat{\boldsymbol{x}}_0 \quad \text{and} \quad \boldsymbol{P}_0$ (2.37) $\lambda = \alpha_u^2(L + \kappa_u) - L$ (2.38) $\boldsymbol{\eta_0^m} = \dfrac{\lambda}{L+\lambda}$ (2.39) $\boldsymbol{\eta_0^c} = \dfrac{\lambda}{L+\lambda} + 1 - \alpha_u^2 + \beta_u$ (2.40) $\boldsymbol{\eta_i^m} = \boldsymbol{\eta_i^c} = \dfrac{1}{2(L+\lambda)}$ (2.41)
Propagation	$\boldsymbol{\chi}_{k-1} = \begin{bmatrix}\hat{\boldsymbol{x}}_{k-1} & \hat{\boldsymbol{x}}_{k-1} \pm \sqrt{(L+\lambda)\boldsymbol{P}_{k-1}}\end{bmatrix}$ (2.42) $\boldsymbol{\chi}^i_{k\|k-1} = f(\boldsymbol{\chi}^i_{k-1}, \boldsymbol{u}_{k-1}), \quad i = 0, 1, ..., 2L$ (2.43) $\hat{\boldsymbol{x}}_{k\|k-1} = \sum_{i=0}^{2L} \boldsymbol{\chi}^i_{k\|k-1}\boldsymbol{\eta_i^m}$ (2.44) $\boldsymbol{P}_{k\|k-1} = \boldsymbol{Q}_{k-1} + \sum_{i=0}^{2L} \boldsymbol{\eta_i^c}(\boldsymbol{\chi}^i_{k\|k-1} - \hat{\boldsymbol{x}}_{k\|k-1})(\boldsymbol{\chi}^i_{k\|k-1} - \hat{\boldsymbol{x}}_{k\|k-1})^T$ (2.45) $\boldsymbol{\Psi}^i_{k\|k-1} = h(\boldsymbol{\chi}^i_{k-1}, \boldsymbol{u}_{k-1}), \quad i = 0, 1, ..., 2L$ (2.46) $\hat{\boldsymbol{y}}_{k\|k-1} = \sum_{i=0}^{2L} \boldsymbol{\Psi}^i_{k\|k-1}\boldsymbol{\eta_i^m}$ (2.47)
Update	$\boldsymbol{P}_k^{yy} = \boldsymbol{R}_k + \sum_{i=0}^{2L} \boldsymbol{\eta_i^c}(\boldsymbol{\Psi}^i_{k\|k-1} - \hat{\boldsymbol{y}}_{k\|k-1})(\boldsymbol{\Psi}^i_{k\|k-1} - \hat{\boldsymbol{y}}_{k\|k-1})^T$ (2.48) $\boldsymbol{P}_k^{xy} = \sum_{i=0}^{2L} \boldsymbol{\eta_i^c}(\boldsymbol{\chi}^i_{k\|k-1} - \hat{\boldsymbol{x}}_{k\|k-1})(\boldsymbol{\Psi}^i_{k\|k-1} - \hat{\boldsymbol{y}}_{k\|k-1})^T$ (2.49) $\boldsymbol{K}_k = \boldsymbol{P}_k^{xy}(\boldsymbol{P}_k^{yy})^{-1}$ (2.50) $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k\|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k\|k-1})$ (2.51) $\boldsymbol{P}_k = \boldsymbol{P}_{k\|k-1} - \boldsymbol{K}_k\boldsymbol{P}_k^{yy}\boldsymbol{K}_k^T$ (2.52)

Table 2.3: Extended Kalman Filter algorithm [8]

Chapter 3

System Modelling

In this section the mathematical model of the robot traveling along a span will be formulated so that the attitude, position and velocity of the robot can be estimated. The position and velocity estimates will be used to perform feedback control of the robot. Moreover, from the position estimates, the shape of the power line can essentially be recreated to find the curvature of the power line. The attitude estimates will be used by the filter algorithm itself, to rotate IMU measurements to the body frame. Moreover, attitude estimates will be used to determine the pose of an on-board inspection camera.

Thus, the robot will be approximated as a point mass moving along a catenary, with an IMU and GPS on board. The attitude and translation equations will be enough to model the system.

The attitude model is constructed using the quaternion differential equation (3.1), where $\bar{q}$ is the quaternion and $\boldsymbol{\omega}$ is the angular velocity. Here the quaternion defines a rotation from the inertial to body frame and the angular velocities are measured about the body frame axes [14].

$$\dot{\bar{q}} = \frac{1}{2}\boldsymbol{\omega} \otimes \bar{q} \tag{3.1}$$

The angular velocity from the above can be written in terms of quaternions, with the scalar part equal to zero. This is obtained when deriving an equation for the quaternion derivative as shown in Appendix B. Thus, equation 3.1 which contains quaternion multiplication, is substituted by an equivalent skew symmetric matrix shown in equation 3.2. [14]

$$\dot{\bar{q}} = \frac{1}{2}[\boldsymbol{\omega}\times]\bar{q} \quad \text{where} \quad [\boldsymbol{\omega}\times] = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{3.2}$$

To complete the attitude model, noise terms affecting any measurements fed to the system must be added. Since angular velocity is obtained from a gyroscope, white noise or sensor drift in the sensor would affect the measurement. These noise parameters are usually given by gyroscope manufacturers in datasheets. The noise terms must be subtracted from the measurement to obtain the actual angular rate equation (3.3) as shown in equation (3.4) where $\boldsymbol{b_\omega}$ is the bias in the angular velocity measurement and $\boldsymbol{\eta_\omega}$ is white noise term in the angular velocity.

$$\boldsymbol{\omega} = \boldsymbol{\omega}_m - \boldsymbol{b_\omega} - \boldsymbol{n_\omega} \tag{3.3}$$

$$\dot{\bar{q}} = \frac{1}{2}[(\boldsymbol{\omega_m} - \boldsymbol{b_\omega} - \boldsymbol{\eta_\omega})\times]\bar{q} \tag{3.4}$$

It will be convenient to write equation (3.4) in terms of system states and noise. In the filter the bias will be a state which will be estimated. A new matrix $\Lambda(\bar{q})$ will be introduced so that the above equation can more easily be written in terms of its states $\bar{q}$ and $\boldsymbol{b_\omega}$ and noise $\boldsymbol{\eta_\omega}$ as shown in equation (3.5). It should be noted that multiplying $\boldsymbol{\Lambda}(\bar{q})$ by a vector is equivalent to multiplying a skew symmetric matrix formed by the same vector by $\bar{q}$ [14]. It is possible to use the above matrix since the last dimension (scalar part) of the bias and noise terms are zero if converted to quaternions [14].

$$\dot{\bar{q}} = \frac{1}{2}[\boldsymbol{\omega}\times]\bar{q} - \frac{1}{2}\boldsymbol{\Lambda}(\bar{q})(\boldsymbol{b_\omega} + \boldsymbol{\eta_\omega}) \quad \text{where} \quad \boldsymbol{\Lambda}(\bar{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{3.5}$$

The translational motion equations are constructed from velocity and acceleration equations. The acceleration will be obtained from a body frame accelerometer, and will be rotated into the reference frame. The reference frame is the inertial frame rotated about the z axis to lie along the line direction. Equations (3.6) and (3.7) describe the above, where $\boldsymbol{s}$ is the position, $\boldsymbol{v}$ is the velocity, $\boldsymbol{a}$ is the acceleration measured by the accelerometer and $R(\bar{q})^T$ is the rotation matrix rotating from body to reference frame as shown in Appendix A.

$$\dot{\boldsymbol{s}} = \boldsymbol{v} \tag{3.6}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{R}(\bar{q})^T \boldsymbol{a} + \boldsymbol{g} \tag{3.7}$$

Similar to the attitude model, random white noise and drift bias from the accelerometer must be included (equations (3.8) and (3.9)).

$$\boldsymbol{a_m} = \boldsymbol{a} + \boldsymbol{b_a} + \boldsymbol{n_a} \tag{3.8}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{R}(\bar{q})^T (\boldsymbol{a_m} - \boldsymbol{b_a} - \boldsymbol{\eta_a}) + \boldsymbol{g} \tag{3.9}$$

Drift or bias noise in IMUs is caused mainly due to temperature while factors such as unmodeled vibrations and rotation errors may introduce large errors. Even with temperature compensation it is important to estimate bias noise and deduct it after each iteration. However, bias noise is coloured in nature and therefore very difficult to estimate. It is common practice to approximate the rate of change of bias to random white noise(equation 3.10). This model usually provides a decent drift compensation.

$$\dot{\boldsymbol{b}}_{\boldsymbol{a}} = \boldsymbol{\eta_{ba}} \quad \text{and} \quad \dot{\boldsymbol{b}}_{\boldsymbol{\omega}} = \boldsymbol{\eta_{b\omega}} \tag{3.10}$$

3.1 Linearization Using Jacobians

It can be seen above that equation (3.9) is nonlinear. This means that an EKF can be used to estimate the states shown in equation (3.11).

$$\boldsymbol{x} = \begin{bmatrix} \bar{q}^T & \boldsymbol{b_\omega}^T & \boldsymbol{s}^T & \boldsymbol{v}^T & \boldsymbol{b_a}^T \end{bmatrix}^T \tag{3.11}$$

Implementing the EKF requires forming matrices A and C of equation (2.16) in the previous chapter.

Among the various differential equations, equation (3.9) requires special attention. The derivative of the rotation matrix needs to be found with respect to the quaternions. This Jacobian matrix will be denoted by sub matrix $\boldsymbol{A_{41}}$ and will be constructed by using the equation (3.12), as this will simplify the problem. (The computation of $\hat{\boldsymbol{\omega}}$ and $\hat{\boldsymbol{a}}$ can be found in equations (3.36) and equation (3.49) respectively)

$$\boldsymbol{A_{41}} = \frac{\boldsymbol{\partial \dot{v}}}{\boldsymbol{\partial q}}\bigg|_{\hat{\boldsymbol{x}}} = \left[\frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_1}}\hat{\boldsymbol{a}} \quad \frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_2}}\hat{\boldsymbol{a}} \quad \frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_3}}\hat{\boldsymbol{a}} \quad \frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_4}}\hat{\boldsymbol{a}}\right]\bigg|_{\hat{\boldsymbol{x}}} \tag{3.12}$$

The various derivatives $\dfrac{\boldsymbol{dR^T}}{\boldsymbol{dq_i}}$ are given by the four equations below in (3.13).

$$\begin{aligned}
\frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_1}} &= \begin{bmatrix} 0 & 2q_2 & 2q_3 \\ 2q_2 & -4q_1 & 2q_4 \\ 2q_3 & -2q_4 & -4q_1 \end{bmatrix} \quad \frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_2}} = \begin{bmatrix} -4q_2 & 2q_1 & -2q_4 \\ 2q_1 & 0 & 2q_3 \\ 2q_4 & 2q_3 & -4q_2 \end{bmatrix} \\
\frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_3}} &= \begin{bmatrix} -4q_3 & 2q_4 & 2q_1 \\ -2q_4 & -4q_3 & 2q_2 \\ 2q_1 & 2q_2 & 0 \end{bmatrix} \quad \frac{\boldsymbol{\partial R^T}}{\boldsymbol{\partial q_4}} = \begin{bmatrix} 0 & 2q_3 & -2q_2 \\ -2q_3 & 0 & 2q_1 \\ 2q_2 & -2q_1 & 0 \end{bmatrix}
\end{aligned} \tag{3.13}$$

The full Jacobian matrix of the system is given in equation (3.14) below.

$$\boldsymbol{A} = \begin{bmatrix} \frac{1}{2}[\hat{\boldsymbol{\omega}}\times] & -\frac{1}{2}\boldsymbol{\Lambda}(\hat{\bar{q}}) & \boldsymbol{0_{4\times 3}} & \boldsymbol{0_{4\times 3}} & \boldsymbol{0_{4\times 3}} \\ \boldsymbol{0_{3\times 4}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} \\ \boldsymbol{0_{3\times 4}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{I_{3\times 3}} & \boldsymbol{0_{3\times 3}} \\ \boldsymbol{A_{41}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{R}(\hat{\bar{q}})^T \\ \boldsymbol{0_{3\times 4}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} & \boldsymbol{0_{3\times 3}} \end{bmatrix} \tag{3.14}$$

Non-additive noise is also linearized using partial derivatives or Jacobians. For a nonlinear differential equation of the form shown in equation (3.15), the covariance can be obtained using equation (3.16). In equation (3.16) matrix $\boldsymbol{G}$ represents the Jacobian matrix and $\boldsymbol{Q_c}$ represents the diagonal matrix with variances of the noisy signals in the continuous time domain assuming the noises are uncorrelated. [20]

$$\dot{x} = f(x, u, w) \tag{3.15}$$

$$\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{Q_c}\boldsymbol{G^T} = \left(\frac{\partial f}{\partial w}\bigg|_{\hat{\boldsymbol{x}}}\right) \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \sigma_n^2 \end{bmatrix} \left(\frac{\partial f}{\partial w}\bigg|_{\hat{\boldsymbol{x}}}\right)^T \tag{3.16}$$

Applying the above to the system, matrices $\boldsymbol{G}$ and $\boldsymbol{Q_c}$ can be found (See equations (3.17) and (3.18)).

$$\boldsymbol{Q_c} = \begin{bmatrix} \sigma_\omega^2 & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \sigma_{b\omega}^2 & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \sigma_a^2 & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \sigma_{ba}^2 \end{bmatrix} \tag{3.17}$$

$$\boldsymbol{G} = \begin{bmatrix} -\frac{1}{2}\boldsymbol{\Lambda}(\hat{\bar{q}}) & \mathbf{0_{4\times3}} & \mathbf{0_{4\times3}} & \mathbf{0_{4\times3}} \\ \mathbf{0_{3\times3}} & \boldsymbol{I_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & -\boldsymbol{R}(\hat{\bar{q}})^T & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} & \boldsymbol{I_{3\times3}} \end{bmatrix} \tag{3.18}$$

3.2 Discretization

The continuous domain equations need to be discretized before applying them to the discrete Kalman filter. The state transition matrix used in the Kalman filter is given in equation (3.19), where T is the sampling time applied to the system [11].

$$\boldsymbol{\Phi} = e^{\boldsymbol{A}T} \tag{3.19}$$

The covariance matrix also needs to be discretized. This is achieved using the following expression (equation (3.20)), where Q_d is the discrete process noise covariance [11].

$$\boldsymbol{Q_d} = \int_T^0 e^{\boldsymbol{A}T}\boldsymbol{G}\boldsymbol{Q_c}\boldsymbol{G^T}e^{\boldsymbol{A^T}T}d\tau \tag{3.20}$$

Evaluating $\boldsymbol{Q_d}$ numerically is tedious, when it comes to systems with large dimensionality, due to the integral and matrix exponential involved. However, this can be evaluated using Van Loan's method, which is fairly easy to implement using MATLAB. First a matrix $\boldsymbol{L}$, with dimensions $2n \times 2n$ where n is the dimension of the state matrix A, is constructed as shown in equation (3.21) [8].

$$\boldsymbol{L} = \begin{bmatrix} -\boldsymbol{A} & \boldsymbol{G}\boldsymbol{Q_c}\boldsymbol{G^T} \\ \boldsymbol{0} & \boldsymbol{A^T} \end{bmatrix} T \tag{3.21}$$

The matrix exponential of $\boldsymbol{L}$ from the above, which is denoted by matrix $\boldsymbol{M}$ in the equation (3.22) below, is then found. The upper left corner of this matrix is is not used, since all the required matrices can be obtained using the second column of matrix $\boldsymbol{L}$. The state transition matrix can be found by transposing the lower right corner matrix of $\boldsymbol{M}$ [8].

$$\boldsymbol{M} = e^{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{\cdots} & \boldsymbol{\Phi^{-1}}\boldsymbol{Q_d} \\ \boldsymbol{0} & \boldsymbol{\Phi^T} \end{bmatrix} \tag{3.22}$$

The above can easily be implemented in MATLAB using the *expm*() function.

3.3 Propagation

Closed form integration methods, such as equation (3.23) with T being the sampling time, can be used to obtain higher degrees of accuracy when propagating discrete equations [8].

$$\boldsymbol{x}_{k+1} = e^{\boldsymbol{A}T}\boldsymbol{x}_k = (1 + \boldsymbol{A}T + \frac{1}{2}(\boldsymbol{A}T)^2 ...)\boldsymbol{x}_k \tag{3.23}$$

The above Taylor series expansion, can be truncated up to the first order terms. This results in Euler integration which is used in equations (3.24), (3.25) and (3.26) below. It must be noted that equation (3.24) only holds true if angular acceleration is close to zero or the sampling frequency is high [1]. Also, bias differential equations do not need to be propagated as they have zero expectation (i.e only consist of white noise)

$$\bar{q}_{k+1} = (\boldsymbol{I_{4\times 4}} + \frac{1}{2}[(\boldsymbol{\omega_m} - \boldsymbol{b_\omega})\times]T)\bar{q}_k \tag{3.24}$$

$$\boldsymbol{s}_{k+1} = \boldsymbol{s}_k + T(\boldsymbol{v}_k) \tag{3.25}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + (\boldsymbol{R}(\bar{q})^T\boldsymbol{a_m} - \boldsymbol{b_a} + \boldsymbol{g})T \tag{3.26}$$

The biases are propagated using equation (3.27).

$$\boldsymbol{b}_{k+1} = \boldsymbol{b}_k \tag{3.27}$$

The error state covariance matrix is computed following equation (2.7) of the previous chapter (equation (3.28)).

$$\boldsymbol{P}_{k+1} = \boldsymbol{\Phi}\boldsymbol{P}_k\boldsymbol{\Phi}^{\boldsymbol{T}} + \boldsymbol{Q_d} \tag{3.28}$$

3.4 Error State Model

The extended Kalman filter can be used to estimate full states or error states. An error state filter will be implemented and compared to the full state version of the EKF. Hence, an error state system model needs to be formulated.

In this application, an error state filter can be advantageous due to the fact that small angle approximation can be used to convert quaternions and the DCM into error states.

For small angles the quaternion vector can be approximated using equation (3.29).

$$\begin{bmatrix} \hat{\boldsymbol{e}}\sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{\delta\theta}/2 \\ 1 \end{bmatrix} = \begin{bmatrix} \vec{\boldsymbol{\delta q}} \\ 1 \end{bmatrix} \tag{3.29}$$

Both $\boldsymbol{\delta\theta}$ and $\vec{\boldsymbol{\delta q}}$ can be used as states (the states are interchangeable using a factor of half as shown in the equation above). In this project $\vec{\boldsymbol{\delta q}}$ will be used. It is important to note that the fourth element of the quaternion q_4 will be omitted, to avoid singularity issues with the associated covariance matrix. In fact, it is difficult to maintain the unit norm constraint on the quaternion because of round-off errors, which leads to instability. Reducing the dimension of the quaternion solves this problem [21].

Usually, the error state is added to the nominal value. However, a multiplicative relationship (equation (3.30)) will be used for quaternion error states since it makes the quaternion unit norm constraint easier to keep.

$$\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}} \tag{3.30}$$

By taking the derivative of equation (3.30) a differential equation for the quaternion error can be found (equation 3.31).

$$\dot{\bar{q}} = \dot{\delta\bar{q}} \otimes \hat{\bar{q}} + \delta\bar{q} \otimes \dot{\hat{\bar{q}}} \tag{3.31}$$

Using the quaternion rate definition of equation (3.1), expressions for $\dot{\bar{q}}$ and $\dot{\hat{\bar{q}}}$ are substituted in the above equation, thus giving equation (3.32). In the equation below, the angular velocity vector has been written in terms of a quaternion with the scalar part set as zero following the derivation in Appendix B [14].

$$\dot{\bar{q}} = \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}\\ 0\end{bmatrix} \otimes \bar{q} = \dot{\delta\bar{q}} \otimes \hat{\bar{q}} + \delta\bar{q} \otimes \left(\frac{1}{2}\begin{bmatrix}\hat{\boldsymbol{\omega}}\\ 0\end{bmatrix} \otimes \hat{\bar{q}}\right) \tag{3.32}$$

From, equation (3.30) an expression for the quaternion error can be obtained by post multiplying both sides by $\hat{\bar{q}}^{-1}$. This is shown below in equation (3.33).

$$\delta\bar{q} = \bar{q} \otimes \hat{\bar{q}}^{-1} \tag{3.33}$$

Rearranging equation (3.32), and performing quaternion multiplication on both sides by $\hat{\bar{q}}^{-1}$, results in the deterministic quaternion error state equation (3.34).

$$\dot{\delta\bar{q}} = \frac{1}{2}\left(\begin{bmatrix}\boldsymbol{\omega}\\ 0\end{bmatrix} \otimes \delta\bar{q} - \delta\bar{q} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}\\ 0\end{bmatrix}\right) \tag{3.34}$$

Now, noise and drift terms need to be added to complete the equation. The term $\boldsymbol{\omega}$ is the true angular velocity value and can be written in terms of the measured angular velocity, a bias and Gaussian noise (equation (3.35)).

$$\boldsymbol{\omega} = \hat{\boldsymbol{\omega}}_{\boldsymbol{m}} - \boldsymbol{b}_{\boldsymbol{\omega}} - \boldsymbol{n}_{\boldsymbol{\omega}} \tag{3.35}$$

Next, the estimated angular velocity is written as the following (equation (3.36)).

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega_m} - \hat{\boldsymbol{b_\omega}} \tag{3.36}$$

Using the fact that the true bias is the sum of the estimated bias and the error bias (equation (3.37)), and by combining equations (3.35) and (3.36), the true angular velocity equation is obtained (equation (3.38)).

$$\boldsymbol{b_\omega} = \hat{\boldsymbol{b}}_{\boldsymbol{\omega}} + \boldsymbol{\delta b_\omega} \tag{3.37}$$

$$\boldsymbol{\omega} = \hat{\boldsymbol{\omega}} - \boldsymbol{\delta b_\omega} - \boldsymbol{n_\omega} \tag{3.38}$$

Substituting the above expression in equation (3.34), and rearranging the equation such that the deterministic and stochastic parts are separated, equation (3.39) is obtained.

$$\dot{\delta\bar{q}} = \frac{1}{2}\left(\begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \otimes \delta\bar{q} - \delta\bar{q} \otimes \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \right) - \frac{1}{2} \begin{bmatrix} \boldsymbol{\delta b_\omega} + \boldsymbol{n_\omega} \\ 0 \end{bmatrix} \otimes \delta\bar{q} \tag{3.39}$$

The above is then manipulated [14] by replacing the quaternion multiplication operators by their respective skew matrices and eliminating some of the infinitesimal second order terms (i.e. the terms where the noises $\boldsymbol{\delta b_\omega}$ and $\boldsymbol{n_\omega}$, are multiplied with $\delta\bar{q}$ are eliminated). Thus, equation (3.40) for the error state quaternion is obtained. As discussed previously, the fourth element (scalar part) of the quaternion is omitted.

$$\dot{\boldsymbol{\delta q}} = -[\hat{\boldsymbol{\omega}}\times]\boldsymbol{\delta q} - \frac{1}{2}(\boldsymbol{\delta b_\omega} + \boldsymbol{n_\omega}) \tag{3.40}$$

The rate of change of the bias error remains equal to a white noise term describing the gyroscope's drift.

Velocity and Acceleration Error State Equations

The position, velocity and bias error states are additive errors . Therefore, the true value is equal to the sum of the actual estimates and the error. (equations in (3.41))

$$\boldsymbol{s} = \hat{\boldsymbol{s}} + \boldsymbol{\delta s}$$
$$\boldsymbol{v} = \hat{\boldsymbol{v}} + \boldsymbol{\delta v} \tag{3.41}$$
$$\tag{3.42}$$

The error state differential equations (3.43) are formed by adding the above to equations (3.6) and (3.7). However, the quaternion multiplicative error must be added to the rotation matrix of equation (3.7).

$$\boldsymbol{\delta \dot{s}} = \dot{\boldsymbol{s}} - \dot{\hat{\boldsymbol{s}}}$$
$$\boldsymbol{\delta \dot{v}} = \dot{\boldsymbol{v}} - \dot{\hat{\boldsymbol{v}}} \tag{3.43}$$

When forming the error state velocity equation, the rotation matrix in equation (3.6) is assumed to be a constant and is factorized leaving out the $\boldsymbol{v} - \hat{\boldsymbol{v}}$ term which is then replaced by the velocity error state (equation(3.44)).

$$\boldsymbol{\delta \dot{s}} = \boldsymbol{\delta v} \tag{3.44}$$

When constructing the error state acceleration equation, a DCM containing the true quaternions is obtained (equation (3.45)), while the gravity term is eliminated. This matrix must be converted in terms of either estimated quaternions or error quaternions and is achieved by using Euler's equation for DCMs (equation (3.46)).

$$\boldsymbol{\delta \dot{v}} = \boldsymbol{R}(\bar{q})^T \boldsymbol{a} - \boldsymbol{R}(\hat{\bar{q}})^T \hat{\boldsymbol{a}} \tag{3.45}$$

$$\boldsymbol{R(q)} = (2q_4^2 - 1) I_{3\times 3} - 2q_4[\boldsymbol{q}\times] + 2\boldsymbol{q}\boldsymbol{q}^{\boldsymbol{T}} \tag{3.46}$$

Using small angle approximation (equation (3.29)) in the above gives equation (3.47). [13]

$$\boldsymbol{R(\delta q)} \approx I_{3\times 3} - 2[\boldsymbol{\delta q}\times] = \begin{bmatrix} 1 & 2\delta q_3 & -2\delta q_2 \\ -2\delta q_3 & 1 & 2\delta q_1 \\ 2\delta q_2 & -2\delta q_1 & 1 \end{bmatrix} \tag{3.47}$$

The DCM in terms of true quaternions can be broken down using quaternion multiplication. This can then be simplified using matrix multiplication (equation (3.48)).

$$\boldsymbol{R}(\bar{q})^T = \boldsymbol{R}(\delta\bar{q} \otimes \hat{\bar{q}})^T = \boldsymbol{R}(\hat{\bar{q}})^T \boldsymbol{R}(\delta\bar{q})^T \tag{3.48}$$

Next, white noise and drift in the accelerometer signal need to be added. The accelerometer noise treatment is similar to that of the gyroscope, and is given by the set of equations in (3.49).

$$\begin{aligned} \boldsymbol{a_m} &= \boldsymbol{a} + \boldsymbol{b_a} + \boldsymbol{n_a} \\ \boldsymbol{b_a} &= \hat{\boldsymbol{b}}_{\boldsymbol{a}} + \boldsymbol{\delta b_a} \\ \hat{\boldsymbol{a}} &= \boldsymbol{a_m} - \hat{\boldsymbol{b_a}} \\ \boldsymbol{a} &= \hat{\boldsymbol{a}} - \boldsymbol{\delta b_a} - \boldsymbol{n_a} \end{aligned} \tag{3.49}$$

Now, substituting equations (3.48), (3.47) and (3.49) in equation (3.45) forms equation (3.50).

$$\begin{aligned} \dot{\boldsymbol{\delta v}}_{\boldsymbol{ned}} = 2\boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}}[\boldsymbol{\delta q}\times]\hat{\boldsymbol{a}} + \boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}}(-\boldsymbol{\delta b} - \boldsymbol{n_a}) \\ + [2\boldsymbol{\delta q}\times]\boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}}(-\boldsymbol{\delta b} - \boldsymbol{n_a}) \end{aligned} \tag{3.50}$$

The last term of the above equation can be approximated to zero since the error quaternion is multiplied by noise states. Then, using the identity, $[\boldsymbol{a}\times]\boldsymbol{b} = -[\boldsymbol{b}\times]\boldsymbol{a}$, the equation can be rearranged to give the final state equation (3.51).

$$\dot{\boldsymbol{\delta v}}_{\boldsymbol{ned}} = -2\boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}}[\hat{\boldsymbol{a}}\times]\boldsymbol{\delta q} + \boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}}(-\boldsymbol{\delta b} - \boldsymbol{n_a}) \tag{3.51}$$

The error state will be denoted by $\boldsymbol{\delta x}$ and is shown in equation (3.52).

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta q^T} & \boldsymbol{\delta b_\omega^T} & \boldsymbol{\delta s^T} & \boldsymbol{\delta v^T} & \boldsymbol{\delta b_a^T} \end{bmatrix}^T \tag{3.52}$$

Using equations (3.40), (3.44) and (3.51), it is possible to write an expression for the state matrix $\boldsymbol{A}$ (equation (3.53)). There is no need for linearization in this case as the differential equations are in a linear form.

$$\boldsymbol{A} = \begin{bmatrix} -[\hat{\boldsymbol{\omega}}\times] & -\frac{1}{2}\boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times3}} \\ -2\boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}}[\hat{\boldsymbol{a}}\times] & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & -\boldsymbol{R}(\hat{\boldsymbol{q}})^{\boldsymbol{T}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \end{bmatrix} \tag{3.53}$$

The covariance matrix for the sensors used is given by equation(3.17)). Matrix $\boldsymbol{G}$ which is to be applied to $\boldsymbol{Q_c}$, according to equation (3.16) is given below.

$$\boldsymbol{G} = \begin{bmatrix} -\frac{1}{2}\boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & -\boldsymbol{R}(\bar{\boldsymbol{q}})^{\boldsymbol{T}} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{I_{3\times3}} \end{bmatrix} \tag{3.54}$$

The state transition matrix and the discrete process noise covariance matrix can be found using Van Loan's method from section 1.2. Equations (3.24) to (3.27), which use full states (i.e $\boldsymbol{x}$), are used for propagation. However, after each iteration, the filter estimates the error states which are injected into the full states using equations (3.30) and (3.41).

3.5 The Measurement Model

Most INS/GPS estimation solutions use GPS position, GPS velocity and magnetometers or cameras for attitude, as measurements. Considering the nature of this project it is not practical to use magnetometers as the 50 Hz magnetic field of the power line (for instance, 1 kA current gives a value of 100 μT at a distance of one meter from the line conductor) is comparable to the Earth's magnetic field (24-65 μT) [1]. Thus, the direction of the power line in the horizontal plane, known from GIS survey information, and the shape it makes in the vertical plane can be used as measurements.

NMEA Option	Data Output
GGA	Time, Latitude, Longitude, Fix quality, Number of satellites, HDOP, Altitude
GSA	Type of fix, PDOP, HDOP, VDOP
RMC	Time, Latitude, Longitude, Speed over ground, Track angle, Date, magnetic variation

Table 3.1: NMEA strings descriptions [9].

3.5.1 GPS Measurements

Either tight or loose coupling approaches could be used to integrate GPS measurements in the Kalman filter. The complexity of the tight coupling approach outweighs its benefits in this application. Therefore loose coupling will be used since it provides reasonable accuracy with relatively little effort.

A GPS module usually provides the user with NMEA data [9]. NMEA strings contain a lot of information, which need to be parsed to extract the position, velocity and DOP values. The only NMEA sentences needed are the GGA, GSA and RMC strings (as seen in Table 3.1).

The reference frame will be constructed by choosing the starting tower of the robot as the origin, and the distances moved along the x and y axes of the reference frame will be calculated from the longitudes and latitudes obtained from the GPS module.

The Haversine formula is commonly used to measure the distance between two points given in latitude and longitude. It is one of the most accurate methods but requires a lot of computation. However, there are less cumbersome methods that can be used with good accuracy over short distances. The earth is assumed to be spherical with a radius of $R_e = 6371$ kilometers, and flat for movement over short distances. Thus, the equirectangular approximation can be used as shown in the equations (3.55) below.

$$\Delta\lambda_L = \lambda_{L_2} - \lambda_{L_1} \quad \text{and} \quad \Delta\Phi_L = \Phi_{L_2} - \Phi_{L_1} \tag{3.55}$$

In the above λ represents the latitude and Φ the longitude in degrees. If λ_1 and Φ_1 is the starting coordinate, then the distances in x and y directions about the point is given by equations (3.56) and (3.57).

$$x = R_e.\Delta\lambda.\cos((\Phi_{L_2} + \Phi_{L_1})/2) \tag{3.56}$$

$$y = R_e.\Delta\Phi_L \tag{3.57}$$

Assuming the Earth's surface is flat, the distance between the two coordinates can easily be calculated using Pythagoras's theorem. Altitude in meters (i.e the z measurement) and velocity in knots (which can be converted to m/s) are obtained as direct measurements from the RMC sentence.

The measurement equations for GPS signals, with η representing measurement noise signals, can be written as follows (equations (3.58) and (3.59)), where ψ_i is the yaw angle as viewed from the inertial frame).

$$s_m = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \eta_{s_{gps}} \tag{3.58}$$

$$v_m = v_{gps} + \eta_{v_{gps}} \tag{3.59}$$

The measured velocity in the above equation represents the velocity along the line in the horizontal plane, and is therefore the velocity along the x axis in the reference frame.

Since tower positions are surveyed, they can be used to obtain the direction of the power line and therefore the yaw angle (shown in equation (3.60), (x_1, y_1) and (x_2, y_2) are the first and second tower coordinates respectively). The yaw angle remains almost constant throughout the motion. If there are two grippers that grip the line, the robot is restricted to follow the line direction as most swinging is related to the roll angle. Also, the towers are surveyed to centimeter accuracy, which makes the yaw error due to GIS surveyed measurements negligible.

$$\psi_i = atan2(\frac{y_2 - y_1}{x_2 - x_1}) \tag{3.60}$$

The GPS horizontal measurements can be projected on the power line as a way of enhancing the measurement. First, the initial and final positions (A and B) of the robot will be translated such that, the initial position will be

moved to the origin of the inertial frame, forming new points $\boldsymbol{A'}$ and $\boldsymbol{B'}$, as shown in figure 3.1. The equations in (3.61), show the vector subtraction required.

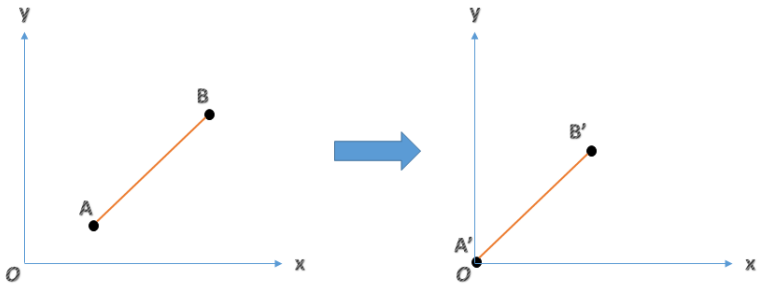

Figure 3.1: Translation of power line beginning and end points.

$$\begin{aligned} \boldsymbol{A'} &= \boldsymbol{A} - \boldsymbol{A} = \boldsymbol{0} \\ \boldsymbol{B'} &= \boldsymbol{B} - \boldsymbol{A} \end{aligned} \tag{3.61}$$

Naturally, all the GPS x and y measurements, will undergo a similar vector translation (equation (3.62)) to form point $\boldsymbol{C'}$.

$$\boldsymbol{C'} = \begin{bmatrix} x'_{gps} \\ y'_{gps} \end{bmatrix} = \begin{bmatrix} x_{gps} \\ y_{gps} \end{bmatrix} - \boldsymbol{A} = \boldsymbol{C} - \boldsymbol{A} \tag{3.62}$$

The new GPS measurements will then be perpendicularly projected on the power line forming the new measurement $\boldsymbol{D'}$ (See figure 3.2) using equation (3.63).

$$\boldsymbol{D'} = \frac{\boldsymbol{C'} \cdot \boldsymbol{B'}}{\boldsymbol{B'} \cdot \boldsymbol{B'}} \boldsymbol{B'} \tag{3.63}$$

The coordinates of the projected point $\boldsymbol{D'}$ are denoted by x_p and y_p respectively and the coordinates of point $\boldsymbol{B'}$ by x_2 and y_2. Also, for convenience $\boldsymbol{C'}$ coordinates will be represented by x_g and y_g.

Thus, the projected x and y coordinates can be written in terms of the translated end point of the line and the translated GPS measurements using equation (3.63) (as shown below in (3.64) and (3.65)).

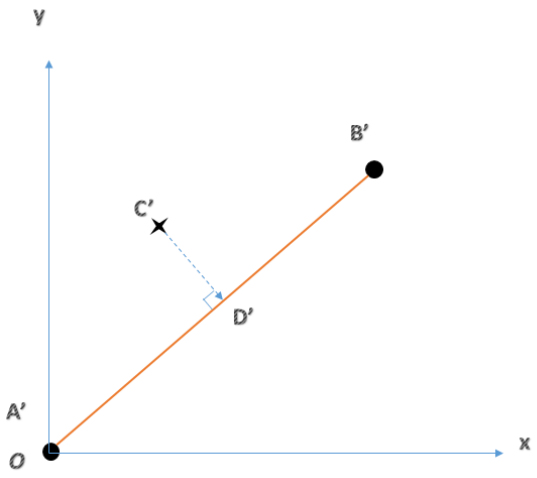

Figure 3.2: Perpendicular projection of GPS translated latitude and longitude.

$$x_p = \frac{x_g x_2^2 + y_g y_2 x_2}{x_2^2 + y_2^2} \tag{3.64}$$

$$y_p = \frac{y_g y_2^2 + x_g y_2 x_2}{x_2^2 + y_2^2} \tag{3.65}$$

Using equation (3.16), an expression for the variances of the above can be found. It should be noted that in this equation, it is assumed that only the GPS measurements have associated noise, resulting in (3.66) and (3.67).

$$\sigma_{x_p}^2 = \left(\frac{x_2^2}{x_2^2 + y_2^2}\right)^2 \sigma_{x_g}^2 + \left(\frac{x_2 y_2}{x_2^2 + y_2^2}\right)^2 \sigma_{y_g}^2 \tag{3.66}$$

$$\sigma_{y_p}^2 = \left(\frac{x_2 y_2}{x_2^2 + y_2^2}\right)^2 \sigma_{x_g}^2 + \left(\frac{y_2^2}{x_2^2 + y_2^2}\right)^2 \sigma_{y_g}^2 \tag{3.67}$$

GPS accuracy is normally expressed in 2D accuracy. This can be in the form of DRMS (Distance root mean square) or CEP (Circular error probability). Any measured point can lie around a circle about the real position with radius given by CEP or DRMS. The probability of a measured point lying

in the circle is 50% for CEP, 65% for DRMS and 99% for 2DRMS. Thus, DRMS represents the noise in the Gaussian sense and is to be used in the filter. In terms of rectangular coordinates variance, DRMS is expressed as follows. [16]

$$DRMS = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{3.68}$$

DRMS accuracy is the averaged accuracy of the GPS module over a period of time. In practice, the variance changes regularly with the satellite constellation geometry. This is resolved by using the DOP (Dilution of Precision) values (equations (1.10) and (1.11)).

The covariance matrix of GPS measurements is usually given by the equation below, which forms an ellipsoid geometrically. To obtain the cross covariance terms, the correlation coefficient between the different variables is required. However, this information is not readily available in GPS module datasheets. Hence the cross covariance terms will be assumed to be zero, further on.

$$\boldsymbol{Q}_{gps} = \begin{bmatrix} \sigma_{x_g}^2 & \sigma_{x_g y_g} & \sigma_{x_g z_g} \\ \sigma_{y_g x_g} & \sigma_{y_g}^2 & \sigma_{y_g z_g} \\ \sigma_{z_g x_g} & \sigma_{z_g y_g} & \sigma_{z_g}^2 \end{bmatrix} \tag{3.69}$$

For the Kalman filter measurement equations, the GPS σ_x and σ_y values are required instead of DRMS. It will be assumed that the longitude and latitude measurements have the same standard deviation, σ_x and σ_y respectively. Equation (3.70) gives the horizontal variance.

$$\sigma_x^2 = \sigma_y^2 = \left(\frac{DRMS}{\sqrt{2}} HDOP\right)^2 = \sigma_h^2 \tag{3.70}$$

Using the above assumption, equations (3.64) and (3.65) can be simplified to produce (3.71) and (3.72).

$$\sigma_{x_p}^2 = \frac{x_2^2}{x_2^2 + y_2^2}\sigma_h^2 \tag{3.71}$$

$$\sigma_{y_p}^2 = \frac{y_2^2}{x_2^2 + y_2^2}\sigma_h^2 \tag{3.72}$$

The measurement equations for x_p and y_p are given in terms of the distance moved along the line s_1 and the known yaw angle (equations in (3.73)).

$$\begin{aligned} x_p &= x\cos(\psi) \\ y_p &= x\sin(\psi) \end{aligned} \tag{3.73}$$

The vertical position will directly be used as measurement with the appropriate variance (as in equation (3.58)). Hence, VDOP (Vertical Dilution of Precision) will be used instead of HDOP to calculate the height variance (equation (3.74)). VDOP is usually always higher than HDOP, resulting in much larger errors for vertical measurements.

$$\sigma_{z_v}^2 = \sigma_z * VDOP \tag{3.74}$$

3.6 The Power Line Shape and Direction

When the powerline is hung between two points it takes the shape of a catenary due to its own weight. The sag of a transmission line conductor depends on its catenary constant which is shown in equation (3.75) (where ζ is the catenary constant, H is the horizontal component of tension along the conductor in Newtons and w is the weight per unit length of the conductor in Newtons per meter). In South Africa, transmission lines are usually designed with a catenary constant of $1800m$ [22].

$$\zeta = \frac{H}{w} \tag{3.75}$$

Assuming that the weight of the cable is spread uniformly over its length the power line is modeled as a catenary (as shown in figure 3.3) with origin at the nadir (i.e. lowest point/ point B in figure 3.3) using equation (3.76) [22].

$$z = \frac{h}{s}(x - x_1)\zeta(cosh(\frac{x_1}{\zeta}) - \frac{x}{\zeta})) \tag{3.76}$$

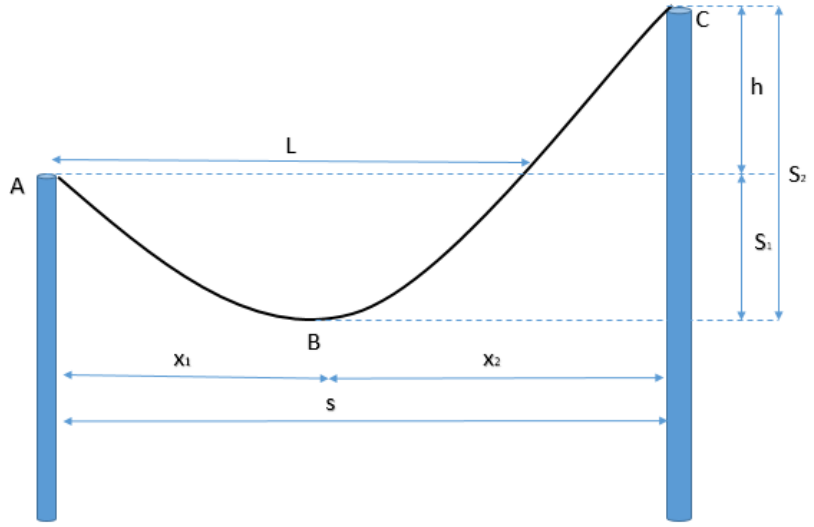

Figure 3.3: Typical power line shape with height difference between supports

As the sag of the conductor is much less than the span, its shape can be approximated by a parabolic function to simplify the model. Suspension towers usually have conductors hanging on insulators that support the vertical forces only. Thus, the forces (tension) acting along each of the spans in a transmission line must be equal, so that the system remains in equilibrium. Therefore, the catenary constant should also be equal along the different spans[23].

It is possible to obtain a relation for a parabolic model of the line in terms of the catenary constant. For a level span the minimum point lies at the midspan as illustrated in figure 3.4.

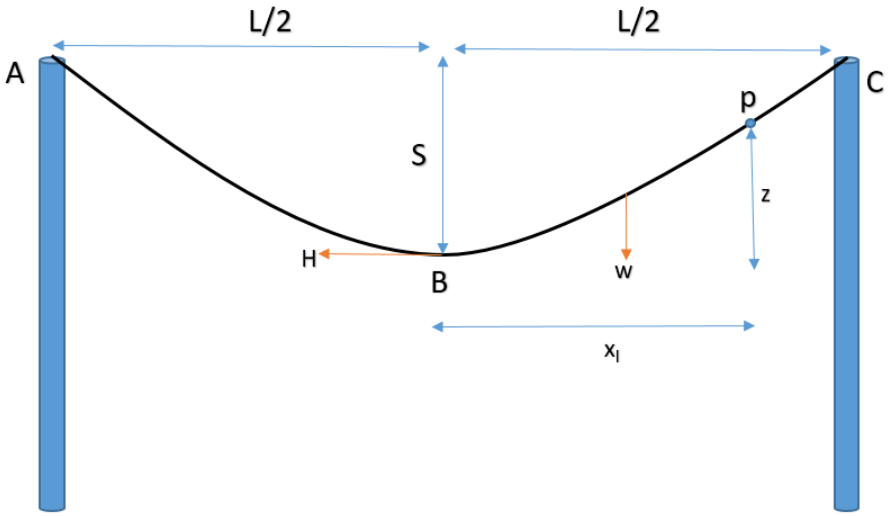

Figure 3.4: Forces acting on the power line

For a distance x_l from point B to a general point p on the line, the weight will act half the distance if the weight is uniformly distributed (figure 3.4). The resulting moment equation about point B is equation (3.77), which is a parabola with origin at point B.

$$H.z = x_l.w.\frac{x_l}{2} \implies z = \frac{x_l^2 w}{2H} = \frac{x_l^2}{2\zeta} \tag{3.77}$$

Hence for a wire span L, the maximum sag S occuring at the midspan $(L/2)$ is computed in equation (3.78).

$$S = \frac{L^2}{8\zeta} \tag{3.78}$$

For modelling, it will be convenient if the origin is moved at point A of figure (3.4) rather than being at point B, so that it matches the reference frame coordinate system. Thus, an equation for the vertical displacement z in terms of its horizontal displacement x along the line is shown in equation (3.79)) below [22].

$$z = \frac{L^2}{8\zeta}\left(\frac{x^2}{L} - \frac{x}{L}\right) = \frac{x^2}{2\zeta} - \frac{Lx}{2\zeta} \tag{3.79}$$

The above relationship takes the form of a quadratic function and can be proven as follows.

The minimum point of a quadratic function is found at $(\frac{-b}{2a}, \frac{-b^2}{4a} + c)$. If point A is the origin then parameter $c = 0$ and parameter b is written in terms of parameter a in equation (3.80).

$$b = -aL \tag{3.80}$$

Then, the y coordinate of the minimum point is equated to the sag of the power line (equation (3.81)).

$$\frac{-b^2}{4a} = -S = \frac{wL^2}{8H} \tag{3.81}$$

An expression for parameter a is then formed after substituting (3.80) in (3.81) (shown in equation (3.82)).

$$a = \frac{w}{2H} \tag{3.82}$$

In reality, the end points may be at different heights as in figure 3.3. This can also be modeled using equation (3.79) provided that the length L of figure 3.3 is known. L can be calculated if x_1 and x_2 are known.

For a known ζ value, sags S_1 and S_2 (of figure 3.3) can be determined from equation (3.77), as shown below in equations (3.83) and (3.84).

$$S_1 = \frac{x_1^2}{2\zeta} \tag{3.83}$$

$$S_2 = \frac{x_2^2}{2\zeta} \tag{3.84}$$

The total span s of figure 3.3 is also equal to the sum of the two distances from x_1 and x_2.

$$s = x_2 + x_1 \tag{3.85}$$

Equation (3.86) is formed for the height difference between the two end points denoted by h, and can be solved simultaneously with the above to yield (3.87).

$$S_2 - S_1 = \frac{s}{2\zeta}(x_2 - x_1) = h \tag{3.86}$$

$$\begin{aligned} x_1 &= \frac{s}{2} - \frac{s}{2\zeta} \\ x_2 &= \frac{s}{2} + \frac{s}{2\zeta} \end{aligned} \tag{3.87}$$

Then, length L can be obtained as shown in equation (3.88). If point C is at a lower height than A, h will be negative and L longer than the span s.

$$L = 2x_1 \tag{3.88}$$

Applying the value of L obtained from the above and the known ζ value in equation (3.79), a model for the power line with end points at different heights can be obtained.

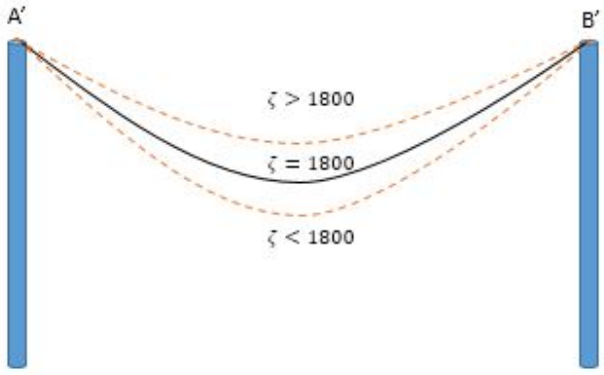

Figure 3.5: Effect of ζ on the line.

The catenary constant ζ varies as temperature changes. This is because, as temperature increases the conductor expands, causing an increase in its length thus increasing sag and decreasing the tension. Typical sag-temperature data for heavy loading is tabulated in Appendix C [7]. The data was converted to SI units and plotted against temperature in figure 3.6. As it can be seen, larger spans have greater sag, and the sag increases with temperature. The sags in the different spans show a similar trend with respect to a change in temperature.

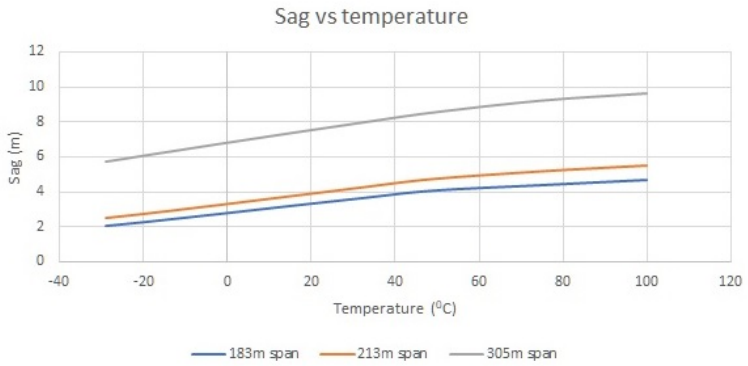

Figure 3.6: Sag against temperature.

Using equation (3.78), the catenary constant ζ was found and plotted against temperature in figure 3.7. The catenary constants for the different spans are similar, and decrease as temperature increases. It must be noted that the catenary constant is related to the sag parameter of a catenary and changes with temperature.

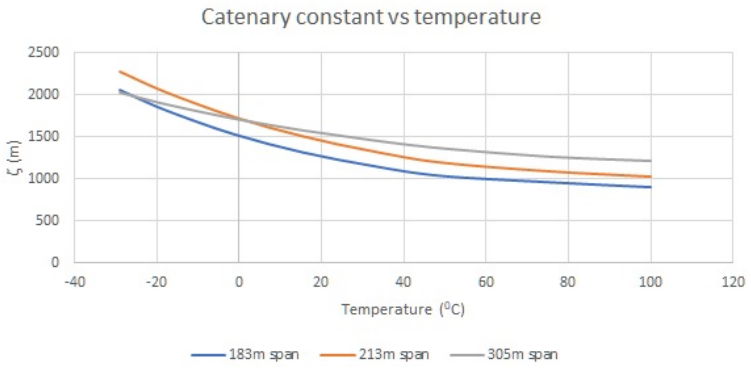

Figure 3.7: ζ against temperature.

It is more convenient to use ζ^{-1}, since it reduces the complexity of the covariance equations later on. An average for the zeta values for the three curves in figure 3.7 was calculated, inverted to find ζ^{-1} and plotted against temperature in figure 3.8. Since the plot is almost linear, the spread of the data is found by dividing the difference between maximum and minimum values of ζ^{-1} by two. Thus, an approximation for the standard deviation was obtained as $0.000252m^{-1}$.

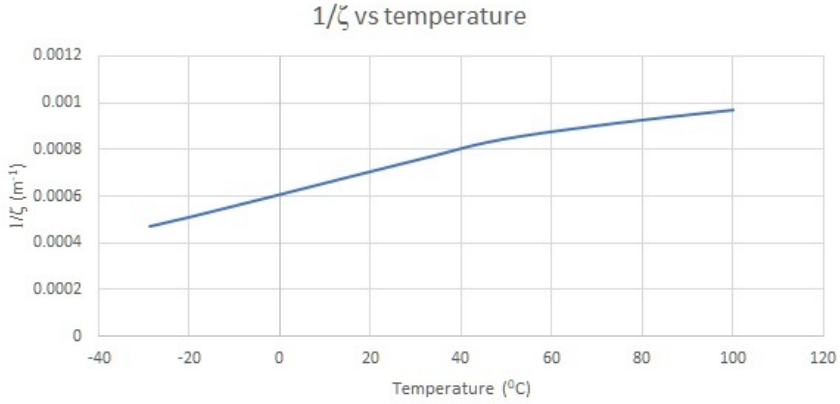

Figure 3.8: $\dfrac{1}{\zeta}$ against temperature.

In the reference frame the yaw angle, which is denoted as ψ, is zero, as in this frame, the x-axis lies in the direction of the power line. However, there may be some uncertainty in the yaw angle due to mechanization errors

between the line, the robot grippers and the robot platform. This error will be approximated to one degree when both grippers are attached and five degrees when only one gripper is attached.

The roll angle ϕ, is zero under perfect conditions. However, due to various effects such as wind, vibrations and the movement of joints, the robot tends to oscillate.

There are two types of oscillations, that of the robot and that of the line. The line usually oscillates due to wind, with two main types of effects, namely; the aeolian effect and wake induced vibrations [22].

According to ESKOM guidelines [22] the line vibrational peak to peak amplitude due to aeolian effects varies between 0.01 to 1 conductor diameter with a frequency between 3 and 150 Hz. However, wake induced vibrations can cause an oscillation amplitude 0.5 to 80 conductor diameter peak to peak at frequencies between 0.15-10 Hz. Aeolian vibrations are usually in the vertical direction and is negligible when compared to wake induced vibrations.

Using the typical value for the catenary constant of 1800 m given in ESKOM literature and equation (3.78), the maximum sag for the cable used for wind testing is determined to be 14 m. Figure 3.9 shows a section at the midspan of the cable swinging to its maximum position of 0.9 m (An Aluminium ACSR wire with a cross sectional area of 240 mm^2 has a diameter of about 23 mm). Angle α is calculated to be 3.7 degrees and the roll of the robot about the gripper mean position β is assumed to be 5 degrees. The roll standard deviation will therefore be assumed to be 10 degrees.

3.7 Kalman Filtering with State Constraints

The robot is constrained to lie on the power line at all times. This fact can be used to provide better estimates by constraining the states for this tracking application. Many methods of integrating constraints in Kalman filters have been described [24]. Model reduction, where the number of states are reduced based on the constraint, and perfect measurement, where the constraints are used as measurements, are two such methods. Equality constraints such as in equation (3.89) satisfy the requirements of this system and will be used with the perfect measurements method.

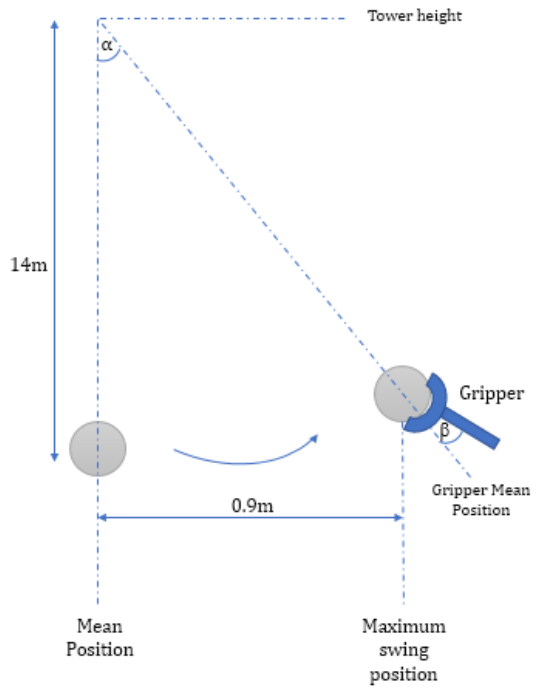

Figure 3.9: Section through cable at midspan showing the cable swinging from its mean position to its maximum position

$$\boldsymbol{D x_k = d} \tag{3.89}$$

Applying the method above, requires the measurement equation to be augmented. This is shown in equation (3.90), where C is the measurement matrix, D the constraint matrix and d the constraint. [24]

$$\begin{bmatrix} \boldsymbol{y_k} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{D} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v_k} \\ \boldsymbol{0} \end{bmatrix} \tag{3.90}$$

Constraints may be nonlinear, with the form of equation (3.91)

$$\boldsymbol{g(x_k) = b} \tag{3.91}$$

Using a Taylor series approximation, the above can be written in a form equivalent to equation (3.89). Equations (3.92) and (3.93) show approximations of matrix $\boldsymbol{D}$ and vector $\boldsymbol{d}$.

$$\boldsymbol{D} = \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) \tag{3.92}$$

$$\boldsymbol{d} = \boldsymbol{b} - \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{k}})\hat{\boldsymbol{x}}_{\boldsymbol{k}} \tag{3.93}$$

The constraint in equation (3.89) is also known as a hard constraint, i.e. it needs to be exactly satisfied. However, in the real world, there might be uncertainty in the constraints. Constraints which only need to be approximately satisfied are called soft constraints. Such constraints have a nominal value which can vary. Therefore soft constraints can be implemented as a mean with some noise associated to it, i.e. a perfect measurement with Gaussian noise. If the noise is non additive, the covariance can be found using equation (3.16) [24].

The robot can be constrained to the line in several ways. The velocity of the robot along the x and y reference axes can be constrained based on the value of known yaw angles. This is shown in equation (3.94).

$$\frac{v_y}{v_x} = tan\psi \implies v_y - v_x tan\psi = 0 \tag{3.94}$$

Since the reference x axis lies along the line the yaw angle is zero. Thus, the above contrains the y axis velocity preventing large values from being estimated, and therefore ensuring the robot lies on the line. The variance of the above equation denoted by $\sigma^2_{d_1}$ can be found using equation (3.16) and is shown below [25].

$$\sigma^2_{d_1} = (-\hat{v}_x)^2\sigma^2_{\psi} + \sigma^2_{v_y} \tag{3.95}$$

Similarly, the position of the robot in the horizontal plane of the reference frame can also be constrained using the yaw angle (equation (3.96)) with variance of the constraint $\sigma^2_{d_2}$ given by equation (3.97). The gradient of the line in the horizontal plane is equal to $\tan\psi$, which is also equal to the velocity in the y direction divided by the velocity in the x direction.

$$\frac{y}{x} = tan\psi \implies y - xtan\psi = 0 \tag{3.96}$$

$$\sigma_{d_2}^2 = \hat{x}^2\sigma_\psi^2 + \sigma_y^2 \tag{3.97}$$

Since the shape of the line can be deduced from its known catenary constant, the vertical position of the robot can be restrained based on the distance moved along the line, assuming a parabolic shape as discussed in the previous section. This constraint is given in equation (3.98) and the variance associated to this constraint $\sigma_{d_3}^2$ is given in equation (3.99).

$$z = \frac{x^2}{2\zeta} - \frac{Lx}{2\zeta} \implies z - \frac{x^2}{2\zeta} + \frac{Lx}{2\zeta} = 0 \tag{3.98}$$

$$\sigma_{d_3}^2 = \left(\frac{-\hat{x}^2 + L\hat{x}}{2}\right)\sigma_{\zeta^{-1}}^2 + \left(-\frac{\hat{x}}{\zeta} + \frac{L}{2\zeta}\right)^2\sigma_x^2 \tag{3.99}$$

Constraint equations are also formed for the quaternions. The yaw angle is constrained to lie along the line direction and the roll angle is constrained to its mean value, zero, with associated noise to cater for swinging of the line and robot. The relationship between quaternions and Euler angles is used to form the constraint equations. The roll angle is constrained as in equation (3.100) with variance $\sigma_{d_4}^2$ given in equation (3.101). The yaw angle is constrained as in equation (3.102), with variance given by equation (3.103).

$$\begin{gathered}\frac{2(q_1q_4 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)} = tan\phi \\ \implies 2q_1q_4 + 2q_2q_3 - tan\phi + 2tan\phi(q_1^2 + q_2^2) = 0 = d_4\end{gathered} \tag{3.100}$$

$$\sigma_{d_4}^2 = (-1 + 2(\hat{q}_1^2 + \hat{q}_2^2))^2\sigma_\phi^2 + 4\hat{q}_1^2\sigma_{q_4}^2 + 4\hat{q}_4^2\sigma_{q_1}^2 + 4\hat{q}_2^2\sigma_{q_3}^2 + 4\hat{q}_3^2\sigma_{q_2}^2 \tag{3.101}$$

$$\begin{gathered}\frac{2(q_3q_4 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)} = tan\psi \\ \implies 2q_3q_4 + 2q_1q_2 - tan\psi + 2tan\psi(q_2^2 + q_3^2) = 0 = d_5\end{gathered} \tag{3.102}$$

$$\sigma_{d_5}^2 = (-1 + 2(\hat{q}_2^2 + \hat{q}_3^2))^2\sigma_\psi^2 + 4\hat{q}_3^2\sigma_{q_4}^2 + 4\hat{q}_2^2\sigma_{q_1}^2 + 4\hat{q}_4^2\sigma_{q_3}^2 + 4\hat{q}_1^2\sigma_{q_2}^2 \quad (3.103)$$

The yaw angle, roll angle and catenary constant are modeled as variables with zero mean and noise. The yaw angle has zero mean with an assumed noise standard deviation, σ_ψ, of one degree, and the roll angle has zero mean with an assumed noise standard deviation, σ_ϕ, of ten degrees. The ζ^{-1} value has a mean of $1/1800\ m^{-1}$, and a standard deviation of (0.000252) m^{-1}. These have been discussed in detail in the previous section.

3.8 Kalman Filter Update

Updating the Kalman filter equations according to table 2.1, requires one to form the measurement matrix. The measurements are the GPS horizontal position projected onto the line, the GPS vertical position and the GPS ground velocity. Thus, using equations (3.73), (3.58) and (3.59), matrix $\boldsymbol{H}$ is formed (equation (3.104)).

$$\boldsymbol{H} = \begin{bmatrix} \mathbf{0_{1\times 7}} & \begin{bmatrix} cos\psi & 0 & 0 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \\ \mathbf{0_{1\times 7}} & \begin{bmatrix} sin\psi & 0 & 0 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \\ \mathbf{0_{1\times 7}} & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \\ \mathbf{0_{1\times 7}} & \mathbf{0_{1\times 3}} & \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} & \mathbf{0_{1\times 3}} \end{bmatrix} \quad (3.104)$$

Similarly, for the constraint equations (3.94) to (3.102), matrix $\boldsymbol{D}$ is formed (equation (3.105), where $\psi = 0$ and $\phi = 0$). Matrix $\boldsymbol{C}$ in table 2.1 used for the update is formed for $\boldsymbol{H}$ and $\boldsymbol{D}$ as in equation (3.90).

$$\boldsymbol{D} = \begin{bmatrix} \mathbf{0_{1\times 4}} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} & \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} & \mathbf{0_{1\times 3}} \\ \mathbf{0_{1\times 4}} & \mathbf{0_{1\times 3}} & \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \\ \mathbf{0_{1\times 4}} & \mathbf{0_{1\times 3}} & \begin{bmatrix} -\frac{\hat{x}}{\zeta} + \frac{L}{2\zeta} & 0 & 1 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \\ \begin{bmatrix} 2q_4 & 2q_3 & 2q_2 & 2q_1 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \\ \begin{bmatrix} 2q_2 & 2q_1 & 2q_4 & 2q_3 \end{bmatrix} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} & \mathbf{0_{1\times 3}} \end{bmatrix} \quad (3.105)$$

Following equation (3.93), the measurements for the constraints are constructed as shown below in equation (3.108).

$$\boldsymbol{d} = \begin{bmatrix} 0 \\ -\dfrac{\hat{x}\hat{v}_x}{\zeta} \\ 0 \\ -\dfrac{-\hat{x}^2}{\zeta} \\ 2(\hat{q}_1\hat{q}_4 + \hat{q}_2\hat{q}_3) \\ 2(\hat{q}3\hat{q}_4 + \hat{q}_1\hat{q}_2) \end{bmatrix} \tag{3.106}$$

The covariance matrix for measurements is formed from the variances in section 1.5 and is given by equation (3.107).

$$\boldsymbol{R_H} = \begin{bmatrix} \sigma^2_{x_p} & 0 & 0 & 0 \\ 0 & \sigma^2_{y_p} & 0 & 0 \\ 0 & 0 & \sigma^2_{\dot{z}_v} & 0 \\ 0 & 0 & 0 & \sigma^2_{v_m} \end{bmatrix} \tag{3.107}$$

Equation (3.108) contains the variances of the noise in the constraints.

$$\boldsymbol{R_d} = \begin{bmatrix} \sigma^2_{d_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma^2_{d_n} \end{bmatrix} \tag{3.108}$$

The full covariance matrix for the measurements is formed in equation (3.109) below.

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_H} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_d} \end{bmatrix} \tag{3.109}$$

The usage of quaternions to represent attitude requires it to always have a norm of unity. Therefore, after each iteration of the EKF and UKF, equation (3.110) is used to ensure this.

$$|\bar{q}| = \frac{\bar{q}}{\sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}} \tag{3.110}$$

For the ErKF, the unity norm constraint is used to calculate the scalar element of the error quaternion (q_4) [14]. This is shown in equation (3.111).

$$\delta\bar{q} = \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \sqrt{1 - \delta q_1^2 + \delta q_2^2 + \delta q_3^2} \end{bmatrix} \tag{3.111}$$

Since the error state has been used, error measurements must also be used. The chain rule (equation (3.112)) can be used to obtain the required measurement Jacobian matrix, where $h(x)$ represents the measurement equation [13]. The standard EKF measurement Jacobian matrix is calculated and multiplied by equation (3.113), producing the ErKF measurement Jacobian matrix.

$$\frac{\partial h}{\partial \boldsymbol{\delta x}} = \frac{\partial h}{\partial \boldsymbol{x}} . \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\delta x}} \tag{3.112}$$

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\delta x}} = \begin{bmatrix} \Lambda(\hat{\bar{q}}) & \mathbf{0_{4\times 3}} & \mathbf{0_{4\times 3}} & \mathbf{0_{4\times 3}} & \mathbf{0_{4\times 3}} \\ \mathbf{0_{3\times 4}} & \boldsymbol{I_{3\times 3}} & \mathbf{0_{3\times 3}} & \mathbf{0_{3\times 3}} & \mathbf{0_{3\times 3}} \\ \mathbf{0_{3\times 4}} & \mathbf{0_{3\times 3}} & \boldsymbol{I_{3\times 3}} & \mathbf{0_{3\times 3}} & \mathbf{0_{3\times 3}} \\ \mathbf{0_{3\times 4}} & \mathbf{0_{3\times 3}} & \mathbf{0_{3\times 3}} & \boldsymbol{I_{3\times 3}} & \mathbf{0_{3\times 3}} \\ \mathbf{0_{3\times 4}} & \mathbf{0_{3\times 3}} & \mathbf{0_{3\times 3}} & \mathbf{0_{3\times 3}} & \boldsymbol{I_{3\times 3}} \end{bmatrix} \tag{3.113}$$

As matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ have been defined, the observabilitv matrix can be constructed. This was done using the obsv() function in MATLAB. The observability matrix had full rank when all measurements are used and hence is fully observable. However during GPS signal outages, when the system is allowed to run with reduced states,the measurement matrix $\boldsymbol{C}$ is formed using only constraint equations. The observability matrix for this case is 14, implying that there two unobservabe states.

3.9 The UKF with Non-Additive Noise

The UKF equations from chapter 2 need to be modified to handle non-additive noise. Thus an augmented state vector and error covariance are defined [20].

The state vector is augmented with noise vectors (equation (3.114)).

$$\hat{\boldsymbol{x}}_k^a = \begin{bmatrix} \hat{\boldsymbol{x}}_k \\ \boldsymbol{\eta}_k \\ \boldsymbol{v}_k \end{bmatrix} \tag{3.114}$$

Similarly, this requires the error covariance matrix to be augmented (equation (3.115))

$$\boldsymbol{P}_k^a = \begin{bmatrix} \boldsymbol{P}_k & 0 & 0 \\ 0 & \boldsymbol{Q}_k & 0 \\ 0 & 0 & \boldsymbol{R}_k \end{bmatrix} \tag{3.115}$$

Due to the above modifications, equations (2.23) and (2.26) need to be modified as the Q and R terms are cancelled since they are already taken into account in the augmented covariance (equations (3.116) and (3.117)). The noise terms are directly added to the state and measurement equations, as they are now defined as states. The overall structure remains the same as discussed in chapter 2.

$$\boldsymbol{P}_{k|k-1} = \sum_{i=0}^{2L} \boldsymbol{\eta_i^c} (\boldsymbol{\chi}_{k|k-1}^i - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{\chi}_{k|k-1}^i - \hat{\boldsymbol{x}}_{k|k-1})^T \tag{3.116}$$

$$\boldsymbol{P}_k^{yy} = \sum_{i=0}^{2L} \boldsymbol{\eta_i^c} (\boldsymbol{\Psi}_{k|k-1}^i - \hat{\boldsymbol{y}}_{k|k-1})(\boldsymbol{\Psi}_{k|k-1}^i - \hat{\boldsymbol{y}}_{k|k-1})^T \tag{3.117}$$

First, sigma points are created using mean values of the states and the error covariance according to equation (2.20).

$$\boldsymbol{\chi}_{k|k-1}^i = f(\boldsymbol{\chi}_{k-1}^i, \boldsymbol{u}_{k-1}, \boldsymbol{\eta}_{k-1}), \quad i = 0, 1, ..., 2L \tag{3.118}$$

The sigma points are propagated through the state equations producing an updated set of sigma points as in equation (3.118) above. Noises (which are now part of the states) are directly added to the propagation equations as shown in equations (3.119), (3.120) and (3.121). Equation (3.25) is used to propagate position.

$$\bar{q}_{k+1} = (\boldsymbol{I_{4\times 4}} + \frac{1}{2}[\boldsymbol{(\omega_m - b_\omega - \eta_\omega)}\times]T)\bar{q}_k \tag{3.119}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + (\boldsymbol{R}(\bar{q})^T \boldsymbol{a_m} - \boldsymbol{b_a} - \boldsymbol{\eta_a} + \boldsymbol{g})T \tag{3.120}$$

$$\boldsymbol{b}_{k+1} = \boldsymbol{b}_k - \boldsymbol{\eta_b} \tag{3.121}$$

The propagated sigma points are then used to calculate an updated mean of the states and covariance using appropriate weighting factors according to equations (2.22) and (3.116).

Similarly, the sigma points are passed through the nonlinear measurement equations ((3.59) and (3.73)) and constraint equations ((3.94), (3.96), (3.98), (3.100) and (3.102)), with noises directly applied to the equations as shown below.

$$\boldsymbol{\Psi}^i_{k|k-1} = h(\boldsymbol{\chi}^i_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{v}_{k-1}), \quad i = 0, 1, ..., 2L \tag{3.122}$$

The measurement covariance is found using equation (3.117) and the rest of the update step uses equations (2.270 to (2.30) to implement the UKF.

3.10 Summary

In this chapter, a mathematical model of the power line inspection robot is derived. Here, the robot is treated as a point mass moving along the power line.

First, the state vector is defined to estimate the robot's attitude (quaternion vector), the gyroscope sensor bias, the robot's position, the robot's velocity, and the accelerometer sensor bias. The continuous-time state propagation model is then constructed using the differential equations for the quaternion attitude kinematics, the gyroscope sensor model (with bias and sensor noise), the position kinematics, the velocity kinematics and the accelerometer sensor model (with bias and sensor noise).

Also, the linearized system matrix is derived using the Jacobian of the non-linear state equations, and the process noise covariance matrix is derived from the system model as a function of the IMU sensor noise variances. Next, the continuous-time system model and the process noise covariance matrix are discretized so that they can be used in the discrete-time Kalman

filter. Closed-form difference equations are derived to propagate both the system state and the state error covariance matrix. The above is performed for both the extended Kalman filter and the error state Kalman filter.

The measurement model is then derived. The measurements include the GPS position, the GPS velocity and state constraints imposed by the shape and direction of the power line. The state constraints are treated as virtual measurements. For the GPS position and velocity measurements, output equations and measurement noise their corresponding variances are derived. The geometry of the power line is then modeled so that the robot can be constrained to he power line. "Measurement noise" is added to each of the constraint equations due to uncertainties in the constraints. Linearized equations for the measurements are derived using Jacobians and the measurement covariance matrix is constructed. Finally, a version of the UKF with non-additive noise is discussed.

In the next chapter, the three filters will first be run in a simulation environment before practical testing.

Chapter 4

Simulations And Results

4.1 Simulation Model

The system needs to be simulated before attempting to test it. This is done using MATLAB, where models for the accelerometer, gyroscope and GPS signals are created. Firstly, equations for the X-Y-Z accelerations in the inertial frame and the pitch angular rate are formed using the right hand rule convention for directions.

For the simulation, the robot is approximated as a point mass moving along a catenary, approximated by a quadratic function. Both the yaw and the roll angles areassumed to be zero in the reference frame. The pitch angle changes as the robot moves along the span. The velocity tangential to the power line, v, is also assumed to be constant at 1 m/s (shown in figure 4.1).

The tangential velocity is resolved into the inertial axes (equation (4.1)) and differentiated to give the inertial accelerations (equation (4.2)), so that the system can be modeled in the inertial frame. It should be noted that angle ψ is constant.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v\cos\theta\cos\psi \\ v\cos\theta\sin\psi \\ v\sin\psi \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} -v\sin\theta\dot{\theta}\cos\psi \\ -v\sin\theta\dot{\theta}\sin\psi \\ -v\cos\theta\dot{\theta} \end{bmatrix} \tag{4.2}$$

Next, the reference frame pitch angle is formed from the power line's shape. It is assumed that the robot's pitch direction is tangential to the line at any point. Thus, the pitch angle is calculated from the derivative of the power line ($ax_l^2 + bx_l + c$, where $c = 0$) as in equation (4.3).

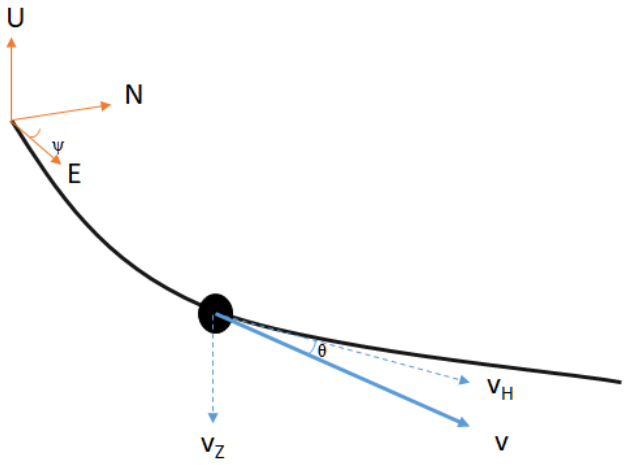

Figure 4.1: Black line: power line seqment, Orange axes: inertial frame with U(Up), N(North) and E(East), ψ is the yaw angle lying in the horizontal plane, θ is the pitch angle lying in the vertical frame, v is the velocity which is tangential to the power line.

$$\theta = \arctan(2ax_l + b) \tag{4.3}$$

The above is then differentiated to give the pitch rate (equation 4.4).

$$\dot{\theta} = \frac{-2a\dot{x}_l}{(2ax_l + b)^2 + 1} = \frac{-2av\cos\theta}{(2ax_l + b)^2 + 1} \tag{4.4}$$

The accelerations in equation (4.2) are integrated twice using the MATLAB integrator block, producing the inertial velocity and position signals. In the integrator blocks the positions are initialized as zero and velocities are initialized with calculated values as per equation (4.1), where the known β and an initial value of θ are used. The horizontal component of the resultant of the inertial velocities in the horizontal plane is calculated so that it canbe used as GPS velocity measurement when noise is added. The positional inertial signals are directly used as GPS position measurements after Gaussian noise is added to them.

White noise is simulated using the band limited white noise block in Simulink. Typically, white noise has a PSD (Power Spectral Density) with total energy of infinity and correlation time zero. Thus, the best theoretical model is band limited noise. Simulink uses a random sequence with a correlation time less than the shortest time constant to produce an approximation. To obtain good results the correlation time, t_c, is specified as equation (4.5), where f_{max} is the maximum bandwidth of the system. The noise intensity is defined using the height of the PSD. To obtain the correct noise intensity from the continuous PSD to the discrete noise covariance, the covariance is scaled by $1/t_c$. The noise power is therefore defined as the covariance multiplied by the correlation time [26].

$$t_c = \frac{2\pi}{100 f_{max}} \tag{4.5}$$

The accelerometer signals are generated by rotating the acceleration vector of equation (4.2) plus the gravity vector to the body frame. This is achieved by first rotating about the z axis by angle β, and then the y axis by angle θ (roll angle is zero). The gyroscope's x and z signals are assumed to be zero and the gyroscope y signal is equivalent to $\dot{\theta}$ (equation (4.4)). Bias noise and white noise are also added to the gyroscope signals. All measurements are passed to each of the three filters (EKF, UKF and ErKF) which are coded in MATLAB function blocks (Appendix G). Each noise signal is then added to its respective state signal, and passed through a zero order hold block to digitize the system.

4.2 Simulation Results

Using the model described in chapter 3, the three filters are simulated. The span of the power line used is 200 m with a catenary constant of 1800 m. The robot is allowed to move at a constant velocity of 2 m/s along the line. The initial quaternion (representing a rotation from the body frame to the reference frame) values are calculated using the known initial yaw angle, roll angle (both zero) and the initial pitch angle, obtained from the power line known parameters ($\theta_i = -\arctan(b)$). Thus, a small error (0.001 radians) is used for the initial error covariance of the quaternions. The initial position is also known with centimeter accuracy, and the initial position variance is set at $(0.01)^2$ (m^2).

Noise variance values used for simulations are obtained from the MPU6050 IMU datasheet. The datasheet specifies gyroscope noise at 0.05 degrees per second and accelerometer noise at 0.004 meters per second square. However to account for vibrations, the noise variance values used for the gyroscope and accelerometer are increased to $0.004^2(rad/s)^2$ and $0.04^2(m/s^2)^2$ respectively. Drift noise for the gyroscope is assumed to be $0.0001^2(rad/s)^2$. It is to be noted that these noises are continuous time noises and are digitized by passing the signals through a zero order hold block. The process noise covariance matrix is formed using the above variance values. The sampling frequency used is allowed to run with reduced measurement equations while the GPS data was not available.

To form the measurement covariance matrix R according to equation (3.109), the standard deviations required are the GPS position variance, (the horizontal positional standard deviation is set to four meters, the vertical positional standard deviation is set to 20 meters and both HDOP and VDOP were assumed to be unity), the GPS velocity (the GPS elocity standard deviation is set to 0.1 meters per second, the catenary constant (the catenary constant standard deviation is set to 0.000252 m), the roll angle (the roll angle standard deviation is set to 0.174 radian, i.e. 10 degrees standard deviation as discussed in the previous chapter) and yaw angle (the yaw angle standard deviation is set to 0.0174 radian, i.e. one degree standard deviation assuming both grippers grip the power line). The GPS measurement variances are set according to the GTPA013 datasheet.

4.2.1 Euler Angles

The estimated quaternions are converted to Euler angles and plotted against time for each filter as shown below.

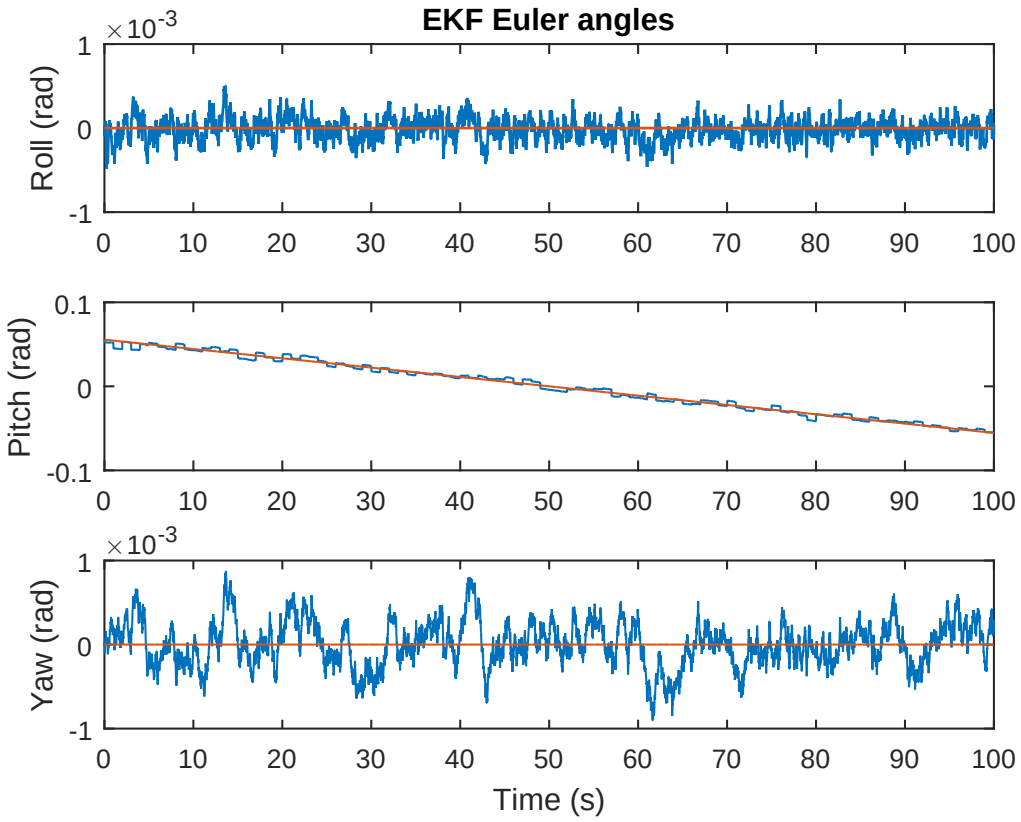

Figure 4.2: Simulation of Euler angles for the EKF - EKF(Blue), True Value (Orange).

The EKF does well in estimating the Euler angles. The yaw and roll angles are close to zero at all times due to the constraints applied to the filter. The estimated pitch angle follows the true value with an uncertainty standard deviation of 0.02 radians (figure D.4).

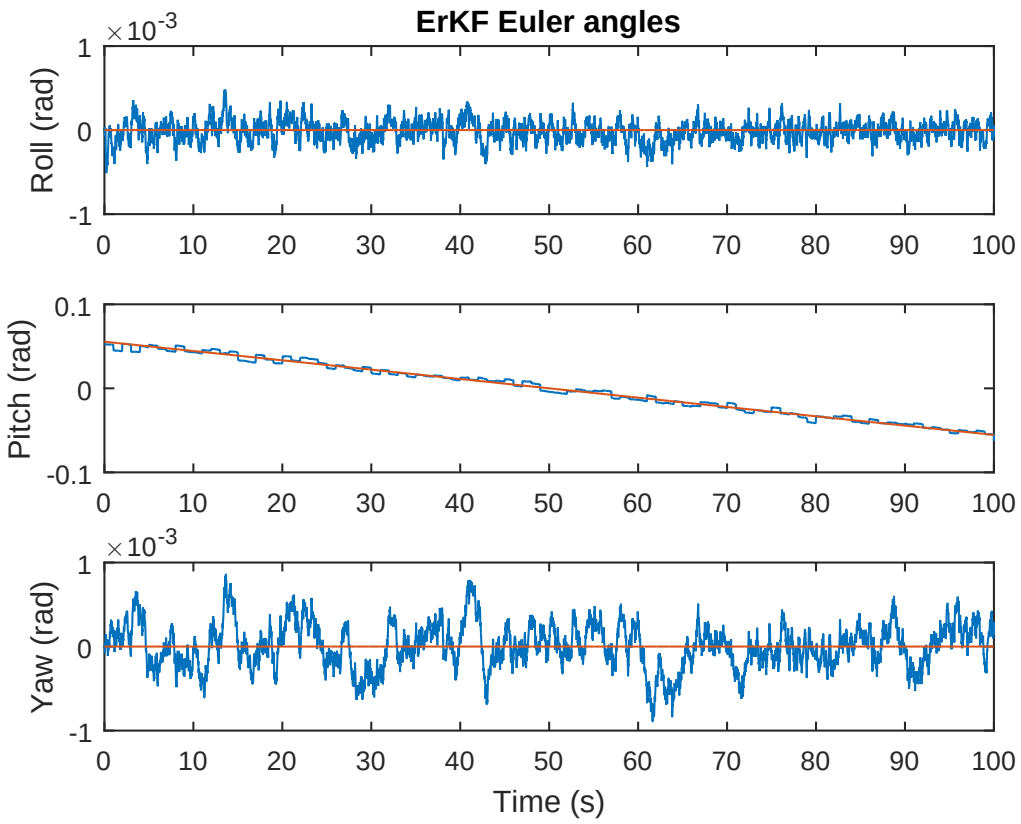

Figure 4.3: Simulation of Euler angles for the ErKF - ErKF(Blue), True Value (Orange).

The results of ErKF is very similar to the result of the EKF, which is to be expected, since the filters are equivalent. Both the EKF and ErKF have similar error bounds, as shown in Appendix D.

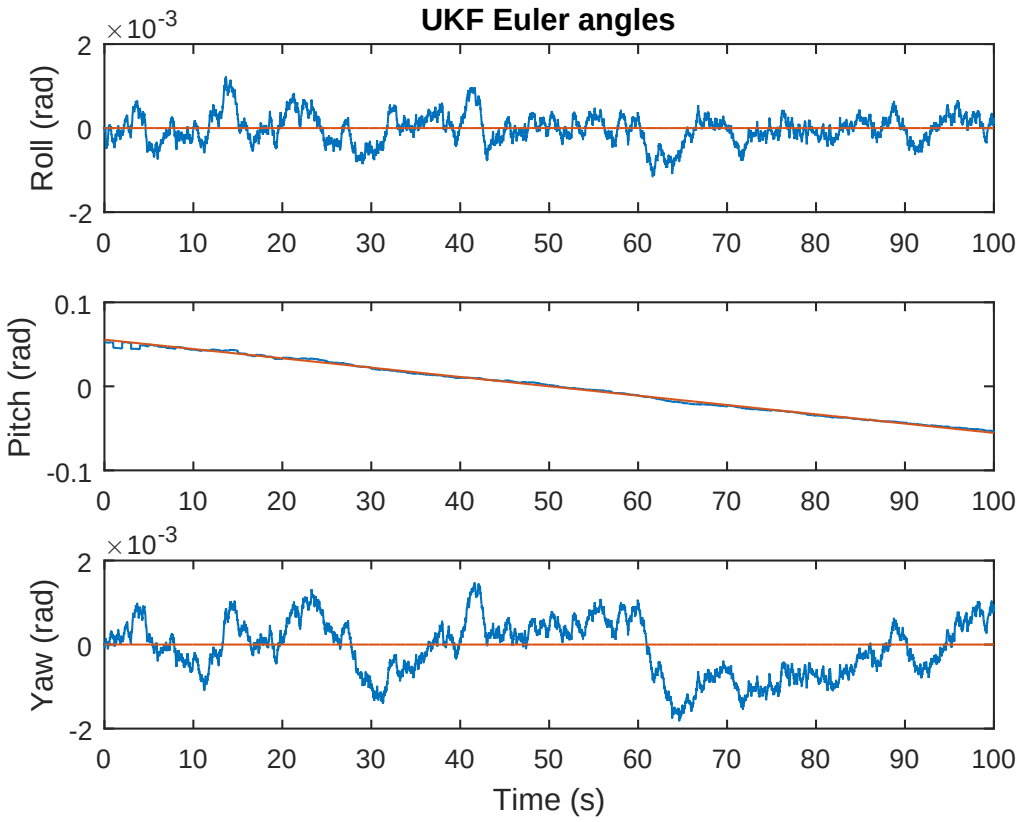

Figure 4.4: Simulation of Euler angles for the UKF - UKF(Blue), True Value (Orange).

The scaling parameters for the UKF are set as follows; $\alpha = 1$, $\beta = 2$ and $\kappa = 0$. Both α and κ are used to set the spread of the data about the mean. Here, only α defines the spread of the sigma points (κ is set to zero), from which λ is calculated. The parameter $\beta = 2$ is optimal for Gaussian distributions.

The UKF tracks the pitch angle with comparable accuracy with a 1σ error of 0.02 radian as shown in figure D.12. Differences between the two filters are expected, as the EKF uses computed Jacobians to linearize the system about the a priori estimate and the UKF propagates a sample of points chosen around the a priori estimate to obtain an updated mean while using the new spread of sigma points to obtain the new covariance.

As it can be seen in the figures, all three filters tend to converge towards similar results.

4.2.2 Position

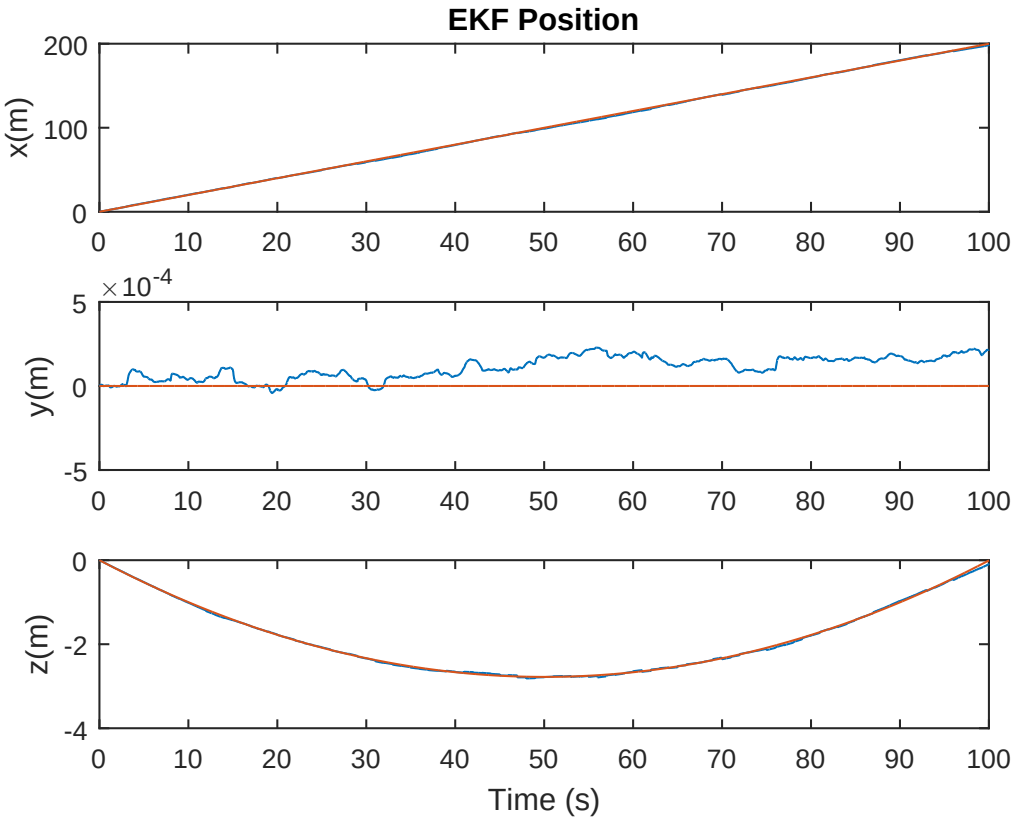

Figure 4.5: Simulation of positions for the EKF - EKF(Blue), True Value (Orange).

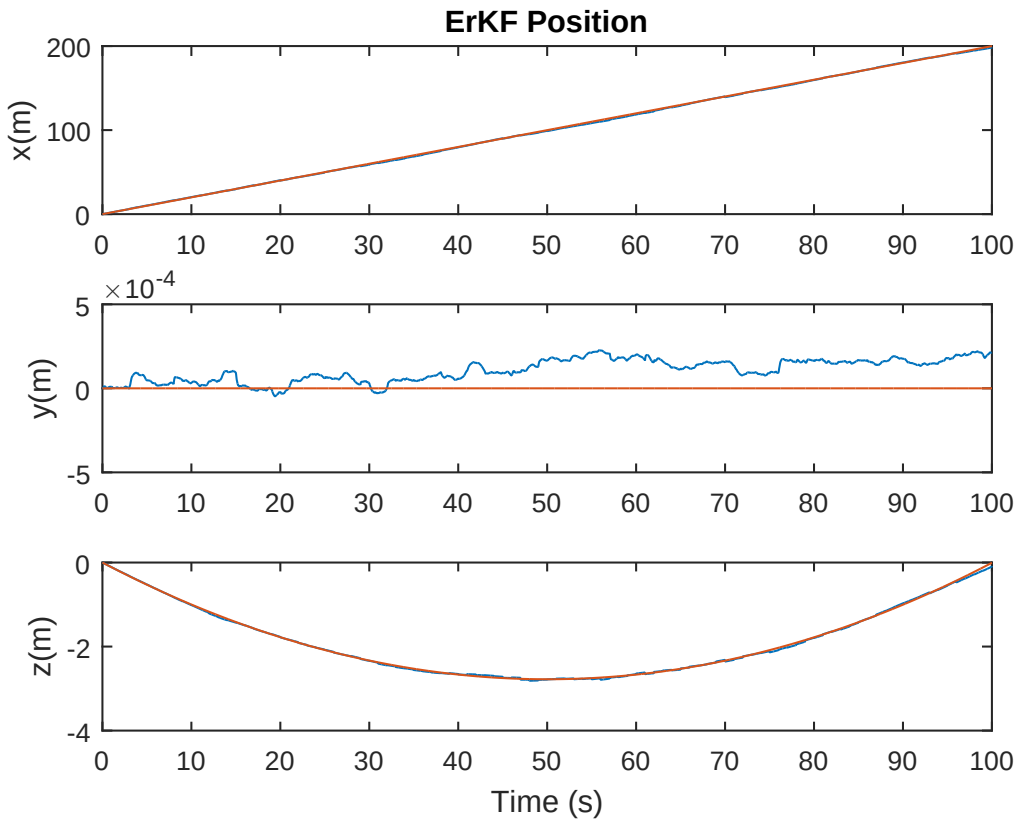

Figure 4.6: Simulation of positions for the ErKF - ErKF(Blue), True Value (Orange).

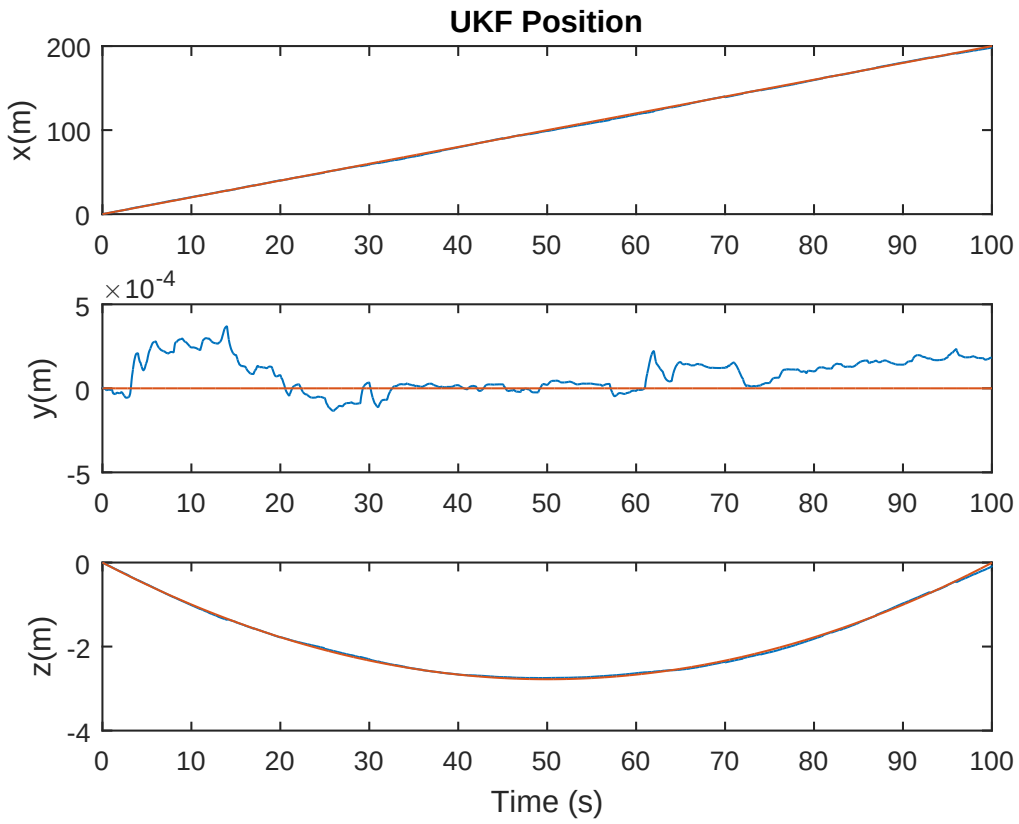

Figure 4.7: Simulation of positions for the UKF - UKF(Blue), True Value (Orange).

During GPS outage the filters run with a reduced set of measurements. This causes a growth in error covariance during GPS outage followed by a rapid fall in the error covariance as GPS signals were obtained, accounting for jagged state error covariances shown in appendix D.

All the filters produce similar results. However, the EKF and ErKF results are very similar. The x position error is zero in the beginning and increases until it settles at 44 cm for the EKF and ErKF (figures D.2 and D.6 show the 1 σ bounds for position). The UKF x position error settles at 42 cm (figure D.10 shows the 3 σ bounds for UKF estimated position).

The error variance of the z position increases as it approaches the midspan

(reaching a 1 σ, of 0.9 m at the midspan for the EKF/ErKF and 0.5 m for the UKF). This is expected, since a change in sag would mean there is more displacement in the middle. The UKF predicts lower error values for the z position.

4.2.3 Velocity

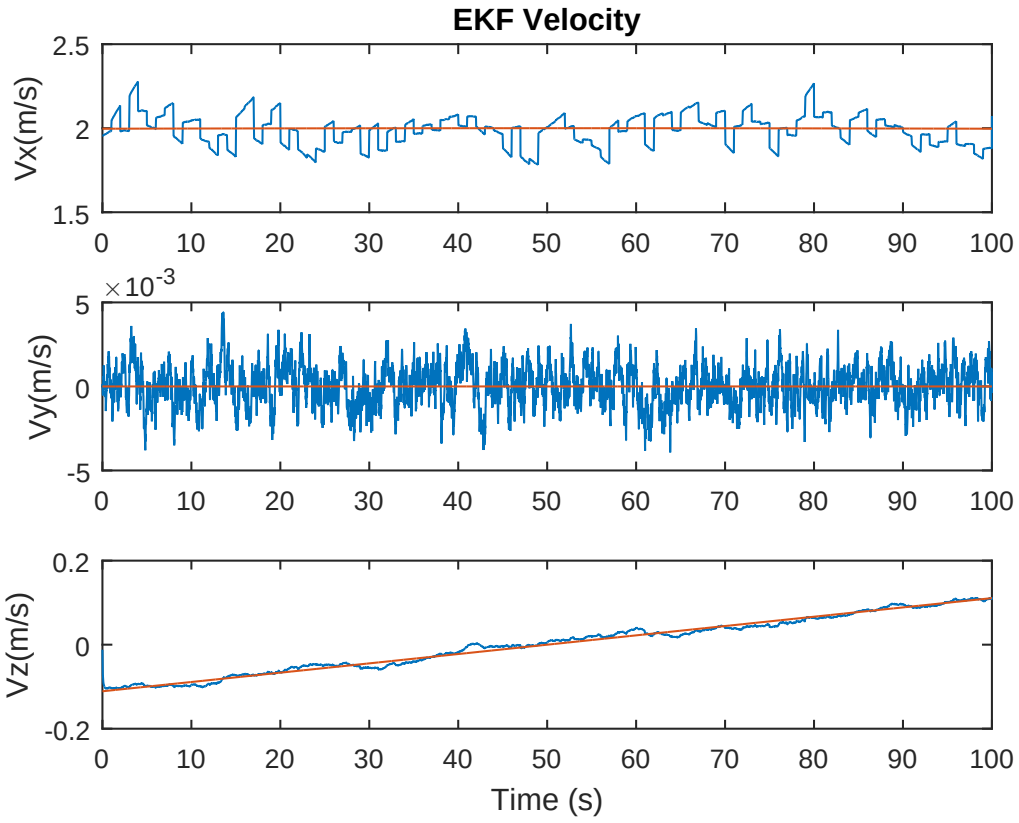

Figure 4.8: Simulation of velocity for the EKF - EKF(Blue), True Value (Orange).

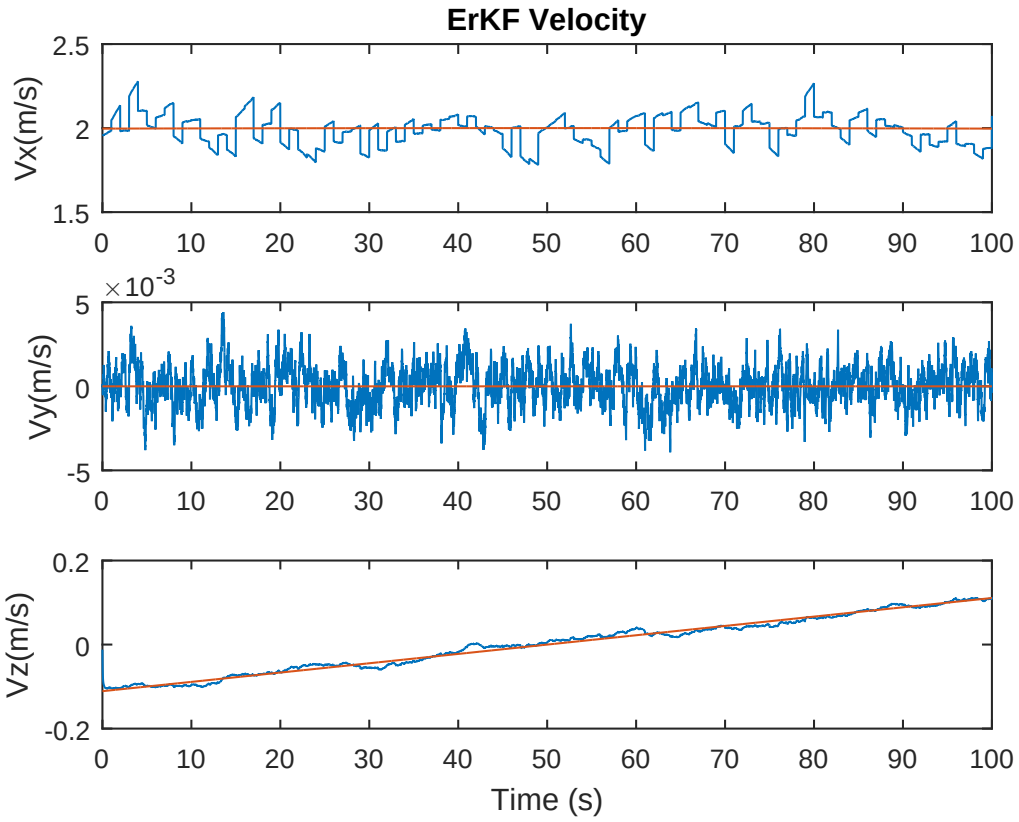

Figure 4.9: Simulation of velocity for the ErKF - ErKF(Blue), True Value (Orange).

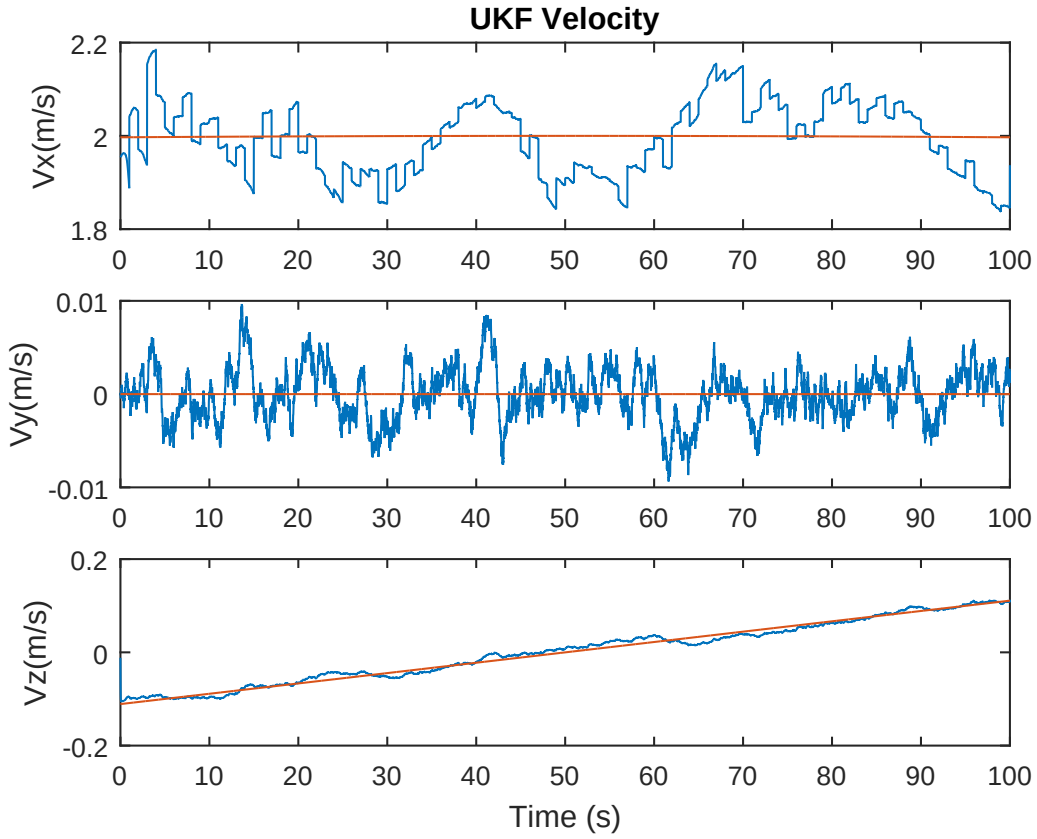

Figure 4.10: Simulation of velocity for the UKF - UKF(Blue), True Value (Orange).

The velocity estimates are propagated using the quaternion estimates. Rotation from the body to the inertial frame with 0.1 radian errors result in 1g acceleration errors, implying that small errors in the quaternions can cause big errors in the velocity during propagation. However, if the gyroscope noise is small, the quaternion error is also small, leading to better velocity estimates.

The bias simulations are shown in figures D.13 and D.14 for angular velocity and acceleration bias respectively. The EKF and ErKF showed similar results. The UKF tracked the angular velocity bias better but was worse in tacking acceleration bias.

A simulation study for the state estimation of a brachiating power line robot is also made for a robot developed by J.Patel at the University of Cape Town. This system model, the EKF simulation and UKF Simulation can be found in Appendix E.

4.3 Testing

The testing is performed by tying a rope between two known GPS coordinates, and dragging a platform along the rope by means of a pulley, to emulate a robot moving along a power line. The rope is tied in such a way that the platform's weight causes the rope to adopt the shape shown in figure 4.11 as it moved along points 1, 2 and 3 (the rope was in tension). The points through which the platform passed would still make a curved shape (shown in dotted line in the figure below) that can be fitted to a quadratic function.

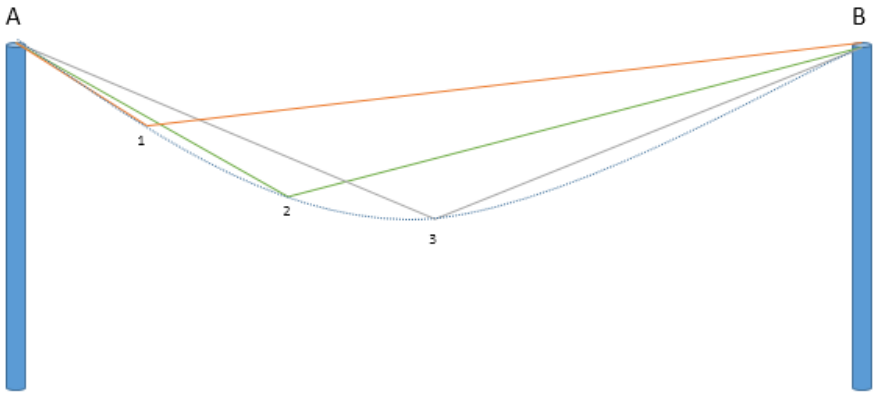

Figure 4.11: Test setup, platform moves from point 1-2-3 creating a curved path in the process.

It is possible to geometrically find the shape that the arrangement makes. A relationship for z, the vertical distance, in terms of x, the horizontal distance moved along, can be found. These are shown in figure **??**, where s is the span.

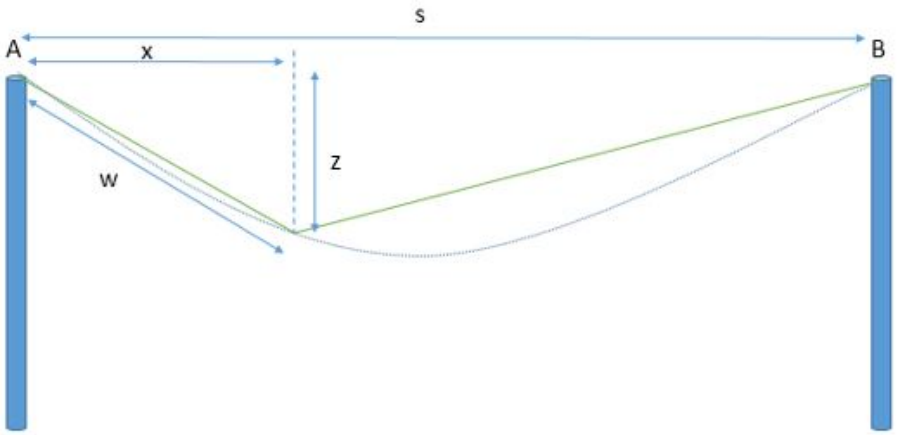

Figure 4.12: Sketch of the true shape of the line used for testing.

With known span and length of the cord l, two equations are formed using Pythagoras theorem (equations (4.6) and (4.7)).

$$z^2 = w^2 - x^2 \tag{4.6}$$

$$z^2 = (l - w)^2 - (s - x)^2 \tag{4.7}$$

Solving the simultaneous equations, the required equation (4.8) is formed as parameter w is eliminated.

$$z = \sqrt{\frac{l^2 - (s - x)^2 + x^2}{4l^2} - x^2} \tag{4.8}$$

A quadratic function is produced with known maximum sag and span according to the equations derived in the previous section. The values obtained for a function $ax^2 + bx$ are 0.0723 and -0.379 for a and b respectively (these values correspond to a catenary constant of 6.917). This quadratic function is plotted along with the true shape given by equation (4.8) in figure 4.13.

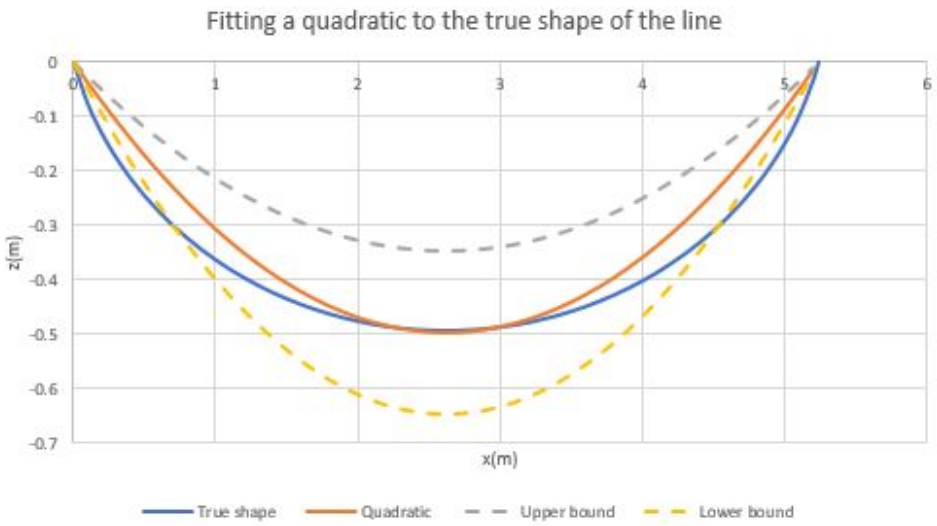

Figure 4.13: Plot of the true shape (blue), quadratic fit (red) and error bounds (dotted)

Although the quadratic shape does not exactly match the true shape, it is close enough, such that a parabolic model can be used. Error bounds are also added (dotted lines in the above figure). These error bounds can be converted to a standard deviation of $0.04m^{-1}$ in the inverse of the catenary constant.

4.3.1 Hardware

The prototype hardware used for testing was set up as shown in figure 4.14. The following are used; IMU - MPU6050, GPS - GTPA013 from Adafruit and an OpenLog module for data logging. These are all interfaced via an Arduino Mega.

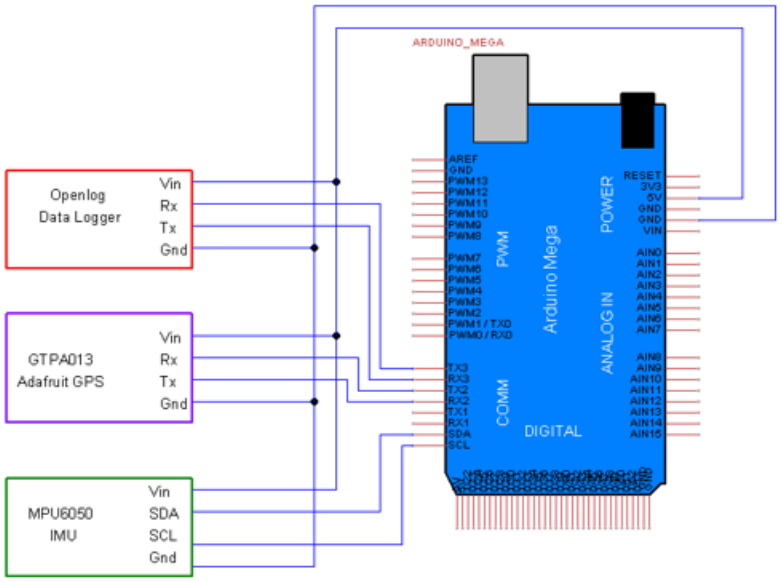

Figure 4.14: Connection of micro controller to sensors

The MPU6050 is a 6-DOF (Gyroscope and accelerometer) MEMS chip that can be sampled at 100 Hz. It uses an I2C bus for communication with 16 bit ADC hardware. The chip is set up to output acceleration values between the range $\pm 2g$ and an angular rate between the range of $\pm 250^0/s$. The smallest range is used to obtain more precise measurements. Before starting the tests, all offsets are removed from the chip by calibrating it using a spirit level. The noise variance values at 100 Hz are calculated from the chip's datasheet to be $(0.04)^2(m/s^2)^2$ for the accelerometer and $(8.73e^{-4})^2(rad/s)^2$ for the gyroscope.

The adafruit ultimate GPS breakout board uses a GTPA013 GPS module. It is capable of tracking up to 22 satellites and outputting GPS NMEA strings (GSA, RMC, VTG and GGA) at a frequency of 1 Hz. The chip is very easy to use as it communicates via USART. A parser is used to obtain the required data from the NMEA strings. The horizontal positional noise standard deviation associated with the chip is specified to be 3 m DRMS and the ground velocity noise standard deviation to be 0.1 m/s. The vertical position's standard deviation noise is not given in the datasheet and is estimated to be at about 20 m, after testing the module at a known

altitude over several experiments.

The OpenLog module is used to log all the various measurements to an SD card. It works over serial communication (USART) and can be used at a speed of 115200 bps.

An Arduino Mega is chosen for the project. It is very easy to use and and has a fast enough processor (16 MHz). Also, enough pins are available for the hardware to be used. The main reason to use the Arduino Mega is because it has libraries that can easily parse GPS data.

The GPS coordinates for the initial and final positions are obtained from Google maps. Therefore, a larger initial position error covariance was used (10 cm). Also a greater standard deviation of 5 degrees was used for the yaw angle.

All the measurements were collected in the SD Card and passed through MATLAB into the filters. Thus, the proof of concept results obtained were processed offline.

4.4 Test Results

Unlike the simulation, the velocity of the platform is not constant along the line. Also, the noise values given in datasheets could not be used, since when testing there are vibrations causing increased noise in the sensors. The noise variances for the IMU are then increased accordingly ($0.01^2(rad/s)^2$ and $0.21^2(m/s^2)^2$ for the gyroscope and accelerometer respectively).

The Euler angles are shown below. As in the simulation the EKF and ErKF estimates are similar, while the UKF produces slightly different results. The general trend of the angles is satisfactory as it follows the same trend as the simulation results.

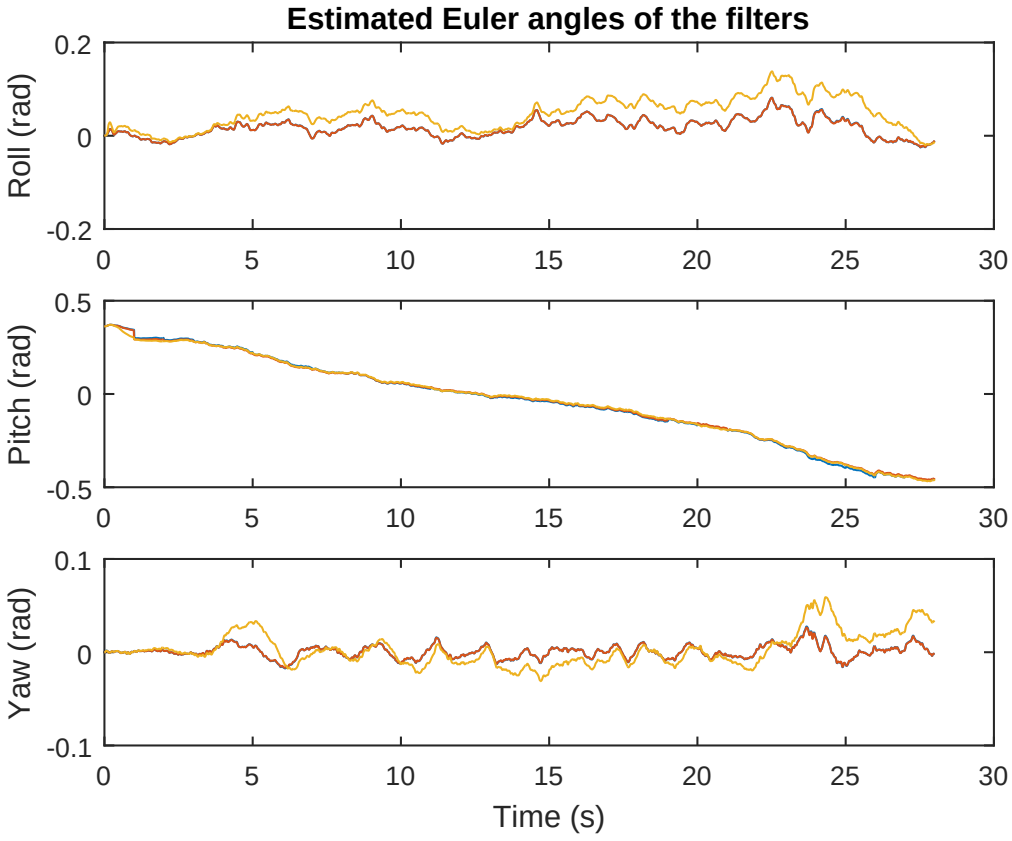

Figure 4.15: Simulation of Euler angles for the three filters - EKF(Blue), ErKF(Red) and UKF(Orange).

4.4.1 Position

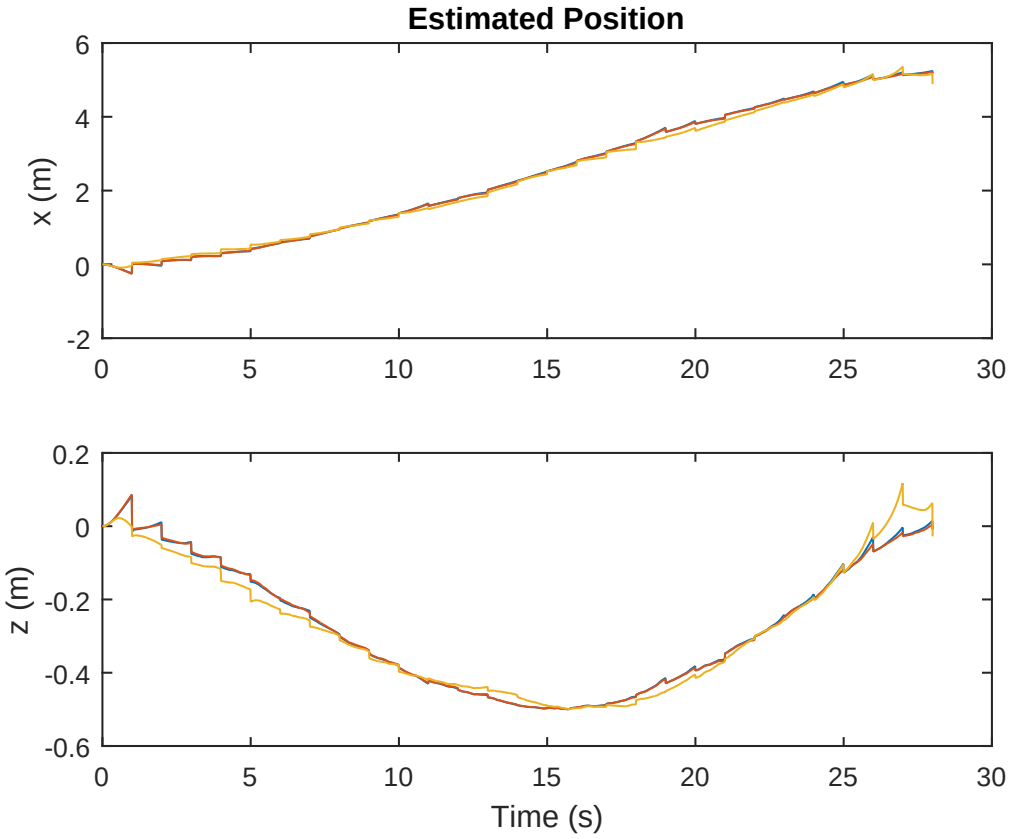

Figure 4.16: Simulation of x and z positions for the three filters - EKF(Blue), ErKF(Red) and UKF(Orange).

During the first second, as the GPS signals are not available, the x position diverges as the filter goes into dead reckoning with only the noisy accelerometer and gyroscope signals as guides. However as the GPS signal is obtained, the position estimate goes back on track. This is why in figure 4.17, the z position is greater than zero and the x position is less than zero for the filters.

The covariance for the z measurement is different from that in the simulation. The covariance in the z measurement is dictated by the error in the catenary constant and the error due to the x measurement (equation

(3.99)). If the catenary constant is very small, σ_x is the dominant part of the covariance which is the case with the test result. Otherwise, the variance associated with the catenary constant dominates the vertical displacement variance, which results in a parabolic shape with the maximum at the midpoint.

All the filters converge and the z values are similar to the true z value obtained from the geometry of the arrangement (figure 4.17). The EKF/ErKF match the true value better than the UKF, which is seen to diverge a bit at the end. The velocities cannot be compared to anything as there is no available data to compare to. Velocity graphs, shown in appendix D, agree with the simulations as v_x and v_y follow a constant trend and v_z increases from a negative value.

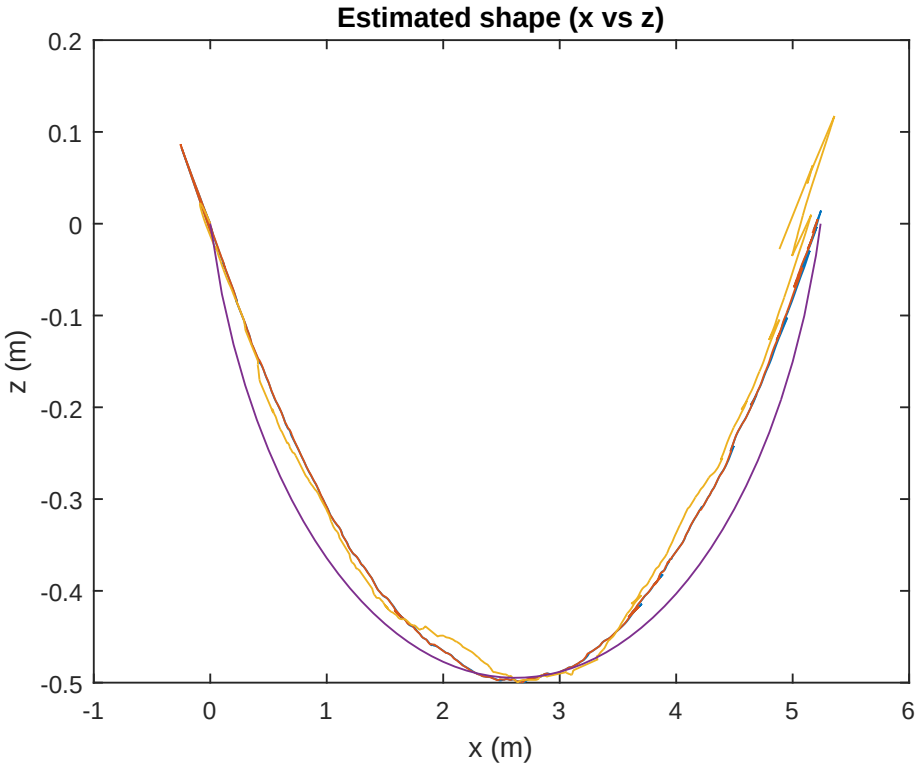

Figure 4.17: Comparison of the three filters with the true shape - EKF(Blue), ErKF(Red), UKF(Orange) and calculated value from geometry (Purle).

The Kalman filter estimates obtained are plotted against GPS measurements in figure D.28. the figure shows that the state estimates are indeed restricted to the power line.

4.5 Filter Performance

The performance of the filters in terms of the average amount of time required for one iteration, for GPS/IMU test and brachiation simulation of Appendix E, is tabulated in table 4.1. The time iss calculated using MATLAB's 'tic' and 'toc' functions and averaged over 50 loops. The numbers in the table are also rounded off. The computer used for the simulation has 8 gigabytes of RAM and a 2.2 Ghz core i7 processor. The GPS/IMU simulation took longer than the brachiation simulation to complete a loop as the system consists of more state variables, and hence higher order matrices.

Filter performance comparison		
Filter Type	GPS/IMU Test(μs)	Brachiation Simulation(μs)
EKF	87	20
ErKF	120	-
UKF	1040	105

Table 4.1: Time taken for filters to complete one iteration.

It is clear from the data in the table that the EKF is the fastest filter. However, the ErKF is not much slower. On the other hand, the UKF is ten times slower than the EKF for the GPS filtering problem and about 5 times slower for the brachiation filtering problem. This is because, the UKF algorithm needs to propagate a lot of sigma points, while the EKF only propagates the mean. However, the UKF has the advantage of being much easier to implement as the Jacobians do not need to be computed.

To compare the filters, the difference between the true state and estimated state is found. The Root Mean Square Error (RMSE)is then calculated as shown in table 4.2.

States	EKF (RMS)	ErKF (RMS)	UKF (RMS)
$\phi(rad)$	3.1×10^{-4}	2.9×10^{-4}	6.7×10^{-4}
$\theta(rad)$	0.0063	0.0063	0.003
$\psi(rad)$	4.9×10^{-4}	4.7×10^{-4}	0.0014
$x(m)$	0.81	0.81	0.79
$y(m)$	3.1×10^{-4}	3.0×10^{-4}	7.7×10^{-4}
$z(m)$	0.038	0.038	0.11
$v_x(m/s)$	0.13	0.13	0.09
$v_y(m/s)$	0.0021	0.0021	0.0069
$v_z(m/s)$	0.020	0.020	0.028

Table 4.2: RMSE for simulated system states.

From the above, it cab be seen that comparable results are obtained for the EKF and ErKF. Overall, the ErKF is only slightly better than the EKF. The UKF produces better results (almost twice as good) for the pitch angle but is a lot worst in estimat9ing the roll and yaw angles. Also, the UKF is only slightly better in estimating the x position, but is not as good as the other two filters in estimating the y and z positions. Similarly, the UKF is better at estimating the velocity aong the x directions and worse than the EKF and ErKF for other velocities.

For the test results, no ground proof data is available for comparison. Thus, the residuals are used to measure performance. The RMS values of the residuals are used as metric and are shown in table 4.3.

Residuals	EKF (RMS)	ErKF (RMS)	UKF (RMS)
$x_p(m)$	0.26	0.26	0.27
$y_p(m)$	0.57	0.58	0.59
$z(m)$	0.67	0.67	0.67
$v_x(m/s)$	0.16	0.15	0.13
$d_1(m/s)$	0.0069	0.0065	0.0051
$d_2(m)$	0.018	0.018	0.002
$d_3(m)$	0.0044	0.0049	0.0187
d_4	0.026	0.025	0.057
d_5	0.0066	0.0066	0.0153

Table 4.3: RMS values of residuals.

The residual RMS values are similar for the EKF and ErKF. The UKF shows

better results for the velocity residual v_x, constraints d_1 and d_2, and shows equivalent results for the z measurement residual. However, it is worse for the five other measurement residuals. Thus, the UKF is worse than the EKF and UKF.

In both the simulation and test result, the EKF and ErKF errors are comparable. This is expected as these filters are equivalent in nature. The UKF does not perform better than the EKF and ErKF. Also, the EKF produces better results in terms of the time taken for completing an iteration. Thus, according to these results, the EKF is the best filter.

4.6 Summary

In this chapter, a simulation of the power line inspection robot is created in MATLAB, with the robot approximated as a point mass moving along a power line. Sensor models for the accelerometer, gyroscope and GPS are also devised. Three filters (EKF, ErKF and UKF) are the implemented following the methods described in chapter three. Estimated states are plotted against true values and the performance of each is evaluated and compared.

Practical testing was carried out by tying a rope between two known GPS points. A test platform is manually dragged by means of a pulley to emulate the robot moving along a power line. The test hardware consists of a microcontroller, a GPS sensor, an IMU and a data logger, which are all mounted on the platform. The measurements obtained are processed offline and the estimated true shape of the robot trajectory is compared to its true shape, verifying the proof of concept.

Finally, the performances of the filters are compared. The next chapter will conclude and provide recommendations.

Chapter 5

Conclusions

The aim of the research was to develop a low cost GPS/IMU system for the optimal state estimation of a power line inspection robot. The robot was modelled as a point mass moving along the power line. Three Kalman filtering techniques, the EKF, ErKF and UKF were designed for estimating the attitude and position of the robot. The state estimates were further improved by constraining the robot to lie on the power line, using constraint equations as pseudo measurements.

Simulations were carried out for each of the filters and the system was tested tand implemented offline as a proof of concept. The simulation and the test result look similar. However, the only ground truth data available for the test results was the shape of the line, which matched well with the estimates.

The EKF and ErKF produced comparable results in terms of covariances and RMS errors as shown by the results in chapter 4. On the other hand, the UKF produced lower error variances, but has lower tracking performance for many states when looking at the estimates with respect to the true values. Moreover, the UKF was by far the most computationally expensive and the EKF required the least computations. Thus, the EKF is the best filter according to the results obtained.

There are several limitations to this project. The first is the inability to use magnetometers. This reduces the amount of available measurements, which would potentially enhance the attitude estimates.

The second limitation was that the GPS sensor only had a 1 Hz refresh rate. The filters are allowed to run in dead reckoning mode when GPS signals are not available. Thus, it is only when GPS signals are received that there are major corrections to the states. A GPS with higher refresh rate would increase the performances of the filters considerably.

The third limitation is that no ground truth data was available for the test. Thus, the performance of the of the filters for the test could only be evaluated

using residuals.

Finally, the test rig was manually actuated and was not tested on an actual robot.

It is recommended to keep sensors aligned with body axes and position them at the center of mass of the platform when performing the tests for best results. Also, the EKF has the best performance in terms of the time taken to execute one loop, with good tracking performance and error bounds when compared to the other filters. Thus, the recommended optimal state estimator for this project is the EKF.

Future work should involve the online implementation of the filters on an actual robot on a power line. However, to compare the filters ground truth data must be obtained. Triangulation with cameras can be used to obtain real time positions, attitude and velocities forming the ground truth data.

Further, when considering a brachiating robot, the filters designed for the swinging motion can be integrated with the GPS/IMU filter, to reduce the amount of sensors used.

Finally, the use of magnetometers in an environment with magnetic interference can be investigated. A method of filtering out magnetometer sensor measurements from the power line's magnetic field will then be needed. If this is acheived, state estimates can be further enhanced.

Bibliography

[1] E. Boje, "Attitude and position estimation for a power line inspection robot," *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 529–535, 2016.

[2] J. Patel and E. Boje, "Brachiating power line inspection robot," in *Applied Robotics for the Power Industry (CARPI), 2014 3rd International Conference on*, pp. 1–6, IEEE, 2014.

[3] A. Werries and J. M. Dolan, "Adaptive Kalman filtering methods for low-cost GPS/INS localization for autonomous vehicles," tech. rep., Carnegie Mellon University, 05 2016.

[4] "Odometría visual en jderobot con sensor rgbd." viewed July 2017, http://jderobot.org/J.benitod-tfg.

[5] "Trilateration vs Triangulation – how GPS receivers work." viewed July 2017, http://gisgeography.com/trilateration-triangulation-gps/.

[6] "UKZN's power line inspection robot, which was recently exhibited in the United States.." viewed November 2016, http://caes.ukzn.ac.za/News/14-06-10/UKZN_power_line_robot_wows_crowds_at_Chicago_expo.

[7] D. Douglass and R. Thrash, *Sag and tension of conductor*. CRC Handbook IEEE Press, 2001.

[8] R. G. Brown and P. Y. Hwang, *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*, vol. 1. Patrick YC New York: Wiley, 1997.

[9] R. Langley, "NMEA 0183: A GPS receiver," *GPS world*, July 1995.

[10] T. Lorimer and E. Boje, "A simple robot manipulator able to negotiate power line hardware," in *Applied Robotics for the Power Industry (CARPI), 2012 2nd International Conference on*, pp. 120–125, IEEE, 2012.

[11] E. Eitelberg, *Optimal Estimation for Engineers*. NOYB press, 1991.

[12] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[13] J. Sola, "Quaternion kinematics for the error-state KF," tech. rep., Laboratoire d'Analyse et d'Architecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, 2012.

[14] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," tech. rep., University of Minnesota, Dept. of Comp. Sci. & Eng., 2005.

[15] A. El-Rabbany, *Introduction to GPS: the Global Positioning System*. Artech House, 2002.

[16] R. B. Langley, "Dilution of precision," *GPS world*, vol. 10, no. 5, pp. 52–59, 1999.

[17] J. Wendel and G. F. Trommer, "Tightly coupled GPS/INS integration for missile applications," *Aerospace Science and Technology*, vol. 8, no. 7, pp. 627–634, 2004.

[18] T. Rowell and E. Boje, "Obstacle avoidance for a power line inspection robot," in *Applied Robotics for the Power Industry (CARPI), 2012 2nd International Conference on*, pp. 114–119, IEEE, 2012.

[19] V. C. Ravindra, V. K. Madyastha, and A. Goyal, "The Equivalence Between Two Well-Known Variants of the Kalman Filter," 2012.

[20] M. Rhudy and Y. Gu, "Understanding nonlinear Kalman filters, part ii: An implementation guide," *Interactive Robotics Letters*, 2013.

[21] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[22] A. B. S-Bisnath and R. Vajeth, *The Planning, Design and Construction of Overhead Power Lines*. Crown Publications, 2005.

[23] E. Lindberg, "The overhead line sag dependence on weather parameters and line current," Master's thesis, Uppsala University, 2011.

[24] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[25] Y. Hel-Or, A. Rappoport, and M. Werman, "Relaxed parametric design with probabilistic constraints," *Computer-Aided Design*, vol. 26, no. 6, pp. 426–434, 1994.

[26] "Band-limited white noise." viewed July 2017, https://www.mathworks.com/help/simulink/slref/bandlimitedwhitenoise.html.

[27] J. Patel, "Design, modelling and control of a brachiating power line inspection robot," Master's thesis, University of Cape Town, 2016.

Appendices

Appendix A

Relation Between DCM and Quaternions

Following the derivation in [14], the DCM can be written in terms of Euler angles as follows.

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta\cos\psi & \cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi - \cos\phi\sin\theta\cos\psi \\ -\cos\theta\sin\psi & \cos\theta\cos\psi - \sin\phi\sin\theta\sin\psi & \sin\theta\cos\psi + \cos\phi\sin\theta\sin\psi \\ \sin\theta & -\sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{A.1}$$

Euler's formula for the DCM is shown in equation A.2.

$$\boldsymbol{R} = \cos\theta\boldsymbol{I_{3\times 3}} - \sin\theta[\boldsymbol{k}\times] + (1-\cos\theta)\boldsymbol{k}\boldsymbol{k}^{\boldsymbol{T}} \tag{A.2}$$

Using half angle formulae in the above yields equation A.3.

$$\boldsymbol{R} = (2\cos^2(\theta/2) - 1) - 2\cos(\theta/2)\sin(\theta/2)[\boldsymbol{k}\times] + (2\sin^2(\theta/2)\boldsymbol{k}\boldsymbol{k}^{\boldsymbol{T}} \tag{A.3}$$

After substituting the elements of equation 1.9 the above can be converted into equation A.4,

$$\boldsymbol{R}(\bar{q}) = (2q_4^2 - 1)\boldsymbol{I_{3\times 3}} - 2q_4[\boldsymbol{q}\times] + 2\boldsymbol{q}\boldsymbol{q}^{\boldsymbol{T}} \tag{A.4}$$

In equation A.5, the above is written explicitly.

$$\boldsymbol{R}(\bar{q}) = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_1q_2-q_3q_4) & 1-2q_1^2-2q_3^2 & 2(q_2q_3+q_1q_4) \\ 2(q_1q_3+q_2q_4) & 2(q_2q_3-q_1q_4) & 1-2q_1^2-2q_2^2 \end{bmatrix} \tag{A.5}$$

Converting from DCM to quaternions is achieved using equation A.6.

$$\bar{q} = \begin{bmatrix} (c_{23} - c_{32})/4q_2 \\ (c_{31} - c_{13})/4q_2 \\ (c_{12} - c_{21})/4q_2 \\ \sqrt{1 + c_{11}^2 + c_{22}^2 + c_{33}^2}/2 \end{bmatrix} \tag{A.6}$$

Appendix B

Quaternion Time Derivative

Following [14], the quaternion time derivative will be derived. L is the local frame and G is the global frame. The term $\bar{q}_G^{L(t)}$ represents quaternion rotation from the local to global frame. The sampling time is represented by Δt

$$\dot{\bar{q}}_t = \lim_{\Delta t \to 0} \frac{1}{\Delta t} (\bar{q}_G^{L(t+\Delta t)} - \bar{q}_G^{L(t)}) \tag{B.1}$$

In the equatin below the term $\bar{q}_G^{L(t+\Delta t)}$ is expanded in terms of the quaternion multiplication.

$$\dot{\bar{q}}_t = \lim_{\Delta t \to 0} \frac{1}{\Delta t} (\bar{q}_{L(t)}^{L(t+\Delta t)} \otimes \bar{q}_G^{L(t)} - \bar{q}_I \otimes \bar{q}_G^{L(t)}) \tag{B.2}$$

Using smalle angle approximation and fctorizing the above equation yields the following.

$$\dot{\bar{q}}_t \approx \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left(\begin{bmatrix} \frac{1}{2}\delta\theta \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \otimes \bar{q}_G^{L(t)} \tag{B.3}$$

Finally, applying limits, the quaternion derivative equation is obtained.

$$\dot{\bar{q}}_t = \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \otimes \bar{q}_G^{L(t)} \tag{B.4}$$

Appendix C

Table of Sag vs Temperature

TABLE 14.6 Sag and Tension Data for 795 kcmil-26/7 ACSR "Drake" 600-ft Ruling Span

Conductor: Drake
795 kcmil-26/7 ACSR
Area = 0.7264 in.^{2}
Creep is ***not*** a factor

Span = 600 ft

NESC Heavy Loading District

					Final		Initial	
Temp, °F	Iœ, in.	Wind, lb/ft^2	K, lb/ft	Resultant Weight, lb/ft	Sag, ft	Tension, lb	Sag, ft	Tension, lb
0	0.50	4.00	0.30	2.509	11.14	10153	11.14	10153
32	0.50	0.00	0.00	2.094	11.54	8185	11.09	8512
−20	0.00	0.00	0.00	1.094	6.68	7372	6.27	7855
0	0.00	0.00	0.00	1.094	7.56	6517	6.89	7147
30	0.00	0.00	0.00	1.094	8.98	5490	7.95	6197
60	0.00	0.00	0.00	1.094	10.44	4725[a]	9.12	5402
90	0.00	0.00	0.00	1.094	11.87	4157	10.36	4759
120	0.00	0.00	0.00	1.094	13.24	3727	11.61	4248
167	0.00	0.00	0.00	1.094	14.29	3456	13.53	3649
212	0.00	0.00	0.00	1.094	15.24	3241	15.24	3241

[a]Design condition.

Figure C.1: Sag vs temperature for a 600 ft span [7].

TABLE 14.5 Tension Differences in Adjacent Dead-End Spans

Conductor: Drake
795 kcmil-26/7 ACSR
Area = 0.7264 in.^{2}
Creep **is** a factor

Span = 700 ft

NESC Heavy Loading District

					Final		Initial	
Temp, °F	Ice, in.	Wind, lb/ft^2	K, lb/ft	Resultant Weight, lb/ft	Sag, ft	Tension, lb	Sag, ft	Tension, lb
0	0.50	4.00	0.30	2.509	13.61	11318	13.55	11361
32	0.50	0.00	0.00	2.094	13.93	9224	13.33	9643
−20	0.00	0.00	0.00	1.094	8.22	8161	7.60	8824
0	0.00	0.00	0.00	1.094	9.19	7301	8.26	8115
30	0.00	0.00	0.00	1.094	10.75	6242	9.39	7142
60	0.00	0.00	0.00	1.094	12.36	5429	10.65	6300[a]
90	0.00	0.00	0.00	1.094	13.96	4809	11.99	5596
120	0.00	0.00	0.00	1.094	15.52	4330	13.37	5020
167	0.00	0.00	0.00	1.094	16.97	3960	15.53	4326
212	0.00	0.00	0.00	1.094	18.04	3728	17.52	3837

[a]Design condition.

Figure C.2: Sag vs temperature for a 700 ft span [7].

Conductor: Drake
795 kcmil-26/7 ACSR
Area = 0.7264 in.^{2}
Creep is **not** a factor

Span = 1000 ft

NESC Heavy Loading District

					Final		Initial	
Temp, °F	Ice, in.	Wind, lb/ft^2	K, lb/ft	Resultant Weight, lb/ft	Sag, ft	Tension, lb	Sag, ft	Tension, lb
0	0.50	4.00	0.30	2.509	25.98	12116	25.98	12116
32	0.50	0.00	0.00	2.094	26.30	9990	25.53	10290
−20	0.00	0.00	0.00	1.094	18.72	7318	17.25	7940
0	0.00	0.00	0.00	1.094	20.09	6821	18.34	7469
30	0.00	0.00	0.00	1.094	22.13	6197	20.04	6840
60	0.00	0.00	0.00	1.094	24.11	5689	21.76	6300[a]
90	0.00	0.00	0.00	1.094	26.04	5271	23.49	5839
120	0.00	0.00	0.00	1.094	27.89	4923	25.20	5444
167	0.00	0.00	0.00	1.094	30.14	4559	27.82	4935
212	0.00	0.00	0.00	1.094	31.47	4369	30.24	4544

[a]Design condition.

Figure C.3: Sag vs temperature for a 1000 ft span [7].

Appendix D

Simulation Error Bounds

The Graphs below show error states with their corresponding 3 σ error bounds.

Simulation Results

EKF Error Bounds

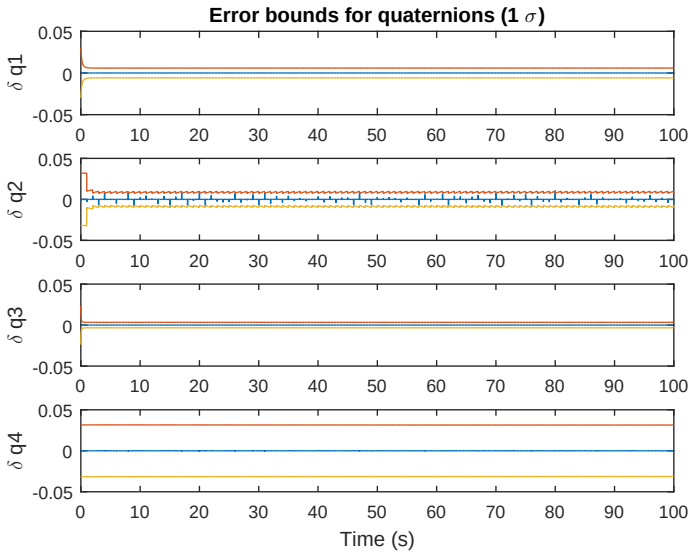

Figure D.1: EKF Quaternion error bounds

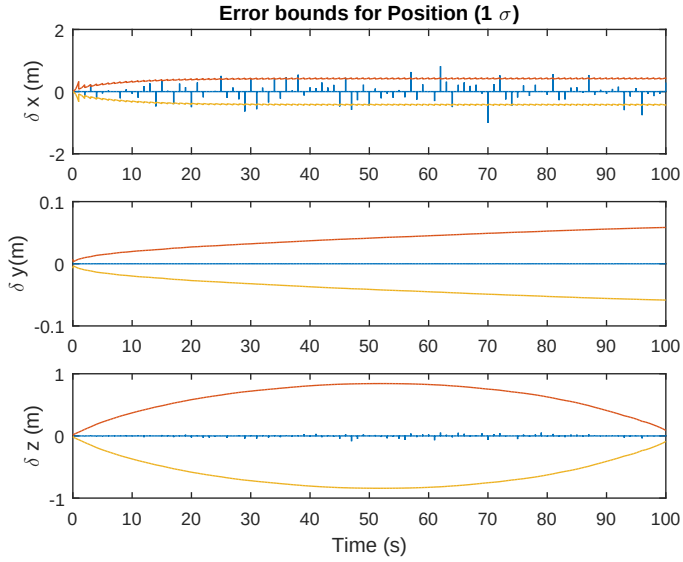

Figure D.2: EKF Position error bounds

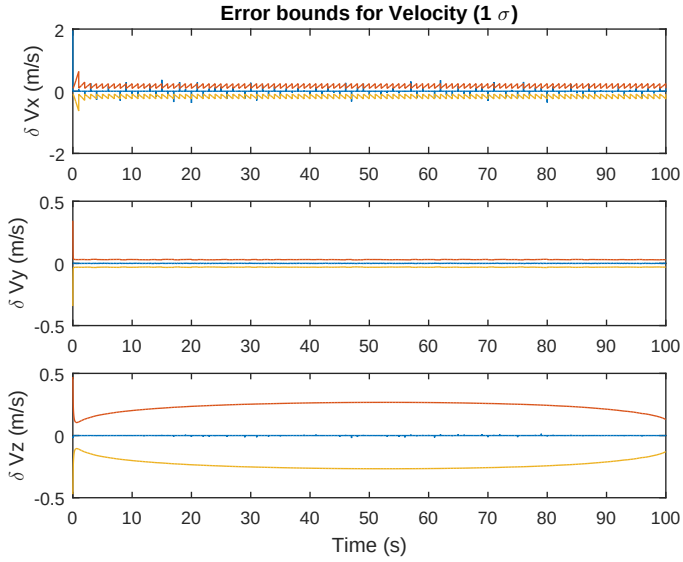

Figure D.3: EKF Velocity error bounds

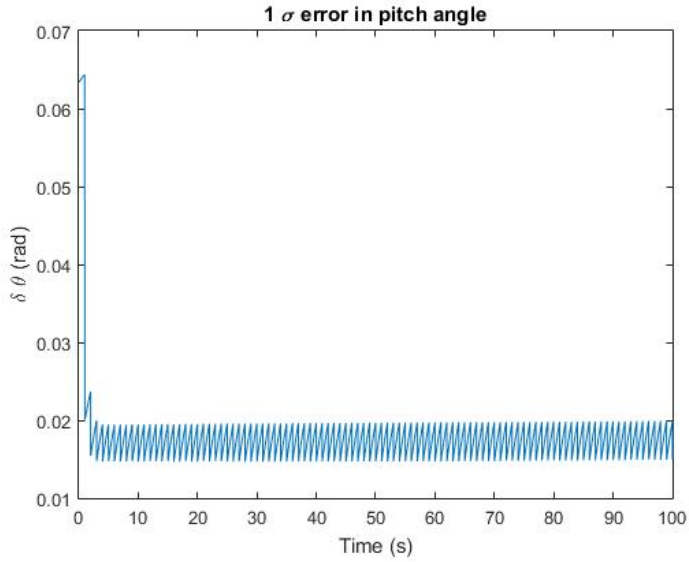

Figure D.4: EKF pitch angle error bounds

ErKF Error Bounds

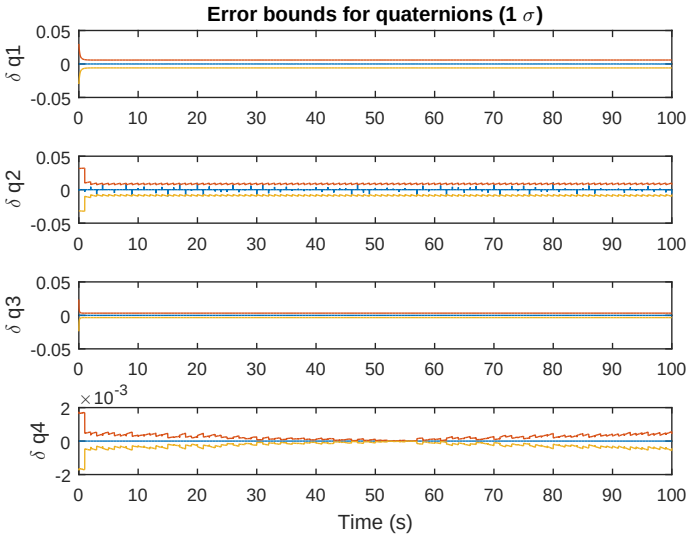

Figure D.5: ErKF Quaternion error bounds

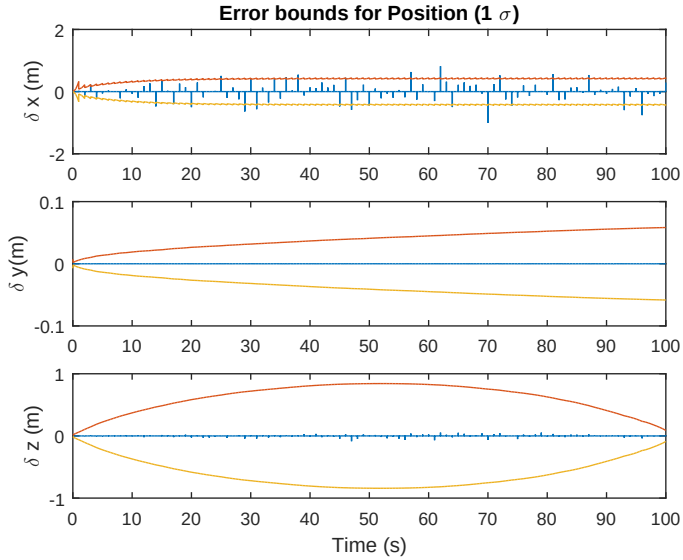

Figure D.6: ErKF Position error bounds

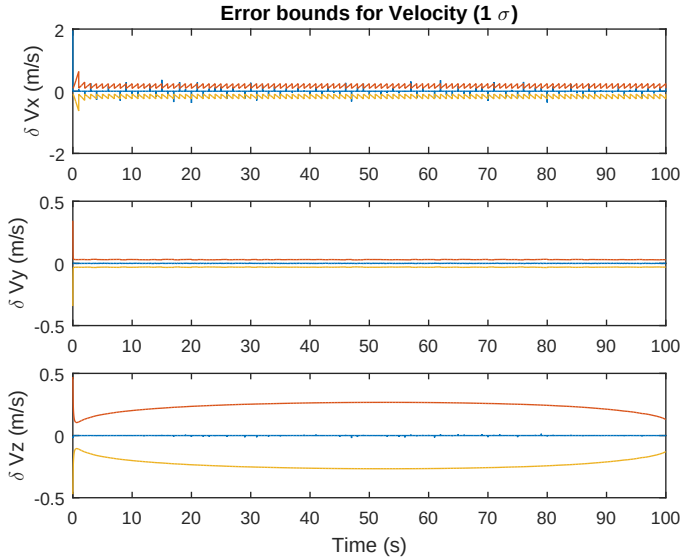

Figure D.7: ErKF Velocity error bounds

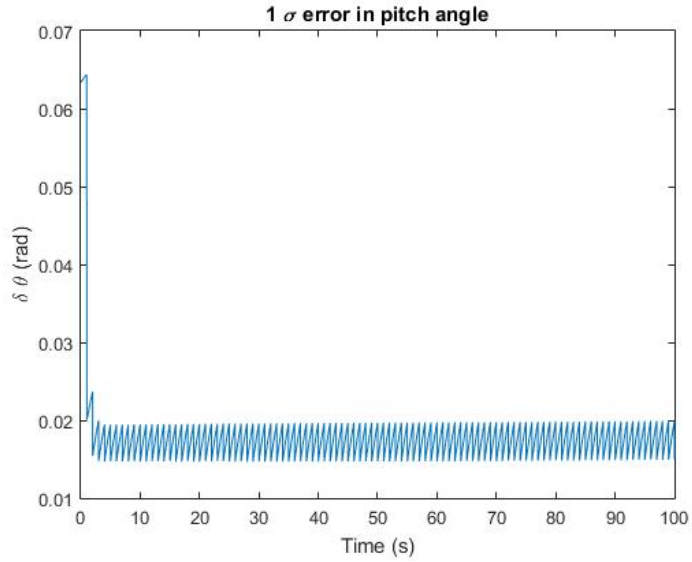

Figure D.8: ErKF pitch angle error bounds

UKF Error Bounds

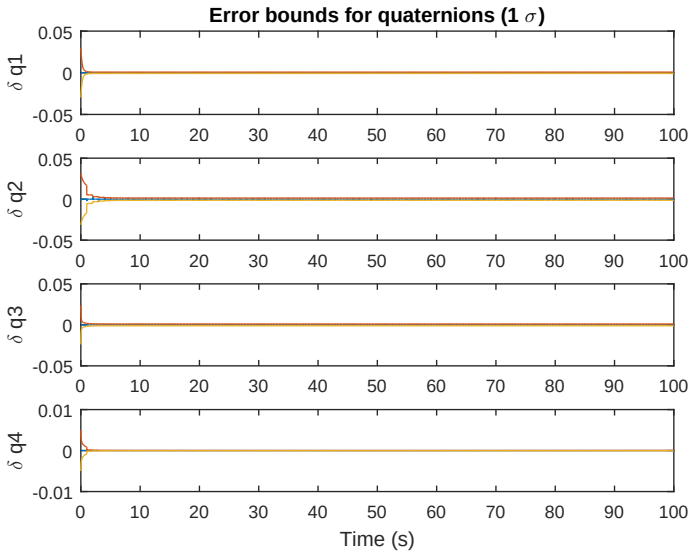

Figure D.9: UKF Quaternion error bounds

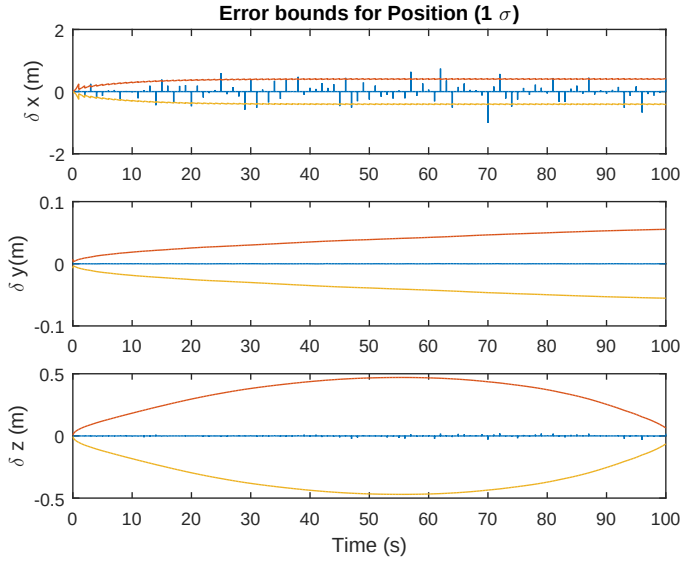

Figure D.10: UKF Position error bounds

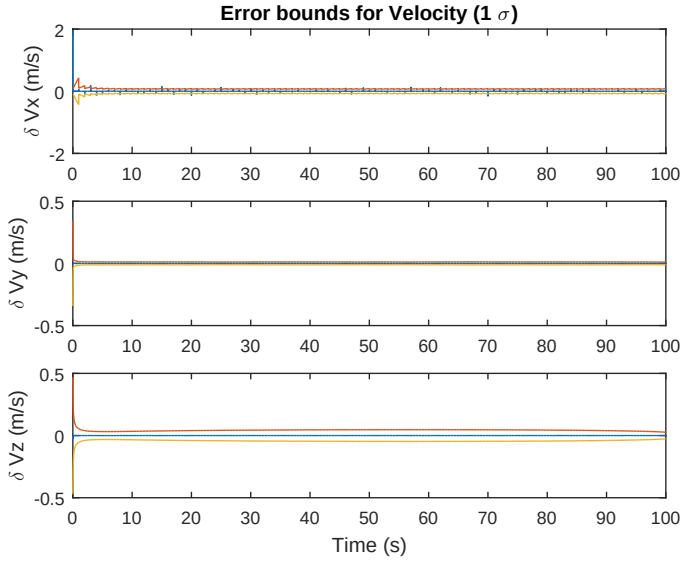

Figure D.11: UKF Velocity error bounds

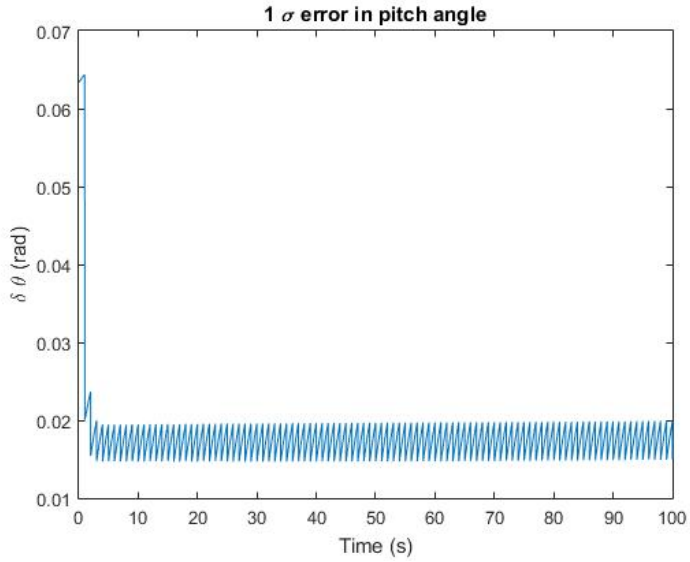

Figure D.12: UKF pitch angle error bounds

Simulation Results for Biases

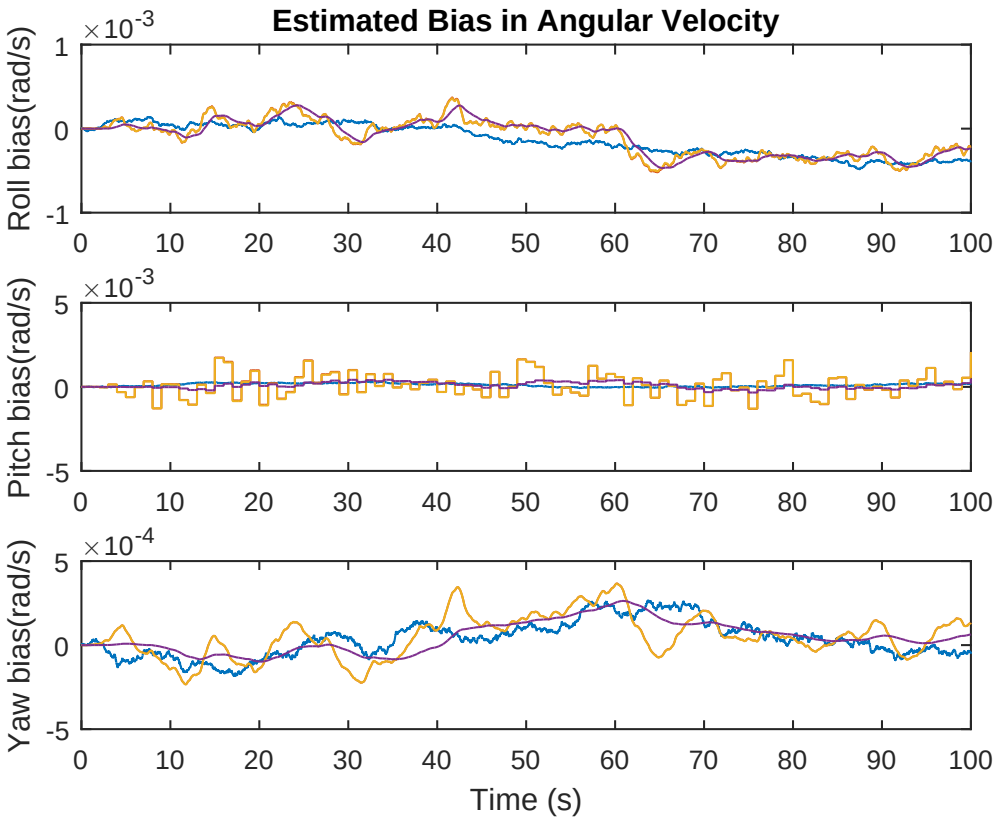

Figure D.13: Simulation of angular velocity bias for the three filters - EKF(Red), ErKF(Orange) and UKF(Purple)and true Value (Blue).

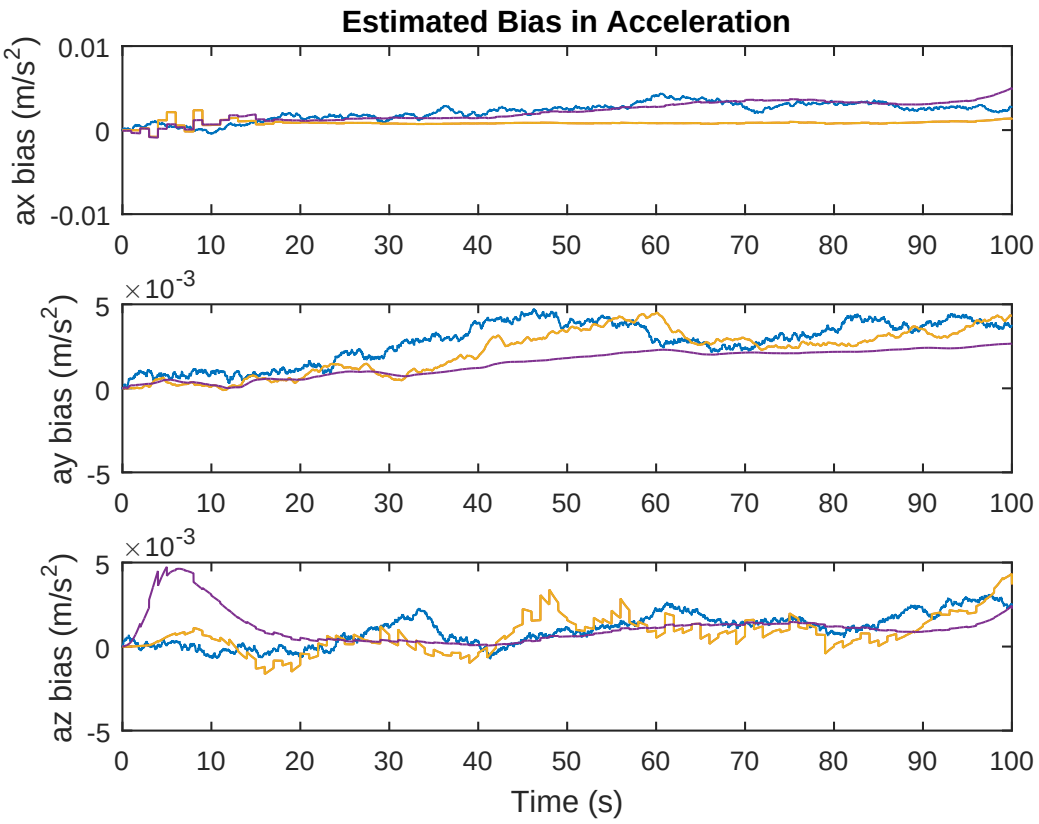

Figure D.14: Simulation of acceleration bias for the three filters - EKF(Red), ErKF(Orange) and UKF(Purple)and true Value (Blue).

Test Results Error Bounds

Velocity

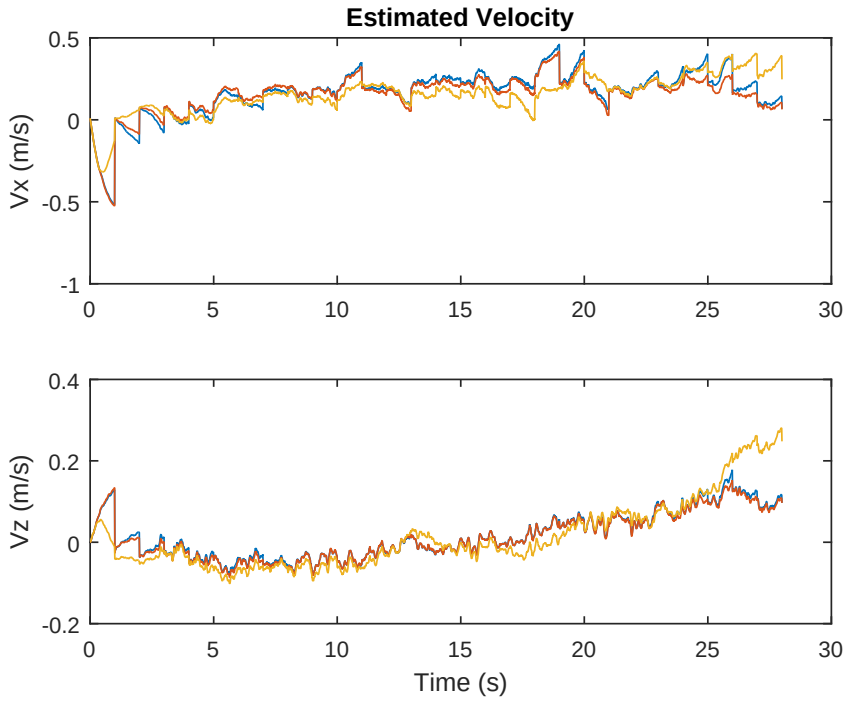

Figure D.15: Velocity estimates for the three filters - EKF(Blue), ErKF(Red) and UKF(Orange).

EKF Error Bounds

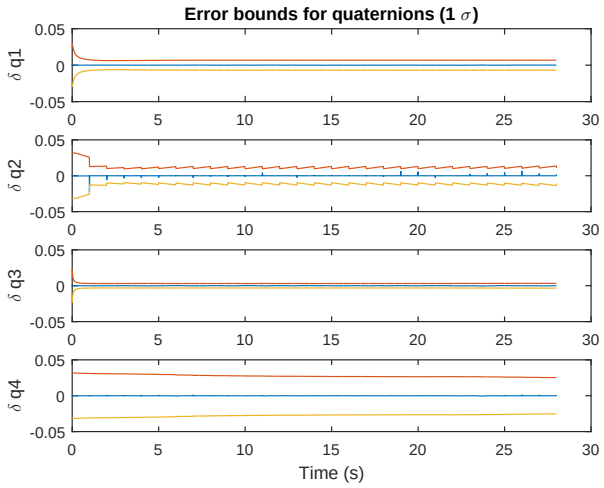

Figure D.16: EKF Quaternion error bounds

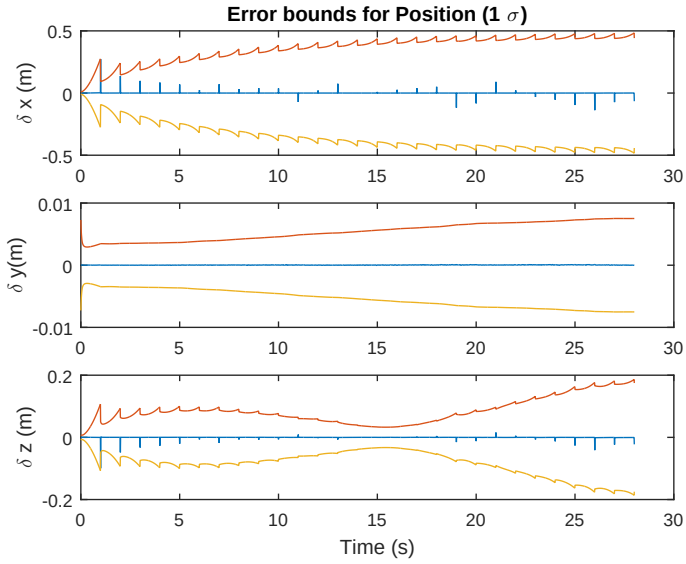

Figure D.17: EKF Position error bounds

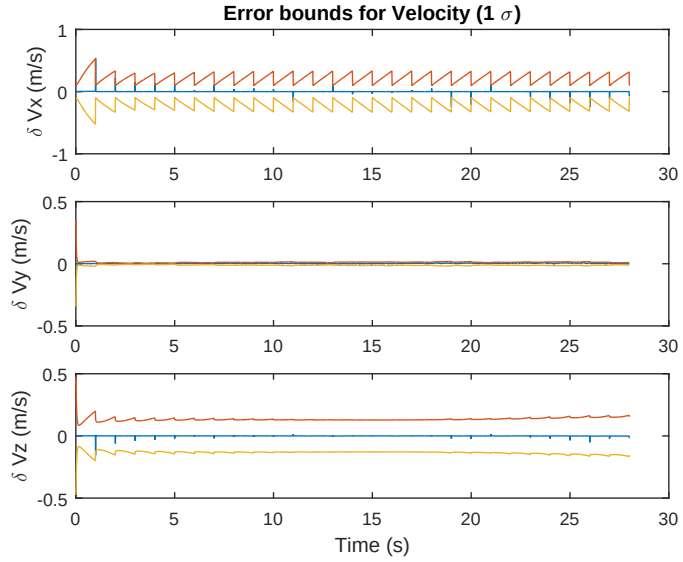

Figure D.18: EKF Velocity error bounds

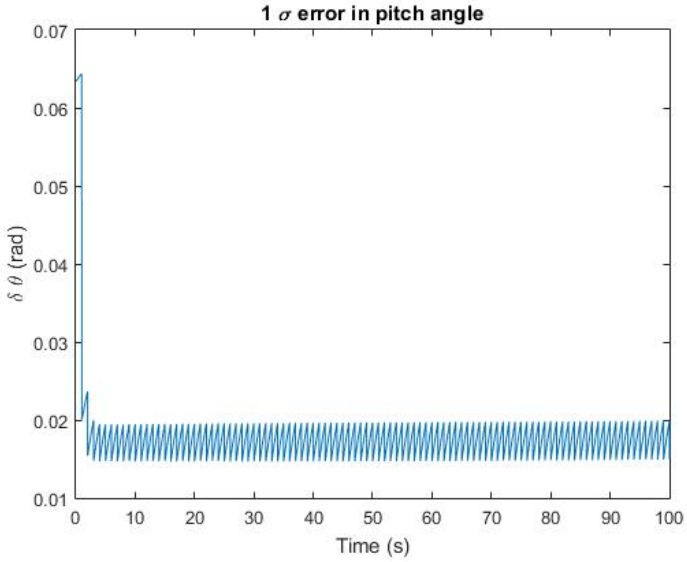

Figure D.19: EKF pitch angle error bounds

ErKF Error Bounds

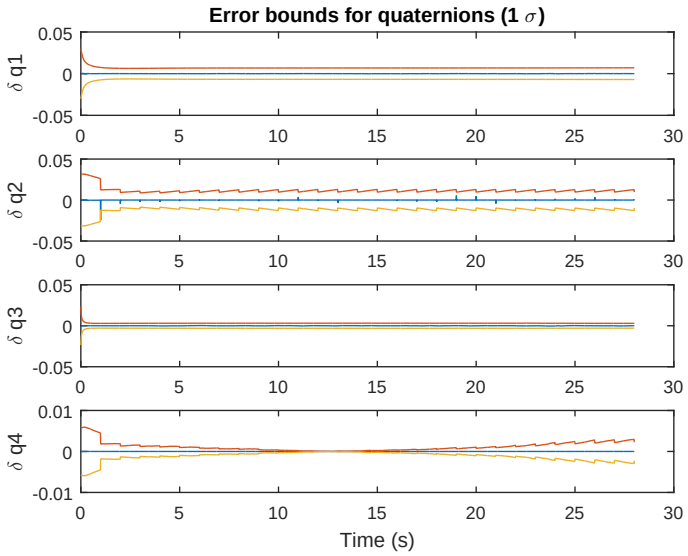

Figure D.20: ErKF Quaternion error bounds

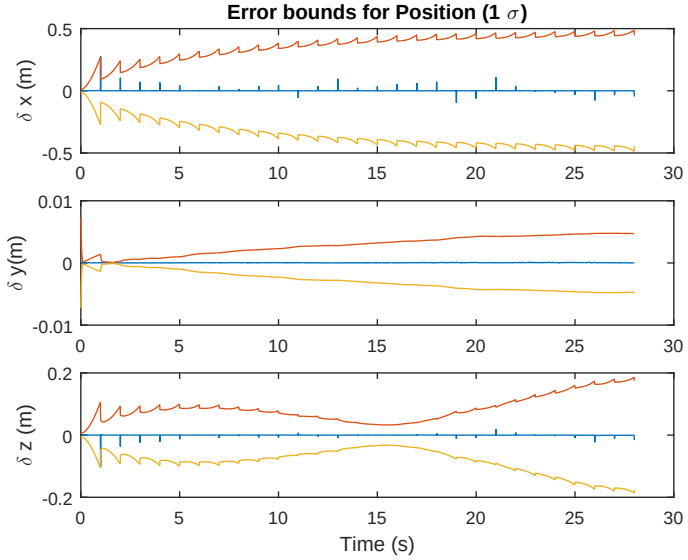

Figure D.21: ErKF Position error bounds

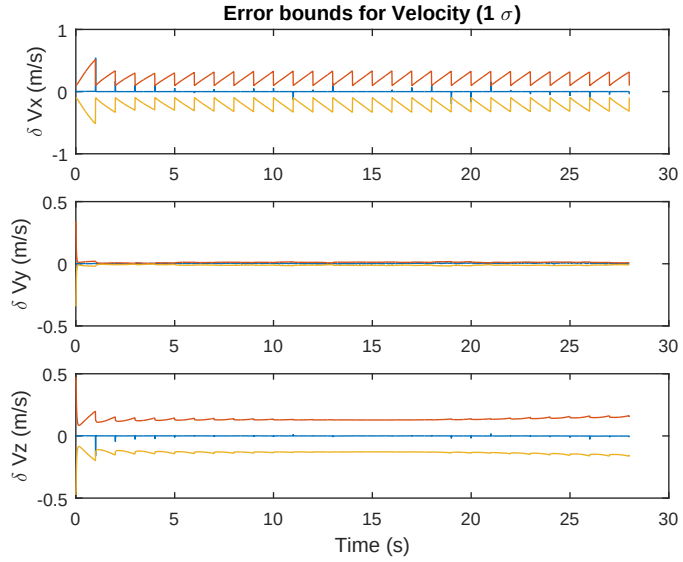

Figure D.22: ErKF Velocity error bounds

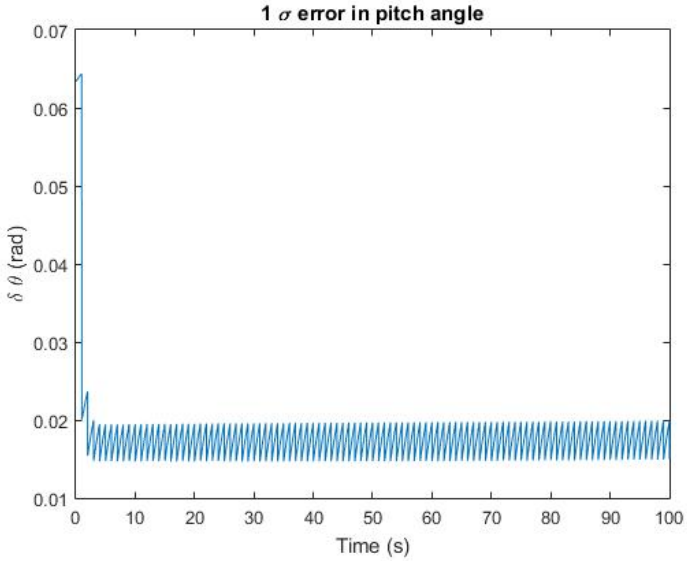

Figure D.23: ErKF pitch angle error bounds

UKF Error Bounds

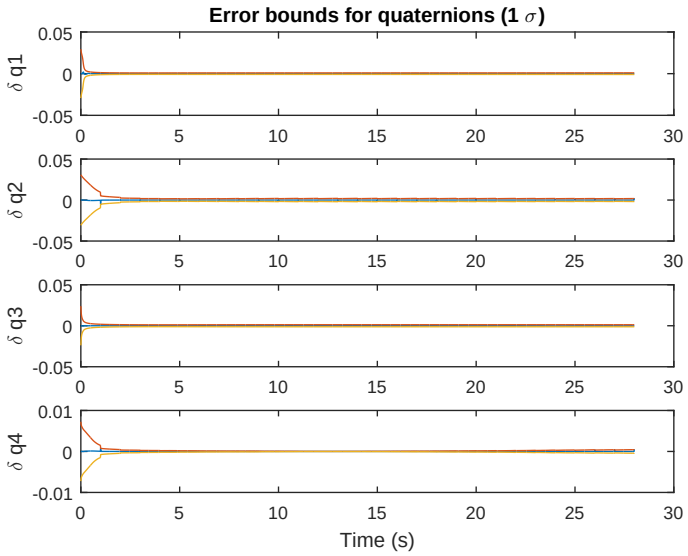

Figure D.24: UKF Quaternion error bounds

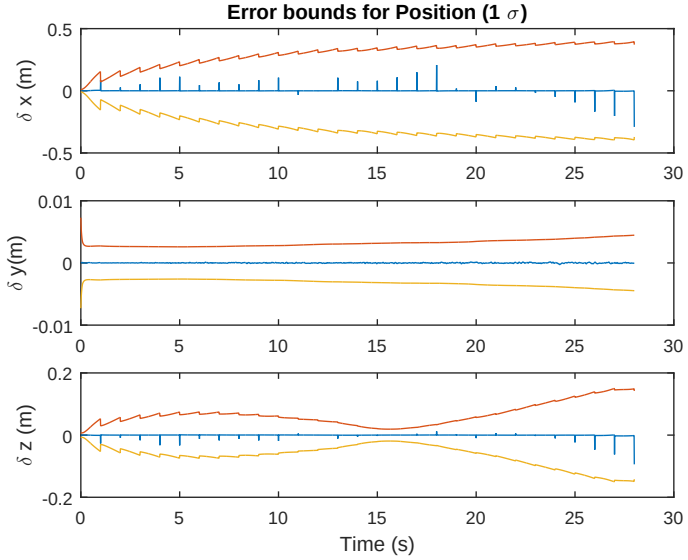

Figure D.25: UKF Position error bounds

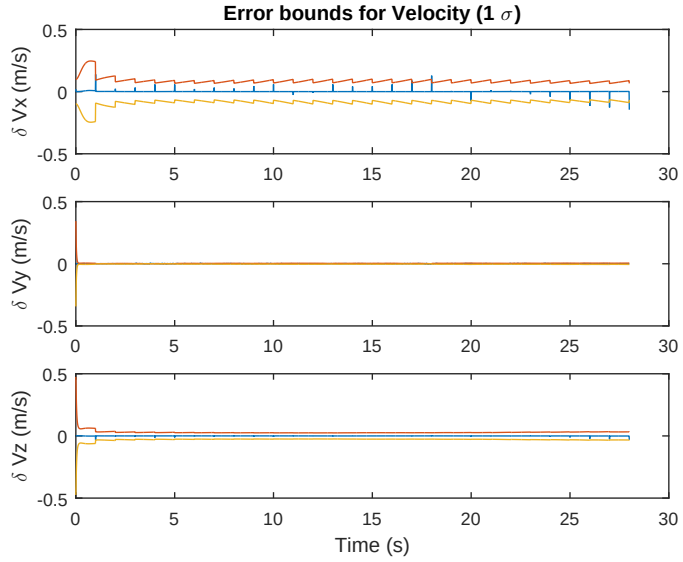

Figure D.26: UKF Velocity error bounds

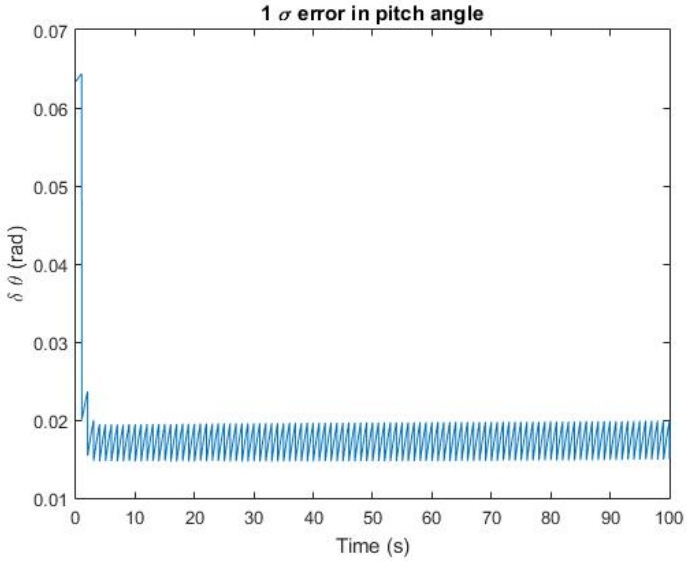

Figure D.27: UKF pitch angle error bounds

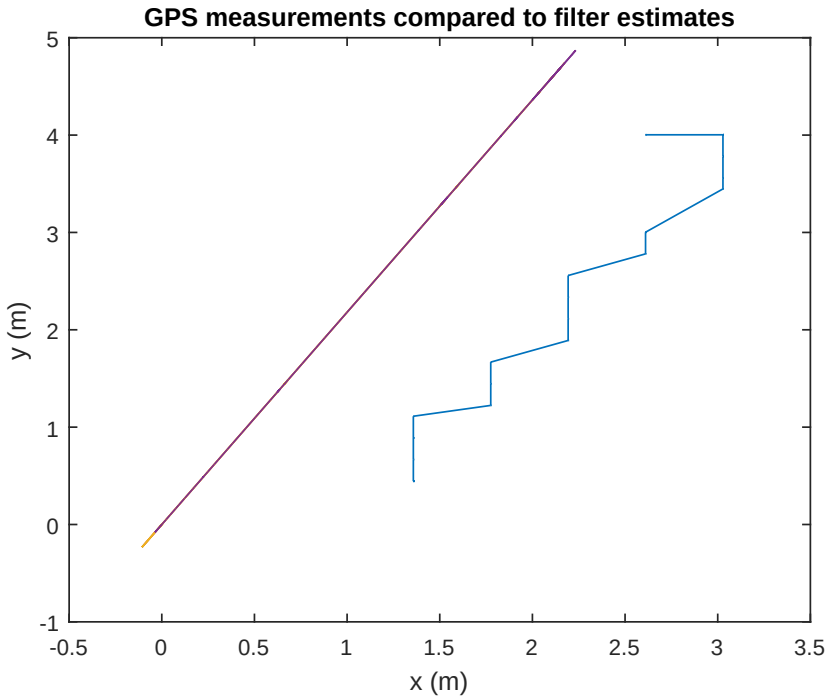

Figure D.28: Comparing GPS measurements to filter estimates - GPS (Blue), EKF (Orange), ErKF (Yellow) and UKF (Purple)

Appendix E

Simulation Study of State Estimation for a Brachiating Power Line Robot

E.1 Brachiation Model

A brachiating robot was developed by J.Patel at the University of Cape Town. When brachiating, the system can be approximated to a two dimensional system, since most of the accelerations occur along the vertical plane in which the power line lies (i.e. it is assumed that there is no rolling and yaw associated with the motion). Each arm of the robot will have an IMU (accelerometer and gyroscope) attached at its center of mass. [27]

The state-space model is derived from the kinematics of the system (derivation shown in appendix E). The accelerations in the x and y directions at the center of mass of each link are given by equations (**??**) and (**??**) (where subscript 1 denotes the upper link and subscript 2 denotes the lower link). To facilitate the derivation, two new angles, α and β, will be defined using θ_1 and θ_2 which are shown in figure E.1 (equations (E.1) and (E.2)).

$$\alpha = 90 - \theta_1 \tag{E.1}$$

$$\beta = 180 + \theta_2 - \alpha = \theta_1 + \theta_2 + 90 \tag{E.2}$$

Finding the displacement of the center of mass of the upper link (link 1) in the inertial frame x and y direction gives equation (E.3).

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l_{c1} \cos\alpha \\ -l_{c1} \sin\alpha \end{bmatrix} \tag{E.3}$$

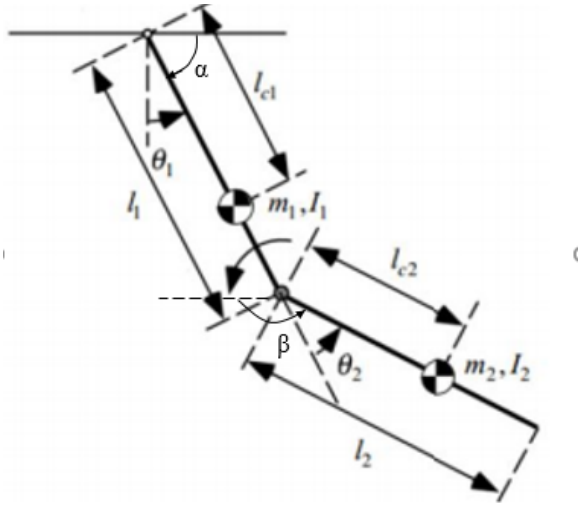

Figure E.1: The Brachiating robot [2].

Taking the derivative of the above to find the inertial frame velocities of the upper link gives equation (E.4).

$$\begin{bmatrix} \dot{x_1} \\ \dot{y_1} \end{bmatrix} \begin{bmatrix} -l_{c1}\dot{\alpha}\sin\alpha \\ -l_{c1}\dot{\alpha}\cos\alpha \end{bmatrix} \tag{E.4}$$

Taking the derivative of the above to find the inertial frame acceleration of the upper link yields equation (E.5).

$$\begin{bmatrix} \ddot{x_1} \\ \ddot{y_1} \end{bmatrix} = \begin{bmatrix} -l_{c1}\ddot{\alpha}\sin\alpha - l_{c1}\dot{\alpha}^2\cos\alpha \\ -l_{c1}\ddot{\alpha}\cos\alpha + l_{c1}\dot{\alpha}^2\sin\alpha \end{bmatrix} \tag{E.5}$$

Finding the displacement of the center of mass of the lower link (link 2) in the inertial frame x and y direction gives equation (E.6).

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} l\cos\alpha - l_{c2}\cos\beta \\ -l\sin\alpha - l_{c2}\sin\beta \end{bmatrix} \tag{E.6}$$

Taking the derivative of the above to find the inertial frame velocities of the lower link as in equation (E.7).

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix} \begin{bmatrix} -l\dot{\alpha}\sin\alpha + l_{c2}\dot{\beta}\sin\beta \\ -l\dot{\alpha}\cos\alpha - l_{c2}\dot{\beta}\cos\beta \end{bmatrix} \tag{E.7}$$

Taking the derivative of the above to find the inertial frame acceleration of the lower link as in equation (E.8).

$$\begin{bmatrix} \ddot{x_2} \\ \ddot{y_2} \end{bmatrix} = \begin{bmatrix} -l\ddot{\alpha}\sin\alpha - l\dot{\alpha}^2\cos\alpha + l_{c1}\ddot{\beta}\cos\beta + l_{c1}\dot{\beta}^2\sin\beta \\ -l\ddot{\alpha}\cos\alpha + l\dot{\alpha}^2\sin\alpha - l_{c2}\ddot{\beta}\cos\beta + l_{c2}\dot{\beta}^2\sin\beta \end{bmatrix} \tag{E.8}$$

The above equations are in the inertial frame, and need to be rotated to the body frame. This is acheived by using a rotation matrix as shown in equation (E.9). Vectors in the first link are rotated by angle $90^0 - \alpha$ and those of the second link by angle β.

$$\boldsymbol{a_i} = \boldsymbol{C_i} \begin{bmatrix} \ddot{x_i} \\ \ddot{y_i} + g \end{bmatrix} \tag{E.9}$$

Finding the acceleration of the upper link in terms of angular velocity and angular acceleration.

$$\begin{bmatrix} a_{x1} \\ a_{y1} \end{bmatrix} = \begin{bmatrix} \sin\alpha & \cos\alpha \\ -\cos\alpha & \sin\alpha \end{bmatrix} \begin{bmatrix} \ddot{x_1} \\ \ddot{y_1} + g \end{bmatrix} \tag{E.10}$$

$$\begin{bmatrix} a_{x1} \\ a_{y1} \end{bmatrix} = \begin{bmatrix} -l_{c1}\ddot{\alpha} + g\cos\alpha \\ -l_{c1}\dot{\alpha}^2 + g\sin\alpha \end{bmatrix} \tag{E.11}$$

Finding the acceleration of the lower link in terms of angular velocity and angular acceleration of the upper link and lower link.

$$\begin{bmatrix} a_{x2} \\ a_{y2} \end{bmatrix} = \begin{bmatrix} \sin\beta & -\cos\beta \\ \cos\beta & \sin\beta \end{bmatrix} \begin{bmatrix} \ddot{x_2} \\ \ddot{y_2} + g \end{bmatrix} \tag{E.12}$$

$$\begin{bmatrix} a_{x2} \\ a_{y2} \end{bmatrix} = \begin{bmatrix} \dfrac{l}{l_{c1}}\ddot{x_1}\sin\beta - \dfrac{l}{l_{c1}}\ddot{y_1}\cos\beta + l_{c2}\ddot{\beta} - g\cos\beta \\ \dfrac{l}{l_{c1}}\ddot{x_1}\cos\beta + \dfrac{l}{l_{c1}}\ddot{y_1}\sin\beta + l_{c2}\dot{\beta}^2 + g\sin\beta \end{bmatrix} \tag{E.13}$$

By rearranging the above, expressions for $\ddot{\alpha}$ and $\ddot{\beta}$ are obtained. These are then used to create the state-space equation as shown in equation (E.14) (The full derivations are shown in the appendix F). The rate of change of α and β are denoted ω_1 and ω_2 respectively, which are also the measured angular velocities in each link.

$$\begin{bmatrix} \omega_1 \\ \dot{\omega_1} \\ \omega_2 \\ \dot{\omega_2} \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \dfrac{1}{l_{c1}}(g\cos\alpha - a_{x1}) \\ \omega_2 \\ \dfrac{1}{l_{c2}}\left[\dfrac{l}{l_{c1}}(a_{x1}\cos(\alpha+\beta) + a_{y1}\sin(\alpha+\beta)) - \left(\dfrac{l}{l_{c1}} - 1\right)g\cos\beta + a_{x2}\right] \end{bmatrix} \tag{E.14}$$

The angular velocities obtained from the gyroscopes are to be used as measurements. The noise from these measurements can be quantified, by using the datasheet standard deviations. On the other hand, the process noise covariance will need to be constructed from the datasheet standard deviation and the state equations.

The process noise will only affect states ω_1 and ω_2 . The covariances for these will be derived independently to simplify the process.

The process noise of $\dot{\alpha}$ is given in equation (E.15).

$$Var(\ddot{\alpha}) = \mathrm{E}\left[\left(\frac{1}{l_{c1}}w_{a_{x1}}\right)\left(\frac{1}{l_{c1}}w_{a_{x1}}\right)\right] = \frac{1}{l_{c1}^2}\sigma_{a_{x1}}^2 \tag{E.15}$$

The process noise of $\dot{\beta}$ is given in equation E.16.

$$Var(\ddot{\beta}) = \mathrm{E}\left[\left(\frac{1}{l_{c1}l_{c2}}\cos(\alpha+\beta)w_{a_{x1}} + \frac{1}{l_{c1}l_{c2}}\sin(\alpha+\beta)w_{a_{y1}} + \frac{1}{l_{c2}}w_{a_{x2}}\right)^2\right] \tag{E.16}$$

It is assumed that both accelerometers have the same standard deviations in all directions (equation (E.17)).

$$\sigma_{a_{x1}}^2 = \sigma_{a_{y1}}^2 = \sigma_{a_{x2}}^2 = \sigma \tag{E.17}$$

This allows the simplification of the equation (E.16)as shown in equation (E.18).

$$Var(\ddot{\beta}) = \frac{\sigma^2}{l_{c2}}\left(1 + \left(\frac{l}{l_{c1}}\right)^2\right) \tag{E.18}$$

The process noise covariance, in matrix form is shown below (equation (E.19)).

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \dfrac{\sigma}{l_{c1}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{\sigma^2}{l_{c2}}\left(1+\left(\dfrac{l}{l_{c1}}\right)^2\right) \end{bmatrix} \tag{E.19}$$

The measurement model for the brachiation system will consist of the gyroscope measurements provided by each IMU (equation(E.20)). Since, angular velocity is a state of the system, the noise measurement covariance is therefore a 2x2 matrix with the gyroscope noise variances (obtained from datasheets, equation E.21) on the main diagonal.

$$\omega_m = \omega + v_\omega + b_\omega \tag{E.20}$$

$$Var(\omega_i) = \sigma_i^2 \quad \text{where} \quad i = 1, 2 \tag{E.21}$$

E.2 Brachiation Simulation

A dynamic model and optimal torque trajectory for the brachiating robot was obtained from J.Patel [27]. These were used to generate angular rates, to which Gaussian noise was added to obtain a model of the gyroscope signals. The angular rate was integrated using the integrator block of MATLAB to get the angles α and β of section 3.4.The EKF and UKF were then simulated, producing the figures 5.2 to 5.6. Variances of $0.3^2(m/s^2)^2$ and $0.1^2(rad/s)^2$ were used for the accelerometer and gyroscope respectively to account for vibrations that may occur in the robot during swinging. The sampling frequency used was 100 Hz.

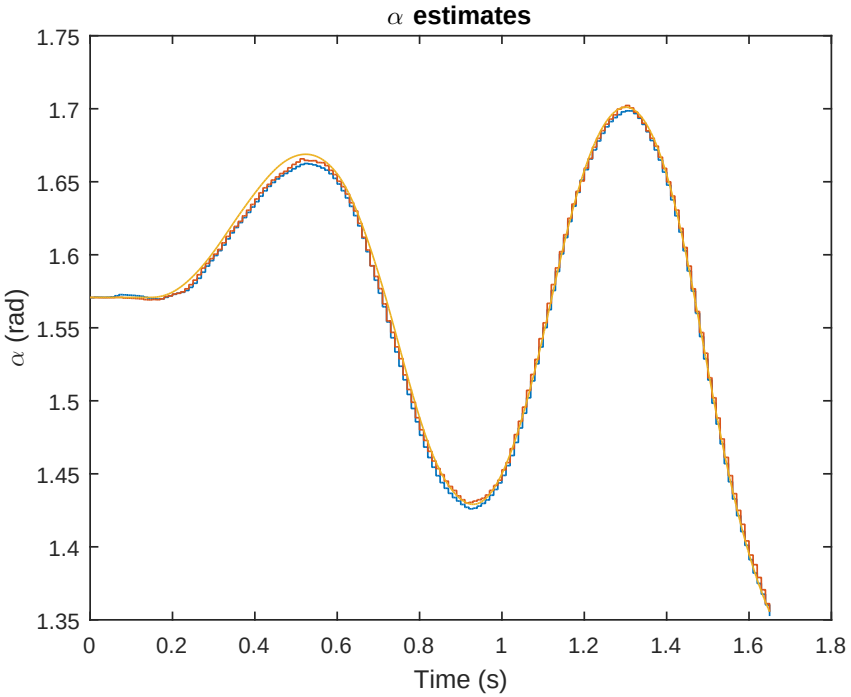

Figure E.2: Simulation of α - EKF(Blue), UKF(Red) and True Value(Yellow).

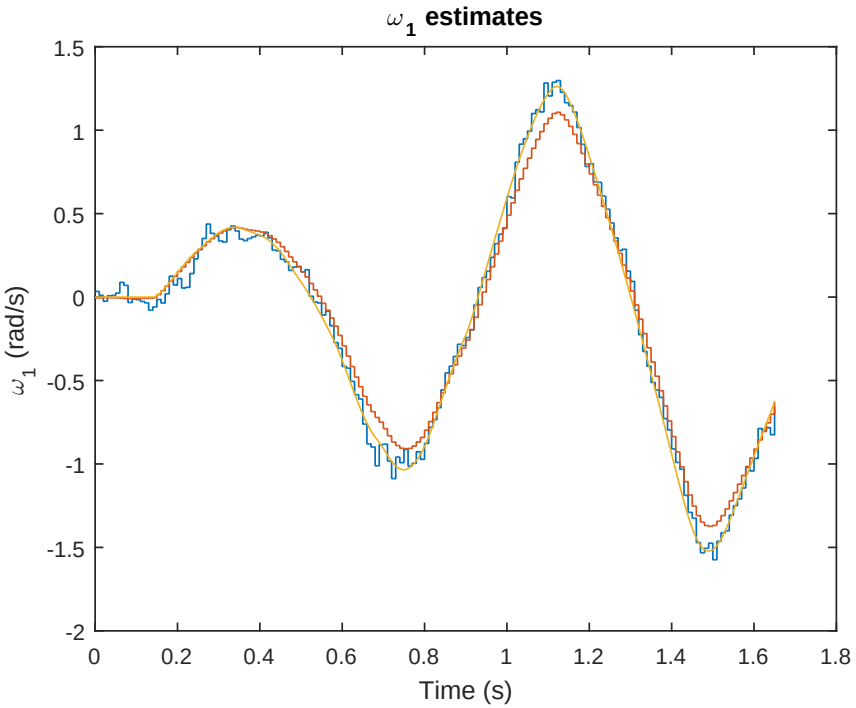

Figure E.3: Simulation of ω_1 - EKF(Blue), UKF(Red) and True Value(Yellow).

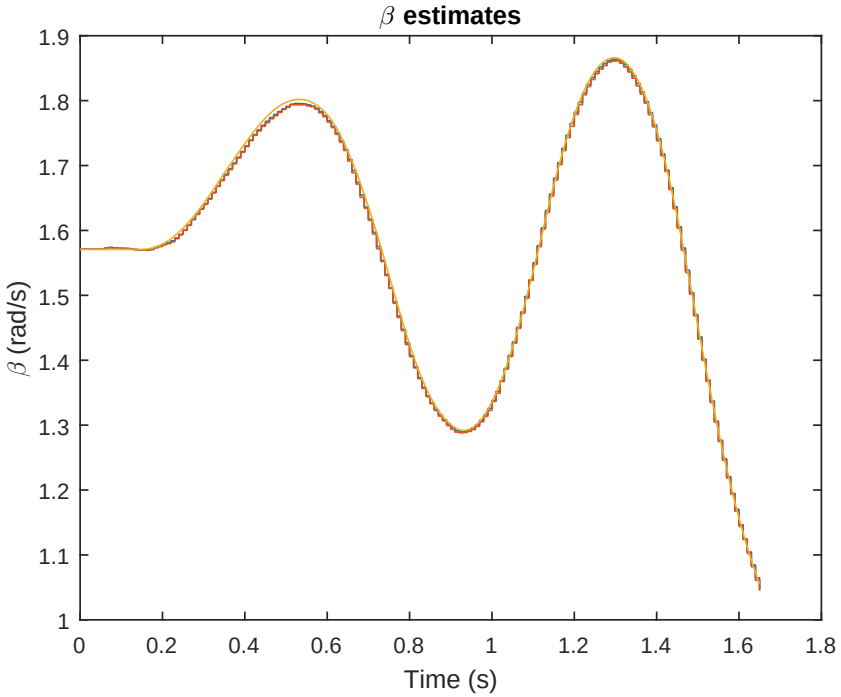

Figure E.4: Simulation of β - EKF(Blue), UKF(Red) and True Value(Yellow).

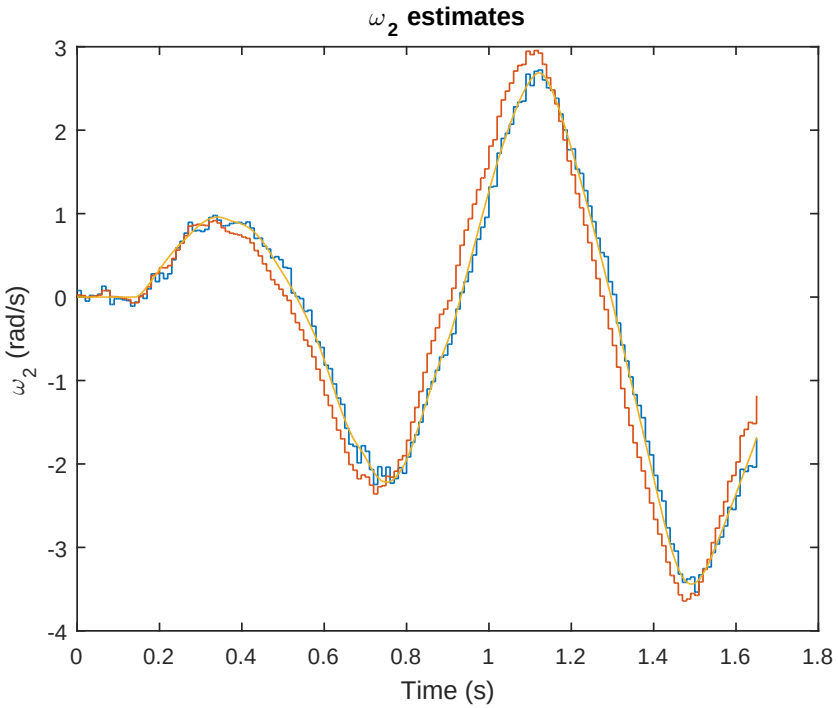

Figure E.5: Simulation of ω_2 - EKF(Blue), UKF(Red) and True Value(Yellow).

The tracking performances of both filters are similar for the angle estimates. However, the EKF angular rate estimates seemed to be closer to the true values as seen in figures 5.3 and 5.5, although they might appear to take longer to settle and have larger predicted error covariances (error states with 3 σ error bounds are shown in the figures below. The error states for the UKF exceed the error bounds, and become greater than the errors in the EKF for both ω_1 and ω_2. Thus, for this problem the EKF is the better solution.

E.3 Brachiation Simulation Error Bounds

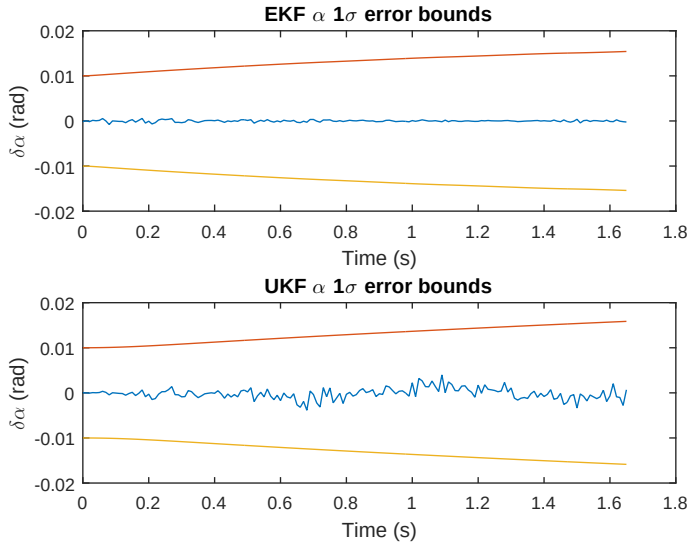

Figure E.6: 3 σ error bounds for angle α

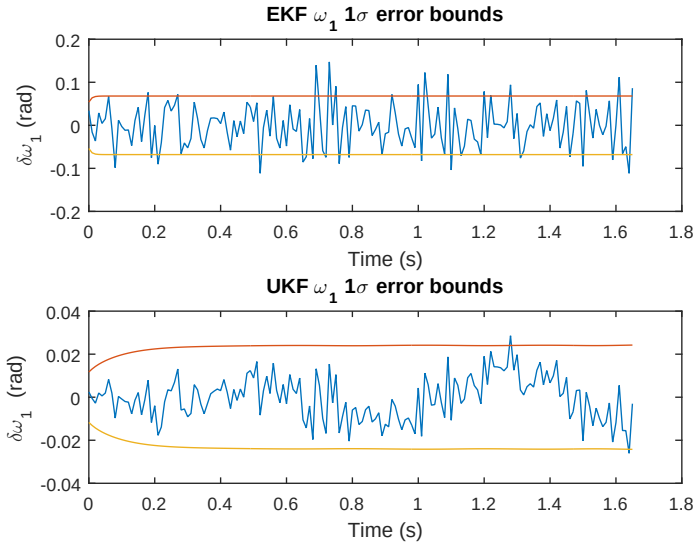

Figure E.7: 3 σ error bounds for ω_1

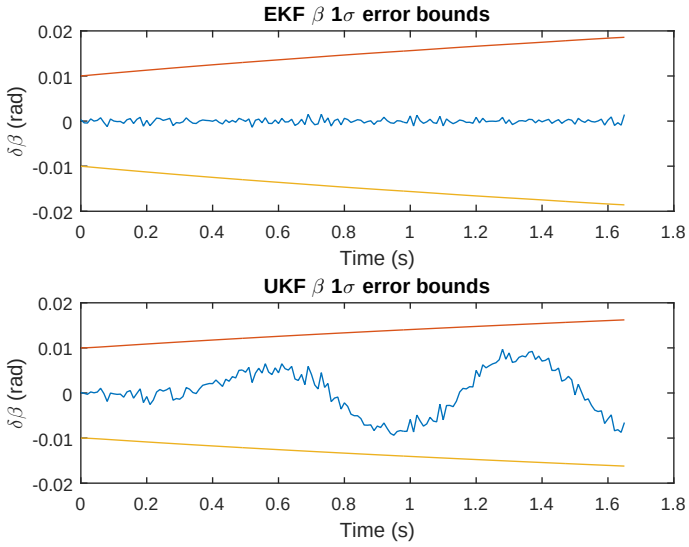

Figure E.8: 3 σ error bounds for angle β

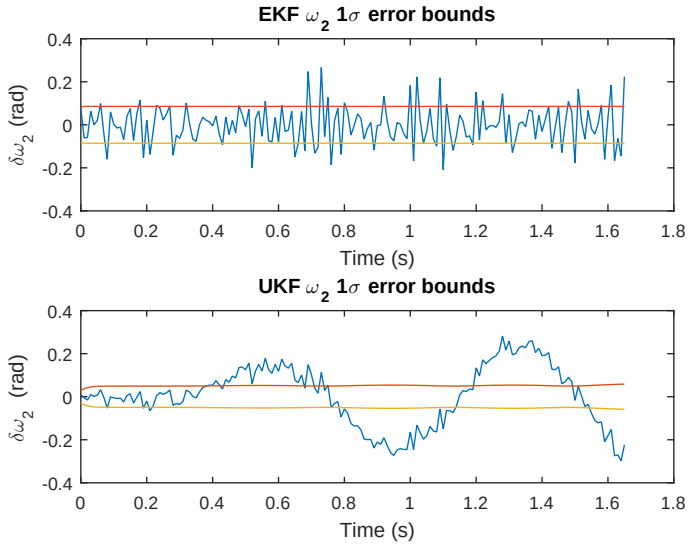

Figure E.9: 3 σ error bounds for ω_2

Appendix F

Simulink Models

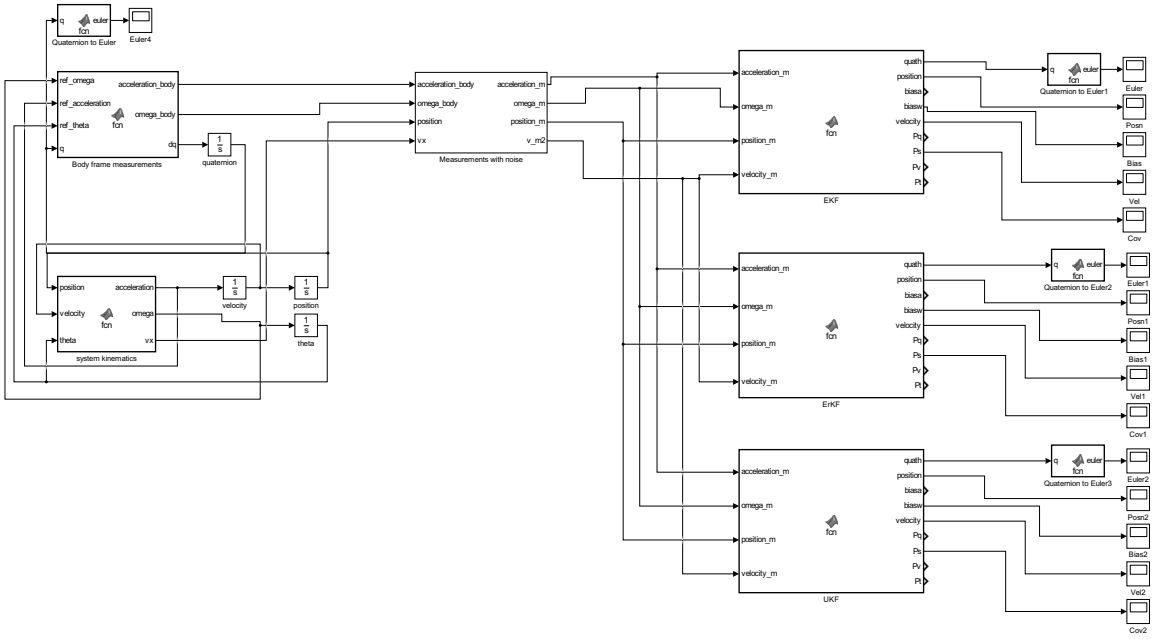

Figure F.1: GPS/IMU Filter Simulink Simulation Model

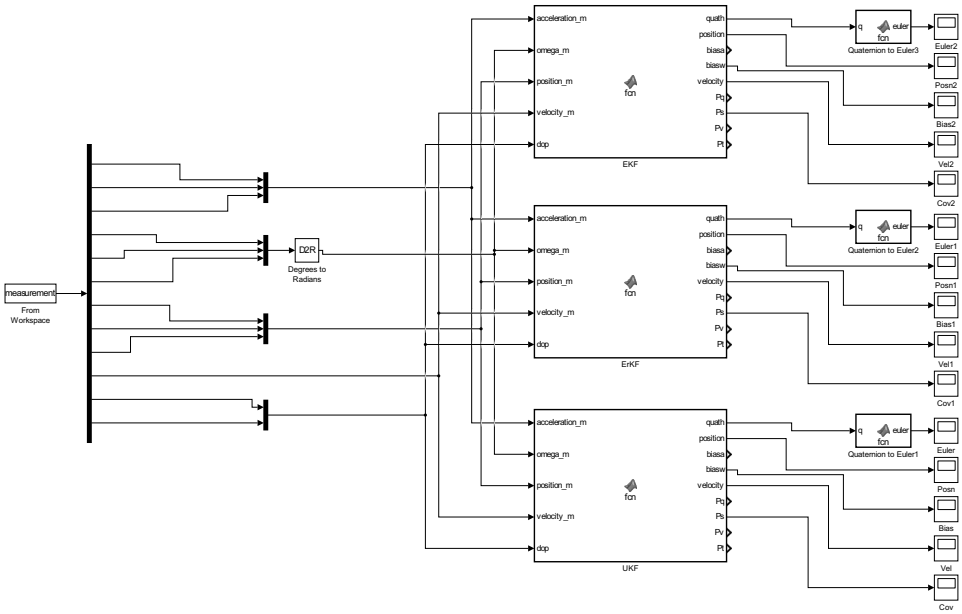

Figure F.2: GPS/IMU Filter Simulink Offline Test Model

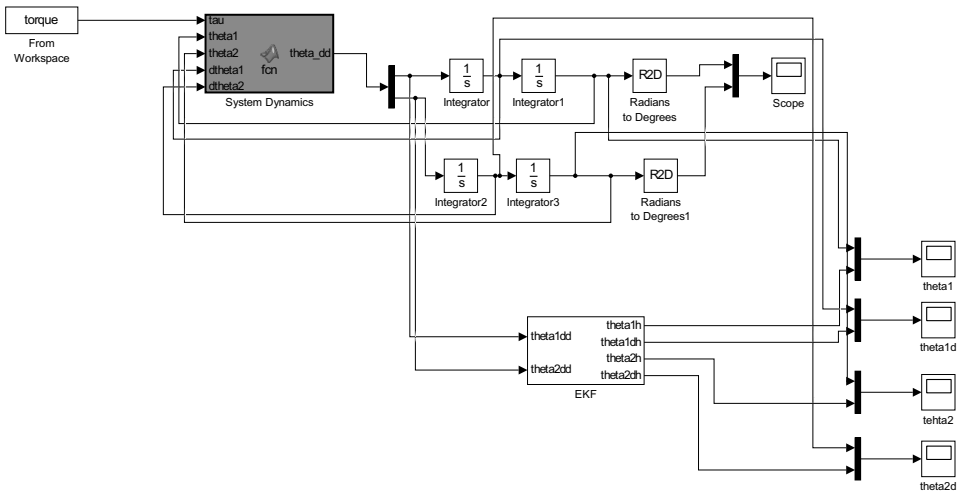

Figure F.3: Brachiation Simulation Simulink Model

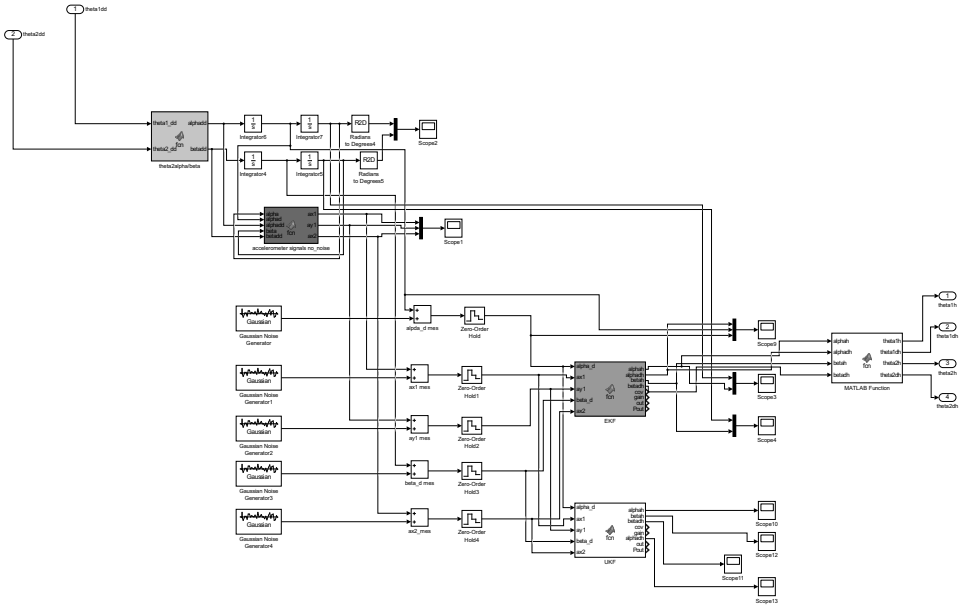

Figure F.4: Brachiation EKF Simulink Model

Appendix G

Code

The MATLAB code used for simulation and testing, the Arduino code and the measurements can be accessed on google drive via the following link:

https://drive.google.com/drive/folders/0B7C-bpXAXSf8ZHRXZkRKeHRzRnM

Appendix H

Ethics Form

APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data.The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook**(available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/usr/ebe/research/ethics.pdf

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Divij Soobhug
Department		Electrical Engineering
Preferred email address of applicant:		sbhdiv002@myuct.ac.za
If a Student	Your Degree: e.g., MSc, PhD, etc.,	MSc Electrical Engineering
	Name of Supervisor (if supervised):	Prof. Edward Boje
If this is a researchcontract, indicate the source of funding/sponsorship		Click here to enter text.
Project Title		Optimal state estimation for a power line inspection robot

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

SIGNED BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Divij Soobhug		**13 Feb 2017**

APPLICATION APPROVED BY	Full name	Signature	Date
Supervisor (where applicable)	Prof. Edward Boje		**12 Aug 2017**
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section1; and for all Undergraduateresearch (Including Honours).	Prof. Edward Boje		**12 Aug 2017**
Chair : Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the above questions.	Click here to enter text.		Click here to enter a date.