University College Cork

Department Of Electrical

&

Electronic Engineering

Module EE4020 -Final Year Project

Final Report

Digital Control of a Quad-Rotor

ucc

Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

Author: Cillian O'Brien

Student Number: 111334286

Supervisor:

Dr. Gordon Lightbody

April 10, 2021

Abstract

The aim of this project is to control the attitude and location of a quad-rotor prototype using the following sensors; 3-axes accelerometer [1], 3-axis compass [2],3-axes gyroscope [3], a range finder [4] and a GPS Receiver [5]

First this report will introduce the concept of a quad-rotor and establish the importance of the project. Next the model will be explained, how the dynamics and kinematics were modeled,estimated followed by the rotations and frames used to explain and understand the quad-rotor.

To complete the model, actuators are approximated to first order systems with parameters obtained experimentally. The sensors are approximated using real sensor measurements so the noise can be modeled correctly, thus allowing the noise to be reduced. This lead to the creation of a model in Simulink and the implementation of attitude estimation by means of a Complementary Filter and Kalman Filter whose outputs were used as an input to an attitude controller.Finally the results of note will be presented and discussed. Note this report should be read in conjunction with S.Coulter and R.Christie final year reports.

Declaration

This was written entirely by the author, except where stated otherwise. The source of any material not created by the author has been clearly referenced. The work described in this report was conducted by the author, except where stated otherwise.

Name: Cillian O'Brien: **Student Signature:**_____

Acknowledgements

First, I would like to thank Dr. Gordon Lightbody for his assistance and advice throughout the course of the project. I would also like to thank him for the insightful knowledge and the patience he showed us throughout the project.

I must thank Hilary Mansfiled for all his technical help, advice and patience over the past seven months. It was very much appreciated.

Finally, I wish to thank my project partners Ross and Simon for their support, patience, and dedication over the past year.

Contents

List of Figures

Code Snippets

List of Tables

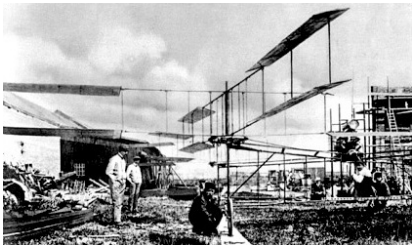
1. Introduction

Since the 1980's, Unmanned Aerial Vehicles (UAV)'s have been a subject of interest for the military. Up until the early/mid 1990's the technology required to implement these Unmanned Vehicles was not available. Today with the advance of miniaturization, more powerful processors and more reliable sensors it is finally possible to produce such devices. As sensors and micro-processors became more viable the interest in Vertical Take-off and Landing (VTOL) aircrafts increased. Hence, universities began to invest in the research of quad-rotors/helicopters due to their advantage over traditional fixed-wing crafts. VTOL are more advantageous in the following situations: rescue operations, delivery of goods, maneuvering in enclosed spaces. The military's interest in these devices stems from their high maneuverability even in minute form while still maintaining the ability to carry significant payloads, which makes them prime candidates for aerial surveillance and monitoring.

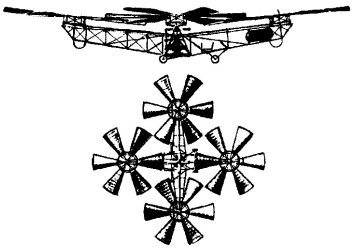
1.1 History

Despite the recent development and interest in the quad-rotor the concept is not new; the first idea for a quad-rotor was developed in 1907 by Louis and Jacques Breguet and was called the "Gyroplane No.1" (see figure 1.1a) [6]. The quad-rotor was propelled by four rotors with 4-blades mounted on the extremities of a cross-shaped structure. To cancel the rotational torques the rotorcraft used diagonally opposed rotors which rotate in opposite direction, the same theory is used in the quad-rotor featured in this report. Even though the rotorcraft achieved lift for a sustained period it could not remain stable enough to consider it a flight. It would take another 50 years for both the control theory and technology to catch up to this revolutionary idea, and allow the first actual flight of a quad-rotor.

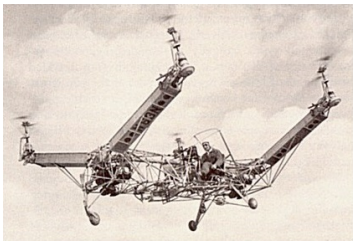
In 1921, the US Air Corps awarded a contract to Dr. George de Bothezat and Ivan Jerome to work on a vertical flight machine (see figure 1.1b) [6]. The result was a four six-bladed quad-rotor mounted at the ends of beams 20 metres in length arranged in a cross like structure. The craft overcame the stability problems present in quad-rotor



(a) Gyroplane No.1



(b) Bothezat's sketch



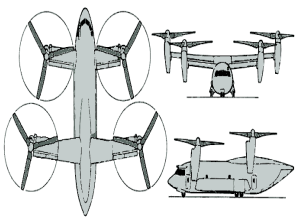
(c) Convertawings Model A

Figure 1.1: Development of the Quad-rotor concept [6]

built up until then by using two variable pitch controlled to ensure stability and control the craft. It was the first craft to prove that vertical flight was possible despite it's limited reach of only 5 meters.

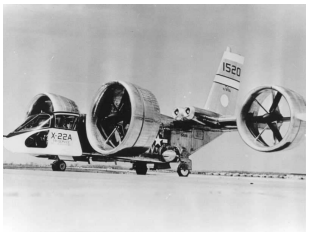
Due to stability problems the quad-rotor design was abandoned in favour of research of the now traditional helicopter. It wasn't until 1956 that the concept of the quad-rotor was once again modernized with the "Convertawings Model A" [8] (see figure 1.1c) whose development greatly improved the four rotor aircraft built by Oehmichen and Bothezat. With a simplified control system and greater power than its predecessors it was the first quad-rotor to fly successfully.

As a result of military interest the quad-rotor has been studied extensively as an alternative to traditional helicopters which were capable of carrying large payloads. The current and most widely known quad-rotor is the 2006 "Bell Boeing Quad Tilt-Rotor" (seen in figure 1.2a), which was based on the two "Curtiss-Wright X-19" and "Bell X-22" (see figures 1.2b and 1.2c respectively) which were two prototype quad-rotors built in the 1960's. The Bell Boeing Quad Tilt-Rotor was designed for military use to transport cargo of up to 9000kg to otherwise unreachable locations.



(a) Bell Boeing Quad Tilt-Rotor.

(b) Curtiss-Wright X-19.



(c) Bell X-22.

Figure 1.2: Further Development of the Quad-rotor Concept [6]

Today, major breakthroughs in the concept of the quad-rotor have taken place in universities across the world. Examples of such break through are as follows; comparison

of fixed-pitch and variable-pitch propeller based quad-rotors, the development of a robust controllers and test of a trajectory algorithm for variable-pitch aircrafts as seen in [9, 10]. A paper written by a Czech researcher was found which develops an Extended Kalman Filter (EKF) along with a LQR based control system for a quad-rotor which was used to adequately control the system [11]. In MIT a prototypes have been developed that are capable of exciting flips, and peak rotational rates exceeding $1600°s^{-1}$ were accomplished using traditional fixed-pitch technology [12]. Research groups have developed prototypes that can fly through windows and narrow gaps and perch on inverted surfaces [13]. A group in ETH zurich are now combining the quad-rotor with traditional control problems such as the inverted pendulum [14]. Other research personnel are currently focusing on mapping of areas using Simultaneous Localization and Mapping (SLAM) and streaming the results to a a remote computer [15]. Note the improvement of the quad-rotors are not just limited to the military and academia; some hobbyists have developed Remote Control (RC) variable-pitch quad-rotors and have posted their findings on RC forums and discussion boards [16].

2. Ethics

Over the recent years quad-rotors have found use in many areas some of which have raised ethical issues. Currently, there are numerous ethical problems, primarily flying quad-rotors in urban areas. For most parts these devices have to abide by the same restrictions as RC planes, i.e to be flown below a maximum of 120 m, 150 m from other people and the operator has to maintain line of sight at all times [17]. For research and commercial use written permission from the national aviation regulator is often required. Today, many hobbyists fly quad-rotors over urban areas: hobbyists feel they should be allowed to fly where they like (even with a camera on board) while the public want to maintain their privacy. Many quad-rotors sold to hobbyists some with holsters for a camera which are used to shoot video while in flight. The general public are very concerned about how this impacts privacy [18]. Clear and well though out rules must be designed to deal with this issue. Due to this privacy issue there have been numerous bills passed in order to deal with this growing problem [19].

Certain companies have taken an interest in quad-rotors due to their capabilities to navigate urban areas with ease. Amazon and Google use programs based around quad-rotors delivering items [20]. Certain medical services are interested in drones to facilitate organ implants to remote locations as they can reach higher speeds, thus, allowing more lives to be saved [21]. The concept of UAV is also being used to help herd cattle in remote locations in Australia [22]. As a result of large companies investing in quad-rotors to such a large degree some people (couriers) are worried that they could loss their jobs to a quad-rotor.

The military's interest in drones has driven the development of the quad-rotor to the level it is a today. The reasons the military have invested in such a platform such ass this must be considered when undertaking a dual use technology like this project Since the conception of the quad-rotor the military have shown great interest because of its potential to reach difficult locations and their lifting capability, which can be seen from the development of the "Bell Boeing Quad Tilt-Rotor". But more recently these drones have been used by the US and Israeli army for surveillance and there have been rumours/videos of quad-rotors being used as tactical weaponry [23, 24].

This potential dual use means this project could be used to help or hinder the progress of mankind.

3. Model Development

3.1 Quad-Rotor Dynamic

Helicopters and quad-rotors are complex flying machines due to their range of maneuverability [25, 26]. Traditional helicopters are equipped with tail rotors to counteract the clockwise/anti-clockwise moments due to the motor. However, the UAV discussed in this report uses a different technique.

Quad-rotors are symmetrical vehicles with four equally sized rotors positioned at the end of their respective mountings. Quad-rotors use multiple rotors so as to produce greater thrust and increase their maneuverability. Note that adjacent propeller blades are orientated opposite to each other. Thus if one set of rotors is spinning clockwise, then the two adjacent propellers have to spin counter-clockwise so that the torques produced by the propellers balance and cancel out if they all spin at the same rate

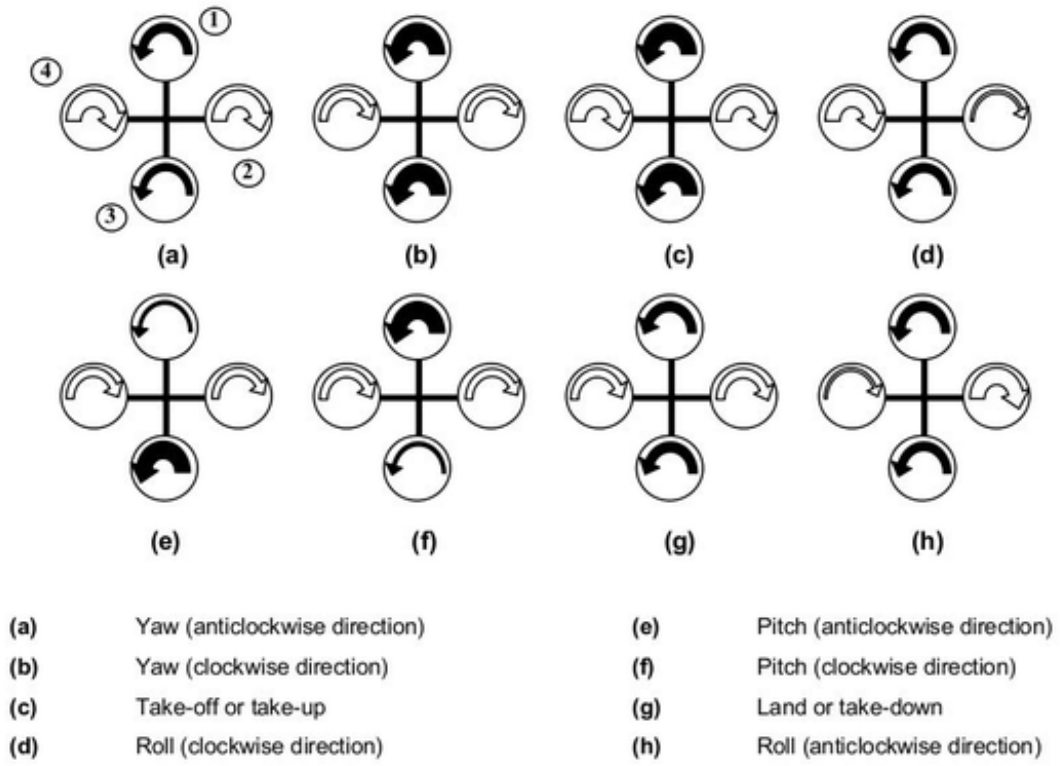


Figure 3.1: Quad-Rotor Torque Pattern

3.1.1 Motor Model

For this project a first order linear model of the motor was used to relate motor speed to PWM[1]. The first order modeled was used as it was difficult to find accurate values for the inductance and resistance of the Brushless DC Motor (BLDCM) that were used in this project.

$$\dot{\omega} = \frac{K_\omega D_i(t) - \omega}{\tau_m} \tag{3.1}$$

Note the value for K_ω was found from the experimental data and is 501.5 Rads^{-2}.

3.1.2 Pitch

Using Figure 3.1 as a reference, the pitch of the quad-rotor can be defined using the following equations once one knows the difference in the moments about the pitch-axis.

$$\tau_\theta = \sum l \times F = \Delta M = bl(\omega_1^2 - \omega_3^2) \tag{3.2}$$

Where b is a constant linking angular acceleration to angular rotation, l is the distance from the point of rotation to the motors and ω is the angular velocity of the motors.Thus, angular acceleration about the pitch axis can be defined as follows:-

$$\ddot{\theta} = \frac{\Delta M}{I_\theta} = \frac{bl(\omega_1^2 - \omega_3^2)}{I_\theta} \tag{3.3}$$

Where ω_1 & ω_3 are the speed of motor one and three respectively.

3.1.3 Roll

Similarly for roll, one can derive the following equation:-

$$\ddot{\phi} = \frac{\Delta M}{I_\phi} = \frac{bl(\omega_2^2 - \omega_4^2)}{I_\phi} \tag{3.4}$$

[1]Note it is the duration of the high portion of the signal that sets the speed of the motor, not the percentage of the duty cycle which is high. This is different to traditional PWM motor control.

3.1.4 Yaw

Each of the spinning rotors creates a reaction torque which can cause the quad-rotor to spin about the z-axis which is positioned through the quad-rotor's centre of mass (assuming a uniform frame and equal distancing of the rotor from the centre of mass). Hence, the following equation for yaw can be defined:-

$$\tau_r = J\dot{\omega} \tag{3.5}$$

The speed of the rotor and the drag coefficient also produce a reaction torque about the yaw axis which can be defined as follows :-

$$\tau_d = d\omega^2 \tag{3.6}$$

Hence the rate of change of yaw is given by the following equation :-

$$\ddot{\psi} = \frac{\sum_{i=1}^{4} \tau_r + \sum_{i=1}^{4} \tau_d}{I_\psi} \tag{3.7}$$

Filling in for τ_r and τ_d into (3.7) one gets the following equation :-

$$\ddot{\psi} = \frac{J(\dot{\omega_1} - \dot{\omega_2} + \dot{\omega_3} - \dot{\omega_4}) + d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)}{I_\psi} \tag{3.8}$$

Where, d is the drag coefficient of the propellers and J is the averaged inertia of the motors.

3.1.5 Movement in the Vertical & Horizontal Directions

To find the position of the quad-rotor in the vertical direction one can apply Newtons second law to find the following equation :-

$$\ddot{z} = \left[\frac{b}{m}(\cos\theta\cos\phi)\sum_{i=1}^{4}\omega_i^2\right] - g \tag{3.9}$$

Note the sum of the force in the vertical direction must be greater than the weight of the quad-rotor to ensure adequate lift.When the quad-rotor is pitched or rolled it must

supply extra thrust to maintain its altitude [27].

If it is required to move in the horizontal plane the resultant trust of the quad-rotor has to be resolved as follows by means of Euler Rotations:-

$$\ddot{x} = \frac{b}{m}(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\sum_{i=1}^{4}\omega_i^2 \tag{3.10}$$

$$\ddot{y} = \frac{b}{m}(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\sum_{i=1}^{4}\omega_i^2 \tag{3.11}$$

3.2 Kinematics and Dynamics Equations

3.2.1 Euler Angles

To describe the attitude and position of an object in free space (e.g an aircraft) a reference frame has to be attached to the object and then another can be attached to the earth, after which a relationship between the two coordinate systems can be defined. As the attitude of the device is expressed in the North-East-Down (NED) frame and its angles & velocity are measured by the 9 Degrees of Freedom (9DOF) in the Body Frame. Therefore it is necessary to be able to map from one frame to the other.

Euler angles [2] are not commutative and they must be applied in sequence to get the correct answer [28, pg. 24]. In this report the convention "*roll,pitch,yaw*", denoted (ϕ, θ, ψ), which are rotations about the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$-axes respectively was adopted. As the accelerometer and gyroscope measurements take place in the Body Frame, the following mapping scheme was introduced so the orientation of the device could be defined in the same frame as the control. A rotation from the NED frame to the Body Frame corresponds to a rotation about yaw $\mathbf{R}(\psi)^\intercal$, then about pitch $\mathbf{R}(\theta)^\intercal$ and finally about roll $\mathbf{R}(\phi)^\intercal$, these equations are presented fully in (3.12).

[2] Note in aviation these form of rotations are commonly refereed to as Tait–Bryan angles and are a sub-set of Euler angles

$$\mathbf{R}_1(\phi)^\intercal \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix}, \ \mathbf{R}_2(\theta)^\intercal \triangleq \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix}, \ \mathbf{R}_3(\psi)^\intercal \triangleq \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Finally combining these three rotations presented in (3.12) resulted in the following [3]:-

$$Rot \triangleq \mathbf{R}(\phi)^\intercal \mathbf{R}(\theta)^\intercal \mathbf{R}(\psi)^\intercal \triangleq \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\theta S_\phi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\theta C_\phi \end{bmatrix} \tag{3.13}$$

where:- *Rot* corresponds to the rotation matrix which maps a vector in the NED frame to the Body Frame. See Appendix B for more information on rotation matrix and Euler Angles

3.2.2 Singularities and Gimbal Lock

Note if ϕ, θ or $\psi = \pi/2$ the rotation matrix presented in (3.13) losses a degree of freedom. (3.14) presents a rotation matrix which has loss a degree of freedom as θ was set equal to $\pi/2$.

$$Rot = \begin{bmatrix} 0 & 0 & -1 \\ S_\phi C_\psi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi & 0 \\ S_\phi S_\psi + C_\phi C_\psi & C_\phi S_\psi - S_\phi C_\psi & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ S_{\phi-\psi} & C_{\phi-\psi} & 0 \\ C_{\phi-\psi} & -S_{\phi-\psi} & 0 \end{bmatrix} \tag{3.14}$$

In aviation this phenomenon is known as Gimbal Lock and corresponds to a reduction in degrees of freedom. As seen from (3.14) a change in ϕ and ψ has the same effect, thus a degree of uncertainty has been introduced as one notation can represent two very different orientations. This is a problem when using Euler Angles, but as the

[3]For Mathematical ease $\sin(\alpha)$ was simplified to S_α and $\cos(\alpha)$ was simplified to C_α. Note α is a place holder for the angle of interest.

quad-rotor was limited to $\pm 30^o$ on the θ and ϕ axis this problem was avoided [4]. The limit on the pitch and roll was a result of the lifting power of the motors, once the angle becomes greater than 40^o the net upwards thrust is less than what is required to maintain the quad-rotor's altitude.

3.3 Inertial Measurement Unit

3.3.1 Accelerometer model

In order to estimate the attitude[5] of the quad-rotor an accelerometer was used to measure the direction of the gravity vector, g , from this the pitch and roll angles can be found. Let g be constant, pointing down along z_n defined in the NED frame with an intensity $g_0 = 9.81$ m s^{-2} and let $\bar{\mathbf{a}}^B$ denote the accelerometer measurement vector. The definition of z_n can be seen in figure 3.2.

The accelerometer used in this project was an ADXL345, which uses the piezoelectric effect to an create electric signal due to accelerative forces acting on a micro-crystal structure inside the device, knowing this,the following equation was derived:-

$$\bar{\mathbf{a}}^B = Rot(g) - a^B + \mu_a + b_a \tag{3.15}$$

Where: $\bar{\mathbf{a}}^B$is the sensor output in m s^{-2}; g is the gravity vector in the NED frame; a^B is the acceleration of the quad-rotor; μ_a is the Gaussian noise component; b_a is the constant bias of the accelerometer.

Note when the 9DOF inertial measurement unit is placed at the center of mass (3.15) can be shown to simplify to (3.16). Note (3.16) doesn't account for the fact that quad-rotor could be in free fall in a horizontal position. If this occurred the output of the accelerometer would be 0. This could potentially be an issue.

Hence,in order to acquire attitude reads from the accelerometer a model was created that utilized the output vectors of the ADXL345 to estimate the attitude of the quad-rotor. (3.16) was used to estimate the attitude of the quad-rotor.

[4]Note the problem associated with a loss of freedom due to singularities can be alleviated by using quaternion, but they introduce their own problems as use dimensions to represent a point in free space.

[5]Attitude are the set of (ϕ, θ, ψ) angles in the Body Frame with respect to the NED frame.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = g \begin{bmatrix} -S_\theta \\ C_\theta S_\phi \\ C_\theta C_\phi \end{bmatrix} + \begin{bmatrix} \mu_{a-x} \\ \mu_{a-y} \\ \mu_{a-z} \end{bmatrix} + \begin{bmatrix} b_{a-x} \\ b_{a-y} \\ b_{a-z} \end{bmatrix} \tag{3.16}$$

Now assuming that the bias and noise on the accelerometer is zero[6] the orientation of the quad-rotor in the NED frame can be calculated. Hence, ϕ and θ can be defined as follows and can easily be shown to be correct by means of (3.16):-

$$\begin{aligned} \phi &= \arctan\left(\frac{a_y}{a_z}\right) \\ \theta &= \arctan\left(\frac{-a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right) \end{aligned} \tag{3.17}$$

3.3.2 Gyroscope model

A similar model was created for the gyroscope as the angular velocity measured by the gyroscope in the Body Frame doesn't correspond directly to the Euler angle rates $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^\intercal$. Instead the rate of change of angle can be defined with respect to the NED frame as follows:-

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & S_\phi C_\theta \\ 0 & -S_\phi & C_\phi C_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.18}$$

Now taking the in the inverse of (3.18) the following equation can be defined:-

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.19}$$

3.3.3 Digital Compass model

The magnetometer is a sensor designed to detect the magnetic North direction, written as $\mathbf{N^I}$. By the definition of the reference frame and neglecting magnetic inclination

[6]These can be removed in code and by filtering methods respectively. These will be developed later in this report.

and magnetic declination, $\mathbf{N^I} = [0,1,0]^{\mathrm{T}}$.

A Honeywell HMC5883L was used in this project. The device works by measuring the change in resistance with a change in the applied magnetic field. The device can measure these change as it is made from strips of permalloy. As the device can measure the change in magnetic field, the orientation of the magnetic field can be estimated using the following:-

$$\bar{\mathbf{N}}^B = RotN^I + \mu_m + b_g$$

Where $\bar{\mathbf{N}}^B$ is the sensor measurement which is subject to a Gaussain measurement noise, μ_m, and a bias term, b_m.

Now, letting the magnetic field act completely through the y component of (3.13) so the quad-rotor will align up with the earths magnetic field along the y axis. Hence, the following relation can be defined:-

$$\begin{bmatrix} \mathbf{N}_x \\ \mathbf{N}_y \\ \mathbf{N}_z \end{bmatrix} = \begin{bmatrix} C_\theta S_\psi \\ S_\phi S_\theta S_\psi + C_\phi C_\psi \\ C_\phi S_\theta S_\psi - S_\phi S_\psi \end{bmatrix} + \begin{bmatrix} \mu_{m-x} \\ \mu_{m-y} \\ \mu_{m-z} \end{bmatrix} + \begin{bmatrix} b_{g-x} \\ b_{g-y} \\ b_{g-z} \end{bmatrix} \tag{3.20}$$

Thus, if both θ and ϕ are known the compass readings can be compensated by means of the filtered θ and ϕ data. This approach was possible as it was decided to filter the θ and ϕ first and then compensate the ψ. An approach similar to the one used on the accelerometer was done when defining ψ, that is, the noise and bias were ignored as they can be filtered before ψ is calculated. Hence, the following equation was derived so ψ could be calculated:-

$$\psi = \arctan\left(\frac{\mathbf{N}_x}{C_\phi C_\theta \mathbf{N}_y - S_\phi C_\theta \mathbf{N}_z - \mathbf{N}_x}\right) \tag{3.21}$$

3.4 GPS Frame Conversion

As one of the goals for this project was GPS navigation by means of way points, which is a method of relating latitude (φ) and longitude (λ) as illustrated in figure 3.2 was required. Thus, this section will deal with the mapping of φ and λ from a fixed frame attached to the earth (denoted Earth Centered Earth Fixed (ECEF) to the NED frame [29].

If the quad-rotor required to move a certain distance in the x, y or z direction,this distance has to be found in the ECEF frame first. The difference between the current location and the desired location must to be calculated. This difference can be found if the φ and λ of the destination is known and current φ and λ the quad-rotor is known. Once this is known, (3.22) can be used to find xyz displacement in the ECEF and this is then mapped to the NED frame using (3.23). Note a similar approach can be used to control the velocity of the quad-rotor in the NED frame.

$$\begin{aligned} N &= \frac{a}{\sqrt{1 - e^2 \sin^2 \varphi}} \\ x &= (N + h)\cos(\varphi)\cos(\lambda) \\ y &= (N + h)\cos(\varphi)\sin(\lambda) \\ z &= [N(1 - e^2) + h]\sin(\varphi) \end{aligned} \tag{3.22}$$

Where:-

h**:** is the height of the quad-rotor from the surface of the planet.

a**:** is the equatorial radius of the earth and is equal to 6,378,137 m

b**:** is the polar radius of the earth and is equal to 6,356,752.3142 m

f**:** is flatting of the earth and is given by the following:- $f = (a - b)/a$

e**:** is the eccentricity of the earth and is defined as follows:- $e = \sqrt{f(2 - f)}$

In order to find out how far the quad-rotor has to go in order to reach the required position[7] (3.22) has to be mapped using the following:-

$$\begin{bmatrix} y_n \\ x_n \\ z_n \end{bmatrix} \triangleq \begin{bmatrix} -S_{\varphi o}C_{\lambda o} & -S_{\varphi o}S_{\lambda o} & C_{\varphi o} \\ -S_{\lambda o} & C_{\lambda o} & 0 \\ -C_{\varphi o}C_{\lambda o} & -C_{\varphi o}S_{\lambda o} & -S_{\varphi o} \end{bmatrix} \begin{bmatrix} x_p - x_o \\ y_p - y_o \\ z_p - z_o \end{bmatrix} \tag{3.23}$$

where:- (x_p, y_p, z_p) is the new point at which the quad-rotor has to move to and (x_o, y_o, z_o) subscript is the current location of the quad-rotor. Similarly the velocity of the quad-rotor can using the heading and speed measurements that the GPS module. These measurements are given by the following equations:-

[7] And hence also control the quad-rotor

$$\begin{aligned} \text{Speed} &= \sqrt{\dot{x_e}^2 + \dot{y_e}^2} \\ \text{Heading} &= \arctan2\left(\frac{\dot{x_e}}{\dot{y_e}}\right) \end{aligned} \tag{3.24}$$

Thus, the rate of change of position can also transformed by means of the following equations.

$$\begin{bmatrix} \dot{y_n} \\ \dot{x_n} \\ \dot{z_n} \end{bmatrix} = \begin{bmatrix} -S_\varphi C_\lambda & -S_\varphi S_\lambda & C_\varphi \\ -S_\lambda & C_\lambda & 0 \\ -C_\varphi C_\lambda & -C_\varphi S_\lambda & -S_\varphi \end{bmatrix} \begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{z_e} \end{bmatrix} \tag{3.25}$$

where:-

$$\begin{aligned} \dot{x_e} &= \sqrt{\frac{(\text{speed}^2)}{1 + \tan(\text{heading})^2}} \\ \dot{y_e} &= \sqrt{(\text{speed})^2 - \dot{x_e}^2} \end{aligned} \tag{3.26}$$

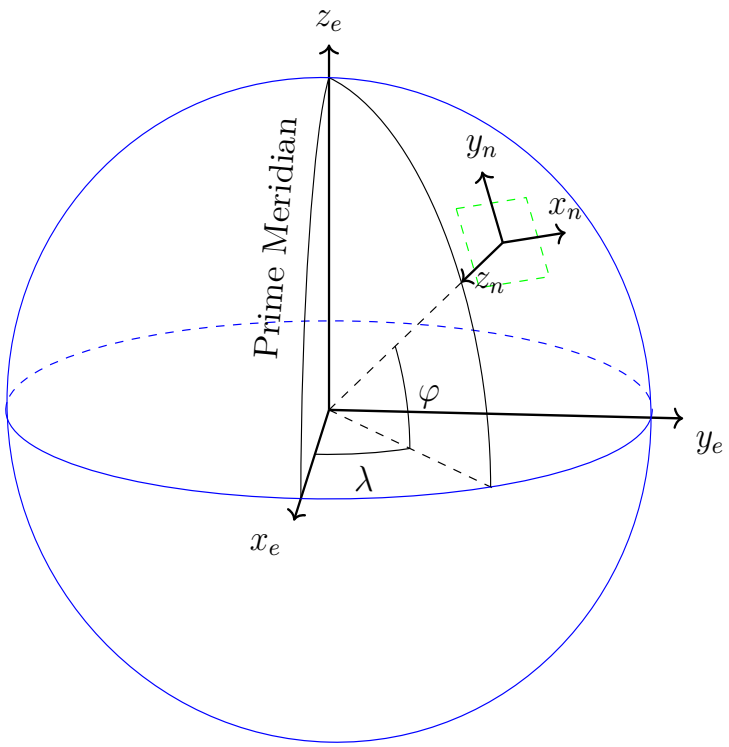


Figure 3.2: Definitions of Various Reference Frames

3.5 Inertial Navigation System

A system which uses Magnetometers, Accelerometers and Gyroscopes to estimate the attitude of a body is often referred to as a Inertial Navigation System (INS). A method of estimating the attitude of quad-rotors has been the focus of substantial amount of research as the attitude data is required for autonomous flight. The attitude of the device dictates the direction in which the quad-rotor flies and thus is required for GPS navigation of a quad-rotor.

As these sensors are not ideal it was required to derive an accurate mathematical model of the quad-rotor and the sensors themselves. As the accelerometer measurements contains linear, angular as well as acceleration due to gravity. This cannot be decoupled easily and hence requires a filter to remove these components. The gyroscope used was not ideal as the measurements tended to drift over time due to temperature. The magnetometer contained non-ideal components as any sources of ferromagnetic material placed close to the device will distort the magnetic field produced by the earth. A method to combine the accelerometer and gyroscope was required to give an adequate estimate of the orientation and was researched in detail. This estimate could then be used with a GPS device in order to control the quad-rotor remotely by means of GPS navigation.

3.5.1 Complementary Filter

The Complementary Filter consists of two filters, a low-pass and a high-pass filter. The input to the low-pass filter is the accelerometer data, since at low accelerations, the accelerometer is considered to approximately measure only acceleration due to gravity. Hence, the orientation can be estimated. The input to the high-pass filter is the gyroscope data since the drift due to the gyroscope is low frequency. The algebraic sum of the outputs of the filters gives the estimate of the orientation.

3.5.2 The Kalman Filter

The Second method investigated was the Kalman Filter, which is much more difficult to design, but returns a more accurate result than the Complementer Filter. The Kalman Filter is based on the statistical properties of the noise in the sensor data, as

well as the noise present in the model of the plant, which is assumed to be Gaussian in nature. Rudolf E. Kálmán first presented it in 1960 when he published his famous paper describing a recursive solution to the discrete-data linear filtering problem [30]

But before such filtering methods are introduced, reasons must be presented for considering such advantaged filtering techniques. As can be seen from figure 3.3 the output from the sensors requires filtering of some form as the current noise is to great to control to allow adequate control of the quad-rotor. But if a low pass filter is used on the accelerometer the phase lag is too great to allow the required control, hence a estimator with less delay is required. Thus, the Complementary Filter and Kalman Filter were investigated.

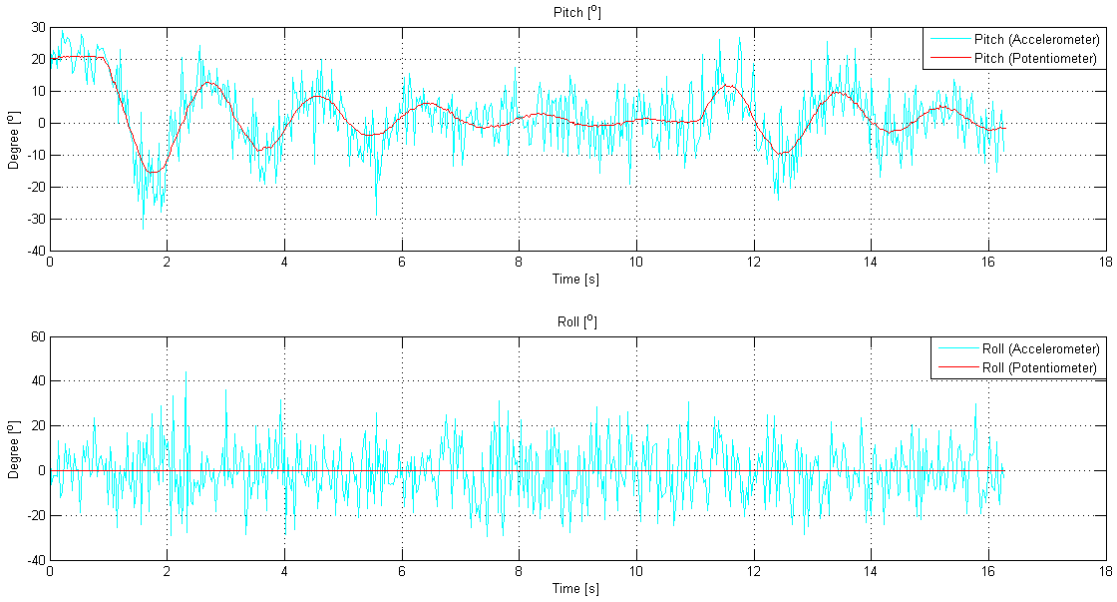


Figure 3.3: Unfiltered Accelerometer Output vs Actual Angle

4. The Complementary Filter

Two different complementary filters were investigated and their responses were modeled in MatLab before they were tested on the Quad-Rotor. This chapter describes in detail these two complementary filters.

4.1 Complementary Filter

The Complementary filter gets its name from the manner in which the high-pass and low-pass filters which make up the Complementary filter are chosen. Hence, a pair of filters are called complementary filters if their transfer functions sum to one at all frequencies in a complex sense, i.e. the phase is zero and the magnitude is one as seen in following:-

$$G_1(s) + G_2(s) = 1 \tag{4.1}$$

Figure 4.1 is a block diagram representation of a first order complementary filter. As can be seen from the figure, the accelerometer data is low pass filtered while the gyroscope data is high pass filtered. The filters used in this report are augmented forms of the filters presented in [31, 32]. The first filter that was investigated first and is depicted in figure 4.1.

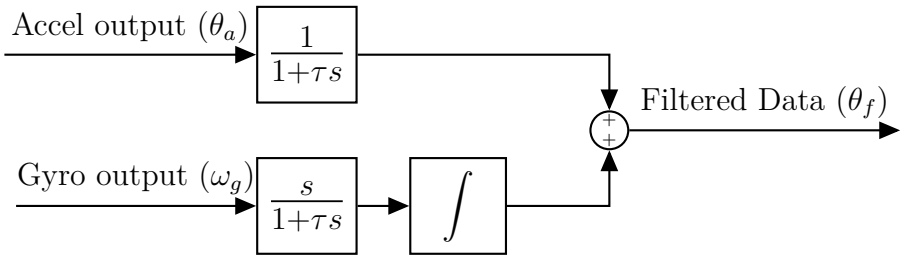


Figure 4.1: Block Diagram of First Order Complementary Filter

4.1.1 First Order Complementary Filter

The first order complementary filter presented in [31] is best represented by (4.2)[1]. As can be seen from (4.2) there is only one tuning parameter (τ), meaning the filter is easy to design. Therefore there is a trade off between ease of use and versatility with the first order filter.

The first order filter gave adequate results, a second order complementary filter will also be presented later which has two tuning parameters, which is a more versatile version of the Complementary Filter.

$$\theta_f = \underbrace{\frac{1}{1+\tau s}}_{G_1(s)}\theta_a + \underbrace{\frac{\tau s}{1+\tau s}}_{G_2(s)}\frac{1}{s}\omega_g \tag{4.2}$$

As can be seen from Figure 4.2 the filters presented in (4.2) are indeed complementary.

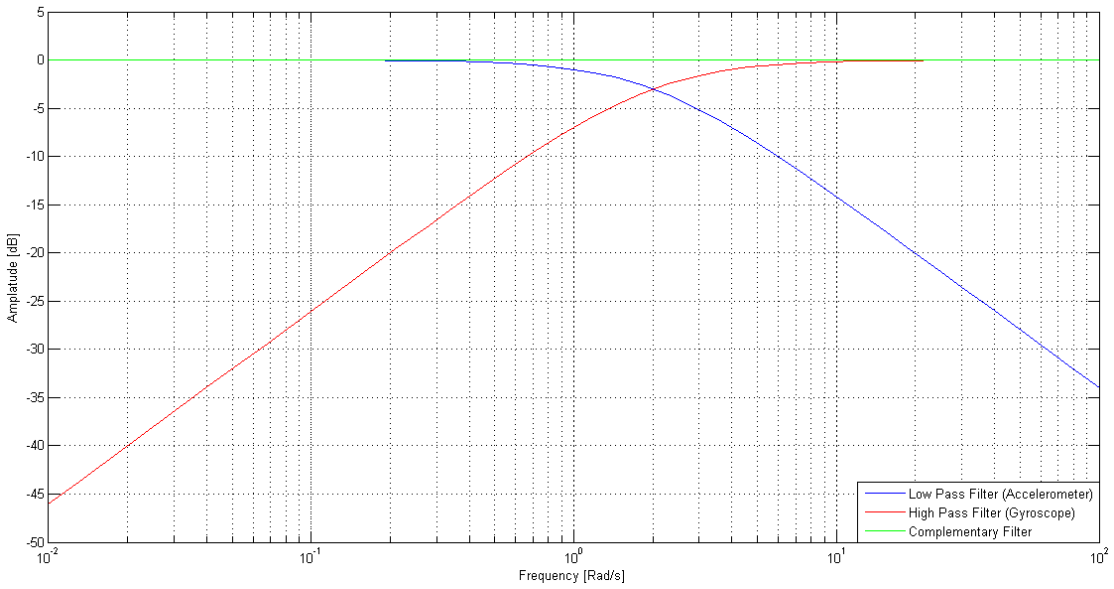


Figure 4.2: Bode plot of first order complementary filter

Note as the orientation required for the control of the quad-rotor is in the NED frame and as the sensor readings were in the Body Frame (B) frame, a mapping was required to ensure the correct angles. These mappings which were presented in (3.17), (3.19) and(4.2). These mappings allowed the high frequency components to be filtered so that an adequate estimation of the orientation could be achieved by means of a

[1]Note it is assumed that the sensors have ideal transfer functions i.e $H_a(s) = H_g(s) = 1$, where $H_a(s)$ & $H_g(s)$ are the transfer functions of the accelerometer and the gyroscope respectively

high-pass on gyroscope data:-

$$\dot{\phi}_{hp} = \omega_x + S_\phi T_\theta \omega_y + C_\phi T_\theta \omega_z - \frac{\phi_{hp}}{\tau} \tag{4.3}$$

$$\dot{\theta}_{hp} = C_\phi \omega_y - S_\phi \omega_z - \frac{\theta_{hp}}{\tau} \tag{4.4}$$

By a similar mapping system the following low frequency components can be derived by means of a low-pass filter on the accelerometer:-

$$\dot{\phi}_{lp} = \frac{1}{\tau}\left[\arctan\left(\frac{a_y}{a_y}\right) - \phi_{lp}\right] \tag{4.5}$$

$$\dot{\theta}_{lp} = \frac{1}{\tau}\left[\arctan\left(\frac{-a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right) - \theta_{lp}\right] \tag{4.6}$$

4.1.2 Difference Equations for first order filter

Equation 4.2 was emulated into the digital domain using Tustin's approximation to integration. Tustin's method was chosen as it ensures a stable mapping into the discrete domain (maps to inside the unit circle) if the continues function is stable.

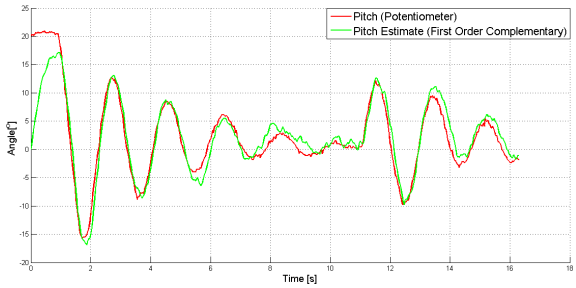
$$\begin{aligned}\theta_f[k] = \frac{1}{T_s + 2\tau}(T_s(\theta_a[k] + \theta_a[k-1] + \tau(\omega_g[k] + \omega_g[k-1])) \\ - (T_s - 2\tau)\theta_f[k-1])\end{aligned} \tag{4.7}$$

Equation 4.7 was implemented on the micro-controller without the frame adjustments stated in (3.16) and (3.19). These reductions were possible as the pitch (θ) and roll (ϕ) angles were limited to $\pm 30^o$, these limitations allowed the trigonometric functions presented in (3.19) and (3.16) to be modeled as unity and θ or ϕ for the *cos* and *sin* functions respectively. This was employed to reduce the sampling time of the critical control loops and can be easily added again if a faster micro-controller with a hardware Floating Point Unit (FPU) [2] is used for the project in the future.

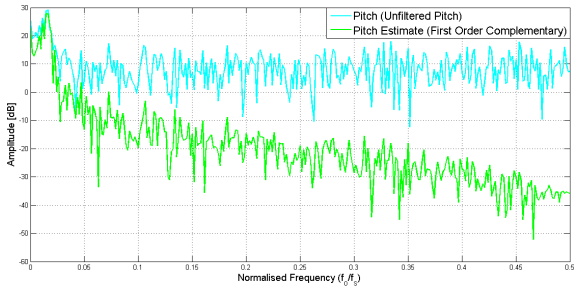
The first order filter shown above was implemented both in Simulink and on the micro-controller along with the attitude controller. The filter was tuned in a similar method presented in algorithm 1 where the phase delay and Root Mean Square Error

[2] A micro-controller that has these requirements is the *Tiva-C LaunchPad* which features a ARM Cortex-M4F which has a hardware FPU

(RMSE) weightings were adjusted until an adequate result was reached. Figure 4.3 compares the estimated pitch angle to the actual value. The results are adequate, but the gyroscope drift has not been fully eliminated and so another method of estimating the attitude of the quad-rotor was investigated. It was first decided to investigate a second order Complementary Filter.



(a) Time domain response of a first order Complementary Filter

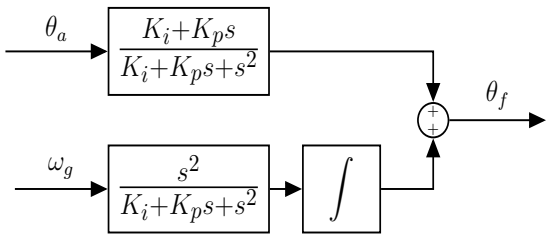


(b) Frequency domain response of a first order Complementary Filter

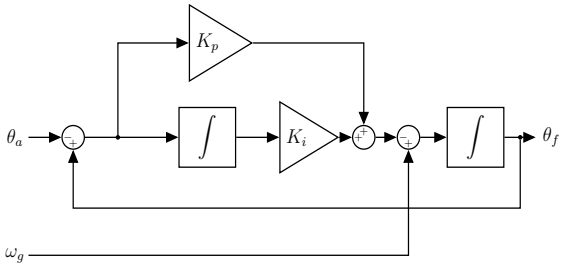
Figure 4.3: Response of a first order complementary filter for $\tau = 0.5$ and $T_s = 30$ ms

4.1.3 Second Order Complementary Filter

As the drift on the gyroscope was still present after implementing the first order complementary filter on the micro-controller it was decided to implement a second order complementary filter similar to the one researched by Mahony & Madgwick [32]. The filter presented in this section has two tuning parameters K_p & K_i, this means that this filter is more versatile than the first order complementary filter which has only one tunning parameter τ.



(a) Block Diagram of Second Order Complementary Filter



(b) Feedback Block Diagram of Second Order Complementary Filter

Figure 4.4: Second Order Complementary Filter Block Diagrams

The second order complementary filter described in figure 4.4 can be defined over

all frequencies by (4.8):-

$$\theta_f = \underbrace{\frac{K_i + K_p s}{K_i + K_p s + s^2}}_{G_1(s)} \theta_a + \underbrace{\frac{s^2}{K_i + K_p s + s^2}}_{G_2(s)} \frac{1}{s} \omega_g \tag{4.8}$$

As can be seen from Figure 4.5 the filters presented in (4.8) are complementary.

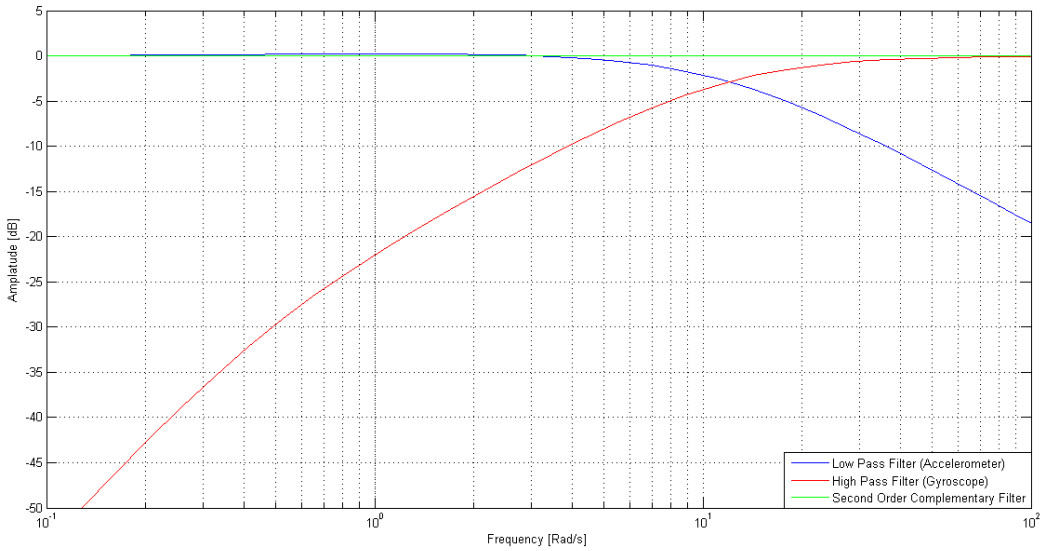


Figure 4.5: Bode plot of a common second order complementary filter

Obviously, and not unexpectedly, this complementary filter is made from 2nd order filters. Note that the filter acting on the acceleration data actually consists of a low-pass plus a band-pass filter.

This result has interesting consequences. As the filters are 2nd order, the frequency response of the acceleration and gyroscope filters are characterized by the resonance frequency and damping factor

$$\omega_0 = \sqrt{K_i} \qquad \xi = \frac{K_p}{2\sqrt{K_i}} \tag{4.9}$$

The damping factor determines the overshoot at the resonance frequency. For manual tuning of this filter a flat frequency response can be achieved by setting $\xi \geq 1$ in (4.9). After doing this, one will get the following tuning criteria.

$$K_i \leq \frac{1}{4} K_p^2 \tag{4.10}$$

The results of this type of tuning can be seen in Figure 4.6 and estimated the orientation of the quad-rotor to a certain degree, but as a whole was unsatisfactory. As a result two other methods of auto-tuning the filter were investigated, one which used a least squares and another which used a grid search. Before a grid based tuning approach could be implemented a discrete time version of the filter had to be created.

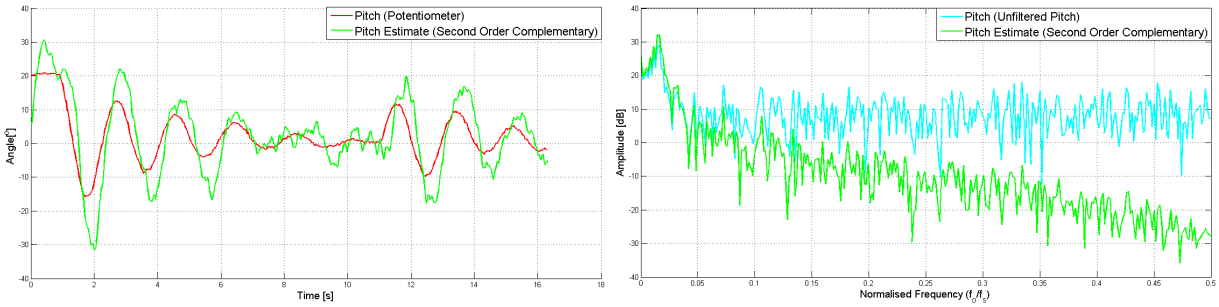
Thus, to model the filter in MatLab $G_1(s)$ and $G_2(s)$ were arranged as follows:-

$$\chi_{hp} = \frac{1}{s}\left[\omega_g - \chi_{hp}\left(\frac{K_i}{s} + K_p\right)\right] \tag{4.11}$$

$$\chi_{lp} = \frac{1}{s}\left[(\theta_a - \chi_{lp})\left(\frac{K_i}{s} + K_p\right)\right] \tag{4.12}$$

Note (4.11) and (4.12) can be combined to yield the following filter (as $\chi_{hp}+\chi_{lp} = \theta_f$), which can also be modeled in MatLab.

$$\theta_f = \frac{1}{s}\left[\omega_g + \left(\frac{K_i}{s} + K_p\right)(\theta_f - \theta_a)\right] \tag{4.13}$$



(a) Time domain response of a Second order Complementary Filter

(b) Frequency domain response of a Second order Complementary Filter

Figure 4.6: Response of a Second order complementary filter for $K_i = 25$, $K_p = 7$ and $T_s = 30$ ms

4.1.4 Difference Equations for second order filter

In order to implement the second order complementary filter on a micro-controller (4.11)and (4.12) were emulated into the digital domain using Tustin's method, which yielded the following equations:-

$$\chi_{hp}[k] = \frac{1}{\eta + 4}\left(2T_s(\omega_g[k] - \omega_g[k-2]) - (\Gamma + 4)\chi_{hp}[k-2] - (\xi - 8)\chi_{hp}[k-1]\right) \tag{4.14}$$

$$\chi_{lp}[k] = \frac{1}{\eta + 4}\left(\Gamma\theta_a[k-2] + \xi\theta_a[k-1] + \eta\theta_a[k] - (\Gamma + 4)\chi_{lp}[k-2] - (\xi - 8)\chi_{lp}[k-1]\right) \tag{4.15}$$

where:-

$$\eta = K_i{T_s}^2 + 2K_pT_s; \quad \Gamma = K_i{T_s}^2 - 2K_pT_s; \quad \xi = 2K_i{T_s}^2$$

Before auto-tuning of the filter was implemented (4.14) and (4.15) where combined to give (4.16). This was the equation that was used to tune the filter. Note (4.13) could also be used to tune the filter, but when the filter is emulated there are distortions introduced by the emulation process.

$$\theta_f[k] = \frac{1}{\eta + 4}(\Gamma\theta_a[k-2] + \xi\theta_a[k-1] + \eta\theta_a[k] + 2T_s(\omega_g[k] - \omega_g[k-2]) - (\Gamma + 4)\theta_f[k-2] - (\xi - 8)\theta_f[k-1]) \tag{4.16}$$

4.2 Tuning of the Complementary Filter using Least Squares

As a manual approach to tuning the second order filter yielded inadequate results, other methods of tuning the filter were investigated. One of the more useful approaches to tuning the filter was the least squares approach. Using (4.13) one can tune the filter in the continuous domain and then emulate it across. This can be done by making the following assumption:-

$$\theta_f = \theta_p \tag{4.17}$$

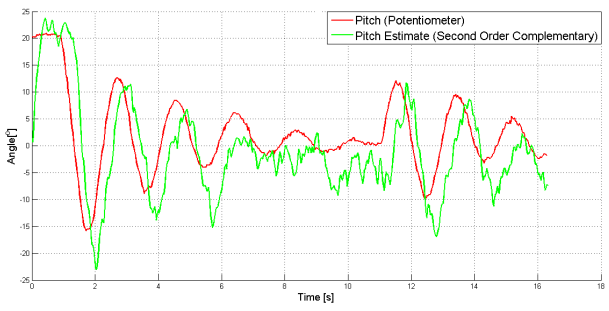
Where θ_p is the angle of the system as given by the potentiometer. In order to accomplish this access to the rate of change of potentiometer angle as denoted as $\dot{\theta_p}$. Hence, a method of tuning the filter as follows:-

$$\begin{aligned} \theta_p &= \frac{1}{s}\left[\omega_g - \left(K_p + \frac{K_i}{s}\right)(\theta_p - \theta_a)\right] \\ \dot{\theta_p} &= \left[\omega_g - \left(K_p + \frac{K_i}{s}\right)(\theta_p - \theta_a)\right] \\ \underbrace{\dot{\theta_p} - \omega_g}_{B_f} &= \underbrace{\left[(\omega_g - \theta_p) + \frac{(\omega_g - \theta_p)}{s}\right]}_{A_f} \begin{bmatrix} K_p \\ K_i \end{bmatrix} \end{aligned} \tag{4.18}$$

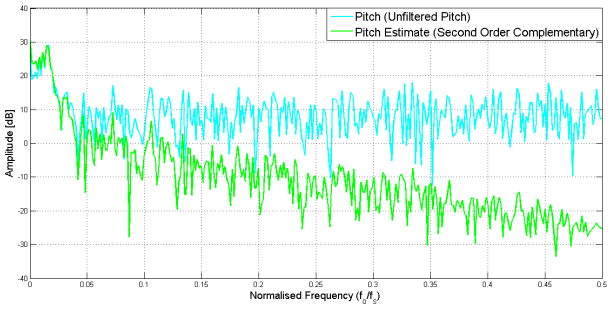
Thus, using the equation developed in (4.18) one can tune a continuous time complementary filter of this form using the following:-

$$\begin{bmatrix} K_p \\ K_i \end{bmatrix} = (A_f^\intercal A_f)^{-1} A_f^\intercal B_f \tag{4.19}$$

Note the method of tuning the filter as shown in (4.19) was done without a $\dot{\theta_p}$ term. Instead a value was set for $(\omega_g - \dot{\theta_p})$ which was a valid approximation as the output of the gyroscope was low pass filtered on-chip with a first order filter which has a cutoff frequency of 100 *Hz*. The results of this tuning method can be seen in Figure 4.7



(a) Time domain response of a Second order Complementary Filter



(b) Frequency domain response of a Second order Complementary Filter

Figure 4.7: Response of a Second order complementary filter for $K_i = 1.3265$, $K_p = 2.79322$ and $T_s = 30$ ms

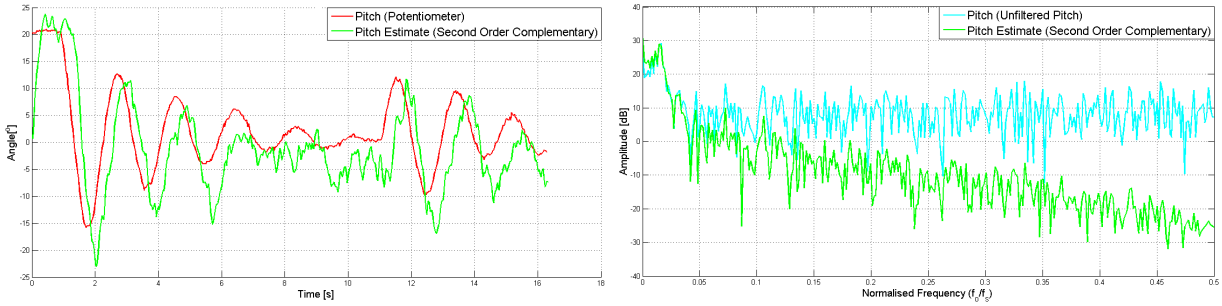
4.3 Self Tunning Algorithm for a Complementary Filter

As the "actual" angle θ_p of the quad-rotor was known when tuning the filter it was decided to do a search using a RMSE (4.20). A cross-correlation was done to find the delay between the filtered angle and the "actual" angle and remove when checking the RMSE term. This had the best results for the tuned filter and a trade off between lag[3] and filtering could be set. Figure 4.8 proves that drift on the gyroscope has been removed, but the phase delay of the filtered signal has increased dramatical.

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n}(\theta_f - \theta_p)^2}{n}} \tag{4.20}$$

The cross-correlation function is defined in (4.21) and is commonly used in signal processing to find the measure of time-lag between two signals.

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m]g[m+n] \tag{4.21}$$



(a) Time domain response of a Second order Complementary Filter

(b) Frequency domain response of a Second order Complementary Filter

Figure 4.8: Response of a Second order complementary filter for $K_i = 1.2802$, $K_p = 7.1919$ and $T_s = 30$ ms

As can be seen from Figure 4.8 the second order Complementary Filter removes the bias from the gyroscope, but there is a large delay introduced. Note there is less noise rejection in the second order filter. Hence, another filter was investigated which will be presented next.

[3]Note with this filter the lag can be set so it does not exceed the maximum allowable phase delay

Algorithm 1 Auto Tuning a Complementary Filter

procedure GRID SEARCH TUNING OF A COMPLEMENTARY FILTER
    Set max value for K_p, K_i, an RMSE value and the associated tolerance for the grid.

    for length($K_{p(it)}$) **do** ▷ $K_{p(it)}$ an array of K_p values to test

        for length($K_{i(it)}$ **do** ▷ $K_{i(it)}$ an array of K_i values to test
            Calculate θ_f the filtered output using (4.16)
            Calculate Signal Delay of the Filtered output by means of equation (4.21)
                by using θ_f and θ_p as it's inputs.
            Remove delay in filtered data by removing samples equal to the value
                returned by the previous line of code.
            Computer the RMSE value of the filtered signal using equation (4.20)
                and θ_f & θ_p as it's inputs.

            if (RMSE < Previous Error) **then**
                $K_p = K_{p(it)}$(i);
                $K_i = K_{i(it)}$ (j);
                Previous Error = RMSE;

    return K_p & K_i

5. The Kalman Filter

In order to design a optimal controller for the quad-rotor a State-Estimator is required. The reason for this is simple: one doesn't have access to all the system states. For example one can't measure the angle velocity of the propellers as the Quad-rotor doesn't have a hall sensor. Due to the use of inexpensive sensors a stochastic filter or full non-linear model of the sensors had to be developed. Thus, the Kalman Filter was chosen as it is a ideal filter and is capable of estimating angle and rate of change of angle with great accuracy. [33].

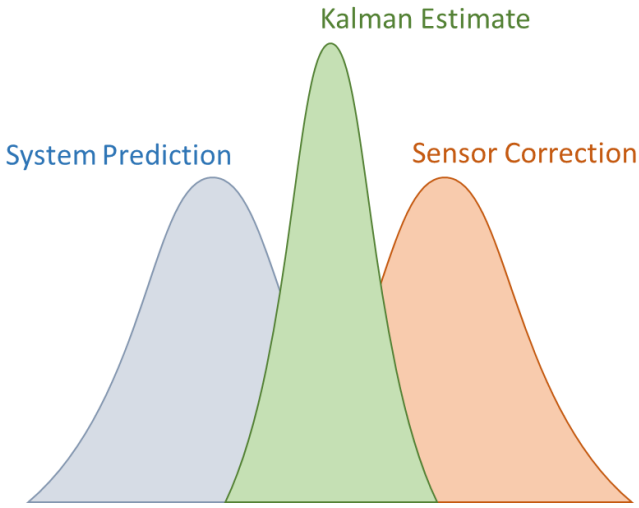


Figure 5.1: Gaussian representation of the Kalman Filter

When a Kalman Filter is being implemented the noise present in the system and senors has to be to Gaussian zero-mean, which is usually denoted as w and v respectively. If this criteria is met the filter will yield a new linear system model which is free of noise. The Filter works by adding a prediction produced by the plant model from known inputs, and a correction from the data measured by sensors. This process can be best visualized by figure 5.1, where the plant estimate and the sensor estimate have been fused together to achieve a best guess of the actual state. The Kalman Filter is similar to the Luenberger Observer [1], but the observer gains are chosen in an optimal manner. The Kalman Gain K defines by how much the model estimate has to be

[1] An Observer is used to estimate unmeasurable states by means of a system model and some measurable outputs.

corrected by the sensor measurements. Hence, if the noise in the sensors is greater than the noise in the model the senor measurements are trusted more to estimate the required states, and thus assigned a greater weighting. If the noise present in the system states is greater than the noise present in the sensor, then sensor readings will be trusted more to estimate the required states. Thus, in short, the Kalman Filter is a state observer for the stochastic case. In order to apply this filter one must define the system in a discrete liner state space form.

$$\begin{aligned} \underline{x}_{k+1} &= \Phi \underline{x}_k + B_d \underline{u}_k + \Gamma \underline{w}_k \\ \underline{z}_k &= C_d \underline{x}_k + \underline{v}_k \end{aligned} \tag{5.1}$$

The filter , whose linear model is described in 5.1, can be seen as a Linear Time-Invariant (LTI) system operating in parallel to the real physical system in order to generate an optimal estimate $\underline{\hat{x}}$ of all states whereas compensating for the noise in the plant [2]. The filter removes/reduces the noise present in the plant by minimizing the error covariance matrix of the estimated states. The covariance matrix is commonly denoted as follows:-

$$\mathbf{P}_k = E[\underline{\tilde{x}_k}\ \underline{\tilde{x}_k}^\top] \tag{5.2}$$

where $\underline{\tilde{x}}_k$ is the error in the estimated states.

Note an expression for the $\mathbf{P}_k$ matrix must be developed from the initial system to begin the filter derivation. Therefore from (5.1), the best state estimate can be achieved :-

$$\underline{\hat{x}}_{k+1|k} = \Phi_k \underline{\hat{x}}_{k|k} + B\underline{u}_k$$

5.1 Development of the Kalman Filter

As seen from figure 5.2 the Kalman Filter is made up of two stages, a predict stage and a correct stage. The predict stage generates an estimate of the required states by means of an ideal system model whose inputs are known. After the estimate is found

[2]Note superscript d notation is dropped for simplicity, E.g ($B_d = B$).

then the covariance error is measured (this is the aspect to minimize). The covariance error indicates by how much the state estimates differ from the ideal signal. After this the correct stage begins and the Kalman gain K is calculated by means of the covariance error and the covariance of the sensor data. Next, the initial estimate is corrected by means of sensor data which is scaled appropriately by the Kalman gain K. Finally the covariance error is corrected by means of the Kalman gain K and after which the Kalman starts, hence the recursive aspect of the Kalman Filter.

Thus, implement the Kalman filter a recursive algorithm needs to developed. In order to accomplish this assume access to the previous estimate $\hat{x}_{k|k}$, as the initial value $\hat{x}_{0|0}$ it is possible to define a recursive. If the previous estimate is known any future value can be calculated, but this can only be done if the noise signal is zero mean that is, $\mathrm{E}[w_k] = \mathrm{E}[v_k] = 0$. Hence, the best estimate can be derived from (5.1) by taking the mean value of $\underline{w}_k$, doing so will yield the following:-

$$\underline{\hat{x}}_{k+1|k} = \Phi_k \underline{\hat{x}}_{k|k} + B\underline{u}_k \tag{5.3}$$

Therefore the one-step-ahead *Predictive measurement* can be defined as follows:-

$$\hat{z}_{k+1} = C\hat{x}_{k+1|k}$$

At time (k+1) , one can measure z_{k+1} from the plant, thus define the *Predict error* $\tilde{z}$ as follows:-

$$\underline{\tilde{z}}_{k+1} = \underline{z}_{k+1} - \underline{\hat{z}}_{k+1} \tag{5.4}$$

Thus, to improve the state estimate one can add some proportion of the prediction error (5.4) to each element of the state vector to drive the prediction error to zero as follows:-

$$\underline{\hat{x}}_{k+1|k+1} = \Phi \underline{\hat{x}}_{k|k} + B\underline{u}_k + K\underline{\tilde{z}}_k \tag{5.5}$$

Knowing (5.3), (5.4) and (5.5) and filling in one gets the following equation:-

$$\underline{\hat{x}}_{k+1|k+1} = \Phi \underline{\hat{x}}_{k|k} + B\underline{u}_k + K[\underline{z}_{k+1} - C(\Phi \underline{\hat{x}}_{k|k} + B\underline{u}_k)] \tag{5.6}$$

Thus matrix/variables are grouped together for ease of computation and yields the

following:-

$$\underline{\hat{x}}_{k+1|k+1} = [I - KC][\Phi\underline{\hat{x}}_{k|k} + B\underline{u}_k] + Kz_{k+1} \tag{5.7}$$

Equation 5.7 can be written as follows.

$$\underline{\hat{x}}_{k+1|k+1} = F\underline{\hat{x}}_{k|k} + H\underline{u}_k + Kz_{k+1} \tag{5.8}$$

Where in (5.8) $F = [I - KC]\Phi$; $H = [I - KC]B$. Thus each matrix is made up of a mix of prediction and correction values.

It was assumed in this project that the separate noise signals were independent of each other, thus allowing $\mathbf{R}$ and $\mathbf{Q}$ to be covariance matrices. A large value for $\mathbf{R}$ implies a lot of noise is present in the measurement data and thus more emphasis is placed in the predictions. A large value of $\mathbf{Q}$ implies that there is more noise in the states than in the measurement data and thus the measurement data is followed more closely. As $\mathbf{P}_k$ is to be minimized, an expression for the estimation of the covariance error can be obtained as follows ("Predict" stage):-

$$\mathbf{P}_k{}^* = \Phi\mathbf{P}_{k-1}\Phi^\intercal + \Gamma\mathbf{Q}\Gamma^\intercal \tag{5.9}$$

Thus the Kalman gain K can be defined as follows. Note the Kalman Gain K must be updated at each iteration to correct the measurement data so as to obtain an optimal estimate of the required states. The equation governing this update process is as follows ("Correct" stage) :-

$$K_k = \mathbf{P}_k{}^* C^\intercal (C\mathbf{P}_k{}^* C^\intercal + \mathbf{R})^{-1} \tag{5.10}$$

Finally the covariance error can be corrected as follows:-

$$\mathbf{P}_k = (I - K_k C)\mathbf{P}_k{}^* \tag{5.11}$$

From (5.9) the performance of this filter depends heavily upon the accuracy of $\mathbf{Q}$ and $\mathbf{R}$. Note $\mathbf{R}$ can often be intelligently estimated from knowledge of the system under control. The derivation of $\mathbf{Q}$ is a problem as often very little real information will be known about the noise present in the states. Therefore $\mathbf{Q}$ is often guessed and $\mathbf{R}$ and $\mathbf{Q}$ are tuned together to get an adequate result for the control of the device.

5.1.1 Kalman Filter Operation

As described in the previous sections, the Kalman Filter extracts state estimates from noisy signals. This can be done by obtaining statistical information of the noise present in both the plant and the measurement data, knowing this the filter can optimally estimate the state of interest. This information is given in form of the plant and measurement noise covariances, $\mathbf{R}$ and $\mathbf{Q}$. The algorithm that the filter performs can be described as follows:

First, the $\mathbf{Q}$ and $\mathbf{R}$ covariance matrices are calculated through trial and error. An initial state estimate $\underline{\hat{x}}_{0|0}$ and its error covariance $\mathbf{P}_0$ are entered. The Kalman Gain matrix, K is computed based on $\mathbf{P}$, $\mathbf{R}$ and the transition matrix Φ. The estimate is then updated with the current measurement data and after which a new error covariance computed. The algorithm then starts again from the beginning. The output from the filter is an estimate of the states and the error covariance matrix, $\mathbf{P}$. Figure 5.2, shown below is a flow diagram of the Kalman Filter algorithm.

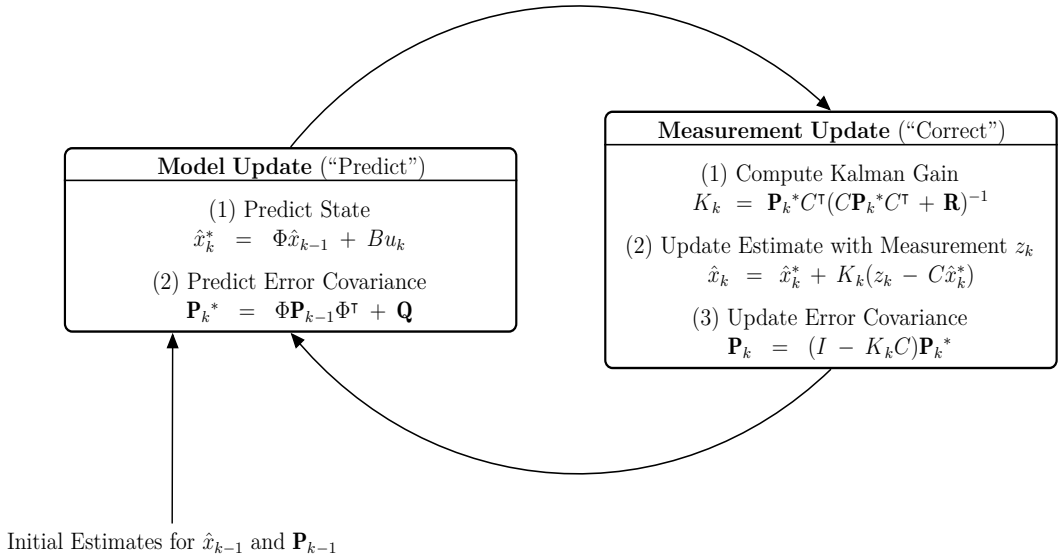


Figure 5.2: Kalman Filter flow diagram

5.2 Kalman Filter Used in Project

There were three Kalman filters developed for this report; a steady state Kalman Filter whose Kalman Gains remains constant over time, a Kinematic Kalman Filter which is based on the Kalman filter presented in Ersson and Hu [34] and an Extended Kalman

Filter. Note all Kalman Filters investigated in this project will be developed in this section, while the implementation and tuning will be presented in Chapter 6

5.2.1 The Static Kalman Filter

The Static Kalman Filter, which will be developed in the this sections uses a constant K which is found by means of the Kalman Filter Algorithm presented in 5.2. In practice, the time-varying Kalman gain K tends towards a steady value [35]. Hence, when the Kalman Filter is used on the quad-rotor the Kalman Gain K will converge to a constant value, thus, a constant Kalman Gain K can be used to estimate the required states. As the Static Kalman Filter uses a constant K term, the observer produced by this method is a optimal Luenberger Observer[3]. The model used in the Kalman Filter for the pitch, roll and yaw axes are the same ones developed in (3.3), (3.4) and (3.8). Note as these are continuous equations, a mapping of the continuous time model to a discrete model has to take place which was done by means of the Matrix Exponential. After the discrete Static Kalman Filter was developed the K terms shown in (5.12) was adjusted by methods presented in Chapter 6 until an optimal response was achieved.

5.2.2 Derivation of state space equations

As the Kalman Filter is required run on a micro-controller, the Kalman Filter expressed in 5.8 has to be expressed in its most compact format so as to reduce computational effort and improve its accessibility. These requirement lead to the following definition for the Static Kalman Filter:-

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}_{k} + \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} u_k + \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}_{k+1} \tag{5.12}$$

For the quad-rotor the state x_1 is angular velocity α and the state x_2 is angular acceleration $\dot{\alpha}$

5.2.3 The Kinematic Kalman Filter

Unlike the Kalman Filter presented in the previous section, the Kalman filter being developed in the following section uses a pure kinematic relationship, thus a model of

[3]This Static Kalman Gain can be found by using the *lqg* commend in MatLab

the plant is not required. This kinematic relationship uses the current gyroscope data to relate the derivative of the rotation matrix to the rotation matrix and is defined as follows:-

$$\dot{Rot} = \underbrace{\begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix}}_{S(\omega_g)} Rot \tag{5.13}$$

Where:-

$S(\omega_g)$**:** is a skewed matrix that contains the set of all proper orthogonal matrices and is defined as $S(\omega_g) = \mathbf{R}|\mathbf{R}\varepsilon\mathbf{R^{3x3}}, \mathbf{R}^\intercal\mathbf{R} = \mathbf{R}\mathbf{R}^\intercal = \mathbf{I}$

ω_x**,** ω_y**,** ω_z**:** are the roll,pitch and yaw angular rates, respectively.

These rates are measured using the gyroscopes onboard the quad-rotor, in the Inertial Measurement Unit (IMU).

Using (5.13), the following continuous state space equations presented in (5.14) can be used to represent the system:-

$$\begin{aligned} \dot{\underline{x}} &= S(\omega_g)\underline{x} \\ \underline{y} &= \underline{x} \end{aligned} \tag{5.14}$$

However, (5.14) is a continuous state space equation. This must be converted to the discrete version since the Kalman Filter must be implemented in discrete form, which is achieved as follows:-

$$\begin{aligned} \underline{x}_k &= \Phi_k x_{k-1} \\ \underline{y}_k &= \underline{x}_k \end{aligned} \tag{5.15}$$

The transformation from the continuous to discrete form is shown in appendix B. $\Phi_k = e^{S(\omega_g)T_s}$, where T_s is the sampling time of the system. Since *Rot* is a rotation matrix, and $\underline{x}$ is a column of Φ (for reasons defined in Section 3.3.1). In this filter the dynamic component of the accelerometer measurement is considered to be a disturbance and is

assumed to be Gaussian. It is therefore treated as noise and must be removed so as to acquire an accurate estimate. After the noise has been removed from the accelerometer output (3.17) can be used to estimate the attitude of the quad-rotor.

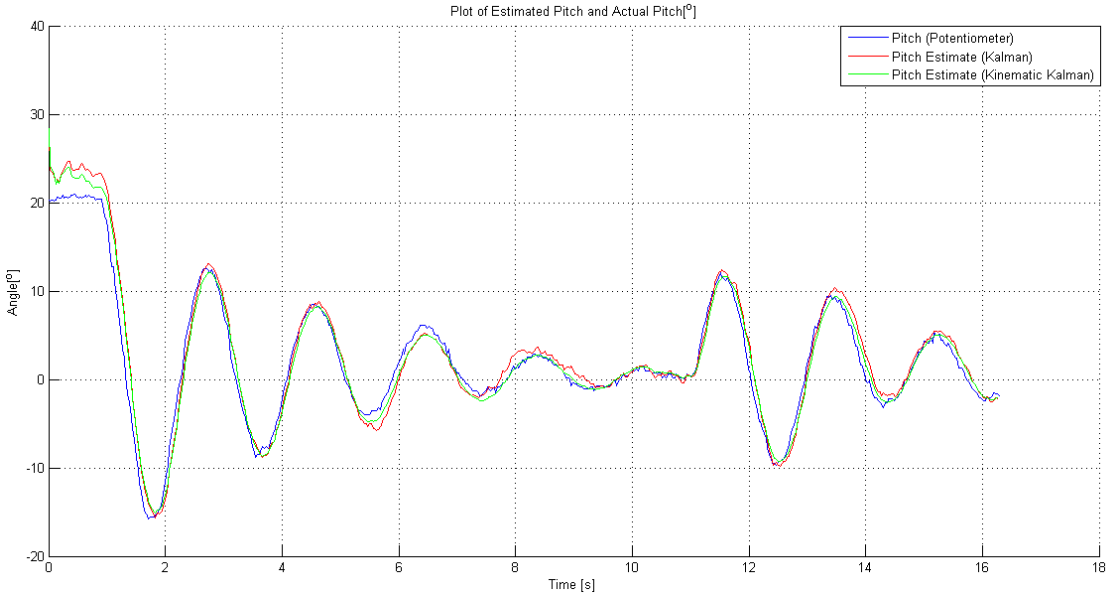


Figure 5.3: Plot of Kalman Filter Estimate, Kinematic Kalman Filter Estimate and Actual Angle of the device

5.2.4 The Extended Kalman Filter

As the gyroscope measurement function $h(x_k, v_k)$ is highly non-linear it was decided that it was worth while looking at a more advanced Kalman Filter which is capable of dealing with non-linear systems: thus the Extended Kalman Filter was explored (see section 3.3.2 for more details on the gyroscope measurement function). Also, as a GPS module is required to implement positional controller on the quad-rotor it was the natural choice to investigate the EKF as it is the de-facto filter used in such systems [36].

As the EKF filters non-linear functions by means of linearizion of the system around the current estimate, a method of finding the partial derivatives of both the process and measurement functions is required. Hence, if the process is governed by the *non-linear* stochastic difference equation:-

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{5.16}$$

and the measurement model by the following stochastic difference equation:-

$$z_k = h(x_k, v_k) \tag{5.17}$$

where the random variables w and v again represent the process and measurement noise.

In practice individual values of the individual w and $v$change at each time step. However, one can approximate the state and measurement without them as follows:-

$$x_k^* = f(\hat{x}_{k-1}, u_k, 0) \tag{5.18}$$

Next the measurement model can be defined by means of the following stochastic difference equation:-

$$z_k = h(x_k^*, 0) \tag{5.19}$$

It is important to note that a fundamental flaw of the EKF is that the distributions of the various random variables are no longer normal after undergoing their respective non-linear transformations. The EKF is simply an ad-hoc estimator. The EKF only approximates the optimality of Bayes' rule by linearization.

Equation (5.18) is linearized around the control input u_k and the previous estimate using (20.9) and the measurement function (20.8) is linearized around the x_k^* using (20.10)

$$A_k = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{array} \right] \Bigg|_{\hat{x}_k, u_k} \tag{5.20}$$

$$C_k = \left[\begin{array}{cccc} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n} \end{array} \right] \Bigg|_{\hat{x}_k^*} \tag{5.21}$$

Note the above Jacobian can be approximated at each stage on a micro-controller by using the Cauchy's integral formula which is defined as follows [37]:-

$$f^{(n)}(z) = \frac{n!}{2\pi i} \oint_{\gamma} \frac{f(\kappa)}{(\kappa - z)^{n+1}} d\kappa \tag{5.22}$$

To implement (20.11) on a micro-controller it must be approximated as follows:-

$$f^{(n)}(z) \approx \frac{n!}{mh} \sum_{j=0}^{m-1} \frac{f(z + he^{i\frac{2\pi j}{m}})}{e^{i\frac{2\pi jn}{m}}} \tag{5.23}$$

The derivation of a complex-step derivative (first partial derivative) approximation is done by an approximation of a non-linear function with a complex variable using the Taylor's series expansion.

$$f(x + ih) = f(x) + ihf^{'}(x) - h^2 \frac{f^{''}(x)}{2!} - ih^3 \frac{f^{'''}(x)}{3!} + h^4 \frac{f^4(x)}{4!} + \dots \tag{5.24}$$

Now taking only the imaginary parts of both sides gives

$$\mathrm{Im}[f(x + ih)] = hf^{'}(x) - h^3 \frac{f^{'''}(x)}{3!} + \dots \tag{5.25}$$

Dividing by h, rearranging and assuming terms higher than h^2 can be ignored since the interval h can be chosen up to the precision of the machine (smallest number the machine can produce) and thus (20.14) can be approximated as follows:-

$$f^{'}(x) = \mathrm{Im}[f(x + ih)]/h \tag{5.26}$$

As (20.15) is not a function of differences it is more accurate than standard finite difference and more importantly partial derivative can be calculated on a micro-controller using (20.15).

As a method of approximating the Jacobian matrix has been presented in (20.15), it is now possible to implement the EKF on a micro-controller. As the computational capabilities of the micro-controller used in this project were insufficient of implementing an EKF it was decided to only simulate the filter. The EKF was implemented in MatLab using the algorithm presented in figure 5.4, and the filtered results can be seen in Figure 5.5. As can be seen from Figure 5.5 produces the best estimate of the attitude of the quad-rotor; this means an EKF would be the ideal filter for both attitude

estimation of the quad-rotor as well as GPS position estimation of the quad-rotor.

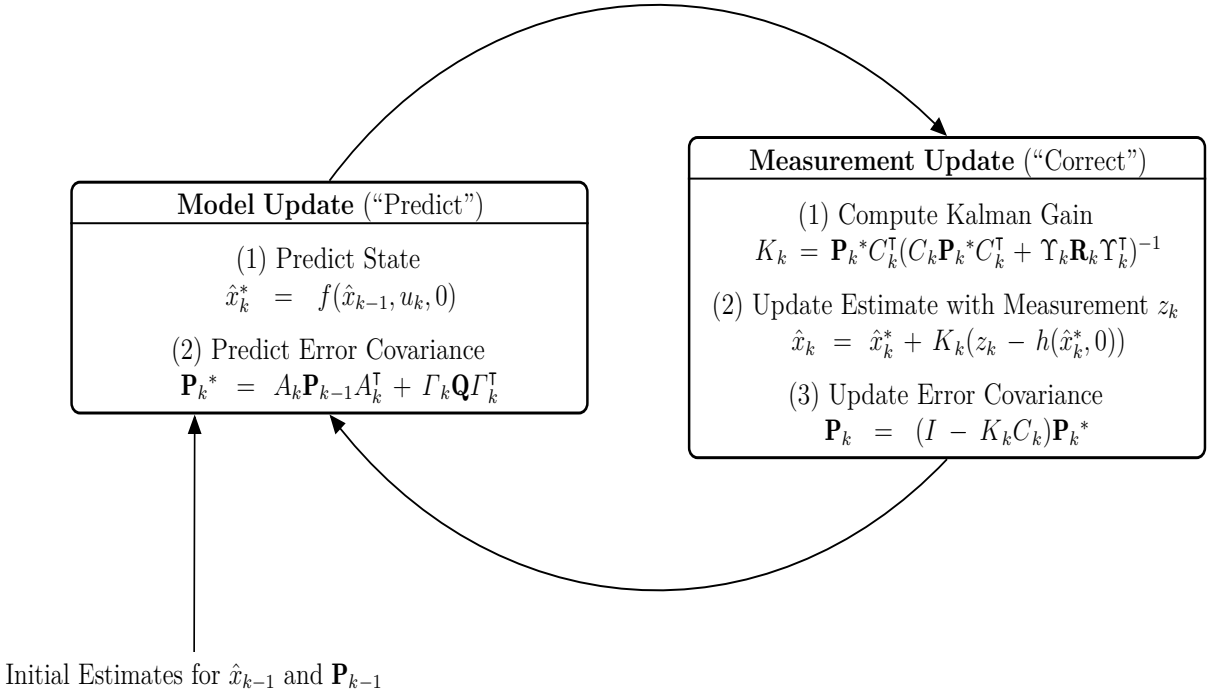


Figure 5.4: Extended Kalman Filter flow diagram

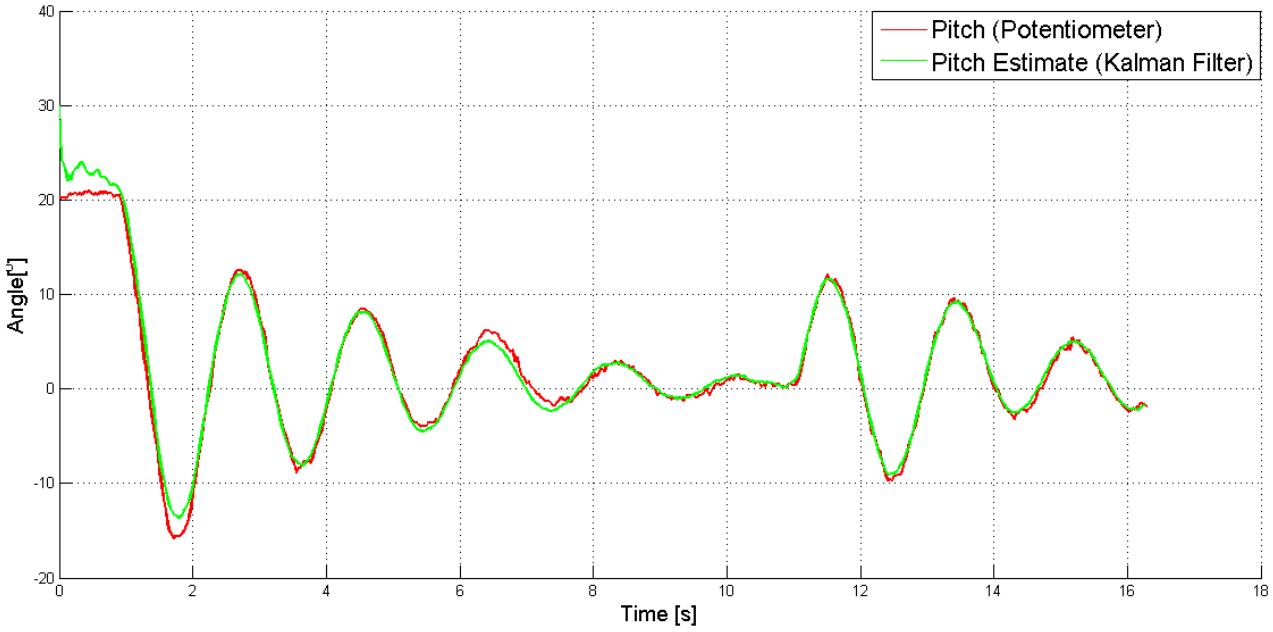


Figure 5.5: Time-domain response of the Extended Kalman Filter

6. Implementation

The Kalman Filters were initially written in MatLab. However the static Kalman Filter was the only Kalman Filter that was tested on the quad-rotor. It was the only one that could be implemented due to the limited computational speed available on the micro-controller.

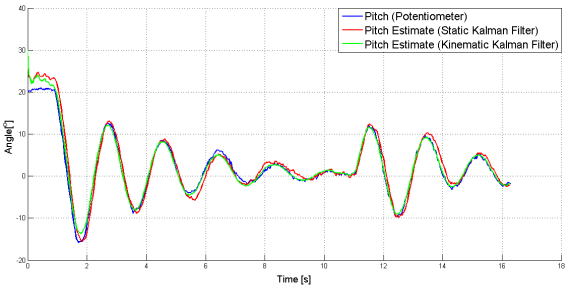
6.1 Testing the Kalman Filter in MatLab

The Kalman Filter was tested in simulation to observe its performance under high and low acceleration by means of comparison of attitude measured by the accelerometer and attitude measured by a potentiometer [1]. The Static and Kinematic Kalman Filters were designed and tested using m-files in MatLab, which can be seen in Appendix E.2.1 and E.2.2 respectively. The models used to estimate the orientation using the Static Kalman Filter are the same models used for the control of the quad-rotor and are presented in (3.3), (3.4), (3.8) and the model used for the Kinematic Kalman Filter was presented in (5.13). In order for the filters to be tuned the noise of the accelerometer, gyroscope and magnetometer were modeled and after which a Kalman Filter was produced so an accurate estimate of the attitude could be acquired. Figure 6.1 clearly shows that the Kalman Filter gives a good estimate of the orientation and smoothing of the accelerometer data does indeed take place.

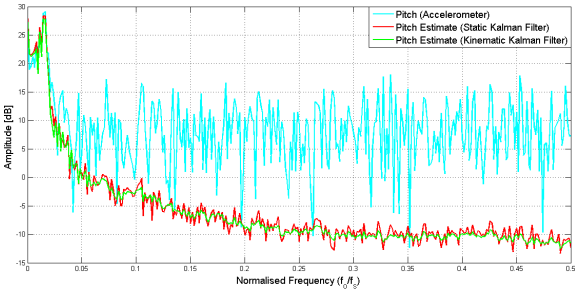
6.2 Testing the Kalman Filter in Simulation

To convert the Matrix Exponentiation function used in MatLab, approximations have to be made as the processing power required to calculate the Matrix Exponentiation in full is too great.

[1] In this case the potentiometer was taken to be the correct attitude as it was aligned along the axis of rotation of the quad-rotor. The potentiometer was a laboratory standard measurement device which produce an analog measurement with low noise, meaning all of the filters produced in the report could be tuned by means of this potentiometer reading.



(a) Time domain comparison of Static and Kinematic Kalman Filter



(b) Frequency domain comparison of Static and Kinematic Kalman Filter

Figure 6.1: Comparison of Kalman Filters used during the Project

6.2.1 Approximations Used

So as to implement the Kinematic Kalman Filter on the micro-controller an approximation to the Matrix Exponential was made. The Matrix Exponential can be defined as follows:-

$$e^{\underline{\mathbf{X}}} = \sum_{k=0}^{\infty} \frac{1}{k!} \underline{\mathbf{X}}^k \tag{6.1}$$

Equation 6.1 is an infinite series, thus, an approximation is required to implement the filter on the micro-controller.

$$e^{S(\omega_g)T_s} = I + S(\omega_g)T_s + \frac{(S(\omega_g)T_s)^2}{2!} + \cdots + \frac{(S(\omega_g)T_s)^5}{5!} \tag{6.2}$$

It was not required to go beyond the 3^{th} term in the approximation of the matrix exponential since terms larger than the 3^{th} were of too small a magnitude to affect the result.

6.3 Testing the Estimator with the Sensors

Section 6.1 shows the Kalman Filters response in simulation. However, when estimating the orientation of the quad-rotor, the actual orientation was required to ensure the estimate produced by the filter was correct. This was achieved by placing the sensors on the quad-rotor, which in turn was mounted on a rig, which had potentiometers to measure the actual orientation. The rig was arranged so that the pitch and roll could not vary more than $\pm 30^o$. A picture of this arrangement can be seen in Figure 6.2.

The **R** was chosen by gathering data while the sensor was level and at rest. This data was called into MatLab where the *cov* command was used to find the covariance of the accelerometer, gyroscope and magnetometer respectively. The **Q** value for the Static Kalman Filter was difficult to acquire without flight data so it had to be approximated. As for the Kinematic Kalman Filter, the **Q** was set to the covariance value of the gyroscope as the Φ of this filter is made from the gyroscope outputs.

The code used to communicate with the 9DOF was modified so that the potentiometer values could be read by means of an ADC on the micro-controller board. After the Micro-controller read the associated values (both 9DOF values and potentiometer values) they were then recorded on a SD card so the data could be called into MatLab. This allowed post processing to be applied to the 9DOF data. The **Q** was adjusted until the potentiometer data matched the estimate produced by the Kalman Filter at both high and low accelerations.

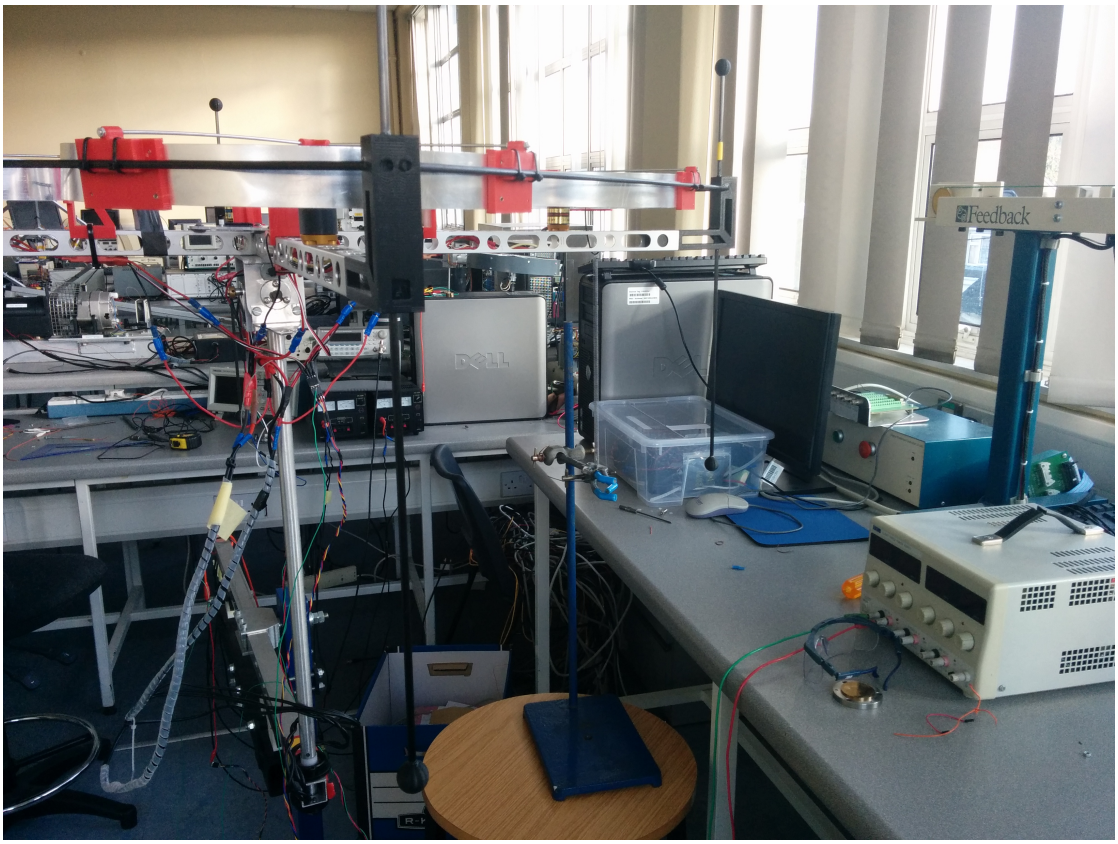


Figure 6.2: Rig used to test the Kalman Filter

6.4 Tuning the Kalman Filter

The Kalman Filter is often used as an estimator, but is widely known to be difficult to tune. Tuning the filter implies determining the **R** and **Q** matrices which control how the filter combines the model and the measurement to acquire an accurate estimate of the state vector. The accuracy of the filter's estimate depends on the **R** and **Q** matrices. Throughout the project many tuning methods were investigated, but the tuning method presented in [38] gave the best/desired results. The **R** matrix can be estimated with ease, as the covariance of the data produced by the sensor while at rest can easily be found.

Estimating the R matrix

As the accelerometer was thought to have a greater covariance at high accelerations (as the accelerometer also measures linear and angler acceleration) it was decided to test the device under high acceleration. In order to see if high acceleration components increased the covariance of the accelerometer data it was decided to gather data on the quad-rotor under high acceleration with motors on. It was found that the high acceleration components where lower than the noise floor produced by the vibrations due to the motors attached to the quad-rotor. The omission of the "high" high acceleration was possible as the magnitude of the angular acceleration of the quad-rotor is quite low. The covariance was found by means of the *cov*() function in MatLab. Hence, the **R** was calculated as follows:-

$$\mathbf{R}_a = \begin{bmatrix} \text{cov}(\phi) & 0 & 0 \\ 0 & \text{cov}(\theta) & 0 \\ 0 & 0 & \text{cov}(\psi) \end{bmatrix} \tag{6.3}$$

Estimating the Q matrix

Calculating the **Q** matrix proved to be quite a difficult task. Very little actual information regrading the noise present in the states was known. However, since **Q** was difficult to calculate without flight data it was decided to use a binary search using the ratio of **Q**/**R** as it was shown in [38] that this type of tuning could lead to an optimal response. In this tuning process the **R** was set to the theoretical value and **Q** was

adjusted by means of a binary search, thus a filter signal similar to the potentiometer value was found. A similar approach was used to tune the Kinematic Kalman Filter, except some initial knowledge was known about the **Q** as the transition matrix of this filter was made up solely of the gyroscope values. This allowed the **Q** matrix for this filter to be set to the covariance values of the gyroscope. This approach produced almost identical results to the Static Kalman Filter with little tuning, but in order to improve the filtering of the Kinematic Kalman Filter a binary search was also used to improve the estimation of this filter and the results of both Kalman Filters can be seen in figure 6.1.

7. Optimal Controller Design

7.1 Optimal Control Design

The previous sections develop methods of estimating the attitude of the quad-rotor by means of sensor fusion. The results from the Kalman Filter are used in developing a mathematical model of the quad-rotor and also as inputs to a controller and estimator. Figure 7.1 gives a block diagram comprising of the different sections required in the overall control of the quad-rotor.

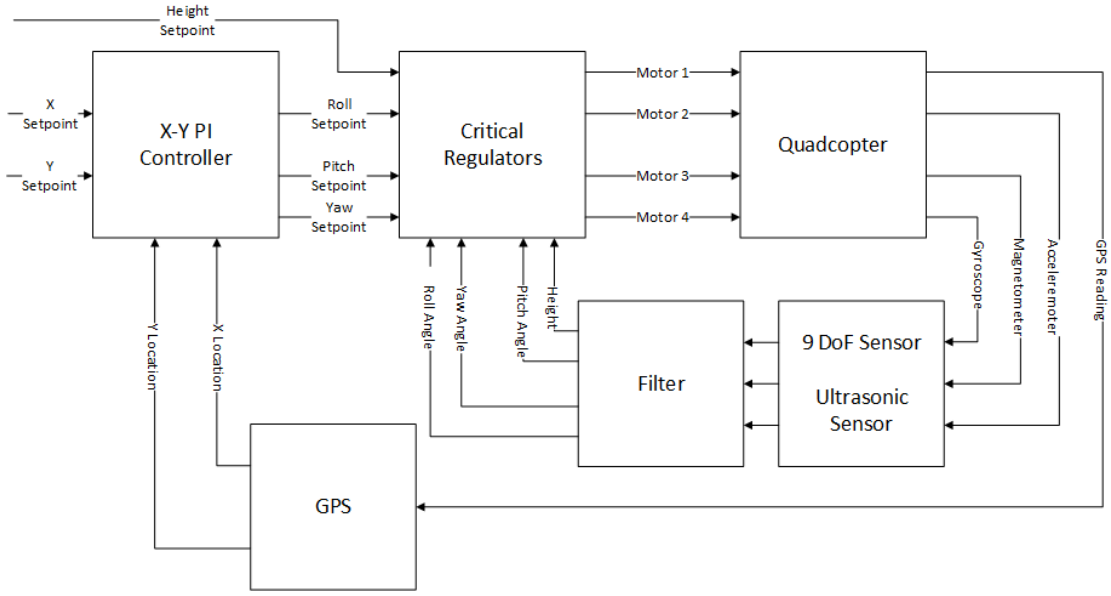


Figure 7.1: Block Diagram of Final control of the Quad-Rotor

The combination of gyroscopes and accelerometers are referred to as smart sensors, which estimates pitch, roll yaw and there respective rates.

The Kalman Filter and controller were designed and simulated using MatLab based on the quad-rotor model presented in 3.1.1. The current control set-points are roll,pitch and yaw, but the control has been developed for x and y to be made the set-points of the system. The current state used in the quad-rotor control scheme are as follows $[\theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}, h, \dot{h}, \Delta\omega]$

Where:-

θ,ϕ,ψ**:** are the angle at which the quad-rotor is with respect to the x, y and z respectively.

$\dot{\theta}, \dot{\phi}, \dot{\psi}$**:** are the rate of change of these angles

$h, \dot{h}$**:** is the height and rate of change of height of the quad-rotor

$\Delta\omega$**:** is the differential thrust difference between the two opposing motors along an axis.

The type of controller designed to control the quad-rotor is an optimal controller, which controls the quad-rotor by selecting regulator gains for the best possible performance. To achieve this, the aspects of the plants behavior that are desired to be controlled must be incorporated in a mathematical expression referred to as the "cost function".

7.2 Design of a LQR Controller

As the control scheme used on the quad-rotor was digital then any optimal controller used to stabilize the system had to be in discrete form too.Hence, the cost function to be minimized was of the discrete form and is defined as follows:-

$$J_N = \frac{1}{2}\sum_{n=0}^{N-1}(x_n^\intercal \mathbf{Q} x_n + u_n^\intercal \mathbf{R} u_n) + \frac{1}{2}x_N^\intercal \mathbf{Q}_N x_N \tag{7.1}$$

where x and u denote the states and inputs of the system respectively and $\mathbf{Q}$ and $\mathbf{R}$ are matrices chosen to apply the desired weights to various states and inputs respectively.

A controller designed by this approach is known as a Discrete Linear-Quadratic Regulator (DLQR) and is tuned by changing the $\mathbf{Q}$ and $\mathbf{R}$ matrices shown in (7.1) to give the optimal response. The $\mathbf{Q}$ and $\mathbf{R}$ matrices set a weighting on the state variable tracking and the control action respectively. If $\mathbf{Q}$ is increased then a penalty is introduced which limits the deviation of the states from their set-points. Following from this if the $\mathbf{Q}$ matrix is increase so too is the feedback gains. In contrast, if the $\mathbf{R}$ matrix is increased, then aggressive control responses are penalized, which leads to a reduction in the feedback terms. Even though an LQR controller is an optimal controller there are limitations inherent in its design. The main failing of an LQR based

control scheme is the fact that its frequency response rolls of at 20 dB per decade [7, pg 584-617]. Since the LQR controller has such a poor frequency response it is often used injunction with a optimal filter such as the Kalman Filter so as to alleviate this problem.

7.2.1 Tuning of the LQR controller

When a LQR based control scheme is implemented there is two methods of tuning the controller, one is to set the **Q** and very **R** until the desired response is achieved. This form of tuning is possible as the LQR controller is derived in a similar manner to Kalman Filter. Hence, a tuning approach similar to that used to tune the Kalman Filter in Chapter 6 can also be used to tune the LQR controller.

The second approach is to use Bryson's Rule [39]. Bryson's Rule works by setting a maximum acceptable value of each state and then penalizes them accordingly. Bryson's Rule is commonly denoted as follows:-

$$\begin{aligned} Q_{ii} &= \frac{1}{(z_i^{max})^2} \qquad & i \in 1, 2, \cdots l \\ R_{ii} &= \frac{1}{(u_j^{max})^2} \qquad & j \in 1, 2, \cdots m \end{aligned} \tag{7.2}$$

Where:-

z_i^{max}**:** is the maximum acceptable value of the i state of the system

u_i^{max}**:** is the maximum acceptable value of the i input of the system.

which means the LQR equation can be redefined as follows:-

$$J_N = \int_0^\infty \left(\sum_{i=1}^{l} Q_{ii} z_i(t)^2 + \sum_{j=1}^{m} R_{jj} u_j(t)^2 \right) \tag{7.3}$$

This method of tuning the LQR controller was advantages as it allowed a limited to be set on states which in real life had limits and could saturate. An example of such a saturation is the angular velocity of the motors and the PWM signal which also has an upper limit.

7.2.2 Observer Design

As the state space controller used on the quad-rotor required unmeasurable states, an observer is required to estimate these unknown states. An observer was designed which used sensor readings and the mathematical model of the quad-rotor presented in Section 3.1 to estimate the unknown states.The observer required knowledge of the desired responses, which is achieved by the placement if its poles. A common method of designing an observer is by means of the Butterworth Constellation Technique, such an observer can easily be designed in MatLab by means of the *place* function.

Butherworth Constellation Technique

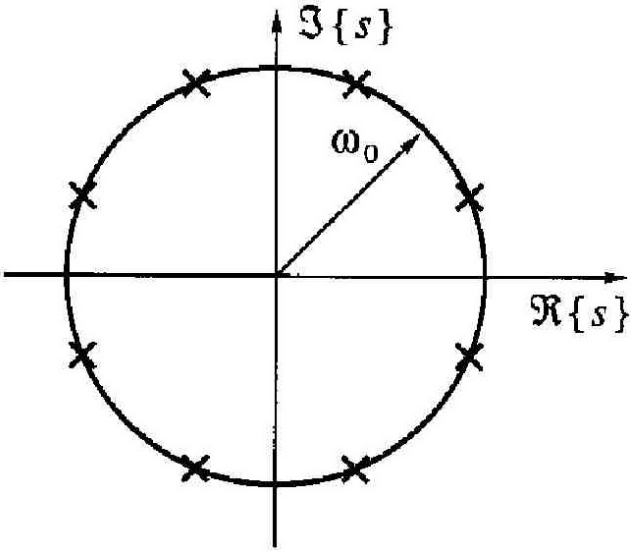


Figure 7.2: 8 Order Butterworth pole placement diagram

Figure 7.2 gives a graphical representation of the placing poles using the Butterworth Technique. The radius of the circle is ω_n, found using the following:-

$$\frac{4}{\zeta\omega_n} = T_{s_{2\%}} \tag{7.4}$$

Where:-

ζ**:** is the maximum acceptable value of the i state of the system

$T_{s_2\%}$**:** is the settling time of the observer.

Once the observer was designed it was tested with the LQR controller which was designed by means of (7.1). The combination of the LQR controller, observer and

Kalman Filter the following response presented in Figure 7.3. Under this controller it was possible to control the roll,pitch and yaw axes.

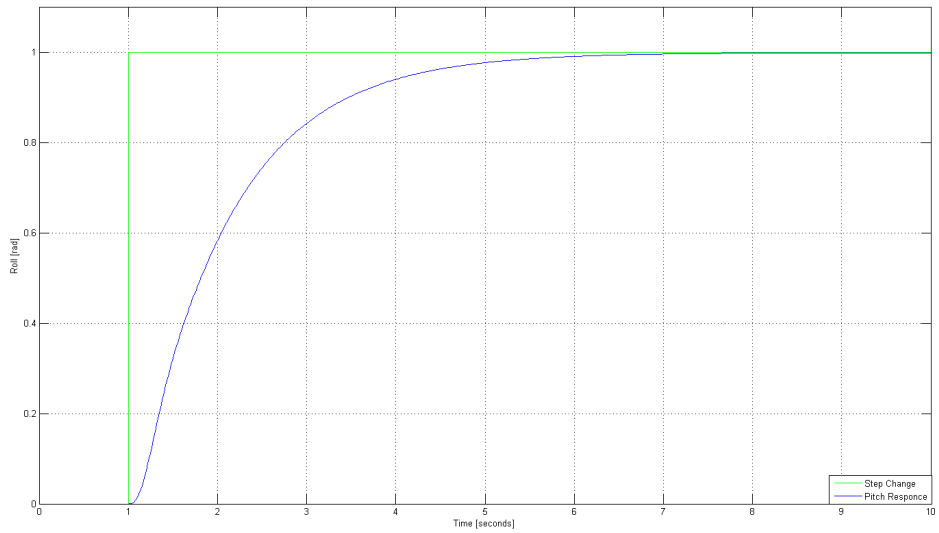


Figure 7.3: Step response of the quad-rotor for the roll axis

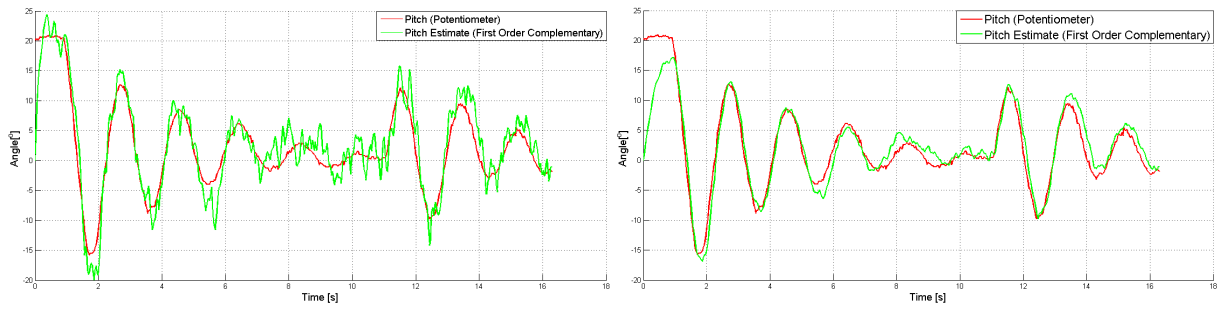
8. Results

This chapter discusses the results obtained when testing both the Complementary Filter and Kalman Filter. Initially the pitch angles were plotted over time with different **Q** values until the best results was obtained. The estimated orientation from the Complementary Filter, Kalman Filter and the potentiometer data were plotted on the same graphs and compared.

8.1 The Complementary Filter

8.1.1 The effect τ on Estimation

The effect of τ on the estimate was tested by varying τ and can be seen in Figure 8.1. Figure 8.1 compares the estimated data from the Complementary Filter with the actual values from the potentiometers. As seen in Figure 8.1 of the value of τ is set too low the accelerometer becomes the driving factor and the estimate remains noisy. While if τ is set too high the gyroscope data is used more than the accelerometer, hence a delay is introduced. Thus, a middle ground which will give an adequate estimate of the attitude of the quad-rotor. The best choice of τ was found to be 0.468 for use with this system.



(a) Pitch estimate with $\tau = 0.1$

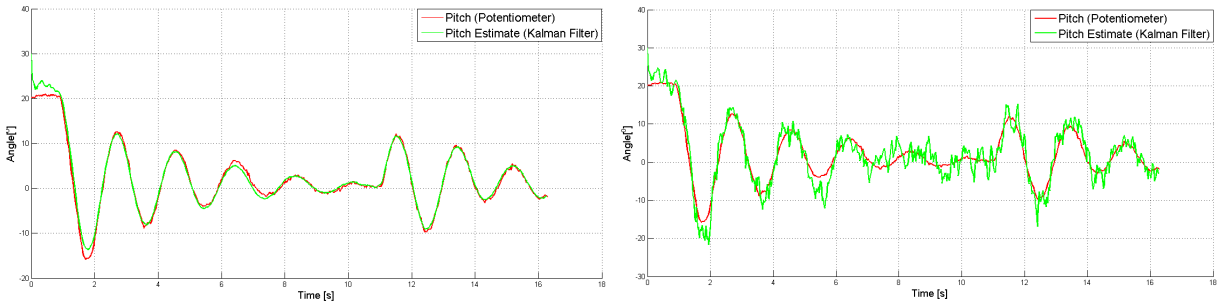
(b) Pitch estimate with $\tau = 0.5$

Figure 8.1: Effect of varying τ on the estimate produced by the Complementary Filter

8.2 The Kalman Filter

8.2.1 Effect of Q on Estimation

The effect **Q** has on the estimated was tested by varying the **Q** matrix and comparing the estimated data from the Kalman Filter with the actual values from the potentiometers. In these tests, **R** was kept constant at its theoretical value since it is the ratio of the two parameters that effect the estimate.



(a) Pitch estimate with $\mathbf{Q}/\mathbf{R} = 8.805 \times 10^{-5}$

(b) Pitch estimate with $\mathbf{Q}/\mathbf{R} = 0.0881$

Figure 8.2: Effect of varying $\mathbf{Q}/\mathbf{R}$ on the estimate produced by the Kalman Filter

Figure 8.2a and 8.2b clearly show that the estimate of the pitch angles varies with **Q**. At a lower **Q** than **R**, the response of the filter is more accurate. This implies that there is less noise on the states relative to the measured signals. The Kalman Filter in this case assumes that the predicted pitch values are reliable and need little correction. Therefore increasing the Kalman gain matrix places more emphasis on the measurements and less on the prediction values [38]

Figure 8.3 shows the cross-correlation of the error between the estimated and actual pitch values. If the noise present in the Kalman Filter's model and the present in the accelerometer and gyros is Gaussian and uncorrelated, Figure 8.3 would be all zero except at the center, where the response would be one. Therefore, Figure 8.3 shows that the noise present in the signals is not uncorrelated.

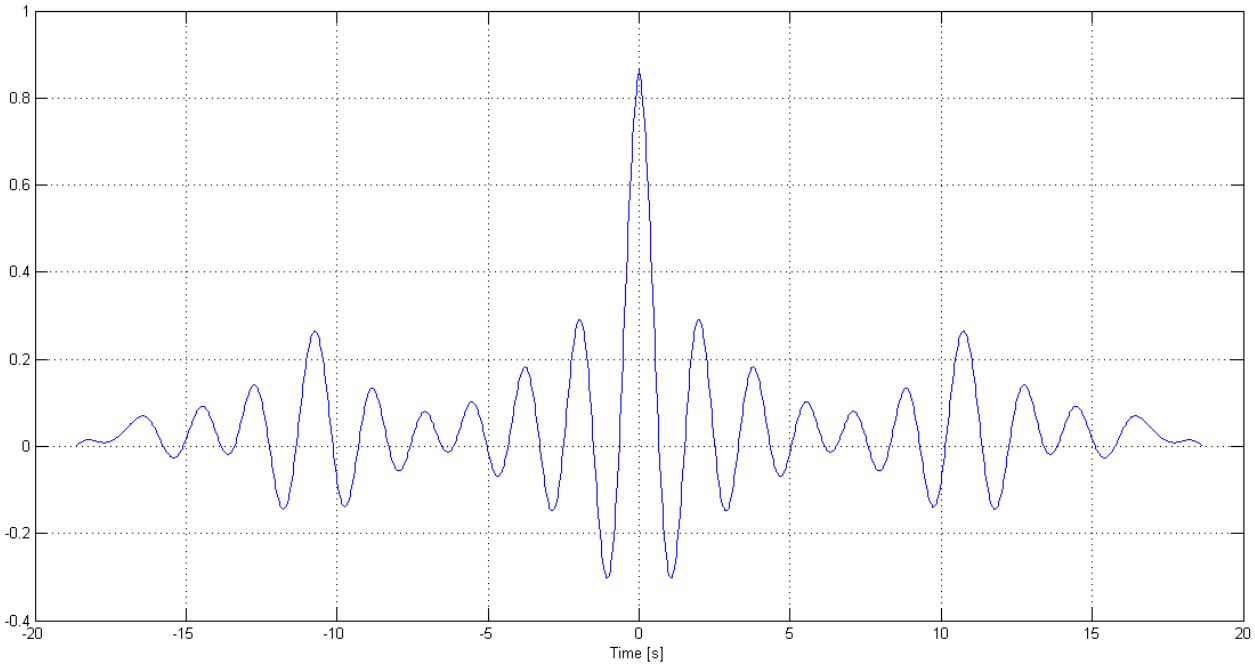


Figure 8.3: Cross-Correlation of estimated pitch error

9. Future Work

To-date the critical controllers have been designed, with the roll,pitch and yaw having been tested on the quad-rotor. Also an xy controller has been designed and the maths required to map GPS data to NED frame where the positional control is implemented has been developed in this report. But as flight data was not collected a Kalman Filter was not implemented to convert the GPS data to the NED frame were the control is implemented. Hence, the next action would be to collect flight data and find $\mathbf{R}$ and $\mathbf{Q}$ matrices for a GPS Kalman Filter. In the development of such a filter it maybe required to increase the up-date rate to the positional controller of the quad-rotor by doing more predict stages in the Kalman Filter and then correcting these "prediction" every n cycles of the Kalman Filter by means of a correction up-date from the GPS module.

After the Kalman Filter is developed, the quad-rotor would be semi/fully autonomous. As the GPS only gives accurate position while having access to four satellites, it might be beneficial to attach a vision system to the quad-rotor so the positional controller will have updates while the GPS link is broken. Also if a vision system is added to the quad-rotor, it would be possible to fly the quad-rotor in locations where GPS guidance is not possible, such as indoors.

If the quad-rotor was capable of flying indoors, then building mapping could be a natural next step. This mapping could be done in two possible ways, by means of Lidar or ultrasonic sensors. The Lidar would be more expensive, but is more accurate, is less prone to interference and has a faster update. The ultrasonic sensors on the order hand are cheaper and require less power, but are less accurate and have a limited of operation.

10. Conclusion

A mathematical mode of the quad-rotor has be developed. Models for the accelerometer, gyroscope and magnetometer have been developed (as seen in Section 3.3)and proven to be correct by means of experiment. A GPS model has also been presented in Section 3.4 and can be used in future projects if a GPS-based device is used.

As the sensors used in the project are not ideal, the attitude of the quad-rotor had to be estimated by means of data fusion. These estimators have been tested in both simulation and on the quad-rotor.

The pitch and roll estimates were compared with actual values and the results showed that the filters worked as required an indeed produced an optimal response. Chapter 6 shows that the Kalman Filter is the optimal choice, but is more computationally intensive to run on some micro-controllers than a Complementary Filter. If the Kalman Filter is not an option, then a Complementary Filter similar to the ones presented in this report can be used for attitude estimate as they are less computational intensive.

The attitude estimators were designed and simulated using Simulink and MatLab. The results showed from these simulations demonstrated that these estimators could be implemented to control the quad-rotor. The Complementary Filter and Kalman Filer where both tested on the quad-rotor with a LQR based control scheme and both were shown to give adequate results.

Finally, the report gives a detailed description of how the filters compare to each other and how they can be implemented on a micro-controller.

Appendices

A. Modeling parameters

	Parameter	Value	Unit
Brushless DC motors	τ_m	0.07	s
	J	80	n kg m^2
	d	0.44531	μ kg m^2 rad^-
	b	0.0189209	
	K_ω	501.5133	rad s^{-2}
Mechanical	m	2.26	Kg
	l	0.338	m
	I_ϕ	0.1274	kg m^2
	I_θ	0.1221	kg m^2
	I_ψ	0.237	kg m^2
	β_ϕ	0.0194	N m s^-
	β_θ	0.0194	N m s^-
	β_ψ	0	N m s^-
	g	9.81	m s^{-2}
	Total thrust	32	N
	damping in x	32	N m s^-
	damping in y		N m s^-
Controller	T_s	20	m s
Electronic Speed Controller (ESC)	PWM_{max}	2	m s
	PWM_{min}	0.9	m s

Table A.1: Table of Essential Quad-Rotor Parameters

B. Rotation Matrix

B.1 Derivation of the Rotation Matrix

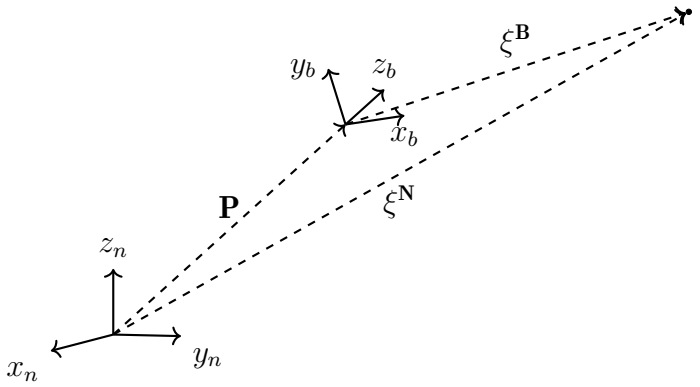


Figure B.1: General Transform of a Vector

The unit vector x_b, y_b, z_b define the (B) frame. In terms of the corresponding vectors of frame (N) these unit direction vectors become x_b^n, y_b^n, z_b^n,where

$$x_b^n = \begin{bmatrix} x_b.x_n \\ x_b.y_n \\ x_b.z_n \end{bmatrix} \tag{B.1}$$

Thus, the orientation of frame (B) with respect to frame (N) can br defined as $3x3$ matrix.

$$R_N^B = [x_b^n, y_b^n, z_b^n] \tag{B.2}$$

The matrix presented in B.2 is referred to as a rotation matrix. It is equal to the rotation that must be applied to (N) to place it at (B)

Hence, from figure B.1 on can define the following equation:-

$$\xi^N = R_N^B \xi^B + P \tag{B.3}$$

Manipulating (B.3) and using the property $(R_N^B)^{-1} = R_B^N = Rot$,

$$\zeta^B = Rot(\zeta^N - P) \tag{B.4}$$

B.1.1 Euler Angle Derivation

As (B.2) has been already defined, it is now possible to define any 3-D vector mapping using the following [1] :-

$$R_N^B = R_N^{B'} R_{B'}^{B''} R_{B''}^{B'''} \tag{B.5}$$

These relationship was used to find the rotation matrix presented in 3.2.1.

B.1.2 Proof of the Transition Matrix used in Kinematic Kalman Filter

$$\begin{aligned} \dot{x} &= S(\omega_g)x \\ \int_{x_{k+1}}^{x_k} \frac{1}{x} dx &= \int_{k+T_s}^{k} S(\omega_g)dt \\ \ln\left|\frac{x_{k+1}}{x_k}\right| &= S(\omega_g)T_s \\ x_{k+1} &= e^{S(\omega_g)T_s}x_k = \Phi_k x_k \end{aligned} \tag{B.6}$$

(B.6) uses the matrix exponential to find the transition matrix at each iteration. In other to implement this on a micro-controller (B.7) can be truncated [2] to a set number of terms, which would all (B.6) to be implemented on a micro-controller at each up-data of the kinematic Kalman Filter.

$$e^{\underline{\mathbf{X}}} = \sum_{k=0}^{\infty} \frac{1}{k!} \underline{\mathbf{X}}^k \tag{B.7}$$

[1] This is commonly refereed to as the general case of for the Euler Angle Relationship

[2] In this project it was found that 5 terms where sufficient to capture the required information of the matrix exponential

C. Definition of Statistics used in the Implementation of the Kalman Filter

C.1 Mean

The mean or average value of a time-varying signal $x(t)$ can be found by taking a total of N measurements of $x(t)$ at regular intervals. Hence, the mean can then be found by means of (C.1)

$$\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k \tag{C.1}$$

Note the mean is also stated/known as the expected value, which is denoted $E[x]$. Therefore $E[x] = \bar{x}$.

C.2 Standard Deviation

The standard deviation is a statistic that gives an indication into how tightly all the measurements $x(t)$ are clustered around the mean value in a set of data. It is found by calculating the root-mean-square of the deviations of the samples from the mean and is defined as follows :-

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(x_k - \bar{x})^2} \tag{C.2}$$

C.3 Variance

The variance of a set of data provides information about how far samples of $x(t)$ are spread around/from the expected value. If the variance is high, it implies the data has a wide spread around the ”mean” and vice versa. The unbiased sample variance is calculated using the following :-

$$\sigma_x^2 = \frac{1}{N-1}\sum_{k=1}^{N}(x_k - \bar{x})^2 \tag{C.3}$$

Note the variance can also be denoted as $\sigma_x^2 = E[(x - \bar{x})^2]$

C.4 Covariance

The covariance is a measure of how much two random variables change together and how strongly they relate. Taking two zero-mean signal $x(t)$ and $y(t)$, the covariance between them can be found using the following.

$$\sigma(x, y) = \frac{1}{N-1} \sum_{k=1}^{N} (x_i - \bar{x})(y_i - \bar{y}) = E[(x - \bar{x})(y - \bar{y})] \tag{C.4}$$

Hence, if $x(t)$ and $y(t)$ are uncorrelated signal, then $\sigma_{xy} = 0$ [1]

C.5 The Covariance Matrix

Taking signal vector, $\underline{\mathbf{x}}(t)$, made up of n random signals $[x_1, x_2, \cdots, x_n]$, the covariance matrix contains all possible covariances between elements of the vector, resulting in the following:-

$$\sigma^2(\underline{\mathbf{x}}) = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1}\sigma_{x_2} & \dots & \sigma_{x_1}\sigma_{x_n} \\ \sigma_{x_2}\sigma_{x_1} & \sigma_{x_2}^2 & \dots & \sigma_{x_2}\sigma_{x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{x_n}\sigma_{x_1} & \sigma_{x_n}\sigma_{x_2} & \dots & \sigma_{x_n}^2 \end{bmatrix} \tag{C.5}$$

Due to the commutative property of the covariance between two signals, a covariance matrix is equal to is transpose. Also, if each data random signal above were independent of all other signals, the covariance matrix would be diagonal (i.e, all non-diagonal elements could be set to zero.) [2]

[1] The Covariance of a signal $x(t)$ and $y(t)$ is most commonly denoted σ_{xy}

[2] Note (C.5) is commonly written as $E[\underline{\mathbf{x}}\underline{\mathbf{x}}^\intercal]$

C.6 Gaussian Variables

Gaussian distribution is a measure of the probability density function of a time-varying signal. The graph of the probabilities of the data plotted against the amplitude of the probability density function results in a bell-shaped curve as shown in figure C.1

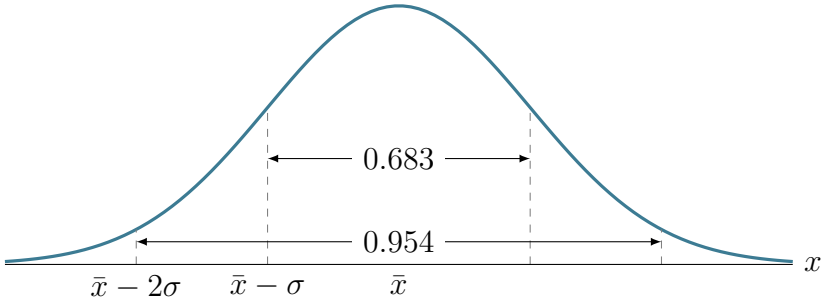


Figure C.1: Gaussian Distribution of a time-varying signal

Note the Gaussian distribution can be represented by (C.6)

$$f(x, \bar{x}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})}{2\sigma^2}} \tag{C.6}$$

C.7 White Noise

White noise can be considered as noise that contains all the possible frequency components at equal probability. It therefore has a flat probability distribution when plotted against frequency. However, it's Gaussian in terms of the amplitude of the noise. The spectral density function of white noise can be seen figure C.2

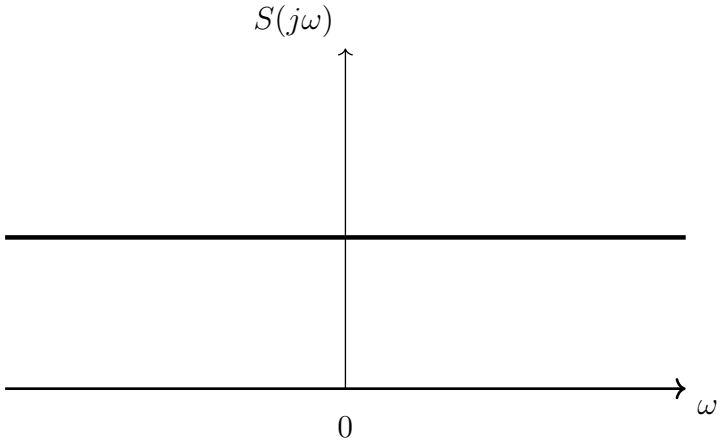


Figure C.2: Spectral density function of white noise

D. Kalman Filter

This appendix draws heavily on the work present in [40]. As stated in the main report the Kalman Filter is a recursive filter which computes corrections to a system based on external measurements. The amount at which the filter corrects the values depends on the current estimate of the system error statistics. Note the filter requires knowledge of linear algebra and stochastic processes. As the filter requires knowledge of stochastic processes it can be quite difficult to understand, fortunately, it can be understood in fairly simple terms. All that is required is an understanding of various common statistical properties, which are presented in Appendix C

Before the Kalman Filter is introduced the concept of a state vector must be defined as it is central to the formulation of the filters algorithm. A state vector is a set of values which are used to describe the "state" of the system. For this project, which is a navigation system, the state vector is given as position,velocity,attitude and sensor errors. Since the filter is used to correct errors in a navigation system, working in terms of error states is convenient and for an arbitrary state x, the estimate can be defined as follows:-

$$\hat{x} = x + \delta x \tag{D.1}$$

where x is the true value of the state and δx is the error in the estimate. The purpose of the Kalman Filter is to estimate this error δx and correct it.

In order to derive such a filter all system equations and measurement equations have to be expanded in a Talyor series. As the error in the states are small, it is only required to keep the first order terms. Hence, the equations are linear. This formulation, where the state errors are filter instead of the states themselves, is known as an extended Kalman Filter.

So as to illustrate the basic principles of the filter,lets begin with the simplest assumptions possible. For a Kalman Filter assume that the mean of all states is zero[1]. For ease the following explanation will only deal with a single state δx. In this

[1]It is required to rearrange the states if there mean is not zero

development there will be some *priori* estimate of the initial standard deviation (σ) and to keep consistent with literature the variance (σ^2) of the state will be $\mathbf{P}$

The dynamics of the error state will, in general, be described by some differential equation. So, to begin with assume the equation takes the simple form

$$\delta\dot{x}(t) = \alpha\delta x(t) \tag{D.2}$$

Where α is some constant, thus, (D.2) has the formal solution.

$$\delta x(t + T_s) = e^{\alpha T_s}\delta x(t) = \Phi(T_s)\delta x(t) \tag{D.3}$$

Where the state transition matrix has been defined as $\Phi(T_s) = e^{\alpha T_s}$. The state transition matrix describes the evolution of the system error states in time, when no measurements are being processed.

By definition, the variance of the error state is $\mathbf{P} = \mathrm{E}[\delta x^2]$. By assumption, $\mathrm{E}[\delta x] = 0$. Applying this operator to (D.3), the following will be arrived at:-

$$\mathbf{P}(t + T_s) = \Phi(T_s)\mathbf{P}(t)\Phi(T_s) \tag{D.4}$$

Note the above equation is defined in this form so as to correspond to the form it will take when considering state vectors of more than one dimension (a non-scaler).

Now lets consider the effect of a measurement on the system. Assume that there is access to an external measurement of x which is corrupted by noise.

$$\tilde{x} = x + v \tag{D.5}$$

Where $\tilde{x}$ is the measured value of x and v is a zero-mean Gaussian white sequence of variance $\mathbf{R}$. Next a measurement residual is defined as follows:-

$$z = \hat{x} - \tilde{x} = \delta x + v \tag{D.6}$$

Since the measurement is corrupted by noise, some method of weighting the measurement in correcting the state variable is required. This problem of computing optimal gains is the heart of the Kalman Filter. Thus, it is required to compute the

Kalman Gains, K, in the following equation:-

$$\hat{x} = \hat{x} - Kz \tag{D.7}$$

Where Kz is the best estimate of the error state.

As $\mathbf{R}$ is the variance of the measurement error and $\mathbf{P}$ is the variance of our estimate of δx,thus, it is expected that the gains are a function of these two variables. Hence, the following result is valid for one dimension:-

$$K = \frac{P}{P + \mathbf{R}} \tag{D.8}$$

Equation D.8 makes logical sense. If the measurement is more accurate than the state error, then $\mathbf{R} \ll \mathbf{P}$ and $K \cong 1$. In this case, (D.8) is approximately the as (D.7). If the state error is more accurate than the measurement, there should be little correction due to the measurement. In this case $\mathbf{R} \gg \mathbf{P}$ and $K \cong \mathbf{P}/\mathbf{R}$ so the correction is very small. Thus the Kalman Gains make a great deal of sense (in the trivial case). Of course, this is not a derivation of the gain equations, but in this case the concern is not but on mathematical rigor, but instead on a simple method of defining the Kalman Gain. This simple method also has the advantage of being computational less intensive than the rigorous Kalman Gain methods which are presented in most text books (see [7, pg 791-794]). Hence, this method of computing the Kalman Gain K is capable of being calculated on a low cost micro-controller.

Once a measurement has been taken, the variance of $\mathbf{P}$ must be up-dated to reflect this new information. The result for one dimension is as follows:-

$$P = PR/(R + P) \tag{D.9}$$

D.1 Basic Kalman Filter for n-dimensional vector

Since x is in general an n-dimensional vector, $\mathbf{P}$ will be an nxn matrix. The above equations will become correspondingly more complicated, but the principle will remain the same. Before the introduction of higher dimensional spaces, lets summarize the process descried thus far. The sequence of events is as follows:

1. The variance of δx is initialized as P_0
2. The variance is propagated forward in time to the first measurement as:-

$$P_1 = \Phi(T_s)P_0\Phi(T_s)$$

3. The gain is computed based on this value of $\mathbf{P}$ and the measurement error variance,

$$K_1 = P_1/(P_1 + \mathbf{R})$$

4. The measurement residual is computed as $z = \hat{x} - \tilde{x}$
5. The state vector is corrected according to $\hat{x} = \hat{x} - K_1 z$
6. The variance of δx is updated to reflect the measurement as

$$P_1' = P_1\mathbf{R}/(\mathbf{R} + P_1)$$

7. This variance is propagated forward to the next measurement as

$$P_1 = \Phi(T_s)P_1\Phi(T_s)$$

Thus, the process begins again

Note, P_k' denotes the value of P immediately after the k^{th} measurement and P_k to denote its value immediately before.

In this description, the noise present in the process $\mathbf{Q}$ has been neglected. This effect takes the form of a random forcing function in (D.2).

$$\delta\dot{x}(t) = \alpha\delta x(t) + v \tag{D.10}$$

where v is a zero-mean Gaussian white noise of power spectral density N. This will add a term to (D.3) as follows:-

$$\begin{aligned}\delta x(t+T_s) &= e^{\alpha T_s}\delta x(t) + \int_t^{t+T_s}\left[e^{\alpha(\tau-t)}\right] v d\tau \\ &= \Phi(T_s)\delta x(t) + \int_t^{t+T_s}\left[\Phi(\tau-t)\right] v d\tau\end{aligned} \tag{D.11}$$

and (D.4) is also adjusted as follows:-

$$\mathbf{P}(t+T_s) = \Phi(T_s)\mathbf{P}(t)\Phi(T_s) + \mathbf{Q} \tag{D.12}$$

Where

$$\mathbf{Q} = N\int_t^{t+T_s}\left[e^{2\alpha(\tau-t)}\right] d\tau \cong NT_s \quad \text{for} \quad 2\alpha T_s \ll 1. \tag{D.13}$$

All other equations remain the same.

As stated previously, δx, is in general, a vector defined as follows:-

$$\delta\underline{x} = \begin{bmatrix}\delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_n\end{bmatrix} \tag{D.14}$$

P now becomes an $n \times n$ matrix defined as follows:-

$$P = E[\delta\underline{x}\delta\underline{x}^\intercal] \tag{D.15}$$

where $\delta\underline{x}^\intercal$ is the transpose of (D.1). This is referred to as the covariance matrix. The diagonal elements will be the variances of the individual error states. The off-diagonal elements will be a measure of the correlations between the corresponding diagonal elements. Correlations are important as they permit the indirect estimate of a state by measuring a correlated state. Correlations may arise through the state transition matrix or through measurements.

The measurement (D.6)

$$\underline{z} = \underline{H}\delta x + v \tag{D.16}$$

Where $\underline{\underline{H}}$ is the measurement (or observation) matrix. The vector form of the measurement assumes that more that one measurement is taken. This equation also assumes that the measurements consist of quantities that can be expanded in terms of the components of the error state vector. We made this linearization assumption at the beginning. Equation D.4 generalizes to the following:-

$$\mathbf{P}(t + T_s) = \Phi(T_s)\mathbf{P}(t)\Phi(T_s)^\intercal + \mathbf{Q} \tag{D.17}$$

Where $\mathbf{Q}$ is a matrix generalization of the process noise term. The state transition matrix will not, in general, be a simple exponential. Computation of this matrix will usually require some form of numerical integration.

For optimal gain (D.8) and (D.9) become the following:-

$$K = PH^\intercal(HPH^\intercal + \mathbf{R})^{-1} \tag{D.18}$$

and

$$P' = [I - KH]P \tag{D.19}$$

In the above, $\mathbf{R}$ is now a matrix of noise associated with each element of the measurement vector. Note this development of the Kalman Filter is not mathematical rigid and is a simplified version which can be used on micro-control. While for a more detailed please consult [30, 36]

E. MatLab Code

E.1 Complementary Filter (MatLab Code)

E.1.1 First Order Complementary Filter

```
function [pitch_est_comp,roll_est_comp] =comp_filter_first_tustin(acc_data,gyro_data
 ,ts,tau_roll,tau_pitch)
% First Order Complamentry Filter used to estimate pitch and roll of the quad-rotor
% function [pitch_est_comp,roll_est_comp] =comp_filter_first_tustin(acc_data,
 gyro_data,ts,tau_roll,tau_pitch)
%
% Inputs:-
%  acc_data    : A logged form of the accelerometer data (denoted $a_x$, $a_y$ and
 $a_z$ in this report, also this input has to be in an array formate)
%  gyro_data   : A logged form of the gyroscope data (denoted $w_x$, $w_y$ and $w_z$
   in this report, also this input has to be in an array formate)
%  ts          : Sampleing of the controller used to collected the gyroscope and
 accelerometer data
%  tau_roll    : Filtering coeffient for the roll axis (sets the cutoff frequency of
   the filter)
%  tau_pitch   : Filtering coeffient for the pitch axis (sets the cutoff frequency
 of the filter)
%
% Output:-
%  pitch_est_comp : Outputs the pitch estimate for the given cutoff frequency (in an
   array formate)
%  roll_est_comp  : Outputs the roll estimate for the given cutoff frequency  (in an
   array formate)
%
%Note this Complementary Filter code used Tustin's Methodd to emulate the analog
 filter to the discrete domain.

wx = [0;gyro_data(:,1)];
wy = [0;gyro_data(:,2)];
wz = [0;gyro_data(:,3)];

%model of the filter

pitch = atan(-acc_data(:,1)./(sqrt(acc_data(:,2).^2 + acc_data(:,3).^2)));
roll  = atan(acc_data(:,2)./acc_data(:,3));

pitch = [0,pitch'];
roll =  [0,roll'];

```

```
30  pitch_est_comp=[0.0];
31  roll_est_comp=[0.0];
32
33  alpha_roll  = ts - 2*tau_roll;
34  beta_roll   = ts + 2*tau_roll;
35
36  alpha_pitch = ts - 2*tau_pitch;
37  beta_pitch  = ts + 2*tau_pitch;
38
39  for  i=1:(length(acc_data(:,1))-1)
40
41  roll_est_comp(i+1)  = 1/beta_roll*(  ts*(roll(i+1) + roll(i)) -alpha_roll*
     roll_est_comp(i) + tau_roll*ts*(wx(i+1) + wx(i)));
42  pitch_est_comp(i+1) = 1/beta_pitch*(  ts*(pitch(i+1) + pitch(i)) -alpha_pitch*
     pitch_est_comp(i) + tau_pitch*ts*(wy(i+1) + wy(i)));
43  end
44
```

Snippet E.1: MatLab Code Used to Implement a First Order Complementary Filter

E.1.2 Second Order Complementary Filter

```
1   function [pitch_est_comp,roll_est_comp]=comp_filter_second_order(acc_data,
    gyro_data,ts,roll_parameters,pitch_parameters)
2   % Second Order Complamentry Filter used to estimate pitch and roll of the quad-
    rotor
3   % function [pitch_est_comp,roll_est_comp]=comp_filter_second_order(acc_data,
    gyro_data,ts,roll_parameters,pitch_parameters)
4   %
5   %
6   % Inputs:-
7   %           acc_data         : A logged form of the accelerometer data (denoted
    $a_x$, $a_y$ and $a_z$ in this report, also this input has to be in an array
    formate)
8   %           gyro_data        : A logged form of the gyroscope data (denoted $w_x$,
     $w_y$ and $w_z$ in this report, also this input has to be in an array formate)
9   %           ts               : Sampleing of the controller used to collected the
    gyroscope and accelerometer data
10  %           roll_parameters  : Array containing the roll tuning parameters as
    follows [damping, wo]
11  %           pitch_parameters : Array containing the pitch tuning parameters as
    follows [damping, wo]
12  %
13  % Output:-
14  %          pitch_est_comp : Outputs the pitch estimate for the given cutoff
    frequency (in an array formate)
15  %          roll_est_comp  : Outputs the roll estimate for the given cutoff
    frequency  (in an array formate)
16
```

```
17      ki_roll       =roll_parameters(2)^2;
18      kp_roll       =2*roll_parameters(1)*roll_parameters(2);
19
20      ki_pit        =     pitch_parameters(2)^2;
21      kp_pit        =  2*pitch_parameters(1)*pitch_parameters(2);
22
23      %varible for the disscert filter
24      Gamma_pit = ki_pit*ts^2 - 2*kp_pit*ts;
25      alpha_pit = Gamma_pit + 4;
26      xi_pit    = 2*ki_pit*ts^2;
27      eta_pit   = ki_pit*ts^2 + 2*kp_pit*ts;
28      beta_pit  = xi_pit - 8;
29      delta_pit = eta_pit + 4;
30
31      Gamma_roll = ki_roll*ts^2 - 2*kp_roll*ts;
32      alpha_roll = Gamma_roll + 4;
33      xi_roll    = 2*ki_roll*ts^2;
34      eta_roll   = ki_roll*ts^2 + 2*kp_roll*ts;
35      beta_roll  = xi_roll - 8;
36      delta_roll = eta_roll + 4;
37
38      sigma = 2*ts;
39
40      wx = [0;0;gyro_data(:,1)];
41      wy = [0;0;gyro_data(:,2)];
42      wz = [0;0;gyro_data(:,3)];
43
44      %model of the filter
45
46      pitch = atan(-acc_data(:,1)./(sqrt(acc_data(:,2).^2 + acc_data(:,3).^2)));
47      roll  = atan(acc_data(:,2)./acc_data(:,3));
48
49      pitch = [0,0,pitch'];
50      roll =  [0,0,roll'];
51
52      pitch_est_comp=[0.1,0.1];
53      roll_est_comp=[0,0];
54      for   i=1:(length(acc_data(:,1))-1)
55      pitch_est_comp(i+2) = (1/delta_pit)*(Gamma_pit*pitch(i) +xi_pit*pitch(i+1) +
        eta_pit*pitch(i+2)+ sigma*(wy(i) - wy(i+2)) -alpha_pit*pitch_est_comp(i) - beta_pit
        *pitch_est_comp(i+1));
56      roll_est_comp(i+2) = (1/delta_roll)*(Gamma_roll*roll(i) +xi_roll*roll(i+1) +
        eta_roll*roll(i+2)+ sigma*(wy(i) - wy(i+2)) -alpha_roll*roll_est_comp(i) -
        beta_roll*roll_est_comp(i+1));
57      end
58
```

Snippet E.2: MatLab Code Used to Implement a Second Order Complementary Filter

E.1.3 Second Order Complementary Filter auto-tune code

```
function [Ki,Kp]=Tune_comp_filter(theta_input,acc_data,gyro_data,tau,ts,
ki_fun_input,kp_fun_input,grid_size)
%
%Inputs      :-
%theta_input : is the actual angle which one wants the filter to output (input is
 in rads)
%acc_data    : is an array of accelemeter data of all three axis (x,y,z)
%gyro_data   : is an array of gyroscope data of all three axis (x,y,z) (inputs is
 in rads/second)
%tau         : is the time delay of the complamentry filter
%ts          : as this is a discrete filter one needs to give the time
%              difference between each sample
%ki_fun_input: is an array containing the min and max values of which the
%              grid can use for Ki (sould be in the form [ki min, ki max])
%kp_fun_input: is an array containing the min and max values of which the
%              grid can use for Kp (sould be in the form [kp min, kp max])
%grid_size   : the grid
%
%Outputs     :-
%Ki          : returns the optiam Ki value for the given costs (RMSE and tau)
%Kp          : returns the optiam Kp value for the given costs

% the following defines the angle with repsct to the y-axis (pitch angle)
% as given by a three axis accelermeter and Euler (1-2-3) rotation from earth to
frame
pitch = atan(-acc_data(:,1)./(sqrt(acc_data(:,2).^2 + acc_data(:,3).^2)));

%values as measured by the gyroscope (note the non-lineartise are not adjusted
for)
wx = [0;0;gyro_data(:,1)];
wy = [0;0;gyro_data(:,2)];
wz = [0;0;gyro_data(:,3)];

pitch = [0,0,pitch'];

% setting up the grid as required by the search
kp_pit_array = linspace(kp_fun_input(1),kp_fun_input(2),grid_size);
ki_pit_array = linspace(ki_fun_input(1),ki_fun_input(2),grid_size);
previous_mean_error = 100000000;

% grid search code
num_times_changed = 1; % counter for the number of times kp and ki have changed
for k = 1: (length(kp_pit_array)-1)

kp_pit = kp_pit_array(k);

for j = 1: (length(ki_pit_array)-1)
```

```
43      ki_pit = ki_pit_array(j);
44
45      theta = theta_input ;
46
47      % the following are filter weighting terms
48      Gamma_pit = ki_pit*ts^2 - 2*kp_pit*ts;
49      alpha_pit = Gamma_pit + 4;
50      xi_pit    = 2*ki_pit*ts^2;
51      eta_pit   = ki_pit*ts^2 + 2*kp_pit*ts;
52      beta_pit  = xi_pit - 8;
53      delta_pit = eta_pit + 4;
54      sigma = 2*ts;
55
56      %reset angle at each loop (for robustnest to ensure no errors)
57      angle_est_comp=[0,0];
58
59      %calculation of the angle estimation (filtered anlge)
60      for  i=1:(length(acc_data(:,1))-1)
61      angle_est_comp(i+2) = (1/delta_pit)*(Gamma_pit*pitch(i) +xi_pit*pitch(i+1) +
    eta_pit*pitch(i+2)+ sigma*(wy(i) - wy(i+2)) -alpha_pit*angle_est_comp(i) - beta_pit
    *angle_est_comp(i+1));
62      end
63
64      %reduction of filtered angle so RMSE can be calculated
65      angle_est_comp = angle_est_comp(1:(length(theta)));
66
67      % the following removes any lag in the filter so one can plot the
68      % results on top of each other so RMSE can be calculated
69      [C21,lag1] = xcorr(angle_est_comp,theta);
70      [~,I1] = max(abs(C21));
71      t31 = lag1(I1);
72      if(t31 ==0)
73      t31 = 1;
74      end
75
76      angle_est_comp = angle_est_comp(t31:end);
77      theta = theta(1:(length(angle_est_comp)));
78
79      mean_square_error    = sqrt(sum((theta - angle_est_comp').^2)/length(theta));
80
81      tau_of_filter = 2/kp_pit; %time constant of the filter
82      if (mean_square_error <previous_mean_error && tau_of_filter < tau)
83
84      Kp = kp_pit;
85      Ki = ki_pit;
86      previous_mean_error = mean_square_error;
87      %disp(['Number of times Kp and Ki have changed', num2str(num_times_changed)])
88      num_times_changed = num_times_changed +1;
89      end
```

```
90
91      end
92      %disp(['interation of outer loop:', num2str(k)])
93      end
94
```

Snippet E.3: MatLab Code Used to auto-tuning of the Complementary Filter

E.2 Kalman Filter (MatLab Code)

E.2.1 Static Kalman Filter

```
function [K,F,H,filter_data]=kalman_filter_general(A_d,B_d,C_d,R,Q,u,z)

%inputs  :-
%A_d     : is the linear model of the state space system
%B_d     : is the linear model of the input impacts states
%C_d     : is the linear model of the measurements used by the filter
%          relates to the systems states (commonly denoted as H for kalman filters)
%R       : is the covariance matrix of the measurements (models the nosie of the
sensors)
%Q       : is the covariance matirx of the plantes      (modeld the noise in the
states)
%u       : is the plant input (can be in matrix form)
%z       : is the avaible state measurements
%
%outputs     :-
%K           : Kalman gain matrix
%F           : F is the matrix that relates the estamate to the system model
%             (see gordon's control notes)
%H           : H is the matrix that relates the estamate to the state measurement
%             (see gordon's control notes)
%filter_data : is the filtered state or estimated state if one preferes

xhat = [0;0];
P = 100*eye(size(A_d));

for i = 1:length(z)

%Predict
P_star = A_d*P*A_d' + Q;
x_star = A_d*xhat + B_d*u(i);

%correct
K = P_star*C_d*inv(C_d*P_star*C_d' + R);
xhat = x_star + K*(z(:,i) - C_d*x_star);
P = (eye(size(A_d)) - K*C_d)*P_star;

filter_data(:,i) = xhat;
end

F = (eye(size(A_d))-K*C_d)*A_d;
H = (eye(size(A_d))-K*C_d)*B_d;

```

Snippet E.4: MatLab Code Used to Implement a Static Kalman Filter

E.2.2 Kinematic Kalman Filter

```
function [Angle_est] =Kinematic_kalman(acc_data,gyro_data,ts,R,Q,P)
% A adtive Kalman Filter whos A (transition) matrix changes with. This transition matrix is made from the gyroscope outputs
% function [Angle_est] =Kinematic_kalman(acc_data,gyro_data,ts,R,Q,P)
%
% Inputs:-
%    acc_data    : A logged form of the accelerometer data (denoted $a_x$, $a_y$ and $a_z$ in this report, also this input has to be in an array formate)
%    gyro_data   : A logged form of the gyroscope data (denoted $w_x$, $w_y$ and $w_z$ in this report, also this input has to be in an array formate)
%    ts          : Sampleing of the controller used to collected the gyroscope and accelerometer data
%    R           : Is the covariance matrix of the measurements (models the nosie of the sensors)
%    Q           : Is the covariance matirx of the plantes      (models the noise in the states)
%    P           : Initial guess of the covariance estimation error
%
% Output:-
%    Angle_est       : Outputs an angle estimate for P,Q and R value

for i = 1:length(ax)
%Accelerometer measuremeterments
z = [ax(i);
ay(i);
az(i)];

%Skew matrix, which uses the gyroscope to model the trasistion matrix
A = [0           ,wz(i)    , -wy(i);
-wz(i)      ,0         , wx(i);
wy(i)       ,-wx(i)   ,0];

%Discretize the continuous time A matrix so a digital estimation of the angle can be acquired
A_d=0;
for q = 1:approx_to_matrix_exp
A_d = A_d + (eye(size(A))*A^(q-1)*Ts^(q-1))/(factorial(q-1));
end

%Predict
P_star = A_d*P*A_d' + Q;
x_star = A_d*xhat;

%correct
K = P_star*C_d*inv(C_d*P_star*C_d' + R);
xhat = x_star + K*(z - C_d*x_star);
P = (eye(size(A_d)) - K*C_d)*P_star;
```

```
41
42      %Estimation of the angle is saved
43      Angle_est(i)        = atan(-xhat(1)/(sqrt(xhat(2)^2 + xhat(3)^2 )))*180/pi;
44      end
45
```

Snippet E.5: MatLab Code Used to Implement a Kinematic Kalman Filter

Bibliography

[1] Analog Devices. Adxl345 (3-axis digital accelerometer datasheet). Available: http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf, (accessed Dec 15, 2014).

[2] Honeywell. Hmc5883l (3-axis gyroscope). Available: http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf, (accessed Dec 15, 2014).

[3] InvenSense. Itg-3200 (3-axis digital compass). Available: https://www.sparkfun.com/datasheets/Sensors/Gyro/PS-ITG-3200-00-01.4.pdf, (accessed Dec 15, 2014).

[4] Elec Freaks. Ultrasonic ranging module hc - sr04). Available: http://www.micropik.com/PDF/HCSR04.pdf, (accessed Feb 8, 2015).

[5] POLSTAR. Pmb -648 high sensitivity gps module. Available:http://www.soselectronic.com/a_info/resource/c/pmb648.pdf, (accessed Dec 15, 2014).

[6] J.G. Leishman. *Principles of helicopter aerodynamics*. Cambridge Univ Pr, 2010.

[7] K. Duttonet et al. *The art of control engineering*. Pearson, Tottenham court road, London, 1997.

[8] C. Gablehouse. *Helicopters and autogiros; a chronicle of rotating-wing aircraft*. Philadelphia, Lippincott, 1967.

[9] M. J. Cutler. Design and control of an autonomous variable-pitch quadrotor helicopter. Master's thesis, Massachusetts Institute of Technology (MIT), 2012.

[10] M. Cutler et al. Comparison of fixed and variable pitch actuators for agile quadrotors. *Navigation and Control Conference (GNC)*, page 763–770, 2011.

[11] M. D. Lellis and C. D. Oliveira. Modeling, identification and control of a quadrotor aircraft. Master's thesis, Czech Technical University in Prague, 2010.

[12] S. Lupashin et al. A simple learning strategy for high-speed quadrocopter multi-flips. *IEEE International Conference on Robotics and Automation (ICRA)*, page 1642–1648, 2010.

[13] D. Mellinger et al. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *In Int. Symposium on Experimental Robotics*, page 1642–1648. IEEE, 2010.

[14] M. Hehn and R. D'Andrea. A flying inverted pendulum. *IEEE International Conference on Robotics and Automation (ICRA)*, page 763–770, 2011.

[15] M.Y. Chen et. al. Designing a spatially aware and autonomous quadcopter. In *Systems and Information Engineering Design Symposium (SIEDS), 2013 IEEE*, pages 213–218. IEEE, 2013.

[16] G. d'Ambrosio and R. Navoni. Hg3 willy [online]. Available: https://www.youtube.com/watch?v=M4uXmekZk-4&feature=youtu.be, (accessed Dec 17, 2014).

[17] T. Lane. Operating drones safely: Follow the rules! [online]. Available: https://www.iaa.ie/news.jsp?i=507&gc=141&p=124&n=445&date_from=2000-01-01&date_to=2100-01-01, (accessed Dec 18, 2014).

[18] S. Brewster. Consumer drones are coming and they will change everything [article]. Available: http://technicsandtime.com/tag/quadrotor/, (accessed December 18, 2014).

[19] S. Brewster. Quadcopters, cameras, and the law [online]. Available: http://www.videouniversity.com/articles/quadcopters-cameras-and-the-law/, (accessed Dec 18, 2014).

[20] Amazon Prime Air. http://www.amazon.com/b?node=8037720011, Accessed Jan. 2015.

[21] Evan Ackerman. Matternet Wants to Deliver Meds with a Network of Quadrotors. Available: http://spectrum.ieee.org/automaton/robotics/medical-robots/

mini-uavs-could-be-the-cheapest-way-to-deliver-medicine, (accessed Feb 03, 2015).

[22] davidh5659. Herding cattle with a drone[online]. Available:https://www.youtube.com/watch?v=6ouXhsAEb5U#start=0:00;end=8:19;cycles=-1;autoreplay=false, (accessed Dec 17, 2014).

[23] FPSRussia. Prototype quad-rotor with machine gun [online]. Available:https://www.youtube.com/watch?v=SNPJMk2fgJU, (accessed Dec 18, 2014).

[24] David Hambling. Armed quadrotors are coming [online]. Available: http://www.popularmechanics.com/technology/military/planes-uavs/armed-quadrotors-are-coming-10720086, (accessed Dec 18, 2014).

[25] G.Hoffmann and H.Huang. *Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment*. American Institute of Aeronautics and Astronautics, 2007.

[26] P.Pounds et al. *Modelling and Control of a Quad-Rotor Robot*. IEEE, 2007.

[27] T. Bresciani. Modelling, identification and control of a quadrotor helicopter. Master's thesis, Lund University, 2008.

[28] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58:15–16, 2006.

[29] John L Crassidis. Sigma-point kalman filtering for integrated gps and inertial navigation. *Aerospace and Electronic Systems, IEEE Transactions on*, 42(2):750–756, 2006.

[30] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[31] K. Sprague et al. Design and applications of an avionics system for a miniature acrobatic helicopter. In *Digital Avionics Systems, 2001. DASC. 20th Conference*, volume 1, pages 3C5–1. IEEE, 2001.

[32] H. G. Min and E. T. Jeung. Complementary Filter Design for Angle Estimation using MEMS Accelerometer and Gyroscope. Available: http://ocw.mit.edu/

courses/new-courses/electrical-engineering-and-computer-science/resources/control-engineering/filtering, (accessed Dec 23, 2014).

[33] W. T. Ang et al. Physical model of a mems accelerometer for low-g motion tracking applications. In *International Conference on Robotics and Automation New Orleans*, page 1345–1351. IEEE, Apr 2004.

[34] Torvald Ersson and Xiaoming Hu. State observers of linear control systems with nonlinear outputs. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 4, pages 3358–3359. IEEE, 2001.

[35] Derek Rowell. Advanced system dynamics and control, discrete time observers and lqg control [handout]. Available:http://web.mit.edu/2.151/www/Handouts/Kalman.pdf, (accessed Jan 18, 2015).

[36] G.Welch and G. Bishop. An introduction to the kalman filter, 2001.

[37] Lai. Kok-Lam et al. New complex-step derivative approximations with application to second-order kalman filtering. In *AIAA Guidance, Navigation and Control Conference, San Francisco, California*, 2005.

[38] K. Kennedy, G. Lightbody, and R. Yacamini. Power systems harmonic analysis using the kalman filter. *PES General Summer Meeting*, July 2003.

[39] G. F. Franklin et al. Feedback control of dynamics systems. *Pretince Hall Inc*, 2006.

[40] Sherryl H. Stovall. Basic Interial Navigation, 1997.

[41] G. Cai et. al. *Unmanned rotorcraft systems*. Springer Science & Business Media, 2011.

[42] C. Jekeli. *Inertial navigation systems with geodetic applications*. Walter de Gruyter, 2001.

[43] Kinonix. Accelerometer errors [online]. Available: http://www.kionix.com/sites/default/files/AN012%20Accelerometer%20Errors.pdf, (accessed Dec 28, 2014).

[44] Ogata . K. *Discrete Control Systems.* NJ: Prentice-Hall, 1987.

[45] Phillips C. L. and Nagle H. T. *Digital Control System Analysis and Design.* NJ: Prentice-Hall, 2nd edn. edition, 1990.

[46] Karney, Charles FF. Transverse Mercator with an accuracy of a few nanometers. *Journal of Geodesy*, 85(8):475–485, 2011.

Week 1: 22nd - 28th September

11.0.1 Objectives

- Research Kalman Filters
- Understand mapping used for aeronautics (Euler angles matrix & Euler Rates Matrix)
- Equipment/Resources required for the project.

11.0.2 Kalman Filters

Kalman filtering, also known as linear quadratic estimation (**LQE)** is best described as a **recursive estimator**. The filter works by using a model of ones system [1] and the readings from a sensor(s). The filter works in two stages: *Predict* and *Update*. The first stage, the *Predict* stage uses the previous state ($\hat{\mathbf{x}}_{k-1|k-1}$)and set point input ($\mathbf{u}_k$)to make a prediction of the devices location. After the *Predict* stage the *Update* stage is carried out and uses the current measured data ($\hat{\mathbf{x}}_{k|k-1}$), the product of the Kalman gain $\mathbf{K}_k$ and prediction error ($\tilde{\mathbf{z}}_k$). [2]

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k \tag{11.1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{z}}_k \tag{11.2}$$

$$\tilde{\mathbf{z}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \tag{11.3}$$

The performance of this filter depends heavily upon the accuracy of **Q** (Covariance matrix for the measurement noise) and **R** (Covariance matrix for the disturbances). Note **R** can often be intelligently estimated from knowledge of the system under control. But the **Q** is more of a problem and often very little real information will be known. Therefore **Q** is often guessed and **R** and **Q** are tuned together to get an adequate result

[1] Kinematics equations which can model the flight of a Quad-rotor
[2] See pg 480 of the 'The art of control engineering' for more information.

for the control of the device.

11.0.3 Euler Angles and Rotational Matrices

To describe the orientation and position of objects in space (parts, tools, aircraft, etc) one attaches a coordinate system to each object and then one gives a description of one coordinate system relative to another. One assumes the existence of a *universal* coordinate system w.r.t which other Cartesian system can be defined.

This report shall be using Tait–Bryan angles. This angling system uses three perpendicular axis of rotation: Roll, Pitch and Yaw which are denoted by (θ, ϕ, ψ) and are rotations about the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$-axes respectfully. This system is used over the Euler angle system because Tait–Bryan angles don't rely on the same rotation twice (e.g in **Z-Y-Z** one relies upon a Roll-Pitch-Roll type axes, thus one has to use a Roll type axis twice).

The following equation relates a body **b** in free space to another point in space **w** which in this case is a fixed point on the world.

$$Rot(\theta, \phi, \psi) = \mathbf{R}(\theta)\mathbf{R}(\phi)\mathbf{R}(\psi) \tag{11.4}$$

Now using the following rotational matrices around the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ axes respectively, one can relate any body **b** in free space to the a fixed frame **w**.

$$\mathbf{R}_1(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\alpha & S_\alpha \\ 0 & -S_\alpha & C_\alpha \end{bmatrix} \tag{11.5}$$

$$\mathbf{R}_2(\alpha) = \begin{bmatrix} C_\alpha & 0 & -S_\alpha \\ 0 & 1 & 0 \\ S_\alpha & 0 & C_\alpha \end{bmatrix} \tag{11.6}$$

$$\mathbf{R}_3(\alpha) = \begin{bmatrix} C_\alpha & S_\alpha & 0 \\ -S_\alpha & C_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11.7}$$

Using the above (11.4) to (11.7) one can generate the *Euler Rates Matrix* which

relates the angler change $\dot{\theta}$ to the angler velocity $\boldsymbol{\omega}$. [3]

$$\boldsymbol{\omega} = \dot{\psi}\hat{\boldsymbol{n}}_{\boldsymbol{3}} + \dot{\theta}\hat{\boldsymbol{b}}'_{\boldsymbol{2}} + \dot{\phi}\hat{\boldsymbol{b}}_{\boldsymbol{1}} \tag{11.8}$$

11.0.4 Required Resources

After reading other Quad-Rotor reports a list of required materials was made up and items already purchased were removed. The following is a list of items which will be needed in the future to complete the project.

Required Materials

- Parallax Propeller.
- Propeller Plug (FTDI, USB to SERIAL chip)
- SD Card.
- SD Card reader.
- Crystal Oscillator (5 MHz)
- Bread/Strip board.
- 3.3 to 5 volt level shifter.

[3] both of this books where used for understanding of frames [41] and [42, pg. 2-27]

Week 2: 29th Sep - 5th Oct

12.1 Objectives

- Understand/Derive Euler angles matrix
- Read up on datasheets for the 9DOF and the I2C protocol

12.2 Research Carried out

12.2.1 Derivation of Euler angle matrix

Last week the rotation matrix was defined using (11.4). If this is solved for a Euler rotation of **XYZ** one will get the following

$$Rot(\phi,\theta,\psi) = \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix}}^{\text{C}} \overbrace{\begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix}}^{\text{B}} \overbrace{\begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{A}} \tag{12.1}$$

$$= \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\theta S_\phi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\theta C_\phi \end{bmatrix} \tag{12.2}$$

Now letting $\hat{e}_i$ be the i^{th} unit vector, the function that maps an Euler angle vector to it's corresponding Euler angle Rates matrix, $E : \mathbb{R}^3 \to \mathbb{R}^{3x3}$, is

$$\mathbf{E}_{ijk}(\theta,\phi,\psi) = [\mathbf{R}_k(\psi)^T \mathbf{R}_j(\theta)^T \hat{\mathbf{e}}_i, \mathbf{R}_k(\psi)^T \hat{\mathbf{e}}_j, \hat{\mathbf{e}}_k] \tag{12.3}$$

Using the notation in (12.1) one will get the following equation.

$$\mathbf{E}_{ijk}(\theta,\phi,\psi) = \mathbf{A}^T\mathbf{B}^T\hat{\mathbf{e}}_\phi + \mathbf{A}^T\hat{\mathbf{e}}_\theta + \hat{\mathbf{e}}_\psi \tag{12.4}$$

After one has solved (12.4) for the Euler angle Rates matrix the angular velocity of the frame can then be found using the following.

$$\omega = \mathbf{E}_{ijk}(\mathbf{u})\dot{\mathbf{u}} \tag{12.5}$$

12.2.2 I²C Protocol and SEN-10324 (9DOF)

I²C most commonly uses 7 bit address. When one sends out an address one uses 8 bits; 7 bits are used to assign the address and the last bit is used to inform the slave if it is reading or writing. Note the read/write bit is the LSD (Least Significant Bit)

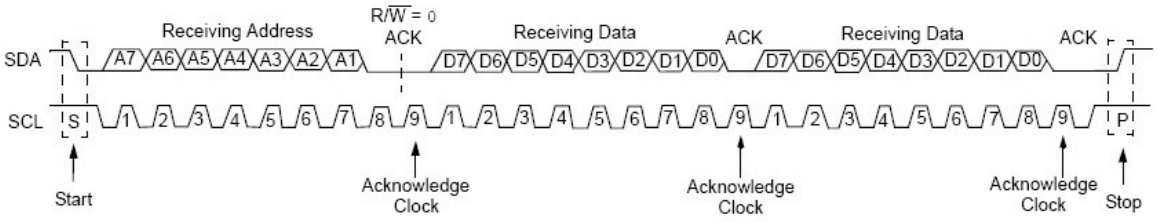


Figure 12.1: Typical I²C write transmission (7-Bit Address)

When communicating with a device one will have to send a 8 bit packet. After the 8 bit a 9 bit is sent to acknowledge that the device has established a connection for communication. Note one can easily they if the master is reading from or writing to a device. If the 8 bit address is odd the master reading only and if the 8 bit address is even the writhing only.

Writing to a Slave Device

1. Send a start Sequence
2. Send the I²C address of the slave with R/W both low (even address)
3. Send the internal register number one wants to write to.
4. Send the data byte.
5. **Optional, send any further data bytes.**
6. Send the stop sequence.

Reading from Slave Device

1. Send a start Sequence
2. Send 0x53 (I^2C address of the ADXL345 (accelerometer) with the R/W bit low (even address)
3. Send 0x00 (Internal address used for device ID check)
4. Send a start sequence again (repeated start)
5. Send 0x53 (I^2C address of the ADXL345 with the R/W bit high (odd address)
6. Read data byte from ADXL345
7. Send the stop sequence.

The 9DOF used in this project[1] uses an accelerometer by Analog Devices a digital Magnetometer by Honeywell and a gyroscope by InvenSense.

12.2.3 ADXL345 Accelerometer

The I^2C address for this devices is Ox53 (followed by the R/$\bar{W}$ bit). This translates to OxA6 for the write and OxA7 for a read (See page 10 of data sheet[2])

12.2.4 HMC5883L Magnetometer

The I^2C address for this devices is Ox1E. This translates to Ox3C for write and Ox3D for a read (See page 17 of data sheet[3])

12.2.5 ITG-3200 Gyroscope

The I^2C address of this device is Ox68 as the A_0 is tied to ground. This translates to OxD0 for the write and OxD1 for a read (See page 18 of data sheet[4])

[1] https://www.sparkfun.com/products/10724
[2] https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf
[3] https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf
[4] https://www.sparkfun.com/datasheets/Sensors/Gyro/PS-ITG-3200-00-01.4.pdf

Week 3: 6^{th} - 12^{th} Oct

13.1 Objectives

- Begin Modelling
- Begin I^2C communication with the 9DOF

13.2 Modelling

To create an accurate model for the control of the quad-rotor physical measurements of the quad-rotor had to be carried out. The that have to be measured are as follows the motor gain and time constants (K_f andτ) , the inertia of the three axes (J_θ, J_ϕ J_ψ).

To ensure that the model is complete enough for a first pass model one would also need to measure the damping coefficient β and the motor drag coefficient D. The damping coefficient can be found while doing the moment of inertia tests while the drag can be found from doing step response on the motor and back solving to find D by using the general motor equation. [1] [2]

$$J\ddot{\Theta} = K_f i - \beta\dot{\Theta} - D \tag{13.1}$$

If one sets i to zero in (13.1) one can find the drag coefficient (D) if the damping factor (β) is known. Also the mass of the quad-rotor must be found, this can easily found using a spring balance or strain gauge.

13.2.1 Motor and ESC Speed Test

The main focus for this week was to collect data that would allow the creation of a accurate model of the quad-rotor. This meant that trust and speed data would have to be collected so characteristic curves could created. But during the initial test of all four motors there was huge a huge difference between motor 4 and the rest of the motor characteristic responses. This meant that all four motor would have to be

[1] Link to explanation of how brushless DC motor workhttp://educypedia.karadimov.info/library/ems_ch12_nt.pdf

[2] Link to a control assignment of a brushless DC motor (similar to the assignment done in third year) http://support.ctc-control.com/customer/elearning/younkin/driveMotorEquations.pdf

recalibrated to ensure that each motor would have the same characteristic responses (within margins of error).

The results of the input control signal (PWM) vs speed can be seen in figure 13.1 (Note the PWM used in this project has a period of 20 ms with a 1-2 ms pulse duration. Therefore the Duty cycle of the PWM signal is 5 - 10 %). The graphs that were generated were linear as expected and were plotted in Matlab with a trend line to find the slope of the best fit line for the data sets that were taken for each motor (using ployfit command). This best fix line allows the DC motors to be normalized so that one motor model can be used for the simulation. Note that in order to get a stable aggressive control the motors will have to be reverted back to there accurate model before the code is written in C++ to ensure the best possible control of the quad-rotor.

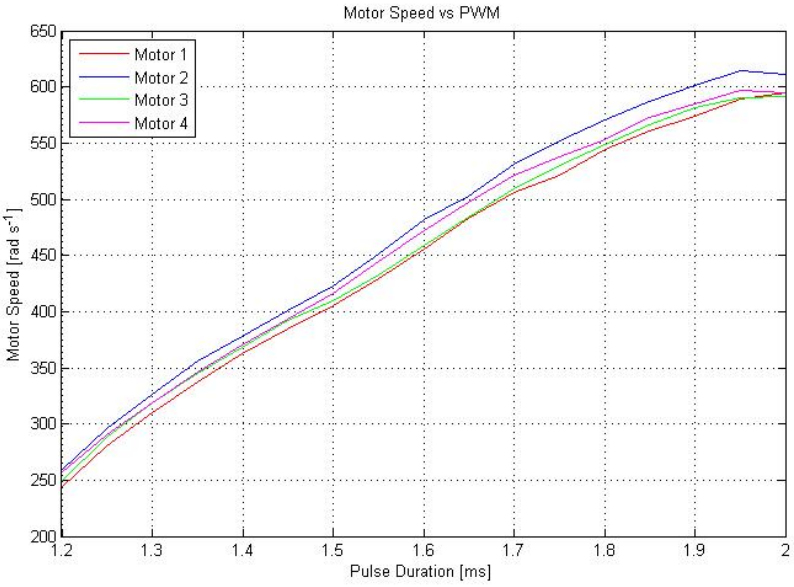


Figure 13.1: PWM pulse duration vs Motor Speed

13.2.2 Quad-Copter weight

The mass of the quad-rotor was acquired using a strain gauge. The mass of the system was found to be 2.26 kg. Thus the motors will have to supply 22.17 N of thrust to keep the quad-rotor to hovering in the same position. This means that each motor will have to supply at least 5.5425 N (0.565 kg) of thrust to keep the quad-rotor a

hovering in free space.

13.2.3 Motor and Propeller Thrust

The thrust tests were carried out using the strain gauge. The strain gauge gave out mass in kg's so the output had to be scaled by gravity (9.81 $\mathrm{m\,s^{-2}}$). The operating point for the quad-rotor was found to be 1.7 ms as seen in figure 13.2. Note this was different to the operating point found by Brendan Barry. Note fro figure 13.2 there is a head room of 10 N. This means the maximum angle the quad-rotor can go through is 45º so as to allow the quad-rotor some degree of manoeuvrability in 3d space. But if the limit the angle to 20º, the quad-rotor will be capable of lifting 0.8 kg.

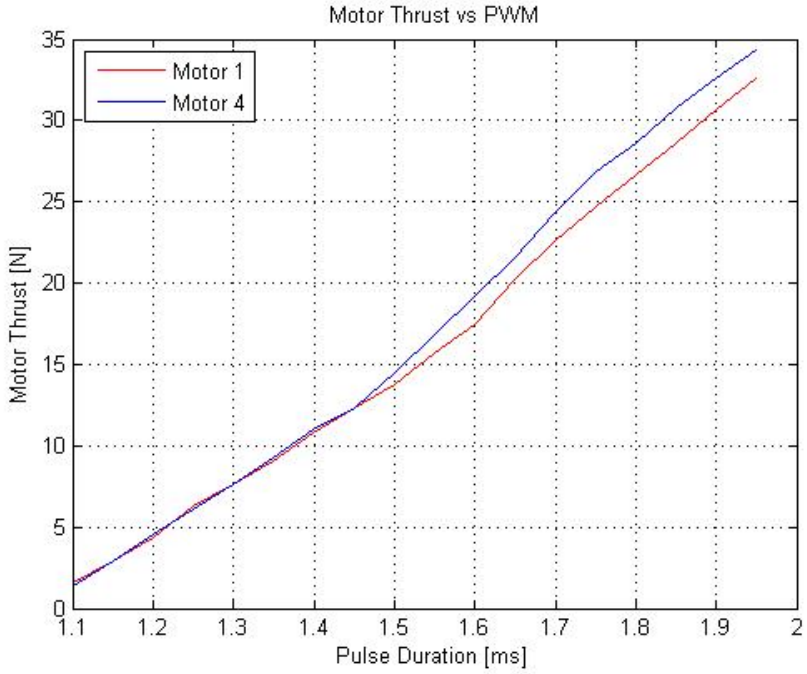


Figure 13.2: PWM pulse duration vs Motor Thrust

13.3 Communication with 9DOF

Basic communication with the 9 DOF was accomplished using the i2c_in (used to write to the device) and i2c_out (used to read from the device) during the week. There was problems communicating with the accelerometer as register 2D was entering sleep mode randomly after a revision to the code was made. The register would go from 0x08 (measurement mode) to 0x00 (wakeup mode) and the x-axis, y-axis and z-axis would then read back 0,0,0 respectfully. This problem has be insulated and hopefully

code for reading the 9DOF will be in the following week.

Week 5: 20^{th} - 26^{th} Oct

14.1 Objectives

- Find the moments of inertia of the about all three axes.

14.2 Inertia about each axis

To find the moment of inertia of the quad-rotor the rig was held by cable at it's end point and allowed to swing about a single axis. To document the amplitude of the quad-rotor at each oscillation a laser pointer was attached to the end of the quad-rotor [1]and to record the period of each oscillation a stop watch [2]. The oscillation was carried out about a low friction surface to ensure an accurate value for the moment of inertia of each axis. The results were entered into Matlab and the following results were acquired for the moment of inertia of the system.

$$I = \begin{bmatrix} I_\phi & 0 & 0 \\ 0 & I_\theta & 0 \\ 0 & 0 & I_\psi \end{bmatrix} = \begin{bmatrix} 0.1274 & 0 & 0 \\ 0 & 0.1221 & 0 \\ 0 & 0 & 0.2370 \end{bmatrix} \tag{14.1}$$

Note as the system is symmetric about the pitch and roll axis one can assume that $I_\phi =\approx I_\theta$.

14.3 Damping

The results for the maximum amplitude were processed using Matlab and the results were plotted yielding figure 14.1.

From figure 14.1 one can acquire the moment of inertia from the slope of the line and thus define the inertia of the system as follows.

[1]This was done so one could acquire the damping of the system about each axis after the test was completed

[2]This was done so one could a quire the moment of inertia about each axis

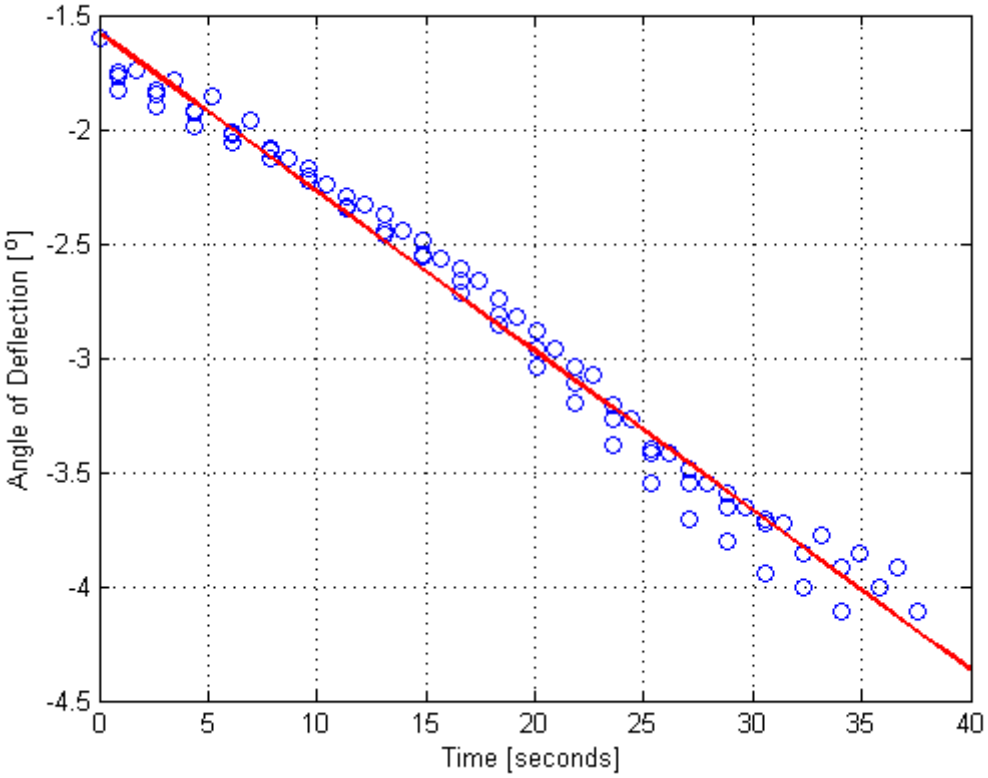


Figure 14.1: Plot of deflection from normal vs Time

$$\beta = \begin{bmatrix} \beta_\phi & 0 & 0 \\ 0 & \beta_\theta & 0 \\ 0 & 0 & \beta_\psi \end{bmatrix} = \begin{bmatrix} 0.0194 & 0 & 0 \\ 0 & 0.0194 & 0 \\ 0 & 0 & \approx 0 \end{bmatrix} \tag{14.2}$$

Week 6: 27th Oct - 2nd Nov

15.1 Objectives

- Create the state space equations for the roll and pitch.
- Make the Simulink model for the Kalman Filter.

15.2 Derivation of state space equations for Kalman filter

The following kalman equation was taken from the [7] and can be seen on page 483 and the derivation of the equation can be seen on page 791 of the same book

$$\hat{\mathbf{x}}_{k+1|k+1} = \Phi\hat{\mathbf{x}}_{k|k} + \Delta\mathbf{u}_k + \mathbf{K}[\mathbf{z}_{k+1} - \mathbf{C}(\Phi\hat{\mathbf{x}}_{k|k} + \Delta\mathbf{u}_k)] \tag{15.1}$$

15.2 is the same 15.1 but the matrix/variables are grouped together for ease of computation.

$$\hat{\mathbf{x}}_{k+1|k+1} = [\mathbf{I} - \mathbf{KC}][\Phi\hat{\mathbf{x}}_{k|k} + \Delta\mathbf{u}_k] + \mathbf{K}\mathbf{z}_{k+1} \tag{15.2}$$

15.2 can be written as follows.

$$\hat{\mathbf{x}}_{k+1|k+1} = F\hat{\mathbf{x}}_{k|k} + H\mathbf{u}_k + \mathbf{K}\mathbf{z}_{k+1} \tag{15.3}$$

Where in 15.3 $F = [\mathbf{I} - \mathbf{KC}]\Phi$; $H = [\mathbf{I} - \mathbf{KC}]\Delta$. Thus each matrix is made up of a mix of prediction and current value with the exception of the Kalman gain $\mathbf{K}$.

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{bmatrix}_k + \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \mathbf{u}_k + \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}_{k+1} \tag{15.4}$$

Where: $\mathbf{x_1}$ is angular velocity α and $\mathbf{x_2}$ is angular acceleration $\dot{\alpha}$

15.3 Simulink model of Kalman filter

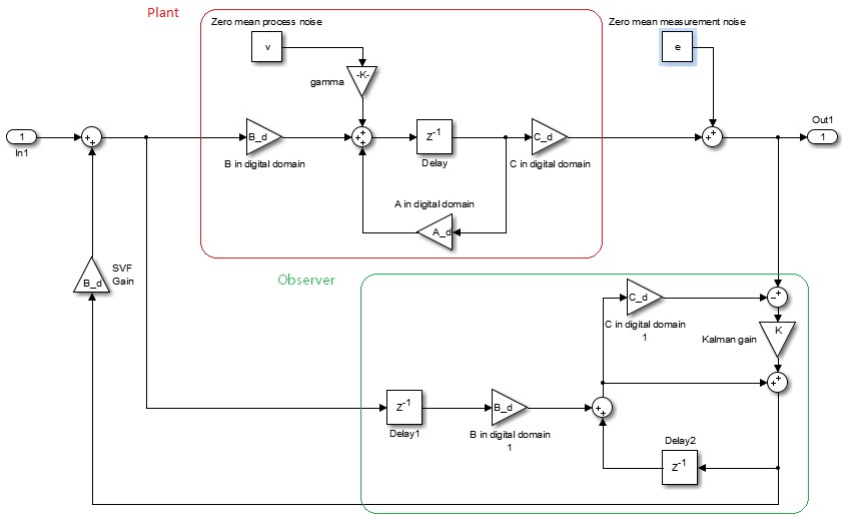


Figure 15.1: Full Kalman Filter Model with plant as seen in [7] pg 483

The Simulink model in figure 15.2 is the observer model used in this project and how it interacts with the plant can be seen in 15.1. Note in this project the Kalman gain **K** is set and is not adjusted during flight.

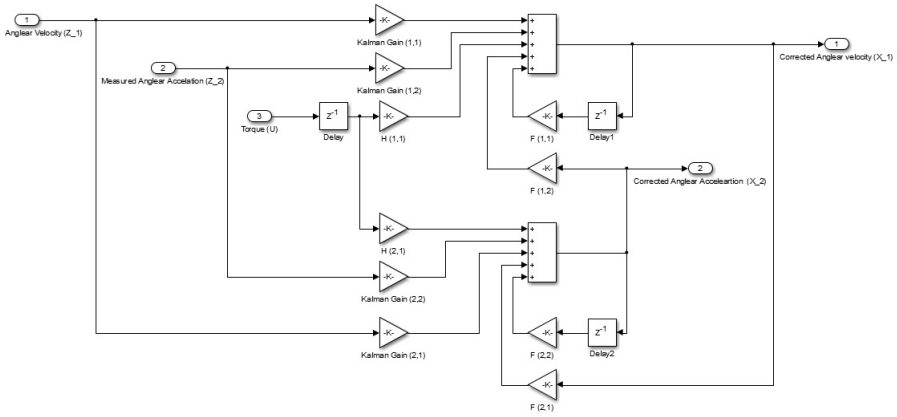


Figure 15.2: Simplified model of the Kalman filter

Week 7: 3rd - 9th Nov

16.1 Objectives

- Read up on methods to remove bias from sensors.
- Analysis and remove offset/mean from accelerometer and gyroscope.
- Come up with a method to find the **R** matrix of the system.

16.2 Accelerometer Bias

Due to the fact of the large bias on the accelerometer a method had to be found in order to reduce it, a better model made to be found or the Kalman filter had to be remodelled. The method agreed approved to deal with this problem and hopefully the correct one is the methods discussed in the following papers [33, 43]. Upon using the sensors there were sensor bias error, however, the sensors can read something different that the required (ideal) at a given location due to many factors including mechanical tolerances in the component parts.

The way to measure and remove bias error from the accelerometer is to do the following:-

1. Place the sensor module on an approximately horizontal surface (table top, granite flat, etc.).

2. Rotate the sensor module 90^o to each of the positions shown in Figure

3. Read the outputs on each axis in each of the six positions.

4. Calculate the sensitivities (where the first subscript is the sensor, the second subscript is the position):

$$S_{xx} = \frac{(a_{x2} - a_{x4})}{2}; \quad S_{yy} = \frac{(a_{y1} - a_{y3})}{2}; \quad S_{zz} = \frac{(a_{z5} - a_{z6})}{2}; \tag{16.1}$$

5. Calculate the sensor module biases:

$$B_x^{0g} = \left(\frac{a_{x1} + a_{x3} + a_{x5} + a_{x6}}{4}\right); \quad B_y^{0g} = \left(\frac{a_{x2} + a_{x4} + a_{x5} + a_{x6}}{4}\right);$$
$$B_z^{0g} = \left(\frac{a_{x1} + a_{x2} + a_{x3} + a_{x4}}{4}\right); \quad (16.2)$$

6. Save B_x^{0g}, B_y^{0g}, B_z^{0g}, S_{xx}, S_{yy}, and S_{zz} local and use these values in all subsequent calculations of the acceleration to get the correct outputs.

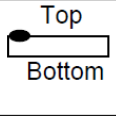
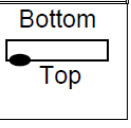
Position	1	2	3	4	5	6
Diagram						

Figure 16.1: Position in which one has to measure the bias for calibration

16.3 Formula to measure angle

16.3.1 Measuring Tilt Angle using One Axis

Accelerometers measure acceleration. That is acceleration due to movement and also acceleration due to gravity. If you want to measure tilt in both x and y axis with a 2-axis accelerometer then you can simply use $\sin x$ where x is the output from one axis of the accelerometer. Remember that beyond +45 and -45 degrees the accuracy will diminish

16.3.2 Measuring Tilt Angle using Three Axis

For accurate measurements of tilt in both the x & y planes the following formulas have to be used[1] .

$$\theta = \arctan\left(\frac{a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right)$$
$$\phi = \arctan\left(\frac{a_y}{\sqrt{{a_x}^2 + {a_z}^2}}\right) \quad (16.3)$$

[1]See http://www.hobbytronics.co.uk/accelerometer-info for more details

Week 1: 19th - 25th January

17.0.1 Accelerometer model

In order to implement the complementary filter a model of the accelerometer had to be created that links the data output of the device to the actual angle of the quad-rotor, hence the following equations where developed.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = g \begin{bmatrix} -S_\theta \\ C_\theta S_\phi \\ C_\theta C_\phi \end{bmatrix} + \begin{bmatrix} \mu_{m-x} \\ \mu_{m-y} \\ \mu_{m-z} \end{bmatrix} + \begin{bmatrix} b_{a-x} \\ b_{a-y} \\ b_{a-z} \end{bmatrix} \tag{17.1}$$

Now if one assumes that the bias and noise on the accelerometer is zero (can be removed in code) one can find the angle of the Quad-rotor in the NED frame . Hence the ϕ and θ can be defined as follows:-

$$\begin{aligned} \phi &= \arctan\left(\frac{a_y}{a_z}\right) \\ \theta &= \arctan\left(\frac{-a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right) \end{aligned} \tag{17.2}$$

17.0.2 Gyroscope model

A similar model for the gyroscope was created as the angular velocity measured by the gyroscope does not correspond to the Euler angle rates $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^\intercal$. Instead one can define the rate of change of angle with respect to the earth frame NED as follows:-

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & S_\phi C_\theta \\ 0 & -S_\phi & C_\phi C_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{17.3}$$

Now taking the in the inverse one gets the following:-

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{17.4}$$

17.1 Complementary Filter

A pair of filters are called complementary filters if their transfer functions sum to one at all frequencies in a complex sense, i.e. the phase is zero and the magnitude is one 17.5.

$$G_1(s) + G_2(s) = 1 \tag{17.5}$$

17.1.1 First Order Complementary Filter

To start with a first order complementary filter was design using (17.6). This first order filter had adequate results.

$$\theta_f = \underbrace{\frac{1}{1+\tau s}}_{G_1(s)} \theta_a + \underbrace{\frac{\tau s}{1+\tau s}}_{G_2(s)} \frac{1}{s} \omega_g \tag{17.6}$$

Using (17.2), (17.4) and(17.6) the high frequency components of the estimated orientation angles where found by a high-pass filter on the gyroscope data:-

$$\dot{\phi}_{hp} = \omega_x + S_\phi T_\theta \omega_y + C_\phi T_\theta \omega_z - \frac{\phi_{hp}}{\tau} \tag{17.7}$$

$$\dot{\theta}_{hp} = C_\phi \omega_y - S_\phi \omega_z - \frac{\theta_{hp}}{\tau} \tag{17.8}$$

Similarly the low-pass equivalents are found using the following:-

$$\dot{\phi}_{lp} = \frac{1}{\tau} \left[\arctan\left(\frac{a_y}{a_y}\right) - \phi_{lp} \right] \tag{17.9}$$

$$\dot{\theta}_{lp} = \frac{1}{\tau} \left[\arctan\left(\frac{-a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right) - \theta_{lp} \right] \tag{17.10}$$

17.1.2 Difference Equations for first order filter

(17.6) was emulated into the digital domain using Tustain's approximation to integration. Tustain's method was chosen as it ensures a stable mapping into the discrete domain (maps to inside the unit circle) if the continues function is stable, forward rectangular rule and backward rectangular rule do not ensure this. [1].

$$\theta_f[k] = \frac{1}{T_s + 2\tau}(T_s(\theta_a[k] + \theta_a[k-1] + \tau(\omega_g[k] + \omega_g[k-1])) - (T_s - 2\tau)\theta_f[k-1]) \quad (17.11)$$

Note it can be easily shown that if there is a bias b_{a-x} on the x-axis of the gyroscope then there will be an offset angle when one integrates the gyroscope reading. For the first order filter this offset (for the x-axis)can be approximated by use of the filter time constant and is defined as follows:-

$$\theta_{offs} \approx \tau b_{a-x}$$

17.1.3 Second Order Complementary Filter

A complementary filter is easily derived by solving the transfer function of the Mahony & Madgwick filter for the angle θ, which yields

$$\theta_f = \underbrace{\frac{K_i + K_p s}{K_i + K_p s + s^2}}_{G_1(s)} \theta_a + \underbrace{\frac{s^2}{K_i + K_p s + s^2}}_{G_2(s)} \frac{1}{s} \omega_g \quad (17.12)$$

Obviously, and not unexpectedly, this complementary filter is build from 2nd order filters. Note that the filter acting on the acceleration data actually consists of a low-pass plus band-pass filter.

This result has interesting consequences. Being 2nd order filters, the frequency response of the acceleration and rotation rate filters are characterized by the resonance frequency and damping factor

$$\omega_0 = \sqrt{K_i} \qquad \xi = \frac{K_p}{2\sqrt{K_i}} \quad (17.13)$$

[1] See http://web.cecs.pdx.edu/~tymerski/ece452/6.pdf for more details

The damping factor determines the overshoot at the resonance frequency. For high-pass (and low-pass) filters the frequency response is flat (and the step response non-oscillatory) for $\xi \geq 1$. This suggests the criterion [2].

$$K_i \leq \frac{1}{4}K_p^2 \tag{17.14}$$

To find the a method of modelling the filter in MatLab $G_1(s)$ and $G_2(s)$ where arranged as follows:-

$$\chi_{hp} = \frac{1}{s}\left[\omega_g - \chi_{hp}\left(\frac{K_i}{s} + K_p\right)\right] \tag{17.15}$$

$$\chi_{lp} = \frac{1}{s}\left[(\theta_a - \chi_{lp})\left(\frac{K_i}{s} + K_p\right)\right] \tag{17.16}$$

Note (17.15) and (17.16) can be combined to yield the following filter (as $\chi_{hp}+\chi_{lp} = \theta_f$), which can also be modelled in MatLab.

$$\theta_f = \frac{1}{s}\left[\omega_g + \left(\frac{K_i}{s} + K_p\right)(\theta_f - \theta_a)\right] \tag{17.17}$$

17.1.4 Difference Equations for second order filter

(17.15), (17.16) were emulated into the digital domain using Tustain's method, which yielded the following equations:-

$$\chi_{hp}[k] = \frac{1}{\eta + 4}\left(2T_s(\omega_g[k] - \omega_g[k-2]) - (\Gamma + 4)\chi_{hp}[k-2] - (\xi - 8)\chi_{hp}[k-1]\right) \tag{17.18}$$

$$\chi_{lp}[k] = \frac{1}{\eta + 4}\left(\Gamma\theta_a[k-2] + \xi\theta_a[k-1] + \eta\theta_a[k] - (\Gamma + 4)\chi_{lp}[k-2] - (\xi - 8)\chi_{lp}[k-1]\right) \tag{17.19}$$

where:-

$$\eta = K_i{T_s}^2 + 2K_pT_s; \quad \Gamma = K_i{T_s}^2 - 2K_pT_s; \quad \xi = 2K_i{T_s}^2$$

Note (17.18) and (17.19) can be combined to give the following:-

[2] See http://www.olliw.eu/2013/imu-data-fusing/ for more details

$$\theta_f[k] = \frac{1}{\eta+4}(\Gamma\theta_a[k-2] + \xi\theta_a[k-1] + \eta\theta_a[k] + 2T_s(\omega_g[k] - \omega_g[k-2]) - (\Gamma+4)\theta_f[k-2] - (\xi-8)\theta_f[k-1]) \quad (17.20)$$

Week 2: 26th January - 1st February

18.1 Objectives

- Acquire an accurate Drag Coefficient (d) so as the Yaw axis could be controlled correctly.
- Help with the design of the Integral Action on the Roll and Pitch axes.
- See if one can use system identification (least squares) to tune the complementary filter.
- Design an algorithm to self tune the Complementary Filter.

18.2 Drag coefficient Test

Upon reviewing the drag coefficient from previous years it was suspected that it was incorrect as it was a hundred times less than the drag coefficients (d) found in similar quad-rotor project reports (see any of the reports in the current research section in the introduction). To measure the force produced by the quad-rotor a strain gauge was attached to an end point of the quad-rotor and the distance from the point of rotation to the point of attachment was recorded. Two motor where then turned on and the force in the ψ-direction was measure by means of the strain gauge. Note the measurement on the strain is in kg so it was scaled by 9.81 m s^{-2} to get force, this was taken to be correct as the stain gauge pre-calibrates by using gravity to related the force applied to it to mass acting on it. The motor speed ω was also measured using an optical tachometer.This was repeated for several PWM inputs to the motors. These experimental values were then used to calculate drag. Using the following one can relate the drag coefficient (d) to force and ω :-

$$\begin{aligned} \tau_d &= d\omega^2 \\ \tau &= r \times F \\ \tau_d &= \frac{r \times F}{\omega^2} \end{aligned} \tag{18.1}$$

18.3 Design of Integral Action using DLQR

The group felt that Integral Action was needed on the θ & ϕ axes, thus Integral Action was required as a pre-compensator is not as robust in the face of modelling errors as in Integral Action. Note these modelling errors poses problems when one is designing controllers using the LQR method as the performance will not be as "Optimal" as the design predicts, but that is a discuss that is far to long to be featured here [1]. There are many ways to achieve integral control, in this report it was decided to add extra states to the system model which are the time integrals of the real states. If the extra state is called z (not to be confused with the sensor measurement z_k) then.

$$z = \int x \ \text{dt} \quad \text{or} \quad \dot{z} = x \tag{18.2}$$

In the discrete time (18.2) can be written as follows:-

$$z_{k+1} = z_n + T_s x_n \tag{18.3}$$

With the addition of this state the system can now be written as follows:-

$$\begin{bmatrix} x \\ z \end{bmatrix}_{k+1} = \begin{bmatrix} \Phi & 0 \\ CT_s & I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}_k + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k \tag{18.4}$$

Hence (18.4) can be written as follows, where the tildes represent the augmented vector and matrices:

$$\tilde{x}_{k+1} = \tilde{\Phi}\tilde{x}_n + \tilde{B}u_k \tag{18.5}$$

Hence using the DLQRoptimal control approach, one will get a matrix of optimal gains that satisfy the following control law:-

$$u_n = -\begin{bmatrix} K_k^p & K_k^I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}_k \tag{18.6}$$

Note as the above problem is a discrete time problem one has to use the following DLQR cost function:-

[1] For more details please see Chapter 12 of [7], a great introduction to optimal control, and for more information please read up on *Discrete Matrix Riccati Equation (DMRE)*. See [44] Section 7.3 and [45] Section 10.6 for more detail on DMRE

$$J_k = \frac{1}{2}\sum_{n=0}^{k-1}(x_n^\intercal Q x_n + u_n^\intercal R u_n) + \frac{1}{2}x_n^\intercal Q_n x_n \tag{18.7}$$

18.4 Tuning of the Complementary Filter using Least Squares

A least squares method of tuning the filter was investigated this week. Using (17.17) one can tune the filter in the continues domain and then emulate it across. This can be done by making the following assumption:-

$$\theta_f = \theta_p \tag{18.8}$$

where θ_p is the angle of the system as given by the potentiometer. Note in order to do this one also needs access to the rate of change of potentiometer angle as denoted as $\dot{\theta}_p$. Hence one can develop a method of tuning the filter as follows:-

$$\begin{aligned}
\theta_p &= \frac{1}{s}\left[\omega_g - \left(K_p + \frac{K_i}{s}\right)(\theta_p - \theta_a)\right] \\
s\theta_p &= \left[\omega_g - \left(K_p + \frac{K_i}{s}\right)(\theta_p - \theta_a)\right] \\
\dot{\theta}_p &= \left[\omega_g - \left(K_p + \frac{K_i}{s}\right)(\theta_p - \theta_a)\right] \\
\dot{\theta}_p - \omega_g &= -\left(K_p + \frac{K_i}{s}\right)(\theta_p - \theta_a) \\
\underbrace{\dot{\theta}_p - \omega_g}_{B_f} &= \underbrace{\left[(\omega_g - \theta_p) + \frac{(\omega_g - \theta_p)}{s}\right]}_{A_f}\begin{bmatrix} K_p \\ K_i \end{bmatrix}
\end{aligned} \tag{18.9}$$

Hence using the equation developed in (18.9) one can tune a continues time complementary filter of this form using the following:-

$$\begin{bmatrix} K_p \\ K_i \end{bmatrix} = (A_f^\intercal A_f)^{-1} A_f^\intercal B_f \tag{18.10}$$

Note the method of tuning the filter as shown in (18.10) was not used in this project as the grouped didn't know $\dot{\theta}_p$, instead an algorithm was create to tune it which is

notes in the following section

18.5 Algorithm used to self tune the Complementary Filter

Due to the fact that the group had access to the"actual" angle θ_p of the device as potentiometer. It was decided to do a search using a RMSE (18.11) with a cross-correlation used to find the delay between the signals so it can be removed when checking the RMSE term.

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n}(\theta_f - \theta_p)^2}{n}} \tag{18.11}$$

The cross-correlation function is defined in (18.12) and is commonly used in signal processing to find the measure of time-lag between to signals, hence it's use here.

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m]g[m+n] \tag{18.12}$$

Algorithm 2 Auto Tuning a Complementary Filter

procedure GRID SEARCH TUNING OF A COMPLEMENTARY FILTER
    $K_{p(max)} = 50$;
    $K_{i(max)} = 50$;
    Grid size = 1000;
    Previous Error = 100000000;
    $K_{p(it)}$ =linspace(0.1,$K_{p(max)}$,Grid size);
    $K_{i(it)}$ =linspace(0.1,$K_{i(max)}$,Grid size);
    for i := 1: (length($K_{p(it)}$)-1) **do**
        for j := 1: (length($K_{i(it)}$)-1) **do**
            θ_f = Filter output using (17.20)
            signal delay = Output of equation (18.12) using θ_f and θ_p as inputs.
            $\theta_f = \theta_f$(signal delay :end)
            Error = Output of equation (18.11) using θ_f and θ_p as inputs.
            if (Error < Previous Error) **then**
                $K_p = K_{p(it)}$(i);
                $K_i = K_{i(it)}$ (j);
                Previous Error = Error;
    return K_p & K_i

Week 3: 2nd - 8th February

- Find and remove bias on the sensors.
- Find the **R** matrix used by the Kalman filter.
- Tune the Kalman Filter and compare it to the Complementary Filter.

19.1 Sensor Bias

The bias on the sensors were found for the accelerometer by the means shown in figure 16.1 and they were found to be as follows:-

$$\begin{bmatrix} b_{a-x} \\ b_{a-y} \\ b_{a-z} \end{bmatrix} = \begin{bmatrix} -2.223 \\ 5.3447 \\ 9.17645 \end{bmatrix} \tag{19.1}$$

The bias on the gyroscope was found by placing the sensor on a flat surface and left to settle for five minutes so that the sensors would only measure there offsets and were found to be as follows:-

$$\begin{bmatrix} b_{g-x} \\ b_{g-y} \\ b_{g-z} \end{bmatrix} = \begin{bmatrix} -34.163 \\ 20.156 \\ -88.66 \end{bmatrix} \tag{19.2}$$

19.2 Measurement of the R matrix

ax It was possible to measure the **R** covariance matrix by obtaining various measurement from the sensors and estimating the noise present. The results were read to m-file and the covariance found using *cov(x)* function in *MatLab*. Hence the **R** matrix was calculated as follows:-

$$\mathbf{R}_a = \begin{bmatrix} \text{cov}(\phi) & 0 & 0 \\ 0 & \text{cov}(\theta) & 0 \\ 0 & 0 & \text{cov}(\psi) \end{bmatrix} \tag{19.3}$$

Where *cov(φ)* denotes the covariance of the ϕ angle from the expected value.

Hence $\mathbf{R}_a$ was found as follows:-

$$\mathbf{R}_a = 1.0e-03 \begin{bmatrix} 0.2710 & 0 & 0 \\ 0 & 0.1980 & 0 \\ 0 & 0 & NaN \end{bmatrix} \tag{19.4}$$

By a similar manner one can find the $\mathbf{R}_g$ and define it as follows:-

$$\mathbf{R}_g = 1.0e-05 \begin{bmatrix} 0.1264 & 0 & 0 \\ 0 & 0.2221 & 0 \\ 0 & 0 & 0.0757 \end{bmatrix} \tag{19.5}$$

19.3 Tuning of the Kalman Filter

Calculating the **Q** matrix proved to be a much more difficult task. Very little real information regarding the noise present in the states was known. However, one method of tuning the filter is to leave on of the matrix fixed and scale the other as it is the ratio of **Q** / **R** which is important as seen in [38]. Hence, it was found that the ratio of the parameters affected the response and to a less degree the individual parameter values. Therefore **R** was set to the theoretical value calculated and **Q** varied until an optimal response was achieved. The results of tuning the filter can be seen in figure Fig: Kalman Filter Estimate and Complementary Filter Estimate

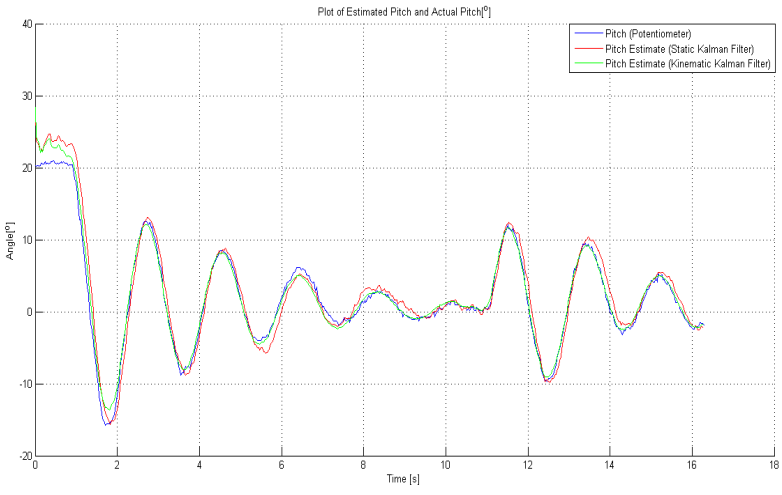
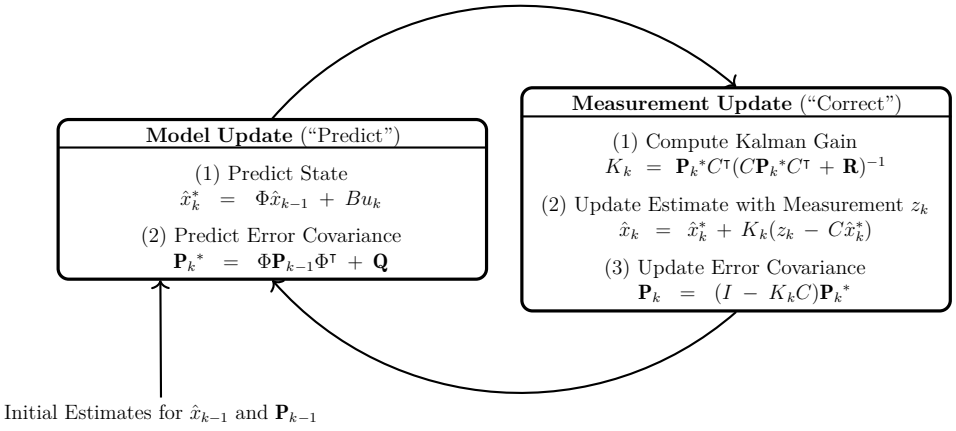


Figure 19.1: Plot of Kalman Filter Estimate, Complementary Filter Estimate and Actual Angle of the device

For the above graph $K_{p\theta}$ and $K_{i\theta}$ are equal to 3.5902 and 2.0522 respectively. While K was found to be as follows:-

$$K_\theta = \begin{bmatrix} 0.032064 & 0.0001017 \\ 0.0085871 & 0.996498 \end{bmatrix} \tag{19.6}$$

19.4 Chart Depicting how the Kalman Filter Works



Week 4: 9^{th} - 15^{th} February

- Tune the Kalman Filter that uses a pure kinematic system model (i.e is independent of a control input matrix)
- Read up on non-linear control
- Investigate the EKF as it is commonly used in GPS systems

20.1 Tuning of a Kalman Filter that uses a pure kinematic

This it was decided to look at and tune an adaptive kalman filter which does no depend on the controller plant inputs. This filter was modelled as follows and is commonly used on quad-rotors.

$$\dot{Rot} = \underbrace{\begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix}}_{S(\omega_g)} Rot \tag{20.1}$$

As the gravity is fixed in the NED frame, it must be converted to the B frame as the accelerometer and gyroscope are fixed to the quad-rotor. Since gravity acts in the z-direction, the third column of the rotation matrix Rot can be used to approximate ϕ and θ of the quad-rotor and hence (20.1)can be redefined as follows knowing that gravity is taken to act in the z-direction.

$$\dot{Rot}_3 = S(\omega_g) Rot_3 \tag{20.2}$$

The continuous state space equation of (20.3) therefore represents the system under control.

$$\begin{aligned} \underline{\dot{x}} &= S(\omega_g)\underline{x} \\ \underline{y} &= \underline{x} \end{aligned} \tag{20.3}$$

However, (20.3) is a continuous state space equation. This must be converted to

the discrete version since the Kalman Filter must be implemented in discrete form, which is achieved as follows:-

$$\begin{aligned} \underline{x}_k &= \Phi x_{k-1} \\ \underline{y}_k &= \underline{x}_k \end{aligned} \tag{20.4}$$

This filter was tuned in a similar manner to the Kalman filter seen last week and the two filters are compared in figure 20.1. Note the two filters give almost identical results (the one featured this week is slightly better), but the filter featured this week is more computational intensive (as Φ varies with time)and does not allow one to filter the gyroscope outputs. I light of these drawbacks the filters works well and would be better suited to systems who's model are hard to develop.

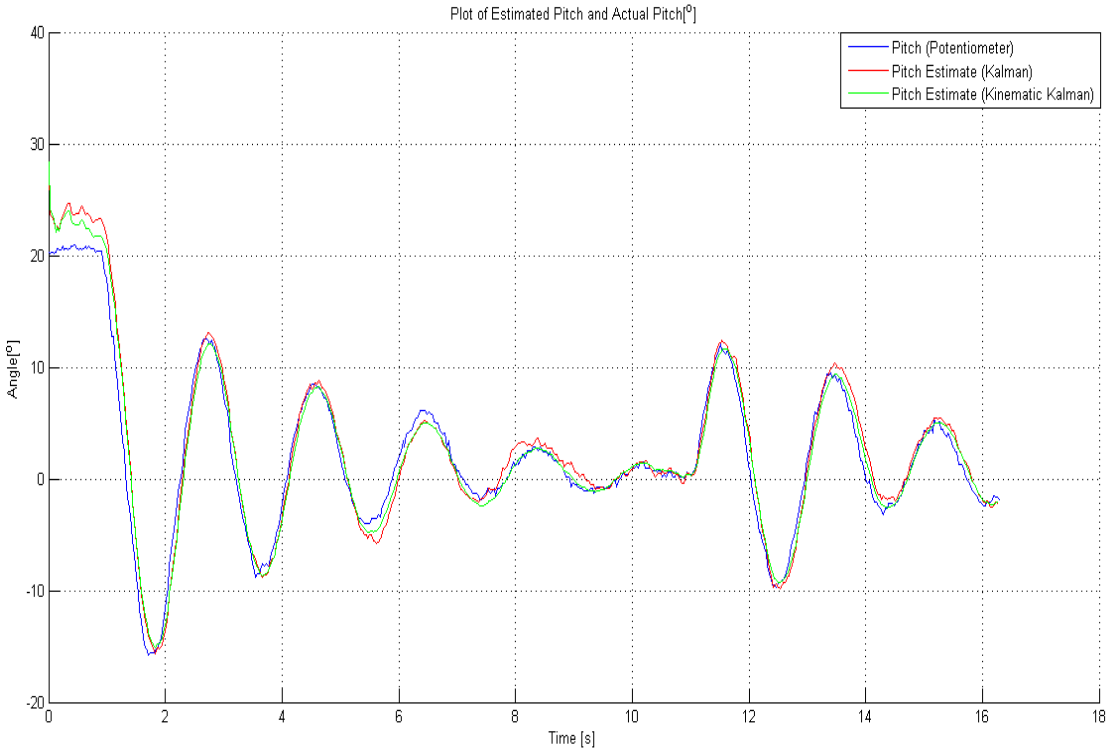


Figure 20.1: Plot of Kalman Filter Estimate, Kinematic Kalman Filter Estimate and Actual Angle of the device

20.2 Introduction to the EKF and associated code

As the measurement function $h(x_k, v_k)$ of the gyroscope is a non-linear function it was decided that it might be worth while looking at a more advanced kalman filter

which is capable of dealing with non-linear systems, thus the EKF was explore. Note as the group wanted to implement a GPS system on the quad-rotor the EKF was the natural choice as it is the de-facto filter used in GPS systems [36].

In something akin to a Taylor series, one most linearise the system around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the face of non-linear relationships. The process is governed by the *non-linear* stochastic difference equation:-

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{20.5}$$

and the measurement model by the following stochastic difference equation:-

$$z_k = h(x_k, v_k) \tag{20.6}$$

where the random variables w and v again represent the process and measurement noise.

In practice of course one does not know the individual values of the noise w and v at each time step. However, one can approximate the state and measurement without them as follows:-

$$x_k^* = f(\hat{x}_{k-1}, u_k, 0) \tag{20.7}$$

and the measurement model by the following stochastic difference equation:-

$$z_k = h(x_k^*, 0) \tag{20.8}$$

It is important to note that a fundamental flaw of the EKF is that the distributions of the various random variables are no longer normal after undergoing their respective non-linear transformations. The EKF is simply an ad-hoc state estimator that only approximates the optimality of Bayes' rule by linearisation.

(20.7) is linearised around the control input u_k and the previous estimate using

(20.9) and the measurement function (20.8) is linearised around the x_k^* using (20.10)

$$A_k = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{array}\right]\Bigg|_{\hat{x}_k, u_k} \tag{20.9}$$

$$C_k = \left[\begin{array}{cccc} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n} \end{array}\right]\Bigg|_{\hat{x}_k^*} \tag{20.10}$$

Note the above Jacobian can be calculated at each stage on a micro-controller by using the Cauchy's integral formula which is defined as follows [37]:-

$$f^{(n)}(z) = \frac{n!}{2\pi i} \oint_\gamma \frac{f(\kappa)}{(\kappa - z)^{n+1}} d\kappa \tag{20.11}$$

If one wants to implement (20.11) on a micro-controller one has to approximated it a closed form power series as follows:-

$$f^{(n)}(z) \approx \frac{n!}{mh} \sum_{j=0}^{m-1} \frac{f(z + he^{i\frac{2\pi j}{m}})}{e^{i\frac{2\pi jn}{m}}} \tag{20.12}$$

The derivation of a complex-step derivative (first partial derivative) approximation is done by an approximation of a non-linear function with a complex variable using the Taylor's series expansion.

$$f(x + ih) = f(x) + ihf'(x) - h^2\frac{f''(x)}{2!} - ih^3\frac{f'''(x)}{3!} + h^4\frac{f^4(x)}{4!} + \ldots \tag{20.13}$$

Now taking only the imaginary parts of both sides gives

$$\mathrm{Im}[f(x + ih)] = hf'(x) - h^3\frac{f'''(x)}{3!} + \ldots \tag{20.14}$$

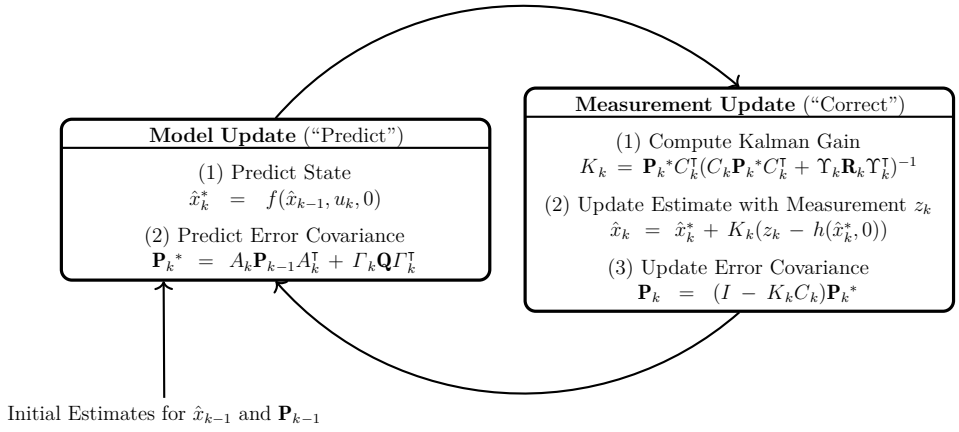
Dividing by h, rearranging and assuming terms higher than h^2 or higher can be ignored since the interval h can be chosen up to the precision of the machine (smallest difference

the machine can produce) and thus (20.14) can be approximated as follows:-

$$f^{'}(x) = \mathrm{Im}[f(x + ih)]/h \tag{20.15}$$

As (20.15) is not a function of differences it is more accurate than standard finite difference and more inportantly one can now calculate a partial derivative on a micro-controller using (20.15)

20.3 Chart Depicting how the EKF Works



Week 5: 16th - 22nd February

- Interface with GPS module
- Test the control and filter code on the quad-rotor
- Use the Kalman Filter to filter the GPS

21.1 Testing

Most of the week was sent testing the control and filters on the quad-rotor. It was found that the delay on the second order complementary filter was to large so the Kalman filter was implemented on the chip. The first order filter was also investigated to see if the delay could be reduced, it was and the results of the filtering can be seen in figure 21.1:-

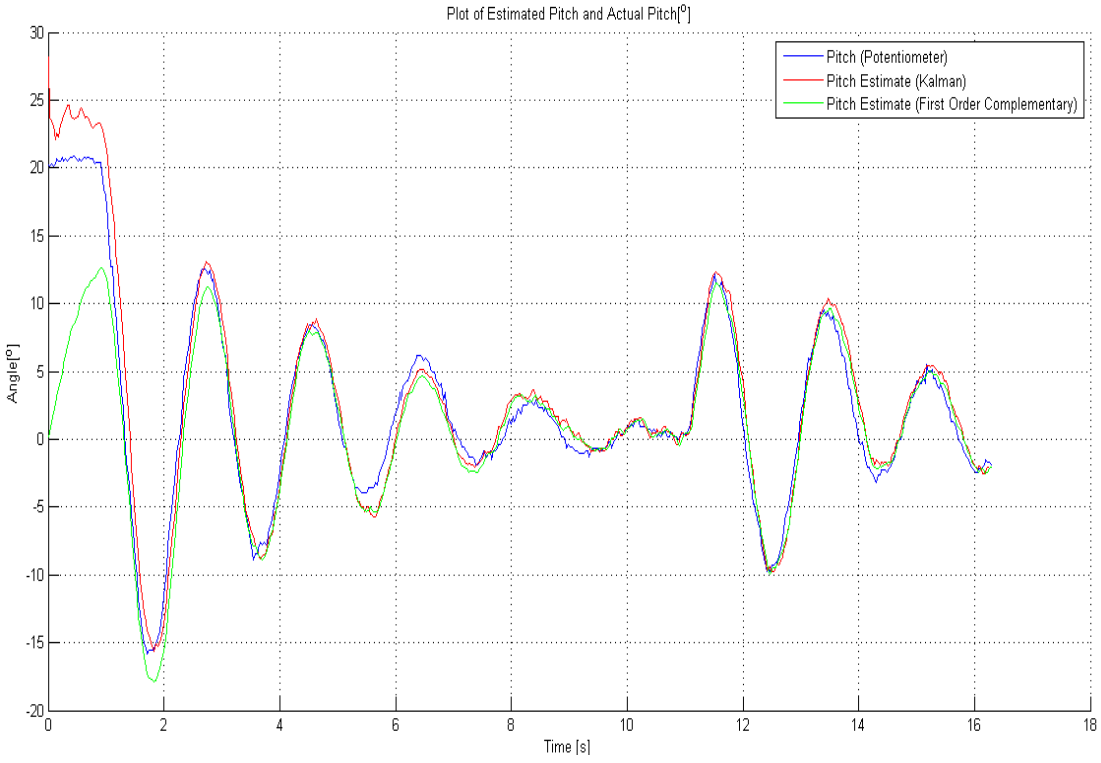


Figure 21.1: Plot of Kalman Filter Estimate, First Order complementary Filter and Actual Angle of the device

Note as the findings in figure 21.1 are promising the filter overshoots when there is high accelerations, the Kalman Filter doesn't have such draw backs. The complementary filter presented in figure 21.1 has a filter parameter of $\tau = 0.5$.

21.2 GPS Mapping & Associated Equations

If one wants to control the quad-rotor using differential equations based on x, y, z coordinate system one will have to map the latitude (φ) and longitude (λ) values given by the GPS sensor. Note that any mapping "unwrapping" of a spherical objecting onto a flat plane will cause distortions and loss of information (this was first mathematically proven by C. F. Gauss). Methods of doing this mapping will now be presented:-

21.2.1 Transverse Mercator Projection

The Transverse Mercator Projection or Gauss–Krüger projection is a conformal mapping of the earth ellipsoid where a central meridian is mapped into a straight line at constant scale. It is widely used in national and international mapping systems around the world and hits a middle ground between computational ease and accuracy. In order to preform this projection one places a model of the Earth on a cylinder as seen in figure 21.2 and "roll" the cylinder to form a conventional map [46].

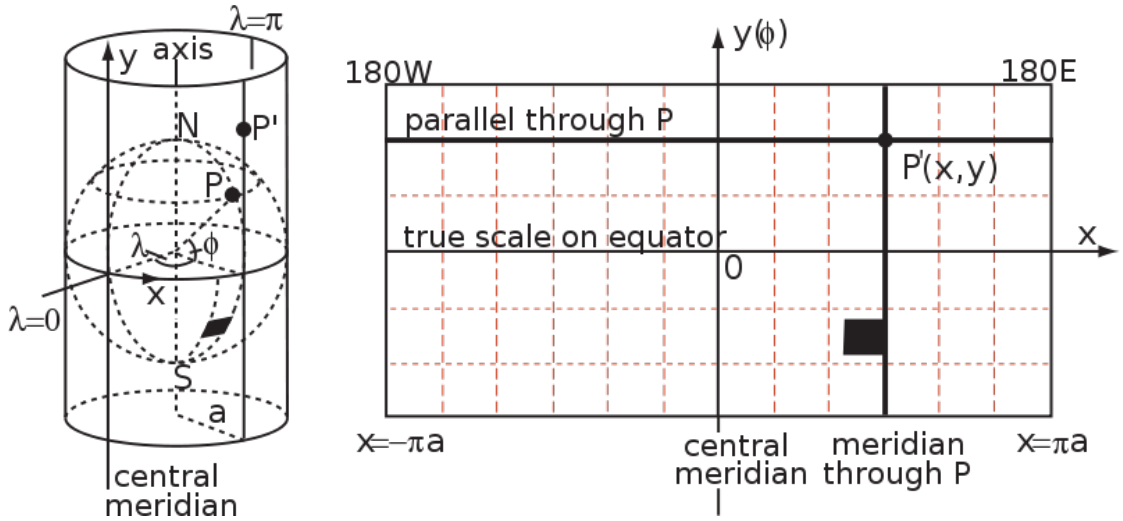


Figure 21.2: Normal aspect of a tangent cylindrical project of a sphere (Transverse Mercator projection)

Using the project defined by figure 21.2 one can calculate x_n & y_n coordinates using the following assuming the y-axis goes through the Prime Meridian in Greenwich.

$$\begin{aligned} x_n(\lambda, \varphi) &= \frac{1}{2} k_o a \ln\left[\frac{1 + \sin\lambda\cos\varphi}{1 - \sin\lambda\cos\varphi}\right] \\ y_n(\lambda, \varphi) &= k_o a \arctan\left[\sec\lambda\cos\varphi\right] \end{aligned} \tag{21.1}$$

Similarity, if one can to find latitude and longitude using only x_n & y_n one can use

the following equations:-

$$\begin{aligned} \lambda &= \arctan\left[\sinh\frac{x_n}{k_o a}\sec\frac{y_n}{k_o a}\right] \\ \varphi(x_n, y_n) &= k_o a \arctan\arcsin\left[\operatorname{sech}\frac{x_n}{k_o a}\sin\frac{y_n}{k_o a}\right] \end{aligned} \tag{21.2}$$

where:-

k_o; is the "point scale" and is commonly taken as 0.9996 for reasons that will not be mentioned here but are listed in [46]

a; is the equatorial radius of the earth and is equal to 6,378,137 m

b; is the polar radius of the earth and is equal to 6,356,752.3142 m

21.2.2 Universal Transverse Mercator

The Universal Transverse Mercator (UTM) projection uses a 2-D Cartesian based system to map the locations on the surface of the earth. Note this system differs from the traditional method of latitude (φ) and longitude (λ) as it is not a single map projection. UTM also divides the Earth into sixty "Zones" , which are six-degree bands of longitude (λ) and it uses the Transverse Mercator Projection in each of these "Zones" [46].

The following formulas are truncated version of the Transverse Mercator: flattening series, these formulae where first derived by J.H.L Krüger in 1912. The mapping use WGS 84 which describes the Earth as an oblate spheroid. In order to map from latitude and longitude (φ,λ) to UTM (x_n,y_n) one has to introduce constants to make the notation easier, this is done first:-

$$\begin{aligned} n &= \frac{f}{2-f}, \quad A = \frac{a}{1+n}\left(1+\frac{1}{4}n^2+\frac{1}{64}n^4+\cdots\right) \\ \alpha_1 &= \frac{1}{2}n-\frac{2}{3}n^2+\frac{5}{16n^3}+\cdots, \quad \alpha_2 = \frac{13}{48}n^2-\frac{3}{5}n^3+\cdots, \quad \alpha_3 = \frac{61}{240}n^3+\cdots \\ \beta_1 &= \frac{1}{2}n-\frac{2}{3}n^2+\frac{37}{96}n^3+\cdots, \quad \beta_2 = \frac{1}{48}+\frac{1}{48}n^3+\cdots, \quad \beta_3 = \frac{17}{480}n^3+\cdots \\ \delta_1 &= 2n-\frac{2}{3}n^2-2n^3+\cdots, \quad \delta_2 = \frac{7}{3}n^2-\frac{8}{5}n^3+\cdots, \quad \delta_3 = \frac{56}{15}n^3+\cdots \end{aligned} \tag{21.3}$$

Where:

f: is flatting and is given by the following:- $f = (a-b)/a$

e: is the eccentricity of the earth and is defined as follows:- $e = \sqrt{f(2-f)}$

Mapping from latitude and longitude (φ,λ) to UTM (x_n,y_n)

Before one can project (φ,λ) to UTM (x_n,y_n) one has to do an interment step[1].

$$\tau' = \sinh\left(\tanh^{-1}(\sin\varphi) - \frac{2\sqrt{n}}{1+n}\tanh^{-1}\left(\frac{2\sqrt{n}}{1+n}\sin\varphi\right)\right) \tag{21.4}$$

$$\xi' = \tan^{-1}(\frac{\tau'}{\cos(\lambda-\lambda_o)}) \tag{21.5}$$

$$\eta' = \sinh^{-1}\left(\frac{\sin(\lambda-\lambda_o)}{\sqrt{\tau'^2+\cos^2\lambda}}\right) \tag{21.6}$$

Now that these are defined one can map (φ,λ) to UTM (x_n,y_n) using the following formulas:-

$$x_n = k_o A\left(\eta' + \sum_{j=1}^{3}\alpha_j\cos(2j\xi')\sinh(2j\eta')\right) \tag{21.7}$$

$$y_n = k_o A\left(\xi' + \sum_{j=1}^{3}\alpha_j\sin(2j\xi')\cosh(2j\eta')\right) \tag{21.8}$$

21.2.3 Basic Method of Mapping (φ,λ) to (x_n,y_n)

For small angles (1^o) one can use the following equations:-

$$\Delta_\varphi = \frac{\pi a(1-e^2)}{180(1-e^2\sin^2\varphi)^{\frac{3}{2}}} \tag{21.9}$$

The distance in meters (correct to 0.01 metres) between ($\varphi \pm 0.5^o$) on the World Geodetic System 1984 (WGS-84) 84 spheroid is given by the following formula:-

$$\Delta_\varphi = 111132.954 - 559.822\cos(2\varphi) + 1.175\cos(4\varphi) \tag{21.10}$$

[1]Please see the following for more details and a full derivation of these formulas [46]

Similarly for longitude:-

$$\Delta_\lambda = \frac{\pi a \cos(\varphi)}{180\sqrt{1 - e^2 \sin^2 \varphi}} \tag{21.11}$$

Week 6: 23rd February- 1st March

22.1 Objectives

- Chose Mapping for GPS to find xyz
- Test control on the Quad-rotor
- Prepare for the seminars

22.2 Test Quad-rotor

Fixing issues with program run-time on Propeller took longer than planned so no progress was made on actual testing.

22.3 GPS conversion from ECEF frame to NED frame

In other for the Quad-rotor to be stabilized in position another frame was introduced, but before the distance to move in the NED frame can be found the distance to travel in the ECEF has to be found. This difference can be found if one knows the latitude φ and longitude λ on is at and the latitude φ and longitude λ the Quad-rotor has to go too. Once this is know, (22.1) can be used to find xyz displacement in the ECEF and then map this displacement using (22.2).

$$\begin{aligned} N &= \frac{a}{\sqrt{1 - e^2 \sin^2 \varphi}} \\ x &= (N + h)\cos(\varphi)\cos(\lambda) \\ y &= (N + h)\cos(\varphi)\sin(\lambda) \\ z &= [N(1 - e^2) + h]\sin(\varphi) \end{aligned} \tag{22.1}$$

Where:-

h: is the height of the quad-rotor from the surface of the planet.

In order to find out how far the quad-rotor has to go in order to reach the required position [1] (22.1) has to be mapped using the following:-

$$\begin{bmatrix} y_n \\ x_n \\ z_n \end{bmatrix} = \begin{bmatrix} -S_{\varphi o}C_{\lambda o} & -S_{\varphi o}S_{\lambda o} & C_{\varphi o} \\ -S_{\lambda o} & C_{\lambda o} & 0 \\ -C_{\varphi o}C_{\lambda o} & -C_{\varphi o}S_{\lambda o} & -S_{\varphi o} \end{bmatrix} \begin{bmatrix} x_p - x_o \\ y_p - y_o \\ z_p - z_o \end{bmatrix} \tag{22.2}$$

where:- p subscript is the new point at which the quad-rotor is has to move to and p subscript is the current location of the quad-rotor. Simialry one can find the velocity of the quad-rotor using the heading and speed measurements that the GPS module using the following:-

$$\begin{aligned} \text{Speed} &= \sqrt{\dot{x}^2 + \dot{y}^2} \\ \text{Heading} &= \arctan\left(\frac{\dot{x}}{\dot{y}}\right) \end{aligned} \tag{22.3}$$

Thus the rate of change of position can also transformed by means of the following equations.

$$\begin{bmatrix} \dot{y_n} \\ \dot{x_n} \\ \dot{z_n} \end{bmatrix} = \begin{bmatrix} -S_{\varphi}C_{\lambda} & -S_{\varphi}S_{\lambda} & C_{\varphi} \\ -S_{\lambda} & C_{\lambda} & 0 \\ -C_{\varphi}C_{\lambda} & -C_{\varphi}S_{\lambda} & -S_{\varphi} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{22.4}$$

where

$$\begin{aligned} \dot{x} &= \sqrt{\frac{(\text{speed}^2)}{1 + \tan(\text{heading})^2}} \\ \dot{y} &= \sqrt{(\text{speed})^2 - \dot{x}^2} \end{aligned} \tag{22.5}$$

22.4 Seminar Presentation

The seminar was on Friday 27th February. Filter graphs recorded for use in presentation.

[1] And hence also control the quad-rotor

Week 7: 2nd - 8th March

- Acquire data for GPS filter design
- Find the **R** for the GPS Kalman Filter.
- Test the Control on the Quad-rotor

23.1 GPS data

The following data was acquired during the course of the

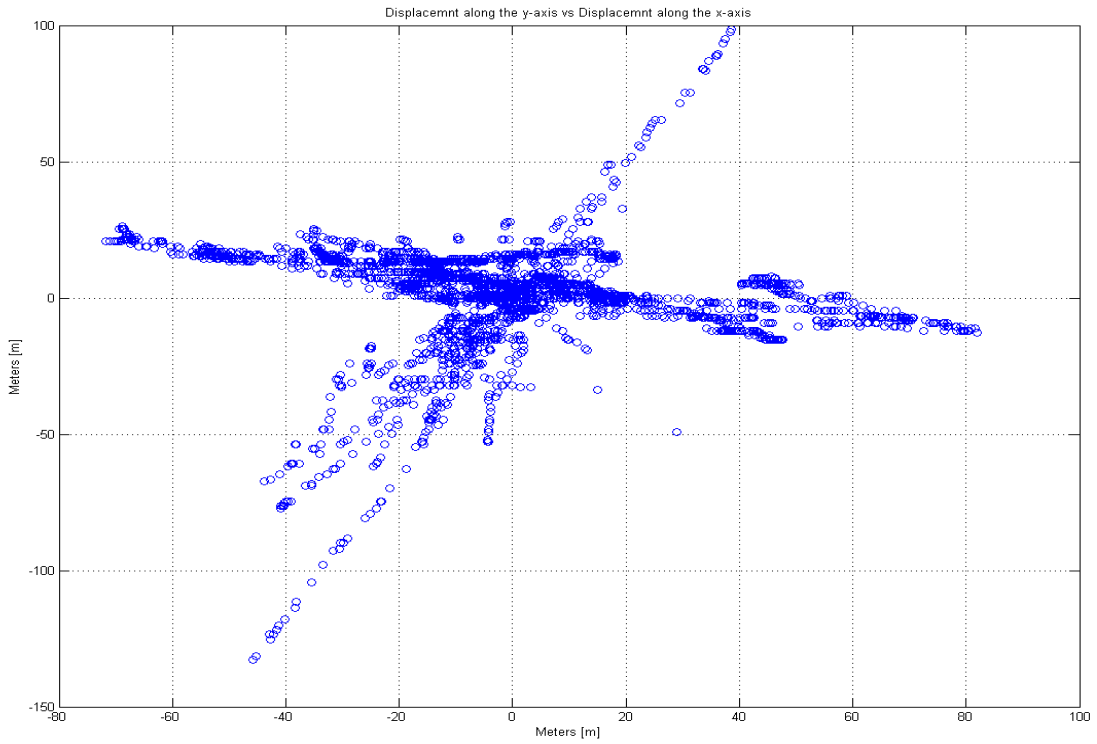


Figure 23.1: Plot of GPS data in x-y plane

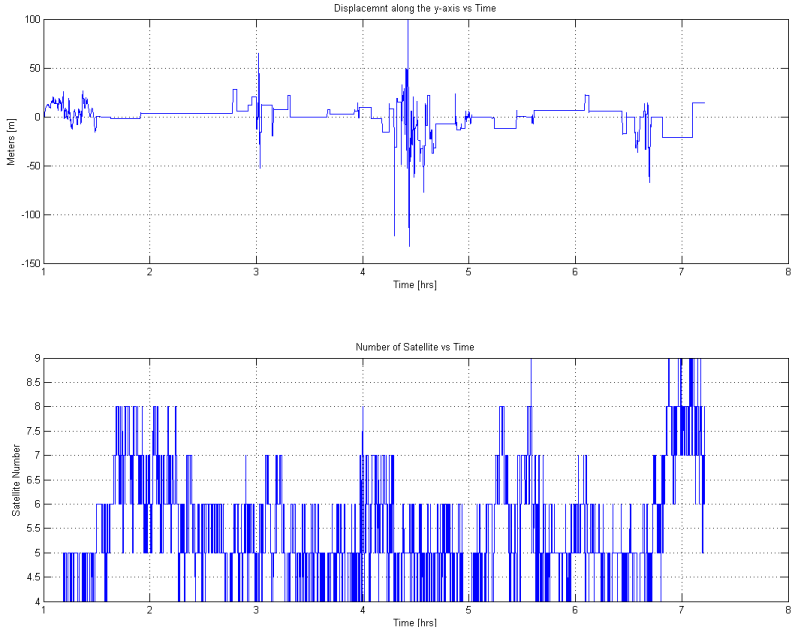


Figure 23.2: Plot of GPS data in displacement in x direction vs time

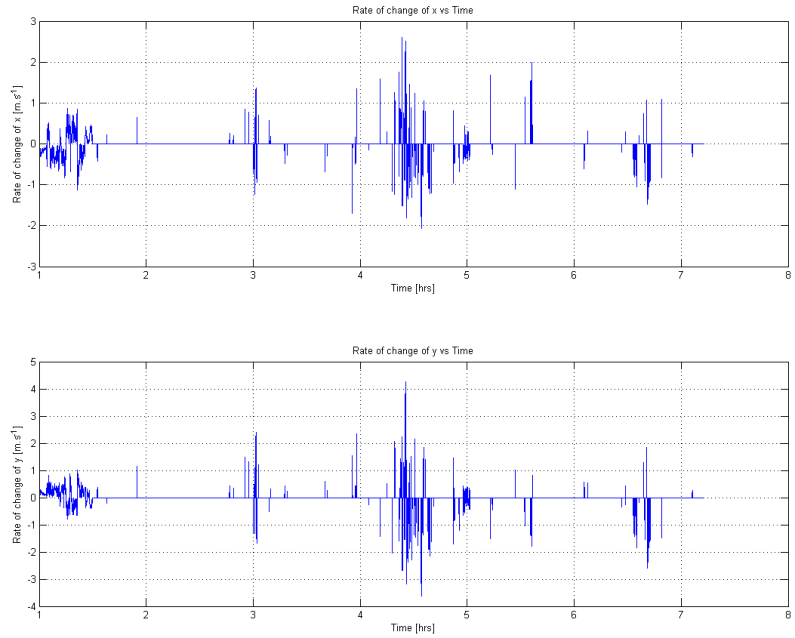


Figure 23.3: Plot of GPS data in x rate vs time

23.2 Find the R for the GPS Kalman Filter.

From data collected the following **R** where found for x and y respectively.

$$\mathbf{R}_{x_n} = \begin{bmatrix} \mathrm{cov(x_n)} & 0 \\ 0 & cov(\dot{x}_n) \end{bmatrix} = \begin{bmatrix} 132.58 & 0 \\ 0 & 0.0301 \end{bmatrix} \tag{23.1}$$

$$\mathbf{R}_{y_n} = \begin{bmatrix} \mathrm{cov(y_n)} & 0 \\ 0 & cov(\dot{y}_n) \end{bmatrix} = \begin{bmatrix} 159.4762 & 0 \\ 0 & 0.0617 \end{bmatrix} \tag{23.2}$$

23.3 GPS model used for Kalman Filter

The x model used by the kalman filter is defined as follows:-

$$\begin{bmatrix} \dot{x_n} \\ \ddot{x_n} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\beta_x}{m} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} U \tag{23.3}$$

and the y modelled as follows.

$$\begin{bmatrix} \dot{y_n} \\ \ddot{y_n} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\beta_y}{m} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} U \tag{23.4}$$

23.4 Test Quad-rotor

Fixing issues with program run-time on Propeller and the quad-rotor was stabilised about the pitch and roll axis.

Complementary Filter

A pair of filters are called complementary filters if their transfer functions sum to one at all frequencies in a complex sense, i.e. the phase is zero and the magnitude is one.

A complementary filter is easily derived by solving the transfer function of the Mahony & Madgwick filter for the angle θ, which yields

$$\theta = \frac{1 + \frac{K_p}{K_i}s}{1 + \frac{K_p}{K_i}s + \frac{1}{K_i}s^2}a + \frac{\frac{1}{K_i}S^2}{1 + \frac{K_p}{K_i}s + \frac{1}{K_i}s^2}\frac{1}{s}\omega \tag{24.1}$$

Obviously, and not unexpectedly, this complementary filter is build from 2^{nd} order filters. Note that the filter acting on the acceleration data actually consists of a low-pass plus band-pass filter.

This result has interesting consequences. Being 2^{nd} order filters, the frequency response of the acceleration and rotation rate filters are characterized by the resonance frequency and damping factor

$$\omega_0 = \sqrt{K_i} \qquad \xi = \frac{K_p}{2\sqrt{K_i}} \tag{24.2}$$

The damping factor determines the overshoot at the resonance frequency. For high-pass (and low-pass) filters the frequency response is flat (and the step response non-oscillatory) for $\xi \geq 1$. This suggests the criterion [1].

$$K_i \leq \frac{1}{4}K_p^2 \tag{24.3}$$

[1] See http://www.olliw.eu/2013/imu-data-fusing/ for more details