Helicopter lab assignment

Group 01
Erling Hjermstad - 528112
Jørgen Grimstad Wisløff - 494599
Thomas Fjorden - 510108

November 24, 2021

Department of Engineering Cybernetics

Abstract

This report describes the design and implementation of different control systems for a helicopter model. The stability of the different approaches are explored. Building from a simple PD and PID controller to more advanced methods, including a linear-quadratic regulator (LQR), Kalman filter for noise reduction and state estimation. Through experimentation on the system with increasingly less well-posed prerequisites, we find that increasing error modes warrant more complex control designs. However, on well-posed systems the advantages a complex control design offer might not be worth the potentially small difference in result compared to a less complex design. This indicates the strength control systems wield, but also the importance of the design of the system. A well-posed system can more easily be controlled, while a good controller can mitigate errors due to assumptions and design flaws.

Contents

1 Introduction **1**

2 Part I - Monovariable control **2**
2.1 Modelling 2
2.2 PD-controller 3
2.3 Discussion 4

3 Part II – Multivariable control **6**
3.1 LQR implementation 6
3.2 Integral LQR 7

4 Part III – Luenberger observer **10**
4.1 Extended state-space formulation and observability 10
4.2 IMU-characteristics and transformations 11
4.3 State estimator 14
4.4 Discussion 15

5 Part IV – Kalman filter **16**
5.1 Discretization 16
5.2 Noise estimate and experimentation 17
5.3 Bonustask: Van Loan's method 19
5.4 Bonustask: IMU-bias estimation 21

6 Conclusion **23**

Appendix **24**

A Constants **24**

B MATLAB Code **24**
B.1 Kalman_correction.m 24
B.2 Kalman_prediction.m 25
B.3 accelTransform.m 25
B.4 euler_rates.m 25

C Simulink Diagrams **26**
C.1 Monovariable control 26
C.2 Multivariable control 26
C.3 Luenberger observer 27
C.4 Kalman filter 28

References **30**

1 Introduction

The helicopter project concerns itself with the control of a helicopter model. The model consists of a drone with two propellers connected to a base with a lever, thus restricting its movements to a ball around the base. Through the project useful experience in the applications of control theory is gained. Putting theory into practice solidifies knowledge, making it more tangible. The project is also good training in approaching a problem systematically, and documenting the progress for reproducible results.

The report is organized chronologically. Initially relevant derivations and theory is discussed, before the experimentation is viewed. Finally each part conclude with a discussion where hypotheses are reflected upon with background in the experimentation. Relevant plots are included.

In appendix A all physical constants and values can be found. The derived constants together with their derivation is also listed. Appendix B contains the MATLAB codes and appendix C the Simulink diagrams.

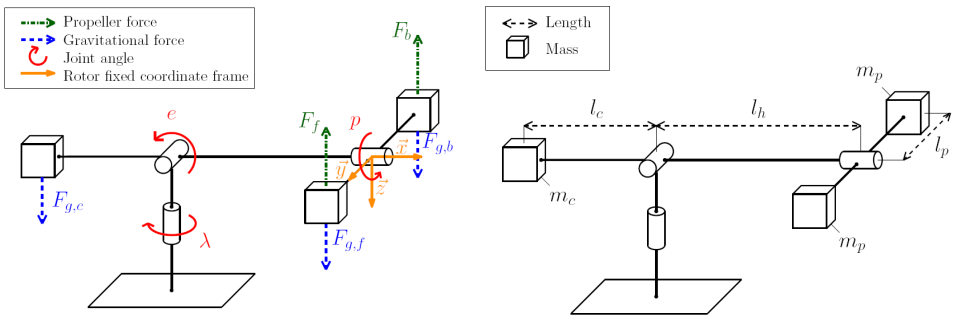


Figure 1: Model of the helicopter setup, with definitions and constant names.

2 Part I - Monovariable control

The first part of the assignment was about monovariable control. Pitch and elevation was to be controlled individually. For the elevation we were given a tuned PID-controller. We were tasked to implement and tune a PD-controller for pitch control.

2.1 Modelling

The first step was to develop a mathematical model of the helicopter. Assuming linearity in the propeller-motors, the motorforces can be expressed as proportional to the applied voltages. The voltages are our control signal u.

$$u = \begin{bmatrix} V_S \\ V_D \end{bmatrix} = \begin{bmatrix} V_f + V_b \\ V_f - V_b \end{bmatrix} \tag{1}$$

Inspection of Figure 1 and Newtons second law for rotation gives the following equation set

$$J_p \ddot{p} = L_1 V_D \tag{2a}$$
$$J_e \ddot{e} = L_2 \cos e + L_3 V_S \cos p \tag{2b}$$
$$J_\lambda \ddot{\lambda} = L_4 V_S \cos e \sin p \tag{2c}$$

The constant values and definitions can be found in the tables of appendix A.

Linearization by eq. (3) of the equation set eq. (2) around the linearization point $\mathbf{x}^* = [p^*, e^*, \lambda^*, V_S^*, V_D^*]^\top = [0, 0, 0, V_{S,0}, 0]^\top$ yields eq. (4)

$$\ddot{\mathbf{x}} = F(\mathbf{x}) \approx F(\mathbf{x}^*) + \nabla F \mid_{\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) \tag{3}$$

$$\ddot{p} = K_1 V_D \tag{4a}$$
$$\ddot{e} = K_2 \tilde{V}_S \tag{4b}$$
$$\ddot{\lambda} = K_3 p \tag{4c}$$

where $\tilde{V}_S = V_S - V_{S,0}$. $V_{S,0}$ is the voltage required to lift the helicopter horizontal measured experimentally. Written in matrix form eq. (4) reads as

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \\ \dot{\lambda} \\ \ddot{\lambda} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ K_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_S \\ V_D \end{bmatrix} \tag{5}$$

2.2 PD-controller

Building on eq. (4a) our PD-controller can be designed. Generally a PD-controller has the form $u = K_p(x_{ref} - x) - K_d\dot{x}$. With $u = V_D$, $x = p$ and $x_{ref} = p_c$ inserting this into eq. (4a) leaves

$$\ddot{p} + K_1 K_{pd}\dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \tag{6}$$

In this task we were concerned with tuning the PD controller for the pitch. By computing the transfer function assuming zero initial conditions we get

$$G(s) = \frac{p}{p_c}(s) = \frac{K_1 K_{pp}}{s^2 + K_1 K_{pd} s + K_1 K_{pp}} \tag{7}$$

The quadratic formula yields the poles of the system.

$$\lambda_{1,2} = \frac{-K_1 K_{pd} \pm \sqrt{(K_1 K_{pd})^2 - 4K_1 K_{pp}}}{2}$$

Summing the poles removes dependence of K_{pp} leaving K_{pd} easily found.

$$K_{pd} = -\frac{\lambda_1 + \lambda_2}{K_1} \tag{8}$$

Subtracting the poles and inserting K_{pd} leaves

$$K_{pp} = \frac{\lambda_1 \lambda_2}{K_1} \tag{9}$$

A critically dampened system of degree two has coinciding poles. In our case this can easily be found to correspond to the condition

$$K_{pd} = 2\sqrt{\frac{K_{pp}}{K_1}} \tag{10}$$

Alternatively the transfer function can be presented as

$$G(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

[1] where $\zeta = \frac{K_{pd}}{2}\sqrt{\frac{K_1}{K_{pd}}}$ and $\omega_0^2 = K_1 K_{pp}$. Here ζ is called the relative dampening factor. ω_0 is the undampened resonance frequency. ζ is of special interest as it determines the dampening of our system. It can be seen that $\zeta = 1$ is equivalent to our criteria for a critically dampened system. A value above 1 is an overdamped system, while under 1 is an underdamped system.

2.3 Discussion

We hypothesized that keeping our poles real, negative and coinciding would give us the best response. Increasingly negative poles would give a faster response. Imaginary conjugate poles would lead to oscillations, while real distinct poles would lead to a slower response.

Interestingly we found that $\zeta = 0$ lead to an unstable response, not the theoretical standing oscillations. This indicates the inevitable inaccuracies in our model. For $\zeta = 0.01$ the standing oscillations appeared, however for large K_{pp} this also became unstable. Increasing ζ gave a less oscillating response, until $\zeta = 1$, where the response no longer oscillated. Increasing ζ even further slowed the response down. Negative values for K_{pd} gave an unstable response, as expected since this means we have poles in the right half of the s-plane. Figure 2 shows a collection of the unstable responses discussed above. Figure 3 shows a collection of stable responses, including a marginally stable response.

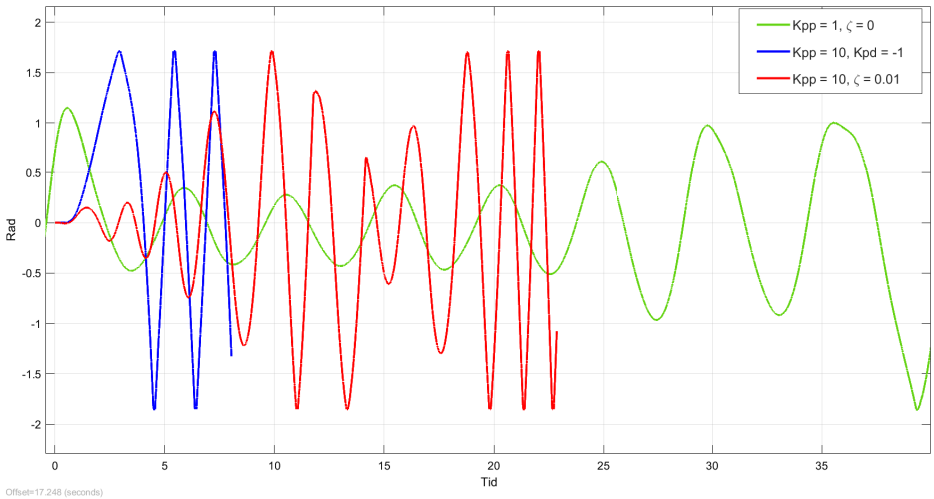


Figure 2: Collection of unstable responses

Since the poles determine the dynamics of the system the equations eq. (9) and eq. (8) can be used to manipulate the pitch response. Theoretically we can make the response as quick as we like, but in practice this is not possible. In addition to the energy demands for an infinitely quick response, a too aggressive tuning could make the system unstable as our model is not exact. However, the motors will saturate at a certain voltage, efficiently limiting our response speed possibilities.

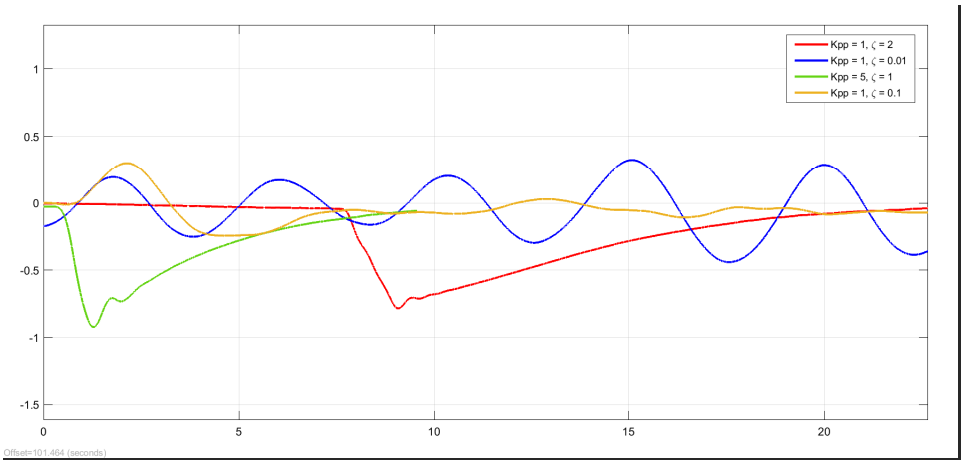


Figure 3: Collection of stable and marginally stable responses to varying excitations

3 Part II – Multivariable control

3.1 LQR implementation

To derive the state-space formulation, the state vector $\mathbf{x} = \begin{bmatrix} p & \dot{p} & \dot{e} \end{bmatrix}^T$ and input vector $\mathbf{u} = \begin{bmatrix} \tilde{V_s} & V_d \end{bmatrix}^T$ is needed. Extracting the corresponding rows and columns from eq. (5) we find

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \ddot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V_s} \\ V_d \end{bmatrix} \tag{11}$$

Where the $\mathbf{A}$ and $\mathbf{B}$ matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix}. \tag{12}$$

To determine if the system (11) is controllable, we need to examine if the controllability matrix has a full rank.

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

If we then take the rank of (13) matrix, $rank(\mathcal{C}) = 3 = n$, we can clearly see that the controllability matrix $\mathcal{C}$, has full rank and therefore the system (11) is controllable.
To track the reference $\mathbf{r} = \begin{bmatrix} p_c & \dot{e}_c \end{bmatrix}^T$, an implementation for a state-feedback controller with reference-feed-forward is needed, following the form

$$\mathbf{u} = \mathbf{Fr} - \mathbf{Kx} \tag{14}$$

Substituting (14) in (11) gives us the equation

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{BFr} \tag{15}$$

By assuming $\lim_{t\to\infty} p = p_c$ and $\lim_{t\to\infty} \dot{e} = \dot{e}_c$, we can assume that $\dot{\mathbf{x}}_\infty = \mathbf{0}$ and $\mathbf{y}_\infty = \mathbf{r}$. By solving (15) for $\mathbf{x}_\infty$ and inserting that into $\mathbf{y}_\infty = \mathbf{Cx}_\infty$, can $\mathbf{F}$ be found.

$$\mathbf{y}_\infty = \mathbf{Cx}_\infty = \mathbf{C}(\mathbf{BK} - \mathbf{A})^{-\mathbf{1}}\mathbf{BFr} = \mathbf{r} \tag{16}$$

By postmultiplying with the inverse of $\mathbf{r}$ and premultiplying with $\begin{bmatrix} \mathbf{C}(\mathbf{BK} - \mathbf{A})^{-\mathbf{1}}\mathbf{B} \end{bmatrix}$, we get that

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}(\mathbf{BK} - \mathbf{A})^{-1}\mathbf{B} \end{bmatrix}^{-1} \tag{17}$$

To complete the state-feedback controller with reference-feed-forward, we still need to find $\mathbf{K}$. The feedback gain matrix $\mathbf{K}$ can be found, using the LQR-function in MATLAB, $\mathbf{K} = lqr(\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R})$. For this to work, we need to assume some values for the cost matrices $\mathbf{Q}$ and $\mathbf{R}$. Where $\mathbf{Q}$ is the cost matrix for the different states, and $\mathbf{R}$ is the cost matrix for $\mathbf{u}$. To change the helicopters behaviour, $\mathbf{Q}$ and $\mathbf{R}$ can be adjusted. For the sake of simplicity, only the diagonal of the matrices were set to a different value than 0. To start with both $\mathbf{Q}$ and $\mathbf{R}$ were set to the identity-matrix. Resulting in a system that had an oscillating elevation, but otherwise stable. From there we knew that adjusting the pitch, caused the elevation to drop, meaning that we wanted to penalize elevation rate. We also wanted a stable helicopter, meaning that pitch would also be penalized, but power consumption and pitch rate would not be penalized as hard. After some trial and error, we came to a reasonable functioning helicopter, despite having loss of height when pitching. This resulted in the values

$$\mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{18}$$

Despite having a reasonable function helicopter with the values in eq. (18), we were not completely happy with the results. We decided therefore to try Bryson's rule. The maximum acceptable pitch was set to $\pi/8$, pitch rate to 0.5, elevation rate to 0.5, V_s to 5 and V_d to 5. Resulting in

$$\mathbf{Q} = \begin{bmatrix} 6.485 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 0.04 & 0 \\ 0 & 0.04 \end{bmatrix}. \tag{19}$$

These values gave us a relatively fast reacting helicopter, but the elevation dropped some, when pitching to fast. Trying more aggressive tuning resulted in either vibration of the motors or the controller adjusting to smaller changes, making the helicopter marginally stable.

3.2 Integral LQR

To achieve a better controllability of the system, integral effect for elevation rate and pitch angle was added

$$\dot{\gamma} = p - p_c \tag{20}$$

$$\dot{\zeta} = \dot{e} - \dot{e_c} \tag{21}$$

Resulting in a new state vector

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \\ \gamma \\ \zeta \end{bmatrix} \tag{22}$$

The helicopter system with integral action can be rewritten as

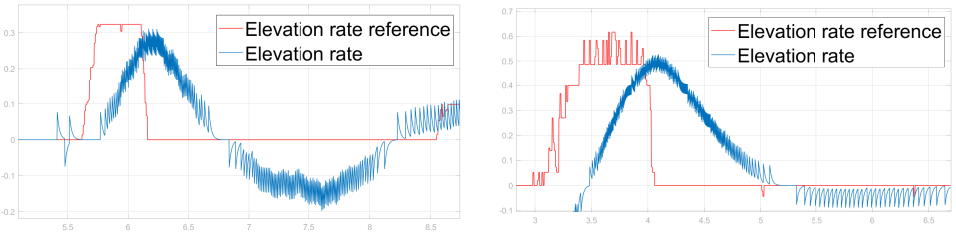
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{r} \tag{23}$$

Resulting in $\mathbf{A}, \mathbf{B}$ and $\mathbf{G}$ matrices

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{24}$$

After implementing this in to Simulink (fig. 21), an expansion of $\mathbf{Q}$ were needed. After some experimentation, we decided to keep the previous values for pitch, pitch rate and elevation rate in $\mathbf{Q}$ eq. (19). After experimentally testing different values, we discovered that penalizing γ heavily, would cause the helicopter to become unstable, reacting to every small deviation on the pitch. We still decided that penalizing γ somewhat heavy, would give the best result. For ζ it was mostly the same, but we decided to go for a lower penalizing value. Resulting in

$$\mathbf{Q} = \begin{bmatrix} 6.485 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{25}$$



(a) Without integral effect (b) With integral effect

Figure 4: Elevation rate response, with and without integral effect

The figure above shows the response for the helicopters elevation rate. Without the integral effect, we can see that the elevation rate never reaches the same value as the reference when stable, no matter how much time goes by. The elevation rate also shoots over or under when setting a new reference. This affects the flying of the helicopter, because we can't reach $\dot{e} = 0$ when $\dot{e}_c = 0$. The result of this is that the helicopter will slowly fall over time. With integral effect on the other hand were able to reach the reference, due to the integral effect. Meaning the helicopter would keep constant elevation over time. It also were also able not to over- or undershoot the reference when going from on to another. Regarding how the integral effect affects our choice of F, we can say that; F is redundant, although more efficient, since the integral effect now will steer towards reference, asymptotically. Since the value of the new states in eq. (22) is supposed to be zero at reference, F can be zero too.

4 Part III – Luenberger observer

To estimate the state values of this observable LTI system, based on the system matrices and the measured outputs, we will be using the Luenberger observer. An observer like this is particularly useful since we have some noisy measurements from the IMU. With the correct tuning a observer can provide better estimates of the state values than what we get from measurements directly.

4.1 Extended state-space formulation and observability

In this part we first had to find the matrices $\mathbf{A}$ and $\mathbf{B}$. Doing so required us to extract the corresponding rows and columns from eq. (5) related to the following state vector and input vector:

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix} \tag{26}$$

This gave us the following $\mathbf{A}$ and $\mathbf{B}$:

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \\ \ddot{\lambda} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix} \tag{27}$$

The full system can be described as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{28a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{28b}$$

In this project we have the privilege of having measurements on all of our states. Therefore the $\mathbf{C}$-matrix is equal to identity. However, to simulate a less well-posed system we can let the $\mathbf{C}$-matrix take other forms.

We can compute the observability matrix of our system using

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{29}$$

If the matrix has equal rank as our number of states the system is observable. After some mathematical experimentation on different **C**-matrices, we found the minimal observable **C**-matrix.

$$\mathbf{C}_{min} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

Inserting our **A** from eq. (27) and $\mathbf{C}_{min}$ from eq. (30) into eq. (29) we get the observability matrix

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ K_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & K_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{31}$$

We can see from eq. (31) that $\mathcal{O}$ is of rank 5, and therefore verifying that the system is observable. We found that the minimal states we needed was 2, e and $\dot{\lambda}$. This was also verified later by testing it with only using e and $\dot{\lambda}$ as states in the estimator. From figure fig. 5 you can see that it was harder to control, as is expected when estimating more states. As seen in figure fig. 6 the output V_S and especially V_D varies rapidly when only measuring two states.
The reason why we need to measure exactly these states goes back to eq. (4c). We can see that we can find p by differentiating $\dot{\lambda}$ once, which is fine. If we on the other hand would like to find λ we would have to integrate, which would lead to unknown integration constants.

4.2 IMU-characteristics and transformations

First of all we compared the new gyroscope outputs to the encoder-rates to ensure the IMU worked as expected as seen in fig. 7 and fig. 8.
To use the accelerometer measurements to indirectly measure the orientation of the helicopter, we use the following two equations:

$$p = arctan\left(\frac{a_y}{a_z}\right) \tag{32a}$$

$$e = arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \tag{32b}$$

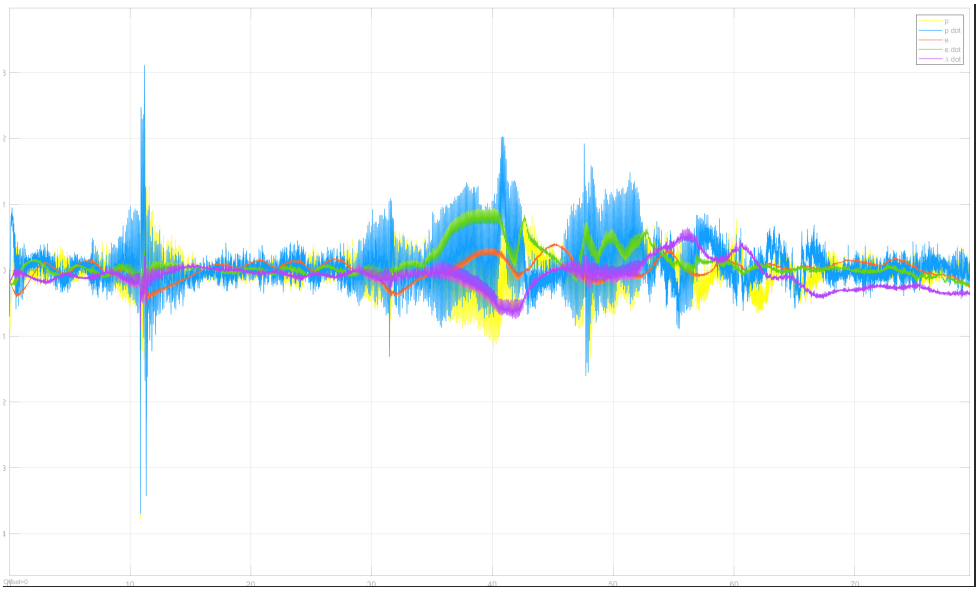


Figure 5: $\hat{x}$ using only two states, e and $\dot{\lambda}$

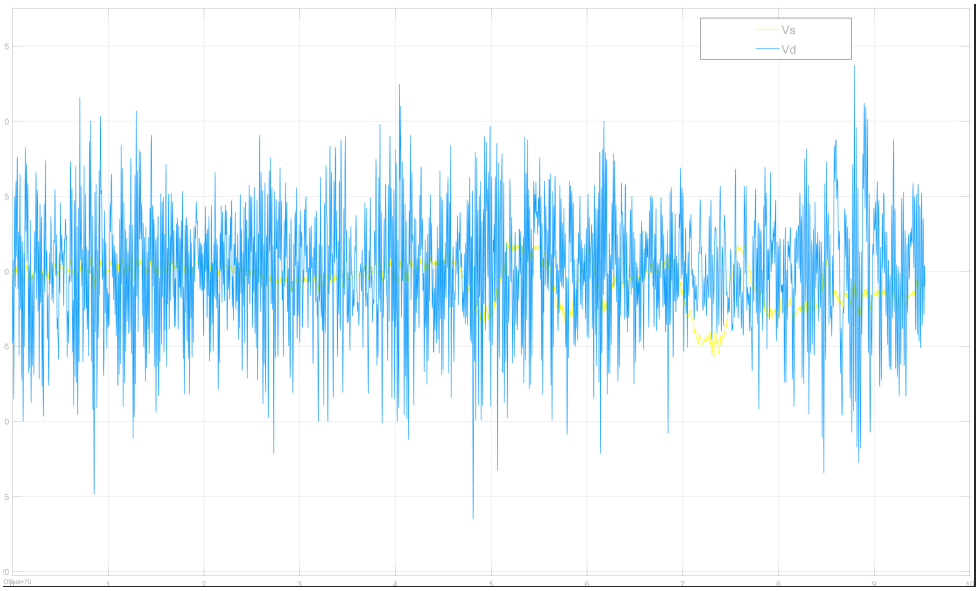


Figure 6: u using only two states, e and $\dot{\lambda}$

Since we want $\mathbf{y}$ to contain all the measurements from the IMU, which is

$$\mathbf{y} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} \tag{33}$$

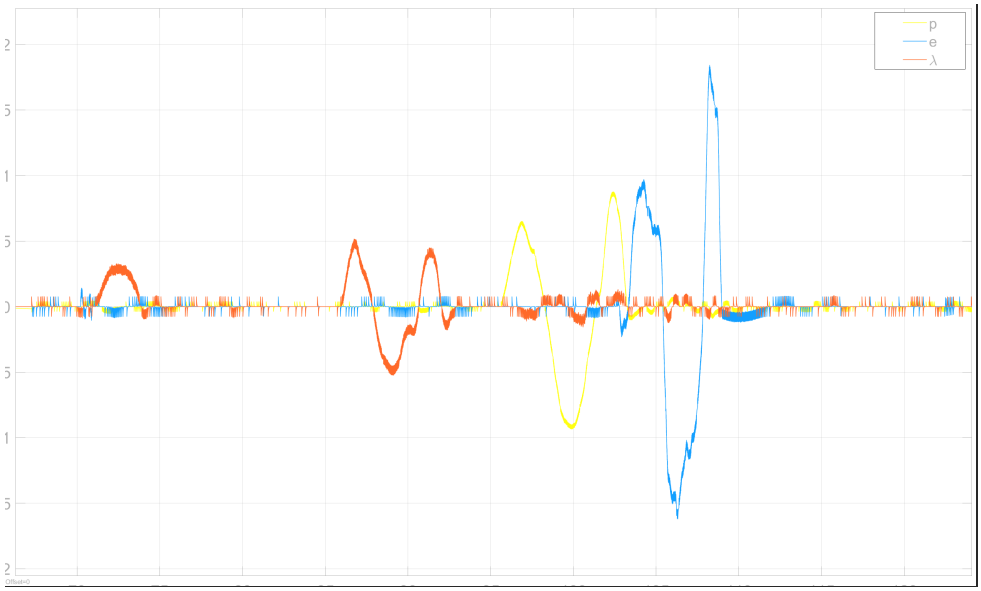


Figure 7: p, e and λ out from the encoder

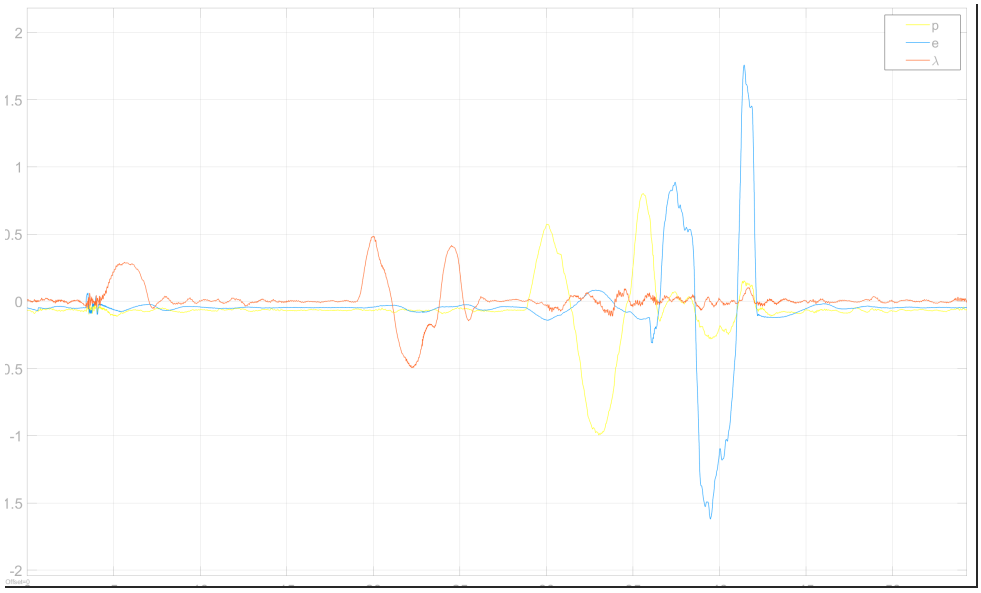


Figure 8: p, e and λ out from the gyroscope

we get the new C-matrix

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

We will be using this C-matrix from this point going forward.

4.3 State estimator

The linear observer we will be using to estimate the states of the system is the Luenberger observer and has the following form:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \tag{35}$$

The **L**-matrix can be chosen such that the estimation error tends to zero.

$$\dot{\hat{\mathbf{x}}} - \dot{\mathbf{x}} = \dot{\mathbf{e}} = (\mathbf{A} - \mathbf{L}\mathbf{C})\mathbf{e} \tag{36}$$

If the pair $\{\mathbf{A}, \mathbf{C}\}$ is observable then the poles of $\mathbf{A} - \mathbf{L}\mathbf{C}$ can be placed as we like by choice of **L**.

It was now time to place some poles and get the helicopter up and flying. We used the place-function in Matlab: $\mathbf{L} = place(\mathbf{A}', \mathbf{C}', p)'$ to place the poles and thereby determine the observer gain L. Our first matrix of poles, p, was $\begin{bmatrix} -1 & -10 & -1 & -10 & -10 \end{bmatrix}^T$ which gave us unstable oscillations for pitch. We tried changing to $\begin{bmatrix} -1 & -2 & -1 & -10 & -10 \end{bmatrix}^T$ which gave us more stability, but we were still not happy with the result. We therefore decided to try complex-conjugate and faster poles, with the first being

$$p = \begin{bmatrix} -30 + 10i & -30 - 10i & -30 + 5i & -30 - 5i & -30 \end{bmatrix}^T$$

This gave us a really stable system, but with some measurement-noise. We therefore decided to try smaller, complex-conjugate poles evenly spread out in the left half-plane.

$$p = \begin{bmatrix} -5e^{i \cdot \frac{\pi}{8}} & -5e^{-i \cdot \frac{\pi}{8}} & -5e^{-2i \cdot \frac{\pi}{8}} & -5e^{-2i \cdot \frac{\pi}{8}} & -5 \end{bmatrix}^T$$

This gave us less noise on the cost of more oscillations and bad regulation to reference. To counter this we tried an even less aggressive tuning.

$$p = \begin{bmatrix} -0.1e^{i \cdot \frac{\pi}{8}} & -0.1e^{-i \cdot \frac{\pi}{8}} & -0.1e^{-2i \cdot \frac{\pi}{8}} & -0.1e^{-2i \cdot \frac{\pi}{8}} & -0.1 \end{bmatrix}^T$$

As expected this made the observer too slow, so slow that it was unstable, it did though have very little noise. For the observer to work our chosen poles

must be faster than the system poles given by the eigenvalues of $\mathbf{A} - \mathbf{BK}$. We reasoned that oscillations probably was not a good choice for a helicopter, therefore choosing real poles after this.

$$p = \begin{bmatrix} -30 & -30 & -30 & -30 & -30 \end{bmatrix}^\top$$

Coinciding poles and a fairly fast pole choice worked well. We tried even more negative and coinciding poles, but the fast response was accompanied by noise being allowed through our observer. The poles above turned out to be a good midpoint.

4.4 Discussion

After having implemented the IMU, certain new problems appeared. The noise was expected and the observer dealt with it reasonably well. However, when tuning our helicopter we noticed another problem. The travel and elevation rate seemed suddenly to be coupled. We hypothesized that this was due to centripetal force, which we did not account for in our transformation from the accelerometer measurements to elevation position. However, we were confused when the elevation rate bias was dependent on travel direction since the centripetal force does not change direction. When inspecting the transform from gyroscope measurements to pitch, elevation and travel rate the reason became clear as the gyroscope measurements would be opposite dependent on travel direction. If our measured e_{IMU} changes, then our regulator will react on false premises.

5 Part IV – Kalman filter

In this part of the lab we introduce the Kalman filter, which is an observer that minimizes the variance of the estimation error by giving us a estimator feedback gain matrix. We will use this as an alternative to the Luenberger observer in the previous task. The discrete-time Kalman filter we will be using can be described by the following equations:

Correction:

$$\mathbf{L}[k] = \bar{\mathbf{P}}[k]\mathbf{C}_d^T(\mathbf{C}_d\bar{\mathbf{P}}[k]\mathbf{C}_d^T + \mathbf{R}_d)^{-1} \tag{37a}$$
$$\hat{\mathbf{x}}[k] = \bar{\mathbf{x}}[k] + \mathbf{L}[k](\mathbf{y}[k] - \mathbf{C}_d\bar{\mathbf{x}}[k]) \tag{37b}$$
$$\hat{\mathbf{P}}[k] = (\mathbf{I} - \mathbf{L}[k]\mathbf{C}_d)\bar{\mathbf{P}}[k](\mathbf{I} - \mathbf{L}[k]\mathbf{C}_d)^T + \mathbf{L}[k]\mathbf{R}_d\mathbf{L}^T[k] \tag{37c}$$

Prediction:

$$\bar{\mathbf{x}}[k+1] = \mathbf{A}_d\hat{\mathbf{x}}[k] + \mathbf{B}_d\mathbf{u}[k] \tag{38a}$$
$$\bar{\mathbf{P}}[k+1] = \mathbf{A}_d\hat{\mathbf{P}}[k]\mathbf{A}_d^T + \mathbf{Q}_d \tag{38b}$$

To complete this part of the lab we had to discretize the continous-time system to find $\mathbf{A}_d$, $\mathbf{B}_d$ and $\mathbf{C}_d$, for this we used the c2d function with 0.002s timestep T_s.

$$\mathbf{A}_d = e^{\mathbf{A}T_s} \tag{39a}$$
$$\mathbf{B}_d = \int_0^{T_s} e^{\mathbf{A}\tau}d\tau\mathbf{B} \tag{39b}$$
$$\mathbf{C}_d = \mathbf{C} \tag{39c}$$

$\mathbf{Q}_d$ was left undetermined for now, as it served as a tuning variable and $\mathbf{R}_d$ was to be determined experimentally later.

5.1 Discretization

In this part we used the following discrete-time model to describe our system:

$$\mathbf{x}[k+1] = \mathbf{A}_d\mathbf{x}[k] + \mathbf{B}_d\mathbf{u}[k] + \mathbf{w}_d[k] \tag{40a}$$
$$\mathbf{y}[k] = \mathbf{C}_d\mathbf{x}[k] + \mathbf{v}_d[k] \tag{40b}$$
$$\mathbf{w}_d \sim \mathcal{N}(\mathbf{0},\mathbf{Q}_d), \quad \mathbf{v}_d \sim \mathcal{N}(\mathbf{0},\mathbf{R}_d) \tag{40c}$$

Our initial thoughts was that higher values on the diagonal of $\mathbf{Q}_d$ would mean less influence on the system by the filter, since we trust the estimates less. The testplan consisted of $\mathbf{Q}_d = inf$, $\mathbf{Q}_d = 0$ and $\mathbf{Q}_d = 10^{-7}$. Since higher $\mathbf{Q}_d$ would mean less influence by the filter we expected that inf along

the diagonal would mean basically overriding the filter and therefore be equal to y. For $\mathbf{Q}_d = 0$ we expected the system to be extremely slow. The last test-value was a random guess of something small that would still make a impact on the system.

5.2 Noise estimate and experimentation

We used the controller with integral LQR from part 2 to stabilize the helicopter at the linearization point and we found that the covariance of the estimated measurementnoise was much higher while flying, as we could expect.
With our $\mathbf{R}_d$ matrix ready and implementation done we could now move on to experimentation. We started off with $\mathbf{Q}_d = inf$ and to our surprise nothing worked. It seemed like our program could not handle inf so we decided to change it to 1000 along the diagonal, which gave us a response similiar to y, which we expected. For $\mathbf{Q}_d = 0$ the rotors actually went backwards, even when laying down on the table. We then wanted to test with a small $\mathbf{Q}_d$ value to see if we could get the helicopter to fly, and with $\mathbf{Q}_d = 10^{-7}$ we got a response that was quick and at the same time pretty similiar to y, as seen in figure fig. 9 and fig. 10. The prediction variance was also found as seen in fig. 11. Even though we got a nice response from $\mathbf{Q}_d = 10^{-7}$, it was too

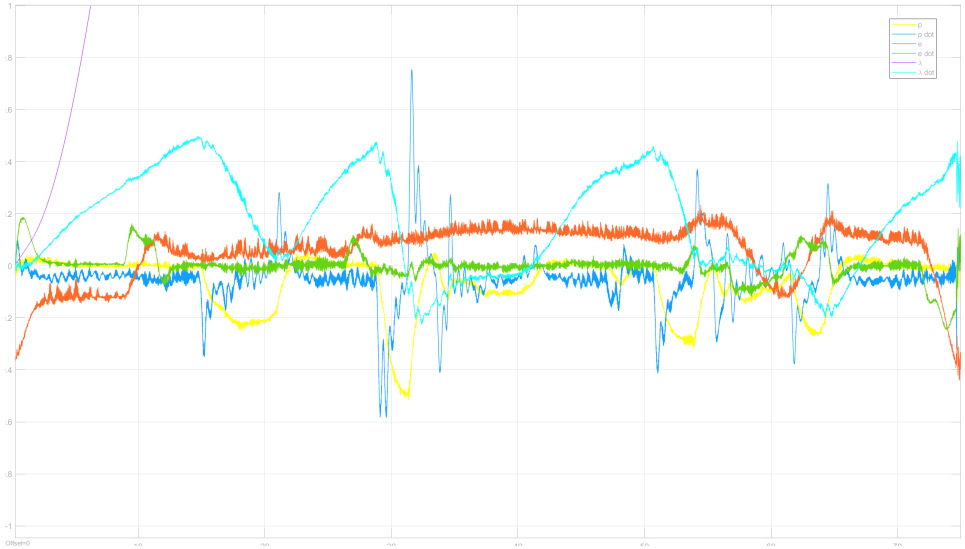


Figure 9: $\hat{x}$ with Q_d= 10^{-7}

low in our opinion, because we had a pretty noisy $\hat{x}$. $\mathbf{Q}_d = 10^{-6}$ was still noisy and we ended up with $\mathbf{Q}_d = 10^{-5}$ as the best result as seen in fig. 12 and fig. 13.
For the final part of this lab we added a manual switch to the new-data signal coming from the IMU, such that we could disconnect it while the program

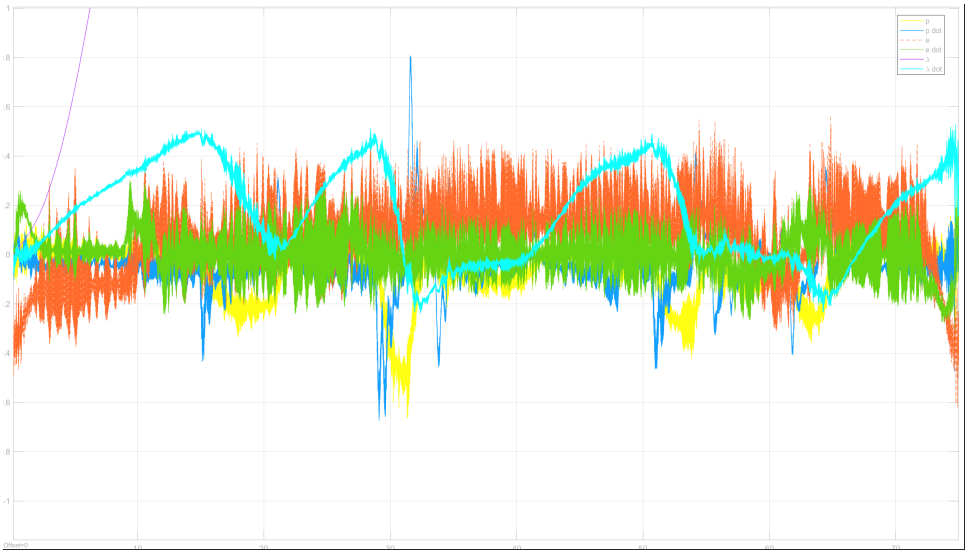


Figure 10: Real values of y

Figure 11: Low and stable $\hat{p}$

is running. This worked very badly, and the helicopter would lose control in matter of seconds after disconnection. We can see from fig. 14 and fig. 15 that there is something very wrong about specially estimated travel, which increases dramatically right after disconnecting the new-data signal at $t \sim 15.2s$. The datasignal gets reconnected at $t \sim 18.5s$ and we can see values reestablishing immediately.

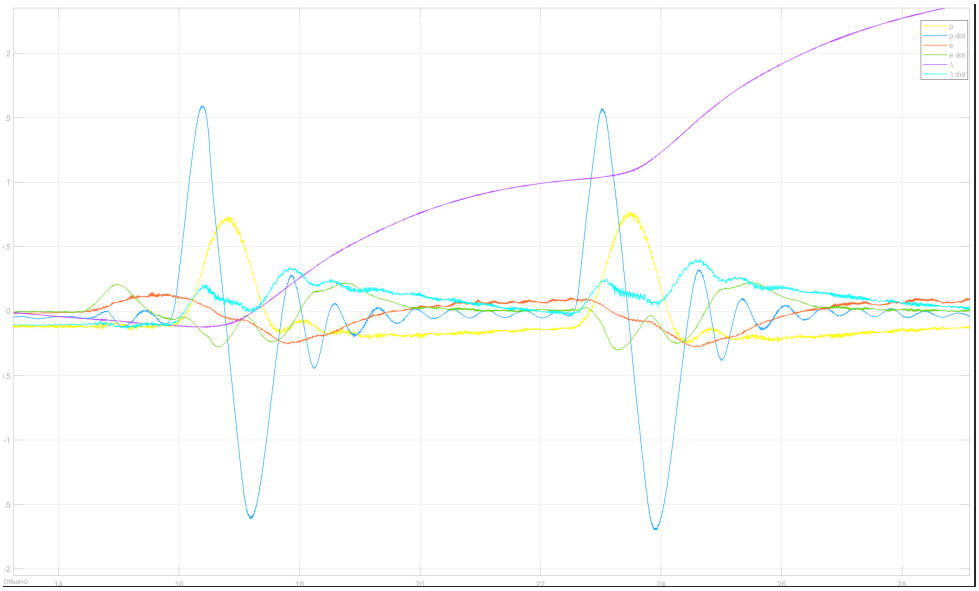


Figure 12: $\hat{x}$ with Q_d= 10^{-5}

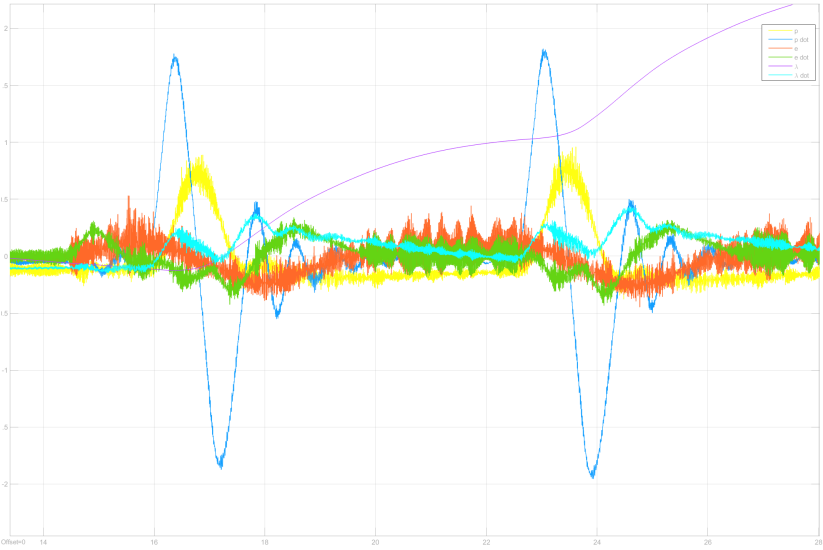


Figure 13: y with Q_d= 10^{-5}

5.3 Bonustask: Van Loan's method

When discretizing our continuous-time model the equation for the model-noise variance $\mathbf{Q}_d$ becomes

$$\mathbf{Q}_d = \int_0^{T_s} e^{\mathbf{A}\tau} \mathbf{Q} e^{\mathbf{A}^\top \tau} d\tau \tag{41}$$

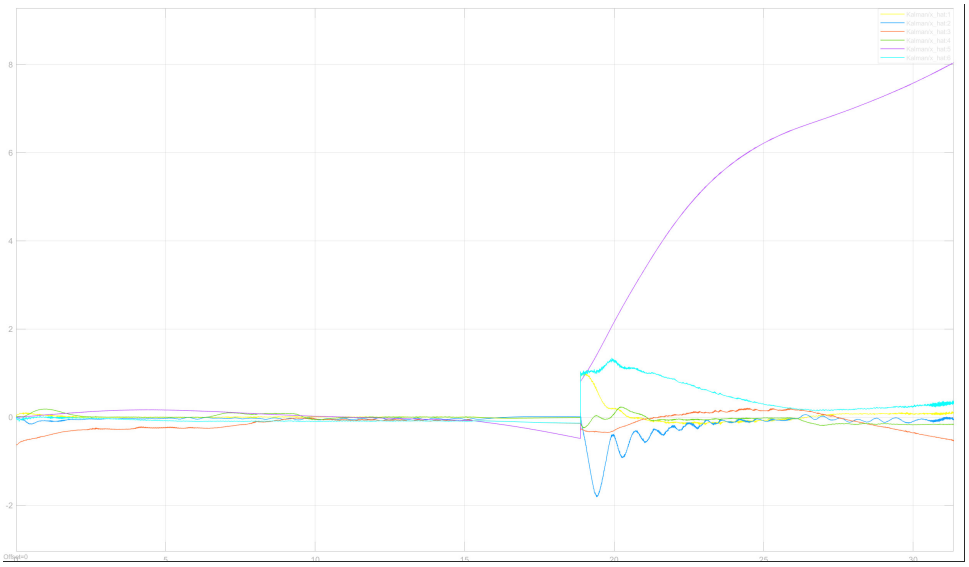


Figure 14: $\hat{x}$ when disconnecting/connecting new-data signal

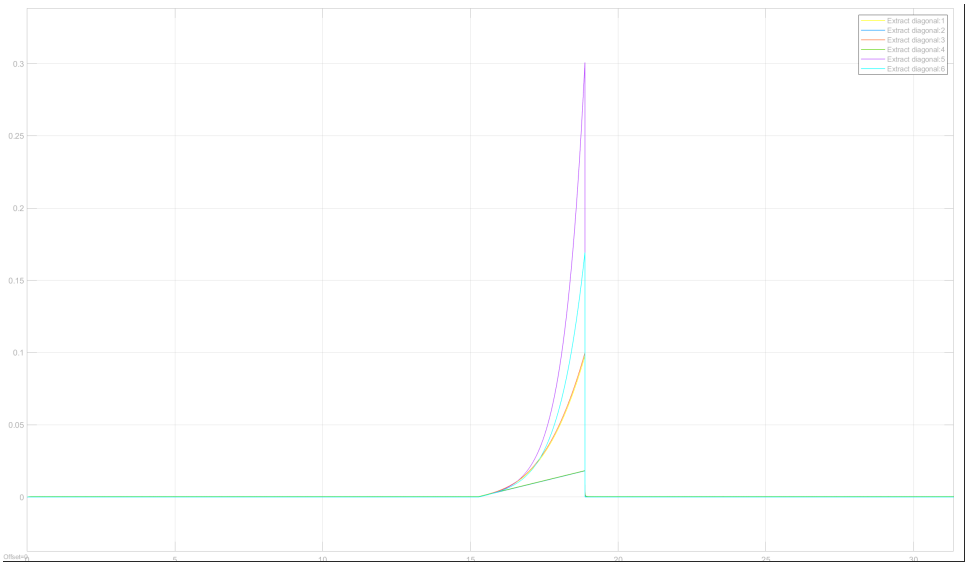


Figure 15: $\hat{p}$ when disconnecting/connecting new-data signal

$\mathbf{Q}$ is the continuous-time model-noise variance. To solve the equation we used Van Loan's method[2].

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{Q} \\ \mathbf{0} & -\mathbf{A}^\top \end{bmatrix} \tag{42a}$$

$$\mathbf{M} = e^{\mathbf{H}T_s} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{0} & \mathbf{M}_{22} \end{bmatrix} \tag{42b}$$

$$\mathbf{Q}_d = \mathbf{M}_{12}\mathbf{M}_{11}^\top \tag{42c}$$

Now we had to find a reasonable estimate for the continuous-time $\mathbf{Q}$. A seemingly equally difficult task as directly estimating $\mathbf{Q}_d$. To do so we implemented a pure simulator of our system fig. 26 and calculated the mean square error between our simulation and the encoder measurements on the simultaneously running helicopter. This served as our measure of inaccuracy, or randomness, between helicopter and model. Creating a diagonal matrix with these values on the diagonal and using this as our $\mathbf{Q}$ did surprisingly give us a $\mathbf{Q}_d$ in the same magnitude as our most well-behaved guesses. The diagonal elements were in the 10^{-5} range, and the off-diagonals smaller or zero.

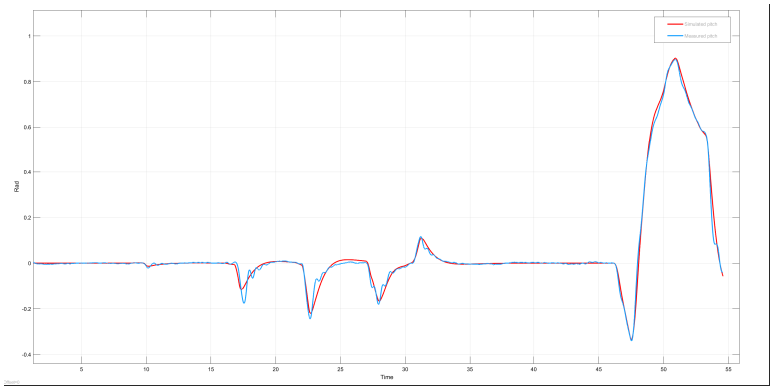


Figure 16: Simulated pitch plotted against measured pitch

5.4 Bonustask: IMU-bias estimation

Earlier in the assignement we noticed a bias between our IMU measurements for p and e compared to the actual values. We counteracted this by adding appropriate constants to the parameters. In actuality this bias is slowly drifting. Therefore we tried to use our Kalman filter to estimate the bias by adding p_{bias} and e_{bias} to our state vector. As there is no coupling between these states, the other states, the derivatives or the input we padded our $\mathbf{A}$- and $\mathbf{B}$-matrices with zeros. The $\mathbf{Q_d}$-matrix was also adjusted to the new number of states to ensure that the noise influence our new states. We model this noise to be the reason for the drift. Finally we modeled the IMU measurements on pitch and elevation to be the sum of our estimate values and the bias through a new $\mathbf{C}$-matrix.

$$\mathbf{C}_{bias} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{43}$$

Our new system was no longer fully controllable as expected, still rank 6, since we have no way to influence our two newest states. But more surprisingly the observability matrix did have rank 7, not 8. As it turned out the elevation bias was not observable in our model. Moving on we reduced our state-space to the 7 observable states. Our helicopter did not respond kindly to the new model, being short on time we decided to optimize the tuning for the 6 dimensional state-space instead of moving on with this model.

6 Conclusion

Through experimentation on an increasingly challenging and realistic model of a helicopter theoretical knowledge in control theory was put to use. We found that increasing model complexity warrant more complex control designs. Nevertheless, a complex controller design might not be worth the difference in outcome compared to a less complex one. on a well-posed system. This illustrates the power control systems have, and also the importance of how the system is designed. Whenever a system is posed well, it is more convenient to control, while a good controller can fight errors that result from assumptions and design flaws.

The model did have some unique challenges connected to its design. Normally a drone or helicopter is not restricted to moving on a ball. This lead some unusual effects due to centripetal force.

A Constants

Table 1: Parameters and values.

Symbol	Parameter	Value	Unit
g	Gravitational constant	9.81	m/s^2
l_c	Distance from elevation axis to counterweight	0.46	m
l_h	Distance from elevation axis to helicopter body	0.66	m
l_p	Distance from pitch axis to motor	0.175	m
m_c	Counterweight weight	1.92	kg
m_p	Mass of helicopter motor	0.72	kg
$V_{S,0}$	Voltage needed to lift elevation horizontal	7.4	V
J_p	Moment of inertia for pitch	0.044	kg m^2
J_e	Moment of inertia for elevation	1.03	kg m^2
J_λ	Moment of inertia for travel	1.08	kg m^2

Table 2: Derived constants

Symbol	Derivation
J_p	$2m_p l_p^2$
J_e	$m_c l_c^2 + 2m_p l_h^2$
J_λ	$m_c l_c^2 + 2m_p(l_h^2 + l_p^2)$
K_f	$-\frac{L_2}{l_h V_{S,0}}$
L_1	$K_f l_p$
L_2	$m_c l_c g - 2m_p l_h g$
L_3	$K_f l_h$
L_4	$K_f l_h$
K_1	L_1/J_p
K_2	L_3/J_e
K_3	$L_4 V_{S,0}/J_\lambda$

B MATLAB Code

B.1 Kalman_correction.m

```
function [x_hat,P_hat] = Correction(y,new_data,C_d, R_d, x_bar,P_bar)

    if(new_data ~= 0)
        L = P_bar*C_d'/(C_d*P_bar*C_d'+R_d);
        x_hat = x_bar+L*(y-C_d*x_bar);
```

```
        P_hat = (eye(6)-L*C_d)*P_bar*(eye(6)-L*C_d)'+L*R_d*L';
    else
        x_hat = x_bar;
        P_hat = P_bar;
    end
end
```

B.2 Kalman_prediction.m

```
function [x_bar,P_bar] = Prediction(u, x_hat, A_d, B_d, Q_d, P_hat)
    x_bar = A_d*x_hat+B_d*u;
    P_bar = A_d*P_hat*A_d' + Q_d;
end
```

B.3 accelTransform.m

```
function [p, e] = orientationFromAcceleration(a)
    ax = a(1); ay = a(2); az = a(3);
    if az == 0
        p = sign(ay)*pi/2;
    else
        p = atan(ay/az);
    end

    if ay == 0 && az == 0
        e = sign(ax)*pi/2;
    else
        e = atan(ax/sqrt(ay^2 + az^2));
    end
end
```

B.4 euler_rates.m

```
function euler_rates = gyro_vec_to_euler_rates(gyro_vec,euler_angles)

    phi = euler_angles(1);
    theta = euler_angles(2);
    %psi = euler_angles(3);


    T = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
         0, cos(phi),            -sin(phi);
         0, sin(phi)/cos(theta), cos(phi)/cos(theta)];
euler_rates = T*gyro_vec;
```

C Simulink Diagrams

Below follows our Simulink diagrams for the different parts.

C.1 Monovariable control

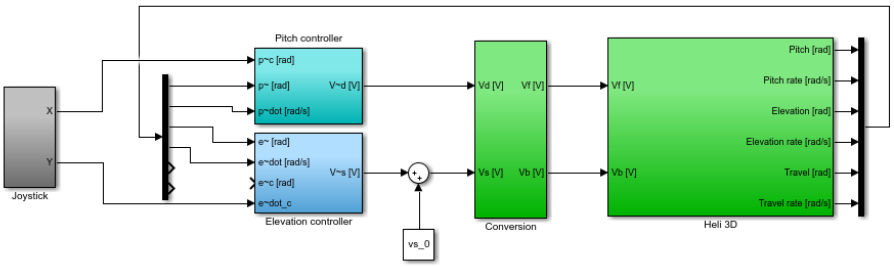


Figure 17: The Simulink diagram for part 1.

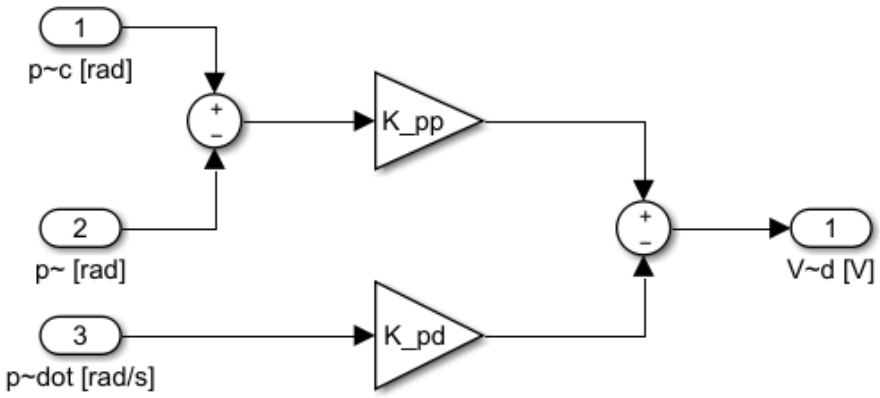


Figure 18: PD-controller block-diagram

C.2 Multivariable control

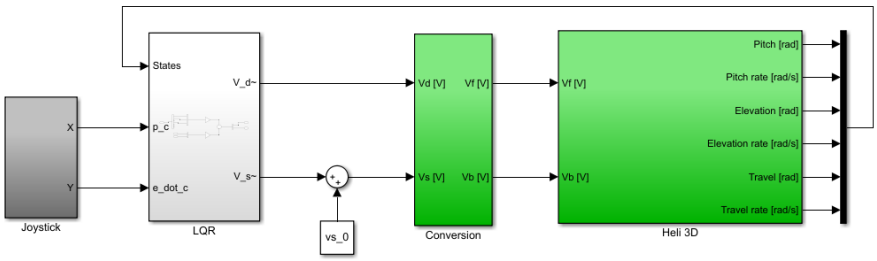


Figure 19: The Simulink diagram for part 2.

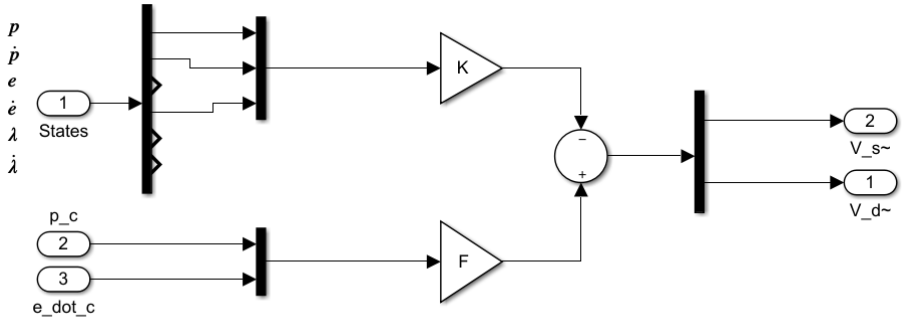


Figure 20: LQR without integral action

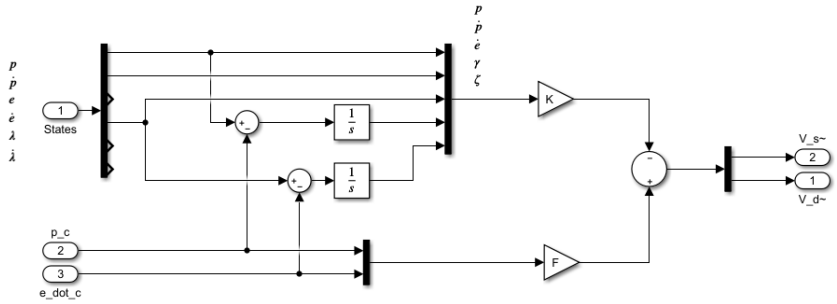


Figure 21: LQR with integral action

C.3 Luenberger observer

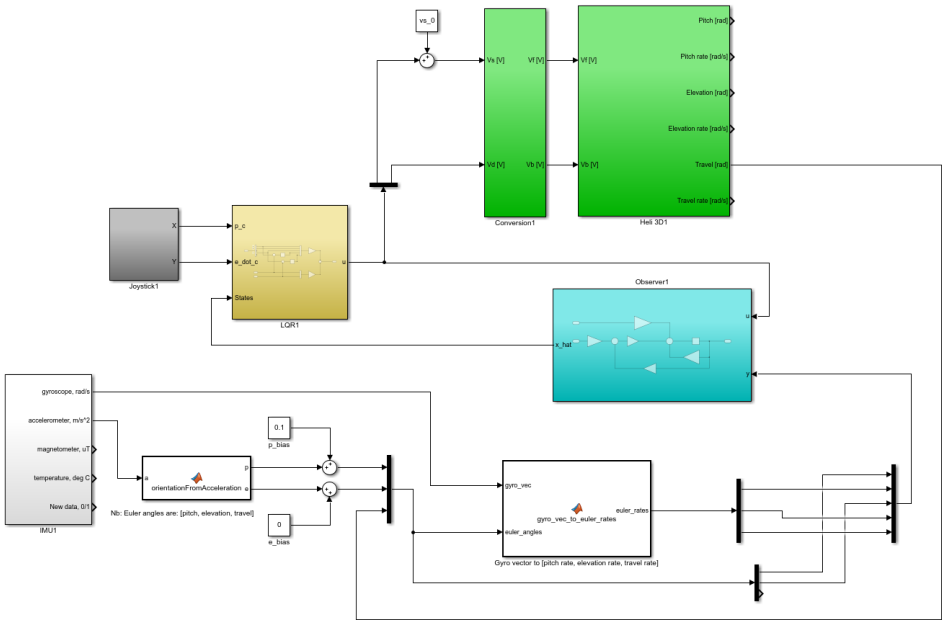


Figure 22: The Simulink diagram for part 3.

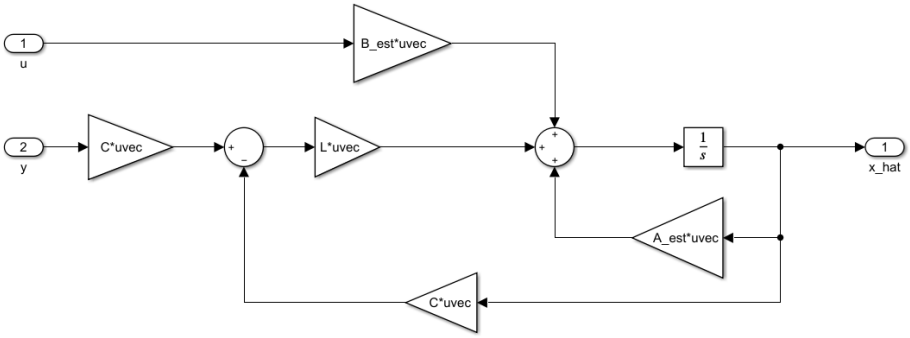


Figure 23: Observer block-diagram

C.4 Kalman filter

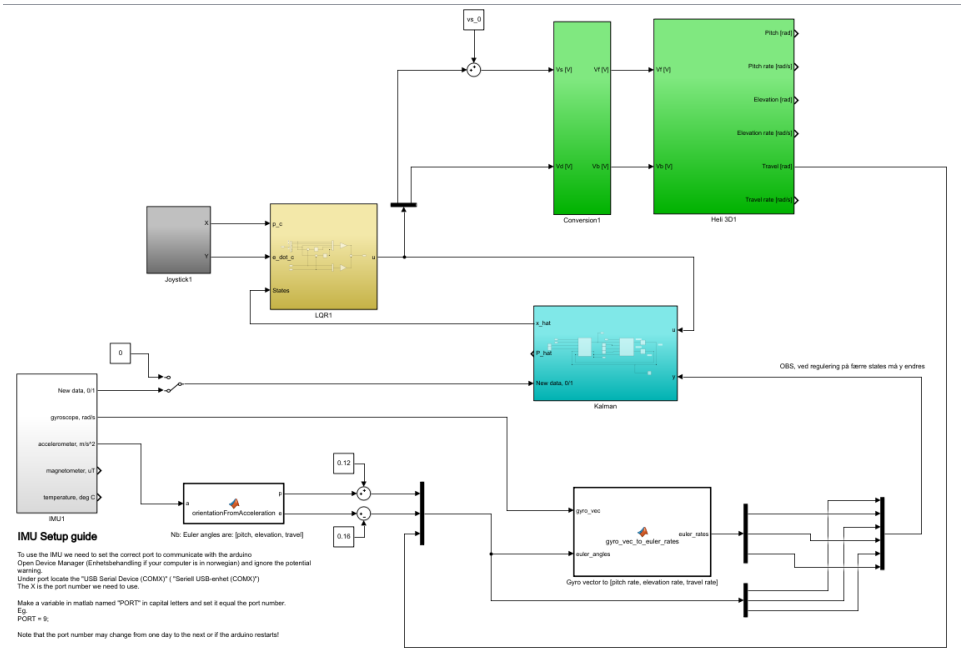


Figure 24: The Simulink diagram for part 4.

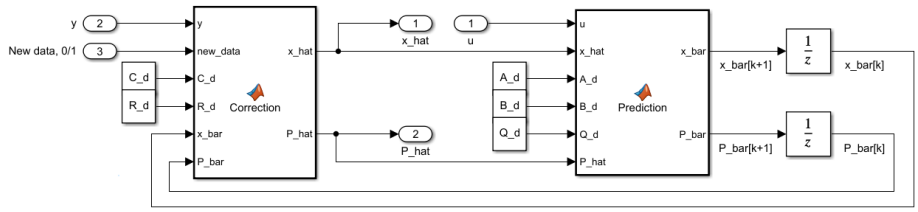


Figure 25: Kalman filter block-diagram

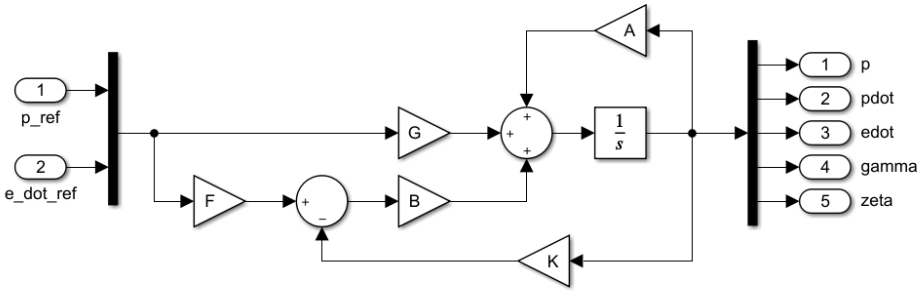


Figure 26: Simulator block-diagram

References

[1] J. Balchen, T. Andresen, and B. Foss. *Reguleringsteknikk*. 2016.

[2] N. Wahlström, P. Axelsson, and F. Gustafsson. *Discretizing stochastic dynamical systems using Lyapunov equations*. `https://www.researchgate.net/publication/260089269_Discretizing_stochastic_dynamical_systems_using_Lyapunov_equations`. 2017.