TTK4115 - Linear System Theory
Boat Lab
Discrete Kalman Filter Applied to a Ship Autopilot

Group 32

Aleksander Asp - 767827
Marius Eskedal - 767880
Per Kvinnesland Omvik - 768682

November 2017

NTNU

Department of engineering cybernetics

Contents

Introduction **3**

System model **4**

1 Part I - Identification of boat parameters **5**
- 1.a Transfer function from δ to ψ . . . 5
- 1.b Boat parameters . . . 5
- 1.c Boat parameters with waves and measurement noise . . . 6
- 1.d Step response . . . 7

2 Part II - Identification of wave spectrum model **9**
- 2.a Estimating PSD of waves . . . 9
- 2.b Deriving analytical expression for PSD . . . 9
- 2.c Identifying resonance frequency ω_0 . . . 10
- 2.d Identifying damping factor λ . . . 10

3 Part III - Control system design **11**
- 3.a PD-controller . . . 11
- 3.b No disturbances . . . 12
- 3.c Current disturbance . . . 13
- 3.d Wave disturbance . . . 14

4 Part IV - Observability **15**
- 4.a State space model . . . 15
- 4.b No disturbances . . . 15
- 4.c Current disturbance . . . 16
- 4.d Wave disturbance . . . 16
- 4.e Both current and wave disturbance . . . 16

5 Part V - Discrete Kalman filter **18**
- 5.a Discretization of continous model . . . 18
- 5.b Estimating the variance of the measurement noise . . . 18
- 5.c Constructing the Kalman filter . . . 19
- 5.d Controlling the ship using feed forward from the estimated bias . . . 20
- 5.e Controlling the ship with wave disturbances . . . 21

A Simulink models **24**
- 1 Part 5.1. . . . 24
- 2 Part 5.3. . . . 24
- 3 Part 5.5. . . . 25

B MATLAB scripts **27**
- 1 5.1.b) . . . 27
- 2 5.1.c) . . . 27
- 3 5.1.d) . . . 28
- 4 5.2.a) . . . 28
- 5 5.2.c) . . . 29
- 6 5.2.d) . . . 29
- 7 5.3.b) . . . 29
- 8 5.3.c) . . . 30
- 9 5.3.d) . . . 30

10 5.5.a) 30
11 5.5.b) 31
12 5.5.c) 31
13 5.5.d) 31
14 5.5.e) 32
15 5.5.e waves) 32

Introduction

The purpose of the given assignment can be divided into four main topics, which all connect. Firstly, using calculations, MATLAB and Simulink, we desire to model and simulate a system affected by stochastic signals. In order to do this, we will utilize some simple identification techniques on parameters that are not explicitly given, which is the second topic. Next, an autopilot is to be designed by employing basic control theory. Lastly we seek to implement a discrete Kalman filter to improve control through wave filtering and disturbance estimation.

System model

The state vector of the model used in this report is $\mathbf{x} = [\xi_w \ \psi_w \ \psi \ r \ b]^T$ in which the individual states are given as

$$\dot{\xi}_w = \psi_w \tag{1a}$$
$$\dot{\psi}_w = -\omega_0^2 \xi_w - 2\lambda\omega_0\psi_w + K_w w_w \tag{1b}$$
$$\dot{\psi} = r \tag{1c}$$
$$\dot{r} = -\frac{1}{T}r + \frac{K}{T}(\delta - b) \tag{1d}$$
$$\dot{b} = w_b \tag{1e}$$
$$y = \psi + \psi_w + v \tag{1f}$$

Where ψ is the average heading of the boat without the high frequency component ψ_w caused by wave disturbances. Furthermore, r is the rotation velocity of the ship about the vertical z-axis, δ is the applied rudder angle relative to the BODY reference frame, b is a bias to the rudder angle and y is the measured heading from the compass. w_w, w_b and v are white noise processes caused by waves, current and the measuring itself.
Following this, the system can be written as a continuous state space model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) + \mathbf{E}\mathbf{w}(t), \quad y(t) = \mathbf{C}\mathbf{x}(t) + v(t) \tag{2}$$

With $\mathbf{x}$ as before, $u = \delta$ and $\mathbf{w} = [w_w \ w_b]^T$.

1 Part I - Identification of boat parameters

1.a Transfer function from δ to ψ

Assuming that there are no disturbances present in our model, we can calculate the transfer function $H(s)$ from δ to ψ, parameterized by T and K. Using (1c) and (1d) we derive the following:

$$\mathcal{L}\{\dot{r}\} = \mathcal{L}\{-\frac{1}{T}r + \frac{K}{T}(\delta - b)\} \qquad , b = 0 \tag{1.1}$$

$$\Rightarrow sR(s) + \frac{R(s)}{T} = \frac{K\delta(s)}{T}$$

$$\Rightarrow R(s) = \frac{K\delta(s)}{sT+1} \tag{1.2}$$

$$\mathcal{L}\{\dot{\psi}\} = \mathcal{L}\{r\}$$

$$\Rightarrow \psi(s) = \frac{R(s)}{s} \tag{1.3}$$

Combining (1.2) and (1.3) we get the transfer function $H(s)$:

$$\frac{\psi(s)}{\delta(s)} = \frac{K}{s(sT+1)} \tag{1.4}$$

This transfer function represents how the general heading $\psi(s)$ is affected by the rudder position $\delta(s)$. From (1.1) we can also determine the units of T and K. Because $\dot{r}$ has unit $[\frac{deg}{s^2}]$, T must have unit $[s]$ and K must have unit $[\frac{1}{s}]$ since δ has unit $[deg]$.

1.b Boat parameters

We want to identify the boat parameters T and K in smooth weather conditions, which corresponds to turning off all the disturbances in our model. Applying a sine input with amplitude 1 will result in an output where the amplitude of the sine wave signal is equal to $|H(j\omega)|$. This is implemented in Simulink and shown in Figure 15 in Appendix A. Repeating this with a different frequency on the input gives us two equations with two unknowns. The given frequencies of our input are $\omega_1 = 0.005$ and $\omega_2 = 0.05$. The resulting amplitudes are measured using the max and min functions in MATLAB after steady state is acquired. The simulation is shown in Figure 1 and the calculations for T and K are as follows:

$$\omega_1 = 0.005 \Rightarrow |H(j\omega_1)| = A_1 = 29.358 \tag{1.5}$$

$$\omega_2 = 0.05 \Rightarrow |H(j\omega_2)| = A_2 = 0.831 \tag{1.6}$$

$$A_1 = \left|\frac{K}{j\omega_1(j\omega_1 T + 1)}\right|$$

$$A_1 = \frac{K}{\sqrt{(\omega_1^2 T)^2 + (\omega_1)^2}}$$

$$\Rightarrow K = A_1\omega_1\sqrt{T^2\omega_1^2 + 1} \tag{1.7}$$

Using ω_2 and A_2 instead and solving for T yields:

$$T = \frac{\sqrt{K^2 - A_2^2\omega_2^2}}{A_2\omega_2^2} \tag{1.8}$$

With equations (1.7) and (1.8), as well as the values from (1.5) and (1.6) we calculate the values of K and T:

$$K = 0.156 \tag{1.9a}$$
$$T = 72.439 \tag{1.9b}$$

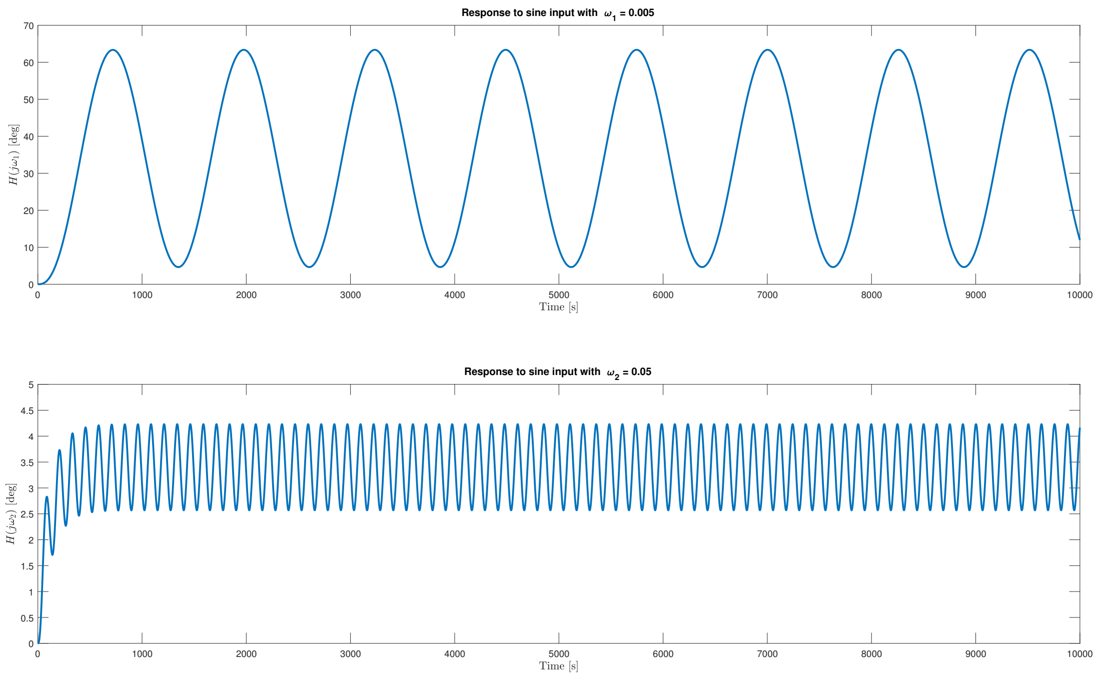

Figure 1: Simulations with two different frequencies, ω_1 and ω_2, where there are no disturbances present in the system.

1.c Boat parameters with waves and measurement noise

Using the same equations and approach as in section 1.b, but with waves and measurement noise turned on in the simulation, we calculate new values for K and T:

$$K = 0.163 \tag{1.10a}$$
$$T = 14.503 \tag{1.10b}$$

With these disturbances turned on, our estimates change. The estimate for K is still adequate, but the T estimate is far from its value in smooth weather conditions. The noise creates random spikes on our output which makes it difficult to get a good estimate in simulated rough weather. This is illustrated in Figure 2.

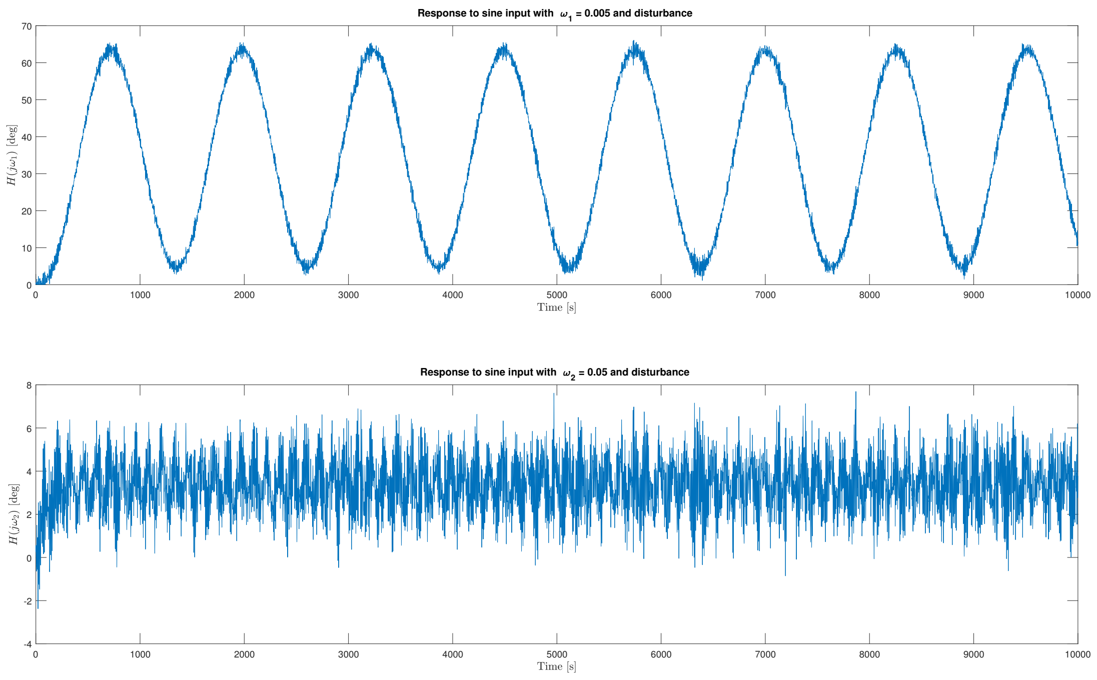

Figure 2: Simulations with two different frequencies, ω_1 and ω_2, where wave disturbance and measurement noise are present in the system.

1.d Step response

A step input of one degree is applied to the rudder at $t = 0$ to check if the model is reasonable. First we apply the step function with no disturbances in the model and we use the corresponding T value given in (1.9b). The simulation is shown in Figure 3.

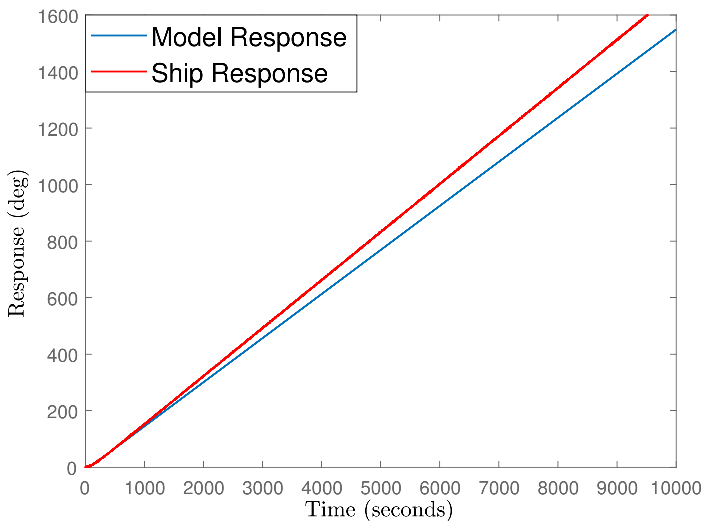

Figure 3: Step response of the ship and the ship model, using the boat parameters associated with no disturbances

Comparing the response of the ship and the model, we observe some deviations. The model strays further from the ship as time passes. However, the offset is of little significance for as long as 1500 - 2000 seconds. Holding a constant rudder angle for this long is unrealistic, meaning that the estimate is reasonable for our purpose. If wave disturbance and measurement noise are turned on, the offset is significantly larger from the beginning, as seen in Figure 4. Even though the response of the model and ship intersect at a point, the offset is too large when t is smaller or larger than at this point, rendering the model unusable in real life situations. The Simulink model for simulation with and without disturbances is shown in Figure 16 in Appendix A.

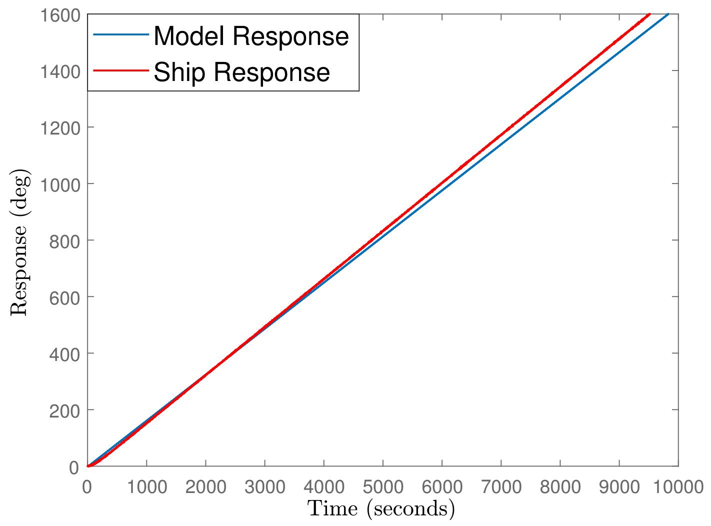

Figure 4: Step response of the ship and the ship model, using the boat parameters where wave disturbance and measurement noise are turned on.

2 Part II - Identification of wave spectrum model

2.a Estimating PSD of waves

In this section we are going to make an analytic model for how the waves disturb our heading. This is done by evaluating an estimate of the power spectral density (PSD) based on samples of the disturbance and choosing parameters for our analytic model that match the estimated spectrum. For our estimate we perform a windowed fourier transform via Welch's method that will result in an accurate estimate for our modelling purposes. This is done through MATLAB command `pwelch()` as shown in Appendix B. The following plot shows for which frequencies the wave disturbance affects our heading.

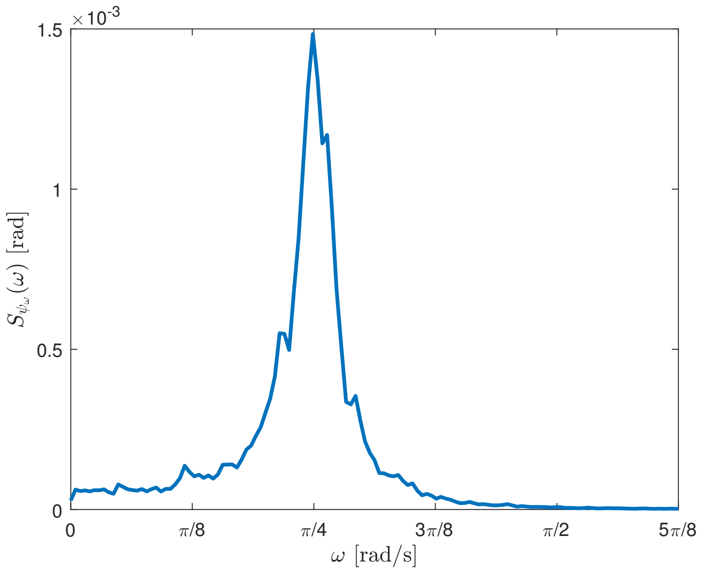

Figure 5: Estimated PSD of waves in relation to heading ψ

2.b Deriving analytical expression for PSD

As we have our estimate, we now need an analytical expression for the PSD of the waves so that we compute the unknowns based on the this estimate.

$$H(s) = \frac{w_w(s)}{\psi_w(s)}$$

$$\dot{\xi_w} = \psi_w \Rightarrow \xi_w(s) = \frac{\psi_w(s)}{s}$$

$$\psi_w(s)(s + \frac{\omega_0^2}{s} + 2\lambda\omega_0) = K_w w_w(s)$$

$$H_{w_w\psi_w}(s) = \frac{\psi_w(s)}{w_w(s)} = \frac{K_w s}{s^2 + 2\lambda\omega_0 s + \omega_0^2} \tag{2.1}$$

The analytical expression for the PSD, $P_{\psi_w}(\omega)$, is found by

$$P_{\psi_w}(\omega) = P_{w_w}(\omega) \cdot H_{w_w\psi_w}(-j\omega) \cdot H_{w_w\psi_w}(j\omega) \tag{2.2}$$

Noting that w_w is a zero mean white noise process with unity variance, we get the following PSD, where $\delta(t)$ is the Dirac delta function.

$$\mathcal{R}_{w_w}(t) = \sigma^2\delta(t) = \delta(t) \quad \Rightarrow \quad \mathcal{F}\{\mathcal{R}_{w_w}\} = P_{w_w}(\omega) = 1$$

$$\Rightarrow P_{\psi_w}(\omega) = H_{w_w\psi_w}(-j\omega) \cdot H_{w_w\psi_w}(j\omega)$$

$$\Rightarrow P_{\psi_w}(\omega) = \frac{K_w^2\omega^2}{\omega^4 + \omega_0^4 + 2\omega_0^2\omega^2(2\lambda^2 - 1)} \tag{2.3}$$

This equation has three unknowns, K_w, ω_0 and λ which we now will identify to complete our model.

2.c Identifying resonance frequency ω_0

To identify the w_0 parameter in our analytical model (2.3), we find the angular frequency that corresponds to max power, σ^2, in our estimated PSD. From figure 5 we observe that the corresponding angular frequency is about $\frac{\pi}{4}$ and by closer inspection through MATLAB functions, we note the following values

$$\omega_0 = 0.7823 \quad \sigma^2 = 0.0385 \tag{2.4}$$

2.d Identifying damping factor λ

To complete our model of the wave response, the damping factor λ is needed. λ was determined by using different values for λ in a defined interval and plotting the resulting PSD against S_{ψ_ω}.This resulted in the λ which gave the best fit, thus completing our analytical model of the waves. This is illustrated in Figure 6. The analytical model is important since we now can define our system, and later our Kalman filter. The values for λ and K_w, where K_w is defined as $K_w = 2\lambda\omega_0\sigma^2$, are

$$\lambda = 0.09 \quad K_w = 0.0054 \tag{2.5}$$

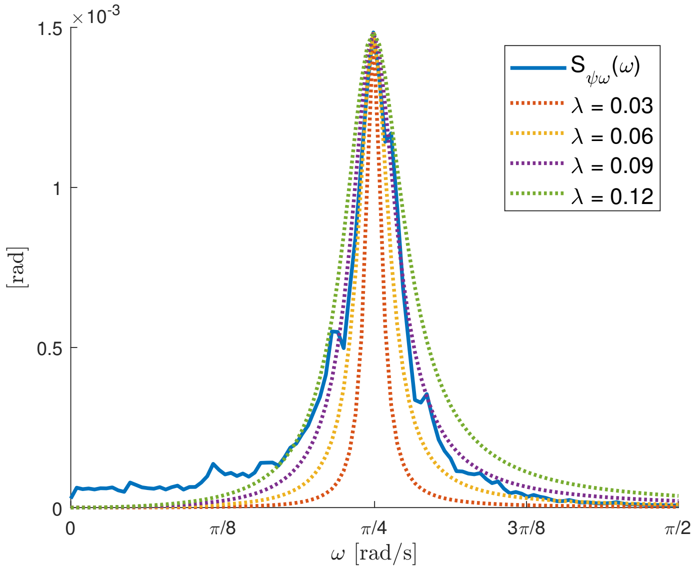

Figure 6: Curve fitting for different λ values

3 Part III - Control system design

3.a PD-controller

We seek to design an autopilot for the ship by using a reference course angle ψ_r for the ship to follow. The autopilot can be implemented as a PD-controller based on the transfer function from δ to ψ in equation (1.4).
The resulting transfer function for the open-loop system becomes

$$\begin{aligned} h_0(s) = H_{pd}(s)H_s(s) &= K_{pd}\frac{1+T_d s}{1+T_f s}\cdot\frac{K}{s(Ts+1)} \\ &= \frac{K_{pd}K}{s(1+T_f s)}, \end{aligned} \tag{3.1}$$

when T_d is chosen equal to T.
The desired cutoff frequency w_c and phase margin ϕ are $0.1\,\mathrm{rad\,s^{-1}}$ and $50°$ respectively. The phase margin of the system is defined as

$$\phi = \angle h_0(jw_c) + 180° \tag{3.2}$$

Inserting for ϕ and h_0:

$$\begin{aligned} \angle\left(\frac{K_{pd}K}{jw_c(1+T_f jw_c)}\right) &= -130° \\ \angle K_{pd}K - \angle jw_c - \angle(1+T_f jw_c) &= -130° \\ 0 - 90° - \arctan(T_f w) &= -130° \\ arctan(T_f w) &= 40° \\ \Rightarrow T_f = \frac{\tan 40°}{w_c} &= 8.391 \end{aligned} \tag{3.3}$$

Additionally, K_{pd} is found using the fact that the magnitude of the transfer function is equal to 1 at the cutoff frequency:

$$\begin{aligned} \left|\frac{K_{pd}K}{jw_c(1+T_f jw_c)}\right| &= 1 \\ \sqrt{\frac{(K_{pd}K)^2}{(T_f w_c^2)^2 + w_c^2}} &= 1 \\ \Rightarrow K_{pd} = \sqrt{\frac{(T_f w_c^2)^2 + w_c^2}{K^2}} &= 0.836 \end{aligned} \tag{3.4}$$

Where K is obtained from equation (1.9) in section 1.a.

3.b No disturbances

Simulating the system without any disturbances in our model, solely measurement noise included, returns a plot from which the quality of the autopilot can be determined. This plot is shown in Figure 7. Without any constraints, the rudder had an initial angle, δ, of over 200 degrees. To keep the rudder angle within its physical limit of ± 90 degrees, a saturation block is added in the Simulink model, which is shown in Figure 17 in appendix A.

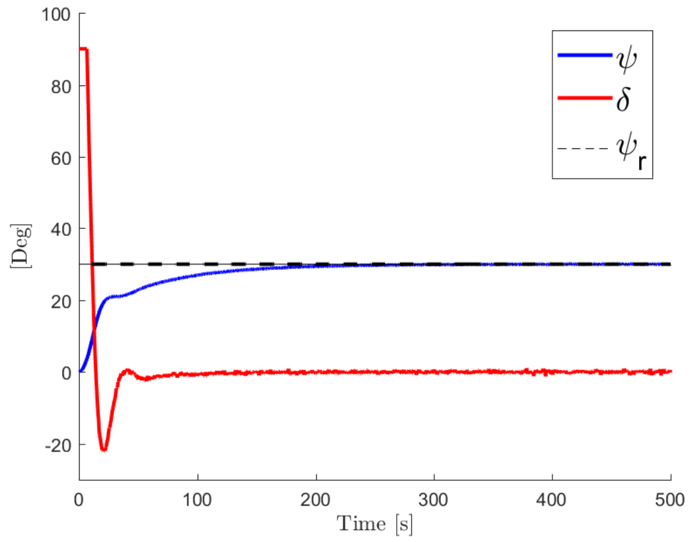

Figure 7: Autopilot without any disturbances

With the given reference heading of $\psi_r = 30°$, the compass value ψ, never exceeds its limits of ± 35 degrees so the model holds. Figure 7 shows that the compass value reaches its reference after approximately 300 seconds, and is only a couple degrees off after 180 seconds. This is an acceptable adjustment period for larger ships, meaning that the autopilot works under these conditions. The simulation also shows that the rudder changes direction long before the compass value has reached the reference. This is due to the derivative effect in the PD-controller which decreases when the average heading ψ, is changing too fast. Not having this effect would result in overshooting the reference due to the inertia and momentum of the ship.

3.c Current disturbance

Simulating the system with the current disturbance and measurement noise turned on yields a slightly different result, which is illustrated in Figure 8.

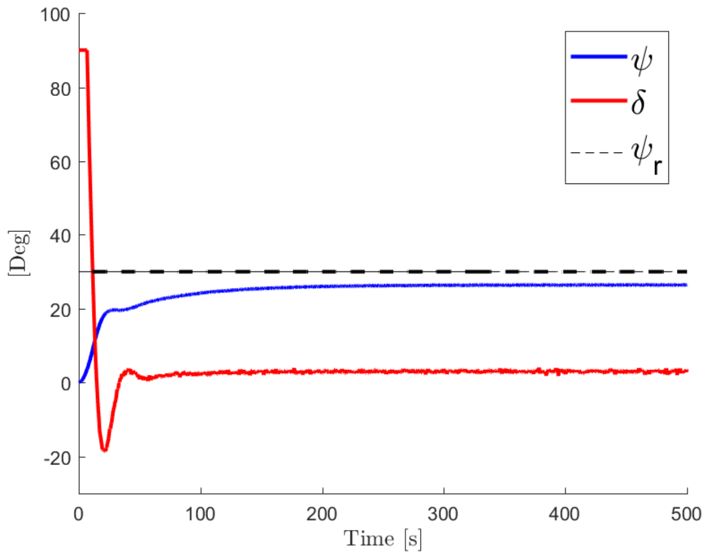

Figure 8: Autopilot with current disturbance and measurement noise.

Figure 8 shows that the rudder angle does not settle at 0 degrees after reaching steady state, but at approximately 3 degrees instead. This is because the current applies a somewhat constant force on the ship, which is compensated for by having a slight angle in the rudder. This is assumed to be the only effect of the current[1]. This results in the ship keeping true even though the rudder angle is not 0. The figure also reveals that our average heading is around 4 degrees below the reference after reaching steady state, which is deeply connected to our rudder angle bias. The controller will not change the average heading of the ship because the current disturbance cancels out the rudder angle, and the rudder angle will not change to attempt to alter the average heading closer to the reference because it is already non-zero and expects to be decreasing the deviation as is.

Having a constant offset of nearly 5 degrees on the compass value is not an acceptable deviation for the autopilot. The autopilot does not work sufficiently under these conditions.

3.d Wave disturbance

Simulating the system with wave disturbance and measurement noise turned on returns the plot shown in Figure 9.

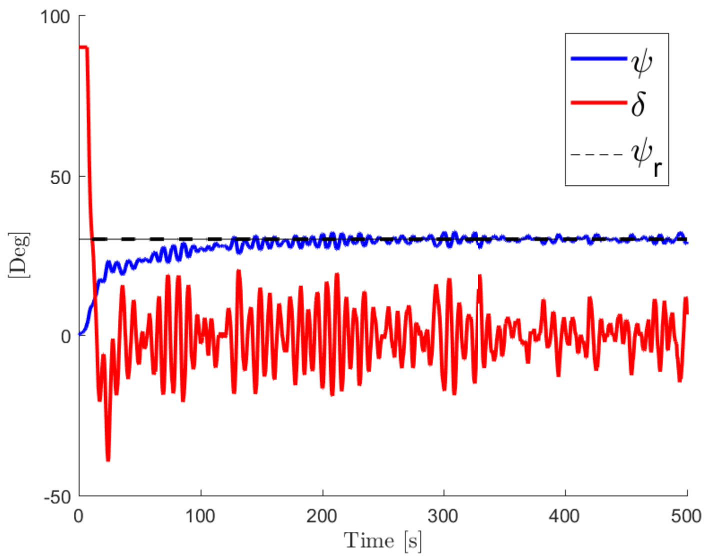

Figure 9: Autopilot with wave disturbance and measurement noise.

From Figure 9 we can immediately see that the autopilot is far from optimal under these conditions. The rudder angle never stabilizes and can vary up to 40 degrees in 5 seconds. Such quick and sizable changes will ruin the rudder over time. These rapid variations are caused by the controller trying to compensate for each and every wave, which is unwanted behaviour. However, the average heading eventually reaches the reference value, and proceeds to oscillate around it, with relatively small spikes.

Even though the reference value is reached, and somewhat held, this autopilot would not be acceptable for real life applications. The changes in the rudder angel are unrealistic and it would wear itself down extremely quickly.

4 Part IV - Observability

4.a State space model

Using the equations stated in (1) and (2), as well as the rest of the information given in the System Model section we can find the matrices **A**, **B**, **C** and **E**.

$$\mathbf{x} = \begin{bmatrix} \xi \\ \psi_w \\ \psi \\ r \\ b \end{bmatrix} \quad u = \delta \quad \mathbf{W} = \begin{bmatrix} w_w \\ w_b \end{bmatrix}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\xi} \\ \dot{\psi}_w \\ \dot{\psi} \\ \dot{r} \\ \dot{b} \end{bmatrix} = \begin{bmatrix} x_2 \\ -w_0^2 - 2\lambda w_0 x_2 + K_w W_1 \\ x_4 \\ -\frac{1}{T} + \frac{K(u-x_5)}{T} \\ W_2 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -w_0^2 & -2\lambda w_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}^T \quad \mathbf{E} = \begin{bmatrix} 0 & 0 \\ K_w & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.1}$$

4.b No disturbances

Without any disturbances, our model is reduced substantially. Our new state vector becomes:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} r \\ -\frac{r}{T} + \frac{K(\delta - b)}{T} \end{bmatrix}, \quad b = 0 \quad \Rightarrow \quad \dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -\frac{x_2}{T} + \frac{Ku}{T} \end{bmatrix}$$

Since there are no disturbances, **w** is removed from the system, meaning we no longer have a relevant **E** matrix. The **A**, **B** and **C** matrices are as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{K}{T} \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \tag{4.2}$$

To determine whether the system is observable or not we compute the observability matrix $\mathcal{O}$:

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

Using the MATLAB function `obsv(A,C)`, we compute the observability matrix to be the following:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3}$$

We observe that $\mathcal{O}$ has full rank, and conclude that the system without disturbances is observable.

4.c Current disturbance

For our model with current disturbances, we get the following state vector and matrices:

$$\mathbf{x} = \begin{bmatrix} \psi \\ r \\ b \end{bmatrix} \quad \Rightarrow \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\psi} \\ \dot{r} \\ \dot{b} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{x_2}{T} + \frac{K(u-x_3}{T} \\ w_b \end{bmatrix}$$
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{K}{T} \\ 0 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \quad \mathbf{E} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.4}$$

The observability matrix is computed as before:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \end{bmatrix} \tag{4.5}$$

This matrix has full rank, meaning our system with current disturbance is observable.

4.d Wave disturbance

Our system with wave disturbance and without current disturbance gives us the following states and matrices:

$$\mathbf{x} = \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \end{bmatrix} \quad \Rightarrow \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\xi} \\ \dot{\psi}_w \\ \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} x_2 \\ -w_0^2 x_1 + -2\lambda w_0 x_2 + K_w w_w \\ r \\ -\frac{r}{T} + \frac{K\delta}{T} \end{bmatrix}$$
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -w_0^2 & -2\lambda w_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}^T \quad \mathbf{E} = \begin{bmatrix} 0 \\ K_w \\ 0 \\ 0 \end{bmatrix} \tag{4.6}$$

Which leads to the observability matrix:

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -w_0^2 & -2\lambda w_0 & 0 & 1 \\ 2\lambda w_0^3 & w_0^2(4\lambda^2 - 1) & 0 & -\frac{1}{T} \\ -w_0^4(4\lambda^2 - 1) & 2\lambda w_0^3(1 - 4\lambda^2) & 0 & \frac{1}{T^2} \end{bmatrix} \tag{4.7}$$

This matrix also has full rank, meaning our system with wave disturbance is observable.

4.e Both current and wave disturbance

To determine whether the complete system is observable we calculate the observability matrix using values from **A** and **C** in (4.1), which result in the following:

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ -w_0^2 & -2\lambda w_0 & 0 & 1 & 0 \\ 2\lambda w_0^3 & w_0^2(4\lambda^2 - 1) & 0 & -\frac{1}{T} & -\frac{K}{T} \\ -w_0^4(4\lambda^2 - 1) & 2\lambda w_0^3(1 - 4\lambda^2) & 0 & \frac{1}{T^2} & \frac{K}{T^2} \\ 2\lambda w_0^5(1 - 4\lambda^2) & w_0^4(-16\lambda^4 + 8\lambda^2) & 0 & -\frac{1}{T^3} & -\frac{K}{T^3} \end{bmatrix} \tag{4.8}$$

The observability matrix for the entire system has full rank, meaning that our model is observable even with all disturbances present. This means that we can make use of estimators, which we will, in Part V - Discrete Kalman filter.

5 Part V - Discrete Kalman filter

5.a Discretization of continous model

In real world circumstances, measurements and calculations will include the use of computers, which operate using discrete values. The most common version of the Kalman filter is therefore the discrete Kalman filter. For our implementation of the filter, we need a model in discrete time. Thus far in the report and simulations, we have used a continuous time model, which will need to be discretized. To attain this, we can utilize Van Loans method described in [2]:

$$\exp\left(\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} T\right) = \begin{bmatrix}\bar{\mathbf{A}} & \bar{\mathbf{B}}\\ \mathbf{0} & \mathbb{I}\end{bmatrix} \tag{5.1}$$

Where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are the exact discretized $\mathbf{A}$ and $\mathbf{B}$ matrices, and T is the sample time for the discrete system. To use this approach with disturbances included we can rewrite (2) to:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}^*\mathbf{u}^* \tag{5.2}$$

With $\mathbf{B}^* = [\mathbf{B}\ \mathbf{E}]$ and $\mathbf{u}^* = [u\ \mathbf{w}^T]^T$. For the discrete model, $\bar{\mathbf{C}} = \mathbf{C}$, and T is 0.1s when the sampling frequency is set at 10 Hz. Using the matrix exponential function `expm()` in MATLAB and decomposing the resulting matrix as shown in (5.1) we arrive at the following set of matrices:

$$\bar{\mathbf{A}} = \begin{bmatrix} 0.9971 & 0.0932 & 0 & 0 & 0 \\ -0.0570 & 0.8659 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & -1.076\cdot10^{-5} \\ 0 & 0 & 0 & 0.9986 & -2.153\cdot10^{-4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \bar{\mathbf{B}} = \begin{bmatrix} 0 \\ 0 \\ 1.077\cdot10^{-5} \\ 2.153\cdot10^{-4} \\ 0 \end{bmatrix},$$
$$\bar{\mathbf{C}} = \begin{bmatrix} 0\\1\\1\\0\\0 \end{bmatrix}^T \quad \bar{\mathbf{E}} = \begin{bmatrix} 2.588\cdot10^{-5} & 0 \\ 5.055\cdot10^{-4} & 0 \\ 0 & -3.590\cdot10^{-7} \\ 0 & -1.0770\cdot10^{-5} \\ 0 & 0.1 \end{bmatrix} \tag{5.3}$$

The discrete time plant model becomes:

$$\mathbf{x}[k+1] = \bar{\mathbf{A}}\mathbf{x}[k] + \bar{\mathbf{B}}u[k] + \bar{\mathbf{E}}\bar{\mathbf{w}}[k], \quad y[k] = \bar{\mathrm{C}}\mathbf{x}[k] + \bar{v}[k] \tag{5.4}$$

5.b Estimating the variance of the measurement noise

To estimate the variance of the measurement white noise, we alter the simulink block diagram so that the disturbances are disabled and the rudder input δ is set to zero. This way, when we measure ψ, we essentially measure the noise itself, because we know that the ship is not moving (ψ remains zero) and waves are not affecting the output.

$$y = \psi + \psi_w + v$$
$$\Rightarrow y = v$$

Moreover, the scope data is exported to the MATLAB workspace, were the function `var()` is used to determine the variance.

$$R = \sigma^2 = 0.0020 \tag{5.5}$$

The Simulink diagram is displayed in Figure 18 in appendix A

5.c Constructing the Kalman filter

The following values for the process noise covariance $\mathbf{Q}$, the initial a priori estimate error covariance $\mathbf{P}_0^-$, and the initial a priori state estimate $\hat{\mathbf{x}}_0^-$ were given in the assigment [1]:

$$E\{\mathbf{w}\mathbf{w}^t\} = \bar{\mathbf{Q}} = \begin{bmatrix} 30 & 0 \\ 0 & 10^{-6} \end{bmatrix}$$

$$\mathbf{P}_0^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.013 & 0 & 0 & 0 \\ 0 & 0 & \pi^2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2.5 \cdot 10^{-3} \end{bmatrix}, \quad \hat{\mathbf{x}}_0^- = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.6}$$

The discrete measurement noise variance $\bar{R}$ is equal to the continuous noise variance divided by T.

$$\bar{R} = \frac{R}{T} = 0.02 \tag{5.7}$$

The following recursive procedure with $k = 0, 1, 2...$ is executed in the Kalman filter:

1. Compute the blending factor/Kalman gain:

$$\mathbf{L}[k] = \mathbf{P}^-[k]\bar{\mathbf{C}}^T(\bar{\mathbf{C}}\mathbf{P}^-[k]\bar{\mathbf{C}}^T + \bar{R})^{-1} \tag{5.8}$$

2. Update the estimate with the measurement to obtain the *a posteriori* estimate:

$$\hat{\mathbf{x}}[k] = \hat{\mathbf{x}}^-[k] + \mathbf{L}[k](y[k] - \bar{\mathbf{C}}\hat{\mathbf{x}}^-[k]) \tag{5.9}$$

3. Update the error covariance matrix:

$$\mathbf{P}[k] = (\mathbb{I} - \mathbf{L}[k]\bar{\mathbf{C}})\mathbf{P}^-[k](\mathbb{I} - \mathbf{L}[k]\bar{\mathbf{C}})^T + \mathbf{L}[k]\bar{R}\mathbf{L}[k]^T \tag{5.10}$$

4. Predict the state and error covariance to obtain the *a priori* estimates:

$$\hat{\mathbf{x}}^-[k+1] = \bar{\mathbf{A}}\hat{\mathbf{x}}[k] + \bar{\mathbf{B}}u[k] \tag{5.11}$$

$$\mathbf{P}^-[k+1] = \bar{\mathbf{A}}\mathbf{P}[k]\bar{\mathbf{A}}^T + \bar{\mathbf{Q}} \tag{5.12}$$

In Simulink, the Kalman filter is implemented using a MATLAB function block. Because the block does not hold the values of variables from one call to the next, the variables are declared as `persistent` so that they are stored in memory. Another precaution we need to make is to consider that the function block does not know if its the first or n'th time its been called, which can complicate initialization. Declaring an `init_flag` which is kept in memory can be set so that the function can keep track of whether or not it needs to initialize the Kalman filter.

The inputs from the filter are the measured compass angle ψ and the rudder angle δ, while the outputs are the filtered a posteriori estimates for compass angle and rudder bias, ψ_p and b_p. Because ψ and δ are continuous variables, we put zero-order holds on the inputs to the discrete filter. To avoid an algebraic loop caused by dependencies in the feedback loop, memory blocks are added to the outputs of the Kalman filter block. The full implementation of the Kalman filter in MATLAB is shown in appendix B. The Simulink block diagrams for the Kalman filtered system are displayed in Figures 19 and 20 in appendix A.

5.d Controlling the ship using feed forward from the estimated bias

In part 3.c, we saw ψ having a clear stationary error compared to the reference. This was caused by the rudder bias applied by the current, and the controller's inability to provide the necessary input to fully adjust for it. To illustrate this: When the measured compass angle is equal to the reference, the error is zero. Without the bias estimation, the PD-controller will as a result apply zero rudder angle. It is therefore clear that when the bias continuously affects δ, it forces the ship away from the reference compass angle, and the reference will be impossible to track.

Because the Kalman filter enables us to estimate the bias, the controller can apply the input needed to counteract the current disturbance. This can be seen in Figure 10, where the bias estimation makes the controller increase the rudder angle sooner compared to what it did without estimation. At the reference, the estimated bias is stable at 3°, so that even though the controller outputs zero rudder angle, the feed forward cancels out the current disturbance. Adding the bias feed forward consequently results in a drastic improvement in reference tracking, and the stationary error is eliminated. Figure 11 illustrates the performance differences between the feed forward assisted autopilot(ψ) and the regular PD-controlled autopilot(ψ_3).

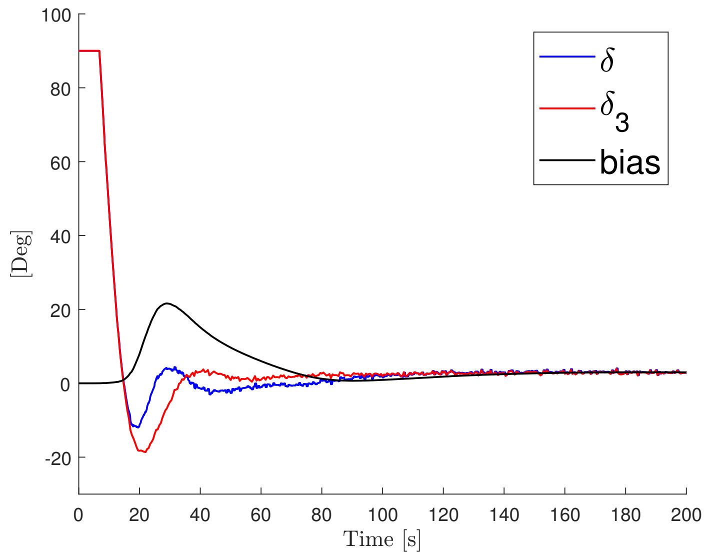

Figure 10: Estimated bias and comparison between rudder angles with(δ) and without(δ_3) feed forward from bias

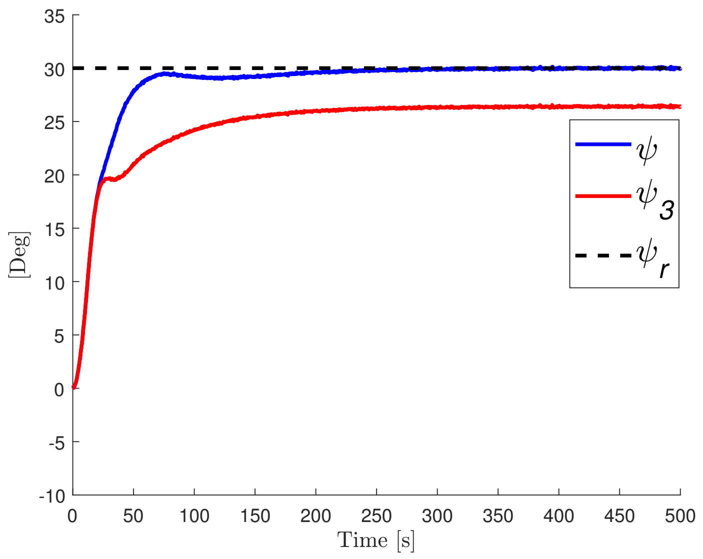

Figure 11: Reference angle and measured compass course with(ψ) and without(ψ_3) feed forward from estimated bias.

5.e Controlling the ship with wave disturbances

From section 3.d we learned that waves cause oscillations in the measured compass angle ψ. When these oscillating values are fed into the PD-controller, the input to the rudder angle fluctuates quite violently, with values varying up to $\pm 40°$ in under 5 seconds. Over extended periods of time, these oscillations will undoubtedly cause great harm to the rudder and its actuators and rotation joints. When we introduce the Kalman filter, the wave filtered ψ is instead used as the feedback component. The filter's ability to disregard the most common wave frequencies significantly reduces the heavy oscillations used as input to the controller. In return, the fast and large rotations of the rudder are greatly dampened. The comparison of rudder movement with(δ) and without(δ_3) wave filtering is shown in Figure 12.

If we study Figure 13, we can see that the measured ψ is similar to ψ in Figure 9 from subsection 3.d. In both instances the ship adjusts its course to the reference, but the course is affected by waves causing small oscillations. Naturally, the waves cannot be cancelled out physically, but the filtered compass angle ψ_p is a lot more gentle on the rudder, which in the real world would be very important for the longevity of vital ship components. Because of this, and the fact that the simulation in subsection 3.d did not include current disturbances, unavoidably leading to a stationary error, we confirm that the performance of the controller implemented with the Kalman filter is superior to the regular controller.

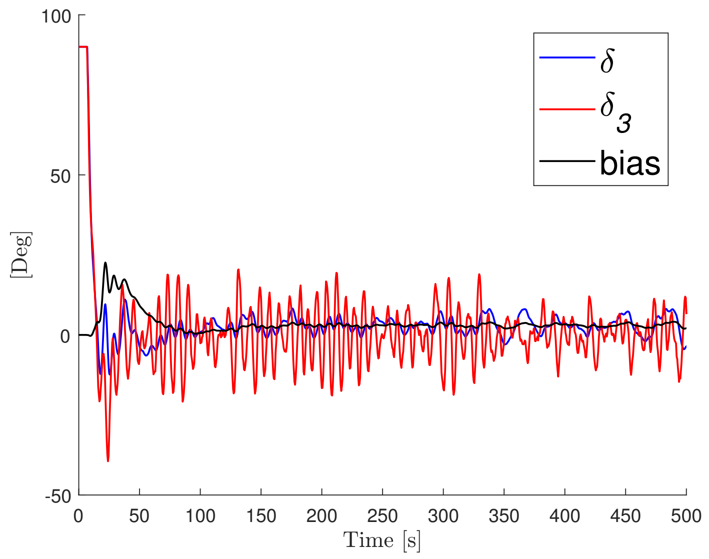

Figure 12: Estimated bias and rudder angle using filtered compass angle(δ), and from subsection 3.d(δ_3)

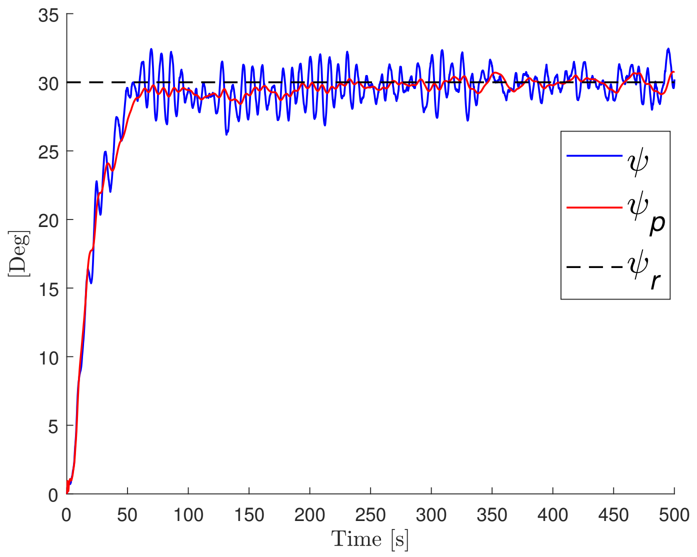

Figure 13: Measured(ψ) and filtered compass course(ψ_p)

To measure the impact the waves have on the measured compass direction, measurement noise and current disturbance in the model need to be disabled. The rudder input δ is also set to zero. The measured ψ is now equal to ψ_w because wave influence is the only parameter affecting the ship. Furthermore, the output of the Kalman filter is changed to ψ_w. The plot of measured ψ_w and filtered ψ_w is in Figure 14, and shows that the Kalman filter estimates the waves proficiently with only small deviations.

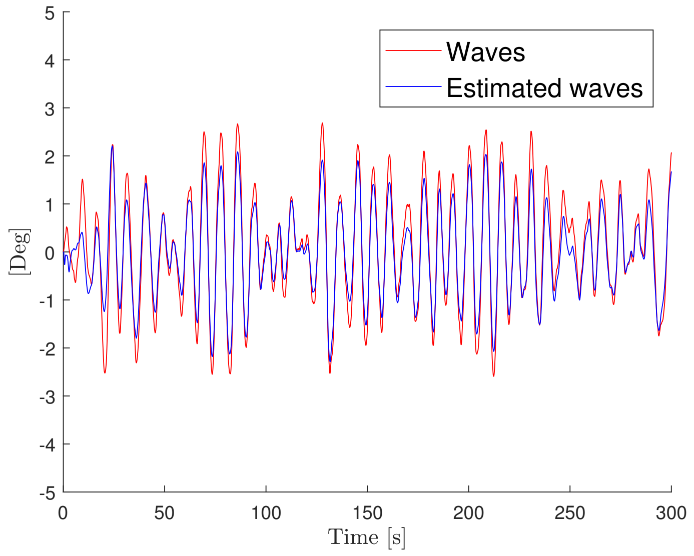

Figure 14: Comparison between actual and estimated waves.

Appendices

A Simulink models

1 Part 5.1.

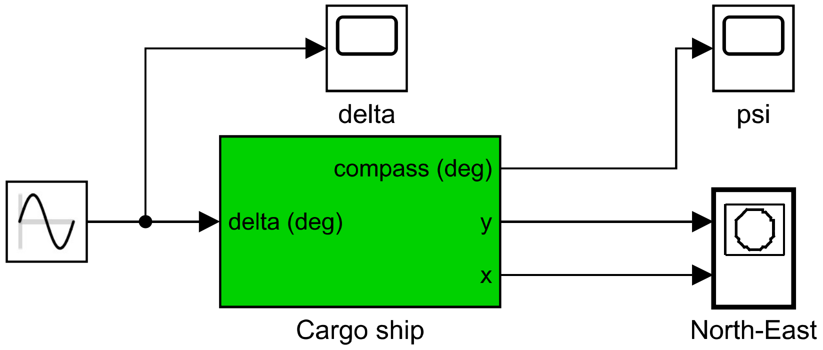

Figure 15: 5.1.b) All cargo ship disturbances disabled. 5.1.c) Noise and waves enabled.

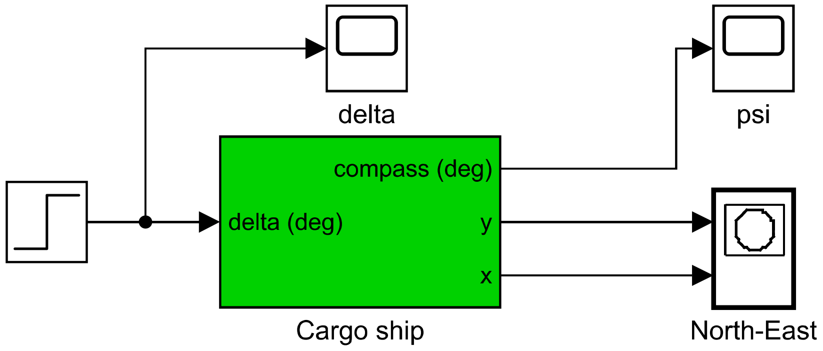

Figure 16: 5.1.d) First with noise and waves disabled, then enabled.

2 Part 5.3.

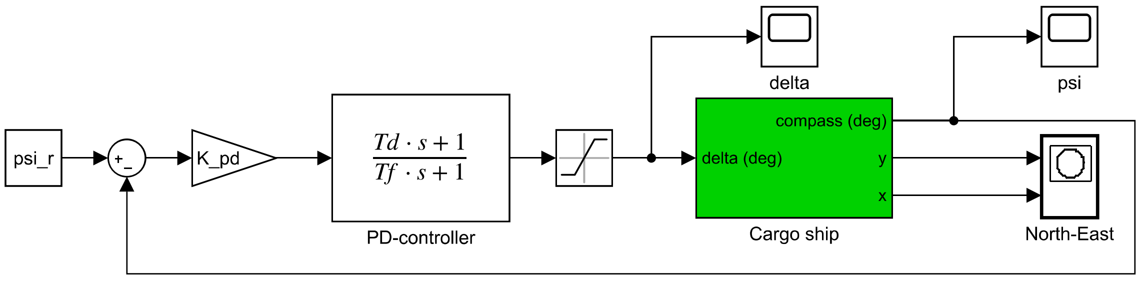

Figure 17: 5.3.b) Noise enabled. 5.3.c) Noise and current enabled. 5.3.d) Noise and waves enabled.

3 Part 5.5.

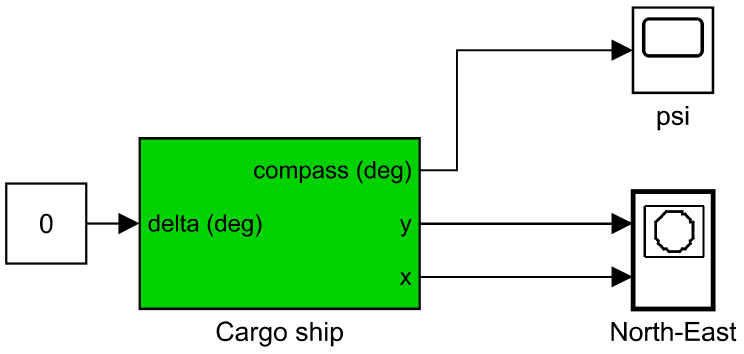

Figure 18: 5.5.b) Only noise enabled.

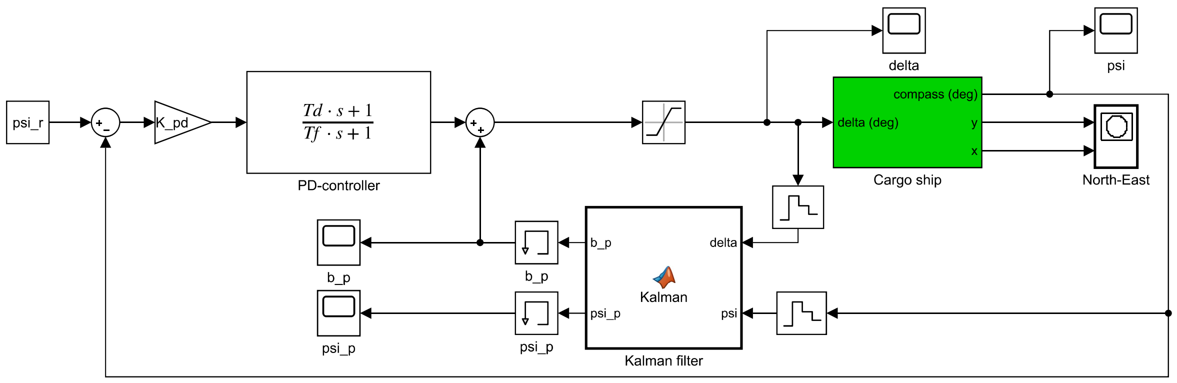

Figure 19: 5.5.d) Noise and current enabled.

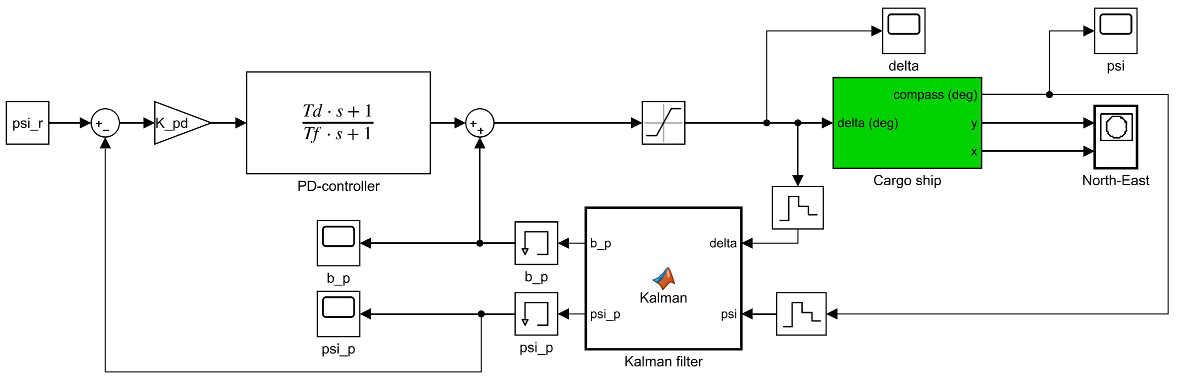

Figure 20: 5.5.e) Feedback from the filtered ψ. Noise, current and waves enabled.

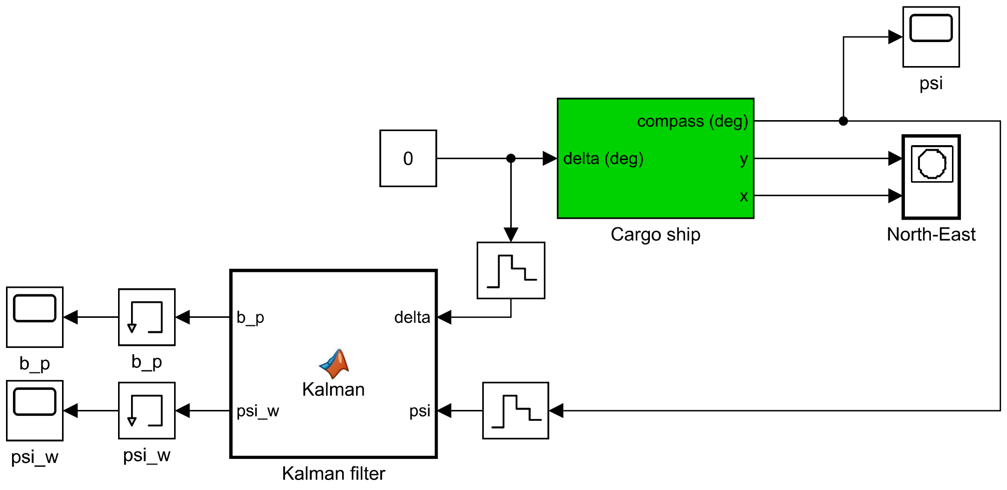

Figure 21: 5.5.e) Finding estimated wave influence. Only waves enabled.

B MATLAB scripts

1 5.1.b)

```
%%Task 1b%%
%Load data from simulations
load('compass1.mat');
load('compass2.mat');

H_len = length(t_1);
%Finding max/min values at steady state part of the response
H1_max = max(H_1(H_len/2:H_len));
H1_min = min(H_1(H_len/2:H_len));
H2_max = max(H_2(H_len/2:H_len));
H2_min = min(H_2(H_len/2:H_len));

A_1 = (H1_max-H1_min)/2;
A_2 = (H2_max-H2_min)/2;

figure
subplot(2,1,1);
plot(t_1,H_1);
title('Response to sine input with {\omega_{1}} = 0.005');
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('$H(j\omega_{1})$ [deg]','FontSize',12,'Interpreter','latex');
axis([0,10000, 0, 70]);

subplot(2,1,2);
plot(t_2,H_2);
title('Response to sine input with \omega_{2} = 0.05');
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('$H(j\omega_{2})$ [Deg]','FontSize',12,'Interpreter','latex');
axis([0,10000, 0, 5]);

%Results
K = 0.156;
T = 72.439;
```

2 5.1.c)

```
%%Task 1c%%
%Load data from simulations
load('compass1_D.mat');
load('compass2_D.mat');

H_len = length(t_1_D);
H_1_D_max = max(H_1_D(H_len/2:H_len));
H_1_D_min = min(H_1_D(H_len/2:H_len));

%Averaging max/min values over the steady state part of the response
for i = H_len/2:H_len-100
   H2_max = max(H_2_D(i-100:i+100));
   H2_min = min(H_2_D(i-100:i+100));
   A2(i-H_len/2+1) = (H2_max - H2_min)/2;
end

A1 = (H_1_D_max-H_1_D_min)/2;
A2 = mean(A2);

figure
subplot(2,1,1);
plot(t_1_D,H_1_D);
title('Response to sine input with {\omega_{1}} = 0.005 and disturbance');
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('$H(j\omega_{1})$ [deg]','FontSize',12,'Interpreter','latex');
axis([0,10000,0,70])

subplot(2,1,2);
plot(t_2_D,H_2_D);
```

```
title('Response to sine input with {\omega_{2}} = 0.05 and disturbance');
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('$H(j\omega_{2})$ [deg]','FontSize',12,'Interpreter','latex');

%Results
K = 0.163;
T = 14.503;
```

3 5.1.d)

```
%%Task 1d%%
%Load data from simulations
load('StepResponse.mat');

%Without disturbances
K = 0.156;
T = 72.439;
Model = tf(K, [T,1,0]);
figure
hold on
step(Model, 10000);
plot(t_SR_ship,SR_ship, 'r');
title('');
xlabel('Time','FontSize',12,'Interpreter','Latex');
ylabel('Response (deg)','FontSize',12,'Interpreter','latex');
lgd = legend('Model Response', 'Ship Response');
lgd.FontSize = 14;

%With disturbances
K = 0.163;
T = 14.503;
Model = tf(K, [T,1,0]);
figure
hold on
step(Model, 10000);
plot(t_SR_ship,SR_ship, 'r');
title('');
axis([0,10000,0,1600]);
xlabel('Time','FontSize',12,'Interpreter','Latex');
ylabel('Response (deg)','FontSize',12,'Interpreter','latex');
lgd = legend('Model Response', 'Ship Response');
lgd.FontSize = 14;
```

4 5.2.a)

```
%%Task 2a%%
load('wave.mat');

Fs = 10;
window_size = 4096;
[S_psi_w,f] = pwelch(psi_w(2,:).*pi/180,window_size,[],[],Fs);
S_psi_w = (S_psi_w./(2*pi));
w = 2*pi.*f;

figure
plot(w,S_psi_w,'LineWidth', 2);
xticks([0, pi/8 pi/4 3*pi/8 pi/2 5*pi/8]);
xticklabels({'0', '\pi/8' '\pi/4' '3\pi/8' '\pi/2' '5\pi/8'});
axis([0,5*pi/8,0,1.5*10^-3]);
xlabel('$\omega$ [rad/s]','FontSize',12,'Interpreter','Latex');
ylabel('$S_{\psi_\omega}(\omega)$ [rad]','FontSize',12,'Interpreter','latex');
```

5 5.2.c)

```
%%Task 2c%%
load('S_psi_w.mat');
[S_psi_w_max, max_index] = max(S_psi_w);
w_0 = w(max_index);
```

6 5.2.d)

```
%%Task 2d%%
%load S_psi_w, S_max,w and w_0
load('data2d.mat');

figure
hold on
plot(w,S_psi_w,'LineWidth', 2);
xticks([0, pi/8 pi/4 3*pi/8 pi/2 5*pi/8])
xticklabels({'0', '\pi/8' '\pi/4' '3\pi/8' '\pi/2' '5\pi/8'})
axis([0,pi/2,0,1.5*10^-3]);
xlabel('$\omega$ [rad/s]','FontSize',12,'Interpreter','Latex');
ylabel('[rad]','FontSize',12,'Interpreter','latex');

sigma = sqrt(S_psi_w_max);

for lambda = 0.03:0.03:0.12
    K_w = 2 * lambda * w_0 * sigma;
    P_psi_w_lambda = (w.*K_w).^2./(w.^4 + w_0^4 + 2*w_0^2*w.^2*(2*lambda^2 -1));
    plot(w, P_psi_w_lambda, ':', 'LineWidth', 2);
end
lgd = legend('S_{\psi\omega}(\omega)','\lambda = 0.03','\lambda = 0.06','\lambda = 0.09','↩
    \lambda = 0.12');
lgd.FontSize = 12;

%best fit
lambda = 0.09;
K_w = 2 * lambda * w_0 * sigma;
```

7 5.3.b)

```
%%Task 3b%%
%load data from simulations
load('data3b.mat');

T = 72.4385;
K = 0.1561;
Td = T;
Tf = 8.391;
K_pd = 0.836;
psi_ref = 30;

figure
hold on
plot(t_compass, compass,'b');
plot(t_delta,delta, 'r');
plot(t_psi_r, psi_r,'--k' );
axis([0,500,-30,100]);
lgd = legend('\psi','\delta','\psi_{r}');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');
```

8 5.3.c)

```
%%Task 3c%%
%load data from simulations, psi_ref,K,T,Td,Tf and K_pd
load('data3c.mat');

figure
hold on
plot(t_compass,compass, 'b');
plot(t_delta,delta, 'r');
plot(t_psi_r, psi_r,'--k');
axis([0,500,-30,100]);
lgd = legend('\psi','\delta','\psi_{r}');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');
```

9 5.3.d)

```
%%Task 3d%%
%load data from simulations, psi_ref,K,T,Td,Tf and K_pd
load('data3d.mat');

figure
hold on
plot(t_compass,compass, 'b');
plot(t_delta,delta, 'r');
plot(t_psi_r, psi_r,'--k');
axis([0,500,-50,100]);
lgd = legend('\psi','\delta','\psi_{r}');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');
```

10 5.5.a)

```
%%Task 5a%%
%load constants K,K_w,lambda,T,w_0
load('data5a.mat');

Fs = 10;
Ts = 1/Fs;
A = [0, 1, 0, 0, 0;-w_0^2, -2*lambda*w_0,0,0,0; 0,0,0,1,0; 0,0,0,-1/T,-K/T; 0,0,0,0,0;];
B = [0;0;0;K/T;0];
E = [0,0;K_w,0;0,0;0,0;0,1];
BE = [B,E];
C = [0,1,1,0,0];
temp = [A, BE];
M = zeros(8,8);
M(1:5, 1:8) = temp;

N = expm(M*Ts);
A_d = N(1:5, 1:5);
B_d = N(1:5,6);
E_d = N(1:5, 7:8);
C_d = C;
D_d = 0;
```

11 5.5.b)

```
%%Task 5b%%
%Load data from simulation
load('measurement.mat');
variance = var(measurement);
```

12 5.5.c)

```
%%Task 5c%%
%load all variables from 5a and 5b
load('constants.mat');
P_0 = [1,0,0,0,0;0,0.013,0,0,0;0,0,pi^2,0,0;0,0,0,1,0;0,0,0,0,0.0025];
x_0 = [0;0;0;0;0];
Q_d = [30,0;0,10e-6];
R_d = variance/Ts;
variables = {A_d,B_d,C_d,E_d,Q,R,x_0,P_0};
```

13 5.5.d)

```
%%Task 5d%%
%load data from simulations
load('data5d.mat');

figure
hold on
plot(t_psi,psi, 'b');
plot(t_psi_3,psi_3, 'r');
plot(t_psi_r,psi_r, '--k');
axis([0,500,-10,35]);
lgd = legend('\psi','\psi_{\it3}','\psi_{\itr}');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');


figure
hold on
plot(t_delta,delta, 'b');
plot(t_delta_3,delta_3, 'r');
plot(t_bias,bias, 'k');
axis([0,200,-30,100]);
lgd = legend('\delta','\delta_{3}','bias');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');
```

```
function [b_p, psi_p]  = Kalman(delta, psi)

persistent init_flag A B C E Q R x_ P_
I = eye(5);
if (isempty(init_flag))
    init_flag = 1;
    %load all the persitent values
    vars = load('variables.mat');
    A = vars.A_d; B = vars.B_d; C = vars.C_d; E = vars.E_d; Q = vars.Q_d; R = vars.R_d; x_↩
         = vars.x_0; P_ = vars.P_0;
end

L = (P_*C')/((C*P_*C'+R));

x = x_+L*(psi-C*x_);
P = (I-L*C)*P_*(I-L*C)' + L*R*L';
```

```
x_ = A*x+B*delta;
P_ = A*P*A'+E*Q*E';
psi_p = x(3); b_p = x(5);
```

14 5.5.e)

```
%%Task 5e%%
%load data from simulations psi_ref,K,T,Td,Tf and K_pd
load('data5e.mat');

figure
hold on
plot(t_psi,psi, 'b');
plot(t_psi_p,psi_p, 'r');
plot(t_psi_r,psi_r, '--k');
axis([0,500,0,35]);
lgd = legend('\psi','\psi_{\itp}','\psi_{\itr}');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');


figure
hold on
plot(t_delta,delta, 'b');
plot(t_delta_3,delta_3, 'r');
plot(t_bias,bias, 'k');
axis([0,500,-50,100]);
lgd = legend('\delta','\delta_{\it3}','bias');
lgd.FontSize = 18;
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');
```

15 5.5.e waves)

```
%%Task 5e_waves%%
%load data from simulations, psi_ref,K,T,Td,Tf and K_pd
load('data5e2.mat');

figure
hold on
plot(t_psi_w,psi_w,'r');
plot(t_psi_p,psi_p,'b');
lgd = legend('Waves','Estimated waves');
lgd.FontSize = 14;
axis([0,300,-5,5]);
xlabel('Time [s]','FontSize',12,'Interpreter','Latex');
ylabel('[Deg]','FontSize',12,'Interpreter','latex');
```

```
function [b_p, psi_w]  = Kalman(delta, psi)

persistent init_flag A B C E Q R x_ P_
I = eye(5);
if (isempty(init_flag))
    init_flag = 1;
    %load all persitent variables
    vars = load('variables.mat');
    A = vars.A_d; B = vars.B_d; C = vars.C_d; E = vars.E_d; Q = vars.Q_d; R = vars.R_d; x_↩
         = vars.x_0; P_ = vars.P_0;
end

L = (P_*C')/((C*P_*C'+R));

x = x_+L*(psi-C*x_);
P = (I-L*C)*P_*(I-L*C)' + L*R*L';
```

```
x_ = A*x+B*delta;
P_ = A*P*A'+E*Q*E';
psi_w = x(2); b_p = x(5);
```

References

[1] Kristoffer Gryte, *TTK4115 - Discrete Kalman Filter Applied to a Ship Autopilot*, NTNU, Version 1.8, October 2016.

[2] C.F. Van Loan, *Computing integrals involving the matrix exponential*, IEEE transactions on automatic control, Vol 23, 3rd issue, 1978.

[3] Morten D. Pedersen, *TTK4115 - Lecture K*, NTNU, 2017.