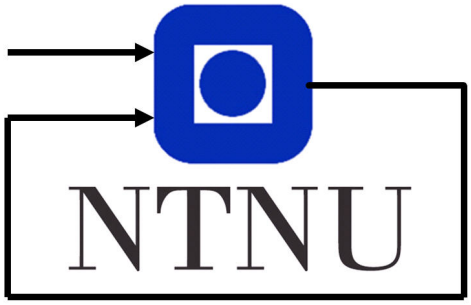
Boat lab report
TTK4115

Group 53
Anders Granberg Drønnen 768686
Bendik Steinkjer Standal 757944
Filip Gornitzka Abelson 768677

November 2017



Department of Engineering Cybernetics

Contents

Introduction

This report contains the methods, results and discussions for the boat lab project in the subject TTK4115 Linear systems theory. Our goal is to implement an autopilot for a cargo ship, which can ultimately try to neutralize disturbance and noise effects on the desired heading. This is achieved by using a discrete Kalman filter. The model of the ship is briefly described mathematically below.

Model

$$\dot{\xi_w} = \psi_w \tag{0.1a}$$
$$\dot{\psi_w} = -\omega_0^2 \xi_w - 2\lambda\omega_0\psi_w + K_w w_w \tag{0.1b}$$
$$\dot{\psi} = r \tag{0.1c}$$
$$\dot{r} = -\frac{1}{T}r + \frac{K}{T}(\delta - b) \tag{0.1d}$$
$$\dot{b} = w_b \tag{0.1e}$$
$$y = \psi + \psi_w + v \tag{0.1f}$$

We can write the system in state-space form.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u + \mathbf{E}\mathbf{w},\ y = \mathbf{C}\mathbf{x} + v \tag{0.2}$$

Where $\mathbf{x}$, u and $\mathbf{w}$ are defined as:

$$\mathbf{x} = \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \\ b \end{bmatrix},\ u = \delta \text{ and } \mathbf{w} = \begin{bmatrix} w_w \\ w_b \end{bmatrix} \tag{0.3}$$

1 Part I - Identification of boat parameters

1.1 Problem a - Transfer function from δ to ψ

If we assume no disturbances, we can set $b = 0$. Then, if we combine equation (0.1c) and equation (0.1d), we can find the transfer function $H(s)$ from δ to ψ.

$$\begin{aligned} \dot{\psi} = r \text{ and } \dot{r} &= -\frac{1}{T}r + \frac{K}{T}(\delta - b) \\ \implies \ddot{\psi} &= \frac{1}{T}\dot{\psi} + \frac{K}{T}\delta \end{aligned} \tag{1.1}$$

Now we will take the Laplace transform of equation (1.1), with the initial conditions of ψ equal to zero. That is, $\dot{\psi}(0) = \psi(0) = 0$.

$$\begin{aligned} \mathcal{L}\left\{\ddot{\psi} = \frac{1}{T}\dot{\psi} + \frac{K}{T}\delta\right\} \\ s^2\psi(s) - s\psi(0) - \dot{\psi}(0) = -\frac{1}{T}(s\psi(s) - \psi(0)) + \frac{K}{T}\delta(s) \\ \psi(s)(s^2 + \frac{1}{T}s) = \frac{K}{T}\delta(s) \\ \implies H(s) = \frac{\psi(s)}{\delta(s)} = \frac{\frac{K}{T}}{s^2 + \frac{1}{T}s} = \frac{K}{s(Ts+1)} \end{aligned} \tag{1.2}$$

1.2 Problem b - Identification of boat parameters

We apply two sine inputs with amplitude 1, but with different frequencies, on the transfer function. $\omega_1 = 0.005$ and $\omega_2 = 0.05$. By examining the magnitude responses $|H(j\omega_1)|$ and $|H(j\omega_2)|$, seen in fig. 1, we obtain a system of equations which we can solve for the unknown boat parameters T and K. Using `MATLAB` command `vpasolve(eqns)` we can solve the system of equations, which yields the following result.

$$K = 0.1561 \text{ and } T = 72.4347$$

The function `vpasolve(eqns)` numerically solves the system of equations `eqns` for variables determined by a `sym` variable. The implementation of this is shown in appendix A.1

1.3 Problem c - Identification of boat parameters, with disturbances

With wave disturbance and measurement noise turned on, we repeat what we did in problem 1b. Due to the inaccuracies caused by the noise and disturbance we cannot obtain reasonable values for the extreme values of the magnitude responses. This can be clearly seen in fig. 2. Therefore, it does

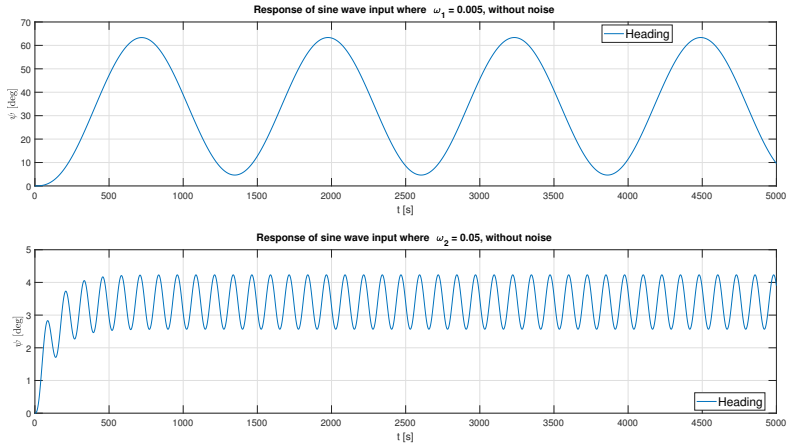


Figure 1: Magnitude response of $H(s)$ for $\omega_1 = 0.005$ and $\omega_2 = 0.05$ in calm conditions

not make sense to calculate the values of the parameters T and K. This could theoretically be solved by using approximations or mean of amplitude values and ignoring the "worst" spikes, e.g. filtering the measurement. However, this is not asked for in this problem.

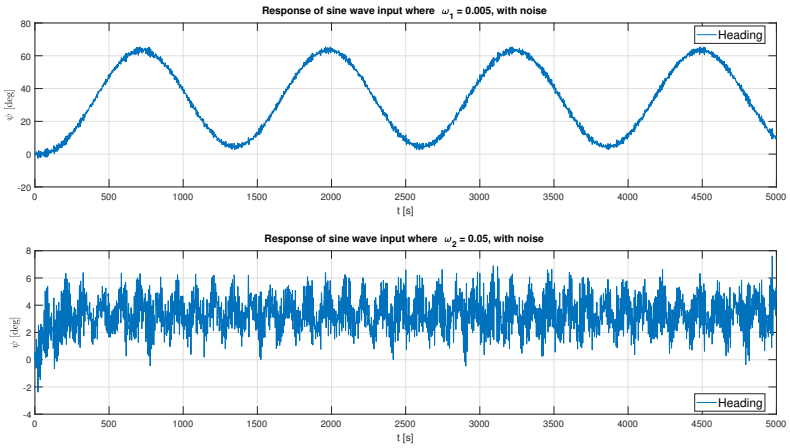


Figure 2: Magnitude response of $H(s)$ for $\omega_1 = 0.005$ and $\omega_2 = 0.05$ with disturbances.

1.4 Problem d - Step input to rudder angle

Now we are going to put a step input of 1 degree to the rudder at $t = 0$, and then compare the step response of the ship and the step response of the

model. As we can see from figure 3, the heading of the model deviates from the heading of the ship as time increases. Because it is a fairly small deviation over a long period of time, the model is good enough for our purpose.

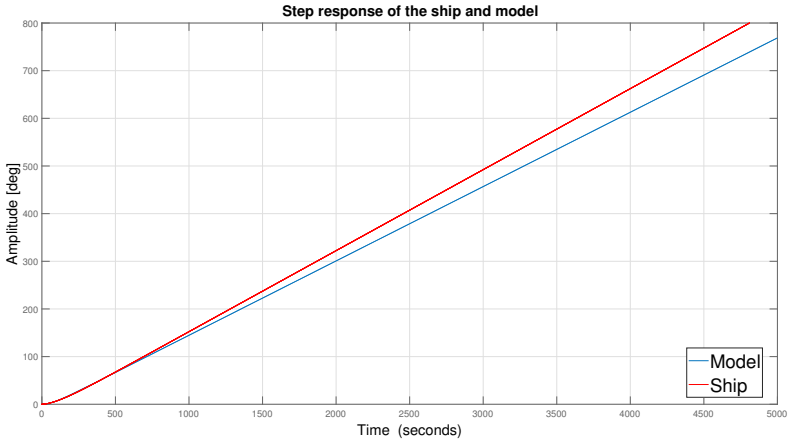


Figure 3: Step response of model and ship

2 Part II - Identification of wave spectrum model

2.1 Problem a - Power Spectral Density function

In order to see how the waves influence the compass measurement, we aim to find an estimate of the Power Spectral Density function of ψ_ω, $S_{\psi_\omega}(\omega)$. The PSD describes how power of a signal is distributed over frequency. In other words, it shows at which frequencies variations of power are strong and at which frequencies variations are weak. In this situation, the PSD will show how waves with given frequencies affects the cargo ship with different magnitudes.

To find the estimate of the PSD, we used the `MATLAB` function `[pxx,f] = pwelch(x,window,noverlap,nfft,fs)`, which returns the two-sided Welch PSD estimates at the frequencies specified in the vector `nfft`. In the function, `x` is the wave influence on the compass measurement ψ_ω, `window` is 4096 (from assignment [1]), and `fs` is the sampling frequency of 10 Hz.

ψ_ω is given in degrees in the file, so we converted the values to radians. Since the input `fs` is given in Hz, the units of the outputs `pxx` and `f` are power per Hz and Hz, respectively. We used the scaling factors $\frac{1}{2\pi}$ and 2π to convert them to the required units $\frac{\text{power}\cdot\text{s}}{\text{rad}}$ and $\frac{\text{rad}}{\text{s}}$, respectively. The plot of $S_{\psi_\omega}(\omega)$ can be seen in fig. 4, and the `MATLAB` script for this problem can be found in appendix A.2.

2.2 Problem b - Analytic expressions

We will combine equation (0.1a) and equation (0.1b) to find the transfer function from w_w to ψ_w.

$$\begin{aligned} \dot{\xi_w} = \psi_w \text{ and } \dot{\psi_w} &= -\omega_0^2\xi_w - 2\lambda\omega_0\psi_w + K_w w_w \\ \implies \dot{\psi_w} &= -\omega_0^2 \int \psi_w - 2\lambda\omega_0\psi_w + K_w w_w \end{aligned} \tag{2.1}$$

To find the transfer function, $H(s)$, we must take the Laplace transform of equation (2.1). The initial conditions of ψ_w are all equal to zero. That is,

$\dot{\psi_w}(0) = \psi_w(0) = 0.$

$$\mathcal{L}\left\{\dot{\psi_w} = -\omega_0^2 \int \psi_w - 2\lambda\omega_0\psi_w + K_w w_w\right\}$$

$$s\psi_w(s) - \psi_w(0) = -\omega_0^2 \frac{\psi_w(s)}{s} - 2\lambda\omega\psi_w(s) + K_w w_w(s)$$

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

$$\implies H(s) = \frac{\psi_w(s)}{w_w(s)} = \frac{K_w}{s + 2\lambda\omega_0 + \frac{\omega_0^2}{s}} = \frac{K_w s}{s^2 + 2\lambda\omega_0 s + \omega_0} \tag{2.2}$$

Now we must find an analytic expression for the Power Spectral Density function of ψ_w, denoted by $P_{\psi_w}(w)$. We will do this by using the transfer function, equation (2.2).

$$P_{\psi_w}(j\omega) = |H(j\omega)|^2 P_{w_w} \tag{2.3}$$

Because w_w is white noise, the PSD (power spectral density), P_{w_w}, is simply equal to the variance of the white noise, $\sigma_{w_w}^2 = 1$. Now we can simplify equation (2.3) further.

$$P_{\psi_w}(j\omega) = |H(j\omega)|^2\sigma_{w_w}^2 = |H(j\omega)|^2 = H(j\omega)H(-j\omega)$$
$$= \frac{jK_w\omega}{(j\omega)^2 + j2\lambda\omega_0\omega + \omega_0}\frac{-jK_w\omega}{(-j\omega)^2 - j2\lambda\omega_0\omega + \omega_0}$$

By doing the necessary calculations and simplifications, we get the following expression for P_{ψ_w}:

$$P_{\psi_w}(j\omega) = \frac{K_w^2\omega^2}{\omega_0^4 + \omega^4 + 2\omega_0^2\omega^2(2\lambda^2 - 1)} \overset{\mathbb{R}}{=} P_{\psi_w}(\omega) \tag{2.4}$$

2.3 Problem c - The resonance frequency

The resonance frequency ω_0 is the wave frequency which has the greatest impact on the heading of the ship. Thus, ω_0 is the maximum value of the estimated Power Spectral Density function $S_{\psi_\omega}(\omega)$. The value can be found simply by observing the plot in fig. 4, or by using the following code in `MATLAB`:

```
[max_pxx, indx] = max(pxx);
omega_0 = omega(indx);
```

We found $\omega_0 = 0.7823$.

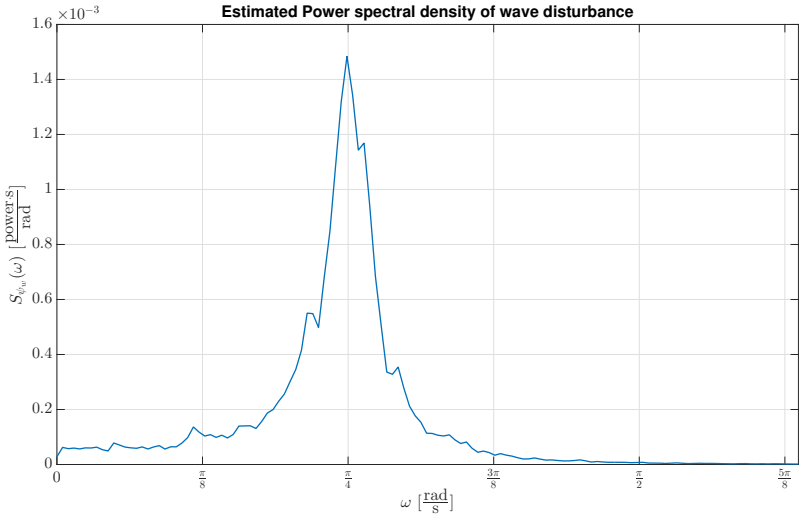


Figure 4: The estimate of the Power Spectral Density function of ψ_ω, $S_{\psi_\omega}(\omega)$

2.4 Problem d - The damping factor

We need to identify the damping factor λ, in order to complete our model for the wave response. We defined $K_w = 2\lambda\omega_0\sigma$, where σ is the square root of the maximum value of $S_{\psi_\omega}(\omega)$. We plotted $P_{\psi_\omega}(\omega)$ for different values of λ, and compared it to the plot of $S_{\psi_\omega}(\omega)$. This can be seen in fig. 5. By trial and error, we found $\lambda = 0.08$ to give a satisfying result. The `MATLAB` script for this problem can be found in appendix A.2.

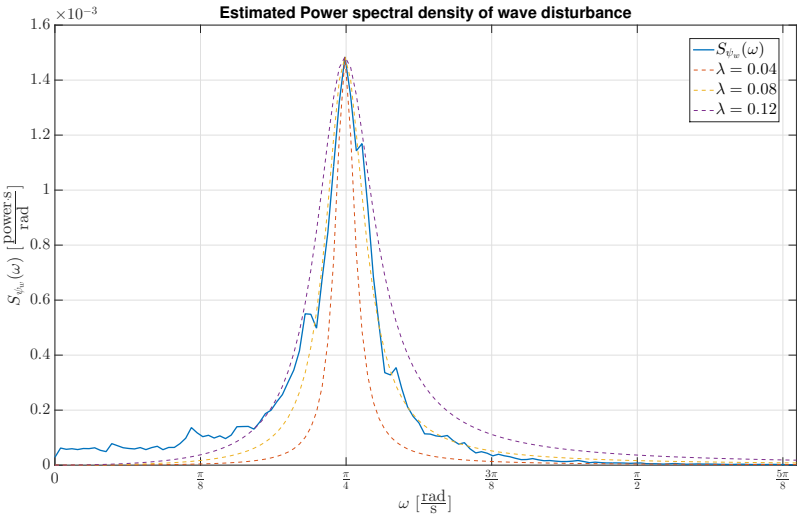


Figure 5: $S_{\psi_\omega}(\omega)$, and $P_{\psi_\omega}(\omega)$ with different values for λ

3 Part III - Control system design

In this section we want to design an autopilot for the ship, such that the ship is able to follow a desired course angle ψ_r. We have used $\psi_r = 30^\circ$ in all simulations done in this part.

3.1 Problem a - PD controller

We want to design a PD controller,

$$H_{pd}(s) = K_{pd}\frac{1 + T_d s}{1 + T_f s} \tag{3.1}$$

based on the transfer function from δ to ψ without disturbances from equation (1.2),

$$H_{ship} = \frac{K}{s(Ts + 1)} \tag{3.2}$$

We get the open loop transfer function by multiplying equation (3.1) and equation (3.2). We want to choose the derivative time constant T_d, such that it cancels out the transfer function time constant T. By choosing $T_d = T$ we end up with the following open loop transfer function.

$$H_0(s) = H_{pd}(s) \cdot H_{ship}(s) = \frac{K_{pd}(1 + Ts)}{1 + T_f s} \cdot \frac{K}{s(Ts + 1)} = \frac{K_{pd}K}{s(1 + T_f s)} \tag{3.3}$$

We will start by determining the value of T_f. The definition of phase margin is $\psi_{\text{phase}} = \angle H_0(j\omega_c) - (-180^\circ)$. As the assignment text states, the value of ψ_{phase} should be 50° and ω_c should be $0.1\ \frac{\text{rad}}{\text{s}}$ approximately. We solve the equation for T_f.

$$\begin{aligned}
\psi_{\text{phase}} &= \angle H_0(j\omega_c) - (-180^\circ) = 50^\circ \\
\angle H_0(j\omega_c) &= \angle \frac{K_{pd}K}{j\omega_c(1 + T_f j\omega_c)} = -130^\circ \\
\underbrace{\angle K_{pd}K}_{=0} - \angle(j\omega_c - \omega_c^2 T_f) &= -\arctan\left(\frac{\omega_c}{-\omega_c^2 T_f}\right) = -130^\circ \\
\implies T_f &= -\frac{1}{\omega_c \tan(130^\circ)} = 8.39
\end{aligned} \tag{3.4}$$

We know that the magnitude of the open loop transfer function is equal to 0 dB at ω_c. 0 dB is equal to 1 in absolute units. We can solve $|H_0(j\omega_c)| = 1$

to find the correct value for K_{pd}.

$$|H_0(j\omega_c)| = \frac{|K_{pd}K|}{|j\omega_c + T_f(j\omega_c)^2|} = 1$$

$$\frac{K_{pd}K}{|j\omega_c - T_f\omega_c^2|} = \frac{K_{pd}K}{\sqrt{\omega_c^2 + (-T_f\omega_c^2)^2}} = 1$$

$$\implies K_{pd} = \frac{\sqrt{\omega_c^2 + T_f^2\omega_c^4}}{K} = 0.836 \tag{3.5}$$

This gives the following bode plot for the open loop system:

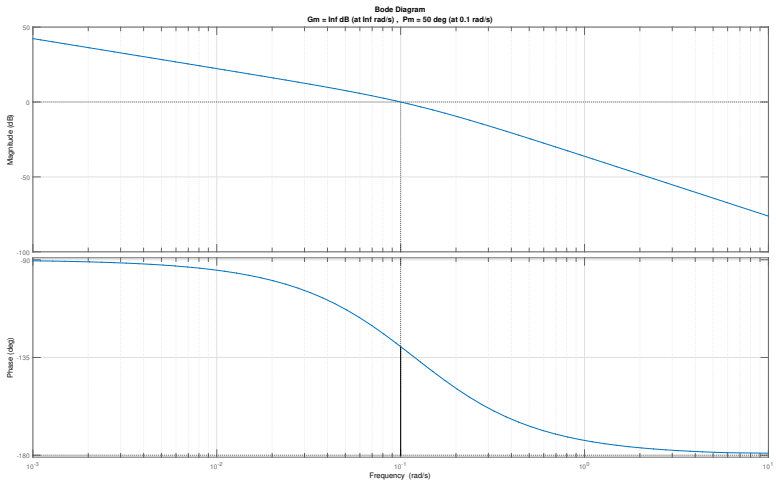


Figure 6: Bode plot for the open loop system

3.2 Problem b - Simulation with measurement noise

In order to simulate the autopilot system using the PD controller found in the previous problem, we needed to modify the `SIMULINK` model. We added the new transfer function, and set the course angle reference $\psi_r = 30°$. Because the ship model in the simulation only holds for small deviations in compass value ψ, we added a saturation block to keep $|\psi|$ inside a value of 35°. The autopilot system can be found in fig. 16 in appendix B.

We simulated the system with measurement noise only, the result can be seen in figure 7. We observe that the heading reaches the reference course angle at around 300 seconds, which is an acceptable result for such a large ship. We conclude that the autopilot works well in such smooth weather conditions.

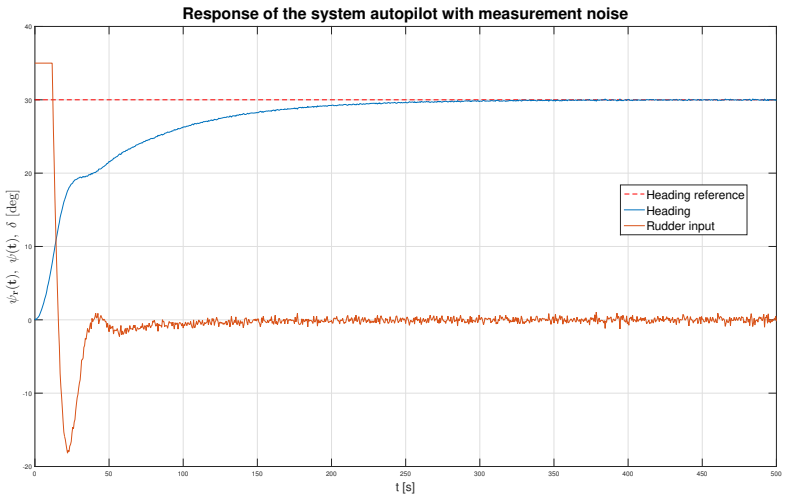


Figure 7: Simulation of the autopilot, with measurement noise

3.3 Problem c - Simulation with measurement noise and current disturbance

We simulated the system again, this time with current disturbance in addition to measurement noise. As mentioned in section 3.2, we need to keep compass value ψ inside a boundary of $\pm 35°$. This satisfies the assumption that the only effect of the current is a rudder angle bias. The system simulation can be seen in fig. 8.

The response of the system looks a lot like the one in section 3.2, but we observe that the heading now has a steady state error of approximately 3.5°. This is because of the current rudder angle bias, which prevents the rudder in reaching the correct value to eliminate the error. This could easily be fixed by adding an integral effect to the controller. In conclusion, the PD controller does not behave satisfactorily when the current disturbance is present.

3.4 Problem d - Simulation with measurement noise and wave disturbance

We simulated the system one last time, now with measurement noise and wave disturbance. The simulation results are shown in fig. 9. This time, the system did reach the reference value. However, due to the wave disturbance, the rudder continuously tried to counteract the effect of the waves. This causes the heading to fluctuate around the desired value. Yet again, the PD-controller is not good enough when disturbance is present. It is obvious

that the rapid fluctuations of the rudder angle does not benefit the system, and will likely break the rudder components in a relatively short amount of time. This is one of the reasons why we want to design a Kalman filter for this autopilot, as we shall see in section 5.

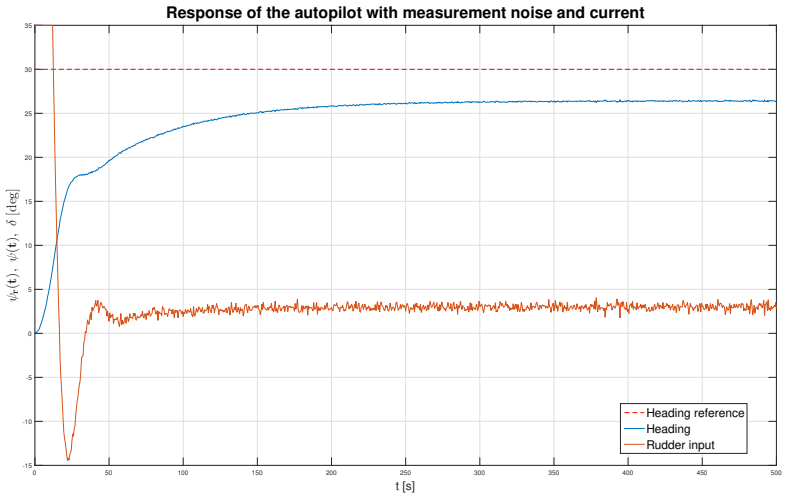


Figure 8: Simulation of the autopilot, with measurement noise and current disturbance

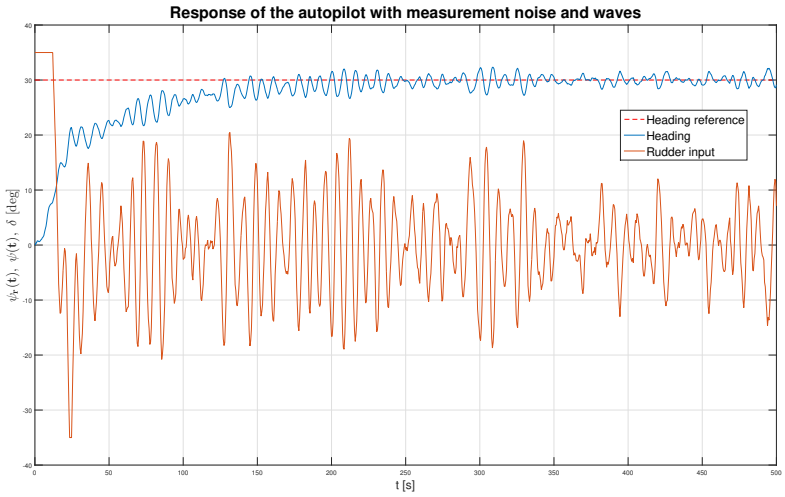


Figure 9: Simulation of the autopilot, with measurement noise and wave disturbance

4 Part IV - Observability

4.1 Problem a - Calculation of state-space matrices

We can find the matrices **A B**, **C** and **E** by taking the derivative of the state vector **x** and solving for the state-space model, as shown in equation (0.2).

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\xi_w} \\ \dot{\psi_w} \\ \dot{\psi} \\ \dot{r} \\ \dot{b} \end{bmatrix} = \begin{bmatrix} \psi_w \\ -\omega_0^2 \xi_w - 2\lambda\omega_0 \psi_w + K_w w_w \\ r \\ -\frac{1}{T} r + \frac{K}{T}(\delta - b) \\ w_b \end{bmatrix}$$

$$\implies \dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix}}_{\mathbf{B}} \delta + \underbrace{\begin{bmatrix} 0 & 0 \\ K_w & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{E}} \mathbf{w} \quad (4.1)$$

$$\implies y = \psi + \psi_w + v = \mathbf{C}\mathbf{x} + v = \underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{C}} \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \\ b \end{bmatrix} + v \quad (4.2)$$

4.2 Problem b - Observability without disturbances

We can determine if a system is observable by looking at the rank of the observability matrix $\mathcal{O}$. If the rank of $\mathcal{O}$ is equal to dimension of the matrix **A**, we say that $\mathcal{O}$ has full rank, and the system is observable. The observability matrix is defined as:

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \quad (4.3)$$

Firstly, we determine if the system is observable without any disturbances. This means that we can rule out the following states: ξ_w, ψ_w and b. We get a new state vector $\mathbf{x} = \begin{bmatrix} \psi & r \end{bmatrix}^T$. To calculate the rank of $\mathcal{O}$, we only need to adjust the matrices **A** and **C**. Using the matrices in equations (4.1) and (4.2), we get:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

By using the MATLAB function `rank(obsv(A, C))`, we find that the rank of the observability matrix is 2, which means that it has full rank. Thus, the system is observable without disturbances.

4.3 Problem c - Observability with current disturbance

We want to see the effect of current disturbance through the heading bias b, so the new state vector is $\mathbf{x} = \begin{bmatrix}\psi & r & b\end{bmatrix}^T$. Using the same approach as in problem b gives us:

$$\mathbf{A} = \begin{bmatrix}0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0\end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$$

By using `rank(obsv(A, C))`, we find that that the rank of $\mathcal{O}$ is 3 (full rank). The system is observable with the current disturbance.

4.4 Problem d - Observability with wave disturbance

Now we will look at the system with only wave disturbance. This gives us the state vector $\mathbf{x} = \begin{bmatrix}\xi_w & \psi_w & \psi & r\end{bmatrix}^T$. $\mathbf{A}$ and $\mathbf{C}$ are now found to be:

$$\mathbf{A} = \begin{bmatrix}0 & 1 & 0 & 0 \\ -\omega_0^2 & 2\lambda\omega_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T}\end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix}0 & 1 & 1 & 0\end{bmatrix}$$

Using `rank(obsv(A, C))` we find that the rank of the $\mathcal{O}$ is 4 (full rank). The system is observable with wave disturbances.

4.5 Problem e - Observability with both disturbances

Finally, we will check the observability of the system with both current and wave disturbances. We now use the state vector $\mathbf{x} = \begin{bmatrix}\xi_w & \psi_w & \psi & r & b\end{bmatrix}^T$ and matrices $\mathbf{A}$ and $\mathbf{C}$ from section 4.1.

$$\mathbf{A} = \begin{bmatrix}0 & 1 & 0 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0\end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix}0 & 1 & 1 & 0 & 0\end{bmatrix}$$

Using `rank(obsv(A, C))` we find that the rank of $\mathcal{O}$ is 5 (full rank). Thus, the system is observable.

The system proved to be observable in all cases. This means that for a

measured output we can determine the initial states, as well as the behavior of the system. This makes use of estimators possible. We'll take advantage of this in the next part of the assignment, when we try to improve our autopilot.

5 Part V - Discrete Kalman Filter

In this section we want to implement a discrete Kalman filter, in order to estimate the bias b, the heading ψ, and the high-frequency wave induced motion on the heading ψ_w.

5.1 Problem a - Discretization

We discretized the model from problem 5.4 part a), using exact discretization. This can easily be done with the `MATLAB` function `c2d`, which converts a model from continuous time to discrete time. We used a sample frequency of 10 Hz, which gives a sampling time of 0.1 s. The discretized matrices were:

$$\mathbf{A_d} = \begin{bmatrix} 0.9970 & 0.0993 & 0 & 0 & 0 \\ -0.0608 & 0.9845 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & -1.0772 \cdot 10^{-5} \\ 0 & 0 & 0 & 0.9986 & -2.1539 \cdot 10^{-4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B_d} = \begin{bmatrix} 0 \\ 0 \\ -1.0772 \cdot 10^{-5} \\ 2.1539 \cdot 10^{-4} \\ 0 \end{bmatrix}$$

$$\mathbf{E_d} = \begin{bmatrix} 3.5999 \cdot 10^{-5} & 0 \\ 7.1811 \cdot 10^{-4} & 0 \\ 0 & -3.5910 \cdot 10^{-7} \\ 0 & -1.0772 \cdot 10^{-5} \\ 0 & 0.1000 \end{bmatrix}$$

Furthermore, we have $\mathbf{C_d} = \mathbf{C}$ and $\mathbf{D_d} = D$.

5.2 Problem b - Variation of measurement noise

To find an estimate of the measurement noise variance, we used the `MATLAB` function `var(A)`, which returns the variance of its argument vector. In `SIMULINK`, the reference heading was set to 0, and only measurement noise was turned on. This means that only the measurement noise on the compass is measured. Running the variance function on this value resulted in a calculated variance of

$$\sigma^2 = 6.066 \cdot 10^{-7}$$

5.3 Problem c - Implementing the Kalman filter

From the assignment [1], we were given

$$\mathbf{w} = \begin{bmatrix} w_w & w_b \end{bmatrix}^T, E\{\mathbf{w}\mathbf{w}^T\} = \mathbf{Q} = \begin{bmatrix} 30 & 0 \\ 0 & 10^{-6} \end{bmatrix}$$

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

where $\mathbf{w}$ is the process noise, $\mathbf{Q}$ is the process noise covariance, $\mathbf{P}_0^-$ is the initial a priori estimate error covariance, and $\hat{\mathbf{x}}_0^-$ is the initial a priori state estimate. Since the process is sampled, $E\{v^2\} = R$ equals the measurement noise variance found in the previous problem divided by the sample interval. This gives:

$$R = \frac{\sigma^2}{Ts} = \frac{6.066 \cdot 10^{-7}}{0.10} = 6.066 \cdot 10^{-6}$$

We chose to implement the Kalman filter using an s-function, shown in appendix A.5 (A modified shell made by Jørgen Spjøtvold). A discrete Kalman filter is made up of the following functions [2], which computes the Kalman gain, estimated states with measurement, error covariance matrix and predictions respectively:

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

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

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

$$\hat{\mathbf{x}}^-[k+1] = \mathbf{A_d}\hat{\mathbf{x}}[k] + \mathbf{B_d}\mathbf{u}[k]$$

$$\mathbf{P}^-[k+1] = \mathbf{A_d}\mathbf{P}[k]\mathbf{A_d}^T + \mathbf{Q}_w$$

Furthermore, to make the s-function access all necessary variables from the workspace, a struct was made up of all previously mentioned discrete state matrices, in addition to the initial state matrices $\mathbf{P}_0^-$ and $\hat{\mathbf{x}}_0^-$. Finally, in the `SIMULINK` implementation, we included a zero-order hold block, in order to convert the continuous system signals to a discrete input for the kalman function block, shown in fig. 18.

The Kalman filter is now able to make estimates for states in the ship model

that were previously unavailable for measuring (such as the rudder bias from the current). This is very useful as we shall see in the following exercises, because it makes it possible to counteract the fairly important effects of these states.

5.4 Problem d - Simulation with measurement noise and current disturbance

The Kalman filter was added to our `SIMULINK` model. We made a feed forward from the estimated bias, to cancel out the bias. The course angle reference was set like in the previous simulations, that is $\psi_r = 30°$. We also kept the saturation block on the input to the cargo ship, to make sure $|\psi|$ didn't exceed 35°. The improved autopilot system with the Kalman filter can be found in fig. 18 in appendix B. We simulated the system with measurement noise and current disturbance. The result of the simulation can be seen in fig. 10.

We see that the heading reaches the reference course angle at around 300 seconds. In contrast, the equivalent simulation in section 3.3 never reached the desired heading because of a steady state error of approximately 3.5°. By implementing a Kalman filter in our system, we are able to remove the stationary error. This is because the estimated bias from the Kalman filter is equal to the error, and adding the estimation to the PD controller signal will cancel the error out. Thus, the effect of the current disturbance is eliminated. This is a highly satisfactory result, and a huge improvement of the autopilot.

5.5 Problem e - Simulation with measurement noise and both disturbances

We adjusted our `SIMULINK` model to use the wave filtered ψ from the Kalman filter as feedback to the regulator, instead of the measured heading from the cargo ship. The altered model can be seen in fig. 19 in appendix B. We simulated the system with the same reference and constraints as in the previous problem, but this time with wave disturbance in addition to measurement noise and current disturbance. The simulation result can be seen in fig. 11.

We observe that regarding the heading, the performance of the new autopilot is quite similar to the equivalent simulation in section 3.4. The system reaches the reference value for the heading, and fluctuates around the desired course. These fluctuations occur because the rudder is trying to counter the effect of the waves. The rudder counteraction is the essential difference between the two autopilots. As we can see from the simulation, the activity of the rudder is far less extreme than what we observed in section 3.4. We still

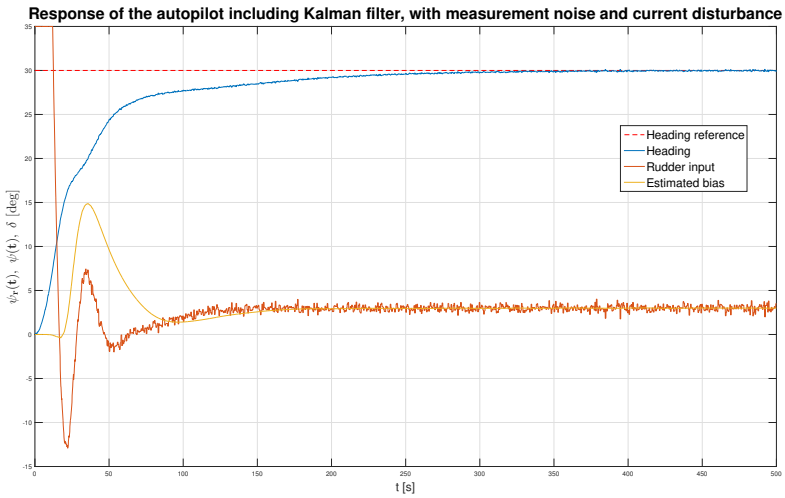


Figure 10: Simulation of the improved autopilot, with measurement noise and current disturbance

have some oscillations, but within a more acceptable range. The new system is less likely to damage the rudder, while still giving a satisfying result for the ship course. These simulations have shown us that utilizing a Kalman filter can make a system more robust, and is very useful in systems affected by noise and disturbances.

Another particularly useful quality of the Kalman filter is that it estimates the wave influence with high accuracy. This makes the model able to counteract the effect of the waves, such that the ship stays on course, while keeping the rudder intact. This is is exactly what we wanted to accomplish with the Kalman filter, as mentioned in section 3.4. To measure the difference of the actual wave influence ψ_ω and estimated wave influence $\hat{\psi}_\omega$, another adjustment was made to our `SIMULINK` diagram which can be examined in fig. 20 in appendix B. The reference was set to 0 and only wave disturbance turned on. The Kalman filter s-function was also slightly modified, by setting the estimated wave influence $\hat{\psi}_\omega$ as output instead of the estimated heading $\hat{\psi}$:

```
sys=[x(8); x(10)];
```

```
sys=[x(7); x(10)];
```

The resulting plot can be seen in fig. 12, where the estimation clearly follows the measured state very well.

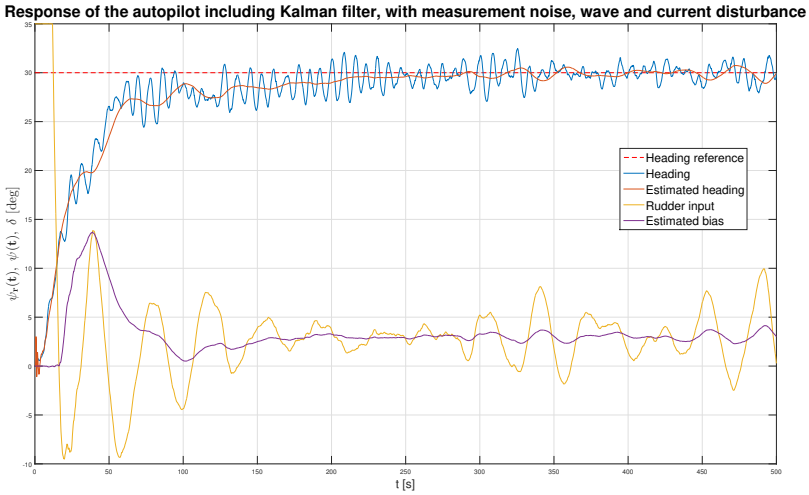


Figure 11: Simulation of the improved autopilot, with measurement noise and both disturbances

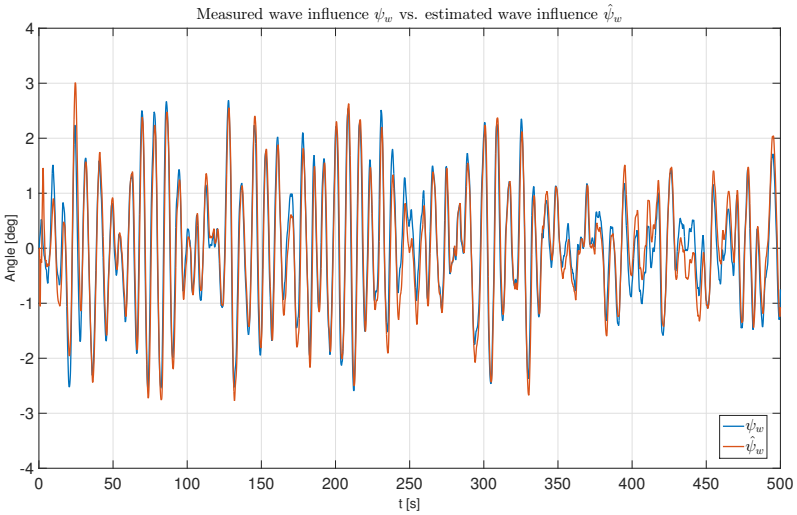


Figure 12: Plot of the actual wave influence against the estimated wave influence

A MATLAB Code

A.1 Part I

```
%% INIT
close all;
clear;
clc;

fig = 1;
addpath('data');

%% PROBLEM 1b

omega_1 = 0.005;
omega_2 = 0.05;

load('P1b_w1.mat');
load('P1b_w2.mat');
load('P1c_w1.mat');
load('P1c_w2.mat');

%Plot response of sine wave
figure(fig);
fig = fig + 1;
subplot(2,1,1)
plot(Compass1.time, Compass1.data, 'LineWidth', 1);
xlabel('t [s]', 'FontSize', 18);
ylabel('$\psi$ [deg]', 'Interpreter', 'latex');
legend({'Heading'} ,'FontSize', 18, 'Location', 'best');
title('Response of sine wave input where \omega_{1} = 0.005', ...
'FontSize', 24);
set(gca,'FontSize',14);
grid on;

subplot(2,1,2)
plot(Compass2.time, Compass2.data, 'LineWidth', 1);
xlabel('t [s]', 'FontSize', 18);
ylabel('$\psi$ [deg]', 'Interpreter', 'latex');
legend({'Heading'} ,'FontSize', 18, 'Location', 'best');
title('Response of sine wave input where \omega_{2} = 0.05', ...
'FontSize', 24);
set(gca,'FontSize',14);
grid on;
```

```
41
42
43  % \\\ Find ampltiude peaks
44  A_1_max = max(Compass1.data(2000:end));
45  A_1_min = min(Compass1.data(2000:end));
46  A_2_max = max(Compass2.data(2000:end));
47  A_2_min = min(Compass2.data(2000:end));
48
49
50  % \\\ Average amplitude value of output
51  A_1 = (A_1_max - A_1_min)/2;
52  A_2 = (A_2_max - A_2_min)/2;
53
54  %\\\ Finding K and T from system output
55  syms T K
56  eqn1 = K/(omega_1*sqrt((omega_1)^2*T^2 + 1)) == A_1;
57  eqn2 = K/(omega_2*sqrt((omega_2)^2*T^2 + 1)) == A_2;
58
59  sol = vpasolve([eqn1, eqn2], [T, K]);
60
61  %Converting sym to double
62  K = double(sol.K);
63  T = double(sol.T);
64
65  %% PROBLEM 1c
66  figure(fig);
67  fig = fig + 1;
68  subplot(2,1,1)
69  plot(Compass3.time, Compass3.data, 'LineWidth', 1);
70  xlabel('t [s]', 'FontSize', 18);
71  ylabel('$\psi$ [deg]', 'Interpreter', 'latex');
72  legend({'Heading'} ,'FontSize', 18, 'Location', 'best');
73  title('Response of sine wave input where \omega_{1} = 0.005', ...
74  'FontSize', 24);
75  set(gca,'FontSize',14);
76  grid on;
77
78  subplot(2,1,2)
79  plot(Compass4.time, Compass4.data, 'LineWidth', 1);
80  xlabel('t [s]', 'FontSize', 18);
81  ylabel('$\psi$ [deg]', 'Interpreter', 'latex');
82  legend({'Heading'} ,'FontSize', 18, 'Location', 'best');
83  title('Response of sine wave input where \omega_{2} = 0.05', ...
84  'FontSize', 24);
```

```
set(gca,'FontSize',14);
grid on;

%% PROBLEM 1d
load('step_simulink.mat');

%Define transfer function
H_tf = tf(K, [T 1 0]);

%Plot step response of model vs simulation
figure(fig);
fig = fig + 1;

step(H_tf, Compass.time(end));
hold on;
plot(Compass.time, Compass.data, 'r', 'LineWidth', 1);
title('Step response of the ship and model', 'FontSize', 22);
legend({'Model', 'Ship'}, 'FontSize', 28, ...
    'Location', 'best');
grid on; hold off;
xlabel('T [s]','FontSize', 22);
ylabel('Amplitude [deg]','FontSize', 22);
set(gca,'FontSize',14);

%% SAVE
addpath('data');
save('data\Part_1_data', 'T', 'K');
```

A.2 Part II

```
%% INIT
close all;
clear;
clc;

fig = 1;
load('wave.mat');
%% PROBLEM 5.2a Calculate estimate of PSD

s_freq = 10; %Hz
window = 4096;
noverlap = [];
nfft = [];

%Find estimated PSD. Converting input from deg to rad
[pxx, f] = pwelch(psi_w(2,:).*(pi/180), window, ...
    noverlap, nfft, s_freq);

%Converting resulting units, from Hz to rad/s%
%and power per Hz to power rad/s.%
omega = 2*pi.*f;
pxx = pxx./(2*pi);


%% PROBLEM 5.2c - Find omega_0

%Plot PSD
figure(fig);
fig = fig + 1;

plot(omega, pxx, 'LineWidth', 2);
axis([0 2 0 16*10^(-4)])

xlabel('$\omega$ [$\frac{\textrm{rad}}{\textrm{s}}$]',...
    'FontSize', 25, 'Interpreter', 'latex');
ylabel('$S_{\psi_{w}}(\omega)$ [rad]','FontSize', 25,...
    'Interpreter', 'latex');
title('Estimated Power spectral density of wave disturbance', ...
'FontSize', 24);
grid on;

%Setting x axis to rad values
```

```
ax = gca;
ax.XTick = 0:pi/8:2;
ax.XTickLabel = {'$0$', '$\frac{\pi}{8}$', '$\frac{\pi}{4}$', ...
    '$\frac{3\pi}{8}$', '$\frac{\pi}{2}$','$\frac{5\pi}{8}$', ...
    '$\frac{3\pi}{4}$'};
ax.TickLabelInterpreter = 'latex';
ax.FontSize = 24;

%Calculating omega_0
[max_pxx, indx] = max(pxx);
omega_0 = omega(indx);

%% PROBLEM 5.2d - Find lambda
%
sigma = sqrt(max_pxx);
lambda_test_values = [0.04 0.08 0.12];

%Plotting S with different values for lambda
figure(fig)
fig = fig + 1;
plot(omega, pxx, 'LineWidth', 2);
axis([0 2 0 16*10^(-4)])

xlabel('$\omega$ [$\frac{\textrm{rad}}{\textrm{s}}$]',...
    'FontSize', 25, 'Interpreter', 'latex');
ylabel('$S_{\psi_{w}}(\omega)$ [rad]','FontSize', 25,...
    'Interpreter', 'latex');
title('Estimated Power spectral density of wave disturbance', ...
'FontSize', 24);
grid on;
hold on;

%Setting x axis to rad values
ax = gca;
ax.XTick = 0:pi/8:2;
ax.XTickLabel = {'$0$', '$\frac{\pi}{8}$', '$\frac{\pi}{4}$', ...
    '$\frac{3\pi}{8}$', '$\frac{\pi}{2}$','$\frac{5\pi}{8}$', ...
    '$\frac{3\pi}{4}$'};
ax.TickLabelInterpreter = 'latex';
ax.FontSize = 24;


for lambda = lambda_test_values
    K_w = 2*lambda*omega_0*sigma;
```

```
87      P_psi_omega = ((K_w*omega).^2)./(omega_0^4 + omega.^4 ...
88      + (omega_0*omega).^2.*(4*lambda^2 - 2));
89      plot(omega, P_psi_omega, '--', 'LineWidth', 1.5);
90  end
91
92  legend({'$S_{\psi_{w}}(\omega)$', '$\lambda = 0.04$', ...
93      '$\lambda = 0.08$', '$\lambda = 0.12$'}, ...
94      'Interpreter', 'Latex', 'Location', 'best', 'FontSize', 25);
95  %Choosing lambda:
96  lambda = 0.08;
97
98  %% SAVE
99  save('data\Part_2_data', 'omega_0', 'lambda', 'K_w');
```

A.3 Part III

```
%% INIT
close all;
clear;
clc;

fig = 1;
load('data\Part_1_data');
%% PROBLEM a
%psi < +-35deg
psi_ref = 30; %degrees
omega_c = 0.1; %rad

T_f = -1/(tan(130*pi/180)*omega_c);
K_pd = sqrt(omega_c^2+T_f^2*omega_c^4)/K;
T_d = T;

%Transfer functions
H_ship = tf(K, [T 1 0]);
H_pd = tf([K_pd*T_d K_pd], [T_f 1]);
H_0 = H_ship * H_pd;

%Plotting
figure(fig);
fig = fig + 1;
[mag, phase] = bode(H_0, 0.1);
margin(H_0);
grid on;

%% PROBLEM b
%Simulation results, autopilot
%Load simulation results
load('data\P3b_reference.mat');
load('data\P3b_heading.mat');
load('data\P3b_rudder_input.mat');

figure(fig)
fig = fig + 1;
plot(heading_reference.time, heading_reference.data, 'r--', ...
    heading.time, heading.data, rudder_input.time,...
    rudder_input.data, 'LineWidth', 1);
xlabel('t [s]', 'FontSize', 18);
ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
```

```
    'FontSize', 20, 'Interpreter', 'latex');
legend({'Heading reference', 'Heading', 'Rudder input'}, ...
    'FontSize', 18, 'Location', 'best');
title('Response of the autopilot with measurement noise', ...
'FontSize', 24);
grid on;

%% PROBLEM c
%Plotting simulation results, autopilot with current disturbance
%Load simulation results
load('data\P3c_reference.mat');
load('data\P3c_heading.mat');
load('data\P3c_rudder_input.mat');

figure(fig)
fig = fig + 1;
plot(heading_reference_c.time, heading_reference_c.data, 'r--', ...
    heading_c.time, heading_c.data, rudder_input_c.time,...
    rudder_input_c.data, 'LineWidth', 1);
xlabel('t [s]', 'FontSize', 18);
ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
    'FontSize', 20, 'Interpreter', 'latex');
legend({'Heading reference', 'Heading', 'Rudder input'},...
    'FontSize', 18, 'Location', 'best');
title('Response of the autopilot with noise and current', ...
'FontSize', 24);
grid on;

%% PROBLEM d
%Plotting simulation results, autopilot with wave disturbance
%Load simulation results
load('data\P3d_reference.mat');
load('data\P3d_heading.mat');
load('data\P3d_rudder_input.mat');

figure(fig)
fig = fig + 1;
plot(heading_reference_d.time, heading_reference_d.data, 'r--', ...
    heading_d.time, heading_d.data, rudder_input_d.time,...
    rudder_input_d.data, 'LineWidth', 1);
xlabel('t [s]', 'FontSize', 18);
ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
    'FontSize', 20, 'Interpreter', 'latex');
legend({'Heading reference', 'Heading', 'Rudder input'},...
```

```
87      'FontSize', 18, 'Location', 'best');
88  title('Response of the autopilot with noise and waves', ...
89  'FontSize', 24);
90  grid on;
91
92  %% NORTH-EAST plot
93  %Load data from simulations
94  load('data\P3b_xy.mat');
95  load('data\P3c_xy.mat');
96  load('data\P3d_xy.mat');
97
98  plot(xy_plot.y.data,xy_plot.x.data,'r--', ...
99      xy_plot_c.y.data,xy_plot_c.x.data, ...
100     xy_plot_d.y.data,xy_plot_d.x.data,'m', 'LineWidth', 2);
101 title('North-East plot of ship course', 'FontSize', 24);
102 ylabel('North [m]', 'FontSize', 20); grid on;
103 xlabel('East [m]', 'FontSize', 20);
104 ax = gca; ax.FontSize = 24; axis([0 150 0 600]);
105 legend({'Without disturbances', 'With current', ...
106 'With waves'}, 'Location', 'best', 'FontSize', 36)
107
108 %% SAVE
109 save('data\part3_data', 'K_pd', 'T_d', 'T_f')
```

A.4 Part V

```
%% INIT
close all;
clear;
clc;

psi_ref = 30;
fig = 1;
load('data/part_1_data.mat');
load('data/part_2_data.mat');
load('data/part3_data.mat');
%% SYSTEM
%From part IV:
A = [0, 1, 0, 0, 0; -omega_0^2, -2*lambda*omega_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];
C = [0, 1, 1, 0, 0];
D = 0;
E = [0, 0; K_w, 0; 0, 0; 0, 0; 0, 1];

%% PROBLEM A - DISCRETIZATION
%Discrete system
Ts = 0.1;

[~, B_d] = c2d(A, B, Ts);
C_d = C;
[A_d, E_d] = c2d(A, E, Ts);
D_d = D;
%% PROBLEM B - VARIANCE
load('data\P5b_measurement_noise.mat');
%Variance in measurement noise
variance = var(measurement_noise.data*pi/180);
%% PROBLEM C - KALMAN FILTER
%New matrices
Q = [30, 0; 0, 10e-6];
P_0_a_priori = [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, 2.5e-3];
x_0_a_priori = [0; 0; 0; 0; 0];

R = variance/Ts; %Variance

sys = struct('A_d', A_d, 'B_d', B_d, 'C_d',...
```

```
43      C_d, 'E_d', E_d, 'Q', Q, 'R', R,...
44      'P_0_a_priori', P_0_a_priori, 'x_0_a_priori', x_0_a_priori);
45
46
47  %% PROBLEM D - Simulation with estimated bias
48  %Load simulation results
49  load('data\P5d_reference.mat');
50  load('data\P5d_heading.mat');
51  load('data\P5d_rudder_input.mat');
52  load('data\P5d_est_bias.mat');
53
54
55  figure(fig)
56  fig = fig + 1;
57  plot(heading_reference_5d.time, heading_reference_5d.data,'r--',...
58      heading_5d.time, heading_5d.data, rudder_input_5d.time,...
59      rudder_input_5d.data, est_bias_5d.time,...
60      est_bias_5d.data, 'LineWidth', 1);
61  xlabel('t [s]', 'FontSize', 18);
62  ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
63      'FontSize', 20, 'Interpreter', 'latex');
64  legend({'Heading reference', 'Heading', 'Rudder input',...
65      'Estimated bias'},'FontSize', 18, 'Location', 'best');
66  title('Response of the autopilot including Kalman filter',...
67  'FontSize', 24);
68  grid on;
69
70  %% PROBLEM E - SIMULATION WITH FILTERET PSI
71  %Load simulation results
72  load('data\P5e_reference.mat');
73  load('data\P5e_heading.mat');
74  load('data\P5e_rudder_input.mat');
75  load('data\P5e_est_bias.mat');
76  load('data\P5e_heading_est.mat');
77  load('data\P5e2_heading_est.mat');
78  load('data\P5e2_heading.mat');
79  load('data\P5e2_est_bias.mat');
80
81  figure(fig)
82  fig = fig + 1;
83  plot(heading_reference_5e.time, heading_reference_5e.data,'r--',...
84      heading_5e.time, heading_5e.data, est_heading_5e.time,...
85      est_heading_5e.data, rudder_input_5e.time,...
86      rudder_input_5e.data, est_bias_5e.time,...
```

```
87      est_bias_5e.data, 'LineWidth', 1);
88  xlabel('t [s]', 'FontSize', 18);
89  ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
90      'FontSize', 20, 'Interpreter', 'latex');
91  legend({'Heading reference', 'Heading', 'Estimated heading',...
92      'Rudder input', 'Estimated bias'},'FontSize',...
93      18, 'Location', 'best');
94  title('Response of the autopilot including Kalman filter', ...
95  'FontSize', 24);
96  grid on;
97
98  %Plotting wave influence
99  figure(fig)
100 fig = fig + 1;
101 plot(heading_5e2.time, heading_5e2.data, est_heading_5e2.time,...
102     est_heading_5e2.data, 'LineWidth', 2)
103 title(['Influence of Measured wave $\psi_{w}$ vs. estimated wave'...
104 ' $\hat{\psi}_{w}$'], 'FontSize', 24, 'Interpreter', 'latex');
105 xlabel('t [s]', 'FontSize', 20);
106 ylabel('Angle [deg]', 'FontSize', 20);
107 legend({'$\psi_{w}$', '$\hat{\psi}_{w}$'}, 'FontSize', 24, ...
108 'Interpreter', 'latex', 'Location', 'best')
109 ax = gca; ax.FontSize = 24; grid on;
110 axis([0 500 -4 4]);
```

A.5 Kalman s-function

```
function [sys,x0,str,ts] = DiscKal(t,x,u,flag,data)
% Shell for the discrete kalman filter assignment in
% TTK4115 Linear Systems.
%
% Author: Jorgen Spjotvold
% 19/10-2003
%

switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]= mdlInitializeSizes(data);

  %%%%%%%%%%%%%
  % Outputs   %
  %%%%%%%%%%%%%

  case 3,
    sys=mdlOutputs(t,x,u,data);
  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%

  case 2,
    sys=mdlUpdate(t,x,u, data);

  case {1,4,}
    sys=[];

  case 9,
      sys=mdlTerminate(t,x,u);
  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

```

```
43 function [sys,x0,str,ts] = mdlInitializeSizes(data)
44 % This is called only at the start of the simulation.
45
46 sizes = simsizes; % do not modify
47
48 sizes.NumContStates  = 0; % Number of continuous states in the system,
49 sizes.NumDiscStates  = 35; % Number of discrete states in the system, m
50 sizes.NumOutputs     = 2; % Number of outputs, the hint states 2
51 sizes.NumInputs      = 2; % Number of inputs, the hint states 2
52 sizes.DirFeedthrough = 1; % 1 if the input is needed directly in the
53 % update part
54 sizes.NumSampleTimes = 1; % Do not modify
55
56 sys = simsizes(sizes); % Do not modify
57
58 x0  = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, (data.P_0_a_priori(:)')]'; % Initi
59
60 str = []; % Do not modify
61
62 ts  = [-1 0]; % Sample time. [-1 0] means that sampling is
63 % inherited from the driving block and that it changes during
64 % minor steps.
65
66
67 function sys = mdlUpdate(t,x,u, data)
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69 % Update the filter covariance matrix and state etsimates here.
70 % example: sys=x+u(1), means that the state vector after
71 % the update equals the previous state vector + input nr one.
72 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73
74 P_priori = reshape(x(11:35), 5, 5);
75 L = P_priori*(data.C_d)'*(data.C_d*P_priori*(data.C_d)' + data.R)^(-1);
76
77 x_priori = x(1:5);
78 x_posteriori = x_priori + L*(u(2) - data.C_d*x_priori); %Input number 2
79
80 I = eye(5);
81 P_posteriori = (I - L*data.C_d)*P_priori*(I - L*data.C_d)' + L*data.R*(
82
83 x_predict = data.A_d*x_posteriori + data.B_d*u(1); %u(1) is first input
84 P_predict = data.A_d*P_posteriori*(data.A_d)' + data.E_d*data.Q*(data.E
85
86 sys=[x_predict; x_posteriori; P_predict(:)];
```

```
87
88 function sys = mdlOutputs(t,x,u,data)
89
90 sys=[x(8); x(10)];
91
92 function sys = mdlTerminate(t,x,u)
93 sys = [];
```

B SIMULINK Diagrams

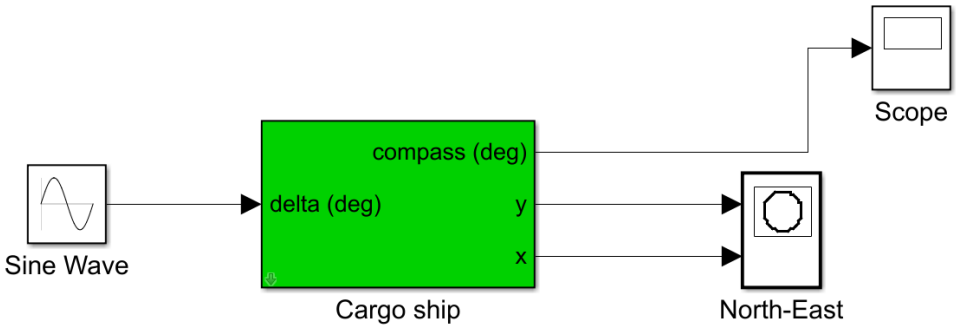


Figure 13: Simulink diagram for problem 5.1b All disturbances are turned off

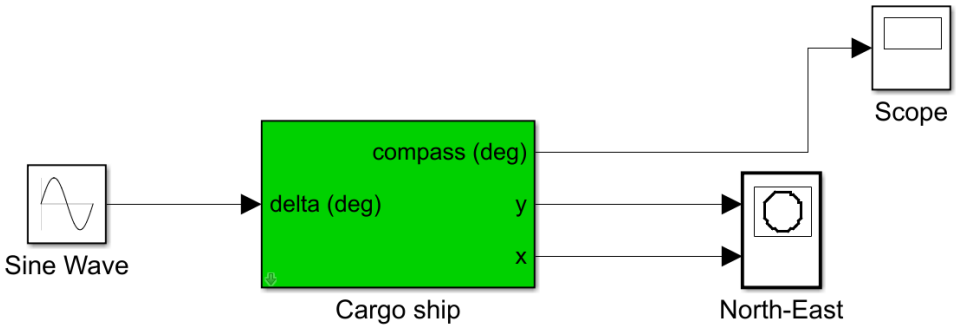


Figure 14: Simulink diagram for problem 5.1c Waves and measurement noise is turned on

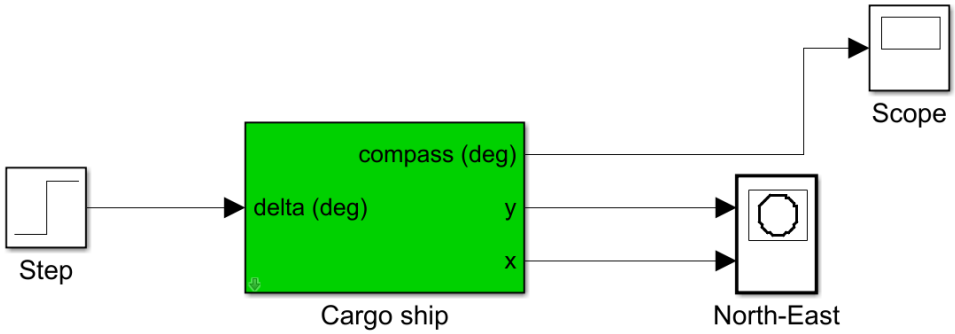


Figure 15: Simulink diagram for problem 5.1d

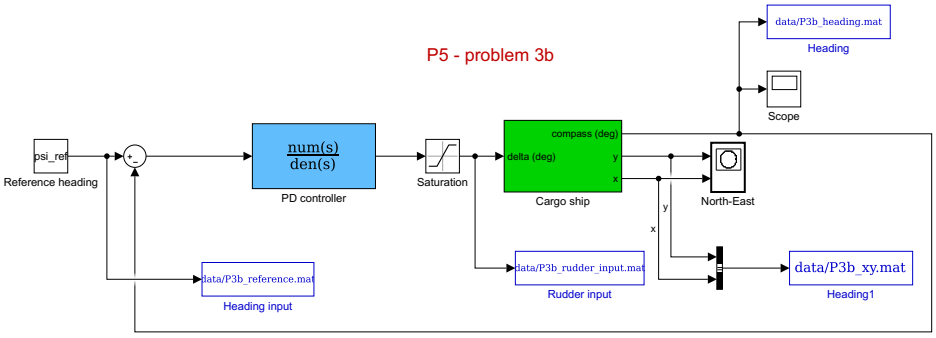


Figure 16: Simulink diagram for problem 5.3b - 5.3d

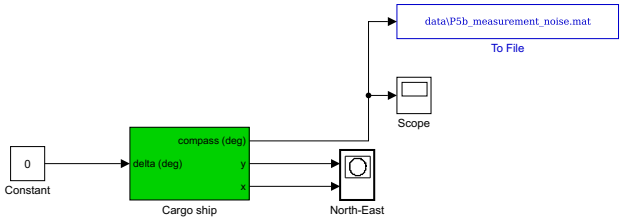


Figure 17: Simulink diagram for problem 5.5b

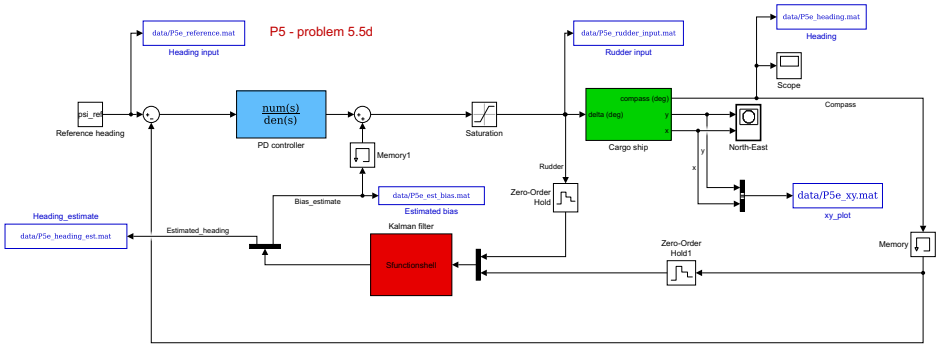


Figure 18: Simulink diagram for problem 5.5c and 5.5d

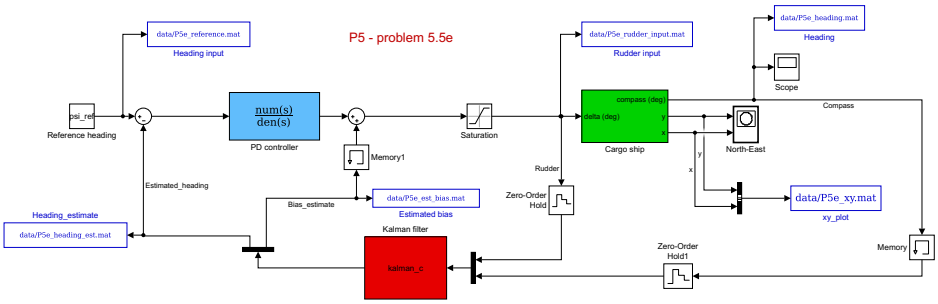


Figure 19: Simulink diagram for problem 5.5e

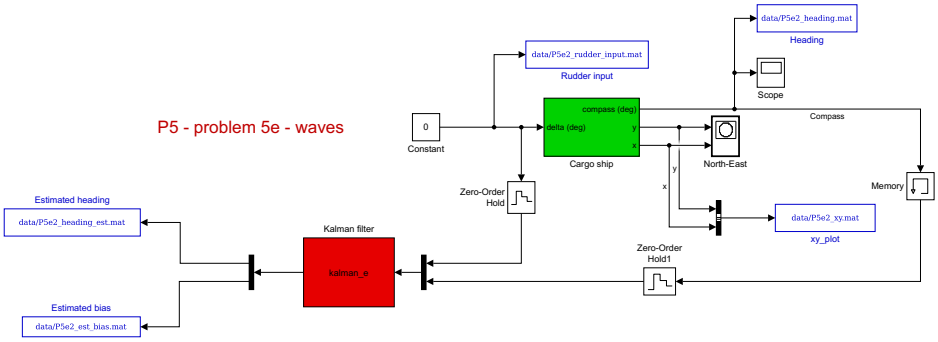


Figure 20: Simulink diagram for finding wave influence in problem 5.5e

References

[1] NTNU Trondheim Institutt for teknisk kybernetikk. *Discrete Kalman Filter Applied to a Ship Autopilot*. https://ntnu.blackboard.com/bbcswebdav/pid-203989-dt-content-rid-5423225_1/courses/194_TTK4115_1_2017_H_1/boat_assignment%281%29.pdf. Accessed: 2017-11.

[2] Morten D. Pedersen. *The Kalman Filter*. https://ntnu.blackboard.com/bbcswebdav/pid-202884-dt-content-rid-5177025_1/courses/194_TTK4115_1_2017_H_1/Lecture_K2.pdf. Accessed: 2017-11-17.