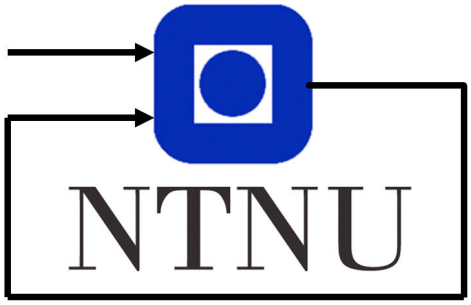
TTK4115 - Linear System Theory
Boat lab assignment

Group 47
Aune, Simen Rogn 771966
Krivokapic, Vuk 253908

November 22, 2017



Department of Engineering Cybernetics

Abstract

This report presents our calculations and discussions to the problems listed in the assignment text. The objective of the assignments are summarized to be:

- To partly model and simulate a continuous system influenced by stochastic signals.
- To use basic identification techniques on parameters that are not explicitly given.
- To use basic control theory to design a simple autopilot.
- To implement a discrete Kalman filter for wave filtering and estimation of disturbances using `MATLAB` and `SIMULINK`.

By the end of this report we will have been through each of the listed objectives.

Contents

1 Identification of the boat parameters

1.1 Transfer function from δ to ψ

$$\dot{\psi} = r \tag{1a}$$

$$\dot{r} = \frac{1}{T}r + \frac{K}{T}(\delta - b) \tag{1b}$$

Our task is to calculate the transfer function form δ to ψ with no disturbances in our system. Because of that, the bias to the rudder angle, $\boldsymbol{b}$, will not be present. By derivating and performing the inverse Laplace on equation 1a, then inserting eq. 22d into eq. 1a, the transfer function will be found.

$$\begin{aligned}
\ddot{\psi} &= \dot{r} \\
&= -\frac{1}{T}r + \frac{K}{T}\delta \\
&= -\frac{1}{T}\dot{\psi} \\
\mathscr{L}^{-1} \Rightarrow s^2 \cdot \psi(s) &= -\frac{1}{T}s \cdot \psi(s) + \frac{K}{T} \cdot \delta(s) \\
(s^2 + \frac{1}{T}s) \cdot \psi(s) &= \frac{K}{T} \cdot \delta(s)
\end{aligned}$$

$$H(s) = \frac{\psi(s)}{\delta(s)} = \frac{K}{Ts^2 + s} = \frac{K}{s(Ts + 1)} \tag{2}$$

The variables T and K are the Nomoto time and gain constants, which is a commonly used model for modelling yaw in marine systems.

1.2 Boat parameters K and T

1.2.1 Without noise or other disturbances

To calculate the boat parameters K and T we give the system a sine input signal. This signal will be sent with the two different values of ω (0.005 and 0.05). The response for each value are as seen in figure 1.

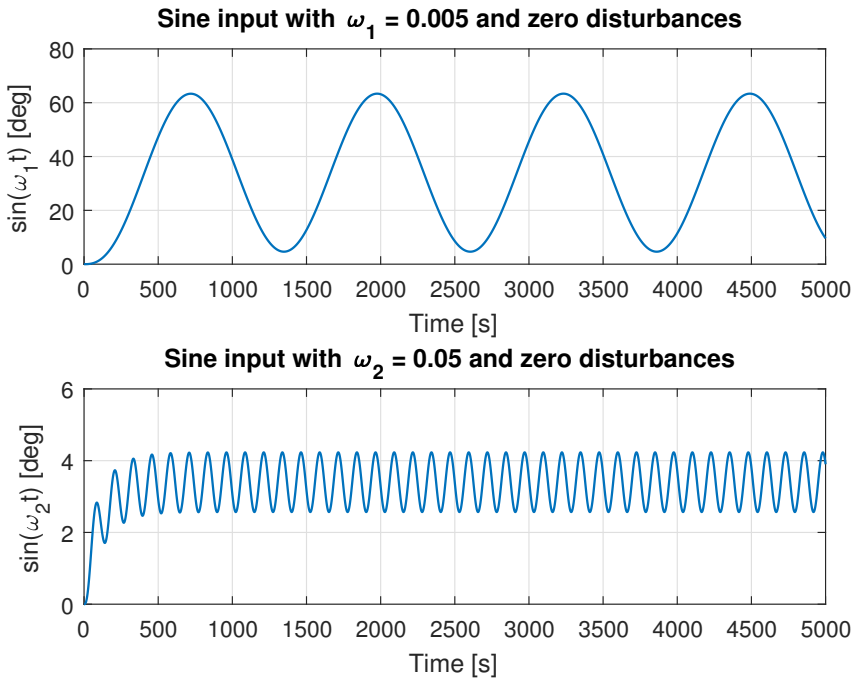


Figure 1: Respones with different sine input signals

From both responses we can calculate the amplitude $\left|H(j\omega)\right|$.

$$A_1 = \left|H(j\omega_1)\right| = \frac{max - min}{2} = \frac{63.36 - 4.642}{2} = 29.359 \tag{3a}$$

$$A_2 = \left|H(j\omega_2)\right| = \frac{max - min}{2} = \frac{4.231 - 2.569}{2} = 0.831 \tag{3b}$$

$$
\begin{aligned}
\left|H(j\omega_n)\right| &= \left|\frac{K}{j\omega_n(j\omega_n T+1)}\right| \\
&= \left|\frac{K}{j^2{\omega_n}^2 T+j\omega_n}\right| \\
&= \left|\frac{K}{j\omega_n-{\omega_n}^2 T}\right| \\
&= \frac{K}{\sqrt{({\omega_n}^2 T)^2+\omega_n^2}} \\
&= \frac{K}{\omega_n\sqrt{\omega_n^2T^2+1}}
\end{aligned}
$$

$$
A_n = \frac{K}{\omega_n\sqrt{\omega_n^2T^2+1}} \tag{4}
$$

We wish to calculate the unknown parameters T and K. In order to accomplish this, we need to have a set of two equations, since we have two unknown constants. From eq. 4 we derive the two equations.

$$
\begin{aligned}
K &= A_1\omega_1\sqrt{{\omega_1}^2T^2+1} \\
K &= A_2\omega_2\sqrt{{\omega_2}^2T^2+1} \\
A_1\omega_1\sqrt{{\omega_1}^2T^2+1} &= A_2\omega_2\sqrt{{\omega_2}^2T^2+1}
\end{aligned}
$$

$$
T = \sqrt{\frac{(A_2\omega_2)^2-(A_1\omega_1)^2}{A_1^2\omega_1^4-A_2^2\omega_2^4}} \tag{5}
$$

By inserting values for A_1, A_2, ω_1 and ω_2 into eq. 5 we get the following constants, without noise or other disturbances in the system.

$$
\begin{aligned}
T &= 72.442 \\
K &= 0.156
\end{aligned}
$$

1.2.2 With measurement noise and waves

With measurement noise and waves turned on (also called tough weather conditions) it is significantly harder to calculate the boat parameters. From figure 2 we try our best to calculate the values of A_1 and A_2

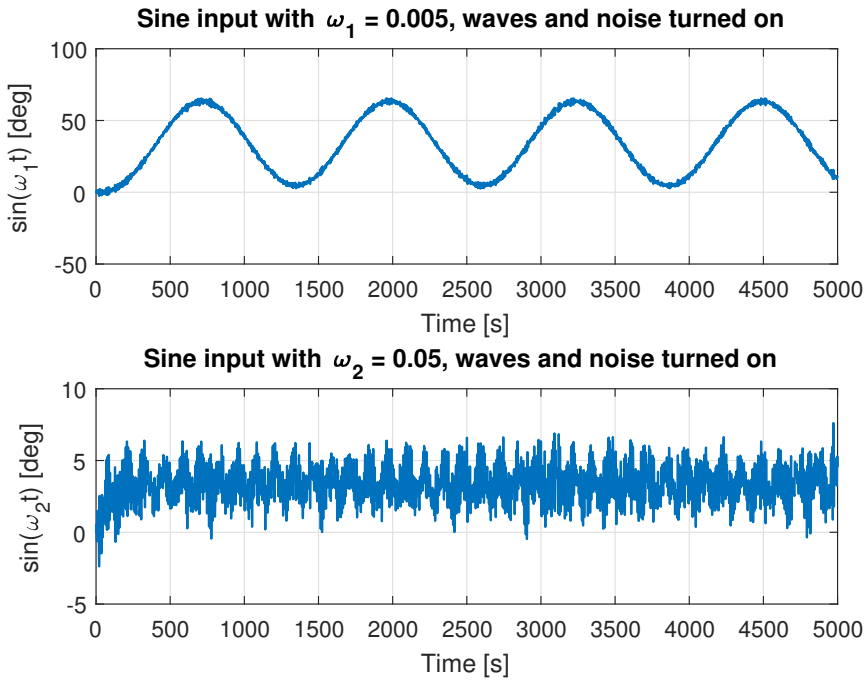


Figure 2: Respones with different sine input signals included noise and waves

$$A_1 = \left|H(j\omega_1)\right| = \frac{max - min}{2} = \frac{63.5 - 4.5}{2} = 29.500 \tag{6a}$$

$$A_2 = \left|H(j\omega_2)\right| = \frac{max - min}{2} = \frac{5.0 - 2.0}{2} = 1.500 \tag{6b}$$

With new amplitude-values the boat parameters T and K differ from the previous set. From eq. 5 we calculate the parameters exactly as in section 1.2.1.

$$T = 34.544$$
$$K = 0.150$$

It is not possible to get a good estimate of the boat parameters with noise and disturbance turned on. In figure 2 we can clearly see noise and disturbance affecting the system.

1.3 Step response (ship and model)

By applying a step response of 1 degree to the rudder we can compare responses of the actual ship with its calculated models (with and without noise and disturbances). Figure 3 demonstrates the three responses in one plot, telling us that the boat parameters calculated without noise and disturbances gives us a better estimate. In the coming tasks we will use model without noise and disturbances.

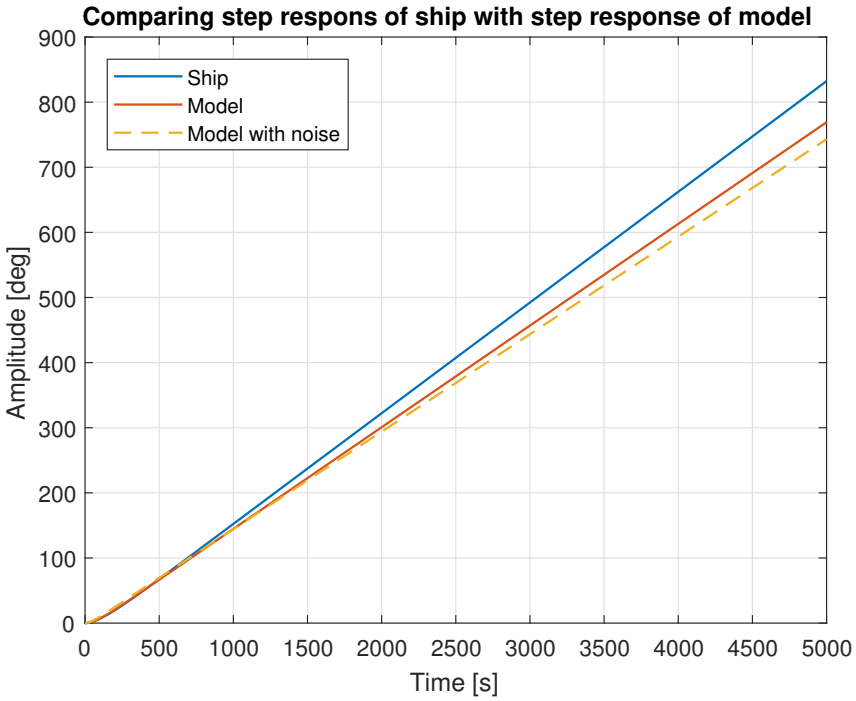


Figure 3: Step response of ship and model

2 Identification of wave spectrum model

2.1 Estimating the Power Spectral Density (PSD)

In this task we are going to calculate an estimate of the Power Spectral Density (PSD). The PSD of a system tells us where the average power is distributed as a function of frequency.
A part of our MATLAB-code regarding the estimated PSD can be seen below. The rest of the script can be found in the appendix.

```
load('wave.mat');
fs = 10;
window = 4096;
noverlap = [];
nfft = [];
[est_psd, f] = pwelch(psi_w(2,:).*(pi/180),window,
   noverlap,nfft,fs);
```

The result of running the MATLAB-code can be seen from figure 4.

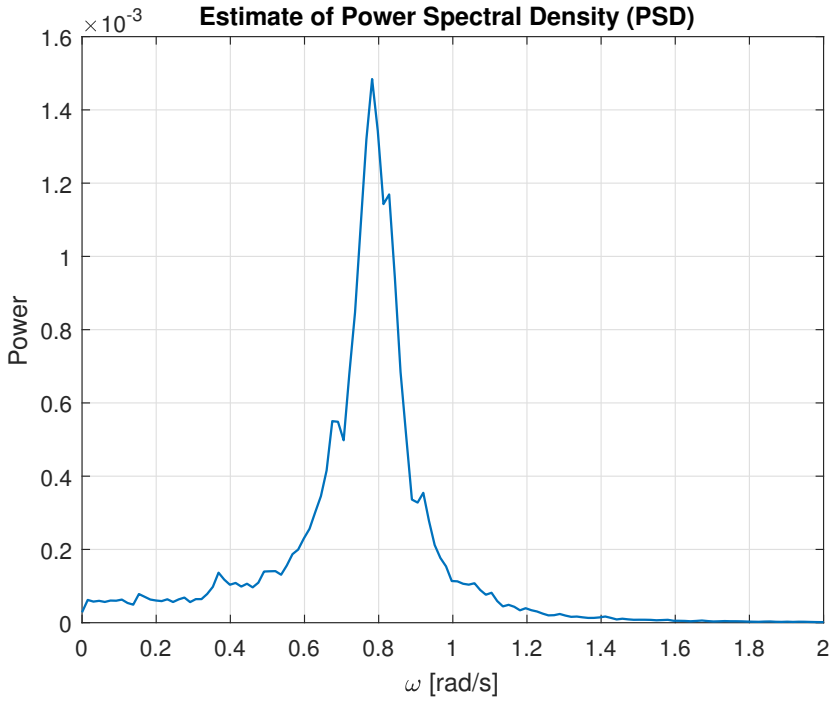


Figure 4: Estimated PSD

2.2 Analytical expressions

2.2.1 Transfer function from ω_w to ψ_w

$$\dot{\psi}_w = -{\omega_0}^2 \xi_w - 2\lambda\omega_0\psi_w + K_w\omega_w \tag{7a}$$
$$\dot{\xi}_w = \psi_w \tag{7b}$$

Laplace tranform of eq. 7b and eq. 7a gives:

$$\mathscr{L}^{-1} \Rightarrow s \cdot \psi_w(s) = -{\omega_0}^2 \xi_w(s) - 2\lambda\omega_0\psi_w(s) + K_w\omega_w(s) \tag{8}$$
$$\mathscr{L}^{-1} \Rightarrow s \cdot \xi_w(s) = \psi_w(s) \Rightarrow \xi(s) = \frac{\psi_w(s)}{s} \tag{9}$$

By inserting eq. 9 into eq. 8 we eventually get the transfer function from ω_ω to ψ_ω.

$$\begin{aligned} s \cdot \psi_w(s) &= -{\omega_0}^2 \frac{\psi_w(s)}{s} - 2\lambda\omega_0\psi_w(s) + K_w\omega_w(s) \\ s^2 \cdot \psi_w(s) &= -{\omega_0}^2 \cdot \psi_w(s) - (2\lambda\omega_0)s \cdot \psi_w(s) + (K_w)s \cdot \omega_w(s) \\ (s^2 + {\omega_0}^2 + 2\lambda\omega_0 s) \cdot \psi_w(s) &= K_w s \cdot \omega_w(s) \end{aligned}$$

$$G(s) = \frac{\psi_w(s)}{\omega_w(s)} = \frac{K_w s}{s^2 + 2\lambda\omega_0 \cdot s + {\omega_0}^2} \tag{10}$$

2.2.2 Power Spectral Density

Analytical expression for the PSD function of ψ_w is calculated below. It will be needed to calculate a value of the damping factor λ later in this assignment.

$$\begin{aligned}
P_{\psi_w}(\omega) &= P_{\omega_w}(\omega) \cdot \left|G(j\omega)\right|^2 \\
&= \left|G(j\omega)\right|^2 \\
&= \left|G(j\omega)\right| \cdot \left|G(-j\omega)\right| \\
&= \frac{K_w \cdot j\omega}{(j\omega)^2 + 2\lambda\omega_0 \cdot j\omega + {\omega_0}^2} \cdot \frac{K_w \cdot (-j\omega)}{(-j\omega)^2 - 2\lambda\omega_0 \cdot (j\omega) + {\omega_0}^2} \\
&= \frac{(K_w\omega)^2}{\omega^4 + {\omega_0}^4 + 2\omega^2{\omega_0}^2 - (2\lambda\omega_0)^2\omega^2} \\
&= \frac{(K_w\omega)^2}{\omega^4 + {\omega_0}^4 + 2\omega^2{\omega_0}^2(2\lambda^2 - 1)}
\end{aligned}$$

$$P_{\psi_w}(\omega) = \frac{(K_w\omega)^2}{\omega^4 + {\omega_0}^4 + 2\omega^2{\omega_0}^2(2\lambda^2 - 1)} \tag{11}$$

2.3 Finding ω_0 from the estimated $S_{\psi_w}(\omega)$

From figure 4 we can obtain the point where the average power distributed as a function of frequency is found at. Using the `Data Cursor` tool in the `MATLAB` plot window, we are able to read the frequency and power where the point lies. We read out $\omega_0 = 0.7823$ and the power to be 0.001484.

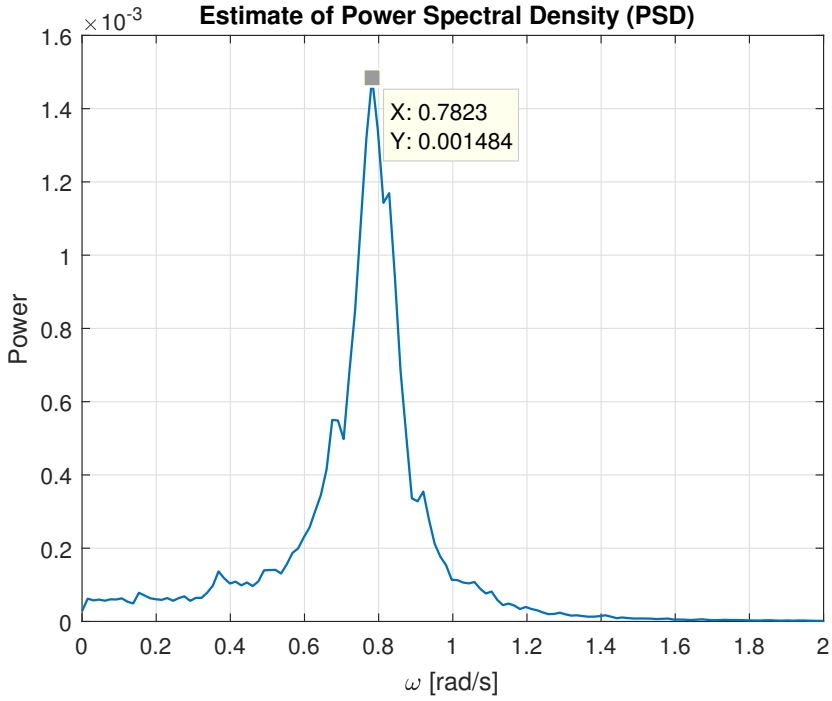


Figure 5: Estimated PSD

2.4 Uncovering λ

In order to achieve a complete model for the wave response we need to identify the damping factor λ. By trial and error when fitting P_{ψ_ω} (eq. 11) to the estimate of the PSD, S_{ψ_ω}, we can come up with a value of the damping factor. From eq. 11 we are introduced to the constant K_ω. This constant is defined by eq. 12.

$$K_\omega = 2\lambda\omega_0\sigma \tag{12}$$

The value for σ is the square root of the power at the point where the average power distributed is located. From figure **??** we get:

$$\sigma = \sqrt{0.001484} = 0.0385 \tag{13}$$

The analytical expression of the PSD found earlier will be used to retrieve this factor. A `MATLAB`-script plotting $P_{\psi_w}(\omega)$ for different values of λ shows which λ that suits the best. What we want is the analytical expression to be as similar to the estimated PSD (figure 4). We conclude with $\lambda = 0.085$. As seen in figure 6, P_{ψ_ω} is fitted nicely to the estimated PSD.

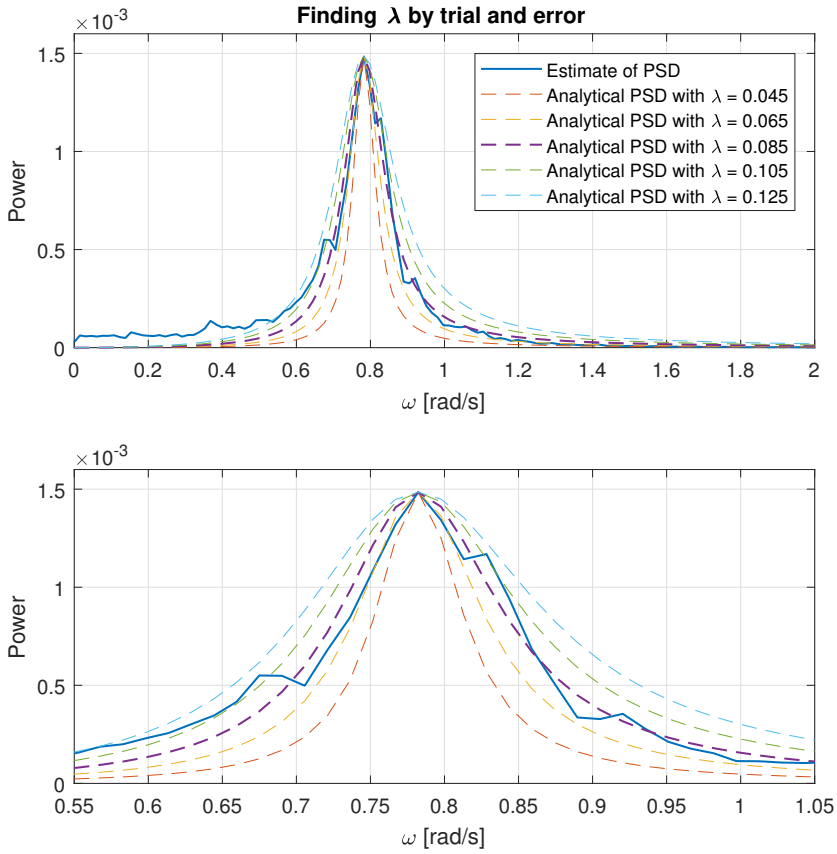


Figure 6: Finding a value for λ

3 Control system design

3.1 Designing of PD-controller

Equation for PD-controller.

$$K_{pd}(s) = K_p \frac{1 + T_d s}{1 + T_f s} \tag{14}$$

From section 1, we have:

$$H_{ship}(s) = \frac{\psi(s)}{\delta(s)} = \frac{K}{Ts^2 + s} = \frac{0.156}{72.442s^2 + 1} \tag{15}$$

Open-loop transfer function H_0 is a combination of H_{pd} and H_{ship}:

$$h_0(s) = H_{ship} H_{pd} = K K_{pd} \frac{1 + T_d s}{(1 + Ts)(1 + T_f s)} \tag{16}$$

Choosing $T_d = T$, such that it cancels out the time constant:

$$h_0(s) = \frac{K K_{pd}}{s(1 + T_f s)} \tag{17a}$$

$$h_0(j\omega_c) = \frac{K K_{pd}}{j\omega_c - T_f \omega_c^2} \tag{17b}$$

$$\angle h_0(j\omega_c) = arctan(\frac{\omega_c}{T_f \omega_c^2}) = arctan(\frac{1}{T_f \omega_c}) \tag{17c}$$

$$\left| h_0(j\omega_c) \right| = \frac{K K_{pd}}{\sqrt{\omega_c^2 + T_f^2 \omega_c^4}} \tag{17d}$$

Our desired $\omega_c = 0.1 rad/sec$ and $\Delta\phi = 50°$.

Finding T_f that satisfies our conditions:

$$\Delta\phi = 180^\circ + \angle h_0(j\omega_c) \tag{18a}$$
$$\angle h_0(j\omega_c) = \Delta\phi - 180^\circ \tag{18b}$$
$$\angle h_0(j\omega_c) = 50^\circ - 180^\circ = -130^\circ \tag{18c}$$
$$arctan(\frac{1}{T_f\omega_c}) = -130^\circ \tag{18d}$$
$$T_f = \frac{1}{\omega_c tan(-130^\circ)} \tag{18e}$$
$$T_f = \frac{1}{0.10tan(-130^\circ)} = 8.39sec \tag{18f}$$

We want our gain to be 0 dB at ω_c. 0 dB = 1 in absolute value. Using 0 dB condition to find K_{pd}:

$$\left|h_0(j\omega_c)\right| = 1 \tag{19a}$$
$$\frac{KK_{pd}}{\sqrt{\omega_c^2 + T_f^2\omega_c^4}} = 1 \tag{19b}$$
$$K_{pd} = \frac{\sqrt{\omega_c^2 + T_f^2\omega_c^4}}{K} \tag{19c}$$
$$K_{pd} = \frac{\sqrt{0.1^2 + 8.39^2 0.1^4}}{0.156} \tag{19d}$$
$$K_{pd} = 0.837 \tag{19e}$$

3.2 Simulation with measurement noise only

From figure 7 it can be seen that the autopilot works well. The ship reaches reference angle after approximately 350 seconds. It can be observed that the input changes direction long time before the rudder reaches the reference. This is the derivative part in action. The controller senses how fast rudder is heading to the reference point and slows it down. That is why we get such a smooth curve in figure 7.

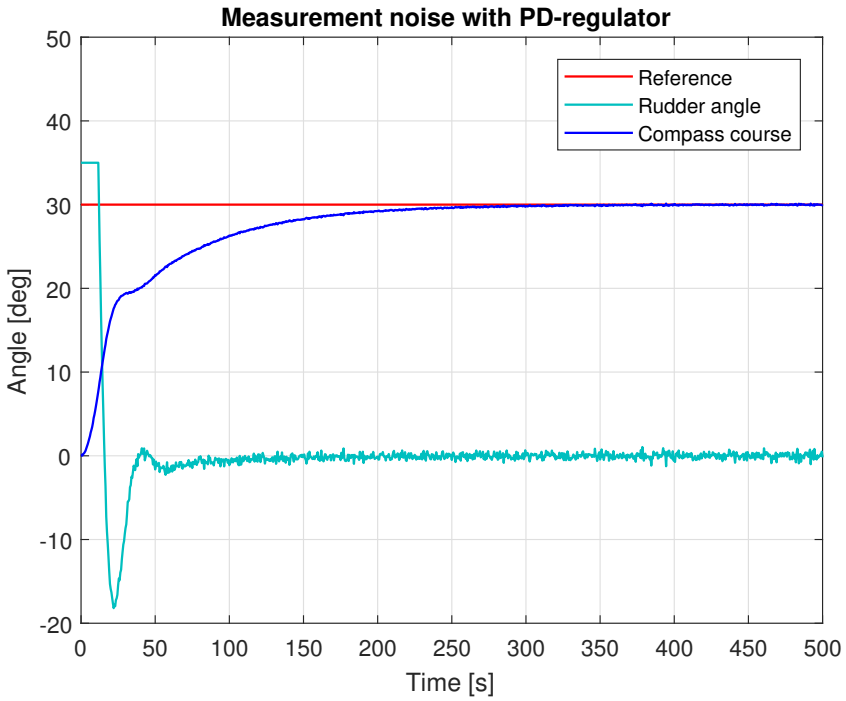


Figure 7: Response with only measurement noise

3.3 Simulation with current disturbance

Current disturbances gives an state error of approximately 3 degrees. In the long run, this error will lead the ship in the totally wrong direction. It can be seen that figure 7 and figure 8 has exactly the same form, except the offset. The offset problem can be solved by an integrator. An integrator would cancel out the offset. Another solution is to somehow make an estimate of the bias value based on disturbances, and use the estimate to cancel out the error. We believe that the Kalman filter can make a good estimate of the bias value and cancel out the error.

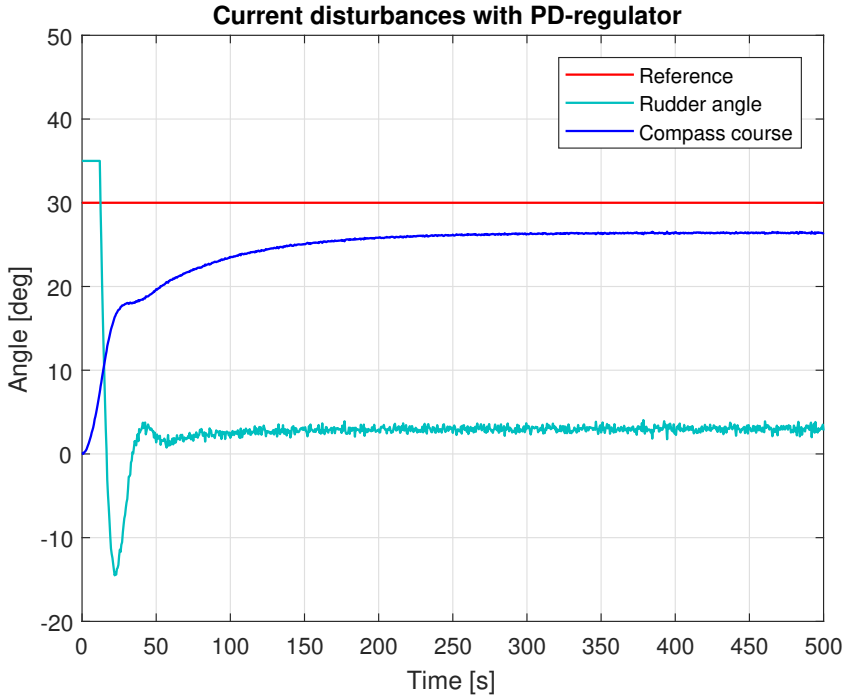


Figure 8: Response with current disturbance

3.4 Simulation with wave disturbance

Waves drags the ship i all directions. As an result of that, wave disturbances are described as high frequency noise. Figure 9 shows how wave disturbances affect our system. We can see that the compass course oscillates around the reference point. The oscillations are not that big, and does not seem to be a big problem. Still, we have to be aware of the fact that our feedback-loop is using the measured compass course to measure the error, which is further used to compute the input signal. A high frequency noise at the measured compass angle can lead to a input signal with high frequency noise, which can make the whole system unstable. To avoid that we need to suppress the noise. We are going to try to make a noise suppression using the Kalman filter.

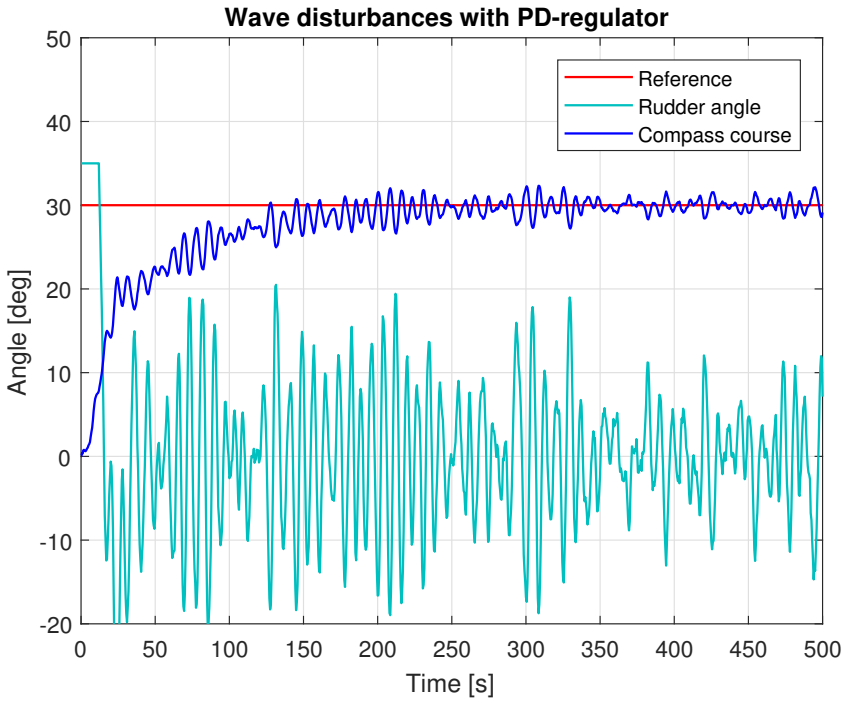


Figure 9: Response with wave disturbance

4 Observability

The system can be written as:

$$\dot{x} = Ax + Bu + E\omega, \quad y = Cx + v \tag{20}$$

$$x = \begin{bmatrix} \xi_\omega \\ \psi_\omega \\ \psi \\ r \\ b \end{bmatrix}, \quad u = \delta, \quad w = \begin{bmatrix} \omega_\omega \\ \omega_b \end{bmatrix} \tag{21a}$$

To calculate the system matrices A, B, C and E we have to look at the model of the ship.

$$\dot{\xi_\omega} = \psi_\omega \tag{22a}$$
$$\dot{\psi_\omega} = -{\omega_0}^2 \xi_\omega - 2\lambda\omega_0\psi_\omega + K_\omega\omega_\omega \tag{22b}$$
$$\dot{\psi} = r \tag{22c}$$
$$\dot{r} = \frac{1}{T}r + \frac{K}{T}(\delta - b) \tag{22d}$$
$$\dot{b} = \omega_b \tag{22e}$$
$$y = \psi + \psi_\omega + v \tag{22f}$$

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

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix}$$

$$E = \begin{bmatrix} 0 & 0 \\ K_w & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

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

4.1 System observability without disturbances

$$x = \begin{bmatrix} \psi \\ r \end{bmatrix} \tag{23}$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{24}$$

The observability command in `MATLAB` gives us the following observability-matrix.

$$\mathcal{O} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{25}$$

The observability-matrix has full rank, hence the system is observable.

4.2 System observability with the current disturbance

$$x = \begin{bmatrix} \psi \\ r \\ b \end{bmatrix} \tag{26}$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{27}$$

The observability command in `MATLAB` gives us the following observability-matrix.

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.0138 & -0.0022 \end{bmatrix} \tag{28}$$

The observability-matrix has full rank, hence the system is observable.

4.3 System observability with the wave disturbance

$$x = \begin{bmatrix} \xi_\omega \\ \psi_\omega \\ \psi \\ r \end{bmatrix} \tag{29}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -{\omega_0}^2 & -2\lambda\omega_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \tag{30}$$

The observability command in `MATLAB` gives us the following observability-matrix.

$$\mathcal{O} = \begin{bmatrix} 0 & 1.0000 & 1.0000 & 0 \\ -0.6120 & -0.1330 & 0 & 1.0000 \\ 0.0814 & -0.5944 & 0 & -0.0138 \\ 0.3638 & 0.1604 & 0 & 0.0002 \end{bmatrix} \tag{31}$$

The observability-matrix has full rank, hence the system is observable.

4.4 System observability both current- and wave disturbances

$$x = \begin{bmatrix} \psi \\ r \end{bmatrix}, \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{32}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -{\omega_0}^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{33}$$

The observability command in `MATLAB` gives us the following observability-matrix.

$$\mathcal{O} = \begin{bmatrix} 0 & 1.0000 & 1.0000 & 0 & 0 \\ -0.6120 & -0.1330 & 0 & 1.0000 & 0 \\ 0.0814 & -0.5944 & 0 & -0.0138 & -0.0022 \\ 0.3638 & 0.1604 & 0 & 0.0002 & 0.0000 \\ -0.0982 & 0.3424 & 0 & -0.0000 & -0.0000 \end{bmatrix} \tag{34}$$

The observability-matrix has full rank, hence the system is observable.

5 Discrete Kalman filter

5.1 Discretization

Discretizing model from part 4 by using the following Matlab script:

```
%% Discretization
f = 10;
T_s = 1/f;
[A_d, B_d] = c2d(A,B,T_s);
[A_d, E_d] = c2d(A,E,T_s);
C_d = C;
D_d = D;
```

After using the script, we get:

$$A_d = \begin{bmatrix} 0.997 & 0.0992 & 0 & 0 & 0 \\ -0.0607 & 0.9838 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & 0 \\ 0 & 0 & 0 & 0.9986 & -0.0002 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; B_d = \begin{bmatrix} 0 \\ 0 \\ 0.0108 \\ 0.2152 \\ 0 \end{bmatrix} \tag{35a}$$

$$C_d = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \; D_d = 0, \; E_d = \begin{bmatrix} 0 & 0 \\ 0.0005 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{35b}$$

5.2 Variance

Finding estimate of the variance using Matlab command:

```
sim('measurement_noise');
variance = var(measurement_noise(:,2).*pi/180);
```

The command is using the simulation of measurement noise, and calculating the variance based on the simulation.

$$\sigma^2 = R = 6.1614 \cdot 10^{-7} \tag{36}$$

5.3 Kalman filter algorithm

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

$$P_0^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.013 & 0 & 0 & 0 \\ 0 & 0 & \pi^2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2.5 \cdot 10^{-3} \end{bmatrix}, \quad x_0^- = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We made our Kalman algorithm to be a function inside a Simulink block. The inputs of our function are the control variable u and measurement y. Outputs of our function are the posteriori estimate of compass course ψ and bias to the rudder angle b.

The algorithm contains a init flag, that is an empty variable. This flag is used to initialize start variables, like P_0^- and x_0^-, before the algorithm can run. After the first run, the algorithm keeps estimating values based on previous measurements.

```
function [psi, b] = kalman_func(u, y, m)

persistent init_flag A B C D E Q R P_minus x_minus I

if isempty(init_flag)
init_flag=1;

A = m.A;
B = m.B;
C = m.C;
D = m.D;
E = m.E;
Q = m.Q;
R = m.R;
P_minus = m.P_0;
x_minus = m.x_0;
I = m.I;
end

L = P_minus*C'*(C*P_minus*C' + R)^-1;
P = (I - L*C)*P_minus*(I - L*C)' + L*R*L';
x = x_minus + L*(y - C*x_minus);
x_minus = A*x + B*u;
P_minus = A*P*A' + E*Q*E';

psi = x(3);
b = x(5);
end
```

The physical connection of the simulink block containing Kalman filter algorithm is shown in Appendix B.8. It can be seen that there are two ZOH blocks connected to the Kalman filter input. The reason is that the system input and system measurement are in continius time, and has to be discretized to be used in a discrete Kalman filter. Our sampling frequency is $f_s = 10Hz$.

5.4 Simulation with the current disturbance

In Section 3.3, we were not able to follow the reference angle perfectly. The reason was that we had an state error of approximately 3 degrees. Our Kalman filter is designed to make an estimate of the bias. If the estimate of our filter is proper, we should be able to add it to the controller input signal, and simply just cancel out the previous error.

Like in Section 3.3, we are using the reference angle $\psi_r = 30°$. Our sampling time is $T_s = 0.1sec$. Simulating the model from Appendix B.8, we get following measurements:

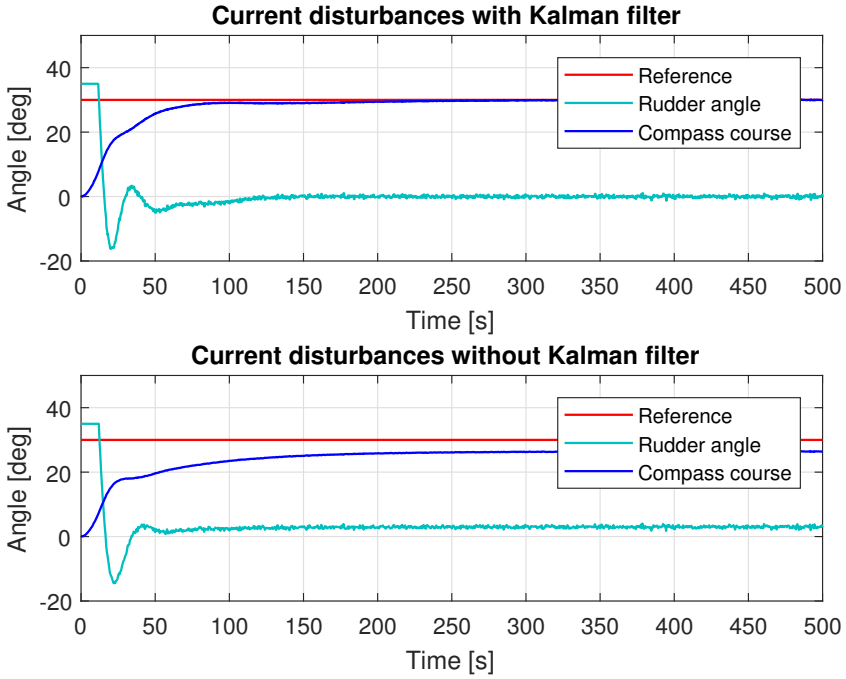


Figure 10: Response of system with current disturbance with and without Kalman filter

Looking at the figure 10 it easy to notice that the state error from Section 3.3 is canceled out, which means that the Kalman filter is estimating proper bias.

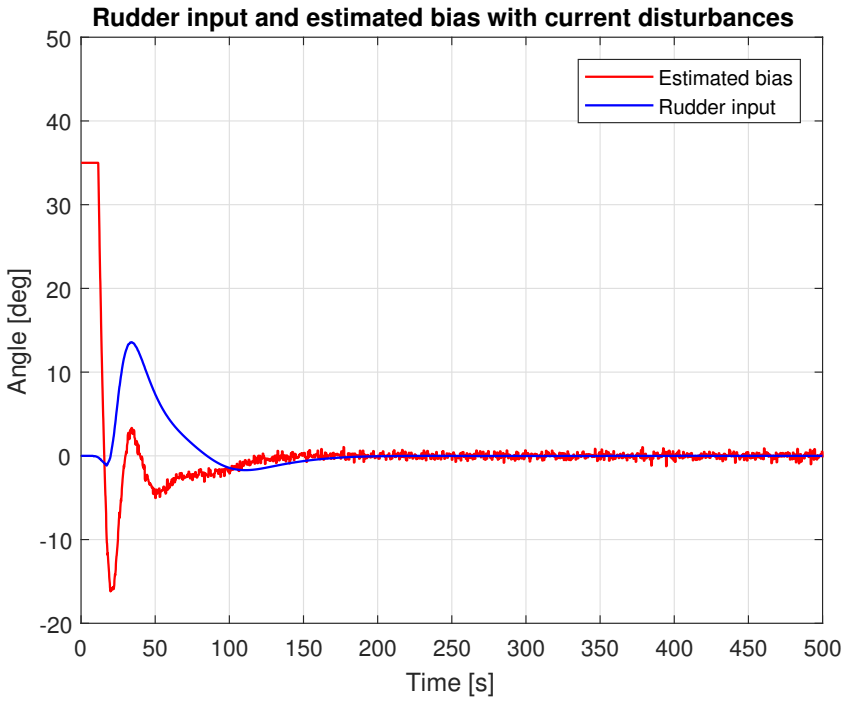


Figure 11: Rudder angle and estimated bias with current disturbances

It is shown in figure 11 that estimated bias opposes the rudder input. By adding the estimated bias to the rudder input in a feed forward loop, we are trying to cancel out the state error from Section 3.3. We can see that both rudder bias and rudder input is zero after approximately 200 seconds. Comparing with figure 10, we cant notice that 200 seconds is the time where the state error gets cancelled out. That is exactly the purpose with Kalman filter in this situation. It estimates a bias that cancels out the state error. Once the state error gets cancelled out, the estimated bias for Kalman filter gets equal to zero.

5.5 Filtered ψ instead of measured heading in the autopilot

In this task we will use a system with both current and wave disturbance.

Waves changes quickly, which means that wave disturbances has high frequency. High frequency disturbances affects our system in the way that our measured compass course changes with the same frequency (Figure 9). As our system is connected in a feedback loop, the error in our system will also have the same high frequency as wave disturbances and measured angle.
The whole idea of using Kalman filter in this case is to estimate an stable compass course that can result into a stable error form the feedback loop, and make the whole system more stable in general. Kalman filter is making the estimate of the compass course by taking average of the high frequency measured compass course signal. Figure 12 shows a comparison between estimated and measured compass course.

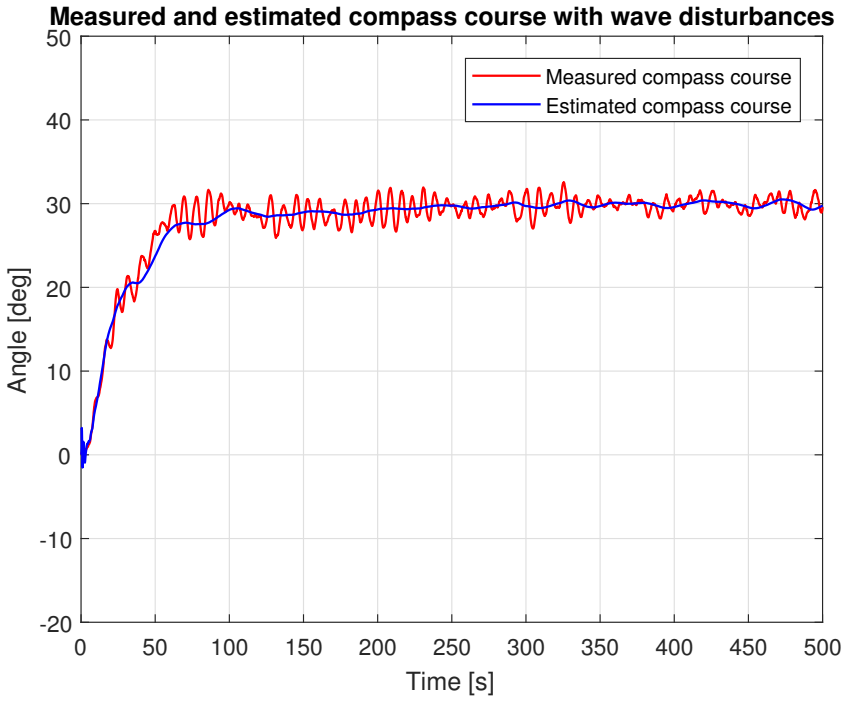


Figure 12: Measured and estimated compass course with wave disturbances

Since we are using both current and wave disturbance in this task, we also have to estimate an bias and connect it to a feed-forward loop to cancel out the state error. The high frequency of measured compass course, caused by wave disturbances, affects the bias signal as well. We are using the Kalman filter to make a stable estimate of the system bias. Figure 13 shows a comparison between estimated and measured bias.

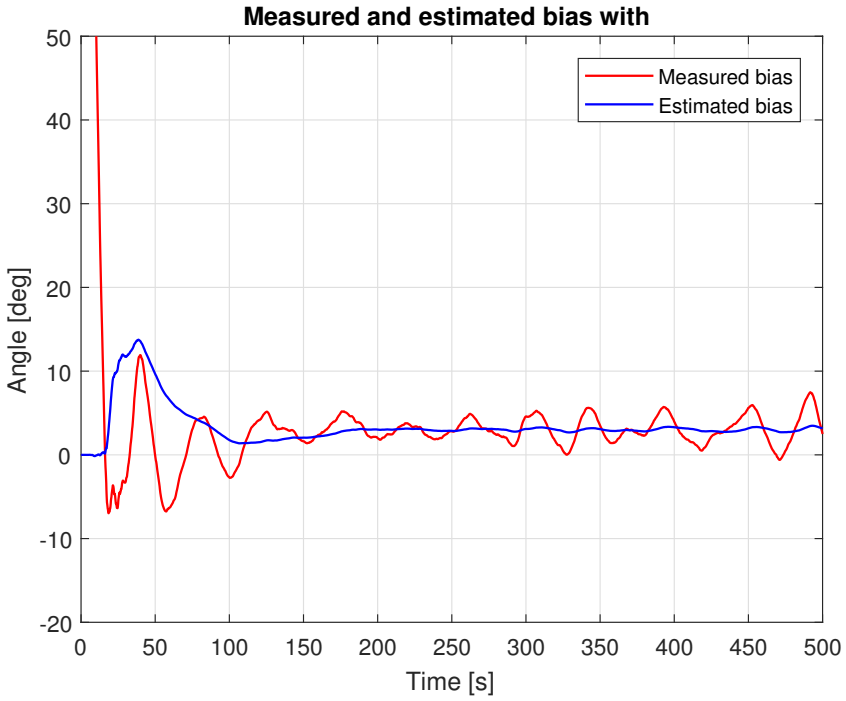


Figure 13: Measured and estimated bias

To compare estimated and actual wave influence, we had to turn off all other except the ”Cargo ship” and Kalman filter blocks. We also had to change output at our Kalman filter function from estimated compass angle to estimated wave influence on the compass angle. The change can be seen in Appendix B.10.

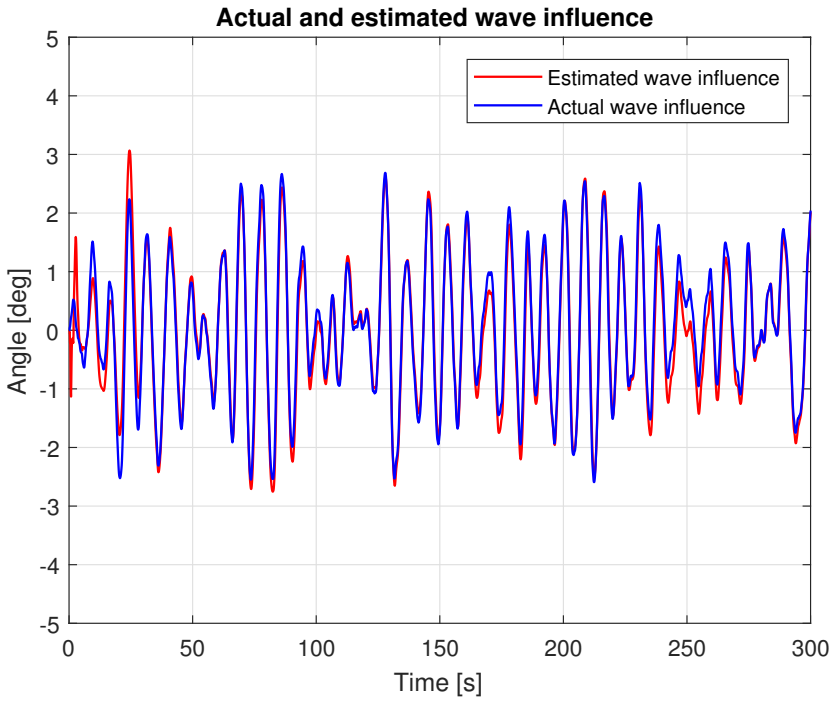


Figure 14: Comparison between estimated and actual wave disturbances

It can be seen in Figure 14 that values of estimated and actual wave disturbances are almost equal. Than leads us to the conclusion that the estimation of wave disturbance done by our Kalman filter is satisfying.

6 Conclusion

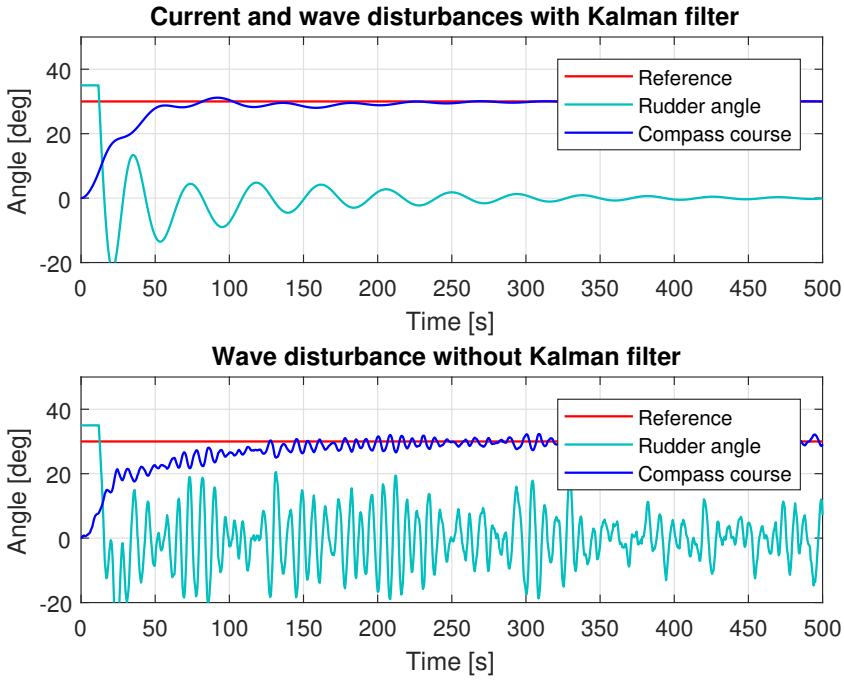


Figure 15: Response of the system with wave and current disturbances using Kalman filter compared with the system with wave disturbances without using Kalman filter

Figure 15 shows how the whole system gets improved by using a Kalman filter to estimate desired values. Signals affected by wave disturbance has high frequency, and are difficult to use in a control system. It is easy to notice from the simulations that Kalman filter does a excellent job in suppressing noise signals. Because of its ability to suppress noise, Kalman filter is a very good tool in ship control, where disturbances occurs all the time.

A MATLAB Code

This section contains all the MATLAB-code used in this lab assignment.

A.1 Part 1.2.1

```
clear all
close all
clc

%% Loading previously generated data
load('boat_param1_sine');     %frequency 0.005
load('boat_param2_sine');     %frequency 0.05

%% Plotting
figure
subplot(2,1,1)
plot(boat_param1(:,1), boat_param1(:,2), 'LineWidth', 1); grid on;
title('Sine input with \omega_1 = 0.005 and zero disturbances')
xlabel('Time [s]')
ylabel('sin(\omega_1t) [deg]')

subplot(2,1,2)
plot(boat_param2(:,1), boat_param2(:,2), 'LineWidth', 1); grid on;
title('Sine input with \omega_2 = 0.05 and zero disturbances')
xlabel('Time [s]')
ylabel('sin(\omega_2t) [deg]')
```

A.2 Part 1.2.2

```
clear all
close all
clc

%% Loading previously generated data
load('boat_param1_sine_noise');     %frequency 0.005
load('boat_param2_sine_noise');     %frequency 0.05

%% Plotting
figure
subplot(2,1,1)
plot(boat_param1_noise(:,1), boat_param1_noise(:,2), 'LineWidth', 1);
    grid on;
title('Sine input with \omega_1 = 0.005, waves and noise turned on')
xlabel('Time [s]')
ylabel('sin(\omega_1t) [deg]')

subplot(2,1,2)
plot(boat_param2_noise(:,1), boat_param2_noise(:,2), 'LineWidth', 1);
    grid on;
title('Sine input with \omega_2 = 0.05, waves and noise turned on')
xlabel('Time [s]')
ylabel('sin(\omega_2t) [deg]')
```

A.3 Part 1.3

```
clear all
close all
clc

%% Variables
w1 = 0.005;
w2 = 0.05;
A1 = 29.359;
A2 = 0.831;
A1_noise = 29.500;
A2_noise = 1.50;

% Boat parameters
T = sqrt(((A2*w2)^2-(A1*w1)^2)/(A1^2*w1^4-A2^2*w2^4));
K = A1*w1*sqrt(T^2*w1^2+1);
T_noise = sqrt(((A2_noise*w2)^2-(A1_noise*w1)^2)/(A1_noise^2*w1^4-
    A2_noise^2*w2^4));
K_noise = A1_noise*w1*sqrt(T_noise^2*w1^2+1);

%% Running simulink file
sim('boat_param_step')

%% Plotting comparison between model and ship
figure
plot(boat_param_step_ship(:,1), boat_param_step_ship(:,2), 'LineWidth'
    , 1); hold on;
plot(boat_param_step_model(:,1), boat_param_step_model(:,2), '
    LineWidth', 1); hold on;
plot(boat_param_step_model_noise(:,1), boat_param_step_model_noise
    (:,2),'--','LineWidth', 1); grid on;
xlabel('Time [s]')
ylabel('Amplitude [deg]')
legend('Ship','Model','Model with noise','Location','Northwest')
title('Comparing step respons of ship with step response of model')
```

A.4 Part 2

```
clear all
close all
clc

%% PSD estimation
load('wave.mat');

fs = 10;
window = 4096;
noverlap = [];
nfft = [];

[est_psd, f] = pwelch(psi_w(2,:).*(pi/180),window,noverlap,nfft,fs);


est_psd = est_psd*(1/(2*pi));   %converting pxx [power per Hz] to pxx
    [(power*sec)/rad]
omega = f.*(2*pi);              %converting f [Hz] to f [rad/s]

[maxValPSD, indexMaxValPSD] = max(est_psd);
omega_0 = omega(indexMaxValPSD);
sigma = sqrt(maxValPSD);

```

```
lambda = 0.045:0.020:0.125;
K_w = 2*lambda.*omega_0*sigma;
psd = (omega.*K_w).^2./(omega.^4 + omega_0^4 + 2*omega_0^2*omega
    .^2*(2*lambda.^2-1));

%% Plotting
%Finding omega_0 and sigma from this plot
figure
plot(omega,est_psd, 'LineWidth',1); grid on;
title('Estimate of Power Spectral Density (PSD)');
xlabel('\omega [rad/s]');
ylabel('Power');
axis([0 2 0 1.6e-3]);

%Finding appropriate lambda value
figure
subplot(2,1,1)
plot(omega,est_psd,'Linewidth',1); hold on;
plot(omega,psd,'--'); grid
title('Finding \lambda by trial and error')
xlabel('\omega [rad/s]');
ylabel('Power');
legend('Estimate of PSD','Analytical PSD with \lambda = 0.045','
    Analytical PSD with \lambda = 0.065',...
    'Analytical PSD with \lambda = 0.085','Analytical PSD with \lambda
        = 0.105','Analytical PSD with \lambda = 0.125');
axis([0 2 0 1.6e-3]);

%A closer look at previous plot
subplot(2,1,2)
plot(omega,est_psd,'Linewidth',1); hold on;
plot(omega,psd,'--'); grid on;
xlabel('\omega [rad/s]');
ylabel('Power');
axis([0.55 1.05 0 1.6e-3]);
```

A.5 Part 3.2

```
clear all
close all
clc

%% Ship model
T = 72.442;
K = 0.156;

%% PD-controller
w_c = 0.1;
T_d = T;
T_f = 1/(w_c * tan(-130*(pi/180)));
K_pd = sqrt(w_c^2 + T_f^2 * w_c^4)/K;

h_0 = tf(K*K_pd, [T_f 1 0]);

%% Simulating
sim('p5p3b');

%% Plotting
figure
plot(PD_measurement_noise(:,1),PD_measurement_noise(:,2),'r','
    LineWidth', 1); hold on;
plot(PD_measurement_noise(:,1),PD_measurement_noise(:,3),'Color',[0
```

```
    0.75 0.75],'LineWidth', 1); hold on;
plot(PD_measurement_noise(:,1),PD_measurement_noise(:,4),'b','
    LineWidth', 1); grid on;
title('Measurement noise with PD-regulator');
xlabel('Time [s]');
ylabel('Angle [deg]');
legend('Reference','Rudder angle','Compass course');
axis([0 500 -20 50]);
```

A.6 Part 3.3

```
clear all
close all
clc

%% Ship model
T = 72.442;
K = 0.156;

%% PD-controller
w_c = 0.1;
T_d = T;
T_f = 1/(w_c * tan(-130*(pi/180)));
K_pd = sqrt(w_c^2 + T_f^2 * w_c^4)/K;

h_0 = tf(K*K_pd, [T_f 1 0]);

%% Simulating
sim('p5p3c');

%% Plotting
figure
plot(PD_current_disturbances(:,1),PD_current_disturbances(:,2),'r','
    LineWidth', 1); hold on;
plot(PD_current_disturbances(:,1),PD_current_disturbances(:,3),'Color'
    ,[0 0.75 0.75],'LineWidth', 1); hold on;
plot(PD_current_disturbances(:,1),PD_current_disturbances(:,4),'b','
    LineWidth', 1); grid on;
title('Current disturbances with PD-regulator');
xlabel('Time [s]');
ylabel('Angle [deg]');     %??????
legend('Reference','Rudder angle','Compass course');
axis([0 500 -20 50]);
```

A.7 Part 3.4

```
clear all
close all
clc

%% Ship model
T = 72.442;
K = 0.156;

%% PD-controller
w_c = 0.1;
T_d = T;
T_f = 1/(w_c * tan(-130*(pi/180)));
K_pd = sqrt(w_c^2 + T_f^2 * w_c^4)/K;

h_0 = tf(K*K_pd, [T_f 1 0]);

%% Simulating
sim('p5p3d');

%% Plotting
figure
plot(PD_wave_disturbances(:,1),PD_wave_disturbances(:,2),'r','
    LineWidth', 1); hold on;
plot(PD_wave_disturbances(:,1),PD_wave_disturbances(:,3),'Color',[0
    0.75 0.75],'LineWidth', 1); hold on;
plot(PD_wave_disturbances(:,1),PD_wave_disturbances(:,4),'b','
    LineWidth', 1); grid on;
title('Wave disturbances with PD-regulator');
xlabel('Time [s]');
ylabel('Angle [deg]');
legend('Reference','Rudder angle','Compass course');
axis([0 500 -20 50]);
```

A.8 Part 5.3

```
clear all
close all
clc

%% PSD estimation
load('wave.mat');

fs = 10;
window = 4096;
noverlap = [];
nfft = [];

[est_psd, f] = pwelch(psi_w(2,:).*(pi/180),window,noverlap,nfft,fs);
       %psi_w is given in degrees (translates to radians)

est_psd = est_psd*(1/(2*pi));   %converting pxx [power per Hz] to pxx
    [(power*sec)/rad]
omega = f.*(2*pi);              %converting f [Hz] to f [rad/s]

[maxValPSD, indexMaxValPSD] = max(est_psd);
omega_0 = omega(indexMaxValPSD);
sigma = sqrt(maxValPSD);

%% Precalculated variables
T = 72.442;
```

```
K = 0.156;
lambda = 0.085;
K_w = 2*lambda*omega_0*sigma;

%% PD-controller
w_c = 0.1;
T_d = T;
T_f = 1/(w_c * tan(-130*(pi/180)));
K_pd = sqrt(w_c^2 + T_f^2 * w_c^4)/K;

%% System matrices
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];

%% Discretization
f = 10;     %frequency [Hz]
T_s = 1/f;  %sampling frequency (0.1 sec)

[A_d, B_d] = c2d(A,B,T_s);
[A_d, E_d] = c2d(A,E,T_s);
C_d = C;
D_d = D;

%% Estimating variance of the measurement noise
sim('measurement_noise');

variance = var(measurement_noise(:,2).*pi/180);

%% Kalman-filter variables
P_0_minus=[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_minus = [0; 0; 0; 0; 0];
Q = [30 0;0 1e-6];
I = eye(5);
R = (6.1614e-7)/T_s;

% struct for Kalman filter
m = struct('A', A_d, 'B', B_d, 'C', C_d, 'D', D, 'E', E_d,'Q' , Q, 'R'
    , R, 'P_0',P_0_minus, 'x_0', x_0_minus, 'I', I);
```

A.9 Part 5.4

```
clear all
close all
clc

%% PSD estimation
load('wave.mat');

fs = 10;
window = 4096;
noverlap = [];
nfft = [];

[est_psd, f] = pwelch(psi_w(2,:).*(pi/180),window,noverlap,nfft,fs);
       %psi_w is given in degrees (translates to radians)

est_psd = est_psd*(1/(2*pi));   %converting pxx [power per Hz] to pxx
    [(power*sec)/rad]
omega = f.*(2*pi);              %converting f [Hz] to f [rad/s]

[maxValPSD, indexMaxValPSD] = max(est_psd);
omega_0 = omega(indexMaxValPSD);
sigma = sqrt(maxValPSD);

%% Precalculated variables
T = 72.442;
K = 0.156;
lambda = 0.085;
K_w = 2*lambda*omega_0*sigma;

%% PD-controller
w_c = 0.1;
T_d = T;
T_f = 1/(w_c * tan(-130*(pi/180)));
K_pd = sqrt(w_c^2 + T_f^2 * w_c^4)/K;

%% System matrices
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];

%% Discretization
f = 10;     %frequency [Hz]
T_s = 1/f;  %sampling frequency (0.1 sec)

[A_d, B_d] = c2d(A,B,T_s);
[A_d, E_d] = c2d(A,E,T_s);
C_d = C;
D_d = D;

%% Kalman-filter variables
P_0_minus=[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_minus = [0; 0; 0; 0; 0];
Q = [30 0;0 1e-6];
I = eye(5);
R = (6.1614e-7)/T_s;

% struct for Kalman filter
m = struct('A', A_d, 'B', B_d, 'C', C_d, 'D', D, 'E', E_d,'Q' , Q, 'R'
    , R, 'P_0',P_0_minus, 'x_0', x_0_minus, 'I', I);

%% Simulating
sim('p5p5d');

%% Plotting
load('PD_current_disturbances.mat');

figure
subplot(2,1,1)
plot(kalman_current_disturbances(:,1),kalman_current_disturbances(:,2)
    , 'r','LineWidth',1); hold on;
plot(kalman_current_disturbances(:,1),kalman_current_disturbances(:,3)
    , 'Color',[0 0.75 0.75],'LineWidth',1); hold on;
plot(kalman_current_disturbances(:,1),kalman_current_disturbances(:,4)
    , 'b','LineWidth',1); grid on;
title('Current disturbances with Kalman filter');
xlabel('Time [s]');
ylabel('Angle [deg]');
legend('Reference','Rudder angle','Compass course');
axis([0 500 -20 50]);

subplot(2,1,2)
plot(PD_current_disturbances(:,1),PD_current_disturbances(:,2), 'r','
    LineWidth',1); hold on;
plot(PD_current_disturbances(:,1),PD_current_disturbances(:,3), 'Color
    ',[0 0.75 0.75],'LineWidth',1); hold on;
plot(PD_current_disturbances(:,1),PD_current_disturbances(:,4), 'b','
    LineWidth',1); grid on;
title('Current disturbances without Kalman filter');
xlabel('Time [s]');
ylabel('Angle [deg]');
legend('Reference','Rudder angle','Compass course');
axis([0 500 -20 50]);
```

A.10 Part 5.5

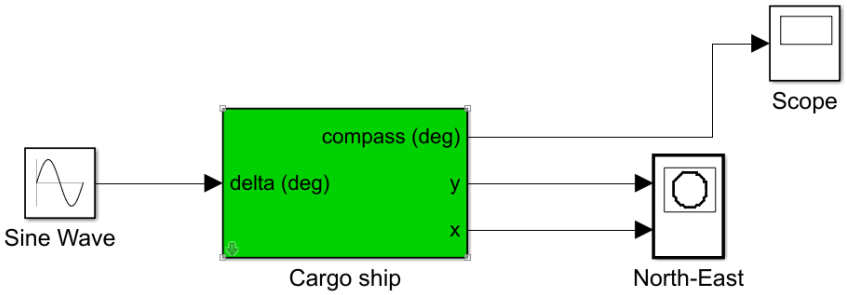
```
clear all
close all
clc

%% PSD estimation
load('wave.mat');

fs = 10;
window = 4096;
noverlap = [];
nfft = [];

[est_psd, f] = pwelch(psi_w(2,:).*(pi/180),window,noverlap,nfft,fs);
       %psi_w is given in degrees (translates to radians)

est_psd = est_psd*(1/(2*pi));   %converting pxx [power per Hz] to pxx
```

```
    [(power*sec)/rad]
omega = f.*(2*pi);      %converting f [Hz] to f [rad/s]

[maxValPSD, indexMaxValPSD] = max(est_psd);
omega_0 = omega(indexMaxValPSD);
sigma = sqrt(maxValPSD);

%% Precalculated variables
T = 72.442;
K = 0.156;
lambda = 0.087;
K_w = 2*lambda*omega_0*sigma;

%% PD-controller
w_c = 0.1;
T_d = T;
T_f = 1/(w_c * tan(-130*(pi/180)));
K_pd = sqrt(w_c^2 + T_f^2 * w_c^4)/K;

%% System matrices
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];

%% Discretization
f = 10;     %frequency [Hz]
T_s = 1/f;  %sampling frequency (0.1 sec)

[A_d, B_d] = c2d(A,B,T_s);
[A_d, E_d] = c2d(A,E,T_s);
C_d = C;
D_d = D;

%% Kalman-filter variables
P_0_minus=[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_minus = [0; 0; 0; 0; 0];
Q = [30 0;0 1e-6];
I = eye(5);
R = (6.1614e-7)/T_s;

% struct for Kalman filter
m = struct('A', A_d, 'B', B_d, 'C', C_d, 'D', D, 'E', E_d,'Q' , Q, 'R'
    , R, 'P_0',P_0_minus, 'x_0', x_0_minus, 'I', I);

%% Simulating
sim('p5p5e');

%% Plotting
load('PD_wave_disturbances.mat');

figure
```

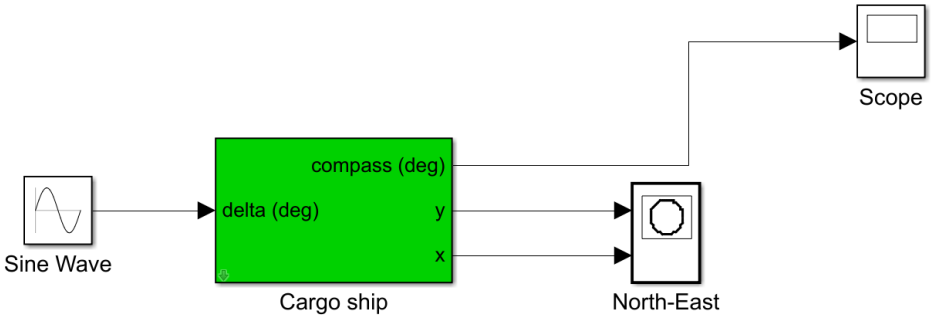
```
76  subplot(2,1,1)
77  plot(kalman_current_and_wave_disturbances(:,1),
        kalman_current_and_wave_disturbances(:,2), 'r','LineWidth',1);
        hold on;
78  plot(kalman_current_and_wave_disturbances(:,1),
        kalman_current_and_wave_disturbances(:,3), 'Color',[0 0.75 0.75],'
        LineWidth',1); hold on;
79  plot(kalman_current_and_wave_disturbances(:,1),
        kalman_current_and_wave_disturbances(:,4), 'b','LineWidth',1);
        grid on;
80  title('Current and wave disturbances with Kalman filter');
81  xlabel('Time [s]');
82  ylabel('Angle [deg]');
83  legend('Reference','Rudder angle','Compass course');
84  axis([0 500 -20 50]);
85
86  subplot(2,1,2)
87  plot(PD_wave_disturbances(:,1),PD_wave_disturbances(:,2), 'r','
        LineWidth',1); hold on;
88  plot(PD_wave_disturbances(:,1),PD_wave_disturbances(:,3), 'Color',[0
        0.75 0.75],'LineWidth',1); hold on;
89  plot(PD_wave_disturbances(:,1),PD_wave_disturbances(:,4), 'b','
        LineWidth',1); grid on;
90  title('Wave disturbance without Kalman filter');
91  xlabel('Time [s]');
92  ylabel('Angle [deg]');
93  legend('Reference','Rudder angle','Compass course');
94  axis([0 500 -20 50]);
95
96  figure
97  plot(task5_e_output(:,1),task5_e_output(:,2), 'r','LineWidth',1); hold
         on;
98  plot(task5_e_output(:,1),task5_e_output(:,3), 'b','LineWidth',1); grid
         on;
99  title('Measured and estimated compass course with wave disturbances');
100 xlabel('Time [s]');
101 ylabel('Angle [deg]');
102 legend('Measured compass course','Estimated compass course');
103 axis([0 500 -20 50]);
104
105 figure
106 plot(task5_e_bias_compare(:,1),task5_e_bias_compare(:,2), 'r','
        LineWidth',1); hold on;
107 plot(task5_e_bias_compare(:,1),task5_e_bias_compare(:,3), 'b','
        LineWidth',1); grid on;
108 title('Measured and estimated bias with');
109 xlabel('Time [s]');
110 ylabel('Angle [deg]');
111 legend('Measured bias','Estimated bias');
112 axis([0 500 -20 50]);
```

B Simulink Diagrams

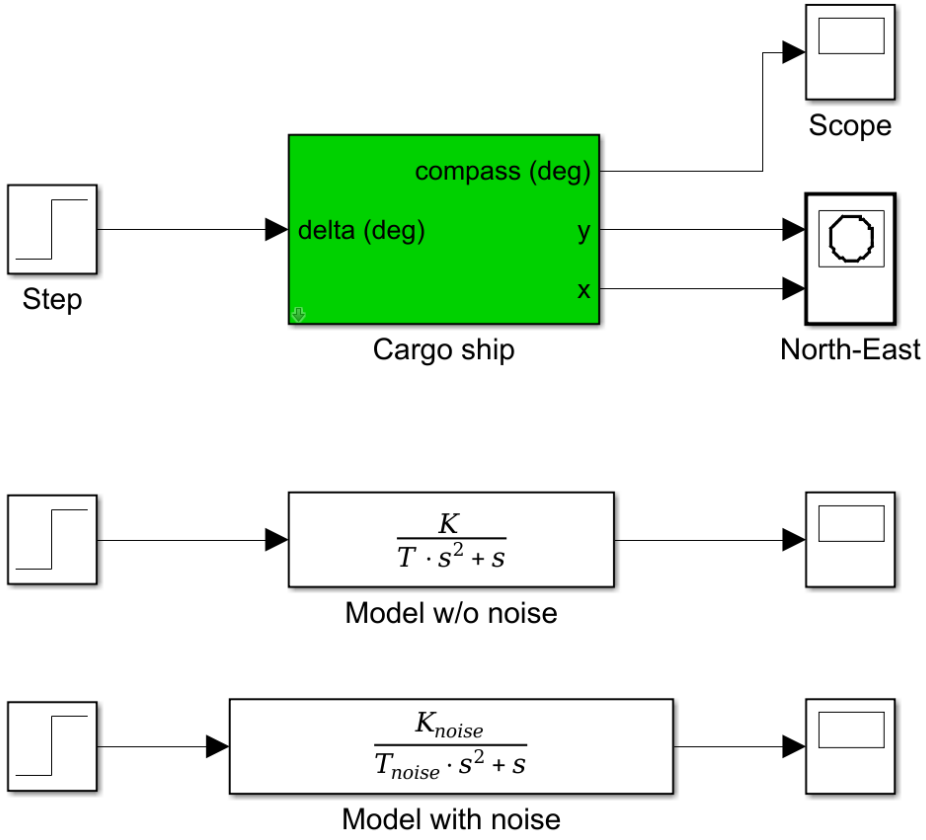
B.1 Part 1.2



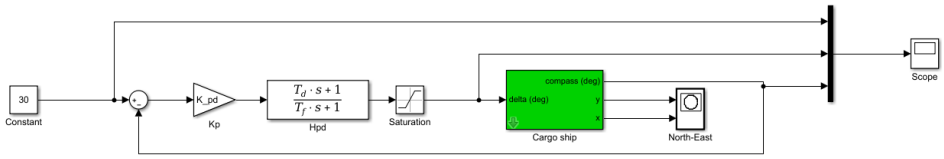
B.2 Part 1.3



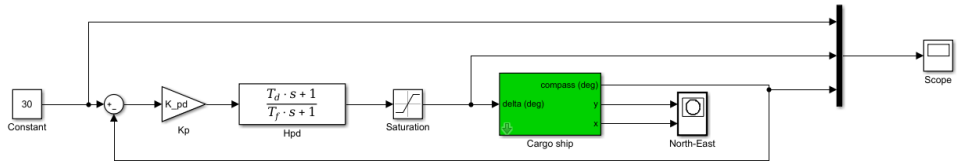
B.3 Part 1.4


Scope
compass (deg)
delta (deg)
y
x
Step
Cargo ship
North-East
$\frac{K}{T \cdot s^2 + s}$
Model w/o noise
$\frac{K_{noise}}{T_{noise} \cdot s^2 + s}$
Model with noise

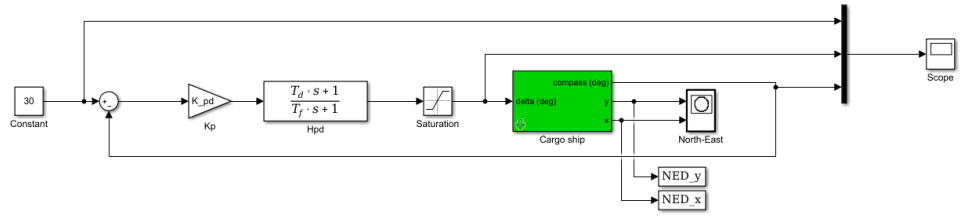
B.4 Part 3.2



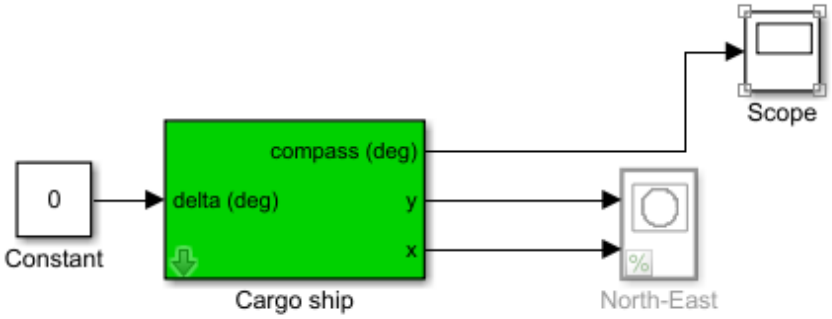
B.5 Part 3.3



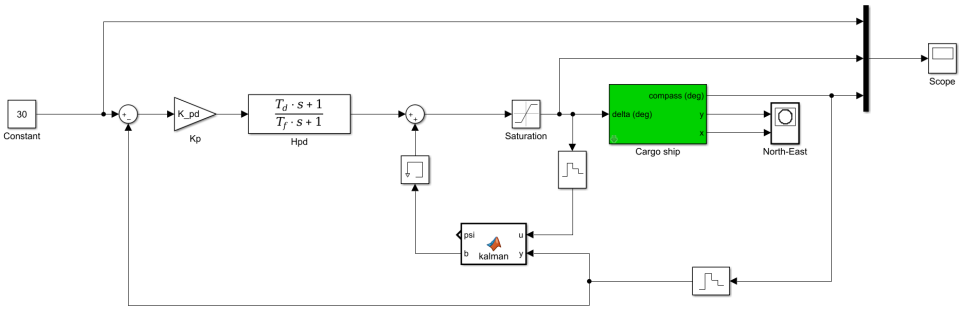
B.6 Part 3.4



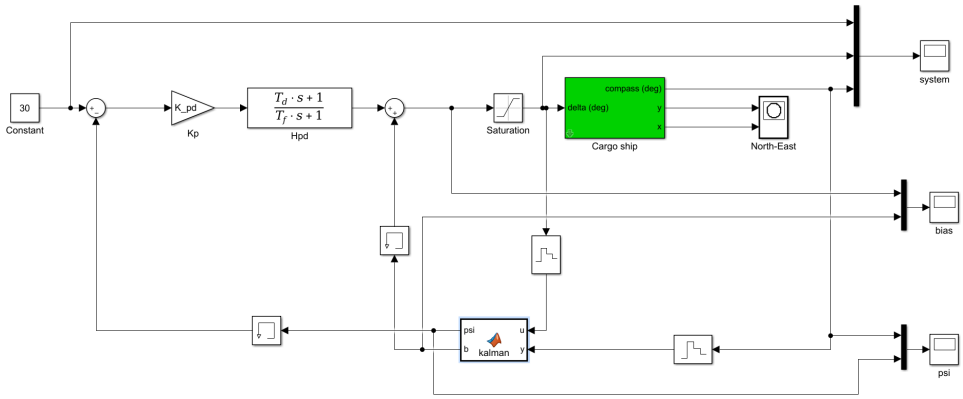
B.7 Part 5.2



B.8 Part 5.4



B.9 Part 5.5



B.10 Part 5.5 (Wave disturbance)