LABREPORT LINEAR SYSTEM THEORY

"Boat lab assignment"

Delivered to
Norwegian University of Science and Technology

Written by
Group 50

Magnus Bolstad
Mads-Emil Kvammen
Per Arne Kjelsvik

Faculty of Information Technology, Mathematics and Electrical Engineering
NTNU
Høyskoleringen 1, Trondheim - 7491

November 18, 2016

TTK4115

Contents

0 System **1**

1 Part I - Identification of the boat parameters **2**

1.a Transfer function from δ to ψ 2

1.b Boat parameters 2

1.c Waves and measurement noise 4

1.d Step response 5

2 Part II - Identification of wave spectrum model **6**

2.a Power spectral density 6

2.b Transfer function from ω_w to ψ_w 7

2.c Resonance frequency, ω_0 8

2.d Damping factor λ 9

3 Part III - Control system design **10**

3.a PD controller 10

3.b Simulation without disturbance 11

3.c Simulation with current disturbance 13

3.d Simulation with wave disturbance 13

4 Part IV - Observability **16**

4.a State space model 16

4.b Without disturbance 16

4.c Current disturbance 17

4.d Wave disturbance 18

4.e Both current and wave disturbance 18

5 Part V - Discrete Kalman filter **20**

5.a Discretization 20
5.b Variance of measurement noise 20
5.c Implementation of discrete Kalman filter 21
5.d Feed forward from estimated bias 23
5.e Wave filtering 25

6 Conclusion **29**

A Appendix A - MATLAB codes **31**

A.1 MATLAB - Part I 31
A.2 MATLAB - Part II 33
A.3 MATLAB - Part III 36
A.4 MATLAB - Part IV 38
A.5 MATLAB - Part V 40
A.6 Kalman MATLAB function 44

B Appendix B - SIMULINK models **45**

B.1 SIMULINK Task5.1.b 45
B.2 SIMULINK Task5.1.c 45
B.3 SIMULINK Task5.1.d 46
B.4 SIMULINK Task5.3.b 46
B.5 SIMULINK Task5.3.c 47
B.6 SIMULINK Task5.3.d 47
B.7 SIMULINK Task5.5.b 48
B.8 SIMULINK Task5.5.d 48
B.9 SIMULINK Task5.5.e 49

0 System

In this report, the ship is modeled using the following equations.

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

Equation 1: Equations used in the model

In this system ψ is the average heading of the ship, ψ_w is a high-frequency component in the heading due to wave disturbance, r is the rotation velocity about the z-axis, b is bias to the rudder angle, δ is the rudder angle relative to the BODY frame, w_b is white noise disturbance from the current and w_w is a zero mean white noise process with unity variance from the waves.

Furtermore, the system can be written as:

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

Equation 2: Equation for the system

where $\mathbf{x}$, u and $\mathbf{w}$ is given by:

$$\mathbf{x} = \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \\ b \end{bmatrix}, \quad u = \begin{bmatrix} \delta \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} w_w \\ w_b \end{bmatrix} \tag{3}$$

1 Part I - Identification of the boat parameters

1.a Transfer function from δ to ψ

In this problem, the transfer function from δ to ψ is calculated. Using the ship model and assuming that there is no effect of the current on the rudder, reveals:

$$\dot{\psi} = r \tag{4}$$

$$\ddot{\psi} = \dot{r} = -\frac{1}{T}r + \frac{K}{T}(\delta - b), \qquad b = 0 \tag{5}$$

$$\ddot{\psi} = -\frac{1}{T}\dot{\psi} + \frac{K}{T}\delta \tag{6}$$

$$\Downarrow \mathcal{L}\{\ddot{\psi}\}$$

$$\psi(s)s^2 = -\frac{1}{T}\psi(s)s + \frac{K}{T}\delta(s) \tag{7}$$

$$\psi(s)(s^2 + \frac{1}{T}s) = \frac{K}{T}\delta(s) \tag{8}$$

$$\mathbf{H(s)} = \frac{\boldsymbol{\psi(s)}}{\boldsymbol{\delta(s)}} = \frac{K}{s(Ts+1)} \tag{9}$$

Equation 9 is the transfer function parametrized by T and K, and it represents how the course will respond to a change in the rudder angle set point. In equation 5, we can see that $\ddot{\psi}$ has unit $[\frac{rad}{s^2}]$, which means that T must have unit $[s]$ and K must have unit $[\frac{1}{s}]$. δ and b are in $[deg]$, which can be converted to rad with the scalar product $\frac{\pi}{180}$.

1.b Boat parameters

In this task all the disturbances in the model are turned off in order to identify the boat parameters T and K. A sine input with amplitude 1 and given frequency is implemented in the `SIMULINK` model. To find both T and K, two input frequencies are required. Given frequencies are $\omega_1 = 0.005$ and $\omega_2 = 0.05$. The amplitude of the output signal for both frequencies will then be equal to $|H_1(j\omega_1)|$ and $|H_2(j\omega_2)|$.

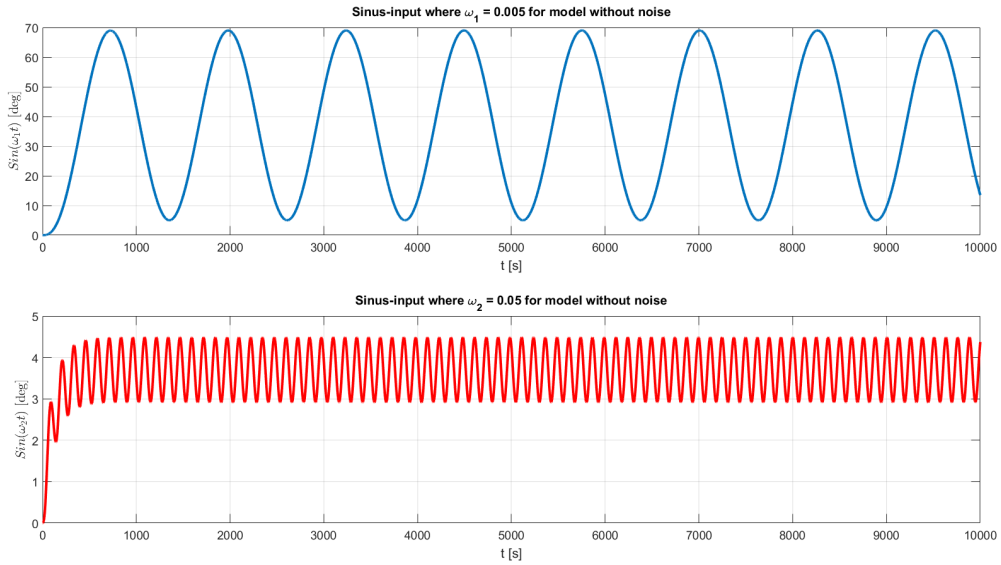

Figure 1: Simulation without disturbances, two different ω

Finding $|H_1(j\omega_1)|$ and $|H_2(j\omega_2)|$ using ω_1 and ω_2

Using the functions max and min in `MATLAB` for the output signal after steady-state is obtained reveals the maximum and minimum values for both frequencies. This is then further used in order to obtain the amplitude in both cases.

$$A_1 = |H_1(j\omega_1)| = \frac{68.96 - 5.022}{2} = 31.9787 \tag{10}$$

$$A_2 = |H_2(j\omega_2)| = \frac{4.485 - 2.916}{2} = 0.7847 \tag{11}$$

$$\begin{aligned} A_1 &= \left| \frac{K}{j\omega_1(Tj\omega_1 + 1)} \right| \\ &= \left| \frac{K}{-T\omega_1^2 + j\omega_1} \right| \\ &= \frac{K}{\sqrt{(-T\omega_1^2)^2 + \omega_1^2}} \\ &= \frac{K}{\omega_1\sqrt{T^2\omega_1^2 + 1}} \\ &\Downarrow \\ K &= A_1\omega_1\sqrt{T^2\omega_1^2 + 1} \end{aligned} \tag{12}$$

$$
\begin{aligned}
A_2 \Rightarrow K &= A_2\omega_2\sqrt{T^2\omega_2^2+1} \\
T^2\omega_2^2+1 &= \left(\frac{K}{A_2\omega_2}\right)^2 \\
T &= \sqrt{\frac{\left(\frac{K}{A_2\omega_2}\right)^2-1}{\omega_2^2}} \\
T &= \sqrt{\frac{K^2-A_2^2\omega_2^2}{\omega_2^2A_2^2\omega_2^2}} \\
T &= \frac{\sqrt{K^2-A_2^2\omega_2^2}}{A_2\omega_2^2}
\end{aligned} \tag{13}
$$

Finding values for K and T

Manipulating equation 12 and 13 and inserting the values found in 10 and 11 in order to find the values for T and K results in:

$$K = 0.1742\,\mathrm{s}^{-1} \tag{14}$$
$$T = 86.5256\,\mathrm{s} \tag{15}$$

1.c Waves and measurement noise

In this problem wave and measurement noise is turned on in the simulation. Using the same equations and approach as in section 1.b, new values for T and K are calculated.

New values for A_1 A_2 K T with disturbances

$$A_1 = 35.1054 \tag{16}$$
$$A_2 = 2.7057 \tag{17}$$
$$K = 0.1761\,\mathrm{s}^{-1} \tag{18}$$
$$T = 16.6751\,\mathrm{s} \tag{19}$$

It is difficult to find good new values for T and K using this approach due to the noise in the signal with lots of random spikes. As a result of this, it is not possible to get good estimates of the boat parameters, as seen in figure 2.

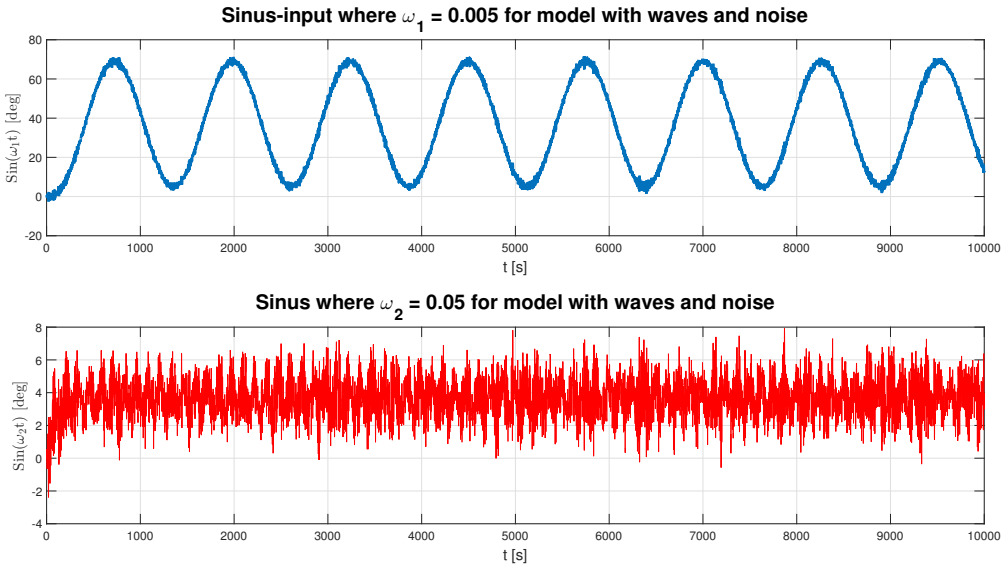

Figure 2: Simulation with disturbances, two different ω

1.d Step response

In this problem a step input of 1° to the rudder at t = 0 is applied to test the validity of the model. Comparing the step response of the model with the step response of the ship as shown i figure 3, reveals an offset in the response that grows with time. For small time periods, the model is good. The problem is when t gets larger, the model is more inaccurate. Anyway, this inaccuracy is not significant before after 2000-3000 seconds, which is an unrealistic long time to hold a constant rudder angle. Hence, the model is a good enough approximation for our purpose. The model could probably be improved further, by tuning the constants T and K, but by trial and error, no significant better results were obtained. Increasing K for instance, resulted in diverging path lines. Hence we decided to use the obtained values from the measurements in the rest of the assignment.

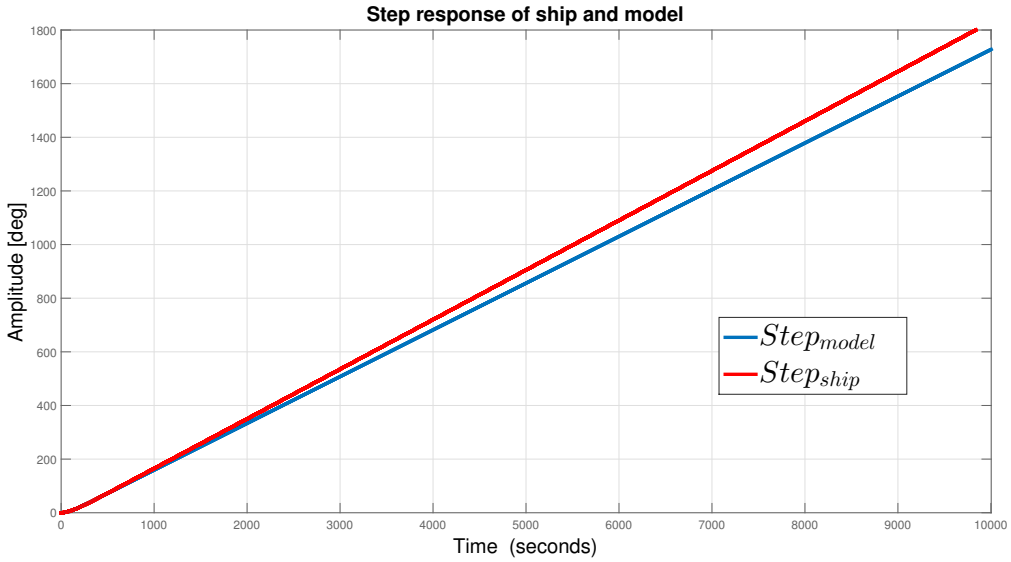

Figure 3: Step response of simulated boat and transfer function of boat model

2 Part II - Identification of wave spectrum model

2.a Power spectral density

I this problem, an estimate of the power spectral density function is to be found. In order to do so, the `MATLAB` function `pwelch` was used. This is a function that returns the PSD estimate of the input signal found using Welch's overlapped segment averaging estimator [3]. The PSD describes the distribution of power into frequency components. Figure 4 shows how the estimate of the PSD is in the model. While obtaining this plot and the estimate of the PSD, the input signal had to be modified so that the input was given in Hz and the output power was given in units power s/rad and the output frequency was given in rad/s. The scaling factors implemented to convert the outputs to the required units was $\frac{1}{2\pi}$ and 2π, respectively. This is the code for the estimated PSD:

Listing 1: `MATLAB` code for estimated psd

```
%% TASK 5.2.a ------ Estimate PSD
F_s = 10;
window = 4096;
noverlap = [];
nfft = [];

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

% \\\ f (Hz) to w (rad/s), and psd (power/pr. HZ) to psd (power s/rad)
omega = 2*pi.*f;
pxx = pxx./(2*pi);
```

2.b Transfer function from ω_w to ψ_w

Finding the transfer function of the wave response (from w_w to ψ_w) using equation 20 and 21:

$$\dot{\xi}_w = \psi_w \tag{20}$$
$$\dot{\psi}_w = -\omega_0^2 \xi - 2\lambda\omega_0\psi_w + K_w w_w \tag{21}$$

Taking the laplace transform on equation 20:

$$\begin{aligned} \xi_w(s) &= \psi_w(s) \\ &\Downarrow \\ \xi_w(s) &= \frac{\psi_w(s)}{s} \end{aligned} \tag{22}$$

Taking the laplace transform on equation 21:

$$s\psi_w(s) = -\omega_0^2\xi_w(s) - 2\lambda\omega_0\psi_w(s) + K_w w_w(s)$$

$$\Downarrow \text{ Inserting equation 22 to find } \frac{\psi_w}{w_w}$$

$$\begin{aligned} s\psi_w(s) &= -\omega_0^2\frac{\psi_w(s)}{s} - 2\lambda\omega_0\psi_w(s) + K_w w_w(s) \\ \psi_w(s)\left(s + 2\lambda\omega_0 + \frac{\omega_2^2}{s}\right) &= K_w w_w(s) \\ H_{w_w,\ \psi_w}(s) = \frac{\psi_w(s)}{w_w(s)} &= \frac{K_w}{s + 2\lambda\omega_0 + \frac{\omega_0^2}{s}} \\ &= \frac{sK_w}{s^2 + 2\lambda\omega_0 s + \omega_0^2} \end{aligned} \tag{23}$$

Finding an analytical expression for the Power Spectral Density function of ψ_w using the transfer function of the wave response model:

$$P_{\psi_w}(j\omega) = P_{w_w}(j\omega) \cdot |H_{w_w,\ \psi_w}(j\omega)|^2 \tag{24}$$

It is stated that w_w is a zero mean white noise process with unity variance. Using this and the example given in [4, p. 117], the PSD for w_w can be found.

$$P_{w_w}(j\omega) = \mathcal{F}\{\mathbf{R}_{w_w}(\tau)\} \tag{25}$$

$$\begin{aligned}\mathbf{R}_{w_w} &= A\delta(\tau) \qquad A = 1 \text{ at unity variance} \\ &= \delta(\tau) \qquad \text{where } \delta(\tau) \text{ is the dirac delta function}\end{aligned} \tag{26}$$

$$P_{w_w}(j\omega) = \mathcal{F}\{\delta(\tau)\} = 1 \tag{27}$$

$$\begin{aligned}\Rightarrow P_{\psi_w}(j\omega) &= 1 \cdot |H_{w_w,\,\psi_w}(j\omega)|^2 \\ &= H(j\omega) \cdot H(-j\omega) \\ &= \frac{j\omega K_w}{(j\omega)^2 + 2\lambda\omega_0(j\omega) + \omega_0^2} \cdot \frac{-j\omega K_w}{(-j\omega)^2 + 2\lambda\omega_0(-j\omega) + \omega_0^2} \\ &= \frac{(\omega K_w)^2}{\Big((\omega_0^2 - \omega^2) + j2\lambda\omega_0\omega\Big)\Big((\omega_0^2 - \omega^2) - j2\lambda\omega_0\omega\Big)} \\ &= \frac{(\omega K_w)^2}{(\omega_0^2 - \omega^2)^2 + (2\lambda\omega_0\omega)^2} \\ &= \frac{(\omega K_w)^2}{\omega^4 + \omega_0^4 + 2\omega_0^2\omega^2(2\lambda^2 - 1)}\end{aligned} \tag{28}$$

2.c Resonance frequency, ω_0

By investigating the plot of the Power Spectral Density in 2.a, the resonance frequency, ω_0, is found. The resonance frequency is the frequency where the Power Spectral Density has its maximum value. Hence, $\omega_0 = 0.7823$.

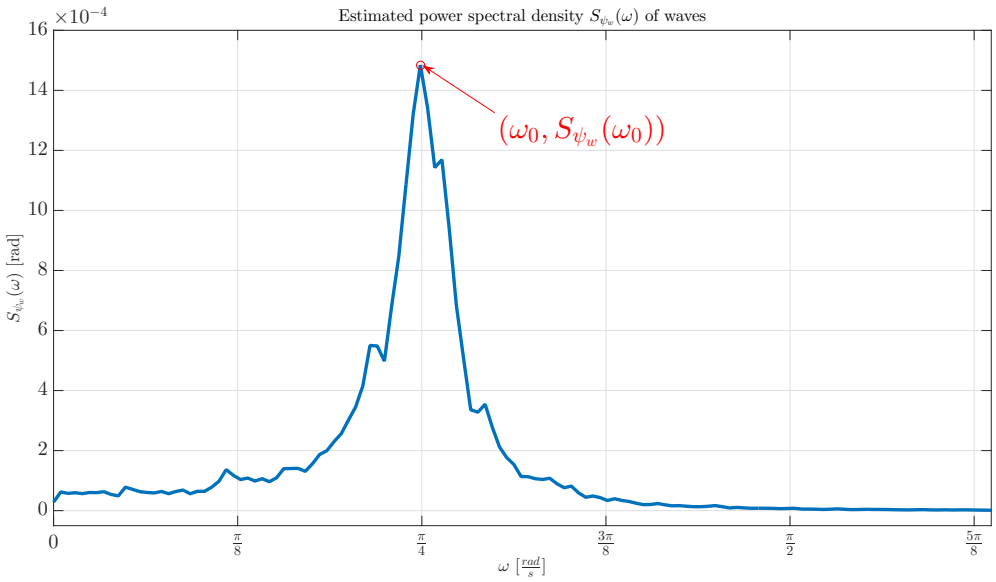

Figure 4: ω_0 was read from the plot and also computed in `MATLAB`

This value was also confirmed by finding it in MATLAB like this (see appendix A.2 for full code):

Listing 2: MATLAB code for ω_0

```
% Find resonance frequency from estimated PSD
[maxPSD, indexAtMaxPSD] = max(pxx);
omega_0 = omega(indexAtMaxPSD);
```

2.d Damping factor λ

To find a complete model for the wave response, the damping factor λ needs to be identified. To do so, K_w was defined as $2\lambda\omega_0\sigma$, where σ^2 is the peak value of $P_{\psi_w}(w)$. By choosing σ^2 as the peak of the estimated PSD, $S_{\psi_w}(w)$, the value could be set. This resulted in only λ having to be chosen in order to do the curve-fitting of $P_{\psi_w}(w)$ and $S_{\psi_w}(w)$. By trial and error in MATLAB, the value for λ was found to be 0.08 when the curves was the closest, as shown in figure 5. In the trial and error testing, closer values to 0.08 than shown in figure 5 was tested, although they are not displayed in the figure since that would make it hard to look at the difference between the high and low values of λ. See appendix A.2 for complete code for 5 and the more detailed plot that is not included.

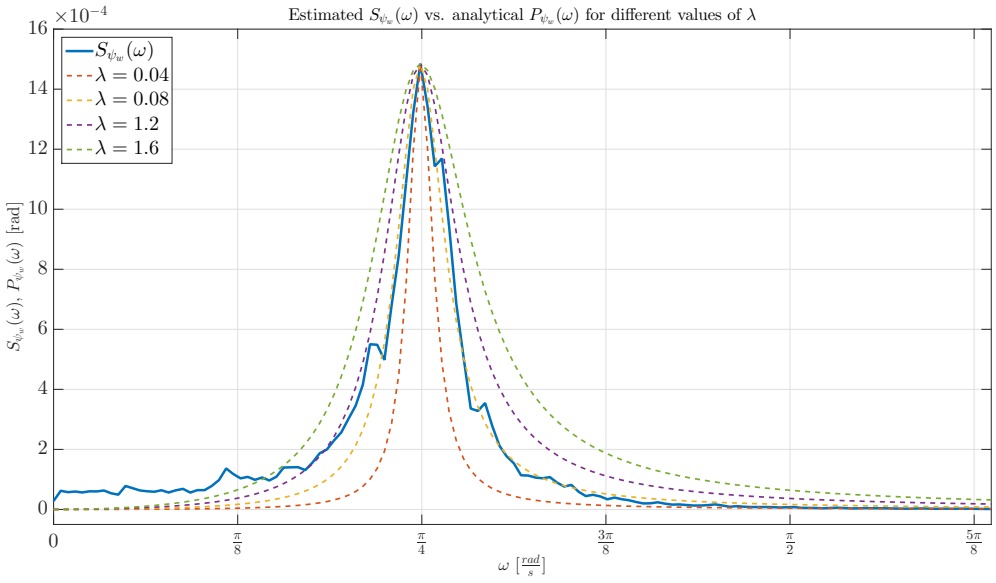

Figure 5: Plot of analytical PSD with different λ. $\lambda = 0.08$ was chosen

It was observed that when λ was set to a higher value, such as 0.8, the wave filtering performed later in the assignment,5.e, became more oscillating. As discussed in 5.e, this is not desirable.

3 Part III - Control system design

3.a PD controller

A PD-controller, $H_{pd} = K_{pd}\frac{1+T_d s}{1+T_f s}$, is to be designed from the transfer function 9. The derivative time constant, T_d, is chosen to be equal to the transfer function time constant, T in 15, such that they cancel. This gives the following transfer function for the open loop system:

$$h_0 = H_{pd}(s)H_{ship}(s) = \frac{KK_{pd}}{(1+T_f s)s} \tag{29}$$

To obtain the desired phase margin and cross frequency
$\varphi = 50$ and $\omega_c = 0.1$ and, the following equations were solved by looking at the angle of the transfer function:

$$\varphi = \angle h_0(j\omega_c) - (-180^\circ) \tag{30}$$
$$50^\circ - 180^\circ = \angle h_0(j\omega_c) \tag{31}$$
$$\angle \frac{KK_{pd}}{(1+T_f s)s} = -130^\circ \tag{32}$$
$$0 - \angle -\omega_c^2 T_f + j\omega_c = -130^\circ \tag{33}$$
$$tan^{-1}(\frac{\omega_c}{-\omega_c^2 T_f}) = 130^\circ \tag{34}$$
$$T_f = -\frac{1}{\omega_c tan(130^\circ)} \tag{35}$$
$$T_f = 8.39s \tag{36}$$

K_{pd} can then be found by

$$|h_0(j\omega_c)| = 1 \tag{37}$$
$$\sqrt{\frac{K^2 Kpd^2}{(1+T_f s)^2 s^2}} = 1 \tag{38}$$

Inserting $K = 0.1742$ into 38 gives:

$$K_{pd} = 0.7494 \tag{39}$$

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

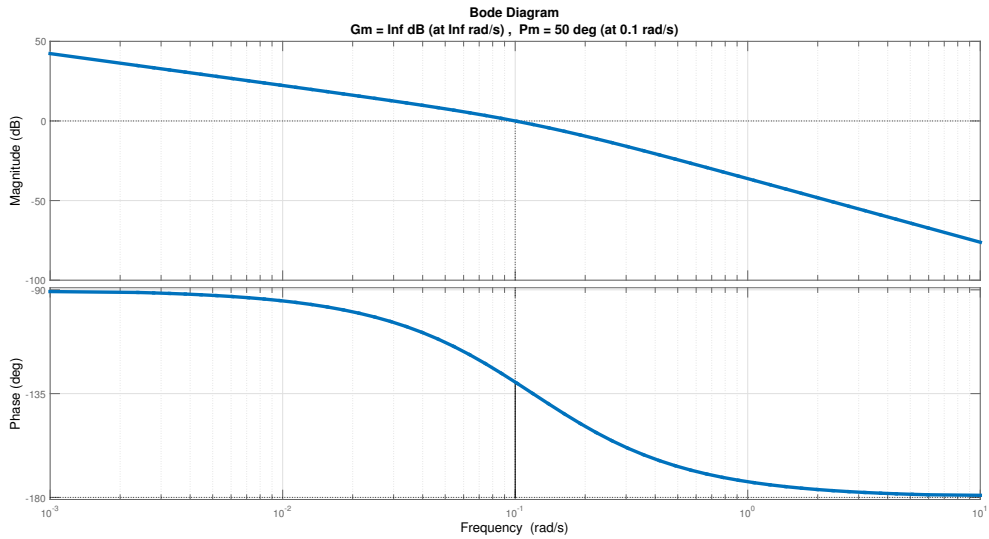

Figure 6: Bode plot of transfer function where $\omega_c = 0.1$ and $\varphi = 50°$ as desired

The T_f constant is what limits the derivative-effect. A higher T_f would for example limit the derivative-effect more than it currently does, which means that the controller would low-pass less frequencies, or rather, allow more frequencies to pass. This would in turn introduce oscillations in the response, and perhaps overshooting, while lowering T_f would make the response slower. Lowering T_d would perhaps make the response faster, since T_d is derivative-effect. However, T_f and T_d remains as defined throughout the rest of the assignment to meet requirements to phase margin φ and cross frequency ω_c.

3.b Simulation without disturbance

The implemented PD-controller gives the following compass course with a course reference of 30°, when no disturbances are present, only measurement noise. To make sure that the rudder set-point stayed within its limits of $\pm 35\,°$, a saturation block was added to the `SIMULINK` model for the rudder input δ. After approximately 320 seconds, the ship has obtained the reference course angle, which is an acceptable result for a large ship. This means that the autopilot works for smooth weather conditions. Figure 7 shows the autopilot with only measurement noise and no other disturbances. For the rest of the assignment, the symbol δ will be used for the rudder input angle, and not the actual rudder angle, which δ was interpreted as before.

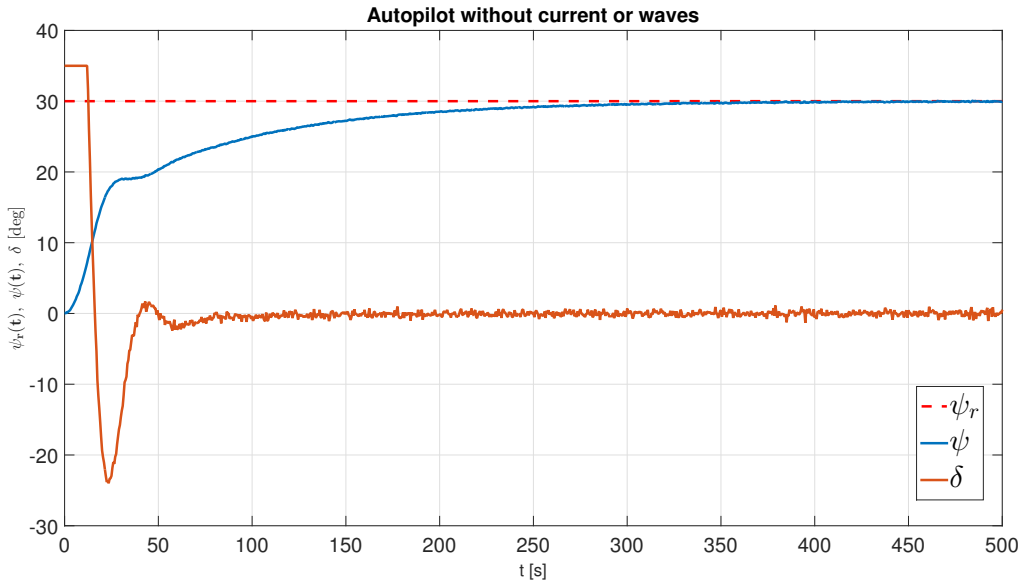

Figure 7: Autopilot without any disturbances

It is interesting to see how the rudder changes direction even before the heading has reached the reference. This is done by the derivative effect because the course is changing too fast, and otherwise it would result in an overshoot off the planned change in the course. This has to be done because of the momentum and the inertia of the ship.

The effect of removing the saturation block entirely was investigated. The rudder control started at an angel of 220°, which is unphysical, as a real rudder can only rotate ±90°.

3.c Simulation with current disturbance

The autopilot is now tested with current disturbance present. It is assumed that the only effect of the current is rudder angle bias. The course angle has the same shape as without the current disturbance, but will now have a steady state error as seen in figure 8. The rudder set-point will get a non-zero value, because of the error from the reference course angle, but the current bias will act in the opposite direction, such that the actual rudder angle will be zero, and the ship will head forward with a stationary error in the compass course. The rudder doesn't reach a high enough value to overcome the current, and turn the ship to its desired course. The ship will head approximately 3° off the desired 30°. This is not satisfactory, and further adjustment have to be made. This steady state error could have been removed by the integral effect in an PI or PID- controller. The PI or PID controller would have used some time to accumulate the error, and correct it, but on the other hand, a Kalman filter will spend some time finding the estimations by iteration. In the end of this assignment, Kalman filtering together with the PD controller is used instead, with sufficiently fast sampling time. Estimate of the current bias is used to cancel the current bias disturbance. The assumption that the current only will affect the rudder angle is a simplification, and in real, the hull of the ship would also be influenced.

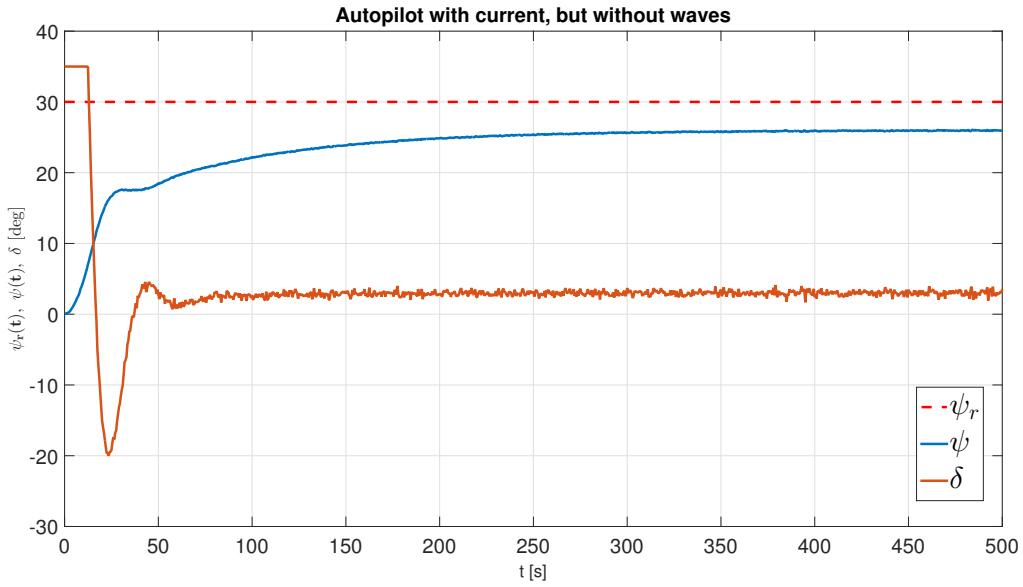

Figure 8: Autopilot with current disturbance

3.d Simulation with wave disturbance

The current disturbance is now turned off, and wave disturbance is included to the `SIMULINK` model. Figure 9 shows the ship in this setting. It can be seen from the figure that the ship reaches its heading. The problem here is that the rudder tries to counter the effect each wave has on the ship. The combination of the rudder changing rapidly and the rocking waves results in lots of fluctuations in the heading of the ship around its desired value.

The rapid changing of the rudder as a result of the waves is important to deal with. In the worst cases the rudder set-point changes by more than 40 degrees in about 4 seconds. This is an unrealistic high value for a large rudder, and by doing this over a period of time, the rudder will be destroyed. It's obvious that this is a matter of importance and needs to be fixed to get a desired behavior from the ship. To solve these problems, a Kalman Filter which uses a feed-forward loop to cancel out the offset from the current disturbance shown in section 3.c and a filtered feedback that smooths the rudder input, δ, so that the rudder doesn't fluctuate as much as shown in figure 9 is implemented, see section 5.

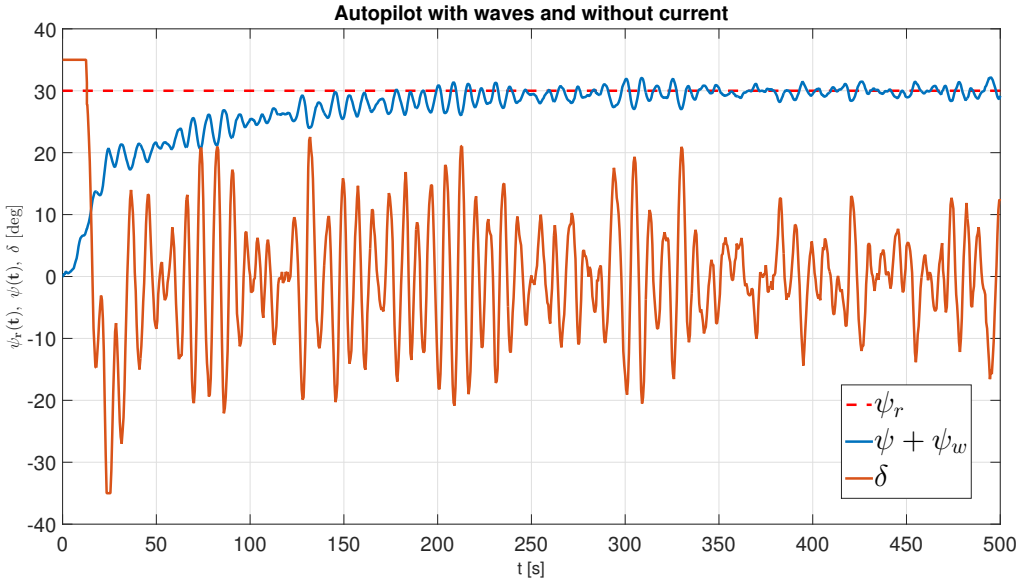

Figure 9: Autopilot with both current and wave disturbance

The rudder input under the different conditions previously discussed are summarized in fig. 10. Current causes the rudder input to have a stationary error, or offset while the waves causes the rudder input to change constantly to counteract the wave disturbance. In the situation with measurement noise only, the rudder input will converge to zero when the reference course is obtained. The effect of the measurement noise, which is present during all of the three conditions, can be seen as small , high frequent oscillations. This is easiest seen on the plot without external disturbance, and the one with current disturbance, after they have reached their steady state error.

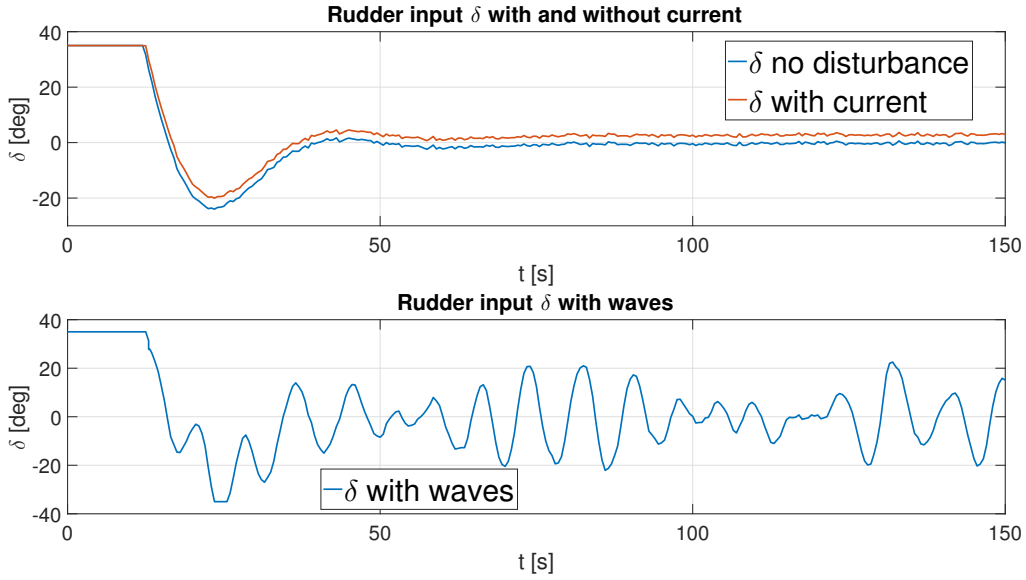

Figure 10: Rudder input in the three cases: No disturbances (except measurement noise), current disturbance and wave disturbance

4 Part IV - Observability

4.a State space model

Using the equations stated in 1 along with equation 2 and 3 in order to find the matrices A, B, C and E:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix}}_{=\mathbf{B}} u + \underbrace{\begin{bmatrix} 0 & 0 \\ K_w & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{=\mathbf{E}} \mathbf{w} \tag{40}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}}_{=\mathbf{C}} \mathbf{x} + v \tag{41}$$

4.b Without disturbance

When there are no disturbances in the system, $b = 0$ and all the wave disturbances are neglected. The state vector is then reduced to:

$$\mathbf{x} = \begin{bmatrix} \psi \\ r \end{bmatrix} \tag{42}$$

Without the disturbances, $\mathbf{w}$ is also removed from the system while u is the same. This gives the following system equation:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix}}_{=\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{K}{T} \end{bmatrix}}_{=\mathbf{B}} u \tag{43}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{=\mathbf{C}} \mathbf{x} + v \tag{44}$$

To find out whether the system is observable or not, the observability matrix is computed:

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{45}$$

[1, p. 197] The code for section 4 is shown in appendix A.4

This can easily be computed by using the `MATLAB` function obsv(A,C), where the A matrix and C matrix are input This gives the following observability matrix:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{46}$$

This matrix has rank equal two, which is full rank. Thus, the system is observable without disturbances.

4.c Current disturbance

In this problem only the current disturbance is included in the system. The wave disturbances is still neglected. This results in the following state vectors:

$$\mathbf{x} = \begin{bmatrix} \psi \\ r \\ b \end{bmatrix}, \; u = \delta, \; w = w_b \tag{47}$$

With the new state vectors the system can be reduced to:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{K}{T} \\ 0 \end{bmatrix}}_{=\mathbf{B}} u + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{=\mathbf{E}} w \tag{48}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{=\mathbf{C}} \mathbf{x} \; + \; v \tag{49}$$

The observability matrix with current disturbance becomes:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.0116 & -0.0020 \end{bmatrix} \tag{50}$$

This observability matrix has rank equal to three, which is also full rank. Thus the system is observable also with the current disturbance.

4.d Wave disturbance

In this problem the wave disturbance is included in the system. The current disturbances is then neglected. This results in the following state vectors:

$$\mathbf{x} = \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \end{bmatrix}, \; u = \delta, \; w = w_w \tag{51}$$

With the new state vectors the system can be reduced to:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix}}_{=\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \end{bmatrix}}_{=\mathbf{B}} u + \underbrace{\begin{bmatrix} 0 \\ K_w \\ 0 \\ 0 \end{bmatrix}}_{=\mathbf{E}} w \tag{52}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}}_{=\mathbf{C}} \mathbf{x} + v \tag{53}$$

The observability matrix with wave disturbance becomes:

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -0.6120 & -1.2517 & 0 & 1 \\ 0.7661 & 0.9548 & 0 & -0.0116 \\ -0.5844 & -0.4290 & 0 & 0.0001 \end{bmatrix} \tag{54}$$

This observability matrix has rank equal to four, which is also full rank. Thus the system is observable also when there is wave disturbances.

4.e Both current and wave disturbance

For this part, the observability for the complete system was to be investigated. This was investigated based on the system shown in equation 40 and 41 as well as the equation for computing the observability matrix shown in equation 45

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ -61.20 & -1.2517 & 0 & 1 & 0 \\ 0.7661 & 0.9548 & 0 & -0.0116 & -0.0020 \\ -0.5844 & -0.4290 & 0 & 0.0001 & 0 \\ 0.2626 & -0.0473 & 0 & 0 & 0 \end{bmatrix} \tag{55}$$

This observability matrix has rank equal to five, which is also full rank. Thus the system is observable with all disturbances.

In part 4 it has been observed that the system is observable in all cases. This means that for a measured output it is possible to determine the initial states as well as the behavior of the system. This is useful for the ship model, as it makes it possible to find where the ship has been and how it has behaved just from the measured outputs and the input during the time. The observability also make it possible to make use of estimators, which will be used in the next part of the assignment, to improve the PD controller made in section 3a.

5 Part V - Discrete Kalman filter

5.a Discretization

To improve the behavior of the ship, a Kalman filter is included to the PD controller. By including this, it's possible to make better estimates of the state variables that are affected by the noise in the system. In this part, exact discretization is used in the Kalman filter based on the model from 4.a. By driscretizing the model, the continuous model is transferred into a discrete counterpart. This is done by using the `MATLAB` function `c2d`, which converts a model from continuous to discrete time. The sampling frequency was given as 10 Hz which gives a sampling time of 0.1 s.

The `MATLAB` function `c2d` only allows two inputs from the state space equations. It is given that $\mathbf{C}_d = \mathbf{C}$ and $D_d = D$ [1, p. 110], which means that only $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{E}$ has to be discretized using the `c2d` function. This should be okay, since it is a linear time-invariant system, which means that the superposition property holds. Superposition says that the net response of two or more inputs at a give time and place is the same as the sum of the response caused by each input by itself. The result of this is that the `c2d` can be simply be run twice. The discretized matrices was found to be:

$$\mathbf{A}_d = \begin{bmatrix} 0.9970 & 0.0993 & 0 & 0 & 0 \\ -0.0608 & 0.9845 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & -0.0000 \\ 0 & 0 & 0 & 0.9988 & -0.0002 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \ \mathbf{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0.0000 \\ 0.0002 \\ 0 \end{bmatrix},$$

$$\mathbf{C}_d = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \ D_d = 0, \ \mathbf{E}_d = \begin{bmatrix} 0 & 0 \\ 0.0007 & 0 \\ 0 & -0.0000 \\ 0 & -0.0000 \\ 0 & 0.1 \end{bmatrix} \tag{56}$$

The `MATLAB` code for this is as follows:

Listing 3: Using `c2d`-function to compute discretized matrices

```
44 % \\\ Exact discretization
45 [Ad, Bd] = c2d(A,B,T_s);    Cd = C;
46 [Ad, Ed] = c2d(A,E,T_s);    Dd = D;
```

5.b Variance of measurement noise

To find an estimate of the variance of the measurement noise, the `MATLAB`-function **var** was used. This function returns the variance of a vector. In `SIMULINK`, the input δ is set to zero while the compass value is measured. As a result of this, only the measurement

noise on the compass is measured. This measurement is imported to `MATLAB` as a vector, and the **var**-function is run based on this. The resulting variance of the measurement noise, R, was:

$$\sigma^2 = 6.1614 \cdot 10^{-7} \tag{57}$$

5.c Implementation of discrete Kalman filter

The discrete Kalman filter is implemented by using `MATLAB`.

Notation:

A priori is denoted as $\hat{\mathbb{x}}^-[k]$

A posteriori is denoted as $\hat{\mathbb{x}}[k]$

The filter is initialized at:

$$\hat{\mathbf{x}}^-(0) = E[\mathbb{x}(0)] = \mathbf{m_{x_0}} \tag{58}$$

$$\begin{aligned}\mathbf{P}^-(0) &= E[\mathbb{e}^-(0)\mathbb{e}^-(0)^\mathbf{T}] \\ &= E[(\mathbb{x}(0) - \mathbf{m_{x_0}})(\mathbb{x}(0) - \mathbf{m_{x_0}})^\mathbf{T}] \\ &= \mathcal{C}_{\mathbf{x_0}}\end{aligned} \tag{59}$$

The stepwise solution implemented in `MATLAB` was:

$$L[k] = \mathbf{P}^-[k]\mathbf{C}^\mathbf{T}(\mathbf{C}\mathbf{P}^-[k]\mathbf{C}^\mathbf{T} + \bar{\mathbf{R}}_\mathbf{v})^{-1} \tag{60}$$

$$\Downarrow$$

$$\hat{\mathbb{x}}[k] = \hat{\mathbb{x}}^-[k] + \mathbf{L}[k](\mathbb{y}[k] - \mathbf{C}\hat{\mathbb{x}}^-[k]) \tag{61}$$

$$\Downarrow$$

$$\mathbf{P}[k] = (\mathbb{I} - \mathbf{L}[k]\mathbf{C})\mathbf{P}^-[k](\mathbb{I} - \mathbf{L}[k]\mathbf{C})^\mathbf{T} + \mathbf{L}[k]\bar{\mathbf{R}}_\mathbf{V}[k]\mathbf{L}[k]^\mathbf{T} \tag{62}$$

$$\Downarrow$$

$$\hat{\mathbb{x}}^-[k+1] = \bar{\mathbf{A}}\hat{\mathbb{x}}[k] + \bar{\mathbf{B}}\mathbf{u}[k] \tag{63}$$

$$\mathbf{P}^-[k+1] = \bar{\mathbf{A}}\mathbf{P}[k]\bar{\mathbf{A}}^\mathbf{T} + \bar{\mathbf{Q}}_\mathbf{W} \tag{64}$$

The initial a priori estimate error covariance and the initial a priori state estimate is given in the assignment [2]. This is implemented in a `MATLAB` function block in `SIMULINK`, with compase course and rudder angle as input, as shown in fig. 26 in appendix 5.d. The output is the estimated values, which are used as feedback and feedforward. The code for the `MATLAB` function block is as follows:

Listing 4: `MATLAB` function block Kalman algorithm

```
function [b,psi] = Kalman_matlab_fnc(u, y, data)
%#codegen
persistent init_flag A B C E Q R P_ x_ I

if (isempty(init_flag))
    init_flag = 1;

    % Initialization for system
    [A,B,C,E,Q,R,P_,x_, I] = deal(data.Ad,data.Bd,data.Cd,data.Ed, ...
        data.Q, data.R, data.P_0, data.X_0, data.I);
end

% 1 - Compute the Kalman Gain
    L = (P_*C')/((C*P_*C'+R));
% 2 - Update estimate with measurment
    x = x_ + L*(y-C*x_);
% 3 - Update error covariance matrix
    P = (I - L*C)*P_*(I-L*C)'+L*R*L';
% 4 - Projet ahead
    x_ = A*x + B*u;
    P_ = A*P*A' + E*Q*E';

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

end
```

The `MATLAB` function block was also edited to take in the variable `data` from the current workspace. In the `MATLAB` code for part 5 (found in appendix A.5) the variable `data` is defined like this:

Listing 5: Defining struct `data` for use in Kalman

```
% \\\ Q is the Process noise covariance
Q = [30 0; 0 10^(-6)];
P_0 = [1 0 0 0 0; 0 0.013 0 0 0; 0 0 pi^2 0 0; 0 0 0 1 0; ...
    0 0 0 0 2.5*10^-4];
X_0 = [0; 0; 0; 0; 0];
R = R/T_s;
I = diag([1 1 1 1 1]);

% \\\ Put data in a struct for use in the Kalman filter
data = struct('Ad',Ad,'Bd',Bd,'Cd',Cd,'Ed', Ed, 'Q',Q,'R', R,'P_0',P_0, ...
              'X_0',X_0, 'I', I);
```

The struct is then dealt to the persistent global variables of the `MATLAB` function block during first initialization where `init_flag` is an empty variable. The output of the function block is also changed to be of size 1 in both output ports. Also, a `Memory` block is applied to the output of the ship and the output of the boat, as in all other `SIMULINK` implementations, to avoid problems with algebraic loops. Lastly there is an zero order hold on the input of the Kalman filter as rudder angle and compass measurements are

continuous. The sample time on these blocks are also $0.1s$. The noise covariance is obtained through an averaging convention.

Estimates for values in the ship model, that are not available for measuring (such as the rudder bias from the current) are now available through the estimators. This is utilized in section 5.d-5.e of the assignment. The effect of changing the sampling frequency was also investigated, and it was concluded that a higher sampling frequency did not improve the results significantly, while a lower sampling frequency resulted in inaccurate measurements.

5.d Feed forward from estimated bias

The a posteriori estimated bias, found by the algorithm in 5.c, is now used in a feed forward to the rudder, such that the bias is cancelled. The reference course, ψ_r, is 30°. With a sample frequency of 10 Hz, the following behavior is obtained with current disturbance:

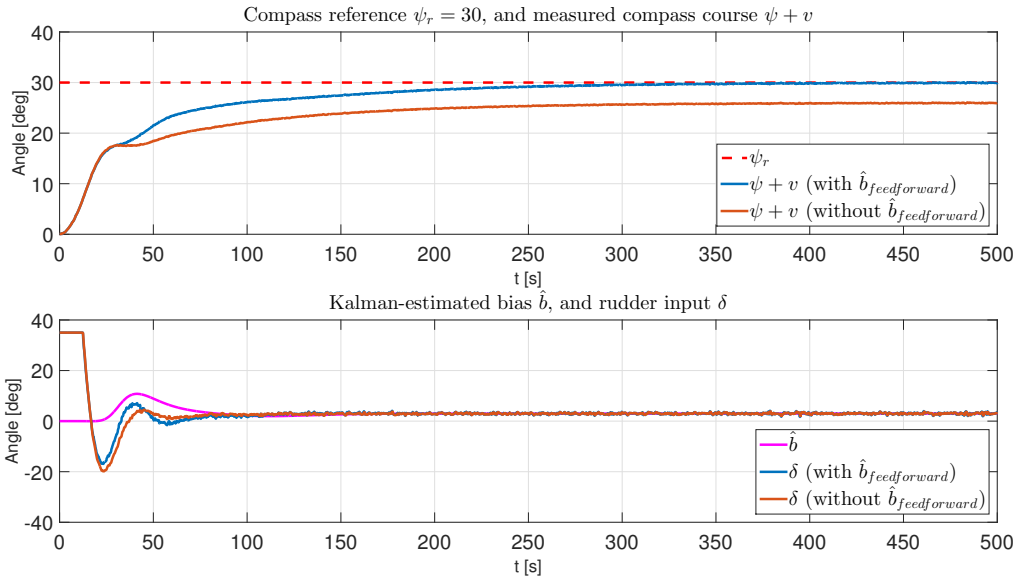

Figure 11: Comparing compass and rudder input angle with and without estimated feed-forward current bias

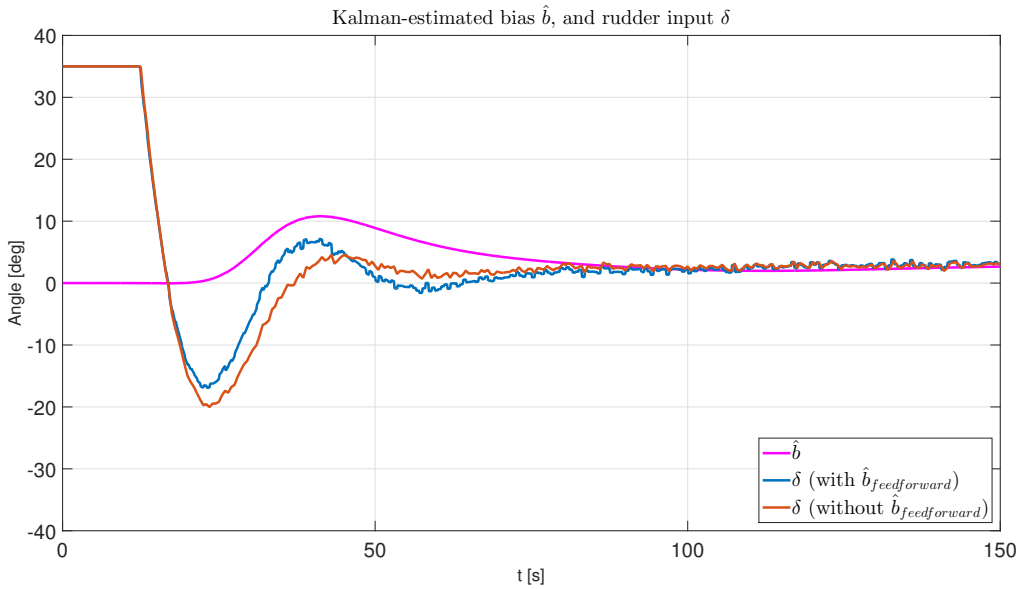

Figure 12: More detailed view of figure 11

The measured compass course is now equal to the reference angle after approximately 320 seconds, and the steady state error of approximately 3 degrees from problem 5.3, part c) is now removed. This is because the estimated bias is equal to the steady state error, and the estimated bias is added to the signal from the PD controller.Hence the current disturbance is cancelled. This performance is satisfactory, and much better than the one in problem 5.3, part c).

It is a difference in the rudder angle in the beginning of figure 11, where the plot picturing the rudder set-point angle with the estimated feed-forward current bias is slightly higher than the one without the feed-forward. This difference is enough to lift the compass angle to its desired value of 30 degrees, and the steady state error is avoided. The reason for the angle to be more accurate now, is that the estimation for the current bias cancels the effect of the current. After a while, both the rudder angles, with and without the feed forward, will be equal. In the case without the feed forward, there will be a steady state error, while in the case with a feed forward, the course angle will follow the reference perfectly.

It is seen from fig. 11 and fig. 12 how the rudder input angles become different from each other when the estimated bias becomes non-zero. This value is used as feed forward, and added to the rudder set-point angle in the case with a Kalman filter. After a while, this course angle is equal to the reference, and the error in the feedback loop is zero. The feed-forward will then control the rudder alone, which is confirmed by the fact that rudder and the estimated bias are equal after approximately 100 s. In the case without Kalman filter, the feedback error to the PD controller will control the rudder. Summarized; the feed forward gives the rudder set-point a large enough value to ivercome the bias from the current.

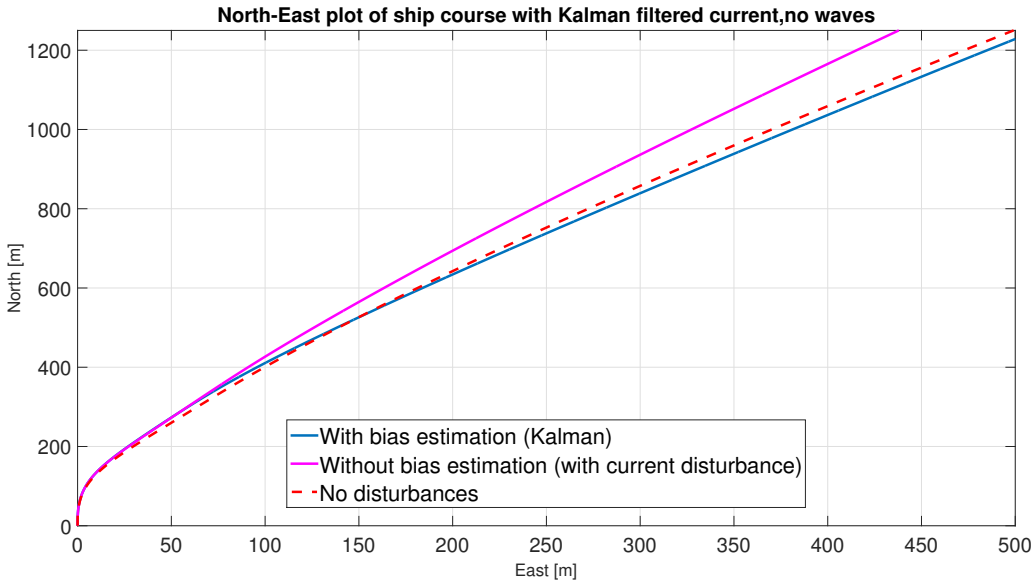

Figure 13: Ship path, both with and without the bias estimation

From fig. 13 it is obvious that the offset in course angle will result in a larger and larger offset from the actual destination, as the ship sails forward, while the controller with feed forward will follow approximately the same path as in the situation without disturbance.

5.e Wave filtering

The waves may change fast and frequently, and hence the measured compass course will also fluctuate a lot. Measurement noise will also contribute to this. Anyway, the average of the high frequency component due to wave disturbance and noise may be close to zero, and hence the course remains quite constant. With the controller using this measured angle as feedback, the rudder may unnecessarily change continuously because of the high frequency waves. To avoid this wear and tear on the rudder, the high frequency component due to wave disturbance, ψ_w, is substituted with the estimated $\hat{\psi}$, in the feedback. This is referred to as wave filtering, and only the dynamics of the course angle will influence the rudder. When both current and waves are present, the feed forward from the estimated bias will cancel the current bias, and the situation is approximately equal to the one in 3.d, where only waves are present.

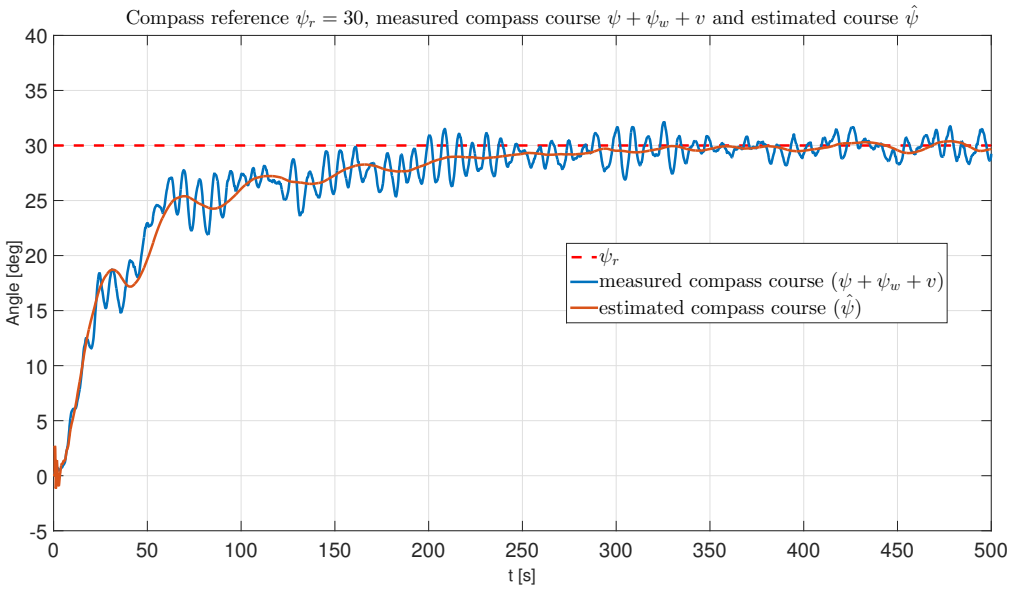

Figure 14: Measured and estimated courses

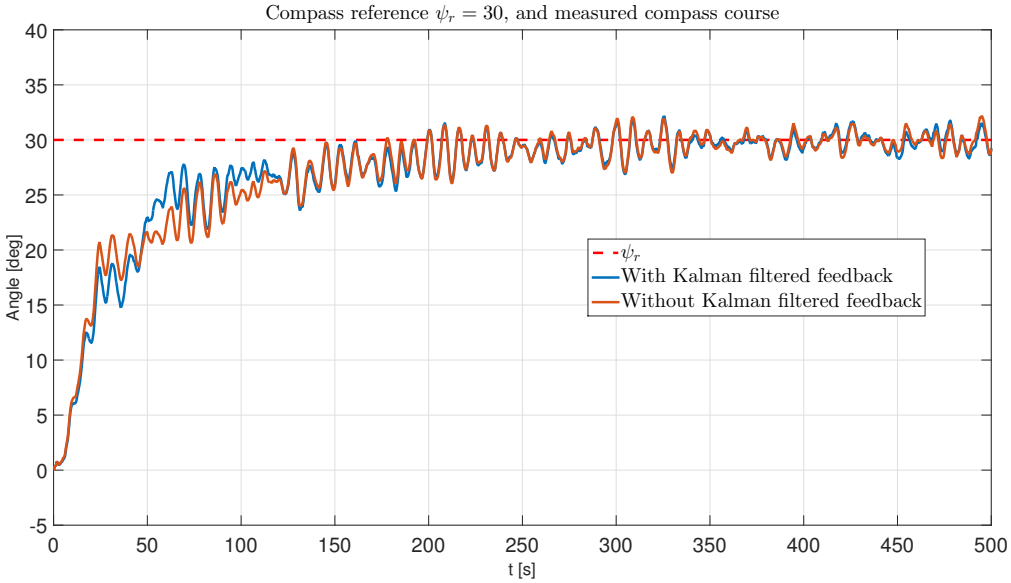

Figure 15: Ship course, with and without filtered feedback

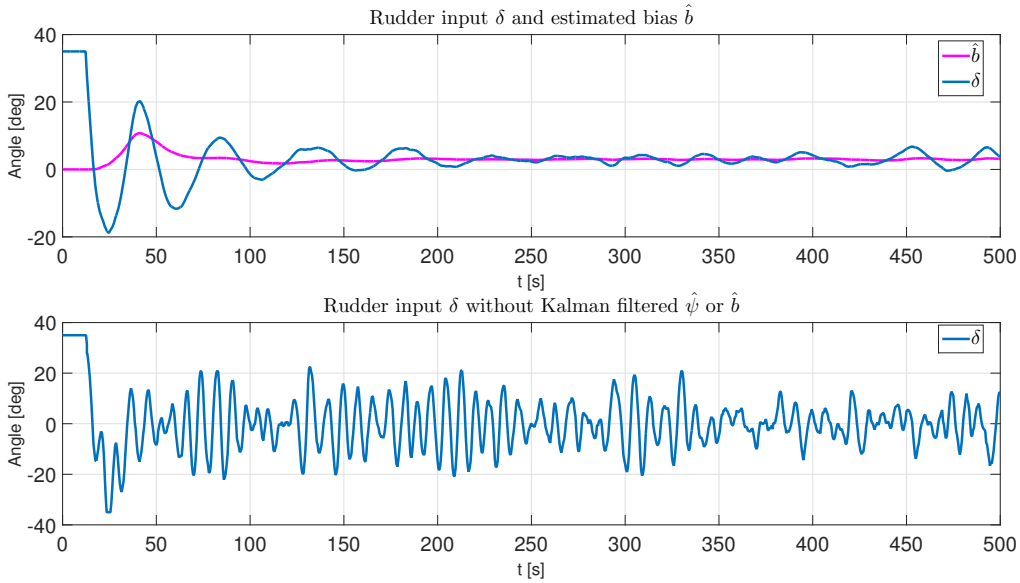

Figure 16: Estimated bias and rudder input angle in the situation with filtered (upper) and non-filtered (lower) feedback

It is obvious from fig. 14 how the estimated, wave filtered $\hat{\psi}$ follows the dynamics of the measured course, but avoids excessive oscillations. The high frequency component of the wave disturbance, as well as the measurement noise is filtered out. The remaining slow oscillations in the compass course, due to the waves rocking the ship, can not be filtered out.

Compared to the controller in section 3.d, the feed forward estimated bias will remove the steady state error, while the filtered feedback avoids unnecessary change in the rudder input.

As seen from fig. 15, the ship will sail with a constant compass course equal to the reference course after approximately four minutes, with small fluctuations because of the waves. It is observed how the ship will have approximately the same course during wave disturbance, independent on whether the actual measured, or the estimated compass course is used as feedback, but the latter alternative clearly avoids unnecessary wear and tear on the rudder. This is because a less fluctuating signal will be sent back to the rudder input, and hence cause a less fluctuating error, as seen from the estimated course in fig. 14. The impact of the estimated feedback can be seen by the rudder input angles in fig. 16. It is clear how the situation with filtered feedback will cause less wear and tear on the rudder due to the filtering of high-frequent noise, which was present in fig. 10, in part 3. The estimated bias has the same shape as in section 5.d.

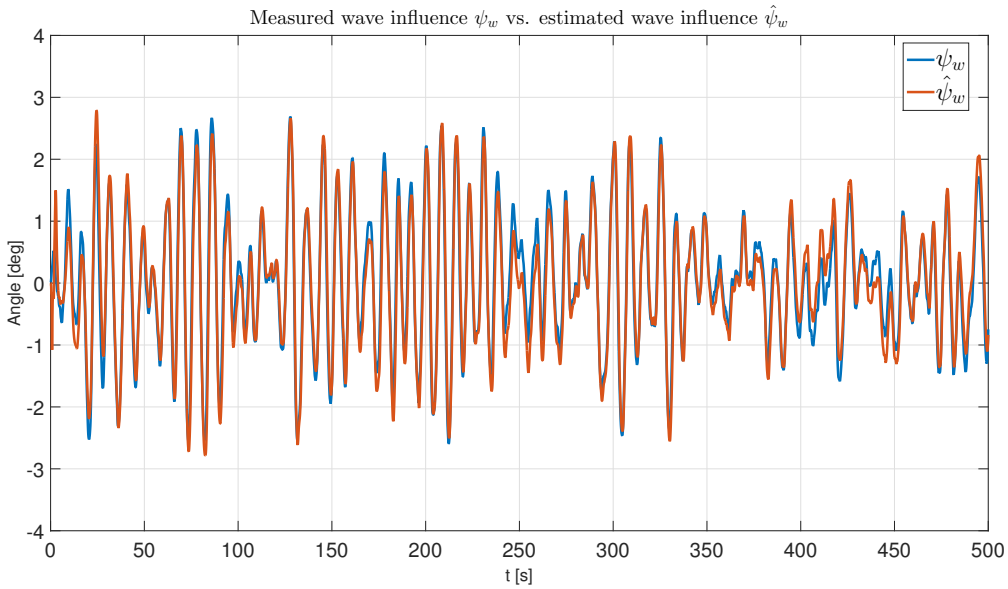

Figure 17: Actual wave influence and estimated and estimated wave influence. Rudder is set to zero to investigate the wave influence on the system only.

To measure the actual wave influence, the measurement noise and current disturbance must be turned off. The rudder input δ to the ship and Kalman filter must also be equal to zero, as seen in the `SIMULINK` implementation in figure 28, appendix B. The implemented Kalman `MATLAB` function block outputs the estimated compass heading, seen in appendix A.6. To output the estimated wave influence $\hat{\psi}_w$ instead, the last line of code must be changed like this

```
23 psi = x(3); b = x(5);   % x(3) is average compass heading
```

```
23 psi_w = x(2); b = x(5);   % Output name also changed to psi_w,
```

Now the ship outputs the wave influence ψ_w and the Kalman filter outputs the estimated wave influence $\hat{\psi}_w$. From 17 it is observed that the estimated high frequency wave influence is quite similar to the actual measured wave influence. This indicates that the estimator is quite good.

From fig. 18 it is seen that the filtered feedback does not improve the course significantly, as discussed above. It is of course important to note that the ship with Kalman filtering has to deal with both current and waves, but it is clear that this is no challenge for the filter. But the reason for why the situation from 3.d is slightly better than the one with Kalman filter may be due to the system with a pure PD-controller is faster. Tuning the PD-controller (as previously discussed), could improve the response time, which would result in the ship turning into correct course a little earlier. This is a minor trade-off compared with no current compensation or wave-filtering. We believe that the result is sufficient as is, and that Kalman-filtering has been a great success for improving the ship's performance.

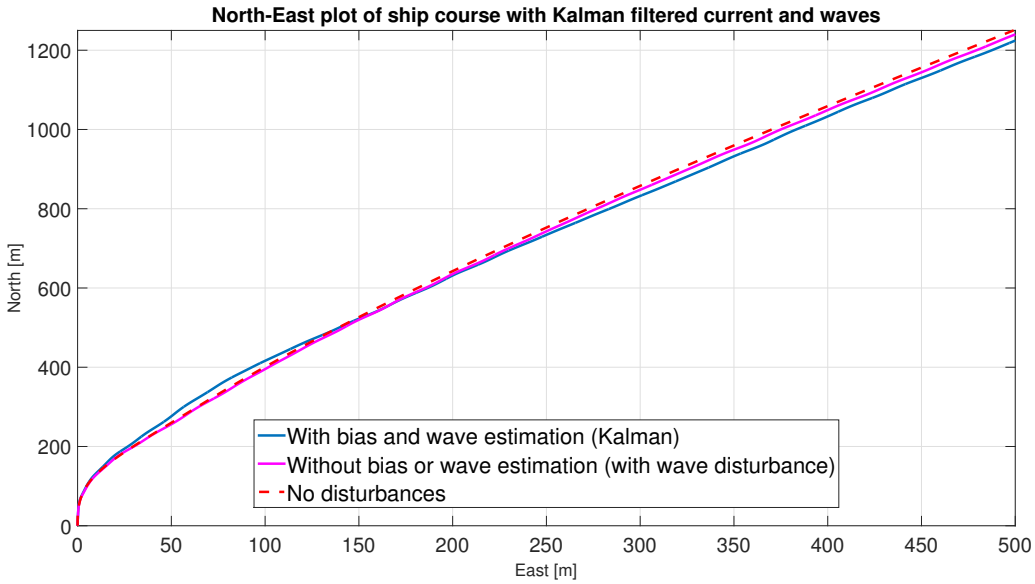

Figure 18: Ship path, both with and without the bias and wave estimation

6 Conclusion

In this assignment it was observed how waves and currents influence the behavior of a ship. It was observed how a simple PD controller with feedback is able to control the ship without disturbances, but it is not good enough to deal with waves and currents.

It was observed how the current created a steady state error in the compass course, compared to the reference. The disturbances from the waves resulted in rapid change in the rudder input angle, which again leads to faster wear and tear.

After confirming that the system was observable in all the considered situations, these problems were solved by using a Kalman filter. A feed forward loop canceled the current offset, and a filtered feedback smoothed the rudder operation. The Kalman filter filtered out the high frequency component of the wave disturbance, as well as measurement noise, wwhich resulted in smoother operation of the rudder.

References

[1] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, international fourth edition, 2013.

[2] Kristoffer Gryte. *TTK4115 - Discrete Kalman Filter Applied to a Ship Autopilot*, volume 1.8. Department of Engineering Cybernetics NTNU, October 2016.

[3] MathWorks. pwelch. `https://se.mathworks.com/help/signal/ref/pwelch.html`, Accessed November 9, 2016.

[4] Patrick Y.C. Hwang Robert Grover Brown. *Introduction Random Signals and Applied Kalman Filtering*. John Wiley & Sons, Inc., fourth edition edition, 2012.

A Appendix A - MATLAB codes

A.1 MATLAB - Part I

```
%% Init
close all
clc
clear variables

figNum = 1;             % Figure number-counter
w_1 = 0.005;            % For use in 5.1.b and 5.1.c
w_2 = 0.05;             % For use in 5.1.b and 5.1.c

% \\\ Load previously simulated data from .mat-files
addpath('Data-files')
load('omega_1.mat')     % 5.1.b - omega_1 is system output
load('omega_2.mat')     % 5.1.b - omega_2 is system output
load('omega_1_E.mat')   % 5.1.c - omega_1_E is system output
load('omega_2_E.mat')   % 5.1.c - omega_2_E is system output
%

%% TASK 5.1.b ----- Without disturbance
H_1 = omega_1;
H_2 = omega_2;
t = length(H_1);

% \\\ Plot system output without waves or noise
figure(figNum); figNum = figNum + 1;
subplot(2,1,1)
plot(H_1(:,1),H_1(:,2), 'LineWidth', 3)
xlabel('t [s]', 'FontSize', 18); ylabel('$Sin(\omega_{1}t)$ [deg])', ...
    'FontSize', 18, 'Interpreter', 'latex')
title('Sinus-input where \omega_{1} = 0.005 for model without noise', ...
    'FontSize', 24);
set(gca,'FontSize',14); grid on;

subplot(2,1,2)
plot(H_2(:,1),H_2(:,2),'r', 'LineWidth', 3)
xlabel('t [s]', 'FontSize', 18); ylabel('$Sin(\omega_{2}t)$ [deg]', ...
    'Interpreter', 'latex', 'FontSize', 18);
title('Sinus-input where \omega_2 = 0.05 for model without noise', ...
    'FontSize', 24)
set(gca,'FontSize',14); grid on;

% \\\ Find ampltiude peaks
A_11 = max(H_1(3500:t,2));
A_12 = min(H_1(3500:t,2));
A_21 = max(H_2(3500:t,2));
A_22 = min(H_2(3500:t,2));

```

```
% \\\ Average amplitude value of output
A_1 = (A_11-A_12)/2;
A_2 = (A_21-A_22)/2;

% \\\ Finding K and T from system output
K = sqrt((A_1^2*w_1^2- ((A_1^2*A_2^2*w_1^4*w_2^2)/(w_2^4*A_2^2)))...
    /(1-(A_1^2*w_1^4)/(w_2^4*A_2^2)));
T = (sqrt(K^2 - A_2^2*w_2^2))/(w_2^2*A_2);
%

%% TASK 5.1.c ----- with disturbance
H_1_E = omega_1_E;
H_2_E = omega_2_E;
t_E = length(H_1_E);

% \\\ Plot system output with waves and noise
figure(figNum)
figNum = figNum + 1;
subplot(2,1,1)
plot(H_1_E(:,1),H_1_E(:,2), 'LineWidth', 3)
xlabel('t [s]', 'FontSize', 18); ylabel('Sin($$\omega_{1}$$t) [deg]', ...
    'Interpreter', 'latex','FontSize', 18)
title('Sin-input where \omega_1 = 0.005 for model with waves and noise',...
    'FontSize', 24)
set(gca,'FontSize',14); grid on;

subplot(2,1,2)
plot(H_2_E(:,1),H_2_E(:,2),'r', 'LineWidth', 1)
xlabel('t [s]', 'FontSize', 18); ylabel('Sin($$\omega_{2}$$t) [deg]', ...
    'Interpreter', 'latex', 'FontSize', 18)
title('Sinus where \omega_2 = 0.05 for model with waves and noise', ...
    'FontSize', 24)
set(gca,'FontSize',14), grid on;

% \\\ Find amplitude peaks
A_11_E = max(H_1_E(3500:t_E,2));
A_12_E = min(H_1_E(3500:t_E,2));
A_21_E = max(H_2_E(3500:t_E,2));
A_22_E = min(H_2_E(3500:t_E,2));

% \\\ Average amplitude for w_1
A_1_E = (A_11_E - A_12_E)/2;

% \\\ Attempt to find average value of amplitude for w_2
s_w_2 = 0;
for i=4000:8000
    A_max = max(H_2_E(i-100:i+100,2));
    A_min = min(H_2_E(i-100:i+100,2));
    s_w_2 = s_w_2 + (A_max-A_min)/2;
end

```

```
A_2_E = s_w_2/4000;

% \\\ Finding K and T from system output
K_E = sqrt((A_1_E^2*w_1^2- ((A_1_E^2*A_2_E^2*w_1^4*w_2^2)/...
    (w_2^4*A_2_E^2)))/(1-(A_1_E^2*w_1^4)/(w_2^4*A_2_E^2)));
T_E = (sqrt(K_E^2 - A_2_E^2*w_2^2))/(w_2^2*A_2_E);
%

%% TASK 5.1.d ------ Step response
load('step_simulink.mat')

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

% \\\ Plot step response of ship and model
figure(figNum);
figNum = figNum + 1;
H_tf_sim = step_simulink;
step(H_tf,length(H_tf_sim)/2)
hold on;
plot(H_tf_sim(:,1),H_tf_sim(:,2),'r', 'LineWidth', 4)
title('Step response of ship and model', 'FontSize', 24)
legend({'$$Step_{model}$$', '$$Step_{ship}$$'}, 'FontSize', 36, ...
    'Interpreter', 'latex');
grid on; hold off;
xlabel('Time ','FontSize', 24); ylabel('Amplitude [deg]','FontSize', 24);
set(gca,'FontSize',14)
%
```

A.2 MATLAB - Part II

```
%% Init
close all
clc
clear variables

figNum = 1;             % Figure number-counter
load('wave.mat');       % Load wave disturbance
%

%% TASK 5.2.a ------ Estimate PSD
F_s = 10;
window = 4096;
noverlap = [];
nfft = [];

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

```

```
% \\\ f (Hz) to w (rad/s), and psd (power/pr. HZ) to psd (power s/rad)
omega = 2*pi.*f;
pxx = pxx./(2*pi);
%

%% TASK 5.2.c ------ Find omega_0
% Plot estimated PSD
figure(figNum)
figNum = figNum+1;
plot(omega,pxx, 'LineWidth', 4)
axis([0 2 -0.00005 16*10^(-4)])
hold on
xlabel('$\omega$ [$\frac{rad}{s}$]', 'FontSize', 20, ...
    'Interpreter', 'latex')
ylabel('$S_{\psi_{w}}(\omega)$ [rad]', 'FontSize', 20, ...
    'Interpreter', 'latex')
title(['Estimated power spectral density $S_{\psi_{w}}(\omega)$ of '...
    'wave disturbance'], 'FontSize', 20, 'Interpreter', 'latex')
grid on;

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;

% Find resonance frequency from estimated PSD
[maxPSD, indexAtMaxPSD] = max(pxx);
omega_0 = omega(indexAtMaxPSD);

% Plot arrow to resonance frequency
plot(omega_0, maxPSD, 'ro', 'MarkerSize', 10);
a = annotation('textarrow', 2.15*[0.23 0.203], [0.79 0.866], 'String', ...
    '($\omega_0, S_{\psi_{w}}(\omega_0)$)', 'Interpreter', 'latex');
a.Color = 'red';
a.FontSize = 36;
hold off
%

%% TASK 5.2.d ------ Finding lambda
% Plot estimated PSD
figure(figNum)
figNum = figNum+1;
plot(omega,pxx, 'LineWidth', 3)
axis([0 2 -0.00005 16*10^(-4)])
hold on
xlabel('$\omega$ [$\frac{rad}{s}$]', 'FontSize', 20, ...
    'Interpreter', 'latex')
ylabel('$S_{\psi_{w}}(\omega)$, $P_{\psi_{w}}(\omega)$ [rad]', ...
    'FontSize', 20, 'Interpreter', 'latex')
```

```
title(['Estimated $S_{\psi_{w}}(\omega)$ vs. analytical' ...
    '$P_{\psi_{w}}(\omega)$ for different values of $\lambda$'], ...
    'FontSize', 20, 'Interpreter', 'latex')
grid on

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;

% Find analytical PSD for several lambdas
sigma = sqrt(maxPSD);
for lambda=0.04:0.04:0.16

K_w = 2*lambda*omega_0*sigma;
pxx_a = (omega.*K_w).^2./(omega.^4 + omega_0^4 + ...
    2*omega_0^2*omega.^2*(2*lambda^2-1));
plot(omega, pxx_a, '--', 'LineWidth', 2);
end
legend({'$S_{\psi_{w}}(\omega)$', '$\lambda = 0.04$',...
    '$\lambda = 0.08$', '$\lambda = 1.2$', '$\lambda = 1.6$'}, ...
    'Interpreter', 'latex', 'FontSize', 24, 'Location', 'northwest');
hold off

% More accurate lambda plotting (not included in report, same result)
figure(figNum)
figNum = figNum+1;
plot(omega,pxx, 'LineWidth', 3)
axis([0 2 -0.00005 16*10^(-4)])
hold on
xlabel('$\omega$ [$\frac{rad}{s}$]', 'FontSize', 20, ...
    'Interpreter', 'latex')
ylabel('$S_{\psi_{w}}(\omega)$, $P_{\psi_{w}}(\omega)$ [rad]', ...
    'FontSize', 20, 'Interpreter', 'latex')
title(['Estimated $S_{\psi_{w}}(\omega)$ vs. analytical '...
    '$P_{\psi_{w}}(\omega)$'], 'FontSize', 20, 'Interpreter', 'latex')
grid on

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=0.07:0.01:0.10
K_w = 2*lambda*omega_0*sigma;
pxx_a = (omega.*K_w).^2./(omega.^4 + omega_0^4 + ...
    2*omega_0^2*omega.^2*(2*lambda^2-1));
```

```
plot(omega, pxx_a, '--', 'LineWidth', 2);
end
legend({'$S_{\psi_{w}}(\omega)$', '$\lambda = 0.07$', ...
    '$\lambda = 0.08$', '$\lambda = 0.9$', '$\lambda = 1.0$'},  ...
    'Interpreter', 'latex', 'FontSize', 24, 'Location', 'northwest');
hold off

%choose lambda to be 0.08
lambda = 0.08;
%
```

A.3 MATLAB - Part III

```
%% Init
close all
clc
clear variables

figNum = 1;     % Figure number-counter
PSI_r = 30;     % Reference angle for simulation
sim_t = 500;    % Simulation time

% \\\ Simulink models
addpath('Simulink models tasks')
%

%% TASK 5.3.a ---- Designing a PD-controller
% \\\ Setting up transfer variable s and constants
s = tf('s');
K = 0.1742;
T = 86.5256;
T_d = T;
w_c = 0.1;

% \\\ Defining the transfer function
T_f = -1/(tan(130*pi/180)*w_c);
K_pd = sqrt(w_c^2+T_f^2*w_c^4)/K;
H_0 = (K*K_pd)/(s*(1+T_f*s));

% \\\ Bode margin plot of transfer function
figure(figNum)
figNum = figNum + 1;
margin(H_0); grid on;
%

%% TASK 5.3.b ---- Simulating without disturbances
load_system('task5_3_b.slx')
sim('task5_3_b.slx')

```

```
% \\\ Plot of autopilot wihtout current and waves
figure(figNum)
figNum = figNum+1;
plot(t,sim_PSI_r, 'r--',t,sim_compass, t, delta, 'LineWidth',3);
title('Autopilot without current or waves', 'FontSize', 24);
xlabel('t [s]', 'FontSize', 20); grid on;
ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
    'FontSize', 20, 'Interpreter', 'latex');
legend({'$\psi_{r}$', '$\psi$', '$\delta$'}, 'Location', ...
    'best', 'FontSize', 36, 'Interpreter', 'latex')
ax = gca; ax.FontSize = 24;

% \\\ Save delta and north-east data for later plotting
delta1 = delta; t1 = t; psi1 = sim_compass;
x1 = north_east(:,1); y1 = north_east(:,2);
%

%% TASK 5.3.c --- Simulating with current, without waves
load_system('task5_3_c.slx')
sim('task5_3_c.slx')

% \\\ Plot of autopilot with current, but without waves
figure(figNum)
figNum = figNum + 1;
plot(t,sim_PSI_r, 'r--',t,sim_compass, t, delta, 'LineWidth',3);
xlabel('t [s]', 'FontSize', 20); grid on;
ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
    'FontSize', 20, 'Interpreter', 'latex');
title('Autopilot with current, but without waves', 'FontSize', 24);
legend({'$\psi_{r}$', '$\psi$', '$\delta$'}, 'Location', ...
    'best', 'FontSize', 36, 'Interpreter', 'latex')
ax = gca; ax.FontSize = 24;

% \\\ Save delta and north-east data for later plotting
delta2 = delta; t2 = t; psi2 = sim_compass;
x2 = north_east(:,1); y2 = north_east(:,2);
%

%% TASK 5.3.d --- Simulating with waves, without current
load_system('task5_3_d.slx')
sim('task5_3_d.slx')

% \\\ Plot of autopilot with waves and without current
figure(figNum)
figNum = figNum + 1;
plot(t,sim_PSI_r,'r--',t,sim_compass, t, delta, 'LineWidth',3);
xlabel('t [s]', 'FontSize', 20); grid on; ax = gca; ax.FontSize = 24;
ylabel('$\mathbf{\psi_{r}(t), \ \psi(t), \ \delta}$ [deg]', ...
    'FontSize', 20, 'Interpreter', 'latex');
title('Autopilot with waves and without current', 'FontSize', 24);
legend({'$\psi_{r}$', '$\psi + \psi_{w}$', '$\delta$'}, 'Location', ...
```

```
    'best', 'FontSize', 36, 'Interpreter', 'latex')

% \\\ Save delta and north_east data for later plotting
delta3 = delta; t3 = t; psi3 = sim_compass;
x3 = north_east(:,1); y3 = north_east(:,2);
%

%% Plotting the different rudder inputs against eachother
figure(figNum)
figNum = figNum+1;

% \\\ Subplot for delta with no disturbance and current disturbance
subplot(2,1,1)
plot(t1,delta1,t2,delta2,'LineWidth', 2);
title('Rudder input \delta with and without current', 'FontSize', 24);
xlabel('t [s]', 'fontSize', 20); grid on;
ylabel('\delta [deg]', 'FontSize', 20);
legend({'\delta no disturbance', '\delta with current'}, 'Location', ...
    'best', 'FontSize', 36)
axis([0 150 -30 40]); ax = gca; ax.FontSize = 24;

% \\\ Subplot for delta with wave disturbance
subplot(2,1,2);
plot(t3,delta3,'LineWidth',2); xlabel('t [s]', 'FontSize', 20);
title('Rudder input \delta with waves', 'FontSize', 24);
ylabel('\delta [deg]', 'FontSize', 20); grid on;
legend({'\delta with waves'}, 'Location', 'best', 'FontSize', 36)
axis([0 150 -40 40]); ax = gca; ax.FontSize = 24;
%

%% Plotting North-East plot
figure(figNum)
figNum = figNum+1;

% \\\ Plot North-East plot for previous cases
plot(y1,x1,'r--',y2,x2,y3,x3,'m', 'LineWidth', 3);
title('North-East plot of ship course', 'FontSize', 24);
ylabel('North [m]', 'FontSize', 20); grid on;
xlabel('East [m]', 'FontSize', 20);
ax = gca; ax.FontSize = 24; axis([0 150 0 600]);
legend({'Without disturbances', 'With current', ...
    'With waves'}, 'Location', 'best', 'FontSize', 36)
%
```

A.4 MATLAB - Part IV

```
%% Init
close all
clc
```

```
clear variables

addpath('Data-files')
% \\\ Constants from previous tasks (witout disturbances)
load('constants_5.2.mat')
load('constants_5.3.mat')
%

%% TASK 5.4.a ------ Finding A, B, C and E
A_bw = [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_bw = [0; 0; 0; K/T; 0];
C_bw = [0 1 1 0 0];
E_bw = [0 0; K_w 0; 0 0; 0 0; 0 1];
%

%% TASK 5.4.b ------ Observabillity without disturbances
A = [0 1; 0 -1/T];
B=[0; K/T];
C= [1 0];
O = obsv(A,C);

% \\\ If rank(O) = 2, we have full rank <=> Observabillity
rank_O = rank(O);
%

%% TASK 5.4.c ------ Observabillity with current
A_b = [0 1 0; 0 -1/T -K/T; 0 0 0];
C_b = [1 0 0];
O_b = obsv(A_b,C_b);

% \\\ If rank(O_b) = 3, we have full rank <=> Observabillity
rank_O_b = rank(O_b);
%

%% TASK 5.4.d ------ Observabillity with wave
A_w = [0 1 0 0; -omega_0^2 -2*lambda*omega_0 0 0; 0 0 0 1; 0 0 0 -1/T];
C_w = [0 1 1 0];
O_w = obsv(A_w,C_w);

% \\\ If rank(O_w) = 4, we have full rank <=> Observabillity
rank_O_w = rank(O_w);
%

%% TASK 5.4.e ------ Observabillity with current and wave
O_bw = obsv(A_bw,C_bw);

% \\\ If rank(O_bw) = 5, we have full rank <=> Observabillity
rank_O_bw = rank(O_bw);
%

```

```
%% ---------- PRINT RESULTS ----------
fprintf(['\n\n\t\t*-------------------------------------'...
    '-------------------------*\n']);
fprintf(['\t\t| \t  TABLE\t\t| ideal\t|current|  wave\t| '...
    'current and wave |\n']);
fprintf(['\t\t*-------------------------------------------'...
    '-------------------*\n']);
fprintf('\t\t| required rank\t|\t2\t|\t3\t|\t4\t|\t\t  5\t\t   |\n');
fprintf('\t\t|  actual rank\t|\t%i\t|\t%i\t|\t%i\t|\t\t  %i \t   |\n', ...
    rank_O, rank_O_b, rank_O_w, rank_O_bw);
fprintf('\t\t|  Observable?\t|  Yes\t|  Yes\t|  Yes\t|\t\t Yes\t   |\n');
fprintf(['\t\t*-------------------------------------------'...
    '-------------------*\n\n']);
%
```

A.5 MATLAB - Part V

```
%% Init
close  all
clc
clear  variables

figNum = 1;         % Figure number-counter
PSI_r = 30;         % Reference angle for simulation
sim_t = 500;        % Simulation time

% \\\ Constants from previous tasks
addpath('Data-files')               % Add folder for .mat-files
load('constants_5.2.mat')           % K_w, lambda, omega_0, sigma
load('constants_5.3.mat')           % K, K_pd, T, T_d, T_f, w_c

% \\\ X and Y output of ship
%     in task 5.3.b/c/d   ///
load('north_east_5.3.mat')          % x1, x2, x3, y1, y2, y3

% \\\ Rudder input to ship
%     in task 5.3.b/c/d   ///
load('rudder_input_5.3.mat')         % delta1, delta2, delta3, t1, t2, t3

% \\\ Psi output of ship
%     in task 5.3.b/c/d   ///
load('compass_measurment_5.3.mat')   % psi1, psi2, psi3

% \\\ Add working path for simulink models
addpath('Simulink models tasks')
%

%% Task 5.5.a ---- Exact discretization
% \\\ Matrices from Task 5.4.a
```

```
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];

% \\\ Sample frequency
F_s = 10;
T_s = 1/F_s;

% \\\ Exact discretization
[Ad, Bd]  = c2d(A,B,T_s);    Cd = C;
[Ad, Ed] = c2d(A,E,T_s);    Dd = D;
%

%% Task 5.5.b ------ Estimate of var(v)
load_system('task5_5_b.slx')
sim('task5_5_b.slx')

% \\\ R is the Measurment noise variance
R = var(sim_compass*pi/180);
%

%% Task 5.5.c ------ Discrete Kalman Filter
% \\\ Q is the Process noise covariance
Q = [30 0; 0 10^(-6)];
P_0 = [1 0 0 0 0; 0 0.013 0 0 0; 0 0 pi^2 0 0; 0 0 0 1 0; ...
    0 0 0 0 2.5*10^-4];
X_0 = [0; 0; 0; 0; 0];
R = R/T_s;
I = diag([1 1 1 1 1]);

% \\\ Put data in a struct for use in the Kalman filter
data = struct('Ad',Ad,'Bd',Bd,'Cd',Cd,'Ed', Ed, 'Q',Q,'R', R,'P_0',P_0, ...
              'X_0',X_0, 'I', I);
%

%% Task 5.5.d ------ Feed forward estimated bias
load_system('task5_5_d.slx')
sim('task5_5_d.slx')

% \\\ Plot measured compass with and without estimated bias
figure(figNum)
figNum = figNum+1;
subplot(2,1,1)
plot(t,sim_PSI_r, 'r--', t, sim_compass, t2, psi2, 'LineWidth', 3)
title(['Compass reference $\psi_{r} = 30$, and measured compass course' ...
    ' $\psi + v$'], ...
    'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
```

```
legend({'$\psi_{r}$', '$\psi + v$ (with $\hat{b}_{feedforward}$)', ...
    '$\psi + v$ (without $\hat{b}_{feedforward}$)'}, 'FontSize', 24,  ...
    'Interpreter', 'latex', 'Location', 'SouthEast')
ax = gca; ax.FontSize = 24; grid  on ;

subplot(2,1,2)
plot(t, b_filtered,'m', t, delta, t2, delta2, 'LineWidth', 3)
title('Kalman-estimated bias $\hat{b}$, and rudder input $\delta$', ...
    'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
legend({'$\hat{b}$', '$\delta$ (with $\hat{b}_{feedforward}$)', ...
    '$\delta$ (without $\hat{b}_{feedforward}$)'}, 'FontSize', 24,  ...
    'Interpreter', 'latex', 'Location', 'SouthEast')
ax = gca; ax.FontSize = 24; grid on;

% \\\ Plotting North-East plot for this task and part 5.3
figure(figNum)
figNum = figNum+1;
plot(north_east(:,2),north_east(:,1),y2, x2, 'm', y1,x1,'r--', ...
    'LineWidth', 3);
title(['North-East plot of ship course with Kalman filtered current,'...
    'no waves'], 'FontSize', 24);
ylabel('North [m]', 'FontSize', 20); grid on;
xlabel('East [m]', 'FontSize', 20);
legend({'With bias estimation (Kalman)', ...
    'Without bias estimation (with current disturbance)',...
    'No disturbances'}, 'FontSize', 26, 'Location', 'best');
ax = gca; ax.FontSize = 24; axis([0 500 0 1250]);
%

%% Task 5.5.e ------ Feed forward estimated bias and wave filtered psi
load_system('task5_5_e.slx')
sim('task5_5_e.slx')

% \\\ Plot measured compass and estimated compass
figure(figNum)
figNum = figNum+1;
plot(t,sim_PSI_r, 'r--', t, sim_compass, t, psi_filtered, 'LineWidth', 3)
title(['Compass reference $\psi_{r} = 30$, measured compass course '...
    '$\psi + \psi_{w} + v$ and estimated course $\hat{\psi}$'], ...
    'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
legend({'$\psi_{r}$', 'measured compass course ($\psi + \psi_{w} + v$)' ...
    , 'estimated compass course ($\hat{\psi}$)'}, 'FontSize', 24, ...
    'Interpreter', 'latex', 'Location', 'best')
ax = gca; ax.FontSize = 24; grid on; axis([0 500 -5 40])

% \\\ Plot measured psi with and without Kalman filtered bias and waves
figure(figNum)
figNum = figNum+1;
plot(t,sim_PSI_r, 'r--', t, sim_compass, t3, psi3, 'LineWidth', 3)
```

```
title('Compass reference $\psi_{r} = 30$, and measured compass course', ...
    'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
legend({'$\psi_{r}$', 'With Kalman filtered feedback', ...
    'Without Kalman filtered feedback'}, 'FontSize', 24, ...
    'Interpreter', 'latex', 'Location', 'best')
ax = gca; ax.FontSize = 24; grid on; axis([0 500 -5 40])

% \\\ Plotting rudder input delta
figure(figNum)
figNum = figNum+1;
subplot(2,1,1)
plot(t, b_filtered,'m', t,delta, 'LineWidth', 3)
title('Rudder input $\delta$ and estimated bias $\hat{b}$', ...
    'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
legend({'$\hat{b}$', '$\delta$'}, 'FontSize', 24,  ...
    'Interpreter', 'latex', 'Location', 'best')
ax = gca; ax.FontSize = 24; grid on;

subplot(2,1,2)
plot(t3, delta3, 'LineWidth', 3)
title(['Rudder input $\delta$ without Kalman filtered $\hat{\psi}$ '...
    'or $\hat{b}$'],'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
legend({'$\delta$'}, ...
    'FontSize', 24, 'Interpreter', 'latex', 'Location', 'best')
ax = gca; ax.FontSize = 24; grid on;

% \\\ Plotting North-East plot compared to 5.3.d
figure(figNum)
figNum = figNum+1;
plot(north_east(:,2),north_east(:,1),y3, x3, 'm', y1,x1,'r--', ...
    'LineWidth', 3);
title(['North-East plot of ship course with Kalman filtered current '...
    'and waves'], 'FontSize', 24);
ylabel('North [m]', 'FontSize', 20); grid on;
xlabel('East [m]', 'FontSize', 20);
legend({'With bias and wave estimation (Kalman)', ...
    'Without bias or wave estimation (with wave disturbance)', ...
    'No disturbances'}, 'FontSize', 26, 'Location', 'best');
ax = gca; ax.FontSize = 24; axis([0 500 0 1250]);


% \\\ Wave influence (current turned off, delta = 0)
load_system('task5_5_e_2.slx')
sim('task5_5_e_2.slx')

% \\\ Plotting wave influence on system
figure(figNum)
figNum = figNum+1;
```

```
plot(t, sim_compass, t, psi_filtered, 'LineWidth', 3)
title(['Measured wave influence $\psi_{w}$ vs. estimated wave influence'...
    ' $\hat{\psi}_{w}$'], 'FontSize', 24, 'Interpreter', 'latex');
xlabel('t [s]', 'FontSize', 20); ylabel('Angle [deg]', 'FontSize', 20);
legend({'$\psi_{w}$', '$\hat{\psi}_{w}$'}, 'FontSize', 24,  ...
    'Interpreter', 'latex', 'Location', 'best')
ax = gca; ax.FontSize = 24; grid on;
axis([0 500 -4 4]);
%
```

A.6 Kalman MATLAB function

```
function [b,psi] = Kalman_matlab_fnc(u, y, data)
%#codegen
persistent init_flag A B C E Q R P_ x_ I

if (isempty(init_flag))
    init_flag = 1;

    % Initialization for system
    [A,B,C,E,Q,R,P_,x_, I] = deal(data.Ad,data.Bd,data.Cd,data.Ed, ...
        data.Q, data.R, data.P_0, data.X_0, data.I);
end

% 1 - Compute the Kalman Gain
    L = (P_*C')/((C*P_*C'+R));
% 2 - Update estimate with measurment
    x = x_ + L*(y-C*x_);
% 3 - Update error covariance matrix
    P = (I - L*C)*P_*(I-L*C)'+L*R*L';
% 4 - Projet ahead
    x_ = A*x + B*u;
    P_ = A*P*A' + E*Q*E';

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

end
```

B Appendix B - SIMULINK models

B.1 SIMULINK Task5.1.b

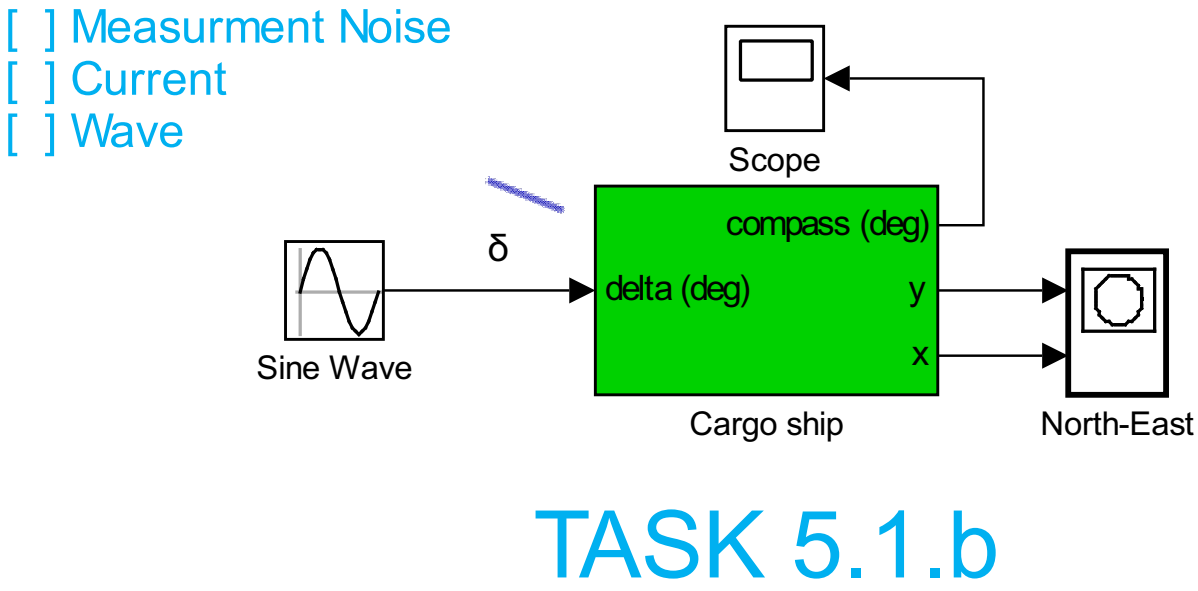

Figure 19: SIMULINK for section 1.b

B.2 SIMULINK Task5.1.c

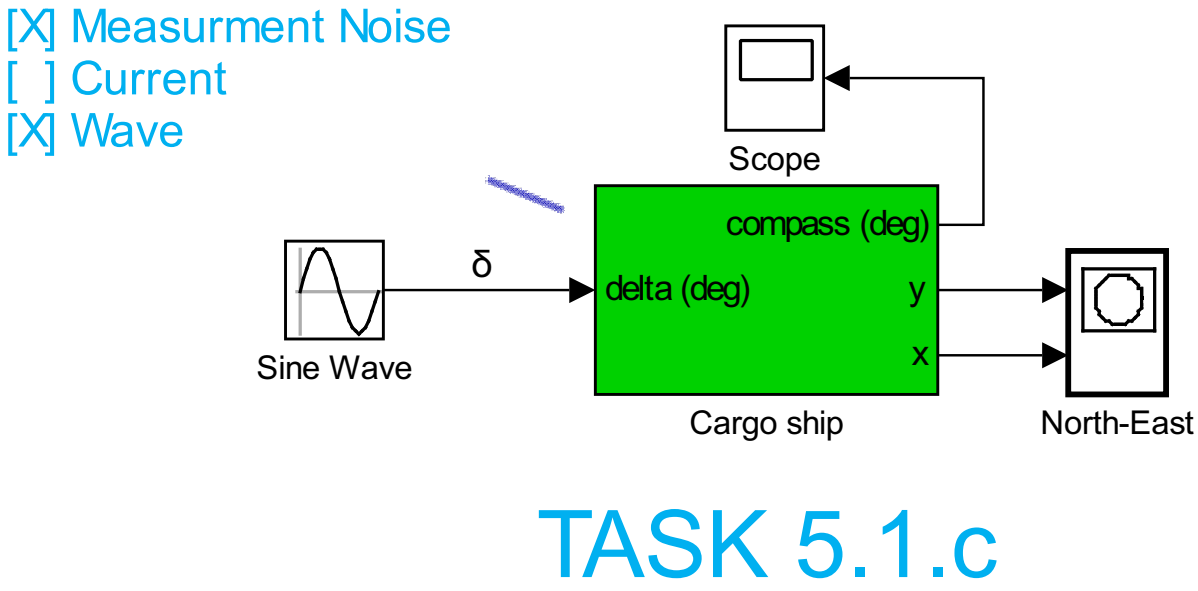

Figure 20: SIMULINK for section 1.c

B.3 SIMULINK Task5.1.d

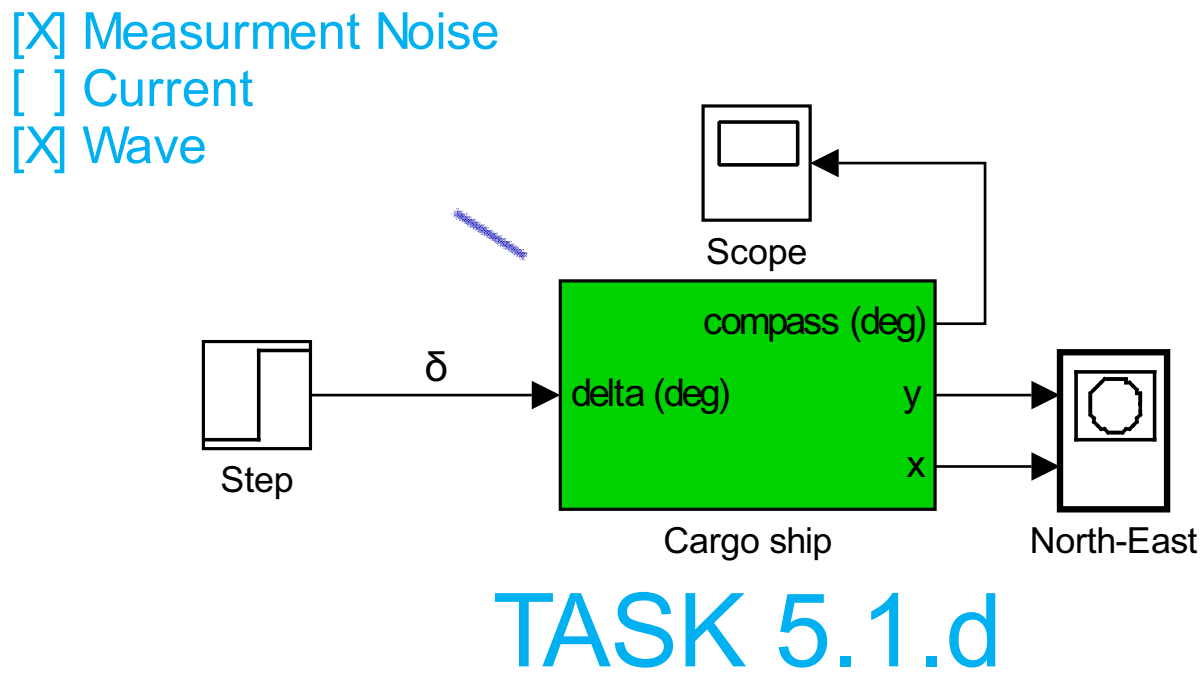

Figure 21: SIMULINK for section 1.d

B.4 SIMULINK Task5.3.b

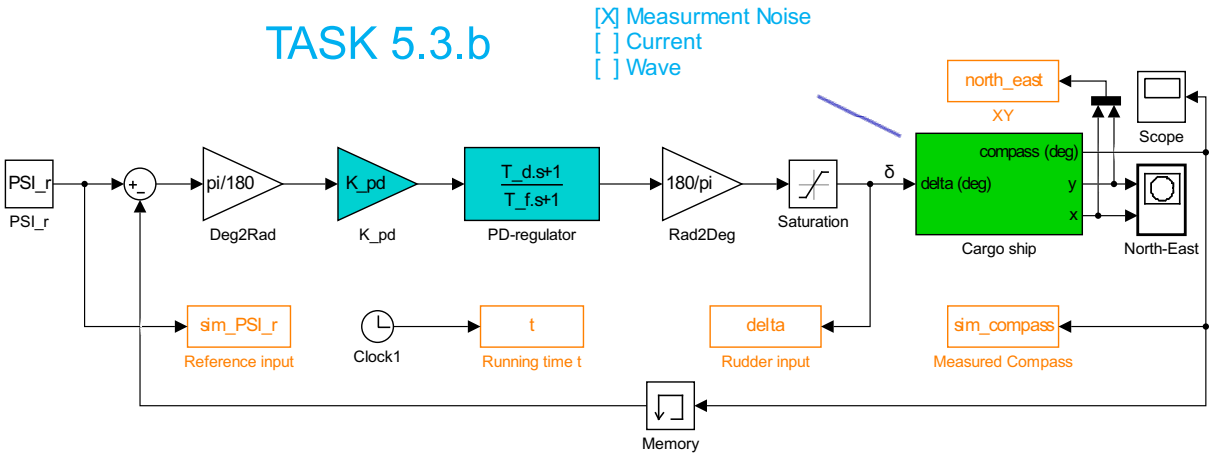

Figure 22: SIMULINK for section 3.b

B.5 SIMULINK Task5.3.c

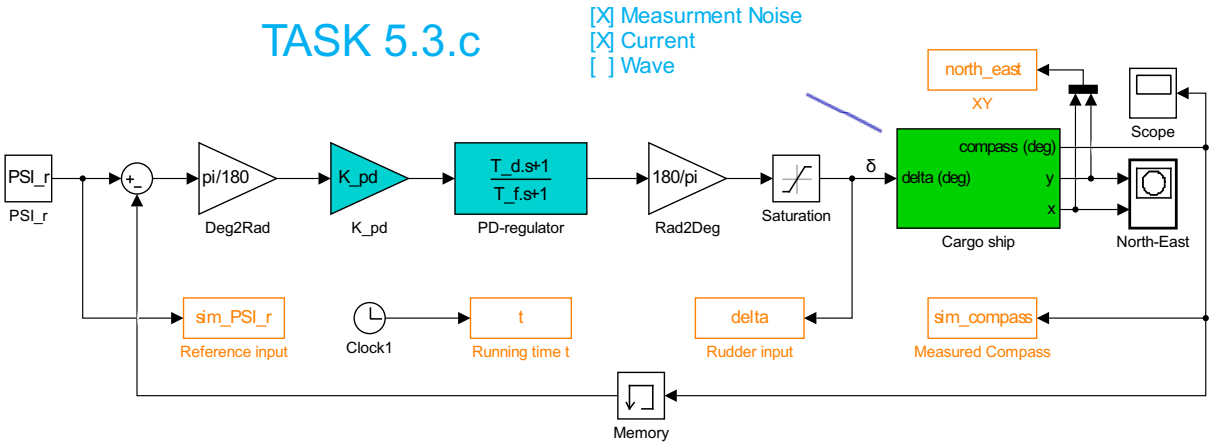

Figure 23: SIMULINK for section 3.c

B.6 SIMULINK Task5.3.d

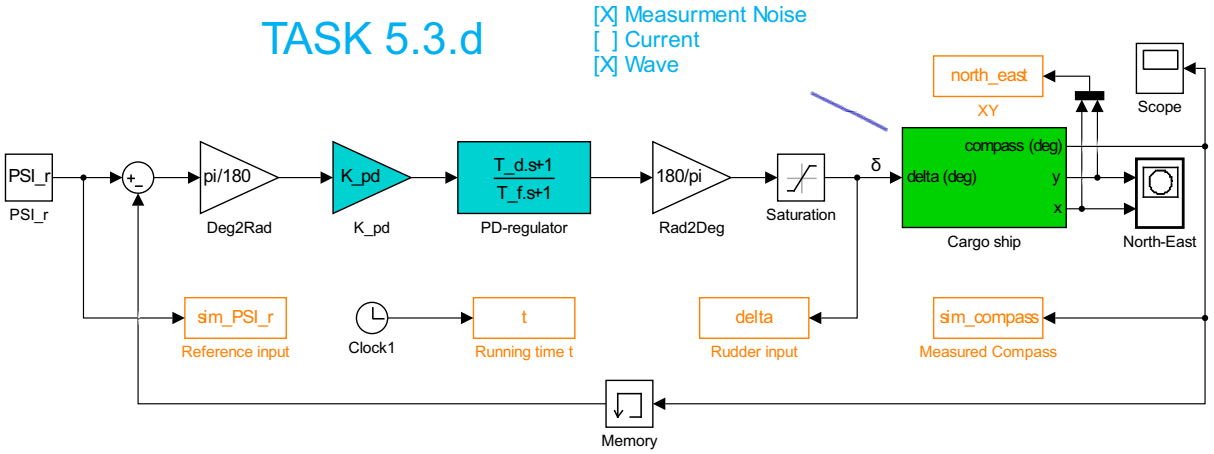

Figure 24: SIMULINK for section 3.d

B.7 SIMULINK Task5.5.b

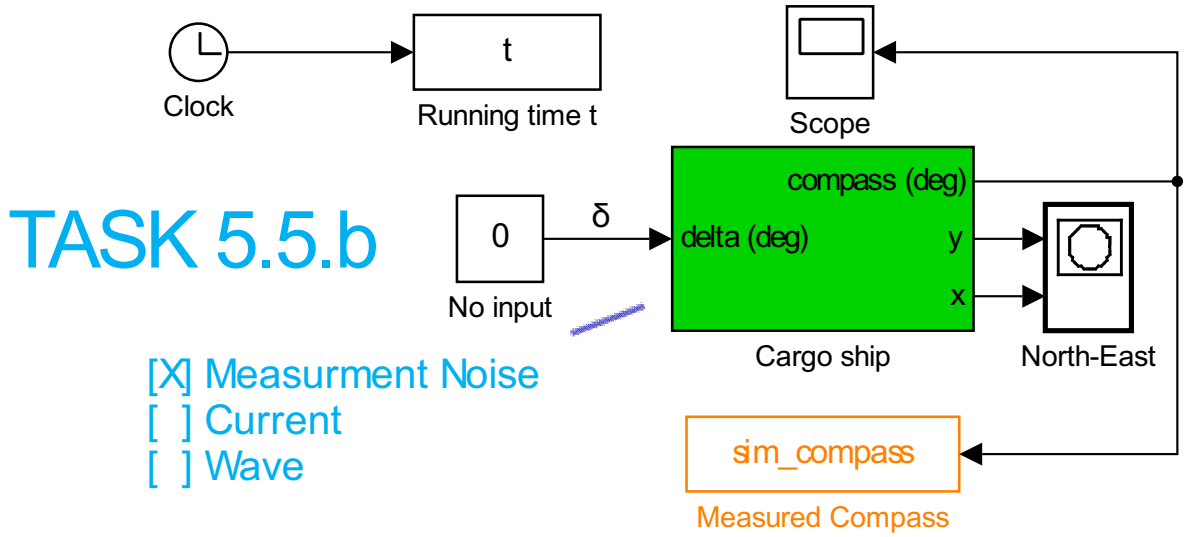

Figure 25: SIMULINK for section 5.b

B.8 SIMULINK Task5.5.d

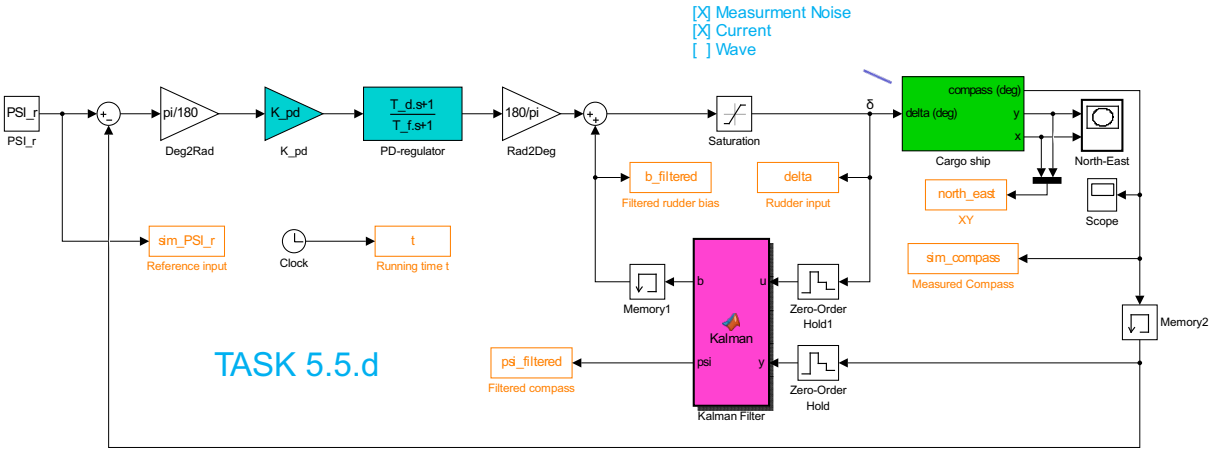

Figure 26: SIMULINK for section 5.d

B.9 SIMULINK Task5.5.e

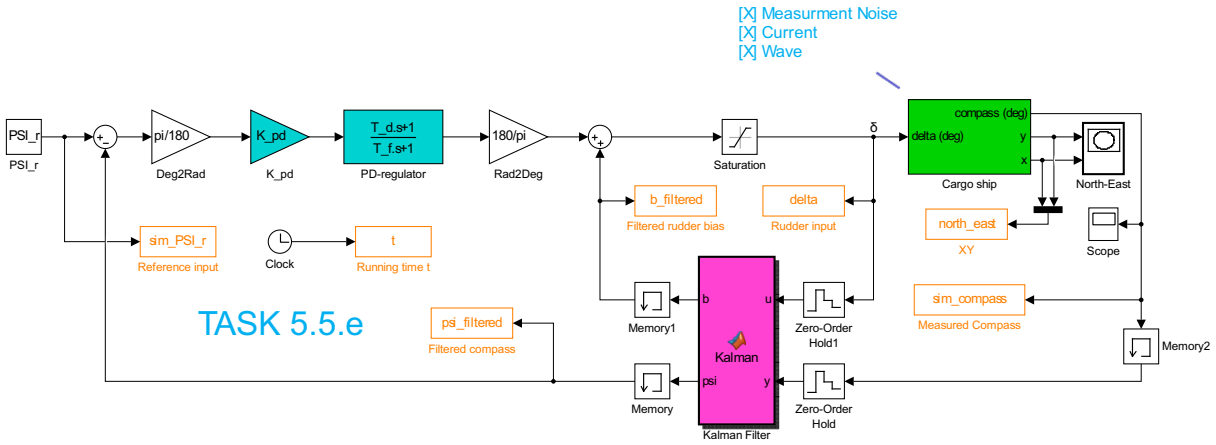

Figure 27: SIMULINK for section 5.e

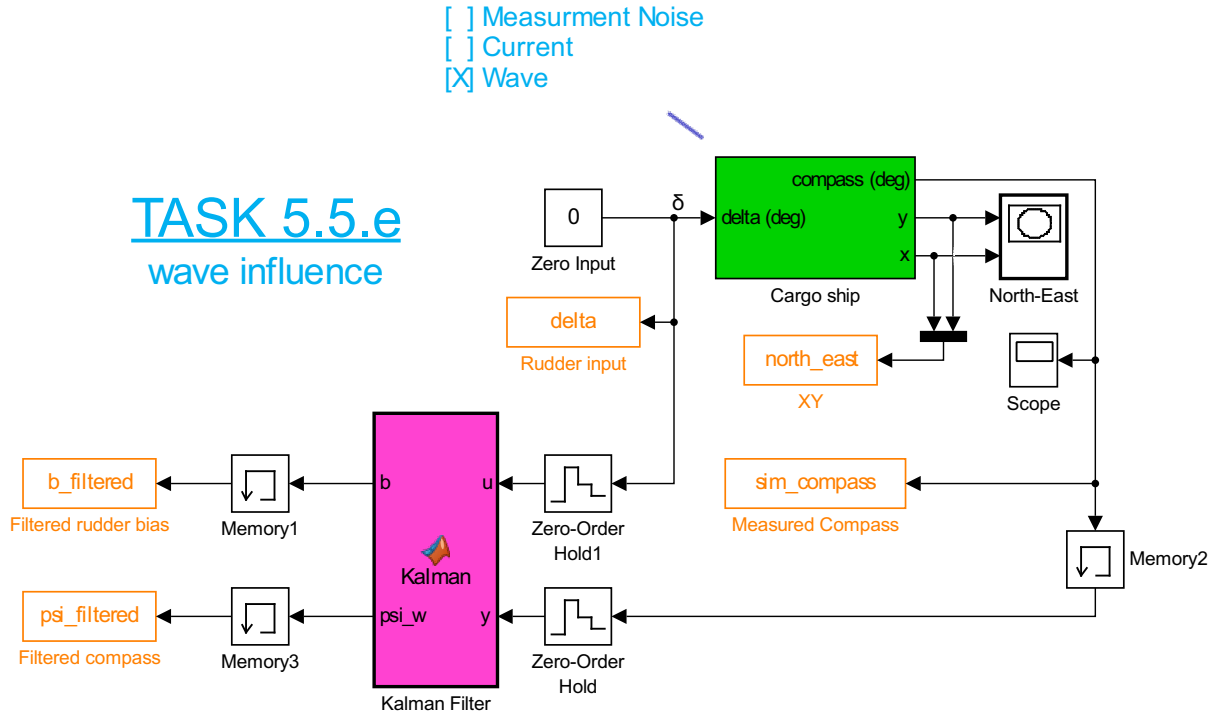

Figure 28: SIMULINK for section 5.e to find the wave influence