Linear Systems TTK4115

Discrete Kalman Filter Applied to a Ship Autopilot

Håkon Espeland – 772703
Ali Al-Jumaili – 772170

Group 63

20. November 2016

NTNU – Trondheim
Norwegian University of
Science and Technology

Contents

Introduction

In this report an autopilot for a cargo ship is developed, and a discrete Kalman filter is applied to the system. To obtain desirable results, stochastic modeling, basic identification and control theory are all used. The entire project is run through Mathworks powerful software MatLab as a simulation of a real system. Graphs, Simulink models and Matlab code are all a part of this report, explaining and displaying our results.

1 Identification of Boat Parameters

In this section a transfer function from the rudder input δ to the average heading ψ is derived for the ship. The simulation model provides estimated parameters for the transfer function, before a comparison between the response of the transfer function and the simulation model is made to evaluate the approximation.

1.1 Transfer function

In order to derive a transfer function from δ to ψ, the we will have to assume no current disturbances, using the equations for $\dot{\psi}$ and $\dot{r}$ given by:

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

$$\dot{r} = -\frac{1}{T}r + \frac{K}{T}\delta \tag{2}$$

The model for the average heading is obtained by inserting (2) into the derivative of (1)

$$\ddot{\psi} = -\frac{1}{T}r + \frac{K}{T}\delta \tag{3}$$

The transfer function from δ to ψ is obtained by applying Laplace-transformation to (3), assuming zero initial conditions.

$$H(s) = \frac{\psi(s)}{\delta(s)} = \frac{K}{s(Ts+1)} \tag{4}$$

The transfer function (4) describes how the rudder (δ) affects the average heading (ψ).

1.2 Identifying Boat Parameteres without Disturbances

To obtain the values of K and T from (4), we applied two sine inputs of different frequency to the simulation, observing the amplitude of the output. By using the fact that

$$A\left|H(j\omega_i)\right| = A_0 \tag{5}$$

where A is the amplitude and is the frequency of the input signal, while A_0 is the amplitude of the output signal. This method provides a set of two equations with the same number of unknown variables, which we solved for T and K.

Expanding equation (5) gives:

$$\left|H(j\omega_i)\right| A = \frac{|K|}{|-T\omega^2 + j\omega|} A = A_0 \tag{6}$$

$$K = \omega\sqrt{T^2\omega^2 + 1}\frac{A_0}{A} \tag{7}$$

$$T = \sqrt{\frac{K^2}{\omega^4}\frac{A^2}{{A_0}^2} - \frac{1}{\omega^2}} \tag{8}$$

Now we have isolated K and T. Next we are denoting ω and A_0 in equation (7) and (8) respectively to ω_1 along with A_{01} and ω_2 along with A_{02}. This provides the following expression for K:

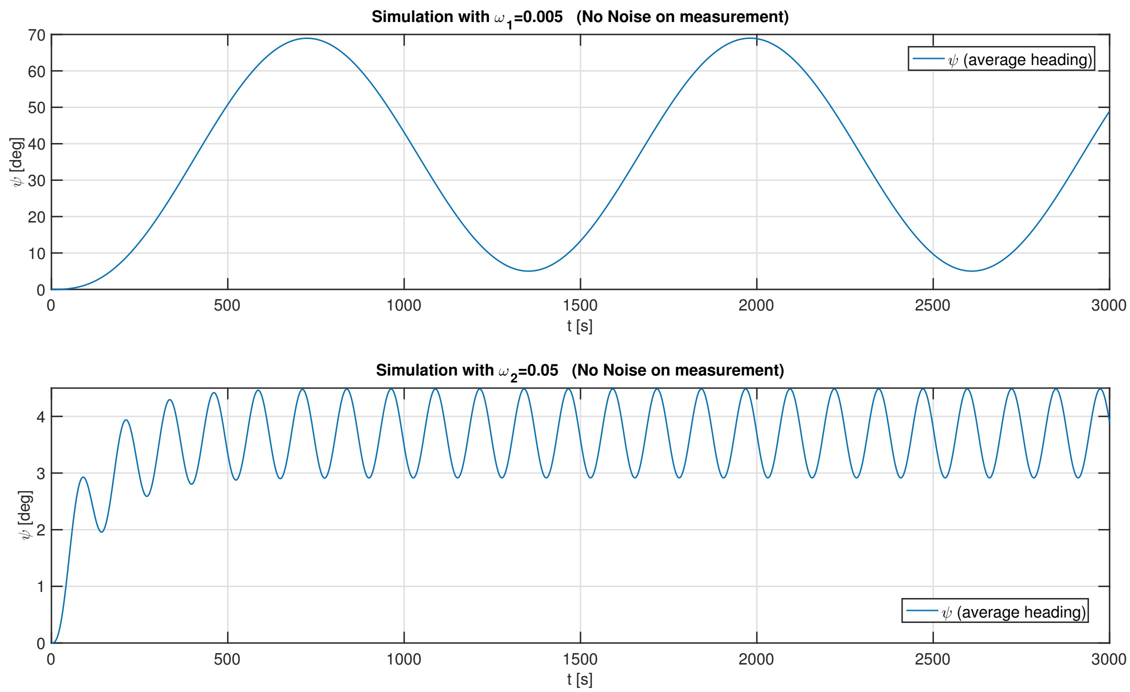
$$K = \omega_1\sqrt{\left(\frac{K^2}{{\omega_2}^4}\frac{A^2}{{A_{0_2}}^2} - \frac{1}{{\omega_2}^2}\right){\omega_1}^2 + 1}\frac{A_{0_1}}{A} \tag{9}$$

$$K = \sqrt{\frac{{A_{0_1}}^2{\omega_1}^2 - \frac{{\omega_1}^4{A_{0_1}}^2}{{\omega_2}^2}}{1 - \frac{{\omega_1}^2{A_{0_1}}^2A^2}{{\omega_2}^4{A_{0_2}}^2}}} \cdot \frac{1}{A} \tag{10}$$

The values for A_{01} and A_{02} are obtained by running the simulation without disturbances including the following constants:

$$\omega_1 = 0.005$$
$$\omega_2 = 0.05$$
$$A = 1$$

Figure 1: Average heading without disturbances A=1



From figure 1 we extract the amplitude of the outputs, respectively A_{01} and A_{02}, which are found by reading and inserting the maximum and minimum values as shown below:

$$A_{0_1} = \frac{68.98 - 5.02}{2} = 31.98$$
$$A_{0_2} = \frac{4.48 - 2.92}{2} = 0.78$$

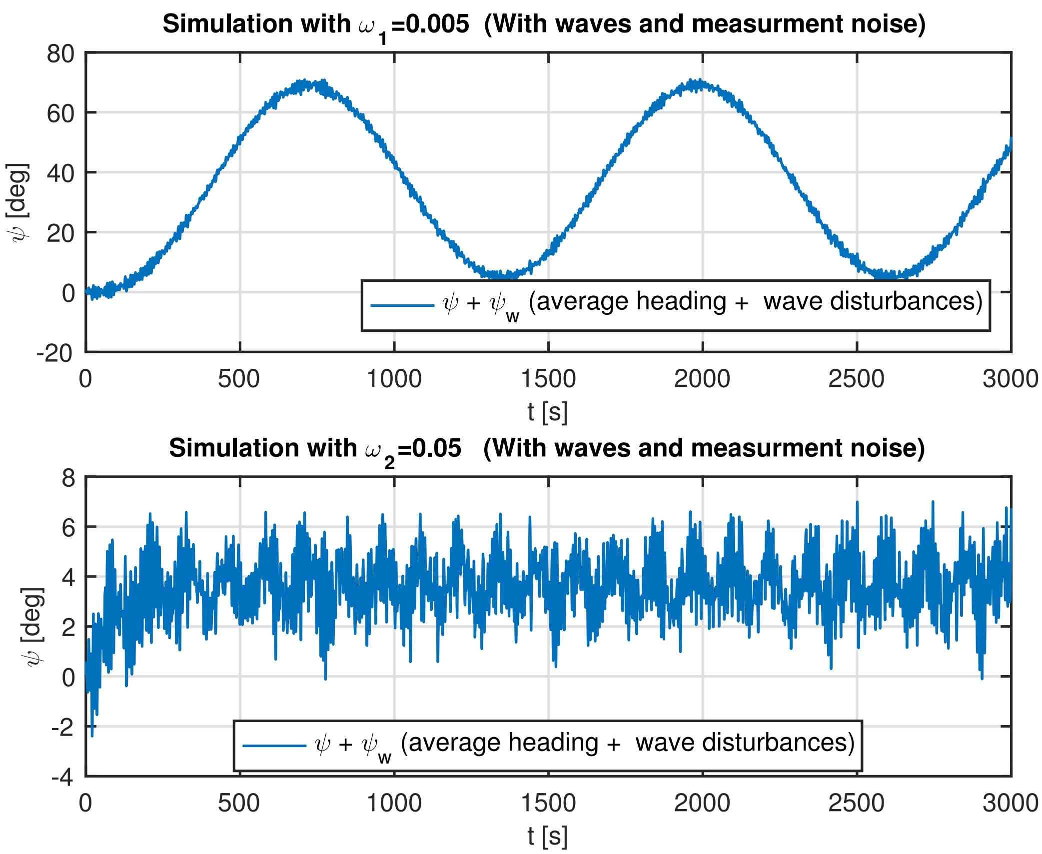
Inserting these values into equation (8) and equation (10) gives:

$$K = 0.1742$$
$$T = 86.5268$$

1.3 Boat Parameters with Wave and Measurement Noise

In this section, the previous section will be repeated, but with disturbance. The disturbance consists of waves and measurement noise. Figure 2 displays the output when $\omega = \omega_1$ and when $\omega = \omega_2$.

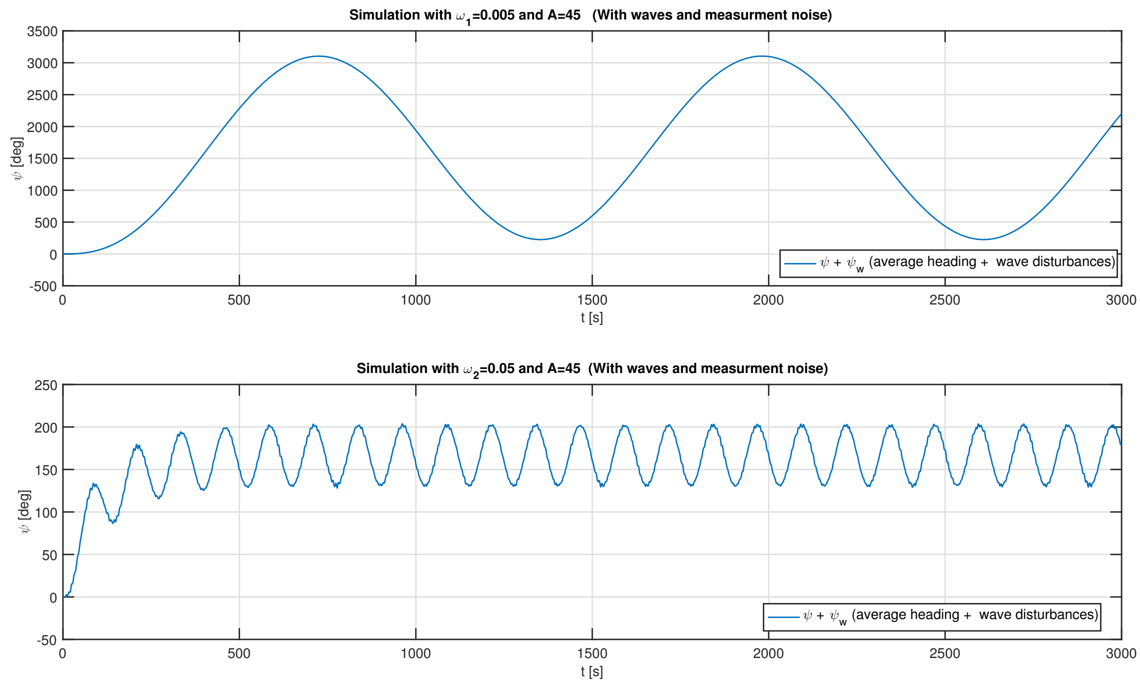
Figure 2: Average heading with waves and measurment noise, A=1



From the figure, it is rather difficult to extract any data thus the signal of the noise ratio is low. It is possible though, by using different software tools inside MatLab, but increasing the amplitude of the input signal provided us with the necessary results. An output signal with less noise is obtained by using the following parameters:

$$
\begin{aligned}
\omega_1 &= 0.005 \\
\omega_2 &= 0.05 \\
A &= 45
\end{aligned}
$$

Figure 3: Compassion with higher amplitude, A=45



These new figures displays a readable format, which is used to calculate the amplitude of the outputs, respectively A_{01} and A_{02}.

$$
\begin{aligned}
A_{0_1} &= \frac{3105 - 224.5}{2} = 1440.25 \\
A_{0_2} &= \frac{202.5 - 130.4}{2} = 36.05
\end{aligned}
$$

Inserting these values into equation (8) and equation (10) gives:

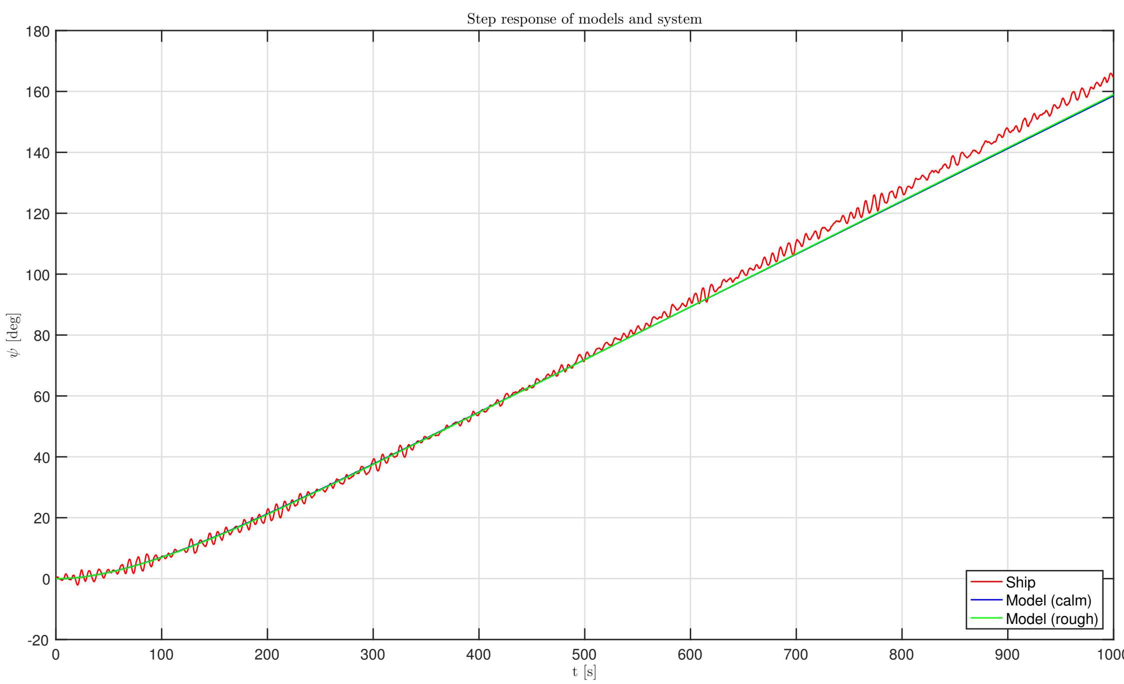
$$
\begin{aligned}
K &= 0.1734 \\
T &= 84.3920
\end{aligned}
$$

Comparing these values to what we obtained in section 1.2, we see that the deviation is rather small, less than 2.5%. The values obtained in section 1.2 are the values that will be used throughout this report.

1.4 Verifying Model Approximation

Figure 4 compares the average heading response $\psi(t)$ of the ship simulation model compared to the estimated model with and without disturbances as a way of simulating weather conditions. The plot shows a good approximation, though the deviation increases with time. Analyzing the plot, it is clear that there is something happening between 300 and 500 seconds, after this the deviation increases and the approximation starts to drift as $t \to \infty$.

Figure 4: Compassion of model and system



2 Identification of Wave Spectrum Model

In this section the Power Spectral Density (PSD) function, P_{ψ_ω} is covered.

2.1 Power Spectral Density Estimate, S_{ψ_ω}

A dataset with how waves have an impact on compass measurements is provided as a time series by ψ_ω. To find an estimate for S_{ψ_ω} we use the following MatLab script with the function $[pxx, f] = ...pwelch(x, window, noverlap, nfft, fs)$ which utilizes discrete Fourier transform to estimate the PSD function from a time series signal. The function basically returns the two-sided Welch PSD estimates at the frequencies specified in the vector, f.

MatLab-script with *pwelch*:

```
F_s = 10;
window = 4096;
noverlap = [];
nfft = [];
[S_psi, f] = pwelch(psi_w(2,:).*(pi/180),window,noverlap,
    nfft,F_s);
omega = 2*pi.*f;
S_psi = S_psi./(2*pi);
```

2.2 Analytical expressions for H_{ψ_ω} and P_{ψ_ω}

To obtain the transfer function $H_{\psi_\omega} = \frac{\psi_\omega}{\omega_\omega}$, which describes how the waves affect the yaw angle, we use the model (11) as base.

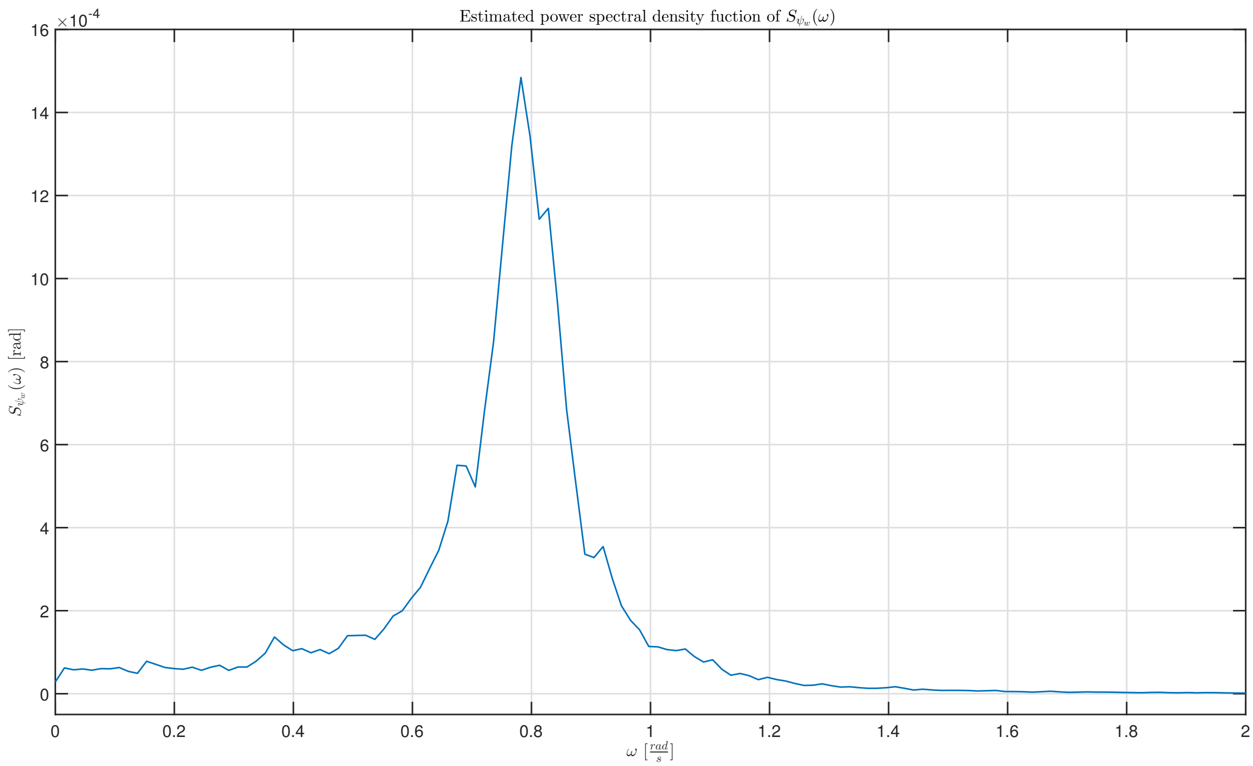
$$\begin{aligned} \dot{\xi}_w &= \psi_\omega \\ \dot{\psi}_\omega &= -{\omega_0}^2\xi_\omega - 2\lambda\omega_0\psi_\omega + K_\omega\omega_\omega \end{aligned} \tag{11}$$

Then Laplace transformation is applied to obtain the transfer function:

$$\begin{aligned} s\xi &= \psi_\omega \\ s\psi_\omega &= -{\omega_0}^2\xi_\omega - 2\lambda\omega_0\psi_\omega + K_\omega\omega_\omega \\ s\psi_\omega &= -{\omega_0}^2\psi_\omega\tfrac{1}{s} - 2\lambda\omega_0\psi_\omega + K_\omega\omega_\omega \end{aligned}$$

$$H_{\psi_\omega} = \frac{\psi_\omega}{\omega_\omega} = \frac{K_\omega s}{s^2 + 2\lambda\omega_0 s + {\omega_0}^2} \tag{12}$$

Figure 5: PSD Estimate, S_{ψ_ω}



To find an analytical expression for P_{ψ_ω}, we use the relationship expressed in (13). S_{ω_ω} is defined by it being *zero mean unity white noise*. This implies that the variance (σ^2) and spectral density (S_{ω_ω}) are equal to 1.

$$P_{\psi_\omega}(j\omega) = H_{\psi_\omega}(j\omega)H_{\psi_\omega}(-j\omega)S_{\omega_\omega} \tag{13}$$

$$P_{\psi_\omega}(\omega) = \frac{{K_\omega}^2\omega^2}{\omega^4 + 2(2\lambda^2 - 1){\omega_0}^2\omega^2 + {\omega_0}^4} \tag{14}$$

2.3 Finding ω_0

ω_0 is the base frequency of the noise ω_ω - the frequency which has the highest energy. Figure 5 shows that is around $\frac{\pi}{4} rad/s$. To find the frequency which has the highest energy, the MatLab command *[M, I] = max(A)*. The following MatLab-script were used to find ω_0:

Finding ω_0:

```
[maxPSD, frequency_index] = max(S_psi);
omega_0 = omega(frequency_index);
```

Following values were obtained:

$$\omega_0 = 0.78233 \frac{rad}{s}$$
$$\sigma^2 = 4.8724 \frac{\deg}{\frac{rad}{s}}$$

2.4 Identifying λ and fitting P_{ψ_ω}

To identify the damping factor (λ), we use curve fitting. Some of the previous calculated values will be used in this identification where we use the MatLab command *x = lsqcurvefit(fun, x0, xdata, ydata, lb, ub)*. *lsqcurvefit* solves non-linear least squares problems. *fun* is a function of *x* and *xdata* where the parameter(s) to be found is *x* and *xdata* is the abscissa data to fit into.

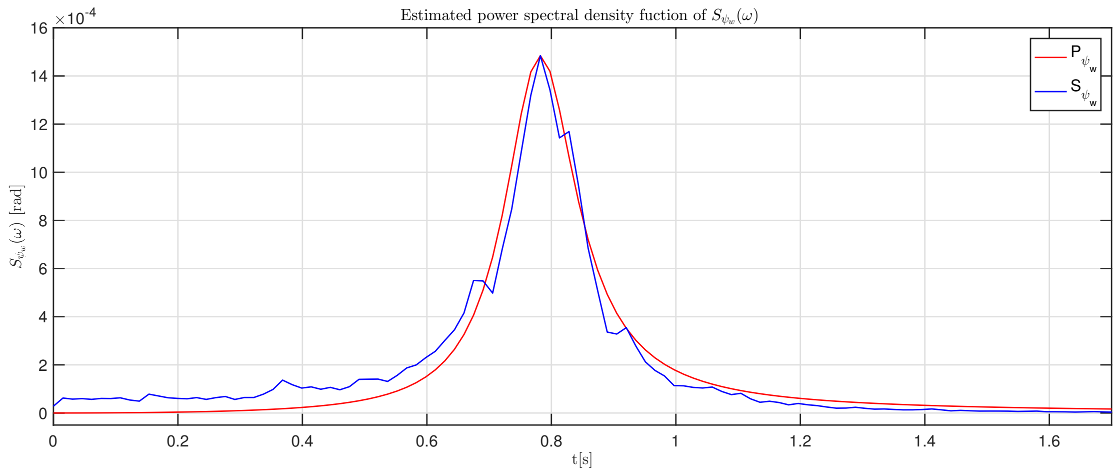
The following script provided $\lambda = 0.086198$.

Finding λ and K_ω:

```
sigma = sqrt(maxPSD);
P_psi_fun = @(lambda,omega) ...
    (4*lambda^2*omega_0^2*sigma^2*omega.^2) ./ ...
    (omega.^4 + (2*lambda^2 - 1)*2*omega_0^2*omega.^2 + ...
    omega_0^4);

lambda0 = 10;
lb=0;
ub=10;
lambda = lsqcurvefit(P_psi_fun,lambda0,omega,S_psi,lb,ub);
K_w = 2*lambda*omega_0*maxPSD;
P_psi = P_psi_fun(lambda,omega);
```

Figure 6: Comparsion



3 Control System Design

In this section the autopilot will be developed. The autopilot uses the desired course angle, ψ_r as reference. For the MatLab simulations in this project, $\psi_r = 30°$, as the linearized model only holds for small deviations in ψ.

3.1 PD Controller Design

From earlier we had the following equation, which the PD controller is based on:

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

This provides the following transfer function for our PD controller:

$$H_{pd}(s) = K_{pd}\frac{1+T_d s}{1+T_f s} \tag{15}$$

T_d is chosen such that the time constant term from the ship transfer function is cancelled, $T_d = T$, and we obtain the open-loop transfer function:

$$H(s) = H_{pd}H_{ship}\frac{KK_{pd}}{T_f s^2 + s} \tag{16}$$

Phase margin and cut-off frequency are respectively chosen to be $\phi = 50°$ and $\omega_c = 0.1 rad/s$. These parameters will help us find K_{pd} and T_f, thus the relationship betweeen cut-off frequency and phase margin are:

$$\phi = 180^o + \angle H(j\omega c) \tag{17}$$

$$1 = H(j\omega c)H(-j\omega c) \tag{18}$$

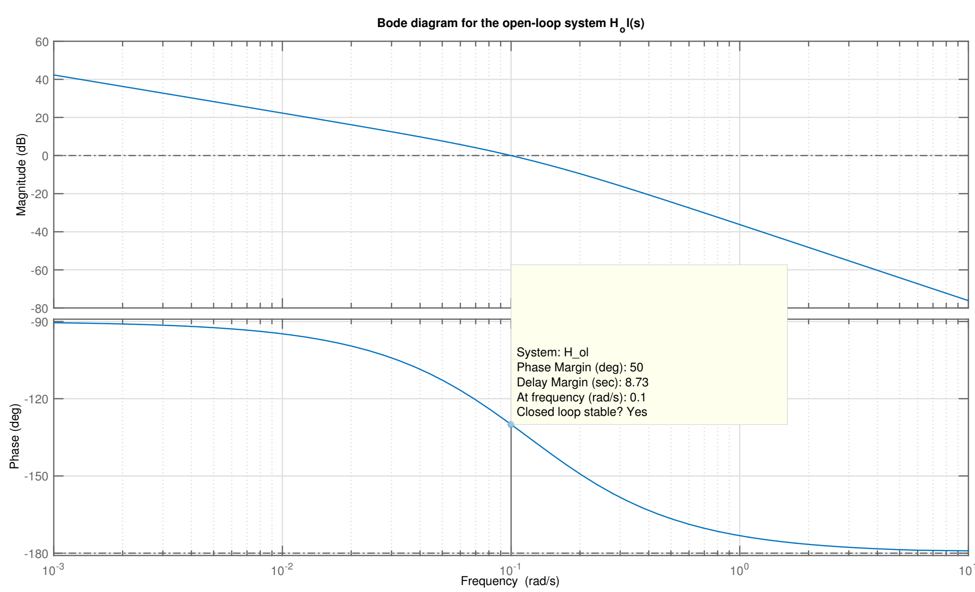
(17) and (18) provide following results for K_{pd} and T_f:

$$T_f = \frac{1}{\tan \phi \omega_c} = 8.391$$

$$K_{pd} = \sqrt{\frac{{T_f}^2 {\omega_c}^4 + {\omega_c}^2}{K^2}} = 0.7493$$

The Bode diagram in Figure 7 shows the results obtained.

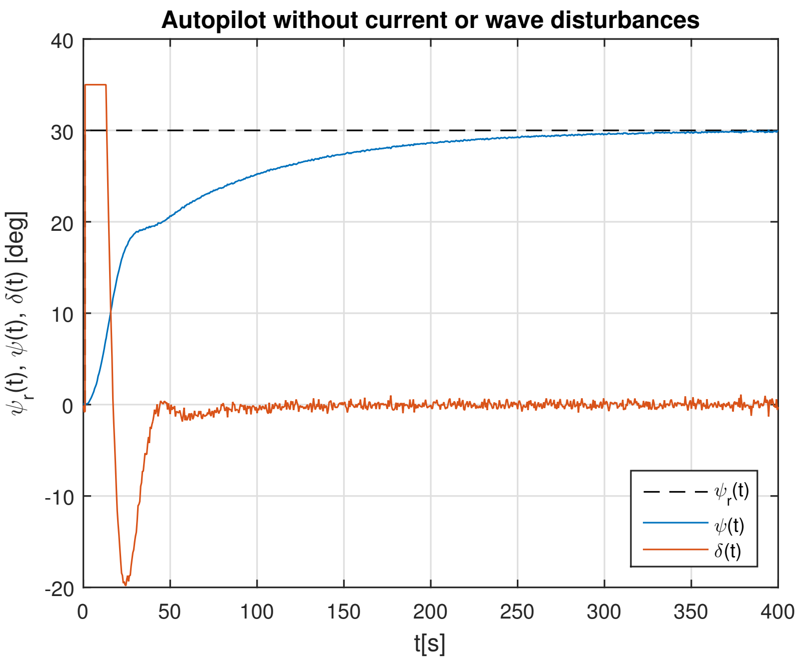
Figure 7: Bode diagram



3.2 Simulating without disturbances

In figure 8 the system simulated without any form of disturbance is displayed. The system is overdamped, but hence its fast convergence to the reference it is only slightly overdamped. The plot of the rudder angle (δ) shows that the actuation is noisy, caused by measurement noise. This might cause undesirable wear on the rudder.

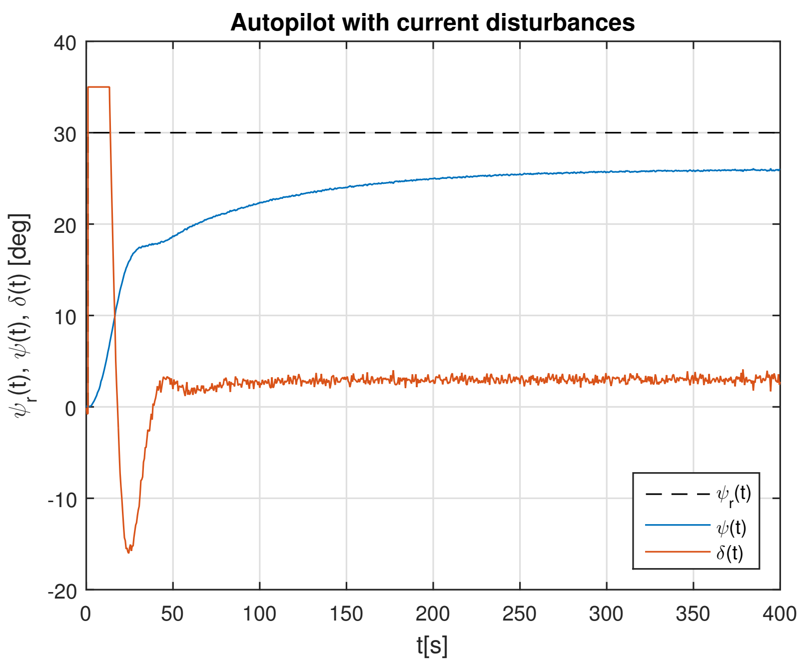
Figure 8: Autopilot without disturbances



3.3 Simulating with current disturbance

This system is simulated with a current disturbance, displayed in figure 9. In the model, the effect of the current is a rudder angle bias (b). The rudder angle bias (b) gives the system a steady-state error, since there is no feed forward or integral action in the controller.

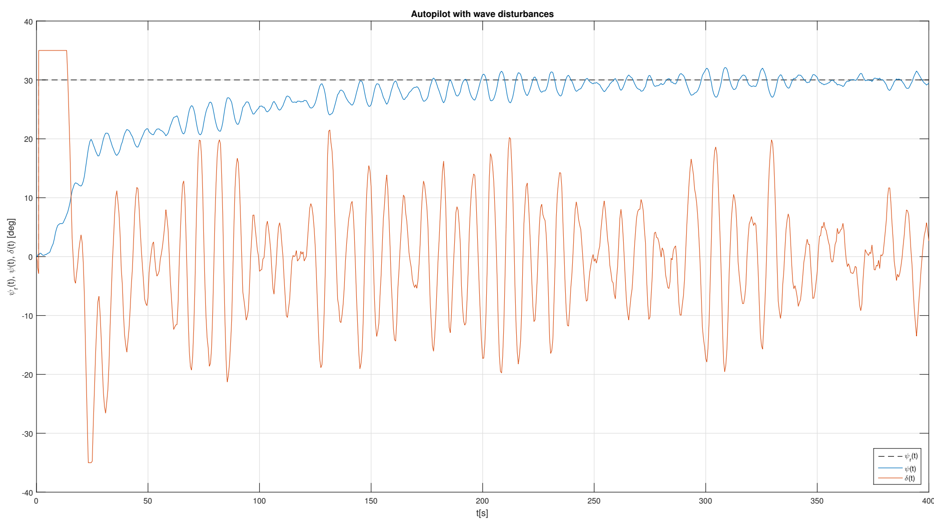
Figure 9: Autopilot with current disturbances



3.4 Simulating with wave disturbance

In figure 10 the system simulated with wave disturbance is displayed. The waves cause a disturbance on the yaw angle (ψ_ω). This disturbance is of high frequency with relative high amplitude, which makes the actuation of the rudder noisy.

Figure 10: Autopilot with wave disturbances



4 Observability

4.1 Derivation of State Space Matrices

The derivation of the State Space Matrices is done with the state vector, input and disturbance as shown here:

$$x = \begin{bmatrix} \xi_\omega \\ \psi_\omega \\ \psi \\ r \\ b \end{bmatrix}, \quad u = \delta, \quad \omega = \begin{bmatrix} \omega_\omega \\ \omega_b \end{bmatrix} \tag{19}$$

We have the following model:

$$\begin{aligned} &\dot{\xi}_2 = \psi_\omega \\ &\dot{\psi}_\omega = -{\omega_0}^2 \xi_\omega - 2\lambda\omega_0\psi_\omega + K_\omega\omega_\omega \\ &\dot{\psi} = r \\ &\dot{r} = -\tfrac{1}{T} + \tfrac{K}{T}(\delta - b) \\ &\dot{b} = \omega_b \\ &y = \psi + \psi_\omega + v \end{aligned} \tag{20}$$

The system can be written as:

$$\dot{x} = \begin{bmatrix} x_2 \\ -{\omega_0}^2 x_1 - 2\lambda\omega_0 x_2 + K_\omega \omega_1 \\ x_4 \\ -\tfrac{1}{T}x_4 - \tfrac{K}{T}x_5 + \tfrac{K}{T}u \\ \omega_2 \end{bmatrix}, \quad y = x_2 x_3 + v \tag{21}$$

This corresponds to a system on the following form:

$$\begin{aligned} &\dot{x} = Ax + Bu + E\omega \\ &y = Cx + v \end{aligned} \tag{22}$$

with matrices:

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

$$E = \begin{bmatrix} 0 & 0 \\ K_\omega & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

To check the observability of the system, we use these following parameters that were calculated in previous sections:

$$\begin{aligned} &K = 0.1742 \\ &T = 86.5268 \\ &\omega_0 = 0.7823 \\ &\lambda = 0.0862 \end{aligned} \tag{23}$$

4.2 Observability without disturbances

Without disturbances, both the state vector from equation (19) and the matrices A and C are rewritten as:

$$x = \begin{bmatrix} \psi \\ r \end{bmatrix} \tag{24}$$

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

This provides an observability matrix which has a rank of 2, meaning the system is observable.

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

4.3 Observability with current disturbance

When current disturbance is applied, it is necessary to rewrite both the state vector and the matrices A and C again, on the form:

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

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

The observability matrix for the system with current disturbance is given by (28), and by using the MatLab command $rank(Ob)$, we can see that it has full row rank of 3, meaning the system is observable.

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

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.0116 & -0.0020 \end{bmatrix} \tag{28}$$

4.4 Observability with wave disturbance

With wave disturbance is applied, the state vector and the matrices A and C are rewritten on the form:

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

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

Using the same MatLab command as in the previous section, we find that the observability matrix has a rank of 4, meaning that this fourth order system is observable.

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -{\omega_o}^2 & -2\lambda\omega_0 & 0 & 1 \\ 2\lambda{\omega_0}^3 & 4\lambda^2{\omega_0}^2 - {\omega_0}^2 & 0 & -\frac{1}{T} \\ -{\omega_0}^4(4\lambda^2-1) & -8\lambda^3{\omega_0}^3 + 4\lambda{\omega_0}^3 & 0 & \frac{1}{T^2} \end{bmatrix}$$

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -0.6120 & -0.1239 & 0 & 1 \\ 0.0825 & -0.1349 & 0 & -0.0116 \\ 0.3635 & 0.1626 & 0 & 0.001 \end{bmatrix} \tag{31}$$

4.5 Observability with wave and current disturbances

With both wave and current disturbances, we use the state vector from (19) with corresponding matrices from section 4.1 to obtain the observability matrix shown in (32). This observability matrix has a rank of 5, meaning this fifth order system is observable.

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ -{\omega_0}^2 & -2\lambda\omega_0 & 0 & 1 & 0 \\ 2\lambda{\omega_0}^3 & 4\lambda^2{\omega_0}^2 - {\omega_0}^2 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ -{\omega_0}^4(4\lambda^2 - 1) & -8\lambda^3{\omega_0}^3 + 4\lambda{\omega_0}^3 & 0 & \frac{1}{T^2} & \frac{K}{T^2} \\ 8\lambda^3{\omega_0}^5 + 4\lambda{\omega_0}^5 & 16\lambda^4{\omega_0}^4 + 12\lambda^2{\omega_0}^4 + {\omega_0}^4 & 0 & -\frac{1}{T^3} & -\frac{K}{T^3} \end{bmatrix}$$

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ -0.6120 & -0.1349 & 0 & 1 & 0 \\ 0.0825 & -0.5939 & 0 & -0.0116 & -0.0020 \\ 0.03635 & 0.1626 & 0 & 0.0001 & 2.3 \cdot 10^{-5} \\ -0.0995 & 0.3415 & 0 & -1.5 \cdot 10^{-6} & -2.7 \cdot 10^{-7} \end{bmatrix} \tag{32}$$

5 Kalman Filter

In this section we implement a discrete Kalman filter to estimate the heading ψ, the bias b and the high-frequency wave-induced motion on the heading ψ_ω. Instead of the compass measurement, we will use the estimated ψ for feedback in the control law, also known as wave filtering.

5.1 Discretization

A discrete model of the system is required to implement the discrete Kalman filter. This is done by using the continous state-space model from section 4.1 in MatLab where exact discretization is used along with a sampling frequency of 10Hz. The MatLab script below shows how this is done by discretizing the matrices in two steps with the function $[Ad, Bd] = c2d(A, B, T)$, once with $\mathbf{B}$ as input matrix and then once with $\mathbf{E}$ considering the white noise $\mathbf{w}$ as input.

Discretization of model:

```
f_s=10;             % Sampling frequency [Hz]
T_s = 1/f_s;        % Sampling time [s]

% Discretize model
[Ad,Bd] = c2d(A,B,T_s);
% Discretize model
[~,Ed] = c2d(A,E,T_s);
Cd=C;
```

Using zero-order hold on the input and sample time T we descitize the matrices as follows:

$$\begin{aligned} A_d &= e^{AT} \\ B_d &= \left(\int_0^T e^{A\tau} d\tau \right) B \\ C_d &= C \end{aligned} \tag{33}$$

The resulting matrices are shown below:

$$A_d = \begin{bmatrix} 0.9970 & 0.0992 & 0 & 0 & 0 \\ -0.0607 & 0.9836 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & 0 \\ 0 & 0 & 0 & 0.9988 & -0.0002 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$E_d = \begin{bmatrix} 0.0015 & 0 \\ 0.0295 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.1 \end{bmatrix}, \quad B_d = 1 \cdot 10^{-3} \cdot \begin{bmatrix} 0 \\ 0 \\ 0.0101 \\ 0.2012 \\ 0 \end{bmatrix}$$

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

5.2 Estimation of measurement noise variance

By simulating the ship with zero input(no waves or current), and by applying the measurement noise we were able to obtain an estimate of the belonging variance. In theory, this would mean that the ship would keep a constant heading, $\psi = 0$, providing the measurement noise as resulting signal. Below you can see how this was extracted in MatLab.

```

simTime = 600; % [s]
simout = sim('ship5b','startTime','0','stopTime',sprintf(
    '%d',simTime)); % no noise
psi = simout.get('compass');
R = var(psi*pi/180);
```

5.3 Implementing the Kalman Filter

The Kalman filter were implemented using a normal MatLab function within the Matlab function Simulink block, according to Appendix B in the assignment text with persistent variables.

In section 5.2 we obtained the variance which in this case is divided by the sample time $T = 0.1s$ and used as the variance of the measurement noise in the filter, $E\{v^2\} = R$. To obtain the desirable units for the filter model, conversions are applied inside MatLab and Simulink.

The Kalman filter is defined as:

$$i_{KF} = \begin{bmatrix} \delta & y \end{bmatrix}^T \tag{34}$$

$$o_{KF} = \begin{bmatrix} \xi_\omega & \psi_\omega & \psi & r & b \end{bmatrix}^T \tag{35}$$

(34) defines the input to the Kalman filter, while (35) defines the output. p_{ii} are diagonal elements of P_k, which corresponds to the variance of the estimation error of $\hat{\psi}_\omega$, $\hat{\psi}$ and $\hat{b}$ respectively.

A zero-order hold is applied to the input of the Kalman filter, with the same sampling time used for the discretization. The Kalman filter block inherits its sampling time from the preceding block, and even though the Kalman filter runs in $10Hz$, the filter model from section 5.1 is still valid. The Kalman filter equations are written in a matlab function called, defined as

$$K_k = P_k^- C^T (C P_k^- C^T + R)^{-1} \tag{36}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (y_k - C \hat{x}_k^-) \tag{37}$$

$$P_k = (I - K_k C) P_k^- (I - K_k C)^T + K_k R K_k^T \tag{38}$$

$$\hat{x}_{k+1} = A \hat{x}_k + B\delta \tag{39}$$

$$P_{k+1}^- = A P_k A^T + E Q E^T \tag{40}$$

5.4 Feed forward from estimated bias

A feed forward is made from the estimated bias to cancel the bias due to current. This implies that the rudder input is now the output of the PD-controller plus the estimated bias $\hat{b}$. Figure 12 shows the response of the ship when $\psi_r = 30$ and current disturbance is applied. Comparing this response with Figure 9, we could clearly see that the feed forward cancels the bias, and the ship is able to slowly reach its desired heading reference. The rudder input is still oscillating due to measurement noise in the feedback loop.

Figure 11 shows that the bias estimate converges in a satisfying way, and how the rudder input oscillates.

Figure 11: Estimated bias and rudder input $\hat{b}$

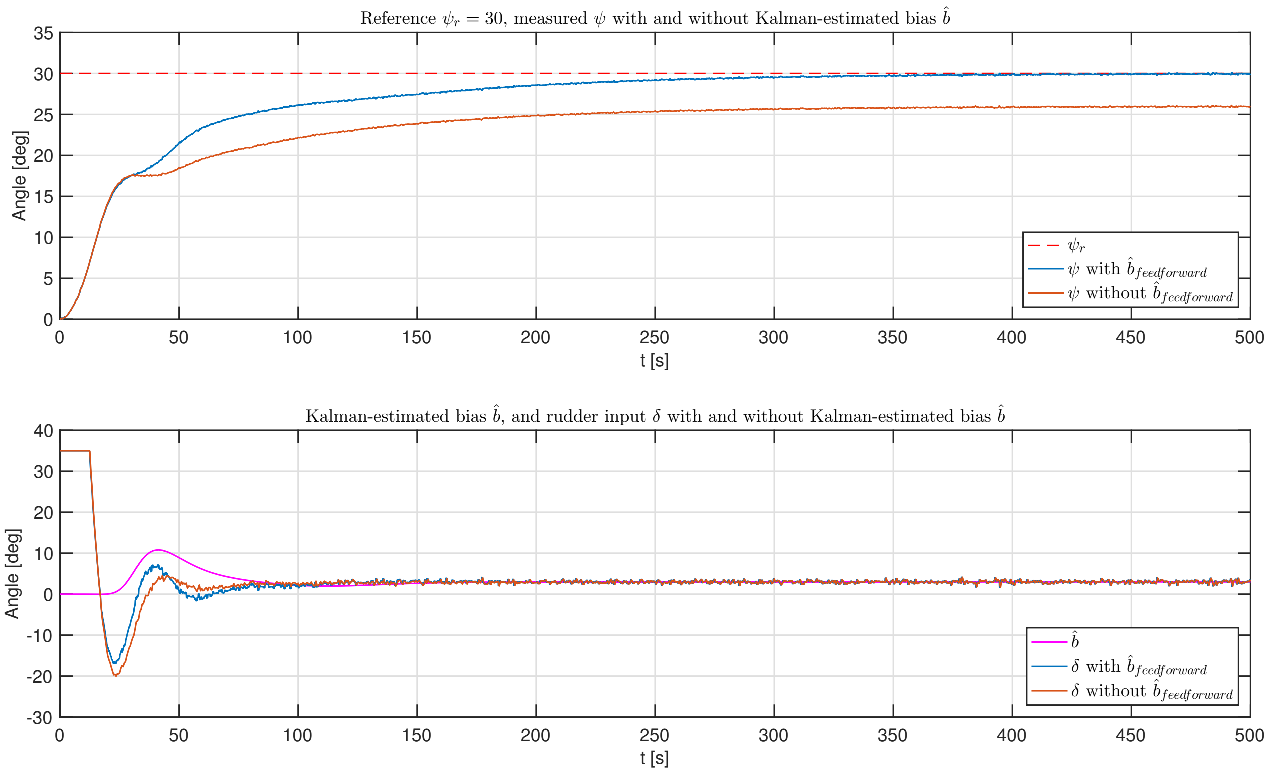
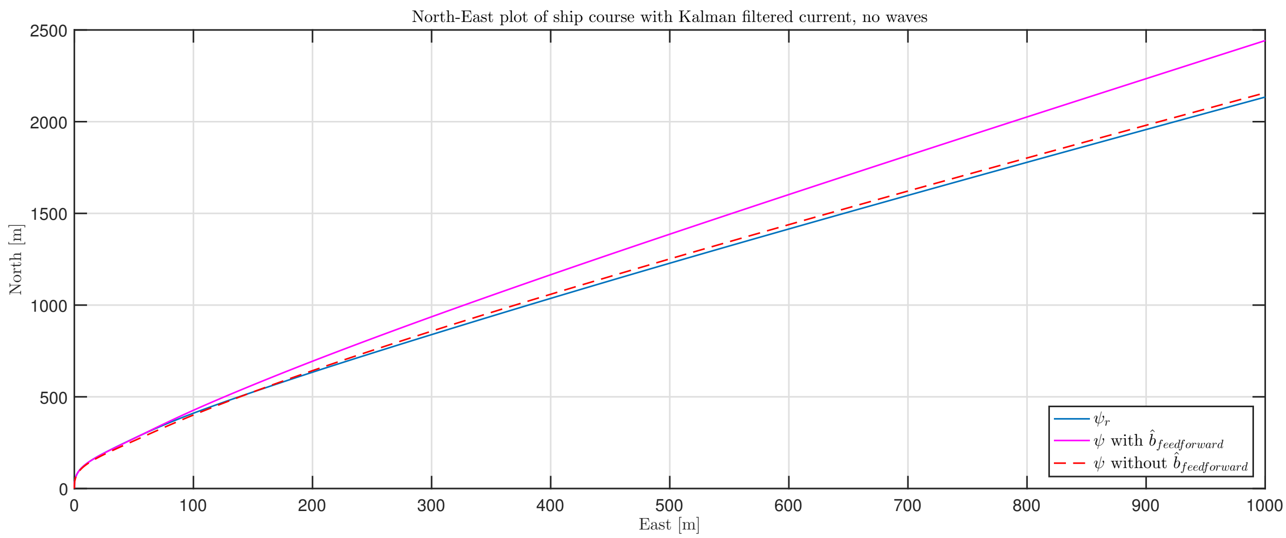


Figure 12: North-East plot of ship course with and without Kalman



5.5 Wave filtering

In this section, the estimated heading $\hat{\psi}$ will be used instead of the measured heading for the feedback in the autopilot, in addition to the feed forward from the estimated bias. This system is also simulated with $\psi_r = 30$, but now with both current and wave disturbances applied. From Figure 13 we can see the estimated compass $\hat{\psi}$ and the measured compass ψ, with the estimated having much less oscillations. In Figure 14 we can see the rudder input and estimated bias, with and without Kalman. Figure 15 shows the heading of the ship with and without Kalman, while Figure 16 shows that the wave bias estimator follows the measured bias, meaning we have a good estimator.

Figure 13: Measured compass and estimated compass

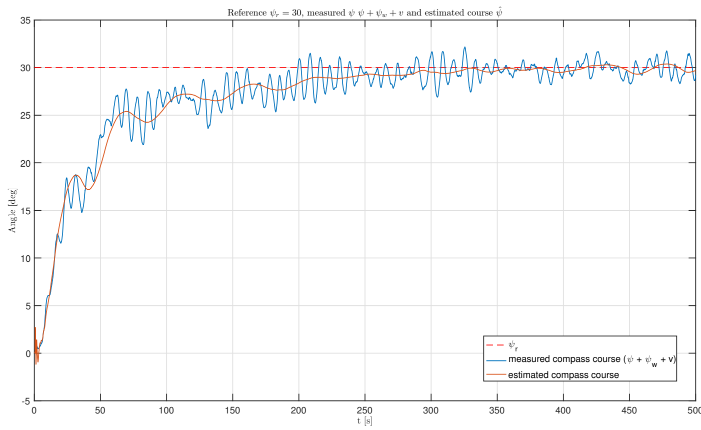


Figure 14: Rudder input and estimated bias

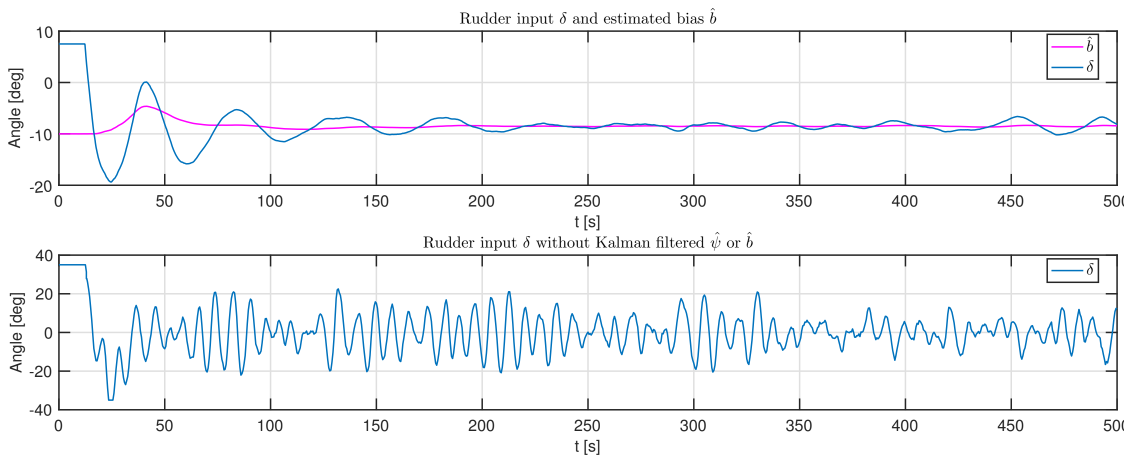


Figure 15: North-East plot with Kalman filtered current and waves

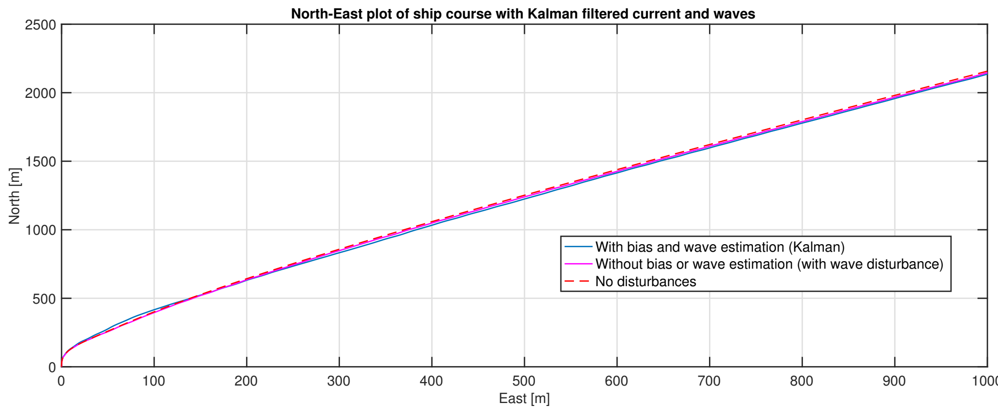
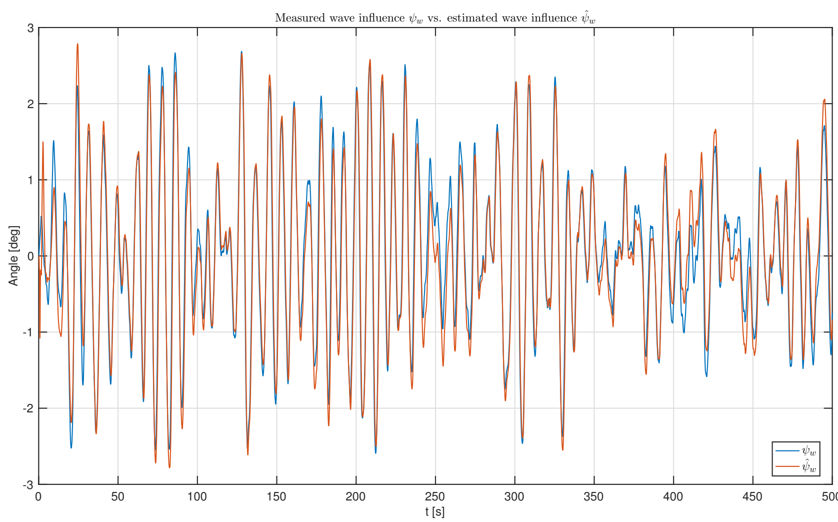


Figure 16: Wave influence



6 Conclusion

After identifying model parameters in section 1, we were able to develop a feasible approximation of the ship model. By analyzing the input and output responses, a ship model with measurement noise, wave disturbance and current disturbance were developed.

The implementation of the discrete Kalman filter was intended for improvement of the control of the ship. To make this implementation possible, we identified the necessary wave base frequency ω_0 and the wave damping factor λ in section 2. Figure 6 shows that the analytically derived and the estimated PSD functions cohere.

A PD controller is designed in section 3 to keep angle of the heading at a desired value. Tuning of this particular controller is done so that the damping part cancels the ship time constant, putting the pgase margin and cut-off frequency values respectively to $\phi = 50^o$ and $\omega_c = 0.1\frac{rad}{s}$. These characteristics are proved in figure 7.

In section 4 we check the observability of the system. This is done by transforming the system into state-space form, and calculating the observability matrices in MatLab. After checking the rank of each observability matrix in section 4, we could declare the system observable independent of the composition of disturbances.

The final part of this assignment was completed in section 5, where a discrete Kalman filter is applied. This was done through discretization of the system using MatLab before the Kalman filter was implemented using a Matlab function within the Matlab Simulink Block with persistent variables, according to Appendix B in the assignment text.

7 Appendix A: MatLab Code

7.1 Part 1 - Identification of boat parameters

```
%% 1b Simulating in calm water(no disturbances) and identification of parameter T and K
omega = 0.005; %rad/s w_1
A=1;            %Amplitude
simTime = 1000; % [s]
simout = sim('ship1b','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');     %get time from workspace
psi_w1 = simout.get('psi'); %get psi from workspace

% run the simulation with  w_2
omega = 0.05; %[rad/s] w_2
A=1;           %Amplitude

simTime = 1000; % [s]
simout = sim('ship1b','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');     %get time from workspace
psi_w2 = simout.get('psi'); %get psi from workspace

%plotting into one window
subplot(2,1,1)
plot(time,psi_w1)
xlabel('t [s]')
ylabel('$\psi$ [deg]')
title('Simulation with $\omega_1=0.005$   (No Noise on measurement)')
legend('\psi (average heading) ', 'Location','SouthEast')
grid on;
subplot(2,1,2)
plot(time,psi_w2)
xlabel('t [s]')
ylabel('$\psi$ [deg]')
title('Simulation with $\omega_w=0.05$   (No Noise on measurement)')
legend('\psi (average heading) ', 'Location','SouthEast')
grid on;

%%code that identifies the T and K value


```

```matlab
%% 1c: Simulating in rough weather(Waves+Noise) Amplitude =1
omega = 0.005; %rad/s %w_1
A=1;            %Amplitude
simTime = 1000; % [s]
simout = sim('ship1c','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');      %get time from workspace
psi_w1_rough = simout.get('psi'); %get psi from workspace

%simulate again for w_2
omega = 0.05;  %rad/s %w_w
A=1;            %Amplitude
simTime = 1000; % [s]
simout = sim('ship1c','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');      %get time from workspace
psi_w2_rough = simout.get('psi'); %get psi from workspace

%Ploting the figures
subplot(2,1,1)
plot(time,psi_w1_rough)
xlabel('t [s]')
ylabel('$\psi$ [deg]')
title('Simulation with $\omega_1=0.005$  (With waves and measurement noise) ')
legend('\psi + \psi_{w} (average heading +  wave disturbances) ', 'Location','SouthEast')
grid on;
subplot(2,1,2)
plot(time,psi_w2_rough)
xlabel('t [s]')
ylabel('$\psi$ [deg]')
title('Simulation with $\omega_w=0.05$   (With waves and measurement noise)')
legend('\psi + \psi_{w} (average heading +  wave disturbances) ', 'Location','SouthEast')
grid on;


%% 1c: simulation in rough weather (waves+noise) with higher amplitude A=45

omega = 0.005; %rad/s %w_1
A=45;           %Amplitude
simTime = 1000; % [s]
```

```
simout = sim('ship1c','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');     %get time from workspace
psi_w1_rough_A45 = simout.get('psi'); %get psi from workspace

%simulate again for w_2
omega = 0.05;  %rad/s %w_w
A=45;              %Amplitude
simTime = 1000; % [s]
simout = sim('ship1c','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');     %get time from workspace
psi_w2_rough_A45 = simout.get('psi'); %get psi from workspace

%Plotting the figures
subplot(2,1,1)
plot(time,psi_w1_rough_A45)
xlabel('t [s]')
ylabel('$\psi$ [deg]')
title('Simulation with $\omega_1=0.005$ and $A=45$   (With waves and measurment noise) ')
legend('\psi + \psi_{w} (average heading +  wave disturbances) ', 'Location','SouthEast')
grid on;
subplot(2,1,2)
plot(time,psi_w2_rough_A45)
xlabel('t [s]')
ylabel('$\psi$ [deg]')
title('Simulation with $\omega_w=0.05$ and $A=45$   (With waves and measurment noise)')
legend('\psi + \psi_{w} (average heading +  wave disturbances) ', 'Location','SouthEast')
grid on;

%% 1d: Comparison of all models
%Make The Transfer Functions
num1=[0.1742];        %with K
den1=[86.52685 1 0]; %with T
H1=tf(num1,den1);     %No noise Transfer Function

num2=[0.1734];        %with K
den2=[84.3920 1 0];  %with T
H2=tf(num2,den2);     %With noise Transfer Function

```

```
%simulate
simTime = 1000; % [s]
simout = sim('ship1d','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');
psi_s = simout.get('psi'); %system
psi_no_noise = simout.get('psi_no_noise'); %model with no noise
psi_with_noise = simout.get('psi_with_noise'); %model with noise

%draw plot
plot(time,psi_s,'r',time,psi_with_noise,'b',time,psi_no_noise,'g')
xlabel('t [s]')
ylabel('$\psi$ [deg]')
legend('Ship','Model (calm)','Model (rough)','Location','southeast')
title('Step response of models and system')
grid on
```

7.2 Part 2 - Identification of wave spectrum model

```
%% 2a: Estimate PSD
load('wave.mat');        % Load wave disturbance

F_s = 10;
window = 4096;
noverlap = [];
nfft = [];
[S_psi,f] = pwelch(psi_w(2,:).*(pi/180),window,noverlap,nfft,F_s);
omega = 2*pi.*f;
S_psi = S_psi./(2*pi);


%% 2c: Find omega_0
% Plot estimated PSD

plot(omega,S_psi, 'LineWidth', 2)
axis([0 2 -0.00005 16*10^(-4)])
hold on
xlabel('$\omega$ [$\frac{rad}{s}$]')
ylabel('$S_{\psi_{w}}(\omega)$ [rad]')
```

```
title(['Estimated power spectral density fuction of $S_{\psi_{w}}(\omega)$ '...
    ])
grid on;


%% 2c: Find resonance frequency from estimated PSD)
[maxPSD,  frequency_index ] = max( S_psi )
omega_0 = omega( frequency_index )

%% 2d: Comparison
sigma = sqrt(maxPSD); % sigma_squared is the peak value of S_psi_w

%using lsqcurvefit

P_psi = @(lambda,omega) ...
    (4*lambda^2*omega_0^2*sigma^2*omega.^2) ./ ...
    (omega.^4 + (2*lambda^2 - 1)*2*omega_0^2*omega.^2 + ...
    omega_0^4);

lambda0 = 10;
lb=0;
ub=10;

lambda = lsqcurvefit(P_psi,lambda0,omega,S_psi,lb,ub);
P_psi = P_psi(lambda,omega);
K_w = 2*lambda*omega_0*maxPSD;

%%% Comparison plot of estimate and analytical
figure
plot(omega, P_psi, 'r')
hold on
plot(omega, S_psi, 'b')
legend('P_{\psi_w}','S_{\psi_w}')
xlim([0 2])
xlabel('t [s]')
ylabel('PSD [deg^2/(rad/s)')
title('Comparison of estimated PSD function $(S_{\psi_w})$ and the analytical $P_{\psi_w}$');
grid on;
```

7.3 Part 3 - Control system design

```
%%From earlier excercies
K=0.1734;
T=84.3920;
%% 5.3a
w_c=0.1; %cutoff frequency [rad/s]
PM=50/180*pi;%Phase margin [rad]
T_d=T;          %chosen such that it cancels the TF time constant
%Make transfer function for controller
T_f=1/(tan(PM)*w_c);
K_pd=sqrt((T_f^2*w_c^4+w_c^2)/K^2);
num_controller=[K_pd*T_d,K_pd];
den_controller=[T_f,1];

H_pd=tf(num_controller,den_controller); %make transfer function for controller

%%Make transfer function for plant
H_ship=tf([K],[T 1 0]);                      %transfer function for plant

%open-loop system
H_ol=H_pd*H_ship;                            %Open loop transfer function

%draw a bode idagram
figure
bode(H_ol);
grid ;
title('Bode plot for the open-loop system H_ol(s)');



%% 5.3b)

ref=30;
simTime=400;

simout = sim('ship3b','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');      %get time from workspace
psi = simout.get('psi'); %get psi from workspace
ref=simout.get('ref');
```

```
%error=simout.get('error'); % no need for this, not gonna plot it.
delta=simout.get('rudder_input');

%Ploting the graphs
figure
plot(time,ref,'black--',time,psi,time,delta) %plots time against refrence, output and rudder_input
legend('\psi_r(t)','\psi(t)','\delta(t)','location','Southeast')
xlabel('t[s]');
ylabel('\psi_r(t), \psi(t), \delta(t) [deg]');
title('Autopilot without current or wave disturbances')
%make y-axis formating become less
grid on;

%% 5.3c)

ref=30;
simTime=400;

simout = sim('ship3c','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');      %get time from workspace
psi = simout.get('psi'); %get psi from workspace
ref=simout.get('ref');
%error=simout.get('error'); % no need for this, not gonna plot it.
delta=simout.get('rudder_input');

%Ploting the graphs
figure
plot(time,ref,'black--',time,psi,time,delta) %plots time against refrence, output and rudder_input
legend('\psi_r(t)','\psi(t)','\delta(t)','location','Southeast')
xlabel('t[s]');
ylabel('\psi_r(t), \psi(t), \delta(t) [deg]');
title('Autopilot with current disturbances')
%make y-axis formating become less
grid on;

%% 5.3d)

ref=30;
simTime=400;
```

```
simout = sim('ship3d','startTime','0','stopTime',sprintf('%d',simTime));
time = simout.get('time');      %get time from workspace
psi = simout.get('psi'); %get psi from workspace
ref=simout.get('ref');
%error=simout.get('error'); % no need for this, not gonna plot it.
delta=simout.get('rudder_input');

%Ploting the graphs
figure
plot(time,ref,'black--',time,psi,time,delta) %plots time against refrence, output and rudder_input
legend('\psi_r(t)','\psi(t)','\delta(t)','location','Southeast')
xlabel('t[s]');
ylabel('\psi_r(t), \psi(t), \delta(t) [deg]');
title('Autopilot with wave disturbances')
%make y-axis formating become less
grid on;
```

7.4 Part 4- Observability

```
load('constants');
%% 5.4a): Finding the 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];
E = [0 0; K_w 0; 0 0; 0 0; 0 1];

%% 5.4b): Observabillity without disturbances
A = [0 1; 0 -1/T];
B=[0; K/T];
C= [1 0];
O = obsv(A,C);
rank(O) % rank(O) = 2 == n -> observable

%% 5.4c) 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);
```

```
rank(O_b)    %rank(O_b) = 3 == n -> observable


%% 5.4d) Observabillity with waves
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);
rank(O_w) % rank(Ob) = 4 == n -> observable

%% 5.4e) Observabillity with current and wave
O_cw = obsv(A,C); % rank(Ob) = 5 == n -> observable
```

7.5 Part 5 - Discrete Kalman filter function

```
function [b,psi] = fcn(u, y, data)

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 - go to next
    x_ = A*x + B*u;
    P_ = A*P*A' + E*Q*E';

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

```
%% 5.1a):
load('constants');

```

```
%Make transfer function for controller
T_f=1/(tan(PM)*w_c);
K_pd=sqrt((T_f^2*w_c^4+w_c^2)/K^2);
num_controller=[K_pd*T_d,K_pd];
den_controller=[T_f,1];

% Discretize model
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];

f_s=10;          % Sampling frequency [Hz]
T_s = 1/f_s;     % Sampling time [s]

% Discretize model
[Ad,Bd] = c2d(A,B,T_s);
% Discretize model
[~,Ed] = c2d(A,E,T_s);
Cd=C;

%% 5b: measure variance

simTime = 600; % [s]
simout = sim('ship5b','startTime','0','stopTime',sprintf(
    '%d',simTime)); % no noise
psi = simout.get('compass');
R = var(psi*pi/180);

%% 5c: Discrete Kalman Filter

Q = [30 0; 0 10^(-6)];
R = R/T_s;
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];
I = diag([1 1 1 1 1]);

% data struct to use in matlab function
 data = struct('Ad',Ad,'Bd',Bd,'Cd',Cd,'Ed', Ed, 'Q',Q,'R
    ', R,'P_0',P_0,'X_0',X_0, 'I', I);


```

```matlab
%% 5d: Feed forward estimated bias

load_system('ship5d.slx')
sim('ship5d.slx')

% Plot measured compass with and without estimated bias
figure(figNum)
figNum = figNum+1;
subplot(2,1,1)
plot(time,ref, 'r--', time, compass, time, compass_kalman)
title(['Reference $\psi_{r} = 30$, measured $\psi$ with and'...
' without Kalman-estimated bias $\hat{b}$']);
xlabel('t [s]');
ylabel('Angle [deg]');
legend({'$\psi_{r}$', '$\psi$ with $\hat{b}_{feedforward}$', ...
    '$\psi$ without $\hat{b}_{feedforward}$'},    ...
    'Interpreter', 'latex', 'Location', 'best')
grid  on ;

subplot(2,1,2)
plot(time, bias,'m', time, rudder_input, time, rudder_input_2)
title(['Kalman-estimated bias $\hat{b}$, and rudder input $\delta$ ' ...
'with and without Kalman-estimated bias $\hat{b}$'], ...
 'Interpreter', 'latex');
xlabel('t [s]');
ylabel('Angle [deg]');
legend({'$\hat{b}$', '$\delta$ with $\hat{b}_{feedforward}$', ...
    '$\delta$ without $\hat{b}_{feedforward}$'},    ...
    'Interpreter', 'latex', 'Location', 'best')
grid on;


%% 5e: Feed forward estimated bias and wave filtered psi
load_system('ship5e.slx')
sim('ship5e.slx')


load_system('ship5e.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)
title(['Reference $\psi_{r} = 30, measured $\psi$ $\psi + \psi_{w} + v and estimated course \hat{\psi}']);
xlabel('t [s]');
ylabel('Angle [deg]');
legend({'\psi_{r}', 'measured compass course ($\psi$ + $\psi_{w} $+ $v$)' , 'estimated compass course ($\hat{\psi}$)'}, 'Location', 'best')
 grid on;


%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,'b', t3, psi3,'g')
title('Compass reference \psi_{r} = 30, and measured compass course', ...
    'Interpreter', 'latex');
xlabel('t [s]');
ylabel('Angle [deg]');
legend({'$\psi_{r}$', 'With Kalman filtered feedback', ...
     'Without Kalman filtered feedback'}, ...
     'Interpreter', 'latex', 'Location', 'best')
grid on;

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

subplot(2,1,2)
```

```
plot(time, rudder_input)
title(['Rudder input $\delta$ without Kalman filtered $\hat{\psi}$ '...
    'or $\hat{b}$'], 'Interpreter', 'latex');
xlabel('t [s]');
ylabel('Angle [deg]');
legend({'$\delta$'}, 'Interpreter', 'latex', 'Location', 'best')
grid on;


% Wave influence (current turned off, delta = 0)
load_system('ship5e_1.slx')
sim('ship5e_1.slx')

% Plotting wave influence on system
figure(figNum)
figNum = figNum+1;
plot(time, compass, time, compass_kalman)
title(['Measured wave influence \psi_{w} vs. estimated wave influence'...
    ' \hat{\psi}_{w}'],'Interpreter', 'latex');
xlabel('t [s]');
ylabel('Angle [deg]');
legend({'$\psi_{w}$', '$\hat{\psi}_{w}$'}, ...
    'Interpreter', 'latex', 'Location', 'best')
grid on;
```

8 Appendix B: Simulink Models

Figure 17: Simulink model from section 1.2/1.3

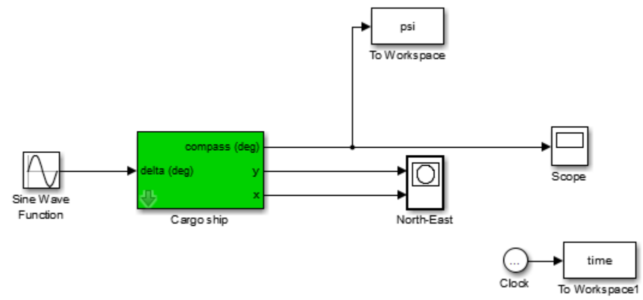


Figure 18: Simulink model from section 3.3/3.4

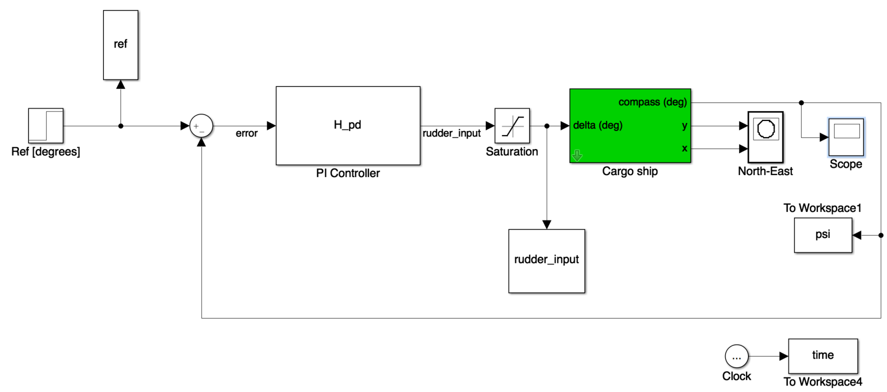


Figure 19: Simulink model from section 5.2

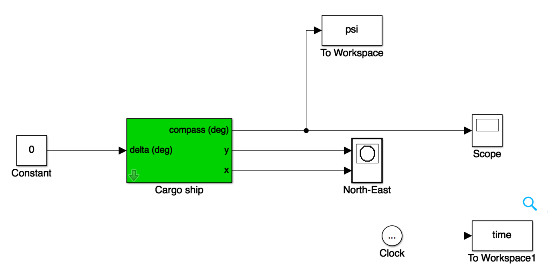


Figure 20: Simulink model from section 5.4

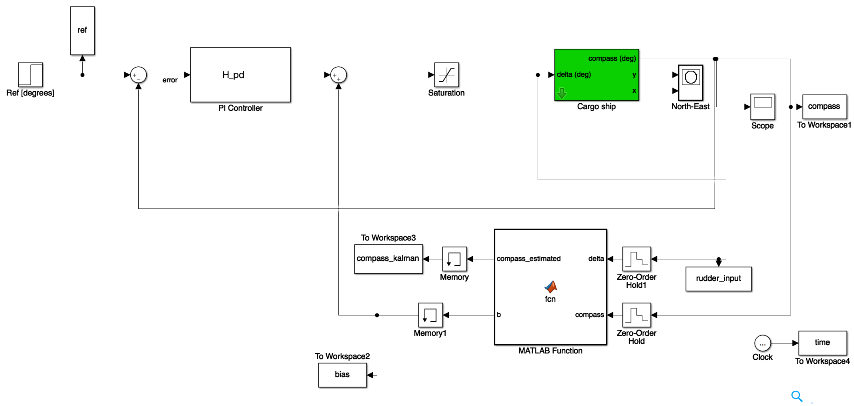


Figure 21: Simulink model from section 5.5