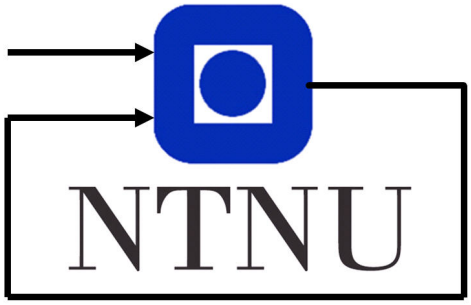
TTK4115 Boat labreport

Group 68
Sondre Bergum, Student-no: 756382
Martin Madsen, Student-no: 756411

November 12, 2018



Department of Engineering Cybernetics

Contents

1 Defining system model

For the complete explanation of NED and BODY frames and the ship dynamics used to derive the model we're about to present see the assignment text.

In the model the state vector is given by: $\mathbf{x} = \begin{bmatrix} \xi_w & \psi_w & \psi & r & b \end{bmatrix}^T$, where:

- ψ - Is the average heading, without wave disturbance
- ψ_w - Is a high-frequency component due to the wave disturbance.
- $\dot{\xi_w} = \psi_w$
- r - Rotational velocity of the boat in BODY frame.
- b - Bias to the rudder angle, caused by current disturbance.

The complete model of the system we will be using in this assignment can be stated as:

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

where y is the measured heading (compass measurement). w_b, w_w and v are white noise processes. Also note that the model in the assignment text that this is derived from is simplified to a 1st order Nomoto-approximation in eq. (1c)-(1d), with Nomoto time and gain constants, T and K. The Nomoto model is common for modelling yaw in marine systems.

Clearly, the system can be written as:

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

with $\mathbf{x} = \begin{bmatrix} \xi_w & \psi_w & \psi & r & b \end{bmatrix}^T$, $u = \delta$ and $\mathbf{w} = \begin{bmatrix} w_w & w_b \end{bmatrix}^T$. The purpose of this model is to estimate the course angle without the wave disturbance. Thus, we model the ship as a system not affected by waves and include the disturbance only in the measurement. Further, the current only affects the rudder angle in the model. This is of course not the case for an actual ship, but it simplifies the Kalman filter design.

2 Identification of the boat parameters

The aim of this section is to use basic identification techniques of system parameters that are not explicity given by the system model.

2.1 Problem a

To calculate the transfer function from δ to ψ, assuming no disturbances($w_w = w_b = 0$), we use the method that you see below. The transfer function $H_{ship}(s)$, parametrized by T and K, can be stated as done in eq. (5).

From eq. (1c)-(1e) we get eq. (3):

$$\dot{\psi} = r \tag{3a}$$

$$\dot{r} = -\frac{1}{T}r + \frac{K}{T}(\delta - b) \tag{3b}$$

$$b = w_b \tag{3c}$$

Rearranging the equations in eq. (3), we can state it as eq. (4):

$$\ddot{\psi} = -\frac{1}{T}\dot{\psi} + \frac{K}{T}(\delta - 0) \tag{4}$$

We then Laplace transform it and reorganize the terms to give us eq. (5):

$$\frac{\psi(s)}{\delta(s)} = H_{ship}(s) = \underline{\frac{K}{s^2T + s}} \tag{5}$$

2.2 Problem b

We want to identify the boat paramaters T and K in calm weather, with no wave or current disturbance. Running the simulation with sine inputs with amplitude 1 and frequencies $\omega_1 = 0.005$ and $\omega_2 = 0.05$ gives a set of equations to derive T and K from. We used the peak to peak analyzer in the Simulink scopes for the amplitude of the output in both problem b) and c) below. The plot of these responses can be seen labeled as "Calm weather" in fig. 1 and fig. 2 respectively. The amplitude values for calm weather are stated in eq. (6).

$$H_1 = |H(j\omega_1)| = 29.36, \quad H_2 = |H(j\omega_2)| = 0.831 \tag{6}$$

this gives us the the set of equations in eq. (7)

$$H_1^2(\omega_1^4T^2 + \omega_1^2) - K^2 = 0 \tag{7a}$$

$$H_2^2(\omega_2^4T^2 + \omega_2^2) - K^2 = 0 \tag{7b}$$

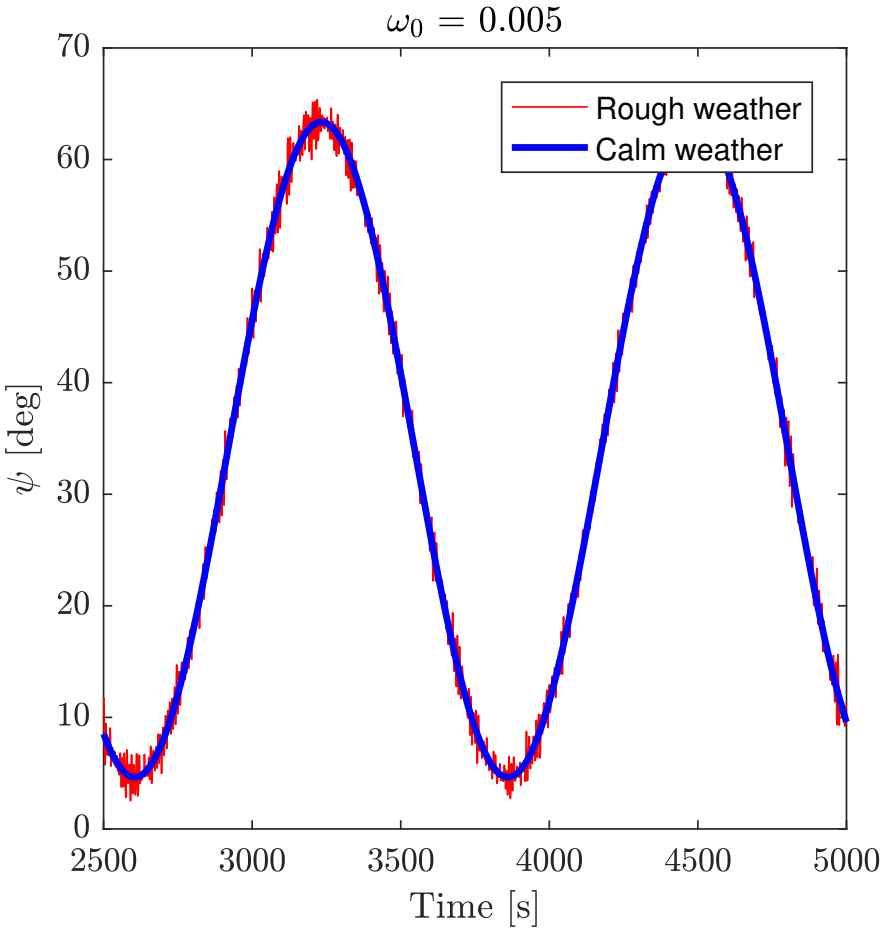


Figure 1: Compass measurement

Solving eq. (7) allows us to determine T and K and their values can be seen in eq. (8).

$$T = \sqrt{\frac{H_1^2\omega_1^2 - H_2^2\omega_2^2}{H_2^2\omega_2^4 - H_1^2\omega_1^4}} = \underline{72.44} \tag{8a}$$

$$K = \sqrt{H_1^2(\omega_1^4 T^2 + \omega_1^2)} = \underline{0.156} \tag{8b}$$

2.3 Problem c

We repeat the process from section 2.2 with waves and measurement noise on. The responses are again in fig. 1 and fig. 2 and we now examine the plots labeled "Rough weather". The extracted amplitudes are given by eq. (9).

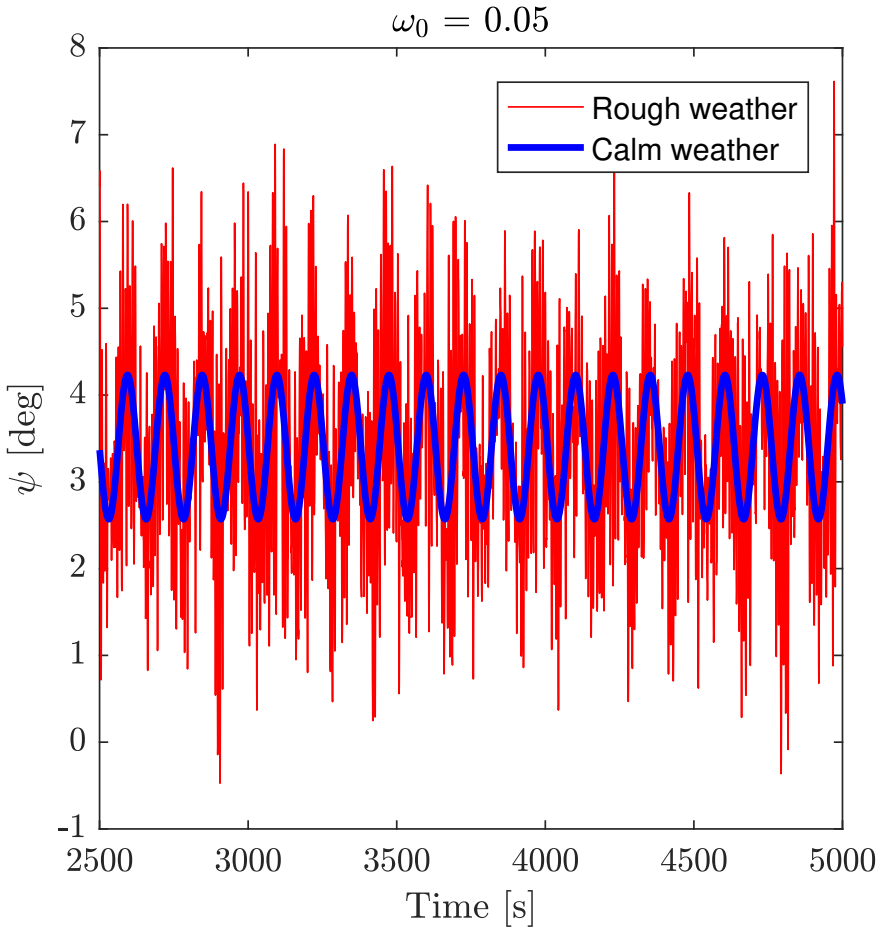


Figure 2: Compass measurement

$$H_1 = |H(j\omega_1)| = 31.145, \quad H_2 = |H(j\omega_2)| = 4.0435 \tag{9}$$

Inserting these amplitudes into eq. (8) gives us an estimate for boat parameters given by eq. (10) which are not a good estimate of the boat parameters.

$$T = i12.79, \quad K = 0.156 \tag{10}$$

We see that the results for rough weather in fig. 1 and fig. 2 are heavily influenced by the disturbances and noise, especially the signal with input frequency $\omega_2 = 0.05$ as seen in fig. 2. This makes our estimates poor because it's very hard to accurately determine an amplitude on the signal mentioned above. Even though we get the same K, T is estimated to be imaginary.

2.4 Problem d

We want to examine if our boat parameters are reasonable by comparing our theoretical model, eq. (5), and the simulated step response of the cargo-ship. We do this by implementing our transfer function in Simulink, and giving both it and the cargo-ship the same step input and plot them side-by-side. The simulink implementation for this is seen in fig. 30 in appendix B.

Applying a step input of 1° on the rudder gives us the step response shown in fig. 3. As can be seen, the values for T and K are pretty good as the step response of the model follows the response of the ship with an acceptably small error.

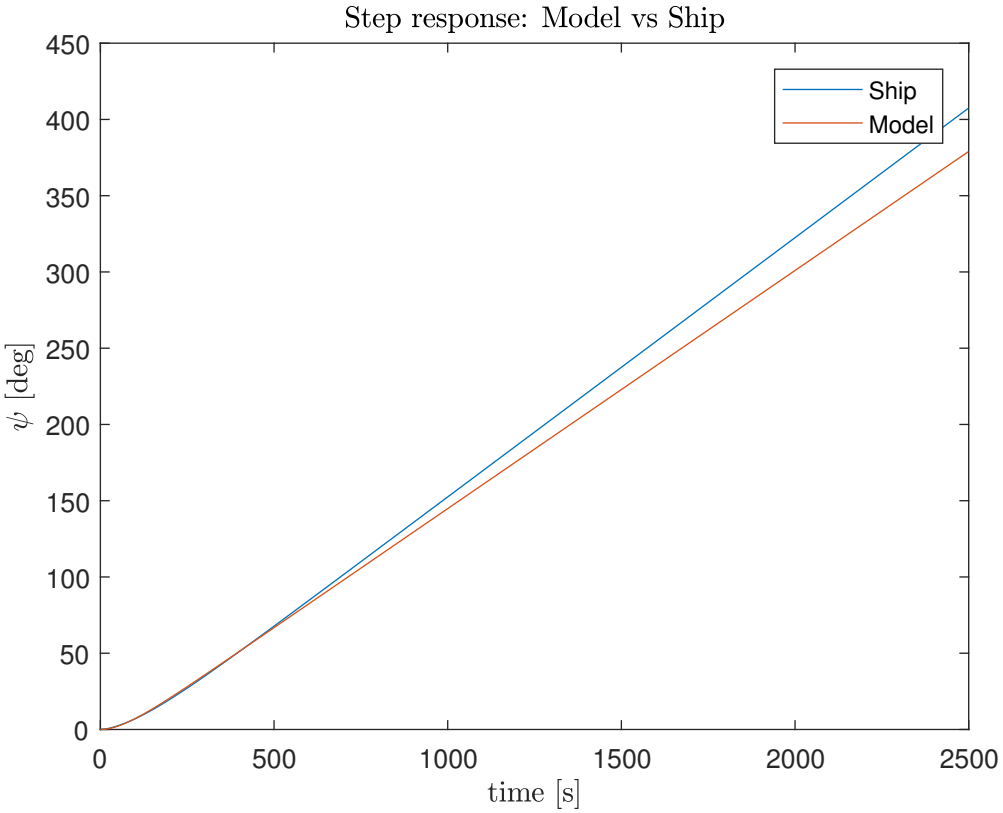


Figure 3: Step response of ship and theoretical model

We also note that our estimation of the boat parameters T and K could be improved. There is a growing discrepancy between our modeled response and the actual cargo-ship simulation. One main cause of this discrepancy is our method of finding the signal amplitude. We use a pretty naive method of just reading the peak-to-peak values of our output in a small interval. Increasing this interval by running the simulation longer or use a better

procedure for amplitude extraction will yield more accurate estimates for the boat parameters. However we deem the results good enough to move on with the Kalman filtering.

In all the following parts of this report measurement noise will be turned on in the ship model when running the simulations.

3 Identification of wave spectrum model

In this part we will analyze the model for the wave spectrum used as a disturbance in our simulations of the system. We will first extract the disturbance data and use it to empirically estimate a Power Spectral Density function for the model, then compare it to the theoretically obtained PSD-function. We will denote the empirically found PSD-function with $S(\omega)$ and the analytically derived one with $P(\omega)$.

3.1 Problem a

When estimating the Power Spectral Density function of ψ_w, $S_{\psi_w}(\omega)$, we load the "wave.mat" file containing the signals used for the wave disturbance. We used the MATLAB function `[Sxx, f] = pwelch(x, window, noverlap, nfft, fs)` and converted the result to the correct units, as shown in the MATLAB code in fig. 18 in appendix A.

This gave us two vectors of data, Sxx and f, and plotting them against one another (plotting $S_{xx}(f)$) with `plot(f, Sxx)` gives us what we are looking for. The result can be seen in fig. 4 marked as the empirically found function. We can then analyze the plot and extract the information we need to look at the corresponding theoretical representation.

3.2 Problem b

We now want to derive the theoretical PSD-function of the wave spectrum.

We state eq. (1a) in the ship model as eq. (11),

$$\frac{d}{dt}\xi_w = \psi_w \quad \Longrightarrow \quad \xi_w = \int \psi_w dt \tag{11}$$

and combining it with eq. (1b) we get an equation for $\dot{\psi_w}$ given by eq. (12)

$$\dot{\psi_w} = -\omega_0 \int \psi_w dt - 2\lambda\omega_0\psi_w + K_w w_w \tag{12}$$

Laplace transforming and reorganizing eq. (12) gives us the transfer function from w_w to ψ_w in eq. (13)

$$\frac{\psi_w(s)}{w_w(s)} = H_w(s) = \frac{sK_w}{s^2 + s2\lambda\omega_0 + \omega_0^2} \tag{13}$$

Then from the theoretical relation in eq. (14), which in our case is stated in eq. (15), we arrive at eq. (16) because w_w is white noise so $P_{w_w}(\omega) = 1$.

$$S_y(\omega) = |H(j\omega)|^2 S_u(\omega), \quad \text{where} \quad H(j\omega) = \frac{y(j\omega)}{u(j\omega)} \tag{14}$$

$$P_{\psi_w}(\omega) = |H_w(j\omega)|^2 P_{w_w}(\omega) \tag{15}$$

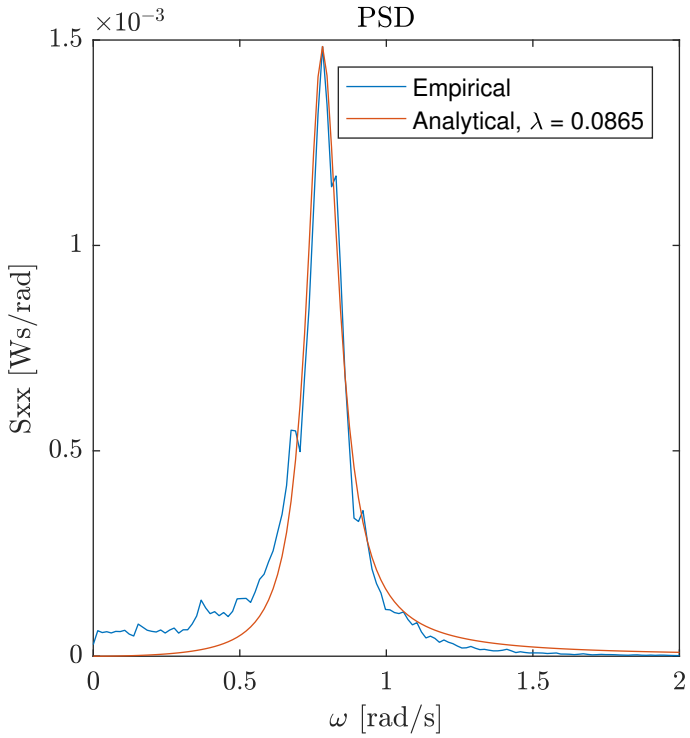


Figure 4: PSD function

$$P_{\psi_w}(\omega) = |H_w(j\omega)|^2 = \underline{\frac{\omega^2 K_w^2}{(\omega^2 - \omega_0^2)^2 + (2\lambda\omega\omega_0)^2}} \tag{16}$$

3.3 Problem c

Analyzing the power spectral density function obtained through MATLAB we get the values in eq. (17).

$$\omega_0 = 0.4915, \quad \sigma^2 = S_{\psi_w}(\omega)_{peak} = 0.0015 \tag{17}$$

This means that the wave spectrum's influence is greatest when the waves have a frequency of ω_0.

3.4 Problem d

Now that we have two representations of the PSD-function for the wave spectrum, we want to match them together as best we can.

Given that $K_w = 2\lambda\omega_0\sigma$ we put it into eq. (16) and get eq. (18)

$$P_{\psi_w}(\omega, \lambda) = \frac{(2\lambda\omega\omega_0\sigma)^2}{(\omega^2 - \omega_0^2)^2 + (2\lambda\omega\omega_0)^2} \tag{18}$$

We then use the MATLAB function `lsqcurvefit(Pxx, 0.1, f, Sxx)` to estimate λ by minimizing the error between the emprically estimated $S_{\psi_w}(\omega)$ and the analytically derived $P_{\psi_w}(\omega, \lambda)$.

The result of the estimation was $\lambda = 0.0865$ and the comparison can be seen in fig. 4.

3.5 Evaluation of results

In this section we are extracting and analyzing all the data available to us regarding the wave spectrum, and we are using a curve-fitting method which surpasses anything we can do ourselves. There is not much we can find to be improved upon in the methodology here.

4 Control system design

In the following problems we will design an autopilot and simulate the system in different conditions.

4.1 Problem a

In the autopilot we will be using a PD-controller which can be stated as eq. (19). When combining this controller with the transfer function for our model we arrive at at the open loop transfer function shown in eq. (21).

$$H_{pd}(s) = K_{pd}\frac{sT_d + 1}{sT_f + 1} \tag{19}$$

Given the previously found $H_{ship}(s)$ in eq. (5) we get:

$$H(s) = H_{pd}(s) \cdot H_{ship}(s) = KK_{pd}\frac{sT_d + 1}{s(sT + 1)(sT_f + 1)} \tag{20}$$

Setting $T_d = T$ to cancel out the ship's time constant we get:

$$H(s) = H_{pd}(s) \cdot H_{ship}(s) = KK_{pd}\frac{1}{s(sT_f + 1)} \tag{21}$$

In the controller we want a phase margin (φ) of approximately 50°, and $\omega_c = 0.1$ rad/s. The phase margin is found at the frequency where the transfer function has a gain of 1 (0 dB), namely ω_c. [3]

$$\varphi = 50^\circ = \angle H(j\omega_c) - (-180^\circ) \tag{22a}$$
$$50^\circ - 180^\circ = \angle H(j\omega_c) \tag{22b}$$
$$-130^\circ = \angle \frac{KK_{pd}}{j\omega_c(j\omega_c T_f + 1)} \tag{22c}$$
$$-130^\circ = 0 - \angle - \omega_c^2 T_f + j\omega_c \tag{22d}$$
$$130^\circ = \arctan(-\frac{\omega_c}{\omega_c^2 T_f}) \tag{22e}$$
$$T_f = -\frac{1}{\omega_c \tan(130^\circ)} = \underline{8.39} \tag{22f}$$

We then find K_{pd} by:

$$|H(j\omega_c)| = 1 \tag{23a}$$

$$\frac{|KK_{pd}|}{|j\omega_c(j\omega_c T_f + 1)|} = 1 \tag{23b}$$

$$\frac{\sqrt{(KK_{pd})^2}}{\omega_c\sqrt{(\omega_c T_f)^2 + 1}} = 1 \tag{23c}$$

$$K_{pd} = \frac{\omega_c\sqrt{\omega_c^2 T_f^2 + 1}}{K} = \underline{0.837} \tag{23d}$$

We can now insert these values for T_f and K_{pd} into the open-loop transfer function, eq. (21), and examine it's bode plot as is done in fig. 5. These values for T_f and K_{pd} gives us a phase margin of $\varphi = 49.2°$ at the cross-frequency $\omega_c = 0.103$ rad/s. The phase-margin and cross-frequency are both close to our goal.

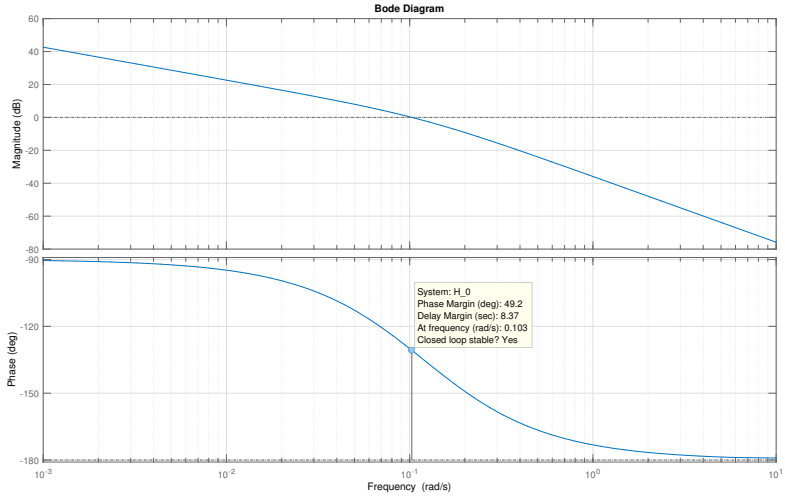


Figure 5: Bode plot for open loop system transfer function

4.2 Problem b

Simulating the autopilot designed in the previous problem with only measurement noise turned on gives a desired result as seen in fig. 6. The ship reaches it's set-point for the heading ($\psi_r = 30°$), which is the most important aspect of it. We have not been given any indication of the size of the ship other than it is a *cargo*-ship. At first we thought that 200-250 seconds, about 4 minutes, was a long time to turn a naval vessel, but that might not be the case for the largest of ships.

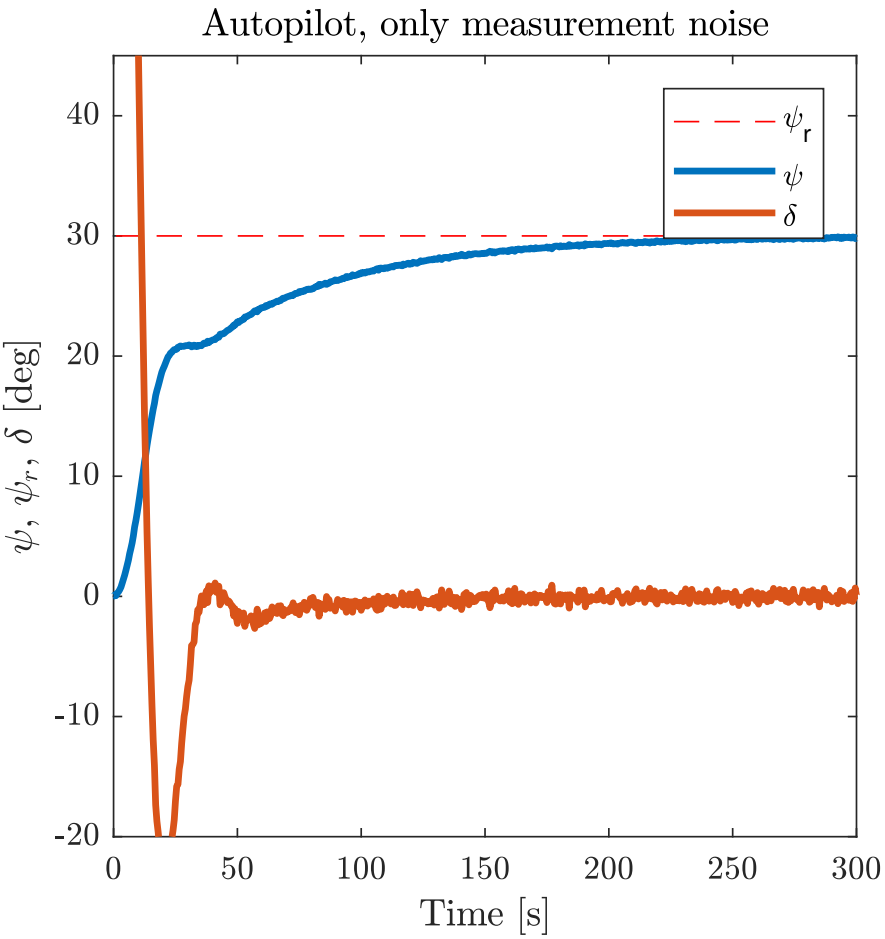


Figure 6: No disturbances, only measurement noise

4.3 Problem c

When we simulate with the current disturbance turned on the ship does not reach it's desired heading as seen in fig. 7. The stationary error is of about 5° which is almost 17 % of the desired change in heading. Having an error this big with a current disturbance (say the Gulf stream over the Atlantic) would make a ship hit the wrong continent on arrival.

4.4 Problem d

In the simulation of the autopilot with wave disturbance, and no current disturbance, we reach and hold the desired heading, but at a fatal cost. As can be seen in fig. 8, the rudder input, δ, is fluctuating very rapidly as the controller is trying to correct the changes in course made by the waves. Making such sudden changes back and forth over a small time-interval when

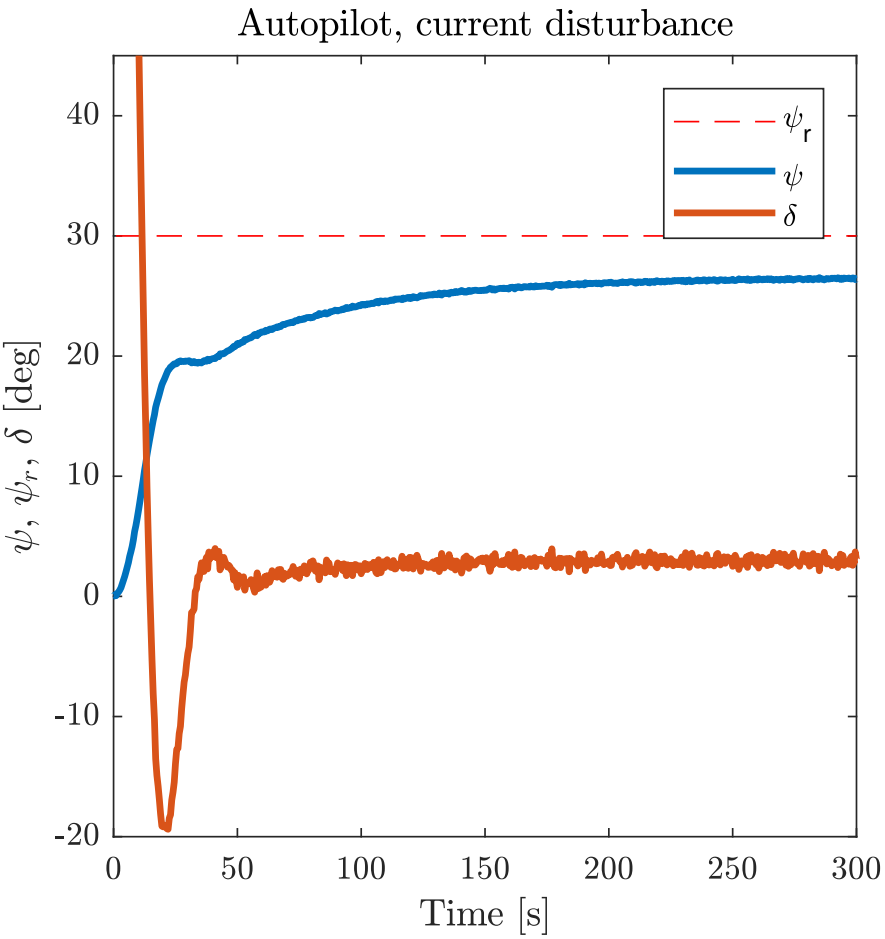


Figure 7: Current disturbance and measurement noise

the rudder probably is the size of a bus and has to move an enormous amount of water, will wear out the actuator for the rudder in no time. Changing the actuator every time a voyage is subjected to rough weather is going to be very cost-inefficient and is not an ideal situation.

4.5 Result evaluation

Initially, when designing the controller we misinterpreted ω_c as the *cutoff*-frequency, instead of the *cross*-frequency (which we thought was denoted ω_0 - where the amplitude crosses the 0 dB line in a bode plot, and the amplitude of the transfer function equals 1). This made our design of the controller somewhat different at first, with the values $T_f = 10$ and $K_{pd} = 0.641$. This gave us a phase margin of $\varphi = 51.8°$ at $\omega_{cross} = 0.0786$ rad/s. With these values we had $\omega_{cutoff} = 0.1$ such that $|H(j\omega_{cutoff})| = \frac{1}{\sqrt{2}} = -3$

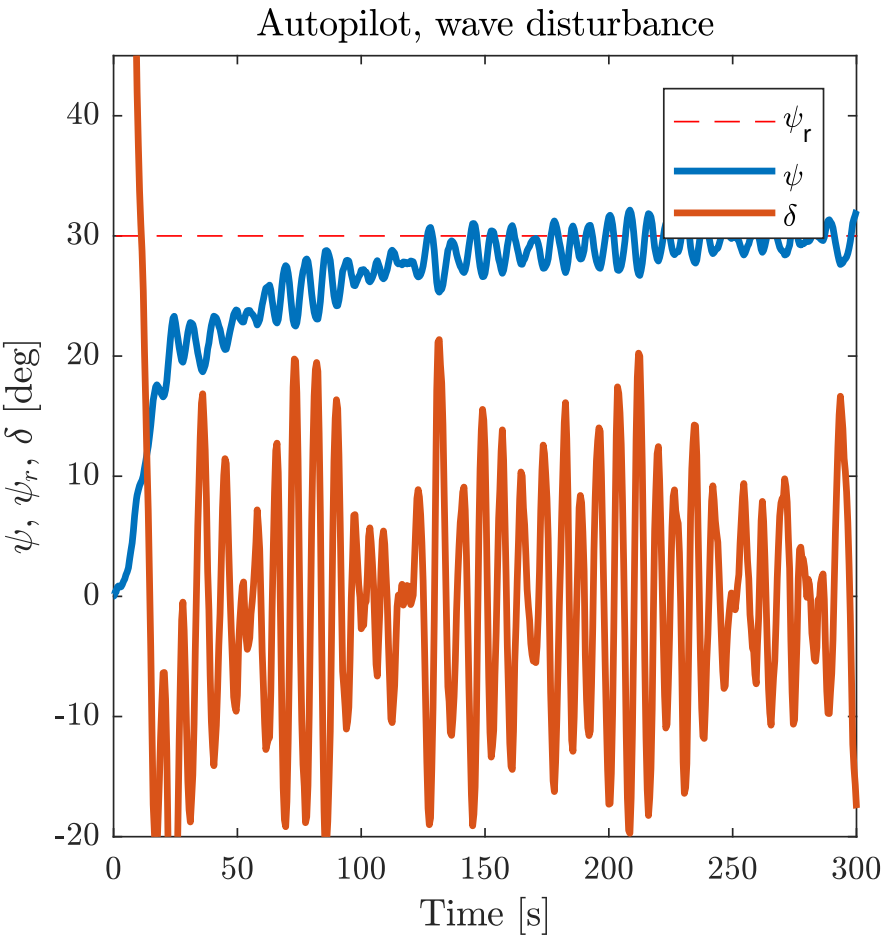


Figure 8: Wave disturbance and measurement noise

dB. This was the controller we used in the following assignments until we noticed the possible mistake we had made. We then ran all the simulations again and what we discovered was that our "mistake" actually gave us a seemingly better result than the controller specified in the assignment. In our simulations of problem b in this part of the assignment the response of our initial controller is shown in fig. 9, and the response of the controller specified by the assignment is shown in fig. 6. Our initial controller reaches the set value marginally faster, has a smoother change in heading and the input, δ has smaller magnitude with less oscillations.

Addressing the stationary error caused by a current disturbance can be done by adding an I-term to the PD-controller as it will change input not only in relation to the error but also how long the error has persisted.

The high frequency rudder change while the system is subjected to a wave disturbance can be limited by applying a low-pass filter on the measured

heading as the controller will then not register the sudden changes in course, but rather have a more gradual correction, if the waves affect the heading too much.

This leads us to conclude that a better autopilot can be found by spending more resources on the tuning of the PD-controller and determining boat-parameters. Despite our findings we chose to use the controller specified in the assignment text in our final results as Kalman filtering is the main purpose of this assignment, and it will address the issues caused by both sources of disturbance.

Another note on the obtained results is that when our set-point is $\psi_r = 30°$, the controller gives an initial input to the ship at approximately $\delta = 140°$. Due to the saturation in the actual ship-model of $|\delta|_{max} = 45°$, we chose not to saturate the output from the controller so we could see the unsaturated response of our controller.

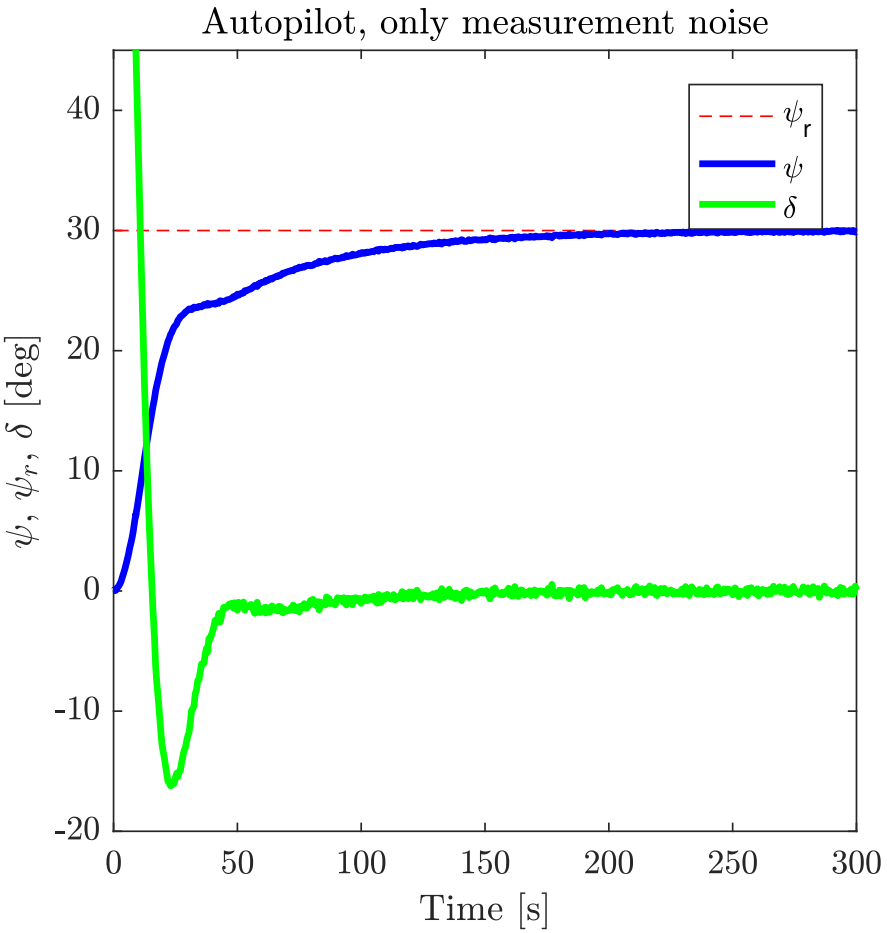


Figure 9: Response of initial controller design without disturbances

5 Observability

The way the disturbances are included in the state space model, as high-frequency states in case of wave disturbances or superpositioned on the original state in case of rudder bias, makes it possible to determine the observability of our system using conventional methods. Our code for these observability tests are found in the appendix in fig. 23.

5.1 Problem a

The system as presented in eq. (1) can be represented in the state-space form of equation eq. (2). The system matrices for the full system with all disturbances are given by eq. (24). Note that removing disturbances corresponds to changing the state vector, $\mathbf{x}$, and the system matrices.

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

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{24}$$

Below we will present the state space model for each scenario, and then consider the observability of each separate case. The observability will be determined by checking if the observability matrix has full rank[2]. Since the observability matrix is only dependent on the system matrices $\mathbf{A}$ and $\mathbf{C}$ the rest of the system matrices have been omitted from each scenario's observability test. The MATLAB-command to do this is given by `obsv(A,C)`.

5.2 Problem b

With no disturbances at all the only states in our system are ψ and r and the system matrices $\mathbf{A}$ and $\mathbf{C}$ are as presented in eq. (25). The system is observable, as shown in eq. (26).

$$\mathbf{x} = \begin{bmatrix} \psi \\ r \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{25}$$

$$rank(\mathbb{O}_{nodisturbance}) = 2 \Rightarrow Observable \tag{26}$$

5.3 Problem c

Introducing a current disturbance to the system is equivalent to augmenting the disturbance-free state-space with the rudder bias, b. The new system

matrices and state vector are found in eq. (27). This system is also observable as shown in eq. (28).

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

$$rank(\mathbb{O}_{current}) = 3 \Rightarrow Observable \tag{28}$$

5.4 Problem d

Introducing a wave disturbance to the disturbance-free system, eq. (25), can be modeled as augmenting the state space with ψ_w and ξ_w. The two augmented states corresponds to the high-frequency component of the heading, caused by the wave disturbance. The new state-space is shown in eq. (29) and by eq. (28) it too is observable.

$$\mathbf{x} = \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \tag{29}$$

$$rank(\mathbb{O}_{wave}) = 4 \Rightarrow Observable \tag{30}$$

5.5 Problem e

Incorporating both wave and current disturbance into our state-space gives the state vector and system matrices shown in eq. (31). By eq. (32) the system, with all disturbances included, is observable.

$$\mathbf{x} = \begin{bmatrix} \xi_w \\ \psi_w \\ \psi \\ r \\ b \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{31}$$

$$rank(\mathbb{O}_{waveandcurrent}) = 5 \Rightarrow Observable \tag{32}$$

5.6 Final note on observability and impact of stochastic signals

The observant reader may note that the nature of our stochastic, random, signals hasn't entered the discussion before now.

When we examined the observability of our system, stochastic disturbances included, we assumed that we could just use the notion of observability for deterministic systems developed in [2]. The reason this holds up in a stochastic environment depends on the way we define observability in stochastic systems. The criteria for observability in a deterministic system is that knowing all prior inputs, $\mathbf{u}$, and measurements, $\mathbf{y}$ will let you uniquely determine the initial state, $\mathbf{x}_0$. This gives rise to the observability test that we use. If the observability matrix has full rank, it follows that its nullity is 0. Note that the nullity of the observability matrix is only dependent upon the system matrices $\mathbf{A}$ and $\mathbf{C}$.

The way we expand this to a stochastic system, such as the one in eq. (2), is to propose that the probability density function of the initial state is shrinking towards a singular point as time approaches infinity. This criteria can be tested the same way as for the deterministic system, by checking the nullity and hence the rank of the observability matrix, leading us to the conclusion that all systems above, with or without disturbances are observable.

6 Discrete Kalman filter

This section explains our method for constructing a discrete Kalman filter for our cargo-ship to estimate a rudder-bias and filter out high-frequency wave-influences, and compares the system response with the Kalman filter to that of the system used in section 4.

6.1 Problem a

To implement a discrete Kalman filter we naturally need a discrete state-space representation of our cargo-ship dynamics. The code needed to perform this discretization with a sample frequency of 10 Hz $\rightarrow$ ($T_s = 0.1$ s) is found in fig. 24 and in parts of fig. 32. We use the built-in functionality in MATLAB to get our discrete system-matrices.

$$\mathbf{A_d} = \begin{bmatrix} 0.9970 & 0.0992 & 0 & 0 & 0 \\ -0.0607 & 0.9835 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & -1.0763 \cdot 10^{-5} \\ 0 & 0 & 0 & 0.9986 & -2.1520 \cdot 10^{-4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B_d} = \begin{bmatrix} 0 \\ 0 \\ 1.073 \cdot 10^{-5} \\ 2.1520 \cdot 10^{-4} \\ 0 \end{bmatrix}$$
$$\mathbf{C_d} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \tag{33}$$

Note that the system matrix $\mathbf{E}$ isn't discretized in our code, neither is it listed here as one of the discrete system matrices since it's only purpose is to discretize $\mathbf{Q}$, and we used van Loan's method to do this, which makes it unnecessary to find $\mathbf{E_d}$. [1]

6.2 Problem b

To find the variance of the measurement noise we ran a simulation with 0° as rudder input so our output would be purely the measurement noise, v. We then extracted the data as a time series and used the Matlab function `var(v)`.

The result gave us the $\mathbf{R}$ for our system, $\mathbf{R} = 0.0020$.

6.3 Problem c

In order to construct a Kalman filter for our system we need a complete discretized model, and so we were provided with the matrices in eq. (34).

$$\mathbf{w} = \begin{bmatrix} w_w & w_b \end{bmatrix}, \quad E\{\mathbf{w}\mathbf{w}^T\} = \mathbf{Q} = \begin{bmatrix} 30 & 0 \\ 0 & 10^{-6} \end{bmatrix},$$
$$\mathbf{P}_0^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.013 & 0 & 0 & 0 \\ 0 & 0 & \pi^2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2.5 \cdot 10^{-4} \end{bmatrix}, \hat{\mathbf{x}}_0^- = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{34}$$

From these given matrices, our discretized system-matrices and the measurement noise variance, $\mathbf{R}$ we have everything needed to construct our discrete Kalman filter. We start by discretizing the disturbance variance matrix $\mathbf{Q}$ using van Loan's method[1] and the measurement noise variance matrix $\mathbf{R}$, the result is seen in eq. (35), where $\mathbf{Q_d}$ denotes the discretized disturbance variance matrix and $\mathbf{R_d}$ denotes the discretized noise variance.

$$\mathbf{Q_d} = \begin{bmatrix} 2.6889 \cdot 10^{-7} & 4.0165 \cdot 10^{-6} & 0 & 0 & 0 \\ 4.0165 \cdot 10^{-6} & 8.0331 \cdot 10^{-5} & 0 & 0 & 0 \\ 0 & 0 & 2.3170 \cdot 10^{-18} & 5.7917 \cdot 10^{-17} & -3.5879 \cdot 10^{-13} \\ 0 & 0 & 5.7917 \cdot 10^{-17} & 1.5443 \cdot 10^{-15} & -1.0763 \cdot 10^{-11} \\ 0 & 0 & -3.5879 \cdot 10^{-13} & -1.0763 \cdot 10^{-11} & 1.000 \cdot 10^{-7} \end{bmatrix}$$
$$\mathbf{R_d} = \frac{\mathbf{R}}{T_s} = 0.0199 \tag{35}$$

To implement our discrete controller in simulink we use code from fig. 25 to construct a data-struct which is in turn sent over to the Kalman filter "fcn-block" in Simulink. Simulink model fig. 32 shows the implementation of the Kalman filter.

The algorithm to perform the discrete Kalman filtering is explained in [1], and our implementation of this algorithm is found in fig. 28. As proven in lecture and in [1], the Kalman filter will make the optimal choice between measurement and estimate at each iteration of the algorithm, and perform a blending of the two signals to compute its desired output. In essence it reduces the variance of its output compared to its input. In our case we feed it our measurement of ψ, apply the Kalman filter, and get a wave-filtered ψ with a lower variance.

Figure 10 shows the outputs of the Kalman filter, $\psi_{filtered}$ and b. These values are obtained by running a simulation without disturbances (only measurement noise) with the model mentioned above, so the Kalman filter takes in the input u and the measurement y, but the output of the filter doesn't enter the closed loop system.

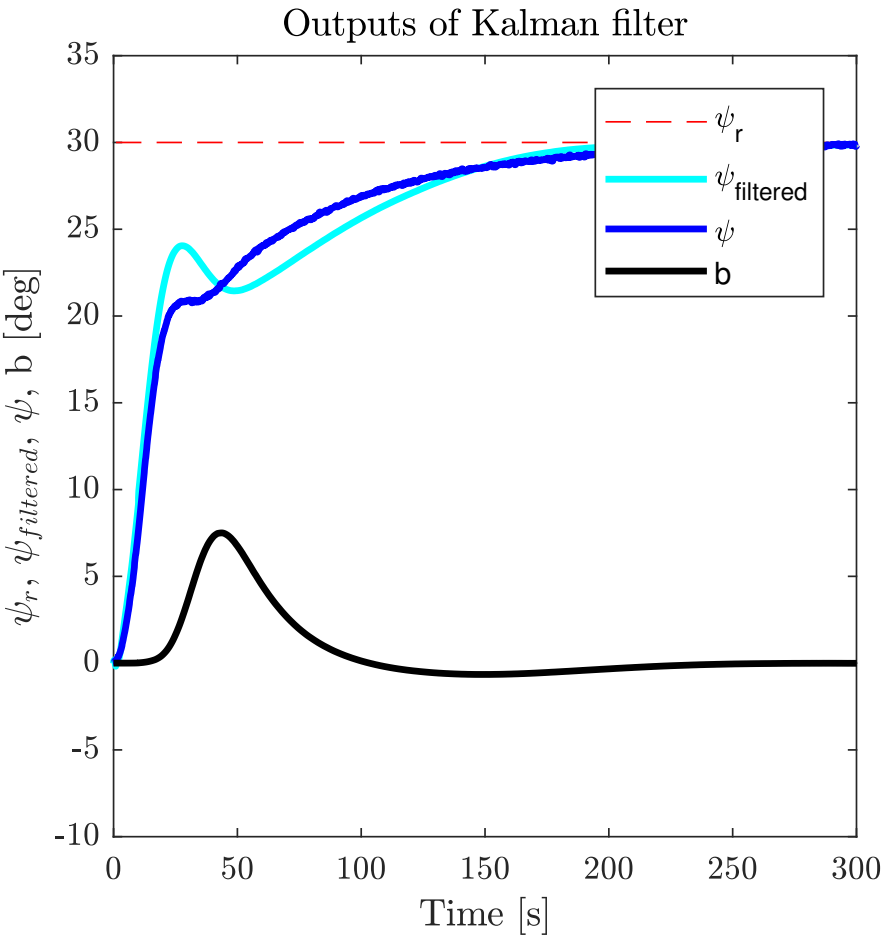


Figure 10: Estimated values form the Kalman filter

6.4 Problem d

The first step to adding the output of the Kalman filter to our closed loop system was to add estimated rudder bias, b, as a feed forward to the closed loop as shown in fig. 33 in appendix B. With this model we ran a simulation with current disturbance and no wave disturbance. The resulting response is shown in fig. 11. If compared to the response from section 4.3, shown in fig. 7 we see that the stationary error is no longer present in the heading, meaning the Kalman filter would have made the addition of an I-term in the controller obsolete.

6.5 Problem e

By changing the feedback loop from the measured heading to the wave-filtered heading from the Kalman filter we aim to further improve our system.

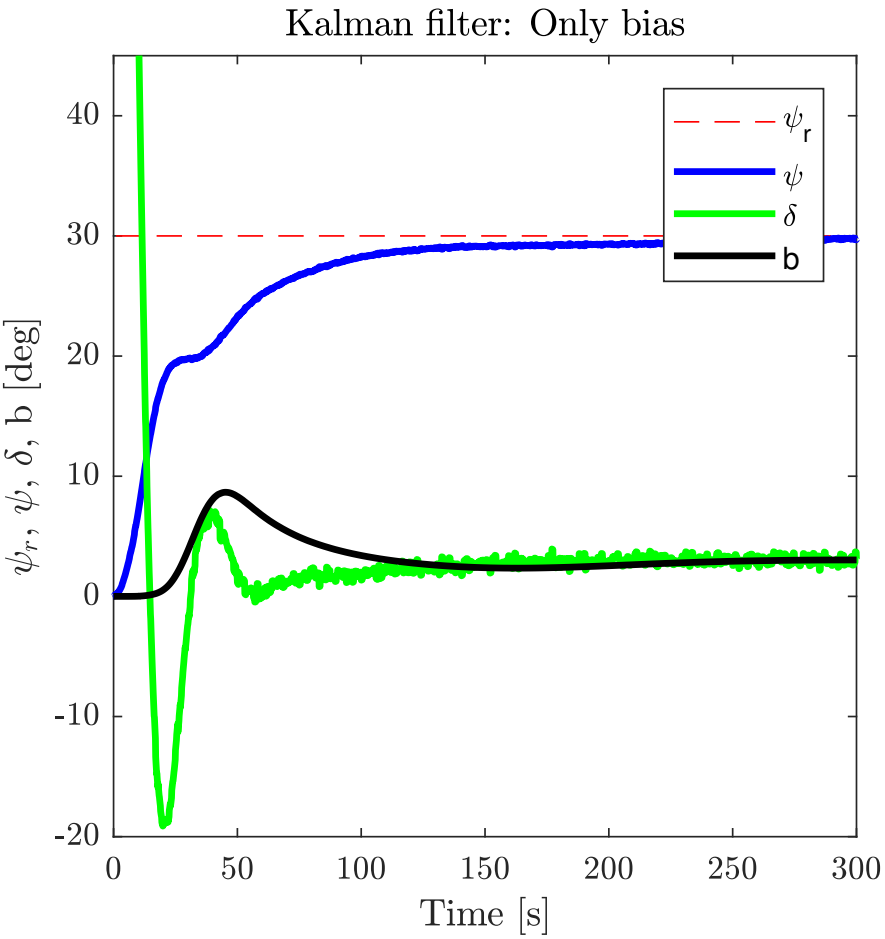


Figure 11: System response with feed forward from estimated bias

We simulate the system with both current and wave disturbance turned on to get the result shown in fig. 12. We compare this response to the one in fig. 8 from section 4.4 and see there's been a dramatic reduction in the magnitude of the high frequency rudder response, δ. The system is no longer trying as hard to correct deviations from the course set-point caused by the waves because the Kalman filter is smoothing out the signal fed back as the process value. Limiting this high-frequency rudder-input makes it much easier on the actuator for the rudder, reducing the wear and tear on it. The Kalman filter has now addressed the most prominent issues with both the sources of disturbance applied to the system without having to implement any of the changes proposed in section 4.5

To smooth out the measured heading the Kalman filter has to estimate the influence the wave disturbance has on it and counteract it. In fig. 13 we compare the actual wave influence on the system to what the Kalman

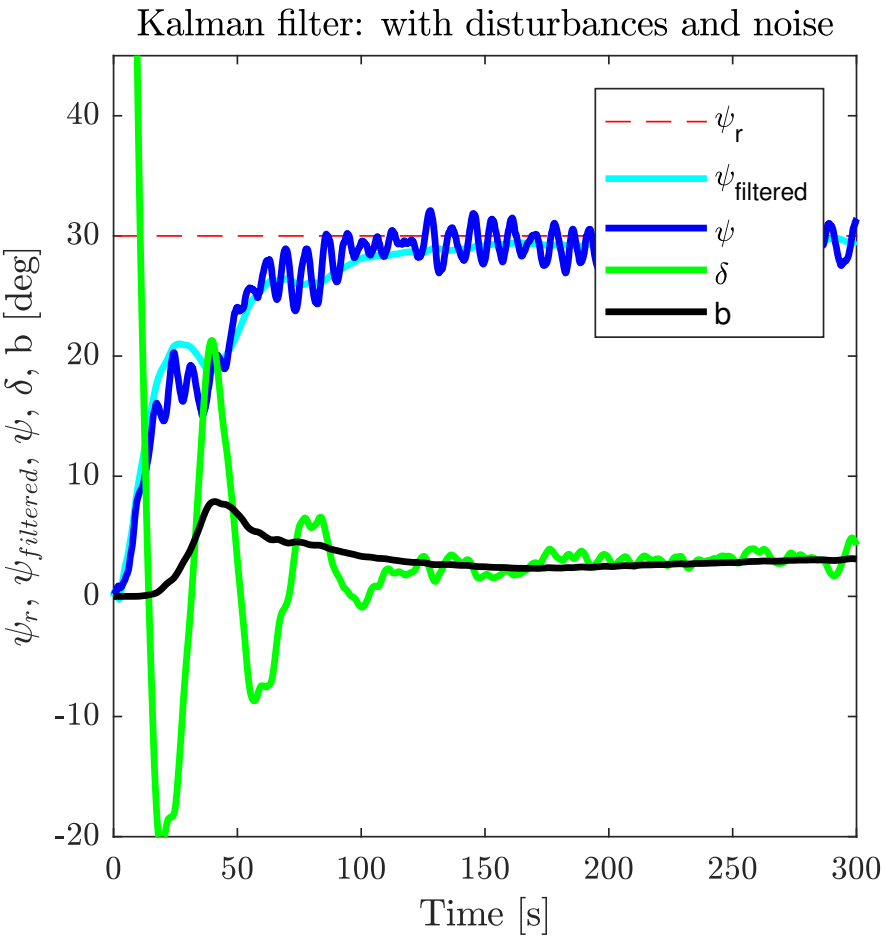


Figure 12: System response and filter values with feed forward and filtered feedback loop

filter estimates it to be. We see that it's a near match, but the filter falls short in the peaks of the wave signal. The estimation of ψ_w is good, but not perfect so there is room for improvement on our Kalman filter. Right now we update our Kalman filter with the same frequency as the wave-signal it is trying to estimate. By increasing the frequency we run our Kalman algorithm with we might get better results as this will cause us to get more estimates per period of the wave signal. However, the trade-off here is that running the Kalman filter seems to be quite computationally heavy already, meaning that any actual implementation on a real cargo-ship would require some serious computing power. Even though we do not have a perfect match on our ψ_w estimate we note that our autopilot performs quite well.

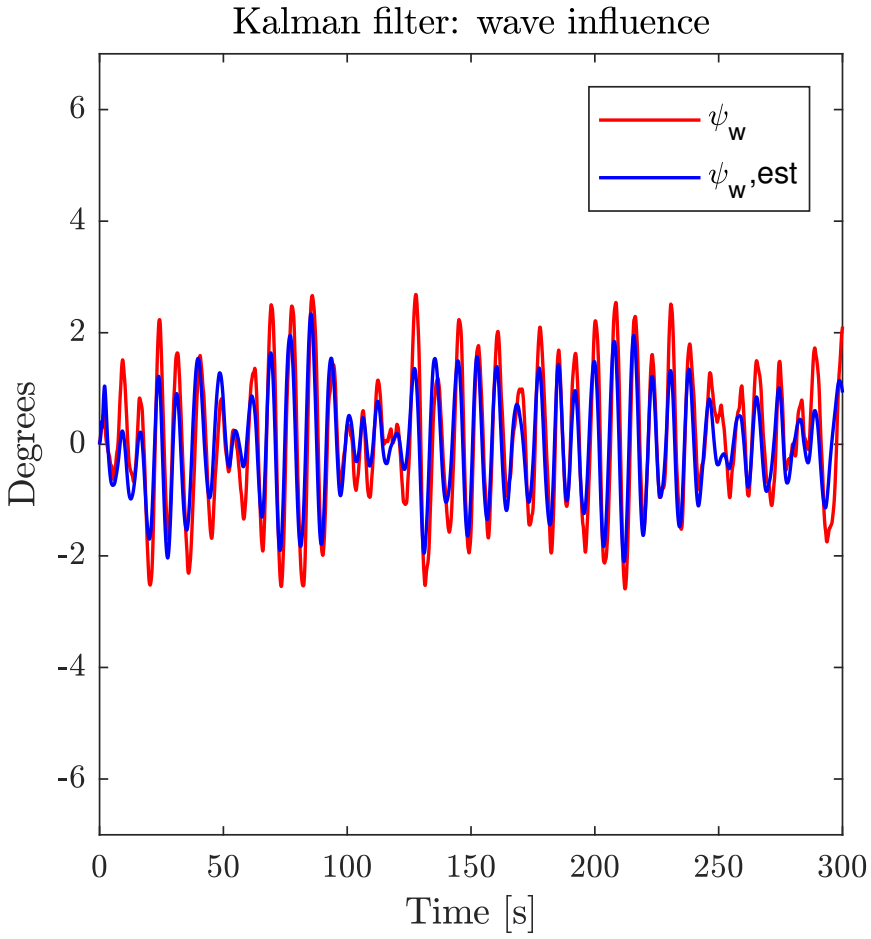


Figure 13: Wave influence and estimated wave influence

A MATLAB code

Figure 14: Initialization used for code in the identification of boat parameters. Name of file: "p5p1_init.m"

```
%Variable initialization
load('wave.mat');
w_1 = 0.005;
w_2 = 0.050;

%Found during our first simulation
abs_H_1 = 29.36;
abs_H_2 = 0.831;

T = sqrt((abs_H_1^2 * w_1^2 - abs_H_2^2 * w_2^2) ...
        /(abs_H_2^2 * w_2^4 - abs_H_1^2 * w_1^4));
K = sqrt(abs_H_1^2 * (w_1^4 * T^2 + w_1^2));
```

Figure 15: Code used to identify boat parameters T and K in calm weather.

```
1
2  w_1 = 0.005;
3  w_2 = 0.05;
4
5  omega = w_1;
6  sim('p5p1b_model.mdl');
7  compass_w1 = BODYheading;
8
9  omega = w_2;
10 sim('p5p1b_model.mdl');
11 compass_w2 = BODYheading;
12
13 plot(compass_w1.Time,compass_w1.Data,compass_w2.Time,compass_w2.Data);
14
15 amplitude_1 = peak2peak(compass_w1.Data(500:4500));
16 amplitude_2 = peak2peak(compass_w2.Data(1000:4500));
17
18 abs_H_1 = amplitude_1/2;
19 abs_H_2 = amplitude_2/2;
20
21 T = sqrt((abs_H_1^2 * w_1^2 - abs_H_2^2 * w_2^2) ...
22     /(abs_H_2^2 * w_2^4 - abs_H_1^2 * w_1^4))
23 K = sqrt(abs_H_1^2 * (w_1^4 * T^2 + w_1^2))
```

Figure 16: Code used to identify boat parameters in rough weather.

```

w_1 = 0.005;
w_2 = 0.05;

omega = w_1;
sim('p5p1c_model.mdl');
compass_w1 = BODYheading;

omega = w_2;
sim('p5p1c_model.mdl');
compass_w2 = BODYheading;

plot(compass_w1.Time,compass_w1.Data,compass_w2.Time,compass_w2.Data);

amplitude_1 = peak2peak(compass_w1.Data(500:4500));
amplitude_2 = peak2peak(compass_w2.Data(1000:4500));

abs_H_1 = amplitude_1/2;
abs_H_2 = amplitude_2/2;

T = sqrt((abs_H_1^2 * w_1^2 - abs_H_2^2 * w_2^2) ...
    /(abs_H_2^2 * w_2^4 - abs_H_1^2 * w_1^4))
K = sqrt(abs_H_1^2 * (w_1^4 * T^2 + w_1^2))
```

Figure 17: Code used to compare our theoretical model to the simulation.

```
run('../Common files/p5p1_init.m');

Gain_T = T;
Gain_K = K;
sim('p5p1d_model.mdl');


set(0, 'DefaultTextInterpreter', 'latex')

plot(BODYheading.Time, BODYheading.Data, ...
    ModelResponse.Time, ModelResponse.Data);
title('Step response: Model vs Ship');
xlabel('time [s]');
ylabel('$\psi$ [deg]');
legend('Ship','Model');
```

Figure 18: Code used in identification of wave spectrum model.

```
load('../Common files/wave.mat');
[Sxx, f] = pwelch(psi_w(2,:).*(pi/180), 4096, [], [], 10); %[Ws]
Sxx = Sxx*(1/(2*pi)); %convert to [W s/rad]
f = f*2*pi; %comes as [Hz] = [1/s] -> convert to [rad/s]

[sigma_sq,n] = max(Sxx); %finc max
sigma = sqrt(sigma_sq)
omega_0 = f(n) %resonance frequenzy

%Analytical PSD
Pxx = @(lambda, omega) ((2*lambda*omega*omega_0*sigma).^2) ...
    ./ ((omega_0^2 - omega.^2).^2 + (2*lambda*omega*omega_0).^2);

lambda = lsqcurvefit(Pxx, 0.1, f, Sxx) %Damping factor

%% Define figure size
width = 10; % cm
height = 10; % cm
fontsize = 10; % points
x = 20; y = 20;
set(0,'DefaultTextInterpreter', 'latex')

%% Set up the figure
fig1 = figure(1);
fig1.Units = 'centimeters';
fig1.Position = [x y width height];

%% Plot the data
plot(f, Sxx); hold on;
plot(f, Pxx(lambda, f)); hold off;
xlim([0 2]);

%% Set up the properties of the axes
ax = gca;
ax.FontUnits = 'points';
ax.FontSize = fontsize;
ax.TickLabelInterpreter = 'latex';
xlabel('$\omega$ [rad/s]')
ylabel('Sxx [Ws/rad]')
legend('Empirical', 'Analytical, \lambda = 0.0865')
title('PSD')

ax.TitleFontSizeMultiplier = 1.1;
%include the following line to export the plot
%hgexport(fig1,'p5p2.eps')
```

Figure 19: Initialization used for code in the PD-controller, optional bode plot commented out. Name of file: "p5p3_init.m"

```
K = 0.156
T = 72.44;

omega_c = 0.1;
T_f = -1/(omega_c*tan(130*pi/180));
Kpd = (sqrt(1 + (omega_c*T_f))*omega_c)/K;
T_d = T;

%comment back to get the bode plot
%s_num = [K];
%s_den = [T 1 0];

%H_s = tf(s_num, s_den);

%pd_num = [Kpd*T_d Kpd];
%pd_den = [T_f 1];

%H_pd = tf(pd_num, pd_den);

%H_0 = H_s*H_pd;

%figure(1);
%hold on
%bode(H_0)
```

Figure 20: Code used to test the auto-pilot in calm weather.

```
run('../Common files/p5p3_init.m');

sim('p5p3b_model');

t = BODYheading.Time(1:1000);
psi_r(1:500) = 30;
delta = RudderInput.Data(1:1000);
psi = BODYheading.Data(1:1000);

%% Define figure size
width = 10; % cm
height = 10; % cm
fontsize = 10; % points
x = 20; y = 20;

set(0,'DefaultTextInterpreter', 'latex')

%% Set up the figure
fig1 = figure(1);
fig1.Units = 'centimeters';
fig1.Position = [x y width height];

plot(psi_r,'red--');hold on
p = plot(t,psi,t,delta); hold off

p(1).LineWidth = 2;
p(2).LineWidth = 2;
ylim([-20 45])
xlim([0 300])

ax = gca; %get the axes handle of the current axes
ax.FontUnits = 'points';
ax.FontSize = fontsize;
ax.TickLabelInterpreter = 'latex'; %Interpret Tick labels as latex

xlabel('Time [s]')
ylabel('$\psi$, $\psi_r$, $\delta$ [deg]')
legend('\psi_r','\psi', '\delta')

title('Autopilot, only measurement noise')

%Set title to be 1.1 times larger than other fonts
ax.TitleFontSizeMultiplier = 1.1;
```

Figure 21: Code used to test the auto-pilot with current disturbance and measurement noise.

```
run('../Common files/p5p3_init.m');

sim('p5p3c_model');

t = BODYheading.Time(1:1000);
psi_r(1:500) = 30;
delta = RudderInput.Data(1:1000);
psi = BODYheading.Data(1:1000);

%% Define figure size
width = 10; % cm
height = 10; % cm
fontsize = 10; % points
x = 20; y = 20;

set(0,'DefaultTextInterpreter', 'latex')

%% Set up the figure
fig1 = figure(1);
fig1.Units = 'centimeters';
fig1.Position = [x y width height];

%% Plot the data
plot(psi_r,'red--');hold on
p = plot(t,psi,t,delta); hold off

p(1).LineWidth = 2;
p(2).LineWidth = 2;
ylim([-20 45])
xlim([0 300])
ax = gca; %get the axes handle of the current axes
ax.FontUnits = 'points';
ax.FontSize = fontsize;
ax.TickLabelInterpreter = 'latex'; %Interpret Tick labels as latex
% Label the axes, set up legend and title
xlabel('Time [s]')
ylabel('$\psi$, $\psi_r$, $\delta$ [deg]')
legend('\psi_r','\psi', '\delta')

title('Autopilot, current disturbance')

%Set title to be 1.1 times larger than other fonts
ax.TitleFontSizeMultiplier= 1.1;
```

Figure 22: Code used to test the auto-pilot with wave disturbance and measurement noise.

```
run('../Common files/p5p3_init.m');

sim('p5p3d_model');

t = BODYheading.Time(1:1000);
psi_r(1:500) = 30;
delta = RudderInput.Data(1:1000);
psi = BODYheading.Data(1:1000);


%% Define figure size
width = 10; % cm
height = 10; % cm
fontsize = 10; % points
x = 20; y = 20;

set(0,'DefaultTextInterpreter', 'latex')

%% Set up the figure
fig1 = figure(1);
fig1.Units = 'centimeters';
fig1.Position = [x y width height];

%% Plot the data
plot(psi_r,'red--');hold on
p = plot(t,psi,t,delta); hold off

p(1).LineWidth = 2;
p(2).LineWidth = 2;
ylim([-20 45])
xlim([0 300])
% Set the fontsize and units correctly
ax.FontUnits = 'points';
ax.FontSize = fontsize;
ax.TickLabelInterpreter = 'latex'; %Interpret Tick labels as latex
% Label the axes, set up legend and title
xlabel('Time [s]')
ylabel('$\psi$, $\psi_r$, $\delta$ [deg]')
legend('\psi_r','\psi', '\delta')

title('Autopilot, wave disturbance')

%Set title to be 1.1 times larger than other fonts
ax.TitleFontSizeMultiplier = 1.1;
```

Figure 23: Code used for checking the observability of our system.

```
%%%%%%%%%% No disturbances
A_none = [0 1; 0 -1/T];
B_none = [0; K/T];
C_none = [1 0];
D_none = 0;
E_none = 0;
sys_none = ss(A_none, B_none, C_none, D_none);
Ob_none = rank(obsv(sys_none))

%%%%%%%%%% Only current disturbances
A_current = [0 1 0; 0 -1/T -K/T; 0 0 0];
B_current = [0; K/T; 0];
C_current = [1 0 0];
D_current = 0;
E_current = [0; 0; 1];
sys_current = ss(A_current, B_current, C_current, D_current);
Ob_current = rank(obsv(sys_current))

%%%%%%%%%% Only wave disturbances
A_waves =  [0 1 0 0;
            -(omega_0)^2 -2*lambda*omega_0 0 0;
            0 0 0 1;
            0 0 0 -1/T];
B_waves = [0; 0; 0; K/T];
C_waves = [0 1 1 0];
D_waves = 0;
E_waves = [0; K_w; 0; 0];
sys_waves = ss(A_waves, B_waves,C_waves, D_waves);
Ob_waves = rank(obsv(sys_waves))

%%%%%%%%%% Both current and wave disturbances
A_all = [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_all = [0; 0; 0; K/T; 0];
C_all = [0 1 1 0 0];
D_all = 0;
E_all = [0 0; K_w 0; 0 0; 0 0; 0 1];
sys_all = ss(A_all, B_all, C_all, D_all);
Ob_all = rank(obsv(sys_all))
```

Figure 24: Initialization used for code in creating the Kalman filter. Name of file: "p5p5_init.m"

```
K = 0.156;
T = 72.44;
omega_c = 0.1;
T_f = -1/(omega_c*tan(130*pi/180));
Kpd = (sqrt(1 + (omega_c*T_f))*omega_c)/K;
T_d = T;
%%%%%%% Variable initialization
load('wave.mat');
[Sxx, f] = pwelch(psi_w(2,:).*(pi/180), 4096, [], [], 10); % [Ws]
Sxx = Sxx*(1/(2*pi)); %convert to [W s/rad]
f = f*2*pi; %comes as [Hz] = [1/s] -> convert to [rad/s]
[sigma_sq,n] = max(Sxx); %finc max
sigma = sqrt(sigma_sq);
omega_0 = f(n); %resonance frequenzy
%Analytical PSD
Pxx = @(lambda, omega) ((2*lambda*omega*omega_0*sigma).^2) ...
    ./ ((omega_0^2 - omega.^2).^2 + (2*lambda*omega*omega_0).^2);
lambda = lsqcurvefit(Pxx, 0.1, f, Sxx); %Damping factor

K_w = 2*lambda*omega_0*sigma;
%%%%%% Continous system
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];
%%%%%%Given in assignment text
Q = [30 0;
       0 1e-6];
P_0_apriori = [1 0 0 0 0;
               0 0.013 0 0 0;
               0 0 pi^2 0 0;
               0 0 0 1 0;
               0 0 0 0 2.5e-3];
x_0_apriori = zeros(5,1);
%%%%% End of handed out values
```

Figure 25: Code used for creating the Kalman filter.

```
load('../Common files/ScopeDataNoise.mat');
run('../Common files/p5p5_init.m');

%Discretization
sys = ss(A, B, C, D);
Ts = 0.1;
sysd = c2d(sys, Ts);

van_loan = [A, E*Q*E';
            zeros(5), -(A')];
van_loan = expm(van_loan*Ts);
Q_w = van_loan(1:5,6:10)*(van_loan(1:5,1:5)');
v = ScopeData.signals.values;
R_v = var(v);
R_v_bar = R_v/Ts;

%Making data struct used in Kalman filter in simulink
data.A = sysd.A;
data.B = sysd.B;
data.C = sysd.C;
data.Q = Q_w;
data.R = R_v_bar;
data.P = P_0_apriori;
data.xhat0 = x_0_apriori;
data.I = eye(5);


%Run sim
sim('p5p5c_model.mdl');

t = BODYheading.Time(1:5000);

psi_r(1:5000) = 30;

psi_measured = BODYheading.Data(1:5000);
psi_filtered = psi_filtered.Data(1:5000);
bias = RudderBias.Data(1:5000);

%Plot the results...
```

Figure 26: Code used for testing the Kalman filter with rudder-bias.

```
1
2  load('../Common files/ScopeDataNoise.mat');
3  run('../Common files/p5p5_init.m');
4
5  %Discretization
6  sys = ss(A, B, C, D);
7  Ts = 0.1;
8  sysd = c2d(sys, Ts);
9
10 van_loan = [A, E*Q*E';
11               zeros(5), -(A')];
12 van_loan = expm(van_loan*Ts);
13 Q_w = van_loan(1:5,6:10)*(van_loan(1:5,1:5)');
14 v = ScopeData.signals.values;
15 R_v = var(v);
16 R_v_bar = R_v/Ts;
17
18 %Making data struct used in Kalman filter in simulink
19 data.A = sysd.A;
20 data.B = sysd.B;
21 data.C = sysd.C;
22 data.Q = Q_w;
23 data.R = R_v_bar;
24 data.P = P_0_apriori;
25 data.xhat0 = x_0_apriori;
26 data.I = eye(5);
27
28 %Run sim
29 sim('p5p5d_model.mdl');
30
31 %load('..\Boat files\workspaceData\5.3')
32 t = BODYheading.Time(1:5000);
33
34 psi_r(1:5000) = 30;
35 psi_measured = BODYheading.Data(1:5000);
36 psi_filtered = psi_filtered.Data(1:5000);
37 delta = UnbiasedRudderInput.Data(1:5000);
38 bias = RudderBias.Data(1:5000);
39
40 %Plot the results...
```

Figure 27: Code used for testing the Kalman filter with wave filtering and rudder-bias.

```
1
2  load('../Common files/ScopeDataNoise.mat');
3  run('../Common files/p5p5_init.m');
4
5  %Discretization
6  sys = ss(A, B, C, D);
7  Ts = 0.1;
8  sysd = c2d(sys, Ts);
9
10 van_loan = [A, E*Q*E';
11               zeros(5), -(A')];
12 van_loan = expm(van_loan*Ts);
13 Q_w = van_loan(1:5,6:10)*(van_loan(1:5,1:5)');
14 v = ScopeData.signals.values;
15 R_v = var(v);
16 R_v_bar = R_v/Ts;
17
18 %Making data struct used in Kalman filter in simulink
19 data.A = sysd.A;
20 data.B = sysd.B;
21 data.C = sysd.C;
22 data.Q = Q_w;
23 data.R = R_v_bar;
24 data.P = P_0_apriori;
25 data.xhat0 = x_0_apriori;
26 data.I = eye(5);
27
28 %Run sim
29 sim('p5p5e_model.mdl');
30
31
32 t = BODYheading.Time(1:5000);
33
34 psi_r(1:5000) = 30;
35 psi_measured = BODYheading.Data(1:5000);
36 psi_filtered = psi_filtered.Data(1:5000);
37 delta = UnbiasedRudderInput.Data(1:5000);
38 bias = RudderBias.Data(1:5000);
39 wave_influence = psi_w(2,1:5000);
40 wave_influence_est = psi_w_est.Data(1:5000);
41
42 %Plot results...
```

Figure 28: Actual implementation of the Kalman filter. This function is performing the update and prediction step of the Kalman algorithm. Script run inside Simulink fcn-block named Kalman filter.

```
function [xi, psi_w, psi, r, bias] = fcn(u,y,data)
persistent xpri xhat P;

A = data.A;
B = data.B;
C = data.C;
Q = data.Q;
R = data.R;
I = data.I;
if isempty(xpri)
    xpri = data.xhat0;
    P = data.P;
end

K = P*C'/(C*P*C' + R);
xhat = xpri + K*(y - C*xpri);
P = (I - K*C)*P*(I - K*C)' + K*R*K';

xpri = A*xhat + B*u;
P = A*P*A' + Q;

xi = xhat(1);
psi_w = xhat(2);
psi = xhat(3);
r = xhat(4);
bias = xhat(5);
```

B Simulink models

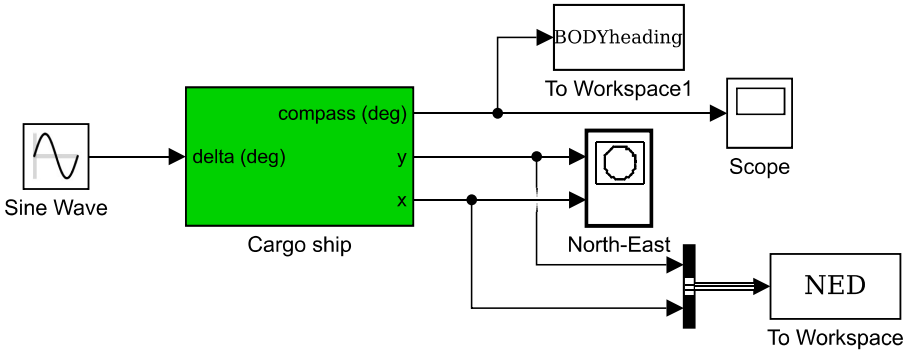


Figure 29: Handed out cargo-ship system with sine wave input and measurements on output.

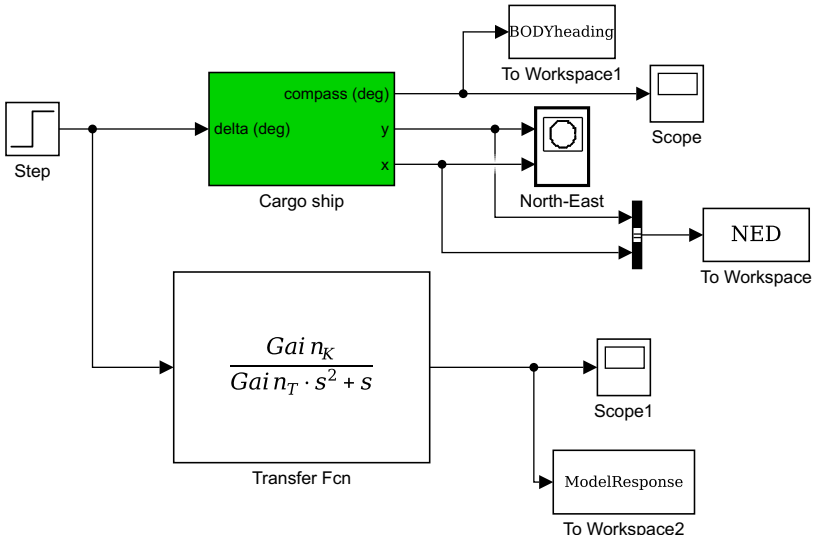


Figure 30: Comparison of simulated cargo ship system with theoretical model.

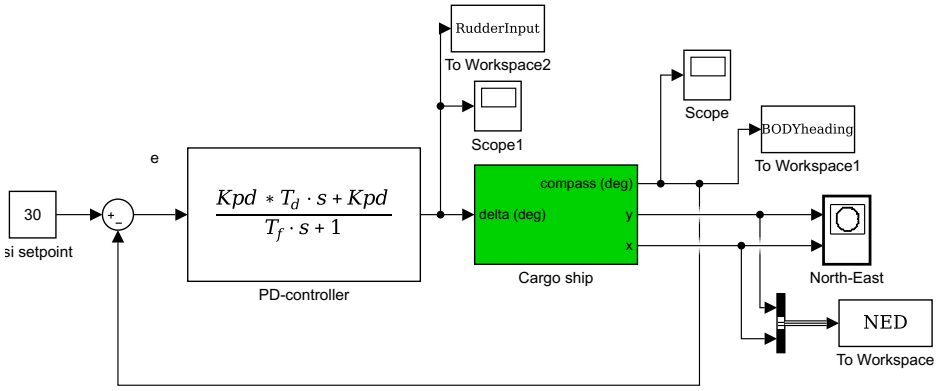


Figure 31: Cargo-ship system with PD-controller.

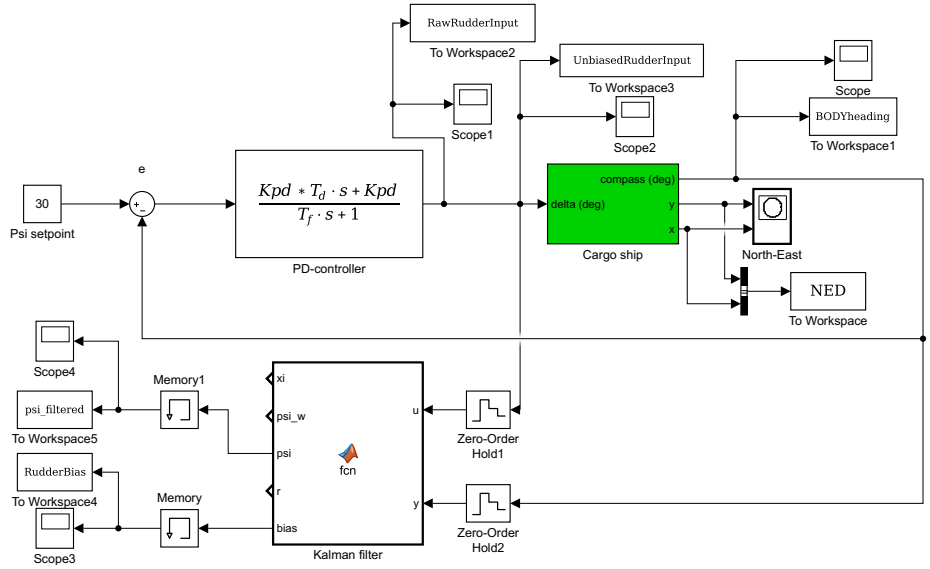


Figure 32: Cargo-ship system with PD-controller and Kalman filter.

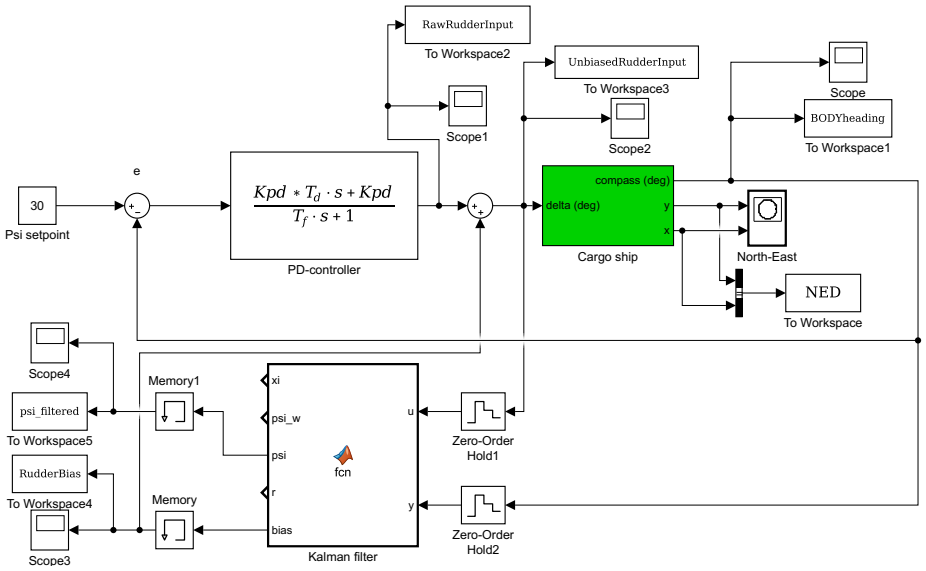


Figure 33: Cargo-ship system with PD-controller and Kalman filter and rudder-bias compensation.

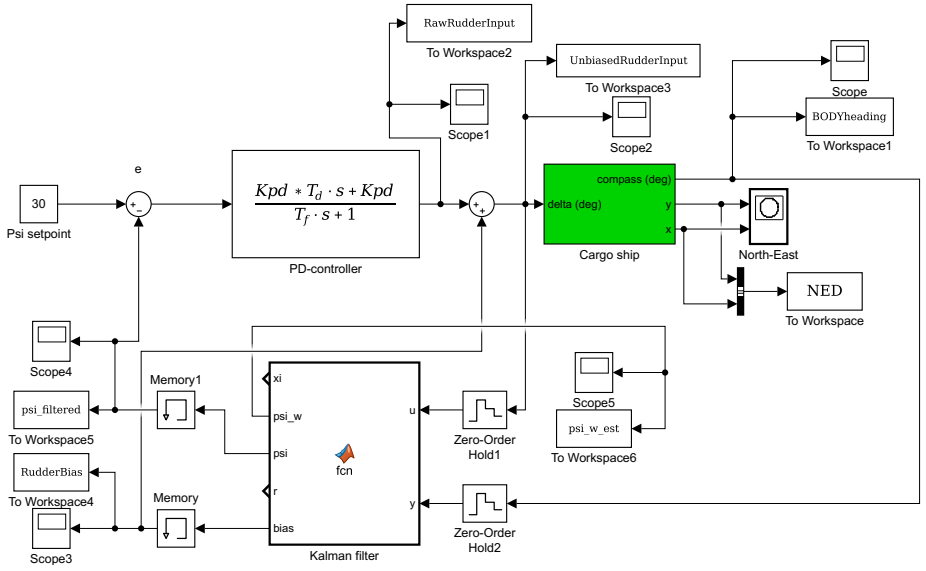


Figure 34: Cargo-ship system with PD-controller and Kalman filter, rudder-bias compensation and wave filtering.

References

[1] .

[2] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, Incorporated, 2014.

[3] B.A. Foss J.G. Balchen T. Andresen. *Reguleringsteknikk*. Institutt for teknisk kybernetikk, NTNU, 2016.