Kalman filter on cargo ship

Åshild Lien, Lars C. M. van der Lee, Johansen, Alexander

October 2017

Contents

1 Part 1: Identification of the boat parameters

1.1 Task a) Finding transfer function

In this task, we want to describe the system without considering the disturbances from waves and current. We start by finding a transfer function from δ to ψ, where δ is the rudder angle relative to the BODY frame, and ψ is the angle between the north axis and the longitudinal axis of the ship. We use the first order Nomoto model from the assignment [2]. The Nomoto equations can be simplified by removing the current and wave terms. This results in:

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

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

where r is rotational velocity about the ship's z-axis, from top to bottom of the ship. T and K are time and game constants, respectively. Applying the Laplace transform to the equations gives

$$s\psi = r \tag{2a}$$

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

From this we obtain

$$s\psi = \frac{K}{T}\frac{\delta}{s + \frac{1}{T}} \tag{3}$$

which gives us the transfer function from δ to ψ:

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

Using this information, we can derive the units for K and T. As r has unit $\left[\frac{rad}{s^2}\right]$ from eq. (1b), we obtain that T has unit $[s]$ and K has unit $\left[\frac{1}{s}\right]$.

1.2 Task b) Identifying T and K parameters

We want to identify the parameters T and K from task 1a) in smooth weather conditions. We turned off current and waves in Simulink, and plotted the response when the input is a sine wave with amplitude 1 and frequencies $\omega_1 = 0.005$ rad/s and $\omega_2 = 0.05$ rad/s. The amplitude of the output then equals $|H(j\omega_1)|$ and $|H(j\omega_2)|$.
By inserting $s = j\omega$ into the transfer function and taking the absolute value, we obtained

$$H(j\omega) = \frac{K}{j\omega(1 + jT\omega)} \tag{5}$$

$$|H(j\omega)| = \frac{K}{\omega\sqrt{1 + T^2\omega^2}} \tag{6}$$

We added a signal generator to the angle input in MATLAB, and plotted the response compass angle with ω_1 and ω_2.

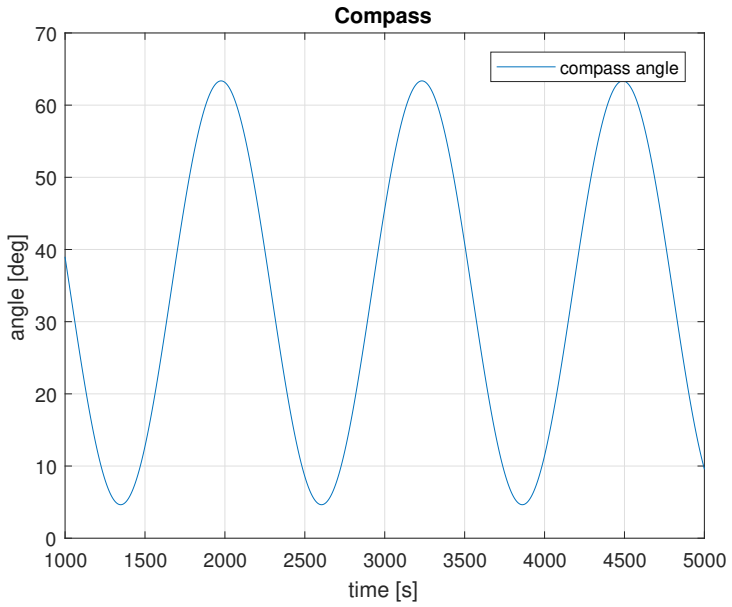

Figure 1: Response of the compass with ω_1

We see from Figure 1 and Figure 2 that the amplitude of the output response when the input frequency is ω_1 is 29.45, and the amplitude for ω_2 is 0.95. Thus, we obtain two equations,

$$\frac{K}{\omega_1\sqrt{1+T^2\omega_1^2}} = 29.45 \tag{7a}$$

$$\frac{K}{\omega_2\sqrt{1+T^2\omega_2^2}} = 0.95 \tag{7b}$$

and solve for T and K by using MATLAB. The code can be found in Appendix B.1 The obtained values are:

$$T = \pm 70.54702\,\mathrm{s} \tag{8}$$

$$K = 0.15566\,\mathrm{s}^{-1} \tag{9}$$

1.3 Task c) Identifying T and K parameters, with measurement noise and waves

We add waves to the system and include a measurement noise in the measurement of the compass. This results in noisy signals as can be seen in Figures

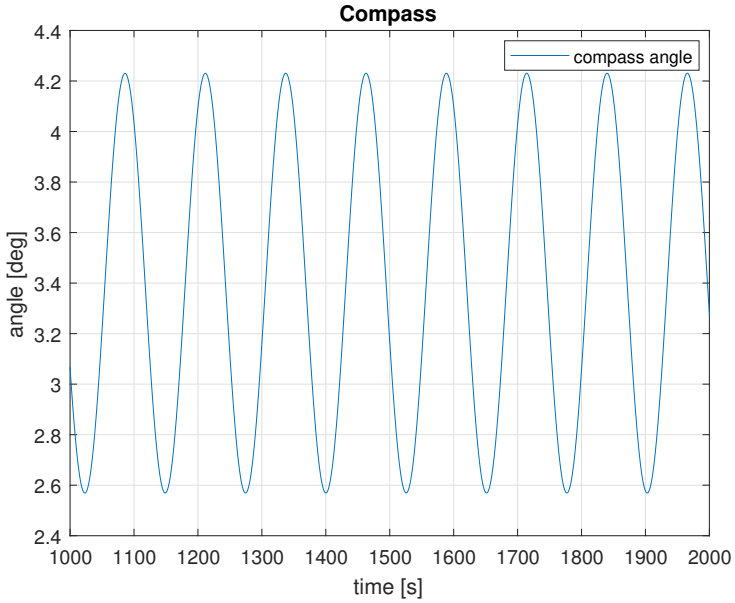

Figure 2: Response of the compass with ω_2

3 and 4. In Figure 3 one can estimate an accurate value for the amplitude as the noise has a small amplitude relative to the original signal. In Figure 4 the amplitude of the system is obscured by the signal noise. By using MATLABs built in function `NonLinearModel.fit(X, y, modelfun, beta0)` where `X` is time, `y` is our values, `modelfun` is defined as `y` $\sim$ `b(1)*sin(`ω`*X + b(2))+b(3)` and `beta0` are some manually estimated values. From this we obtain coefficients that seem to coincide with what we expect of the system, they also resemble the response we saw without noise. A closeup of the nonlinear regression can be seen in Figure 5 and Figure 6. Using the approximated values we obtain,

$$\frac{K}{\omega_1\sqrt{1+T^2\omega_1^2}} = 29.35 \tag{10a}$$

$$\frac{K}{\omega_2\sqrt{1+T^2\omega_2^2}} = 0.82 \tag{10b}$$

and solve for T and K:

$$T = \pm 73.612\,\mathrm{s} \tag{11}$$

$$K = 0.156\,\mathrm{s}^{-1} \tag{12}$$

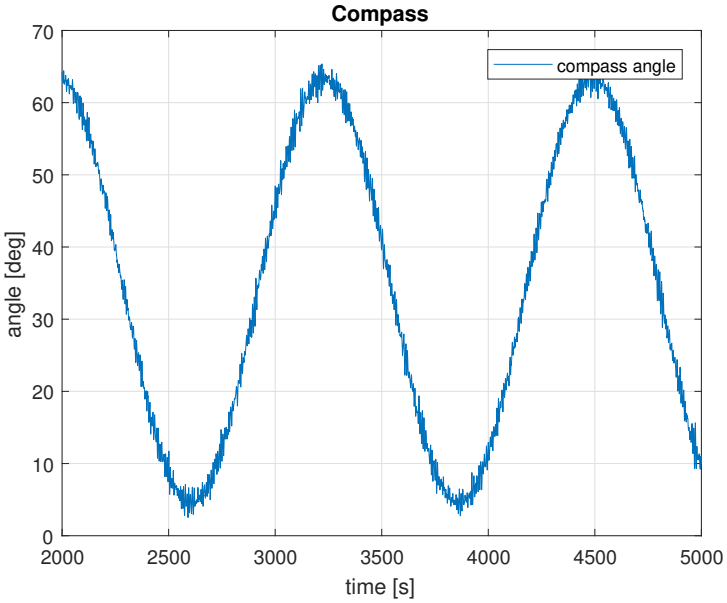

Figure 3: Response of the compass with noise, waves and ω_1

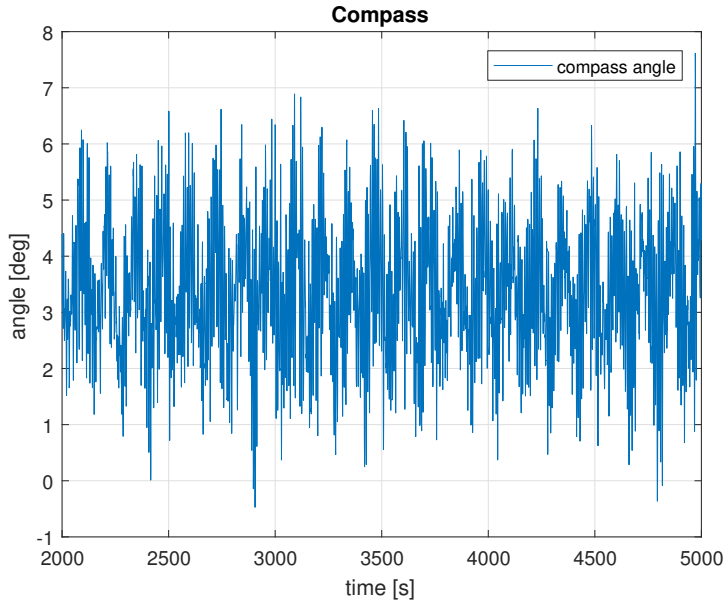

Figure 4: Response of the compass with noise, waves and ω_2

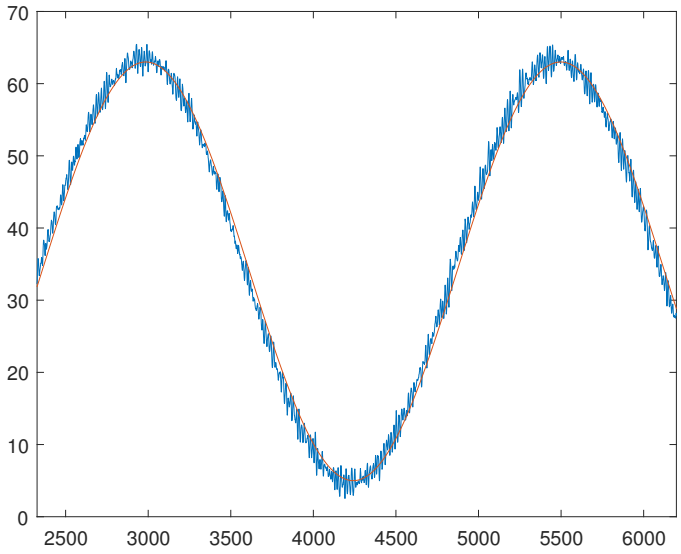

Figure 5: Response of the compass with noise, waves and ω_1 (blue) and the nonlinear regression(red)

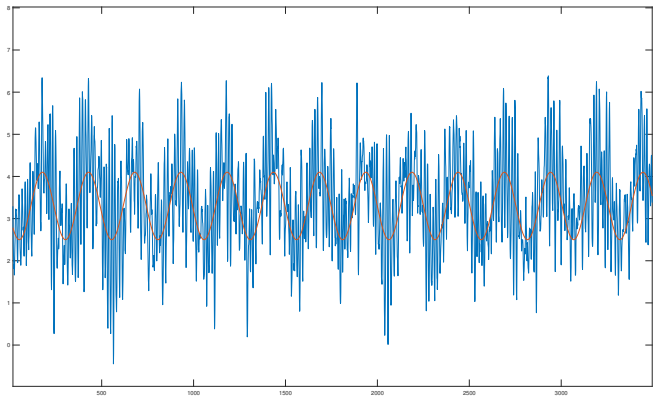

Figure 6: Response of the compass with noise, waves and ω_2(blue) and the nonlinear regression(red)

1.4 Task d) Comparing ship response with model

In this task we applied a step input at t = 0 to compare the ship with our model. The implementation of the transfer function can be seen in Appendix A.1, Figure 19. The physical and theoretical response can be seen in Figure 7.

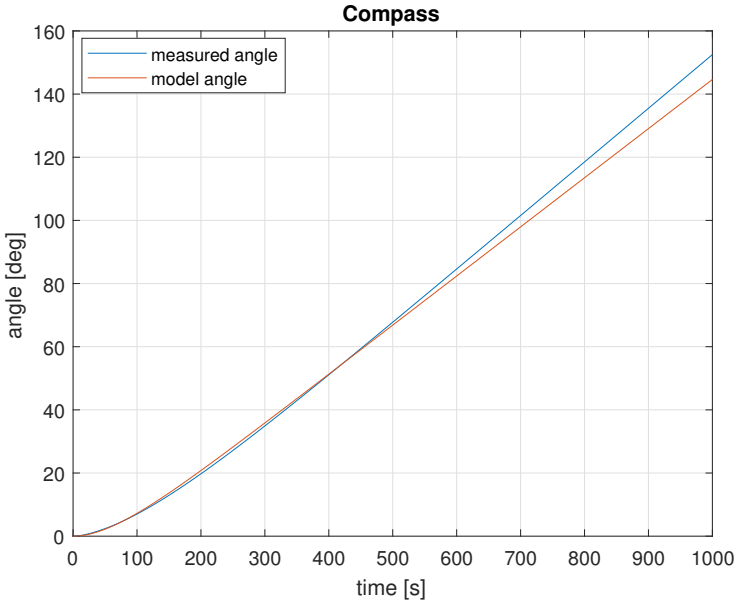

Figure 7: Response of the modeled ship and the measurement of the actual ship

The model is a good approximation the beginning, but the model and the system start to diverge as time goes on. This is good enough for our purposes as we will be able to monitor the error in our system and adjust for it. Therefor it is most important that the model approximates the system well for relatively small values of t, in which case our model is more than satisfactory.
This value might be further improved by trying to fine tune K and T, though the current values seem very close to optimal. The main way to get a better model would therefore be to incorporate more physical models of the ship in our system.

2 Part 2: Identification of wave spectrum model

2.1 Task a) Power Spectral Density function

We want to find an estimate $S_{\psi_w}(\omega)$ of the Power Spectral Density function of ψ_w. We use the MATLAB function `[pxx,f] = pwelch(x, window, noverlap, nfft, fs)`. The input `x` is a vector with ψ_w, which is the wave influence on the compass measurement. As this vector is given in degrees, the resulting `pxx`

and `f` have to be scaled with $\frac{1}{2\pi}$ and 2π to obtain the desired units. We use `window` of 4096, and leave `noverlap` and `nfft` unspecified. We have a sampling frequency of 10 Hz, thus the parameter `fs` is set to 10. The return parameter `pxx` is the PSD estimate of the input signal, and `f` is a frequency vector. The code can be found in Appendix B.2.

2.2 Task b) Analytical transfer function and PSD

We wish to find an analytical expression of the transfer function from the noise w_w to ψ_w, the wave influence. We start with the Nomoto equations:

$$\dot{\xi_w} = \psi_w \tag{13a}$$
$$\dot{\psi}_w = -\omega_0^2 \xi - 2\lambda\omega_0\psi_w + K_w w_w \tag{13b}$$

We apply the Laplace transform, and obtain

$$s\xi = \psi_w \tag{14a}$$
$$s\psi_w = -\omega_0^2 \xi - 2\lambda\omega_0\psi_w + K_w w_w \tag{14b}$$

We insert $\xi = \frac{\psi_w}{s}$ into eq. (14b), and obtain the transfer function

$$\hat{H}(s) = \frac{\psi_w}{w_w} = \frac{sK_w}{s^2 + 2\lambda\omega_0 s + \omega_0^2} \tag{15}$$

We next want to find an analytical expression for the PSD function for ψ_w, which is given by

$$P_{\psi_w} = \left|\hat{H}(j\omega)\right|^2 P_{w_w} \tag{16}$$

As the input w_w is Gaussian white noise with unity variance, we get $P_{w_w} = 1$, because the PSD for white noise is the variance of the noise. We thus only need to compute $\left|\hat{H}(j\omega)\right|^2$ in order to obtain the PSD:

$$\hat{H}(j\omega) = \frac{j\omega K_w}{-\omega^2 + j2\lambda\omega_0\omega + \omega_0^2} \tag{17}$$
$$P_{\psi_w} = \left|\hat{H}(j\omega)\right|^2 = \frac{\omega^2 K_w^2}{\left(\omega_0^2 - \omega^2\right)^2 + \left(2\lambda\omega_0\omega\right)^2} = \frac{\omega^2 K_w^2}{\omega_0^4 + \omega^4 + 2\omega_0^2\omega^2\left(2\lambda^2 - 1\right)} \tag{18}$$

2.3 Task c) Finding ω_0

We find the resonance frequency ω_0 at the maximum value of S_x. This frequency was found to be $\omega_0 = 0.7823$ using MATLABs `max(pff)` function. A plot of can be seen in Section 2.3. The frequency ω_0 corresponding to the maximum value of S_x is marked by what looks like a orange balloon at the top of a mountain, flying in a very long string.

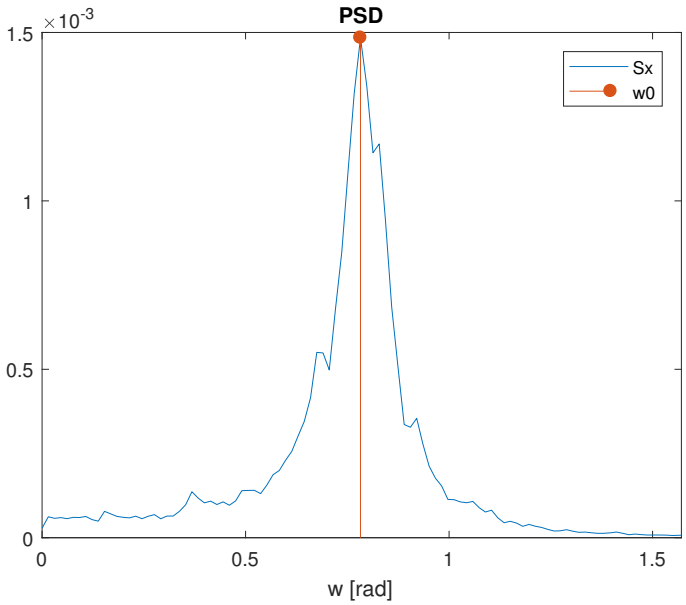

Figure 8: Finding the maximum value of S_x

2.4 Task d) Finding damping factor

We wish to find the damping factor λ. We define $K_w = 2\lambda\omega_0\sigma$, with ω_0 from problem c). σ^2 is the peak value of P_{ψ_w}, which we defined as the peak value of the estimate, as this is what we want P_{ψ_w} to match. We found λ by trial and error. Using MATLAB, we plotted several values for λ inserted in P_{ψ_w} against the estimated S_{ψ_w}. We see from Figure 9 that $\lambda = 0.08$ is a good choice.
In order to find a better estimate for λ, we used MATLAB's `lsqcurvefit` function. This function takes four input parameters; a non-linear function `fun(x, xdata)`, a starting point `x0`, and two sets of data, `xdata` and `ydata`. In our case, `fun(x, xdata)` is the PSD P_{ψ_w}, where `x` is the parameter we want to estimate, in our case λ, and the dataset `xdata` is given from the first row of `psi_w`. `ydata` is the second row of the same `psi_w` matrix. Further, we used the obtained λ from trial and error, $\lambda = 0.08$, as the starting point `x0`. The `lsqcurvefit` function returns a value of λ that gives the best fit. The obtained λ is 0.0857. The code can be found in Appendix B.2.

3 Part 3: Control system design

3.1 Task a) PD controller

We want to design a PD controller so that the ship will follow a reference course ψ_r. We want the phase margin of the open-loop system to be 50 degrees, and

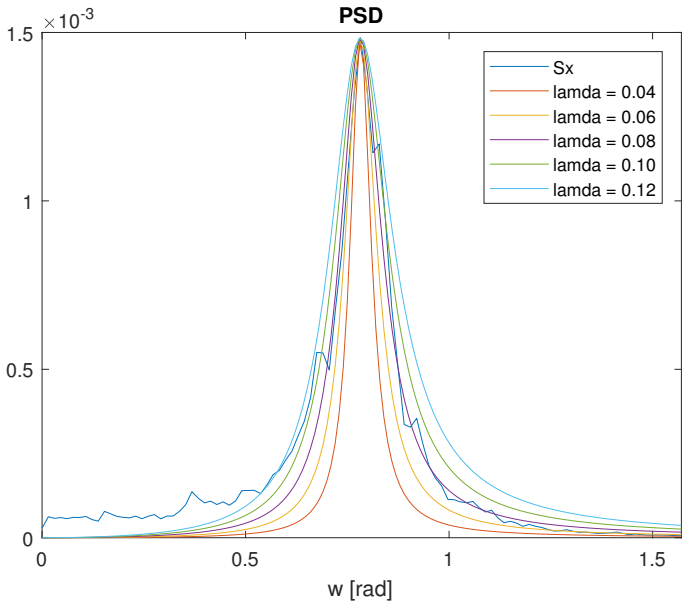

Figure 9: P_{ψ_w} with different λ's, plotted against S_{ψ_w}.

the cutoff frequency ω_c should be 0.10 rad/s. The PD controller is given by

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

where K_{pd} is the controller gain, and T_d and T_f are time constants. The controller is designed based on the transfer function from δ to ψ given in Equation (4). We get an open-loop system given by

$$H_{pd}(s) * H_{ship}(s) = \frac{K_{pd}K}{s}\frac{1+T_d s}{(1+T_f s)(1+Ts)} \tag{20}$$

We choose $T_d = T$, so that they cancel each other out. We then obtain

$$H_{pd}(s) * H_{ship}(s) = \frac{K_{pd}K}{s}\frac{1}{1+T_f s} \tag{21}$$

Next, we find the cutoff frequency, which is the frequency ω_c for which the absolute value of the ship transfer function multiplied with the transfer function for the controller equals 1. $|H_{pd}(j\omega_c) * H_{ship}(j\omega_c)| = 0\text{dB} = 1$:

$$\left|\frac{K_{pd}K}{j\omega_c - T_f\omega_c^2}\right| = 1. \tag{22}$$

Rewriting, we obtain

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

The phase margin should be 50 degrees and is given by

$$\phi = \angle\left(H_{pd}(j\omega_c) * H_{ship}(j\omega_c)\right) - (-180^\circ) \tag{24}$$

As K_{pd} and K are real numbers, they do not contribute to the phase. Thus, we are left with

$$\angle\left(H_{pd}(j\omega_c) * H_{ship}(j\omega_c)\right) = \angle\frac{1}{s} + \angle\frac{1}{1+T_f s} = -90^\circ - \arctan(T_f\omega_c) \tag{25}$$

Thus, we obtain from eqs. (24) and (25)

$$T_f = \frac{\tan(50^\circ - 180^\circ + 90^\circ)}{\omega_c} = 8.4\,\mathrm{s} \tag{26}$$

Rewriting eq. (23) gives

$$K_{pd} = \frac{\omega_c\sqrt{1+T_f^2\omega_c^2}}{K}. \tag{27}$$

Inserting gives us a final value for K_{pd}

$$K_{pd} = 0.8390. \tag{28}$$

3.2 Task b) Simulation with measurement noise

We simulated the system with the PD controller and a reference course $\psi_r = 30^\circ$. The result can be seen in Figure 10. We see that the ship is able to hold the reference angle. The rudder input is very high in the beginning, as the error into the controller is large when the system starts, the actual ship is only able to turn the rudder 45°so any input above this will be cut off at 45. The ship needs about 150 seconds to reach the reference course. The ship model is specified to only hold for compass angles $\psi = \pm 35^\circ$. We see from the plot that this contraint is never violated.

3.3 Task c) Simulation with measurement noise and current

We then added the current disturbance to the system. We see from Figure 11 that the ship is not able to reach the reference course, but has a stationary offset of 3.5°. The input u never goes to zero, as the regulator attempts to counteract the water-current. Because we are using a PD-controller when the ship reaches this offset the turning force of the ship will equal the force it is being pushed by the current. As a PD-controller will not change unless the error changes the boat will never reach an equilibrium state unless the current changes or an integral effect is introduced into the regulator.

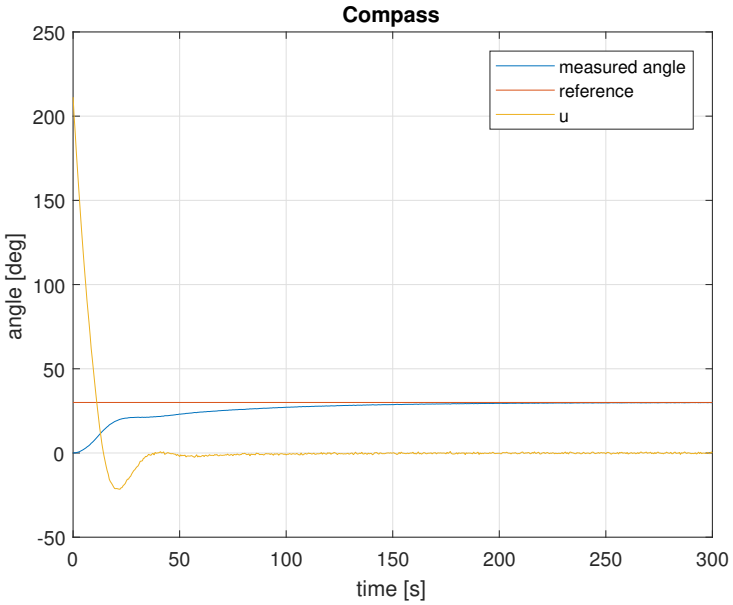

Figure 10: Ship course, rudder input and reference course with measurement noise.

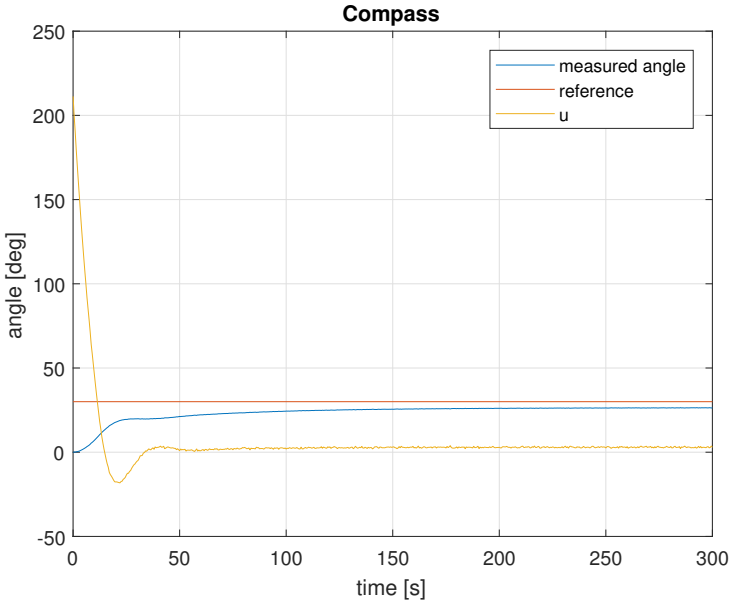

Figure 11: Ship course, rudder input and reference course with measurement noise and current.

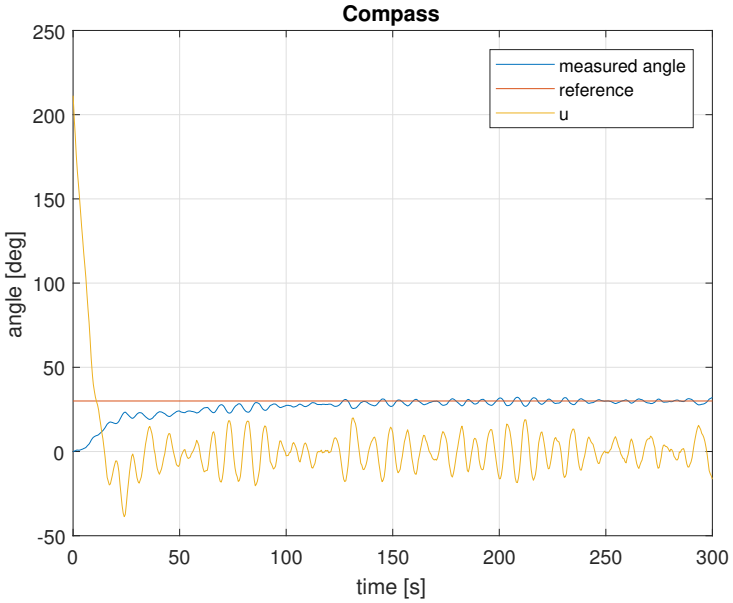

Figure 12: Ship course, rudder input and reference course with measurement noise and waves.

3.4 Task d) Simulation with measurement noise and waves

We removed the current disturbance and added waves to the system. The result can be seen in Figure 12. As we see, the PD controller reaches the reference course, but the controller is not able to remove the high frequent noise from the waves. All the pushing around caused by the waves means that the regulator constantly has to change the rudder angle. Such response is not optimal as it will cause excessive wear on the actuator. We will later look at how to solve this using a Kalman filter such that we don't need to change the amplitude of the input as much.

4 Part 4: Observability

4.1 Task a) Finding matrices

The Nomoto equations can be written as a matrix equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w}, \qquad \mathbf{y} = \mathbf{C}\mathbf{x} + v \tag{29}$$

with $\mathbf{x} = [\xi_w,\ \psi_w,\ \psi,\ r,\ b]^\top$, $u = \delta$ and $\mathbf{w} = [w_w,\ w_b]^\top$.

The Nomoto equations are given by

$$\dot{\xi} = \psi_w \tag{30a}$$
$$\dot{\psi}_w = -\omega_0^2 \xi - 2\lambda\omega_0 \psi_w + K_w w_w \tag{30b}$$
$$\dot{\psi} = r \tag{30c}$$
$$\dot{r} = -\frac{1}{T} r + \frac{K}{T}(\delta - b) \tag{30d}$$
$$\dot{b} = w_b \tag{30e}$$
$$y = \psi + \psi_w + v \tag{30f}$$

From this, we find

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{31}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix} \tag{32}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{33}$$

$$\mathbf{E} = \begin{bmatrix} 0 & 0 \\ K_w & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{34}$$

4.2 Task b) Observability without disturbance

We wish to check if the system is observable when we do not consider the wave and current disturbances. In order to be observable, the rank of the observability matrix $\mathcal{O}$, given by

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{35}$$

must be equal to the number of rows in $\mathbf{A}$.

Without the disturbances, the matrices $\mathbf{A}$ and $\mathbf{C}$ reduce to

$$\mathbf{A_{calm}} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix}, \qquad \mathbf{C_{calm}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{36}$$

By using the MATLAB command `obsv()`, which takes two arguments `A` and `C` corresponding to $\mathbf{A_{calm}}$ and $\mathbf{C_{calm}}$, we obtained

$$\mathcal{O}_{\text{calm}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{37}$$

We see that $\text{rank}(\mathcal{O}_{\text{calm}}) = 2$, witch is equal to the number of rows in $\mathbf{A_{calm}}$. Thus, the system is observable when the current and wave disturbances are not considered.

4.3 Task c) Observability with current

With only the current disturbance, the matrices $\mathbf{A}$ and $\mathbf{C}$ reduce to

$$\mathbf{A_c} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{C_c} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{38}$$

By using the MATLAB command `obsv()`, which takes two arguments `A` and `C` corresponding to $\mathbf{A_c}$ and $\mathbf{C_c}$, we obtained

$$\mathcal{O}_{\text{c}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \end{bmatrix} \tag{39}$$

We see that $\text{rank}(\mathcal{O}_{\text{c}}) = 3$, witch is equal to the number of rows in $\mathbf{A_c}$. Thus, the system is observable when the wave disturbances are ignored.

4.4 Task d) Observability with waves

With only the wave disturbance, the matrices $\mathbf{A}$ and $\mathbf{C}$ reduce to

$$\mathbf{A_w} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix}, \qquad \mathbf{C_w} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \tag{40}$$

By using the MATLAB command `obsv()`, which takes two arguments `A` and `C` corresponding to $\mathbf{A_w}$ and $\mathbf{C_w}$, we obtained

$$\mathcal{O}_{\text{w}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \end{bmatrix} \tag{41}$$

We see that $\text{rank}(\mathcal{O}_{\text{w}}) = 4$, which is equal to the number of rows in $\mathbf{A_w}$. Thus, the system is observable when the currents are not considered.

4.5 Task e) Observability with current and waves

With both current and waves, the matrices $\mathbf{A}$ and $\mathbf{C}$ will be as given in Equation (31) and Equation (33). By using the MATLAB command `obsv()`, witch takes two arguments `A` and `C` corresponding to $\mathbf{A}$ and $\mathbf{C}$, we obtained

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ -0.6120 & -0.1341 & 0 & 1 & 0 \\ 0.0821 & -0.5941 & 0 & -0.0142 & -0.0022 \\ 0.3636 & 0.1617 & 0 & 0.0002 & 0 \\ -0.0990 & 0.3419 & 0 & 0 & 0 \end{bmatrix} \tag{42}$$

We see that $\text{rank}(\mathcal{O}) = 5$, witch is equal to the number of rows in $\mathbf{A}$. Thus, the system is observable in waves and currents as well.

As we have seen, the system is observable both with and without the different disturbances. This means that we can determine the behaviour of the ship when we know the output [3], no matter the weather conditions. This is an important result for the next task, which will cover the Kalman filter. In order to apply the Kalman filter, the system must be observable. This is because the Kalman filter computes estimates of the system states from knowledge of the output and input to the system. This would not give meaningful results if the system was unobservable.

5 Part 5: Discrete Kalman filter

5.1 Task a) Discretization

We want to discretize the system in problem 4, part a), with a sample frequency of 10 Hz. The system should be discretized with exact discretization. We used the MATLAB command `[Ad, Bd] = c2d(A, B, T)` where `Ad` and `Bd` are the discretized versions of the argument matrices `A` and `B`, and the argument `T` is the sample time. In our case, as the sampling frequency is 10 Hz, the sample time is $T = 1/10\,\text{s} = 0.1\,\text{s}$. To obtain the discrete version of $\mathbf{E}$, we used the same command with `A` and `E`.

$$\mathbf{A}_d = \begin{bmatrix} 0.9970 & 0.0992 & 0 & 0 & 0 \\ -0.0607 & 0.9836 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & 1.102 * 10^{-5} \\ 0 & 0 & 0 & 0.9986 & -0.0002 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{43}$$

$$\mathbf{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0.0110 \\ 0.2205 \\ 0 \end{bmatrix} \tag{44}$$

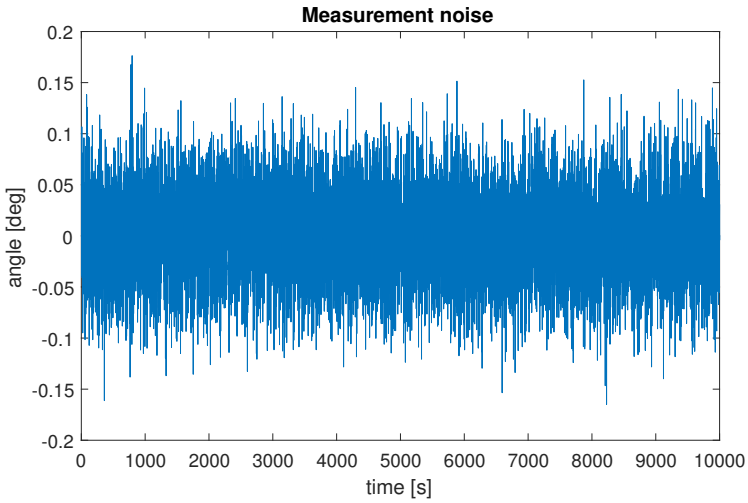

Figure 13: Measurement noise when compass stays at zero.

$$\mathbf{E}_d = \begin{bmatrix} 2.57 * 10^{-5} & 0 \\ 5.13 * 10^{-4} & 0 \\ 0 & -3.68 * 10^{-7} \\ 0 & -1.10 * 10^{-5} \\ 0 & 0.1 \end{bmatrix} \tag{45}$$

The matrix $\mathbf{C}$ does not change when the system is discretized.

5.2 Task b) Variance estimation

We want to estimate the variance of the measurement noise. In order to do that, we simulated the boat with rudder input angle set to zero, and without waves and current. Thus, only the measurement noise is present in the compass measurement. The resulting plot of the measured angle can be seen in Figure 13. We stored the measurement in a vector in MATLAB, and used the command `var` to find the variance. The command takes one input argument, which is the measurement vector. Resulting variance is

$$\sigma_w^2 = 0.002\,\text{deg}^2 \tag{46}$$

This means that our compass has very little variance, thus it is a very good sensor.

5.3 Task c) Discrete Kalman filter

In this task, we want to implement a discrete Kalman filter in Simulink using a MATLAB s-function.

Our discrete system is of the following form:

$$\mathbf{x}[k+1] = \mathbf{A}_d\mathbf{x}[k] + \mathbf{B}_d\mathbf{u}[k] + \mathbf{E}_d\mathbf{w}[k] \tag{47}$$

$$y[k] = \mathbf{C}_d\mathbf{x}[k] + v[k] \tag{48}$$

The discrete covariance matrix for $\mathbf{w}$ was given in the assignment text as $\mathbf{Q}$:

$$E\left[\mathbf{w}\mathbf{w}^\top\right] = \mathbf{Q} = \begin{bmatrix} 30 & 0 \\ 0 & 10^{-6} \end{bmatrix} \tag{49}$$

The discrete covariance matrix for v is a scalar, given as $E\left[v^2\right] = R = \sigma_w^2/T$, where σ_w^2 is the variance found in part 5b) and T is the sampling time. In addition, we are given an initial estimate of the error covariance matrix $\mathbf{P}_0^- = E\left[\mathbf{e}^-\mathbf{e}^{-\top}\right]$, where $\mathbf{e}^-$ is the estimation error, i.e. the difference between the measured state and the estimated: $\mathbf{e}^-[k] = \mathbf{x}[k] - \hat{\mathbf{x}}_0^-[k]$. The initial a priori estimate of $\mathbf{x}$ is given as $\hat{\mathbf{x}}_0^- = [0,\, 0,\, 0,\, 0,\, 0]^\top$, and the error covariance matrix is given as

$$\mathbf{P}_0^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.013 & 0 & 0 & 0 \\ 0 & 0 & \pi^2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2.5 * 10^{-3} \end{bmatrix} \tag{50}$$

In the following equations, we omit $[k]$ from the equations where the time variable k does not provide any information of importance.
With a priori knowledge of the state, $\hat{\mathbf{x}}^-$, we wish to find a better a posteriori estimate, $\hat{\mathbf{x}}$, given by

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}^- + \mathbf{L}\left(y - \mathbf{C}_d\hat{\mathbf{x}}^-\right) \tag{51}$$

where $\mathbf{L}$ is a blending factor we obtain by applying the Kalman filter. The Kalman filter determines $\mathbf{L}$ in such a way that if the error covariance is large, it weights the measurement more, and if the measurement noise covariance R is large, it weights the estimated state more. In other words, the filter trusts the output y more if the estimate $\hat{\mathbf{x}}$ is more likely to be wrong.

Next, we wish to find an a posteriori error covariance matrix $\mathbf{P} = E\left[\mathbf{e}\mathbf{e}^\top\right]$. By inserting $\hat{\mathbf{x}}$ from Equation (51), we obtain

$$\mathbf{P} = (\mathbf{I} - \mathbf{L}\mathbf{C}_d)\,\mathbf{P}^-\,(\mathbf{I} - \mathbf{L}\mathbf{C}_d)^\top + \mathbf{L}R\mathbf{L}^\top \tag{52}$$

The variance of the error $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ is found on the diagonal of the error matrix $\mathbf{P}$. Thus, minimizing the trace of $\mathbf{P}$ will minimize the variance of the error. In order to do this, we set the derivative of $\mathbf{P}$ with respect to $\mathbf{L}$ equal to zero, obtaining

$$\mathbf{L} = \mathbf{P}^-\mathbf{C}_d^\top\left(\mathbf{C}_d\mathbf{P}^-\mathbf{C}_d^\top + R\right)^{-1} \tag{53}$$

We obtain the a posteriori estimates of $\hat{\mathbf{x}}$ and $\mathbf{P}$ by inserting $\mathbf{L}$ as found in Equation (53) into Equation (51) and Equation (52), respectively.
We can now find a priori estimates $\hat{\mathbf{x}}^- [k+1]$ and $\mathbf{P}^- [k+1]$ for the next discrete time interval. We ignore the disturbance $\mathbf{w}$ when estimating $\hat{\mathbf{x}}^-$, because it has zero mean and is uncorrelated with all previous $\mathbf{w}$'s [1]. Thus, we obtain

$$\hat{\mathbf{x}}^- [k+1] = \mathbf{A}_d \hat{\mathbf{x}} [k] + \mathbf{B}_d \mathbf{u} [k] \tag{54}$$

The estimate of the error matrix $\mathbf{P}^- [k+1]$ is obtained by using the definition $\mathbf{P}^- [k+1] = E \left[\mathbf{e} [k+1]^- \mathbf{e} [k+1]^{-\top} \right]$. By inserting $\mathbf{e} [k+1]^- = \mathbf{A}_d \mathbf{e} [k+1] + \mathbf{E}_d \mathbf{w}$, we obtain the following results:

$$\mathbf{P}^- [k+1] = \mathbf{A}_d \mathbf{P} [k] \mathbf{A}_d^\top + \mathbf{E}_d \mathbf{Q} \mathbf{E}_d^\top \tag{55}$$

With these estimates, another iteration can be done. Thus, we now have a recursive algorithm for the discrete Kalman filter. We implemented this algorithm using MATLAB s-functions. The s-function is called with a Simulink block. MATLAB s-functions call different functions in different parts of the simulation. We implemented four of these functions: `mdlInitializeSizes` which initializes the block, `mdlUpdate` which updates the states, `mdlOutputs` which calculates the outputs from the block, and `mdlTerminate` which is called at the end of the simulation. The complete implementation can be found in Appendix B.3.

The initialization function, `mdlInitializeSizes`, configures the system to have 35 states. The five first entries in the state vector are the a priori estimates $\hat{\mathbf{x}}^-$. The five next entries are the a posteriori state estimates $\hat{\mathbf{x}}$. The 25 last states are the entries of the a priori error matrix $\mathbf{P}^-$. The function sets the initial values of $\hat{\mathbf{x}}_0^-$ and $\mathbf{P}_0^-$. The values are passed to the function through the struct `data`.

The update function, `mdlUpdate`, preforms the Kalman algorithm explained above. First, it converts $\mathbf{P}_0^-$ from the state vector into a 5×5 matrix. Next, the Kalman gain $\mathbf{L}$ is computed from eq. (53). The necessary system matrices are passed to the function through the `data` struct, which contains $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$, $\mathbf{E}_d$, R and $\mathbf{Q}$ as well as $\mathbf{P}_0^-$ and $\hat{\mathbf{x}}_0^-$ which are used by the initialization function. The function then updates the a posteriori state estimate and error matrix, from eq. (51) and eq. (52).

The output function, `mdlOutputs`, simply returns the desired outputs from the Kalman filter. In our case, the output should be the a posteriori estimates of the compass angle, ψ, and the bias to the rudder angle, b. The termination function only returns an empty matrix.

The s-function was included in the Simulink diagram using a s-function block. Inputs to the block is the rudder input angle and the course angle output from the Cargo ship block. A zero-hold block was used on the inputs to the s-function, because the inputs are continuous. The Simulink diagram can be found in Figure 21 in Appendix A.3, which is the diagram used in task 5e).

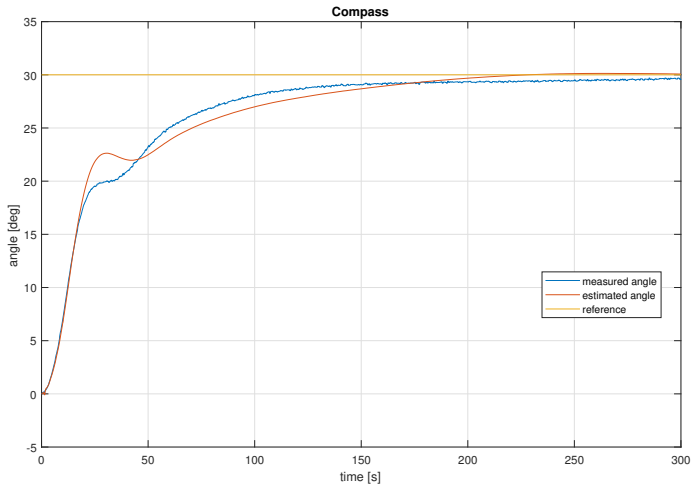

Figure 14: Ship course measurements and estimates.

5.4 Task d) Feed forward

With the Kalman filter implemented, we can connect it to the system. Our Kalman filter has two outputs. An estimated compass-heading, and an estimated rudder bias. In this part, we are going to use the rudder bias to counteract the stationary deviation generated by the water currents. We connect a feed forward from the estimated rudder bias to the rudder input, as shown in .

Add simulink

The estimated bias is added to the regulator's output, and together they remove the stationary deviation. As we can see by comparing Figure 14 and Figure 11, the ship now reaches the reference value instead of staying just below it. It should also be noted that when using a Kalman filter, the ship achieved the desired course with a current faster than the PD-regulator did in calm waters.

5.5 Task e) Simulation with wave and current

In the final task of this lab, we use the second Kalman filter output, the estimated compass heading. This is sent to the PD-controller, instead of the actual measurement.

Comparing the behaviour in fig. 16 with the response from fig. 12, we see that the biggest improvement is found in the rudder angle. In problem three, the rudder was fluctuating a lot. These high frequency changes will cause wear on the actuator, and the rudder might not be able to go fast enough to keep up. With the Kalman filter, the rudder is controlled a lot smoother. Besides some movement in the beginning, it stays almost stationary. This is a lot better for

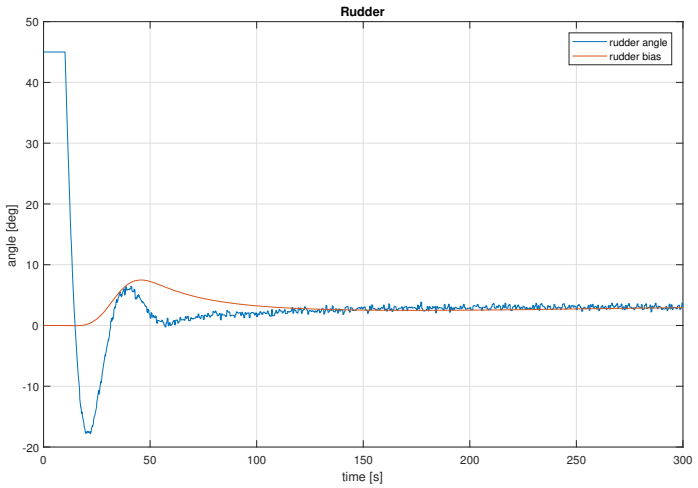

Figure 15: Ship rudder input and bias.

the actuator health.

The response in fig. 16 shows some influence from the waves. The ship reaches the reference heading, but swings around here. We saw in the previous parts that this isn't caused by the measurement noise or water currents. We can look into this by comparing the estimated wave influence to the actual wave influence. To get the actual wave influence, we set the rudder input to zero, and turn of measurement noise and currents. Estimated wave influence can be received from the Kalman filter. Wave influence is the second state, and by modifying our S-function slightly we can access this data.

Using the data for wave influence, we can compare them. This can be seen in Figure 18. The plot shows a deviation between the estimated and the actual wave influence. This means that the Kalman filter struggles to precisely estimate the wave behaviour. An estimate that is to small means that our ship won't be able to cope with the waves perfectly, as we saw from fig. 16.

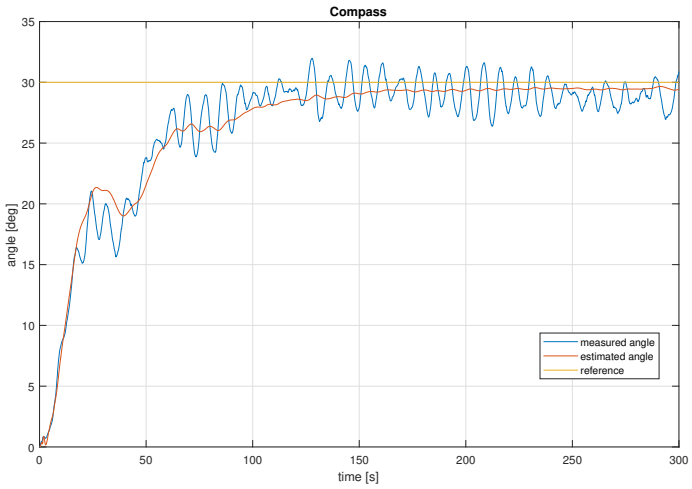

Figure 16: Ship course with measurement noise, waves and current using a Kalman filter.

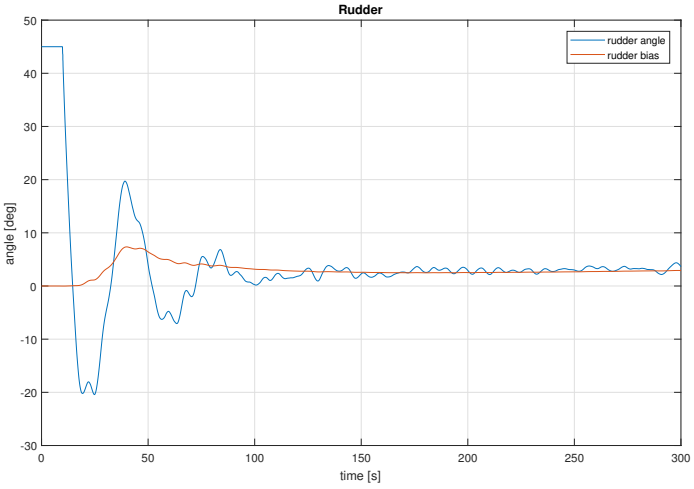

Figure 17: Rudder input and bias with measurement noise, waves and current.

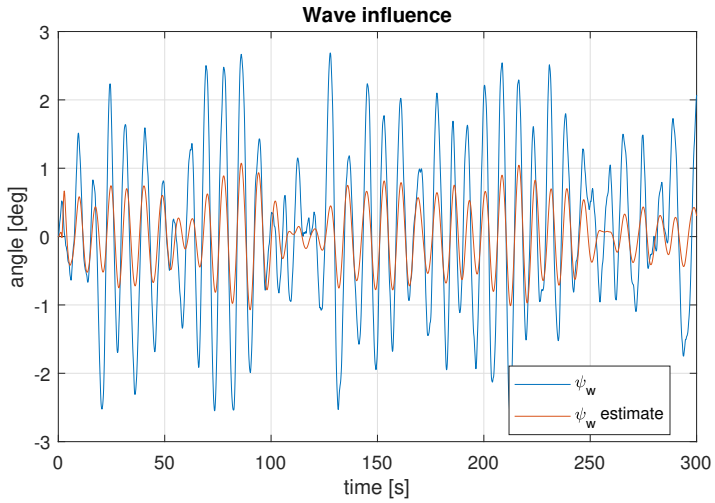

Figure 18: Actual and estimated wave influence

6 Conclusion

During this lab assignment we have looked at ship behaviour in calm and rough waters. A simple PD-controller can steer the ship to the wanted heading in calm waters. Once we add currents, waves and measurement noise, it struggles.

Water currents caused a stationary deviation, while waves caused high frequency fluctuations in the rudder, which might damage our actuator. We implemented a Kalman filter to counteract these effects.

Our Kalman filter has two outputs. The rudder bias is used to counteract the stationary deviation. The estimated heading is sent to our controller, and this filtered feedback smoothed the rudder control and got rid of the measurement noise.

A Simulink models

A.1 Task 1

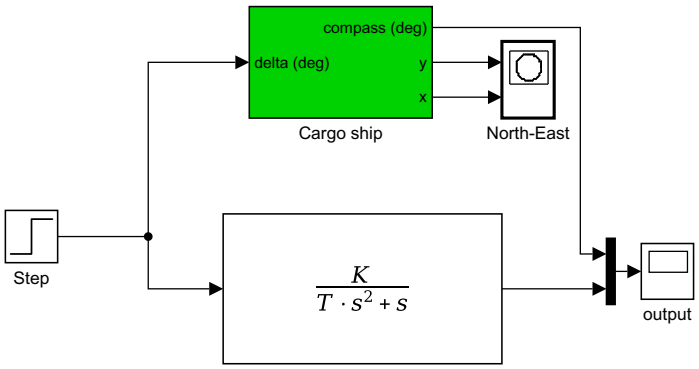

Figure 19: Simulink diagram for ship response compared to model, for task 1.4.

A.2 Task 3

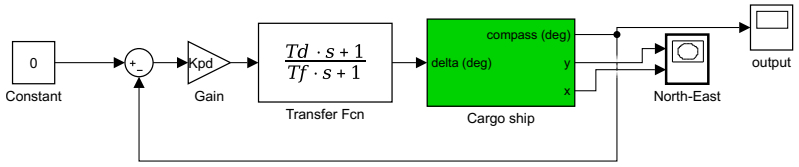

Figure 20: Simulink diagram with PD controller, for task 3.3.

A.3 Task 5

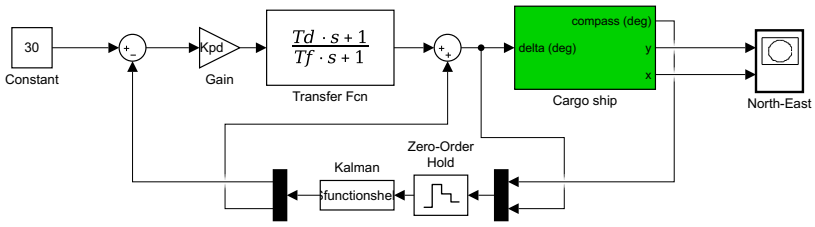

Figure 21: Simulink diagram with Kalman filter, for task 5.5.

B MATLAB implementations

B.1 Task 1.2 and 1.3

```
clc; clear all; close all;

syms T K;
omega1 = 0.005;
omega2 = 0.05;
ans1 = 29.35;
ans2 = 0.82;
eqns = [K ./(omega1 * sqrt(1 + T^2 * omega1^2)) == ans1,...
        K ./(omega2 * sqrt(1 + T^2 * omega2^2)) == ans2];
vars = [T K];
%Solve for T and K
[solT, solK] = solve(eqns, vars);

% We want the positive solutions
solT = abs(solT(1));
solK = abs(solK(1));
fprintf("T = %.5f\n", solT);
fprintf("K = %.5f\n", solK);
```

B.2 Task 2.1 and 2.3

```
clear all;
close all;

load('wave.mat')
[Sx, f] = pwelch(deg2rad(psi_w(2,:)), 4096, [], [], 10);

Sx = Sx./(2*pi);
omega = f .* (2*pi);

figure;
plot(omega, Sx)
title('PSD');
legend('Sx');
xlabel('w [rad]');
xlim([0, pi/2])

index = find(Sx == max(Sx))
w_0 = omega(index)
hold on;
stem(w_0, max(Sx), '')
ylim([0, 1.5*10^(-3)])
hold off;
sigma = sqrt(max(Sx));
fun = @(l, omega)...
    (( omega.^2*(2*l*w_0*sigma).^2)./((w_0^2 - omega.^2).^2 ...
    + (2*l*w_0.*omega).^2));

l_0 = [0.08];

l = lsqcurvefit(fun, l_0, omega, Sx);
```

B.3 Task 5.3

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

switch flag,

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

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

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

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

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

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

end

function [sys,x0,str,ts]=mdlInitializeSizes(data)
% This is called only at the start of the simulation.
```

```
45
46  sizes = simsizes; % do not modify
47
48  sizes.NumContStates  = 0; % Number of continuous states in the system
49  sizes.NumDiscStates  = 35; % Number of discrete states in the system
50  sizes.NumOutputs     = 2; % Number of outputs
51  sizes.NumInputs      = 2; % Number of inputs
52  sizes.DirFeedthrough = 0; % 1 if the input is needed directly in the
53  % update part
54  sizes.NumSampleTimes = 1;
55
56  sys = simsizes(sizes);
57
58  x0(1:5)  = data.x0; % Initial values for the discrete states
59  x0(6:10) = data.x0;
60  x0(11:35) = data.P_(:)';
61  str = [];
62
63  ts  = [-1 0]; % Sample time. [-1 0] means that sampling is
64  % inherited from the driving block and that it changes during
65  % minor steps.
66
67
68  function x=mdlUpdate(t,x,u,data),
69  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
70  % Update the filter covariance matrix and state etsimates here.
71  % example: sys=x+u(1), means that the state vector after
72  % the update equals the previous state vector + input nr one.
73  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74
75  % u(1) is compass angle
76  % u(2) is rudder input
77  % x(1:5): a priori state estimate
78  % x(6:10): a posteriori state estimate
79  % x(11:35): a priori P-
80
81  P_ = reshape(x(11:35), 5, 5); % P_ is now 5x5 matrix
82
83  % Compute Kalman gain
84  L = P_ * (data.Cd)' * (data.Cd*P_*(data.Cd)' + data.R)^-1;
85
86  % Update estimate with measurement
87  x(6:10) = x(1:5) + L*(u(1) - data.Cd*(x(1:5)));
88  % Update P
89  P = (eye(5) - L*data.Cd)*P_ *(eye(5) - L*data.Cd)' + L*data.R*L';
90
```

```
% A priori estimates, for next iteration
x(1:5) = data.Ad * x(6:10) + data.Bd*u(2);
P_ = data.Ad*P*(data.Ad)' + data.Ed*data.Q*(data.Ed)';
x(11:35) = P_(:);

function sys=mdlOutputs(t,x,u, data)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the outputs here
% example: sys=x(1)+u(2), means that the output is the first state+
% the second input.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% sys(1): compass angle
% sys(2): bias to the rudder angle

sys(1) = x(8);
sys(2) = x(10);

function sys=mdlTerminate(t,x,u)
sys = [];
```

References

[1] Robert Grover Brown and Patrick Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. Wiley, 2012.

[2] Department of Engineering Cybernetics. "Discrete Kalman Filter Applied to a Ship Autopilot". 2016.

[3] Wikipedia. *Observability — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2017]. 2017. URL: `https://en.wikipedia.org/wiki/Observability`.