NTNU
Department of Engineering Cybernetics

TTK4115 LINEAR SYSTEM THEORY

Discrete Kalman Filter Applied to a Ship's Autopilot

Report

Group no. 19

November 25, 2011

By

Kenan Trnka, student no. 716377
and
Torjus Sveier Ottemo, student no. 716345

Abstract

This report deals with the construction of a ship autopilot system. The purpose of the control system is to be able to set a reference compass angle, and have the ship follow that course in spite of measurement noise or disturbances like current and waves. The controller used to control the compass course is a PD-controller. Unfortunately this controller is not able to cope well with the disturbances applied to the system. We therefore implement a Kalman filter which gives a much smoother feedback for the reference since the measurement is littered with noise and wave disturbance, and we also compensate for the current by using a feed forward from the Kalman filter. Throughout the report we calculate and find different parameters and matrices needed for the final result, as well as implementing both controller and Kalman filter. Extensive simulation and function plotting is done in order to display the properties of the system.

Preface

This report is the result of the boat lab we had in the fall of 2011 in the course TTK4115 Linear System Theory. This assignment has been one of the more interesting ones we have had yet, mainly because of the sometimes steep learning curve and the soothing feeling when hard work pays off in the form of a functional system. The theory, knowledge and experience aquired in this lab is without doubt very valuable to use in the future with other applications, not necessarily just control systems for vehicles, but a wide variety of control problems.

As always we have tried to keep the report brief. This is after all a report, not a dissertation, and hence much of the theoretical background given in the assignment will not be repeated here. We have rather focused on the results and approaches used.

Contents

Chapter 1

Introduction

A brief introduction to the system is needed. The ship has it's own frame of reference, BODY, which is stationary from the ships point of view, with the x-axis along the longitudinal axis and the y-axis along the transverse axis. We also have another frame of reference, NED, which is the North-East coordinate system, with the x-axis facing north and the y-axis facing east. The ships rudder angle is relative to the BODY frame. It is desirable to make the ship travel in a certain course in the NED frame, while controlling the ships rudder relative to the BODY frame. Therefore, some conversion needs to be done between the two frames of reference. By doing some simplifications in the way the ship is allowed to move (only in the yaw and sway degree of freedom), modelling for low speeds to remove non-linear effects and also take disturbances into account, we end up with the following differential equations describing the system:

$$\begin{aligned}
\dot{\xi} &= \psi_w \\
\dot{\psi}_w &= -\omega_0^2 \xi_w - 2\lambda\omega_0\psi_w + K_w w_w \\
\dot{\psi} &= r \\
\dot{r} &= -\tfrac{1}{T} r + \tfrac{K}{T}(\delta - b) \\
b &= w_b \\
y &= \psi + \psi_w + v
\end{aligned} \tag{1.1}$$

This can be represented in a state space equation like this:

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u + \mathbf{E}\mathbf{w} \\
y &= \mathbf{C}\mathbf{x} + v
\end{aligned} \tag{1.2}$$

This is the modeled system, and will be used in the rest of the report.

Chapter 2

Identification of the boat parameters

2.1 Finding the transfer function $\frac{\psi}{\delta}(s)$

We start by finding the transfer function from ψ to λ. From the preliminary, we have the equation

$$\dot{r} = -\frac{1}{T}r + \frac{K}{T}(\delta - b) \tag{2.1}$$

Since $r = \dot{\psi}$ and we assume no disturbances, we set $b = 0$ and replace r in the equation and then Laplace transform it:

$$\ddot{\psi} = -\frac{1}{T}\dot{\psi} + \frac{K}{T}\delta \Rightarrow s^2\psi = -\frac{1}{T}\psi s + \frac{K}{T}\delta \tag{2.2}$$

We then rearrange the equation, yielding the transfer function

$$\psi(s^2 + \frac{1}{T}s) = \frac{K}{T}\delta \Rightarrow \frac{\psi}{\delta}(s) = \frac{K}{(s^2 + \frac{1}{T}s)T} = \frac{K}{Ts^2 + s} \tag{2.3}$$

$$\Rightarrow \frac{\psi}{\delta}(s) = \frac{K}{Ts^2 + s} \tag{2.4}$$

2.2 Finding T and K

The parameters T and K can be found by applying different frequencies to the system, and measuring the amplitude of the output. Having previously found the transfer function, we can find its absolute value, substituting $s = j\omega$:

$$|H(j\omega)| = \frac{\sqrt{K^2}}{\sqrt{\left(T(j\omega)^2\right)^2 + (j\omega)^2}} = \frac{K}{\sqrt{(-T\omega^2)^2 + (j\omega)^2}} = \frac{K}{\sqrt{T^2\omega^4 + \omega^2}} \tag{2.5}$$

The absolute value when a certain frequency is applied is some number A. We insert the frequency ω_1 and set $|H(j\omega_1)|$ equal to its amplitude A_1:

$$|H(j\omega_1)| = \frac{K}{\sqrt{T^2{\omega_1}^4 + {\omega_1}^2}} = A_1 \Rightarrow K = A_1\sqrt{T^2{\omega_1}^4 + {\omega_1}^2} \tag{2.6}$$

We then do the same thing for ω_1, but this time we replace K with the expression above:

$$|H(j\omega_2)| = \frac{A_1\sqrt{T^2{\omega_1}^4 + {\omega_1}^2}}{\sqrt{T^2{\omega_2}^4 + {\omega_2}^2}} = A_2 \Rightarrow A_1\sqrt{T^2{\omega_1}^4 + {\omega_1}^2} = A_2\sqrt{T^2{\omega_2}^4 + {\omega_2}^2} \tag{2.7}$$

Now it's just a simple task solving for T

$$A_1{}^2(T^2\omega_1{}^4 + \omega_1{}^2) = A_2{}^2(T^2\omega_2{}^4 + \omega_2{}^2) \tag{2.8}$$

$$T^2(A_1{}^2\omega_1{}^4 - A_2{}^2\omega_2{}^4) = A_2{}^2\omega_2{}^2 - A_1{}^2\omega_1{}^2 \tag{2.9}$$

$$\Rightarrow T = \pm\sqrt{\frac{A_2{}^2\omega_2{}^2 - A_1{}^2\omega_2{}^1}{A_1{}^2\omega_1{}^4 - A_2{}^2\omega_2{}^4}} \tag{2.10}$$

And then inserting T in the expression for K

$$\Rightarrow K = A_1\sqrt{\frac{A_2{}^2\omega_2{}^2 - A_1{}^2\omega_1{}^2}{A_1{}^2\omega_1{}^4 - A_2{}^2\omega_2{}^4}\omega_1{}^4 + \omega_1{}^2} \tag{2.11}$$

In the Simulink model we now add a sine wave block to the rudder input and simulate with the two different frequencies, $\omega_1 = 0.005$ and $\omega_2 = 0.05$ and plot the response:

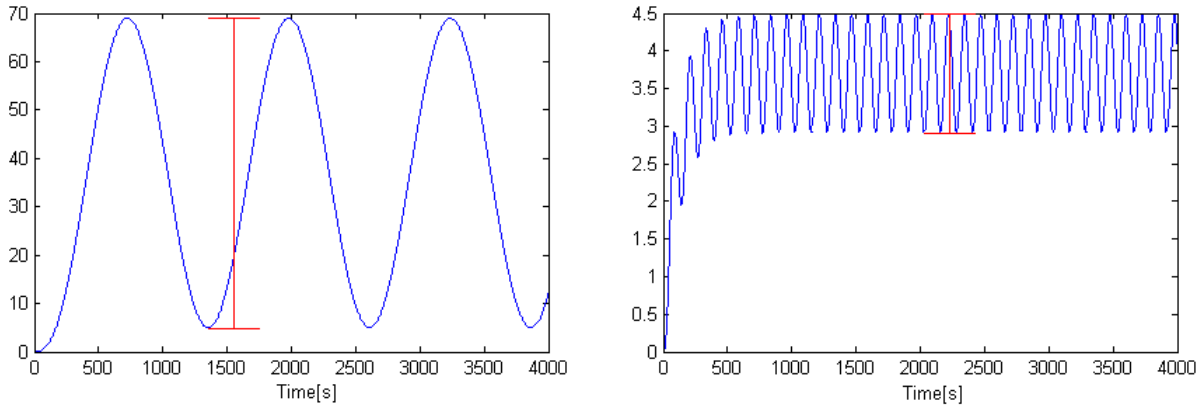

Figure 2.1: To the left $\omega_1 = 0.005$ and to the right $\omega_2 = 0.05$

From the plot we can read the following amplitudes

Frequency	Amplitude
$\omega_1 = 0.005$	$A_1 = 31.95$
$\omega_2 = 0.05$	$A_2 = 0.7846$

And we finally find the answer for K and T by substituting the numerical values of the amplitudes and frequencies into the equations 2.10 and 2.11:

$$T = 86.4404 \tag{2.12}$$

$$K = 0.1740 \tag{2.13}$$

2.3 Testing with noise

With measurement noise turned on, the output signal is still very readable. We can read the amplitude with relative ease, making the calculations of T and K an easy task now that the formulas are developed.

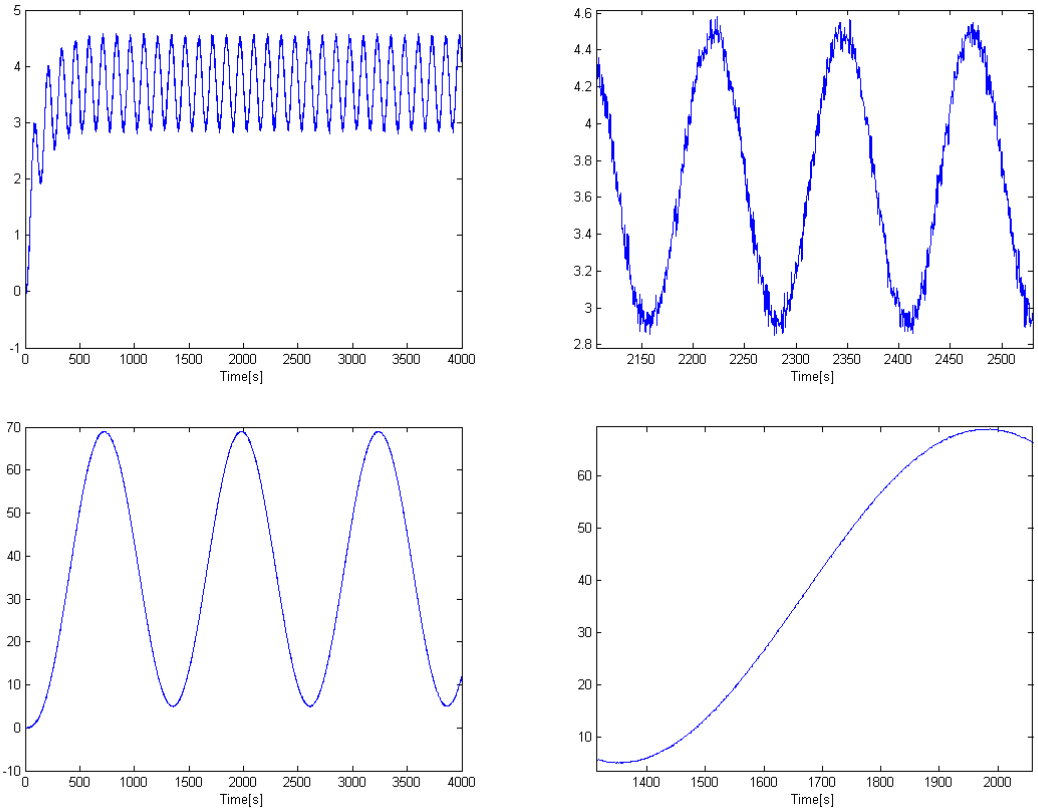

Figure 2.2: Plots with measurement noise for both frequencies to the left and zoomed in versions to the right.

We can see that the signal is more jagged than before, but still very readable, so the effect of the disturbance is basically negligible when reading the amplitude.

2.4 Step response

Testing whether or not our model fits well is important. In order to do that, we can apply a step input to both our model and the ship and see how they compare with eachother. The model response should obviously be identical in the ideal case, however there have been made a couple of approximations and simplifications, so the two responses will not be the same. By adding a step response to the ships input in Simulink and calculate the step response of the transfer function $\frac{\psi}{\delta}(s)$, we can compare the two. The plot can be seen in appendix A.1. There are obviously some differences in the responses as time increases, but it's a pretty good approximation, and with a decent controller it should not present a problem.

Chapter 3

Identification of wave spectrum model

3.1 Finding S_{ψ_w} using `pwelch()`

We now start analyzing the wave disturbance. To do this, we load the file containing the wave information, called `wave.mat`. We then use MATLAB and Welch's method in order to find the frequenzy components contained in the wave information. We use a sample frequency of 10 Hz and a window size of 4096. After the Welch computation we multiply with 2π since `pwelch()` scales everything down with 2π.

```
load wave;

Fs = 10;
window = 4096;

[Pxx, w] = pwelch(psi_w(2,:)*pi/180,window,[],[],10);

plot(w*2*pi,Pxx*2*pi);
xlim([0 2]);
```

The resulting plot is found in A.2

3.2 Finding P_{ψ_w} and $\frac{\psi}{w}(s)$

The analytical expression for P_{ψ_w} is defined as[1]:

$$P_{\psi_w} = |H(j\omega)|^2 P_{w_w}(j\omega) \tag{3.1}$$

In other words, to find the PSD function, we need to know the transfer function from ψ to w and the PSD function for the white noise. We start by Laplace transforming the differential equation for ψ, where $\dot{\xi} = \psi$:

$$\dot{\psi} = -{\omega_0}^2\xi - 2\lambda\omega_0\psi + K_w w \tag{3.2}$$

$$s\psi = -{\omega_0}^2\frac{\psi}{s} - 2\lambda\omega_0\psi + K_w w \tag{3.3}$$

$$\psi(s + \frac{{\omega_0}^2}{s} + 2\lambda\omega_0) = K_w w \tag{3.4}$$

[1]Brown p. 130

$$\frac{\psi}{w}(s) = \frac{Kw}{(s + \frac{{\omega_0}^2}{s} + 2\lambda\omega_0)} = \frac{sKw}{(s^2 + 2\lambda\omega_0 s + {\omega_0}^2)} \tag{3.5}$$

We must now calculate the squared absolute value of the transfer function, and substitute $s = j\omega$. This can be done by multiplying the transferfunction with its complex conjugate:

$$|H(j\omega)|^2 = H(j\omega)H(j\omega)^* = \frac{j\omega K_w \cdot -j\omega K_w}{(-\omega^2 + j2\lambda\omega_0\omega + {\omega_0}^2)(-\omega^2 - j2\lambda\omega_0\omega + {\omega_0}^2)} = \tag{3.6}$$

$$\frac{\omega^2 K_w^2}{(\omega^4 - \omega^2{\omega_0}^2 + \cancel{j2\lambda\omega_0\omega^3} - \cancel{j2\lambda\omega_0\omega^3} + \cancel{j2\lambda\omega_0{}^3\omega} - \cancel{j2\lambda\omega_0{}^3\omega} + 2^2\lambda^2{\omega_0}^2\omega^2 - \omega^2{\omega_0}^2 - {\omega_0}^4)} \tag{3.7}$$

$$\frac{\omega^2 K_w^2}{(\omega^4 - 2\omega^2{\omega_0}^2 + 4\lambda^2{\omega_0}^2\omega^2 + {\omega_0}^4)} = \frac{\omega^2 K_w^2}{(\omega^4 + (4\lambda^2 - 2){\omega_0}^2\omega^2 + {\omega_0}^4)} \tag{3.8}$$

The constant K_w is defined as $2\lambda\omega_0\sigma$. This finally yields the expression for P_{ψ_w}. Since w is unity white noise, $P_{w_w} = 1$, and we end up with

$$P_{\psi_w} = \frac{4\lambda^2{\omega_0}^2\sigma^2\omega^2}{(\omega^4 + (4\lambda^2 - 2){\omega_0}^2\omega^2 + {\omega_0}^4)} \tag{3.9}$$

3.3 Finding ω_0 and σ

Now that we have calculated S_{ψ_w} in MATLAB, it is straight forward to extract both ω_0 and σ by using this MATLAB script:

```
[maxValue, i] = max(Pxx)
freqMax = w(i)
w0 = freqMax*2*pi
sigma = sqrt(maxValue*2*pi)
```

The values we find are $\omega_0 = 0.7823$ and $\sigma = 0.4841$.

3.4 Identifying the damping factor λ

By inserting the values for ω_0 and σ in P_{ψ_w} while adjusting λ, we found 0.09 to be a fitting value when compared to S_{ψ_w}. The resulting plot can be seen in the appendix. The analytical PSD function has some differences compared to the PSD function found with `pwelch()`. This is expected though, since the analytical PSD function is based on some approximations and simplifications.

Chapter 4

Autopilot design

4.1 Designing the controller

We are going to design a PD controller given by the equation:

$$H_{pd}(s) = K_{pd}\frac{1+T_d s}{1+T_f s} \tag{4.1}$$

Our system is given by the transfer function from δ to ψ, which we have previously shown to be:

$$H_{ship}(s) = \frac{\delta}{\psi}(s) = \frac{K}{s(1+Ts)} \tag{4.2}$$

and, when given $T_d = T$ to cancel out the transfer function time constant, we get:

$$H_0(s) = H_{pd}(s) \cdot H_{ship}(s) = \frac{K_{pd}K}{s(1+T_f s)} \tag{4.3}$$

To adapt the PD controller, we must first calculate the absolute value and angle of the system:

$$|H_0(j\omega)| = \frac{K_{pd}K}{|j\omega(1+jT_f\omega)|} \tag{4.4}$$

$$|H_0(j\omega)| = \frac{K_{pd}K}{|-T_f\omega^2 + j\omega|} \tag{4.5}$$

$$|H_0(j\omega)| = \frac{K_{pd}K}{\sqrt{T_f^2\omega^4 + \omega^2}} \tag{4.6}$$

We then calculate the angle of the system, by first expanding the system fraction:

$$H_0(j\omega) = \frac{K_{pd}K}{-T_f\omega^2 + j\omega} \cdot \frac{-T_f\omega^2 - j\omega}{-T_f\omega^2 - j\omega} \tag{4.7}$$

$$H_0(j\omega) = \frac{K_{pd}K(-T_f\omega^2 - j\omega)}{T_f^2\omega^4 + \omega^2} = -\frac{K_{pd}KT_f\omega^2}{T_f^2\omega^4 + \omega^2} - j\frac{K_{pd}K\omega}{T_f^2\omega^4 + \omega^2} \tag{4.8}$$

The phase angle is then given by (where num is the numerator of the system):

$$\angle H_0(j\omega) = -\arctan\left(\frac{\Im(num)}{\Re(num)}\right) \tag{4.9}$$

which gives us

$$\angle H_0(j\omega) = -\arctan\left(\frac{-K_{pd}K\omega}{-K_{pd}KT_f\omega^2}\right) = -\arctan\left(\frac{1}{T_f\omega}\right) \tag{4.10}$$

Adapting the PD controller to the system by setting $\omega_c = 0.1$ and the phase margin $\Phi = 50°$:

$$180° - \Phi = -\arctan\left(\frac{1}{T_f\omega_c}\right) \tag{4.11}$$

$$\Rightarrow T_f = -\frac{1}{\omega_c \tan(180° - \Phi)} \tag{4.12}$$

which gives the value $T_f = 8.391$. Moving on to K_{pd}, setting $\omega_c = 0.1$ in the absolute value equation:

$$|H_0(j\omega_c)| = \frac{K_{pd}K}{\sqrt{T_f^2\omega_c^4 + \omega_c^2}} = 1 \tag{4.13}$$

$$\Rightarrow K_{pd} = \frac{\sqrt{T_f^2\omega_c^4 + \omega_c^2}}{K} \tag{4.14}$$

which gives the value $K_{pd} = 0.7502$. The margin plot based on equation (4.3) is found in appendix A.4 and was calculated using the following script:

```
K   = 0.1740
Tf  = 8.391
Kpd = sqrt(Tf^2*omega_c^4+omega_c^2)/K

a = [Kpd*K]
b = [Tf 1 0]
H = tf(a,b)
margin(H)
```

To see the implementation of the controller in Simulink, see appendix figure D.2.

4.2 Simulation with measurement noise

In figure 4.1 we simulate with only measurement noise.

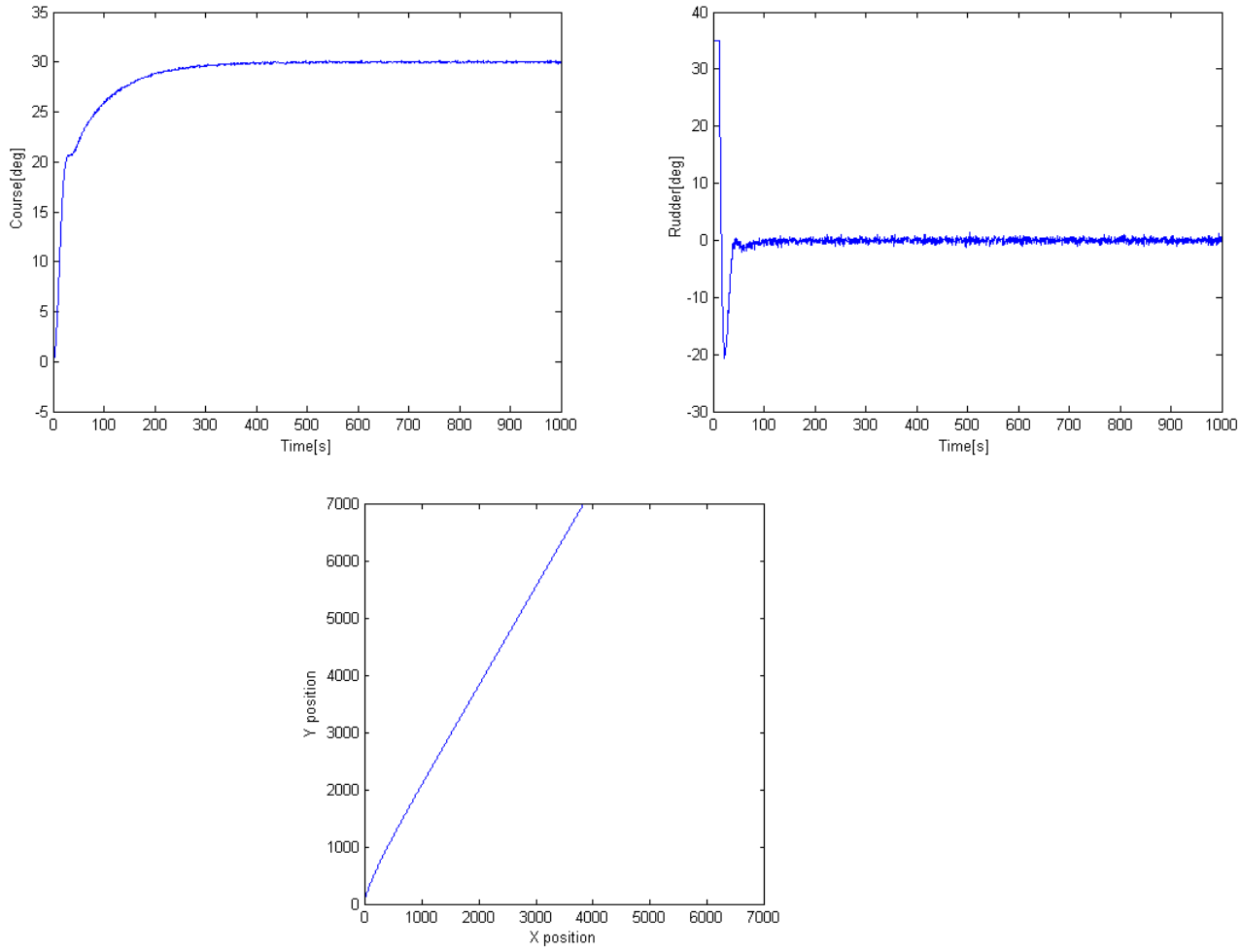

Figure 4.1: Simulation with only measurement noise. Top left is compass course, top right is rudder input and bottom center is North-East position.

Here we see the controller stabilizing the compass course at around 30 degrees, which is expected. We can also see from the rudder input that it stabilizes around 0. Unfortunately there is significant noise in the rudder input as a consequence of the measurement noise. This noise poses a problem, since it can cause much wear on the rudder's actuator. This should definitely be adressed in the final solution.

4.3 Simulation with current disturbance and measurement noise

In figure 4.2 we simulate with current disturbance as well as measurement noise.

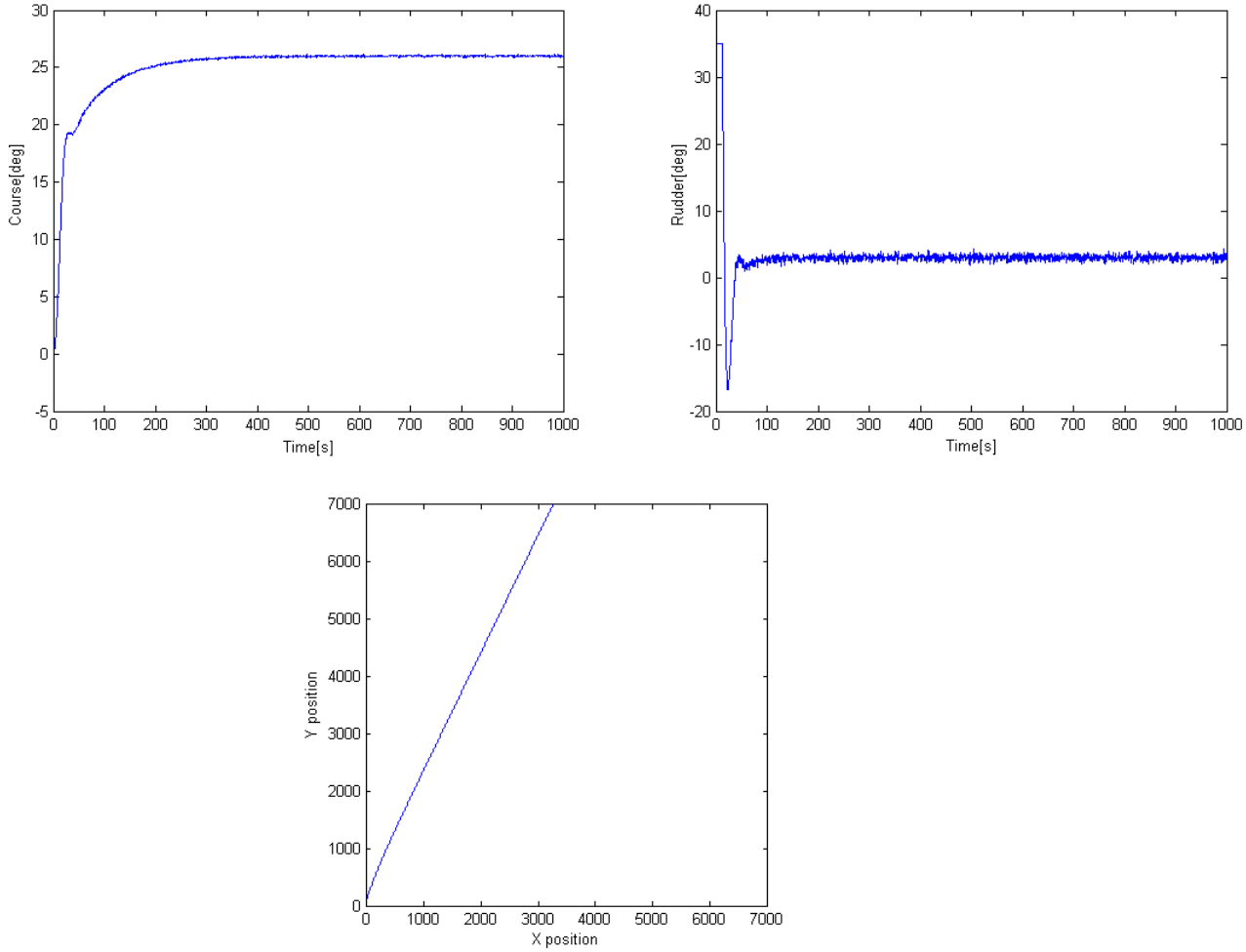

Figure 4.2: Simulation with current disturbance and measurement noise. Top left is compass course, top right is rudder input and bottom center is North-East position.

We see right away that there are serious problems with the compass course. It has stabilized around 25 degrees in spite of the reference being 30 degrees. This is obviously because of the current's modeled behaviour as a influence on the rudder angle. In the plot for the North-East position we see that the angle between the Y-axis and the position graph is narrower than before, about 25 degrees as expected.

4.4 Simulation with wave disturbance and measurement noise

In figure 4.3 we simulate with wave disturbance as well as measurement noise.

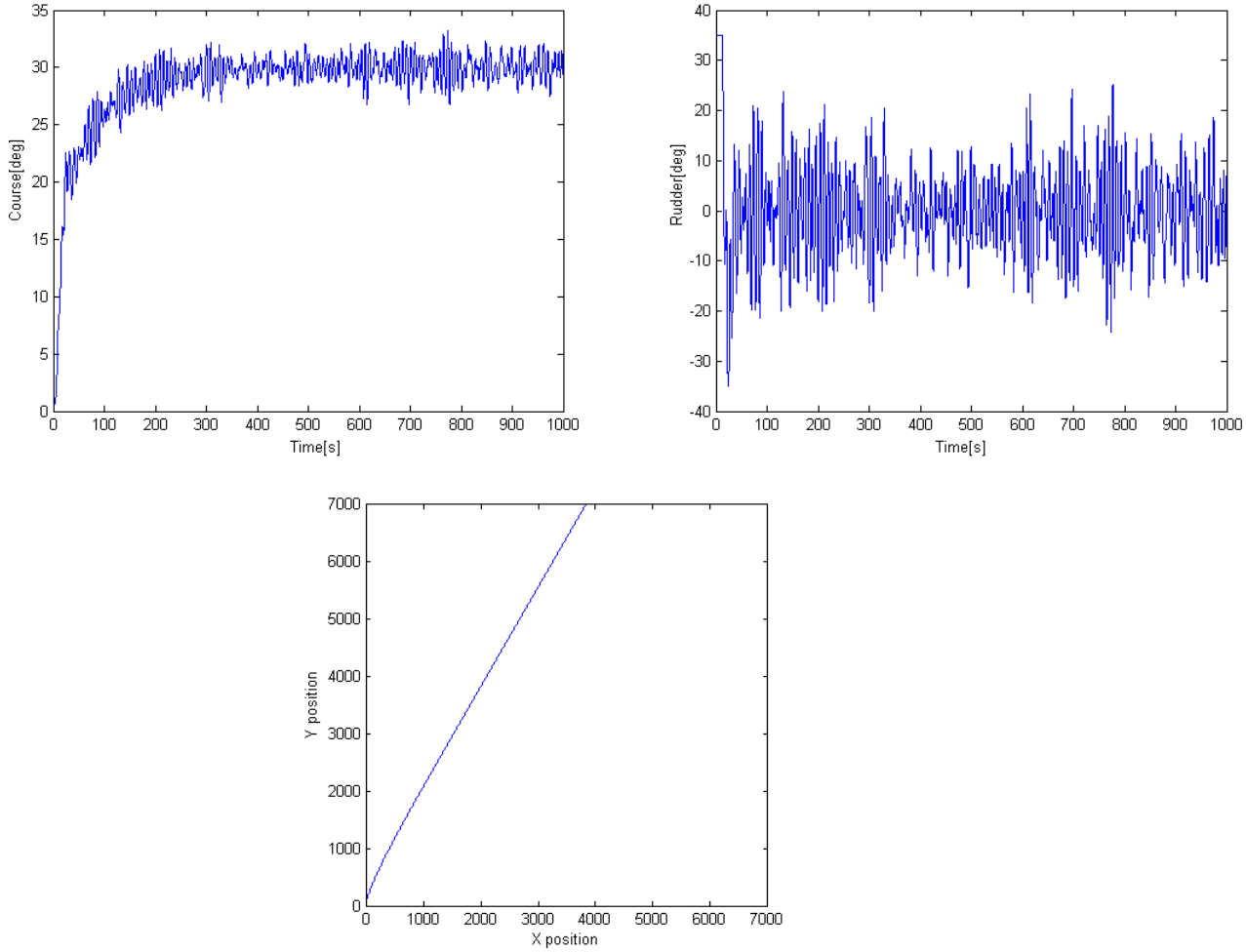

Figure 4.3: Simulation with wave disturbance and measurement noise. Top left is compass course, top right is rudder input and bottom center is North-East position.

In this last case, we have have only wave disturbance enabled. We see right away that the compass course and rudder input is extremely noisy. This would cause immense stress on the rudders actuator and shorten its life dramatically. This is a problem that has to be adressed in the final control system. However, the ship has a nice, straight path in the North-East plot, although it is somewhat wiggly.

Chapter 5

Observability

5.1 The matrices of the system

Based on the differential equations we can make the following matrices to describe our system:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_0 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix} \tag{5.2}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{5.3}$$

$$\mathbf{E} = \begin{bmatrix} 0 & 0 \\ K_w & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.4}$$

5.2 Calculation of observability

We are interested to find out if the system is observable under different cases. The cases we wish to investigate are:

- without any disturbances
- with only current disturbance
- with only wave disturbance
- with both current and wave disturbance

All the calculations were done generally by using this code

```
O = obsv(A,C);
us = length(A) - rank(O)
```

The **A** and **C** matrices are changed for each of the cases. If the variable `us` (unobservable states) evaluates to 0, we know that the system is observable.

5.2.1 Without any disturbances

When we don't have any disturbances, we are left with only two differential equations, so A and C reduce to

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.5}$$

When calculated, we find that $rank(\mathcal{O}) = dim(A)$, so the system is observable.

5.2.2 With current disturbance

We now take into account the current disturbance. This means we have one more differential equation to include compared to the previous case. This yields

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{5.6}$$

We then proceed with the observability calculation, and we quickly see that $rank(\mathcal{O}) = dim(A)$, so the system is observable also in this case.

5.2.3 With wave disturbance

This step is equal to the previous, but with a different disturbance, namely the wave disturbance.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_0 & -2\lambda\omega_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \tag{5.7}$$

And after the observability calculation we once again see that $rank(\mathcal{O}) = dim(A)$ so the system is observable.

5.2.4 With both current and wave disturbance

For this last observability calculation, we leave (4.1) and (4.3) unaltered. Yet again the system is indeed observable. The importance of observability must not be ignored when dealing with the Kalman filter. Only an observable system can be realized through such a filter.

Chapter 6

Kalman filter design

6.1 Discretization of the system model

The discretization of the model is given by the following equations:

$$\mathbf{A_d} = \mathbf{e^{AT}} = \mathscr{L}^{-1}\{(\mathbf{sI} - \mathbf{A})^{-1}\}_{\mathbf{t=T}} \tag{6.1}$$

$$\mathbf{B_d} = \left(\int_{\tau=\mathbf{0}}^{\mathbf{T}} \mathbf{e^{A\tau}} \mathbf{d}\tau\right) \mathbf{B} \tag{6.2}$$

$$\mathbf{C_d} = \mathbf{C} \tag{6.3}$$

$$\mathbf{E_d} = \left(\int_{\tau=\mathbf{0}}^{\mathbf{T}} \mathbf{e^{A\tau}} \mathbf{d}\tau\right) \mathbf{E} \tag{6.4}$$

Where $T = 1/f_s$, and our sampling frequency $f_s = 10$Hz. Manual discretization is very time consuming, so we make use of the MATLAB function `c2d`:

```
%A,B,E already loaded into workspace
[ad, bd] = c2d(A, B, 1/10)
[ad, ed] = c2d(A, E, 1/10)
```

which gives us

$$\mathbf{A_d} = \begin{bmatrix} 0.9971 & 0.0932 & 0 & 0 & 0 \\ -0.0570 & 0.8659 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0999 & 0 \\ 0 & 0 & 0 & 0.9988 & -0.0002 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

$$\mathbf{B_d} = \begin{bmatrix} 0 \\ 0 \\ 1.0065 \cdot 10^{-5} \\ 2.0126 \cdot 10^{-4} \\ 0 \end{bmatrix} \tag{6.6}$$

$$\mathbf{C_d} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{6.7}$$

$$\mathbf{E_d} = \begin{bmatrix} 0.0033 & 0 \\ 0.0635 & 0 \\ 0 & 3.355 \cdot 10^{-7} \\ 0 & 1.0065 \cdot 10^{-5} \\ 0 & 0.1 \end{bmatrix} \tag{6.8}$$

6.2 Estimate of measurement noise variance

We estimate the variance of the measurement noise by running the system stationary for a period of time, record the measurement noise and use the MATLAB function `var` to calculate the variance. Since this value is based on degrees, we need to convert it to radians. It's important to remember when dealing with variances that when you convert the variance from one random variable X to another $Y = aX$, you have to square a, i.e. $Var(Y) = Var(aX) = a^2 Var(X)$. In our case, we have found the variance for the variable v, so since $r = \frac{\pi}{180} \cdot v$ and $Var(r) = R$ we have:

$$R = \left(\frac{\pi}{180}\right)^2 \cdot \mathrm{Var}\,(v) = 6.2428 \cdot 10^{-7} \tag{6.9}$$

6.3 Kalman filter implementation

The next step is to implement a Kalman filter. There are several ways to implement a Kalman filter in Simulink. By using the S-function block in Simulink, we can implement functions in several different languages, like Matlab, C and Fortran. We settled on using the MATLAB scripted framework given to us in the assignment. There are several reason for this, for one we realized a filter implemented with C would be a much more complex implementation and be very time consuming to implement. Secondly, by using the framework, the student assistants could give a helping hand since they were familiar with it. And finally, we have used Matlab in every aspect of this lab and therefore it felt natural to continue using this language.

To implement the filter we first have to take a look at the filter loop as seen in figure 6.1:

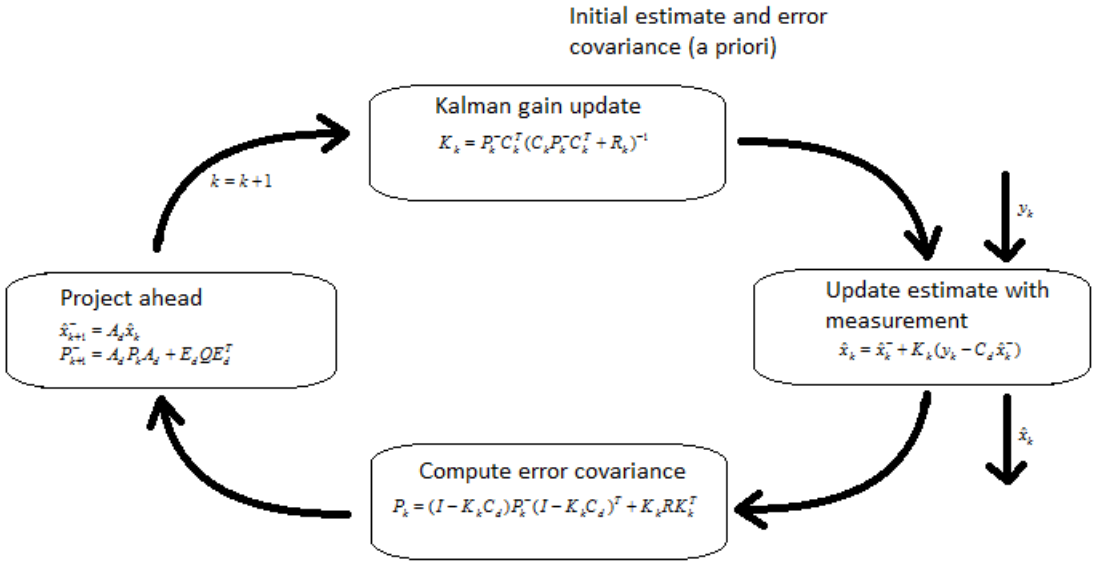

Figure 6.1: The Kalman filter loop

The expressions used in figure 6.1 are, for the sake of clarity:

- $\mathbf{K_k} = \mathbf{P_k}^{-}\mathbf{C_d}^{\mathbf{T}}(\mathbf{C_d}\mathbf{P_k^-}\mathbf{C_d}^{\mathbf{T}} + \mathbf{R_k})^{-\mathbf{1}}$
- $\hat{\mathbf{x}}_\mathbf{k} = \hat{\mathbf{x}}_\mathbf{k}^- + \mathbf{K_k}(\mathbf{y_k} - \mathbf{C_d}\hat{\mathbf{x}}_\mathbf{k}^-)$
- $\mathbf{P_k} = (\mathbf{I} - \mathbf{K_k}\mathbf{C_d})\mathbf{P_k^-}(\mathbf{I} - \mathbf{K_k}\mathbf{C_d})^{\mathbf{T}} + \mathbf{K_k}\mathbf{R}\mathbf{K_k^T}$
- $\mathbf{P_{k+1}^-} = \mathbf{A_d}\mathbf{P_k}\mathbf{A_d} + \mathbf{E_d}\mathbf{Q}\mathbf{E_d^T}$
- $\hat{\mathbf{x}}_{\mathbf{k+1}}^- = \mathbf{A_d}\hat{\mathbf{x}}_\mathbf{k}$

We see that there are a couple differences in this loop compared to the one in the book[1], besides matrix naming differences. Our matrices are assumed to be time invariant and therefore constant. Only the $\mathbf{P_k}/\mathbf{P_k^-}$ matrices, $\mathbf{K_k}$ and $\hat{\mathbf{x}}_\mathbf{k}$ are time variant. Subscript d therefore means discrete. We do have optimal Kalman gain[2], but we still use the general equation $\mathbf{P_k} = (\mathbf{I}-\mathbf{K_k}\mathbf{C_d})\mathbf{P_k^-}(\mathbf{I} - \mathbf{K_k}\mathbf{C_d})^{\mathbf{T}}+\mathbf{K_k}\mathbf{R}\mathbf{K_k^T}$ instead of $\mathbf{P_k} = (\mathbf{I}-\mathbf{K_k}\mathbf{C_d})\mathbf{P_k^-}$ when we compute the a posteriori error covariance matrix. This is a computational heavier operation, but ought to yield the same result, especially since it's the more general version. It should also be noted that in the project ahead part of the loop, where we find the a priori error covariance for the time $k+1$, we use $\mathbf{E_d}\mathbf{Q}\mathbf{E_d^T}$ instead of just $\mathbf{Q}$. This is simply because we have have two process noises, w_w and w_b, resulting in $\mathbf{Q}$, a diagonal 2x2 matrix. See appendix B for further information.

These expressions are implemented into the framework. We use a state vector, `x` which holds the elements in $\mathbf{P_k}$, $\hat{\mathbf{x}}_\mathbf{k}^-$ and $\hat{\mathbf{x}}_\mathbf{k}$. This vector is passed from one function to another. In `mdlOutputs` we output states no. 8 and 10, which are ψ and b of $\hat{\mathbf{x}}$. For the entire filter, please see appendix B.10 and for implementation in Simulink, see appendix figure D.1.

[1] Brown, page 219

[2] Optimal Kalman gain means the expression yielding the minimum mean-square error estimate, $\frac{d(trace\mathbf{P_k})}{d\mathbf{K_k}} = 0$, Brown, page 217

6.4 Feed forward from estimated bias

The bias estimated by the Kalman filter is fed into the system, with current disturbance enabled. All plots can be found in appendix B.1 We can see in plot B.1 that the current disturbance is basically eliminated by the forwarded bias estimate. The ship's movement is shown in plot B.2. The input to the ship's rudder in this case is shown in plot B.3. A plot of the estimated rudder bias is included in figure B.4. When we compare the compass output and position plot of the ship with that in figure 4.2, showing simply the PD regulator autopilot, we see that the forward fed bias allows the autopilot to correctly steer the ship to the desired compass heading of 30 degrees, instead of around 25 degrees without the forward fed bias. The autopilot therefore performs very well under these circumstances.

6.5 Feed forward from estimated bias and wave filtered heading

The bias estimate is fed into the system, and the estimated heading is used in the autopilot. The system is simulated with measurement noise, and wave and current disturbances. All plots can be seen in appendix B.2. Figure B.5 shows the output of the compass, with and without feedback. The ships movement is shown in plot B.6. We see that the autopilot manages to steer the ship correctly even with current and wave disturbances. The input to the ship's rudder, when both the Kalman states are used for the simulation with current and waves enabled, are shown in plot B.7. A plot of the estimated rudder bias is included in figure B.8. Figure B.9 shows a comparison between actual and measured wave disturbance. In order to compare these, we have to change the Kalman filter slightly in order to be able to read the estimated ψ_w. We do the following changes:

```
%mdlInitializeSizes(data)
sizes.NumOutputs     = 3;
....
%mdlOutputs(t,x,u,data)
sys = [x(7) x(8) x(10)];
```

This allows us to read out the estimated wave disturbance. We can see from B.9 that the estimate deviates in the beginning, but eventually catches up and follow the actual disturbance very well. For the system as a whole, it behaves much better than when only the PD controller was implemented. This is espacially evident in the rudder input, since there is no longer violent oscillations present in the signal. All in all we have designed what appears to be a very robust autopilot, atleast according to the design requirements given to us.

Chapter 7

Discussion

We've been largely successful in establishing good results based on the plots made and the behaviour observed in the final steps of the implementation of the autopilot. The process, starting with analyzing the system with disturbances, understanding the ships dynamics and onwards to adressing the control problem using a PD controller together with the Kalman filter gave a solid boost in both motivation and learning. Our biggest bump in the road was probably the implementation of the filter, since there were some not so obvious changes that had to be done compared to the general Kalman loop described in Brown. Understanding the structure and program flow of the S-function also contributed to the implementation of the filter being the most time consuming.

The usefulness of analyzing data sets, such as that stored in `wave.mat`, showed how we can better understand disturbances and extract important data to counter their effects. Fitting analytical expressions to real world data was also a useful lesson, not to be underestimated. Since we have a wide array of tools for analytical expressions of most kinds, it's indeed very practical to make an analytical description of the real world in order to make models that closely resembles its real world counter parts.

If we had more time and a little more guts, we would implement the filter using `C`. The reason for this is simple, `C` is a powerful low level language and also much faster than the scripted `Matlab` language, and since we will undoubtly use S-functions in future simulations, it would be of great advantage to master this skill. However, we're convinced that in later courses we will have the courage to try.

Chapter 8

Conclusion

This report has described the implementation of a control system for a ship. The results achieved have demonstrated that through careful modelling of a real world scenario, with some approximations and simplifications, we can take into account noise and disturbance and deal with them in a clever way by using tools like Kalman filter. Welch's method has been used to analyze frequency components in wave disturbance data, where the we found $\omega_0 = 0.7823$ and we have designed a PD-controller fit for our needs, with a phase margin of 50 degrees and a cut off frequenzy at $\omega_c = 0.1$ rad/s. The controller was tested under various circumstances, with different disturbances or lack thereof. Though the controller by its own managed decent when measurement noise and wave disturbance was applied, the rudder input had violent oscillations, which can cause wear and tear in a real world application. When the controller was tested with current disturbance, the compass course deviated from the reference by a couple of degrees, which is an unwanted behaviour. We then tested the observability of the system with and without disturbances and it was observable in all cases. We then proceeded to the discretization of the system, since the Kalman filter itself is discrete and hence need discrete matrices. The filter was then implemented using an S-function in Simulink. The bias feed forward from the Kalman filter eliminated our previous trouble with the current disturbance and the filtered ψ gave a very smooth rudder input behaviour while still maintaining a steady compass course equal to the reference.

Appendix A

Plots

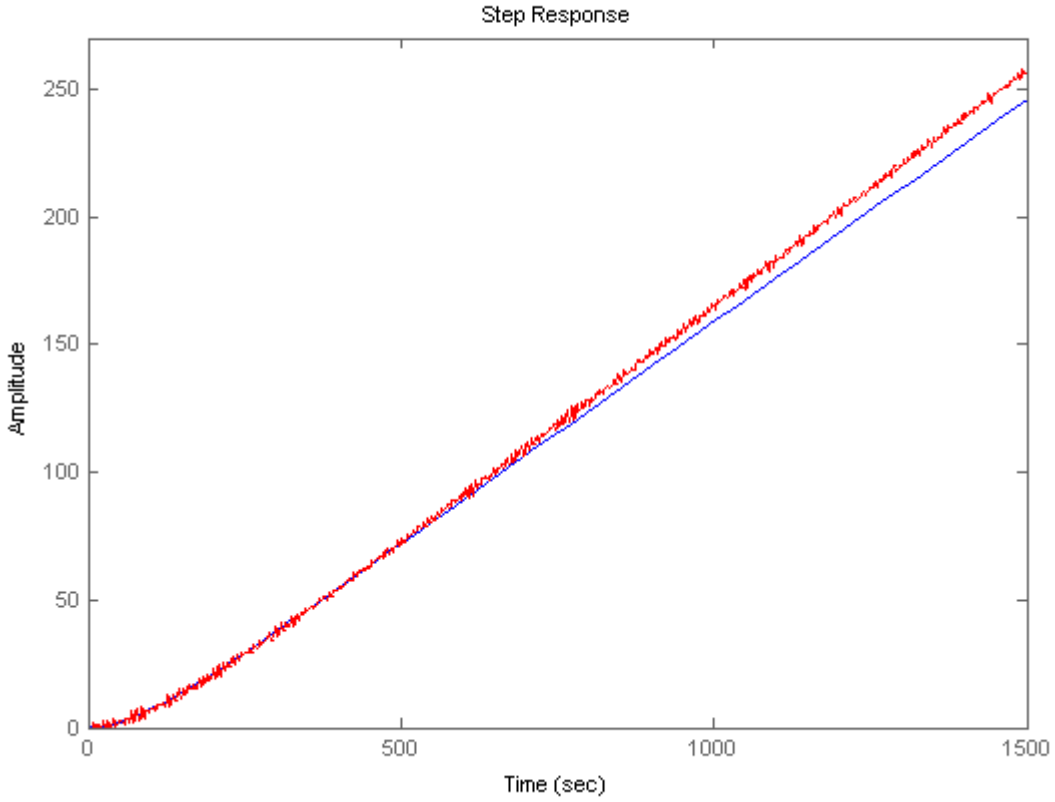

Figure A.1: The blue line is the step response of the model, while the red line is the step response of the ship. Both measurement noise and wave disturbance was turned on for the ship. In the case of lacking wave disturbance, the red line would be about the same, just less jagged.

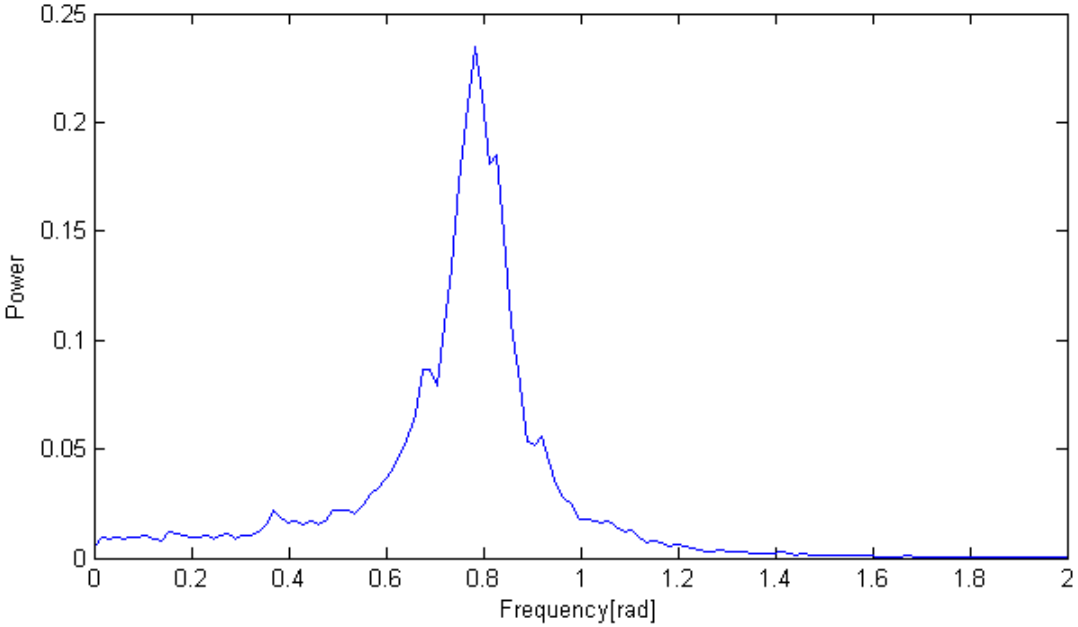

Figure A.2: The PSD found by using `pwelch()`. Note the spike, this is the dominant frequency of the waves.

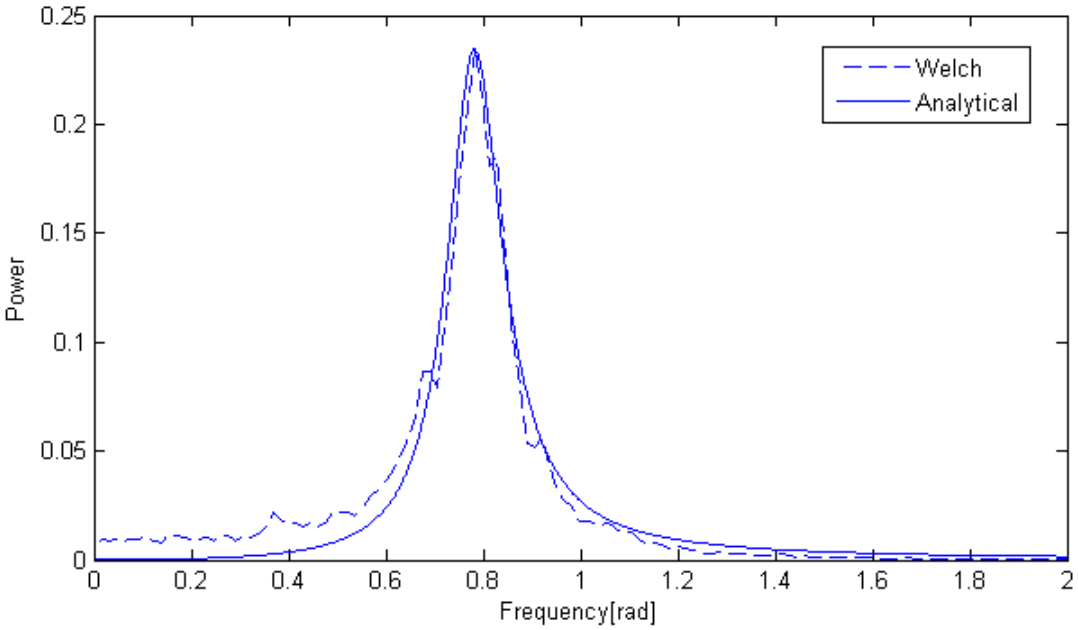

Figure A.3: The dotted graph is S_{ψ_w} using `pwelch()`, while the solid graph is P_{ψ_w} with $\lambda = 0.09$

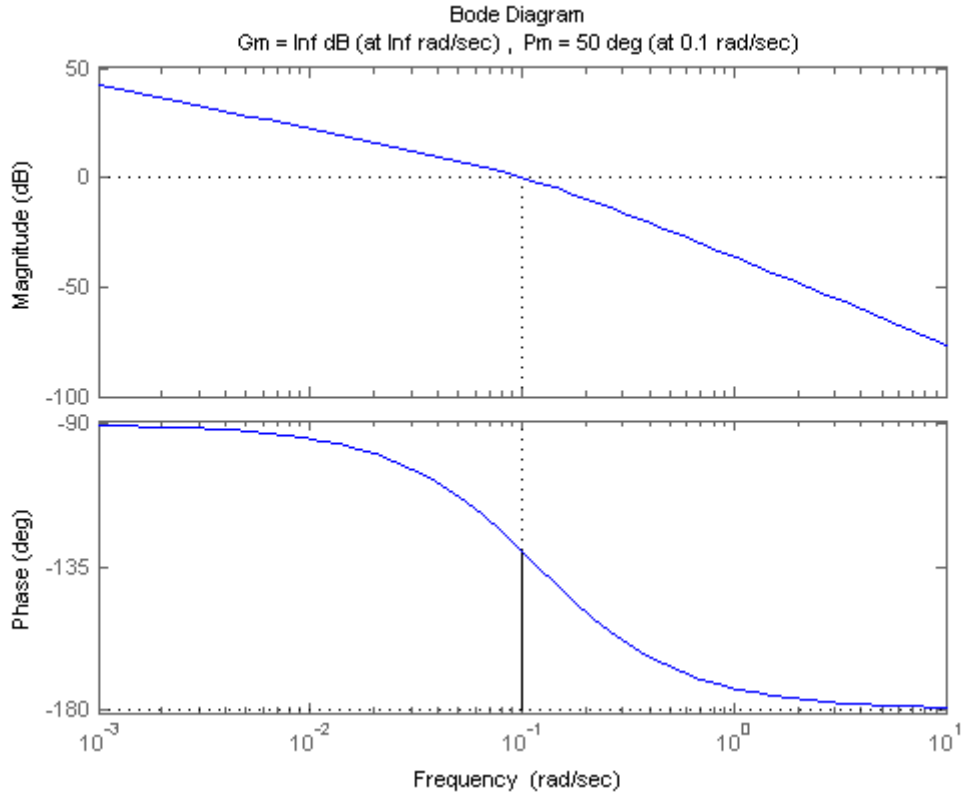

Figure A.4: The margin plot of the system with a PD controller. Note that ω_c is 0.1 rad/s and the phase margin is 50 degrees, just as intended.

Appendix B

B.1 Feed forward from estimated bias

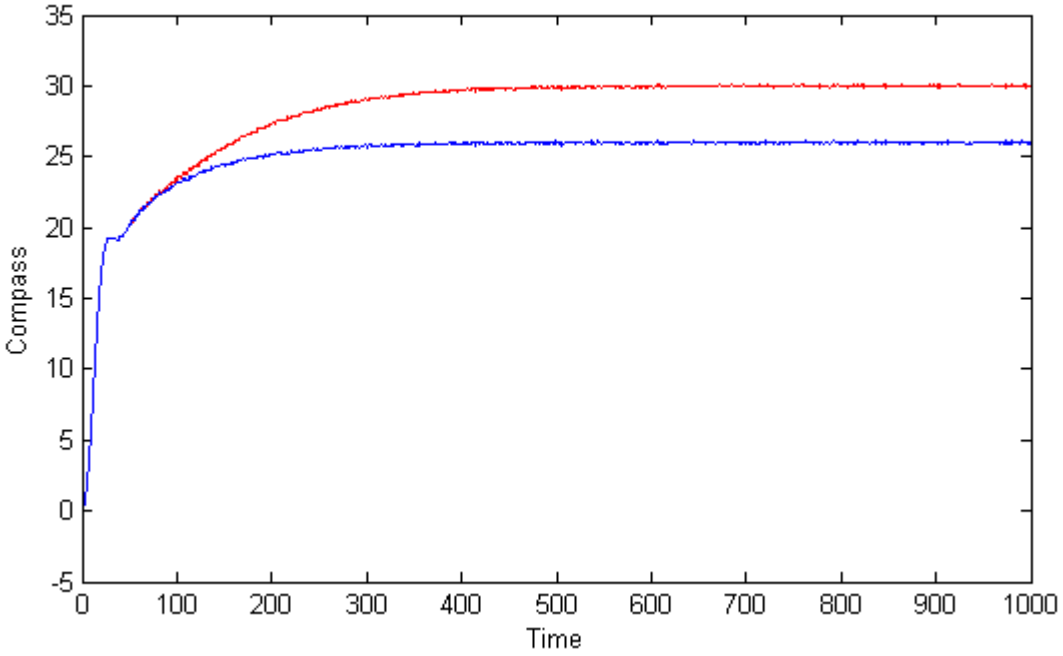

Figure B.1: Comparison of compass output with and without estimated bias forwarded. The blue plot represents no feed forward, while the red has feed forward.

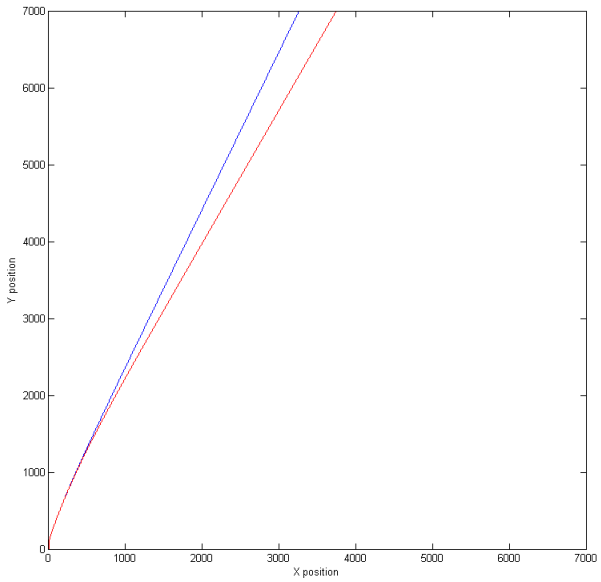

Figure B.2: Comparison of positions with and without feed forward. Red represents feed forward, while blue represents no feed forward.

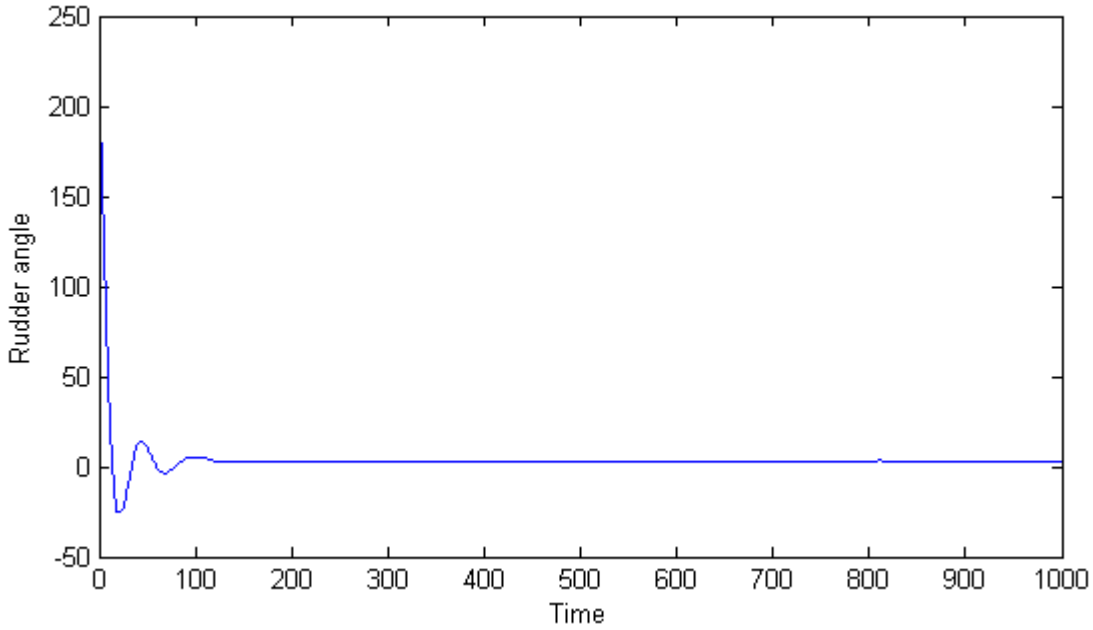

Figure B.3: The input to the ship's rudder with feed forward from the estimated bias, as output by the autopilot. This input is saturated before reaching the actual rudder.

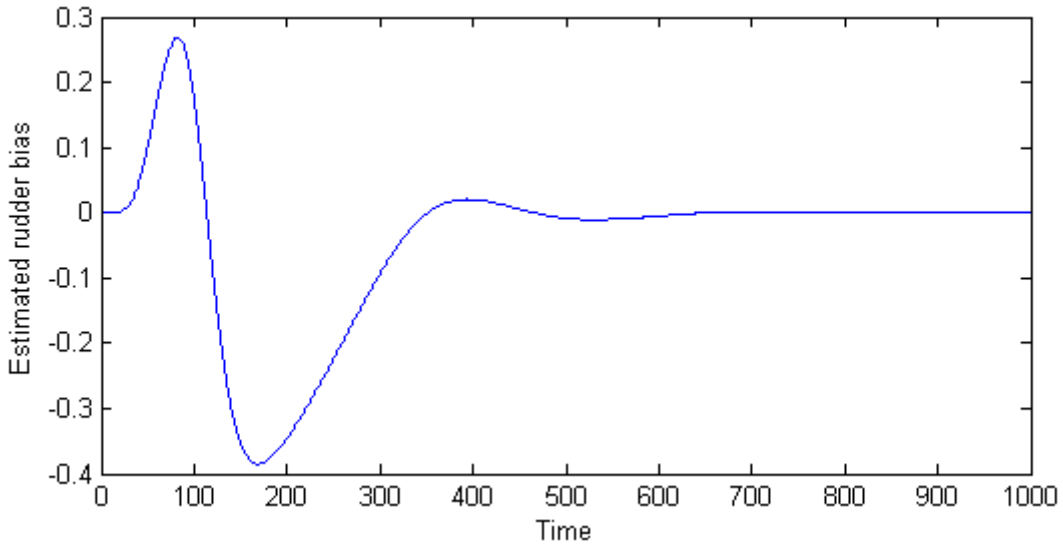

Figure B.4: Estimated rudder bias

B.2 Feed forward from estimated bias and wave filtered heading

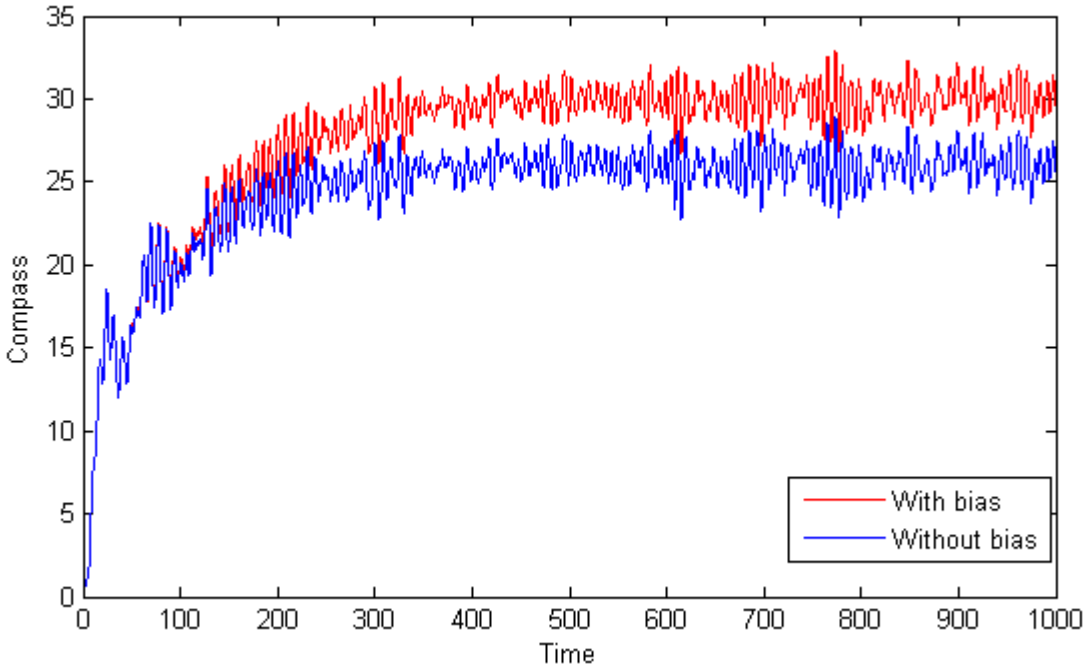

Figure B.5: Comparison of compass output with and without estimated bias forwarded. The blue plot represents no feed forward, while the red has feed forward.

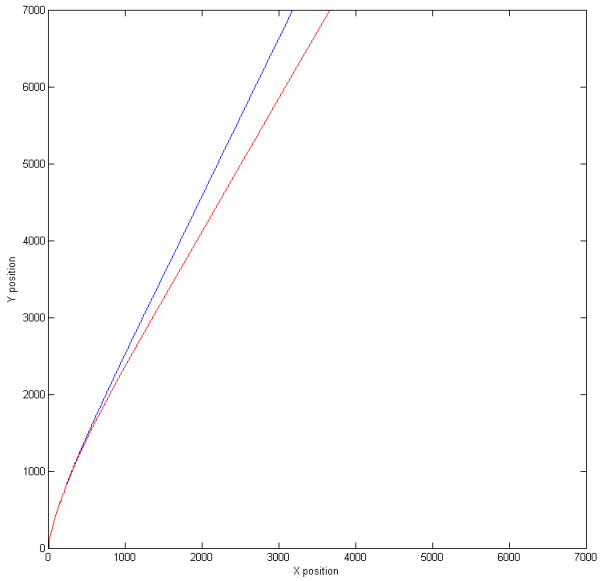

Figure B.6: Comparison of positions with and without feed forward. Red represents feed forward, while blue represents no feed forward.

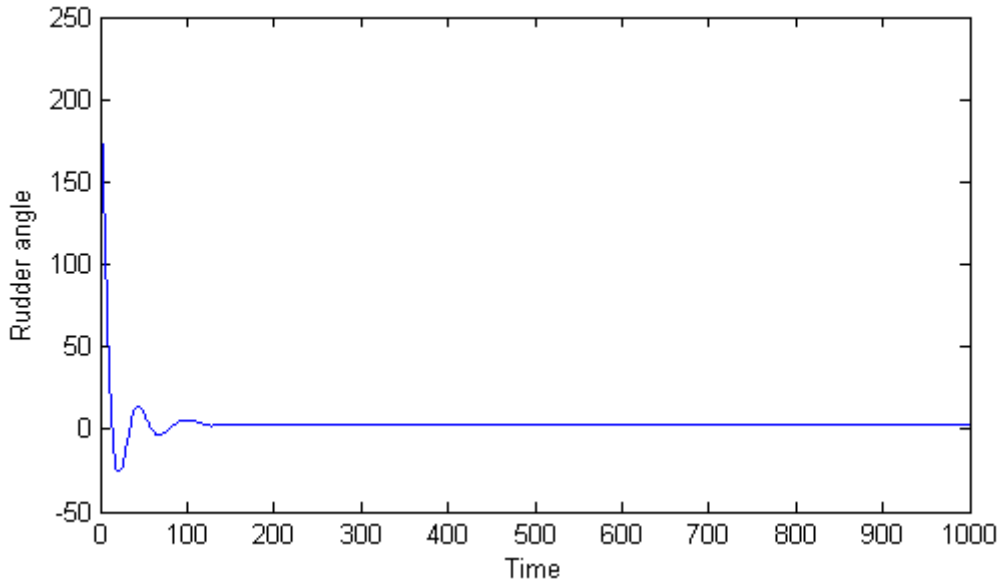

Figure B.7: The input to the ship's rudder with feed forward from the estimated bias, and wave filtered heading, as output by the autopilot. This input is saturated before reaching the actual rudder.

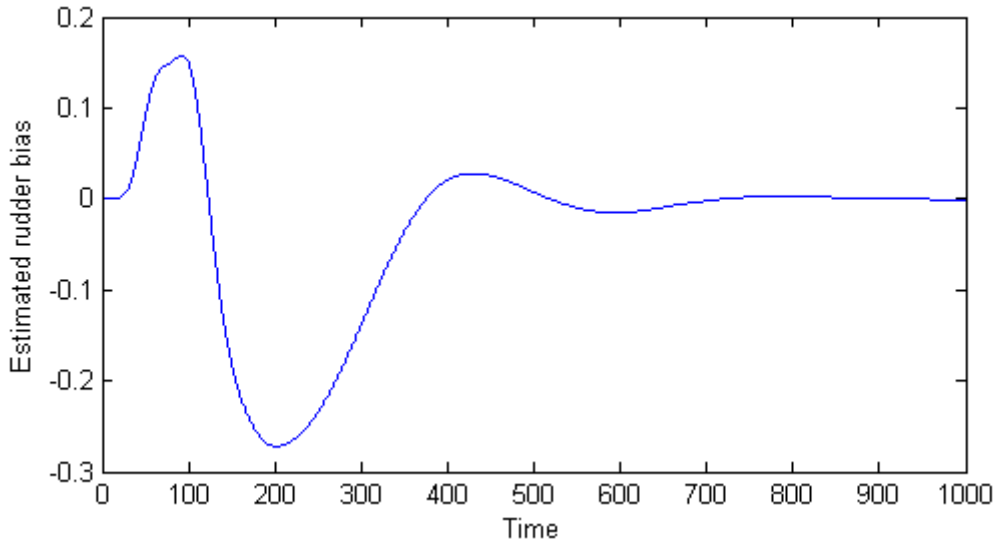

Figure B.8: Estimated rudder bias

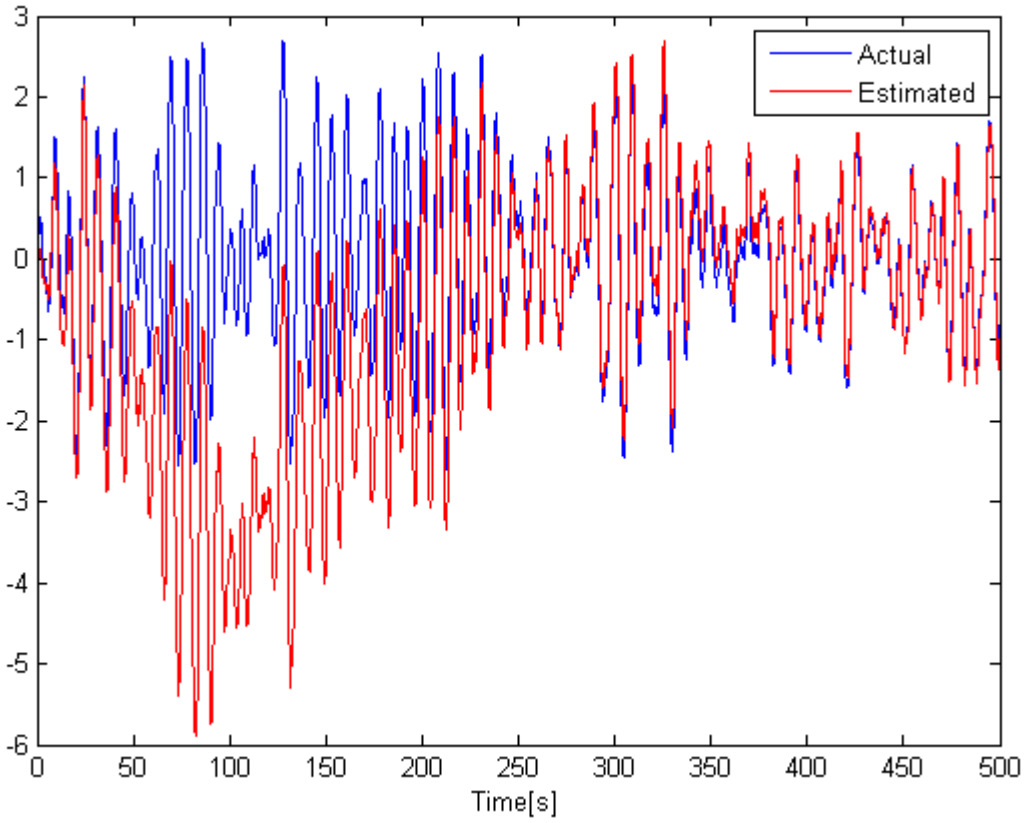

Figure B.9: Actual wave disturbance and estimated wave disturbance

```
function [sys,x0,str,ts] = kalman(t,x,u,flag,data)
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(data);
  case 3,
    sys=mdlOutputs(t,x,u,data);
  case 2,
    sys=mdlUpdate(t,x,u,data);
  case {1,4,}
    sys=[];
  case 9,
      sys=mdlTerminate(t,x,u);
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes(data)
sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 35;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [data.x0_',zeros(1,5),data.P0_(:)'];
str = [];
ts  = [-1 0];

function sys=mdlUpdate(t,x,u,data)
%Calculate the Kalman gain
P_=reshape(x(11:35),sqrt(length(x(11:35))),sqrt(length(x(11:35))));
Kk = P_*data.Cd'*inv(data.Cd*P_*data.Cd' + data.R);
%Update xhat based on the measurement
xhat_ = x(1:5);
xhat = xhat_ + Kk*(u(1) - data.Cd * xhat_);
%Calculate the a posteriori error covariance matrix
I = eye(5);
P = (I - Kk * data.Cd)*P_*(I-Kk*data.Cd)'+Kk*data.R*Kk';
%Project ahead
xhat_ = data.Ad*xhat + data.Bd*u(2);
P_ = data.Ad * P * data.Ad' + data.Ed*data.Q*data.Ed';
sys=[xhat_',xhat',P_(:)'];

function sys=mdlOutputs(t,x,u,data)
sys = [x(8) x(10)];

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


Figure B.10: The implementation of the Kalman filter. Please note that most of the comments were omitted in order to keep the code compact. It should nonetheless be quite obvious that the kalman loop is executed in the `mdlUpdate` function and ψ and b are pushed out through `mdlOutputs`.

Appendix C

Explanation of $\mathbf{EQE^T}$

In this section we explain why we multiply the $\mathbf{Q}$ matrix with $\mathbf{E_d}$ and $\mathbf{E_d^T}$. Let's take a look at a simple matrix, let's say the 3x2 matrix

$$\mathbf{E} = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \\ e_{31} & e_{32} \end{bmatrix} \tag{C.1}$$

and the 2x2 matrix

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix} \tag{C.2}$$

When we do the operation $\mathbf{EQE^T}$, the following matrix is produced:

$$\mathbf{EQE^T} = \begin{bmatrix} {e_{11}}^2 q_{11} + {e_{12}}^2 q_{22} & e_{11}e_{21}q_{11} + e_{12}e_{22}q_{22} & e_{11}e_{31}q_{11} + e_{12}e_{32}q_{22} \\ e_{11}e_{21}q_{11} + e_{12}e_{22}q_{22} & {e_{21}}^2 q_{11} + {e_{22}}^2 q_{22} & e_{21}e_{31}q_{11} + e_{22}e_{32}q_{22} \\ e_{11}e_{31}q_{11} + e_{12}e_{32}q_{22} & e_{21}e_{31}q_{11} + e_{22}e_{32}q_{22} & {e_{31}}^2 q_{11} + {e_{32}}^2 q_{22} \end{bmatrix} \tag{C.3}$$

As we can see, all elements are on the form of covariance calculation based on the covariances in $\mathbf{Q}$ and the elements of $\mathbf{E}$ which is after all the disturbance matrix. This is then added to the rest of the expression for $\mathbf{P_{k+1}^-}$.

Appendix D

Simulink diagrams

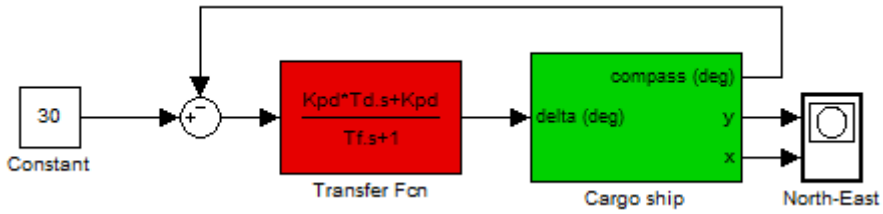

Figure D.1: The system with PD controller.

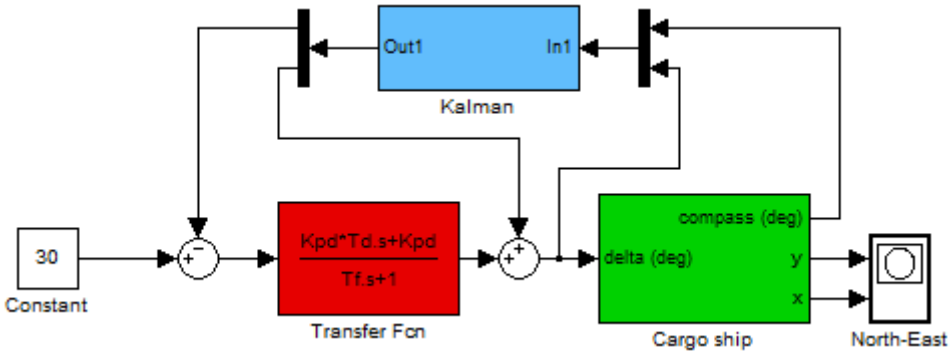

Figure D.2: The system with both PD controller and Kalman filter.

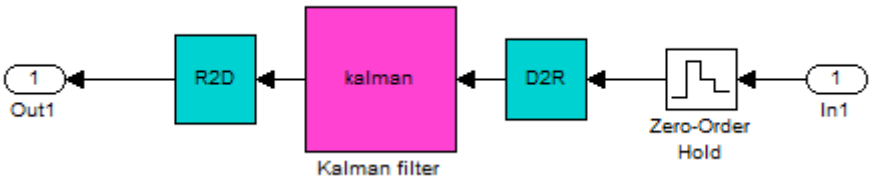

Figure D.3: Inside the Kalman filter block

Bibliography

[1] Robert Grover Brown, Patrick Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley and sons, Inc., 3rd Edition, 1997.

[2] Chi-Tsong Chen, *Linear System Theory and Design*. Oxford University Press, 3rd Edition, 2009.