TMR4240 Project Part II:
Design of Dynamic Positioning System
Control System Design

Group 15
Andreas Haugland [742503]
Yngve Lilleeng Jenssen [477957]
Simen Troye Røang [768063]
Sindre Sagsveen Slåttum [767814]

NTNU – Trondheim
Norwegian University of
Science and Technology
Department of Marine Technology

Contents

1 Introduction

The project given in the course *TMR4240 Marine Control Systems 1* aims to increase the participants knowledge in designing and validating surrounding systems for a DP positioned vessel. This first part contains the implementation of a current model, reference model and a controller using Matlab and Simulink.

The report is organized such that each task in the assignment has a corresponding section. The sections are introduced by defining the objective of the task, explaining the starting point and how the task was solved. Thereafter follows the specific findings of each sub-problem or a brief discussion of the results.

Methods used to derive mathematical expressions in the system are explained briefly and the resulting equations are given in the different sections. Entire derivations are not included in the report. Further on, for relevant plots and step responses for the different simulations are found in the parts where they are discussed, along with supplementing plots and figures. MATLAB code may be found in the Appendix and will be uploaded with the rest of the files.

An overview of the Stimulink model used for the simulations can be seen in figure 1. The subsystems can be seen in the sections associated to the subsystems.

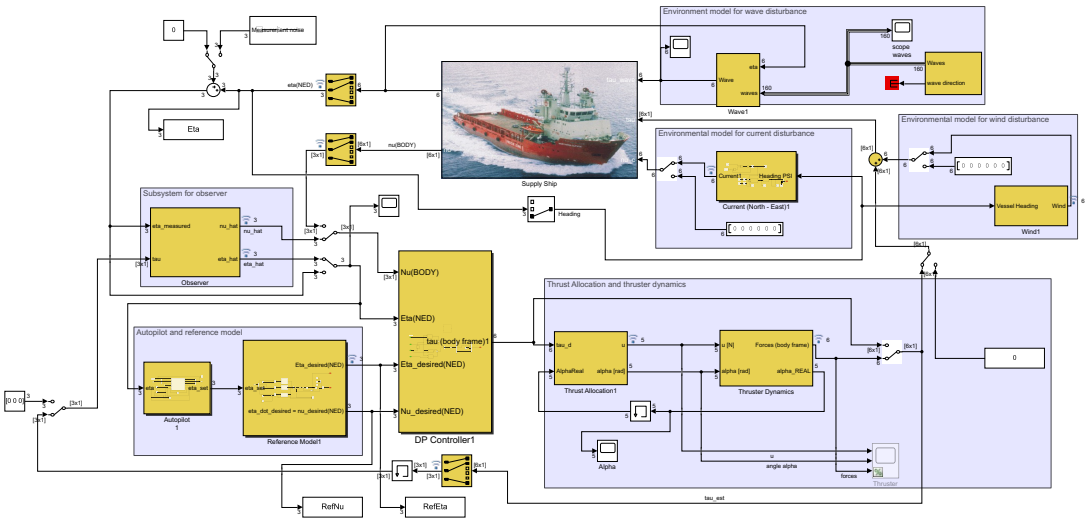


Figure 1: Simulink model overview of the complete system

2 Process plant model

The purpose of the process plant model is to give a detailed mathematical description of the vessel dynamics, systems, components and surroundings. The process plant model is highly detailed and is comprised of high fidelity models. These models are used to simulate the real physical properties of the system. This simulator can then be used for design and validation of control systems. The better the model captures the real system the more accurate the control system will be, and as a result a bad model can lead to a bad designed controller.

2.1 Vessel dynamics

The mathematical model of a marine vessel can be separated into two terms, one low-frequency (LF) model and one wave-frequency (WF) model. For a conventional ship, such as a PSV, only the horizontal plane degrees of freedom are of interest. These three are the surge-, sway-and yaw-motion.

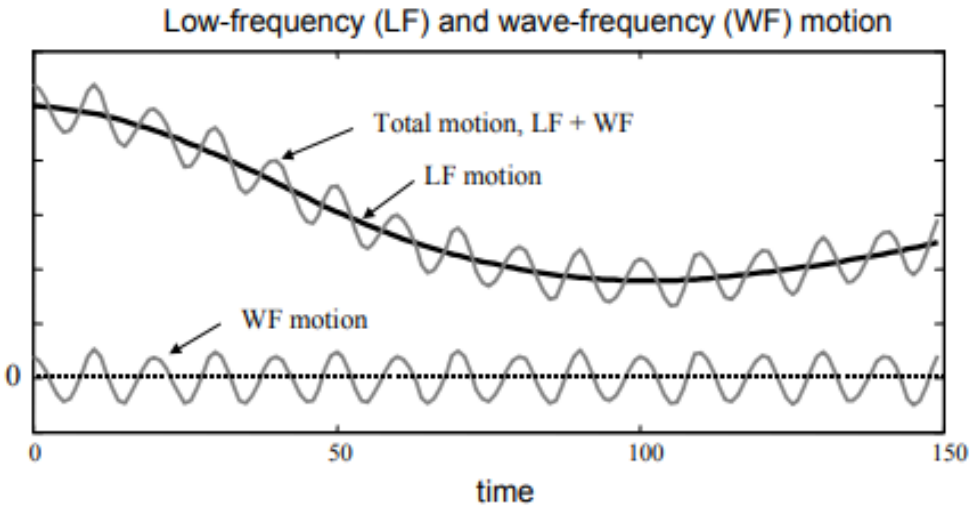


Figure 2: Splitting of low frequency and high frequency motion

2.1.1 Low-frequency model

LF equation (7.67).

$$\mathbf{M}\dot{\nu} + \mathbf{C}_{RB}(\nu)\nu + \mathbf{C}_A(\nu_r)\nu_r + \mathbf{D}(\kappa, \nu_r) + \mathbf{G}(\eta) = \tau_{env} + \tau_{moor} + \tau_{ice} + \tau_{thr} \quad (1)$$

Here you have the thrust loads, mooring loads, environmental (i.e. wave/current/wind loads) and ice loads on the right side of the equation. On the left side of the equation $\mathbf{M}\dot{\nu}$ is the generalized inertia forces and $\mathbf{M}$ is the generalized mass matrix containing mass, moments of inertia and added mass coefficients. The $\mathbf{C}_{RB}(\nu)\nu + \mathbf{C}_A(\nu_r)\nu_r$ is the Coriolis and centripetal force. The force coming from the rigid-body motions are $\mathbf{C}_{RB}(\nu)\nu$. The centripetal and Coriolis force coming from the potential part of current is $\mathbf{C}_A(\nu_r)\nu_r$. This force is coming from the added mass. $\mathbf{D}(\kappa, \nu_r)$ is the force coming from the generalized damping and current forces. The $\mathbf{G}(\eta)$ is a generalized restoring force coming from the hydrostatic forces. The matrices $\mathbf{M}$, $\mathbf{C}_{RB}(\nu)$ and $\mathbf{C}_A(\nu_r) \in \mathbb{R}^{6X6}$. $\mathbf{D}(\kappa, \nu_r)$ and $\mathbf{G}(\eta)$ $\in \mathbb{R}^{3X3}$. The loads that we not consider in this model, for this part of the project is τ_{moor}, τ_{ice} and τ_{env}.

2.1.2 Wave-frequency model

The WF motion is caused by first-order wave loads. The wave loads on the vessel can in turn be split into motion caused by wave reaction and wave excitation. The reaction forces also called radiation forces occur as the vessel is forced to oscillate with the wave frequency. These hydrodynamic loads are the added mass forces, wave radiation damping forces and restoring forces. The effect on the vessel as it is restrained from oscillating is the excitation forces. In the excitation forces you have the forces from the undisturbed wave which is the Froude-Krylof and diffraction forces and moment coming from that vessels influence of the surrounding waves.

$$\mathbf{M}(\omega)\ddot{\eta}_{R\omega} + \mathbf{D}_p(\omega)\dot{\eta}_{R\omega} + \mathbf{G}\eta_{R\omega} = \tau_{wave} \quad (2)$$

Here η_{Rw} is a motion vector in a hydrodynamic frame which is a reference parallel frame.

2.2 Kinematics

The study of the dynamics of a vessel can be divided into kinematics and kinetics, where kinematics discusses and treats the geometrical aspects of motion and the parts associated with kinetics concerns the analysis of forces causing motion. This section aims to define motion variables, reference frames and conversion between the reference frames (Fossen 2011, chapter 2) [3].

2.2.1 Motion variables and reference frames

Theory for this part is motivated mostly by Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control chapter 2 [3].

For marine craft moving in six degrees of freedom, six independent coordinates are necessary to determine position and orientation. The three first coordinates and their time derivatives correspond to the position and translation motion along the x, y and z axes, while the last three coordinates and their time derivatives are used to describe orientation and rotational movement (Fossen T.I, page 15) [3]. This leads up to the *SNAME* notation for marine vessels, seen in table 1. This is denoted as the vectors η, containing position and Euler angles, and ν, containing linear and angular velocities.

DOF		Forces and moments	Linear and angular velocities	Position and Euler angles
1	Surge	X	u	x
2	Sway	Y	v	y
3	Heave	Z	w	z
4	roll, heel	K	p	ϕ
5	pitch, trim	M	q	θ
6	yaw	N	r	ψ

Table 1: Notation of SNAME (1950) for marine vessels

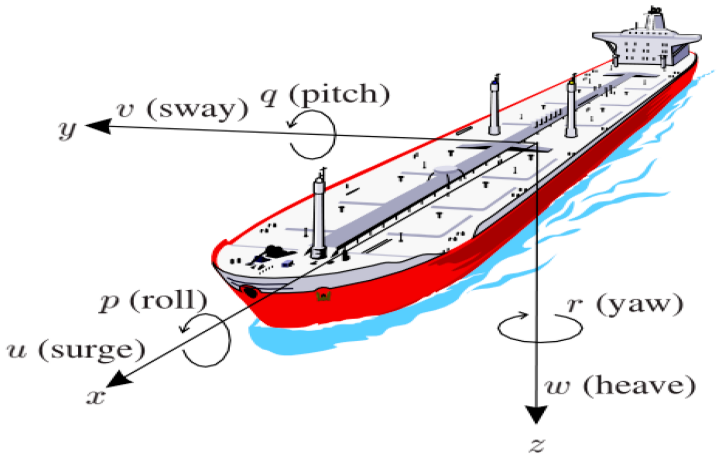


Figure 3: Figure displaying some of the motion variables on a ship

When analysing motion of marine craft in 6 DOF, it is convenient to define two Earth-centered coordinate frames, and several geographic reference frames. *Fossen* defines the earth centered frames as "Earth-centered inertial" (ECI) and "Earth-Centered-Earth-Fixed" (ECEF) reference frames, and the geographic reference frames as "North-East-Down" (NED) and the body-fixed reference frame (BODY).
ECI $i = (\mathrm{x}_i, \mathrm{y}_i, \mathrm{z}_i)$: is an inertial frame for terrestrial navigation. It is a non-accelerating frame in which Newtons laws of motion apply. The origin is located at the center of the earth.
ECEF $e = (\mathrm{x}_e, \mathrm{y}_e, \mathrm{z}_e)$: has its origin fixed to the center of the earth, but the axes rotate relative to the inertial frame ECI, which is fixed in space. This coordinate system is usually used for global guidance, navigation and control, for instance for intercontinental transit.
NED $n = (\mathrm{x}_n, \mathrm{y}_n, \mathrm{z}_n)$: This is the coordinate system we refer to in our everyday life. It is often defined as the tangent plane on the surface of the earth moving with the craft, but with x and y pointing towards true North and East, while the z axis points downwards normal to the earths surface. The location of n relative to e is determined using two angles l and μ.
BODY $b = (\mathrm{x}_b, \mathrm{y}_b, \mathrm{z}_b)$: is a moving coordinate frame that is fixed to the craft. The position and orientation of the craft are described relative to the inertial reference frame, while the linear and angular velocities of the craft should be expressed in BODY. The origin is usually chosen to be a point somewhere in the middle of the ship, near the water line. The BODY axes are defined as x_b: longitudinal axis (directed from aft to fore), y_b: transverse axis (directed to starboard) and z_b: normal axis (directed from top to bottom).

Figure 4 displays how the different reference frames are related to each other.

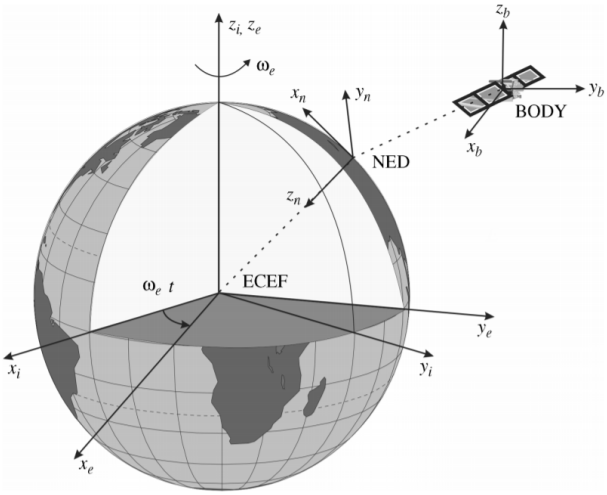


Figure 4: figure displaying the relationship between the reference systems

2.2.2 Converting reference frame

For converting between the different reference frames one uses a rotation matrix. In general you rotate the reference frames with $\eta_2 = [\phi, \theta, \psi]$, but in the case of maneuvering a vessel you only rotate the frames in yaw so you only use ψ. For converting from body-fixed to earth-fixed reference frames velocity you use:

$$\dot{\eta} = \mathbf{R}(\psi)\nu \tag{3}$$

$$R(\psi) = \mathbf{C}_{z,\psi}^{T} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

2.3 MSS

Marine systems simulator is a simulation environment capable of representing several aspects of a marine system such as hydrodynamics, structural mechanics, marine machinery, electric power generation, navigation and control. This system is created using Matlab and Simulink. The MSS Simulink library contains vessel blocks that utilizes the models previously discussed. In this project the block *Supply Ship* is used to simulate the vessel dynamics.
In part 1 the block uses two inputs, the τ_{thr}, and the current vector, ν_c. τ_{thr} is in this case the direct output from the controller. The response of the vessel is outputted as the vessel position, η, and the vessel velocity, ν.

3 Control plant model

The control plant model captures only the essential behaviour of our vessel. This model contains only main physical properties of the process. This simplified model can be implemented in the controller design in order to archive better performance and robustness. Similarly to the process plant model the vessel dynamics can be separated several parts. This includes a low-frequency model, a wave frequency model, a bias model and a measurement part.

The complete control plant model is a combination of of all these models, and can be written as:

$$\dot{\xi} = \mathbf{A}_w \xi \tag{5}$$

$$\dot{\eta} = \mathbf{R}(\psi_d)\nu \tag{6}$$

$$\dot{\mathbf{b}} = -\mathbf{T}_b^{-1}\mathbf{b} + \mathbf{E}_\mathbf{w}\mathbf{w}_\mathbf{b} \tag{7}$$

$$\dot{\nu} = -\mathbf{M}^{-1}\mathbf{D}\nu + \mathbf{M}^{-1}\mathbf{R}^T(\psi_d)\mathbf{G}\eta + \mathbf{M}^{-1}\mathbf{R}^T(\psi_d)\mathbf{b} + \tau \tag{8}$$

$$y = \eta + \eta_w + \mathbf{v} \tag{9}$$

3.1 Low-frequency model

By simplifying equation (1) we can derive a nonlinear low-frequency model in surge, sway and yaw with sufficient details for a control plant model (Sørensen 2018, p158) [4]. The equations are given by:

$$\dot{\eta} = \mathbf{R}(\psi)\nu \tag{10}$$

$$\dot{\nu} = -\mathbf{M}^{-1}\mathbf{D}\nu + \mathbf{M}^{-1}\mathbf{R}^T(\psi_d)\mathbf{G}\eta + \mathbf{M}^{-1}\mathbf{R}^T(\psi_d)\mathbf{b} + \tau \tag{11}$$

In these equations $\mathbf{v} = [u, v, r]^T$ is the velocity vector. $\eta = [x, y, \psi]^T$ is the position vector. $\mathbf{b} \in \mathbb{R}^3$ is the bias vector, commonly selected to be the first order Markov model, $\dot{\mathbf{b}} = -\mathbf{T}_b^{-1} + \mathbf{E}_b\mathbf{w}_b$. In the Markov model $\mathbf{w}_b$ is a zero-mean Gaussian white noise vector. The bias model handles the error in the modelling, the slowly varying forces and moment from second-order environmental loads. Lastly the $\tau = [\tau_x, \tau_y, \tau_\psi]^T$ is the control parameter input vector. Similarly to the process plant model $\mathbf{M}$ is the generalized mass matrix, $\mathbf{D}$ is the force from generalized damping and current forces. Lastly $\mathbf{G}$ is the restoring force matrix. In a DP system with no mooring lines there is no stiffness and $\mathbf{G}$ is zero. This results in elimination of the term containing G in equation 11.

The generalized mass matrix and damping matrixes were given in part two of the project and

$$\mathbf{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} = \begin{bmatrix} 6.8177e6 & 0 & 0 \\ 0 & 7.8784e6 & -2.5955e6 \\ 0 & -2955e6 & 3.57e9 \end{bmatrix} \tag{12}$$

$$\mathbf{D} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} = \begin{bmatrix} 2.6486e5 & 0 & 0 \\ 0 & 8.8164e5 & 0 \\ 0 & 0 & 3.3774e8 \end{bmatrix} \tag{13}$$

3.2 Wave-frequency model

The wave-frequency model for a controller can be modelled as a set of uncoupled harmonic oscillators with a damping term. You are interested to model this WF motion so you can filter it out as a process noise in the observer. This is because the controller cant compensate for this forces/motions.

$$\dot{\xi}_w = \mathbf{A}_w \xi_w + \mathbf{E}_w \mathbf{w}_w \tag{14}$$

$$\eta_w = \mathbf{C}_w \xi_w \tag{15}$$

Here $\mathbf{A}_\omega$ is the system matrix, $\mathbf{E}_\omega$ is the disturbance matrix and $\mathbf{C}_\omega$ is the measurement matrix. The $\xi_w \in \mathbb{R}^6$ is the motions and veloities in the WF model, $\mathbf{w}_w \in \mathbb{R}^3$ is the zero-mean Gaussian white noise vector.

The wave frequency model parameters are slowly varying quantities depending on the sea state.
The matrices in the model can be implemented like this:

$$A_w = \begin{bmatrix} \mathbf{0}_{3x3} & \mathbf{I}_{3x3} \\ -\mathbf{\Omega}^2 & -2\mathbf{\Lambda}\mathbf{\Omega} \end{bmatrix} \tag{16}$$

$$\mathbf{C}_w = \begin{bmatrix} \mathbf{0}_{3x3} & \mathbf{I}_{3x3} \end{bmatrix}, \mathbf{E}_w = \begin{bmatrix} \mathbf{0}_{3x3} \\ \mathbf{K}_w \end{bmatrix} \tag{17}$$

Here $\mathbf{\Omega} = \text{diag}\{\omega_1, \omega_2, \omega_3\}$, $\mathbf{\Lambda} = \text{diag}\{\xi_1, \xi_2, \xi_3\}$ and $\mathbf{K}_w = \text{diag}\{K_{w1}, K_{w2}, K_{w3}\}$. ω_i is dependent on the sea state and is sat to $\frac{2\pi}{T_p}$ where T_p is the peak period of the wave spectrum.
The relative damping ratio ζ_i is depending on the narrow bandedness of the motion. Typical values are 0.05-0.10 (p.159 lecture notes, Marine Cybernetics[4]). For a Jonswap spectrum the value is recommended to be sat to 0.10(p.217 Fossen 2011[3]). K_w is filter gains dependent on the sea state. A convenient way to define K_w is $K_w = 2\zeta_i \omega_i \sigma$ where $\sigma^2 = \max_{0<\omega<\infty} S(\omega)$(p.216 Fossen 2011[3]).

3.3 Bias model

The bias model corresponds to equation 7. For $\mathbf{b} \in \mathbb{R}^3$, a common used model is the *Markov model*, written in equation 7 as,

$$\dot{\mathbf{b}} = -\mathbf{T}_b^{-1}\mathbf{b} + \mathbf{E}_\mathbf{w}\mathbf{w}_\mathbf{b}$$

Here $\mathbf{wb} \in \mathbb{R}^3$ is a zero mean white gaussian noise, $\mathbf{E_b} \in \mathbb{R}^3$ is a diagonal scaling matrix, where the values chosen are listed in the section for the extended kalman filter. $\mathbf{T_b} \in \mathbb{R}^3$ is a diagonal bias time constant matrix. typically chosen with large values, such as $T_b = 1000s$, which guarantees that the Markov model converges after a certain time.

Another model for $\mathbb{R}^3$ suggested by Sørensen 2018 (p. 159) [4], and Fossen 2011 (p. 306, eq 11.86) [3], is the Wiener process, or a "random walk", where equation 7 is reduced to

$$\dot{\mathbf{b}} = \mathbf{E_w w_b}$$

.

3.4 Measurement

The measurement is given by equation 9, written as:

$$y = \eta + \eta_w + \mathbf{v}$$

where η_w is the same as in equation 15, and $\mathbf{v}$ is a zero mean white gaussian noise representing model uncertainty. the measurement $\mathbf{y}$ is the sum of the LF and WF components corresponding to the GNSS and compass measurement.

After implementation of the environment wave model, η_w is included in the process plant, and the η comming out from the ship mss-block includes η and η_w. The white measurement noise $\mathbf{v}$ is set to be zero.

4 Environmental models

This part of the report contains the derivation of the environmental models, which is limited to only a current disturbance for part one. In order to make the simulator more realistic the environmental introduces a disturbance to the system.

4.1 Current model

Implementation of the current velocity and direction is done by using a ramp, saturation and current model block in Simulink. The ramp block let us vary the current direction or current magnitude during the simulation. The model is implemented as shown in figure 5. The current block is from the MSS GNC library and creates a vector in NED-frame with the current velocity components in north and east direction. The down component of the current is equal to zero because we only consider the horizontal plane.
The current velocity vector ν_c can be written like this:

$$\nu_c = [V_c \cos(\psi_c), V_c \sin(\psi_c), 0]^T \tag{18}$$

The magnitude and heading of the current modeled as a Gauss-Markov process (Sørensen p.117) [4].

$$\dot{V}_c + \mu V_c = w \tag{19}$$
$$V_{c,min} \leq V_c \leq V_{c,max} \tag{20}$$
$$\dot{\psi_c} + \mu_2 \psi_c = w_2 \tag{21}$$
$$\psi_{c,min} \leq \psi_c \leq \psi_{c,max} \tag{22}$$

Where w is white noise with power of 0.1 and μ is equal to 0.001. These parameters let's the heading ψ variate randomly between 5 °and -5 °around the mean wind direction.

Since the input to the supply ship must be in body frame, the velocity vector is transformed into body frame coordinates with the yaw transposed rotation matrix in yaw from the MSS toolbox. This is closer described in the reference frame chapter. Only here we are converting from NED-frame to body-frame which gives the following equation.

$$\nu_{C,BF} = R(\psi)^{-1} \nu_{C,NED} \tag{23}$$

The relative velocity between the ship and current is taken into account inside the supply ship model.
The current environmental model implementation is shown in figure (5) and (6).

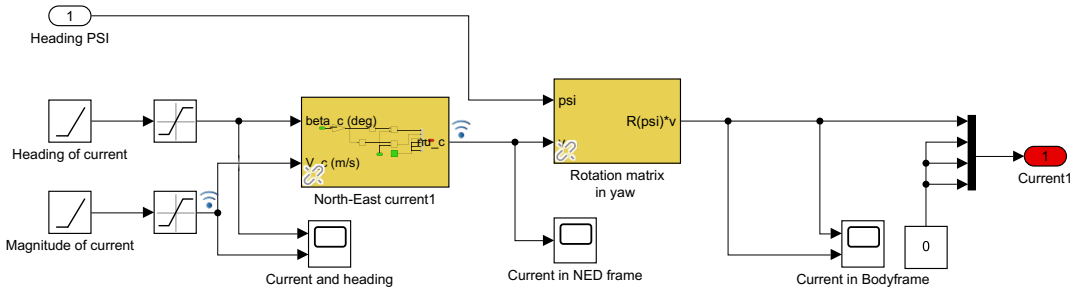


Figure 5: Simulink model overview of the current model

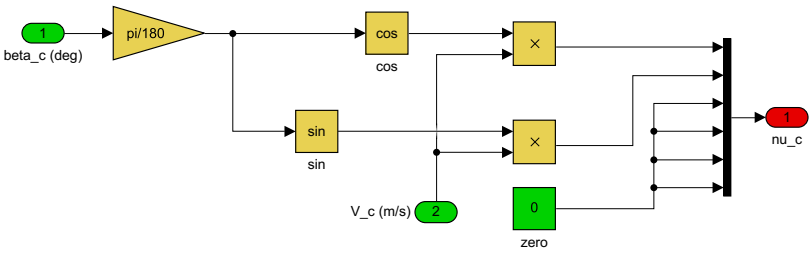


Figure 6: Simulink model overview of the subsystem in the current model

4.2 Wind model

The wind is divided into a mean, slowly varying and gust component. The mean term is set to a constant value and the slowly varying component is implemented by a first order Gauss-Markov Process shown in equation (24) where $\mu = 0.001$ similar to the current heading variation component. The white noise power of w is equal to 0.005 which is the same as in exercise 2, this gives an approximately varying velocity with amplitude 5 [m/s].

$$\dot{\bar{U}} + \mu \cdot \bar{U} = w \tag{24}$$

The direction is also varying between -5 °and 5 °around the mean direction, this is implemented almost identically with the current model by using a Gauss-Markov Process as shown in (22).
To estimate the gust component, a gust spectrum is used to calculate a vector of gust magnitudes over a given time interval. This is done in the init wind.m file and will therefore be the same for all the simulations. A spectrum of the Harris type is used to calculate the wind gusts.
The wind force is calculated using the given wind coefficients from table .... By checking the relative angle of the wind on the vessel a coefficient C_d is found for each degree of freedom. These values are given in the C_w matrix which is defined in init wind.m. Because of this, the chosen coefficient and therefore force is the one closest to the values in the wind coefficients table. To calculate the wind forces during the simulation, a Matlab function block is added in the wind block, this function takes care of the logic in choosing the correct coefficients and calculating the correct forces.

The wind model is implemented as figure 7 illustrates, and the code can be found in the appendix and in the submitted files:

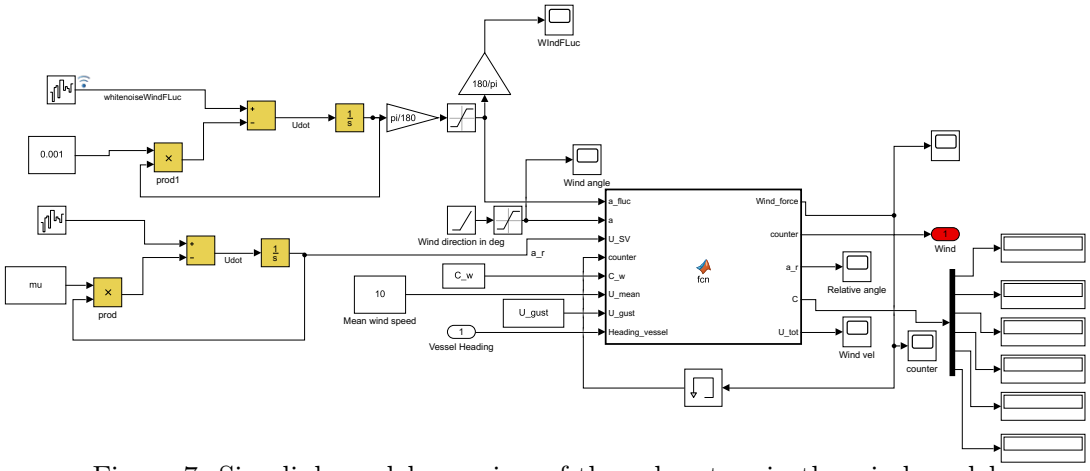


Figure 7: Simulink model overview of the subsystem in the wind model

4.3 Wave model

The wave loads where calculated from a JONSWAP wave spectrum using a wave block from the MSS toolbox. This generated 160 wave components. In the wave block it was possible to adjust parameters like peak frequency, significant waveheight etc. By passing the waves and heading of the vessel in another MSS toolbox block which calculates the forces in surge, sway, heave, pitch, roll and yaw and then into the model.
For the modelling of the wave loads both 1.order and 2. order wave loads are considered. The 2.order wave loads consists of a mean component and a slowly varying component. The 1.order waveloads response are filtered out in the observer.

5 Reference model

A reference model is deemed necessary for the DP control system since it is required that our DP controller is able to work in both stationkeeping and when following a trajectory. For the later part when the vessel is set to follow a multiple reference points, an autpilot is implemented. The reference model was implemented in a NED-frame and lets us choose a global position.

5.1 Theory

The reference model is implemented to smoothen the reference signal and minimize the difference between the reference signal. This is done to isolate the vessel and controller from a pure step function, such as a change in the reference value from 0 to 50 meters.

The reference model we choose was the cascade reference model(page 181-182) where you have a first order low-pass filter and a mass-spring damper system. The first order set-point low-pass filter smoothens the signal and will eliminate step changes in the output from the proportional part of the controller. The mass-spring damper system takes into account the dynamics of the system because the vessel needs to follow the trajectory. The reference model is given by equation 25 and equation 26.

$$\mathbf{a}_d^e + \mathbf{\Omega}\mathbf{v}_d^e + \mathbf{\Gamma}\mathbf{x}_d^e = \mathbf{\Gamma}\mathbf{x}_{ref} \tag{25}$$

$$\dot{\mathbf{x}}_{ref} = -\mathbf{A}_f\mathbf{x}_{ref} + \mathbf{A}_f\eta_r \tag{26}$$

Where the equation for the matrices are:

$$\mathbf{\Omega} = diag\{2\zeta_i\omega_i\}, i = 1, 2, 3 \tag{27}$$

$$\mathbf{\Gamma} = diag\{\omega_i^2\}, i = 1, 2, 3 \tag{28}$$

$$\mathbf{A}_f = diag\{\frac{1}{t_i}\}, i = 1, 2, 3 \tag{29}$$

The reference model yielded the following Simulink model, as seen below in figure 8.

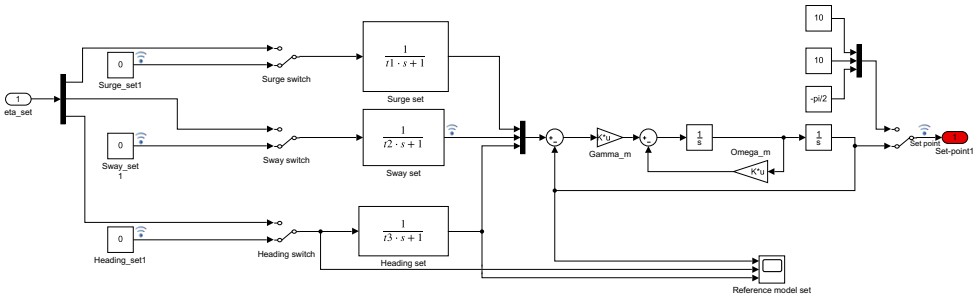


Figure 8: Simulink model overview of the reference model

5.2 Tuning of reference model

The tuning of the reference model is done by individually scoping each position trajectory and compare it with the input reference trajectory. To avoid oscillations, the damping, ζ_i, is tuned to critically damped or over damped. For a critically damped system $\zeta = 1$ and for a over damped system, $\zeta > 1$. The damping ζ_i was first set to 1 and then increased if any oscillations occurred. This was done individually for each state in combination with tuning of the controller if necessarily.

The natural period corresponding to ω_i is usual between 50-200 seconds for vessels of this size. It was first set to 200 seconds, which gave us a ω by the formula: $(\omega_i = \frac{\pi}{100})$ and then reduced until the response started to oscillate. If the reference trajectory was too slow, it was necessary to increase the natural frequency of the mass-spring-damper system. In other words increase ω_i. If the reference trajectory was to far from the vessel state, ω_i needed to be decreased to compensate for the difference. The time constant for the lowpass-filter was tuned using that the response of the filter should reach 63.2 % of its maximum value at time t_i for each state.

i	t	ζ	ω
1	20	1	$\frac{\pi}{100}$
2	20	1	$\frac{\pi}{100}$
3	20	1.2	$\frac{\pi}{150}$

Table 2: Ended tuning-parameters of the reference model

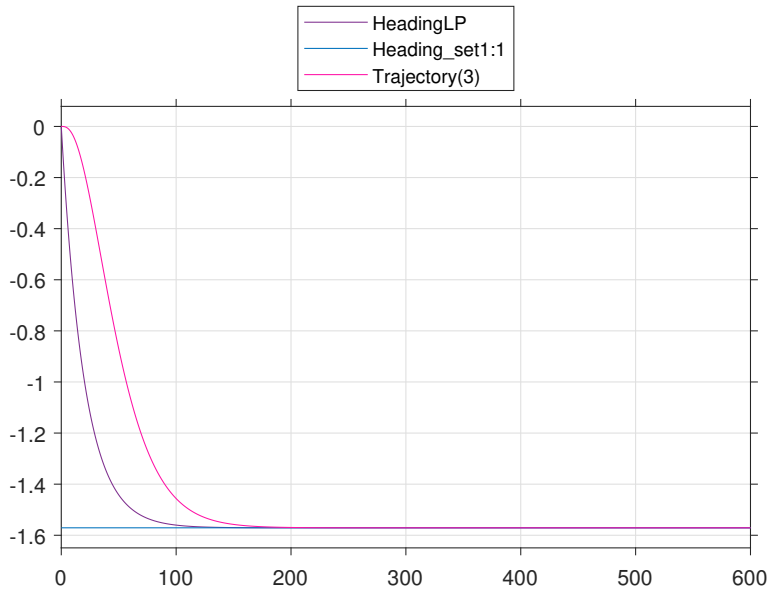


Figure 9: Effect of the reference model in heading

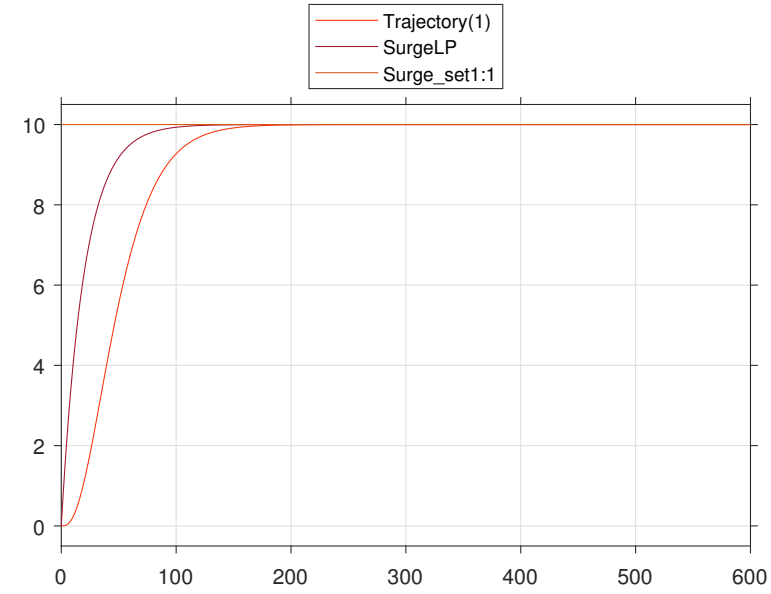


Figure 10: Effect of the reference model in surge

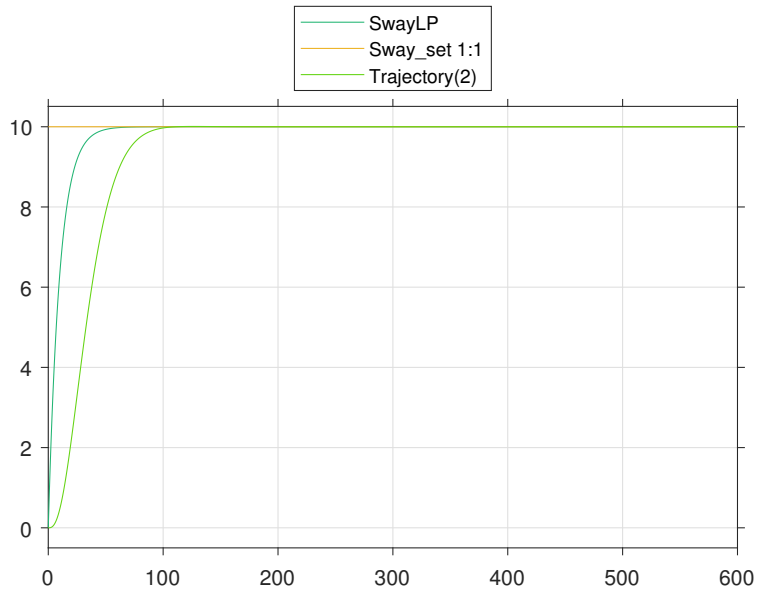


Figure 11: Effect of the reference model in sway

5.3 Changes in the reference model for part 2

Because of the increasing number enviromental forces and the limitations to the thrust, the reference model had to be tuned slower response. This was done by increasing the natural frequencies of the reference model and increasing the damping ratio.
With different simulations the reference model was tuned slower until the the response was following the reference signal as good as possible. The new tuning of the reference model was done in this table.

i	t	ζ	ω
1	20	1	$\frac{\pi}{50}$
2	10	1	$\frac{\pi}{45}$
3	20	1	$\frac{\pi}{50}$

Table 3: Final tuning-parameters of the reference model

5.4 Autopilot

In order for the vessel to follow a list of set-points an autopilot is implemented. The autopilot feeds the reference model with the correct reference signal in accordance with the current set-point. After the vessel has kept a steady position for a selected amount of time the autopilot then updates the reference signal to the next set-point. The autopilot

is created using a MATLAB function block. This allows us to run a custom MATLAB function for each step of the simulation.

The autopilot uses a number of inputs. The vessel position, *eta*, and list of set-points, *eta_set*, are used in a series of conditional statements. These statements checks if the vessel is positioned correctly in surge, sway and yaw within a desired accuracy to the set-point.

If all these three statements are true a timer is started to calculate time at the position. The *clock*-variable tells the current simulation time, and the *start_time*-variable stores the simulation time when the vessel first was in position. The difference between these two variables becomes the time at the position. If the vessel loses the position, the time variable is zeroed. After the position is kept steady for the desired amount of time, the set-point counting variable, p, is incremented by one. The autopilot then outputs the the next reference signal obtained from the next row in the set-point matrix. Should the vessel not keep steady for the desired time the time counting variable is simply zeroed, this variable is also zeroed after the p-variable is incremented.

Figure 12 shows the implementation of the autopilot in simulink.

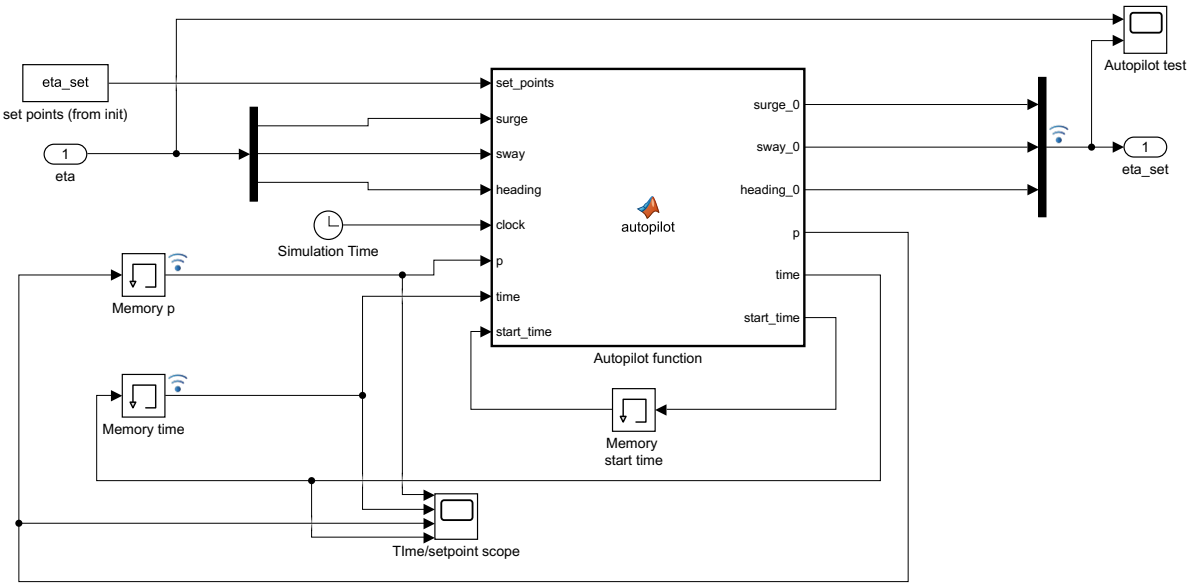


Figure 12: Simulink model overview of the autopilot

5.4.1 Changes in the autopilot for project part 2

The autopilot module was changed to include a code segment for which way the vessel should rotate given a heading reference, such that it chooses the path of shortest rotation.

6 Observer design

For the second part of the project we were tasked with designing a DP control system for a ship where only the position and heading were available with GNSS and compass measurement, and the controller desired forces. Our observer has to estimate both the velocities and position, and the bias caused by non-modeled effects and filter out high frequency motions caused by wave loads.

The inclusion of an observer to our model, will increase our redundancy and help us recreate the signals if needed, through estimation. The wave filtering introduced will also ensure a safer operation due to the filtering of rapid changing disturbances in the wave load, and thus they will be ignored in the controller.

For this task, we chose to derive and implement a model for a extended Kalman filter (EKF), and an nonlinear passive observer (NPO). Theory for the EKF are motivated by section 11.3 in Fossen [3], lecture notes from lecture 7 in *Marine control systems I* [2], and section 7.2.3 in Sørensen [4]. The theory for the NPO is motivated from section 11.4 in Fossen [3], and section 7.2.4 in Sørensen [4].

6.1 Nonlinear passive observer

The motivation behind the NPO was to obtain an observer for a nonlinear system which also could guarantee a global stability result. Another motivating factor was to reduce the time consuming process of tuning the Kalman filter online (Fossen, section 11.4, p 311) [3].

The nonlinear passive observer guarantees global convergence of all estimation errors to zero, hence only one set of observer gains is needed to cover the whole state space. The number of observer tuning parameters are significantly reduced, and the wave filter parameters are directly coupled by the dominating wave frequency. This is derived in the tuning section of the NPO later in this report.

6.1.1 Implementation of the NPO

The theory for the nonlinear passive observer is to assume that the model is deterministic and not stochastic. The stability of the model is determined by a SPR-Lyapunov analysis. Assumptions for the Lyapunov analysis is:

- **A1** Position and heading sensor noise are neglected, that is $\mathbf{v} = \mathbf{0}$
- **A2** Amplitude of the wave induced motion ψ_w is small.
- $\mathbf{R}(\psi) \approx \mathbf{R}(\psi + \psi_w)$

6.1.2 Nonlinear passive observer model

The NPO system equations are given as:

$$\dot{\hat{\xi}} = \mathbf{A}_w\hat{\xi} + \mathbf{K}_1 \tag{30}$$

$$\dot{\hat{\eta}} = \mathbf{R}(\psi_d)\hat{\nu} + \mathbf{K}_2 \tag{31}$$

$$\dot{\hat{b}} = -\mathbf{T}_b^{-1} + \mathbf{K}_3 \tag{32}$$

$$\mathbf{M}\dot{\hat{\nu}} = -\mathbf{D}\hat{\nu} - \mathbf{R}^T(\psi_d) + \tau + \mathbf{R}^T(\psi_d)\mathbf{K}_4 \tag{33}$$

$$\hat{y} = \hat{\eta} + \mathbf{C}_w\hat{\xi} \tag{34}$$

These equations yields the following block diagram, seen in figure 13, which can also be found in Fossen 2011 (p.312, figure 11.8) [3]

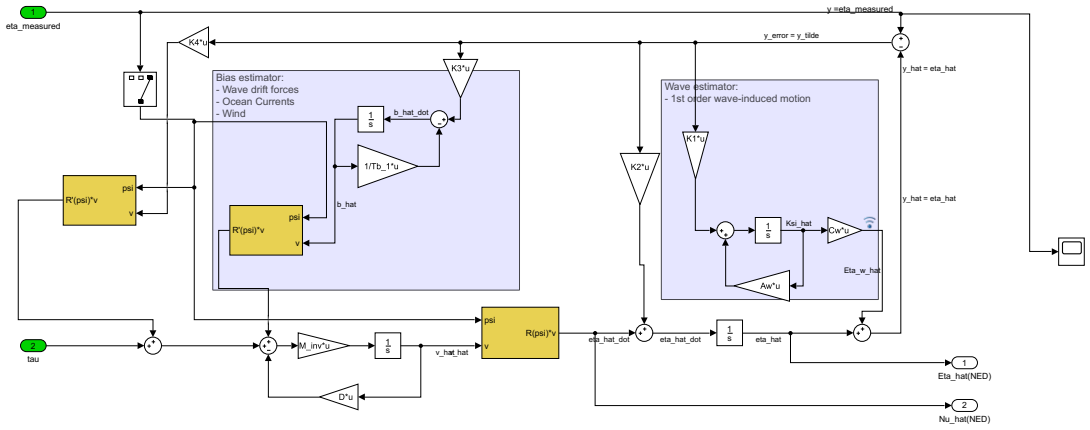


Figure 13: Nonlinear passive observer block diagram implemented in simulink

Here the Observer gain matrices have diagonal structure.
The terms corresponding to the gain matrices are correction terms. The term $\mathbf{K}_1\tilde{y}$ and $\mathbf{K}_2\tilde{y}$ corresponding to a notch and a low pass filter. The term $\mathbf{K}_3\tilde{y}$ corresponds to a correction term for the bias dynamics, while $\mathbf{K}_4\tilde{y}$ is a correction term for the velocity dynamics. The gain matrices are implemented as follows:

$$\mathbf{K}_1 = \begin{bmatrix} diag\{k_1, k_2, k_3\} \\ diag\{k_4, k_5, k_6\} \end{bmatrix} \tag{35}$$

$$\mathbf{K}_2 = diag\{k_7, k_8, k_9\} \tag{36}$$

$$\mathbf{K}_3 = diag\{k_{10}, k_{11}, k_{12}\} \tag{37}$$

$$\mathbf{K}_4 = diag\{k_{13}, k_{14}, k_{15}\} \tag{38}$$

6.1.3 Tuning of the NPO

To ensure passivity of the NPO and to relate the observer to the sea state you have to interpret these tuning rules for $\mathbf{K}_1$, $\mathbf{K}_2$:

$$k_i = -2(\zeta_{ni} - \zeta_i)\frac{\omega_{ci}}{\omega_i}, i = 1, 2, 3 \tag{39}$$

$$k_i = 2\omega_i(\zeta_{ni} - \zeta_i), i = 4, 5, 6 \tag{40}$$

$$k_i = \omega_{ci}, i = 7, 8, 9 \tag{41}$$

Here ζ_i and ω_i, peak frequency and relative damping of wave spectrum, are parameters coming from the WF model. The parameters ω_{ci} and $\zeta_n i$ are filter parameters where $\omega_{ci} > \omega_i$ is the cutoff frequency. $\zeta_{ni} > \zeta_i$ is damping and it was set to 1 as exemplified in the Sørensen (p.166) [4]. The tuning of $\mathbf{K}_3$ and $\mathbf{K}_4$ where implemented as in lecture 7 state estimation (p.78) [2]. The implementation where performed in the following manner.

$$\mathbf{K}_3 = c_1 \mathbf{K}_4 \tag{42}$$

$$\mathbf{K}_4 = c_2 * diag[\mathbf{M}(\mathbf{1},\mathbf{1}), \mathbf{M}(\mathbf{2},\mathbf{2}), \mathbf{M}(\mathbf{3},\mathbf{3})] * diag([c_3 c_4 c_5]) \tag{43}$$

As initial tuning we selected $c_1 = 0.1$, $c_2 = 0.01$, and $c_3 = c_4 = c_5 = 1$.

The final tuning led to the result illustrated in figure 14. Here we we ran the four corner test, with state estimate feedback, thrust allocation and wave loads enabled. This was done to check its robustness. As can be seen, the NPO filtered out most of the rapid wave loads, and the estimate were more smooth than the measurement containing both η and η_w.

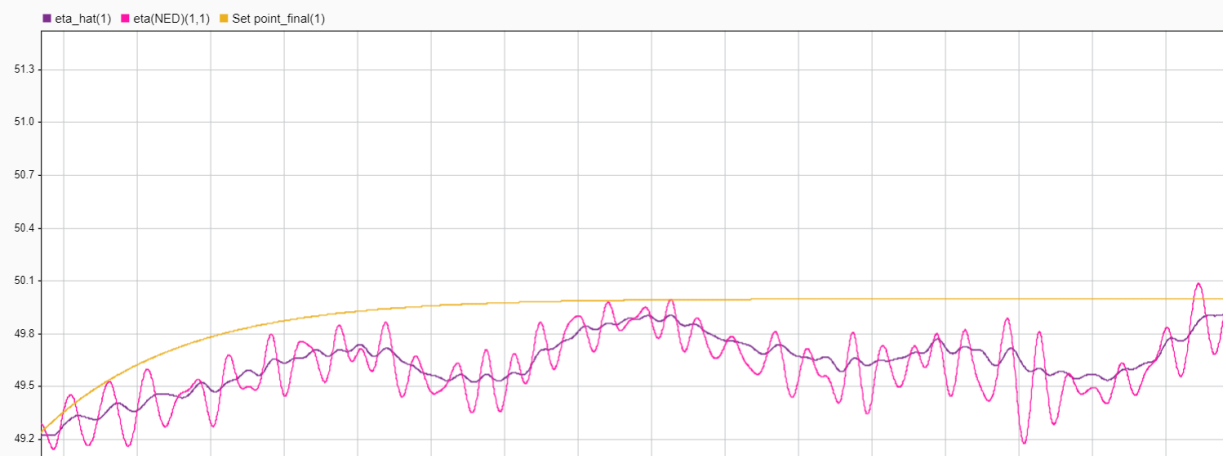


Figure 14: Comparison of filtered estimate $\hat{\eta}_1$ (semi-smooth, purple) and measured position η_1 (high frequency, pink) for the NPO, and position reference (yellow).

6.2 Extended Kalman Filter

The Kalman filter is an efficient recursive filter that estimates the states of a linear and, in the case of the EKF, nonlinear dynamic system (Fossen, T. I 2011, page 296) [3]. It is widely used in sensor and navigation systems, due to it's ability to reconstruct unmeasured states, remove noise from the state estimates, and due to the filter equations ability to behave like a predictor in case of temporarily loss of measurements.

When designing a Kalman filter, one have to assume that the system model is observable. This condition is necessary to guarantee that the estimated states converges to the actual states. Further on, if the system is observable, we can reconstruct the state vector $\mathbf{x} \in \mathbb{R}^n$ from the measurement vector $\mathbf{y} \in \mathbb{R}^m$ and input vector $\mathbf{u} \in \mathbb{R}^p$.

6.2.1 Extended Kalman filter model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w}$$
$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$$

where the state vector $\mathbf{x}$ is given as

$$\mathbf{x} = \begin{bmatrix}\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6, N, E, \psi, b_1, b_2, b_3, u, v, r\end{bmatrix}^T \in \mathbb{R}^{15}$$

The nonlinear vector field $\mathbf{f}(\mathbf{x})$, and the system matrices $\mathbf{B}$, $\mathbf{E}$ and $\mathbf{H}$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{A_w}\xi \\ \mathbf{R}(\psi)\nu \\ -\mathbf{T_b^{-1}} + \mathbf{E_b w_b} \\ -\mathbf{M^{-1}D}\nu - \mathbf{M^{-1}R}(\psi)^{\mathbf{T}}\mathbf{b} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{0}_{6x3} \\ \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} \\ \mathbf{M^{-1}} \end{bmatrix}, \tag{44}$$

$$\mathbf{E} = \begin{bmatrix} \mathbf{E_w} & \mathbf{0}_{6x3} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{E_b} \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix}, \mathbf{H} = \begin{bmatrix} \mathbf{C_w} & \mathbf{I}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix} \tag{45}$$

The submatrices contained in the matrices above, are the same as mentioned in section 3 about the control plant model.

6.2.2 Extended Kalman filter algorithm

State estimation through a Kalman filter is based on an algorithm of several steps. Table 11.3 in Fossen 2011 [3] and section 7.2.3 in Sørensen 2018 [4] illustrate this algorithm. The algorithm is listed below.

The Kalman filter, a stochastic observer, estimates the states by computing a Kalman gain $\mathbf{K}$ that minimizes the trace of the covariance matrix $\hat{\mathbf{P}}(k)$. This is done recursively by following through the steps in table 4, explained as follows:

Discrete-time extended Kalman filter (EKF)	
Design Matrices	$\mathbf{Q}(\mathrm{k}) = \mathbf{Q}(\mathrm{k})^T > 0, \mathbf{R}(k) = \mathbf{R}(k)^T > 0$
Initial conditions	$\bar{x}(0) = \bar{x}_0, \bar{P}(0) = E[(\mathbf{x}(0) - \hat{x}(0))(\mathbf{x}(0) - \hat{x}(0))^T] = \mathbf{P}_0$
Kalman gain matrix	$\mathbf{K}(\mathrm{k}) = \bar{P}(\mathrm{k})\mathbf{H}^T(k)[\mathbf{H}(k)(k)\mathbf{H}^T(k) + \mathbf{R}(k)]^{-1}$
State estimate update	$\hat{\mathbf{x}}(\mathrm{k}) = \bar{\mathbf{x}}(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k)$
Error covariance update	$\mathbf{P}(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]^T + \mathbf{K}(k)\mathbf{R}(k)\mathbf{K}^T(k),$ $\hat{\mathbf{P}}(k) = \hat{\mathbf{P}}^T(k) > 0$
State estimation propagation	$\bar{\mathbf{x}}(k+1) = \mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k))$
Error covariance propagation	$\bar{\mathbf{P}}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{P}}(k)\mathbf{\Phi}^T(k) + \mathbf{\Gamma}(k)\mathbf{Q}(k)\mathbf{\Gamma}^T(k)$
Discrete time quantities found by forward Euler integration	$\mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) \approx \hat{\mathbf{x}}(k) + h[\mathbf{f}(\hat{\mathbf{x}}(k)) + \mathbf{B}\mathbf{u}(k))]$ $\mathbf{\Phi}(k) \approx \mathbf{I} + h\frac{\partial \mathbf{f}(\hat{\mathbf{x}}(k), \mathbf{u}(k))}{\partial \mathbf{x}}\|_{\mathbf{x}(k)=\hat{\mathbf{x}}(k)}$ $\mathbf{\Gamma}(k) \approx h\mathbf{E}$

Table 4: Extended Kalman filter algorithm

1. Initializing the prediction, or priori estimate $\bar{\mathbf{x}}_0$, and the priori covariance matrix $\bar{\mathbf{P}}_0$.
2. Computing the Kalman gain $\mathbf{K}$.
3. Update the estimate $\hat{\mathbf{x}}(k)$ with measurements $\mathbf{y}(k)$.
4. Compute the new covariance matrix $\hat{\mathbf{P}}(k)$ for the updated estimate $\hat{\mathbf{x}}(k)$.
5. Calculate the a posteriori predictions, or project ahead and predict $\bar{\mathbf{x}}(k+1)$ and $\bar{\mathbf{P}}(k+1)$

The sampling time h is set to be equal to 0.1. Dimension on covariance matrices is $\mathbb{R}^{15x15}$, same as with the Jacobian used in the forward Euler. The process and measurement covariance matrices are $\mathbf{Q}(k) \in \mathbb{R}^{6x6}$, and $\mathbf{R}(k) \in \mathbb{R}^{3x3}$.

6.2.3 Tuning of the extended Kalman filter

The tuning of the EKF was an extensive task until we knew were to start. We were informed that a good starting point was to set $\mathbf{E}_b$ to the identity matrix, and to keep both $\mathbf{E}_b$ and $\mathbf{R}$ constant and focus the tuning on the $\mathbf{Q}$-matrix. What matters the most is the ratio $\frac{Q}{R}$, better control over the changes done is gained when only focusing on tuning $\mathbf{Q}$. $\mathbf{R}$ was set to:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \tag{46}$$

$$\mathbf{Q} = \mathbf{I}_{6x6} \tag{47}$$

After the initial tuning, the 3 first diagonal terms, corresponding to the WF in $\mathbf{Q}$, was tuned such that it modelled the WF as good as possible. To achieve this we compared the measured motion $\mathbf{y}$ with the WF motion η_w, and compared this to the η_w from the NPO.

The three last terms the in the Q-matrix corresponds to the response of the bias, which should behave like a critically damped response. Not oscillatory and not to slow. The tuning of $\mathbf{Q}(\mathbf{4},\mathbf{4}), \mathbf{Q}(\mathbf{5},\mathbf{5}), \mathbf{Q}(\mathbf{6},\mathbf{6})$ was initially put to 10^{13}, then reduced until the dimensions of the estimated bias matched the estimated bias from the NPO.

The final values of the EKF tuning can be seen in the equations

$$\bar{\mathbf{x}}_0 = zeros(15,1), \bar{\mathbf{P}}_0 = diag(ones(1,15)), \tag{48}$$

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10e6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10e6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10e6 \end{bmatrix} \tag{49}$$

The tuning of the EKF resulted in a wave load filtering seen in figure 15. As can be seen, the rapid wave loads were filtered out in the estimate and smoothed out in comparison to the measured states. The simulation which led to this plot, was done with state estimation feedback, wave loads and thrust allocation.

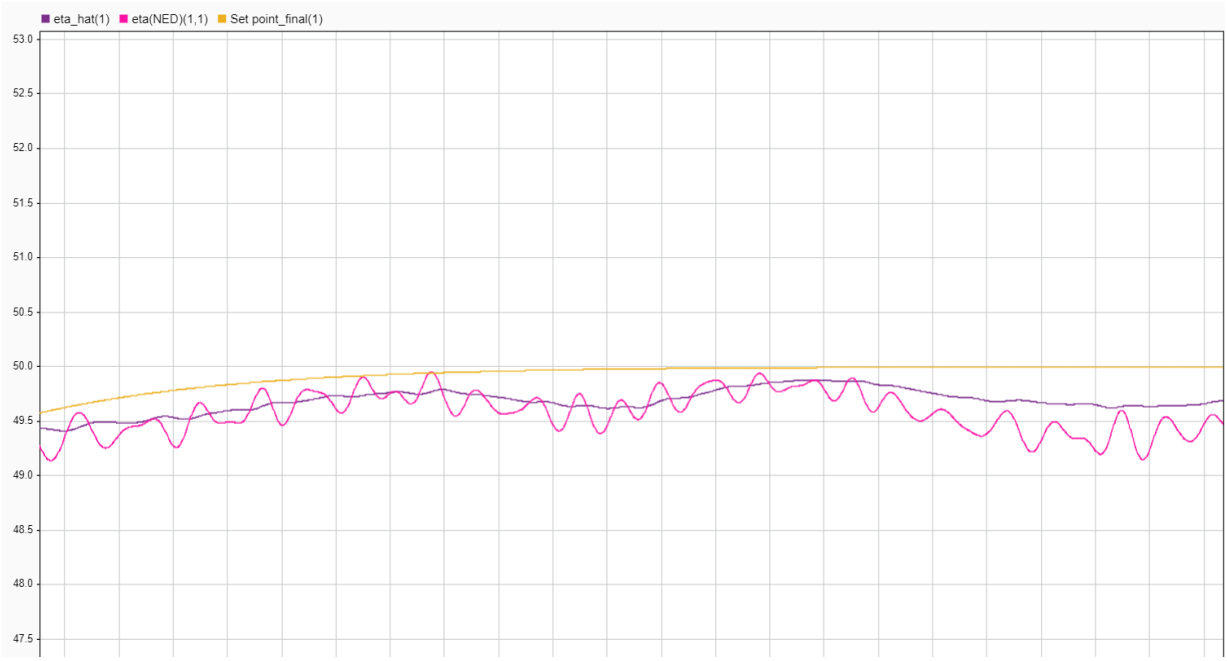


Figure 15: Comparison of filtered estimate $\hat{\eta}_1$ (semi-smooth, purple) and measured position η_1 (high frequency signal, pink) for the EKF with the position reference (yellow)

The extended Kalman filter implementation can be seen in figure 16

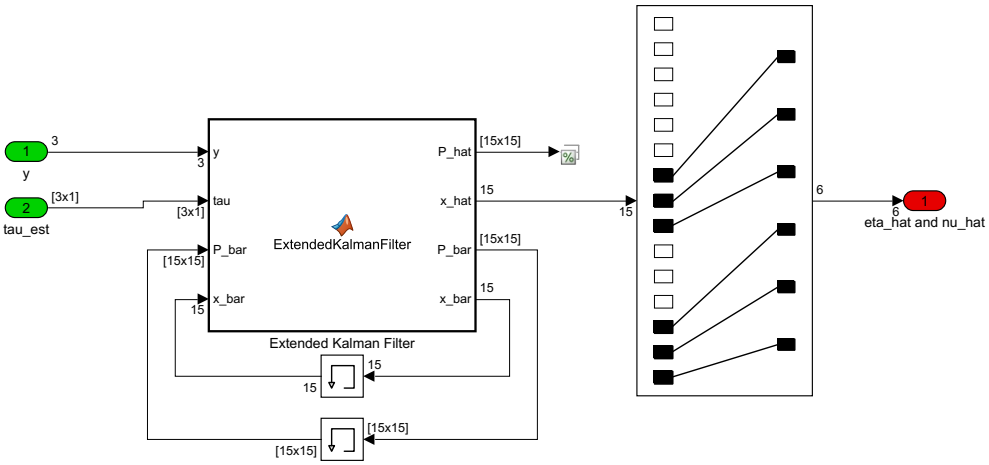


Figure 16: Extended Kalman filter block diagramimplemented in simulink

6.3 Separation principle and motivation for controller swap

The observer stability is based on the separation principle. For the LQG-control we have to measure the position and heading such that the velocities and bias terms can be estimated. This is done under the assumption that the states $\mathbf{x}$ can be replaced with the estimates $\hat{\mathbf{x}}$, which can be calculated by either a stochastic observer (KF/EKF), or a deterministic observer (NPO) (Fossen, p.450) [3]. The stability of the system is also not affected by the pole placement in the observer until the estimates are introduced to the system as feedback.
Fossen, 2011 (section 13.3.9) also performs a case study on nonlinear separation principle for PD controller-observer design. This section states, and demonstrate, that a globally converging observer and a PD control law, plus a nonlinear term of observer bias estimates can be combined to compensate for slowly varying environmental disturbances.
These arguments led to us changing from a PID-controller, implemented in part 1 of the project, to a LQG-controller. The LQG-controller is a form of PD-controller. The theory and tuning of the LQG-controller is written in the section which comprises the controller.

6.3.1 Observer selection

Although the EKF is able to recreate signals through estimation, there is no proof for global asymptotic stability when the system is linearized (Sørensen p.168) [4]. Both Sørensen and Fossen (section 11.4 p.310-311), also states that it is difficult and time consuming to tune the stochastic state estimator with 15 states and 120 covariance equations, and that it is hard to obtain convergence of the bias estimates when using the EKF algorithm in DP.

The difficulties in tuning, and due to the NPO guaranteeing global exponential stability, led to us choosing the NPO over the EKF for simulation five, six and seven. For simulation five, plots are included which illustrates the four corner test for both the NPO and EKF, even though the assignment asks us to choose only one. This is also shown in figure 17. This is done to illustrate that both the estimators are able to performer the task they where designed for. Further motivation for why the NPO surpassed the EKF was a fault in the implementation of the EKF not discovered until late in the project, close to the deadline. Further motivation for the observer choice can be seen in section 10.4.1.

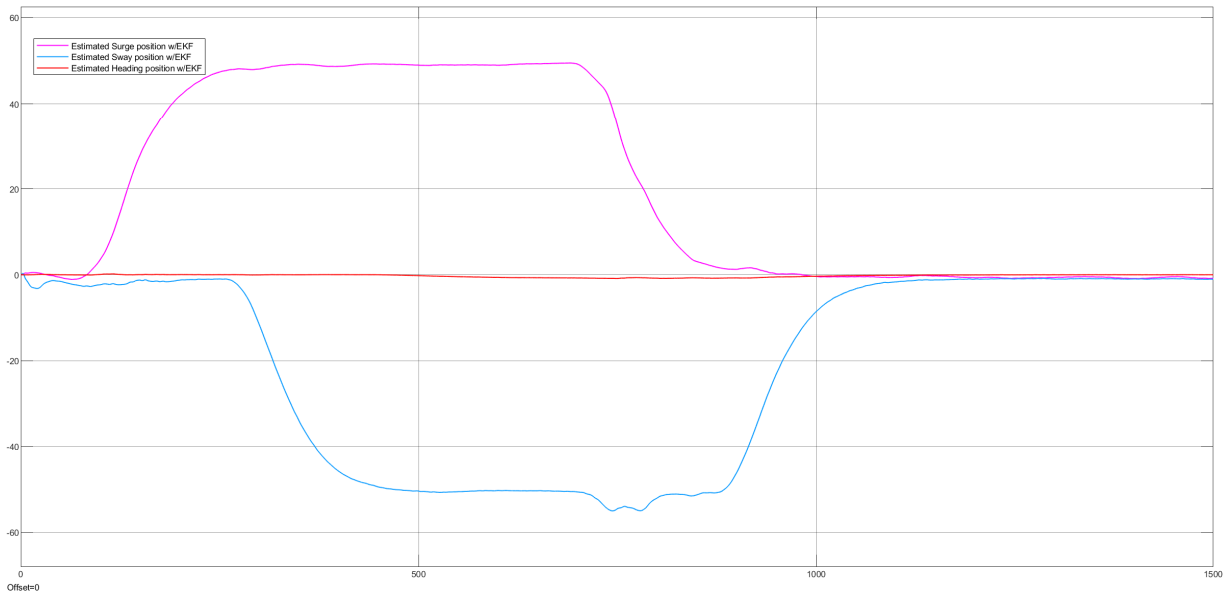


Figure 17: State estimate feedback test of the EKF through a four corner test (all environmental loads active) corresponding to sim 5

7 Controller design

The problem at hand is to create a dynamic positioning system for a vessel, enabling the vessel to either stay in the same position or to follow a reference trajectory when exposed to disturbances such as current, waves and/or wind. To solve this problem, a wide array of controllers may be implemented to combat the environmental disturbances mentioned and a PID-controller is one of them.

7.1 PID-controller

Even though the PID controller is one of the simpler controllers that can solved the DP problem, it does so without any robustness issues. The PID controllers properties regarding how simple it is to implement, its extensive use in different industries and its efficiency in solving the problem, makes it a great starting point for the DP controller. All terms in the controller are deemed necessary. The integral term is needed to remove the stationary error, i.e $e_\infty = \eta_{d\infty} - \eta_\infty$, while the derivative term is needed to introduce damping to the system.

The controller is given in eq. 50, where it's written in the vector reduced form.

$$\tau^{(b)}(1,2,6) = \begin{bmatrix} \tau_u \\ \tau_v \\ \tau_\psi \end{bmatrix} = R^T(\psi)\tau^n = R^T(\psi)[K_p(e) - K_d(\dot{\eta}) + K_i \int_0^\infty [e(x)dx] \tag{50}$$

The reduction of $\tau^{(b)}$ from a 6x1 vector to a 3x1 vector comes from the assumption that roll pitch and heave already are stabilized and are introduced later in the DP controller-block, as constant value 0. This can be seen in in figure 18

As for the other terms, the error between the desired position η_d and η is defined as

$$e = \eta_d - \eta$$

The derivative of η is given as

$$\dot{\eta} = R(\psi)\nu = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^{(T)}$$

And the controller gains are defined as diagonalized matrixes given as K_j where j denotes p, i and d terms.

$$K_j = \begin{bmatrix} K_{ju} & 0 & 0 \\ 0 & K_{jv} & 0 \\ 0 & 0 & K_{jr} \end{bmatrix}$$

The DP controller model with it's associated PID-controller was implemented in Simulink as seen in figure 18 and figure 19

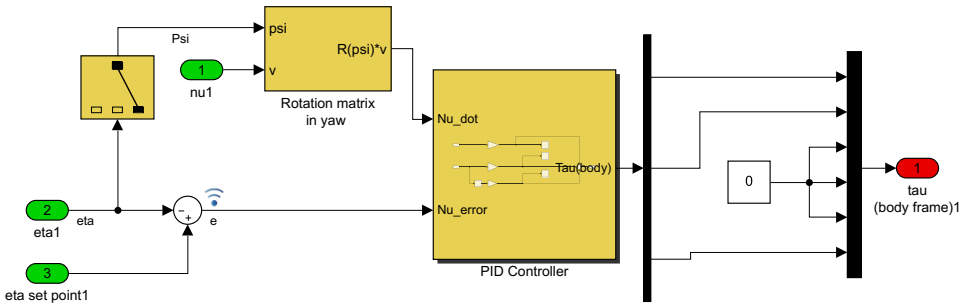


Figure 18: Simulink model overview of the DP system controller

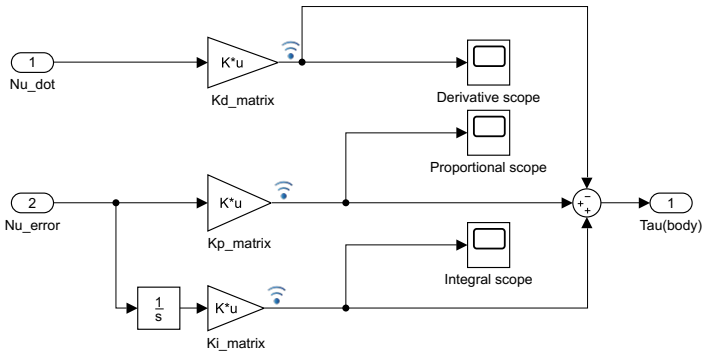


Figure 19: Simulink model overview of the PID-controller

7.2 Tuning of system and controller gains.

The chosen controller was tuned roughly by guesstimation and a somewhat bastardized Ziegler-Nichols method. The overall aim of the tuning part is to ensure a stable system, i.e ensuring that all of our poles has a negative real value in the closed loop.
The tuning was done by first by increasing the proportional terms until oscillations in η was achieved, then lowered a bit. Following this, the derivative was increased to introduce some damping to the system, and to smoothen the responses. Lastly the integral terms were tuned until the remaining stationary deviancy was eliminated within reasonable bounds.

Following this, a phase with more individual tuning was done. This was to done to ensure a faster response, and to make sure that the vessel did not overshoot the reference value. At the start of this phase, the measurements undershot the reference value, and the response was decaying slowly. Further discussion of this problem hinted that we could have had a zero in the right half plane, and that one of the poles were close to negative zero. The positive zero is what could have given the inverse response at the start of the simulation and the very small pole (approximately $\lambda = -0.0088$) is what gave the slow response. The poles of the system was found by equation eq: 51 and eq: 52:

The Laplace transform of the system looked like this:

$$\frac{e}{u}(s) = \frac{s}{s^2 + \frac{K_p}{K_d}s + \frac{K_i}{K_d}} \tag{51}$$

The poles/eigenvalues, which should have a negative real part, then becomes:

$$_2\lambda_1 = -\frac{K_p}{K_d} \pm \sqrt{(\frac{K_p}{K_d})^2 - 4\frac{K_i}{K_d}} \tag{52}$$

After noting the issue with the slow and inverse response, further tuning was done. This yielded the PID-gains listed in table 5.

	Surge	Sway	Yaw
K_p	$4 \cdot 10^5$	$2 \cdot 10^5$	$1 \cdot 10^7$
K_i	$3 \cdot 10^3$	$2 \cdot 10^3$	$1 \cdot 10^3$
K_d	$6 \cdot 10^6$	$5 \cdot 10^6$	$8 \cdot 10^4$

Table 5: Controller gains

7.3 LQG-controller

For part two of the project, we chose to discard the PID-controller in the prior subsections, and implement a LQG in its place. The LQG(Linear quadratic Gaussian) feedback controller is a way of implementing a P, PD or PID controller by minimizing the function:

$$\min_{e,\tau_{pd}} J = \min_{e,\tau_{PD}} \lim_{T\to\infty} \int_0^T \mathbf{e}^T\mathbf{Q}\mathbf{e} + \tau_{pd}^T\mathbf{P}\tau_{pd} \tag{53}$$

Its called quadratic because every vector is multiplied by each side of the matrix in each term in the equation. In this equation the state error weighting matrix $\mathbf{Q}^T = \mathbf{Q} \geq 0$ and the control input weighting matrix $\mathbf{P}^T = \mathbf{P} \geq 0$. $\mathbf{e} \in \mathbb{R}^6$, $\mathbf{Q} \in \mathbb{R}^{6x6}$ and $\mathbb{R}^{3x3}$

The weighting of the cost function parameters $\mathbf{Q}^T$ and $\mathbf{P}^T$ determines the prioritisation of the different controller aspects. A reasonable start point for the weights can be found with Bryson's rule.

$$\mathbf{Q_{ii}} = \frac{1}{max\ acceptable\ value\ of\ e_i^2} \tag{54}$$

$$\mathbf{P_{jj}} = \frac{1}{max\ acceptable\ value\ of\ u_j^2} \tag{55}$$

These equations corresponds with eq. (53) in the following manner.

$$\min_{x,u} J = \Sigma_{l=1}^n \int_0^\infty q_i x_i^2(t)dt + \Sigma_{j=1}^m \int_0^\infty p_j u_j^2(t)dt \tag{56}$$

The $\mathbf{G}$ matrix in the output of the controller $\mathbf{u} = \mathbf{Gx}$ is determined by the LQR algorithm. This matrix is found by Matlab, and the use of the command *G = lqr(A, B, Q, R).*

7.4 Tuning the LQG-controller

In tuning the LQG controller you can use the certainty equivalent priciple and tune the controller without the estimator just the real states from the model block and then it will in theory work with the estimator afterwards. First the Q and the P matrix where sat to the identity matrices. A simulation with current was ran and vessel was highly drifting. To again tune the Bryson's rule was used and from thrust allocation it was known that the max thrust was of order 10^5 and mass was of order 10^6. Then used $\frac{1}{(10^6)^2}$ in every diagonal term in the P matrix. In the Q matrix the max deviation in position was sat to 0.3 meters in surge and sway and the max deviation in yaw was sat to 5 degrees. In the velocity a max deviation in linear velocity was sat to 5 m/s and 81 degrees/s. The velocity errors was sat high because they where thought of as less importance. This was done for a first initial tuning. The final final tuning of $\mathbf{Q}$ and $\mathbf{P}$ were found as follows:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{5^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{5^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{(\frac{1}{3})^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{(\frac{\sqrt{(10)}}{10})^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{(\frac{\sqrt{2}}{20})^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{0.0174^2} \end{bmatrix}, \tag{57}$$

$$\mathbf{P} = \begin{bmatrix} \frac{1}{(2 \cdot 10^4)^2} & 0 & 0 \\ 0 & \frac{1}{(2 \cdot 10^4)^2} & 0 \\ 0 & 0 & \frac{1}{(2 \cdot 10^6)^2} \end{bmatrix} \tag{58}$$

The LQG gain $\mathbf{K}$ was found by using the MATLAB-function K = lqr(A,B,Q,P), where A, and B are the state space matrixes. The function lqr(A,B,Q,P) uses the algebraic riccatti equation to solve for a K which minimizes the cost function J [1].

Implementation of the LQG controller can be found in the DP block, and is illustrated in figure **??**

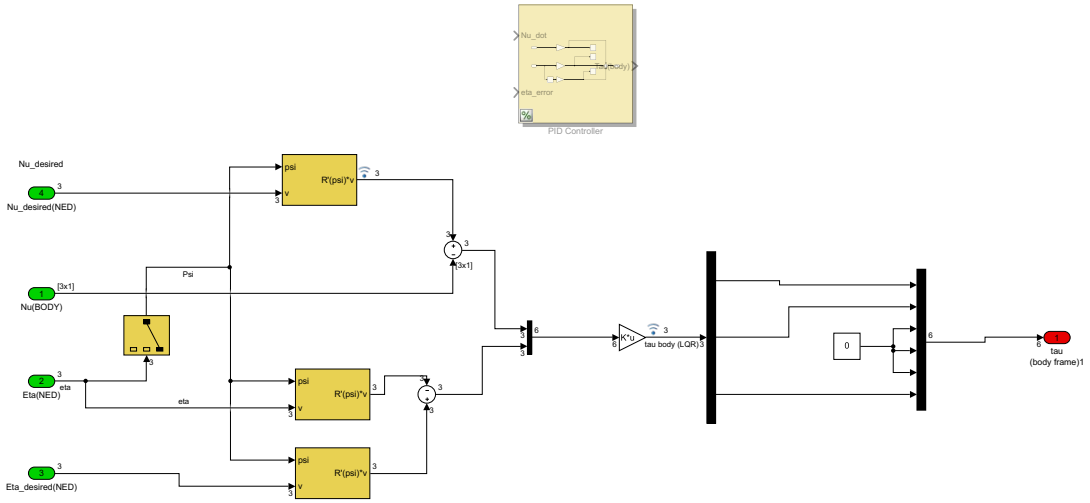


Figure 20: Simulink model overview of the LQG-controller

8 Thrust Allocation

An important part of a DP-system is the thrust allocation (TA) algorithm. The purpose of this algorithm is to manage the thrust and rotation of each propulsion unit in order to maneuver the vessel as the controller dictates.
Our vessel is equipped with 5 thrusters. Two tunnel thrusters are placed in the bow of the boat to give ample yaw momentum. Three azimuth thrusters are free to rotate around the z-axis with an angle α. This gives the azimuths ability to provide forces in both x and y direction. Since our number of thrusters is greater than the number of degrees of freedom ($5 > 3$), hence our system is over-actuated giving the possibility to optimize the system.

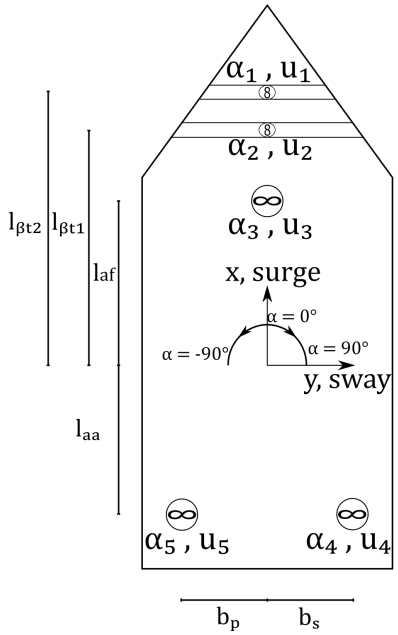


Figure 21: Thrust configuration

Theory for the thrust allocation algorithm are motivated by Fossen 2011 (section 12.3.1) [3]. Equation (59) describes the relationship between the actuator forces and moments from the controller, τ, with the forces and angles to each thruster, u and α respectively.

$$\tau = \mathbf{T}(\alpha)\mathbf{K}\mathbf{u} \tag{59}$$

Here $\mathbf{T}(\alpha)$ is the thrust configuration matrix dependent on the azimuth angles α. $\mathbf{K}$ is the diagonal force coefficient matrix containing the force attributes from each thruster. $\mathbf{u}$ is the control vector for the thrusters, giving each thruster the necessary force to maneuver the ship as desired. τ is the output vector from the controller. The matrices and vectors for our system can be seen in equation (60) - (64) using numbering in agreement

with figure 21.

$$\mathbf{T}(\alpha) = \mathbf{T_{3x5}} = \begin{bmatrix} 0 & 0 & cos(\alpha_3) & cos(\alpha_4) & cos(\alpha_5) \\ 1 & 1 & sin(\alpha_3) & sin(\alpha_4) & sin(\alpha_5) \\ l_{bt1} & l_{bt2} & l_{af} & -l_{aa}sin(\alpha_4) - b_s cos(\alpha_4) & l_{aa}sin(\alpha_5) - b_p cos(\alpha_5) \end{bmatrix} \tag{60}$$

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_\Psi \end{bmatrix} \tag{61}$$

$$\mathbf{K} = diag[K_1\ K_2\ K_3\ K_4\ K_5] \tag{62}$$

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} \tag{63}$$

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} \tag{64}$$

8.1 Extended thrust allocation

In order to obtain the output from the thrust allocation algorithm (TAA) proposed above a nonlinear optimization problem must be solved, due to the nonlinear part of equation (60) dependent on α. To avoid this the azimuth thrusters can be split into two fixed thrusters, one in surge and one in sway direction. Our extended, non angle dependent, thrust matrix then becomes a 3x8 matrix.

$$\mathbf{T}_e = \mathbf{T}_{e\,3x8} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ l_{bt1} & l_{bt2} & 0 & l_{af} & b_s & l_{aa} & b_p & l_{aa} \end{bmatrix} \tag{65a}$$

$$\begin{matrix} T_1 & T_2 & A_{fwd\,1\,X} & A_{fwd\,Y} & A_{stern\,SB\,X} & A_{stern\,SB\,Y} & A_{stern\,P\,X} & A_{stern\,P\,Y} \end{matrix} \tag{65b}$$

We can now calculate the $\mathbf{u}_e$ using equations (59) and (65).

$$\mathbf{u}_d = \mathbf{K}^{-1} pinv(\mathbf{T_e})\tau_\mathbf{e} \tag{66}$$

Here *pinv(T_e)* is the Matlab function for the pseudo inverse of T_e. With $\mathbf{u}_d$ we can calculate thruster control vector $\mathbf{u}$ and the angles, α, for the non-split thruster layout.

$$\mathbf{u}_{azimuth\,i} = \sqrt{u_{e\,i}^2 + u_{e\,i+1}^2} \tag{67}$$

$$\alpha_{azimuth\,i} = atan2(\frac{u_e\ i\ sway}{u_e\ i\ surge}) \tag{68}$$

8.2 Implementation of thruster allocation

The implementation of the extended thrust allocation algorithm as described above can be seen in appendix I. In addition to the algorithm proposed above some additional steps was needed in order to assure robust performance.

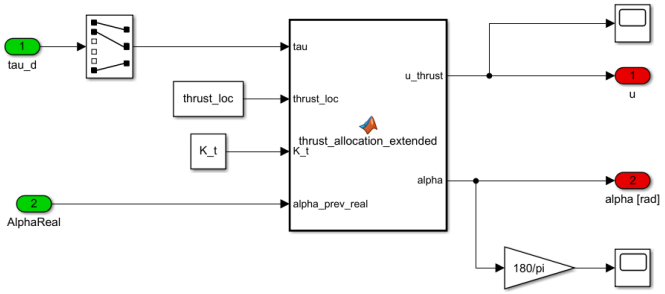
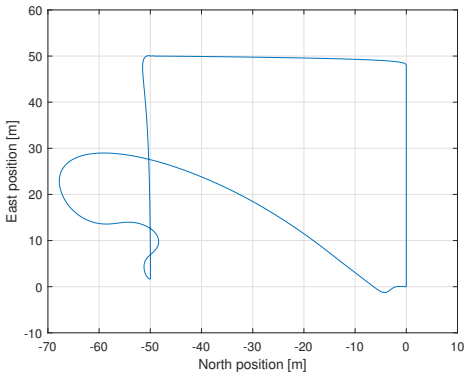
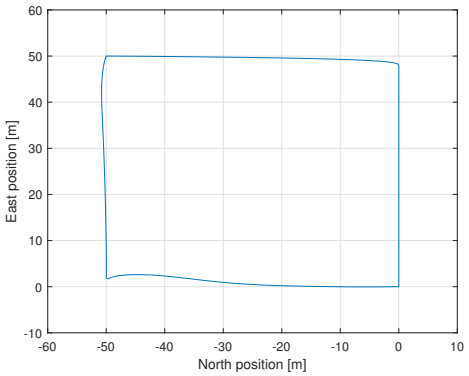


Figure 22: Simulink model overview of the thrust allocation block

In previous iterations of our thrust allocation algorithm the system preformed well in the four corner test until the thruster needed make a half turn in order to generate thrust for backing.
The problem with the basic extended thrust allocation algorithm in series with the premade thruster dynamics block, handling saturation and force limiting. With our basic extended thrust allocation algorithm the control vector **u** is fully saturated after holding position in one direction, when the azimuths then changed angles, the force was still high causing the ship maneuver heavily off course, before the azimuths had rotated into position.



(a) Unfinished TAA, 4 corner test



(b) Final TAA, 4 corner test

Our solution to this problem was to limit the thrust from the azimuths as long as the actual angle of the azimuths is far from the angles demanded by the thrust allocation algorithm.

8.3 Forbidden sector

A simple algorithm to limit the 2 stern azimuths from moving in the forbidden sectors was implemented. The algorithm checks if the angles are within the forbidden sector and changes the angle to the closes border of the sector.
A problem with this implementation is that the thruster angle do not corresponds to with the thruster force anymore. This will lead to an unwanted thrust in surge, sway and yaw since the force is applied in another direction than first intended. We tried to implement a new calculation of the thrust vector u in order to counteract this problem, but this implementation was not successful.

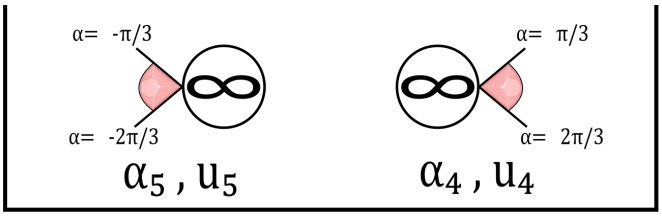


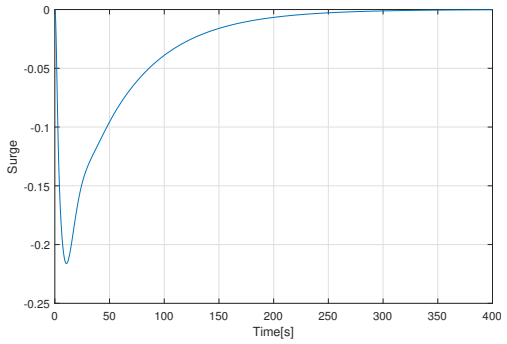
Figure 24: Forbidden sectors for stern azimuth thrusters

9 Simulation results part I

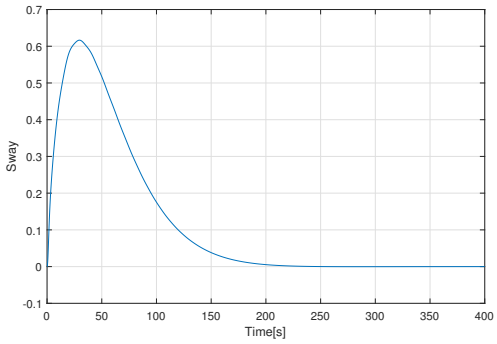
In order to validate our control system and tuning parameters a series of tests were performed. These simulation runs tested different aspects of our system, and on several occasions these runs led to major changes in the system and parameters.

9.1 Simulation 1

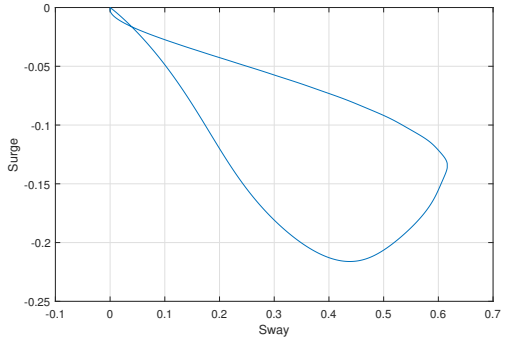
Simulation 1 is about checking the DP station-keeping capabilities of the control system by holding the position (0,0,0) with a current of 0.5 [m/s] from north-west. The result of this simulation is presented below in figure 25 and 26.



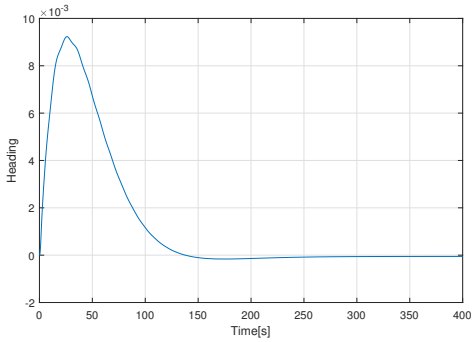
(a) North position



(b) East position



(c) North and Earth position

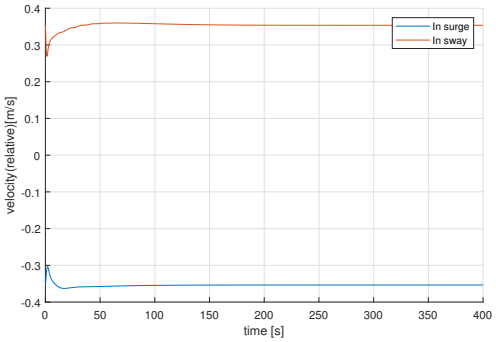


(d) Heading

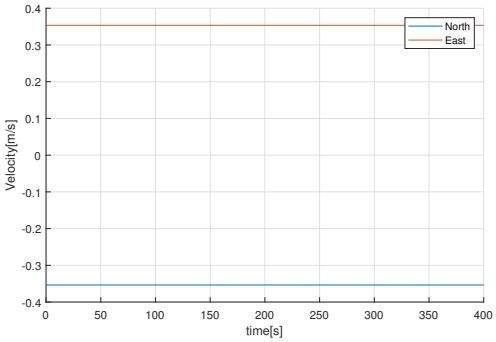
Figure 25: Simulation 1

9.1.1 Discussion simulation 1

In this simulation a constant current is dragging on the vessel. The current hits the vessel at a 135°angle in NED-frame, and as a result the vessel is pushed in both surge and sway. The heading deviates only 0.51°off the reference before the controller adapts to the current. The controller effectively removes the effect from the current and leaves no stationary deviation. It takes the controller around 150 seconds to adapt to the current forces. For a 80 meter vessel this seems acceptable.



(a) Current in body-fixed xyz frame

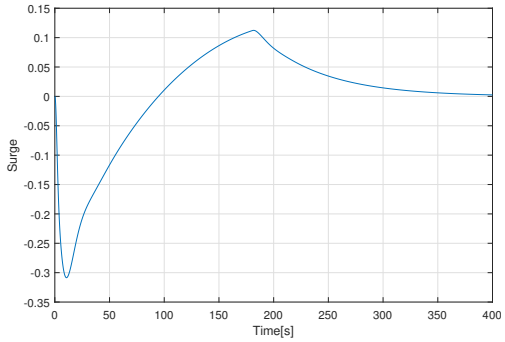


(b) Current in NED-frame

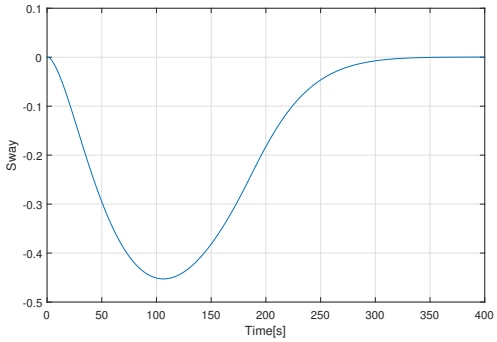
Figure 26: Current in body-fixed and NED-frame

9.2 Simulation 2

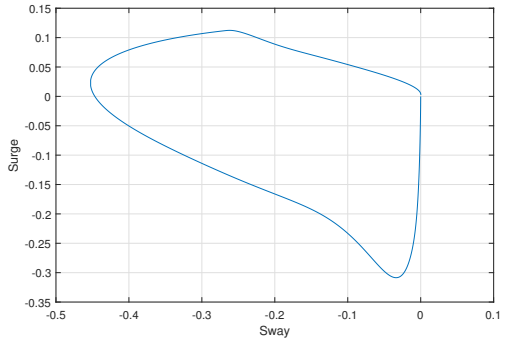
In simulation 2 the current heading is varying linearly from 180 to 270 which is from north to east. The DP set-point is set to (0,0,0) with a current velocity of 0.5 [m/s]. The position in surge, sway and heading is plotted individually in figure (27). To verify that the current is varying as excepted, the plots of the current velocities in NED-frame and body-frame can be seen in figure (28).



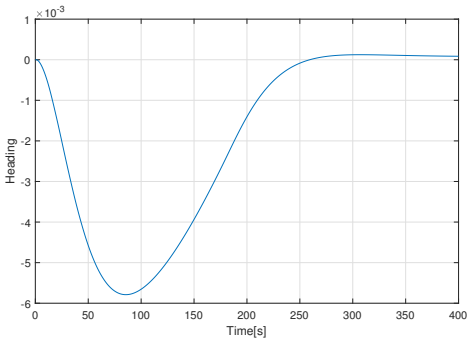
(a) North position



(b) Earth position

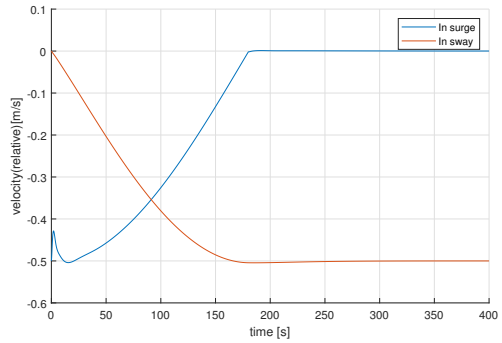


(c) North and Earth position

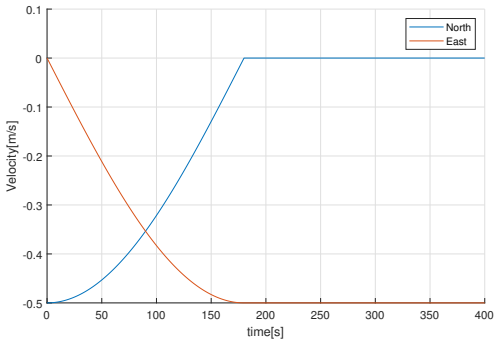


(d) Heading

Figure 27: Simulation 2 - Positions and rotation



(a) Current in body-fixed xyz frame



(b) Current in NED-frame

Figure 28: Simulation 2 - Current in body fixed and NED frame

9.2.1 Discussion simulation 2

Simulation 2 is a test of a varying disturbance where the current direction is varying from a course of 180°to 270°. This is testing the controller ability to adapt the environmental forces. It is important that the integral term builds up to keep the stationary deviation as small as possible. The maximum drift is around 0.5 m in sway and 0.3 m in surge which is good.

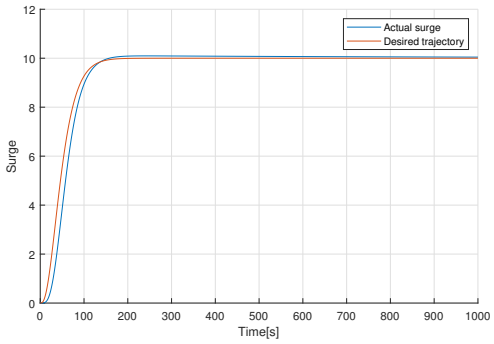
The controller is fast and accurate, it does not overshoot that much. It was briefly discussed if the controller had fulfilled it's purpose, since there is some overshoot in surge and sway, but it was concluded that the overshoot was due to the current changing over time, and not the controller failing on fulfilling its purpose.

Other than this, the controller does what it is supposed to do, and seems to be implemented correctly. This can be seen in the plots related to simulation 2, where it is shown that the current is coming from north and turns east, and the vessel first drifts south and then west.
The current is also plotted to check that it's actual turning like it should and how the relative current directions are in body-fixed frame.

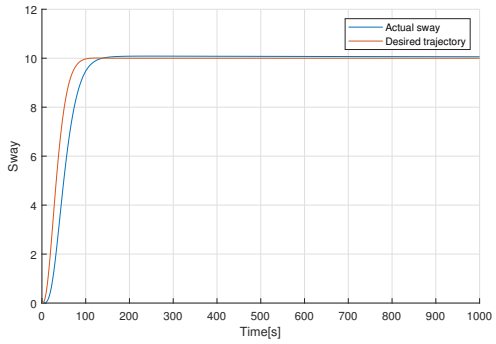
9.3 Simulation 3

Simulation 3 is about checking how the vessel responds to a set-point of (10,10,-$\frac{\pi}{2}$) from initial position of (0,0,0) with and without reference model. Both simulations is preformed without environmental forces which in this case is without current forces.

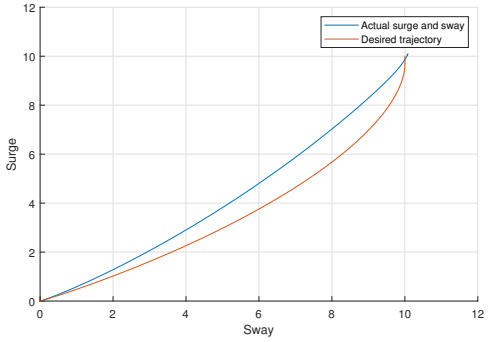
9.3.1 With reference model



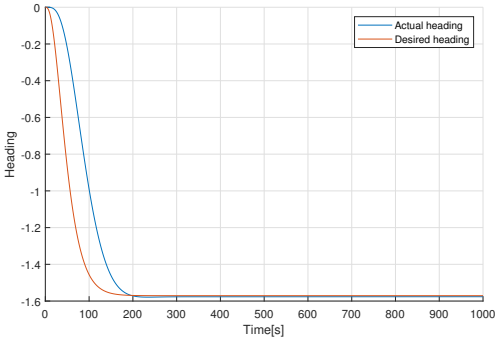
(a) North position



(b) Earth position



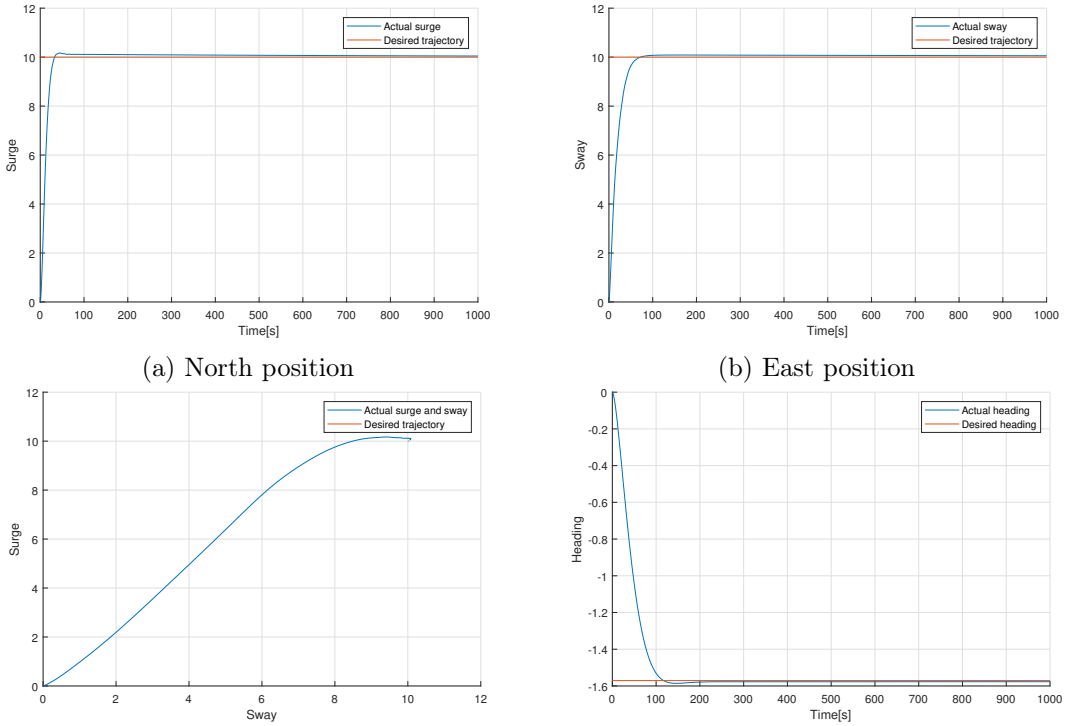
(c) North and Earth position



(d) Heading

Figure 29: Simulation 3 with reference model

9.3.2 Without reference model



(a) North position

(b) East position

(c) North and East position

(d) Heading

Figure 30: Simulation 3 without reference model

9.3.3 Discussion simulation 3

Simulation 3 was about checking the impact of the reference model on the response. By using the same parameters as in simulation 1 and 2 one observes a much steeper and faster response without reference model. This is probably due to the lack of force limits in our model. In figure (29) the response is following the reference, which means that the reference is calibrated well for this model. There is no overshoot which also means our controller is fulfilling its purpose. Figure (30) is the simulation without reference model. Here we see an overshoot, not much, but this means that the error is jumping from 0 to 10 very fast which then results in a jumping control signal. However the response is much better than expected.

9.4 Simulation 4

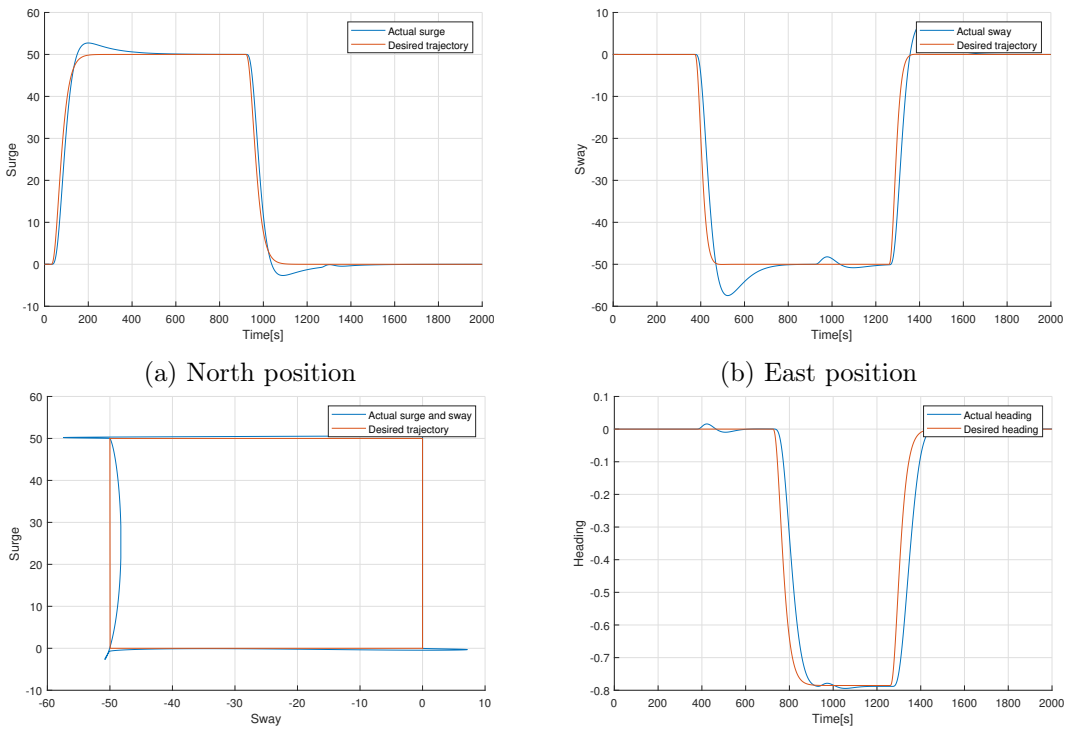
Simulation 4 is about a DP 4 corner test without environmental forces. To preform this simulation, we made an autopilot function that checks the position and sets a new set-point if some criteria is achieved. The autopilot function is closer described in section 5.4, and the code can be seen in Appendix A.

For the following simulations, parameters for the autopilot was set with a high degree of precision in order to fully test the controller performance. Parameters are presented in table 6.

Accuracy sway & spacesurge	1 $[m]$
Accuracy heading	1 $[deg]$
Time at destination	30 $[sec]$

Table 6: Autopilot parameters

9.4.1 With same parameters as in simulation 1, 2 and 3



(a) North position

(b) East position

(c) North and East position

(d) Heading

Figure 31: Simulation 4: with same parameters as in sim 1, 2 and 3

Figure (32) shows the autopilot variables when using the same tuning parameters as in simulation 1, 2 and 3.

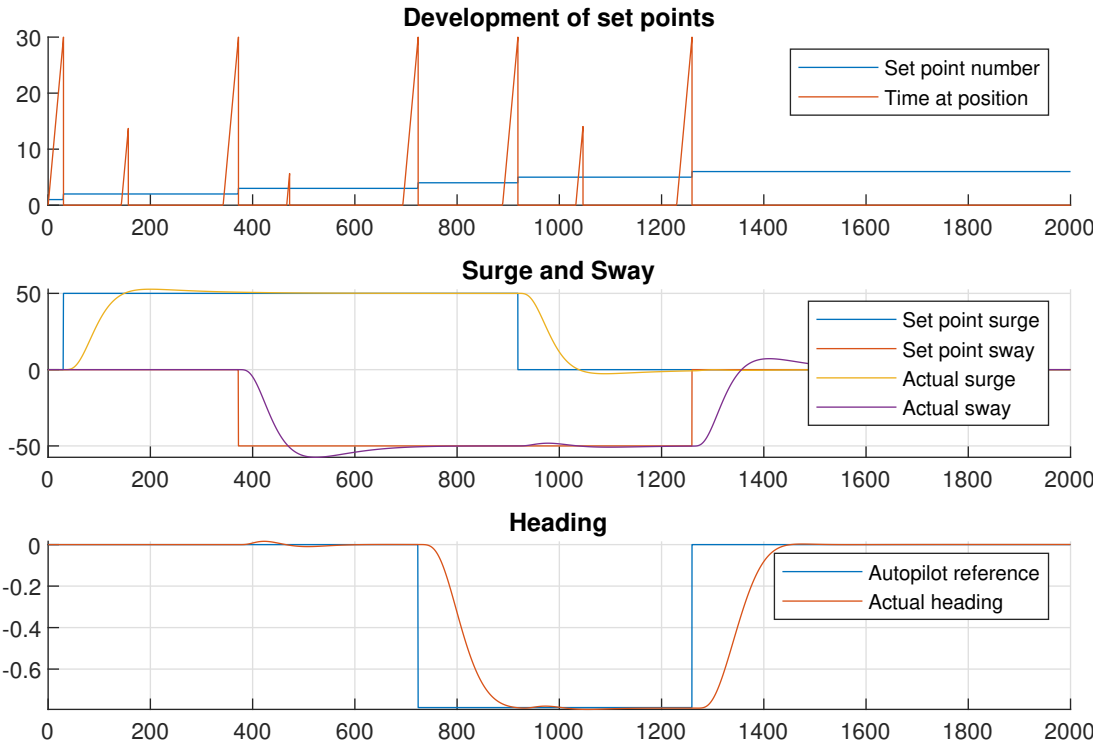
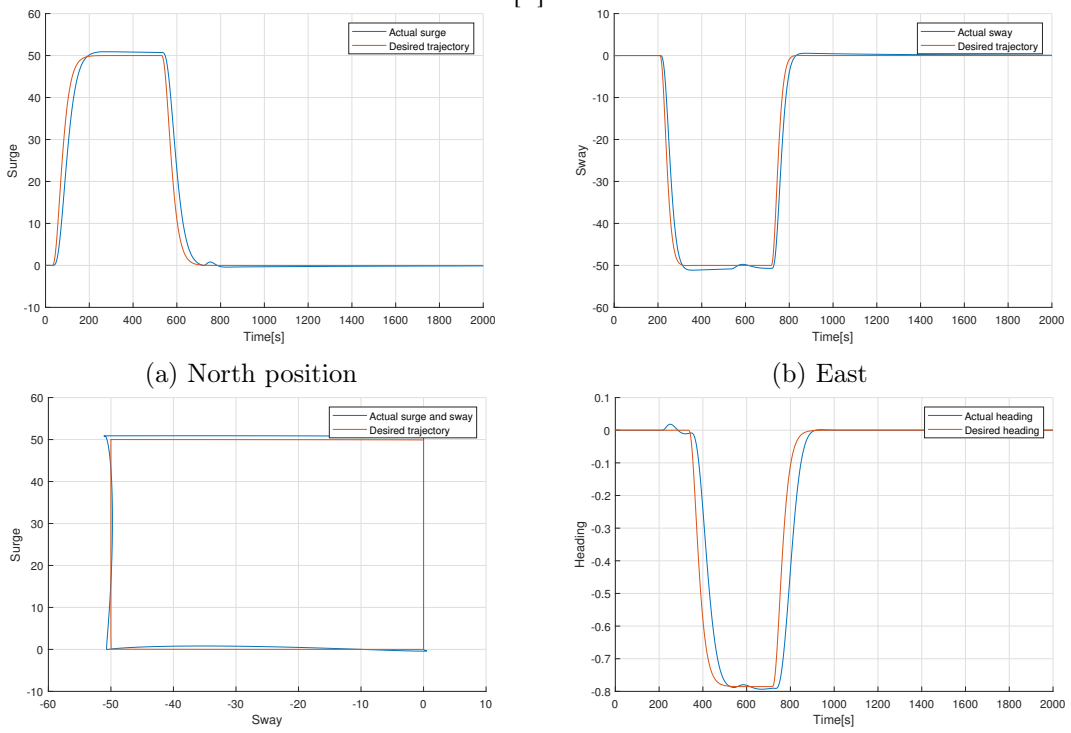


Figure 32: Autopilot variables

9.4.2 With a new set of parameters

[h]



(a) North position

(b) East

(c) Surge and sway

(d) Heading

Figure 33: Simulation 4 with a new set of parameters

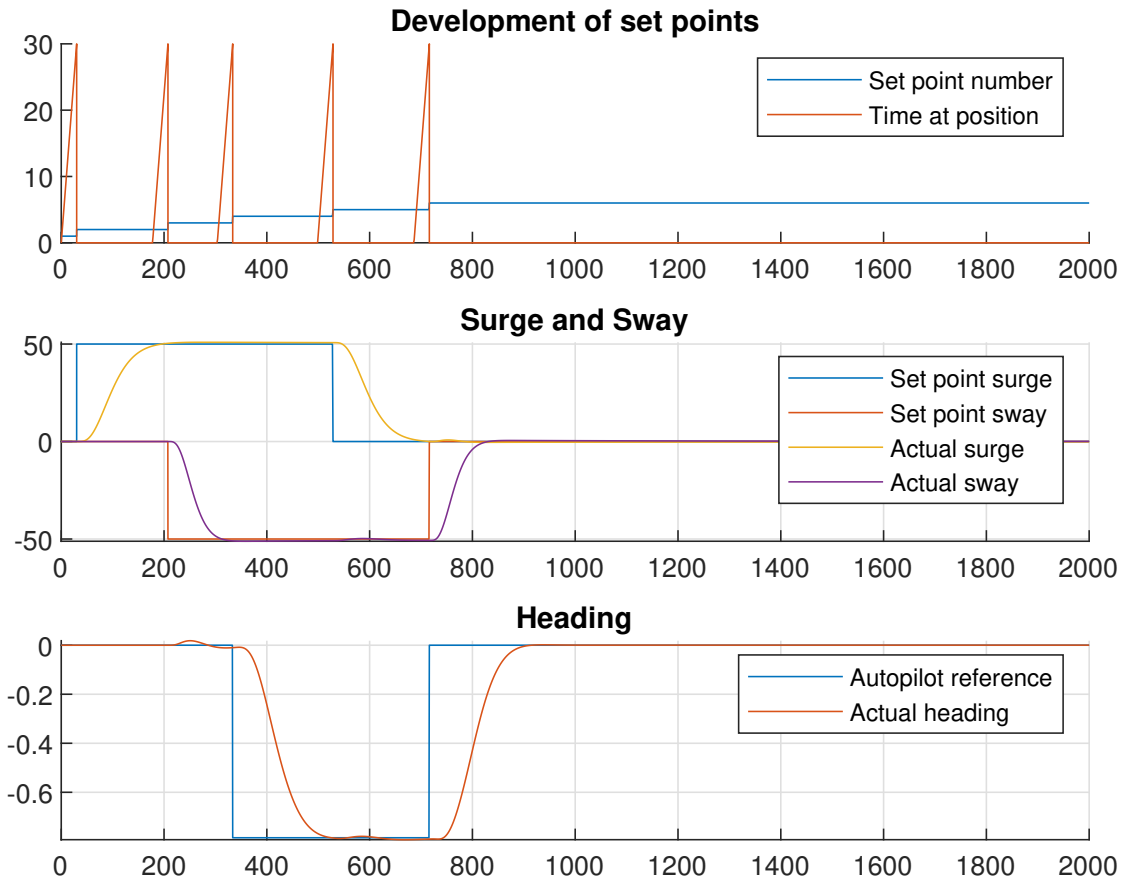


Figure 34: Autopilot variables

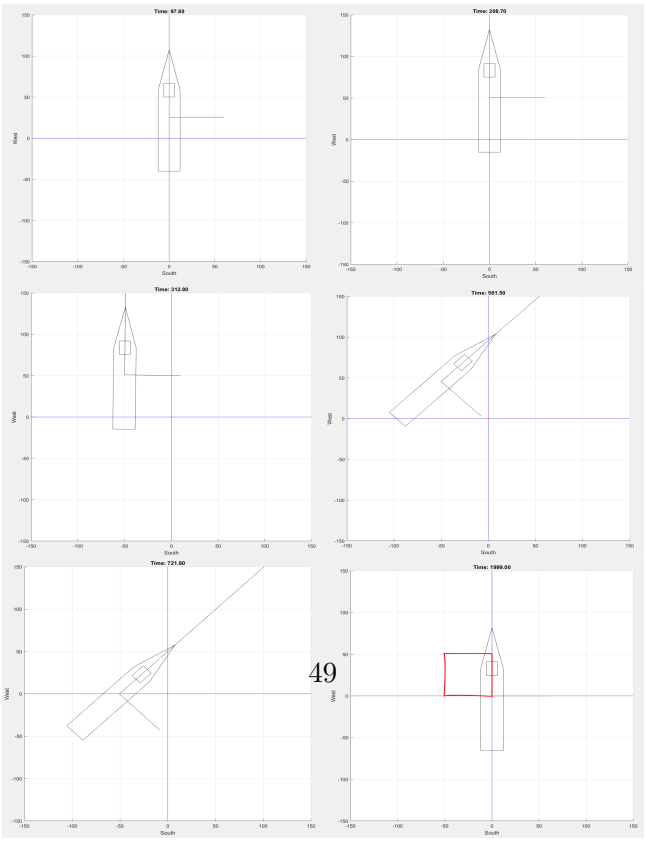


Figure 35: Simulation 4: Plot of the vessel position through the different set-points

A function was made in order to visualize and troubleshoot the vessels position and response. The plot in figure 35 animates the vessel in earth fixed reference frame. The plot function is found in the appendix A.3.

9.4.3 Discussion simulation 4

In the first plot in figure 34 we can see the development of the set point number as the vessel fulfills the time criteria at the position. From the second plot we can see how the time variable starts counting once the vessel has caught up with the reference. Comparing the first plot with the second and third we can also see how the reference value changes with changing set points.
The tuning parameters are changed in order to get a response which does not overshoot the reference. In this simulation a overshoot seems to make the system slower because the vessels has to return in order to hold the position as desired. With the new set of parameters the vessel is holding the position much closer to the reference than with the old parameters.+

9.5 Discussion

The ship was tuned such that it should be robust enough to withstand each simulation individually without the need of new tuning for each simulation. In simulation 3 and 4 the controller and reference model was robust enough to not overshoot and hold within a range of under 1 meter in surge and sway and under 30 degrees in heading.

However the simulation 4 has a small overshoot with the same tuning as for simulation 1, 2 and 3. The tuning parameters was then modified which gave the results shown in section 9.4.2. One can see that the response from 31 is approximately 600 seconds slower because it does not overshoot the reference hence the autopilot precision parameters is harder to reach. This means that the autopilot needs only one try to hold the position for 30 seconds. This is easy to see by comparing figure (34) and (32).

It is noted that the content of the report from Project part 1 will only contain the relevant sections and does not need to have the conclusion section. However, Project part 1 report should contain the discussion(s) to the simulation(s) performed for Part 1. The report after Project part 2 will contain all sections and the results from simulations in Project part 2.

10 Simulation results part II

A total of seven simulations was performed in this part of the project. The aim of the first five simulations was to validate the performance and correctness of our system. The last two simulations was performed in order to enhance our understanding of both the system implemented and marine systems in general

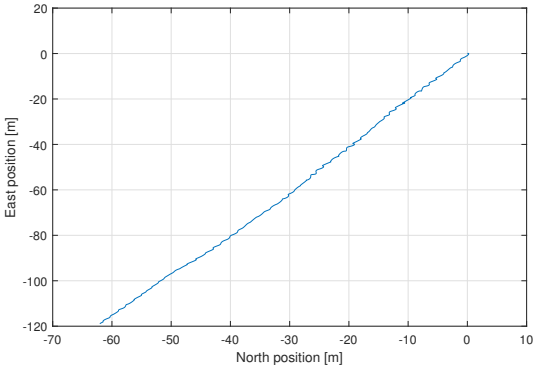
10.1 Simulation 1 - Environmental loads

In the first simulation the effects from the environmental forces on the vessel was tested. Non of the thrusters was engaged which allowed the model to drift.

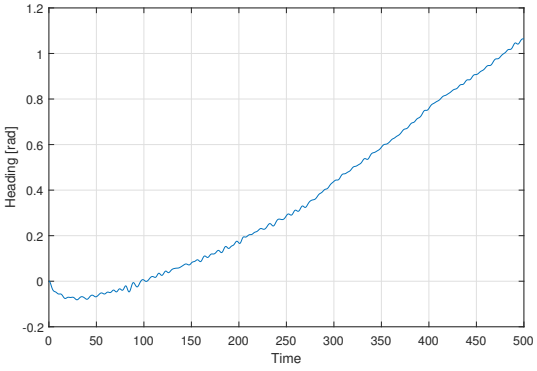
	Heading	$\bar{V}$ **[m/s]**	**Variation**
Wind	South (π [rad])	10	$\pm$ 5 $^\circ$
Current	West (3/2 π [rad])	0.2	$\pm$ 5 $^\circ$
	Heading	$\mathbf{H}_s$ **[m]**	$\mathbf{T}_p$ **[s]**
Waves	Southwest (5/4 π [rad])	2.5	9

Table 7: Environmental loads parameters

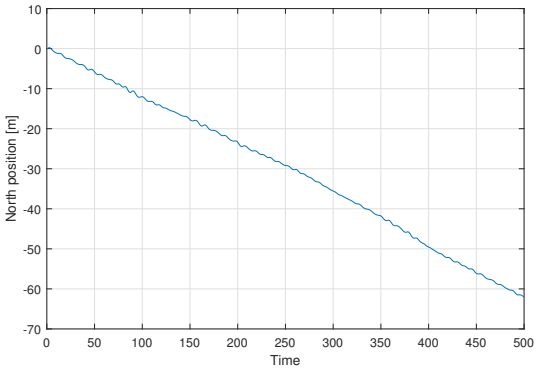
With the data from table 7 dialed into our system the results below was obtained.



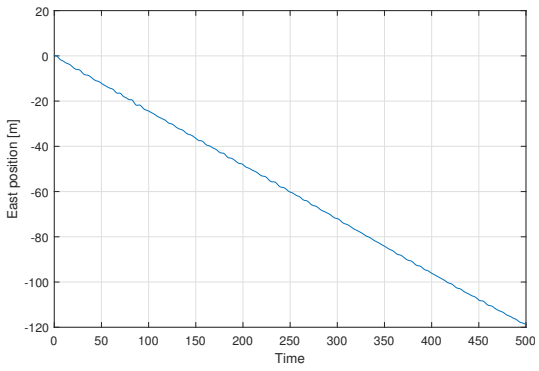
(a) Simulation 1, XY-plot



(b) Simulation 1, heading angle



(a) Simulation 1, position in north



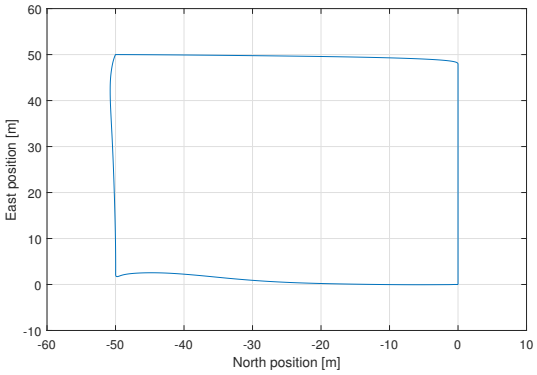
(b) Simulation 1, Position in east

10.1.1 Discussion simulation 1

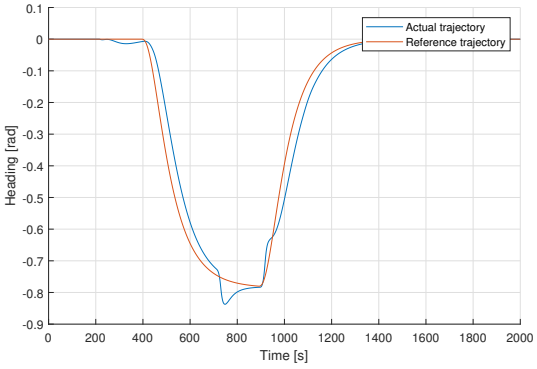
From figure (36a) we can see that the vessel moves in negative surge and negative sway, south west. This is exactly as expected given the environmental loads all propagate from north, east and northeast.
The heading changes to -6 °before it changes over and rotates steadily to 60 °before the simulation ends. This can be due to variation of the wind and current +- 5 degrees and the variations of intensity. A fact that the waves are stochastic can give some variations in the wave forces that can tilt the boat one way. From the plot of the current, wave and wind forces it can be seen that the yaw moment in waves have a large amplitude and that the wind load first get a positive moment after the boat starts turning because when it is facing north. Since the wind is coming from north, heading south, the flow will be symmetric about the longitudinal axis of the ship and hence no moment in yaw.

10.2 Simulation 2 - DP and thrust allocation

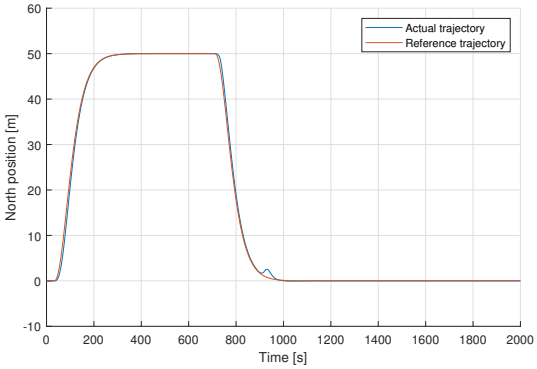
With the environmental loads disabled our system is ran with DP-controller and thrust allocation connected. With this simulation our controller and thrust allocation algorithm is tested in calm sea so the performance can be validated without disturbance. A series of set-points is delivered from the autopilot such that the vessel follows the given four corner test.



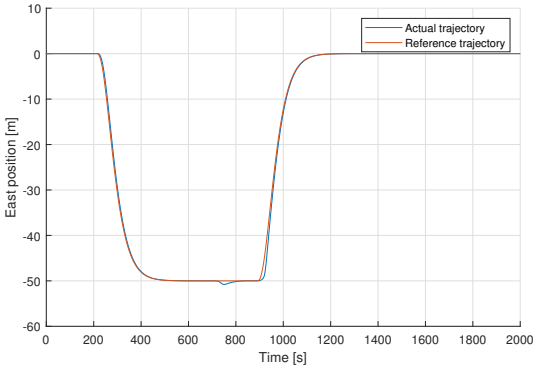
(a) Simulation 2, XY-plot



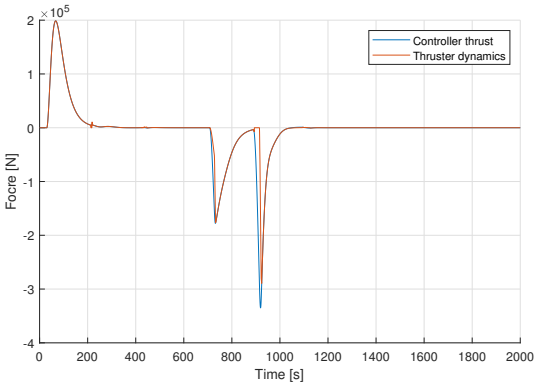
(b) Simulation 2, heading angle



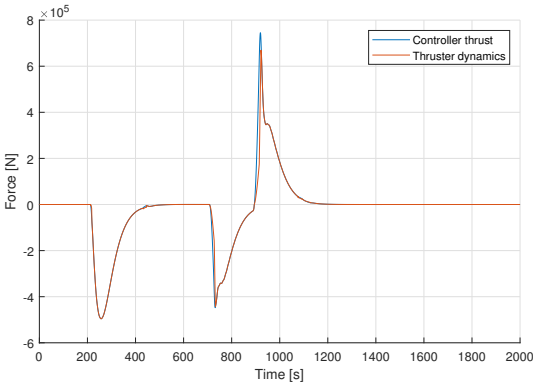
(a) Simulation 2, north position



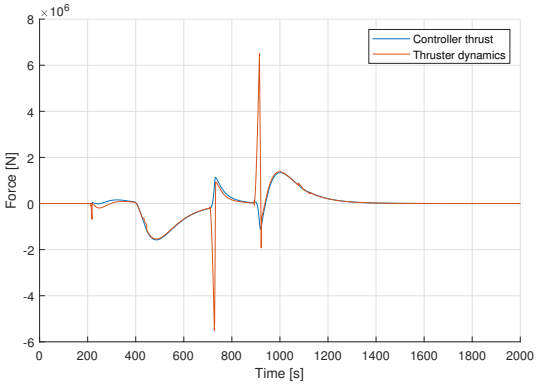
(b) Simulation 2, east position



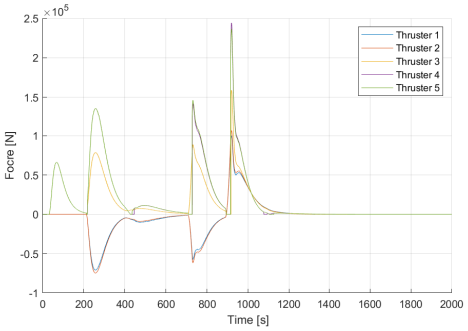
(a) Simulation 2, Controller force and TD force, north



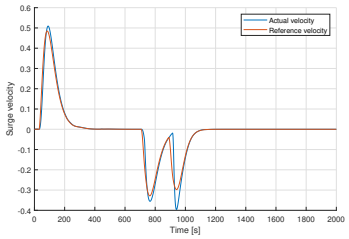
(b) Simulation 2, Controller force and TD, east



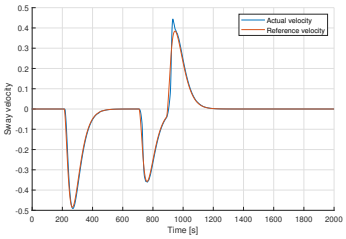
(a) Simulation 2, Controller momentum and TD momentum, heading



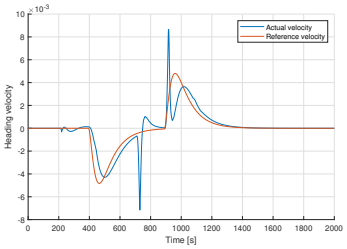
(b) Simulation 2, **u**, set-point force for each thruster



(a) Sim 2, north velocity



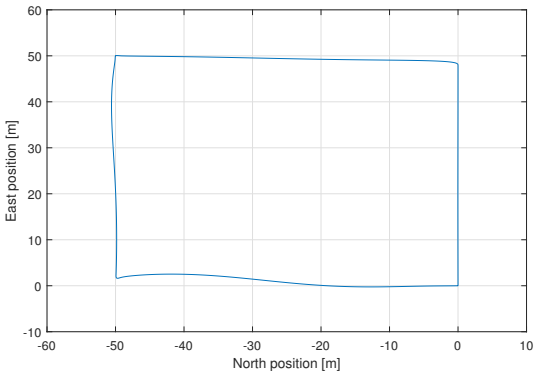
(b) Sim 2, east velocity



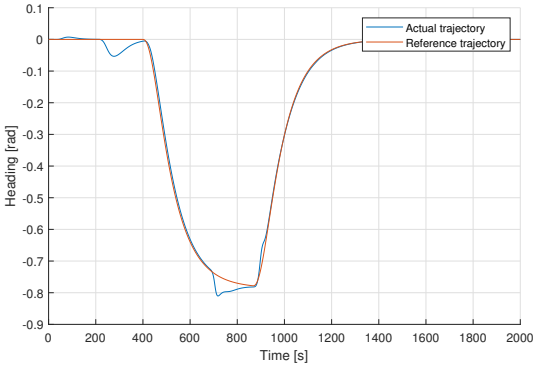
(c) Sim 2, heading velocity

10.2.1 Thruster failure

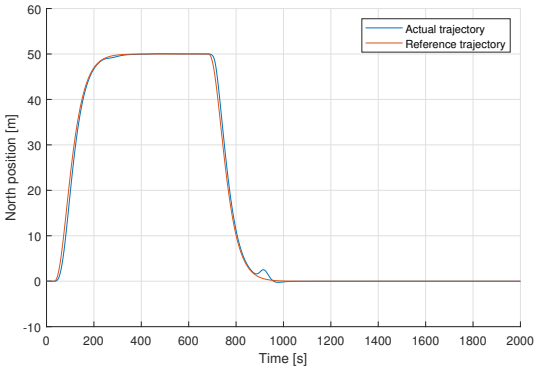
A further test of the system was performed with thrusters 2 and 5 disabled. Except for the disabling of the thrusters the simulation was ran identically to simulation 2.



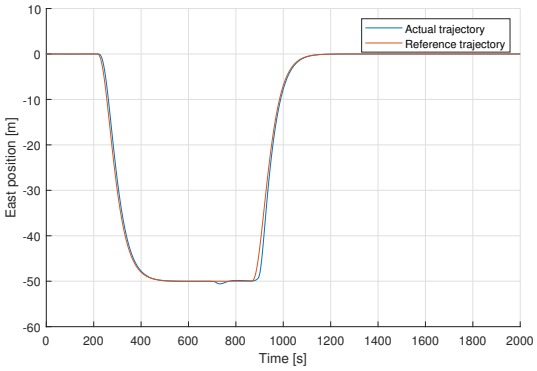
(a) Simulation 2, XY-plot, thruster failure



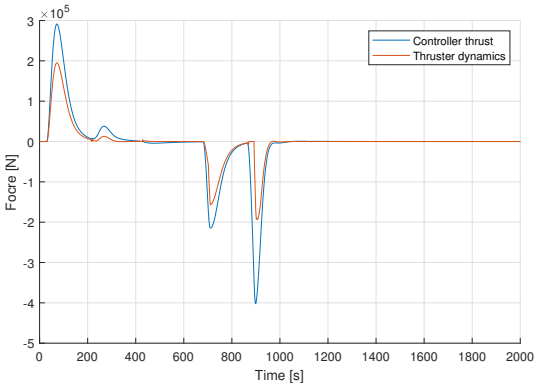
(b) Simulation 2, heading angle, thruster failure



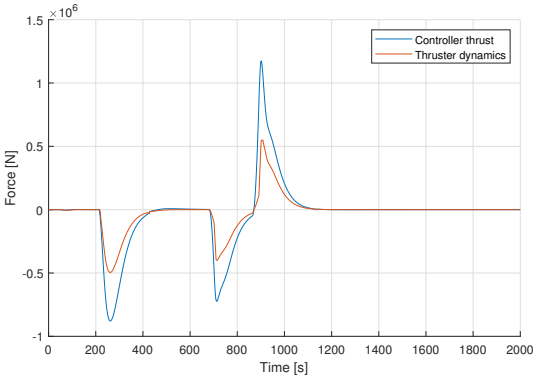
(a) Simulation 2, north, thruster failure



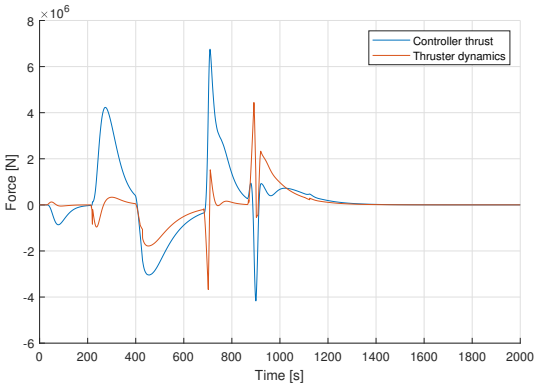
(b) Simulation 2, east, thruster failure



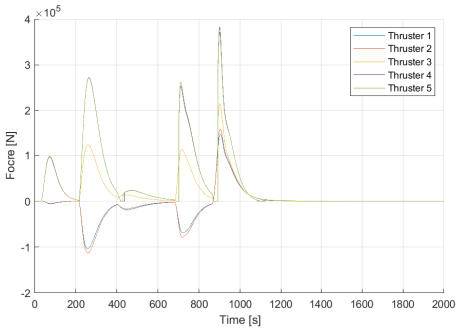
(a) Simulation 2, Controller force and TD force, north, UNKNOWN



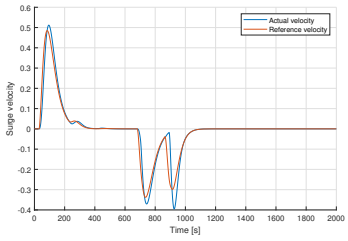
(b) Simulation 2, Controller force and TD, east



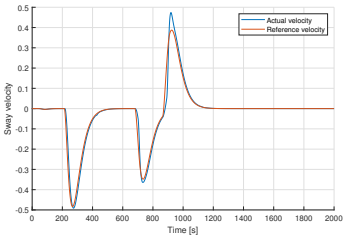
(a) Simulation 2, Controller momentum and TD momentum, heading



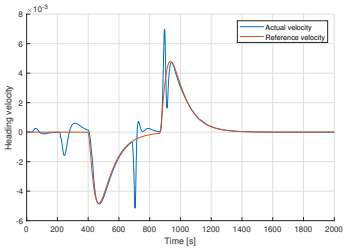
(b) Simulation 2, **u**, set-point force for each thruster, thruster failure



(a) Sim 2, north velocity



(b) Sim 2, east velocity



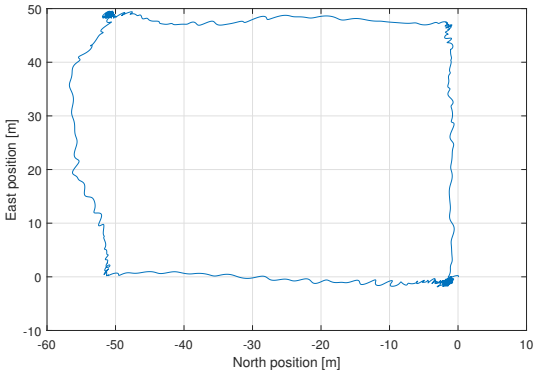
(c) Sim 2, heading velocity

10.2.2 Discussion simulation 2

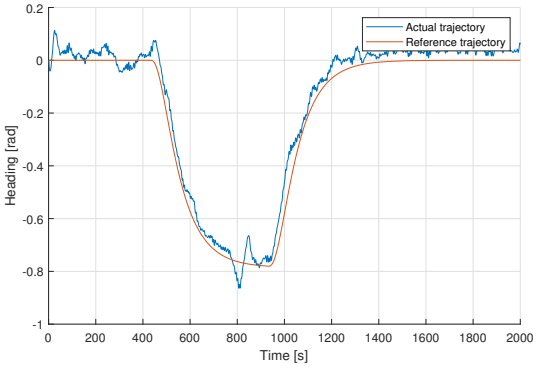
In figure 41a one can see a spike of unwanted yaw momentum at around 730 seconds. By comparing this plot to the one in figure 45b we can see that the time corresponds to the sway direction movement which probably is the cause of this spike. The same movement can also be seen in figure 44a where it disturbs the surge motion.
The controller momentum does not correspond to the thruster momentum in yaw 41a. One can see spikes in the actuation produced by the thruster dynamics. This is probably because the vessel demands a change in sway and yaw at the same time, and because all the thrusters works at the same time the peak of moment occurs. From figure 41b one can see the individual thruster forces. The spike may be caused by an unwanted momentum in yaw due to rapid actuation in surge and/or sway, figure 45a and figure 45b, because of the orientation of the azimuth thrusters. The cause of this is described in more detail in section 8.3.
In the thruster failure part, thruster 2 and 5 are disabled in the thruster allocation block. One can still see forces from thrust allocation algorithm in figure 46b because the algorithm does not know anything about this. Even with two failed thruster, the vessel has no problem following the reference trajectory. One can see another spike in the heading just before it turns in figure 43b at around 300 seconds into the simulation. This is most likely caused by the sway movement, and because one of the bow thrusters are disabled, the vessel is exposed to an unwanted moment which turns the ship.

10.3 Simulation 3 - DP and environmental forces

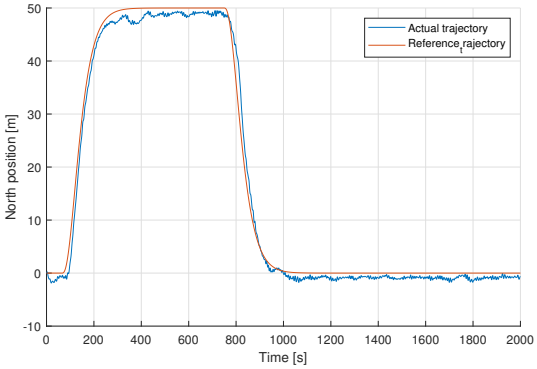
In order to illustrate the benefit of having an observer in a DP-system a simulation was performed with all systems and loads enabled, except the observers.



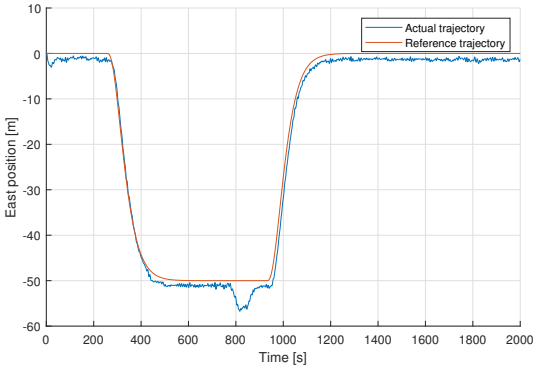
(a) Simulation 3, XY-plot



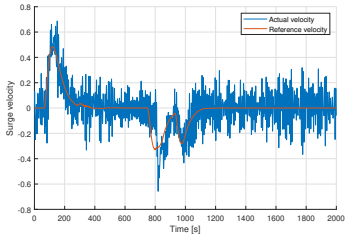
(b) Simulation 3, heading angle



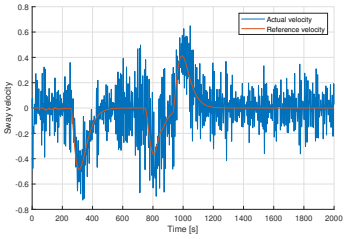
(a) Simulation 3, North position



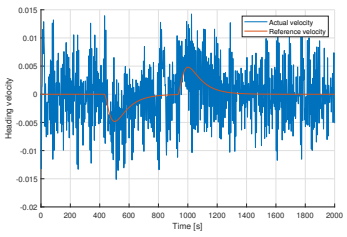
(b) Simulation 3, East position



(a) Sim 3, North velocity



(b) Sim 3, East velocity



(c) Sim 3, heading velocity

10.3.1 Discussion simulation 3

Without an observer, or any other filter, enabled the wave noise are introduced directly into the DP-controller. This in turn makes the azimuth angles and thruster forces scramble to eliminate the noise. This behaviour in not desirable and in long term harmful for system longevity. In addition to mechanical fatigue the vessel is not able to manoeuvre as smoothly desired. Both the position and velocity are oscillatory. Due to the fact that the LQG is a pd-controller these effect are multiplied by a gain. The velocity oscillations is the worst when it comes to frequency. The velocity also oscillates around zero, adding to

the effect. The position is oscillatory as well, but not to the same extend as the velocity. Both the velocity and the position is multiplied by a gain giving forces. These oscillatory forces makes the control dynamics faster than the thruster dynamics allow them to be. The actual velocities are full of noise as a result of no filtration. The LQG, PD-controller, is highly influenced by noise

10.4 Simulation 4 - Observer selection

Simulation 4 tasks us to run a simulation with the same environmental forces as in simulation 1 with a fixed desired DP force. Additionally a simulation without wave forces was performed comparing the two observers to each other.

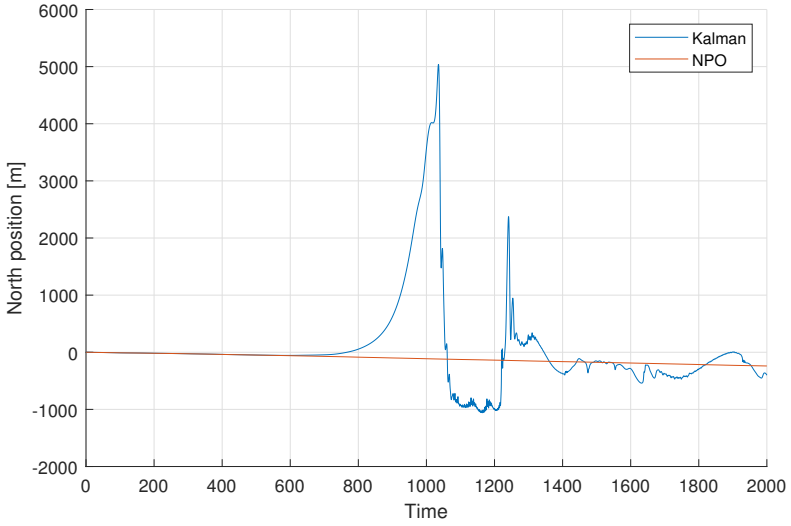


Figure 51: Simulation 4; KF and NPO in north

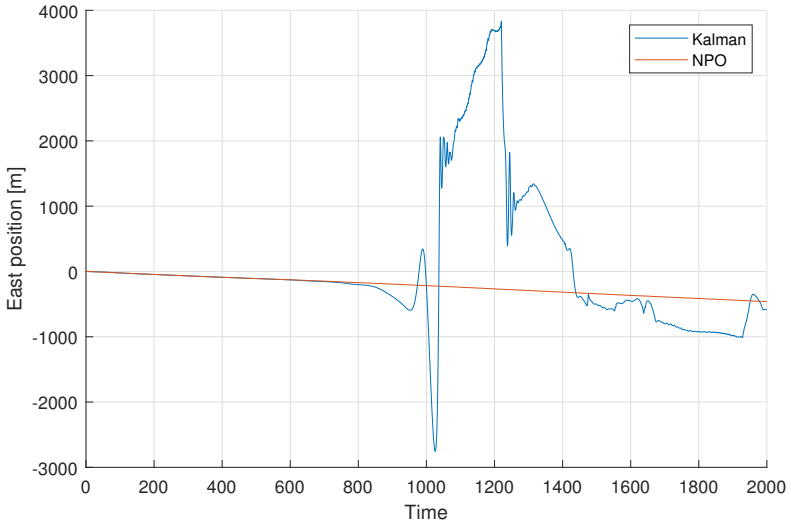


Figure 52: Simulation 4; KF and NPO in east

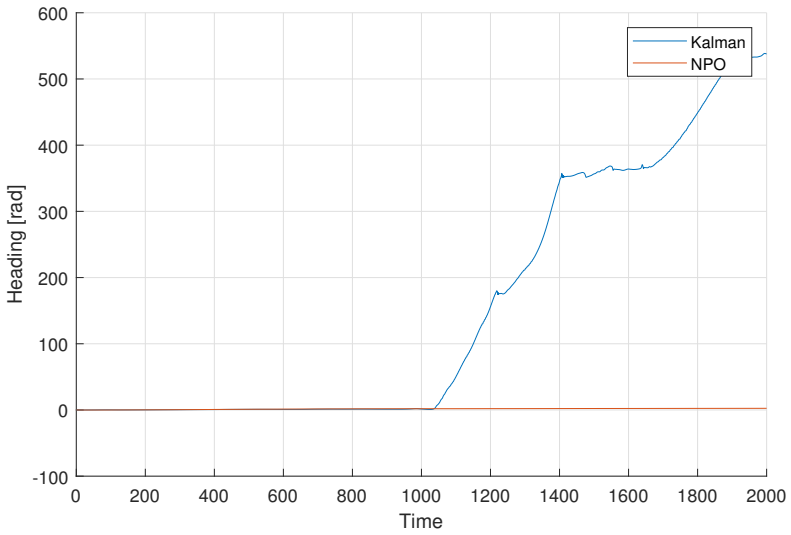
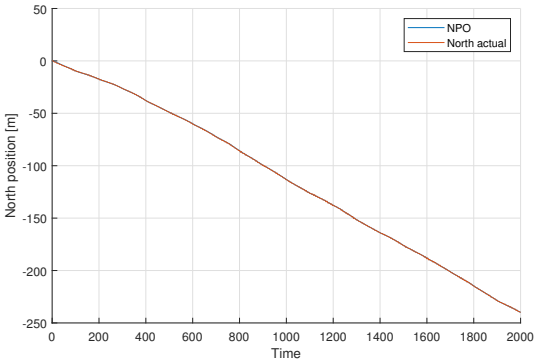
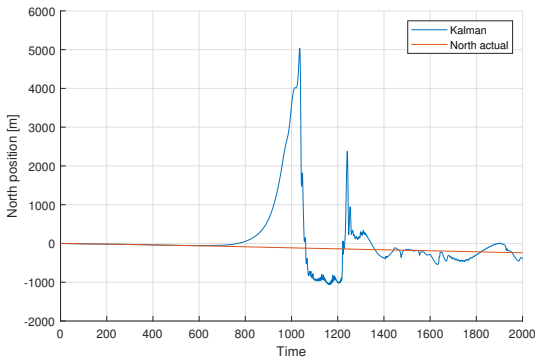


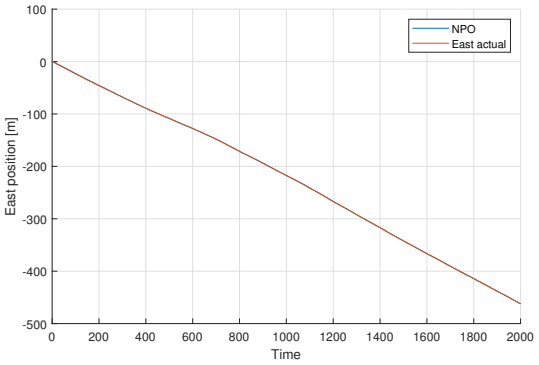
Figure 53: Simulation 4; KF and NPO in heading



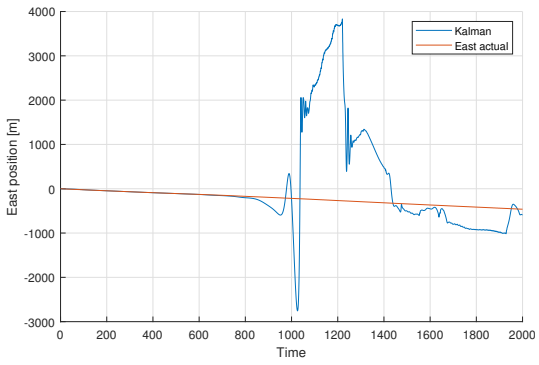
(a) Simulation 4, NPO and actual measurements in surge



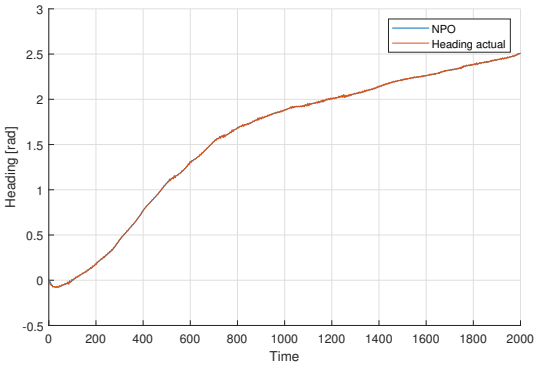
(b) Simulation 4, KF and actual measurements in north



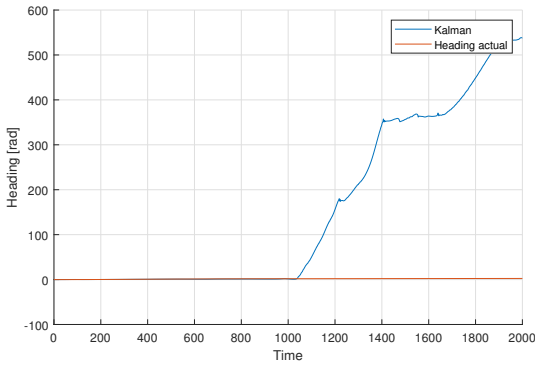
(a) Simulation 4, NPO and actual measurements in east



(b) Simulation 4, KF and actual measurements in east



(a) Simulation 4, NPO and actual measurements in heading



(b) Simulation 4, KF and actual measurements in heading

10.4.1 Discussion simulation 4

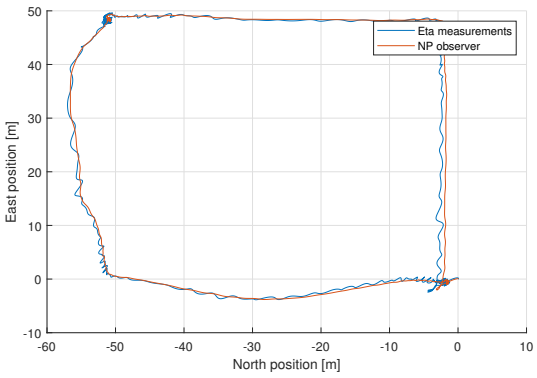
Comparing the NPO and the EKF performance in this simulation, there is a major deviance that can be spotted in the performance of the NPO and EKF. The EKF spikes and the estimated states strays from the measured states at around 1000 seconds. Our leading hypothesis for why the spikes occurs could be due to either the thrust allocation function doing something funky which it should not do, or due to some unstability caused by the system moving out of an area where the EKF is locally/linearized stable.

As a conclusion for observer selection, the NPO was chosen as the observer of choice. This is based on the arguments in section 6.3 and from the results in the simulation. As mention in section 6.3.1, the NPO also seemed far superior as a state estimator until the final moments before the deadline, where we discovered a fault in the implementation of the EKF. Although we had already decided that the NPO was the observer of choice, we were inspired to do some more simulations with the EKF, and compare the two different observers further by testing their performance in the following simulations.

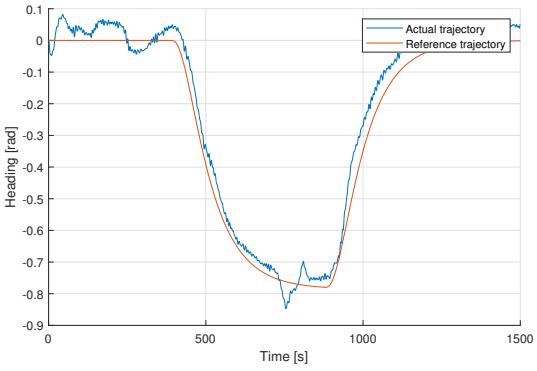
10.5 Simulation 5 - Complete simulation

A final simulation is ran in order to validate our complete system with environmental loads applied. After running simulation 4 we concluded that NPO gave the best results, given our tuning. As with previous simulations the environmental loads from simulation 1 is used 7. For the sake of comparison, we also ran simulation 5 with the EKF as the chosen estimator.

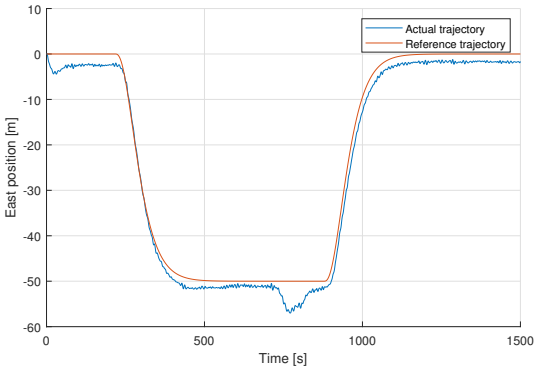
10.5.1 Complete simulation with NPO



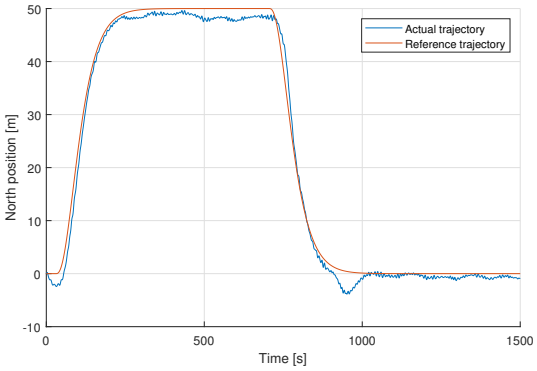
(a) Simulation 5, XY-plot



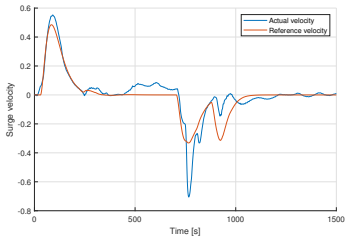
(b) Simulation 5, heading angle



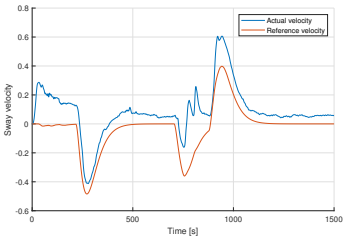
(a) Simulation 5, North



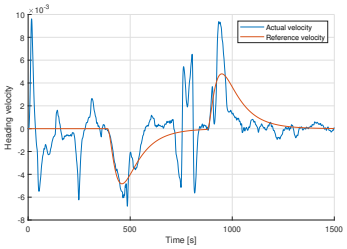
(b) Simulation 5, East



(a) Sim 5, North velocity

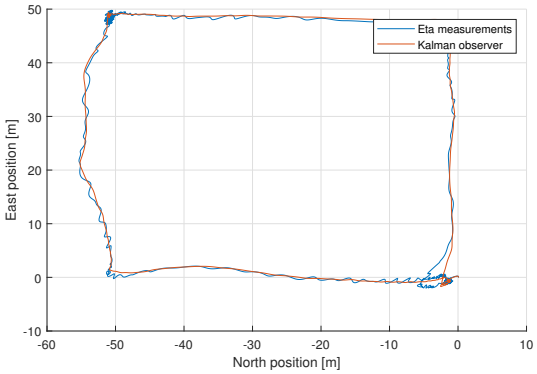


(b) Sim 5, East velocity

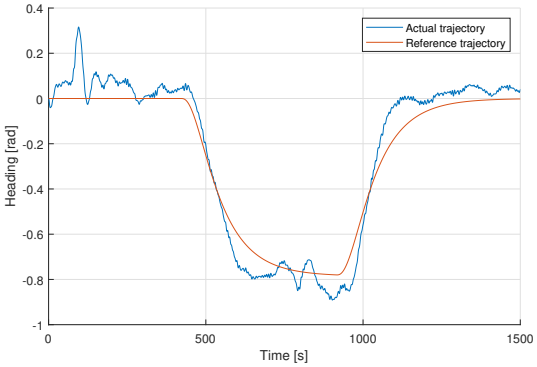


(c) Sim 5, Heading velocity

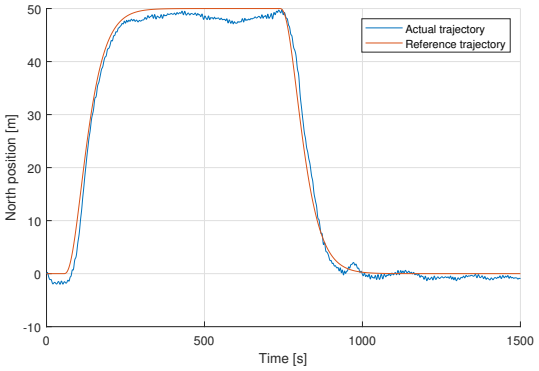
10.5.2 Complete simulation with EKF



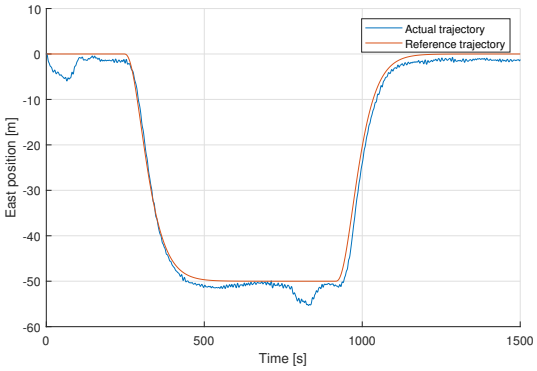
(a) Simulation 5, XY-plot, EKF



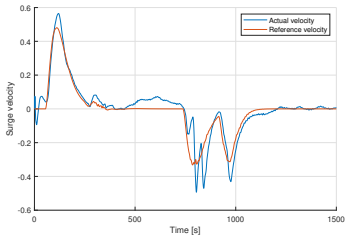
(b) Simulation 5, heading angle, EKF



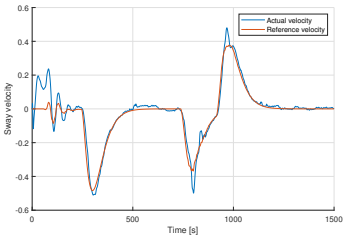
(a) Simulation 5, North, EKF



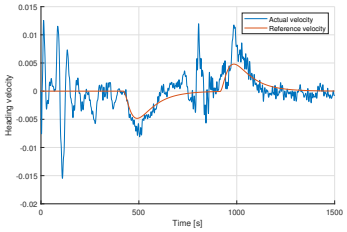
(b) Simulation 5, East, EKF



(a) Sim 5, North velocity, EKF



(b) Sim 5, East velocity, EKF



(c) Sim 5, Heading velocity, EKF

10.5.3 Discussion simulation 5

Regarding the observer of choice, the NPO, we note that ship is managing to perform the path following objective, although with some spikes and deviation in the estimated states. This can be seen in the prior subsection, "Complete simulations with NPO". There are some undershooting the reference value in both surge and sway, which could possibly be explained by a lack of integral effect. The integral effect could remove the stationary deviance which can be noticed for the segments where movement in a state is rather stationary. It is worth mentioning that the LQG in combination with the NPO should be able to do this, and a reason for the lack of this property in our system, could be non-optimal tuning of the NPO.
Regarding the simulation run where the EKF was chosen as an observer, the same could be noticed. Although the stationary deviancy is slightly smaller, the variance in the states are more present. This is something that most likely can be removed with more extensive tuning.
Both estimators performed their task within reasonably small margins (approx. 2-3 [m]) for a vessel of our dimensions.

10.6 Simulation 6 - Capability Plot

In this simulation a capability plot is created for the vessel. This test is performed in order to validate the DP-system with environmental loads from different directions. This

plot visualizes the percentage of maximum force needed to keep the vessel stationary. The vessel is kept steady at a set point while the environmental loads move co-linearly with 10 degrees increment. The wind velocity is set to $U_3 = 15$ [m/s] and the current velocity is set to $U_c = 0.2$ [m/s]. The wave parameters are set to $H_s = 5$ [m] and $T_p = 10$ [s].
The simulation was ran using a for-loop, running each simulation for 300 seconds, and then incriminating the direction of the environmental loads by 10 degrees for the next step. The percentage utilization of max force for each thruster is calculated before the total thrust force utilization percentage is calculated.

This simulation is also done with both the NPO and EKF as our observers

10.6.1 Capability plot with NPO

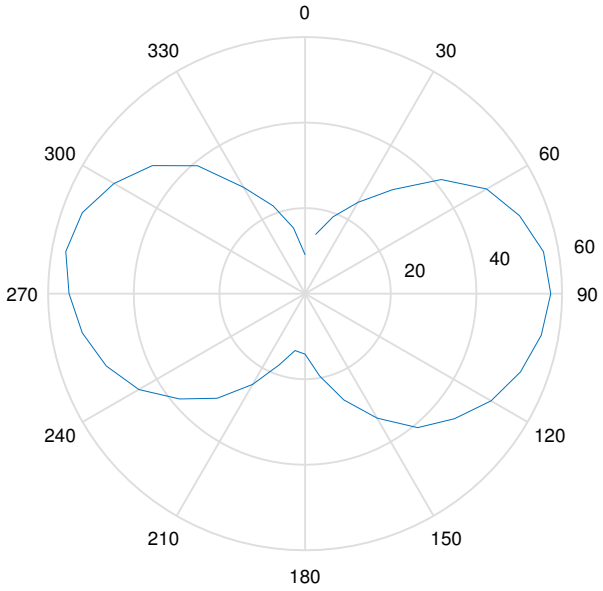


Figure 63: Simulation 6, Capability plot, NPO

10.6.2 Capability plot with EKF

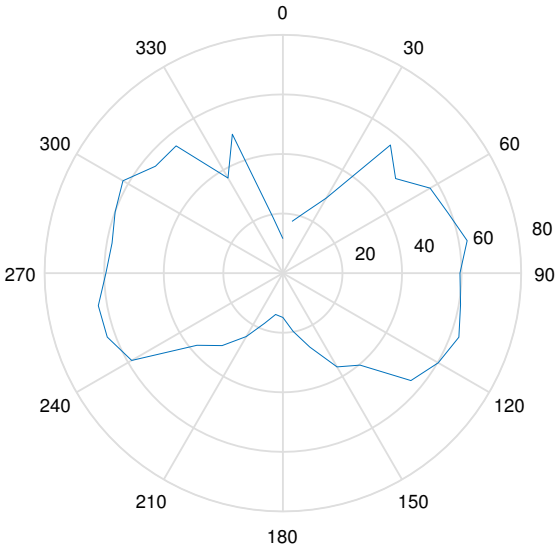


Figure 64: Simulation 6, Capability plot, EKF

10.6.3 Discussion simulation 6

From the shape of figures 63 and 64 one can see that the vessel uses less force when the environmental loads are applied head on or to the stern. Hence the vessel is more power efficient in these conditions. Similarly the vessel requires more power when the environmental loads are applied to starboard or port side. These results seems logical given that the area in which the forces act's on are greater from starboard and port compared to head on and from the stern. In accordance with this the vessel uses more force to hold the position.
Directly from port and starboard the average force was close to 60 % and when the force's was applied from stern or bow, the average force was around 20%.

This plot can be used for keeping position and to minimize the power usage by changing heading into the weather.

10.7 Simulation 7 - Observer robustness

To verify if our system and observes works in extreme conditions, we were tasked with increasing the environmental loads. The waves were set to a significant wave height of 8m, the period to 13 seconds and the same current and wind values as from simulation 1 10.1.

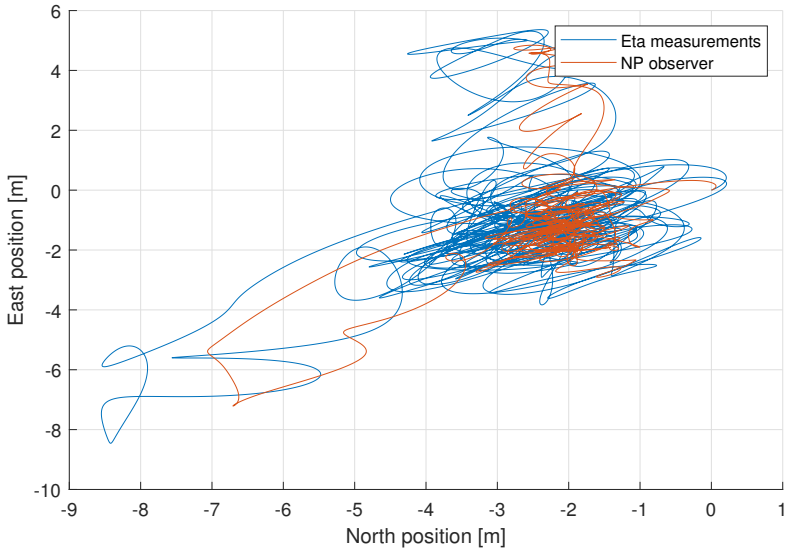
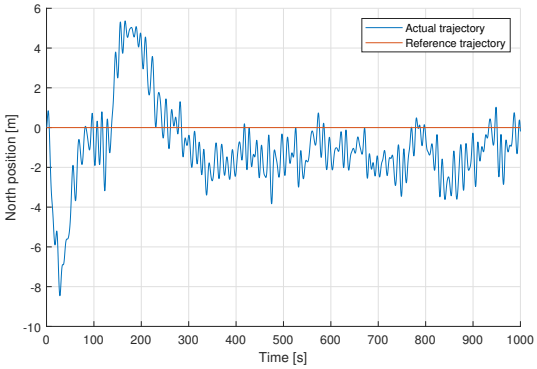
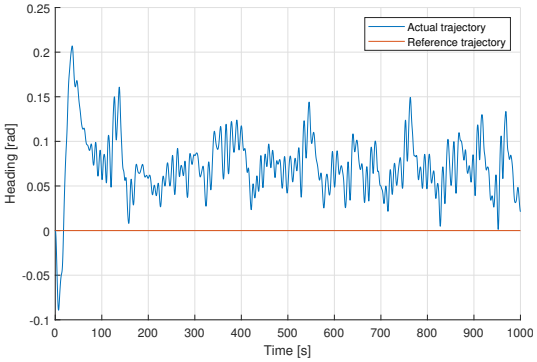


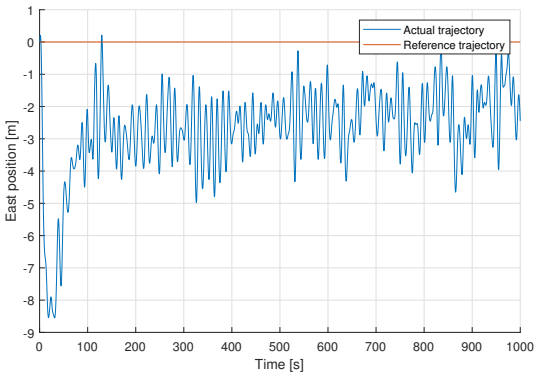
Figure 65: Simulation 7, XY



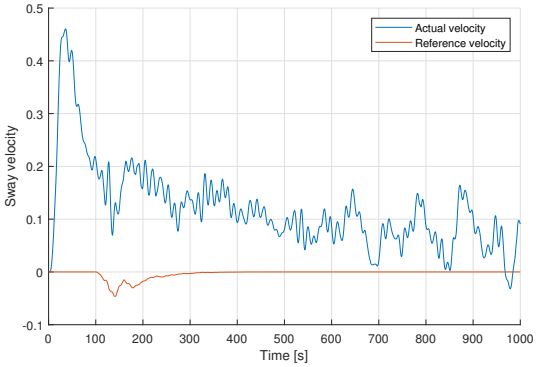
(a) Simulation 7, North with reference



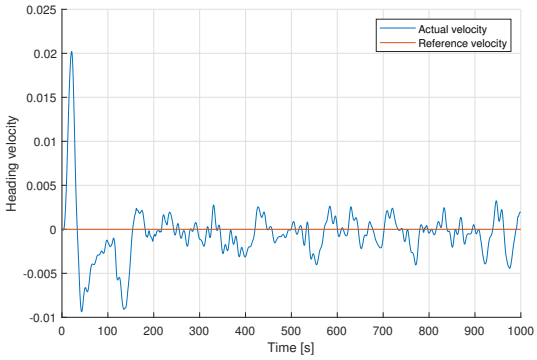
(b) Simulation 7, Heading with reference



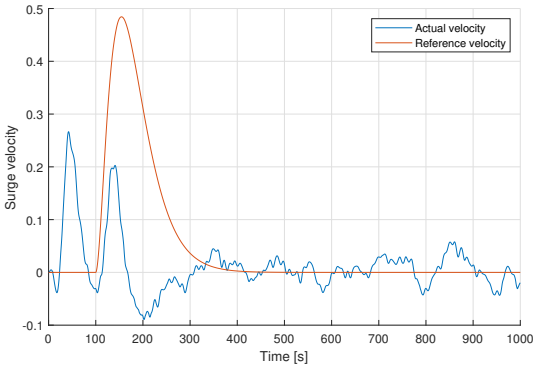
(a) Simulation 7, East with reference



(b) Simulation 7, East velocity



(a) Simulation 7, Heading velocity



(b) Simulation 7, North velocity

10.7.1 Simulation 7 - Observer robustness EKF

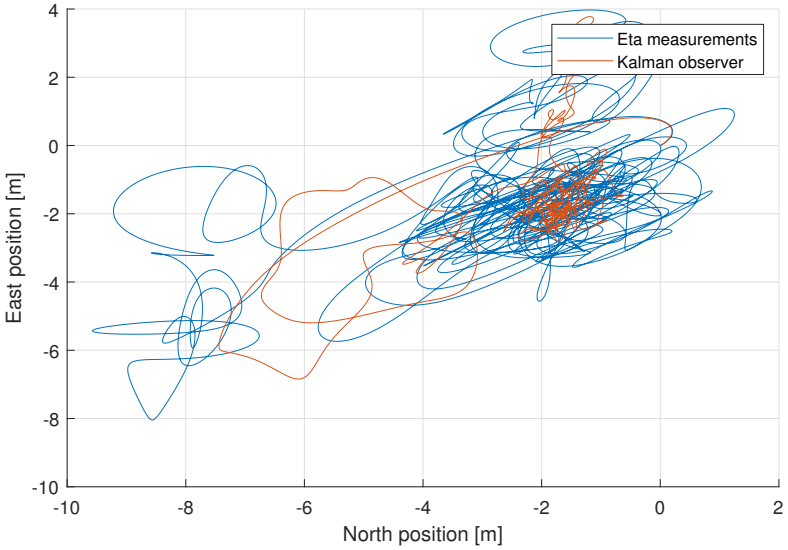
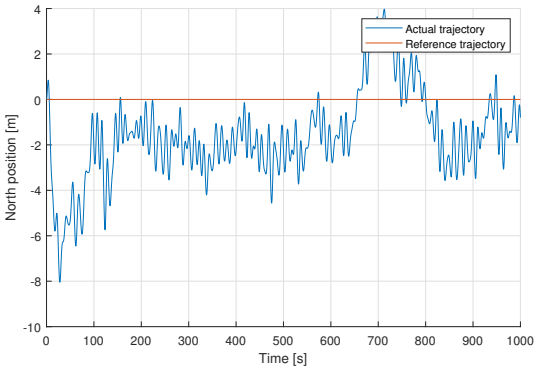
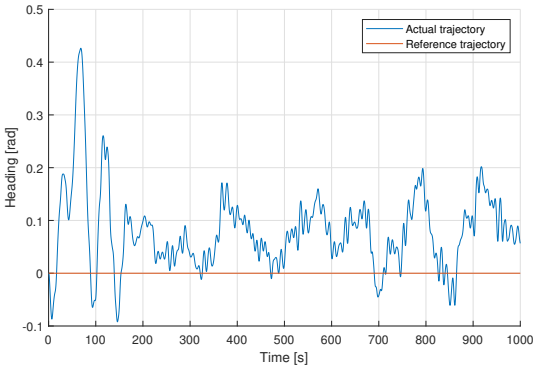


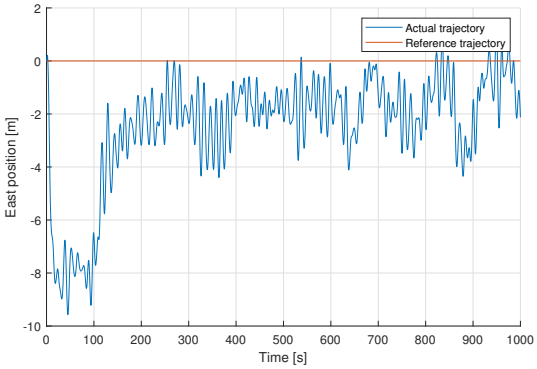
Figure 69: Simulation 7, XY EKF



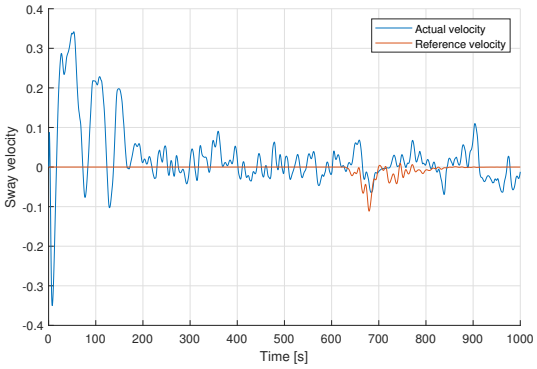
(a) Simulation 7, North with reference, EKF



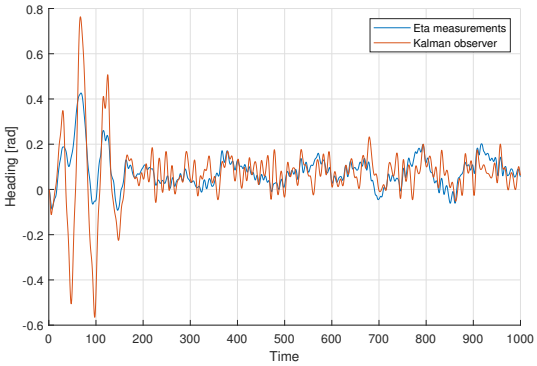
(b) Simulation 7, Heading with reference, EKF



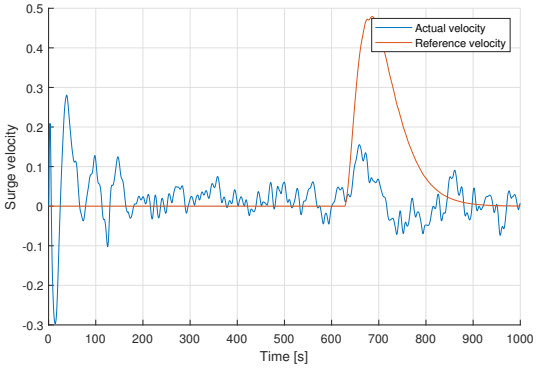
(a) Simulation 7, East with reference, EKF



(b) Simulation 7, North velocity



(a) Simulation 7, Heading velocity, EKF



(b) Simulation 7, North velocity, EKF

10.7.2 Discussion simulation 7

With the high environmental loads the system struggles to keep steady position. The NPO deviates from position with a maximum distance of 5 meters in sway direction, and a steady state maximum heading deviation of 8.5 °. The EKF performers better with a

maximum deviation og 4 meters in sway direction. The steady state maximum heading deviation is similar to the NPO, with a deviation of 8.5 °.
The results indicates that the EKF behaves better outside the tuned area. While the system is tuned for a $T_p = 9$ [s] this simulation is ran with a $T_p = 9$ [s]. This can be explained by the higher bias tuning of the EKF compared to the NPO. The NPO was tuned to have a bias estimate which compensated for the JONSWAP wave spectrum, while the EKF had a slightly more aggressive bias estimate tuning, i.e the last three elements of $\mathbf{Q}$ were set to a higher value. This is most likely what caused the EKF to perform better under a more extreme environmental load, than what the NPO did. The three first elements of $\mathbf{Q}$ were tuned to filter out wave loads comming from a calmer wave disturbance. This led to a worse wave filtering, which can be seen in the high frequency spikes.

11 Conclusion

After this project we have thouroughly understood how complex and how important the trust allocation module is for a system of this size. We started the simulations with a thrust allocation with a forbidden sector larger than the one we finished with, which resulted in the plots seen under simulation 3, where the controller managed to follow the path even though the system didn't use an observer to filter out high frequency loads. The fact that we managed to follow the path with only a controller, surprised us. Allthough it was not up to safety standards, as the thrusters had to rotate immensely to compensate for the high frequency disturbances.

The DP control system with thrust allocation, two different state estimators, autopilot and environmental models implemented in this project satisfied the control objective of path following and station keeping in different sea states. Both the nonlinear passive observer and the extended Kalman filter performed as intended. Although there were some hard to explain spikes as mentioned in simulation 5, section 10.5 and simulation 7, section 10.7. The performance of the DP-system done in simulation 5, showed that both observers were capable of operating within reasonable operating limits for a DP-system under these environmental conditions. But it is worth mentioning that more tuning could be done to further improve performance. This, and other improvements to the system is suggested in the section regarding further work.and the tuning could be implemented.

The capability plots showed a similar performance with average thrust for both estimators, and both estimators managed to operate with state estimation feedback under the environmental conditions set by the assignment in simulation 5, section 10.6.3.

12 Further work

This project could be further improved by performing a more excessive tuning, both on the NPO and the EKF. The bias estimate section of the NPO we're only initially tuned to compensate for the wave load from a JONSWAP wave spectrum, and not tuned to compensate for current and wind loads. For the EKF, we tuned the $\mathbf{Q}$-matrix to compensate for wave-frequency loads by tuning the three first elements, and the bias

The bias matrix $\mathbf{E_b}$ could have been increased to compensate for the steady state deviation we had on the reference position and the estimated position.

For improving the controller things that could have improved the performance could have been implemented:

- A wind feedforward term could better have compensated for the wind loads
- A model reference feedforward control action.
- A integral action.

1) could have made the controller compensate better for the wind feedforward. 2) could improved following trajectories. 3) could have removed steady state deviations like

in simulation 5. But a drawback introducing an integral effect to the system, is that we lower the systems phase, thus reducing the bandwidth where the system is stable. We also increase the systems time constant, which might make the inner loops too slow for the outer loops to manage to successfully perform its objective.

In general drawbacks of implementing more force terms from a controller its increased complexity(more parameters to tune).

The observers and controllers could also be implemented with hybrid control techniques and switch between settings and tuning parameters for different environmental conditions. This could lead to a more robust system which could operate under both the environmental loads from simulation 5 and 7. Another modification for increasing the performance in simulation 7 is using the control plant model for the extreme seas where you are not filtering out a WF model(1.order wave motion). Also one thing could be to have an adaptive observer where the WF model have the ability to change with the sea state.

Regarding the thruster allocation system, we had some issues with unwanted moments due to the forbidden zones being too large. This was fixed in part 2 before starting the simulations by decreasing the zone, thus making the force vector coming from the thrusters that aligned up, more uniform. A fix to this issue could be to either implement a more intricate thruster allocation system which used a third thruster to compensate for the dis-aligned thrust vector, or to introduce a weighting function. The weighting functions purpose is to reduce the thrust coming from a vector that has been locked due to its wish to enter the forbidden zone, since it otherwise gives too much thrust in an unwanted direction.

References

[1] Sørensen, Asbgeir J (2018): *Marine Cybernetics towards autonomous marine operations and systems*

[2] Brodtkorb H, Astrid (2019): *Lecture notes on State estimation*

[3] Fossen, T.I (2011) *Handbook of Marine Craft Hydrodynamics and Motion control.* John Wiley and Sons, ltd.

[4] T. I. Fossen and T. Perez (2004). Marine Systems Simulator (MSS). URL: https://github.com/cybergalactic/MSS

A MATLAB Code

In this appendix relevant code segments can be found. Matlab code regarding plotting functions are not included here unless it is also a part of a more relevant script. Theese files can be found in the submitted zip folder.

B Main initialization

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% init()                                                                        %
%                                                                               %
% Set initial parameters for part1.slx and part2.slx                            %
%                                                                               %
% Created:      2018.07.12  Jon Bj rn                                             %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
%% loading parameters
load('supply.mat');
load('supplyABC.mat');
load('thrusters_sup.mat')


%% initial parameters

% Initial position x, y, z, phi, theta, psi
eta0 = [0,0,0,0,0,0]';
% Initial velocity u, v, w, p, q, r
nu0 = [0,0,0,0,0,0]';

%% Given matrixes

M = [6.8177e6, 0, 0; 0, 7.8784e6 -2.5955e6; 0, -2.5955e6 3.57e9];
% Mass matrix(rigid body + added mass) in surge, sway, yaw

D = [2.6486e5, 0, 0; 0, 8.8164e5 0; 0,0, 3.3774e8];
% Damping matrix (DL + Dm)

%% Initialization of systems
init_autopilot;
init_controller;
init_observer;
init_wind;
```

```
init_thrustalloc;

disp('System has been initialized');
```

C Autopilot initialization

```
%% Autopilot:
    %Initializing changing parameters:
    global p
    p = 1;          %Counter for set points
    global time
    time = 0;       %Initializing time counter
    global start_time
    start_time = 0; %Initializing start time
    %Set points
    eta_set = [0 0 0; 50 0 0;50 -50 0; 50 -50 -pi/4; 0 -50 -pi/4; 0 0 0];

    disp('Auto pilot has been initialized.');
```

D Controller initialization

```
%% Reference model: Filter parameteres

t1 = 20;
t2 = 20;
t3 = 20;

% mass spring damper system
omega1 = pi/100;
omega2 = pi/100;
omega3 = pi/150;
zeta1 = 1;
zeta2 = 1;
zeta3 = 1.2;
gamma_m = [omega1^2 0 0; 0 omega2^2 0; 0 0 omega3^2];
omega_m = [2*zeta1*omega1 0 0; 0 2*zeta2*omega2 0;0 0 2*zeta3*omega3];

%% PID-Controller: Gain matrixes (part 1)
%consists of [Kp_x 0 0 ; 0 kp_y ; 0 0 kp_psi] etc
%%Theese tuning parameters are good for simulation 3 and 4
Kp_matrix = [2*10^(5) 0 0 ; 0 4*10^(5) 0 ; 0 0 1*10^(7)];
%
Ki_matrix = [2*10^(2) 0 0 ; 0 7*10^(2) 0 ; 0 0 1*10^(3)]; %Ki_surge = 5*10^3
%
```

```
Kd_matrix = [4*10^(6) 0 0 ; 0 5*10^(6) 0 ; 0 0 8*10^(4)];

%%Theese tuning parameters are good for simulation 1 and 2
% Kp_matrix = [4*10^(5) 0 0 ; 0 2*10^(5) 0 ; 0 0 1*10^(7)];
%
% Ki_matrix = [3*10^(3) 0 0 ; 0 2*10^(3) 0 ; 0 0 1*10^(3)]; %Ki_surge = 5*10^3
%
% Kd_matrix = [6*10^(6) 0 0 ; 0 5*10^(6) 0 ; 0 0 8*10^(4)];

%% LQG-Controller: Calculation of matrixes and gains.

H_36 = zeros(3,6); % reduction matrix for i = 3
H_36(1,1) = 1;
H_36(2,2) = 1;
H_36(3,6) = 1;
H_33 = eye(3);

A_lqg = [-inv(M)*D, zeros(3); eye(3), zeros(3)]; % state space modell A matrix
B_lqg = [inv(M)*H_33; zeros(3)*H_33]; % state space modell B matrix
E_lqg = [inv(M);zeros(3)];
C_lqg = [zeros(3), eye(3)];
% Weighting matrices
Q_lqg = [0.04 0 0 0 0 0; ...   %Velocity
     0 0.04 0 0 0 0; ...   %
     0 0 9 0 0 0; ...    %
     0 0 0 10 0 0; ...     %Position
     0 0 0 0 200 0; ...    %
     0 0 0 0 0 3280];       %
P_lqg = [(2*10^4)^-2 0 0; ...'
     0 (2*10^4)^-2 0; ....
     0 0 (2*10^6)^-2];


% Ricatti equation stationary(finding R_inf)
% 0 = -A*R_inf - A'*R_inf + R_inf*B*inv(P)*B'*R_inf-Q
k_lqg = lqr(A_lqg,B_lqg,Q_lqg,P_lqg);
% Stationary gain
% G = inv(P)*B'*R_inf

disp('Controller has been initialized.');
```

E Observer initialization

```
%%%%%%%%%%%%%%%% Given matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M_inv = inv(M);
%% Wave frequency constants
%%%%%%%%%%%%%%%%%%%%%%%%Wave frequency constants %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fossen page 217: Assuming jonswap
zetan_1 = 0.1; zetan_2 = 0.1; zetan_3 = 0.1;
wn_1 = 2*pi/9; wn_2 = 2*pi/9; wn_3 = 2*pi/9; %(Tp = 9 sec)
Kw_1 = 1; Kw_2 = 1; Kw_3 = 1;

%Bias time constants
%Studass sa dette var typisk h y grad, for    kunne garantere at b_dot g r mot nul
Tb_1 = 1000; Tb_2 = 1000; Tb_3 = 1000;

% Putting stuff into a vector
Zetan = [zetan_1, zetan_2, zetan_3];
Wn =[wn_1, wn_2, wn_3];
Kw = [Kw_1, Kw_2,Kw_3];
Tb = [Tb_1,Tb_2, Tb_3];

%% Kalman filter %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step Length
h_step=0.1;

%measurement noise
noise_scalar = 1;

%%%%%%%%%%%%%%%% Design Parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Position and heading measurement noise
R_kf = diag([1, 1, 0.1]);
%Process noise variance
Q_kf = diag([0.06,0.06,0.25,2*10^(6),2*10^(6),1*10^(6)]);

%Initial conditions:[Ksi, eta, b, nu]
x0 = zeros(15,1);
P0 = diag(ones(1,15));

%% State Matrices for observer state space
%%%%%%%%%%%%%%%% State Matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x_dot = f(x) + Bu + Ew, y = Hx +v
Aw = [zeros(3,3) eye(3) ;
    -diag([Wn(1), Wn(2), Wn(3)])*diag([Wn(1), Wn(2), Wn(3)])',...
```

```
    diag([-2*Zetan(1)*Wn(1) -2*Zetan(2)*Wn(2), -2*Zetan(3)*Wn(3)])];

B_kf = [zeros(6,3) ; zeros(3,3) ; zeros(3,3) ; M_inv];

%Diagonal bias scaling matrix parameters
E_b1 = 1; E_b2 = 1; E_b3 = 1;
E_b = diag([E_b1, E_b2, E_b3]);

E_w = [zeros(3,3) ; diag([Kw_1, Kw_2, Kw_3])];
E_kf = [E_w zeros(6,3) ;...
 zeros(3,3) zeros(3,3) ;...
 zeros(3,3) E_b ; zeros(3,3) zeros(3,3)];

Cw = [zeros(3,3) eye(3)];
H_kf = [Cw eye(3) zeros(3,6)];

EKFData = struct('Aw', Aw,'B_kf', B_kf,'E_kf', E_kf,'H_kf', H_kf,...
    'Q_kf', Q_kf, 'R_kf', R_kf,'x0', x0,'P0', P0,'D', D,'M_inv', M_inv,...
    'h_step', h_step, 'Zetan', Zetan,'Wn', Wn,'Kw', Kw,'Tb', Tb);


%% Nonlinear passive observer

k1 = -2*(1-zetan_1)*1.2*wn_1/wn_1; k2 = k1; k3 = k1;
k4 = 2*wn_1*(1-zetan_1); k5 = k4; k6 = k4;
k7 = 1.2*wn_1; k8 = k7; k9 = k7;

K1 = [diag([k1,k2,k3]); diag([k4 k5 k6])]   ;
K2 = diag([k7,k8,k9]);
K4 = 0.01*diag([1 0.3 1]).*diag(M);
K3 = 0.1*K4;

disp('Observers has been initialized.');
```

F Wind initialization

```
%% Wind constant

global U_mean;
global S;
global U_gust;
global C_w;
global  counter;
counter = 1;
C_w = importdata('wind_coefficients.txt',' ',1);
C_w = C_w.data; % [alpha, Cx, Cy, Cz, C_theta, C_phi, C_phi]

k_wind = 0.003;
mu = 0.001;
U10 = 12;
z = 3;
h = 1;
w = 0.005;
w_angle = 5*pi/180;

z0 = 10*exp(-2/(5*sqrt(k_wind)));
%U_mean = U10 * 5/2 * sqrt(k) * log(z/z0);
U_mean = 10;

nfreq = 100;
L = 1800;
k_wind = 0.0026;

for i = 1:nfreq
    f(i) = 0.01 + (i-1)/(nfreq-1) * 0.99;
    ftilde = L*f(i)/U10;
    S(i) = (4*k_wind*L*U10) / ((2+ftilde^2)^(5/6));
end
phi = 2*pi * rand(nfreq,1);
counter2 = 0;

for t = 0:0.1:50000
    counter2 = counter2 +1;
    U_gust(counter2+1) = 0;
    for i = 1:nfreq
        U_gust(counter2+1) = U_gust(counter2+1) + sqrt(2*S(i)*(f(2)-f(1))) * cos(2*
    end
```

```
42 end
43
44 disp('Wind model has been initialized.');
```

G Thrust allocation initialization

```
%% Thrust allocation

l_aa = -28.85;
l_bt1 = 39.3;
l_bt2 = 35.3;
l_af = 31.3;
b_s = 5;
b_p = -5;

global thrust_loc
thrust_loc = [l_bt1, l_bt2, l_af, l_aa, b_s, b_p];
global K_t
K_t = eye(8);

disp('Thrust allocation model has been initialized.');
```

H Autopilot function

```
function [surge_0, sway_0, heading_0, p, time, start_time] = autopilot(set_points,
%Inputs current surge, sway, heading and outputs ref values surge_0,
%sway_0, heading_0

time_crit = 30;  %Time criteria at set point before moving
accu = 5;        %Accuracy/Tollerance
accu_h = 5 * pi/180;   %Heading accuracy



dim = size(set_points);

if p >= dim(1)
    time = 0;
else
    %Checking positions:
    if (set_points(p, 1)-accu) <= surge && surge <= (set_points(p, 1) + accu)

        if (set_points(p, 3)-accu_h) <= heading && heading...
                <= (set_points(p, 3)+ accu_h)

            if (set_points(p, 2)-accu) <= sway && sway...
                    <= (set_points(p, 2)+ accu)
%Checking if vessel is at desired position for desired amount of time
                if time == 0
                   start_time = clock;  %Start time of counting time in pos.
                   time = 0.01;
                else%Counting time at position
                    time = (clock-start_time);     %Counting time at position
                end
                if time >= time_crit
                    p = p + 1;    %Incrementing the set point
                    if p > dim(1)
                        p = dim(1);     %Last point met
                    end
                    time = 0;                   %Zeroing time counter
                end
            else
                time = 0;   %No longer in correct position, stop counting.
            end
        else
```

```
            time = 0;
        end
    else
        time = 0;
    end
end

%Setting referance points:
surge_0 = set_points(p, 1);
sway_0 = set_points(p, 2);
heading_0 = set_points(p, 3);

end
```

I Extended thrust allocation function

```
function [u_thrust, alpha] = thrust_allocation_extended(tau, thrust_loc, K_t, alpha
%Calculates the control vector for each thruster

T_extended = [0 0 1 0 1 0 1 0; ...
    1 1 0 1 0 1 0 1;
    thrust_loc(1) thrust_loc(2) 0 thrust_loc(3) thrust_loc(5) ...
    thrust_loc(4) thrust_loc(6) thrust_loc(4)];

u_e = inv(K_t)*pinv(T_extended)*tau;

u_d_3 = sqrt(u_e(3)^2 + u_e(4)^2); %Azimuth forward
u_d_4 = sqrt(u_e(5)^2 + u_e(6)^2); %Azimuth stern, starboard
u_d_5 = sqrt(u_e(7)^2 + u_e(8)^2); %Azimuth stern, port

%Thrust angles, alpha
alpha = zeros(5, 1);
alpha(1) = pi/2;                        %Tunnel thruster 1
alpha(2) = pi/2;                        %Tunnel thruster 2
alpha(3) = atan2(u_e(4), u_e(3));       %Azimuth forward
alpha(4) = atan2(u_e(6), u_e(5));       %Azimuth aft, starboard
alpha(5) = atan2(u_e(8), u_e(7));       %Azimuth aft, port

u_thrust = [u_e(1); u_e(2); u_d_3; u_d_4; u_d_5];

%Limiting thrust to azimuths under heavy position change
for i = 3:5
   if(alpha(i) > alpha_prev_real(i) + 10*pi/180 || alpha(i) < alpha_prev_real(i) -
      u_thrust(i) = 0;
   end
end

%Checking forbidden sectors:
    %Azimuth stern, starboard
    if alpha(4, 1) > pi/4 && alpha(4, 1) <= 3*pi/8 %Upper half of sector
        alpha(4, 1) = pi/4;
    elseif alpha(4, 1) > 3*pi/8 && alpha(4, 1) < pi/2 %Lower half of sector
        alpha(4, 1) = pi/2;
    end
    %Azimuth stern, port
    if alpha(5, 1) > -pi/4 && alpha(5, 1) <= -3*pi/8 %Upper half of sector
        alpha(5, 1) = -pi/4;
```

```
42      elseif alpha(5, 1) > -3*pi/8 && alpha(5, 1) < -pi/2 %Lower half of sector
43          alpha(5, 1) = -pi/2;
44      end
45  end
```

J Wind function

```
function [Wind_force,counter, a_r,C,U_tot]  = fcn(a_fluc, a,U_SV,counter, C_w, U_mea
counter = counter + 1;
a_r = 0; %relative angle
Wind_force = [0 0 0 0 0 0]';

C_x = 0;
C_y = 0;
C_z = 0;
C_theta = 0;
C_phi= 0;
C_psi = 0;
C = zeros(6,1);

%Se om vinkelen er mellom 0 og 2pi
a_r = a+a_fluc-Heading_vessel;
%For testing purpose:
%a_r = u;

%S  hente ut tilh rende verdier fra appendix A
temp = a_r*180/pi;
for i = 1:(length(C_w)-1)
    if(temp >=C_w(i,1)  &&temp <C_w(i+1,1))
C_x = C_w(i,2);
C_y = C_w(i,3);
C_z = C_w(i,4);
C_theta = C_w(i,5);
C_phi= C_w(i,6);
C_psi = C_w(i,7);


    end


%Total wind velocity
U_tot = U_mean + U_SV + U_gust(counter);
    %coefficient matrix
C = [C_x,C_y,C_z,C_theta,C_phi,C_psi]';

Wind_force = abs(U_tot)^2*[C(1),C(2),C(3),C(4),C(5),C(6)]';


```

```
42 end
```

K Extended Kalman filter function

```
function estimatedEtaAndNu = ExtendedKalmanFilter(y,tau, Q, R, x0, P0, D, M_inverse
persistent x_bar
persistent P_bar

if isempty(x_bar) % TODO
    x_bar= x0;
end

if isempty(P_bar)
    P_bar = P0;
end


%%%%%%%%%%%%%%%%%%%% Update A matrix%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s_psi = sin(x_bar(8));
c_psi = cos(x_bar(8));

R_psi = [   c_psi   -s_psi  0;
            s_psi   c_psi   0;
            0       0       1; ];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%% Kalman state matrices %%%%%%%%%%%%%%%%%%%%%%%%%%
% x =[xi_1 til x_6, N(7), E(8), psi(9),b1(10), b2(11), b3(12), u(13),
% v(14),r(15)
% [J]=Jac(psi,b1,b2,u,v, M_inv,D)
PHI=eye(15) + J(x_bar(9), x_bar(10), x_bar(11), x_bar(13), x_bar(14), M_inverse, D)*
GAMMA = h*E;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Kalman algorithm%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kalman gain matrix
K = P_bar*H'*inv(H*P_bar*H'+R);
% Update state estimate
x_hat = x_bar + K*(y-H*x_bar);
% corrector - compute error covariance
P_hat = (eye(15)-K*H)*P_bar*(eye(15)-K*H)'+K*R*K';

x_bar= x_hat + h*(f(x_hat, R_psi, M_inverse, D) + B*tau);
P_bar=PHI*P_hat*PHI' + GAMMA*Q*GAMMA';
```

```
42 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43
44 %%%%%%%%%%%%Output of the desired estimates (eta and nu)%%%%%%%%%%%%%
45 estimatedEtaAndNu=[x_hat(7);x_hat(8);x_hat(9);x_hat(13);x_hat(14);x_hat(15)];
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47 end
```