# TMR4243 Project:
# Final Report:
# *Backstepping control with integral action of 3-DOF DP surface vessel*


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

**Abstract**

This report contains the derivation and stability proof of a backstepping controller with integral action on a surface vessel in 3 degrees of freedom as well as the design of a nonlinear observer with deadreckoning functionality, pathfollowing capabilities and a non optimized extended thrust allocation design for both simulation in Matlab/Simulink, and HIL. The report concludes with satisfactory performance in regard of path following capabilities, observer performance, controller performance and thrust allocation design and introduces further work such as optimizing the thrust allocation.

# Contents

## Sections and corresponding assignments

| Section | Assignments | Section | Assignments |
|---|---|---|---|
| 2.1) | Case A: Task 1.1 and 1.2 | 8.1.1) | Case C, Task 1.1 |
| 2.2) | Fixed Azimuth | 8.1.2) | Case C, Task 1.2 |
| 2.2.1) | Case A, Task 2.1 | 8.1.3) | Case C, task 1.3 |
| 2.2.2) | Case A, Task 2.2 and 2.3 | 8.1.4) | Case C, task 1.4 |
| 2.2.3) | Case A, Task 2.4, 2.5 and 4.2 | 8.2) | Kinematic Control Design |
| 2.3) | Varying Azimuth | 8.2.1) | Case C, task 2.1 |
| 2.3.1) | Case A, Task 3.1, 3.2 and 3.3 | 8.2.2) | Case C, task 2.2 |
| 2.3.2) | Case A, Task 3.4, 3.5 and 4.3 | 8.2.3) | Case C, task 2.3 |
| 3.1) | Simulink implementation | 8.2.4) | Case C, task 2.4 |
| 4.1) | HIL Implementation. Case A, task 5.1, 5.2, 5.3 | 8.3) | Update Laws |
| 4.2) | Case A, task 5.4, 5.5, 5.6 | 8.3.1) | Case C, task 3.1 |
| 5.1) | Case B, task 1.1 and 1.2 | 8.3.2) | Case C, task 3.2 |
| 5.2) | Case B, Reduced observer design | 8.3.3) | Case C, task 3.3 |
| 5.2.1) | Case B, task 2.1 | 8.3.4) | Case C, task 3.4 |
| 5.2.2) | Case B, task 2.2 | 8.4.1) | Case C, task 4.1 |
| 5.2.3) | Case B, task 2.3 | 8.4.2) | Case C, task 4.2 |
| 5.3) | Full observer design | 8.4.3) | Case C, task 4.3 |
| 5.3.1) | Case B, task 3.1 | 8.4.4) | Case C, task 4.4 |
| 5.3.2) | Case B, task 3.2, 3.3. Also in appendix B1 | 8.4.5) | Case C, added integral action |
| 5.4) | Task 3.4) | 9) | Task 5.1 |
| 5.5) | Task 3.5 | 9.1.1) | Case C, task 5.2 |
| 6) | Simulation | 9.1.2) | Case C, task 5.2 |
| 6.1) | Case B, task 4.1 | 9.2 | Case C, task 5.3 |
| 6.1.1) | Case B, task 4.2 | 9.2.1) | Case C, task 5.3 |
| 6.1.2) | Case B, task 4.3 | 9.2.2) | Case C, task 5.3 |
| 6.1.3) | Case B, task 4.4 and 4.5 | 11) | HIL, Case C, task 6.1 |
| 7) | Case B HIL, task 5.1 | 12) | Additional Tasks, task 1.1 |
| 8) | Case C Theory | 13) | Additional Tasks, task 1.2 |
| 8.1) | Path Parameterization | 14) | Additional Tasks, task 1.3 |

Table 1: Table displaying which sections that handle which tasks from the case study.

# Part I
# Preliminaries

## Introduction

This report contains the final report for the group project in TMR4243. The scope of the project is to design a backstepping controller, a nonlinear observer, deadreckoning functionality and thrust allocation for a three degrees of freedom surface vessel with DP capabilites and path following capabilities. As stated in the abstract, and later in the conclusion, the final results were satisfying, and the objective of the project are met in the sections throughout the report.

The report is built as can be seen in the table of contents where first a system dynamic model are presented. This is followed by the various case studies, A, B and C in sequence before the additional simulation tasks are presented. A final conclusion and further work section summarizes how good the design and model performs given the scope of the project.

To keep the final report close to the page limitation, a few figures and mathematical derivations were moved to appendix C for excess figures that we would have included given a longer report, and mathematical derivations were moved to appendix B. Simulink realizations and matlab code can be seen in appendix D and appendix E.

The table in section  connects the sections with the tasks from the case studies.

Further more, a zip file containing runnable files for all simulations shown in the final report and a few that are customized to handle user input are included in the final hand in. All files are based on MasterFile.slx, but slightly modified to recreate the exact simulation that are used in the figures in the report.

## Cooperation

The start of the project started with an even work load distrubution, as it was easier to assign time slots to physically meet up to work on the project. Being present physically also means it was easier to ensure that everyone was working on something. For the later parts of the assignment (from simulation in Case B to the end of the final report), most of the work load was handled by two group members. This was due to their ability to work on their own without being told what to do and will to learn how to solve the bugs.
Of course, we could've been better at delegating parts of the project, but it comes a point in time where it is better to just do it yourself instead of keeping on asking.

Generally debugging and reimplementing parts of the controller, observer and other functionalities was the major time consumption, as most of the theory was correct, according to the guidance assistants, on the first attempt. As for other delays in the project, the HIL-PC was a major contribution. A week was spent on case A trying to get the PC to work, which further delayed every other progress report. Virtual cabling problems with thruster output from control allocation to thrusters in simulink simulation model was also an issue it took far too long to discover.

We agree that it was a rewarding and challenging project, but parts of the group feel that the work load will be skewed because due to the deadline restrictions and that some people are willing to invest more time to manage to solve the problems properly.

# 1 Low speed dynamics and linear thrust mapping

The following section desribes the The vessel C/S Enterprise I low speed dynamics and linear thrust mapping. These equations and matrices are common for all parts of the final report.

$$\begin{aligned} \dot{\eta} &= \mathbf{R}(\psi)\nu \\ \mathbf{M}\dot{\nu} &= -\mathbf{D}\nu + \tau \end{aligned} \tag{1}$$

With a bias the low speed dynamics are given by:

$$\begin{aligned} \dot{\eta} &= \mathbf{R}(\psi)\nu \\ \mathbf{M}\dot{\nu} &= -\mathbf{D}\nu + \mathbf{R}(\psi)^T b + \tau \\ \dot{b} &= 0 \end{aligned} \tag{2}$$

$\tau$ is continuous and bounded. The inertia matrix $\mathbf{M}$ and linear damping matrix are given by:

$$\mathbf{M} = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 24 & 0.53 \\ 0 & 0.53 & 2.8 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 0.66 & 0 & 0 \\ 0 & 1.3 & 2.8 \\ 0 & 0 & 1.9 \end{bmatrix} \tag{3}$$

The rotation matrix transforming from body to NED are given by

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

which satisfies $\frac{d}{dt}\mathbf{R}(\psi) = \mathbf{R}(\psi(t))S(\dot{\psi}(t))$, where $\mathbf{S}(r)$ is given as

$$\mathbf{S}(r) = \begin{bmatrix} 0 & -r & 0 \\ r & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = -\mathbf{S}(r)^T \tag{5}$$

The vessel is modelled with a linear thrust mapping given by.

$$\begin{aligned} \dot{\eta} &= \mathbf{R}(\psi)\nu \\ \mathbf{M}\dot{\nu} &= -\mathbf{D}\nu + \tau \\ \tau &= \mathbf{B}(\alpha)\mathbf{K}\mathbf{u} \end{aligned} \tag{6}$$

In the linear thrust mapping $\tau = \mathbf{B}(\alpha)\mathbf{K}\mathbf{u}$ is the configuration setup where $\tau \in \mathbb{R}^3$ is body frame thrust vector, $u \in \mathbb{R}^3$ is the input vector and $\alpha \in \mathbb{R}^2$ is the VSP angles.
The linear thrust mapping assumes that thruster $i$ produces thrust $T_i = k_i u_i$, where K is given as

$$\mathbf{K} = \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & K_3 \end{bmatrix}, k_1 = 1.030, k_2 = 1.030, k_3 = 2.629$$

The VSP thruster input $u_1$ and $u_2$ takes value $\in [0, 1]$ and angles $\alpha = [\alpha_1, \alpha_2] \in (-\pi, \pi]$. The bow thruster input $u_3$ takes value in $[-1, 1]$.

# Part II
# Case Study A

This part of the report contains answers to the assignments in case study A, where a fixed and varying azimuth thrust allocation is derived. Various results from the case study are presented with relevant figures added.

## 2 Theory

### 2.1 C/S Enterprise I - Modeling

A hand drawn figure showing the C/S Enterprise with its thruster configuration, NED axis and Body frame can be seen under in Figure 1.

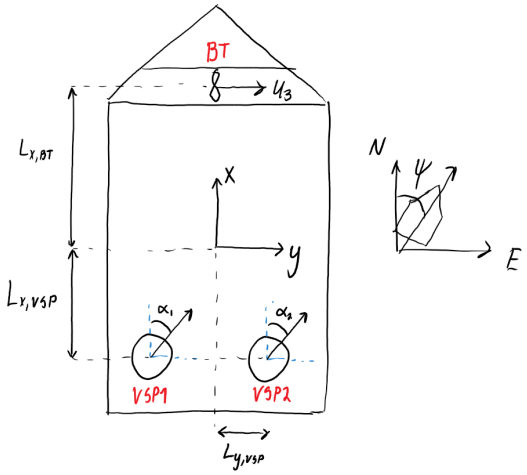



Figure 1: Thruster configuration of C/S Enterprise I

An accompanynig table of thruster coordinates were derived as follow, where the distances are the distance away from the origin of the fixed body frame.

| Thruster name | Number $[i]$ | $l_{i,x}$ [m] | $l_{i,y}$ [m] |
|---|---|---|---|
| VSP 1 | 1 | -0.4574 | -0.055 |
| VSP 2 | 2 | -0.4574 | 0.055 |
| Bow thruster | 3 | 0.3875 | 0 |

Table 2: Thruster coordinates

## 2.2 Fixed Azimuth thrust allocation

The fixed thrust implementation can be seen in appendix E, section E, and in figure D in appendix D.

One can find a fixed azimuth thrust allocation by inverting the thrust mapping when $u$ and $\alpha$ takes a valid value. I.e find a valid $u_{cmd}$ and $\alpha_{cmd}$ that yields.

$$\tau_{ref} = \mathbf{B}(\alpha_{cmd})\mathbf{K}\mathbf{u}_{cmd}$$

### 2.2.1 Derivation of thrust configuration matrix $\mathbf{B}(\alpha)$

The thrust configuration matrix were derived as follows. We define the thrust load vector as:

$$\tau = \begin{bmatrix} f \\ r \times f \end{bmatrix}, r = [l_x, l_y, l_z], f = [F_x, F_y, F_z]$$

where the cross product yields:

$$r \times F = \begin{vmatrix} \hat{i} & \hat{j} & k \\ l_x & l_y & l_z \\ F_x & F_y & F_z \end{vmatrix}, \Rightarrow \tau = \begin{bmatrix} F_x \\ F_y \\ F_z \\ l_y F_z - l_z F_y \\ l_z F_x - l_x F_z \\ l_x F_y - l_y F_x \end{bmatrix} \Rightarrow \tau = \begin{bmatrix} F_x \\ F_y \\ l_x F_y - l_y F_x \end{bmatrix} \tag{7}$$

The last simplification is that we reduce from 6 DOF to 3 DOF and only consider surge, sway and yaw.

Each thruster will give us a contribution in x, y and moment about the center of gravity. The VSPs are implemented in the same manner. The bow thruster is a tunnel thruster that can only provide thrust in sway.

Contribution from each thruster in surge:

$$f_{x1} = T_1 \cos(\alpha_1), f_{x2} = T_2 \cos(\alpha_2), f_{x3} = 0$$

Contribution from each thruster in sway:

$$f_{y1} = T_1 \sin(\alpha_1), f_{y2} = T_2 \sin(\alpha_2), f_{y3} = T_3$$

Moment contribution in yaw:

$$\begin{gathered} m_{z1} = l_{x1} f_{y1} - l_{y1} f_{x1} = l_{x1} T_1 \sin(\alpha_1) - l_{y1} T_1 \cos(\alpha_1), \\ m_{z2} = l_{x2} f_{y2} - l_{y2} f_{x2} = l_{x2} T_2 \sin(\alpha_2) - l_{y2} T_2 \cos(\alpha_2), \\ m_{z3} = l_{x3} f_{y3} = l_{x3} T_3 \end{gathered}$$

This gives us the following thrust configuration matrix

$$B(\alpha) = \begin{bmatrix} cos(\alpha_1) & cos(\alpha_2) & 0 \\ sin(\alpha_1) & sin(\alpha_2) & 1 \\ l_{x1} sin(\alpha_1) - l_{y1} cos(\alpha_1) & l_{x2} sin(\alpha_2) - l_{y2} cos(\alpha_2) & l_{x3} \end{bmatrix}$$

Using the theory from p.401 in Fossen 2011 [1], we see that this corresponds well with thruster configuration characteristics for a 3 DOF azimuth thruster and tunnel thruster.

### 2.2.2 Physical representation of K, u and B($\alpha$)

The vector u is the magnitude of thrust each thruster will give from zero to one where one is max. K scales the thrust from the range 0-1 to the real value this means that $K_u$ represents the forces from each thruster. $B(\alpha)$ transforms the forces of each thruster to the net force in surge, sway and yaw.

The physical significance of $B(\alpha)$ beeing singular is that we can't generate force or moment in the direction where it is singular. From page 4-5 in Johansen 2013 [6] : "Rank-deficiency of B means that no force or moment can be generated in certain direction of the space R(m) where $\tau_c$ belongs. This means that all commands $\tau_c$ cannot be achieved, even without considering saturation...."

The values for $\alpha$ that gives a singular B($\alpha$) is: $\alpha = \frac{\pi}{2}$. The singular values was obtained using symbolic math toolbox in Matlab and solving the $det(\mathbf{B}) = 0$.

### 2.2.3 Procedure $\beta$ and fixed azimuth implementation

The procedure $\beta(u_*, \alpha)$ can be implemented as a for-loop which tests the u values for thruster 1 and 2. For each thruster it tests that if $u < 0$. If this holds true, the thruster actuation $u$ is set to $|u|$ and alpha is rotated 180 degrees($\alpha = \alpha + \pi$). In order to make sure alpha is whithin the defined interval, that is $(-\pi, \pi]$ another if statement is implemented which either adds $2\pi$ or subtracts $2\pi$. The corresponding matlab code can be found in appendix E.

The results for three cases of input [1]:

$$1): \begin{bmatrix} \tau_{ref1} \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \frac{1}{2} \\ \frac{\pi}{2} \\ \frac{\pi}{2} \end{bmatrix} \Rightarrow \begin{bmatrix} u_{cmd1} \\ \alpha_{cmd1} \end{bmatrix} = \begin{bmatrix} 0.06461 \\ 0.0646 \\ 0.4310 \\ -\frac{\pi}{2} \\ -\frac{\pi}{2} \end{bmatrix}, 2): \begin{bmatrix} \tau_{ref2} \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ \pi \\ \pi \end{bmatrix} \Rightarrow \begin{bmatrix} u_{cmd2} \\ \alpha_{cmd2} \end{bmatrix} = \begin{bmatrix} 0.0709 \\ 0.9709 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$3): \begin{bmatrix} \tau_{ref3} \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \frac{1}{2} \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} u_{cmd3} \\ \alpha_{cmd3} \end{bmatrix} = \begin{bmatrix} 0.50751 \\ 1 \\ 0.3804 \\ \pi \\ 0 \end{bmatrix}$$

From the results of running the algorithm it is seen that $0 <= u_1 <= 1$ , $0 <= u_2 <= 1$ and $-\pi < \alpha <= \pi$. The inversion works, even when we have a singularity for angle $\frac{\pi}{2}$ due to the use of the pseudoinverse *pinv* in Matlab.

## 2.3 Varying azimuth thrust allocation

The varying thrust implementation can be seen in appendix E, section E, and in figure D in appendix D

[1] Corresponding to task 2.5 and 4.2 from case A

### 2.3.1 Procedure and derivation of the extended thrust allocation

Following the assignment from case study A, we devise the following procedure to ensure validity of the inversion. The given $\check{u}$ contains the vertical and horizontal components of $u$ and thus the inverse transformation is equivalent to the composition of forces. The following formula is used to implement the transformation:

$$u_i = \sqrt{\check{u}_{ix}^2 + \check{u}_{iy}^2}, \quad \alpha_i = atan(\frac{\check{u}_{iy}}{\check{u}_{ix}}) \tag{8}$$

In order to ensure that the angles are within $(-\pi, \pi]$, function $atan2$ is applied instead of $atan$, since the output range of $atan$ is $[-\pi/2, \pi/2]$.
The linear thrust map can be stated as follows:

$$\tau = \check{\mathbf{B}}\check{\mathbf{K}}\check{\mathbf{u}}, \tag{9}$$

where $\check{\mathbf{B}} \in \mathbb{R}^{3\times5}$, $\check{\mathbf{K}} \in \mathbb{R}^{5\times5}$, and $\check{u} = [\check{u}_{1x}, \check{u}_{1y}, \check{u}_{2x}, \check{u}_{2y}, \check{u}_3]^T$.

Using the extended thrust allocation method for rotatable actuators, which can be seen on p.402 in Fossen 2011 [1], we can decompose the thrust configuration from task two, which is nonlinear in $\alpha$, to surge and sway components.

This gives us the following extended thruster configuration matrix $\check{\mathbf{B}}$ as:

$$\check{\mathbf{B}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ l_{y1} & l_{x1} & l_{y2} & l_{x2} & l_{x3} \end{bmatrix}$$

The $\check{\mathbf{K}}$ is also extended where the VSP's get two columns each. The values for $k_1$ corresponds to the first VSP k-coefficient in sway, $k_2$ corresponds to the first VSP k-coefficient in sway, etc. The k-components for each VSP is calculated by solving the equation under, where i = 1 = surge component, i = 2 = sway component. We assume that the components in surge and sway should be equal to each other, since the VSP is symmetrical, and that the k-coefficients for VSP1 = VSP2. This is further supported by p.403 in Fossen [1], which

$$K = \begin{bmatrix} k_1 & 0 & 0 & 0 & 0 \\ 0 & k_2 & 0 & 0 & 0 \\ 0 & 0 & k_3 & 0 & 0 \\ 0 & 0 & 0 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_5 \end{bmatrix}, k_1 = k_2 = k_3 = k_4 = 1.030, k_5 = 2.629$$

The complete extended thrust vector is now written out from eq. 9:

$$\tau = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ l_{y1} & l_{x1} & l_{y2} & l_{x2} & l_{x3} \end{bmatrix} \begin{bmatrix} k_1 & 0 & 0 & 0 & 0 \\ 0 & k_1 & 0 & 0 & 0 \\ 0 & 0 & k_2 & 0 & 0 \\ 0 & 0 & 0 & k_2 & 0 \\ 0 & 0 & 0 & 0 & k_3 \end{bmatrix} \begin{bmatrix} \check{u}_{1x} \\ \check{u}_{1y} \\ \check{u}_{2x} \\ \check{u}_{2y} \\ \check{u}_3 \end{bmatrix} \tag{10}$$

Here, $\check{K}\check{u}$ represents the forces each thruster produces in surge and sway direction. $\check{B}$ is the transformation from decomposed thruster forces to forces in surge, sway and moment in yaw. This means that $\check{B}$ has to be an $\mathbb{R}^{3\times5}$ matrix.

### 2.3.2 Varying azimuth implementation

The varying thrust implementation and yielded the following results, Case 1:

$$\tau_{ref} = [1 \quad 1 \quad 0.5]^T \tag{11}$$

yields:

$$u_{cmd} = [0.4746 \quad 0.5046 \quad 0.4302]^T, \alpha_{cmd} = [-0.1345 \quad -0.1265]^T \tag{12}$$

For case 2:

$$\tau_{ref} = [2 \quad 0 \quad 0]^T \tag{13}$$

yields:

$$u_{cmd} = [0.9709 \quad 0.9709 \quad 0]^T, \ \alpha_{cmd} = [0 \quad 0]^T \tag{14}$$

From the results of running the algorithm it is seen that $0 \leq u_1 \leq 1$ , $0 \leq u_2 \leq 1$ and $-\pi < \alpha \leq \pi$

The extended thrust allocation algorithm we have derived and implemented on the previous task is an unconstrained algorithm, meaning there are no bounds on the vector elements in $f_i$, $u_i$, and their time derivatives. This will yield an unrealistic model for how the actuators will move in reality. It is possible to find a "optimal" distribution of control forces for each DOF, since we have an equal or larger amount of control inputs than degrees of freedom, through explicit methods (p404 Fossen 2011 [1]).

# 3 Simulation

The results from the simulation regarding case study A are presented in section 2.2.3 and section 2.3.2.

## 3.1 Implementation

A wrapper function was made as seen in figure 20 in appendix D. Implementation of the thrust allocation can be seen in figure 21 in appendix D, where switching between the two thrust allocation methods can be done with a manual switch. For further implementations we only consider the extended thrust allocation.

# 4 HIL test

## 4.1 Input ranges and implementation

The mapping of the connections from the joystick to the simulink model and output from the simulink model to the GUI were found by experimenting with the HIL computer. The input ranges from the two joysticks are as shown in table 3.

where L2, R2, Xjoy and Yjoy are continuous and L1, R1, left, right, down, up, select and start are discrete with feasible values 0 and 1. Besides, it is worth mentioning that the initial values for R2 and L2 (without pressing) are -1, while those of Xjoy and Yjoy are 0.

Table 3: Input ranges for the joystick

| Task 5.1 | **Range:** | |
|---|---|---|
| **Button** | **from** | **to** |
| L2 | -1 | 1 |
| R2 | -1 | 1 |
| Xjoy | -1 | 1 |
| Yjoy | -1 | 1 |
| L1 | 0 | 1 |
| R1 | 0 | 1 |
| left | 0 | 1 |
| right | 0 | 1 |
| down | 0 | 1 |
| up | 0 | 1 |
| select | 0 | 1 |
| start | 0 | 1 |

The thrust allocation algorithm were implemented in Veristand as illustrated in Figure 2.The trust allocation limits to only use the VSPs because you are not controlling sway and thus it should be zero. Any use of the bow thrusters will generate force in sway direction which cannot be compensated.

The inputs for the extended/varying thrust allocation are somewhat different from the fixed thrust allocation implementation. Y and X position of joystick should control the surge and sway force respectively while R2 and L2 should control the yaw moment, clockwise and counterclockwise respectively. However, as is emphasized above, the initial values for L2 and R2 are both -1. In order to compensate this, constant 1 should be added to both inputs. What's more, it would be ideal if the control input for yaw moment is the difference between R2 and L2.

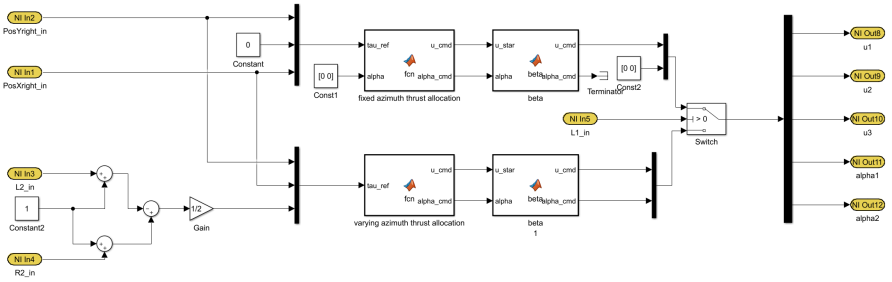



Figure 2: Simulink implementation for ctrl_custom

## 4.2 Graphical interface

To switch between two modes of thrust allocation, the unused button L1 is applied. When L1 is being pressed, the fixed azimuth thrust allocation is activated while the varying azimuth thrust allocation is activated without L1 being pressed.

Graphical interface is shown in Figure 3. All three inputs for thrusters and the VSP angles are presented. It is believed that these values are enough to evaluate the outcomes of thrust allocation. Also, two graphs are introduced to present the position of the vessel in 3 DOFs.

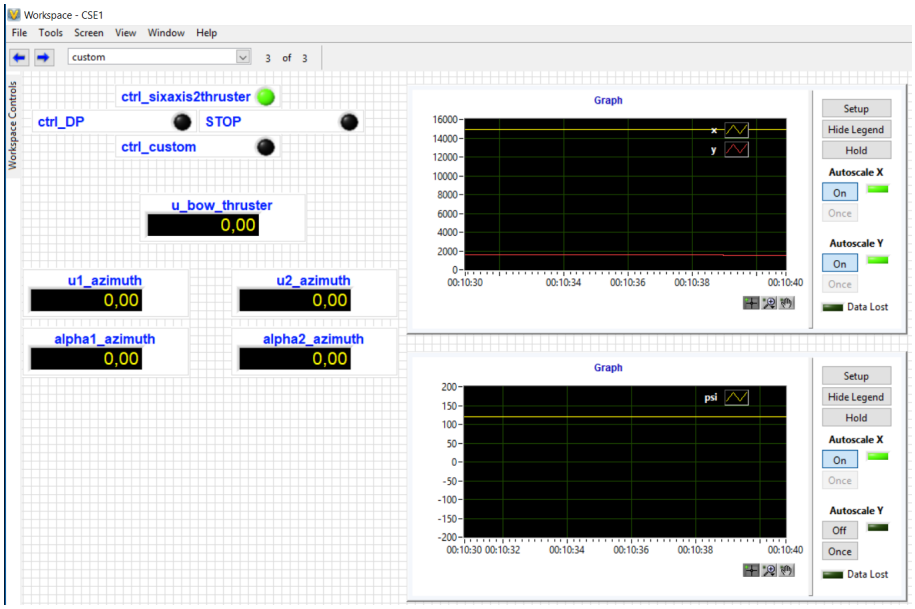



Figure 3: Simulink implementation for ctrl_custom

As in shown in Figure 3, two graphs are applied to present the position of the simulated vessel. Both graphs here can function well and they indicate that the thrust allocation algorithm works as expectation.

# Part III
# Case Study B

This part contains the derivations of equations and stability proofs for a nonlinear observer. This is followed by testing the observers robustness in open loop simulation with various measurement noise. The model is also augmented to include deadreckoning and a signal failure simulation.

## 5 Theory

### 5.1 Modeling

The variables are defined as (p.19 Fossen 2011 [1]):

$$\eta = \begin{bmatrix} \eta_N \\ \eta_E \\ \psi \end{bmatrix} \in \mathbb{R}^3 \in NED, \nu = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \in \mathbb{R}^3 \in BODY, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \in \mathbb{R}^3 \in NED, \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \in \mathbb{R}^3 \in NED$$

Where $\eta$ and $\nu$ is the generalized position and velocities used to describe motion in 3 degrees of freedom. $\tau$ is defined as a vector of forces and moments or generalized forces. **b** is the bias term and counts for unmodeled uncertainties, forced (p.13 Fossen 2011 [1]).

$M = M_{RB} + M_A$ is the system inertia matrix consisting of the rigid body mass and the added mass. $D$ is the damping matrix, which for low speeds are dominated by linear damping terms. (p110 and p82, Fossen 2011 [**Fossen2011**]).

To verify if M is a symmetric matrix, the MATLAB function *transpose()* were applied. To verify that the matrix also is positive definite, we use the property that if all the eigenvalues are positive, then the matrix is positive definite. Using command *eig()* in Matlab, we get the following eigenvalues for $M$ and $D + D^T$:

$$\lambda_{\mathbf{M,i}} = \begin{bmatrix} 2.7868 \\ 16 \\ 25.0132 \end{bmatrix}, \lambda_{\mathbf{D+D'}} = \begin{bmatrix} 0.3364 \\ 1.3200 \\ 6.0636 \end{bmatrix}$$

Thus both $M$ and $D + D^T$ are positive definite.

## 5.2 Simplified Observer Design

In this section we will present the simplified observer design for our model in the absence of bias. We assume that $\eta$ and $\tau$ are measured, their estimates are denoted as $\hat{\eta}$ and $\hat{\tau}$ and the errors are defined as $\tilde{\eta} = \eta - \hat{\eta}$, and $\tilde{\nu} = \nu - \hat{\nu}$.

### 5.2.1 Presenting the simplified observer

Our system is defined as:

$$\begin{aligned} \dot{\eta} &= \mathbf{R}(\psi)\nu \\ \mathbf{M}\dot{\nu} &= -\mathbf{D}\nu + \mathbf{R}(\psi)^T b + \tau \\ \dot{\mathbf{b}} &= 0 \end{aligned} \tag{15}$$

which reduces down to the following system model when b = 0:

$$\begin{aligned} \dot{\eta} &= \mathbf{R}(\psi)\nu \\ \mathbf{M}\dot{\nu} &= -\mathbf{D}\nu + \tau \end{aligned} \tag{16}$$

The task at hand is to present the observer that corresponds to the error dynamics given by equation 6 in the case study.

$$\begin{aligned} \dot{\tilde{\eta}} &= \mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta} \\ \mathbf{M}\dot{\tilde{\nu}} &= -\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T L_2\tilde{\eta} \end{aligned} \tag{17}$$

Introducing $y = \eta$ and replicating the measurement with an estimate of the measurement we get $\hat{y} = \hat{\eta}$, and the measurement error as $\tilde{y} = y - \hat{y} = \eta - \hat{\eta}$. Following the method used for deriving for the Luenberger estimator.

**Step 1:** Replicate our system as an estimate with injection gains ($L_1$ and $L_2$) and measurement error $\tilde{y}$:

$$\begin{aligned} \dot{\hat{\eta}} &= \mathbf{R}(\psi)\hat{\nu} + L_1\tilde{\eta} \\ \mathbf{M}\dot{\hat{\nu}} &= -\mathbf{D}\hat{\nu} + \tau + \mathbf{R}(\psi)^T L_2\tilde{\eta} \end{aligned} \tag{18}$$

**Step 2:** Calculate the error dynamics: Although Step 1 is sufficient to answer the task at hand, we can prove that this observer corresponds to the error dynamics. This is done by inserting the result from Step 1 and the reduced simplified system dynamics into $\tilde{\eta} = \eta - \hat{\eta}$ and $\tilde{\nu} = \nu - \hat{\nu}$, differentiating and calculating, which gives

$$\begin{aligned} \dot{\tilde{\eta}} = \dot{\eta} - \dot{\hat{\eta}} &= \mathbf{R}(\psi)\nu - \mathbf{R}(\psi)\hat{\nu} - L_1\tilde{\eta} \\ \Rightarrow \dot{\tilde{\eta}} &= \mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta} \end{aligned}$$

and

$$\begin{aligned} \mathbf{M}\dot{\tilde{\nu}} = \mathbf{M}(\dot{\nu} - \dot{\hat{\nu}}) &= -\mathbf{D}\nu + \tau + \mathbf{D}\hat{\nu} - \tau - \mathbf{R}(\psi)^T L_2\tilde{\eta} \\ &= -\mathbf{D}(\nu + \hat{\nu}) - \mathbf{R}(\psi)^T L_2\tilde{\eta} \\ \Rightarrow \mathbf{M}\dot{\tilde{\nu}} &= -\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T L_2\tilde{\eta} \end{aligned}$$

#### 5.2.2 Autonmous or nonautonomous and linear or nonlinear

Following the definition from Fossen 2011 [1], and Khalil [7], a nonlinear system $\dot{x} = f(x)$ is a an autonomous system if $f(x)$ does not explicitly depend on t. Furthermore, the rotation matrix $\mathbf{R}(\psi)$ is nonlinear and time dependent, thus making our system a nonlinear, non-autonomous system because the heading $\psi$ depends on time.

#### 5.2.3 Stability proof of the simplified error dynamics

The task at hand is to show that the equilibrium $(\tilde{\eta}, \tilde{\nu}) = 0$ is UGES for the observer derived in sec 5.2.1.

**Step 0: Properties of injection gains and matrix P**: We have that $L_1 = L_1^T > 0$, $L_2 = L_2^T > 0$, and $L_1L_2 + L_2L_1 > 0$.

$$\mathbf{P} = \begin{bmatrix} L_2 & 0 \\ 0 & M \end{bmatrix}$$

**Step 1: Introducing our Lyapunov function candidate**:

$$V(\tilde{\eta}, \tilde{\nu}) = x^T\mathbf{P}x = \tilde{\eta}^T L_2\tilde{\eta} + \tilde{\nu}^T\mathbf{M}\tilde{\nu}, \in \mathbf{C}^1, > 0, \forall(\tilde{\eta}, \tilde{\nu}) \neq 0 \tag{19}$$

The Lyapunov function candidate is positive since $L_2$ and $\mathbf{M}$ are positive definite. (p.1-2, Svenn and Roger, [4])

**Step 2: Time differentiating** $V(\tilde{\eta}, \tilde{\nu})$(the derivation process can be found in Appendix B.2):

$$\dot{V} = \dot{\tilde{\eta}}^T L_2 \tilde{\eta} + \tilde{\eta}^T L_2 \dot{\tilde{\eta}} + \dot{\tilde{\nu}} \mathbf{M} \tilde{\nu} + \tilde{\nu}^T \mathbf{M} \dot{\tilde{\nu}}$$

$$\Rightarrow \dot{V} = -\tilde{\eta}(L_1^T L_2 + L_2 L_1)\tilde{\eta} - \tilde{\nu}^T(\mathbf{D}^T + \mathbf{D})\tilde{\nu} \tag{20}$$

**Step 3: Conclusion:** Following proof 1 given in the lecture note for *Observer for simplifed DP model: Design and proof* [3], and the results from section 5.1, we now have that $\dot{V}$ is negative definite and the equilibrium point is UGES due to the conditions

$$\mathbf{D} + \mathbf{D}^T > 0$$
$$L_1 L_2 + L_2 L_1 = L_1^T L_2 + L_2 L_1 > 0$$

being satisfied. Note that this is only UGES for the simplified system, and not for the overall system, as we will introduce bias later.

## 5.3 Full observer design

In this section we will design a full state observer for our case study. The complete system we're modeling is:

$$\begin{aligned} \dot{\eta} &= \mathbf{R}(\psi)\nu \\ \mathbf{M}\dot{\nu} &= -\mathbf{D}\nu + \mathbf{R}(\psi)^T b + \tau \\ \dot{\mathbf{b}} &= 0 \end{aligned} \tag{21}$$

### 5.3.1 Presenting the observer

Replicating the process in section 5.2.1, we get the the observer:

$$\begin{aligned} \dot{\hat{\eta}} &= \mathbf{R}(\psi)\hat{\nu} + L_1\tilde{\eta} \\ \mathbf{M}\dot{\hat{\nu}} &= -\mathbf{D}\hat{\nu} + \mathbf{R}(\psi)^T\hat{b} + \tau + \mathbf{R}(\psi)^T L_2\tilde{\eta} \\ \dot{\hat{b}} &= L_3\tilde{\eta} \end{aligned} \tag{22}$$

Which gives the error dynamics as:

$$\begin{aligned} \dot{\tilde{\eta}} &= \mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta} \\ \mathbf{M}\dot{\tilde{\nu}} &= -\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T\tilde{b} - \mathbf{R}(\psi)^T L_2\tilde{\eta} \\ \dot{\tilde{b}} &= -L_3\tilde{\eta} \end{aligned} \tag{23}$$

### 5.3.2 Positive definiteness of $P$

The proving process of the positive definiteness of $P$ can be found in Appendix B.1.

### 5.3.3 Stability of the full observer design with LaSalle-Yoshizawa

The task at hand is to show that the equillibrium $(\tilde{\eta}, \tilde{\nu}, \tilde{b}) = 0$ is UGS. This is shown through the following steps:

**Step 0: Properties of injection gains and matrix P:**
P $> 0$ iff $L_1L_2 - L_3 > 0$, and that we can assume that$L_1L_2 - L_3 > 0$.
**Step 1: Introducing our Lyapunov function candidate (LFC)**:

$$W_1(\tilde{\eta}, \tilde{\nu}, \tilde{b}) = x^T\mathbf{P}x := \tilde{\eta}^T L_2\tilde{\eta} + \tilde{\nu}^T\mathbf{M}\tilde{\nu} + \tilde{b}^T L_3^{-1}L_1\tilde{b} - 2\tilde{\eta}^T\tilde{b} > 0, \forall(\tilde{\eta}, \tilde{\nu}) \neq 0 \tag{24}$$

In other words the LFC is a at least one time continuously differentiable function, which positive definite $\forall(\tilde{\eta}, \tilde{\nu}, \tilde{b}) \neq 0 \Rightarrow$ lower bounded by 0. Furthermore, the LFC is also radially unbounded.

**Step 2: Time differentiating the LFC:**
When time differentiating equation 24 we get the following result seen in equation 25. The detailed derivation process can be found in Appendix B.3:

$$\dot{W}_1 = -2\tilde{\eta}^T[L_1L_2 - L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu} \tag{25}$$

Using the results from Step 0 in sec 5.3.3, we have that $\dot{W}_1(\tilde{\nu}, \tilde{\eta}) < 0$, $\forall(\tilde{\nu}, \tilde{\eta}) \neq 0$, meaning that $W_1(\tilde{\nu}, \tilde{\eta}, \tilde{b})$ is non-increasing.

**Step 3: Finding $\mathbf{Y}(\tilde{\eta}, \tilde{\nu})$:**
The final step before we can prove UGS for our error dynamics is to find the continous function $Y_1(\tilde{\eta}, \tilde{\nu})$ listed as equation 8 in the assignment *Case Study B*.

$$\begin{aligned}\dot{W}_1 &= -2\tilde{\eta}^T[L_1L_2 - L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu} \\ &\leq -\tilde{\eta}^T[L_1L_2 - L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu} = Y(\tilde{\eta}, \tilde{\nu}) \leq 0\end{aligned}$$

As can be seen, $Y_1(\tilde{\eta}, \tilde{\nu})$ is a continous function that also satisfies the conditions $\dot{W}_1 \leq Y_1(\tilde{\eta}, \tilde{\nu}) \leq 0$. It also implies that if $Y_1(\tilde{\eta}, \tilde{\nu}) = 0 \iff (\tilde{\eta}, \tilde{\nu}) = 0$.

**Step 4: Applying LaSalle-Yoshizawa:**
Following LaSalle-Yoshizawa's Theorem, found in section A.1:
Condition i) has been met in section 5.3.3, Step 1.
Condition ii) has been met in section 5.3.3 Step 2, and Step 3, with negative semidefiniteness; since our differentiated LFC does not tell us anything about the trajectory of $\tilde{b}$, we can not conclude with it beeing negative definite. Allthough we know that the trajectory of $\tilde{b}$ has to converge to something, due to the error dynamics given in section 5.3.1, where it is scaled by something that is nonsingular (the rotation matrix).
Condition iii) has been met in section 5.3.3, Step 1.

LaSalle-Yoshizawa's theorem thus states that all solutions $\mathbf{x}(t) = \mathbf{f}(\mathbf{x}, t)$ are uniformly globally bounded and:

$$\lim_{t\to\infty} W(x(t)) = 0 \Rightarrow \lim_{t\to\infty} Y(\tilde{\eta}, \tilde{\nu}) = 0$$

And following the final implication in section 5.3.3, Step 3, we have that

$$\lim_{t\to\infty} Y(\tilde{\eta},\tilde{\nu}) = \lim_{t\to\infty} (\tilde{\eta}(t),\tilde{\nu}(t)) = 0$$

Thus the equilibrium is UGS.

#### 5.3.4 Auxiliary function 2

**Step 1: Conditions** The task at hand is to find a continous function $Y_2$ that satisfies

$$\dot{W}_2(\tilde{\eta},\tilde{\nu},\tilde{b},t) \leq Y_2(\tilde{\eta},\tilde{\nu},\tilde{b},\phi(t)) \tag{26}$$

Where we define $\phi(t) = (cos(\psi(t)), sin(\psi(t)), \dot{\psi}(t))$. Furthermore, we also want to ensure that the following two conditions are satisfied:

$$Y_1(\tilde{\eta},\tilde{\nu}) = 0 \Rightarrow Y_2(\tilde{\eta},\tilde{\nu},\tilde{b},\phi(t)) \leq 0 \tag{27}$$

$$Y_1(\tilde{\eta},\tilde{\nu}) = Y_2(\tilde{\eta},\tilde{\nu},\tilde{b},\phi(t)) = 0 \Rightarrow (\tilde{\eta},\tilde{\nu},\tilde{b}) = 0 \tag{28}$$

**Step 2: Differentiating**: Using

$$W_2(\tilde{\eta},\tilde{\nu},\tilde{b},t) = -\tilde{b}^T R(\psi) M\tilde{\nu} \tag{29}$$

And time differentiationg to find $\dot{W}_2(\tilde{\eta},\tilde{\nu},\tilde{b},t)$:

$$\begin{aligned}
\dot{W}_2(\tilde{\eta},\tilde{\nu},\tilde{b},t) &= -\dot{\tilde{b}}^T R(\psi)\mathbf{M}\tilde{\nu} - \tilde{b}^T(\dot{\mathbf{R}}(\psi)\mathbf{M}\tilde{\nu} + \mathbf{R}(\psi)\mathbf{M}\dot{\tilde{\nu}}) \\
&= (L_3\tilde{\eta})^T\mathbf{R}(\psi)\mathbf{M}\tilde{\nu} - \tilde{b}^T\mathbf{R}(\psi)S(\dot{\psi})\tilde{\nu} - \tilde{b}^T\mathbf{R}(\psi)(-D\tilde{\nu} + \mathbf{R}(\psi)^T\tilde{b} - \mathbf{R}(\psi)^T L_2\tilde{\eta}) \\
&= \tilde{\eta}^T L_3\mathbf{R}(\psi)\mathbf{M}\tilde{\nu} - \tilde{b}^T\mathbf{R}(\psi)S(\dot{\psi})\tilde{\nu} + \tilde{b}^T\mathbf{R}(\psi)\mathbf{D}\tilde{\nu} - \tilde{b}^T\tilde{b} + \tilde{b}^T L_2\tilde{\eta} \\
&:= Y_2(\tilde{\eta},\tilde{\nu},\tilde{b},\phi(t))
\end{aligned} \tag{30}$$

**Step 3: Conclusion** From this it can be seen that $Y_2$ is a function of $\phi$, since $R = R(\psi(t))$ and $\dot{\psi} = \dot{\psi}(\phi)$
If $Y_1 = 0$ gives $(\tilde{\eta},\tilde{\nu}) = (0,0)$ then

$$Y_2 = -\tilde{b}^T\tilde{b} \leq 0 \tag{31}$$

When $Y_2 = 0$ then $\tilde{b} = 0$ This gives

$$Y_1 = Y_2 = 0 \implies (\tilde{\eta},\tilde{\nu},\tilde{b}) = 0 \tag{32}$$

,
And thus the conditions in Step 1 have been met.

### 5.4 Stability of the full observer design - Matrosov

The current objective is to prove that the equilibrium $(\tilde{\eta},\tilde{\nu},\tilde{b}) = 0$. This is done by using Matrosov's nested theorem, which can be found in Appendix A.2, and the conditions listed in step 1 in the following segment.

**Step 1: Conditions** We define a Lyapunov function $V$ and two auxiliary functions $W_1$ and $W_2$ from section 5.3.3 and section 5.3.4. The time derivative of these functions are bounded from above by $Y_0, Y_1$ and $Y_2$ which are continuous negative definite functions.

$$V = \tilde{\eta}^T L_2 \tilde{\eta} + \tilde{\nu}^T \mathbf{M} \tilde{\nu} \tag{33}$$

$$\Rightarrow \dot{V} = -\tilde{\eta}(L_1^T L_2 + L_2 L_1)\tilde{\eta} - \tilde{\nu}^T(\mathbf{D}^T + \mathbf{D})\tilde{\nu} := Y_0(\tilde{\eta}, \tilde{\nu}) \leq 0 \tag{34}$$

$$W_1(\tilde{\eta}, \tilde{\nu}, \tilde{b}) = x^T \mathbf{P} x := \tilde{\eta}^T L_2 \tilde{\eta} + \tilde{\nu}^T \mathbf{M} \tilde{\nu} + \tilde{b}^T L_3^{-1} L_1 \tilde{b} - 2\tilde{\eta}^T \tilde{b} \tag{35}$$

$$\Rightarrow \dot{W}_1 = -2\tilde{\eta}^T [L_1 L_2 - L_3]\tilde{\eta} - \tilde{\nu}^T [\mathbf{D}^T + \mathbf{D}]\tilde{\nu} \tag{36}$$

$$\leq -\tilde{\eta}^T [L_1 L_2 - L_3]\tilde{\eta} - \tilde{\nu}^T [\mathbf{D}^T + \mathbf{D}]\tilde{\nu} = Y_1(\tilde{\eta}, \tilde{\nu}) \leq 0 \tag{37}$$

$$W_2 = -\tilde{b}^T \mathbf{R}(\psi) \mathbf{M} \tilde{\nu} \tag{38}$$

$$\Rightarrow \dot{W}_2 = \tilde{\eta}^T L_3 \mathbf{R}(\psi) \mathbf{M} \tilde{\nu} - b^T (\dot{\mathbf{R}}(\psi)\mathbf{M} - \mathbf{R}(\psi)\mathbf{D})\tilde{\nu} - \tilde{b}^T \tilde{b} + \tilde{b}^T L_2 \tilde{\eta} := Y_2 \leq 0 \tag{39}$$

From Matrosov's nested theorem we know that if $\dot{V}_0 = 0 \Rightarrow \tilde{\eta} = \tilde{\nu} = 0 \Rightarrow Y_0 = Y_1 = 0$ and $Y_2 = Y_1 = Y_0 = 0 \Rightarrow \tilde{\eta} = \tilde{\nu} = \tilde{b} = 0$. Assumptions:

1. The origin of the system is UGS from section 5.3.3.
2. We have $V_1, W_1$ and $W_2$ and a number $\mu$ that bounds $V_1, W_1$ and $W_2$ such that $max(V_1, W_1, W_2, \phi) \leq \mu$. We known this because $\psi$ and $\dot{\psi}$ is bounded which implies that $Y_2$ and $\phi$ is bounded.
3. $\dot{V}, \dot{W}_1, \dot{W}_2 \leq Y_i$ for i = 0,1,2.

# 6 Simulation

In this part the nonlinear observer was implemented and tested the open loop.

## 6.1 The full observer performance

The full observer can be seen implemented in figure 22 in appendix D, and are modeled after the observer equations given in section 5.3.1. The selected observer gains were initially set according to Fossens example for nonlinear passive DP observer (p. 319 Fossen 2011, [1]), and then tuned slightly to get a quick enough response time. Although the values might be a bit aggressive concidering the scale of the model, the final values we landed on were:

$$\mathbf{L_1} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \mathbf{L_2} = 20\mathbf{M} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{L_3} = 0.1 * \mathbf{L}_2$$

These values also ensured that the condition $L_1 L_2 - L_3 > 0$ is met.

### 6.1.1 Observer performance with bias

The artificial bias $b = [0.8 \quad -0.3 \quad 0]^T$ was implemented as instructed. Figure 4 illustrates the observers performance compared with the true value of the position and the error between the estimated and true value.

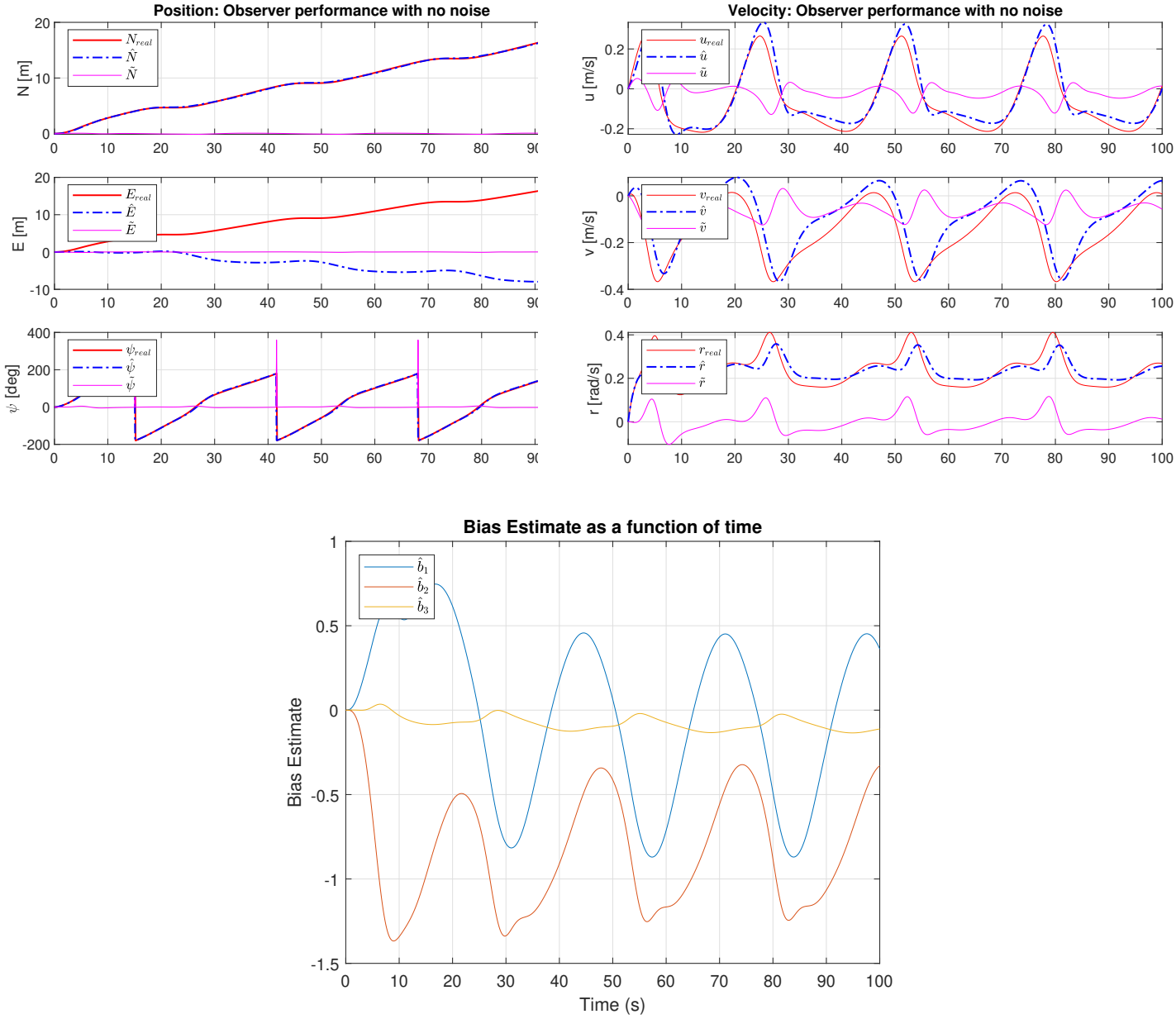



Figure 4: Simulation 3 - Observer performance with bias and no noise. Top left: Position. Top right: Velocities. Bottom: Bias estimation

As can be seen, the performance of the the observer is well within satisfactory bounds for all

states included in the observer. The error between true values and estimated values are $\approx 0$, and the velocity error stays low. The bias drops to zero whenever the correction term is close to zero.

### 6.1.2 Observer performance with measurement noise

The measurement noise was implemented as a band limited white noise block in simulink which we scaled with a constant noise gain, $\mu$, and added to the measurement signal before the observer. We tested with the values $\mu \in [0.1, 1, 10]$ and plotted the performance of the observer compared with the measurement. The following figures illustrates the performance with $\mu = [0.1, 1, 10]$ and as can be seen the observer performance is satisfying. The estimate follows the measurement and true values to a great extent, although there are some issues when the noise gain is set too high.

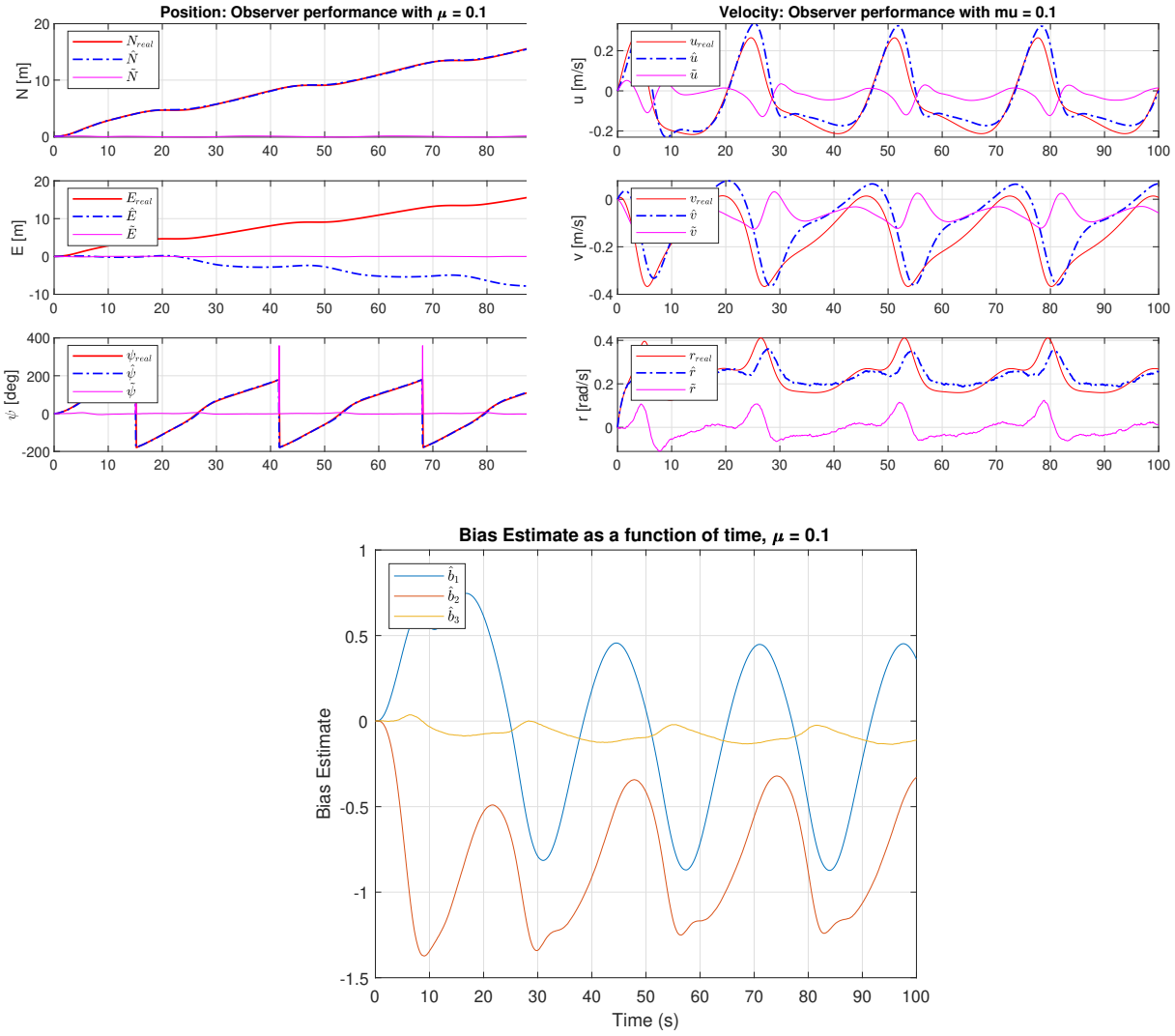



Figure 5: Simulation 4 - Observer performance with measurement noise $\mu = 0.1$. Top left: Position. Top right: Velocities. Bottom: Bias estimation

### 6.1.3 Dead Reckoning and loss of signal simulation

This module was remade completely during the last week, as the results from the progress report could not be recreated. The result is the implementations found in appendix D. Figure 23 illustrates how we solved simulating a loss of signal, with the user defined code found in appendix E, and figure 22 in appendix D with the user defined code found in E displays how we implemented the nonlinear control. The final iteration in both the runnable files and the master file uses this design, and not a simulink diagram as progress report B originally illustrated. The result is something that is quite robust, especially for short periods of time.

The vessel enters dead reckoning when first we have a simulated loss of signal, which then a

deadreckoning function detects and passes through a logic bool to the nonlinear observer. If deadreckoning is true, we multiply the observer gains with 0, and cancel out the correction term from the observer, leaving us with a predictor which estimates based on a constant bias and $\tau$. Considering the availability of sensors for measuring position and the nonlinear observer, this seemed to be the best solution at hand. Another solution would've been to integrate up accelerometers to estimate the position, which we did to great success in the simulation model, but as we didn't have this available for the model scale test, this method was discarded.

The result is something that is very robust for a certain amount of time, but the longer the vessel stays in deadreckoning mode, the longer the vessel position estimate drift away from the true position. This is expected due to the lack of a correction term. The drift illustrated figure 8 and 7 shows this. The drift would've been smaller if we simulated loss of signal further out in the simulation, when the vessel hits a sort of equillibrium. A figure displaying loss of signal for one second can be found in figure 17 in appendix C.

The drift in both position and velocity estimates in theese figures are also expected due to the same reasons as mentioned above.

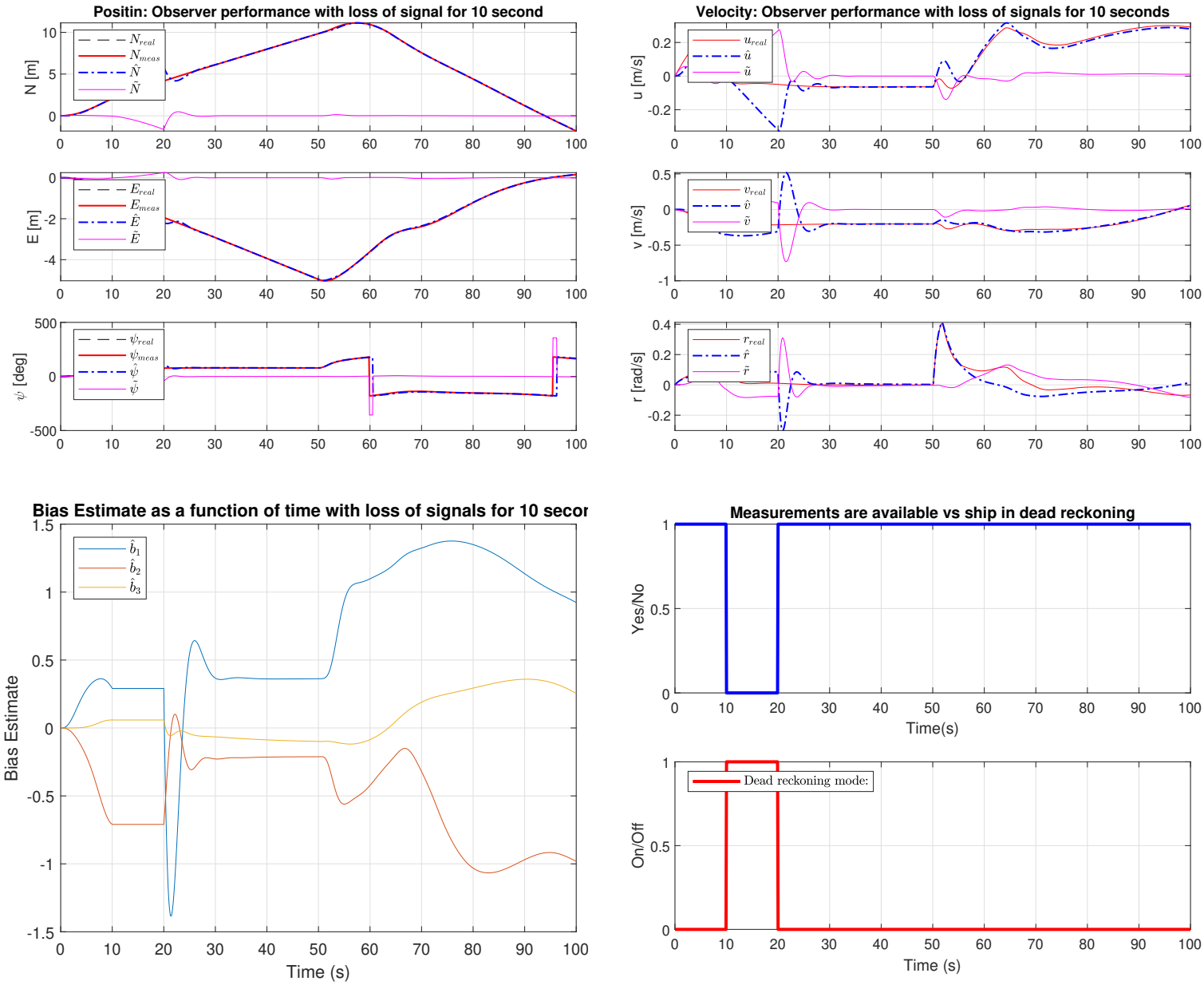



Figure 6: Simulation 6 - Observer performance with loss of signal for 10 second. Top left: Position. Top right: Velocities. Bottom left: Bias estimation. Bottom right, Dead reckoningmode enabled vs loss of signal detected

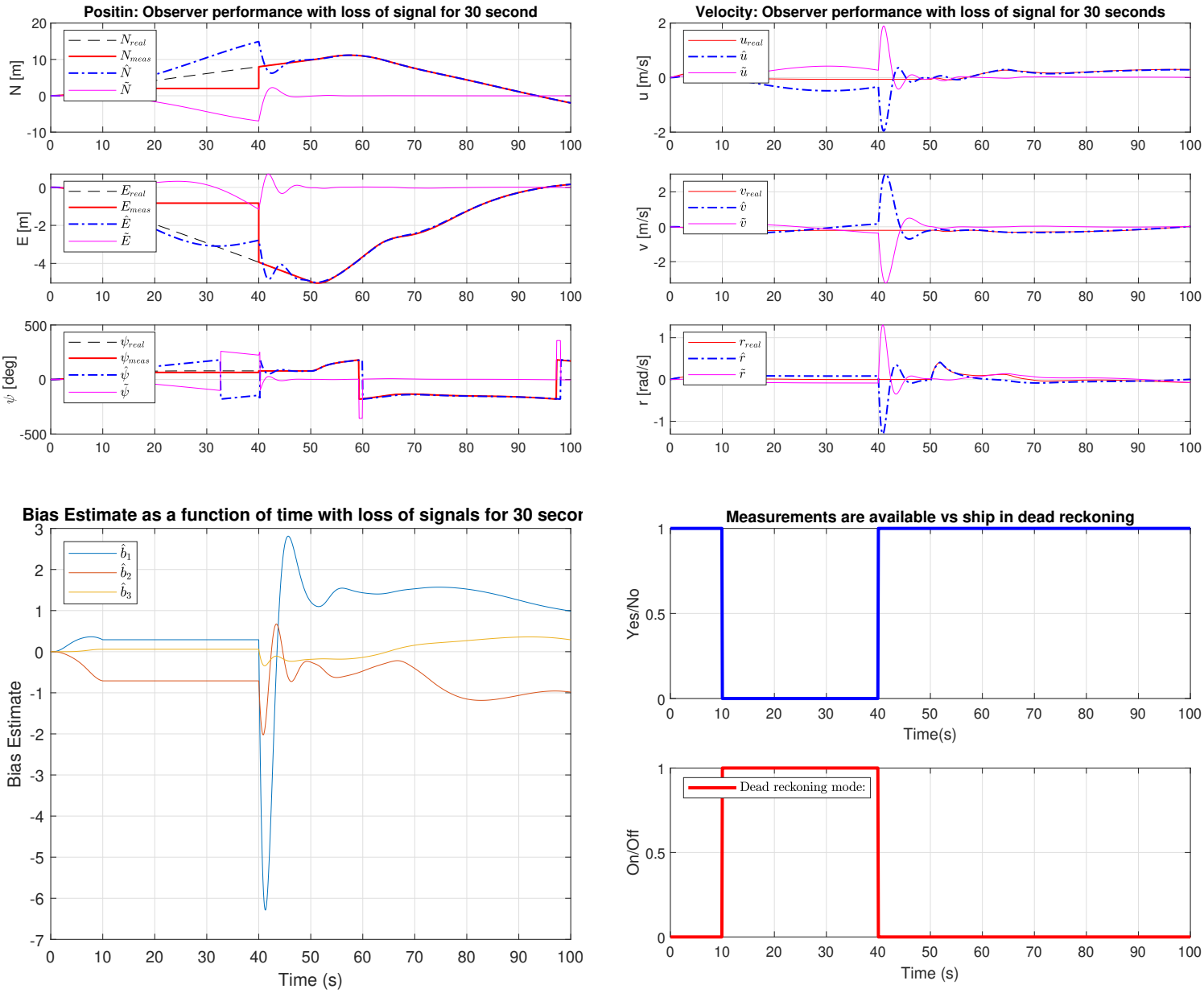
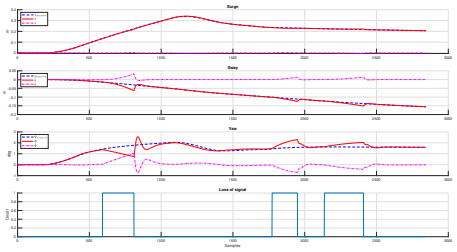



Figure 7: Simulation 6 - Observer performance with loss of signal for 30 second. Top left: Position. Top right: Velocities. Bottom left: Bias estimation. Bottom right, Dead reckoningmode enabled vs loss of signal detected

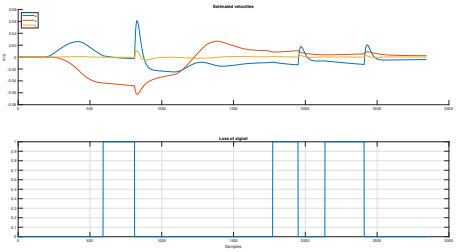
# 7 HIL test

In order to test the loss of signal function of our dead reckoning we implemented a switch on the sixaxis controller($L_2$) that changes from measured position to $\eta = [0 \quad 0 \quad 0]^T$ to simulate a

loss of signal. The results are shown in Figure 8a. The velocity from the observer is also logged in the HIL-setup. The results are presented in figure 8b



(a) Measured position,estimated position and error



(b) Estimated velocity from the observer

Figure 8: HIL - dead reckoning

The position estimate in surge is very close to the actual position. However the position in sway and yaw drifts when we use use the estimated position as feedback to the observer. Velocity estimation is also somewhat inaccurate in deadreckoning mode, as one can see a sudden increase in surge velocity and a decrease in sway velocity after the signal returns.

# Part IV
# Case Study C

This part of the report contains the solving of the maneuvering problem. The maneuvering problem is divided into a geometric task and a dynamic task.
**Geometric task:**
For a absolutely continous function s(t), force the output $\eta$ to converge to $\eta_d(s(t))$. Explained mathematically that is:

$$\lim_{x\to\infty} |\eta(t) - \eta_d(s(t))| = 0 \tag{40}$$

**Dynamic task:**
Control $\dot{s}$ to converge to desired speed assignment $U_s(s,t)$ that is,

$$\lim_{x\to\infty} |\dot{s}(t) - U_s(s(t),t)| = 0 \tag{41}$$

## 8 Theory

### 8.1 Path parametrization

To ensure that the vessel are capable of following both an elliptical and a straight line path where the heading follows the tangent with a constant speed $U_{ref}$, we introduce the path parameter s. From here the desired heading, $\eta_d$ and $U_s$ are calculated and differentiated. At last defining the parametrizations $p_d$ for both elliptical path and straight line path.

### 8.1.1 Expressing the derivatives of $\psi_d$ through $p_d$

$$\psi_d = f(x'_d, y'_d) = \arctan\frac{dy}{dx} = \arctan\frac{y'_d}{x'_d}$$
$$\psi'_d = \frac{\partial f}{\partial x'_d}x''_d + \frac{\partial f}{\partial y'_d}y''_d$$
$$\psi''_d = \frac{\partial^2 f}{\partial x'^2_d}(x''_d)^2 + \frac{\partial^2 f}{\partial y'^2_d}(y''_d)^2 + 2\frac{\partial^2 f}{\partial x'_d y'_d}x''_d y''_d + \frac{\partial f}{\partial x'_d}x'''_d + \frac{\partial f}{\partial y'_d}y'''_d$$

where the partial derivatives are expressed as

$$\frac{\partial f}{\partial x'_d} = \frac{-y'_d}{{y'_d}^2 + {x'_d}^2}, \quad \frac{\partial f}{\partial y'_d} = \frac{x'_d}{y'^2_d + {x'_d}^2}$$
$$\frac{\partial^2 f}{\partial x'^2_d} = \frac{2x'_d y'_d}{({y'_d}^2 + {x'_d}^2)^2}, \quad \frac{\partial^2 f}{\partial y'^2_d} = \frac{-2x'_d y'_d}{({y'_d}^2 + {x'_d}^2)^2}, \quad \frac{\partial^2 f}{\partial x'_d y'_d} = \frac{{y'_d}^2 - {x'_d}^2}{({y'_d}^2 + {x'_d}^2)^2}$$

### 8.1.2 Speed assignment

$$\dot{s} \longrightarrow U_s = h(x'_d, y'_d) = \frac{U_{ref}}{|p'_d|} = \frac{U_{ref}}{\sqrt{{x'_d}^2 + {y'_d}^2}} \tag{42}$$
$$U'_s = \frac{\partial h}{\partial x'_d}x''_d + \frac{\partial h}{\partial y'_d}y''_d \tag{43}$$

where

$$\frac{\partial h}{\partial x'_d} = -\frac{U_{ref}x'_d}{({x'_d}^2 + {y'_d}^2)^{3/2}}, \quad \frac{\partial h}{\partial y'_d} = -\frac{U_{ref}y'_d}{({x'_d}^2 + {y'_d}^2)^{3/2}}$$

### 8.1.3 Straight line parametrization

Straight line parametrization is derived from vector geometry from the figure. The parametrization with is derivatives becomes.

$$p_d(s) = (p_2 - p_1)s + p_1 = sp_2 + (1-s)p_1$$
$$p'_d = p_2 - p_1,\ p''_d = 0,\ p'''_d = 0$$

### 8.1.4 Ellipsoidal parametrization

Define first two coordinate systems. One in the center of the ellipsoid $(\bar{x}, \bar{y})$ and one in an arbitrary point $(x, y)$.

$$x = C_x + \bar{x}, y = C_y + \bar{y}$$

The coordinates are then parametrized

$$\bar{x} = r_x\cos(2\pi s), \bar{y} = r_y\sin(2\pi s)$$
$$x = C_x + r_x\cos(2\pi s), y = C_y + r_y\sin(2\pi s)$$
$$p_d(s) = [x_d(s), y_d(s)]^T = \begin{bmatrix} C_x + r_x\cos(2\pi s) \\ C_y + r_y\sin(2\pi s) \end{bmatrix} = p_0 + \mathbf{R}_{xy}\zeta(s)$$

Where

$$p_0 = [C_x \ C_y]^T, \zeta(s) = [\cos(2\pi s) \ \sin(2\pi s)]^T, \mathbf{R}_{xy} = \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix}$$

The parametrization satisfies the ellipsoidal equation.

$$\frac{\bar{x}^2}{r_x^2} + \frac{\bar{y}^2}{r_y^2} = 1 \tag{44}$$

The derivatives becomes:

$$p_d' = \begin{bmatrix} -r_x 2\pi \sin(2\pi s) \\ r_y 2\pi \cos(2\pi s) \end{bmatrix}, p_d'' = \begin{bmatrix} -r_x (2\pi)^2 \cos(2\pi s) \\ -r_y (2\pi)^2 \sin(2\pi s) \end{bmatrix}, p_d''' = \begin{bmatrix} r_x (2\pi)^3 \sin(2\pi s) \\ -r_y (2\pi)^3 \cos(2\pi s) \end{bmatrix}$$

## 8.2 Kinematic control design

From the kinematic model $\dot{\eta} = \mathbf{R}(\psi)\nu$, and $z_1 := \mathbf{R}(\psi)^T(\eta - \eta_d(s))$ we're going to show that our assumed control law with the given control Lyapunov function yields a satisfying result and to derive the expressions $V_1'$ for both straight-line path and ellipsoidal path.

### 8.2.1 Differentiation of $V_1$

$$\begin{aligned} \dot{V}_1(\eta, s, \dot{s}) = z_1^T \dot{z}_1 &= z_1^T [\dot{\mathbf{R}}^T(\psi)(\eta - \eta_d) + \mathbf{R}^T(\psi)(\dot{\eta} - \dot{\eta_d})] \\ &= z_1^T [\mathbf{S}^T(\dot{\psi})\mathbf{R}^T(\psi)(\eta - \eta_d) + \mathbf{R}^T(\psi)(\dot{\eta} - \dot{\eta_d})] \\ &= z_1^T [\mathbf{S}^T(\dot{\psi}) z_1 + \mathbf{R}^T(\psi)(\dot{\eta} - \dot{\eta_d})] \end{aligned} \tag{45}$$

Given $S = -S^T$, we can prove that $z_1^T S z_1 = 0$:

$$\begin{aligned} z_1^T \mathbf{S} z_1 &= \frac{1}{2} z_1^T \mathbf{S} z_1 + \frac{1}{2} z_1^T \mathbf{S} z_1 \\ &= \frac{1}{2} z_1^T \mathbf{S} z_1 + \frac{1}{2} (z_1^T \mathbf{S}^T z_1)^T = \frac{1}{2} z_1^T \mathbf{S} z_1 - \frac{1}{2} z_1^T \mathbf{S} z_1 = 0 \end{aligned} \tag{46}$$

With the result above, we have

$$\dot{V}_1(\eta, s, \dot{s}) = z_1^T [\mathbf{R}^T(\psi)\dot{\eta} - \mathbf{R}^T(\psi)\dot{\eta_d}] = z_1^T [\nu - \mathbf{R}^T(\psi)\eta_d' \dot{s}] \tag{47}$$

### 8.2.2 CLF analysis with control law

Substituting $\nu = \alpha_1$ in Eq. 47,where $\alpha_1$ is

$$\alpha_1(\eta, s) = -\mathbf{K}_p z_1 + \mathbf{R}(\psi)^T \eta_d'(s) U_s(s) \tag{48}$$

we have that

$$\dot{V}_1(\eta, s, \dot{s}) = -z_1^T \mathbf{K}_p z_1 + z_1^T \mathbf{R}(\psi)^T \eta_d' (U_s - \dot{s}) \tag{49}$$

Since $K_p$ is positive definite, the quadratic form $z_1^T K_p z_1$ is greater than or equal to $\lambda_{min}(\mathbf{K}_p)|z_1|^2$ , thus $-z_1^T \mathbf{K}_p z_1 \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2$ is valid (nonlinear control p.362 [7]). Furthermore, the partial derivative of $V_1(\eta, s)$ with respect to s is

$$V_1'(\eta, s) = z_1^T z_1' = -z_1^T \mathbf{R}(\psi)^T \eta_d' \tag{50}$$

Inserting Eq.50 into Eq. 49 yields

$$\dot{V}_1(\eta, s, \dot{s}) = -z_1^T \mathbf{K}_p z_1 - z_1^T \mathbf{R}(\psi)^T \eta_d' \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - V_1'(\lambda, s)(U_s - \dot{s}) \tag{51}$$

### 8.2.3 Deriving the expression for $V_1'$ for straight-line path

From section 8.2.2 we have derived the expression for $V_1'(\eta, s)$:

$$V_1'(\eta, s) = -(\eta - \eta_d(s))^T \eta_d'(s) \tag{52}$$

With the result in section 8.1.3, for a straight-line path we have

$$\eta = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \eta_d(s) = \begin{bmatrix} sx_2 + (1-s)x_1 \\ sy_2 + (1-s)y_1 \\ \arctan(\frac{y_2 - y_1}{x_2 - x_1}) \end{bmatrix}, \eta_d'(s) = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ 0 \end{bmatrix}, \eta_d'(s) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{53}$$

Here $\eta_d''$ is also included because it will be used later in the DP maneuvring control design.

### 8.2.4 Deriving the expression for $V_1'$ for Ellipsoidal path

Equation 52 is used for the ellipsoidal path Lyapunov function as well, were we insert the following variables:

$$\eta_d(s) = \begin{bmatrix} x_d \\ y_d \\ \psi_d \end{bmatrix} = \begin{bmatrix} C_x + r_x \cos(2\pi s) \\ C_y + r_y \sin(2\pi s) \\ \arctan(\frac{y_d'}{x_d'}) \end{bmatrix}, \eta_d'(s) = \begin{bmatrix} x_d' \\ y_d' \\ \psi_d' \end{bmatrix} = \begin{bmatrix} -r_x 2\pi \sin(2\pi s) \\ r_y 2\pi \cos(2\pi s) \\ \frac{\partial f}{\partial x_d'} x_d'' + \frac{\partial f}{\partial y_d'} y_d'' \end{bmatrix} \tag{54}$$

$\eta_d''$ is included as the same reason as for the straight line path.

$$\eta_d''(s) = \begin{bmatrix} x_d'' \\ y_d'' \\ \psi_d'' \end{bmatrix} = \begin{bmatrix} -r_x(2\pi)^2 \cos(2\pi s) \\ -r_y(2\pi)^2 \sin(2\pi s) \\ \frac{\partial^2 f}{\partial x_d'^2}(x_d'')^2 + \frac{\partial^2 f}{\partial y_d'^2}(y_d'')^2 + 2\frac{\partial^2 f}{\partial x_d' y_d'} x_d'' y_d'' + \frac{\partial f}{\partial x_d'} x_d''' + \frac{\partial f}{\partial y_d'} y_d''' \end{bmatrix} \tag{55}$$

## 8.3 Update laws

The maneuvering problem requires an update law to say something about how $\dot{s}$ should be updated. There are several different methods to handle this, and the following sections handles a few methods.

### 8.3.1 Tracking update law

The tracking update law is:

$$\dot{s} = U_s(s) \tag{56}$$

The Lyapunov function time derivative bound in section 8.2.2 becomes:

$$\dot{V}_1 \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - V_1'(\lambda, s)(U_s - \dot{s}) = -\lambda_{min}(\mathbf{K}_p)|z_1|^2 \leq 0 \tag{57}$$

This gives that $z_1 = 0$ to be UGES.
This is called tracking update law because the following equations,

$$\dot{\xi} = U_s(t, \xi), s = \xi \tag{58}$$

where $\bar{p}_d(t) := p_d(s(t))$ forms a tracking problem of p(t) $\rightarrow \bar{p}_d(t)$, or $e(t) = p(t) - \bar{p}_d(t) \rightarrow 0$.

### 8.3.2 Gradient update law

The gradient update law is with $\mu \geq 0$

$$\dot{s} = U_s(s) - \mu V_1^s \tag{59}$$

This will make the time derivative of the Lyapunov function become.

$$\dot{V}_1 \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - \mu(V_1^s)^2 \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 \leq 0 \tag{60}$$

This make $z_1 = 0$ gives UGES.
This is called a gradient update law since $V_1^s$ is the gradient for the Lyapunov function with respect to s. Furthermore $\dot{s}$ takes feedback from $V_1^s$, which ensures that s is drawn to a minimizer of the Lyapunov function.

### 8.3.3 Modified gradient update law

The unit-tangent gradient update law is.

$$\dot{s} = U_s(s) - \frac{\mu}{|\eta_d'|} V_1^s \tag{61}$$

Here is still $\mu \geq 0$

$$\begin{aligned} \dot{V}_1 &\leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - V_1^s(U_s(s) - \dot{s}) \\ &= -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - \frac{\mu}{|\eta_d'|}(V_1^s)^2 \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 \end{aligned} \tag{62}$$

This still makes $z_1 = 0$ UGES.

The modified gradient update law, or unit tangent gradient update law normalizes $V_1'$ to use the unit tangent vector along the path. This is displayed in the following equation, where $\bar{\eta_d}$ is the unit tangent vector.

$$\dot{s} = U_s(s) - \mu \frac{V_1'}{|\eta_d'|} = U_s(s) - \mu \frac{\eta_d'^T}{|\eta_d'|}(\eta - \eta_d) = U_s(s) - \mu \bar{\eta_d}^T e \tag{63}$$

### 8.3.4 Filtered gradient update law

The filter gradient update law is.

$$\dot{s} = U_s(s) - \omega_s, \ \dot{\omega_s} = -\lambda(\omega_s - \mu V_1'(\eta, s)) \tag{64}$$

The new Lyapunov function becomes

$$W_1(\eta, s, \omega_s) = V_1(\eta, s) + \frac{1}{2\lambda\mu}\omega_s^2 \tag{65}$$

The derivative of the Lyapunov function the becomes.

$$\begin{aligned}\dot{W}_1(\eta, s, \omega_s) &= \dot{V}_1 + \frac{1}{\lambda\mu}\omega_s\dot{\omega}_s \leq -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - V_1'(V_1 - (U_s - \omega_s)) + \frac{1}{\lambda\mu}\omega_s(-\lambda(\omega_s - \mu V_1')) \\ &= -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - V_1'\omega_s - \frac{1}{\mu}\omega_s^2 \ = V_1'\omega_s = -\lambda_{min}(\mathbf{K}_p)|z_1|^2 - \frac{1}{\mu}\omega_s^2\end{aligned} \tag{66}$$

Here both the results obtained in the previous section and the update law is used. This makes $z_1 = 0$ UGES.

$$\dot{\omega_s} = -\lambda(\omega_s - \mu V_1^s)\omega_s = -\lambda\omega_s + \lambda\mu V_1^s \overset{\mathcal{L}}{\Rightarrow} \omega_s = \frac{\lambda}{s+\lambda}(\mu V_1^s) \tag{67}$$

The last step is from Laplace transformation of the equation. This is called a filtered gradient update law because the gradient term $V^s(\eta, s)$ is input to a first order lowpass filter in the form of $\omega_s = \frac{\lambda}{s+\lambda}$ before entering the path speed dynamics $\dot{s}$.

## 8.4 DP maneuvering control design

The DP maneuvering control design is made with the modified gradient update law, where a two step backstepping process designs the control law. Prior derivations of the LCFs used, their derivatives and their criteria to ensure stability are taken in consideration when designing the virtual control in the first step and the control law $\tau$ in the second step. The first step introduces a new error term $z_2 = v - \alpha_1(\eta, s)$ and modify $\dot{V}_1$ to include $z_2$. Then derivatives for using in the second Lyapunov function are calculated. Then a new Lyapunov function for step 2 $V_2$ is defined and $\dot{V}_2$ is calculated. At last the control law $\tau$ is defined so $\dot{V}_2$ is negative definite. To ensure proper bias compensation in the final control law, an integral term was added to the controller. This is handled in section 8.4.3.

### 8.4.1 Calculating new $\dot{V}_1$ with based on the virtual control

Inserting $\nu = z_2 + \alpha_1(\eta, s)$ and the modified gradient update law into the result in section 8.3.3, we have:

$$\begin{aligned}\dot{V}_1(\eta, s, \dot{s}) &= z_1^T[\nu - \mathbf{R}^T(\psi)\eta_d'\dot{s}] = z_1^T\left(z_2 + \alpha_1(\eta, s) - \mathbf{R}(\psi)^T\eta_d'(s)\dot{s}\right) \\ &= z_1^T\left(z_2 - \mathbf{K}_p z_1 + \mu\mathbf{R}(\psi)^T\frac{\eta_d'(s)}{|\eta_d'(s)|}V_1'(\eta, s)\right)\end{aligned} \tag{68}$$

$$\dot{\alpha}_1(\eta, s) = -\mathbf{K}_p\dot{z}_1 + \frac{d}{dt}\left[\mathbf{R}(\psi)^T\eta_d'(s)U_s(s)\right] \tag{69}$$

The two terms should be calculated separately,

$$\begin{aligned}\dot{z}_1 &= \dot{R}(\psi)^T(\eta - \eta_d) + \mathbf{R}(\psi)^T(\dot{\eta} - \dot{\eta_d}) \\ &= \mathbf{S}(\dot{\psi})^T\mathbf{S}(\psi)^T(\eta - \eta_d) + \nu - \mathbf{R}(\psi)^T\eta_d'(s)\dot{s} \\ &= -\mathbf{S}(\dot{\psi})z_1 + \nu - \mathbf{R}(\psi)^T\eta_d'(s)\left(U_s(s) - \frac{\mu}{|\eta_d'(s)|}V_1'(\eta, s)\right)\end{aligned} \tag{70}$$

Here is the calculation for the second term.

$$\begin{aligned}\frac{d}{dt}\left[\mathbf{R}(\psi)^T\eta_d'(s)U_s(s)\right] &= \dot{\mathbf{R}}(\psi)^T\eta_d'(s)U_s(s) + \mathbf{R}(\psi)^T\dot{\eta}_d'(s)U_s(s) + \mathbf{R}(\psi)^T\eta_d'(s)\dot{U}_s(s) \\ &= \mathbf{S}(\dot{\psi})^T\mathbf{R}(\psi)^T\eta_d'(s)U_s(s) + \mathbf{R}(\psi)^T\dot{s}\left[\eta_d''(s)U_s(s) + \eta_d'(s)U_s'(s)\right]\end{aligned} \tag{71}$$

where $\eta_d'(s), \eta_d''(s)$ are dependent on the path parametrization and $U_s'(s)$ is already expressed in section 8.2.3 and 8.2.4
Based on the vessel dynamics, we have $\dot{\nu} = \mathbf{M}^{-1}(-\mathbf{D}v + \tau)$

$$\dot{z}_2 = \dot{\nu} - \dot{\alpha}_1(\eta, s) = \mathbf{M}^{-1}(-\mathbf{D}\nu + \tau) - \dot{\alpha}_1(\eta, s) \tag{72}$$

### 8.4.2 Computing $\dot{V}_2$ and control law $\tau$

Given $V_2(\nu, \eta, s) = V_1(\eta, s) + \frac{1}{2}z_2^T\mathbf{M}z_2$, we can find $\dot{V}_2$ as follows:

$$\dot{V}_2(\nu, \eta, s) = \dot{V}_1(\eta, s, \dot{s}) + z_2^T\mathbf{M}\dot{z}_2 \ = z_1^T(\nu - \mathbf{R}(\psi)^T\eta_d'(s)\dot{s}) + z_2^T(-\mathbf{D}\nu + \tau - \mathbf{M}\dot{\alpha}_1) \tag{73}$$

We've already proved $\dot{V}_1(\nu, \eta, s, \dot{s})$ is negative definite in 8.3.3. To ensure $\dot{V}_2(\nu, \eta, s)$ is negative definite, the second term in 45 should stay negative definite. Firstly, find the expression for $\nu$ with $z_2$ and $\alpha_1(\eta, s)$, that is $\nu = z_2 + \alpha_1(\eta, s)$. Then, insert $\nu$ into the expression for $\dot{V}_2(\nu, \eta, s)$, we have

$$\begin{aligned}\dot{V}_2(\nu, \eta, s) &= \dot{V}_1(\nu, \eta, s) + z_2^T\left(\tau - \mathbf{D}(z_2 + \alpha_1(\eta, s)) - \mathbf{M}\dot{\alpha}_1(\eta, s)\right) \\ &= \dot{V}_1(\nu, \eta, s) + z_2^T(\tau - \mathbf{D}\alpha_1(\eta, s) - \mathbf{M}\dot{\alpha}_1(\eta, s)) - z_2^T\mathbf{D}z_2\end{aligned} \tag{74}$$

Given $\mathbf{D} > 0$, $-z_2^T\mathbf{D}z_2 < 0$ is valid. In order to propose a control law, it should be ensured that the second and third term in (74) is negative definite. Let $\tau - \mathbf{D}\alpha_1(\eta, s) - \mathbf{M}\dot{\alpha}_1(\eta, s) = -\mathbf{K}_v z_2$, where $\mathbf{K}_v > 0$. Thus the control law for $\tau$ can be expressed as

$$\tau = \mathbf{D}\alpha_1(\eta, s) + \mathbf{M}\dot{\alpha}_1(\eta, s) - \mathbf{K}_v z_2 \tag{75}$$

This leaves us with

$$\begin{aligned}\dot{V}_2(\nu, \eta, s) &= \dot{V}_1(\nu, \eta, s) + z_2^T(-K_v z_2) - z_2^T\mathbf{D}z_2 \\ &= \dot{V}_1(\nu, \eta, s) - z_2^T\mathbf{K}_v z_2 - z_2^T\mathbf{D}z_2 < 0\end{aligned} \tag{76}$$

Which is negative definite.

### 8.4.3 Integral action in backstepping design

After implementing our backstepping controller and tested for both path parameterizations, a steady state error was discovered. This is caused by the constant bias in our model, acting as a constant environmental load such as constant current. To combat this, we introduced integral action to our controller design. As a result of following method B in *Skjetne & Fossens paper on integral control in backstepping [5]* an augmented state space were introduced, where the integrator $\xi = \int z_2$ were introduced as:

$$\dot{\xi} = z_2$$

Introducing this in our control law, resulted in

$$\tau = \mathbf{D}\alpha_1(\eta, s) + \mathbf{M}\dot{\alpha}_1(\eta, s) - \mathbf{K}_v z_2 - \mathbf{K}_i \xi \tag{77}$$

Leading us to our closed loop system given as

$$\dot{z}_1 = -\mathbf{S}(\dot{\psi})z_1 + \nu - \mathbf{R}(\psi)^T \eta_d'(s) \left( U_s(s) - \frac{\mu}{|\eta_d'(s)|} V_1'(\eta, s) \right), \tag{78}$$

$$\dot{\xi} = z_2, \;\; z_2 = nu - \alpha_1(\eta, s) \tag{79}$$

From here it is quite easy to prove the systems stability. Again, following the method in [5], we're keeping $V_1$ the same as before, and augmenting $V_2$ as follows:

$$V_2(\xi, \nu, \eta, s) = V_1(\eta, s) + \frac{1}{2}\xi^T \mathbf{K}_i \xi + \frac{1}{2} z_2^T \mathbf{M} z_2$$

This gives us $\dot{V}_2$ as:

$$\begin{aligned} \dot{V}_2(\nu, \eta, s) &= \dot{V}_1(\eta, s, \dot{s}) + \xi^T \mathbf{K}_i \dot{\xi} + z_2^T \mathbf{M} \dot{z}_2 \\ &= z_1^T(\nu - \mathbf{R}(\psi)^T \eta_d'(s)\dot{s}) + \xi^T \mathbf{K}_i z_2 + z_2^T(-\mathbf{D}\nu + \tau - \mathbf{M}\dot{\alpha}_1) \\ &= z_1^T(\nu - \mathbf{R}(\psi)^T \eta_d'(s)\dot{s}) + z_2^T(-\mathbf{D}\nu + \tau - \mathbf{M}\dot{\alpha}_1 + \xi^T \mathbf{K}_i) \end{aligned} \tag{80}$$

$z_1^T(\nu - \mathbf{R}(\psi)^T \eta_d'(s)\dot{s})$ is already accounted for in prior sections, and
$z_2^T(-\mathbf{D}\nu + \tau - \mathbf{M}\dot{\alpha}_1 + \xi^T \mathbf{K}_i)$ is handled the same way as in section 8.4.2. From here we see that the control law given in equation 77 renders the second term in $\dot{V}_2$ negative definite, and Matrosovs theorem A.2 renders the equillibrium point $(\xi, z_1, z_2)$ UGAS.

# 9 Simulation

The simulation part of case C is presented in this section with plots and discussion. After a lot of tuning, a set of controller gains for both the elliptical and the straight line path was found.

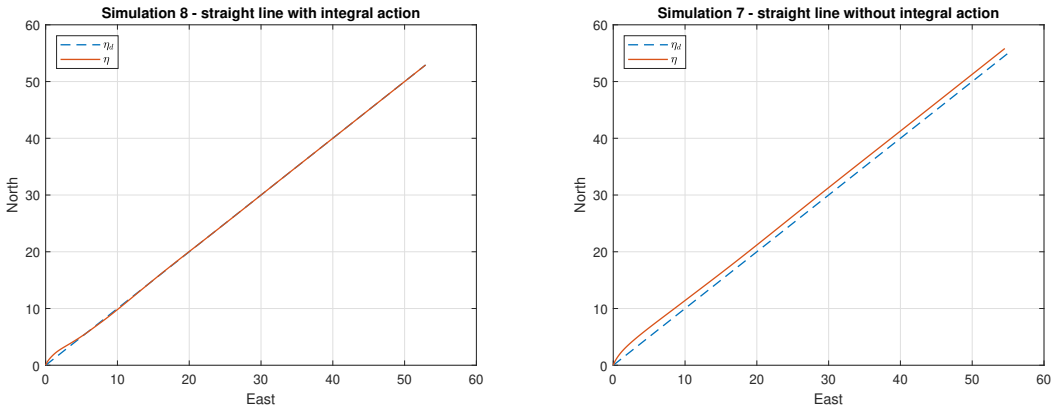
$$\mathbf{K_p} = 0.08 \begin{bmatrix} 1.1 & 0 & 0 \\ 0 & 1.3 & 0 \\ 0 & 0 & 1.4 \end{bmatrix}, \mathbf{K_v} = 1 \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 15 \end{bmatrix}, \mathbf{K_i} = 5 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

where $K_i$ is the extra integral gain matrix.

The realized controller controller can be seen in figure 24 in appendix D, with the corresponding code seen in appendix E.

## 9.1 Straight line simulation

### 9.1.1 With and without integral action




(a) Without integral action

(b) With integral action

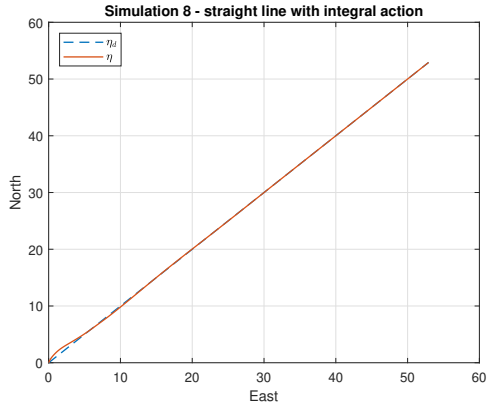
Figure 9: Straight line with and without integral action

As shown in figure 9, that start at [0 0 0] and follows as path with a heading of 45 degrees, a steady state error emerges. To compensate for this, an integral action was added to the controller and the result is shown in figure 9b).

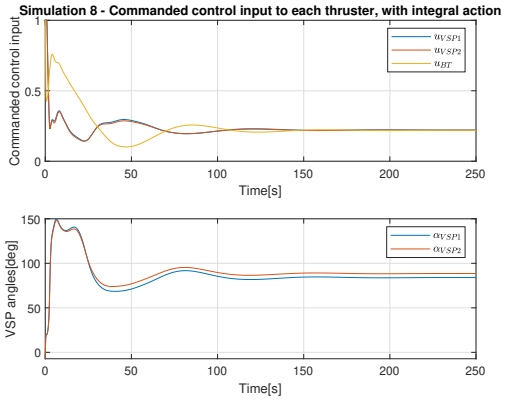
### 9.1.2 With integral action

The main part of this report from here on only consider the results we got with integral action included in our controller. The relevant plots from simulations performed without integral action can be found in appendix C.
. This is due to the reason that the controller with integral action included yielded better performance on all experiments during development of the controller. The same argument goes for elliptical path aswell.

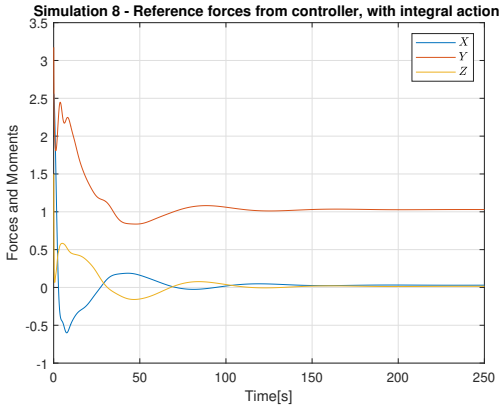
From figure 10a), one can see the vessel overshoots the reference trajectory by a few centimeters. This is probably because the vessel starts with a heading of zero degrees, straight north, and needs to change it in order to follow the desired trajectory. As a result of this in combination with the artificial bias, the commanded thrust experiences a peak and therefore some of the thrusters saturates. When the vessel reaches steady state controller forces, one can observe that the thrust in surge direction is close to zero and thrust in sway is positive. This is due to the artificial bias in NED-frame which is providing a positive force in north and a negative force in east. In practice the vessel uses power to hold its heading and sway position while the thrust of the vessel is produced by the environmental forces or artificial bias.



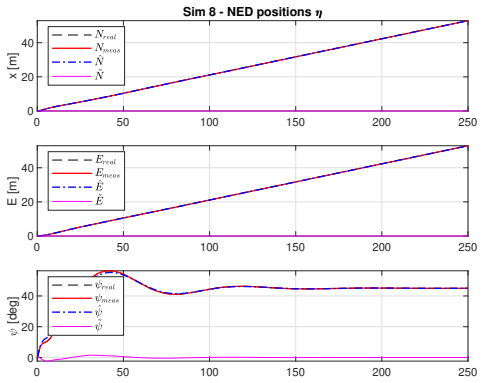

(a) XY-plot of the position in NED-frame



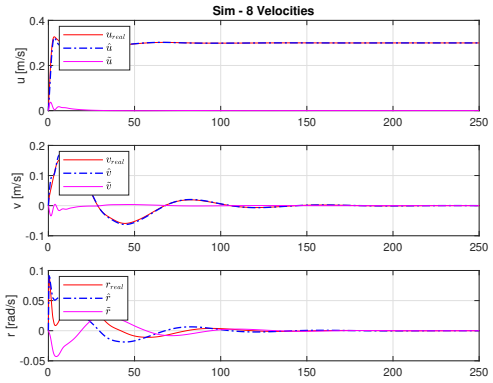

(b) Commanded control input to thrusters




(c) Reference forces in surge sway and yaw




(d) $\eta$ as function of time

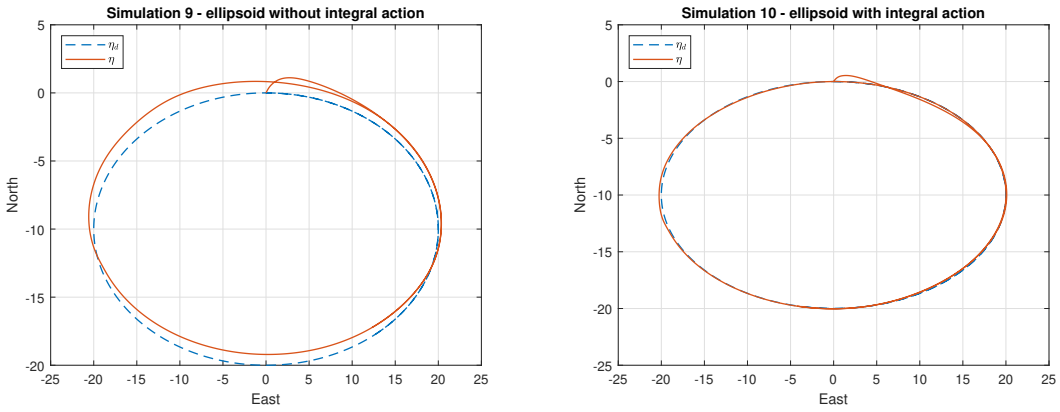



(e) Velocities

Figure 10: Simulation 8 - Straight line with integral action

## 9.2 Elliptical path

### 9.2.1 Comparison of elliptical path with and without integral action



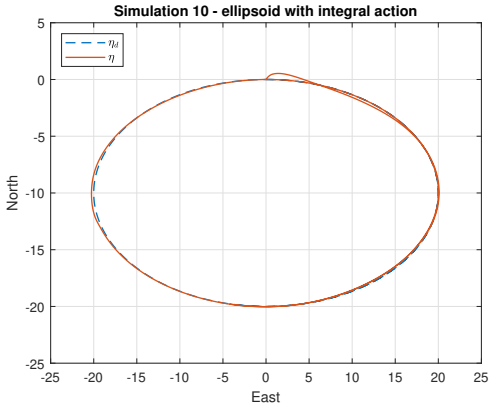

(a) without integral action

(b) With integral action

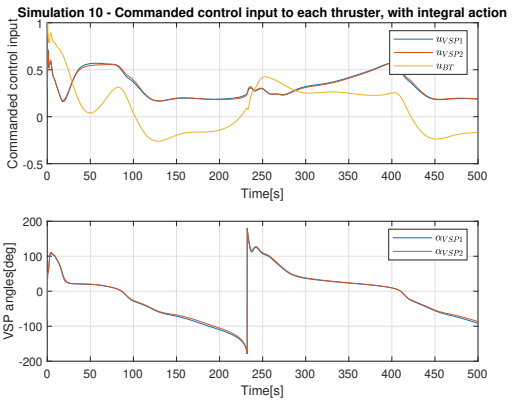
Figure 11: Simulation 9 and 10

As shown in figure 11, with a start postition of [0; 0; 0] and an eliptical path, one can observe the same state error as for the straight line path. Because of the steady state error a controller with integral action is preferable. The relevant figures and plots for this test without integral action can be seen in figure 19 in appendix C. The steady state error makes sense compared with the artificial bias which gives environmental forces from south to north and east to west since the achieved trajectory is shifted north-west.

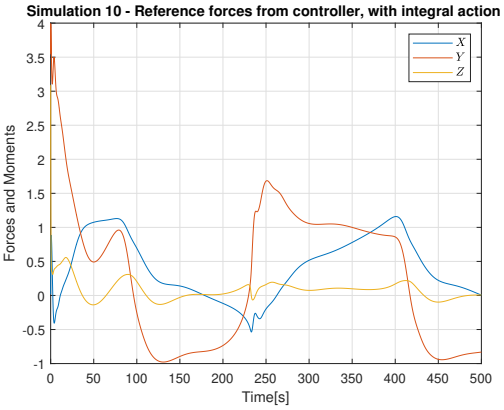
### 9.2.2 Elliptical path with integral action



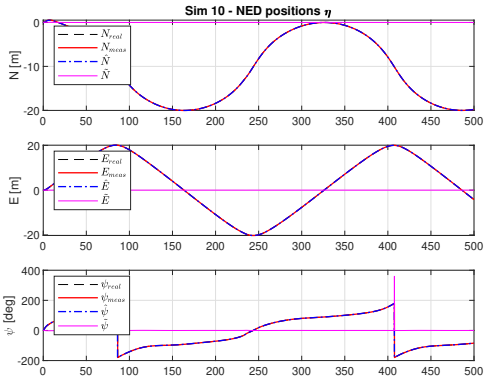

(a) XY-plot of the position in NED-frame



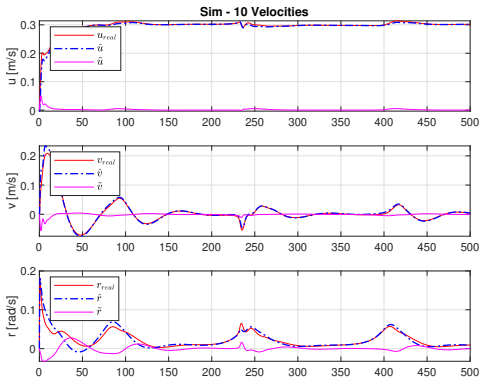

(b) Commanded control input to thrusters




(c) Reference forces in surge sway and yaw




(d) $\eta$ as function of time




(e) Velocities

Figure 12: Simulation 10 - with integral action

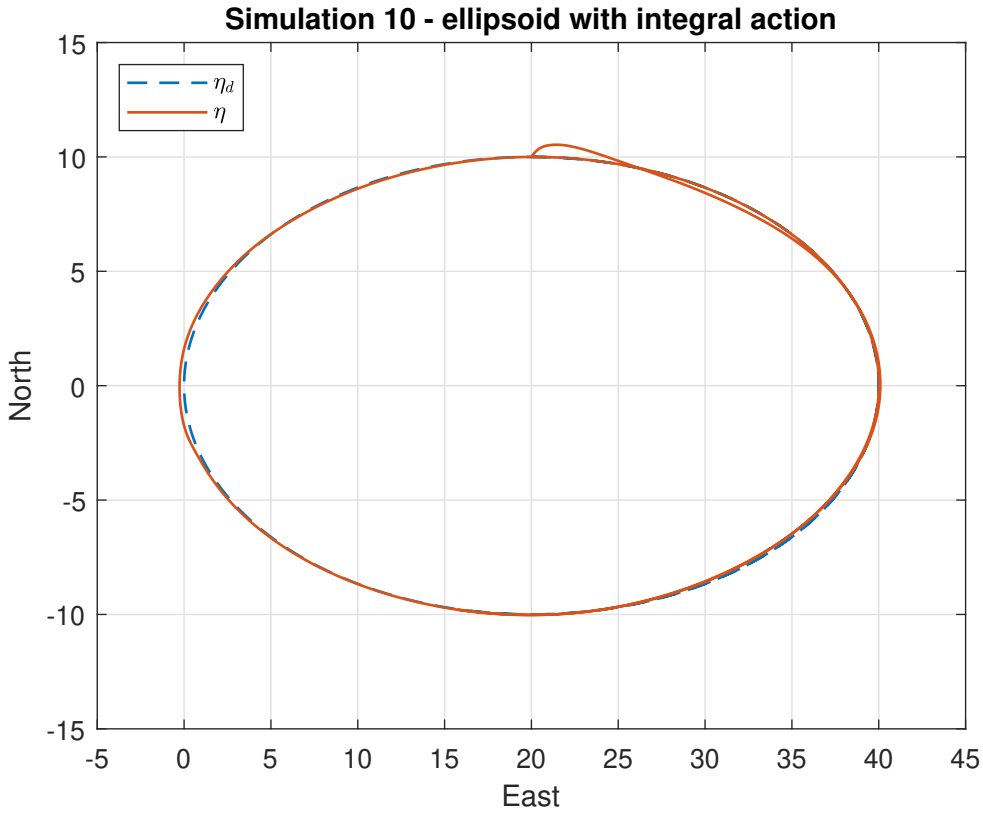



Figure 13: Staring position of the vessel = $[10, 20, 0]^T$

The results from simulation 10, elliptical path with integral action is shown in figure 12. They show very satisfactory results with no thruster saturation except from in the very beginning, probably due to a rapid heading change in combination with the artificial bias. The vessel quickly manages to follow the elliptical path with very high accuracy. Starting position of the vessel is on the line of the ellipsoid because this seems logical, however the heading is not tangentially to the path which explains the sudden peak in thruster forces because the heading is changing rapidly.
By setting the starting position of the vessel to an $[10, 20, 0]^T$, the initialization of the controller is tested. The results of this test is shown in figure 13 and shows that the initial position does not influence the controller performance. The starting position is set in the initialization file for each simulation.

# 10 HIL test

Due to our groups delay with Case A and B, combined with the campus lockdown, the HIL part for Case C was not done. The implementation would have been done the same way as we've done the Simulink implementations, where a up until this point unused button on the controller would've been mapped to change the variable called *pathparameter* from 1 to 0, thus also changing what kind of path the vessel follows. Adding the reminding functionality such as

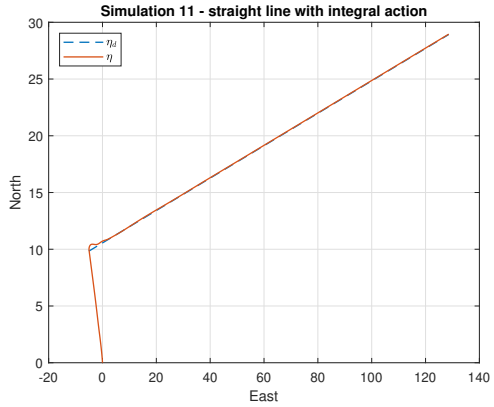
letting the reference speed be changable would've been quite easy to implement if we would've had the box available.

# Part V
# Extra simulations

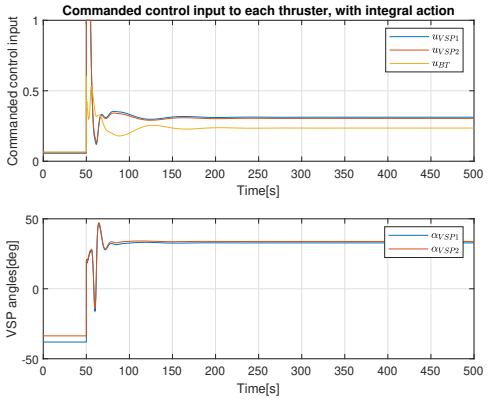
Because of the COVID-19 situation, the lab part of this subject was canceled. Therefore some additional simulation tasks was given to compensate for this.

## 11 Additional simulation 1 - simulation 11

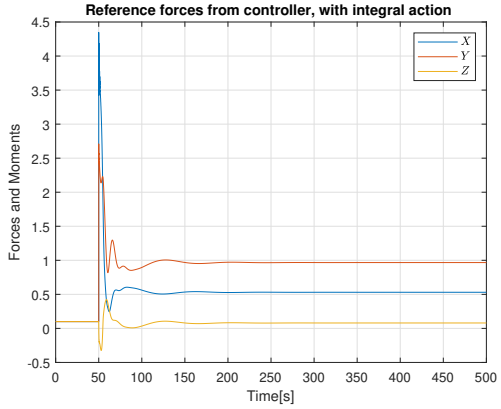
The simulations starts with direct control ($\tau_{ref} = [0.1 \quad 0.1 \quad 0.1]^T$) for 50 seconds and then creates a straight line trajectory from its position in the direction of the heading. To implement this simulation a autopilot MATLAB user defined function was made. This uses the simulation time and the given position of the vessel to create a straight line path. The function block outputs a signal for resetting the integral wind-up and s-variable before each new path is calculated. It also switches from a constant direct thrust to the controller output thrust after 50 seconds.



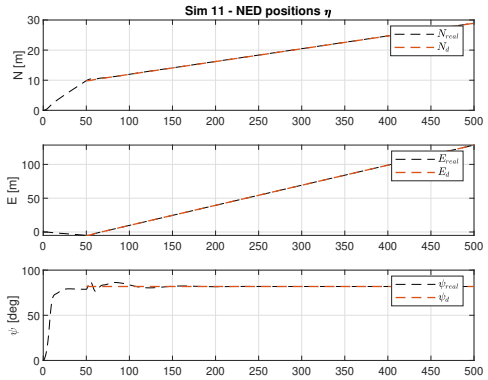

(a) XY-plot of the position in NED-frame




(b) Commanded control input to thrusters




(c) Reference forces in surge sway and yaw




(d) $\eta$ and $\eta_d$ as a function of time

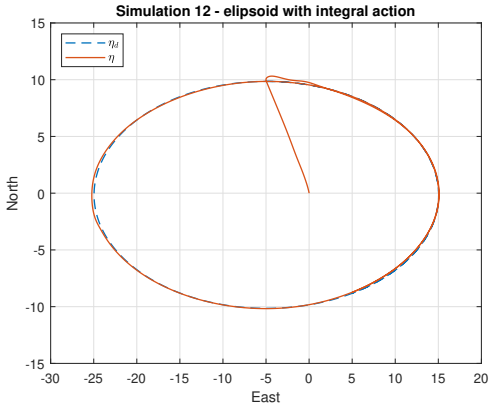
Figure 14: Simulation 11 - straight line after constant tau

From figure 14 c) we observe a spike in the controller reference forces in surge, sway and yaw. This is probably because the boat is moving in north-west direction, but the desired straight line path is in the north-east direction. Because the trajectory is in the heading direction of the vessel, it doesn't need to change change the heading, and therefore the controller yaw moment is significantly lower than the surge and sway force.
In figure 14 b), the commanded control input from the thrust allocation is displayed. In order to keep its position, the vessel saturates the port side Voith Schneider propeller in order to keep the surge and sway position. This could probably be avoided by tuning the controller and the observer a bit less aggressive. It could also be because the thrust allocation algorithm is not optimized.

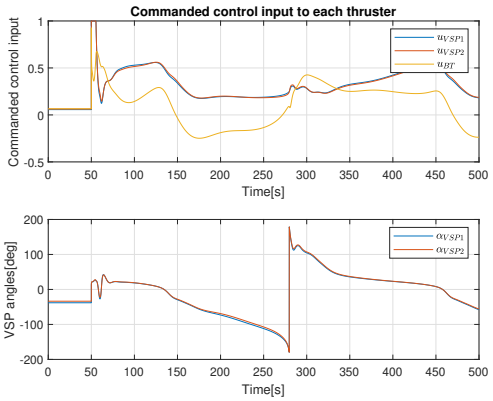
# 12 Additional simulation 2 - simulation 12

This task is almost identical with task 1.1. The only difference is the path parameter value. For an ellipsoid it's equal to 0 and for a straight line it's 1. The size of the ellipsoid is defined in the corresponding initialization file. This could also be done by defining the ellipsoid such that the

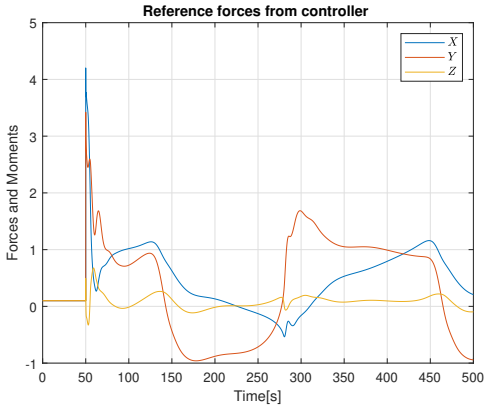
heading of the vessel is tangentially to the curve at the starting point. Then the ellipsoid would be rotated and the vessel would not need to change its heading like it does in 15a).



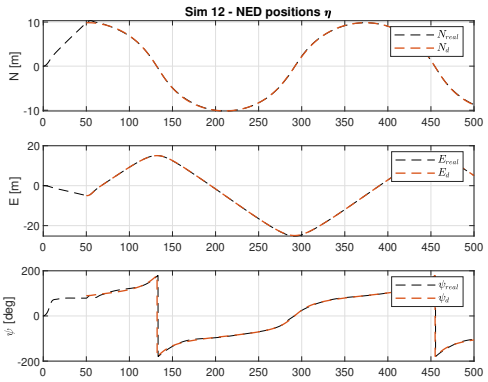

(a) XY-plot of the position in NED-frame




(b) Commanded control input to thrusters




(c) Reference forces in surge sway and yaw




(d) $\eta$ and $\eta_d$ as a function of time

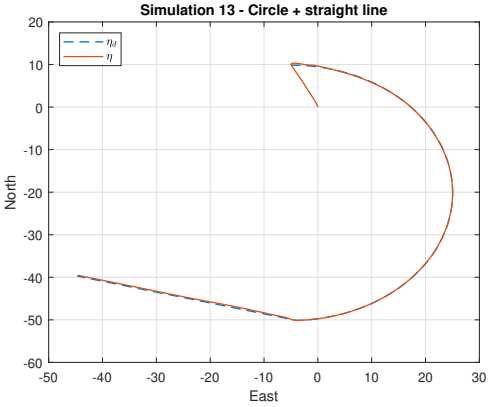
Figure 15: Simulation 12 - ellipsoidal path after constant tau

The vessel starts with a constant $\tau_{ref} = [0.1 \quad 0.1 \quad 0.1]^T$ in 50 seconds followed by a predefined elliptical path. From figure 15a), one can see an overshoot in the beginning after, but the vessel rapidly minimizes the error and follows the trajectory. Because of the rapid change in heading, both Voith Schneider propellers experiences maximum thrust in order to slow the vessel down and reach desired position. This could probably also be avoided by tuning the observer and controller a bit less aggressive. However it manages to reach desired position in both North, East and $\Psi$ as shown in figure 15d).

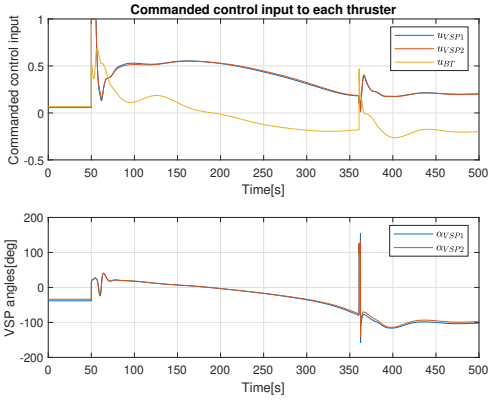
# 13 Additional simulation 3 - simulation 13

This task is a combination of task 1.1 and 1.2 and should include a shifting between circular and straight line path. The criteria for determining whether the vessel has finished the circular path is the distance between time instance t=50s and the current position. If the difference

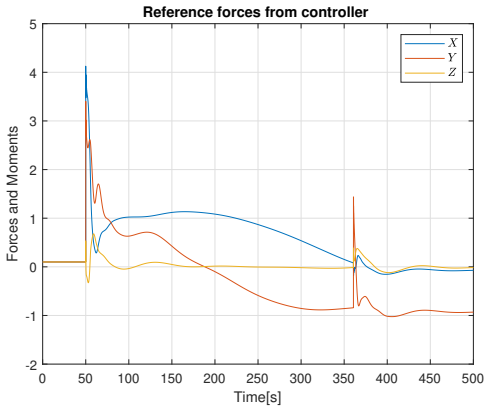
between that and two times of the radius is within an empirical tolerance, it switches to a straight line path in the desired heading tangentially to the circle. This tolerance, however, is dependent on the radius of the circle and there's no better way than trial and error to find the optimal values.
Another way of checking if the ship have completed a half circle is to check the s-value from the controller, this was implemented but for some reason the controller crashed and we did not have time to find out why.



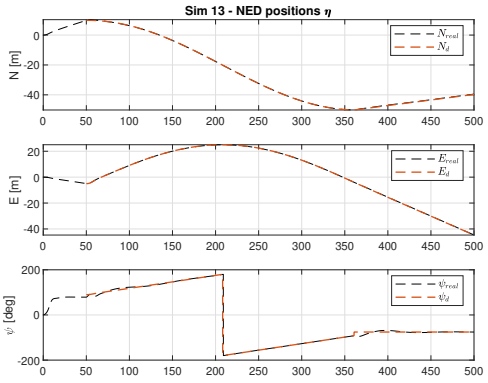

(a) XY-plot of the position in NED-frame




(b) Commanded control input to thrusters




(c) Reference forces in surge sway and yaw




(d) $\eta$ and $\eta_d$ as a function of time

Figure 16: Simulation 13 - half circle and straight line after constant tau

Similar to the results in the previous tasks, the thrusters experience maximum thrust whenever the path is switched. Because the heading change is smaller when changing from the half circle to straight line path, the thrusters do not saturate, but we still have a spike in the thrust and angle change of the thruster which is due to the rapid change of desired position and heading.

# Part VI
# Conclusion and final remarks

## 14 Conclusion and further work

With the exception of the minor HIL implementation on case study C, everything in this project works up to par with what is expected of this design. The final product is a robust system which handles the design criteria left by the case studies and final report. The vessel is able to start at any position, and perform any of the two maneuvers at any point during the simulation. The constant bias is handled with an integral term, which pretty much negates all stationary deviancy. All in all we're satisfied with the design and the performance of the system.

As for further work. If we would have continued working on this project there are a few items that we would have implemented. Among these items are:

- 1) **A nonlinear observer that can handleg complex environmental loads:** The observer designed in this project is designed to handle the constant bias term very well, but it is not designed to handle high frequency wave loads or high frequency measurement noise very well. An option could be to introduce a nonlinear passive observer, an extended kalman filter or both in sequence (exogenous kalman filter).
- 2) **Constrained/optimized extended thrust allocation;** The thrust allocation model in our design is as mentioned an unconstrained thrust allocation model. To ensure a more realistic implementation, this model should have been constrained to some degree to ensure that the thrusters do not rotate infinetly fast, and introduce forbidden sectors where the thrusters can not operate.
- 3) **Further improve on the autopilot:** Implementing a waypoint generator with a lookahead distance.
- 4) **Sideslip compensation:** This was attempted in the experimental phase. The sideslip compensator can be seen in the graveyard of prior implementations in the master file. It is a mechanism that both fully actuated vessels and under actuated vessels may use to combat environmental loads to ensure a better tracking objective result.
- 5) **Soften the spikes when changing mode:** The design illustrates spikes in plots when the vessel changes path following modes. A theory to combat this could be to introduce some kind of filtering or smoothing effect at the first few time instants where we turn over to a different path following mode.

# References


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

[2] Tor A. Johansen  Fossen, T.I: a paper on *Control Allocation - A Survey*

[3] Roger Skjetne (2020): Lecture notes 7 for TMR4243 *Lecture notes on observer design* Department of Marine Technology, NTNU

[4] Svenn Are Værnø  Roger Skjetne (2017): Lecture note: *Observer for simplified DP model: Design and proof* Department of Marine Technology, NTNU

[5] Roger Skjetne  Tor I. Fossen (2004): *On Integral Control in Backstepping: Analysis of Different Techniques* Department of Marine Technology  Department of Engineering Cybernetics, NTNU

[6] Tor A. Johansen and Thor I. Fossen (2012): *Control allocation - A survey* Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway.

[7] Hassan K. Khalil (2014): *Nonlinear Control* Pearson

# A Theorems

## A.1 LaSalle-Yoshizawa's Theorem

Let $\mathbf{x_e} = 0$ be the equilibrium point of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{t})$, and assume that $\mathbf{f}(\mathbf{x}, t)$ is locally Lipschitz in x. Let V: $\in \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}_+$ be a continously differentiable function V(x,t) satisfying

i) V(x,t) $>$ 0 and V(0) = 0
ii) $\dot{V}(x,t) = \frac{\partial V(x,t)}{\partial t} + \frac{\partial V(x,t)}{\partial \mathbf{x}} \mathbf{f}(x,t) \leq -\mathbf{W}(\mathbf{x}) \leq 0$
iii) $V(x,t) \to \infty$ as $||x|| \to \infty$

where W(x) is a continous function. Then all solutions x(t) of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{t})$ are uniformly globally bounded and:

$$\lim_{t \to \infty} W(x(t)) = 0$$

In addition if W(x) $>$ 0, then the equilibrium point $\mathbf{x_e} = 0$ is UGAS (p. 536 Fossen 2011 [**Fossen2011**]).

## A.2 Matrosov's Theorem

Concider the system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t), \mathbf{x}(0) = \mathbf{x_0}, x \in \mathbb{R}^n \tag{81}$$

If for this system there exists:

- a locally Lipschitz function V: $\mathbb{R} \times \mathbb{R}^n \to \mathbb{R}_+$
- a continous positive semi-definite function U: $\mathbb{R}^n \to \mathbb{R}_+$
- functions $\alpha_1$, $\alpha_2 \in \kappa_\infty$

such that

1. $\alpha_1(||x||) \leq V(t,x) \leq \alpha_2(||x||) \forall (t,x) \in \mathbb{R} \times \mathbb{R}^n$
2. $\dot{V}(t,x) \leq -U(x)$ for almost all (t,x) $\in \mathbb{R} \times \mathbb{R}^n$

and for each $0 < \delta \leq \Delta$ and H(0, $\Delta$) $\subseteq \mathbb{R}^n$ there exist:

- a locally Lipschits function W : $\mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$
- a continous function Y: $\mathbb{R}^n \to \mathbb{R}$
- a strictly positive numbers $\epsilon_1, \epsilon_2, \psi > 0$

such that

3. max $\{W(t,x)|, |Y(x)|\} \leq \psi \forall (t,x) \in \mathbb{R} \times H(0,\delta)$
4. $\dot{W}(t,x) \leq Y(x) \forall (t,x) \in \mathbb{R} \times \mathbb{R}^n$
5. $\mathbf{x} \in H(\delta, \Delta) \bigcap \{\mathbf{x} : U(\mathbf{x}) \leq \epsilon_1\} \Rightarrow Y(\mathbf{x} \leq -\epsilon_2$

Then the origin of the system 81 is UGAS (p. 537 Fossen 2011 [**Fossen2011**]).

# B Mathematical Derivations

## B.1 Positive definiteness of P, full observer design

By assuming that $L_1, L_2$ and $L_3$ is diagonal and positive definite, and defining P

$$\mathbf{P} = \begin{bmatrix} L_2 & 0 & -\mathbf{I} \\ 0 & M & 0 \\ -\mathbf{I} & 0 & L_3^{-1}L_1 \end{bmatrix}$$

We can show that $P > 0$ by using a symmetric block matrix Y defined below.

$$\mathbf{P} = \mathbf{Y} = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

where $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are defined as follows.

$$\mathbf{C} = \begin{bmatrix} M & 0 \\ 0 & L_3^{-1}L_1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & -I \end{bmatrix}, \mathbf{A} = \begin{bmatrix} L_2 \end{bmatrix}$$

From the hint given in task 3.2 the symmetric block matrix Y is positive definite iff $\mathbf{A}$ and $\mathbf{C} - \mathbf{B^T A^{-1} B}$ is positive definite. Since $\mathbf{A} = \mathbf{L_2}$ thats assumed positive definite. What's left is to show that $\mathbf{C} - \mathbf{B^T A^{-1} B}$ positive definite.

$$\begin{aligned} \mathbf{C} - \mathbf{B^T A^{-1} B} &= \begin{bmatrix} M & 0 \\ 0 & L_3^{-1}L_1 \end{bmatrix} - \begin{bmatrix} 0 \\ -I \end{bmatrix} L_2^{-1} \begin{bmatrix} 0 & -I \end{bmatrix} \\ &= \begin{bmatrix} M & 0 \\ 0 & L_3^{-1}L_1 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} L_2^{-1} \\ &= \begin{bmatrix} M & 0 \\ 0 & L_3^{-1}L_1 - L_2^{-1} \end{bmatrix} \end{aligned}$$

One can see from this result that $L_3^{-1}L_1 - L_2^{-1} > 0$ in order to achieve a positive definite matrix $\mathbf{P}$. This can be rewritten to $L_1L_2 - L_3 > 0$.

## B.2 Time differenting V for simplified observer

$$\begin{aligned} \dot{V} &= \dot{\tilde{\eta}}^T L_2 \tilde{\eta} + \tilde{\eta}^T L_2 \dot{\tilde{\eta}} + \dot{\tilde{\nu}} \mathbf{M} \tilde{\nu} + \tilde{\nu}^T \mathbf{M} \dot{\tilde{\nu}} \\ &= [\mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta}]^T L_2 \tilde{\eta} + \tilde{\eta}^T L_2 [\mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta}] \\ &+ M^{-1}[-\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T L_2 \tilde{\eta}]^T \mathbf{M}\tilde{\nu} + \tilde{\nu}^T \mathbf{M}[\mathbf{M}^{-1}(-\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T L_2 \tilde{\eta})] \\ &= -\tilde{\eta}^T (L_1^T L_2 + L_2 L_1)\tilde{\eta} - \tilde{\nu}^T (\mathbf{D}^T + \mathbf{D})\tilde{\nu} \end{aligned}$$

## B.3 Time differenting the LFC

$$\begin{aligned}
\dot{W}_1 &= \dot{\tilde{\eta}}^T L_2\tilde{\eta} + \tilde{\eta}^T L_2\dot{\tilde{\eta}} + \dot{\tilde{\nu}}\mathbf{M}\tilde{\nu} + \tilde{\nu}^T\dot{\mathbf{M}} + \dot{\tilde{b}}^T L_3^{-1}L_1\tilde{b} + \tilde{b}^T L_3^{-1}L_1\dot{\tilde{b}} \\
&= [\mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta}]^T L_2\tilde{\eta} + \tilde{\eta}^T[\mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta}] \\
&+M^{-1}[-\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T L_2\tilde{\eta}]^T\mathbf{M}\tilde{\nu} + \tilde{\nu}^T\mathbf{M}[\mathbf{M}^{-1}(-\mathbf{D}\tilde{\nu} - \mathbf{R}(\psi)^T L_2\tilde{\eta})] \\
&+[-L_3\tilde{\eta}]^T L_3^{-1}L_1\tilde{b} + \tilde{b}TL_3^{-1}L_1[[-L_3\tilde{\eta}] - 2[\mathbf{R}(\psi)\tilde{\eta} - L_1\tilde{\eta}]^T\tilde{b} - 2\tilde{\eta}^T[-L_3\tilde{\eta}] \\
&= [\tilde{\nu}^T\mathbf{R}(\psi)^T - \tilde{\eta}^T L_1^T]L_2\tilde{\eta} + \tilde{\eta}^T L_2[\mathbf{R}(\psi)\tilde{\nu} - L_1\tilde{\eta}] \\
&+[-\tilde{\nu}^T\mathbf{D}^T - \tilde{b}^T\mathbf{R}(\psi) - \tilde{\eta}^T L_2^T\mathbf{R}(\psi)]\tilde{\nu} + \tilde{\nu}^T[-\mathbf{D}\tilde{\nu} + \mathbf{R}(\psi)^T\tilde{b} - \mathbf{R}(\psi)^T L_2\tilde{\eta}] \\
&+[-\tilde{\eta}^T L_3^T]L_3^{-1}L_1\tilde{b} + \tilde{b}L_1L_3^{-1}[-L_3\tilde{\eta}] - 2[\tilde{\nu}^T\mathbf{R}(\psi)^T - \tilde{\eta}^T L_1^T]\tilde{b} - 2\tilde{\eta}^T[-L_3\tilde{\eta}]
\end{aligned}$$

Finishing the multiplication and sorting terms that cancel out, the equation starts to reduce down..

$$\begin{aligned}
\Rightarrow \dot{W}_1 &= \tilde{\nu}^T\mathbf{R}(\psi)^T L_2\tilde{\eta} - \tilde{\nu}^T\mathbf{R}(\psi)^T L_2\tilde{\eta} + \tilde{\eta}^T L_2\mathbf{R}(\psi)\tilde{\nu} - \tilde{\eta}^T L_2\mathbf{R}(\psi)\tilde{\nu} \\
&-\tilde{\eta}^T L_1^T L_2\tilde{\eta} - \tilde{\eta}^T L_2 L_1\tilde{\eta} - \tilde{\nu}^T\mathbf{D}^T\tilde{\nu} - \tilde{\nu}^T\mathbf{D}\tilde{\nu} \\
&+\tilde{b}^T\mathbf{R}(\psi)\tilde{\nu} - 2\tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} + \tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} - \tilde{\eta}^T L_3^T L_3^{-1}L_1\tilde{b} - \tilde{b}^T L_1 L_3^{-1}L_3\tilde{\eta} + 2\tilde{\eta}^T L_1^T\tilde{b} + 2\tilde{\eta}^T L_3^T\tilde{\eta} \\
&= -\tilde{\eta}^T[L_1^T L_2 + L_2 L_1 - 2L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu} \\
&+\tilde{b}^T\mathbf{R}(\psi)\tilde{\nu} + \tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} - \tilde{\eta}^T L_1\tilde{b} - \tilde{b}^T L_1\tilde{\eta} - 2\tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} + 2\tilde{\eta}^T L_1^T\tilde{b} \\
&= -\tilde{\eta}^T[L_1^T L_2 + L_2 L_1 - 2L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu} \\
&+\tilde{b}^T\mathbf{R}(\psi)\tilde{\nu} + \tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} - 2\tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} \\
&-\tilde{\eta}^T L_1\tilde{b} - \tilde{b}^T L_1\tilde{\eta} + 2\tilde{\eta}^T L_1^T\tilde{b}
\end{aligned}$$

The following holds true:

$$\begin{aligned}
\tilde{b}^T\mathbf{R}(\psi)\tilde{\nu} + \tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} - 2\tilde{\nu}^T\mathbf{R}(\psi)^T\tilde{b} = 0 \\
-\tilde{\eta}^T L_1\tilde{b} - \tilde{b}^T L_1\tilde{\eta} + 2\tilde{\eta}^T L_1^T\tilde{b} = 0
\end{aligned}$$

since they all are scalar and we may "transpose them around", following the rules for transposing, making them cancel out. This leaves us with

$$\dot{W}_1 = -\tilde{\eta}^T[L_1^T L_2 + L_2 L_1 - 2L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu}$$

And since $L_1 = L_1^T$ and $L_2 = L_2^T$, This gives our differentiated LFC as

$$\begin{aligned}
\dot{W}_1 &= -\tilde{\eta}^T[2L_1 L_2 - 2L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu} \\
&= -2\tilde{\eta}^T[L_1 L_2 - L_3]\tilde{\eta} - \tilde{\nu}^T[\mathbf{D}^T + \mathbf{D}]\tilde{\nu}
\end{aligned}$$

# C Figures and plots

## Deadreckoning mode for one second

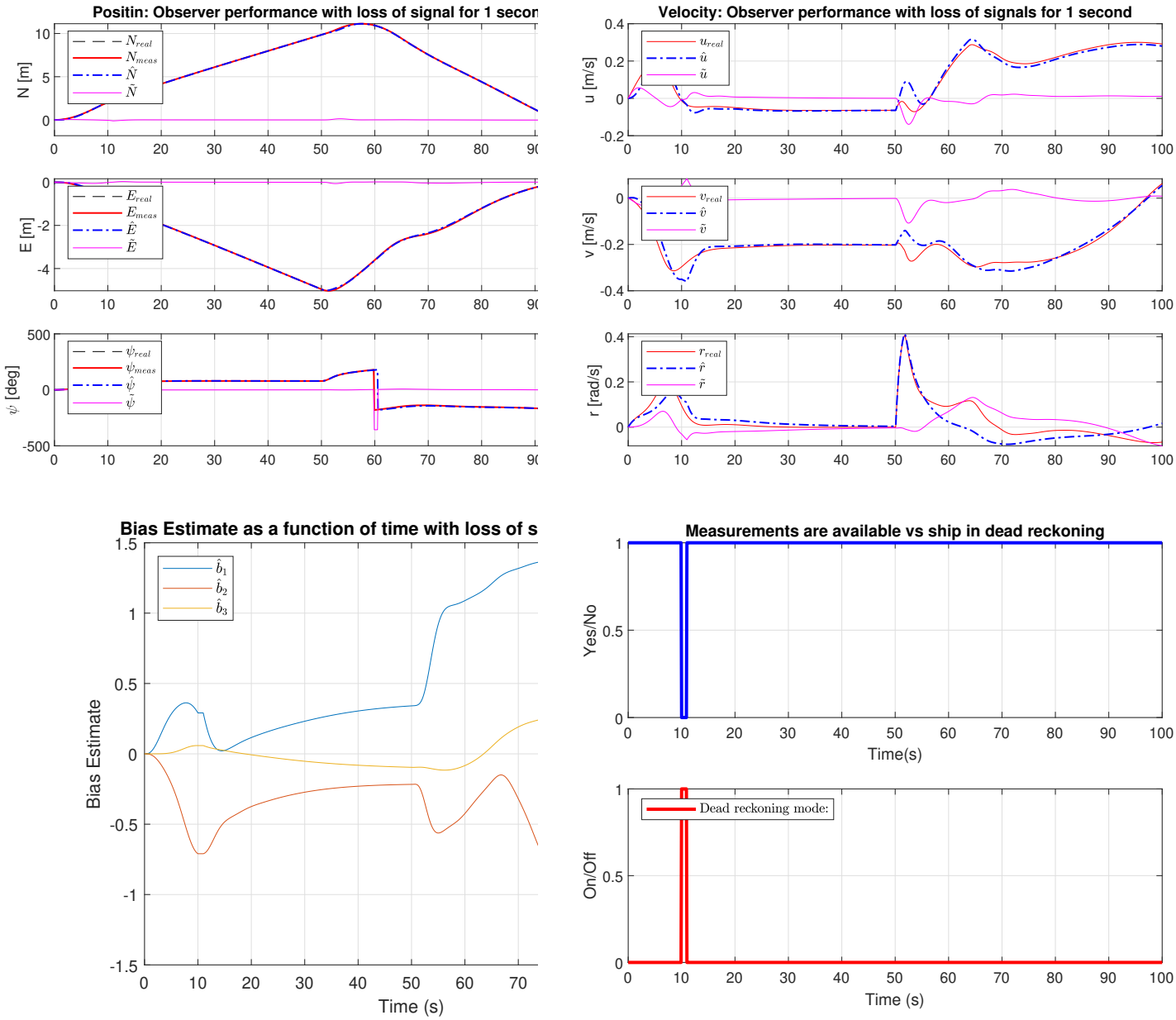
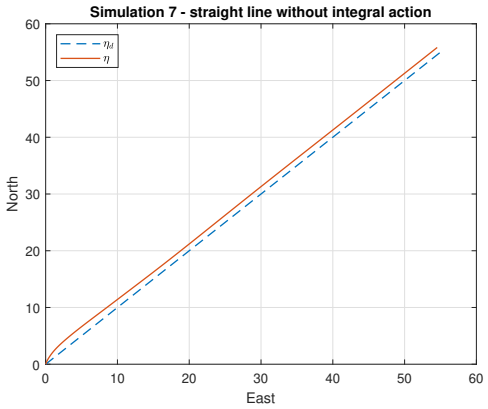
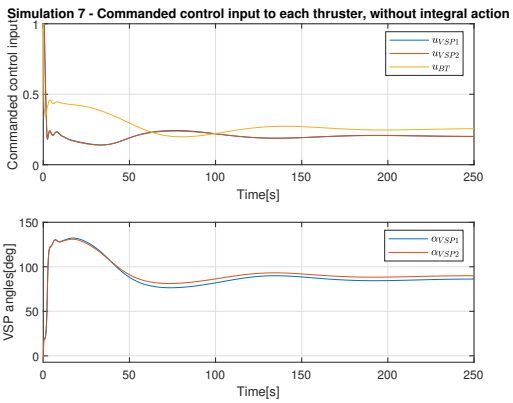



Figure 17: Simulation 6 - Observer performance with loss of signal for 1 second. Top left: Position. Top right: Velocities. Bottom left: Bias estimation. Bottom right, Dead reckoningmode enabled vs loss of signal detected

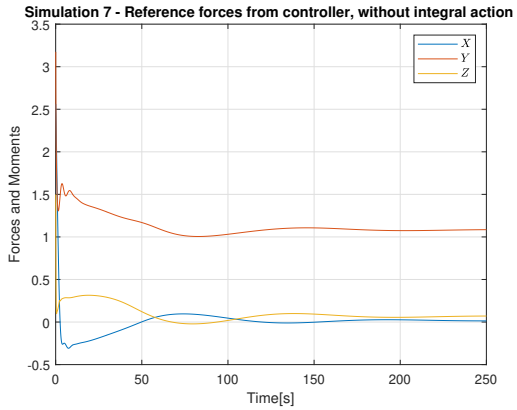
## Straight line figures without integral



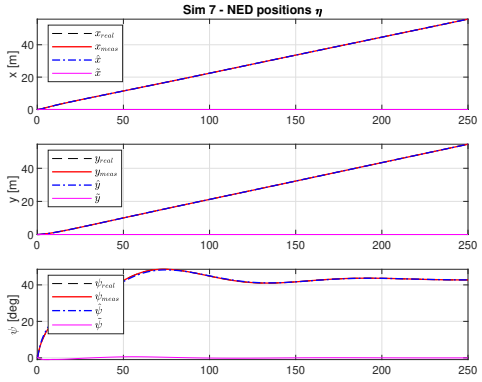

(a) XY-plot of the position in NED-frame



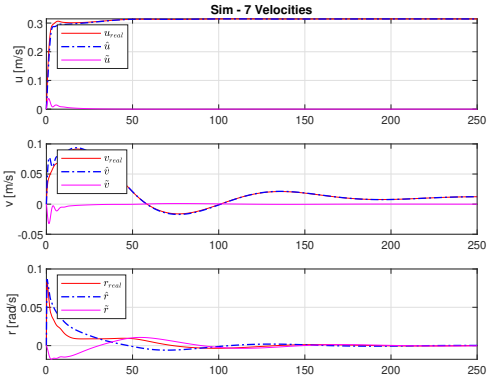

(b) Commanded control input to thrusters




(c) Reference forces in surge sway and yaw



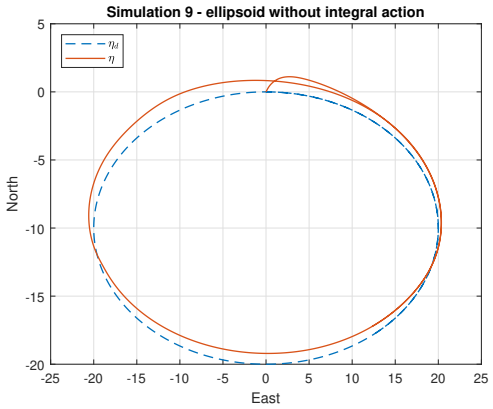

(d) $\eta$ as function of time



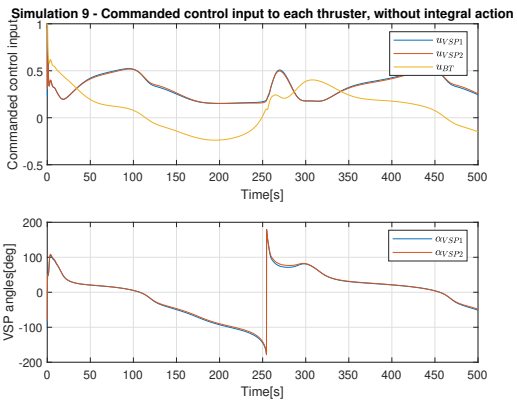

(e) Velocities

Figure 18: Simulation 7 - straight line without integral action

## Ellipsoidal path without integral

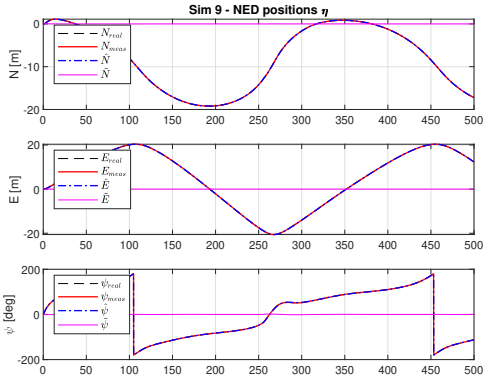



(a) XY-plot of the position in NED-frame

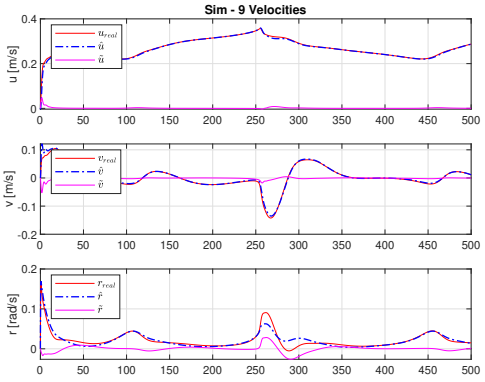



(b) Commanded control input to thrusters


(c) Reference forces in surge sway and yaw




(d) $\eta$ as function of time




(e) Velocities

Figure 19: Simulation 9 - ellipsoidal path without integral action

# D Simulink realizations

## Wrapper Function

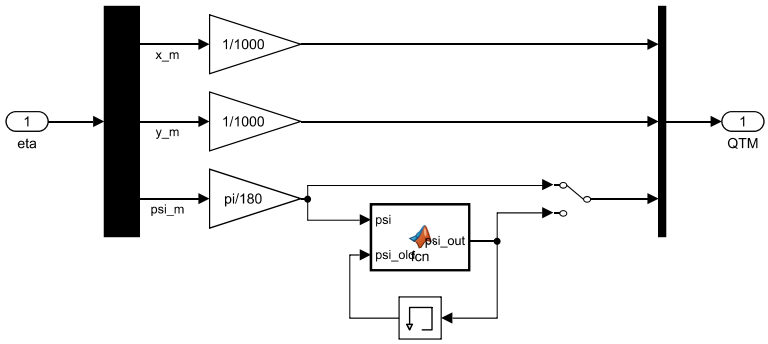



Figure 20: Wrapper function realization

## Thrust allocation implementation

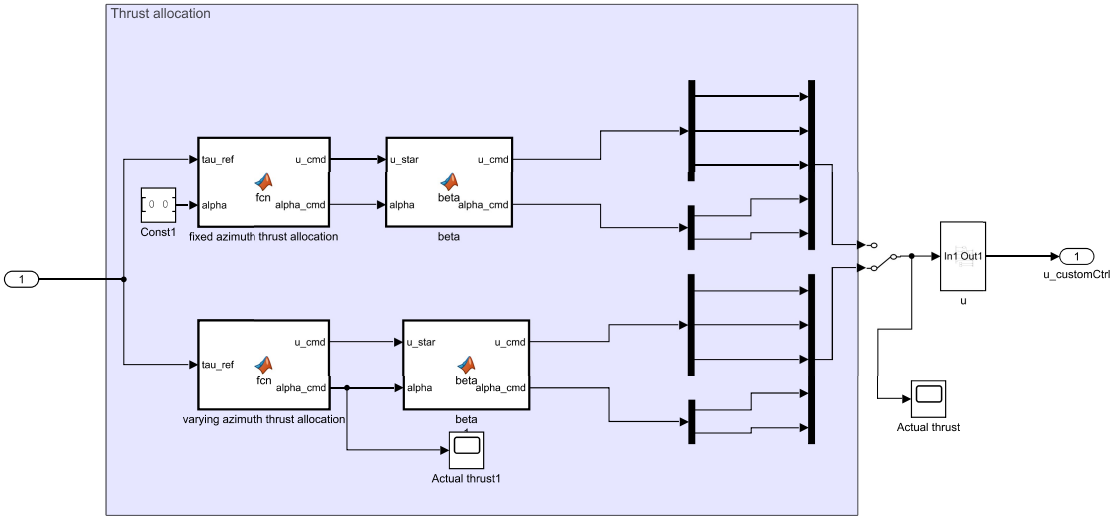



Figure 21: Thrust allocation implementation

## Nonlinear Observer

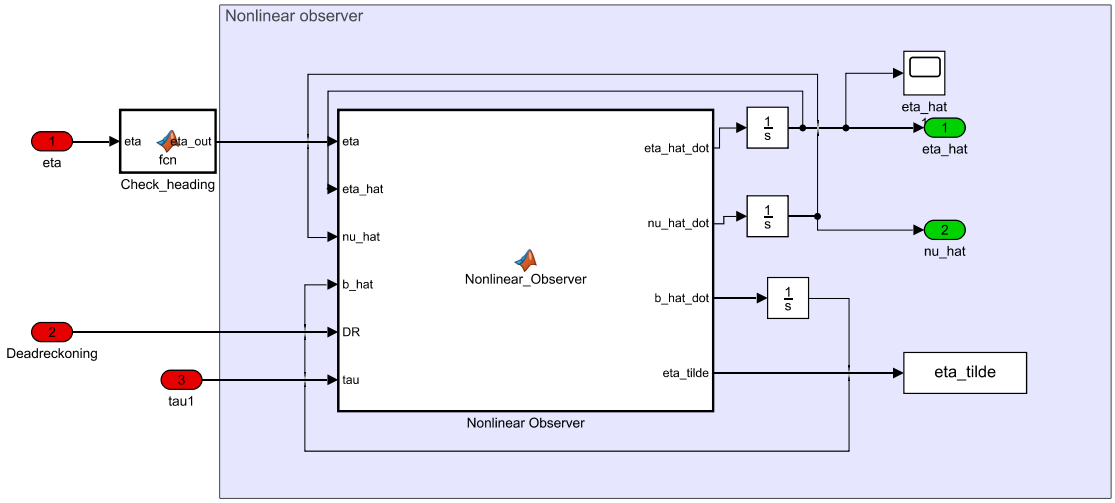



Figure 22: Nonlinear Observer Realization

## Deadreckoning and Loss of signal simulation

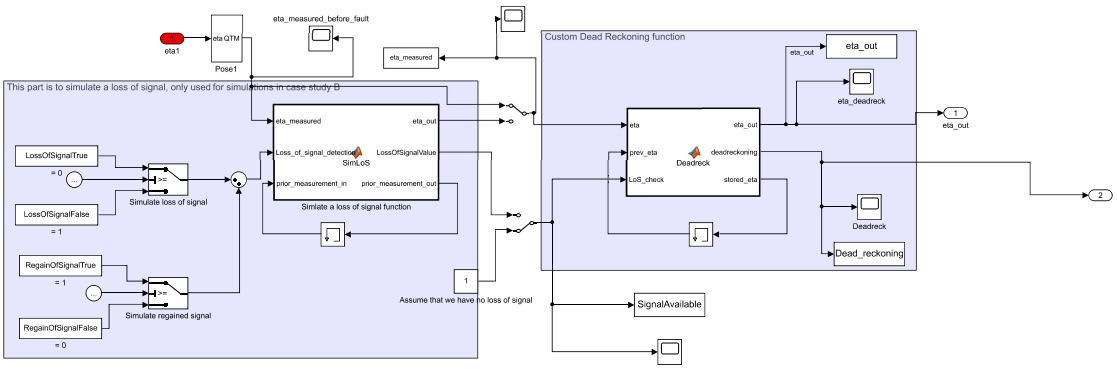



Figure 23: Deadreckoning and Loss of signal simulation

## Backstepping Controller

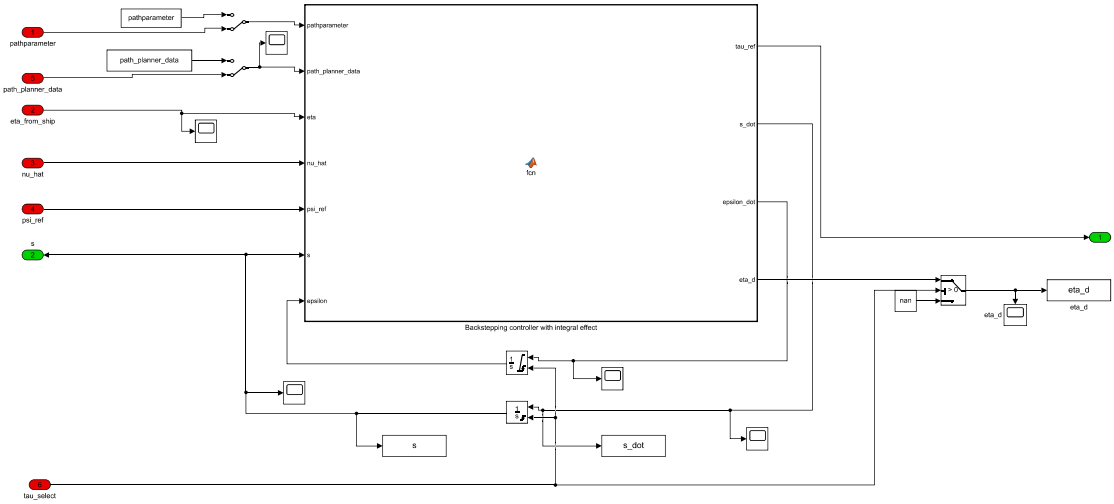



Figure 24: Backstepping controller realization

## Autopilot for extra simulations

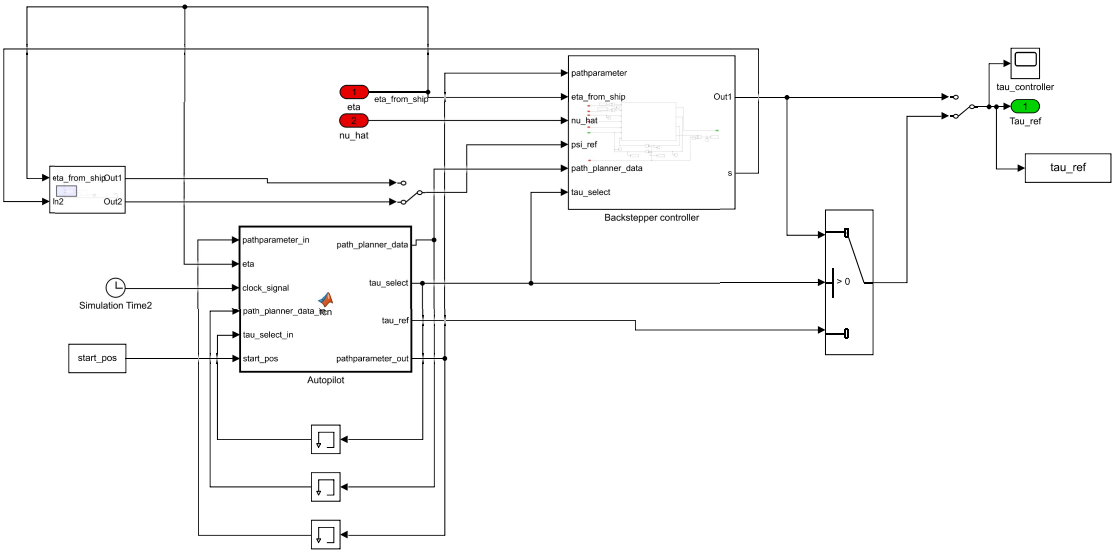



Figure 25: Backstepping controller realization

# E Matlab Code

## Fixed Azumith thrust allocation

```
function [ u_cmd , alpha_cmd ] = fcn(tau_ref , alpha)

k1 = 1.030; k2 = 1.030; k3 = 2.629;
k = [k1;k2;k3];
K = diag(k);    %Thrust coefficient matrix

Lx = -0.4574; Ly = 0.055; L_BT = 0.3875;  %Thruster cooridinates

B_alpha = [                         cos(alpha(1))
,                    cos(alpha(2))                              ,   0 ;
                                              sin(alpha(1))
,                     sin(alpha(2))                               ,   1 ;
                      - Ly * cos(alpha(1)) + Lx * sin(alpha(1)) , Ly * cos(alpha(2

u_cmd = pinv(B_alpha*K) * tau_ref;
alpha_cmd = alpha;

end
```

```
function [ u_cmd , alpha_cmd ] = beta ( u_star , alpha )

for i=1:2
    if u_star(i) < 0
        u_star(i) = abs(u_star(i));
        alpha(i) = alpha(i) - pi;
        if alpha(i) <= -pi
            alpha(i) = alpha(i) + 2*pi ;
        end
    end

    if u_star(i) > 1
        u_star(i) = 1;
    end
end

u_cmd = u_star;
alpha_cmd = alpha;

end
```

### Varying Azimuth thrust allocation

```
function [ u_cmd , alpha_cmd ] = fcn(tau_ref)

k1 = 1.030; k2 = 1.030; k3 = 2.629;
k = [k1;k1;k2;k2;k3];
K = diag(k);    %Thrust coefficient matrix

Lx = -0.4574; Ly = 0.055; L_BT = 0.3875;  %Thruster cooridinates

B_alpha = [ 1, 0, 1, 0, 0 ;
            0, 1, 0, 1, 1 ;
           -Ly, Lx, Ly, Lx, L_BT];

u_star = pinv( B_alpha * K ) * tau_ref ;

u_cmd = zeros(3,1);
alpha_cmd = zeros(2,1);

u_cmd(1) = sqrt(u_star(1)^2 + u_star(2)^2) ;
u_cmd(2) = sqrt(u_star(3)^2 + u_star(4)^2) ;
u_cmd(3) = u_star(5) ;

%%Returns values in [-pi/2, pi/2]
% alpha_cmd(1) = atan(u_star(2)/u_star(1));
% alpha_cmd(2) = atan(u_star(4)/u_star(3));

% Returns angles in  [-pi, pi]
alpha_cmd(1) = atan2(u_star(2),u_star(1));
alpha_cmd(2) = atan2(u_star(4),u_star(3));
end
```

```
function [ u_cmd , alpha_cmd ] = beta ( u_star , alpha )

for i=1:2
    if u_star(i) < 0
        u_star(i) = abs(u_star(i));
        alpha(i) = alpha(i) - pi;
        if alpha(i) < -pi
            alpha(i) = alpha(i) + 2*pi ;
        end
    end
%     alpha
%Saturate VSP to give maximum 100% thrust
    if u_star(i) > 1
        u_star(i) = 1;
    end
end

```

```
%Saturate bow thruster to a maximum of 1 (100% thrust)
if(u_star(3) > 1)
    u_star(3) = 1;
elseif u_star(3) <-1
    u_star(3) = -1;
end

u_cmd = u_star;
alpha_cmd = alpha;

end
```

### Nonlinear Oboserver

```
function [eta_hat_dot, nu_hat_dot, b_hat_dot, eta_tilde] = Nonlinear_Observer(eta

persistent M D
    [M, D] = deal(Controller_Data.M, Controller_Data.D);

scaling = 0.01/0.1;
L1 = scaling*Observergains(1:3,1:3);
L2 = scaling*Observergains(1:3,4:6);
L3 = scaling*Observergains(1:3,7:9);

psi = eta(3,1);

R_psi = [ cos(psi)  -sin(psi)   0  ;
          sin(psi)    cos(psi)  0  ;
          0             0         1  ] ;
R_tr = transpose(R_psi);

if DR
    L1 = 0*L1;
    L2 = 0*L2;
    L3 = 0*L3;
end
eta_tilde = eta - eta_hat;
eta_hat_dot = R_psi*nu_hat + L1*eta_tilde;
nu_hat_dot = M\(-D*nu_hat + R_tr*b_hat + tau + R_tr*L2*eta_tilde);
b_hat_dot = L3*eta_tilde;

end
```

### Deadreckoning and loss of signal simulation

```
function [eta_out, LossOfSignalValue, prior_measurement_out] = SimLoS(eta_measure

% Simulates a loss of signal/frozen signal, where it keeps sending out the
% same measurement while the LossOfSignalDetection returns 0 as stated in
% task 4.4 and 4.5 from case B

    if not(Loss_of_signal_detection)
        eta_out = prior_measurement_in;
        prior_measurement_out = prior_measurement_in;
        LossOfSignalValue = Loss_of_signal_detection;

    else
        eta_out = eta_measured;
        prior_measurement_out = eta_measured;
        LossOfSignalValue = Loss_of_signal_detection;
    end
end
```

```
function [eta_out, deadreckoning, stored_eta] = Deadreck(eta, prev_eta,
LoS_check)

    if not(LoS_check) %If loss of signal is true, i.e LoS_check = 0
        dead_signal = true;
        for i = 1:3 %Check if current measurement and 3 prior positions are equal
            if eta ~= prev_eta(i,:)'
              dead_signal = false;
            end
        end

        if dead_signal %If four last signal values are equal, set output to be eq
            deadreckoning = 1;
            eta_out = eta;

        else %Signal is not dead or frozen - Do normal stuff
            deadreckoning = 1;
            eta_out = eta;

        end
    else %If measurements are available - Do normal stuff
        deadreckoning = 0;
        eta_out = eta;
    end

stored_eta = prev_eta;
stored_eta(1:2,:) = prev_eta(end-1:end,:);
stored_eta(end,:) = eta';
end
```

### Backstepping Controller with integral action

```
function [tau_ref, s_dot, epsilon_dot,eta_d]  = fcn(pathparameter, path_planner_d
    %Calculation of of the different factors from manyfactors function

%Declaring constants from workspace and dealing them from Controller_Data
%from init
persistent K_p K_v K_i M D U_ref mu

    [K_p, K_v, K_i, M, D, U_ref, mu] = deal(Controller_Data.K_p,...
        Controller_Data.K_v, Controller_Data.K_i, Controller_Data.M,...
        Controller_Data.D, Controller_Data.U_ref, Controller_Data.mu);

    start_pos = path_planner_data(1:3);
    end_pos = path_planner_data(4:6);
    Radius = path_planner_data (7:8);
    Center = path_planner_data(9:10);

    rx = Radius(1);
    ry = Radius(2);

    Cx = Center(1); %center of elipsoid
    Cy = Center(2);

%Check if heading is close to pi/-pi and change the definition to [0,2pi];
if (abs(eta(3,1)) > 0.5*pi && eta(3,1) < 0)
    eta(3,1) = eta(3,1)+2*pi;
end

    psi_meas = eta(3,1);
%     psi = psi_ref;
    r = nu_hat(3,1);
%     integral_effect = [0; 0; yaw_V_err_integral];

%% Path planner
    if not(pathparameter) %If elipsoid
%% Calculating terms for eta and its derivatives
        xdf = (-rx*2*pi*sin(2*pi*s)); %x_d'
        ydf = (ry*2*pi*cos(2*pi*s)); %y_d'
        xdff = -rx*(2*pi)^2*cos(2*pi*s); %x_d''
        ydff = -ry*(2*pi)^2*sin(2*pi*s); %y_d''
        xdfff = rx*(2*pi)^3*sin(2*pi*s); %x_d'''
        ydfff = -ry*(2*pi)^3*cos(2*pi*s); %y_d'''

        df_dydf =  xdf/(ydf^2+xdf^2); %df/dx_d'
        df_dxdf = -ydf/(ydf^2+xdf^2); %df/dy_d'
        ddf_dxddf = 2*xdf*ydf/(xdf^2+ydf^2)^2; %d^2f/dx_d'^2
        ddf_ddydf = -2*xdf*ydf/(xdf^2+ydf^2)^2; %d^2f/dy_d'^2
        ddf_dxdfdydf = (ydf^2-xdf^2)/(ydf^2+xdf^2)^2; %d^2f/(dx_d'dy_d')
```

```
48
49          dh_dxdf = -U_ref*xdf/(xdf^2+ydf^2)^(3/2);
50          dh_dydf = -U_ref*ydf/(xdf^2+ydf^2)^(3/2);
51
52
53  %% ssa
54          DesiredHeading = atan2(ydf,xdf);
55          val_1 = DesiredHeading;
56          val_2 = DesiredHeading + 2*pi;
57          val_3 = DesiredHeading - 2*pi;
58
59          vals = [val_1, val_2, val_3];
60
61          [m, i] = min(abs(vals - psi_meas));
62
63          DesiredHeading_ssa = vals(i);
64  %       DesiredHeading_ssa
65
66  %% Declaring the outputs
67           %eta_d
68  %         eta_d = [Cx+rx*cos(2*pi*s), Cy+ry*sin(2*pi*s), DesiredHeading]';
69          eta_d = [Cx+rx*cos(2*pi*s), Cy+ry*sin(2*pi*s), DesiredHeading_ssa]';
70  %         eta_d = [Cx+rx*cos(2*pi*s), Cy+ry*sin(2*pi*s), atan2(ydf,xdf)]';
71
72          %eta_d'
73          eta_df = [xdf, ydf, df_dxdf*xdff+df_dydf*ydff]';
74
75          %eta_d''
76          eta_dff = [xdff, ydff, ddf_dxddf*xdff^2+ddf_ddydf*ydff^2+2*ddf_dxdfdydf*x
77
78          %U_s'
79          U_sf = dh_dxdf*xdff + dh_dydf*ydff;
80
81          %U_s
82          U_s = U_ref/sqrt(xdf^2+ydf^2);
83
84      else %else straight line
85
86          Trajectory = (end_pos(:,1) - start_pos(:,1))*s + start_pos(:,1);
87
88
89              DesiredHeading = atan2((end_pos(2,:)-start_pos(2,:)),(end_pos(1,:)-st
90  %           DesiredHeading = atan((end_pos(2,:)-start_pos(2,:))/(end_pos(1,:)-start
91
92
93  %           DesiredHeading
94  %% ssa
95          val_1 = DesiredHeading;
96          val_2 = DesiredHeading + 2*pi;
```

```
97          val_3 = DesiredHeading - 2*pi;
98
99          vals = [val_1, val_2, val_3];
100
101         [m, i] = min(abs(vals - psi_meas));
102
103         DesiredHeading_ssa = vals(i);
104
105         %%
106
107         %eta_d
108 %           eta_d = transpose([Trajectory(1), Trajectory(2), DesiredHeading]);
109         eta_d = transpose([Trajectory(1), Trajectory(2), DesiredHeading_ssa]);
110 %         eta_d = transpose([s*x2+(1-s)*x1, s*y2+(1-s)*y1, atan2(y2-y1,x2-x1)]);
111
112         %eta_d'
113         eta_df = transpose([end_pos(1,:) - start_pos(1,:), end_pos(2,:) - start_p
114 %         eta_df = transpose([x2-x1, y2-y1, 0]);
115
116         %eta_d''
117         eta_dff = transpose([0 0 0]);
118
119         %U_s'
120         U_sf = 0;
121
122         %U_s
123         U_s = U_ref/norm(eta_df,2);
124
125     end
126
127 %% Output calculations
128     V1s = -transpose(eta-eta_d)*eta_df; %eq (38)
129
130     R_psi = [ cos(psi_meas)  -sin(psi_meas)   0  ;
131               sin(psi_meas)    cos(psi_meas)  0  ;
132                  0              0          1  ] ;
133     R_tr = transpose(R_psi);
134
135     S_r = [ 0  -r   0  ;
136             r  0   0   ;
137             0  0   0   ] ;
138
139     S_tr = transpose(S_r);
140
141     %S_dot
142     s_dot = U_s - mu*V1s/norm(eta_df,2);
143     % s_dot = U_s + mu*eta_df'/norm(eta_df,2)*(eta - eta_d); %eq: 55
144
145     %Calculate Z1_dot
```

```
146     z1_dot = -S_r*R_tr*(eta - eta_d) + nu_hat - R_tr*eta_df*s_dot;
147 
148     z_1 = R_tr*(eta-eta_d);
149 
150 
151     %Calculate d_dt
152     d_dt = S_tr*R_tr*eta_df*U_s + R_tr*s_dot*(eta_dff*U_s + eta_df*U_sf);
153 
154     alpha_1_dot = -K_p*z1_dot + d_dt;
155 
156     alpha_1 = -K_p*z_1 + R_tr*eta_df*U_s;
157 
158     % Tau
159     z_2 = nu_hat - alpha_1;
160 
161     epsilon_dot = z_2;
162 
163 
164 %With integral effect added
165     tau_ref = D*alpha_1 + M*alpha_1_dot - K_v*z_2 - K_i*epsilon;
166 
167 %Withhout integral effect
168 %  tau_ref = D*alpha_1 + M*alpha_1_dot - K_v*z_2;
169 
170 
171 end
```

### Autopilot function

```
function [path_planner_data, tau_select,tau_ref, pathparameter_out] = fcn(pathpar
%Straight-line
% start_pos = [0;0;0];
end_pos = [10;10;0];
%elipsoid pathparameter = 0, straight line = 1
pathparameter_out = pathparameter_in;
%Elliptical
rx = 10;
ry = 20;
Cx = 0; %center of elipsoid
Cy = 0;
path_planner_data = [start_pos ; end_pos ; rx ; ry ; Cx ; Cy];
% path_planner_data = zeros(10,1);
tau_ref = [0.1;0.1;0.1];
tau_select = tau_select_in;
if clock_signal==50
    start_pos = [eta(1);eta(2);eta(3)];
    end_pos = [cos(eta(3))*200;sin(eta(3))*200;eta(3)];
    Cx = eta(1)-rx;
    Cy = eta(2);
    path_planner_data = [start_pos ; end_pos ; rx ; ry ; Cx ; Cy];
    tau_select = 1;

end

if clock_signal > 50
    path_planner_data = path_planner_data_in;

end

end
```