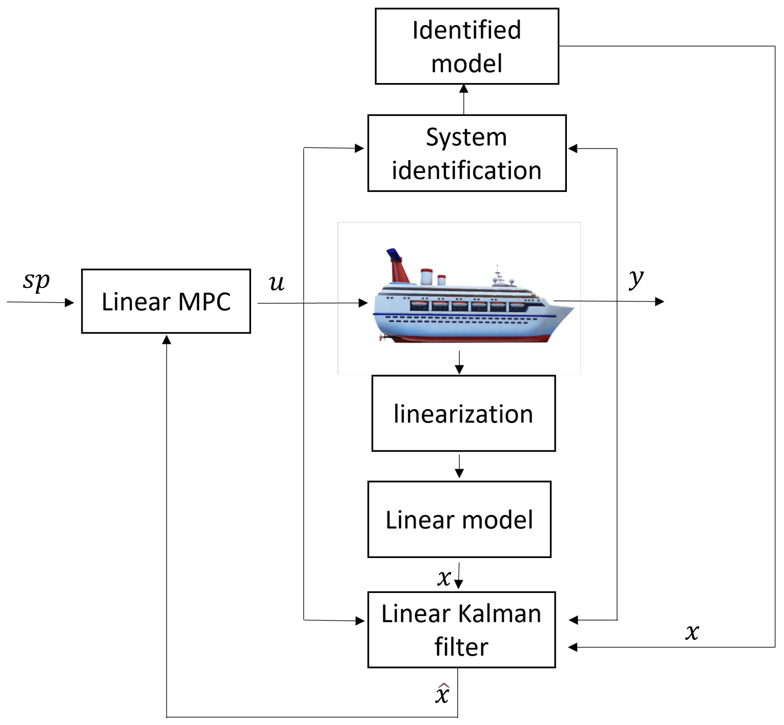
University of
South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2022
Industrial IT and Automation

Dynamic positioning, system identification and control of marine vessels



Nour Mohamad Bargouth

Faculty of Technology, Natural Sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis 2022
Title: *Dynamic positioning, system identification and control of marine vessels*
Pages: *160*
Keywords: *Dynamic positioning, MIMO systems, model predictive control MPC, optimal control with integral action, system identification*

Student: *Nour Mohamad Bargouth*
Supervisor: *David Di Ruscio*
External partner: *None*

Summary:
The dynamic positioning system is a system that aims to keep a marine vessel at the desired position under different weather circumstances by controlling its own propeller and thruster system. Different types of controllers can be used to achieve that. In this thesis, standard MPC, MPC with integral action, and optimal control with integral action are implemented and tested to control the position of a selected vessel. The theory behind the vessel movement at sea, environmental disturbances acting on it and the propeller and thruster system forces and moment are defined. The low-frequency part of the Balchen non-linear model is used in this thesis. It is linearized and the stability, controllability and observability have been analysed. The linear model is used to develop a linear MPC controller and optimal control. The accuracy of the mathematical model depends on the accuracy of some parameters related to the studied vessel dynamics and geometry. Therefore, the Deterministic and Stochastic system identification and Realization algorithm is proposed to get an identified model and formulate a model-free MPC and model-free optimal control. MATLAB software is used to perform all simulations that shows the efficiency of the developed control system in tracking the desired position under different environmental disturbances.

Preface

The dynamic positioning of marine vessels is an important issue especially in oil and gas industry for safety and economy demands. Many research had been done to develop efficient dynamic positioning systems but one of the challenges was about finding an accurate mathematical model to be used in developing model based control systems.

This master's thesis develops a dynamic positioning DP system based mainly on Balchen model. It explores the efficiency of different types of controllers, standard MPC, MPC with integral action and optimal control with integral action.

In addition, it studies the possibility of using system identification method to get an identified model from input- output data and use it to develop a model-free MPC and model-free optimal control with integral action.

The developed control system is implemented and tested, and it was efficient in tracking the desired position under different weather circumstances.

The entire work was carried out at university of south- eastern Norway, Porsgrunn. MathWorks' MATLAB was used to perform all simulations.

I would like to thank my supervisor, Associate Professor David Luigi De Ruscio for his guidance and supervision throughout the semester. My gratitude goes to my family who supported me and who I would not succeed without their patience and support.

Porsgrunn, 14th May 2022

Nour Mohamad Bargouth

Contents

List of Figures

List of Tables

Nomenclature

Symbol	**Explanation**
x	state vector
u	input vector
y	output vector
x_{su}	vessel position in surge [m]
x_{sw}	vessel position in sway [m]
ψ	vessel heading, vessel position in yaw [rad]
v_{su}	vessel speed in surge [m/s]
v_{sw}	vessel speed in sway [m/s]
v_{ψ}	vessel heading rate [rad/s]
$v_{c_{su}}$	water current speed in surge [m/s]
$v_{c_{sw}}$	water current speed in yaw [m/s]
N_c	water current moment in yaw
$F_{w_{su}}$	wind force in surge [N]
$F_{w_{sw}}$	wind force in sway [N]
N_w	wind moment in yaw direction [Nm]
$F_{t_{su}}$	thrust force in surge [N]
$F_{t_{sw}}$	thrust force in sway [N]
N_t	thrust moment in yaw [Nm]
$R(\psi)$	Transformation matrix
ρ	wind density [kg/m^3]
V_w	measured wind speed
V_{w_r}	relative wind speed
γ	wind angle of attack
β	measured wind direction
C_x, C_y, C_N	wind coefficients
A_F	windage area for head wind
A_L	windage area for beam wind
L	vessel overall length
τ	Generalized thruster force
T	Configuration matrix
K	Coefficient diagonal matrix
l_i	Thruster number i lever arm

Symbol	**Explanation**
F_w	wind force
F_t	Thruster force
F_c	water current force
d_1, d_2, d_3, d_4	drag and moment coefficients
m_1, m_2, m_3	inertial coefficients
η_1, η_2, η_3	zero mean Gaussian white noise
w	Process noise
v	measurement noise

Abbreviations	**Explanation**
DP	Dynamic Positioning
GPS	Global Positioning System
DOF	Degrees Of Freedom
NED	North, East, and Down coordinate system
3D	three dimensional
MPC	Model Predictive Control
PID	Proportional, Integral, Derivative
LQ	Linear Quadratic
MIMO	multiple input multiple output
DSR	Deterministic and Stochastic System Identification and Realization

1 Introduction

In this chapter, an overview of dynamic positioning systems is given. Previous research about DP systems are introduced in addition to the objectives of this report. Finally, the report structure is listed at the end of this chapter.

1.1 Overview of Dynamic positioning systems

A dynamic positioning system is a computer-based system that aims to keep a vessel in a desired position under environmental disturbances by manipulating its thruster and propulsion system [1]. It gets data from different types of sensors, wind sensors and position sensors. Wind sensors measure wind speed and wind direction. While position sensors, usually GPS and gyro-compass, give information about vessel position and vessel heading respectively. The dynamic positioning system uses the collected data to compute a control signal that manipulates the propulsion system.

DP system is a MIMO system with three inputs and three outputs. System inputs are thrusters' forces in surge and sway, and thrusters moment around the vertical axis z. While the system outputs are vessel position in surge and sway, and the vessel heading in yaw. These movements are described in detail in chapter 2. Inputs, outputs, states, and disturbances are illustrated in figure 1.1

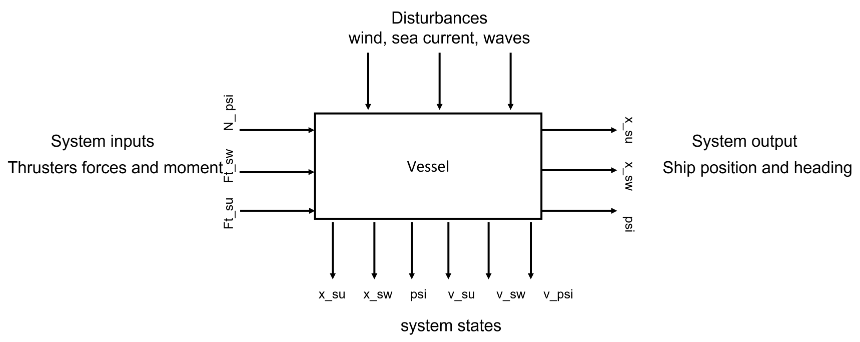


Figure 1.1: Overall block diagram of a vessel model.

Dynamic positioning system consists of three subsystems. The first one is the power subsystem that supplies the DP system with desired power. Thruster subsystem that supplies the DP system with variable thrust force and direction. Lastly, the DP control subsystem that coordinates with other subsystems to control vessel position [2].

Some typical applications of DP systems are drilling ships, diving support vessels, mine sweepers and supply vessels where it is used to increase the safety demands [3].

1.2 Previous research

Research about dynamic positioning systems began early in 1960 in the USA where a manual DP system is developed [4]. In 1977, Nils Albert Jenssen, Steinar Sælid and Professor Jens Balchen tested their new control system for dynamic positioning on board Stolt-Nielsens Rederi's vessel 'Seaway Eagle' [4] [1]. Balchen idea was to discard the classical control technique, PID control, and use the concept of modern control theory such as kalman filter and optimal control [1]. This DP system is used by Kongsberg våpen industry. Kongsberg had delivered its first dynamic positioning system in 1977, and since that it has supplied more than 4000 DP systems [3].

Kongsberg has dynamic positioning systems in different modes. High Precision control mode that keeps the vessel in its position in any weather conditions, Relaxed control mode where it is not guaranteed that the vessel stays in its operating position and it is used in calm weather conditions, and Green DP mode that reduces the CO_2 emissions by as much as 20 percent by using non-linear model predictive control method [3].

Thor I Fossen [5] has also worked with DP systems and developed a mathematical model that describes vessel dynamics. He has developed a Marine Systems Simulator Toolbox in MATLAB, *MSS*, to simulate DP systems [6].

1.3 Objectives

Since the control subsystem is the core of the dynamic positioning system, the focus in this report will be on the control subsystem. The control method that is described in detail in this report is a model predictive control method MPC which is implemented and tested in simulations that are performed in MATLAB. This method is a model-based method and the mathematical model that is used is the Balchen model [1].

Balchen model is a non-linear model that is derived from Newtons' second law. It is linearized in this report and implemented in MATLAB to compare the behaviour of the linear and the non-linear model. The linear model is used to develop a linear MPC. The linear MPC in its standard form is implemented and tested, but it takes too much time to solve the optimization problem due to unnecessary calculations. Therefore, reduced size MPC is developed and tested. A more simplified MPC developed by David Di Ruscio is also implemented and tested in the report. To get a control system that is independent of process- and measurement noise, linear MPC with integral action is developed.

To investigate the possibility of using a model-free control method, Deterministic and Stochastic system identification and Realization algorithm "DSR" is used. Some experiments in closed-loop simulation are done to collect inputs- an outputs data. Then *dsr-e* function is used to get an identified model to be used to develop a model- free MPC control system. Figure 1.2 shows the block diagram of the DP control system developed in this report.

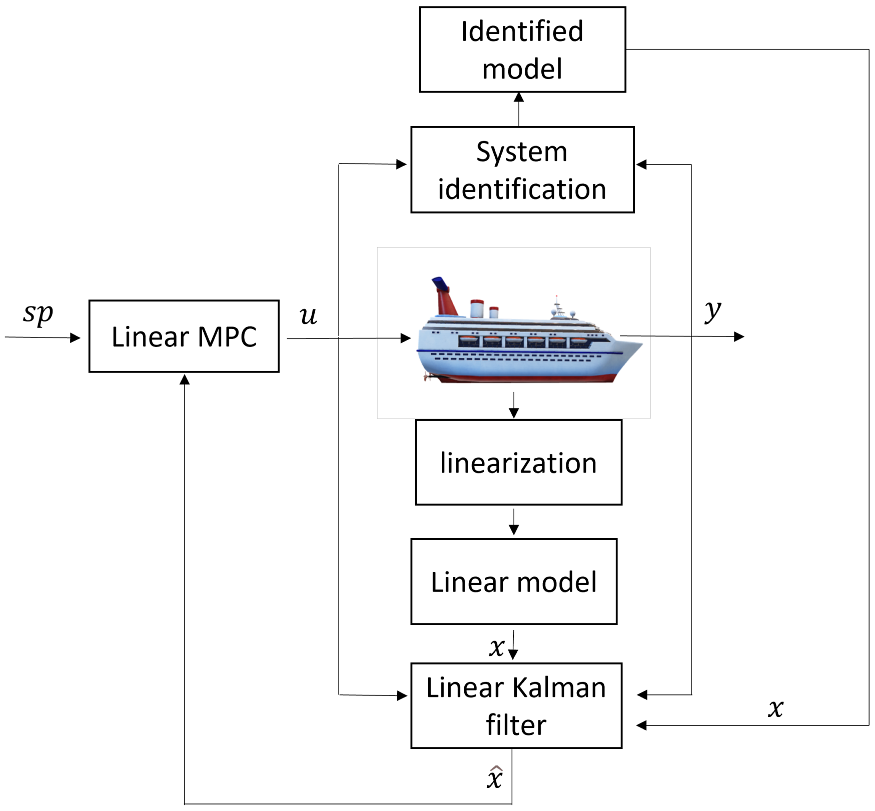


Figure 1.2: Block diagram of DP control system with linear MPC, linear model and identified model

1.4 Report structure

This report consists of six chapters. The first chapter is an introduction that gives an overview of a dynamic positioning system, previous work with it and objectives of this report.

The second chapter describes the theory behind vessel movement and coordinate systems. Furthermore, it illustrates forces acting on a vessel at sea bed and how to calculate them. It defines Balchen mathematical model, Kalman filter, MPC control, and system identification.

The third chapter describes the methods used to develop an efficient DP system, and the fourth one shows the simulations performed and discuss the results.

The fifth chapter discusses the limitations to the project and some ideas for future work. While the last one brings conclusion to the outcomes of this report.

2 Theory

This chapter describes some basic theories about the vessel movement along two different coordinate systems and the different forces acting on a vessel at sea. In addition, the theory behind Model predictive control, Kalman filter and system identification algorithm is given in this chapter.

2.1 Vessel movement and coordinate systems

A floating vessel that moves freely in a 3D space, can move in six degrees of freedom, 6 DOF. These degrees of freedom describe the vessel motion at sea. Three of them describe translational movements along x, y and z axes, while the other three describe the rotational movements around those axes [5].

The translational movements along x, y and z axes are surge, sway and heave movements respectively. Whereas the rotational movements around x, y and z axes are pitch, roll and yaw respectively. The 6 DOF are illustrated in figure 2.1.

A dynamic positioning system controls three movements of a vessel. It controls surge, sway, and yaw movements, 3 DOF. It is important to differentiate between two coordinate systems. The first one is NED coordinate system that refers to north, east and down [5]. It is the same coordinate system that is used everyday to describe wind direction for example. Whereas the second one is Body frame coordinate system that has the same origin as the NED frame, but its axes are parallel to the vessel body axes [1]. The two coordinates system are illustrated in figure 2.2.

To calculate the position and velocity of a vessel, all forces acting on the vessel must be transformed from NED coordinate system to Body-frame coordinate system. Whereas the calculated position is then transformed to NED coordinate system. That transformation is done by using transformation matrix $R(\psi)$ given by equation 2.1 [5].

$$R(\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

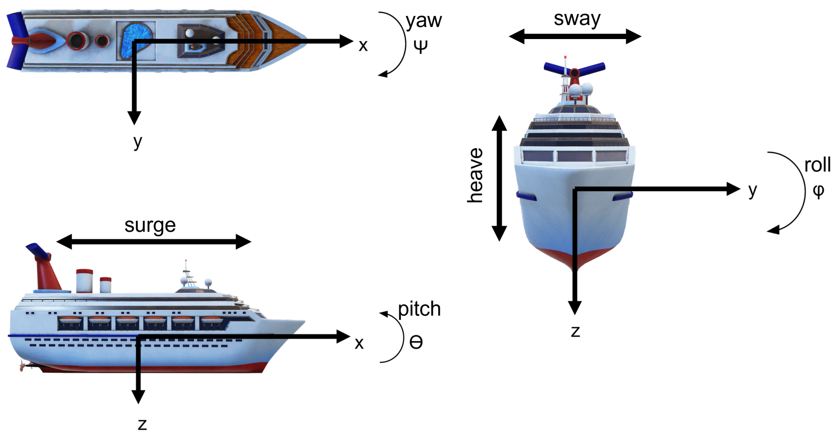


Figure 2.1: Six degrees of freedom of a vessel, a modified figure in [7]

2.2 Forces acting on the vessel

This section describes the different types of forces acting on a vessel. These forces are external environmental forces: wind, water current and sea waves forces, and internal forces which are vessel's thrusters and propulsion forces.

2.2.1 Wind forces and moment

Wind forces and moment acting on a moving vessel are proportional to the square of the relative wind speed V_{w_r}, to wind angle of attack γ and to the windage area [5]. The windage area is the area of a vessel that is exposed directly to the wind, and it depends on the geometry of the vessel [8].

In dynamic positioning systems, wind forces in surge and sway, and wind moment in yaw are needed to be calculated and included in the control system to compensate the effect of wind forces and moment on the vessel movement.

Wind forces and moment can be computed by using equation 2.2 for symmetric vessels [5].

$$F_w = \begin{bmatrix} F_{w_{su}} \\ F_{w_{sw}} \\ N_w \end{bmatrix} = \frac{1}{2}\rho V_{w_r}^2 \begin{bmatrix} C_x \cos(\gamma) A_F \\ C_y \sin(\gamma) A_L \\ C_N \sin(2\gamma) A_L L \end{bmatrix} \tag{2.2}$$

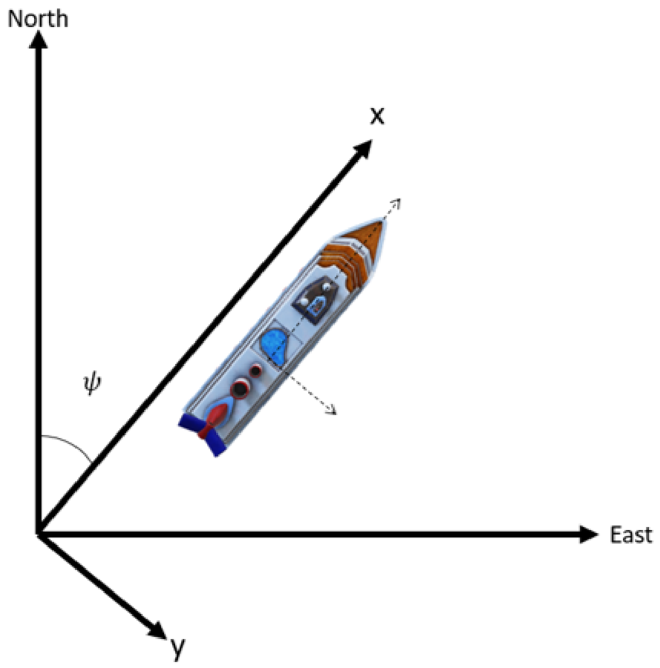


Figure 2.2: NED and Body coordinate systems

Where:

$F_{w_{su}}$: wind force in surge [N].

$F_{w_{sw}}$: wind force in sway [N].

N_w: wind moment in yaw [N.m].

ρ: the density of the wind which equals approximately to 1.23 $[kg/m^3]$ [8].

V_{w_r}: relative wind speed $[m/s]$ that is computed using equation 2.3.

C_x, C_y, C_N: wind coefficients that takes values, $C_x \in \{0.50, 0.90\}$, $C_y \in \{0.70, 0.95\}$, and $C_n \in \{0.05, 0.20\}$ [5].

A_F: windage area for head wind, where the wind has longitudinal direction as shown in figure 2.3.

A_L: windage area for beam wind, i.e. wind from transverse direction as seen in figure 2.4.

L: vessel over all length $[m]$

$$V_{w_r} = \sqrt{v_{w_{sur}}^2 + v_{w_{swr}}^2} \tag{2.3}$$

where:

$v_{w_{su_r}}$: relative wind speed in surge which is the deviation between the vessel speed in surge v_{su} and the wind speed in surge $v_{w_{su}}$.

$v_{w_{sw_r}}$: relative wind speed in sway which is the deviation between vessel speed in sway v_{sw} and wind speed in sway $v_{w_{sw}}$.

$$v_{w_{su}} = V_w \cos(\beta - \psi) \tag{2.4}$$

$$v_{w_{su}} = V_w \sin(\beta - \psi) \tag{2.5}$$

where:

V_w: measured wind speed [m/s].

β: measured wind direction [rad].

Wind scale is listed in table 2.2.1 [9].

Table 2.1: Wind scale

Wind speed [m/s]	**Description**
0.5 - 1.8	light air
1.9 - 3.3	light breeze
3.4 - 5.4	gentle breeze
5.5 - 7.9	breeze
8.0 - 11.0	fresh breeze
11.1 - 14.1	strong breeze
14.2 - 17.2	near gale
17.3 - 20.8	gale
20.9 - 24.4	strong gale
24.5 - 28.5	storm
28.6 - 32.6	violent storm
> 32.6	hurricane

ρ, A_F, A_L,L are constants, and by assuming for simplicity that C_x, C_y and C_N are constants, so the wind force can be expressed by equation 2.6.

$$F_w = \begin{bmatrix} F_{w_{su}} \\ F_{w_{sw}} \\ N_w \end{bmatrix} = \begin{bmatrix} d_{w_1} V_{w_r}^2 \cos(\gamma) \\ d_{w_2} V_{w_r}^2 \sin(\gamma) \\ d_{w_3} V_{w_r}^2 \sin(2\gamma) \end{bmatrix} \tag{2.6}$$

where:

$d_{w_1} = \frac{1}{2}\rho C_x A_F$

$d_{w_2} = \frac{1}{2}\rho C_y A_L$

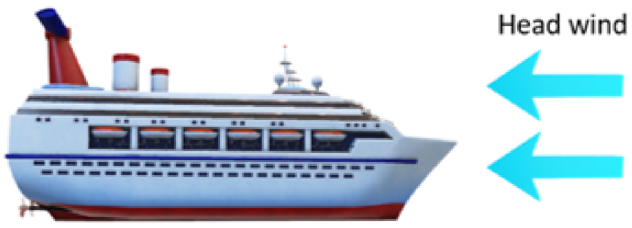


Figure 2.3: Head wind acting on a vessel

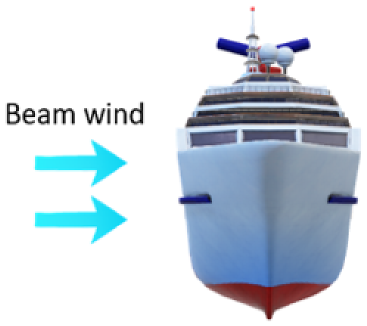


Figure 2.4: Beam wind acting on a vessel

$d_{w_3} = \frac{1}{2}\rho C_N A_L L$

MATLAB script that is used to calculate wind force in Body coordinate system is available in Appendix C.

2.2.2 Water current forces and moment

Water current forces acting on a vessel, are hydrodynamic forces that are proportional to the square of the difference between the vessel velocity and the water current velocity [9].

To add the current speed effect to the vessel motion equations of Balchen [1], it should be transformed to the Body-frame coordinate system. It is done by using the transpose of the transformation matrix $R(\psi)$ given by equation 2.1.

Balchen [1], modelled the water current speed as slowly varying parameters by using equations 2.7 to 2.9.

$$\dot{v}_{c_N} = \eta_{c_N} \tag{2.7}$$

$$\dot{v}_{c_E} = \eta_{c_E} \tag{2.8}$$

$$\dot{N}_{c\psi} = \eta_{c\psi} \tag{2.9}$$

where:

v_{c_N}: water current velocity in North direction in the NED coordinate system.

v_{c_E}: water current velocity in East direction in the NED coordinate system.

$N_{c\psi}$: water current moment in yaw in NED coordinate system.

$\eta_{c_N}, \eta_{c_E}, \eta_{c\psi}$: zero mean white noise processes.

$$\begin{bmatrix} v_{c_{su}} \\ v_{c_{sw}} \\ N_c \end{bmatrix} = R(\psi)^T \begin{bmatrix} v_{c_N} \\ v_{c_E} \\ N_{c\psi} \end{bmatrix} \tag{2.10}$$

where:

$v_{c_{su}}$: water current velocity in surge direction in the Body coordinate system.

$v_{c_{sw}}$: water current velocity in sway direction in the Body coordinate system.

N_c: water current effect in yaw in Body coordinate system.

2.2.3 Wave force

A dynamic positioning system can be simulated under influence of wave-induced forces by separating the first-order and second-order effects [5]:

- **First-order wave-induced forces**: wave-frequency motion are zero-mean oscillatory motions.

- **Second-order wave-induced forces**: wave drift forces are nonzero slowly varying components.

When designing DP systems, it is important to evaluate robustness and performance in the presence of waves [5]. Wave forces can be divided into two components, mean slowly varying component and an oscillatory component. Oscillatory components must be compensated differently by a feedback control system [5].

Mean component can be removed by using integral action while the oscillatory component usually is removed by using a cascaded notch and low-pass filter [5].

There are different wave force models that can be used for prediction. One of them is the State-Space Model for Wave Responses. This model represents wave forces in a simple and effective way and can be used in simulating and testing of feedback control systems.

In this report, wave forces are neglected and not implemented in the control system, but for further and more detailed information, interested readers can see [5].

2.2.4 Thruster and propulsion forces and moment

This section describes how to compute the forces and moment to be generated by each thruster by knowing the overall thruster forces in surge and sway directions and the overall thruster moment in yaw direction.

There are different types of thrusters. Most common types are main propellers, tunnel thrusters, and azimuth thrusters. The main propellers affect the vessel position in surge. Tunnel thrusters affect vessel position in sway and azimuth thrusters can rotate the vessel about z axis and affect the heading of the vessel.
The generalized thruster forces in 3DOF (surge, sway and yaw) expressed in Body frame is given by equation 2.11.

$$\tau = \begin{bmatrix} F_x \\ F_y \\ l_x F_y - l_y F_x \end{bmatrix} \tag{2.11}$$

where: l_x, l_y are thrusters lever arms with respect to the CO (Centre of the vessel).

The thrusters configuration matrix T can be used to calculate the force generated by each thruster in each direction. The T matrix have number of columns equal to number of thrusters and have number of rows equal to degrees of freedom, three rows in dynamic positioning systems. The first row of T configures thrusters forces in surge, the second row configures thrusters forces in sway, and the third row configures thrusters moments in yaw. The generalized force is:

$$\tau = TF \tag{2.12}$$

where:

$$F = Ku \tag{2.13}$$

where:

K: thrust coefficients diagonal matrix.

u: control input vector.

Example:

In a case with two main propellers, one azimuth thruster and two tunnel thrusters, the thruster force τ is given by equation 2.14.

$$\tau = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ -l_1 & -l_2 & 0 & l_3 & l_4 & l_5 \end{bmatrix} \begin{bmatrix} K_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & K_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & K_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & K_5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_{3_x} \\ u_{3_y} \\ u_4 \\ u_5 \end{bmatrix} \tag{2.14}$$

where:

l_1, l_2, l_3, l_4 and l_5 are thrusters lever arms with respect to CO. Figure 2.5 illustrates these lever arms.

K_1, K_2, K_3, K_4 and K_5 are thrusters coefficients.

u_1: control input for the first thruster which is a main propeller.

u_2: control input for the second thruster which is a main propeller.

u_{3_x} and u_{3_y} are the components in x and y of the azimuth thruster control input.

$$u_{3_x} = u_3 \cos(\alpha) \tag{2.15}$$

$$u_{3_y} = u_3 \sin(\alpha) \tag{2.16}$$

$$u_3 = \sqrt{u_{3_x}^2 + u_{3_y}^2} \tag{2.17}$$

$$\alpha_3 = atan2(u_{3_y}, u_{3_x}) \tag{2.18}$$

α_3 is the azimuth thruster angle.

u_4 and u_5 are the control inputs of the tunnel thrusters.

In this report, the calculated optimal control is the generalized thrusters forces in surge and sway and the generalized thrusters moment in yaw.

2.3 Mathematical model

To study the dynamics of marine vessels, Balchen model is used. Balchen divided the model that describes vessel dynamics into two parts. A law-frequency model induced by wind, water current and thrusters forces, and a high-frequency model induced by sea waves forces [1].

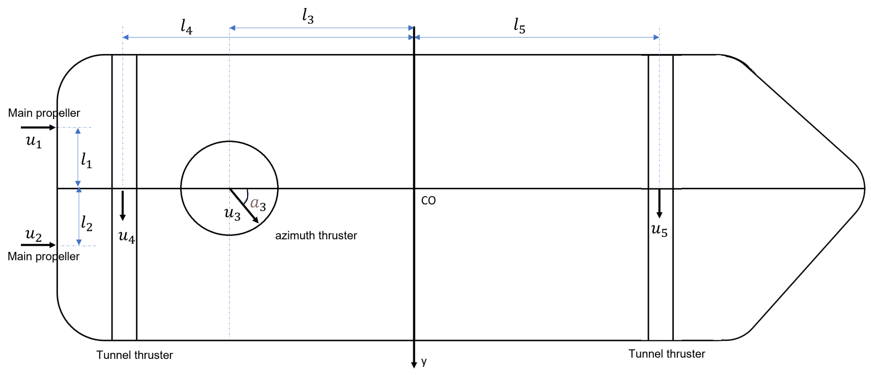


Figure 2.5: Schematic drawing showing the thruster configuration for an example vessel

Balchen model is derived from Newton's second law which states that the acceleration of a body multiplies by its mass equals to the sum of forces acting on it.

$$ma = \sum F \tag{2.19}$$

where:

m: mass of a body $[kg]$.

a: acceleration of the body $[m/s^2]$.

F: forces acting on the body $[N]$.

where in the case of a vessel:

$$F = F_w + F_c + F_t \tag{2.20}$$

where:

F_w: wind force $[N]$.

F_c: water current force $[N]$.

F_t: thruster force $[N]$.

In this report, high-frequency model induced by sea waves is neglected and only the Low-frequency model for Balchen [1], given in equations 2.21 to 2.26, has been implemented and used in the control system.

$$\frac{dx_{su}}{dt} = v_{su} \tag{2.21}$$

$$\frac{dx_{sw}}{dt} = v_{sw} \tag{2.22}$$

$$\frac{d\psi}{dt} = v_{\psi} \tag{2.23}$$

$$\frac{dv_{su}}{dt} = -\frac{d_1}{m_1}|v_{su} - v_{c_{su}}|(v_{su} - v_{c_{su}}) + \frac{1}{m_1}(F_{w_{su}} + F_{t_{su}}) + \eta_1 \tag{2.24}$$

$$\frac{dv_{sw}}{dt} = -\frac{d_2}{m_2}|v_{sw} - v_{c_{sw}}|(v_{sw} - v_{c_{sw}}) + \frac{1}{m_2}(F_{w_{sw}} + F_{t_{sw}}) + \eta_2 \tag{2.25}$$

$$\frac{dv_{\psi}}{dt} = -\frac{d_3}{m_3}|v_{\psi}|v_{\psi} - \frac{d_4}{m_3}|v_{sw} - v_{c_{sw}}|(v_{sw} - v_{c_{sw}}) + \frac{1}{m_3}(N_c + N_w + N_t) + \eta_3 \tag{2.26}$$

where:

x_{su}: vessel position in surge [m].

x_{sw}: vessel position in sway [m].

ψ: vessel heading in yaw [rad].

v_{su}: vessel velocity in surge [m/s].

v_{sw}: vessel velocity in sway [m/s].

v_{ψ}: vessel heading rate [rad/s].

d_1, d_2, d_3, d_4: drag and moment coefficients. These are generally given as functions of the difference between the vessel heading and water current direction [1]. In this report it is assumed that water current has the same direction as the vessel. Therefore, these coefficients are assumed to be constants.

m_1, m_2, m_3: inertial coefficients which are assumed to be constants [1].

η_1, η_2, η_3: are assumed to be zero mean Gaussian white noise processes. In this report, it is assumed that there is no noise affecting the model.

As seen in equations 2.24 to 2.26, implementing the non-linear model needs good knowledge about some parameters related to the vessel dynamics and geometry.

From [1], some values of parameters $d_1, d_2, d_3, d_4, m_1, m_2$ and m_3 are available and they take the values in table 2.3. Wind forces in surge and sway directions and wind moment in yaw can be calculated by using equation 2.2. Those calculations need information about windage area of head wind A_F, windage area of beam wind A_L and over all length L of the studied vessel. In this report, some values are chosen and listed in table 2.3.

Table 2.2: Studied vessel parameters

parameter	value
d_1	5×10^{-5} $[\frac{N}{(m/s)^2}]$
d_2	22×10^{-5} $[\frac{N}{(m/s)^2}]$
d_3	12×10^{11} $[\frac{N.m}{(m/rad)^2}]$
d_4	225×10^{-15} $[\frac{N.m}{(m/s)^2}]$
m_1	2.4×10^7 $[kg]$
m_2	4×10^7 $[kg]$
m_3	4.5×10^{10} $[kg.m^2]$
A_F	500 $[m^2]$
A_L	1100 $[m^2]$
L	73.2 $[m]$

2.4 MPC

MPC refers to Model Predictive Control. So it is based on the availability of a mathematical model to the system. The system model is used to predict future outputs and states which are used to formulate the optimization problem. By solving the optimization problem, optimal control values are found. Only the first computed optimal control is used to update the system state. This procedure is repeated at each time step until the end of the simulation time [10].

There are many advantages of using MPC. One of them is that MPC can handle constraints and process variables simply and more efficient than in PID control [11].

Another advantage of MPC is that cross coupling in multiple input and multiple output (MIMO) systems are taken into consideration in an optimal way. MPC is a simple method for controlling MIMO systems [12].

2.5 Kalman filter

Kalman filter is an effective filter that is used to estimate unmeasured states of a linear or non-linear dynamic systems. It can also remove white and coloured noise from the state estimates. There are different versions of Kalman filter. In this report, predictor-corrector version of the discrete time Kalman filter will be described. The Kalman filter given by equation 2.27 has a predictor part and a corrector part. The predictor part $\bar{x}_{k+1}$ predicts the states from previous states estimates using the vessel linear model as seen in equation 2.28. While the corrector part uses the current time measurements to correct the predicted values and give the minimum variance estimate [5].

$$\hat{x}_{k+1} = \bar{x}_{k+1} + K(y_k - \hat{y}_k) \tag{2.27}$$

$$\bar{x}_{k+1} = A\bar{x}_k + Bu_k + BF_w \tag{2.28}$$

2.6 System identification

Identification is the exercise of developing a mathematical relationship, mathematical model, between the inputs and the outputs of a system based on observed or measured data [13].

Outputs constitute all the measured signals which one wishes to predict. While inputs refer to all variables that influence the outputs. The input set consists of both the inputs that can be manipulated, thrusters forces and moment in DP systems, and those that cannot be adjusted but can be measured, wind forces and moment in DP systems. Figure 2.6 illustrates the principle of system identification.

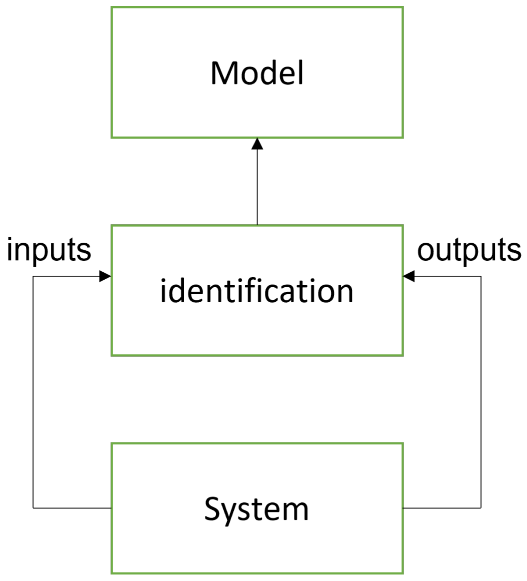


Figure 2.6: system identification illustration

It is not possible to estimate a precise model from finite number of samples data. This is because a single record of data is only one of the several possible data records for the same experiment. It means that repeating an experiment, while fixing values of all controllable factors, generates a different set of readings. This happens because of the randomness in disturbances and measurement noise. So the estimated model is only one among many models that could have been estimated from other possible realizations.

3 Methods

In this chapter, methods used to develop a control subsystem in a DP system are described in details. Firstly, the DP Balchen non-linear model is implemented and simulated in MATLAB to study the behaviour of the selected vessel and how it is affected by different forces. Secondly, the model is linearized to get a simpler version to be implemented and used in MPC control system. MPC control system is developed in four versions. The first one is the full size MPC. While the second one is reduced size MPC. The last two versions are simple MPC and simple MPC with integral action. Finally, system identification method is applied to develop a model-free MPC control system and model-free optimal control with integral action.

3.1 Open loop simulation

Nonlinear law-frequency model of Balchen is implemented in MATLAB. Where the six differential equations 2.21 to 2.26 given in chapter 2 are implemented. MATLAB script for implementing the non linear model is given in appendix B. Simulink is used to see the current time simulation, and how changes in wind speed and direction and changes in thrusters forces affect the vessel position. Implementation in Simulink is shown in figure 3.1 and simulation results are available in chapter 4.

3.2 Linearization

To linearize a non-linear model given by equations 3.1 and 3.2, Taylor series expansion, given by equations 3.3 and 3.4, can be used around an operating point (x_{op}, u_{op}) [14].

$$\dot{x} = f(x,u) \tag{3.1}$$

$$y = g(x,u) \tag{3.2}$$

$$f(x,u) \approx f(x_{op},u_{op}) + \left.\frac{\partial f}{\partial x^T}\right|_{x_{op},u_{op}} (x - x_{op}) + \left.\frac{\partial f}{\partial u^T}\right|_{x_{op},u_{op}} (u - u_{op}) \tag{3.3}$$

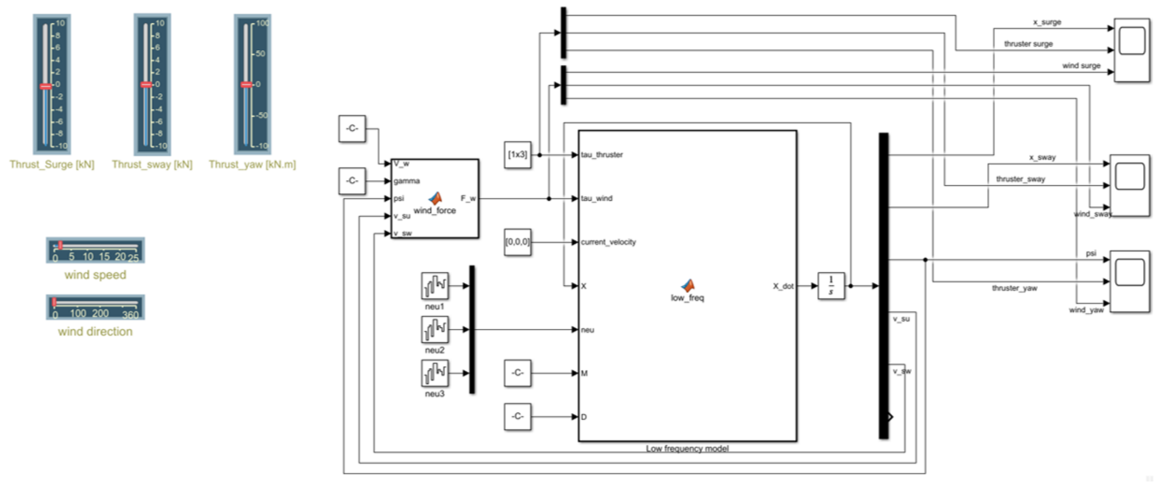


Figure 3.1: Open-loop simulation blocks in Simulink

$$g(x,u) \approx g(x_{op},u_{op}) + \left.\frac{\partial g}{\partial x^T}\right|_{x_{op},u_{op}} (x - x_{op}) + \left.\frac{\partial g}{\partial u^T}\right|_{x_{op},u_{op}} (u - u_{op}) \tag{3.4}$$

By defining the deviation variables in equations 3.5 to 3.7 and continuous time model matrices in equations 3.8 to 3.10, a continuous time linear state space model, described in equations 3.11 and 3.12, is gotten.

$$\delta x = x - x_{op} \tag{3.5}$$

$$\delta u = u - u_{op} \tag{3.6}$$

$$\delta y = y - y_{op} \tag{3.7}$$

$$A_c = \left.\frac{\partial f}{\partial x^T}\right|_{x_{op},u_{op}} \tag{3.8}$$

$$B_c = \left.\frac{\partial f}{\partial u^T}\right|_{x_{op},u_{op}} \tag{3.9}$$

$$D_c = \left.\frac{\partial g}{\partial x^T}\right|_{x_{op},u_{op}} \tag{3.10}$$

$$\delta \dot{x} = A_c \delta x + B_c \delta u \tag{3.11}$$

$$\delta y = D_c \delta x \tag{3.12}$$

As mentioned before, DP system is a MIMO system with three inputs, three outputs and six states. Inputs vector, outputs vector and states vector are given by equations 3.13, 3.14 and 3.15 respectively.

$$u = \begin{bmatrix} F_{t_{su}} \\ F_{t_{sw}} \\ N_t \end{bmatrix} \tag{3.13}$$

$$y = \begin{bmatrix} x_{su} \\ x_{sw} \\ \psi \end{bmatrix} \tag{3.14}$$

$$x = \begin{bmatrix} x_{su} \\ x_{sw} \\ \psi \\ v_{su} \\ v_{sw} \\ v_{\psi} \end{bmatrix} \tag{3.15}$$

Taylor series expansion is used to linearize Balchen model. The six differential equations of states are defined in equations 3.16 to 3.21.

$$f_1 = \frac{dx_{su}}{dt} = v_{su} \tag{3.16}$$

$$f_2 = \frac{dx_{sw}}{dt} = v_{sw} \tag{3.17}$$

$$f_3 = \frac{d\psi}{dt} = v_{\psi} \tag{3.18}$$

$$f_4 = \frac{dv_{su}}{dt} = -\frac{d_1}{m_1}|v_{su} - v_{c_{su}}|(v_{su} - v_{c_{su}}) + \frac{1}{m_1}(F_{w_{su}} + F_{t_{su}}) + \eta_1 \tag{3.19}$$

$$f_5 = \frac{dv_{sw}}{dt} = -\frac{d_2}{m_2}|v_{sw} - v_{c_{sw}}|(v_{sw} - v_{c_{sw}}) + \frac{1}{m_2}(F_{w_{sw}} + F_{t_{sw}}) + \eta_2 \tag{3.20}$$

$$f_6 = \frac{dv_{\psi}}{dt} = -\frac{d_3}{m_3}|v_{\psi}|v_{\psi} - \frac{d_4}{m_3}|v_{sw} - v_{c_{sw}}|(v_{sw} - v_{c_{sw}}) + \frac{1}{m_3}(N_w + N_t + N_c) + \eta_3 \tag{3.21}$$

To calculate the continuous time system matrices A_c, B_c and D_c, equations 3.22 to 3.24 are used.

$$A_c = \begin{bmatrix} \frac{\partial f_1}{\partial x_{su}} & \frac{\partial f_1}{\partial x_{sw}} & \frac{\partial f_1}{\partial \psi} & \frac{\partial f_1}{\partial v_{su}} & \frac{\partial f_1}{\partial v_{sw}} & \frac{\partial f_1}{\partial v_{\psi}} \\ \frac{\partial f_2}{\partial x_{su}} & \frac{\partial f_2}{\partial x_{sw}} & \frac{\partial f_2}{\partial \psi} & \frac{\partial f_2}{\partial v_{su}} & \frac{\partial f_2}{\partial v_{sw}} & \frac{\partial f_2}{\partial v_{\psi}} \\ \frac{\partial f_3}{\partial x_{su}} & \frac{\partial f_3}{\partial x_{sw}} & \frac{\partial f_3}{\partial \psi} & \frac{\partial f_3}{\partial v_{su}} & \frac{\partial f_3}{\partial v_{sw}} & \frac{\partial f_3}{\partial v_{\psi}} \\ \frac{\partial f_4}{\partial x_{su}} & \frac{\partial f_4}{\partial x_{sw}} & \frac{\partial f_4}{\partial \psi} & \frac{\partial f_4}{\partial v_{su}} & \frac{\partial f_4}{\partial v_{sw}} & \frac{\partial f_4}{\partial v_{\psi}} \\ \frac{\partial f_5}{\partial x_{su}} & \frac{\partial f_5}{\partial x_{sw}} & \frac{\partial f_5}{\partial \psi} & \frac{\partial f_5}{\partial v_{su}} & \frac{\partial f_5}{\partial v_{sw}} & \frac{\partial f_5}{\partial v_{\psi}} \\ \frac{\partial f_6}{\partial x_{su}} & \frac{\partial f_6}{\partial x_{sw}} & \frac{\partial f_6}{\partial \psi} & \frac{\partial f_6}{\partial v_{su}} & \frac{\partial f_6}{\partial v_{sw}} & \frac{\partial f_6}{\partial v_{\psi}} \end{bmatrix}_{x_{op},u_{op}} \tag{3.22}$$

$$B_c = \begin{bmatrix} \frac{\partial f_1}{\partial F_{t_{su}}} & \frac{\partial f_1}{\partial F_{t_{sw}}} & \frac{\partial f_1}{\partial N_t} \\ \frac{\partial f_2}{\partial F_{t_{su}}} & \frac{\partial f_2}{\partial F_{t_{sw}}} & \frac{\partial f_2}{\partial N_t} \\ \frac{\partial f_3}{\partial F_{t_{su}}} & \frac{\partial f_3}{\partial F_{t_{sw}}} & \frac{\partial f_3}{\partial N_t} \\ \frac{\partial f_4}{\partial F_{t_{su}}} & \frac{\partial f_4}{\partial F_{t_{sw}}} & \frac{\partial f_4}{\partial N_t} \\ \frac{\partial f_5}{\partial F_{t_{su}}} & \frac{\partial f_5}{\partial F_{t_{sw}}} & \frac{\partial f_5}{\partial N_t} \\ \frac{\partial f_6}{\partial F_{t_{su}}} & \frac{\partial f_6}{\partial F_{t_{sw}}} & \frac{\partial f_6}{\partial N_t} \end{bmatrix}_{x_{op},u_{op}} \tag{3.23}$$

$$D_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.24}$$

D_c is the output matrix that explains the relationship between outputs vector and states vector where the outputs in a DP system are the same as the first three states, position in surge, position in sway and position in yaw respectively.

Linearization is done at steady state. The operating point is chosen where the centre of gravity of the studied vessel is at the origin of the coordinate system. Also, the yaw angle is chosen to be zero at the operating point, and the vessel is at rest, i.e. zero velocities.

$$x_{op} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.25}$$

$$u_{op} = \begin{bmatrix} d_1|v_{c_{su}}|v_{c_{su}} - F_{w_{su}} \\ d_2|v_{c_{sw}}|v_{c_{sw}} - F_{w_{sw}} \\ d_4|v_{c_{sw}}|v_{c_{sw}} - N_w - N_c \end{bmatrix} \tag{3.26}$$

If it is assumed that at the operating point, there is no disturbances i.e. water current velocity is zero and wind force is zero, the control signal at the operating point will be zero as seen in equation 3.27

$$u_{op} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.27}$$

At the operating point (x_{op}, u_{op}) given by equations 3.25 and 3.27, and by adding the term of the effect of the wind force, as known inputs, the following state space model is gotten.

$$\dot{x} = A_c x + B_c u + B_c F_w \tag{3.28}$$

$$y = D_c x \tag{3.29}$$

The continuous time state space model is discretized by using *c2d* function in MATLAB to get a discrete time state space model of the form in equations 3.30 and 3.31.

$$x_{k+1} = Ax_k + Bu_k + BF_{w_k} \tag{3.30}$$

$$y_k = Dx_k \tag{3.31}$$

To compare the behaviour of the linear and non-linear model under wind force disturbances, a MATLAB script available in appendix E is implemented. To let the wind velocity and wind direction changes slowly along the prediction horizon, a MATLAB function *slowly.m* is developed. *slowly* function is available in Appendix F. The simulation results are available in chapter 4.

3.3 Controllability, observability and stability analysis

After linearizing Balchen model of a DP system, controllability, observability and stability of the system are analyzed.

The pair (A,B) is controllable if and only if the rank of the controllability matrix C equals to the system order n_x [15]. The controllability matrix is given by equation 3.32.

$$C = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n_x-1}B \end{bmatrix} \in R^{n_x \times n_x . n_u} \tag{3.32}$$

where:

n_x: is the system order, number of states.

n_u: number of inputs.

The pair (D,A) is observable if and only if the rank of the observability matrix O equals to the system order n_x [15]. The observability matrix is given by equation 3.33.

$$O = \begin{bmatrix} D \\ DA \\ DA^2 \\ \vdots \\ DA^{i-1} \end{bmatrix} \in R^{m.i \times n_x} \tag{3.33}$$

Results of analysing controllability, observability and stability of the system are available in chapter 4.

3.4 Kalman filter

Kalman filter gain is calculated offline before the simulation loop. So steady state Kalman gain is calculated. *dlqe.m* MATLAB function is used to calculate Kalman gain for discrete time systems that have the form in equations 3.34 and 3.35.

$$x_{k+1} = Ax_k + Bu_k + Gw_k \tag{3.34}$$

$$y_k = Dx_k + v_k \tag{3.35}$$

This function takes A, G, D, Q, R as inputs and gives K, P, Z, E as outputs.

Where: w_k: process noise.

v_k: measurement noise.

$Q = E(ww')$: process noise covariance.

$R = E(vv')$: measurement noise covariance.

K: Kalman gain.

P: Riccati solution.

Z: Error covariance.

E: Estimator poles.

3.5 LQ optimal control with integral action

An LQ optimal control with integral action has approximately the same properties as a standard PID-controller [16]. This type of control system can only be used with linear models, so the linearized model given by equations 3.30 and 3.31 is used. But there is always some noise related to the model and to the measurements. Therefore the model given in equations 3.30 and 3.31 can get the form of discrete stochastic model given in equations 3.36 and 3.37.

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.36}$$

$$y_k = Dx_k + v_k \tag{3.37}$$

The linear quadratic problem of a discrete time model for infinite time horizon and with integral action is given by equation 3.38.

$$J_i = \frac{1}{2} \sum_{k=1}^{\infty} (y_k - r)^T Q (y_k - r) + \Delta u_k^T P \Delta u_k \tag{3.38}$$

Where:

r: is the desired vessel position, reference point.

Q and P: are positive weighting matrices that can be chosen by trial and error to get acceptable response.

To solve the linear quadratic problem independent of the process disturbances w_k and measurement noise v_k, the state space model can be written in the deviation form given by equations 3.39 and 3.40.

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k \tag{3.39}$$

$$y_k = y_{k-1} + D\Delta x_k \tag{3.40}$$

As stated by David Di Ruscio[16], the augmented state space model is given by equations 3.41 and 3.42.

$$\begin{bmatrix} \Delta x_{k+1} \\ y_k - r \end{bmatrix} = \begin{bmatrix} A & 0_{n\times m} \\ D & I_{m\times m} \end{bmatrix} \begin{bmatrix} \Delta x_k \\ y_{k-1} - r \end{bmatrix} + \begin{bmatrix} B \\ 0_{m\times r} \end{bmatrix} \Delta u_k \tag{3.41}$$

$$y_k - r = \begin{bmatrix} D & I_{m\times m} \end{bmatrix} \begin{bmatrix} \Delta x_k \\ y_{k-1} - r \end{bmatrix} \tag{3.42}$$

The new format of the model gets the form in equations 3.43 and 3.44

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}\Delta u_k \tag{3.43}$$

$$\tilde{y}_k = \tilde{D}\tilde{x}_k \tag{3.44}$$

By solving the quadratic problem (minimizing the objective function with respect to the control deviation Δu_k), the optimal control deviation is given by equation 3.45.

$$\Delta u_k = \begin{bmatrix} G_1 & G_2 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ y_{k\_1} - r \end{bmatrix} \tag{3.45}$$

The control signal at time index k can be calculated from the previous control signal and two terms, one is related to the state deviation and the second is related to the error (deviation between the output and the reference) [7] as seen in equation 3.46.

$$u_k = u_{k\_1} + G_1\Delta x_k + G_2(y_{k\_1} - r_k) \tag{3.46}$$

The optimal feedback matrix $G = \begin{bmatrix} G_1 & G_2 \end{bmatrix}$ is calculated by using the `dlqdu_ pi` function in MATLAB [17].

MATLAB script that is used to simulate LQ optimal control with integral action is available in appendix K and simulation results are described in chapter 4.

To handle input amplitude constraints, input limitations is applied to the calculated optimal control. MATLAB script that simulate LQ optimal control with integral action with input amplitude constraints is available in appendix L and simulation results are described in chapter 4.

3.6 MPC

In this section, different versions of MPC are described. The standard MPC, reduced size MPC, simple MPC and simple MPC with integral action are discussed.

3.6.1 Standard MPC

In this section, standard MPC is described in details. The objective function in DP systems is to minimize the error between the system output and the reference point and to minimize the control signal as seen in equation 3.47.

$$J_k = \frac{1}{2}\sum_{k=1}^{N}(e_k^T Q_k e_k + u_{k-1}^T P_k u_{k-1}) \tag{3.47}$$

where:

N: the prediction horizon.

Q_k: weighting matrix for error e at time instant k.

P_k: weighting matrix for input u at time instant k.

This optimization problem is subject to equality constraints in equations 3.48 to 3.50.

$$x_{k+1} = Ax_k + Bu_k + BF_{w_k} \tag{3.48}$$

$$y_k = Dx_k \tag{3.49}$$

$$e_k = y_k - r_k \tag{3.50}$$

The optimization problem has to be reformulated to get the standard form in equation 3.51.

$$J = \frac{1}{2}Z^T HZ + C^T Z \tag{3.51}$$

This can be done by defining Z as unknown variables u, x, e and y.

$$J = \frac{1}{2}\begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix}^T \begin{bmatrix} H_{11} & 0 & 0 & 0 \\ 0 & H_{22} & 0 & 0 \\ 0 & 0 & H_{33} & 0 \\ 0 & 0 & 0 & H_{44} \end{bmatrix} \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}^T \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} \tag{3.52}$$

By comparing equations 3.47 and 3.52, and by assuming that weighting matrices Q and P are constants along the prediction horizon, matrices H_{11}, H_{22}, H_{33} and H_{44} are given in equations 3.53 to 3.56. In addition constant C is gotten and it is found and equals to zeros as in equation 3.57.

$$H_{11} = \begin{bmatrix} P & 0 & \cdots & 0 \\ 0 & P & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P \end{bmatrix}_{N \times N} \tag{3.53}$$

$$H_{22} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_x \times N.n_x} \tag{3.54}$$

$$H_{33} = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q \end{bmatrix}_{N \times N} \tag{3.55}$$

$$H_{44} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_y \times N.n_y} \tag{3.56}$$

$$C = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{N.nz \times 1} \tag{3.57}$$

where:

nz: number of unknown variables. $n_z = n_u + n_x + n_y + n_y$

Interested readers can find more details in [10].

Also equality constraints have to be rearranged to get the standard form shown in equation 3.58.

$$A_{eq}Z = B_{eq} \tag{3.58}$$

where:

A_{eq}: unknown part of equality constraints.

B_{eq}: known part of equality constraints.

Equation 3.58 can be written in more detailed form as in equation 3.59.

$$\begin{bmatrix} A_{eq1_u} & A_{eq1_x} & A_{eq1_e} & A_{eq1_y} \\ A_{eq2_u} & A_{eq2_x} & A_{eq2_e} & A_{eq2_y} \\ A_{eq3_u} & A_{eq3_x} & A_{eq3_e} & A_{eq3_y} \end{bmatrix} \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} = \begin{bmatrix} B_{eq1} \\ B_{eq2} \\ B_{eq3} \end{bmatrix} \tag{3.59}$$

The first row in matrix A_{eq} gotten from the first equality constraint given in equation 3.48. While the second row is gotten from the second equality constraint given in equation 3.49 and the third row is gotten from the third and last equality constraint in equation 3.50.

By comparing the equality constraints given by equations 3.48 to 3.50 and the standard form of equality constraint in equation 3.59, it is found that the matrices in the block matrix A_{eq} are given by equations 3.60 to 3.71, and matrices in B_{eq} are given by equations 3.72 to 3.74.

$$A_{eq1_u} = \begin{bmatrix} -B & 0 & 0 & \cdots & 0 \\ 0 & -B & 0 & \cdots & 0 \\ 0 & 0 & -B & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -B \end{bmatrix} \tag{3.60}$$

$$A_{eq1_x} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ -A & I & 0 & \cdots & 0 \\ 0 & -A & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -A & I \end{bmatrix} \tag{3.61}$$

$$A_{eq1_e} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_x \times N.n_y} \tag{3.62}$$

$$A_{eq1_y} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_x \times N.n_y} \tag{3.63}$$

$$A_{eq2_u} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_y \times N.n_u} \tag{3.64}$$

$$A_{eq2_x} = \begin{bmatrix} -D & 0 & 0 & \cdots & 0 \\ 0 & -D & 0 & \cdots & 0 \\ 0 & 0 & -D & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -D \end{bmatrix} \tag{3.65}$$

$$A_{eq2_e} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_y \times N.n_y} \tag{3.66}$$

$$A_{eq2_y} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \end{bmatrix} \tag{3.67}$$

$$A_{eq3_u} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_y \times N.n_u} \tag{3.68}$$

$$A_{eq3_x} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N.n_y \times N.n_x} \tag{3.69}$$

$$A_{eq3_e} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \end{bmatrix} \tag{3.70}$$

$$A_{eq3_y} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \end{bmatrix} \tag{3.71}$$

$$B_{eq1} = \begin{bmatrix} Ax_0 + BF_{w_1} \\ BF_{w_2} \\ BF_{w_3} \\ \vdots \\ BF_{w_N} \end{bmatrix} \tag{3.72}$$

$$B_{eq2} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.73}$$

$$B_{eq3} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix} \tag{3.74}$$

For more details about forming matrices A_{eq} and B_{eq}, interested readers can see [10].

After formulating the optimization problem in the standard form, *quadprog.m* function in MATLAB is used to solve it.

The MATLAB script that is used to implement the standard MPC is available in Appendix G.

The standard MPC is simulated for 15 minutes with a time step of 10 seconds. Prediction horizon is 10 steps forward. Wind and water current disturbances are simulated using functions in Appendices C and D and assumed to be known disturbances along the the prediction horizon. In addition, the set-points along the prediction horizon are known and are fed to the MPC control system. The linearized model of Balchen is used in the linear MPC algorithm to calculate the optimal control values. Only the first optimal control value which is calculated at each time step is used to be fed to the system which is represented by the non-linear model of Balchen. The MATLAB script that is developed to do this simulation is available in appendix H, and the simulation results are described in chapter 4.

3.6.2 Reduced size MPC

After implementing the standard MPC with ($n_z = n_u + n_x + n_y + n_y$) unknown variables, the simulation time was very long due to unnecessary calculations since many of the matrices have zeros elements. Therefore, the optimization problem size has been reduced in this section. Where two of the equality constraints have been substituted in the objective function, the output equality constraint and the error equality constraint.

e_k in equation 3.47 has been substituted by equation 3.50 as seen in equation 3.75.

$$J_k = \frac{1}{2}\sum_{k=1}^{N}((y_k - r_k)^T Q_k (y_k - r_k) + u_{k-1}^T P_k u_{k-1}) \tag{3.75}$$

By rearranging equation 3.75 and by knowing that $Q_k = Q_k^T$ since Q is a symmetric matrix, equation 3.76 is gotten.

$$J_k = \frac{1}{2}\sum_{k=1}^{N}(y_k^T Q_k y_k + u_{k-1}^T P_k u_{k-1}) - 2(Q_k r_k)^T y_k + r_k^T Q_k r_k \tag{3.76}$$

Since $r_k^T Q_k r_k$ is a constant term, so it can be removed from the objective function in equation 3.76.

y_k in equation 3.76 is substituted by output equality constraint in equation 3.49. Then the objective function gets the form in equation 3.77.

$$J_k = \frac{1}{2}\sum_{k=1}^{N}(x_k^T \tilde{Q}_k x_k + u_{k-1}^T P_k u_{k-1}) - 2(D^T Q_k r_k)^T x_k \tag{3.77}$$

Where:

$\tilde{Q} = D^T Q D$

To use *quadprog* solver in MATLAB, equation 3.77 has to be formulated in the standard form given by equation 3.51.

where:

$$J = \begin{bmatrix} u_{k-1} \\ x_k \end{bmatrix}^T = \begin{bmatrix} H_{11} & 0 \\ 0 & H_{22} \end{bmatrix} \begin{bmatrix} u_{k-1} \\ x_k \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ x_k \end{bmatrix} \tag{3.78}$$

By comparing equation 3.78 with equation 3.77, it is found that H_{11}, H_{22}, C_1 and C_2 are given by equations 3.79 to 3.82 respectively.

$$H_{11} = \begin{bmatrix} P & 0 & \cdots & 0 \\ 0 & P & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P \end{bmatrix}_{N \times N} \tag{3.79}$$

$$H_{22} = \begin{bmatrix} \tilde{Q} & 0 & \cdots & 0 \\ 0 & \tilde{Q} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{Q} \end{bmatrix}_{N \times N} \tag{3.80}$$

$$C_1 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{N.n_u \times 1} \tag{3.81}$$

$$C_2 = \begin{bmatrix} D^T Q_1 r_1 \\ D^T Q_2 r_2 \\ \vdots \\ D^T Q_N r_N \end{bmatrix}_{N \times 1} \tag{3.82}$$

After reformulating the objective function, the equality constraints should be also reformulated to get the standard form given by equation 3.58. In the reduced size MPC, there is only one equality constraint given by equation 3.48.

So the standard form of the equality constraint is given by equation 3.83

$$\begin{bmatrix} A_{eq_u} & A_{eq_x} \end{bmatrix} \begin{bmatrix} u_{k-1} \\ x_k \end{bmatrix} = B_{eq} \tag{3.83}$$

By comparing equation 3.48 with equation 3.83, it is found that A_{eq_u}, A_{eq_x} and B_{eq} are given by equations 3.84 to 3.86 respectively.

$$A_{eq_u} = \begin{bmatrix} -B & 0 & 0 & \cdots & 0 \\ 0 & -B & 0 & \cdots & 0 \\ 0 & 0 & -B & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -B \end{bmatrix} \tag{3.84}$$

$$A_{eq_x} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ -A & I & 0 & \cdots & 0 \\ 0 & -A & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -A & I \end{bmatrix} \tag{3.85}$$

$$B_{eq} = \begin{bmatrix} Ax_0 + BF_{w_1} \\ BF_{w_2} \\ BF_{w_3} \\ \vdots \\ BF_{w_N} \end{bmatrix}_{N.n_x \times 1} \tag{3.86}$$

The reduced size MPC is simulated for 15 minutes with a time step of 10 seconds. Prediction horizon is 10 steps forward. Wind and water current disturbances are simulated using functions in Appendices C and D and assumed to be known disturbances along the

the prediction horizon. In addition, the set-points along the prediction horizon are known and are fed to the MPC control system. The linearized model of Balchen is used in the reduced size linear MPC to calculate the optimal control values. Only the first optimal control value which is calculated at each time step is used and fed to the system which is represented by the non-linear model of Balchen. The MATLAB script that is used to develop the reduced size MPC algorithm is available in appendix I. While the MATLAB script that is developed to do the simulation is available in appendix J, and the simulation results are described in chapter 4.

3.6.3 Simple MPC

In the previous section, the size of the quadratic problem in the standard MPC method in section 3.6.1 has been reduced from $N \times (n_u + n_x + n_y + n_y)$ to $N \times (n_u + n_x)$ in the reduced size MPC in section 3.6.2. A more simple method is developed by David Di Ruscio [11], where a linear state space model given by equations 3.87 and 3.88 can always be written as a prediction model that takes the standard form in equation 3.89. In this method, QP size is reduced to $N \times n_u$.

$$x_{k+1} = Ax_k + Bu_k \tag{3.87}$$

$$y_k = Dx_k \tag{3.88}$$

$$y_{k+1/L} = F_L u_{k/L} + p_L \tag{3.89}$$

Where:

L: is prediction horizon.

F_L: a constant matrix derived from the process model and given in equation 3.90.

p_L: is a vector that is dependent of a number of inputs and outputs older than time instant k as well as the model parameters and given by equation 3.91.

$$F_L = \begin{bmatrix} O_L B & H_L^d \end{bmatrix} \tag{3.90}$$

where:

O_L: is the extended observability matrix of the pair A, D.

H_L^d: is the Toepliz matrix of impulse response matrices $DA^{i-1}B$.

$O_L, O_L B$ and H_L^d can be calculated by using *ss2h.m* MATLAB function that is developed by David Di Ruscio [18].

$$p_L = \begin{bmatrix} O_L A x_k \end{bmatrix} \tag{3.91}$$

The objective function to be minimized is given by equation 3.92.

$$J_k = (y_{k+1/L} - r_{k+1/L})^T Q (y_{k+1/L} - r_{k+1/L}) + u_{k/L}^T P u_{k/L} \tag{3.92}$$

By substituting $y_{k+1/L}$ in the objective function by its value in equation 3.89 and eliminating constants terms, equation 3.93 is gotten.

$$J_k = u_{k/L}^T(F_L^T QF_L + P)u_{k/L} + 2(F_L^T Q(p_L - r_{k+1/L}))u_{k/L} \tag{3.93}$$

By comparing equation 3.93 with the standard form of the objective function given in equation 3.94, it is found that H and C can be calculated by using equations 3.95 and 3.96 respectively.

$$J_k = u_{k/L}^T H u_{k/L} + C^T u_{k/L} \tag{3.94}$$

$$H = F_L^T QF_L + P \tag{3.95}$$

$$C = 2(F_L^T Q(p_L - r_{k+1/L}) \tag{3.96}$$

Then *quadprog.m* function in MATLAB is used to solve the optimization problem and find the control optimal values along the prediction horizon. To achieve the sliding horizon strategy, only the first calculated optimal control is used to update the system.

The MATLAB script that is used to implement the simple MPC method is available in appendix M, and the simulation results are available in chapter 4.

3.6.4 Simple MPC with integral action

The same method that is used in simple MPC, is used in simple MPC with integral action, but here the SSM in deviation form given by equations 3.39 and 3.40 is used. Therefore, the objective function to be minimized contains the control deviation Δu_{k_L} as shown in equation 3.97.

$$J_k = \Delta u_{k/L}^T H \Delta u_{k/L} + C^T \Delta u_{k/L} \tag{3.97}$$

where:

$$H = F_L^T QF_L + P \tag{3.98}$$

and:

$$C = 2(F_L^T Q(p_L - r_{k+1/L}) \tag{3.99}$$

$\tilde{x_k}$ is used to calculate p_L instead of x_k in equation 3.91. Where $\tilde{x_k} = \begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix}$ and therefore the optimal increment in control input can be calculated using the *quadprog.m* function. Then the optimal control at time instant k is calculated by using equation 3.100.

$$u_k = u_{k-1} + \Delta u_k \tag{3.100}$$

The *quadprog.m* function can handle constraints efficiently as input amplitude constraints and the input rate of change constraints. In DP systems, the control inputs are the thrusters forces and moment. Therefore, there are minimum and maximum limits related to the number of thrusters that are used, type of them, and the maximum power that can be produced by each of them. The input amplitude constraints can be set as bounds as in equation 3.101 or it can be reformulated to be as inequality constraints as in equations 3.102 and 3.103. It is assumed that there is no input rate of change constraints for simplicity.

$$u_{min} \leq u \leq u_{max} \tag{3.101}$$

$$u \leq u_{max} \tag{3.102}$$

$$-u \leq -u_{min} \tag{3.103}$$

u_k can be written in a relationship with Δu_k as in equation 3.104 [12].

$$u_k = S\Delta u_k + cu_{k-1} \tag{3.104}$$

where: $S \in R^{n_u.L\times n_u.L}$ and $c \in R^{n_u.L\times n_u}$ are matrices with ones and zeroes as shown in equations 3.105 and 3.106. Those matrices can be calculated by using *scmat.m* function in MATLAB, developed by David Di Ruscio [19].

$$S = \begin{bmatrix} I_r & 0_r & \cdots & 0_r \\ I_r & I_r & \cdots & 0_r \\ \vdots & \vdots & \ddots & \vdots \\ I_r & I_r & \cdots & I_r \end{bmatrix} \tag{3.105}$$

$$c = \begin{bmatrix} I_r \\ I_r \\ \vdots \\ I_r \end{bmatrix} \tag{3.106}$$

The variable to be optimized is the control increment Δu. Therefore, equation 3.104 have to be substituted in equations 3.102 and 3.103 to get the desired inequality constraints shown in equations 3.107 and 3.108.

$$S\Delta u \leq u_{max} - cu_{k-1} \tag{3.107}$$

$$-S\Delta u \leq -u_{min} + cu_{k-1} \tag{3.108}$$

Inequality constraints available in equations 3.107 and 3.108 have to be reformulated to take the standard form in equation 3.109.

$$A_i\Delta u \leq B_i \tag{3.109}$$

where:

$$A_i = \begin{bmatrix} S \\ -S \end{bmatrix} \tag{3.110}$$

and:

$$B_i = \begin{bmatrix} u_{max} - cu_{k-1} \\ -u_{min} + cu_{k-1} \end{bmatrix} \tag{3.111}$$

A_i and B_i can be used directly in *quadprog.m* function. MATLAB script that is used to simulate simple MPC with integral action and input amplitude constraints is available in appendix N and simulation results are discussed in chapter 4.

3.7 System identification

This chapter describes how inputs values are generated and applied to the system to get the outputs data, and how to use *dsr _e.m* function to get the matrices of the identified model.

3.7.1 Experiments to collect data

Some experiments have been done in MATLAB to collect input, output data-set to use it in system identification method. Input values have been gotten by using LQ optimal control with integral action method and by applying step change using *prbs1.m* function [20] to get good values for the experiments. Figure 3.2 shows the input-output data gotten from the experiment to be used in system identification method.

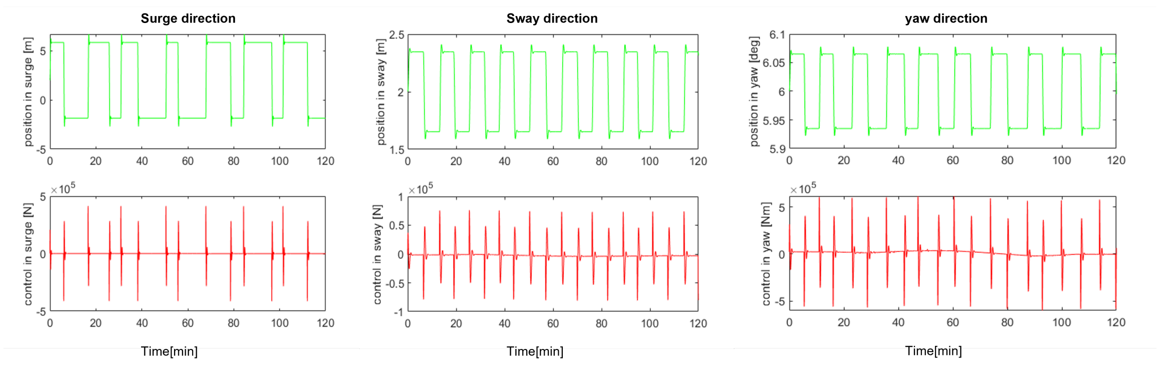


Figure 3.2: Input-output data gotten from the experiment, where the upper plot shows the simulated output data in surge, sway and yaw and the lower plot shows the input data in surge, sway and yaw

Input values have been applied to the non-linear model of Balchen to get the output values. Two sets of data, each of 7200 samples have been collected. One set to be used in *dsr _e.m* function to get an identified state space model, and the other one is used for validation of the model.

Data generation and acquisition is considered as the most important step in system identification since the confidence of the final model depends on the quality of the collected data [13].

The MATLAB script that is used to do the experiments and collect input-output data is available in appendix O.

3.7.2 The identified model

To identify the linear state space model, the system identification algorithm as Deterministic and Stochastic System Identification and Realization "DSR" is used. The *dsr _ e.m* function is used. It takes Y,U,L,g,J and n as inputs. Where Y is output data set, U is input data set, L is the identification-horizon and J is the horizon (into the past) used to define instruments from the data which is used to remove noise [21], and g is 0 in closed loop systems.

The *dsr _ e.m* function estimates the the matrices (A,B,D,E,K,F) in the discrete time combined deterministic and stochastic dynamic model on innovations form and the initial state x_0. The identified model and the results of using it in the LQ optimal control with integral action and MPC control are available in chapter 4.

4 Results

This chapter shows the results of open-loop simulation, comparing linear and non-linear models, stability, controllability and observability analysis, comparing Kalman filter states estimates with states from the non-linear model, MPC different methods simulations and optimal control with integral action. At last the results of using system identification to get a suitable identified system and develop a model-free MPC and model-free optimal control with integral action are dressed.

4.1 Open loop simulation

This section shows an open-loop simulation in *Simulink* by using manual control. This simulation is done to study the effect of changes in wind force and control signal on the vessel position in surge, sway and yaw.

Figure 4.1 has three plots. The upper one shows the surge position, the middle one shows the thruster force in surge that is changing manually here. While the lower plot shows the wind force that is changing according to changes in wind relative speed and wind direction as explained in chapter 2.

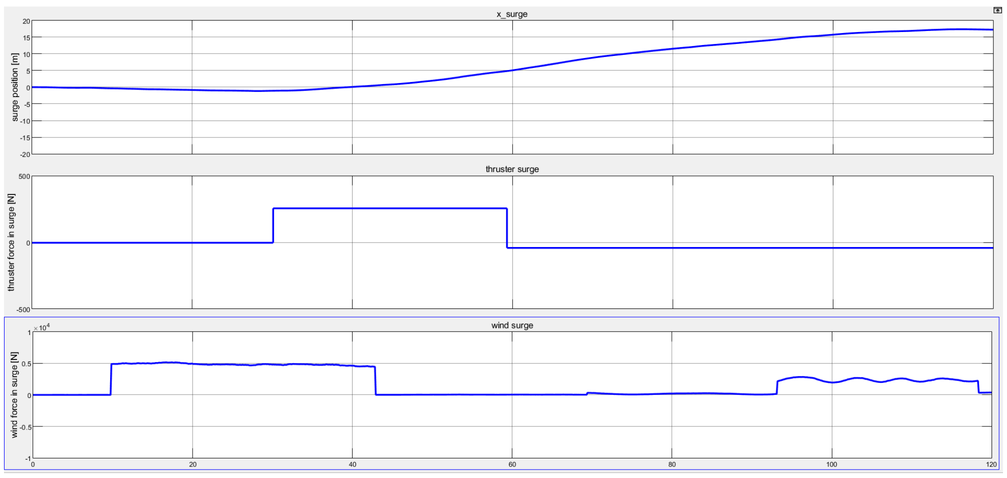


Figure 4.1: open-loop simulation in surge direction

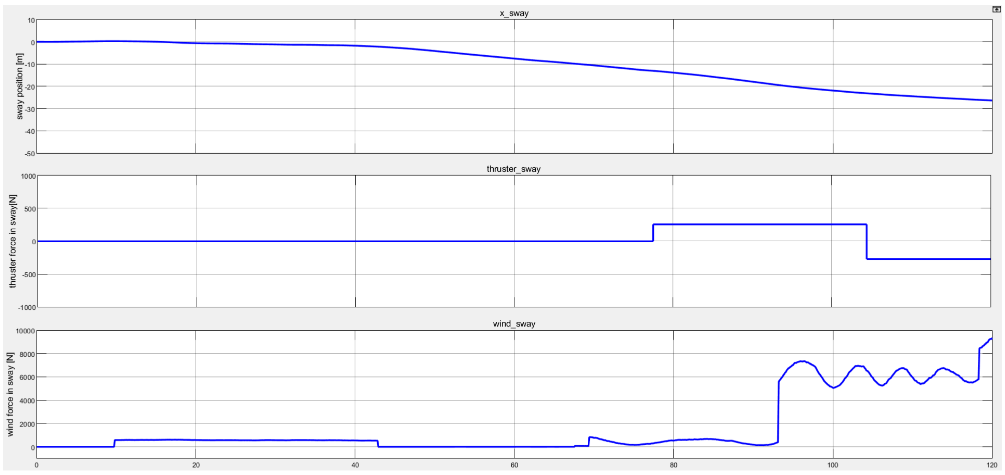


Figure 4.2: open-loop simulation in sway direction

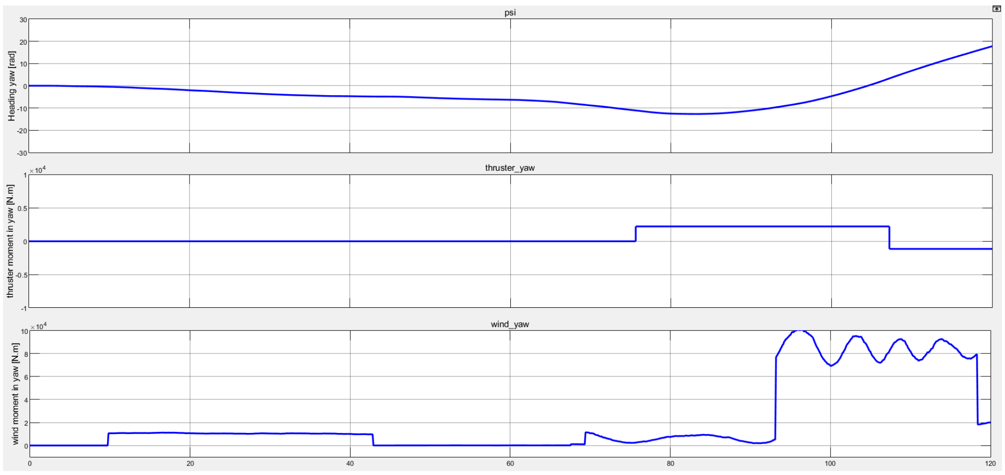


Figure 4.3: open-loop simulation in yaw direction

By studying figure 4.1, its is found that increasing in wind force causes decreasing in surge position and vice versa. While increasing in thruster force in surge increasing the vessel surge position.

By studying vessel motion in sway and yaw directions by looking at figures 4.2 and 4.3, it is found that the effect of the wind is as the same as its effect in surge direction where increasing in wind force in sway and wind moment in yaw leads to decreasing in vessel position in sway and vessel heading respectively and vice versa.

As a result, when wind forces and moment increases, thruster forces and moment have to be increased in order to keep the vessel in its position.

4.2 Linearization

This section shows the results of linearizing the low-frequency part of the non-linear model of Balchen and then compares the behaviour of the non-linear model of a vessel and the linearized SSM under wind force disturbances. Where wind speed takes random values between 0 and 20 $[m/s]$ along the simulation, and wind direction takes random values between 0 and 360 $[deg]$.

By using Taylor expansion as described in chapter 3, linear continuous time state space model matrices A_c, B_c and D_c are gotten as shown in equations 4.1 to 4.3

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -2\frac{d_1}{m_1}|v_{c_{su}}| & 0 & 0 \\ 0 & 0 & 0 & 0 & -2\frac{d_2}{m_2}|v_{c_{sw}}| & 0 \\ 0 & 0 & 0 & 0 & -2\frac{d_4}{m_3}|v_{c_{sw}}| & 0 \end{bmatrix} \tag{4.1}$$

$$B_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_1} & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 \\ 0 & 0 & \frac{1}{m_3} \end{bmatrix} \tag{4.2}$$

$$D_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.3}$$

Figure 4.4 illustrates how the vessel position in surge changes by the effect of wind force in surge. Where the first subplot shows a comparison between the linear and non-linear model and it is obvious that they fit each other perfectly.

This simulation is under no control signal, therefore the control signal is zero along the simulation. The third subplot shows wind force in surge direction that is calculated by using $wind-force.m$ function available in appendix C. The wind force is proportional to wind velocity and wind direction which are seen in subplots 4 and 5 respectively.

Wind velocity is approximately $20[m/s]$ at the first quarter of the simulation time, whereas the wind direction is approximately $270[deg]$ and increases to $290[deg]$ between the minute 2.5 and 4, and therefore the wind force is almost zero in surge direction and not affected the vessel position in surge until the minute 2.5. After $5[min]$ of starting the simulation, the wind velocity decreased to about 19 $[m/s]$ where the wind direction takes the value

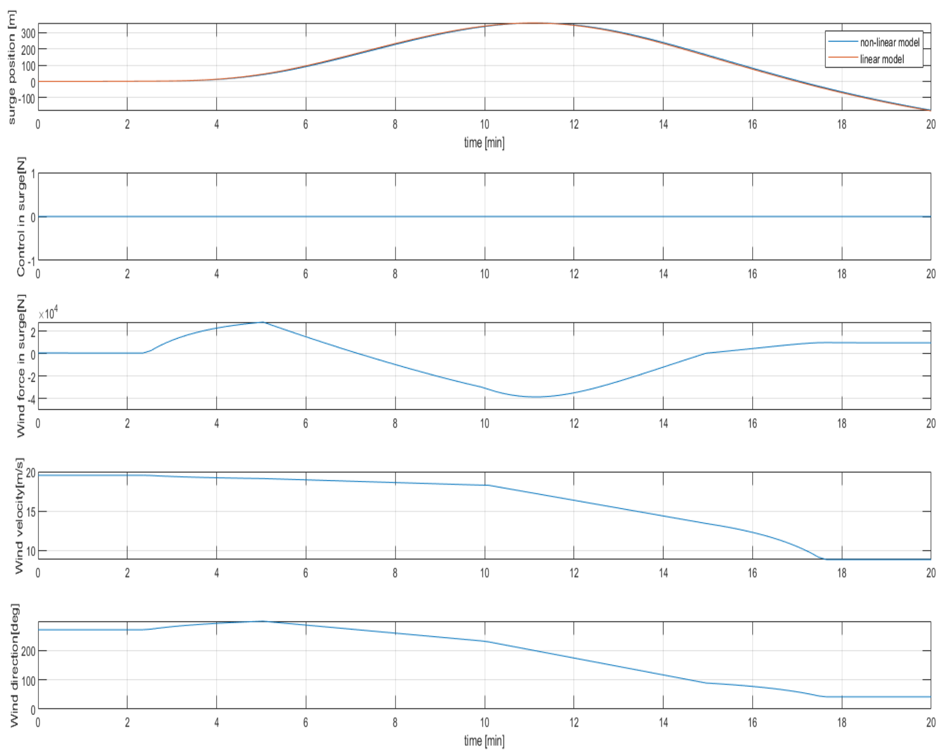


Figure 4.4: Comparing linear and non-linear model in surge direction

$300[deg]$. Therefore, the wind force in surge increases to about $5 \times 10^4[N]$. Then it decreases again causing increasing in vessel position in surge.

Figure 4.5 shows how the vessel position changes in sway direction by the effect of wind force in sway. Where the first subplot shows a comparison between the linear and non-linear model and it is obvious that they fit each other perfectly.

As it is shown in subplot 3 in figure 4.5, wind force in sway is $-20 \times 10^4[N]$ and then begin to increase due changes in wind direction. Where when the wind direction is about $270[deg]$ or $90[deg]$, the wind force has the maximum effect in sway direction as explained in chapter 3. This increasing in wind force causes the vessel position in sway to decrease to reach $-2155[m]$ after $20[min]$.

Figure 4.6 shows how the vessel position in yaw direction is changing and how the wind moment in yaw affects it. The first subplot shows a comparison between the linear and non-linear model and they fit each other perfectly. Subplot 3 shows changes in wind moment in yaw direction. It begins at $0[Nm]$ and begin to decrease at time $2.35[min]$ until it reaches $-1.3^6[Nm]$ at $4.5[min]$. This change in wind moment is due to increment in

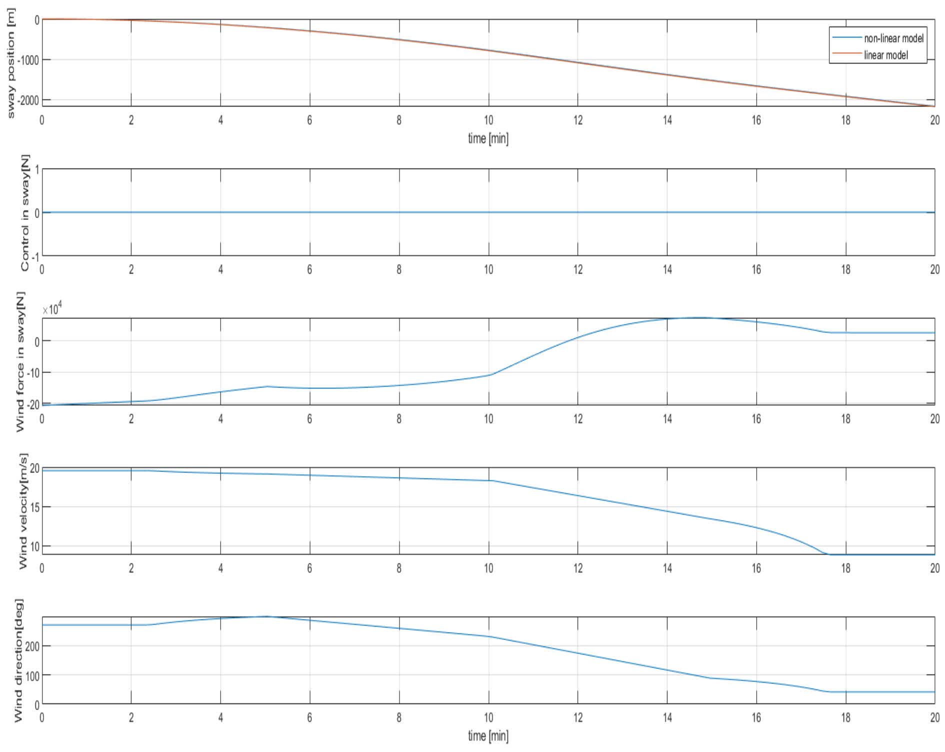


Figure 4.5: Comparing linear and non-linear model in sway direction

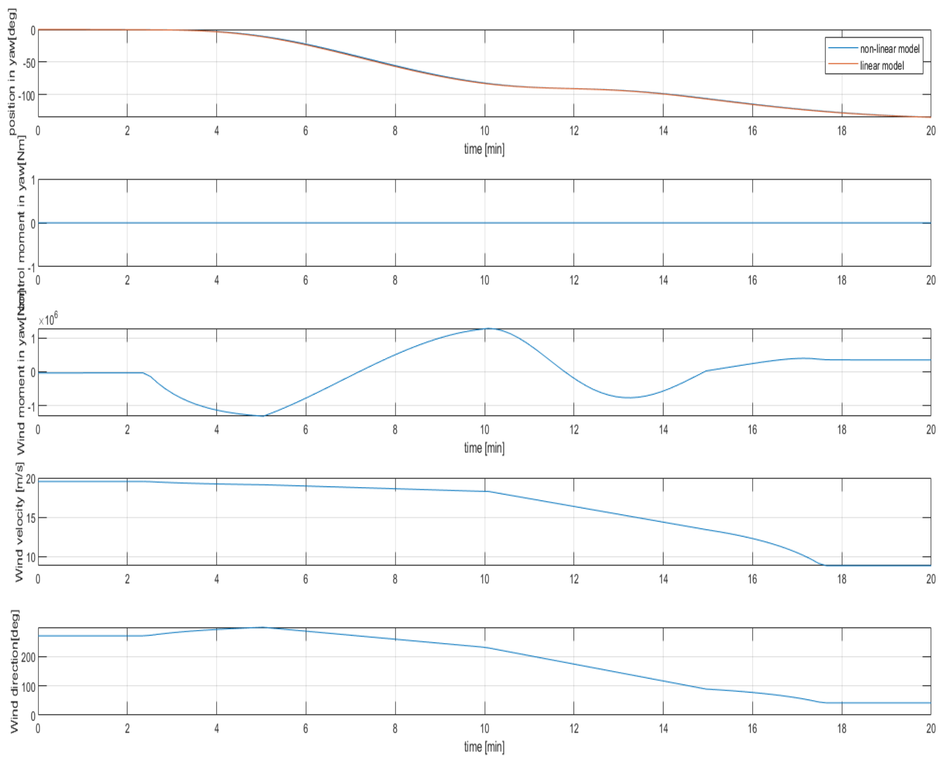


Figure 4.6: Comparing linear and non-linear model in yaw direction

wind direction from 270$[deg]$ to 300$[deg]$. Changes in wind moment causes deviation in yaw direction shown in the first subplot of figure 4.6.

4.3 Stability, Controllability and observability analysis

This section shows the results of analysing stability, controllability and observability of the linearized system given by equations 3.30 and 3.31.

As mentioned in chapter 3, to analyze the stability of the system, eigenvalues of the system matrix A of the discrete time model should be located inside the unity circle. Equation 4.4 shows the eigenvalues of matrix A calculated in MATLAB by using $eig.m$ function and the absolute value of all eigenvalues is less than one. It means that the system is stable.

$$eig(A) = \begin{bmatrix} 1.000 \\ 1.000 \\ 1.000 \\ 0.999 \\ 1.000 \\ 0.999 \end{bmatrix} \tag{4.4}$$

To study the controllability of the system, controllabity matrix of the pair (A, B) is calculated in MATLAB by using *ctrb.m* function and its rank is calculated by using *rank.m* function. The rank of the controllability matrix is 6 and it is equal to the system order. It means that the system is controllable.

To calculate the observability matrix of the pair (A, D), *obsv.m* function in MATLAB is used. In addition, *rank.m* function is used to calculate the rank of the observability matrix. The rank of the observability matrix is 6 and it is equal to the system order. The system is observable.

4.4 Kalman filter

As mentioned in chapter 3, Kalman filter is used to estimate immeasurable states v_{su}, v_{sw} and v_{ψ} to be fed to the MPC controller as initial states and used to calculate optimal control values.

Before adding Kalman filter to the control system, it has been tested to check its behaviour and how the estimated states differs from that ones from the model. This test makes it possible to adjust the parameters that are used to calculate Kalman gain and reduce the error between the estimated states and states from the model.

Figure 4.7 shows the two states in surge direction, surge position [m] in the upper plot and surge velocity [m/sec] in the lower plot. The estimated surge position fits the surge position from the model excepting a small deviation between 6 and 8 [min] when the vessel changes its position due to control forces and moment to achieve the desired set-point.

The estimated surge velocity fits the vessel velocity from the model perfectly along the whole simulation time.

Figure 4.8 shows a comparison between the estimated states in sway direction and the corresponding states from the model. The upper plot shows the position in sway [m], where the lower one shows the vessel velocity in sway direction [m/s]. The estimated sway position fits the sway position from the model excepting a small deviation when the vessel position in sway changes from 2 to 6 [m]. While the estimated sway velocity fits the one calculated from the model perfectly.

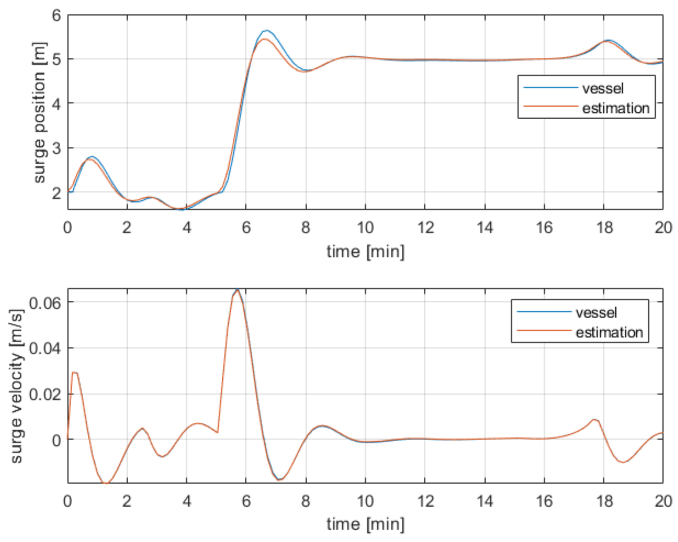


Figure 4.7: Comparison between position estimate in surge and the vessel position in surge from the non-linear model in the upper plot, and a comparison between vessel velocity estimate in surge and vessel velocity in surge from the non-linear model in the lower plot

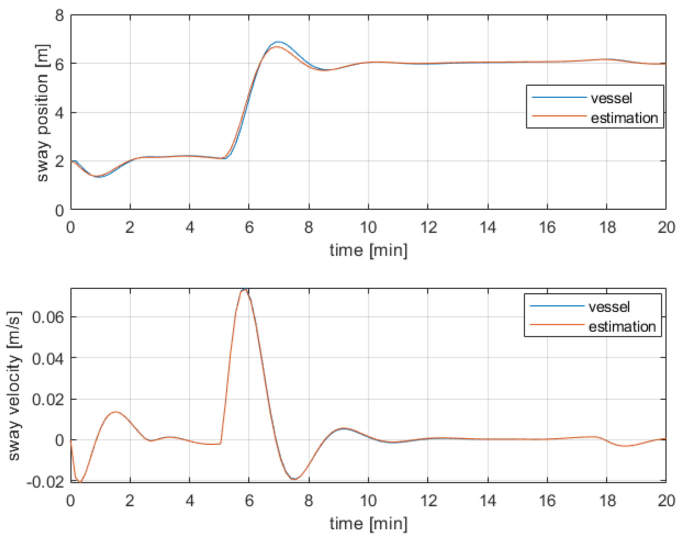


Figure 4.8: Comparison between position estimate in sway and the vessel position in sway from the non-linear model in the upper plot, and a comparison between vessel velocity estimate in sway and vessel velocity in sway from the non-linear model in the lower plot

Figure 4.9 shows the vessel heading (position in yaw) in [deg] in the first plot, and vessel heading rate [rad/s] in the second plot. The estimated heading fits the calculated heading excepting when the vessel heading changes from 12 to 14 [deg]. While the estimated vessel

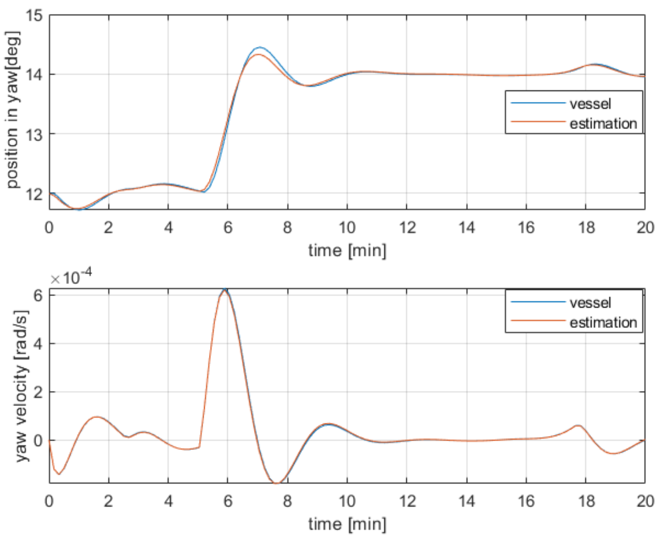


Figure 4.9: Comparison between vessel heading estimate and the vessel heading from the non-linear model in the upper plot, and a comparison between vessel heading rate estimate and vessel heading rate from the non-linear model in the lower plot

heading rate fits the calculated one perfectly.

4.5 MPC

This section shows the results of running the simulations of implemented standard MPC, reduced size MPC, simple MPC and simple MPC with integral action.

4.5.1 Standard MPC

This section shows the results of simulating standard MPC described in section 3.6.1.

Two simulations have been done. The first one is by assuming that all states are measurable, and the second one by using a Kalman filter to estimate immeasurable system states v_{su}, v_{sw} and v_{ψ}.

Simulation by assuming measurable states

The set-point to the surge position of the studied vessel is chosen to be $0[m]$ from the origin of the Body coordinate system at the first $5[min]$ of the simulation period. The set-point in surge changes two times during the simulation period to be $4[m]$ from $5[min]$ to $10[min]$ and then is set to go back to $3[m]$ from $10[min]$ and to the end of the simulation.

Figure 4.10 shows that at time $2.3[min]$ wind force began to decrease which causes the vessel position in surge to begin to increase, but thrusters force in surge increases to compensate wind force and keep the vessel in its position.

At time $5[min]$, the set-point in surge increases and therefore, the control system increases thruster force in surge to about $10 \times 10^3[N]$. That causes increasing in vessel position in surge to about $5[m]$. the control system has detected that overshoot and the thrusters force begin to decrease until it reaches $-4 \times 10^4[N]$. It takes about $2[min]$ until the vessel position in surge reaches the desired set-point.

Wind force in surge is changed slowly from time $5[min]$ to $10[min]$. The control system response to this decreasing by slowly increasing in thrusters force in surge direction. This increasing can be seen between 6.8 [min] and 9.5 [min]. Before and after that time, control system was responding to changes in set-point. As a result, the control system is able to track the set-point under wind disturbances in surge direction.

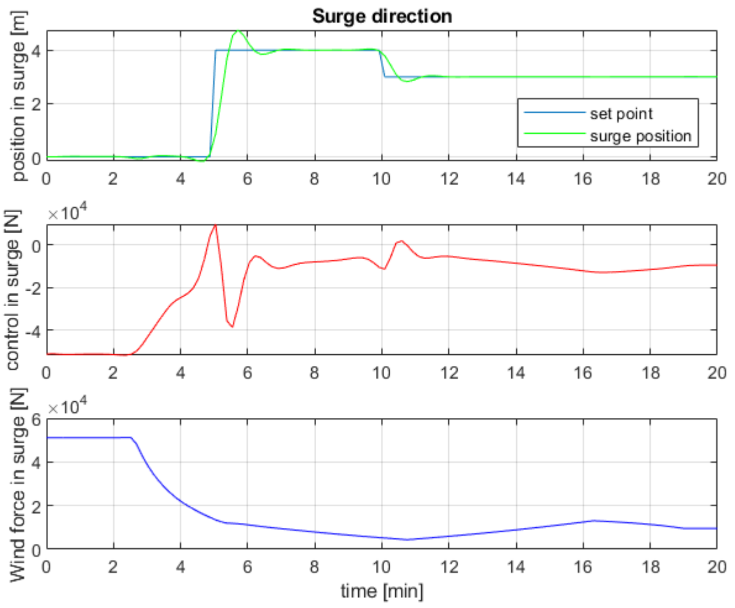


Figure 4.10: Position in Surge in the upper subplot, control force in surge direction in the middle subplot and the wind force in surge direction in the lower subplot

In sway direction simulation, the set-point changes two times, at time 5[min] and 10 [min]. Where the vessel position in sway begins at the origin, 0 [m], and increases to 4 [m] after 5 [min]. Then it increases gain to 8[m] after 10 [min]. The first plot of figure 4.11 shows this changes in set-point along the simulation period. Where there is a small overshoot

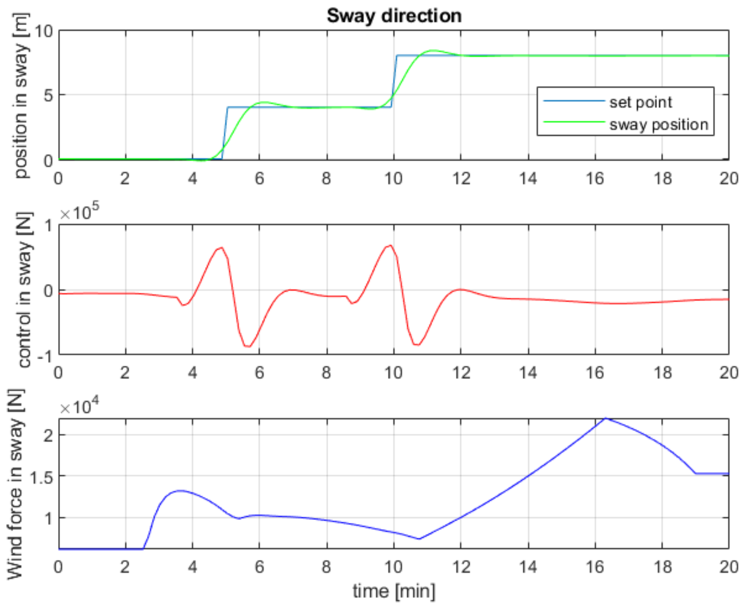


Figure 4.11: Position in Sway in the upper subplot, control force in sway direction in the middle subplot and the wind force in sway direction in the lower subplot

and the position begins to change before the change in set-point. This happens due to feed forward from both the future set-point and the future wind force.

The second plot of figure 4.11 shows the response of the control system to the set-point changes and how it can compensate wind force disturbance in sway direction which is illustrated in the third plot of figure 4.11. Control force in sway takes values between -9×10^4 [N] and 8×10^4 [N].

Direction in Yaw, vessel heading, along the simulation period is shown in the first plot of figure 4.12. The vessel heading set-point changes at time 5[min] from 0[deg] to 14[deg]. Therefore, the MPC controller produces a thruster moment in yaw to stabilize the vessel heading. Between time 10.5 [min] and 16.3[min], the wind moment increases and the control value decreases to compensate this change in the wind moment and to keep the vessel heading in the desired position.

After calculating vessel position in Body coordinate system (surge, sway and yaw), the position is transformed to NED coordinate system by multiplying the position vector (output) by the transformation matrix $\boldsymbol{R}_\psi$ available in equation 2.1. Figure 4.13 shows the development in the vessel position in NED coordinate system along the simulation time.

The vessel was at the origin when the simulation started. At time 5 min, the set point is changed, so the desired position is changed to $(4,4)$ in body coordinate system and the

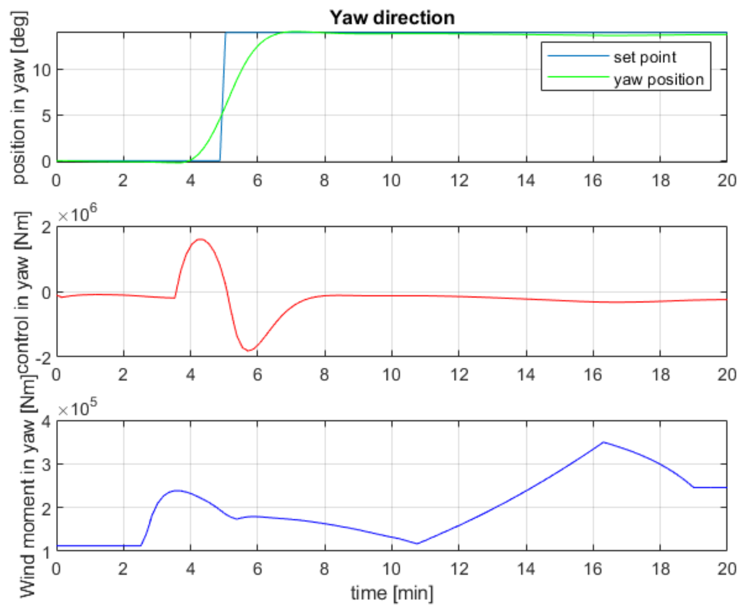


Figure 4.12: Position in yaw in the upper subplot, control moment in yaw direction in the middle subplot and the wind effect in yaw direction in the lower subplot

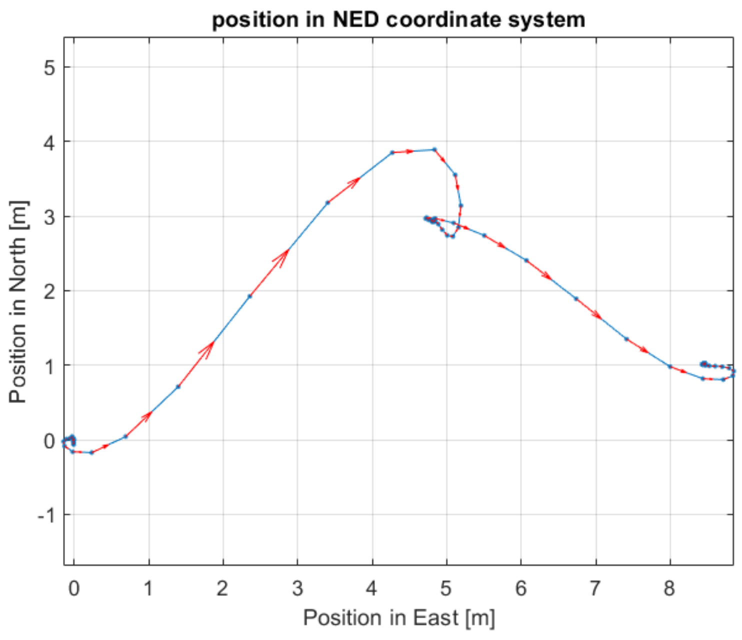


Figure 4.13: Position in NED coordinate system

heading is changed to 14[deg]. This position in Body frame equals to (2.9,4.5) in NED coordinate system. Figure 4.13 shows that the MPC controller in standard form was able to move the vessel to its new desired position under wind force disturbances. This takes about 2[min] to move the vessel.

At time 10[min], the set-point changes from (4,4)[m] to (3,8)[m] in Body coordinate system, and the vessel heading is chosen to be the same. The new position equals to (1,8.5) [m] in NED coordinate system. The MPC controller performs well and was able to manage the vessel position and moves it to the desired place.

Simulation by using a Kalman filter to estimate immeasurable states

In this simulation, Kalman filter is used to estimate immeasurable states of the vessel.

The first plot of figure 4.14 shows the development of the surge position along the simulation time. Where the control force in surge generated to keep the vessel in the desired position under wind force is seen in the second plot. The third plot shows the wind force in surge and how it changes along the simulation. The control acts in opposite way to the wind force to stabilize the vessel position.

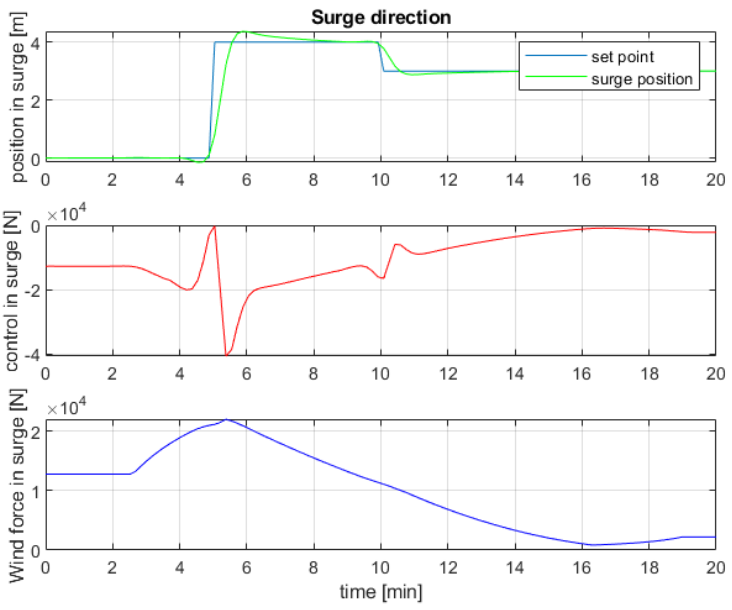


Figure 4.14: Position in Surge in the simulation where Kalman filter is used to estimate immeasurable states, control force in surge, and wind force in surge

By comparing figures 4.10 and 4.14, the overshoot is less when Kalman filter is used to estimate immeasurable states.

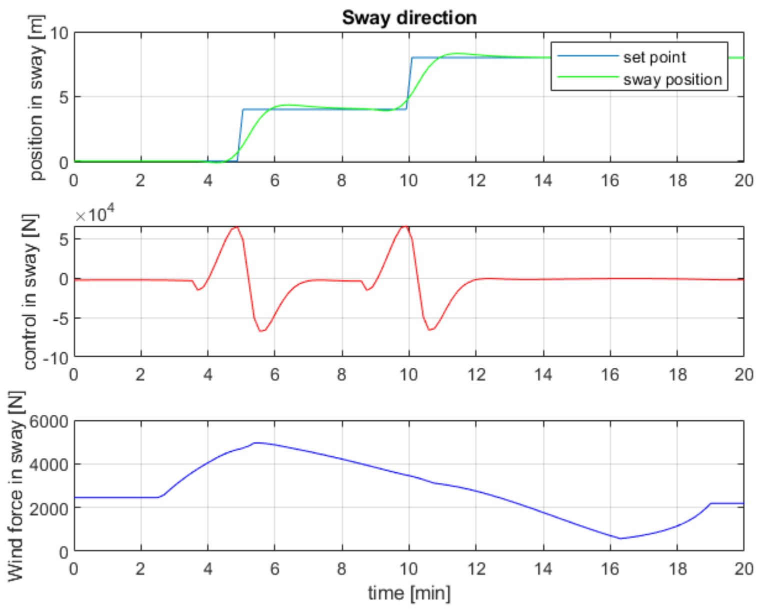


Figure 4.15: Position in sway in the simulation where Kalman filter is used to estimate immeasurable states, control force in sway, and wind force in sway

Figure 4.15 shows vessel position in sway along the simulation period 20[min]. The second plot shows control force in sway and the third plot shows wind force in sway along the simulation.

By comparing figures 4.11 and 4.15, it is found that there is less overshoot in sway position by using Kalman filter to estimate immeasurable states.

Figure 4.16 shows the vessel position in yaw direction [deg] in the first plot. The control moment in yaw [Nm] is seen in the second plot. While wind moment is illustrated in the third plot.

The MPC control is able to stabilize the vessel heading under wind disturbances. In addition it is able to track the changes in set-point in yaw direction.

By comparing figures 4.12 and 4.16 it is found that the overshoot in yaw direction when the Kalman filter is used to estimate immeasurable states is greater than that when it is assumed that all states are measurable.

Figure 4.17 shows the vessel position development in NED coordinate system. The vessel starts at the origin of the coordinate system, then it moves to the new desired position $(4.5, 2.9)$ [m] in North-East directions respectively. After that, it moves to a new desired position $(8.3, 1)$ [m]. By comparing this figure with figure 4.13 when all states are assumed to be measured, it is found that the vessel motion is smoother with less deviation from the set-point by using Kalman filter to estimate immeasurable states.

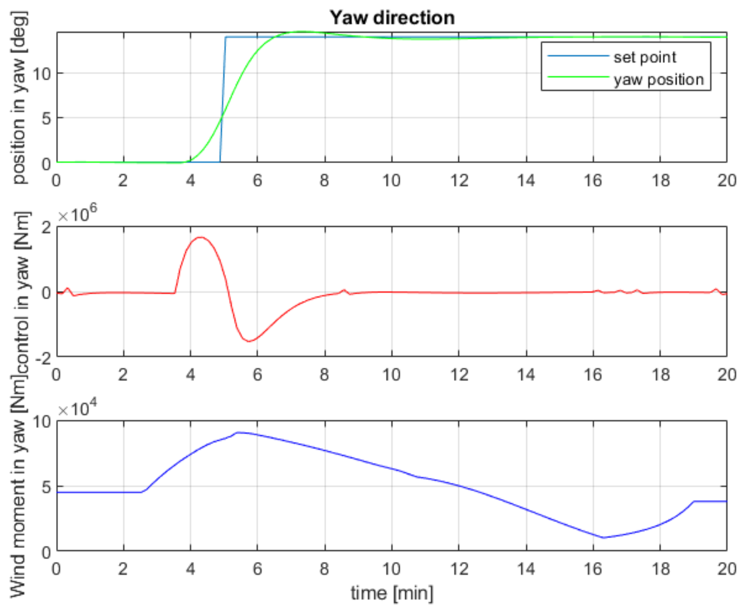


Figure 4.16: Vessel heading in the simulation where Kalman filter is used to estimate immeasurable states, control moment in yaw, and wind moment in yaw

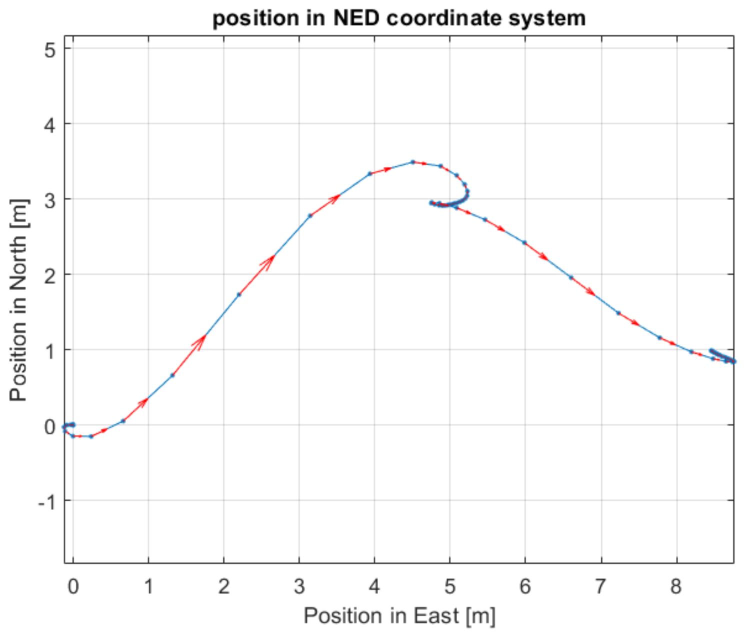


Figure 4.17: Position in NED coordinate system

4.5.2 Reduced size MPC

In this section, reduced size MPC is simulated. Two simulations is about to discuss, one by assuming measurable states and the second one by using Kalman filter to estimate immeasurable states.

Simulation by assuming measurable states

Figure 4.18 shows surge position development in the first plot [m], control in surge [N] in the second plot and wind force in surge direction [N] in the third plot.

Reduced size MPC is able to move the vessel into the desired set-point under wind forces. Where at time 5[min] the set-point changes from 0[m] to 4 [m] and changes again at time 10 min to 3 [m]. The control system acts good to track the set-point and compensate wind-force that was 6×10^4 [N] at the start of the simulation and reduced slowly to about 1×10^4[N] between 2.5 [min] and 10.6[min]. Increasing in wind force from 10[min] to 19[min] causes reducing in control force from -1×10^4 [N] to -6×10^4 [N].

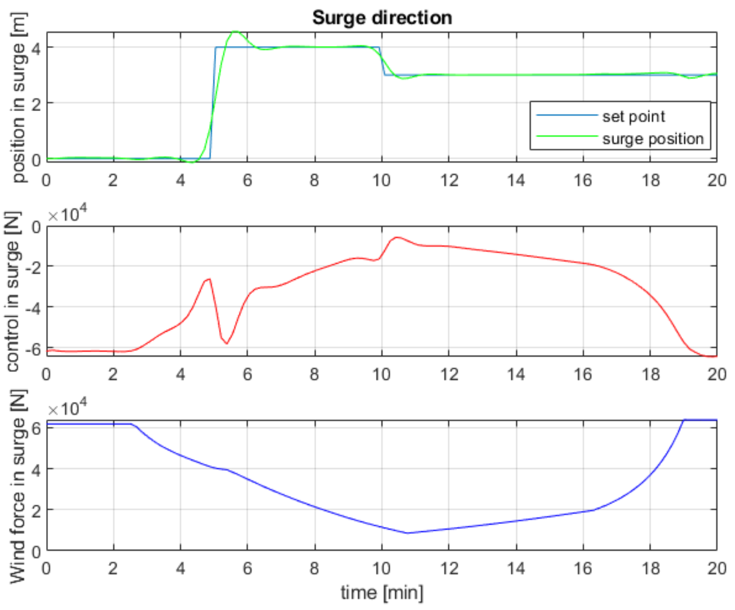


Figure 4.18: Position in surge direction [m] in the first plot, control force in surge [N] developed by reduced size MPC in the second plot and wind force in surge [N] in the third plot

Figure 4.19 shows the changes in vessel position in sway direction and how it tracks the set-point perfectly. The second plot shows the control force in sway direction and how it changes to stabilize the vessel position under wind forces that are seen in the third plot.

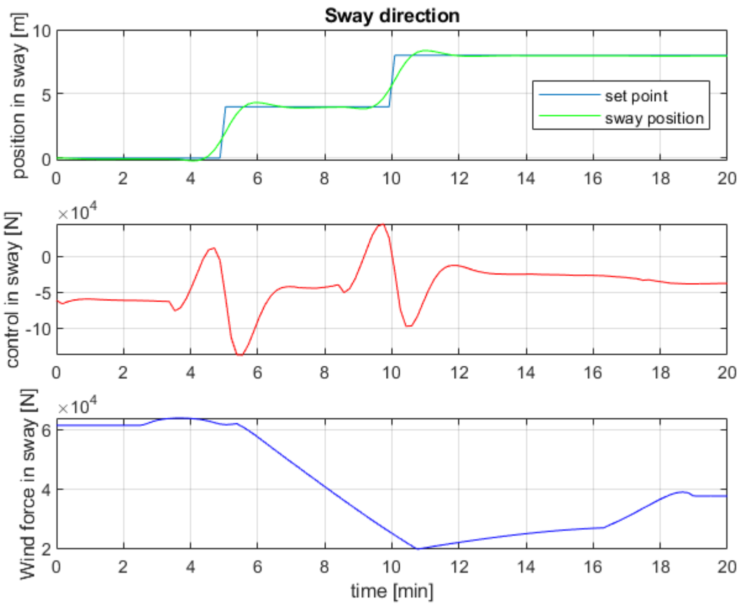


Figure 4.19: Position in sway direction [m] in the first plot, control force in sway [N] developed by reduced size MPC in the second plot and wind force in sway [N] in the third plot

Figure 4.20 shows the vessel heading, yaw direction [deg]. There is a small deviation between the yaw position and the desired set-point between time 1 min and 4 [min] and also after 8 [min] from the starting of the simulation. So the reduced size MPC was able to let the vessel heading be as near as possible to the desired set-point, but cannot reach a zero error. The second plot shows the control moment [Nm] and how it changes in attempt to track the desired set-point under wind moment that is shown in the third plot.

Figure 4.21 shows the vessel position development in NED coordinates system. At the start of the simulation, the vessel was at the origin, then it moves to a new desired position 4.5[m] in east and 2.9[m] in north after a small deviation as shown in figures 4.18 and 4.19. After 10[min] from the start of the simulation, the desired position is changed to 8.4 [m] in east and 1[m] in north. The control system responds to this changed and generates forces and moment to move the vessel into the new desired position under wind disturbances.

Simulation by using Kalman filter to estimate immeasurable states

Figure 4.22 shows the surge position development along the simulation time. The vessel position in surge was 0[m] at the start of the simulation, and it changes to 4[m] at time 5[min] from the start of the simulation. After 5 [min], it is also changed to 3[m].

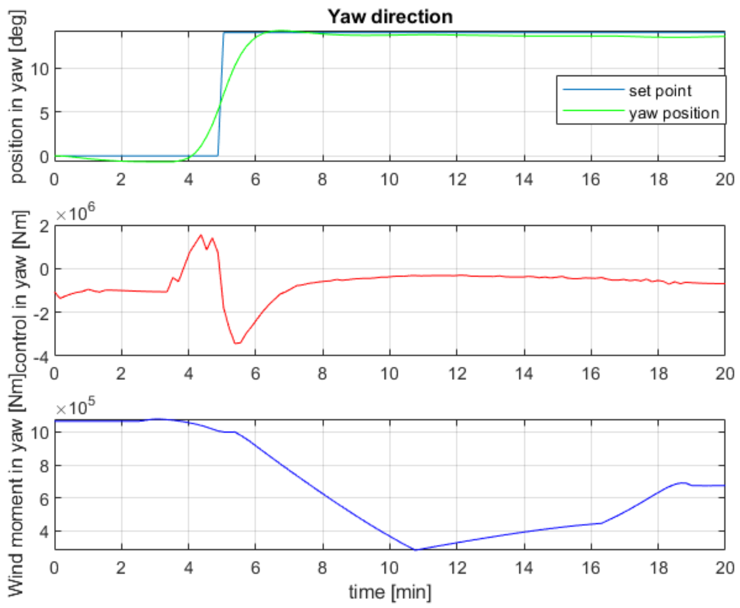


Figure 4.20: Vessel heading direction [deg] in the first plot, control moment in yaw [Nm] developed by reduced size MPC in the second plot and wind moment in yaw [Nm] in the third plot

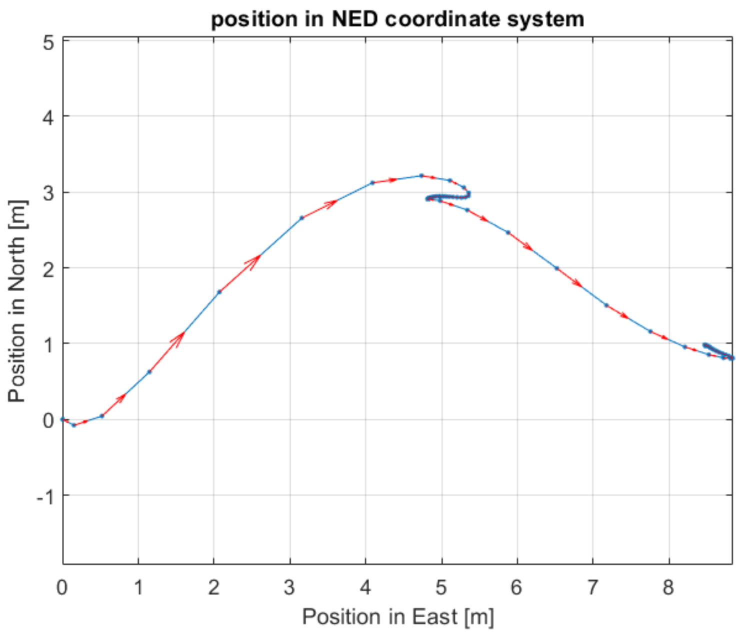


Figure 4.21: Position in NED coordinate system by using reduced size MPC and assuming measurable states

The reduced size MPC control system responds to that change in set-point and succeeds in tracking the set-point and keep the vessel in its desired position under wind force disturbances which are shown in the third plot of figure 4.22. By comparing figures 4.18 and 4.22, it is found the there is less overshoot in surge position when Kalman filter is used to estimate immeasurable states.

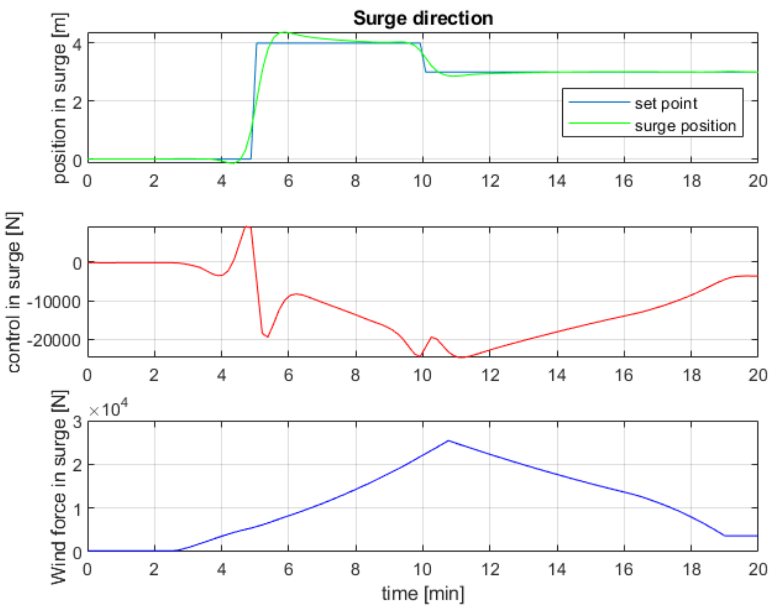


Figure 4.22: Position development in surge by using reduced size MPC and Kalman filter to estimate immeasurable states in the first plot, control force in surge in the second plot and wind force in surge in the third plot

Figure 4.23 shows vessel position in sway direction. At the simulation start, the vessel was at 0[m]. The set-point in sway is changed to 4[m] after 5[min] and changed again to 8[m] after 10[min] of the simulation starting time. The controller generates a force in sway direction to track the desired set-point. This force and how it changes along the simulation time is shown in the second plot of figure4.23. Where the third plot shows the wind force in sway direction along the simulation time. There was no wind at the beginning of the simulation, but after 3[min] it starts to increase until it reaches 2.9×10^4[N] at 10.8[min]. After that, it begins to decrease until it reaches 0.5×10^4[N] at the end of the simulation. The control system is able to keep the vessel in its desired position in sway under that wind force changes.

By comparing figures 4.19 and 4.23, it is found that the vessel position in sway got a smaller overshoot when Kalman filter is used to estimate immeasurable states.

Figure 4.24 shows vessel position in yaw direction [deg], vessel heading, in the first plot. The vessel heading was 0 [deg] at the simulation start. After 5[min] the heading set-point

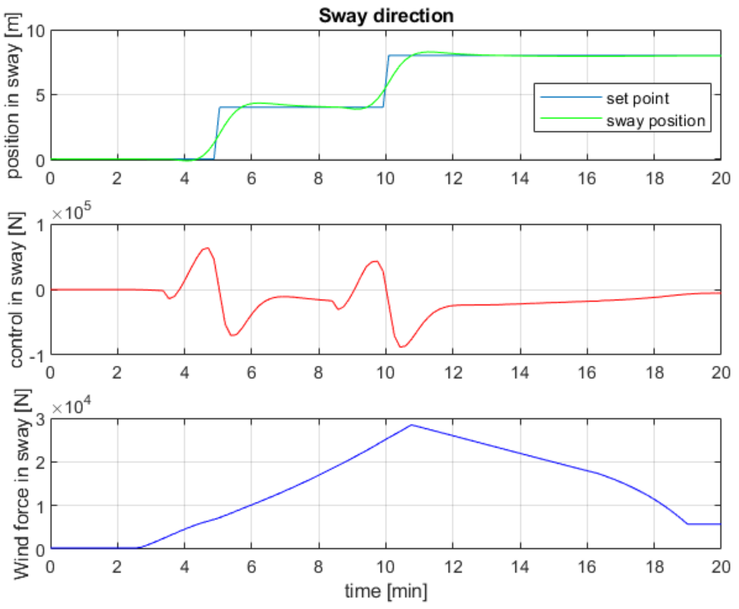


Figure 4.23: Position development in sway by using reduced size MPC and Kalman filter to estimate immeasurable states in the first plot, control force in sway in the second plot and wind force in sway in the third plot

is changed to 14[deg]. The control system was able to track the desired set-point with a small deviation between 9[min] and 15[min] of the simulation.

Figure 4.25 shows the vessel position in NED coordinate system along the simulation time. The vessel was at the origin of the NED coordinate system when the simulation started, then it moves to its new desired set point $(2.9, 4.5)$[m] in north and east directions respectively. After that, the control system generated forces and moment to move the vessel to its new desired position $(1, 8.4)$[m] in north and east direction respectively.

By comparing figures 4.21 and 4.25, it is found that the vessel position deviation from the set-point is less by using Kalman filter to estimate immeasurable states.

4.5.3 Simple MPC

This section shows the simulations done by using simple MPC to control the vessel position. The simple MPC reduces the size of the optimization problem in the reduced size MPC described in chapter 3 from $(N \times (n_x + n_u))$ to $(N \times n_u)$.

Figure 4.26 shows the vessel position development in surge direction and by not including wind force disturbances in the MPC control system to estimate the optimal control. The figure shows deviation between the set-point and the vessel position in surge. This

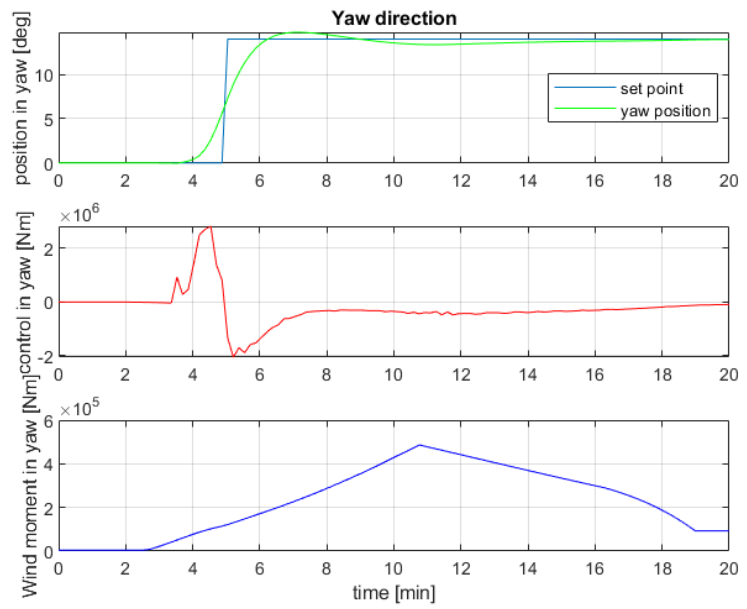


Figure 4.24: Vessel heading development by using reduced size MPC and Kalman filter to estimate immeasurable states in the first plot, control moment in yaw in the second plot and wind moment in yaw in the third plot

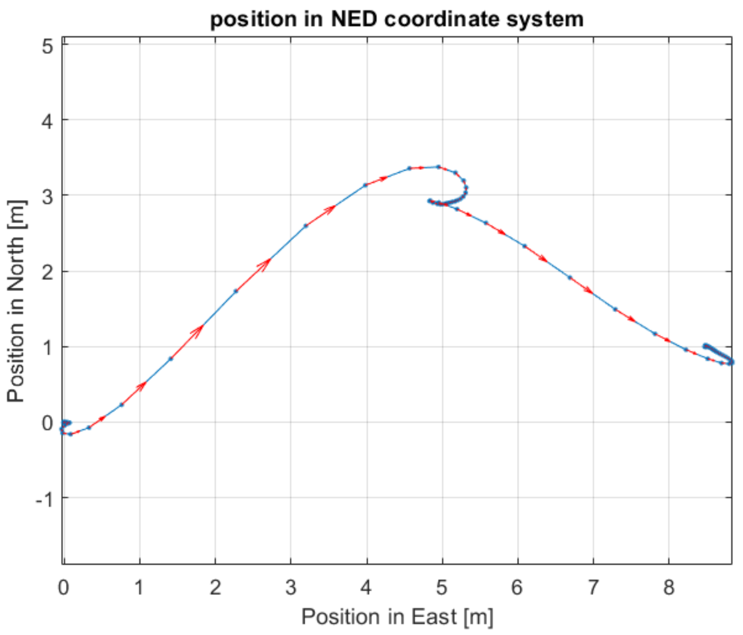


Figure 4.25: Position in NED coordinate system developed by using reduced size MPC and Kalman filter

deviation is due to wind force disturbances that affects the vessel position and shown in the third plot of figure 4.26. The wind force in surge decreases from 4 [min] to about 12 [min] of the simulation time. The control signal increases during that time in attempt to compensate wind force effect on the vessel position in surge, but could not achieve zero error. Control signal development along the simulation is shown in the second plot of figure 4.26.

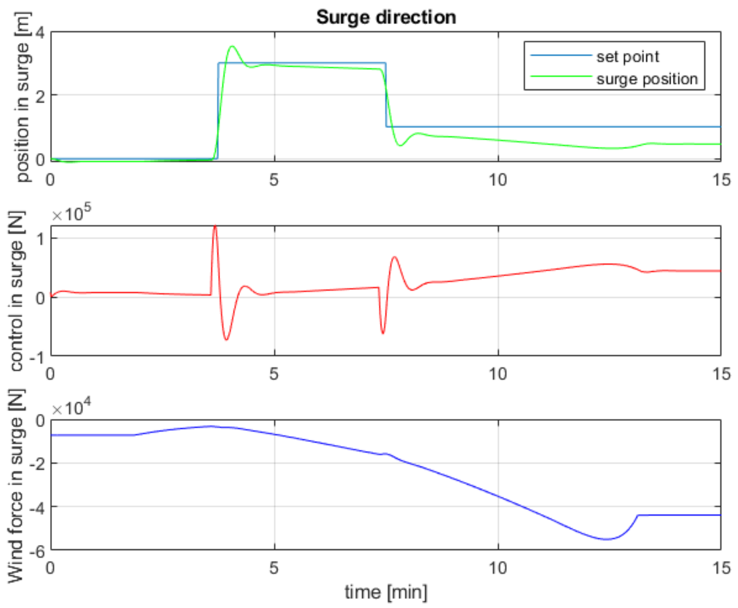


Figure 4.26: Position development in surge using simple MPC in the first plot, control force in surge in the second plot and wind force in surge in the third plot

The first plot of figure 4.27 shows vessel position development in sway direction subject to wind force in sway shown in the third plot. The control signal that is developed to control the vessel position in sway is shown in the second plot of figure 4.27. The controller attempts to stabilize the vessel in the sway direction despite of wind force in sway. It succeeded in achieving a small error. Where the first plot shows a very small deviation between the vessel position in sway and the desired set-point between 8 [min] and 15 [min] of the simulation time.

Figure 4.28 shows the vessel heading, position in yaw direction, during the simulation time. The control system has been able to stabilize the vessel heading despite of wind moment that is shown in the third plot of figure 4.28.

The vessel position in NED coordinate system is shown in figure 4.29. The vessel was at the origin of the NED coordinate system at the start time. After 3.75 [min], the set-points in the three directions (surge, sway and yaw) are changed. The vessel moves to the new

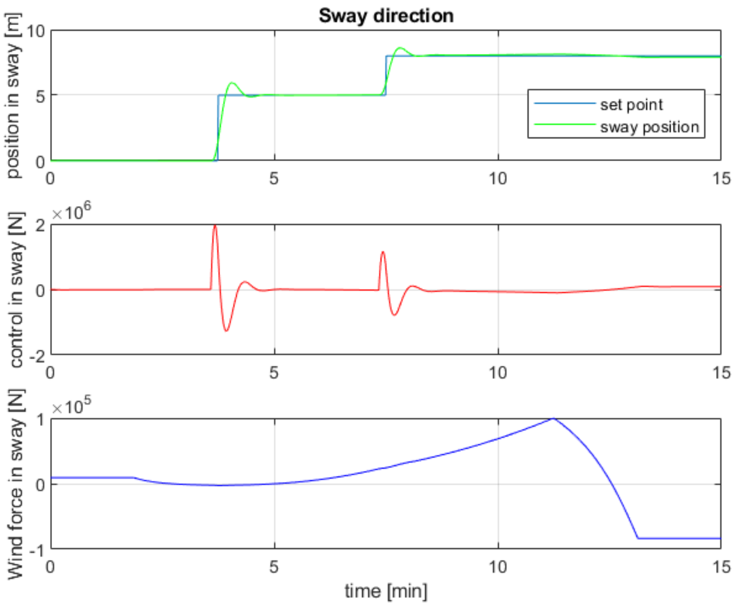


Figure 4.27: Position in sway using simple MPC in the first plot, control force in sway in the second plot and wind force in sway in the third plot

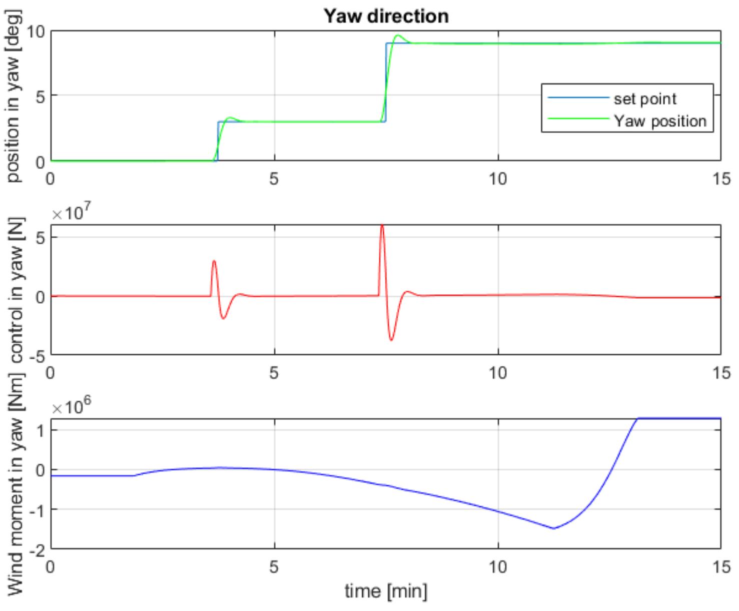


Figure 4.28: Position in yaw using simple MPC in the first plot, control moment in yaw direction in the second plot and wind effect in yaw in the third plot

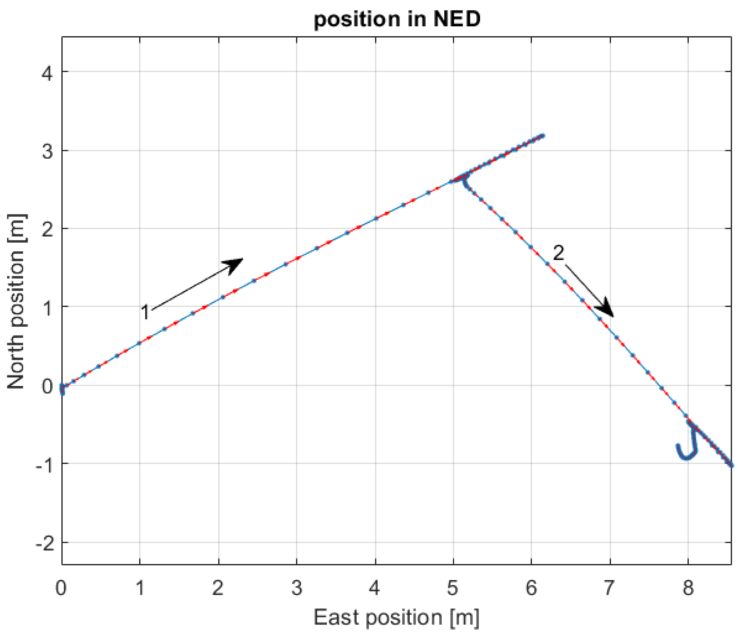


Figure 4.29: Position in NED coordinate system using simple MPC

desired set-point but with some error in surge direction. After 7.5 [min] of the simulation starting time, the set-points are changed again, and the control system is able to move the vessel into the new desired set-point but with deviation in surge as shown in figure 4.26.

4.5.4 Simple MPC with integral action

This section shows simulation results of using simple MPC with integral action developed by David D. Ruscio [12] and described in chapter 3.

Figure 4.30 shows the vessel position in surge direction in the first plot and how the control system is able to track the set-point under wind force in surge shown in the third plot of figure 4.30. The control signal calculated by the control system along the simulation is shown in the second plot.

By comparing figures 4.26 and 4.30, it is shown that the simple MPC with integral action is more efficient than the standard simple MPC in tracking the desired position despite of the wind force affecting the vessel.

Figure 4.31 shows vessel position in sway direction in the first plot. The control signal is shown in the second plot and the wind force in sway is illustrated in the third plot. The simple MPC with integral action has been able to track the desired set-point with zero error comparing to the standard simple MPC behaviour shown in figure 4.27.

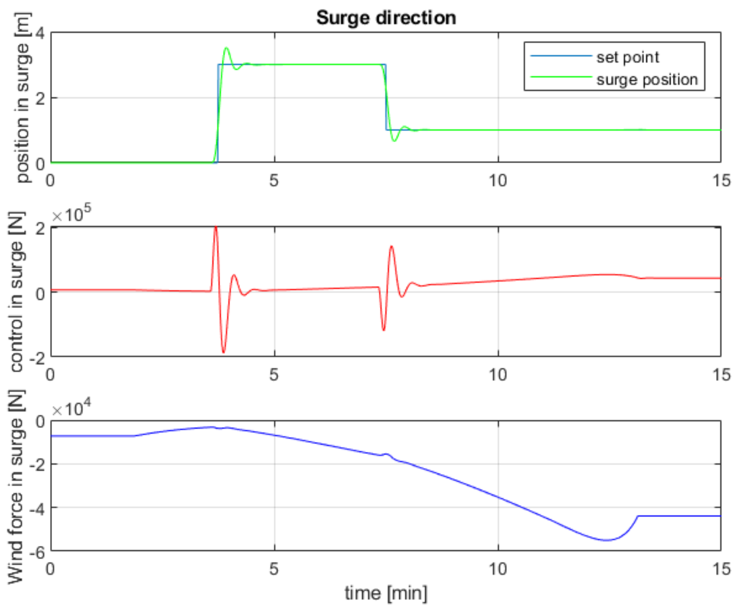


Figure 4.30: Position in surge by using simple MPC with integral action in the first plot, control force in surge in the second plot and wind force in surge in the third plot

Figure 4.32 shows the vessel heading along the simulation time. The simple MPC with integral action is able to track the desired heading with zero error as same as the standard simple MPC behaviour shown in figure 4.28.

Figure 4.33 shows the vessel position in NED coordinate system. The vessel was at the origin at the start time. After 3.75 [min], the desired position is changed to 3[m] in surge and 5 [m] in sway. This is equivalent to 2.7 [m] in North direction and 5.2 [m] in East direction. The control system moves the vessel to the desired set-point despite of wind forces and moment. After 7.5 [min] of the simulation starting, the desired is changed again to 1 [m] in surge and 8 [m] in sway which is equivalent to -0.3 [m] in north and 8.1 [m] in east. The control system was able to move the vessel to its new desired position.

4.6 LQ optimal control with integral action

This section describes simulation results of using LQ optimal control with integral action. Two simulations are done, the first one without limitations on control amplitude, and the second one shows the behaviour when control signals amplitudes are limited.

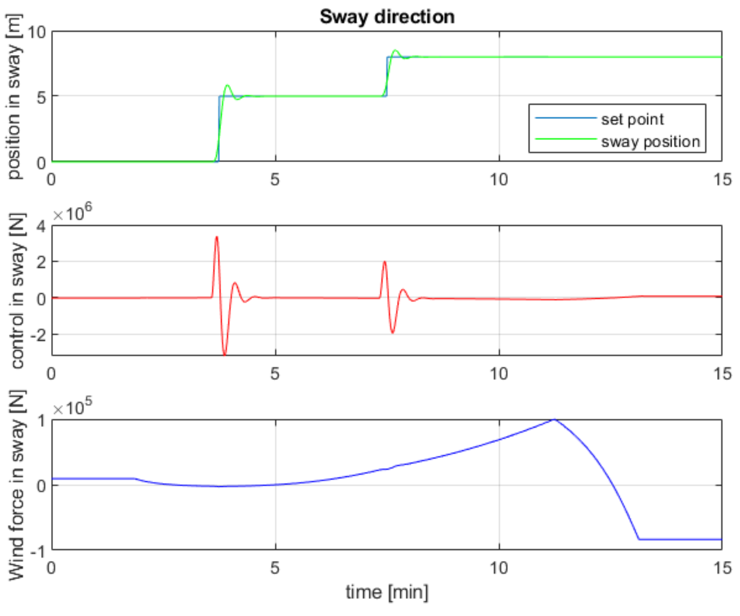


Figure 4.31: Position in sway by using simple MPC with integral action in the first plot, control force in sway in the second plot and wind force in sway in the third plot

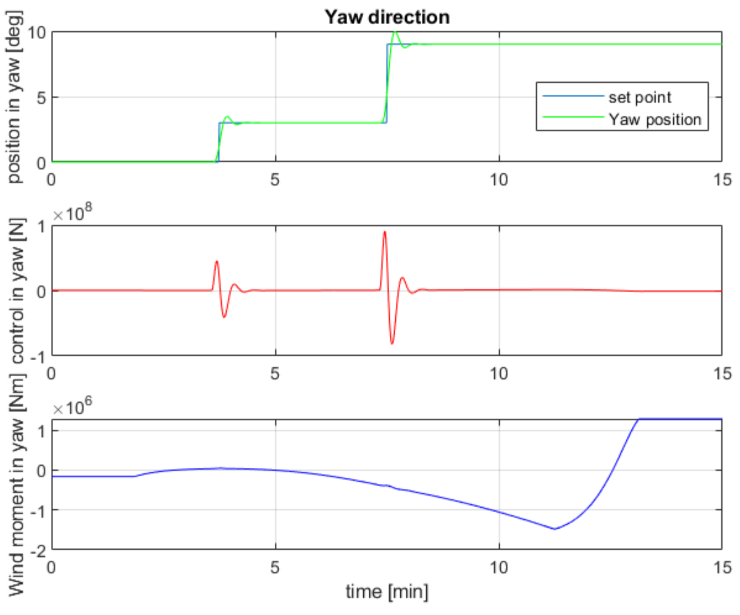


Figure 4.32: Position in yaw by using simple MPC with integral action in the first plot, control moment in yaw in the second plot and wind moment in yaw in the third plot

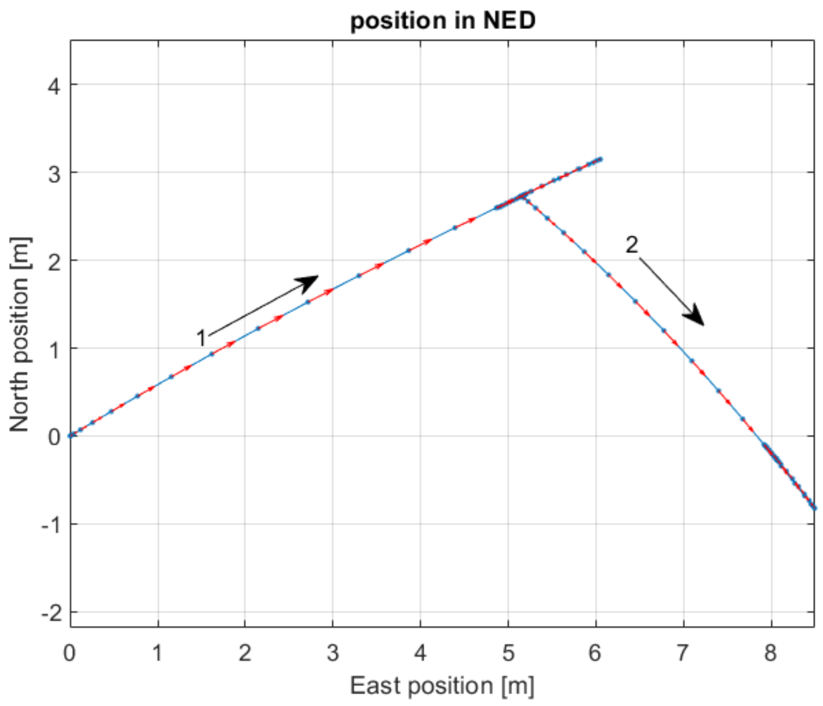


Figure 4.33: Position in NED coordinate system by using simple MPC with integral action

4.6.1 LQ optimal control without constraints

By applying LQ optimal control method described in chapter 3 to get the optimal control value, it is found that this type of control is able to stabilize the vessel position despite of wind disturbances.

Figure 4.34 shows the vessel position in surge direction in the first plot and how it changes according to changes in set-point. The control system generates a control signal in surge shown in the second plot of figure 4.34 that make the vessel moves in surge direction to reach the desired position and keeps it despite of wind force in surge shown in the third plot of figure 4.34.

The vessel was at 0 [m] in surge direction at the simulation starting time. After 5 [min] of the simulation, the set-point is changed to 15 [m] and changed again to 8 [m] after 10 [min] of the simulation starting time.

Figure 4.35 shows the vessel position in sway direction in the first plot. The vessel was at the origin of the body coordinate system at the simulation starting time. After 5 [min], the set-point in sway is changed to 20 [min]. The control system generates thrust in sway, shown in the second plot, to respond to changes in the set-point. The vessel moves to the new desired position and it takes about 3 [min] to reach it. The thrust force generated to achieve that is about 4.5×10^5 [N].

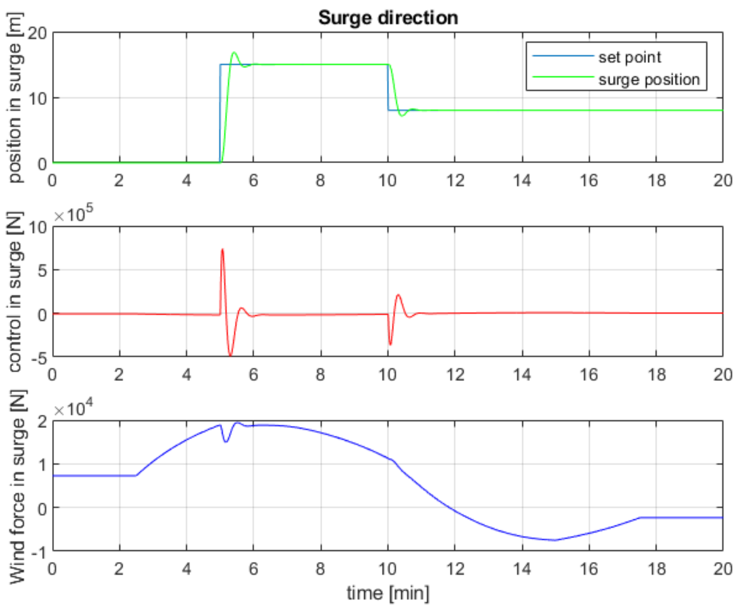


Figure 4.34: Position in surge using LQ with integral action int the first plot, control force in surge in the second plot and wind force in surge in the third plot

After 10 [min] of the simulation starting time, the set-point in sway is changed again from 20 [m] to 6 [m]. The control system responds to that change and generates sway thrust equals to about -4.5×10^5 [N]. The generated thrust moves the vessel to the new desired position in sway and it takes about 2.5 [min] to achieve that.

Figure 4.36 shows the vessel heading, position in yaw direction [deg] in the first plot. The vessel heading was 0 [deg] at the simulation starting time. After 5 [min], the desired heading is changed to 14 [deg]. The control system generates thruster moment to rotate the vessel to achieve the new desired heading. The generated moment is bout 6×10^6 [Nm]. This moment is able to compensate the effect of the wind disturbance on the vessel heading.

After 10 [min] of the simulation starting time, the desired heading is changed to 6 [deg]. The control system responds to that change by generating thrusters moment which is approximately equals to -3.9×10^6 [Nm] to rotate the vessel and achieve the desired heading. The control system is able to stabilize the vessel heading in the desired position despite of the effect of wind moment shown in the third plot of figure 4.36.

Figure 4.37 shows the vessel position in NED coordinate system along the simulation time. The vessel was at the origin of the NED coordinate system at the simulation starting time. It moves upward and then rotates to east to reach the new desired position which is 15[m] in surge and 20 [m] in sway. The new desired position in NED coordinate system is 9.7 [m]

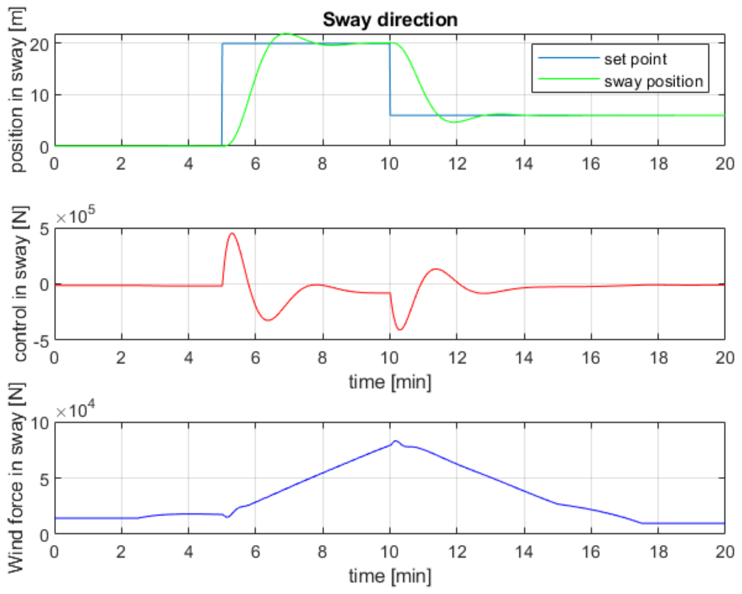


Figure 4.35: Position in sway using LQ with integral action in the first plot, control force in sway in the second plot and wind force in sway in the third plot

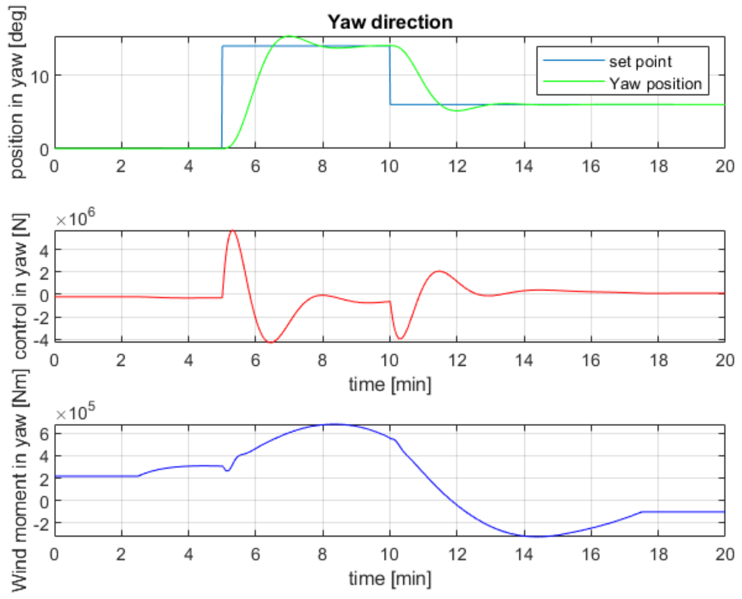


Figure 4.36: Position in yaw using LQ with integral action in the first plot, control moment in yaw in the second plot and wind moment in yaw in the third plot

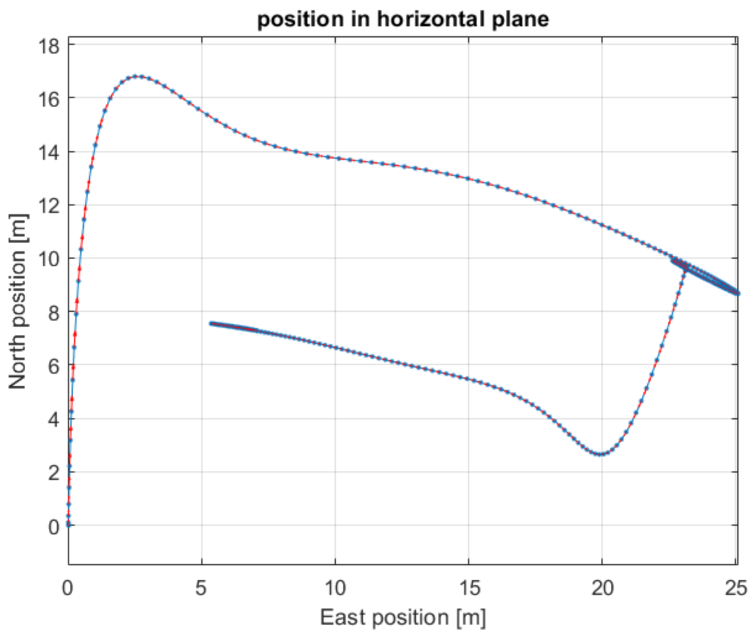


Figure 4.37: Position in NED coordinate system using LQ with integral action

in North and 23 [m] in east. The control system generates thrusters forces and moment to move the vessel to its new desired position as shown in figure 4.37.

After 10 [min] of simulation starting time, the desired position is changed to 8 [m] in surge and 6 [m] in sway which is equivalent to 7.3 [m] in North and 6.8 [m] in East. The control system is able to move the vessel to this new desired position as shown in Figure 4.37.

4.6.2 LQ optimal control with constraints

This section shows the simulation results by limiting the inputs amplitude by using if-else statements. The MATLAB script that is used to do the simulation is available in appendix L.

Figure 4.38 shows the vessel position in surge direction in the first plot. The second plot shows the limited control force in surge. The generated force is limited to take a maximum value of 1×10^5 [N] and minimum value of -1×10^5 [N]. The generated force in surge is able to move the vessel to the desired position in surge. Comparing with figure 4.34, it takes longer time to reach the desired position when the control force is limited. Where without limits, it took about 1 [min] to move the vessel from 0 [m] to 15 [m] in surge, but with limited control amplitude, it took about 1.5 [min] to move the vessel the same distance.

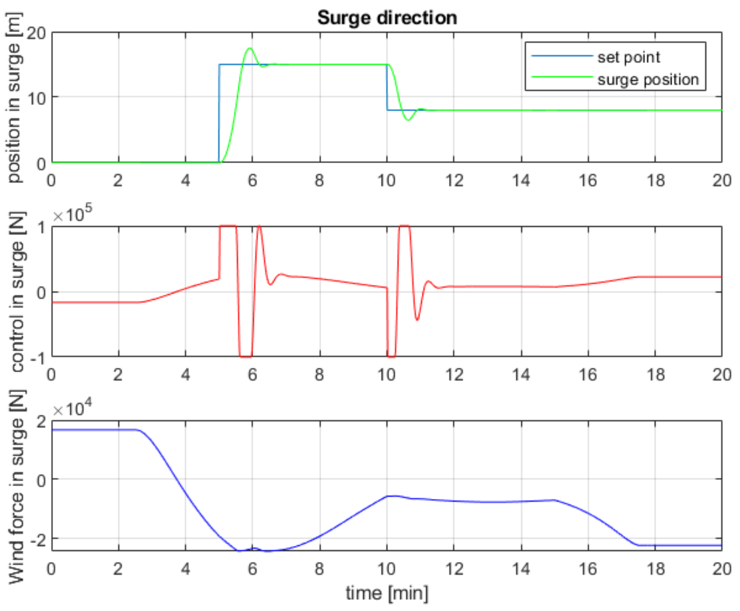


Figure 4.38: Position in surge using LQ with integral action with input amplitude limits in the first plot, limited control force in surge in the second plot and wind force in surge in the third plot

Figure 4.39 shows the vessel position in sway and how it is developed along the simulation time by limiting the control amplitude in sway. The second plot shows the limited control force in sway while the third plot shows the wind force in sway direction [N]. When the set-point in sway is changed to 20 [m], the control system attempts to generate the required force in sway to track the set-point, but due to control amplitude limitations, the force in sway is limited to take values between -1^5 [N] and 1×10^5 [N]. This limitation makes the vessel deviates from the desired position with a big overshot as shown in the first plot of figure 4.39. The set-point is changed again before the control system succeeds in achieving zero error. The new set-point in sway is 8 [m] and it is clearly that the control system was able to track the set-point and make the vessel moves to the desired position in sway after about 4 [min] of last change of the set-point.

Comparing to figure 4.35, it is found that limiting the control amplitude in sway, making it takes longer time to track the set-point which is expected.

Figure 4.36 shows the vessel heading development along the simulation time in the first plot, control in yaw in the second plot, and the wind moment in the third plot. The control amplitude is limited to take values between -1×10^7 [Nm] and 1×10^7 [Nm]. The change in set-point during the simulation time, makes the control system responds by generating thusters moment within the limits. The generated moment shown in the second plot was able to make the vessel rotate to the desired position in yaw.

Figure 4.41 shows the vessel position development in the NED coordinate system along the

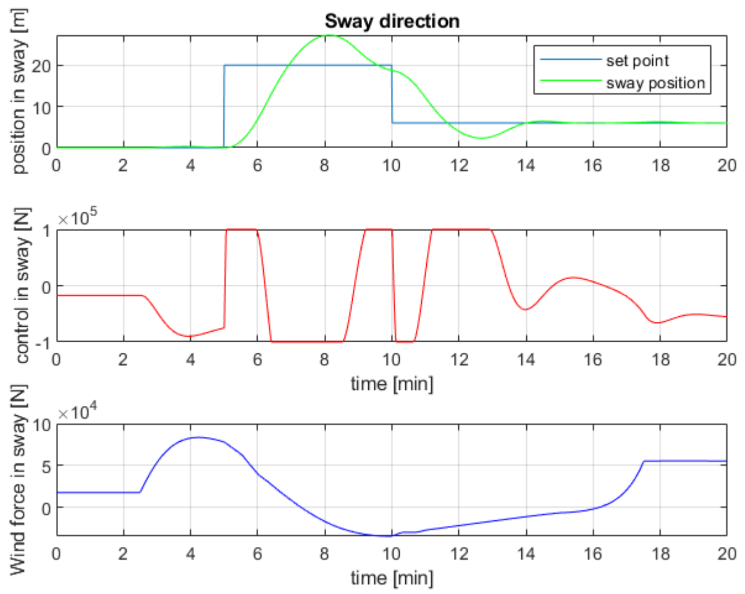


Figure 4.39: Position in sway using LQ with integral action with input amplitude limits in the first plot, limited control force in sway direction in the second plot and wind force in sway in the third plot

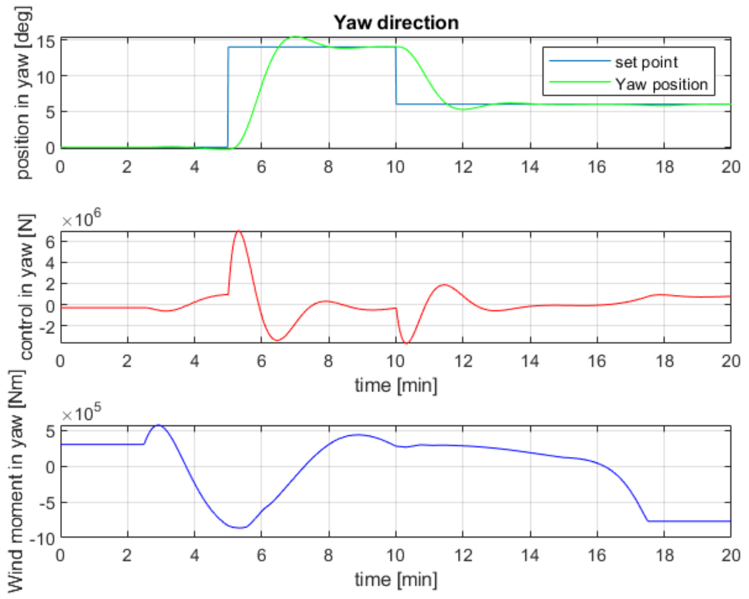


Figure 4.40: Position in yaw using LQ with integral action with input amplitude limits in the first plot, limited control moment in yaw in the second plot and wind moment in yaw in the third plot

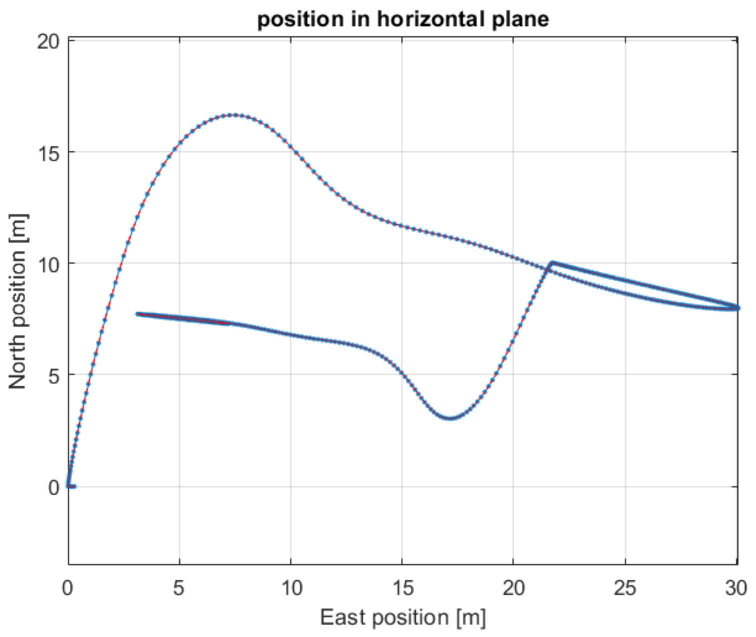


Figure 4.41: Position in NED coordinate system using LQ with integral action with input amplitude limits

simulation time. The control system was able to move the vessel to the desired positions but comparing to figure 4.37, the vessel deviated greater from the desired position when the inputs are limited and this is expected because without limits, the generated forces and moment is greater and therefore the response is faster in tracking the set-point and achieving zero error.

4.7 Comparison between simple MPC with integral action and LQ optimal control with integral action

This section compares the behaviour of MPC control with integral action and LQ optimal control with integral action.

The upper plot of figure 4.42 shows the development of the vessel position in surge direction by using the simple MPC with integral action with the black line and with LQ optimal control with integral action with red line. It is found that the simple MPC with integral action acts faster than LQ optimal control with integral action and with a smaller deviation from the desired set-point.

The lower plot of figure 4.42 shows the control signal i surge direction developed by simple MPC with integral action with the black line and the control signal in surge developed by LQ optimal control with integral action with the red line. It is found that the simple MPC with integral action acts before the LQ optimal control with integral action and with smaller amplitude. The upper plot of figure 4.43 shows the development of the

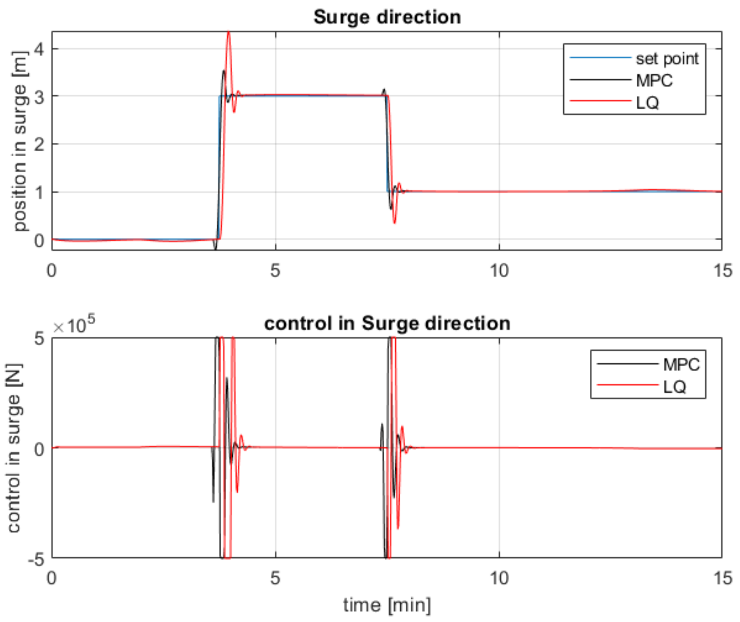


Figure 4.42: Comparison between simple MPC with integral action and LQ optimal control with integral action in surge direction

vessel position in sway direction by using the simple MPC with integral action with the black line and with LQ optimal control with integral action with red line. It is found that the simple MPC with integral action acts faster than LQ optimal control with integral action and with approximately the same deviation from the desired set-point.

The lower plot of figure 4.43 shows the control signal i sway direction developed by simple MPC with integral action with the black line and the control signal in surge developed by LQ optimal control with integral action with the red line. It is found that the simple MPC with integral action acts before the LQ optimal control with integral action.

The upper plot of figure 4.44 shows the development of the vessel position in yaw direction by using the simple MPC with integral action with the black line and with LQ optimal control with integral action with red line. It is found that the simple MPC with integral action acts faster than LQ optimal control with integral action and with a smaller deviation from the desired set-point.

The lower plot of figure 4.44 shows the control signal i yaw direction developed by simple MPC with integral action with the black line and the control signal in surge developed by LQ optimal control with integral action with the red line. It is found that the simple

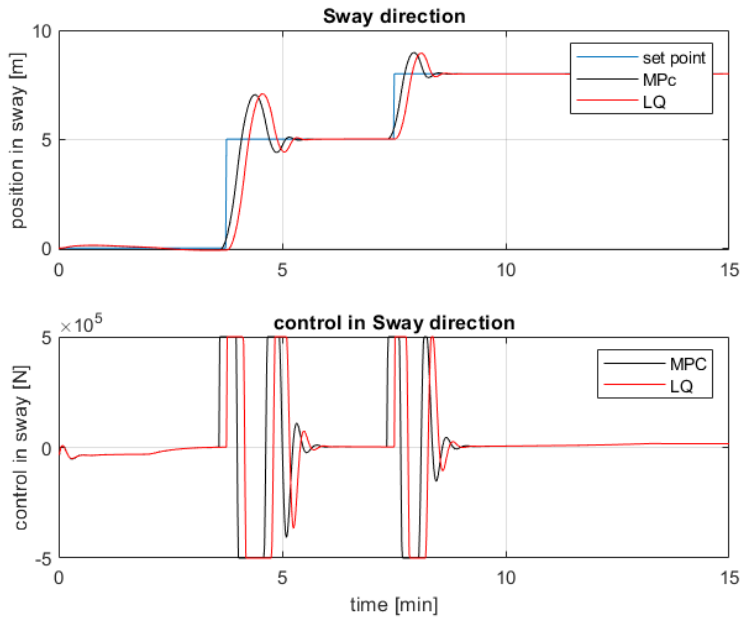


Figure 4.43: Comparison between simple MPC with integral action and LQ optimal control with integral action in sway direction

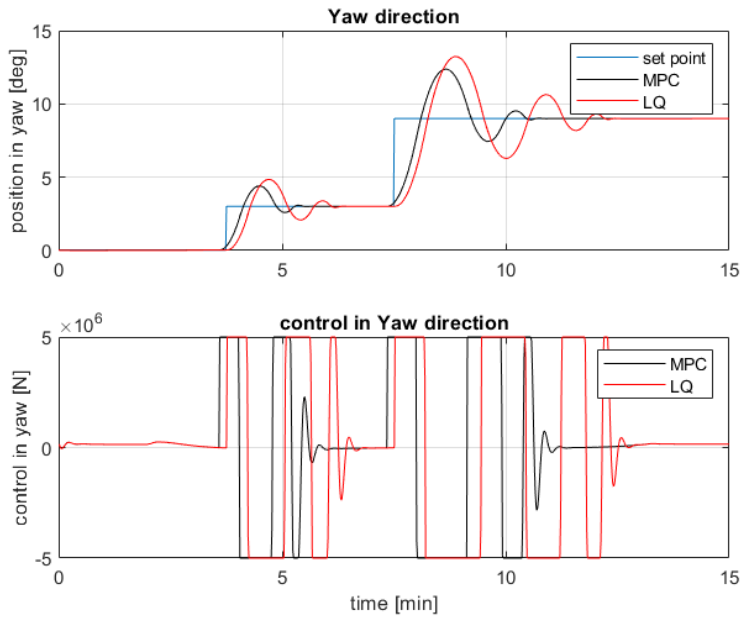


Figure 4.44: Comparison between simple MPC with integral action and LQ optimal control with integral action in yaw direction

MPC with integral action acts before the LQ optimal control with integral action and with smaller amplitude.

4.8 System identification

This section describes the result of identifying a SSM of the studied vessel from a set of known input- output data. In addition the identified system is used to develop a model-free simple MPC with integral action and model-free LQ optimal control with integral action.

4.8.1 Identified state space model

By using the *dsr _ e.m* function to identify a SSM, system matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}$ and $\boldsymbol{E}$ and kalman gain $\boldsymbol{K}$ are gotten and given in equations 4.5 to 4.9.

$$A = \begin{bmatrix} 1.0001 & 0.0005 & -1.9253 & 0.0063 & 0.5407 & -0.0819 \\ -0.0000 & 0.9999 & 0.5413 & -0.0081 & 1.9229 & -0.1264 \\ -0.0000 & -0.0000 & 1.0000 & -0.0002 & 0.0000 & 0.0357 \\ -0.0000 & -0.0000 & 0.0008 & 1.0001 & -0.1004 & -1.9956 \\ 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.9997 & -0.0497 \\ -0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & 1.0005 \end{bmatrix} \tag{4.5}$$

$$B = 1 \times 10^{-6} \times \begin{bmatrix} -0.5106 & -0.0144 & 0.0000 \\ 0.1429 & -0.0509 & -0.0000 \\ 0.1768 & -0.0001 & -0.0000 \\ -0.0005 & 0.0027 & -0.0000 \\ -0.0002 & -0.0176 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{4.6}$$

$$D = \begin{bmatrix} -0.6809 & 0.1908 & -0.7071 & 0.0032 & 0.0007 & -0.0000 \\ -0.1904 & -0.6801 & 0.0007 & 0.0332 & 0.7072 & -0.0001 \\ -0.0098 & -0.0340 & -0.0029 & -0.7063 & -0.0023 & -0.7070 \end{bmatrix} \tag{4.7}$$

$$E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.8}$$

$$K = \begin{bmatrix} -3.4045 & -0.9522 & -0.2028 \\ 0.9531 & -3.4003 & -0.2555 \\ 0.7071 & -0.0008 & 0.0481 \\ -0.0013 & 0.1793 & -3.5256 \\ -0.0007 & -0.7065 & -0.0729 \\ 0.0000 & 0.0000 & 0.7079 \end{bmatrix} \tag{4.9}$$

To study system stability, controllability and observability, eigenvalues of system matrix A, rank of controllability matrix and rank of observability matrix are calculated respectively. Eigenvalues of system matrix A are shown in equation 4.10. The system is unstable since there are some eigenvalues that are located outside the unity circle, but they are very close to the unity circle bounds.

$$eig(A) = \begin{bmatrix} 1.0000 + 0.0017i \\ 1.0000 - 0.0017i \\ 0.9987 + 0.0000i \\ 0.9995 + 0.0000i \\ 1.0013 + 0.0000i \\ 1.0008 + 0.0000i \end{bmatrix} \tag{4.10}$$

The controllability matrix of the pair (A,B) is calculated using *ctrb.m* function in MATLAB. While the rank of the controllability matrix is calculated using *rank.m* function. The rank of the controllability matrix is 6 and it is equal to the system order n_x so the system is controllable.

The observability matrix of the pair (A,D) is calculated using *obsv.m* function in MATLAB. While the rank of it is calculated using *rank.m* function. The rank of the observability matrix is 6 and it is equal to the system order n_x so the system is observable.

4.8.2 Model-free simple MPC with integral action

The identified system in section 4.8.1 is used in simple MPC with integral action described in section 3.6.4. Weighting matrices are tuned to get an acceptable behaviour. The MATLAB script that is used to simulate the behaviour of the developed control system and its effect on the vessel is available in appendix P.

Figure 4.45 shows the vessel position in surge in the first plot and how it tracks the set-point under the control force effect that is shown in the second plot and the wind force shown in the third plot.

Figure 4.46 shows the vessel position in sway direction. The set-point is changed three times along the simulation period. The simple MPC with integral action which is developed by using the identified linear state space model was able to move the vessel to the desired position under wind disturbance shown in the third plot of figure 4.46.

Figure 4.47 describes the vessel heading development along the simulation period in the first plot where the set-point is changed three times and the vessel rotates to track the set-point each time it is changed. The second plot shows the control moment that rotates the vessel to achieve zero error between its heading and the heading set-point. The control system was able to stabilize the vessel heading despite of wind effect in yaw direction shown in the third plot of figure 4.47.

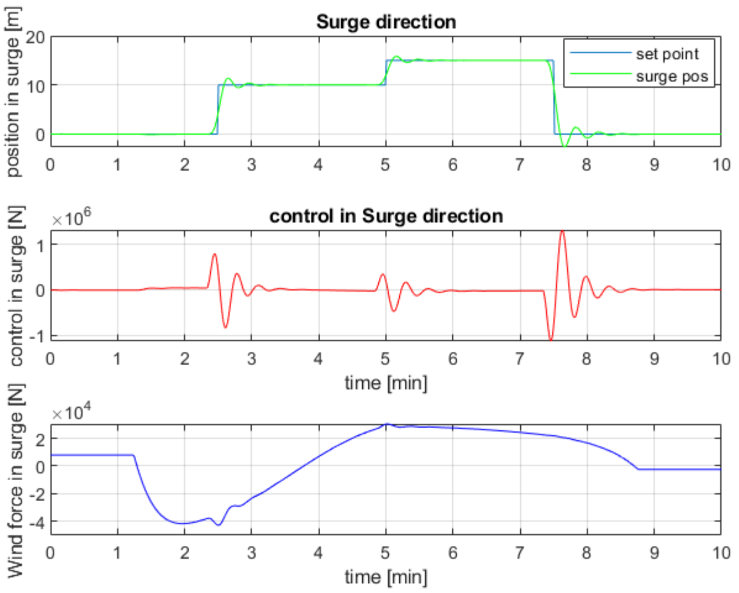


Figure 4.45: Position in surge using simple MPC with integral action and system identification

Figure 4.48 shows the vessel position in NED coordinate system along the simulation period. The numbered arrows shows the movement direction. The first arrow shows the vessel movement from the origin of NED coordinate system to the first desired position which is 10 [m] in surge and 10 [m] in sway and that is equivalent to 7.7 [m] in North and 11.9 [m] in East. While the second arrow shows the vessel movement to the second desired position which is 15 [m] in surge and 20 [m] in sway. This is equivalent to 10 [m] in North and 22.9 [m] in East. The third and last arrow shows the vessel movement to the last desired position which is 0 [m] in surge and 15 [m] in sway. This position is equivalent to -3.6 [m] in North and 14.6 [m] in East. Where the position in NED coordinate system is calculated by multiplying the position in Body coordinate system with the transformation matrix $R(\psi)$ in equation 2.1.

4.8.3 Model-free LQ optimal control with integral action

This section shows results of simulating LQ optimal control with integral action which is developed using the identified linear state space model.

Figure 4.49 shows the vessel position in surge direction in the first plot. The LQ optimal control with integral action generates thrust in surge direction, shown in the second plot, that was able to move the vessel to the desired position despite of the wind force in surge that is shown in the third plot.

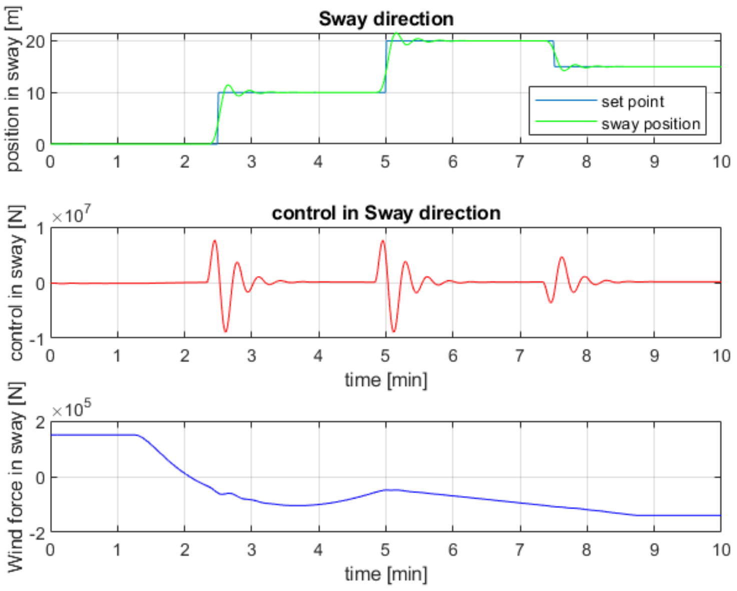


Figure 4.46: Position in sway using simple MPC with integral action and system identification

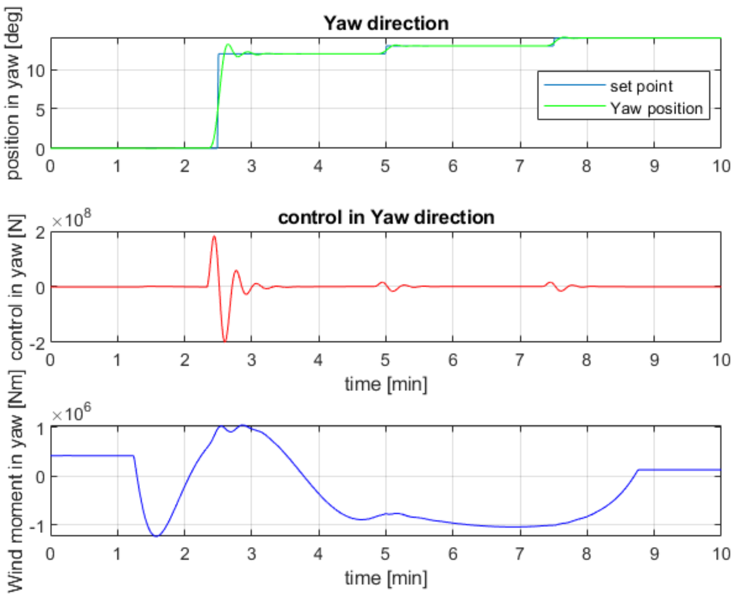


Figure 4.47: Position in yaw using simple MPC with integral action and system identification

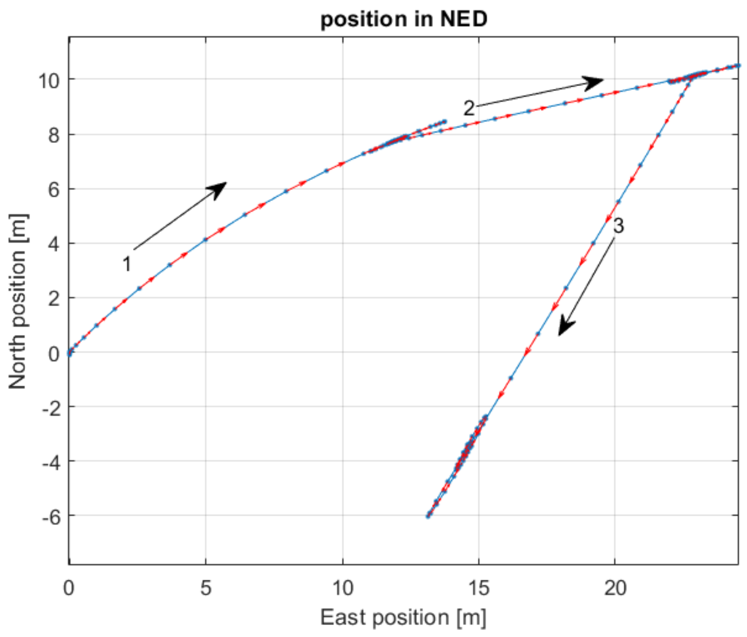


Figure 4.48: Position in NED coordinate system using simple MPC with integral action and system identification

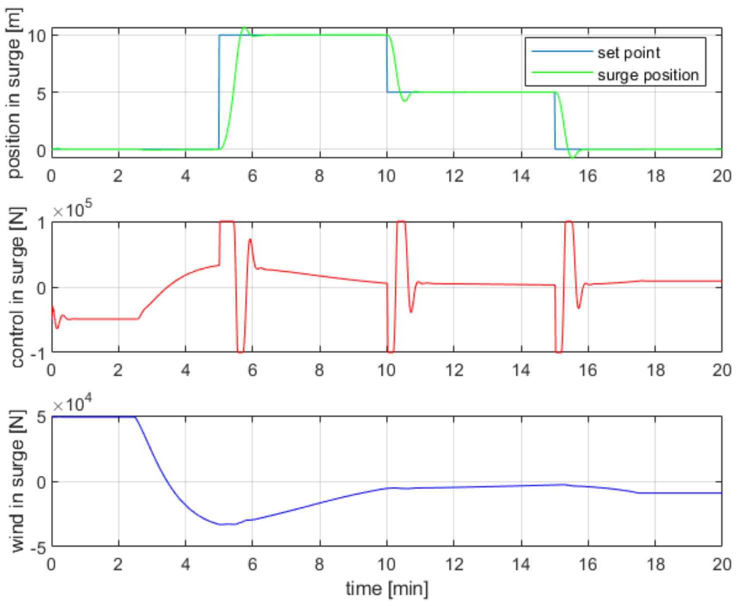


Figure 4.49: Position in surge using LQ with integral action and system identification

Figure 4.50 shows the vessel position in sway direction and how it is developed along the simulation period. Where the set-point is changed three times in sway direction and the

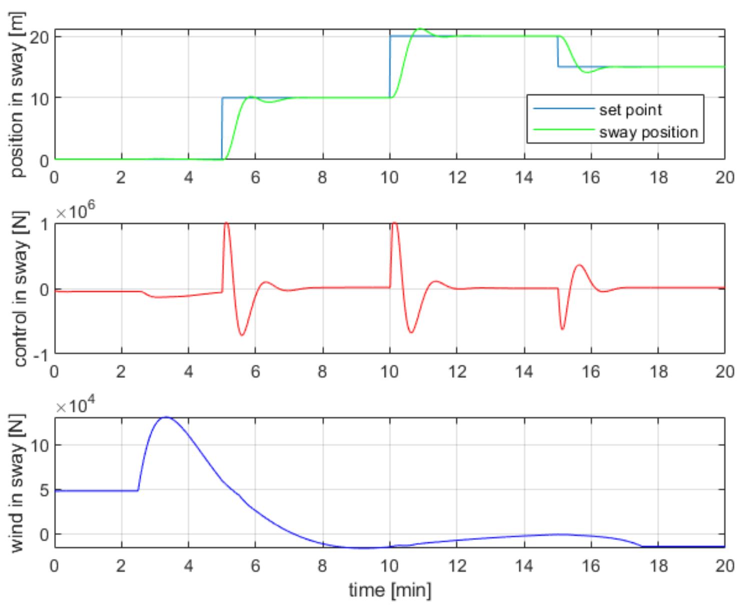


Figure 4.50: Position in sway using LQ with integral action and system identification

control system responds to those changes by generating appropriate force in sway. The generated force moves the vessel to track the set-point despite of the wind force in sway. The generated force in sway direction is shown in the second plot and the wind force in sway is shown in the third plot.

Figure 4.51 shows the development of vessel heading along the simulation period. The control system generates thrusters general moment to rotate the vessel to achieve the desired heading even with the existence of wind disturbance effect in yaw that is shown in the third plot of figure 4.51.

Figure 4.52 shows the vessel position in NED coordinate system. Since the set-point is changed three times along the simulation period, the vessel changed its position three times to track the desired positions. The vessel was at the origin of the NED coordinate system at the simulation starting time. It moves in the direction of the first arrow to the first new desired position that is 10 [m] in surge and 10 [m] in sway. This is equivalent to 7.7 [m] in North and 11.9 [m] in East. Then the vessel moves to the second desired set-point that is 5 [m] in surge and 20 [m] in sway which is equivalent to 1.5 [m] in North and 20.6 [m] in East. After that, the vessel moves the last desired position that is 0 [m] and 15 [m] in sway. This is equivalent to -3.6 [m] in North and 14.6 [m] in East. To convert the position from Body coordinate system (surge, sway) to the NED coordinate system (North, East), the transformation matrix in equation 2.1 is used.

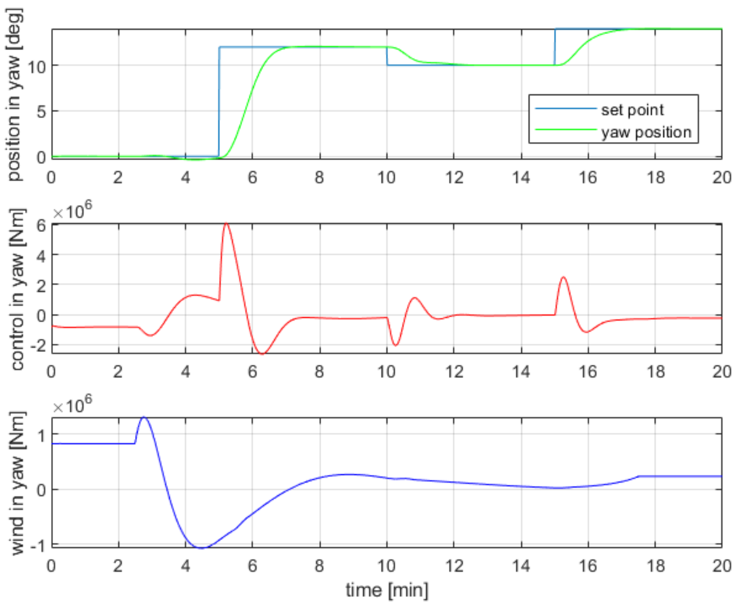


Figure 4.51: Position in yaw using LQ with integral action and system identification

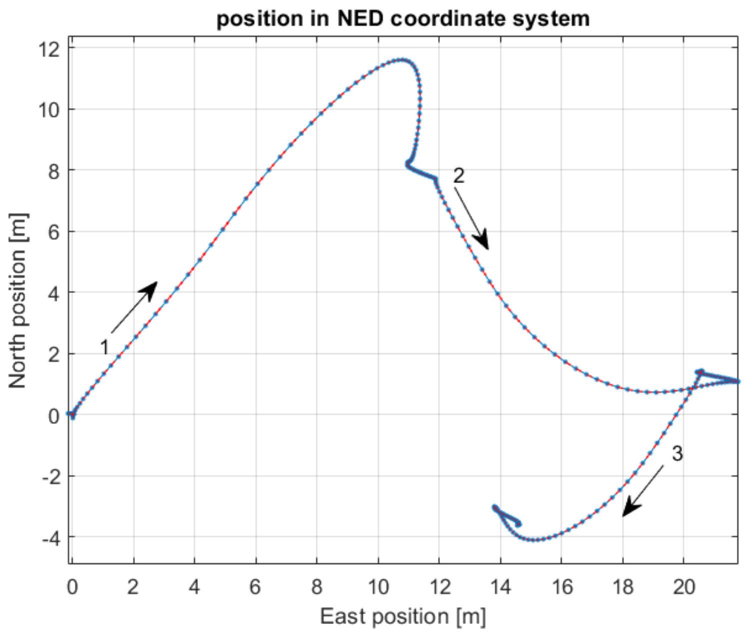


Figure 4.52: Position in NED coordinate system using LQ with integral action and system identification

5 Discussion

The aim of this project is to develop a control subsystem of a Dynamic Positioning system to control vessel position. In addition, it aims to investigate the possibility of developing a model-free MPC to solve the challenges related to accurately defining the mathematical model parameters.

Most of previous research in Dynamic positioning systems use the mathematical model developed by Thor I Fossen [5]. It includes a big number of parameters and needs a deep knowledge about the studied vessel.

In 1980 Balchen derived a simple mathematical model from Newtons' second low to develop a DP system using optimal control and kalman filter. This project reuses a modified version of Balchen model and develops an MPC control system.

The simple MPC with integral action is the most effective control system in controlling vessel position comparing with other tested MPC algorithms in this report. It is the superior one due to the short simulation time and because it does not depends on known or unknown disturbances since it uses state space model in deviation form (integral action).

Simple MPC with integral action is more effective comparing with optimal control with integral action since it deals with feed-forward and updates optimal values in each sampling time including the effect of disturbances on the vessel. In addition, it has smoother change in control signal. MPC with integral action shows fast dynamic response and superior performance.

As mentioned before, any vessel mathematical model needs defining some parameters related to vessel hydrodynamics and geometry. The system identification method is used to solve the challenge of defining an accurate mathematical model. The Deterministic and Stochastic system identification and Realization algorithm is used to get an identified model from input-output data. This model is used to develop a model-free MPC that is implemented and tested. Model-free MPC with integral action was effective in controlling vessel position and tracking the desired set-point under different weather circumstances.

For future work it is suggested that:

- Non-linear MPC can be developed and tested.

- High-frequency part of Balchen model can be included in the model.

- Studying thrusters and propulsion system to include input deviation constraints in MPC algorithm.

- Defining the configuration matrix and thruster coefficient matrix to calculate forces and moment produced by each thruster and not only the generalized thrusters forces and moment.

- Proposed developed model-free simple MPC can be tested on a real vessel.

6 Conclusion

The theory behind vessel movement and the different coordinate systems were dressed. External and internal forces acting on a vessel moving at sea are described. A modified version of Bachen non-linear model is defined and linearized to get a simple version to be implemented in four different versions of MPC. The four MPC versions are: standard MPC, reduced size MPC, simple MPC and simple MPC with integral action. The linearized modified Balchen model is also used in the LQ optimal control with integral action.

All previous mentioned control methods are implemented in MATLAB and tested. The superior control was simple MPC with integral action. It responded effectively to changes in set-point and control the studied vessel position under different wind forces and moment.

The Deterministic and Stochastic system identification and Realization algorithm is proposed. An identified linear state space model is gotten and used in a free model simple MPC with integral action and tested also in optimal control with integral action. This algorithm solves the challenge of defining an accurate mathematical model to be implemented in the model based control algorithms.

Bibliography

[1] J. G. Balchen, N. A. Jenssen, E. Mathisen and S. Sælid, 'A Dynamic Positioning System Based on Kalman Filtering and Optimal Control,' *Modeling, Identification and Control*, vol. 1, no. 3, pp. 135–163, 1980. DOI: 10.4173/mic.1980.3.1.

[2] U. N. C. O. EXPERTISE, 'INTRODUCTION TO DYNAMIC POSITIONING (DP) SYSTEMS,' p. 26, 2019.

[3] 'Dynamic positioning systems.' (2022), [Online]. Available: https://www.kongsberg.com/maritime/products/positioning-and-manoeuvring/dynamic-positioning/.

[4] 'The story behind dynamic positioning.' (2021), [Online]. Available: https://www.kongsberg.com/kmagazine/2014/3/story-behind-dynamic-positioning/.

[5] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* Trondheim (Norway): John Wiley & Sons Ltd, 2021.

[6] 'Thor. I. Fossen and T. Perez (2004). marine systems simulator (mss).' (2021), [Online]. Available: https://github.com/cybergalactic/MSS.

[7] MP-08-21, 'A simulated comparison of controller types for dynamic positioning of naval vessels,' 2022.

[8] 'The four important factors for a ship's windage area calculations.' (2020), [Online]. Available: https://thenavalarch.com/the-four-important-factors-for-windage-area-calculations/.

[9] F. A. Haugen, 'Modeling, simulation and control,' 2021.

[10] R. Sharma, 'Lecture notes for the course IIA 4117: Model Predictive Control,' 2019.

[11] D. D. Ruscio, 'Model predictive control and optimization,' 2019.

[12] D. Di Ruscio, 'Model Predictive Control with Integral Action: A simple MPC algorithm,' *Modeling, Identification and Control*, vol. 34, no. 3, pp. 119–129, 2013. DOI: 10.4173/mic.2013.3.2.

[13] A. K. Tangirala, *PRINCIPLES OF SYSTEM IDENTIFICATION: Theory and Practice.* Boca Raton: Taylor & Francis Group, 2015.

[14] D. D. Ruscio, 'System theory state space analysis and control theory,' 2018.

[15] D. D. Ruscio, 'Optimal model based control: System analysis and design,' 2021.

[16] D. Di Ruscio, 'Discrete LQ optimal control with integral action: A simple controller on incremental form for MIMO systems,' *Modeling, Identification and Control*, vol. 33, no. 2, pp. 35–44, 2012. DOI: `10.4173/mic.2012.2.1`.

[17] '*dlqdu_pi.m*.' (2018), [Online]. Available: `https://web01.usn.no/~davidr/sce4106/ovinger/integrator_ex/matlab/dlqdu_pi.m`.

[18] '*ss2h.m*.' (2018), [Online]. Available: `https://web01.usn.no/~davidr/sce4106/empc_files/ss2h.m`.

[19] '*scmat.m*.' (2018), [Online]. Available: `https://web01.usn.no/~davidr/sce4106/empc_files/scmat.m`.

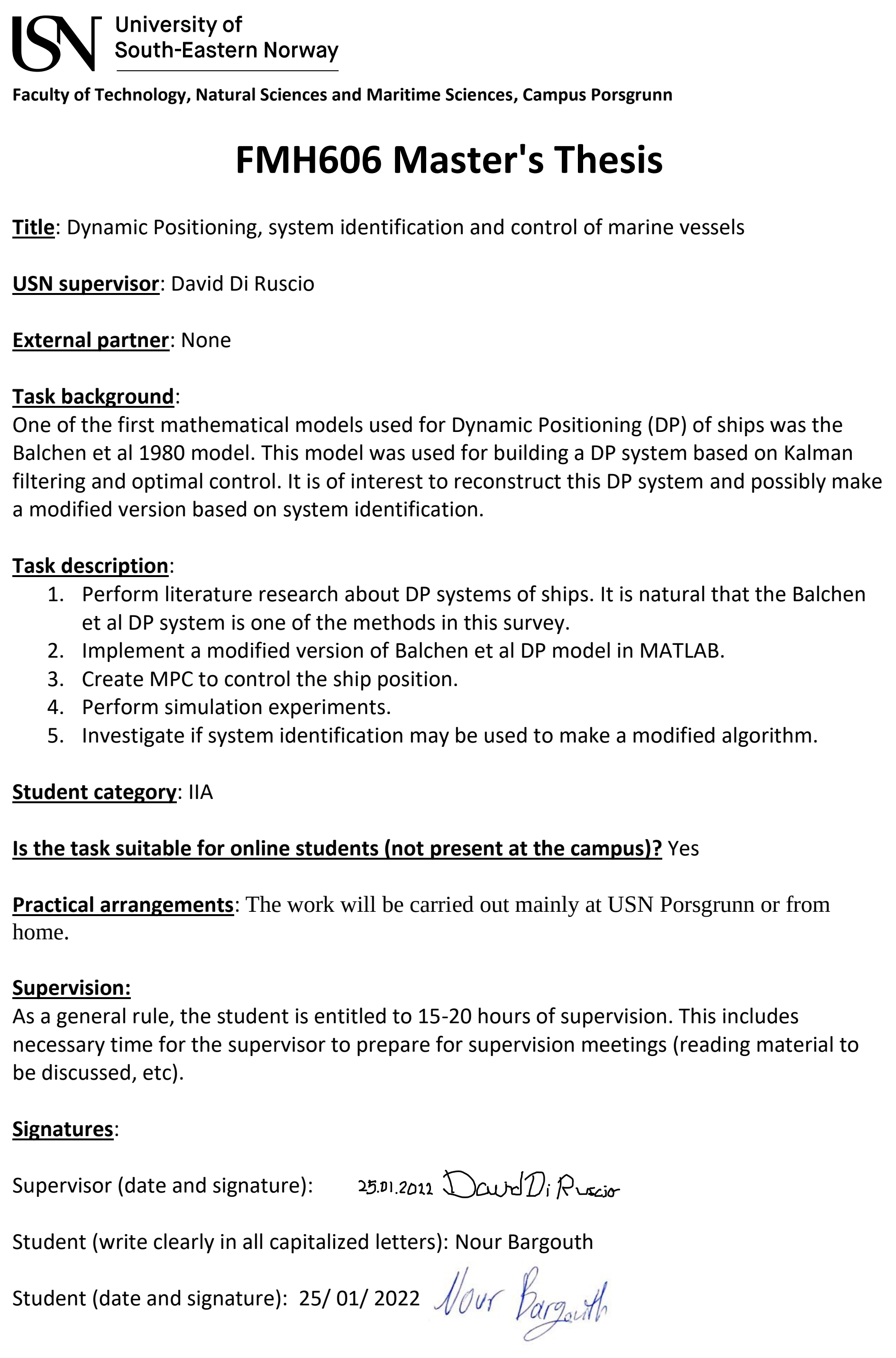
[20] '*prbs1.m*.' (2000), [Online]. Available: `https://davidr.no/iia2217/ovinger/oppg13/prbs1.m`.

[21] D. D. Ruscio, 'Subspace system identification: Theory and applications,' 2022.

[22] '*plot_dir.m*.' (2010), [Online]. Available: `https://se.mathworks.com/matlabcentral/fileexchange/1676-plot-with-direction`.

Appendix A

Task Description



University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Dynamic Positioning, system identification and control of marine vessels

USN supervisor: David Di Ruscio

External partner: None

Task background:
One of the first mathematical models used for Dynamic Positioning (DP) of ships was the Balchen et al 1980 model. This model was used for building a DP system based on Kalman filtering and optimal control. It is of interest to reconstruct this DP system and possibly make a modified version based on system identification.

Task description:
1. Perform literature research about DP systems of ships. It is natural that the Balchen et al DP system is one of the methods in this survey.
2. Implement a modified version of Balchen et al DP model in MATLAB.
3. Create MPC to control the ship position.
4. Perform simulation experiments.
5. Investigate if system identification may be used to make a modified algorithm.

Student category: IIA

Is the task suitable for online students (not present at the campus)? Yes

Practical arrangements: The work will be carried out mainly at USN Porsgrunn or from home.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): 25.01.2022 David Di Ruscio

Student (write clearly in all capitalized letters): Nour Bargouth

Student (date and signature): 25/ 01/ 2022 Nour Bargouth

Appendix B

MATLAB script for implementing Dynamic positioning continuous non-linear model of Balchen

```
%% implementing Dynamic positioning continuous non-linear model of Balchen
function x_dot = nonlinear(x, F_t, F_w, F_c, M, D, neu)
%% definitions:
% x = [x_su, x_sw, psi, v_su, v_sw, v_psi]' vessel position in surge, sway
% and yaw and vessel velocities in surge, sway and yaw
% F_t = [F_t_su, F_t_sw, N_t]' thruster forces in surge and sway and
% thrusters moment in yaw
% F_w = [F_w_su, F_w_sw, N_w]' wind forces in surge and sway and wind
% moment in yaw (Body frame)
% F_c = [v_c_su, v_c_sw, N_c]' water current velocities in surge and sway
% and water current moment in yaw (Body frame)
% D = [d1, d2, d3, d4] drag and moment coefficients
% M = [m1, m2, m3] initial coefficients
% neu = [neu1, neu2, neu3] zero mean gaussian white noise
%% states equations:
dx_su_dt = x(4);
dx_sw_dt = x(5);
dpsi_dt = x(6);
dv_su_dt = - D(1)/M(1)*abs(x(4) - F_c(1))*(x(4) - F_c(1)) + 1/M(1)*(F_w(1)  + F_t↙
(1)) + neu(1);
dv_sw_dt = - D(2)/M(2) * abs(x(5) - F_c(2))*(x(5) - F_c(2)) + 1/M(2)*(F_w(2) + F_t↙
(2)) + neu(2);
dv_psi_dt = -D(3)/M(3)*abs(x(6))*x(6) - D(4)/M(3)*abs(x(5) - F_c(2))*(x(5) - F_c↙
(2)) + 1/M(3)*(F_w(3) + F_t(3) + F_c(3)) + neu(3) ;
x_dot = [dx_su_dt, dx_sw_dt, dpsi_dt, dv_su_dt, dv_sw_dt, dv_psi_dt]';
R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];% rotational matrix to↙
transform from bodey frame to NED frame

end
```

Appendix C

MATLAB script for calculating wind forces and moment

```
% function to calculate wind forces and moment in Body frame coordinate
% system
function F_w = wind_force(V_w, betta, psi,v_su, v_sw, A_F, A_L, L)
%V_w: wind speed [m/s] in NED frame
%betta: wind direction [deg] in NED frame
%psi: vessel heading [rad]
C_x = 0.6;
C_y = 0.8;
C_n = 0.1;
% C_x, C_y, C_n: wind coefficients: assumed to be constants
betta = betta * pi/180; % convert to radian
u_w = V_w * cos(betta - psi); % wind speed in surge
v_w = V_w * sin (betta - psi);% wind speed in sway
u_rw = v_su - u_w; % wind relative speed in surge
v_rw = v_sw - v_w; % wind relative speed in sway
V_r = sqrt(u_rw^2 + v_rw^2);%wind relative speed
rho = 1.23; % wind density [kg\m^3]
F_w_su = 0.5*rho*C_x*A_F*cos(betta)* V_r^2;
F_w_sw = 0.5*rho*C_y*A_L*sin(betta)* V_r^2;
N_w = 0.5*rho*C_n*A_L*L*sin(2*betta)* V_r^2;
F_w = [F_w_su, F_w_sw, N_w]';
end
```

Appendix D

MATLAB script for calculating water current velocity in Body frame

```
% function to calculate water current velocities and moment in Body frame
% to use it in the vessel model to calculate position and velocities
function F_c = current_force(V_c, x)
%V_c: water current velocities and moment in NED frame
R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
F_c = R'*V_c; %tranform from NED to body coordinate system
```

Appendix E

MATLAB script for comparing linear and non-linear models

```
%compare between linear and non-linear model:
clc
clear all
close all
parameters;
%% simulation settings
dt = 10; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 20; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps
%% intialization:
x_non = [0 0 0 0 0 0]';
x = [0,0,0,0,0,0]';
u = [0;0;0];
%% Disturbances:
V_c = rand(3,N)*10;% water current speed in earth coordinate system
V_w = slowly(N)*20;% wind speed
gamma = slowly(N)*360;% wind direction
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1), gamma(1), x(3),x(4), x(5), A_F, A_L, L);
%% Preallocation of arrays for plotting:
x_linear = zeros(N,6);
x_non_linear = zeros(N,6);
U = zeros(N,3);
Fw= zeros(N,3);
time = linspace(0, N*dt, N)';
[A, B, D] =  linear_SSM (F_c, M, Drag,dt);
%% simulation loop
for i = 1:N
    if i ==N/3
        u = [-1e3; 1e3; 1e5];
    end
    F_c = current_force(V_c(:,i),x);
    F_w = wind_force(V_w(i), gamma(i), x(3),x(4), x(5), A_F, A_L, L);
    % storing variables at time instant i
    x_non_linear(i,:) = x_non;
    x_linear(i,:) = x;
    Fw (i,:)= F_w;
    U(i,:) = u;
    % calculating position using linear and non-linear models:
    x_dot_non = nonlinear(x_non, u, F_w, F_c, M, Drag, neu);
    x_non = x_non +x_dot_non *dt;
    x = A*x + B*u + B *F_w;
end

%% plotting:
figure(1)
subplot(5,1,1),plot(time/60, x_non_linear(:,1),time/60, x_linear(:,1)),xlabel('time↙
[min]'),ylabel('surge position [m]'),grid
legend('non-linear model', 'linear model');
subplot(5,1,2),plot(time/60, U(:,1)),ylabel('Control in surge[N]'),grid
subplot(5,1,3),plot(time/60, Fw(:,1)),ylabel('Wind force in surge[N]'),grid
subplot(5,1,4),plot(time/60, V_w),ylabel('Wind velocity[m/s]'),grid
subplot(5,1,5),plot(time/60, gamma),xlabel('time [min]'),ylabel('Wind direction↙
[deg]'),grid

figure(2)
subplot(5,1,1),plot(time/60, x_non_linear(:,2),time/60, x_linear(:,2)),xlabel('time↙
[min]'),ylabel('sway position [m]'),grid
legend('non-linear model', 'linear model');
subplot(5,1,2),plot(time/60, U(:,2)),ylabel('Control in sway[N]'),grid
subplot(5,1,3),plot(time/60, Fw(:,2)),ylabel('Wind force in sway[N]'),grid
subplot(5,1,4),plot(time/60, V_w),ylabel('Wind velocity[m/s]'),grid
subplot(5,1,5),plot(time/60, gamma),xlabel('time [min]'),ylabel('Wind direction↙
[deg]'),grid

figure(3)
subplot(5,1,1),plot(time/60, x_non_linear(:,3)*180/pi,time/60, x_linear(:,3) ↙
*180/pi),xlabel('time [min]'),ylabel(' position in yaw[deg]'),grid
legend('non-linear model', 'linear model');
subplot(5,1,2),plot(time/60, U(:,3)),ylabel('control moment in yaw[Nm]'),grid
subplot(5,1,3),plot(time/60, Fw(:,3)),xlabel('time [min]'),ylabel('Wind moment in↙
yaw[Nm]'),grid
subplot(5,1,4),plot(time/60, V_w),ylabel('Wind velocity [m/s]'),grid
subplot(5,1,5),plot(time/60, gamma),xlabel('time [min]'),ylabel('Wind direction↙
[deg]'),grid
```

Appendix F

MATLAB script for slowly.m

```
% slowly function creates N slowly changed variables
function A = slowly(N)
A = zeros (N,1);
a = rand(4,1);
for i = 1:N
    if i <= N/4
        A(i,1) = a(1,1);
    elseif i >N/4 && i <= N/2
        A(i,1) = a(2,1);
    elseif i >N/2 && i <= 3*N/4
        A(i,1) = a(3,1);
    else
        A(i,1) = a(4,1);
    end
end
A = smooth(A,0.5);
```

Appendix G

MATLAB script for standard MPC algorithm

```
function [u_opt, x_opt,e_opt, y_opt] = optimal_standard(A,B,D,r,W,x0,N)
%size of states,outputs and inputs:
nx = 6; ny = 3; nu = 3;
%size of the unknow vector z
nz = N*(nx + nu + 2*ny);
%weighting matrices
Q = [1e4 0 0; 0 1e4 0; 0 0 1e8]; %tuning weight for errors
P = [1e-5 0 0; 0 1e-6 0; 0 0 1e-15] ;%tuning weight for inputs
%% build matrices
%% matrices in the standard form
H11 = kron(eye(N),P);
H22 = zeros(N*nx,N*nx);
H33 = kron(eye(N),Q);
H44 = zeros(N*ny,N*ny);
H = blkdiag(H11,H22,H33,H44);
c = zeros(nz,1);
%% matrices in the equality constraints
%% from equation: x_kp1 = A x_k + B u_k
Ae1u = -kron(eye(N),B);
Ae1x = eye(N*nx)-kron(diag(ones(N-abs(-1),1),-1),A);
Ae1e = zeros(N*nx,N*ny);
Ae1y = zeros(N*nx,N*ny);
Fw = W(:,2:N);
Fw = Fw(:);
be1 = [A*x0 + B*W(:,1);kron(eye(N-1),B)*Fw];
%be1 = [A*x0;zeros((N-1)*nx,1)];
%% from equation: y_k = D x_k
Ae2u = zeros(N*ny,N*nu);
Ae2x = -kron(eye(N),D);
Ae2e = zeros(N*ny,N*ny);
Ae2y = eye(N*ny);
be2 = zeros(N*ny,1);
%% from equation: e_k = y_k - r_k
Ae3u = zeros(N*ny,N*nu);
Ae3x = zeros(N*ny,N*nx);
Ae3e = eye(N*ny);
Ae3y = eye(N*ny);
be3 = r(:);
Ae=[Ae1u Ae1x Ae1e Ae1y;
     Ae2u Ae2x Ae2e Ae2y;
     Ae3u Ae3x Ae3e Ae3y];
be=[be1;be2;be3];
%% bounds (not specified so assume between -inf to +inf)
ZL=(-Inf*ones(nz,1));
ZU=(Inf*ones(nz,1));
%% solving the QP
options = optimoptions('quadprog','Display','off');
z_opt = quadprog(H,c,[],[],Ae,be,ZL,ZU,x0,options);
%% extract results
Ua = z_opt(1+N*(0) :N*(nu),1); %control inputs
Xa = z_opt (1+N*(nu) :N*(nu+nx),:); %states
Ea = z_opt (1+N*(nu+nx) :N*(nu+nx+ny),:); %error in tracking
Ya = z_opt (1+N*(nu+nx+ny) :N*(nu+nx+ny+ny),:); %outputs
%rearrange the data
u_opt = reshape(Ua,nu,N); %arranged control inputs
x_opt = reshape(Xa,nx,N); %arranged states
e_opt = reshape(Ea,ny,N); %arranged errors
y_opt = reshape(Ya,ny,N); %arranged outputs
```

Appendix H

MATLAB script for simulating standard MPC

```
% Using standard MPC to control DP system
clc
clear all
close all
parameters;% to call parameters file
%% simulation settings
dt = 10; % time step [sec]
t_start = 0; % simulation starting time [ min]
t_stop = 20; %simulation stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulation steps
L = 10;%prediction horizon
%% setpoint
sp = zeros(3, (N+L));
for m = 1:N+L
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [4 ; 4 ; 14*pi/180];
   elseif m>N/2
        sp(:,m)= [3 ; 8; 14*pi/180];
   end
end
%% intialization:
x_mod = [sp(:,1); 0; 0; 0];% initial state is chosen to be at rest ant at the↙
desired initial setpoint
x_est= x_mod;
%% Disturbances:
V_c = rand(3,(N+L))*10;% water current speed in NED coordinate system as a random↙
number
V_w = slowly((N+L))*20;% wind speed in NED coordinate system as a random number↙
between 0 and 20 [m/sec]
gamma = slowly((N+L))*360;% wind direction in NED coordinate system  as a random↙
number between 0 and 360 [deg]
F_c = current_force(V_c(:,1),x_mod); % water current velocities in Body frame at↙
initial time
F_w = wind_force(V_w(1), gamma(1), x_mod(3),x_mod(4), x_mod(5), A_F, A_L, Long);%↙
wind forces and moment in Body frame at initial time
u = -F_w;
%% Preallocation of arrays for plotting:
position = zeros(3,N);
position_NED = zeros(3,N);
u_array= zeros(3,N);
W = zeros(3,N);
time = linspace(0, N*dt, N)';
%% calculating discrete time linear State space model matrices:
[A, B, D] = linear_SSM(F_c, M, Drag,dt);
%% calculating kalman filter gain:
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e2,1e2,1e2]);
K=dlqe(A,G,D,Q_k,R_k);
%% simulation loop:
for k = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x_mod(3)) -sin(x_mod(3)) 0; sin(x_mod(3))  cos(x_mod(3)) 0; 0 0 1];
    y = D*x_mod;
    r = sp(:,k+1:L+k);
    % disturbances
    for n= 1:L
        Fw(:,n) = wind_force(V_w(n+k-1), gamma(n+k-1), x_mod(3),x_mod(4), x_mod(5),↙
A_F, A_L, Long);
    end
    F_c = current_force(V_c(:,k),x_mod);
    % save vaiables for plotting
    position(:,k) = y;
    position_NED(:,k) = R*y;
    u_array(:,k) = u;
    W(:,k) = wind_force(V_w(k), gamma(k), x_mod(3),x_mod(4), x_mod(5), A_F, A_L,↙
Long);
    % solve optimization problem along the prediction horizon L with known↙
setpoints and wind forces along L
    [u_opt, x_opt] = optimal_standard(A,B,D,r,Fw,x_mod,L);% assuming measurable↙
states
    %[u_opt, x_opt] = optimal_standard(A,B,D,r,Fw,x_est,L);% using states estimates↙
from Kalman filter
    u = u_opt(:,1);
    % updating the system by applying the first calculated optimal control
     dx = nonlinear(x_mod, u, W(:,k), F_c, M, Drag, neu);
     x_mod = x_mod + dt* dx;
     % using current time output to estimate states by kalman filter
     x_est = A*x_est + B*u + B*W(:,k) + K*(y - D*x_est);
end
%% plotting results
figure(1),
subplot(311);plot(time/60,sp(1,1:N),time/60,position(1,:),'g-');ylabel('position in↙
surge [m]');
legend('set point', 'surge position');grid
title('Surge direction');
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');grid
subplot(313);plot(time/60,W(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,1:N),time/60,position(2,:),'g-');ylabel('position in↙
sway [m]');
legend('set point', 'sway position');grid
title('Sway direction');
subplot(312);plot(time/60,u_array(2,:),'r-'); ylabel('control in sway [N]');grid
subplot(313);plot(time/60,W(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,1:N)*180/pi,time/60,position(3,:)*180/pi,'g-');↙
ylabel('position in yaw [deg]');
legend('set point', 'yaw position');grid
title('Yaw direction');
subplot(312);plot(time/60,u_array(3,:),'r-'); ylabel('control in yaw [Nm]');grid
subplot(313);plot(time/60,W(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment in↙
yaw [Nm]');grid
figure(4);grid
plot_dir(position_NED(2,:)',position_NED(1,:)');xlabel('Position in East [m]');↙
ylabel('Position in North [m]');%Copyright (c) 2010, Kangwon Lee%All rights
title('position in NED coordinate system');
```

Appendix I

MATLAB script for reduced MPC algorithm

```
function [u_opt, x_opt] = optimal_reduced(A,B,D,r,W,x0,N)
%size of states,and inputs:
nx = 6; nu = 3;
%size of the unknow vector z
nz = N*(nx + nu);%N*6
%weighting matrices
Q = [5e2 0 0; 0 1e3 0; 0 0 1e8]; %tuning weight for errors
Q_tilde =D'*Q*D;
P = [1e-6 0 0; 0 1e-7 0; 0 0 1e-16] ;%tuning weight for inputs
%% build matrices
%% matrices in the standard form
H11 = kron(eye(N),P);
H22 = kron(eye(N),Q_tilde);
H = blkdiag(H11,H22);
r = r(:);
c = [zeros(N*nu,1);kron(eye(N),-D'*Q)*r];
%% matrices in the equality constraints
%% from equation: x_kp1 = A x_k + B u_k
Ae1u = -kron(eye(N),B);
Ae1x = eye(N*nx)-kron(diag(ones(N-abs(-1),1),-1),A);
Fw = W(:,2:N);
Fw = Fw(:);
be1 = [A*x0+ B*W(:,1);kron(eye(N-1),B)*Fw];
%%
Ae=[Ae1u Ae1x];
be=be1;
%% bounds (not specified so assume between -inf to +inf)
ZL=(-Inf*ones(nz,1));
ZU=(Inf*ones(nz,1));
%% solving the QP
options = optimoptions('quadprog','Display','off');
z_opt = quadprog(H,c,[],[],Ae,be,ZL,ZU,x0,options);
%% extract results
Ua = z_opt(1+N*(0) :N*(nu),1); %control inputs
Xa = z_opt (1+N*(nu) :N*(nx+nu),:); %states
%rearrange the data
u_opt = reshape(Ua,nu,N); %arranged control inputs
x_opt = reshape(Xa,nx,N); %arranged states
end
```

Appendix J

MATLAB script for simulating reduced size MPC

```
% Using reduced size MPC to control DP system
clc
clear all
close all
parameters;% to call parameters file
%% simulation settings
dt = 10; % time step [sec]
t_start = 0; % simulation starting time [ min]
t_stop = 20; %simulation stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulation steps
L = 10;%prediction horizon
%% setpoint
sp = zeros(3, (N+L));
for m = 1:N+L
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [4 ; 4 ; 14*pi/180];
   elseif m>N/2
        sp(:,m)= [3 ; 8; 14*pi/180];
   end
end
%% intialization:
x_mod = [sp(:,1); 0; 0; 0];% initial state is chosen to be at rest ant at the↙
desired initial setpoint
x_est= x_mod;
%% Disturbances:
V_c = rand(3,(N+L))*10;% water current speed in NED coordinate system as a random↙
number
V_w = slowly((N+L))*20;% wind speed in NED coordinate system as a random number↙
between 0 and 20 [m/sec]
gamma = slowly((N+L))*40;% wind direction in NED coordinate system  as a random↙
number between 0 and 360 [deg]
F_c = current_force(V_c(:,1),x_mod); % water current velocities in Body frame at↙
initial time
F_w = wind_force(V_w(1), gamma(1), x_mod(3),x_mod(4), x_mod(5), A_F, A_L, Long);%↙
wind forces and moment in Body frame at initial time
u = -F_w;
%% Preallocation of arrays for plotting:
position = zeros(3,N);
position_NED = zeros(3,N);
u_array= zeros(3,N);
W = zeros(3,N);
time = linspace(0, N*dt, N)';
%% calculating discrete time linear State space model matrices:
[A, B, D] = linear_SSM(F_c, M, Drag,dt);
%% calculating kalman filter gain:
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e2,1e2,1e2]);
K=dlqe(A,G,D,Q_k,R_k);
%% simulation loop:
for k = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x_mod(3)) -sin(x_mod(3)) 0; sin(x_mod(3))  cos(x_mod(3)) 0; 0 0 1];
    y = D*x_mod;
    r = sp(:,k+1:L+k);
    % disturbances
    for n= 1:L
        Fw(:,n) = wind_force(V_w(n+k-1), gamma(n+k-1), x_mod(3),x_mod(4), x_mod(5),↙
A_F, A_L, Long);
    end
    F_c = current_force(V_c(:,k),x_mod);
    % save vaiables for plotting
    position(:,k) = y;
    position_NED(:,k) = R*y;
    u_array(:,k) = u;
    W(:,k) = wind_force(V_w(k), gamma(k), x_mod(3),x_mod(4), x_mod(5), A_F, A_L,↙
Long);
    % solve optimization problem along the prediction horizon L with known↙
setpoints and wind forces along L
    [u_opt, x_opt] = optimal_reduced(A,B,D,r,Fw,x_mod,L);% assuming measurable↙
states
    %[u_opt, x_opt] = optimal_reduced(A,B,D,r,Fw,x_est,L);% using states estimates↙
from Kalman filter
    u = u_opt(:,1);
    % updating the system by applying the first calculated optimal control
     dx = nonlinear(x_mod, u, W(:,k), F_c, M, Drag, neu);
     x_mod = x_mod + dt* dx;
     % using current time output to estimate states by kalman filter
     x_est = A*x_est + B*u + B*W(:,k) + K*(y - D*x_est);
end
%% plotting results
figure(1),
subplot(311);plot(time/60,sp(1,1:N),time/60,position(1,:),'g-');ylabel('position in↙
surge [m]');
legend('set point', 'surge position');grid
title('Surge direction');
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');grid
subplot(313);plot(time/60,W(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,1:N),time/60,position(2,:),'g-');ylabel('position in↙
sway [m]');
legend('set point', 'sway position');grid
title('Sway direction');
subplot(312);plot(time/60,u_array(2,:),'r-'); ylabel('control in sway [N]');grid
subplot(313);plot(time/60,W(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,1:N)*180/pi,time/60,position(3,:)*180/pi,'g-');↙
ylabel('position in yaw [deg]');
legend('set point', 'yaw position');grid
title('Yaw direction');
subplot(312);plot(time/60,u_array(3,:),'r-'); ylabel('control in yaw [Nm]');grid
subplot(313);plot(time/60,W(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment in↙
yaw [Nm]');grid
figure(4);grid
plot_dir(position(2,:)',position(1,:)');xlabel('Position in East [m]'); ylabel↙
('Position in North [m]');%Copyright (c) 2010, Kangwon Lee%All rights
title('position in NED coordinate system');
```

Appendix K

MATLAB script for simulating LQ optimal control with integral action

```
clc
clear all
close all
parameters;

%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 20; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps

%% setpoint
sp = zeros(3, (N));
for m = 1:N
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [15 ; 20 ; 14*pi/180];
   elseif m> N/2
        sp(:,m)= [8 ; 6; 6*pi/180];
   end
end
%% Disturbances:
V_c = rand(3,N)*20;% water current speed in earth coordinate system as a random↙
number
V_w = slowly(N)*20;% wind speed as a random number
gamma = slowly(N)*360;% wind direction as a random number

%% initialization:
x0 = [sp(1,1) sp(2,1) sp(3,1) 0 0 0]';
x = x0;
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1,1), gamma(1,1), x(3),x(4), x(5), A_F, A_L, L);
u = -F_w;
%% Preallocation of arrays for plotting:
y_array = zeros(3,N);
u_array= zeros(3,N);
Fw = zeros(3,N);
time = linspace(0, N*dt, N)';
%% state space model system matrices
[ A, B, D] = linear_SSM(F_c, M, Drag,dt);
%% calculating control gaing
Q = [1e2 0 0; 0 1e0 0; 0 0 1e3]; %tuning weight for errors
P = [1e-7 0 0; 0 1e-7 0; 0 0 1e-10] ;%tuning weight for inputs
[G1,G2]= dlqdu_pi(A,B,D,Q,P);
%% kalman filter gain
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e3,1e3,1e3]);
K=dlqe(A,G,D,Q_k,R_k);
%% simulation loop:
x_est = x;
y = D*x;
x_old = x;
y_old = y;
for i = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
    % disturbances
    F_c = current_force(V_c(:,i),x);
    F_w = wind_force(V_w(i,1), gamma(i,1), x(3),x(4), x(5), A_F, A_L, L);
    y = D*x;
    y_array(:,i) = y;
    y_NED(:,i)= R*y;
    u_array(:,i) = u;
    Fw(:,i) = F_w;
    u = u + G1*(x_est-x_old) + G2*(y_old - sp(:,i));
    x_old = x_est;
    y_old = y;
    dx = nonlinear(x, u, F_w, F_c, M, Drag, neu);
    x = x + dt*dx;
    x_est = A*x_est + B*u + B*F_w+ K*(y- D*x_est);
end
%% plotting
figure(1),
subplot(311);plot(time/60,sp(1,:),time/60,y_array(1,:),'g-');ylabel('position in↙
surge [m]');
title('Surge direction');legend('set point','surge position');grid
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');grid
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,:),time/60,y_array(2,:),'g-');ylabel('position in↙
sway [m]');
title('Sway direction');legend('set point','sway position');grid
subplot(312);plot(time/60,u_array(2,:),'r-');xlabel('time [min]'); ylabel('control↙
in sway [N]');grid
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,:)*180/pi,time/60,y_array(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');
title('Yaw direction');legend('set point','Yaw position');grid
subplot(312);plot(time/60,u_array(3,:),'r-');xlabel('time [min]'); ylabel('control↙
in yaw [N]');grid
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment↙
in yaw [Nm]');grid
figure(4);grid
plot_dir(y_NED(2,:)',y_NED(1,:)');xlabel('East position [m]'); ylabel('North↙
position [m]');
title('position in horizontal plane');
```

Appendix L

MATLAB script for simulating LQ optimal control with integral action with constraints

```
clc
clear all
close all
parameters;

%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 20; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps

%% setpoint
sp = zeros(3, (N));
for m = 1:N
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [15 ; 20 ; 14*pi/180];
   elseif m> N/2
        sp(:,m)= [8 ; 6; 6*pi/180];
   end
end
%% Disturbances:
V_c = rand(3,N)*20;% water current speed in earth coordinate system as a random↙
number
V_w = slowly(N)*20;% wind speed as a random number
gamma = slowly(N)*360;% wind direction as a random number

%% initialization:
x0 = [sp(1,1) sp(2,1) sp(3,1) 0 0 0]';
x = x0;
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1,1), gamma(1,1), x(3),x(4), x(5), A_F, A_L, L);
u = -F_w;
%% Preallocation of arrays for plotting:
y_array = zeros(3,N);
u_array= zeros(3,N);
Fw = zeros(3,N);
time = linspace(0, N*dt, N)';
%% state space model system matrices
[ A, B, D] = linear_SSM(F_c, M, Drag,dt);
%% calculating control gaing
Q = [1e2 0 0; 0 1e0 0; 0 0 1e3]; %tuning weight for errors
P = [1e-7 0 0; 0 1e-7 0; 0 0 1e-10] ;%tuning weight for inputs
[G1,G2]= dlqdu_pi(A,B,D,Q,P);
%% kalman filter gain
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e3,1e3,1e3]);
K=dlqe(A,G,D,Q_k,R_k);
%% control limits
u_min = [-1e5; -1e5; -1e7];
u_max = [1e5; 1e5; 1e7];
% u_min = [-inf; -inf; -inf];
% u_max = [inf; inf; inf];
%% simulation loop:
x_est = x;
v = D*x;
```

```
x_old = x;
y_old = y;
for i = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
    % disturbances
    F_c = current_force(V_c(:,i),x);
    F_w = wind_force(V_w(i,1), gamma(i,1), x(3),x(4), x(5), A_F, A_L, L);
    y = D*x;
    y_array(:,i) = y;
    y_NED(:,i)= R*y;
    u_array(:,i) = u;
    Fw(:,i) = F_w;
    u = u + G1*(x_est-x_old) + G2*(y_old - sp(:,i));
        %% limit control
    if u(1)<u_min(1)
        u(1) = u_min(1);
    elseif u(1)>u_max(1)
        u(1) = u_max(1);
    end
    if u(2)<u_min(2)
        u(2) = u_min(2);
    elseif u(2)>u_max(2)
        u(2) = u_max(2);
    end
    if u(3)<u_min(3)
        u(3) = u_min(3);
    elseif u(3)>u_max(3)
        u(3) = u_max(3);
    end
    x_old = x_est;
    y_old = y;
    dx = nonlinear(x, u, F_w, F_c, M, Drag, neu);
    x = x + dt*dx;
    x_est = A*x_est + B*u + B*F_w+ K*(y- D*x_est);
end
%% plotting
figure(1),
subplot(311);plot(time/60,sp(1,:),time/60,y_array(1,:),'g-');ylabel('position in↙
surge [m]');
title('Surge direction');legend('set point','surge position');grid
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');grid
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,:),time/60,y_array(2,:),'g-');ylabel('position in↙
sway [m]');
title('Sway direction');legend('set point','sway position');grid
subplot(312);plot(time/60,u_array(2,:),'r-');xlabel('time [min]'); ylabel('control↙
in sway [N]');grid
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,:)*180/pi,time/60,y_array(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');
title('Yaw direction');legend('set point','Yaw position');grid
subplot(312);plot(time/60,u_array(3,:),'r-');xlabel('time [min]'); ylabel('control↙
in yaw [N]');grid
```

```
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment↙
in yaw [Nm]');grid
figure(4);grid
plot_dir(y_NED(2,:)',y_NED(1,:)');xlabel('East position [m]'); ylabel('North↙
position [m]');
title('position in horizontal plane');
```

Appendix M

MATLAB script for simulating simple MPC

```
%% implementing simple MPC developed by David Di Ruscio in a DP system
clc
clear all
close all
parameters;

%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 15; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps
L =10;% prediction horizon
%% setpoint
sp = zeros(3, (N+L));
for m = 1:N+L
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [3 ; 5 ; 3*pi/180];
   elseif m>N/2
        sp(:,m)= [1 ; 8; 9*pi/180];
   end
end
%% Disturbances:
V_c = rand(3,N)*10;% water current speed in NED coordinate system as a random ↙
number
V_w = slowly(N)* 20;% wind speed in NED coordinate system as a random number
gamma = slowly(N)*360;% wind direction in NED coordinate system as a random number
%% initialization:
x0 = [sp(1,1) sp(1,2) sp(1,3) 0 0 0]';
x = x0;
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1), gamma(1), x(3),x(4), x(5), A_F, A_L, Long);
u = -F_w;
%% Preallocation of arrays for plotting:
y_pos = zeros(3,N);
u_array= zeros(3,N);
Fw = zeros(3,N);
y_NED= zeros(3,N);
time = 1:N;
%% calculating linear SSM matrices
[ A, B, D] = linear_SSM ( F_c, M, Drag,dt);
%% calculating kalman filter gain:
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e2,1e2,1e2]);
K=dlqe(A,G,D,Q_k,R_k);
%%
Q = [1e3 0 0; 0 1e3 0; 0 0 1e9]; %tuning weight for errors
P = [1e-10 0 0; 0 1e-10 0; 0 0 1e-9] ;%tuning weight for inputs
[HdL,OL,OLB]=ss2h(A,B,D,zeros(3,3),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(P,L);
H=FL'*Qt*FL+Rt;
%%
x_est = x;
%% simulation loop:
```

```
for k = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
    F_c = current_force(V_c(:,k),x);
    F_w = wind_force(V_w(k), gamma(k), x(3),x(4), x(5), A_F, A_L, Long);
    y = D*x;
    y_pos(:,k) = y;
    y_NED(:,k) = R*y;
    u_array(:,k) = u(:,1);
    Fw(:,k) = F_w;
    pL=OL*A*x_est;
    r = sp(:,k+1:L+k);
    r = r(:);
    f=FL'*Qt*(pL-r);
    u = quadprog(H,f);
    u = reshape(u,3,L); %arranged control inputs
    dx = nonlinear(x, u(:,1), F_w, F_c, M, Drag,neu);% update the system by using↙
first calculated optimal control
    x= x+ dt*dx;
    % using current time output to estimate states by kalman filter
     x_est = A*x_est + B*u(:,1)+B*F_w+ K*(y - D*x_est);
end
%% plotting
figure(1),
subplot(311);plot(time/60,sp(1,1:N),time/60,y_pos(1,:),'g-');ylabel('position in↙
surge [m]');
title('Surge direction');legend('set point','surge position');grid
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');grid
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,1:N),time/60,y_pos(2,:),'g-');ylabel('position in↙
sway [m]');
title('Sway direction');legend('set point','sway position');grid
subplot(312);plot(time/60,u_array(2,:),'r-'); ylabel('control in sway [N]');grid
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,1:N)*180/pi,time/60,y_pos(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');
title('Yaw direction');legend('set point','Yaw position');grid
subplot(312);plot(time/60,u_array(3,:),'r-'); ylabel('control in yaw [N]');grid
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment↙
in yaw [Nm]');grid
figure(4);grid
plot_dir(y_NED(2,:)',y_NED(1,:)');xlabel('East position [m]'); ylabel('North↙
position [m]');%Copyright (c) 2010, Kangwon Lee%All rights
title('position in NED');
```

Appendix N

MATLAB script for simulating simple MPC with integral action

```
%% implementing simple MPC with integral action developed by David Di Ruscio in a↙
DP control system
clc
clear all
close all
parameters;
%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 15; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps
L =10;% prediction horizon
%% setpoint
sp = zeros(3, (N+L));
for m = 1:N+L
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [3 ; 5 ; 3*pi/180];
   elseif m>N/2
        sp(:,m)= [1 ; 8; 9*pi/180];
   end
end
%% Disturbances:
V_c = rand(3,N)*10;% water current speed in NED coordinate system as a random↙
number
V_w = slowly(N)* 20;% wind speed in NED coordinate system as a random number
gamma = slowly(N)*360;% wind direction in NED coordinate system as a random number
%% initialization:
x0 = [sp(1,1) sp(1,2) sp(1,3) 0 0 0]';
x = x0;
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1), gamma(1), x(3),x(4), x(5), A_F, A_L, Long);
u = -F_w;
%% Preallocation of arrays for plotting:
x_pos = zeros(6,N);
u_array= zeros(3,N);
Fw = zeros(3,N);
y_array = zeros(3,N);
time = 1:N;
%% Calculating linear SSM matrices
[ A, B, D] = linear_SSM ( F_c, M, Drag,dt);
nx = size(A,1); % number of system states
nu = size(B,2);% number of system inputs
ny = size(D,1); % number of system outputs
%% calculating kalman filter gain:
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e2,1e2,1e2]);
K=dlqe(A,G,D,Q_k,R_k);
%%
Q = [1e2 0 0; 0 1e2 0; 0 0 1e9]; %tuning weight for errors
P = [1e-9 0 0; 0 1e-10 0; 0 0 1e-9] ;%tuning weight for inputs
%% augmented system matrices:
At = [A zeros(nx,ny); D eye(ny,ny)];
Bt= [B ; zeros(ny,nu)];
Dt = [D eye(ny,ny)];
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
```

```
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(P,L);
H=FL'*Qt*FL+Rt;
%%
y = D*x;
x_est = x;
x_old = x_est;
y_old = y;
xt = [x-x_old;y_old];
%% simulation loop:
for k = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
    F_c = current_force(V_c(:,k),x);
    F_w = wind_force(V_w(k), gamma(k), x(3),x(4), x(5), A_F, A_L, Long);
    y = D*x;
    x_pos(:,k) = x;
    u_array(:,k) = u(:,1);
    y_array(:,k) = R*y;
    Fw(:,k) = F_w;
    pL=OL*At*xt;
    r = sp(:,k+1:L+k);
    r = r(:);
    f=FL'*Qt*(pL-r);
    du = quadprog(H,f);
    du = reshape(du,3,L); %arranged control inputs
    u = u + du(:,1);
    x_old = x_est;
    y_old = y;
    %dx = nonlinear(x,u, F_w, F_c, M, Drag, [rand(1)*0.05;rand(1)*0.01;rand(1)*0.↙
001]);
    dx = nonlinear(x,u, F_w, F_c, M, Drag, neu);
    x= x+ dt*dx;
    % using current time output to estimate states by kalman filter
    x_est = A*x_est + B*u(:,1)+B*F_w+ K*(y - D*x_est);
    xt = [x_est-x_old;y_old];
end
%% plotting
figure(1),
subplot(311);plot(time/60,sp(1,1:N),time/60,x_pos(1,:),'g-');ylabel('position in↙
surge [m]');
title('Surge direction');
legend('set point','surge position');
grid
subplot(312);plot(time/60,u_array(1,:),'r-');xlabel('time [min]'); ylabel('control↙
in surge [N]');
title('control in Surge direction');
grid
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,1:N),time/60,x_pos(2,:),'g-');ylabel('position in↙
sway [m]');
title('Sway direction');
legend('set point','sway position');
grid
subplot(312);plot(time/60,u_array(2,:),'r-');xlabel('time [min]'); ylabel('control↙
in sway [N]');
title('control in Sway direction');
grid
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,1:N)*180/pi,time/60,x_pos(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');
title('Yaw direction');
legend('set point','Yaw position');
grid
subplot(312);plot(time/60,u_array(3,:),'r-');xlabel('time [min]'); ylabel('control↙
in yaw [N]');
title('control in Yaw direction');
grid
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment↙
in yaw [Nm]');grid
figure(4);grid
plot_dir(y_array(2,:)',y_array(1,:)');xlabel('East position [m]'); ylabel('North↙
position [m]');%Copyright (c) 2010, Kangwon Lee%All rights
title('position in NED');
```

Appendix O

MATLAB script for experiments to collect input-output data

```
%% do closed loop simulation to collect data
clc
clear all
close all
parameters;
%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 120; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps

%% setpointset
sp = [ones(1, N)*2;ones(1, N)*2;ones(1, N)*6*pi/180];
sp_val = [ones(1, N)*2;ones(1, N)*2;ones(1, N)*6*pi/180];
%% Disturbances:
V_c = rand(3,N)*10;% water current speed in earth coordinate system as a random↙
number
V_w = slowly(N)* 5;% wind velocity in NED coordinate system as a random number↙
between 0 and 10[m/s]
gamma = slowly(N)*360;% wind direction in NED coordinate system as a random number↙
between 0 and 360[deg]
%% initialization:
x = [sp(1,1) sp(2,1) sp(3,1) 0 0 0]';
x_val = [sp_val(1,1) sp_val(2,1) sp_val(3,1) 0 0 0]';
F_c = current_force(V_c(:,1),x);%calculate current force in Body frame
F_w = wind_force(V_w(1), gamma(1), x(3),x(4), x(5), A_F, A_L, Long);%calculate wind↙
force in Body frame
F_w_val = wind_force(V_w(1), gamma(1), x_val(3),x_val(4), x_val(5), A_F, A_L, ↙
Long);%calculate wind force in Body frame
u = -F_w;
u_val = -F_w_val;
%% Preallocation of arrays for plotting:
u_array= zeros(3,N);
y_array= zeros(3,N);
Fw =zeros(3,N);
U_val = zeros(3,N);
Y_val = zeros(3,N);
Fw_val = zeros(3,N);
time = linspace(0, N*dt, N)';

%%
[ A, B, D] = linear_SSM (F_c, M, Drag,dt);
%% LQ optimal control
Q = [100 0 0; 0 100 0; 0 0 1e2]; %tuning weight for errors
P = [1e-7 0 0; 0 1e-7 0; 0 0 1e-14] ;%tuning weight for inputs
[G1,G2]= dlqdu_pi(A,B,D,Q,P);
%%
delta_u1= prbs1(N,250,800)*1e5;
delta_u2= prbs1(N,350,400)*1e4;
delta_u3= prbs1(N,300,450)*1e5;
delta_u1_val= prbs1(N,850,250)*1e4;
delta_u2_val= prbs1(N,400,700)*1e5;
delta_u3_val= prbs1(N,600,450)*1e6;
%% simulation loop:
y = D*x;
y_val = D*x_val;
x_old = x;
x_old_val = x_val;
```

```
y_old = y;
y_old_val = y_val;
for i = 1:N
    F_c = current_force(V_c(:,i),x);
    F_c_val = current_force(V_c(:,i),x_val);
    F_w = wind_force(V_w(i), gamma(i), x(3),x(4), x(5), A_F, A_L, Long);
    F_w_val = wind_force(V_w(i), gamma(i), x_val(3),x_val(4), x_val(5), A_F, A_L, ↙
Long);
    %% calculate outputs:
    y = D*x;
    y_val = D*x_val;
    %% save input and output values:
    u_array(:,i) = u;
    y_array(:,i) = y;
    U_val(:,i) = u_val;
    Y_val(:,i) = y_val;
    Fw(:,i) = F_w;
    Fw_val(:,i)=F_w_val;
    %% calculate new input values:
    u = u + G1*(x-x_old) + G2*(y_old - sp(:,i))+[delta_u1(i);delta_u2(i);delta_u3 ↙
(i)] ;
    u_val = u_val + G1*(x_val-x_old_val) + G2*(y_old_val - sp_val(:,i))+ ↙
[delta_u1_val(i);delta_u2_val(i);delta_u3_val(i)] ;
    %% save old values before updating:
    x_old = x;
    y_old = y;
    x_old_val = x_val;
    y_old_val = y_val;
    %% update states:
    x = x + dt*nonlinear(x, u, F_w, F_c, M, Drag, [rand(1)*0.03,rand(1)*0.03,rand ↙
(1)*0.0001] );
    x_val = x_val + dt*nonlinear(x_val, u_val, F_w_val, F_c_val, M, Drag, [rand(1) ↙
*0.03,rand(1)*0.01,rand(1)*0.0001]);
end
%% plotting
figure(1),
subplot(311);plot(time/60, sp(1,:),time/60,y_array(1,:),'g-');ylabel('position in↙
surge [m]');legend('set point','surge position');
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('wind in surge↙
[N]');
figure(2),
subplot(311);plot(time/60, sp(2,:),time/60,y_array(2,:),'g-');ylabel('position in↙
sway [m]');legend('set point','sway position');
subplot(312);plot(time/60,u_array(2,:),'r-'); ylabel('control in sway [N]');
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('wind in sway↙
[N]');
figure(3),
subplot(311);plot(time/60, sp(3,:)*180/pi,time/60,y_array(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');legend('set point','yaw position');
subplot(312);plot(time/60,u_array(3,:),'r-'); ylabel('control in yaw [Nm]');
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('wind in yaw↙
[Nm]');
figure(5),
subplot(311);plot(time/60,sp_val(1,:),time/60,Y_val(1,:),'g-');ylabel('position in↙
surge [m]');legend('set point','surge position');
subplot(312);plot(time/60,U_val(1,:),'r-'); ylabel('control in surge [N]');
subplot(313);plot(time/60,Fw_val(1,:),'b-');xlabel('time [min]'); ylabel('wind in↙
surge [N]');
figure(6),
subplot(311);plot(time/60,sp_val(2,:),time/60,Y_val(2,:),'g-');ylabel('position in↙
sway [m]');legend('set point','sway position');
subplot(312);plot(time/60,U_val(2,:),'r-'); ylabel('control in sway [N]');
subplot(313);plot(time/60,Fw_val(2,:),'b-');xlabel('time [min]'); ylabel('wind in↙
sway [N]');
figure(7),
subplot(311);plot(time/60,sp_val(3,:)*180/pi,time/60,Y_val(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');legend('set point','yaw position');
subplot(312);plot(time/60,U_val(3,:),'r-'); ylabel('control in yaw [Nm]');
subplot(313);plot(time/60,Fw_val(3,:),'b-');xlabel('time [min]'); ylabel('wind in↙
yaw [Nm]');
%% saving data
Data=[y_array',u_array'];
save('data.txt','Data','-ASCII')
Data_val=[Y_val',U_val'];
save('data_val.txt','Data_val','-ASCII')
```

Appendix P

MATLAB script for simulating simple MPC with integral action using the identified system

```
%% implementing simple MPC with integral action developed by David Di Ruscio in a ↙
DP control system
clc
clear all
close all
load data.txt
parameters;
%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 10; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps
L =10;% prediction horizon
%% setpoint
sp = zeros(3, (N+L));
for m = 1:N+L
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [10 ; 10 ; 12*pi/180];
   elseif m>N/2 && m<=3*N/4
        sp(:,m)= [15 ; 20; 13*pi/180];
   elseif m>3*N/4
        sp(:,m)= [0 ; 15; 14*pi/180];
   end
end
%% Disturbances:
V_c = rand(3,N)*20;% water current speed in NED coordinate system as a random ↙
number
V_w = slowly(N)* 20;% wind speed in NED coordinate system as a random number
gamma = slowly(N)*360;% wind direction in NED coordinate system as a random number
% V_c = [20;20;20];% water current speed in NED coordinate system as a random ↙
number
% V_w = 10;% wind speed in NED coordinate system as a random number
% gamma = 25;% wind direction in NED coo
%% initialization:
x0 = [sp(1,1) sp(1,2) sp(1,3) 0 0 0]';
x = x0;
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1), gamma(1), x(3),x(4), x(5), A_F, A_L, Long);
u = -F_w;
%% Preallocation of arrays for plotting:
x_pos = zeros(6,N);
u_array= zeros(3,N);
Fw = zeros(3,N);
y_array = zeros(3,N);
time = linspace(0, N*dt, N)';
%% Calculating linear SSM matrices
[ A, B, DD] = linear_SSM ( F_c, M, Drag,dt);
%% data to get the identified model
Y=[data(:,1) data(:,2) data(:,3)];
U=[data(:,4) data(:,5) data(:,6)];
[A,B,D,E,K,F,x_0,Ef1,Yp]=dsr_e(Y,U,2,0,2,6);
nx = size(A,1); % number of system states
nu = size(B,2);% number of system inputs
ny = size(DD,1); % number of system outputs
%%
Q = [2.2e2 0 0; 0 2.6e4 0; 0 0 1e8]; %tuning weight for errors
```

```
P = [1e-7 0 0; 0 1e-7 0; 0 0 1e-15] ;%tuning weight for inputs
%% augmented system matrices:
At = [A zeros(nx,ny); D eye(ny,ny)];
Bt= [B ; zeros(ny,nu)];
Dt = [D eye(ny,ny)];
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(P,L);
H=FL'*Qt*FL+Rt;
%%
y = DD*x;
x_est = x;
x_old = x_est;
y_old = y;
xt = [x-x_old;y_old];
%% simulation loop:
for k = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
    F_c = current_force(V_c(:,k),x);
    F_w = wind_force(V_w(k), gamma(k), x(3),x(4), x(5), A_F, A_L, Long);
    y = DD*x;
    x_pos(:,k) = x;
    u_array(:,k) = u(:,1);
    y_array(:,k) = R*y;
    Fw(:,k) = F_w;
    pL=OL*At*xt;
    r = sp(:,k+1:L+k);
    r = r(:);
    f=FL'*Qt*(pL-r);
    du = quadprog(H,f);
    du = reshape(du,3,L); %arranged control inputs
    u = u + du(:,1);
    x_old = x_est;
    y_old = y;
    dx = nonlinear(x,u, F_w, F_c, M, Drag, neu);
    x= x+ dt*dx;
    % using current time output to estimate states by kalman filter
    x_est = A*x_est + B*u(:,1)+ K*(y - D*x_est);
    xt = [x_est-x_old;y_old];
end
%% plotting
figure(1),
subplot(311);plot(time/60,sp(1,1:N),time/60,x_pos(1,:),'g-');ylabel('position in↙
surge [m]');
title('Surge direction');
legend('set point','surge position');
grid
subplot(312);plot(time/60,u_array(1,:),'r-');xlabel('time [min]'); ylabel('control↙
in surge [N]');
title('control in Surge direction');
grid
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
surge [N]');grid
figure(2),
subplot(311);plot(time/60,sp(2,1:N),time/60,x_pos(2,:),'g-');ylabel('position in↙
sway [m]');
```

```
title('Sway direction');
legend('set point','sway position');
grid
subplot(312);plot(time/60,u_array(2,:),'r-');xlabel('time [min]'); ylabel('control↙
in sway [N]');
title('control in Sway direction');
grid
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('Wind force in↙
sway [N]');grid
figure(3),
subplot(311);plot(time/60,sp(3,1:N)*180/pi,time/60,x_pos(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');
title('Yaw direction');
legend('set point','Yaw position');
grid
subplot(312);plot(time/60,u_array(3,:),'r-');xlabel('time [min]'); ylabel('control↙
in yaw [N]');
title('control in Yaw direction');
grid
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('Wind moment↙
in yaw [Nm]');grid
figure(4);grid
plot_dir(y_array(2,:)',y_array(1,:)');xlabel('East position [m]'); ylabel('North↙
position [m]');%Copyright (c) 2010, Kangwon Lee%All rights
title('position in NED');
```

Appendix Q

MATLAB script for simulating LQ optimal control using the identified system

```
% test the identified model in the DP control system
clc
clear all
close all
load data.txt
parameters;
%% simulation settings
dt = 1; % time step [s]
t_start = 0; % simulating starting time [ min]
t_stop = 20; %simulating stopping time[ min]
N = ceil(60*(t_stop - t_start)/ dt ); % simulating steps
%% setpoint
sp = zeros(3, (N));
for m = 1:N
   if m<=N/4
        sp(:,m) = [0; 0 ; 0*pi/180];% north, east, yaw setpoint
   elseif m> N/4 && m<=N/2
        sp(:,m)= [10 ; 10 ; 12*pi/180];
   elseif m>N/2 && m<=3*N/4
        sp(:,m)= [5 ; 20; 10*pi/180];
   elseif m>3*N/4
        sp(:,m)= [0 ; 15; 14*pi/180];
   end
end
%% Disturbances:
V_c = rand(3,N)*20;% water current speed in earth coordinate system as a random↙
number
V_w = slowly(N)*20;% wind speed as a random number
gamma = slowly(N)*360;% wind direction as a random number
%% initialization:
x0 = [sp(1,1) sp(2,1) sp(3,1) 0 0 0]';
x = x0;
F_c = current_force(V_c(:,1),x);
F_w = wind_force(V_w(1,1), gamma(1,1), x(3),x(4), x(5), A_F, A_L, Long);
u = -F_w;
%% Preallocation of arrays for plotting:
y_array = zeros(3,N);
y_NED = zeros(3,N);
u_array= zeros(3,N);
Fw=zeros(3,N);
time = linspace(0, N*dt, N)';
%% Get the DD output matrix
[ AA, BB, DD] = linear_SSM(F_c, M, Drag,dt);
%% data to get the identified model
Y=[data(:,1) data(:,2) data(:,3)];
U=[data(:,4) data(:,5) data(:,6)];
[A,B,D,E,K,F,x_0,Ef1,Yp]=dsr_e(Y,U,2,0,2,6);
%% calculating control gain
Q = [1e2 0 0; 0 1e2 0; 0 0 2e4]; %tuning weight for errors
P = [1e-7 0 0; 0 1e-7 0; 0 0 1e-9] ;%tuning weight for inputs
[G1,G2]= dlqdu_pi(A,B,D,Q,P);
%% control limits
u_min = [-1e5; -1e6; -1e7];
u_max = [1e5; 1e6; 1e7];
%% initialization
x = x0;
x_est = x;
v = D*x;
```

```
x_old = x;
y_old = y;
%% simulation loop:
for i = 1:N
    % transformation matrix to tansform from body to NED coordinate system
    R = [cos(x(3)) -sin(x(3)) 0; sin(x(3))  cos(x(3)) 0; 0 0 1];
    F_c = current_force(V_c(:,i),x);
    F_w = wind_force(V_w(i,1), gamma(i,1), x(3),x(4), x(5), A_F, A_L, Long);
    y = DD*x;
    y_array(:,i) = y;
    y_NED(:,i) = R*y;
    u_array(:,i) = u;
    Fw(:,i) = F_w;
    u = u + G1*(x_est-x_old) + G2*(y_old - sp(:,i));
    % limit control
    if u(1)<u_min(1)
        u(1) = u_min(1);
    elseif u(1)>u_max(1)
        u(1) = u_max(1);
    end
    if u(2)<u_min(2)
        u(2) = u_min(2);
    elseif u(2)>u_max(2)
        u(2) = u_max(2);
    end
    if u(3)<u_min(3)
        u(3) = u_min(3);
    elseif u(3)>u_max(3)
        u(3) = u_max(3);
    end
    % update old values
    x_old = x_est;
    y_old = y;
    % update system
    dx = nonlinear(x, u, F_w,F_c, M, Drag, neu);
    x = x + dt*dx;
    % esimate immeasurable states
    x_est = A*x_est + B * u + K*(y- D*x_est);
end
%% plotting
figure(1),
subplot(311);plot(time/60, sp(1,:),time/60,y_array(1,:),'g-');ylabel('position in↙
surge [m]');legend('set point','surge position');grid
subplot(312);plot(time/60,u_array(1,:),'r-'); ylabel('control in surge [N]');grid
subplot(313);plot(time/60,Fw(1,:),'b-');xlabel('time [min]'); ylabel('wind in surge↙
[N]');grid
figure(2),
subplot(311);plot(time/60, sp(2,:),time/60,y_array(2,:),'g-');ylabel('position in↙
sway [m]');legend('set point','sway position');grid
subplot(312);plot(time/60,u_array(2,:),'r-'); ylabel('control in sway [N]');grid
subplot(313);plot(time/60,Fw(2,:),'b-');xlabel('time [min]'); ylabel('wind in sway↙
[N]');grid
figure(3),
subplot(311);plot(time/60, sp(3,:)*180/pi,time/60,y_array(3,:)*180/pi,'g-');ylabel↙
('position in yaw [deg]');legend('set point','yaw position');grid
subplot(312);plot(time/60,u_array(3,:),'r-'); ylabel('control in yaw [Nm]');grid
subplot(313);plot(time/60,Fw(3,:),'b-');xlabel('time [min]'); ylabel('wind in yaw↙
[Nm]');grid
```

```
figure(4);grid
plot_dir(y_NED(2,:)',y_NED(1,:)');xlabel('East position [m]'); ylabel('North↙
position [m]');title('position in NED coordinate system');
```