University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2023

Industrial IT and Automation

System Identification and Dynamic Positioning of Ships

Manuel Giraldo

250912

Faculty of Technology, Natural sciences and Maritime Sciences

Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 master's Thesis, 2023

Title: System Identification and Dynamic Positioning of Ships

Number of pages: 113

Keywords: MPC, PID Controller, LQ Optimal Control, Dynamic Positioning

Student: Manuel Giraldo

Supervisor: David Di Ruscio

External partner: None

Summary:

Dynamic Positioning of vessels has an important role in the safety of operations in the industries where the use of marine vessels is involved. Therefore, it is necessary to have an adequate implementation of system identification and accurate control methods for the vessels.

The objectives of this work are: Implementing a mathematical model of the vessel, performing system identification, developing accurate control methods, and comparing these methods under different conditions of operation.

The model was simplified discarding the drag coefficients and then its behavior was measured. The system identification took place by using the inputs and response of the closed loop system. PID, LQ Optimal control with integral action and MPC with integral action were selected as control methods. These methods were then compared under different benchmarks.

The simplified model behaves adequately. For the unconstrained control signals, LQ is the best option, whereas for the constrained case, MPC was better. The PID controllers are computationally cheap, however, very sensitive to wind disturbances which means that they need to be tuned constantly.

Preface

For millennia, the seas have been a source of resources and a way of commerce for civilizations, almost every culture has developed some type of transport over water. The use of Dynamic Positioning Systems for marine vessels is a crucial part in several industries that depend on them for safety and effectiveness in their operations. This has been researched broadly after the first half of the 20th century.

Works like that from Balchen [1] have brought models that are used nowadays to develop control devices which will help in the dynamic positioning. The control theory has several approaches for dynamic positioning. Some of the first were PID controllers; which use proportional, integral, and derivative action; some other methods depend on a model, that can either be provided analytically of obtained by using system identification.

This thesis is concerned with the implementation of an analytical model, the identification of a system model, and the use of this identified system to develop and compare different control methods for the dynamic positioning of a marine vessel.

Porsgrunn, May 1st of 2023

Manuel Giraldo

Contents

Nomenclature

LQ	Linear Quadratic
MPC	Model Predictive Control
NED	North, East Down
PID	Proportional, Integrative, and Derivative

1 Introduction

This document presents the results of the implementation of a marine vessel dynamic positioning system. The Dynamic Positioning of vessels is a well-researched topic. It has relevance in several industries like Oil and Gas, Logistics, Marine Research, and Aerospace, where the incorrect positioning of a vessel can trigger material and human lives losses, or irreparable damage to the sea nature life. Thus, implementing accurate control algorithms for the positioning of marine vessels is paramount to having a safe and reliable operation.

1.1 Previous Work

In 1980, Jens G. Balchen wrote the paper [1] that serves as main guide for this thesis work. The paper summarizes the results of developing a mathematical model for a marine vessel for what back then was called “Kongsberg Våpenfabrikk”. The model is divided into a Low Frequency and a high frequency component. The low frequency component shows the influence of wind and water currents in the model, while the high frequency shows the effect of Waves. In most subsequent works, the high frequency component is ignored, as the actuators of the vessels cannot compensate for the action of the waves.
The paper of N. Bargoth, C. Dalen, and D. Di Ruscio [2] is also of considerable assistance for this thesis, since it contains a more clear representation of the low frequency component of the model found in [1]. It also provided inspiration about how to perform a comparison between different control methods.

1.2 Objectives

The purpose of this work is to implement a non-linear mathematical model of a vessel and its behavior at sea; perform a system identification to obtain an identified linearized model; and to implement control methods that will interact with the identified model to control the non-linear model. The main source of information for this work are [1], and [2]. Where the mathematical model has been formulated and simplified.

1.3 Methods

The model implemented in [1] has low frequency and high frequency components. This work focuses on the low-frequency part, which accounts for the wind and the action of water currents. The model is then simplified by discarding the drag coefficients. A closed loop system is then implemented wherein the control is performed using PID controllers.

The control signals and the outputs of the closed loop system are gathered and used to perform the system identification using the dsr_e function from the system identification toolbox.

The resulting A, B, and D model matrices will be used to implement LQ Optimal Control with Integral action and MPC with integral action, which gives three control methods in total. These methods will be evaluated and compared under different benchmarks like settling time, overshot and input signal usage.

1.4 Document Structure

In chapter 2, the mathematical model will be formulated. It then will be implemented in MATLAB and simplified. The initial and simplified versions of the model will be compared.

In chapter 3 the PID controllers are implemented, then in chapter 4 the system identification takes place. In chapters 5 and 6, the LQ optimal control with integral action and MPC with integral action methods will be implemented, with a short mathematical explanation for each of them.

In chapter 7 the performance of the PID, LQ with integral action, and MPC with integral action methods will be compared.

2 Working With the Model

The equations and the workings of the dynamic model were implemented using as guidelines the works from [1], and [2]. Some changes were implemented in the use of the ship reference frame with respect to the method used in [2].

[1] provides the main guidelines for this master's thesis, particularly for the mathematical model. The model is divided into High-Frequency motion and Low-Frequency motion components. The focus of this master's thesis is the low-frequency motion component, which represents the motions induced by wind, thrust, and water currents. In the Surge, Sway, and Yaw coordinates, as explained in [1].

2.1 NED Coordinate Frame and Vessel's Coordinate Systems

The model Is described with two coordinate systems, which share the same origin. One system has the absolute earth-based coordinates; this will be used to define the absolute set points and position of the vessel during the simulation, it will be referred as the NED coordinate frame (short for North, East, Down, as per [3]). The other will be parallel to the vessel; as the angle of the vessel with respect to the NED frame changes, there will be an angle offset between the NED coordinate frame and the vessel's frame.

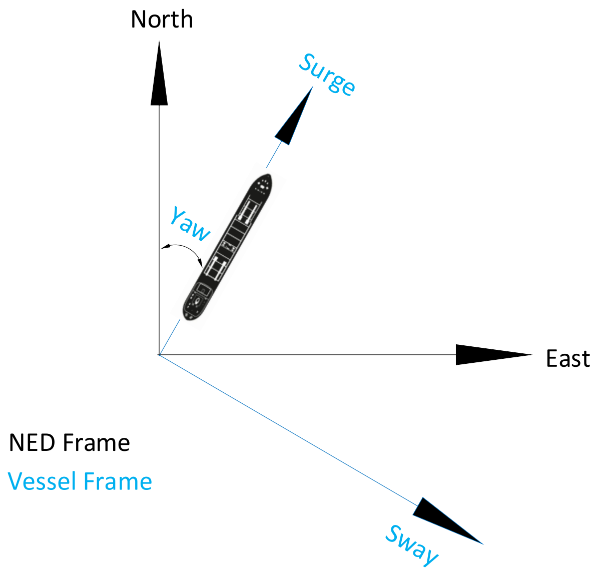


Figure 2-1 NED and Vessel's Coordinate Frames

Figure 2-1 shows the interaction between the ship and both coordinate systems. As yaw changes, the relative values of surge and sway in the vessel's frame with regards to the absolute values in the NED frame.

2.2 Model Equations

The drag coefficients (d_1, d_2, d_3, and d_4) are considerably low, whereas the momentum coefficients (m1, m2, and m3) are higher. The coefficients values designated in equations (2.1) to (2.6) are shown in Table 2-1.

Table 2-1. Values of drag and momentum coefficients.

Coefficient	Value
d_1	$5 \cdot 10^{-5}$
d_2	$21 \cdot 10^{-5}$
d_3	$1.1 \cdot 10^{-10}$
d_4	$201 \cdot 10^{-15}$
m_1	$4 \cdot 10^{6}$
m_2	$4 \cdot 10^{7}$
m_3	$4.7 \cdot 10^{10}$

The model can be described with the following equations, as per [2]:

$$\dot{x}_1 = x_4 \tag{2.1}$$

$$\dot{x}_2 = x_5 \tag{2.2}$$

$$\dot{x}_3 = x_6 \tag{2.3}$$

$$\dot{x}_4 = -\frac{d_1}{m_1}\left|x_4 - v_{c_{su}}\right|\left(x_4 - v_{c_{su}}\right) + \frac{1}{m_1}\left(F_{w_{su}} + u_1\right) \tag{2.4}$$

$$\dot{x}_5 = -\frac{d_2}{m_2}\left|x_5 - v_{c_{sw}}\right|\left(x_5 - v_{c_{sw}}\right) + \frac{1}{m_1}\left(F_{w_{sw}} + u_2\right) \tag{2.5}$$

$$\dot{x}_6 = -\frac{d_3}{m_3}|x_6|x_6 - \frac{d_4}{m_3}\left|x_5 - v_{c_{sw}}\right|\left(x_5 - v_{c_{sw}}\right) + \frac{1}{m_3}\left(N_w + u_3 + N_c\right) \tag{2.6}$$

Equations (2.1) to (2.6) correspond to the change of different states through time, these states are:

$$x_1: Position\ in\ Surge\ Direction$$

$$x_2: Position\ in\ Sway\ Direction$$

$$x_3: Orientation\ in\ Yaw\ Angle$$

$$x_4: Velocity\ in\ Surge\ Direction$$

$$x_5: Velocity\ in\ Sway\ Direction$$

$$x_6: Angular\ Velocity\ in\ Yaw$$

We also have that:

$$v_{c_{su}}: Water\ Current\ Velocity\ in\ Surge\ Diretion\ [m/s]$$

$$v_{c_{sw}}: Water\ Current\ Velocity\ in\ Sway\ Diretion\ [m/s]$$

$$N_c: Water\ Current\ Moment\ in\ Yaw\ [Nm]$$

$$F_{W_{su}}: Wind\ Force\ in\ Surge\ Direction\ [N]$$

$$F_{W_{sw}}: Wind\ Force\ in\ Sway\ Direction\ [N]$$

$$N_w: Wind\ Moment\ Yaw\ [Nm]$$

As it can be seen in equations (2.4), and (2.5), the left terms are multiplied by a division of drag coefficients by momentum coefficients; also, for Equation (2.6), the left and middle terms show the same multiplication. Considering the low magnitude of the drag coefficient, which is in the numerator of the division, and the high magnitude of the momentum coefficient, which is in the denominator; these terms could be approximated to zero without affecting the precision of the model.

This also means, that for this model, the wind velocity is of greater significance than the velocity of the water current, as the water velocity is multiplied by the division of drag coefficients over momentum coefficient in (2.4), (2.5), and (2.6).

It is worth noting that the system's equations are expressed in relation to the vessel's coordinate frame, which means that to obtain the position of the vessel in north and east coordinates, it is required to perform a conversion. To achieve this, it is required to use a conversion matrix, as seen in equation (2.7).

$$R = \begin{bmatrix} Cos(x_3) & -Sin(x_3) & 0 \\ Sin(x_3) & Cos(x_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

The conversion matrix R will be used also to calculate the forces of the wind and water current in the surge and sway directions (in the vessel coordinate frame), as these phenomena will be first simulated in the NED frame.

2.3 Programing the Model

MATLAB was selected for the simulation of the system since it offers useful control libraries. It was decided to use classes, which provides more flexibility to implement changes in the code and the simulations. The scripts are based on work from [3]. The code used for this and all subsequent sections can be found in Appendix A.

2.3.1 BalchenModel Class

A class was developed to replicate the behavior of the vessel. It has a constructor method that has as inputs the drag coefficients, the momentum coefficients, the initial states of the vessel, and parameters like the frontal windage area, the lateral windage area, and the overall length of the vessel. The model class is composed of other methods that will simulate the effect of wind and water current on the vessel. With this information, another method will calculate the change in the states also considering the input signals.

2.3.1.1 Constructor Method

In the constructor method, the parameters of the vessel model are initialized. It receives as inputs the drag and momentum coefficients, the initial values of the states, the neu coefficients, the frontal windage area the lateral windage area, and the vessel's length:

```
%% Constructor
function obj = BalchenModel(d,m,x,neu, A_F, A_L, L)
  obj.D = d; %Drag Coefficients
  obj.M = m; % Momentum coefficients
  obj.X = x; % Initial States
  obj.Neu = neu;
  obj.A_F = A_F; %Frontal windage area
  obj.A_L = A_L; %Lateral windage area
  obj.L = L; %Vessel's length
end
```

2.3.1.2 VesselModel Method

In this method, the state equations are implemented. It receives as inputs the current states of a vessel's instance, the input signals, the wind and water current forces, and the momentum and drag coefficients.

As outputs, it will generate an updated rate of change of the model's states.

```
%% Method that contains the vessel model, it will be called by Runge-Kutta
function f = VesselModel(obj, x, u, F_w, V_c, m, d, neu)

      f=zeros(6,1);
      f(1) = x(4);
      f(2) = x(5);
      f(3) = x(6);

      f(4) = - (d(1)/m(1))*abs(x(4) - V_c(1))*(x(4) - V_c(1)) +
      (1/m(1))*(F_w(1) + u(1)) + neu(1);
      f(5) = - (d(2)/m(2))*abs(x(5) - V_c(2))*(x(5) - V_c(2)) + (1/m(2))*(F_w(2)
      + u(2)) + neu(2);
      f(6) = - (d(3)/m(3))*abs(x(6))*x(6) - (d(4)/m(3))*abs(x(5) - V_c(2))*(x(5)
      - V_c(2)) + (1/m(3))*(F_w(3) + u(3) + V_c(3)) + neu(3);

end
```

2.3.1.3 RungeKutta Method

To obtain the state values, it is needed to integrate the output of the VesselModel method. The differential equations of the model are solved using Runge-Kutta's method, as obtained from [4].

This method receives as inputs the same inputs of the VesselModel method. It calls the VesselModel method several times to obtain an averaged calculation of the value of x in the next timestep.

```
%% Runge Kutta to integrate and get updated state values
function x_next = RungeKutta(obj, x, u, F_w, v, dt, m, d, neu)

    K1 = obj.VesselModel(x, u, F_w, v, m, d, neu);
    K2 = obj.VesselModel(x+K1.*(dt/2), u, F_w, v, m, d, neu);
    K3 = obj.VesselModel(x+K2.*(dt/2), u, F_w, v, m, d, neu);
    K4 = obj.VesselModel(x+K3.*(dt), u, F_w, v, m, d, neu);

    x_next = x + (dt/6).*(K1+2.*K2+2.*K3 + K4);

end
```

2.3.1.4 CurrentVelocity Method

The method receives the current velocity in the north, east and yaw components in the NED system and converts them to the vessel's coordinate frame.

```
%% Transform the water velocity to the Ship reference
function V_c = CurrentVelocity(obj, W_c, state)  %Receives as input the
current velocities and converts them to the ship frame of reference.

    R = [cos(state(3)) -sin(state(3)) 0;
        sin(state(3)) cos(state(3)) 0;
        0 0 1];

    V_c = R.'*W_c;

end
```

2.3.1.5 WindForce Method

This method receives as inputs an absolute wind velocity value, its incidence angle, the vessel's yaw, the vessel's velocity in surge, the vessel's velocity in sway, and vessel's body parameters. These are used to calculate the force applied by the wind on the vessel in the surge and sway axis. The logic and the values in this method were obtained from [3].

```
%% Calculate wind forces in the ship frame of reference
function forcesArray = WindForce(obj, windVel, wAngle, vYaw,vSpeedSurge,
vSpeedSway, A_F, A_L, L)
    %windVel: wind speed [m/s] in NED frame
    %wAngle: wind direction [deg] in NED frame
    %vYaw: vessel heading [rad]
    Cx = 0.6;
    Cy = 0.8;
    Cn = 0.1;
    % Cx, Cy, Cn: wind coefficients: assumed to be constants
    wAngle = wAngle * pi/180;
    uw = windVel * cos(wAngle - vYaw); % windspeed in surge
    vw = windVel * sin (wAngle - vYaw);% windspeed in sway
    urw = vSpeedSurge - uw; % wind relative speed in surge
    vrw = vSpeedSway - vw; % wind relative speed in sway
    Vr = sqrt(urw^2 + vrw^2);%wind relative speed
    rho = 1.23; % wind density [kg\m^3]
    windForceSurge = 0.5*rho*Cx*A_F*cos(wAngle)* Vr^2; %Wind force in
    surge
    windForceSway = 0.5*rho*Cy*A_L*sin(wAngle)* Vr^2; %wind force in sway
    windMoment = 0.5*rho*Cn*A_L*L*sin(2*wAngle)* Vr^2; %Wind moment
    forcesArray = [windForceSurge, windForceSway, windMoment]';
end
```

2.3.1.6 UpdateState Method

With the UpdateState method, the other methods are called and used to update the state of the class's instance. It first generates a calculation of the water current and wind forces in the vessel's reference frame. Then, using these values, it calls the RungeKutta method, which itself will call the VesselModel's method. The output will be the updated states values, which will be also changed as the class's instance's properties.

```
%% Method that will be called to update the state
function [x, Y] = UpdateState(obj, u, W_c, W_v,gamma, dt)
    %W_c = Water current velocities in the NED frame
    %W_v = wind velocities in the NED frame
    %gamma = wind direction [deg] in NED frame

    V_c = obj.CurrentVelocity(W_c, obj.X);
    F_w = obj.WindForce(W_v, gamma, obj.X(3),obj.X(4), obj.X(5), obj.A_F,
    obj.A_L, obj.L);
    new_state = obj.RungeKutta(obj.X, u, F_w, V_c, dt, obj.M, obj.D,
    obj.Neu);
    obj.X = new_state;
    x = obj.X;
    Y = ConvertToNED(obj, obj.X(1:3));

end
```

2.3.2 EnvironmentCondition Class

An instance of this class will be called to simulate the weather conditions. The class does not have a creator method; however, it has methods to represent the wind and water current.

2.3.2.1 SimulateWind Method

This method will receive as input, an average wind speed, which will be multiplied by random values to obtain a varying wind velocity vector, the size of this vector will be given by an input "N" corresponding to the number of wind speed values to be generated, this N value should be equal to the number of timesteps of the simulation.

It will also output an array of angles, which represent the direction of the wind in the NED coordinate frame.

2.3.2.2 SimulateWater Method

This method will receive as inputs "N" for the length of the simulation, an average current speed, and a typeOfSignal value, which indicates if the method will simulate sinusoidal water currents or steps.

2.4 Simulation and comparison of Drag Coefficients

A simulation script was developed, in which the model is subject to wind and water current disturbances; then, its position in the west axis of the NED framework is plotted against the position in the north axis of the NED framework.

The water current disturbances were recreated using sinusoidal signals, as seen in Figure 2-2; whereas the wind disturbances were simulated using random values that were smoothed using the smooth function from MATLAB, as displayed in Figure 2-3. With this in place, it was possible to simulate the response of the model to said disturbances with the given drag coefficient values and with these approximated to zero.

After performing the simulation, the position in the NED framework of the model in both cases was compared and the covariance of the differences between the two cases was calculated.

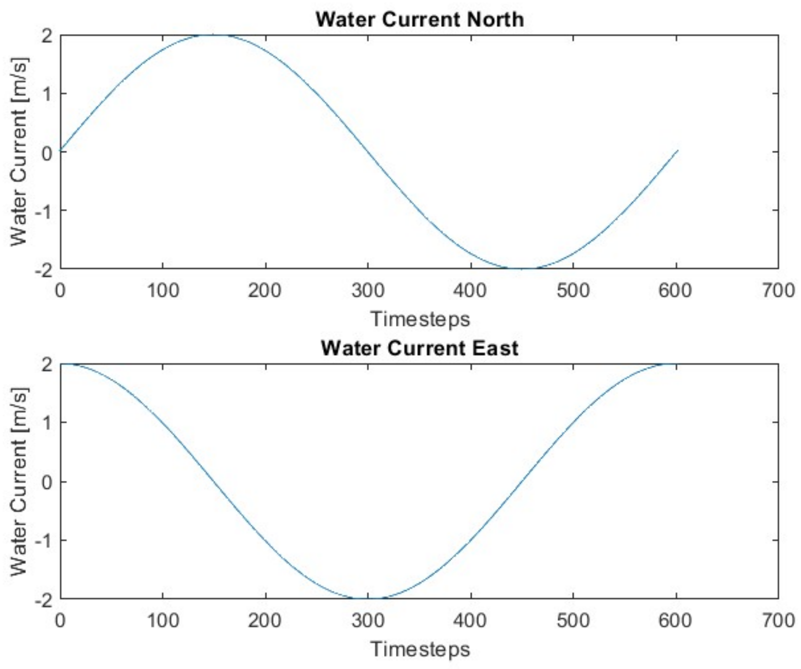


Figure 2-2 Water Current Simulation for Drag Coefficients Comparison

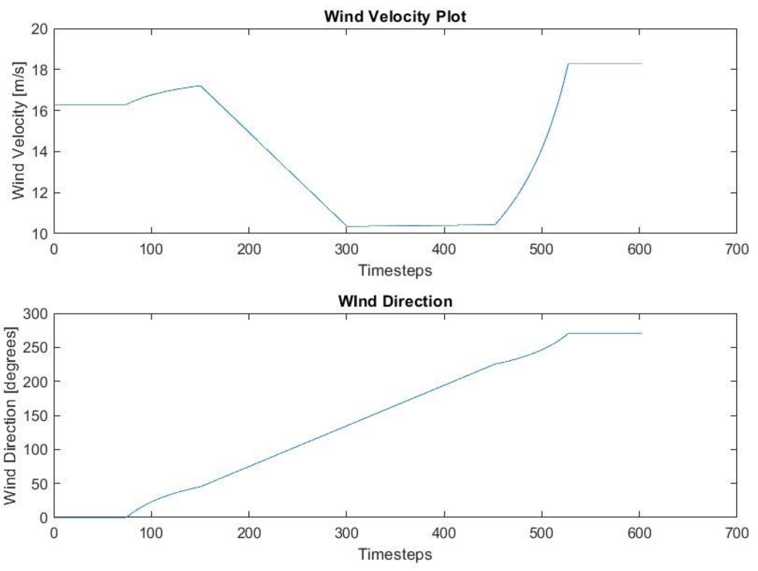


Figure 2-3 Wind Speed Simulation for Drag Coefficients Comparison

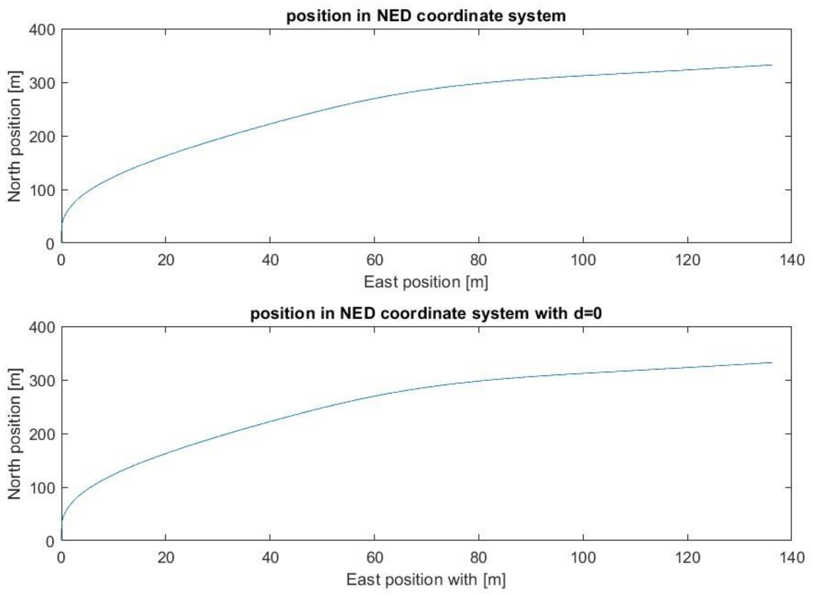


Figure 2-4 Position of the Vessel in the NED Framework for both Drag Coefficient Values

In Figure 2-4 the change in the position for the vessel in the NED framework is shown. In the top plot there is the position for the given drag coefficients in [2], while in the bottom plot, there is the same but for the drag coefficients equal to zero.

After performing a visual inspection of Figure 2-4, it can be concluded that both alternatives have a similar behavior. This can also be evidenced in Figure 2-5, where the difference between the two alternatives to each of the states is shown; the difference in positions in the North and East axis in NED, and in Yaw have absolute values that could be deemed insignificant.

This will be further demonstrated by calculating the variance of the difference between both results.

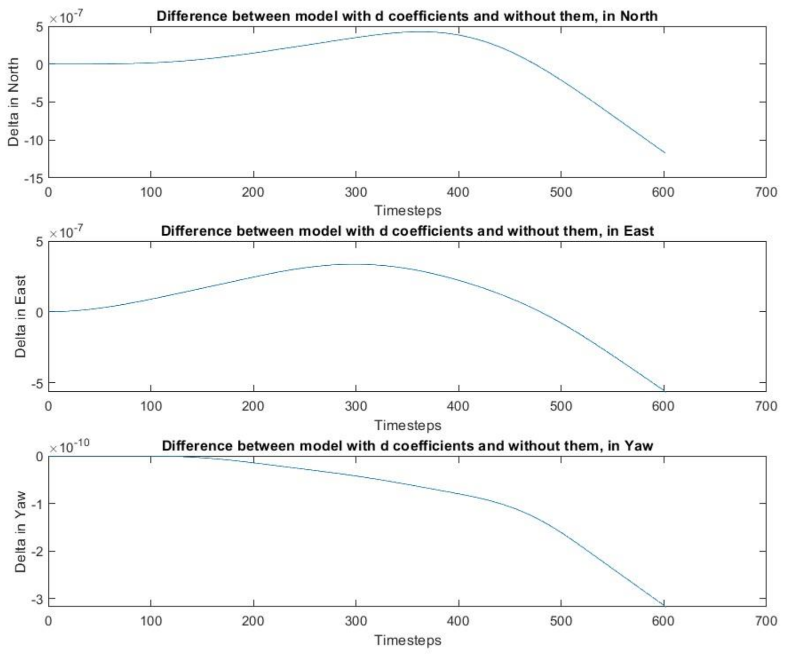


Figure 2-5 Difference of the Three Main Outputs of the Models with Given Drag Coefficients and with these Equal to Zero

2.4.1 Covariance of Differences of the model with given Drag Coefficients and with Them Equal to Zero

The covariance calculation is given by:

$$E = \frac{\Delta x^T \cdot \Delta x}{N} \tag{2.8}$$

Where Δx is the difference between the two alternatives of the model; N is the number of samples, and E is the covariance.

This value needs to be calculated three times, as three vectors of values need to be analyzed: Position in North axis of NED during the simulation time, position in East axis of NED during the simulation time, and Yaw during the simulation time.

This is implemented in MATLAB and the results are seen in Figure 2-6. The low covariance means that there is a considerable similarity in the outputs between the model with the given drag coefficient values and with the drag coefficients equal to zero. This aligns with the results showcased in Figure 2-4 and Figure 2-5. Which brings to the conclusion that the terms in the left in equations (2.4), (2.5), and (2.6) can be made equal to zero without a meaningful effect in the accuracy of the model.

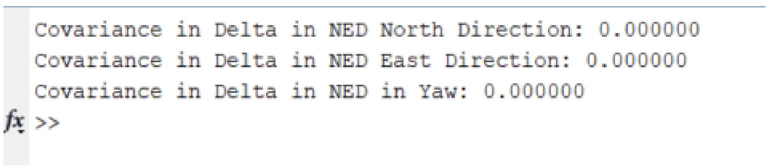

```
Covariance in Delta in NED North Direction: 0.000000
Covariance in Delta in NED East Direction: 0.000000
Covariance in Delta in NED in Yaw: 0.000000
fx >>
```

Figure 2-6 Results of Covariance Analysis for the Model's Alternatives

Figure 2-6 Shows the result of calculating the covariance of the differences, which is zero. This means that for this vessel's parameters, there is no need to use the drag coefficients in the model.

2.5 Conversion of the Setpoint from the NED Coordinate Frame to the Vessel's

The positions of a marine vessel are given in absolute earth-based coordinates, in longitude and latitude. However, the controls in the ship send signals to actuators that are inside the vessel and therefore, within its coordinate frame.

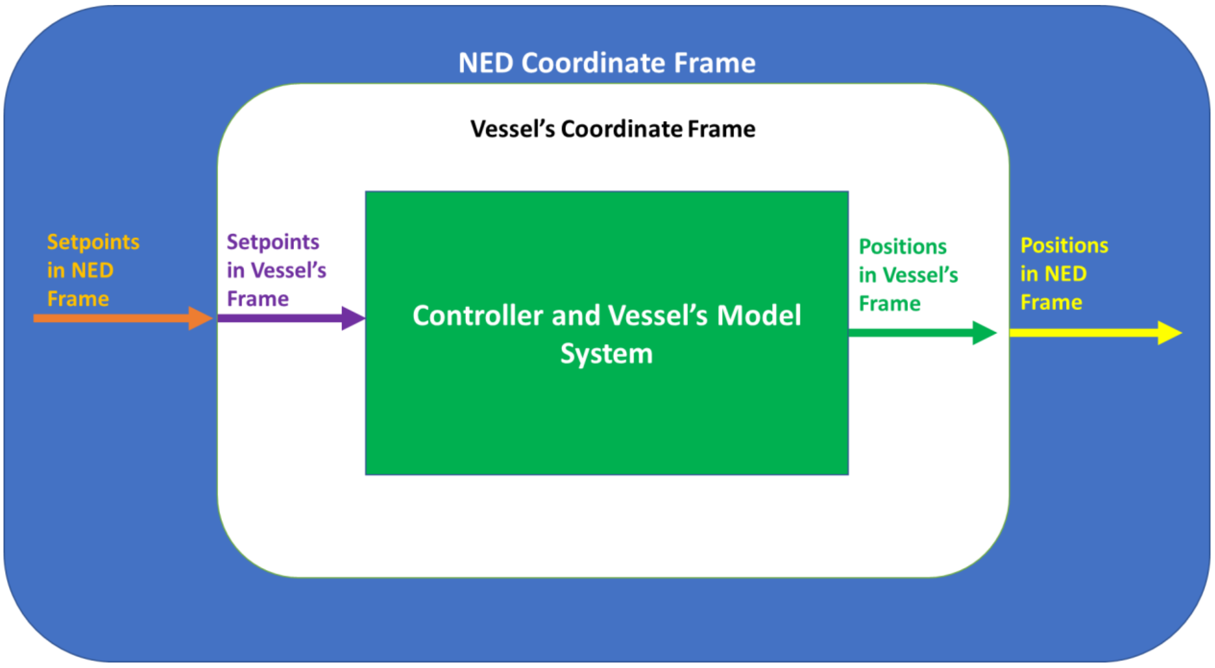

NED Coordinate Frame
Vessel's Coordinate Frame
Setpoints in NED Frame
Setpoints in Vessel's Frame
Controller and Vessel's Model System
Positions in Vessel's Frame
Positions in NED Frame

Figure 2-7 Setpoints and Positions of Vessel in NED and Vessel Coordinate Frames

Figure 2-7 shows the setpoints and positions pertaining to each of the coordinate frames. As the vessel changes position and angle throughout the simulation, the setpoints values in its frame of reference will change too.

This makes it necessary to convert the setpoints from the absolute NED coordinates to the vessel's coordinate frame before feeding them to the control algorithms in every timestep of the simulation. Likewise, the positions obtained from the control algorithm, which are established in the vessel's coordinate frame, need to be converted back to the NED frame to compare them against the setpoints.

2.6 Linearization of the Model

The linearized model is required to implement LQ and MPC controllers with integral action. It will be used to calculate an optimized output signal that will be used to control the model.

According to [5], the linearization will be implemented to obtain a state-space model in the form:

$$\dot{x}(t) = A_c x(t) + B_c u(t) \tag{2.9}$$

$$y(t) = D_c x(t) \tag{2.10}$$

To make this, the matrices A_c, B_c, and D_c need to be obtained. The non-linear model can be expressed in the form:

$$\dot{x}(t) = f\big(x(t), u(t)\big) \tag{2.11}$$

$$y(t) = g\big(x(t), u(t)\big) \tag{2.12}$$

So,

$$Ac = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}, \frac{\partial f_1}{\partial x_1}, \cdots \frac{\partial f_1}{\partial x_{n_x}} \\ \frac{\partial f_2}{\partial x_1}, \frac{\partial f_2}{\partial x_1}, \cdots \frac{\partial f_2}{\partial x_{n_x}} \\ \vdots \\ \frac{\partial f_{n_x}}{\partial x_1}, \frac{\partial f_{n_x}}{\partial x_1}, \cdots \frac{\partial f_{n_x}}{\partial x_{n_x}} \end{bmatrix},$$

$$Bc = \begin{bmatrix} \frac{\partial f_1}{\partial u_1}, \frac{\partial f_1}{\partial u_1}, \cdots \frac{\partial f_1}{\partial u_{n_u}} \\ \frac{\partial f_2}{\partial u_1}, \frac{\partial f_2}{\partial u_1}, \cdots \frac{\partial f_2}{\partial u_{n_u}} \\ \vdots \\ \frac{\partial f_{n_u}}{\partial u_1}, \frac{\partial f_{n_u}}{\partial u_1}, \cdots \frac{\partial f_{n_u}}{\partial u_{n_u}} \end{bmatrix},$$

And

$$Ac = \begin{bmatrix} \frac{\partial g_1}{\partial x_1}, \frac{\partial g_1}{\partial x_1}, \cdots \frac{\partial g_{n_y}}{\partial x_{n_x}} \\ \frac{\partial g_2}{\partial x_1}, \frac{\partial g_2}{\partial x_1}, \cdots \frac{\partial g_{n_y}}{\partial x_{n_x}} \\ \vdots \\ \frac{\partial g_{n_x}}{\partial x_1}, \frac{\partial g_{n_x}}{\partial x_1}, \cdots \frac{\partial g_{n_y}}{\partial x_{n_x}} \end{bmatrix},$$

These expressions for A_c, B_c, and D_c need to be evaluated close to a stable operating point, which in this case, will be zero for y, u, and x. Then, these values are discretized using MATLAB's c2d function.

2.6.1 Use of the Simplified Model

As seen before, for this vessel’s characteristics, the drag coefficients can be approximated to zero, which simplifies the equations considerably. Equations (2.4), (2.5), and (2.6) become:

$$\dot{x}_4 = \frac{1}{m_1}\left(F_{w_{su}} + u_1\right) \tag{2.13}$$

$$\dot{x}_5 = \frac{1}{m_1}\left(F_{w_{sw}} + u_2\right) \tag{2.14}$$

$$\dot{x}_6 = \frac{1}{m_3}\left(N_w + u_3 + N_c\right) \tag{2.15}$$

These simplified expressions are used in the function linearize_model.m which will produce matrices A, B, and D as outputs; by creating the matrices A_c, B_c, and D_c, performing the derivatives on equations (2.1), (2.2), (2.3), (2.13), (2.14), and (2.15); and discretizing them. This script can be found in Appendix A.

```
unction [A, B, D] = linearize_model(u, d, m, dt)
Ac = [0 0 0 1 0 0;
      0 0 0 0 1 0;
      0 0 0 0 0 1;
      0 0 0 0 0 0;
      0 0 0 0 0 0;
      0 0 0 0 0 0;];
%The last three rows of Ac are zero because we are using the simplified
%model where d is approximated to zero as it is very small compared to the
%m values.

Bc = [0 0 0;
      0 0 0;
      0 0 0;
      (1/m(1)) 0 0;
      0 (1/m(2)) 0;
      0 0 (1/m(3))];
%Same here, the terms where there is a d divided by m are approximated to
%zero.

Dc = [1 0 0 0 0 0;
      0 1 0 0 0 0;
      0 0 1 0 0 0;];

sys = ss(Ac,Bc,Dc,0);
ds = c2d(sys,dt);
A = ds.a; B = ds.b; D = ds.c;
end
```

3 Dynamic Positioning Using PID Controllers

PID is the acronym for Proportional, Integral, and Derivative. This type of controller can be tuned without the need to have a lot of knowledge about the dynamic model of the vessel. It can be tuned in a trial-and-error fashion, which makes it very simple to implement. It is important to know that the model has a double integrator and hence, the derivative term will be needed.

The mathematical expression for the parallel PID controller is:

$$u_k = K_p\left(s_{p_k} - y_k\right) + \frac{K_P T_s}{T_i}\sum_{i=1}^{k}\left(s_{p_i} - y_i\right) + K_p T_d \frac{e_k - e_{k-1}}{T_s} \tag{3.1}$$

Where:

$u{:}\, The\ control\ signal$

$s_p{:}\, The\ setpoint$

$y{:}\, The\ current\ output\ value$

$K_p{:}\, The\ proportional\ constant\ of\ the\ controller$

$T_i{:}\, The\ integral\ constant\ of\ the\ controller$

$T_d{:}\, The\ derivative\ constant\ of\ the\ controller$

$T_s{:}\, The\ timestep$

$e_k{:}\, The\ error, or\ the\ difference\ between\ s_p\ and\ y\ in\ a\ given\ time\ k$

The derivative term of the controller provides increased response to sudden changes in disturbances in setpoints. However, this response needs to be relaxed as it would generate violent changes in the control signal and the outputs. It is necessary to add a lowpass filter to smooth the change in the error.

The equation for the lowpass filter was obtained from [6], and is:

$$y_k = (1 - a)y_{k-1} + au_k \tag{3.2}$$

$$a = \frac{T_s}{T_f + T_s} \tag{3.3}$$

Where:

$T_f{:}\, The\ filter\ time\ constant\ (a\ multiple\ of\ T_s\ usually)$

$T_s{:}\, Timestep$

$u_k{:}\, Unfiltered\ input\ signal, in\ this\ case, the\ error\ e_k$

$y_k{:}\, The\ filtered\ output\ signal, in\ this\ case, the\ filtered\ error\ e_k$

These equations are implemented in MATLAB using a class named PIDcontroller.m. Surge, Sway and Yaw will have independent PID controllers, each will be initialized as an instance of the PIDcontroller class, with specific proportional, integrative, and derivative parameters.

3.1 Tuning the PID Controllers for Surge, Sway, and Yaw

Since there is a closed loop, the method selected for the tuning of the controllers will be the Ziegler Nichols Method, without the wind and water current disturbances, as described in [7].

A K_p is selected for the surge control signal, while T_i is set to a high value so that the integrative term of the PID is approximated to zero; and the T_d term is set to zero. This process will be repeated for the sway and yaw controllers, so it will be only shown for surge.

$K_p = 1000$

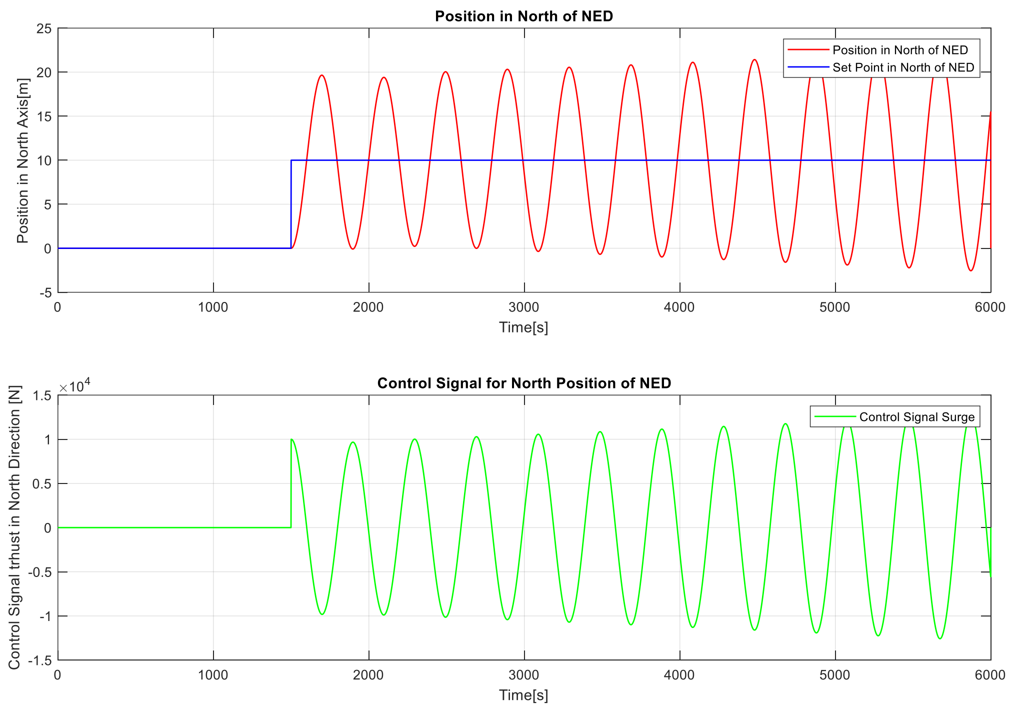


Figure 3-1 Tunning of PID controller using Ziegler-Nichols, first value of Kp

As seen in Figure 3-1, the oscillatory position signal has a period of approximately 400 seconds. According to the Ziegler-Nichols method, T_i should be $T_i = P_u/2$. So, T_i should be equal to 200. Then, $K_p = 0.6 \cdot K_{pu}$ and $T_d = T_i/4$.

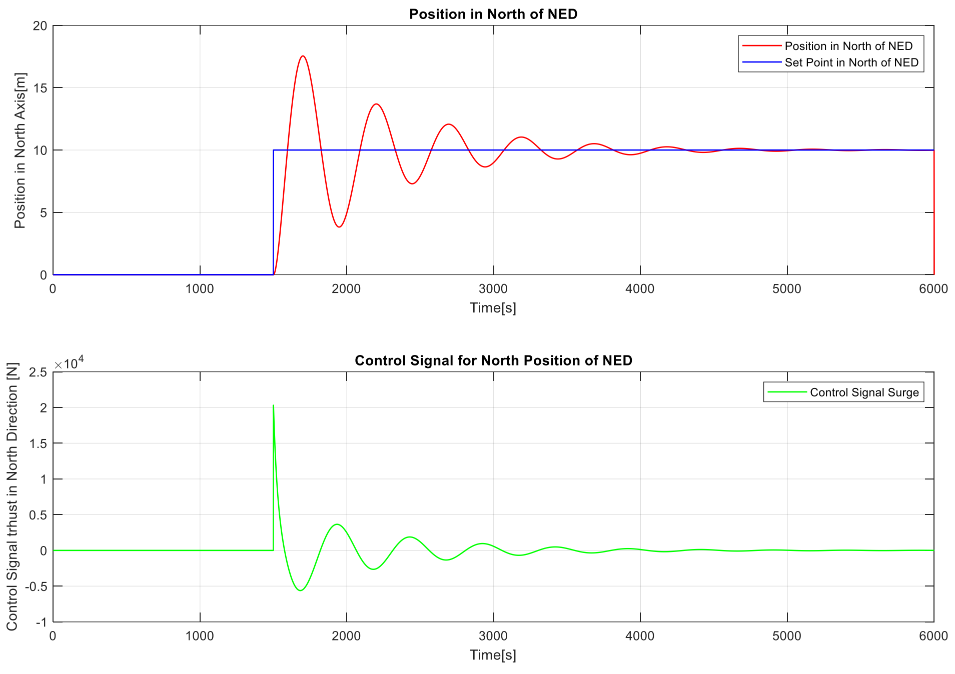


Figure 3-2 Behavior of PID Controller for Surge Kp=600, Ti=200, Td=50

Figure 3-2 shows that there is some instability, so the constants need to be tweaked. The response is improved by increasing the value of the derivative term, hence increasing the T_d constant value.

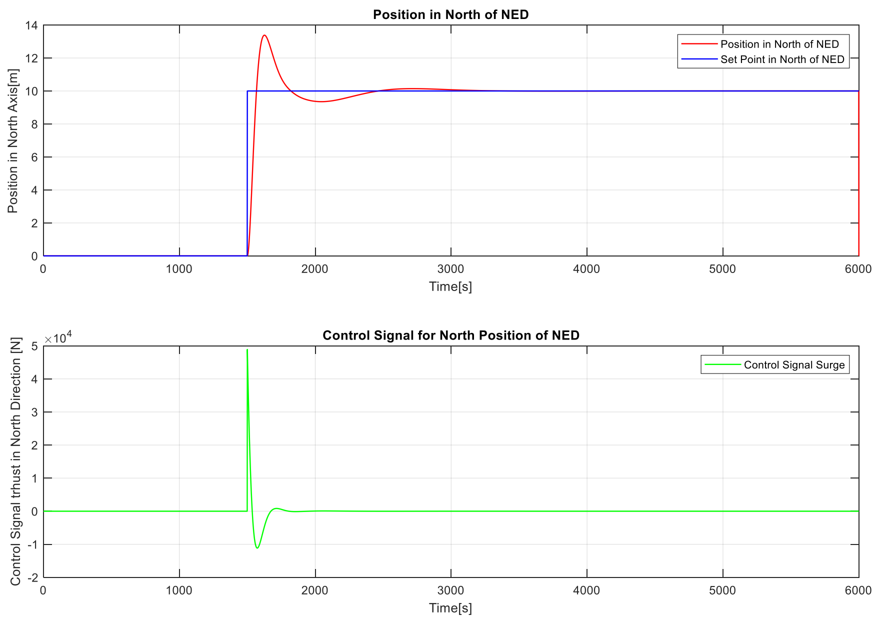


Figure 3-3 Behavior of the PID Controller with Kp=600, Ti=200, and Td=150

As seen in Figure 3-3, there is an improvement on the behavior of the controller after manually changing the constants of the PID controller.

For the sway PID controller, the process is repeated, with the result shown in Figure 3-4. It displays some overshot produced by the derivative term. However, since there is a double integrator in the model, it is necessary to use the derivative term of the PID controller.

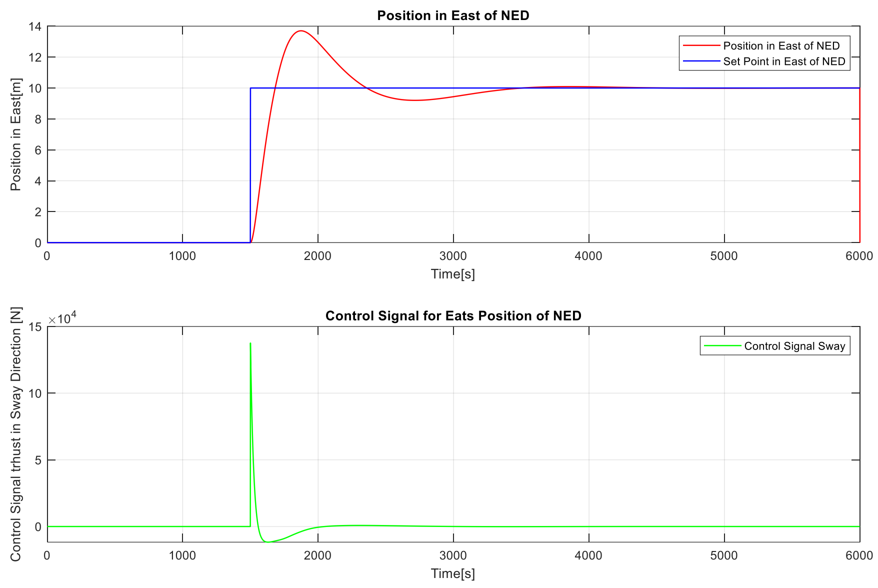


Figure 3-4. Behavior of Sway PID Controller with Kp=900, Ti=550, and Td=300

Then, for the yaw controller, the process is repeated until the system reaches stability, with the result shown in Figure 3-5. The magnitude of the K_p constant is noticeably bigger than for the other controllers.

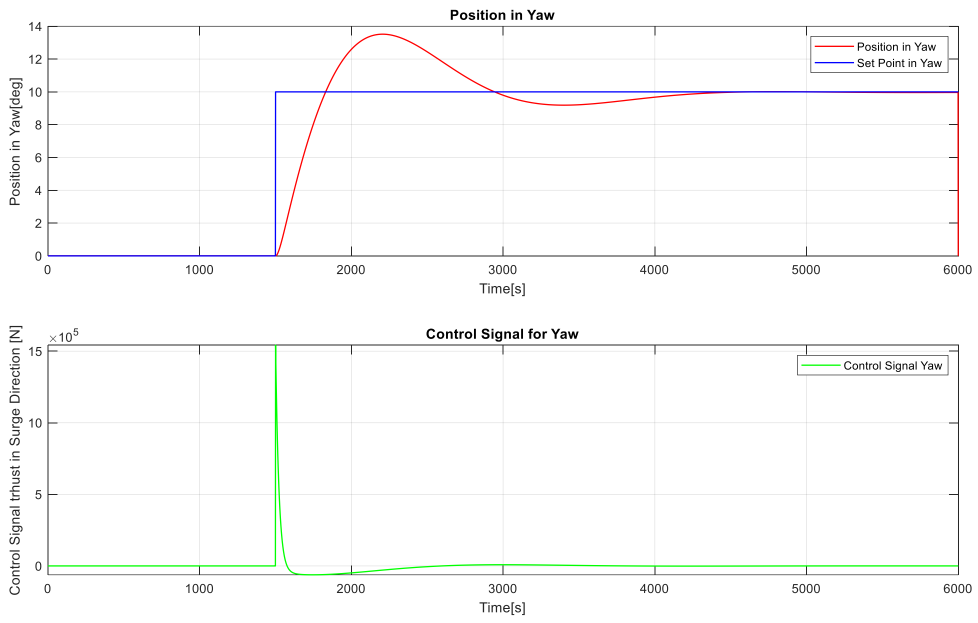
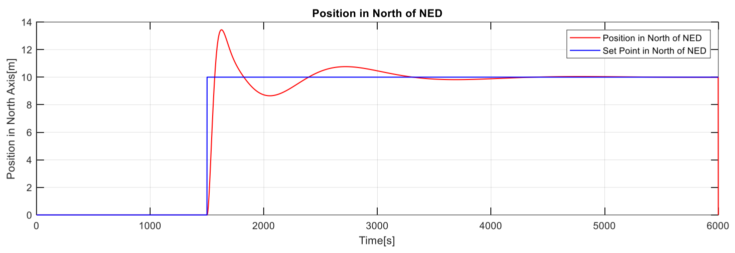


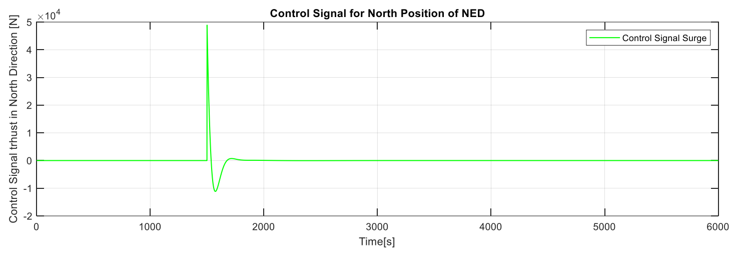
Figure 3-5. Yaw Controller with Kp=500000, Ti=1500, and Td=350.

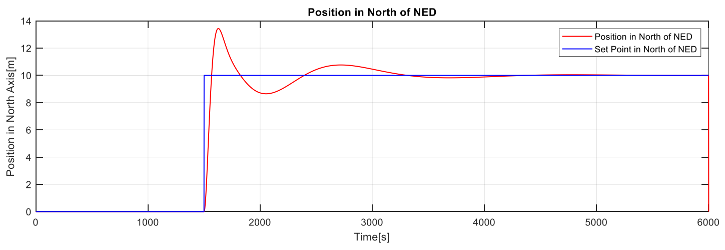
3.1.1 Coupling of States

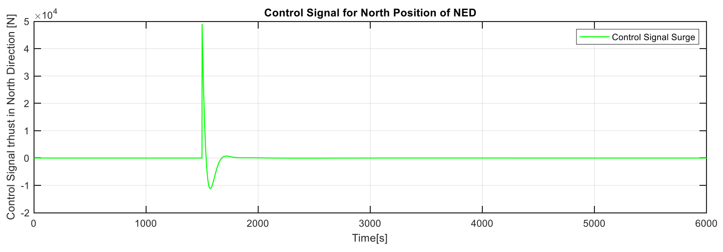
The Yaw of the ship has significance in the performance of the controllers for surge and sway. As the setpoint for yaw is increased, the response of the surge and sway controllers deteriorates. After a threshold, the position in north and east of the ship in the NED coordinate system is unsatisfactory.

When the Setpoint for Yaw is 20, the response of the three controllers is acceptable but starts showing instability this is portrayed in Figure 3-6. The oscillations start to show in the Surge and Sway controllers.









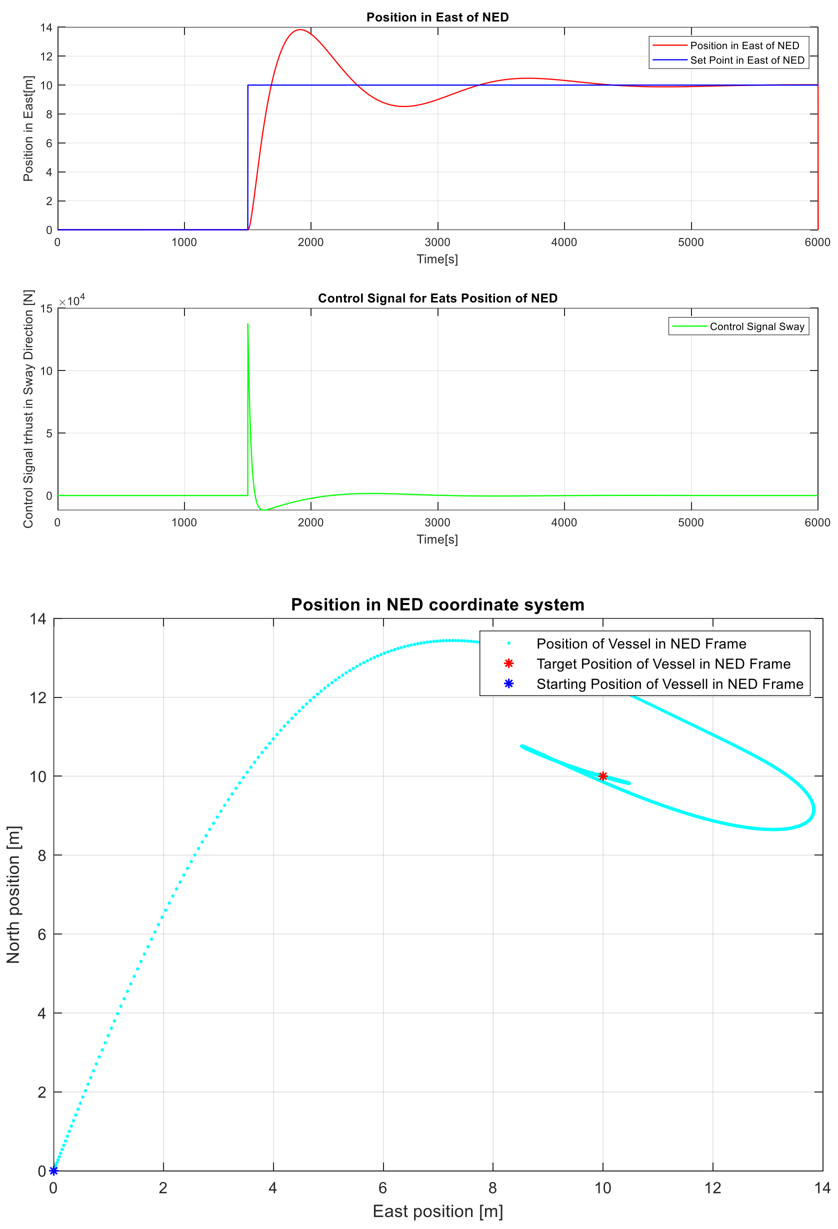
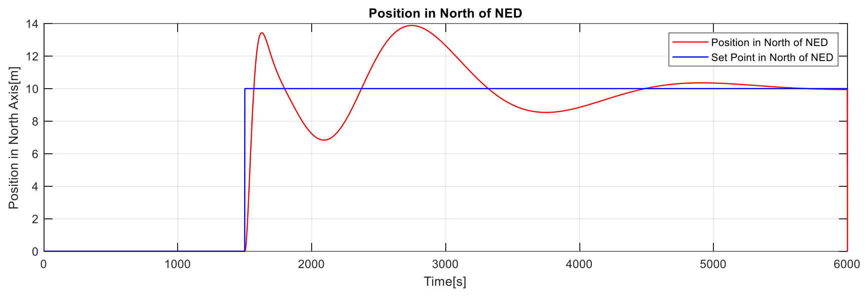
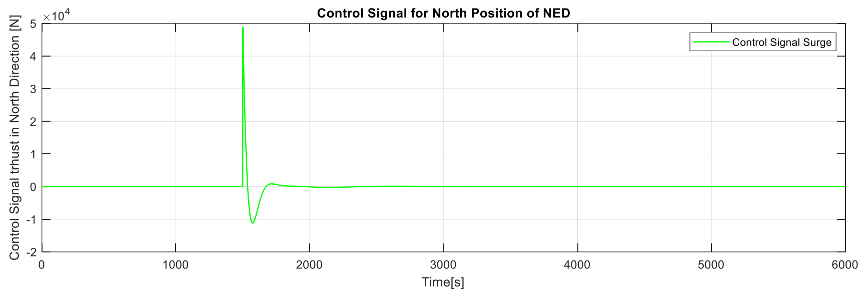
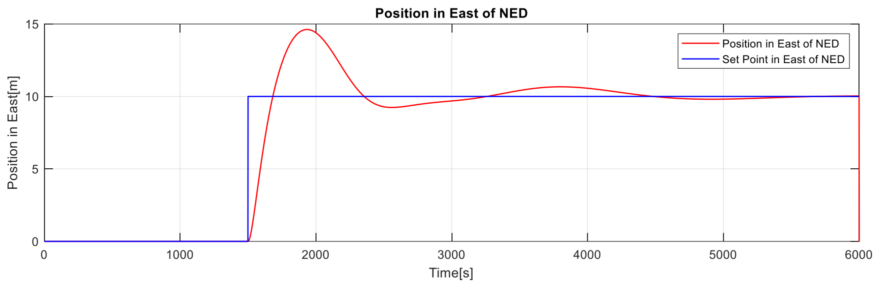


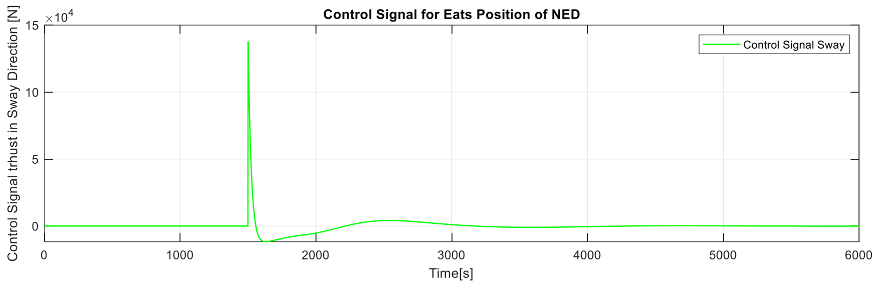
Figure 3-6 Behavior of Controllers when Yaw's Set Point is Set to 20 Degrees

The response is still adequate for the three controllers and at the end, the ship is positioned where it should be. However, when the yaw setpoint is increased to 60 degrees, the responses deteriorate:


Position in North of NED
Position in North of NED
Set Point in North of NED
Position in North Axis[m]
Time[s]
0
2
4
6
8
10
12
14
0
1000
2000
3000
4000
5000
6000


Control Signal for North Position of NED
Control Signal Surge
Control Signal trhust in North Direction [N]
Time[s]
$\times 10^4$
-2
-1
0
1
2
3
4
5
0
1000
2000
3000
4000
5000
6000


Position in East of NED
Position in East of NED
Set Point in East of NED
Position in East[m]
Time[s]
0
5
10
15
0
1000
2000
3000
4000
5000
6000


Control Signal for Eats Position of NED
Control Signal Sway
Control Signal trhust in Sway Direction [N]
Time[s]
$\times 10^4$
0
5
10
15
0
1000
2000
3000
4000
5000
6000

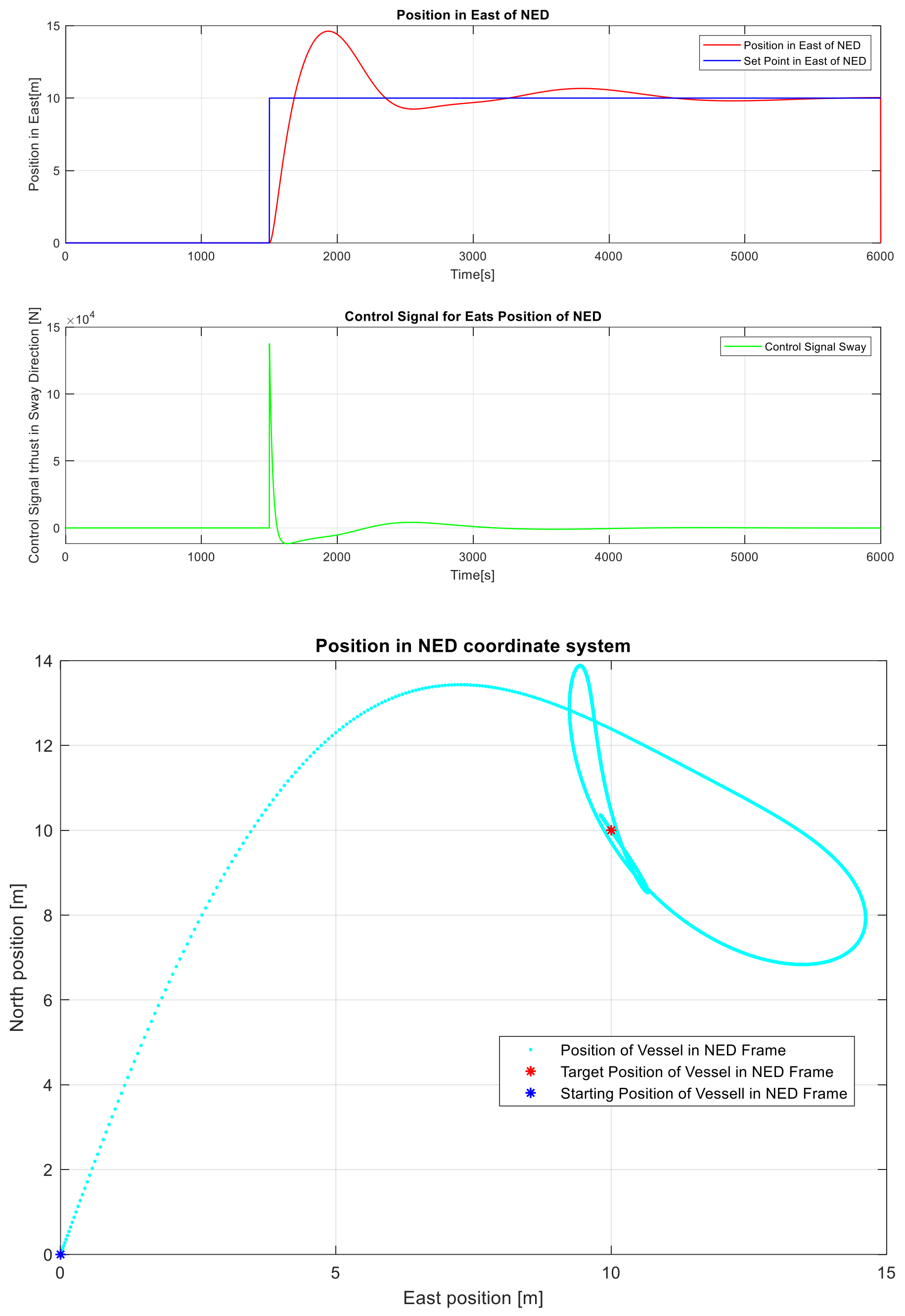


Figure 3-7. Behavior of Controllers When the Yaw's Set Point is 60 Degrees.

Figure 3-7 shows that, although the controller makes the vessel achieve the desired setpoint of 60 degrees in Yaw, the controller in Surge starts showing suboptimal performance, with oscillations in the positioning of the vessel.

It could be argued that the yaw could act as an additional disturbance in the model over the surge and sway subsystems.

3.2 PID Controllers and Disturbances

Now, having tuned the PIDs, it is necessary to add the disturbances; in this case, these will be low frequency components of wind and water current. To simulate them, a MATLAB class was created, with methods that simulate the wind speed and direction, and the water current in the north and east directions of the NED framework. The code to simulate the disturbances was obtained from [3].

3.2.1 Wind Disturbance

The simulation time is split in 4 and a random multiplier is selected for each segment. Then, this multiplier will multiply a selected average wind speed. This will create a vector of N wind speeds, which will be smoothed out using MATLAB's smooth function.

Then, another function will produce a vector of angles, which will represent the wind direction. This is needed to further simulate the angle of attack of the wind on the vessel.

3.2.2 Water Current Disturbance

For this disturbance, sinusoidal functions are used for each of the axis of the NED frame. These have an amplitude in m/s which is selected manually.

3.2.3 Response with the Disturbances

First, the amplitudes of the disturbances are selected as low values; with a wind speed of 5 m/s.

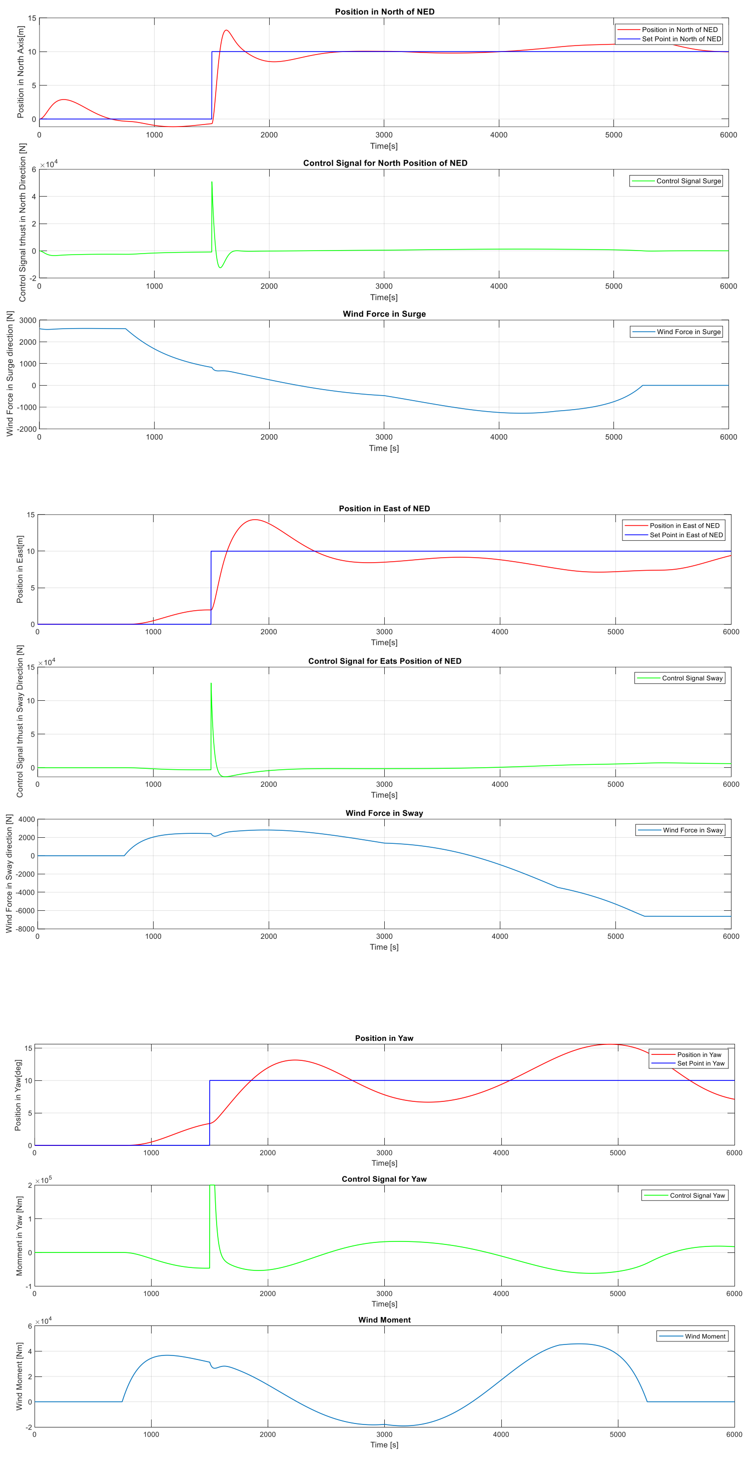


Figure 3-8 PID Controllers Response with 5 m/s Wind Speed

Figure 3-8 shows that the controllers let the vessel reach the north and east setpoints, but not the yaw setpoint. In general, for the three controllers, the tuning must be performed again when the average wind speed is increased.

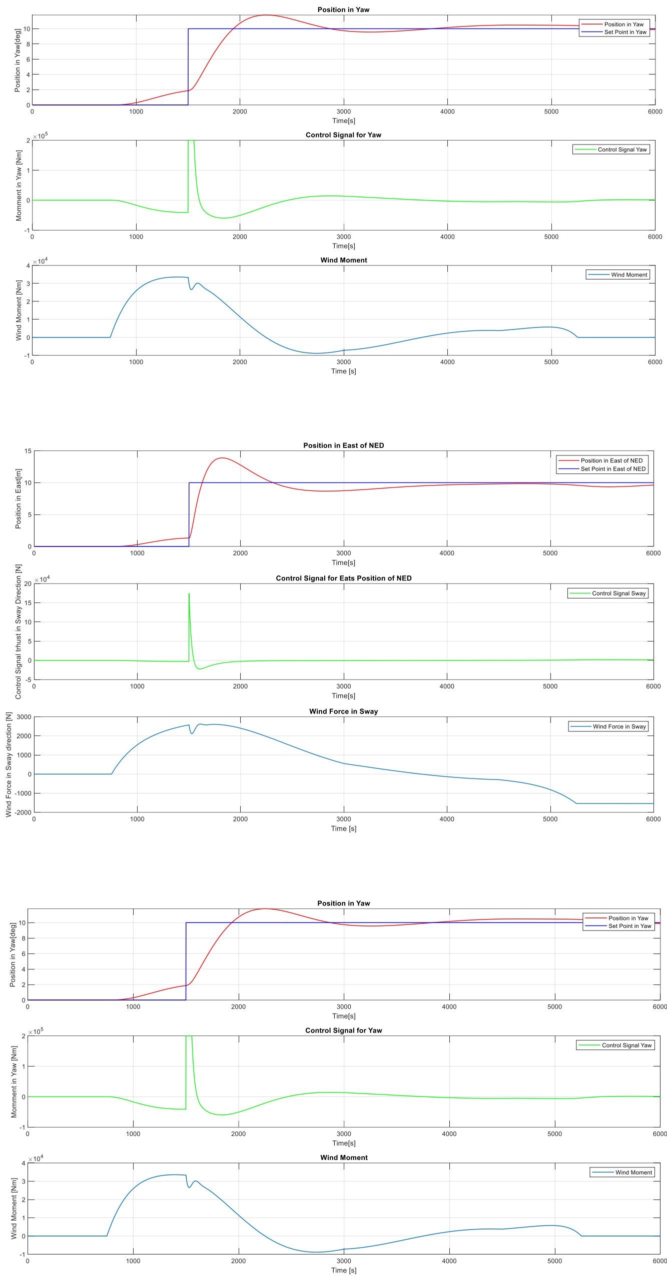


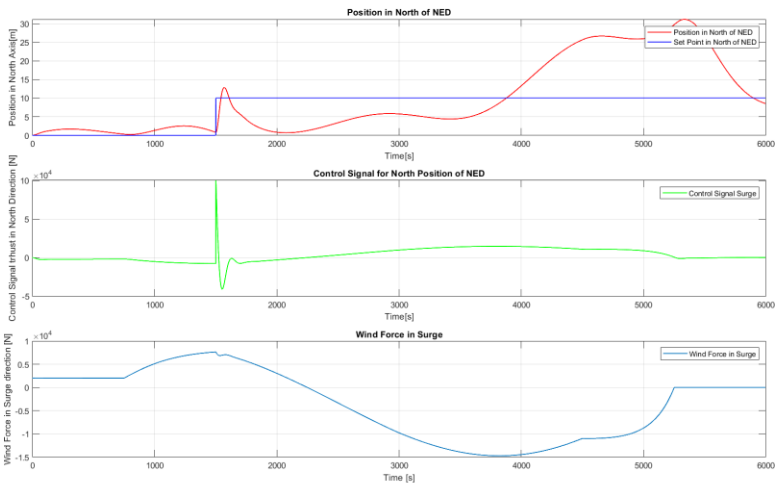
Figure 3-9 Controller's Response in Northe After Retuning

As seen in Figure 3-9, the setpoints are reached after using different parameters for the controllers, however, the performance of the system drops after introducing the disturbances. The current parameters of the Controllers are in Table 3-1.

Table 3-1 PID Parameters for 5 m/s average winds

Parameter	**Value**
Kp Surge PID	600
Ti Surge PID	200
Td Surge PID	300
Kp Sway PID	1200
Ti Sway PID	550
Td Sway PID	300
Kp Yaw PID	800000
Ti Yaw PID	1500
Td Yaw PID	350

Nevertheless, when the wind speed is increased, the behavior of the system deteriorates considerably. Now, the average wind speed will be increased to 20 m/s.



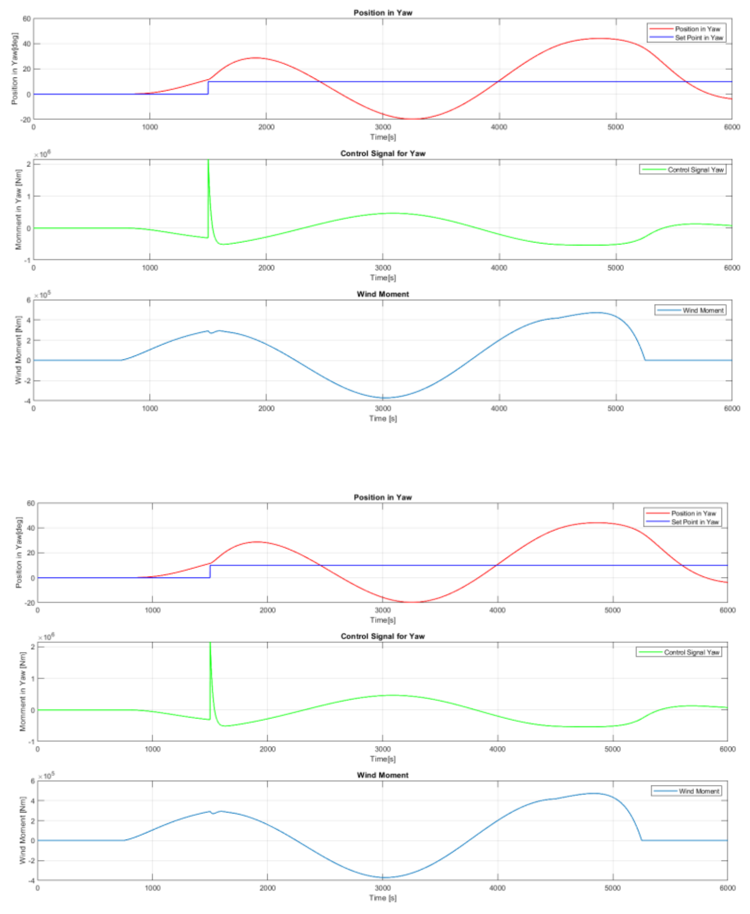


Figure 3-10 Controllers and System Response with Average Wind Speed of 20 m/s

It is possible to see in Figure 3-10 that the controllers are not able to take the vessel to the desired setpoints. The Surge controller gets to the setpoint later, but its performance is inadequate.

The parameters of the PID controllers need to be changed again to match the environment conditions, this is a trial-and-error process which stops when the behavior of the system is close to desired. With the magnitude of the disturbances, it is difficult to guarantee that the setpoints will be matched completely.

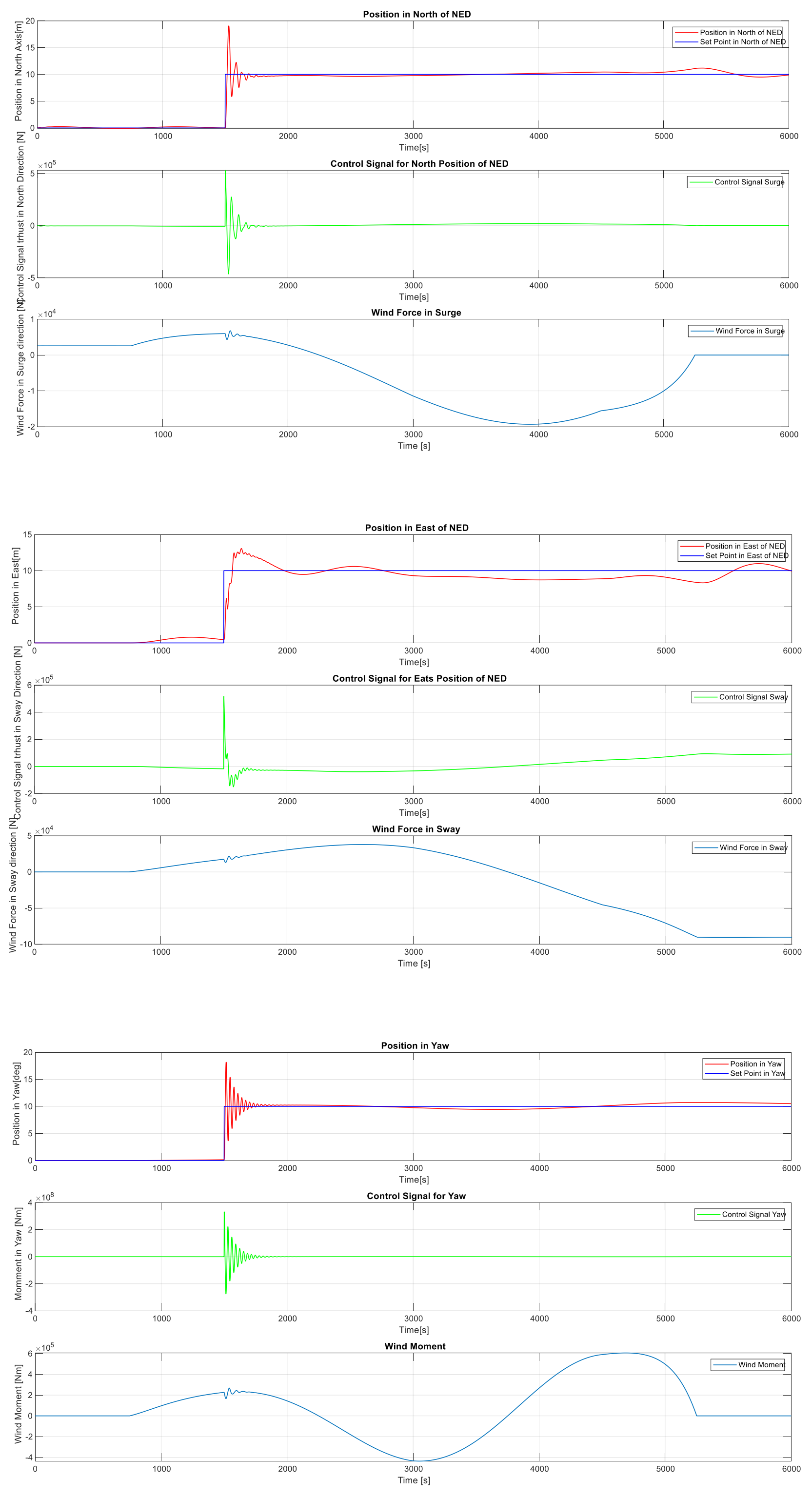


Figure 3-11 Performance of System with Average Wind Speed of 20 m/s and Tuned PID Controllers

In Figure 3-11, the controllers reach the setpoints, however, the Yaw controller displays severe oscillations right after the change in its setpoint, which means that it could be unstable. It can be also seen in Table 3-2 that a considerable increment in the proportional gain of the Yaw controller is needed.

Table 3-2 PID Controllers Parameters for Average Wind Speeds of 20 m/s

Parameter	**Value**
Kp Surge PID	600
Ti Surge PID	200
Td Surge PID	300
Kp Sway PID	1200
Ti Sway PID	550
Td Sway PID	300
Kp Yaw PID	800000
Ti Yaw PID	1500
Td Yaw PID	350

3.3 Viability of the Use of PID Controllers

Given the coupling of the yaw with the other states and the susceptibility of the system to disturbances, the sole use of PID controllers does not seem to be practical for Dynamic Positioning. It could be useful to have a way to integrate all the states in the same controller. Also, the use of feedforward control can improve the response of the controllers. It seems that with every set of disturbances or change in the yaw set point, there is a need to tune the controllers once again.

4 System Identification

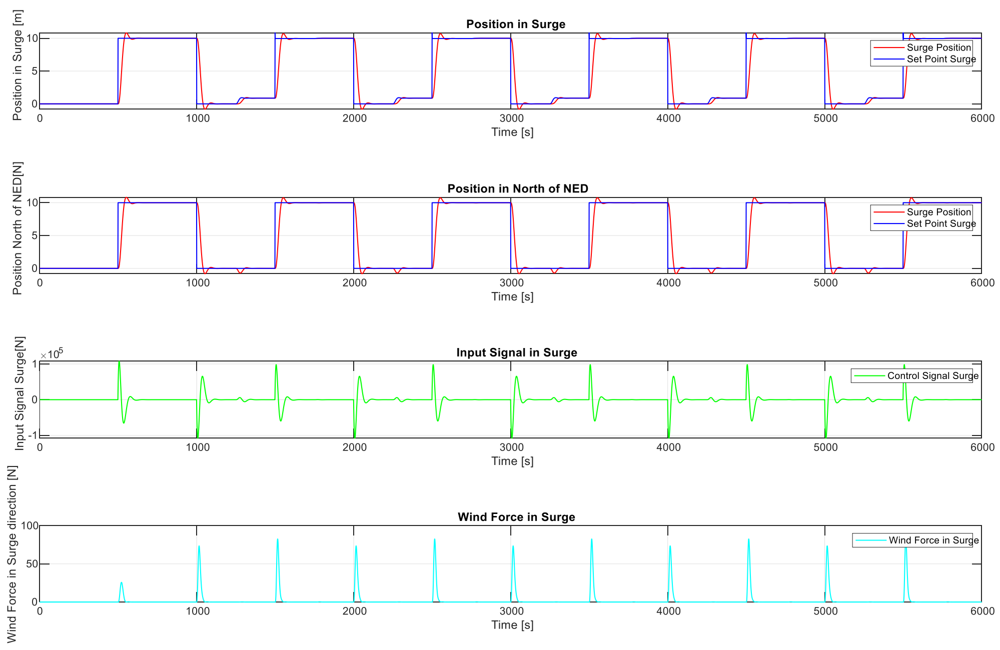
The system identification is performed through an experiment in which a set of impulses are generated in the input signals; these impulses will produce an effect on the model, which will produce a set of outputs. In the case of the vessel, the outputs are in the positions in north, east and yaw.

The system identification is performed on the closed loop model, which means that this will also account for the presence of a controller.

The result will be matrices A, B, D from the state space model.

4.1.1 Obtaining the Input and Output Signals

The linearized model is used to simulate the closed loop response of the system following a set of predefined setpoints. The resulting input values (u) and output (y) are stored in a .mat file, which will be used then for the system identification.



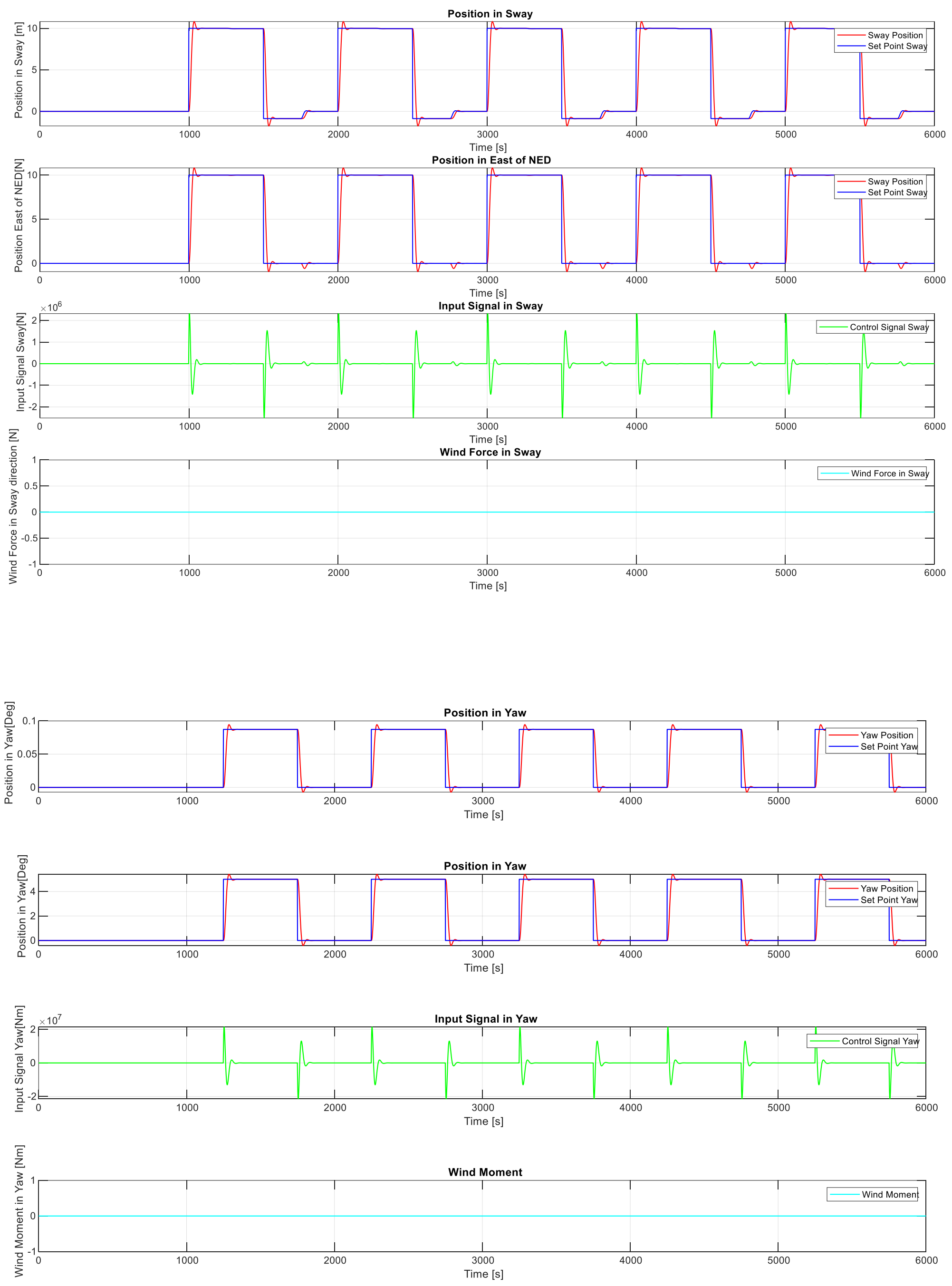


Figure 4-1 Setpoints, Input and Output Signals Using Linearized Model

In Figure 4-1 the behavior of the system for the three main states is observed. The setpoints in surge and sway resemble but do not match exactly step impulses as they were transformed to the vessel’s coordinate frame and then fed to the closed loop system before generating the control signals.

4.1.2 Using the U and Y signals in dsr_e function

The input and output arrays are then used in a system identification script. Using the dsr_e function from the DSR toolbox obtained from [8]. The purpose is to identify a deterministic and stochastic state-space model in the form:

$$x_{k+1} = Ax_k + Bu_k + Ce_k \tag{4.1}$$

$$y_k = Dx_k + Eu_k + Fe_k \tag{4.2}$$

Where x represents the states, u the control inputs, y the outputs, and e represents random error with unit variance $E(e_k {e_k}^T) = I$.

The dsr_e function receives as inputs the array of output signals, Y; the array of input values, U; The estimation horizon L; a g value, which is equal to zero if the function will work with a closed loop system; a number of past horizons, which in this case is equal to 2; and the model order, which for this case, is 6, as the system has 6 states.

The U and Y values are imported from a .mat file called sysIdData.mat. This file is created with another MATLAB script called input_output_generation.m, which generated the input and output signals and exported them.

When the U and Y values are read in a MATLAB script, these are then introduced in the dsr_e function to produce a model. The system identification is executed in a script called system_identification.m.

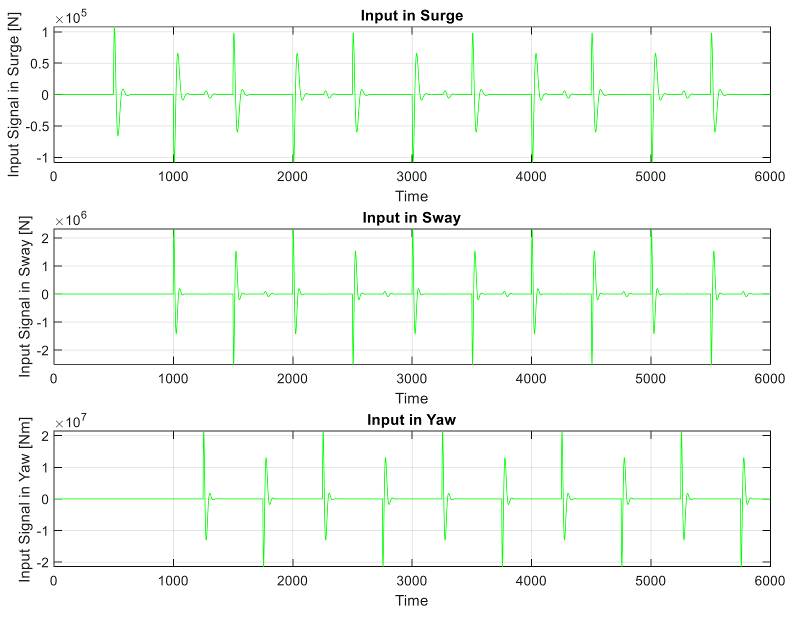


Figure 4-2 Input Signals for System Identification

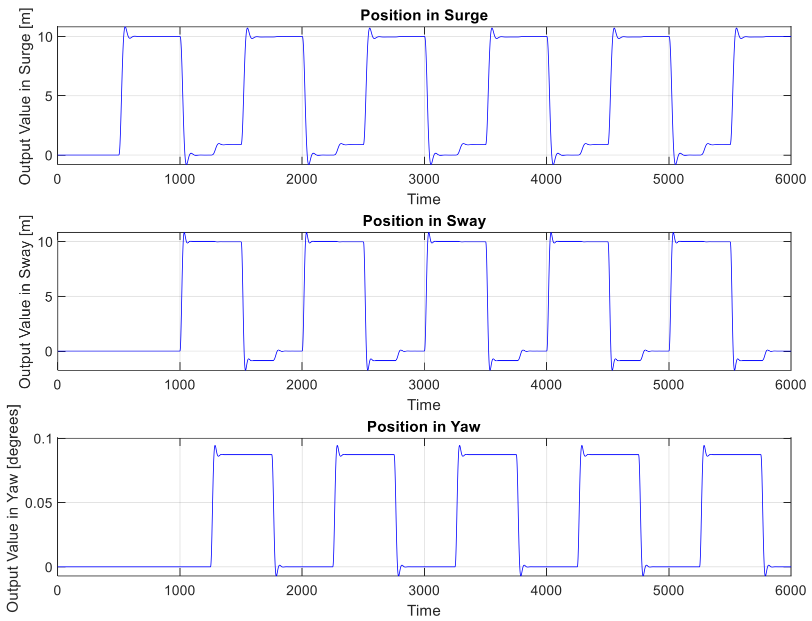


Figure 4-3 Output Values for System Identification

Figure 4-2 and Figure 4-3 display the input and output values used in the system_identification.m script, it is possible to notice that they have the same values as the arrays generated in the input_output_generation.m script. The system_identification.m script will generate the matrices for the state space model.

4.1.3 The resulting Model

After the executing the system_identification.m script, found in appendix A. Matrices A, B, and D as per equations (16) and (17) are obtained, with the values as shown in Table 4-1, Table 4-2, and Table 4-3 respectively:

Table 4-1 System Identification A Matrix Values

1.000116	-0.00059	0.592608	0.186525	1.055447	0.005346
-4.37E-05	1.000865	-1.01657	-0.28442	0.62102	0.004036
-3.11E-06	7.19E-06	0.999288	-6.41E-05	-0.00144	0.339657
1.08E-05	-2.20E-05	-0.00082	0.9993	0.006541	-1.17656
-1.22E-07	4.46E-07	-8.42E-05	-1.71E-05	0.999857	0.017156
-3.28E-09	-1.12E-08	2.49E-05	7.38E-06	-5.55E-06	1.00056

Table 4-2 System Identification B Matrix Values

-6.33E-07	-1.44E-08	-1.88E-13
1.44E-07	-6.34E-08	-1.83E-13
-2.31E-07	2.48E-08	-1.53E-11
-7.31E-08	6.55E-09	5.31E-11
-2.57E-07	-2.42E-08	-7.74E-13
1.08E-10	-6.09E-12	-3.01E-11

Table 4-3 System Identification D Matrix Values

-0.562867	0.1277356	0.4646695	0.1393493	0.5146498	-0.0002304
-0.128179	-0.562528	-0.496132	-0.138544	0.4848966	0.0001565
-0.002698	-0.001728	-0.159969	0.555015	-0.0080428	0.7068428

According to [9], the system is observable and controllable if the controllability and observability matrices have the same number of singular values as states of the system, which in this case is 6. The observability matrix is formed using matrices A, and D; whereas the controllability matrix is made of matrices A, and B.

Table 4-4 System Identiffication Observability Matrix

4.0542
4.054
4.0538
1.0319
1.0319
1.0319

Table 4-5 System Identification Controllability Matrix

4.68E-06
4.68E-07
3.35E-07
3.40E-08
0.00E+00
0.00E+00

The result of the single value decompositions of the observability and controllability matrices can be seen in Table 4-4, and Table 4-5 respectively.

For the observability matrix, all the single values are bigger than zero. This means that all the states of the system are observable. However, for the controllability matrix, only the four first single values are bigger than zero. So, only the first four states of the system are controllable. This does not make the control algorithms inviable since there are only three outputs to be controlled, which are the same as the first three states of the closed loop system.

The performance of this identified model will now be tested by using it in the LQ with integral action and the MPC with integral action control algorithms.

5 LQ Optimal Control with Integral Action

LQ is the short version of Linear Quadratic optimal control, where the goal is to reduce a cost function in the form of:

$$J_i = \frac{1}{2}\sum_{k=i}^{\infty}((y_k - r)^T Q(y_k - r) + \Delta u_k^T P \Delta u_k) \tag{5.1}$$

As explained in [10], to make a model that is independent to the unknown disturbances; integral action must be included, which is achieved by augmenting it. This augmented model can be expressed in the discrete state-space form, but first, some modifications need to be set in place:

$$\underbrace{\begin{bmatrix} \Delta x_{k+1} \\ y_k - r \end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix} A & 0_{n\times m} \\ D & I_{m\times m} \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \Delta x_k \\ y_{k-1} - r \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B \\ 0_{m\times r} \end{bmatrix}}_{\tilde{B}} \Delta u_k \tag{5.2}$$

$$\underbrace{y_k - r}_{\tilde{y}_k} = \underbrace{\begin{bmatrix} D & I_{m\times m} \end{bmatrix}}_{\tilde{D}} \underbrace{\begin{bmatrix} \Delta x_k \\ y_{k-1} - r \end{bmatrix}}_{\tilde{x}_k} \tag{5.3}$$

With this augmentation, the state-space model can be expressed as:

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}\Delta u_k \tag{5.4}$$

$$\tilde{y}_k = \tilde{D}\tilde{x}_k \tag{5.5}$$

The use of the state-space model in equations (5.4) and (5.5), and the cost function in equation (5.1) represents the LQ optimal control problem, which can be synthetized like:

$$\Delta u_k = \begin{bmatrix} G_1 & G_2 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ y_{k-1} - r \end{bmatrix} \tag{5.6}$$

Or,

$$u_k = u_{k-1} + G_1 \Delta x_k + G_2(y_{k-1} - r) \tag{5.7}$$

It is not common to be able to measure the states. That is why an observer can be included in the deviation model. The difference between the estimated states of two iterations will be used in the second term of equation (5.7).

$$\Delta\bar{x}_{k+1} = A\Delta\bar{x}_k + B\Delta u_k + K(y_k - y_{k-1} - D\Delta\bar{x}_k) \tag{5.8}$$

Where K is a Kalman gain.

5.1 Implementation of LQ Optimal Control in MATLAB

The LQ optimal control is implemented in a script called dlq_integral.m. It unpacks A, B, and D matrices from the identified system, which were stored in the closedLoopModel.mat file. Then used them to create a Kalman gain matrix and to extend the state-space model to include the integral action and to generate the G1 and G2 matrices.

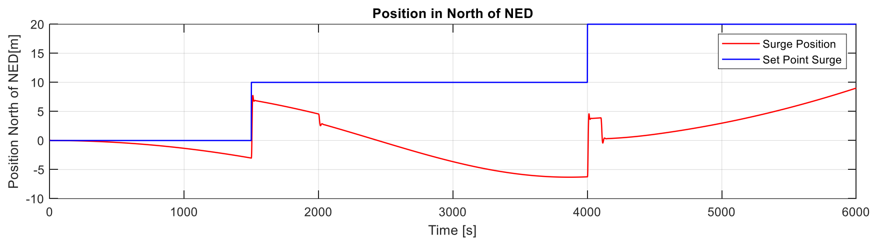
5.1.1 Finding the Kalman Filter Gain

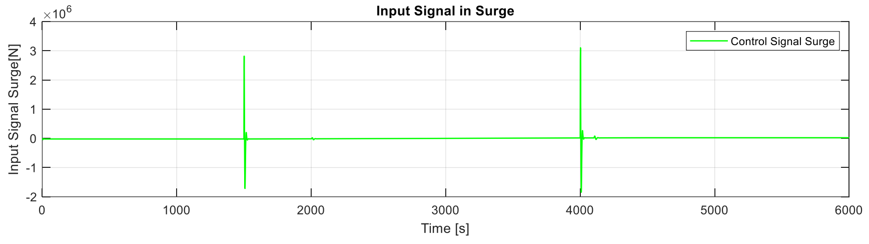
To obtain the Kalman gain, which will be used in the state estimation, the dlqe function from the control toolbox [10] Is used.

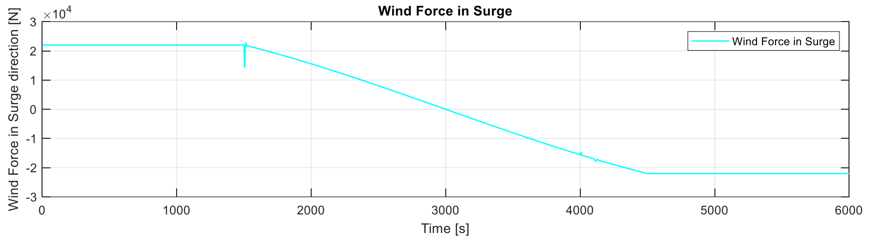
```
%% kalman filter gain
% The kalman filter is recalculated, as the K matrix from the identified
% model does not perform well during the state estimation.
G=0.01*eye(6);
Q_k = diag([0.09, 0.11, 0.09,0.8,0.5,1.5]);
R_k = diag([1e3,1e3,1e3]);
K=dlqe(A,G,D,Q_k,R_k);
```

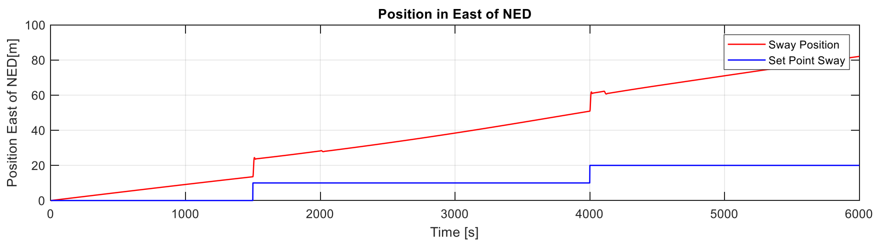
The dlqe function receives as arguments the matrices A, and D from the identified state space model; a matrix G which accounts for noise, with the same size as there are states, so 6; and Q_k, and R_k which are process and sensor noise covariance matrices.

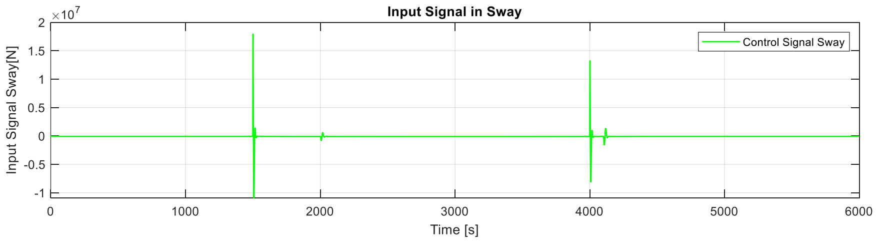
Although the dsr_e function the system identification generates a K matrix as output, the performance of the system with this is not adequate. Figure 5-1 shows the behavior of the three outputs while using the Kalman gain matrix obtained from the system identification. The controllers never achieve their setpoints.

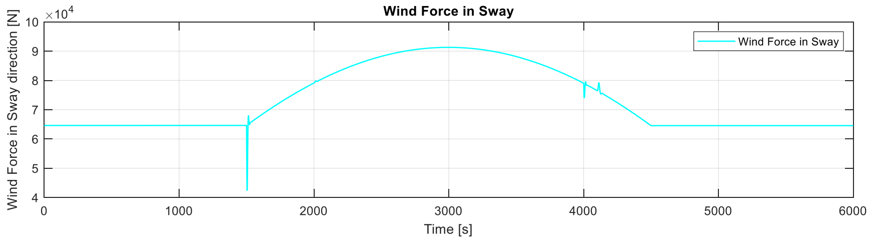

Position in North of NED
Position North of NED[m]
Time [s]
Surge Position
Set Point Surge

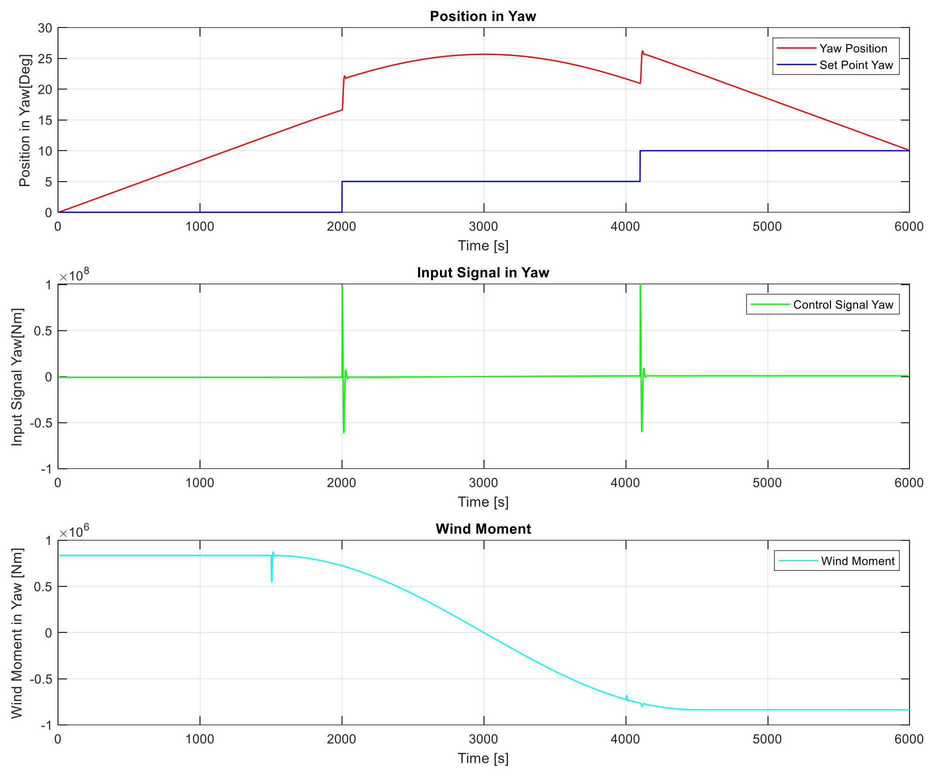

Input Signal in Surge
Input Signal Surge[N]
×10^6
Time [s]
Control Signal Surge


Wind Force in Surge
Wind Force in Surge direction [N]
×10^4
Time [s]
Wind Force in Surge


Position in East of NED
Position East of NED[m]
Time [s]
Sway Position
Set Point Sway


Input Signal in Sway
Input Signal Sway[N]
×10^7
Time [s]
Control Signal Sway


Wind Force in Sway
Wind Force in Sway direction [N]
×10^4
Time [s]
Wind Force in Sway



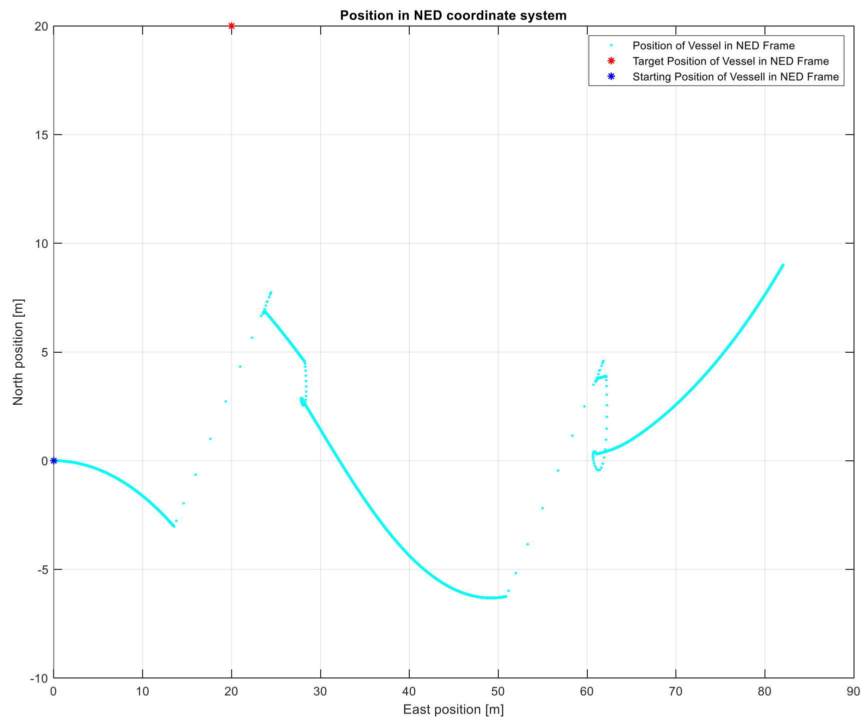
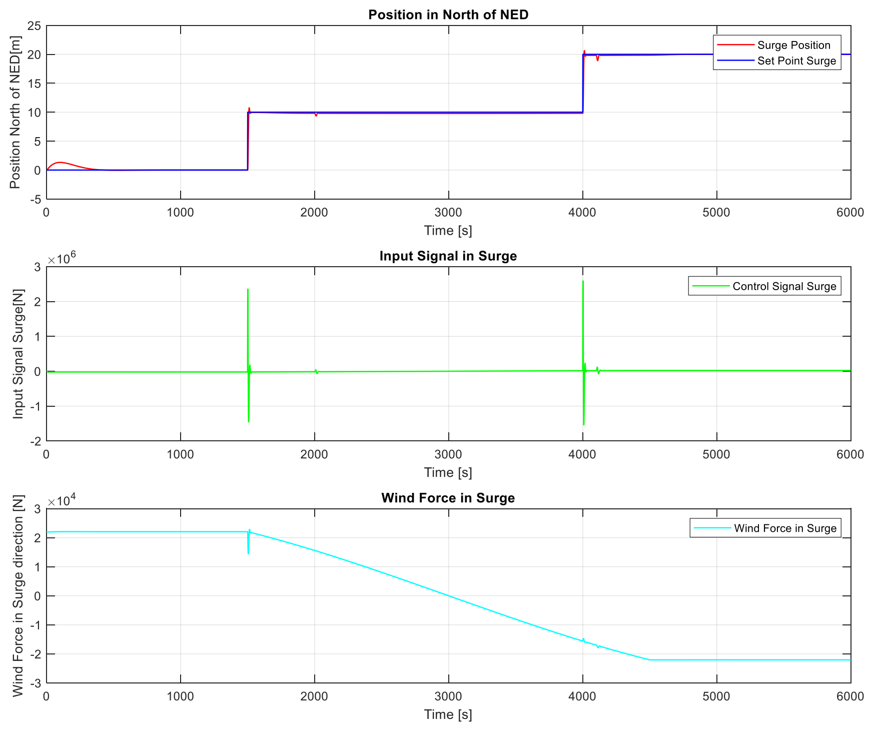
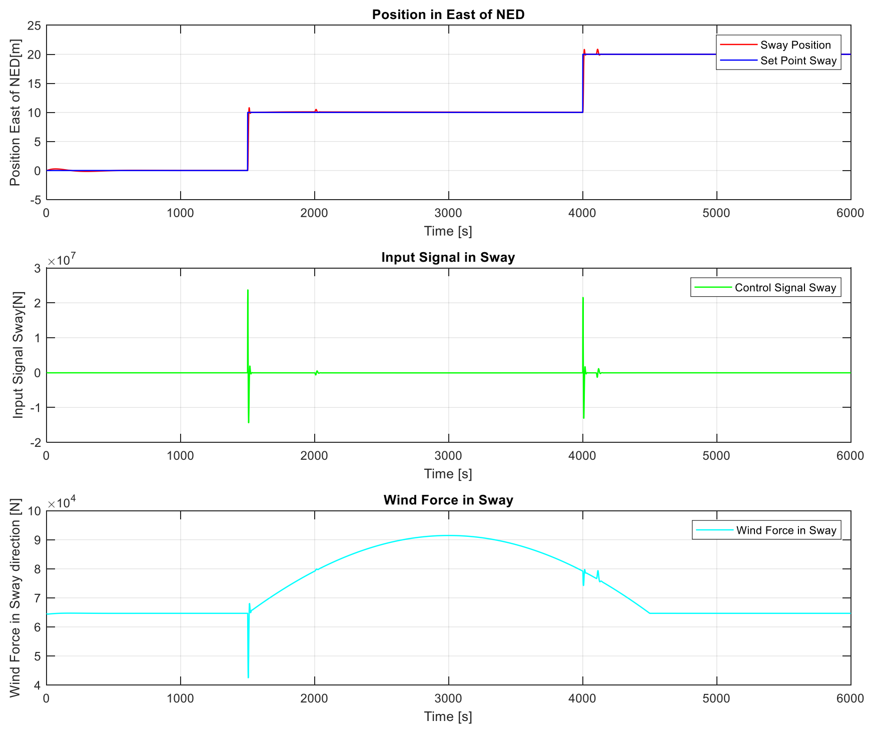


Figure 5-1 Response of the System with LQ and K from Identified Model

This behavior improves when the Kalman gain used is obtained from the dlqe function, as seen in Figure 5-2. The controllers maintain the vessel on the setpoints almost during the whole simulation.





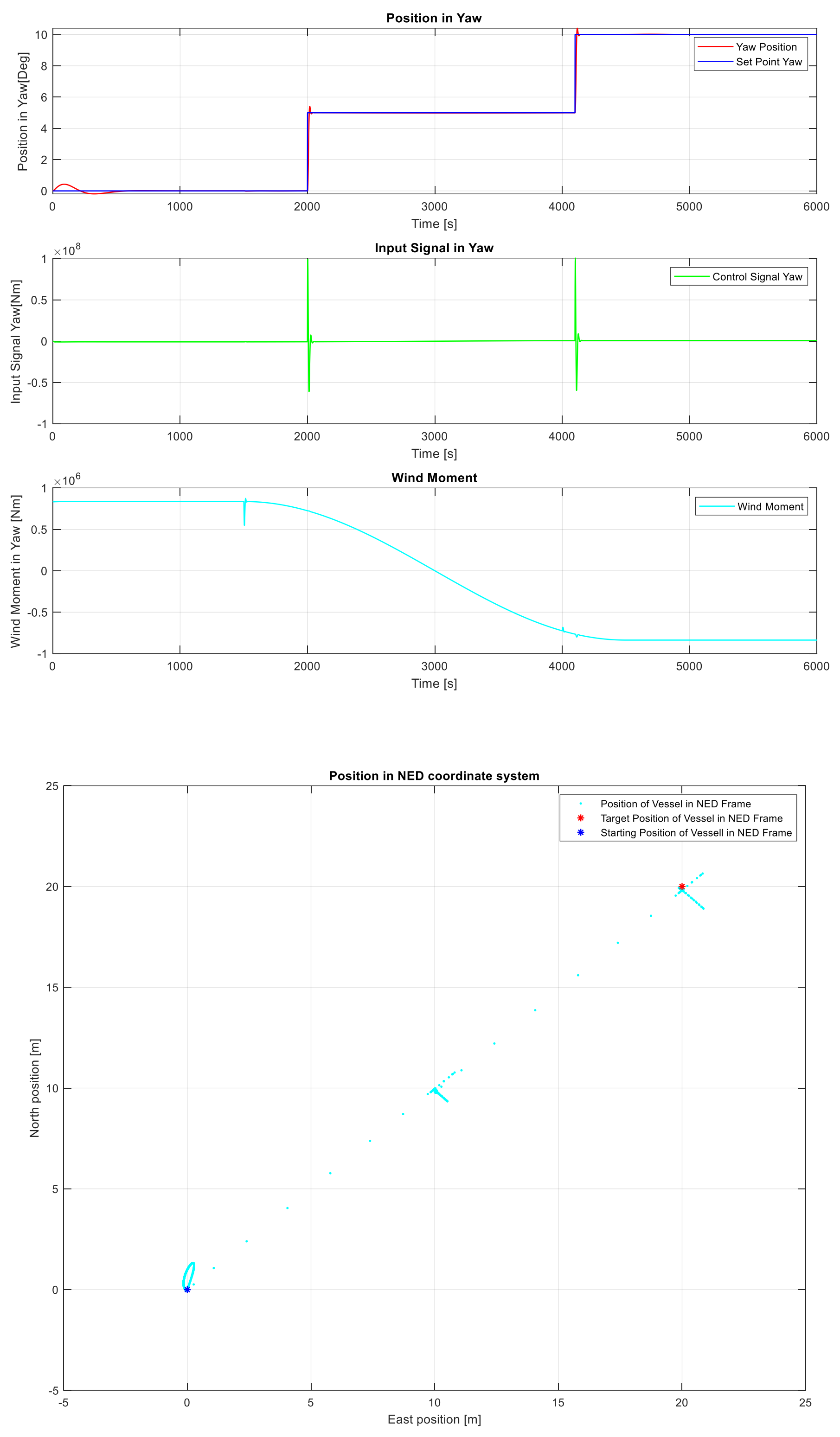


Figure 5-2 Response of the System with LQ and K from dlqe Function

5.1.2 G Matrix and Extended Model

The dlqu_pi.m function from [10] is used. This function receives as inputs A, B, D matrices from the identified model, and column vectors q and p, which are used to create Q and P diagonal weight matrices. It then creates extended A_t, B_t, and D_t matrices, which are used in the dlqr function to find the G matrix.

```
%% Obtaining the G matrix
q = [1e6 0 0; 0 1e7 0; 0 0 1e8];
p = [1e-5 0 0; 0 1e-6 0; 0 0 1e-10];

[G1, G2, At, Bt, Dt, Rr]=dlqdu_pi(A,B,D,q,p);
```

Inside the dlqu_pi.m function:

```
function [G1,G2,At,Bt,Dt,Rr]=dlqdu_pi(A,B,D,Q,Rw);
%% Make augmented state space model for LQ-design.
nx=size(A,1); nu=size(B,2); ny=size(D,1);
At=[A,zeros(nx,ny);D,eye(ny,ny)];Bt=[B;zeros(ny,nu)];Dt=[D,eye(ny,ny)];
Qt=Dt'*Q*Dt;

%% Solve Riccati-equation and compute feedback matrix.
[K,Rr]=dlqr(At,Bt,Qt,Rw);
G=-K;
G1=G(:,1:nx); G2=G(:,nx+1:nx+ny);
```

5.1.3 Calculation of Control Signal, Update of Outputs and State Estimation

Inside the main loop, the control signal is calculated, then it is used along with the current values of the disturbances to simulate the model’s response. Inside the iteration, the states are estimated once again and the output values are updated.

```
%Obtaining the updated control signal
    u = u + G1*(x_est-x_old) + G2*(y_old - sp_ship);

    U(:,i)=u; %Storing the control signal for ploting

    x_old = x_est; %Updating the old states for the next iteration
    y_old = y; %Uptading the outputs for the next iteration

%Obtaining the value of the weather disturbance for plotting
    F_w(:,i) = model.WindForce(W_v(i),gamma(i), x(3), x(4), x(5), A_F, A_L, L);

%Calculating the water current force for plotting
    V_c(:,i)= model.CurrentVelocity(W_c(:,i),x);

%Generating the non-linear model's response
    [x, NED_Position] = model.UpdateState(u, W_c(:,i), W_v(i), gamma(i), dt);

%Updating the state estimation
    x_est = A*x_est + B * u + K*(y - D*x_est);
```

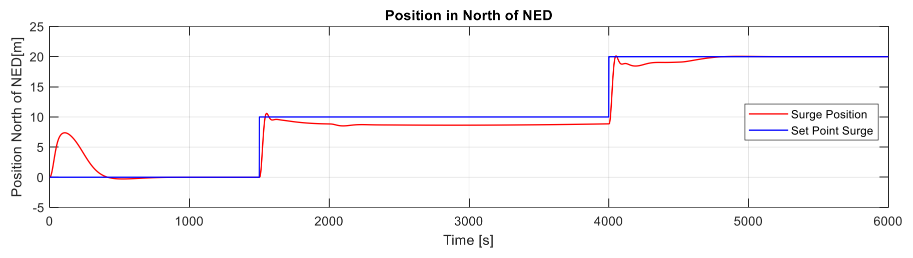
5.1.4 Selection of q and p weights

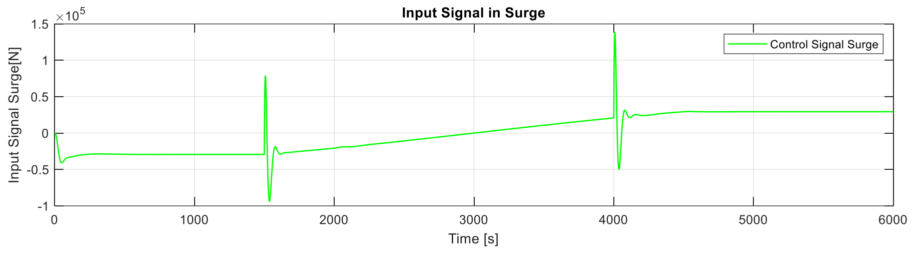
The selection of the q and p values could be described as trial-and-error. The q and p vectors represent the weights given in the algorithm to minimize the error and the change in the control signal respectively. In other words, these values determine how important the suppression of the error or keeping a small change in the u values are. These values are relative to each other, so a bigger q and a smaller p state that there is a much bigger priority in keeping the error in check than keeping smooth changes in u.

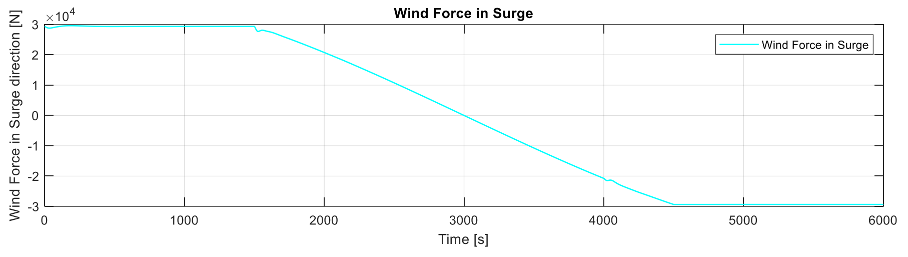
Let us say, that the following values of q and p are used:

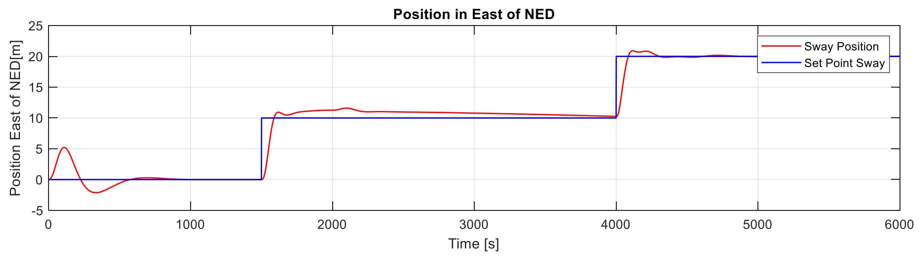
```
% Controller weights
q = [1e4 0 0; 0 1e4 0; 0 0 1e6];
p = [1e-3 0 0; 0 1e-3 0; 0 0 1e-6];
```

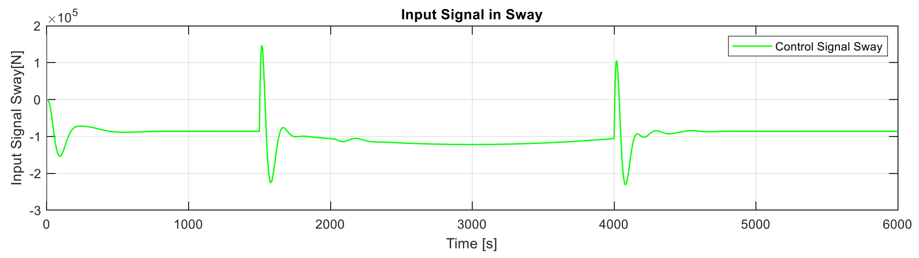
In this case, for a wind speed of 15 m/s, the behavior of the system would look like in Figure 5-3. The response of the controllers does not align the vessel completely to the set points, while the disturbances manage to deviate the vessel. However, the magnitude of the control impulses is smaller compared to the values in Figure 5-4, which indicates that this configuration could use less energy in the dynamic positioning.

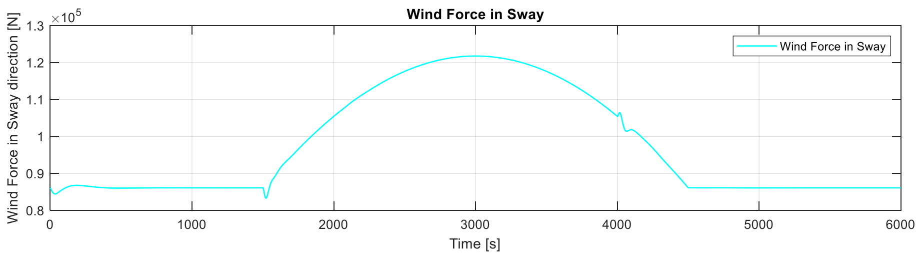

Position in North of NED
Position North of NED[m]
Time [s]
Surge Position
Set Point Surge


Input Signal in Surge
×10^5
Input Signal Surge[N]
Time [s]
Control Signal Surge


Wind Force in Surge
×10^4
Wind Force in Surge direction [N]
Time [s]
Wind Force in Surge


Position in East of NED
Position East of NED[m]
Time [s]
Sway Position
Set Point Sway


Input Signal in Sway
×10^5
Input Signal Sway[N]
Time [s]
Control Signal Sway


Wind Force in Sway
×10^5
Wind Force in Sway direction [N]
Time [s]
Wind Force in Sway

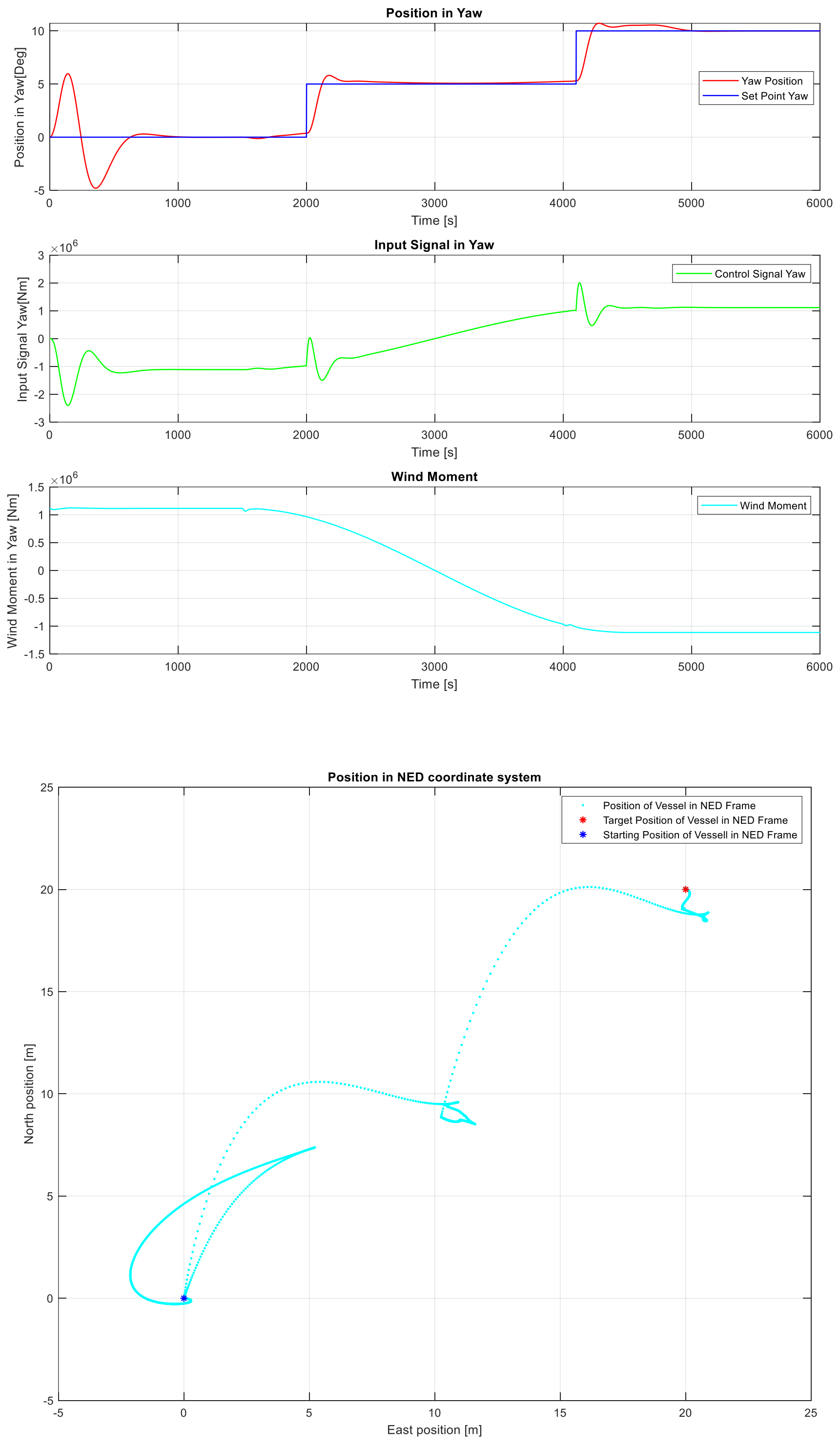
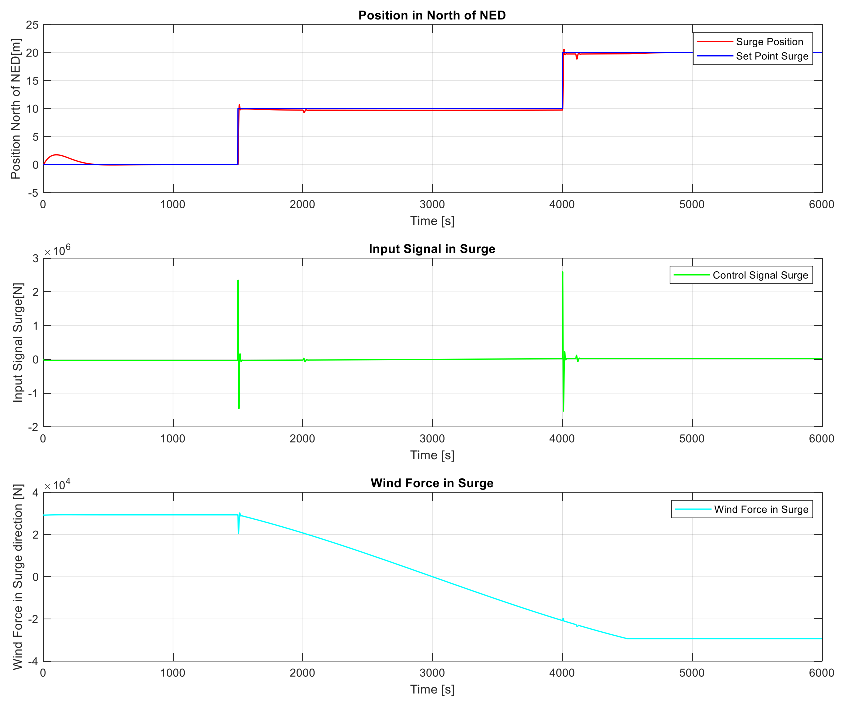
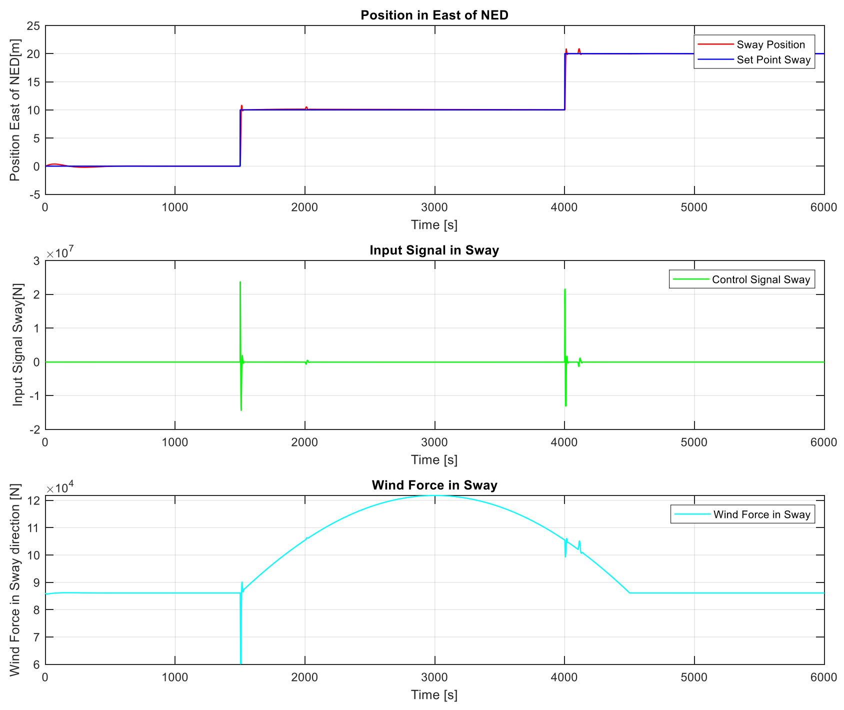


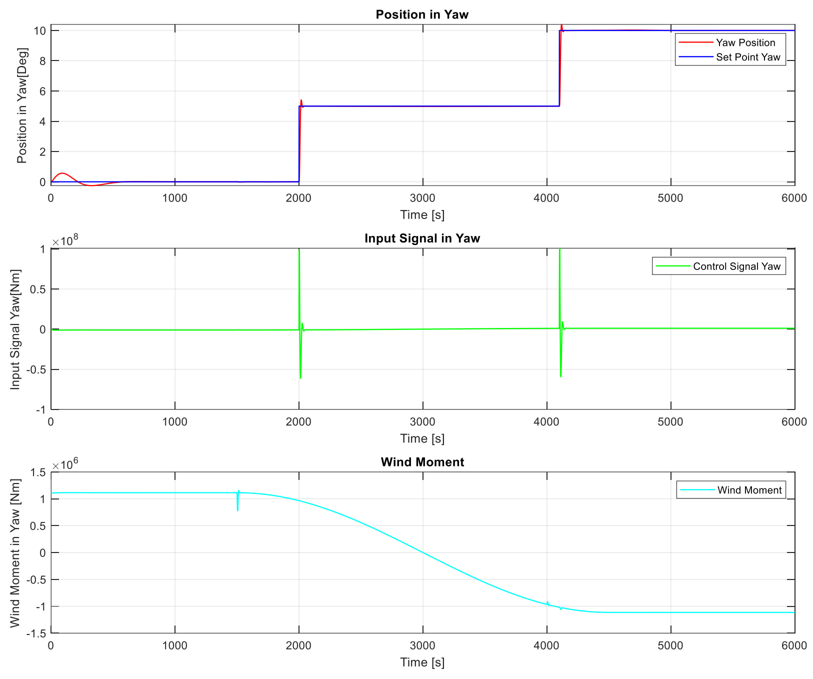
Figure 5-3 Behavior of LQ System with Relatively Higher p Values

Now, running the simulation with the original values, where the p vector has considerably lower values:

```
%% Obtaining the G matrix
q = [1e6 0 0; 0 1e7 0; 0 0 1e8];
p = [1e-5 0 0; 0 1e-6 0; 0 0 1e-10];
```







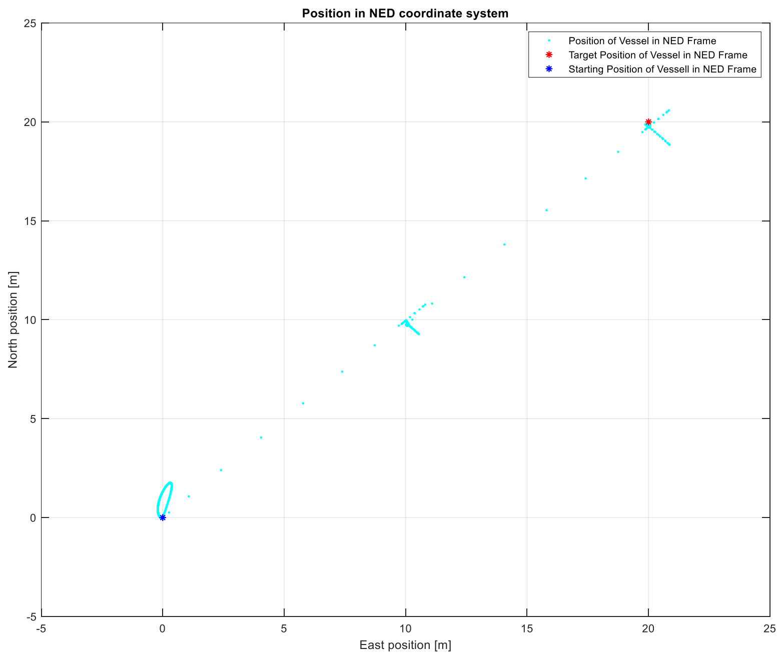


Figure 5-4 Behavior of the System with Relatively Lower p Values

Figure 5-4 displays the response of the system with higher q values compared to the p values. In this case, error suppression is prioritized. In which, the controller performs a considerably better job. Nevertheless, the absolute values of the control signals are one order of magnitude higher.

At the end, there needs to be a balance between the error tolerance and the amount of energy used to keep that error to a minimum. It is also worth noting that the wind disturbance is 15 m/s, which is in the upper bounds of the winds found in the northern sea, meaning that it would be an extreme case, were most likely, the use of dynamic positioning would not be enough.

6 MPC with Integral Action

MPC is the acronym for Model Predictive Control, where a linearized model of the vessel is found and then used to obtain an input signal based on a simulated response of this linearized model to the setpoints, the current state of the disturbances, and preestablished inequality and equality constraints. This calculation of the response and of the input signal is performed in every timestep for L future timesteps. L represents the prediction horizon; namely, a set of predicted future responses of the model based on its current state.

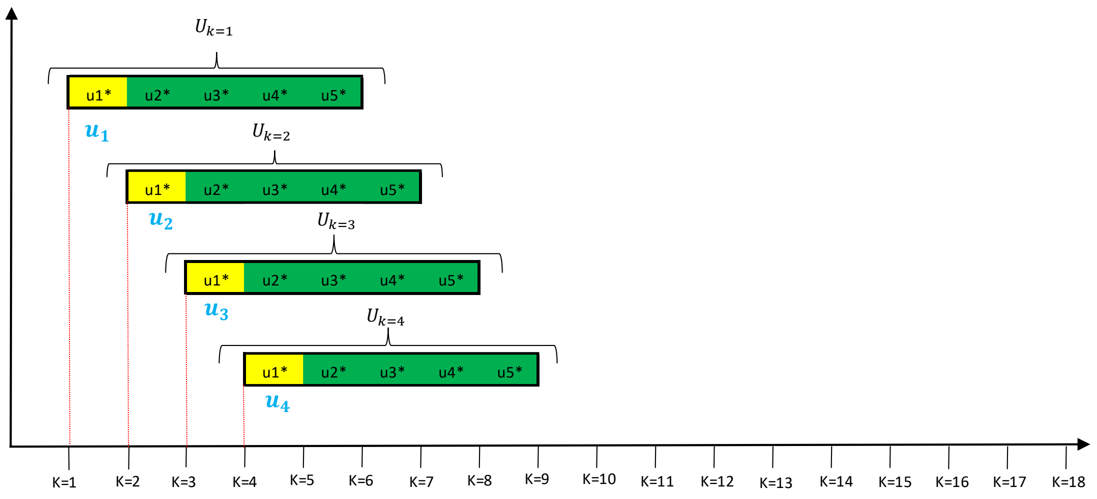


Figure 6-1 Representation of the Sliding Horizon and Different Input Values

Figure 6-1 illustrates how MPC with and sliding horizon strategy works: In every timestep, an array of input signals with a length of 5 is calculated. This means that the prediction horizon is of 5 timesteps. Then, only the first element of the array is selected and used as input in the controlled system. The process is repeated in the next time step.

The value of u, or the input signal in every timestep is obtaining by minimizing a cost function:

$$J_k = \sum_{i=1}^{L} (y_{k+i} - r_{k+i})^T Q (y_{k+i} - r_{k+i}) + \Delta u_{k+i-1}{}^T P \Delta u_{k+i-1} \tag{6.1}$$

In equation (6.1), L is the prediction horizon. The cost function J_k is calculated in every time step. Q and P are diagonal weighting matrices will determine which goal Is pursued more aggressively; whether reducing the offset between y and r or controlling the size of the changes in u for every timestep.

As with LQ optimal control, an extended model is used to include the integral action. In this case, an extended model will be implemented too, but in a Δu formulation:

$$\underbrace{\begin{bmatrix} \Delta x_{k+1} \\ y_k \end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix} A & 0_{n\times m} \\ D & I_{m\times m} \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B \\ 0_{m\times r} \end{bmatrix}}_{\tilde{B}} \Delta u_k \tag{6.2}$$

$$y_k = \underbrace{[D \quad I_{m \times m}]}_{\tilde{D}} \underbrace{\begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix}}_{\tilde{x}_k} \tag{6.3}$$

With the formulation in Equations (6.2) and (6.3), the state-space model can be augmented as:

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}\Delta u_k \tag{6.4}$$

$$y_k = \tilde{D}\tilde{x}_k \tag{6.5}$$

Parting from the augmented state space model, a prediction model can be formulated:

$$y_{k+1|L} = F_L u_{k|L} + p_L \tag{6.6}$$

Where

$$F_L = [O_L B \quad H_L^d] \tag{6.7}$$

$$p_L = O_L A x_k \tag{6.8}$$

L is the prediction horizon, F_L is a matrix whose values are constant and is calculated from the augmented state space model, p_L is a vector that depends on older inputs and outputs.

According to [11]. The problem can be formulated in terms of u, or which would be Δu for the augmented model.

$$J = u^T H u + 2f^T u + J_0 \tag{6.9}$$

Where

$$H = F^T Q F + P \tag{6.10}$$

$$f = F^T Q(p - r) \tag{6.11}$$

$$J_0 = (p - r)^T Q(p - r) \tag{6.12}$$

The derivative of equation (6.9) can be obtained as:

$$\frac{\partial J}{\partial u} = 2Hu + 2f \tag{6.13}$$

Which means that:

$$u^* = -H^{-1} f \tag{6.14}$$

6.1 Implementation of Integral MPC in MATLAB

In this case, equation 27 will have a change, in which Δx_{k+1} will correspond to the change in the state estimation from timestep to timestep. For this, it is necessary to use a state estimator that will work inside the main loop; in this case, a Kalman filter, whose constant K needs to be defined:

```
%% Calculating Kalman filter gain
if  iflag==0 || iflag==1
% if iflag==1
    % 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);
end
```

Again, for this controlling method, it is needed to define weights:

```
% Controller weights
%Weighting parameters, p is for delta u, q is for error
q = [1e6 0 0; 0 1e7 0; 0 0 1e8];
p = [1e-4 0 0; 0 1e-4 0; 0 0 1e-10];
```

Then, the augmented model is created with the imported A, B, and D matrices from the identified model in state space form.

```
%% augmented system matrices for Integral MPC:
%x ̃_(k+1)= A x̃ ̃_k+ B Δu_k
%y _k= D x̃ ̃_k
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; % From Theory: H= F^T QF+PH= F^T QF+P
```

Inside the main loop, the p_L vector is calculated in every timestep, it has a length of L, which is the prediction horizon. Also, the next L points in the setpoint vector are selected and used with p_L to calculate f and in turn, Δu_k:

```
    %Calculation of pL
    pL=OL*At*xt;

    %Selection of L next setpoints for comparison
    ref = sp(:,i+1:L+i);
    for k=1:L
        %The selected setpoints need to be converted to the vessel's frame
        ref(:,k)=R'*ref(:,k);
    end
    %Adding just the first reference to the setpoint's vector
    sp_ship_vector(:,i)=ref(:,1);

    ref = ref(:);
    %Finding the value of f
    f=FL'*Qt*(pL-ref);
```

Then, the value of Δu_k is calculated. According to the sliding horizon strategy, only the first value of u is selected and used in the model:

```
    %du = quadprog(H,f);
    du=-inv(H)*f;
    du = reshape(du,3,L); %arranged control inputs
    u = u + du(:,1);
```

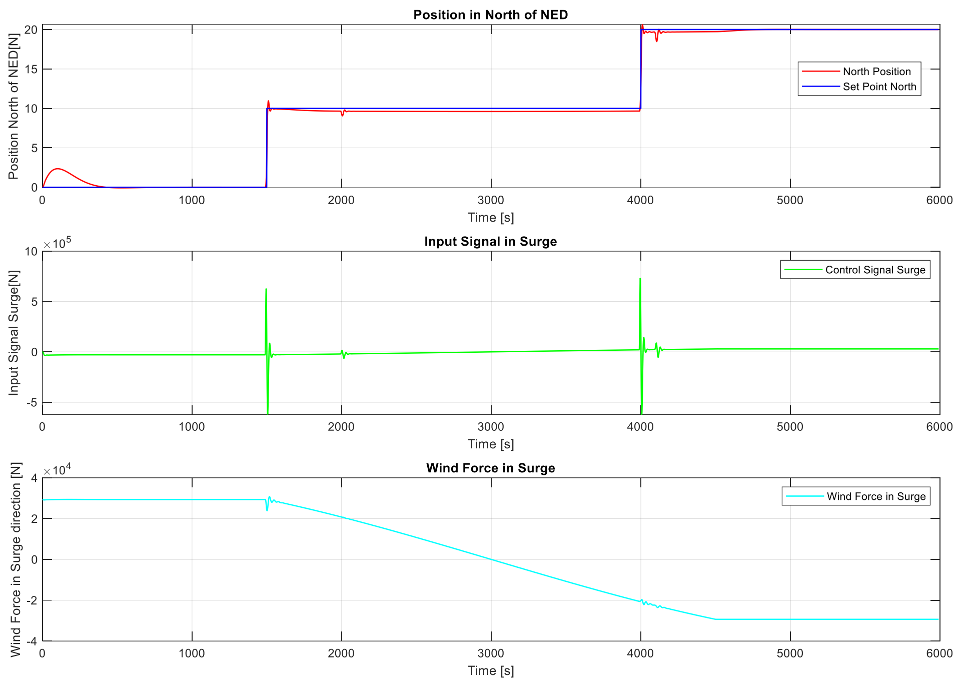
If the MPC problem is constrained, quadprog needs to be used, but in this case, the problem will be formulated as an unconstrained MPC.

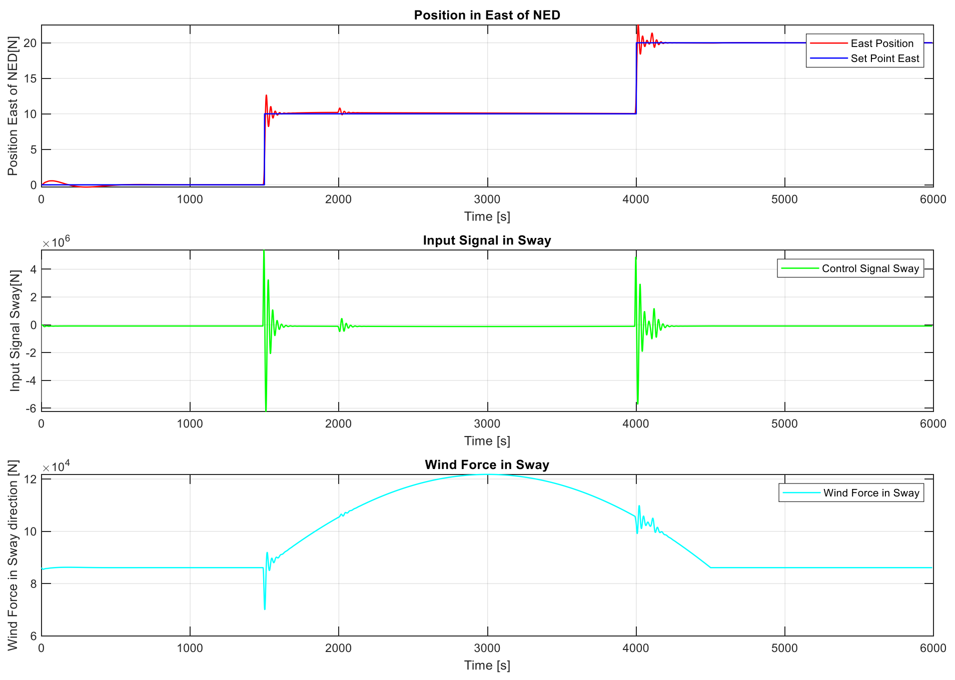
Finally, as with LQ Optimal control, the model is simulated with the new u and disturbance values and the states are estimated.

```
%Updating the model, estimation of states and update of xt which is
%used to calculate pL
[x, NED_Position] = model.UpdateState(u, W_c(:,i), W_v(i), gamma(i), dt);
x_est = A*x_est + B * u + K*(y - D*x_est);
xt = [x_est-x_old;y_old];
```

6.1.1 Unconstrained vs Constrained MPC

Figure 6-2 shows the behavior of the MPC controller with a windspeed of 15 m/s. The same speed used in the LQ Integral controller. The controller manages to follow the setpoints almost exactly.


Position in North of NED
Position North of NED[N]
North Position
Set Point North
Time [s]
Input Signal in Surge
Input Signal Surge[N]
$\times 10^5$
Control Signal Surge
Time [s]
Wind Force in Surge
Wind Force in Surge direction [N]
$\times 10^4$
Wind Force in Surge
Time [s]


Position in East of NED
Position East of NED[N]
East Position
Set Point East
Time [s]
Input Signal in Sway
Input Signal Sway[N]
$\times 10^6$
Control Signal Sway
Time [s]
Wind Force in Sway
Wind Force in Sway direction [N]
$\times 10^4$
Wind Force in Sway
Time [s]

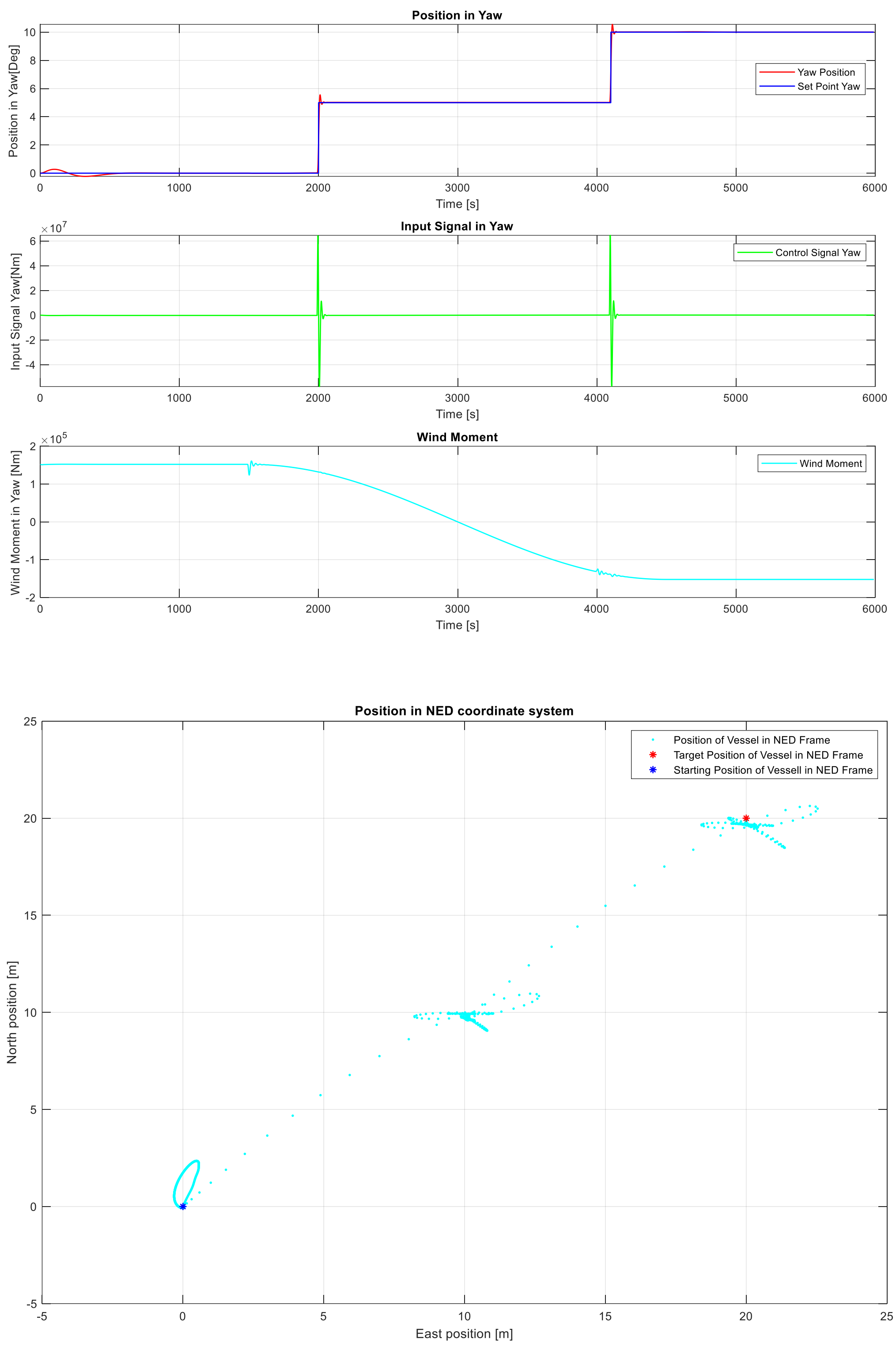
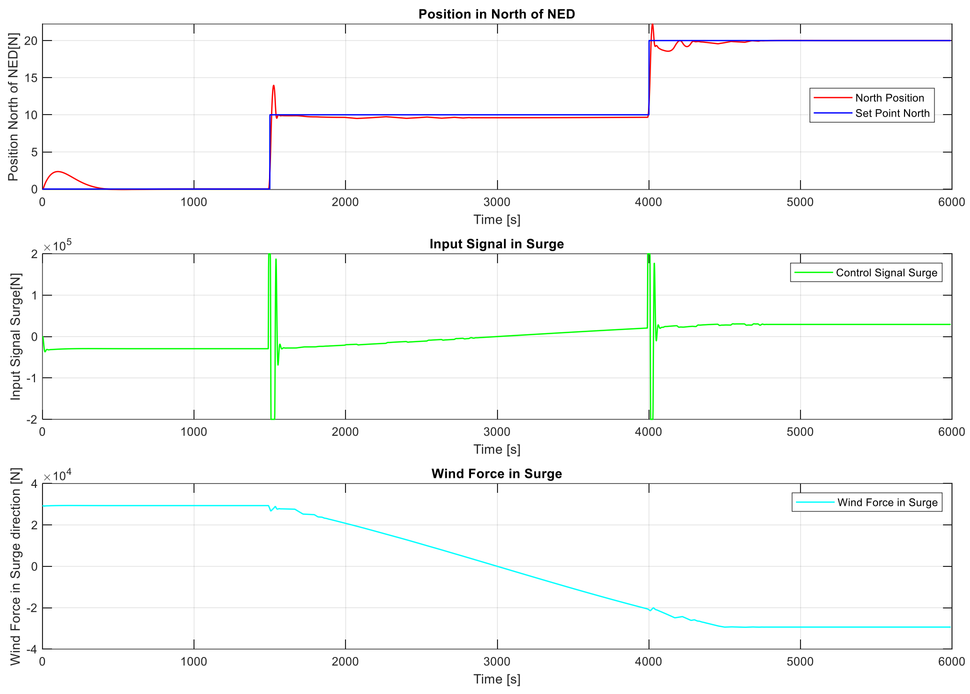


Figure 6-2 Performance of the System with Unconstrained MPC

However, as the value of u is not constrained, it reaches high values, especially for the sway and the yaw states. This could be limited by using the upper and lower limits in the quadprog function. The limits can be stablished as a difference between u and the threshold value, which for surge will be 2e5, for sway 3e5, and for yaw 1e6:

```
%Defining lowe and upper bounds for u
LB = [-2e5-u(1); -3e5-u(2);-1e6-u(3)];
UB = [2e5-u(1); 3e5-u(2);1e6-u(3)];
du = quadprog(H,f,[],[],[],[],LB,UB);
```



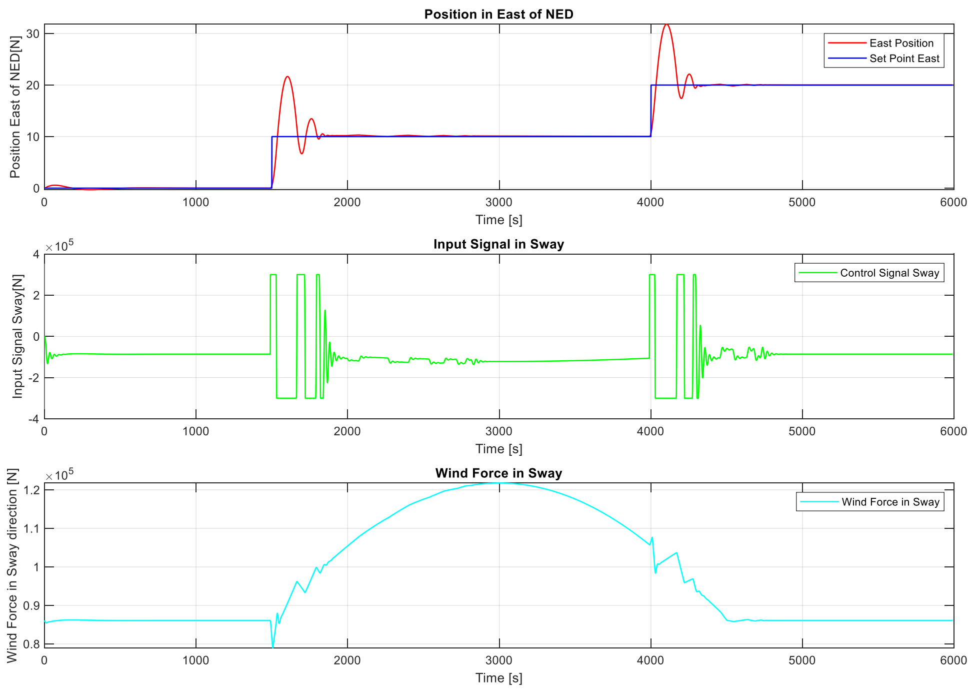
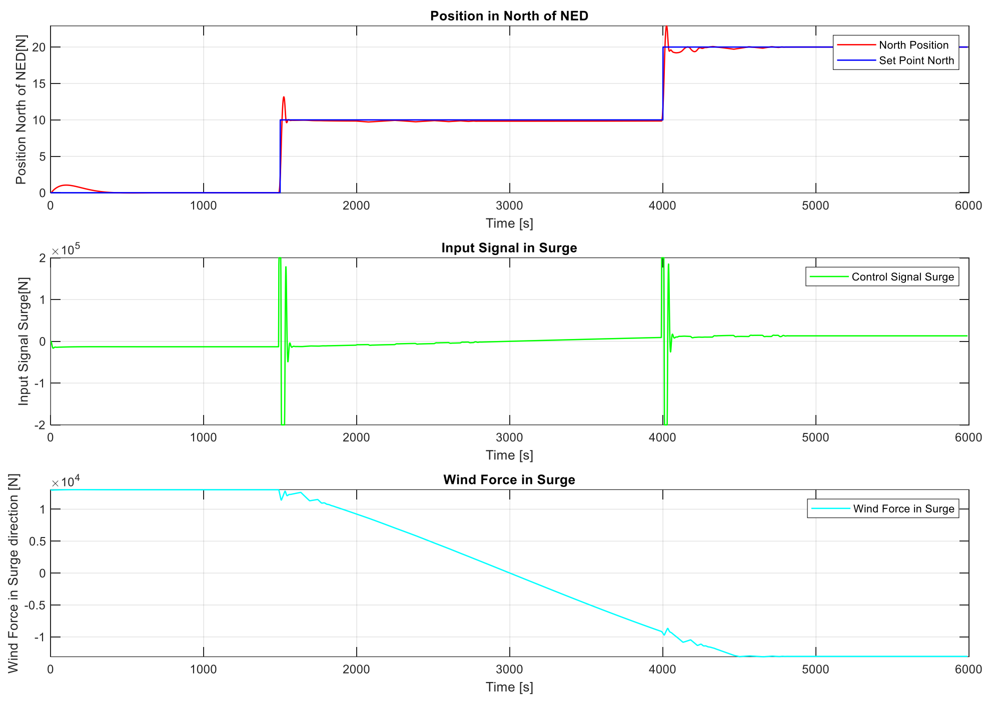
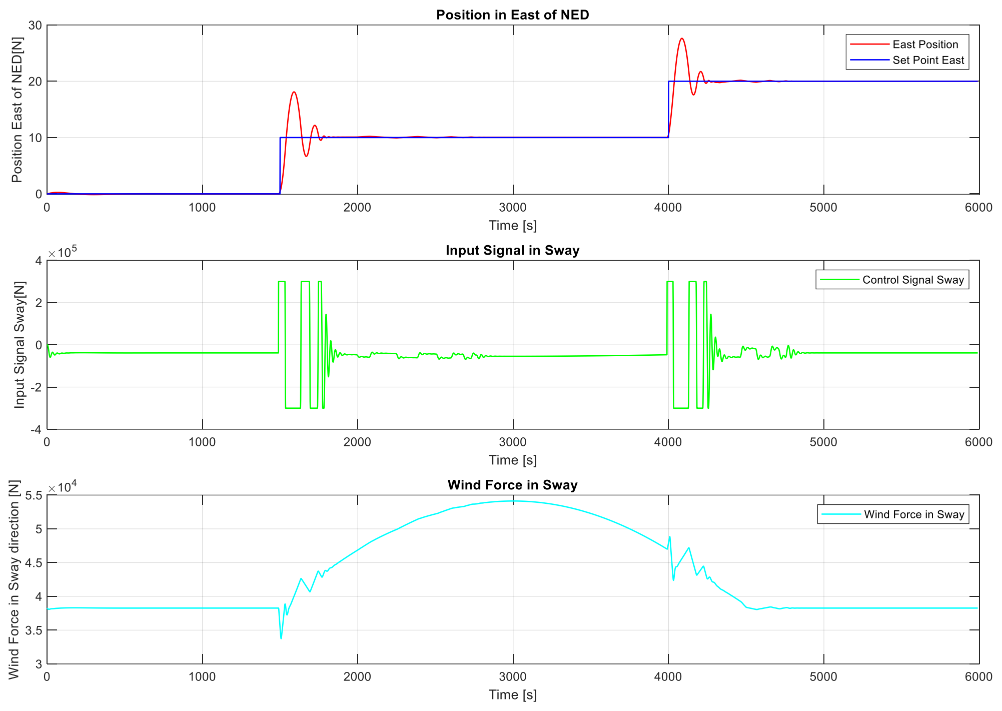


Figure 6-3 Performance of Integral MPC Controller with Clipping of u Values

Using the same q and p weighting values, the system will behave as shown in Figure 6-3. Although the controller shows oscillations after reaching the setpoints, it manages to stabilize itself, performing especially well for the surge state (North Position in NED). The controller for the east position presents a considerable overshot however, with a highest value of 30 m, while the setpoint was 20m. The yaw controller shows an overshot of 3 degrees and posterior oscillations.

Nevertheless, 15 m/s is a considerable wind speed. The average wind speeds in the northern sea fluctuate around 10 m/s, so it is a good idea to see how the controller behaves with that wind speed.


Position in North of NED
Position North of NED[N]
North Position
Set Point North
Time [s]
Input Signal in Surge
Input Signal Surge[N]
$\times 10^5$
Control Signal Surge
Time [s]
Wind Force in Surge
Wind Force in Surge direction [N]
$\times 10^4$
Wind Force in Surge
Time [s]


Position in East of NED
Position East of NED[N]
East Position
Set Point East
Time [s]
Input Signal in Sway
Input Signal Sway[N]
$\times 10^5$
Control Signal Sway
Time [s]
Wind Force in Sway
Wind Force in Sway direction [N]
$\times 10^4$
Wind Force in Sway
Time [s]

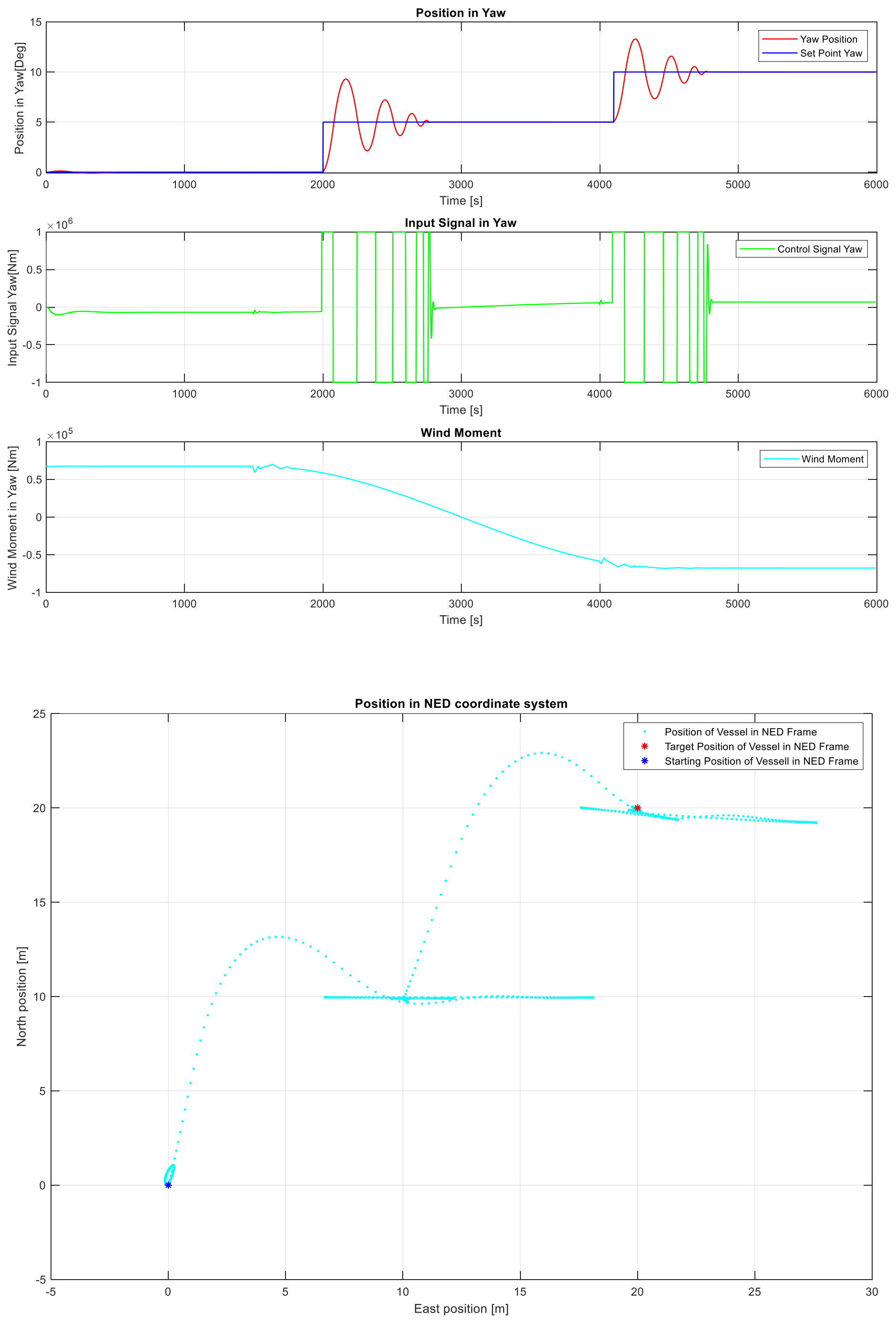
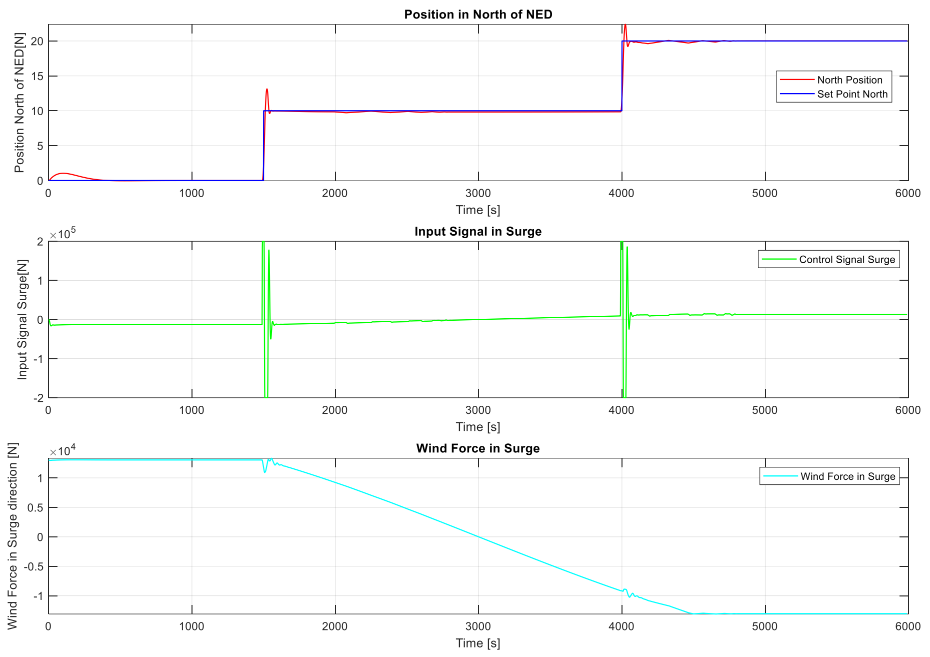
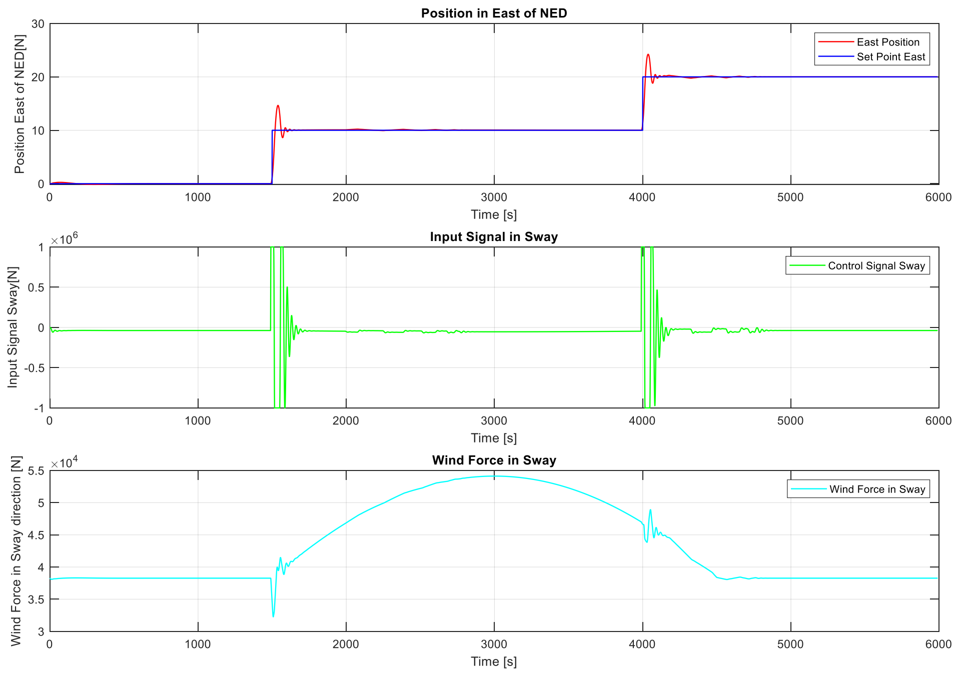


Figure 6-4 Behavior of Integral MPC Controller with 10 m/s Wind Speed

In Figure 6-4, the behavior of the sway controller improves, however it maintains the oscillations. In a real case scenario, this behavior could be accepted, nevertheless, it could be improved if the limit for the sway controller is relaxed:

```
%Defining lower and upper bounds for u
LB = [-2e5-u(1); -1e6-u(2);-1e6-u(3)];
UB = [2e5-u(1); 1e6-u(2);1e6-u(3)];
```


Position in North of NED
Position North of NED[N]
North Position
Set Point North
Time [s]
Input Signal in Surge
Input Signal Surge[N]
$\times 10^5$
Control Signal Surge
Time [s]
Wind Force in Surge
Wind Force in Surge direction [N]
$\times 10^4$
Wind Force in Surge
Time [s]


Position in East of NED
Position East of NED[N]
East Position
Set Point East
Time [s]
Input Signal in Sway
Input Signal Sway[N]
$\times 10^6$
Control Signal Sway
Time [s]
Wind Force in Sway
Wind Force in Sway direction [N]
$\times 10^4$
Wind Force in Sway
Time [s]

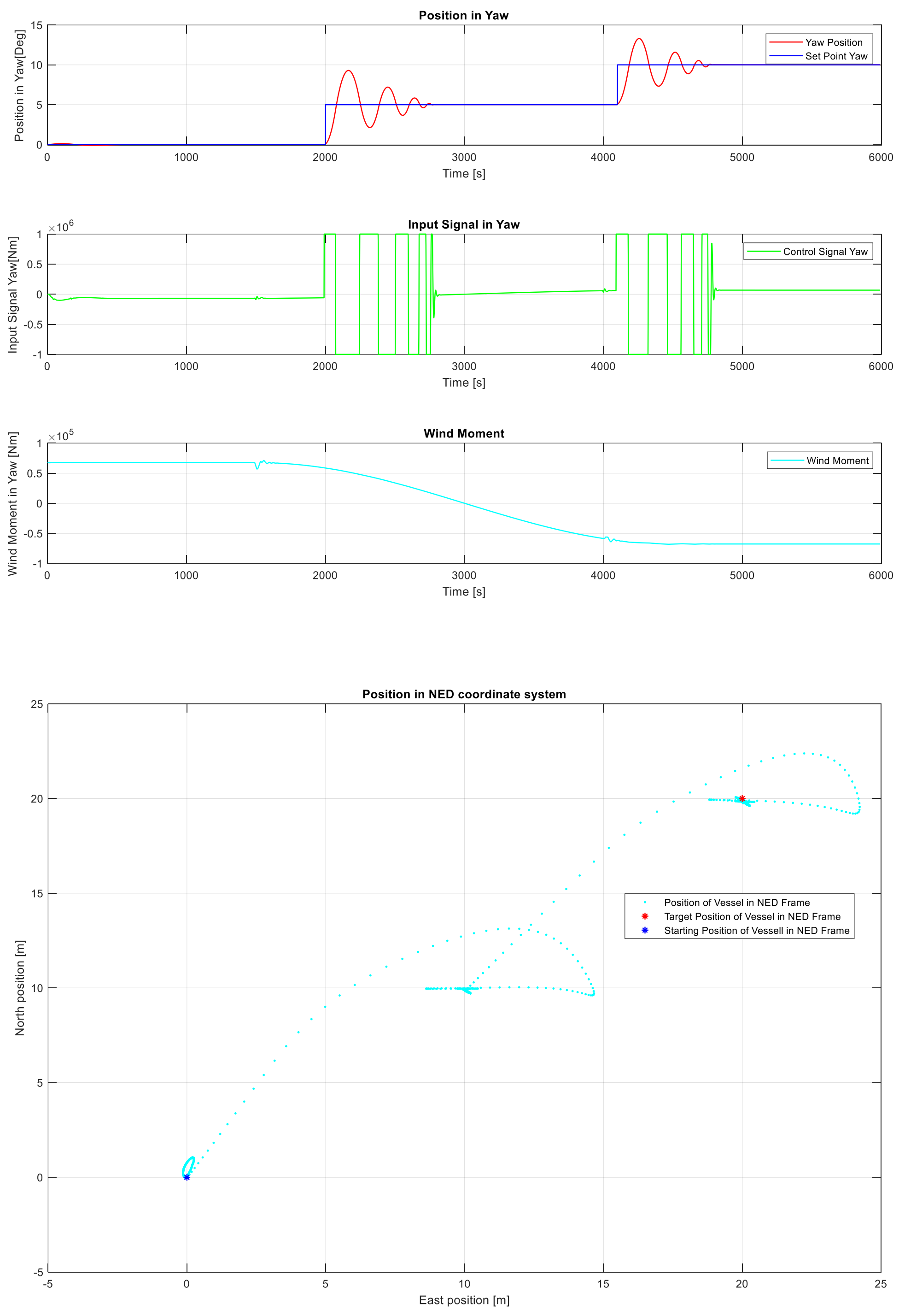


Figure 6-5 Performance of the Integral MPC Controller with Relaxed u Limits for Sway

Figure 6-5 shows that the sway controller behaves much better, reducing the overshot to a minimum and showing practically no oscillations. Every change in the input limits, however, comes with a cost since higher absolute control signal values will represent a higher energy expense in a real-life system.

7 Comparison of PID, LQ, and MPC Methods

The use of PID controllers, LQ Optimal control with integral action, and MPC with integral action have been discussed before. Although all the methods have proven being effective in positioning the vessel, it is necessary to perform a test under the same conditions for the three methods.

7.1 Experiment Design

As per [12], the maximum average hourly wind speed in the northern sea is approximately equal to 10 m/s. Hence, the wind conditions will be simulated with winds of this magnitude. In that same source, it is also stated that the most frequents wind directions are from north-west during the winter months and south-west during the summer months. That is why, to test the different control methods, the windspeed will be kept at 10 m/s and the direction of the wind will be maintained in north-west for the first half of the simulation and then changed to south-west for the second half. The setpoints will be the same for all the controller types: from 0 to 10m for north and east positions, then from 10m to 20m for north and east positions, and from 0 to 5 degrees, followed by 5 to 10 degrees for the yaw angle.

7.1.1 Benchmarks

The controllers will be compared in a qualitative and quantitative way. The settling time on the setpoints will be measured, as well as the overshot and the root mean square error for the simulation.

7.1.1.1 Settling Time

The time at which the controller stabilizes at the setpoint (if this is the case), will be measured on the plot after the simulation is run. Figure 7-1 shows an example of how the settling time will be measured for each of the controllers. There is a tag for the new setpoint and a tag for when the controller stabilizes in it.

There are 2 step changes in the setpoint. The settling time will be determined by measuring the point in time when the oscillations disappear, even though there is still some minimal offset that is gradually reduced.

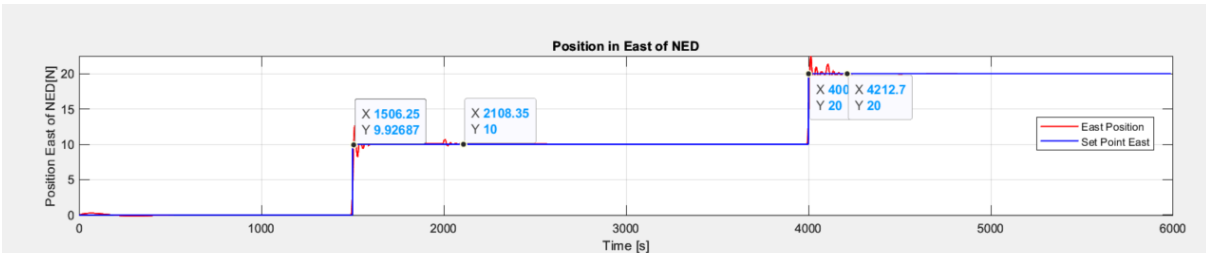


Figure 7-1 Example of Measuring of Time for Settling

7.1.1.2 Overshot and Oscillations

This will be a qualitative and quantitative analysis. If the controller shows overshot, it will be measured using the tags of the figure tool in MATLAB, as seen in Figure 7-1.

The oscillations will be compared qualitatively, identifying which controller shows more frequent and ample oscillations.

7.1.1.3 Root Mean Squared Error

The root mean squared error is a common method to measure the deviation of two arrays. In this case, the two arrays will be the set points in the NED coordinate frame and the position of the vessel in said frame. The RMSE is calculated using the formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(sp_i - y_i)^2} \tag{7.1}$$

Where:

sp_i: A given setpoint value in instant i in the Earth coordinate frame.

y_i: A given position measurement in instant i in the Earth coordinate frame.

N: Number of samples.

Equation 40 shows that the RMSE will be calculated using the absolute values of the differences between the setpoints and the outputs.

The setpoint vector and the position vector will be compared. It is worth noting that the setpoints are stablished in the earth's coordinate frame, so the position to be compared to, is in earth's coordinate frame as well.

7.1.2 Input Signals

The controllers will be tested first with unrestricted input signals and then establishing limits on them. The limits will be the same used in the last configuration of the MPC with integral action:

```
%Defining lowe and upper bounds for u
LB = [-2e5-u(1); -1e6-u(2);-1e6-u(3)];
UB = [2e5-u(1); 1e6-u(2);1e6-u(3)];
```

The Total Value index will be used to compare the amount of u used in each of the methods. According to [13], It is defined as:

$$TV = \sum_{k=1}^{\infty}|\Delta u_k| \tag{7.2}$$

And $\Delta u_k = u_k - u_{k-1}$. Which is the control signal rate of change.

7.2 Control with PID Controllers

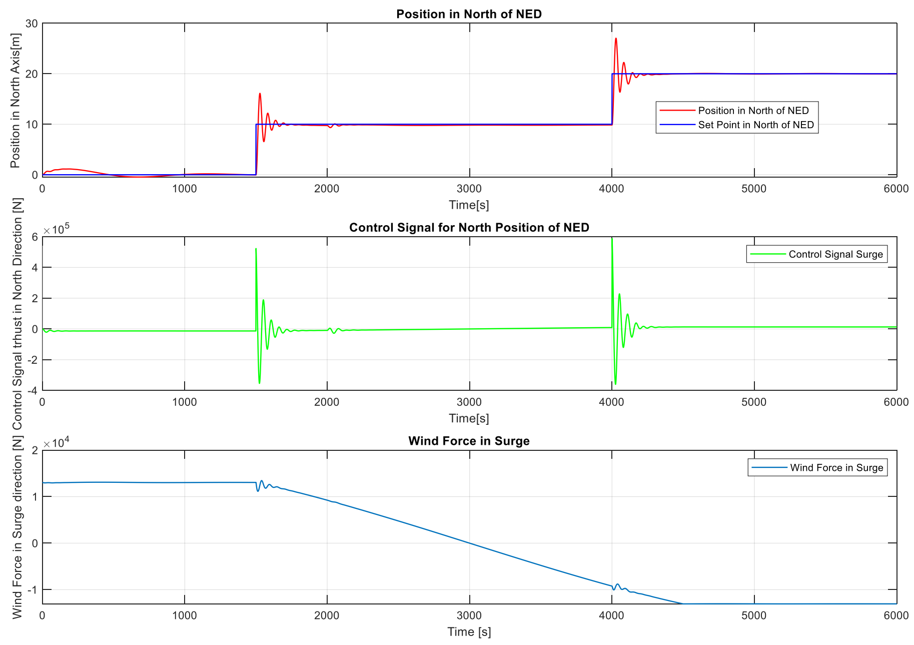
The controllers will have the last selected k_p, T_i, and T_d parameters. The values of these can be seen in Table 7-1.

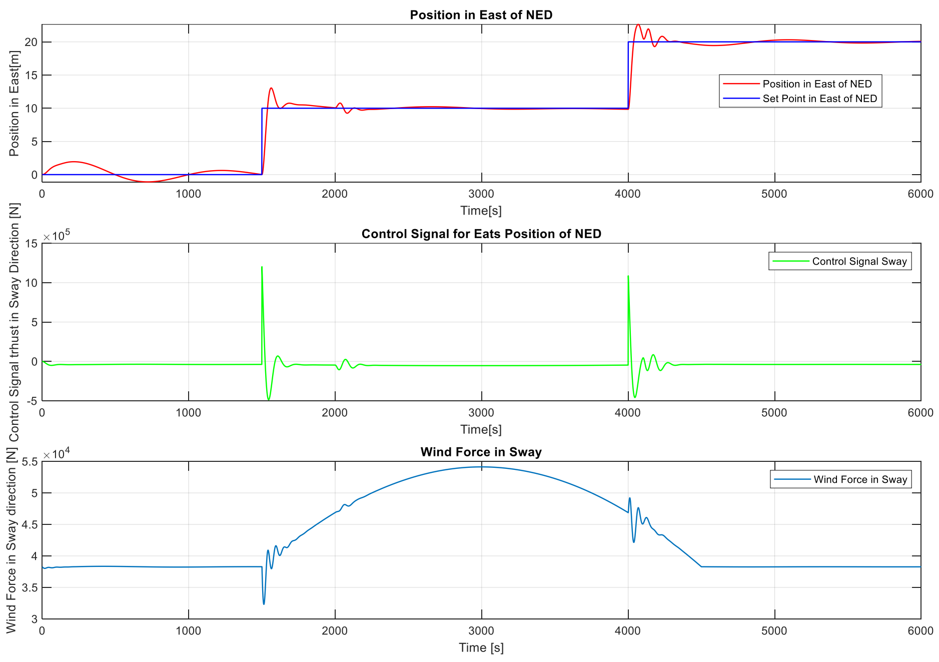
Table 7-1 PID Controllers Parameters for Comparison

Parameter	**Value**
Kp Surge PID	3500
Ti Surge PID	70
Td Surge PID	300
Kp Sway PID	5000
Ti Sway PID	50
Td Sway PID	500
Kp Yaw PID	800000
Ti Yaw PID	1500
Td Yaw PID	350

7.2.1 Performance Without Clipping the Control Signal

Figure 7-2 displays the behavior of the system with PID controllers. In general terms, the controllers reach the setpoints, with some oscillations in the surge controller, and some sluggishness in the yaw controller. The wind disturbance has an immediate effect on the position of the vessel, causing that the input signals always have a value different from zero.


Position in North of NED
Position in North Axis[m]
Position in North of NED
Set Point in North of NED
Time[s]
Control Signal for North Position of NED
Control Signal trhust in North Direction [N]
$\times 10^5$
Control Signal Surge
Time[s]
Wind Force in Surge
Wind Force in Surge direction [N]
$\times 10^4$
Wind Force in Surge
Time [s]


Position in East of NED
Position in East[m]
Position in East of NED
Set Point in East of NED
Time[s]
Control Signal for Eats Position of NED
Control Signal trhust in Sway Direction [N]
$\times 10^5$
Control Signal Sway
Time[s]
Wind Force in Sway
Wind Force in Sway direction [N]
$\times 10^4$
Wind Force in Sway
Time [s]

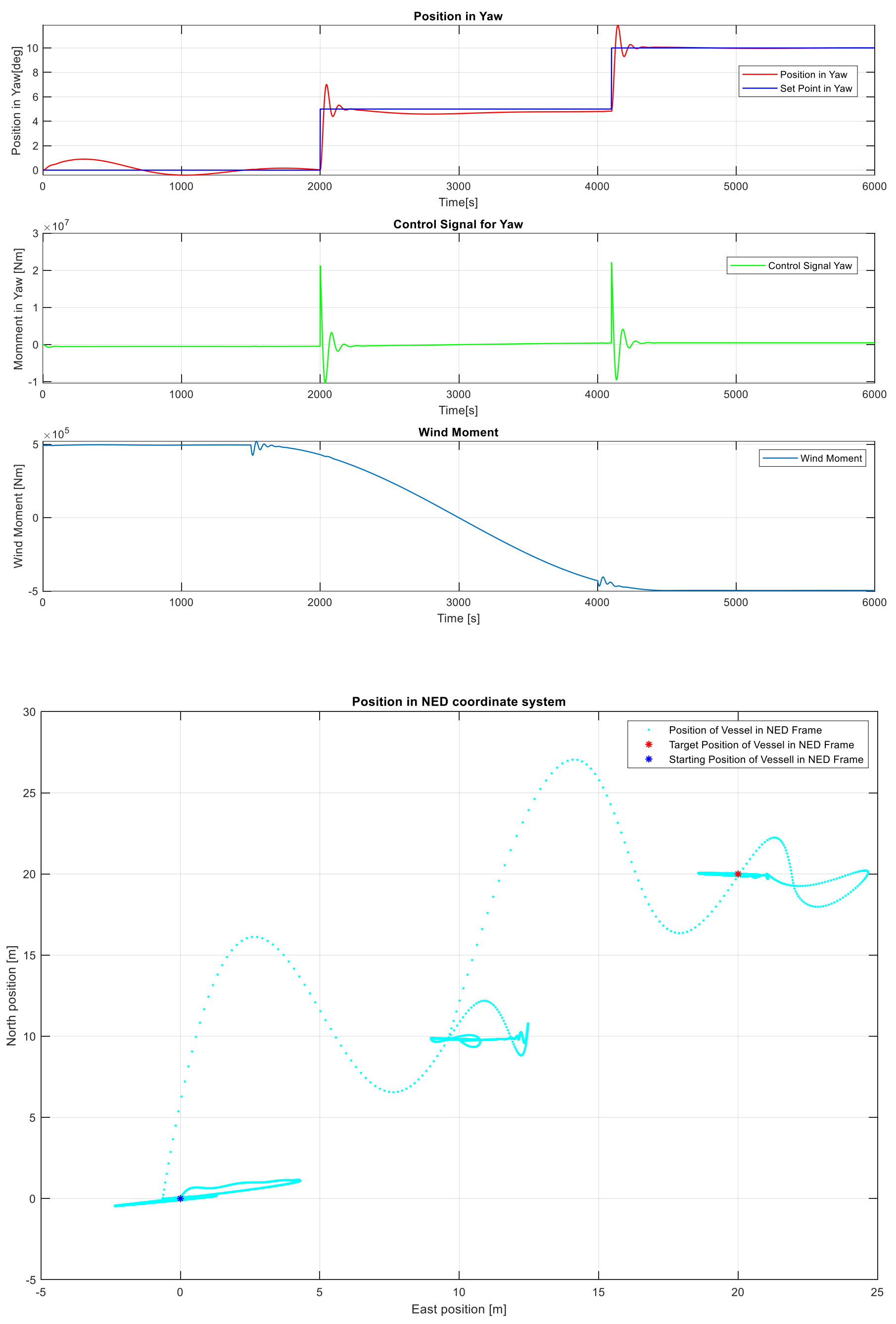


Figure 7-2 General Performance of PID Without Input Limits

7.2.1.1 Settling Time

The settling times for the controllers in each state and for each timestep are shown in Figure 7-3 and summarized in Table 7-2.

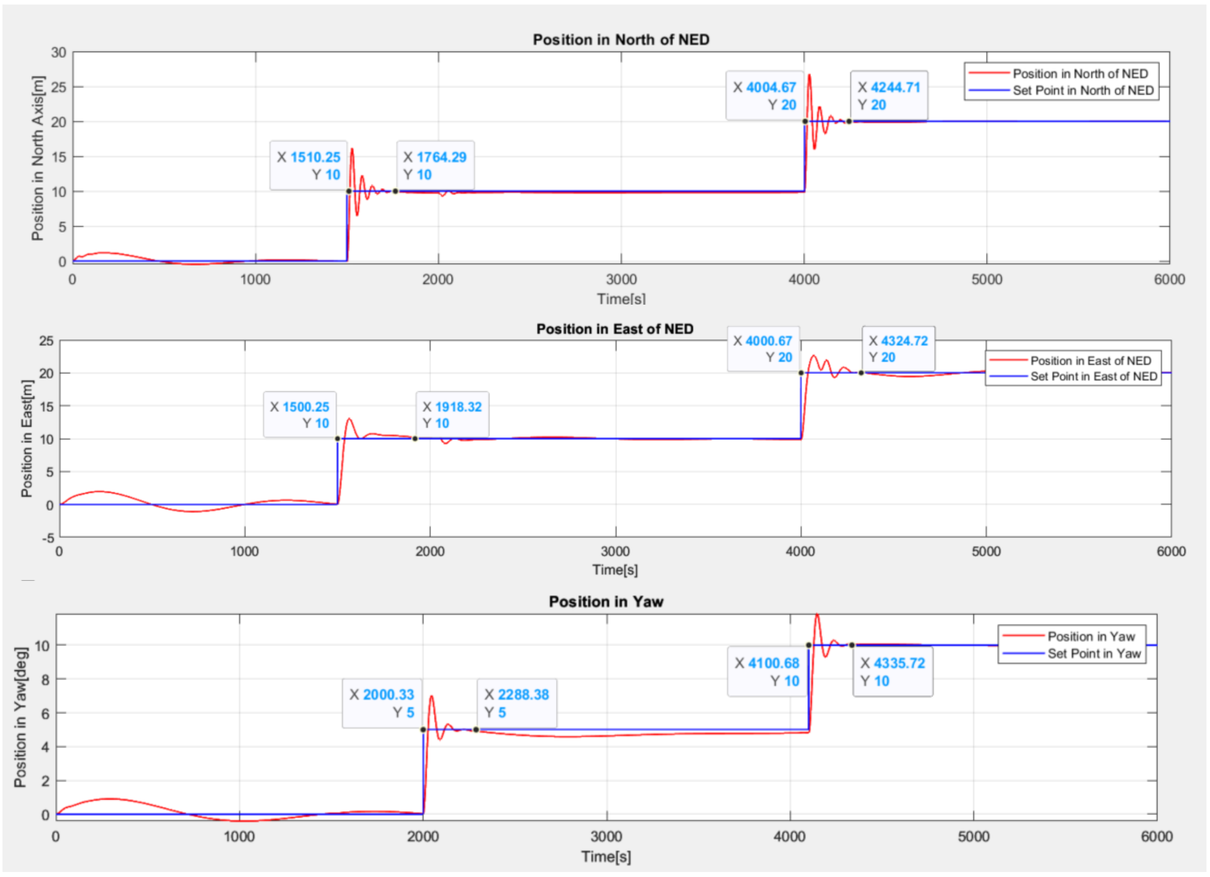


Figure 7-3 Settling Times for PID Controllers without Input Limits

Table 7-2 Settling Times for Unclipped PID Controllers

Controller and Setpoint	**Approximate Time [s]**
Surge Controller Step 1	273
Surge Controller Step 2	294
Sway Controller Step 1	418
Sway Controller Step 2	324
Yaw Controller Step 1	288
Yaw Controller Step 2	235

7.2.1.2 Overshot and Oscillations

The overshot measurements can be seen in Figure 7-4 and are summarized in Table 7-3.

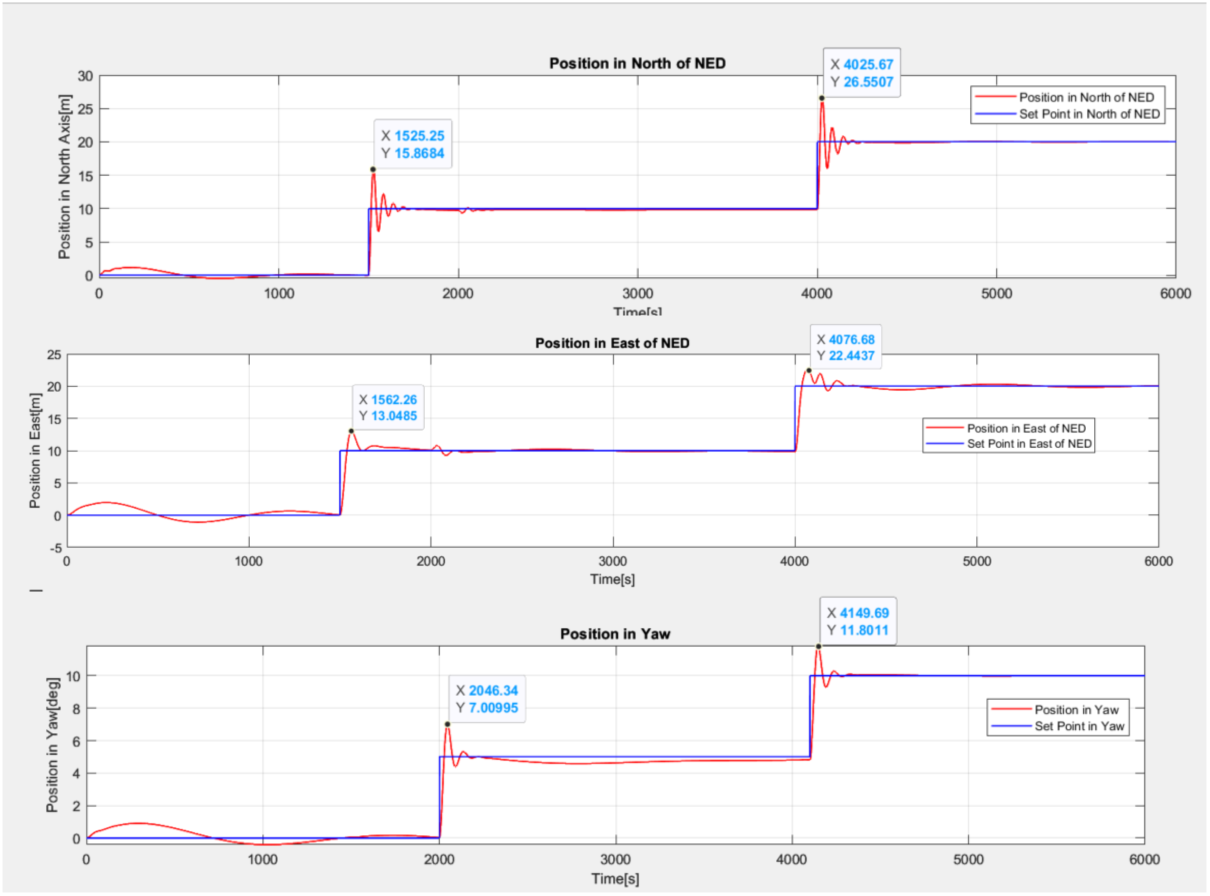


Figure 7-4 Overshot and Oscillations of PID Controllers without Input Limits

Table 7-3 Overshot and Oscillations of PID Controllers Without Input Limits

Controller and Setpoint	Overshot [m or Deg.]
Surge Controller Step 1	5.86 [m]
Surge Controller Step 2	6.5 [m]
Sway Controller Step 1	3.04 [m]
Sway Controller Step 2	2.44 [m]
Yaw Controller Step 1	2 [Deg.]
Yaw Controller Step 2	1.8 [Deg.]

Figure 7-4 shows that the surge controller has the most oscillations, which decrease in amplitude gradually around the setpoint. This behavior is not ideal, but given the model's sensitivity to wind, they would not affect the functionality of the vessel, especially considering that the controller stabilizes the vessel around the setpoints.

7.2.1.3 RMSE

The values for each of the axis of the NED frame are shown in Table 7-4:

Table 7-4 RMSE for PID Controllers without Limiting Input Values

Axis	RMSE
North Position [m]	0.7923
East Position [m]	0.9551
Yaw Angle [Deg.]	0.4744

7.2.1.4 Maximum and Average U Values

The behavior of the control signals is shown in Figure 7-5, whereas the maximum and average values are shown in Table 7-5.

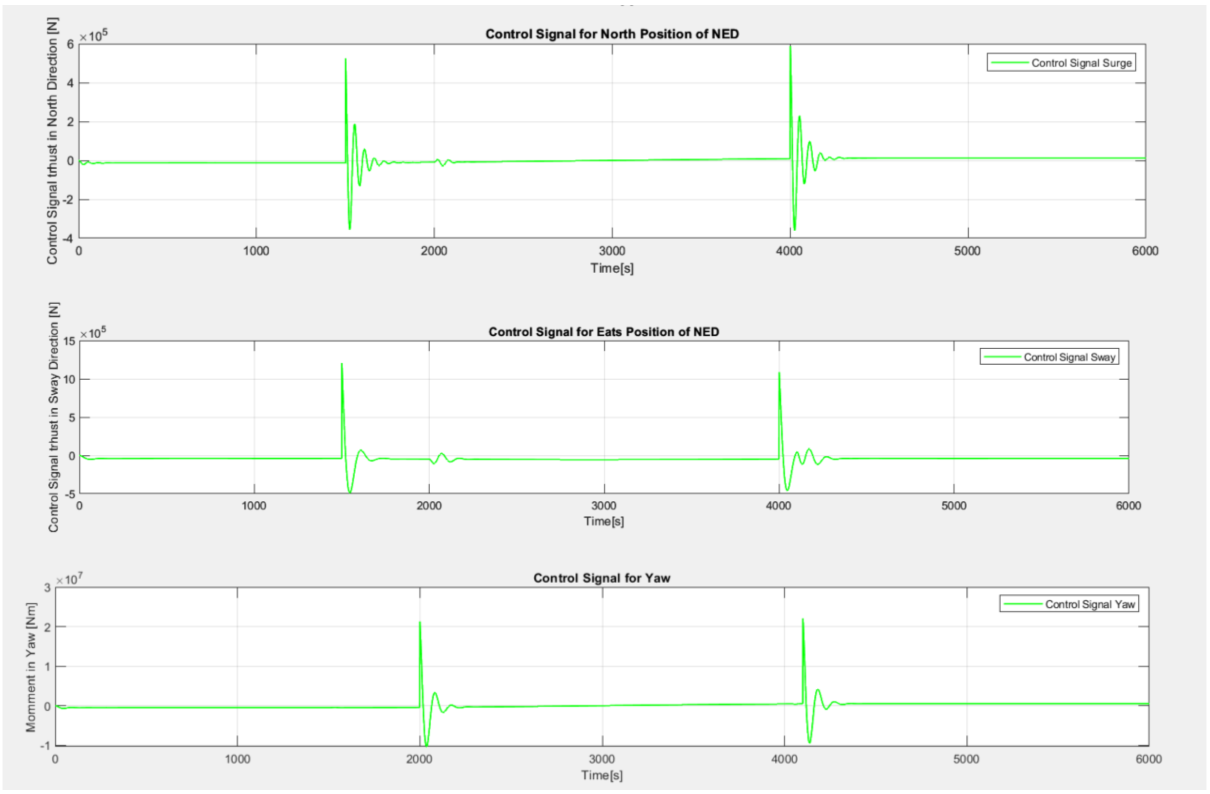


Figure 7-5 U Values for PID Controllers without Limited Control Signal

Table 7-5 Maximum and Average U Values for PID Controllers without Limited Input Signal

State	U Value
Max. Surge [N]	5.88E+05
Mean Surge [N]	6.00E+00
Max. Sway [N]	1.20E+06
Mean Sway [N]	-4.35E+04
Max. Yaw [Nm]	2.21E+07
Mean Yaw [Nm]	2.11E+02

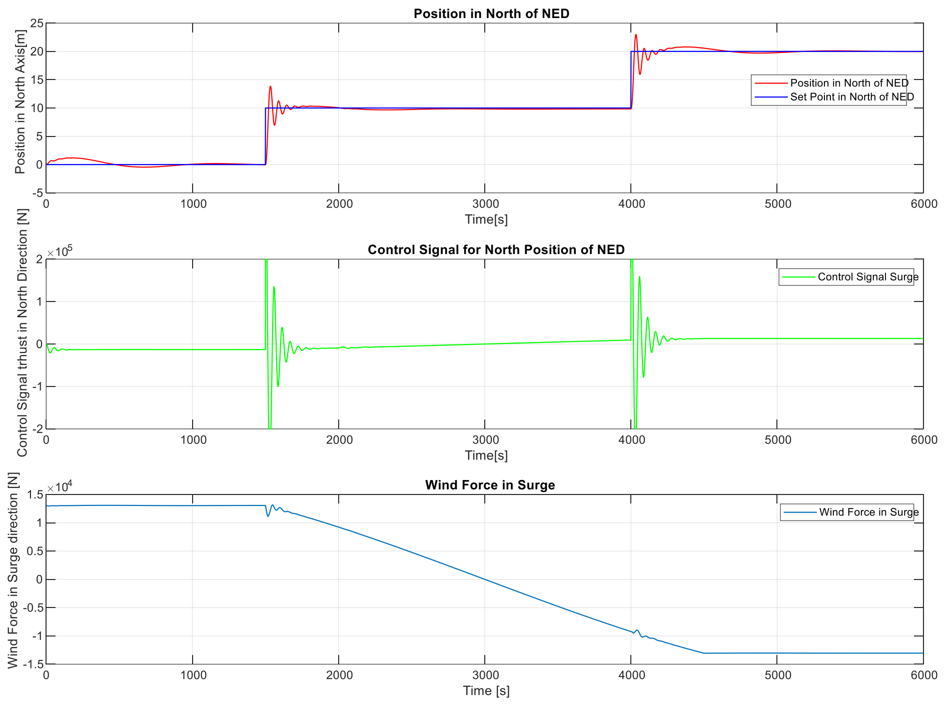
The Total Values are found in Table 7-6.

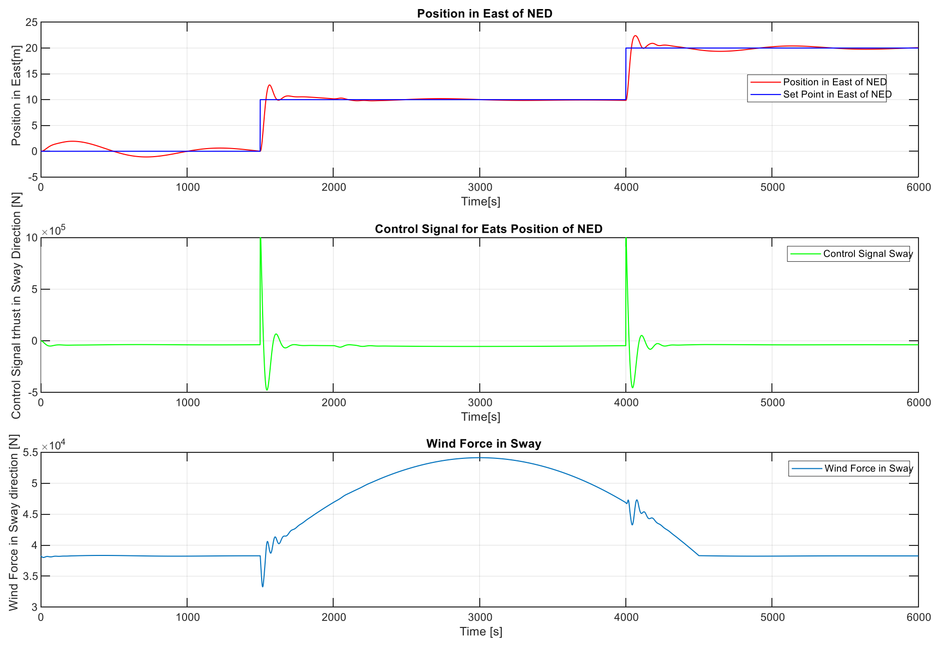
Table 7-6 Total Values for PIDs without Limited Input Signals

Controller	**Total Value (TV)**
Surge	5.92E+06
Sway	8.02E+06
Yaw	1.52E+08

7.2.2 Performance of PID Controller Limiting the Input Signal

Figure 7-6 shows the behavior of the PID controllers after limiting the input signals, the oscillations are increased in the surge controller and the behavior of the yaw controller deteriorates. Although it reaches the setpoints, its response is more sluggish.


Position in North of NED
Position in North Axis[m]
Time[s]
Position in North of NED
Set Point in North of NED
Control Signal for North Position of NED
Control Signal trhust in North Direction [N]
×10^5
Control Signal Surge
Wind Force in Surge
Wind Force in Surge direction [N]
×10^4
Time [s]
Wind Force in Surge


Position in East of NED
Position in East[m]
Time[s]
Position in East of NED
Set Point in East of NED
Control Signal for Eats Position of NED
Control Signal trhust in Sway Direction [N]
×10^5
Control Signal Sway
Wind Force in Sway
Wind Force in Sway direction [N]
×10^4
Time [s]
Wind Force in Sway

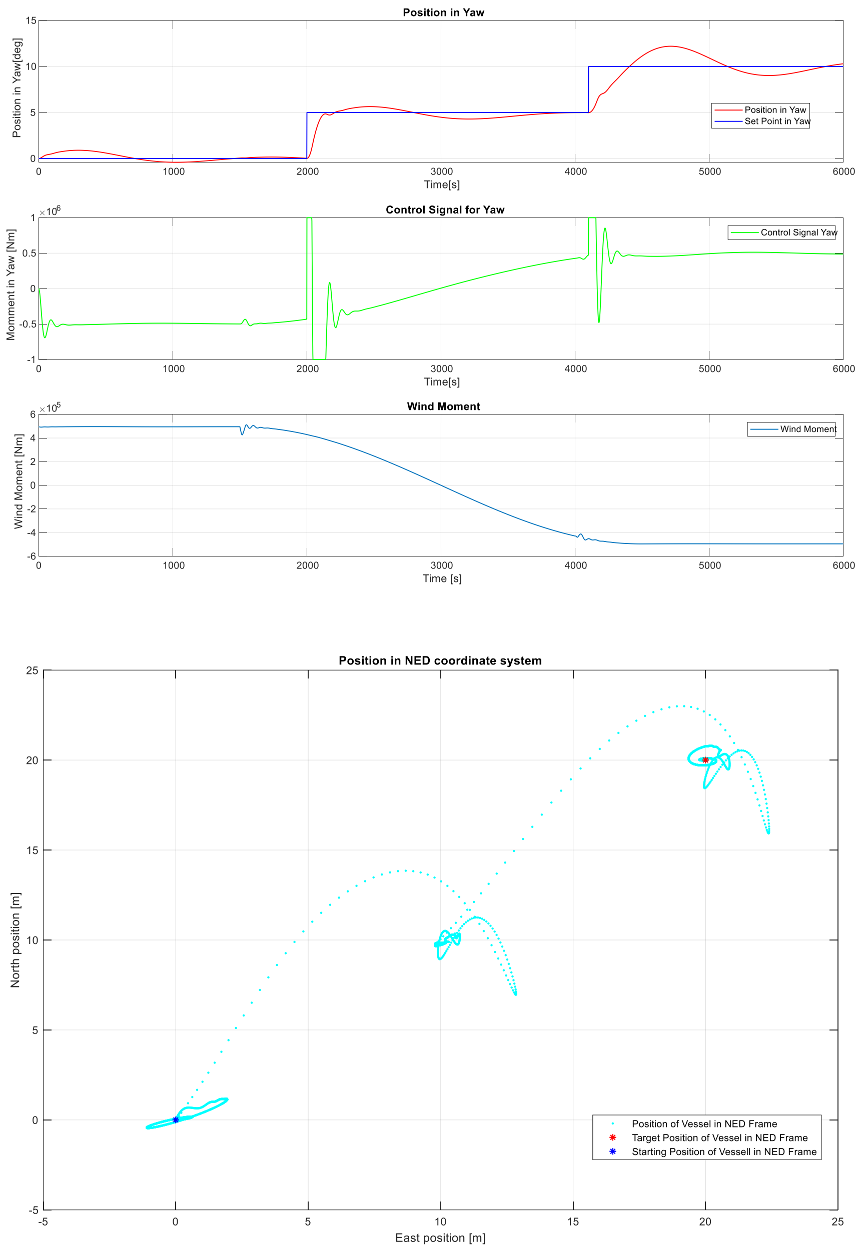


Figure 7-6 Performance of PID Controllers with Limited Input Signals

7.2.2.1 Settling Time

The settling times for the controllers in each state and for each timestep are shown in Figure 7-7. The settling times deteriorate considerably, particularly for the yaw controller.

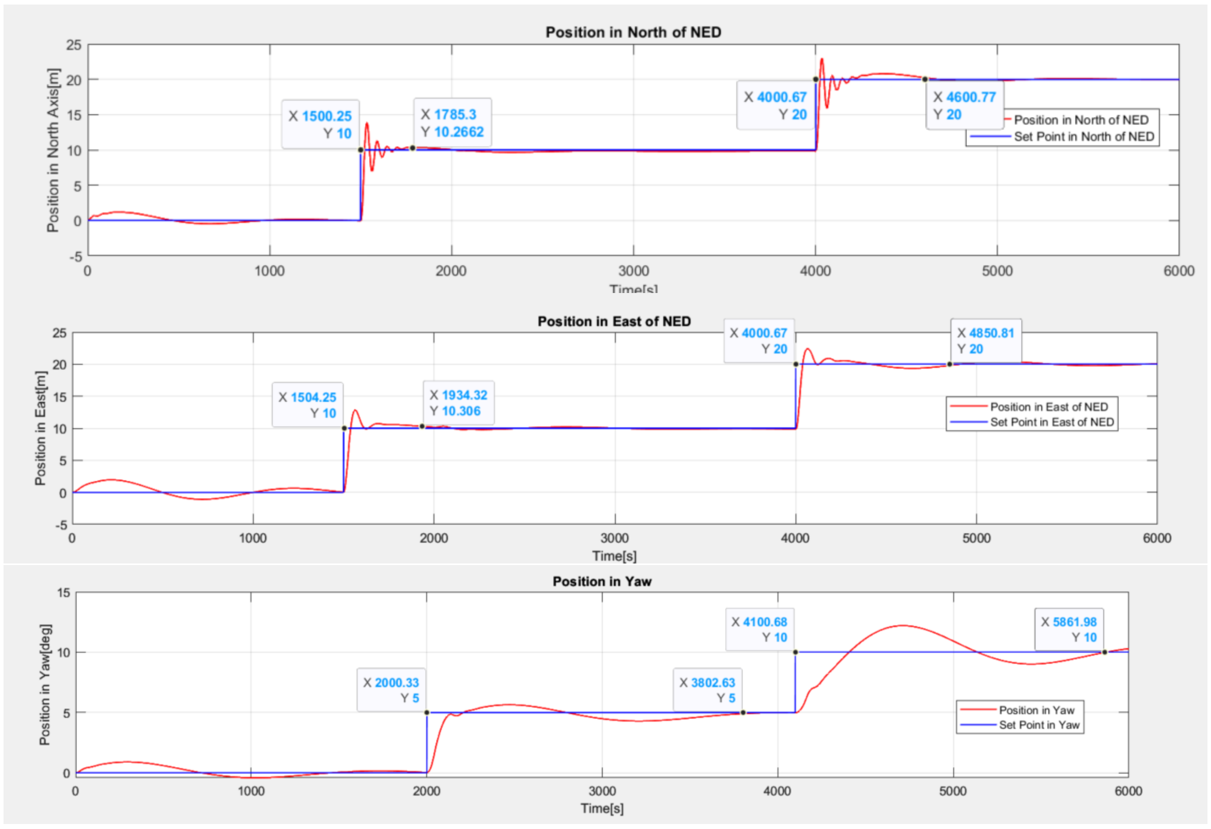


Figure 7-7 Settling Times for PID Controllers with Input Limits

Table 7-7 Settling Times for PID Controllers with Input Limits

Controller and Setpoint	**Approximate Time [s]**
Surge Controller Step 1	283
Surge Controller Step 2	600
Sway Controller Step 1	430
Sway Controller Step 2	850
Yaw Controller Step 1	1802
Yaw Controller Step 2	1761

7.2.2.2 Overshot and Oscillations

The values of the overshot for each controller are shown in Figure 7-8 and summarized in Table 7-8. The oscillations are like when the input signals are not limited. However, the overshot decreases for all the controllers. As the input signal is limited, the reactiveness of the controllers diminishes, so there will be less overshot as the response is slower.

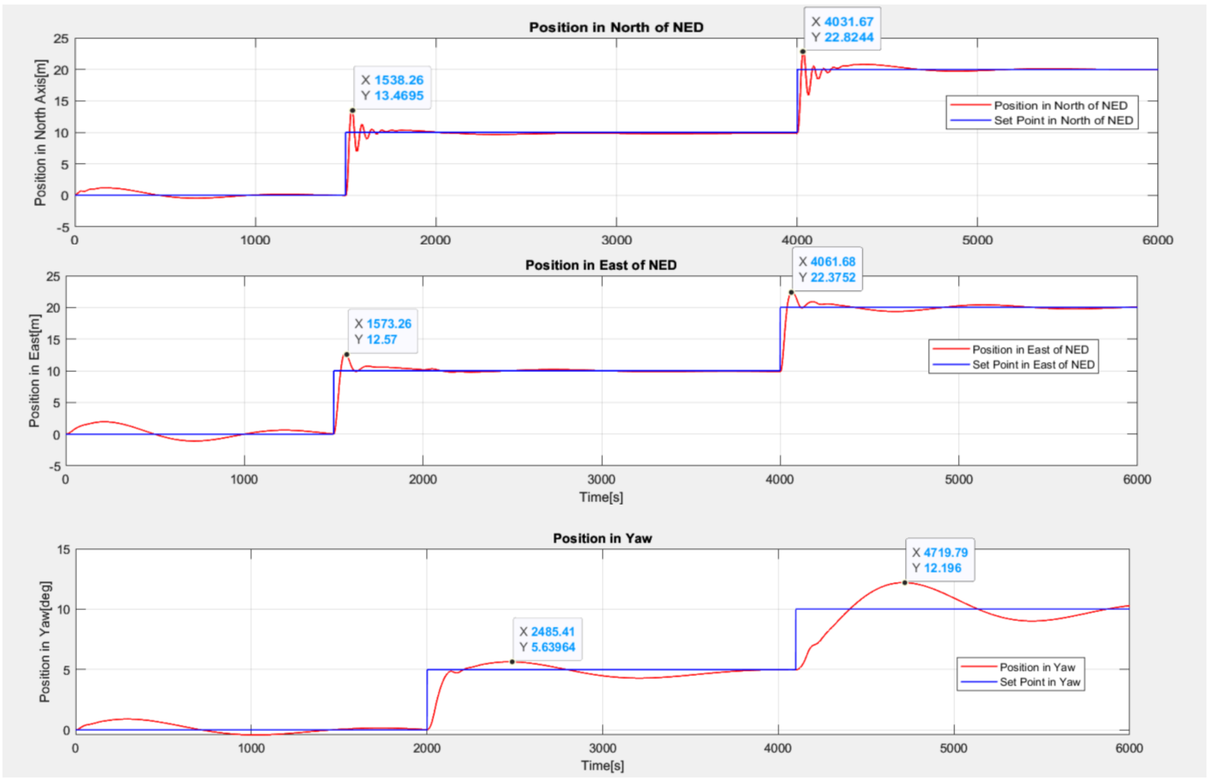


Figure 7-8 Overshot and Oscillations of PID Controllers with Input Limits

Table 7-8 Overshot and Oscillations of PID Controllers with Input Limits

Controller and Setpoint	Overshot [m or Deg.]
Surge Controller Step 1	3.46 [m]
Surge Controller Step 2	2.8 [m]
Sway Controller Step 1	2.6 [m]
Sway Controller Step 2	2.3 [m]
Yaw Controller Step 1	0.6 [Deg.]
Yaw Controller Step 2	2.1 [Deg.]

7.2.2.3 RMSE

The values for each of the axis of the NED frame are shown in Table 7-9.

Table 7-9 RMSE for PID Controllers Limiting the Input Signals

State	RMSE
North Position [m]	0.7965
East Position [m]	0.9608
Yaw Angle [Deg.]	1.0256

7.2.2.4 Maximum and Average U Values

The behavior of the control signals is shown in Figure 7-9, whereas the maximum and average values are shown in Table 7-10. It is no surprise that the maximum absolute values are the same as the limits, as the control inputs cannot exceed these. The average values for the yaw are doubled, most likely because the controller compensates the effect of the limit by being activated for a longer time.

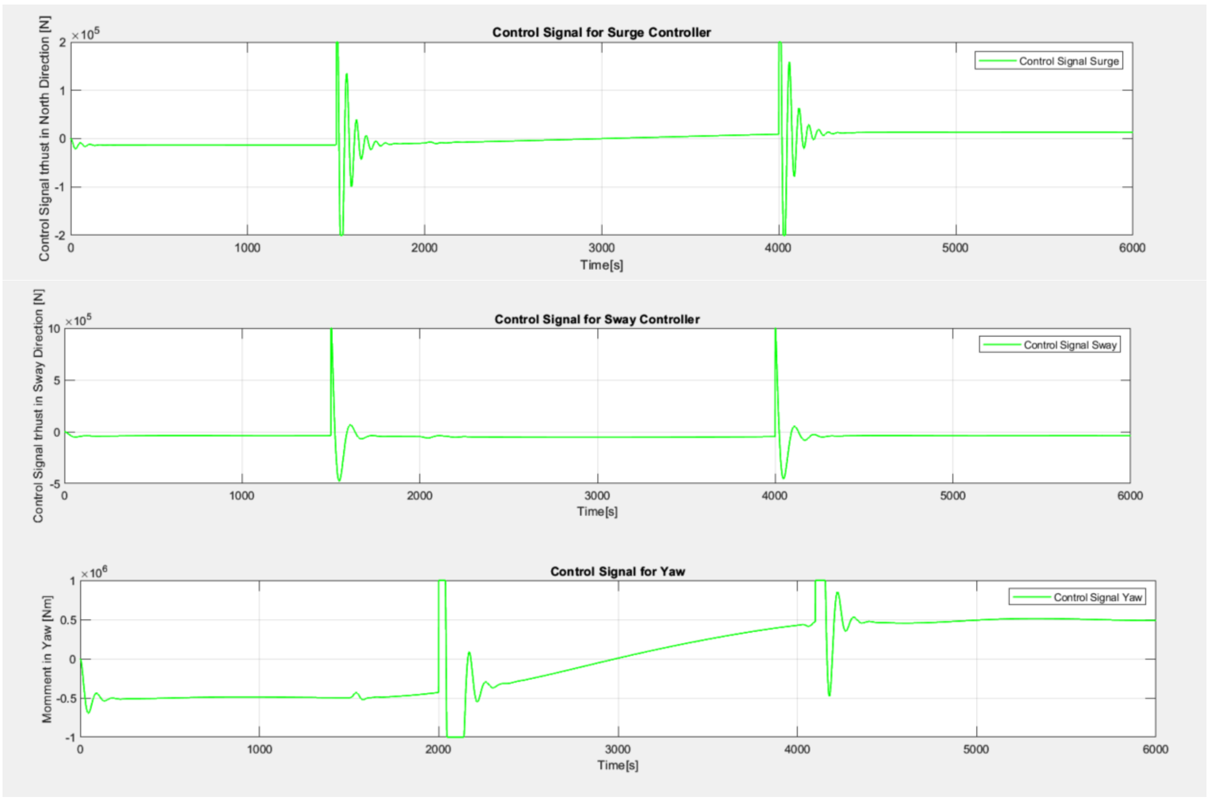


Figure 7-9 U Values for PID Controllers with Limited Control Signal

Table 7-10 Maximum and Average U Values for PID Controllers with Limited Input Signal

State	U Value
Max. Surge [N]	2.00E+05
Mean Surge [N]	6.00E+00
Max. Sway [N]	1.00E+06
Mean Sway [N]	-4.35E+04
Max. Yaw [Nm]	1.00E+06
Mean Yaw [Nm]	4.44E+02

The Total values for the control signals of the three controllers can be seen in

Table 7-11.

Table 7-11 Total Values for PIDs with Limits in the Input Signals

Controller	Total Value (TV)
Surge	8.97E+06
Sway	7.21E+06
Yaw	8.80E+08

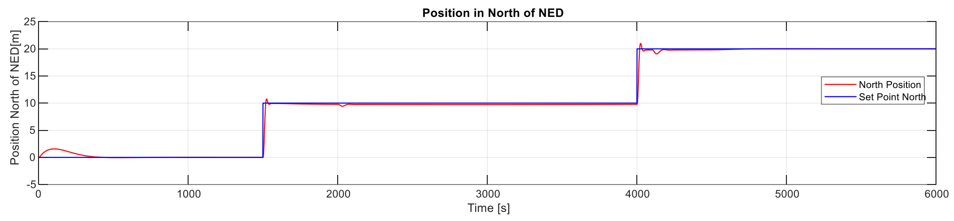
7.3 Performance with LQ-Optimal Control with Integral Action

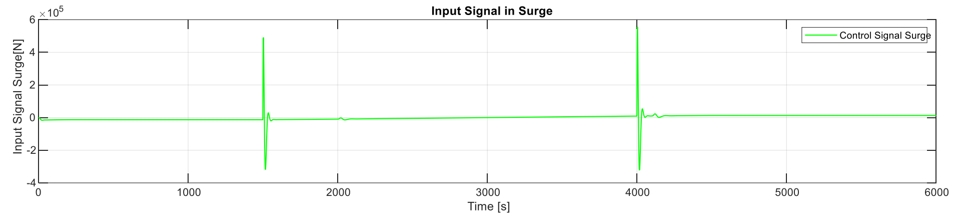
In this section, the results of the experiment with the LQ Optimal Controller with Integral action will be shown. In this case, as the windspeed is reduced to 10 m/s, the difference between q and p can be relaxed, so these values are used:

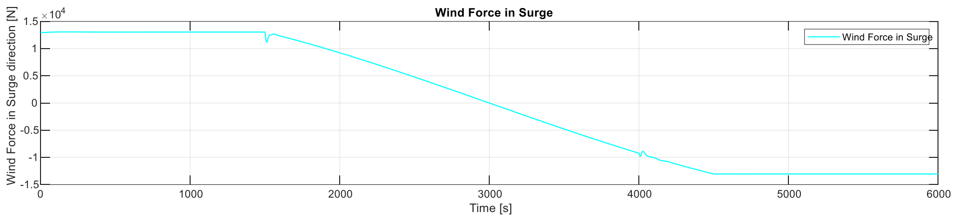
```
% Controller weights
%Weighting parameters, p is for delta u, q is for y
% p=1e-6; q=10000;

q = [1e4 0 0; 0 1e4 0; 0 0 1e6];
p = [1e-5 0 0; 0 1e-5 0; 0 0 1e-9];
```

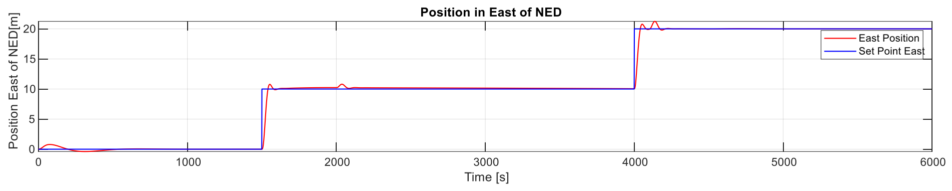
7.3.1 Performance Without Clipping the Control Signal

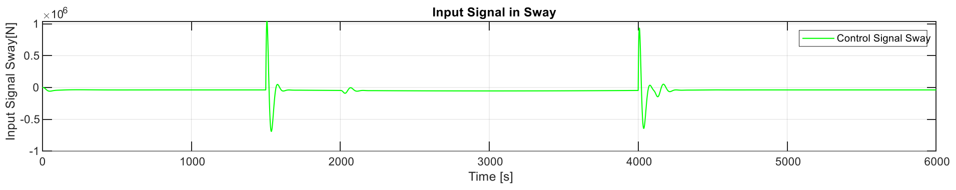
The results are shown in Figure 7-10. At first look, this controller is more aggressive than the PID controllers. The setpoints are followed almost exactly as specified, there are no oscillations, and the overshot is minimal or nonexistent.

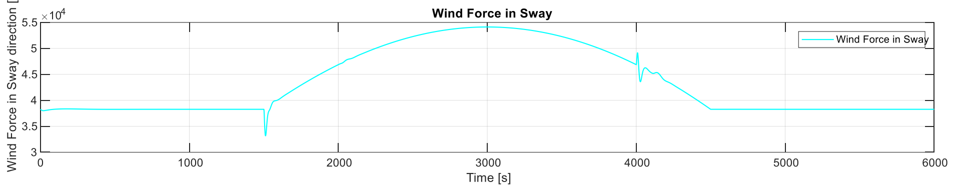

Position in North of NED
Position North of NED[m]
25
20
15
10
5
0
-5
0
1000
2000
3000
4000
5000
6000
Time [s]
North Position
Set Point North


Input Signal in Surge
$\times 10^5$
Input Signal Surge[N]
6
4
2
0
-2
-4
0
1000
2000
3000
4000
5000
6000
Time [s]
Control Signal Surge


Wind Force in Surge
$\times 10^4$
Wind Force in Surge direction [N]
1.5
1
0.5
0
-0.5
-1
-1.5
0
1000
2000
3000
4000
5000
6000
Time [s]
Wind Force in Surge


Position in East of NED
Position East of NED[m]
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000
Time [s]
East Position
Set Point East


Input Signal in Sway
$\times 10^6$
Input Signal Sway[N]
1
0.5
0
-0.5
-1
0
1000
2000
3000
4000
5000
6000
Time [s]
Control Signal Sway


Wind Force in Sway
$\times 10^4$
Wind Force in Sway direction [N]
5.5
5
4.5
4
3.5
3
0
1000
2000
3000
4000
5000
6000
Time [s]
Wind Force in Sway

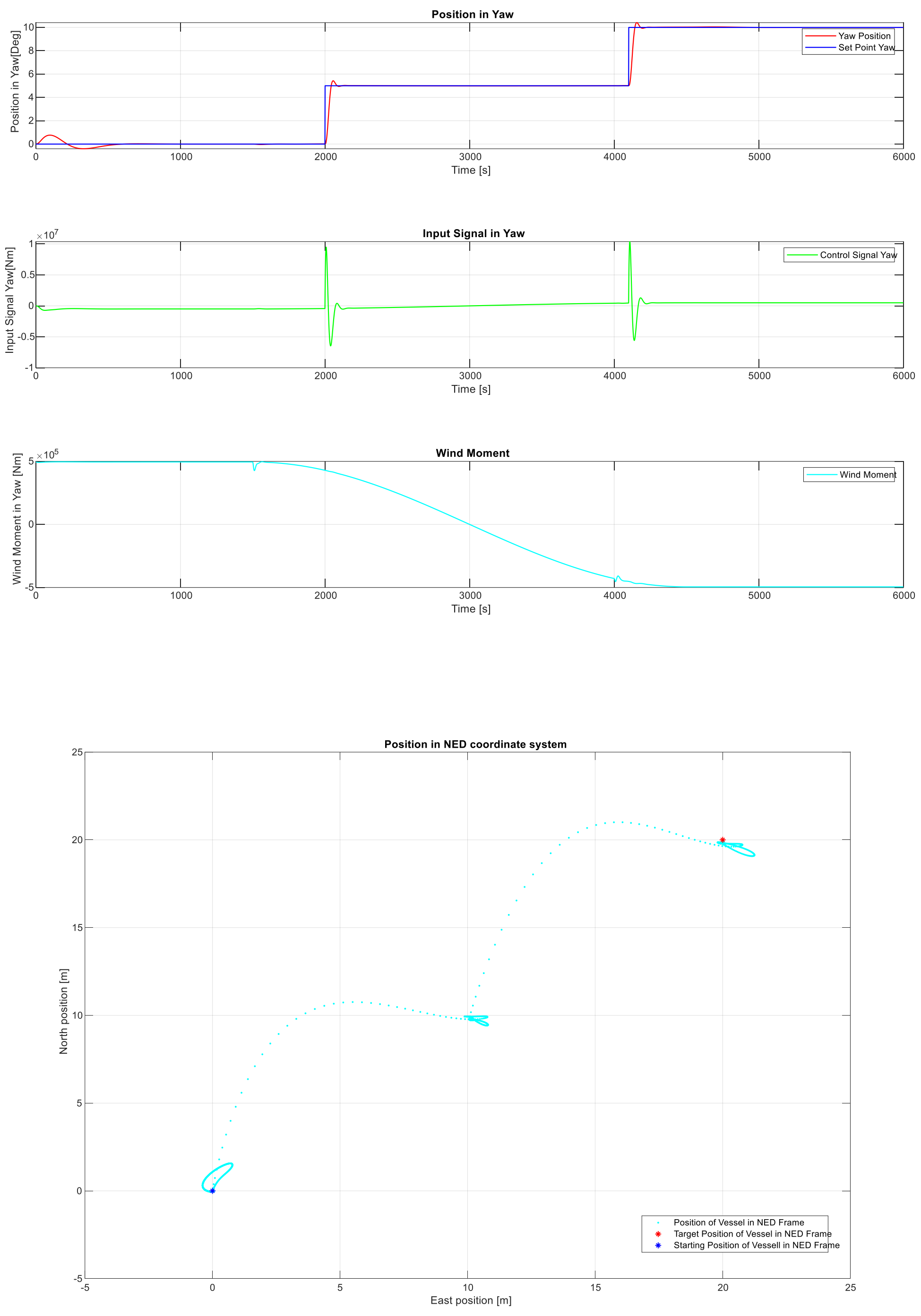


Figure 7-10 Performance of LQ Optimal Control with Integral Action without Input Limits

7.3.1.1 Settling time

The settling times are portrayed in Figure 7-11 and summarized in Table 7-12. There is a considerable improvement with regard to the PID controllers. The best settling time is shown in the surge control.

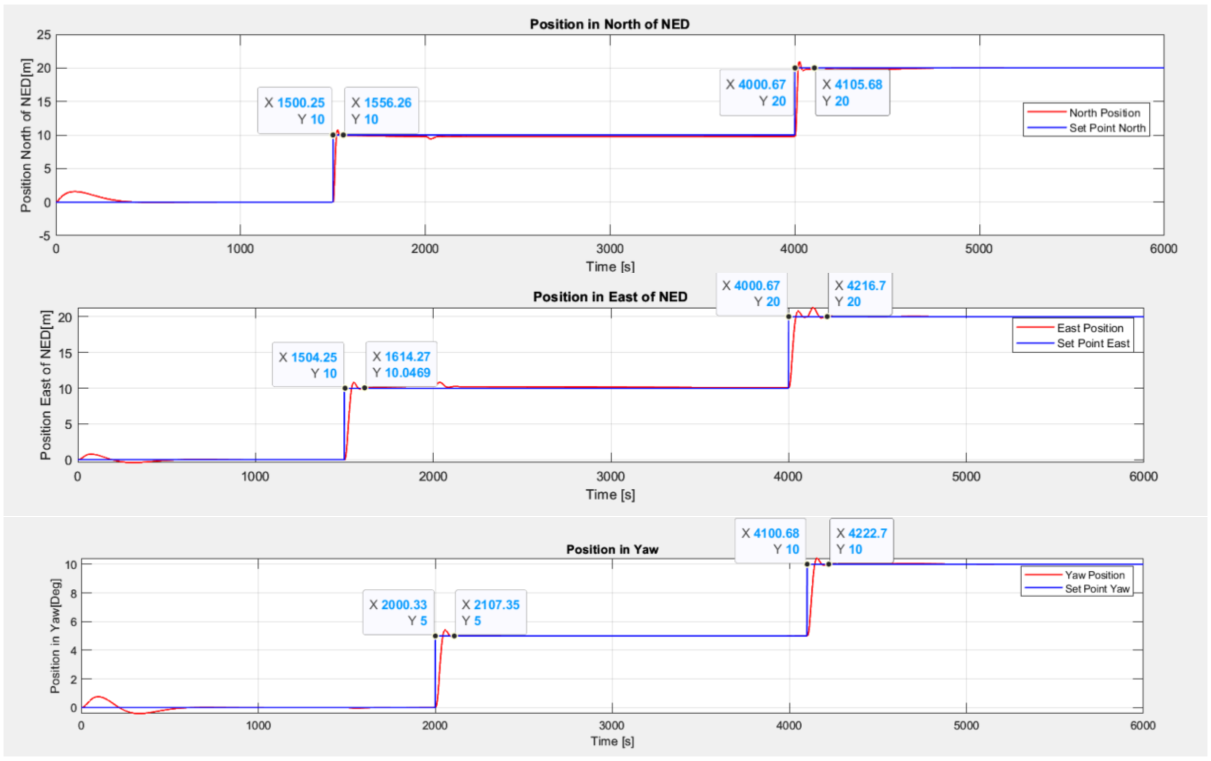


Figure 7-11 Settling Times for LQ Optimal Control without Input Limits

Table 7-12 Settling Times for LQ Optimal Control without Input Limits

Controller and Setpoint	Approximate Time [s]
Surge Controller Step 1	56
Surge Controller Step 2	105
Sway Controller Step 1	114
Sway Controller Step 2	216
Yaw Controller Step 1	107
Yaw Controller Step 2	222

7.3.1.2 Overshot and Oscillations

The overshot and oscillations of the LQ optimal controller with integral action are shown in Figure 7-12 and summarized in Table 7-13. The overshot is considerably less than with the PID controllers while the oscillation in the north and yaw axis is zero, and in the east axis is severely reduced.

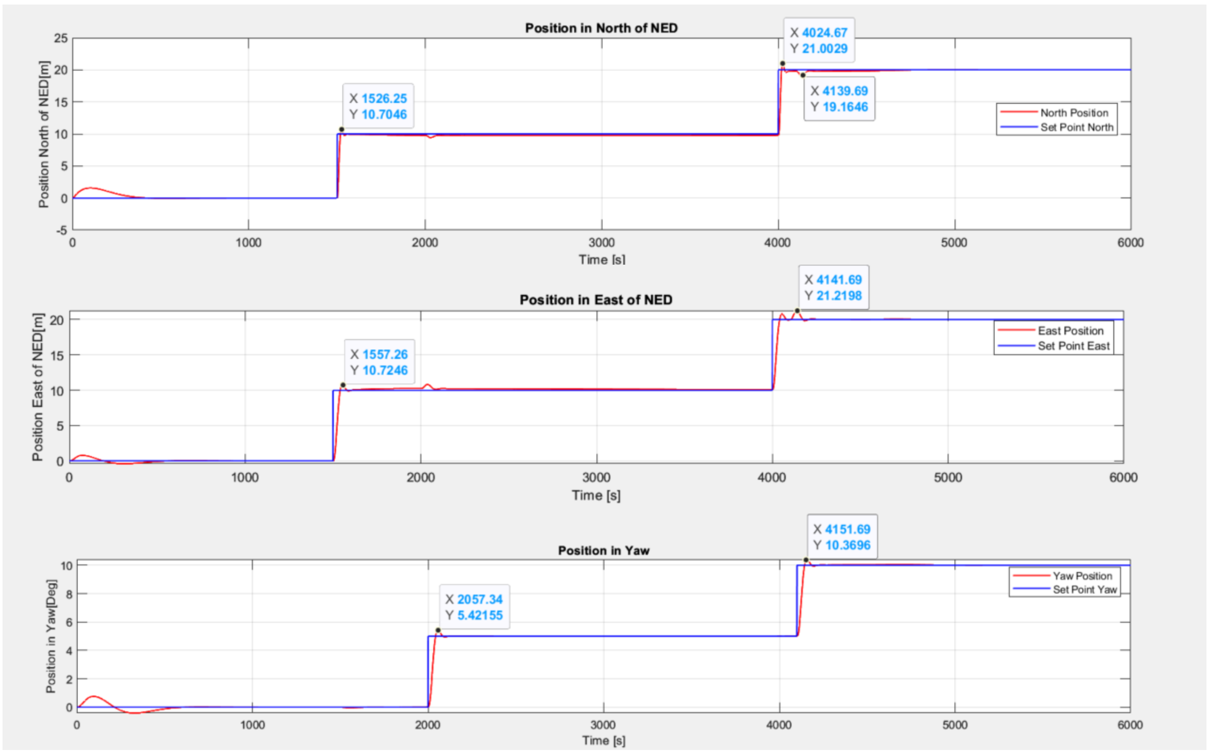


Figure 7-12 Overshot and Oscillations of LQ Optimal Control without Input Limits

Table 7-13 Overshot of LQ Optimal Control without Input Limits

Controller and Setpoint	Overshot [m or Deg.]
Surge Controller Step 1	0.7 [m]
Surge Controller Step 2	1 [m]
Sway Controller Step 1	0.7 [m]
Sway Controller Step 2	1.2 [m]
Yaw Controller Step 1	0.4 [Deg.]
Yaw Controller Step 2	0.3 [Deg.]

7.3.1.3 RMSE

The results can be seen in Table 7-14. These values are approximate zero, which indicates that for most of the simulation, the controller follows the setpoints almost exactly. This is remarkable considering the presence of the disturbances and model’s sensitivity to it.

Table 7-14 RMSE of LQ Optimal Controller without Input Limits

State	RMSE
North Position [m]	0.6166
East Position [m]	0.779
Yaw Angle [Deg.]	0.414

7.3.1.4 Maximum and Average U Values

The input values are shown in Figure 7-13 and the maximum and mean absolute values are shown in Table 7-15. The action in the yaw axis is higher than in the other axis.

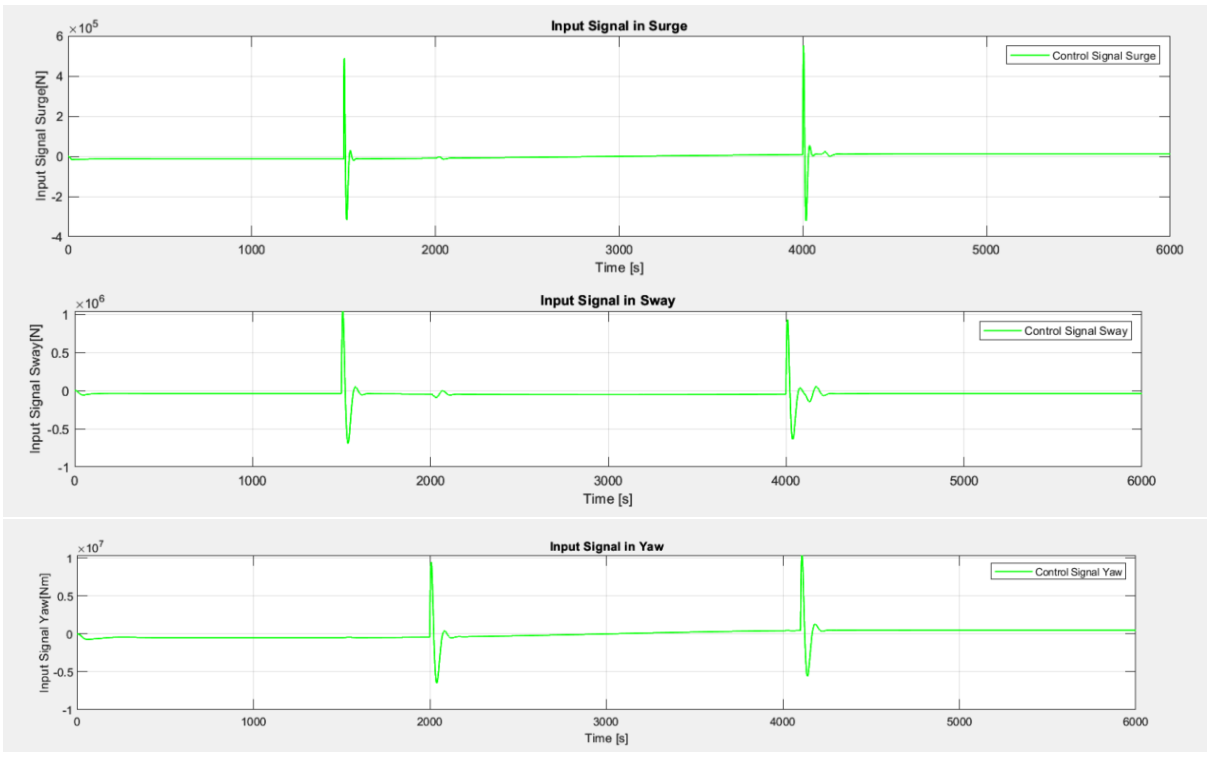


Figure 7-13 Input Values of LQ Optimal Control with Integral Action without Input Limits

Table 7-15 Input Values of LQ Optimal Control with Integral Action without Input Limits

State	U Value
Max. Surge [N]	5.54E+05
Mean Surge [N]	6.00E+00
Max. Sway [N]	1.04E+06
Mean Sway [N]	-4.35E+04
Max. Yaw [Nm]	1.04E+07
Mean Yaw [Nm]	2.17E+02

The total values are shown in Table 7-16.

Table 7-16 Total Values for LQ Controller without Control Signal Limits

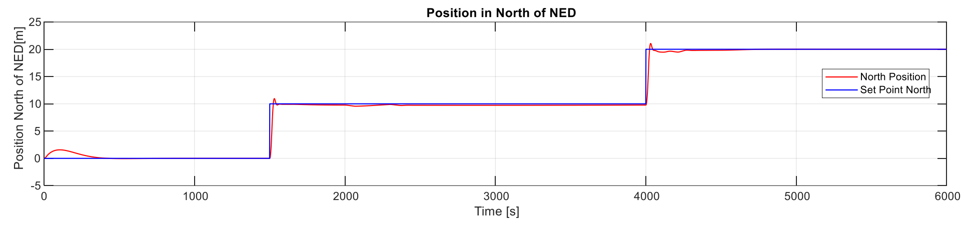
Controller	Total Value (TV)
Surge	3.69E+06
Sway	7.75E+06
Yaw	6.99E+07

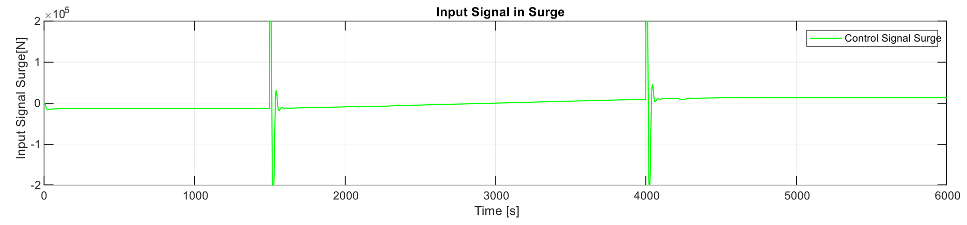
7.3.2 Performance Clipping the Control Signal

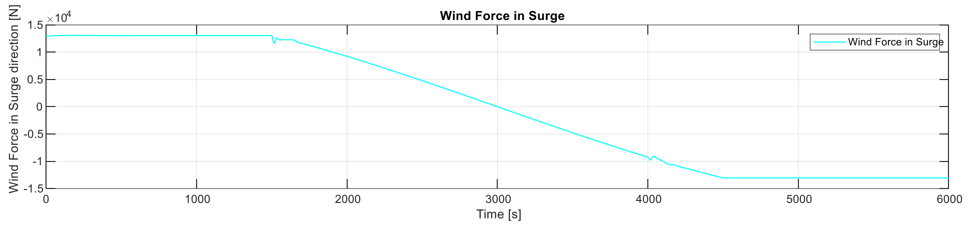
As the controller has shown to be more aggressive, it is also possible to try with lower limits for the input signals, the values for the sway controller are an order of magnitude smaller than in the case of the PID controllers.

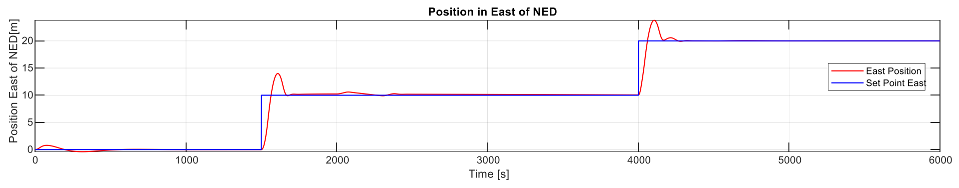
```
%Clipping the control signal
if abs(u(1)) > 2e5
    u(1) = 2e5*sign(u(1));
end
if abs(u(2)) > 2e5
    u(2) = 2e5*sign(u(2));
end
if abs(u(3)) > 1e6
    u(3) = 1e6*sign(u(3));
end
```

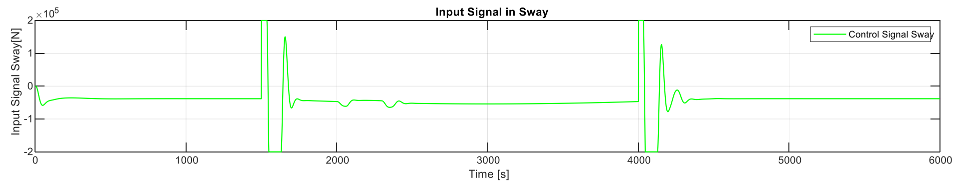
Figure 7-14 show the behavior of the system. Although the allowed control signal values are lower, the behavior of the controller does not seem too affected. There is an overshot in the yaw angle that stabilizes, then, the setpoints are matched almost perfectly, with a small delay for the second setpoint.

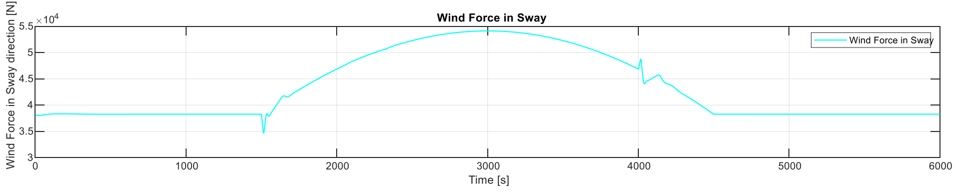

Position in North of NED
Position North of NED[m]
Time [s]
North Position
Set Point North


Input Signal in Surge
×10^5
Input Signal Surge[N]
Time [s]
Control Signal Surge


Wind Force in Surge
×10^4
Wind Force in Surge direction [N]
Time [s]
Wind Force in Surge


Position in East of NED
Position East of NED[m]
Time [s]
East Position
Set Point East


Input Signal in Sway
×10^5
Input Signal Sway[N]
Time [s]
Control Signal Sway


Wind Force in Sway
×10^4
Wind Force in Sway direction [N]
Time [s]
Wind Force in Sway

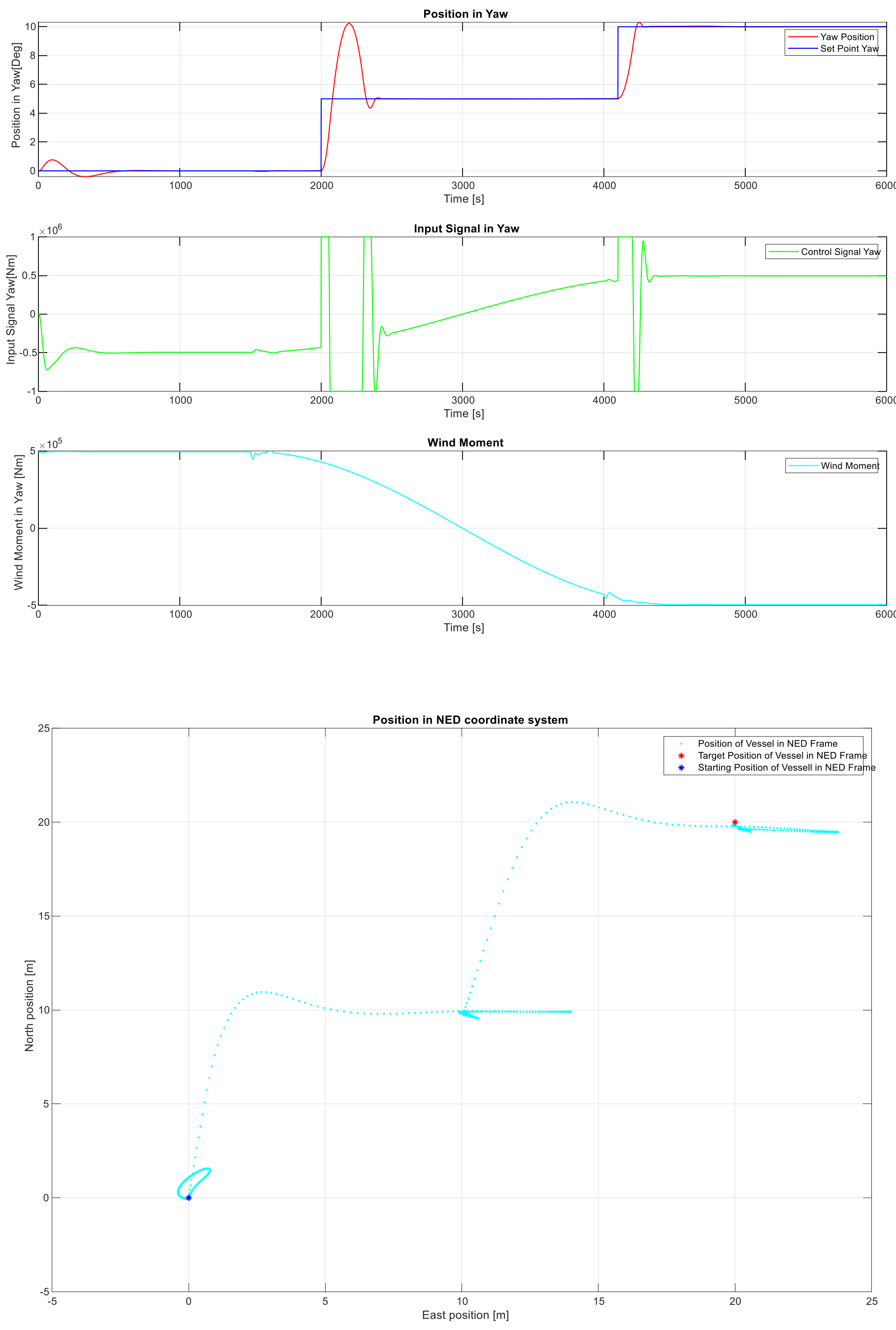


Figure 7-14 Performance of LQ Optimal Control with Integral Action with Input Limits

7.3.2.1 Settling Time

The settling times have increased but in a way that is not significative for the performance of the controller. This can be seen in Figure 7-15. The values are summarized in Table 7-17.

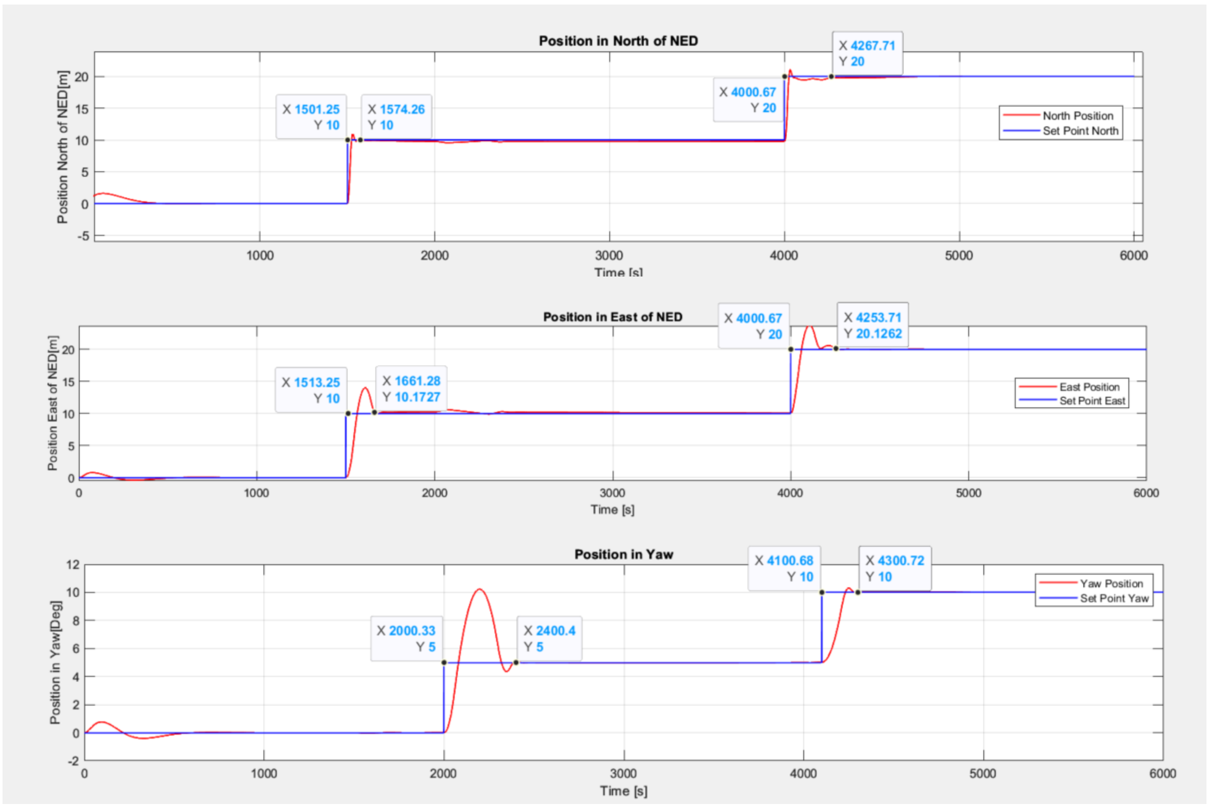


Figure 7-15 Settling Times for LQ Optimal Control with Input Limits

Table 7-17 Settling Times for LQ Optimal Control with Input Limits

Controller and Setpoint	Approximate Time [s]
Surge Controller Step 1	74
Surge Controller Step 2	267
Sway Controller Step 1	161
Sway Controller Step 2	253
Yaw Controller Step 1	400
Yaw Controller Step 2	300

7.3.2.2 Overshot and Oscillations

The results are shown in Figure 7-16 and summarized in Table 7-18. There is a larger overshot for the yaw controller in the first step, the setpoint is 5 degrees but the angle jumps to 10 degrees to then settle at 5 degrees, which is acceptable. Other than this, there is no considerable overshot, and the oscillations are non-existent.

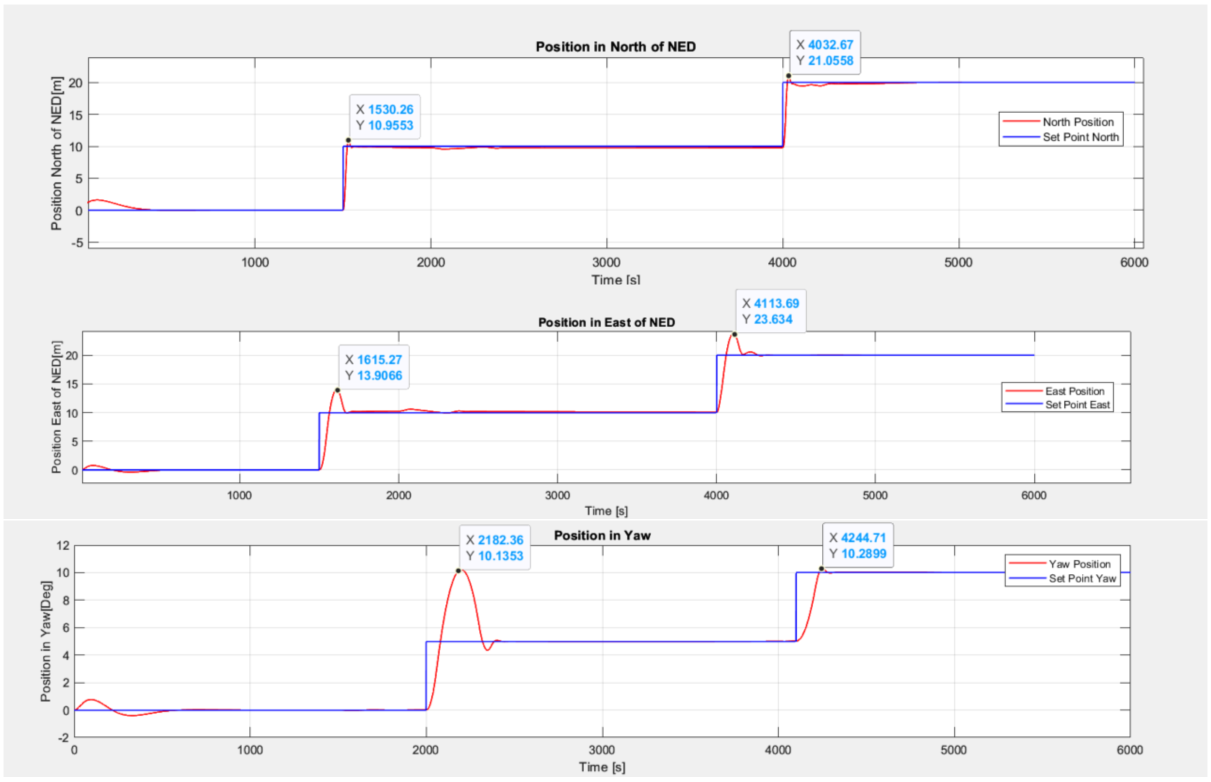


Figure 7-16 Overshot and Oscillations of LQ Optimal Control with Input Limits

Table 7-18 Overshot of LQ Optimal Control with Input Limits

Controller and Setpoint	Overshot [m or Deg.]
Surge Controller Step 1	0.9 [m]
Surge Controller Step 2	1.05 [m]
Sway Controller Step 1	3.9 [m]
Sway Controller Step 2	3.6 [m]
Yaw Controller Step 1	5.1 [Deg.]
Yaw Controller Step 2	0.2 [Deg.]

7.3.2.3 RMSE

Table 7-19 summarizes the values. The RMSE for the north position is like the result in the case without input limits, whereas it increases for the east position and the yaw angle. This has to do with the increase in the overshot for these two.

Table 7-19 RMSE of LQ Optimal Controller with Input Limits

State	RMSE
North Position [m]	0.6999
East Position [m]	1.1153
Yaw Angle [Deg.]	1.013

7.3.2.4 Maximum and Average U Values

Again, as seen in Figure 7-17 and in

Table 7-20, the maximum absolute values in the input signals correspond to the limits provided. The averages are like the values without input limits.

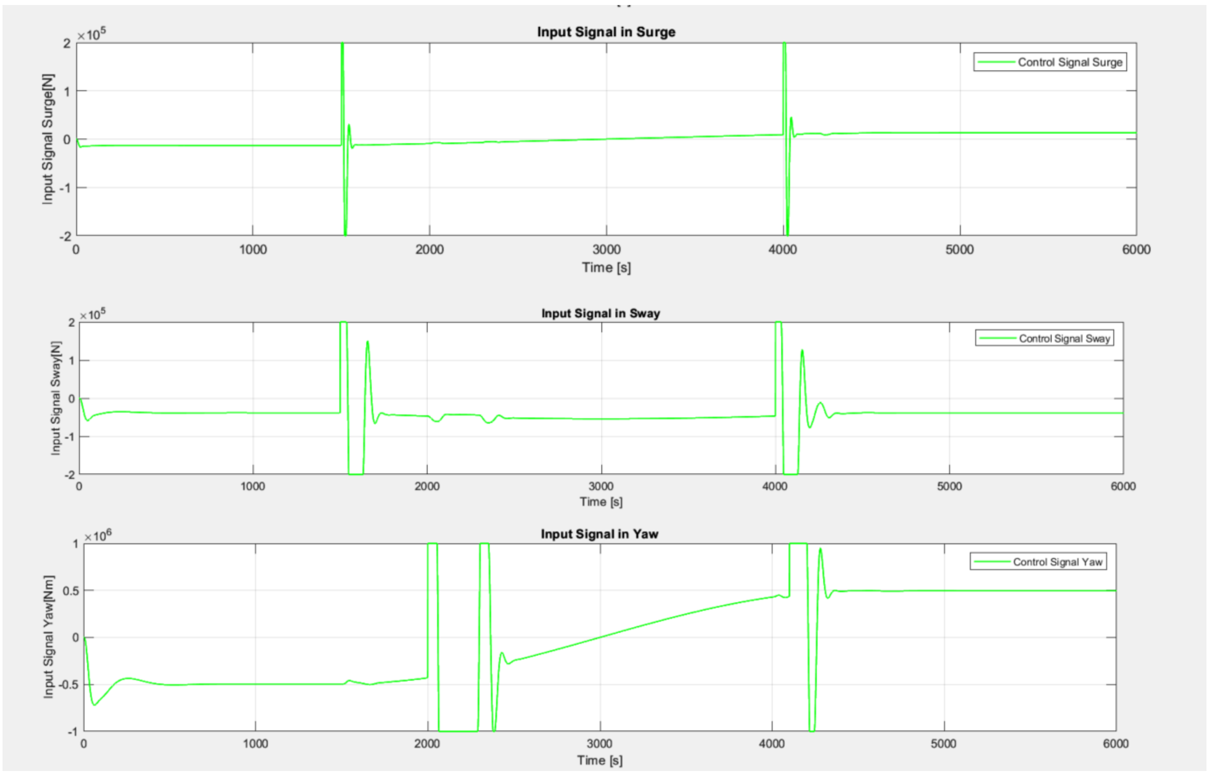


Figure 7-17 Input Values of LQ Optimal Control with Integral Action with Input Limits

Table 7-20 Input Values of LQ Optimal Control with Integral Action with Input Limits

State	U Value
Max. Surge [N]	2.00E+05
Mean Surge [N]	6.00E+00
Max. Sway [N]	2.00E+05
Mean Sway [N]	-4.35E+04
Max. Yaw [Nm]	1.00E+06
Mean Yaw [Nm]	2.19E+02

The Total Values can be seen in Table 7-21.

Table 7-21 Total Values for LQ with Control Signal Limits

Controller	Total Value (TV)
Surge	4.35E+06
Sway	2.48E+07
Yaw	6.62E+08

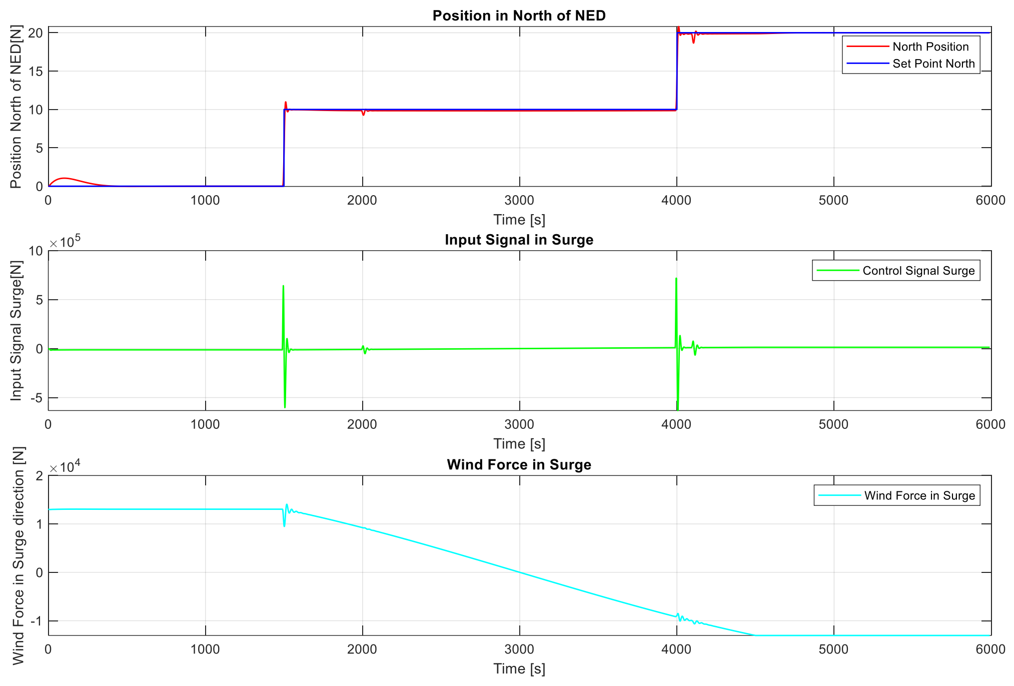
7.4 Performance with MPC and Integral Action

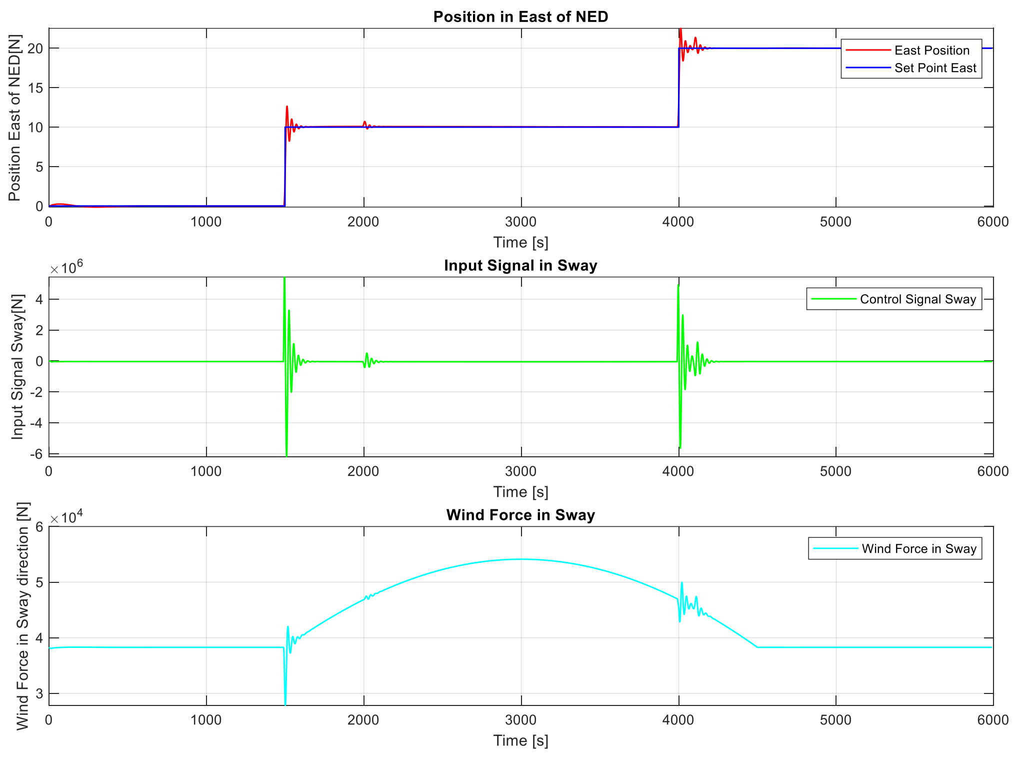
Now, the results with MPC and integral action are presented. This method has shown to be more sensitive to the variation in values on q and p than LQ with integral action, which makes necessary to use the original selected values:

```
% Controller weights
%Weighting parameters, p is for delta u, q is for error
q = [1e6 0 0; 0 1e7 0; 0 0 1e8];
p = [1e-4 0 0; 0 1e-4 0; 0 0 1e-10];
```

7.4.1 Performance Without Clipping the Control Signal

As seen in Figure 7-18, the controller follows the setpoints closely. There are some oscillations in the east position controller, but in general the behavior is very adequate. It also responds very quickly to the changes in the wind, and the changes in setpoint do not destabilize it.


Position in North of NED
Position North of NED[N]
North Position
Set Point North
Time [s]
Input Signal in Surge
Input Signal Surge[N]
$\times 10^5$
Control Signal Surge
Time [s]
Wind Force in Surge
Wind Force in Surge direction [N]
$\times 10^4$
Wind Force in Surge
Time [s]


Position in East of NED
Position East of NED[N]
East Position
Set Point East
Time [s]
Input Signal in Sway
Input Signal Sway[N]
$\times 10^6$
Control Signal Sway
Time [s]
Wind Force in Sway
Wind Force in Sway direction [N]
$\times 10^4$
Wind Force in Sway
Time [s]

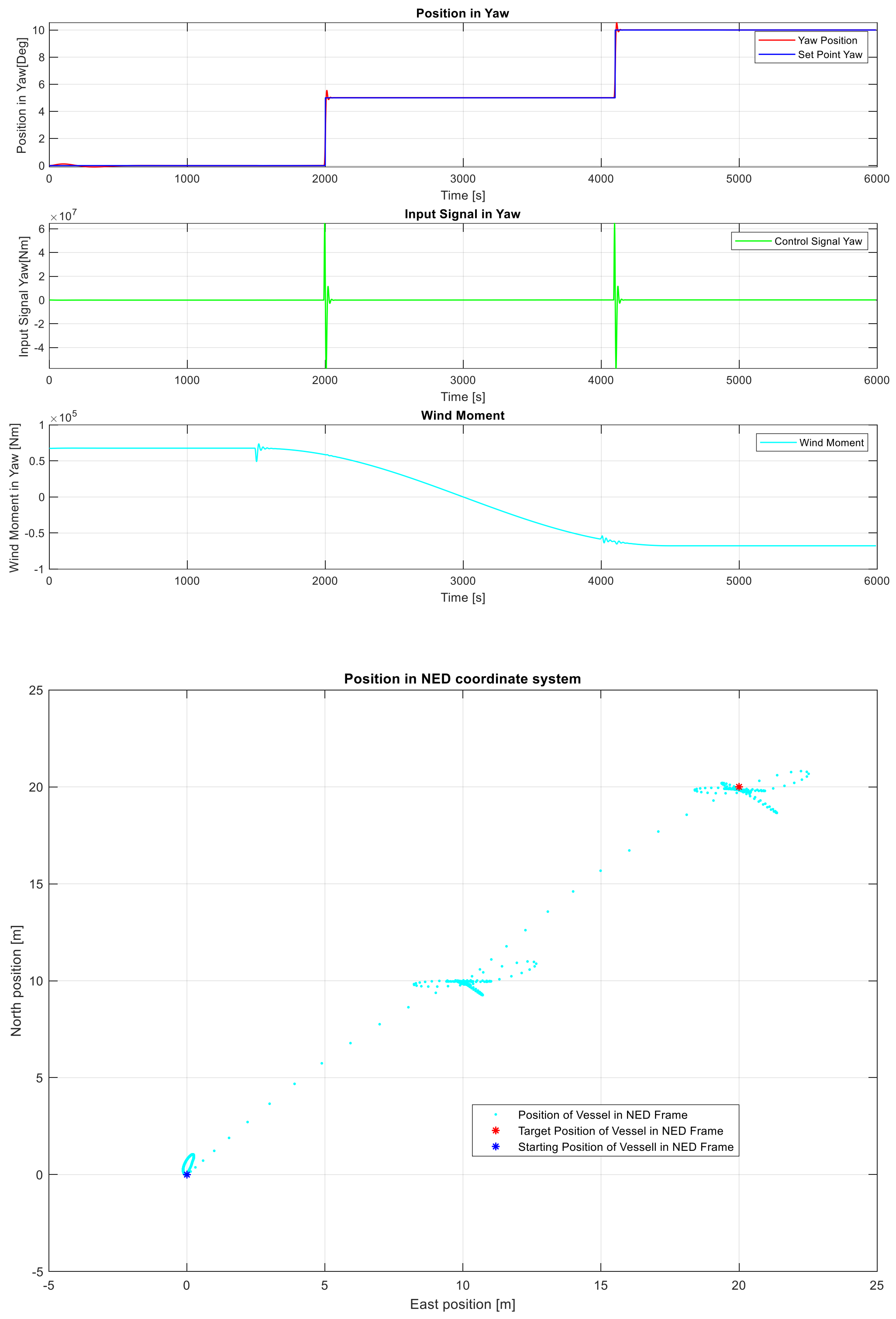


Figure 7-18 Performance of the System with Unconstrained MPC with Integral Action

7.4.1.1 Settling Time

The settling times are shown in Figure 7-19 and summarized in Table 7-22. There is a reduction of these times in comparison to the LQ optimal controller.

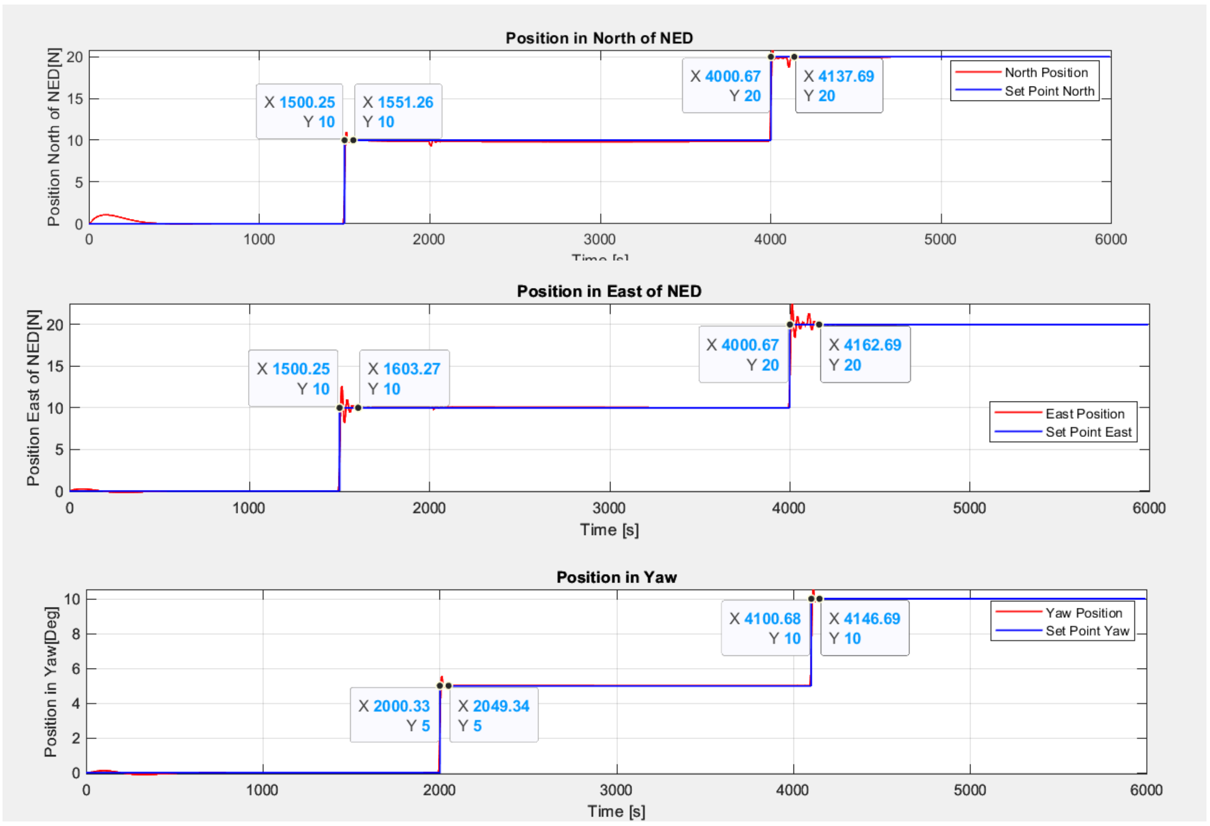


Figure 7-19 Settling Times for Unconstrained MPC with Integral Action

Table 7-22 Settling Times for Unconstrained MPC with Integral Action

Controller and Setpoint	Approximate Time [s]
Surge Controller Step 1	51
Surge Controller Step 2	137
Sway Controller Step 1	103
Sway Controller Step 2	162
Yaw Controller Step 1	49
Yaw Controller Step 2	146

7.4.1.2 Overshot and Oscillations

The overshot values are shown in Figure 7-20 and summarized in Table 7-23. There is a reduction regarding the LQ Optimal controller with integral action. Although there is an increase in the oscillations in the east position controller, these are not significant, and the controller manages to stabilize the vessel quickly. Also, the overshot in the yaw controller is considerably reduced.

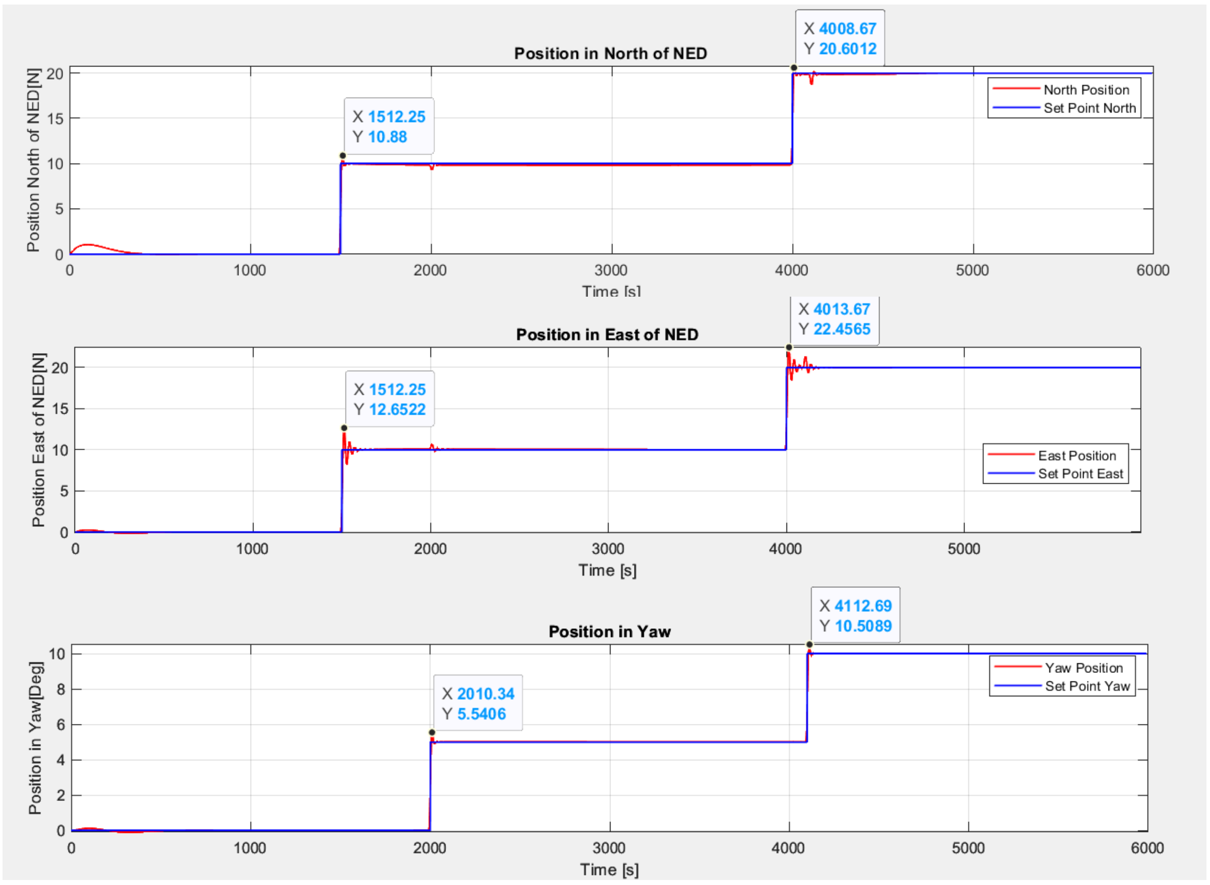


Figure 7-20 Overshot and Oscillations of Unconstrained MPC with Integral Action

Table 7-23 Overshot values of Unconstrained MPC with Integral Action

Controller and Setpoint	Overshot [m or Deg.]
Surge Controller Step 1	0.88 [m]
Surge Controller Step 2	0.6 [m]
Sway Controller Step 1	2.6 [m]
Sway Controller Step 2	2.4 [m]
Yaw Controller Step 1	0.5 [Deg.]
Yaw Controller Step 2	0.5 [Deg.]

7.4.1.3 RMSE

The RMSE values for the unconstrained MPC with integral action are shown in Table 7-24. There are maintained bellow 1m for the north and east controllers, and below 0.5 degrees for the yaw angle controller, which is adequate, meaning that there is practically no deviation from the setpoints except for the settling periods of the controller and the changes in the disturbances.

Table 7-24 RMSE of Unconstrained MPC with Integral Action

State	RMSE
North Position [m]	0.8615
East Position [m]	0.8583
Yaw Angle [Deg.]	0.4189

7.4.1.4 Maximum and Average U Values

Figure 7-21 and Table 7-25 portray the values for the u signals in the unconstrained MPC controller. This case shows increased oscillations, particularly for the east position controller.

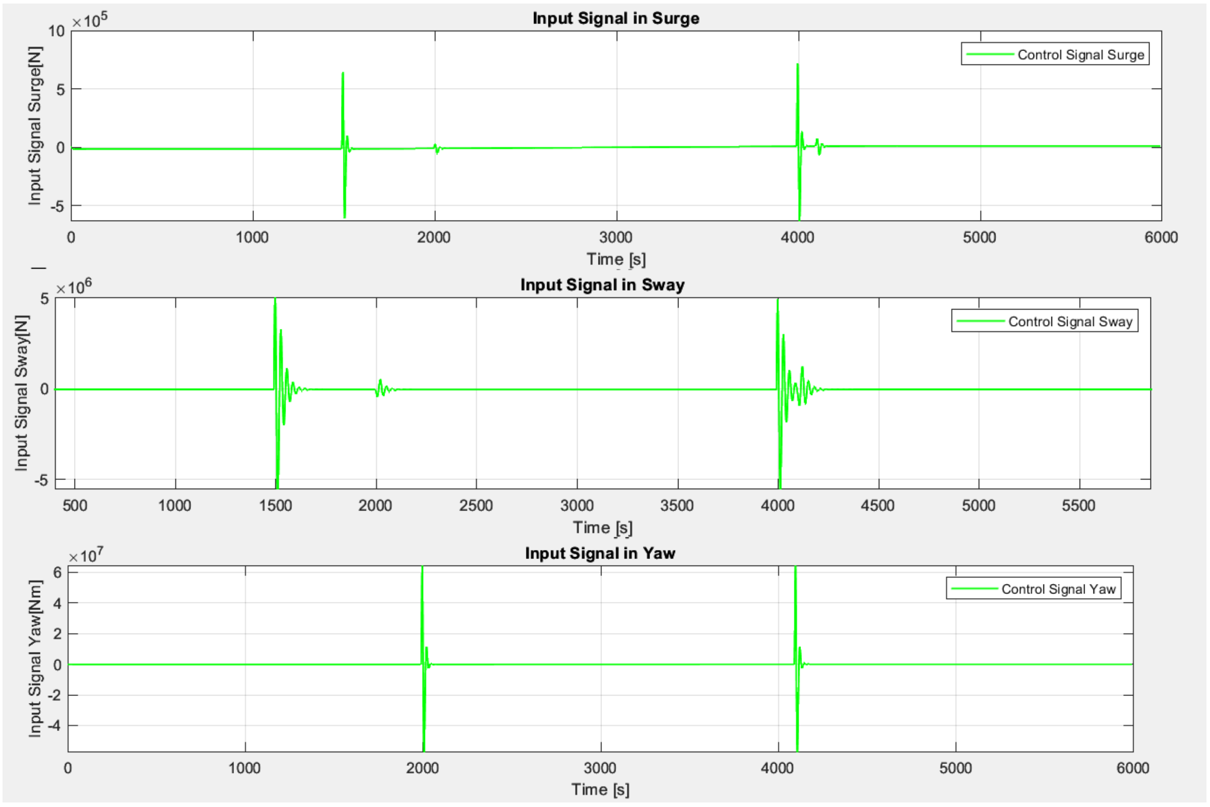


Figure 7-21 Input Values of Unconstrained MPC with Integral Action

Table 7-25 Maximum and Average Input Values of Unconstrained MPC with Integral Action

State	U Value
Max. Surge [N]	7.19E+05
Mean Surge [N]	-1.60E+01
Max. Sway [N]	6.19E+06
Mean Sway [N]	-4.34E+04
Max. Yaw [Nm]	6.45E+07
Mean Yaw [Nm]	-8.50E+01

The Total Values are found in Table 7-26.

Table 7-26 Total Values for MPC without Constraints in the Control Signals

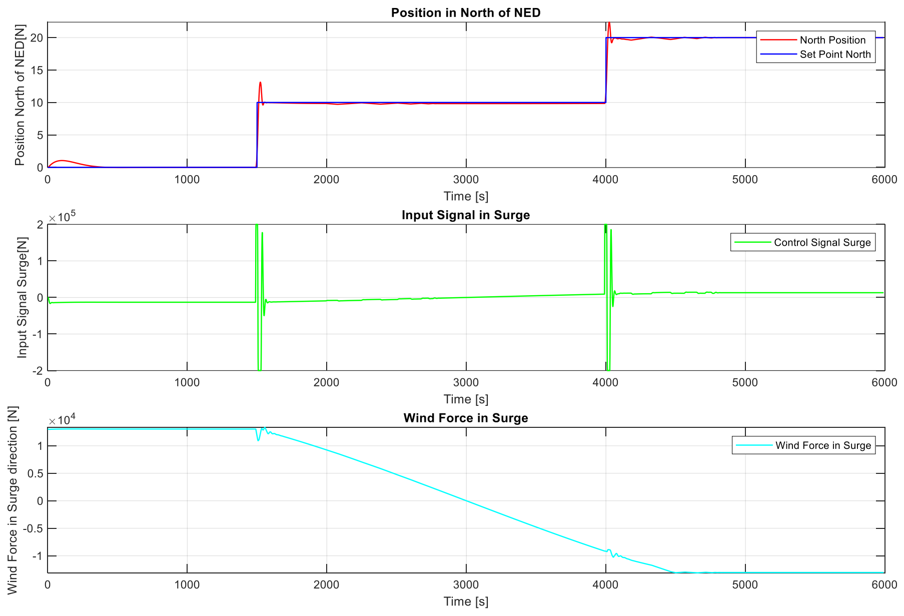
Controller	Total Value (TV)
Surge	6.41E+06
Sway	8.65E+07
Yaw	5.49E+08

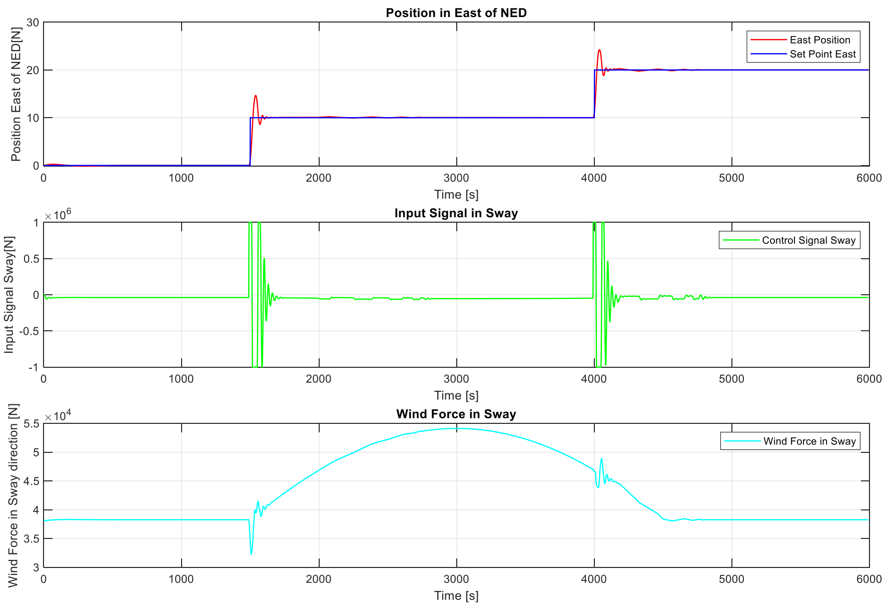
7.4.2 Performance Clipping the Control Signal

In the case of the constrained MPC, the quadprog function will be used. This function can be provided the limit values of u, which are defined as:

```
%Defining lowe and upper bounds for u
LB = [-2e5-u(1); -1e6-u(2);-1e6-u(3)];
UB = [2e5-u(1); 1e6-u(2);1e6-u(3)];
```

The system performs as shown in Figure 7-22. The north and east position controllers have a similar behaviour to the unconstrained case, whereas the yaw controller shows oscillations in the setpoint changes. However, the controllers seem to respond adequately to the disturbances.





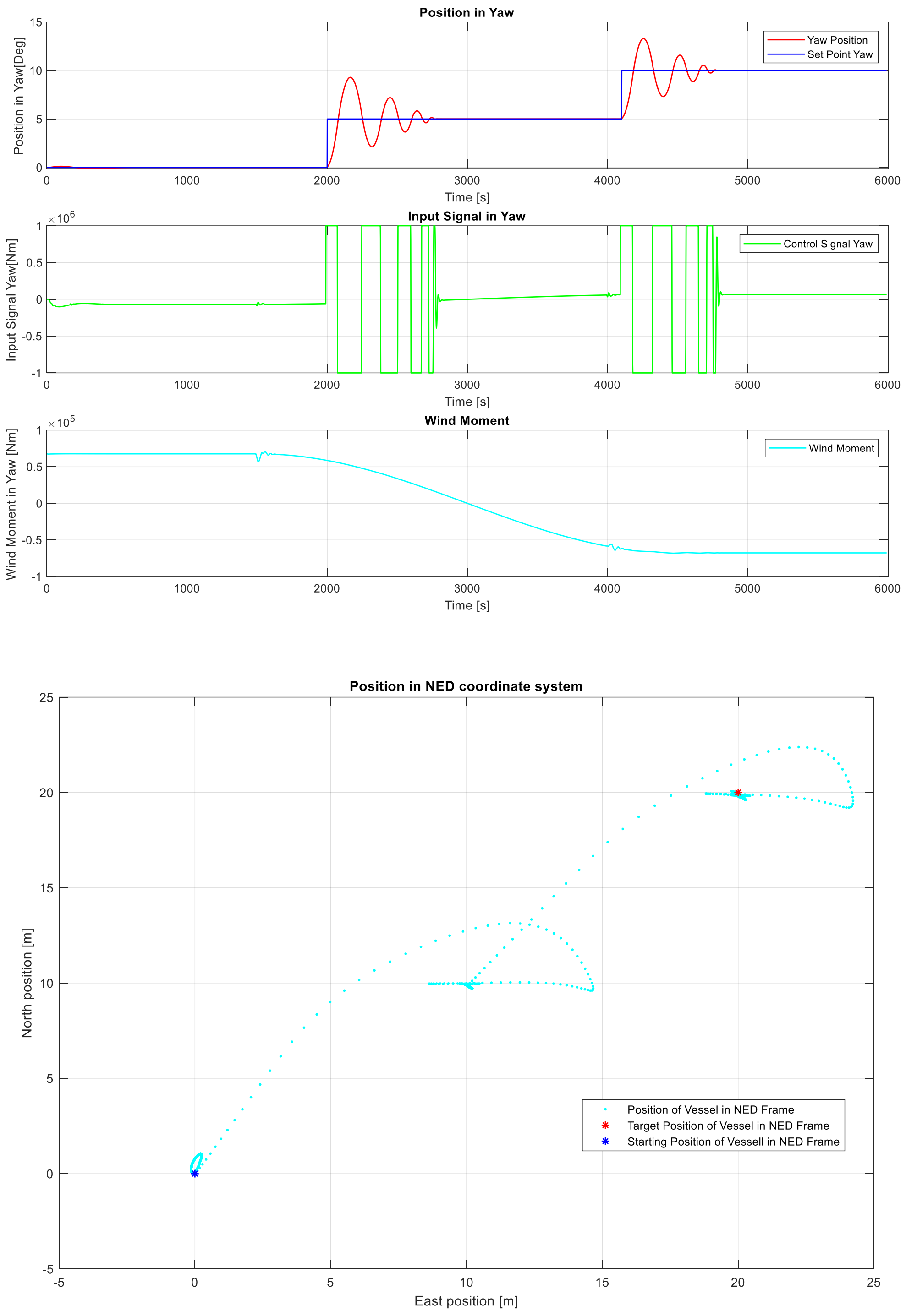


Figure 7-22 Performance for constrained MPC Controller with Integral Action

7.4.2.1 Settling Time

The settling times are shown in Figure 7-23 and summarized in Table 7-27. The values are maintained somewhat similar for the north and east controller. However, they increase considerably for the yaw controller. As the input signal is clipped, there is less momentum from the actuator, which in other terms means that there is less energy applied to adjust the yaw position, hence, the controller takes more time to stabilize the vessel. Nevertheless, this behavior continues to be acceptable, as the yaw is maintained in the setpoint after the settling.

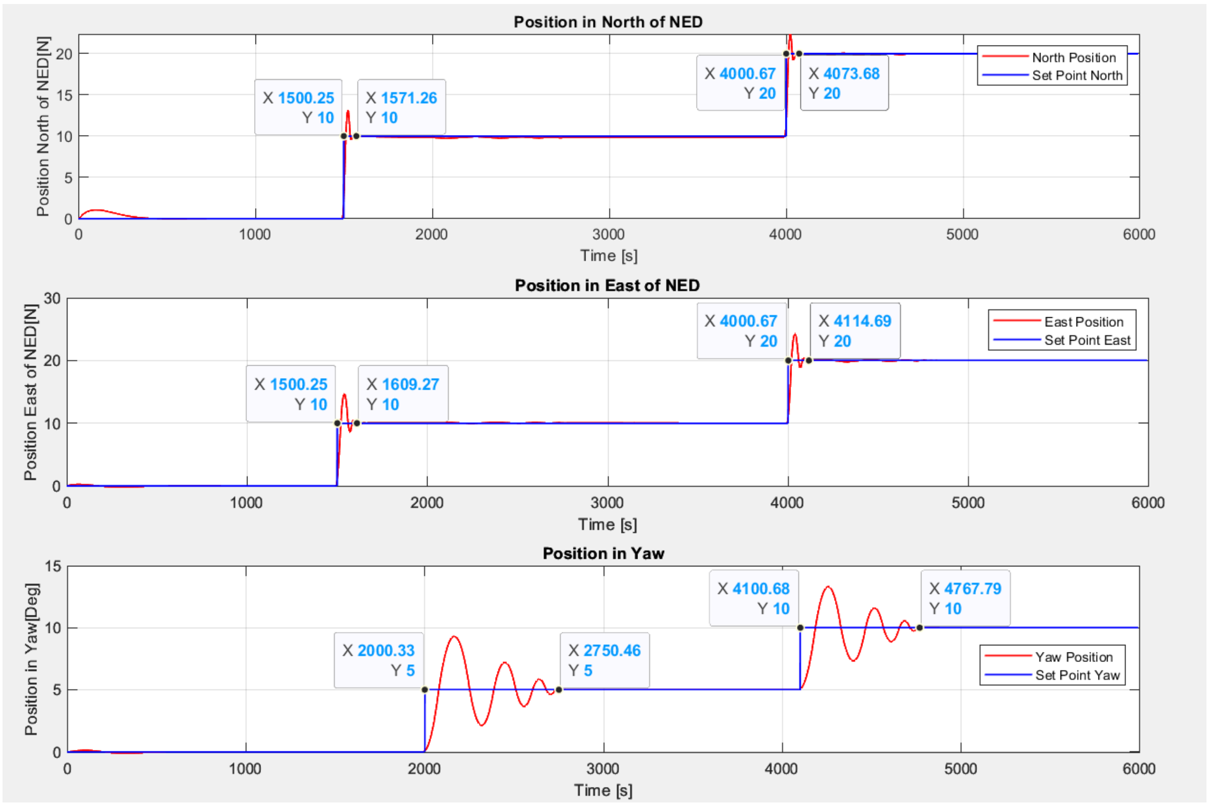


Figure 7-23 Settling Times for Constrained MPC with Integral Action

Table 7-27 Settling Times for Constrained MPC with Integral Action

Controller and Setpoint	**Approximate Time [s]**
Surge Controller Step 1	71
Surge Controller Step 2	73
Sway Controller Step 1	109
Sway Controller Step 2	114
Yaw Controller Step 1	750
Yaw Controller Step 2	667

7.4.2.2 Overshot and Oscillations

The values can be seen in Figure 7-24 and Table 7-28. The overshot increases in all the three states; however, this increment is more noticeable in the yaw angle, with 4.3 degrees for the first step increase, and 3.3 degrees for the second step increase. Nevertheless, as with the unconstrained case, these values are acceptable since they do not represent a considerable deviation and the controller keeps the vessel in the setpoint after the settling.

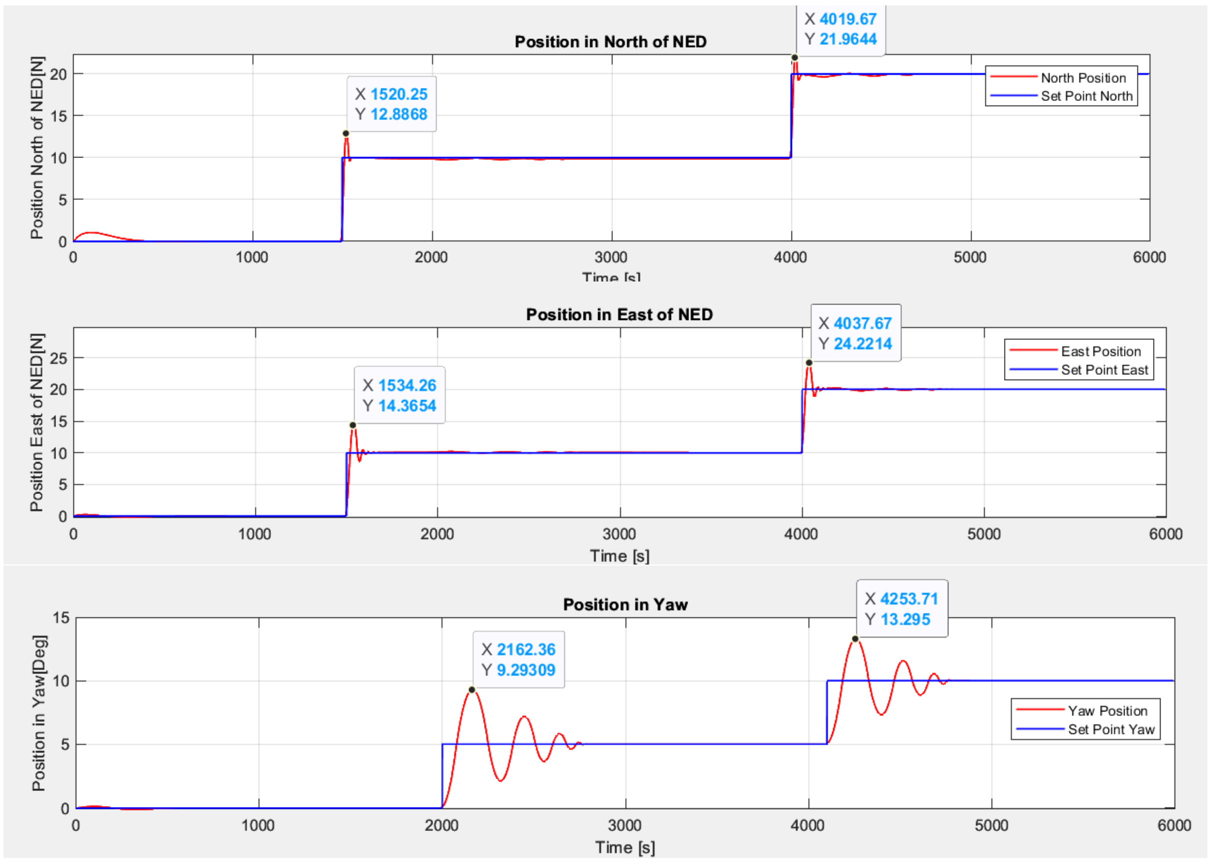


Figure 7-24 Overshot and Oscillations for Constrained MPC with Integral Action

Table 7-28 Overshot Values for Constrained MPC with Integral Action

Controller and Setpoint	Overshot [m or Deg.]
Surge Controller Step 1	2.8 [m]
Surge Controller Step 2	1.9 [m]
Sway Controller Step 1	4.3 [m]
Sway Controller Step 2	4.2 [m]
Yaw Controller Step 1	4.3 [Deg.]
Yaw Controller Step 2	3.3 [Deg.]

7.4.2.3 RMSE

The RMSE values are summarized in Table 7-29. There is a doubling in the value for the yaw angle. This is explained by the increase in oscillations and the longer settling time for this constrained controller.

Table 7-29 RMSE Values for Unconstrained MPC Controller with Integral Action

State	RMSE
North Position [m]	0.9201
East Position [m]	1.007
Yaw Angle [Deg.]	1.0974

7.4.2.4 Maximum and Average U Values

The values of the input signals are shown in Figure 7-25 and Table 7-30. Again, the maximum values correspond to the predefined limits. There is an increase in the changes in the yaw controller; the limit stops it to generate much bigger input signals, whereas for east and north, the limits are very close to the possible maximums, that is why the change in the behavior of north and east is not as drastic as with yaw.

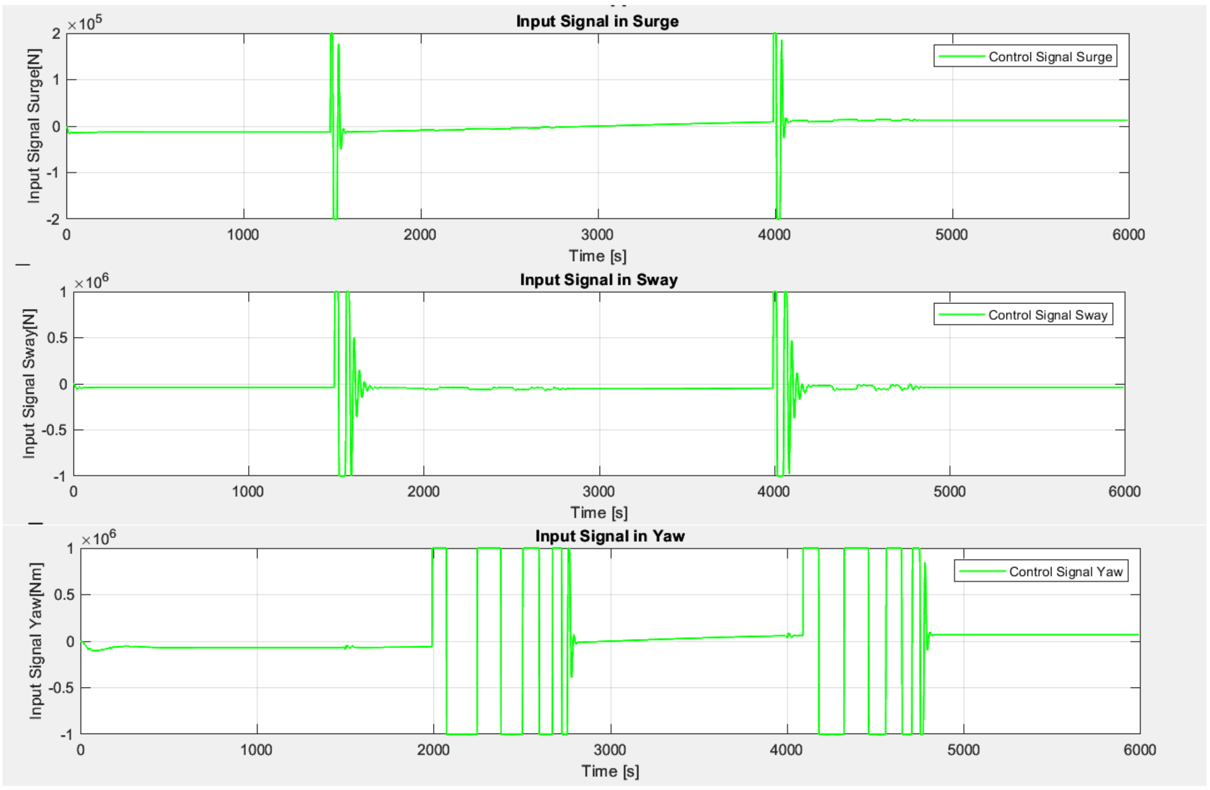


Figure 7-25 Input Values of Constrained MPC with Integral Action

Table 7-30 Maximum and Average Input Values for Constrained MPC with Integral Action

State	**U Value**
Max. Surge [N]	2.00E+05
Mean Surge [N]	-1.76E+02
Max. Sway [N]	1.00E+06
Mean Sway [N]	-4.34E+04
Max. Yaw [Nm]	1.00E+06
Mean Yaw [Nm]	-8.50E+01

The total values for u can be seen in Table 7-31.

Table 7-31 Total Values for MPC with Constrained Control Signals

Controller	Total Value (TV)
Surge	2.61E+06
Sway	2.23E+07
Yaw	3.77E+07

7.5 Comparison of the Three Methods

Having observed the behavior of the controllers and measured the settling times, overshot and oscillations, and the RMSE. It is possible to make a qualitative and quantitative comparison of the results. A score will be assigned to each system based on the previously selected benchmarks.

7.5.1 With Input Signal Limits

First, the comparison is made on the unconstrained controllers. In general, this configuration offers the best results with the tradeoff that in some cases, the input signals will reach high values, which in a real-life scenario could go against the specifications of the actuators in the vessel.

7.5.1.1 General Behavior

First, with a visual inspection of the three methods, it is determined, that all achieve the goal of reaching the setpoints and stabilizing the vessel around them. Also, their performance against disturbances is acceptable; so, one point per state to control is given to each, as shown in Table 7-32.

Table 7-32 Comparison of General Performance without Input Limits

Benchmark	PID	LQ + Integral Action	MPC + Integral Action
Performance North	Good	Good	Good
Performance East	Good	Good	Good
Performance Yaw	Good	Good	Good
Score	**3**	**3**	**3**

7.5.1.2 Settling Times

Although the three systems present acceptable settling times, there are some notable differences between the use of PID controllers and the other methos. LQ and MPC are almost twice as fast as the PIDs. The average settling time for the PIDs is 305.33 seconds, whereas for LQ and MPC it is 136.67, and 108 seconds respectively, as seen in Table 7-33. This is why two points are awarded to MPC and LQ, and only one to PID.

Table 7-33 Comparison of Settling Times Without Input Limits

Controller and Setpoint	PID [s]	LQ + Integral Action [s]	MPC + Integral Action [s]
Surge Controller Step 1	273	56	51
Surge Controller Step 2	294	105	137
Sway Controller Step 1	418	114	103
Sway Controller Step 2	324	216	162
Yaw Controller Step 1	288	107	49
Yaw Controller Step 2	235	222	146
Average [s]	**305.33**	**136.67**	**108**
Score	**1**	**2**	**2**

7.5.1.3 Overshot and Oscillations

After comparing the overshot values, it is concluded that despite the differences in average values. These are low and acceptable for all three controller types. Therefore, 2 points are awarded to each method. The summary can be seen in Table 7-34.

Table 7-34 Comparison of Overshots Without Input Limits

Controller and Setpoint	Overshot PID	Overshot LQ + Integral Action	Overshot MPC + Integral Action
North Controller Step 1 [m]	5.86	0.7	0.88
North Controller Step 2 [m]	6.5	1	0.6
East Controller Step 1 [m]	3.04	0.7	2.6
East Controller Step 2 [m]	2.44	1.2	2.4
Yaw Controller Step 1 [Deg.]	2	0.4	0.5
Yaw Controller Step 2 [Deg.]	1.8	0.3	0.5
Average for North and East [m]	**4.46**	**0.9**	**1.62**
Average for Yaw [Deg.]	**1.9**	**0.35**	**0.5**
Score	**2**	**2**	**2**

As per Table 7-35 shows, the oscillations for the North position in the PIDs and for the East position with MPC can be improved, nevertheless, all controllers have an at least acceptable behavior.

Table 7-35 Qualitative Comparison of Oscillations without Input Limits

Oscillations	PID	LQ + Integral Action	MPC + Integral Action
Performance North	Acceptable	Good	Good
Performance East	Good	Good	Acceptable
Performance Yaw	Good	Good	Good
Score	**2.5**	**3**	**2.5**

7.5.1.4 RMSE

The RMSE comparison can be seen in Table 7-36. The results are very similar for the three methods. Therefore, the whole score of 2 points is granted for all. It is worth noting that these RMSE values are low, which indicates that the three controllers behave appropriately.

Table 7-36 Comparison of RMSE without Limiting Input Signals

Axis	**RMSE PIDs**	**RMSE LQ + Integral Action**	**RMSE MPC + Integral Action**
North Position [m]	0.7923	0.6166	0.8615
East Position [m]	0.9551	0.779	0.8583
Yaw Angle [Deg.]	0.4744	0.414	0.4189
Score	**2**	**2**	**2**

7.5.1.5 Input Signals

The comparison of the maximum and mean input values for each state and each controller type can be found in Table 7-37. The highest absolute values are highlighted in yellow. It is noticeable that while unconstrained, MPC produces the highest input signals, which aligns with the aggressive response it showed.

The biggest differences are seen in the maximum absolute values for the sway and yaw controllers. These values, although very high, occur only in a very limited window of time, when the controller activates a response to a change in setpoints or disturbances.

It is worth noting that there is a difference in the mean values of the yaw controller, where the PID and LQ controllers have an absolute value close to 200, whereas the value for the MPC controller is 85. This conveys a general tendency of the LQ controller behavior to use bigger values of u in the yaw controller, and hence, use more energy.

Table 7-37 Comparison of Input Values without Limits for Input Signals

State	**U PIDs**	**U LQ + Integral Action**	**U MPC + Integral Action**
Max. Surge [N]	5.88E+05	5.54E+05	7.19E+05
Max. Sway [N]	1.20E+06	1.04E+06	6.19E+06
Max. Yaw [Nm]	2.21E+07	1.04E+07	6.45E+07
Mean Surge [N]	6.00E+00	6.00E+00	-1.60E+01
Mean Sway [N]	-4.35E+04	-4.35E+04	-4.34E+04
Mean Yaw [Nm]	2.11E+02	2.17E+02	-8.50E+01

To have more insight into the performance of the control signals, it is a good idea to compare the total values of the three methods, the score will be calculated based on the Total Value Results.

Table 7-38 Comparison of Total Values for unconstrained input signals

Controller	**PIDs TV**	**LQ + Integral Action TV**	**MPC + Integral Action TV**
Surge	5.92E+06	3.69E+06	6.41E+06
Sway	8.02E+06	7.75E+06	8.65E+07
Yaw	1.52E+08	6.99E+07	5.49E+08
Score	**2.1**	**3**	**1.5**

As seen in Table 7-38, the total usage of u is bigger in the MPC with integral action controller (highlighted in yellow), and less in LQ with integral action. In this benchmark, the best result is obtained from LQ with integral action, so the biggest score is given to this controller.

7.5.1.6 Scores and Controller Selection

Table 7-39 contains the count of all the scores, wherein the highest score was achieved by the LQ Optimal controller with Integral action. This controller showed good general behavior and the generated control signals do not have the highest peaks. Nevertheless, more tuning is needed to obtain a better performance.

Table 7-39 Scores Summary Without Control Signal Limits

Benchmark	**PIDs**	**LQ + Integral Action**	**MPC + Integral Action**
General Behaviour	3	3	3
Settling Times	1	2	2
Overshot	2	2	2
Oscillations	2.5	3	2.5
RMSE	2	2	2
U usage (TV)	2.1	3	1.5
Total	**12.6**	**15**	**13**

7.5.2 Without Limiting Input Signals

Now the results for the controllers with constrained control signal will be compared.

7.5.2.1 General Behavior

The controllers maintain a general adequate performance, wherein the vessel is taken to the given setpoints and stabilized. Nevertheless, there is an increase in oscillations in yaw for the three controllers. The general assessment is seen in Table 7-40.

Table 7-40 Comparison of General Performance with Input Limits

Benchmark	**PID**	**LQ + Integral Action**	**MPC + Integral Action**
Performance North	Good	Good	Good
Performance East	Good	Good	Good
Performance Yaw	Acceptable	Acceptable	Acceptable
Score	**2.5**	**2.5**	**2.5**

7.5.2.2 Settling Times

There is an increase in the settling times, which is to be expected as the input signals are limited. The results summarized in Table 7-41 show that the settling times average double for the MPC and LQ controllers and almost triples for the PID controllers. The most increase is seen in the yaw controllers, which aligns with the general behavior comparison.

Table 7-41 Comparison of Settling Times Limiting Input Signals

Controller and Setpoint	PID [s]	LQ + Integral Action [s]	MPC + Integral Action
Surge Controller Step 1	283	74	71
Surge Controller Step 2	600	267	73
Sway Controller Step 1	430	161	109
Sway Controller Step 2	850	253	114
Yaw Controller Step 1	1802	400	750
Yaw Controller Step 2	1761	300	667
Average	**954.33**	**242.50**	**297**
Score	**0.5**	**1**	**1**

7.5.2.3 Overshot and Oscillations

There is an increase in the overshot values for all the controllers in the three states, as seen in Table 7-42. However, the values are deemed appropriate.

Table 7-42 Comparison of Overshot Values Limiting Input Signals

Controller and Setpoint	PID [m or Deg.]	LQ + Integral Action [m or Deg.]	MPC + Integral Action [m or Deg.]
North Controller Step 1 [m]	3.46	0.9	2.8
North Controller Step 2 [m]	2.8	1.05	1.9
East Controller Step 1 [m]	2.6	3.9	4.3
East Controller Step 2 [m]	2.3	3.6	4.2
Yaw Controller Step 1 [Deg.]	0.6	5.1	4.3
Yaw Controller Step 2 [Deg.]	2.1	0.2	3.3
Average for North and East [m]	**2.79**	**2.36**	**3.3**
Average for Yaw [Deg.]	**1.35**	**2.65**	**3.8**
Score	**2**	**2**	**2**

The oscillations comparison is shown in Table 7-43. There is an improvement in the oscillations in the MPC controller for the east position, with the tradeoff of more oscillations in yaw. The PID controllers show some oscillations in the North position that are not a big concern for the performance of the system.

Table 7-43 Comparison of Oscillations Limiting Input Signals

Oscillations	PID	LQ + Integral Action	MPC + Integral Action
Performance North	Acceptable	Good	Good
Performance East	Good	Good	Good
Performance Yaw	Good	Good	Acceptable
Score	**2.5**	**3**	**2**

7.5.2.4 RMSE

The RMSE increase in all controllers, but they are maintained in an acceptable range, for this reason, the score is maintained, the comparison is given in Table 7-44.

Table 7-44 Comparison of RMSE with Input Limits

Axis	RMSE PIDs	RMSE LQ + Integral Action	RMSE MPC + Integral Action
North Position [m]	0.7965	0.6999	0.9201
East Position [m]	0.9608	1.1153	1.007
Yaw Angle [Deg.]	1.0256	1.013	1.0974
Score	**2**	**2**	**2**

7.5.2.5 Input Signals

The maximum values are the same in all controllers because of the limits. The maximum mean values are found in the MPC for the surge action, and in the PIDs for the yaw action. In Sway, the mean values for all the controllers are very similar. As seen in Table 7-45.

Table 7-45 Comparison of Input Values with Limits for Input Signals

State	U PIDs	U LQ + Integral Action	U MPC + Integral Action
Max. Surge [N]	2.00E+05	2.00E+05	2.00E+05
Max. Sway [N]	1.00E+06	2.00E+05	1.00E+06
Max. Yaw [Nm]	1.00E+06	1.00E+06	1.00E+06
Mean Surge [N]	6.00E+00	6.00E+00	-1.76E+02
Mean Sway [N]	-4.35E+04	-4.35E+04	-4.34E+04
Mean Yaw [Nm]	4.44E+02	2.19E+02	-8.50E+01

The total values summary is found in Table 7-46. There is change in comparison to the case without input limits. Now, the PID controller has the most usage of u in Surge and Yaw, whereas the largest value for sway is for LQ. It is also worth noting that the total values increase for LQ and PID controllers with the control signal constraints, while they decrease for the MPC method. Therefore, MPC is the best in this benchmark.

Table 7-46 Comparison of Total Values of u for with Limits in the Input Signals

Controller	PIDs TV	LQ + Integral Action TV	MPC + Integral Action TV
Surge	8.97E+06	4.35E+06	2.61E+06
Sway	7.21E+06	2.48E+07	2.23E+07
Yaw	8.80E+08	6.62E+08	3.77E+07
Score	**1.5**	**1.5**	**3**

7.5.2.6 Scores and Controller Selection

The final scores for the case with restrained control signals are found in Table 7-47. The performance of the controllers decreases with regards to the case without input constraints, but the controllers behave adequately enough. Nevertheless, MPC proves to be more resilient to changes in the constraints by having a smaller total value index, which aligns to the logic of MPC control, where a set of equality and inequality constraints, and input limits are considered for the calculation of u. In this case, the best option is the MPC control with integral action.

Table 7-47 Scores Summary with Control Signal Limits

Benchmark	PIDs	LQ + Integral Action	MPC + Integral Action
General Behaviour	2.5	2.5	2.5
Settling Times	0.5	1	1
Overshot	2	2	2
Oscillations	2.5	3	2
RMSE	2	2	2
U usage (TV)	2	2.5	3
Total	**11**	**12**	**12.5**

7.6 Analysis of the Results

The three control methods show acceptable behavior either with unlimited or limited control signals; in all methods and with the two configurations, the setpoints are followed closely and when the setpoint is constant, the position settles nicely. However, the response of the PID deteriorates the most when the limits are set in place.

The settling times are acceptable for the three methods in the constrained and unconstrained case, but the increase is more substantial for the PID controllers.

The overshot is kept in adequate values for all the cases and there is even an improvement in the values for the PID controllers, which Is seen when comparing Table 7-34 to Table 7-42.

The oscillations deteriorate with the limits in the control signals, increasing considerable in the MPC controller for the yaw angle. Nevertheless, some slow oscillations are to be expected giving the type of application.

The RMSE values for the three controllers in the constrained and unconstrained cases are good, this aligns with the overall behavior of the controllers.

Perhaps the deciding factor is the usage of u. The comparison of the total values of u are replicated here for easy of reading in Table 7-48 and

Table 7-49. The highest total values for both cases are highlighted in yellow and the lowest in green. It is seen that PID and LQ have lower usage for the unconstrained case, but when the constraints are implemented, the lower values are in MPC, for all the cases. This hints some capability of the MPC method to better adapt and account for the constraints and would make it a better option over the other two.

It comes with a cost though, as the use of quadprog makes the execution of the main control loop much slower.

Table 7-48 Comparison of Total Values for unconstrained input signals

Controller	PIDs TV	LQ + Integral Action TV	MPC + Integral Action TV
Surge	5.92E+06	3.69E+06	6.41E+06
Sway	8.02E+06	7.75E+06	8.65E+07
Yaw	1.52E+08	6.99E+07	5.49E+08
Score	**2.1**	**3**	**1.5**

Table 7-49 Comparison of Total Values of u for with Limits in the Input Signals

Controller	**PIDs TV**	**LQ + Integral Action TV**	**MPC + Integral Action TV**
Surge	8.97E+06	4.35E+06	2.61E+06
Sway	7.21E+06	2.48E+07	2.23E+07
Yaw	8.80E+08	6.62E+08	3.77E+07
Score	**1.5**	**1.5**	**3**

8 Conclusions

After the implementation of the model, it is observed that its behavior does not change considerably after the drag coefficients are approximated to zero. This measure was taken since the drag coefficients have very small magnitudes and in equations (2.4), (2.5), and (2.6); it is seen that they are divided by the momentum coefficients, which are substantially bigger. After comparing both versions of the model, there is not a noticeable difference in behavior, as the covariance between them for all the states was close to the computer's mathematical zero.

The use of a PID controller poses an advantage over what could be called "model dependent" methods, since there is no need for a deep mathematical understanding of the system before being able to tune a controller that shows an adequate behavior. This was seen in chapter 3, where the PID controllers were tuned using the Ziegler-Nichols method. Nevertheless, it was also noticed that the controllers are very susceptible to disturbances and their parameters needed to be tuned with every change in the windspeed values.

The use of the PID controllers provides an aiding hand when performing the system identification. The position and input signals resulting from the PID simulation were then used as inputs in the dsr_e function to obtain the matrices A, B, and D of the estimated linearized model. This provided a much more accurate model which in turn could be better controlled with LQ and MPC methods.

The tuning of Q and P matrices in LQ Optimal control with integral action is analog to the tuning of a PID controller. It is a trial-and-error process where a balance had to be reached between reducing the error and limiting the variation in the control signal u. Nevertheless, LQ is a much more robust controller which accounts for the previously obtained model matrices and a Kalman filter for state estimation, which makes its response considerably better than PID's.

It could be argued that MPC is simply the implementation of LQ Optimal control in every iteration of the control loop with the addition of a sliding horizon. The main advantage of MPC is that it provides a way to account for inequality and equality constraints to obtain an optimized possible response. However, this increases the computational costs considerably.

The comparison of the three methods under the same disturbance conditions show that in general, all have an acceptable performance, being LQ Optimal Control with Integral Action the best for unconstrained control signals and MPC with integral action when the signals are constrained. Nevertheless, MPC is interesting as the usage of u was reduced when the constraints were implemented, whereas with PID and LQ this increased. So, in this case, MPC with integral action seems like the best option.

It is important to note that in real-life cases, more than one method is used to control complex systems like a marine vessel. MPC poses a challenge due to the high computational costs, which only increase when more constraints are added, or the prediction horizon is increased. As per [5]. A good idea would be to use MPC to perform a high-level control, in which It is used to establish setpoints or most optimal operational windows, while LQ optimal control with Integral action or the PID controllers are used to do the setpoint tracking.

References

[1] N. A. J. E. M. S. S. Jens G. Balchen, "A Dynamic Positioning System Based on Kalman Filtering and Optimal Control," *Modelling, Identification and Control*, vol. 1, no. 3, pp. 135-163, 1980.

[2] C. D. D. D. R. Nour Bargoth, "Dynamic positioning, system identification and," *Modeling, Identification and Control,* vol. 43, no. 3, pp. 111-117, 2022.

[3] N. M. Bargouth, "Dynamic positioning, system identification and control of marine vessels," University of South-Eastern Norway, Porsgrunn, 2022.

[4] H. P. L. Svein Linge, "Solving Ordinary Differential Equations," in *Programming for Computations - Python*, Springer Open, 2020, pp. 254-257.

[5] R. Sharma, Lecture notes for the course IIA 4117: Model Predictive Control, Porsgrunn: University of South-Eastern Norway, 2019.

[6] H.-P. Halvorsen, "Hardware in the Loop Simulation and Testing," [Online]. Available: https://www.halvorsen.blog/documents/teaching/courses/industrialit/lab_assignment/Hardware-in-the-Loop%20Simulation%20Lab.pdf. [Accessed 01 03 2023].

[7] F. A. Haugen, "Tuning of PID Controllers," in *Modeling, Simulation and Control*, Porsgrunn, University of South-Eastern Norway, 2021, pp. 290-294.

[8] D. D. Ruscio, "D-SR Toolbox for MATLAB," [Online]. Available: https://davidr.no/iia2217/d-sr/d-sr_e.html. [Accessed 17 February 2023].

[9] D. D. Ruscio, SUBSPACE SYSTEM IDENTIFICATION Theory and applications Lecture notes, Porsgrunn: Telemark University College, 2014.

[10] D. D. Ruscio, OPTIMAL MODEL BASED CONTROL: System Analysis and Design, Porsgrunn: Telemark University College, 2022.

[11] D. D. Ruscio, MODEL PREDICTIVE CONTROL and optimization, Porsgrunn: University of South-Eastern Norwa, 2019.

[12] "Climate and Average Weather Year Round in North Sea," Weather Spark, [Online]. Available: https://weatherspark.com/y/25440/Average-Weather-in-North-Sea-New-York-United-States-Year-Round. [Accessed 14 March 2023].

[13] D. D. Ruscio, "On Tuning PI Controllers for Integrating Plus Time Delay Systems," *Modeling, Identification and Control,* vol. 31, no. 4, pp. 145-164, 2010.

Appendices

Appendix A MATLAB Scripts: See scripts.zip file