ES155 Course Project

An Automated Naval Steering System

Arba Shkreli, Ezenia Diaz-Lembert, Eric Shi, Matthias Fischer

December 15, 2021

1. Introduction

The approximate four thousand sailboats sold in 2020 in the United States alone stresses the importance of accurate direction and functionality of such sailboats distributed across the country. In the same year, The Coast Guard tracked a 26.3% increase in boating accidents (from 4,168 in 2019 to 5,265 in 2020) [1] [2]. Be it a result attributed to the COVID-19 pandemic or other external factors, the increasing number of sailboat owners calls for attention toward potential root causes of accidents that can make an enjoyable ride into a nightmare. One of such factors that can lead to failures is a lack of reliable velocity control.

The ability for a sailboat to reach a specified velocity quickly (and remain at that velocity) can be instrumental in regulating marine traffic and preventing accidents. The aim of this paper is to study how to design a control system so that this objective can be achieved. That is, given a target velocity, the boat should be able to reach it in a short period of time, with little overshoot and oscillation. In other words, this is a desired behavior of the system we are designing because overshoot and oscillation in velocity are potentially dangerous in different sailing situations.

The system is meant to control motion in two dimensions (specified by the x- and y- directions), where the actuation mechanism is a propeller force that can be broken into x and y components. We assume that we are able to control the force of the propeller directly, in both directions. Our model is not concerned with how the force is achieved in both directions, as that would require more information that we do not have, and pursuing it is beyond the scope of this paper, as we are exploring a general velocity control system.

The dynamical system that we are choosing to study models the acceleration of the boat due to the force applied by the propeller, and the drag force from the water which opposes the motion of the boat. We discuss later on how the model can be augmented to consider the effect of water currents.

The actual implementation of the control system designed here needs information on how the boat is propelled and how to control the propulsion in two dimensions. That is, a rudder would have to be introduced, and modeling of how the propeller force can be broken into components based on the rudder would have to be done. As a result, we anticipate that the boat will have a single propeller and rudder to physically implement our control system, hopefully limiting the economic costs of implementation. However, as we have stated, there are many unknowns with regard to what kind of propulsion and steering system is needed to provide the necessary actuation that the controller requests. Nevertheless, our control goal is to reach a given reference velocity, and while we seek to minimize costs of carrying it out, we also do not want to compromise on the increased safety of traveling on the boats.

2. System modeling

As outlined above, this project aims to generate a simple model that can bring a ship or other naval vehicle to maintain a certain velocity, minimizing the error between the desired position of the ship and the actual position of the ship.
To model the system physics, the boat is assumed to be constrained to move in a plane, so in the x- and y-direction. We will ignore rotation of the boat to further simplify the dynamics. Additionally, we will model the boat velocities directly, such that we have:
$q = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$.
This choice was made, because including the absolute positions x and y as variables would mean that equilibrium points only occur for zero velocity. However, linearizations of these equilibrium points were not meaningful. Using velocities as our state space variables means we have equilibrium points at certain velocities that we can linearize around.

The boat will have a propeller system, which can exert a force $F_{p,x}$ in the x-direction and a force $F_{p,y}$ in the y-direction. The assumption here is that both propeller forces can be controlled entirely independently, and act only directly in the x- and y-directions.
As the boat moves through the water, it experiences a drag force. We will ignore viscous fluid effects here, and assume that the drag can be modeled using the drag equation shown below, where ρ denotes the density of water, A_x and A_y are the frontal areas of the ship in the x-direction and y-direction, and $C_{d,x}$ and $C_{d,y}$ are the drag coefficients. We will assume that the ship is shaped as a sphere with radius 1m, and a mass of 250 kg. This means we have $A_x = A_y = 2\pi r^2$ and $C_{d,x} = C_{d,y} = 0.5$ [5]. The drag can then be calculated as:

$$F_{d,x} = \frac{1}{2}\rho A_x C_{d,x}(v_x)^2 \cdot \operatorname{sign}(v_x)$$
$$F_{d,y} = \frac{1}{2}\rho A_y C_{d,y}(v_y)^2 \cdot \operatorname{sign}(v_y)$$.

Here, the final coefficient serves to ensure the drag is acting in the correct direction. We are assuming that drag in x and y can be modeled independently.

We can also account for currents if desired, by setting:
$v_x = v_{x,real} - v_{current,\,x}$
$v_y = v_{y,real} - v_{current,\,y}$.
For conciseness we will not include the term for the current velocity in our further considerations.

Using the assumptions above we can write the system in the standard state-space form, as below:

$$q = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$
$$\dot{v_x} = \frac{1}{M}(-F_{d,x} + F_{prop,x}) = \frac{1}{M}\left(-\frac{\rho}{2}A_x c_{d,x} v_x^2 \operatorname{sign}(v_x) + F_{prop,x}\right)$$
$$\dot{v_y} = \frac{1}{M}(-F_{d,y} + F_{prop,y}) = \frac{1}{M}\left(-\frac{\rho}{2}A_y c_{d,y} v_y^2 \operatorname{sign}(v_y) + F_{prop,y}\right)$$.

We have the control input $u_1 = F_{prop,x}$ and $u_2 = F_{prop,y}$.
The output is the velocity of the boat:

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot q$$

.

If this model is solved successfully, we can determine an F_p that drives the boat to the velocity given as a reference. It may also be possible to have a variable reference, such that the boat follows a certain trajectory. The key assumptions made include the constraint to two degrees of freedom with no rotation, and drag calculation, which ignores viscous effects and assumes that drag can be calculated with the drag equation. We are also making substantial assumptions about the direction in which the propeller force can act. Even though the model makes these simplifying assumptions, it is expected to be capable of achieving the goals set out in the introduction and at the beginning of this chapter.

3. Simulation

If we now set up a model of the system, we can illustrate the position of the equilibrium points of the system. It will only have one equilibrium point. If we set the propeller forces to certain values, the boat will accelerate until the propeller force equals the drag force in both directions. Each equilibrium point is stable, if the boat somehow gains more speed resulting in more drag than the propeller can provide, the boat will slow down, until the speed is again such that propeller and drag force cancel. The same applies to slow downs, the boat will speed up again until it reaches the equilibrium velocity. There are also no regions of instability, for any initial condition, the equilibrium point will be reached. Shown below are phase diagrams with trajectory plots for different propeller forces, and initial conditions at zero velocity. This confirms the considerations above.

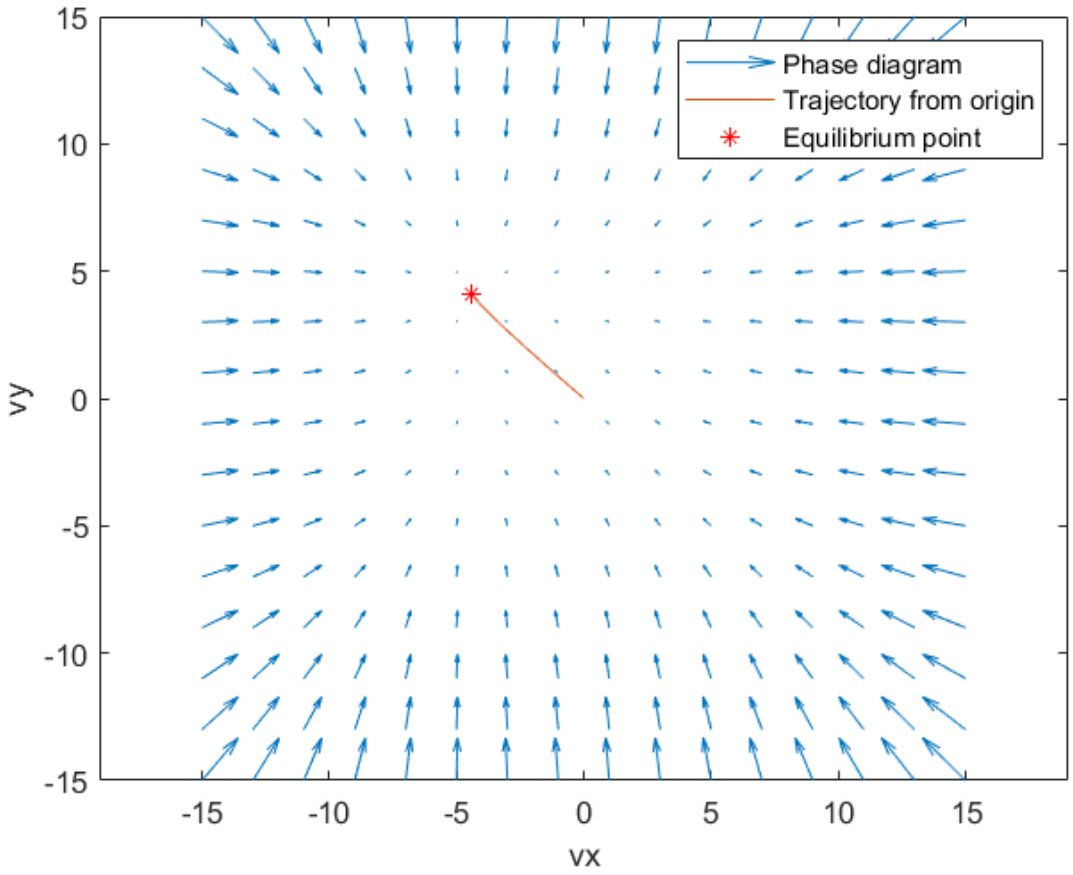


Figure 1: Phase diagram and trajectory for zero initial velocity and $F_{prop,x} = -15000\ N,\ F_{prop,y} = 13000\ N.$

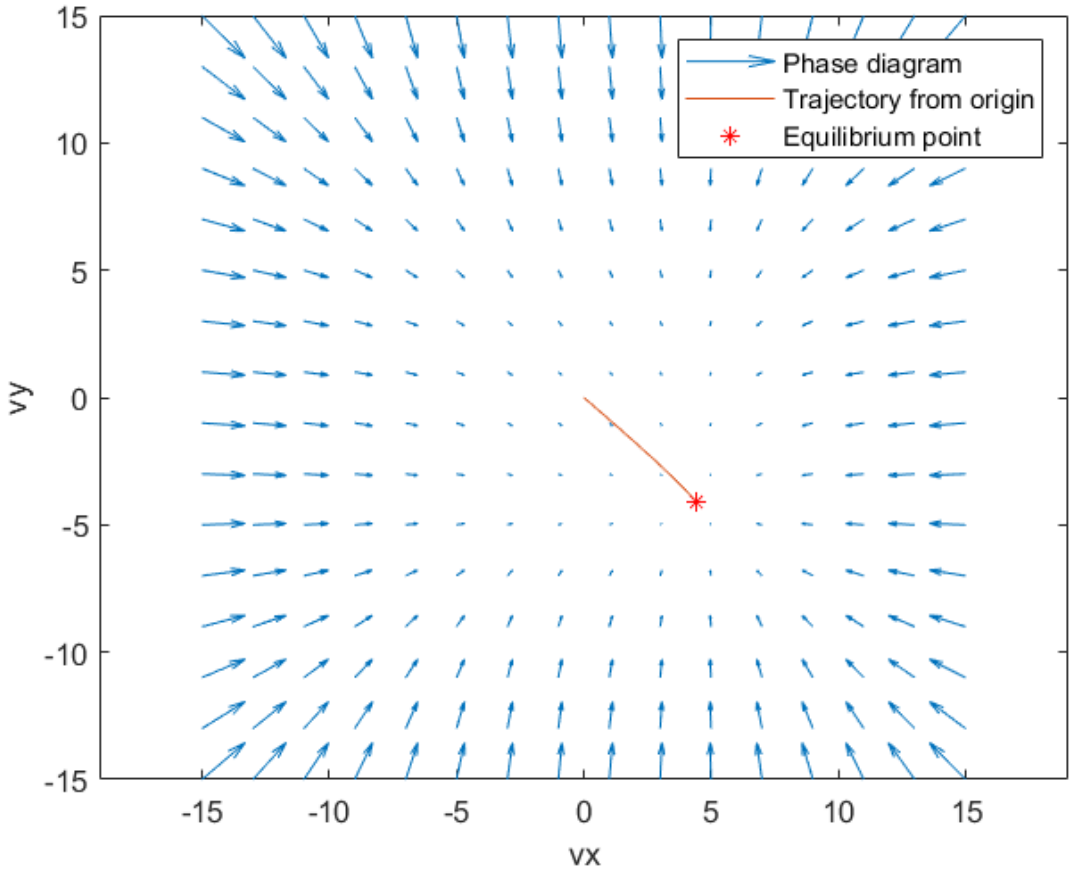


Figure 2: Phase diagram and trajectory for zero initial velocity and $F_{prop,x} = 15000\ N,\ F_{prop,y} = -13000\ N.$

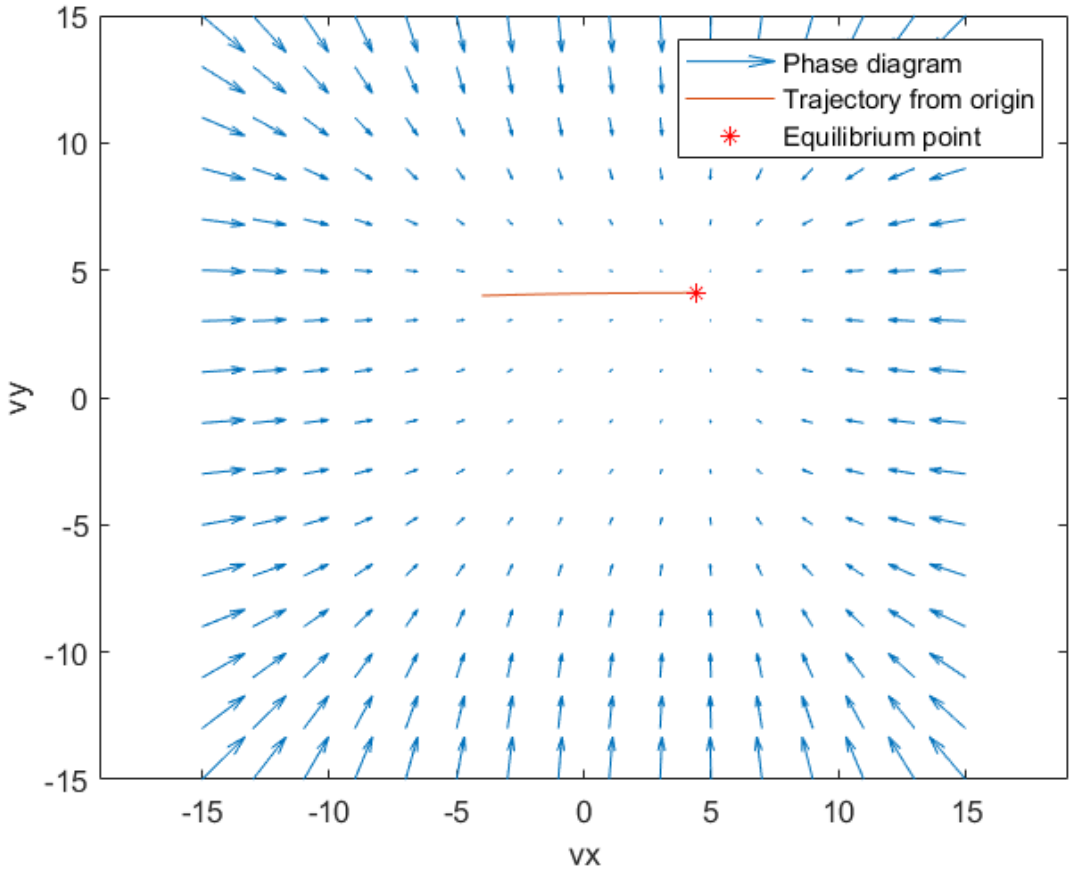


Figure 3: Phase diagram for initial $v_x < 0 \; and \; v_y > 0$ *and* $F_{prop,x} = 15000\ N,\ F_{prop,y} = 13000\ N.$

4) Linearization

At this stage, we want to linearize the nonlinear system around its equilibrium points so that we can not only evaluate its stability, but also in later stages use it to design an estimator and controller.

Recalling that the dynamics for our system are as follows:

$$q = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

$$\dot{v_x} = \frac{1}{M}(-F_{d,x} + F_{prop,x}) = \frac{1}{M}\left(-\frac{\rho}{2}A_x c_{d,x} v_x^2 \text{sign}(v_x) + F_{prop,x}\right)$$

$$\dot{v_y} = \frac{1}{M}(-F_{d,y} + F_{prop,y}) = \frac{1}{M}\left(-\frac{\rho}{2}A_y c_{d,y} v_y^2 \text{sign}(v_y) + F_{prop,y}\right)$$

We need to find the equilibrium points of our system for a nonzero propeller force. We do this by considering the cases when the propeller force is positive and negative (in the x and y direction, respectively).

$$\dot{q} = 0 \implies \dot{v_x} = 0, \dot{v_y} = 0$$

In the x-direction, if $F_{prop,x} > 0$, we know that the equilibrium x velocity will be positive as well (by sheer physical considerations). As a result, $sign(v_{x,eq}) = 1$.

$$F_{prop,x} > 0$$

$$v_{x,eq} = \sqrt{\frac{2F_{prop,x}}{\rho A_x c_{d,x}}}$$

We choose only the positive solution from this square root, since as we have stated, only a positive equilibrium velocity for a positive propeller force makes sense in our case.

We undertake a similar process to obtain the equilibrium velocity when the propeller force is negative. This time, the equilibrium velocity will be negative from once again thinking about our physical system. Thus, we have:

$$F_{prop,x} < 0 \implies v_{x,eq} < 0$$

$$F_{prop,x} = -\frac{\rho}{2}A_x c_{d,x} v_x^2$$

$$v_{x,eq} = -\sqrt{\frac{-2F_{prop,x}}{\rho A_x c_{d,x}}}$$

We can more generally express our equilibrium x velocity as

$$v_{x,eq} = \text{sign}(F_{prop,x})\sqrt{\frac{2|F_{prop,x}|}{\rho A_x c_{d,x}}}$$

Because we are assuming that the movement in the x and y direction are independent of each other, we analogously express the equilibrium y velocity as

$$v_{y,eq} = \text{sign}(F_{prop,y})\sqrt{\frac{2|F_{prop,y}|}{\rho A_y c_{d,y}}}$$

Now, we must take the Jacobian of our system and evaluate at the equilibrium points.

To find the ‘A’ matrix:

$$\begin{bmatrix} \frac{\partial}{\partial v_x}\left[\frac{1}{M}\left(-\frac{\rho}{2}A_x c_{d,x} v_x^2 \text{sign}(v_x) + F_{prop,x}\right)\right] & \frac{\partial}{\partial v_y}\left[\frac{1}{M}\left(-\frac{\rho}{2}A_x c_{d,x} v_x^2 \text{sign}(v_x) + F_{prop,x}\right)\right] \\ \frac{\partial}{\partial v_x}\left[\frac{1}{M}\left(-\frac{\rho}{2}A_y c_{d,y} v_y^2 \text{sign}(v_y) + F_{prop,y}\right)\right] & \frac{\partial}{\partial v_y}\left[\frac{1}{M}\left(-\frac{\rho}{2}A_y c_{d,y} v_y^2 \text{sign}(v_y) + F_{prop,y}\right)\right] \end{bmatrix}$$

We will have to make sign considerations based on the propeller force's sign when evaluating these partial derivatives, as done below:

$$F_{prop,x} > 0 \implies \text{sign}(v_x) = 1$$

$$v_{x,eq} = \sqrt{\frac{2F_{prop,x}}{\rho A_x c_{d,x}}}$$

$$\frac{\partial \dot{v_x}}{\partial v_x} = \frac{1}{M}(-\rho A_x c_{d,x} v_{x,eq})$$

$$\frac{\partial \dot{v_x}}{\partial v_y} = 0$$

$$F_{prop,x} < 0 \implies \text{sign}(v_x) = -1$$

$$v_{x,eq} = -\sqrt{\frac{-2F_{prop,x}}{\rho A_x c_{d,x}}}$$

$$\frac{\partial \dot{v_x}}{\partial v_x} = \frac{1}{M}(\rho A_x c_{d,x} v_{x,eq})$$

$$\frac{\partial \dot{v_x}}{\partial v_y} = 0$$

The calculations are analogous for v_y. We find that the linearization effectively removes the sign-dependency of the nonlinear system, thus we get the following 'A' matrix which works for input forces of any sign:

$$\begin{bmatrix} \frac{1}{M}\left(-\rho A_x c_{d,x}\sqrt{\frac{2|F_{prop,x}|}{\rho A_x c_{d,x}}}\right) & 0 \\ 0 & \frac{1}{M}\left(-\rho A_y c_{d,y}\sqrt{\frac{2|F_{prop,y}|}{\rho A_y c_{d,y}}}\right) \end{bmatrix}$$

The calculations for the 'B' matrix are straightforward

$$\begin{bmatrix} \frac{\partial \dot{v_x}}{\partial F_{prop,x}} & \frac{\partial \dot{v_x}}{\partial F_{prop,y}} \\ \frac{\partial \dot{v_y}}{\partial F_{prop,x}} & \frac{\partial \dot{v_y}}{\partial F_{prop,y}} \end{bmatrix} = \begin{bmatrix} \frac{1}{M} & 0 \\ 0 & \frac{1}{M} \end{bmatrix}$$

For the 'C' matrix, we are interested in outputting both the x- and y- velocities, so it will be:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The 'D' matrix is simply the zero matrix, because the output is not directly affected by the input.

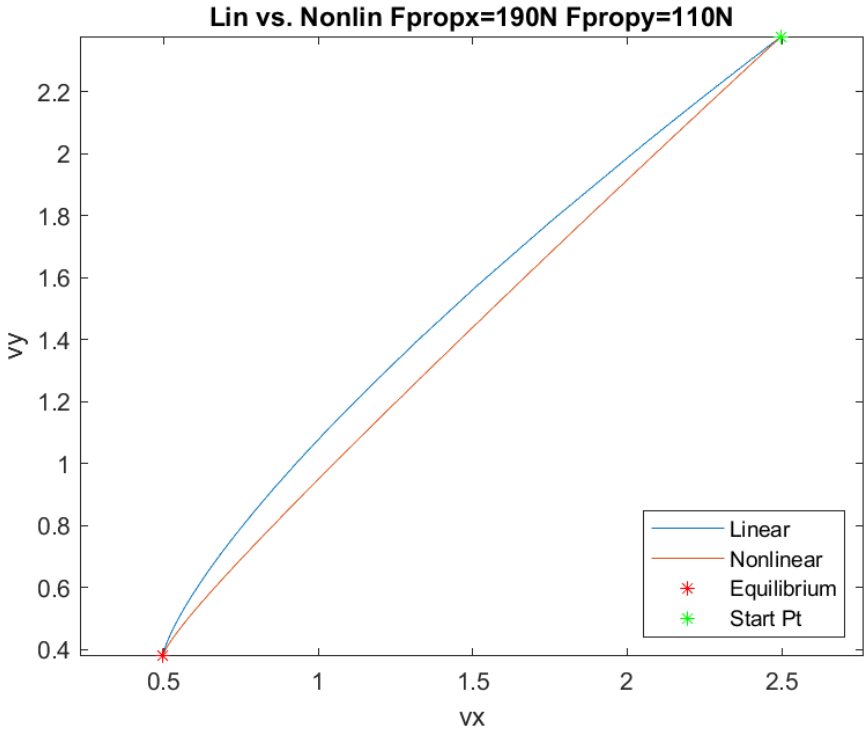
If we plug in various propeller constant propeller forces (positive and negative) and system parameters, and calculate the eigenvalues using MATLAB, we'll always find that all eigenvalues of the linearization for each set of input forces have negative real parts, indicating that the system is stable at equilibrium (see "Milestone4Final.mlx"), and that stability is not affected by

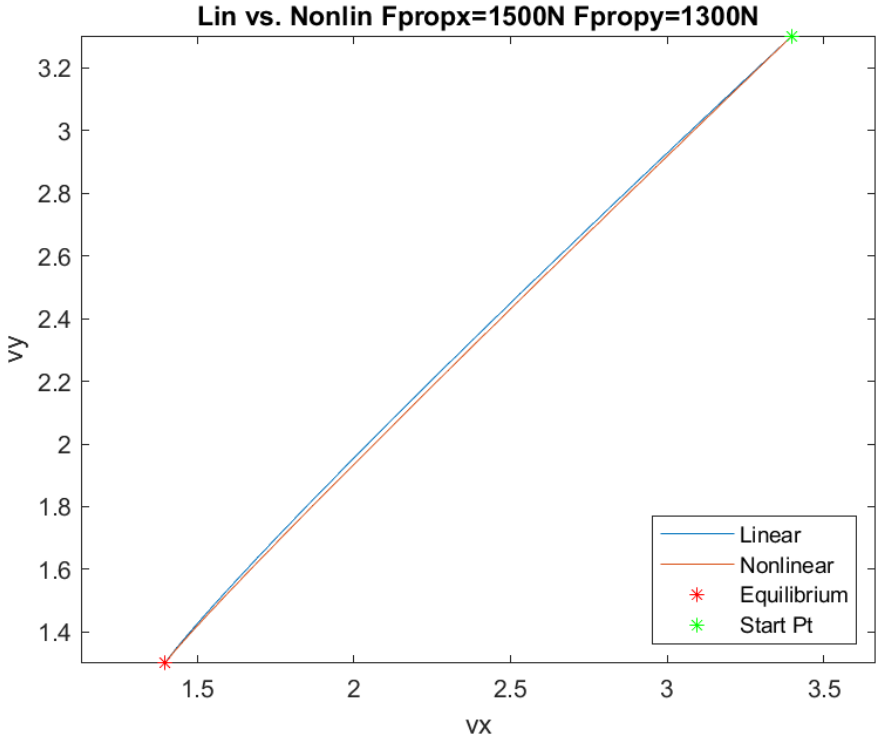
the system parameters. Physically, this makes sense because we expect that after some time that the boat has had an applied propeller force, against the force of drag, that it would settle to a constant velocity. Within reasonable physical limits (for example, the boat does not tip over) of the system this behavior should not change depending on the magnitude or sign of the propeller forces.

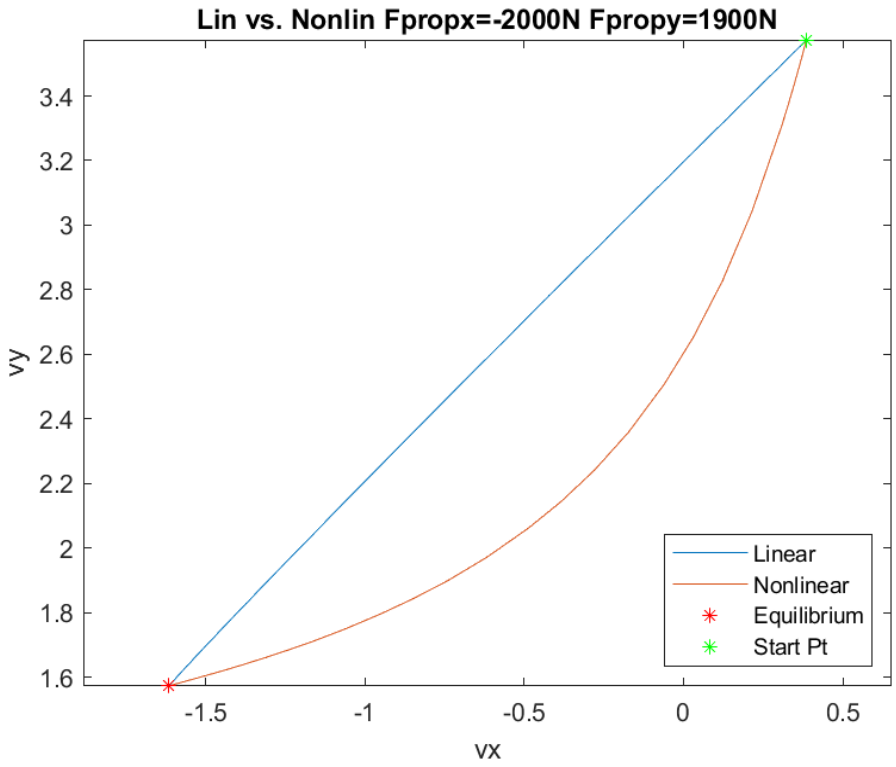
We are also interested in seeing how closely the linearization captures the behavior of the nonlinear system. For this, we will look at 5 different scenarios, each with their own set of initial values and constant input propeller forces, which yield different A, B, C, D matrices. The calculations of these matrices and their eigenvalues is done in the MATLAB file. Both the linearization and the nonlinear system will start from the same initial velocity which is a small shift from equilibrium (i.e. the “Start Point” on the graphs), and evolve separately. We seek to compare the differences in their evolution.

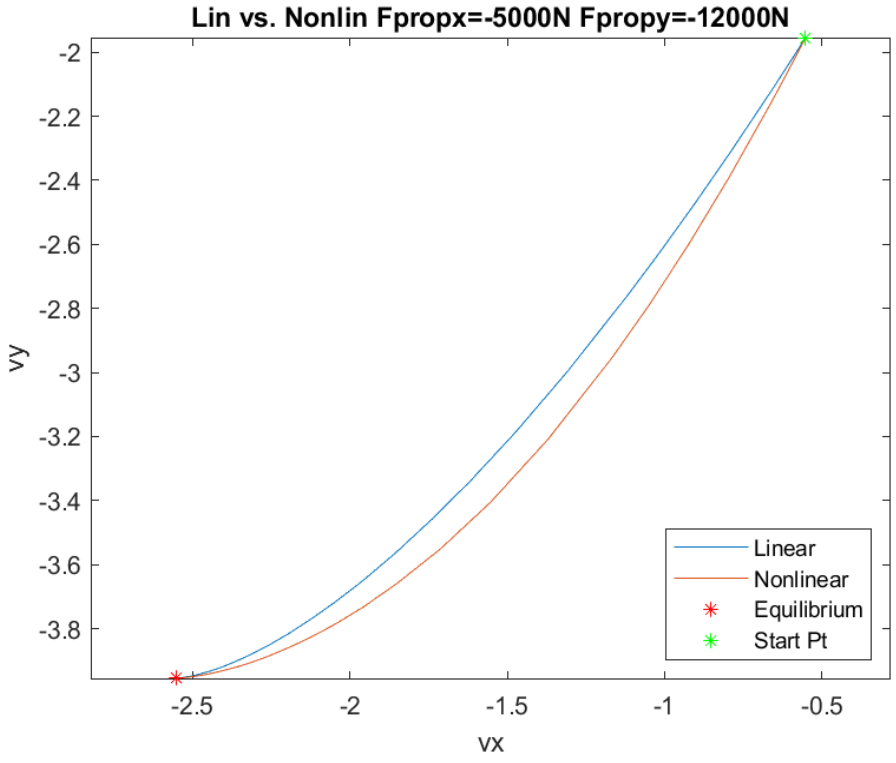
By “small shift,” we mean that the starting x- and y- velocities will both be respectively 2 m/s away from the calculated equilibrium velocities which vary based on arbitrary propeller forces around which the system has been linearized. Depending on the input forces, this “small” shift might actually be rather significant, but we are interested to see how far we can push the linearization, as our controller which we hope to use on the nonlinear system will be derived from it.

In the following set of graphs, we plot the linear and nonlinear systems together with the same set of constant input forces. The magnitude of the forces is different per graph, in fact increasing:


Lin vs. Nonlin Fpropx=190N Fpropy=110N
vy
vx
Linear
Nonlinear
Equilibrium
Start Pt


Lin vs. Nonlin Fpropx=1500N Fpropy=1300N
vy
vx
Linear
Nonlinear
Equilibrium
Start Pt


Lin vs. Nonlin Fpropx=-2000N Fpropy=1900N
3.4
3.2
3
2.8
2.6
2.4
2.2
2
1.8
1.6
vy
-1.5
-1
-0.5
0
0.5
vx
Linear
Nonlinear
Equilibrium
Start Pt


Lin vs. Nonlin Fpropx=-5000N Fpropy=-12000N
-2
-2.2
-2.4
-2.6
-2.8
-3
-3.2
-3.4
-3.6
-3.8
vy
-2.5
-2
-1.5
-1
-0.5
vx
Linear
Nonlinear
Equilibrium
Start Pt

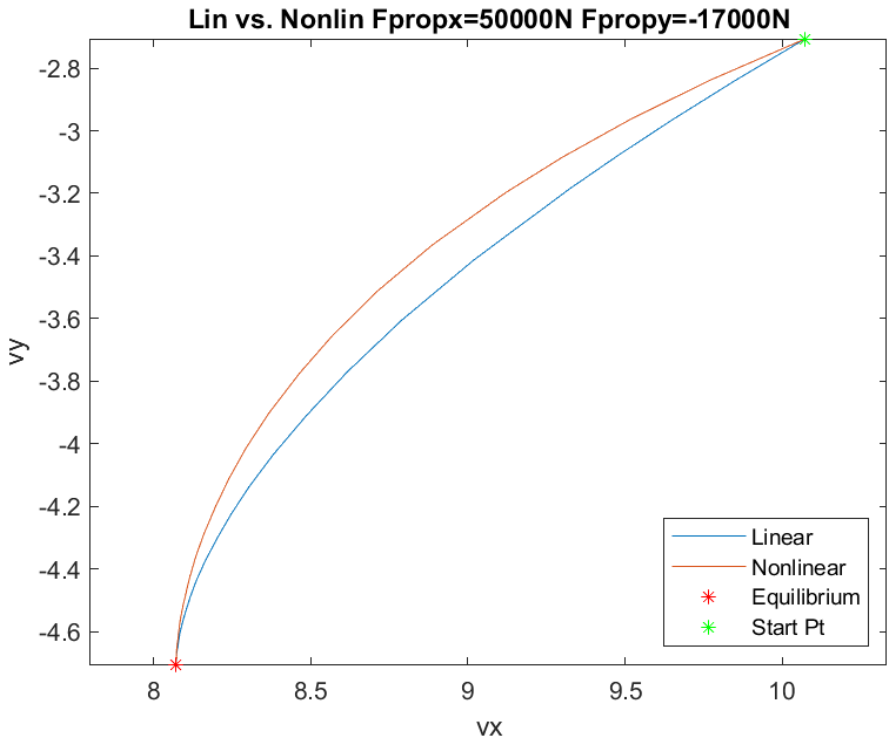


Figure 4: Linearized responses for different initial propeller forces

With different input forces, qualitatively we can say that the linearization does a fairly good job of capturing the behavior of the nonlinear system. The case in which it noticeably fares worse is with the set of input forces $F_{prop,x} = -2000N$ and $F_{prop,y} = 1900N$. It will be interesting to see what effect the differences that do exist in behavior have on the performance of the linear controller we will design.

5. Controller

Having a linearization of our system, we can now employ the reachability test to determine whether the system is reachable (or 'controllable'). Thus, we compute the reachability matrix $[B \quad BA]$. We find that this matrix has rank 2, the number of dimensions of our state vector, passing the reachability test. Intuitively this makes sense because our actuator directly affects both the x- and y- velocities. We may hence proceed with controller design.

As we have previously discussed, our control objective is to have the boat reach a certain reference x- and y- velocity. For this, we are interested in designing a controller of the form $u = -Kx + k_r r$, where x is the state of the system (which in our case we've actually called q, where

$r = \begin{bmatrix} v_{x,\,desired} \\ v_{y,\,desired} \end{bmatrix}$.

Plugging in a controller of the form $u = -Kx + k_r r$ into our system results in the following state space model:

$$\tilde{x} = (A - BK)x + Bk_r r$$
$$y = Cx$$

Including an estimator, the state space model will look like the following:

$$\frac{d}{dt}\begin{bmatrix} x \\ e \end{bmatrix} = \underbrace{\begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} x \\ e \end{bmatrix} + \begin{bmatrix} BK_r \\ 0 \end{bmatrix} r$$

$$y = Cx$$

Note that the estimator, which accounts for some expected noise in the system (where error $e = x - \tilde{x}$), is only applicable to the linear model and not the nonlinear model as the linear model contains the A matrix a Luenberger estimator requires.

Since we are tracking and driving the velocity in both the x and y directions, our new A matrix $\tilde{A}$ for $\tilde{x}$ will consist of four 2x2 matrices, where K is equal to some 2x2 matrix dependent on the eigenvalues of A.

$$K = \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix}$$

Changing the eigenvalues desired in this matrix allows one to create different controllers for the system to observe system dynamics in both the linear and nonlinear model. To find K, the 'place' function in MATLAB is used, where $K = place(A, B, eig(A))$. The controller is built from the linearization around fixed propeller forces $F_{propx} = 100\,N$, $F_{propy} = 100\,N$ and a set of given eigenvalues. We will test different eigenvalues for this matrix which correspond to different controllers and compare how the linear and nonlinear systems behave in terms of whether and how they reach a given reference of $v_x = -1.5\,m/s$, $v_y = -1.5\,m/s$. Both the linear and nonlinear systems have the same initial starting velocities, which are a constant shift from the equilibrium velocities by -5 m/s for x- and y- respectively.

Each of the following scenarios show the behavior of the linear vs. nonlinear system with a set of eigenvalues. Plotted are the x- and y- velocities over time, as well as the reference velocities (in this case the x- and y- references are the same).

Scenario 1:

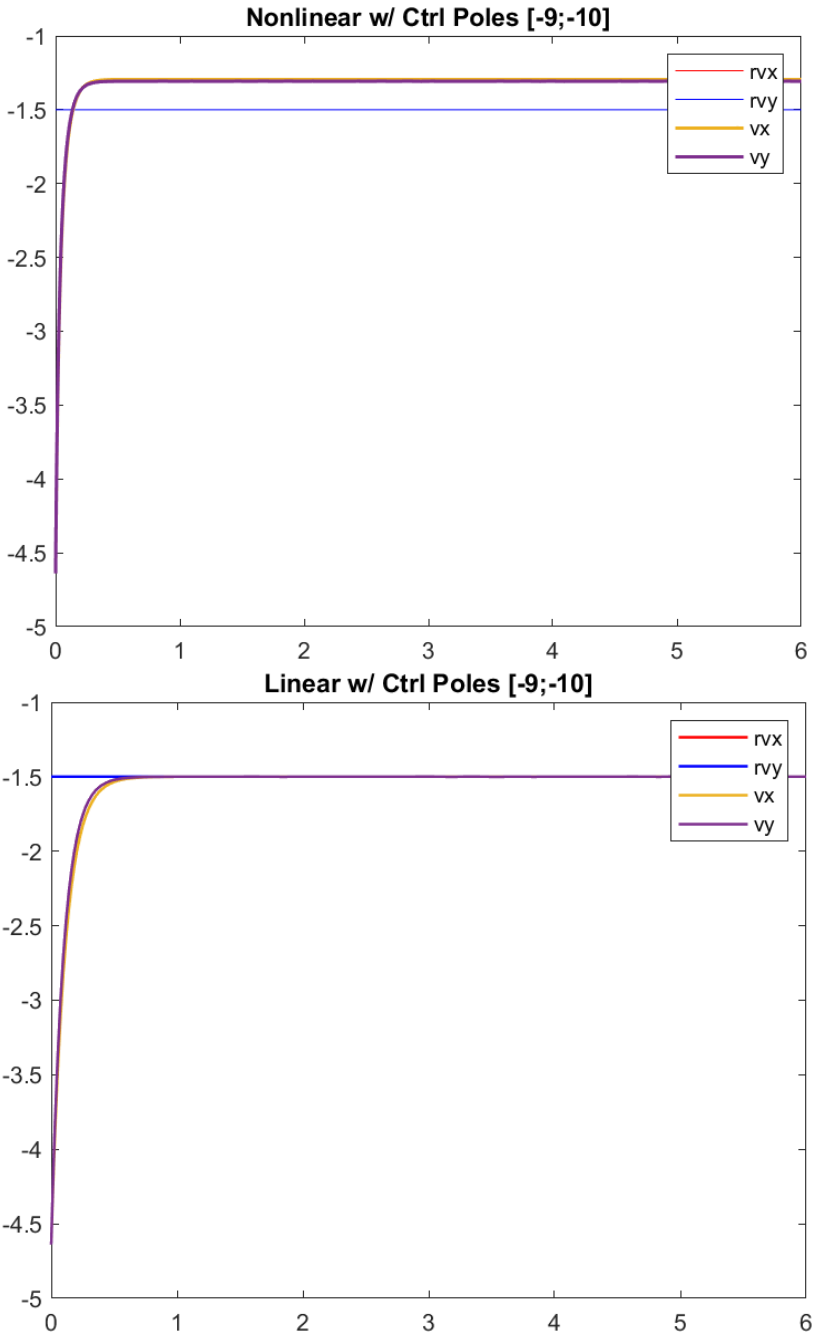


Figure 5: Controller response for poles [-9; -10]

We notice that the linear system reaches reference without overshoot and oscillation, and quickly. The nonlinear system does not reach the reference, with a steady-state error of about 20%, but does not pose any oscillatory behavior.

Scenario 2:

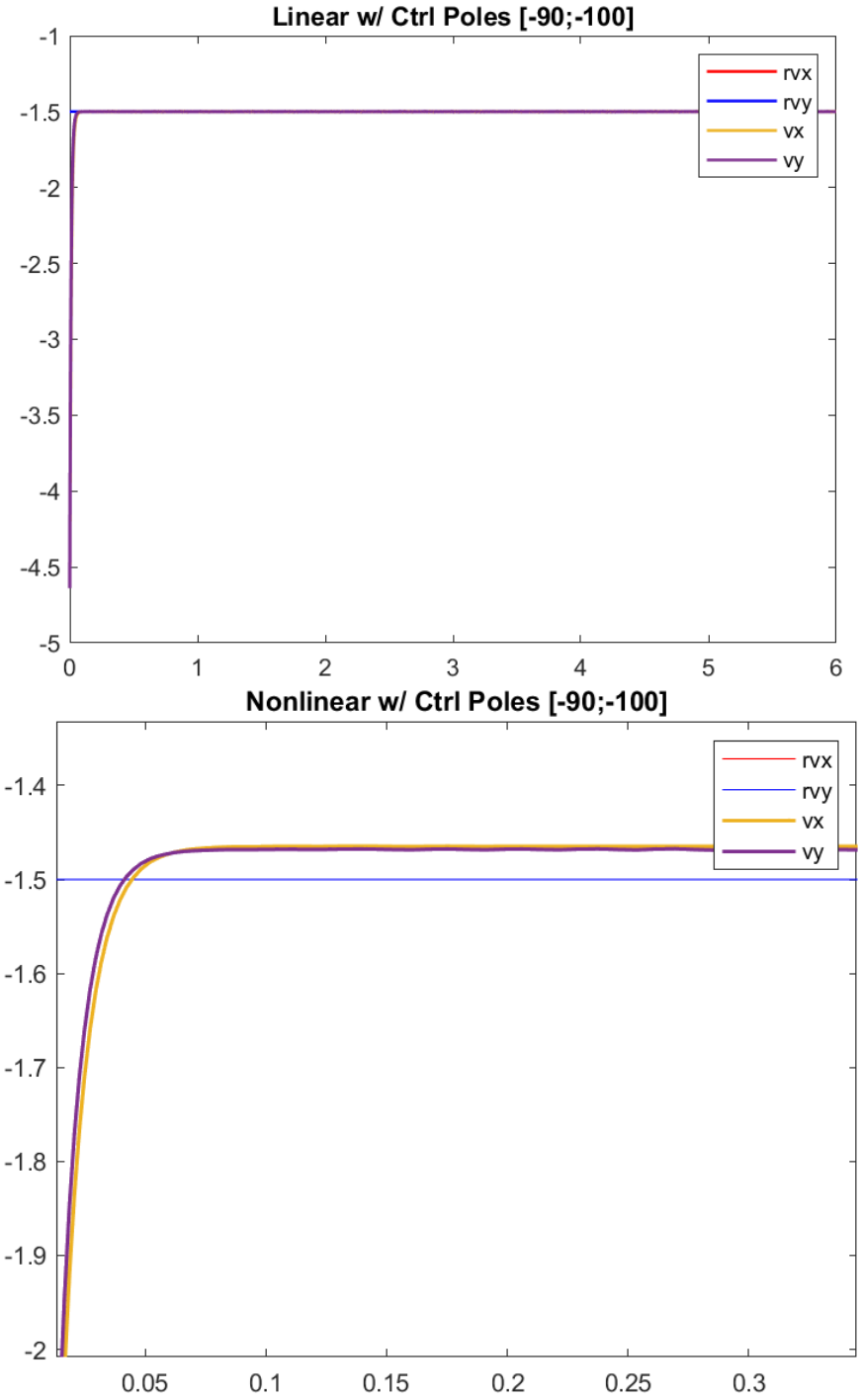


Figure 6: Controlled response for poles with lower negative real poles

As expected, more aggressive eigenvalues made the rise time smaller for both systems, and for the nonlinear system, reduced the steady-state error.

Scenario 3:

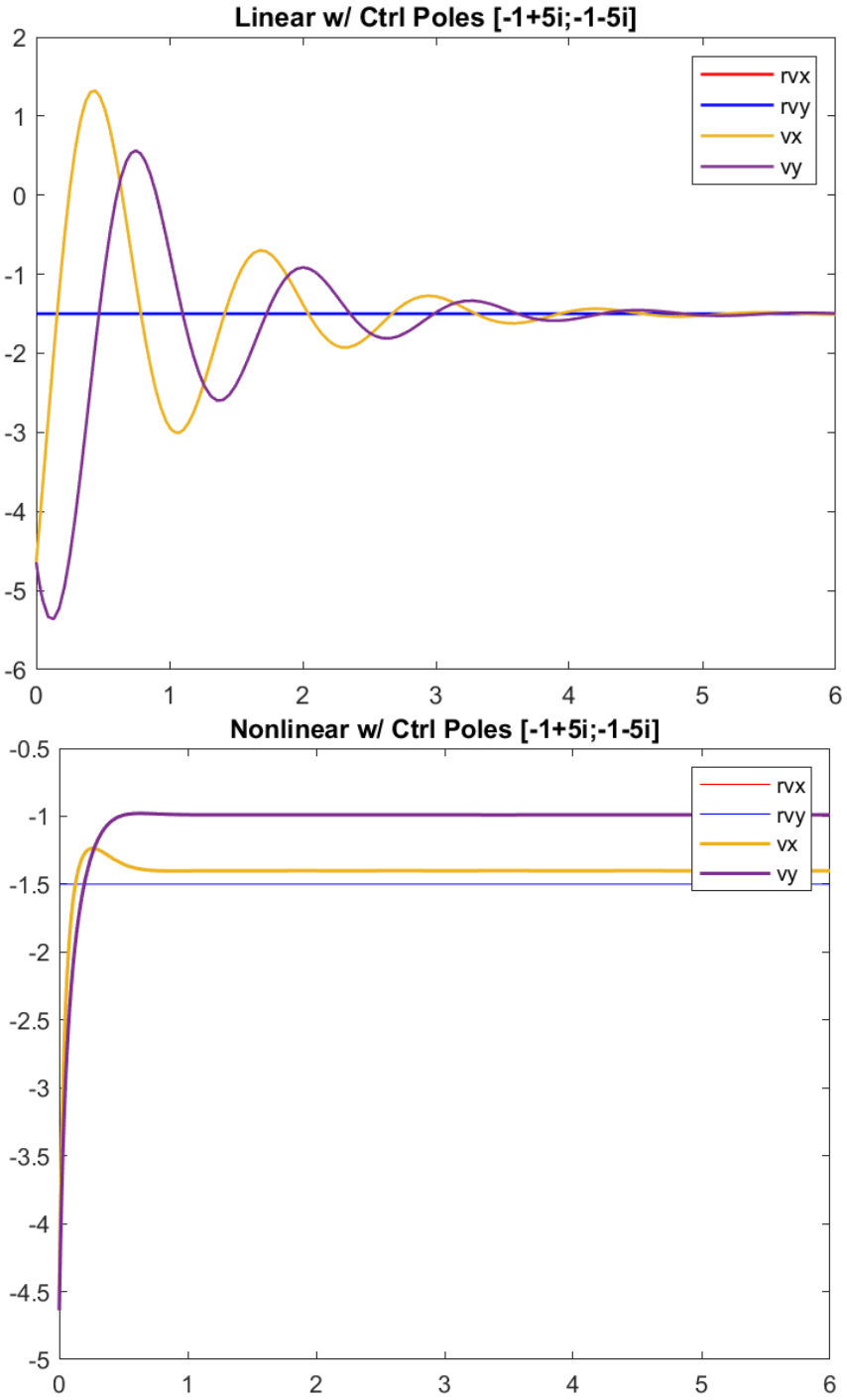


Figure 7: Controlled response with complex eigenvalues

The imaginary part of the eigenvalues brought noticeable oscillatory behavior for the linear system, since it did not have a large enough negative real part to dampen it quickly. There is also overshoot in the linear system, but it does reach the reference value, as expected. The nonlinear system does not have the oscillation (not a significant one, at least) of its linear

counterpart, but once again it fails to reach the reference value, which for the y- velocity the steady state error is very significant (50%). We tried increasing in magnitude the negative real part of the eigenvalues in the next scenario.

Scenario 4:

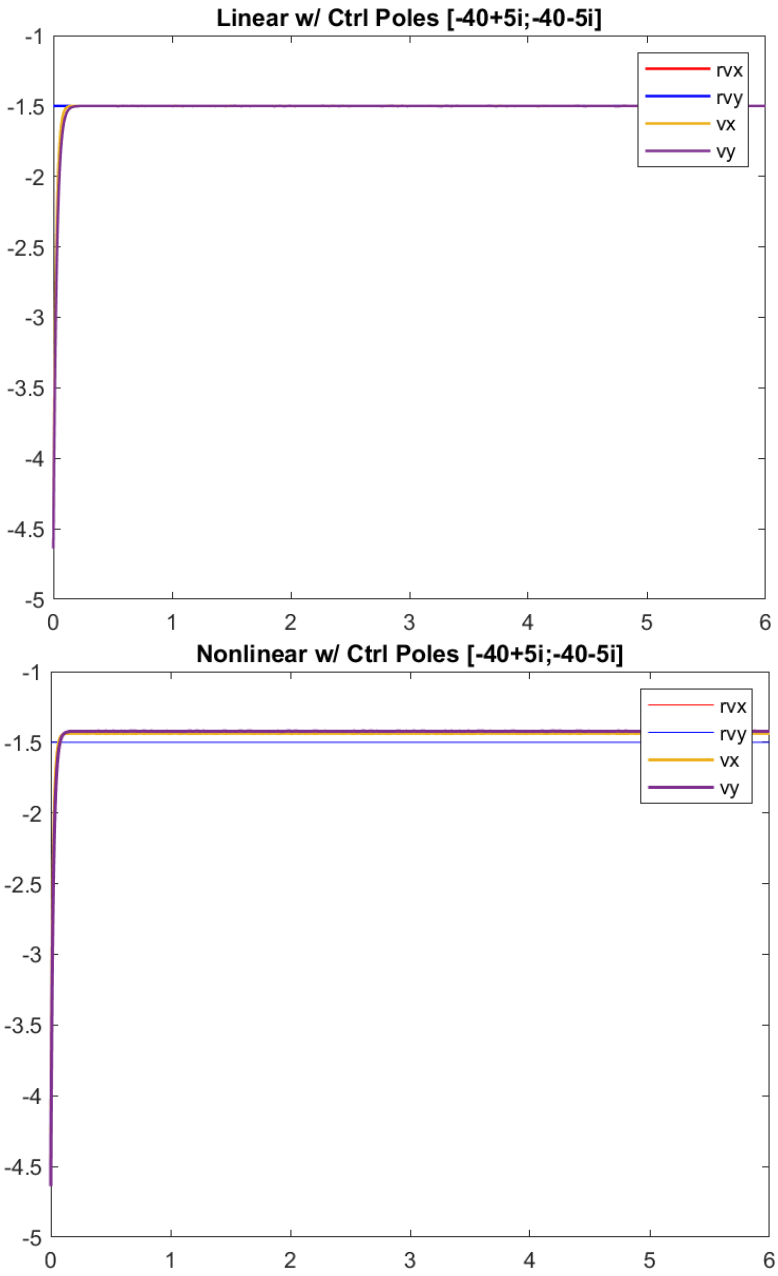


Figure 8: Controlled response with complex poles and lower negative real parts

As expected, the stronger real part did dampen both the linear and nonlinear system more quickly, and reduced steady-state error in the latter, and there is no longer the noticeable discrepancy between the x- and y- velocities.

Scenario 5:

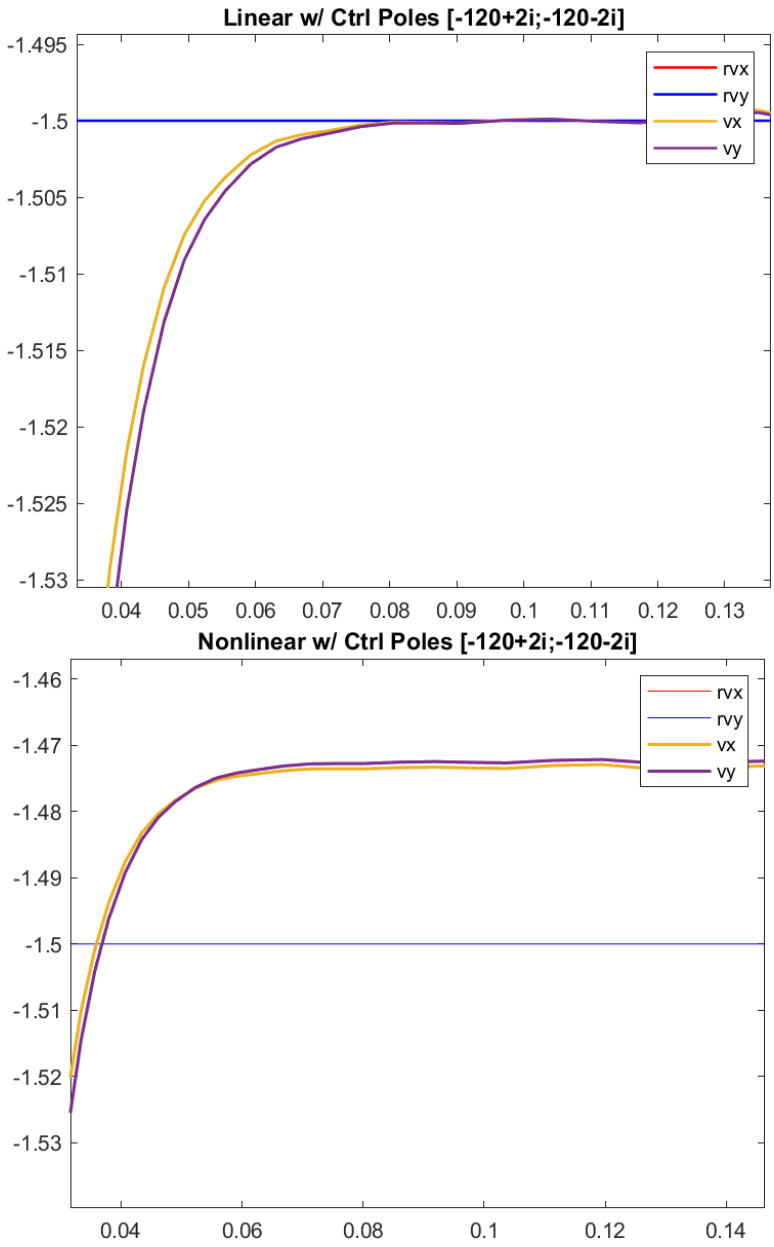


Figure 9: Controlled response with low negative real parts, and reduced imaginary magnitude

In this scenario we combine the findings of previous experiments, and implement an increase in the magnitude of the negative real part of the eigenvalues, and decrease the imaginary part. As a result, the oscillation has decreased and the steady-state error is perhaps acceptably small (depending on the design specifications). Even if they were not, we could always make the negative real part even larger in magnitude, assuming that this is implementable in the physical system.

These experiments were done assuming that the system has access to the true x and y velocities, but in the next step our linear system will include a Luenberger estimator. Unfortunately, we could not simulate our nonlinear system with the same estimator, because the Luenberger estimator requires an “A” matrix of a linear system in order for it to be plugged into the dynamics successfully. Should we be equipped with nonlinear estimation methods in the future, we would be very interested to see the effect of them here.

6. Estimator

The next step will be to design an estimator for our system, since in reality our system will not be able to access the true x- and y- velocities, only estimates. Our output measurements will be v_x and v_y. First, we determine whether our system is even observable, since otherwise we will not be able to design an estimator.

The observability matrix is in our case $\begin{bmatrix} C \\ CA \end{bmatrix}$.
As we show in the attached code (“Estimator.mlx”), this matrix has rank 2, which is the number of dimensions of our state vector, passing the observability test. Thus, we may proceed with the design of the estimator.

We can design our estimator similar to the controller above. Our new state space model has the following form:

$$\dot{x} = Ax + Bu$$
$$y = Cx$$
$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$$
$$u = K\hat{x} + k_r r$$

We have $e = x - \hat{x}$, such that we can write:
$\dot{e} = \dot{x} - \dot{\hat{x}} = (A - LC)e$.

We can then write the full dynamics of x and e as follows:

$$\frac{d}{dt}\begin{bmatrix} x \\ e \end{bmatrix} = \begin{bmatrix} A - BK & B \\ 0 & A - LC \end{bmatrix}\begin{bmatrix} x \\ e \end{bmatrix} + \begin{bmatrix} BK_r \\ 0 \end{bmatrix} r$$

To ensure that we have a successful Luenberger observer, we design $(A - LC)$ to have only eigenvalues with negative real parts. We can do this using $place(A',\ C',\ [pole\ 1,\ pole\ 2])$ in Matlab.

We can simulate this system using ode45. As our initial values we might choose:

$$\begin{bmatrix} x \\ e \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0.3 \\ -0.2 \end{bmatrix}.$$

This means we have an initial velocity and an error between the actual system and the estimated system. As a reference value we will target $r = [-10, -10]$.

Plotted below is the response for a few combinations of estimator and controller eigenvalues. As we see, the system reaches the reference value quickly and accurately, unless we introduce positive poles. This is to be expected. We also see that negative poles of greater magnitude induce a sharper response. Therefore, we can improve transient performance by increasing the magnitude of the real parts of negative poles. While the steady state error is not shown below, this was found to be negligible, $<10^{-4}$ for the systems shown in Figure 4 and 5.

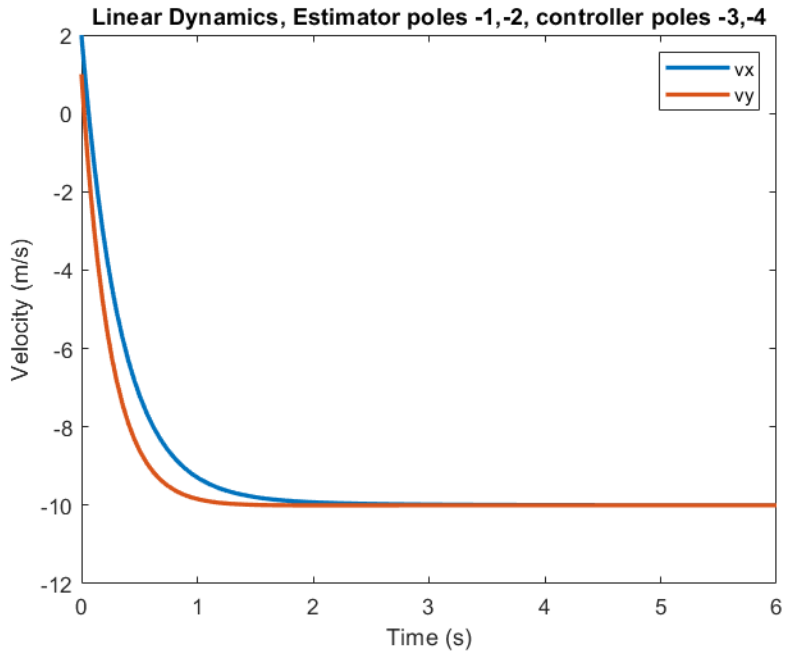


Figure 10: Estimator and controller system response for real eigenvalues of small magnitude, resulting in longer settling time.

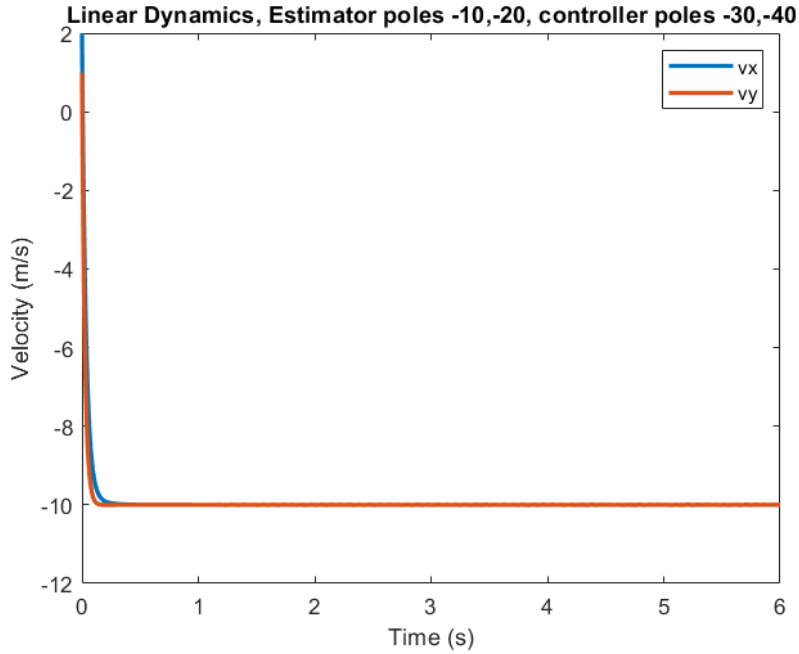


Figure 11: Estimator and controller response for poles with higher negative real magnitude, result in faster settling time.

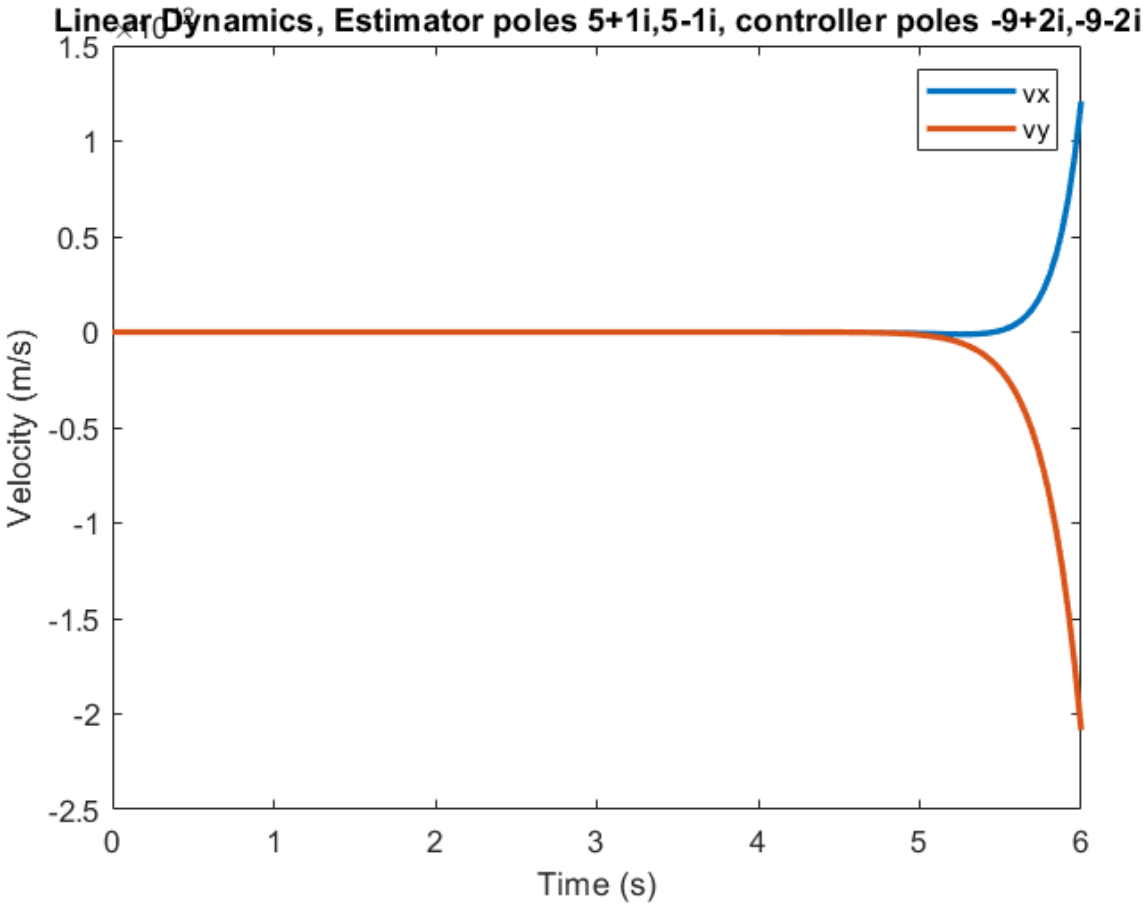


Figure 12: Positive poles result in system instabilities and should be avoided.

If we increase the magnitude of the imaginary part of the poles, this results in more unpredictable and unreliable performance, but the system is still stable. However, steady state error increases. The system in Figure 6 is still off by 4% after 6 seconds.

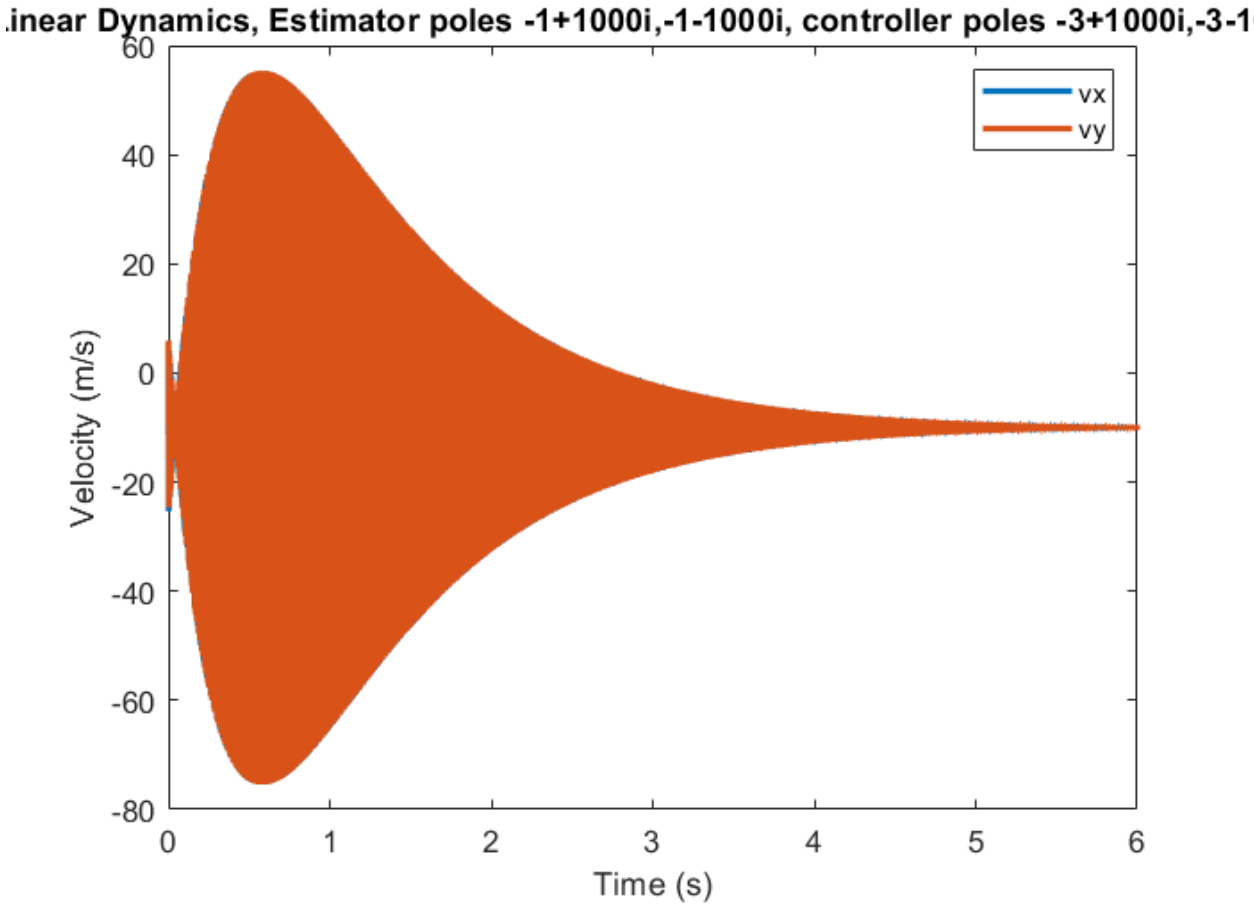


Figure 13: Poles with large imaginary components result in overshoot and undesirable transient behavior, but the system is still stable

Simulation of the nonlinear dynamics with the estimator was not possible, because it both calculating a stable observer and calculating the error dynamics involves a linearized A matrix.

7. Controller design

In this section we will develop and evaluate a set of engine controllers to ensure the boat can track a reference signal with high accuracy, despite noise and disturbances.

Step 1
To do so, we will first calculate the transfer function of our differential equations. To simplify the calculations we will use a single control input and a single observed output, v_x. We can create the block diagram for this system in Simulink, as shown below.

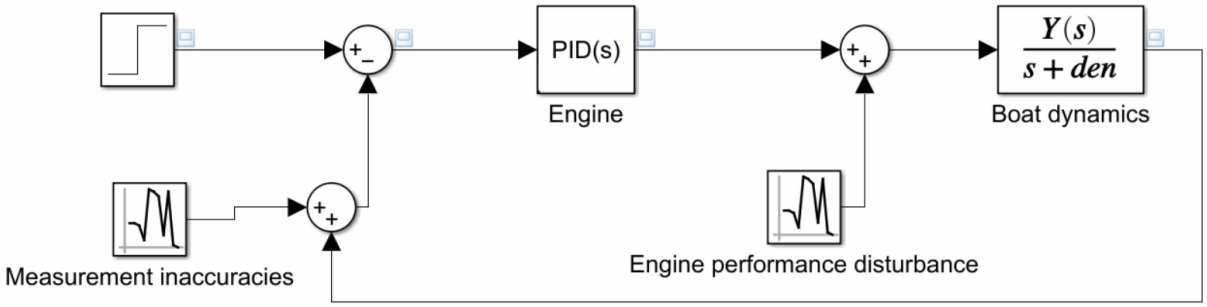


Figure 14: Block diagram of boat system in x-dimension only

We have a constant step input serving as the reference speed. This is compared to the measured current speed, to serve as an input into a PID controller. The PID controller output motor force is then combined with engine disturbances. These engine disturbances might result from time-varying flow complexities around the engine blades, or imperfections in the motor output speed. They could also result from current variabilities. This actual engine force is then inputted into a time-varying transfer function representing the boat. The transfer function is discussed and derived below. Next, the actual speed is combined with measurement noise, and combined with the reference signal.

Step 2
We can calculate the plant transfer function by taking the Laplace transform of the linearized system, reduced to a single control input:

$$\dot{v}_x = a_{11} v_x + b_1 u$$

$$\dot{v}_x = \frac{1}{M}\left(-\rho A_x c_{d,x}\sqrt{\frac{2|F_{prop,x}|}{\rho A_x c_{d,x}}}\right) \cdot v_x + \frac{1}{M}U)$$

$$V_x(s)s = \frac{1}{M}\left(-\rho A_x c_{d,x}\sqrt{\frac{2|F_{prop,x}|}{\rho A_x c_{d,x}}}\right) V_x(s) + \frac{1}{M}U_x(s)$$

$$P(s) = \frac{V_x(s)}{U_x(s)} = \frac{\frac{1}{M}}{\left(s + \frac{1}{M}\left(\rho A_x c_{d,x}\sqrt{\frac{2|F_{prop,x}|}{\rho A_x c_{d,x}}}\right)\right)}$$

$$q = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

$$\dot{v_x} = \frac{1}{M}(-F_{d,x} + F_{prop,x}) = \frac{1}{M}\left(-\frac{\rho}{2} A_x c_{d,x} v_x^2 \text{sign}(v_x) + F_{prop,x}\right)$$

$$\dot{v_y} = \frac{1}{M}(-F_{d,y} + F_{prop,y}) = \frac{1}{M}\left(-\frac{\rho}{2} A_y c_{d,y} v_y^2 \text{sign}(v_y) + F_{prop,y}\right)$$

If we consider the Bode plot of P(s) below, we can see that to get good tracking performance and low steady state error, our controller would need to add substantial gain to the system.

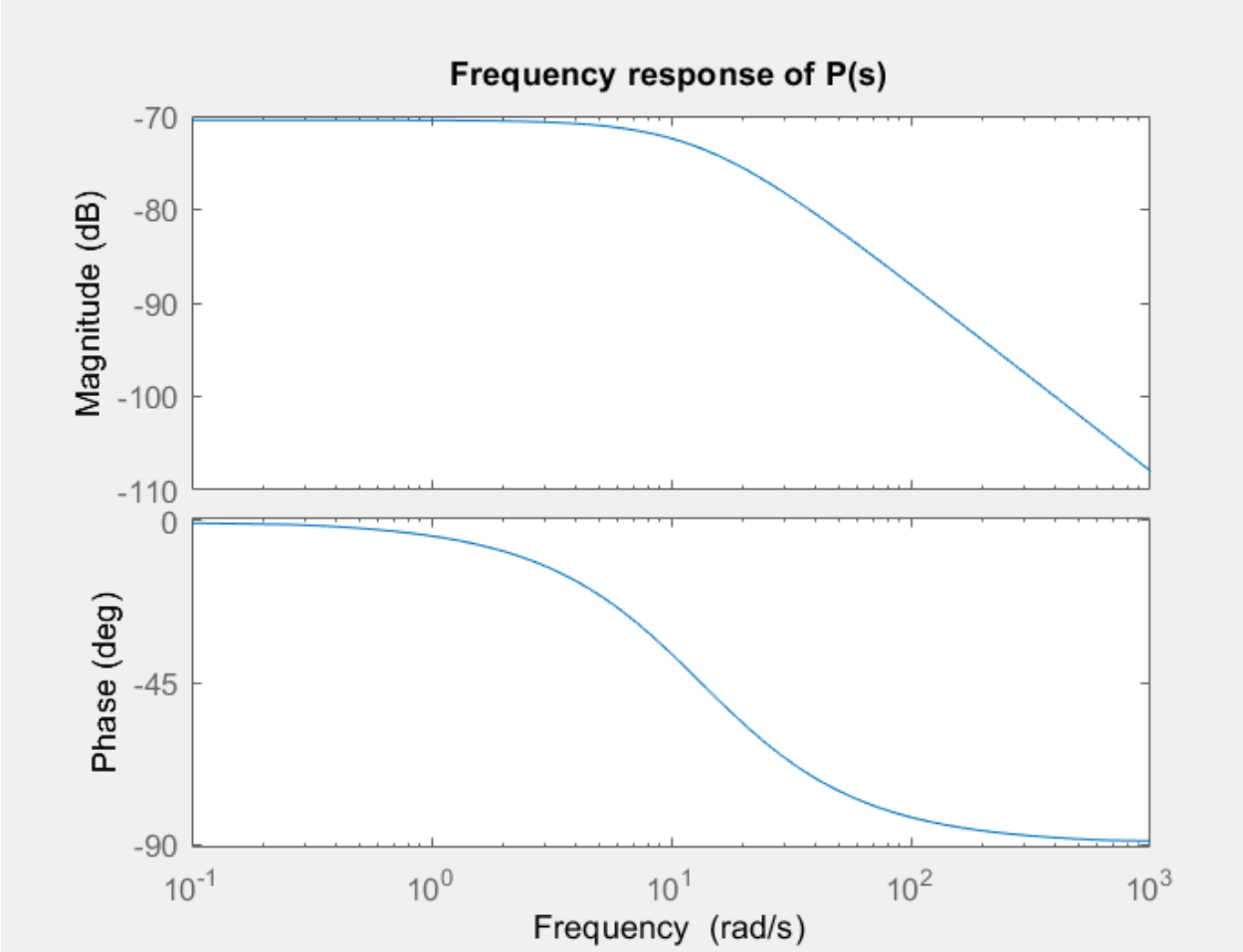


Figure 15: Bode plot of P(s)

Step 3

Next, we proceeded by ignoring noise for now, and designed PID controllers.

Starting off with just proportional control, we chose $c = k_p = 4000$. This gave us a stable loop, an infinite gain margin, and a 146° phase margin. However, we get a significant steady state error, because L(0) is only 1.203. Shown below are the Bode plot for L, and the step response for the closed loop system.

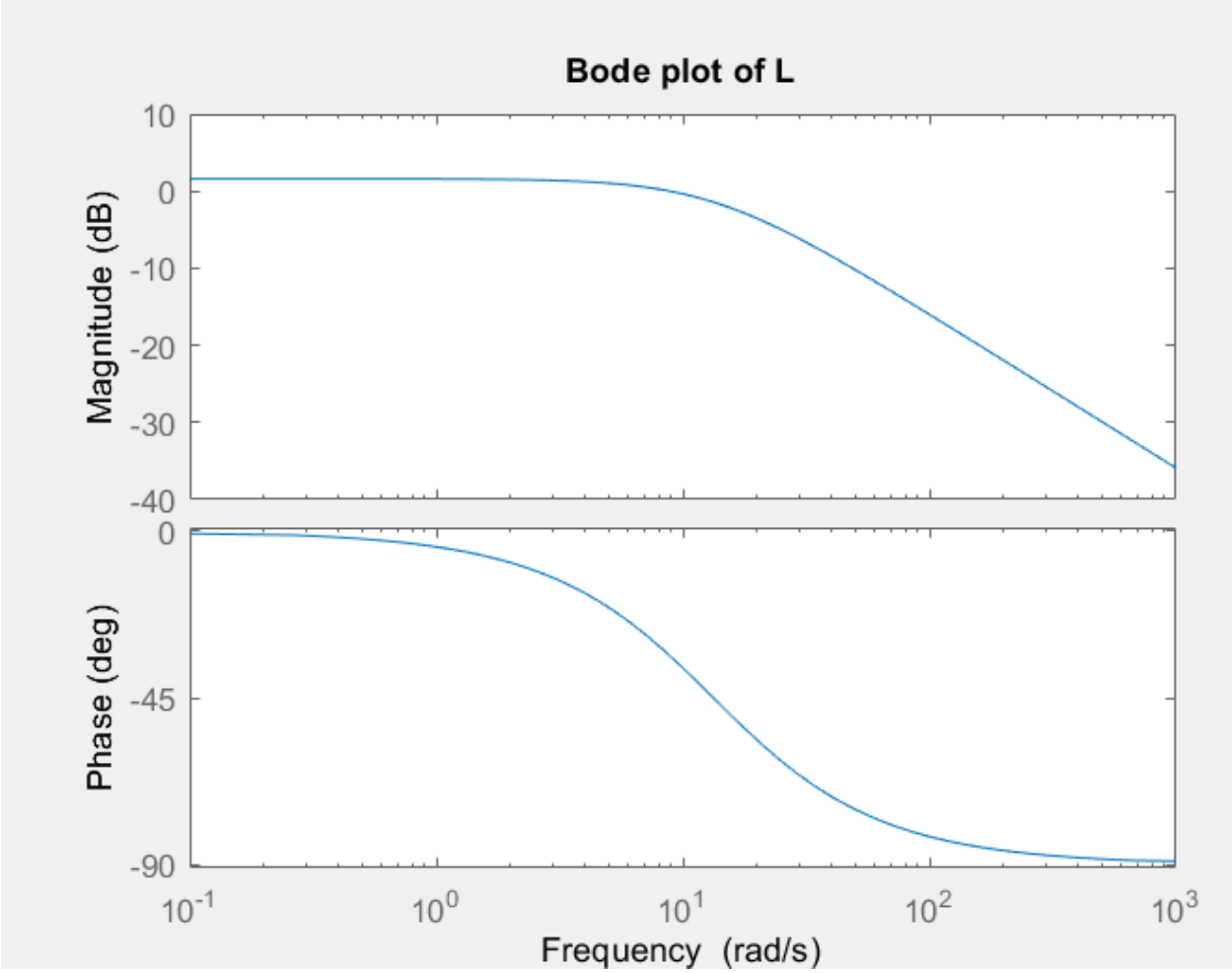


Figure 16: Bode plot for pure proportional control (k_p=4000) showing inadequate tracking performance.

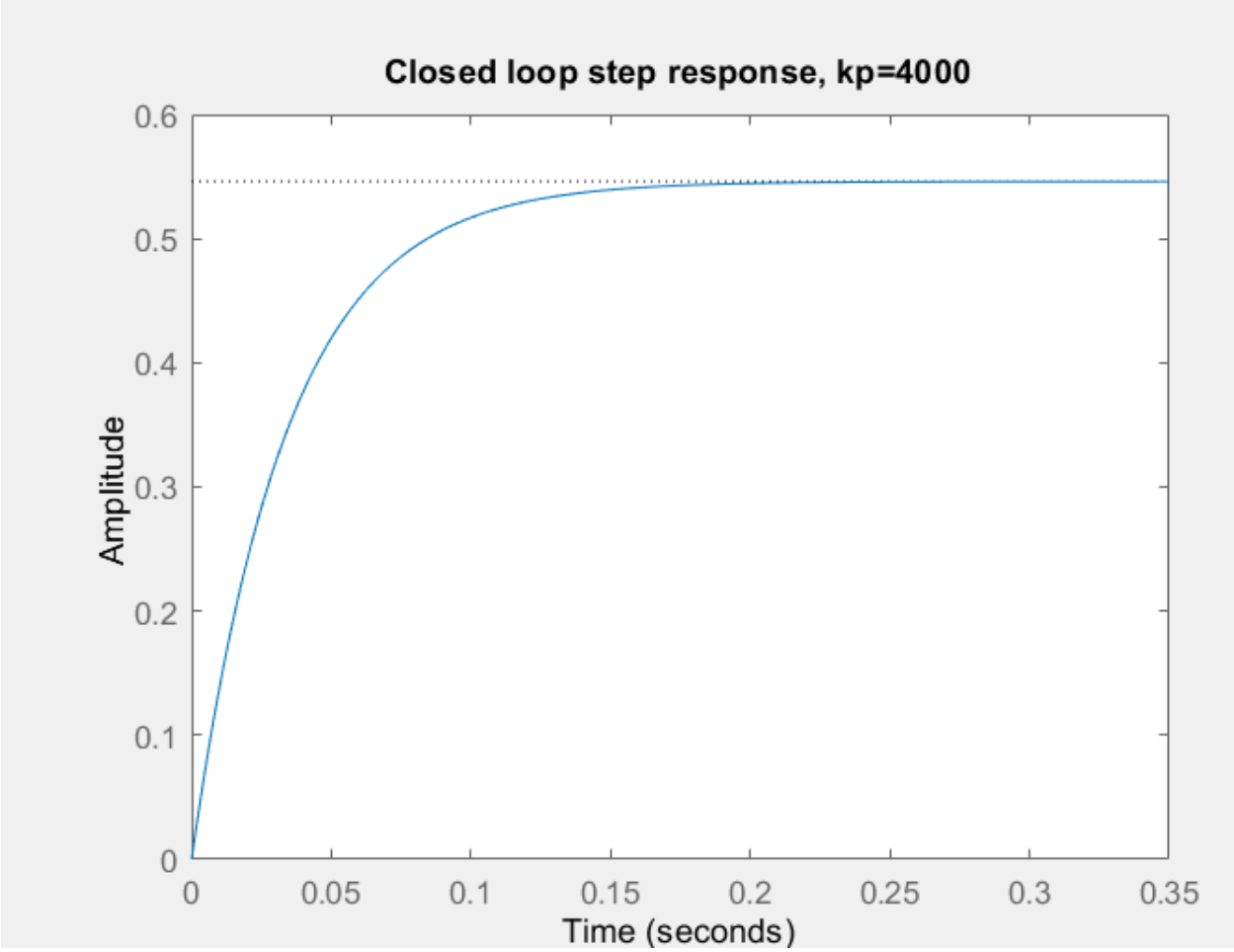


Figure 17: Step response for pure proportional control.

Next we added integral gain, using $k_p = 4000$, $k_i = 200$. As we can see in the step response and bode plots below, steady state error is eliminated (because L(0) is infinite), we have an infinite gain margin, and a phase margin of 146°. This makes sense, because integral control serves to eliminate steady state error, and add gain at low frequencies.

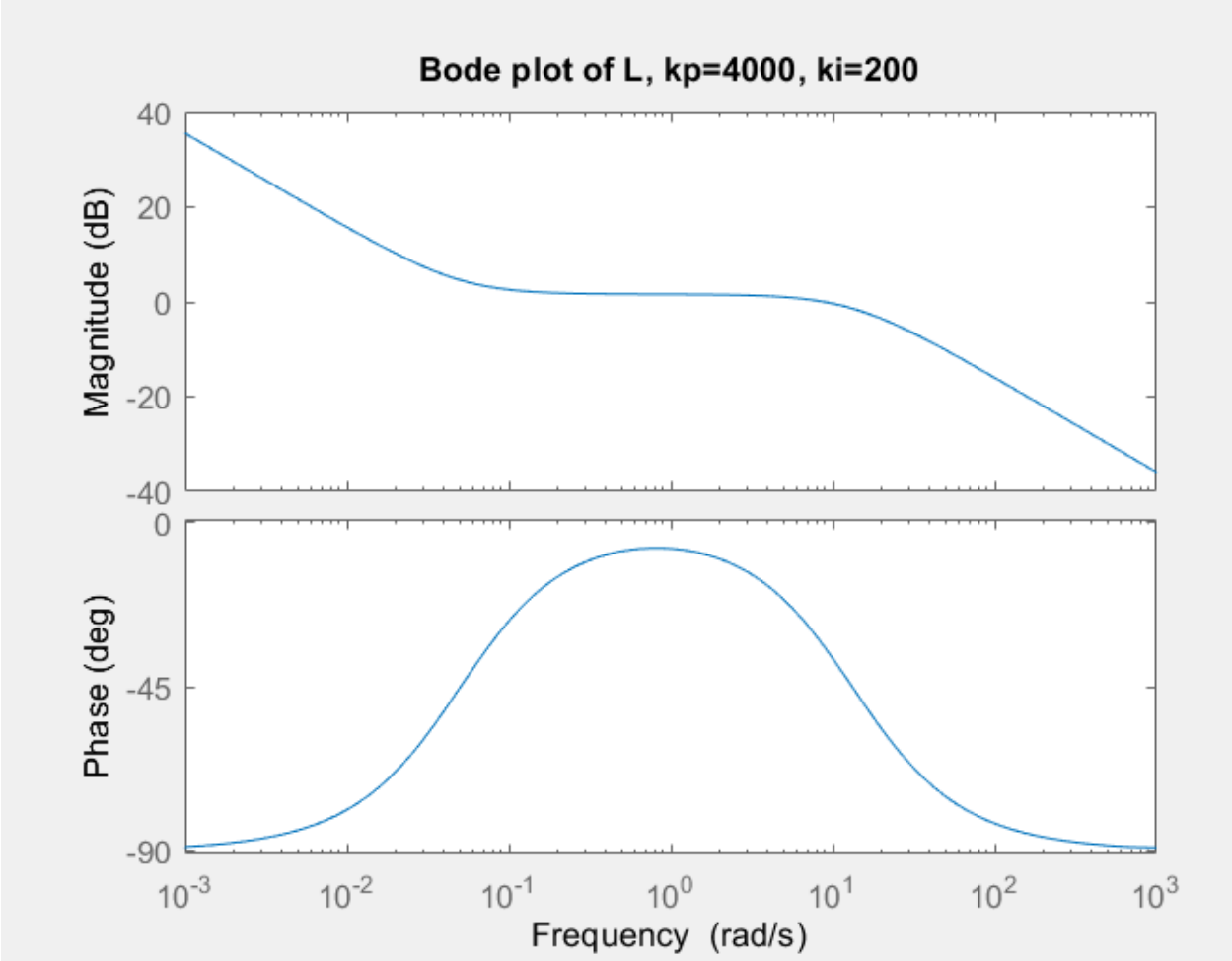


Figure 18: Bode plot of L(s) for PI control, shows substantially improved gain at low frequencies.

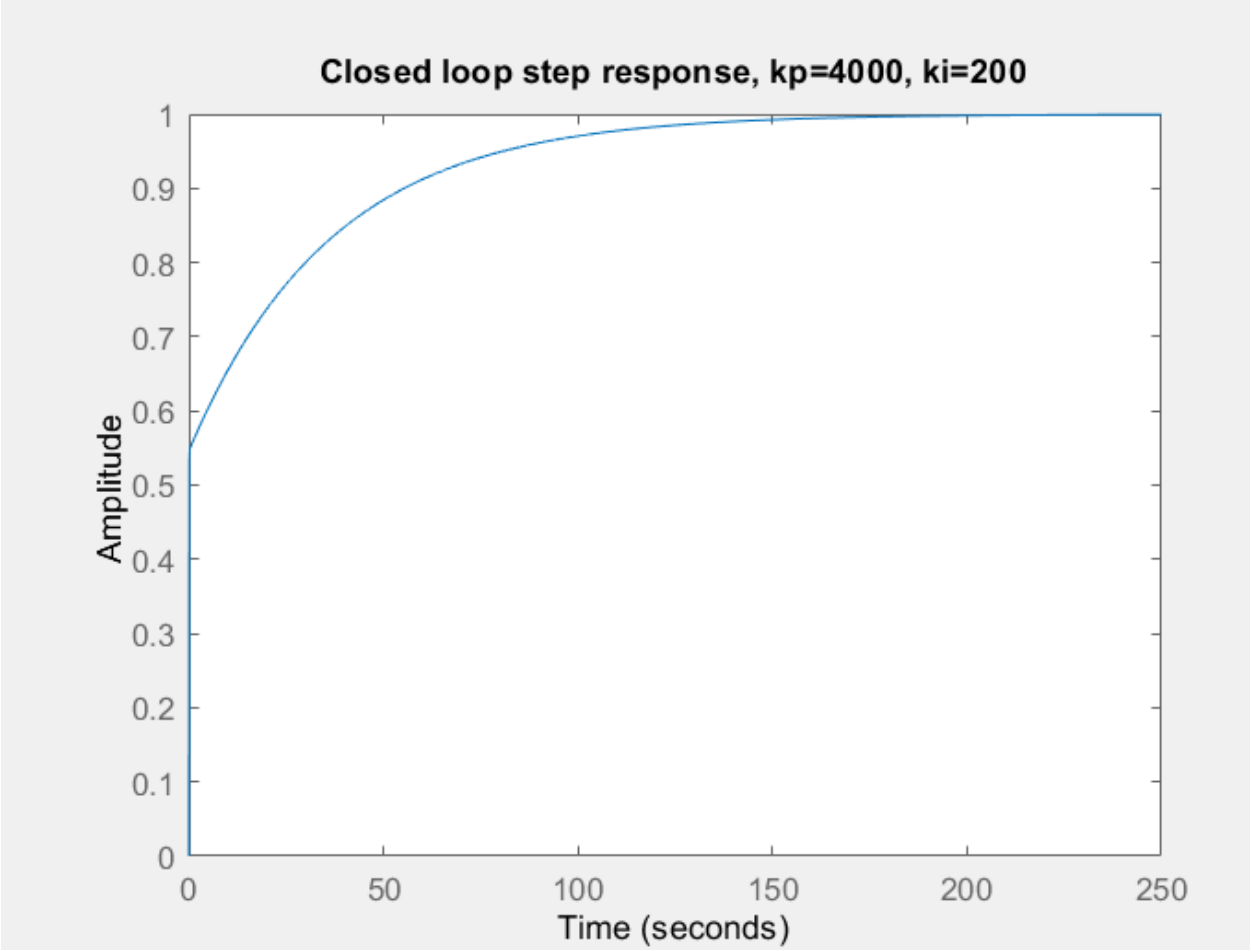


Figure 19: Closed loop step response for PI controller shows zero steady state error, but very slow response

However, if we consider the step response in Figure 19, we see the system has a very long rise time, of more than 80 seconds to reach 95%. Also the tracking performance is poor, it exceeds 10% at 10^{-2} rad/s already, as we can see from the error bode plot below.

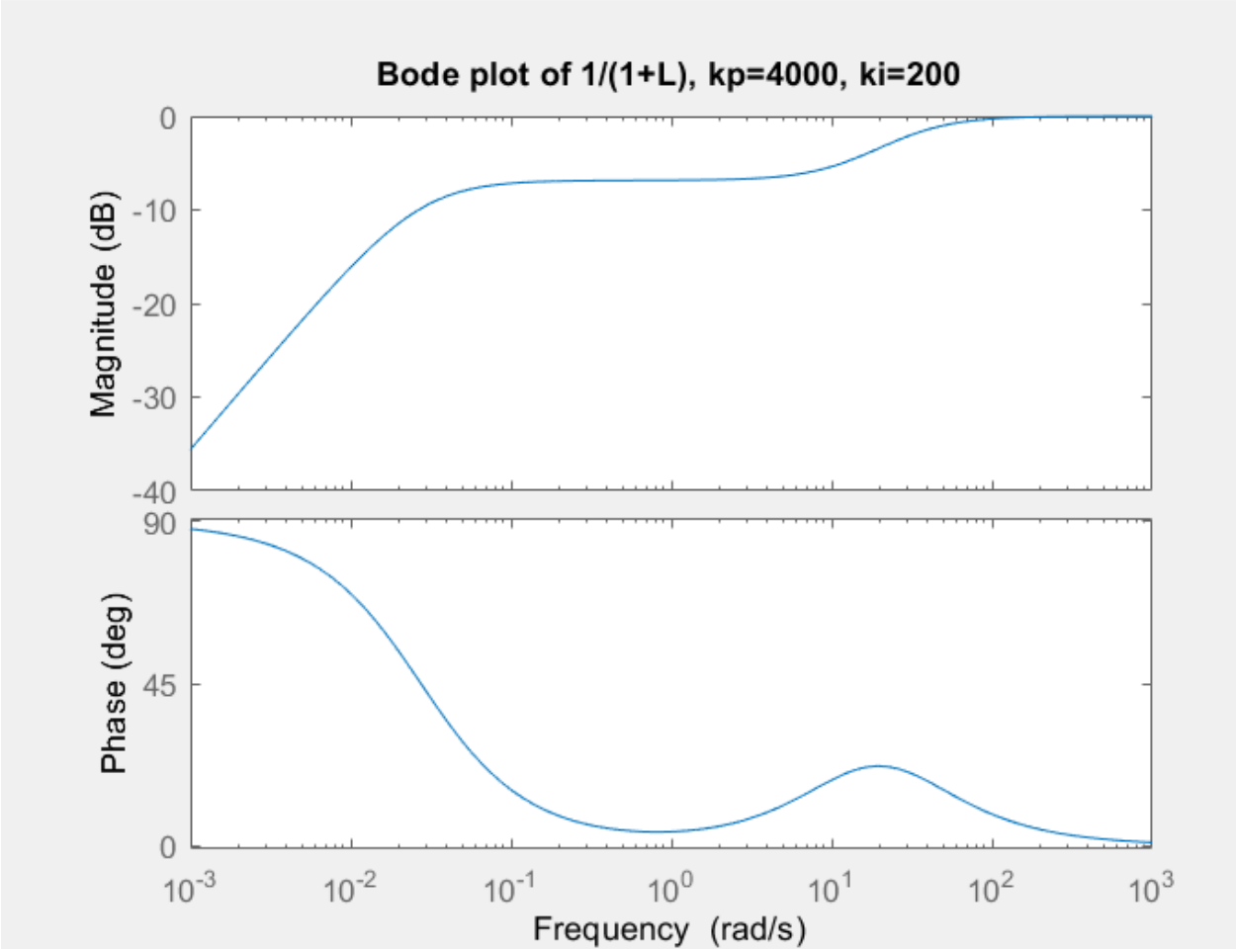


Figure 20: Tracing error bode plot

To improve these shortcomings, we add derivative control, and substantially increase the proportional and integral gain to $k_p = 40000$, $k_i = 200000$, $k_d = 4000$. This results in a system with less than 10% tracking error at all frequencies, and short rise times of <1 second. We continue to have no steady state error, an infinite gain margin and an infinite phase margin.

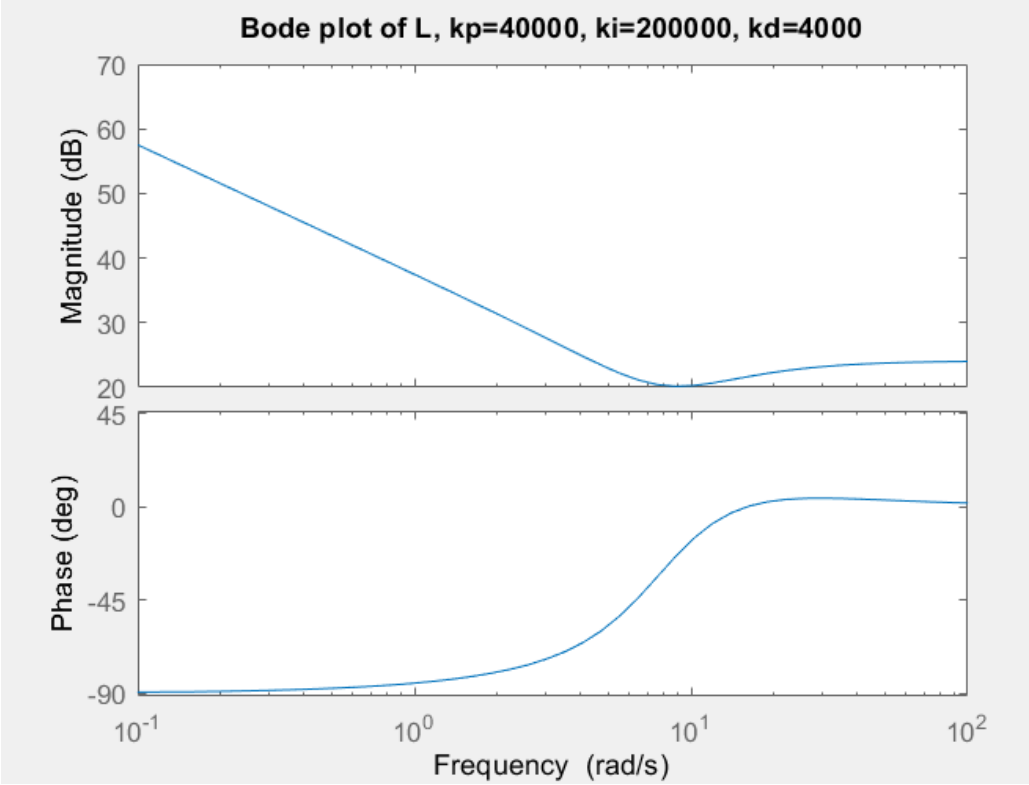


Figure 21: Bode plot of L for full PID control

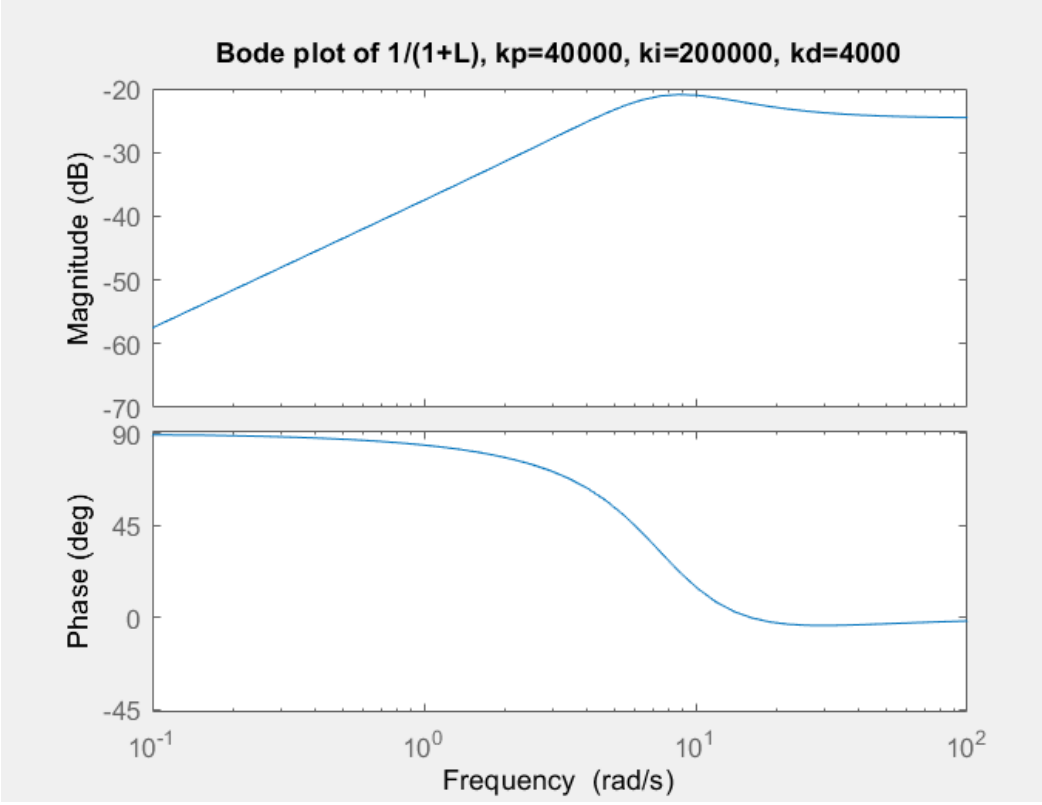


Figure 22: Bode plot of error dynamics for full PID control

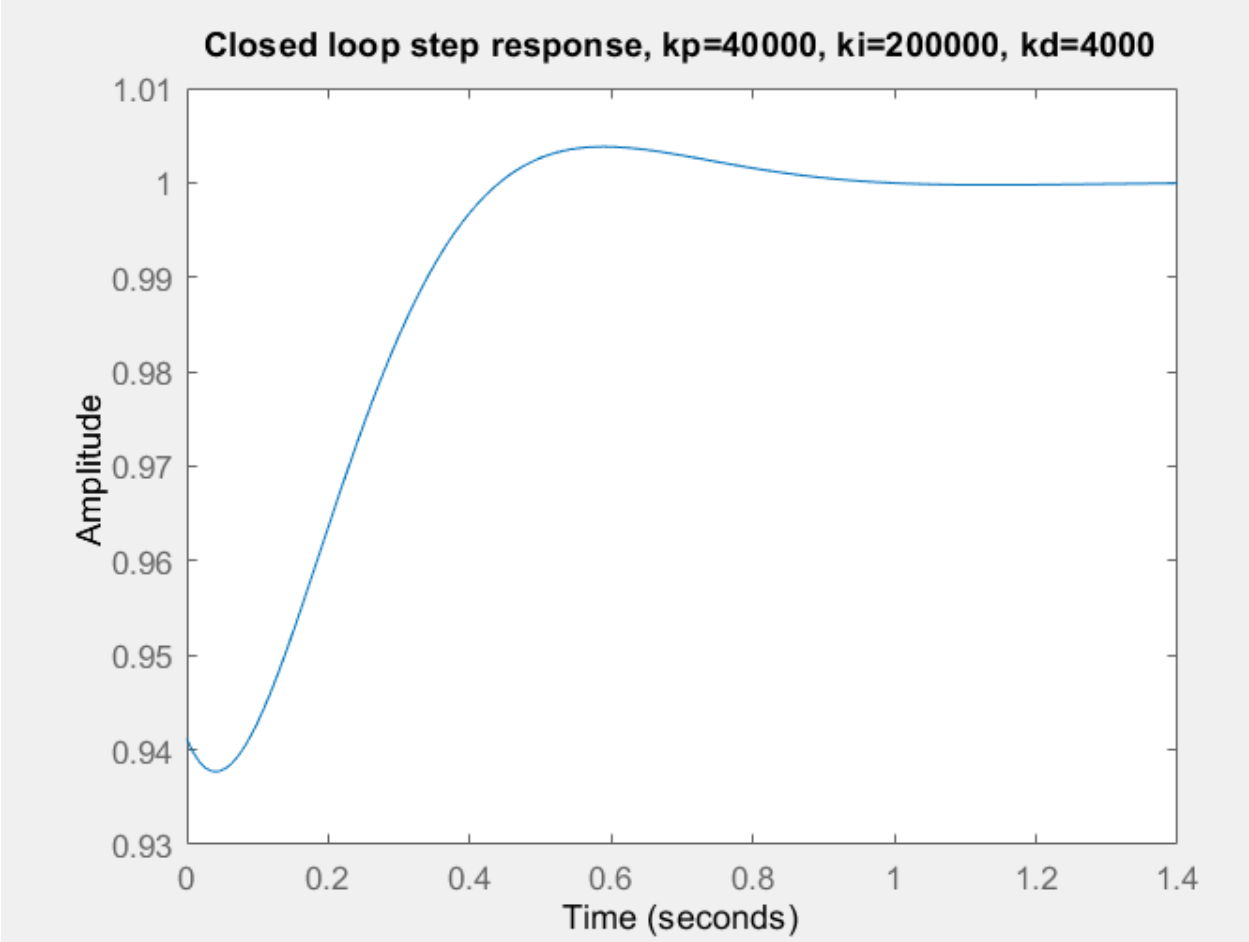


Figure 23: Step response for full PID control.

Step 4

The translations of the constraints on the system in regards to steady state error, tracking error, and phase margin are essential when developing an effective controller.

To set constraints on the behavior of the system as time goes to infinity, an ideal controller requires a steady state error of less than 2%. By the Final Value Theorem, this would translate to ensuring the error signal at steady state while in the time-domain, or equivalently, the frequency domain transfer function when s = 0, is less than 2%. In this case, we define $G_{r->e}$ to be the transfer function from the reference signal to the error.

$$G_{r\to e} = \frac{1}{1+L(s)}$$

$$\lim_{t\to\infty} e(t) = \left[\frac{1}{s}G_{r\to e}(s)\right]\cdot s\Big|_{s=0} = \frac{1}{1+L(0)}$$

For L(s) large or much greater than 1, we can approximate:

$$\frac{1}{1+L(0)} \approx \frac{1}{L(0)}$$
$$\frac{1}{L(0)} < 2\% \rightarrow \frac{1}{L(0)} < 0.02$$
$$L(0) > 50$$
$$20\ \log_{10}|L(0)| > 20\ \log_{10}|50|$$
$$20\ \log_{10}|L(0)| > 34\,dB$$

This means that the zero frequency gain is constrained to be above the magnitude 34 dB so that the steady state error is less than 2%.

A controller for the system also requires a tracking error of less than 10% from 0 to 1 rad/s:
$$\frac{1}{|L(iw)|} < 10\% \rightarrow \frac{1}{|L(iw)|} < 0.1$$
$$|L(iw)| > 10$$
$$20\ \log_{10}|L(iw)| > 20\ \log_{10}|10|$$
$$20\ \log_{10}|L(iw)| > 20dB$$
This means that from 0 to ω, the frequency gain should be above the magnitude of 20 dB.

Our system also requires a phase margin larger than 30 degrees, which can be depicted on the Bode phase plot:
$$P_m > 30^\circ$$

The PID controller designed in Step 3 fulfills all these specifications, where the controller $C(s)$ is of the form $C(s) = kp + \frac{ki}{s} + kd \cdot s$, where kp = 40000, ki = 200000, and kd = 4000. Step and Bode plots are observed in *Figure 16* and *Figure 18*.

Other controllers were made using MATLAB's *controlSystemDesigner* tool.

One of such controllers is a PID controller, where $C(s)$ can also be interpreted as:
$$C(s) = \frac{k(s+z_1)(s+z_2)}{s}$$
Where k is the gain, and z_1 and z_2 are two real zeros. This means that after implementing two zeros and an integrator, a pole at 0, the controller is mainly designed through setting k such that the tracking error constraint is fulfilled and placing the zeros such that the phase requirement is satisfied.

For this controller, the following equation holds:
$$C(s) = \frac{2931.5(s+3.396)(s+26.47)}{s},\ k = 2931.5$$

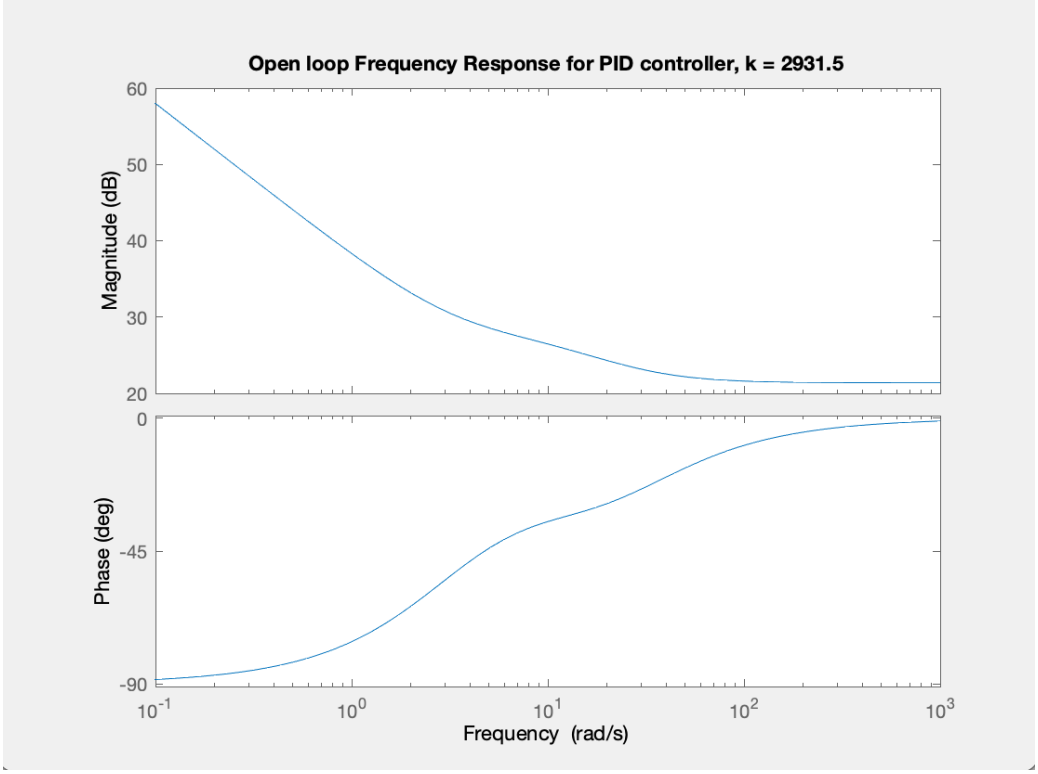


Figure 24: Bode plot of open loop system for full PID control, k = 2931.5

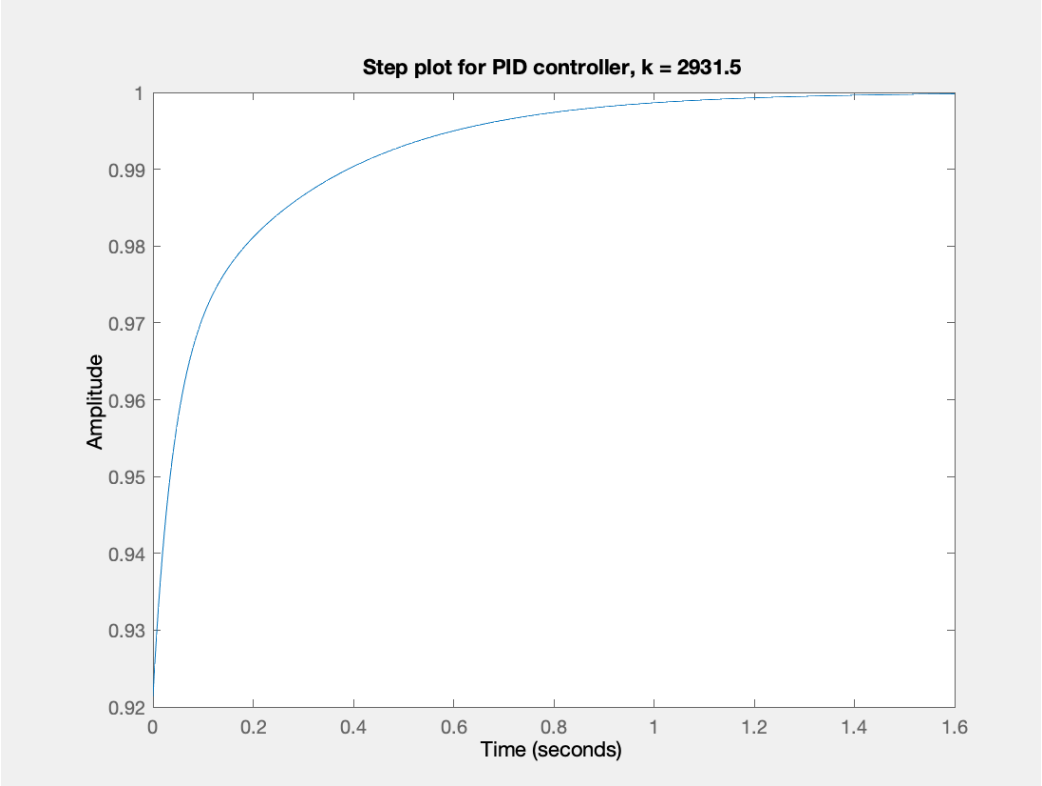


Figure 25: Step response for full PID control, k = 2931.5

As denoted by the Bode plot in *Figure 24*, the controller satisfies the steady state error requirement by reaching a magnitude greater than 34 dB at L(0). the controller also satisfies the tracking error and phase margin requirement by having a magnitude greater than 20 dB from 0 to ω (as a result of setting the proportional gain), and an infinite phase margin.

The step response of the closed loop system is shown in *Figure 25*. For this selected controller, there is an overshoot of 0%, zero steady state error, a rise time of 0 seconds, and a settling time of around 0.2 seconds. While this controller satisfies the requirements, the rise time is extremely aggressive and may not perform the best in a real life setting.

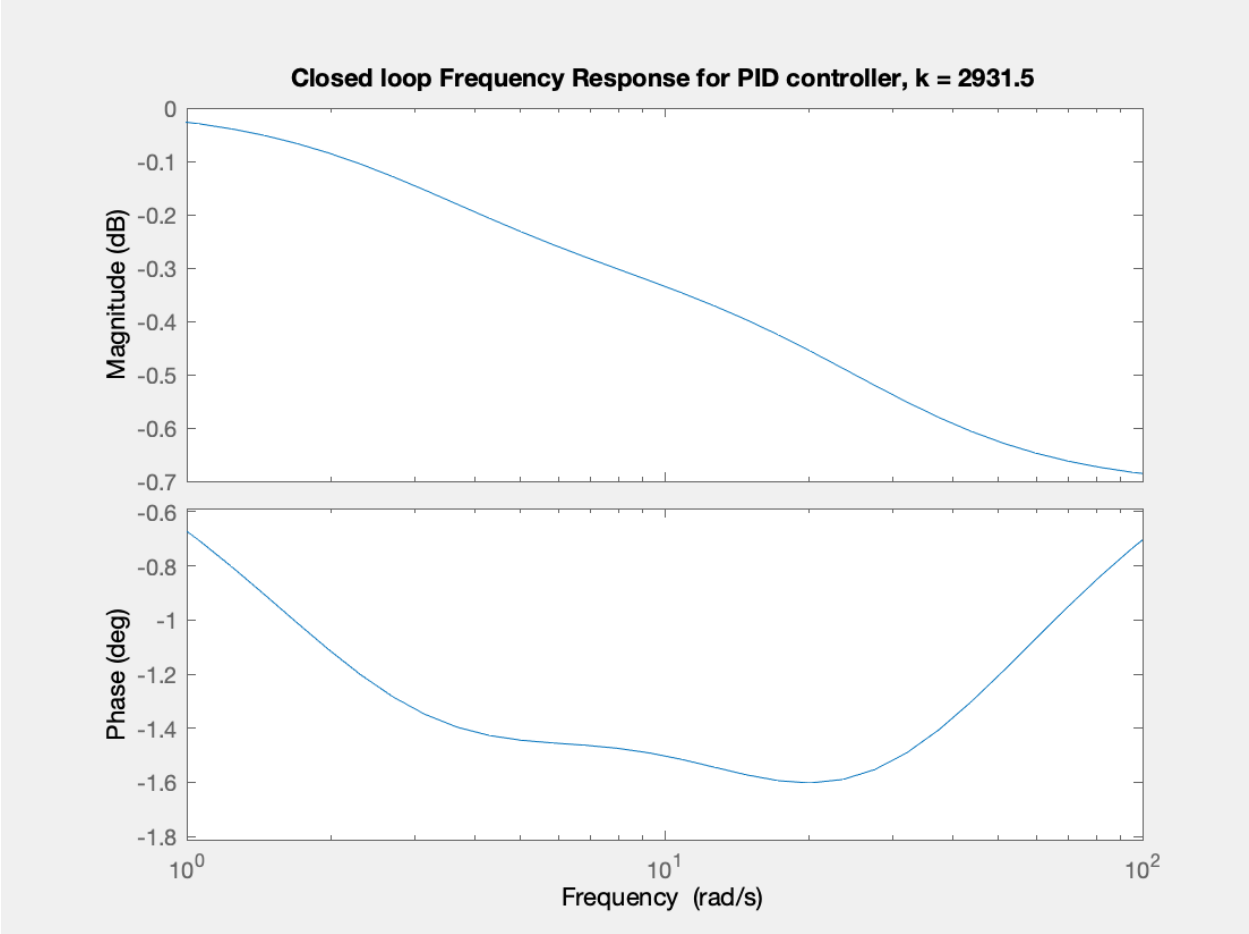


Figure 26: Closed loop frequency response for full PID controller, k = 2931.5

Another controller observed was a PD controller with Lead Compensation and an additional pole and zero to maximize the phase margin and increase gain:

$$C(s) = \frac{7681.6(s+4.21)(s+7.502)}{(s+0.326)(s+3.027)}$$

,where values of the poles are greater than those of the zeros.

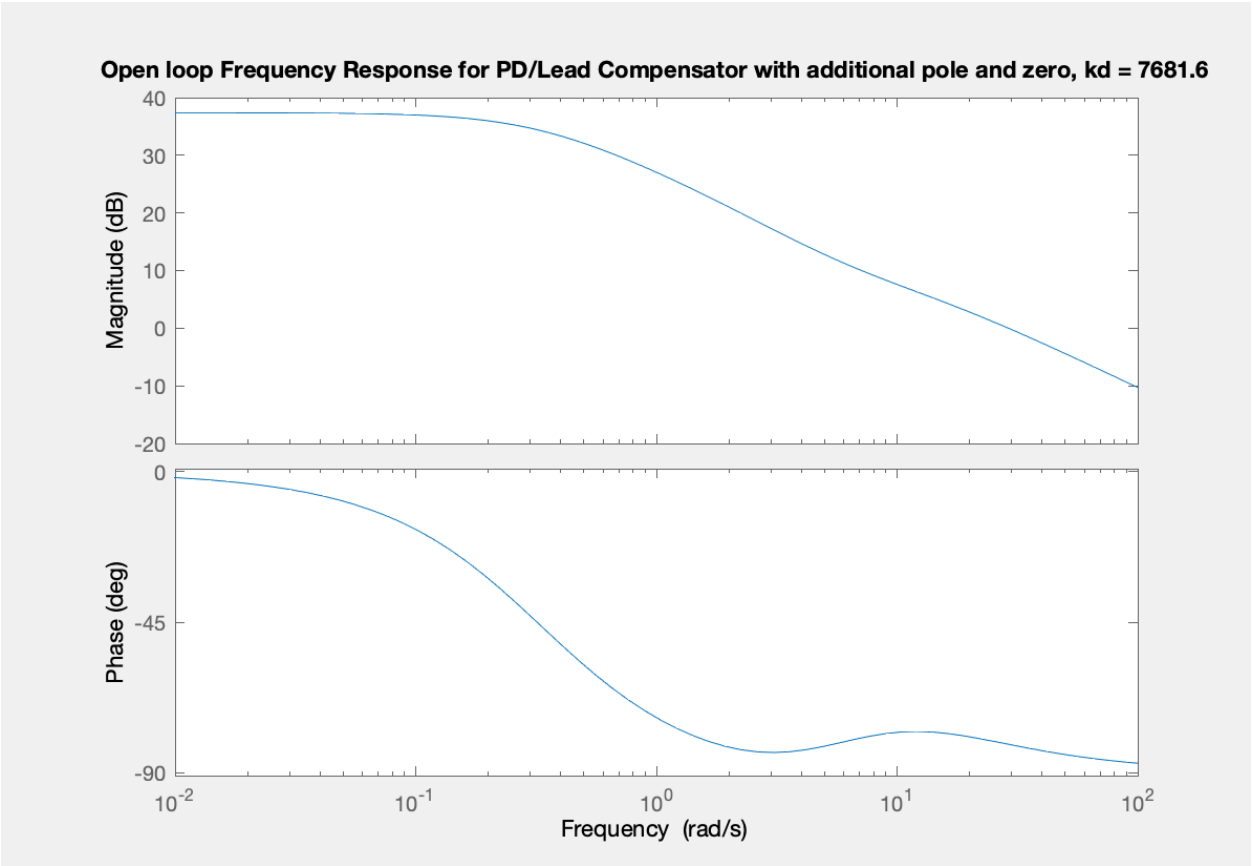


Figure 27: Bode plot of open loop system for Lead Compensator

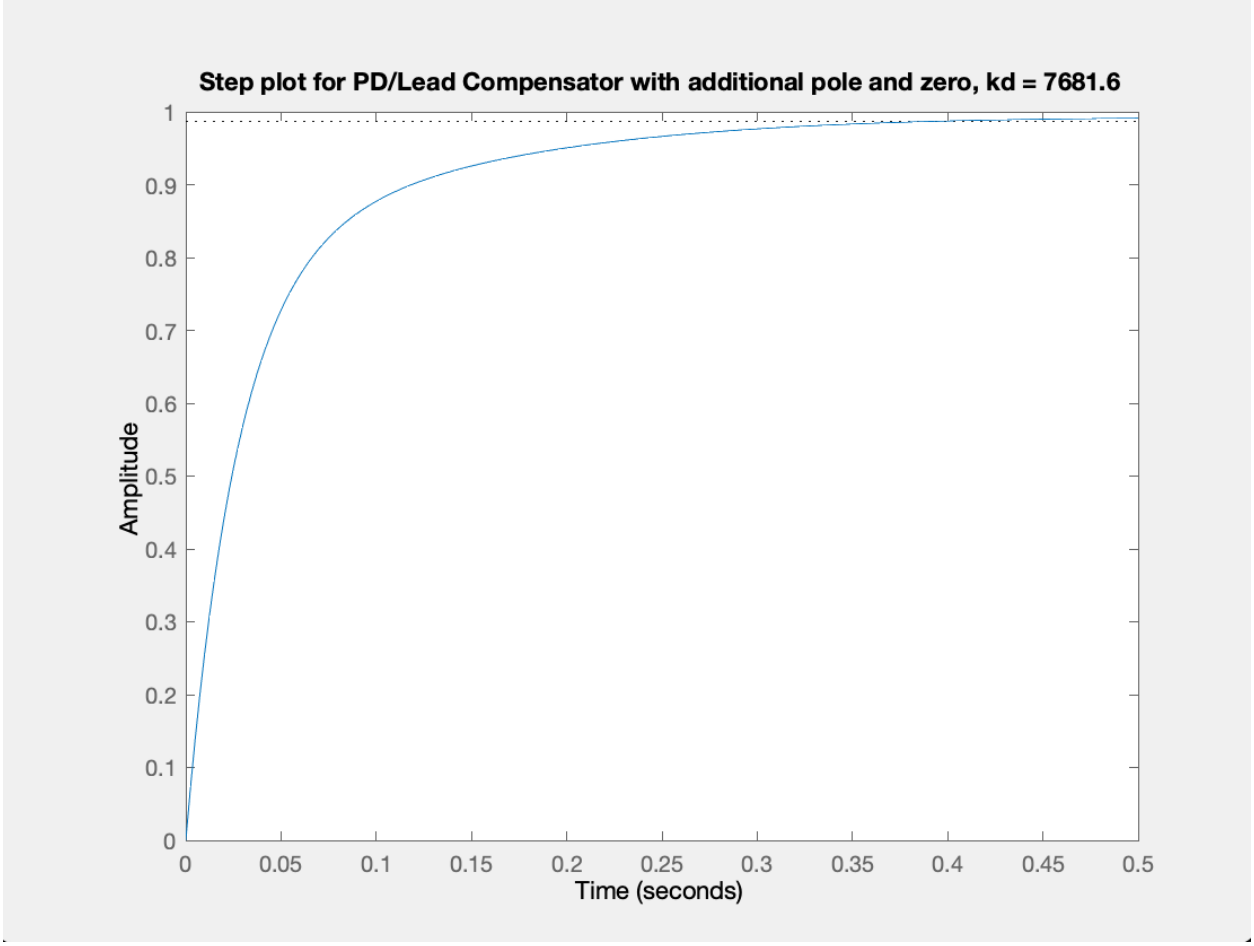


Figure 28: Step response for Lead Compensator

As denoted by the Bode plot in *Figure 27*, the controller satisfies the steady state error requirement by reaching a magnitude greater than 34 dB at L(0). the controller also satisfies the tracking error and phase margin requirement by having a magnitude greater than 20 dB from 0 to ω (as a result of setting the proportional gain), and an infinite phase margin.

The step response of the closed loop system is shown in *Figure 28*. For this selected controller, there is a slight overshoot of 0.5131, a steady state error of less than 1.4%, a rise time of 0.1045 seconds, and a settling time of around 0.2528 seconds.

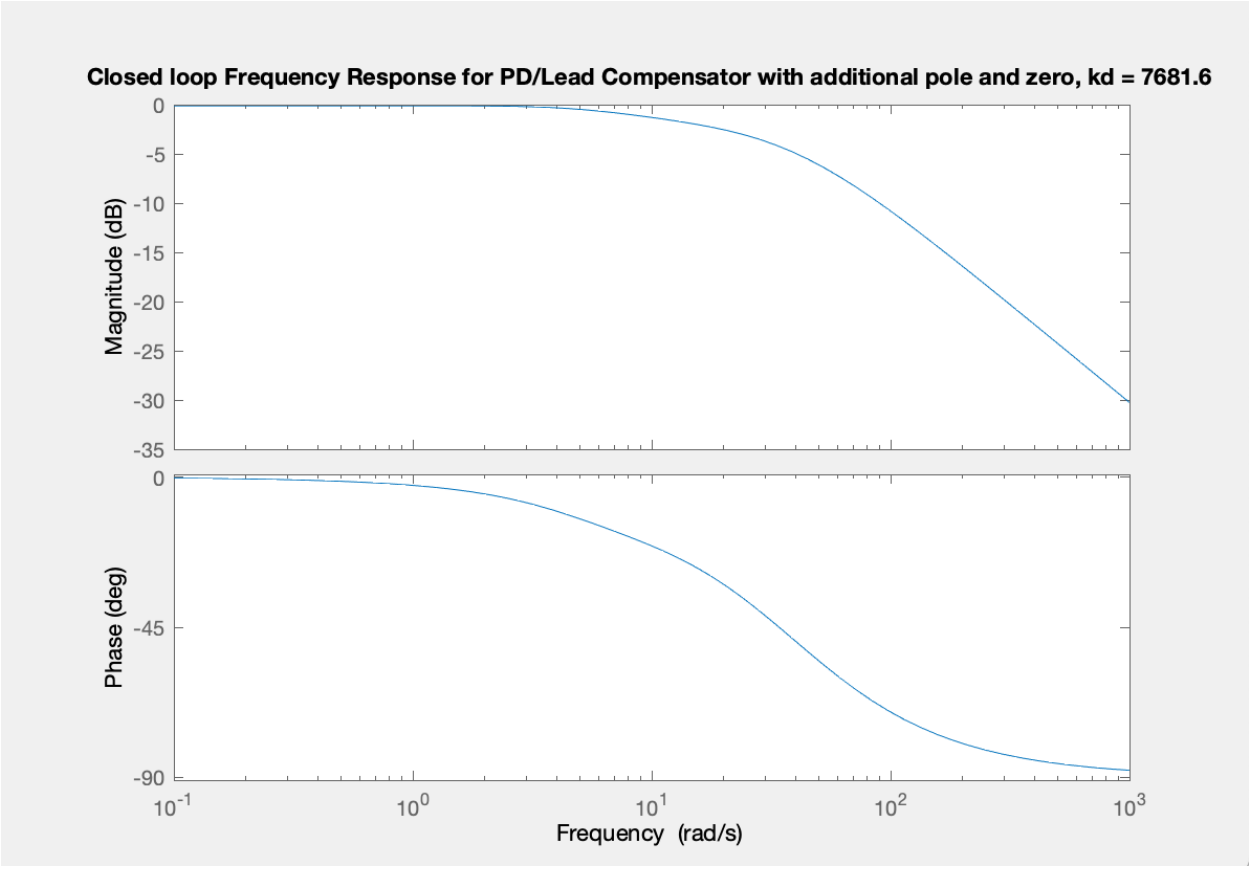


Figure 29: Closed loop frequency response for Lead Compensator

Step 5

The closed loop frequency responses for the controllers discussed in Steps 3 and 4 give substantial information about the behavior of the system in response to noise and other external disturbances.

For an output y in a system, we know that:

$$y = \frac{P(s)C(s)}{1+P(s)C(s)}(r-n) + \frac{1}{1+P(s)C(s)} \cdot d$$
$$= \frac{L(s)}{1+L(s)}(r-n) + \frac{1}{1+L(s)} \cdot d$$

Which means that the closed loop frequency responses in the Bode plots for the controllers can detail effects from the reference signal (r) and the noise (n) on the system. Ideally, some flatter zero gain for lower frequencies in the magnitude plot would signify that the output y would closely follow the desired reference signal r, and decreasing behavior on the magnitude plot for higher frequencies would lead to a system that can reject really high frequencies or noise.

For the PID controller in *Figure 21*, the magnitude flattens out at 20dB, indicating that the high-frequency noise rejection may not be strong. we can start to see the flatter behavior of the

gain plot for smaller values of ω as the phase reaches 0°. In *Figure 29*, the Lead compensator acts more like a low pass filter given that the gain is exactly zero (along with the phase at L(0)) and relatively flat for low frequencies, which means that the output is following the desired reference well while canceling out large frequencies due to noise and other disturbances.

One can observe the decreasing behavior of the gain plots in *Figure 26* and *Figure 29* as ω increases, which means that the system is effectively rejecting noise at those higher frequencies using these controllers.

The performance of the system can also be observed with a sinusoidal input:

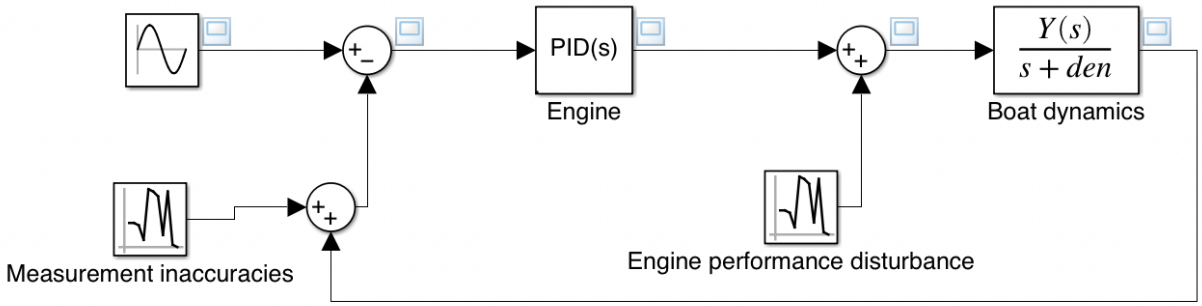


Figure 30: Block diagram of system with sinusoidal reference value

Testing the full PID controller from Step 3 with kp = 40000, ki = 200000, and kd = 4000, and an input sine wave with frequency ω = 1 in Simulink (where variance is 0.05, corresponding to 5% of reference input) results in the following system response:

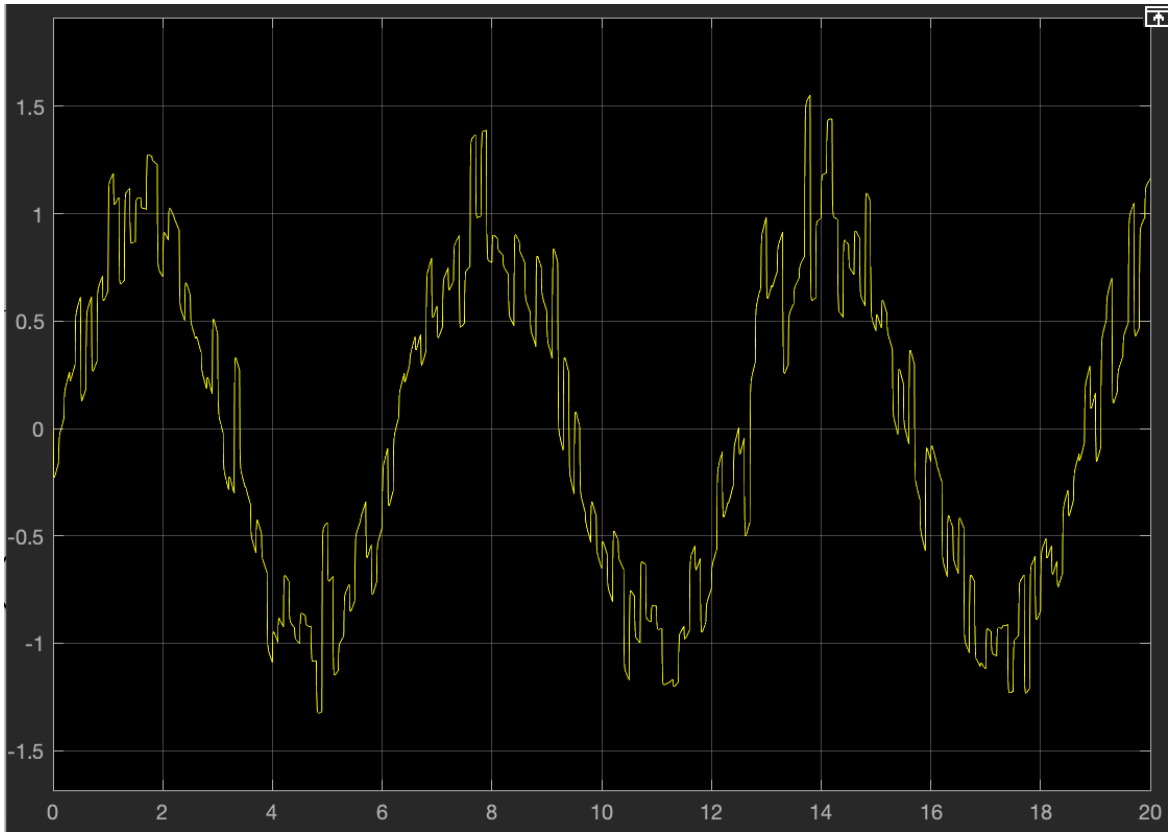


Figure 31: Sinusoidal response with added measurement noise simulation (5% variance relative to reference, $\omega = 1$)

Changing the frequency value from $\omega = 1$ to $w = 10$ achieves the following system response:

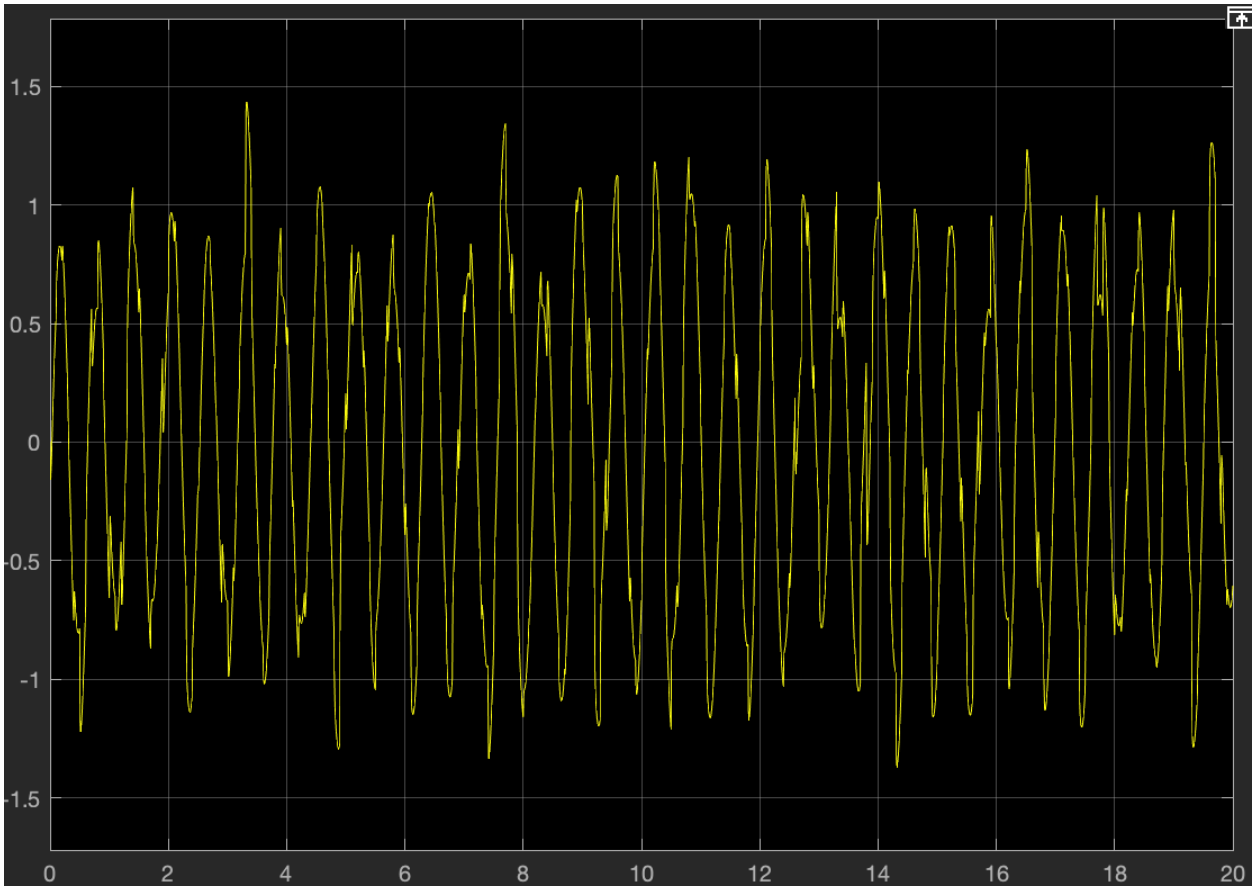


Figure 32: Sinusoidal response with added measurement noise simulation (5% variance relative to reference, $\omega = 10$)

Increasing the frequency even more, to $\omega = 100$, achieves the following system response:

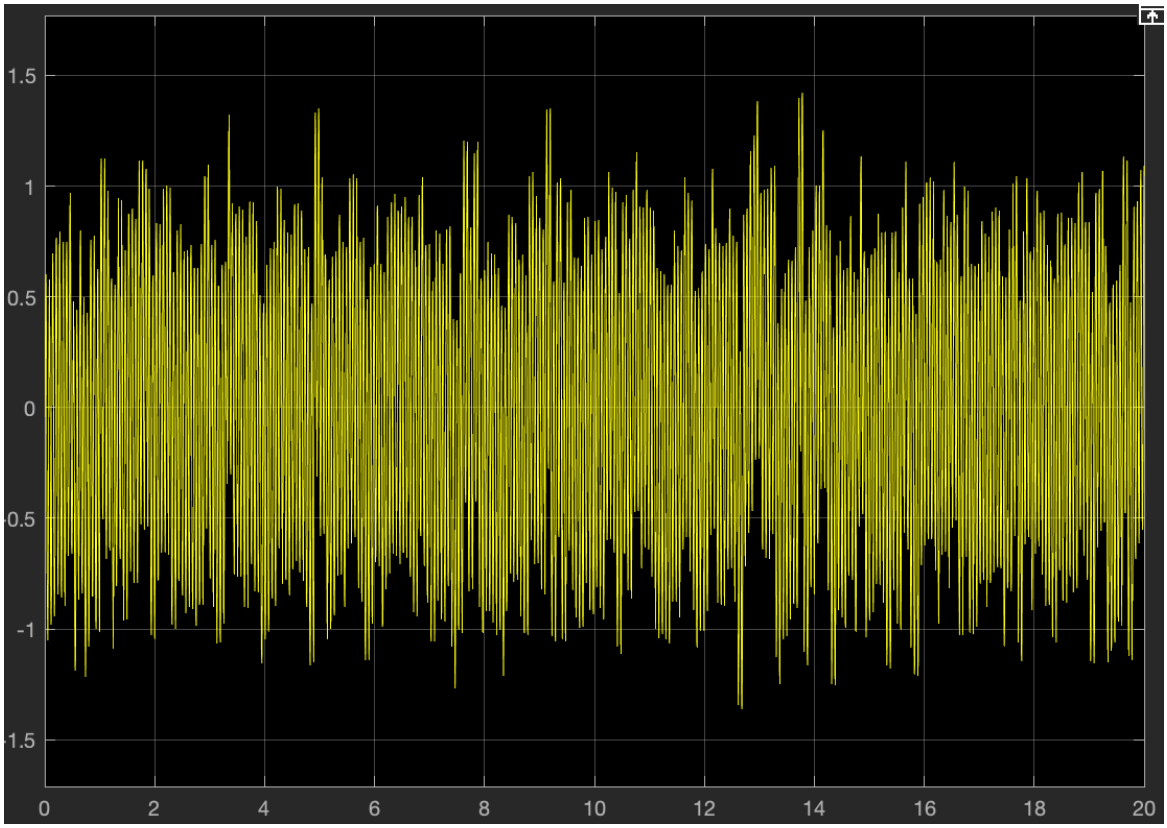


Figure 33: Sinusoidal response with added measurement noise simulation (5% variance relative to reference, $\omega = 100$)

To further analyze these system responses, we can look at the bode plots of the closed loop system from the reference signal to the output, from the noise to the output, and disturbance to the output:

$$G_{r \to y} = \frac{L(s)}{1 + L(s)}, \text{ reference to output}$$

$$G_{n \to y} = -\frac{L(s)}{1 + L(s)}, \text{ noise to output}$$

$$G_{d \to y} = \frac{P(s)}{1 + L(s)}, \text{ disturbance to output}$$

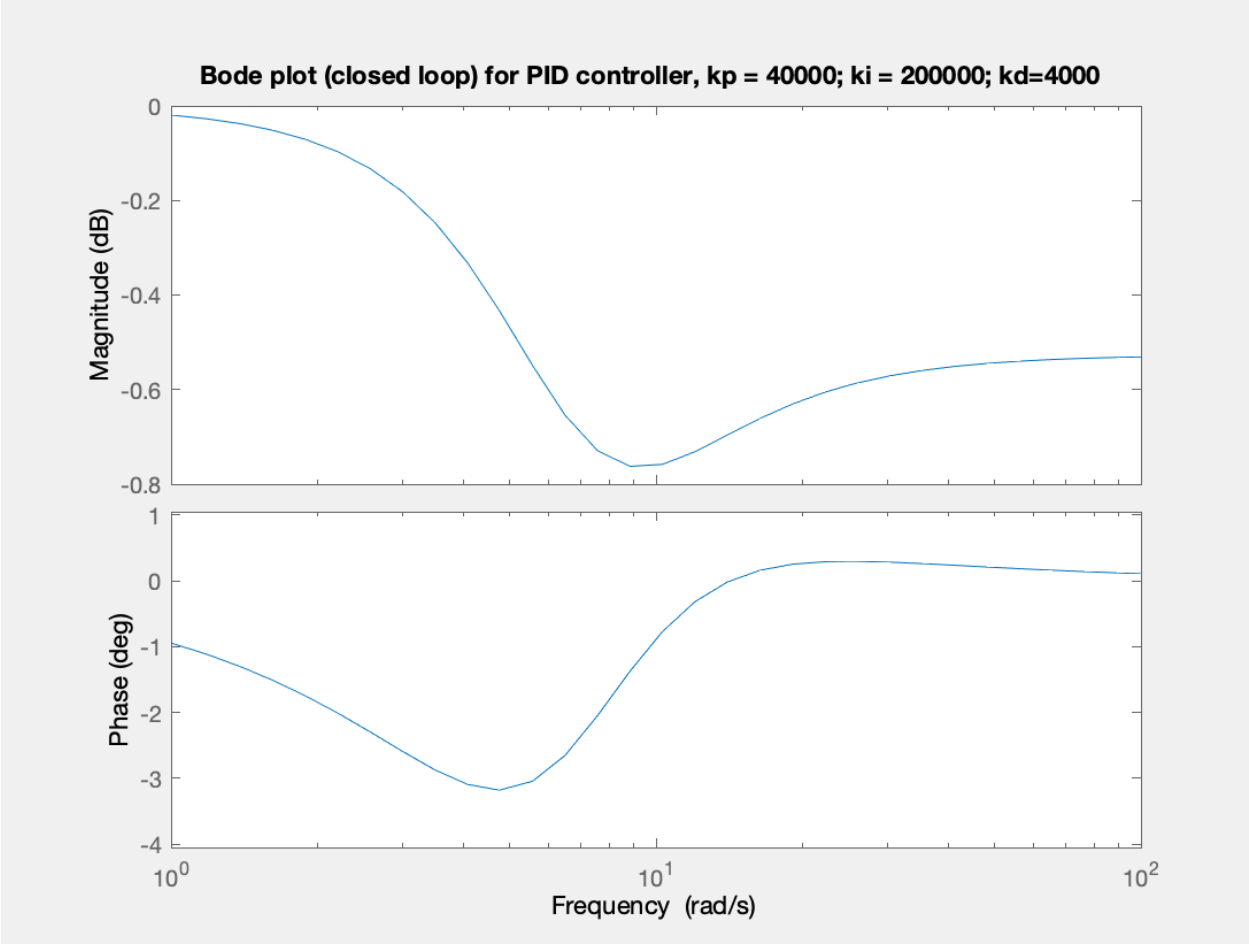


Figure 34: Bode Plot for closed loop system (reference to output) for full PID control

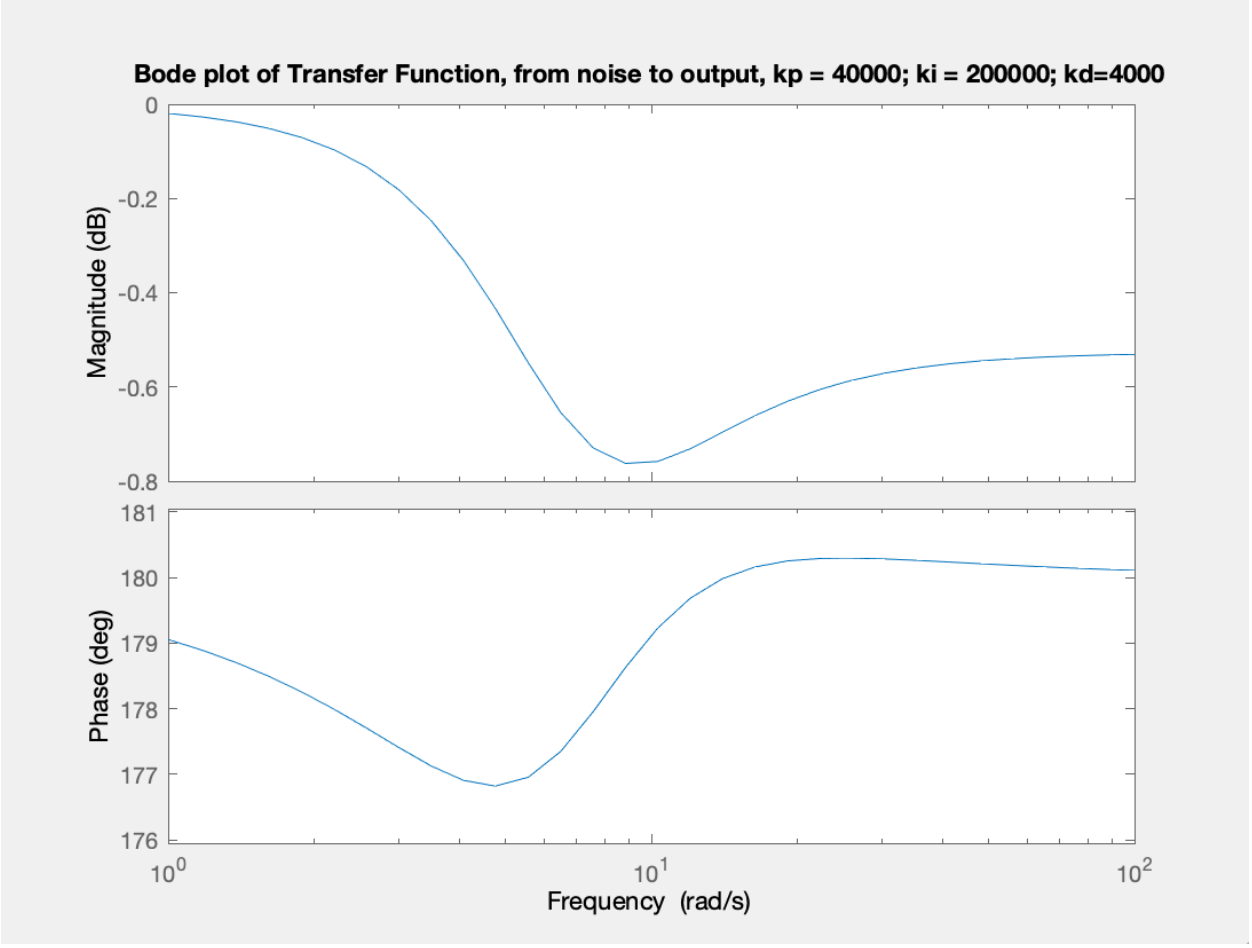


Figure 35: Bode Plot for closed loop system (noise to output) for full PID control

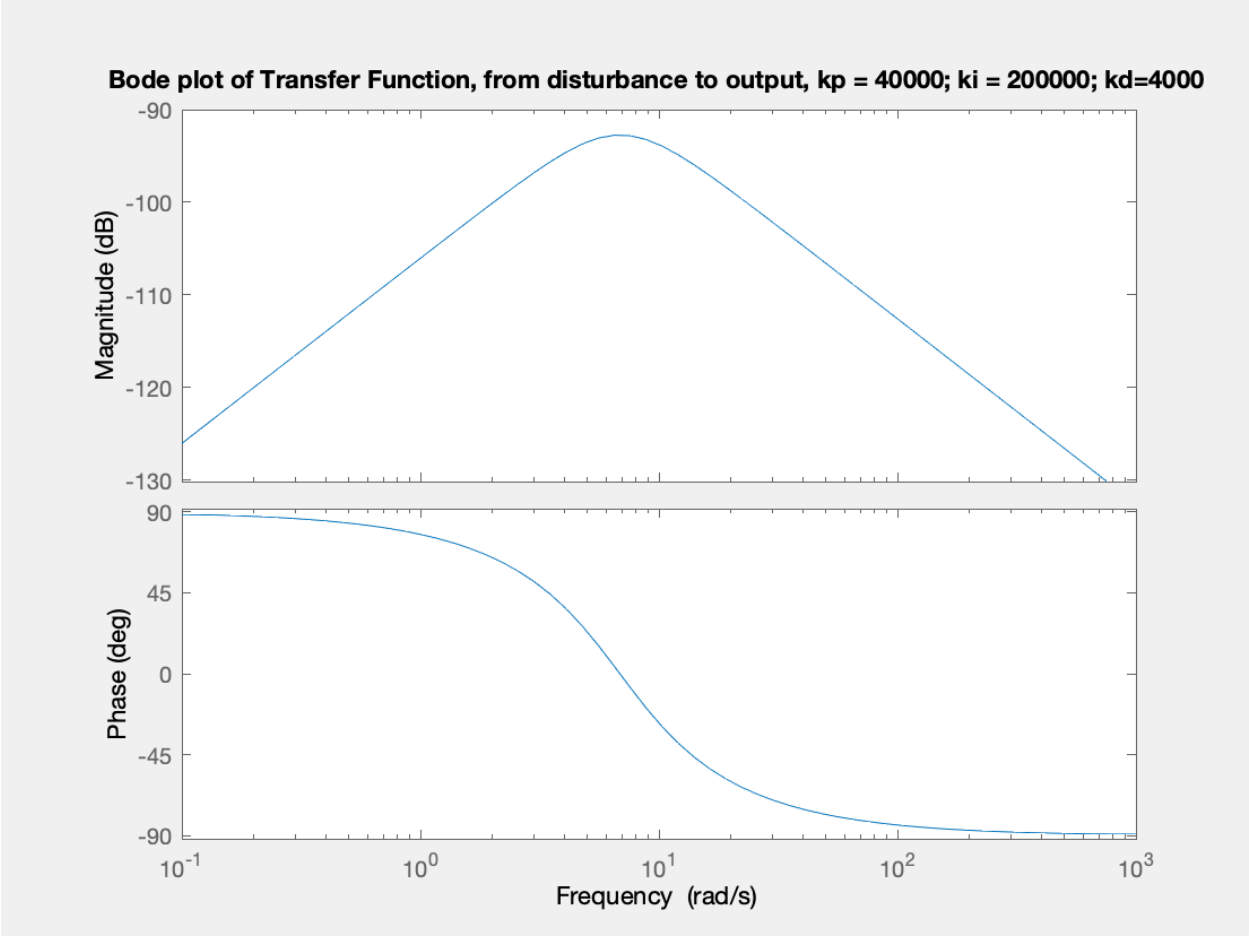


Figure 36: Bode Plot for closed loop system (disturbance to output) for full PID control

The response is generally good from reference signal to output, but one can still observe the effects noise has on the system even when the controller is applied. In particular, one can note that the system response is much better given higher frequencies rather than lower frequencies. A clear example of this is in *Figure 36*, where we can see that at lower frequencies, there is a lot of disturbance (be it from the engine, or other external disturbances) affecting the system, and as the frequency increases, the magnitude decreases significantly along with the phase. For plots such as *Figure 35*, we can also note that as frequency increases the gain decreases and eventually flattens out to a much lower value than at lower frequencies, which is reflected in the system responses in the scope.

Step 6

When we tested the last combination of $k_p = 40000,\ k_i = 200000,\ k_d = 4000$ in Simulink with added measurement noise (variance 0.05, corresponding to 5% of reference input), we still had a somewhat reasonable step response as shown below.

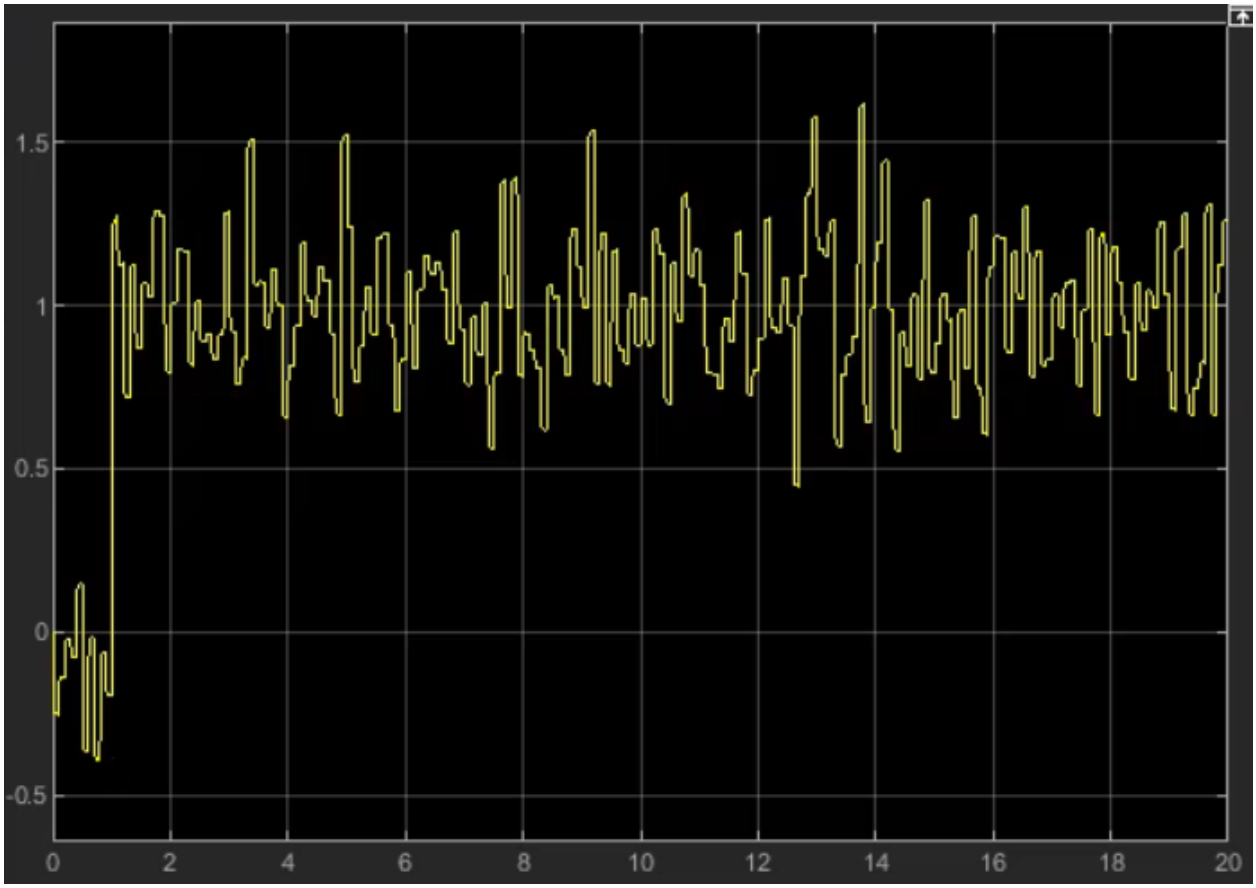


Figure 37: Step response with added measurement noise simulation (5% variance relative to reference)

If instead we add just 2% noise, the response is better, but there is still substantial variability, the noise is certainly being amplified, because the variability of the step response is higher than the variability of the noise. This is likely due to the high k_d value.

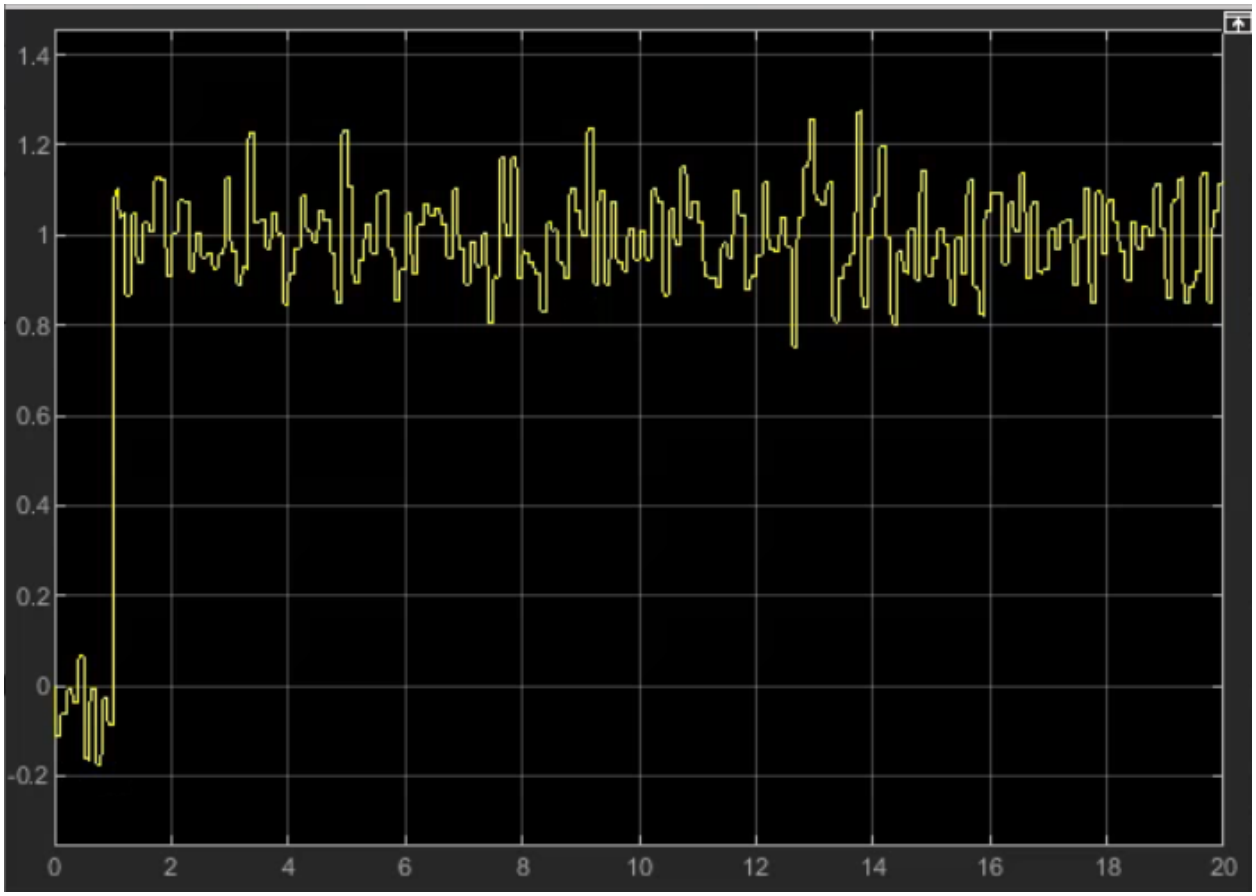


Figure 38: Step response for 2% noise

But this hypothesis was not confirmed, even when we set the derivative gain to zero, it did not substantially change the variability of the step response.

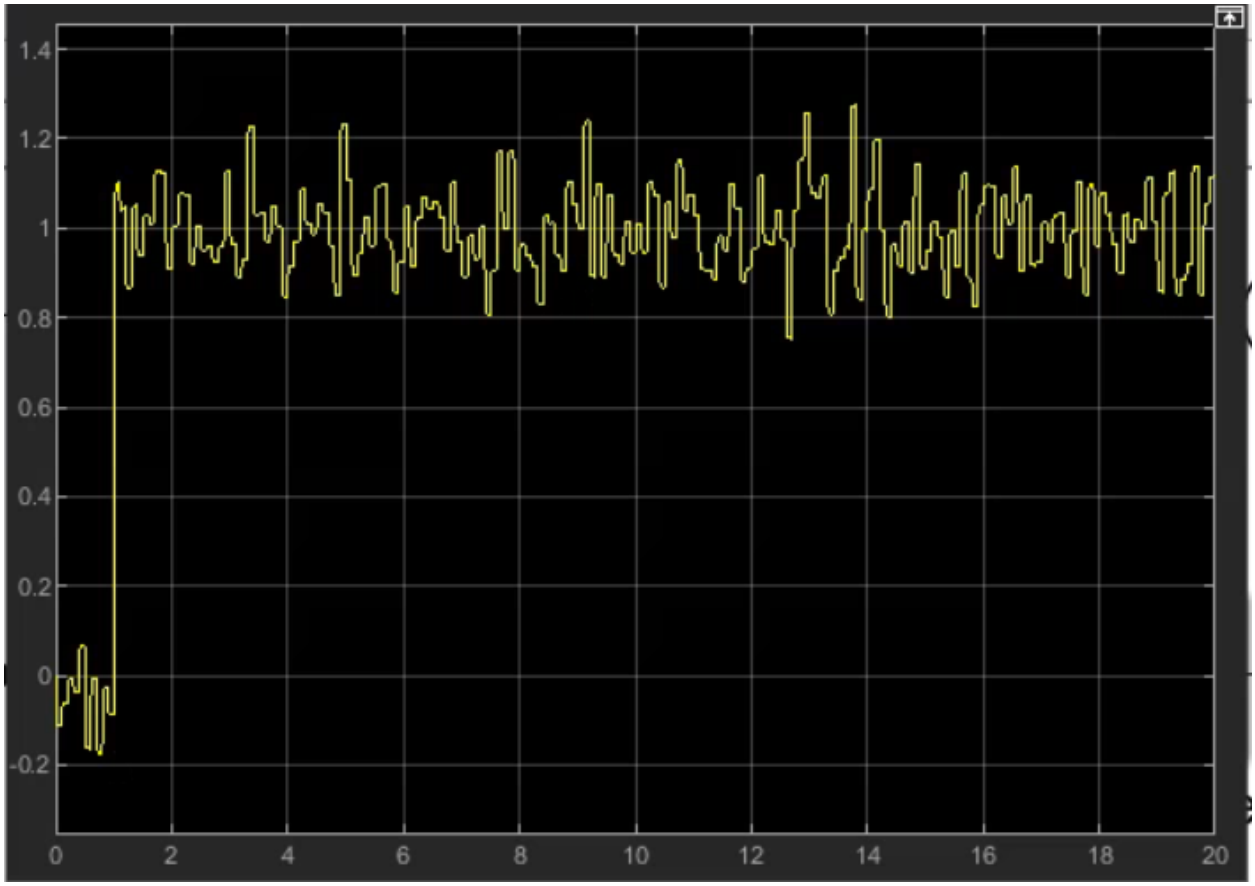


Figure 39: Step response for 2% error and 0 derivative gain

We also tried very high measurement noise of 30% with the original derivative gain. This substantially reduced the tracking performance, at this point the noise results in deviations of more than 100% of the step response from the reference value.

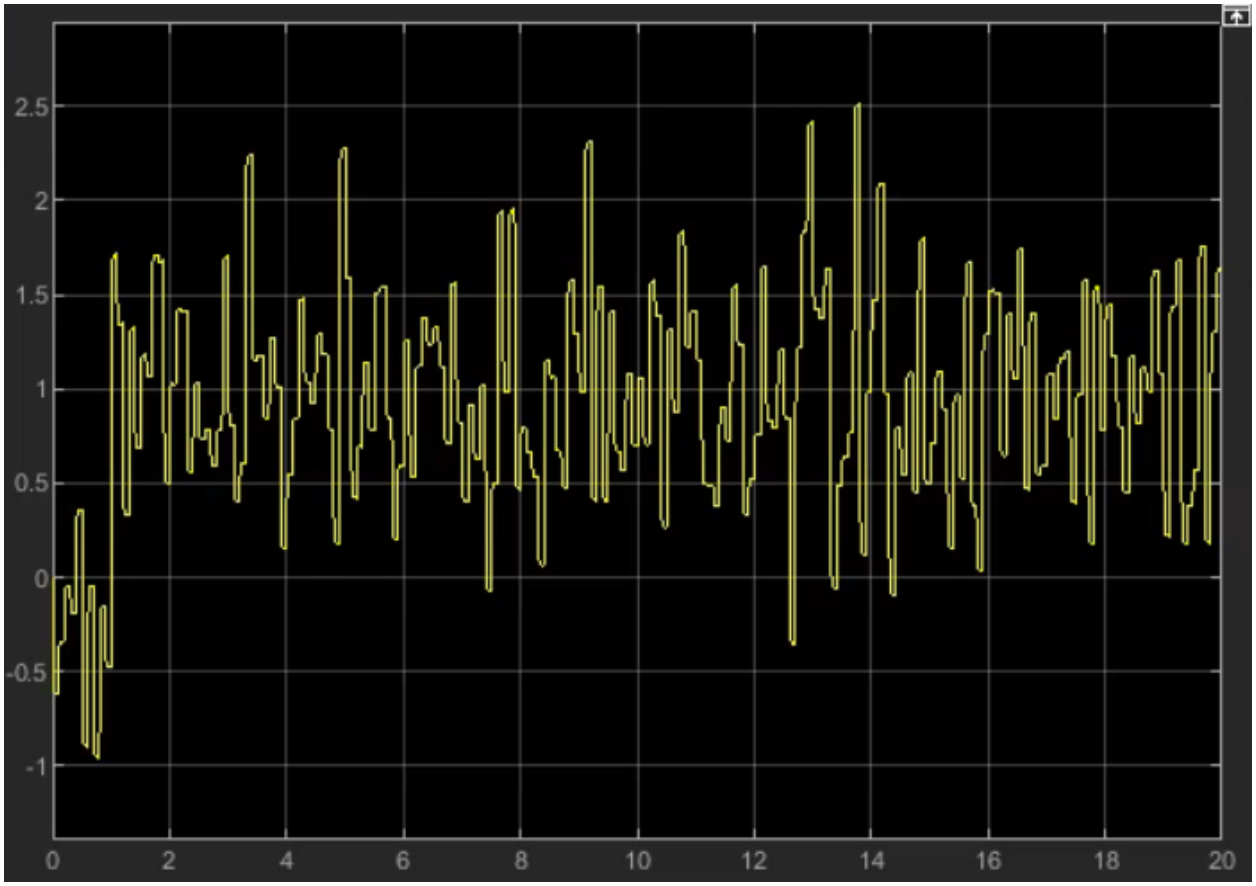


Figure 40: Step response for full PID control with 30% measurement noise

When we added noise to the engine/controller output instead, we chose much higher disturbances, because the engine force is on the order of 10^4. But with a variance of 10000 (max engine force is about 40000, and steady state force is <1000), we still had very strong noise resistivity, as can be seen in the step response below. This is likely because the weight and resulting inertia of the boat acts as a filter of high frequency engine noise.

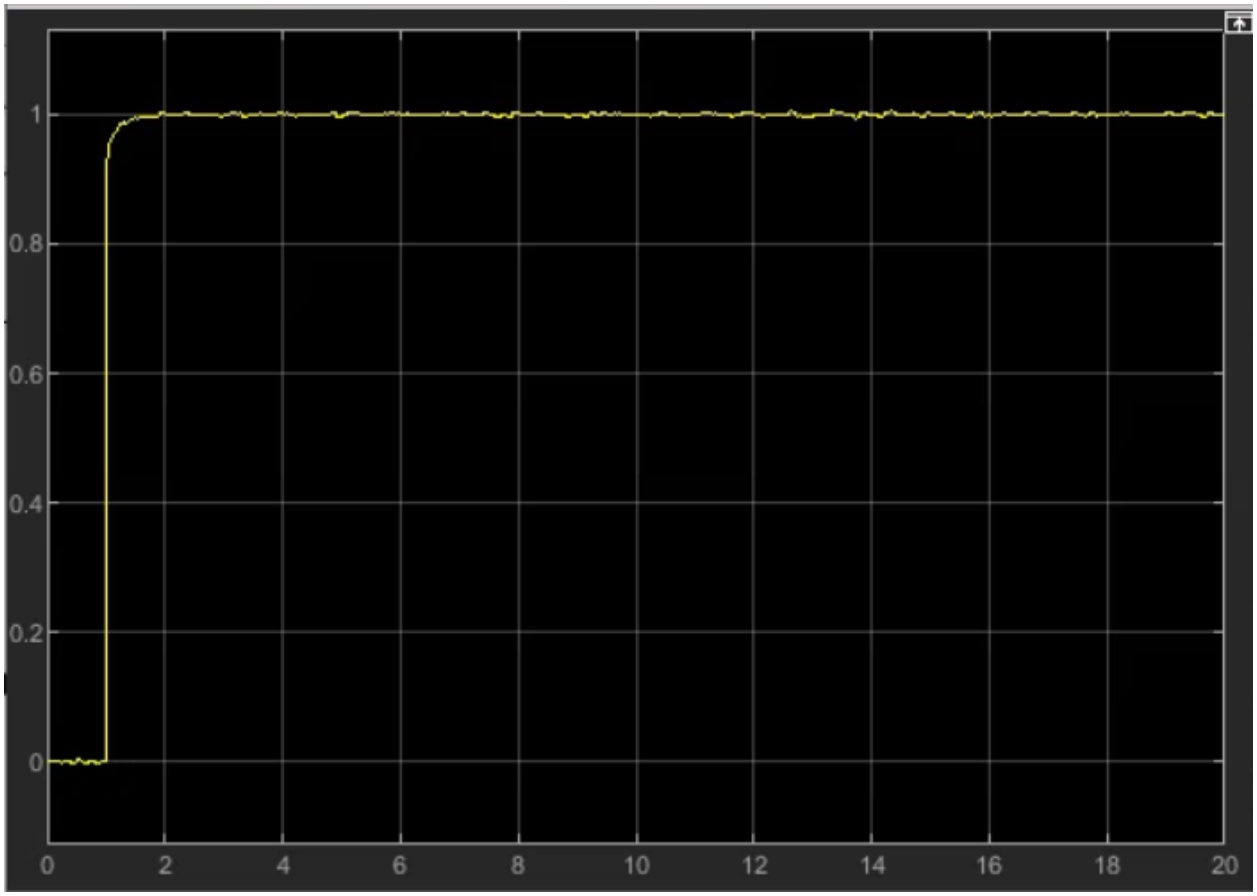


Figure 41: Engine noise with variance 10000 (25%-30% of max engine force)

Step 7: Simulating the closed-loop dynamics of the original nonlinear system with the designed PID controller

The process of simulating the closed-loop dynamics of the original nonlinear system with the controller we designed based on the linearization was easier if we transformed the controller transfer function into its time domain equivalent. Because the x- and y- components of the velocity are considered separate in our model, we could make our 1D controller into a 2D one:

$$U(s) = k_p + \frac{k_i}{s} + k_d s$$

$$u(e) = k_p e + k_i \int_0^\tau e(\tau) d\tau + k_d \dot{e}, \text{ where } e \text{ is error b/n target \& current state}$$

Our simulation will require use of an intermediary variable,

$$\mu = \int_0^\tau e(\tau) d\tau$$

.

$$\dot{\mu} = e = r - v$$

$$\ddot{\mu} = \dot{e} = -\dot{v}$$

The controller can now be expressed as:

$$u = k_p \dot{\mu} + k_i \mu + k_d \ddot{\mu}$$

We will augment the state variable, which included the x- and y- velocities, to include μ and its derivative as well.

$$q = \begin{bmatrix} v \\ \mu \\ \dot{\mu} \end{bmatrix} \implies \dot{q} = \begin{bmatrix} \dot{v} \\ \dot{\mu} \\ \ddot{\mu} \end{bmatrix}$$

Plugging in the controller equation into the expression for $\dot{v}$, and rearranging since the controller itself has dependency on $\dot{v}$, we obtain that the closed loop equations for $\dot{v}$ are:

$$\dot{v_x} = \frac{-\frac{\rho}{2M} A_x c_{d,x} v_x^2 \text{sign}(v_x) + \frac{k_p}{M}\dot{\mu_x} + \frac{k_i}{M}\mu_x}{1 + \frac{k_d}{M}}$$

$$\dot{v_y} = \frac{-\frac{\rho}{2M} A_y c_{d,y} v_y^2 \text{sign}(v_y) + \frac{k_p}{M}\dot{\mu_y} + \frac{k_i}{M}\mu_y}{1 + \frac{k_d}{M}}$$

More details can also be found in “ClosedLoop_Nonlinear_PID.mlx”. In this form, the nonlinear system is simulatable using ‘ode45’ in MATLAB.

Choosing a reference value of $v_x = 3\ m/s$ and $v_y = -3\ m/s$, we had the following behavior:

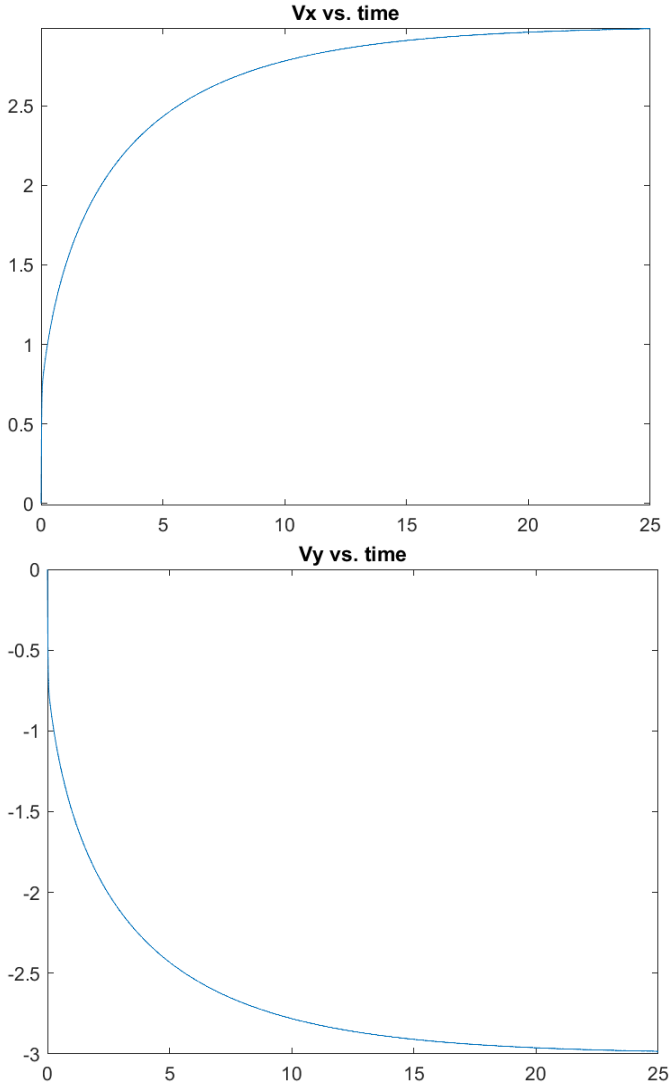


Figure 42: Controlled response of nonlinear system

We notice that the rise time is significant, though this is to be expected considering that the system in question is a boat accelerating in water, and physically the times are realistic. Looking at the graph zoomed out, it seems like the trajectory is a smooth curve towards the reference with no overshoot and oscillation. However, a closer look reveals some interesting behavior:

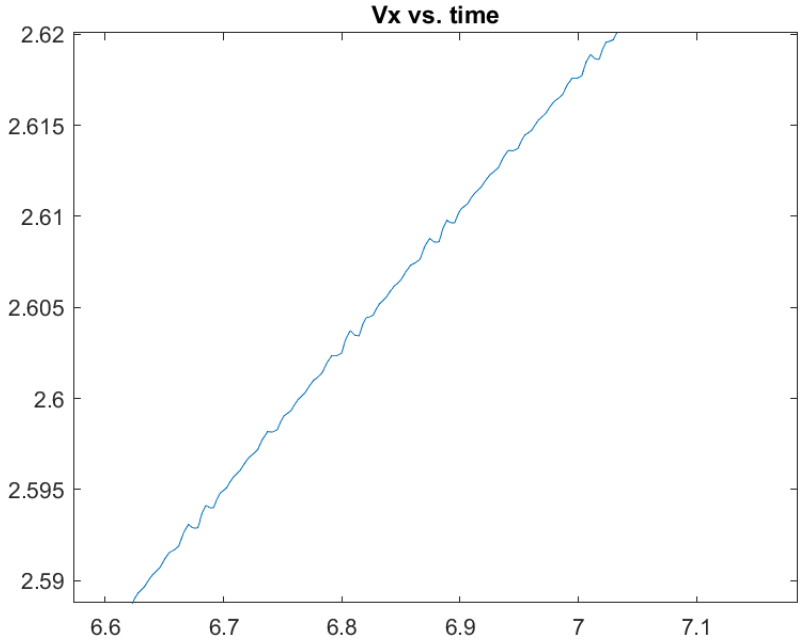


Figure 43: Numerical inaccuracies in controlled response

As we can see, the graph is noticeably jagged, and periodically so. While to get to within 5-10% of the target the system takes a physically acceptable amount of time, to reach the target velocity exactly does take a significant amount of time. The following graph's upward trajectory tells us that eventually the system will reach an x velocity of 3 m/s, but the time scale shows how long it will take:

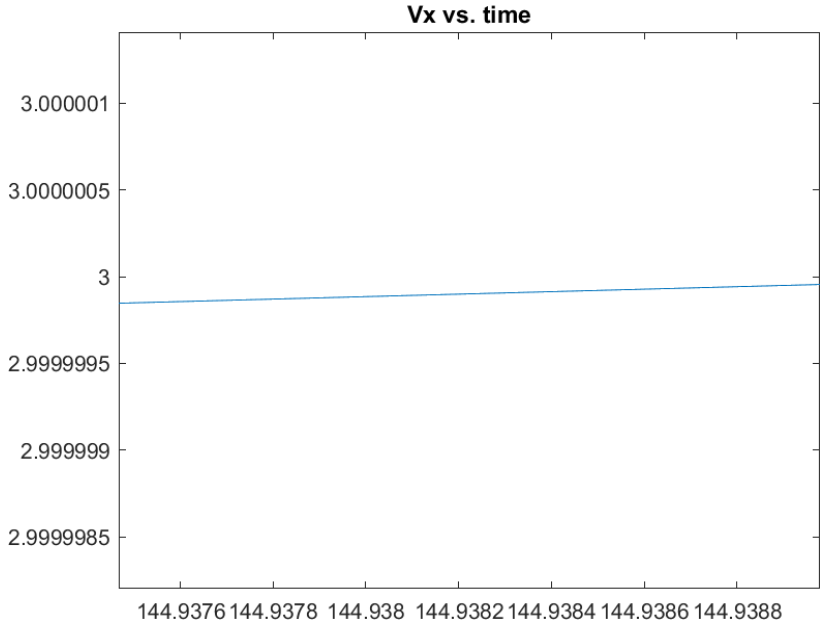


Figure 44: Inaccuracies in steady-state response

Running the simulation even longer, we notice very small overshoot and oscillation. We will not give this too much consideration, since the nonlinear system simulation requires discrete integration, and will never quite capture the true behavior of the continuous system anyway:

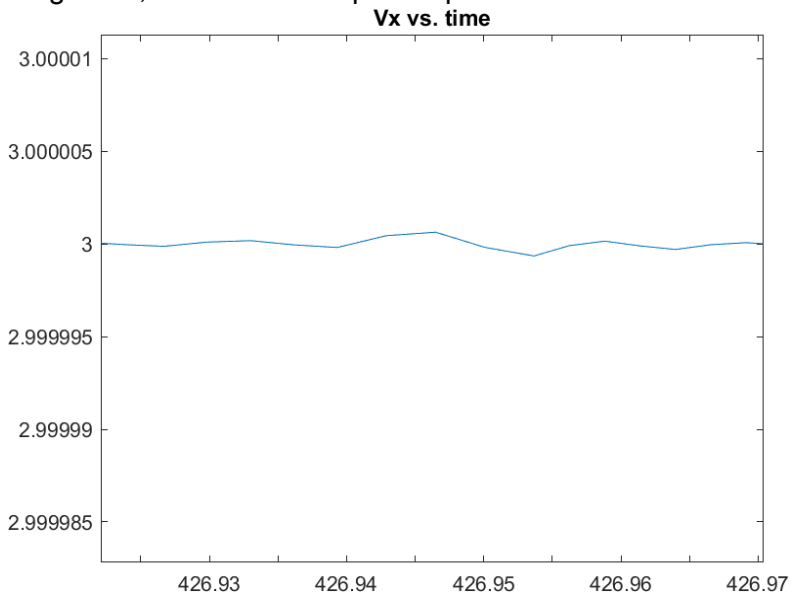


Figure 45: Inaccuracies in steady-state response

Another aspect of our controller we noticed was when the initial velocity was positive and the reference was greater and also positive, the controller brought the velocity down first very quickly before bringing it to the reference:

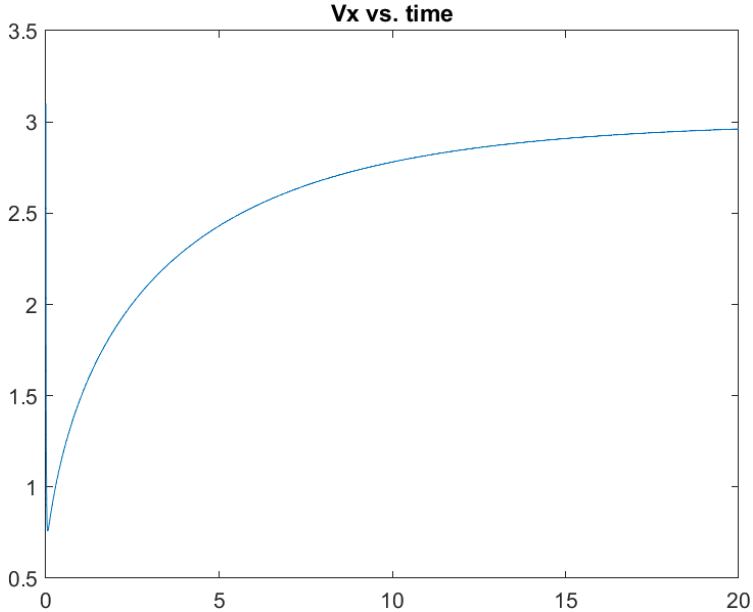


Figure 46: Unexpected transient behavior of initial system response

We suspect that this behavior is because the integral term starts as zero since no time has passed. Initially we thought that perhaps the derivative term would also have an effect, since there is a sharp change in error due to a given reference, but as we'll see when we give a new reference to a system that has been running for some time (thus no zero integral term), we

notice that there is no odd “dipping” behavior. The following graph is simulating the continuation of the same system as above, which was given a reference x velocity of 2 m/s. The time is in seconds after the first simulation:

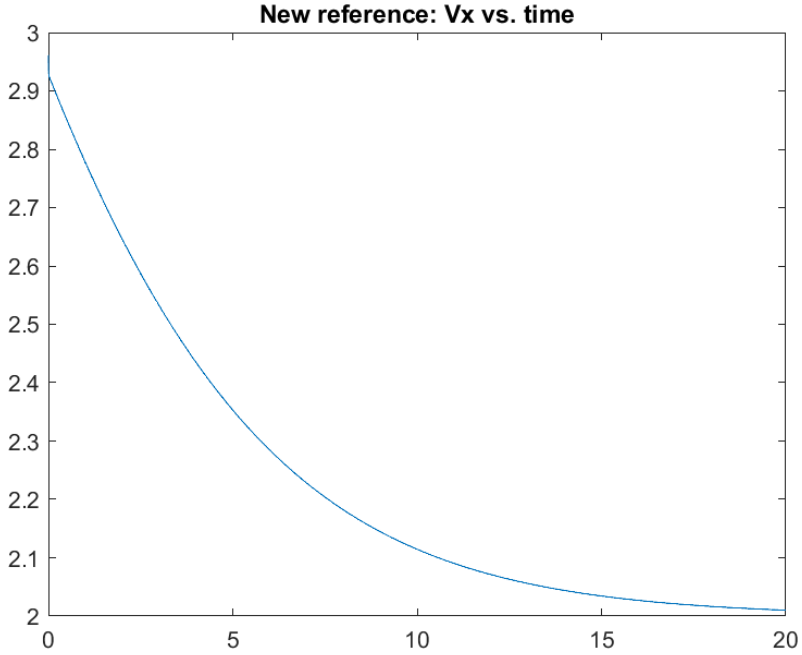


Figure 47: Long-term response of controlled system tracks reference well

Therefore, because the system now does not dip or pose other strange behavior before reaching the new reference, we assume that the controller needs to run for some time in order to accurately track a reference signal without unexpected/undesired behavior first.

Should a system like this be implemented concretely in hardware, design considerations would have to be made about how to deal with that sharp turn to zero before reaching reference if the system has not been running before. As it has been the case throughout this project, other considerations include whether the controllers we design are physically achievable (also whether they are within cost and other design constraints). Additionally, it would have to be further studied whether the model we have made use of is adequate in real life, and exactly what kinds of changes would need to be made to it in order for the controller built on top of it to achieve the control goal.

Bibliography

[1] “2020 U.S. Boating Statistical Abstract: Sailboat Sales Trends Now Available.” https://www.nmma.org/statistics/article/23731 (accessed Oct. 23, 2021).

[2] C. K. Fredrickson, “U.S. Coast Guard releases 2020 Boating Safety Statistics Report,” *Coast Guard Maritime Commons Blog*, Jun. 30, 2021. https://mariners.coastguard.blog/2021/06/30/u-s-coast-guard-releases-2020-boating-safety-statistics-report/ (accessed Nov. 21, 2021).

[3] R. Urquhart, “Know-How: Rigging Emergency Rudders,” *Sail Magazine*. https://www.sailmagazine.com/diy/know-how-rigging-emergency-rudders (accessed Oct. 23, 2021).

[4] “Rudder failure and 1,500 miles to sail – Yachting World.” https://www.yachtingworld.com/features/rudder-failure-1500-miles-to-sail-69460 (accessed Oct. 23, 2021).

[5] “Drag Coefficient.” https://www.engineeringtoolbox.com/drag-coefficient-d_627.html (accessed Oct. 22, 2021).

Contributions:

Work package	**Contributor**
Milestone 1: Problem description	Ezenia, (Eric in presentation)
Milestone 2: System modeling	Matthias, Arba
Milestone 3: Simulation	Arba, Eric
Milestone 4: Linearization	Arba
Milestone 5: Controller	Ezenia, Arba
Milestone 6: Estimator	Matthias
Milestone 7: Control design	
Step 1 - 3, 6	Matthias
Step 4 - 5	Ezenia
Step 7	Arba