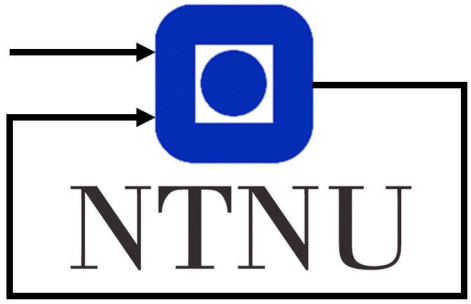
State estimation of a Maglev system with the Luenberger observer



Author:
Sverre Graffer

Supervisor:
Damiano Varagnolo

Co-Supervisor:
Hans A. Engmark

Specialization project
Department of Engineering Cybernetics
Norwegian University of Science and Technology

January 17, 2024

Preface

Several previous projects have been done by others on Maglev systems. This led to the point were it was reasonable to start testing an observer for this system.

Acknowledgements

Thanks to Damiano Varagnolo and Hans A. Engmark for supervising me during this project. It has been an exciting project where you have helped me whenever I got stuck with difficult problems.

Abstract

A Luenberger observer was developed for the Maglev system and tested in simulation. During this project, it was found that it was not easy to estimate the states in the system, and that the Luenberger observer, at least the one developed in this project, worked poorly. A more advanced observer than the Luenberger observer is probably needed to estimate the states of the Maglev system well enough to use the estimates for control.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Description
levmag	The levitating magnet
observer	Luenberger observer
LQR	Linear quadratic regulator
KF	Kalman filter

1

Introduction

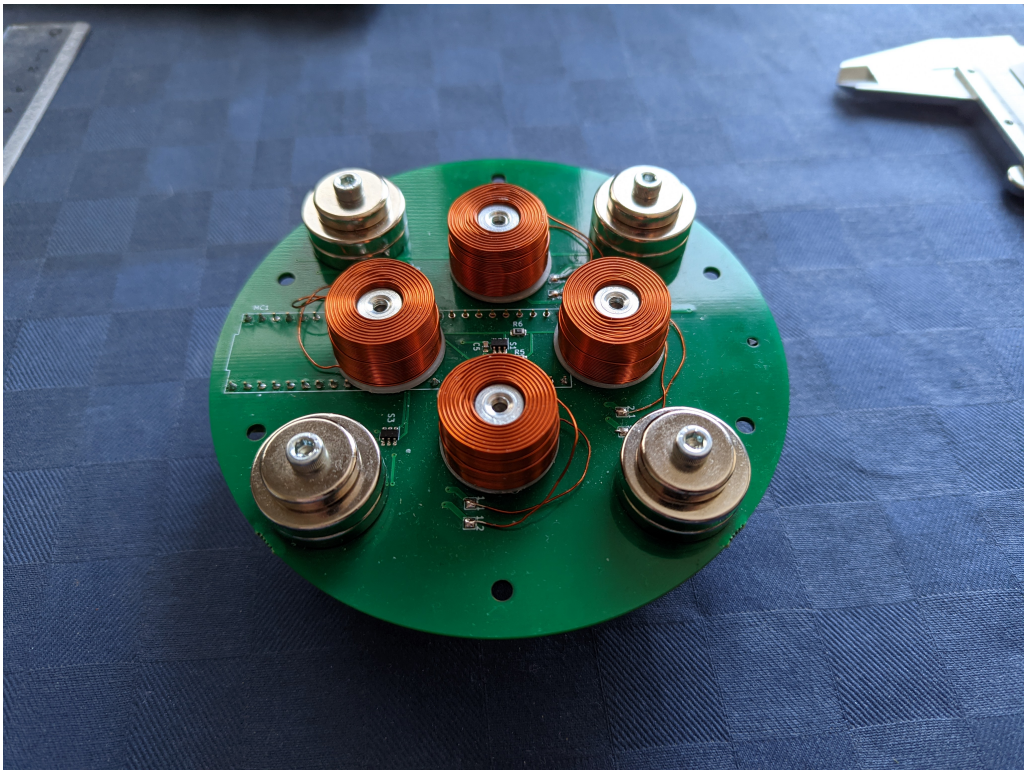


Figure 1.1: A picture of the real physical Maglev system

1.1 Background

This section will describe what has been provided at the start of the project.

1.1.1 Physical system

Previous work done by others have resulted in the design of a Maglev system. There exists two physical versions of this system that has been used during this project. They were very similar. The only significant differences were that the top layer of the permanent magnets were a little smaller than on the first version. Also, one solenoid connection was reversed by error in the first version. This wiring was done correctly in the second system. The system consists of these main components:

- Four permanent magnets that provideenough upwards force on the levmag to keep it floating a certain distance above the center of the platform without power.
- Four solenoids that can give extra magnetic field. The size of the magnetic field from these are controlled by the Teensy. They are necessary to stabilize the system.
- Three sensors that measure the magnetic field. One in the center, and two off center.
- The levmag which is a permanent magnet. This magnet should levitate approximately over the center of the PCB when the system is working as intended.
- A Teensy 4.1 microcontroller.

The Teensy microcontroller is used for reading the magnetic field from the sensors and setting the output current in the solenoids.

1.1.2 Mathematical model

Two nonlinear mathematical model of the system exists, Model 1 and Model 2. [1] Model 1 is more accurate than Model 2, while also being more computationally heavy. They describe the dynamics of the position and orientation of the levmag as a function of the states and the inputs. They also describe the measurements as a function of the states and the inputs. A Matlab implementation of this model was provided by Hans. This makes it possible to simulate the system, for example in combination with a controller or an observer.

1.2 Problem description

In the future it is desirable to be able control the physical system using state feedback controllers. Because the states are not measured directly, an observer is necessary to be able to use such controllers. The objective is to arrive at a state observer that can estimate the true states of the system. The goal is that this observer should provide state estimates that are good enough for such a controller to use.

1.3 Delimitations

This report will develop and test a continuous time Luenberger observer in simulation, both outside and inside the control loop. A discrete time Luenberger observer will be suggested for possible future use on the real physical system.

1.4 Structure of the report

The report will be contained in chapters named Method, Results, Discussion and Conclusion.

2

Method

The Luenberger observer is described in this chapter.

2.1 Continuous Luenberger observer for simulation

This section describes how to arrive at a Luenberger observer, starting from the provided nonlinear model.

2.1.1 Linearization

The nonlinear model of the real system is on this form:

$$\dot{x} = f(x, u) \tag{2.1}$$
$$y = h(x, u) \tag{2.2}$$

The state vector consists of these elements:

$$x = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma & \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix}^\top \tag{2.3}$$

The six first elements in the state vector x describes the position (x, y and z) and the orientation (α, β and γ) of the levmag. The six last elements are the time derivatives of the first six.

A linear model is needed to develop the observer. The model should therefore be linearized in the equilibrium point x_lp and u_lp:

$$x\_lp = \begin{bmatrix} 0 & 0 & z\_eq & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{2.4}$$
$$u\_lp = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{2.5}$$

The result of the linearization is a model on this form:

$$\dot{x} = A(x - x\_lp) + B(u - u\_lp) + \dot{x}\_lp \tag{2.6}$$
$$y = C(x - x\_lp) + D(u - u\_lp) + y\_lp \tag{2.7}$$

The provided Matlab script already contain functionality to calculate the matrices, A, B, C and D. Because the linearization happens in an equilibrium point, $\dot{x}\_lp$ can be removed:

$$\dot{x}\_lp = f(x\_lp, u\_lp) = 0 \tag{2.8}$$

The measurement in the equilibrium can be found like this:

$$y\_lp = h(x\_lp, u\_lp) \tag{2.9}$$

2.1.2 The continuous Luenberger observer

It is convenient to have a continuous Luenberger observer for simulation such that both the nonlinear model and the observer can be simulated at once in Matlab. This way, the observer can also be included as a part of the closed loop. These equations describe the continuous Luenberger observer:

$$\dot{\hat{x}} = A(\hat{x} - x\_lp) + B(u - u\_lp) + L(y - \hat{y}) \tag{2.10}$$
$$\hat{y} = C(\hat{x} - x\_lp) + D(u - u\_lp) + y\_lp \tag{2.11}$$

This is a continuous time system that can be simulated together with the nonlinear model. To use this, a suitable L-matrix must be chosen.

2.1.3 Determining the L matrix

Let the estimation error be:

$$e = x - \hat{x} \tag{2.12}$$

Then the dynamics of the estimation error is[2]:

$$\dot{e} = (A - LC)e \tag{2.13}$$

The observer tuning problem is to choose L such that the observer error dynamics are, as a rule of thumb, significantly faster than the controller dynamics. If the pair (C, A) is observable it should be possible to place the eigenvalues of $A - LC$ arbitrarily[3]. The PLACE function in Matlab can be used to place the eigenvalues of $A - LC$ at specified locations. Choosing 10 eiegenvalues that give good performance is not trivial. Therefore, it might be helpful to use the LQR function to create the L matrix. Even though this function is usually used for creating gain matrix for state feedback controllers, it can also be used to create the observer gain L matrix.

2.2 Observability of the system

Checking the observability of the nonlinear model is complicated. Instead, the observability of the linearized version is checked. The rank of the observability matrix is 10 which is not full rank. This means that the linearized model is not observable. If one looks closely

at the A matrix of the linearized system, one can see that state γ and $\dot{\gamma}$ forms an independent system. They don't affect the other states, and they are not affected by the other states. When looking at the C matrix, one can see that they do not affect the measurements directly. When looking at the B matrix, one can see that none of the inputs affects these states. This means that these states are both unobservable and uncontrollable in the linearized model. This suggests that removing these states from the linearized model will create an observable system. This is done, and new matrices, $A_{red}, B_{red}, C_{red}$ and D_{red} are created to be used for in the observer. This means that $\hat{x}$ will have 10 elements instead of 12 like this:

$$x = \begin{bmatrix} x & y & z & \alpha & \beta & \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} \end{bmatrix}^\top \tag{2.14}$$

In fact, this reduced model is already included in the provided script where it was used to create the LQR controller for the system.

2.3 Simulate the system and observer together

This section describes how to simulate the Luenberger observer together with the nonlinear model and also how to use the estimates for control.

2.3.1 Combine the systems

The Matlab function used to simulate the system, ode15s, takes a function of the form $\dot{x} = f(t, x)$. To create such a system that contains both the nonlinear model and the observer, it is necessary to create a new state vector that contains both the states and the state estimates. Here, it was done this way:

$$X = \begin{bmatrix} x \\ \hat{x} \end{bmatrix} \tag{2.15}$$

$$\dot{X} = F(t, X) = \begin{bmatrix} f(x, u) \\ A_{red}(\hat{x} - x\_lp) + B_{red}(u - u\_lp) + L(y - \hat{y}) \end{bmatrix} \tag{2.16}$$

$\hat{y}$ in 2.16 must be replaced by 2.11 and y must be replaced with $h(x, u)$. A controller function $u(x)$ or $u(\hat{x})$ must be defined such that ode15s know what the value of u is during simulation. The provided Matlab script already has an LQR controller which will be used here:

$$u(x) = -Kx \tag{2.17}$$

If the state estimates are to be used for control, this controller must be used instead:

$$u(\hat{x}) = -K_{red}\hat{x} \tag{2.18}$$

2.4 Discrete Luenberger observer for the real system

This section is about presenting a discrete version of the Luenberger observer, which are more suitable for use in the real system.

2.4.1 Discretization

A discretized version of 2.6 and 2.7 is needed. Because the term $\dot{x}\_lp = 0$, it will be removed from 2.6 before discretization. The result is this:

$$x[k+1] = x\_lp + A_d(x[k] - x\_lp) + B_d(u[k] - u\_lp) \tag{2.19}$$

$$y[k] = C(x[k] - x\_lp) + D(u[k] - u\_lp) + y\_lp \tag{2.20}$$

The matrices A_d and B_d can be found using c2d? in Matlab like this? Discretizing 2.7 does not change the expression. The C and D matrices does not change during discretization, so the same matrices as in 2.7 can be used.

2.4.2 The discrete Luenberger observer

For the real system it is convenient to have a discrete Luenberger observer because the real measurements are discrete. This is also relatively easy to implement, and no software for simulation is needed. These equations describe the continuous Luenberger observer:

$$\hat{x}[k+1] = x\_lp + A_d(\hat{x}[k] - x\_lp) + B_d(u[k] - u\_lp) + L_d(y[k] - \hat{y}[k]) \tag{2.21}$$

$$\hat{y}[k] = C(\hat{x}[k] - x\_lp) + D(u[k] - u\_lp) + y\_lp \tag{2.22}$$

3

Results

This chapter contains results from testing the Luenberger observer in simulation. First in a situation where the controller is using the states for control instead of the state estimates. This can be thought of as ”cheating” because the real states would not be available in the real system. The observer is tuned while simulating this way. Then the state estimates are used for control, and stability is tested for different initial values.

Because of space issues, the simulation plots were placed in the Appendix. A table describing the simulations that was done is here:

Table 3.1: Simulation table

Simulation number	Controller used	Initial values	Stable
1	States	$x\_lp + x_{offset}$	Yes
2	State estimates	$x\_lp + 0.3 * x_{offset}$	Yes
3	State estimates	$x\_lp + 0.4 * x_{offset}$	No

The appear in the same order in both the table and the appendix.

$$x_{offset} = \begin{bmatrix} 0.002 & -0.002 & 0.015 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{3.1}$$

The initial values for the observer was always equal to the equilibrium.

3.1 Observer tuning

The result of using two different strategies for calculating the L matrix is described in this section.

3.1.1 Placing eigenvalues

The observer eigenvalues was first attempted placed like this in Matlab:

Code snippet 3.1: Eigenvalue placement

```
obs_poles = [-3;-4;-5;-6;-7;-8;-9;-9.5;-10;-10.5;];
L = place(Ared.', Cred.', obs_poles).';
```

Several vectors of eigenvalues was tried, but they all resulted in this error message:

```
>> place(Ared.', Cred.', obs_poles).'
Error using place
The "place" command could not place the poles at the specified locations. Probable causes include:
* (A,B) is nearly uncontrollable
* The specified locations are too close to each other.
```

Figure 3.1: Error message when using place

This approach seemed hopeless, so a different approach was tried instead.

3.1.2 Using LQR for observer tuning

In an attempt to arrive at any L matrix, the LQR function in Matlab was used. Even though it is mostly used for control, it can also be used to calculate the gain matrix of an observer. This approach was tried:

Code snippet 3.2: Tuning the observer with LQR

```
Qobs = diag([1 1 1 1 1 1 1 1 1 1]);
Robs = eye(9);
L = lqr(Ared.', Cred.', Qobs, Robs).';
```

This succeeded in returning an L matrix, which allowed the observer to at least be tested.

3.1.3 Observer eigenvalues

The observer eigenvalues chosen by the LQR function can be viewed like this:

Code snippet 3.3: Displaying observer eigenvalues

```
eig(Ared - L*Cred)
```

This is the result:

Table 3.2: Table of observer eigenvalues (All values must be multiplied with 10^3).

-5.5964 + 0.0000i
-5.5961 + 0.0000i
-0.2806 + 0.0000i
-0.0000 + 0.5429i
-0.0000 - 0.5429i
-0.0000 + 0.5429i
-0.0000 - 0.5429i
-0.0061 + 0.0000i
-0.0002 + 0.0000i
-0.0002 + 0.0000i

As one can see, four of them is so close to the imaginary axis that they are written as 0 in this representation.

3.2 Simulating with the observer outside the loop

The simulations in this section does not include the observer as a part part of the closed loop, so the tuning of Q_{obs} and R_{obs} can be done without affecting the stability of the system. The goal here was to make the state estimates converge quickly to the states. These Q_{obs} and R_{obs} matrices seemed to give the best results:

Code snippet 3.4: Tuning Q_{obs} and R_{obs}

```
Qobs = diag([1e8 1e8 1e5 1e3 1e3  1e4 1e4 1e2 1e1 1e1]);
Robs = 5*eye(9);
```

3.2.1 Plot of states and state estimates

Simulation 1 in table 3.1 corresponds to this section. When looking at this plot it does not look very promising, considering that this is the best observer tuning that was achieved. The estimate of the position x, y, z uses significant time before it gets close to the corresponding states. α and β seems to do a little better, but their estimates have obvious oscillations that does not disappear. The time derivatives does not look any better. The estimates of, $\dot{x}, \dot{y}$, seems to not converge or converge very slowly.

3.3 Simulating with observer as part of the loop

In this part, the measurement are used to estimate the states, and then the state estimates are used to calculate a control input. This is the most interesting part of the results, because this part simulates what one would actually do on the real system.

3.3.1 Testing different initial conditions

When the same initial conditions used in Simulation 1 was used here, the controller was not able to stabilize the system. An attempt was made to see how close the initial conditions would need to be to the equilibrium before the controller could stabilize the system. The result was that the initial conditions in Simulation 2 was close enough, while the ones in Simulation three was not.

3.3.2 Plot of states and state estimates

Simulation 3 is clearly unstable. Simulation 2 shows that the estimates of x, y, z converge quicker than in Simulation 1. The oscillations in the estimate of α and β in Simulation 1 has here affected the actual α and β such that they are now oscillating. The estimates of $\dot{x}, \dot{y}$ seem to do somewhat better than in Simulation 1, but it does still not look great.

3.4 A second attempt at placing the eigenvalues

After finding a gain matrix L with LQR, it would be interesting to try to move the eigenvalues which are closest to the imaginary axis a little bit further to the left and see what the result is. The following Matlab script should be able to do it:

Code snippet 3.5: Eigenvalue placement

```
obs_poles = eig(Ared - L*Cred);
obs_poles = obs_poles - [0;0;0;6;6;6;6;0;6;6];
L = place(Ared.', Cred.', obs_poles).';
```

This resulted in an observer gain matrix L, with some very large elements. The largest one was $1.1187 * 10^9$.

4

Discussion

It is clear that the combination of the observer and the controller did not perform as well as hoped. This chapter discusses why this might be the case.

4.1 Bad observability

The place command often failed, mentioning that the pair $(A^\top, C^\top)$ might be nearly uncontrollable.

When using LQR, some of the observer eigenvalues ended up very close to the imaginary axis which implies slow convergence. This suggests that LQR did not want to place them further to the left because doing so would use too much ”input”.

When attempting to move some of the eigenvalues suggested by LQR, the L matrix got some very large elements.

All of this suggests that the linearized system is close to being unobservable and that moving some of the observer eigenvalues require a lot of gain. If an such an L matrix with high gain were to be used in a real system with measurement noise, the noise will be amplified and might cause trouble.

It is also possible that the nonlinear model is more observable than what the linearization in the equilibrium suggests. If this is the case, it would mean that the observability of the nonlinear system is not fully ”exploited” by the Luenberger observer because it is based on a linearization in the equilibrium.

Model 2 from [1] is the one that is used for everything. This is the least accurate of them, and it might be that Model 1 gives better observability.

4.2 Assuming that the system is linear

The term with the L-matrix works by affecting the derivative of the state estimates whenever the real measurement and the estimate of the measurements are not equal. When working properly, this term will over time move the state estimates to the true states such

that the estimate of the measurement is equal to the true measurement. The linearized measurement function will however not be equal to the nonlinear measurement function, $h(x, u)$, outside of the linearization point. This means that $\hat{y} = y$ might be true when $\hat{x} = x$ is false. This might explain why some of the state estimates in Simulation 1 did not converge until the actual state came somewhat close to it's equilibrium.

4.3 Bad L-matrix

Even though significant time was used in an attempt to tune the L-matrix, it might have been possible that there exists some other L matrix that would perform better, or even good.

4.4 The eigenvalues and the estimates

Some of the eigenvalues also had large imaginary components which is probably why there was very significant oscillation in some of the estimates. As seen in Simulation 2, this introduced oscillations to the actual states through the controller, which is unwanted.

4.5 Further work

A more advanced observer is likely needed, for example one of these:

- Normal linear KF.
- Unscented KF.
- Extended KF.
- Moving horizon estimator.

This will probably be needed:

- Implement model 2 described in [1] as functions for the Teensy.
- Need estimate of process noise covariance, to be able to use KF.
- Find ground truth to be able to test the observer on the real system.
- Maybe ”move” the processing from the microcontroller to PC?

5

Conclusion

A Luenberger observer has been tested and found to be barely able to estimate 10 out of the 12 states. This is only tested in simulation where there are no measurement noise. Based on the simulation results, it will likely not work very well on the real physical system. Tuning the observer proved to be difficult, and did not give the wanted results.

Feeding an LQR controller with state estimates from the Luenberger observer did only stabilize the system with initial values very close to the equilibrium. This indicates that this approach would not work well on the real system where significant measurement noise is present. However it might also work better (or worse) than expected because the nonlinear model used in the simulation is itself an approximation of the dynamics of the real physical system.

If one wants to estimate the states well enough to use the estimates for control, a more advanced observer than the Luenberger observer will likely be needed.

Bibliography

[1] H. A. Engmark and K. T. Hoang, "Modeling and control of a magnetic levitation platform," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 7276–7281, 2023, 22nd IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589632300705X

[2] Wikipedia, "State observer," 2023, last accessed 16 December 2023. [Online]. Available: https://en.wikipedia.org/wiki/State_observer

[3] MathWorks, "Pole placement," 2023, last accessed 17 December 2023. [Online]. Available: https://www.mathworks.com/help/control/getstart/pole-placement.html

Appendix

A Simulation plots

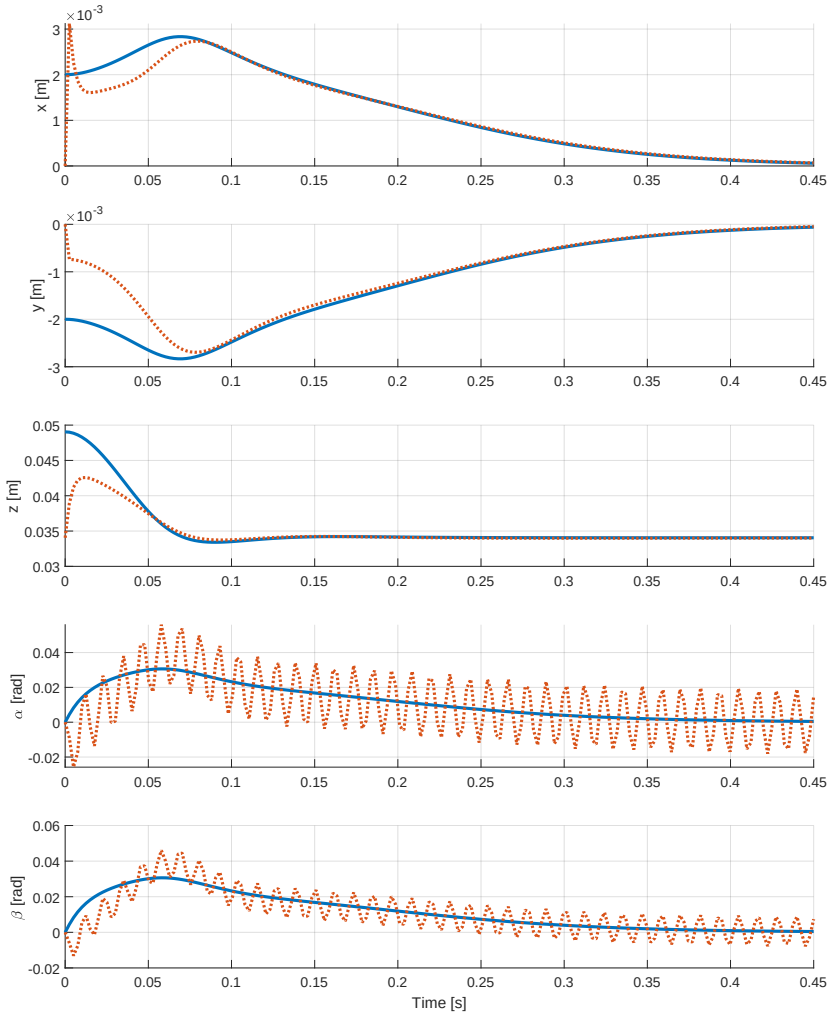
Simulation plots:

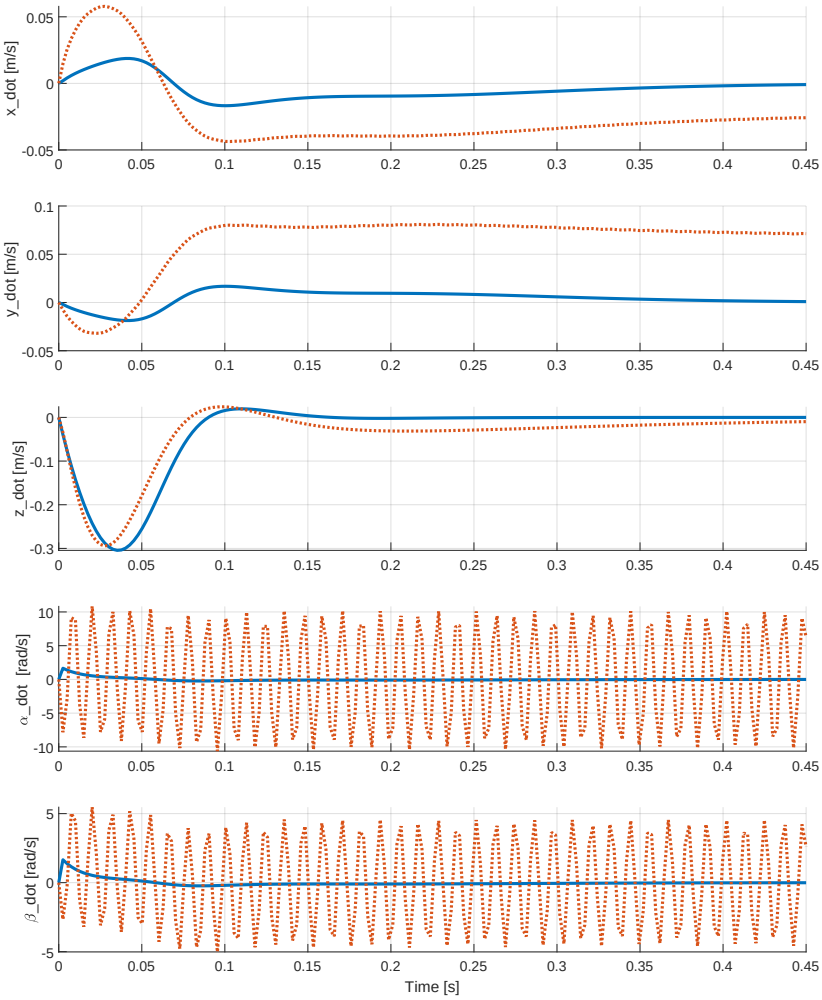
This part of the appendix contains plots of simulations used in the Results chapter. Blue lines represent the states, and the dotted orange lines represent the state estimates. The two states y and $\dot{y}$ are not shown in the plots, because they are only simulated and not observed by the observer. These plots are included here because they would take up a lot of space in the Result chapter.

The next two pages contain plots from a simulation where the observer is outside the closed loop, and these initial conditions are used:

$$x_0 = x_{lp} + x_{offset}$$

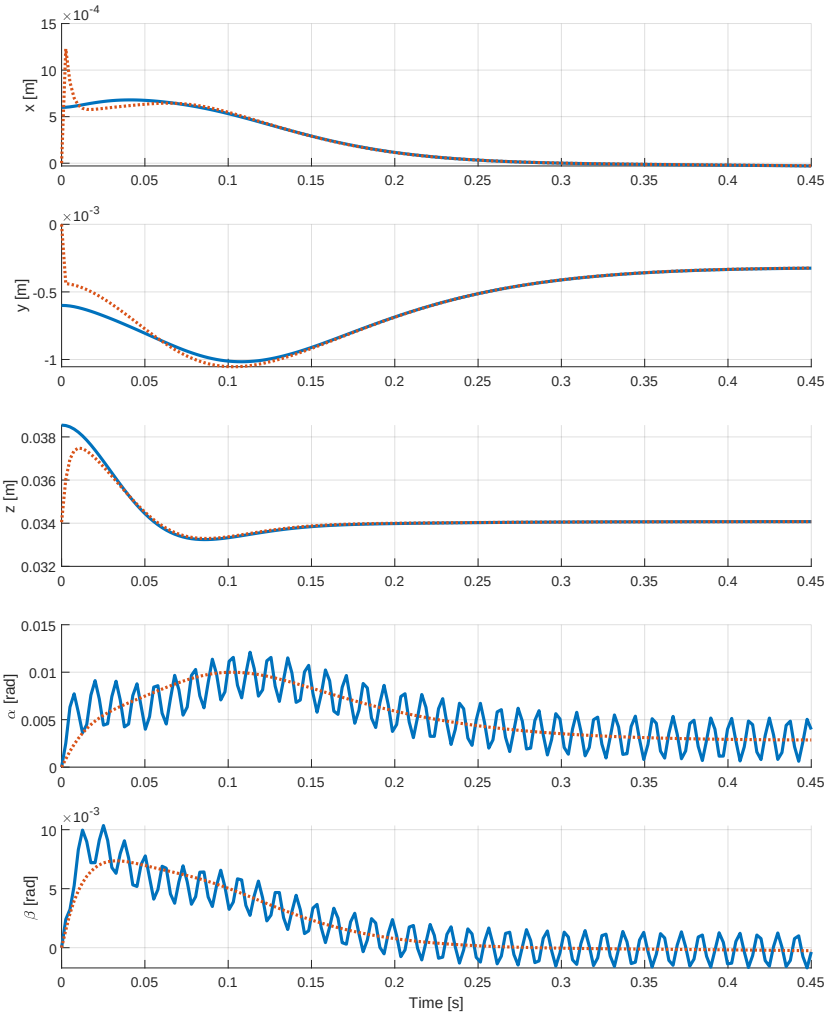
$$x_{offset} = \begin{bmatrix} 0.002 \\ -0.002 \\ 0.015 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

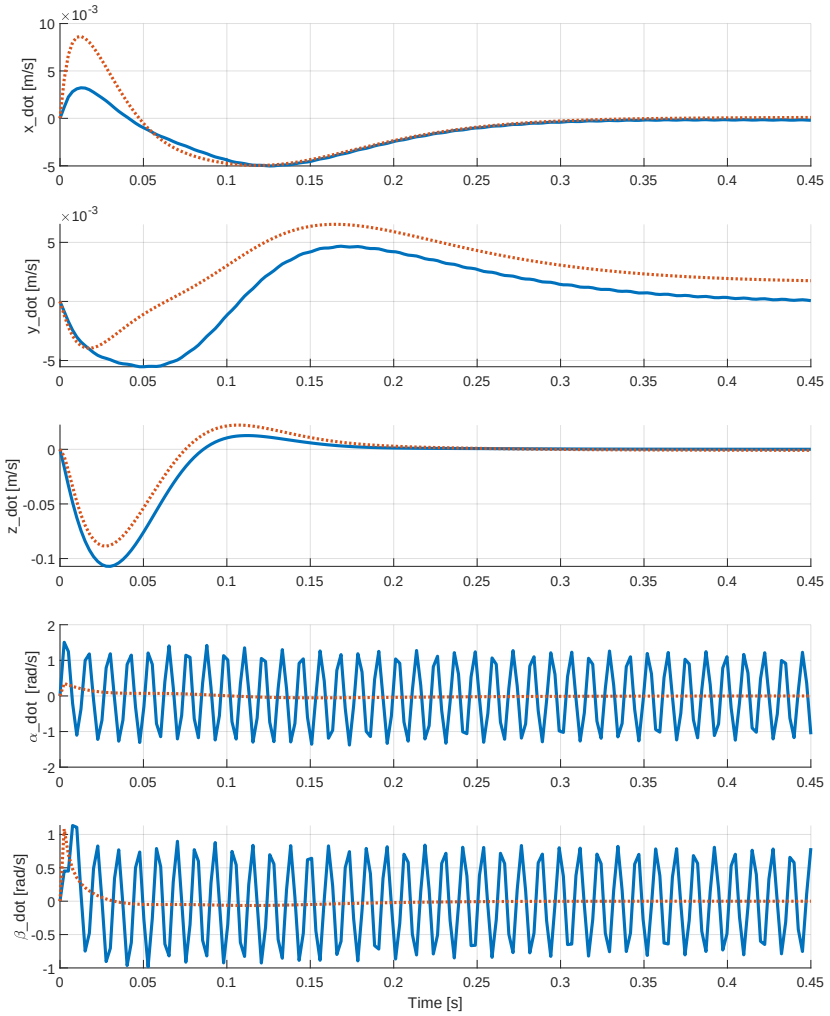

×10^-3
x [m]
×10^-3
y [m]
z [m]
α [rad]
β [rad]
Time [s]


x_dot [m/s]
y_dot [m/s]
z_dot [m/s]
α_dot [rad/s]
β_dot [rad/s]
Time [s]

The next two pages contain plots from a simulation where the observer is part of the closed loop, and these initial conditions are used:

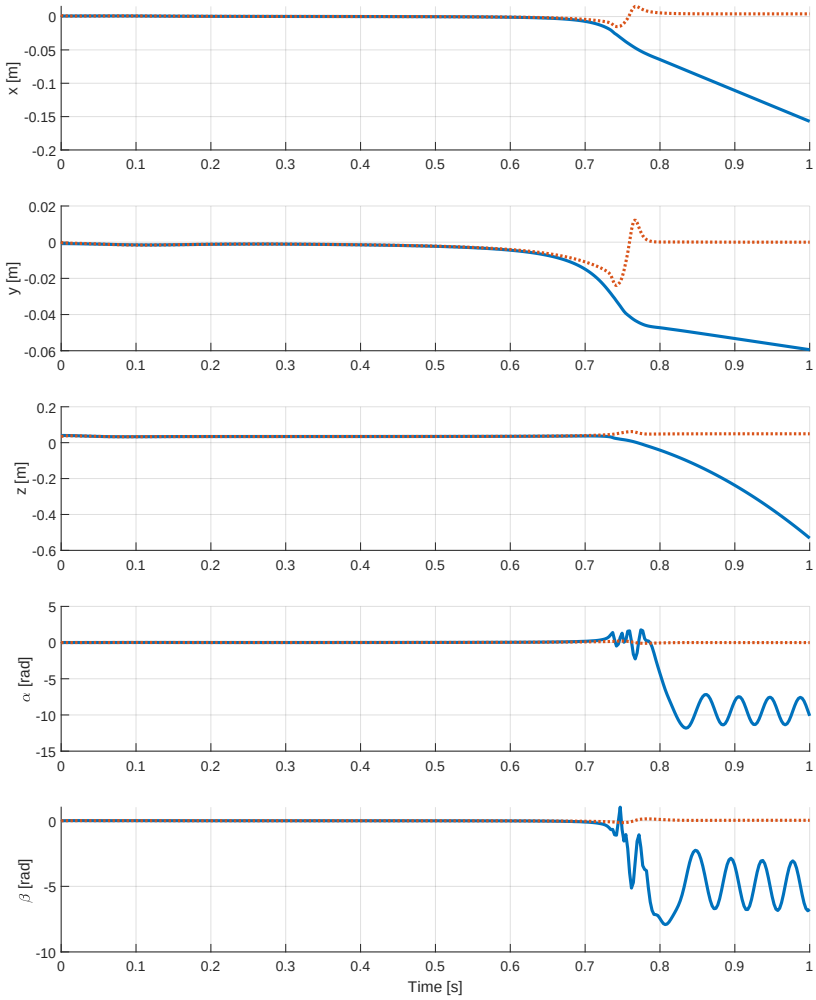
$$x_0 = x_{lp} + 0.3\, x_{offset}$$

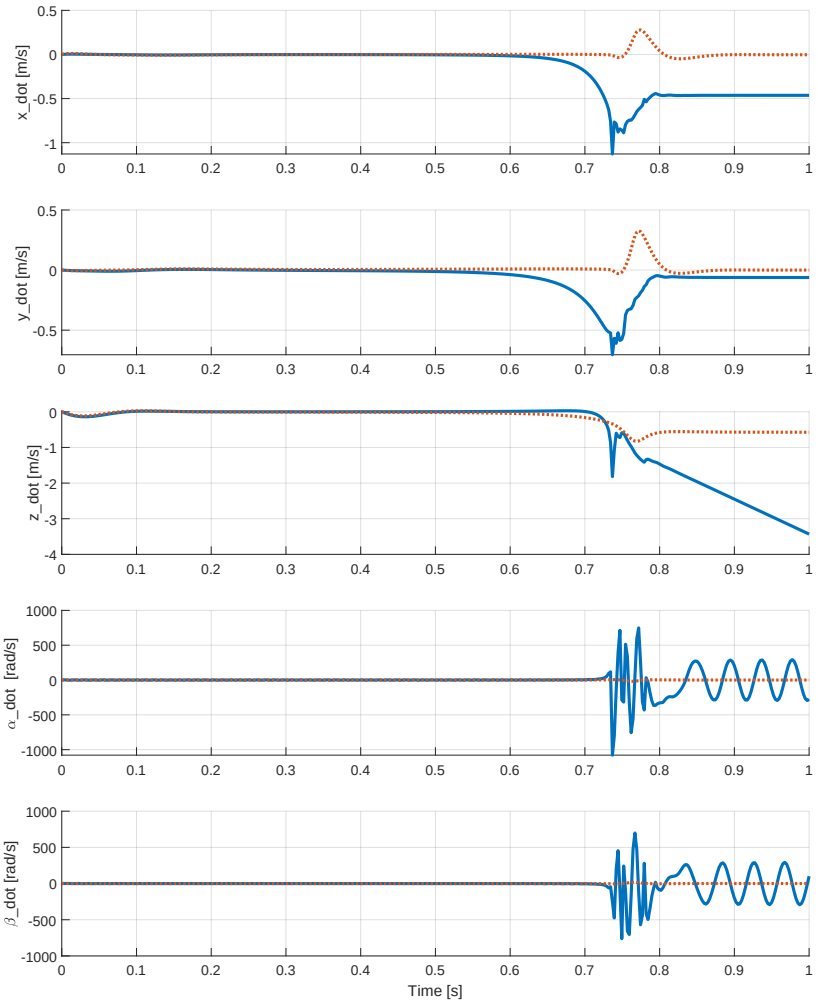

×10^-4
15
10
5
0
x [m]
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
×10^-3
0
-0.5
-1
y [m]
0.038
0.036
0.034
0.032
z [m]
0.015
0.01
0.005
0
α [rad]
×10^-3
10
5
0
β [rad]
Time [s]


×10^-3
x_dot [m/s]
y_dot [m/s]
z_dot [m/s]
α_dot [rad/s]
β_dot [rad/s]
Time [s]

The next two pages contain plots from a simulation where the observer is part of the closed loop, and these initial conditions are used:

$$x_0 = x_{lp} + 0.4\, x_{offset}$$


x [m]
y [m]
z [m]
α [rad]
β [rad]
Time [s]


x_dot [m/s]
y_dot [m/s]
z_dot [m/s]
α_dot [rad/s]
β_dot [rad/s]
Time [s]