Project Report:

Control of a Two-Wheel Self-Balancing Robot

MSE 483 – Modern Control Systems

Members – Student ID

Sepehr Rezvani – 301291960

[redacted]

Report Due Date: April 28th, 2021

Table of Contents

Introduction

Devices like Hoverboard, One-Wheel Scooter, E-Bike Electric Scooters are becoming more popular in the recent years. Goal of this project was to simulate a two-wheel self-balancing robot that is similar to Segway Professional Transporter. Due to the pandemic, the laboratories were closed, and no physical product was built. However, simulations are essentially applicable to physical products as well. Materials of Modern Control Systems course were used to achieve the following the objectives for this project: State-space formulation; Controllability and observability analysis; Apply state feedback control and state observers to the system; Analyze data collected from simulations, and suggest ways to improve the design based on data; Use computer tools (MATLAB, Simulink, Simscape Multibody) to apply design and test procedure to the design.

Modelling

The Self-Balancing Robot design is based on LEGO Mindstorms NXT. This section is based on a document written by Yorihisa Yamamoto, that is used to choose appropriate design variables [1].

Figure 1 shows structure of NXTway-GS. Since this project was done remotely, there are no physical components built. However, MATLAB and Simulink files can be used to balance the physical device once it is built. In this report, we DC Motor and sensors' values are important for optimizing simulation results.

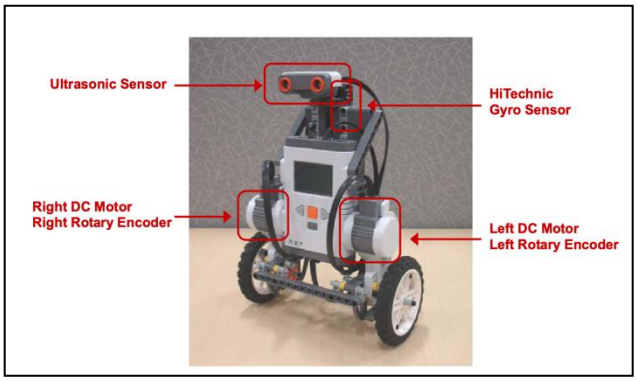

Figure 1 - NXTway-GS two-wheel robot design

Figure 2 and Figure 3 show side view and top view of the device, respectively. Table 1 includes parameters on the previous two figures.

Table 1 - Two-Wheel Robot Parameters

Parameter	Description	Unit
g = 9.81	Acceleration Due to Gravity	$\left[\frac{m}{s^2}\right]$
m = 0.03	Wheel Weight	$[kg]$
R = 0.04	Wheel Radius	$[m]$
$[J_w = \frac{mR^2}{2}]$	Wheel Inertia Moment	$[kgm^2]$
M = 0.6	Body Weight	$[kg]$
W = 0.14	Body Width	$[m]$

D = 0.04	Body Depth	m
H = 0.144	Body Height	m
L = $\frac{H}{2}$	Center of Mass Distance from Wheel Axle	m
$J_\psi = \frac{ML^2}{3}$	Body Pitch Inertia Moment	kgm^2
$J_\phi = \frac{M(W^2+D^2)}{12}$	Body Yaw Inertia Moment	kgm^2
$J_m = 1 \times 10^{-5}$	DC Motor Inertia Moment	kgm^2
R_m = 6.69	DC Motor Resistance	Ω
K_b = 0.468	DC Motor Back EMF Constant	$\frac{V \times s}{rad}$
K_t = 0.317	DC Motor Torque Constant	$\frac{N \times m}{A}$
n = 1	Gear Ratio	-
f_m = 0.0022	Friction Coefficient between Body and DC Motor	-
f_w = 0	Friction Coefficient between Wheel and Floor	-
ψ	Body Pitch Angle	rad
$\theta_{l,r}$	Wheel Angle (left and right)	rad
θ_{m_l,m_r}	DC Motor Angle	rad

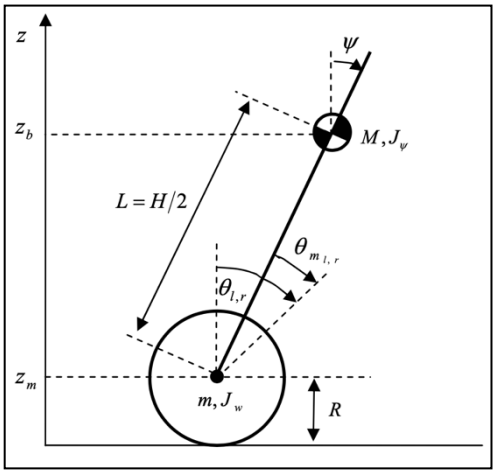

Figure 2 - Side view of two-wheel robot

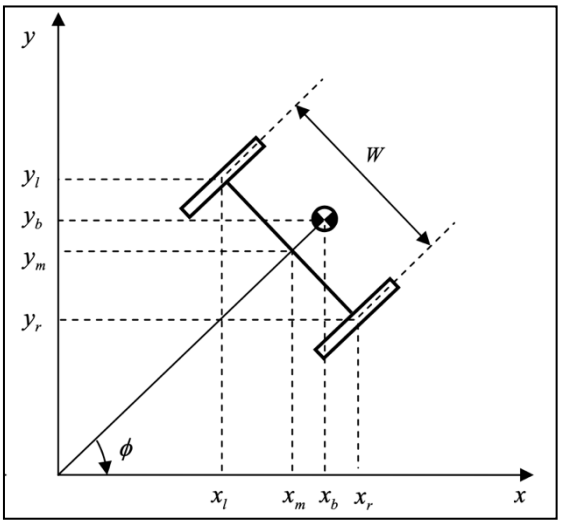

Figure 3 - Top view of two-wheel robot

Linearization

Once the equations of motion were derived for the two-wheeled self-driving robot and state equations were made, the linearization process was fairly simple to do. We linearized the state equations about a balance point of the robot (when $\psi \rightarrow 0$, $\sin(\psi) \rightarrow \psi$ & $\cos(\psi) \rightarrow 1$) and neglected second order terms. The result of this linearization are the following linear equations.

$$[(2m + M)R^2 + 2J_w + 2n^2J_m]\ddot{\theta} + (MLR - 2n^2J_m)\ddot{\psi} = F_\theta \quad (1)$$

$$(MLR - 2n^2J_m)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\psi = F_\psi \quad (2)$$

These equations can be expressed in the following form where the input, u, is the voltage of the electric motor which turns the robot's wheels.

$$E = \begin{bmatrix} (2m+M)R^2 + 2J_w + 2n^2J_n & MLR - 2n^2J_m \\ MLR - 2n^2J_m & ML^2 + J_\psi + 2n^2J_m \end{bmatrix} \tag{3}$$

$$\det(E) = E(1,1)E(2,2) - E(1,2)^2$$

$$\alpha = \frac{nK_t}{R_m}, \qquad \beta = \frac{nK_tK_b}{R_m} + f_m$$

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}, \quad \boldsymbol{x} = [\theta, \psi, \dot{\theta}, \dot{\psi}]^T$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A(3,2) & A(3,3) & A(3,4) \\ 0 & A(4,2) & A(4,3) & A(4,4) \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ B(3) \\ B(4) \end{bmatrix}$$

Where,

$A(3,2) = -MgLE(1,2)/\det(E)$

$A(3,3) = -[(\beta + f)E(2,2) + 2\beta E(1,2)]/\det(E)$

$A(3,4) = \beta(E(2,2) + 2E(1,2))/\det(E)$

$A(4,2) = -MgLE(1,1)/\det(E)$

$A(4,3) = [(\beta + f)E(1,2) + 2\beta E(1,1)]/\det(E)$

$A(4,4) = -\beta(E(1,2) + 2E(1,1))/\det(E)$

$B(3) = \alpha(E(2,2)/2 + E(1,2))/\det(E)$

$B(4) = \alpha(E(1,2)/2 + E(1,1))/\det(E)$

Analysis

Non-Linear System

The main purpose of designing a nonlinear model in Simulink is to understand the relationship between psi and theta. As mentioned previously, variable phi was ignored in Simulink simulation to lower complexity. Figure 4 shows the results of scope from $\theta, d\theta, \psi, d\psi$, in yellow, blue, red and green, respectively. The oscillation seen on Figure 4 is due to exchange of potential to kinetic energy between theta and psi. The Simulink model is designed based on Equation 4 and Equation 5 and an image of the open-loop block diagram can be seen below in Figure 4.

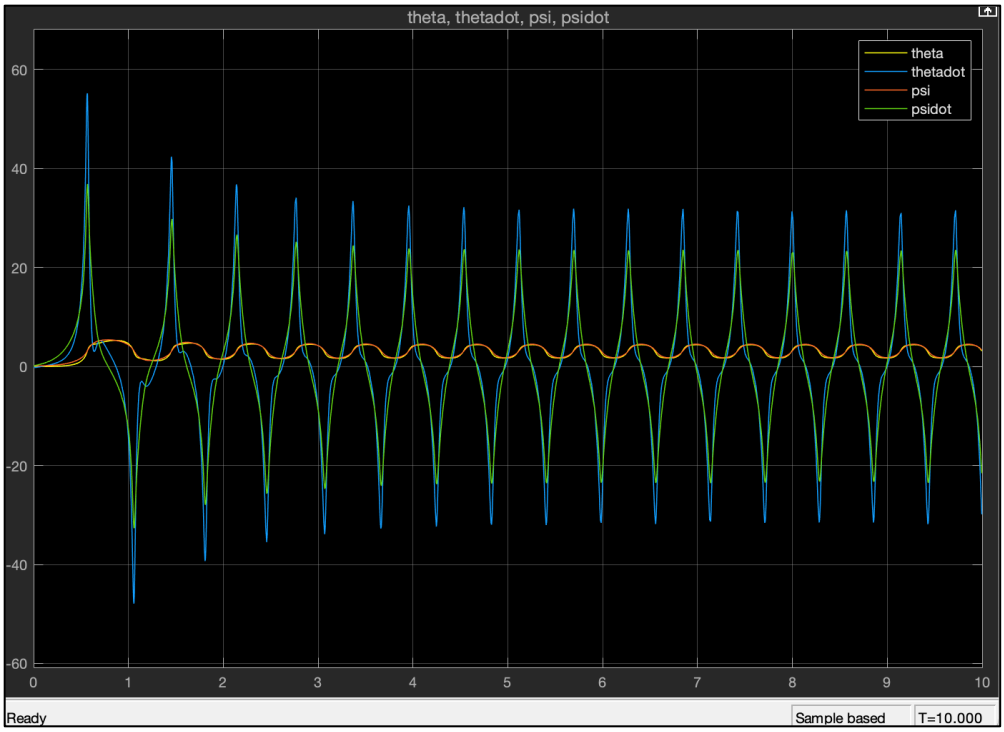

Figure 4: Theta, Psi, and their derivatives for the nonlinear model

$$[(2m + M)R^2 + 2J_w + 2n^2J_m]\ddot{\theta} + (MLRcos(\psi) - 2n^2J_m)\ddot{\psi} - MLR\dot{\psi}^2\sin(\psi) - [(2m + M)R^2\theta + MLRsin(\psi)]\dot{\phi}^2 = F_\theta \quad (4)$$

$$(MLRcos(\psi) - 2n^2J_m)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgLsin(\psi) - (MLR\theta + ML^2\sin(\psi))\dot{\phi}^2\cos(\psi) = F_\psi \quad (5)$$

Open-Loop Stability Analysis

By simply looking at an image of a two-wheeled self-balancing robot, we can expect that the system is very likely to be unstable when in an open loop (uncontrolled) form.

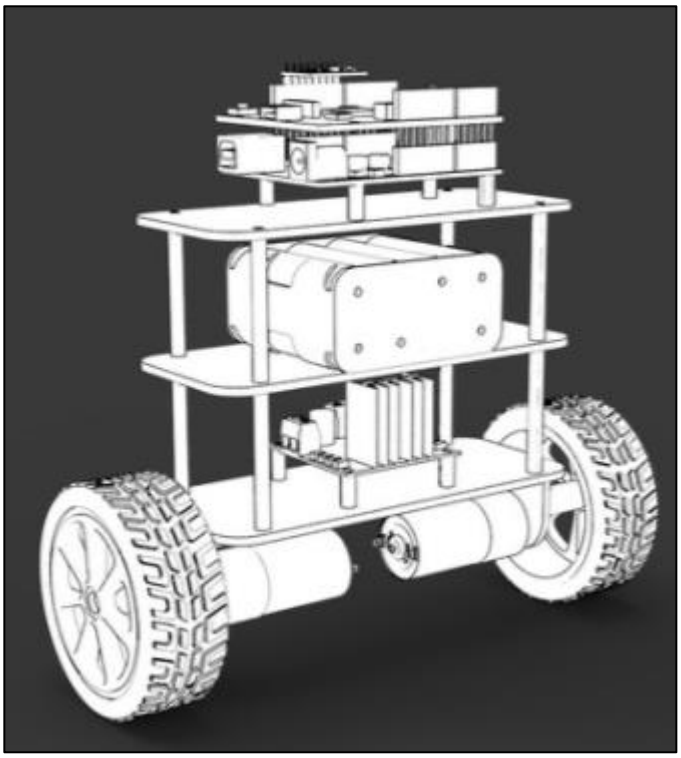

Figure 5

We can see that to stabilize the system, the robot would have to move in the same direction as its body pitch angle, ψ, to stay balanced. By taking the linear state equations found in section 2.0 of this report and creating a Matlab script with them, we found that the step response of the system is indeed unstable since all the state variables reach asymptotes and approach infinity.

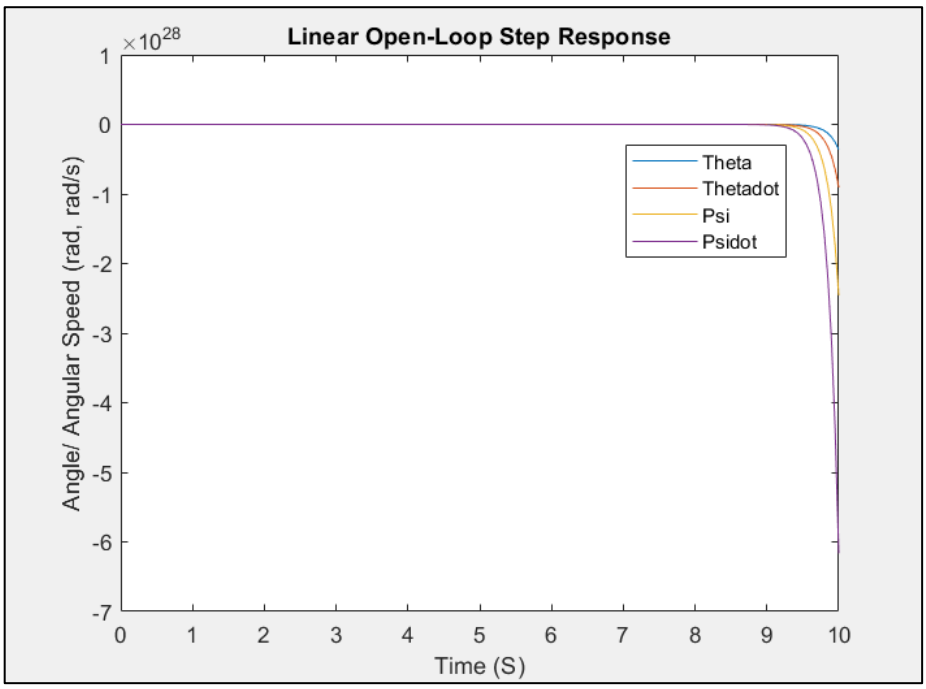

Figure 6

Further inspection of the system through the usage of Matlab's '*eig(.)*' function shows that the linear system has a positive eigenvalue, verifying the instability of the open-loop system.

```
>> eig(sys)

ans =

         0
 -160.6649
    6.8021
   -5.3505
```

Observability Analysis

The observability of each of the system's state variables was determined by first choosing a C matrix such that the output of the system is each of the state variables ($\theta, \psi, \dot{\theta}, \dot{\psi}$). This C matrix can be seen below.

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

To determine observability, the observability matrix, Q, was found and its rank was determined both by using the Matlab *rank(.)* function as well as by hand. The Q matrix as well as its rank as determined by Matlab can be seen below.

$$Q = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -409.7 & -104.6 & 104.6 \\ 0 & 269.6 & 54.6 & -54.6 \\ 0 & -409.7 & -104.6 & 104.6 \\ 0 & 269.6 & 54.6 & -54.6 \\ 0 & 71079.9 & 16658.5 & -17068.2 \\ 0 & -37080.9 & -8690.4 & 8960.0 \\ 0 & 71079.9 & 16658.5 & -17068.2 \\ 0 & -37080.9 & -8690.4 & 8960.0 \\ 0 & -11427333.0 & -2674612.9 & 2745692.8 \\ 0 & 5976473.7 & 1398341.1 & 2745692.8 \end{bmatrix}$$

```
>> rank(Q)

ans =

     4
```

As we can see, the rank of the matrix is 4 which is equal to the dimension of x and hence all of the state variables of the system are observable.

Controllability Analysis

Using the A and B matrices discussed in Yamamoto's paper [1] a controllability analysis can be found. Using state feedback controllability theorems, a system is controllable if its controllability matrix P shares the rank of A. This can be found by taking the determinant of P and verifying that it is non zero. A zero determinant for the P matrix implies redundancies of rows and there for linear combinations of rows. The P matrix is defined as follows:

$$P = [B \quad AB \quad A^2B \quad A^3B]$$

Using this equation, the matrix seen below was found.

$$P = \begin{bmatrix} 0 & 101.7 & -1619.1 & 2.6e+6 \\ 0 & -53.1 & 8446.6 & -1.4e+6 \\ 101.7 & -1619.1 & 2.6e+6 & -4.2e+8 \\ -53.1 & 8446.6 & -1.4e+6 & 2.2e+8 \end{bmatrix}$$

The determinant of this matrix was then found to be a non-zero value (-9.09e+10), leading to the proof that the system is controllable. This was later verified by using the controllability function in MATLAB (ctrb).

Design

Observer Design

Through the observability analysis it was determined that all state variables are observable. The A matrix transposed and C matrix transposed were deemed to be controllable and observable, thus any negative real eigenvalue can be chosen independently from the state feedback controller. The eigenvalues chosen for this observer are –400 repeated as it is close to 5 times greater than the state feedback control eigenvalues; This gives a fast system response. The observer system in Simulink uses inputs u and y with output x. The observer gain matrix is given as Ob and acts as the feedback loop for the system. As for connecting the observer into the feedback controller, the output of the state feedback controller U feeds into the observer again in conjunction with the Y output of the main systems. The output of the observer X goes into the state feedback controller.

```
Ob =
  400.0000         0    1.0000         0
         0  400.0000         0    1.0000
         0 -409.7180  295.3701  104.6299
         0  268.6273   54.5833  345.4167
```

Controller Design

The controller was designed by first determining a desired response for the output of the system. We decided to test two different cases and chose the percent overshoot to be 5% for both cases, and a step time of 1s and 4s for the first and second cases, respectively. To determine the eigenvalues of the system so that we could design the controller gain, we approximated our system by using a second order system with a known transfer function and found the eigenvalues of this second order system. The following equations show this process.

$$H(s) = \frac{{\omega_n}^2}{s^2 + 2\xi\omega_n s + {\omega_n}^2}$$

$$P.O. = 100\, e^{\frac{-\pi\xi}{\sqrt{1-\xi^2}}} = 5 \;\therefore\; \xi = 0.6906$$

$$t_s = \frac{4}{\xi\omega_n} = 1 \;\therefore\; \omega_{n1} = 1.498,\; \omega_{n2} = 5.792$$

After inputting these values to the transfer function, we were able to determine that our desired eigenvalues were:

Case 1: $[-4 + 3.22i \quad -4 - 3.22i \quad -40 \quad -60 \quad -120]$

Case 2: $[-1 + 0.805i \quad -1 - 0.805i \quad -10 \quad -15 \quad -100]$

Where the first two eigen values in each of the previous vectors were the eigenvalues determined through the second order transfer function, and the remaining 3 were added such that they were about 10x further in the left half plane. As we can see, a fifth eigenvalue was added such that we could use an integral based controller. This fifth eigenvalue was selected to be another 10 times further than the prior two based on what was seen in the notes, however when decreasing this value to around the same magnitude of the others a nearly identical result is achieved (small variations in initial overshoot, slight graph changes). This shows that if there is an interference with the observer at these high magnitude eigenvalues, then they can be decreased without a negative effect on the controller. This method of control requires us to find an additional K value (K_I) as well as matrices A_{CO} and B_O, these matrices were constructed in the following way.

$$A_{CO} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}$$

$$B_O = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

Where the C matrix was selected such that we could control the robot's wheel angle. This was chosen due to the fact that when C is monitoring any other variable, the controllability matrix using A_{CO} and B_O always results in an uncontrollable system. This is also referenced in the reference paper [1] where the engineers stated that theta was the only controllable variable under this method.

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

When equating the characteristic equation of the $A_{CO} - B_O K$ matrix to the desired characteristic equation built from the eigenvalues we determined, we determined the following K-values for each of our two cases.

Note: the 5th value of each matrix is the K_I value

Case 1: $K = [-472.2 \quad -1215.4 \quad -96.9 \quad -187.1 \quad 1336.2]$

Case 2: $K = [-6.0 \quad -71.4 \quad -4.7 \quad -8.4 \quad 4.3]$

With these K values and an r value of 1 and 1000 for case 1 and 2 respectively, the following outputs were returned.

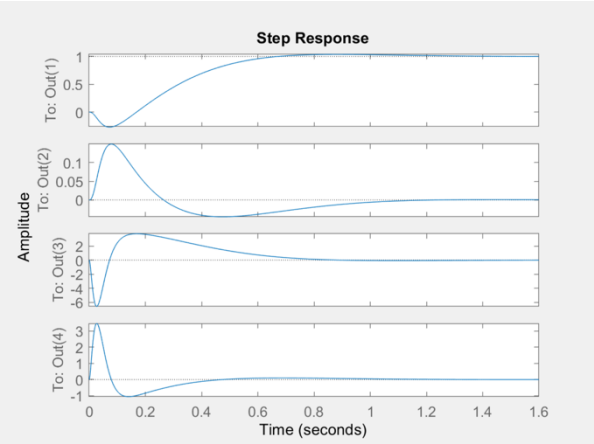

Figure 7: Case 1 Matlab Script Results (x1 = θ, x2 = ψ, x3 =θ̇, x4 =ψ̇) in degrees and degrees per second

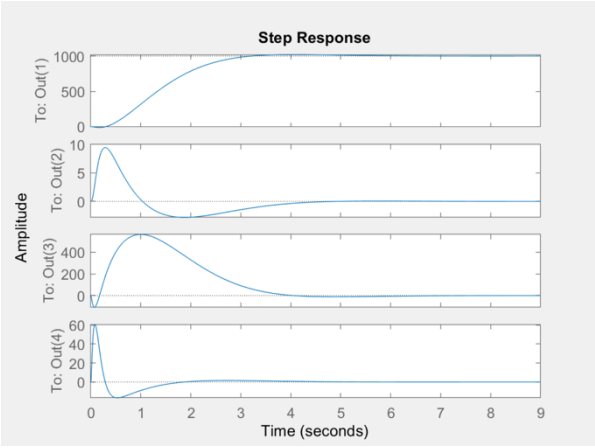

Figure 8: Case 2 Matlab Script Results (x1 = θ, x2 = ψ, x3 =θ̇, x4 =ψ̇) in degrees and degrees per second

In Figure 7, the initial step input of 1 is used as the reference point. The system stabilizes within 1 second with minimal overshoot for the controlled variable of θ. This shows that the steady state correction works in eliminating the steady state error. With Figure 8, case 2 is applied and a much larger

theta displacement can be reached with little change in the body tilt angle of the robot. This shows that with larger settling times, greater velocities can be reached with little instability of the system.

As an aside, this is most likely how the robot in the reference is designed [1] by changing the settling time to be some value which allows for a steady velocity while also keeping the body upright.

Finally, when the reference is scaled, every variable is scaled. Therefore, if the controller from case 1 was used to achieve the 1000-degree change seen in case 2, the change in tilt angle would be about 100 degrees, which obviously would mean the robot would fall over. This goes back to what was mentioned previously, where the engineers from the reference most likely modify their controller based on how long the input is applied to correct the stability accordingly.

Simulation Results

The following figures are taken from scopes of the observer-based feedback controller in Simulink without the scaled reference variable. These graphs demonstrate both the state feedback controller and the observer functioning when a step input is applied. Slight differences are noticeable near the peaks and troughs of the graphs, but they are minimal errors between them.

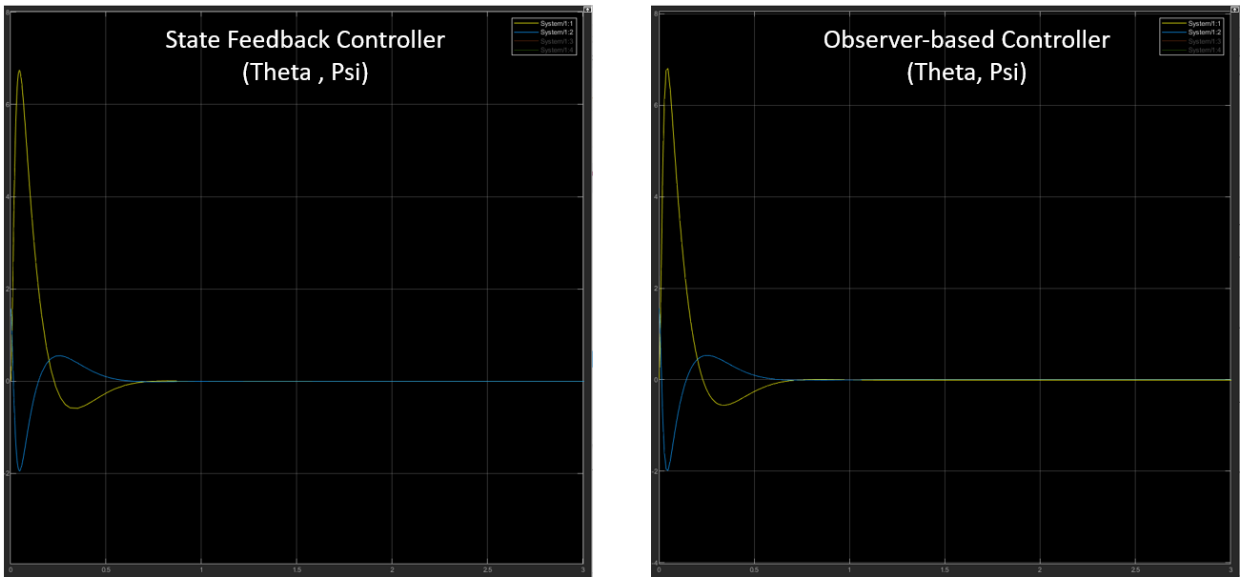

Figure 9: State feedback controller system output (left) and observer-based controller system output (right)

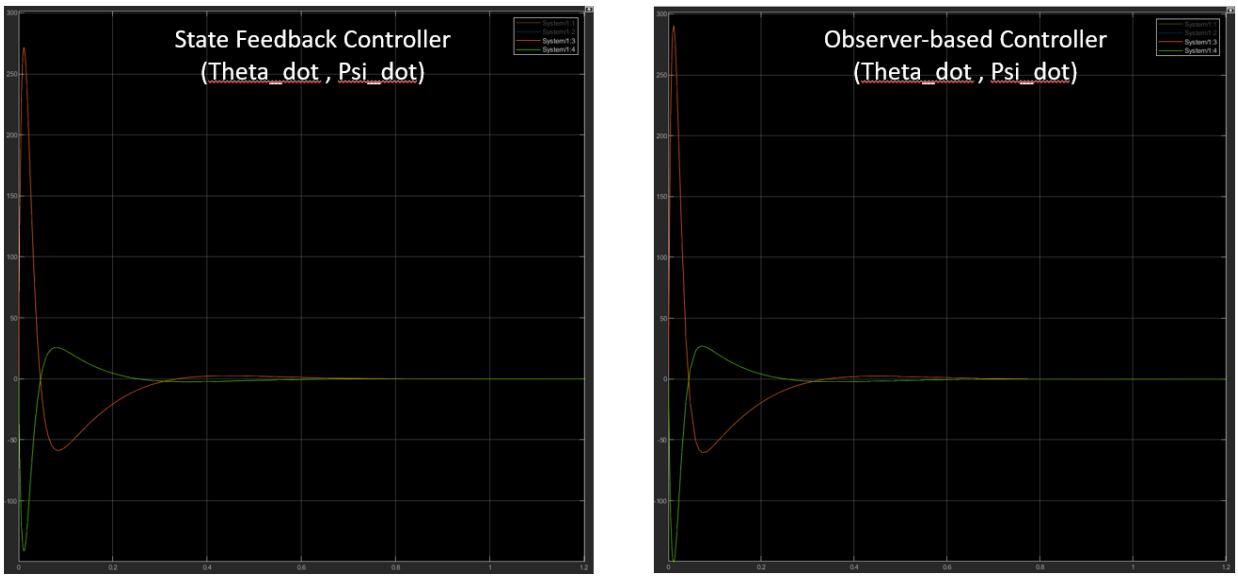

Figure 10: State feedback controller system output (left) and observer-based controller system output (right)

Furthermore, the controller's effect on the originally unstable system is apparent, as all values settle near zero. There should be a steady state correction as was seen in the controller design section earlier, however implementation of this into the Simulink model caused problems. This is most likely due to an incorrect integration of this controller into the system, since it works on its own in the theoretical MATLAB simulations. With that being said, if the controller with its steady state error correction was inserted into the system correctly, we could expect the Simulink output to look very similar if not identical.

Conclusions

Overall, the steps taken to create a self-balancing robot were taken and completed successfully, with some minor inconveniences in complete integration of the design. Firstly, the state space system was created and linearized based on the paper [1] and set up the framework for the rest of the project. Next, it was proven that the two wheeled robot was unstable mathematically. Although this was intuitive, the mathematical representation of this proved theory to reinforce the conclusion. The system was then checked for its controllability and observability before embarking on an endeavour that may have led nowhere. Once this was proven, both the controller and observer were designed to output the response that was expected. It was only once this got to the integration of the steady state error corrector component of the controller that issues arose. With all of the above steps that were taken however, a controlled stable system where all values come to stability at zero was produced in Simulink, which is an improvement on the system without any type of control. Finally, the concepts from the course were all utilized in this class and adequately displayed a real-life application of the material.

References

[1] Y. Yamamoto, "NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT -," 03 03 2008. [Online]. Available: http://www.pages.drexel.edu/%7Edml46/Tutorials/BalancingBot/files/NXTway-GS%20Model-Based_Design.pdf. [Accessed 01 02 2021].