MECHENG 561 Final Project

Valerie Chen, Jonathan Zarger

April 17, 2018

1 Abstract

The plant we controlled for this project was a model of a Segway. A Segway is a personal transportation vehicle that can use two wheels to balance, move forward, and turn. A rider can make it translate forward or backward by leaning in those respective directions. The Segway is a different presentation of the inverted pendulum problem. We constrained the system to one dimensional lateral motion (no turning) to simplify the model and implemented closed loop control to stabilize the system. The plant input was voltage to the wheels and the plant outputs were the rider angle and Segway lateral position. Disturbances we considered were voltage disturbances and torque disturbances applied to the motors.

We simulated the model and two different controllers with Simulink and used MATLAB plots to show the results. The first controller was an inner-loop/outer-loop architecture that was robust to voltage disturbances but had a long settling time and a high rise time. The second controller was a LQR controller/estimator/integrator augmentation system that provided a stable and reasonably fast response that rejected disturbances (both voltage and torque) and noise.

2 Introduction

The Segway PT was invented by Dean Kamen and publicly released in December of 2001. It was anticipated to revolutionize personal transportation, but fell short of this expectation. Regardless, Segways are still used today by police forces, emergency medical services, and occasionally by the general public as personal transportation devices. [1]

As mentioned before, the Segway is a realization of a classic example used in dynamics and controls problems: the inverted pendulum. An inverted pendulum is a pendulum that has a center of mass above its pivot point, making it an inherently unstable system that requires active balancing by moving the pivot or applying a torque to the pendulum at the pivot point. [2] In the case of a Segway, active balancing is achieved by moving the pivot point, and this is how a rider can make a Segway translate forward or backward by leaning to shift their center of mass.

Lots of work has been done in the past surrounding the stabilization of inverted pendulum systems, but much of this is experimental as opposed to utilizing a model-based design approach. In addition, much of this design assumes a continuous-time as opposed to a discrete-time system.

In this paper, we first outline a simplified dynamic model of a Segway in Section 3, Problem and Modeling. Next, we develop two discrete-time controller architectures to stabilize the system in Section 4, Solution and Discrete Controller Design. We simulate the controlled system and discuss our results in Section 5, Results and Simulation, and then give a brief recap of our findings in the final section.

3 Problem and Modeling

For the purposes of this project, we consider the Segway system dynamics derived in [3]. For the dynamic model of the Segway, we first consider a simplified state space model of a DC motor.

$$\begin{bmatrix} \dot{i} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{R}{L} & \frac{k_e}{L} \\ \frac{k_m}{I_R} & \frac{-k_e}{I_R} \end{bmatrix} \begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} & 0 \\ 0 & -\frac{1}{I_R} \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix} \text{ [3]}$$

In this model, i is the motor current, ω is the motor speed, R is the terminal resistance, L is the terminal inductance, k_e is the back emf constant, k_m is the motor constant, I_R is the inertia of the motor, V_a is the applied voltage, and τ_a is the load torque applied to the motor.

Next, we consider the free body diagram of one of the Segway wheels in Figure 1.

In the figure, θ_w is the angle of rotation of the wheel, P_R and H_R are the vertical and horizontal forces on the wheel from the rider/handlebars, respectively, M_w is the mass of the wheel, C_R is the torque applied to the wheel by the motor, and H_{fR} is the force of friction on the wheel. Note in this model that weight force and normal force of the wheel are neglected because the wheel does not accelerate vertically.

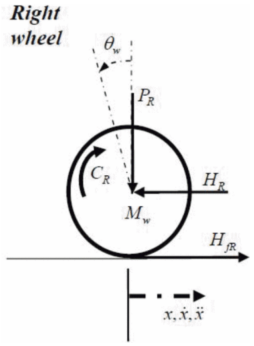

Figure 1: Free body diagram of Segway wheel

Using Newton's second law, we can say that

$$\Sigma F_x = M_w \ddot{x} = H_{fR} - H_R \quad (1)$$

$$\Sigma M_o = I_w \ddot{\theta}_w = C_R - H_{fR} r \quad (2)$$

as shown in [3]. In Equations 1 and 2, ΣF_x is the sum of the forces in the x-direction (as defined in the figure), ΣM_o is the sum of moments about the center of the wheel (positive clockwise convention), I_w is the moment of inertia of the wheel, and r is the radius of the wheel.

We also consider the free body diagram of the simplified mass representing the rider and the handlebars of the Segway in Figure 2.

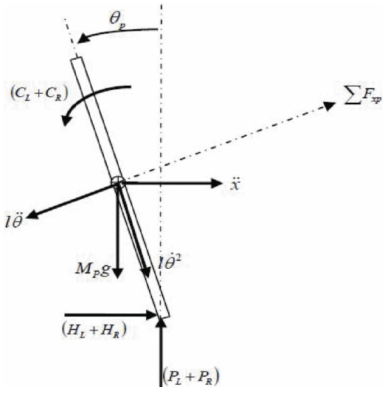

Figure 2: Free body diagram of simplified mass representing rider and Segway handlebars

In the figure, θ_p is the angle of the mass with respect to the vertical, C_L and C_R represent the torque applied by the left and right motors, respectively, l is the length, M_p is the mass, and H_L and P_L are the horizontal and vertical forces on the mass from the left wheel.

Applying Newton's second law along the ΣF_{xp} direction as shown in the figure, we can write

$$\Sigma F_{xp} = M_p \ddot{x} \cos(\theta_p) = (H_L + H_R)\cos\theta_p + (P_L + P_R)\sin\theta_p - M_p l\ddot{\theta}_p - M_p g \sin\theta_p \quad (3)$$

$$\Sigma M_o = I_p \ddot{\theta}_p = -(H_L + H_R) l \cos\theta_p + (P_L + P_R) l \sin\theta_p + (C_L + C_R) \quad (4)$$

as shown in [3]. By combining all of the above equations and making the assumption that the Segway wheels roll without slip ($x = r\theta_w$), we can write the nonlinear equations of motion as

$$(M_p l^2 + I_p)\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a + M_p g l \sin\theta_p = -M_p l \ddot{x}\cos\theta_p \tag{5}$$

$$\frac{2k_m}{Rr}V_a = \left(2M_w + \frac{2I_w}{r^2} + M_p\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\dot{x} + M_p l\ddot{\theta}_p \cos\theta_p - M_p l\dot{\theta}_p^2 \sin\theta_p \tag{6}$$

Linearizing the above equations and supposing that $\theta_p = \pi + \phi$, we can write the linearized state space model of the system as

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2k_m k_e(M_p l r - I_p - M_p l^2)}{Rr^2\alpha} & \frac{M_p^2 g l^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{2k_m k_e(r\beta - M_p l)}{Rr^2\alpha} & \frac{M_p g l \beta}{\alpha} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2k_m(M_p l r - I_p - M_p l^2)}{Rr\alpha}\\ 0\\ \frac{2k_m(r\beta + M_p l)}{Rr\alpha}\end{bmatrix} V_a [3]$$

where

$$\alpha = I_p\beta + 2M_p l^2\left(M_w + \frac{I_w}{r^2}\right) \tag{7}$$

$$\beta = 2M_w + \frac{2I_w}{r^2} + M_p \tag{8}$$

The states of this system are lateral position (x) and lateral velocity ($\dot{x}$) of the wheel and angular position (ϕ) and angular velocity ($\dot{\phi}$) of the simplified mass representing the rider and handlebars. The input to the plant is an applied motor voltage (V_a). We believe that we can reasonably generate or estimate measurements of all of the states of the system. The quantities x, $\dot{x}$, and ω can be measured using rotary encoders (we will not model these dynamics for this project). The quantities ϕ and $\dot{\phi}$ can be generated from inertial sensing by a 3-axis accelerometer and a 3-axis gyroscope. The current, i, can be measured from a standard implementation of a current sensing circuit.

The real values for the variables chosen were estimated from quick research, and are as follows: $R = 2\ \Omega$, $L = 0.01$ H, $k_e = 0.1\ \frac{\text{V·s}}{\text{rad}}$, $k_m = 0.1\ \frac{\text{N·m}}{\text{A}}$, $I_R = 0.01\ \text{kg}\cdot\text{m}^2$, $M_w = 0.3$ kg, $I_w = 0.012\ \text{kg}\cdot\text{m}^2$, $r = 0.1$ m, $l = 2$ m, $M_p = 100$ kg, $I_p = 133\ \text{kg}\cdot\text{m}^2$, and $g = 9.81\ \frac{\text{kg·m}}{\text{s}^2}$.

4 Solution and Discrete Controller Design

We propose two controller architectures for this problem. The first will be an inner-loop/outer-loop architecture, and the second will be a LQR control with an estimator and integrator augment.

4.1 Inner-Loop/Outer-Loop Controller

This controller was built using a dual loop architecture and classical control techniques. Two separate discrete-time controllers were developed to control the position of the cart (x) and the angle of the mass on the Segway (ϕ). For simplicity and ease of modeling, only the plant of the Segway was simulated, and disturbances were modeled as voltage disturbances.

The goal of this controller was to stabilize both x and ϕ when a unit step was applied to x. In other words, when commanded to move forward 1 m, we wanted the Segway to do so with minimal settling time, overshoot, and steady state error while keeping the rider as vertical as possible.

The Simulink model of this controller is shown in Figure 3.

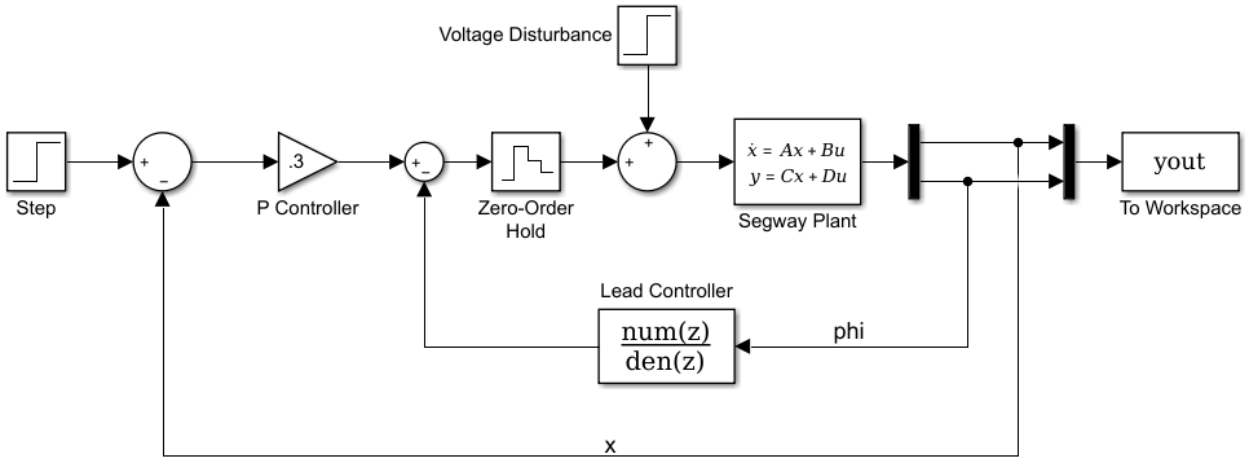

Figure 3: Simulink model of dual-loop controller

The first step in this controller design was to discretize the system using the MATLAB command *c2d*. A sample time of 1 kHz was chosen for this design. To design a controller for the system using classical control techniques, we then transformed the system back into continuous time using the bilinear transformation and the MATLAB command *d2c*. We then observed the root locus of the transfer function relating $\phi(\nu)$ to a voltage input, $u(\nu)$, where ν, the variable used in the bilinear transform, is equal to $\frac{2000(z-1)}{(z+1)}$. This root locus can be seen in Figure 4.

From the root locus, it is clear that the system is unstable for any gain. Hence, a lead controller was added to pull the closed loop poles to the left half plane. The lead controller selected was $D(\nu) = \frac{\nu+5}{\nu+20}$, and the root locus of the system multiplied by this controller is shown in Figure 5. Using this root locus, a gain of 80 was selected to result in four stable closed loop poles, resulting in a final controller of $D(\nu) = \frac{80(\nu+5)}{\nu+20}$, or $D(z) = \frac{80z-79.6}{z-0.9802}$. With this controller, the closed-loop discrete system had poles at 0.9802, 0.9963, 1, and 1.

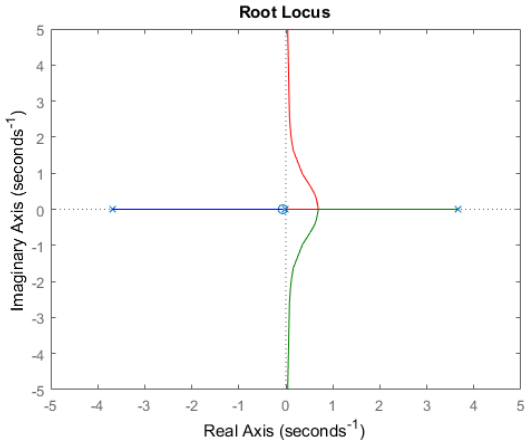

Figure 4: Root locus of $\frac{\phi(\nu)}{u(\nu)}$ without controller

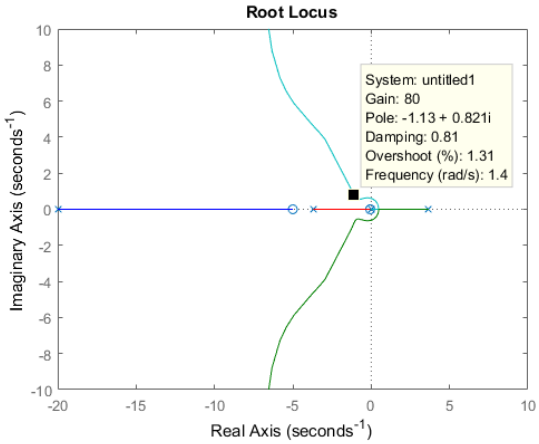

Figure 5: Root locus of $\frac{\phi(\nu)}{u(\nu)}$ with controller

The results of this controller can be seen below in Section 5. After designing a stable controller for ϕ, the plant with the lead controller was treated as one system, and a feedback controller for x was designed to stabilize this system. The new plant that we were using to control x was found using an equation from EECS 565: $\widetilde{P}_{xu} = P_{xu} * (1 + D(\nu)P_{\phi u})^{-1}$. [4] In this equation, $\widetilde{P}_{xu}$ represents

the transfer function from u to x including the Segway plant and the controller we previously designed for ϕ. The variable P_{xu} represents the transfer function from u to x when considering just the Segway plant. The variable $D(\nu)$ represents the controller we designed above for ϕ, and $P_{\phi u}$ represents the transfer function from u to ϕ when considering just the Segway plant.

The root locus of $\widetilde{P}_{yx}$ is shown below in Figure 6. We can see from the root locus that with a gain of 0.3, we can achieve four stable closed loop poles, so a P Controller gain of 0.3 was added to the x feedback loop as shown in Figure 3. With this controller, the closed-loop discrete system had poles at 0.9830, 0.9989 $\pm$ 0.0008j, and 0.9993.

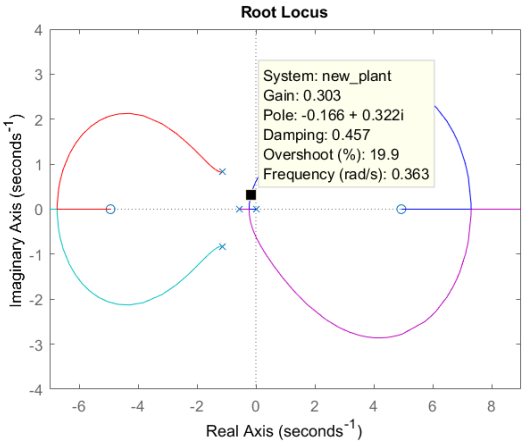

Figure 6: Root locus of $\widetilde{P}_{yx}$

The Simulink model also contains a block injecting voltage disturbances into the model. In reality, these should be torque disturbances on the motor, but for ease of modeling, the motor model is only considered in the controller designed in the next section. Hence, any torque disturbances are simulated as voltage disturbances in this model.

4.2 LQR Control/Estimator/Integrator

This controller was built with techniques for design and analysis discussed in EECS 565: Linear Feedback Control [4]. It will involve the use of an optimal controller and estimator designed in discrete time, and an integrator augmentation to handle steady state error and disturbance rejection. We first constructed the plant model in state space, and then added the integrator augmentation. The following equation is the plant model, where A_p is the Segway model presented above, A_m is the model of the motor. The B and C matrices follow this trend as well.

$$A_{\text{aug}} = \begin{bmatrix} A_p & 0 & 0 \\ 0 & A_m & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$B_{\text{aug}} = \begin{bmatrix} B_p \\ B_m \\ 0 \end{bmatrix}$$

$$C_{\text{aug}} = \mathbf{I}_{7\text{x}7}$$

The goal of this architecture is to design a controller that will be reasonably easy to tune, reliably able to estimate states, robust to noise, and robust to disturbances. The integrator augmentation is added to handle input disturbance, which is presented as a realistic issue for this system.

This generates a $7x7$ A matrix including the integrator augmentation. The B matrix is for a single input for the purpose of this design stage, and only includes the voltage input into the motor. The torque disturbance input will be discussed later. We assume the sensor reading will be handled as discussed in the earlier section, where a reasonable estimate can be obtained of each state.

This model is converted to a digital plant representation using the MATLAB *c2d* command. The discrete time state space matricies will be labeled as A_d,B_d, C_d, and D_d. A sample rate of 1.5 kHz was chosen for this design, which we believe to be reasonable based on our experience with embedded systems.

The first step is building the LQR controller. This is accomplished by using MATLAB's *dlqr* function on the discrete time model. The chosen weights for this controller were:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 \end{bmatrix}$$

These weights were chosen from a combination of empirical testing and intuition behind how different states should be penalized. This generated a K vector that could be used for state feedback control. Because of the integrator augmentation, K can be partitioned by $K = \begin{bmatrix} K_1 | K_2 \end{bmatrix}$, where K_1 contains the state feedback gains and K_2 contains the integrator gain.

The next step is to design the observer. We encountered stability issues with the MATLAB *dlqe* and transposed *dlqr* commands, so we settled for a sub-optimal controller designed with transposed *place*. The chosen poles were $P_{\text{obs}} = \begin{bmatrix} .05 & -.05 & .1 & -.1 & -.11 & .11 & .15 \end{bmatrix}$. The observer matrix L was partitioned the same way as K into L_1 and L_2.

With both the controller and observer designed, the system was placed into a Simulink model, which can be seen below in Figure 7. This model includes places to add an input disturbance, the secondary torque input disturbance, model noise, and measurement noise. The state space block on the top right contains a plant model with the motor model, and the"Estimator-Controller" block contains the state space equations:

$$A = A_d - B_d * K_1 - L_1 * C_d$$

$$B = L_1$$

$$C = \begin{bmatrix} K_1 \\ \mathbf{0} \end{bmatrix}$$

The plant model used in the "Estimator-Controller" state space block contains a discretized model of the plant model with the motor model, but without the integrator augmentation.

In the Simulink model, the discrete system begins at the"Sampler" block taken from an EECS 561 sample model, and the system ends at the "Zero-Order Hold" block.

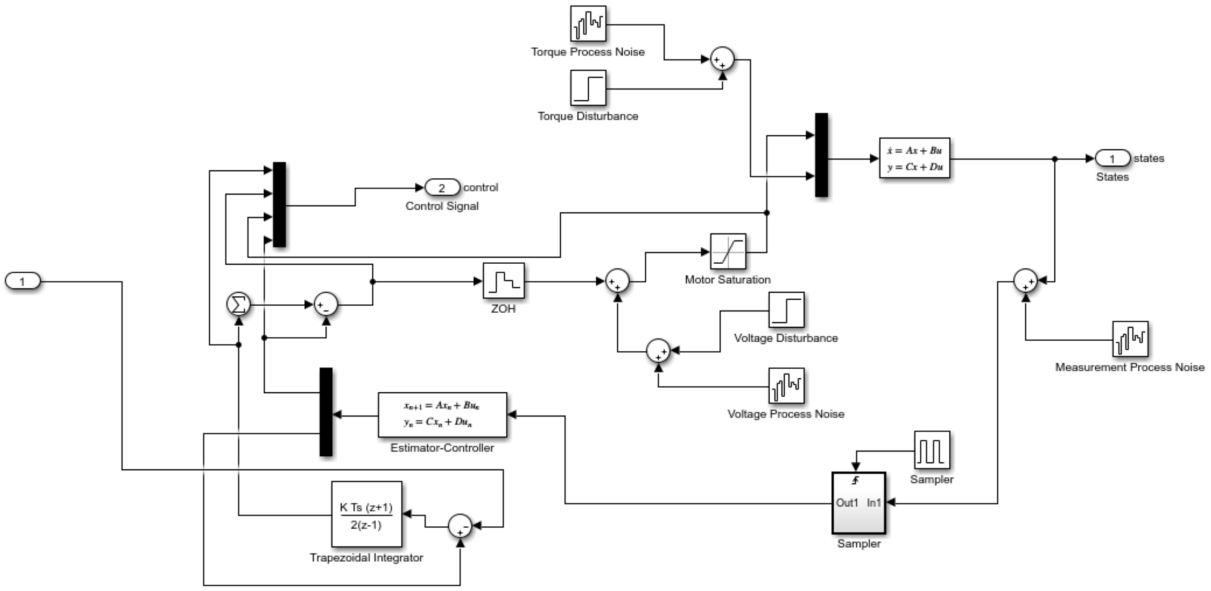

Figure 7: Simulink model of LQR controller and estimator with integrator

We chose to implement the integrator in discrete time as a trapezoidal integrator. It is more stable than a Forward-Euler integrator, but less computationally expensive than a method like Runge-Kutta.

The Simulink model also includes a saturation block on the motor plant input representing the limitation on the motor voltage as the battery voltage. Quick research states that Segway battery voltages can be 72 volts, so that value was set as the saturation value. It is assumed that the motor can be driven in both directions by an H-Bridge, so the saturation bounds are -72 volts to +72 volts.

5 Results and Simulation

The primary performance objectives were related to tracking and disturbance. The input to the closed loop system was a reference lateral position to move to while also keeping the inverted pendulum part of the system stable. We also aimed to reject disturbance voltages and torques injected into the motors.

5.1 Inner-Loop/Outer-Loop Controller Results

The dual loop controller provided a Segway response that would keep the rider upright as it moved, but unfortunately the response time was rather slow. In these simulations, it took the Segway around 15 seconds to travel 1 meter (as shown in Figure 8), which is a speed of approximately 0.07 m/s, whereas beginner Segways can travel at speeds of up to 2.7 m/s. [1] It was difficult to decrease the rise time of the system without inducing large oscillations and unstable behavior.

Despite being slow, the system was robust to disturbances. In Figure 9, a step input of 1 m was applied at time $t = 0$ and then a voltage disturbance of 3 V was applied at time $t = 20$. It does take the Segway another 15-20 seconds to settle, but the Segway eventually does stabilize at a new setpoint of 11 m. Even with this disturbance, the rider experiences an angular displacement of no more than ± 1 degree according to the simulation.

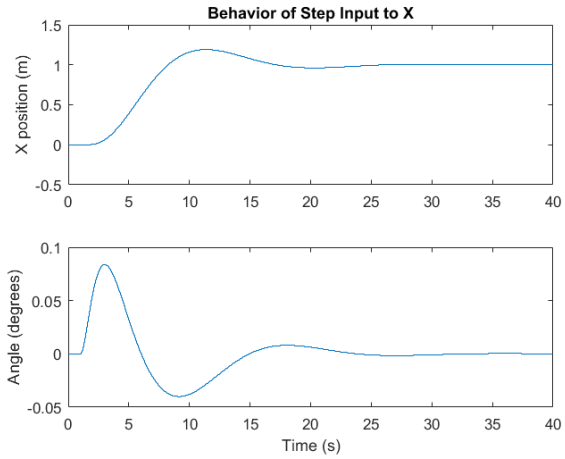

Figure 8: Step Input

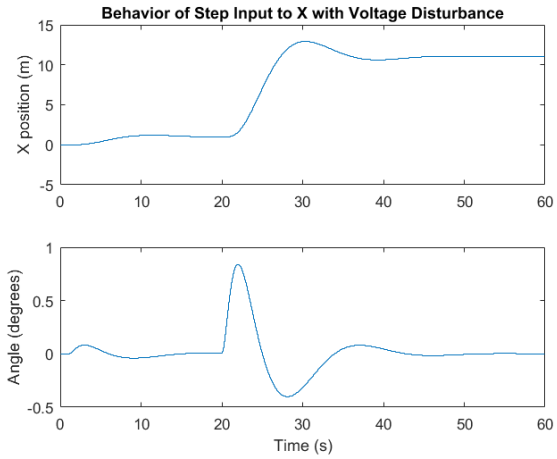

Figure 9: Step Input with Voltage Disturbance

5.2 LQR Control/Estimator/Integrator Results

This controller provided a stable and reasonably fast response that rejected disturbance and noise. The controller was tested with several different input scenarios, which included a step input to x, initial conditions on x and ϕ, and a ramp input to x. These same scenarios are then tested with added model noise and measurement noise.

Figure 10 shows the response of the system to a step input of 1 m to x position at $t = 1$, a torque disturbance of 25 N-m at $t = 40$, and a voltage disturbance of 5 V at $t = 80$. It can be shown that the controller drove the Segway 1 m in under 10 s, and had a small and stable response to angle. It also successfully rejected both of the disturbances with only a relatively small transient response.

Figure 11 shows the response of the system to an initial condition in ϕ. It shows that the ϕ initial condition is rejected, and the response in x returns to 0.

Figure 12 shows the response of the system to an initial condition in x. It shows that the x initial condition is rejected, and the response in ϕ returns to 0.

Figure 13 shows the response of the system to a ramp input in x. The ramp input is successfully tracked. The value of ϕ stays positive during the tracking of the ramp input, as would be expected during Segway motion.

A first observation is that there is certainly an issue with the scaling of ϕ for this model. The trends appear to behave correctly, but the values appear to be at least an order of magnitude too small. It is unclear whether this is an issue with the constants chosen during model design, or an issue with the model itself.

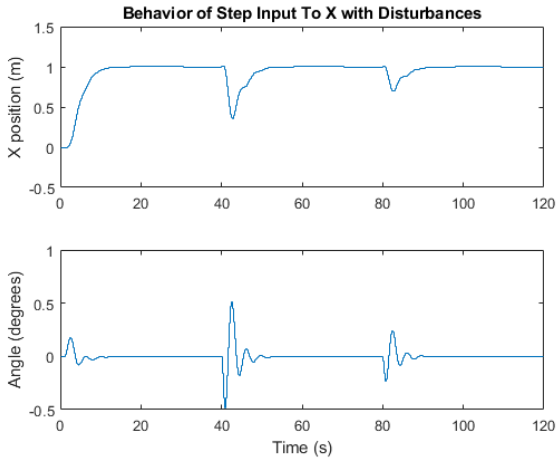

Figure 10: Step Input

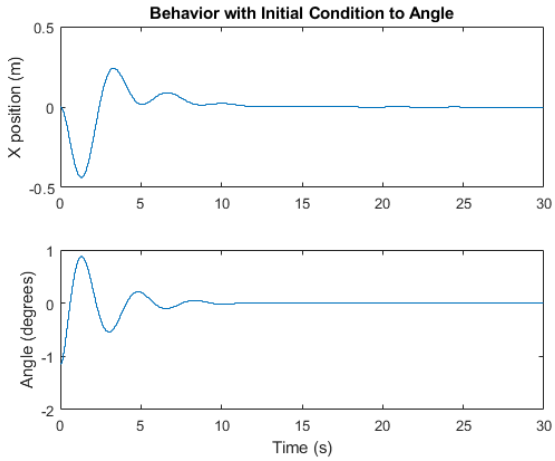

Figure 11: Initial Condition in ϕ

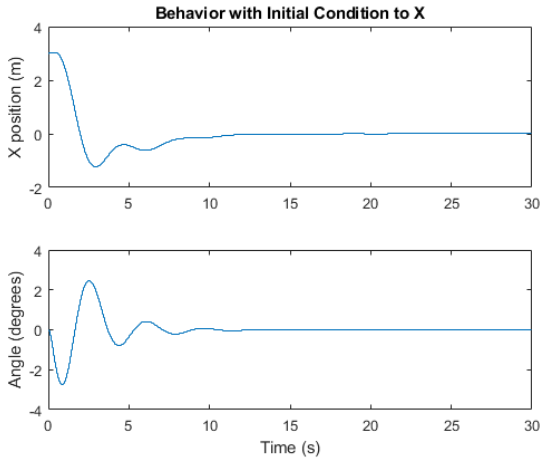

Figure 12: Initial Condition in x

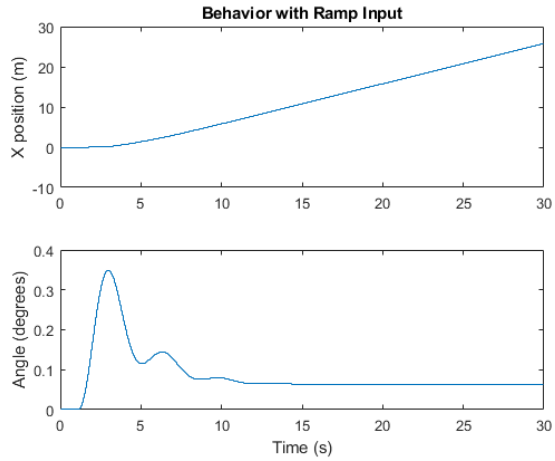

Figure 13: Ramp Response

The following is the same four scenarios but with added process and measurement noise. The noise was injected through Simulink's "Band-Limited White Noise" block with 10^{-6} noise power in measurements, and 10^{-3} noise power in the disturbances. The results in Figures 14, 15, 16, and 17 are nearly the same as the previously discussed plots, though with visible noise added. However, this does not significantly change the stability or steady state behavior.

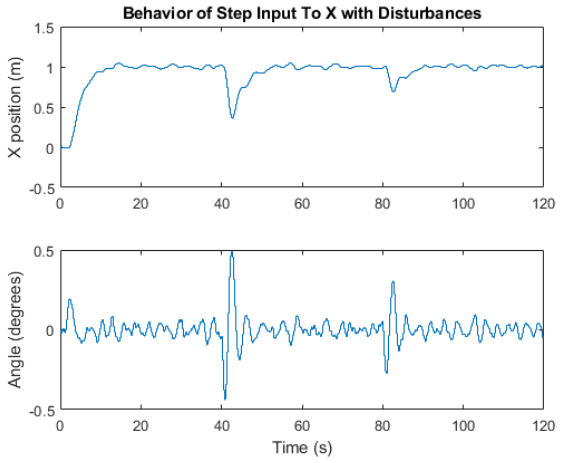

Figure 14: Step Input with Noise

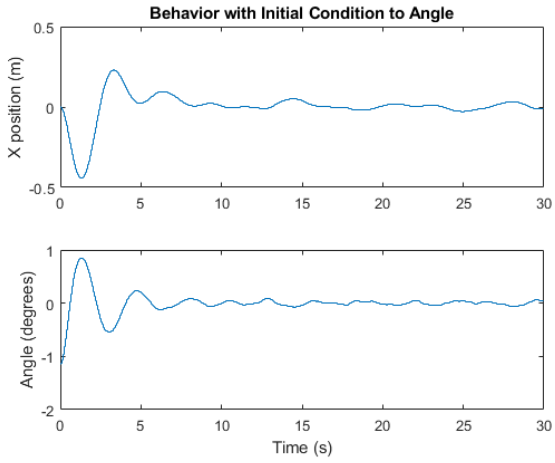

Figure 15: Initial Condition in ϕ with Noise

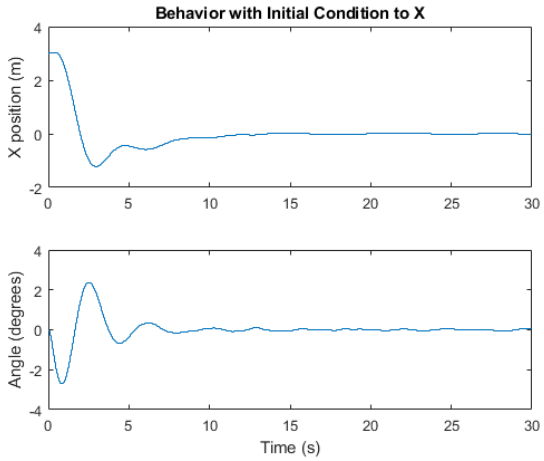

Figure 16: Initial Condition in x with Noise

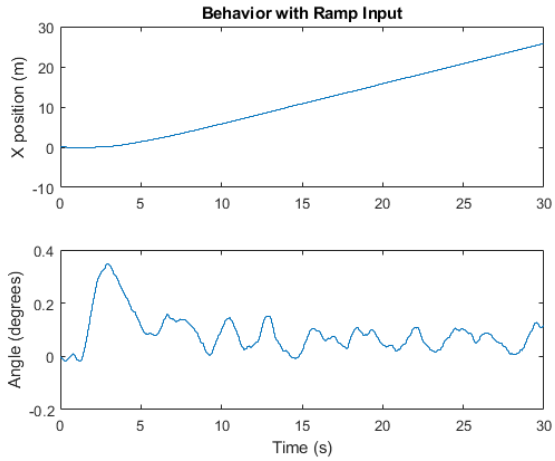

Figure 17: Ramp Response with Noise

6 Conclusion

In this paper, we examined a Segway, a two-wheeled personal transportation vehicle that can move forward and backward depending on the direction a rider leans. We presented a dynamic model of a Segway and designed two different controllers for this model using both classical control methods (root locus) and modern control techniques (LQR). The dual-loop controller designed using the root locus was robust to voltage disturbances, but had a relatively slow response time. The modern controller, on the other hand, provided a stable and fast response that rejected disturbances (both voltage and torque) and noise. Possible extensions to this project could include reducing the response time of the dual-loop controller, obtaining more accurate values for all the constants, and modeling more of the sensor dynamics to make a more accurate, higher-fidelity model.

Valerie wrote the Introduction and Problem/Modeling sections of this report. She also designed the inner-loop/outer-loop controller. Jon wrote the Abstract and designed the LQR/Estimator/Integrator controller. Team members collaborated on the Conclusion.

References

[1] Wikipedia contributors. (2018, April 4). Segway PT. In *Wikipedia, The Free Encyclopedia.* Retrieved 19:49, April 17, 2018, from https://en.wikipedia.org/w/index.php?title=Segway_PT&oldid=834119363.

[2] Wikipedia contributors. (2018, January 19). Inverted pendulum. In *Wikipedia, The Free Encyclopedia.* Retrieved 20:17, April 17, 2018, from https://en.wikipedia.org/w/index.php?title=Inverted_pendulum&oldid=821341065.

[3] Optimal control of Segway personal transporter, Reza Babazadeh, Ataollah Gogani Khiabani, Hadi Azmi. http://ieeexplore.ieee.org/document/7483129/.

[4] A First Graduate Course in Feedback Control, J.S. Freudenberg, EECS 565 Winter 2018 Book Edition

A Code

A.1 Dual Loop Design

```
R    = 2;
L    = 0.01;
ke   = 0.1;
km   = 0.1;
IR   = 0.01;
Mw   = 0.3;
Iw   = 0.012;
r    = 0.1;
l    = 2;
Mp   = 100;
Ip   = 133;
g    = 9.81;

beta = 2*Mw + 2*Iw/r^2 + Mp;
alpha = Ip*beta + 2 * Mp * l^2 * (Mw + Iw/r^2);

Ap = [0,                   1,                              0,                  0;
      0,   2*km*ke*(Mp*l*r-Ip-Mp*l^2)/(R*r^2*alpha),   Mp^2*g*l^2/alpha,   0;
      0,                   0,                              0,                  1;
      0,      2*km*ke*(r*beta-Mp*l)/(R*r^2*alpha),     Mp*g*l*beta/alpha,  0];

Bp = [0;
      2*km*(Mp*l*r-Ip-Mp*l^2)/(R*r*alpha);
      0;
      2*km*(r*beta+Mp*l)/(R*r*alpha)];

A = [Ap, zeros(4,2);
      zeros(1,4), R/L, ke/L;
      zeros(1,4), km/IR, -ke/IR];


B = [Bp;1/L;0];
E = [zeros(5,1);-1/IR];
F = zeros(4,1);
Cp = [1, 0 ,0, 0;
     0, 0, 180/pi, 0];

C = [Cp, zeros(2,2);
     zeros(1,4), 1 0;
```

```
      zeros(1,5), 1 ];

D = zeros(4,2);

Dp = zeros(size(Cp, 1), size(Bp, 2));

T = 1/1000;


sys_con = ss(Ap,Bp,Cp,Dp);
sys_dis = c2d(sys_con,T, 'zoh');
sys_dis_blt = d2c(sys_dis, 'tustin');
sys_dis_blt_tf = tf(sys_dis_blt);

sim('design12_mdl')
subplot(2,1,1)
stairs(tout,yout(:,1))
title('Behavior of Step Input to X with Voltage Disturbance')
ylabel('X position (m)')
hold on
subplot(2,1,2)
stairs(tout,yout(:,2))
ylabel('Angle (degrees)')
xlabel('Time (s)')

%% Design a Lead Compensator for phi
close all
figure
rlocus(sys_dis_blt_tf(2))
axis([-5 5 -5 5])
figure
s = tf('s');
z = tf('z', T);
rlocus(sys_dis_blt_tf(2)*(s+5)/(s+20))
axis([-20 10 -10 10])
controller1 = 80*(s+5)/(s+20);
dis_controller = c2d(controller1, T, 'zoh')

%% Design a Lead Compensator for x
close all
% Figure out the actual plant (with phi controller)
S2 = 1/(1 + controller1*sys_dis_blt_tf(2));
new_plant = sys_dis_blt_tf(1)*S2;
new_plant = minreal(new_plant);
figure
rlocus(new_plant)
axis([-7 9 -4 4])
```

A.2 LQR Design

```
R  = 2;
L  = 0.01;
ke = 0.1;
km = 0.1;
IR = 0.01;
Mw = 0.3;
Iw = 0.012;
r  = 0.1;
l  = 2;
Mp = 100;
Ip = 133;
g  = 9.81;
```

```
13
14 beta = 2*Mw + 2*Iw/r^2 + Mp;
15 alpha = Ip*beta + 2 * Mp * l^2 * (Mw + Iw/r^2);
16
17 Ap = [0,                          1,                                    0,            0;
18       0,   2*km*ke*(Mp*l*r-Ip-Mp*l^2)/(R*r^2*alpha),    Mp^2*g*l^2/alpha,  0;
19       0,                          0,                                    0,            1;
20       0,     2*km*ke*(r*beta-Mp*l)/(R*r^2*alpha),     Mp*g*l*beta/alpha,  0];
21
22 Bp = [0;
23       2*km*(Mp*l*r-Ip-Mp*l^2)/(R*r*alpha);
24       0;
25       2*km*(r*beta+Mp*l)/(R*r*alpha)];
26
27 A = [Ap, zeros(4,2);
28       zeros(1,4), R/L, ke/L;
29       zeros(1,4), km/IR, -ke/IR];
30
31
32 B = [Bp;1/L;0];
33 E = [zeros(5,1);-1/IR];
34 %B = [B E];
35 F = zeros(4,1);
36 Cp = [1, 0 ,0, 0;
37       0, 0, 180/pi, 0];
38
39 C = [Cp, zeros(2,2);
40       zeros(1,4), 1 0;
41       zeros(1,5), 1 ];
42 C = eye(6,6);
43
44 D = zeros(6,1);
45
46 Aaug = [A zeros(6,1);1 zeros(1,6)];
47 Baug = [B(:,1);zeros(1,1)];
48 Caug = [eye(7,7)];
49 Daug = zeros(7,1);
50
51 sys_con = ss(Aaug,Baug,Caug,Daug);
52
53 T = 1/1250;
54
55 sys_dis = c2d(sys_con,T,'zoh');
56
57
58 [Ad Bd Cd Dd] = ssdata(sys_dis);
59
60
61 Q = diag([1,.1,1000,1000,.0001,1,100]);
62 R = eye(1,1);
63
64 Kd = dlqr(Ad,Bd,Q,R);
65
66
67 Kd1 = Kd(1:6);
68
69 Kd2 = abs(Kd(7));
70 pobs = [.05  -.05  .1  -.1 -.11 .11  .15];
71
72 sys_con2 = ss(A,B,C,D);
73 sys_dis2 = c2d(sys_con2,T);
74 [Ad2 Bd2 Cd2 Dd2] = ssdata(sys_dis2);
```

```
75
76
77 Ld = place(Ad',Cd',pobs)';
78 Ld1 = Ld(1:6,1:6);
79 Ld2 = (Ld(:,7));
80
81 voltage_disturbance = [1e-3];
82 %voltage_disturbance = 0;
83 measurement_noise = [1e-5 1e-8 1e-8 1e-8 1e-8 1e-7];
84 %measurement_noise = 0;
85 t_final = 120;
86 t = 0:T:t_final;
87 t_step = 1;
88 t_dist = 10;
89 clear r
90 for i = 1:length(t)
91 if t(i) <= t_step
92    r(i) = 0;
93  elseif t(i) >= t_step
94       r(i) = 1;
95  end
96
97 end
98 t = t';
99 r = r';
100 x0=zeros(13,1);
101 sim('design_final');
102
103 figure(1)
104 clf
105 subplot(2,1,1)
106
107 stairs(tout,yout(:,1))
108 title('Behavior of Step Input To X with Disturbances')
109 ylabel('X position (m)')
110 subplot(2,1,2)
111
112 stairs(tout,rad2deg(yout(:,3)))
113 ylabel('Angle (degrees)');
114 xlabel('Time (s)');
115
116 t_final = 30;
117 r = zeros(size(r));
118 x0=[0;0;-2e-2;0;0;0;0;0;0;0;0;0;0];
119 sim('design_final');
120
121 figure(2)
122 clf
123 subplot(2,1,1)
124 stairs(tout,yout(:,1))
125 title('Behavior with Initial Condition to Angle')
126 ylabel('X position (m)')
127 subplot(2,1,2)
128 stairs(tout,rad2deg(yout(:,3)))
129
130 ylabel('Angle (degrees)');
131 xlabel('Time (s)');
132
133 r = zeros(size(r));
134 x0=[3;0;0;0;0;0;0;0;0;0;0;0;0];
135 sim('design_final');
136
```

```
figure(3)
clf
subplot(2,1,1)
stairs(tout,yout(:,1))
title('Behavior with Initial Condition to X')
ylabel('X position (m)')
subplot(2,1,2)
stairs(tout,rad2deg(yout(:,3)))
ylabel('Angle (degrees)');
xlabel('Time (s)');

for i = 1:length(t)
if t(i) <= t_step
   r(i) = 0;
 elseif t(i) >= t_step
      r(i) = t(i);
end
end

x0=[0;0;0;0;0;0;0;0;0;0;0;0;0;0];
sim('design_final');

figure(4)
clf
subplot(2,1,1)
stairs(tout,yout(:,1))
title('Behavior with Ramp Input')
ylabel('X position (m)')
subplot(2,1,2)
stairs(tout,rad2deg(yout(:,3)))
ylabel('Angle (degrees)');
xlabel('Time (s)');


r = sawtooth(t/(2*pi),.5);
t_final = 40;
x0=[0;0;0;0;0;0;0;0;0;0;0;0;0;0];
sim('design_final');


figure(5)
clf
subplot(2,1,1)
stairs(tout,yout(:,1))
title('Behavior with Triangle Wave Input')
ylabel('X position (m)')
subplot(2,1,2)
stairs(tout,rad2deg(yout(:,3)))
ylabel('Angle (degrees)');
xlabel('Time (s)');
```