Sensor Stabilization on a Segway Robot Using LQR Control

2.151 Final Project Report

G. Chen, A. Lenhard, I. Montanaro, S. Ni, J. Santillan

7 December 2020

Abstract

The sagittal dynamics and control of a stabilized, front-facing, actuated sensor suite on a segway robot are considered. The goal of this project is to develop the dynamical model of the actuated sensor suite, linearize said model to analyze its stability, stabilize the system using Linear Quadratic Regulator (LQR) control, and simulate the system using MATLAB. The LQR cost function takes into account limitations of the sensor and motors, and is weighted accordingly to produce an ideal control effort for sensor stabilization.

1 Introduction

To test autonomy algorithms on hardware, a common choice for a ground robot is the Jackal [1] (Figure 1). However, one of its drawbacks is its low maximum speed of 2.0 m/s. If one wants to develop algorithms for high speed navigation [2], a new robot has to be used during hardware testing. An alternative ground robot with a higher maximum speed of 5.0 m/s is the Loomo segway robot (Figure 2). This robot resolves the speed issue, but a 2-wheeled segway robot will heavily pitch when accelerating or decelerating unlike a stable 4-wheeled robot like the Jackal. Autonomy algorithms require readings from onboard sensors, i.e. for localization, mapping, etc. If these sensors are mounted on the robot, then the robot's pitch will also affect the sensor readings. For example, when accelerating, the segway will pitch forward and a camera mounted on the robot will end up looking at the floor, which is not useful sensor data.

Figure 1: 4-wheeled Jackal robot.

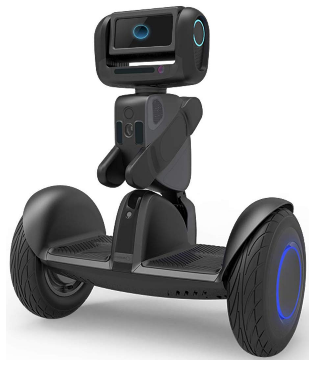
Figure 2: 2-wheeled Loomo robot.

1.1 Approach

Instead of mounting the sensors in a fixed way, directly on the robot, it would be helpful to have an alternative. To alleviate this pitching issue, we can model the dynamics of an *actuated* sensor suite (i.e. camera, lidar, etc.) mounted on the segway and develop a control scheme to ensure the suite stays level and provides consistent sensor readings. This requires control of both the wheel actuation to move the robot, and the simultaneous actuation of the platform upon which the sensors will sit. Because there are so many applications to this type of system, we found it to be an exciting project.

To model the Loomo robot with an attached sensor suite, a few assumptions are made for simplification.

1. The sensor suite is modeled as a point mass at the end of a pendulum. The sensor can be attached at the end of a very light 3D printed plate.
2. Mass of the segway is evenly distributed and the center of mass is in the center of its height. This may not be realistic, but it simplifies calculations.

The parameters of the Loomo model are tabulated in Table 1 below. They are based on the available Loomo specifications from its developer [3], but some educated guesses were made since many of the details are not publicly available.

Table 1: Specifications of Loomo model

Total mass	19 kg
Height from ground to top	0.64 m
Wheel radius	0.14 m
Height from wheel axle to top	0.5 m
Estimated total wheel mass	4 kg

The sensor suite by construction will be 1 kg total mass and have a length of 0.15 m.

The motors and surrounding hardware involved in the actuation of the wheels and sensor suite are imperfect, and there is friction in the bearings and between the wheels and ground. Since the magnitude of the effects is a function of the associated rotational velocities, we model this as damping with estimated damping constants as follows:

B_w: 0.1 Nm/rad/s (wheel bearings)
B_s: 0.1 Nm/rad/s (sensor suite bearing)
B_r: 0.02 Nm/rad/s (rolling friction on the ground) [4]

This report details the derivation of the dynamical model of the system, the design of its controller, and results from the simulation.

2 Dynamics

Definitions of variables used in the segway and sensor suite system are described below and displayed in Figure 3. (Please refer to next page.)

α: angle of wheel from vertical of world frame
θ: angle of segway shaft from vertical of world frame
β: angle of sensor suite from the segway shaft
x_c: x position of the segway center of mass
y_c: y position of the segway center of mass
x_s: x position of the sensor suite center of mass
y_s: y position of the sensor suite center of mass
R: radius of the segway wheel
L: length to the segway center of mass (half the segway axle)
l_p: length from the segway center of mass to the sensor suite connection
l_s: length of the sensor suite connection rod
m: mass of segway shaft

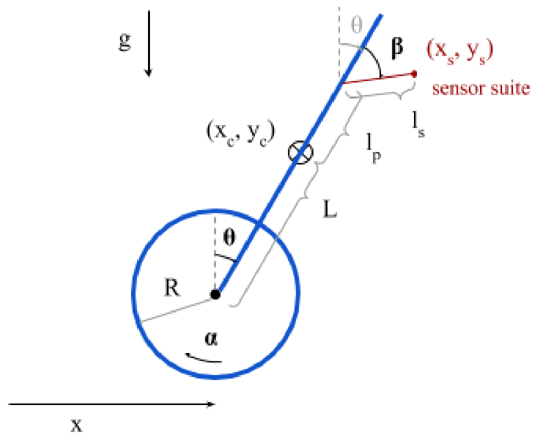

Figure 3: Sensor suite on a segway.

m_w: mass of segway wheels
m_s: mass of sensor suite (point mass)
There are existing literature modeling the dynamics of inverted double pendulums on carts.[5] However, in this study, we took a different approach as the second "link" of the double pendulum is actuated. The equations of motion of the combined system (i.e. segway robot and its sensors) are derived via Lagrangian methods. We define α, θ, and β as the generalized coordinates for this system. Examining the kinematics of the segway robot, we can write,

$$x_c = R\alpha + L\sin\theta$$

$$y_c = R + L\cos\theta$$

Similarly, the kinematic equations for the senor suite are as follows,

$$x_s = R\alpha + (L + l_p)\sin\theta + l_s\sin(\theta+\beta)$$

$$y_s = R + (L + l_p)\cos\theta + l_s\cos(\theta+\beta)$$

Consequently, considering the components of potential energy T and potential energy U due to the wheel, segway, and sensor suite, we applied Lagrange's equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial\mathcal{L}}{\partial\dot{q}_i}\right) - \frac{\partial\mathcal{L}}{\partial q_i} = f_i \text{ for } i \in \{\alpha,\theta,\beta\}$$

where $\mathcal{L} = T - U$, and obtained three equations of motion (corresponding to each of the generalized coordinates) as follows,

$$\tau_\alpha = 2a\left[\ddot{\alpha}\right] + c\left[\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right] + e\left[(\ddot{\theta}+\ddot{\beta})\cos(\theta+\beta) - (\dot{\theta}+\dot{\beta})^2\sin(\theta+\beta)\right]$$

$$\tau_\theta = 2b\left[(\ddot{\theta}+\ddot{\beta})\right] + c\left[\ddot{\alpha}\cos\theta\right] + d\left[(2\ddot{\theta}+\ddot{\beta})\cos\beta - (2\dot{\theta}+\dot{\beta})\dot{\beta}\sin\beta\right] + e\left[\ddot{\alpha}\cos(\theta+\beta)\right] + 2f\left[\ddot{\theta}\right] - w\left[\sin\theta\right] - z\left[\sin(\theta+\beta)\right]$$

$$\tau_\beta = 2b\left[\ddot{\theta}+\ddot{\beta}\right] + d\left[\ddot{\theta}\cos\beta + \dot{\theta}^2\sin\beta\right] + e\left[\ddot{\alpha}\cos(\theta+\beta)\right] + 2h\left[\ddot{\beta}\right] - z\left[\sin(\theta+\beta)\right]$$

where coefficients a, b, c, f, h, v, w, z are defined in the Appendix 8.1. For a thorough derivation of the Lagrangian mechanics, please also refer to 8.1.

2.1 Generalized Forces

Let τ_α be the generalized force (torque) associated with the α coordinate, τ_θ for the θ coordinate, and τ_β for the β coordinate. To find the actual values of these variables, we performed a power analysis. Power is the product of torque and speed, so the power input to drive each component can be used to solve for the generalized force (or torque, as in the following cases).

For the wheel actuation (associated with the generalized coordinate α), we write:

$$\tau_\alpha \dot{\alpha} = \tau_w \dot{\alpha}$$

For the shaft rotation,

$$\tau_\theta \dot{\theta} = -\tau_w \dot{\theta}$$

because the torque that actuates the wheel actually produces an opposite reaction torque on the shaft.

Similarly, for the sensor suite actuation,

$$\tau_\beta \dot{\beta} = \tau_s \dot{\beta}$$

since the actuation inputs power into the system.

We also need to consider damping in the system, which, in our case, is caused by friction. The first type of friction in the system results from the contact of the wheels on the ground. For this rolling friction, we associated a damping constant of B_r. Additionally, there is friction in each of the joints. We use the constants B_w and B_s, respectively, to describe the damping at the points of wheel actuation and sensor suite actuation.

In the case of damping at the point of wheel actuation, we have to consider the relative velocity, defined as $\dot{\alpha} - \dot{\theta}$.

Putting it all together, we have

$$\tau_\alpha = \tau_w - B_r \dot{\alpha} - B_w(\dot{\alpha} - \dot{\theta})$$

$$\tau_\theta = -\tau_w + B_w(\dot{\alpha} - \dot{\theta})$$

$$\tau_\beta = \tau_s - B_s \dot{\beta}$$

The damping constants outlined above construct the B matrix for our state equations:

$$\begin{bmatrix} \tau_\alpha \\ \tau_\theta \\ \tau_\beta \end{bmatrix} = \begin{bmatrix} \tau_w \\ -\tau_w \\ \tau_s \end{bmatrix} - \begin{bmatrix} B_r + B_m & -B_m & 0 \\ -B_m & B_m & 0 \\ 0 & 0 & B_s \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix}$$

2.2 Linearization

To find an approximate linear representation for small motions about rest when the segway and suite system is in a fully upright pose, we linearized about $\alpha = 0, \theta = 0, \beta = 0, \dot{\alpha} = 0, \dot{\theta} = 0, \dot{\beta} = 0$. This resulted in the following linear equations:

$$\tau_\alpha = [2a]\ddot{\alpha} + [c + e]\ddot{\theta} + [e]\ddot{\beta}$$

$$\tau_\theta = [c + e]\ddot{\alpha} + [2(b + d + f)]\ddot{\theta} + [2b + d]\ddot{\beta} + [-w]\theta + [-z](\theta + \beta)$$

$$\tau_\beta = [e]\ddot{\alpha} + [2b+d]\ddot{\theta} + [2(b+h)]\ddot{\beta} + [-z](\theta+\beta)$$

Translating these linearized equations into matrix form and plugging in for our generalized forces results in the following matrix system of equations:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_w \\ \tau_s \end{bmatrix} = \begin{bmatrix} 2a & c+e & e \\ c+e & 2(b+d+f) & 2b+d \\ e & 2b+d & 2(b+h) \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\theta} \\ \ddot{\beta} \end{bmatrix} + \begin{bmatrix} B_r+B_m & -B_m & 0 \\ -B_m & B_m & 0 \\ 0 & 0 & B_s \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -w & -z \\ 0 & -z \end{bmatrix} \begin{bmatrix} \theta \\ \theta+\beta \end{bmatrix}$$

$$\boldsymbol{H\tau} = \boldsymbol{I} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\theta} \\ \ddot{\beta} \end{bmatrix} + \boldsymbol{B} \begin{bmatrix} \dot{\alpha} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} + \boldsymbol{C} \begin{bmatrix} \theta \\ \theta+\beta \end{bmatrix}$$

After rearranging, we find our linearized state equations to be:

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \theta \\ \theta+\beta \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ \multicolumn{2}{c}{\boldsymbol{I}^{-1}\boldsymbol{C}} & \multicolumn{3}{c}{-\boldsymbol{I}^{-1}\boldsymbol{B}} \end{bmatrix} \begin{bmatrix} \theta \\ \theta+\beta \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \multicolumn{2}{c}{\boldsymbol{I}^{-1}\boldsymbol{H}} \end{bmatrix} \boldsymbol{\tau}$$

$$\theta+\beta = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \theta+\beta \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix}$$

It should be noted that $\theta+\beta$ and not β is used as a state variable because the angle of the sensor suite in the world frame is $\theta+\beta$, which is the angle we want driven to 0 radians to keep the sensor suite upright regardless of the segway angle. β was previously used as a generalized coordinate because β represents the angle of the sensor suite actuator and can be more readily related to the actuator's generalized torque.

3 Controller Design

To determine if our system is controllable, the controllability matrix $\mathbf{C}_{ont}$ for the system, given by

$$\mathbf{C}_{ont} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}$$

was used as a metric. It was found using the `ctrb()` command in MATLAB with the **A** and **B** matrices used as inputs. The rank of the controllability matrix is 5, so the system is fully controllable from the inputs. Similarly, the observability of the system needs to be determined. The observability matrix for the system was found using the `obsv()` command in MATLAB with the **A** and **C** matrices used as inputs. The rank of the observability matrix is 5, so it can be concluded that all the states are fully observable from the outputs.

Since our linearized system is fully controllable, we are able to design a full-state feedback controller for the linearized system. We used the Linear Quadratic Regulator (LQR) approach to design our full-state feedback controller. The LQR algorithm finds a control effort $\mathbf{u}(t)$ that minimizes a cost function involving a tradeoff between control effort and system performance. The cost function V to be minimized is given by,

$$\min_{\mathbf{u}(t)} V = \int_{\tau=t}^{\tau=T_f} (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})\, d\tau$$

where **x** is the state vector, **Q** is the state penalty matrix, and **R** is the input penalty matrix.

The state and input penalty matrices are guided by the design of a controller that satisfies what we want our maximum state and input values to be. In our case, the values of interest are the deviations of θ and $\theta + \beta$ from 0 radians. For our system the maximum state and input deviations are:

$$\theta_{max} = 0.1 \text{ rad}$$
$$(\theta + \beta)_{max} = 0.03 \text{ rad}$$
$$\tau_{wmax} = 10 \text{ N} \cdot \text{m}$$
$$\tau_{smax} = 3 \text{ N} \cdot \text{m}$$

These values were selected for a variety of reasons. θ_{max} was selected with the segway tipping point in mind. $(\theta + \beta)_{max}$ was calculated as the angle that would cause a lidar to view the floor within its 25 meter range. [6] τ_{wmax} and τ_{smax} were selected based off of the wheel motor and sensor suite motor specifications, respectively. It should be noted that this is not a guarantee that these states are limited to the specified maximum values, however this provides an appropriate starting point for the controller design. After carefully tuning the these parameters, we arrived at $\hat{\tau}_{w_{max}} = 5 \text{ N} \cdot \text{m}$ and $\hat{\tau}_{s_{max}} = 1 \text{ N} \cdot \text{m}$. Consequently, the state penalty matrix **Q** can be written as

$$\boldsymbol{Q} = \begin{bmatrix} \theta_{max}^{-2} & 0 & 0 & 0 & 0 \\ 0 & (\theta + \beta)_{max}^{-2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Similarly, the input penalty matrix **R** can be written as

$$\boldsymbol{R} = \begin{bmatrix} \hat{\tau}_{w_{max}}^{-2} & 0 \\ 0 & \hat{\tau}_{w_{max}}^{-2} \end{bmatrix}$$

After inputting these values into our two penalty matrices, the optimal gains were found to generate the closed-loop system. The closed-loop pole-zero plot is shown in Figure 4.

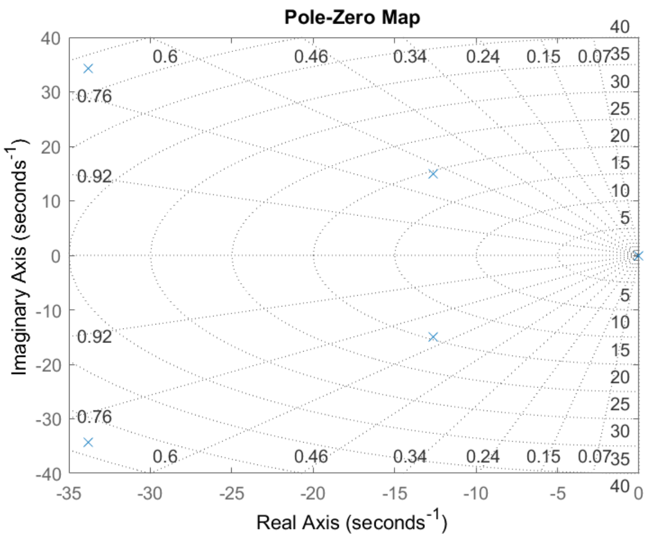

Figure 4: Pole-Zero Map of LQR Closed-Loop System.

4 Results

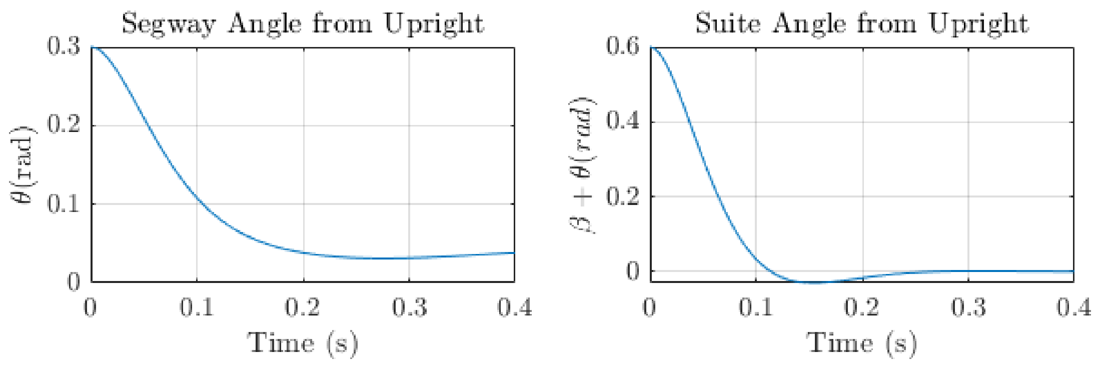

Figure 5: Initial condition response to `x0 = [0.3; 0.3; 25; 0; 0]` - Angles.

The graphs above demonstrate the quick performance of the system. The angle of the segway shaft settles very quickly, but it does have some steady state error, hovering at around 0.03 radians; however, the sensor suite angle is what really matters. Similar to the segway angle, it converges very quickly. The difference is that it has negligible error.

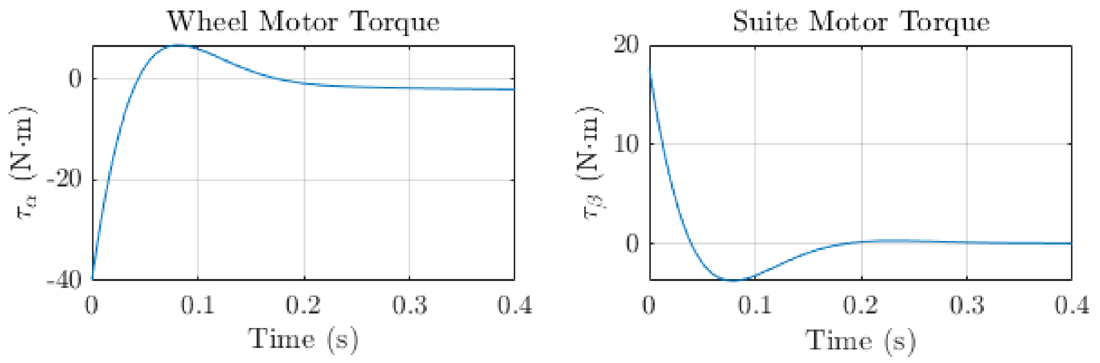

Figure 6: Initial condition response to `x0 = [0.3; 0.3; 25; 0; 0]` - Torques.

Since the Loomo robot starts off in an unstable configuration, it is at risk of falling over. Therefore, its motors must be quick to counteract any moments due to gravity; as a result, the motors start off with high-magnitude values of torque, as can be seen in the graphs above. Both motors rapidly tend to zero, and soon the suite motor does not have to provide very much power. The wheel motor continues to provide a small amount of torque, keeping the segway in motion. Intuitively, this makes sense: the Loomo will never get to a perfect state of equilibrium, and even if it did, it would require some amount of input torque to keep the system in motion and ultimately balanced.

The speed of the wheel started off at 25 rad/s, which translated to a tangential speed of 3.5 m/s, and reaches a maximum speed of around 4.2 m/s. These values approach the limit of the Loomo's capabilities, but it is still able to comfortably able to achieve these readings. The segway angular velocity has a relatively narrow trough in its corresponding graph and then quickly bounces back to near zero velocity. Similar to the segway, the suite angular velocity has a trough in the same place as the segway; however, since the measure of the suite angle is $\theta + \beta$, having a negative velocity from both the segway and the suite is what causes the suite angle to overshoot in Figure 5. We see a slight peak in the suite angular velocity to make up for this behavior.

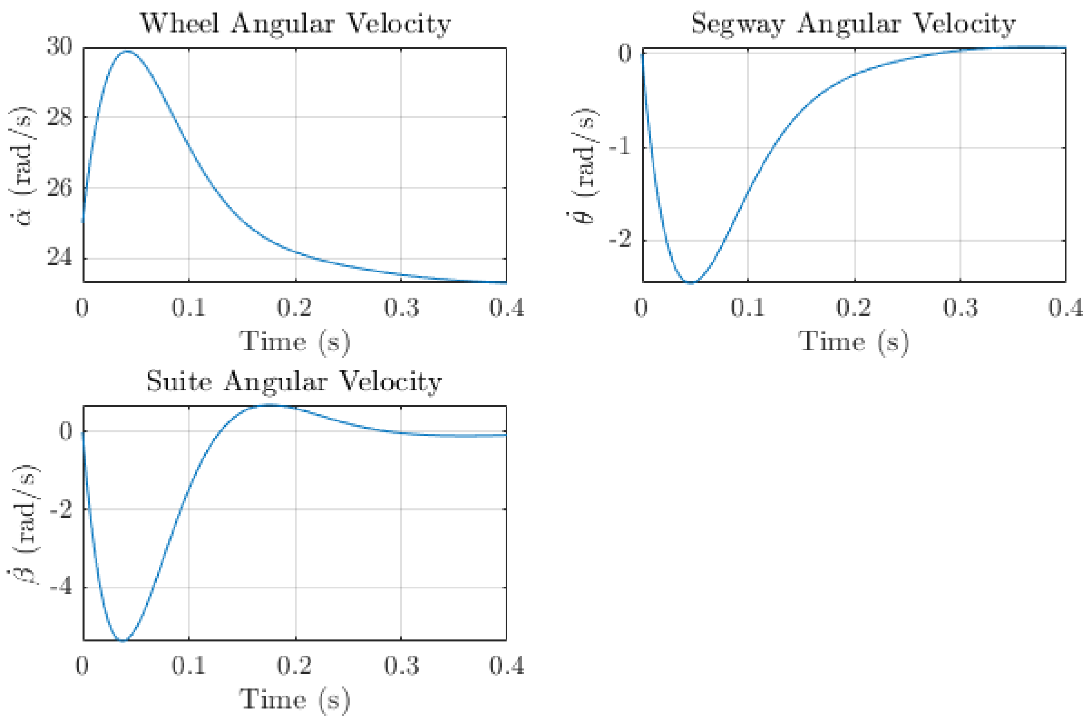

Figure 7: Initial condition response to `x0 = [0.3; 0.3; 25; 0; 0]` - Speeds.

5 Conclusion

The graphs of our initial condition show that the Linear Quadratic Regulator works quite well for our segway-suite system. To reiterate, the maximum values that are input in the Q and R matrices are just a starting point, not a strict rule. Iterating on the parameters allowed for better performance, and surely there are further optimizations that can ameliorate the response of our system. For example, these space-state trajectories take for granted that real world actuators cannot immediately go from idle to active, there is some delay.

In addition, there are many terms in the dynamics equations that end up dropping out due to linearization: while this is mostly justified when operating around the equilibrium configuration, this system could benefit from a higher fidelity model, and maybe linearizations around a few different points. This would allow the Loomo greater flexibility and give better insight on the limits of our control scheme.

6 Project Tasks

Allison: dynamics, control, Matlab implementation, simulation, report
Isabella: dynamics, simulation, presentation slides, report
Susan: dynamics, control, Matlab implementation, report
Jason: dynamics, result analysis, presentation slides, report
George: dynamics, result analysis, presentation slides, report

7 References

[1] *Jackal UGV - Small Weatherproof Robot - Clearpath.* URL: `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`.

[2] Jesus Tordesillas et al. "FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments". In: (May 2020). URL: `https://arxiv.org/pdf/2001.04420.pdf`.

[3] *Loomo spec.* URL: `https://www.segway.com/loomo/loomo-spec/`.

[4] Neville Hogan. "Segway™ Dynamics". In: (Oct. 2020).

[5] Alexander Bogdanov. "Optimal Control of a Double Inverted Pendulum on a Cart". In: (Dec. 2004).

[6] Tony Huang. *RPLIDAR-A3 Laser Range Scanner$_R$obotLaserRangeScanner*. URL: `https://www.slamtec.com/en/Lidar/A3`.

8 Appendix

8.1 Derivation of Lagrangian Mechanics

To get our equations of motion via Lagrange mechanics, we need to calculate the total potential and kinetic energy of our system. We start by determining the center of mass locations and their respective derivatives.

$$x_c = R\alpha + L\sin\theta$$

$$y_c = R + L\cos\theta$$
$$\dot{x}_c = R\dot{\alpha} + L\dot{\theta}\cos\theta$$
$$\dot{y}_c = -L\dot{\theta}\sin\theta$$

$$x_s = R\alpha + (L+l_p)\sin\theta + l_s\sin(\theta+\beta)$$
$$y_s = R + (L+l_p)\cos\theta + l_s\cos(\theta+\beta)$$
$$\dot{x}_s = R\dot{\alpha} + (L+l_p)\dot{\theta}\cos\theta + l_s(\dot{\theta}+\dot{\beta})\cos(\theta+\beta)$$
$$\dot{y}_s = -(L+l_p)\dot{\theta}\sin\theta - l_s(\dot{\theta}+\dot{\beta})\sin(\theta+\beta)$$

To find the total potential energy we broke the system down into three parts, the wheel, the segway, and the sensor suite. Their respective potential energies are PE_w, PE_{segway}, and PE_s.

$$PE_w = 0$$
$$PE_{segway} = mgL\cos\theta$$
$$PE_s = m_s g y_s = m_s g[R + (L+l_p)\cos\theta + l_s\cos(\theta+\beta)]$$

$$\begin{aligned} PE &= PE_w + PE_{segway} + PE_s \\ &= 0 + mgL\cos\theta + m_s g[R + (L+l_p)\cos\theta + l_s\cos(\theta+\beta)] \\ &= m_s gR + [mgL + m_s g(L+l_p)]\cos\theta + m_s g l_s\cos(\theta+\beta) \end{aligned} \tag{1}$$

Similarly to how we found the total potential energy, to find the total kinetic energy we first calculated KE_w, KE_{segway}, and KE_s.

$$\begin{aligned} KE_w &= \frac{1}{2}m_w\dot{x}^2 + \frac{1}{2}I_w\dot{\alpha}^2 \\ &= \frac{1}{2}m_w(R\dot{\alpha})^2 + \frac{1}{2}\left[\frac{1}{2}m_w R^2\right]\dot{\alpha}^2 \\ &= \frac{3}{4}m_w R^2\dot{\alpha}^2 \end{aligned} \tag{2}$$

$$\begin{aligned}
KE_{segway} &= \frac{1}{2}m(\dot{x}_c^2 + \dot{y}_c^2) + \frac{1}{2}I\dot{\theta}^2 \\
&= \frac{1}{2}m\left((R\dot{\alpha} + L\dot{\theta}\cos\theta)^2 + (-L\dot{\theta}\sin\theta)^2\right) + \frac{1}{2}\left[\frac{1}{3}mL^2\right]\dot{\theta}^2 \\
&= \frac{1}{2}mR^2\dot{\alpha}^2 + mLR\dot{\alpha}\dot{\theta}\cos\theta + \frac{1}{2}mL^2\dot{\theta}^2\cos^2\theta + \frac{1}{2}mL^2\dot{\theta}^2\sin^2\theta + \frac{1}{6}mL^2\dot{\theta}^2 \\
&= \frac{1}{2}mR^2\dot{\alpha}^2 + mLR\dot{\alpha}\dot{\theta}\cos\theta + \frac{2}{3}mL^2\dot{\theta}^2
\end{aligned} \tag{3}$$

$$\begin{aligned}
KE_s &= \frac{1}{2}m(\dot{x}_s^2 + \dot{y}_s{}^2) + \frac{1}{2}I_s\dot{\beta}^2 \\
&= \frac{1}{2}m((R\dot{\alpha} + (L+l_p)\dot{\theta}\cos\theta + l_s(\dot{\theta}+\dot{\beta})\cos(\theta+\beta))^2 \\
&+ (-(L+l_p)\dot{\theta}\sin\theta - l_s(\dot{\theta}+\dot{\beta})\sin(\theta+\beta))^2 + \frac{1}{2}\left[m_s l_s^2\right]\dot{\beta}^2 \\
&= \frac{1}{2}m_s\Big[R^2\dot{\alpha}^2 + (L+l_p)^2\dot{\theta}^2\cos^2\theta + l_s^2(\dot{\theta}+\dot{\beta})^2\cos^2(\theta+\beta) + 2R(L+l_p)\dot{\alpha}\dot{\theta}\cos\theta \\
&+2Rl_s(\dot{\theta}+\dot{\beta})\dot{\alpha}\cos(\theta+\beta) + 2l_s(L+l_p)\dot{\theta}(\dot{\theta}+\dot{\beta})\cos\theta\cos(\theta+\beta) \\
&+ (L+l_p)^2\dot{\theta}^2\sin^2\theta + 2l_s(L+l_p)\dot{\theta}(\dot{\theta}+\dot{\beta})\sin\theta\sin(\theta+\beta) + l_s^2(\dot{\theta}+\dot{\beta})^2\sin^2(\theta+\beta) + \frac{1}{2}m_s l_s^2\dot{\beta}^2 \\
&= \frac{1}{2}m_s R^2\dot{\alpha}^2 + \frac{1}{2}m_s(L+l_p)^2\dot{\theta}^2 + \frac{1}{2}m_s l_s^2(\dot{\theta}+\dot{\beta})^2 + m_s R(L+l_p)\dot{\alpha}\dot{\theta}\cos\theta \\
&+ m_s R l_s(\dot{\theta}+\dot{\beta})\dot{\alpha}\cos(\theta+\beta) + m_s l_s(L+l_p)\dot{\theta}(\dot{\theta}+\dot{\beta})\cos\beta + \frac{1}{2}m_s l_s^2\dot{\beta}^2
\end{aligned} \tag{4}$$

$$\begin{aligned}
KE &= KE_w + KE_{segway} + KE_s \\
&= \left[\frac{1}{2}m_s R^2 + \frac{3}{4}m_w R^2 + \frac{1}{2}mR^2\right]\dot{\alpha}^2 + \left[\frac{1}{2}m_s l_s^2\right](\dot{\theta}+\dot{\beta})^2 + \left[m_s R(L+l_p) + mRL\right]\dot{\alpha}\dot{\theta}\cos\theta \\
&+ \left[m_s l_s(L+l_p)\right]\dot{\theta}(\dot{\theta}+\dot{\beta})\cos\beta + \left[m_s R l_s\right]\dot{\alpha}(\dot{\theta}+\dot{\beta})\cos(\theta+\beta) \\
&+ \left[\frac{1}{2}m_s(L+l_p)^2 + \frac{2}{3}mL^2\right]\dot{\theta}^2 + \left[\frac{1}{2}m_s l_s^2\right]\dot{\beta}^2
\end{aligned} \tag{5}$$

After finding the potential and kinetic energy of the system, we defined constants to simplify the rest of our dynamics analysis. These constants are as follows:

Kinetic Energy Constants

$$a = \frac{1}{2}m_s R^2 + \frac{3}{4}m_w R^2 + \frac{1}{2}mR^2$$

$$b = \frac{1}{2}m_s l_s^2$$

$$c = m_s R(L+l_p) + mRL$$

$$d = m_s l_s(L+l_p)$$

$$e = m_s R l_s$$

$$f = \frac{1}{2}m_s(L+l_p)^2 + \frac{2}{3}mL^2$$

$$h = \frac{1}{2}m_s l_s^2$$

Potential Energy Constants

$$v = m_s g R$$
$$w = mgL + m_s g(L + l_p)$$
$$z = m_s g l_s$$

Plugging the kinetic energy constants into (5) results in:

$$KE = [a]\dot{\alpha}^2 + [b](\dot{\theta} + \dot{\beta})^2 + [c]\dot{\alpha}\dot{\theta}\cos\theta + [d]\dot{\theta}(\dot{\theta} + \dot{\beta})\cos\beta + [e]\dot{\alpha}(\dot{\theta} + \dot{\beta})\cos(\theta + \beta) + [f]\dot{\theta}^2 + [h]\dot{\beta}^2 \tag{6}$$

Plugging the potential energy constants into (1) results in:

$$PE = [v] + [w]\cos\theta + [z]\cos(\theta + \beta) \tag{7}$$

Now that we have finalized our kinetic and potential energy equations, we can compute our Lagrangian and follow Lagrange's method.

$$\begin{aligned}\mathcal{L} &= KE - PE \\ &= [a]\dot{\alpha}^2 + [b](\dot{\theta} + \dot{\beta})^2 + [c]\dot{\alpha}\dot{\theta}\cos\theta + [d]\dot{\theta}(\dot{\theta} + \dot{\beta})\cos\beta + [e]\dot{\alpha}(\dot{\theta} + \dot{\beta})\cos(\theta + \beta) + [f]\dot{\theta}^2 + [h]\dot{\beta}^2 \\ &- [v] - [w]\cos\theta - [z]\cos(\theta + \beta)\end{aligned} \tag{8}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = f_i$$

Where f_i is the generalized force (torque) associated with $\dot{q}_i$ and q_i.

Generalized coordinates $\{\alpha, \theta, \beta\}$

α-coordinate:

$$\frac{\partial \mathcal{L}}{\partial \dot{\alpha}} = 2a\dot{\alpha} + c\dot{\theta}\cos\theta + e(\dot{\theta} + \dot{\beta})\cos(\theta + \beta)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{\alpha}} = 2a\ddot{\alpha} + c\left[\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right] + e\left[(\ddot{\theta} + \ddot{\beta})\cos(\theta + \beta) - (\dot{\theta} + \dot{\beta})^2\sin(\theta + \beta)\right]$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = 0$$

$$\tau_\alpha = 2a\,[\ddot{\alpha}] + c\left[\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right] + e\left[(\ddot{\theta} + \ddot{\beta})\cos(\theta + \beta) - (\dot{\theta} + \dot{\beta})^2\sin(\theta + \beta)\right]$$

θ-coordinate:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} &= 2b(\dot{\theta} + \dot{\beta}) + c\dot{\alpha}\cos\theta + d(2\dot{\theta} + \dot{\beta})\cos\beta + e\dot{\alpha}\cos(\theta + \beta) + 2f\dot{\theta} \\ \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} &= 2b(\ddot{\theta} + \ddot{\beta}) + c\left[\ddot{\alpha}\cos\theta - \dot{\alpha}\dot{\theta}\sin\theta\right] + d\left[(2\ddot{\theta} + \ddot{\beta})\cos\beta - (2\dot{\theta} + \dot{\beta})\dot{\beta}\sin\beta\right] \\ &+ e\left[\ddot{\alpha}\cos(\theta + \beta) - \dot{\alpha}(\dot{\theta} + \dot{\beta})\sin(\theta + \beta)\right] + 2f\ddot{\theta} \\ \frac{\partial \mathcal{L}}{\partial \theta} &= -c\dot{\alpha}\dot{\theta}\sin\theta - e\dot{\alpha}(\dot{\theta} + \dot{\beta})\sin(\theta + \beta) + w\sin\theta + z\sin(\theta + \beta)\end{aligned} \tag{9}$$

$$\tau_\theta = 2b\left[(\ddot{\theta} + \ddot{\beta})\right] + c\,[\ddot{\alpha}\cos\theta] + d\left[(2\ddot{\theta} + \ddot{\beta})\cos\beta - (2\dot{\theta} + \dot{\beta})\dot{\beta}\sin\beta\right] + e\,[\ddot{\alpha}\cos(\theta + \beta)] + 2f\left[\ddot{\theta}\right] - w\,[\sin\theta] - z\,[\sin(\theta + \beta)]$$

β-coordinate:

$$\frac{\partial \mathcal{L}}{\partial \dot{\beta}} = 2b(\dot{\theta}+\dot{\beta}) + d\dot{\theta}\cos\beta + e\dot{\alpha}\cos(\theta+\beta) + 2h\dot{\beta}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{\beta}} = 2b(\ddot{\theta}+\ddot{\beta}) + d\left[\ddot{\theta}\cos\beta - \dot{\theta}\dot{\beta}\sin\beta\right] + e\left[\ddot{\alpha}\cos(\theta+\beta) - \dot{\alpha}(\dot{\theta}+\dot{\beta})\sin(\theta+\beta)\right] + 2h\left[\ddot{\beta}\right]$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = -d\left[\dot{\theta}(\dot{\theta}+\dot{\beta})\sin\beta\right] - e\left[\dot{\alpha}(\dot{\theta}+\dot{\beta})\sin(\theta+\beta)\right] + z\left[\sin(\theta+\beta)\right]$$

$$\tau_\beta = 2b\left[\ddot{\theta}+\ddot{\beta}\right] + d\left[\ddot{\theta}\cos\beta + \dot{\theta}^2\sin\beta\right] + e\left[\ddot{\alpha}\cos(\theta+\beta)\right] + 2h\left[\ddot{\beta}\right] - z\left[\sin(\theta+\beta)\right]$$

8.2 MATLAB source code

8.2.1 Controller + Plotting

```
% Group 6 Final Project
% Sensor Stabilization on a Segway Robot Using LQR Control
% A. Lenhard, I. Montanaro, S. Ni, J. Santillan, G. Chen

clear all, close all,  clc

% Parameters
m = 15;       % segway shaft mass (kg)
L = 0.25;     % axle to mass center (m)
m_w = 4;      % 2 wheels mass (kg)
R = 0.14;     % wheel radius (m)
Bw = 0.1;     % bearing friction wheel motor (N-m/rad/s)
Br = 0.02;    % rolling friction (N-m/rad/s)
Bs = 0.1;     % bearing friction suite motor (N-m/rad/s)
g = 9.8;      % gravity (m/s^2)

m_s = 1;      % sensor suite mass (kg)
l_s = 0.15;    % sensor suite length (m)
l_p = 0.05;    % length from segway com to sensor suite (m)

% ----- DYNAMICS -----

% coefficients from dynamics analysis for simplicity
a = R^2*(1/2*m_s+1/2*m+3/4*m_w);
b = 1/2*m_s*l_s^2;
c = R*(m_s*(L+l_p)+m*L);
d = m_s*l_s*(L+l_p);
e = R*m_s*l_s;
f = 1/2*m_s*(L+l_p)^2+2/3*m*L^2;
h = 1/2*m_s*l_s^2;
w = g*(m_s*(L+l_p)+m*L);
z = m_s*g*l_s;

H = [1 0;
     -1  0;
     0   1];                    % input weighting
```

```
I = [2*a c+e       e;
     c+e 2*(b+d+f) 2*b+d;
     e   2*b+d     2*(b+h)];  % inertia matrix
k = [0  0;
     -w -z;
     0  -z];                  % gravity "stiffness", (N-m/rad)
Bf = [(Br+Bw) -Bw 0;
      (-Bw)    Bw  0;
      0     0   Bs];          % damping B matrix(N-m/rad/s)


% states: segway angle from upright        (theta)
%         sensor suite angle from upright (beta+theta)
%         segway wheel angular velocity    (omega_w)
%         segway angular velocity          (omega)
%         sensor suite angular velocity    (omega_s)

A = [0 0 0 1 0;
     0 0 0 1 1;
     inv(I)*k -inv(I)*Bf];
B = [zeros(2,2); inv(I)*H];
C = [0 1 0 0 0];  % output: (beta + theta) angle of sensor suite from upright

% ----- LQR CONTROLLER ------
t_p_max = 0.1;    % maximum deviation of angle from upright (rad)
tb_p_max = 0.03;  % maximum deviation of theta&beta (rad)
tau_max = 5;     % maximum motor torque (N-m)
tau_s_max = 1;    % maximum motor torque for suite (N-m)

Q = [1/t_p_max^2 0 0 0 0;
     0 1/tb_p_max^2 0 0 0;
     zeros(3, 5)];
R = [1/tau_max^2 0;
     0 1/tau_s_max^2];

[K, S, ep] = lqr(A, B, Q, R);
% first row of K - gains for wheel motor torque input
% second row of K - gains for suite motor torque input

sys = ss(A-B*K, B, C, 0);

% ----- PLOTTING -----

set(0,'defaultTextInterpreter','latex');
set(0,'defaultLegendInterpreter','latex')
set(0,'defaultAxesTickLabelInterpreter','latex');

% initial condition response

x0 = [0.3; 0.3; 25; 0; 0];  % intial condition

[y,t,x] = initial(sys, x0, 0.4);

save('sim_data.mat', 'y', 't', 'x');
```

```

plot_states(t, x, K, 3, "Initial Condition Response");

% ----- PLOTTING HELPER FUNCTION -----

function [] = plot_states(t, x, K, fig_num, suptitle)
    mode = 'paper'; % either 'paper' or 'slide'
    if (mode == 'paper')
        r = 4; c = 2;
    elseif (mode == 'slide')
        r = 2; c = 4;
    end

    figure(fig_num);
    sgtitle(suptitle)

    subplot(r,c,1);
    plot(t, x(:,1));
    grid on

    title('Segway Angle from Upright');
    ylabel('$\theta$(rad)');
    xlabel('Time (s)');

    subplot(r,c,2);
    plot(t, x(:,2));
    grid on
    title('Suite Angle from Upright');
    ylabel('$\beta + \theta (rad)$');
    xlabel('Time (s)');

    subplot(r,c,3);
    plot(t, (K(1,:)*x')');
    grid on
    title('Wheel Motor Torque');
    ylabel('$\tau_\alpha$ (N$\cdot$m)');
    xlabel('Time (s)');

    subplot(r,c,4);
    plot(t, (K(2,:)*x')');
    grid on
    title('Suite Motor Torque');
    ylabel('$\tau_\beta$ (N$\cdot$m)');
    xlabel('Time (s)');

    subplot(r,c,5);
    plot(t, x(:,3));
    grid on
    title('Wheel Angular Velocity');
    ylabel('$\dot{\alpha}$ (rad/s)');
    xlabel('Time (s)');

    subplot(r,c,6);
    plot(t, x(:,4));
    grid on
    title('Segway Angular Velocity');
    ylabel('$\dot{\theta}$ (rad/s)');
    xlabel('Time (s)');

    subplot(r,c,7);
    plot(t, x(:,5));
    grid on
    title('Suite Angular Velocity');
    ylabel('$\dot{\beta}$ (rad/s)');
    xlabel('Time (s)');
end
```

8.2.2 Simulation

```
% Group 6 Final Project
% Leveling a sensor suite on a segway
% A. Lenhard, I. Montanaro, S. Ni, J. Santillan, G. Chen

clear all, close all,  clc

load('sim_data.mat','x', 't'); % data obtained from controller design file
tspan = t;
states = x;

figure('units','normalized','outerposition',[0 0 1 1]);
animateSol(tspan, states)

function animateSol(tspan, states)
    li_w = 0.076;        % lidar width (m)
    li_h = 0.041;        % lidar height (m)
    L = 0.25;            % axle to mass center (m)
    l_s = 0.15;          % length from segway axle to com of sensor suite(m)
    l_sb = l_s-(li_h/2); %length of sensor suite connection rod (m)
    l_p = 0.05;          % length from segway com to sensor suite (m)
    R = 0.14;            % wheel radius (m)

    hold on
    % different lines/circles to animate
    ground =plot([0],[0],'k','LineWidth',1);
    TH = 0:.1:2.1*pi; % theta values to plot circle for wheel
    wheel = plot([0] + R*cos(TH), [R] + R*sin(TH),'k','LineWidth',2);
    shaft = plot([0],[0],'LineWidth',3);

    % T-shape for sensor suite platform
    suite = plot([0],[0], 'r', 'LineWidth',2);
    suite_bar = plot([0], [0], 'r', 'LineWidth',2);

    % lines for lidar
    lidar_top = plot([0],[0], 'm', 'LineWidth',1);
    lidar_bot = plot([0],[0], 'm', 'LineWidth',1);
    lidar_left = plot([0],[0], 'm', 'LineWidth',1);
    lidar_right = plot([0],[0], 'm', 'LineWidth',1);

    % initializing figure labels
    xlabel('x (m)'); ylabel('y (m)');
    h_title = title('t = 0.0 s');
    axis equal
    axis([-0.25 2 -0.1 1]);
    set(ground, 'XData', [-1 2.5]);
    set(ground, 'YData', [0 0]);
    x = 0; % initializing x position
    t_diff = tspan(2)-tspan(1);

    % Step through and update animation
    for i = 1:length(tspan)
        % An initial pause so the figure has time to fully open before the
```

```
            animation starts running
        if(i == 2)
            pause(0.7)
        end
        % current time and state values
        t = tspan(i);
        current_state = states(i,:);
        theta = current_state(1);
        beta_theta = current_state(2);
        dalpha = current_state(3);
        x = x + (dalpha*R*t_diff);

        % getting current positions of key points from current states
        axle = [x R]; % center of wheel
        shaft_top = axle + [2*L*sin(theta)  2*L*cos(theta)]; % top of segway
            shaft
        suite_connection = axle + [(L+l_p)*sin(theta) (L+l_p)*cos(theta)]; %
            suite platform location on shaft
        suite_top = suite_connection + [sin(beta_theta)*l_sb cos(beta_theta)*
            l_sb]; %suite rod top endpoint
        suite_left = suite_top +[cos(beta_theta)*(li_w*1.25)/2 -sin(beta_theta
            )*(li_w*1.25)/2]; % suite T left endpoint
        suite_right = suite_top +[-cos(beta_theta)*(li_w*1.25)/2 sin(
            beta_theta)*(li_w*1.25)/2]; % suite T right endpoint

        % four corners of lidar: top left, top right, bottom left, bottom
            right
        lidar_tl = suite_connection + [sin(beta_theta)*(l_sb+li_h) cos(
            beta_theta)*(l_sb+li_h)]+[cos(beta_theta)*(li_w/2) -sin(beta_theta
            )*(li_w/2)];
        lidar_tr = suite_connection + [sin(beta_theta)*(l_sb+li_h) cos(
            beta_theta)*(l_sb+li_h)]+[-cos(beta_theta)*(li_w/2) sin(beta_theta
            )*(li_w/2)];
        lidar_bl = suite_top +[cos(beta_theta)*(li_w/2) -sin(beta_theta)*(li_w
            /2)];
        lidar_br = suite_top +[-cos(beta_theta)*(li_w/2) sin(beta_theta)*(li_w
            /2)];

        set(h_title,'String',  sprintf('t = %.2f s',t) ); % update title

        % update all lines and circles for animation
        set(wheel,'XData',[axle(1)+R*cos(TH)]);
        set(wheel,'YData',[axle(2)+R*sin(TH)]);

        set(shaft,'XData',[axle(1) shaft_top(1)]);
        set(shaft,'YData',[axle(2) shaft_top(2)]);

        set(suite,'XData',[suite_connection(1) suite_top(1)]);
        set(suite,'YData',[suite_connection(2) suite_top(2)]);

        set(suite_bar,'XData',[suite_left(1) suite_right(1)]);
        set(suite_bar,'YData',[suite_left(2) suite_right(2)]);

        set(lidar_top, 'XData', [lidar_tl(1) lidar_tr(1)]);
        set(lidar_top, 'YData', [lidar_tl(2) lidar_tr(2)]);

        set(lidar_bot, 'XData', [lidar_bl(1) lidar_br(1)]);
        set(lidar_bot, 'YData', [lidar_bl(2) lidar_br(2)]);

        set(lidar_left, 'XData', [lidar_bl(1) lidar_tl(1)]);
        set(lidar_left, 'YData', [lidar_bl(2) lidar_tl(2)]);

        set(lidar_right, 'XData', [lidar_br(1) lidar_tr(1)]);
        set(lidar_right, 'YData', [lidar_br(2) lidar_tr(2)]);

        pause(.05) % slow down animation to see progression
    end
end
```