MAE 3260 Final Group Work: Exploring a System of Interest

Report

Title: The Dog Within You

Topic of Interest: Spot (Boston Dynamic's Robot Dog)

Abstract

Our group will be modeling the system of a robot dog. Specifically, we will be focusing on the dynamics and control of one of the legs because each leg is a mechanical duplicate of each other, except for their coordinates. We will develop a simplified rigid body model of one leg joint and implement a proportional derivative controller to study fundamental control behavior, tuning Kp and Kd gains, and evaluating robustness under external disturbance (uneven terrain). Building on this, we will expand to full leg control by coordinating the hip and knee joints and incorporating kinematic transformations to achieve desired foot positions. A CAD model with quarter scale proportions will be used to obtain mass properties such as mass distribution, center of mass, and inertia tensors to ensure accurate physical representation. Overall, we will explore joint control, coordination of multiple joints, and we will integrate MATLAB simulation, CAD modeling, and control theory in the context of legged robotics.

Students/Roles:

Student	**Task/Role**	**Portfolio**
Name	1-3 sentence summary of student's work and key take-aways	Link to results in student's individual portfolio
Swecha	CAD model and extraction of relevant parameters for state-space modeling	https://github.com/Cornell-MAE-UG/fall-2025-portfolio-Swecha314/blob/main/_projects/System%20Dynamics%20Analysis%20of%20Boston%20Dynamic's%20Spot
Jason Bak	Matlab script and explanation of code, PD controller	https://cornell-mae-ug.github.io/fa25-portfolio-jasonbak02/SpotLeg
Liz	Developed the dynamic equations and PD control model for the robot-dog leg, showing how proportional and derivative gains affect stability and tracking performance	

List of MAE 3260 concepts or skills used in this group work:
- MATLAB
- Fusion 360
- PD Controllers (Kp and Kd gain)

Pages 2-9 include two pages for each student:
- Summarize technical work, including figures
- For references, use numbers/brackets (e.g. [1]) in text with list - IEEE citation format.

- Students can combine section (i.e. 4 pages for students A and B), but will get the same grade

Swecha Agarwal
Fusion 360 Model

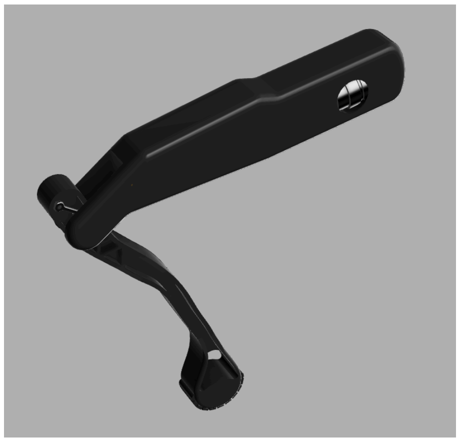
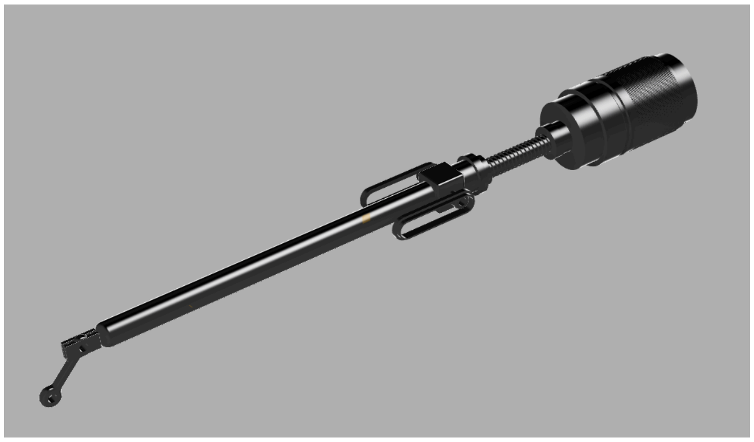



Figure 1 **Figure 2**

The modeled subsystem is a single leg inspired by Boston Dynamic's Spot robot and consists of a thigh body, shin body, foot, a hip joint, and a knee joint actuated by a linear actuator. The hip is driven by a rotary, custom made motor at the top of the thigh, while the knee rotates through a threaded screw-driven linear actuator mounted along the thigh and connected to a crank on the shin. Together, these two actuators create a two degree freedom leg capable of hip and knee flexibility, and their configuration matches the two link planar model used in our system dynamics analysis. The shin and the foot form the lower segment, which is the link positioned away from the robot's main body. The linear actuator is necessary in order to reduce inertia located far away from the hip joint, putting a lower torque load on the hip motor.

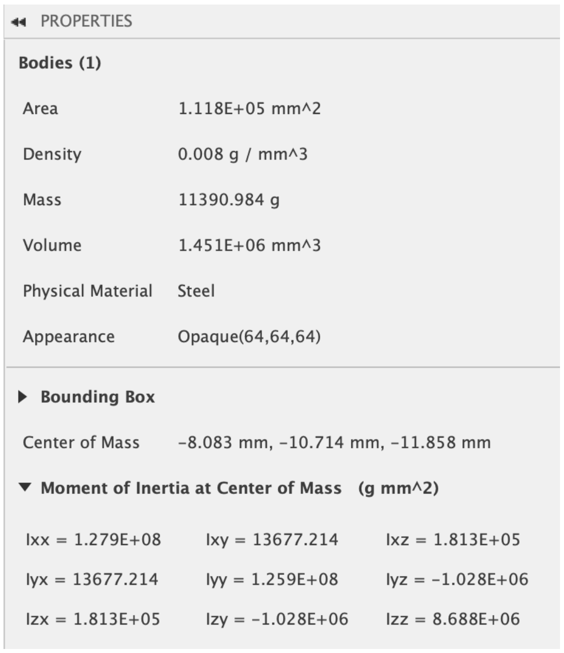
The thigh and shin bodies were designed with curved and filleted geometries to lower mass while keeping stiffness around high stress regions, such as the joint attachment areas. The foot incorporates grooves for traction and a rounded base that supports a range of contact angles. Internal cavities were added to reduce mass without compromising load-bearing regions. The actuator brackets were positioned to make sure that the system matches the planar assumptions used later in the state space modeling.

Although the CAD geometry is fully three dimensional, the leg is modeled as a planar two-link system with two generalized coordinates: theta hip and theta knee. In system dynamics terms. The leg behaves as a two DoF mechanical system with (1) states defined by joint angles and joint angular velocities, (2) inputs defined by the torques generated by the hip and knee actuators, and (3) outputs defined by the joint angles and foot position. The CAD model is necessary to provide the physical parameters that determine the inertia matrix M(theta), gravitational terms G(theta), and the torques. The linear actuator determines how the applied motor torque becomes knee torque. By supplying accurate link masses and inertias, the CAD model ensures that the state space model of the leg reflects an accurate version of the real Spot.

Parameter Extraction from Fusion 360

Fusion 360 was used to obtain geometric and initial parameters used in the system dynamics model and the MATLAB control implementation. Only the out of plane inertia Izz is used because planar rotation occurs about that axis. Since the actuator is rigidly mounted along the thigh, its mass is grouped with the thigh for modeling. To the right is an example of the properties extracted from Fusion 360 per body.

The following parameters form the physical foundation of the two-link dynamic model used in subsequent sections. They directly influence the joint torques required for movement and the behavior of the PD controlled knee and hip joints simulated in MATLAB.


PROPERTIES

Bodies (1)

Area	1.118E+05 mm^2
Density	0.008 g / mm^3
Mass	11390.984 g
Volume	1.451E+06 mm^3
Physical Material	Steel
Appearance	Opaque(64,64,64)

Bounding Box

Center of Mass −8.083 mm, −10.714 mm, −11.858 mm

Moment of Inertia at Center of Mass (g mm^2)

Ixx = 1.279E+08	Ixy = 13677.214	Ixz = 1.813E+05
Iyx = 13677.214	Iyy = 1.259E+08	Iyz = −1.028E+06
Izx = 1.813E+05	Izy = −1.028E+06	Izz = 8.688E+06

Component	Mass (kg)	Length (m)	Inertia Izz (kg * mˆ2)
Thigh	11.39	0.374	2.94*10ˆ-2
Shin	2.26	0.376	1.05*10ˆ-2
Linear Actuator	1.39	-	-

Elizabeth Liku
Dynamic Modeling and PD Control Design

The robot dog leg was modeled as a planar rigid-body system consisting of a hip joint and a knee joint. Each segment was treated as a rigid link with mass and inertia values later obtained from the CAD model. The joints were modeled as revolute joints with viscous damping included to represent frictional effects, and actuation was represented as a direct torque input at each joint. The generalized coordinates for the system were defined as the hip angle θ_1 and the knee angle θ_2. Using rigid-body dynamics, the equations of motion for the two-link system were written in standard robotic form as:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) = \tau$$

where $M(\theta)$ is the inertia matrix, $C(\theta, \dot{\theta})\dot{\theta}$ contains the Coriolis and centrifugal terms, $G(\theta)$ represents the gravitational torques, and τ is the vector of applied actuator torques.

For controller design and for implementation in MATLAB, the knee joint was also modeled as a single-degree-of-freedom rotational system. This simplified model is expressed as

$$I\ddot{\theta} + b\dot{\theta} + mglsin(\theta) = \tau$$

where I is the rotational inertia, b is the damping coefficient, m is the link mass, g is gravitational acceleration, and l is the distance to the center of mass. This equation captures the combined effects of inertia, damping, gravity, and applied control torque, and directly supports the spring–mass–damper interpretation used in the simulation.

Each joint was regulated using a proportional–derivative feedback controller. The control law applied at the joint was defined as

$$\tau = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta})$$

where K_p is the proportional gain, K_d is the derivative gain, θ_d is the desired joint angle, and $\dot{\theta}_d$ is the desired joint velocity. The proportional term produces a restoring torque based on joint position error, while the derivative term provides damping based on joint velocity error. This PD control structure allows the joint to track a desired trajectory while remaining stable when disturbances are applied, which is consistent with the behavior observed in the MATLAB simulations.

To analyze system stability and dynamic response, the nonlinear gravitational term was linearized about small joint angles using the approximation sin(θ)≈θsin(θ)≈θ. This resulted in the linearized plant model

$$I\ddot{\theta} + b\dot{\theta} + mgl\theta = \tau$$

Taking the Laplace transform of this equation gives us the open-loop transfer function:

$$\frac{\Theta(s)}{T(s)} = \frac{1}{Is^2+bs+mgl}$$

When the PD controller is included in the feedback loop, the closed-loop transfer function becomes

$$\frac{\Theta(s)}{\Theta_d(s)} = \frac{K_d s + K_p}{Is^2 + (b+K_d)s + (mgl+K_p)}$$

This expression shows how the proportional gain affects tracking stiffness and how the derivative gain influences system damping. Increasing K_p improves tracking accuracy but may cause oscillations if K_d is too small, while increasing K_d increases damping and improves stability.

The control system follows a standard feedback structure in which the desired joint trajectory is compared with the measured joint angle using encoder feedback to generate a tracking error. This error is processed by the PD controller to generate actuator torque, and the resulting joint motion is continuously fed back to regulate the system. This structure directly supports the trajectory tracking and disturbance rejection tests performed in MATLAB.

The dynamic equations and control laws derived in this modeling framework were implemented directly in the MATLAB simulation. After realistic mass and inertia values were obtained from the CAD model, these parameters were substituted into the equations to ensure physical accuracy. The controller gains were then tuned using these updated parameters to evaluate tracking performance and disturbance response. This modeling and control framework provided the mathematical foundation that supports the simulation results presented in the following section.

After first analyzing the knee joint as a single-degree-of-freedom system, these same PD control ideas were extended to the full hip–knee leg model to better represent how the robot dog actually moves. In this case, motion at the hip directly affects the behavior of the knee due to dynamic coupling between the two joints. When the hip moves quickly, it introduces additional inertial effects that the knee controller must react to, which explains why some overshoot and lag were observed in the MATLAB results. This showed that tuning the PD gains for one joint alone is not always enough when multiple joints are moving together. By using the full two-link dynamic equations, we were able to better understand how the joints interact and why stronger damping was sometimes needed at the knee for stability. This also helped justify the higher derivative gains used during the disturbance tests. Overall, extending the controller to the coupled hip–knee system made the simulation more realistic and provided a better understanding of how coordinated joint control is required for stable and smooth leg motion.

Jason Bak
MATLAB Logic

Upon further inspection, we realized that Spot's leg acts similarly to a spring-mass damper system, where the goal is to correct the motion caused by some disturbance input. In this case, our "disturbance" is the energy supplied to drive the hip joint. The linear actuator then responds with a torque to reangle the shin joint in order to achieve a specific foot position. For this model, the linear actuator behaves as both the damper and spring. If Spot were to land on a rough and uneven surface, it would need some way to dissipate the energy to stabilize itself instead of bouncing uncontrollably.

When writing the MATLAB script, I chose to implement a PD controller to model the effects of the linear actuator. This is so the proportional and derivative aspects of the controller would provide a restoring force proportional to both the position and velocity of the thigh joint. In doing this, we can construct a simpler model showing how the shin joint rotates in response to the rotation of the thigh joint.

To achieve this, I specified a “desired” knee angle, or my reference trajectory. Given an input torque, I would have the shin joint respond and rotate itself to get as close to this angle as possible:

```
theta_knee_desired = theta_knee_base + compensation_gain * thigh_error;
```

Where `theta_knee_base` represents the nominal walking trajectory for our robot dog defined by a sinusoidal function. Compensation gain is a constant tested through trial and error and multiplied by the thigh error, which is the difference between the actual and expected thigh angles.

I then defined an “actual” knee angle to represent the angle of the knee after our PD controller applies torque. The goal of our program is to get the actual knee angle as close as possible to the desired knee angle, however dynamics, inertia, and disturbance often prevents this from happening.

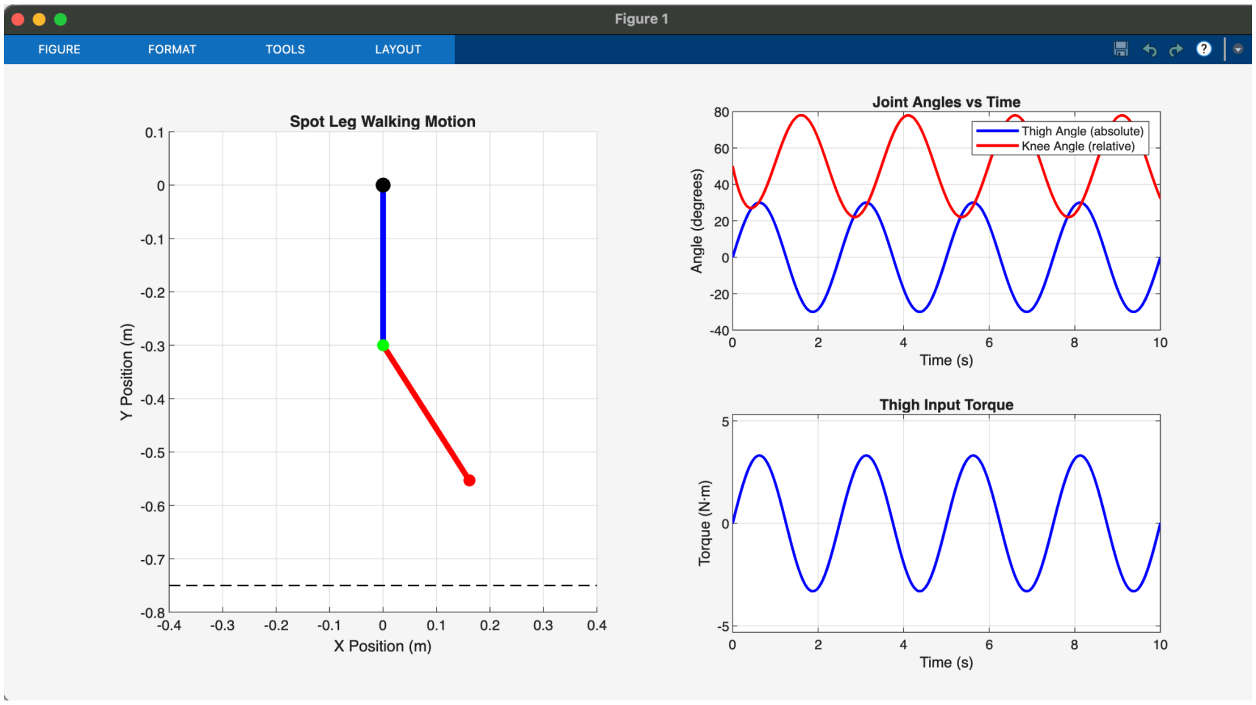
At each timestep, we identify the error, or difference between the desired and actual knee angle:

```
error = theta_knee_desired - theta_knee(i);
```

A positive error suggests that the knee must bend more, and a negative error suggests that the knee must extend more. At the end of the program, I take the average of the absolute value of these errors in order to tune my gain values (more on this later). The absolute value is used so that positive and negative errors do not cancel each other out. A 0° error would suggest that the controller has perfect tracking of the knee which is impossible to achieve in real life. A 1°-3° error would suggest excellent tracking and a very responsive controller. As the mean tracking error increases, we can state that the controller is less responsive than it was before.

Testing

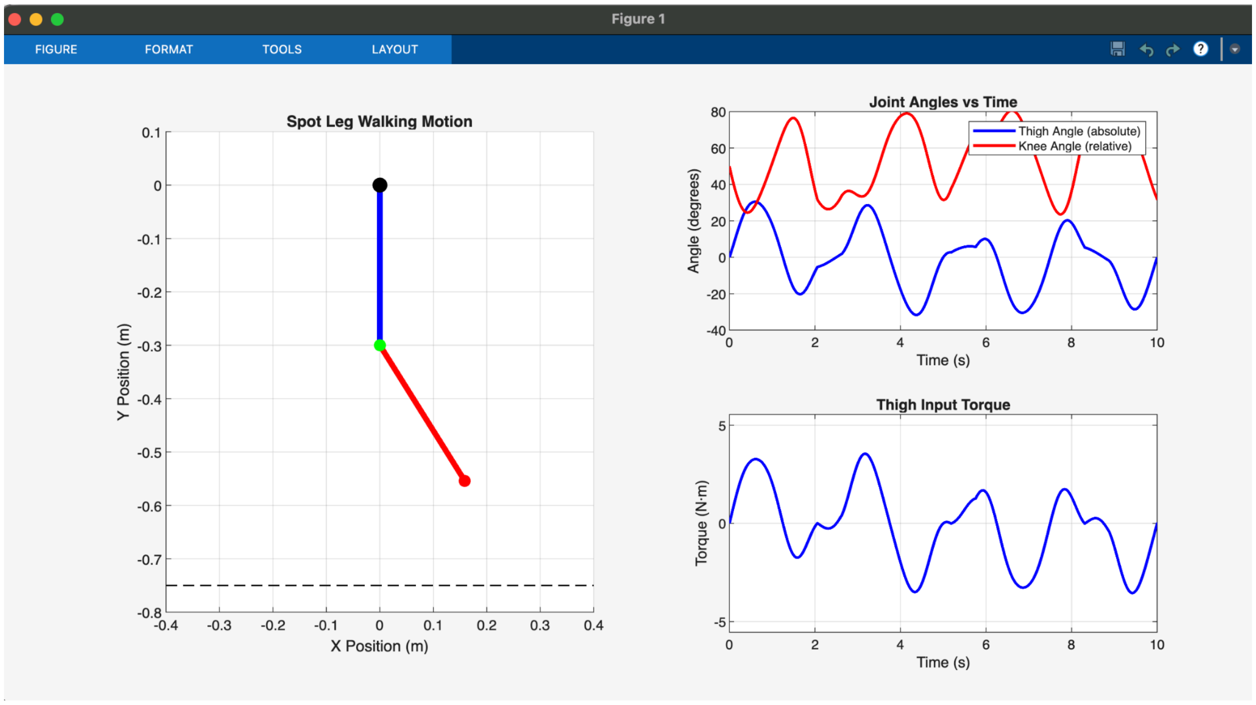
My first test involved a steady sinusoidal response. After tuning the Kd and Kp values, I achieved these results:



```
=== PD Controller Performance ===
Kp: 80.00, Kd: 20.00
Mean tracking error: 2.79 degrees
Max knee angle: 78.02 degrees
Min knee angle: 22.09 degrees
Knee angle range: 55.93 degrees
```

We can see that our mean tracking error is below 3° which suggests a responsive controller. We can then examine our plots to look for trends. Although not perfectly aligned, we can see that the thigh and knee angle try to maintain an inverse relationship – if the thigh rotates clockwise, the shin will rotate counterclockwise. We also take note of our maximum and minimum knee angle which has a difference of ~50°. I initially thought this angle was pretty high, but after walking around for a bit, I realized this knee angle actually made sense for how much the thigh angle was changing.

I then decided to take my testing to the next level by giving an unpredictable disturbance torque. Looking at the thigh input torque graph below, we can see an unsteady sinusoidal shape with unpredictable torque amplitudes that vary with time. This would be a more realistic simulation that would model unpredictable situations our robot dog might find itself in. After tuning the Kp and Kd, I achieved these results:



```
=== PD Controller Performance ===
Kp: 150.00, Kd: 35.00
Mean tracking error: 8.54 degrees
Max knee angle: 80.61 degrees
Min knee angle: 23.60 degrees
Knee angle range: 57.00 degrees
```

As we can see, the mean tracking error for this test is almost triple that of our first test which suggests a less responsive controller. However, for the sharp and unpredictable changes in torque, I would call this an acceptable error. We can also observe that the inverse relationship between the knee and thigh angle is maintained throughout this test as well. Another similarity we can observe is that the range of the knee angle stays around 50°, which suggests that my controller works properly!

Tuning Kp and Kd

After tuning my gain values for a while, I came across a fairly useful interpretation. Our Kp value responds to the position of our knee and tries to correct our position by applying a correction torque. Using a Kp value of 150 Nm is the same thing as saying "for every degree of error we have, apply 150 Nm of torque." This is how our knee is able to track the desired angle, which is constantly changing because of our walking motion.

On the other hand Kd tracks our angular speed and acts similar to a damper in a mass spring damper system. Using a Kd value of 35 would mean applying 35 Nm of torque to slow down the rotation for every °/s too fast or slow we are rotating. This prevents the Kp from causing oscillations that are too wild.

Citations:

1. Boston Dynamics. (n.d.). *Geometry and frames*. Retrieved from https://dev.bostondynamics.com/docs/concepts/geometry_and_frames
2. Miller, S. (2025). *Running robot model in Simscape* (Version 24.2.2.7) [Computer software]. GitHub. Retrieved September 21, 2025, from https://www.mathworks.com/matlabcentral/fileexchange/64237-running-robot-model-in-simscape
3. U.S. Patent Application No. US20180169868A1. (n.d.). Retrieved from https://patents.google.com/patent/US20180169868A1/en