Modelling, Design, and Control of a Spherical Rolling Robot

by

Anthony D. Stuart

Submitted to the
Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2019

Signature redacted

Signature of Author: _____
Department of Mechanical Engineering
May 10, 2019

Signature redacted

Certified by: _____
Sangbae Kim
Associate Professor
Thesis Supervisor

Signature redacted

Accepted by: _____
Maria Yang, PhD
Professor of Mechanical Engineering
Undergraduate Officer

Modelling, Design, and Control of a Spherical Rolling Robot

by

Anthony D. Stuart

Submitted to the Department of Mechanical Engineering
on May 9, 2019 in Partial Fulfillment of the
Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

ABSTRACT

Spherical rolling robots offer a number of advantages over traditional wheeled robots. Their geometry allows for increased mobility and the inability to tip over, while their enclosed design provides protection for the electronic and mechanical components driving the robot. However, these advantages come with significantly more complex system dynamics, with design parameters impacting system performance in sometimes non-intuitive ways.

This thesis explores the effect of different design parameters such as sphere mass and center of mass location on the performance of a spherical rolling robot. A kinematic and dynamic model of the system was created, along with a first iteration of the mechanical design to generate realistic starting values for the simulation. The system dynamics were linearized and used to design a linear quadratic regulator (LQR) controller. Sphere mass and center of mass location were varied across a range of values and simulated to analyze the system's ability to quickly converge to position and velocity set points. This study found that increasing the mass of the sphere resulted in a linear increase in settling time, while increasing the radial distance between the center of mass and the sphere caused an exponential decrease in settling time, up until a critical point.

Thesis Supervisor: Sangbae Kim
Tile: Associate Professor

Acknowledgements

I would like to thank the people who have helped me along the journey of writing my Bachelor's thesis. First, I would like to thank Joao Ramos for providing me with guidance and advice during the course of this thesis, as well as for giving me the opportunity to UROP with him for the past two years. Working with you has been invaluable to my development as an engineer. I would also like to thank Professor Sangbae Kim for giving me the opportunity to work on this project and for serving as my UROP faculty advisor.

I would also like to thank my family. To my mother and father, thank you for being an invaluable support system during my time at MIT. I would not have made it to this point without all the sacrifices that you made to help me succeed.

Table of Contents

List of Figures

Section 1: Introduction

1.1: Spherical Robot Overview

Spherical rolling robots are interesting devices that offer a number of advantages over traditional wheeled robots. One large benefit of a spherical design is its inherent stability. A spherical rolling robot cannot be tipped over, allowing it to endure large disturbances while remaining operational. Spherical robots can also be designed to be holonomic, or capable of moving instantaneously in any direction. The enclosed nature of ball-shaped robots provides the components within the drive mechanism protection from external contaminants and impact loads while also providing a smooth and safe exterior for humans to interact with. These advantages can make spherical robots attractive solutions for applications involving traversing rough terrain, surviving falls and impacts, and interacting with humans [1].

1.2: Mechanism Types

Spherical robots typically move by shifting the location of their center of mass within the sphere, creating a torque that causes the robot to roll in the desired direction. This type of design, referred to as Barycenter Offset Design (BCO) [2], can be accomplished with a number of different mechanisms, and a single optimal design has yet to emerge. One solution is to use a wheel-based design, in which a wheeled platform drives up the inside of the sphere, shifting the center of mass and causing the robot to rotate. Additional spherical bearings or wheels are often included to prevent the drive unit from tipping within the ball, and a gyroscope or IMU can also be added to allow for closed loop control. This type of mechanism is used in the Star Wars BB8 droid and Sphero 2 toys produced by Sphero, Inc, an example of which can be seen in Figure 1-1 below [3].

Figure 1-1: Sphero 2, a spherical rolling robot toy that uses internal drive wheels to move https://qph.fs.quoracdn.net/main-qimg-895c3fd6ab66022c7b021a76198311f6-c z

Figure 1-2: GroundBot, a pendulum driven security robot designed by Rotundus inc. [4]

Another common BCO design is a pendulum-based design, where an actuator is mounted to a weighted pendulum which rotates to shift the center of mass of the system. One example of a pendulum-driven system is GroundBot by Rotundus, a spherical robot designed for security applications seen in Figure 1-2 above [4]. Pendulum-driven robots have the advantage of being cheap and easy to build, but have difficulty driving up steep inclines.

While BCO designs account for the majority of spherical robots, other propulsion methods exist as well. Shell transformation robots such as Kim's Kisbot [5] are propelled by causing the external sphere of the robot to transform in shape to push the robot in the desired direction. Exploration has also been done into wind-powered spherical robots, such as the tumbleweed-inspired design proposed by NASA seen in Figure 1-4 to utilize the strong winds on the surface of Mars as a power source for next generation rovers [6].

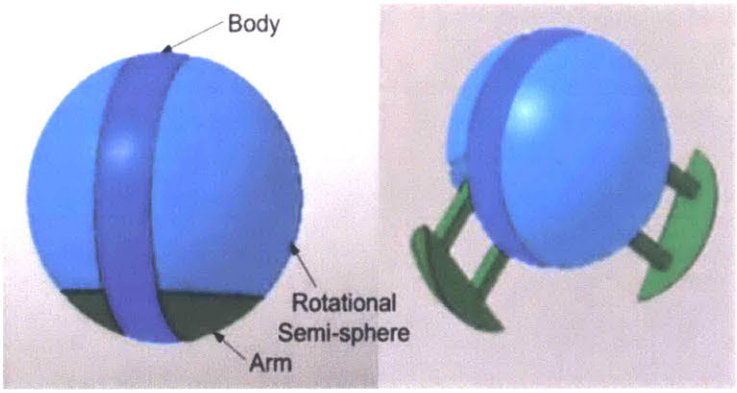

Figure 1- 3: KisBot, a spherical robot propelled by arms that extend and retract from the outer shell [5]

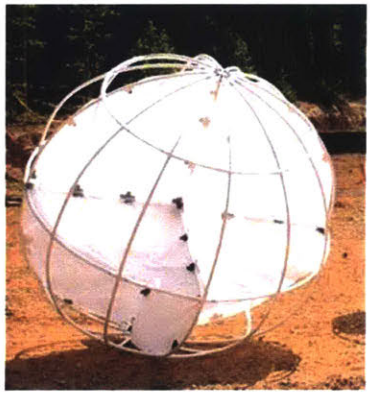

Figure 1- 4: Tumbleweed-inspired wind-powered robot from NASA [7]

1.3: Thesis Purpose

The variety of spherical rolling robot designs illustrates the importance of considering how design parameters impact the performance of the system, as certain parameters such as actuator torque and center of mass location can have tradeoffs in terms of desired system behavior. For instance, designing a spherical robot with strong disturbance rejection requires higher actuator torque (and thus lower speed), while a robot that prioritizes high top speed would require a high speed, low torque actuator. The goal of this thesis is to use system modelling to decide on optimal spherical robot design parameters for different applications. This paper focuses on optimizing the design of a wheel-based BCO robot for disturbance rejection and acceleration, examining how varying different aspects of the design, namely sphere mass and center of mass location, impact these performance areas.

1.4: Thesis Organization

This thesis is organized as follows. Section 2 focuses on the kinematic and dynamic modelling of the spherical robot system. Section 3 focuses on the design of the first iteration of the robot. Section 4 details the linear quadratic regulator (LQR) controller design, and section 5 focuses on the simulation results. Finally, section 6 contains the summary and conclusion.

Section 2: Modelling

2.1: Kinematic Modelling

Prior to designing the specific of the spherical rolling robot, a kinematic and dynamic model was constructed. The model was created with the following assumptions

1. The wheel of the robot rolls without slipping on the inside of the sphere
2. The external sphere rolls in a straight line without slipping on the ground
3. The internal drive mechanism has no frictional losses
4. The inertia of the wheel is negligible

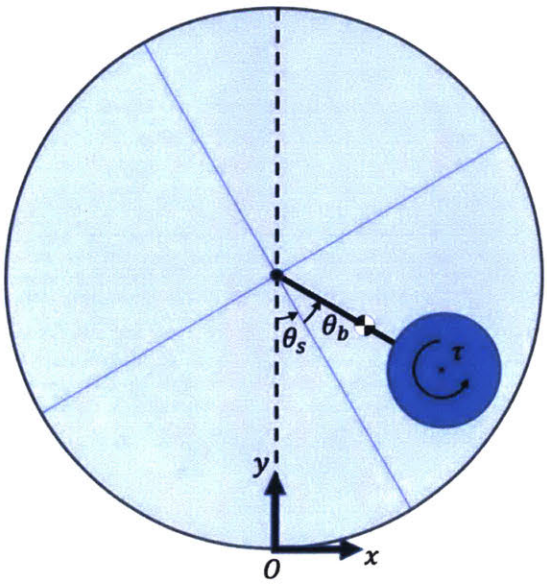

Figure 2-1: Simplified spherical robot model with state variables and actuator torque

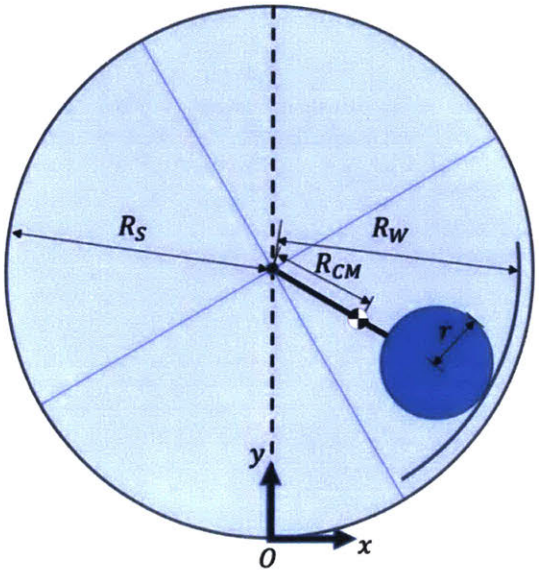

Figure 2-2: Simplified spherical robot model with radii indicated

Variable descriptions

x_s , y_s	Position of sphere with respect to ground reference frame
x_b , y_b	Position of body center of mass with respect to ground frame
θ_s	Angle of reference line on sphere with respect to the y-axis
θ_b	Angle of body with respect to reference line on sphere
m_s	Mass of spherical shell

m_b	Mass of body
I_s	Inertia of sphere
I_b	Inertia of body
R_S	Radius of sphere
R_{CM}	Radius of body center of mass
R_w	Radius at which wheel contacts the inside surface of the sphere
r	Radius of the wheel
g	Gravitational acceleration constant
τ	Applied motor torque acting at the wheel

A simplified model of the spherical robot can be seen in Figure 2-1 and 2-2 above. A fixed two-dimensional coordinate system was established at the ground level. The coordinates of the sphere were defined as (x_s, y_s), and the coordinates of the robot body was defined as (x_b, y_b). These coordinates can be written as:

$$x_s = \theta_s R_s \tag{1}$$

$$y_s = R_s \tag{2}$$

$$x_b = x_s + R_{CM}\sin(\theta_s + \theta_b) \tag{3}$$

$$y_b = y_s - R_{CM}\cos(\theta_s + \theta_b) \tag{4}$$

The velocities of the body were obtained by differentiating equations (3) and (4):

$$\dot{x_b} = \dot{x_s} + R_{CM}\left(\dot{\theta_s} + \dot{\theta_b}\right)\cos(\theta_s + \theta_b) \tag{5}$$

$$\dot{y_b} = R_{CM}\left(\dot{\theta_s} + \dot{\theta_b}\right)\sin(\theta_s + \theta_b) \tag{6}$$

Differentiating equations (5) and (6) yields the system accelerations:

$$\ddot{x_b} = \ddot{x_s} - R_{CM}\left(\dot{\theta_s} + \dot{\theta_b}\right)^2 \sin(\theta_s + \theta_b) + R_{CM}\left(\ddot{\theta_s} + \ddot{\theta_b}\right)\cos(\theta_s + \theta_b) \tag{7}$$

$$\ddot{y_b} = R_{CM}\left(\dot{\theta_s} + \dot{\theta_b}\right)^2 \cos(\theta_s + \theta_b) + R_{CM}\left(\ddot{\theta_s} + \ddot{\theta_b}\right)\sin(\theta_s + \theta_b) \tag{8}$$

2.2: Dynamics Modelling

Equations of motion for the system were derived using the Lagrange Method. The Lagrange is written as follows

$$L = T - P \tag{9}$$

Where T is equal to the total kinetic energy of the system and P is equal to the total potential energy of the system. A simplified model of the system can be seen in Figure 2-1, where the ground is defined as the zero potential energy surface. The total energy values were obtained by calculating the respective energies of the spherical shell and the robot body and summing them together. The kinetic and potential energy of the sphere can be written as follows:

$$T_{sphere} = \frac{1}{2} m_s \dot{x_s}^2 + \frac{1}{2} I_s \dot{\theta_s}^2 \tag{10}$$

$$P_{sphere} = m_s g R \tag{11}$$

For the body, the kinetic and potential energies can be written as:

$$T_{body} = \frac{1}{2} m_b \dot{x_b}^2 + \frac{1}{2} I_b (\dot{\theta_s} + \dot{\theta_b})^2 \tag{12}$$

$$P_{body} = m_b g y_b \tag{13}$$

Summing these values together gives the systems total potential and kinetic energies:

$$T = \frac{1}{2} (m_s \dot{x_b}^2 + m_s \dot{x_s}^2) + \frac{1}{2} \left(I_b (\dot{\theta_s} + \dot{\theta_b})^2 + I_s \dot{\theta_s}^2 \right) \tag{14}$$

$$P = m_s g R + m_b g y_b \tag{15}$$

Plugging equations (14) and (15) into equation (9) returns the Lagrange

$$L = \frac{1}{2} (m_s \dot{x_b}^2 + m_s \dot{x_s}^2) + \frac{1}{2} (I_b (\dot{\theta_s} + \dot{\theta_b})^2 + I_s \dot{\theta_s}^2) - (m_s g R + m_b g y_b) \tag{16}$$

The Lagrange can be used to obtain the systems equation of motion using the relationship:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q_i}} \right) - \frac{\partial L}{\partial q_i} = Q_i \tag{17}$$

Where q_i is the ith generalized coordinate of the system and Q_i is the non-conservative force associated with that coordinate. θ_s and θ_b, the relative angles of the outer sphere and internal body respectively, were selected as generalized coordinates. Since the only input force to the system is from the motor torque τ, Q_s and Q_b are equal to the torques applied by the motor on the sphere and robot body respectively. For a motor torque τ, a contact force of $-\frac{\tau}{r}$ is applied to the bottom of the sphere. After multiplying this force by the sphere's radius, the torque applied to the sphere can be computed as:

$$Q_s = -\frac{\tau}{r} R_s \tag{18}$$

Similarly, the torque applied to the body can be computed as:

$$Q_b = -\frac{\tau}{r}(R_w - R_{cm} - r) \tag{19}$$

By plugging equation (16) and (18) into (17) and evaluating for $q_i = \theta_s$, the equations of motion for the sphere was obtained. Similarly, plugging equation (16) and (19) into (17) with $q_i = \theta_b$ returns the equation of motion for the robot body. These values were evaluated using the MATLAB script found in Appendix A.

Section 3: Mechanical Design

3.1: Design Overview

After constructing a dynamic model of the basic system, an initial design was created. The overall structure is similar to that of most standard BCO wheeled spherical robots, consisting of an external shell, two drive wheels powered by motors that drive up the inside of the shell, an omni-wheel in contact with the top of the shell to prevent the robot from tipping, and the necessary electronics to power the motors. The following design requirements were used to make decisions.

1. The robot's components must be easy to manufacture
2. All components of the robot must fit inside a 12-inch diameter sphere
3. The robot must be able to operate continuously for 1 hour on a single charge
4. The robot must be able to reach a speed of 5 m/s

Components were selected based on their ability to meet the design requirements and how accessible they were. For motor selection, the power required to roll a sphere of mass 5 kg at a speed of 5 m/s with a startup time of 2 seconds was calculated. The kinetic energy of a thin sphere rolling without slipping with the desired properties is written as:

$$K = \frac{5}{6}mv^2 = 104.17\ \text{J} \tag{20}$$

Dividing this by 2 seconds yields a minimum power requirement of

$$P = \frac{\Delta E}{\Delta t} = \frac{104.17\ \text{J}}{2\ \text{s}} = 52.08\ \text{W} \tag{21}$$

Based on this information and considering power losses due to friction, two 90W RE35 Maxon motors and motor controllers were selected. The operation time requirement was used to select a battery. Assuming an operation time of 1 hour at the previously calculated power, a battery specification of 53 Watt-hour can be calculated. To meet this requirement, a 6000 mA-hour 15V

LiPo battery was selected, with an energy of 90 Watt-hours. A NI MyRio was selected as the microcontroller for its ease of use, and the robot structure was built with water jet aluminum components and standoffs. A CAD model of the first iteration of the spherical robot design can be seen below.

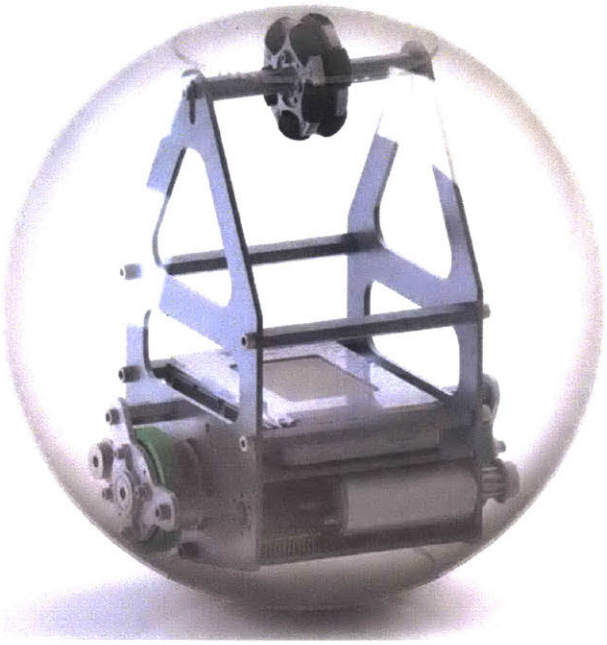

Figure 3-1: Initial CAD model of spherical rolling robot

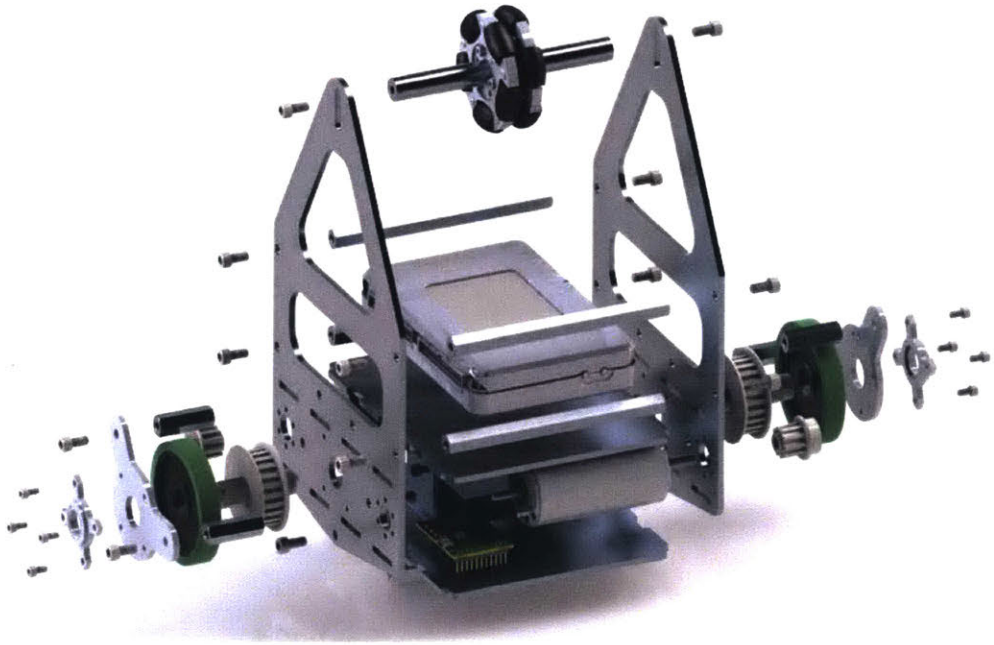

Figure 3-2: Initial CAD model of spherical rolling robot (Exploded View)

The numerical values for the variables used in the dynamic modelling were evaluated in Solidworks.

m_s	Mass of spherical shell = 1.18 kg
m_b	Mass of body = 2 kg
I_s	Inertia of sphere = 0.016 kg-m^2
I_b	Inertia of body = 0.013 kg-m^2
R_S	Radius of sphere = 0.146 m
R_{CM}	Radius of body center of mass = 0.04 m
R_w	Radius at which wheel contacts the inside surface of the sphere = 0.112 m
r	Radius of wheel = 0.03 m

3.2: Variables for Optimization

Once a first design iteration was completed, design variables were selected based on which factors could be reasonably modified while still fitting the size constraints and using the same components. The variables selected were:

R_{CM}	Radius of body center of mass
m_s	Mass of sphere

Where the radius of the center of mass is determined by weight distribution within the robot and the mass of the sphere is changed by modifying sphere thickness and material.

Section 4: Controller Design

4.1 Linear Quadratic Regulator Overview:

A linear quadratic regulator (LQR) feedback controller was selected to control the robot. LQR is advantageous in this application because it is able to find the optimal pole placement for the closed loop system based on a cost function considering the convergence of the state variables and actuator effort. It is also linear, making it relatively easy to implement with low computational power.

4.2: Linearized Equations of Motion

In order to apply LQR, the equations of motion of the system derived in section 2 must be linearized and put into the form.

$$\dot{q} = Aq + B\tau \tag{22}$$

Where $\dot{q}$ is the derivative of the state vector $\dot{q} = [\dot{\theta}_s \quad \dot{\theta}_b \quad \ddot{\theta}_s \quad \ddot{\theta}_b]^T$,and τ is the actuator torque. This was accomplished by computing Jacobian matrices of the state variables and their derivatives.

$$\mathrm{J} = \begin{bmatrix} \frac{\partial \dot{q}_1}{\partial x_1} & \cdots & \frac{\partial \dot{q}_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \dot{q}_m}{\partial x_1} & \cdots & \frac{\partial \dot{q}_m}{\partial x_n} \end{bmatrix} \tag{23}$$

Where $x = q$ for the A matrix and $x = \tau$ for the B matrix. The equations of motion were linearized about the point $\theta_s = \theta_b = \dot{\theta}_s = \dot{\theta}_b = 0$. After setting the Jacobian to 0 at this point, the A and B matrices were computed using MATLAB.

4.3: Change of Basis

While using the relative angles θ_s and θ_b as state variables were convenient for computing the equations of motion, it is more intuitive to track and control absolute positions of the robot body and external sphere with respect to the ground. The variables x and θ, representing the x-

position of the center of the sphere and the absolute angle of the robot body with respect to the vertical were selected. The following relationships can be derived between the absolute states and relative states.

$$x = \theta_s R_s \tag{24}$$

$$\theta = \theta_s + \theta_b \tag{25}$$

Differentiating equations (24) and (25) with respect to time yields the following additional relationships:

$$\dot{x} = \dot{\theta}_s R_s \tag{26}$$

$$\dot{\theta} = \dot{\theta}_s + \dot{\theta}_b \tag{27}$$

These equations were used to construct the following transformation matrix T to covert between $q = [\dot{\theta}_s \quad \dot{\theta}_b \quad \ddot{\theta}_s \quad \ddot{\theta}_b]^T$ and $q_T = [x \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$.

$$T = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & R_s & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{28}$$

The A and B matrices were converted to the new basis.

$$A_T = T^{-1} A T \tag{29}$$

$$B_T = TB \tag{30}$$

4.3: Cost Function and Control Law

Once the linearized dynamics of the system in the desired basis was obtained, a cost function was constructed in the form

$$J = \int_0^\infty q_T^T Q q_T + u^T R u \, dt \tag{31}$$

In this formulation, Q is a matrix that determines the relative importance of each of the states to the controller, and R is a scalar that scales the importance of actuator effort. Increasing a diagonal

value in the Q matrix increases the cost of slow convergence of that respective state, while increasing the value of R increases the cost of high control effort. Since LQR selects gain values that minimize overall cost, increasing Q and R result in the system prioritizing minimizing convergence time and minimizing actuator effort respectively. Since both position and velocity convergence was of roughly equal importance, the following Q matrix was selected for the controller.

$$Q = \begin{bmatrix} \frac{1}{R_S} & & & \\ & 1 & & \\ & & \frac{1}{R_S} & \\ & & & 1 \end{bmatrix} \tag{32}$$

With the entries corresponding to the variables x and $\dot{x}$ being scaled by $1/R_S$ to account for the difference in units. Since actuator effort was also of roughly equal importance, an R value of 1 was selected. Inputting the transformed matrices A_T and B_T into the lqr command in MATLAB along with Q and R yielded the control matrix K.This matrix was used in the control law to determine actuator torque

$$\tau = -K\left(q_T - q_{T,desired}\right) \tag{34}$$

Section 5: Simulation Result

5.1: Setup

To evaluate the performance of the spherical robot as the variables of the sphere's mass m_S and center of mass location R_{CM} were changed, a simulation was constructed. LQR was used to compute a gain matrix K for each combination of system parameters. The MATLAB solver ode45 was used to solve for the value of the transformed state vectors $q_T = [x \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$ at discrete time steps given the equations of motion, control law, and initial condition of the system. An animation that plotted the sphere and internal robot's position at each time step was created to allow for easy visualization of the system's performance, as seen in Figure 5-1 below.

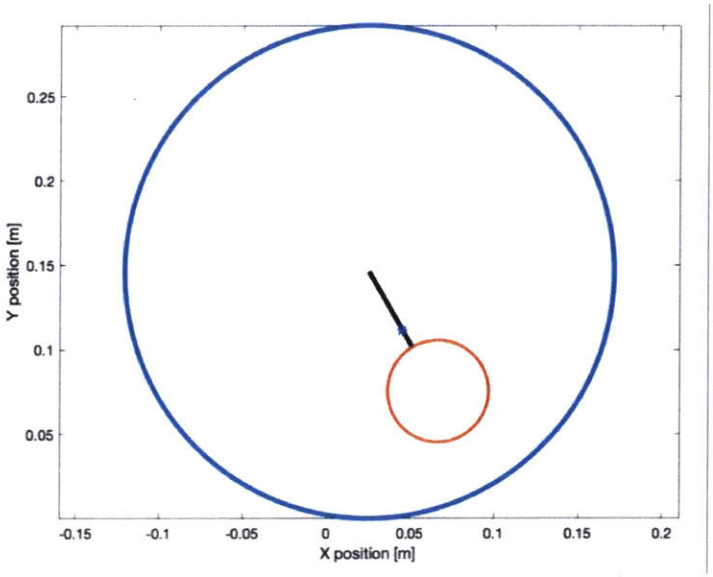

Figure 5-1: MATLAB animation of spherical rolling robot

5.2: Gain Matrix

Gain matrices of the form $K = [K_1 \; K_2 \; K_3 \; K_4]$ corresponding to the transformed state variables $q_T = [x \; \theta \; \dot{x} \; \dot{\theta}]^T$ were calculated for different values of sphere mass and center of mass radius. Mass was varied from 0.5 kg to 8 kg in increments of 0.5 kg, with additional points measured at

0.25 kg and 1.18 kg. A plot of each gain value as a function of sphere mass can be seen in Figure 5-2 below.

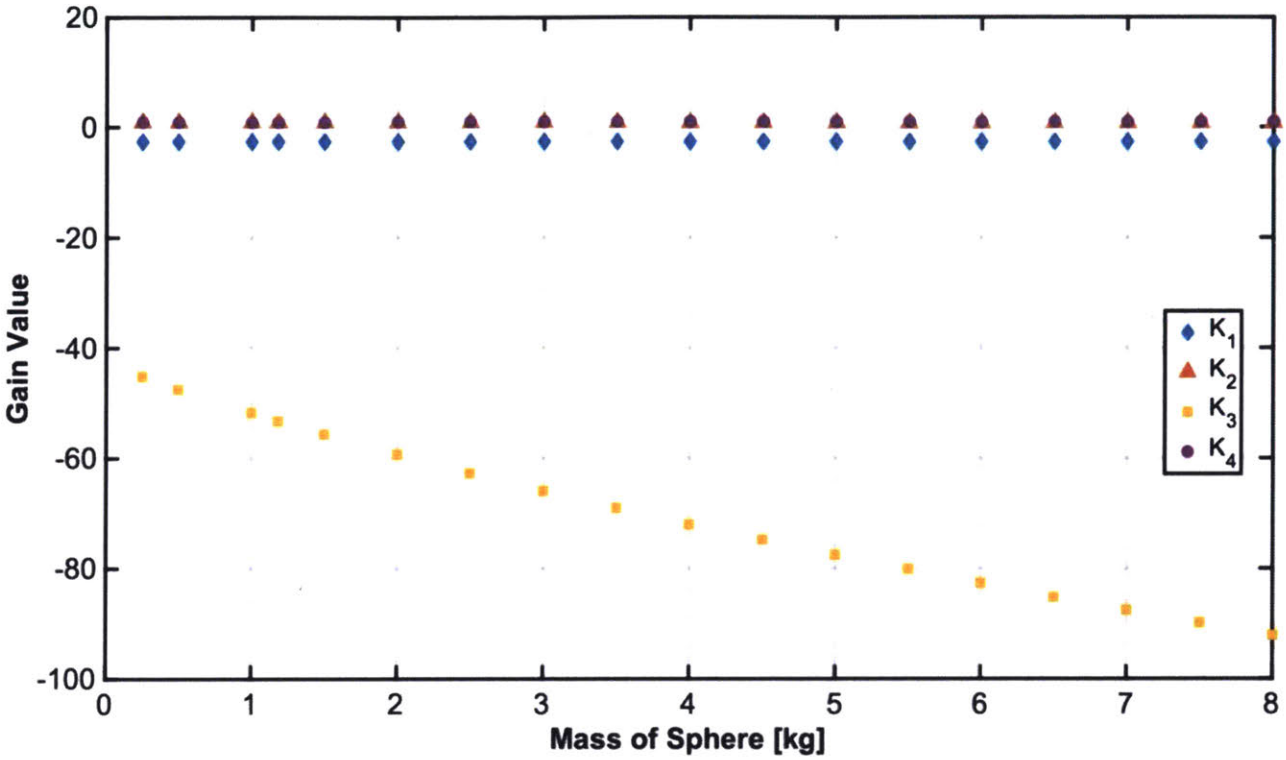

Figure 5-2: Gain matrix values as a function of sphere mass in kg

Gain value K_3, corresponding to the state $\dot{x}$, became consistently more negative for larger values of sphere mass, while the remaining gains stayed roughly constant.

Center of mass location was varied from 0.01 m to 0.08 m in increments of 0.01 m, with additional values recorded at R_{CM}= 0.005 m and 0.0818 m. A plot of each gain value as a center of mass radius can be seen in Figure 5-3 below.

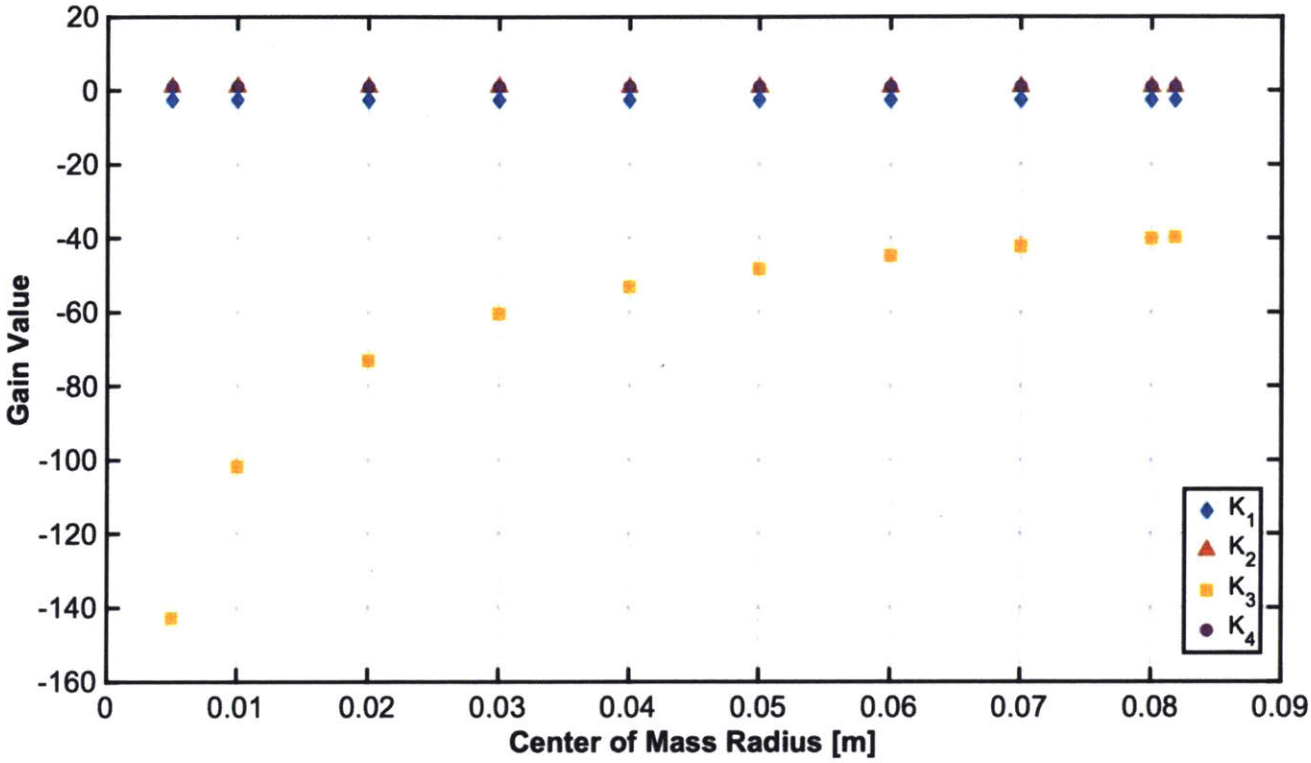

Figure 5-3: Gain matrix values as a function of center of mass radius in meters

Gain value K_3, corresponding to the state $\dot{x}$, increased in value logarithmically and decreased in magnitude as center of mass radius increased, while the remaining gains stayed roughly constant.

5.3: Disturbance Rejection

Once gain matrices were computed, the performance of the system under different criteria was analyzed. The first criteria analyzed was disturbance rejection, specifically the robot's ability to return to the default position of $q_T = [0\ 0\ 0\ 0]^T$ in the shortest time. In order to stay within the linear regime, an initial angular disturbance of 10 degrees was imposed, and the settling time, defined as the time it takes for the steady state error to decrease to a threshold value of 10^{-4}, was measured for sphere masses varying from 0.5 kg to 8 kg in increments of 0.5 kg , with additional

points measured at 0.25 kg and 1.18 kg. Plots of select step responses can be seen in Figure 5-4. A plot of settling time as a function of sphere mass can be seen below in Figure 5-5.

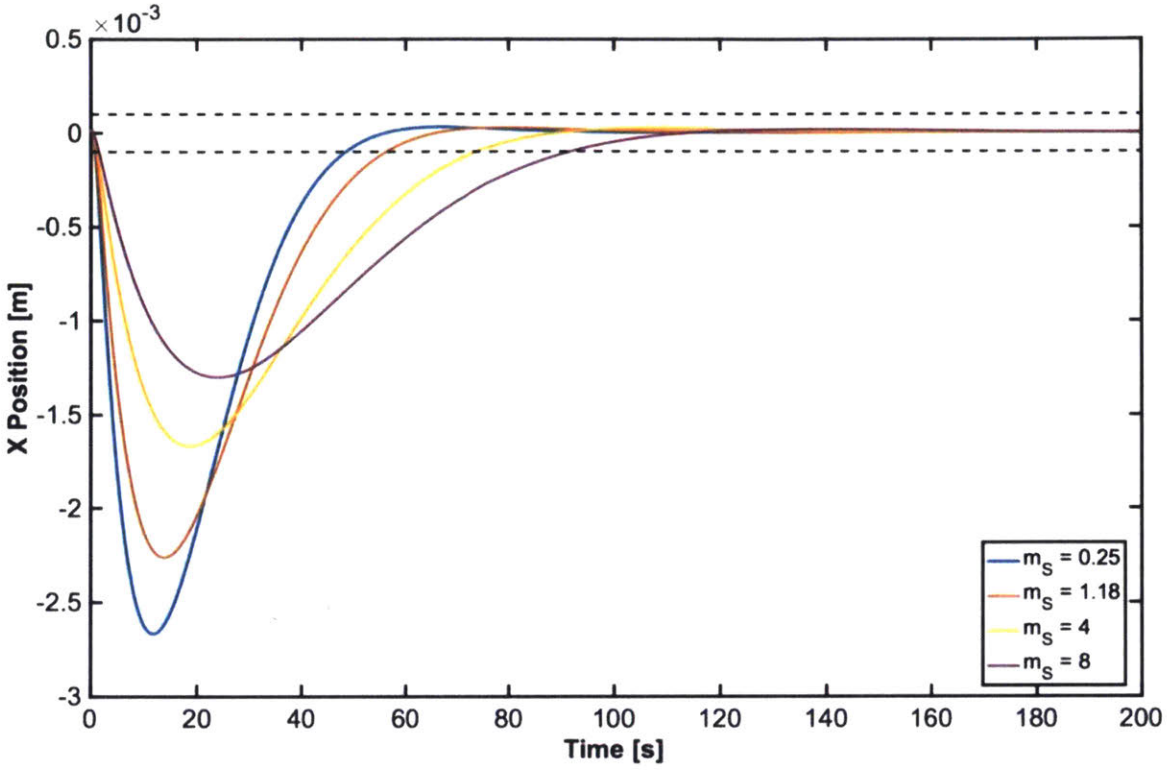

Figure 5-4: X-position step response for select sphere masses. Black lines indicate settling time window of $\pm 10^{-4}m$. Units for m_s are in kg.

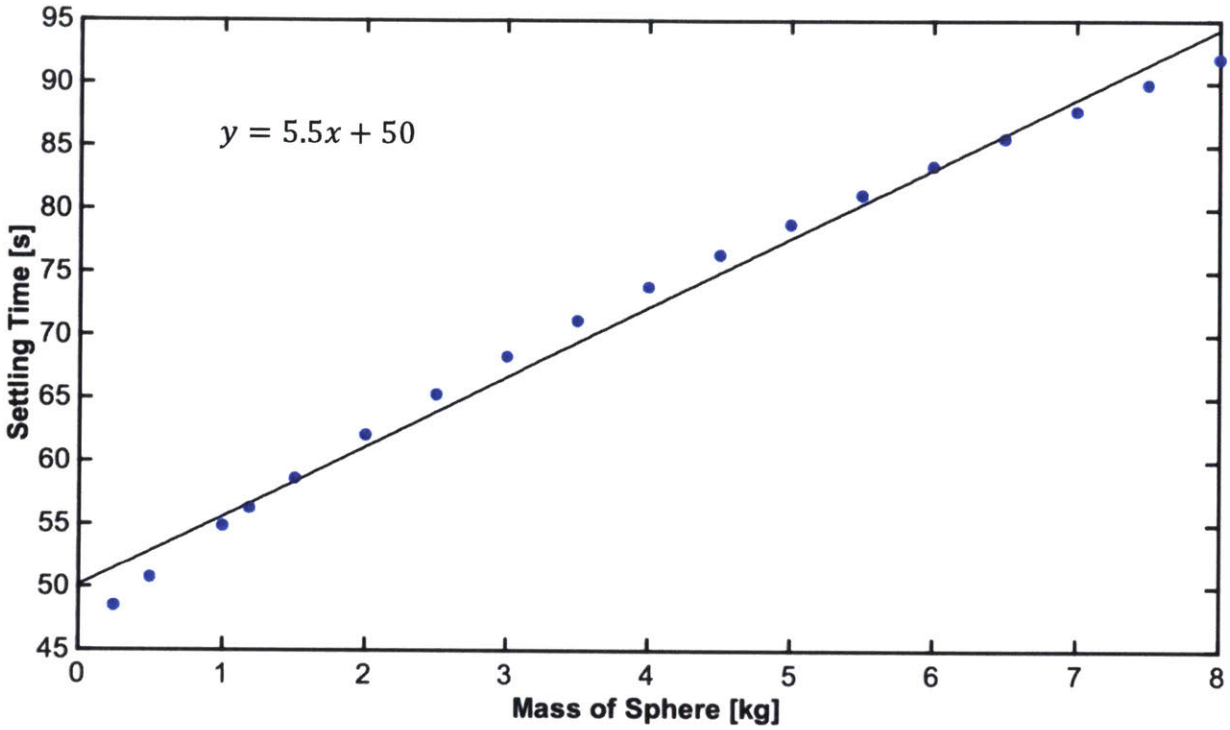

Figure 5-5: Settling time as a function of sphere mass for position control. Black curve represents a linear fit

Settling time was found to increase linearly with sphere mass at a slope of 5.5 s/kg.

Disturbance rejection was also measured as a function of center of mass radius R_{CM}. R_{CM} was varied between 0.01 and 0.09 m with a step size of 0.01 with additional points measured at 0.005 m and 0.0818 m. Plots of select stable step responses and settling times can be seen in Figure 5-4 and Figure 5-5 below.

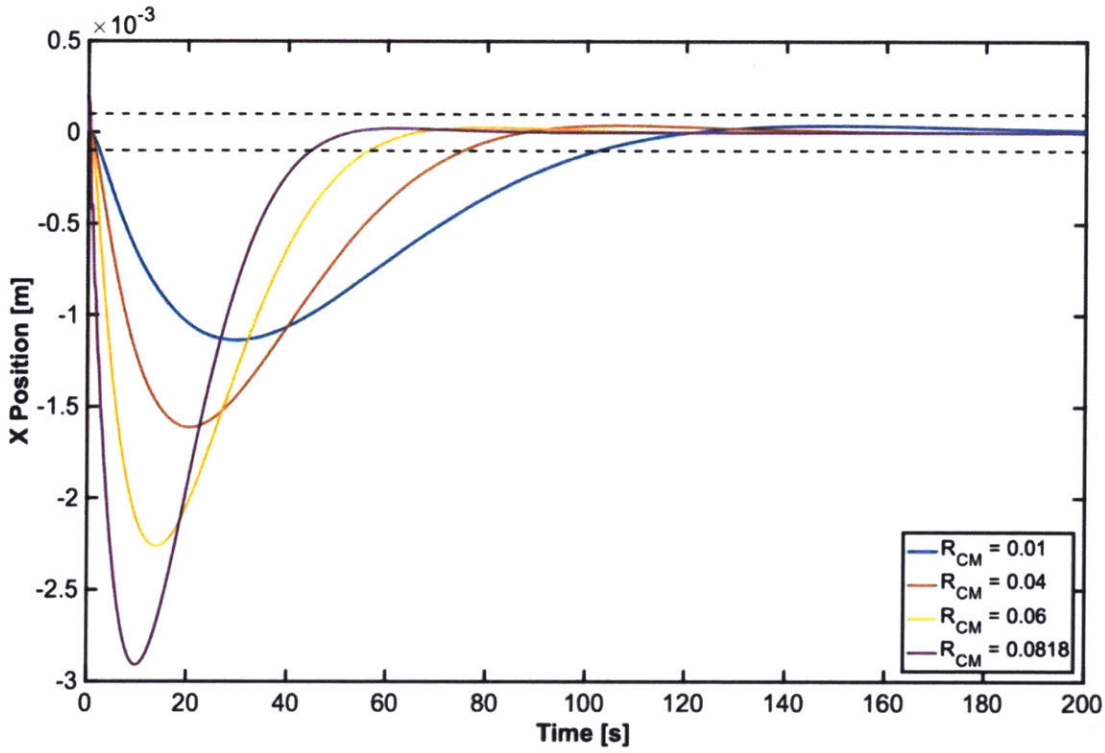

Figure 5-6: X-position step response for center of mass locations. Black lines indicate settling time window of $\pm 10^{-4} m$. Units for R_{CM} are in meters

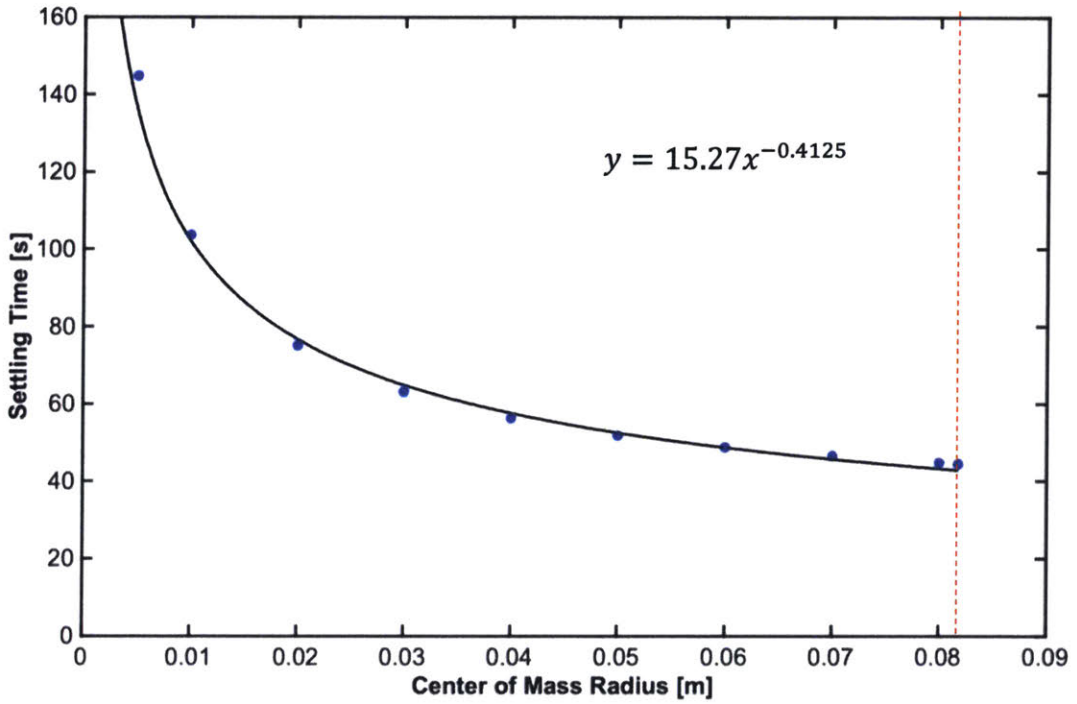

Figure 5-7: Settling time as a function of sphere mass for position control. Black curve represents an exponential fit. Dotted red line represents critical center of mass radius value above which system goes unstable

Running the simulation revealed that increasing R_{CM} above a critical value of 0.0818 m, equal to the center of the wheel location $R_w - r$, resulted in the system going unstable. Increasing R_{CM} prior to this point resulted in an exponential decay in settling time, following a function of $t_s = 15.27{R_{CM}}^{-0.4125}$.

5.4: Velocity Control

The second criteria analyzed was the speed at which the spherical robot was able to converge to a desired linear velocity $\dot{x}$. A desired speed of 0.02 m/s was used for the simulation, and an initial body displacement of 10 degrees was imposed. The settling time of the system was measured for sphere masses varying from 0.5 kg to 8 kg in increments of 0.5 kg , with additional points measured at 0.25 kg and 1.18 kg. Plots of select step responses can be seen in Figure 5-8. A plot of settling time as a function of sphere mass can be seen below in Figure 5-9.

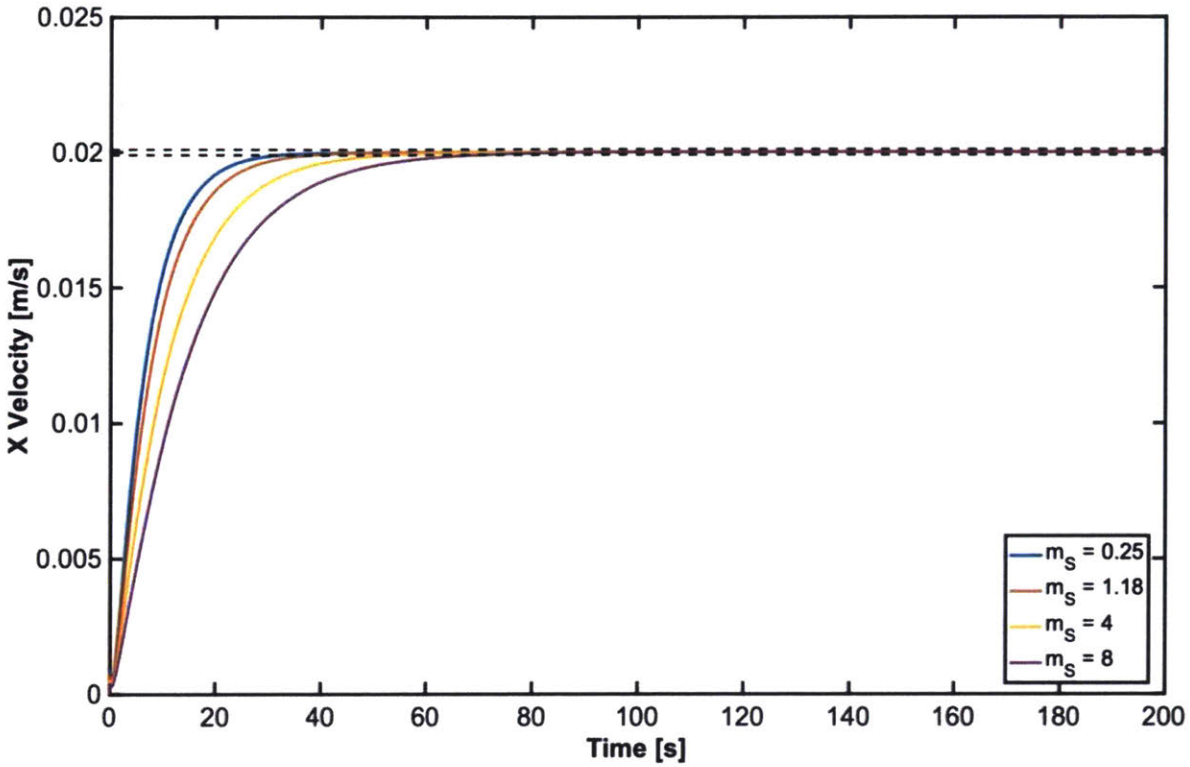

Figure 5-8: X-velocity step response for select sphere masses. Black lines indicate settling time window of $\pm 10^{-4}$m/s

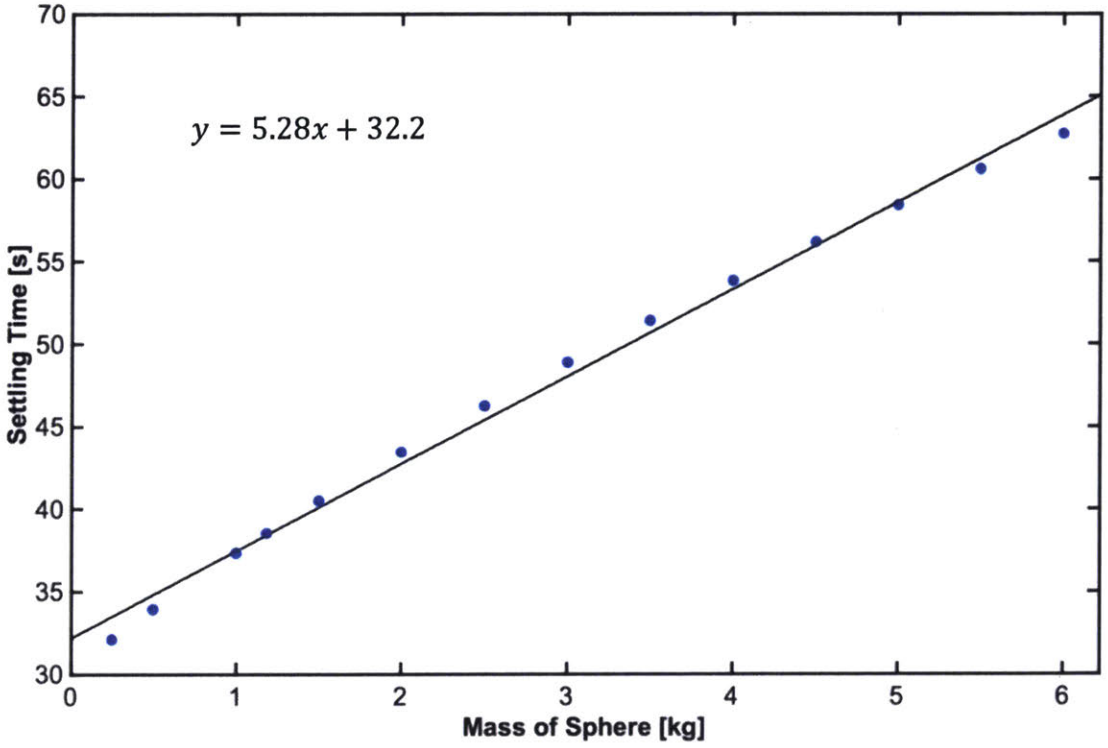

Figure 5-9: Velocity settling time as a function of sphere mass. Black curve represents a linear fit

Increasing sphere mass caused settling time to increase at a linear rate of approximately 5.28 second/kg.

Velocity convergence was also measured as a function of center of mass radius R_{CM}. R_{CM} was varied between 0.01 m and 0.09 m, with additional points measured at 0.005 m and 0.0818 m. Plots of select stable step responses and settling times can be seen in Figure 5-8 and Figure 5-9 below.

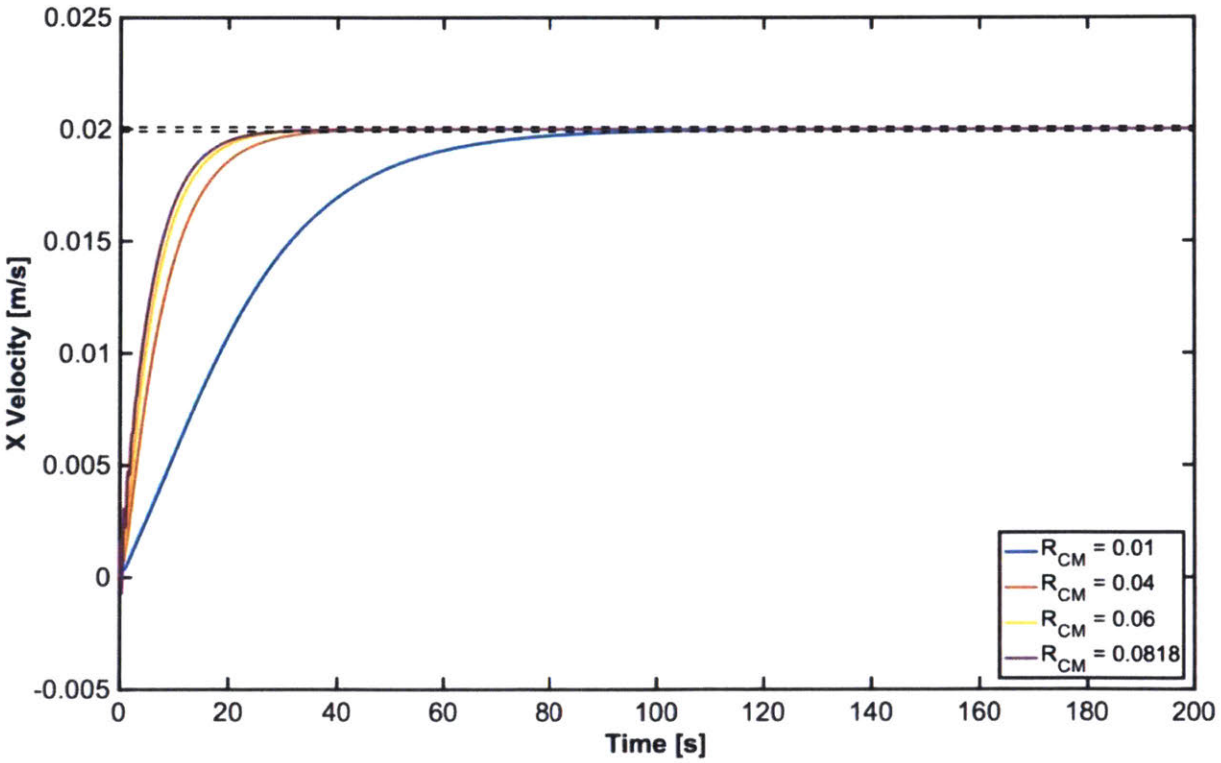

Figure 5-10: X-velocity step response for select center of mass positions. Black lines indicate settling time window of $\pm 10^{-4}$ m/s

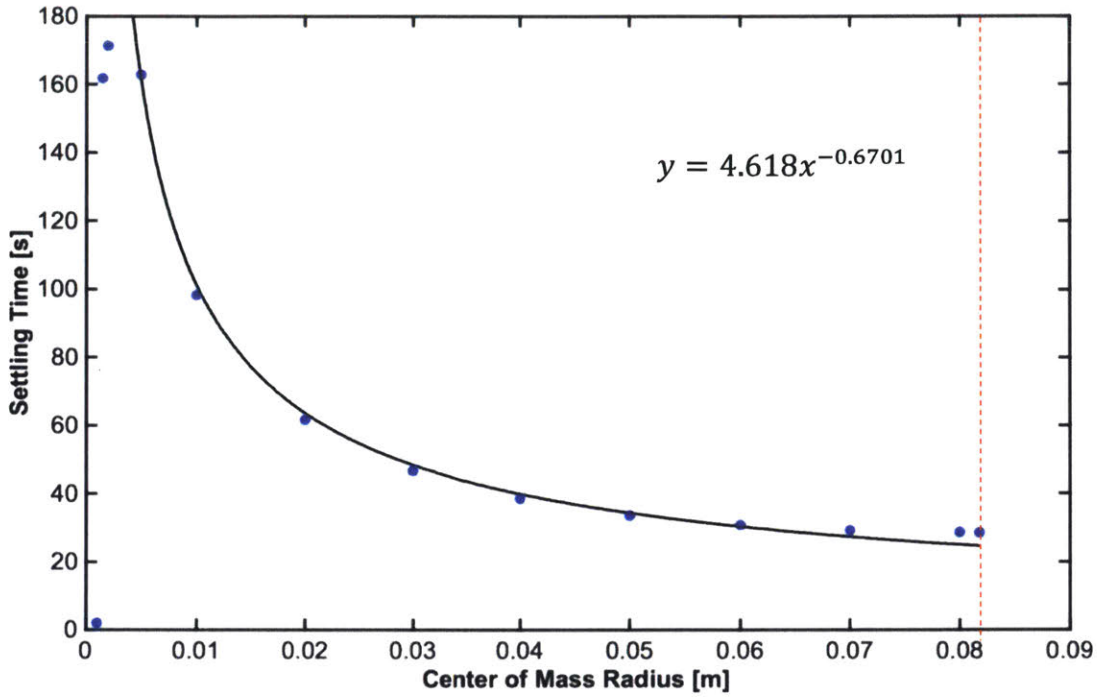

Figure 5-11: Settling time as a function of center of mass position. Black curve represents an exponential fit. Red line indicates critical center of mass radius above which system goes unstable.

Running the simulation revealed that increasing R_{CM} above a critical value of 0.0818 m, equal to the center of the wheel location $R_w - r$, resulted in the system going unstable. Increasing R_{CM} prior to this point resulted in an exponential decay in settling time, following the function $t_s = 4.618{R_{CM}}^{-0.6701}$.

5.5: Discussion

Simulating the system for different sphere masses and center of mass positions shows that increasing the sphere's mass increases settling time at a roughly linear rate, and increasing the distance from the center of mass to the sphere's center causes an exponential decrease in settling time up until a critical point for both position and velocity control. This behavior seems to inversely correlate to value of K_3 in the gain matrix. K_3 decreases linearly with sphere mass and increases logarithmically with center of mass distance, suggesting that the change in controller gain values

is a contributing factor to the system's behavior. The overall system dynamics likely contribute to the data as well. Increasing sphere mass increases the system's overall inertia, making it slower to accelerate from rest and explaining the increased damping at higher mass values.

5.6: Design Recommendations

Based on these trends, a design that minimizes sphere mass and has a large center of mass radius while staying below 0.08 m is recommended for minimizing position and velocity settling time. A low sphere mass can be accomplished by using lightweight material like plastic with a small thickness. Using a 12-inch diameter polyethylene plastic ball with a thickness of 0.125 inches would reduce the sphere weight from 1.18 kg to 0.76 kg, so this could be an appropriate choice for the outer sphere given that the structure must also be stiff enough to not deform significantly under the weight of the robot. Since the benefits in settling time for increasing center of mass position taper off as it nears the instability point of 0.0818 m, a center of mass radius of around 0.06 m could make a reasonable choice. This puts the center of mass a safe distance away from the instability point while only causing an increase of two seconds in settling time compared to the minimum value. This lower center of mass could be achieved by lowering the location of the motors in the original design, or by adding weight to the bottom of the robot.

Section 6: Summary and Conclusion

6.1: Conclusions

This thesis found that increasing the mass of the sphere resulted in a linear increase in settling time for both sphere position and velocity. Increasing the center of mass radius resulted in an exponential decay in settling time for both sphere position and velocity up until a critical point of $R_w - r$ was reached, after which the system became unstable. Based on this information, minimizing the mass of the sphere and maximizing the distance of the center of mass from the center of the sphere up until the critical point are the best design decisions for converging to the desired setpoint the fastest for the controller used in this study. Based on realistic sphere materials and geometries, a sphere mass of 0.76 kg was selected. A center of mass position of 0.06 m from the center of the sphere was chosen, balancing the diminishing returns in increasing settling time further with the danger of designing too close to the instability point.

6.2: Limitations and Next Steps

This study has a number of limitations. First, the linearized system dynamics used in LQR are only valid for small angles, meaning that applying large angle disturbances would result in the system going unstable. The simulation also did not consider friction, transmission backlash, system compliance, sampling rate, control rate, or other real-world phenomenon that would impact the actual system's performance. The values in this study for sphere mass and center of mass position were chosen by hand and changed independently rather than evaluated using an optimization script. Potential next steps for this study would be to create a program that selects the sphere mass and center of mass combination that provides optimal performance and to test the controllers performance on the physical robot to verify the model's accuracy.

Appendix A: MATLAB Code

This code calculates the values of the state variables x, theta, dx, and dtheta. It takes the system parameters as inputs.

```
%clc;
%clear all
close all;

global m_s m_r R_s R_w r_w r_cm I_r I_s k g T_r q_T_ref

m_s = 5; %mass of sphere [kg]
m_r = 2; %mass of robot body [kg]
R_s = 0.146; %radius of sphere [m]
R_w = 0.112; %contact radius of robot wheel[m]
r_w = 0.0301625; %wheel radius [m]
r_cm = 0.04; %radius from center of circle to robot center of mass [m]
I_r = 0.0135518; %wheel moment of inertia [kg*m^2]
I_s = 2/3*m_s*R_s^2; %sphere moment of inertia [kg*m^2]
g = 9.81; %[m/s^2]
k = [ -2.6171    0.6687  -44.5023    0.9631];
%k = [0 0 0 0];
q_T_ref = [0;0;0.02;0]; %controlled variable (x_s; theta_g; dx_s; dtheta_g)

T_r = [R_s 0 0 0;   %Transformation matrix from states to [X_s, theta_g,
dX_s, dtheta_g
      1  1 0 0;    %Where X_s = x position of sphere and theta_g = global
position of robot body
      0  0 R_s 0;
      0  0 1  1];

q_T_ref_transform = T_r*q_T_ref; %controlled variable (x_s; theta_g; dx_s;
dtheta_g)
q0 = [0; 10 ; 0; 0]*pi/180;
Tsim = 200;


[T, Q] = ode45(@SIM,[0 Tsim],q0);
THETA_S = Q(:,1);
THETA_B = Q(:,2);
DTHETA_S = Q(:,3);
DTHETA_B = Q(:,4);

Tau  = -k*(Q'-q_T_ref_transform);

X_S = THETA_S*R_s;
DX_S = DTHETA_S*R_s;
Y_S = R_s*ones(size(X_S));

X_B = X_S + r_cm*sin(THETA_S +THETA_B);
Y_B = R_s - r_cm*cos(THETA_B +THETA_S);
```

```
dT = 0.04;
T_d = 0:dT:Tsim;
THETA_S_d = interp1(T,THETA_S,T_d);
THETA_B_d = interp1(T,THETA_B,T_d);
tic

for i = 1:length(T_d)

    animate(THETA_S_d(i),THETA_B_d(i),1)

end
toc
```

LQR

This code calculates the gain matrix values using LQR. It takes the system parameters and the cost function components Q and R as inputs.

```
global m_s m_r R_s R_w r_w r_cm I_r I_s g

A = [0, 0, 1, 0;
     0, 0, 0, 1;
     (R_s*g*m_r^2*r_cm^2)/(I_s*m_r*R_s^2*r_cm^2 + I_r*I_s*R_s^2 +
m_r*m_s*r_cm^2 + I_r*m_r + I_r*m_s),
(R_s*g*m_r^2*r_cm^2)/(I_s*m_r*R_s^2*r_cm^2 + I_r*I_s*R_s^2 + m_r*m_s*r_cm^2 +
I_r*m_r + I_r*m_s), 0, 0;
    -(I_s*g*r_w*R_s^2*m_r*r_cm + g*r_w*R_s*m_r^2*r_cm^2 + g*r_w*m_r^2*r_cm +
g*m_s*r_w*m_r*r_cm)/(r_w*(I_s*m_r*R_s^2*r_cm^2 + I_r*I_s*R_s^2 +
m_r*m_s*r_cm^2 + I_r*m_r + I_r*m_s)), -(I_s*g*r_w*R_s^2*m_r*r_cm +
g*r_w*R_s*m_r^2*r_cm^2 + g*r_w*m_r^2*r_cm +
g*m_s*r_w*m_r*r_cm)/(r_w*(I_s*m_r*R_s^2*r_cm^2 + I_r*I_s*R_s^2 +
m_r*m_s*r_cm^2 + I_r*m_r + I_r*m_s)), 0, 0];

B = [0;
     0;
    -(R_s*(I_r*R_s^2 + I_r*R_s*R_w + R_s^2*m_r*r_cm^2 - I_r*R_s*r_w +
R_w*m_r*r_cm - m_r*r_cm*r_w + R_s*R_w*m_r*r_cm^2 -
R_s*m_r*r_cm^2*r_w))/(r_w*(I_s*m_r*R_s^2*r_cm^2 + I_r*I_s*R_s^2 +
m_r*m_s*r_cm^2 + I_r*m_r + I_r*m_s));
    (R_w*m_r + R_w*m_s - m_r*r_w - m_s*r_w + I_r*R_s^3 + R_s^3*m_r*r_cm^2 +
I_r*R_s^2*R_w + I_s*R_s^2*R_w - I_r*R_s^2*r_w - I_s*R_s^2*r_w +
R_s^2*m_r*r_cm - 2*R_s*m_r*r_cm*r_w + R_s^2*R_w*m_r*r_cm^2 -
R_s^2*m_r*r_cm^2*r_w + 2*R_s*R_w*m_r*r_cm)/(r_w*(I_s*m_r*R_s^2*r_cm^2 +
I_r*I_s*R_s^2 + m_r*m_s*r_cm^2 + I_r*m_r + I_r*m_s))];


T_r = [R_s 0 0 0;    %Transformation matrix from states to [X_s, theta_g,
dX_s, dtheta_g
       1  1 0 0;     %Where X_s = x position of sphere and theta_g = global
position of robot body
       0  0 R_s 0;
       0  0 1  1];
```

```
A_2 = T_r*A/T_r;

B_2 = T_r*B;



Q = diag([1/R_s 1 1/R_s 1]); %Cost function for states (higher value means I
prioritize speed of response)
R = 1;

[K,S,E] = lqr(A_2,B_2,Q,R);
```

Animation

This code creates an animation of the spherical robot moving with time. It takes the system parameters and the state and time vectors as inputs.

```
function moving_graph = animate(THETA_S, THETA_B, n)

global m_s m_r R_s R_w r_w r_cm I_r I_s tau

X_S = THETA_S*R_s;
Y_S = R_s*ones(size(X_S));

X_B = X_S - r_cm*sin(THETA_S +THETA_B); %NOTE: Changed all of the X equations
after this to X_S-rcm*sin(theta_s+theta_b)
Y_B = R_s - r_cm*cos(THETA_B +THETA_S);

X_Wheel = X_S -(R_w-r_w)*sin(THETA_S +THETA_B);
Y_Wheel = R_s -(R_w-r_w)*cos(THETA_S +THETA_B);

X_line = X_S -(R_w-2*r_w)*sin(THETA_S +THETA_B);
Y_line = R_s -(R_w-2*r_w)*cos(THETA_S +THETA_B);



figure(n);
plot(X_S+R_s*cos(0:0.001:2*pi),R_s+R_s*sin(0:0.001:2*pi),'LineWidth',3) %Plot
of sphere
hold on

plot([X_S,X_line],[R_s,Y_line],'k-','LineWidth',3) %Plot of line from center
of sphere to edge of wheel
plot(X_B,Y_B,'b*') %Plot robot center of mass
plot(X_Wheel+r_w*cos(0:0.001:2*pi),Y_Wheel+r_w*sin(0:0.001:2*pi),'LineWidth',
2) %Plot of wheel

axis equal
xlabel('X position [m]')
ylabel('Y position [m]')
hold off
```

```
end
```

Lagrange

This code symbolically calculates the equations of motion for the system. It also calculates the linearized A and B matrices used in LQR.

```
syms R_s R_w r_w r_cm I_s I_r m_r m_s g tau
syms theta_s theta_b dtheta_s dtheta_b ddtheta_s ddtheta_b

x_s = theta_s/R_s;
dx_s = dtheta_s/R_s;

x_r = x_s + r_cm*sin(theta_b +theta_s);
y_r = R_s - r_cm*cos(theta_b +theta_s);

dx_r = dx_s + r_cm*cos(theta_b + theta_s)*(dtheta_s + dtheta_b);
dy_r = r_cm*sin(theta_b + theta_s)*(dtheta_b + dtheta_s);

%% Kinetic Energy of Sphere

K_s = 1/2*m_s*dx_s^2 + I_s/2*dtheta_s^2;
P_s = m_s*g*R_s;

%% Kinetic Energy of Robot


K_r = m_r/2*(dx_r^2 +dy_r^2) + I_r/2*(dtheta_b + dtheta_s)^2;
P_r = m_r*g*y_r;

K = K_s + K_r;
P = P_r+P_s;
L = K-P;

dL_theta_s = simplify(diff(L,theta_s),'Steps',200);
dL_dtheta_s = simplify(diff(L,dtheta_s),'Steps',200);
ddL_dtheta_s = simplify(diff(dL_dtheta_s,dtheta_s)*ddtheta_s +
diff(dL_dtheta_s,theta_s)*dtheta_s + diff(dL_dtheta_s,theta_b)*dtheta_b +
diff(dL_dtheta_s,dtheta_b)*ddtheta_b,'Steps',200);

dL_theta_b = simplify(diff(L,theta_b),'Steps',200);
dL_dtheta_b = simplify(diff(L, dtheta_b),'Steps',200);
ddL_dtheta_b = simplify(diff(dL_dtheta_b,dtheta_b)*ddtheta_b +
diff(dL_dtheta_b,theta_b)*dtheta_b + diff(dL_dtheta_b,theta_s)*dtheta_s +
diff(dL_dtheta_b,dtheta_s)*ddtheta_s,'Steps',200);



Q_b = simplify(ddL_dtheta_b - dL_theta_b,'Steps',200);
Q_s = simplify(ddL_dtheta_s - dL_theta_s,'Steps',200);
```

```
EOM = A\b;
theta_s_EOM = simplify(EOM(1),'Steps',500);
theta_b_EOM = simplify(EOM(2),'Steps',500);

f = [dtheta_s;dtheta_b;theta_s_EOM;theta_b_EOM];
x = [theta_s; theta_b; dtheta_s; dtheta_b];

A = jacobian(f,x);
B = jacobian(f,tau);

A_linear = subs(A,[theta_b theta_s dtheta_b dtheta_s],[0 0 0 0]);
B_linear = subs(B,[theta_b theta_s dtheta_b dtheta_s],[0 0 0 0]);
```

Bibliography

[1] Crossley, V.A. (2006). "A Literature Review on the Design of Spherical Rolling Robots"

[2] Chase, R., and Pandya, A. (2012). "A Review of Active Mechanical Driving Principles of Spherical Robots." MDPI, Multidisciplinary Digital Publishing Institute, <https://www.mdpi.com/2218-6581/1/1/3/htm> (May 9, 2019).

[3] Allain, R. (2015). "The Physics of How That Star Wars BB-8 Toy Works". [online] WIRED. Available at: https://www.wired.com/2015/09/physics-star-wars-bb-8-toy-works/ [Accessed 10 May 2019].

[4] "Rotundus", 2008. [Online]. Available: http://www.rotundus.se/design.html.

[5] Kim, Y.-M., Ahn, S.-S., and Lee, Y.-J. (n.d.). "KisBot: New Spherical Robot with Arms." Kyungpook National University.

[6] Dunbar, B. (n.d.). "The Tumbleweed Rover is on a Roll." NASA, NASA, <https://www.nasa.gov/missions/earth/f_tumbleweed.html> (May 10, 2019).

[7] Orcutt, M. (2010). "Researchers Test a Next-Gen, Wheel-Free Mars Rover." Popular Mechanics, Popular Mechanics, <https://www.popularmechanics.com/space/moon-mars/a5845/tumbleweed-mars-rover/> (May 10, 2019).