# Autonomous Landing for Tumbling Robots

RBE 502 Group Project Proposal

Brian Francis Alethea Saine Naseem Shah

***Abstract*— Cats are incredibly agile creatures who are known for their ability to land on their feet when falling. Their natural ability to reorient while falling is a valuable technique to replicate when developing robots that have to land from variable situations. This project is aimed at creating a model that replicates the real-world situation of a cat falling and landing upright. A 3D object with 4 legs was created to replicate the body of a cat that will be used for the simulation. The object will be simulated using controllers in a closed loop system modeled in Matlab.**

***Index Terms*—controller, Matlab, simulation**

## I. Introduction

The field of biomimietics refers to the study and recreation of preexisting natural phenomena whether for robotics, material science, or psychology. [1] Nature provides a wealth of inspiration for developing robotic systems. Evolution has had years to develop mechanisms and processes that are often much more advanced than humans would be able to develop independently. Adapting that knowledge to be practically reapplied is an effective way to create new technologies and expand the realm of possibility for human technology. Many of these changes are widely known and highly intuitive, airplanes are shaped like diving birds to reduce drag, hook and loop tape is shaped like the seed of a bur, and even the fishing net is a human facsimile of a spider's web. [2]

Robotics is a field that draws heavy inspiration from nature and in particular animals and animal behavior. Selective pressures from the environment have tuned animals to be both efficient and highly flexible in ability. Companies such as Boston Dynamics have developed animal duplicates that are simplified forms of real life animals like dogs or even humans. [3] These robots are designed to be capable of all of the same tasks and motions as the original animal, but with the added benefit of increased mechanical strength and computerized control. Robotic biomimicry is not exclusively limited to recreating animals in a one-to-one fashion. Another more limited approach is, where a conventional robot looks to adopt a useful feature or skill from a natural phenomenon. For example, the use of stigmergy or environmental communication is a technique that ants utilize that is seeing a rise of popularity in swarm robotics. [4] Our project falls into this second category, as our goal is not to create a cat robot, but to replicate its behavior. We hope to develop a robotic platform that is capable of self righting, and nature offers an infamous solution in the form of the cat.

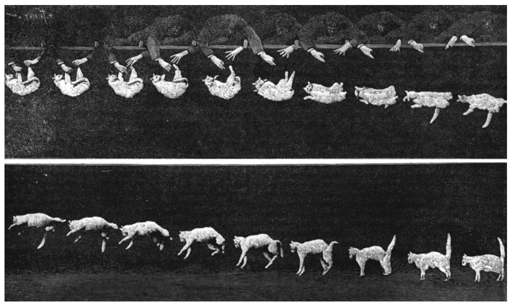

Fig. 1: Cat demonstrating self righting reflex [5]

## II. Background

Cats are notorious for their ability to land upright when dropped from even short distances. The reason that cats in particular are such a useful model for autonomous landing is because they are able to generate any amount of rotation without the aid of any external forces like wings or fins. They utilize their flexible spine and a sense of gravitational inertia to rotate their bodies around even when falling through the air. [6] We set out to develop a robot that utilized the same physical principles as a falling cat in order to self-right while tumbling without the aid of thrusters or other cumbersome, non-conservative mechanisms.

The application of such a system could be range from correcting the angle of landing rovers on planetary missions to helping balance a more complex robotic system during a jump or fall. Since it is important to minimize both the requisite size and complexity when designing for robots as both characteristics incur a high monetary and difficulty cost. By recreating the internal ability to self right using only internal torque generation we will create a simple but effective methodology for controlling orientation.

The core principle behind the physics of the system comes from the conservation of angular momentum. We are designing a conservative system which means there are exclusively conservative forces acting on the system as far as the rotation is concerned. These forces do not change the overall momentum of the system, which will therefore be constant throughout the whole process. This means that if one component is attempting to spin clockwise the system will spin; clockwise so that there is no new momentum generated, when one part of a system generates a rotation in one direction the system as a whole will generate a counter-rotation to maintain the overall system momentum. Cats utilize their tail and flexibility to generate opposing rotations in the front and back half of their body in order to both start and stop spinning. Our robot will be utilizing a momentum or reaction wheel to accomplish the same thing. These wheels work by placing relatively massive wheels in the system attached to a motor and generating rotation in the opposite direction that the system is intended to spin. By placing wheels along both the z and x axis of the robot we will be capable of generating and stopping rotation motion along both those axes.

## *A. Related Works*

The falling cat problem itself is a highly studied phenomenon with numerous publications on robots that are capable of self righting. [7] These papers all attempt to replicate the agility of a cat in a variety of ways primarily focusing on generating reasonable platforms that can generate the internal torque necessary to rotate. The proposed solutions range from robotic tails, to flexible limbs and spines, to the flywheel approach that we elected to use. There are also trajectory planners for generalized models that offer a way to determine what sort of path the robot should take in order to reach its desired final pose over a fixed time frame. [8] This paper seeks to develop and propose an additional controller and model for landing a tumbling legged robot with a primary focus on the dynamics and control scheme at hand.

## *B. Problem Statement*

In this project, we aim to develop a controller, model, and robotic solution for aerial self-righting and tumble prevention. This robot will be able to be land upright after free-falling with a given rotational velocity.The primary challenges and contributions we hope to resolve are as follows:

- Develop a controller that can take in dynamic motion and return a counter rotation plan
- Create a robot that is capable of orienting itself without external torques
- Utilize the momentum properties of our system to counteract uncontrolled tumble

## III. METHODOLOGY

In this section we discuss the method for deriving the model. Starting with the physical model, we obtained the dynamical model using the moments of inertia, masses, and other known parameters. We then determined a strategy for the control system and implemented the system in simulation. Matlab was used to model the dynamics and controller, and simulate the system.

## *A. 3D Robot Model*

For this project, the team wanted to incorporate a 3D model of a cat to use in the simulations. A simplified model of a cat was created in AutoCAD.The object is compromised of a body, 4 legs, two reaction wheels, and an accelerometer. While the reaction wheels and accelerometer weren't physically built on the 3D model, they are incorporated in the simulation. A prototype of the robot is shown in in Fig. 2.

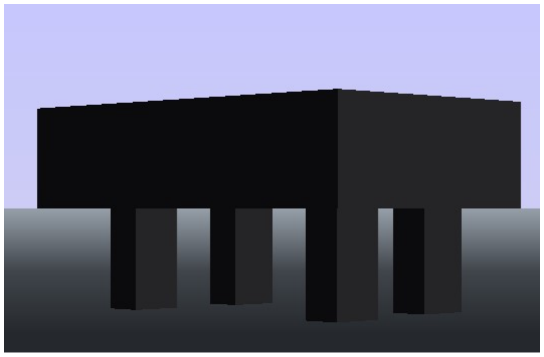
Fig. 2: Autonomous Lander Model

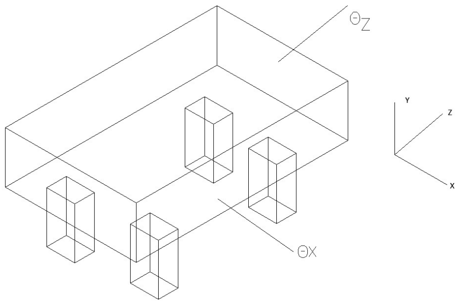

Fig. 3: Autonomous Lander Orientation

For orientation references in deriving the model equations, refer to Figs. 3, 4 and 5.

## *B. Moments of Inertia*

$I_{BZ}$: Moment of inertial of the body about the z axis

$I_{BX}$: Moment of inertial of the body about the x axis

$I_{LX}$: Moment of inertial of the legs about the x axis

$I_{LZ}$: Moment of inertial of the legs about the z axis

$I_{FX}$: Moment of inertial of the feet about the x axis

$I_{FZ}$: Moment of inertial of the feet about the z axis

## *C. Masses*

$m_B$: Mass of the robotic lander body, including the two reaction wheels

$m_L$: Mass of each leg

$m_F$: Mass of each foot (we will baseline foam for the feet and therefore assume negligible mass)

$m_B$: Mass of the robotic lander body, including the two reaction wheels

$m_L$: Mass of each leg

$m_F$: Mass of each foot (we will baseline foam for the feet and therefore assume negligible mass)

$m_T$: Total mass of lander: $m_B + 4m_L + 4m_F$

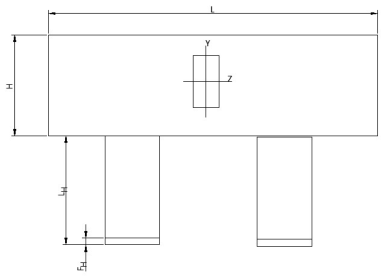

Fig. 4: Side View

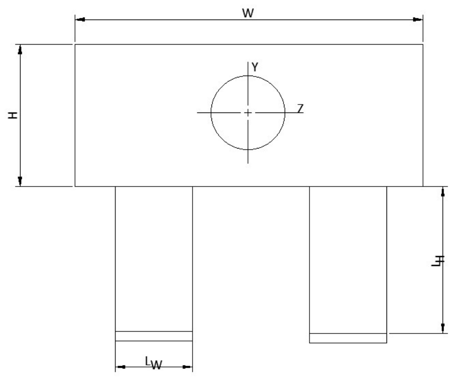

Fig. 5: Front View

$m_{RW_X}$: Mass of reaction wheel normal to the x axis

$m_{RW_Z}$: Mass of reaction wheel normal to the z axis

$m_T$: Total mass of lander: $m_B + 4m_L + 4m_F$

$m_{RW_X}$: Mass of reaction wheel normal to the x axis

$m_{RW_Z}$: Mass of reaction wheel normal to the z axis

## *D. Equation of Motion*

Moments of inertia about the z axis

$$I_{BZ} = \tfrac{1}{12}m_B(H^2 + W^2)$$
$$I_{LZ} = 4 \times [\tfrac{1}{12}m_L(L_H^2 + L_W^2) + m_L r_{LZ}^2]$$
$$I_{FZ} = 4 \times [\tfrac{1}{12}m_F(F_H^2 + F_W^2) + m_F r_{FZ}^2]$$

Moments of inertia about the x axis

$$I_{BX} = \tfrac{1}{12}m_B(H^2 + L^2)$$
$$I_{LX} = 4 \times [\tfrac{1}{12}m_L(L_H^2 + L_L^2) + m_L r_{LX}^2]$$
$$I_{FX} = 4 \times [\tfrac{1}{12}m_F(F_H^2 + F_L^2) + m_F r_{FX}^2]$$

$I_{RW_X}$ depends on the chosen reaction wheel

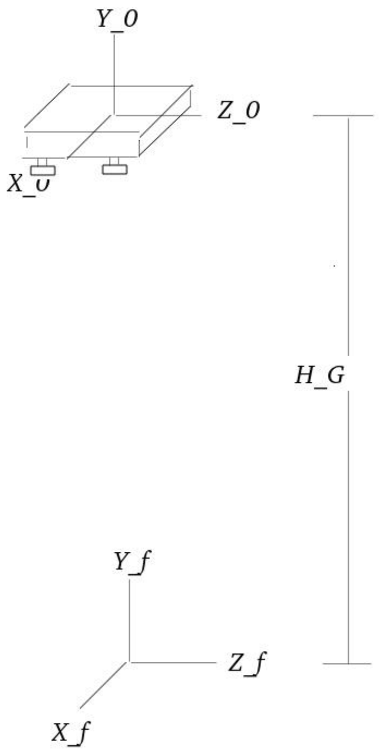

Fig. 6: Model Dropped from an initial height

Moments of inertia about the y axis

$$I_{BY} = \tfrac{1}{12}m_B(L^2 + W^2)$$
$$I_{LY} = 4 \times [\tfrac{1}{12}m_L(L_L^2 + L_W^2) + m_L r_{LY}^2];\ m_F \approx 0, I_{FX} \approx 0$$
$$I_{FY} = 4 \times [\tfrac{1}{12}m_F(F_L^2 + F_W^2) + m_F r_{FY}^2];\ m_F \approx 0, I_{FY} \approx 0$$

The internal kinetic energy of the system with no external force from the reaction wheels is given by:

$$KE = \frac{1}{2}I\omega_x^2 + \frac{1}{2}I\omega_y^2 + \frac{1}{2}I\omega_z^2 + \frac{1}{2}m_T v^2$$

where

$$\omega_x = \tfrac{d\theta_x}{dt} = \dot{\theta}_x;\ \omega_y = \dot{\theta}_y;\ \omega_z = \dot{\theta}_z;$$
$$v = \dot{y}^2;$$

The potential energy of the system is given by:

$$PE = mgh = mg(H_g - y)$$

where the inertial reference frame is the final position of the lander. h, the height above the ground equals the initial height in the positive y direction $H_g$minus the change in position of the lander, y. Refer to Fig. 6

The Lagrange equation of motion is:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = \tilde{F}_{q_i}$$

where $\tilde{F}_{q_i}$ are the non conservative forces acting on the lander, including the control torques $\tau$ for the actuators and drag (which we may ignore).

The Lagrangian can be written as:

$$\mathcal{L} = KE - PE$$
$$\mathcal{L} = \tfrac{1}{2}I_{BZ}\dot{\theta}_Z^2 \;+\; \tfrac{1}{2}I_{LZ}\dot{\theta}_Z^2 + \;\tfrac{1}{2}I_{BX}\dot{\theta}_X^2 \;+\; \tfrac{1}{2}I_{LX}\dot{\theta}_X^2$$
$$+\;\tfrac{1}{2}I_{BY}\dot{\theta}_Y^2 \;+\; \tfrac{1}{2}I_{LY}\dot{\theta}_Y^2 + \;\tfrac{1}{2}m\dot{y}^2 \;-mg(H_G - y)$$

The configuration of the system that we're interested in is given by $\theta_x$, the angle about the x-axis, $\theta_y$ the angle about the y-axis, $\theta_z$ the angle about the z-axis, and $y$the height above the ground. The general coordinates are given by:

$$q = [\theta_x \;\theta_y \;\theta_z \;y]$$

The Lagrange equation of motion using : $\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = \tilde{F}_{q_i}$ and making substitutions is as follows:

$$[I_{BZ}+I_{LZ}]\ddot{\theta}_z+[I_{BX}+I_{LX}]\ddot{\theta}_x+[I_{BY}+I_{LY}]\ddot{\theta}_y+m\ddot{y}^2-mg = \tau$$

which can be written as:

$$\begin{pmatrix} I_{BX}+I_{LX} & 0 & 1 & 0 \\ 0 & I_{BY}+I_{LY} & 0 & 0 \\ 0 & 0 & I_{BZ}+I_{LZ} & 0 \\ 0 & 0 & 0 & m \end{pmatrix} \begin{pmatrix} \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \\ \ddot{y} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ -mg \end{pmatrix} = \begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \tau_{dy} \end{pmatrix}$$

From this, we will derive a state space model.

Since we do not care about yaw, and are ignoring freefall control, the equations for control reduce to:

$$(I_{BX}+I_{LX})\ddot{\theta}_x = \tau_x$$
$$(I_{BZ}+I_{LZ})\ddot{\theta}_z = \tau_z$$

In cannonical form, this becomes:

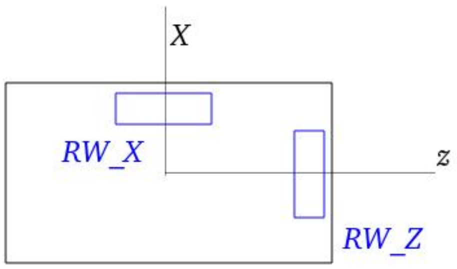

Fig. 7: Reaction Wheels - Orientation

$$\begin{pmatrix} I_{BX}+I_{LX} & 0 \\ 0 & I_{BZ}+I_{LZ} \end{pmatrix} \begin{pmatrix} \ddot{\theta}_x \\ \ddot{\theta}_z \end{pmatrix} = \begin{pmatrix} \tau_x \\ \tau_z \end{pmatrix}$$

These can be written in state space format where $f_1 = \theta_x \;f_2 = \theta_z \;f_3 = \dot{\theta}_x \;f_4 = \dot{\theta}_z$ as:

$$\begin{pmatrix} \dot{f}_1 \\ \dot{f}_2 \\ \dot{f}_3 \\ \dot{f}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{\tau_x}{I_{BX}+I_{LX}} \\ \frac{\tau_z}{I_{BZ}+I_{LZ}} \end{pmatrix}$$

### E. *Control Strategy*

Initially, the system may be rotating, and we will first stabilize it by setting the desired angular velocity to zero and ignoring the desired angle. Once stabilization has been achieved, the objective will be to achieve the desired angle.

The desired angle will be achieved by setting a desired velocity for the system and the time for which the velocity should be applied.

A system velocity can be achieved by setting the reaction wheel velocity.

The closed loop control system is shown in Fig. 9.

The reaction wheel's inertia and angular velocity are related to the system's inertia and angular velocity as follows:

$$I_{RW_X}\omega_{RW_X} + I_{S_X}\omega_{S_X} + I_{S_X} = h_{total_X}$$
$$I_{RW_Z}\omega_{RW_Z} + I_{S_Z}\omega_{S_Z} + I_{S_Z} = h_{total_Z}$$

where $h_{total_X}$ for example is the total angular momentum of the system about the X axis, $\omega_{S_X}$ is the vehicle's angular velocity about the x-axis, and $I_{RW_X}$ is the reaction wheel's moment of inertia about the x axis. Refer to Figs. 7 and 8

For the vehicle to not rotate, $h_{total_X}$ and $h_{total_Z}$ must be 0. Therefore, initially:

$$\omega_{RW_X} = -\omega_{S_X}\frac{I_{S_X}+I_{RW_X}}{I_{RW_X}}$$

and

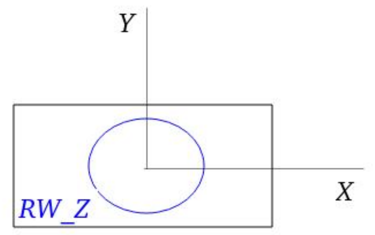

Fig. 8: Front View of z-axis Reaction Wheel

$$\Delta\omega_{RW_X} = -\Delta\omega_{S_X}\frac{I_{S_X} + I_{RW_X}}{I_{RW_X}}$$

When the measured system angular velocity about the x axis, $\omega_{S_X}$, is close enough to zero and the system is considered to no longer be rotating in either axis, the angle $\theta_S$ will be controlled.

This will be achieved by setting a specified duration and calculating the constant $\omega_S$ necessary to achieve the rotation. The desired constant angular velocity of the reaction wheel, $\omega_{RW}$ necessary to achieve this rotation, will then be calculated and used in the velocity controller.

The system controller consists of several blocks. The velocity controller outputs the torques to the system model. As input, it takes a desired system velocity, and converts it to a desired reaction wheel velocity. It will use a PD control law to update the torques.

The Angle controller is more of a decision making block. As input, it takes the desired reference angle for the upright lander position, and the current angle for the system. It also will need knowledge of the angular velocity and acceleration rates. The angle controller has two modes of operation. If the system is spinning initially, it sets the output system angular velocity to zero. Once the system has been deemed stable, the mode switches to angle control, in which case, it calculates a desired constant system angular velocity and time to achieve the angle of rotation, and passes that data to the velocity controller.

The velocity controller will receive a desired system velocity and potentially a time from the previous block. It will calculate the desired reaction wheel velocity and attempt to reach the RW velocity using a PD control law.

The sensor block will use a 9-DOF IMU to measure angular velocity and acceleration, which will be used by the Angle Controller for decision making on the mode of operation and desired velocity.

## IV. Simulation

### A. Robot Simulation

In order to simulate the Autonomous Lander, the team used two main softwares: Simulink and AutoCAD. AutoCAD was used to design a physical model in stl format that could be used in a 3D Matlab environment. The model designed is shown in an earlier section above. Simulink is MATLAB's control design package and included in it is Simscape Multibody which allows you to simulate mechanical systems. The Autonomous Lander is modeled using a closed loop system shown in Fig. 9 below.

From the model above, there are 2 physical components: the landing pad and the autonomous lander. The landing pad provides a reference point by which the autonomous lander should stop rotating and correct itself to an upright position. Initially the design included physically modeled reaction wheels, but these caused an unaccounted for precession that required them to be replaced with direct torque. In order to test the controller, the lander is dropped from a set height of 70 meters and a specified displacement angle.

As the "cat" falls the Cartesian acceleration values are recorded by a simulated accelerometer and passed into an orientation calculator. Since real life gyro sensors can be inaccurate, sensitive to high velocity rotations, or prohibitively expensive we elected to calculate our orientation in this method to more closely model a realistic robotic scenario. The calculator passes roll and pitch values based on the relative magnitudes of acceleration in each local Cartesian direction. Since gravity applies a constant acceleration in the global -Z, if the acceleration is being read locally to have components in the X or Y direction we know that the lander has been rotated. The simulated IMU is also used to measure angular velocities values as well to be included in the PD control block.

The values of pitch and roll become the state errors since the desired orientation is to have no rotation in either axis. Velocity error is also the measured velocity as the desired state also has no angular rotation.

## V. Results

### A. Simulation Setup

The simulation was tested using Matlab Simulink using the model mentioned earlier. When the "Run" button is clicked, the simulation occurs and a "Mechanics Explorers" tab opens in Matlab. In this tab, you can view the simulation of the model falling, rotating, and landing upright.

For our experiment we have a few assumptions: only the X and Z axis are being controlled, the time frame for the simulation is 10 seconds, the model is starting from rest at an angle specified in the world coordinate system, and the simulation is successful when the model situates itself upright

Using these assumptions, we will analyze the graphs and pictures produced from the simulation in the following section.

In addition to Simscape Multibody, the system was also prototyped and simulated in Matlab scripts using the ODE45 solver, with similar results. The gains were tuned manually for both sets of simulation to remove spinning or tumbling. The resulting gains were different, likely as a result of the model differences including discrete IMU measurements. The resulting angular positions, velocities and control inputs were therefore different as well. The script result was smoother but the Simulink result is likely to be more realistic.

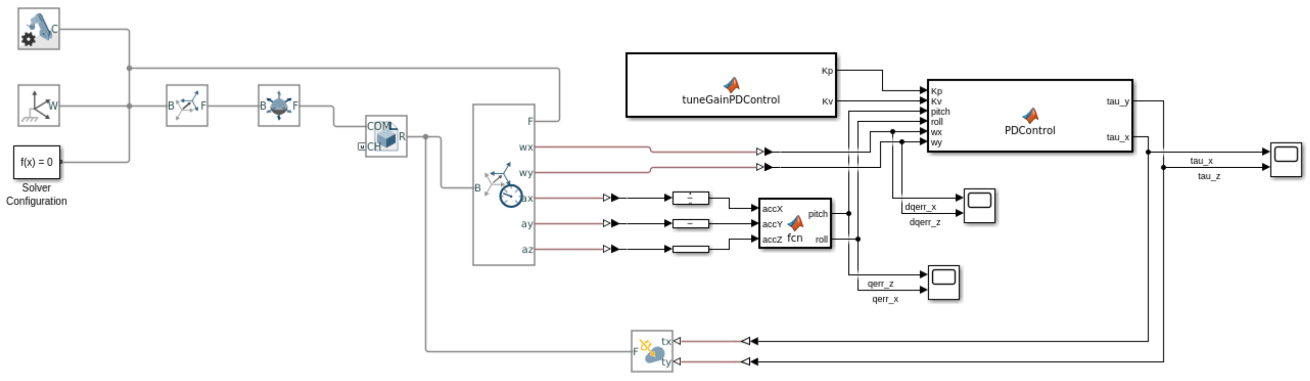

Fig. 9: System Controller Model.

### B. Simulation Results

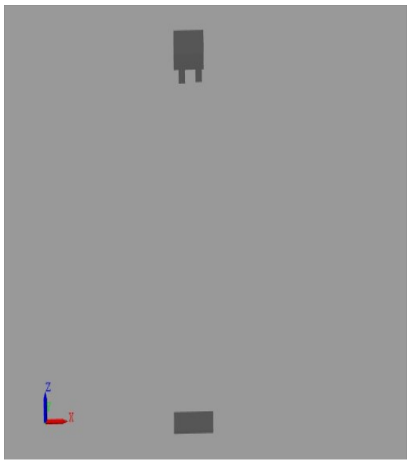

Fig. 10: Autonomous Lander at ”Peak Height”

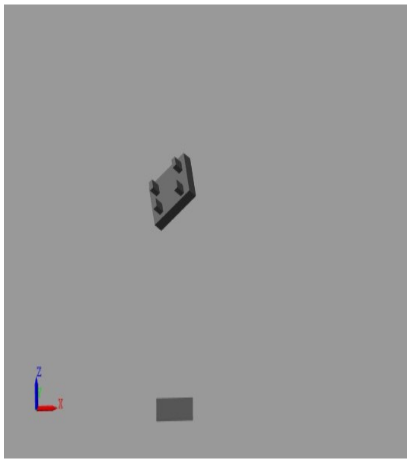

Fig. 11: Autonomous Lander at mid-fall

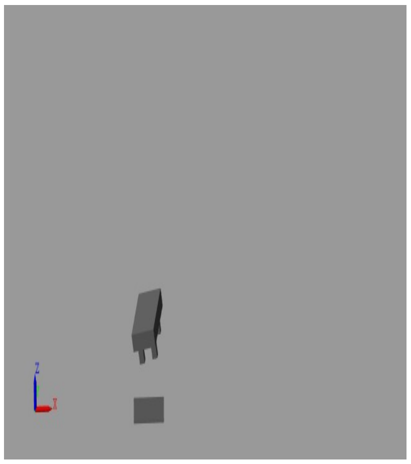

Fig. 12: Autonomous Lander self correcting before landing

The figures show screen captures and performance metrics from running the simulation. In Fig. 10 it shows that the autonomous lander is X m above the landing pad at an angle. Fig. 11 shows the lander in mid rotation during its descent in another simulation. Fig. 12 shows the lander positioning itself to be upright when it lands on the landing pad.

The errors in the angular position converge to zero in less than ten seconds. This is shown in Fig. 13 for a displacement of 15 degrees in both the x and y axis. The angular velocity error also converges to zero as shown in Fig. 14. The torque starts off around 10 N-m but reaches zero as the system reaches its desired position and velocity as seen in Fig. 15. The designed controller is capable of resolving and stabilizing not only small displacement but also an inversion of the lander. As shown in Fig. 16 the lander can converge a large displacement in the same amount of time.

The gain was tuned for the maximum displacement of 180 degrees about the x axis with no angular displacement about the z axis, and then separately for a maximum displacement

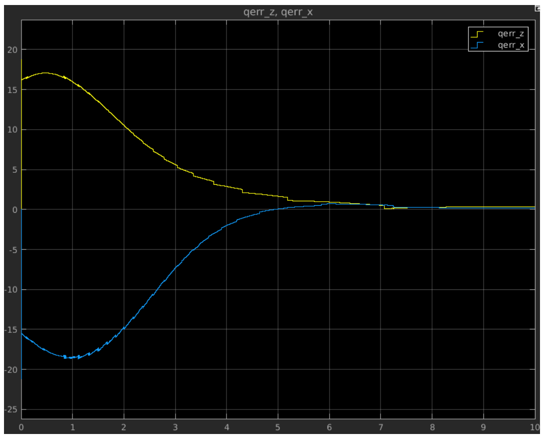

Fig. 13: Angular position error over time

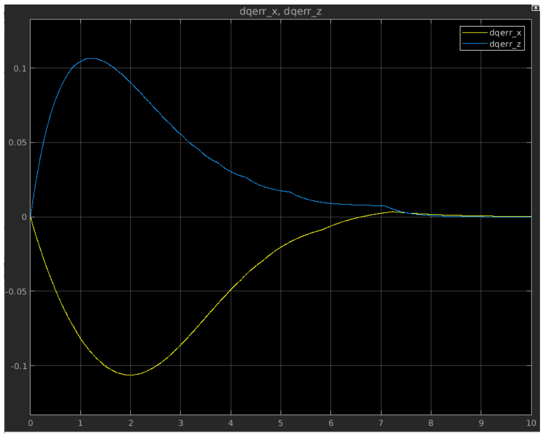

Fig. 14: Angular velocity error over time

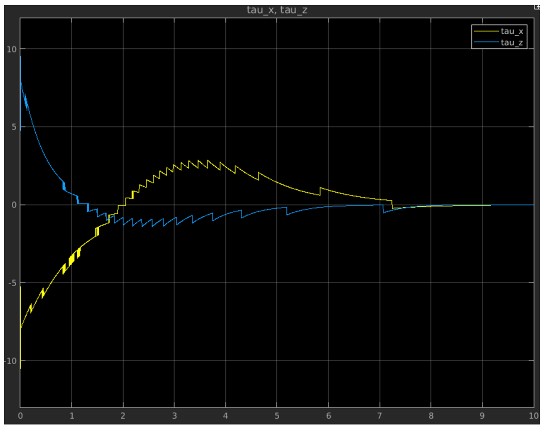

Fig. 15: Torque over time

of 180 degrees about the z axis with no angular displacement about the x axis. This resulted in a different velocity error gain ($K_D$ or $K_v$)for each scenario. However, even though the gains were used in the same matrix, large displacements about both x and y axes simultaneously resulted in discontinuous angular positions over time along with steady state errors. It is possible that a PID controller may have solved this issue but there was

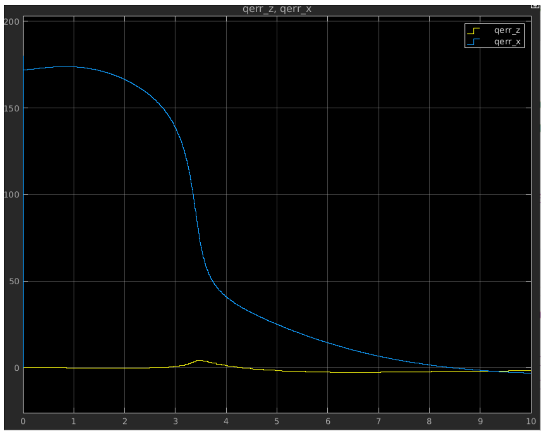

Fig. 16: Torque over time

not enough time to investigate this.

## VI. Conclusion and Future Work

We successfully demonstrated controlling the angular orientation and velocity of the system. We did not however implement a reaction wheel in Simulink. Our focus in this effort was to control the position and angular velocity of the system. Adding a reaction wheel would be an area for future work.

For this, the torque can be scaled by the reaction wheel's moment of inertia (with drag not taken into consideration) to produce an angular velocity for the reaction wheel. The system velocity can then be derived from the reaction wheel velocity and system inertia.

Another area for future work would be finding a controller that results in stable behavior for an initial configuration that has simultaneous large angular displacements about both x and z axes in Simscape. This was not an issue for the script based simulation.

Filtering the control input could also be a future endeavor.

Future work can also include adding an initial angular velocity offset.

## References

[1] J. F. V. Vincent, "Biomimetics–a review," *Engineering in Medicine*, vol. 223, 2009.

[2] L. T. B. H. Lazaara Ilieva, Isabella Ursano and H. Dahy, "Biomimicry as a sustainable design methodology—introducing the 'biomimicry for sustainability' framework," *Biomimetics*, vol. 37, no. 7, 2022.

[3] B. Dynamics. Research. [Online]. Available: https://www.bostondynamics.com/research

[4] B. F. Peter Korošec, Jurij Šilcac, "The differential ant-stigmergy algorithm," *Information Sciences*, vol. 192, pp. 82–97, 2012.

[5] Étienne Jules Marey, "Falling cat," 1894, [Online; accessed April 27, 2013]. [Online]. Available: https://en.wikipedia.org/wiki/Cat$_righting_reflex/media/File : Falling_cat_1894.jpg$

[6] H. D. Nguyen. How does a cat always land on its feet ? [Online]. Available: https://web.archive.org/web/20010410235503/http://helix.gatech.edu/Classes/ME3760/1998Q

[7] R. J. F. D. E. K. Thomas Libby, Aaron M. Johnson; Evan Chang-Siu, "Comparative design, scaling, and control of appendages for inertial reorientation," *Transactions on Robotics*, vol. 32, no. 6, pp. 1380 – 1398, 2016.

[8] P. M. W. Vince Kurtz, He Li and H. Lin, "Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics," 2022.