Lab 1 Report

Launcher Device

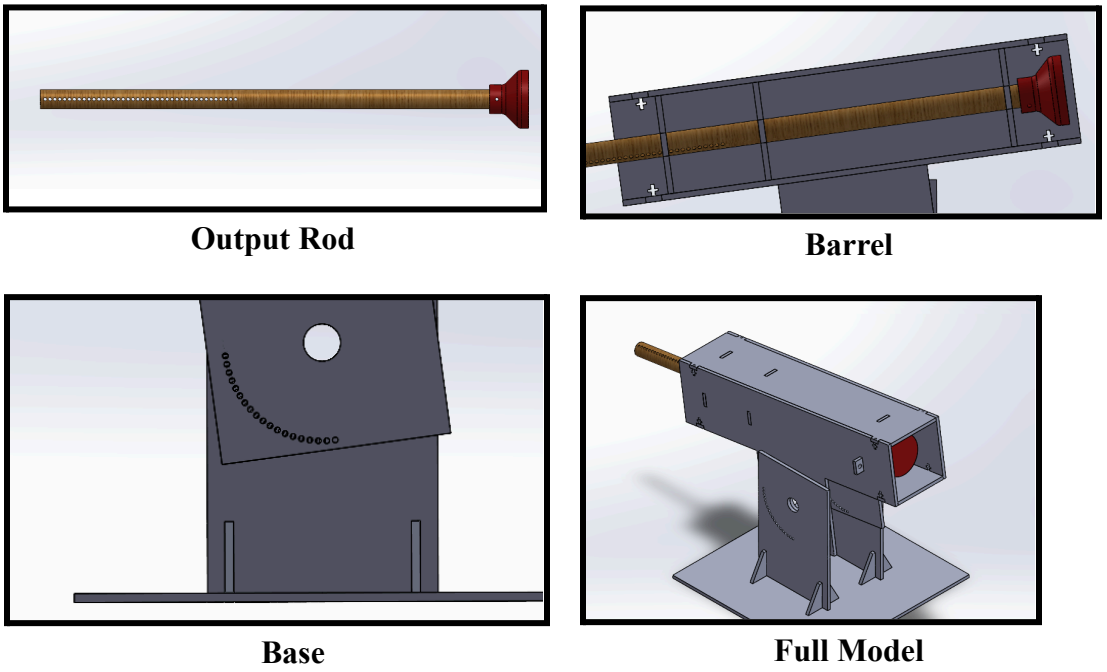


Figure 1: *SolidWorks CAD Model of the catapult. This model includes the launcher barrel, as well as the rod. This model does not include the bungee cords, which were attached to the end of the launcher barrel and the rod during physical production.*

Our launcher (**Figure 1**) has three main components: a tube, a barrel, and an output rod. The base is made of MDF and includes a flat plate with two vertical MDF boards mounted on it. These boards have a series of small holes spaced in 5-degree increments, which allow the barrel angle to be adjusted (**Figure 2.1**). Each board also has a larger center hole where a rod is inserted, letting the barrel pivot to the desired angle. The barrel, which is also made of MDF, is attached to the base through this center pivot rod. Inside the barrel are circular supports that help guide the output rod so it moves straight during launch (**Figure 2.2**). Energy is stored using bungee cords that are connected from the front of the barrel to the back end of the output rod. The output rod has holes along its rear section spaced 0.5 inches apart. These holes allow the user to change how far the rod is pulled back, which controls the amount of energy initially stored and the amount of energy used at launch.

Performance

During our first test on demo day 1, we were able to successfully launch the ball about 4 meters using a pullback of 8.5 inches on the output rod and an angle of 35 degrees. However, during demo day part 2, we were not able to hit the target, which required a 13.7g ball to land at 4 meters and reach a height of 0.5 meters at 2.5 meters. On our first attempt, we used the maximum pullback of 8.5 inches and an angle of 10 degrees. This led to the ball launching too low, going under the hole, and landing early. On our second attempt, we kept the pullback at 8.5 inches but increased the angle to 15 degrees. This time, the ball went over the hole but landed slightly short of the target at about 3.8 meters.

Model

The model consists of three main sections: finding energies, finding kinematic properties, and finding the best launch angle. Pursuing an energy balance approach, the following equations were used to find the bungee cord energy, the energy lost from the friction between the rod and the supports, and the energy lost from drag (**Equations 1-3**). These energies were combined to find the net energy used when the ball leaves the system. This net energy value was then used to calculate the initial x and y velocities. From the x and y initial velocities, we formed a loop that finds the x and y positions of the ball at different times. The loop breaks when y < 0, letting us make sure that we can properly measure where the ball will land.

$$E_b = (k * x^{b-1})/(b - 1)$$

Equation 1: *Energy stored in the bungee cord. k is the spring constant, b is the non-linearity exponent, and x is the length of pullback.*

$$W_f = \mu * m * g * x$$

Equation 2: *Energy dissipated from work due to friction.* μ *is the coefficient of kinetic friction, m is the mass, g is the gravitational constant, and x is the length of pullback.*

$$W_d = 1/2 * \rho * (v *.6)^2 * Cd * A * x$$

Equation 3: *Energy dissipated due to drag force.* ρ *is the density of air, v is the velocity of the object, Cd is the coefficient of drag of the ball, A is the cross-sectional area, and x is the length of pullback.*

To find the correct launch angle and pullback strength, the user has to first input the desired gate distance, ball mass, gate height, and total flight distance of the ball. These values are then input into a binary search algorithm. In essence, the binary search algorithm loops through the trajectory code for each angle from 10 to 85 degrees. For every angle, the code tests to see what the correct pullback distance is by going through 15 iterations and clearing out what values get

the ball too far or too short from the desired distance. After finding the correct pullback distance, it checks to see what height the ball would be at the gate distance. At the end, the error between the ball's gate height and the ball's total distance is minimized to see what the best launch angle and pullback length is. Once the error is minimized, the energy required in the system, the launch angle, the pullback length, and the error are all printed.

In developing our model, we aimed to represent real-world conditions as closely as possible by incorporating the friction coefficients of the launcher materials, the spring constant of the bungee cords measured on the MTS machine (**Figure 3**), and the drag coefficient of the spherical projectile. Despite using experimentally derived and literature-based values, the model did not perfectly capture real-world behavior. To adjust for this, we incorporated a "correction factor" that scaled the required input energy needed during the launch through real-world tests before demo day. When calibrating, we first launched our projectile at a constant angle, only varying the power, and then measured the distance that each test went. We then used the same conditions used in the experimental projectile launch and placed them in a theoretical code-based launch to compare values. These datasets were then put into a for loop, which tracks the error between the experimental and the theoretical, multiplied by a calibration factor. At the end of the loop, the lowest error's calibration factor is taken out and placed within the code.

An additional method used to validate our model was comparing our assumed acceleration values to the actual acceleration data collected by the telemetry device on our first demo day. **Figure 4** depicts the x-acceleration, y-acceleration, z-acceleration, and acceleration magnitude graphed against time. The individual acceleration directions remain relatively constant while the ball is airborne, indicating minimal spin and allowing us to neglect the Magnus Effect in our model. The two spikes in acceleration are assumed to be the moments when the ball is "punched" by our launcher and the moment of ground impact, respectively. Immediately after the first acceleration magnitude spike of 33.84 m/s^2, it drops to ~3 m/s^2, which we can assume to be our drag acceleration, since the telemetry device measures proper acceleration, neglecting gravity.

Figure 5 depicts our model's idealized understanding of the projectile's acceleration magnitude on demo day. The net acceleration is greater than gravity on the ascent of its flight path because drag and gravity are oriented in relatively the same direction, causing their acceleration magnitudes to be constructive. The opposite takes place on the ball's descent. **Figure 6** depicts acceleration broken down by direction. The horizontal acceleration illustrates how our model accounts for drag being proportional to the square of net velocity. For this trial, the model assumed an acceleration due to drag of approximately $0.75 m/s^2$.

The telemetry data is highly noisy due to the short flight duration relative to the sampling rate, which made it extremely difficult to refine our drag coefficient solely based on the demo day

data. The actual drag acceleration was ~4x larger than what our model predicted. To refine our drag coefficient value, we slowly increased it, running consecutive real-world trials until our model matched the real-world height and length results.

Improvements

An issue that we ran into on testing day is that after our second run, our simulation showed that the correct settings were an angle between 10 and 15 degrees and a pullback length longer than what our launcher allowed. Based on this, one way we could have improved the physical design is by reducing the angle adjustment increments from 5 degrees to around 2-3 degrees, which would allow for more precise angle control. We could also have added another bungee cord to increase the amount of energy per distance pulled back, since we had already reached the max pullback length in our design. In addition, the output rod could only be pulled back in 0.5-inch increments, so using smaller increments would have allowed for finer control and more accurate energy adjustment. With these changes that would allow us to achieve a more precise angle as well as higher launch potential energy and therefore kinetic energy, and considering the fact that at 15 degrees it went over and at 10 degrees it went under, we think that our ball could have possibly gone through the hole.

To improve the model, a couple of things could be further examined and executed. First, the force vs displacement of the bungee cord should have been more properly implemented into the code. When creating the code, it was assumed that the force of the bungee cord would not be linear; it would be similar to $k * x^b$, where k and b are constants. While the graph of force vs length found a k value through a linear regression of the data, the bungee cord's non-linear function necessitated the introduction of another non-linear variable, b. Since b was not explicitly given as an independent variable within the graph, as the MTS gives a static snapshot of stiffness, b was tuned to an educated guess. To improve the model performance, we could have conducted power regression on our force vs length graph to calculate a more tailored b value for our selected bungee cords.

Additionally, in the model, the means of finding the x and y position vectors should have been more carefully considered. Euler integration, the way we found the positions at each time step, is not very proficient when examining fast-moving objects. It tends to cause drift, which at the end, will make the theoretical values much less accurate than necessary. Instead, Runge-Kutta 4th Order (RK4) should have been used. RK4 is a method where four slopes are created to find the slope at the next timestep. Through each slope calculation, it refines the accuracy until it reaches the 4th slope, which acts as the true calculation. When using RK4 instead of Euler's integration, one can more properly check changing values of a fast-moving object, for example, changing the drag force.

Finally, additional telemetry data collection at a higher sampling rate would have been useful to validate our drag coefficient. The variation in the data over a short flight time made it difficult to select a consistent value. In the future, we would ideally conduct more tests with the telemetry device, then average their drag acceleration values across multiple tests to get a more consistent value.

Appendix

Demo Day Launch

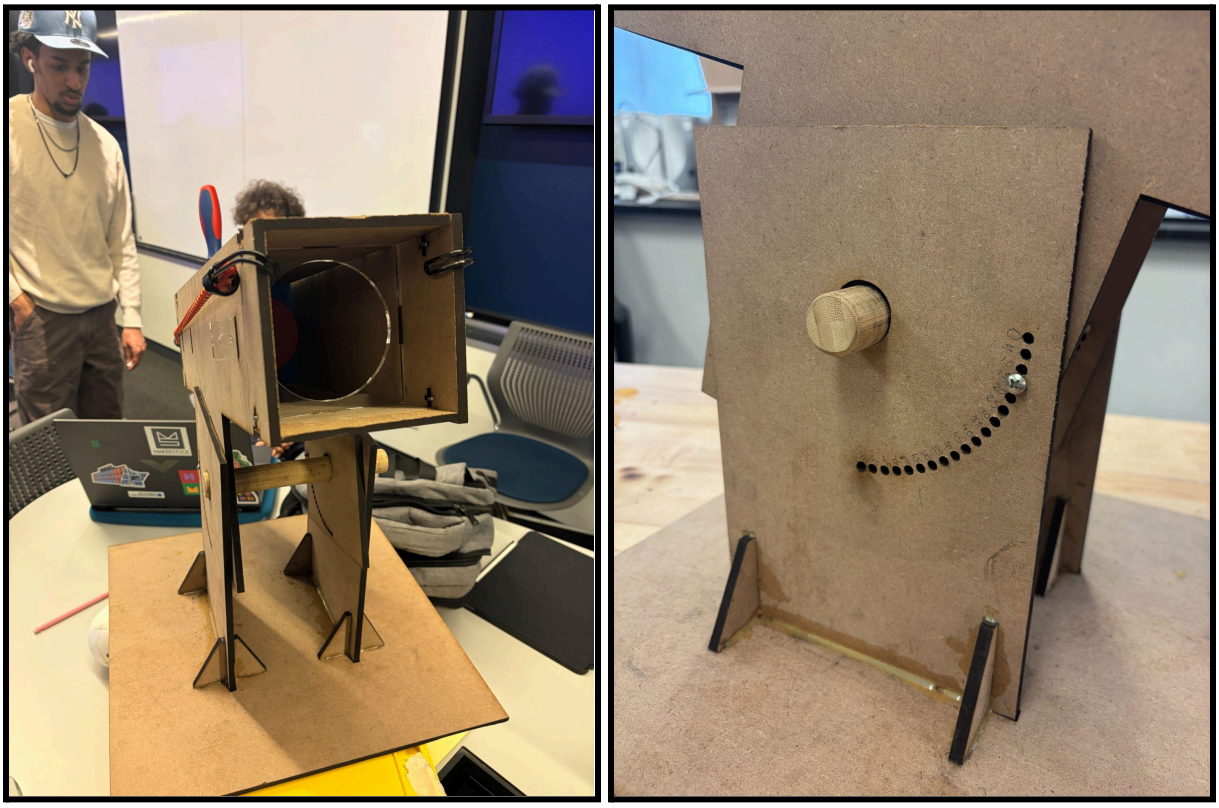


Figure 2.1: *Left: launcher barrel and ball holder. Right: angle adjustment system with angle notations.*

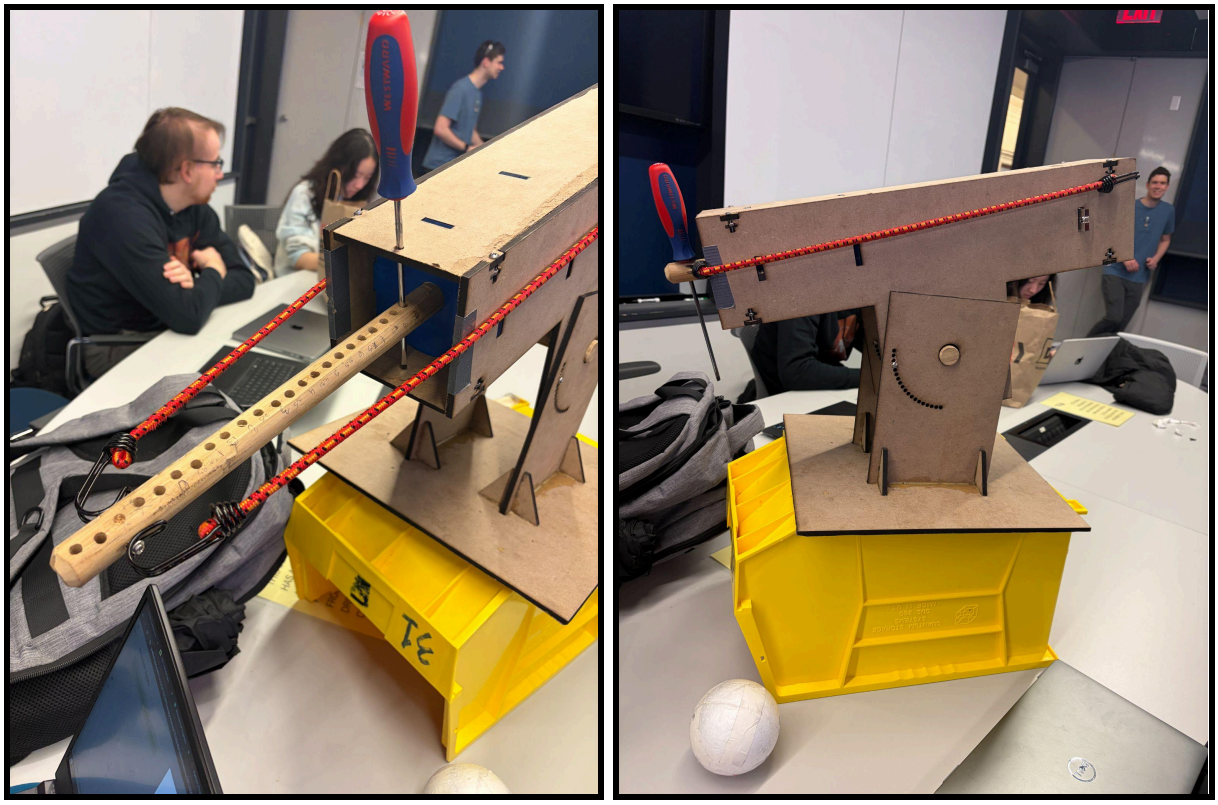


Figure 2.2: *Left: power adjustment with spring displacement notations and trigger mechanism. Right: completed launcher setup.*

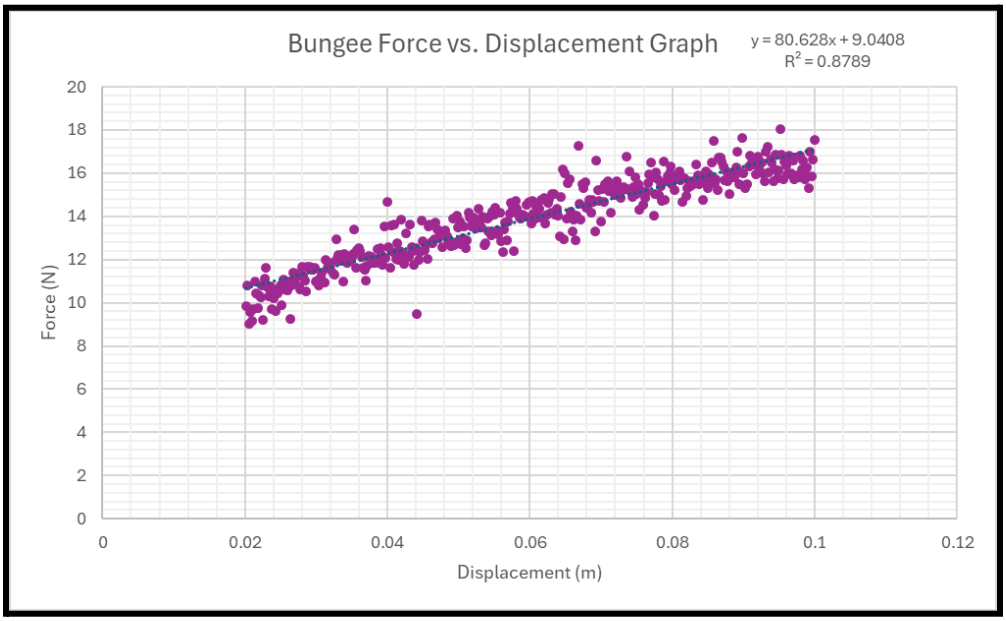


Figure 3*: Force vs Displacement graph for the bungee cables on the MTS Criterion. The test resulted in a spring coefficient of 80.628 N/m found through linear regression.*

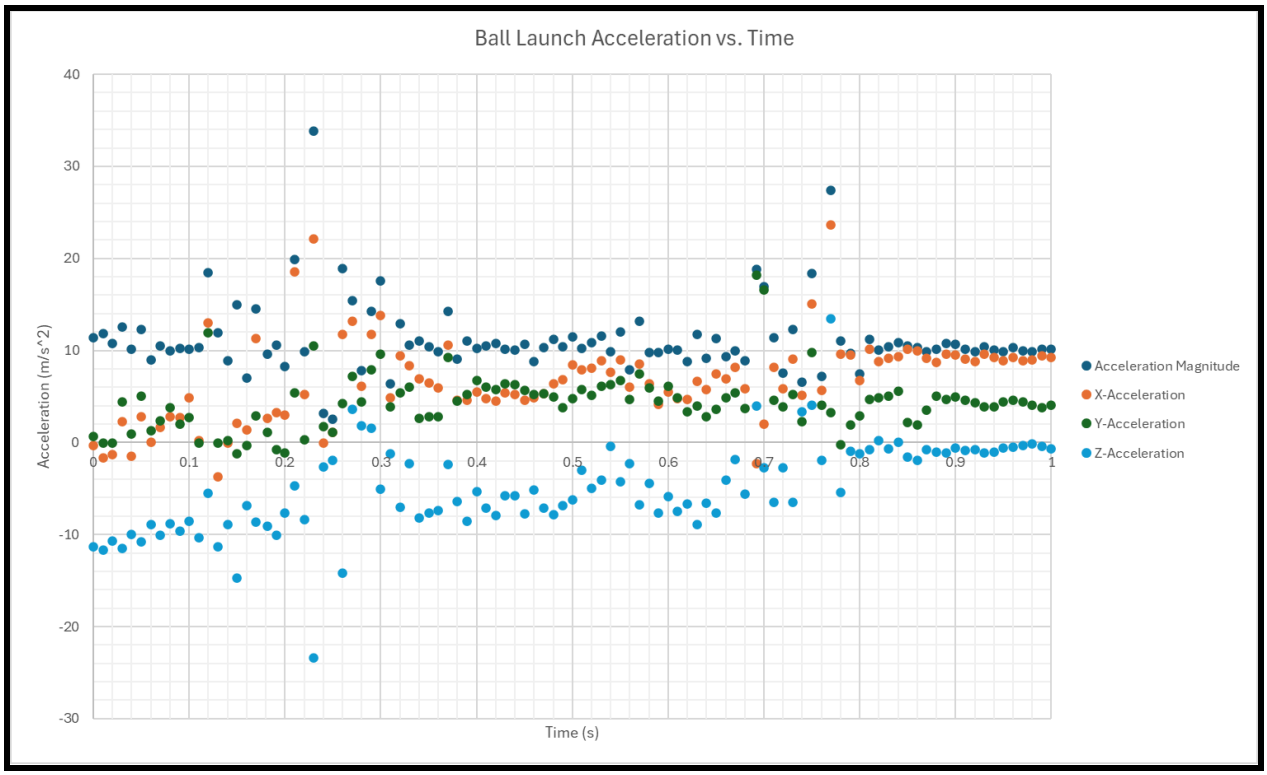


Figure 4: *Acceleration profile from the first demo day, with a range of ~4m. The two spikes indicate the moments of launch and impact. The launch spike is followed by moments of ~3 m/s² net acceleration, which is assumed to be drag during flight.*

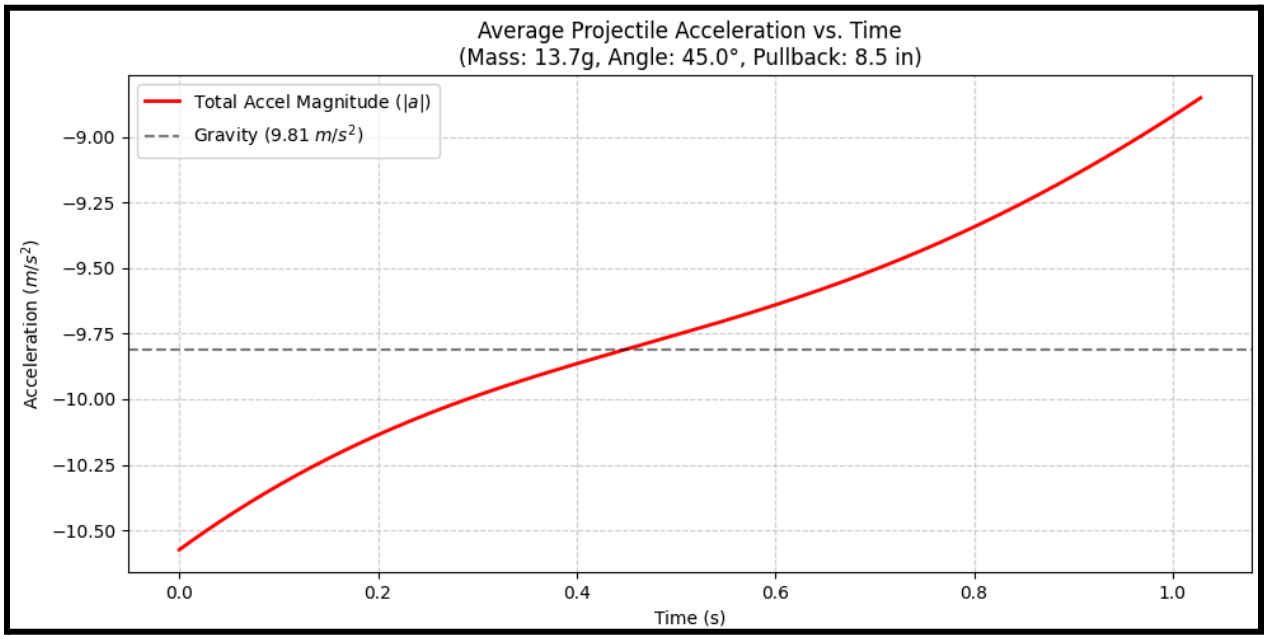


Figure 5*: Average Projectile Acceleration vs Time. The mass was assumed to be 13.7g, the angle was assumed to be 45 degrees, and the pullback was assumed to be 8.5 inches. The gravitational constant is displayed through the gray dashed line.*

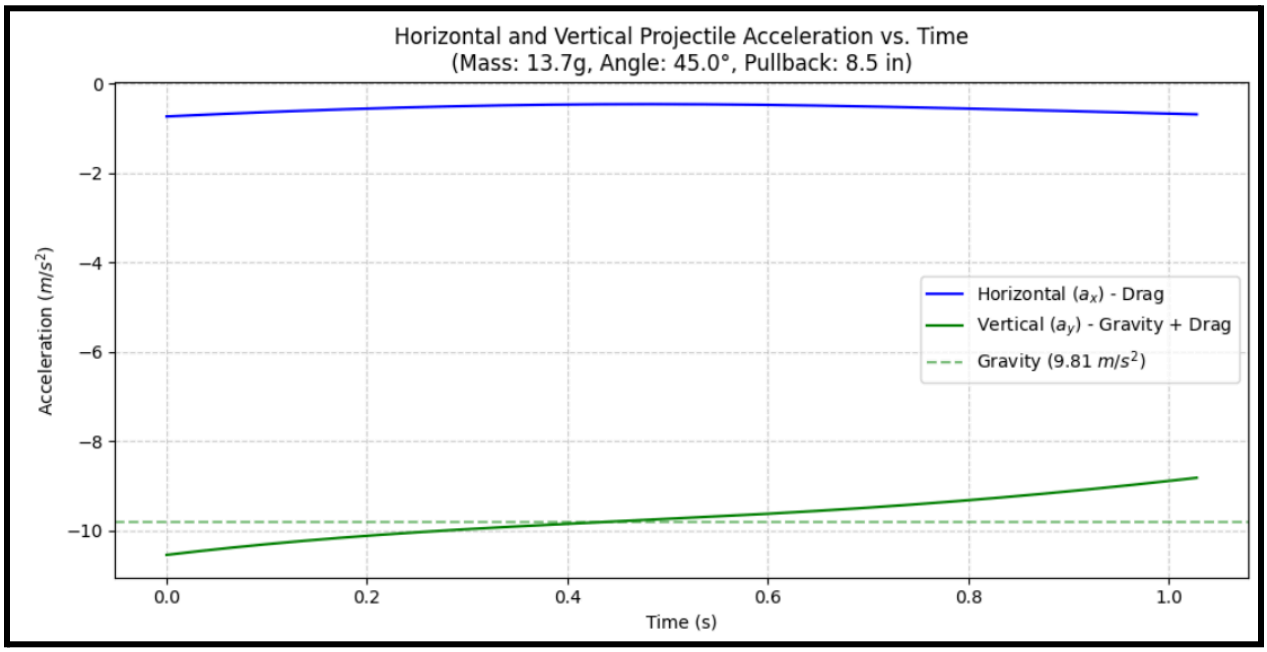


Figure 6: *Horizontal and Vertical Projectile Acceleration vs Time. The mass was assumed to be 13.7g, the angle was assumed to be 45 degrees, and the pullback was assumed to be 8.5 inches. The gravitational constant is displayed through the green dashed line.*

Code