Analyzing Acceleration of a Mounted Airglider on an Inclined Air Track using Arduino and Python

Clyde Villacrusis, Nathan Joshua, Wendy Mapaye

Physics 4AL, Fall Quarter 2023, November 5, 2023
Lab Section 3

Abstract

The goal of this lab activity is to confirm a theoretical prediction based on Newton's Laws of Motion and geometry via experiment. This was done using an Arduino based setup with a bluetooth module and ultrasonic sensor that collects data for time and distance from a cardboard wall. After conducting the experiment 3 times, Python was used to analyze this data to calculate acceleration of the setup and angle of elevation of the inclined air track on which it was placed. Through the analysis, we found that our results for Run 1 and Run 3 were in agreement with the expected angle of elevation of 2.7 degrees obtained by hand measurements. We also found that our Run 1 acceleration agreed with the predicted acceleration of -0.46 m/s^2 while Run 3 was quite close to the predicted value. Run 2 on the other hand did not agree with the prediction. Overall, these results demonstrate that the theoretical predictions are more or less accurate and agree with experimental results.

Introduction

The goal of this lab activity is to confirm a theoretical prediction based on Newton's Laws of Motion and geometry via experiment. The experiment involves a mounted glider that is released from rest on an inclined air track.

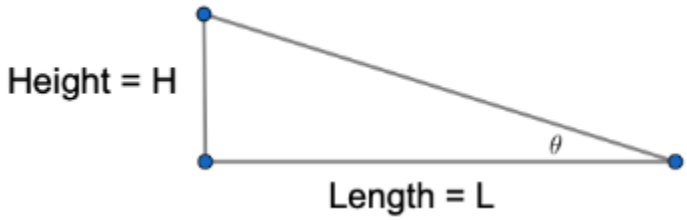

Figure 1: A visual representation of what the tilted air track looks like.

As per Newton's second law of motion, the total force acting on an object is a product of its mass and acceleration. If friction is made negligible, the only force acting on the mounted glider is the gravitational force. Along the axis of the inclined air track, the gravitational force acting on the glider is reduced to its gravitational component given by:

$$F = -mgsin\theta$$

Equation 1: Force acting on mounted glider along the axis of the air track

Then, the corresponding acceleration of the mounted glider and angle of elevation of the air track are given by:

$$a = -gsin\theta$$

Equation 2.1: Acceleration of mounted glider along the axis of the air track

$$\theta = sin^{-1}(-a/g)$$

Equation 2.2: Angle of elevation of the air track

We want to confirm if this theoretical result is in agreement with actual experimental data. Therefore, the setup atop the glider is configured in such a way that it measures the time and distance between itself and a fixed cardboard wall at the end of the track as it is released from rest. Given information about time and distance we can use the following kinematic equation to solve for acceleration,

$$x = x_0 + v_0 t + \frac{1}{2} a_0 t^2$$

Equation 3: Kinematic equation that gives us the distance x at a certain time t, where v_0 is the initial velocity, x_0 is the initial position and the acceleration a_0 is constant.

The setup atop the glider is an Arduino based setup with an ultrasonic sensor to make its measurements. Arduinos are micro-controllers that collect and analyze data while controlling various systems such as LEDs, ultrasonic sensors and bluetooth modules. Ultrasonic sensors are used to measure the distance and time from objects using soundwaves. They sense proximity and detect objects under high reliability. In particular, the HC-SR04 Ultrasonic Sensor used in the experiment emits a chirp using its transceiver which is reflected off of an object and returns to its detector. The distance is then calculated using the following equation,

$$d = \frac{1}{2} v_s \Delta t$$

Equation 4: Distance between object and HC-SR04 Ultrasonic Sensor where v_s is the speed of sound and Δt is the time elapsed since the soundwave left the transceiver until it returns to the detector

The data that is obtained using the Ultrasonic Sensor will require curve-fitting before **Equation 3** can be applied to obtain acceleration. This analysis can be done using Python. We hypothesize that our Python based analysis of the data collected by the Ultrasonic Sensor will yield values for acceleration of the mounted glider that agree with the theoretical prediction given by **Equation 2.1.**

Methods

Equipment

- Arduino board
- Python
- HC-SR04 Ultrasonic Sensor
- Four 1.5V batteries
- Voltmeter
- Battery holder
- HC-06 Bluetooth module
- Arduino IDE
- Air Track
- Tape
- Cardboard
- Ruler
- Books
- Glider

Begin by setting up the Arduino board so that it has the HC-SR04 Ultrasonic Sensor and HC-06 bluetooth module mounted on it as shown in **Figure 3**. The circuit diagram that describes the wirings for the setup can be seen in **Figure 2**. Following completion of the wiring process, the Arduino must be connected to a computer via USB so that the required code can be uploaded to the device from the Arduino IDE. The required code is written in C++ and uses the SoftwareSerial library for serial communication between the computer and the Arduino's digital pins.

In the code, one pin is set as a receiver and another is set as a transmitter to enable bluetooth communication with the computer. The remaining code ensures that the time and distance data collected by the Ultrasonic sensor is displayed in the Serial Monitor of the Arduino IDE. Once the code is uploaded, remove the USB cable and connect the setup to a battery holder. The battery holder must contain four batteries with at least 1.4 V each so that the setup can continue to function without losing power during data collection. This can be checked beforehand using a voltmeter.

At this point, the bluetooth module's red LED should have started blinking. This is a sign that it is not connected to the computer. Turn on the bluetooth on the computer and connect to the bluetooth module. The red LED should have stopped blinking. On the computer, select any COM port in the Arduino IDE and open the Serial Monitor to see the ultrasonic sensor values.

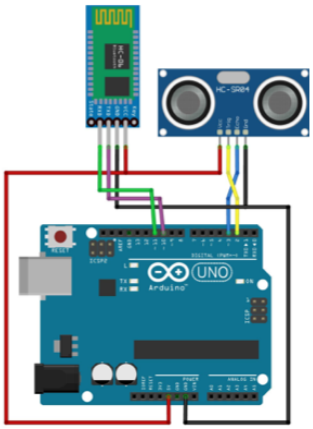

Figure 2: Circuit setup we followed from Lab 1D in order to properly wire our ultrasonic sensor and bluetooth module

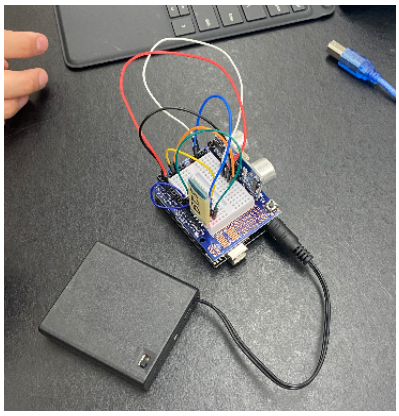

Figure 3: Our completed Arduino set-up after following Arduino wiring instructions and attaching it to our battery pack

Figure 4: Our experiment setup after attaching our Arduino to the glider and facing your ultrasonic sensor in the direction of our cardboard barrier.

Place the completed Arduino setup onto the air track glider as shown in **Figure 4**. Use tape to ensure that the Arduino is tightly bound to the glider. After this, attach a cardboard wall to the end of the air track to act as the object at a distance. Following that, turn on the air supply for the air track and set it to its maximum output. This will ensure that the glider will move freely along the air track allowing us to ignore the effects of friction.

Ensure that the air track is level. Once this is done, find the maximum distance from the cardboard where the Ultrasonic Sensor generates coherent data. This can be done by moving the glider setup farther away from the cardboard wall, while keeping an eye on the serial monitor data until it becomes evidently noisy.

The next step is to tilt the air track. Use a mount such as a stack of books to raise the end of the air track that does not have the cardboard wall taped to it. The air track should look like **Figure 1.** Measure the new height of the air track, H. Using this new height and the length of the air track, we can use trigonometry to determine the elevation angle by hand using the following equation,

$$\theta = sin^{-1}(H/length)$$

Equation 5: Angle of elevation of the air track measured by hand

Once the angle of elevation and the maximum distance for coherent data are known, set the glider to the max distance and get ready to collect data. Make sure that you provide nearly zero acceleration as you release it from the max distance so that we can assume that it is released from rest. As mentioned before, friction is also considered to be negligible since the glider is moving freely along the air track. Therefore, the only force acting on the glider effectively is the gravitational force. Repeat this process three times to check if the results are consistent.

After collecting the data, Python can be used to analyze it. The data is first converted to .txt files so that it can be read into the program. After uploading the txt files to the Python IDE, we can graph the data using scatterplots from the matplotlib library to see what it looks like. If all the preceding steps were completed without mistakes, the scatterplots should look nearly quadratic. Isolate the quadratic part of the data by using time and distance cut-offs to remove the noisy data at the beginning and end. Next, use the np.polyfit() function to obtain the best-fit curve for the data. This curve will be parabolic in nature and can be described by the following equation:

$$ax^2 + bx + c = 0$$

Equation 6: Equation describing best-fit curves of degree 2, where a, b and c are fit coefficients

Once this is done, we can compare **Equation 6** and **Equation 3** to find the acceleration of the glider and its uncertainty. The following equations describe these calculations:

$$a_0 = 2a$$

Equation 7.1: Acceleration calculated using fit-coefficients

$$\delta a_0 = |a_0| \frac{\delta a}{|a|}$$

Equation 7.2: Uncertainty in Acceleration

The value of δa in **Equation 7.2** can be obtained from the covariance matrix of the np.polyfit() function. The elevation angle and its uncertainty can then be calculated using **Equation 2.2** in the following manner,

$$\theta = sin^{-1}(\frac{2a}{g})$$

Equation 8.1: Angle of elevation calculated using fit-coefficients

$$\Delta\theta = \frac{\Delta a_0}{\sqrt{g^2 - a_0^2}}$$

Equation 8.2: Uncertainty in angle of elevation

Results

By following the procedure detailed in the Methods section, we found that the maximum distance before the ultrasonic sensor data became noisy was 45 cm. Following that, 3 runs of the experiment were conducted. The data collected from these runs were then converted to txt files and analyzed using Python. The scatterplots of the raw data obtained can be seen in **Figures 5-7.**

Scatterplots of distance vs. time raw data for three trials

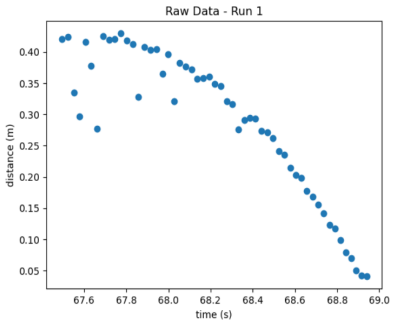

Figure 5: Raw distance v. time scatterplot data of Run 1

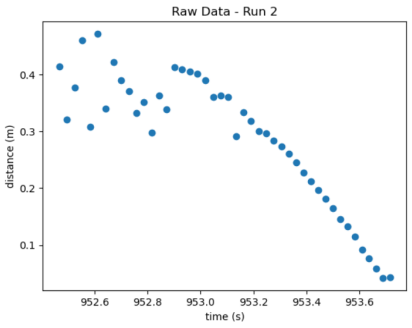

Figure 6: Raw distance v. time scatterplot data of Run 2

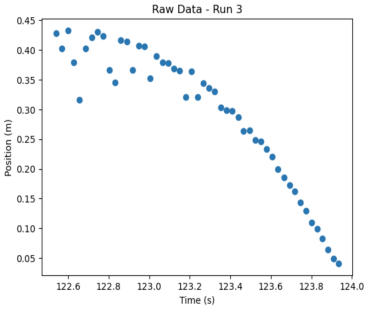

Figure 7: Raw distance v. time scatterplot data of Run 3

In each of the three runs, the initial and final data was noisy and contained inaccurate data points. To get more accurate results, we isolated the data that seem consistent and coherent by only taking certain time intervals. After cutting off the noisy data from the beginning and end, we get nearly quadratic plots as seen in **Figures 8-10**.

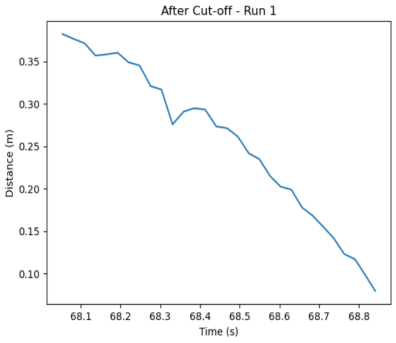

Figure 8: Distance v. time graph after implementing time cutoffs on Run 1

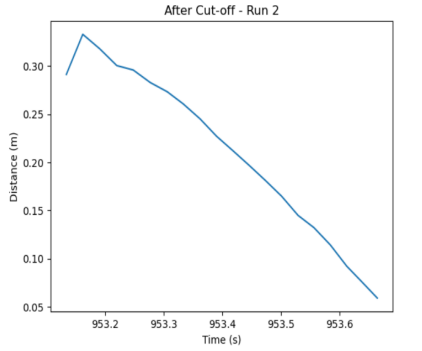

Figure 9: Distance v. time graph after implementing time cutoffs of Run 2

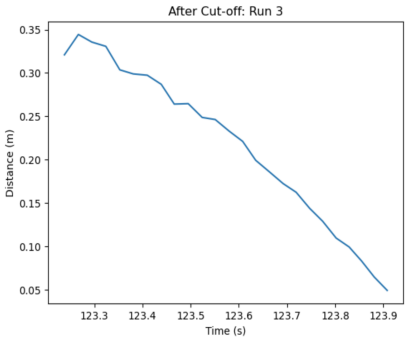

Figure 10: Distance v. time graph after implementing time cutoffs on Run 3

The next step was to create a best-fit curve as described by **Equation 6**. These were used to calculate the accelerations of the glider and the angles of elevation of the air track. We were able to fit our data into a quadratic equation, as seen in **Figures 11-13.**

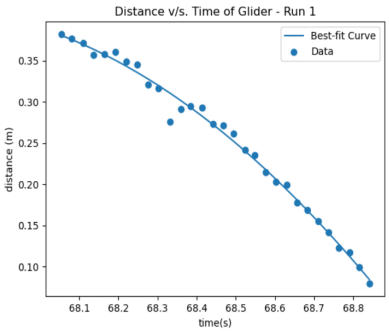

Figure 11: Distance v. time data of Run 1 with the quadratic line of best fit

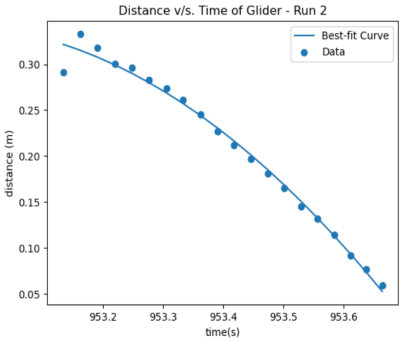

Figure 12: Distance v. time data of Run 2 with the quadratic line of best fit

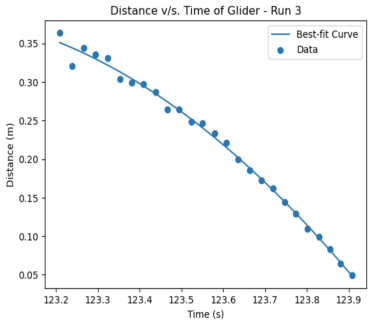

Figure 13: Distance v. time data of Run 3 with the quadratic line of best fit

The corresponding fit-coefficients that were found for each of the three runs are presented in **Table 1**. Using the numpy function, we were able to find these fit coefficients.

Table 1 - Best fit equations for each run

Fit-Coefficients	a	b	c
Run 1	-0.24375357033120165	32.99189611674982	-1115.9424086089366
Run 2	-0.556084771672345	1059.8334864666144	-504979.7779155987
Run 3	-0.341755470707448	84.02674458710041	-5164.501339270104

Best fit equations for each run using Equation 6:

Trial 1: $x(t) = (-0.2437)t^2 + 32.99t - 1115.942$
Trial 2: $x(t) = (-0.5560)t^2 + 1059.83t - 504979$
Trial 3: $x(t) = (-0.3417)t^2 + 84.02t - 5164.50$

The accelerations and associated uncertainties were calculated using **Equations 7.1** and **7.2** and are presented in **Table 2**.

Table 2 - Acceleration and uncertainty of accelerations

Run No.	Acceleration (m/s^2)	Uncertainty (m/s^2)
1	-0.5	0.1
2	-1.1	0.2
3	-0.68	0.08

The angles of elevation were calculated using **Equations 8.1** and **8.2** and are presented in **Table 3**.

Table 3 - Angle of elevation and uncertainty

Run No.	Angle of Elevation (degrees)	Uncertainty (degrees)
1	2.9	0.4
2	7.0	1.0
3	3.04	0.45

Before starting the experiments, the angle of elevation of the air track was calculated by hand using **Equation 5** and was found to be 2.7 degrees. This is in agreement with our results from Run 1 and Run 3 as shown in **Table 3** since it is within our margins of uncertainty. The corresponding acceleration of the glider that can be calculated using **Equation 2.1** is -0.46 m/s^2 which is also in agreement with our results from Run 1 since it is within the margin of uncertainty.

Conclusion

In our lab activity, we set out to find the acceleration of a mounted glider on an inclined air track using an Arduino with an ultrasonic sensor and a bluetooth module. We hypothesized that the values that we would gather from the Ultrasonic Sensor and Python analysis should accurately correspond with the theoretical predictions given by **Equation 2.1** and **Equation 2.2.** Based on the results of the Python based analysis of the 3 runs of the experiment, we came close to our hypothesized acceleration of -0.46 m/s^2. Run 1's acceleration of -0.5 ± 0.1 m/s^2 falls within the margin of error for our prediction. Moreover, Run 3's acceleration of -0.68 ± 0.08 m/s^2 is quite close to the expected value. Our hypothesized angle of elevation was 2.7 degrees. Runs 1 and 3 and their corresponding angle of elevations of 2.9 ± 0.4 degrees and 3.04 ± 0.45 degrees fall within the margin of error of our prediction. Run 2 gave us inaccurate results where the acceleration was -1.1 m/s^2 and the angle of elevation was 7.0 ± 1.0 degrees. We believe that this is due to the possibility of unnoticed errors.

One possible systematic error was the small or medium time delay of releasing the glider and starting the serial monitor. These delays could have skewed the results of our Run 2 data, showing large discrepancies in the fit coefficients and the angle of elevation. Another possible systematic error could have been that we did not accurately level the air track. This would've resulted in an unstable, quadratic graph as we saw in **Figure 6** with several noisy data points, affecting the fit coefficients, acceleration, angle of elevation, and uncertainties. Another possible error could be parallax error while measuring the height of the air track.

To improve our experiment so that we can yield better results, we can increase the number of runs, so that we don't have to rely on one inaccurate run when analyzing our data. The overall data will average out the inaccurate data as we do more and more runs. Another change that could be made to improve the experiment would be to ensure that the batteries powering the Arduino have enough voltage (> 2V) throughout the experiment. Since the experiment had been running for a long time, it is possible that the batteries would have lost some of the required power which can lead to noisy data and a messier output from the Ultrasonic Sensor. Most importantly, making sure that all the steps in the **Methods** section are followed diligently can reduce the chances of errors.

Overall, this experiment demonstrated that we can use **Equation 2.1** and **Equation 2.2** are more or less accurate predictions that agree with experimental results. With clean data and minimized human error, the calculated accelerations and angles should agree with the predictions.