**DS4 Drop**

**LAB #5**

**SECTION 5**

**SUBMITTED BY:**

**Jackson Collalti**

**SUBMISSION DATE:**

**10/10/2022**

**Problem**

Develop a code that detects free fall and narrates the action. During week one you will create a program that will detect the distance the controller fell using time. Week 2, modify your program from week 1 that calculates the distance with air resistance and the error caused by air resistance. To narrate your program will need to print a period for every second waiting, then during free fall print a “!” every half second. In the end, an output should state the distance fell and the duration of the fall. Week 2 should also include the % difference in fall as well as the corrected fall distance.

**Analysis**

Part 1

Inputs:

include time, acceleration in x, y, and z directions

Outputs:

Distance and time have fallen

Desired output format:

<First Name> <Last Name>

<Netid>

Ok, I'm now receiving data.

I'm Waiting ..............

Help me! I'm falling!!!!!!!!!!!!!

Ouch! I fell 1.633 meters in 0.577 seconds.

Variables:

timer for printing periods and exclamation marks, distance, and start fall time

Given Equations:

x1 = x0 + vt + (1/2) g t^2

g =9.8

Assumptions:

You may assume no air resistance

Part 2

Inputs: include time, acceleration in x, y, and z directions

Outputs: Distance with air resistance and the percent difference in distances.

Desired Output Format:

<First Name> <Last Name>

<Netid>

Ok, I'm now receiving data.

I'm Waiting ..............

Help me! I'm falling!!!!!!!!!!!!!

Ouch! I fell 1.633 meters in 0.577 seconds.

Compensating for air resistance, the fall was 8.777 meters. This is 7% less than computed before.

Variables:

timer for printing periods and exclamation marks, distance, and start fall time. Variables for the equation below previous time and velocity then current velocity and time. Variable for percent difference in distances.

Given Equations

vi =vi-1 + g (1 − mag(acc)) (ti −ti-1 )

xi = xi-1 + vi (ti −ti-1 )

**Design**

Using close\_to and mag functions form previous labs is a great place to start. I first am going to need a loop to print periods after every second of waiting. I will plan on using a timer variable and compare that to time. This will need to loop while the controller isn’t moving. Meaning a close\_to of the mag function will be the condition of my first loop.

My second loop will be similar when it comes to printing exclamation marks. This timer will be shorter because the fall time will be a short period of time. I plan on making the interval every 200 milliseconds.

After the second loop’s condition fails I’ll need to calculate the distance the controller fell using the equations. To find the time I plan on having a variable that is equal to the time the controller starts falling. This means, fall time = (program runtime) - (time before the start of fall). Then I need to print the distance and fall time.

For part two, loops one and two act very similarly, but in loop two I need to use the equation for velocity and position. In the end, I’ll need to divide fall time with resistance by the original fall time and get that into a percentage. After that, I can print the results of the % difference and fall distance with air resistance.

**Testing**

Before doing the demo of dropping the controller, I need to run my code with the given test data from previous attempts to make sure the program runs as intended. During this testing, I’ll need to do actual drops of short length. This will be around the range of a 0.5-meter drop to a meter-long drop that ends with me catching it. After passing those tests then I move on to actually dropping the controller in a box a significant distance second or third floor of Coover.

**Comments**

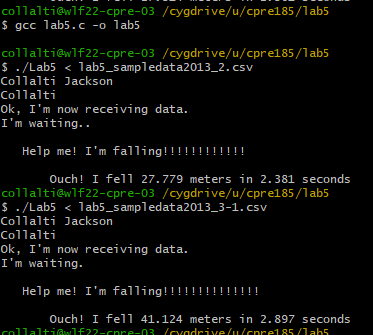
During this lab I had to get used to creating and working with many variables especially in part two trying to calculate the distance including air resistance. I also had to work with loops since that was the main thing that decide if your program worked or not. Problems encountered were that I had to use different tolerances for loop two when running test data set 1 than data sets 2 and 3 because of max values. For data set one I used while(mag(ax, ay, az)<=4) for data sets 2 and three is used while(mag(ax, ay, az)<=5). More reasoning later on with mag graphs.

**Lab Questions**

**1.** Do the same drop 5 times in the classroom and record the distances. How consistent are your results? What could cause any variation?

For my drops in the classroom, the max distance I could drop was about a meter because of the cord length. This meant that the drop times were very short resulting in similar values, but with some errors. Part of it was I was eyeballing distances so the desired distances weren’t exact. Of my five trails, the range was between 1.1 meters to 0.7 meters. Three of my five were about 0.9 meters.

**2.** Run your program with sample data from a previously recorded drop by using the following command: ./DS4Drop < lab5\_sampledata2013\_1.csv How far is it from the third floor railing to the bottom floor according to your code, using the sample data?



**3.** In your report, include a graph of the magnitude of the acceleration as a function of time from the sample data. Label where the freefall is happening, where it hits the ground, and your tolerances. Explain and justify any tolerances you are using. To get this graph in excel, implement the mag function by typing the following into the box E1: =SQRT(B1\*B1+C1\*C1+D1\*D1) Extend this calculation to every row by clicking the dragging the small black box that appears when cell E1 is highlighted down to the last row of data. Graph columns E vs A. You will not be able to use the sample data file as input to your code once you add column E. Rename the file once you have your graph and redownload the original file.

**lab5\_sampledata2013\_1.csv**



**lab5\_sampledata2013\_2.csv**

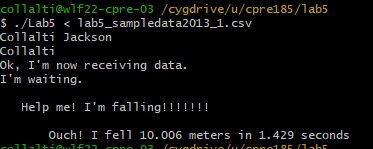


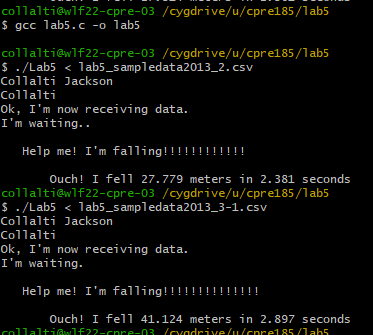
**lab5\_sampledata2013\_3-1.csv**

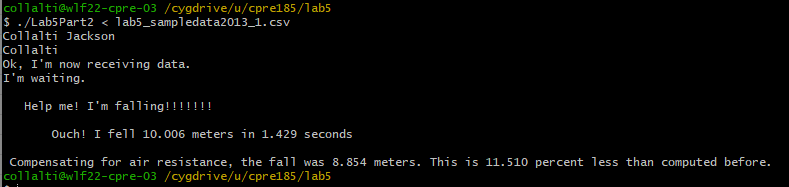


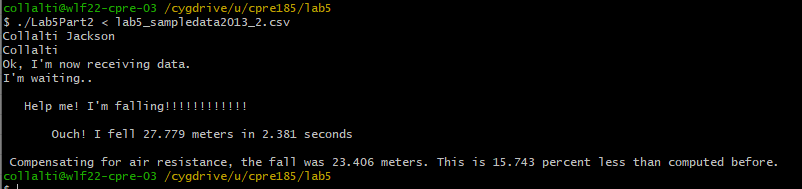
As stated in the comments I had different tolerance values for 1 compared to 2 and 3. For data set 1, the mag never reaches 5 till after the controller hits the ground thus the reason I had a tolerance value of 4. As seen in data set 3 (Red Circle) I couldn’t use a tolerance of 4 because then the program would stop early before the controller would hit the ground. I decided on using a tolerance of 5 for data set 2 because it will won’t stop early mid-flight.

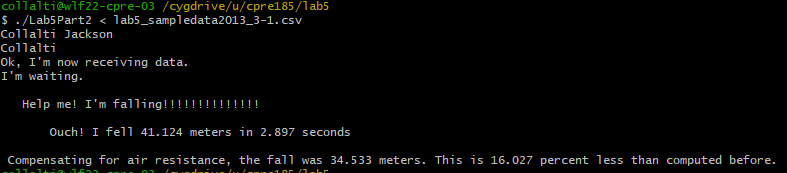
**4.** Demo your source code to an undergraduate TA using our 3 story drop data set and have them enter your demo score into the “Lab 05 Demo” column in Blackboard. Be sure your comment your the code and functions that you write.



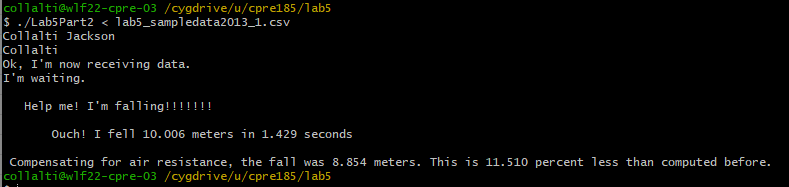








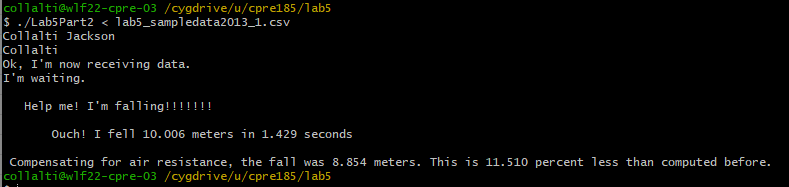
**1.** How much difference tends to occur in drops in the lab area? From the 2nd floor?



11.510 percent difference

**2.** How far is it from the second-floor railing to the bottom floor according to your code? If your program is not done by end of this week's lab, you can capture data ds4rd.exe data to a file using: ./ds4rd.exe -d 054c:05c4 -D DS4\_BT -t -g > output.csv Test your code by running: ./exdrop <

lab5\_sampledata2013\_1.csv 3.



8.854 meters

**3.** What issues arose in implementing Part 2?

The hardest part was definitely figuring out how to do the last value of time and the current value of time. At first, I started by having the equations outside of the loop but I quickly realized that this wouldn’t work.

**Lab5 Part 1 Code**

/\*-----------------------------------------------------------------------------

- CPRE 185 Lab 02

- Name:Jackson Collalti

- Section:5

- NetID:collalti

- Date:9/27/22

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Includes

-----------------------------------------------------------------------------\*/

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

/\*-----------------------------------------------------------------------------

- Defines

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Prototypes

-----------------------------------------------------------------------------\*/

int close\_to (double tolerance, double point, double value);

double mag(double ax, double ay, double az);

int seconds(int t);

/\*-----------------------------------------------------------------------------

- Implementation

-----------------------------------------------------------------------------\*/

int main(void)

{

int t, b1, b2, b3, b4;

double St=0.0;

int timer = 0;

double ax, ay, az, gx, gy, gz;

double toleranceNM=1.0;

double dist=0.0;

;

printf("Collalti Jackson\nCollalti\n");

printf("Ok, I'm now receiving data.\n");

printf("I'm waiting.");

scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

while(close\_to(0.2, 1, mag(ax, ay, az))==1){

{

scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

if(t>=timer+1000){

printf(".");

fflush(stdout);

timer=t;

}

}

}

St = t;

printf("\n\n Help me! I'm falling");

fflush(stdout);

while(mag(ax, ay, az)<=5){

scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

if(t>=timer+200){

printf("!");

fflush(stdout);

timer=t;

}

}

St=(t-St)/1000.0;

dist = 0.5 \* 9.8 \* (St\*St);

printf("\n\n Ouch! I fell %.3lf meters in %.3lf seconds", dist,St);

return 0;

}

int close\_to(double tolerance, double point, double value) {

if(point-tolerance<=value && value<=point+tolerance){

return 1;

}

else {

return 0;

}

}

double mag(double ax, double ay, double az){

double Mag;

Mag = sqrt(ax \* ax + ay \* ay + az \* az);

}

**Lab5 Part 2 Code**

/\*-----------------------------------------------------------------------------

- CPRE 185 Lab 02

- Name:Jackson Collalti

- Section:5

- NetID:collalti

- Date:9/27/22

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Includes

-----------------------------------------------------------------------------\*/

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

/\*-----------------------------------------------------------------------------

- Defines

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Prototypes

-----------------------------------------------------------------------------\*/

int close\_to (double tolerance, double point, double value);

double mag(double ax, double ay, double az);

int seconds(int t);

/\*-----------------------------------------------------------------------------

- Implementation

-----------------------------------------------------------------------------\*/

int main(void)

{

int t, b1, b2, b3, b4;

double St=0.0;

int timer = 0;

double ax, ay, az, gx, gy, gz;

double toleranceNM=1.0;

double dist=0.0;

double diffT = 0.0;

double tempT=0.0;

double velo;

double position=0.0;

double percentDif=0.0;

printf("Collalti Jackson\nCollalti\n");

printf("Ok, I'm now receiving data.\n");

printf("I'm waiting.");

scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

while(close\_to(0.2, 1, mag(ax, ay, az))==1){

{

scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

if(t>=timer+1000){

printf(".");

fflush(stdout);

timer=t;

}

}

}

St = t;

printf("\n\n Help me! I'm falling");

fflush(stdout);

tempT = St;

velo = 0.0;

while(mag(ax, ay, az)<=5){

scanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);

if(t>=timer+200){

printf("!");

fflush(stdout);

timer=t;

}

diffT = (t - tempT) / 1000.0;

velo = velo + 9.8 \* (1 - mag(ax, ay, az)) \* (diffT);

position = position + velo \* (diffT);

tempT = t;

fflush(stdout);

}

St=(t-St)/1000.0;

dist = 0.5 \* 9.8 \* (St\*St);

percentDif = (1 - position / dist) \* 100;

printf("\n\n Ouch! I fell %.3lf meters in %.3lf seconds", dist,St);

printf("\n\n Compensating for air resistance, the fall was %.3lf meters. This is %.3lf percent less than computed before."

,position, percentDif);

return 0;

}

int close\_to(double tolerance, double point, double value) {

if(point-tolerance<=value && value<=point+tolerance){

return 1;

}

else {

return 0;

}

}

double mag(double ax, double ay, double az){

double Mag;

Mag = sqrt(ax \* ax + ay \* ay + az \* az);

}