# Wiidrop

## Lab #3: Section J

**Submitted By: Wen Zhanghao**

**Submission Date: March 4, 2015**

# Overview

Over all, we are given two .csv file to analyze the data in it. There are four columns in the file that represents time, x, y, and z, (the force that the wiimote detected). In the Part 1, we used MATLAB to draw the graph that time versus magnitude. We analyzed the graph to understand what the situation of object when it is falling.

In Part 2, basically , we write code and redirect the input from .csv file to show how wiimote fall through screen. Character like exclamation and period are used to represent the time of object's waiting, falling. Finally, the drop time and drop displacement have been calculated.

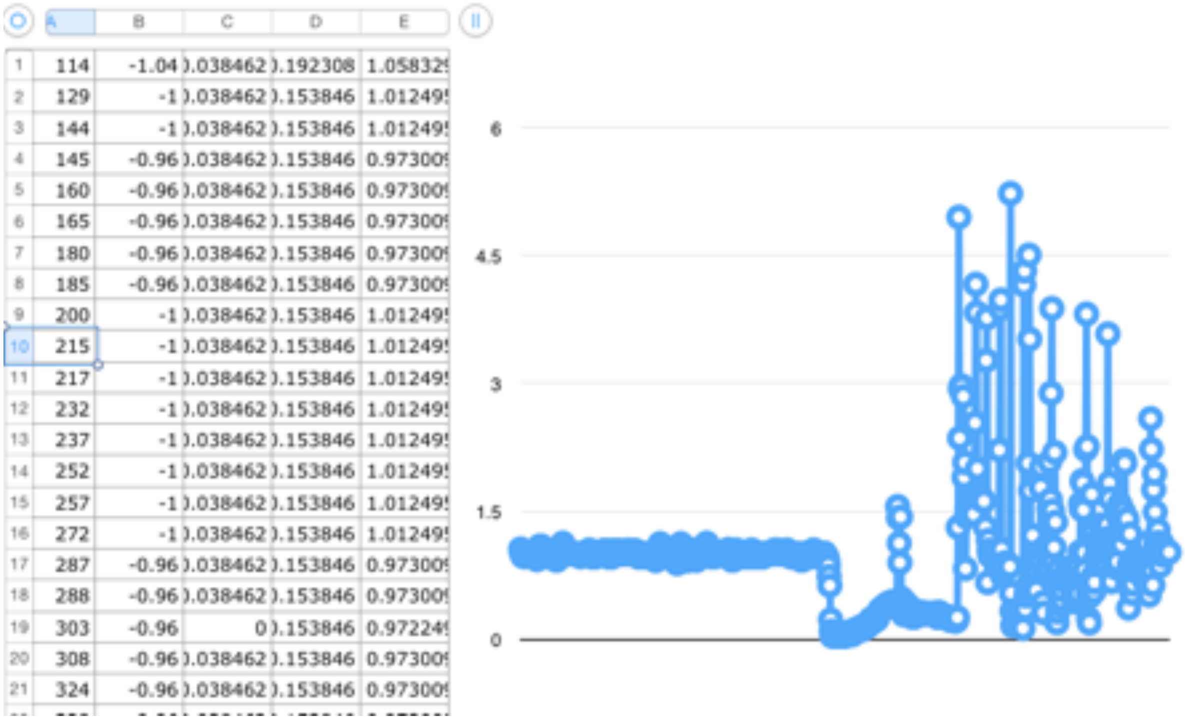
# Analysis

First of all, I figure it out that why at the beginning of dropping the net force is not zero. And how wiimote 'feel' the force. According to the graph, we labelled to divide graph into three parts, waiting, dropping, and impacts. During this process, we can roughly evaluate at what moment what motion is happening. To see the exact data, we collect and transfer .csv file into excel file and compute one column for magnitude and check the exact mag value corresponding specific time. In addition, we can pick part of data in excel to draw much more clear graph than matlab.

As required, we need design certain amount of period and exclamation to describe the time. So the bid idea is use count code to record times and print certain things.

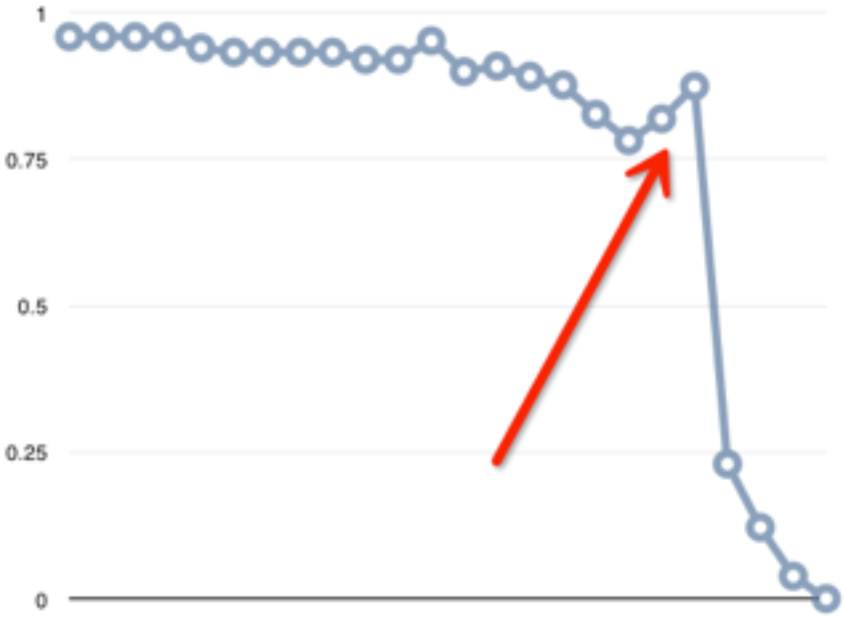
This data is from real falling object, so it is not in the ideal situation, which means that lots of 'noise' are produced. We need to define tolerance to avoid error when dealing with 'noise'. I created a .c file that measure the range of magnitude in the certain amount of time zone.

# Design



|  | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 114 | -1.04 | ).038462 | ).192308 | 1.058329 |
| 2 | 129 | -1 | ).038462 | ).153846 | 1.012495 |
| 3 | 144 | -1 | ).038462 | ).153846 | 1.012495 |
| 4 | 145 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 5 | 160 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 6 | 165 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 7 | 180 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 8 | 185 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 9 | 200 | -1 | ).038462 | ).153846 | 1.012495 |
| 10 | 215 | -1 | ).038462 | ).153846 | 1.012495 |
| 11 | 217 | -1 | ).038462 | ).153846 | 1.012495 |
| 12 | 232 | -1 | ).038462 | ).153846 | 1.012495 |
| 13 | 237 | -1 | ).038462 | ).153846 | 1.012495 |
| 14 | 252 | -1 | ).038462 | ).153846 | 1.012495 |
| 15 | 257 | -1 | ).038462 | ).153846 | 1.012495 |
| 16 | 272 | -1 | ).038462 | ).153846 | 1.012495 |
| 17 | 287 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 18 | 288 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 19 | 303 | -0.96 | 0 | ).153846 | 0.972249 |
| 20 | 308 | -0.96 | ).038462 | ).153846 | 0.973009 |
| 21 | 324 | -0.96 | ).038462 | ).153846 | 0.973009 |

This is data analysis from excel. At here we can check the exact value.And we can redraw much more detailed picture to see the local trends graph, such as the picture below.



In my Range.c file that measure the range to avoid bad influence of ‘noise’, I looked at graph first and roughly conclude at what moment of time the trend changed, and before that changed, I defined storeMax and storeMin value to record max and min value of magnitude going through this range of time. For example, for the first drop, the drop time somewhere around 4422, so I make time < 4500 to compute the max value of mag. The same method are used to the range of dropping and the second drop. Below is a summary data chart.

| The Range for Waiting: | | |
|---|---|---|
| | Max | Min |
| First Drop | 1.126585 | 0.886699 |
| Second Drop | 1.306891 | 0.885449 |
| Max/Min range input | 1.35 | 0.85 |

| The Range of Dropping | | |
|---|---|---|
| | **Max** | **Min** |
| **First Drop** | 0.88 | 0.1 |
| **Second Drop** | 0.73 | 0.0 |
| **Max/Min range Input** | 0.88 | 0 |

I was confused about why the initial magnitude is not zero if the object remain the same. After online research I know that the force detected is by a kind of sensor. Therefore, sensor cannot detect the force exerted on wiimote by earth because that is not directly touched. The directly touched is the third floor of Coover so the sensor below the object can detect the normal force. So this is the reason wiimote is at around one net force initially. After dropping, object’s sensor cannot feel gravitity but the air resistance, and as the velocity becomes bigger the air resistance become bigger, too. So there is a small positive slope near the end of dropping before the first compact in the graph because sensor can catch the air force.

In the lab4.c main file, I defined 3 tolerance to make sure the accuracy. and G represent 9.81 gravity. Because we do not know the number of data user would give to us so we cannot use for loop. So the while loop has been used with the condition of

basic number add/minus tolerance. In the while loop, there are scanf code that take the values, mag is to compute the magnitude quickly after the scan. We know that each row has time, x, y, and, z, means that we can use count code to count how many lines read to printf characters. If the value of mag over number the range, the loop stops and the time should be stored. Then the second loop starts, The second loop is doing the same way, and also the time has been recorded to compute the distance dropped, which use the equation y = 1/2 g t^2
t = t2 - t1.

One thing is deserved mention is that we should initial the value of mag to start the first times loop of loops. And if the loops starts, in the loop the mag value changed as loop over and over again.

## Extra Credit Part

The ideal acceleration formula is not suitable when dealing with the real situation with air resistance, or air force that may comes from any direction. The other way to compute a more precise value of distance is use Riemann Sum.

$$v_i = v_{i-1} + a_i(t_i - t_{i-1})$$
$$x_i = x_{i-1} + v_i(t_i - t_{i-1})$$

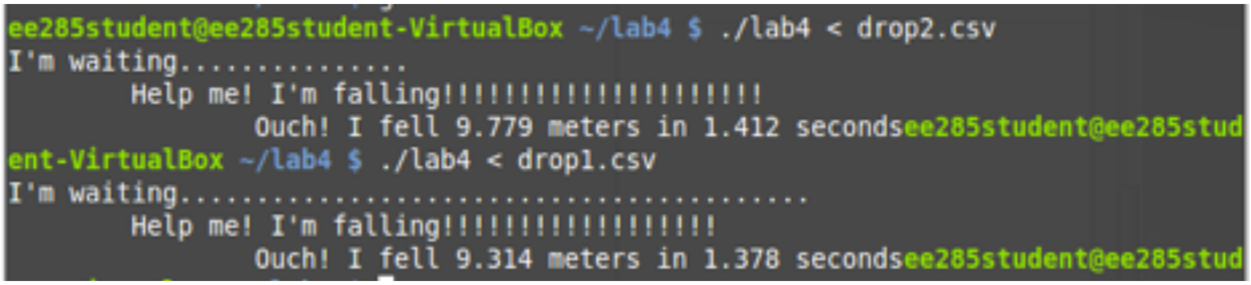
In  this formula, we compute the sum of current velocity by adding the previous velocity and velocity change. The velocity change is delta T multiplied by acceleration which is given by the magnitude in the graph with the g-unit. One important part is if we want to get correct value, the unit is important so we should convert the unit given into standard unit. change millisecond to second and g-unit to m/s^2.

Delta change are being recorded from the difference from current time and previous time value, I used a variable to store the time before a new time comes to be scan, and then use current time to store the value after the scan, and compute with previous variable and then change the unit.

The distance changed are added one by one with the starting value is zero. Each time we scan the value from .csv file, I compute the mag of acceleration, and delta t to calculate the velocity, and then use current x = previous x+delta T*current velocity to update the final distance at the end of loop.

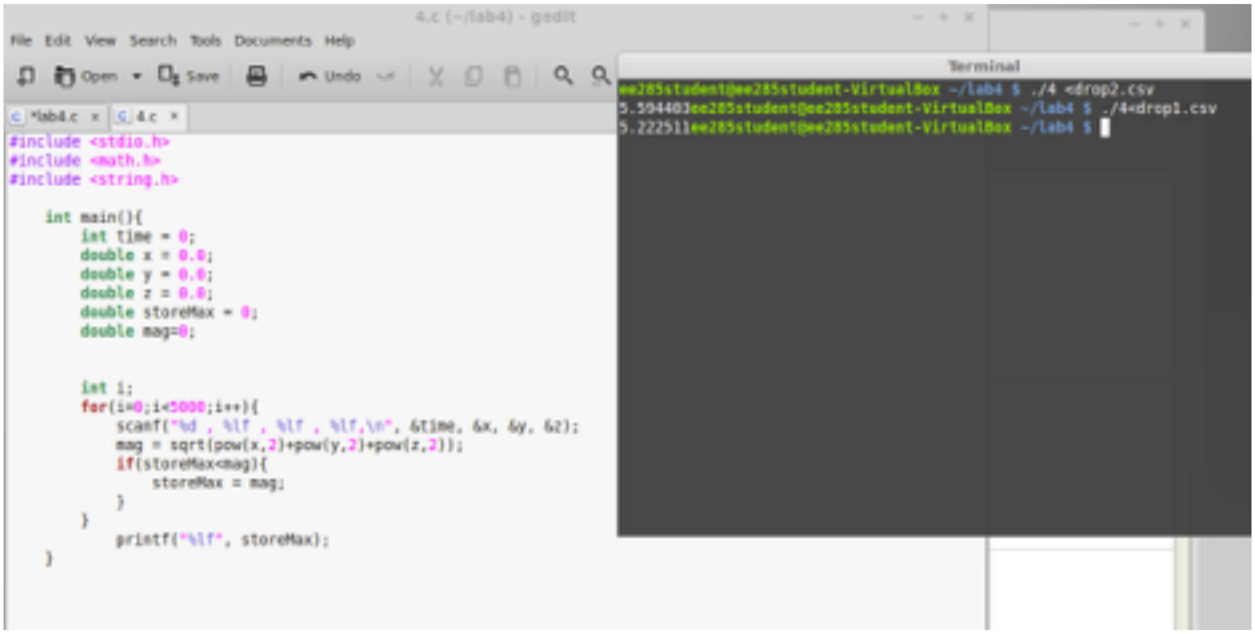
# Testing

Here is my test result:



```
ee285student@ee285student-VirtualBox ~/lab4 $ ./lab4 < drop2.csv
I'm waiting...............
        Help me! I'm falling!!!!!!!!!!!!!!!!!!!!!
                Ouch! I fell 9.779 meters in 1.412 secondsee285student@ee285stud
ent-VirtualBox ~/lab4 $ ./lab4 < drop1.csv
I'm waiting.........................................
        Help me! I'm falling!!!!!!!!!!!!!!!!!!
                Ouch! I fell 9.314 meters in 1.378 secondsee285student@ee285stud
```

Here is a common mistake I made when I try to use for loop to compute the range of data before wiimote start falling. Based on the first graph printed, I supposed that the unit of horizontal line is time adding one each time, so I plug in i < 5000 for store the max value in the first 5000 milliseconds. Therefore, the value turned out over 5 more than expected around 1.

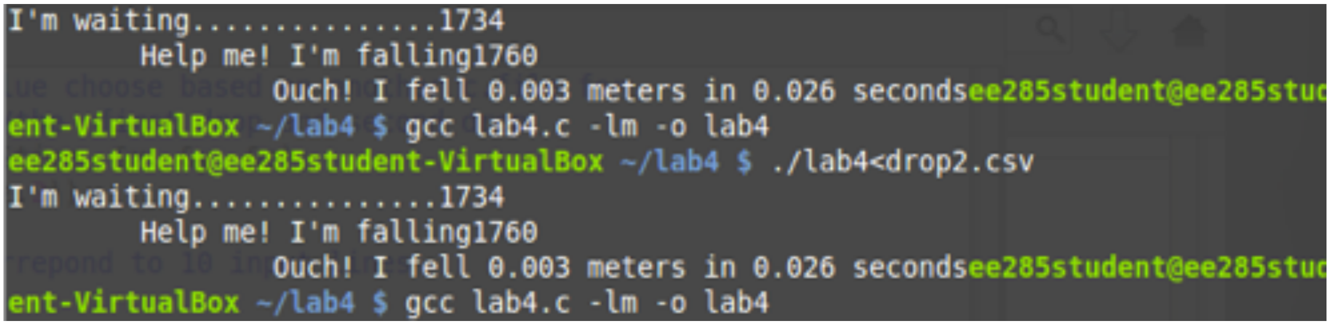


```
#include <stdio.h>
#include <math.h>
#include <string.h>

    int main(){
        int time = 0;
        double x = 0.0;
        double y = 0.0;
        double z = 0.0;
        double storeMax = 0;
        double mag=0;


        int i;
        for(i=0;i<5000;i++){
            scanf("%d , %lf , %lf , %lf,\n", &time, &x, &y, &z);
            mag = sqrt(pow(x,2)+pow(y,2)+pow(z,2));
            if(storeMax<mag){
                storeMax = mag;
            }
        }
            printf("%lf", storeMax);
    }
```

```
ee285student@ee285student-VirtualBox ~/lab4 $ ./4 <drop2.csv
5.594403ee285student@ee285student-VirtualBox ~/lab4 $ ./4<drop1.csv
5.222511ee285student@ee285student-VirtualBox ~/lab4 $ 
```

As a mater of fact, time does not count one by one. Look the graph below we can see that time skip tens value at one time, so the actually i value represents the very left column. At last, I used time as condition. First, based on graph, roughly collect the time when wiimote start drop, so I get time condition to compute exact magnitude range by using Range.c file.

There is a small trick in the second drop, there is a little rebound point that stop the loop continue, which lead a small delta time. So we should be careful about the tolerance.



```
I'm waiting...............1734
       Help me! I'm falling1760
              Ouch! I fell 0.003 meters in 0.026 secondsee285student@ee285stud
ent-VirtualBox ~/lab4 $ gcc lab4.c -lm -o lab4
ee285student@ee285student-VirtualBox ~/lab4 $ ./lab4<drop2.csv
I'm waiting...............1734
       Help me! I'm falling1760
              Ouch! I fell 0.003 meters in 0.026 secondsee285student@ee285stud
ent-VirtualBox ~/lab4 $ gcc lab4.c -lm -o lab4
```

## Comments

The skills of while loops and count code have been improved. Especially the condition of while loop, how to find the condition, the computation's order under the loop, how each variable changed based on what the order of computation. This lab is long-term programming which allows us to put more efforts in it. Always follow the idea do not get lost and make comments on the paper rather than simply note on the computer. Another skills of scan have been learned. The format of scan should be perfectly match the input format.

Another finding is that when we continue the unfinished code last time, it would take me more time than just reprogramming it again. So may be for understand what is going on about last code, I better redo it. For the change, I hope we can know more about how it works more quickly in order to program more efficiently.