2012

Where Is the Best Place to Sit on a Roller Coaster? Forces, Physics, and Fun at Disneyland

Kelsey Lubetich
Scripps College

Where is the Best Place to Sit on a Roller Coaster? Forces, Physics, and Fun at Disneyland

A Thesis Presented

by

Kelsey Lubetich

To the Keck Science Department

Of Claremont McKenna, Pitzer, and Scripps Colleges

In partial fulfillment of

The degree of Bachelor of Arts

Senior Thesis in Physics

April 23, 2012

Table of Contents

Abstract

The work presented in this thesis was undertaken to quantitatively determine the best place to sit on a roller coaster. Maximizing the time spent feeling weightless and the highest value of negative Z acceleration were used as criteria for the best seat. Acceleration values were measured on the California Screamin' roller coaster at Disney California Adventure Park using an iPhone and an application to record data from its accelerometers. After analyzing acceleration data, it was determined that the front row had the greatest negative acceleration in the z direction and was therefore the "best place" to sit.

Introduction

Most people who enjoy roller coasters have a favorite place to sit when riding, but no quantitative reasons for sitting there. When I ride roller coasters, especially roller coasters at Disneyland, my favorite sensation is weightlessness. Because I find it the most fun feeling, maximizing the time spent feeling weightless is the criteria for the best place to sit for this thesis. I was interested in looking at the time and acceleration values needed for free fall, so an accelerometer was used on a roller coaster to take measurements of acceleration. The answer to the question, "where is the best place to sit on a roller coaster?" will be the seat that has to greatest negative acceleration in the z direction and the most time spent in "free fall."

Materials and Methods

In order to measure the forces while on a roller coaster, an accelerometer is needed. Conveniently, today's iPhones come with an accelerometer already inside, and one only needs an application to access the raw data that is recoded by the phone's sensors. Many applications exist, but only two were used in this experiment. The first application used was Accelerometer Data Pro (ADP) by Wavefront Labs. ADP records and stores the X, Y, and Z values of the iPhone's accelerometer data relative to the phone's three axes[1], as shown in Figure 1. Values of acceleration are recorded in units of g. After initial trials, it was determined that ADP did not measure the sufficient values of acceleration needed for this experiment.

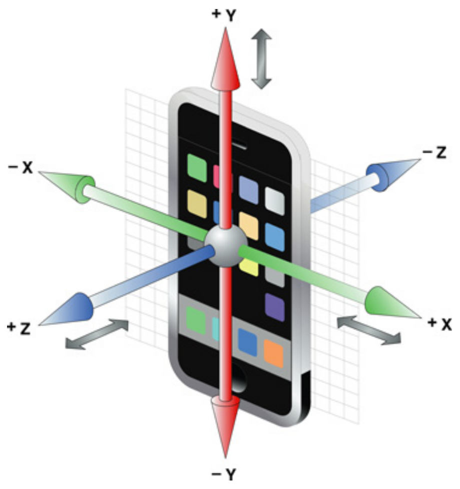


Figure 1: The X, Y, and Z-axes of an iPhone.[2] The top of the iPhone is in the +y direction while the front of the phone is in the +z direction.

The second application used in this experiment was Sensor Data (SD) by Wavefront Labs. SD was selected for use in all trials because it has the option of recording the User Acceleration, which represents the acceleration that the user is giving the iPhone, neglecting the constant effect of gravity. With SD, the X, Y, and Z values of the acceleration are all zero when the device is sitting still. An extended discussion of how to use SD to record and save data can be found in Appendix 1.

In this experiment, California Screamin' at Disney California Adventure Park was the only roller coaster used to measure forces. Screamin' has one inversion, so the coaster has seats with individual shoulder restraints. When sitting in the seat with the restraint on, the iPhone was held with the back, or –z direction, flat against the seat, and the +y direction facing the front of the coaster train. When the left seats were measured, the phone was held on the outside of the seat with my left hand between my leg and the outside wall, so that the phone was closest to the outside edge of the car. For the right seats, the phone was closest to the right edge of the car and was held with my right hand, as shown in Figure 2. It was not possible to hold the phone completely flush against the seat because of how it was being held, but it was kept as flat and parallel to the seat as possible while still maintaining a firm hold.

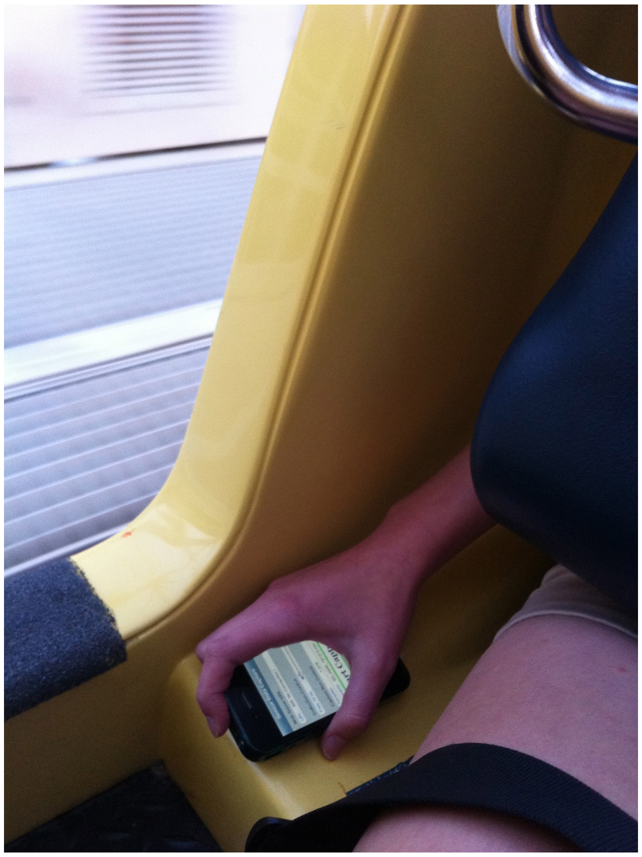


Figure 2: The iPhone being held while measuring the right seat of the car. The phone is held in the right hand between the rider and the outside edge of the car.

Ideally, the acceleration for every seat on Screamin' would have been measured and compared. A train on Screamin' has twelve rows for a totally of twenty-four seats. Given the time constraints of this thesis, there was not enough time to reasonably gather data for every seat of the train. To simplify comparison, I decided to only look at the front and back rows of the train. By looking at these two rows, I measured the acceleration for four different seats: row 1 left, row 1 right, row 12 left, and row 12 right.

After riders are seated and restrained on Screamin', the cars move to the launch portion of the track. In order to have all the trials begin at roughly the same time, the Start Capture button was pressed after the train had come to a stop at the launch station. Because the train was stopped, ideally all accelerations would begin at zero and it would be easy to see when the train suddenly started moving. Even so, trains occasionally pause for different time intervals before launch, so this was taken into consideration when the data was analyzed.

Before trials were run on Screamin', SD had to be calibrated to make sure it reported accurate values of acceleration. This was accomplished by dropping the iPhone straight down for each of its six directions. If SD worked correctly, the values reported for acceleration should have been positive when the phone was dropped with a positive axis facing down and negative with a negative axis facing down. The results of this calibration can be found in Table 1 and were found to be opposite of what was expected. This error was corrected by multiplying all data by a factor of -1 before analysis.

Axis Facing Down	+x	-x	+y	-y	+z	-z
Value of Acceleration (g)	-1	1	-1	1	-1	1

Table 1: SD calibration data.

Two different iPhones were used for acceleration measurement. One phone was used to measure the acceleration in the left seats while the other was used to measure the acceleration in the right seats. Both phones were calibrated and were found to have the same calibration error previously discussed. Because two phones were used for measurement, it was possible to measure both the left and right seat of a row at the same time. The left and right data for each trial of row 1 or row 12 occurred during the same ride on Screamin'. This was done to minimize outside factors such as differing masses of the train that could affect the speed and therefore acceleration of the ride. It was not possible to record all four seats for the same ride.

Results and Analysis

Before true data analysis could begin, the raw data was extensively manipulated to get it into a useful form. Because the data was recorded at a frequency of 10 Hz, there are around a thousand data points for each trial. When all data points were considered, the resulting graphs were very spiky with outlying data points. To remedy this, a rolling average of three points was used for all plots. The data was averaged using the Java code found in Appendix 2. Once the data was averaged, the X, Y, and Z acceleration data were all multiplied by -1 to fix the direction error found during calibration.

Because each trial started at slightly different times, the timestamps for all the trials didn't match up so the runs could not yet be compared. When Screamin' starts at the launch, the train is stationary until, all of a sudden, the train rapidly accelerates forward while the track remains flat. This launch was useful in determining when the actual ride began because it is easy to see on a graph of the data when the train suddenly accelerates forward (Figure 3). The time point just before the Y acceleration rapidly increases was taken to be the new zero time for each trial. The value of that time was subtracted from all future data points to adjust for the new time reference. With the adjusted time values, just after t=0 the rapid increase in Y acceleration can be seen (Figure 4). Because all trials share this rapid Y increase, the data could now be easily matched up with all trials beginning at t=0.

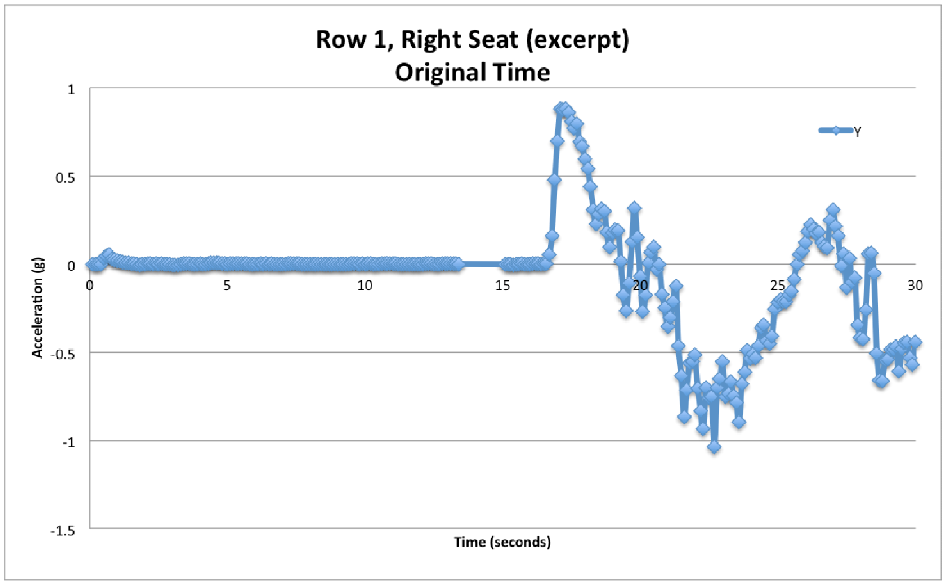


Figure 3: Graph of Y acceleration using the original time of row 1, right seat, trial 4.

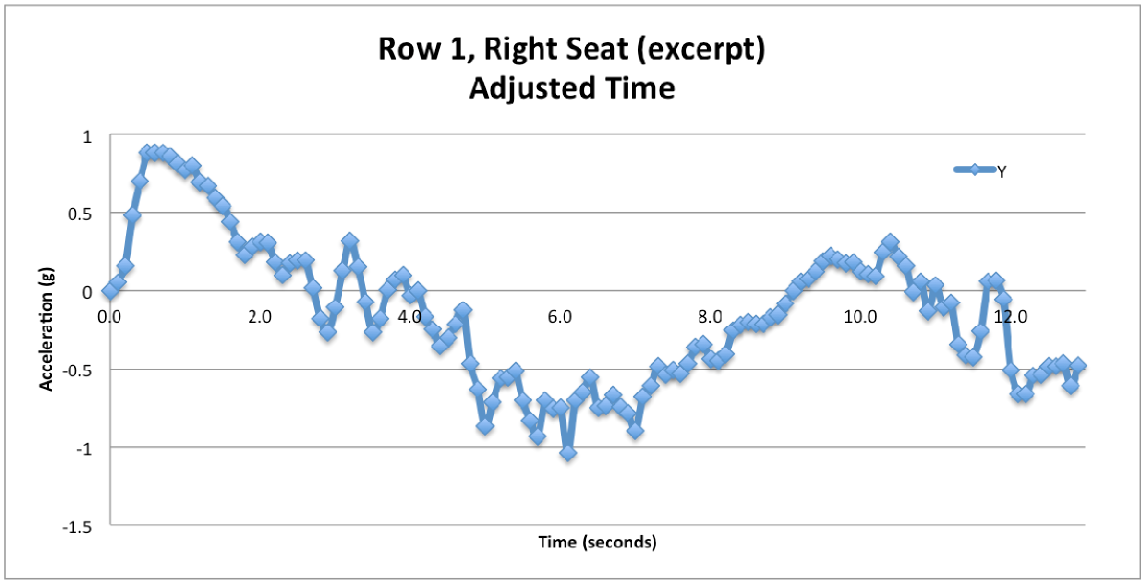


Figure 4: Graph of Y acceleration using the adjusted time of row 1, right seat, trial 4.

Once all the trials were time adjusted, there was still too much data to reasonably compare. To compensate for this, all the trials for each seat were averaged together to come up with one set of data for each of the four seats measured. Finally, the data was condensed into sets that could be graphed with four lines, one for each seat, for the X, Y, and Z accelerations. These graphs for the X, Y, and Z Acceleration can be found in Appendix 3. In all three graphs, row 1 left is the blue line, row 1 right is red, row 12 left is green, and row 12 right is purple.

I have defined the best seat as the seat with the greatest negative Z acceleration and the most time spent in free fall. There are many times during Screamin' where the rider feels like she is floating, but only a few of these moments occur when the track is also flat and would therefore only have acceleration in the z direction. To determine when the rider experiences free fall, a YouTube video[4] of the ride was compared to experiences on the coaster. Notable track events, the times at which they occurred, and the perceived additional forces accompanying them were recorded in Table 2.

There are four notable places the rider feels like she is floating. The first time free fall occurs, the track is angled at the top of the first hill, meaning some of the Z acceleration is split in the X and Y directions. This means that the tops of the bunny hills are the only places where the rider both floats and the track is flat.

Track Event	**Video Time**	**Ride Time**	**Perceived Extra Force**
Launch	0:21	0”	L ←
End of Flat Launch	0:26	5”	No extra force
Top of 1st Hill	0:29	8”	L ↑ (floating)
Bottom of Lift Hill	0:47	26”	L ↓
Top of 2nd Hill	0:58	37”	No extra force
Start of Loop	1:15	54”	No extra force
Top of Loop	1:17	56”	Γ ↑
Bottom of Loop	1:19	58”	L ↓
Bottom of Post-Loop Hill	1:27	66”	L ↓
Start of Bunny Hills	1:39	77”	No extra force
First Dip	1:41	80”	L ↓
Top of 1st Bunny Hill	1:42	81”	L ↑ (floating)
Second Dip	1:43	82”	L ↓
Top of 2nd Bunny Hill	1:44	83”	L ↑ (floating)
Third Dip	1:46	85”	L ↓
Top of 3rd Bunny Hill	1:47	86”	L ↑ (floating)
End of Ride	2:02	101”	L →

Table 2: For each track event on California Screamin’, the time it occurred in the video was recorded, then the actual time from the ride launch, and finally the direction of the additional force that the rider perceives. “L” represents the seat of the roller coaster when it is upright while “Γ” represents the seat when it is upside down, something that only occurs once during the loop of the coaster.

Because I am interested in the maximum time spent in free fall, from this point on I will only be looking at the Z acceleration. There are three bunny hills on Screamin’, and they occur roughly from 77” to 86”. As expected, around these times there are peaks and valleys on the graph of Z Acceleration that correspond with the hills (Figure 5). The valleys of Figure 13 correspond to the tops of the bunny hills while the peaks correspond to the dips.

The rider feels like she is floating when the train goes over the peaks on the bunny hills, so free fall occurs when there is negative Z acceleration.

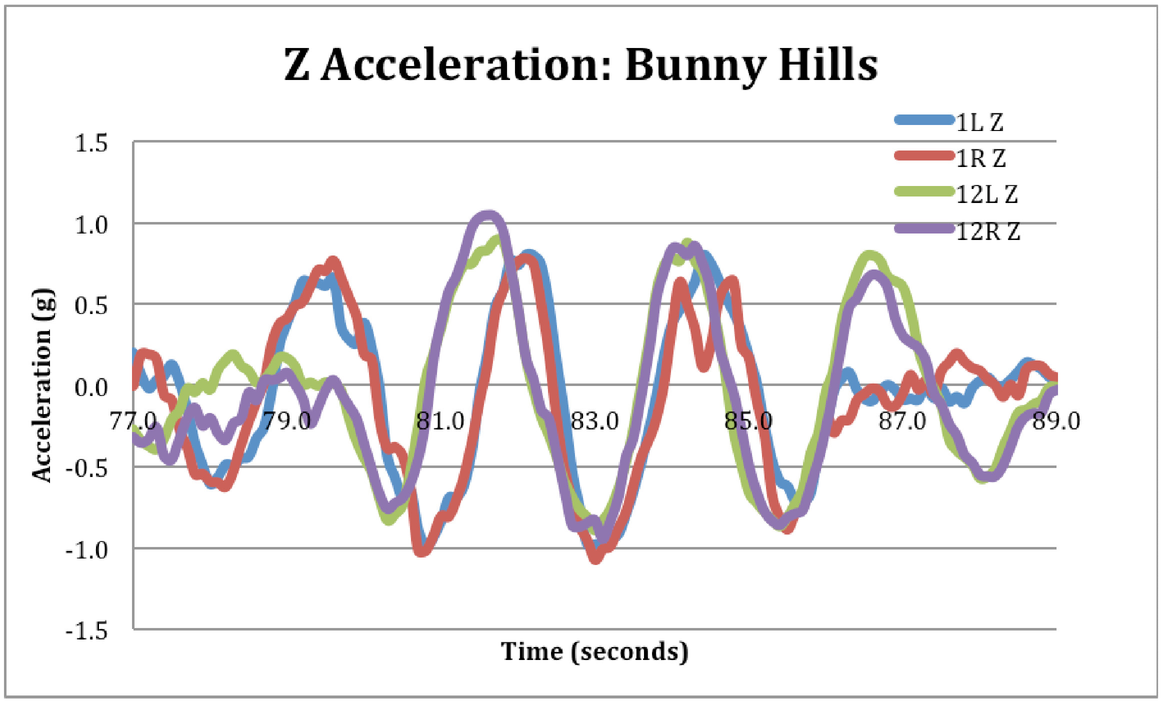


Figure 5: Graph of the Z Acceleration during the "bunny hills".

When looking at the graph of the bunny hills, each seat has three valleys that correspond to the tops of the three bunny hills, but the valleys occur at different times for row 1 and row 12. This is because the train is so long that the entire train cannot experience the bunny hills at the same time. When the front of the train reaches the second bunny hill, it's about the same time that the back of the train crests the first hill. The hills are roughly two seconds apart and it's about two seconds that the front and back seats are apart on the graph. For row 1, the bunny hilltops occur at ~81, ~83, and ~86 while for row 12, they occur at ~83, ~86, and ~88. See Appendix 5 for individual graphs of row 1 and 12 over the bunny hills. A rider feels like she is floating when there is a valley in the acceleration in the Z direction. The seat with the highest value of negative Z acceleration has been defined as the best place to sit on the roller coaster.

The highest values of negative Z acceleration were found for each of the three bunny hills. The complete data table can be found in Appendix 4, but the maximum values have been recorded in Table 3. For every hill, the front row right seat has the highest value of negative acceleration. The second hill has the greatest value of acceleration recorded with a = -1.07949g.

	Maximum Negative Z Acceleration (g)			
	1L Z	**1R Z**	**12L Z**	**12R Z**
First Hill	-0.99666	-1.02482	-0.89938	-0.95076
Second Hill	-0.98391	-1.07949	-0.87305	-0.86473
Third Hill	-0.72246	-0.89294	-0.57801	-0.56740

Table 3: The maximum negative Z accelerations at the top of all three bunny hills.

If the left or right position of the car is ignored, the values for each row can be averaged together, as is done in Table 4. After the accelerations are averaged for each row, row 1 has the highest negative values of acceleration over all three hills. The difference in the acceleration between row 1 and row 12 is 0.08567g for the first hill and increases linearly for the next two hills. The greatest difference in acceleration between the rows occurs on the third hill.

	Maximum Negative Z Acceleration (g)		
	Row 1	**Row 12**	**Difference**
First Hill	-1.01074	-0.92507	0.08567
Second Hill	-1.03170	-0.86889	0.16281
Third Hill	-0.80770	-0.57270	0.23500

Table 4: The average negative Z acceleration for each row over the bunny hills.

In addition to the maximum value of negative acceleration, the additional criterion of time spent in free fall is being used to determine the best seat. In order to measure the time spent in free fall, for the purposes of this experiment, free fall is defined as a value of acceleration that is less that -0.75g in the Z direction. With this definition, the time spent in free fall for each seat is shown in Table 5. The front row right seat spends the most time in free fall on nearly every hill, something that corresponds with its higher negative acceleration values.

	Time Spent in Free Fall (seconds)			
	1L Z	**1R Z**	**12L Z**	**12R Z**
First Hill	0.3	0.4	0.4	0.5
Second Hill	0.6	0.6	0.3	0.5
Third Hill	0.0	0.2	0.0	0.0

Table 5: Time spent in free fall for the three bunny hills. Free fall has been defined as a time with an acceleration less than -0.75g.

I defined the criteria for the best seat to be the seat with the greatest value of negative Z acceleration and the greatest time spent in free fall. Using this definition, the front row of California Screamin' is the best place to sit. However, the definition of free fall acceleration may be incorrect. During the ride, the rider feels like she is floating, if only briefly, during all the bunny hills in both the front and back rows of the train. According to my free fall definition, the rider does not spend time in free fall on every hill. In future research, perhaps a better definition of what constitutes free fall can be determined.

Conclusion

The best place to sit on a roller coaster is the front row because it has both the greatest values of negative Z acceleration and the greatest time spent in free fall. This theory has currently only been tested on the California Screamin' roller coaster, but the definition will likely be true for other coasters of a similar type. A future application of this research could be to test this theory on other roller coasters. The methods are fairly easy to replicate and the application used is under $10 if the user already has an iPhone. It is unknown if the result of the front row best seat would hold true for different types of coasters, such as coasters with more inversions, without inversions, or without bunny hills, but the same definition of the best seat can be used. The results of this experiment were surprising; in the past I have always enjoyed sitting in the back of roller coasters more than the front because I thought the back had the wilder ride. According to the acceleration data recorded during this experiment, that is not the case. I enjoy the feeling of weightlessness on roller coasters, so from now on I will be sitting in the front. If you enjoy weightlessness as well and only have one ride on a roller coaster, sit in the front for the best experience.

Acknowledgments

This thesis would not have been possible without the encouragement and support of Scot Gould and his approval of my crazy idea of a topic. I am also grateful to Thomas Dershem for his support. *Grazie mille,* dear readers. I am forever indebted to Shannon Lubetich for being my right-seat buddy and picture twin and enduring the arduous task of many trips to Disneyland. I would also like to thank the amazing commenters on my Touring Plans blog post for sharing their favorite spots on roller coasters, even though I did not end up sharing their opinions in my introduction.

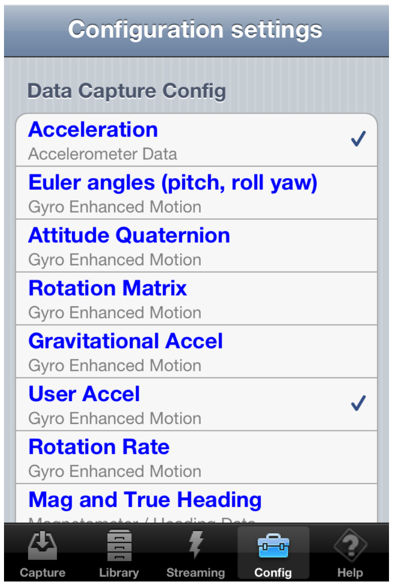
Works Cited

1. "Accelerometer Data." 2010. Wavefront Labs. <http://wavefrontlabs.com/Wavefront_Labs/Accelerometer_Data.html>.
2. "Event Handling Guide for iOS." IOS Developer Library. 3 Oct. 2011. Apple, Inc. <https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/EventHandlingiPhoneOS.pdf>.
3. "Sensor Data." 2010. Wavefront Labs. <http://wavefrontlabs.com/Wavefront_Labs/Sensor_Data.html>.
4. TheCoasterViews. "California Screamin' (HD POV) Disney's California Adventure." YouTube. 29 Nov. 2009. <http://www.youtube.com/watch?v=v_cHDsHuMbE>.

Appendix 1: Equipment and Software

Sensor Data (SD) by Wavefront Labs was used in this experiment (Figure 6). SD can record data from more of the iPhone's sensors than ADP, including the 3D accelerometer and 3D gyroscope.[3] In this experiment, the Acceleration, essentially the same data recorded by ADP, and User Acceleration were recorded. The data values that the user wants to record are simply clicked in the Configuration Settings screen (Figure 7). The User Acceleration represents the acceleration that the user is giving to the iPhone, while factoring out gravity. With SD, the X, Y, and Z values of the acceleration are all zero when the device is sitting still.

Figure 6: SD's title screen.



Configuration settings

Data Capture Config

Acceleration ✓
Accelerometer Data

Euler angles (pitch, roll yaw)
Gyro Enhanced Motion

Attitude Quaternion
Gyro Enhanced Motion

Rotation Matrix
Gyro Enhanced Motion

Gravitational Accel
Gyro Enhanced Motion

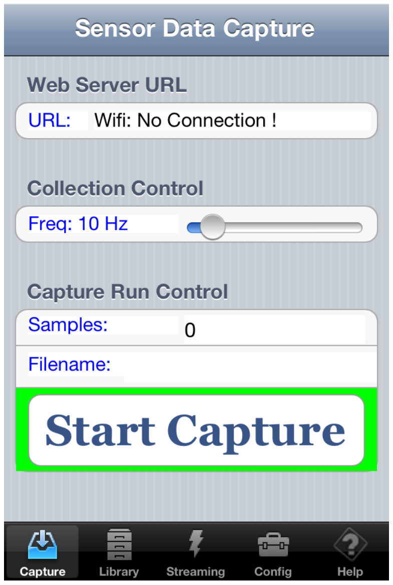
User Accel ✓
Gyro Enhanced Motion

Rotation Rate
Gyro Enhanced Motion

Mag and True Heading

Figure 7: Some of the data that can be recorded using SD. The checked options are the only ones that SD actually records and saves.

To record a data run, the appropriate configuration settings are selected, and then the Start Capture button is pressed (Figure 8). SD automatically selects a filename for the data run. The iPhone needs to remain in its "awake" setting with the screen active while SD is recording. While SD is running, the number of Samples increases and the Start Capture button changes to a Stop Capture button. When the run is over, the Stop Capture button is pressed and the data file is automatically saved to the SD Library (Figure 9).



Sensor Data Capture

Web Server URL

URL: Wifi: No Connection !

Collection Control

Freq: 10 Hz

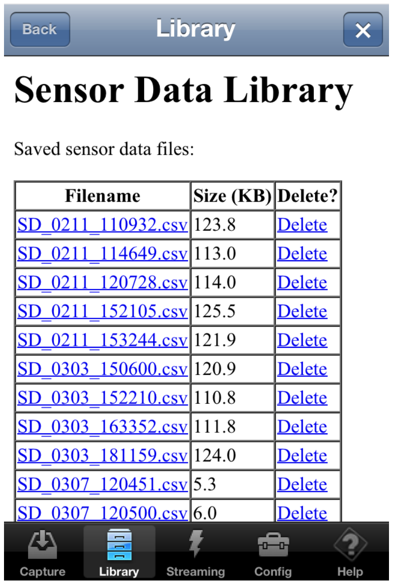
Capture Run Control

Samples: 0

Filename:

Start Capture

Figure 8: The Capture screen for SD. The frequency that values are recorded can be selected from 1 to 100 Hz.



Sensor Data Library

Saved sensor data files:

Filename	Size (KB)	Delete?
SD_0211_110932.csv	123.8	Delete
SD_0211_114649.csv	113.0	Delete
SD_0211_120728.csv	114.0	Delete
SD_0211_152105.csv	125.5	Delete
SD_0211_153244.csv	121.9	Delete
SD_0303_150600.csv	120.9	Delete
SD_0303_152210.csv	110.8	Delete
SD_0303_163352.csv	111.8	Delete
SD_0303_181159.csv	124.0	Delete
SD_0307_120451.csv	5.3	Delete
SD_0307_120500.csv	6.0	Delete

Figure 9: SD's Library. Files are named based on the date and time of recording. Files saved in SD can later be exported onto a computer.

Once collected, to export the data onto a computer, plug in the iPhone and open iTunes. After clicking on the iPhone under Devices, navigate to the Apps tab (Figure 10). Under File Sharing, once SD is selected, individual files can be clicked on and saved them to a location on your computer (Figure 11).

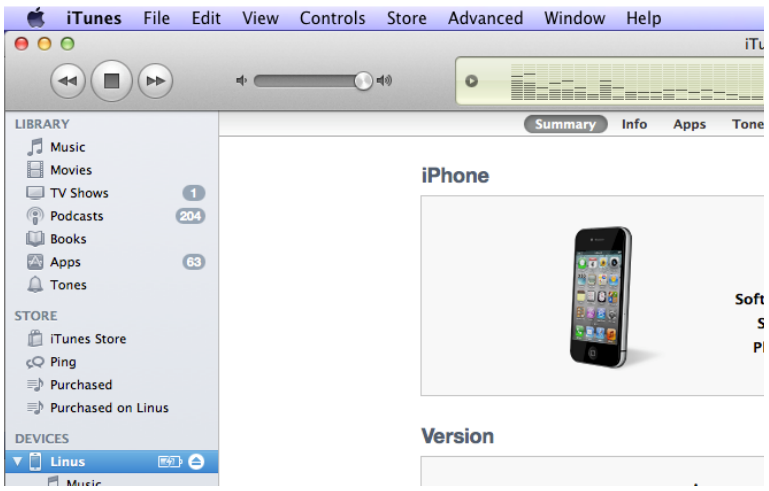
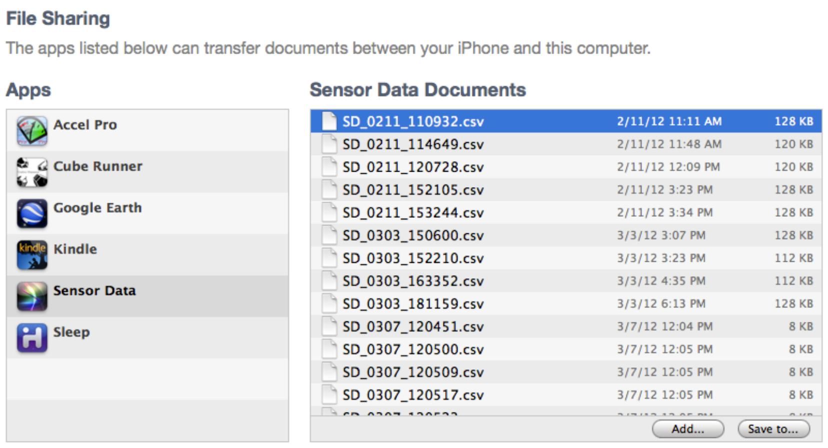


Figure 10: Click the iPhone under Devices and navigate to the Apps tab.



File Sharing

The apps listed below can transfer documents between your iPhone and this computer.

Sensor Data Documents

SD_0211_110932.csv	2/11/12 11:11 AM	128 KB
SD_0211_114649.csv	2/11/12 11:48 AM	120 KB
SD_0211_120728.csv	2/11/12 12:09 PM	120 KB
SD_0211_152105.csv	2/11/12 3:23 PM	128 KB
SD_0211_153244.csv	2/11/12 3:34 PM	128 KB
SD_0303_150600.csv	3/3/12 3:07 PM	128 KB
SD_0303_152210.csv	3/3/12 3:23 PM	112 KB
SD_0303_163352.csv	3/3/12 4:35 PM	112 KB
SD_0303_181159.csv	3/3/12 6:13 PM	128 KB
SD_0307_120451.csv	3/7/12 12:04 PM	8 KB
SD_0307_120500.csv	3/7/12 12:05 PM	8 KB
SD_0307_120509.csv	3/7/12 12:05 PM	8 KB
SD_0307_120517.csv	3/7/12 12:05 PM	8 KB

Figure 11: SD files can be saved to your computer.

Appendix 2: Java Code

AverageThree.java

```java
import java.io.*;
import java.util.ArrayList;

/**
 *      @author Kelsey Lubetich
 *      @date 2/20/12
 **/

/*
 * A class that takes in a file of numbers, and then averages them together in
 *              groups of three values.  The average values are returned in a new file.
 */
public class AverageThree {

        //two ArrayLists that keep track of the original list of values
        //      and the calculated list of averages
        private ArrayList<Double> valueList = new ArrayList<Double>();
        private ArrayList<Double> averagedList = new ArrayList<Double>();

        //the number of lines in the list of values,
        //      starts out at 0
        private int valueListCount = 0;

        //the number of lines in the averaged list of values,
        //      also starts out at 0
        private int averagedListCount = 0;

        /*
         * @param filename, the file of numbers to be averaged
         * the constructor opens the file, and reads in its contents.
         *      the numbers in the file are put into the valueList arraylist.
         */
        public AverageThree(String filename){
                //lets the user know that the file is being read in
                System.out.println("Reading file...");

                //used in file input
                FileReader reader;
                BufferedReader buffer;

                try {
                        //creates the objects used during file input
                        reader = new FileReader(filename);
                        buffer = new BufferedReader(reader);

                        //reads the first line of the file and starts the line count
                        //      at 0
                        String line = buffer.readLine();
                        valueListCount = 0;

                        //keeps track of the number read in off the file
                        double number;

                        //the program keeps reading in lines until its value is null
                        while(line != null){
                                //gets the double from the line and adds the number to the
                                //      arraylist
                                number = Double.parseDouble(line);
                                valueList.add(number);

                                //reads in the next line of the file and updates the line //
                                        count
                                line = buffer.readLine();
                                valueListCount++;
                }

                //closes the file input objects
                reader.close();
                buffer.close();

        } catch (FileNotFoundException e) {
                System.out.println("Error: File not found.");
        } catch (IOException e) {
                System.out.println("Error: I/O Excpetion");
        }
}


/*
 * takes the valueList and averages it together over sets of three
 *      values.  the values are the put into the averagedList arraylist.
 */
public void average(){
        //lets the user know the file is now being averaged
        System.out.println("Averaging file...");

        //all the values used to calculate the average start at 0
        averagedListCount = 0;
        double val1 = 0, val2 = 0, val3 = 0;
        double average = 0;

        //gets the first three values in valueList
        val1 = valueList.get(averagedListCount);
        val2 = valueList.get(averagedListCount + 1);
        val3 = valueList.get(averagedListCount + 2);

        //values keep getting averaged until there are not enough values
        //      to average together at the end, or when there are less than
        //      three values left in the arraylist
        while(averagedListCount+3 < valueListCount){
                //the average is the sum of all three values divided by 3.  then
                //      the average is added to the averagedList arraylist and its
                //      count is increased.
                average = (val1 + val2 + val3) / 3;
                averagedList.add(average);
                averagedListCount++;

                //then the next three values are taken from valueList.  each time,
                //      two of the previous values are averaged along with one new
                //      one.
                val1 = valueList.get(averagedListCount);
                val2 = valueList.get(averagedListCount + 1);
                val3 = valueList.get(averagedListCount + 2);
        }

        //the averagedList can optionally be printed out to the console
        //System.out.println(averagedList);
}


/*
 * @param outoutFilename, the file where the averaged values will be
 *      printed
 * prints the averagedList out into the specified file line by line
 */
public void output(String outputFilename){
        //lets the user know the file is printing
        System.out.println("Printing file...");

        //used in file output
        FileWriter file;
        PrintWriter writer;

        //the number of lines that have been printed out
        int writingCount = 0;
```

```
		try {
			//creates the objects used during file output
			file = new FileWriter(outputFilename);
			writer= new PrintWriter(file);

			//the program keeps printing lines as long as there are still
			//	values in averagedList that have not been printed yet
			while(writingCount < averagedListCount){
				//prints the value in averagedList at the specified count,
				//	then updates the count
				writer.println(averagedList.get(writingCount));
				writingCount++;
			}

			//closes the file output objects
			file.close();
			writer.close();
		} catch (IOException e) {
			System.out.println("Error: I/O Exception");
		}

		//lets the user know that the program has finished
		System.out.println("Done!");
		System.out.println();
	}
}
```

Row1Left.java

```
/*
 * runs AverageThree for all the row 1, left seat data
 */
public class Row1Left {

	public static void main(String[] args) {
		//Row 1, Left seat
		//trial 1
		/*
		 * "row1LT1time.txt"
		 * "row1LT1X.txt"
		 * "row1LT1Y.txt"
		 * "row1LT1Z.txt"
		 *
		 * "row1LT1timeAverage.txt"
		 * "row1LT1XAverage.txt"
		 * "row1LT1YAverage.txt"
		 * "row1LT1ZAverage.txt"
		 */
		AverageThree av3 = new AverageThree("row1LT1time.txt");
		av3.average();
		av3.output("row1LT1timeAverage.txt");

		av3 = new AverageThree("row1LT1X.txt");
		av3.average();
		av3.output("row1LT1XAverage.txt");

		av3 = new AverageThree("row1LT1Y.txt");
		av3.average();
		av3.output("row1LT1YAverage.txt");

		av3 = new AverageThree("row1LT1Z.txt");
		av3.average();
		av3.output("row1LT1ZAverage.txt");

		//trial 2
		/*
 * "row1LT2time.txt"
 * "row1LT2X.txt"
 * "row1LT2Y.txt"
 * "row1LT2Z.txt"
 *
 * "row1LT2timeAverage.txt"
 * "row1LT2XAverage.txt"
 * "row1LT2YAverage.txt"
 * "row1LT2ZAverage.txt"
 */
av3 = new AverageThree("row1LT2time.txt");
av3.average();
av3.output("row1LT2timeAverage.txt");

av3 = new AverageThree("row1LT2X.txt");
av3.average();
av3.output("row1LT2XAverage.txt");

av3 = new AverageThree("row1LT2Y.txt");
av3.average();
av3.output("row1LT2YAverage.txt");

av3 = new AverageThree("row1LT2Z.txt");
av3.average();
av3.output("row1LT2ZAverage.txt");

//trial 3
/*
 * "row1LT3time.txt"
 * "row1LT3X.txt"
 * "row1LT3Y.txt"
 * "row1LT3Z.txt"
 *
 * "row1LT3timeAverage.txt"
 * "row1LT3XAverage.txt"
 * "row1LT3YAverage.txt"
 * "row1LT3ZAverage.txt"
 */
av3 = new AverageThree("row1LT3time.txt");
av3.average();
av3.output("row1LT3timeAverage.txt");

av3 = new AverageThree("row1LT3X.txt");
av3.average();
av3.output("row1LT3XAverage.txt");

av3 = new AverageThree("row1LT3Y.txt");
av3.average();
av3.output("row1LT3YAverage.txt");

av3 = new AverageThree("row1LT3Z.txt");
av3.average();
av3.output("row1LT3ZAverage.txt");

//trial 4
/*
 * "row1LT4time.txt"
 * "row1LT4X.txt"
 * "row1LT4Y.txt"
 * "row1LT4Z.txt"
 *
 * "row1LT4timeAverage.txt"
 * "row1LT4XAverage.txt"
 * "row1LT4YAverage.txt"
 * "row1LT4ZAverage.txt"
 */
av3 = new AverageThree("row1LT4time.txt");
av3.average();
av3.output("row1LT4timeAverage.txt");

av3 = new AverageThree("row1LT4X.txt");
```

```
                av3.average();
                av3.output("row1LT4XAverage.txt");

                av3 = new AverageThree("row1LT4Y.txt");
                av3.average();
                av3.output("row1LT4YAverage.txt");

                av3 = new AverageThree("row1LT4Z.txt");
                av3.average();
                av3.output("row1LT4ZAverage.txt");

                //trial 5
                /*
                 * "row1LT5time.txt"
                 * "row1LT5X.txt"
                 * "row1LT5Y.txt"
                 * "row1LT5Z.txt"
                 *
                 * "row1LT5timeAverage.txt"
                 * "row1LT5XAverage.txt"
                 * "row1LT5YAverage.txt"
                 * "row1LT5ZAverage.txt"
                 */
                av3 = new AverageThree("row1LT5time.txt");
                av3.average();
                av3.output("row1LT5timeAverage.txt");

                av3 = new AverageThree("row1LT5X.txt");
                av3.average();
                av3.output("row1LT5XAverage.txt");

                av3 = new AverageThree("row1LT5Y.txt");
                av3.average();
                av3.output("row1LT5YAverage.txt");

                av3 = new AverageThree("row1LT5Z.txt");
                av3.average();
                av3.output("row1LT5ZAverage.txt");

                //trial 6
                /*
                 * "row1LT6time.txt"
                 * "row1LT6X.txt"
                 * "row1LT6Y.txt"
                 * "row1LT6Z.txt"
                 *
                 * "row1LT6timeAverage.txt"
                 * "row1LT6XAverage.txt"
                 * "row1LT6YAverage.txt"
                 * "row1LT6ZAverage.txt"
                 */
                av3 = new AverageThree("row1LT6time.txt");
                av3.average();
                av3.output("row1LT6timeAverage.txt");

                av3 = new AverageThree("row1LT6X.txt");
                av3.average();
                av3.output("row1LT6XAverage.txt");

                av3 = new AverageThree("row1LT6Y.txt");
                av3.average();
                av3.output("row1LT6YAverage.txt");

                av3 = new AverageThree("row1LT6Z.txt");
                av3.average();
                av3.output("row1LT6ZAverage.txt");

        }
}
```

Row1Right.java

```java
/*
 * runs AverageThree for all the row 1, right seat data
 */
public class Row1Right {

	public static void main(String[] args) {
		//Row 1, Right seat
		//trial 1
		/*
		 * "row1RT1time.txt"
		 * "row1RT1X.txt"
		 * "row1RT1Y.txt"
		 * "row1RT1Z.txt"
		 *
		 * "row1RT1timeAverage.txt"
		 * "row1RT1XAverage.txt"
		 * "row1RT1YAverage.txt"
		 * "row1RT1ZAverage.txt"
		 */
		AverageThree av3 = new AverageThree("row1RT1time.txt");
		av3.average();
		av3.output("row1RT1timeAverage.txt");

		av3 = new AverageThree("row1RT1X.txt");
		av3.average();
		av3.output("row1RT1XAverage.txt");

		av3 = new AverageThree("row1RT1Y.txt");
		av3.average();
		av3.output("row1RT1YAverage.txt");

		av3 = new AverageThree("row1RT1Z.txt");
		av3.average();
		av3.output("row1RT1ZAverage.txt");

		//trial 2
		/*
		 * "row1RT2time.txt"
		 * "row1RT2X.txt"
		 * "row1RT2Y.txt"
		 * "row1RT2Z.txt"
		 *
		 * "row1RT2timeAverage.txt"
		 * "row1RT2XAverage.txt"
		 * "row1RT2YAverage.txt"
		 * "row1RT2ZAverage.txt"
		 */
		av3 = new AverageThree("row1RT2time.txt");
		av3.average();
		av3.output("row1RT2timeAverage.txt");

		av3 = new AverageThree("row1RT2X.txt");
		av3.average();
		av3.output("row1RT2XAverage.txt");

		av3 = new AverageThree("row1RT2Y.txt");
		av3.average();
		av3.output("row1RT2YAverage.txt");

		av3 = new AverageThree("row1RT2Z.txt");
		av3.average();
		av3.output("row1RT2ZAverage.txt");

		//trial 3
		/*
		 * "row1RT3time.txt"
 * "row1RT3X.txt"
 * "row1RT3Y.txt"
 * "row1RT3Z.txt"
 *
 * "row1RT3timeAverage.txt"
 * "row1RT3XAverage.txt"
 * "row1RT3YAverage.txt"
 * "row1RT3ZAverage.txt"
 */
av3 = new AverageThree("row1RT3time.txt");
av3.average();
av3.output("row1RT3timeAverage.txt");

av3 = new AverageThree("row1RT3X.txt");
av3.average();
av3.output("row1RT3XAverage.txt");

av3 = new AverageThree("row1RT3Y.txt");
av3.average();
av3.output("row1RT3YAverage.txt");

av3 = new AverageThree("row1RT3Z.txt");
av3.average();
av3.output("row1RT3ZAverage.txt");

//trial 4
/*
 * "row1RT4time.txt"
 * "row1RT4X.txt"
 * "row1RT4Y.txt"
 * "row1RT4Z.txt"
 *
 * "row1RT4timeAverage.txt"
 * "row1RT4XAverage.txt"
 * "row1RT4YAverage.txt"
 * "row1RT4ZAverage.txt"
 */
av3 = new AverageThree("row1RT4time.txt");
av3.average();
av3.output("row1RT4timeAverage.txt");

av3 = new AverageThree("row1RT4X.txt");
av3.average();
av3.output("row1RT4XAverage.txt");

av3 = new AverageThree("row1RT4Y.txt");
av3.average();
av3.output("row1RT4YAverage.txt");

av3 = new AverageThree("row1RT4Z.txt");
av3.average();
av3.output("row1RT4ZAverage.txt");


//trial 5
/*
 * "row1RT5time.txt"
 * "row1RT5X.txt"
 * "row1RT5Y.txt"
 * "row1RT5Z.txt"
 *
 * "row1RT5timeAverage.txt"
 * "row1RT5XAverage.txt"
 * "row1RT5YAverage.txt"
 * "row1RT5ZAverage.txt"
 */
av3 = new AverageThree("row1RT5time.txt");
av3.average();
av3.output("row1RT5timeAverage.txt");

av3 = new AverageThree("row1RT5X.txt");
```

```
		av3.average();
		av3.output("row1RT5XAverage.txt");

		av3 = new AverageThree("row1RT5Y.txt");
		av3.average();
		av3.output("row1RT5YAverage.txt");

		av3 = new AverageThree("row1RT5Z.txt");
		av3.average();
		av3.output("row1RT5ZAverage.txt");

		//trial 6
		/*
		 * "row1RT6time.txt"
		 * "row1RT6X.txt"
		 * "row1RT6Y.txt"
		 * "row1RT6Z.txt"
		 *
		 * "row1RT6timeAverage.txt"
		 * "row1RT6XAverage.txt"
		 * "row1RT6YAverage.txt"
		 * "row1RT6ZAverage.txt"
		 */
		av3 = new AverageThree("row1RT6time.txt");
		av3.average();
		av3.output("row1RT6timeAverage.txt");

		av3 = new AverageThree("row1RT6X.txt");
		av3.average();
		av3.output("row1RT6XAverage.txt");

		av3 = new AverageThree("row1RT6Y.txt");
		av3.average();
		av3.output("row1RT6YAverage.txt");

		av3 = new AverageThree("row1RT6Z.txt");
		av3.average();
		av3.output("row1RT6ZAverage.txt");

	}
}
```

Row12Left.java

```
/*
 * runs AverageThree for all the row 12, left seat data
 */
public class Row12Left {

	public static void main(String[] args) {
		//Row 12, Left seat
		//trial 1
		/*
		 * "row12LT1time.txt"
		 * "row12LT1X.txt"
		 * "row12LT1Y.txt"
		 * "row12LT1Z.txt"
		 *
		 * "row12LT1timeAverage.txt"
		 * "row12LT1XAverage.txt"
		 * "row12LT1YAverage.txt"
		 * "row12LT1ZAverage.txt"
		 */
		AverageThree av3 = new AverageThree("row12LT1time.txt");
		av3.average();
		av3.output("row12LT1timeAverage.txt");
```

```
av3 = new AverageThree("row12LT1X.txt");
av3.average();
av3.output("row12LT1XAverage.txt");

av3 = new AverageThree("row12LT1Y.txt");
av3.average();
av3.output("row12LT1YAverage.txt");

av3 = new AverageThree("row12LT1Z.txt");
av3.average();
av3.output("row12LT1ZAverage.txt");

//trial 2
/*
 * "row12LT2time.txt"
 * "row12LT2X.txt"
 * "row12LT2Y.txt"
 * "row12LT2Z.txt"
 *
 * "row12LT2timeAverage.txt"
 * "row12LT2XAverage.txt"
 * "row12LT2YAverage.txt"
 * "row12LT2ZAverage.txt"
 */
av3 = new AverageThree("row12LT2time.txt");
av3.average();
av3.output("row12LT2timeAverage.txt");

av3 = new AverageThree("row12LT2X.txt");
av3.average();
av3.output("row12LT2XAverage.txt");

av3 = new AverageThree("row12LT2Y.txt");
av3.average();
av3.output("row12LT2YAverage.txt");

av3 = new AverageThree("row12LT2Z.txt");
av3.average();
av3.output("row12LT2ZAverage.txt");

//trial 3
/*
 * "row12LT3time.txt"
 * "row12LT3X.txt"
 * "row12LT3Y.txt"
 * "row12LT3Z.txt"
 *
 * "row12LT3timeAverage.txt"
 * "row12LT3XAverage.txt"
 * "row12LT3YAverage.txt"
 * "row12LT3ZAverage.txt"
 */
av3 = new AverageThree("row12LT3time.txt");
av3.average();
av3.output("row12LT3timeAverage.txt");

av3 = new AverageThree("row12LT3X.txt");
av3.average();
av3.output("row12LT3XAverage.txt");

av3 = new AverageThree("row12LT3Y.txt");
av3.average();
av3.output("row12LT3YAverage.txt");

av3 = new AverageThree("row12LT3Z.txt");
av3.average();
av3.output("row12LT3ZAverage.txt");

//trial 4
/*
 * "row12LT4time.txt"
 * "row12LT4X.txt"
 * "row12LT4Y.txt"
 * "row12LT4Z.txt"
 *
 * "row12LT4timeAverage.txt"
 * "row12LT4XAverage.txt"
 * "row12LT4YAverage.txt"
 * "row12LT4ZAverage.txt"
 */
av3 = new AverageThree("row12LT4time.txt");
av3.average();
av3.output("row12LT4timeAverage.txt");

av3 = new AverageThree("row12LT4X.txt");
av3.average();
av3.output("row12LT4XAverage.txt");

av3 = new AverageThree("row12LT4Y.txt");
av3.average();
av3.output("row12LT4YAverage.txt");

av3 = new AverageThree("row12LT4Z.txt");
av3.average();
av3.output("row12LT4ZAverage.txt");

//trial 5
/*
 * "row12LT5time.txt"
 * "row12LT5X.txt"
 * "row12LT5Y.txt"
 * "row12LT5Z.txt"
 *
 * "row12LT5timeAverage.txt"
 * "row12LT5XAverage.txt"
 * "row12LT5YAverage.txt"
 * "row12LT5ZAverage.txt"
 */
av3 = new AverageThree("row12LT5time.txt");
av3.average();
av3.output("row12LT5timeAverage.txt");

av3 = new AverageThree("row12LT5X.txt");
av3.average();
av3.output("row12LT5XAverage.txt");

av3 = new AverageThree("row12LT5Y.txt");
av3.average();
av3.output("row12LT5YAverage.txt");

av3 = new AverageThree("row12LT5Z.txt");
av3.average();
av3.output("row12LT5ZAverage.txt");

//trial 6
/*
 * "row12LT6time.txt"
 * "row12LT6X.txt"
 * "row12LT6Y.txt"
 * "row12LT6Z.txt"
 *
 * "row12LT6timeAverage.txt"
 * "row12LT6XAverage.txt"
 * "row12LT6YAverage.txt"
 * "row12LT6ZAverage.txt"
 */
av3 = new AverageThree("row12LT6time.txt");
av3.average();
av3.output("row12LT6timeAverage.txt");

av3 = new AverageThree("row12LT6X.txt");
av3.average();
```

```
                av3.output("row12LT6XAverage.txt");

                av3 = new AverageThree("row12LT6Y.txt");
                av3.average();
                av3.output("row12LT6YAverage.txt");

                av3 = new AverageThree("row12LT6Z.txt");
                av3.average();
                av3.output("row12LT6ZAverage.txt");

        }
}
```

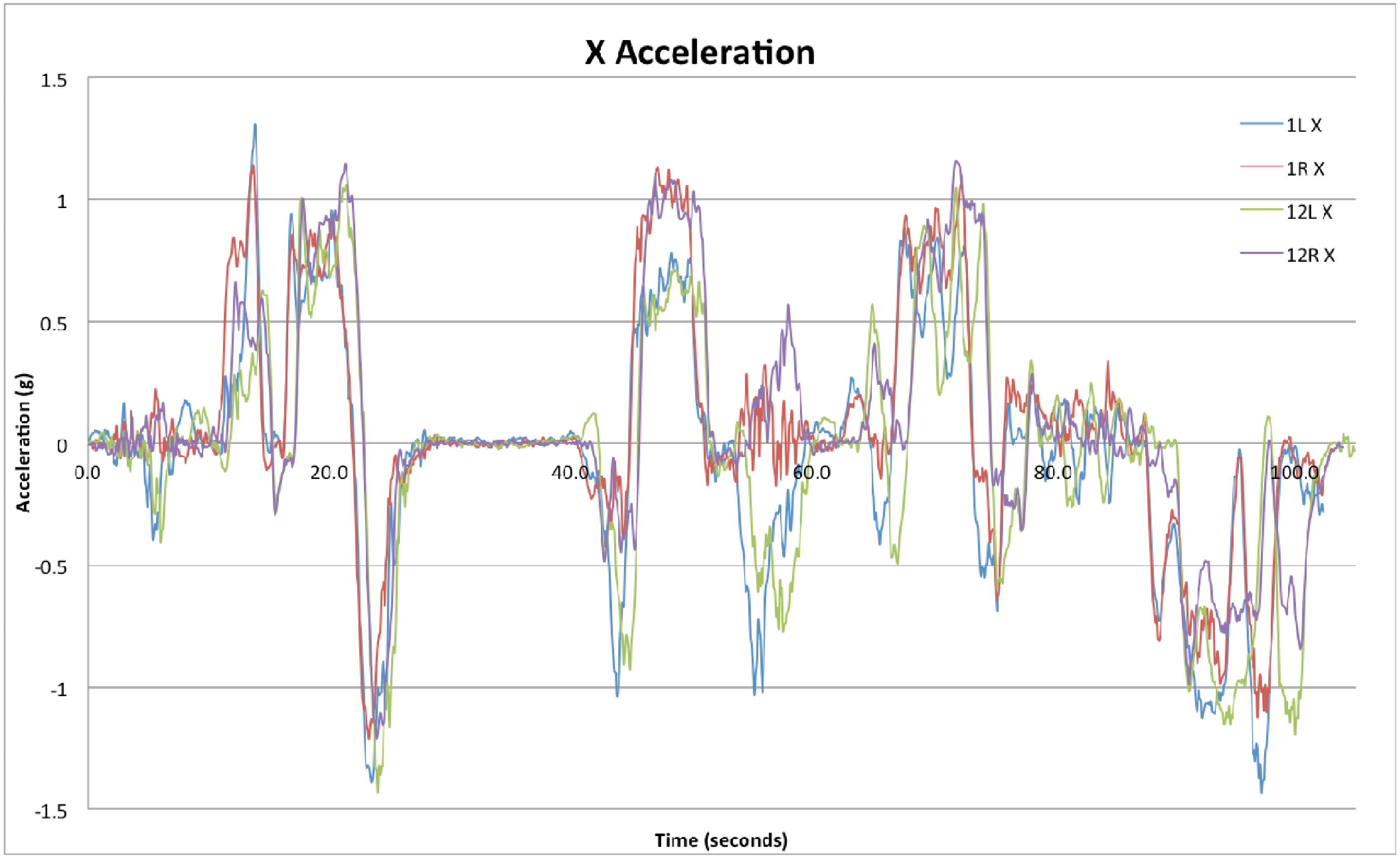
Row12Right.java

```
/*
 * runs AverageThree for all the row 12, right seat data
 */
public class Row12Right {

        public static void main(String[] args) {
                //Row 12, Right seat
                //trial 1
                /*
                 * "row12RT1time.txt"
                 * "row12RT1X.txt"
                 * "row12RT1Y.txt"
                 * "row12RT1Z.txt"
                 *
                 * "row12RT1timeAverage.txt"
                 * "row12RT1XAverage.txt"
                 * "row12RT1YAverage.txt"
                 * "row12RT1ZAverage.txt"
                 */
                AverageThree av3 = new AverageThree("row12RT1time.txt");
                av3.average();
                av3.output("row12RT1timeAverage.txt");

                av3 = new AverageThree("row12RT1X.txt");
                av3.average();
                av3.output("row12RT1XAverage.txt");

                av3 = new AverageThree("row12RT1Y.txt");
                av3.average();
                av3.output("row12RT1YAverage.txt");

                av3 = new AverageThree("row12RT1Z.txt");
                av3.average();
                av3.output("row12RT1ZAverage.txt");

                //trial 2
                /*
                 * "row12RT2time.txt"
                 * "row12RT2X.txt"
                 * "row12RT2Y.txt"
                 * "row12RT2Z.txt"
                 *
                 * "row12RT2timeAverage.txt"
                 * "row12RT2XAverage.txt"
                 * "row12RT2YAverage.txt"
                 * "row12RT2ZAverage.txt"
                 */
                av3 = new AverageThree("row12RT2time.txt");
                av3.average();
                av3.output("row12RT2timeAverage.txt");

                av3 = new AverageThree("row12RT2X.txt");
```

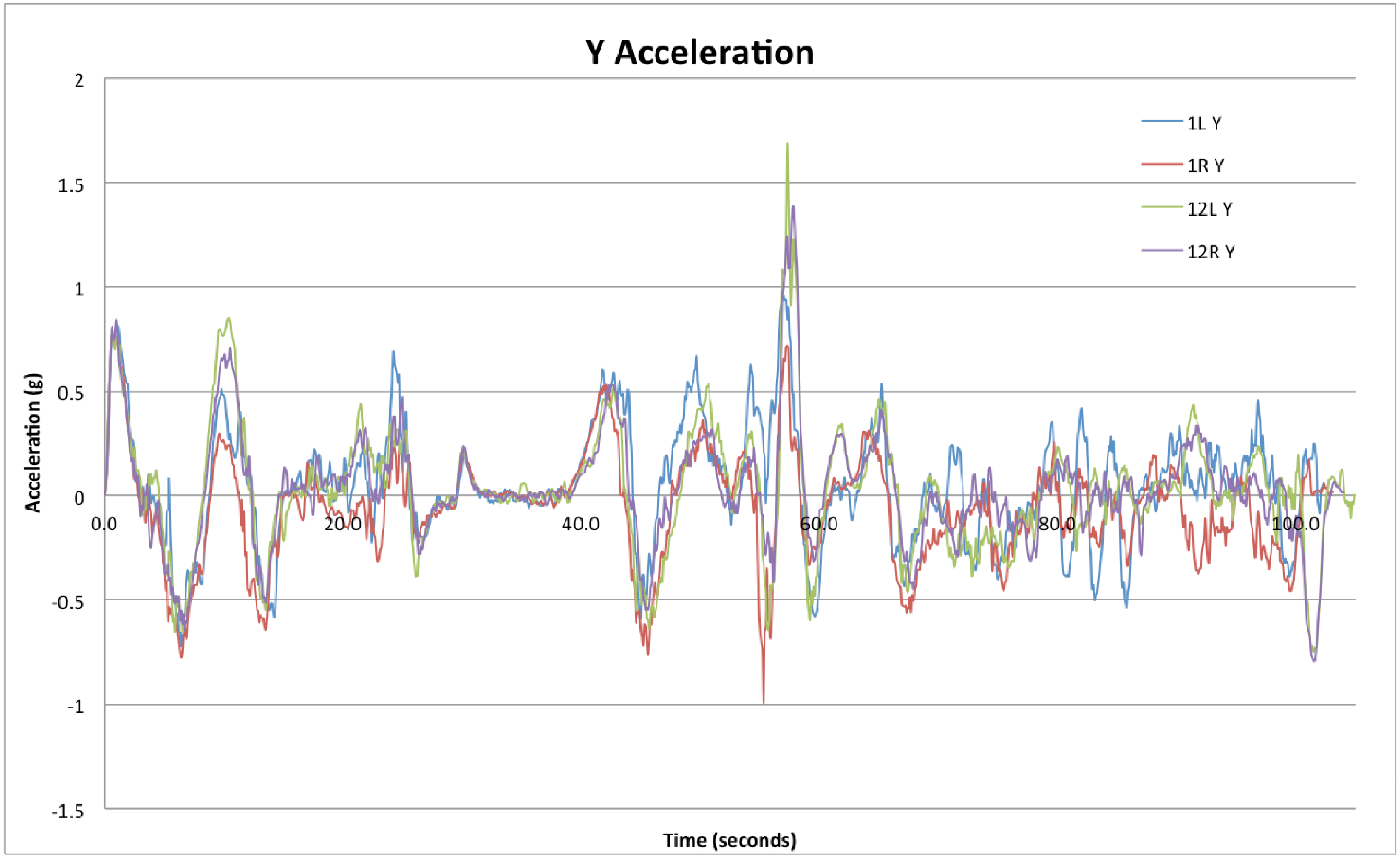
```
av3.average();
av3.output("row12RT2XAverage.txt");

av3 = new AverageThree("row12RT2Y.txt");
av3.average();
av3.output("row12RT2YAverage.txt");

av3 = new AverageThree("row12RT2Z.txt");
av3.average();
av3.output("row12RT2ZAverage.txt");

//trial 3
/*
 * "row12RT3time.txt"
 * "row12RT3X.txt"
 * "row12RT3Y.txt"
 * "row12RT3Z.txt"
 *
 * "row12RT3timeAverage.txt"
 * "row12RT3XAverage.txt"
 * "row12RT3YAverage.txt"
 * "row12RT3ZAverage.txt"
 */
av3 = new AverageThree("row12RT3time.txt");
av3.average();
av3.output("row12RT3timeAverage.txt");

av3 = new AverageThree("row12RT3X.txt");
av3.average();
av3.output("row12RT3XAverage.txt");

av3 = new AverageThree("row12RT3Y.txt");
av3.average();
av3.output("row12RT3YAverage.txt");

av3 = new AverageThree("row12RT3Z.txt");
av3.average();
av3.output("row12RT3ZAverage.txt");

//trial 4
/*
 * "row12RT4time.txt"
 * "row12RT4X.txt"
 * "row12RT4Y.txt"
 * "row12RT4Z.txt"
 *
 * "row12RT4timeAverage.txt"
 * "row12RT4XAverage.txt"
 * "row12RT4YAverage.txt"
 * "row12RT4ZAverage.txt"
 */
av3 = new AverageThree("row12RT4time.txt");
av3.average();
av3.output("row12RT4timeAverage.txt");

av3 = new AverageThree("row12RT4X.txt");
av3.average();
av3.output("row12RT4XAverage.txt");

av3 = new AverageThree("row12RT4Y.txt");
av3.average();
av3.output("row12RT4YAverage.txt");

av3 = new AverageThree("row12RT4Z.txt");
av3.average();
av3.output("row12RT4ZAverage.txt");

//trial 5
/*
 * "row12RT5time.txt"
 * "row12RT5X.txt"
```

```
                 * "row12RT5Y.txt"
                 * "row12RT5Z.txt"
                 *
                 * "row12RT5timeAverage.txt"
                 * "row12RT5XAverage.txt"
                 * "row12RT5YAverage.txt"
                 * "row12RT5ZAverage.txt"
                 */
                av3 = new AverageThree("row12RT5time.txt");
                av3.average();
                av3.output("row12RT5timeAverage.txt");

                av3 = new AverageThree("row12RT5X.txt");
                av3.average();
                av3.output("row12RT5XAverage.txt");

                av3 = new AverageThree("row12RT5Y.txt");
                av3.average();
                av3.output("row12RT5YAverage.txt");

                av3 = new AverageThree("row12RT5Z.txt");
                av3.average();
                av3.output("row12RT5ZAverage.txt");

                //trial 6
                /*
                 * "row12RT6time.txt"
                 * "row12RT6X.txt"
                 * "row12RT6Y.txt"
                 * "row12RT6Z.txt"
                 *
                 * "row12RT6timeAverage.txt"
                 * "row12RT6XAverage.txt"
                 * "row12RT6YAverage.txt"
                 * "row12RT6ZAverage.txt"
                 */
                av3 = new AverageThree("row12RT6time.txt");
                av3.average();
                av3.output("row12RT6timeAverage.txt");

                av3 = new AverageThree("row12RT6X.txt");
                av3.average();
                av3.output("row12RT6XAverage.txt");

                av3 = new AverageThree("row12RT6Y.txt");
                av3.average();
                av3.output("row12RT6YAverage.txt");

                av3 = new AverageThree("row12RT6Z.txt");
                av3.average();
                av3.output("row12RT6ZAverage.txt");

        }
}
```

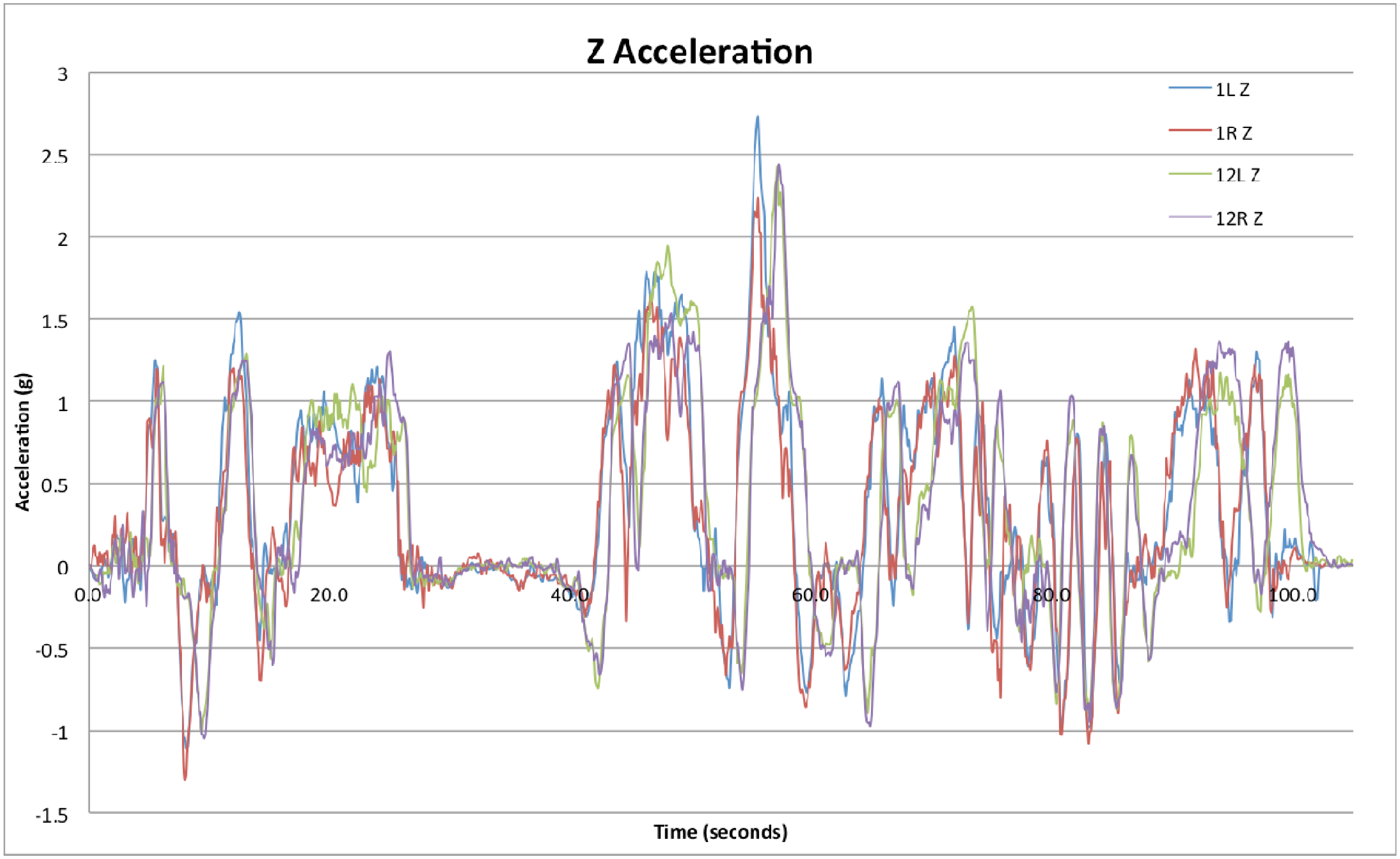
Appendix 3: Acceleration Graphs



Graph of the X Acceleration for all four seats.



Graph of the Y Acceleration for all four seats.



Graph of the Z Acceleration for all four seats.

Appendix 4: Z Acceleration Bunny Hills Data Table

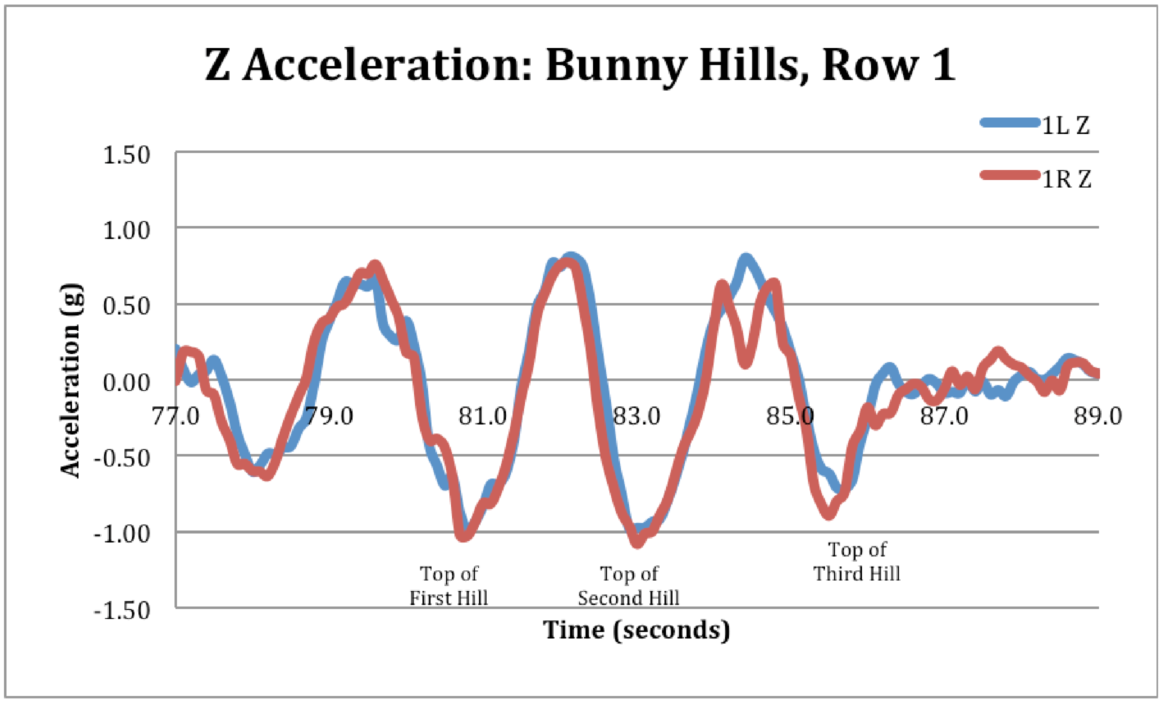
This data table is an excerpt from the Z acceleration data. It shows the values of the Z Acceleration for all four seats over the bunny hills. The highest negative values for Z acceleration have been highlighted. The top of the first hill is highlighted in yellow, the top of the second is green, and the top of the third hill is pink. Acceleration values that qualify as being in free fall, that is, are less than -0.75g, are shown in red text.

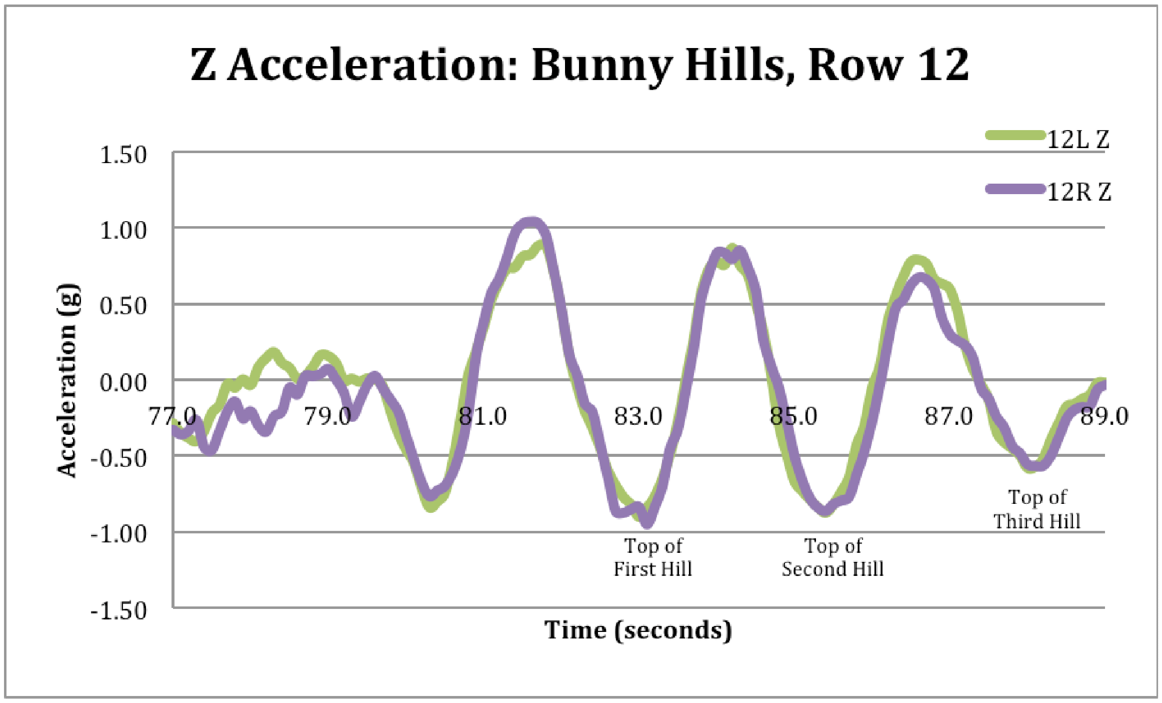
Time	1L Z	1R Z	12L Z	12R Z
77.0	0.19578	-0.00988	-0.27982	-0.32698
77.1	0.06565	0.18306	-0.34695	-0.35990
77.2	-0.01999	0.18455	-0.38096	-0.33607
77.3	0.02977	0.15628	-0.40129	-0.26481
77.4	0.05503	-0.06472	-0.33452	-0.44279
77.5	0.12697	-0.09790	-0.21607	-0.46305
77.6	0.01548	-0.27891	-0.15940	-0.33404
77.7	-0.14536	-0.39048	-0.02638	-0.20855
77.8	-0.35519	-0.55032	-0.04842	-0.14512
77.9	-0.49369	-0.55106	0.00565	-0.25586
78.0	-0.60896	-0.59751	-0.03206	-0.20785
78.1	-0.56546	-0.60418	0.08660	-0.30255
78.2	-0.48990	-0.63121	0.14591	-0.34110
78.3	-0.49269	-0.53277	0.18411	-0.24159
78.4	-0.45235	-0.37194	0.11529	-0.20913
78.5	-0.43968	-0.23092	0.08115	-0.05160
78.6	-0.32769	-0.10618	0.01020	-0.09793
78.7	-0.25523	0.01509	0.02815	0.02298
78.8	-0.03851	0.25261	0.08842	0.02441
78.9	0.24368	0.37178	0.16498	0.03878
79.0	0.38492	0.40754	0.15909	0.07005
79.1	0.50682	0.48094	0.11083	-0.00094
79.2	0.63812	0.50747	-0.00114	-0.08546
79.3	0.62731	0.60075	0.00984	-0.24352
79.4	0.62684	0.70342	-0.00962	-0.14322
79.5	0.61032	0.69913	0.01192	-0.03484
79.6	0.65548	0.76162	0.00300	0.02525
79.7	0.36072	0.65563	-0.03933	-0.05582
79.8	0.28163	0.53461	-0.20501	-0.13251
79.9	0.25983	0.41580	-0.35043	-0.21136
80.0	0.38210	0.18424	-0.45573	-0.36755
80.1	0.22170	0.14749	-0.54598	-0.52612
80.2	-0.02302	-0.22278	-0.70895	-0.67660
80.3	-0.43880	-0.39463	-0.83670	-0.76660
80.4	-0.56853	-0.38649	-0.80006	-0.73057
80.5	-0.69945	-0.44287	-0.74073	-0.69849
80.6	-0.64477	-0.63941	-0.53402	-0.61134
80.7	-0.88342	-1.02188	-0.25987	-0.46153
80.8	-0.99666	-1.02482	0.03046	-0.22340
80.9	-0.93610	-0.93056	0.18657	0.14819
81.0	-0.83590	-0.81109	0.34948	0.38076
81.1	-0.69429	-0.81376	0.53038	0.57173
81.2	-0.69167	-0.70648	0.64623	0.66695
81.3	-0.61188	-0.55730	0.73564	0.80747
81.4	-0.40047	-0.34910	0.74257	0.96348
81.5	-0.04237	-0.10156	0.81370	1.02541
81.6	0.19124	0.11775	0.82611	1.03965
81.7	0.47701	0.42832	0.88452	1.02966
81.8	0.57716	0.56935	0.88270	0.94958
81.9	0.76488	0.68631	0.71651	0.72296
82.0	0.73798	0.75578	0.45930	0.46691
82.1	0.80106	0.77500	0.16107	0.16670
82.2	0.79745	0.73906	-0.03911	0.03039
82.3	0.73832	0.49346	-0.20185	-0.15539
82.4	0.49316	0.18925	-0.31578	-0.21520
82.5	0.14457	-0.21180	-0.44880	-0.42736
82.6	-0.16294	-0.51911	-0.59640	-0.64805
82.7	-0.51875	-0.70799	-0.69264	-0.86622
82.8	-0.75469	-0.87146	-0.77036	-0.87575
82.9	-0.98391	-0.96012	-0.81948	-0.85292

83.0	-0.97947	-1.07949	-0.89938	-0.83779
83.1	-0.97746	-1.01346	-0.83905	-0.95076
83.2	-0.93966	-0.99525	-0.76451	-0.83892
83.3	-0.91956	-0.88544	-0.64587	-0.69892
83.4	-0.79909	-0.78743	-0.48170	-0.45139
83.5	-0.65004	-0.61671	-0.26800	-0.32422
83.6	-0.47829	-0.44622	0.00394	-0.07164
83.7	-0.27341	-0.34012	0.26475	0.20043
83.8	-0.05786	-0.20520	0.57454	0.52904
83.9	0.19437	0.01559	0.73327	0.70084
84.0	0.37665	0.35406	0.77855	0.83304
84.1	0.46134	0.62928	0.75972	0.83259
84.2	0.54342	0.48820	0.87020	0.79378
84.3	0.64003	0.33507	0.75510	0.84917
84.4	0.79495	0.10470	0.68883	0.73657
84.5	0.75324	0.23285	0.50702	0.58726
84.6	0.65464	0.50259	0.31683	0.25880
84.7	0.54080	0.61208	0.03412	0.09160
84.8	0.44517	0.63392	-0.29333	-0.05523
84.9	0.33565	0.24206	-0.49002	-0.27243
85.0	0.17507	0.16116	-0.66603	-0.50004
85.1	0.01224	-0.10166	-0.72957	-0.66187
85.2	-0.24518	-0.30970	-0.78616	-0.78115
85.3	-0.46378	-0.67989	-0.83561	-0.84097
85.4	-0.59129	-0.81992	-0.87305	-0.86473
85.5	-0.62317	-0.89294	-0.81671	-0.81905
85.6	-0.71558	-0.79636	-0.73394	-0.79545
85.7	-0.72246	-0.74075	-0.63961	-0.77282
85.8	-0.66413	-0.43764	-0.42537	-0.62339
85.9	-0.42570	-0.33734	-0.29620	-0.46956
86.0	-0.25402	-0.18031	-0.04608	-0.27056
86.1	-0.04162	-0.29703	0.09667	-0.03096
86.2	0.03814	-0.22395	0.38572	0.23682
86.3	0.07791	-0.21481	0.55196	0.46932
86.4	-0.02837	-0.09901	0.68251	0.52718
86.5	-0.08444	-0.05939	0.77956	0.62469
86.6	-0.09716	-0.02477	0.79228	0.67340
86.7	-0.04422	-0.04479	0.76262	0.65861
86.8	0.00087	-0.12722	0.66310	0.59333
86.9	-0.03082	-0.13261	0.63390	0.40399
87.0	-0.08979	-0.05638	0.59831	0.29845
87.1	-0.08375	0.05750	0.44962	0.25618
87.2	-0.08968	-0.03153	0.21754	0.22289
87.3	-0.00085	0.02320	0.07009	0.13902
87.4	-0.08033	-0.05981	-0.02641	-0.05419
87.5	-0.01770	0.08097	-0.13164	-0.12638
87.6	-0.10093	0.13565	-0.34080	-0.24405
87.7	-0.07160	0.19186	-0.40854	-0.31524
87.8	-0.11239	0.13838	-0.44930	-0.43440
87.9	-0.01867	0.10090	-0.49525	-0.48335
88.0	0.02501	0.08318	-0.57801	-0.55713
88.1	0.04809	0.03142	-0.56969	-0.56740
88.2	0.00300	-0.00749	-0.51777	-0.56579
88.3	-0.00303	-0.07774	-0.38831	-0.49899
88.4	0.03370	-0.00260	-0.28172	-0.38101
88.5	0.08429	-0.06732	-0.17401	-0.25705
88.6	0.13831	0.08984	-0.15693	-0.19901
88.7	0.12417	0.11511	-0.12302	-0.17890
88.8	0.09764	0.11082	-0.09669	-0.17392
88.9	0.04991	0.06227	-0.01878	-0.06438
89.0	0.03528	0.04164	-0.01527	-0.03476

Appendix 5: Bunny Hills Graphs

Row 1 and row 12 have been split into two different graphs so it is easier to see at what time the rows go over the tops of the bunny hills. The three bunny hills have been labeled on the graphs.


Z Acceleration: Bunny Hills, Row 1
1L Z
1R Z
Acceleration (g)
Top of First Hill
Top of Second Hill
Top of Third Hill
Time (seconds)


Z Acceleration: Bunny Hills, Row 12
12L Z
12R Z
Acceleration (g)
Top of First Hill
Top of Second Hill
Top of Third Hill
Time (seconds)