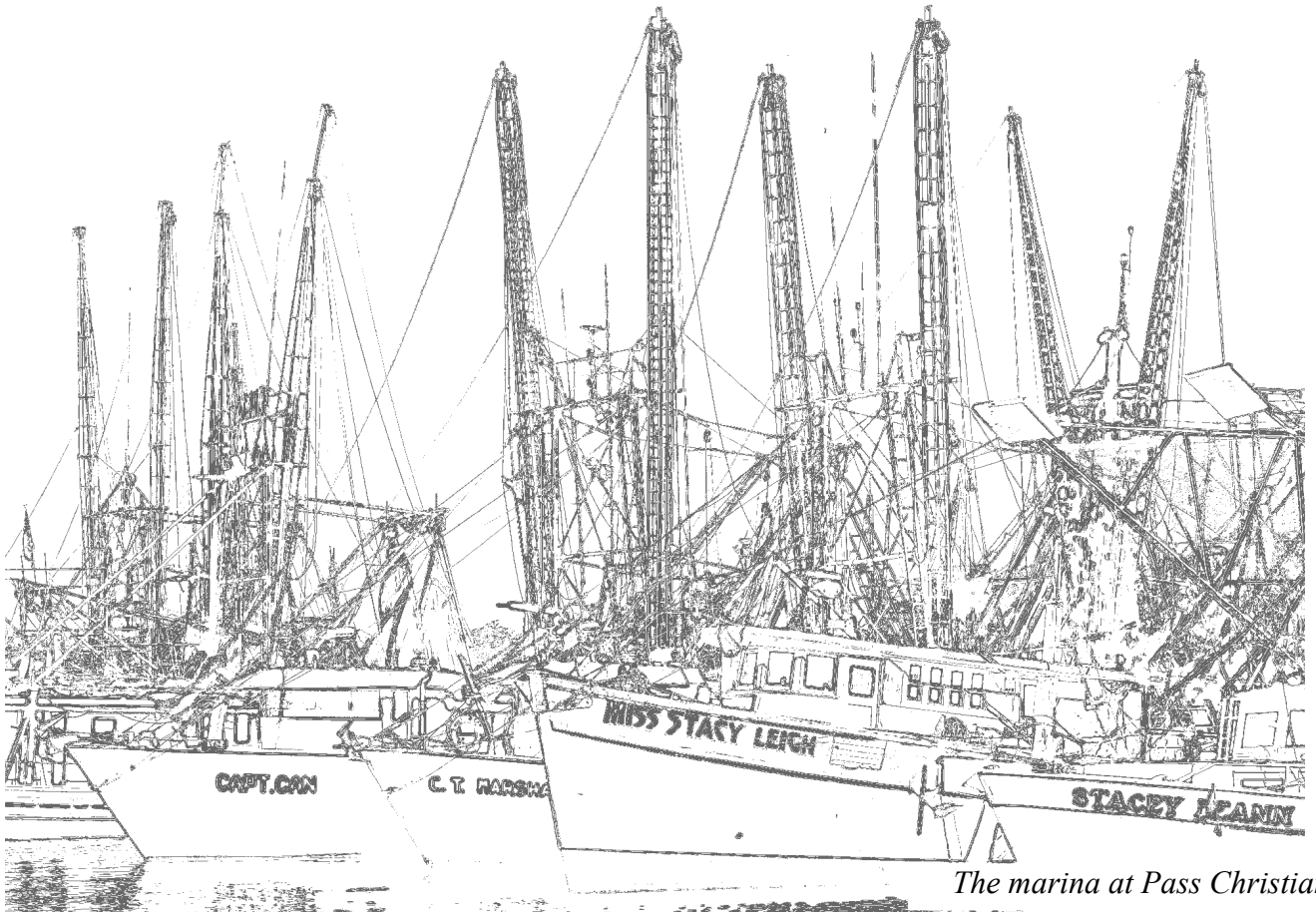
Gulf Coast Data Concepts
www.gcdataconcepts.com

Calculating the Acceleration of Gravity

Test Report: TR003

Revision: Rev A
Date: June 20, 2017

Abstract: Occasionally, it's a good idea to check some of the fundamental laws of physics for validity. You never know if those physicists made a mistake or if the nature of the universe recently took a turn for the worse. Here, a USB accelerometer data logger made by Gulf Coast Data Concepts was used to determine the periodicity of a simple gravity pendulum and to calculate acceleration of gravity. The test determined the local gravity as 32.24082 ft/sec^2, which is within 1% of Earth's nominal gravity of 32.1740 ft/sec^2. The results indicate that gravity does, in fact, still work, and all is well in the universe.



The marina at Pass Christian, MS

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Document Conventions

This test report describes in detail the background information, procedure, and analysis method used to conduct an experiment using a GCDC data logger. This experiment is an example application using an accelerometer that is both educational and fun.

Each section also presents relevant tips and warnings to help the user repeat the experiment or make improvements.

This icon indicates a helpful tip that may enhance the results or add a new perspective to the objective.

This icon indicates a warning, restriction, or limitation that the user should be aware of regarding the experiment or logger operation.

1.2 Repeating this Experiment

Obviously, a GCDC data logger is required to repeat the test procedure. The exact product type is described in section 4. However, the raw data and analysis methods are provided such that a user may recreate the same results without repeating the test process. Stepping through the analysis process with the data files will help the user understand the steps and see the expected results. The raw data files, spreadsheets, and R scripts are available for download at www.gcdataconcepts.com.

A spreadsheet is used for simple analysis, so the user should be familiar with manipulating data and creating plots in a spreadsheet. We recommend Microsoft Excel or OpenOffice Calc.

"R" is used for more complex data analysis. R is a simple command line programming environment that can manipulate large data sets using common math commands as well as complex function libraries. The software is compact, free, and available at www.r-project.org for Windows, Mac, and Linux. The R scripts provided herein are designed to run on Windows and may not be fully compatible with Mac and Linux systems. Specifically, file path references will not translate properly to the Mac and Linux systems. These differences will be noted in the script comments.

1.3 Disclaimers

This test report is presented "as-is" and for informational purposes only. No claims, representations or warranties, whether expressed or implied, are made to the safety and performance of the procedures described herein. Furthermore, we do not make any guarantee that your son or daughter will get an "A" on their science project, which is probably due tomorrow morning...but that's not our problem, and you should have started earlier anyway.

2 Background Information

A gravity pendulum is simply a weight swinging from a pivot point (see Figure 1). Hanging vertically, the pendulum is at its equilibrium position. Drawing the pendulum back introduces potential energy into the system. Releasing the pendulum allows it to "fall" towards its equilibrium position. The inertia allows the pendulum to swing past vertical and continue "up", storing energy for the next swing back. The time needed for a pendulum to complete a full swing cycle is called the period. The back and forth motion, or oscillation, will continue until the initial potential energy is lost to friction, aerodynamic drag, and other such effects.

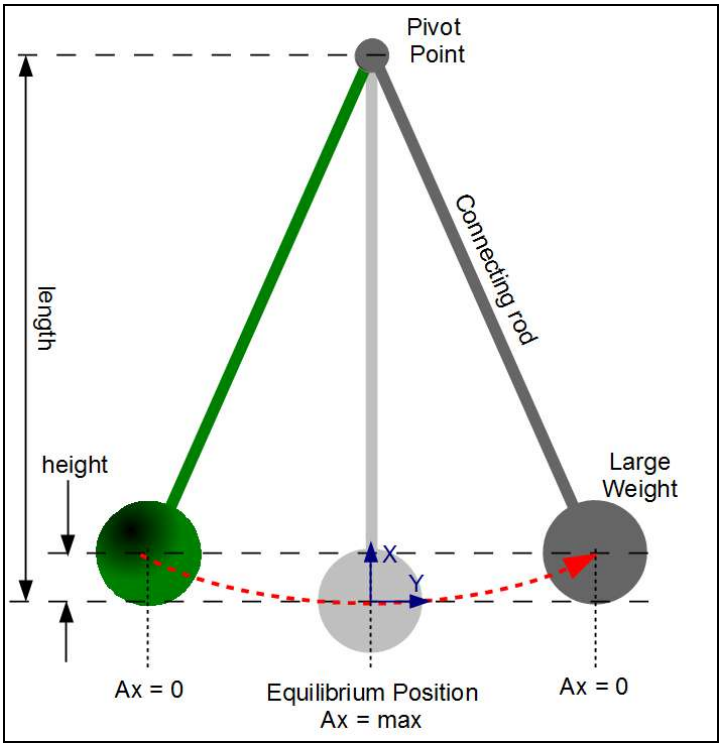


Figure 1: Diagram of a Pendulum

Galileo Galilei discovered that the period was dependent on the length of the pendulum, and, in 1656, Christiaan Huygens used this idea to build the first pendulum clock. The pendulum clock concept was a vast improvement in timing precision, and it continued as the standard timing method for hundreds of years.

Several assumptions simplify the pendulum system into an easy mathematical model. First, a thin connecting rod and a large mass pendulum diminish the effects of friction and air drag. Also, a long connecting rod and small swing angle allow the motion to be approximated as a simple, harmonic motion. Therefore, the oscillation period is defined by the following formula:

$$T = 2\pi\sqrt{\frac{l}{g}} \qquad \textbf{Equation 1}$$

Where l is the length of the pendulum between the pivot point and center of gravity, g is the gravitational acceleration, and T is the period of one complete swing.

Although the pendulum is typically used as a timing instrument, it can also function as a tool to measure gravity – called a gravimeter. Using Equation 2 and solving for g:

$$g = \frac{l}{\left(\frac{T}{2\pi}\right)^2} \qquad \textbf{Equation 2}$$

Equation 2 shows that the local gravity can be calculated using the precise measurement of the pendulum length and period. The pendulum length is easy to obtain using common methods (ie, a measuring tape). This report describes how to measure the pendulum period, and consequently determine gravity, using a precisely timed accelerometer data logger. The accelerometer is orientated with the x-axis inline with the connecting rod such that the acceleration profile of Ax oscillates to a maximum with each pass through the equilibrium point. The repeating acceleration profile is used as a basis for determining the pendulum's period.

3 Objective

This test calculated the acceleration of gravity using a simple gravity pendulum.

4 Experiment Setup

4.1 Materials and Equipment

Table 1: Materials and Equipment List

Description	Quantity	Comments
Weight	1	Standard 2lb exercise weight
Eyehook	1	Available at most hardware stores
X16-1D USB accelerometer	1	Purchase online from GCDC. Any accelerometer in the X16 series will work.
High strength fishing line	-	This experiment uses Berkley Fireline. Other string-like materials would be fine, but note that regular string might stretch or twist.
Electrical tape	-	Any tape will suffice
Measuring tape	1	12' long retracting tape measure

Figure 2: Supplies for Experiment

4.2 Equipment Setup

4.2.1 Logger Configuration

Figure 3 lists the configuration settings for the X16-1D accelerometer data logger. A simple text editor, such as Wordpad or Notepad++, was used to modify the config.txt file.

```
;Example X16-1D config file
;set sample rate
;available rates 12, 25, 50, 100, 200, 400
samplerate = 100
;record constantly
deadband = 0
deadbandtimeout = 0
;set file size to 5 minutes of data
samplesperfile = 30000
;set status indicator brightness
statusindicators = high
;rebootOnDisconnect
```

Figure 3: Logger Configuration

Set the RTC so that the data files will have the correct date and time. This is important to correlating the data to the notes taken during the test. See the X16-1D user manual regarding the RTC initialization procedure.

4.2.2 Test Setup

For this experiment, a small dumbbell weight served as the pendulum bob, and an eyehook attached to a door frame worked as the pivot point. The effective length of the pendulum arm is the distance between the pivot point and the center of mass. The string is negligible weight, so the center of mass is located at the middle of the dumbbell.

The following steps outline the procedure for the experiment setup:

1. Attached eyehook to the top of a standard door frame (see Figure 4).
2. Tied fishing line around weight.
3. Taped the logger onto the weight with the On/Off button facing up for easy access.
4. Threaded the fishing line through the eyehook and knotted it so that the weight hovered several inches above the ground (see Figure 5). Note that the weight does not have to hang evenly, as this will not affect the outcome.
5. Measured the length of the fishing line from the knot at the eyehook to the middle of the weight (see Figure 4). This distance is the effective length of the pendulum.

Figure 4: Mounting and Measurement of Pendulum

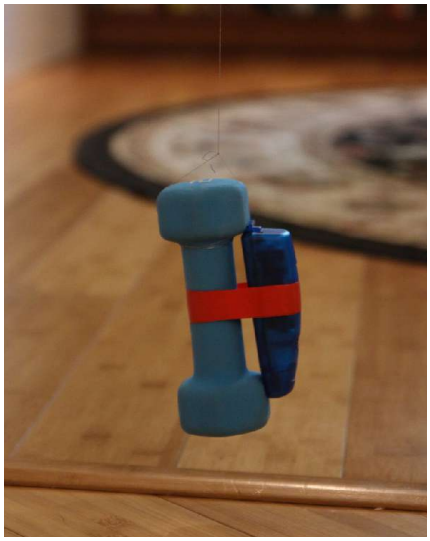


Figure 5: Pendulum Weight and X16-1D Logger

Figure 6: Experiment Setup

4.3 Software

The analysis was performed using "*R*", which is a free open source mathematics package available at www.r-project.org. An R script called "pendulum_analysis.r" was developed to analyze the test data from the accelerometer data logger and determine the time period of the pendulum (see Appendix A for full script).

The script converted the accelerometer time series data to the frequency domain using a Fast Fourier Transform (FFT) function. Consecutive segments of data were taken from the file and processed

through the FFT until the entire file was analyzed. All the FFT results were averaged together into a single spectral plot of acceleration versus frequency. This method averaged out intermittent changes in behavior and increased the signal-to-noise ratio of the results.

The plot illustrated the dominant frequency of the swinging pendulum, which is due to the maximum acceleration in the x-axis as the pendulum swings past the vertical position. A complete swing includes the pendulum passing through the vertical position twice. Therefore, the swing frequency of the pendulum is half the dominant frequency, and the swing period is the inverse of the that frequency.

The script and the raw data files used in this report are available to download from the GCDC website. Copy the script file to the root directory of the X16-1D logger to allow easy access from the R console. Copy the data files to a temporary directory on the data logger.

5 Procedure

5.1 Test Procedure

1. Turned on the logger, pulled the weight back approximately 18 inches while keeping the string taut, and released, causing the weight to swing. Note that the weight does not have to swing in a straight motion, as the analysis procedure takes this into account.
2. Allowed the pendulum to swing for 5 minutes before stopping it and turning off the logger.

5.2 Analysis Procedure

Figure 7 illustrates the R console commands used to implement the analysis script.

The following steps outline the procedure for using the R script to analyze the data (see Figure 7):

1. Saved the script “pendulum_analysis.r” file to the root directory of the data logger. This made it easier to access the script from the R console.
2. Loaded the script code into R using the “source” command.
3. Assigned the pendulum length determined from the test setup to the variable called “length”.
4. Used the function “pendulum_Analysis” to determine the period of the pendulum and assigned the result to the variable called “period”. A plot showing the low frequency spectral response appeared, which will indicated the pendulum oscillation frequency.
5. Calculated gravity using Equation 2 and assigned the result to “g”.
6. Entered “g” to see the answer expressed in ft/sec^2. Earth's nominal gravity is 32.1740 ft/sec^2.

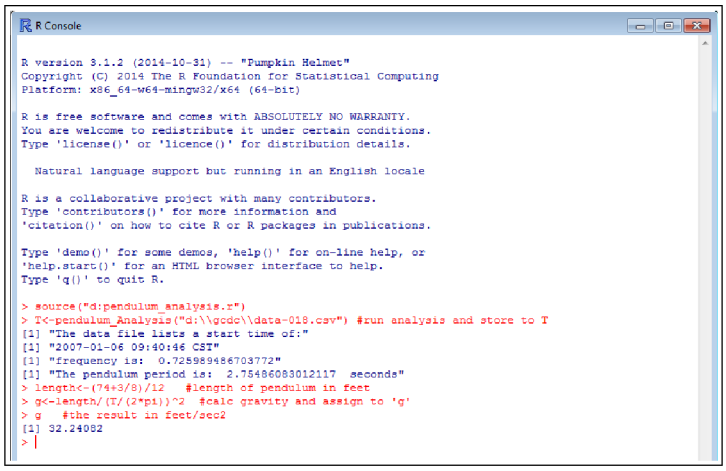


Figure 7: Using the Script form the R Console

6 Results

R automatically plotted the results of the analysis and determined the dominant frequency to be 0.725989 Hz, as illustrated in Figure 8. There are two acceleration peaks per cycle since the pendulum swings past center twice. Therefore, the frequency of the pendulum is half that of the FFT result, or 0.3629945 Hz. The resulting pendulum period was 2.75486 seconds.

R calculated gravity using the 2.75486 second period and Equation 2, as seen in Figure 7. Gravity equaled 32.24082 ft/sec^2.

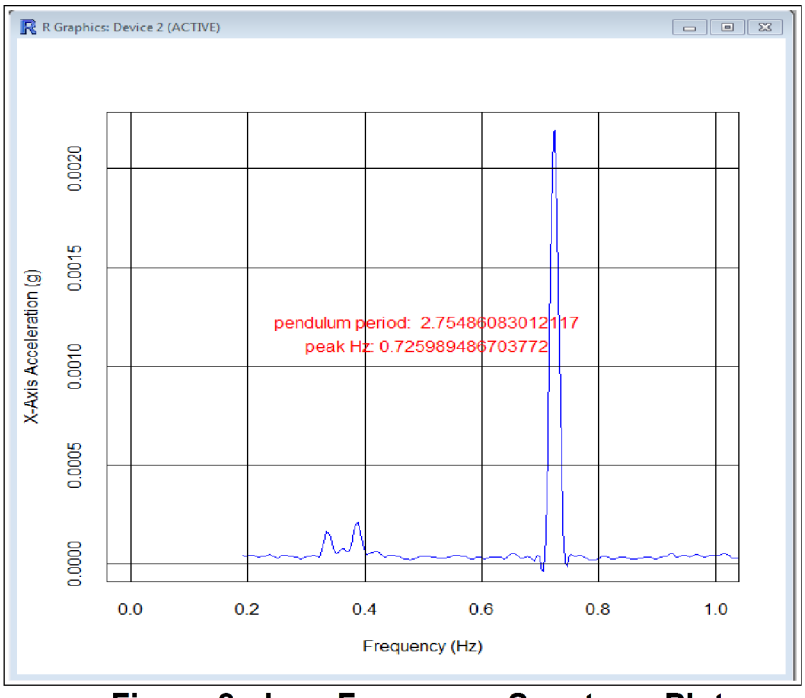


Figure 8: Low Frequency Spectrum Plot

7 Discussion

Using R to determine acceleration due to gravity resulted in a very accurate assessment of Earth's gravity. The test determined the local gravity as 32.24082 ft/sec^2, which is within 1% of Earth's nominal gravity of 32.1740 ft/sec^2.

8 Conclusion

Calculating the acceleration of gravity to within 1% of actual using such a crude experiment is impressive. Furthermore, we can conclude that the gravity still works, and all is well in the universe (at least in Waveland, MS).

9 Appendix A: R Script – pendulum_analysis.r

```
#Gulf Coast Data Concepts
#June 20, 2017
#file name: "pendulum_analysis"
#R script for processing X16 accelerometer data
#to find the dominate frequency within a data set.  The
#script is designed to work with the Pendulum Gravity Test procedure
#-------------------------------------------------
#This script takes a segment of data (a block) and converts
#the time series into a frequency series (Fast Fourier Transform)
#The FFT output represents the energy level for each frequency bin
#The analysis repeats by pulling successive blocks of data
#throughout the data set.  Each block is plotted during the analysis.
#The results of each FFT are summed and averaged, then the dominate
#frequency is determined (resonant freq)
#-------------------------------------------------

#####################################################################
#Functions must be declared first before the script can utilize them #
#####################################################################

#the next few lines of code build a Hann Window filter
#for processing successive FFTs. It's good to filter the
#input data such that there are no discontinuities in the
#transitions between blocks of data.  Hann Window is a common
#way to do this.
hannWindow<-function(block){
  #first, initialize the array with a zero
  hann.window<-0
  #the first element of the hann.window array has a 0 so the
  #index will be initialized to '2' to address the next array element
  z<-2
  #start a loop that will build an element array of factors
  #representing a Hann filter
  while(z<(block+1)){
    #this is the formula for a Hann filter
    temp<-0.5*(1-cos((2*pi*z)/(block-1)))
    #take the new factor and append it to the hann.window array
    hann.window<-append(hann.window, temp)
    #increment the index by 1
    z<-z+1
  }
  return(hann.window)
}

#####################################################################
# START OF MAIN SECTION                                             #
#####################################################################
pendulum_Analysis<-function(dataFile){

  ###########################################################
  #the following parameters affect the analysis algorithm  #
  ###########################################################

  #open the file and read the first six lines
  #the third line contains the start_time for the X16
  lines<-readLines(dataFile,6)
  #parse the lines
  date<-unlist(strsplit(lines[3],","))
  #combine the 2 and 3 entries into a date/time
  d<-paste(date[2],date[3])
  #convert the date/time text into a R type date
  startDate<-as.POSIXlt(d)

  print("The data file lists a start time of:")
  print(startDate)
```

```
#pull the sample rate setting from the first data file
temp<-unlist(strsplit(lines[5],","))
SR<-as.numeric(temp[2])
#note that the X2 logger puts the sample rate on the 3rd line
#so change the previous index from '2' to '3'

#block is a segment of data processed through the FFT algorithm
#More data is better
block<-SR*120 #two minutes of data

#the script progresses through data set pulling
#blocks of data.  Each block of data can overlap the
#previous block of data.  Overlapping the data improves
#the signal-to-noise of the results but increases
#computations and time
#define the FFT overlap, 1=0%, 2=50%, 4=75%, etc
overlap<-4   #75% of data will overlap previous block of data
              #and 25% will be new data from the file

#####################################################################
# Analysis parameters finished.  Now, set up variables before       #
# starting the main part of script                                  #
#####################################################################

#cfactor is used to convert the raw data into "g".  The factor
#is different depending on the logger type and sensor range.
#X16 series  -> 2048
#X200               -> 163.8
#X2, low gain       -> 6554
#X2, high gain-> 13108
cfactor<-2048

#initialize three arrays to store the xyz data
dataX<-array(0)
dataY<-array(0)
dataZ<-array(0)

#a time value is calculated for each FFT, time increases with
#each new block of data as the script proceeds through a file.
#The time values are stored to timearray.
time<-0

#timearray stores all the FFT time values and is used later for
#plotting purposes
timearray<-array(0)

#build a history of FFTs
fftHistory<-array(0)

#build a Hann Window filter
#when processing successive FFTs it's good to filter the
#input data such that there are no discontinuities in the
#transitions.  Hann Window is a common way to do this.
hann.window<-hannWindow(block)

#build an array of frequency values based on fft size
#this is used for plotting the spectral output
x.axis<-0:(block/2-1)
x.axis<-x.axis*((SR/2)/(block/2))

colNames<-c("time", "Ax", "Ay","Az")
dataIn<-read.table(dataFile, sep=",", comment=";", col.names=colNames, fill=TRUE)

#use spline to resample the data into evenly spaced samples.  The accelerometer
#sensor streams data at the set sample rate but this may vary depending on
#the accuracy of the sensor's clock.  For example, configuring to sample at 400Hz
#may actually result in data recorded at 410Hz.  We need the resulting data to be
#very accurate and consistent.  Therefore, as data arrives to the logger CPU, it is time
```

```
  #stamped using an independent and accurate real time clock.  These time stamps
  #are assumed to be accurate.  We will resample the data based on
  #the time stamps to ensure the time series data is consistent and accurate.
  ################################################################################

  #determine the total time recorded in the file by finding the
  #largest time stamp and subtracting the smallest time stamp
  timeFile=max(dataIn[,1])-min(dataIn[,1])
  #total number of samples that should be in the file assuming
  #the sample rate used.  In reality, the sensor may produce data
  #slower or faster than the set rate, that's why we are resampling
  numberSamples=timeFile*SR
  #The time stamps are based on the real time clock so these values are
  #accurate.  However, the arrival of the data could be different than the
  #set sample rate due to the sensor's clock error.  Therefore, we will
  #use a cubic spline algorithm to resample each axis such that the
  #sample timing is consistent.  This will improve the accuracy of the FFT
  #by correcting the sample rate error from the sensor.
  resample<-spline(dataIn[,1],dataIn[,2],n=numberSamples,xmin=min(dataIn[,1]),xmax=max(dataIn[,1]))
  dataTime<-unlist(resample[1])
  dataX<-unlist(resample[2])
  resample<-spline(dataIn[,1],dataIn[,3],n=numberSamples,xmin=min(dataIn[,1]),xmax=max(dataIn[,1]))
  dataTime<-unlist(resample[1])
  dataY<-unlist(resample[2])
  resample<-spline(dataIn[,1],dataIn[,4],n=numberSamples,xmin=min(dataIn[,1]),xmax=max(dataIn[,1]))
  dataTime<-unlist(resample[1])
  dataZ<-unlist(resample[2])

  #Convert the raw data into g's
  dataX<-(dataX)/cfactor
  dataY<-(dataY)/cfactor
  dataZ<-(dataZ)/cfactor

  #use the x-axis data in the FFT analysis
  #change the following code accordingly if you want to check the Ay or Az
  data<-dataX

  #this is the main loop to calculate each FFT.  First, a block data points
  #is pulled from the input data and the Hann filter applied to it.  An FFT is
  #performed on the filtered data.  The "Mod" function calculates the magnitude
  #using the real and imaginary output of the FFT.  The results of the FFT are
  #summed.  A time value is calculated for each FFT but this is not important
  #for the analysis
  i<-1
  while((i+block)<length(data)) {
    datablock<-data[i:((block-1)+i)] #get the data
    data.hann<-hann.window*datablock #apply the filter
    data.fft<-fft(data.hann)         #perform FFT
    data.mag<-Mod(data.fft)          #calculate magnitude
    #use the first half of the FFT results, the second half is "imaginary" numbers
    data.mag.half<-data.mag[1:((length(data.mag))/2)]

    #correct the results
    fftOutput<-data.mag.half/(block/2)

    #calculate the time stamp for this particular block of data
    time<-(i+block)/SR/60

    #uncomment the following lines if you want to see the spectral plot as it's calculated
    ##combine the FFT and x.axis into one array for plotting purposes
    #output<-array(c(x.axis, fftOutput), dim=c(length(x.axis),2)) #combine arrays
    ##plot the output in log scale on the y-axis
    #plot(output, type="l", xlim=c(0,SR/2), ylim=c(1e-6,1e6), log="y", tck=1, xlab="Frequency",
ylab="Total RMS Acceleration (g)")
    #text(round(time, 1), x=70, y=50)
    #text("elapsed minutes", x=70, y=10)

    #add the fft result to the history
```

```
    fftHistory<-fftHistory+fftOutput

    #save the time stamp associated with this block of data
    timearray<-append(timearray, time)

    #increment the index by a percentage of the block
    #defined by the overlap variable.
    i<-i+(block/overlap)

    #back to the top of loop and repeat everything for
    #the next block of data
  }

  #convert time into date
  timeDate<-startDate+timearray*60

  #average out the FFT history
  fftHistory<-fftHistory/(length(timearray))

  #combine the fftHistory with the x.axis frequencies
  output<-array(c(x.axis, fftHistory), dim=c(length(x.axis),2))
  #the analysis will now look at the low frequencies between 0.25 and 2 hz
  #a 5 to 9 foot long pendulum will be less than 1 Hz
  lF<-0.2
  hF<-2
  high<-round(length(fftHistory)*hF/(SR/2)) #high freq index pt
  low<-round(length(fftHistory)*lF/(SR/2))  #low freq index pt
  #resample the data to interpolate between the data points
  ns<-(hF-lF)*300
  temp<-spline(output[low:high,1],output[low:high,2],n=ns,xmin=output[low,1],xmax=output[high,1])
  x<-unlist(temp[1])  #time
  y<-unlist(temp[2])  #FFT result (acceleration)
  #combine the new resampled data into a new array
  rsFFT<-array(c(x,y),dim=c(length(x),2))
  #find the peak within the resampled data range
  maxValue<-max(rsFFT[,2])
  freq<-rsFFT[which.max(rsFFT[,2]),1]
  #display the result
  print(paste("frequency is: ", freq))

  #plot averaged spectral response for the entire data set
  plot(rsFFT, type="l", tck=1, xlab="Frequency (Hz)", ylab="X-Axis Acceleration (g)", xlim=c(0,1),
ylim=c(0,max(rsFFT[,2])), col="blue")
  text(paste("peak Hz:",freq), x=freq*.7, y=maxValue/2, col="red")

  #the pendulum period is the time needed for a complete swing.  The pendulum
  #crosses vertical twice for each complete swing.  therefore, the period is half the
  #calculated frequency.  time = inverse(frequency)
  period<-1/(freq/2)
  text(paste("pendulum period: ",period), x=freq*.7, y=maxValue/1.8, col="red")
  #display result
  print(paste("The pendulum period is: ", period, " seconds"))

  #return the pendulum period
  return(period)

  #function complete
}
```

End of Document