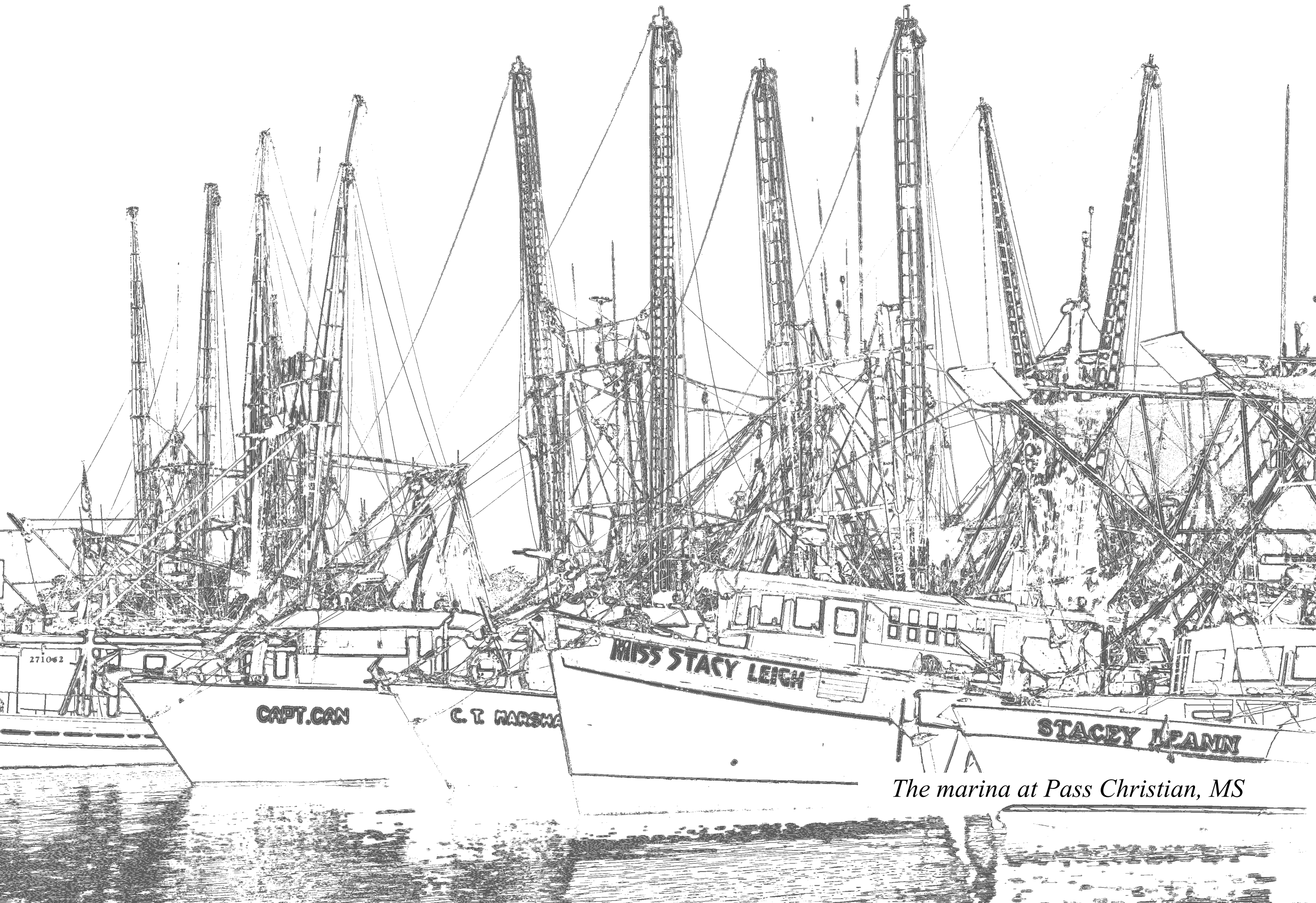
Gulf Coast Data Concepts
www.gcdataconcepts.com

Detecting a Freight Train

Test Report: TR004

Revision: A
Date: June 7, 2017

Abstract: For those who have experienced tornadoes and hurricanes, the extreme winds roaring overhead can sound like a passing freight train. It's an eerie and frightening sound that seems to shake the Earth. Fortunately, we have not had the opportunity to study this phenomenon directly. However, a nearby railway allowed us to evaluate the vibration of a freight train using the X2-2 logger. Using the X2-2 provided an easy indication of a passing train using the infrasonic range.



The marina at Pass Christian, MS

Table of Contents

1 Introduction....1
1.1 Document Conventions....1
1.2 Repeating this Experiment....1
1.3 Disclaimers....1
2 Background Information....2
3 Objective....2
4 Experiment Setup....2
4.1 Materials and Equipment....2
4.2 Equipment Setup....3
4.3 Software....3
5 Procedure....4
5.1 Test Procedure....4
5.2 Analysis Procedure....4
6 Results....4
7 Discussion....6
8 Conclusion....6
9 Appendix A: R Script – train_vibration.r....7

List of Figures

Figure 1: Train....2
Figure 2: Logger Configuration....3
Figure 3: Total RMS Acceleration....4
Figure 4: RMS Acceleration for Train 1....5
Figure 5: Base Noise Level....5
Figure 6: Vibration Level with Train Present....6

List of Tables

Table 1: Materials and Equipment List....2

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

For those who have experienced tornadoes and hurricanes, the extreme winds roaring overhead can sound like a passing freight train. It's an eerie and frightening sound that seems to shake the Earth. Fortunately, we have not had the opportunity to study this phenomenon directly. However, a nearby railway allowed us to evaluate the vibration of a freight train using the X2-2 logger. Is the vibration similar to a tornado or hurricane? Let's hope we don't find out...

3 Objective

The test analyzed data collected from an X2-2 accelerometer data logger to detect vibration caused by the passing of freight trains.

4 Experiment Setup

4.1 Materials and Equipment

Table 1: Materials and Equipment List

Description	Quantity	Comments
X2-2 Accelerometer Data Logger	1	Purchase online from GCDC.
Small 2-lb rock	1	Keeps the logger in place
Duct tape	-	
Access to a railway	-	

Figure 1: Train

4.2 Equipment Setup

4.2.1 Logger Configuration

The vibration from a freight train is very broad spectrum, but this analysis used the infrasonic range (<20Hz). The logger was configured to collect constantly at 256Hz, which allowed a 0-128Hz frequency band spectrum from the Fast Fourier Transform (FFT).

```
;X2-2 configuration
;set sample rate
samplerate = 256
;record constantly
deadband = 0
deadbandtimeout = 0
;set file size to about 3 minutes of data
samplesperfile = 50000
;set status indicator brightness
statusindicators = Normal
microres
gain = low
```

Figure 2: Logger Configuration

4.2.2 Test Setup

The X2-2 logger is relatively small and light weight so laying it on the ground or in grass will not provide good coupling to the vibration. The logger was attached to a small 2 pound rock using the duct tape. The weight of the rock kept the logger in place on the ground. Another option was to bury the logger in the ground but the rock method was quick and simple.

4.3 Software

An R script was developed to analyze the test data (see Appendix A for full script). The script converted the accelerometer data to the frequency domain using a Fast Fourier Transform (FFT) function. The FFT function analyzed 1000 samples, or about 4 seconds of data, and created a spectral response between 0 Hz and 128 Hz. Consecutive segments of data were taken from the file until the entire data set was processed. The script filtered the FFT results such that only signals between 1 and 20 Hz were further analyzed. This removed the effects of gravity (<1Hz).

The beginning of the data set did not have a train present so this time frame was used to calculate a baseline noise level. As the analysis progressed, the FFT spectrum was compared to the baseline noise level. A train was detected if the spectral difference exceeded a threshold RMS value of 0.1. The script animates these results to a plot window and overlays "train detected" when a train is passing.

The script saved the output data as a text file, which was later imported into a spreadsheet to access better plotting capabilities.

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 X2-2 logger to allow easy access from the R console. Copy the data files to a temporary directory on the data logger. The script will prompt the user for the drive and directory location of the data.

5 Procedure

5.1 Test Procedure

The following steps outline the procedures taken to collect data on the vibration of freight trains.

1. Started the X2-2 data logger and tossed it into the grass, about 30 feet from the railway.
2. Allowed the logger to record data for about 12 hours before recovering and turning off the logger (data collection started at approximately 9am and ended at 9pm).

5.2 Analysis Procedure

The data was processed using using "*R*", which is a free open source mathematics package available at www.r-project.org. The *R* script used is presented in the Appendix (section 9).

6 Results

Figure 3 summarizes the total RMS acceleration detected in the infrasonic range (between 1-20 Hz) throughout the 12 hour period. This method provided an easy indication of a passing train.

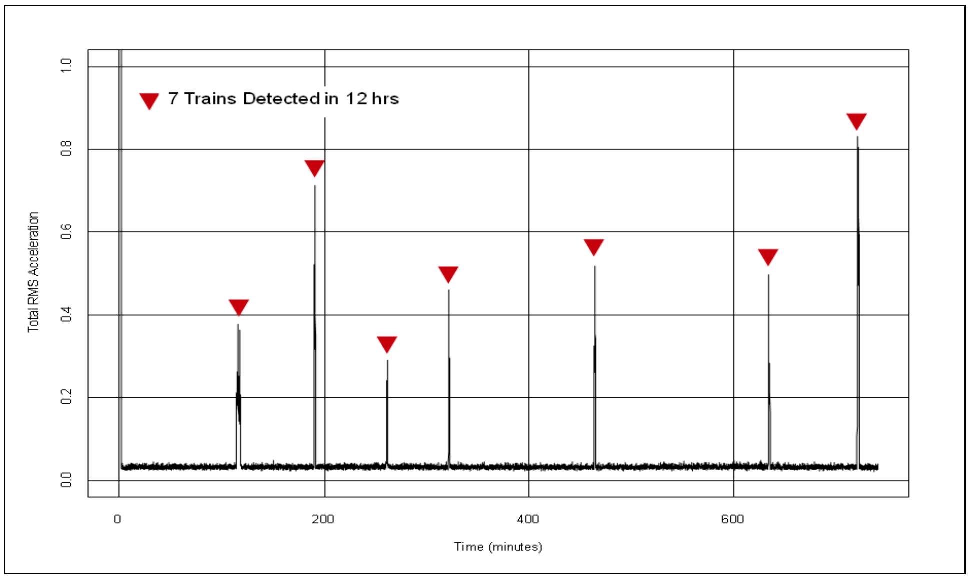


Figure 3: Total RMS Acceleration

Figure 4 is a closer look at the first train detected. The train took 4 minutes to pass by.

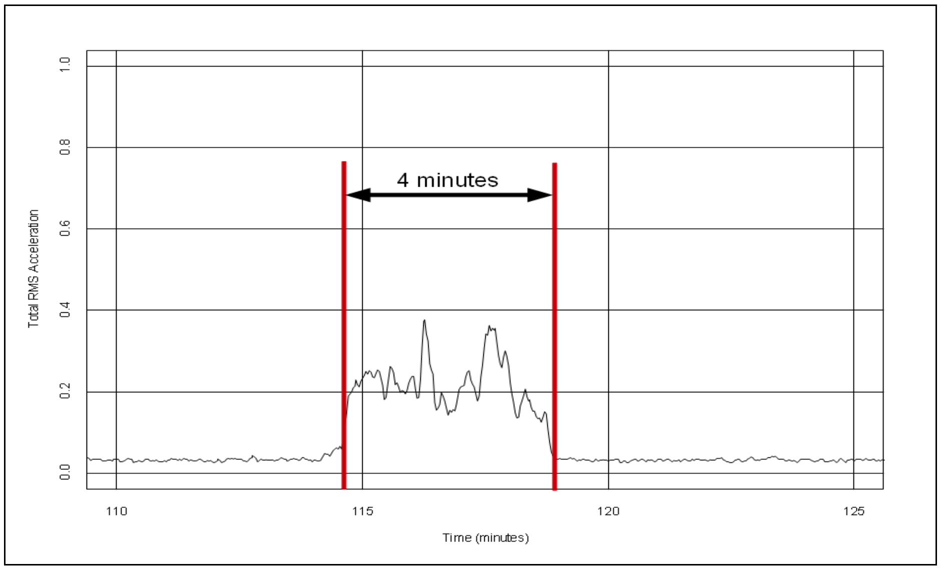


Figure 4: RMS Acceleration for Train 1

Figure 5 illustrates the base noise level reported by the X2-2 when no train is present. Figure 6 illustrates the response due to the vibration of a passing train.

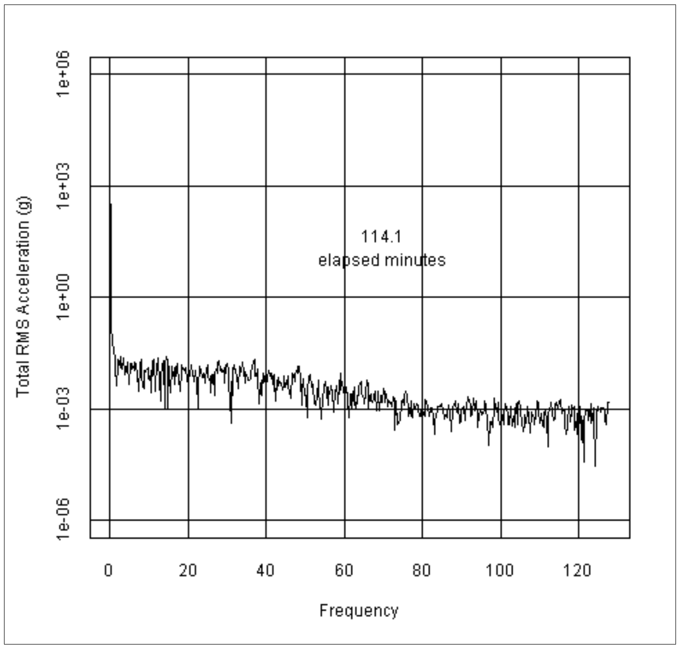


Figure 5: Base Noise Level

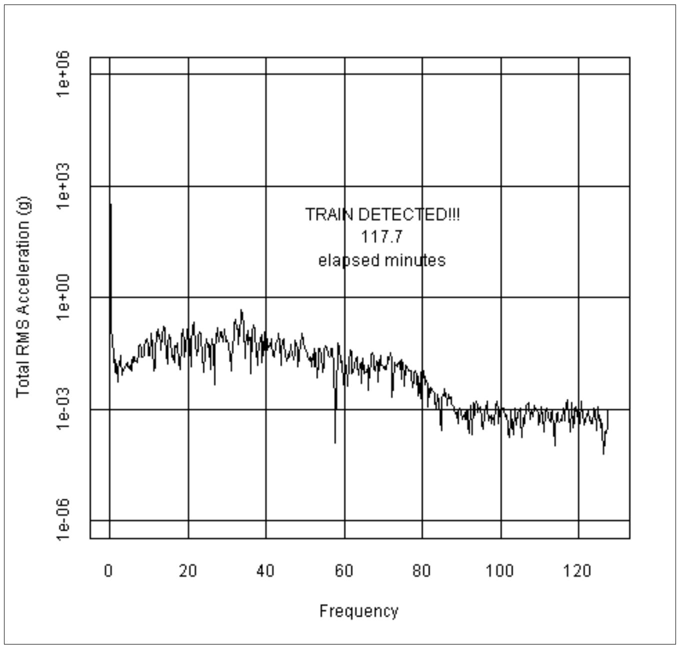


Figure 6: Vibration Level with Train Present

7 Discussion

The results of whether or not the vibration of a freight train equates to a tornado have yet to be verified. Honestly, this is a question best left unanswered.

Given the intensity of the vibration, there was no need in placing the X2-2 so close to the railway. Low frequency vibration travels much easier than higher frequencies, so the logger could be place further from the tracks. A follow-up experiment may investigate the furthest distance allowed to detect a train. In this case, coupling to the ground is important and may require mounting the X2-2 into the ground.

8 Conclusion

Using the X2-2 provided an easy indication of a passing train using the infrasonic range.

"Infrasonic" is a range of frequencies below the typical capability of human hearing (<20 Hz). There is evidence that people are aware of such low frequencies at a subconscious level. Some people speculate that infrasonics influence our state of awareness and could be a basis of haunted houses and paranormal phenomenon. Can the X2-1 logger detect ghosts? Hmmm...

9 Appendix A: R Script – train_vibration.r

```
#Gulf Coast Data Concepts
#August 25, 2013
#file name: "train_vibration.r"
#R script for processing X2 accelerometer data
#to find vibration level of a freight train

#define the drive location of the data files
#this is the drive or path that contains the GCDC directory
drive<-"d:"
#the directory containing the data files
directory<-"\\HSY 230 kV Yard Pile Vibration Recording\\GCDC Data 2013Dec09-13\\MEL-X #2
SN.CCDC42013000080\\X080\\GCDC"
#prefix name of the data files
filename<-"DATA-"
#the data logger creates files sequentially numbered
#the following parameters define the file numbers to analyze
#for example, files 377 through 380
fileNumStart<-3
fileNumFinish<-515

#to create an animated GIF of the spectral response, first download
#and install the "animation" package into R
#also install graphicsmagick, which is used to combine the png images into GIF
#within R enter the following command:
#saveGIF(source("train_vibration.r"), movie.name="c:\\train.gif", img.name="train", convert = "gm
convert", cmd.fun=system, clean=TRUE)
#this next line sets the pause interval between frames
#ani.options(interval=0.05)

#the following parameters affect the analysis algorithm

#block is a segment of data processed through the FFT algorithm
block<-1000

#overlap sets how much of the next block of data
#overlaps the previous block of data
overlap<-2     #defines the FFT overlap, 2=50%, 4=25%, etc

#sample rate of the data set as collected by the X2-1 data logger
SR<-50  #this is the sample rate

#defines the number of trailing FFT results to average when
#comparing to the noise level.  Large number increases signal-to-noise
#but introduces a longer time lag
fftTrailSize<-4

#number of initial FFT results to collect at the beginning of
#analysis (first file) to use in creating a noise floor as a
#basis for comparison
fftNoiseSize<-64
#count how many FFT results have been accumulated for noise calc
fftNoiseIncr<-0

#cutoffL and cutoffH define the frequency band to monitor
#everything outside the frequency band is removed
#in the case of a train, we area looking at the infrasonics between 0-20Hz
cutoffL<-1
cutoffH<-20

#define array to hold list of data files
dataFiles<-array(0)

#these lines will build a list using the path, directory, and file numbers defined
#at the beginning of the script
g<-as.numeric(fileNumStart)
n<-1
while(g<(as.numeric(fileNumFinish)+1)){
        #add leading zeros if need to make valid file name
        if(g<10)
```

```
                temp<-paste("00",as.character(g),sep="")
        else if(g<100)
                temp<-paste("0",as.character(g),sep="")
        else
                temp<-as.character(g)
        name<-paste(filename, temp, ".csv", sep="")
        path<-paste(drive, "\\", directory, "\\", name, sep="")
        dataFiles[n]<-path
        g<-g+1
        n<-n+1
}

#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. When a new file is opened,
#timeF is used to carry the last time value into the new file data
time<-0
timeF<-time

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

#store the total RMS Acceleration for the current FFT
vibRMS<-0

#build a history of total RMS acceleration for each FFT
vibrationHistory<-array(0)

#create filter factor table
#this loop creates a list of numbers used
#to filter the FFT output to just the frequencies
#within the cutoffL/cutoffH band
n<-1
filterFactors<-array(0,dim=c(block/2))
while(n<(block/2+1)){
        if((SR*n/block)>cutoffL){
                filterFactors[n]<-1
                #anything higher than cutoffH is multiplied
                #by zero and removed from output
                if((SR*n/block)>cutoffH){
                        filterFactors[n]<-0
                }
        #lower than cutoffH is removed
        }else{
                filterFactors[n]<-0
        }
        n<-n+1
}

#the next few lines of code build a Hann Window filter
#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
```

```
}

#initialize a matrix to hold the trailing results of the
#spectral energies
fftTrail<-matrix(0, nrow=block/2, ncol=fftTrailSize)

#this will hold the sum of the trailing data
fftTrailSum<-array(0, dim=c(block/2))

#fftAvg holds the avg spectral energy for the trailing ffts
fftAvg<-array(0, dim=c(block/2))

#array to hold the first FFT results used to
#determine the noise floor
fftNoiseTrail<-matrix(0, nrow=block/2, ncol=fftNoiseSize)
fftNoiseSum<-array(0, dim=c(block/2))
fftNoiseAvg<-array(0, dim=c(block/2))

#anything above the average is considered a noteworthy signal
vibrationSignal<-array(0, dim=c(block/2))

#freqarray saves the dominant frequency (ignoring DC to 0.1 hertz)
freqarray<-array(0)

#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))

#this loop will read the input file into three arrays
f<-1   #start at the second line, first line has output file name
while(f<(length(dataFiles)+1)){
       print(paste("Opening data file:  ", dataFiles[f]))
       data<-read.table(dataFiles[f], sep=",", comment=";")
       dataX<-c(data[[2]])
       dataY<-c(data[[3]])
       dataZ<-c(data[[4]])

       #X2-1 conversion
       #convert raw data into g's
       #in low gain, divide raw data by 6554
       #in high gain, divide raw data by 13108
       dataX<-(dataX)/6554
       dataY<-(dataY)/6554
       dataZ<-(dataZ)/6554

       #calc the rms
       data<-sqrt(dataX^2+dataY^2+dataZ^2)
       #or use the vertical z-axis only
       #data<-dataZ

       #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.  A trailing average is calculated.
       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
              data.mag.half<-data.mag[1:((length(data.mag))/2)]   #use first half of FFT

              #new variable name for now
              fftOutput<-data.mag.half

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

              #shift the trailing data and add in the new fft results
              n<-1
```

```
		while(n<(fftTrailSize)){
			fftTrail[,n]<-fftTrail[,n+1]
			n<-n+1
		}
		#add in the newest fft results
		fftTrail[,fftTrailSize]<-fftOutput

		#average the trailing data
		n<-1
		fftTrailSum<-0
		while(n<(fftTrailSize+1)){
			fftTrailSum<-fftTrailSum+fftTrail[,n]
			n<-n+1
		}
		fftAvg<-fftTrailSum/fftTrailSize

		#we need to calculate the typical noise floor
		#the start of the first file will be used to
		#determine the noise, fftNoiseSize defines how much
		#data is used to calc the noise floor
		if(fftNoiseIncr<fftNoiseSize){
			fftNoiseIncr<-fftNoiseIncr + 1
			n<-1
			fftNoiseSum<-0
			while(n<(fftNoiseSize+1)){
				fftNoiseSum<-fftNoiseSum+fftNoiseTrail[,n]
				n<-n+1
			}
			fftNoiseAvg<-fftNoiseSum/fftNoiseSize

			#shift the trailing data and add in the new fft results
			n<-1
			while(n<(fftNoiseSize)){
				fftNoiseTrail[,n]<-fftNoiseTrail[,n+1]
				n<-n+1
			}
			fftNoiseTrail[,fftNoiseSize]<-fftOutput
		}


		#the signal is anything above the average noise
		#use the filter array to isolate frequency band
		vibrationSignal<-(fftAvg-fftNoiseAvg)*filterFactors

		#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

		#this line plots 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)")
		if(vibRMS>0.1){
			text("TRAIN DETECTED!!!", x=70, y=200)
		}
		text(round(time, 1), x=70, y=50)
		text("elapsed minutes", x=70, y=10)

		#calculate the RMS energy
		vibRMS<-sqrt(sum(vibrationSignal^2))
		#add the new value to the history array
		vibrationHistory<-append(vibrationHistory, vibRMS)

		#save the time 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)
		#pause for 0.1 seconds so the plots don't flash by too quick
		#Sys.sleep(0.1)

		#uncomment the next line if you are building a animated GIF
```

```
                #ani.pause()
        }
        #update timeF to track time into the next data file
        timeF<-time

        #move to the next file
        f<-f+1  #increment f
}
#end of main loop

#combine the vibration history data and time array into one data array
vibrationVStime<-array(c(timearray, vibrationHistory), dim=c(length(timearray),2))

#plot history versus time
plot(vibrationVStime, type="l", tck=1, xlab="Time (minutes)", ylab="Total RMS Acceleration (g)",
ylim=c(0,1))

#write the results to the output file
#uncomment the next two lines to have the results stored to a file
#outputDataFile<-paste(drive,"\\","analysis_output_of_Files_", as.character(fileNumStart),"-",
#as.character(fileNumFinish),".txt", sep="")
#write.table(vibrationVStime, outputDataFile, sep="\t")
```

End of Document