LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

| **Technical Note** | **LIGO-T1600150-v1** | June 30 2016 |
|---|---|---|

# Progress Report 2: Acoustic Emissions in Metals

Danielle Frostig, Gabriele Vajente

*Distribution of this document:*

LIGO Scientific Collaboration

Draft

**California Institute of Technology**
**LIGO Project, MS 18-34**
**Pasadena, CA 91125**
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

**Massachusetts Institute of Technology**
**LIGO Project, Room NW17-161**
**Cambridge, MA 02139**
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

**LIGO Hanford Observatory**
**Route 10, Mile Marker 2**
**Richland, WA 99352**
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

**LIGO Livingston Observatory**
**19100 LIGO Lane**
**Livingston, LA 70754**
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

# 1 Motivation

## 1.1 LIGO and Maraging Steel Blades

The Large Interferometer Gravitational-Wave Observatory (LIGO) is a ground-based system of large-scale detectors designed to observe gravitational waves. The experiment uses an enhanced Michelson interferometer to measure the distance between test masses in order to detect minute ripples in space-time caused by gravitational waves. The Advanced LIGO detectors must be extremely sensitive in order to successfully detect gravitational waves. For example, at the low frequency end of the audio band, between 10 and 20 Hz, the horizontal motion of the test masses needs to be on the order $10^{-19} m/\sqrt{Hz}$ [1]. In order to detect this displacement–which is four orders of magnitude smaller than a proton–within a 4 km long detector arm, complex noise isolation systems are employed.

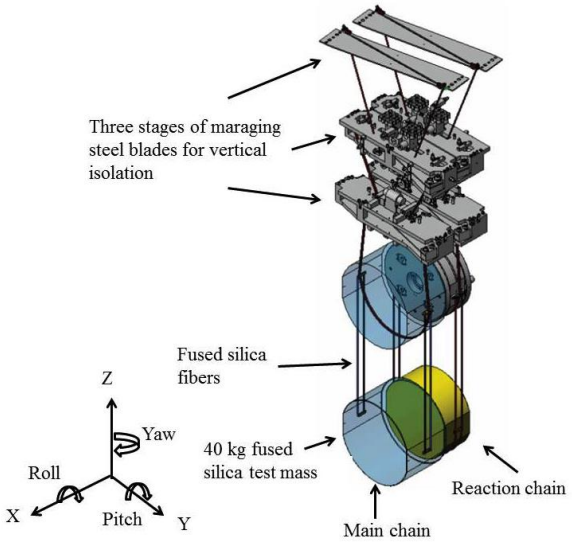


Figure 1: The Advanced LIGO suspension system, featuring a quadruple pendulum horizontal isolation system and three stages or maraging steel cantilever springs. The diagram is used courtesy of [7].

Local seismic activity is a prominent source of noise in the ground-based detectors. In order to isolate the test mass system from this noise, a quadruple pendulum isolates the system horizontally and three levels of maraging steel cantilever spring pairs isolate it vertically (Figure 1). Noise originating in the cantilevers can propagate throughout the system, especially in the upper intermediate stage (UIM), and cause vertical displacement in the system. Due to the curvature of the Earth, the test masses, hanging along the pull of Earth's gravity, must undergo corrections to be held truly parallel to each other [2]. Consequently, the "vertical" displacements become coupled with horizontal displacement and becomes a relevant source of noise in the horizontal direction that could potentially obscure gravitational wave signals.

### 1.2 Crackling Noise

Crackling noise results when a system subject to slowly changing external conditions responds via discrete crackling events due to a nonlinear conversion of energy [3]. This noise can occur in a variety of systems, from earthquakes on fault lines to stock market fluctuations [4]. However, not all systems respond to external force through crackling noise. The nonlinear conversion of energy in crackling events falls between the limits of a system responding to external forces in a single event–such as a piece of chalk snapping in half–and a system responding with small, similar sized events–such as popcorn popping. A simple example of crackling noise occurs when a sheet of paper is crumpled. Slowly changing external force (e.g. a hand crumpling the paper) causes the system (paper) to respond with a nonlinear conversion of energy (the crumpling sound heard) [4].

Ideally, systems could be assumed to behave elastically, meaning the resultant strain exhibited is proportional to the stress applied. However, non-linear deviations occur in systems loaded past yield stress. Consequently, metal deformations could occur through discrete releases of strain, and therefore cause crackling noise. There is a possibility of non-linear up-conversion of low frequency excitations (below 1 Hz) to high frequency (audio) noise in elastic metals such as maraging steel. Crackling noise has been detected in the plastic regime and so far there has been no direct measurement of mechanically up-converted noise in the elastic regime [1].

The goal of the investigation is to experimentally detect crackling noise in the maraging steel blades used in the Advanced LIGO experiment. The cantilevers used are in the elastic regime at about 50% of the yield stress. It is expected that low frequency motion in the suspension could result in a non-linear up-conversion of displacement noise. These events are further expected to be time-correlated with the stress or stress rate of the low frequency motion. Consequently, the project could detect not only crackling noise events, but could also find and increased rate of events with increased driving force or rate of force [1].

## 2 Problem

The goal of the project is to directly detect crackling events in maraging steel. Additionally, even if the experiment does not ultimately detect crackling events, it can help set upper limits for the amplitude and rate of such events. Two experiment types have been designed previously in order to detect crackling noise in maraging steel: one based on interferometry and one on ultrasonic Acoustic Emission (AE). Both experiments are being run simultaneously and this project focuses on the AE approach.

In the interferometric experiment, a pair of blade springs supporting optical instruments in a Michelson interferometer are driven by a low frequency force. If a discrete crackling event occurs, a resultant displacement should occur between the detected signals in each blade [1]. This experimental design is limited because the blade acts as a low-pass filter, which can obscure some of the noise. Additionally, in order to keep the optical system close the operating point, the maximum blade motion is limited to few tens of microns. A potential alternative is to use ultrasonic

AE microphones on blades loaded with a range of weights. Although the ultrasonic microphones are not as sensitive as the interferometer, they can be employed at higher frequencies to possibly directly detect single crackling events. Previously, this experiment was run with a variety of materials stressed with a range of weights and no crackling events were detected [5].

In the first set of tests, a maraging steel blade was loaded with an 11 kg mass, 50% of the maximum weight. An attached voice-coil actuator was driven with a DAC through an amplifier. Additionally, passive low-pass filters were used to prevent mistaking high-frequency events from the power amplifier for crackling events [5].

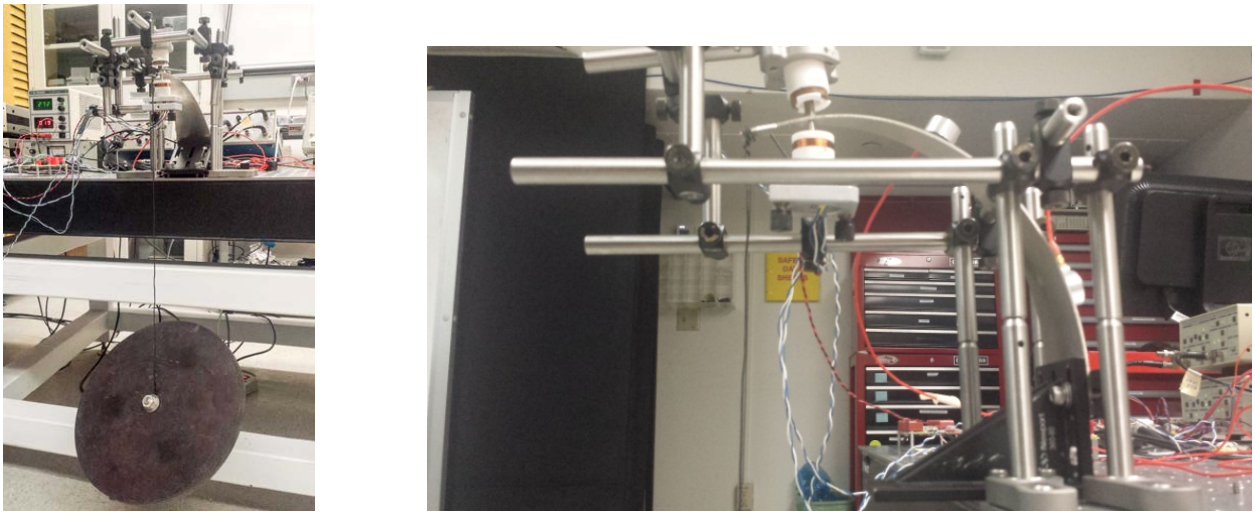

Figure 2: Maraging steel blade loaded with an 11 kg mass (right) and 16 kg mass (left).

The system was excited with a sinusoidal displacement of about 500 um at 0.4 Hertz every other hour, with the drive off for the alternate hours. No significant change was measured between the on and off periods, implying no crackling noise was excited by the low frequency drive [3].

Lack of evidence of crackling noise was similarly found in a blade loaded with 16 kg, 75% of the nominal yield stress, and clamped vertically. Despite days of data, no difference was detected between on and off periods. Similarly, the experiment yielded the same results when repeated with brass or high carbon steel blades. The results imply any crackling noise that could have occurred did so below the noise floor. An upper limit on noise of $10^{-15}\frac{m}{\sqrt{Hz}}$ from 30kHz to a few hundred kHz was set in maraging and high carbon steel blades [3].

The overall goal of this project is to further improve this experimental setup and continue the search for crackling noise. Ideally, this will include the calibration of the microphones to energy output, improvement of data analysis, testing variable amplitudes and rates on the drive, and possibly even finding evidence of crackling noise.

# 3 Progress

## 3.1 Microphone Calibration

Score Dunegan microphones [6] are used to collect the ultrasonic AE data for this experiment. These piezoelectric contact ultrasonic microphones can detect in-plane and out-of-plane waves in the range of hundreds of kilohertz [6]. They are secured directly onto the blades with an incompressible medium, typically wax. One goal of the project is to calibrate the microphones' output to the energy released in the blade, a more useful output for the sake of the experiment. The most straightforward way to calibrate the microphones is to compare their output with the known energy release of a simple system – in this case, dropping a steel ball onto the blade. By comparing the energy just before and after one bounce, the energy released into the blade can be approximated and used to calibrate the microphone.

### 3.1.1 Motion-Tracking

Figure 3: Early experimental setups for testing motion-tracking. A first iteration of the experiment (right) demonstrates the binder clip and coffee stirrer release mechanisms. A second iteration of the experiment (left) has a monochromatic background and demonstrates motion-tracking. The colors have been inverted for ease of viewing.

With the resources available, the most accurate and efficient way to measure the height of the ball after one bounce is to video record the experimental setup and employ motion-tracking software. A simple ball drop experiment was set up in an office consisting of a General Relativity textbook, a binder clip, and a pen spring, which was later replaced by small steel balls ranging from 3/64" to 1/4" in diameter (Figure 3a). Videos of the experiment were taken with a Samsung Galaxy and analyzed using *Tracker*, a free motion-tracking software.[1] The program could track a ball with relative success, but could not export the data in a useful format.

Consequently, MATLAB was employed for the motion-tracking and analysis. The script is based on Kalman filtering for predicting and tracking the motion of the object (See appendix). The code

[1] *Tracker* can be found at http://physlets.org/tracker/

was written based on the kalmanFilterForTracking[2] function in MATLAB and built from there.

The motion-tracking software is based on tracking the color of an object, which initially posed minor difficulty as the first experimental setup contained multiple objects that are silver in color. Consequently, a second ball drop experiment was set up with a completely black backdrop composed of a router and another spare computer part (Figure 3b). The MATLAB script ran with increased success with the new background.

### 3.1.2 Motion-Tracking Data Analysis

A Samsung Galaxy Note 5 was used for recording the motion of the ball. The slow motion video capture was recorded at 120 frames per second (fps) and a duplicate still frame was added in between each frame, for a total of 240 fps. Consequently, only every other video frame was used as to not repeat data points.

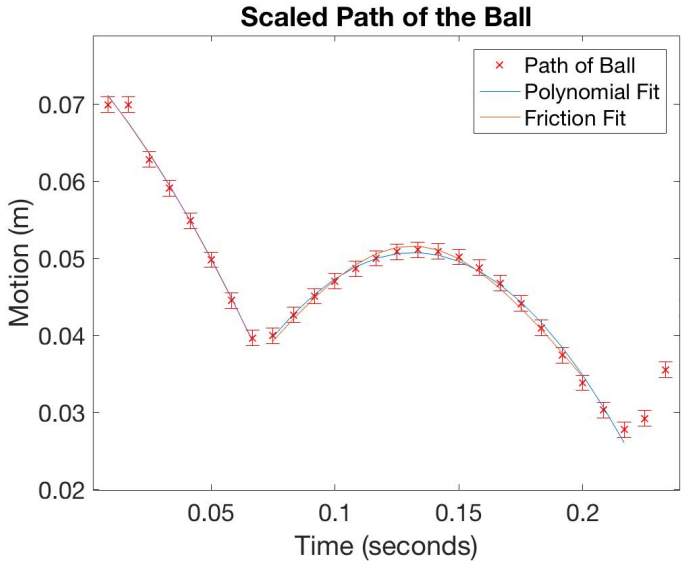


Figure 4: A sample output graph of the motion-tracking script for a ball drop on 7/20/16. The graph plots the path of the ball, a least-squares parabolic fit, and a least squares curve fit assuming friction in the air. The parabolic and friction fits are difficult to distinguish in this graph. There are also error bars on each data point assuming a measurement error of $\approx 1 \times 10^{-3} m$

In addition to collecting data for the position of the ball, velocity and acceleration were also analyzed (Figure 4, Figure 6, Figure 7). In order to analyze the data points, simple models were fit to the data. In the first, a constant acceleration in the y-direction, g, was assumed (Equation 1). Consequently, a linear and parabolic equations could be fit to the velocity and position graphs respectively (Equation 2, Equation 3).

[2]The function and documentation can be found at http://www.mathworks.com/help/vision/examples/using-kalman-filter-for-object-tracking.html

$$\frac{dv}{dt} = g \tag{1}$$

$$v(t) = k_1 x + k_2 \tag{2}$$

$$s(t) = k_1 x^2 + k_2 x + k_3 \tag{3}$$

Additionally, another set of fits was analyzed with the data assuming acceleration is not constant but rather slows due to friction in the air (Equation 4) [8]. The velocity and position solutions to this differential equation were then also fit to, and plotted with, the data (Equation 5, Equation 6) [3].

$$\frac{dv}{dt} = g - cv^2 \tag{4}$$

$$v(t) = \frac{\sqrt{g} \cdot tanh(\sqrt{g}\sqrt{c}(k+t))}{\sqrt{c}} \tag{5}$$

$$s(t) = \frac{log(cos(\sqrt{g}\sqrt{c}(k_1+t))}{c} + k_2 \tag{6}$$

The first bounce of each path was defined as when the velocity changed from negative to positive the first two times. The residuals from both the parabolic and friction fits were plotted for each trial. The residuals, and RMS values, for each fit were on the same order of magnitude and both exhibited clear patterns, indicating they might not be ideal fits (for example, Figure 5). However, for the scope of this experiment, both fits were considered sufficient and the friction fit was used for data analysis.

Energy released into the blade due to a ball's bounce was estimated with the kinetic energy lost by the ball during a bounce (Equation 7). The velocity right before and right after a bounce was estimated from the velocity fits with friction at the moment of the bounce, approximated as when velocity changes from negative to positive (Figure 7).

$$\Delta KE = \frac{1}{2} m (v_\circ^2 - v_f^2) \tag{7}$$

### 3.1.3 Calibration Experimental Design

Even though the initial experimental setup was effective for testing motion-tracking software, an improved setup was needed for the microphone calibration. An adjustable aluminum frame was built for dropping the ball from various heights and distances. A simple electromagnet - consisting of a battery, a switch, and a wire coiled around an iron screw - was placed at the end as a mechanism for dropping the magnetic steel balls (Figure 8). Additionally, a camera mount, a ruler for video calibration, and a black back drop for motion-tracking were added.

[3] Equations 5 and 6 were solved using *Mathematica*

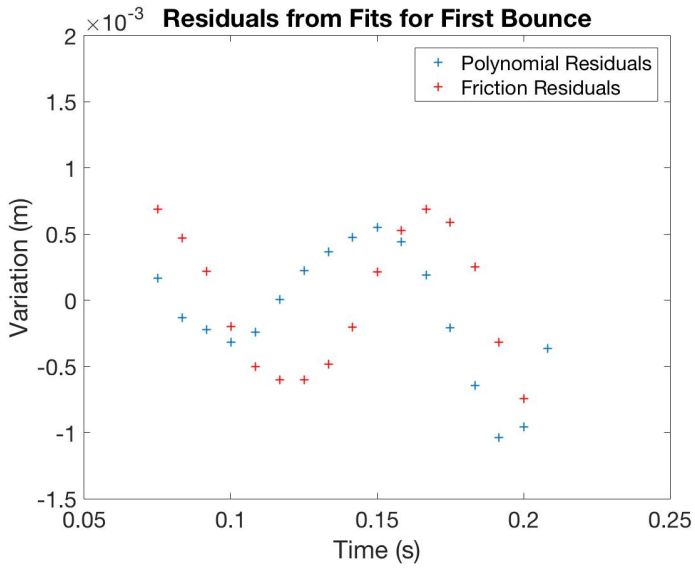


Figure 5: The residuals of the parabolic and friction-based fits for the first bounce of a sample trial taken on 7/20/16. Both fits are similar and exhibit a clear pattern, indicating that they are not ideal fits. The friction fit has a slightly lower RMS of $2.33\times10^{-4}$m versus a RMS of $5.39\times10^{-4}$m for the parabolic fit.

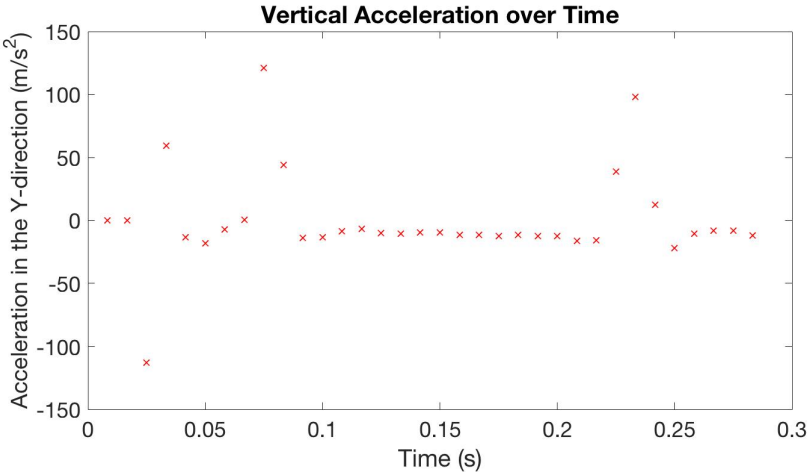


Figure 6: Acceleration over time for the sample video taken on 7/20/16. The graph exhibits a somewhat constant set of values between bounces. For this trial, the average acceleration between 0.1 and 0.2 seconds is -10.12 $\pm$28.79$\frac{m}{s^2}\frac{m}{s^2}$, a 3.31 percent error from the acceleration due to gravity in Los Angeles: -9.796 $\frac{m}{s^2}$.

### 3.1.4 Electronic Setup and Data Acquisition

The maximum sampling rate Cymac computer used for data acquisition in the microphone calibration experiment is 64kHz. This limits the analysis of the microphone output to the 30kHz Nyquist frequency. In order to counteract this limitation, a few techniques are used to down-convert the high-frequency signals in the microphone output.

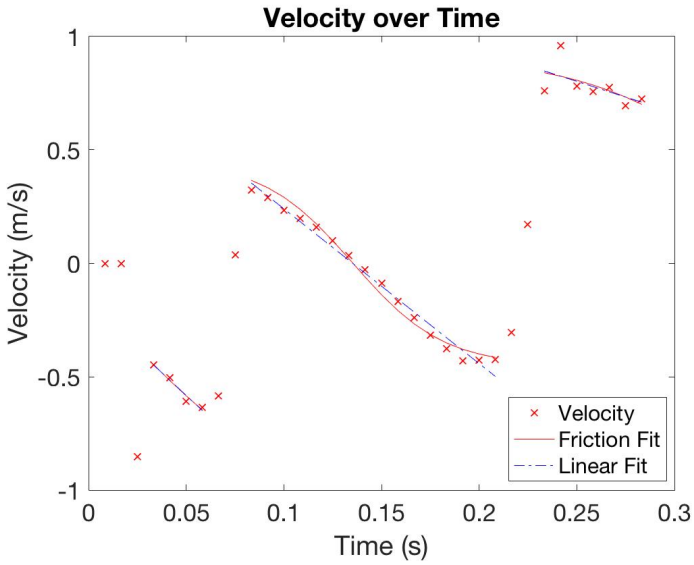


Figure 7: Velocity over time for a sample trial taken on 7/20/16. The initial drop and each subsequent bounce are plotted with both a linear fit and a fit which tries to account for friction. The change in velocity during each bounce allows for the calculation of kinetic energy lost in each bounce.

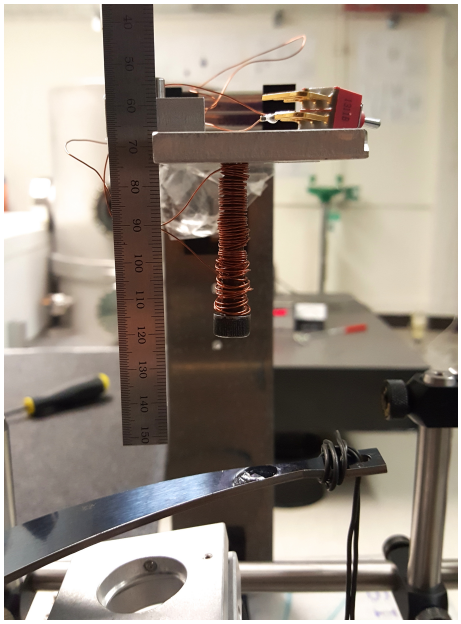

Figure 8: The electromagnet release mechanism, featuring a battery, switch, wire coiled around an iron screw, and a ball at the end, ready for release. A small ruler provides a scale in the plane of the ball to help with distance calibrations when analyzing videos.

A 5MHz quartz Local Oscillator (L.O.) was used to output a square wave. The system was powered by about 18 V from a DC power supply and two counters (five count and two count) were used to obtain a 100kHz wave from the 5MHz L.O. Each microphone signal was passed through a ferrite bead to prevent the addition of extraneous signals and amplified 10 times by a preamplifier. The signal was then derived from two AESmart302A conditioners, which produced a broadband

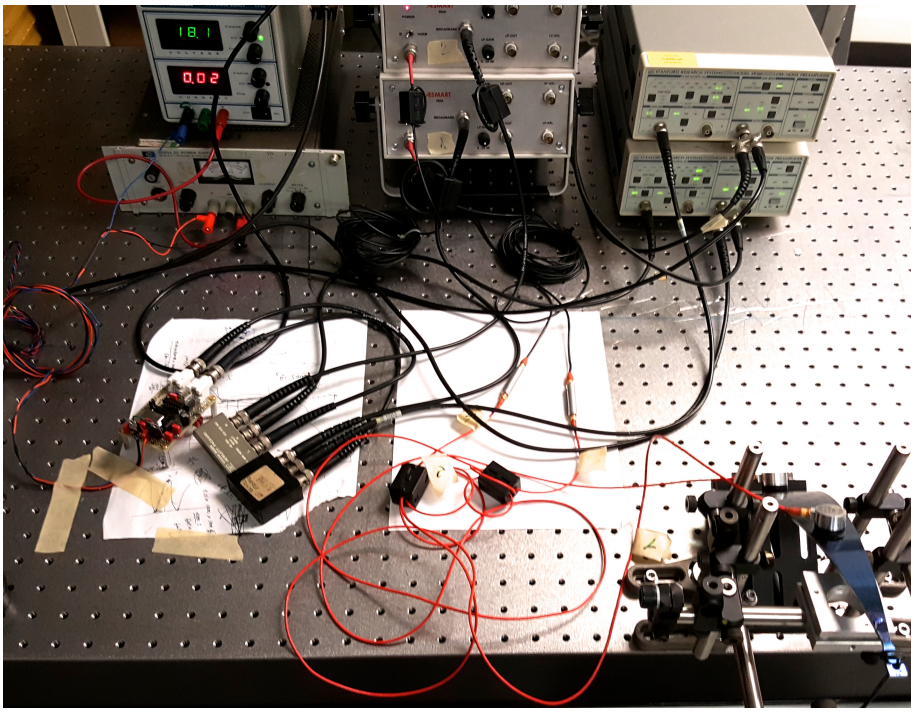

Figure 9: The electronic setup for the microphone calibration design. From left to right, in the upper row there are two DC power supplies, two conditioners, and two preamplifiers. In the middle row there is the quartz Local Oscillator, two mixers, two ferrite beads, and two preamlifiers. In the bottom right corner is the loaded blade with two microphones mounted with wax.

output.

Two HP1053A mixers were then used to multiply the broadband output from the microphones with the output of the L.O. This is done in order to down-convert higher frequency bands to the requisite baseband: 0-30kHz. Because the 100kHz wave is a square wave with odd harmonics at different amplitudes, the mixer can down-convert multiple frequency bands to the baseband, which can then be sampled by the computer. The multiplied signal is sent to a Stanford Research Systems low-noise preamplifier to be low-pass filtered (0-100kHz) and amplified 100 times.

The amplified signal was then sent from the amplifier to the CYMAC computer and an oscilloscope, in order to analyze and visually check the output data respectively. The CYMAC computer is a real time digital system with ADC and DAC 16 bit channels.

The digital signal can be viewed with dataviewer. We are interested in the two demodulated outputs of the microphones, sampled at 65kHz, and the RMS of each signal, saved at 4kHz and low pass filtered. The signal is recorded in units of volts and the RMS in units of volts squared. The RMS is useful for analyzing electrical power and will ultimately be used to analyze the microphones' relationship with the energy input in the blade.

The data was plotted using ligoDV in MATLAB. Initial tests found that a ball dropping on the blade over-saturated the microphones. Consequently, for the sake of the calibration experiment, the preamplifier was set to a lower amplification and a correction factor will be added to the cal-

culations. Tests found a good balance between maximizing sensitivity and avoiding saturation occurred at 10 times amplification, versus the original 100 times amplification.

### 3.1.5 Data Analysis

The output of the microphones was analyzed in the LIGO DataViewer (ligoDV) app in MATLAB. For each microphone, the voltage over time and the root mean squared (RMS) of the voltage over time were collected and plotted (Figure 10, Figure 11). The area under the curve of each peak of the RMS data was extracted for correlation analysis.

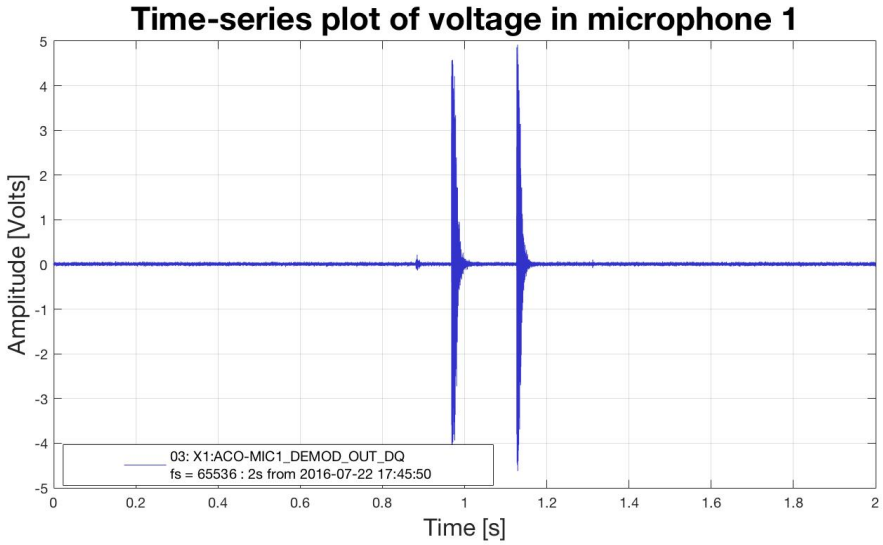


Figure 10: A plot of voltage measured in microphone one ovber time for a sample ball drop in which the ball bounced off the blade twice.

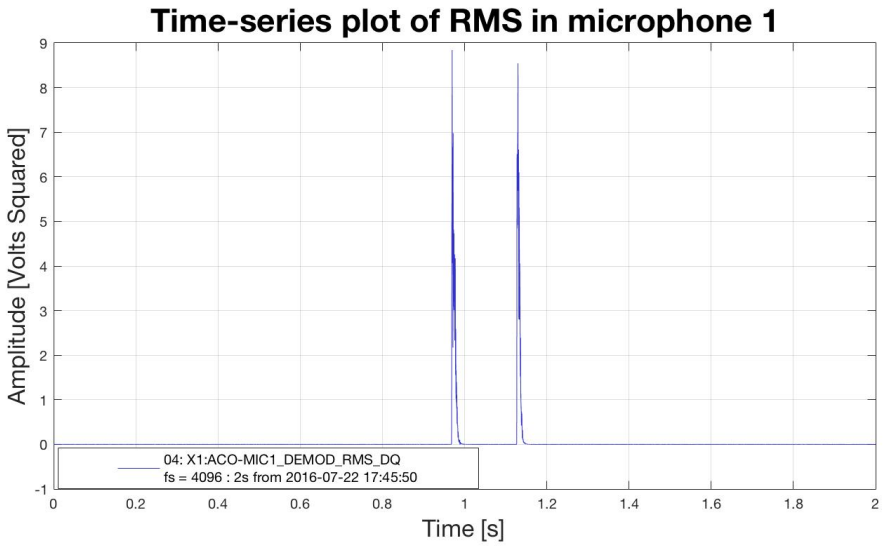


Figure 11: A plot of the RMS of voltage measured in microphone one over time for a sample ball drop in which the ball bounced off the blade twice.

# 4 Looking Forward

The most immediate next goal of the project is to finish setting up the microphone calibration experiment. Right now, I need to figure out a way to obtain useful data that is not saturated. To do this I need to find the right balance between a size of the ball and the height of the drop so that there is a low amount of energy released, but a long enough drop to obtain enough data points for analysis. Then, I can run the experiment many times from various heights, locations on the blade, and with balls of different weights. Then, I will figure out how to correlate the microphone output with the approximate energy released in the blade. Throughout this process it will be important to note if the data seems to make sense (e.g. does the distance between the ball and the microphone correlate to the output) and to obtain an estimate for uncertainty in the calibration.

Once the microphone is calibrated, I will rerun the previous experiment to repeat the results and then aim to improve the experiment. For example, the preexisting experiment only compares output for when the drive is on or off. This does not test for the possible relationship between the amplitude of the force or the derivative of force and the rate and size of crackling events. To analyze this, I could vary external force and analyze the resultant rate of variations in the data or changes in the noise floor. There are two possible ways to approach this. The first is to modify an algorithm used in the interferometer experiment [1] in order to find noise modulated coherently with the already running low frequency drive. The second is to look for small events in the microphone data and correlate them with external force. This approach could be done using LIGO software designed to detect transient gravitational waves. Ideally,I will test both approaches and based on the success of each, decide to mainly pursue either approach, both, or another solution entirely. However, there will most likely not be time to do everything. The ultimate goal of the project is to detect discrete crackling events but the project will still be successful even just by furthering the experiment via the above steps and by finding an upper limit on crackling noise.

There are four weeks left of LIGO SURF. Below is a tentative schedule of the weeks that have transpired and the weeks to come.

| Week | Task |
|---|---|
| 6/13 | Orientations, safety training, paperwork, and lectures. |
| 6/20 | Setting up motion-tracking and initial ball drop experiment. |
| 6/27 | Finish setting up microphone calibration experiment. |
| 7/4 | Further develop MATLAB script for motion-tracking. |
| 7/11 | Trip to Livingston. Troubleshoot code and experimental problems. |
| 7/18 | Begin running calibration experiment and do initial data analysis. |
| 7/25 | Finish calibration experiment. |
| 8/1 | Running experiment with variable drive forces and frequencies. |
| 8/8 | Data analysis with crackle experiment algorithm and continuous running of experiment. |
| 8/15 | Data analysis with LIGO software and continuous running of experiment. |

Table 1: A tentative schedule for the weeks to come.

# 5 Appendix

Below is the MATLAB code used for motion-tracking.

```
% Clear previous data and figures.
clc;
close all;
clear all;

% Open video player
videoReader = vision.VideoFileReader('ball28.mp4');

% Initialize MATLAB functions to video
videoPlayer = vision.VideoPlayer('Position',[100,100,500,400]);
foregroundDetector = vision.ForegroundDetector(...
    'NumTrainingFrames',10,'InitialVariance',0.05);
blobAnalyzer = vision.BlobAnalysis(...
    'AreaOutputPort',false,'MinimumBlobArea',70);

% Set empty matrices for later
location = [];
detect = [];
acc = [];
vel = [];

% Set the pixel to m conversion
scale = 8.184/48000;
```

```
24
25 for i=1:200
26 kalmanFilter = []; isTrackInitialized = false;
27    while ¬isDone(videoReader)
28      % Set first frame as background
29      colorImage  = step(videoReader);
30      % Define foreground
31      foregroundMask = step(foregroundDetector, rgb2gray(colorImage));
32      % Define an object as variant from the first frame
33      detectedLocation = step(blobAnalyzer,foregroundMask);
34      isObjectDetected = size(detectedLocation, 1) > 0;
35
36      if ¬isTrackInitialized
37        % Configure a Kalman filter if an object is detected for first time
38        % and initialize tracking.
39        if isObjectDetected
40            kalmanFilter = configureKalmanFilter( ...
41                'ConstantAcceleration',detectedLocation(1,:), ...
42                [1 1 1]*1e5, [25, 10, 10], 25);
43          isTrackInitialized = true;
44        end
45        label = ''; circle = zeros(0,3);
46      else
47        % If an object is detected again, track location with Kalman
48        % filtered predictions.
49        if isObjectDetected
50          predict(kalmanFilter);
51          trackedLocation = correct(kalmanFilter, detectedLocation(1,:));
52          % Place this location in a matrix
53          location(i) = scale*(480 + (-1*trackedLocation(2)));
54          i = i+1;
55          label = 'Corrected';
56        else
57          % If there is no object detected after an object has been seen
58          % Predict the location and store in the location matrix.
59          trackedLocation = predict(kalmanFilter);
60          location(i) = scale*(480 + (-1*trackedLocation(2)));
61          i = i+1;
62          label = 'Predicted';
63        end
64        circle = [trackedLocation, 5];
65      end
66
67      colorImage = insertObjectAnnotation(colorImage,'circle',...
68          circle,label,'Color','red');
69      step(videoPlayer,colorImage);
70
71    end
72 end
73
74 % Take every other frame and set time.
75 half=location(2:2:end);
76 len = length(half);
77 lenb= [1:len];
78 time = lenb/120;
79
```

```
80
81 % Calculate velocity (simple)
82 for i=2:len
83     vel(i) = (half(i)-half(i-1))/(1/120);
84 end
85
86 % Calculate acceleration
87 for i=2:len
88     acc(i) = (vel(i)-vel(i-1))/(1/120);
89 end
90
91 % Isolate first bounce
92
93 % Assume whole clip is first bounce
94 start = 1;
95 term = len;
96
97
98 % Set start and end frames for first bounce
99 for j=1:len-1
100     % If velocity changes signs, set start frame.
101    if vel(j) < 0 & vel(j+1)>0;
102       start = j+1;
103       for i=j+2:len-1
104           % If it changes signs again, set end frame.
105           if vel(i) < 0 & vel(i+1)>0;
106               term = i;
107               break
108           end
109       end
110       break
111    end
112 end
113
114
115 % Fit to velocity
116 fun = @(c,time)(sqrt(9.796)*tan(sqrt(c(1))*sqrt(9.796)*c(2) ...
117     + sqrt(c(1))*sqrt(9.796)*time))/(-sqrt(c(1)));
118 c = [0.00625,1];
119 vel_fit = lsqcurvefit(fun,c,time(start+1:term-1),vel(start+1:term-1));
120
121 % Fit to velocity initial drop
122 funf = @(f,time)(sqrt(9.796)*tan(sqrt(f(1))*sqrt(9.796)*f(2) ...
123     + sqrt(f(1))*sqrt(9.796)*time))/(-sqrt(f(1)));
124 f = [0.00625,1];
125 velf_fit = lsqcurvefit(funf,f,time(2:start-1),vel(2:start-1));
126
127 % Fit to velocity second bounce
128 funh = @(f,time)(sqrt(9.796)*tan(sqrt(f(1))*sqrt(9.796)*f(2) ...
129     + sqrt(f(1))*sqrt(9.796)*time))/(-sqrt(f(1)));
130 h = [0.00625,1];
131 velh_fit = lsqcurvefit(funh,h,time(term:end),vel(term:end));
132
133 % Calculate velocity with air resistance
134 for i=start:term
135     vel2(i) = (sqrt(9.796)*tanh(sqrt(vel_fit(1))*sqrt(9.796)*vel_fit(2)...
```

```
        + sqrt(vel_fit(1))*sqrt(9.796)*time(i)))/(-sqrt(vel_fit(1)));
end


% Recalulate position
for i=start+1:term-2
    pos(i) = trapz(time(start:i),vel2(start:i))+half(start);
end

% Fit to position for first bounce
d = [0.00625,-0.1179,1];
fund = @(d,time)(log(cos(sqrt(d(1))*sqrt(9.796)*(d(2)+time)))/d(1)+d(3));
pos_fit = lsqcurvefit(fund,d,time(start:term-2),half(start:term-2))

% Fit to position for initial fall
e = [-1,-0.1179,0];
fune = @(e,time)(log(cos(sqrt(e(1))*sqrt(9.796)*(e(2)+time)))/e(1)+e(3));
pose_fit = lsqcurvefit(fune,e,time(1:start-1),half(1:start-1))

%fun(vel_fit,time(start))
%funf(velf_fit,time(start))


% Plot velocity over time and fits
figure;
plot(time,vel,'rx',time(start+1:term-1), ...
    fun(vel_fit,time(start+1:term-1)),'r',time(1:start), ...
    funf(velf_fit,time(1:start)),time(term:end), ...
    funh(velh_fit,time(term:end)))
% Initial drop
Ac0 = polyfit(time(2:start-2),vel(2:start-2),1);
yfit0 = Ac0(1)*time(2:start-2)+Ac0(2);
yfit0_all = Ac0(1)*time+Ac0(2);
% First bounce
Ac = polyfit(time(start:term-1),vel(start:term-1),1);
yfit = Ac(1)*time(start:term-1)+Ac(2);
yfit_all = Ac(1)*time+Ac(2);
%Second Bounce
Ac2 = polyfit(time(term+2:end),vel(term+2:end),1);
yfit2 = Ac2(1)*time(term+2:end)+Ac2(2);
yfit2_all = Ac2(1)*time+Ac2(2);
hold on;
plot(time(start:term-1),yfit,'b-.', ...
    time(term+2:end),yfit2,time(2:start-2),yfit0);
title('Velocity over Time');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
set(gca,'FontSize',18);

% Find velocity values right befroe and after first bounce
a = yfit0_all(start-1)
b = yfit_all(start-1)
% Find velocity values right befroe and after second bounce
n = yfit_all(term)
m = yfit2_all(term)

```

```
% Find a best fit parabola for the first bounce and find the maximum.
[p1, S, mu] = polyfit(time(start:term), half(start:term) , 2);
[y3,Δ] = polyval(p1, time(start:term),S,mu);
mean(Δ);
time_max = -.5*(p1(2)/p1(1));
y1_max = polyval(p1,time_max);

% Find a best fit parabola for the drop and find the maximum.
p4 = polyfit(time(1:start-1), half(1:start-1) , 2);
y4 = polyval(p4, time(1:start-1));
time_max1 = -.5*(p4(2)/p4(1));
y4_max = polyval(p1,time_max1);

% Plot motion
err(1:len) = 0.001;
figure;
errorbar(time,half, err, 'rx')
hold on
plot(time(start:term),y3,time(start:term-2),fund(pos_fit, ...
    time(start:term-2)),time(1:start-1),fune(pose_fit,time(1:start-1)), ...
    time(1:start-1),y4);
title('Scaled Path of the Ball')
xlabel('Time (seconds)');
ylabel('Motion in the Y-Direction (m)');
set(gca,'FontSize',18);
legend('Path of Ball','Polynomial Fit', 'Friction Fit');

% Plot residuals from fit
res3 = half(start:term) - y3;
res4 = half(start:term-2) - fund(pos_fit,time(start:term-2));
figure, plot(time(start:term),res3,'+',time(start:term-2),res4,'r+')
title('Residuals from Fits for First Bounce');
xlabel('Time (s)');
ylabel('Variation (m)');
legend('Polynomial Residuals','Friction Residuals');
set(gca,'FontSize',18);

% Find RMS on residuals
rms3 = sqrt(sum(res3.^2)/length(res3));
rms4 = sqrt(sum(res4.^2)/length(res4));

% Percent error
perc3 = (res3.*100)./half(start:term);
perc4 = (res4.*100)./half(start:term-2);

% Plot acceleration over time
figure;
plot(time,acc,'rx')
%Acel = polyfit(time(start+4:term-4),acc(start+4:term-4),1);
%yfit1 = Acel(1)*time(start+4:term-4)+Acel(2);
%hold on;
%plot(time(start+4:term-4),yfit1,'b-.');
title('Acceleration over Time');
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
%legend('Data', 'y= -8.9624x + 1.0123');
```

```
248 set(gca,'FontSize',18);
249
250 mn   = mean(acc(start+2:term-4));
```

# References

[1] Vajente, G., et al. ”An Instrument to Measure Non Linear Mechanical Noise in Metals in the Elastic Regime.” (2016): LIGO Laboratory, California Institute of Technology, Pasadena. Submitted to Review of Scientific Instruments.

[2] Weinstein, Alan. ”Intro to LIGO.” June 2016.

[3] Paoletti, Federico, Vajente, Gabriele. ”Acoustic Ultrasonic Measurements to Investigate Crackling Noise in Maraging Steel Blade Springs.” LIGO-T1500510-v1 (2015)

[4] James P. Sethna, Karin A. Dahmen, Christopher R. Myers, ”Crackling Noise.” Nature, 410, 242-250, 8 March 2001.

[5] Paoletti, Federico, Vajente, Gabriele, Ni, Xiaoyue. ”Acoustic Emissions in Maraging Steel Blades in the Elastic Regime.” (2015)

[6] ”SE9125-M AE Sensors.” Score Atlanta Inc, n.d. Web. 12 May 2016. https://score-atlanta.com/products/AE%20Sensors/SE9125-M

[7] Ni, Xiaoyue, et al. ”Crackling Noise in Gravitational Wave Detectors.” (2016): LIGO Laboratory, California Institute of Technology, Pasadena. LIGO RD. https://www.ligo.caltech.edu/LA/page/research-development

[8] Taylor, John R. ”Classical Mechanics.” (2005): University of Colorado. 58-60.