Simon Popecki

ME 747

Lab 4

PART 1: POTENTIOMETER ACCELEROMETER

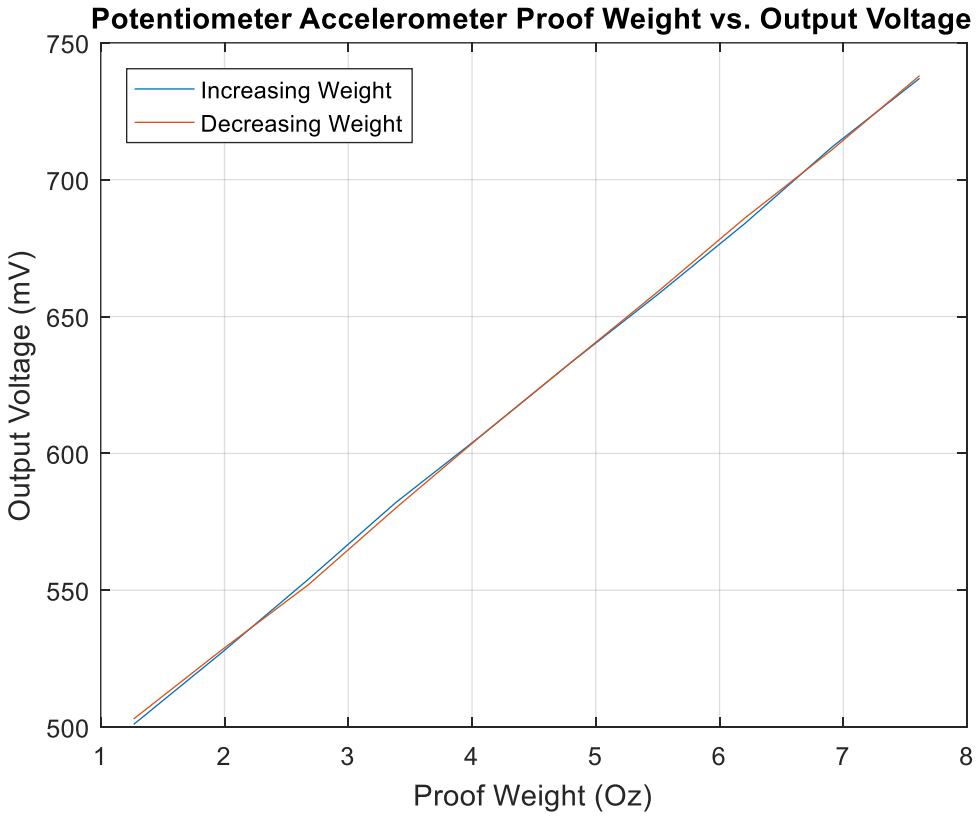

Figure 1

Figure 1 shows the calibration plot of the potentiometer accelerometer. Proof weights were added whilst output voltage was recorded, the results are linear. Measurement noise is about equal to hysteresis error for this unit. The sensitivity of this sensor was found to be .0372 V/Oz. For computations, the mean value of the increasing and decreasing weights was used.

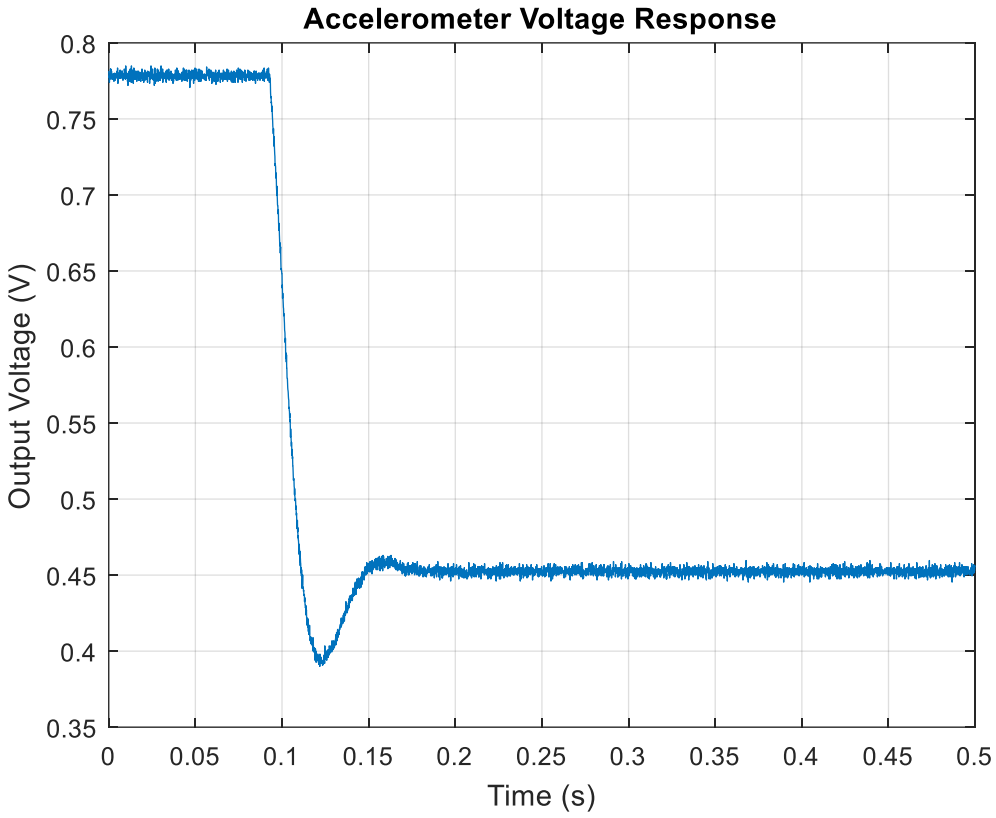

Figure 2

Figure 2 shows the change in accelerometer voltage with respect to time in response to a perturbation. The system is second order with the following parameters:

- Spring Constant: 69.5 Oz/in
- Natural Frequency: 126.8 rad/s
- Effective Mass: .00320 Slugs
- Damping Ratio: .57

The sensitivity of the accelerometer was found to be: .0001606 $V*s^2/in$. The maximum acceleration this instrument can theoretically measure is 4,845 in/s^2, which is roughly 12.5 Gs. The bode plot of the output in voltage for an acceleration input is shown in Figure 3 on the next page. Sinusoidal inputs at frequencies above 10^2 radians per second will suffer severe attenuation, and should not be measured with this instrument.

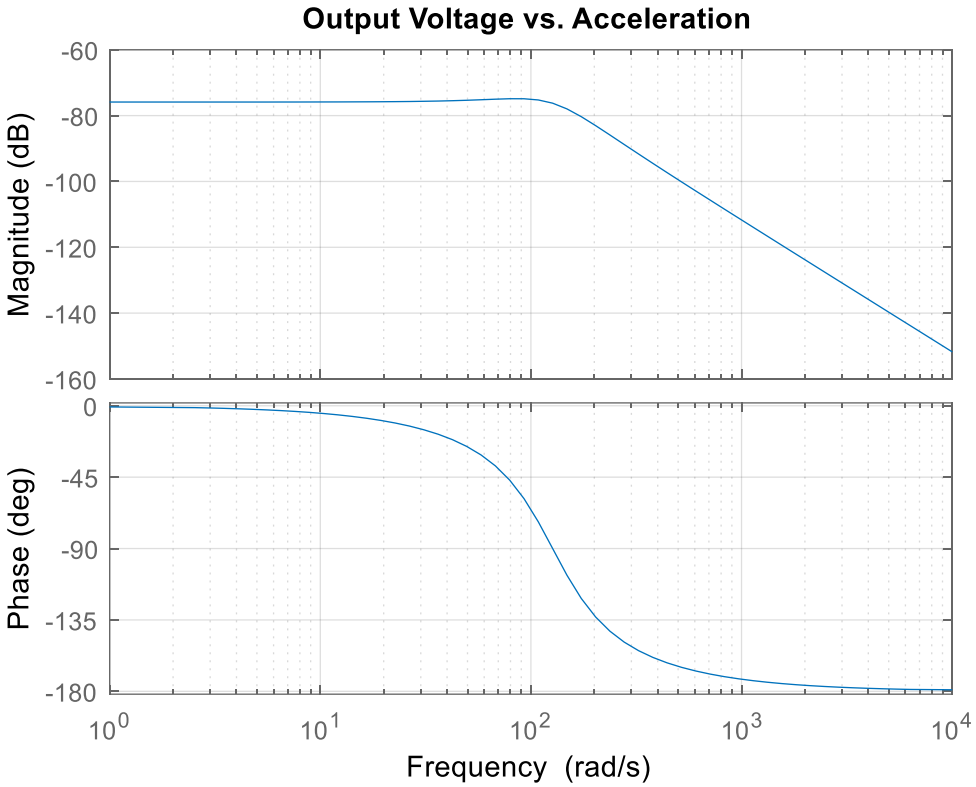

Figure 4

PIEZOELECTRIC FORCE SENSOR

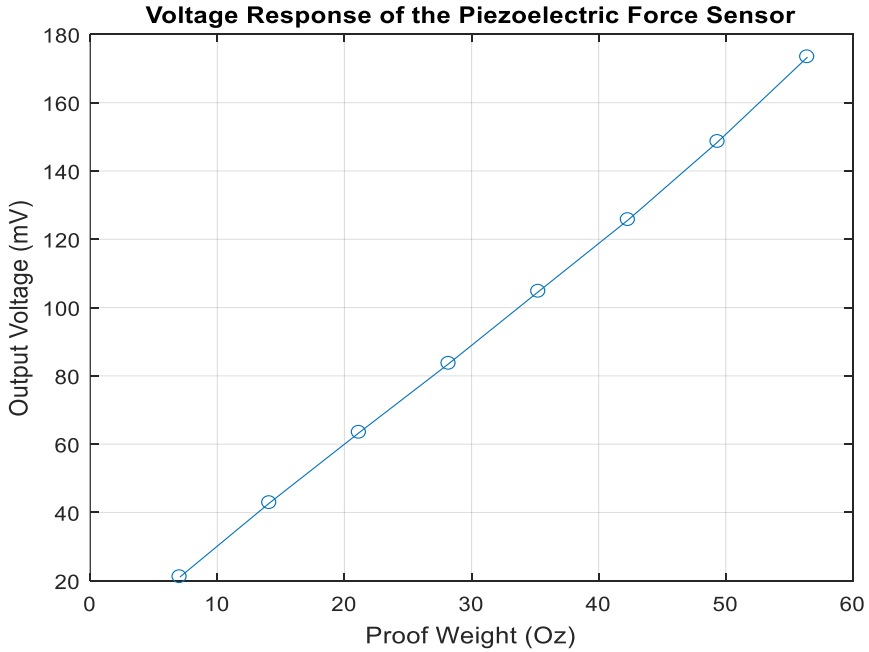

Figure 3

Figure 4 on the previous page shows the calibration curve of the piezoelectric force sensor. The sensitivity of this sensor was found to be: .0030 V/Oz. The full scale error percentage was 1.67%.

The decay time constant was calculated to be 10.6 seconds. This sensor would not be a good choice for measuring steady state forces, as steady state typically implies that a measurement must be held for longer than a few seconds. A ten second time constant implies that only a fraction of a second yields an accurate measurement (i.e. within a few percent). If the time constant was several orders of magnitude larger, this sensor would be a better choice for steady state measurements.

The natural frequency of the system/structure was calculated to be 10,406 rad/s, making this system vulnerable to vibrations in the immediate 1.66 kHz range. These frequencies should be avoided, as they will cause the system to resonate. This sensor is still a better choice for high frequencies compared to the potentiometer accelerometer.

IMPULSE LOADING AND VIBRATION

The parameters for the foam mass system are listed below:

- Spring constant: 81.7 lb/in
- Natural Frequency: 125.6 rad/s
- Damping Ratio: .2284
- Damping Coefficient: .2971 lb*s/in

Governing Differential Equation:

$$F(t) = \frac{\frac{1}{k}}{\frac{1}{\omega_n^2} + \frac{2\xi}{\omega_n} + 1}$$

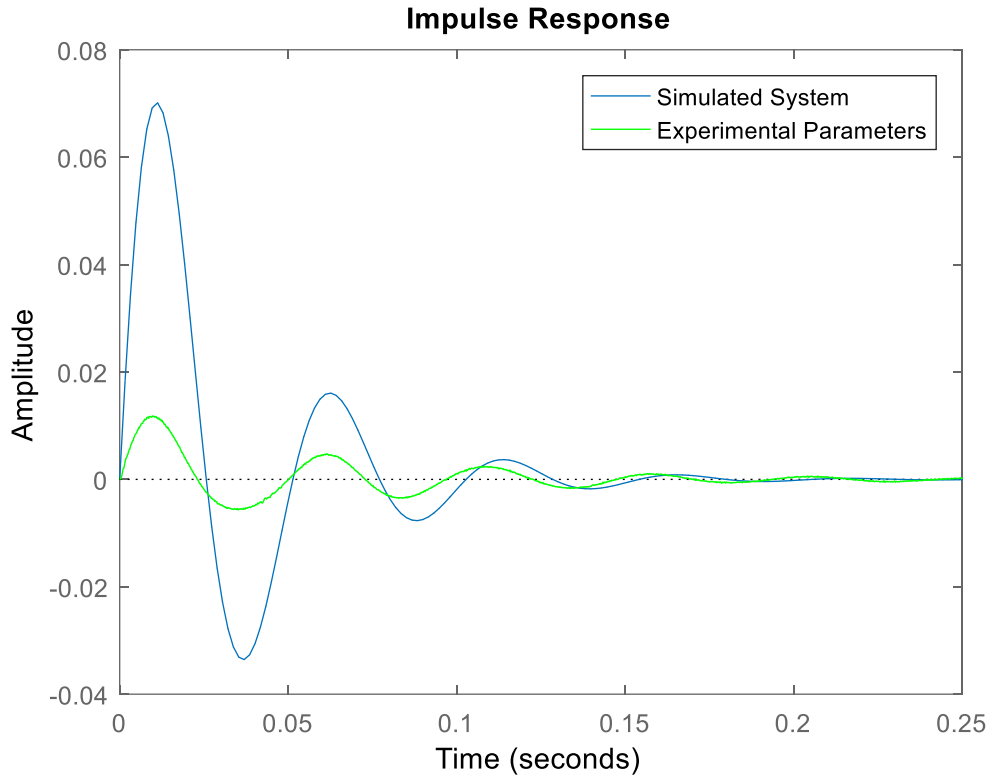

Figure 5

Figure 5 shows a plot of the simulated and experimental impulse responses. The peaks on the experimental curve tended to be shifted more to the left compared to the simulated value. The differences in amplitude can be ignored, and is a factor of adjusting system gain to make both curves fit nicely on the same figure. Aside from the minor skew, the simulated and experimental systems match fairly well. It is noted that it is impossible to replicate an impulse waveform in real life, because there is always a duration of time that will pass as the sensor is struck with an object – true impulse exerts a force over an infinitesimally small duration of time.

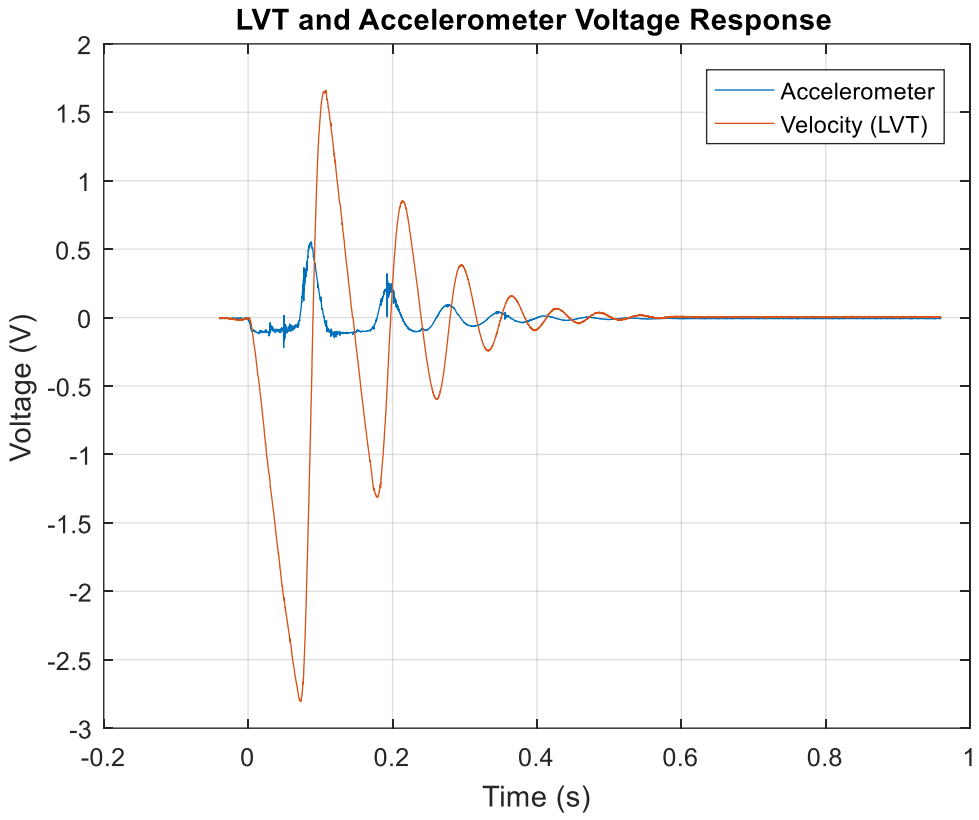

Figure 6

Figure 6 shows both the LVT and accelerometer voltage output for the duration of time where a weight was dropped onto a foam pad. The sensitivity of the accelerometer was calculated to be: .0002813 $V*s^2/in$, the sensitivity of the LVT was calculated to be .1181 V*s/in.

Figure 7 on the next page shows the raw results of integrated accelerometer data. The integration removed a lot of the noise from the signal, however there is a clear linear skew in the upwards direction. This was corrected for, and a secondary figure (Figure 8) shows the corrected results. The integrated signal is nearly the same, however it is flipped about the X-axis – this is caused by the gain of the inverting amplifier being -1. The issue of skew occur in the second lab as well.

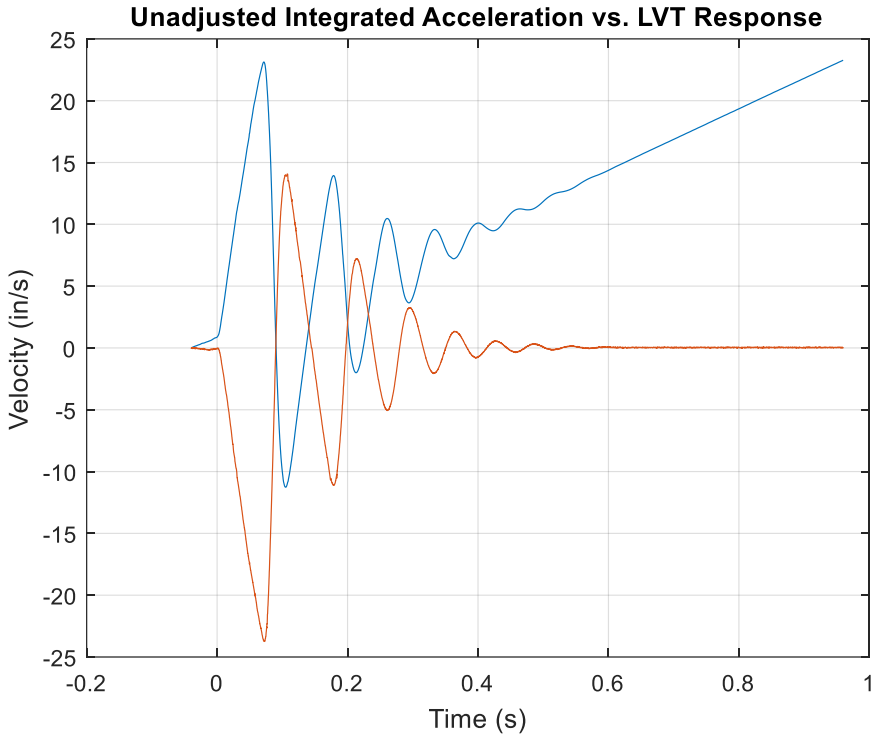

Figure 7

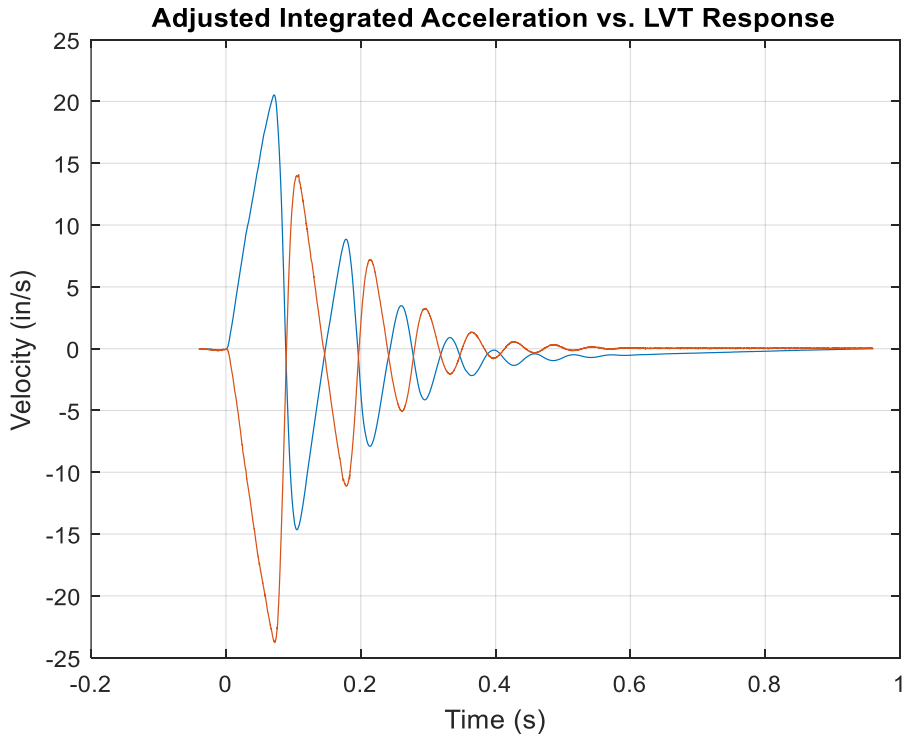

Figure 8

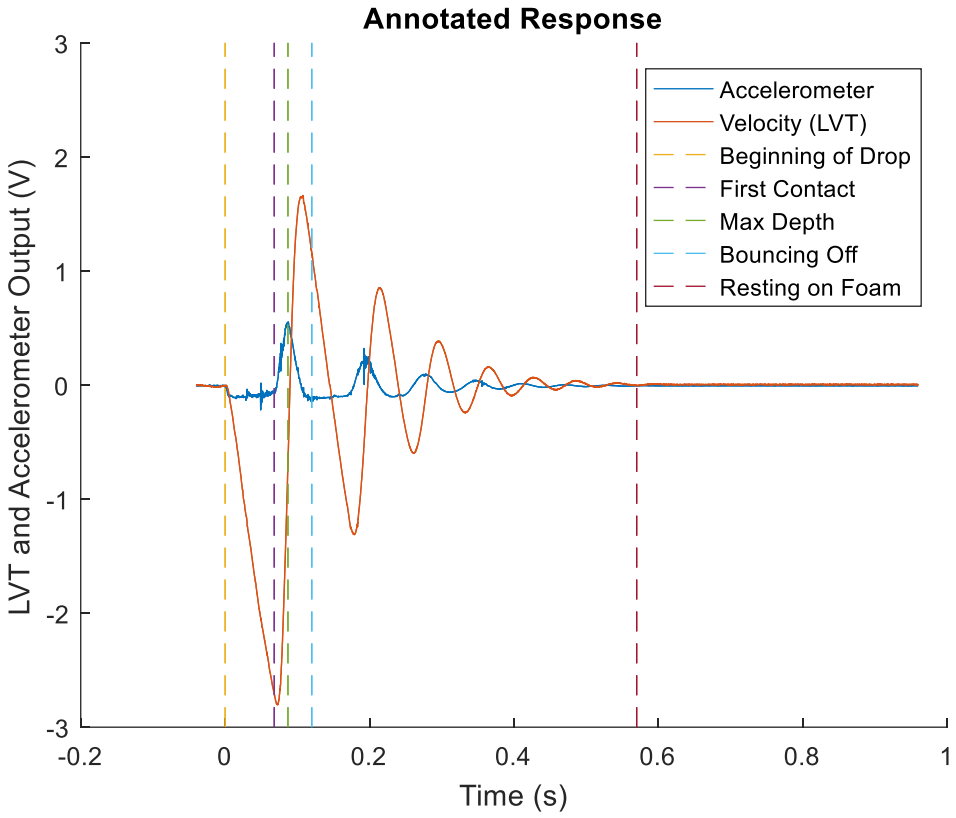

Figure 9

Figure 9 shows the voltage response from Figure 6, annotated to show what is going on at important points in time in the foam mass system. The vertical lines and corresponding legend entries show times of interest.

The maximum velocity of the core was calculated to be: 23.7 in/s.

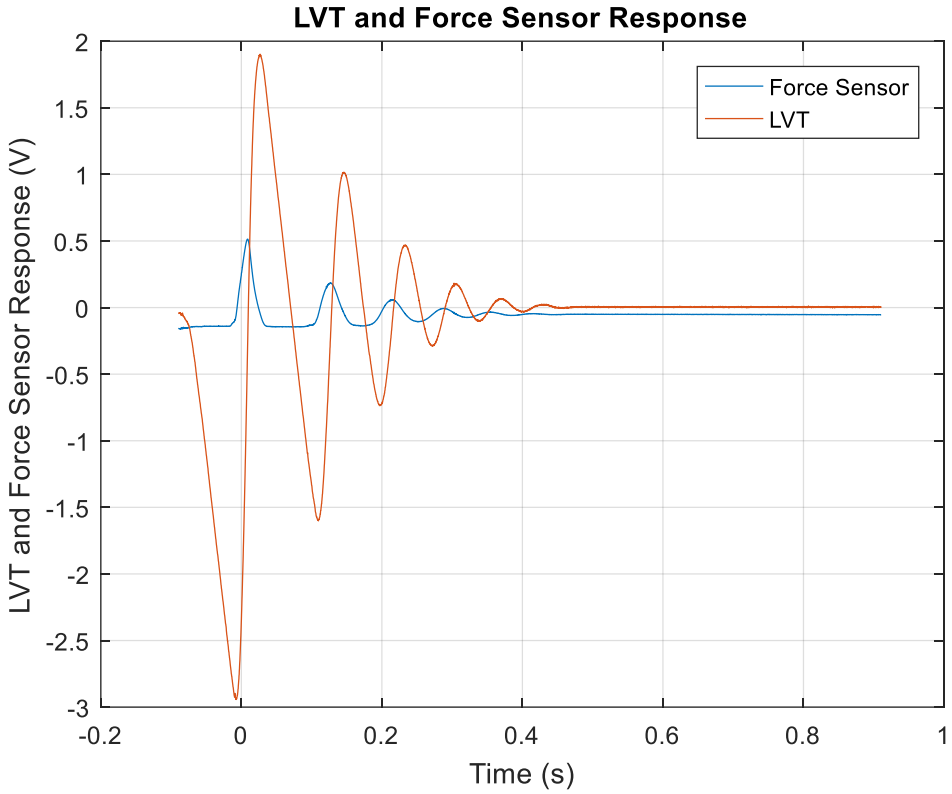

Figure 10

Figure 10 shows the output of a force sensor and LVT with respect to time. Note that when force rises from zero, the LVT shows a peak on the other side of the X-axis. The force on the foam at maximum core velocity was calculated to be .216 lbs. The steady state force between the foam and the core was found to be .2129 lbs. The total mass of the core with attached sensors was calculated to be .0066 slugs.

```
%Simon Popecki
%ME 747
%13 Novermber 2017
%Lab 4
clear all;
close all;


%% Part 1 Potentiometer Accelerometer
load lab4data.mat
mass_nobucket = [20 40 60 80 100 120 140 160 180 200]; %grams
bucketWeight = 15.9; %grams
testmass = mass_nobucket+bucketWeight; %testmass (g) is the total mass on the
sensor
testmassoz = testmass*0.035274; %converting from grams to ounces because
Prof. Fussell likes imperial
eoinc = [501 527 554 582 607 633 658 684 712 737]; %mV, output voltage with
weight increasing
eodec = [503 528 552 580 607 633  659 686 711 738]; %mV, output voltage with
weight decreasing
eoavg = (eoinc+eodec)/2; %averageing the output voltages of increasing and
decreasing values
bestfit = polyfit(testmassoz,eoavg,1); %making a best fit line to find
sensitivity

sensitivity = bestfit(1)/1000; %the sensitivity of the scale in volts per
ounce

% Plotting eo vs. weight
figure(1)
plot(testmassoz,eoinc,testmassoz,eodec)
title('Potentiometer Accelerometer Proof Weight vs. Output Voltage')
xlabel('Proof Weight (Oz)')
ylabel('Output Voltage (mV)')
legend('Increasing Weight','Decreasing Weight','location','northwest')
grid on

% Plotting eo vs. time
figure(2)
plot(time,eout)
title('Accelerometer Voltage Response')
xlabel('Time (s)')
ylabel('Output Voltage (V)')
grid on

bottompeak = .39; %V, from manual inspection of figure 2 - the lowest peak
finalvalue = .4528; %V, from manual inspection of figure 2 - the ending value
percentovershoot = finalvalue/bottompeak; %finding the percent overshoot for
use with the table on canvas
zeta = .52; %from the table, estimated value
peaktime = .029; %s, the time to the peak - obtained visually from plot
omegan = pi/(peaktime*(sqrt(1-(zeta^2)))); %rad/s, natural frequency
w = ((.454-.392)/sensitivity); %Oz, equivalent weight
m = w*0.00194256; %slugs, effective mass
k = (16/12)*(omegan^2)*m; %oz/in, the spring constant - !highly dependant on
peak time (16/12 for converison from slugs to Oz/in)
```

```
b = 2*zeta/omegan*k; %damping ratio, unknown imperial units with ounces and
inches

accelsensitivity = ((m*16/12)*sensitivity); %((Oz*s*s)/ft)*(V/Oz) ->
V/(in/(s*s)) divided by 12 to convert from feet to inches, 16 to convert to
Oz
maxaccel = (.7781/accelsensitivity); %in/(s*s), maximum sensor acceleration
(is about 12 and a half Gs) (.7781 is resting voltage)

system = tf(accelsensitivity,[(m*16/12)/k b/k 1]);
figure(3);
bode(system);
title('Output Voltage vs. Acceleration')
grid on

%% Part 2.1 Piezoelectric Force Sensor

%calibration data
%everything is called -two now because it's part two
masstwo = [.2 .4 .6 .8 1 1.2 1.4 1.6]; %kg
weighttwo = (masstwo*2.20462)*16; %converting from kilograms to Earth Oz
eouttwo = [21.1 42.8 63.4 83.6 104.7 125.68 148.56 173.35]; %mV

figure(4)
plot(weighttwo,eouttwo,'-o')
title('Voltage Response of the Piezoelectric Force Sensor')
xlabel('Proof Weight (Oz)')
ylabel('Output Voltage (mV)')
grid on

bestfittwo = polyfit(weighttwo,eouttwo,1); %creating a linear best fit line
to determine sensitivity
sensitivitytwo = bestfittwo(1); %mV/Oz, the slope is the sensitivity
sensitivitytwo = sensitivitytwo/1000; %V/Oz, converting from millivolts to
volts per ounce

calibrationline = sensitivitytwo*weighttwo+(bestfittwo(2)/1000);
maxdifference = max((eouttwo/1000)-calibrationline)/(calibrationline(end));
%maximum difference determines the error
fserror = maxdifference*100; %full scale error percent

load lab4data.mat

smdvolt = smooth(decayvoltage);
smdvolt = smooth(smdvolt); % !warning! more advanced smoothing techniques
require too much computational power

vstart = .11; %from visual inspection of figure 5
tstart = .06291; %s, start time corresponding to vstart

vtau = .632*(smdvolt(end)-vstart)+vstart; %finding the time constant with the
63.2% method

for i = length(decaytime):-1:1
```

```
    if smdvolt(i) > vtau %finding the point where voltage exceeds vtau,
working in backwards direction
        tauindex = i;
        break;
    end
end

ttau = decaytime(tauindex); %s, finding the point in time of tau

figure(5)
plot(decaytime,smdvolt,ttau,vtau,'o')
title('Piezo Decay Test Plot')
xlabel('Time (s)')
ylabel('Output Voltage (V)')
legend('Voltage Response','First Time Constant')
grid on

figure(6)
plot(omegat,omegav)


deltat = .001277-.0006732; %time difference between points, from inspection
of figure 6 - second peaks
dampednaturalfrequency = 1/deltat*2*pi; %rad/s

%% Part 2.2 Piezoelectric Force Sensor

load lab4data.mat
avgv = mean(dvoltage(1:200)); %finding an average voltage to adjust for shift
dvoltage = dvoltage-avgv; %subtracting the difference from the voltage array
timeint = dtime-.6134; %setting the time such that the peturbation starts at
t = 0 s

[~,~,maxvoltage,~] = peakdet(dvoltage,0.002);
Td = diff(timeint(maxvoltage));
Tdselect = Td(1); %picking first value

%log decrement method
n = 2; %two peak separation
sigman = (1/(n-1))*log(.0144/.003298); %peak numbers picked from visual
inspection
zetan = sigman/sqrt((4*pi^2)+(sigman^2));

wdn = 2*pi/Tdselect; %rad/s, finding damped natural frequency
wnn = wdn/sqrt(1-(zetan^2)); %rad/s, converting to undamped with zetan

mn = 2/32.17; %slugs, mass of the plate
kn = ((wnn^2)*mn)/12; %lb/in, spring constant
K = 1/(kn*16); %gain K
bn = 2*zetan/wnn*kn; %damping coefficient, should be lb*s/in

system2 = tf(K, [1/(wnn^2) 2*zetan/wnn 1]);

figure(7)
```

```
impulse(system2)
hold on
plot(timeint,dvoltage/1000/sensitivitytwo,'g')
legend('Simulated System','Experimental Parameters')

figure(8)
plot(timeint,dvoltage)
title('test')
xlabel('Time (s)')
ylabel('Voltage (V)')
grid on

%% Part 3 Vibration Analysis
load lab4data.mat
acceltime = LVTaccel(:,1);
acceltime = acceltime+.04929;
accel = LVTaccel(:,2);
LVT = LVTaccel(:,4); %measures velocity

gvolts = -.1086; %V, from gravity, the acceleration during freefall, picked
from a stable point on the plot
accelerometersensitivity = gvolts/386.09; %V/(in/s^2), the sensitivity of the
acceleromter

%using method suggested by lab partner
startindex = 0;
endindex = 0;
for i = 1:length(acceltime)
    if (acceltime(i)>0.029 && endindex==0)
        endindex = i;
        break;
    elseif (acceltime(i)>0.0064 && startindex==0)
        startindex = i;
    end
end

timeint = acceltime(endindex)-acceltime(startindex);
dvelocity = timeint*386.09; %in/s
LVTinterval = LVT(endindex)-LVT(startindex);
LVTsensitivity = abs(LVTinterval/dvelocity); %LVT sensitivity V/(in/s)

figure(9)
plot(acceltime,accel,acceltime,LVT)%,acceltime(startindex),accel(startindex),
'o',acceltime(endindex),accel(endindex),'o')
legend('Accelerometer','Velocity (LVT)')
title('LVT and Accelerometer Voltage Response')
ylabel('Voltage (V)')
xlabel('Time (s)')
grid on

%part b
integratedacceleration =
cumtrapz(acceltime,(accel/accelerometersensitivity));

figure(10)
```

```
plot(acceltime,integratedacceleration,acceltime,LVT/LVTsensitivity)
title('Unadjusted Integrated Acceleration vs. LVT Response')
ylabel('Velocity (in/s)')
xlabel('Time (s)')
grid on

%normalizationslope = (integratedacceleration(end)-
integratedacceleration(1))/(acceltime(end)-acceltime(1)); %slope of the
skewed curve
%Normalizing
deltaY = integratedacceleration(end)-integratedacceleration(1); %finding the
total vertical skew
biasRange = linspace(0,deltaY,length(integratedacceleration)); %creating an
array of deltaYs
biasRange = biasRange'; %turning row vector into column vector
normia = integratedacceleration-biasRange; %subtracting bias from the
original data

figure(11)
plot(acceltime,normia,acceltime,LVT/LVTsensitivity)
title('Adjusted Integrated Acceleration vs. LVT Response')
ylabel('Velocity (in/s)')
xlabel('Time (s)')
grid on

%part c

figure(12)
plot(acceltime,accel,acceltime,LVT,[0 0],[-3 3],'--',[0.068 0.068],[-3 3],'--
',[0.087 0.087],[-3 3],'--',[0.12 0.12],[-3 3],'--',[0.57 0.57],[-3 3],'--')
title('Annotated Response')
xlabel('Time (s)')
ylabel('LVT and Accelerometer Output (V)')
legend('Accelerometer','Velocity (LVT)','Beginning of Drop','First
Contact','Max Depth','Bouncing Off','Resting on Foam')

%part d
%calculate maximum core velocity
corevelocity = abs(LVT/LVTsensitivity);
maxcorev = max(corevelocity); %in/s

%part e

forcesensitivity = 0.491; %V/lb

[~,ind] = max(abs(ch1)); %finding the index of maximum value on ch1 (force,
but in volts)
fmax = ((abs(ch0(ind)-ch0(1)))/forcesensitivity); %lbs, maximum force in
array

fsteady = (abs(ch0(end)-ch0(1)))/forcesensitivity; %lbs, force at rest/steady
state
masspd = fsteady/32.17; %slugs, mass of the core

figure(13)
```

```
plot(ftime,ch0,ftime,ch1)
title('LVT and Force Sensor Response')
xlabel('Time (s)')
ylabel('LVT and Force Sensor Response (V)')
grid on
legend('Force Sensor','LVT')
```