Homework #1: Instrument responses and deconvolution **Due: 5:00 PM 01/30/17**

Please read the following questions carefully and make sure to answer the problems completely. In your MATLAB script(s), please include the problem numbers with your answers. Use the *Publish* function in MATLAB to publish your script to a *pdf* document; find something similar if not using MATLAB. LaTeXwith *mcode.sty* or some other language will work. For more on the *Publish* functionality within MATLAB see http://www.mathworks.com/help/matlab/matlab_prog/publishing-matlab-code.html. Upload your *pdf* file to Blackboard under Assignment #1. Your filename should be *GEOPH677_HW1_Lastname.pdf*. Hint: You can achieve this automatically by calling your MATLAB script *GEOPH677_HW1_Lastname.m*.

Intended Learning Outcomes

Students will be able to:

1. Derive the impulse response of an electromagnetic seismograph
2. Use poles/zeros and other instrument information to create and plot the phase and amplitude spectra of a seismograph
3. Relate the impulse response of an instrument to the data is measures and the actual ground displacement
4. Use MATLAB code from Matt Haney to create a stable and causal inverse filter for instrument deconvolution

Part 1: An electromagnetic velocity sensor *(35 pts.)*

Section 12.3.2 in Aki and Richards (2002) explains how an electromagnetic velocity sensor (Figure 1) works and derives the total damping given by this type of system (e.g. eqn 12.49). Starting from the beginning of this section complete or answer the following:

- Explain the Biot-Savart law. *(5 pts.)*

  When there is no time-dependence, Maxwell's equations give the Biot-Savart law, which describes the magnetic field (B) generated by an electric current (I) and vice versa.

- Explain Onsager's reciprocal theorem. *(5 pts.)*

  Onsager's reciprocal theorem links different types of forces to different flows of energy. For example, in the case below we say that $F_R = G_m I$, where F_R is the mechanical force linked to electron flow I via the coefficient G_m. In the other direction we have $V = G_e \dot{z}(t)$, which is the electromotive force linked to the particle velocity via the coefficient G_e. I use G_m here to represent the coefficient for the mechanical force and G_e to represent the coefficient for the electromotive force. Onsager's reciprocal theorem states that $G_m = G_e$. These types of coefficients are found throughout the physical world. To me it is not so obvious that force and flows of different physical properties are linked via simple coefficients when you go from one force to another and vice versa for flows. Onsager first came up with this when looking at thermal and electrical systems, but these ideas hold for mechanical systems as well.

- How would you compute l given a coil with radius r and number of coils n. *(5 pts.)*

  The total length of a wire l with n coils and radius r is the sum individual coils. One coil length is $2\pi r$, the circumference. Therefore, the total length of the wire is $l = 2\pi rn$.

- Derive equation 12.49 (show all steps). *(15 pts.)*

  As in the first example of the vertically hanging mass with the dash-pot, we have a damped harmonic oscillator. We need to start by balancing the forces. The difference in the electromagnetic sensor is that the dash-pot is replaced by the internal resistance of the recorder, which is the terminal that is shunted by resistance R. The balanced force equation looks like

$$M\ddot{z}(t) + F_R + kz(t) = -M\ddot{u}(t). \tag{1}$$

  In order to derive equation 12.49 we need to determine F_R. To match the derivation done in class with the dash-pot, we need to find F_R in terms of $\dot{z}(t)$. To do this we start with the Biot-Savart law

$$F_R = IlB, \tag{2}$$

  where I is the current in the wire of length l generated by the magnetic-flux density B. We can introduce a velocity ($\dot{z}(t)$) term if we consider the mechanical power *consumed* by the resistance. You can think of this last statement as the mechanical power converted to electrical power. In order to make this connection, we need two powers. First, the mechanical power is expressed as

$$F_R\dot{z}(t) = IlB\dot{z}(t). \tag{3}$$

  Second, the electrical power is expressed as

$$IV = IlB\dot{z}(t), \tag{4}$$

  where V is voltage and it is apparent that $V = lB\dot{z}(t)$.

  Here is where the authors make an interesting observation. They note that if they write $V = lB\dot{z}(t) = G\dot{z}(t)$ (where $G = lB$), they can also write $F_R = IlB = IG$. Think about this. Is it obvious that something moving at $\dot{z}(t)$ would generate a voltage determined by lB and that some current I would generate a mechanical force F_R determined by that same lB? This is an example of Onsager's reciprocal theorem.

  From here we want to find F_R in terms of $\dot{z}(t)$, G, and the total resistance of the system $R_t = R + R_o$, where R is the resistance of the shunt and R_o is the resistance of the coil. From Ohm's law we know that

$$V = G\dot{z}(t) = IR_t \rightarrow I = \frac{G\dot{z}(t)}{R + R_o}. \tag{5}$$

  Now we can insert the current into the F_R equation so that

$$F_R = GI = G\frac{G\dot{z}(t)}{R + R_o} = \frac{G^2\dot{z}(t)}{R + R_o}. \tag{6}$$

  We can plug this into our original balanced force equation

$$M\ddot{z}(t) + \frac{G^2\dot{z}(t)}{R + R_o} + kz(t) = -M\ddot{u}(t) \tag{7}$$

  and rearrange to arrive to arrive equation 12.47.

$$\ddot{z}(t) + w_o^2 z(t) = -\ddot{u}(t) - \frac{1}{M}\frac{G^2\dot{z}(t)}{R + R_o}. \tag{8}$$

  Thinking back to class, we rewrote the damping coefficient in terms of ϵ. We can do that here and we get

$$2\epsilon = \frac{1}{M}\frac{G^2\dot{z}(t)}{R + R_o}. \tag{9}$$

The final part of this problem is to recognize that we could have two damping forces, one from the dash-pot and one from the electromagnetic coil. To do this we add another force term to our balanced equation

$$M\ddot{z}(t) + F_R + F_d + kz(t) = -M\ddot{u}(t), \tag{10}$$

where F_d represents the dash-pot force. Combining these two damping forces and writing in terms of *epsilon* we would get

$$\epsilon = \epsilon_o + \frac{1}{2M}\frac{G^2\dot{z}(t)}{R + R_o}, \tag{11}$$

where ϵ_o is that from the dash-pot.

Note: Zongbo noted that you can do this derivation with Faraday's law and skip all this Onsager reciprocal theory stuff. Just think about the magnet or the coil moving at $\dot{z}(t)$.

- When was the electromagnetic sensor introduced in seismology and who introduced it? *(5 pts.)*

  End of Section 12.3.2 on page 628 (Aki and Richards, 2002) says that Galitzin introduced the electromagnetic seismograph in 1914.

Note: If you need a copy of Aki and Richards come talk to me. Some of the answers to the above questions can be found with a search on Google.

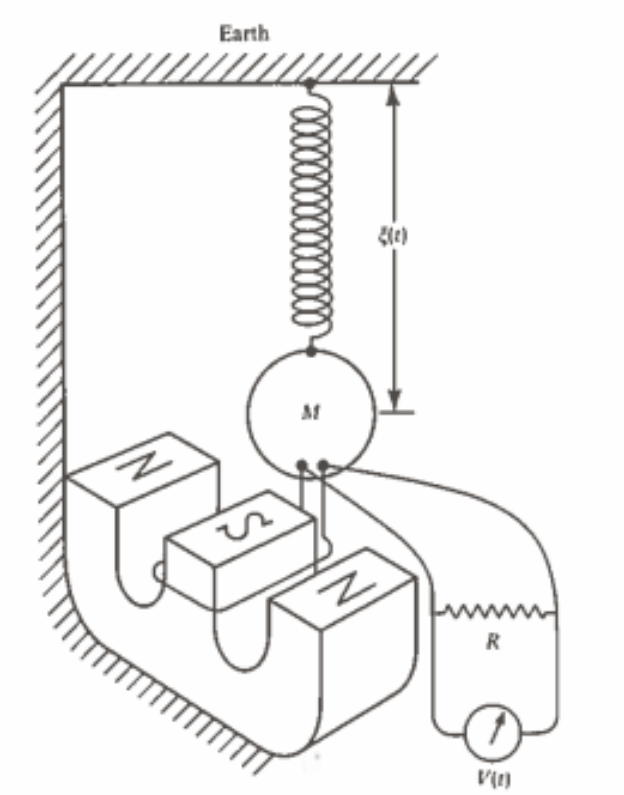

Figure 1: An electromagnetic seismometer – replace $\zeta(t)$ with $z(t)$ to follow our derivations in class.

Part 2: Creating the frequency response *(30 pts.)*

In the *Data/* folder on the GIT repository you will find a miniseed file.

XX.BSM7.HHZ_MC-PH1_0426_20170112_180000.miniseed

Load this into a computer however you like. You will need to access the header information to determine the sampling rate of the digital time series.

```
w = waveform('DATA/XX.BSM7.HHZ_MC-PH1_0426_20170112_180000.miniseed','seed');

% in case there are gaps in the time series (marked by NaN) we can interpolate a
% meaningful value
w = fillgaps(w, 'interp');

% demean the time series - this removes any DC shift
w = demean(w);
% detrend the time series - this removes linear drift
w = detrend(w);

plot(w,'xunit','date'); grid on;
h = gcf;
set( findall( h, '-property', 'FontSize' ), 'FontSize', 18 );
set( findall( h, '-property', 'FontName' ), 'FontName', 'Helvetica' );
set( findall( h, '-property', 'FontWeight' ), 'FontWeight', 'Bold' );

set( h, 'Position', [100 100 1400 500] );
set( h, 'PaperPositionMode', 'auto' );
print( h, '-dpng', 'figs/raw.png');
```

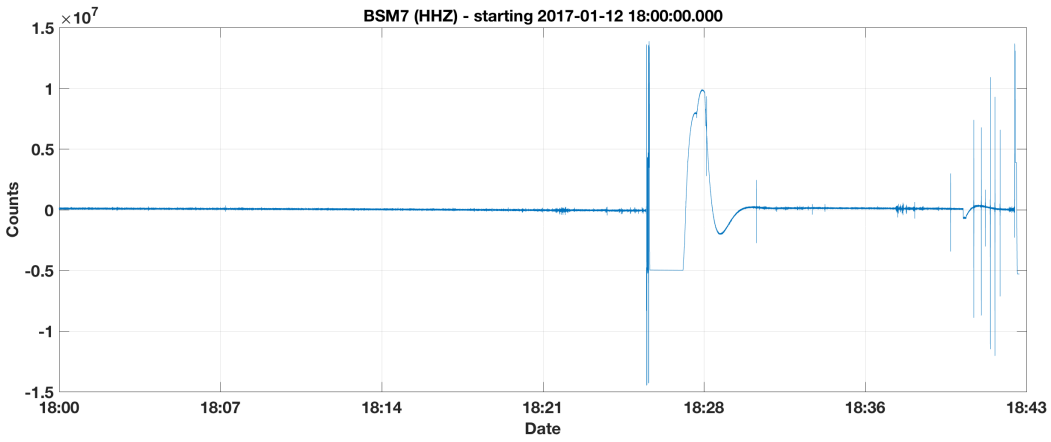

Figure 2: Seismic data after demean and detrend.

```
% Stomp test for meridians: 2017-01-12
%
% Alex: 18:41:10
% Tom: 18:41:30
% Diego: 18:41:40
% Zongbo: 18:41:54
% Matt: 18:42:07
% Rebekah: 18:42:21

% start time for extraction
year   = 2017;
month  = 01;
```

```
day    = 12;
hour   = 18;
minute = 40;
second = 0.0;

t1 = datenum( year, month, day, hour, minute + 0.6, second );
% add 3 minutes to set the end time for extraction
t2 = datenum( year, month, day, hour, minute + 2.4, second );

w2 = extract(w, 'time', t1, t2);
fprintf('Start %s\n',datestr(get(w2,'start')));
fprintf('End %s\n',datestr(get(w2,'end')));

plot(w2,'xunit','date'); grid on;
h = gcf; axis('tight');
set( findall( h, '-property', 'FontSize' ), 'FontSize', 18 );
set( findall( h, '-property', 'FontName' ), 'FontName', 'Helvetica' );
set( findall( h, '-property', 'FontWeight' ), 'FontWeight', 'Bold' );

set( h, 'Position', [100 100 1400 500] );
set( h, 'PaperPositionMode', 'auto' );
print( h, '-dpng', 'figs/raw_crop.png');
```

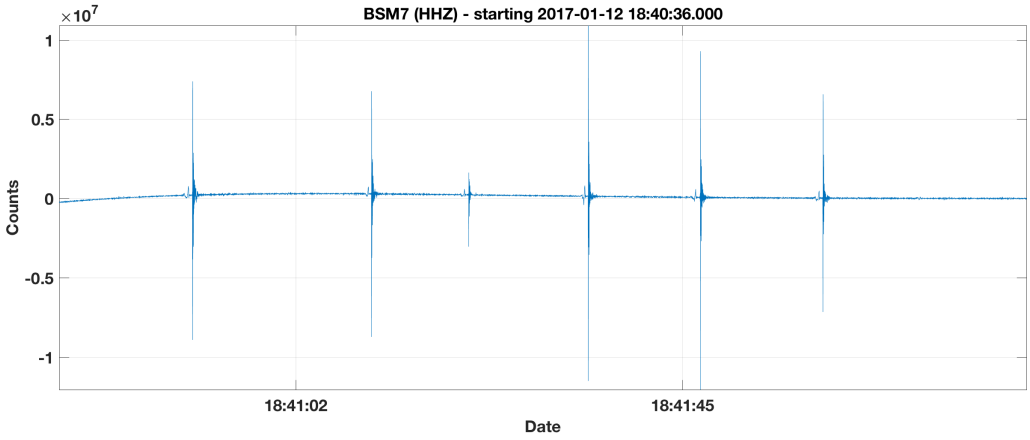

Figure 3: Seismic data after demean and detrend. This is the zoom around the time period of interest.

We first do the poor-human's instrument correction. This is a term given when someone just converts from counts to m/s. I did some more digging and found I gave some incorrect data for the Meridian sensor. Here is what we need.

```
% The following information comes from the manual for the 120s Meridian
% sensor.

% The conversion goes like
%
% velocity = Amplitude [counts] * A/D conversion factor [V/counts] \
%            Gain * Sensitivity [V / m/s]
%
% e.g. see https://www.passcal.nmt.edu/content/instrumentation/field-procedures/working-responses-get-units-displ

% For the Meridian Compact, sensitivity is as follows for each gain
% sensitivity: 3.00e8, 6.00e8, 1.20e9, 3.00e9 counts/V
% gain:        1,      2,      4,      10

% Our gain was 1, so the sensitivity is 3.00e8
ADC = 3.00e8; % counts/V

% The normalization factor A0 = 4.34493e+17 (this makes I(f=1 Hz) = 1 )
A0 = 4.34493e+17; % [ (rad/s)^-5 ] not important until we build the frequency response

% We can also look at the sensitivity in terms of V/m/s
% At 1 Hz, this is sensitivity = 754.3; % [V / m/s]
sensitivity = 754.3; % [V / m/s]

w3   = w2; % copy the waveform
data = double(w3); % get the data
data = data * (ADC^-1); % [V] keep in mind that our ADC includes the gain already
data = data ./ sensitivity; % [m/s]
data = data .* 1e6; % convert to [um/s]

w3 = set(w3,'data',data); % put correct data back into waveform object
w3 = set(w3,'units','\mum/s'); % set the corrected units

plot(w3,'xunit','date'); grid on;
h = gcf; axis('tight');
set( findall( h, '-property', 'FontSize' ), 'FontSize', 18 );
set( findall( h, '-property', 'FontName' ), 'FontName', 'Helvetica' );
set( findall( h, '-property', 'FontWeight' ), 'FontWeight', 'Bold' );

set( h, 'Position', [100 100 1400 500] );
set( h, 'PaperPositionMode', 'auto' );
print( h, '-dpng', 'figs/raw_crop_poor.png');
```

No we can see that our units start to make sense. We are talking about micrometer per second velocities. Finally, we want to filter in a certain frequency band and view our nice stomps.

```
w4 = w3;
w4 = demean(w4); % built in function for waveform object
w4 = detrend(w4); % built in function for waveform object

fmin = 0.1; % [Hz] lowest frequency
fmax = 100; % [Hz] highest frequency

% create a Butterworth bandpass filter with 2 poles
fobj = filterobject('b', [fmin fmax], 2);

% Make sure to taper the edges before filtering
tmax = 1 / fmin; % [s] maximum period
nmax = tmax * get( w4,'freq' ); % [npts] length of tmax in samples
R    = nmax / get( w4,'data_length' );
```

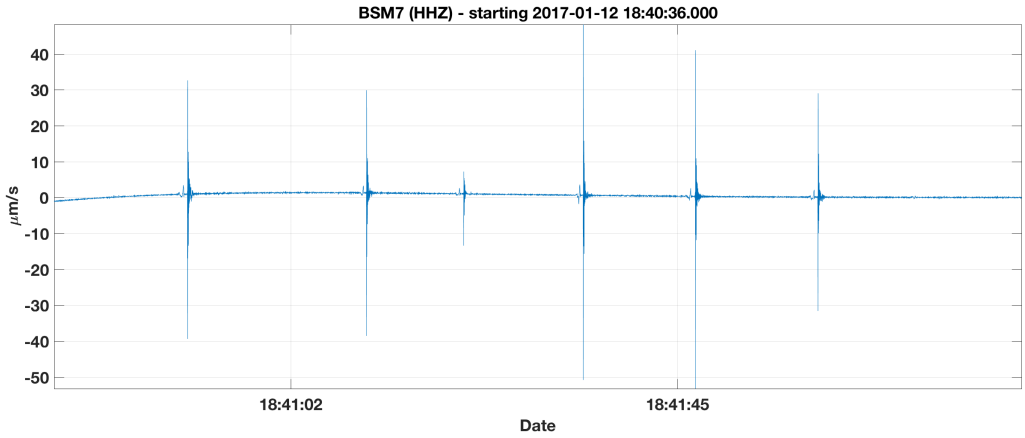

Figure 4: Seismic data after demean and detrend and sensitivity correction. This is the zoom around the time period of interest.

```
w4   = taper( w4, 2*R ); % built in function for waveform object

% apply the filter in both directions (acausal) - this is a zero phase
% filter which is helpful because it doesn't disperse different frequency
% components. the caveat is that it can spread energy so arrivals may appear
% slighter earlier than they actually are
w4 = filtfilt(fobj, w4); % filtfilt is zero phase application of filter

plot(w4,'xunit','date'); axis('tight'); grid on;
h = gcf;
set( findall( h, '-property', 'FontSize' ), 'FontSize', 18 );
set( findall( h, '-property', 'FontName' ), 'FontName', 'Helvetica' );
set( findall( h, '-property', 'FontWeight' ), 'FontWeight', 'Bold' );

set( h, 'Position', [100 100 900 900] );
set( h, 'PaperPositionMode', 'auto' );
print( h, '-dpng', 'figs/raw_crop_poor_filtered.png');
```

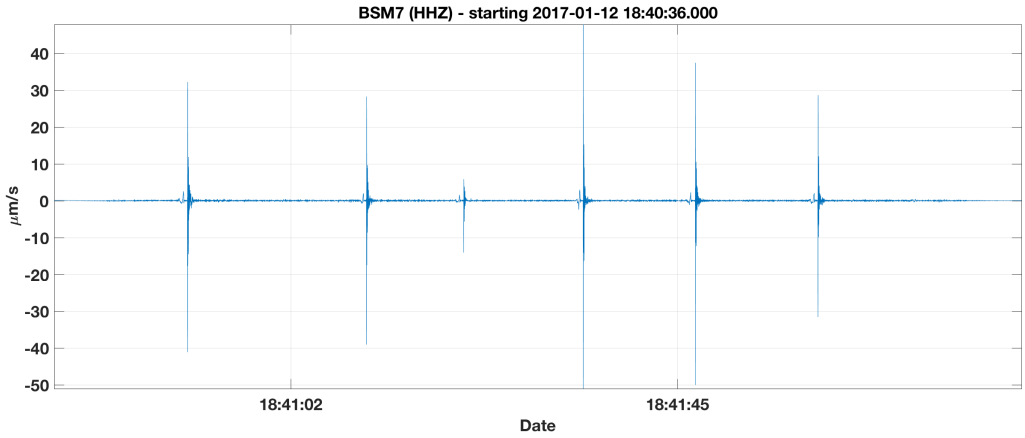

Figure 5: Seismic data after demean and detrend, sensitivity correction and bandpass filter. This is the zoom around the time period of interest.

We can take the information from the headers (e.g. npts, dt) and use the poles and zeros below to create our own frequency response for the Nanometrics Meridian sensors. *Keep in mind you will need to build a frequency vector and make sure to note the units of the poles and zeros below when are plugging them into functions. Some functions want Hz, some want rad*Hz (or rad/sec).*

Note that the units are in rad/sec; I told you before that they were in 1/s. I went back to the Meridian manual and check plots of their frequency response. If we use that the units are rad/sec for the poles and zeros, our frequency response matches the one plotted in the manual.

```
% RESP file is here:
% https://ds.iris.edu/NRL/sensors/nanometrics/RESP.XX.NS353..BHZ.MeridianPosthole.120.1200

% Or look at 120s Meridna manual pg. 137

zeros = [ % the zeros in rad/sec
     0.000000E+00;
     0.000000E+00;
    -3.920000E+02;
    -1.960000E+03;
    -1.490000E+03 + 1i*1.740000E+03;
    -1.490000E+03 + 1i*-1.740000E+03
    ];

poles = [ % the poles in rad/sec
    -3.691000E-02 + 1i* 3.702000E-02;
    -3.691000E-02 + 1i*-3.702000E-02;
    -3.430000E+02 + 1i* 0.000000E+00;
    -3.700000E+02 + 1i* 4.670000E+02;
    -3.700000E+02 + 1i*-4.670000E+02;
    -8.360000E+02 + 1i* 1.522000E+03;
    -8.360000E+02 + 1i*-1.522000E+03;
    -4.900000E+03 + 1i* 4.700000E+03;
    -4.900000E+03 + 1i*-4.700000E+03;
    -6.900000E+03 + 1i* 0.000000E+00;
    -1.500000E+04 + 1i* 0.000000E+00
    ];

% It is worth noting that according to the Meridian manual, these values are in the S-domain and in angular frequ

dt = 1 / get(w,'freq'); % [s/samp] time sample interval

npts = numel( double(w2) ); % same as get(w,'data_length')

% I want to make a frequency vector based on my data in the time domain
nfft   = 2^(nextpow2(npts)+1);
fs     = 1  / dt;                    % sample frequency
df     = fs / nfft;                  % sample interval in fourier domain
fNyq   = fs / 2;                     % Nyquist sampling frequency

fArray = ( 0 : ( nfft - 1 ) ) .* df; % frequency vector

% set the correct negative part of frequency array
fArray( fArray >= fNyq ) = fArray( fArray >= fNyq ) - ( fNyq * 2 );

% fArray = fftshift( fArray ); % [Hz] linear frequency

wArray = 2*pi*fArray; % [rad/s] angular frequency to match poles and zeros

B = poly(zeros); % convert poles to polynomial
A = poly(poles); % convert zeros to polynomial

INST_RESP = A0 * freqs( B, A, wArray ); % frequency domain instrument response

h = figure;
```

```
subplot(2,1,1); % plot the amplitude of the INST_RESP
plot( fArray, abs(INST_RESP), 'b','Linewidth',2); grid on;
axis('tight'); %xlabel('Frequency [Hz]');
ylabel('Amplitude [m/s/Hz]');
hold on; title('Amplitude Spectral Density');
xlim([0 fNyq]); % plot only positive frequencies so we can use log scale in x-axis
ylim([0 1.1]);
set(gca,'XScale','log');

subplot(2,1,2); % the plot the phase of the INST_RESP
plot( fArray, angle(INST_RESP).*180/pi, 'b','Linewidth',2); grid on;
title('Phase spectra');
xlabel('Frequency [Hz]'); ylabel('Phase [^o]'); hold on;
xlim([0 fNyq]); ylim([-180 180]);
set(gca,'XScale','log');

set( findall( h, '-property', 'FontSize' ), 'FontSize', 18 );
set( findall( h, '-property', 'FontName' ), 'FontName', 'Helvetica' );
set( findall( h, '-property', 'FontWeight' ), 'FontWeight', 'Bold' );

set( h, 'Position', [100 100 1400 700] );
set( h, 'PaperPositionMode', 'auto' );
print( h, '-dpng', 'figs/meridian_instrument_response_FD.png');
% You can compare this figure with that on page 139 of the Meridian manual
```

If you are using MATLAB, you will want to use the functions **poly.m** and **freqs.m** to build the polynomials of the numerator and denominator and compute the frequency domain response. Multiply your response by the value A0=4.344928E+17; this is the conversion factor that normalizes the frequency response to unit amplitude (i.e. 1) at the frequency f_0. In the case of the Nanometrics Meridian sensor, f_0=1 Hz.

Note: There are two zeros equal to zero for this sensor. That means it is a velocity sensor. If there were three zeros equal to zero this would be a displacement sensors, and if there were one zero equal to zero this would be an acceleration sensor.

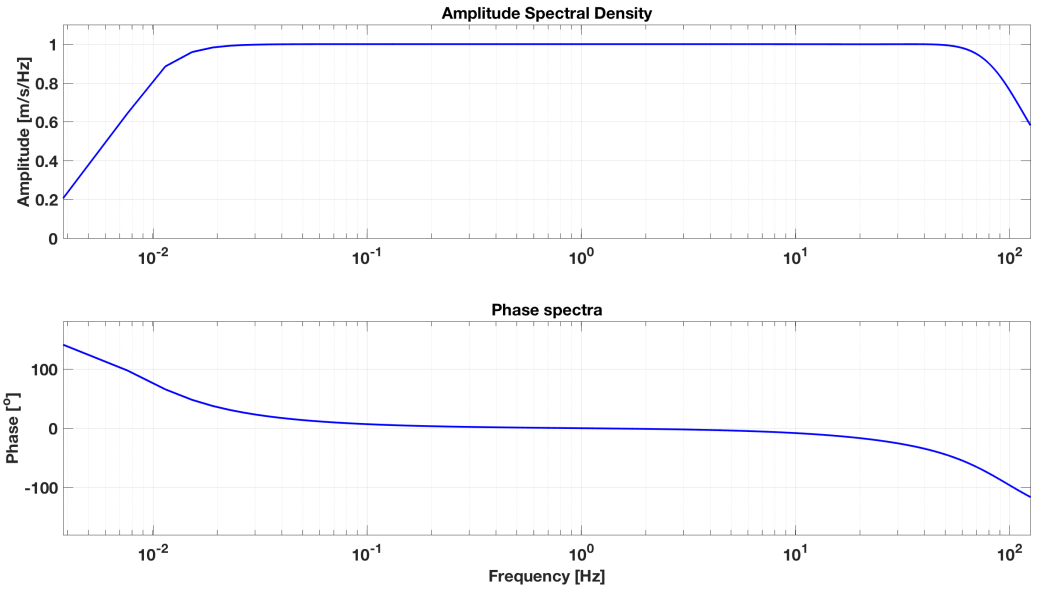

Figure 6: The frequency response. See Meridian manual page 139 for comparison with the 120s and 20s sensors.

After you have built the frequency response, you should convert to the time domain and visualize the

following:

- Amplitude spectral density *(5 pts.)*
- Phase spectra *(5 pts.)*
- The time-domain response *(5 pts.)*

You will likely note that the time-domain response is not causal. If that is the case, read through the Haney et al. (2012) paper to determine how they enforce causality with the Kramers-Kronig relationship. Plot your causal time-domain response after making this correction. *(5 pts.)*

I am not counting the time-domain plots in this homework. It is not exactly correct to just take the ifft() of the Laplace domain frequency response. The Laplace transform is actually the $F(i\omega)$ Fourier transform, so simply the taking the inverse is not correct. You should still see the influence of causality and the Kramers-Kronig relationship when you do take the ifft() of the Laplace domain frequency response. Computationally computing the inverse Laplace transform is not easy from what I have read.

Discuss what the amplitude, phase and time-domain plots tell you about this instrument and how it records different frequencies or resonates with time. *(10 pts.)*

You should note that the time-domain plot varies depending on what you set *df* and f_{Nyq} when you build your frequency response. I used the default time length and *dt* from the data I took. That was a long time series so it ensured a small *df*. You will not though that my causal time-domain impulse response is still not perfect. I would need to rebuild my frequency domain response with optimal sampling parameters to get a better spike.

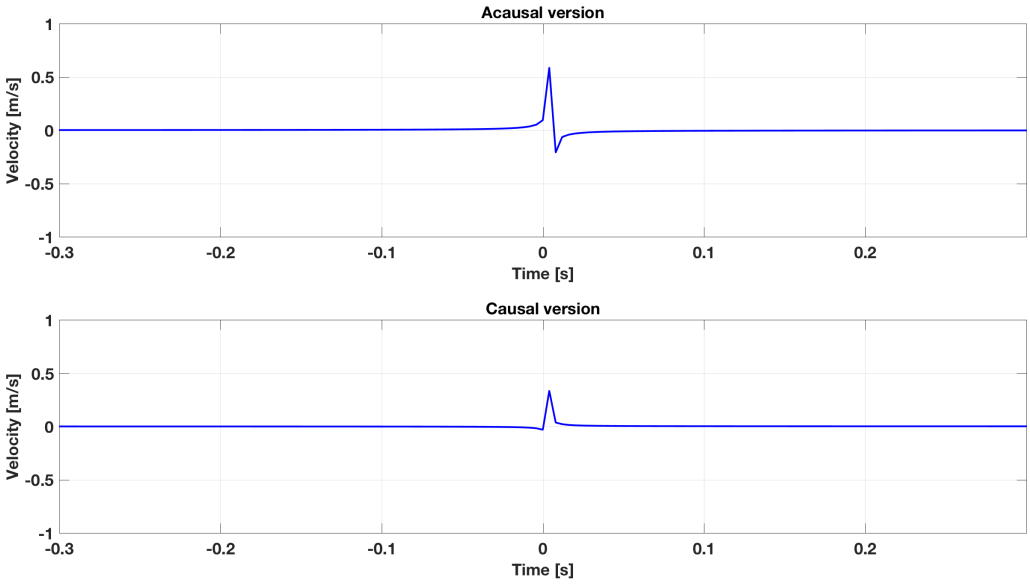

Figure 7: The time domain impulse response.

Part 3: Instrument deconvolution *(35 pts.)*

Note: Dylan gave you the wrong advice in the original homework. Haney computes A0 automatically in his code and I did not give you the ADC conversion factor. Another thing I am not sure about is Haney's units on the poles and zeros. He multiplies his poles and zeros by 2π, and when I do this I get a nice result that matches the poor-human's version of the instrument deconvolution. But the units of the poles and zeros are already in rad/s according to the Meridian user manual and when I built the frequency response using the raw poles and zeros in the S-domain, it matched their impulse response. So there is something not correct in the manual or Haney's code, or I don't understand some subtle difference between the poles and zeros.

Use the function *GEOPH677/Codes/Haney2012_CausalInstrumentCorrection/rm_instrum_resp.m* from the GIT repository to apply the instrument deconvolution given the poles and zeros from the last question. *(15 pts.)* In this case, set the sensitivity to 4.000000E+05. In this case setup the following inputs:

```
addpath('/Users/dmikesell/GIT/GEOPH677/Codes/Haney2012_CausalInstrumentCorrection/');

% Meridian Information

% Our gain was 1, so the sensitivity is 3.00e8
ADC = 3.00e8; % counts/V

% We can also look at the sensitivity in terms of V/m/s
% At 1 Hz, this is sensitivity = 754.3; % [V / m/s]
sensitivity = 754.3; % [V / m/s]

gain = ADC * sensitivity; % [counts/ m/s]

% apply same filter as beginning
w5 = w2;
w5 = demean(w5); % built in function for waveform object
w5 = detrend(w5); % built in function for waveform object
w5 = taper( w5, 2*R ); % built in function for waveform object
w5 = filtfilt(fobj, w5); % filtfilt is zero phase application of filter

%-------------------------------------------------------------------------
% rawdata:      vector of raw uncorrected data in digital counts
rawdata = double(w5);
% badvals:      value of data during telemetry drop or clip (e.g., -2^31)
badvals = NaN;
% samplrate:    the sampling rate (Hz)
samplrate = get(w5,'freq');
% pols:         poles (radians/sec, not Hz)
pols = poles*2*pi;
% zers:         zeros (radians/sec, not Hz)
zers = zeros*2*pi;
% flo and fhi:  frequency range over which to get instrument response (Hz)
flo = fmin;
fhi = fmax;
% ordl          Butterworth order at flo, the low cutoff (between 2 and 4)
ordl = 2;
% ordh          Butterworth order at fhi, the high cutoff (between 3 and 7)
ordh = 3;
% digout:       inverse gain (m/s/count)
digout = 1/gain;
% digoutf:      frequency of normalization (Hz)
digoutf = 1;
% ovrsampl:     over-sampling factor for digital filter accuracy (e.g., 5)
ovrsampl = 10;
% idelay:       intrinsic delay in the acquisition system
idelay = 0;
```

```
prcdata = rm_instrum_resp(rawdata,badvals,samplrate,pols,zers,...
    flo,fhi,ordl,ordh,digout,digoutf,ovrsampl,idelay);

w5 = set(w5,'data',prcdata.*1e6);
w5 = set(w5,'units','\mum/s');

plot(w5,'xunit','date'); axis('tight'); grid on;

h = gcf;
set( findall( h, '-property', 'FontSize' ), 'FontSize', 18 );
set( findall( h, '-property', 'FontName' ), 'FontName', 'Helvetica' );
set( findall( h, '-property', 'FontWeight' ), 'FontWeight', 'Bold' );

set( h, 'Position', [100 100 1400 500] );
set( h, 'PaperPositionMode', 'auto' );
print( h, '-dpng', 'figs/decon.png');
```

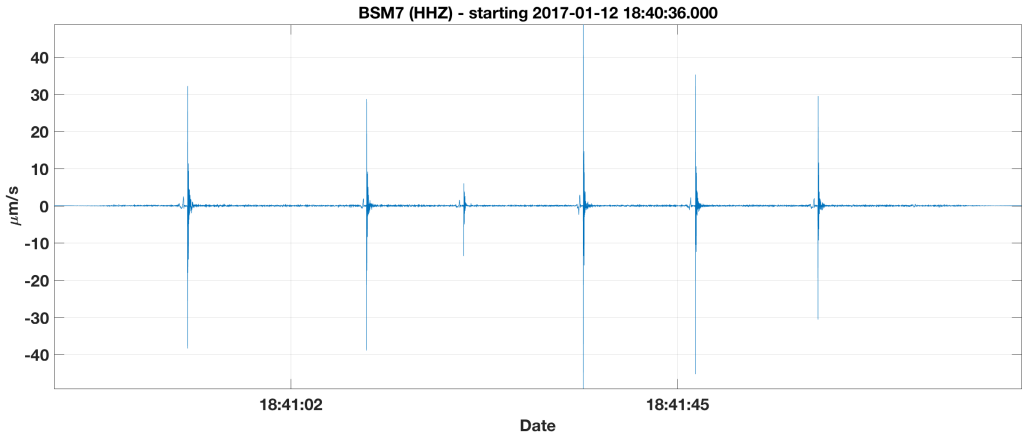

Figure 8: Deconvolved data.

After applying the instrument correction to the data, cut the data around your stomp. Compare the data before and after deconvolution. *Note: You will have to normalize the amplitudes to compare counts to m/s units. Concentrate on the phase and initial arrivals.* Comment on the differences and similarities. *(15 pts.)*

Make a plot of your deconvolved stomp. You can also make a plot of the normalized comparison. *(5 pts.)*

The stomps were made at the following times.

```
% Stomp test for meridians: 2017-01-12
%
% Alex: 18:41:10
% Tom: 18:41:30
% Diego: 18:41:40
% Zongbo: 18:41:54
% Matt: 18:42:07
% Rebekah: 18:42:21
```

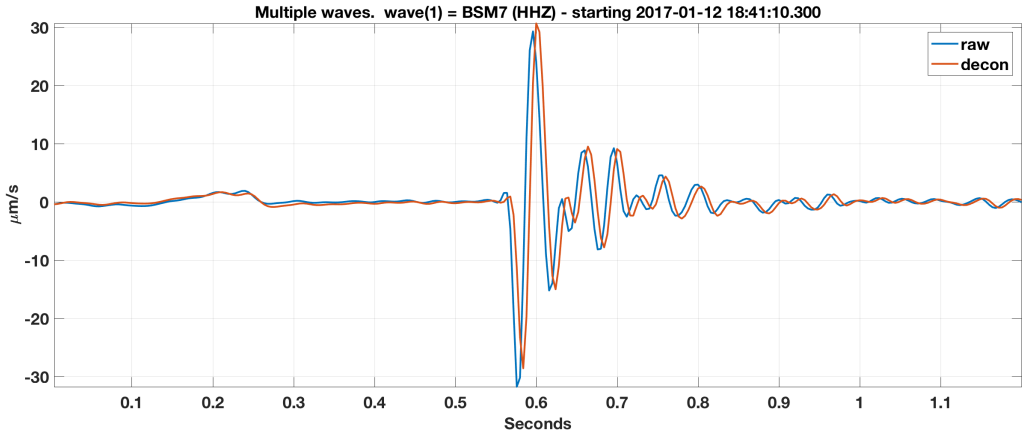

Figure 9: Alex's stomp with bandpass 1 to 50 Hz.

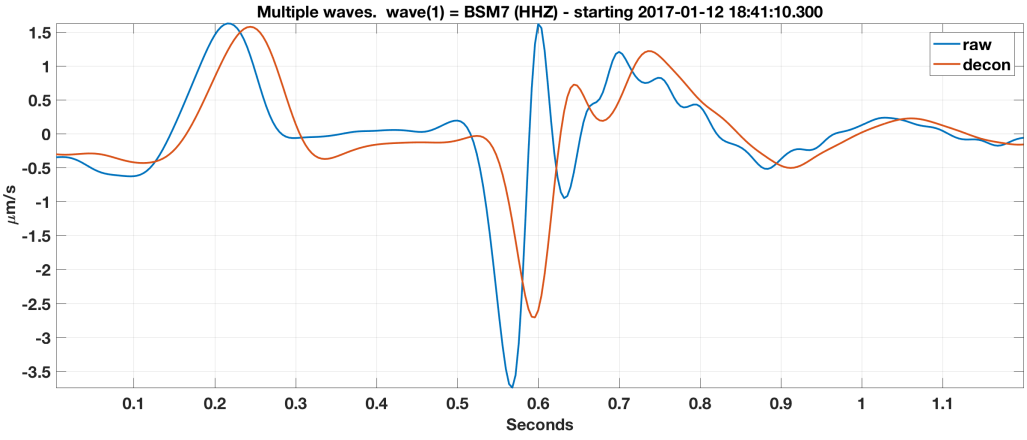

Figure 10: Alex's stomp with bandpass 0.5 to 10 Hz.