LIGOAnalysis2Detectors

Analyzes data from https://www.gw-openscience.org/events/GW150914/ using MATLAB. The first observations of gravitational waves in the LIGO collaboration Hanford, Washington and Livingston, Louisiana interferometers for a black hole merger are reconstructed using band pass and notch filtering and their relative time delay determined via cross correlation.

Author: D. Carlsmith, University of Wisconsin-Madison

Clear variables and close open figures.

```
clear all; close all;
```

Use 4096 Hz sample rate (smoothed) raw data files containing 32 s of strain data centered on the event.

Loop over data files for the two detectors.

```
nfile=2
```

```
nfile =
     2
```

```
for ifile=1:nfile
```

Define data file name.

```
filename = 'H-H1_LOSC_4_V2-1126259446-32.txt';
if ifile == 2
    filename = 'L-L1_LOSC_4_V2-1126259446-32.txt';
end
```

Call function created interactively with Import wizard to read the file. (Function is attached.) Each file contains strain for sequential times with the same start time.

```
strain = importfile(filename);
```

Scale strain to sensible values rather than 0.001 fm in 1 km.

```
meanstrain=mean(strain)
```

```
meanstrain =
   5.8955e-23
meanstrain =
  -1.0522e-18
```

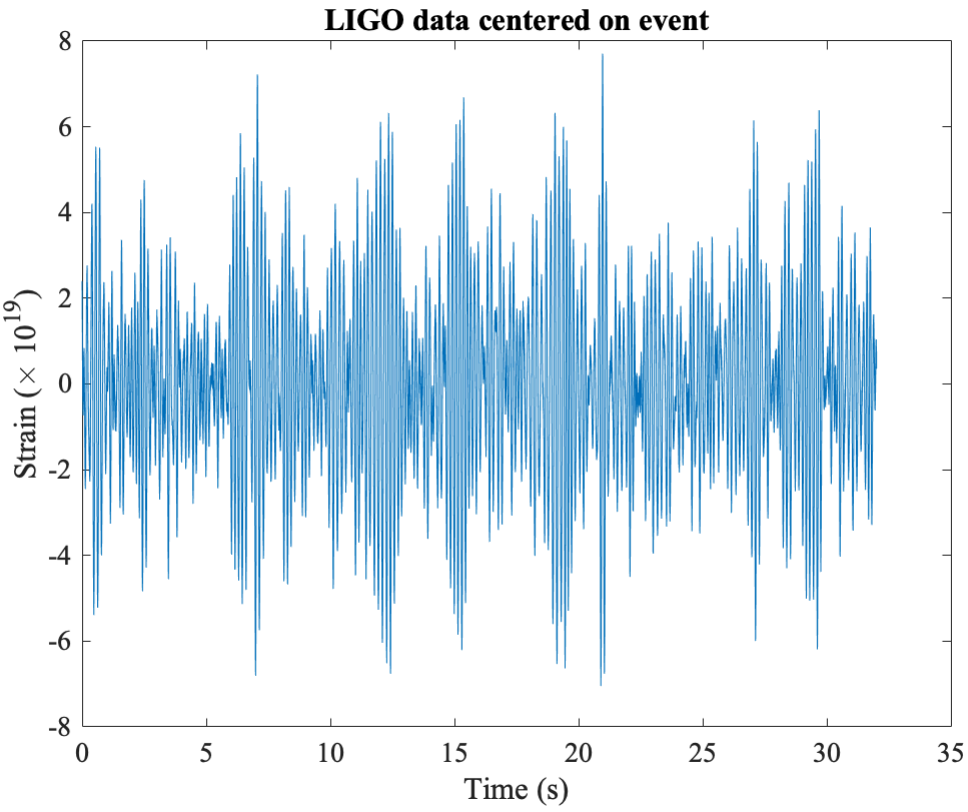
```
strain=(strain-meanstrain)*10^(19);
```

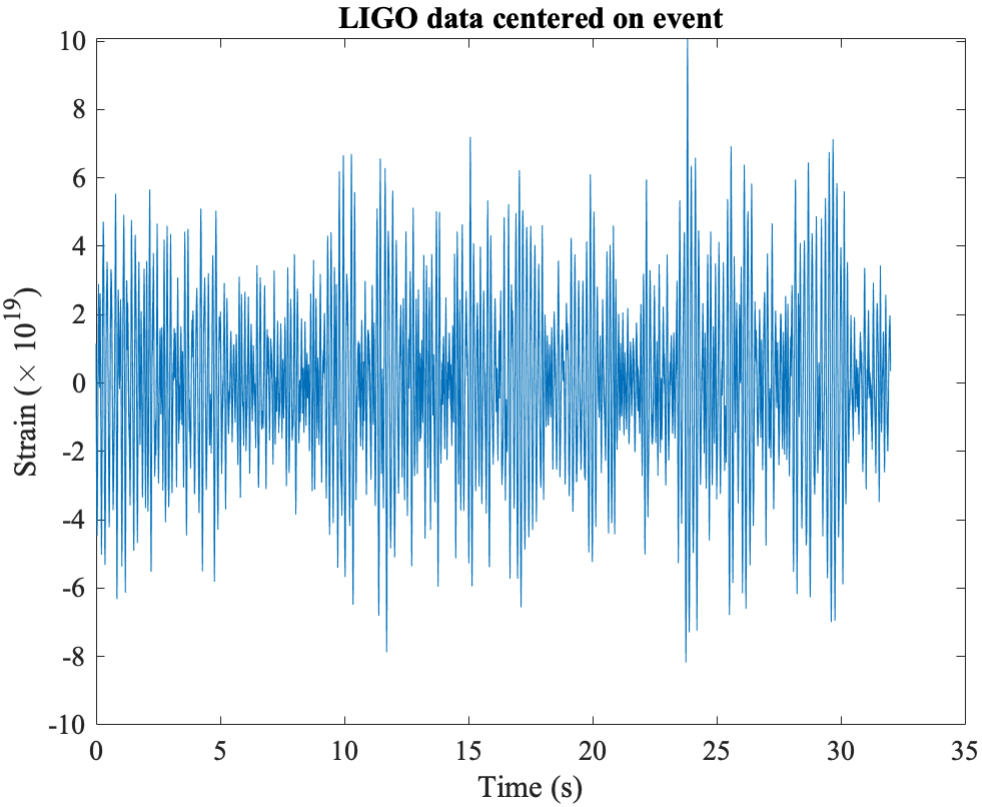
Compute times from published sample rate.

```
fs = 4096; %Hz
time=(1:length(strain))/fs;
time=time';
```

Plot strain vs time.

```
figure;
plot(time,strain)
xlabel('Time (s)');ylabel('Strain (\times 10^{19})');
title('LIGO data centered on event')
```



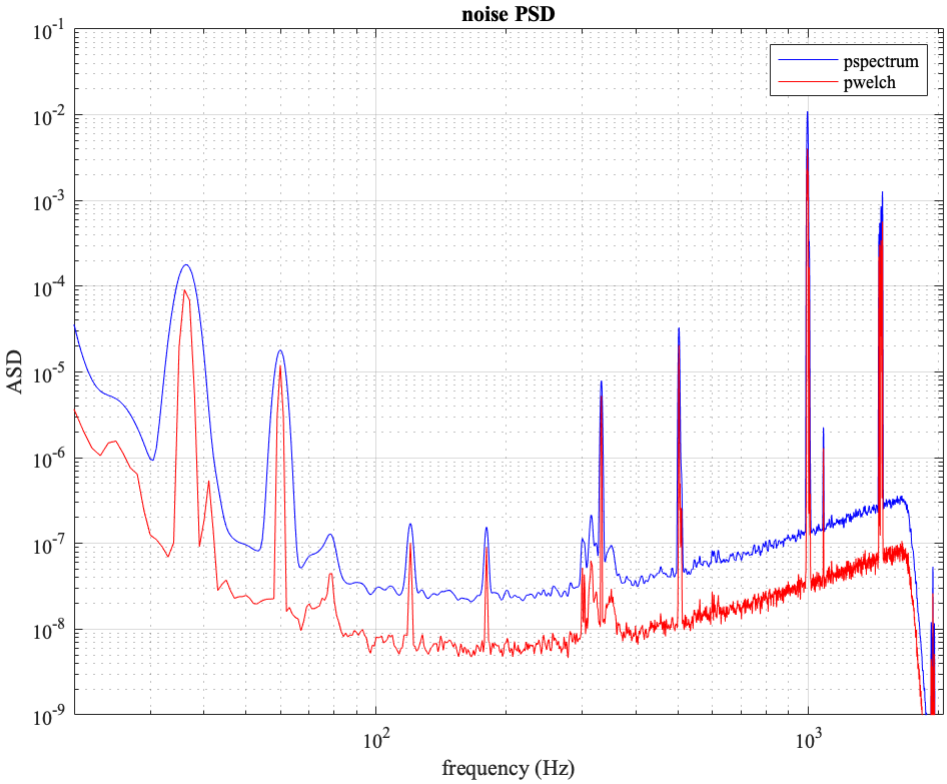


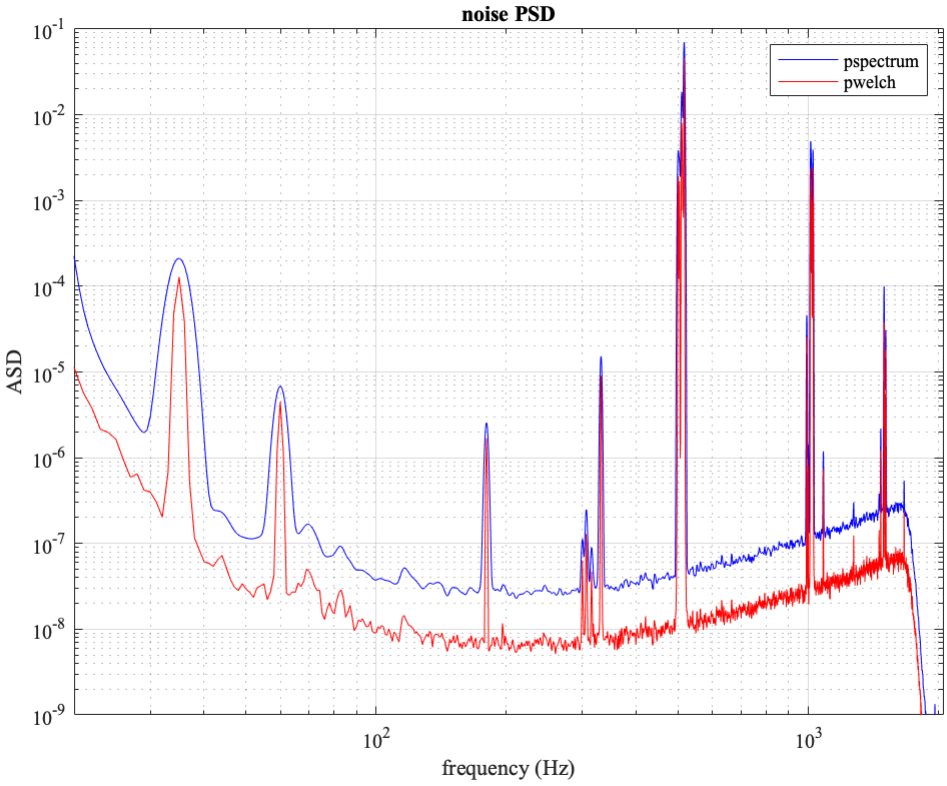
Plot power spectrum using `pspectrum`. This function performs an FFT with some smoothing modification. See https://en.wikipedia.org/wiki/Spectral_density_estimation and https://en.wikipedia.org/wiki/Window_function .

```
[p,f]=pspectrum(strain,fs);
```

Compare to LIGO's plot_data.m welch psd algorithm performed by `pwelch`.

```
nfft = fs;
[psdL,freqL] = pwelch(strain,hann(nfft),nfft/2,nfft,fs);
asdL = sqrt(psdL);% asdL is the square root of the power psdL
figure
loglog(f, p,'b');% plot power versus frequency for pspectrum
hold on
xlim([20, 2048]);% frequency limits for plot
loglog(freqL, psdL,'r') % add power versus frequency for pwelch
xlim([20, 2048]) ;ylim([1.e-9, .1])
grid on
xlabel('frequency (Hz)');ylabel('ASD')
title(' noise PSD ');hold off
legend('pspectrum','pwelch')
```





Find noise peaks in the power spectrum using peak finder `findpeaks`.

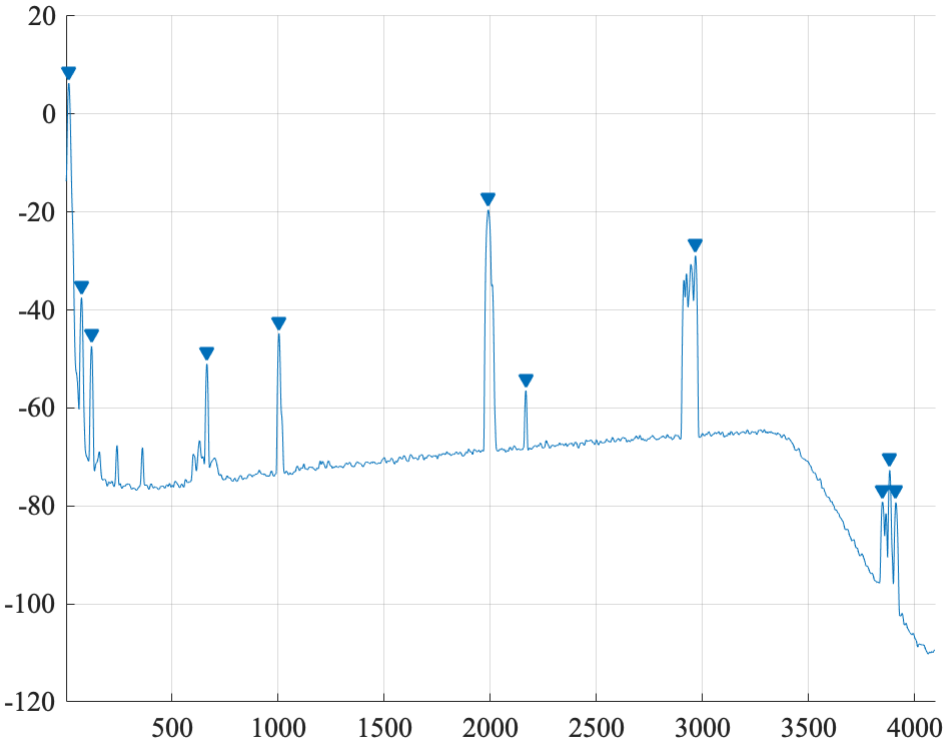
```
q=10*log10(p);
figure
hold on
```

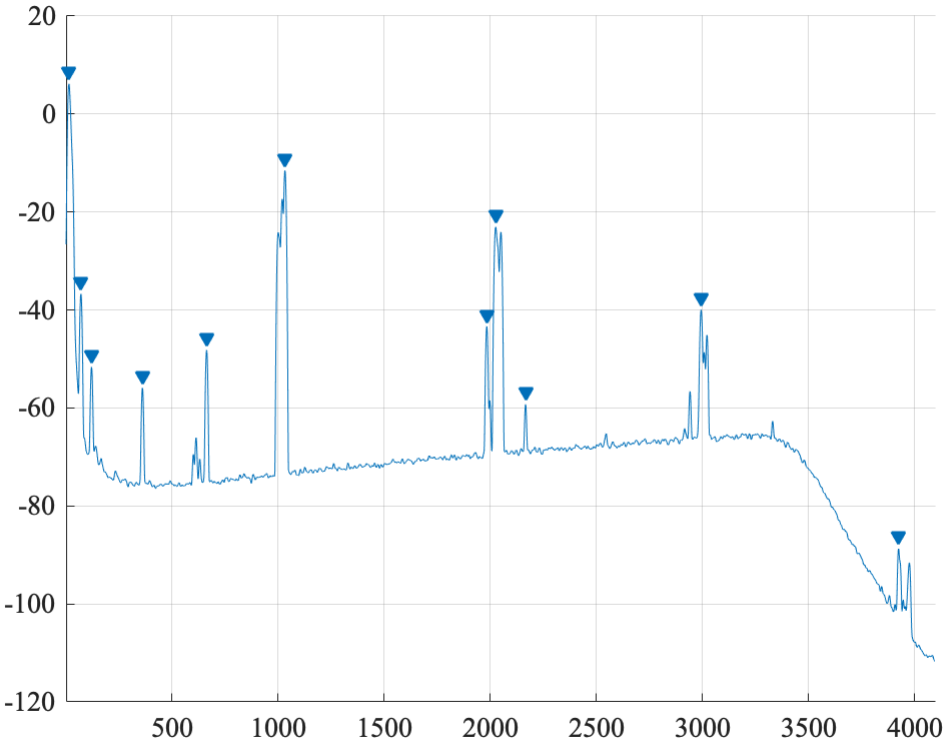
Require peak height of 10 and sort in descending order of importance/height.

```
[pks,locs,w,prom] = findpeaks(q,'MinPeakProminence', 10, 'SortStr','descend');
```

Call it again without collecting results to make a plot.

```
findpeaks(q,'MinPeakProminence', 10, 'SortStr','descend');
```





Apply a bandpass filter with lower limit 35 Hz and upper limit 350 Hz per published analysis and superpose. Use 100db not the `bandpass` default 60 db attenuation.

```
strainband=bandpass(strain,[35,350],fs,'StopbandAttenuation',100);
```

Apply notch filters to remove various peaks previously found. Note the peaks were found before the band pass filter was applied so peak locations and widths near band edges were properly identified.

The notch filters have a central frequency and a width for half power. The peak finding gives a centeral frequency specified by `locs` and a width `w` which refers to the base of the peak.

https://www.mathworks.com/help/signal/ug/remove-the-60-hz-hum-from-a-signal.html

```
strainfilt=strainband;
for i = 1:length(pks)% loop over peaks
    fc=f(locs(i));% frequency of the peak
    if fc>20;%&fc<350 % limit the number of notch filters
        f1=fc-w(i)/6;f2=fc+w(i)/6; %use fraction of base width.
        if w(i)>fc;f1=0.1;end
```

We use a 2nd order filter.

```
         d = designfilt('bandstopiir','FilterOrder',2, ...
                   'HalfPowerFrequency1',f1,'HalfPowerFrequency2',f2, ...
                   'DesignMethod','butter','SampleRate',fs);
```

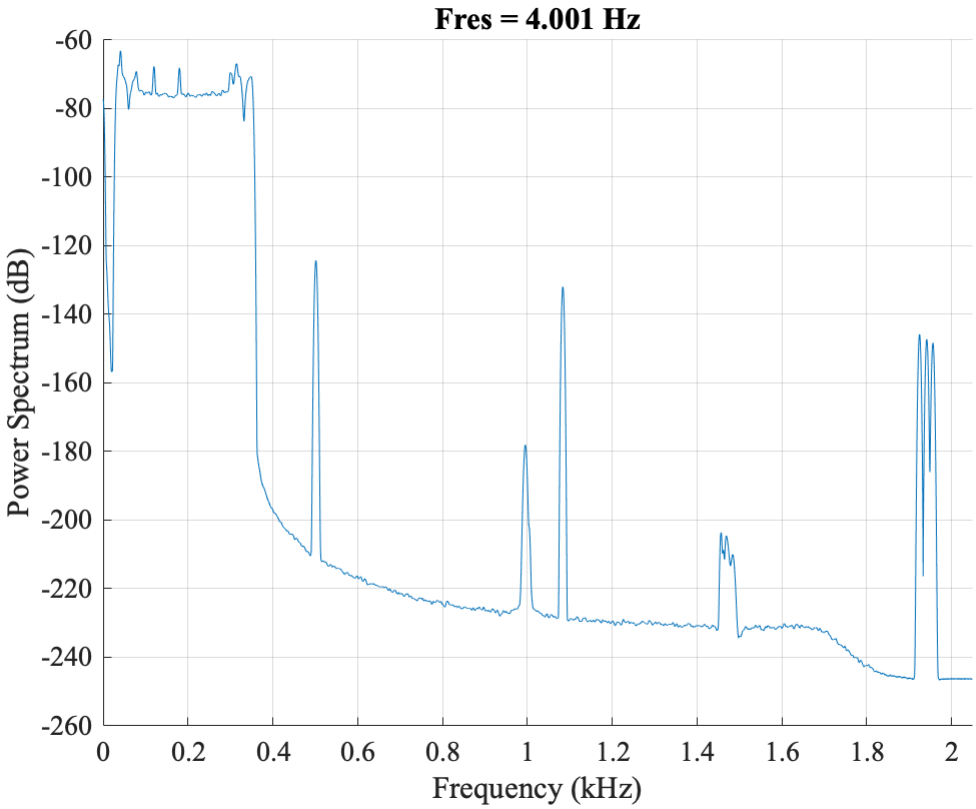
```
        strainfilt = filtfilt(d,strainfilt);
    end
end
```

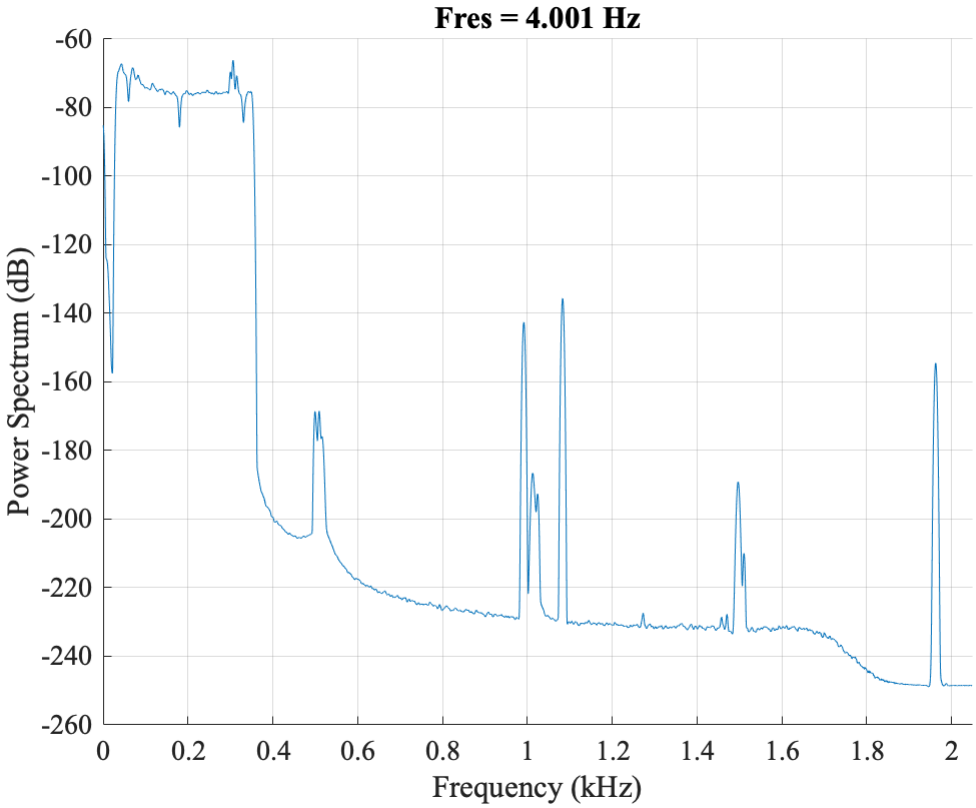
Renormalize filtered strain data to zero mean.

```
strainfiltmean=mean(strainfilt);
strainfilt=strainfilt-strainfiltmean;
```

Plot the power spectrum of the filtered data.

```
figure
hold on
[p,f]=pspectrum(strainfilt,fs);
pspectrum(strainfilt,fs);
```





Standard deviation of the strain signal.

```
strainsig=std(strainfilt);
```

Create window around published time near the center of the 32 s signal, and make initial mask to drop endpoints and glitches from data.

```
tstart=16.00;tend=16.45;
mask=time<tend&time>tstart&strainfilt<3*strainsig&strainfilt>-3*strainsig;
figure
plot(time(mask),strainfilt(mask))
```

Save the subsample of the filtered data.

```
strainsav(ifile,:)=strainfilt(mask);
timesav(ifile,:)=time(mask);
```

Overlap published plot which spans a fraction of the signal window.

```
[Htime,Hstrain] = importHfile('HanfordGW150914DataPlot.txt');
if ifile==2
    [Htime,Hstrain] = importHfile('LivingstonGW150914DataPlot.txt');
end
Htime=Htime+16; % plot data starts at 16 s into the 32 s sample.
Hmask = Htime<tend&Htime>tstart;
```

```
mask2= mask&time>Htime(1);
```

There seems to be a scale factor difference of 100 from 1e19 and 1e21 strain units.

```
scale=mean(abs(strainfilt(mask2)))/mean(abs(Hstrain(Hmask)))
```

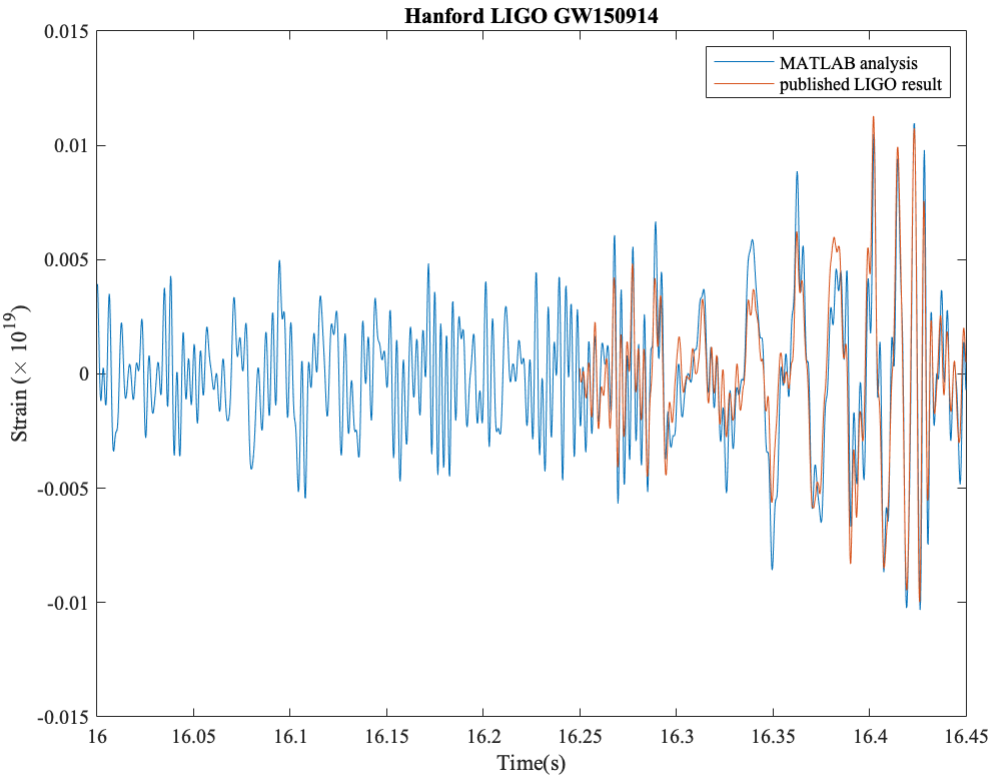
```
scale =
     0.011515
scale =
     0.010709
```

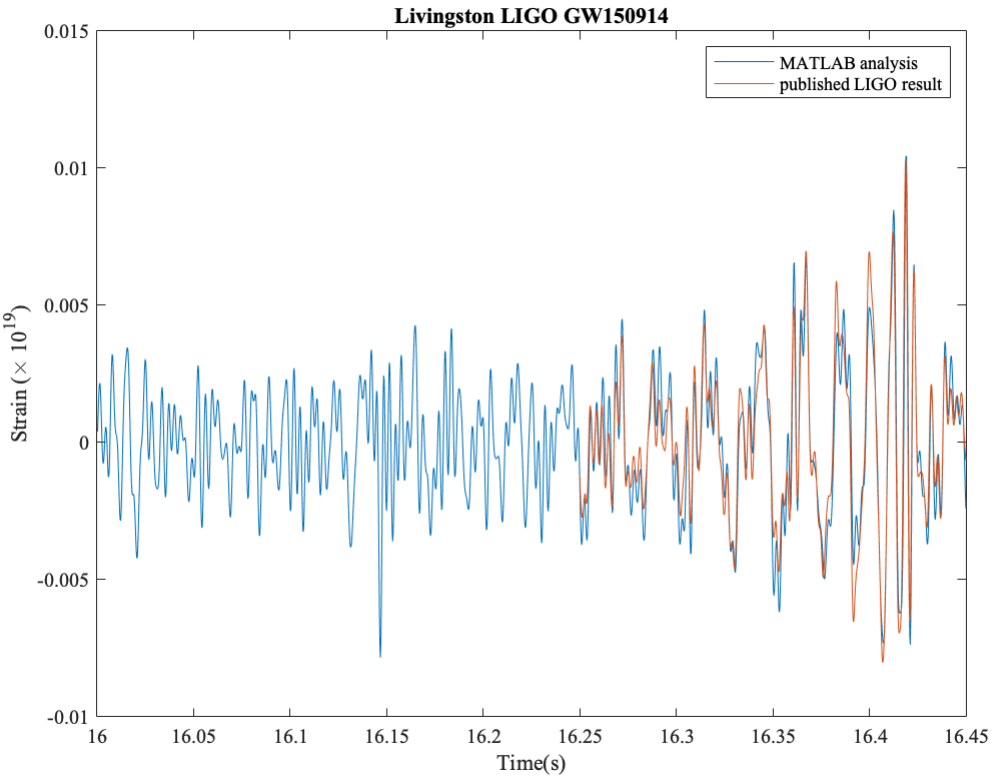
Set the scale factor to exactly 100.

```
scale = 0.01;
```

Add the published result selected and filter by LIGO to the plot.

```
hold on
Hstrain=Hstrain*scale;
plot(Htime(Hmask),Hstrain(Hmask))
xlabel('Time(s)')
ylabel('Strain (\times 10^{19})')
if ifile==1; title('Hanford LIGO GW150914');end
if ifile==2; title('Livingston LIGO GW150914');end
legend('MATLAB analysis','published LIGO result')
hold off
```

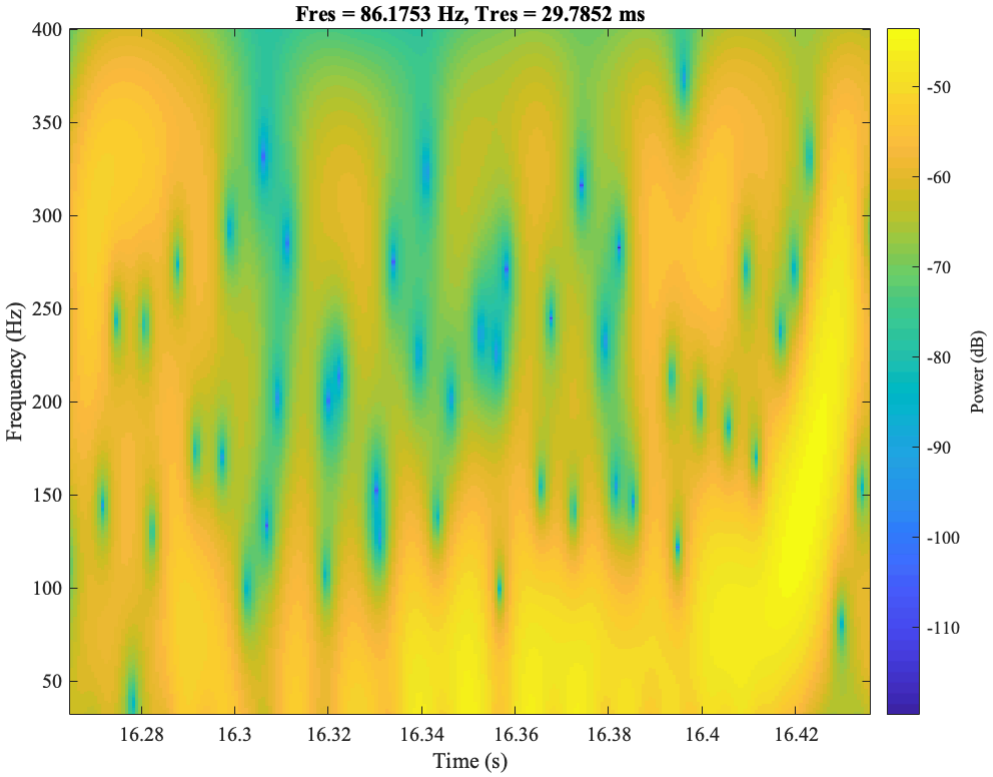


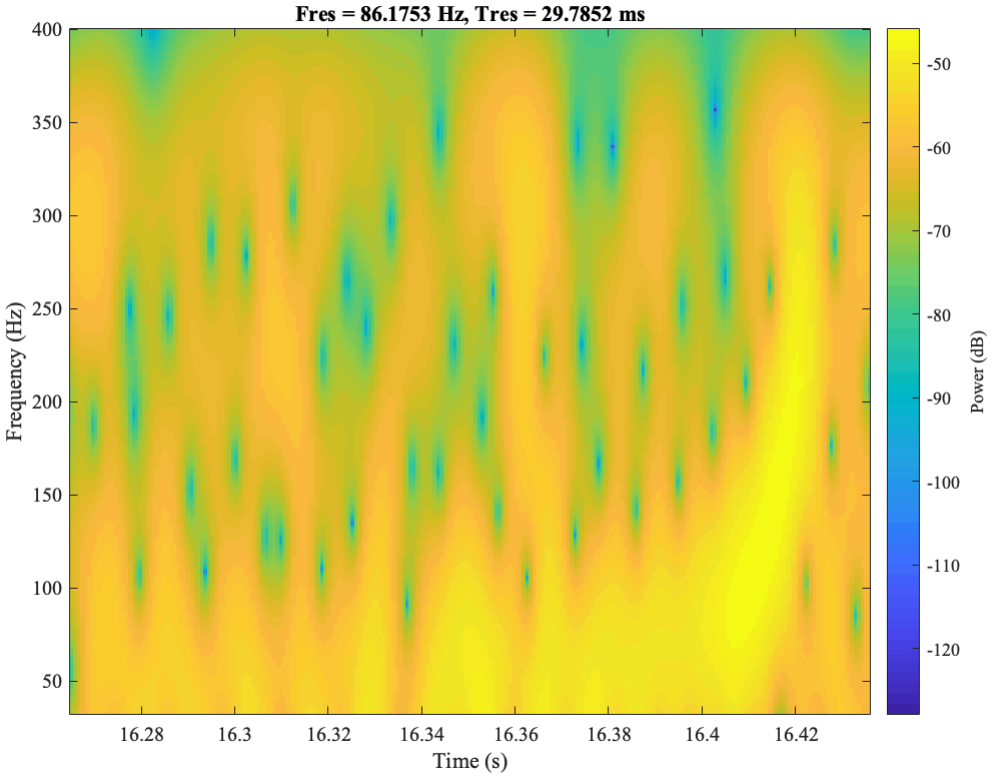


Notice the raw strain levels were 5-8 units and we are looking at .005 or 6e-4 of the raw amplitudes. The nature of the filters used may account for the differences.

Make a spectrogram that shows the chirp of frequency as the holes inspiral and orbit faster and faster before merging in one black hole using `pspectrum` and specifying 'spectogram'. Specify a window over which to calculate the spectrum small compared to the duration of the signal and the large fractional overlap of successive windows.

```
pspectrum(strainfilt(mask2),time(mask2),'spectrogram','FrequencyLimits',...
    [32,400],'TimeResolution',0.03,'OverlapPercent',98)
```





The chirp is most evident as the rising band after 16.4 s.

```
end % end of loop over files
```

Quit if number of files processed is equal to 1.

```
if nfile==1
    return
end
```

Compute correlation between the two detectors.

```
[acor,lag] = xcorr(strainsav(1,:),strainsav(2,:));
```

Find the maximum in the correlation and the corresponding time difference.

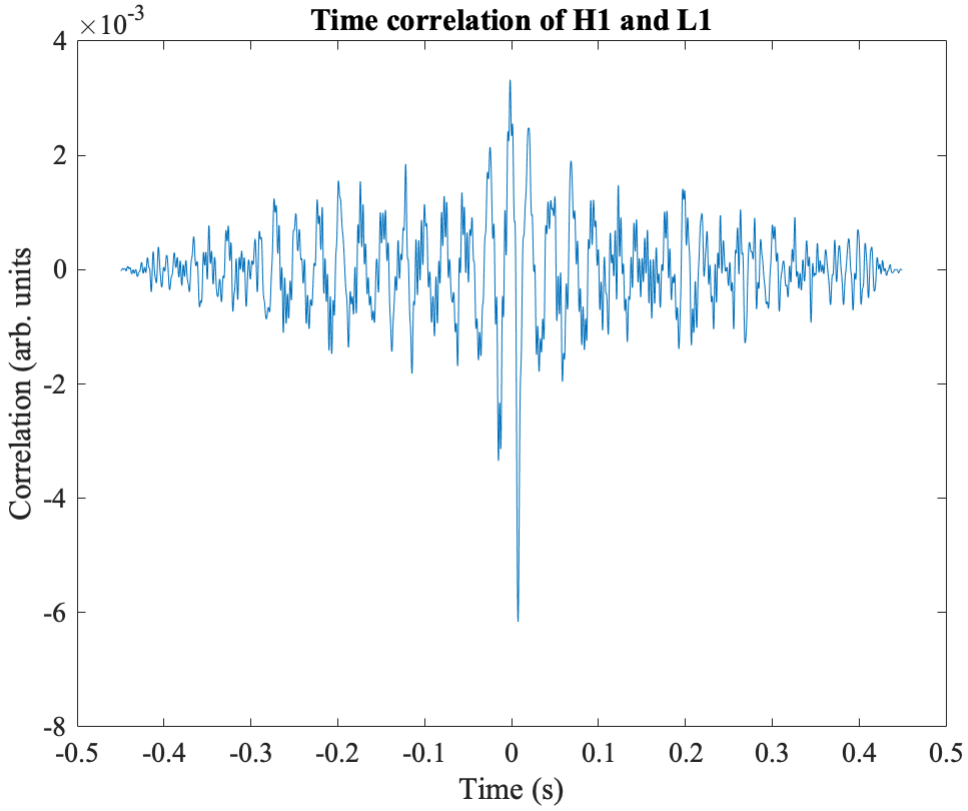
```
[~,I] = max(abs(acor));
lagDiff = lag(I);
```

The first file earlier gives negative result. The following delay is positive so the first 2nd (l1) arrives earlier. Quoted value is 6.9+.5-.4

```
timeDiff = lagDiff/fs
```

```
timeDiff =
    0.0075684
```

```
figure
plot(lag/fs,acor)
title('Time correlation of H1 and L1')
xlabel('Time (s)');ylabel('Correlation (arb. units')
```



One can access the bulk data at https://www.gw-openscience.org/data/ and apply this algorithm to conduct a search oneself.

The end.

```
function strain = importfile(filename, startRow, endRow)
%IMPORTFILE Import numeric data from a text file as column vectors.
%   STRAIN = IMPORTFILE(FILENAME) Reads data from text file FILENAME for
%   the default selection.
%
%   STRAIN = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME.
%
% Example:
%   strain = importfile('H-H1_LOSC_4_V2-1126259446-32.txt',4, 131075);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2018/11/10 12:13:58

%% Initialize variables.
delimiter = ' ';
if nargin<=2
    startRow = 4;
    endRow = inf;
end
```

```
%% Format for each line of text:
%   column1: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%*s%*s%*s%*s%*s%*s%*s%*s%*s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter, 'Mult
for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, 'Delimiter',
    dataArray{1} = [dataArray{1};dataArrayBlock{1}];
end

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Allocate imported array to column variable names
strain = dataArray{:, 1};


end
function [Htime,Hstrain] = importHfile(filename, startRow, endRow)
%IMPORTFILE Import numeric data from a text file as column vectors.
%   [HTIME,HSTRAIN] = IMPORTFILE(FILENAME) Reads data from text file
%   FILENAME for the default selection.
%
%   [HTIME,HSTRAIN] = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data
%   from rows STARTROW through ENDROW of text file FILENAME.
%
% Example:
%   [Htime,Hstrain] = importfile('HanfordGW150914DataPlot.txt',2, 3442);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2018/11/10 15:52:47

%% Initialize variables.
delimiter = ' ';
if nargin<=2
    startRow = 2;
    endRow = inf;
end
```

```
%% Format for each line of text:
%   column1: double (%f)
% column2: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%*s%*s%*s%*s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
textscan(fileID, '%[^\n\r]', startRow(1)-1, 'WhiteSpace', '', 'ReturnOnError', false);
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter, 'Mult
for block=2:length(startRow)
    frewind(fileID);
    textscan(fileID, '%[^\n\r]', startRow(block)-1, 'WhiteSpace', '', 'ReturnOnError', false);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, 'Delimiter',
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Allocate imported array to column variable names
Htime = dataArray{:, 1};
Hstrain = dataArray{:, 2};




end
```