Contents

Import the relevant signal data

I converted the csv to a text file prior to using it because it was easier to import.

```
signal_data_file = 'Signal.txt';                    % Grab the file name
delimiter = ',';                                     % Set delimiter as a comma since data is seperated via commas
signal_data = importdata(signal_data_file,delimiter); % Import the data
time = 0:0.1:180.0;                                  % Create a time vector based on the sampling rate. (10Hz = 0.1s s
panning 1801 points)

% Initialize two empty arrays which will dynamically grow as we find peaks
peaks = [];             % Will hold the peak values
timestamps = [];        % Will hold the time stamps at which the peaks are found

signal_mean_value = 0.6833; % The expected true signal mean value
```

Detrend the signal to remove the linear bias

```
% We know that y(t) = s(t) + at. We also know that sum(s(t)) / length(t) = 0.6833
% We can use this information to derive the linear bias 'a':
% sum( y(t) - s(t) ) / length(time)  =  0.6833
% => sum(y(t)) - sum(s(t)) = 0.6833 * length(time)
% => sum(s(t)) = sum(y(t)) - ( 0.6833 * length(time) )

sum_st = sum(signal_data) - ( 0.6833 * length(time) );  % Sum of s(t)

% We know that sum(s(t)) = sum(at) = a * sum(time) therefore:
a = sum_st / sum(time);
signal_detrended = zeros(1,length(signal_data));        % Preallocate space for speed

% For loop which removes linear bias from each data point
for i = 1:length(signal_data)
    signal_detrended(i) = signal_data(i) - (a * time(i));   % We recover the original signal
end
```

Grab the peak values and positions of the peaks

```
% Construct a simple peak detection routine using a for loop. We do this by computing the
% gradients that come before and after a specific point.
for i = 2:length(signal_detrended)-1

    last_Slope = (  signal_detrended(i) - signal_detrended(i-1) ) / (0.1);    % Keeps track of the previous gradient
    next_Slope = (  signal_detrended(i+1) - signal_detrended(i) ) / (0.1);    % Keeps track of the next gradient

    if (last_Slope > 0) && (next_Slope < 0)     % If the gradient goes from positive to negative
        peaks = [peaks signal_detrended(i)];    % Append the peak value to the peak array
        timestamps = [timestamps time(i)];        % Append the time to the locations array (where the peak occured)
    end
end
```

Prepare data structures prior to valid peak detection routine

```
% Combine the peak and location arrays into a single data structure. This
% makes it easier to handle both sets of data at the same time.
peaks_locs = [peaks' timestamps'];

% Add a column of 'ones' to the data object. This column serves to 'flag'
% which peaks are valid (1) and which are invalid (0) All the peaks are
% initially flagged as valid.
peaks_locs = [peaks_locs ones(length(peaks_locs),1)];

% Sort the rows in ascending order based on the time stamps
peaks_locs = sortrows(peaks_locs,2,{'ascend'});

% Sort the peaks in descending order seperately (stored in 'P')
% Grab the indices of these peaks (stored in 'I')
[P, I] = maxk(peaks_locs(:,1),length(peaks_locs));
```

Begin valid peak routine

```
% For loop to interate over the peaks in descending order and marks them valid or invalid
for j = 1:length(peaks_locs)

    % If statement to run the routine only when the peak is valid
    if peaks_locs(I(j),3) == 1

        current_maxPeak = P(j); % Grab the current peak value

        % Find the index of the first peak that is within the last 5
        % seconds current peak.
        index_minusFive = find(  (peaks_locs(:,2) >= peaks_locs(I(j),2)-5) & (peaks_locs(:,2) < peaks_locs(I(j),2))  , 1,
'first'  );

        % Find the index of the last peak that is within the next 5
        % seconds current peak.
        index_plusFive  = find(  (peaks_locs(:,2) <= peaks_locs(I(j),2)+5) & (peaks_locs(:,2) > peaks_locs(I(j),2))  , 1,
'last'   );

        % ----------------------------- % % ----------------------------- %
        if ~isempty(index_minusFive)                    % If the index is not empty
            i = index_minusFive;

            % While loop iterating over the peaks within the last 5 seconds of the current peak.
            while i < I(j)
                if peaks_locs(i,1) > current_maxPeak/2  % If the peak is smaller than current peak amplitude / 2
                    peaks_locs(i,3) = 0;                % Set it to invalid
                end
                i = i+1;
            end
        end

        % We follow the same logic for peaks that are within the next 5
        % seconds of the current peak
        if ~isempty(index_plusFive)
            i = index_plusFive;

            while i > I(j)
                if peaks_locs(i,1) > current_maxPeak/2
                    peaks_locs(i,3) = 0;
                end
                i = i-1;
            end
        end
        % ----------------------------- % % ----------------------------- %
    end
```

```
end
```

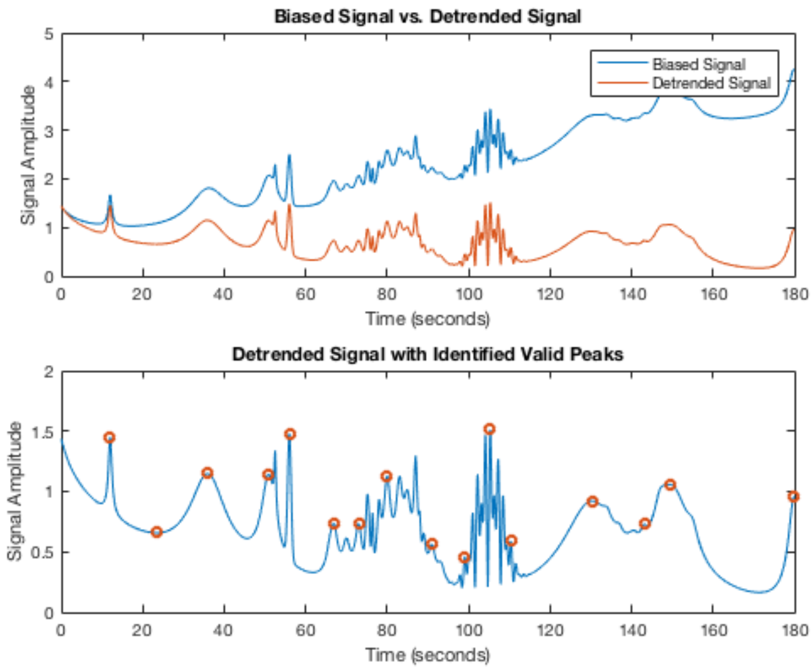
Determine the special peak

```
valid_rows = (peaks_locs(:,3) == 1);        % Determine which rows have 'valid' peaks
valid_peaks = peaks_locs(valid_rows,1:2);   % Extract them into a seperate data structure

special_peak = min(valid_peaks(:,1));            % Grab the minimum valid peak value
time_index = find(peaks == special_peak);        % Find the index at which the special peak occurs in peaks
sample_number = 1 + timestamps(time_index) * 10; % Use the time index to calculate the sample number
                                                 % We add +1 to ensure it is 1-based
% Display the values
disp('The special peak has a value of: ')
disp(special_peak)
disp('It is located at sample number: ')
disp(sample_number)
```

```
The special peak has a value of:
    0.4609

It is located at sample number:
   992
```

Plotting signals

```
% Plot the biased signal vs. the detrended signal
subplot(2,1,1)
plot(time,signal_data)
hold on
plot(time,signal_detrended)
ylabel('Signal Amplitude')
xlabel('Time (seconds)')
title('Biased Signal vs. Detrended Signal')
legend({'Biased Signal','Detrended Signal'})

% Plot the detrended signal and where its 'valid' peaks are
subplot(2,1,2)
plot(time,signal_detrended)
hold on
scatter(valid_peaks(:,2),valid_peaks(:,1))
ylabel('Signal Amplitude')
xlabel('Time (seconds)')
title('Detrended Signal with Identified Valid Peaks')

% ========================================================================= %
```



Published with MATLAB® R2017b