Eric Gallimore
SIO221A Final Exam

Task 1

To determine an expression for $\widehat{H}(\sigma)$, we begin with a cross-spectral estimate, the general form for which is

$$C_{Xy} = \frac{a_X^* a_y}{\Delta\sigma}$$

Now, we recognize that, in our measurements, the Fourier coefficients a_X are actually composed of two components: the signal, a_x, and some incoherent noise, a_n. When substituted, this yields

$$C_{(x+n)y} = \frac{(a_x + a_n)^* a_y}{\Delta\sigma}$$

Or, with algebraic rearrangement,

$$C_{(x+n)y} = \frac{a_x^* a_y + a_n^* a_y}{\Delta\sigma}$$

Now, we want to determine the frequency response. We begin with the general form:

$$H(\sigma) = \frac{C_{Xy}(\sigma)}{E_X(\sigma)}$$

In addition to incorporating the above expression $C_{(x+n)y}$ as the cross-spectral estimate, we recognize that the variance spectrum is actually composed of the superposition of two components: the spectrum of the signal, E_x, and the spectrum of the noise, E_n. Therefore,

$$\widehat{H}(\sigma) = \frac{C_{(x+n)y}}{E_x + E_n}$$

This can be rewritten to develop a noise-to-signal ratio term, E_n/E_x:

$$\widehat{H}(\sigma) = \frac{C_{(x+n)y}}{E_x(1 + \frac{E_n}{E_x})}$$

At this point, it is useful to note (as shown, expanded, above) that

$$C_{(x+n)y} = C_{xy} + C_{ny}$$

Since we believe that the noise is uncorrelated with our signal (we have no reason to think otherwise), by the definition of the cross spectrum,

$$C_{ny} = 0$$

And, therefore

$$C_{(x+n)y} = C_{xy}$$

So,

$$\widehat{H}(\sigma) = \frac{C_{xy}}{E_x(1 + \frac{E_n}{E_x})}$$

Now, we determine the relationship between $\widehat{H}(\sigma)$ and $H(\sigma)$.

$$\frac{\widehat{H}(\sigma)}{H(\sigma)} = \frac{E_x}{C_{xy}} \frac{C_{xy}}{E_x(1 + \frac{E_n}{E_x})}$$

$$\frac{\widehat{H}(\sigma)}{H(\sigma)} = \frac{1}{(1 + \frac{E_n}{E_x})}$$

$$\widehat{H}(\sigma) = \frac{H(\sigma)}{(1 + \frac{E_n}{E_x})} = \frac{1}{1 + \frac{E_n}{E_x}} H(\sigma)$$

From this, we can see that the noise decreased the gain of $\widehat{H}(\sigma)$ relative to $H(\sigma)$ by a factor of (1+NSR). It didn't change the phase.

(The effect of other inputs of noise in this system is discussed in the course notes and is not relevant at this point since the noise is incoherent.)

SIO221A Final Task 2 (Eric Gallimore)

Table of Contents

Initialization

```
function final()

clear();
close all;
load('velb4degl.mat');

% Figure out how to partition this to get 50 DOF
dof = 50;
len_complete = size(velb4, 2);
points_per = floor(len_complete / dof);
```

2 - Reusing code from HW7

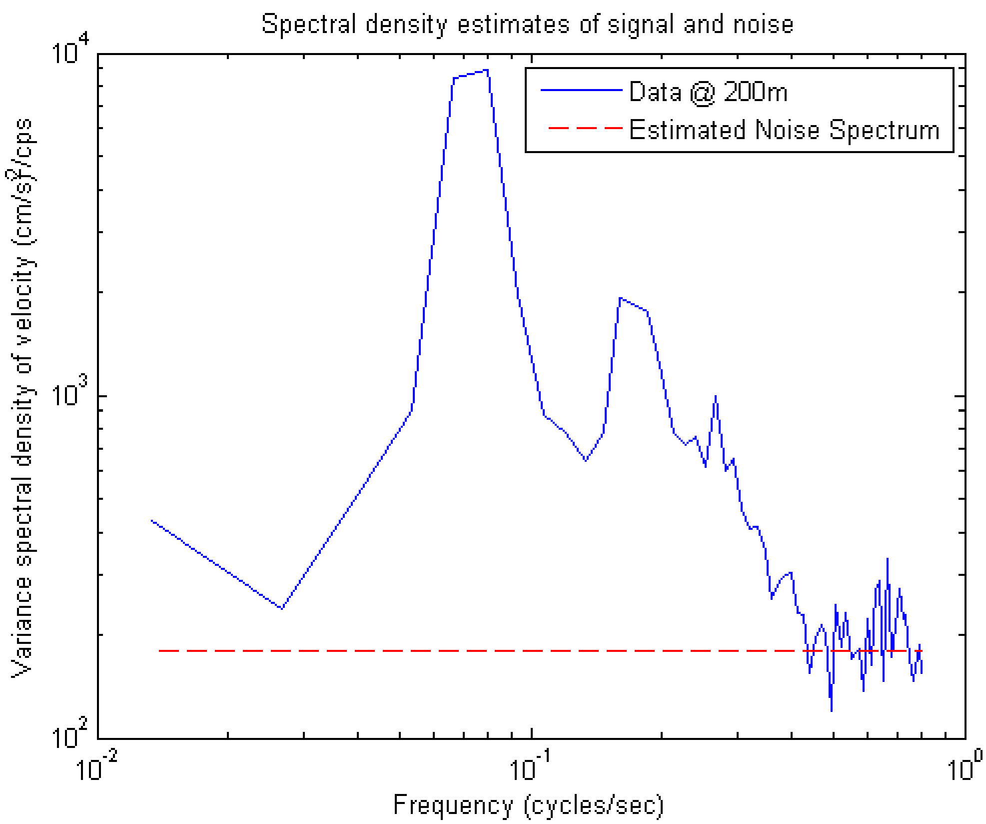
Estimate H(sigma) between ranges 100 and 90 using spectra and cross spectral estimates calculated at 50 dof, as per Homework 6. Plot the Gain and Phase functions at positive and negative frequencies.

```
sample_interval = 0.625; % seconds

velocity_100 = reshape(velb4(100,:), [], 1);
velocity_90 = reshape(velb4(90,:), [], 1);
velocity_80 = reshape(velb4(80,:), [], 1);

[cross_spec_100_90, freq_bins] = crossspectrum(...
      velocity_90,velocity_100,sample_interval, dof);
[ehat_100_real, ehat_100, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_multidof(velocity_100, sample_interval, dof);
[ehat_90_real, ehat_90, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_multidof(velocity_90, sample_interval, dof);
```

2A - Plot power spectrum of wave velocity and estimate of noise

find the noise floor through visual inspection of the plot below. It's white, and it looks like it's around 180.

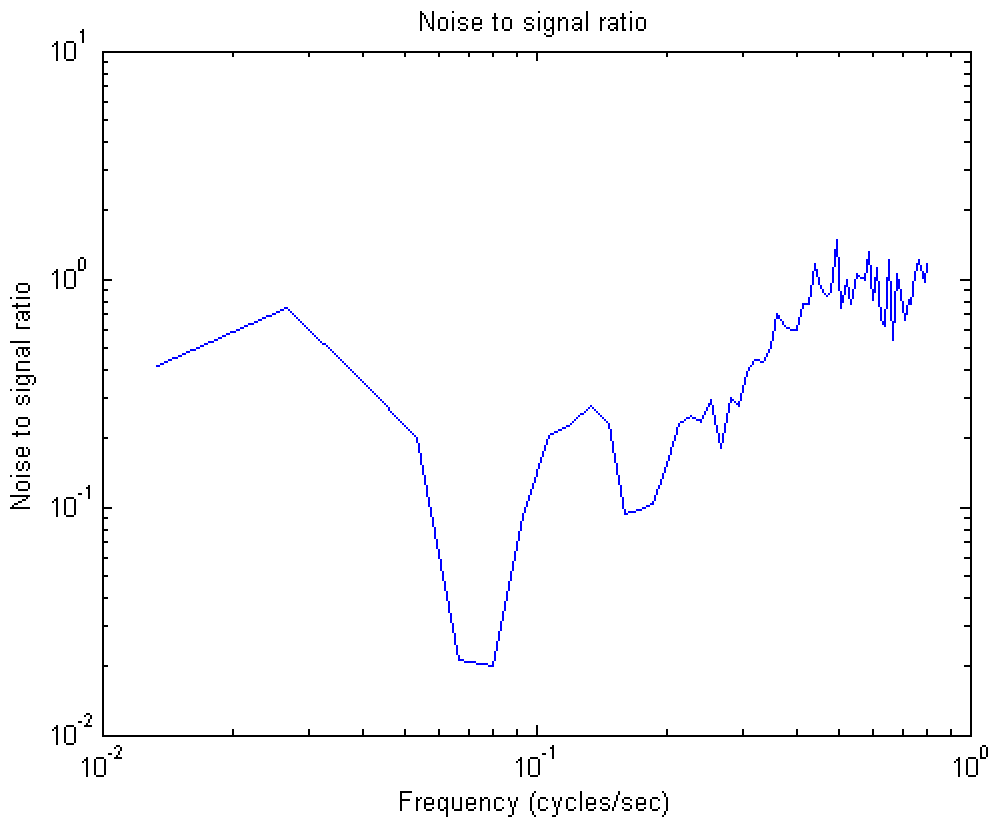
```
ehat_noise_real = ones(1,length(freq_bins_real)) .* 180;
ehat_noise = ones(1,length(freq_bins)) .* 180;

figure();
clf();
loglog(freq_bins_real, ehat_100_real);
hold on;
loglog(freq_bins_real, ehat_noise_real, 'r--');
xlabel('Frequency (cycles/sec)');
legend('Data @ 200m','Estimated Noise Spectrum');
title('Spectral density estimates of signal and noise');
ylabel('Variance spectral density of velocity (cm/s)^2/cps');
```



2B - Plot noise to signal ratio

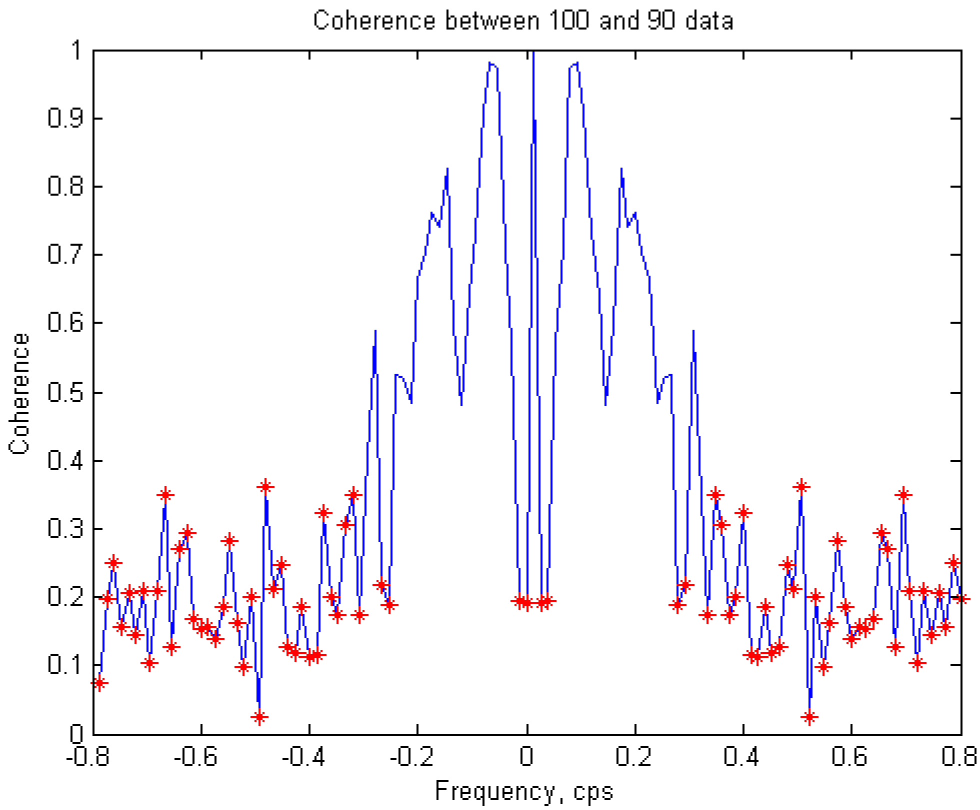
```
nsr_real = ehat_noise_real ./ ehat_100_real;
nsr = ehat_noise ./ ehat_100;
figure();
clf();
```

```
loglog(freq_bins_real, nsr_real);
xlabel('Frequency (cycles/sec)');
title('Noise to signal ratio');
ylabel('Noise to signal ratio');
```



2C setup - Check coherence

```
% find the frequencies where the choherence suggests that our cross
% spectral estimates are unreliable.

coherence = abs(cross_spec_100_90) ./ sqrt(ehat_100 .* ehat_90);
% a coherence of less than 0.4 isn't good...
bad_freq_bins = find(fftshift(coherence) < 0.4);
bad_coh_index = find((coherence) < 0.4);

figure()
plot(freq_bins, fftshift(coherence))
hold on;
co_shifted = fftshift(coherence);
plot(freq_bins(bad_freq_bins), co_shifted(bad_freq_bins), 'r*');
xlabel('Frequency, cps');
ylabel('Coherence');
title('Coherence between 100 and 90 data');
```



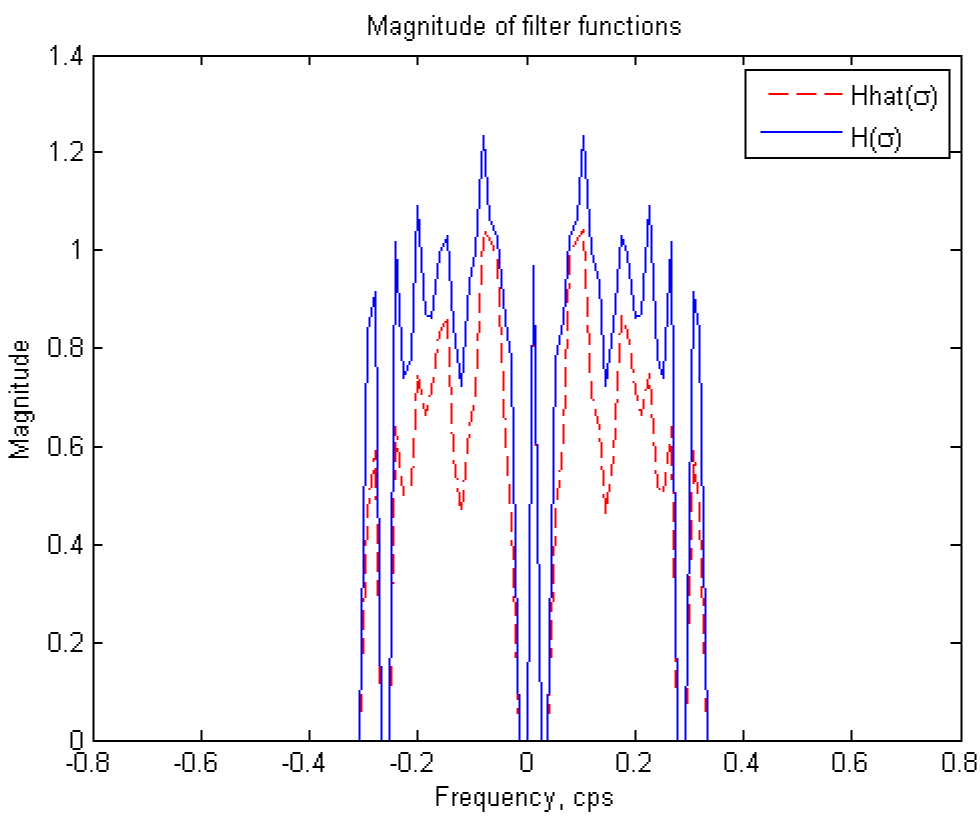
2C - Plot frequency responses

See part 1 for source of these equations

```
% zero estimates in frequency bands with bad coherence
H_hat = cross_spec_100_90(1:length(freq_bins)) ./ ehat_100;
%H_hat_zeroed = fftshift(H_hat);
%H_hat_zeroed(bad_freq_bins) = 0;
H_hat_zeroed = H_hat;
H_hat_zeroed(bad_coh_index) = 0;

H = H_hat .* (1 + nsr);
%H_zeroed = fftshift(H);
%H_zeroed(bad_freq_bins) = 0;
H_zeroed = H;
H_zeroed(bad_coh_index) = 0;

% uncorrected for noise
mag_h_hat = abs(H_hat_zeroed);
mag_h = abs(H_zeroed);

figure()
plot(freq_bins, fftshift(mag_h_hat), 'r--')
hold on;
plot(freq_bins, fftshift(mag_h));
%plot(freq_bins(bad_freq_bins), mag_h(bad_freq_bins), 'r*');
```

```
xlabel('Frequency, cps');
ylabel('Magnitude');
legend('Hhat(\sigma)', 'H(\sigma)');
title('Magnitude of filter functions');

disp('It is conceivable that waves might grow as they propagate eastward')
disp('due to energy input (from wind forcing?).  That said, I''m ')
disp('skeptical about that.');
disp('Some experimentation suggests that the error goes doen when H is')
disp('scaled, but I don''t have a solid explanation for this yet.');

% Try scaling H
% H = H / 1.233;
```

```
It is conceivable that waves might grow as they propagate eastward

due to energy input (from wind forcing?).  That said, I'm
skeptical about that.
Some experimentation suggests that the error goes doen when H is
scaled, but I don't have a solid explanation for this yet.
```

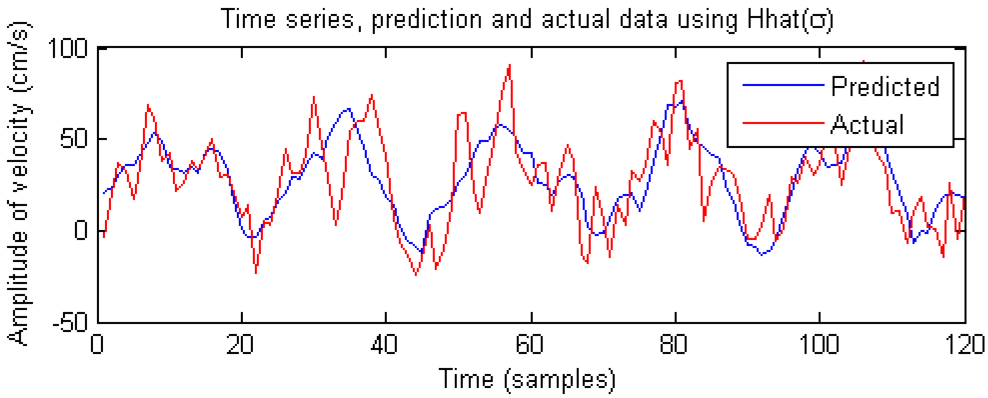


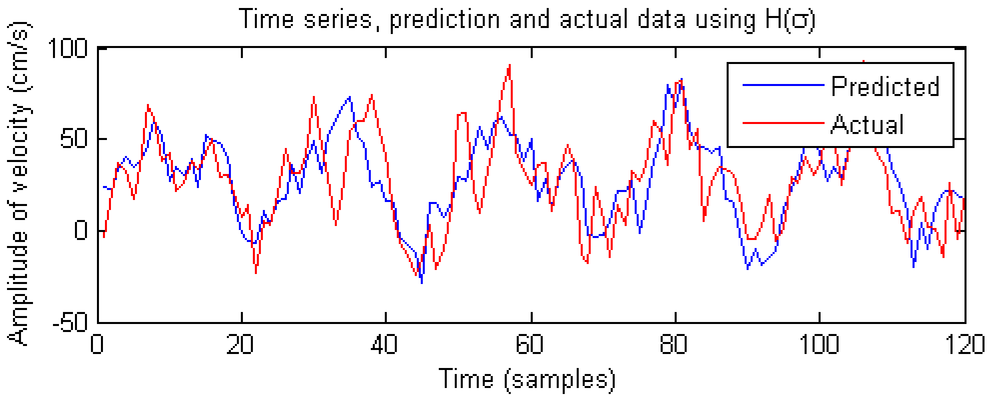
Part 2D - Time series prediction

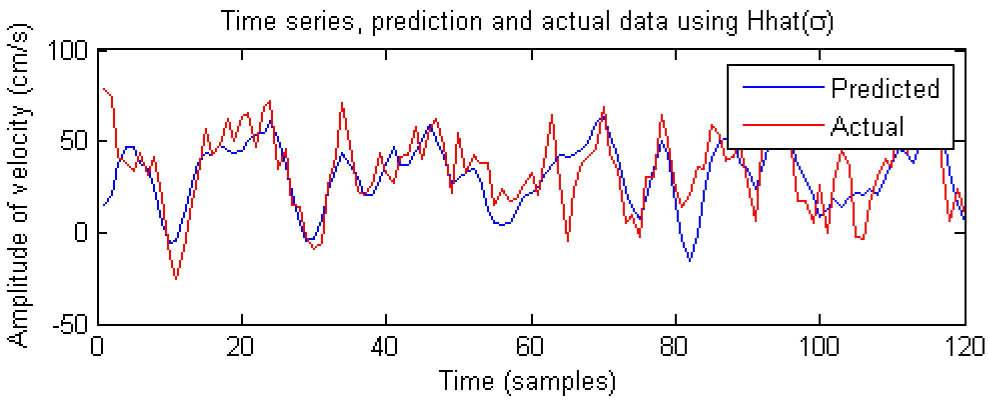
Note that using the "zeroed" version of H and Hhat increased the error, so they aren't used here. (This result is surprising.)

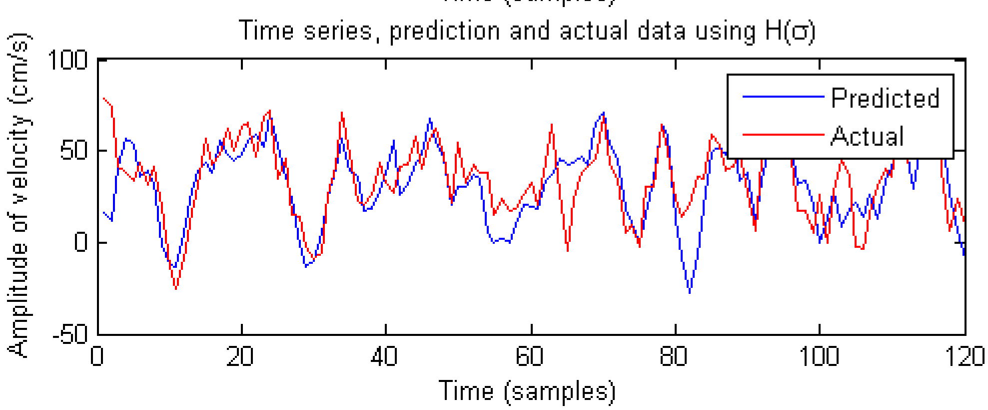
```
% You will probably have broken the original data series into 25
```

```
% segments to calculate H(! ). So go back through the 25 segments,
% Fourier transform the velocity data at range “90” (the forcing input),
% multiply by the 50 dof H(! ), and then inverse-transform to get the
% predicted record as a function of time at range “80”.
% Hint: Take care to treat the negative frequencies of the input data and of
% H(! ) correctly, so that your prediction is real. The inverse-transform
% output should have a very small imaginary part if this is done correctly.

% Testing:
%H_100_90 = ones(1,length(H_100_90));

% Get the 25 segments
number_of_sets = 25;
set_size = floor(length(velocity_90) / number_of_sets);
 for i=1:number_of_sets
        start_idx = (i-1) * set_size + 1;
        end_idx = start_idx + set_size - 1;

        data = velocity_90(start_idx:end_idx);
        a_90(:,i) = fft(data');
        a_predicted_80 = a_90(:,i)' .* H_hat;

        a_predicted_80_better = a_90(:,i)' .* H;

        predicted_80(:,i) = ifft(a_predicted_80');
        predicted_80_better(:,i) = ifft(a_predicted_80_better');

        error(:,i) = predicted_80(:,i) - velocity_80(start_idx:end_idx);
        error_better(:,i) = predicted_80_better(:,i) - velocity_80(start_idx:end_i


        if ((i==5) || (i==20))
            figure();
            subplot(2,1,1);
            plot(1:length(predicted_80(:,i)), predicted_80(:,i))
            hold on;
            plot(velocity_80(start_idx:end_idx), 'r')
            hold off;
            legend('Predicted', 'Actual');
            xlabel('Time (samples)');
            ylabel('Amplitude of velocity (cm/s)');
            title('Time series, prediction and actual data using Hhat(\sigma)');
            subplot(2,1,2);
            plot(1:length(predicted_80_better(:,i)), predicted_80_better(:,i))
            hold on;
            plot(velocity_80(start_idx:end_idx), 'r')
            hold off;
            legend('Predicted', 'Actual');
            xlabel('Time (samples)');
            ylabel('Amplitude of velocity (cm/s)');
            title('Time series, prediction and actual data using H(\sigma)');
        end

 end
```


Time series, prediction and actual data using Hhat(σ)
Amplitude of velocity (cm/s)
Time (samples)
Predicted
Actual

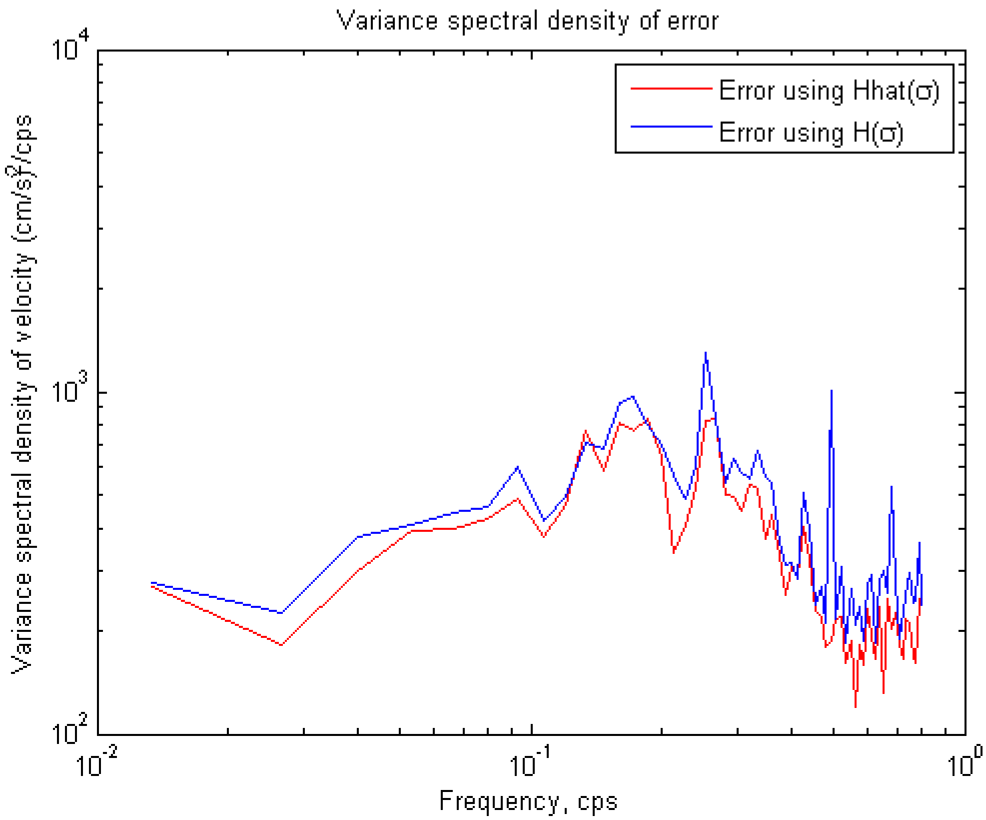

Time series, prediction and actual data using H(σ)
Amplitude of velocity (cm/s)
Time (samples)
Predicted
Actual


Time series, prediction and actual data using Hhat(σ)
Amplitude of velocity (cm/s)
Time (samples)
Predicted
Actual


Time series, prediction and actual data using H(σ)
Amplitude of velocity (cm/s)
Time (samples)
Predicted
Actual

Part 2E

```
 %  Quantify the accuracy of the prediction by forming time series of
% Error (t) = predicted(t) –velb4(80,t)
% in each of the 25 segments.
% Form power spectra of
% Error(! ) and velb4(80, ! ) at 50 dof. Plot log-log on the same plot.
% (This is the noise-to-signal ratio)
% Also, for two of your sub-records, plot velb4(80, t) and predicted(t) on
% the same plot. Which aspects of the wavefield are well-predicted which
% are not?

error = reshape(error, [], 1,1);
error_better = reshape(error_better, [], 1,1);
[ehat_error_real, ehat_error, error_freq_bins_real, error_freq_bins, error_delta_f
    varspec_est_multidof(error, sample_interval, dof);
[ehat_error_better_real, ehat_error_better, error_better_freq_bins_real, ...
    error_better_freq_bins, error_delta_f] = ...
    varspec_est_multidof(error_better, sample_interval, dof);

[ehat_80_real, ehat_80, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_multidof(velocity_80, sample_interval, dof);

figure();
loglog(freq_bins_real, ehat_error_real, 'r');
hold on;
loglog(freq_bins_real, ehat_error_better_real, 'b');
legend('Error using Hhat(\sigma)', 'Error using H(\sigma)');
xlabel('Frequency, cps');
ylabel('Variance spectral density of velocity (cm/s)^2/cps');
title('Variance spectral density of error');
```



Part 2F

```
disp('Accounting for measurement noise seems to have hurt our prediction.');
disp('This is almost certainly an error somwhere in this program that I');
disp('have not yet identified.  (Particularly since the time series');
disp('does look improved.)');
disp('The wind waves are affected more than the swell because the data');
disp('at those frequencies have a lower signal-to-noise ratio.  Thus,');
disp('accounting for some of the noise will have a proportionally');
disp('greater effect.');
```

Accounting for measurement noise seems to have hurt our prediction.
This is almost certainly an error somwhere in this program that I
have not yet identified. (Particularly since the time series
does look improved.)
The wind waves are affected more than the swell because the data
at those frequencies have a lower signal-to-noise ratio. Thus,
accounting for some of the noise will have a proportionally
greater effect.

```
end


function [cross_spec, freq_bins] = crossspectrum(data_a, data_b, sample_interval,
    % We get 2 DOF from each set of data (thanks to real/imag components.
    num_sets = ceil(dof / 2);
```

```
    len_complete = size(data_a, 1);
    points_per = floor(len_complete / num_sets);

    delta_f = (1/sample_interval) / points_per;  % same as fs / N
    freq_bins = (points_per/2 * delta_f + delta_f):delta_f:(points_per/2 * delta_f

    % Get the variance spectral densitity for each subset.
    for i=1:num_sets
        start_idx = (i-1) * points_per + 1;
        end_idx = start_idx + points_per - 1;

        a_a(:,i) = fft(data_a(start_idx:end_idx));
        a_b(:,i) = fft(data_b(start_idx:end_idx));

        crosses(i,:) = bsxfun(@times, conj(a_a(:, i)), a_b(:,i));
    end

    % normalize FFT by n^2
    % Get the average values (average of columns)
    cross_spec = mean(crosses) ./ delta_f ./ points_per^2;
    %cross_spec = reshape(cross_spec, [], 1, 1);

    % get rid of the first (mean) value.

end
```

Variance Spectral Density Functions

Take the given data and provide a variance density estimate The freq_bins array will be in reciprocal units to the units of sample_interval. ehat has units of (data_units^2) per (1/sample_interval_units)

```
function [ehat_real, ehat, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_real(data, sample_interval)

    % First, do the FFT to get an array of Fourier coefficients
    a = fft(data);

    % Figure out our frequency bins.
    N = length(data);
    delta_f = (1/sample_interval) / N;  % same as fs / N
    freq_bins_real = 0:delta_f:(N/2 * delta_f);
    freq_bins = (-N/2 * delta_f + delta_f):delta_f:(N/2 * delta_f);
    %freq_bins = 0:delta_f:N*delta_f - delta_f;

    % We are dealing with a real signal, so we only care about positive, real
    % freqency components.  To get the power, we want to "fold" both sides of
    % the spectrum together.  Since they are the same, we can just take one
    % side and multiply by 2.
    % To preserve variance, we have to normalize this result by dividing by
    % N^2, due to the Matlab implementation of fft.
    ehat_real = 2 * abs(a(1:length(freq_bins_real))).^2 ./ delta_f ./ ...
        (N^2);
```

```
    % Also, figure out the full spectrum
    ehat = abs(a(1:length(freq_bins))).^2 ./ delta_f ./ (N^2);

    % The first frequency bin will hold the mean, not a component of the
    % variance.  Since we are interested in the variance spectrum, drop
    % the first bin.
    %ehat = ehat(2:end);
    ehat_real = ehat_real(2:end);
    freq_bins_real = freq_bins_real(2:length(ehat_real)+1);

end

function [ehat_real, ehat, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_avg(data, sample_interval, set_size, number_of_sets)

    % Get the variance spectral densitity for each subset.
    for i=1:number_of_sets
        start_idx = (i-1) * set_size + 1;
        end_idx = start_idx + set_size - 1;

        [ehat_parts_real(i, :), ehat_parts(i, :), freq_bins_real, freq_bins, delta
            varspec_est_real(data(start_idx:end_idx), sample_interval);
    end

    % form the average
    % Get the average values (average of columns)
    ehat = mean(ehat_parts);
    ehat_real = mean(ehat_parts_real);

end

function [ehat_real, ehat, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_multidof(data, sample_interval, dof)

    % We get 2 DOF from each set of data (thanks to real/imag components.
    num_sets = ceil(dof / 2);

    len_complete = size(data, 1);
    points_per = floor(len_complete / num_sets);

    [ehat_real, ehat, freq_bins_real, freq_bins, delta_f] = ...
        varspec_est_avg(data, sample_interval, points_per, num_sets);
end
```

Published with MATLAB® 7.12