Table of Contents

SIO221A Homework #6 (Eric Gallimore)

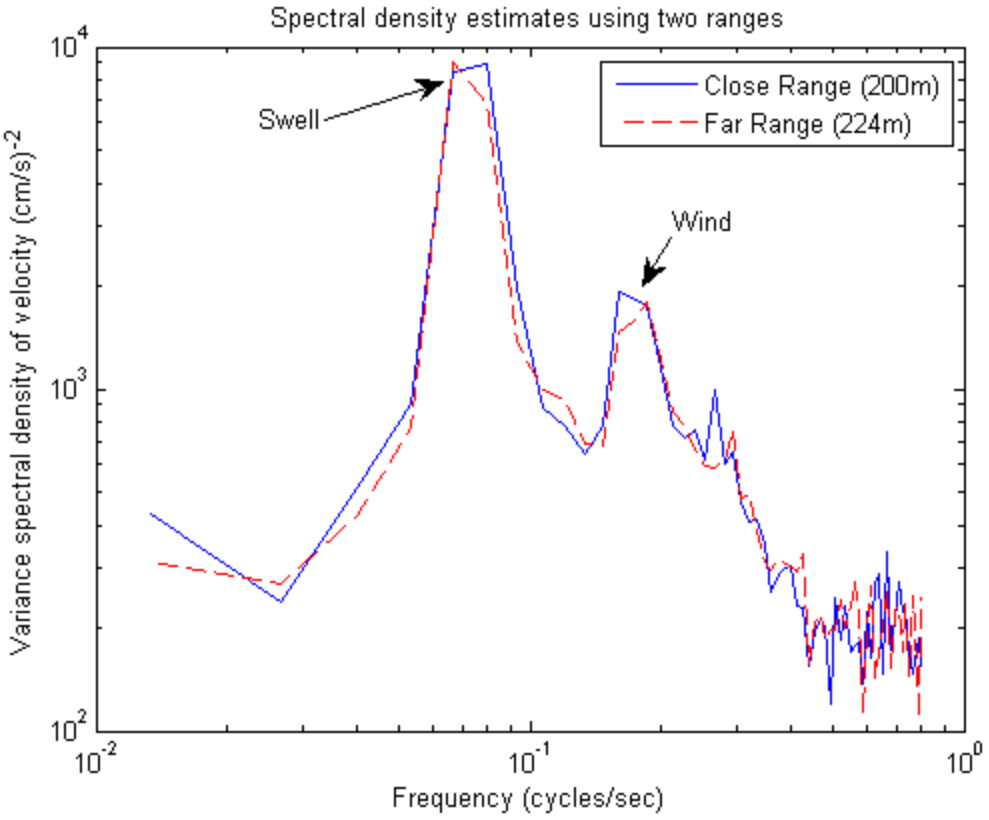
```
function hw6()

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);

for i=1:dof
    records.velocity(:,:,i) = velb4(:,(i-1)*points_per+1:i*points_per);
    records.time = 0:0.625:0.625*points_per; % times
    records.space = 0:2:2*219; % spacing
end
```

Task 1: Present power spectral estimates of the horizontal velocity

Ehat(sigma) at 50 degrees of freedom for two ranges separated by ?r=24m. Plot log-log on the same plot. Identify the swell peak, wind-wave peak. State the frequency in cycles/s, please.

```
% Get two records, 24m apart.
% From previous problems, we know that data in the spatial range 10:201 is
% good.  Each row is separated by 2m.  So, pick two rows 12 indexes apart.
% Index 10 = 20m (or maybe more... don't know what index 1 is, actually.)
% index 22 = 44m
%velocity_close(:,:) = records.velocity(10, :, :);
%velocity_far(:,:) = records.velocity(22, :, :);


% Get two records, 24m apart.
% From previous problems, we know that data in the spatial range 10:201 is
% good.  Each row is separated by 2m.  So, pick two rows 12 indexes apart.
% Index 100 = 200m (or maybe more... don't know what index 1 is, actually.)
% index 112 = 124m
% Both records will use the same frequency spacing, so we don't need
```

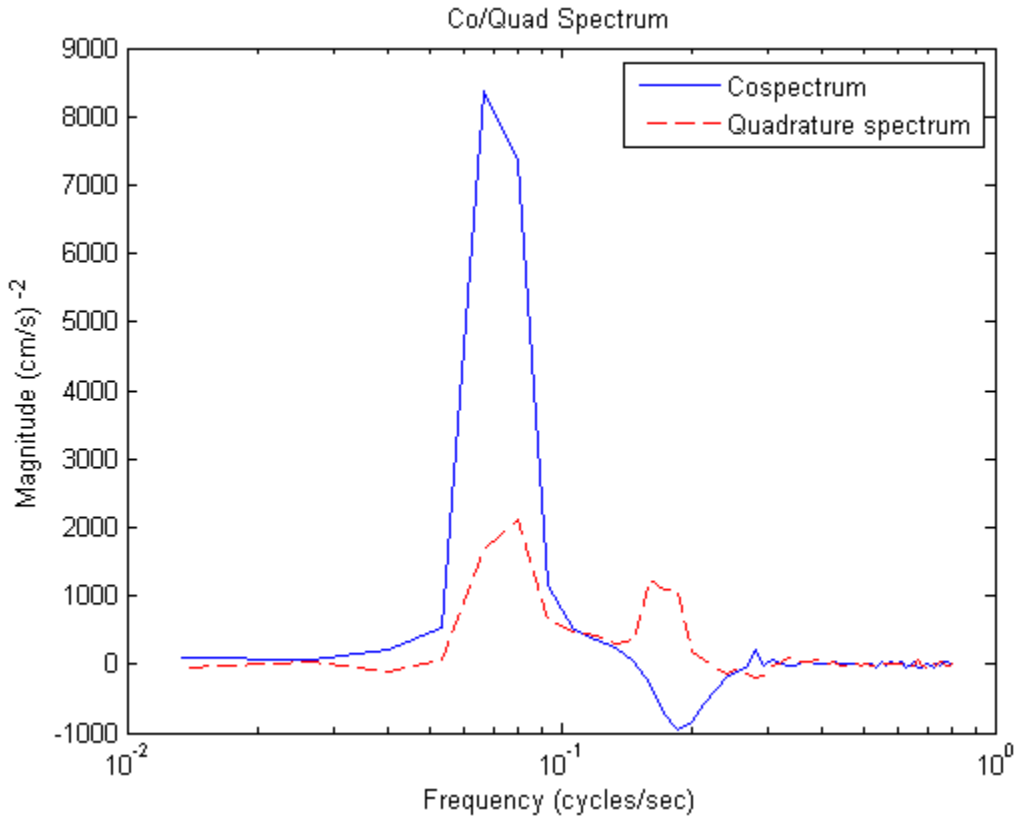
```
% different variables for that.
sample_interval = 0.625; % seconds
velocity_close = reshape(velb4(100,:), [], 1);
velocity_far = reshape(velb4(112,:), [], 1);
[ehat_close_real, ehat_close, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_multidof(velocity_close, sample_interval, dof);
[ehat_far_real, ehat_far, freq_bins_real, freq_bins, delta_f] = ...
    varspec_est_multidof(velocity_far, sample_interval, dof);

figure();
clf();
loglog(freq_bins_real, ehat_close_real);
hold on;
loglog(freq_bins_real, ehat_far_real, 'r--');
% I don't think we have a single axis label here.
%ylabel('Variance of velocity (m^2/s * hours)')
xlabel('Frequency (cycles/sec)');
legend('Close Range (200m)','Far Range (224m)');
title('Spectral density estimates using two ranges');
ylabel('Variance spectral density of velocity (cm/s) ^-^2');
annotation(gcf(),'textarrow',[0.643378519290928 0.618352450469239],...
    [0.696581196581197 0.63960113960114],'TextEdgeColor','none',...
    'String',{'Wind'});
annotation(gcf(),'textarrow',[0.334723670490094 0.440041710114703],...
    [0.836606837606838 0.881766381766382],'TextEdgeColor','none',...
    'String',{'Swell'});
```



Task 2: Present co and quadrature spectral estimates at 50 d.o.f.

Plot log (frequency, cps) vs. linear ordinate.

```
[cross_spec, freq_bins] = crossspectrum(velocity_close, velocity_far, sample_inter

figure()
% I
co = 2.*real(cross_spec(1:60));
semilogx(freq_bins, co);
hold on;
% Q
quad = 2.*imag(cross_spec(1:60));
semilogx(freq_bins, quad, 'r--');
legend('Cospectrum', 'Quadrature spectrum');
xlabel('Frequency (cycles/sec)');
title('Co/Quad Spectrum');
ylabel('Magnitude (cm/s) ^-^2');
```



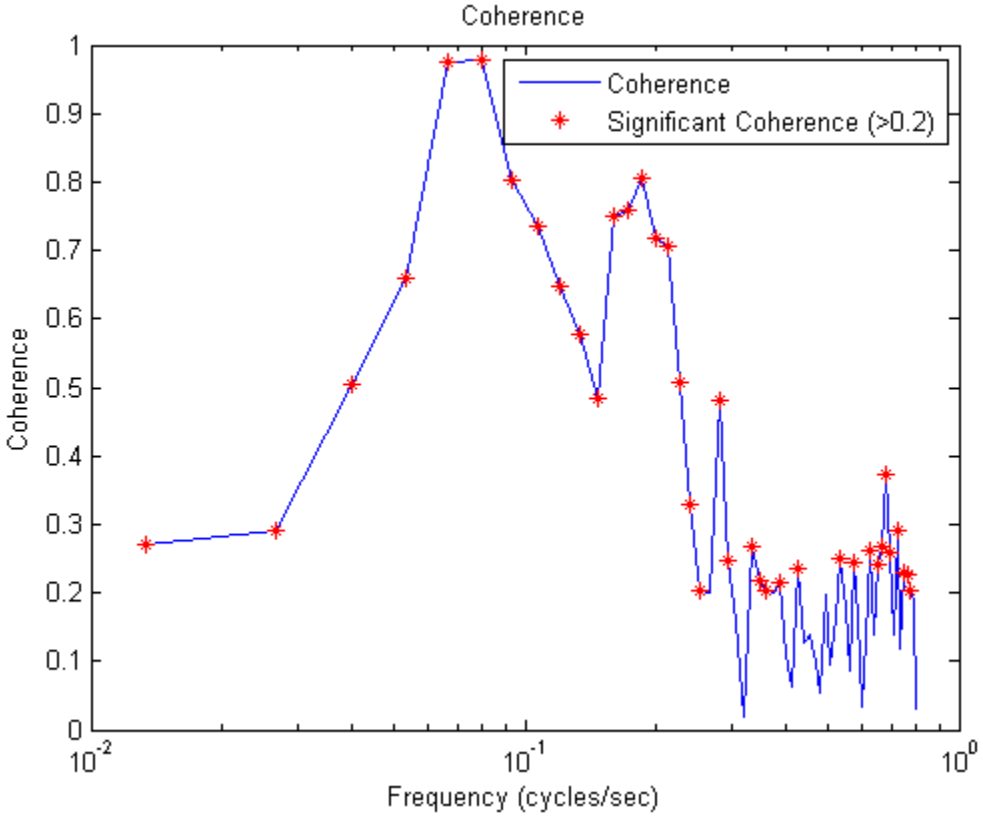
Task 3: Present estimates of coherence and phase (in degrees) at 50 DOF

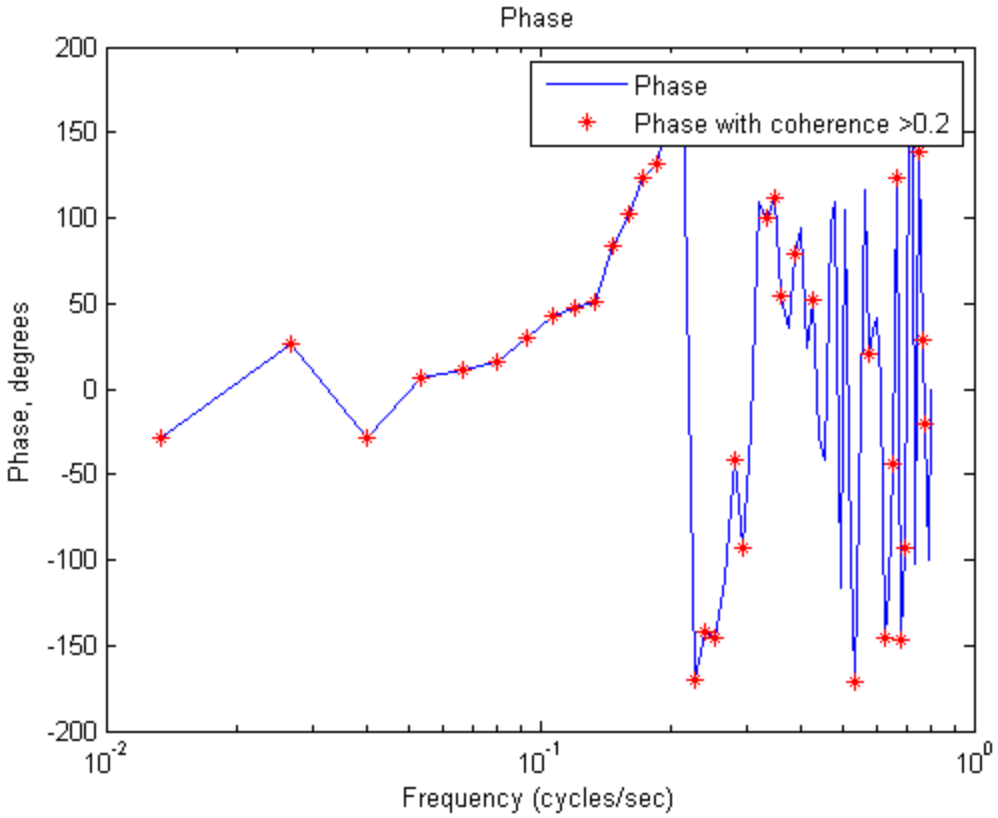
On the phase plot, re-plot the phase estimate in each frequency band as an asterisk (*), but only in those frequency bands where the coherence is significant at the 67% confidence level.

```
coherence = 2.*abs(cross_spec(1:60)) ./ sqrt(ehat_close_real .* ehat_far_real);

% angle(z) is the same as atan2(imag(z), real(z))
phase_radians = angle(cross_spec(1:60));
phase_degrees = phase_radians * (180/pi);

significant_coherence = coherence(coherence > 0.2);

figure()
semilogx(freq_bins, coherence);
hold on;
semilogx(freq_bins(coherence > 0.2), significant_coherence, 'r*');
legend('Coherence', 'Significant Coherence (>0.2)');
xlabel('Frequency (cycles/sec)');
title('Coherence');
% Coherence is unitless, since it is a PSD/PSD
ylabel('Coherence');



figure()
semilogx(freq_bins, phase_degrees);
hold on;
semilogx(freq_bins(coherence > 0.2), phase_degrees(coherence > 0.2), 'r*');
legend('Phase', 'Phase with coherence >0.2');
xlabel('Frequency (cycles/sec)');
title('Phase');
ylabel('Phase, degrees');



disp('Positive phase corresponds to waves arriving at the near receiver');
disp('after the far receiver.  So, if the range of the receivers increases');
disp('as we go west, that means that the waves are moving east.');
fprintf('\n');
disp('The wavenumber bandwidth is inversely proportional to the coherence.');
disp('Over frequencies where the coherence is low, the dispersion relation');
disp('not as well defined... that is, the energy in the wavenumber spectrum');
disp('is spread (looks less like a delta function).  Where the coherence is');
disp('high, the energy is tightly constrained within wavenumber "bins".');
```

```
Positive phase corresponds to waves arriving at the near receiver
after the far receiver.  So, if the range of the receivers increases
as we go west, that means that the waves are moving east.

The wavenumber bandwidth is inversely proportional to the coherence.
```

Over frequencies where the coherence is low, the dispersion relation not as well defined... that is, the energy in the wavenumber spectrum is spread (looks less like a delta function). Where the coherence is high, the energy is tightly constrained within wavenumber "bins".

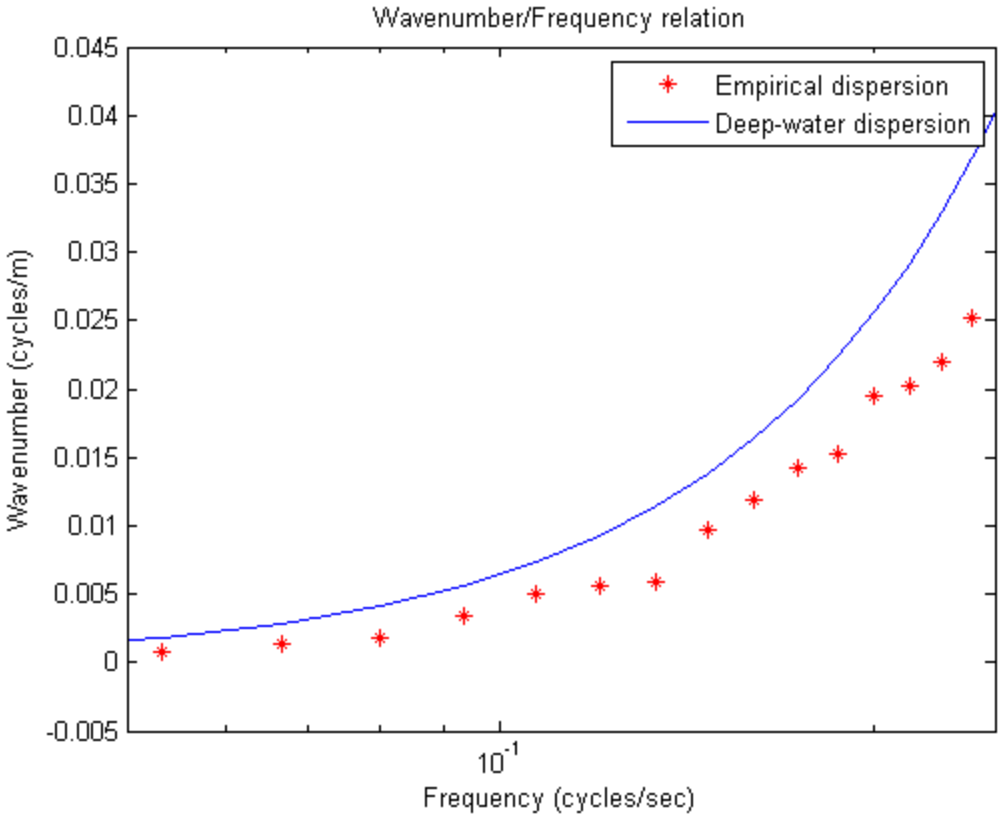




Task 4/5

the 24 comes from the 24m spacing. The 2pi is to get it in cycles per meter. We need to "unwrap" the phase, which we can do because we have an idea of what is going on here, so we know it is just aliased.

```
wavenumbers = unwrap(phase_radians(coherence > 0.2)) / (2*pi*24);

figure()
semilogx(freq_bins(coherence > 0.2), wavenumbers, 'r*');
hold on;

k = 2*pi*(freq_bins).^2 / 9.8;
semilogx(freq_bins, k);

% zoom in on the area of interest
xlim([0.05, 0.25]);

legend('Empirical dispersion', 'Deep-water dispersion');
xlabel('Frequency (cycles/sec)');
title('Wavenumber/Frequency relation');
ylabel('Wavenumber (cycles/m)');
```




```
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 = 0: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;
```

```
    % get rid of the first (mean) value.
    cross_spec = cross_spec(2:end);
    freq_bins = freq_bins(2:end);

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 = 0:delta_f:(N * 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)-1)).^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 = freq_bins(2:end);
    freq_bins_real = freq_bins_real(2:end);

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