L22: class exercises – Try to implement/solve the following problems in MATLAB.

Time series

In this step you will create your own time series that contains multiple periodic components, a linear trend and noise. Create a time vector from 0:1000 with a time step of dt=1.

Now create a periodic signal (x_p) composed of three sine functions with 1) A1=2, f1=1/50; 2) A2=1, f2=1/15, and 3) A3=0.5, f3=1/5 summed together. For example, sine wave one would be A1*sin(2*pi*f1*t). Plot this wave and label the axes.

Next, create a linear trend (x_tr) that has a slope of 0.005 and y-intercept at 0. Add this to the x_p curve and plot on top of the individual x_p curve. Do you notice a linear trend?

Finally, generate zero-mean random Gaussian noise with **randn()**. The noise should have a standard deviation of one. *Note: Prior to calling **randn()**, make sure to set **rng(0)** so that everyone in class has the same random noise.* Add this noise to the periodic and trend data so that you have a time series with all three components. Plot this data on top of the periodic only signal.

You should now have three signals.

1. $x_p(t)$ = periodic signal composed of three sine curves
2. $x_1(t) = x_p(t) + x_{tr}(t)$ = periodic plus linear trend
3. $x_2(t) = x_p(t) + x_{tr}(t) + x_n(t)$ = periodic plus linear trend plus noise

1 Periodogram – Power Spectral Density

Use **nextpow()** to find the next power of two of the number of data points. Set **nfft=**$2^{nextpow(N)}$, where N is the number of data. Then call **[Pxx,f]=periodogram(x_p,[],nfft,fs)**, where *nfft* is the number of points in the fft (an *even* number of points makes the fft symmetric and using powers of two enables this to be done easily). x_p is the periodic only signal; *fs* is the sampling frequency 1/dt. Then plot the output with the y-limits set from 0:1000;

```
nfft = 2^nextpow2(N);
[Pxx,f] = periodogram(x_p,[],nfft,fs);

figure;
plot(f,Pxx); grid on;
xlabel('Frequency [Hz]' ylim([0 1000]););
ylabel('Power');
title('Auto-Spectrum');
ylim([0 1000]);
```

Describe the output. What do you notice about the upper frequency limit?

Repeat the periodogram computation and plotting using 1) the periodic + trend signal and 2) the periodic, trend and noise signal. Describe the differences in the plots.

Now vary the noise from a standard deviation of 1 to 2 to 3 to 4. What happens to you plots?

2 Data detrending

We previously used the **polyfit()** and **polyval()** functions to remove the trend in the GPS data in HW8. MATLAB has another built in function called **detrend()** that we will now use. **detrend()** has two options: 'constant' (polynomial order 0) and 'linear' (polynomial order 1). Apply the **detrend()** function to each of your three signals two times in order to remove any constant (i.e. DC) and linear trends in the data. Replot the periodogram and compare with the previous non-detrended periodogram. What happened?

3 Cross Spectral Density and Coherence

Use two sine waves with identical periodicities τ=5 (equivalent to f3=0.2=1/5 in previous example) and amplitudes equal to two. Add a phase shift to the second time series so that one signal has a phase lead of $2\pi/5$.

```
psi = 2*pi*f3; % a phase shift term

x = 2 * sin( 2*pi*f3*t      ); % signal without phase shift
y = 2 * sin( 2*pi*f3*t + psi); % signal with phase shift (actually a phase lead)
```

To determine the coherency between these two signals use **mscohere()**.

```
[Cxy,f] = mscohere(x,y,[],0,nfft,fs);

figure;
plot(f,Cxy); grid on; ylim([0 1.1]); xlim([0 0.5]);
xlabel('Frequency');
ylabel('Coherence');
title('Coherence');
```

What does this result tell us about each frequency in the two time series?

Now compute the cross-spectrum of the two time series with **cpsd()**.

```
[Pxy,f] = cpsd( x, y, [], 0, nfft, fs );

amp   = abs(Pxy);
phase = angle(Pxy);

figure;
subplot(1,3,1)
plot(t,x,'b-',t,y,'r-'); grid on;
axis([0 50 -2 2])

subplot(1,3,2)
plot( f, amp ); grid on; xlim([0 0.5]);
xlabel('Frequency')
ylabel('Power [a.u.]')
title('Cross-Spectrum')

subplot(1,3,3)
plot( f, phase ); grid on; xlim([0 0.5]);
xlabel('Frequency')
ylabel('Phase Difference [rad]');
title('Cross-Spectrum')
```

What do these plots tell you?

The phase difference

The phase difference plot must be interpreted with the *Power* in mind. The *Power* plot shows the amplitude at each frequency that these two time series have in common. Based on the time series, what frequency should the power be max? Which frequency(ies) should the power be small.

Use **interp1** to interpolate the phase function at exactly 0.2 Hz.

```
psi2 = interp1( f, phase, 0.2 ); % find the value at 0.2
```

Compare this to the original phase shift you added to the signal, *psi*. Do they match?

It is even more interesting to convert this phase shift to a time shift and compare the original time series data. To convert the phase difference to a time shift just divide by 2π. Does this time shift estimate match with the time shift you observe between the two signals? Check the peaks for instance.

More complicated data

Load the *series1.txt* and *series2.txt* from the L21. Remember that the first column contains ages in kiloyears, which are unevenly spaced. The second column contains oxygen-isotope values measured on calcareous microfossils (foraminifera). The data sets contain 100, 40 and 20 kyr cyclicities and they are overlain by Gaussian noise. In the 100 kyr frequency band, the second data series (series 2) is shifted by 5 kyrs with respect to the first data series (series 1).

You will need to first make sure to interpolate these data to a regular sampling in time. Then compute the auto-spectra (i.e. periodogram) of each time series and plot. What are the periods that you see?

Compute the cross spectra and analyze the phase differences at each of the dominant periods. Do you recover a 5 kyr time shift in the 100 kyr frequency band?