

ENVS422: Data Analysis of Environmental Records

Week 5

Fourier smoothing, spectra, cross-spectra.

What happens if you extend the FFT with zeros?

```
t=[0:999]';  
x=sin(2*pi*t/100)*0.1+cumsum(randn(1000,1))/10; Create a time series  
length 1000
```



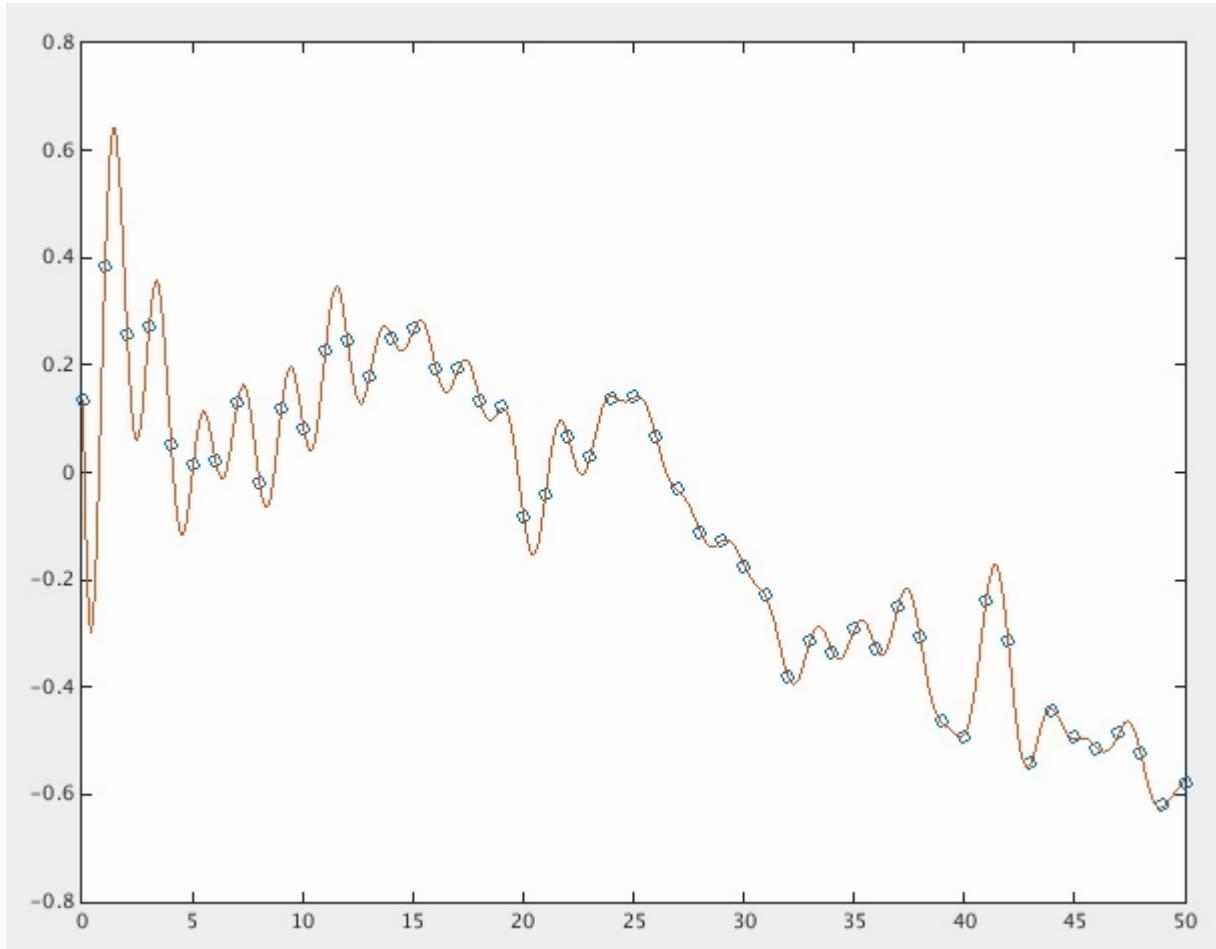
```
xft=fft(x);  
xfts=fftshift(xft); Calculate FT and shift so that zero  
frequency is at the centre
```



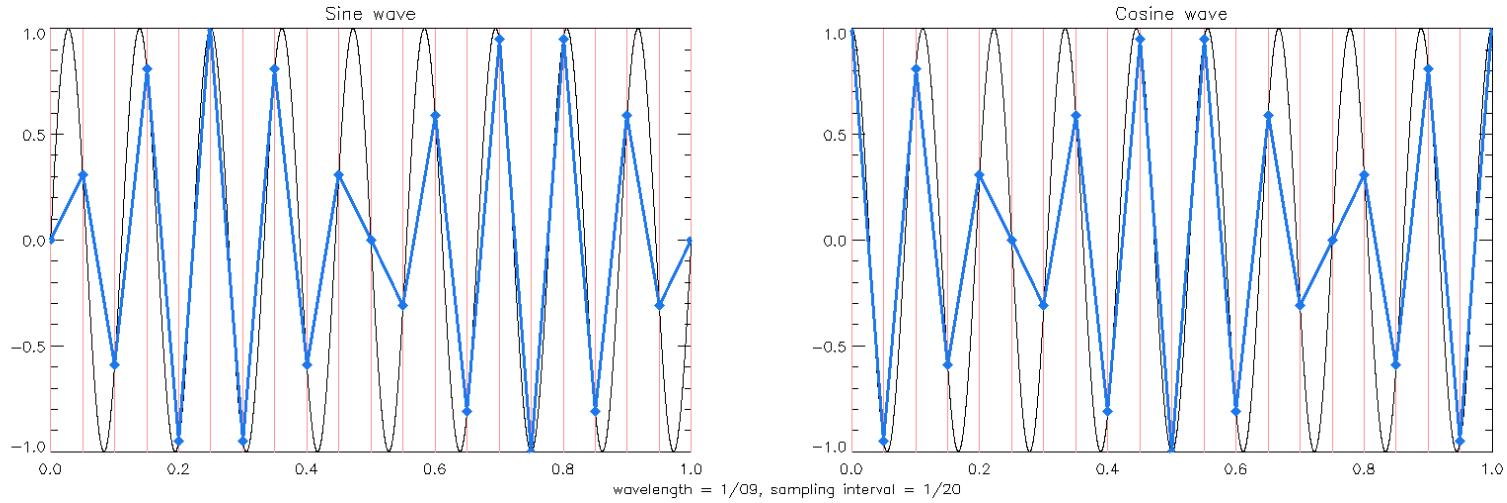
```
xftpad=[repmat(xfts*0,5,1);xfts;repmat(xfts*0,5,1)]*11  
xftpad=ifftshift(xftpad); Pad with zeros at each end so it is 11 times as long  
(and multiply by 11), then shift back so zero  
frequency is at the beginning
```



```
xx=ifft(xftpad,'symmetric'); Transform back into time domain, and create a  
corresponding time vector  
t1=[0:10999]'/11;  
hold off  
plot(t(1:51),x(1:51),'o') Plot the result  
hold on  
plot(t1(1:551),xx(1:551))
```



We have 11 times as many data points as before, but frequencies higher than the original Nyquist are set to zero. This is a way of interpolating data. However, it relies on the data being perfectly resolved (no aliasing), otherwise the peaks it invents between data points are fictitious.



If there really is no signal at periods shorter than the Nyquist period, this allows the reconstitution of the original (black) curves from the subsampled (blue) ones.

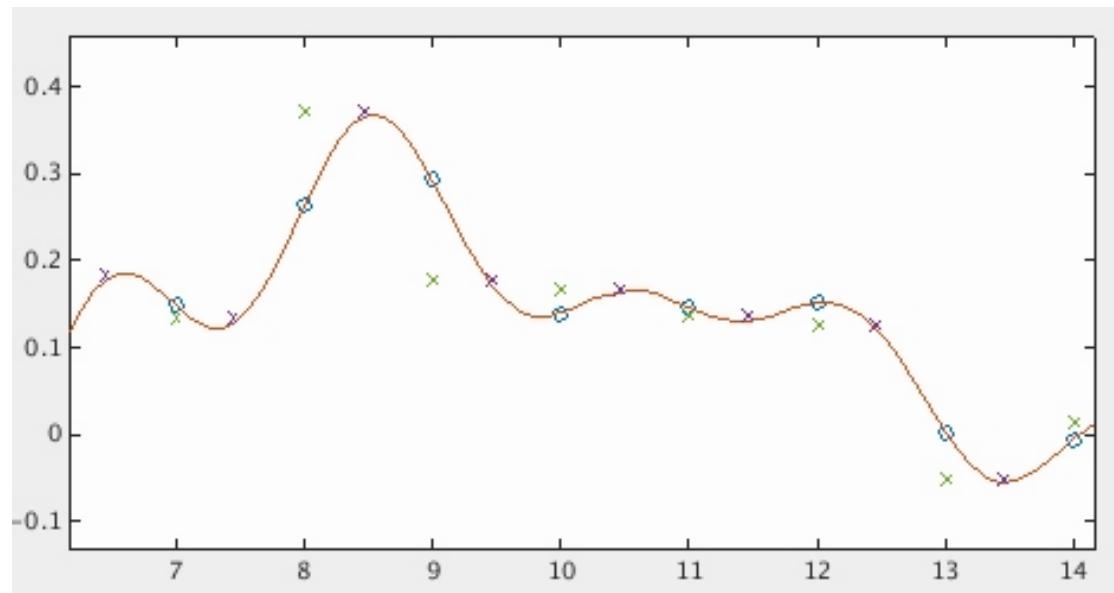
Unless you have already low pass filtered the data, or you are certain there is no energy at unresolved frequencies, this is a dangerous form of interpolation.

If you wanted to shift the time series in time, by a fraction of a sampling time step, you could do that by interpolating like this, then subsampling again at shifted times. However, there's a better way to do that: Shift the phases of all the sine and cosine terms by the right amount to move the time series.

```
function tser2=foushift(tser,rnum)
%
% returns tser2 which is time series tser, shifted
% along by rnum time intervals, where rnum
% need not be an integer. Does this by calculating FFT of time series
% and advancing the phase of each component by the appropriate amount.
%
n1=size(tser,1);
filt=complex(zeros(n1,1));
f1=(1:1+floor(n1/2 + 1))';
f1=exp(rnum*2*pi*f1*complex(0,1)/n1);
filt(1:1+floor(n1/2))=f1(1:1+floor(n1/2));
filt(n1/2+1:n1-1)=conj(fliplr(f1(1:floor((n1-1)/2))));
tser2=ifft(filt.*fft(tser),'symmetric');
end
```

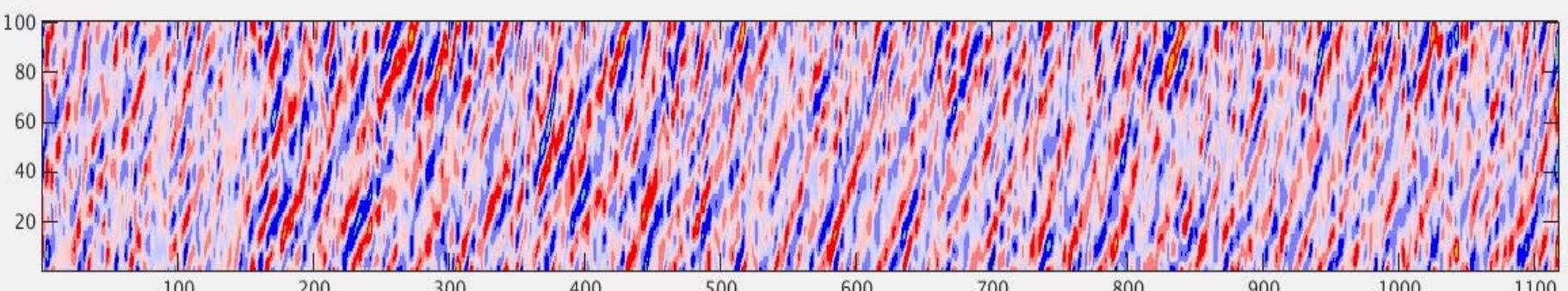
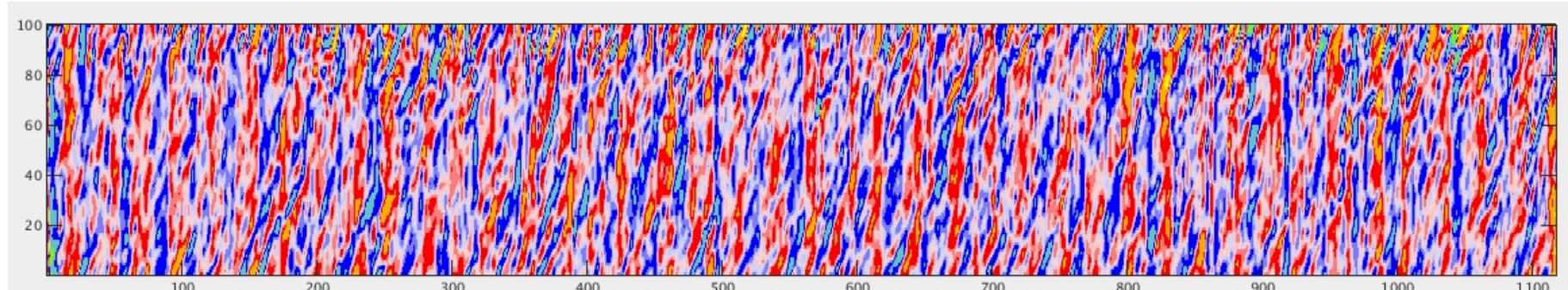
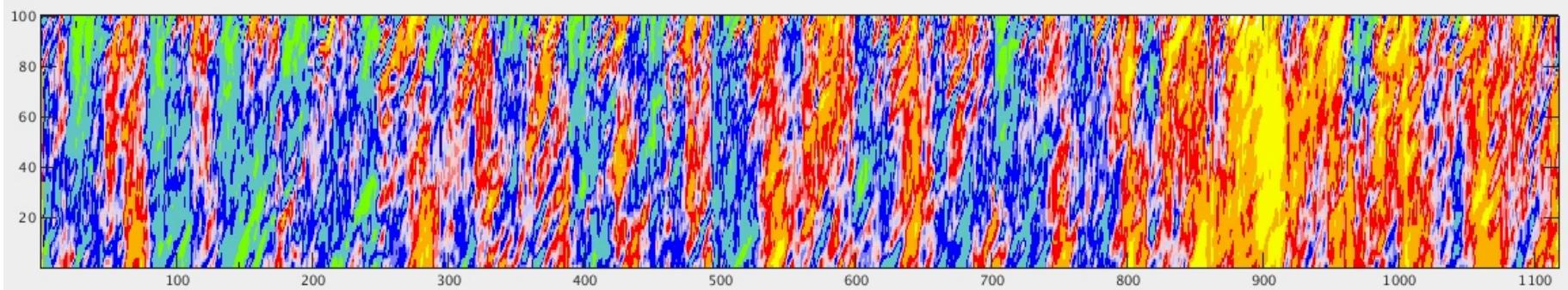
As before, take a time series tser (diamonds) and user FT to interpolate to 11 points for every 1 you had before (curve)

Then take the same time series and use phase shifting to move it by 5/11 time steps
tser2=foushift(tser,5/11)



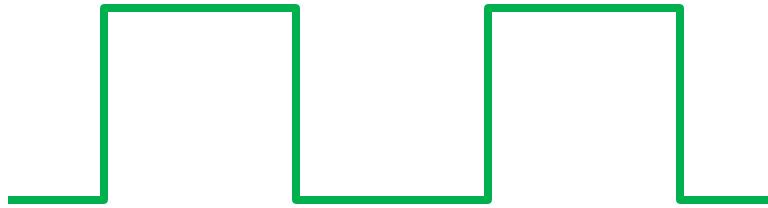
Plot the shifted time series (yellow x), and the shifted time series plotted with time also shifted so that it falls on the original (interpolated) curve.

Again, this is dangerous if you have energy at frequencies higher than the Nyquist.

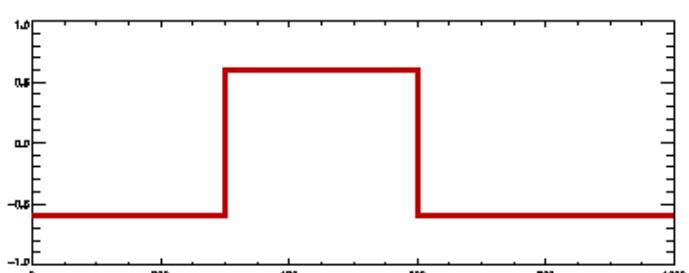
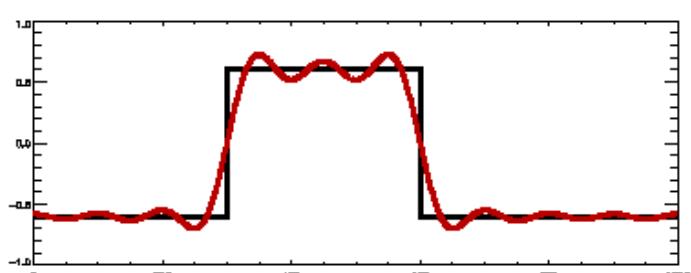
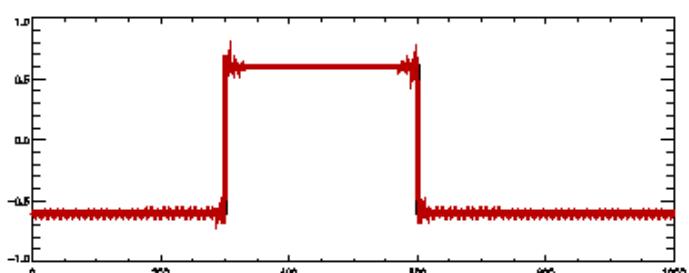
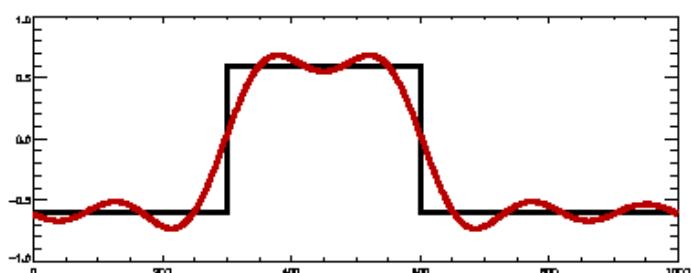
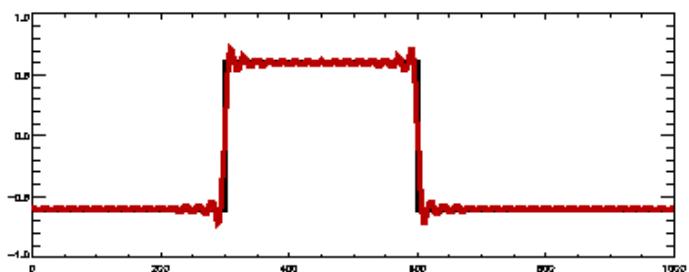
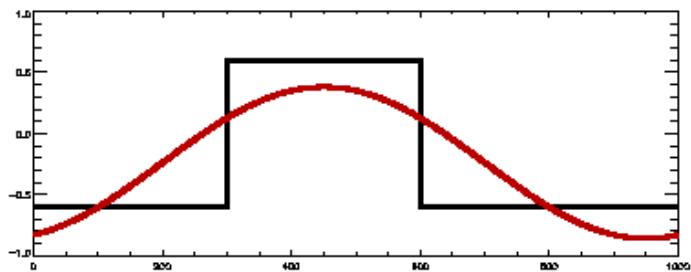


Recall how we used boxcar filters to extract particular ranges of period or wavelength. Boxcar filtering is not always the best way to filter (it depends on your application)...

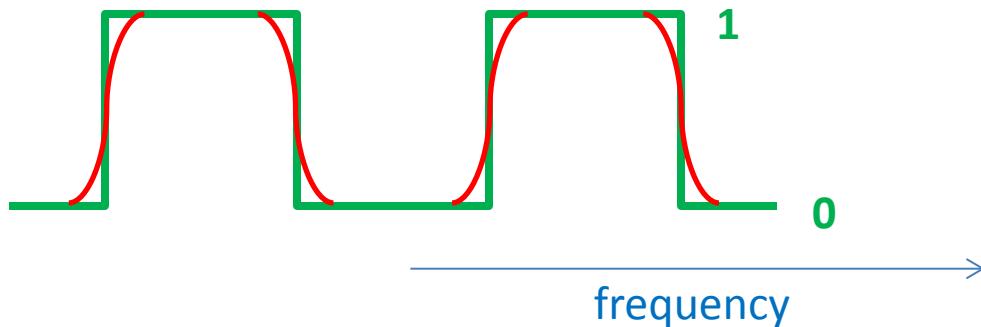
Advantages and disadvantages of boxcar filters



- They do literally extract a particular range of frequencies
- A timeseries $y(t)$ can be split into its filtered part $y_f(t)$, and the remainder $y_r(t) = y(t) - y_f(t)$. Then y_f and y_r remain orthogonal (uncorrelated).
- You might think that cutting off the high frequencies would be a good way of smoothing a time series, but as we saw before, sharp steps are not simply smoothed by doing this. We introduce “ringing”, or Gibbs’ phenomenon:

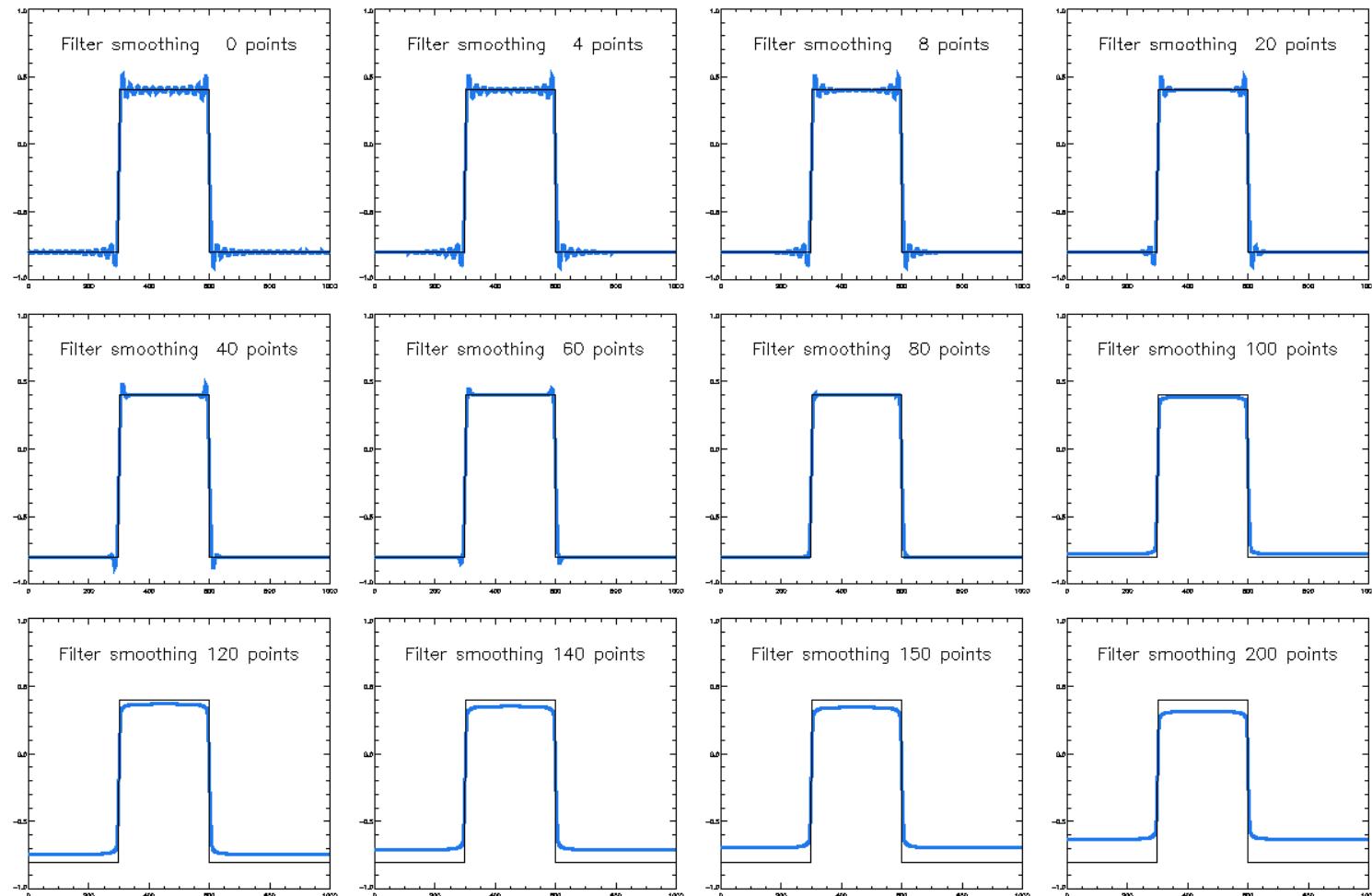


Rather than a boxcar filter, smooth off the edges:

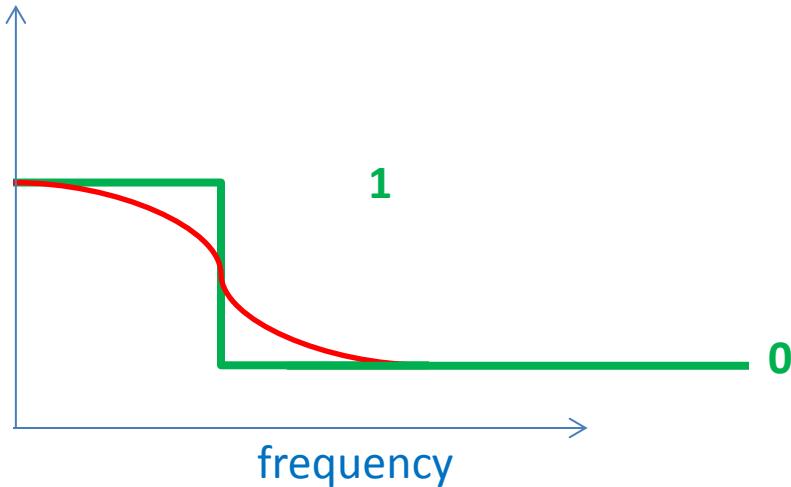


- Now the range of frequencies used is “blurred”
- The filtered part $y_f(t)$, and the remainder $y_r(t) = y(t) - y_f(t)$ are not now orthogonal (frequencies at which the filter is not 0 or 1 appear in both time series, with different amplitudes, so they will always be positively correlated).
- But this can be an effective way of reducing ringing, so that Fourier filtering acts like a smoother:

“top hat” time series, 1000 points long, smoothed with a filter which allows periods of 25 points or more through. Boxcar filter, and various smoothed filters.

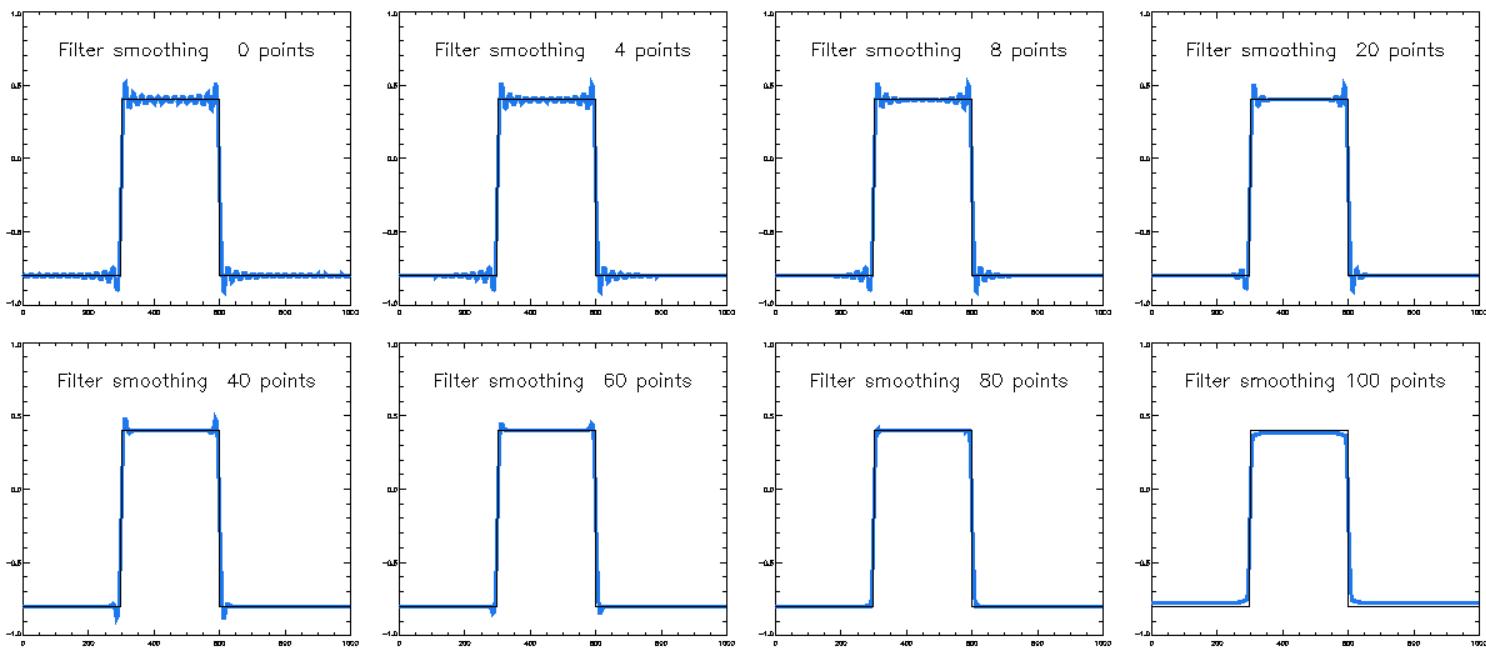


Note how too much tapering means that your filter isn't 1 anywhere, so even the longest period signal is reduced.



A good compromise (for this sinusoidal taper) occurs when the taper is just wide enough to make no difference at frequency zero (i.e. the half width of the taper is the same as the frequency of the low-pass filter. This gives small ringing, without reducing the amplitude of longer period oscillations.

Other filters use different shaped tapers with the aim of reducing the ringing more.



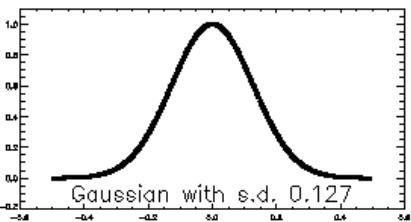
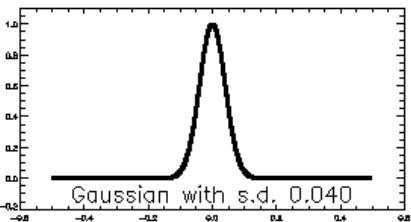
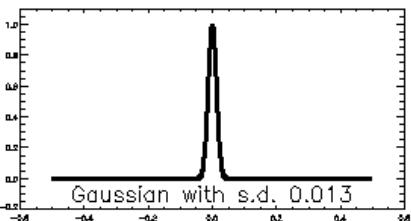
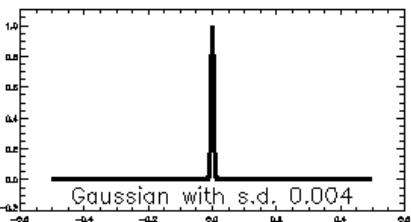
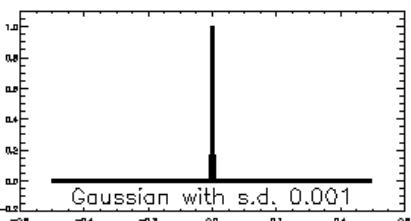
The sharper the cutoff of the filter in frequency, the further the ringing spreads from the step in time.

The FT of a delta function (spike) in time has equal amplitudes at all frequencies

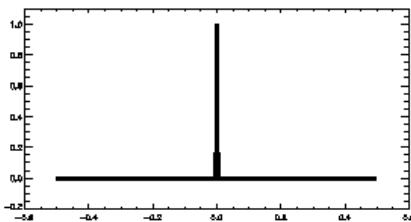
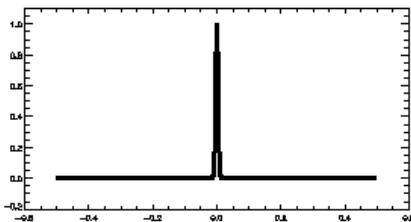
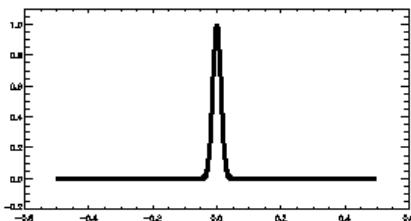
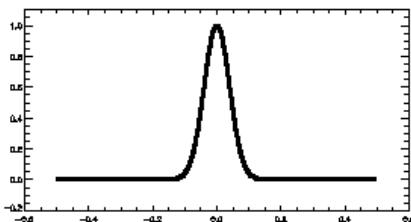
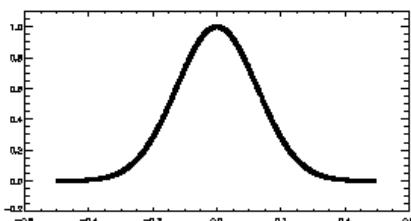
The inverse FT of a delta function in frequency is a pure sine wave in time – i.e. not localised at all.

The more you squeeze a signal to a narrow range of time, the wider the frequency range. The more you squeeze the frequency range, the wider the range in time. Like the uncertainty principle in quantum mechanics, you can't know the precise time and the precise frequency of a signal simultaneously.

Time series



Amplitude of FT



This is shown most clearly with the Gaussian

The FT of a Gaussian has amplitudes which are Gaussian (plotted here as amplitudes of the complex values, with frequency zero at the centre).

Narrow Gaussians in time translate to broad Gaussians in frequency, and vice versa.

$$W_t \times W_f = \text{constant}$$

where W_t is the standard deviation of the Gaussian in time, and W_f is the standard deviation of the Gaussian in frequency

This means:

If you want to know how the frequency content of a time series varies with time, you have to compromise.

You can chop the time series into lots of short segments, and calculate the FT of each, but you will only be able to divide it into broadly-spaced frequencies.

You can chop the time series into longer segments and calculate the FT of each. You will then have fine resolution in frequency, but will not be able to see rapid changes in the spectrum.

There is no way round this. It would make no sense to speak of a spectrum varying more rapidly than the sine waves it is made up of.

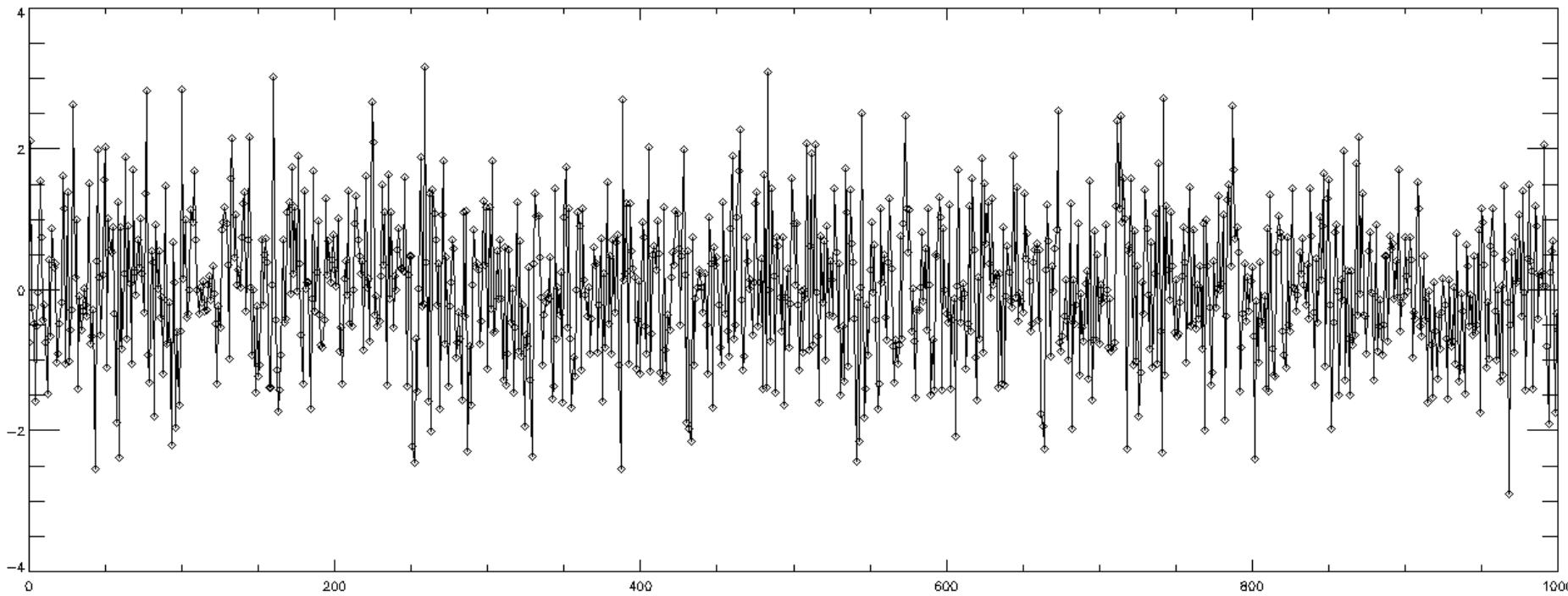
If you filter a time series with a very narrow band pass filter, the result will be a slowly modulated sine wave.

Two such filtered time series are much more likely to correlate by coincidence than unfiltered time series of the same length.

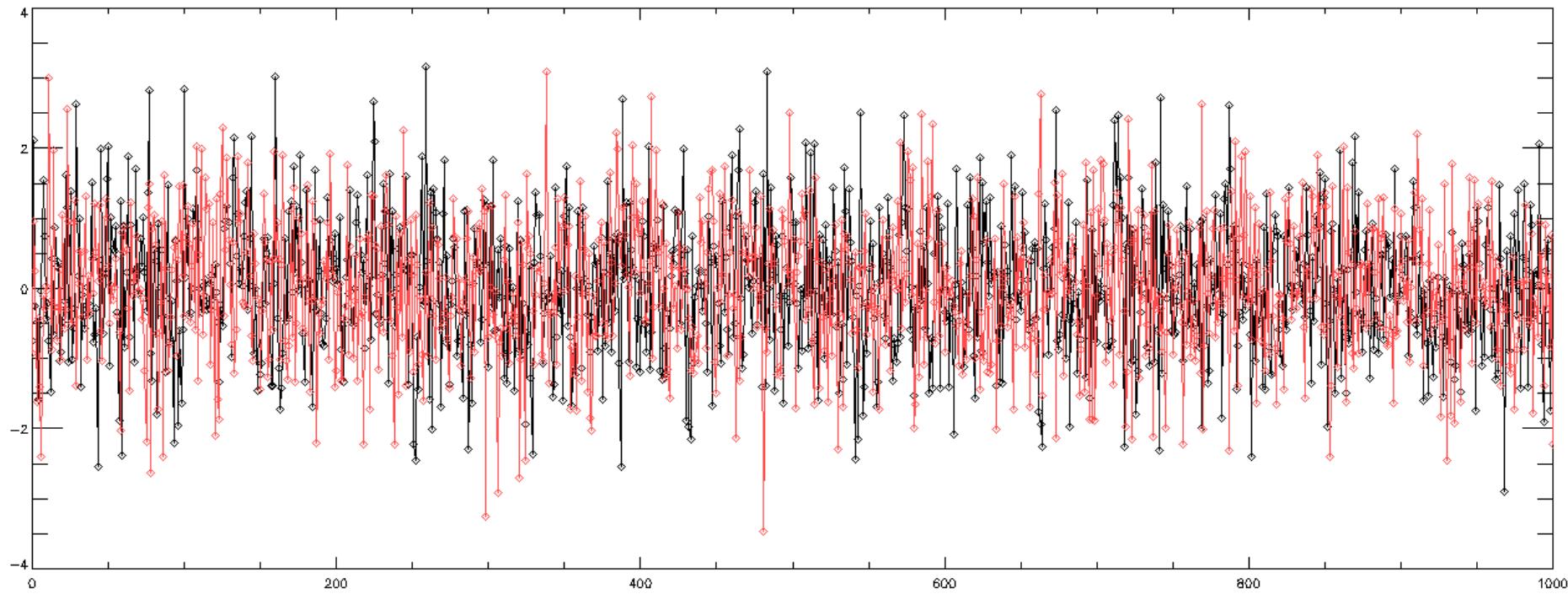
Spectra: white noise

- White noise is the same as Gaussian uncorrelated noise. Each value is randomly chosen independently of each other.
- Instrumental noise often has a white noise component, e.g. from thermal fluctuations in the electronics which are different for each measurement.
- No real processes can be “white” at all frequencies, because that would mean they would be oscillating infinitely fast (no relationship between the measurement at t and that at $t+\delta t$, however small δt is).
- In this sense, white noise represents a process with no memory.

Example of white noise with standard deviation 1



White noise includes all frequencies equally. It can also be generated by performing a Fourier transform on white noise



The same white noise time series as before
Real part of the FT of that time series, renormalized to standard deviation 1

Spectra: red noise

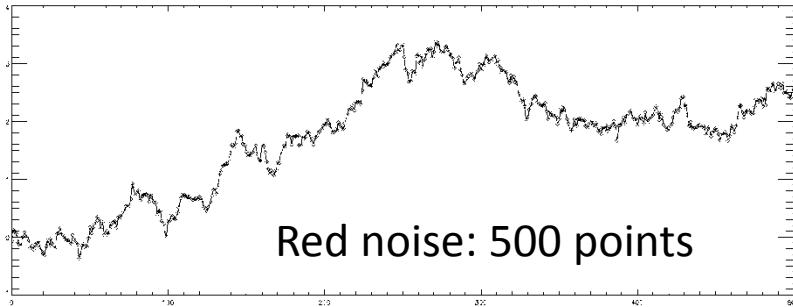
- Red noise is strongly correlated. It can be generated by making a time series in which each point is given by the previous one, plus a Gaussian random change. This is a form of “random walk”.
- This means that d/dt of red noise is white noise.
- Equally, it means that red noise can be thought of as the integral of white noise.

Spectra: red noise

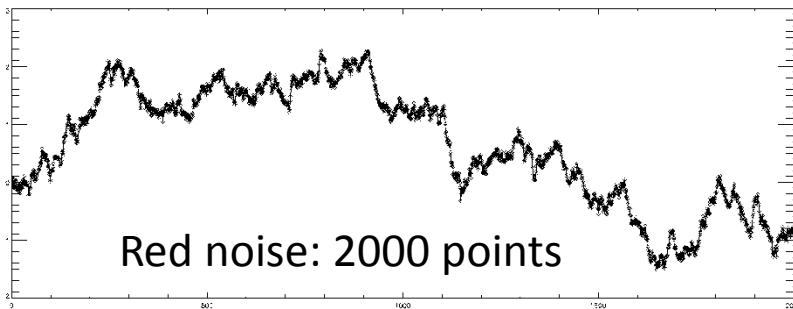
- d/dt of $\sin(2\pi ft)$ is $2\pi f \times \cos(2\pi ft)$
- d/dt of $\cos(2\pi ft)$ is $-2\pi f \times \sin(2\pi ft)$
- So differentiating is like multiplying the sine coefficients of the FT by $2\pi f$, and the cosine coefficients by $-2\pi f$ (you can actually differentiate like this: take the FT, multiply the coefficients using the different frequencies, and then take the inverse FT).
- Similarly, integrating is like dividing the sine coefficients of the FT by $2\pi f$, and the cosine coefficients by $-2\pi f$.

Spectra: red noise

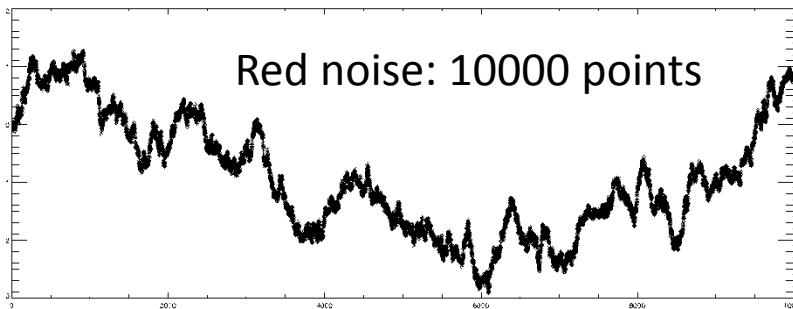
- If we can generate a white noise time series from an FT of white noise, then we can generate a red noise time series from an FT of white noise divided by frequency.
- It is an example of “power law” noise. The energy in a given frequency interval is proportional to a power of the frequency (-2 in the case of red noise, since the amplitudes were divided by f , and energy is proportional to amplitude squared).
- Red noise has peculiar properties: the random walk meanders ever further from its start point (expected distance is proportional to square root of time). This means its standard deviation gets bigger as the time series gets longer.
- Many geophysical (including ocean) processes look more like red noise than white noise.



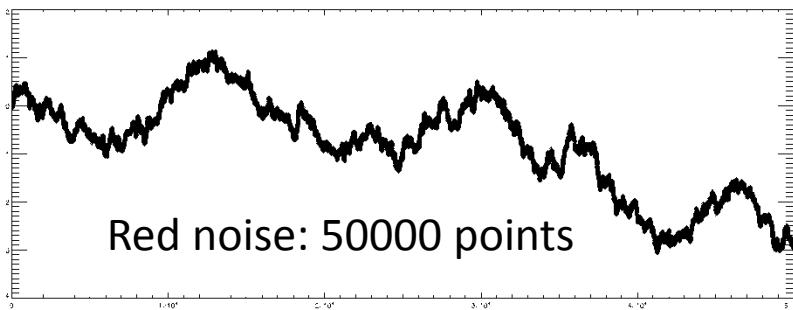
Red noise: 500 points



Red noise: 2000 points



Red noise: 10000 points



Red noise: 50000 points

Red noise is fractal – it looks the same at all magnifications

Each of these curves includes the previous one as its beginning.

White noise is also fractal.

So is any “power law” noise, in which the expected amplitude of each Fourier coefficient is proportional to frequency raised to a particular power.

Red noise appears dominated by the longest time scale available.

Other forms of power law noise can be generated from the Fourier transform of Gaussian white noise, multiplied by the frequency raised to the appropriate power.

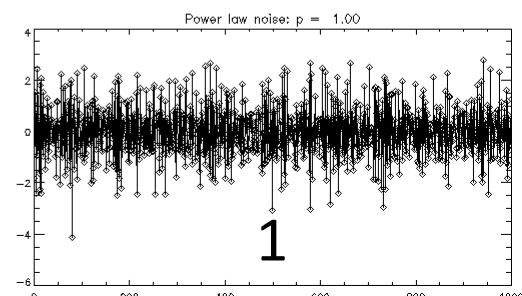
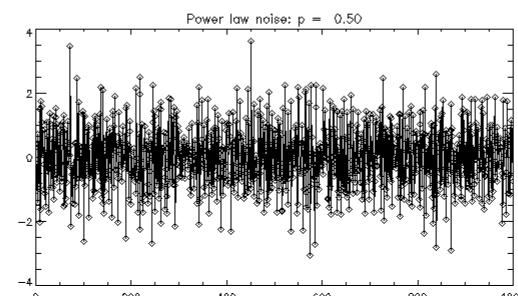
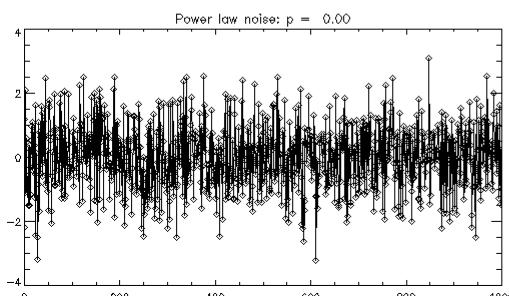
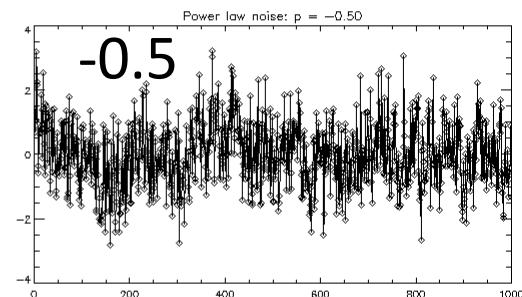
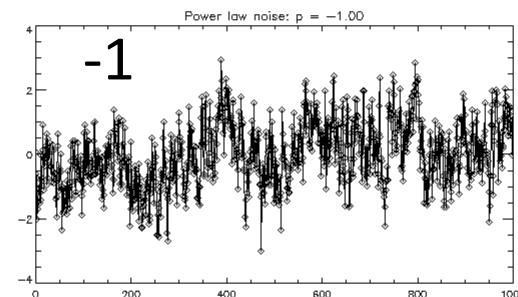
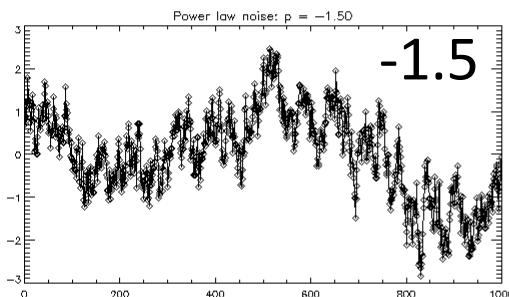
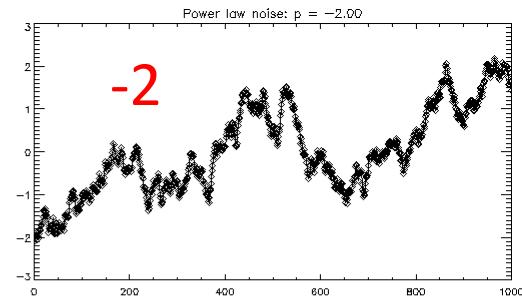
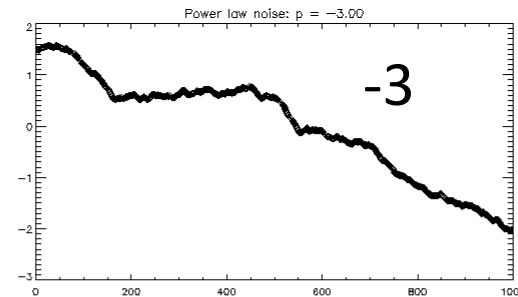
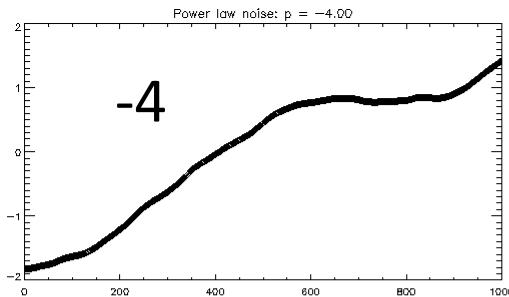
One form of noise, known as “flicker noise” or “pink noise”, has energy per unit frequency range proportional to f^{-1} , so it can be generated from an FT of Gaussian random (white) noise multiplied by $f^{-0.5}$.

Power laws steeper than f^{-2} become ever more dominated by the longest time scales. For example, f^{-4} noise (which can be generated by integrating white noise twice, as well as by the more general method above) produces very smooth-looking curves which are usually dominated by the trend.

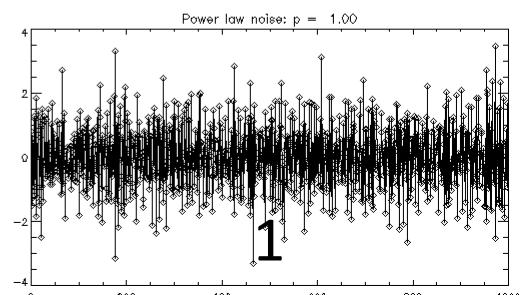
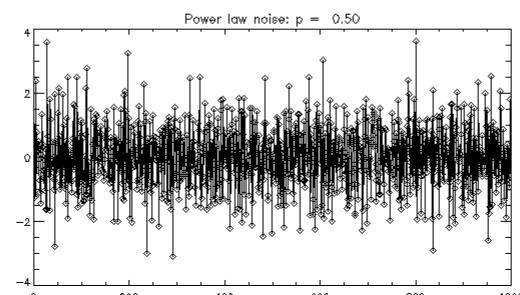
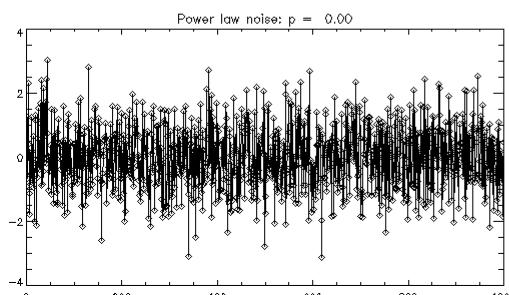
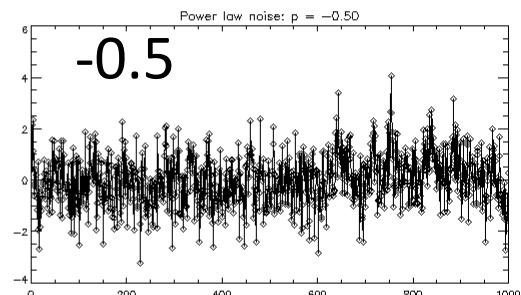
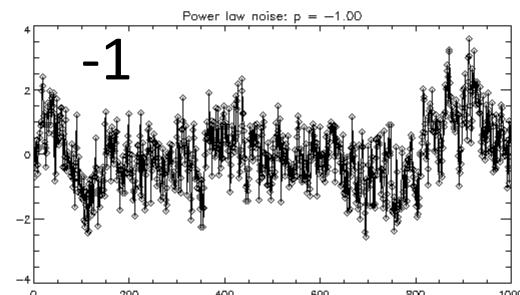
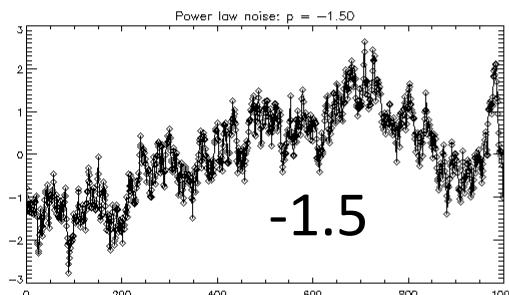
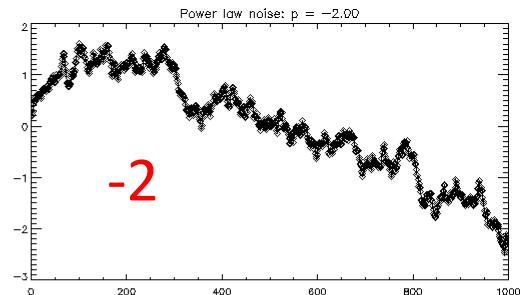
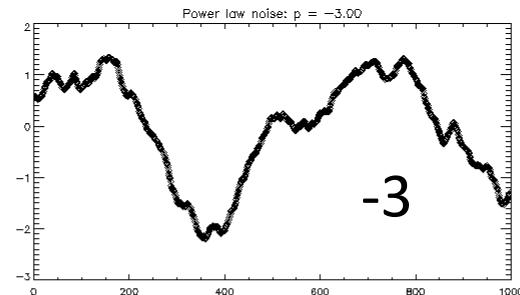
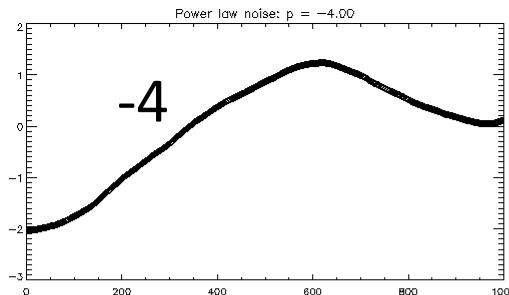
The Fourier method of generating power law noise has one problem: the time series generated is always periodic, so the last and first points must be close together (unlike real red noise). A way round this is to generate a longer time series and just use the first half.

Different power laws produce time series with quite distinctive different “looks”.

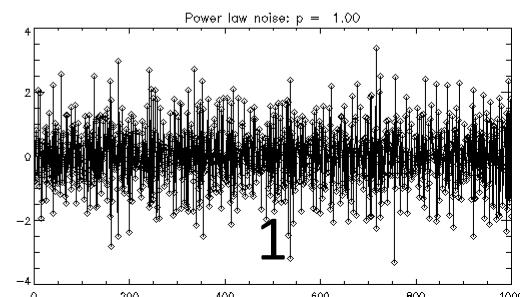
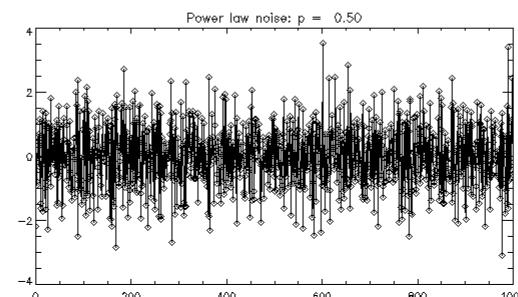
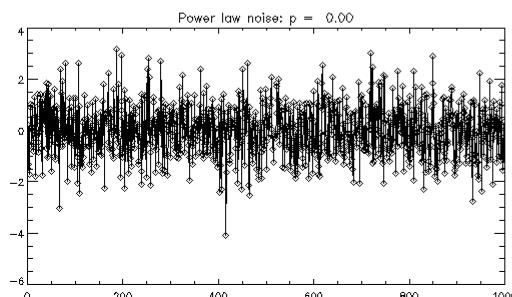
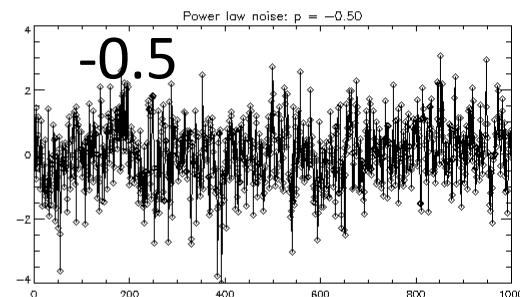
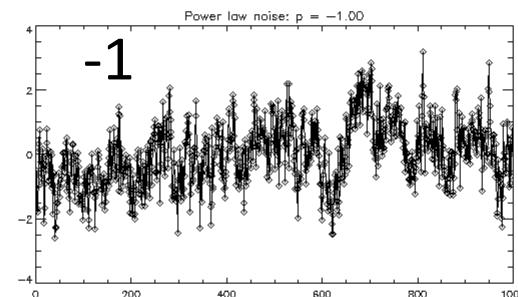
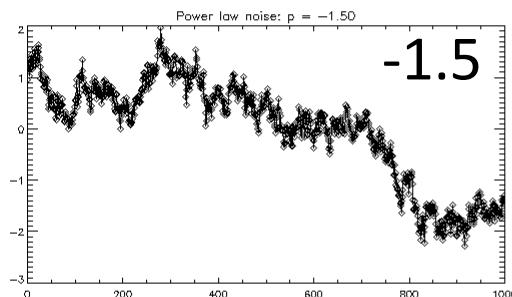
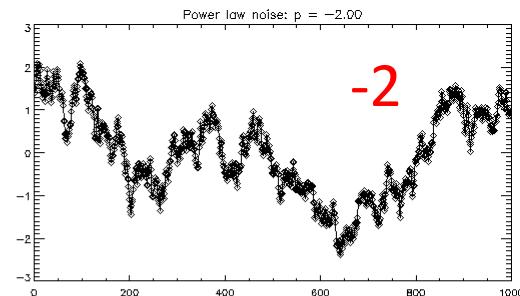
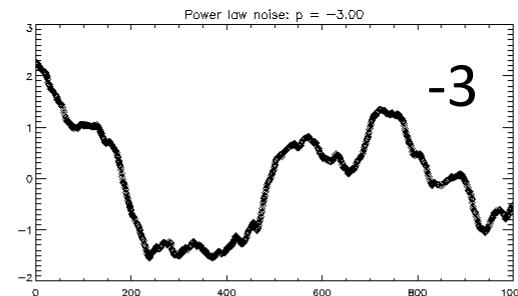
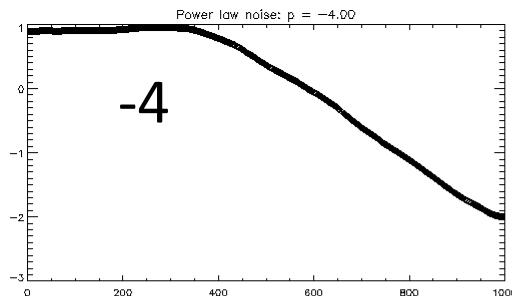
Examples of time series generated from random power law noise



Examples of time series generated from random power law noise



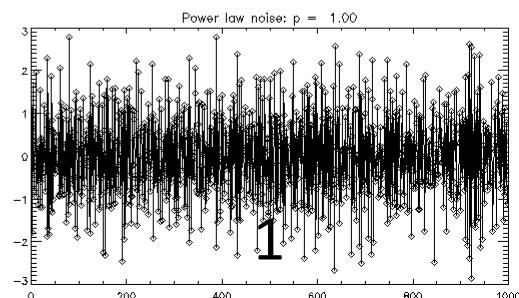
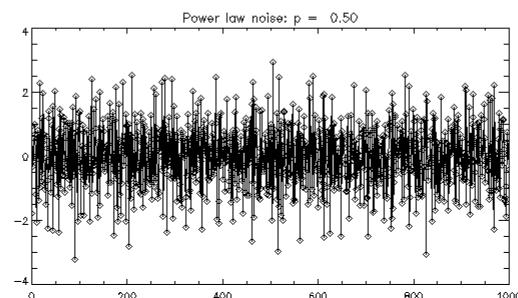
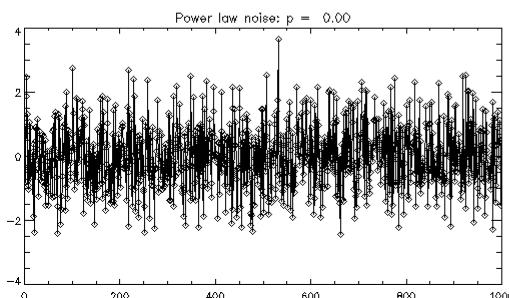
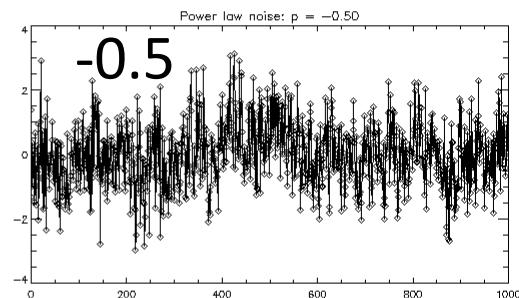
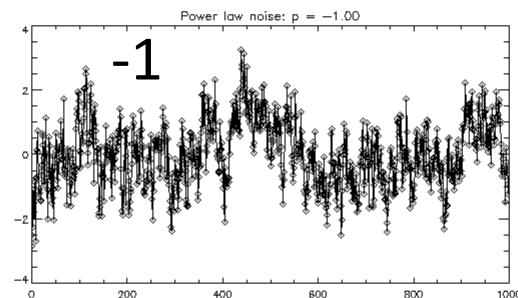
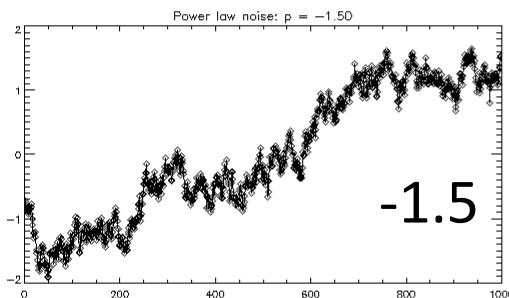
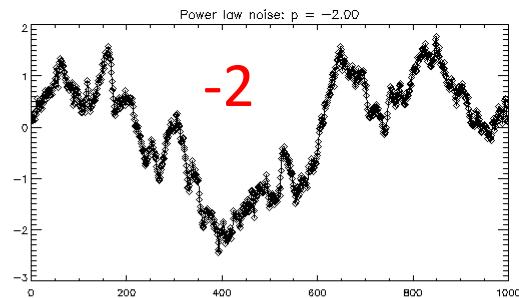
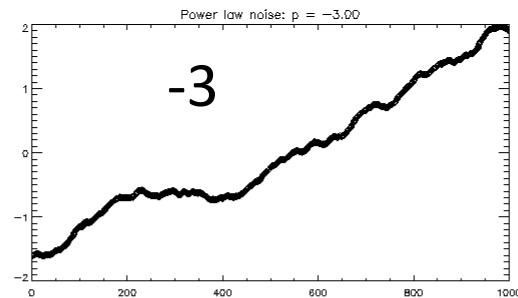
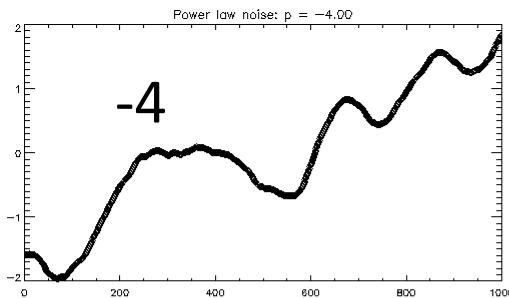
Examples of time series generated from random power law noise



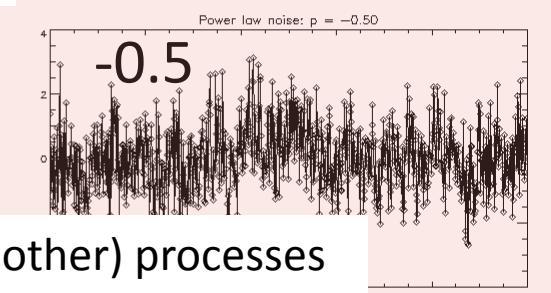
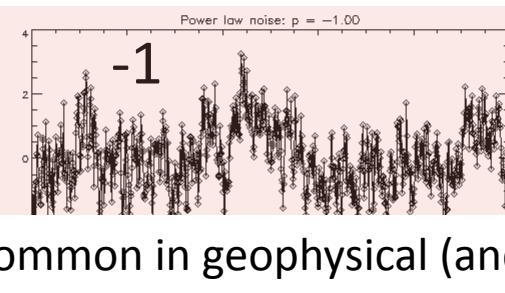
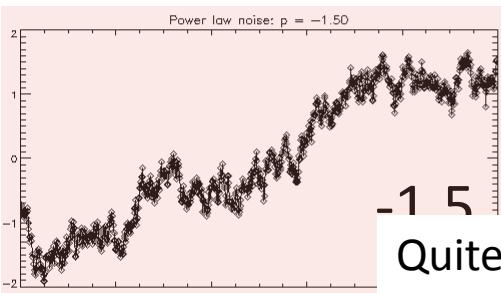
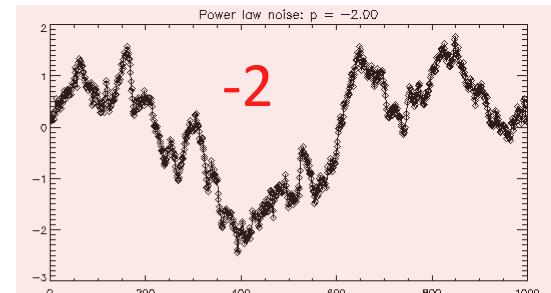
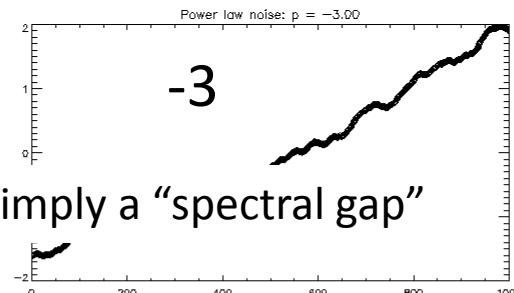
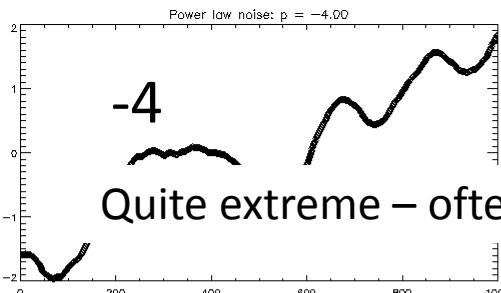
0

0.5

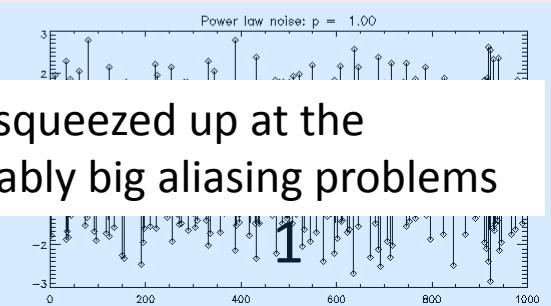
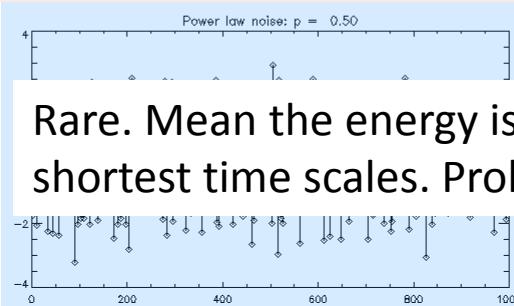
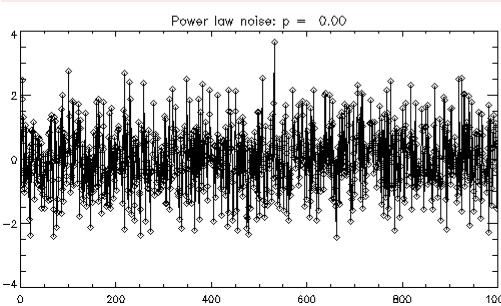
Examples of time series generated from random power law noise



Examples of time series generated from random power law noise



Quite common in geophysical (and other) processes



Rare. Mean the energy is squeezed up at the shortest time scales. Probably big aliasing problems

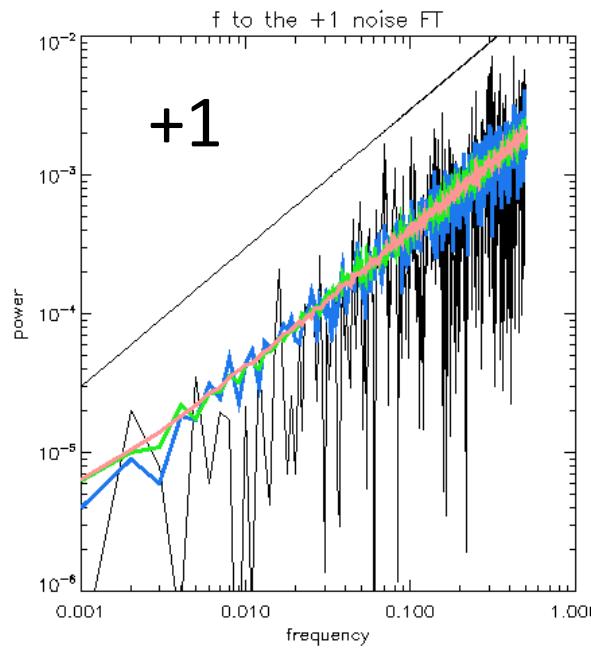
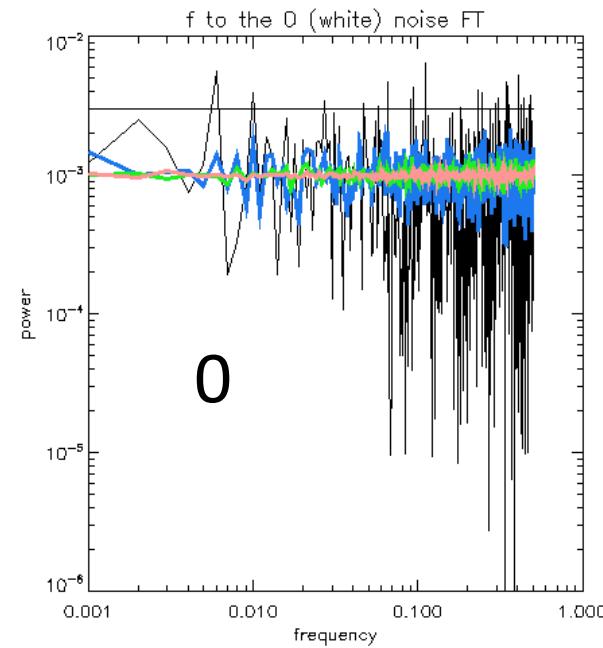
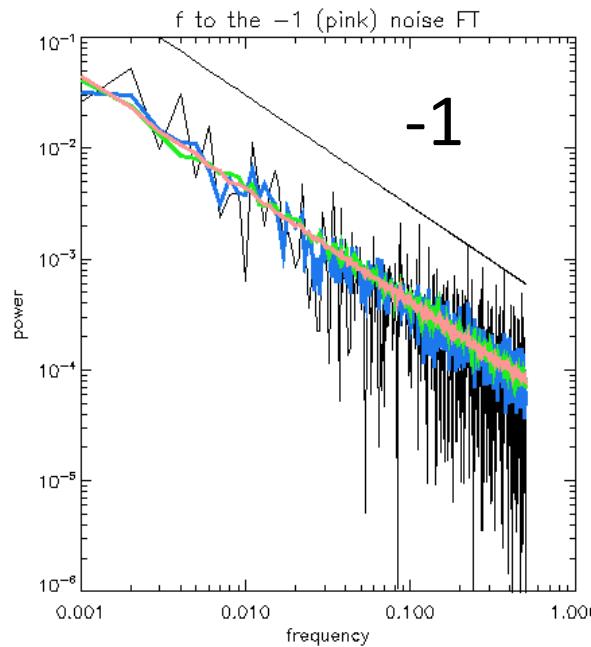
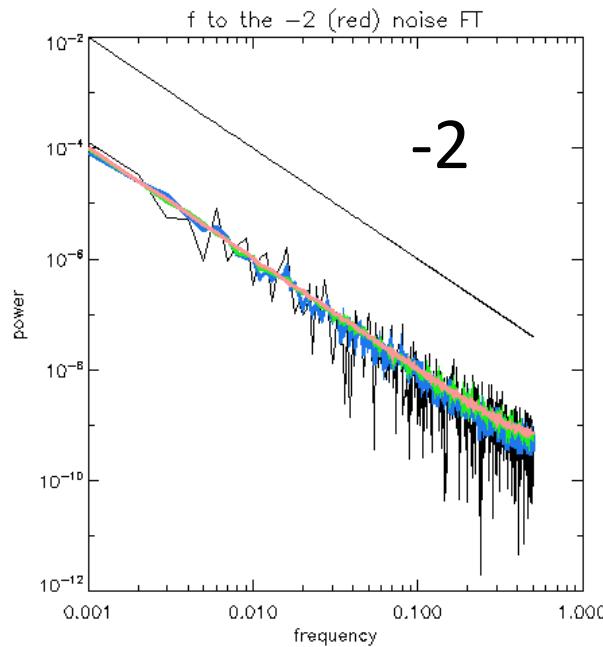
The idea of a “spectrum” of variability: apportioning the energy by frequency

The Fourier transform of a time series does precisely this. Each pair (a for cosine, b for sine) of coefficients tells you the variance at that frequency (a^2+b^2).

However, the real world is not usually limited to just a finite length of time series. Usually you have a measured section of data from an ongoing longer time series, or one particular realization of a time series which can be repeated many times (as an experiment is repeated, say).

If you believe there is some underlying process leading to a particular distribution of energy as a function of frequency, then that distribution is the spectrum, and your time series only allows you to calculate an approximation of that spectrum. The Fourier transform amplitude squared (or periodogram) gives a very noisy estimate of that spectrum, because you are estimating one number for every two numbers in the time series. This gives very little scope for statistical averaging.

One way round this is to have multiple time series (or split your time series up into sections), and then average the periodograms from each section...



Generate a time series of length 1 million.

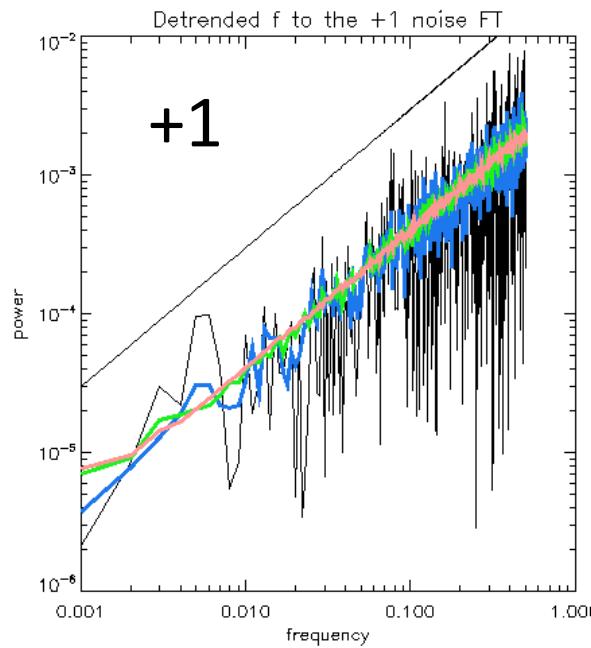
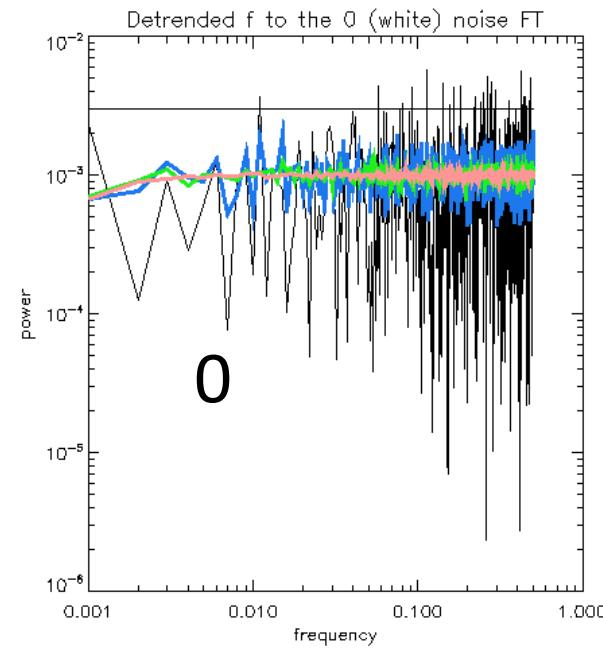
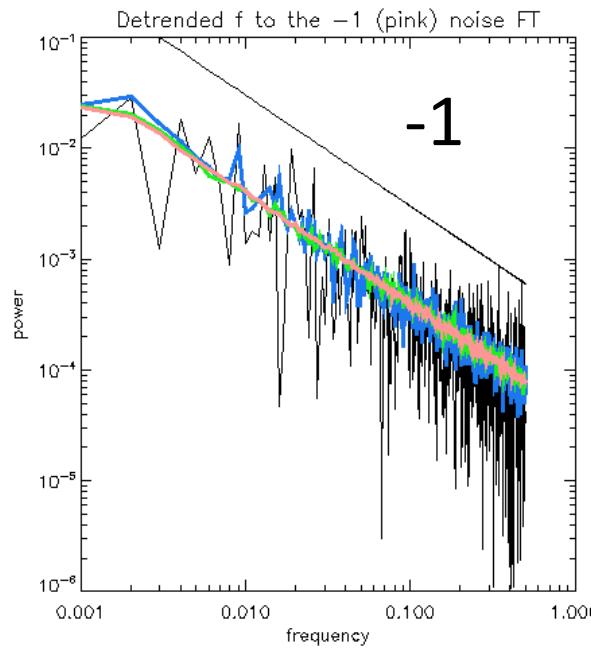
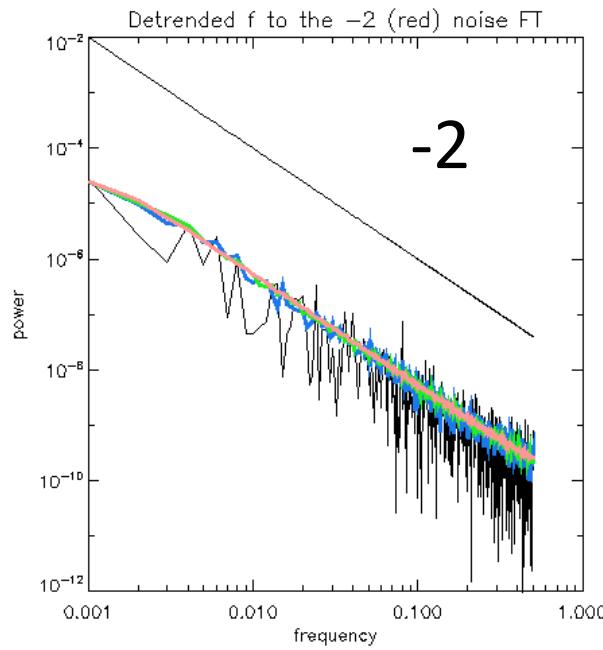
Perform FT on 1000 sections each of length 1000.

Sum squares of sine and cosine coefficients at each frequency

Average over n FTs

Single realization
[Average of 10](#)
[Average of 100](#)
[Average of 1000](#)

Averaging random noise – scatter reduces by factor $1/\sqrt{n}$



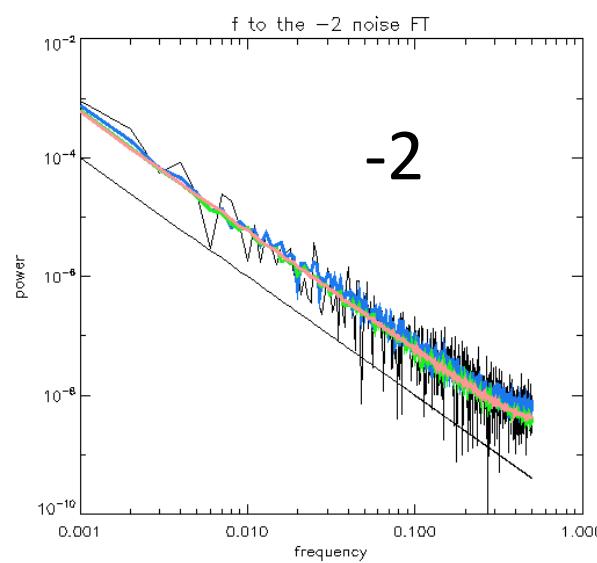
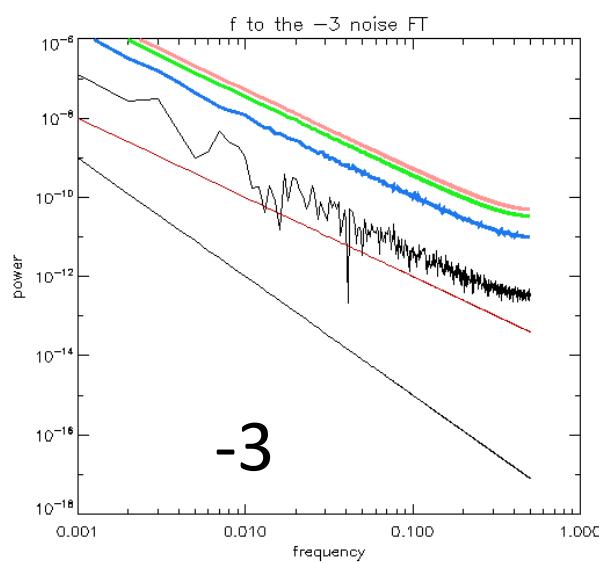
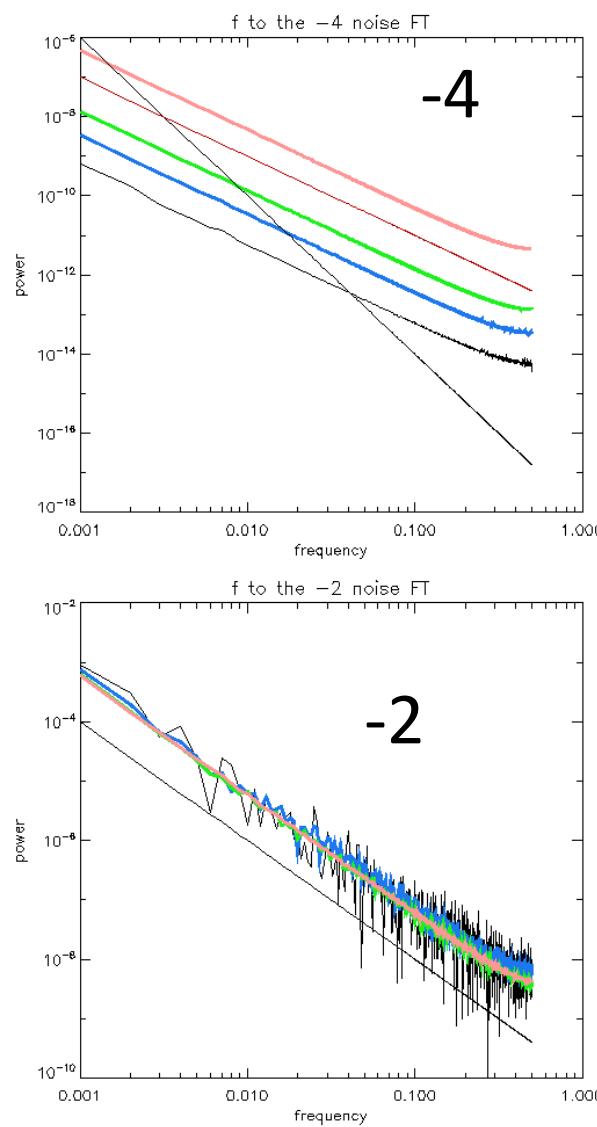
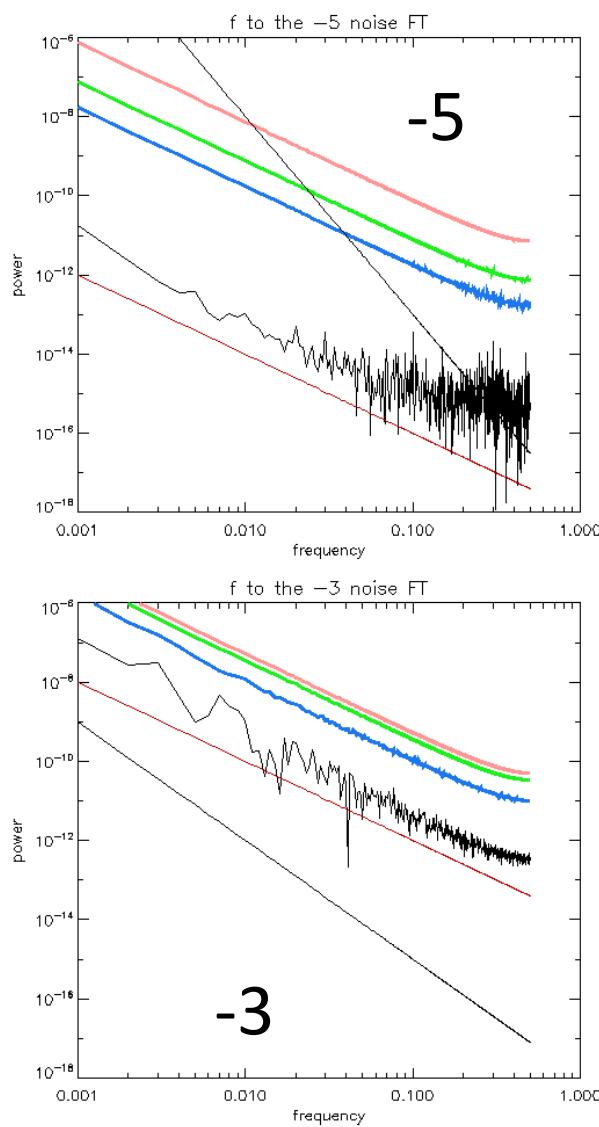
Subtract trend from each segment before calculating FTs

Note slight drop in power at lowest frequency (0 and -1)

Note for -2, drop in power at all frequencies.

Sawtooth wave (trend) produces a spectrum with power proportional to f^{-2} .

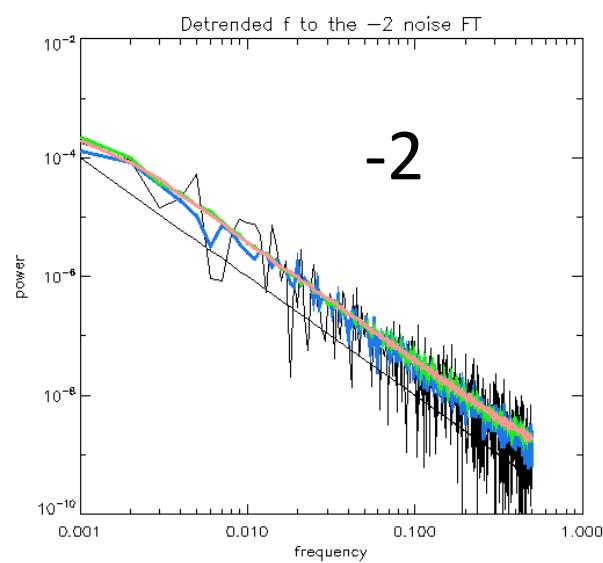
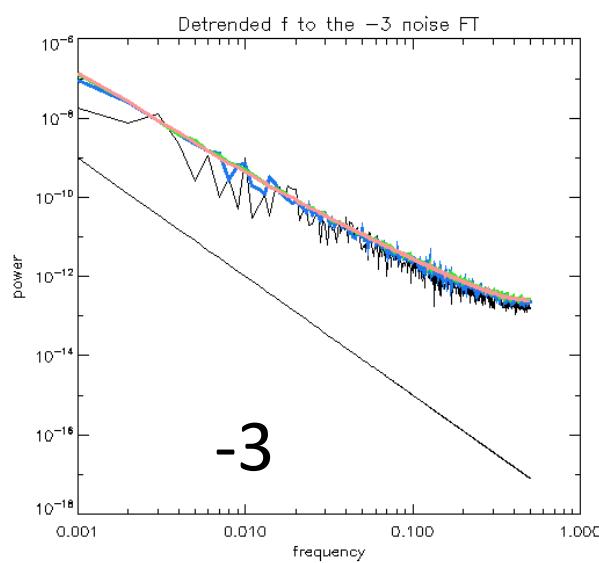
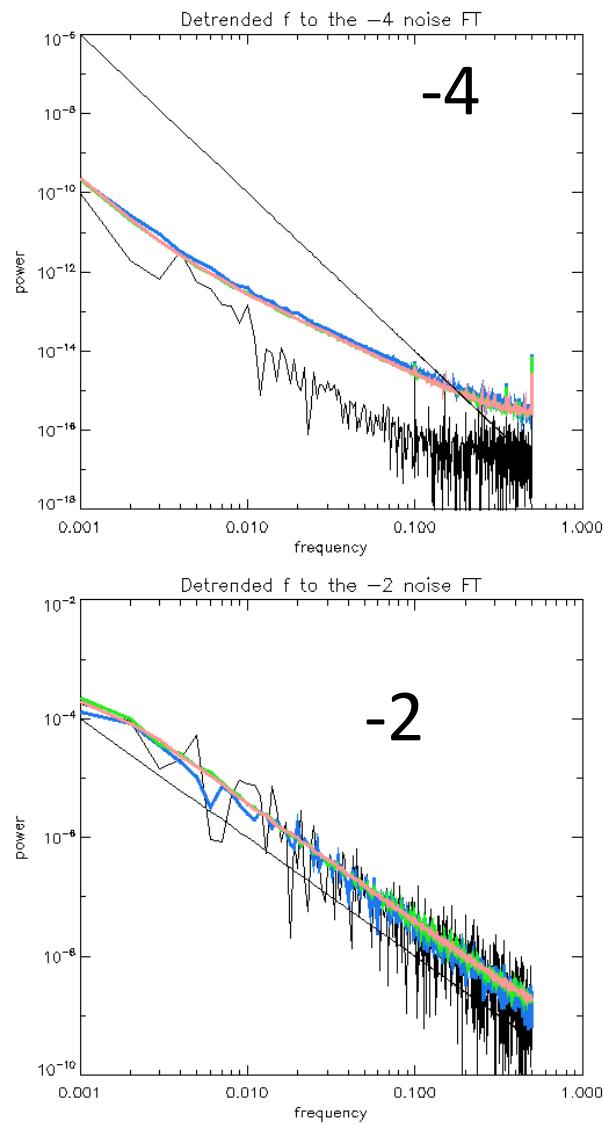
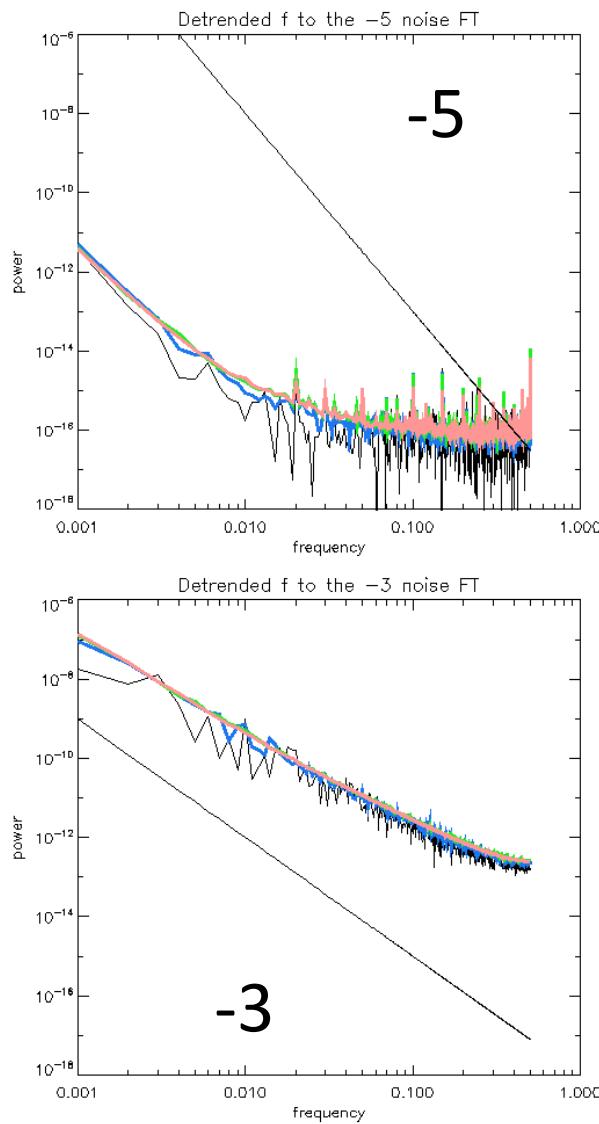
Single realization
[Average of 10](#)
[Average of 100](#)
[Average of 1000](#)



This means that steeper power laws behave badly when estimating a spectrum. Because the trend is an important part of any segment, the associated f^{-2} behaviour dominates.

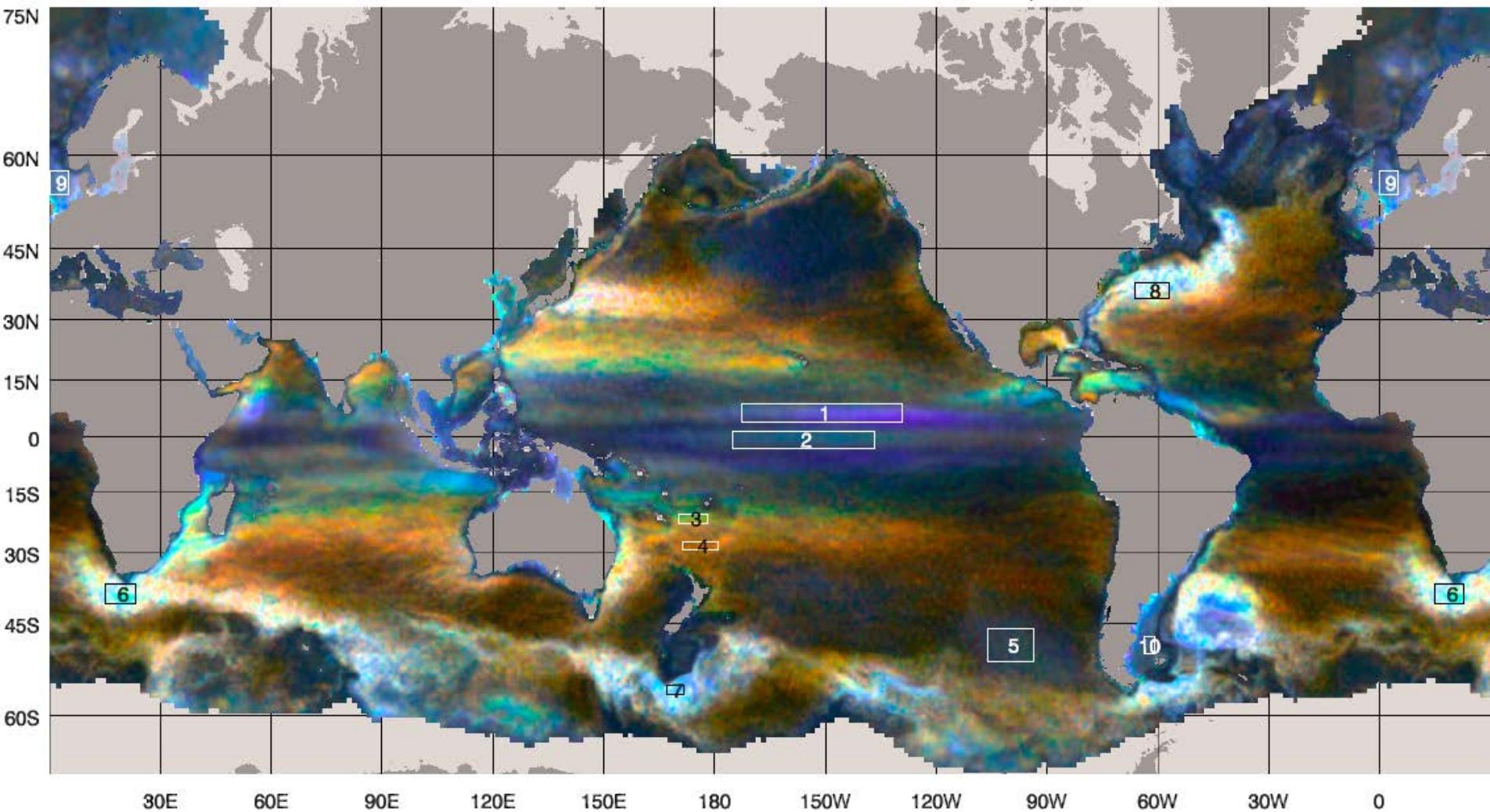
The black line here is the input power law as before. The red line represents f^{-2} .

In a sense, such steep power law processes do not have a spectrum.



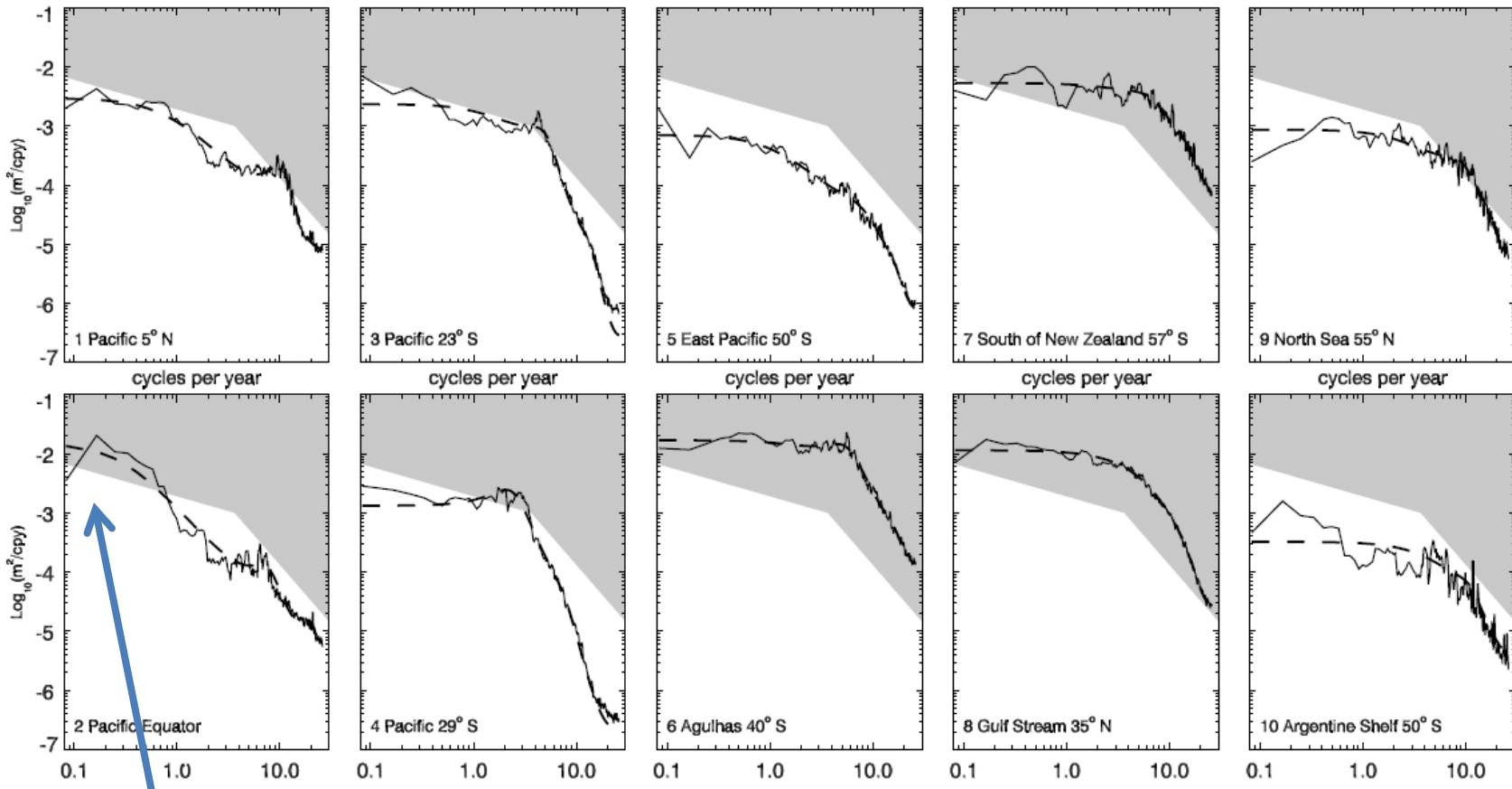
As you might expect, detrending the segments before calculating the FFT makes a big difference in this case.

It doesn't allow you to reproduce the power law, but it does reduce the f^{-2} noise which may well be swamping other signals at higher frequencies.



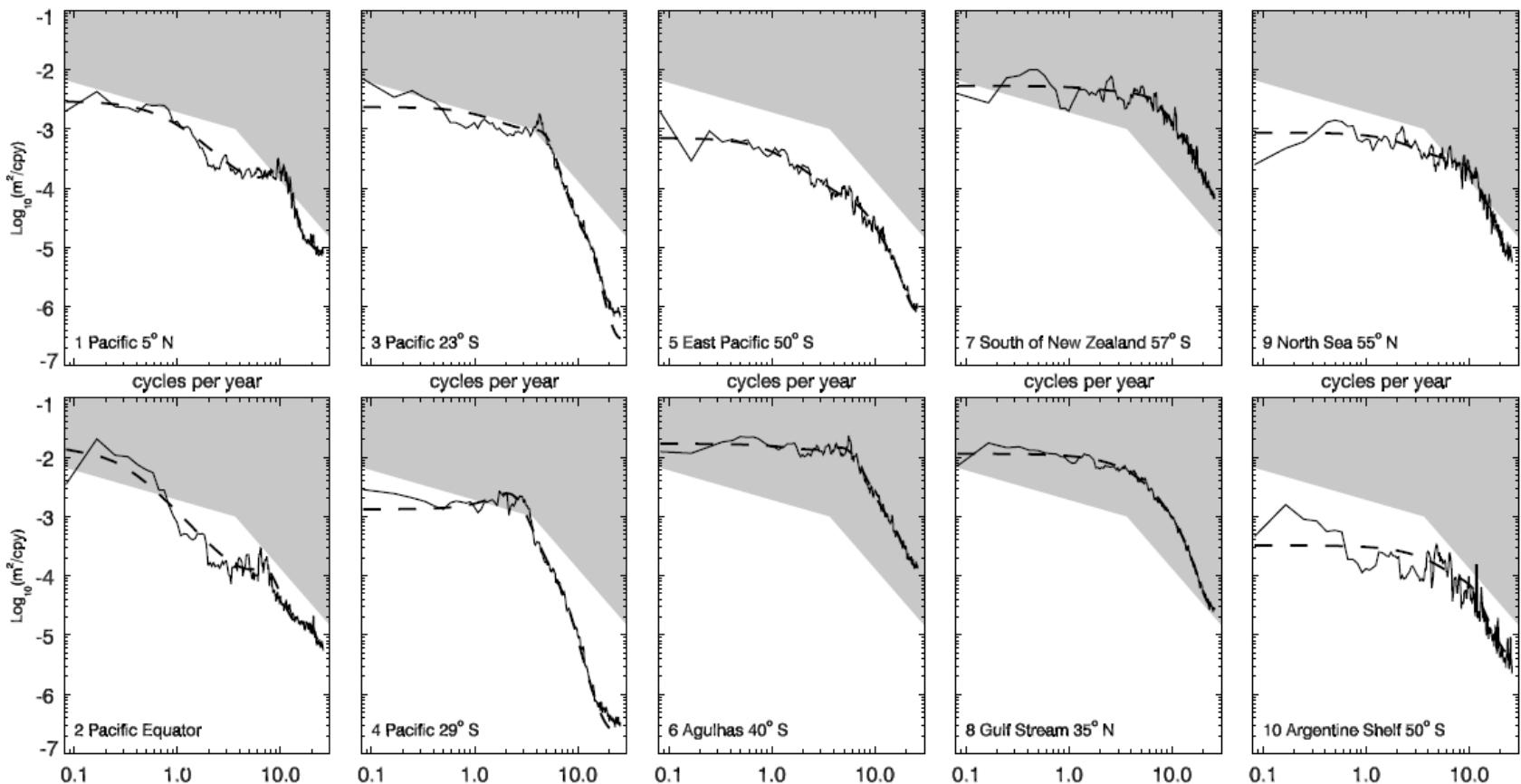
Hughes, C. W., and S. D. P. Williams (2010), The color of sea level: Importance of spatial variations in spectral shape for assessing the significance of trends, *J. Geophys. Res.*, 115, C10048, doi:10.1029/2010JC006102

Shading shows $f^{-0.5}$ and f^{-2} behaviour. Some slopes are around f^{-4} , but not extending to the longest period (so spectral estimation is not a problem).



Note the possible steep slope at long periods here. May mean spectrum is poorly resolved (Pacific equator is dominated by ENSO, with long periods dominant).

In many cases the steeply sloping region starts at a well-defined frequency (close to the shortest period at which baroclinic Rossby waves can exist). The energy drops rapidly at shorter periods because the waves are absent.



Issues in spectral estimation

- Need to average multiple realizations to reduce noise (otherwise it is almost as big as any signal). Can do this by cutting up your time series and averaging the FT powers (there are other ways too).
- Need to minimize “edge effects” from the individual sections not being periodic. Can do this by multiplying by a “window” which reduces to zero (or near zero) at the edges.
- Using window loses data – can improve the calculation by overlapping windows.
- Using window reduces power in the time series. Need to correct for this.
- All requires great care to correctly account for these complications and provide a meaningful estimate of the likely noise.
- Matlab provides a number of spectrum estimating routines designed for different purposes. The one which will be generally useful to us is `pwelch`.

```
dt=7  
lenx=size(x,1)  
  
[pxx,f,pxxc]=pwelch(x,1/dt,'ConfidenceLevel',0.95)
```

Calculates the power spectrum pxx, and corresponding frequencies f, for a time series x with timestep dt. Also returns an upper and lower bound (in pxxc) within which there is a 95% chance that the “true” spectrum lies. Choose different confidence intervals by changing the 0.95

The frequencies it calculates the power spectrum at may be unexpected. They result from a complicated internal calculation. It uses a “hamming window” ($1+\cos$), with 50% overlap, and averaging over 8 windowed segments.

```
[pxx,f,pxxf]=pwelch(x,lenwin,1/dt,'ConfidenceLevel',0.95)
```

does the same but using window lengths lenwin, allowing you to change the number of segments (though the resulting number depends on the window overlap and the data length)

If you want more control over the output list of frequencies, try this:

```
function [pxx,f,pxxc,nseg]=cwhspec(x,dt,winwid,nperfreq,pconf,fracoverlap)
%
% calculates spectrum of time series x, given time step dt
% using hamming windows of width winwid (measured in time units)
% overlapped by the fraction fracoverlap (typical value 0.5).
%
% nperfreq specifies the number of estimates to give for each independent
% frequency.
%
% pconf specifies the percentage confidence interval to supply
%
% returns:
%
% pxx = power spectral density
% f = corresponding frequencies (cycle per unit time)
% pxxc = upper and lower limits on pxx, at specified confidence
% nseg = number of windowed segments used
%
lennx=size(x,1);
lenseg=floor(winwid/dt);
noverlap=floor(fracoverlap.*lenseg);
nseg=floor((lennx-noverlap)/(lenseg-noverlap));
ww=hamming(lenseg);
[pxx,f,pxxc]=pwelch(x,ww,noverlap,lenseg.*nperfreq,1./dt,'ConfidenceLevel',pconf./100);
end
```

What is this “nperfreq” business?

Matlab, by default, doesn’t simply perform an FFT of the individual segment time series: It pads it with zeros first, either to 256, or to the next power of 2 bigger than the length of the segment.

This has the effect of producing a smooth spectrum

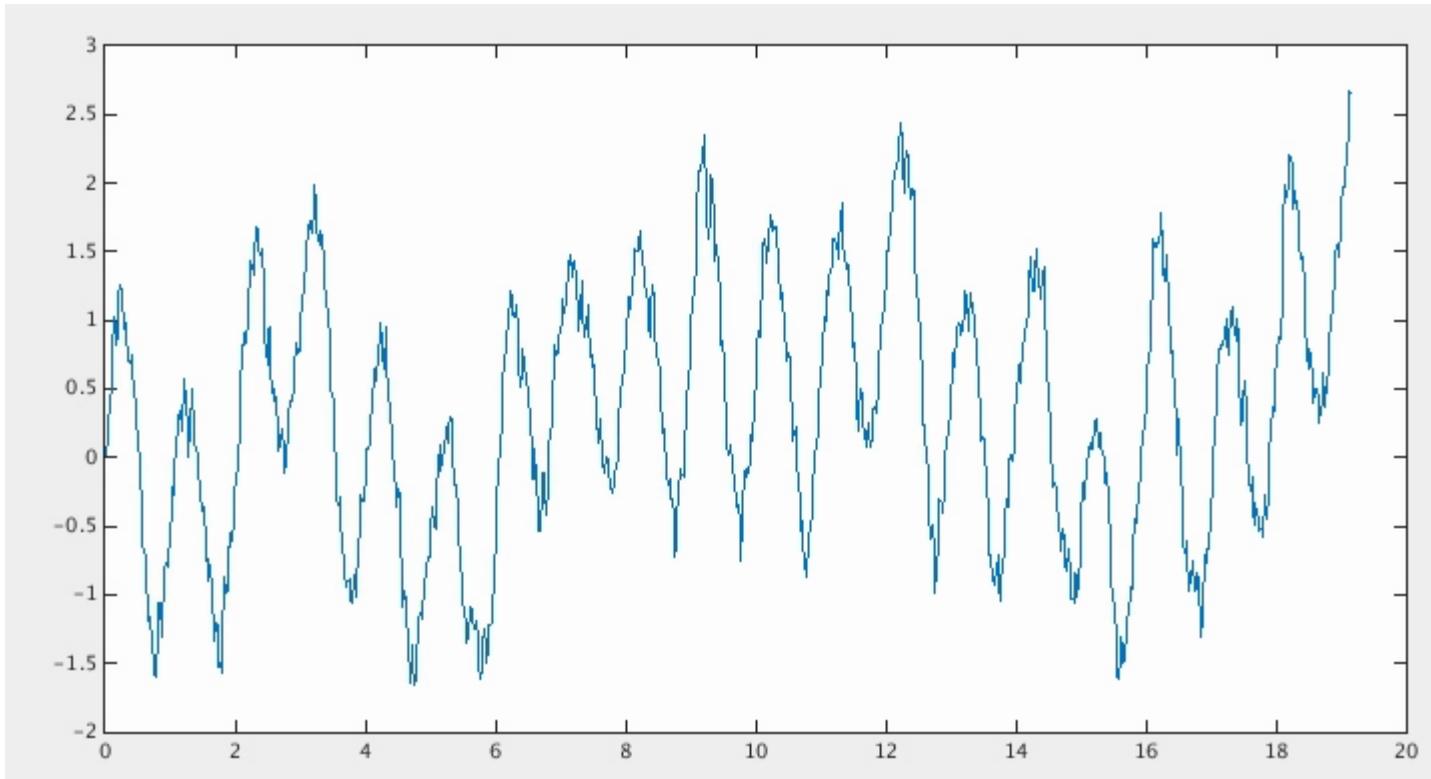
In the above code, nperfreq=1 forces the FFT to match the segment length with no padding, so each frequency estimate is independent.

It can be useful to smooth, so as to see peaks which do not lie exactly on the exactly sampled frequencies. Using e.g. nperfreq=10 will give 10 points per independent frequency, and produce a smoother plot.

Doing it this way means the level of smoothing and oversampling is under your control

An example time series of 1000 points measured 1 every 7 days (just over 19 years).

Red noise, plus an annual cycle of amplitude 1,
and a 10 per year cycle of amplitude 0.1.

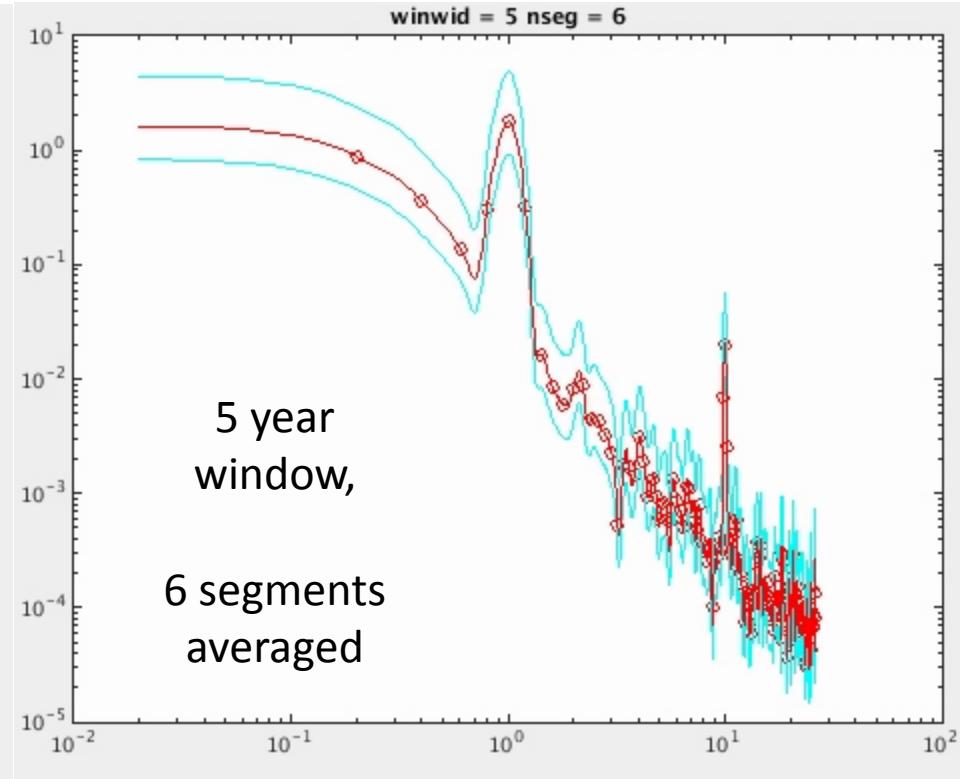
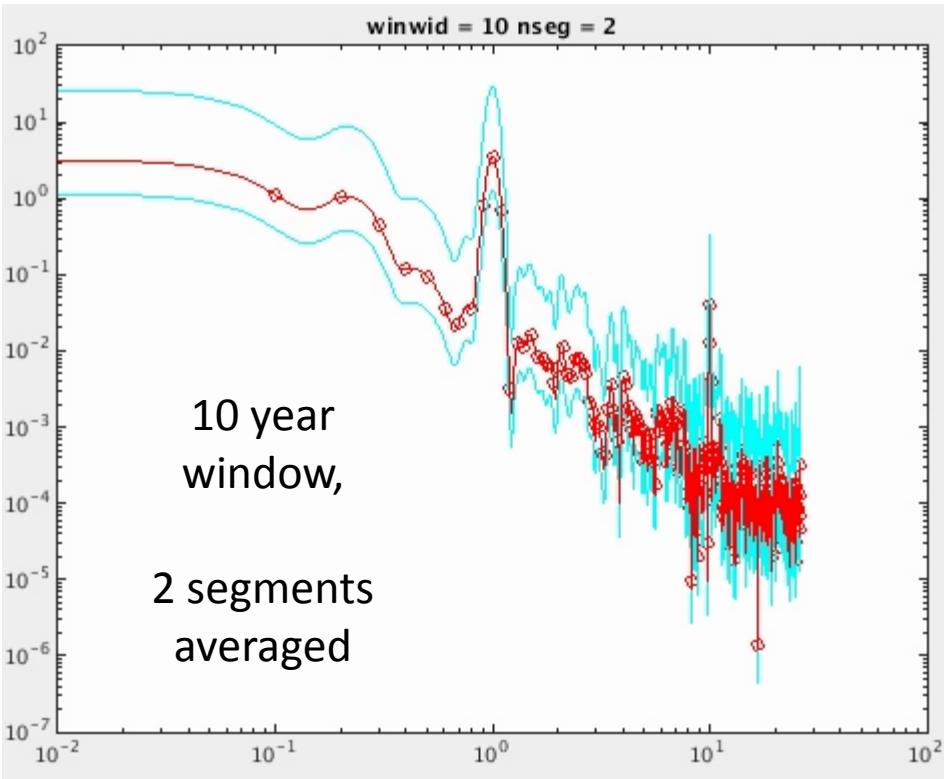


Calculate spectrum with a variety of window widths. Illustrate the trade off between resolution in frequency and reduction of noise.

Diamonds: nperfreq=1, shows number of independent points

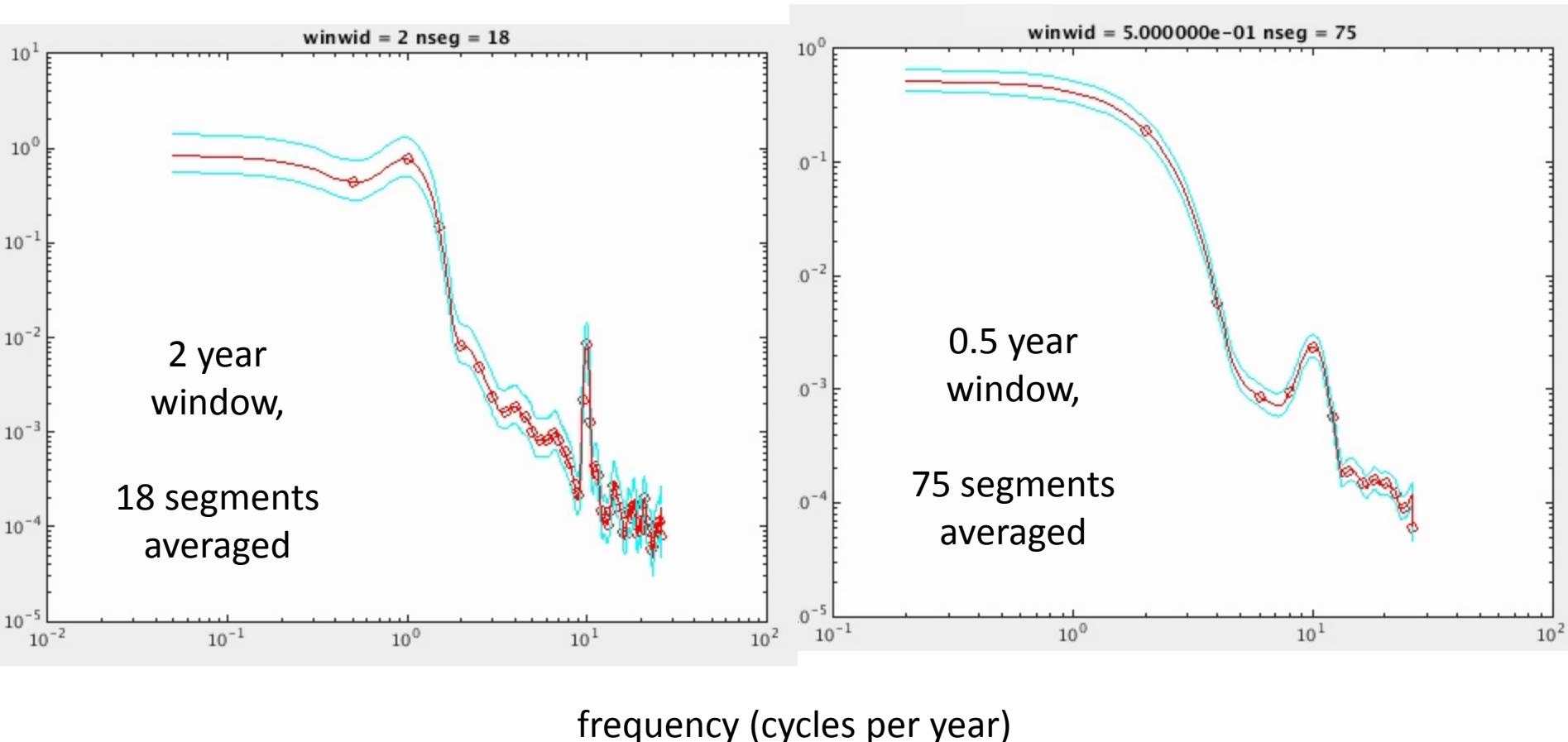
Line: nperfreq=10

Cyan: shows 95% noise range



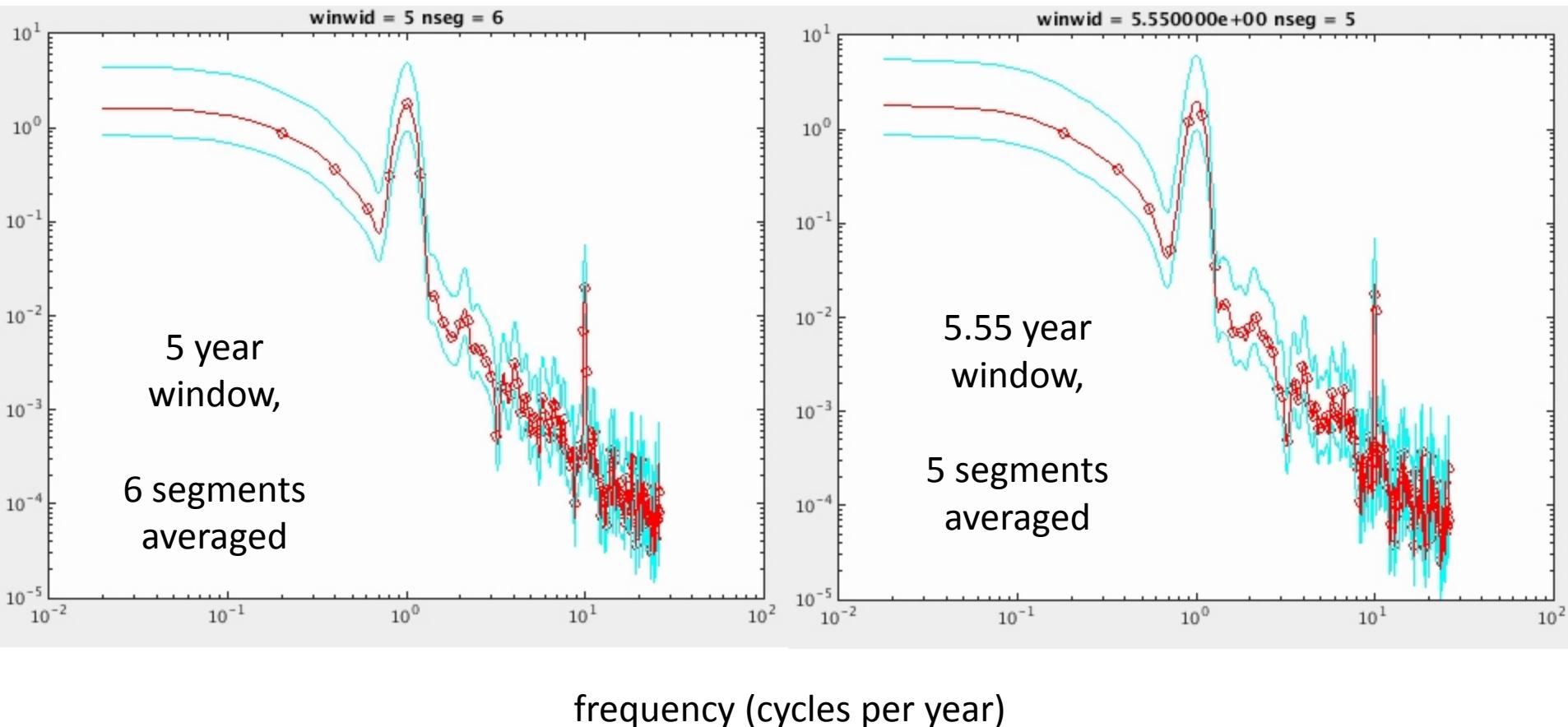
frequency (cycles per year)

Diamonds: nperfreq=1, shows number of independent points
Line: nperfreq=10
Cyan: shows 95% noise range



Note how noise limits are at a constant distance from the spectrum – because the error bounds are a constant multiple of the spectrum (and the plot is logarithmic).

An advantage of oversampling the frequencies: When there is a frequency which is not an exact fraction of the window width, the peak may lie between sampled frequencies.



Units and normalization

These are plots of “Power Spectral Density”. i.e. the power (variance) per unit distance along the frequency axis.

If my time series was sea level in m, and the time was in years, then the units would be $\text{m}^2 \text{ (cycle per year)}^{-1}$.

The integral of the PSD (i.e. the sum of the values multiplied by the frequency interval $df = f(2)-f(1)$) should equal the variance of the time series. This would be exactly true of a periodogram normalized in the same way. Because of the various complications with the sampling and averaging, this is not exactly true of the spectrum – the integral of the spectrum is an estimate of the variance of the underlying (infinitely long) time series. Recall, red noise and steeper have a variance which grows continuously with length of the time series, so the spectral estimate will be poor in these cases.

Where is the power? “variance preserving” spectra.

Log-log plots can be good for spectra because of the power law scaling which is often observed (power laws appear as straight lines on such plots).

However, they make it difficult to see where the energy is concentrated. The area under a particular section of the spectrum is proportional to the energy in that frequency band, but only if plotted on a linear-linear scale.

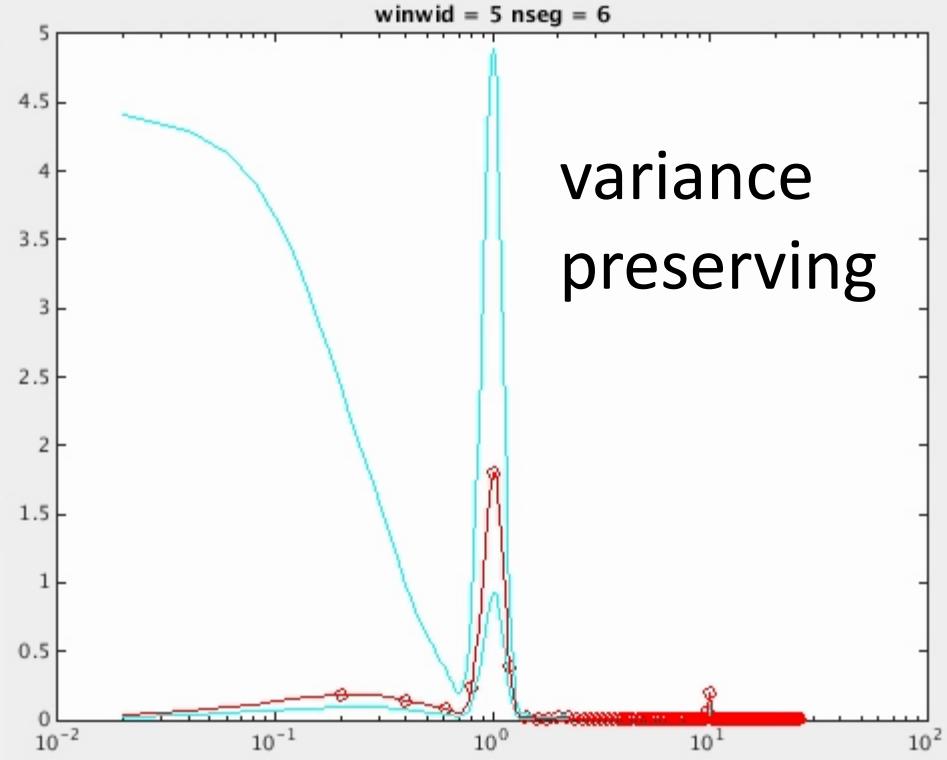
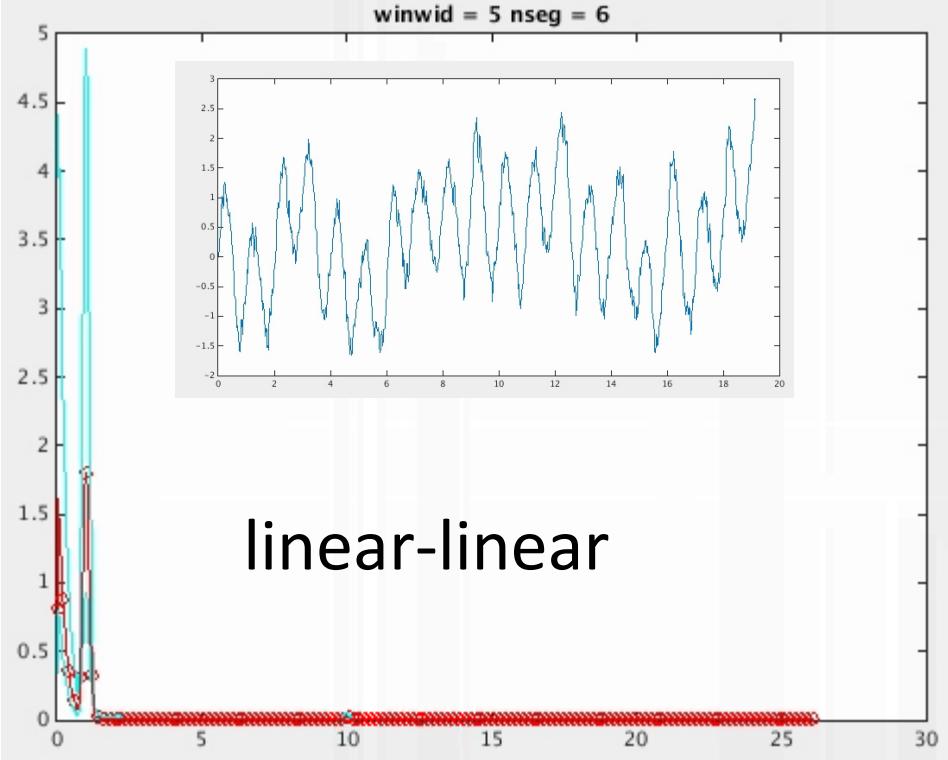
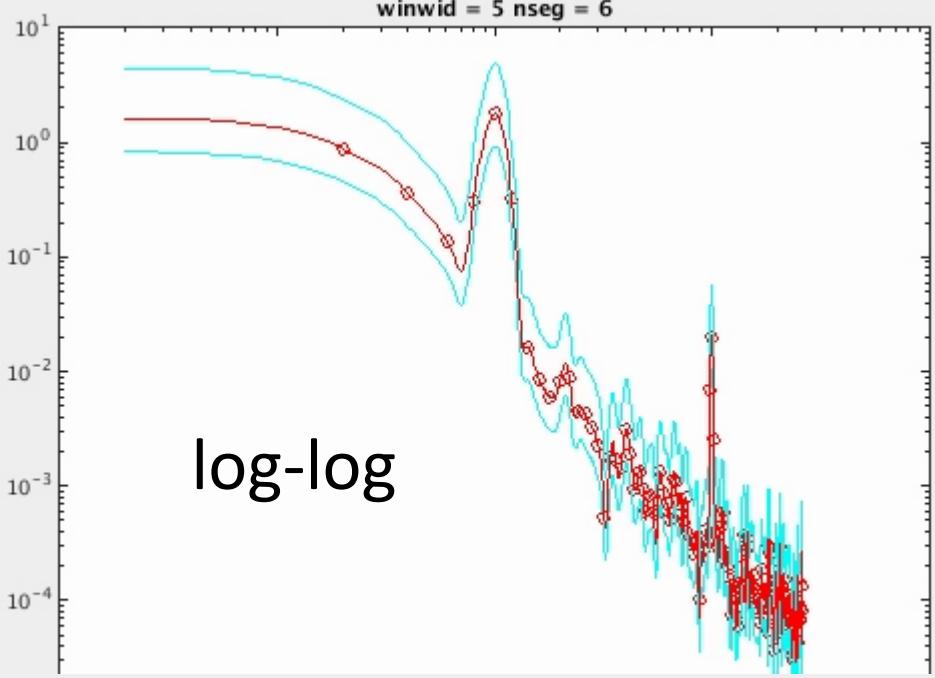
Such scales are rarely much use. All the information tends to be squeezed up at one end, and smaller signals get lost completely.

A compromise is the variance preserving spectrum, which is linear in the y (power) axis, but logarithmic on the x (frequency) axis. To compensate for the logarithmic “squeezing” of frequency, power is multiplied by f (because $d(\log(f))/df = 1/f$). This means the contribution to variance from a particular frequency range is again proportional to the area under the curve.

The y axis would then have units m^2 per unit $\log_e f$ (natural log rather than \log_{10} , though that could be fixed by dividing power by $\log_e 10 = 2.3026\dots$)

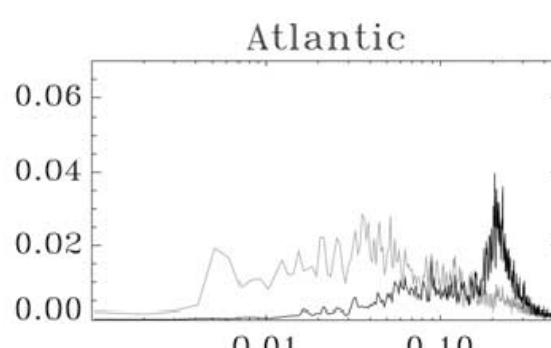
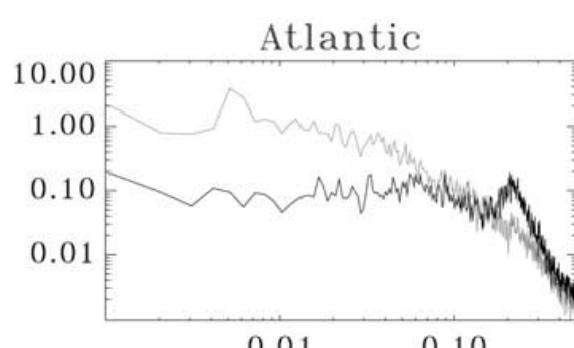
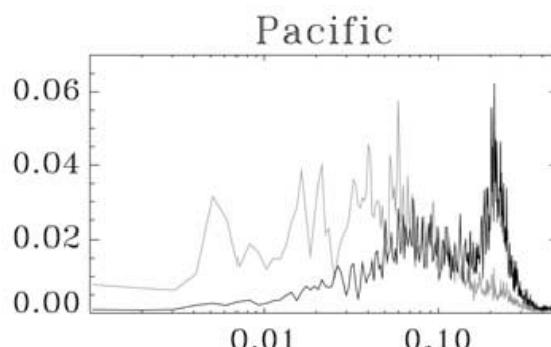
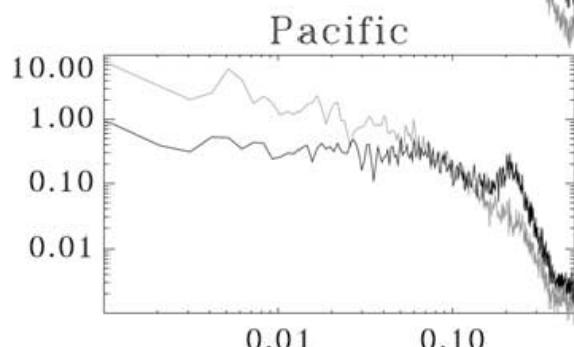
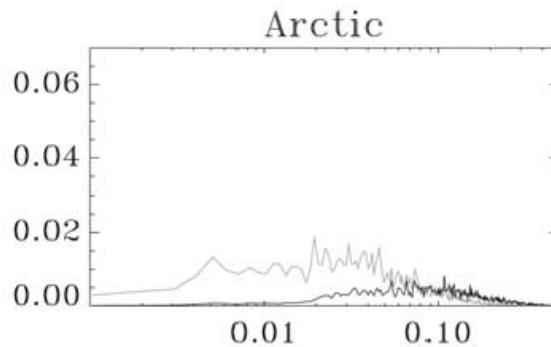
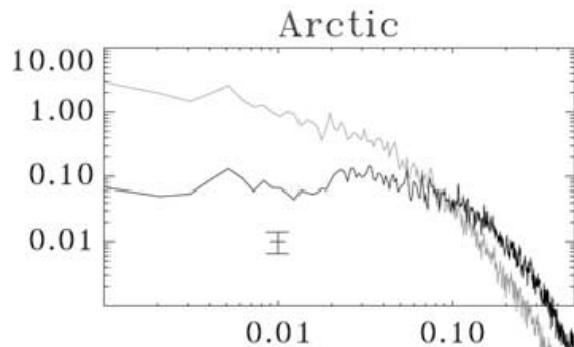
The same spectrum plotted 3 different ways.

Annual amplitude was 1, 10 per year
amplitude 0.1, so the power is 100
times smaller



log-log

variance preserving



cycles per day

spectra of mass of ocean (pale) and of ocean plus atmosphere (dark) integrated over 3 ocean basins.

The dominance of energy at periods shorter than about 50 days (and especially near 5 days) is much clearer in variance preserving form.

Frequency vs Period

- It can be a nuisance having to translate between frequency and period – period is the more natural quantity to think in, so why not plot spectra as a function of period?
- There's no harm (and it can be useful) in marking particular periods, or adding a second x-axis to label the periods. This can be a good idea.
- However, using period for the axis makes an important difference. You are implicitly saying that the power is per unit period instead of per unit frequency. The same way you had to multiply by f to make it power per unit (log frequency), you have to multiply by f^2 to make it power per unit period.
- This can be done but you will confuse everyone if you do! You will be plotting the Spectral Power Distribution (often used for spectra of light) rather than the Power Spectral Density.
- When you see spectral plots labelled with period, in oceanography papers (they do appear), they are almost always PSD with the wrong labels on the axes, not SPD.
- In short – adding period information to your PSD plot can be useful, but leave the main x-axis in frequency. Anything else can produce confusion.