Wavelet Analysis

In this demo we will learn to apply wavelet analysis. This is a great way to visualize the variability in your time series.

We'll be working with data from the m1244 mooring in the Labrador Sea, which you can get in NetCDF form here. (This is included in the full distribution of the course materials.)

This notebook requires my jlab toolbox to be installed. You will also need to have set up Jupyter Lab to work with Matlab, following these instructions.

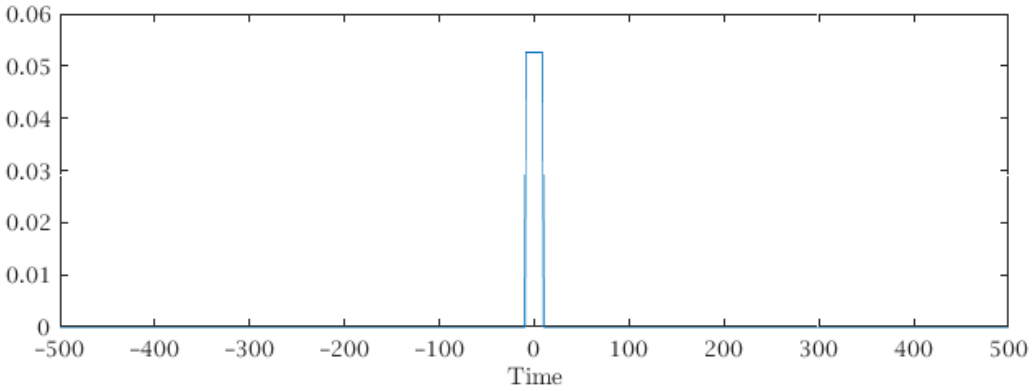
Simple Smoothing Revisited

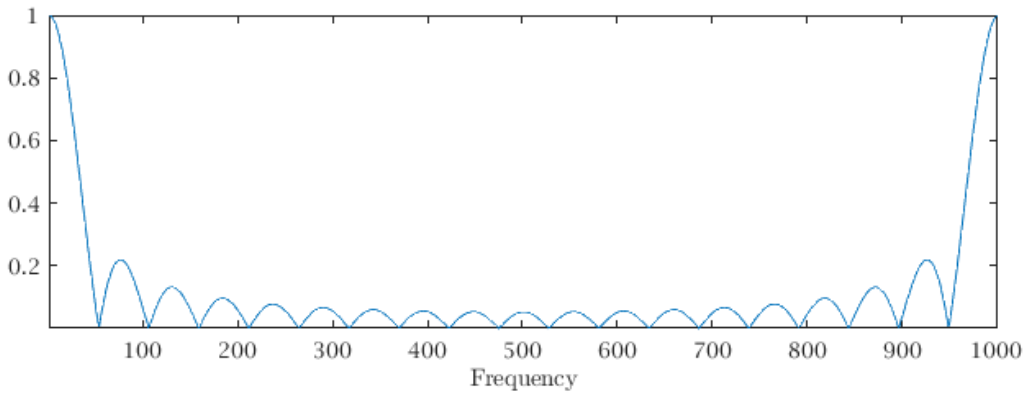
First we will review simple smoothing of our data.

Generally, the goal of smoothing is to remove high frequencies. This is why we often use the term lowpass, because we want to design a filter through which will let the low frequencies pass through, but which will reject the high ones.

Let's see how the boxcar or rectangle function performs as a lowpass filter. Whereas, as we have seen, the Fourier transform of a Gaussian is also a Gaussian, the Fourier transform of a boxcar is a very bouncy function.

In [1]:
```
set(groot,'defaultfigurepaperposition',[1 1 7 5]) %set default figure size

t=[1:1001]';t=t-mean(t);
g=zeros(size(t));
g(abs(t)<10)=1/19;%this is a 19-point boxcar filter

subplot(2,1,1),plot(t,g),xlabel('Time')
subplot(2,1,2),plot(abs(fft(g))),xlabel('Frequency'),axis tight
```



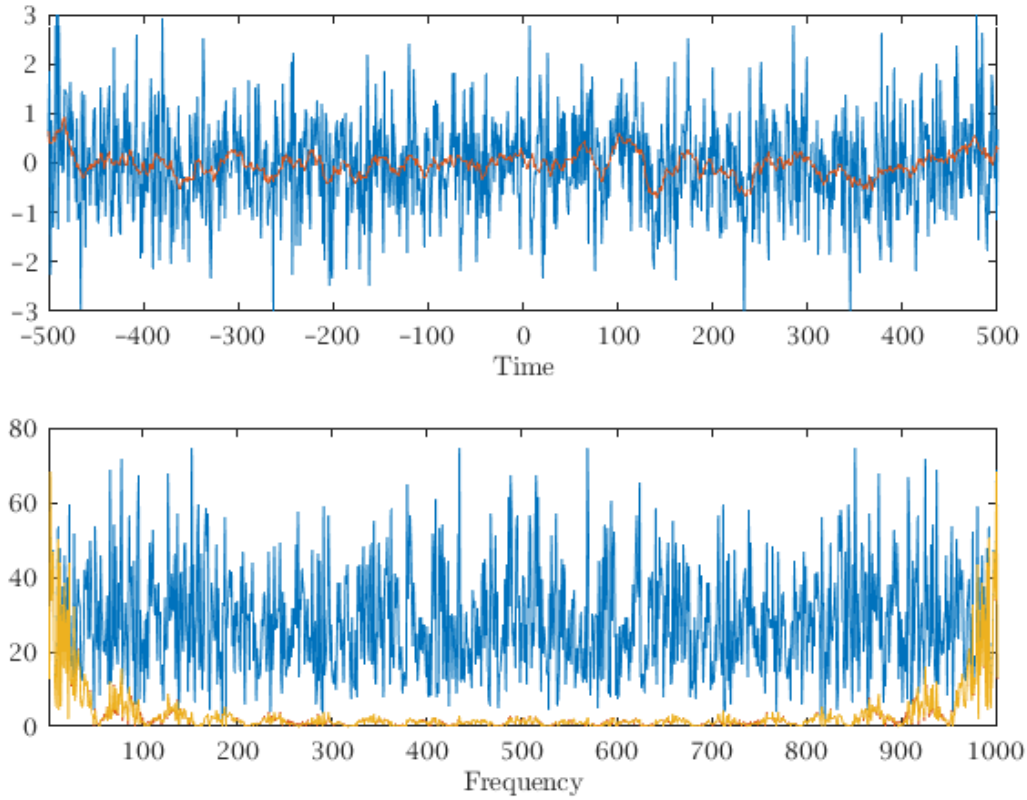


Notice that rather than decaying to zero, the magnitude of the Fourier transform hits zero and bounces back repeatedly. This has undesireable consequences, as we will see.

Now we will use that boxcar to smooth a dataset. For simplicity, we will use a dataset consisting of white noise. We will look at what happens in time and also in frequency.

In [2]:

```
x=randn(1001,1);
fx=vfilt(x,ones(19,1)./19,'mirror');

subplot(2,1,1),plot(t,[x fx]),xlabel('Time'),ylim([-3 3])
subplot(2,1,2),plot(abs([fft(x) fft(fx) fft(x).*fft(g)])),
xlabel('Frequency'),axis tight,ylim([0 80])
```



In the time domain, we see that the orange curve is a smoothed version of the blue curve, as expected. However, in the frequency domain, we see something quite odd.

The very lowest frequencies do indeed pass through, as desired. Yet one also sees bumps of energy extending to high frequencies that also passed through. Evidently the boxcar is not a very good lowpass filter! If we are just interested in quickly smoothing the data, it works fine, but if we are interested in looking at the frequency, we see it leaves a strange imprint.

Our time spent studying Fourier theory has an explanation for what is happening:

"Convolution in the time domain is multiplication in the frequency domain!"

When we smooth our time series by the boxcar function, this is accomplished through the convolution operation. Convolving two time seres, as we have stressed repeatedly, is equivalent to multiplying their Fourier transforms.

In fact we have plotted three curves in the lower panel of this plot. The orange curve is the Fourier transform of our smoothed noise time series. The yellow curve is the Fourier transform of the original noise time series multiplied by the Fourier transform of the boxcar. As you can see, they are virtually identical, as expected from the convolution theorem.

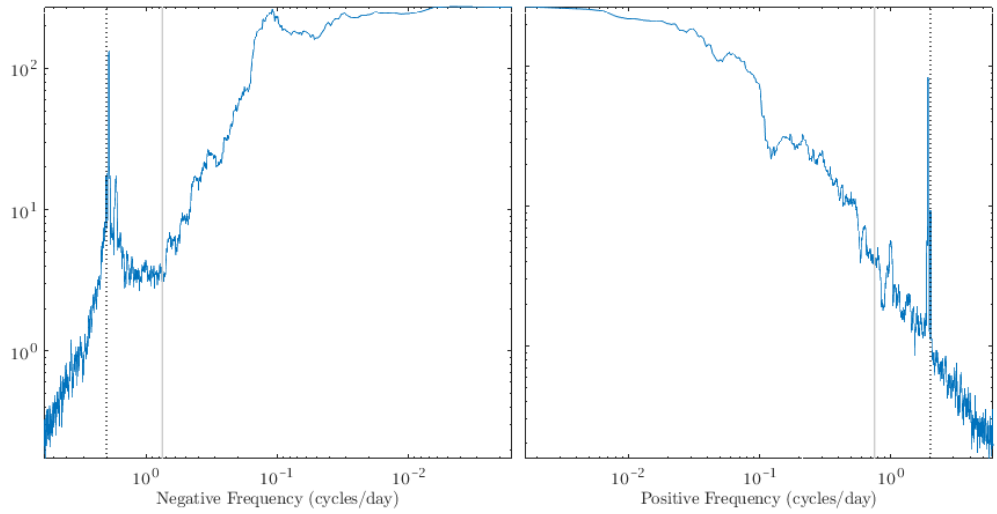
The bumps of the Fourier transform of the boxcar are called “sidelobes” or a “ringing” effect. Such ringing is due to the discontinuous nature of the boxcar. If we want to get rid of this ringing, we need a function that is more smooth in the time domain. This why we use the will typically use the Hanning window for simple smoothing, as it has a nicer Fourier transform.

Bandpass Filtering

Let's turn our attention now to a new application, the construction of a bandpass rather than lowpass filter. For this we will again use the deepest velocity at the m1244 mooring. First recall the rotary spectra of this time series.

In [3]:
```
load m1244
use m1244
dt=num(2)-num(1);          %Sample rate in days
cv=cv(:,4);cv=cv-mean(cv);
[psio,lambda]=sleptap(length(cv),16);
[f,spp,snn,spn]=mspec(dt,cv,psio,'cyclic');
subplot(1,2,1),plot(f,snn),flipx
xlabel('Negative Frequency (cycles/day)')
subplot(1,2,2),plot(f,spp)
xlabel('Positive Frequency  (cycles/day)')
for i=1:2
    subplot(1,2,i)
    xlog,ylog,axis tight,ax=axis;
    vlines(2,'k:'),vlines(max(f)/8,'C')
end
set(gcf,'paperposition',[0 0 11 5 ])
packfig(1,2,'columns') %pack the column subplots together
```

```
SLEPTAP calculating tapers of length 512.
```

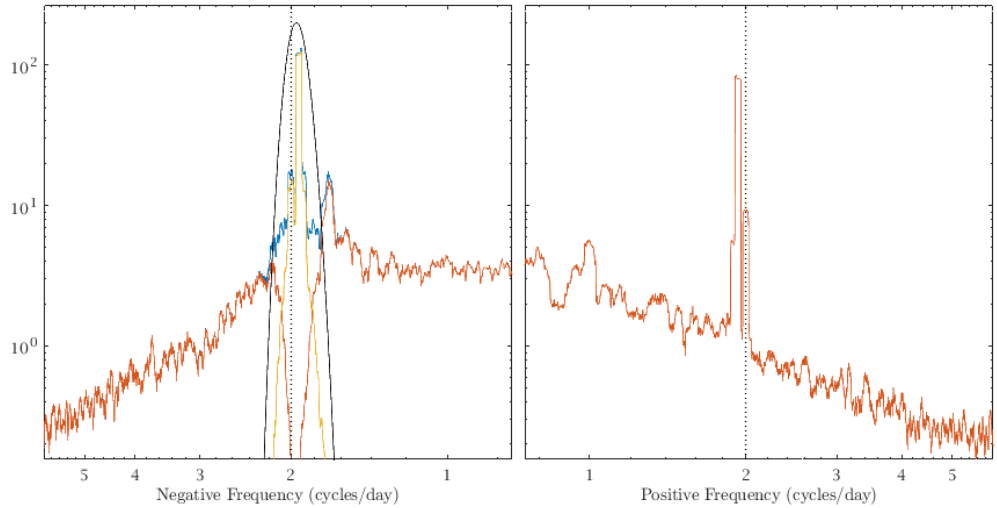


We will construct a narrowband filter to remove the whole inertial peak (which includes a contribution from the semidiurnal tide at this latitude). Let's just run the code and then understand it afterward.

In this next plot, we will zoom on on the region to the high-frequency side of the vertical gray lines in the previous plot.

In [4]:
```
omega=2*pi*1200/length(cv);
cvo=frac(1,sqrt(2))*conj(wavetrans(conj(cv),{3,100,omega}));
cvr=cv-cvo;%Residual minus filtered version
[f,sppo,snno]=mspec(dt,cvo,psio,'cyclic');%Spectrum of filtered version
[f,sppr,snnr]=mspec(dt,cvr,psio,'cyclic');%Spectrum of residual

%The next line returns the filter used by wavetrans
[psi,psif]=morsewave(length(cv),3,100,omega,'bandpass');
subplot(1,2,1),plot(f,[snn snnr snno])
axis(ax),xlog,ylog,xlim([max(f)/8 max(f)]),flipx
hold on,plot(f,50*squared(psif(1:length(f))),'k') %get positive side only
xlabel('Negative Frequency (cycles/day)'),vlines(2,'k:')
```

```
subplot(1,2,2),plot(f,[spp sppr sppo])
axis(ax),xlog,ylog,xlim([max(f)/8 max(f)])
xlabel('Positive Frequency (cycles/day)'),vlines(2,'k:')
set(gcf,'paperposition',[0 0 11 5 ])
packfig(1,2,'columns') %pack the column subplots together
```



Blue is the original time series, yellow is the bandpassed time series, orange is the residual (original-bandpassed), and black is the wavelet (with a rescaled y-axis for presentational purposes). Note that the original and bandpassed time series are virtually identical outside of the wavelet footprint.

As you can see, the negatively-rotating inertial peak has been removed, but the positive spectrum has not been touched at all. This means that the filtered version cvo has effectively isolated the inertial signal.

The wavelet transform essentially convolves a filter, the wavelet, with the original signal. This means we multiply the wavelet's Fourier transform by the Fourier transform of the data. As the wavelet is a *narrowband* signal, this essentially shuts off all frequencies except those inside the band. Subtracting gives us the original signal with that band removed.

Filter Parameter Choices

Let's investigate further the choices of parameters.

The morse wavelets, constructed by the lines

```
morse = analytic_wavelet.GeneralizedMorseWavelet(gamma, beta)
```

```
psi, psif = morse.make_wavelet(len(cv_centered), omega)
```

are controlled by two parameters, gamma and beta, together with the filter frequency omega.

Gamma is a parameter controlling the filter shape, beta is a parameter controlling the filter width in frequency, and omega is the filter center frequency in radians per sample interval. We will understand all of these more later.

Note, a very important feature of the wavelet filter is that it is one-sided. That is, if we tell it to implement a bandpass filter centred around frequency 10, we mean frequency +10 and not

frequency -10. Only positively phasors will be affected and negatively rotating phasors.

To cause the wavelet transform to affect negative frequencies, as we wish to do here for the inertial filtering, we conjugate the input signal---which we saw will conjugate its Fourier transform but also flip positive and negative frequencies. Then to put the frequencies back, we also conjugate the output.

Notice also the factor of 1/sqrt(2) in the line

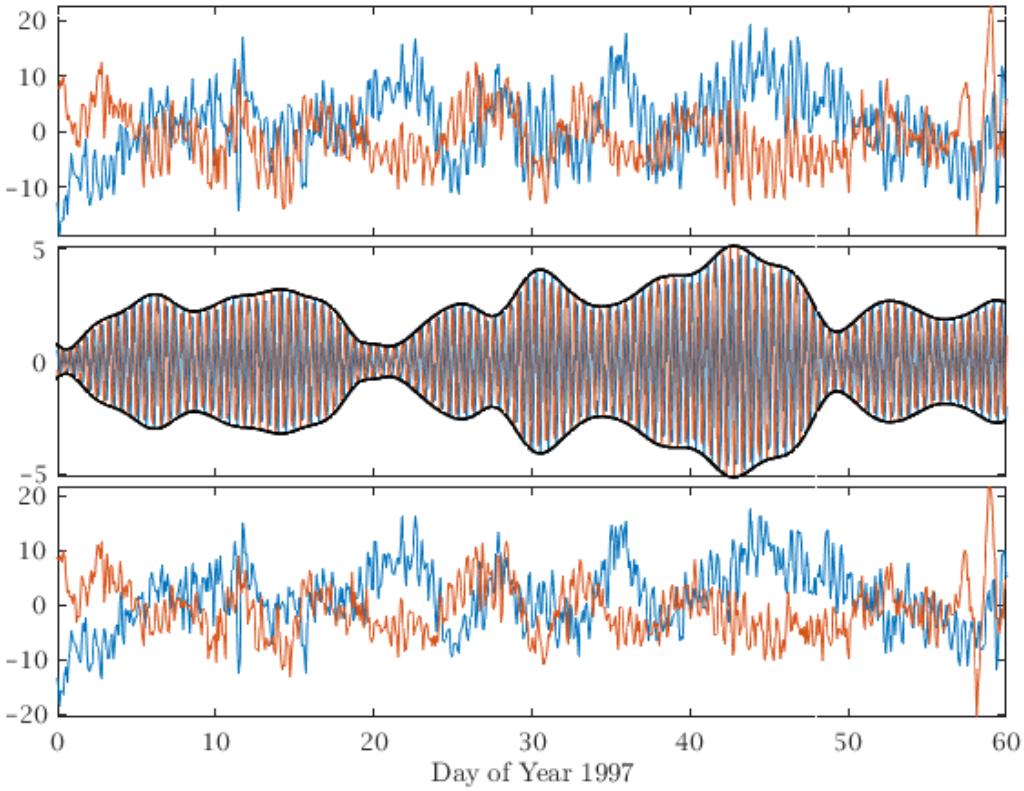
```
cvo = 1/np.sqrt(2)*np.conj(analytic_wavelet.analytic_wavelet_transform...)
```

This factor arises because there is a factor of sqrt(2) ambiguity in defining the amplitude of a real-valued versus a complex-valued sinusoid. Essentially we have defined a real-valued sinusoid to have amplitude one, with a consequence being that when we use wavetrans to form a bandpassed version of a complex valued signal, we need to divide by sqrt(2).

I chose the gamma and beta parameters as follows. Gamma is generally set to 3 for reasons we will see later. I found the frequency by noticing that the tidal peak was centered on about the 1200th Fourier frequency after calling plot(spp). The I picked beta by increasing it until it subjectively looked right.

Here is what the original, bandpassed, and residual time series look like, zoomed in to a 60 day period. Note the different y-limits for the middle subplot.

In [5]:
```
subplot(3,1,1),uvplot(num-datenum(1996,12,31),cv),axis tight
subplot(3,1,2),uvplot(num-datenum(1996,12,31),cvo),axis tight
hold on,plot(num-datenum(1996,12,31),[abs(cvo) -abs(cvo)],'k','linewidth',2)
subplot(3,1,3),uvplot(num-datenum(1996,12,31),cvr),axis tight
for i=1:3,subplot(3,1,i),xlim([0, 60]),end
packfig(3,1,'rows')
xlabel('Day of Year 1997')
```



Rather than just lowpassing, we have selectively removed only the inertial peak. A very useful facet of this analysis is that we now have the amplitude of our extracted signal, the inertial band in this case, as plotted with the heavy curve in the second panel.

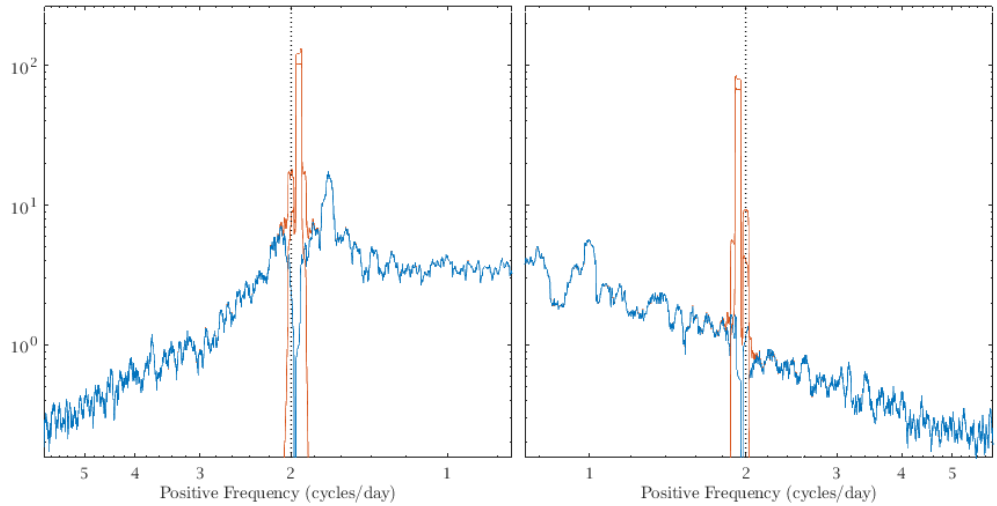
We could also now proceed to look at the variance ellipses, statistics, etc, of the inertial band and residual separately. Combining simple statistics with bandpass filtering in this way can be very illuminating.

Next let's remove the tidal peak instead. This will be done by working with the positive and negative sides separately. However, for reference, code to do the same thing using the real and imaginary parts is also included, If one uses the real and imaginary parts of the data, there is no factor of sqrt(2). But since the wavelet is a one-sided bandpass, It will return something complex-valued if we give it a real-valued input. Consequently we need to take the real part of the both output variables.

In [6]:

```
cvo1=wavetrans(cv,{3,500,2*pi*1200/length(cv)});
cvo2=conj(wavetrans(conj(cv),{3,500,2*pi*1200/length(cv)}));
cvo=frac(1,sqrt(2))*(cvo1+cvo2);
%%This next bit is the same as the above
%[uplus,vplus]=wavetrans(real(cv),imag(cv),{3,500,2*pi*1200/length(cv)});
%cvo=real(uplus)+1i*real(vplus); %Note must take real part!
cvr=cv-cvo;
[f,sppo,snno]=mspec(dt,cvo,psio,'cyclic');
[f,sppr,snnr]=mspec(dt,cvr,psio,'cyclic');
%figure
subplot(1,2,1),plot(f,[snn snnr snno])
axis(ax),xlog,ylog,xlim([max(f)/8 max(f)]),linestyle U T,flipx
xlabel('Positive Frequency (cycles/day)'),vlines(2,'k:')
subplot(1,2,2),plot(f,[spp sppr sppo])
axis(ax),xlog,ylog,xlim([max(f)/8 max(f)]),linestyle U T
xlabel('Positive Frequency (cycles/day)'),vlines(2,'k:')
```

```
set(gcf,'paperposition',[0 0 11 5 ])
packfig(1,2,'columns')
```



We see that the tidal peak has indeed been removed from both sides.

Now that we have a general understanding of the wavelet transform as implementing a one-sided bandpass, we will turn to the wavelet transform itself.

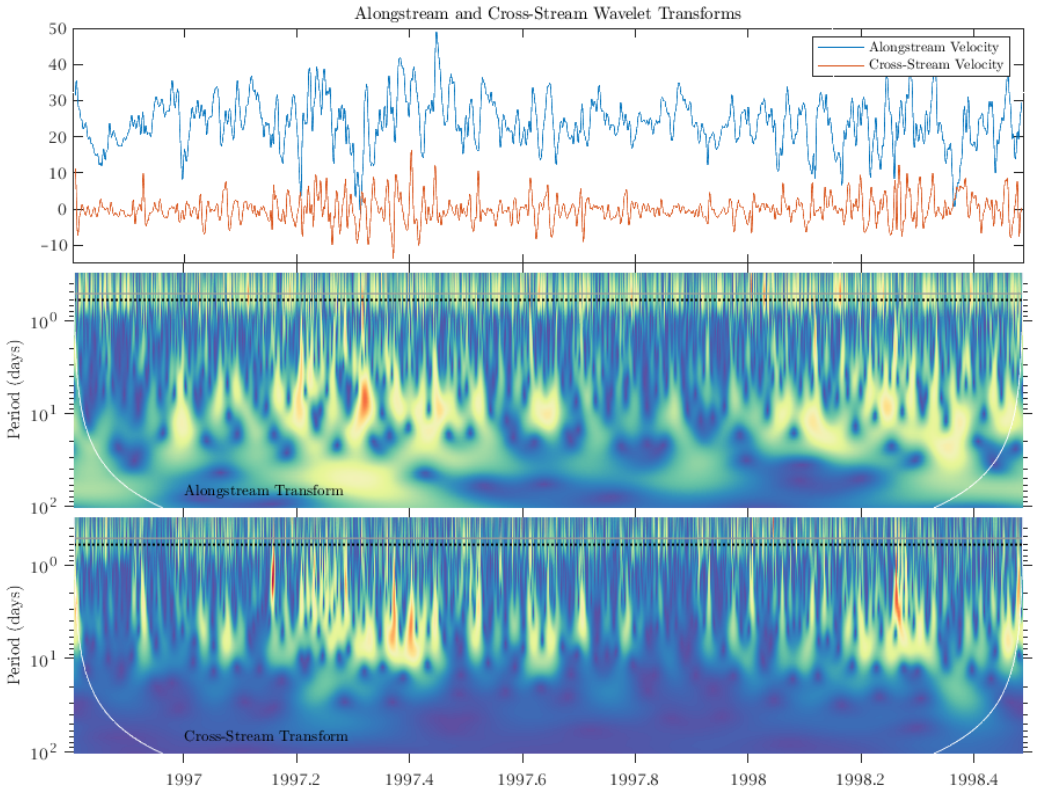
Wavelet Analysis

Wavelet analysis is an informative visualization tool as well as the jumping-off point for a number of rigorous feature identification methods. Let's get a feeling for it by returning to the example of the deepest depth at the m1244 mooring. We'll take a look at the wavelet transforms of the alongstream and cross-stream velocity components we worked with in the earlier labs.

In [19]:

```
use m1244
dt=num(2)-num(1); %Sample rate in days
phi=angle(mean(cv(:,4)));
cv=rot(-phi)*cv(:,4);

gamma=3;beta=2;
fs=morsespace(gamma,beta,length(cv));
[wu,wv]=wavetrans(real(cv),imag(cv),{gamma,beta,fs});

h=wavespecplot(yearfrac(num),vfilt(cv,24),dt*2*pi./fs,wu,wv);
axes(h(1)),ylim([-15 50]),caxis([0 15])
legend('Alongstream Velocity','Cross-Stream Velocity')

for i=2:3
    axes(h(i)),ylabel('Period (days)')
    hlines(1./(tidefreq('m2')*24/2/pi),'E')
    hlines(1./(corfreq(lat)*24/2/pi),'1.5k:')

    %width of edge effect region in days
    L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
    plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
end

axes(h(1)),title('Alongstream and Cross-Stream Wavelet Transforms')
axes(h(2)),text(1997,70,'Alongstream Transform','color','k')
axes(h(3)),text(1997,70,'Cross-Stream Transform','color','k')
```

```
set(gcf,'paperposition',[0 0 11 8 ])
```



The curving white lines mark the so-called edge effect regions, outside of which the transform should be regarded as untrustworthy. The simple expression for the edge effect regions that we use here is derived in this paper, see Eqn. 2.8 and Figure 3 therein.

This decomposes the variability as a function of both time and scale, where the latter can be interpreted as a period. Here, we see three major features. The first is an internal wave / tidal band. For reference the period of the semidiurnal tide (a half day) has been marked with a solid gray line, and that of the Coriolis frequency (about 14.6 hours at this latitude) has been marked with a dotted black line. A band of energetic variability is seen in both transforms that is localized at shorter periods, or higher frequencies, than the inertial period.

The second major feature is a series of vertically elongated structures that have their maximum expression in the range of periods from 2 to 10 days. This is a very characteristic appearance of eddy-like variability. In general, wave-like variabilty tends to appear as a horizontal band, where time-localized features such as eddies tend to appear in the wavelet transform as vertically elongated features. Notice that the eddy band possibly (?) presents some seasonal variability, with elevated levels in the range of 0.2--0.4 (February to April) in both years.

A final feature is low-frequency energy, below about a ten day period, which is seen only in the alongstream and not in the cross-stream component.

Thus, the main message here is that the variability seen in the spectrum arises from different structural causes, in particular, a wave-like aspect together with an eddy-like aspect. This is not apparent in the spectrum as it does not show temporal structure.

It should be pointed out that what we are doing with the wavelet transform at this point is only visualization. In order to go further and say something quantitative about the variability takes a lot more work. Therefore most of the time we are satisfied with a qualitative or impressionistic interpretation of the wavelet transform. For certain problems, we can create (or apply) very powerful quantitative methods based on the wavelet transform, but we have to proceed carefully because they take a long time to learn.

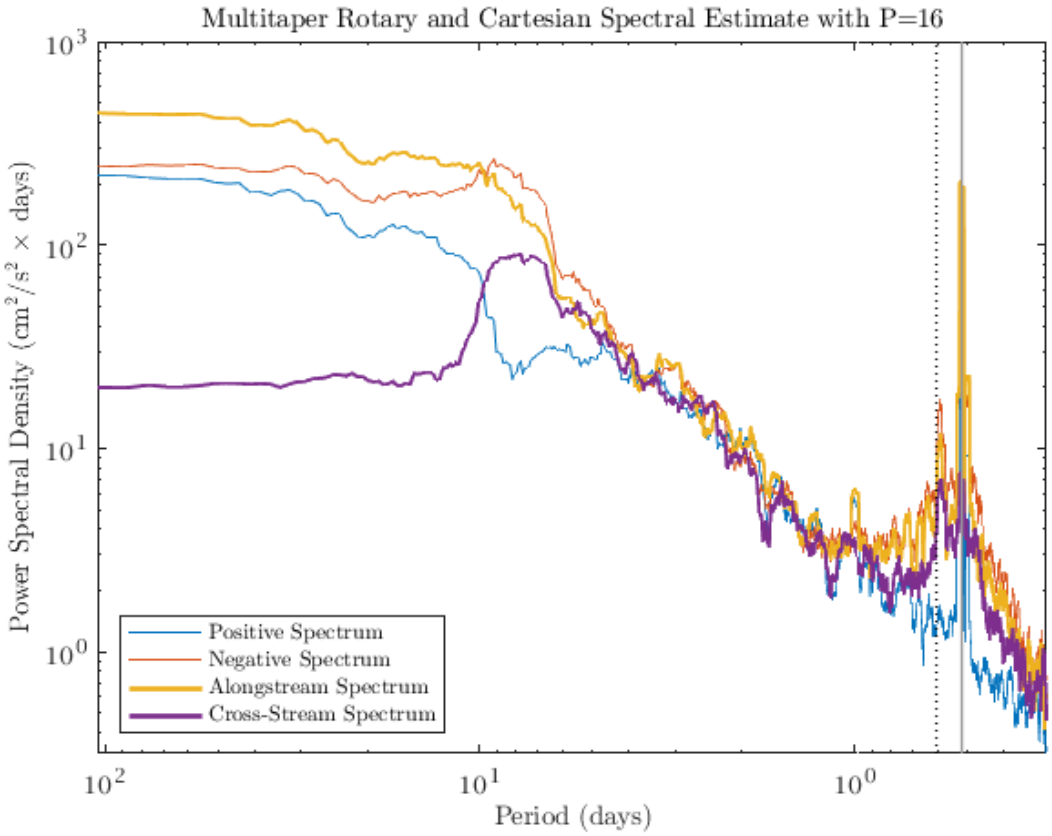
The most important thing to note about the wavelet transform is that it is not a property of your time series. It is a *joint function* of your time series and the analyzing wavelet. Therefore, it is essentially to always show the time series together with the transform! In addition, you should always at least tell the wavelet parameters, if not also show the wavelet. Showing a wavelet transform without the associated time series is totally meaningless.

For comparison, we'll show the multitaper spectral estimate we created a few days ago. To facilitate comparision with the wavelet transform, we will use period rather than frequency as the x-axis, and set the limits to match the wavelet transform.

In [20]:

```
P=16;
psi=sleptap(length(cv),P);
[f,suu,svv]=mspec(dt,real(cv),imag(cv),psi,'cyclic');

h=plot(1./f,[spp snn 2*suu 2*svv]);,xlog,ylog,ylim(10.^[-0.5 3]),flipx
%See comment on factors of 2 in spectrum lab

%Set x-limits of Fourier spectrum to the same as those of the wavelet transform
xlim([min(dt./(fs/2/pi)) max(dt./(fs/2/pi))])

linestyle 0.5T 0.5U 2V 2W
vlines(1./(tidefreq('m2')*24/2/pi),'E')
vlines(1./(corfreq(lat)*24/2/pi),'k:')

xlabel('Period (days)')
ylabel('Power Spectral Density (cm$^2$/s$^2$ $\times$ days)')
legend(h,'Positive Spectrum','Negative Spectrum','Alongstream Spectrum','Cross-Stream
title('Multitaper Rotary and Cartesian Spectral Estimate with P=16')
```

```
SLEPTAP calculating tapers of length 512.
```



The Coriolis period and semidiurnal tidal period are marked as they were in the wavelet transform.

In the spectrum, we see again the three regimes of a small-scale wave band, a sloping isotropic region in the range from 1 day to 10 days period (corresponding to the eddy band in the wavelet transform), and finally a low-frequency region below 10 days period in which the alongstream spectrum levels are substantially elevated relative to the cross-stream levels.

It should be clear that the wavelet transform and the specta are representing these regimes very differently and are therefore complementary. The wavelet transform gives us information about what is happening as a function of time, whereas the spectra have superior frequency resolution and are able to, for example, resolve multiple separate lines at high frequencies where the wavelet transform sees only a broad band.

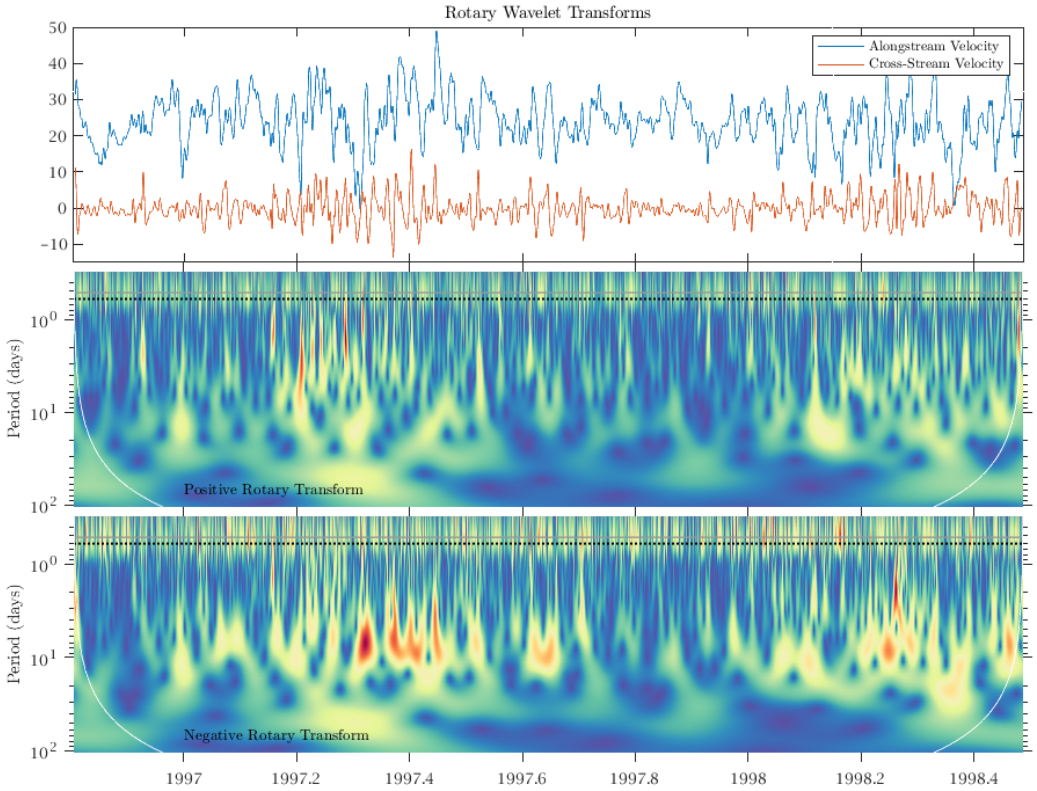
You might think that if you average horizontally in the wavelet transform, you would obtain an estimate of the Fourier spectrum. This is true in a sense, however, an adjustment has to be made for the different ways the Fourier spectrum and wavelet transform are representing variability across different scales or frequencies. See Section 4a of this paper for details.

Let's take a look at the positive and negative rotary wavelet transforms.

In [21]:

```
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(cv));
[wp,wn]=wavetrans(cv,conj(cv),{gamma,beta,fs});
%We compute the negative rotary transform with a conjugation

h=wavespecplot(yearfrac(num),vfilt(cv,24),dt./(fs/2/pi),wp,wn);
axes(h(1)),ylim([-15 50]),caxis([0 15])
legend('Alongstream Velocity','Cross-Stream Velocity')
```

```
for i=2:3
    axes(h(i)),ylabel('Period (days)')
    hlines(1./(tidefreq('m2')*24/2/pi),'E')
    hlines(1./(corfreq(lat)*24/2/pi),'1.5k:')

    L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
    plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
end
axes(h(1)),title('Rotary Wavelet Transforms')
axes(h(2)),text(1997,70,'Positive Rotary Transform','color','k')
axes(h(3)),text(1997,70,'Negative Rotary Transform','color','k')
set(gcf,'paperposition',[0 0 11 8 ])
```



We see that the high-frequency band is stronger on the negative rotary side, as expected. There is also a hint that the energetic eddy-like events extend to somewhat larger scales on the negative rotary side then on the positive rotary side; we see the same hint in the rotary Fourier spectra above. This could suggest a hypothesis regarding eddy polarity or eddy center cross-stream placement, as both of those will effect the rotation sense of currents of advected eddies. No significant asymmetry is seen in the rotary transforms at periods longer than 10 days, in keeping with the rotary Fourier spectra.

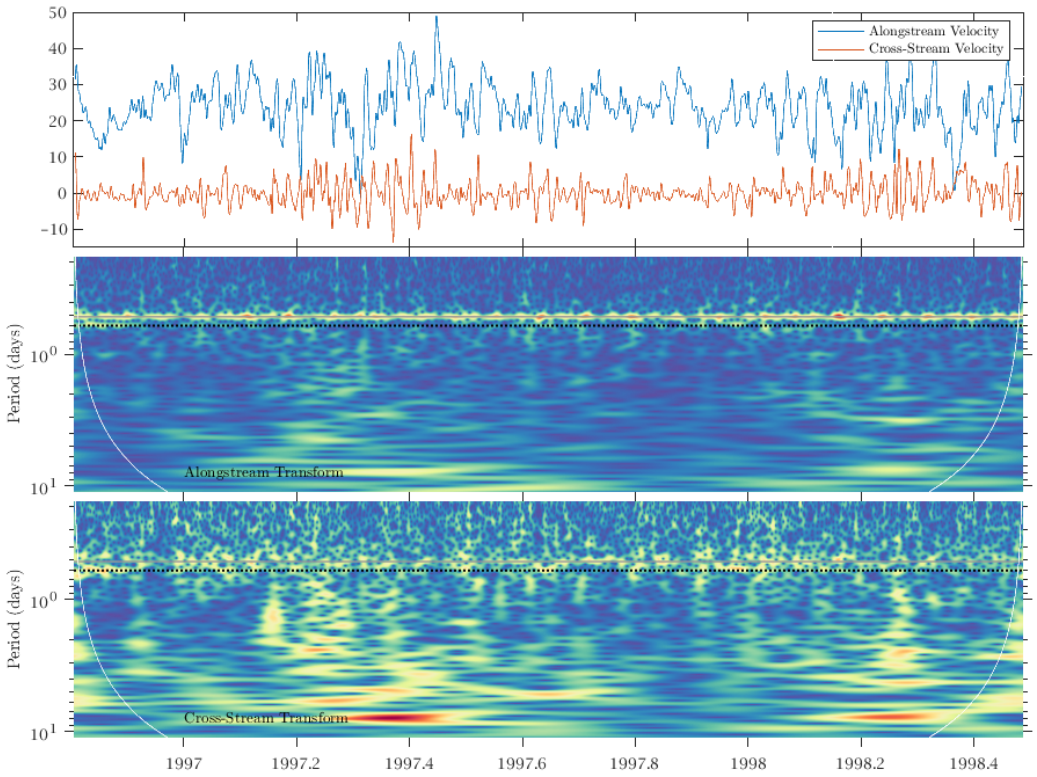
All in all, this plot appears less informative than the alongstream/downstream plot, although there are other cases where the rotary wavelet analysis is quite revealing.

Let's see what happens if we change the wavelet transform to try to obtain a high resolution in frequency. We're looking here at the alongstream/cross-stream transform.

In [22]:

```
gamma=3;beta=200;
fs=morsespace(gamma,beta,length(cv));
[wu,wv]=wavetrans(real(cv),imag(cv),{gamma,beta,fs});
```

```
h=wavespecplot(yearfrac(num),vfilt(cv,24),dt./(fs/2/pi),wu,wv);
axes(h(1)),ylim([-15 50]),caxis([0 15])
legend('Alongstream Velocity','Cross-Stream Velocity')

for i=2:3
    axes(h(i)),ylabel('Period (days)')
    hlines(1./(tidefreq('m2')*24/2/pi),'E')
    hlines(1./(corfreq(lat)*24/2/pi),'1.5k:')

    L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
    plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
end
axes(h(2)),text(1997,8,'Alongstream Transform','color','k')
axes(h(3)),text(1997,8,'Cross-Stream Transform','color','k')
set(gcf,'paperposition',[0 0 11 8 ])
```



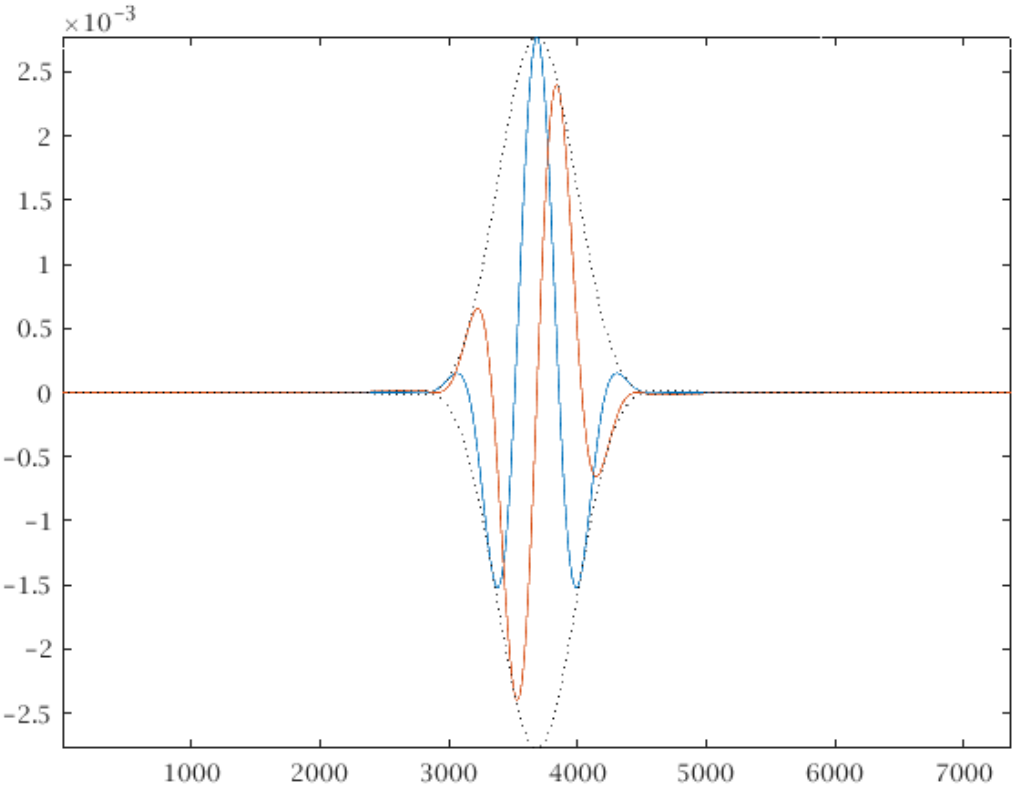
We do see much improved frequency resolution---the wavelet transform exhibits a much narrower stripe in the vicinity of the semidiurnal peak---but this comes at the expense of degraded temporal resolution, wiping out essentially all the temporal structure we could see before. Note also the much smaller y-axis: we're only resolving up to about 10 days, corresponding to the eddy band identified earlier, because our time series just isn't long enough to support an analysis using these settings.

This is a terrible plot. Most of what you are seeing here are meaningless artifacts. Please do make plots like this one. Its point is simply to show that resolving details of frequency structure is not really what wavelets are good at.

A Look at the Wavelets

To understand the wavelet transform, let's first take a look at the wavelets. Here we look at one particular wavelet, from the 55th frequency band from the earlier plot.

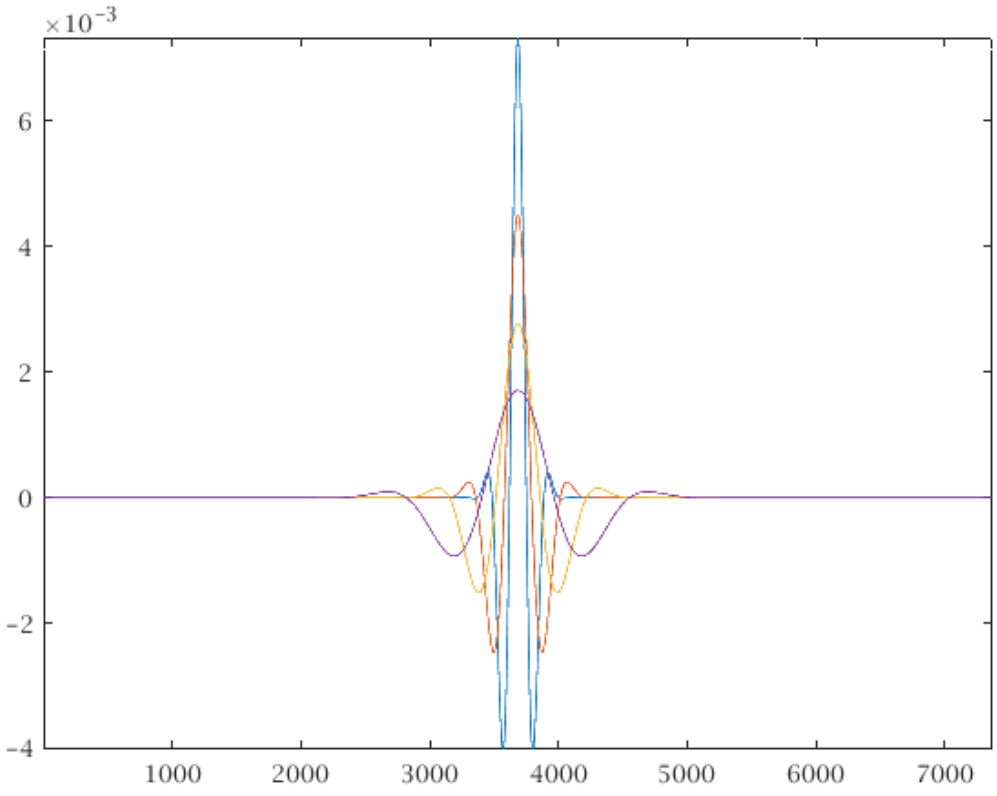
In [29]:
```
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(cv));
[psi,psif]=morsewave(length(cv),gamma,beta,fs);
uvplot(psi(:,55)),axis tight
fso=fs(55);  %Remember this frequency for later
hold on, plot(abs(psi(:,55)),'k:')
hold on, plot(-abs(psi(:,55)),'k:')
```



As you can see, the wavelet strongly resembles a complex exponential multiplied by a Gaussian. The parameter beta sets the number of oscillations within the window, as we shall see later. The number of oscillations is about $2P/\pi$, where $P=\sqrt{\beta \gamma}$. Here $P=\sqrt{6}\approx 2.45$ so $2P/\pi\approx 1.56$. The wavelet executes about one and a half complete periods before it decays. This evident, for example, in the fact that the real part obtains two smaller maxima off-center, indicating that a full cycle has been completed.

The variable psi contains an array of wavelets, one at each frequency in fs. Let's look at some more of the wavelets. The real parts of four of the longer wavelets in the time domain are shown in the next figure.

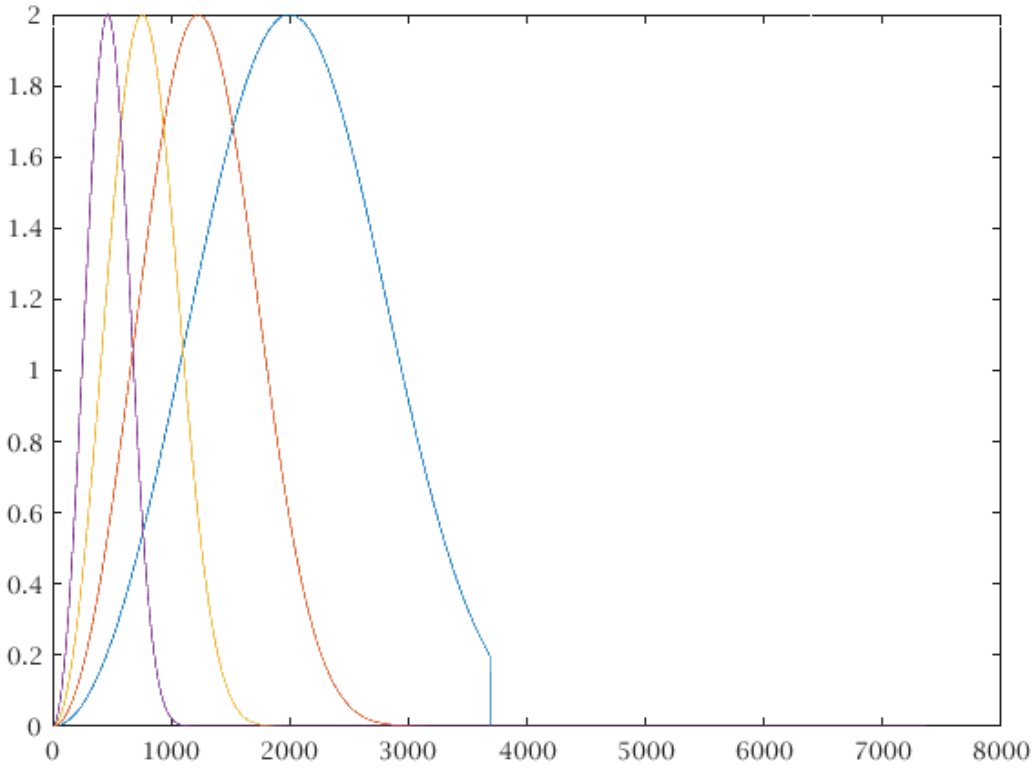
In [30]:
```
plot(real(psi(:,[45 50 55 60]))),axis tight
```



It should be clear that these are merely stretched-out and rescale copies of each other. The wavelet transform involves a *convolution* or *smoothing* operation of these rescaled functions, thus localizing the variability in different frequency bands, due to the scaling theorem combined with the convolution theorem.

Four of the shorter wavelets are shown in the Fourier domain in the next figure.

In [31]:
```
plot(psif(:,[1 6 11 16]))
```



As you can see they are bandpass filters centered on different bands according to the scaling theorem.

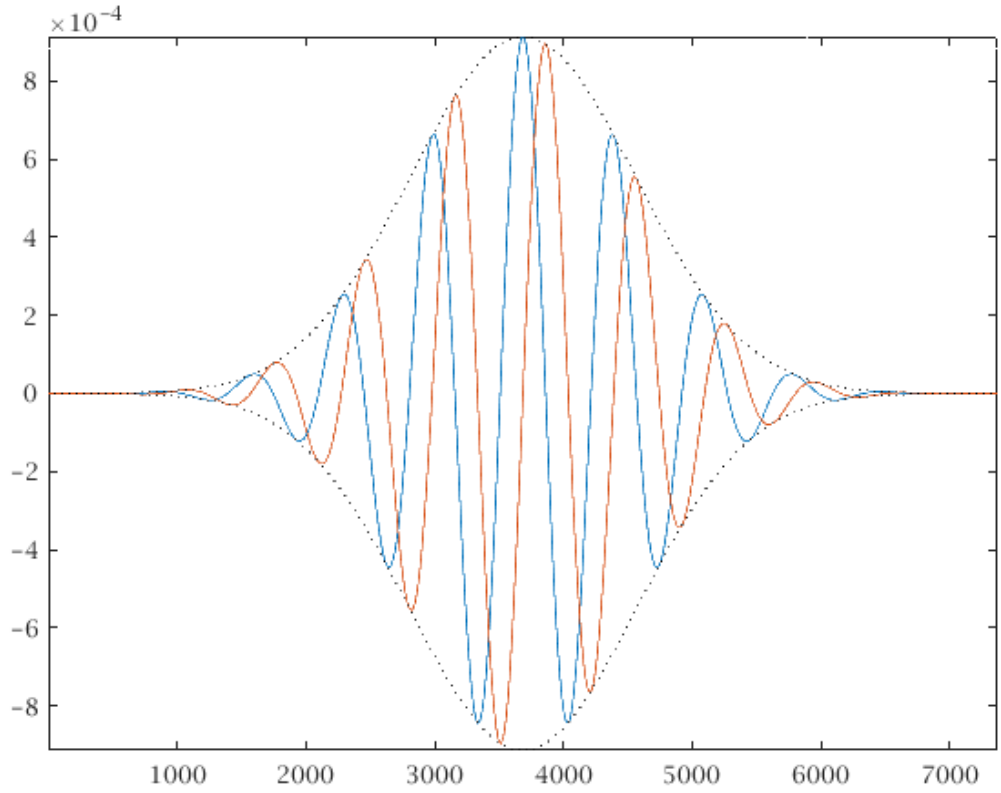
Here we are just plotting the discrete Fourier transform with Matlab's usual frequency convention. The shortest wavelet in time is the blue wavelet, which is the most extended in the frequency domain; this is a consequence of the scaling theorem we discussed in class. The shortest wavelet is truncated at the Nyquist because these wavelets are set to vanish exactly for negative frequencies. The wavelet code chooses the highest-frequency wavelet such that this truncation does not exceed a specified threshold, chosen to be 1/10 of the wavelet's peak value.

Observe that the wavelets all have the same peak value, which is equal to two. Thus, the wavelets are acting as *bandpass filters*; this is a consequence of a particular choice of normalization that is made by default in `wavetrans`. Another type of normalization demands that the rescaled wavelets all have the same energy, which gives a different amplitude scaling in the frequency domain. We find the bandpass normalization to be prefereable. The factor of 2 ensures that the maximum magnitude of the wavelet transform of $\cos\omega t$ is equal to one for any choice of ω.

If we increase β, the wavelet becomes more oscillatory. Let's take a look choosing $\beta=20$.

In [32]:

```
gamma=3;beta=20;
%Recompute the frequency array, with non-default settings
fs=morsespace(gamma,beta,{0.1,pi},{1,length(u)});
[psi,psif]=morsewave(length(u),gamma,beta,fs);
%Find the new wavelet having the same frequency as wavelet #55 we looked at earlier
[temp,index]=min(abs(fs-fso));
uvplot(psi(:,index)),axis tight
hold on, plot(abs(psi(:,index)),'k:')
hold on, plot(-abs(psi(:,index)),'k:')
```



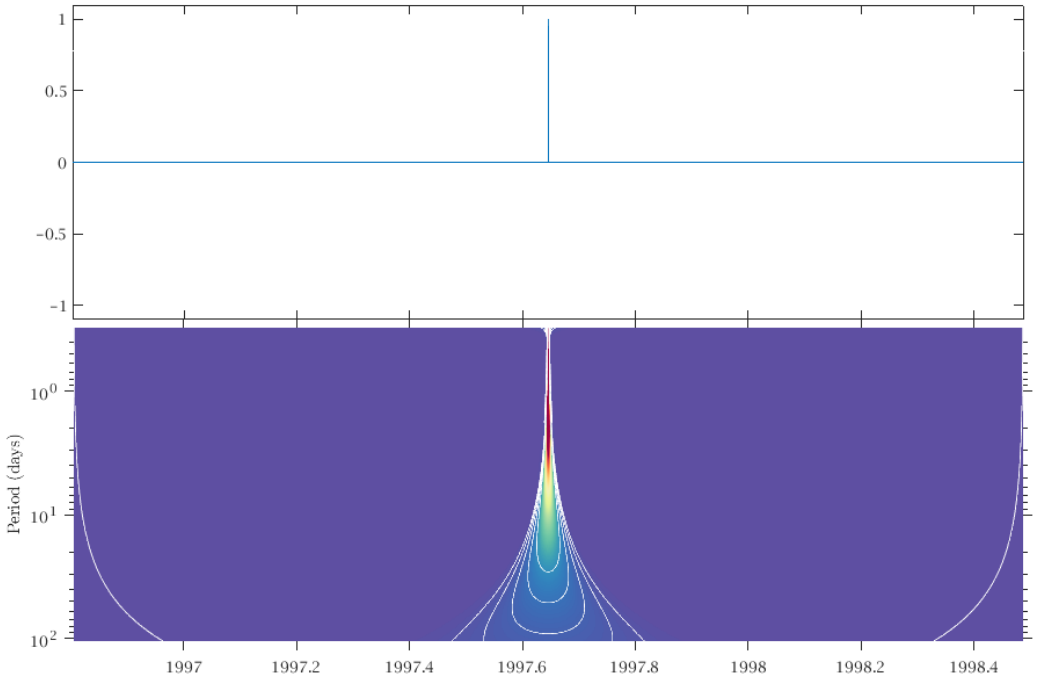
As you can see, the wavelet is more oscillatory. Recalling the rule of thumb that the number of complete cycles in the wavelet is about $2P/\pi=2\sqrt{\beta\gamma}/\pi=2\sqrt{60}/\pi\approx 5$, we find this agrees well with the plot.

Understanding the Wavelet Transform

A good way to understand what the wavelet transform is telling you is to take wavelet transforms of simple signals. First, we will take the wavelet transform of a delta function, then some boxcar or rectangle functions, then some sinusoids, and finally some noise processes.

In [92]:

```
u=zeros(size(cv)); %Create a time series of zeros the same length as my data
u(round(end/2))=1; %Set the central point to a value of one

%This is all essentially the same as before
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(u));
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),u,dt./(fs/2/pi),wu);  %Convert rad/day to cycles/day
axes(h(1)),ylim([-1.1 1.1])
axes(h(2)),ylabel('Period (days)'),
contour(yearfrac(num),dt./(fs/2/pi),log10(abs(wu')),[-3.25:0.25:-2.25],'w')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
caxis([0 .05])
set(gcf,'paperposition',[0 0 11 7 ])
```

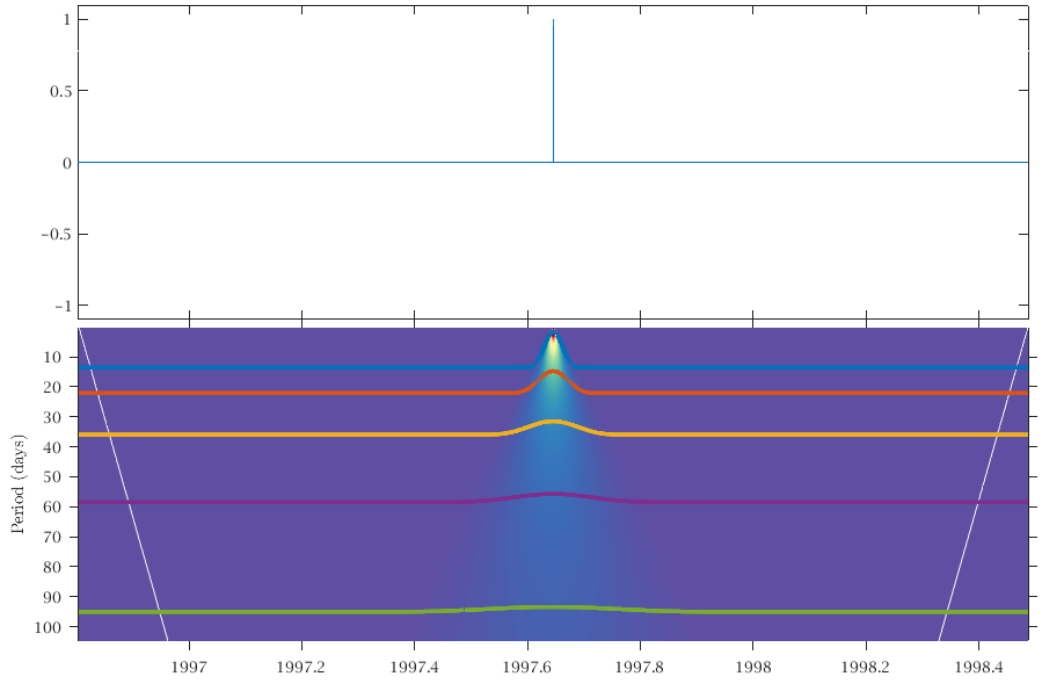


Here we've added some contours of the wavelet transform modulus for reference. We see the wavelet transform has the shape of a cusp pointing upwards toward the discontinuity. Note that the cone of influence around the discontinuity is essentially the same as the edge effects.

It is informative to plot this with a linear y-axis.

In [93]:

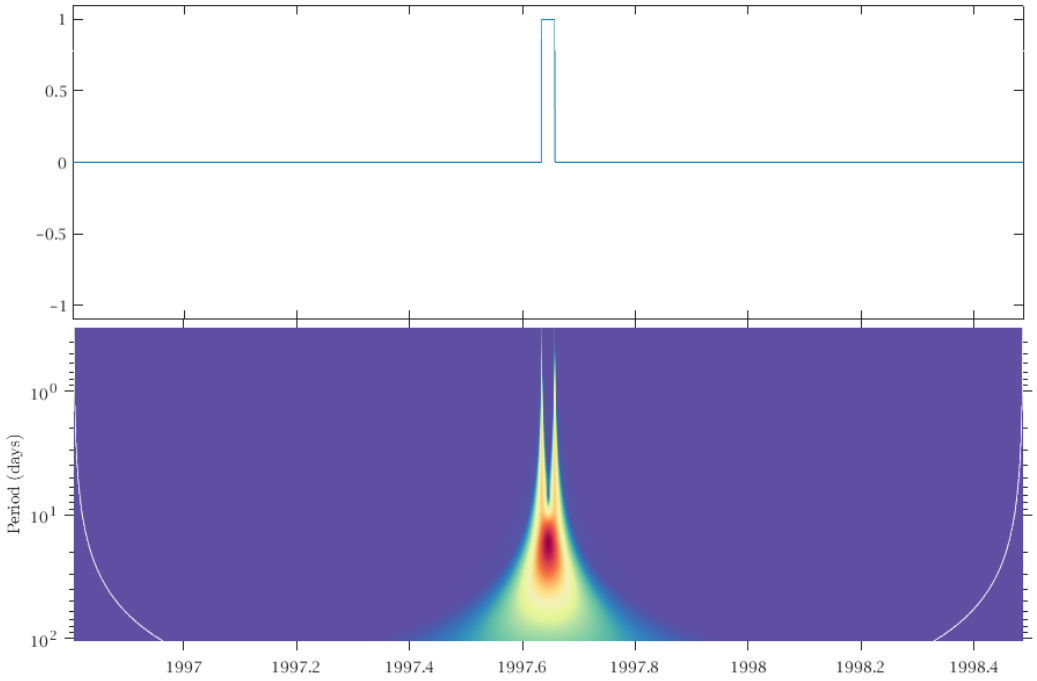
```
h=wavespecplot(yearfrac(num),u,dt./(fs/2/pi),wu);  %Convert rad/day to cycles/day
axes(h(1)),ylim([-1.1 1.1])
axes(h(2)),ylabel('Period (days)'),ylin
%contour(yearfrac(num),dt./(fs/2/pi),log10(abs(wu')),[-4:0.25:10],'w')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
set(gcf,'paperposition',[0 0 11 7 ])
caxis([0 .05])

index=[40 45 50 55 60];
sty='TUVWXYZ';
for i=1:length(index)
    h=plot(yearfrac(num),-1000*(abs(wu(:,index(i))))+dt./(fs(index(i))/2/pi));
    linestyle(h,['3' sty(i)])
end
```



Here we've added colored lines at several scales that show the shape of modulus of the wavelet transform. This is in fact the modulus of rescaled versions of the wavelet itself, since the wavelet transform of a delta function places rescaled copies of the wavelet at each period or scale.

In [103…

```
u=zeros(size(cv));                %Create a time series of zeros the same length as my
u(round(end/2)-50:round(end/2)+50)=1; %Set the central 101 points to a value of one

%This is all essentially the same as before
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(u));
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),u,dt./(fs/2/pi),wu);  %Convert rad/day to cycles/day
axes(h(1)),ylim([-1.1 1.1])
axes(h(2)),ylabel('Period (days)')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
set(gcf,'paperposition',[0 0 11 7 ])
```

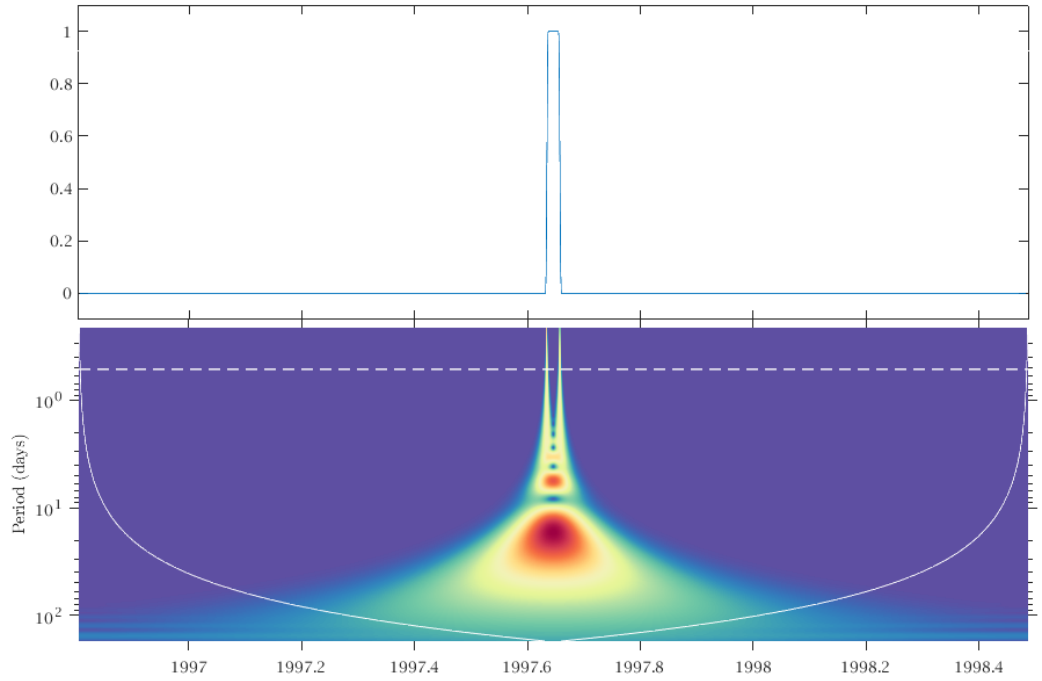


At high frequencies, we have two upward-pointing wedges indicating the locations of the discontinuities. This is quite typical of the way the wavelet transform represents time-localized signals. The period at which the transform maximum occurs is indicating the duration of the event. At scales at which the wavelet is larger than the signal---which occurs around where the two 'wedges' join together---we are essentially seeing the *wavelet*. That is, as you recall the wavelet transform is just a smoothing with stretched-out versions of the wavelet. That is what you are seeing between periods of about 10 days and 100 days: the magnitudes of the wavelets themselves. This is why I emphasize that the wavelet transform is a *joint function* of the signal and the analyzing wavelet.

To emphasize this, let's change the wavelet to the more oscillatory set used above.

In [104…

```
gamma=3;beta=20;
%Recompute the frequency array, with non-default settings
fs=morsespace(gamma,beta,{0.1,pi},{1,length(u)});
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),vfilt(u,24),dt./(fs/2/pi),wu);  %Convert rad/day to cycl
axes(h(1)),ylim([-.1 1.1])
axes(h(2)),
ylabel('Period (days)')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');

set(gcf,'paperposition',[0 0 11 7 ])
```



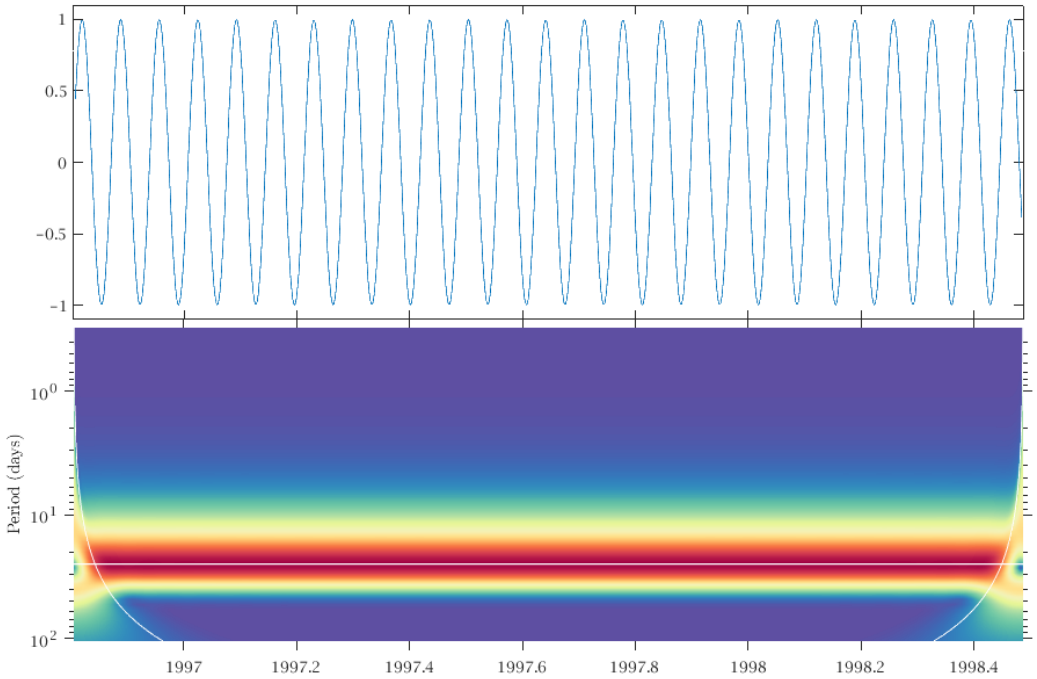
Although the signal has not changed, the wavelet transform has. Some small-scale structure has emerged due to the interaction of the boxcar with the oscillations of the wavelet; this is not meaningful, and indicates that one should not use a highly oscillatory wavelet to analyze a highly localized signal! The stretching out of the signal in time for periods greater than about 10 days is due to the change in width of the *wavelet*.

Thus, as I keep emphasizing, unless one knows the wavelet one cannot correctly interpret the wavelet transform. A wavelet is like a lens. It lets you see aspects of your signal, but it blurs other aspects. Unless you know the nature of the lens, you cannot interpret the image that it presents to you.

We will proceed now to take the wavelet transform of a highly *frequency-localized* signals as opposed to the highly time-localized boxcar.

In [105…

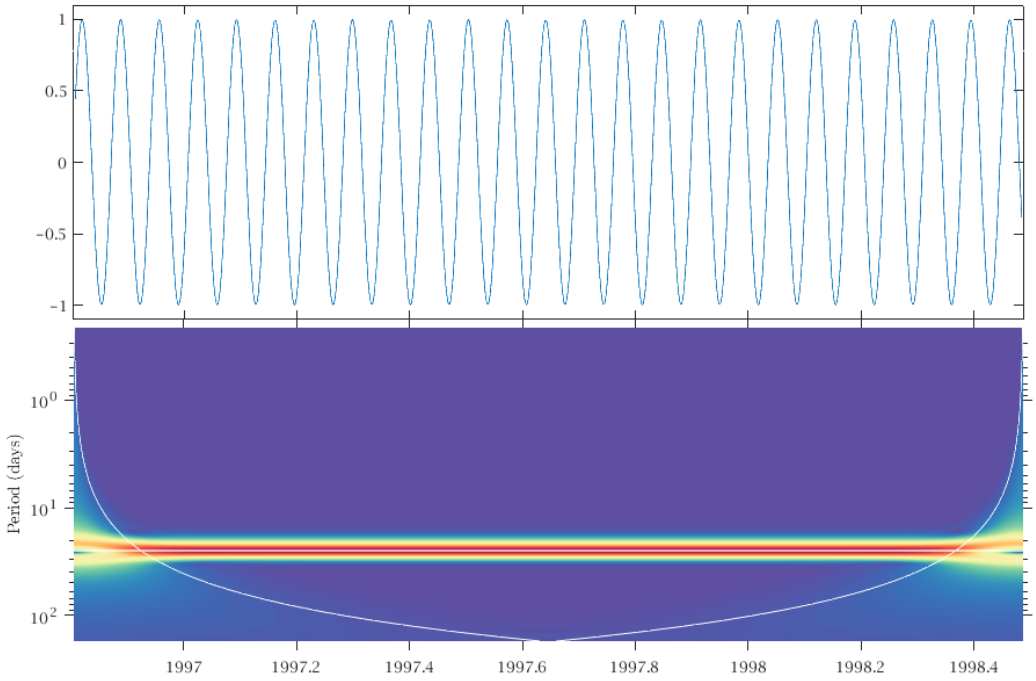
```
u=cos(2*pi*2*num/50);  %Create a low-frequency cosine

%This is all essentially the same as before
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(u));
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),vfilt(u,24),dt./(fs/2/pi),wu);  %Convert rad/day to cycl
axes(h(1)),ylim([-1.1 1.1])
axes(h(2))
ylabel('Period (days)')
hlines(1./(2/50),'w')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
set(gcf,'paperposition',[0 0 11 7 ])
```



Here the location of the cosine's frequency has been marked with a solid white line. The question here is how to interpret the vertical width of this band? The width of this band is due to the fact that we are seeing the frequency-domain extension of the wavelet. Changing to the more oscillatory set of wavelets, the transform again changes.

In [97]:

```
gamma=3;beta=20;
%Recompute the frequency array, with non-default settings
fs=morsespace(gamma,beta,{0.1,pi},{1,length(u)});
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),vfilt(u,24),dt./(fs/2/pi),wu);  %Convert rad/day to cycl
axes(h(1)),ylim([-1.1 1.1])
axes(h(2)),hlines(1./(tidefreq('m2')*24/2/pi),'w--')          %Convert rad/hour to cyc
ylabel('Period (days)')
hlines(1./(2/50),'w')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
set(gcf,'paperposition',[0 0 11 7 ])
```



The band in the wavelet transform is narrower in the frequency direction because the wavelet are narrower.

You can't get around seeing the imprint of the wavelet, but you can account for it mentally. In general, if the signal structure is broader than the wavelet in a certain direction (time or frequency), the wavelet transform will show the signal; but if the signal is narrower, the wavelet transform will show the wavelet

These look totally different. Here we see again something that we observed in the spectra, which is that the cross-stream flow lacks a low-frequency component that is present in the downstream flow. One tentative hint emerging from this plot is that the time period of greatest activity in the eddy band is also the time period at which the low-frequency variability is strongest; could they be related? Such information about timing of variability cannot be recovered from the spectrum.

Finally, just as we can compute the rotary Fourier power spectrum, we can also compute the rotary wavelet spectrum. This quantifies the positively and negatively-rotating variability, but distributed across the time/frequency plane.

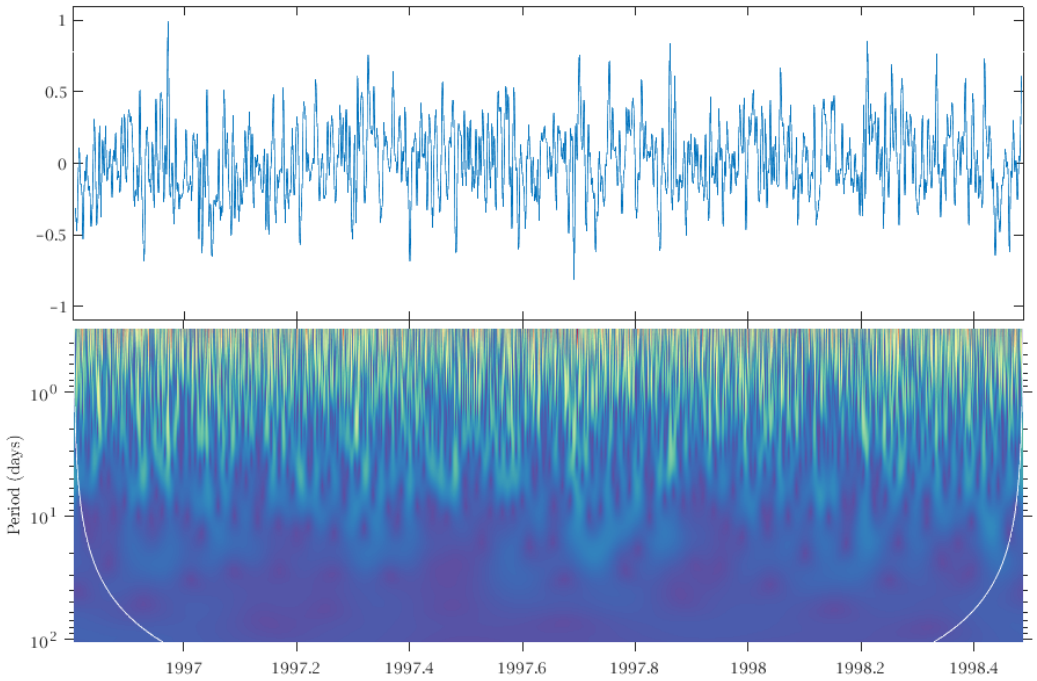
The time variability is now very spread out, but the tidal peak has become very narrow. This plot is less informative overall, but does show the tides better. If we continued like this, we would eventually be able to discern parallel lines from the different tidal components. This illustrates the tradeoff between time and frequency resolution. Generally for visualizing variability, a more time-localized setting is more appropriate.

Let's finish by looking at the transforms of an array of some noise processes. Here is the transform of Gaussian white noise.

In [116…

```
u=randn(size(num));  %Create a array of white noise

%This is all essentially the same as before
```

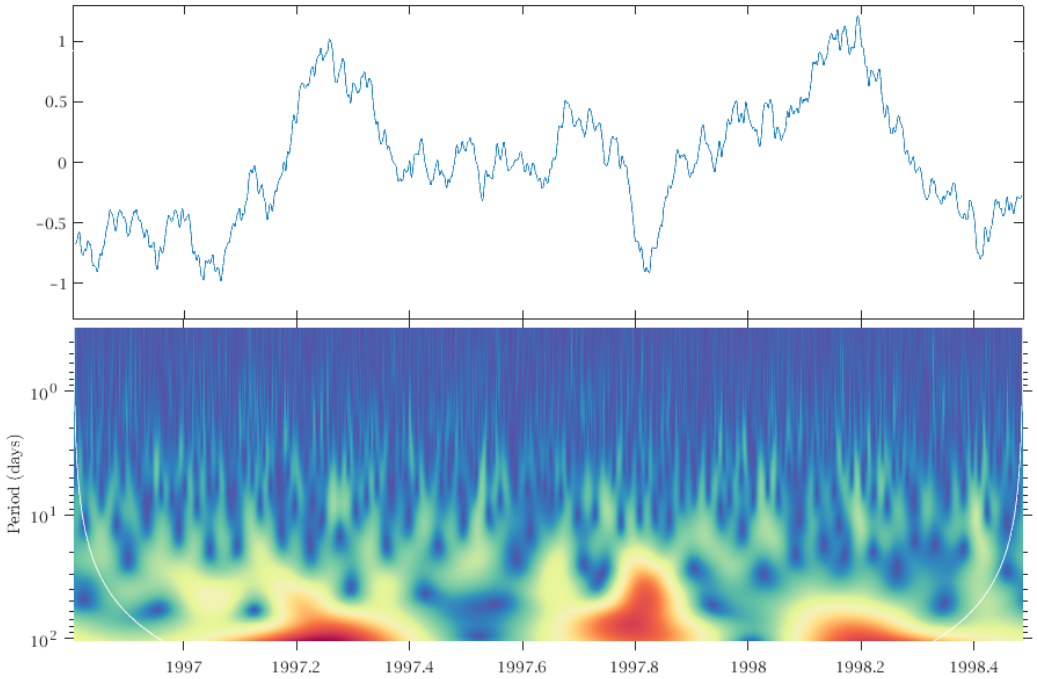
```
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(u));
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),vfilt(u,24),dt./(fs/2/pi),wu);  %Convert rad/day to cycl
axes(h(1)),ylim([-1.1 1.1])
axes(h(2))
ylabel('Period (days)')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
set(gcf,'paperposition',[0 0 11 7 ])
```



And here is the transform of the integral of a red noise processes, defined as the cumulative sum of white noise.

In [118…

```
rng(1);%set the random number generator seed so this plot always looks the same
u=cumsum(randn(size(num)));  %Create a array of red noise
u=u-mean(u);u=u./std(u)/2;

%This is all essentially the same as before
gamma=3;beta=2;
fs=morsespace(gamma,beta,length(u));
wu=wavetrans(u,{gamma,beta,fs});

h=wavespecplot(yearfrac(num),vfilt(u,24),dt./(fs/2/pi),wu);  %Convert rad/day to cycl
axes(h(1)),ylim([-1.3 1.3])
axes(h(2))
ylabel('Period (days)')
L=dt*2*sqrt(2)*sqrt(gamma*beta)./fs;
plot(yearfrac([num(1)+L/2 num(end)-L/2]),dt*2*pi./fs,'color','w');
set(gcf,'paperposition',[0 0 11 7 ])
```



Finish

Try making some narrowband filtrations and also wavelet transforms of your data. It is a good idea to also take wavelet transforms of some simple signals of the same length as your data, such as a boxcar and a sinusoid, to get a feeling for how the wavelet transform is representing these.

The End