Introduction to Spectral Analysis

In this assignment, we will look at the basics of spectral analysis. As complex-valued or bivariate data is quite common in the earth sciences, we will work with horizontal velocity data from an oceanographic current meter.

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.)

Many thanks to Jan-Adrian Kallmyr for helping with translating the Matlab tutorial into Python.

In [1]:

```
import warnings
warnings.filterwarnings('ignore')  #suppress some warnings about future code changes

import netCDF4 as nc
import xarray as xr
import numpy as np
import datetime          #https://docs.python.org/3/library/datetime.html
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.signal as sg  #Package for signal analysis
import scipy.ndimage as si #Another package for signal analysis
#from scipy.interpolate import interp1d  #for converting cell to grid-centered coordin
#from scipy import stats    #Used for 2D binned statistics
#from mpl_toolkits.axes_grid1 import make_axes_locatable #For plotting interior coloba
#import cartopy.crs as ccrs
import spectrum

#from scipy import signal, ndimage
from scipy import fft as spfft
from scipy.fft import fft
from scipy.stats import chi2
from scipy.special import digamma
#function for coverting Matlab's datenum into Python's datetime
#from https://gist.github.com/victorkristof/b9d794fe1ed12e708b9d
#with modifications to support array input and output
def datenum_to_datetime(datenum):
    """
    Convert Matlab datenum into Python datetime.

    Args:
        datenum: Date in datenum format

    Returns:
        Datetime object corresponding to datenum.
    """
    date=[]
    for day in datenum:
        date.append(datetime.datetime.fromordinal(int(day)) \
           + datetime.timedelta(days=day%1) \
           - datetime.timedelta(days=366))
    date=np.array(date)
    return date

plt.rcParams["figure.figsize"] = (10,6.5) #set default figure size
plt.rcParams['font.size'] = 14
plt.rcParams['font.family'] = 'serif'
```

Let's take a quick look at the dataset.

In [2]:
```
datadir = "../data/"  #choose this appropriate for your system
filename = "ml244.nc"
ds = xr.open_dataset(datadir+filename) #Load the dataset
ds = xr.Dataset.transpose(ds)  #Enforce the convection that time is in rows
print(ds)
```

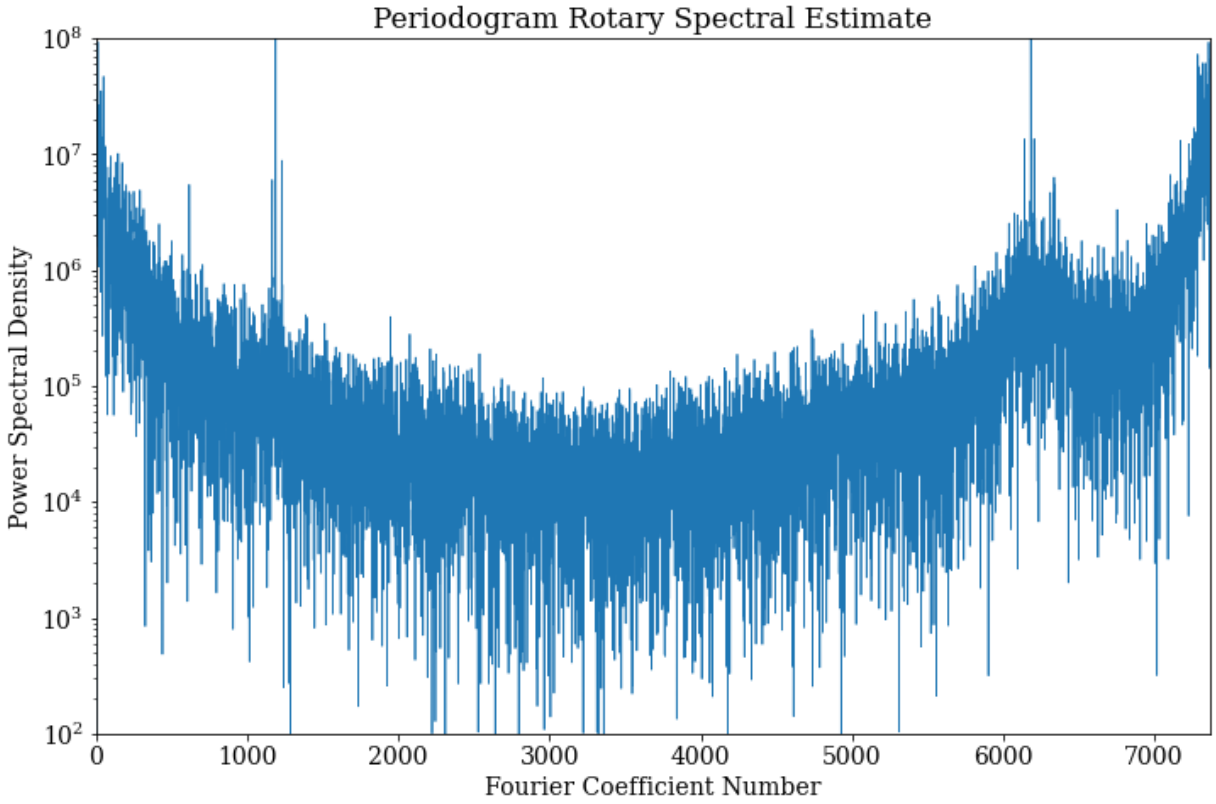
```
<xarray.Dataset>
Dimensions:  (time: 7371, depth: 4)
Dimensions without coordinates: time, depth
Data variables:
    lat      float64 ...
    lon      float64 ...
    num      (time) float64 ...
    depths   (depth) float64 ...
    u        (time, depth) float64 ...
    v        (time, depth) float64 ...
    t        (time, depth) float64 ...
    p        (time, depth) float64 ...
```

The Periodogram, a.k.a. the Naive Spectral Estimate

The first point to know is that the spectrum is not something that can be computed. Instead, it is estimated. The true spectrum is a theoretical object that can't be computed unless you have an infinite amount of time and perfect sampling. The things that can be computed are called spectral estimates or estimated spectra.

Firstly, we will look at the modulus-squared Fourier transform. It is common for this to be referred to as 'the spectrum'. However, this terminology is incorrect and misleading. Instead, the modulus-squared Fourier transform is a type of spectral estimate called the *periodogram*. It is known, actually, to be a very poor spectral estimate for reasons we will learn about in the course notes.

In [3]:
```
#date = datenum_to_datetime(ds["num"].date)     #date as a datetime object
cv = ds['u'][:,3].data +1j*ds['v'][:,3].data   #complex-valued velocity at deepest de

fig, ax = plt.subplots(1, 1)
ax.semilogy(np.abs(fft(cv-np.mean(cv)))**2,linewidth=0.75)
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e2, 1e8)
fig.tight_layout()

plt.xlabel('Fourier Coefficient Number')
plt.ylabel('Power Spectral Density')
plt.title('Periodogram Rotary Spectral Estimate');
```



The x-axis is simply the index number of the terms in the squared discrete Fourier transform.

Here we have set the y-axis to be logarithmic. This is often useful in dealing with spectra. Note also that we have removed the mean prior to taking the fft, which minimizes broadband bias from the zero frequency.

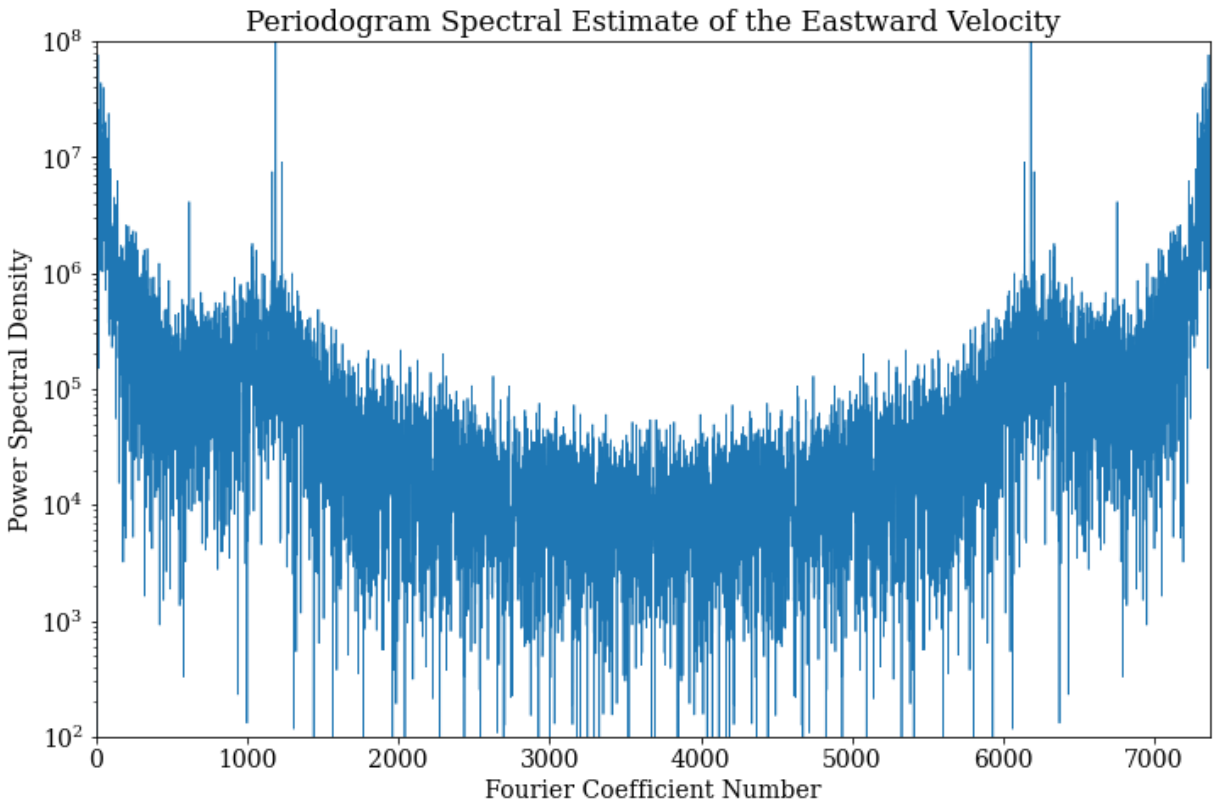
In the array output by fft, positive frequencies (circles rotating in a counterclockwise sense) are on the left, while negative frequencies (circles rotating in a clockwise sense) are on the right, appearing in reverse order as we have discussed in class. Thus the highest resolvable frequency, called the Nyquist frequency, is in the middle.

You can see clearly that the periodogram roughly has a certain symmetry if you reflect it about the middle. However, it is not completely symmetric because, with a complex-valued signal such as velocity, the twin frequencies (positive and negative rotations at the same absolute frequency) are not required to cancel.

We will discuss the nature of the features we see here shortly. For the moment, we compare this with the periodogram of just the eastward velocity, the real part of our complex-valued velocity time series:

In [4]:

```
fig, ax = plt.subplots(1, 1)
ax.semilogy(np.abs(fft(np.real(cv-np.mean(cv))))**2,linewidth=0.75)
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e2, 1e8)
fig.tight_layout()

plt.xlabel('Fourier Coefficient Number')
plt.ylabel('Power Spectral Density')
plt.title('Periodogram Spectral Estimate of the Eastward Velocity');
```



Now, the periodogram is perfectly symmetric about the center or Nyquist frequency. As discussed in the course notes, twin frequencies occur in conjugate pairs, with the same magnitudes but reversed phase. Because of this, two complex exponentials rotating in opposite directions cancel to yield a real-valued, phase-shifted sinuosoid. The symmetry we see in the periodogram is a reflection of the fact that the time series is real-valued.

Adding two oppositely-rotating circles in general leads to an ellipse. When the magnitude of one circle vanishes the result is a circle. When the magnitude of both circles are identical, the result is a line, and this is the case for a real-valued time series.

One-Sided Spectra

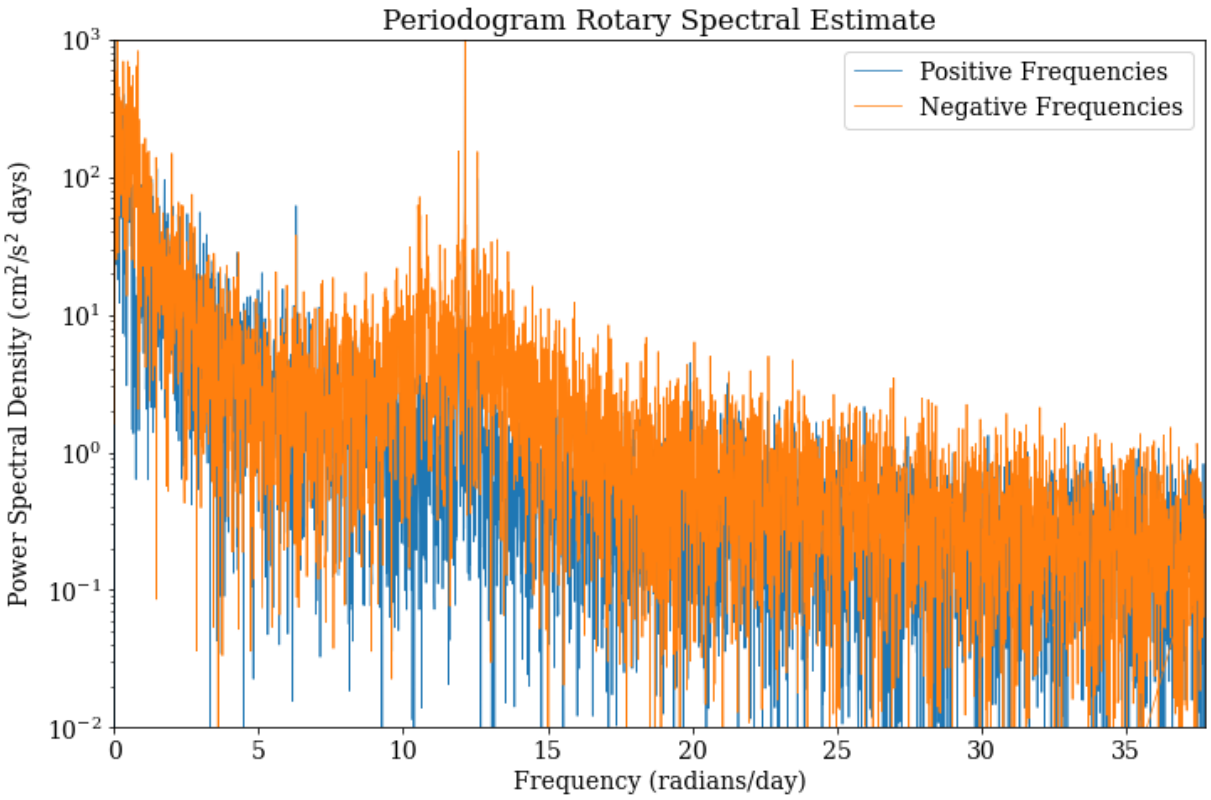
There are several inconvenient aspects of presenting the periodogram in this way. Firstly, it is a little bit of a hassle to figure out where the frequencies are. Secondly, it is a bit odd to see the positive and negative frequencies meet in the middle at the Nyquist. Thirdly, the y-axis should be normalized in a meaningful way.

We'll make a new plot that addresses these issues.

In [5]:

```
dt = ds["num"][1].data - ds["num"][0].data #sampling interval in days

f, S = sg.periodogram(cv-np.mean(cv), fs=1/dt) #fs = sampling frequency (cyclic)
omega = f*2*np.pi #convert cyclic frequency to radian frequency

fig, ax = plt.subplots(1, 1)
ax.semilogy( omega[np.where(omega>=0)],S[np.where(omega>=0)],linewidth=0.75)#plot pos
ax.semilogy(-omega[np.where(omega<=0)],S[np.where(omega<=0)],linewidth=0.75)#plot neg
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-2, 1e3)
fig.tight_layout()
```

```
ax.legend(['Positive Frequencies','Negative Frequencies'])
plt.xlabel('Frequency (radians/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Periodogram Rotary Spectral Estimate');
```



Here the two-sided periodgram has been split into two one-sided portions, called the positive and negative rotary spectra, each containing roughly half as many data points as the original time series. Their structure is still hard to see at the moment; we will get to that later.

We have also normalized the y-axis in a sensible way. The spectrum integrates to the variance, so we would like our spectral estimate to do so, too. For the two-sided periodgram with an odd number of points, using radian frequency, the correct formula to recover the variance is

In [6]:
```
(1/2/np.pi)*(omega[1]-omega[0])*np.sum(S)  #value of the summed spectrum
```

Out[6]:
```
94.5525803225848
```

where the omega[1]-omega[0] is a differential, as would appear in an intergal. When we compare this to the directly computed variance, we find

In [7]:
```
np.std(cv)**2
```

Out[7]:
```
94.55258032258482
```

verifying that our spectral estimate is correctly normalized to obtain the variance as computed in the time domain. These numbers imply a standard deviation of $\sigma=\sqrt{94.5}=9.7$ cm/s.

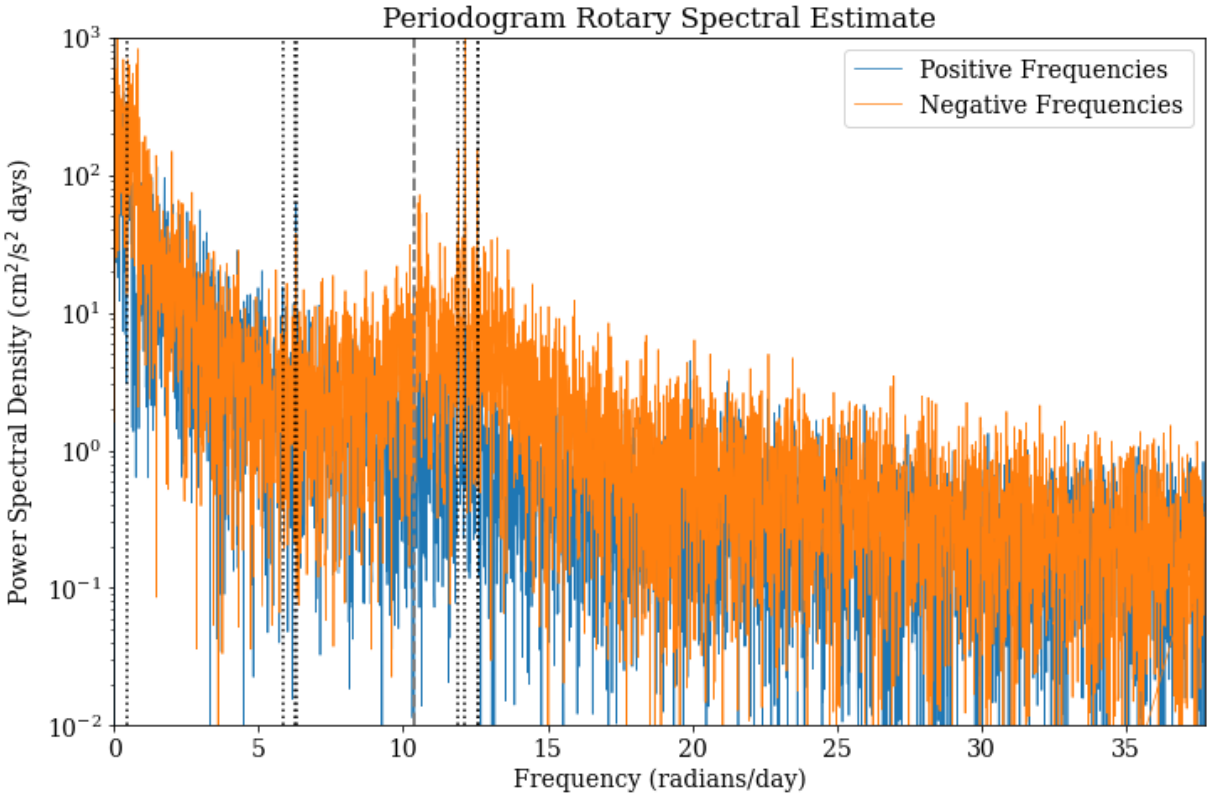
This explains the units of the spectrum. Its units must the square of whatever the units are of your time series, in this case cm/s, divided by the frequency units, in this case 1/days (recall that radians are dimensionless). This is because the spectrum is a partitioning of variance across frequencies.

Radian and Cyclic Frequencies

Next we put down some markers of some meaningful frequencies: the Coriolis frequency at the latitude of the mooring location---around which the oceanic internal wave field is concentrated---and eight prominent tidal frequencies.

In [8]:

```
#define functions to return the Coriolis and tidal frequencies
def corfreq(lat):
    """
    The Coriolis frequency in rad / day at a given latitude.

    Args:
        lat: Latitude in degree

    Returns:
        The Coriolis frequecy at latitude lat
    """
    omega=7.2921159e-5;
    return 2*np.sin(lat*2*np.pi/360)*omega*(3600)*24;

def tidefreq():
    """
    Eight major tidal frequencies in rad / day.  See Gill (1982) page 335.

    Args:
        None

    Returns:
        An array of eight frequencies
    """
    return 24*2*np.pi/np.array([327.85,25.8194,24.0659,23.9344,12.6584,12.4206,12.00


fig, ax = plt.subplots(1, 1)
ax.semilogy( omega[np.where(omega>=0)],S[np.where(omega>=0)],linewidth=0.75)#plot pos
ax.semilogy(-omega[np.where(omega<=0)],S[np.where(omega<=0)],linewidth=0.75)#plot neg
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-2, 1e3)
fig.tight_layout()

ax.vlines(tidefreq(),ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color="black")
ax.vlines(corfreq(ds["lat"].data),ax.get_ylim()[0],ax.get_ylim()[1],linestyle="--", c
ax.legend(['Positive Frequencies','Negative Frequencies'])

plt.xlabel('Frequency (radians/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Periodogram Rotary Spectral Estimate');
```



The tidal frequencies appear in three groups. From right to left, there are four semidiurnal tidal frequencies, three diurnal frequencies, and one low-frequency tide, the Mf lunal fortnightly tide at about one cycle per 13.6 days.

As we have discussed, when a frequency is expressed in units of radians per unit time, it is called a radian or angular frequency. When a frequency is expressed in units of cycles per unit time, it is called a cyclic frequency. This is more easily seen in how one writes sinusoid or a complex exponential:

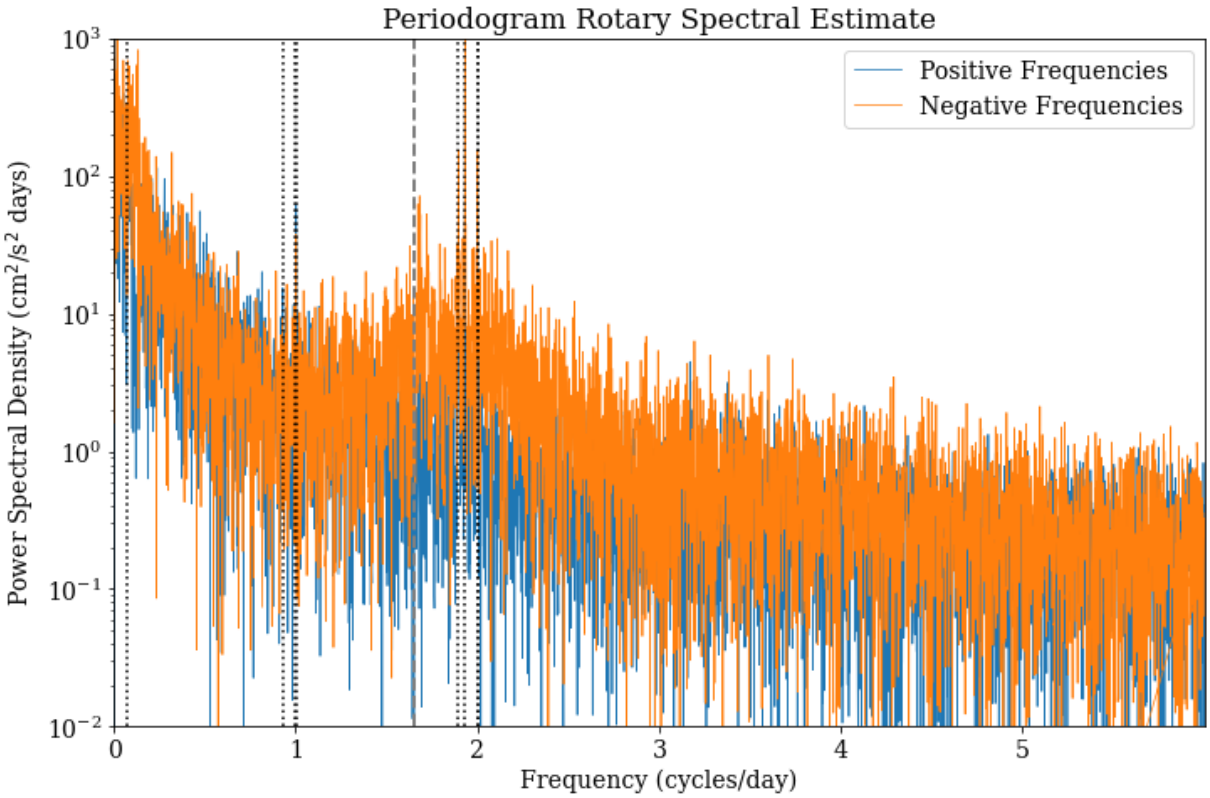
\begin{equation} e^{i \omega t}~~~(\mathrm{radian}) \quad\quad \mathrm{vs.}\quad\quad e^{2\pi i f t}~~~(\mathrm{cyclic}) \end{equation}
Thus the relationship between the radian frequency ω and the cyclic frequency f is $f=\omega/2\pi$.

I find it useful to work with both types of frequencies. Radian frequencies are convenient for theoretical expressions. However, cyclic frequencies are more intutive and therefore to be preferred when plotting or quoting values.

So now we redo the above plot in cyclic frequency, and covert from radians per day to cycles per day.

In [9]:
```
f, S = sg.periodogram(cv-np.mean(cv), fs=1/dt) #fs = sampling frequency (cyclic)

fig, ax = plt.subplots(1, 1)
ax.semilogy( f[np.where(f>=0)],S[np.where(f>=0)],linewidth=0.75)#plot positive side
ax.semilogy(-f[np.where(f<=0)],S[np.where(f<=0)],linewidth=0.75)#plot negative side
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-2, 1e3)
fig.tight_layout()

ax.vlines(tidefreq()/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color='
ax.vlines(corfreq(ds["lat"].data)/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle
```

```
plt.legend(['Positive Frequencies','Negative Frequencies'])
plt.xlabel('Frequency (cycles/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Periodogram Rotary Spectral Estimate');
```



We see that the group of semidiurnal tides occur at 2 cycles per day, as expected.

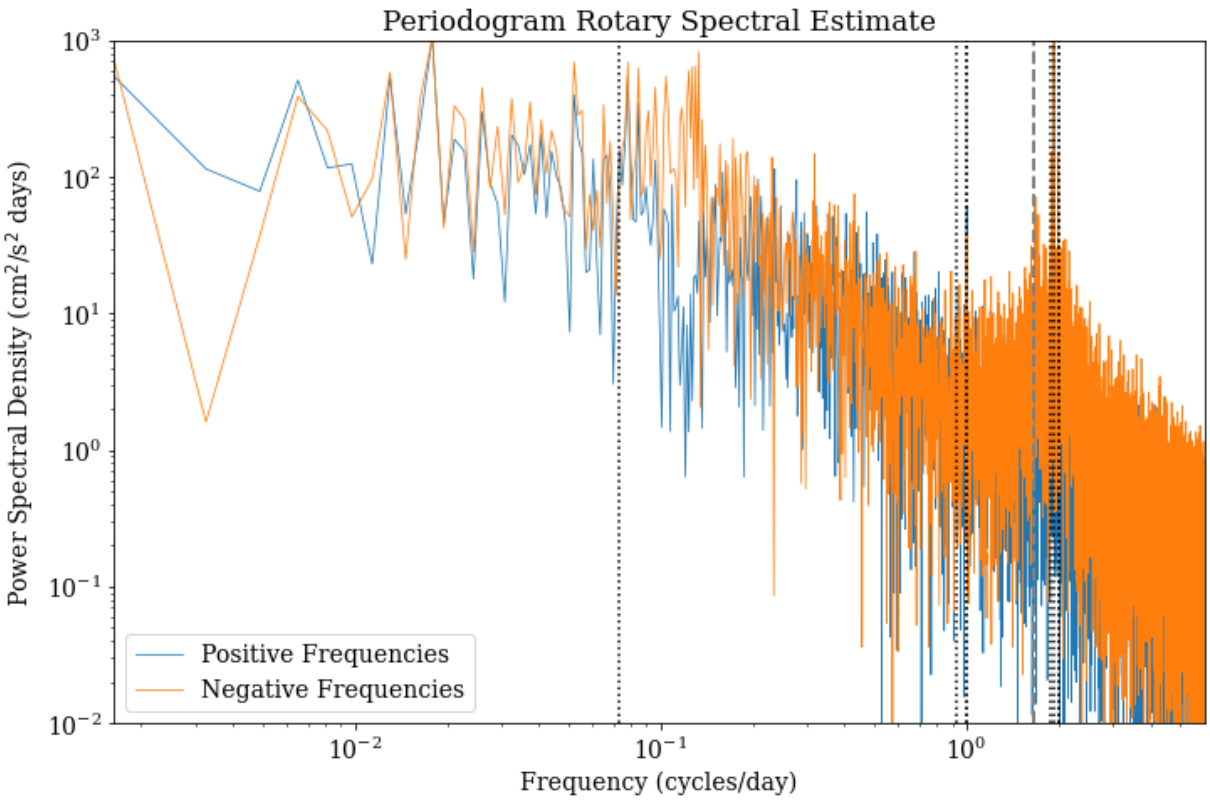
Note that changing to cycles per day has also changed the y-level of the spectral estimate (by absorbing a factor of $1/(2\pi)$), such that the spectral estimate continues to satisfy the normalization discussed earlier:

In [10]:
```
(f[1]-f[0])*np.sum(S)  #value of the summed spectrum
```

Out[10]:
```
94.5525803225848
```

Finally, it is often convenient to look at spectra with log/log axes:

In [11]:
```
fig, ax = plt.subplots(1, 1)
ax.loglog( f[np.where(f>0)],S[np.where(f>0)],linewidth=0.75)#plot positive side
ax.loglog(-f[np.where(f<0)],S[np.where(f<0)],linewidth=0.75)#plot negative side
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-2, 1e3)
fig.tight_layout()

ax.vlines(tidefreq()/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color='
ax.vlines(corfreq(ds["lat"].data)/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle

ax.legend(['Positive Frequencies','Negative Frequencies'])
plt.xlabel('Frequency (cycles/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Periodogram Rotary Spectral Estimate');
```



Again we see the tidal frequencies appearing in three groups. Here, we can see that there are many more points at higher frequencies than at lower frequencies on account of the change to a logarithmic x-axis. The lowest nonzero frequency, the leftmost frequency appearing in this log-log plot, is an important quantity called the Rayleigh frequency that we have been discussing. The Rayleigh frequency in cyclic units is simply one over the length of the time series, and is equal to one cycle per 614 days or 0.0016 cycles per day for this time series.

(Note that as an alternative to log/log axes, one sometimes encounters in oceanography something called a "variance preserving spectrum". This is a bad idea so we won't use it.)

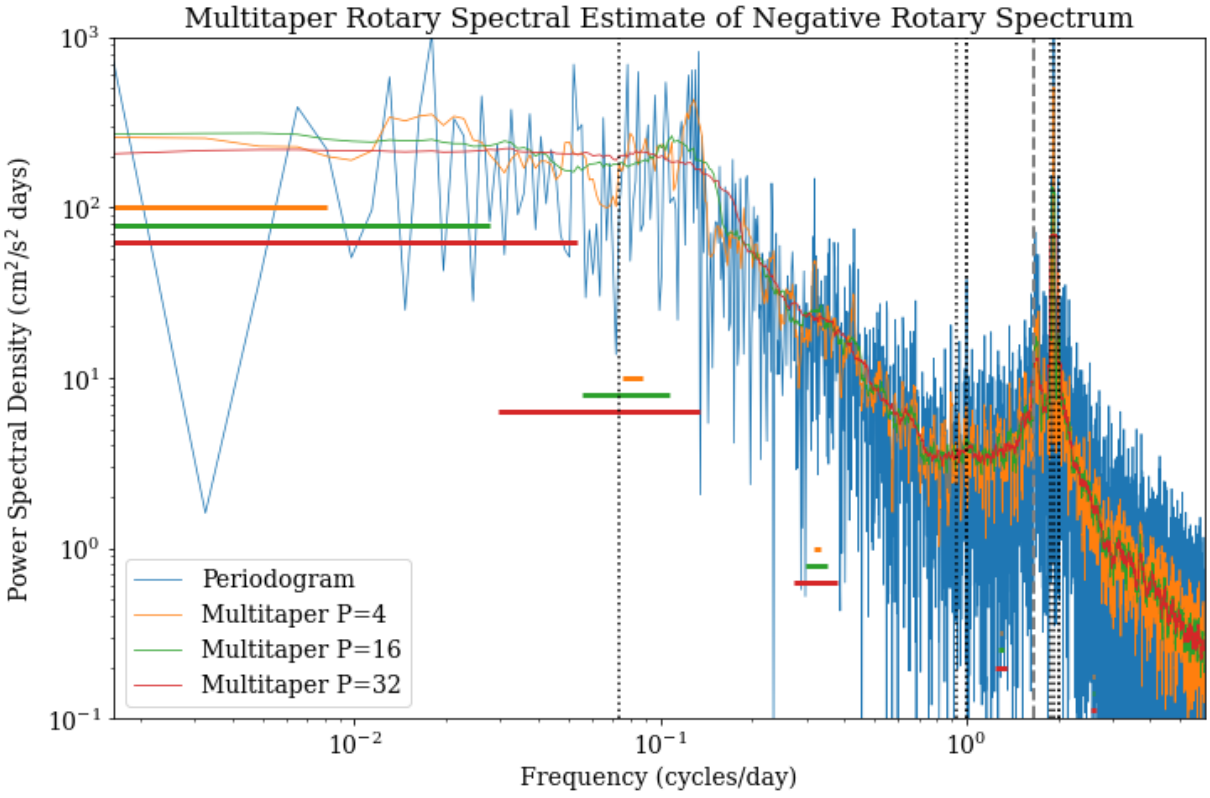
The Multitaper Method

The periodogram is now presented with meaningful units and on a meaningful frequency axis. However, we see that it has a very 'fuzzy' appearance on account of a high degree of variability. This high variance of the spectral estimate is a major, but not the only, problem with the periodogram. To remove this variance, we need to do some type of smoothing or averaging. The most convenient way to accomplish this is implicitly, through the use of a type of spectral estimate termed the multitaper method.

For the moment we will plot only the negative or clockwise side, and will try three different degrees of smoothing that we compare with the raw periodogram estimate, which involves no smoothing.

In [12]:
```
P = (4,16,32)  #define an array of different smoothing lengths

S=np.zeros((np.size(cv),4))
f, S[:,0] = sg.periodogram(cv-np.mean(cv), fs=1/dt) #fs = sampling frequency (cyclic)

for i in range(len(P)):
```

```
    psi, eigs = spectrum.mtm.dpss(np.size(cv), NW=P[i], k=2*P[i]-1)
    Zk, weights, eigenvalues = spectrum.mtm.pmtm(cv-np.mean(cv), k=2*P[i]-1,  NFFT=
    S[:,1+i]=np.mean(np.transpose(np.abs(Zk)**2), axis=1) * dt
    #need to multiply by dt because pmtm doesn't know about sample rate

fig, ax = plt.subplots(1, 1)

linecolors=[]
for n in range(np.size(S,1)):
    ax.loglog(-f[np.where(f<0)],np.transpose(S[np.where(f<0),n]),linewidth=0.75)#plo
    if n>0:
        linecolors.append(plt.gca().lines[-1].get_color())  #remember the line colors

ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-1, 1e3)
fig.tight_layout()

#show smoothing extents at different frequencies
io=(1,50,200,800,1600)
yo=(10**2,10**1,10**0,10**(-0.5),10**(-0.75))
for i in range(np.size(io)):
    for n in range(len(P)):
        if i==0:
            plt.hlines(yo[i]*10**(-n/10),f[io[i]],f[io[i]+P[n]],linewidth=3,color=li
        else:
            plt.hlines(yo[i]*10**(-n/10),f[io[i]-P[n]],f[io[i]+P[n]],linewidth=3,col

ax.vlines(tidefreq()/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color='
ax.vlines(corfreq(ds["lat"].data)/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle

ax.legend(['Periodogram','Multitaper P=4','Multitaper P=16','Multitaper P=32'],loc='l
plt.xlabel('Frequency (cycles/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Multitaper Rotary Spectral Estimate of Negative Rotary Spectrum');
```



The quantity P in the code is called the time-bandwidth product, and it sets the degree of smoothing across frequencies, with a larger P value leading to smoother spectra.

The horizontal lines represent the effective smoothing that has been applied by the multitaper method, which has the effect of smoothing adjacent frequencies in a window that has a width of $2P-1$ Fourier coefficients. For example, with $P=16$ we are essentially smoothing over 31 adjacent frequencies. The black lines shows the *half-width* of the smoother $(2P-1)/2\approx P$ in the vicinity of the zero frequency, and the full width of $2P-1$ frequencies elsewhere.

The fact that the apparent width of the smoothing changes is due to the fact that we are presenting this plot with a logarithimic x-axis. This explains why the spectra appear to become more rough as one moves toward higher frequencies.

In practical terms you simply increases the time-bandwidth product until you feel you have obtained a suitable degree of smoothing. Deciding on an appropriate degree of smoothing is generally something that is done by 'feel', depending on what features you are interested in investigating, rather than something that has an objectively correct answer.

When the smoothing is too small, the spectral estimate is poor because of too much variance. When the smoothing is too large, the spectral estimate is poor not because it varies, but because it is systematically different from the true spectrum; this problem is called bias. These two problems trade off against each other. One of the challenges in signal analysis is trying to make a sensible choice in this bias-variance tradeoff.

Rotary Spectra

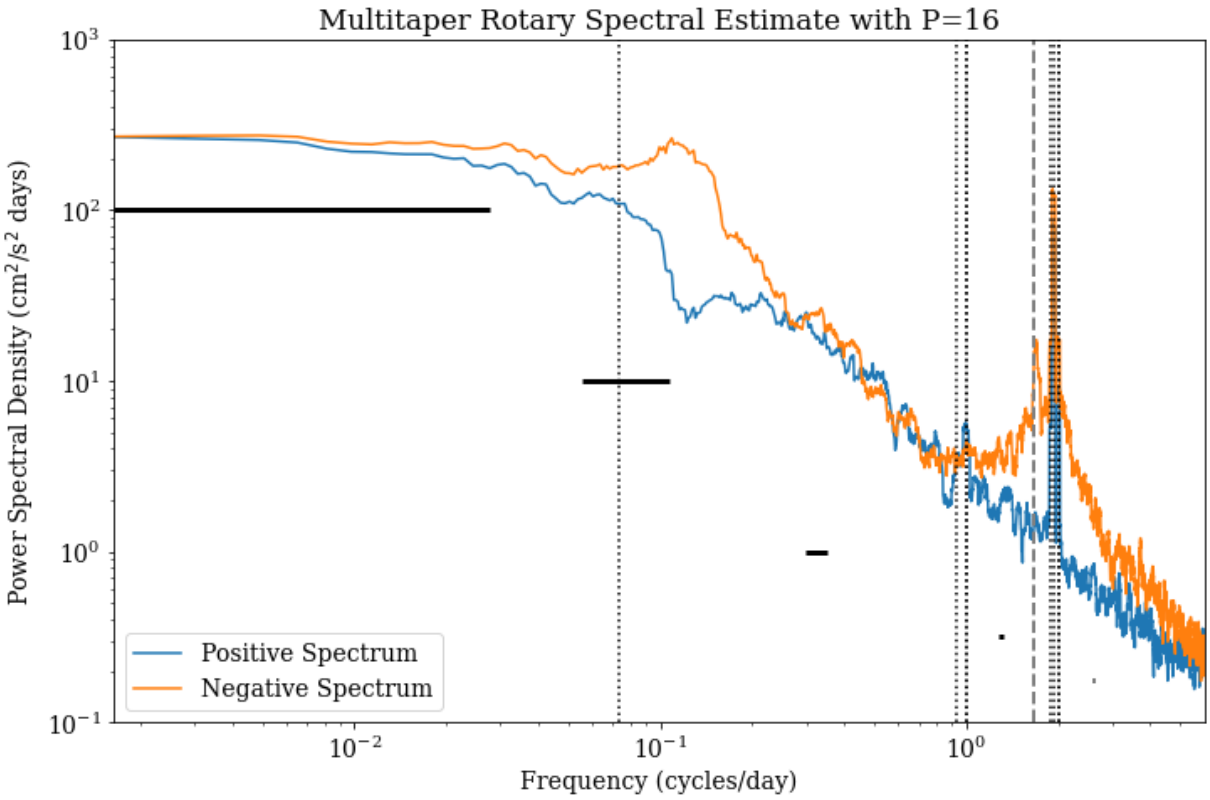
We will now compare the positive and negative rotary spectra using the $P=16$ multitaper estimate.

In [13]:

```
P = 16
psi, eigs = spectrum.mtm.dpss(np.size(cv), NW=P, k=2*P-1)

Zk, weights, eigenvalues = spectrum.mtm.pmtm(cv-np.mean(cv), k=2*P-1,  NFFT=np.size(
S=np.mean(np.transpose(np.abs(Zk)**2), axis=1) * dt
#need to multiply by dt because pmtm doesn't know about sample rate
print((f[1]-f[0])*np.sum(S)) #verify variance is approximately recovered

#to compute adaptive version just FYI
#Zk, weights, eigenvalues = spectrum.mtm.pmtm(cv-np.mean(cv), k=2*P-1,  NFFT=np.size(c
#S=np.mean(np.transpose(np.abs(Zk)**2) * weights, axis=1) * dt

fig, ax = plt.subplots(1, 1)
ax.loglog( f[np.where(f>0)],S[np.where(f>0)])#plot positive side
ax.loglog(-f[np.where(f<0)],S[np.where(f<0)])#plot negative side
ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-1, 1e3)
fig.tight_layout()

#show smoothing extents at different frequencies
io=(1,50,200,800,1600)
yo=(10**2,10**1,10**0,10**(-0.5),10**(-0.75))
for i in range(np.size(io)):
    if i==0:
        plt.hlines(yo[i],f[io[i]],f[io[i]+P],linewidth=3,color='k')
    else:
        plt.hlines(yo[i],f[io[i]-P],f[io[i]+P],linewidth=3,color='k')
```

```
ax.vlines(tidefreq()/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color='
ax.vlines(corfreq(ds["lat"].data)/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle

ax.legend(['Positive Spectrum','Negative Spectrum'],loc='lower left')
plt.xlabel('Frequency (cycles/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Multitaper Rotary Spectral Estimate with P=16');
```

```
94.26238970843409
```



Note the substantial differences between the two sides. On the negative side---the side that supports inertial oscillations in the Northern Hemisphere---we see a broad wave band surrounding the semidiurnal tide, which at this latitude is close to the inertial frequency. These are internal waves clustered around the inertial frequency. On the positive side, there is a very narrow semidiurnal peak and no broadband peak at all.

The reason that the orange peak is so narrow is that free waves are not supported on this side. (We know this from our physical understanding of the internal wave dispersion relation for near-inertial waves.) This leads us to hypothesize that the narrow peak on the positive side is the barotropic tide, not the so-called baroclinic tide associated with excited internal waves. (This could be investigated more by looking at the vertical shear in this frequency band on the two sides of the spectrum.)

Moving towards lower or subinertial frequencies, the positive and negative spectra quickly converge, starting around say one cycle per day. Here we see that there is background slope that runs continuously through most of the spectrum, with the negatively rotating peak rising above this slope. Loosely speaking, the slope represents geostrophic turbulence and boundary current variability, while the peak broadband represents inertia-gravity waves.

At still lower frequencies, at say one cycle per ten days, a possible peak is observed in the negatively rotating portion. We will check in a moment whether that peak is significant.

Below that, both sides of the spectrum transitions to a constant value. This transition appears to occur *before* the smoothing width coming from the lowest frequencies is encountered, and therefore appears physically meaningful, i.e. is does not appear to be an artifact of smoothing out the lowest frequencies. This interpretation is supported by the previous plot, where the low-frequency plateau is still visible using the $P=4$ setting.

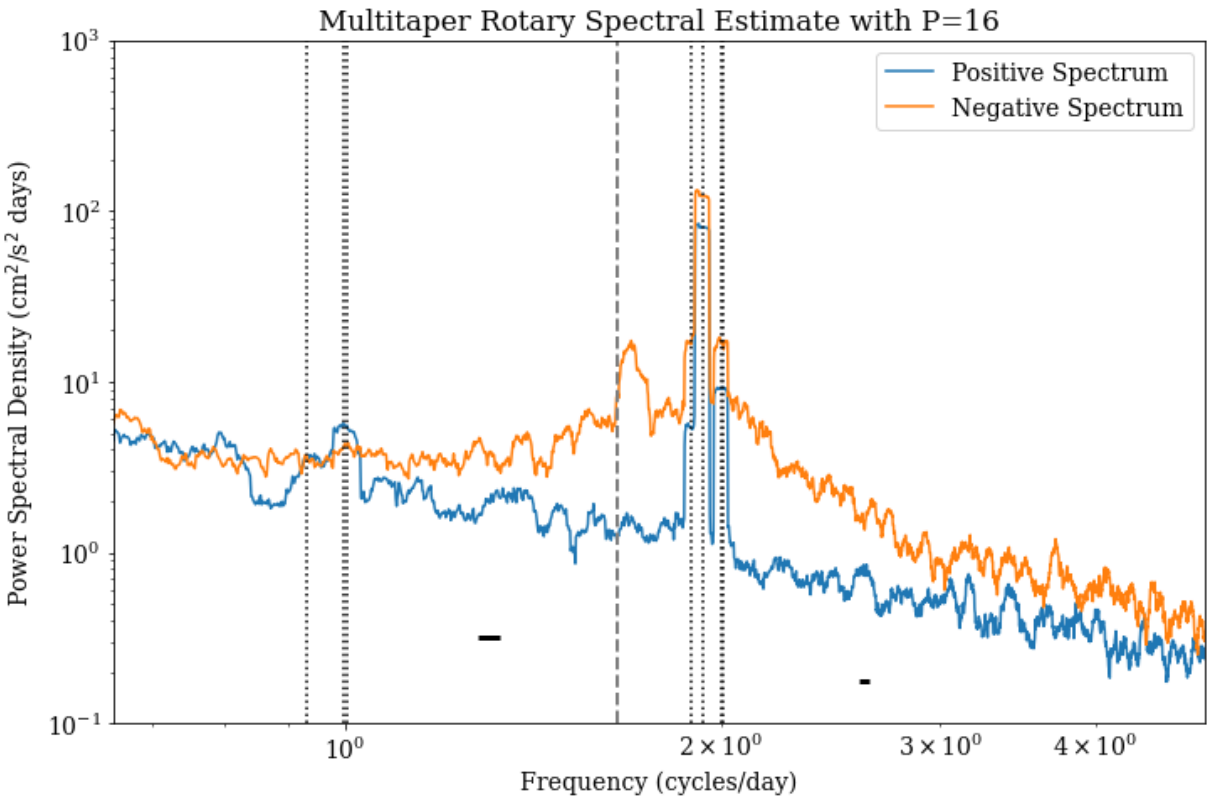
Moving to higher frequencies from the inertial peak, the negative spectrum becomes closer to, but remains stronger than, the positive spectrum up to the highest resolvable frequency. This behavior likely reflects the known polarization behavior for internal waves.

Given the inertial frequency f_o and the buoyancy frequency N, the horizontal currents due to internal waves form an ellipse, the eccentricity of which is known as a function of frequency. This ellipse becomes a circle at f_o where the internal waves become pure inertial oscillations, and a straight line at N, where the waves become pure internal gravity waves. Thus, what we are seeing here is the internal wave spectrum becoming more linearly polarized and less circularly polarized---more like gravity waves and less like inertial oscillations---as we head towards the buoyancy frequency.

Let's take a closer look at the near-inertial peak by zooming in. We'll remake the same plot with a tighter x-axis.

In [14]:

```
fig, ax = plt.subplots(1, 1)
ax.loglog( f[np.where(f>0)],S[np.where(f>0)])#plot positive side
ax.loglog(-f[np.where(f<0)],S[np.where(f<0)])#plot negative side
ax.autoscale(enable=True, tight=True)
ax.set_xlim(f[400], f[3000])   # <------------------------------ Tighten x-axis
ax.set_ylim(1e-1, 1e3)
fig.tight_layout()

#show smoothing extents at different frequencies
for i in range(np.size(io)):
    if i==0:
        plt.hlines(yo[i],f[io[i]],f[io[i]+P],linewidth=3,color='k')
    else:
        plt.hlines(yo[i],f[io[i]-P],f[io[i]+P],linewidth=3,color='k')

ax.vlines(tidefreq()/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color='
ax.vlines(corfreq(ds["lat"].data)/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle

ax.legend(['Positive Spectrum','Negative Spectrum'],loc='upper right')
plt.xlabel('Frequency (cycles/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Multitaper Rotary Spectral Estimate with P=16');
```



Here we can see that there are actually individual lines for the three different semidiurnal tidal components. These have significant magitude on both the positive and negatives sides of the specturm, meaning that these oscillations would be elliptically and not circularly polarized. We could study these tidal oscillations more using a narrowband filter if we wished. A smaller peak at the some of the diurnal tidal lines is also apparent, but only in the positively rotating component.

Now let's take a look at the spectral estimates with confidence intervals superposed. Here, we will plot the negative and positive rotary spectral estimates separately.

In [15]:

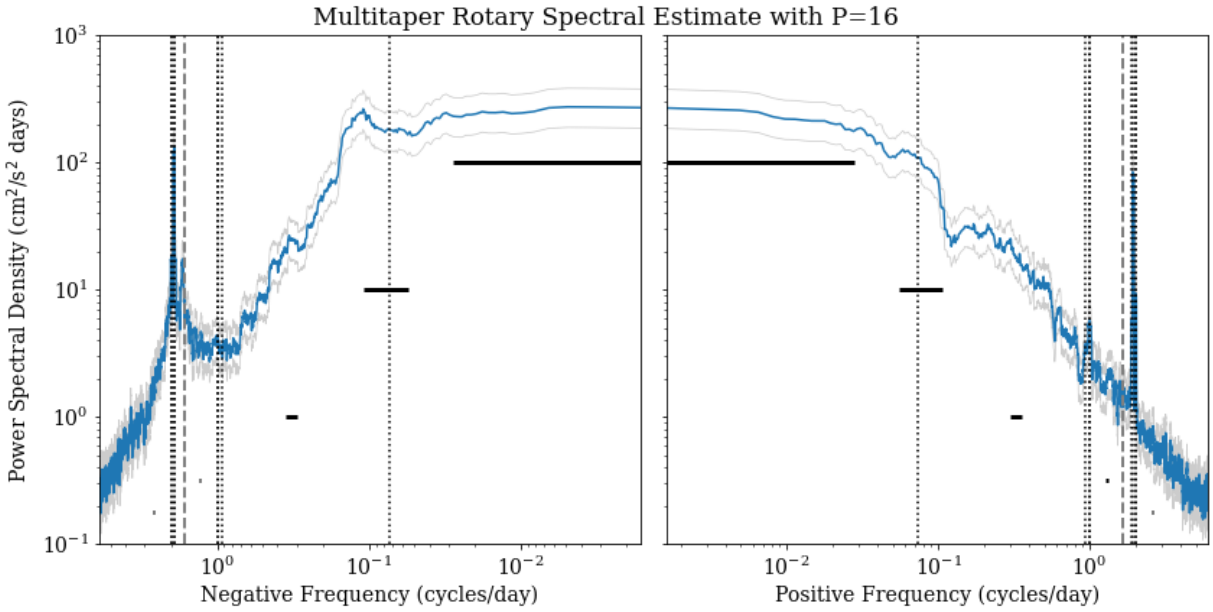
```
#define a function for computing confidence intervals.

def mconf(K,gamma,str):
    """
    Compute symmetric confidence intervals for multitaper spectral estimation.

    Args:
        K: Number of tapers used in the spectral estimate, normally 2*P-1
        gamma: confidence level, e.g., 0.95
        str: 'lin' to return confidence intervals for linear axis
             'log' to return confidence intervals for log10 axis

    Returns:
        ra,rb:  Ratio factors for confidence interval

    If S0 is the true value of the spectrum and S is the spectral estimate,
    then the confindence interval is defined such that

        Probability that ra < S/S0 < ra = gamma     (linear case)
        Probability that ra < log10(S)/log10(S0) < ra = gamma (log10 case)

    The confidence interval will be S*ra to S*rb for the spectral values
    in linear space, or S*10^ra to S*10^rb in log10 space.  If log10(S)
    is plotted rather than S with a logarithmic axis, the latter would
    become log10(S)+ra to log10(S)*rb.
```

```
    """

    dx=0.0001
    #Compute pdf symmetrically about unity

    if str=='lin':

        x1=np.arange(1-dx/2,-1,step=-dx)*2*K  #from one to minus one
        x2=np.arange(1+dx/2,3,step=dx)*2*K #from 1 to three
        fx=(chi2.pdf(x1,2*K)+chi2.pdf(x2,2*K))*2*K
        sumfx=np.cumsum(fx)*dx

        ii=np.where(sumfx>=gamma)[0][0]
        ra=x1[ii]/2/K
        rb=x1[ii]/2/K

    elif str=='log':

        xo=np.log(2)+np.log(1/2/K)+digamma(K) #see pav15-arxiv
        c=np.log(10)
        xo=xo/c  #change of base rule

        x1=np.power(10,np.arange(xo-dx/2,xo-2,step=-dx))
        x2=np.power(10,np.arange(xo+dx/2,xo+2,step=dx))

        fx1=c*2*K*np.multiply(x1,chi2.pdf(2*K*x1,2*K))
        fx2=c*2*K*np.multiply(x2,chi2.pdf(2*K*x2,2*K))

        sumfx=np.cumsum(fx1+fx2)*dx

        ii=np.where(sumfx>=gamma)[0][0]
        ra=np.log10(x1[ii])
        rb=np.log10(x2[ii])

    return ra,rb
```

In [16]:

```
P = 16
psi, eigs = spectrum.mtm.dpss(np.size(cv), NW=P, k=2*P-1)
ra,rb = mconf(2*P-1,0.95,'log') #compute confidence intervals

Zk, weights, eigenvalues = spectrum.mtm.pmtm(cv-np.mean(cv), k=2*P-1,  NFFT=np.size(
S=np.mean(np.transpose(np.abs(Zk)**2), axis=1) * dt

fig, ax = plt.subplots(1, 2,sharey=True,figsize=(12,6))
ax[0].loglog(-f[np.where(f<0)],(10**ra)*S[np.where(f<0)],color=(0.8,0.8,0.8),linewid
ax[0].loglog(-f[np.where(f<0)],(10**rb)*S[np.where(f<0)],color=(0.8,0.8,0.8),linewid
ax[0].loglog(-f[np.where(f<0)],S[np.where(f<0)])#plot negative side
ax[0].invert_xaxis()
ax[0].set_ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
ax[0].set_xlabel('Negative Frequency (cycles/day)')

ax[1].loglog(f[np.where(f>0)],(10**ra)*S[np.where(f>0)],color=(0.8,0.8,0.8),linewidtl
ax[1].loglog(f[np.where(f>0)],(10**rb)*S[np.where(f>0)],color=(0.8,0.8,0.8),linewidtl
ax[1].loglog( f[np.where(f>0)],S[np.where(f>0)])#plot positive side
ax[1].autoscale(enable=True, tight=True)
ax[1].set_ylim(1e-1, 1e3)
ax[1].set_xlabel('Positive Frequency (cycles/day)')

for n in range(np.size(ax)):
    ax[n].autoscale(enable=True, tight=True)
    ax[n].vlines(tidefreq()/2/np.pi,ax[1].get_ylim()[0],ax[1].get_ylim()[1],linestyle
    ax[n].vlines(corfreq(ds["lat"].data)/2/np.pi,ax[1].get_ylim()[0],ax[1].get_ylim()
```

```
    io=(1,50,200,800,1600)
    yo=(10**2,10**1,10**0,10**(-0.5),10**(-0.75))

    plt.sca(ax[n])
    for i in range(np.size(io)):
        if i==0:
            plt.hlines(yo[i],f[io[i]],f[io[i]+P],linewidth=3,color='k')
        else:
            plt.hlines(yo[i],f[io[i]-P],f[io[i]+P],linewidth=3,color='k')

fig.tight_layout()
fig.suptitle('Multitaper Rotary Spectral Estimate with P=16',y=0.995);
```



Based on our understanding of the statistical properities of the multitaper spectral estimate, the true spectrum should lie within the gray lines 95% probability. This means the near-inertial peak is highly statistically sigificant, whereas the low-frequency peak on the negative side is just barely signficant at the 95% level. Most of the other peaks and valleys we see along the sloping portion of the spectra are within the expected envelope of random fluctuations and are therefore not significant.

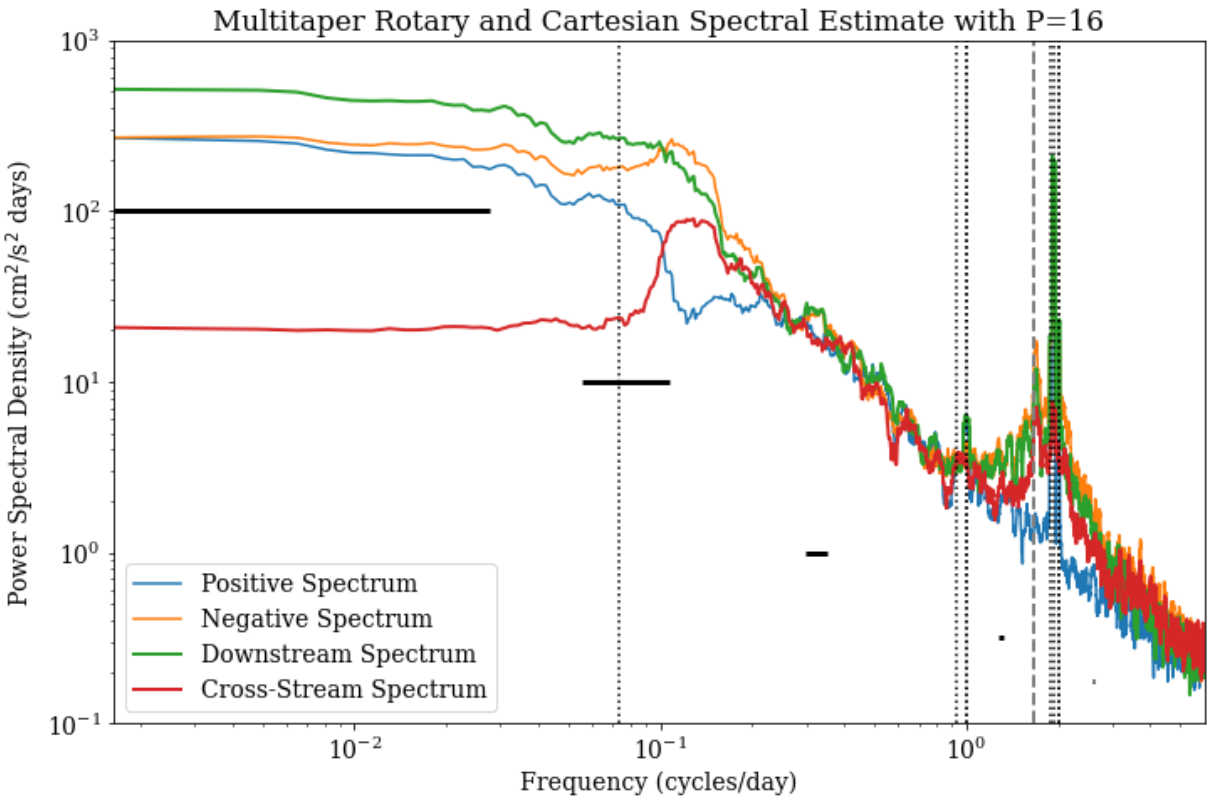
Cartesian Spectra

Next we will compare the rotary spectra with the Cartesian spectra. We will learn more about this in class, however, the basic idea is that we have a choice as to how to split up the spectrum of our complex-valued velocity signal $z=u+iv$. We can choose to look at the rotary spectrum, separating positive and negative frequencies, or alternatively, we can choose to look at the spectra of u and v considered separately. We refer to the resulting pair of one-sides spectra as the Cartesian spectra, to distinguish these from the rotary spectra.

Here we will look not at the Cartesian spectra in the usual eastward / northward coordinate system, but rather in a frame aligned with the mean flow. We do this by defining a rotated time series $\tilde z =\tilde u+ i\tilde v=z e^{-i\phi}$ where ϕ is the direction of the mean flow.

In [17]:
```
phi=-np.angle(np.mean(cv));
cvr=cv*np.exp(1j*phi)
```

```
P = 16
psi, eigs = spectrum.mtm.dpss(np.size(cv), NW=P, k=2*P-1)

Zk, weights, eigenvalues = spectrum.mtm.pmtm(cv-np.mean(cv), k=2*P-1,  NFFT=np.size(
S=np.mean(np.transpose(np.abs(Zk)**2), axis=1) * dt

Zk, weights, eigenvalues = spectrum.mtm.pmtm(np.real(cvr-np.mean(cvr)), k=2*P-1,  NFI
sxx=2*np.mean(np.transpose(np.abs(Zk)**2), axis=1) * dt

Zk, weights, eigenvalues = spectrum.mtm.pmtm(np.imag(cvr-np.mean(cvr)), k=2*P-1,  NFI
syy=2*np.mean(np.transpose(np.abs(Zk)**2), axis=1) * dt


fig, ax = plt.subplots(1, 1)
ax.loglog( f[np.where(f>0)],S[np.where(f>0)])#plot positive side
ax.loglog(-f[np.where(f<0)],S[np.where(f<0)])#plot negative side
ax.loglog( f[np.where(f>0)],sxx[np.where(f>0)],linewidth=2)#plot downstream
ax.loglog( f[np.where(f>0)],syy[np.where(f>0)],linewidth=2)#plot crossstream

ax.autoscale(enable=True, tight=True)
ax.set_ylim(1e-1, 1e3)
fig.tight_layout()

#show smoothing extents at different frequencies
io=(1,50,200,800,1600)
yo=(10**2,10**1,10**0,10**(-0.5),10**(-0.75))
for i in range(np.size(io)):
    if i==0:
        plt.hlines(yo[i],f[io[i]],f[io[i]+P],linewidth=3,color='k')
    else:
        plt.hlines(yo[i],f[io[i]-P],f[io[i]+P],linewidth=3,color='k')

ax.vlines(tidefreq()/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle=":", color='
ax.vlines(corfreq(ds["lat"].data)/2/np.pi,ax.get_ylim()[0],ax.get_ylim()[1],linestyle

ax.legend(['Positive Spectrum','Negative Spectrum','Alongstream Spectrum','Cross-Strea
plt.xlabel('Frequency (cycles/day)')
plt.ylabel('Power Spectral Density (cm$^2$/s$^2$ days)')
plt.title('Multitaper Rotary and Cartesian Spectral Estimate with P=16');
```



At low frequencies, we see some very interesting behavior. The low-frequency plateau is associated with the along-stream currents, where we see the along-stream spectrum is more than an order of magnitude stronger than the cross-stream spectrum. This is not unsurprising and likely reflects seasonal or other low-frequency variability in the current strength.

The peak we saw on around one cycle per 10 days in the negative rotary spectrum now appears to be associated with a promient peak in the cross-stream currents. It is unclear whether this represents a quasi-peridodic fluctuation of the cross-stream currents at that frequency, or whether we should interpret this as meaning that lower frequency fluctuations of the cross-stream currents are somehow suppressed or diminished. In either case, the low-frequency bevahior appears to more suitably decomposed into its along- and cross-stream components rather than the rotary components.

A rule of thumb is that a phenomena is most compactly described by the spectral pair exhibiting the highest degree of anisotropy. In the inertial band, we see a much higher degree of anisotropy for the rotary pairing than for the Cartesian pairing. This occurs because the near-inertial band consists of nearly circular ellipses. In that band, the near-inertial variability projects strongly onto both Cartesian components.

The point is that whether the rotary or Cartesian perspective on the spectra is most suitable is not a property of the time series. Rather is it a property of the phenomenon. Even within one time series, we may need to use different perspectives to best analyze phenomena at different frequencies. In general, you should always look at the rotary spectrum, and also the Cartesian spectra rotated either to align with the bathymetry, or the mean current, or the direction of highest variance.

Summary

This lab has walked you through how to choose multi-taper spectral estimation, how these relate to the raw Fourier coefficients, and also how to begin interpreting what you see. You can now try to apply some of the ideas to your own time series.

A few notes: (i) If you don't have complex-valued data, then you only have one spectrum per time series, and the rotary spectra decomposition is not useful. (ii) The periodogram is only shown to illustrate the improvement that we made when we switched to the multitaper estimate. Please do not interpret this as suggesting that the periodogram is something you are supposed to look at and analyze.

The End