Geol 497D/GeoSci697D - Spring 2020
Programming and Data Analysis

William P. Clement

Lab 9: Fast Fourier Transform

Chapters in Learning Scientific Programming with Python (LSPP):
Random numbers; Chapter 6.7
Fourier Transforms; Chapter 6.8

In this laboratory exercise, you will look at the frequency spectrum of a time series using the Fast Fourier Transform. Python functions and scripts. Make sure to **comment your code** and include the **proper Python header** information.

Part I: Fourier Series and Transforms: The Basis of Spectral Analysis

The *Fourier series* for any function is (Agnew et al., 2013):

$$x(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(2\pi n t) + \sum_{n=1}^{\infty} b_n \sin(2\pi n t) \tag{1}$$

where

$$a_n = 2\int_{-1/2}^{1/2} x(t)\cos(2\pi n t)\,dt \tag{2}$$

and

$$b_n = 2\int_{-1/2}^{1/2} x(t)\sin(2\pi n t)\,dt \tag{3}$$

for $n = 1, 2, \ldots$. Be careful. The Fourier series has a number of equivalent representations that differ usually by the coefficients to the a_n and b_n terms. Also, the $\frac{1}{2}$ integration limits indicate that the function is periodic.

Recall Euler's formula,

$$e^{i\theta} = \cos(\theta) + i\sin(\theta). \tag{4}$$

So we can re-write the above equation using complex exponentials

$$e^{i2\pi n t} \quad \text{for} \quad n = -\infty, \ldots, -2, -1, 0, 1, 2, \ldots, \infty, \tag{5}$$

so that the Fourier series becomes

$$x(t) = \sum_{n=-\infty}^{\infty} \tilde{x}_n e^{i2\pi n t} \tag{6}$$

with the coefficients

$$\tilde{x}_n = \int_{-1/2}^{1/2} x(t)\, e^{-i2\pi n t}\,dt \tag{7}$$

for $n = -\infty, \ldots, -2, -1, 0, 1, 2, \ldots, \infty$

The Fourier series can be used to write *any* function as an infinite sum of sines and cosines!! Note that sinusoids are defined by their amplitude and frequency.

Continuous Fourier Transforms

For *continuous* functions, the *Fourier transform* is (Agnew et al., 2013) (page 28)

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft}\, dt \tag{8}$$

and the *inverse Fourier transform* is

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{i2\pi ft}\, df \tag{9}$$

The energy in the time series is defined as (page 66):

$$E = \int_{-\infty}^{\infty} |g(t)|^2\, dt. \tag{10}$$

Parseval's theorem is (page 66):

$$\int_{-\infty}^{\infty} |g(t)|^2\, dt = \int_{-\infty}^{\infty} |G(f)|^2\, df. \tag{11}$$

Discrete Fourier Transforms

In the real world, we collect data at discrete intervals. We thus only have discrete intervals in the Fourier domain. We define the *Fourier frequencies* as (Percival and Walden, 1993) (page 109):

$$f_n \equiv \frac{n}{N\Delta t} \qquad \text{with} \quad n = 0,\, 1,\, \ldots,\, N-1. \tag{12}$$

Here, Δt is the sample interval in either time or space and N is the number of samples.

The *discrete Fourier transform* is (page 110):

$$G_n = \Delta t \sum_{t=0}^{N-1} g_t e^{-i2\pi f_n t \Delta t} = \Delta t \sum_{t=0}^{N-1} g_t e^{-i2\pi nt/N}. \tag{13}$$

and the *inverse discrete Fourier transform* is:

$$g_t = \frac{1}{N\Delta t} \sum_{n=0}^{N-1} G_n e^{i2\pi nt/N}. \tag{14}$$

In these equations, the subscript on G or g indicates discrete samples.

Parseval's theorem in discrete form is (page 111)

$$\Delta t \sum_{t=0}^{N-1} |g_t|^2 = \frac{1}{N\Delta t} \sum_{t=0}^{N-1} |G_n|^2 \tag{15}$$

which states that the energy in the time series *is equal to* the energy in the power spectrum. Nice to know – know one, know the other – know in one domain, know in the other domain without computation!

There are three good reasons for the Fourier functions (Hamming, 1997):

1. time invariance;
2. linearity;
3. the reconstruction of the original signal from the *equally spaced* samples is simple and easy to understand.

Also, dealing with *aliasing* is an important factor in choosing the Fourier domain. A *single* high frequency will appear later as a *single* low frequency in the Nyquist frequency band. The same is not true for any other set of functions, for example, powers of t. Under equal spaced sampling and reconstruction a single high power of t will go into a polynomial (many terms) of lower powers of t (Hamming (1997), page 166). This concept is a bit difficult. Aliasing can be a real problem if we are not careful collecting and transforming our data. We must make sure that we do not alias our data. We must collect our data such that the sample interval ensures we collect the frequencies we are interested in. The Nyquist frequency, $f_{Nyq} = \frac{1}{2\Delta t}$. The Nyquist frequency is the highest frequency we can discern in the data.

Part II: Time Series Analysis of Sinusoids

You will construct a wave from 3 sine waves and then add noise to this wave. Sinusoids are linear functions which means that we can add them together and they behave as we would expect.

The form of the sinusoids (waves) is:

$$wave = Asin(2\pi ft - \phi), \tag{16}$$

where A is the amplitude, f is the frequency, t is the time, and ϕ is the phase of the wave.

You should have the time run from 0 to 5 time units (like, seconds, days, or years), with a sample interval of 0.001. One thing to notice is the factor of 2π in front of the frequency. For sinusoids, the period of the wave is 2π. That is, the sinusoid goes around a unit circle 2π time units. (If you are having trouble with this, DuckDuckGo sine waves.) Sometimes, we use angular frequency (ω) as the frequency instead of f. In this case,

$$\omega = 2\pi f. \tag{17}$$

You should recognize that for $f = 1$, the sinusoid has one full cycle as the time goes from 0 to 1.

Start with 3 separate sine waves:

1. amplitude = 1.0; frequency = 2.0; phase = 0
2. amplitude = 2.0; frequency = 1.0.0; phase = $\frac{\pi}{3}$
3. amplitude = 1.5; frequency = 3.0; phase = $\frac{\pi}{6}$

The other equations are similar. You will then add these 3 waves together:

$$wave = wave_1 + wave_2 + wave_3 \tag{18}$$

Next, you should add noise to the wave to simulate data more realistically.

$$wave_n = wave + 0.1 * noise, \tag{19}$$

where `noise` is Gaussian random noise with a 0 mean and a standard deviation of 1. See Chapter 6.7 of LSPP for a better understanding of generating random numbers with python. You should reduce the amplitude of the noise by multiplying the noise by 0.1. In general, the random noise needs to be modified to an appropriate level for the problem. Often, generated noise has a much larger amplitude than the input values.

Once you have made the time series, $wave_n$, you should take the fast Fourier transform (fft) of this series. This is the standard method of time series analysis, although there are many other techniques. You should consult Chapter 6.8 of LSPP for more about using the Fourier transform in python.

Use the command `np.fft.fft()` to determine the Fourier transform. You will also use `np.fft.fftfreq()` to determine the frequency interval of the spectrum. Again, check the text for more information. You will only want to plot the frequencies from 0 to the Nyquist frequency. Check with the text to learn how to do this. Since we are dealing with a real-valued time series, we only need the non-negative frequencies to describe the spectrum.

When you plot the data, create two figures. The *first figure* will show the four waves that you created, each wave in its own subplot. Plot each wave and its noisy counterpart in its own subplot so that you can see how the noise affects the data and also how the 3 waves add together.

The *second figure* will have 3 subplots that show

- the time series of the combined noisy wave,
- its amplitude spectrum, and
- its phase spectrum.

For the spectrum plots, only plot the x-axis from 0 to 5 Hz. The maximum frequency in the wave should be 3. However, the fft computes the maximum frequency as 1/(2*sample rate), which in our case is 1/0.002 or 500 Hz. Keeping all 500 Hz for the spectrum will squinch the first frequencies into a very small part of the plot.

To find the amplitude of a complex number or vector, use `np.abs()`; to find its phase, use `np.angle()`. The power is the square of the amplitude. As the text states, the way the fft is designed, you need to divide the fft by n, the number of samples in the fft. (**NB**: According to LSPP, you multiply by $2/n$, not $1/n$, but this gives the wrong amplitude, I think. Perhaps you should check this for yourself.) If you do this correctly, you will see that the plotted amplitudes are the same as the amplitudes of the waves (1, 4, 2.25). You should also see that the phase spectrum returns the input phases of the waves. The phase spectrum looks like a mess and is not particularly informative; this is why it is not often shown. Make sure you read Chapter 6.8 to better understand the fft.

After you successfully write your fft code, try selecting the positive frequencies only using the `np.where()` command. I outline the procedure below.

```
wave1_fft= np.fft.fft(wave1)
sample_freq = np.fft.fftfreq(wave1.size, d=dt)
pidxs = np.where(sample_freq >= 0)
freqs1 = sample_freq[pidxs]
power1 = (np.abs(wave1_fft[pidxs]))**2 / len(wave1)
```

Look at the line with the command `np.where`. This command selects those frequencies that are greater than and equal to 0. For real-valued data, which all data is, the spectrum is symmetric about 0, so we only need to look at those frequencies that are 0 or greater. `np.where` returns the indices where its argument is true. In our case, when `sample_freq` is 0 or greater.

Again, you should create a figure with 3 subplots, one for the input time series, one for the power spectrum, and one for the phase spectrum.

Part III: Time Series Analysis of Whately Water Level Data

Now that you have written a code to compute the spectrum of a test data set, use the water level data from Whately, MA that you analyzed in the previous lab and compute its power spectrum. In this case, you will plot the power spectrum on a log plot. You will notice that the range of values for the power is several orders of magnitude. Using a simple linear axis for these values will not show much interesting. Use `plt.semilogy()` to plot the power data. In python, you can chose to plot either axis with a log scale, `plt.semilogx` for the x-axis or `plt.semilogy()` for the y-axis. `plt.log()` will plot both axes with a log scale.

Your code should plot one figure with two plots; the input time series as the top plot and the power spectrum (with a log scale for the y-axis) as the lower plot. Look at the power spectrum and notice the strong amplitude at 2 cycles/day. This corresponds to the Earth tides that have high tides twice a day. The water level is strongly influenced by tides! I find that really amazing.

Bibliography

D. C. Agnew, R. L. Parker, and C. Constable. Geophysical data analysis: Fourier methods. Class notes, SIO 223B, Scripps Institution of Oceanography, 2013.

R. W. Hamming. *The Art of Doing Science and Engineering: Learning to Learn.* Gordon and Breach Science Publishers, 1997.

D. B. Percival and A. T. Walden. *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques.* Cambridge University Press, 1993.