Lesson 20: Fourier Analysis

1 Fourier Transform

Fourier analysis is a powerful tool in physics. We often deal with data that consist of a time–dependent amplitude, such as an audio signal or an X-ray signal. Fourier analysis tells us which frequencies make up the signal.

Any periodic signal can be described as a Fourier series, that is, a sum of sine's and cosine's. The corresponding concept for a (possibly) non–periodic signal $a(t)$ is the *Fourier transform*:

$$A(f) = \int_{-\infty}^{\infty} a(t) e^{-2\pi i f t} dt \tag{1}$$

The inverse transform is

$$a(t) = \int_{-\infty}^{\infty} A(f) e^{2\pi i f t} df \tag{2}$$

The function $a(t)$ describes the signal in the time domain, while $A(f)$ describes the signal in the frequency domain.

Recall Euler's formula, which shows that the complex exponential is a combination of trig functions: $\exp(-2\pi i f t) = \cos(2\pi f t) - i \sin(2\pi f t)$. Consider a signal $a(t)$ that contains a term $\cos(2\pi f_0 t)$. If $f = f_0$ in the integral (1), the $\cos(2\pi f_0 t)$ in $a(t)$ will overlap the $\cos(2\pi f t)$ in $\exp(-2\pi i f t)$ to give a positive contribution for all t. Correspondingly, the integral (1) yields a large value for A when $f = f_0$. On the other hand, if $f \neq f_0$, then the factors $\cos(2\pi f_0 t)$ and $\cos(2\pi f t)$ from $a(t)$ and $\exp(-2\pi i f t)$ will not overlap for all t. The integrand in (1) will be positive for some values of t and negative for others. This leads to cancellations in the integral over t and a small (if not zero) value for A. In this way, the Fourier transform (1) "picks out" the frequencies contained in the time domain signal $a(t)$ and builds the frequency domain signal $A(f)$.

Homework: Consider the signal $a(t) = \cos(6\pi t) e^{-t^2}$. Plot a graph of $a(t)$ vs. t for $-4.0 \leq t \leq 4.0$. Compute the Fourier transform $A(f)$ using Simpson's rule integration. Plot both the real and imaginary parts of $A(f)$ vs. f for $-6.0 \leq f \leq 6.0$. Is A peaked at the expected frequencies? Repeat the analysis with cos changed to sin.

Typically, the time domain signal $a(t)$ is real. In this case the frequency domain signal satisfies $A(-f) = [A(f)]^*$, where the asterisk denotes complex conjugation. This relation shows that $A(f)$ is completely characterized by its positive frequency values. In other words, given the amplitude $A(f)$ for positive frequencies, we can use the relation $A(-f) = [A(f)]^*$ to determine the amplitude for negative frequencies.

2 Discrete Fourier Transform

Most often our time domain signal is know only at discrete points in time. The discrete signal might come from an experiment, where the detectors and instrumentation record the amplitude at discrete times. The discrete signal might come from a numerical calculation where the time evolution takes place in discrete steps. Let's assume the signal is sampled at times $t = k\Delta t$ for $k = 0, \ldots, N-1$, and denote the amplitude at these times by $a_k = a(t_k)$. The Fourier transform (1) for the continuous function $a(t)$ is replaced by the *discrete Fourier transform*,

$$A_n = \sum_{k=0}^{N-1} a_k e^{-2\pi i k n/N} \tag{3}$$

with $n = 0, \ldots, N-1$. The inverse transformation is

$$a_k = \frac{1}{N} \sum_{n=0}^{N-1} A_n e^{2\pi i k n/N} \tag{4}$$

Physically, a_k is the amplitude of the time domain signal at times $t_k = k\Delta t$ for $k = 0, \ldots, N-1$. Correspondingly, A_n is the amplitude of the frequency domain signal at frequencies

$$f_n = \frac{n}{N\Delta t} \ , \qquad n = 0, \ldots, N/2-1, -N/2, \ldots, -1 \tag{5}$$

if N is even. If N is odd, the frequencies are defined by $f_n = n/(N\Delta t)$ for $n = 0, \ldots, (N-1)/2, -(N-1)/2, \ldots -1$.

The *fast Fourier Transform* (FFT) is a very efficient computational algorithm for computing the discrete Fourier transform (3). Note that there are N frequency domain amplitudes A_n, and each one of these is constructed as a sum of N terms. Thus, a direct calculation of the A_n's requires a computation time that is proportional to N^2. The FFT reorganizes the calculation in a clever way so that the computation time is proportional to $N \log(N)$. For large N, this can be a huge savings.

The numpy library includes a built–in implementation of the FFT in the `numpy.fft` sub–library. Given an array `a` containing the time domain amplitudes a_k, the command `A = numpy.fft.fft(a)` will create an array `A` containing the frequency domain amplitudes A_n. The frequencies themselves can be computed from the definitions given above; alternatively, they can be generated with the command `f = numpy.fft.fftfreq(`$N, \Delta t$`)`.

Homework: Consider the signal $a(t) = \cos(2\pi f_0 t)e^{-t^2}$. Sample this function at $N = 100$ equally spaced times from $t = -4.0$ to $t = 4.0$, using $f_0 = 3.0$. Use the numpy FFT commands to compute the discrete Fourier transform A_n, and plot of graph of the real and imaginary parts of A_n vs. f. Is it peaked at the expected frequencies?

3 Nyquist frequency and aliasing

The maximum frequency obtained from the discrete Fourier transform is $f_n = n/(N\Delta t)$ with $n = \pm N/2$. This defines the *Nyquist critical frequency*

$$f_c \equiv 1/(2\Delta t) \tag{6}$$

Consider a signal $a(t)$ that includes a term $\cos(2\pi f_c t)$. It has a peaks at $t = 0$, $2\Delta t$, *etc*, and troughs at $t = \Delta t$, $3\Delta t$, *etc*. This is the highest frequency that can be represented by the discrete amplitudes a_k. If the continuum signal $a(t)$ contains a term with frequency higher than f_c, some of the peaks and troughs will be missed between the sampling times. In this case the discrete Fourier transform will incorrectly map the high frequency signal onto a lower frequency.

As an example, consider a signal $\cos(15\pi t)$ that is sampled at intervals $\Delta t = 0.1$. This signal has frequency $f = 7.5$, which is above the Nyquist critical frequency $f_c = 1/(2\Delta t) = 5$. The discrete amplitudes are

$$a_0 = \cos(0) = 1 \tag{7a}$$
$$a_1 = \cos(1.5\pi) = 0 \tag{7b}$$
$$a_2 = \cos(3.0\pi) = -1 \tag{7c}$$
$$a_3 = \cos(4.5\pi) = 0 \tag{7d}$$

etc. The discrete Fourier transform will interpret this as a signal with frequency $f = 2.5$. To see why, let's compute the amplitudes for the signal $\cos(5\pi t)$, sampled at intervals $\Delta t = 0.1$. They are:

$$a_0 = \cos(0) = 1 \tag{8a}$$
$$a_1 = \cos(0.5\pi) = 0 \tag{8b}$$
$$a_2 = \cos(1.0\pi) = -1 \tag{8c}$$
$$a_3 = \cos(1.5\pi) = 0 \tag{8d}$$

etc. Precisely the same results!

The process of "mapping" the high frequency components of a signal (with $f > f_c$) onto lower frequencies (with $f < f_c$) is called *aliasing*. It is important to recognize that the discrete Fourier transform can contain spurious signals that have been aliased from the high frequency components of the original time domain signal.

Exercise 1: Continue with the analysis from the previous Homework. What is the Nyquist critical frequency f_c for this sampling rate? Plot A_n vs. f for frequency values $f_0 = 4$, 5, *etc*, continuing beyond the critical frequency. What happens to the graph as f_0 is increased beyond f_c?

4 Power spectrum

The power at each frequency is, roughly, the square of the amplitude A_n. More precisely, the power spectrum $P(f)$ is defined for non-negative frequencies

$$f_n = \frac{n}{N\Delta t} \ , \quad n = 0, 1, \ldots N/2 \tag{9}$$

(where N is assumed to be even) by

$$P(f_0) = \frac{1}{N^2}|A_0|^2 \tag{10a}$$

$$P(f_n) = \frac{1}{N^2}\left[|A_n|^2 + |A_{N-n}|^2\right] \ , \quad n = 1, 2, \ldots N/2 - 1 \tag{10b}$$

$$P(f_c) = \frac{1}{N^2}|A_{N/2}|^2 \tag{10c}$$

Note that $|A_n|^2$ is the absolute value of the complex amplitude: $|A_n|^2 \equiv (A_n)(A_n)^*$. The complex conjugate can be found with numpy's `conj()` function. A graph of the power $P(f)$ vs. f is sometimes called a periodogram.

Homework: Download `Lesson20_Data.txt` from the course Moodle page. The data is arranged in two columns. The first column is time, and the second column is the amplitude of a signal. Write a program using the built–in FFT functions to generate two plots: i) amplitude vs. time for the original signal, and ii) the periodogram (power vs. frequency).

Note: To plot the periodogram, you will need to construct an array containing the non-negative frequencies (9). If `f = numpy.fft.fftfreq(`$N, \Delta t$`)`, then the non-negative frequencies consist of the elements `f(0:N/2)` along with the Nyquist frequency f_c.