Lecture notes 4: Fourier Analysis

Definitions

There are many common (and confusing, but ultimately trivial!) differences in defining the *Fourier transform*. One common defintion is

$$F(\nu) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi\nu t} dt$$

Thus $F(\nu)$ gives the wavenumber representation of the function $f(t)$. The inverse transform can be written

$$f(t) = \int_{-\infty}^{\infty} F(\nu) e^{i2\pi\nu t} d\nu$$

$F(\nu)$ is in general a complex function whose interpretation may be aided by expression in the polar coordinate form $F(\nu) = A(\nu)e^{i\phi(\nu)}$, where $A(\nu)$ and $\phi(\nu)$ are real functions where $A(\nu) =\mid F(\nu) \mid$ is the amplitude and $\phi(\nu) = \arg[F(\nu)]$ is the phase. Note that we then can write the inverse transform as

$$f(t) = \int_{-\infty}^{\infty} A(\nu) e^{i[2\pi\nu t + \phi(\nu)]} d\nu,$$

which is seen to be a recombination of all the frequency components of f(t). Each component is a complex sinusoid of the form $e^{2\pi i\nu t}$ whose amplitude is $A(\nu)$ and whose initial phase (at $t = 0$) is $\phi(\nu)$. This interpretation of the Fourier transform clearly shows its relation to the *Fourier series*[1]

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

with coefficients given by

$$\begin{aligned} a_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx) dx \\ b_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx) dx. \end{aligned}$$

It is common to perform the substitution $\nu = \omega/2\pi$ which gives

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt \\ f(\omega) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(t) e^{i\omega t} d\omega \end{aligned}$$

[1] Note that by using Euler's formula $e^{inx} = \cos(nx) + i\sin(nx)$ a more concise (and modern) form can be used: $f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$ with $c_n = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-inx} dx$.

Properties

Fourier transforms exhibit a number of properties that are very useful:

- Linearity
$$af(t) + bg(t) \Longleftrightarrow aF(\omega) + bG(\omega)$$
- Multiplication
$$f(t)g(t) \Longleftrightarrow \frac{1}{2\pi}(F \otimes G)(\omega)$$
where we define the *convolution* $\otimes$ as
$$K(\omega) \equiv \int_{-\infty}^{\infty} F(\omega')G(\omega - \omega')d\omega'$$
- Convolution
$$(f \otimes g)(t) \Longleftrightarrow F(\omega)G(\omega)$$
- Conjugation
$$\overline{f(t)} \Longleftrightarrow \overline{F(-\omega)}$$
- Scaling
$$f(at) \Longleftrightarrow \frac{1}{|a|}F\left(\frac{\omega}{a}\right)$$
- Time reversal
$$f(-t) \Longleftrightarrow F(-\omega)$$
- Time shift
$$f(t - t_0) \Longleftrightarrow e^{-i\omega t_0}F(\omega)$$
- Parsevals theorem
$$\int_{-\infty}^{\infty} f(t)\overline{g(t)}dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\overline{G(\omega)}d\omega$$

Sampling Theorem and Aliasing

We will in general be dealing with functins $h(t)$ which are sampled at evenly spaced intervals in time (or space). Let Δ denote the time (space) interval between consequtive samples.

For any sampling interval Δ, there is a special frequency ν_c called the *Nyquist frequency* given by
$$\nu_c \equiv \frac{1}{2\Delta}$$
The critical sampling of a sine wave is two sample points per cycle. There are two aspects of the critical frequency. First, if the original signal is bandwith limited to frequencies smaller than ν_c then the function is completely determined by its discrete samples. This is the *sampling theorem.* However, if a signal is *not* bandwidth limited to less than the Nyquist frequency then the power that lies outside the range $-\nu_c < \nu < \nu_c$ is spuriously moved into that range. This phenomena is called *aliasing.*

FFTs

Let us now estimate a Fourier transform from a finite number of sampled points. Suppose that we have N conscutive sampled values

$$h_k \equiv h(t_k) \qquad t_k \equiv k\Delta \qquad k = 0, 1, 2, \ldots, N-1$$

so that the sampling interval is Δ. Also assume N is even. With N numbers of input, we cannot produce more than N independent numbers of output. So, instead of trying to estimate the Fourier transform $H(\nu)$ at all values of ν in the range $-\nu_c$ to ν_c, let us seek estimates at only the discrete values

$$\nu_n \equiv \frac{n}{N\Delta}, \qquad n = \frac{-N}{2}, \ldots, \frac{N}{2}$$

The extreme values of n correspond to the lower and upper values fo the Nyquist critical frequency range.

The remaining step is to approximate the continuous tranform by a discrete sum

$$H(\nu_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i \nu_n t} dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi i \nu_n t_k} \Delta = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N}$$

The final summation is called the *discrete Fourier transform* of the N points h_k. Let us denote it by H_n:

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} \tag{1}$$

Up to now we have taken the view that index n varies from $-N/2$ to $N/2$. However, since it is clear that equation 1 is periodic in n we also have that $H_{-n} = H_{N-1}$. With this in mind it is customary to let n vary from 0 to $N-1$ (one period). When this convention is followed the *zero* frequency corresponds to $n = 0$, positive frequencies $0 < \nu < \nu_c$ correspond to valuses $1 \leq n \leq N/2-1$, while negative frequencies $-\nu_c < \nu < 0$ correspond to $N/2_1 \leq n \leq N-1$. The value $n = N/2$ corresponds to both $\nu = \nu_c$ and $\nu = -\nu_c$.

The discrete *inverse* transform which recovers h_k form the H_n is

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n/N}$$

How do we implement the discrete transform in code? The brute strength approach takes of order N^2 operations: Define W as the complex nubmer

$$W \equiv e^{2\pi i/N}$$

Then equation 1 can be written

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k,$$

i.e. the vector h_k is multiplied by a matrix whose $(h,k)^{th}$ element is the constant W to the power $n \times k$. The matrix multiplication produces a vector result whose components are H_n. This multiplication evidently needs N^2 complex multiplications.

In fact, the discrete Fourier transform can be achieved in $N \log_2 N$ operations with an algorithm called the it Fast Fourier Transform or *FFT*. Here is one derivation of the FFT, that of Danielson and Lanczos in 1942. They showed that a discrete transform of length N can be rewritten as the sum of two discrete transforms, each of length $N/2$. One of the two is formed from the even numbered points of the original N, the other from the odd-numbered points.

$$\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{2\pi i2jk/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi i(2j+1)k/N} f_{2j+1} \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_{2j+1} \\
&= F_k^e + W^k F_k^o
\end{aligned}$$

F_k^e denotes the k^{th} component of the Fourier transform of length $(N/2)$ formed from the even components of the original f_j, while F_k^o is the corresponding transform of lenght $(N/2)$ from the odd components.

This procedure can be applied recursively; having reduced the problem of computing F_k to that of computing F_k^e and F_k^o, one can do the same reduction of F_k^e to the problem of computing the transform of *its* $N/4$ even-numbered inputs data, F_k^{ee}, and $N/4$ odd-numbered data F_k^{eo}. When we start with an original N which is a integer power of 2 (one should only use FFTs with N a power of 2, padding the input data with zeroes is necessary) we can continue applying the Danielson-Lanczos method until the original data is subdivided all the way down to transforms of length one. The Fourier transfrom of lenght one is just the identity operation that copies its one input number into its output slot. Thus, for every pattern of e's and o's numbering $\log_2 N$ in all there is a one-point transform that is given by

$$F_k^{eoeeoeo\cdots oee} = f_n \qquad \text{for some } n$$

Which value of n corresponds to which pattern? Reverse the pattern of e's and o's, let $e = 0$ and $o = 1$ and one will have, *in binary*, the value of n. This is because the successive subdivisions of the data into odd and even are tests of successive least significant bits of n. Thus by rearranging the input vector f_j in bit reversed order so that the indivdual numbers are in the order obtained by bit reversing j. The points are given as one-point transforms. These are recombined with the adjacent number to give two-point transforms, then combine adjacent pairs again to give 4-point transforms, and so on, using the Danielson-Lanczos formula at every step.

The FFT therefore has a structure where first the data are sorted into bit reversed order and thereafter the transforms of length $2, 4, 8, /ldots, N$ are computed implementing the Danielson-Lanczos formula.

Exercises

Experiment with the `fft` function in `idl`. First make an x-axis, *eg* `idl> x=findgen(2000)/100.*2.*!pi`

1. Compute the amplitude of the transform of $f = \sin x$. In plotting this amplitude, what should be used for the x-axis?

2. What happens if you set the edges of f to zero: *eg* multiply f by a step-function s such as `idl> s=fltarr(2000) & s[200:1800]=1.0`? Overplot the amplitude of the tranform as you narrow the region of s that is equal to one.

3. Compute the transform of step-functions of various widths.

4. Add sinusoidal functions to f with different frequencies and check the resulting transforms. What happens to the transform if you add a constant to f?

5. Consider a function g given by the sum of three sinusoidals of differing frequency. Construct such a g and remove one of the frequencies from g using forward and back transforms FFT's in `idl`.