The Discrete Fourier Transform

Let $a = (a_0, a_1, ..., a_{n-1})$ be a sequence of complex numbers. The *Discrete Fourier Transform* of a is a new sequence of numbers $y = (y_0, y_1, ..., y_{n-1})$ where

$$y_k = \sum_{j=0}^{n-1} a_j \left(\omega_n{}^k\right)^j \qquad (1)$$

Here ω_n is the complex n^{th} root of unity, $\omega_n = e^{2\,\pi\,i/n}$.

The Fast Fourier Transform

Formula (1) provides a straight-forward algorithm for computing the discrete Fourier transform of a sequence of numbers. Since the formula requires that we sum n terms for each of n values of k, the basic algorithm requires a number of elementary arithmetic operations that is proporational to n^2. The Fast Fourier Transform (FFT) replaces the basic calculation (1) with a more efficient algorithm that requires a number of elementary arithmetic operations that is proportional to $n\ lg\ n$. For large values of n, $n\ lg\ n$ is significantly smaller than n^2, so the FFT will represent a significant improvement over the naive algorithm used to compute the DFT.

The starting point for the FFT is the observation that the summation on the right hand side of (1) looks like a polynomial. Specifically, if we introduce

$$A(x) = \sum_{j=0}^{n-1} a_j\, x^j$$

we see that the formula for the DFT can be written

$$y_k = A(\omega_n{}^k)$$

FFT is based on a divide and conquer strategy. Specifically, we introduce two new functions

$$A^{[0]}(x) = a_0 + a_2\, x + a_4\, x^2 + \cdots + a_{n-2}\, x^{n/2-1}$$

$$A^{[1]}(x) = a_1 + a_3\, x + a_5\, x^2 + \cdots + a_{n-1}\, x^{n/2-1}$$

and note that

$$A(x) = A^{[0]}(x^2) + x\, A^{[1]}(x^2)$$

Where the recursion comes in is when we note that the computation of $A^{[0]}$ and $A^{[1]}$ looks like the computation of a DFT on the two subsequences $(a_0, a_2, a_4, ..., a_{n-2})$ and $(a_1, a_3, a_5, ..., a_{n-1})$.

The trick that makes FFT successful is the fact that the sequences $y^{[0]}$ and $y^{[0]}$ returned by Recursive-FFT($a^{[0]}$) and Recursive-FFT($a^{[1]}$) can be easily combined to produce the desired sequence y.

Pseudocode for the FFT

Recursive-FFT(a)
1. $n = length[a]$
2. **if** n==1
3. **then return** a
4. $\omega_n = e^{2\pi\, i/n}$
5. $\omega = 1$
6. $a^{[0]} = (a_0,\ a_2,\ a_4,\ ...,\ a_{n-2})$
7. $a^{[1]} = (a_1, a_3, a_5, ..., a_{n-1})$
8. $y^{[0]} =$ Recursive-FFT($a^{[0]}$)
9. $y^{[1]} =$ Recursive-FFT($a^{[1]}$)
10. **for** $k = 0$ **to** $n/2$ - 1 **do**
11. $y_k = y_k^{[0]} + \omega\ y_k^{[1]}$
12. $y_{k+n/2} = y_k^{[0]}$ - $\omega\ y_k^{[1]}$
13. $\omega = \omega\ \omega_n$
14. **return** y

Computing the inverse DFT

Another useful mathematical accident is that the formula for the inverse DFT looks very similar to (1):

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \left(\omega_n^{-j}\right)^k$$

This means that we can use the same algorithm as before with the roles of a and y switched, ω_n replaced by ω_n^{-1}, and an extra multiplication by a factor of $1/n$.

Complex Fourier Series

Consider now the problem of solving the BVP with periodic boundary conditions on the interval $[-l, l]$:

$$-\frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} = \lambda\ u(x)$$

$$u(-l) = u(l)$$

$$\frac{\mathrm{d}u}{\mathrm{d}x}(-l) = \frac{\mathrm{d}u}{\mathrm{d}x}(l)$$

This problem is essentially the problem of finding eigenfunctions and eigenvalues of the differential operator

$$L_P\, u = -\frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2}$$

on a space of periodic functions.

One solution to this BVP is that the eigenfunctions are functions of the form

$$u_n^{(1)}(x) = \sin(\frac{n\,\pi}{l}\,x)$$

$$u_n^{(2)}(x) = \cos(\frac{n\,\pi}{l}\,x)$$

for $n = 1, 2, \ldots$ and

$$u_0(x) = 1$$

for $n = 0$. The associated eigenvalues are

$$\lambda_n = \frac{n^2\,\pi^2}{l^2}$$

Another solution to the BVP uses complex-valued eigenfunctions

$$u_n(x) = e^{i\,n\,\pi\,x/l}$$

for $n = 0, \pm 1, \pm 2, \ldots$ and associated eigenvalues

$$\lambda_n = \frac{n^2\,\pi^2}{l^2}$$

These eigenfunctions are legitimate eigenfunctions in the sense that they solve the BVP. Additionally, these eigenfunctions also satisfy appropriate orthogonality conditions, provided we introduce the appropriate inner product for complex-valued functions. If $u(x)$ and $v(x)$ are two complex-valued functions, we can define an L^2 inner product by

$$(u(x),v(x)) = \int_{-l}^{l} u(x)\overline{v(x)}\,dx$$

Here $\overline{v(x)}$ is the complex conjugate of $v(x)$.

It turns out that the eigenfunctions $u_n(x)$ are orthogonal with respect to this inner product.

$$(u_n(x)\ ,\ u_m(x)) = \int_{-l}^{l} e^{i\,n\,\pi\,x/l}\overline{e^{i\,m\,\pi\,x/l}}\,dx = \frac{l}{i\,(n\text{-}m)\,\pi}\ e^{i\,(n\text{-}m)\,\pi\,x/l}\Big|_{-l}^{l}$$

$$= \frac{l}{i\,(n\text{-}m)\,\pi}\ e^{i\,(n\text{-}m)\,\pi} - \frac{l}{i\,(n\text{-}m)\,\pi}\ e^{-i\,(n\text{-}m)\,\pi}$$

$$= \frac{l}{i\,(n\text{-}m)\,\pi}\,(\cos((n\text{-}m)\,\pi) - \cos(-(n\text{-}m)\pi))$$

$$= 0$$

For $n = m$ we have

$$(u_n(x)\ ,\ u_n(x)) = \int_{-l}^{l} e^{i\,n\,\pi\,x/l}\overline{e^{i\,n\,\pi\,x/l}}\,dx\ = \int_{-l}^{l} e^{i\,n\,\pi\,x/l}e^{-i\,n\,\pi\,x/l}\,dx = \int_{-l}^{l} 1\,dx = 2\ l$$

Complex Fourier series

Since the complex exponential functions

$$u_n(x) = e^{i\,n\,\pi\,x/l}$$

are legitimate eigenfunctions, we can now seek to express arbitrary functions $f(x)$ as combinations of these eigenfunctions.

$$f(x) = \sum_{n\,=\,-\infty}^{\infty} c_n\ e^{i\,n\,\pi\,x/l}$$

The coefficients c_n are the *complex Fourier coefficients* of $f(x)$. As usual, these coefficients are computed via the orthogonality properties of the eigenfunctions $u_n(x)$.

$$(f(x),u_k(x)) = \left(\sum_{n\,=\,-\infty}^{\infty} c_n\ e^{i\,n\,\pi\,x/l}\ ,\ u_k(x)\right) = c_k\ (u_k(x)\ ,\ u_k(x)) = c_k\ (2\ l)$$

or

$$c_k = \frac{(f(x)\ ,\ u_k(x))}{2\ l} = \frac{1}{2\ l}\int_{-l}^{l} f(x)\ e^{-i\,k\,\pi\,x/l}\,dx$$

The main computational effort in computing a Fourier expansion lies in computing these integrals. If we want to compute the Fourier coefficients for, say, $k = -N$ up through and including $k = N - 1$ for some N we will have to compute a total of $2\ N$ integrals.

Worse yet, some functions $f(x)$ do not allow us to easily compute the resulting integrals in closed form. In those cases, we have to estimate the Fourier coefficients the computing numerical estimates for the integrals. One technique for doing this is the trapezoid rule

$$\int_a^b g(x)\,dx \approx \frac{h}{2}\left[g(x_0) + 2\sum_{j\,=\,1}^{N\text{-}1} g(x_j) + g(x_N)\right]$$

The trapezoid rule assumes that we are subdividing the interval $[a,b]$ into N equal length subdivisions of length $h = (b\text{-}a)/N$ at the division points $x_j = a + j\ h$.

To apply the trapezoid rule to the problem of estimating Fourier coefficients, we subdivide the interval $[-l,l]$ into $2\ N$ subdivisions of length $h = (2\ l)/(2\ N) = l/N$ at division points $x_j = j\ h$ for j ranging from $-N$ to N. This gives us

$$c_k = \frac{1}{2\,l}\int_{-l}^{l} f(x)\; e^{-i\,k\,\pi\,x/l}\,dx \approx \frac{1}{2\,l}\,\frac{h}{2}\left[f(x_{-N})\,e^{-i\,k\,\pi\,x_{-N}/l} + 2\sum_{j=-N+1}^{N-1} f(x_j)\,e^{-i\,k\,\pi\,x_j/l} + f(x_N)\,e^{-i\,k\,\pi\,x_N/l}\right]$$

If we assume that $f(-l) = f(l)$ this simplifies to

$$c_k = \frac{1}{2\,N}\left[\sum_{j=-N}^{N-1} f(x_j)\,e^{-i\,k\,\pi\,x_j/l}\right]$$

Using the FFT to compute Fourier coefficients

The final formula we developed above bears an intriguing resemblance to an in inverse FFT.

If we take advantage of the periodicity of the function $f(x)$ and the exponential terms, we can shift the summation by recognizing that for j from $-N$ up to -1 we have

$$f(x_j)\,e^{-i\,k\,\pi\,x_j/l} = f(x_{j+2N})\,e^{-i\,k\,\pi\,x_{j+2N}/l}$$

so that the portion of the sum from $-N$ up to -1 can be replaced by a sum from N to $2\,N - 1$.

$$c_k = \frac{1}{2\,N}\left[\sum_{j=0}^{2N-1} f(x_j)\,e^{-i\,k\,\pi\,x_j/l}\right]$$

Also, if we rewrite the terms

$$e^{-i\,k\,\pi\,x_j/l}$$

as

$$e^{-i\,2\,\pi\,k\,x_j/(2l)} = e^{-i\,2\,\pi\,k\,(j\,(2l)/(2\,N))/(2l)} = e^{-i\,2\,\pi\,k\,j\,/(2\,N)}$$

we recognize these factors as

$$\left(\omega_{2N}{}^{-j}\right)^k$$

Thus we see that the coefficients we are trying to compute look like

$$c_k = \frac{1}{2\,N}\left[\sum_{j=0}^{2N-1} f_j\,\left(\omega_{2N}{}^{-j}\right)^k\right]$$

where $f_j = f(x_j) = f(j\;h)$. This is precisely the form of an inverse FFT of a sequence of $2\;N$ data points $\{f_j\}$. The accompanying Mathematica notebook will show the details in an example.