4. Discrete Fourier Transform

In the second lecture we covered the Fourier transform of continuous functions but when we work with digital data, functions are sampled at discrete points which we will assume are uniformly spaced (i.e. at a time interval of Δt for time series or Δx for spatial data).

Discrete Fourier Transform

The Fourier series of a periodic function can be written in terms of complex exponentials as

$$y(t) = \sum_{k=-\infty}^{\infty} c_k \exp\left(\frac{ik2\pi t}{T}\right) \tag{4-1}$$

with

$$c_k = \frac{1}{T}\int_{-T/2}^{T/2} y(t)\exp\left(\frac{-ik2\pi t}{T}\right) dt \tag{4-2}$$

To construct the discrete Fourier Transform, we will replace y(t) by a discrete representation $y_j = 0, 1, 2, \ldots, N\text{-}2, N\text{-}1$ with the sample interval Δt related to the period T by $N\Delta t = T$. Equation (4-2) becomes a discrete Fourier transform (DFT)

$$c_k = \frac{1}{N\Delta t}\sum_{j=0}^{N-1} y_j \exp\left(-ik\frac{2\pi}{N\Delta t} j\Delta t\right)\Delta t = \frac{1}{N}\sum_{j=0}^{N-1} y_j \exp\left(-i\frac{2\pi}{N} jk\right) \tag{4-3}$$

Now consider c_{k+N}

$$\begin{aligned} c_{k+N} &= \frac{1}{N}\sum_{j=0}^{N-1} y_j \exp\left(-i(k+N)\frac{2\pi}{N} j\right) \\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j \exp\left(-ik\frac{2\pi}{N} j\right)\exp\left(-iN\frac{2\pi}{N} j\right) \\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j \exp\left(-i\frac{2\pi}{N} jk\right)\exp(-i2\pi j) \\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j \exp\left(-i\frac{2\pi}{N} jk\right) = c_k \end{aligned} \tag{4-4}$$

where we have used the fact that $\exp(-i2\pi j) = 1$ for integer j. Thus, c_k is completely defined by specifying N values – it just repeats itself for values outside this range of indices.

In *MATLAB* and in most other implementations of the DFT, the components are specified for

$$c_k, \quad k = 0, 1, \ldots . N-2, N-1 \tag{4-5}$$

If we specified the Fourier transform based on its derivation from a Fourier series and our understanding that the components at negative and positive frequencies define the phase while summing to yield a real time series, we would specify the components

$$\begin{aligned} c_k, \quad & k = -\left(\frac{N}{2}-1\right), \ldots . -1, 0, 1, \ldots . N/2 \quad \text{for } N \text{ even} \\ & k = -\frac{(N-1)}{2}, \ldots . -1, 0, 1, \ldots . \frac{(N-1)}{2} \quad \text{for } N \text{ odd} \end{aligned} \tag{4-6}$$

Since c_k repeats every N values, the ranges defined by equations (4-5) and (4-6) are completely equivalent although the alternate ordering can be confusing at first. To change from one convention to the other we only need to reorder the components on the DFT. In *MATLAB* the function `fftshift` does this reordering for you.

Inverse Discrete Fourier Transform

We might guess from inspection of equation (4-1) that the inverse discrete Fourier transform is given by

$$y_j = \sum_{k=0}^{N-1} c_k \exp\left(i \frac{2\pi}{N} jk \right) \tag{4-7}$$

To show that this works we can substitute for c_k using equation (4-3)

$$y_j = \sum_{k=0}^{N-1} \left[\frac{1}{N} \sum_{l=0}^{N-1} y_l \exp\left(-i \frac{2\pi}{N} lk \right) \right] \exp\left(i \frac{2\pi}{N} jk \right) = \frac{1}{N} \sum_{l=0}^{N-1} y_l \sum_{k=0}^{N-1} \exp\left(-i \frac{2\pi}{N} (l - j) k \right) \tag{4-8}$$

Now the sum over k is a geometric progression of the form

$$S = 1 + r + r^2 \ldots + r^{N-1} \tag{4-9}$$

with

$$r = \exp\left(-i \frac{2\pi}{N} (l - j) \right) \tag{4-10}$$

If $l = j$ we can see that the exponent term, r is unity and the sum $S = N$. If $l \neq j$ we can sum the geometric expression by first multiplying equation (4-9) by r

$$rS = r + r^2 \ldots + r^{N-1} + r^N \tag{4-11}$$

Subtracting equation (4-11) from equation (4-9) yields

$$(1 - r) S = \left(1 - r^N \right) \tag{4-12}$$

which gives the following expression for the sum

$$S = \frac{\left(1 - r^N \right)}{(1 - r)} \tag{4-13}$$

If we substitute equation (4-10) into (4-13) the numerator is

$$1 - \exp\left(-i 2\pi (l - j) \right) = 0 \tag{4-14}$$

So all the $l \neq j$ terms sum to zero and we can see that equation (4-7) is correct.

Fourier Transform Pairs

If we replace c_k with Y_k in equations (4-3) and (4-7) we get a Discrete Fourier transform pair in the conventional notation

$$\begin{aligned} Y_k &= \frac{1}{N} \sum_{j=0}^{N-1} y_j \exp\left(-\frac{i 2\pi jk}{N} \right) \\ y_j &= \sum_{k=0}^{N-1} Y_k \exp\left(\frac{i 2\pi jk}{N} \right) \end{aligned} \tag{4-15}$$

Now as for the continuous Fourier Transform, there is an ambiguity in terms of the multiplying term in front of the inverse and discrete Fourier transform. In *MATLAB* the DFT pair is defined by

$$Y_k = \sum_{j=0}^{N-1} y_j \exp\left(-\frac{i2\pi jk}{N}\right)$$
$$y_j = \frac{1}{N}\sum_{k=0}^{N-1} Y_k \exp\left(\frac{i2\pi jk}{N}\right) \qquad \textit{MATLAB} \text{ (4-16)}$$

Thus, compared with equation (5-15) the terms Y_k are N times as big and the inverse DFT requires the introduction of a $1/N$ term. One could also write a symmetric form

$$Y_k = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} y_j \exp\left(-\frac{2\pi ikj}{N}\right)$$
$$y_j = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} Y_k \exp\left(\frac{2\pi ikj}{N}\right) \qquad (4\text{-}17)$$

but I am not aware of anyone who uses this convention.

The Fast Fourier Transform (FFT).

From the expression for the discrete Fourier transform shown in equation (4-16), it is clear that calculating each term for a real time series requires N multiplications of a real number and a complex exponential or $2N$ multiplications. Now all, N terms require $2N^2$ operations which is reduced to N^2 when we remember the symmetry properties of the Fourier transform.

However, it turns out that the Fourier transform can be computed in only $N \log N$ operations when the number of samples is a power of 2 (i.e., $N = 2^M$ where M is an integer). We will not cover the details, but we can explain it briefly as follows. The forward transform of equation (4-16) can be written as two series over the odd and even terms to yield

$$\begin{aligned} Y_k &= \sum_{j=0}^{N-1} y_j \exp\left(-\frac{2\pi ikj}{N}\right) \\ &= \sum_{j=0}^{N/2-1} y_{2j} \exp\left(-\frac{2\pi ik2j}{N}\right) + \sum_{j=0}^{N/2-1} y_{2j+1} \exp\left(-\frac{2\pi ik(2j+1)}{N}\right) \\ &= \sum_{j=0}^{N/2-1} y_{2j} \exp\left(-\frac{2\pi ikj}{N/2}\right) + \exp\left(-\frac{2\pi ik}{N}\right) \sum_{j=0}^{N/2-1} y_{2j+1} \exp\left(-\frac{2\pi ikj}{N/2}\right) \\ &= P_k + \exp\left(-\frac{2\pi ik}{N}\right) Q_k \end{aligned} \qquad (4\text{-}18)$$

We have expressed Y_k in terms of a sum of two Fourier transforms P_k and Q_k, each for a time series of $N/2$ samples. The reason why this reduces the number of calculations is that because the periodicity of the Fourier transform we can write

$$P_k = P_{k+\frac{N}{2}} \qquad (4\text{-}19)$$

so the calculation of all the terms in P and Q requires only half the number of calculations required for all the terms in G. Now if N is a power of 2 we can keep on subdividing the Fourier series until we are left with series of two-point sequences. The book keeping becomes fairly complex to express, but the net result is that the number of operations is reduced to $N \log N$, an enormous computational saving for long time series. Without the FFT, Fourier transforms would be much less prevalent in computational data analysis.