Fourier transform of real discrete data

Today we will discuss how to apply Fourier transform to real data, which is always sampled at **discrete** times and is **finite** in duration. Note: Fourier analysis also used in image processing. Images have **finite** resolution. Instead of discrete time steps, one has discrete points in space (pixels).

- Real data is sampled in finite time steps Δt: $y(t) \to y(m\Delta t)$, where $m = 0, 1, 2, \ldots N-1$ and N is the number of points. Only consider equally spaced time steps.
- Data is of finite duration: $T = N\Delta t$

Notation:

- Time steps $t_m \equiv m\Delta t$
- Function/Signal at discrete times: $y(m\Delta t) = y(t_m) \equiv y_m$

Notes

How to discretize the Fourier transform

Since our data is finite, cannot evaluate the Fourier integral below

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \qquad (1)$$

Task: Compute a measure from finite data sets sampled on discrete time steps that *resembles/approximates* the Fourier transform.

In the following we will develop the **discrete Fourier transform**.

Notes

Developing the discrete Fourier transform

Let's go back to the complex Fourier series for periodic functions with period T. Consider the Fourier representation of $y(t)$ on an interval from 0 to T. (Before, in lecture 3 we considered $-T/2$ to $T/2$)

$$y(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\omega t}, \text{ where}$$

$$c_n = \frac{1}{T} \int_0^T y(t) e^{-i2\pi nt/T} dt = \frac{1}{T} \int_0^T y(t) e^{-in\Delta\omega t} dt,$$

where the discrete frequencies of the Fourier series representation are

$$n\Delta\omega \equiv n\frac{2\pi}{T}$$

Notes

Developing the discrete Fourier transform

Let's approximate the integral for c_n by a summation: $\int_0^T f(x)dx \approx \Delta x \sum_{i=0}^{N-1} f(i\Delta x)$.

$$\begin{aligned} c_n = \frac{1}{T} \int_0^T y(t) e^{-in\Delta\omega t} dt &\approx \frac{\Delta t}{T} \sum_{m=0}^{N-1} y(m\Delta t) e^{-in\Delta\omega m\Delta t} \\ &\approx \frac{1}{N} \underbrace{\sum_{m=0}^{N-1} y(m\Delta t) e^{-i2\pi nm/N}}_{Y(n\Delta\omega)} \end{aligned}$$

where we used $\Delta t = T/N$ and $\Delta\omega = \frac{2\pi}{T}$. Note that Δt cancels out in the exponential.

Notes

The discrete Fourier transform

We define the discrete Fourier transform (DFT) as:

$$Y(n\Delta\omega) = \sum_{m=0}^{N-1} y(m\Delta t) e^{-i2\pi nm/N},$$

This can be written more concisely as

$$Y_n = \sum_{m=0}^{N-1} y_m e^{-i2\pi nm/N}, \qquad (2)$$

since $y(m\Delta t) = y(t_m) = y_m$.

Still need the inverse discrete transform. Want it to be exact, such that we recover the y_n's from the Y_m's. Next we will find the inverse transform by using orthogonality.

Notes

The inverse discrete Fourier transform

Consider the sum

$$S_N = \sum_{k=0}^{N-1} e^{ik\alpha}$$

If $\alpha = 0$ then $S_N = N$, since every term is 1. The sum above is just a geometric series:

$$\sum_{k=0}^{N-1} r^k = r^0 + r^1 + \ldots + r^{N-1} = \frac{1 - r^N}{1 - r}$$

Therefore,

$$S_N = \sum_{k=0}^{N-1} e^{ik\alpha} = \frac{1 - e^{i\alpha N}}{1 - e^{i\alpha}}$$

Notes

The inverse discrete Fourier transform

Want S_N to be zero for $\alpha \neq 0$. This means

$$e^{i\alpha N} = 1$$

so

$$\alpha = 2\pi l/N, \text{ where } l \text{ is integer}$$

Therefore,

$$S_N = \sum_{k=0}^{N-1} e^{i2\pi kl/N} = \begin{cases} N, & l = 0 \\ 0, & l \neq 0 \end{cases}$$

Notes

The inverse discrete Fourier transform

Now, let's express integer l as difference between two integers m and n ($l = m - n$) to obtain orthogonality relation:

$$\sum_{k=0}^{N-1} e^{i2\pi km/N} e^{-i2\pi kn/N} = N\delta_{m,n} \qquad (3)$$

Let's return to the discrete Fourier transform (eqn. (2))

$$Y_n = \sum_{m=0}^{N-1} y_m e^{-i2\pi nm/N},$$

and multiply both sides with $e^{i2\pi kn/N}$ and sum over n.

Notes

The inverse discrete Fourier transform

$$\begin{aligned}\sum_{n=0}^{N-1} Y_n e^{i2\pi kn/N} &= \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} y_m e^{-i2\pi nm/N} e^{i2\pi kn/N} \\ &= \sum_{m=0}^{N-1} y_m \sum_{n=0}^{N-1} e^{-i2\pi nm/N} e^{i2\pi kn/N} \\ &= \sum_{m=0}^{N-1} y_m N\delta_{k,m} \\ &= N y_k,\end{aligned}$$

where we used the orthogonality of the sums (eqn. (3))

Notes

The inverse discrete Fourier transform

Therefore, we can identify the inverse discrete Fourier transform

$$Y_n = \sum_{m=0}^{N-1} y_m e^{-i2\pi nm/N} \quad , \quad \text{DFT} \tag{4}$$

$$y_m = \frac{1}{N}\sum_{n=0}^{N-1} Y_n e^{i2\pi nm/N} \quad , \quad \text{inverse DFT} \tag{5}$$

Similar to Fourier transform, prefactors and sign convention in the exponential can vary in different books and code libraries. The product of the prefactors always has to be $1/N$.

Notes

Comments on the DFT

- Discrete Fourier transform can be thought of as a truncated complex Fourier series. This means higher frequency components are missing. This is important when dealing with real signals (more later) → How well does the DFT approximate the Fourier transform ?
- Even though it arose from an approximation of the Fourier series, mathematically it is well defined (orthogonality of the sums).

Let's look at the computational implementation

Notes

Fast Fourier Transform

- To transform N data points, need to compute N summations over order N points. Therefore, computation time goes as N^2. For higher dimensions d, it goes as N^{2d}.
- The fast Fourier transform (Cooley and Tukey 1965), can reduce the computational effort dramatically: $N^2 \rightarrow N log_2 N$. e.g. for $N = 2^{16} = 65536$ need $\approx 10^6$ instead of $\approx 4 \cdot 10^9$ evaluations.
- The FFT is the most important and widely used algorithm!

Notes

The Fast Fourier Transform

Algorithm is based on regrouping terms in the summation. Let's assume, that N is a power of 2. Now consider the discrete transform and decompose into even and odd terms:

$$\begin{aligned} g(n\Delta\omega) &= \sum_{m=0}^{N-1} f(m\Delta t)e^{-i2\pi mn/N} \\ &= \sum_{m=0,\text{even}}^{N-2} f(m\Delta t)e^{-i2\pi mn/N} + \sum_{m=1,\text{odd}}^{N-1} f(m\Delta t)e^{-i2\pi mn/N} \\ &= \sum_{j=0}^{N/2-1} f(2j\Delta t)e^{-i2\pi(2j)n/N} \\ &\quad + \sum_{j=0}^{N/2-1} f((2j+1)\Delta t)e^{-i2\pi(2j+1)n/N} \end{aligned}$$

Notes

The Fast Fourier Transform

The two sums can be rewritten as two DFT's with $N/2$ points that run over even and odd numbered points, respectively.

$$\begin{aligned} g(n\Delta\omega) &= \sum_{j=0}^{N/2-1} f(2j\Delta t)e^{-i2\pi jn/(N/2)} \\ &\quad + e^{-i2\pi n/N} \sum_{j=0}^{N/2-1} f((2j+1)\Delta t)e^{-i2\pi jn/(N/2)} \\ &= \underbrace{g_{even}(n\Delta\omega)}_{N/2 \text{ points}} + e^{-i2\pi n/N} \underbrace{g_{odd}(n\Delta\omega)}_{N/2 \text{ points}} \end{aligned}$$

Notes

The Fast Fourier Transform

Instead of N^2 operations, we now have $2(N/2)^2$ operations. Continue decomposition into even and odd points. If $N = 2^k$, then after k steps, there will be N transforms each containing just one point. Therefore, need $N \cdot k = N \log_2 N$ operations.

FFT

The FFT requires $N = 2^k$ and the computing time goes as $N \log_2 N$

Actual implementation a bit more tricky (see Giordano, Nakanishi Appendix C3).

Notes

Properties of DFT - Periodicity

The DFT is **periodic** in time and frequency space. Shift indices by N:

$$Y_{n+N} = \sum_{m=0}^{N-1} y_m e^{-i2\pi(n+N)m/N} = \sum_{m=0}^{N-1} y_m e^{-i2\pi nm/N} \underbrace{e^{-i2\pi m}}_{=1} = Y_n$$

$$y_{m+N} = \frac{1}{N} \sum_{n=0}^{N-1} Y_n e^{i2\pi(m+N)n/N} = \frac{1}{N} \sum_{n=0}^{N-1} Y_n e^{i2\pi nm/N} \underbrace{e^{i2\pi n}}_{=1} = y_m$$

If one continues summation beyond $(N-1)$, then Y_n and y_m repeat.

Notes

Properties of DFT - Parseval's theorem

Using the orthogonality of the sums (eqn. (3)) one can show

$$\sum_{m=0}^{N-1} x_m y_m^* = \frac{1}{N} \sum_{n=0}^{N-1} X_n Y_n^* \tag{6}$$

analogous to what we had for Fourier transform. If $x = y$, then we recover Parseval's theorem:

$$\sum_{m=0}^{N-1} |x_m|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |X_n|^2 \tag{7}$$

Notes

Properties of DFT - Convolution

Using the orthogonality of the sums (eqn. (3)) one can show the Convolution theorem for the DFT:

$$(x \otimes y)_n = \sum_{l=0}^{N-1} x_l \underbrace{\left(\sum_{p=-\infty}^{\infty} y_{n-l-pN} \right)}_{\text{periodic extension of } y_{n-l}} = \underbrace{\frac{1}{N} \sum_{k=0}^{N-1} X_k Y_k e^{i2\pi kn/N}}_{\text{inverse DFT of } X \cdot Y} \tag{8}$$

where p is an integer.
Note that Convolution needs N^2 operations. By performing it in frequency domain using FFT it is much faster! However, need to be careful since convolution "wraps around" due to periodicity.

Notes

Properties of DFT - Correlation

Using the orthogonality of the sums (eqn. (3)) one can show the Correlation theorem for the DFT (Wiener-Khinchin):

$$(x \odot y)_n = \sum_{l=0}^{N-1} x_l^* \underbrace{\left(\sum_{p=-\infty}^{\infty} y_{n+l-pN} \right)}_{\text{periodic extension of } y_{n+l}} = \underbrace{\frac{1}{N} \sum_{k=0}^{N-1} X_k^* Y_k e^{i2\pi kn/N}}_{\text{inverse DFT of } X^* \cdot Y} \tag{9}$$

Again, get performance increase by doing correlation in frequency space using FFT, especially for higher dimensions.

Notes

Properties of DFT - Shifting

Analogous to the Fourier transform, shifting the time axis or frequency axis by l gives rise to an additional phase factor:

$$\begin{aligned} DFT[y_{m-l}]_k &= Y_k e^{-i2\pi kl/N} \\ DFT[y_m e^{i2\pi ml/N}]_k &= Y_{k-l} \end{aligned}$$

Notes

What about the frequency cutoff?

Since we truncated the complex Fourier series to arrive at the DFT, the DFT has a frequency limit!

For N data points, the discrete frequencies are $\omega_n = \frac{n2\pi}{T} = \frac{n2\pi}{N\Delta t}$ or $f_n = \frac{n}{N\Delta t}$, where $n = 0 \ldots (N-1)$.

However, $f_{N-1} = \frac{N-1}{N\Delta t}$ is **NOT** the highest frequency that the DFT can resolve!!
To illustrate that consider the symmetry of the DFT.

Notes

DFT for a real data points

The DFT is given by

$$Y_n = \sum_{m=0}^{N-1} y_m e^{-i2\pi nm/N}$$

Let's assume that the y_m's are real ($y_m^* = y_m$). Then we can write

$$\begin{aligned} Y_{N/2-n} &= \sum_{m=0}^{N-1} y_m e^{-i2\pi m(N/2-n)/N} = \sum_{m=0}^{N-1} y_m e^{-i\pi m} e^{i2\pi mn/N} \\ Y^*_{N/2+n} &= \sum_{m=0}^{N-1} y_m e^{i2\pi m(N/2+n)/N} = \sum_{m=0}^{N-1} y_m e^{i\pi m} e^{i2\pi mn/N} \end{aligned}$$

but $e^{-i\pi m} = e^{i\pi m} = \cos(\pi m)$. Therefore, $Y_{N/2-n} = Y^*_{N/2+n}$.

Notes

Nyquist frequency

For a real signal, half of the points are redundant because of the symmetry around $f_{N/2} = \frac{N}{2}\frac{1}{N\Delta t} = \frac{1}{2\Delta t}$. It's enough to compute Y_n from $n = 0 \ldots (N/2-1)$ for even N.
The highest independent frequency is therefore:

Nyquist frequency

$$f_{\text{Nyquist}} \frac{N/2-1}{N\Delta t} \approx \frac{1}{2\Delta t}$$

This is called the **Nyquist frequency**.

Notes

Nyquist frequency and sampling theorem

The frequencies $f \geq f_{\text{Nyquist}}$ are actually the negative frequencies! (Remember that a simple sin/cos function was represented by a one positive and one negative frequency component in the Fourier transform/complex Fourier series. It is the same for the DFT).

Shannon-Nyquist sampling theorem

A function $f(t)$ that contains frequencies **less** than f_{max} is completely determined if it is sampled at points that are $\Delta t = \frac{1}{2f_{max}}$ apart.

Notes

Nyquist frequency and sampling theorem

Example: Human ear can hear up to $20,000$ Hz. In order to capture the whole audio spectrum, need a sampling rate that is at least twice as large: $2 \cdot 20,000 = 40,000$ Hz. CD's use $44,100$ Hz, for example.

Notes

Notes

Notes

Notes