Lecture 8 - Signal processing with DFT

Goal today is to demo how discrete FT (DFT) used for signal processing & function approximation.

Recall FS:

$$f:[0,2L]\longrightarrow\mathbb{R},\quad c_k := \frac{1}{2L}\int_0^{2L} f(x)\exp\left(-\frac{i\pi kx}{L}\right)dx$$

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \exp\left(\frac{i\pi kx}{L}\right),\quad c_k\in\mathbb{C}.$$

Simply truncate the sum

$$f(x) \approx \sum_{k=-N/2}^{N/2-1} c_k \exp(-ik\pi x/L),\quad c_k\in\mathbb{C}$$

can be implemented

Henceforth, we write $\hat{f} := (c_{-N/2}, \dots, c_{N/2-1}) \in \mathbb{C}^N$ & call it the DFT of $f(x)$ in the parlance of continuing FS from last lecture.

The fast Fourier transform (FFT) & its inverse (IFFT) are algorithms that compute $\hat{f}$ for a function $f(x)$ given at a set of equidistance points.

More precisely, consider N points

$x_0 = 0, x_1 = \delta x, x_2 = 2\delta x, \dots, x_{N-1} = (N-1)\delta x$

with increments $\delta x = \frac{2L}{N-1}$ (we also call this a uniform grid or mesh.)

input		output
$f(x_0), \dots, f(x_{N-1})$	FFT → / ← IFFT	$c_{-N/2}, \dots, c_{N/2-1}$

3.1 From FFT to DFT

FFT is one of the most important algorithms in modern computing & signal processing. It was designed back in the 60's by Cooley & Tukey.

FFT has two important aspects that make it so powerful: ① it cost $\Theta(N \log N)$ operations to compute when N is # of samples in signal. ② It is very accurate.

What FFT computes:

Consider function $f:[0, 2L] \to \mathbb{R}$ & let $f_n := f(x_n)$ for $n = 0, \dots, N-1$ as before.

Then FFT returns coefficients

$$\tilde{c}_k = \sum_{n=0}^{N-1} f_n \exp\left(-2\pi i \frac{nk}{N}\right)$$

This is the Numpy convention! MATLAB does the same. But other software might normalize differently. This can be very confusing.

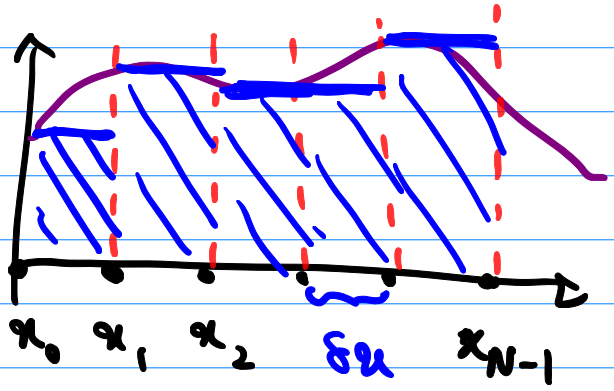
Compare this to FS!

$$\tilde{c}_k = \sum_{n=0}^{N-1} f_n \exp\left(-2\pi i \frac{kn}{N}\right)$$

$$c_k = \frac{1}{2L} \int_0^{2L} f(x) \exp\left(-\pi i \frac{kx}{L}\right) dx$$

approx integral with a Riemann sum.

Break $[0, 2L]$ into N intervals of length $\delta x = \frac{2L}{N}$



$$c_k \approx \frac{1}{2L} \sum_{n=0}^{N-1} f(x_n) \exp\left(-i\pi \frac{k x_n}{L}\right) \delta x$$

$$= \frac{1}{2L} \sum_{n=0}^{N-1} f_n \exp\left(-i\pi \frac{k 2L n}{NL}\right) \frac{2L}{N}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} f_n \exp\left(-2\pi i \frac{kn}{N}\right) = \frac{1}{N} \tilde{c}_k$$

So, we need to scale the output of FFT by $\frac{1}{N}$ to get the correct vals that approx the FS.

Note: In general you don't need this in your codes if you primarily work with the discrete transform & its inverse.

3.2 Fitting & function approximation with FFT

See demo notebook for rest of lecture.