- Fast Fourier Transform is a method of calculating

FFT
$\downarrow$
$O(n(\log n))$
complexity

$\boxed{\text{Discrete Fourier Transform}}$ - $O(n^2)$ Computational complexity

Matrix *

$$\begin{bmatrix} F_0 \\ f_1 \\ f_2 \\ F_3 \\ F_4 \\ F_5 \\ \vdots \end{bmatrix} \xrightarrow{\text{DFT}} \begin{bmatrix} \hat{F}_0 \\ \hat{F}_1 \\ \hat{F}_2 \\ \hat{F}_3 \\ \hat{F}_4 \\ \vdots \end{bmatrix}$$

Fourier Transform

$$\hat{f}_k = \sum_{J=0}^{n-1} F_J \, e^{-\frac{i 2\pi J k}{n}}$$

Inverse Fourier transform

$$F_k = \sum_{J=0}^{n-1} \left(\hat{F}_J \, e^{\frac{i 2\pi J k}{n}} \right) \frac{1}{n}$$

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j \, e^{\frac{-2\pi i j k}{n}}$$

$$\omega_n = e^{\frac{-2\pi i}{n}}$$

computationally complex

$$\begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \\ \hat{f}_4 \\ \vdots \\ \hat{f}_n \end{bmatrix} = \begin{array}{c} k=0 \\ k=1 \\ k=2 \\ \vdots \\ k=n \end{array} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & & \ddots & & & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} * \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_n \end{bmatrix}$$

DFT matrix

$\therefore$ FFT can be used for

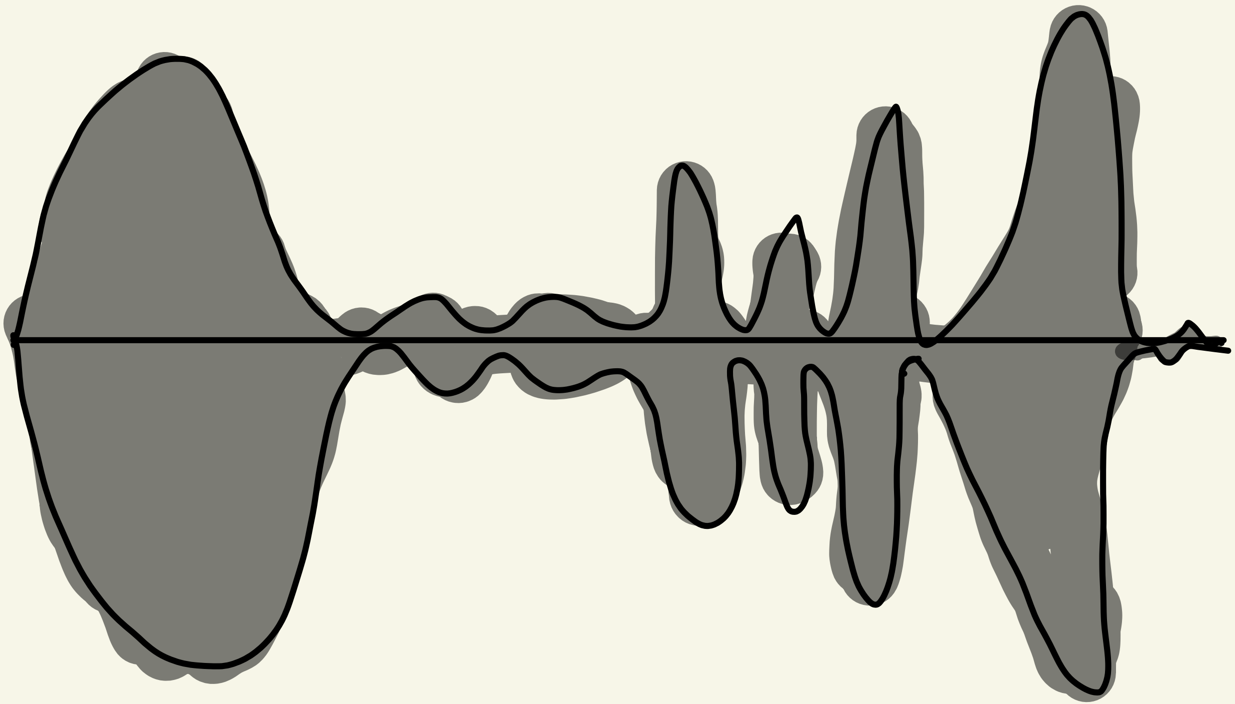
- Derivatives – PDE
- Denoise data
- Analysis of data
- Audio / Image Compressions

We break the DFT matrix into smaller components for Computational efficiency

5 sec audio signal

Sampling Rate = 44,100 Hz/sec

Total no of samples = 5 * 44,100

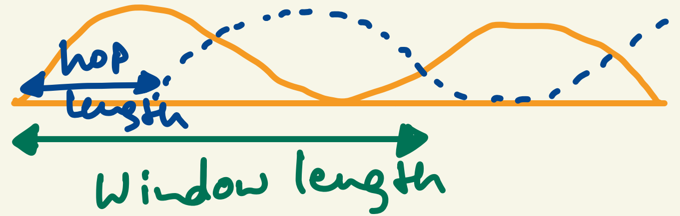


according to Nyquist-Shannon theorem

Max freq that can be accurately represented is Half the sampling rate

STFT

n-fft
adding zeros at the end of each windowed frame



increases the no of frequency bins