Computer Project: Fast Fourier Transform

Zexi Fan, 2200010816

21st May 2025

1 Problem Setting

1.1 Problem One: Non-periodic Convolution Using FFT

Let x and h be two non-periodic vectors with compact support, whose components are defined as

$$x_n = \begin{cases} \sin\left(\frac{n}{2}\right), & n = 1, 2, \ldots, M-1, \\ 0, & \text{otherwise}, \end{cases}$$

$$h_n = \begin{cases} \exp\left(\frac{1}{n}\right), & n = 1, 2, \ldots, Q-1, \\ 0, & \text{otherwise}, \end{cases}$$

with $Q \leq M$. Taking $Q = 200$ and $M = 500$, compute the non-periodic convolution

$$y_n = \sum_{q=0}^{Q-1} h_q \, x_{n-q},$$

by means of the fast Fourier transform (FFT) and compare the result with the direct convolution computed from its definition.

1.2 Problem Two: Frequency Filtering via FFT

Define the function

$$f(t) = e^{-t^2/10}\left[\sin(2t) + 2\cos(4t) + 0.4\sin t \, \sin(50t)\right].$$

Sample f at points

$$y_k = f\left(\frac{2k\pi}{256}\right), \quad k = 0, 1, \ldots, 256,$$

and compute its discrete Fourier transform coefficients $\hat{y}_k$ for $k = 0, 1, \ldots, 256$ using the FFT. Exploiting the conjugate symmetry

$$y_{256-k} = \overline{y_k},$$

one identifies the low-frequency coefficients $\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_m$ and $\hat{y}_{256-m}, \ldots, \hat{y}_{256}$ (for a suitably small m). By setting

$$\hat{y}'_k = \begin{cases} \hat{y}_k, & k \le m \text{ or } k \ge 256 - m, \\ 0, & \text{otherwise}, \end{cases} \qquad m = 6,$$

filter out the high-frequency components. Then apply the inverse FFT to $\{\hat{y}'_k\}$ to obtain the filtered samples y'_k. Plot and compare y_k and y'_k for various values of m.

2 Algorithms

2.1 Discrete Fourier Transform (DFT)

Let $\mathbf{y} = (y_0, y_1, \ldots, y_{N-1})^\top$. Its discrete Fourier transform (DFT) $\hat{\mathbf{y}} = \mathbf{c} = (c_0, c_1, \ldots, c_{N-1})^\top$ is defined by

$$c_k = \sum_{j=0}^{N-1} y_j \, e^{-2\pi i jk/N}, \quad k = 0, 1, \ldots, N-1,$$

where i is the imaginary unit and $\omega = e^{-2\pi i/N}$ is a primitive Nth root of unity. The inverse DFT recovers $\mathbf{y}$ from $\mathbf{c}$ by

$$y_j = \frac{1}{N} \sum_{k=0}^{N-1} c_k \, e^{2\pi i jk/N}, \quad j = 0, 1, \ldots, N-1.$$

If $N = 2^m$, the vector $\mathbf{y}$ can be split into its even and odd indexed components:

$$\mathbf{y}_{\text{even}} = (y_0, y_2, \ldots, y_{N-2})^\top, \quad \mathbf{y}_{\text{odd}} = (y_1, y_3, \ldots, y_{N-1})^\top.$$

Compute the DFTs of these length-$N/2$ vectors, yielding $\mathbf{c}_{\text{even}}(k)$ and $\mathbf{c}_{\text{odd}}(k)$. Then

$$c_k = c_{\text{even}}(k) + \omega^k \, c_{\text{odd}}(k), \quad c_{k+N/2} = c_{\text{even}}(k) - \omega^k \, c_{\text{odd}}(k),$$

for $k = 0, 1, \ldots, N/2 - 1$. Denote by M_N and A_N the numbers of complex multiplications and additions needed for an N-point FFT. One obtains the recurrences

$$\begin{aligned} M_{2^k} &= 2M_{2^{k-1}} + 2^{k-1}, \quad M_1 = 0, \\ A_{2^k} &= 2A_{2^{k-1}} + 2^{k-1}, \quad A_1 = 0, \end{aligned}$$

which yield

$$M_N = \tfrac{1}{2} N \log_2 N, \quad A_N = N \log_2 N,$$

so that the FFT requires only $O(N \log_2 N)$ operations.

2.2 Computing Convolution via DFT

For two vectors $\mathbf{x}$ and $\mathbf{h}$ of lengths M and Q, respectively, their convolution

$$y_n = \sum_{q=0}^{Q-1} h_q \, x_{n-q}$$

is obtained by zero-padding to the same length $N \geq M + Q - 1$, computing their DFTs $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{h}}$, forming the pointwise product

$$\widehat{\mathbf{y}} = \widehat{\mathbf{x}} \odot \widehat{\mathbf{h}},$$

and then applying the inverse DFT:

$$\mathbf{y} = \mathrm{IDFT}(\widehat{\mathbf{y}}).$$

This reduces the naive $O(MQ)$ multiplications and additions per output sample to $O(N \log_2 N) = O((M + Q)\log(M + Q))$ overall.

3 Result Analysis

3.1 Problem One: Non-periodic Convolution Using FFT

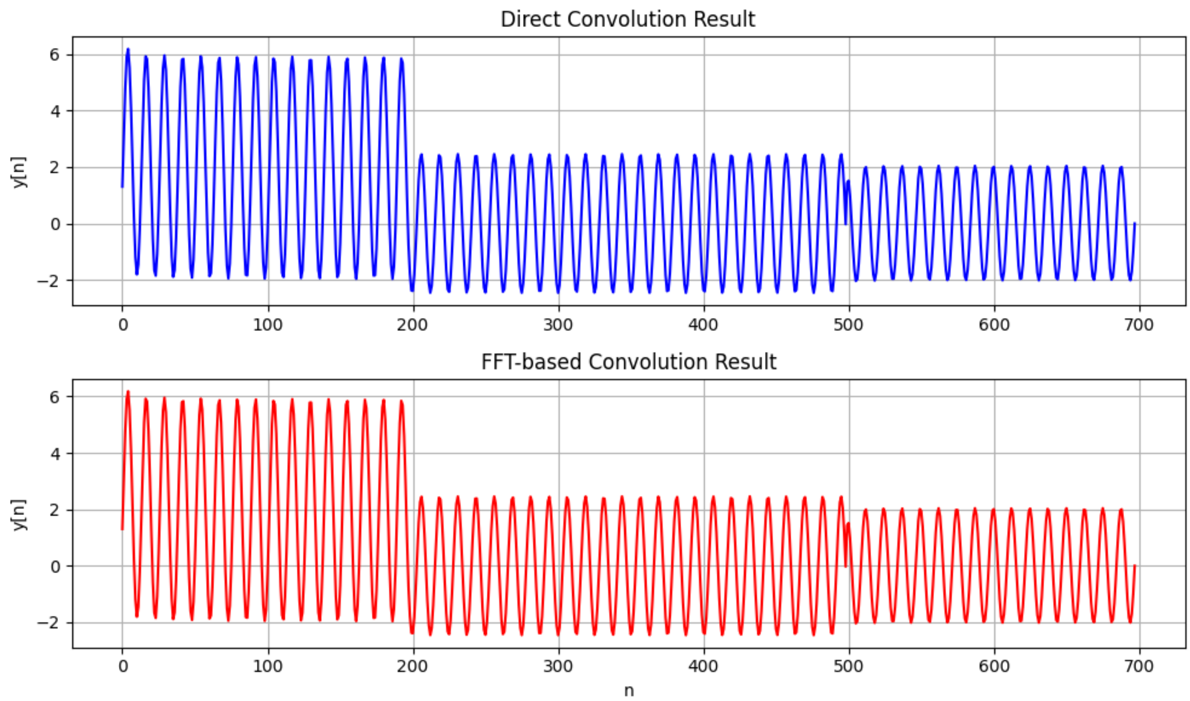

Figure 1: Original Signals v.s. Filtered Signals

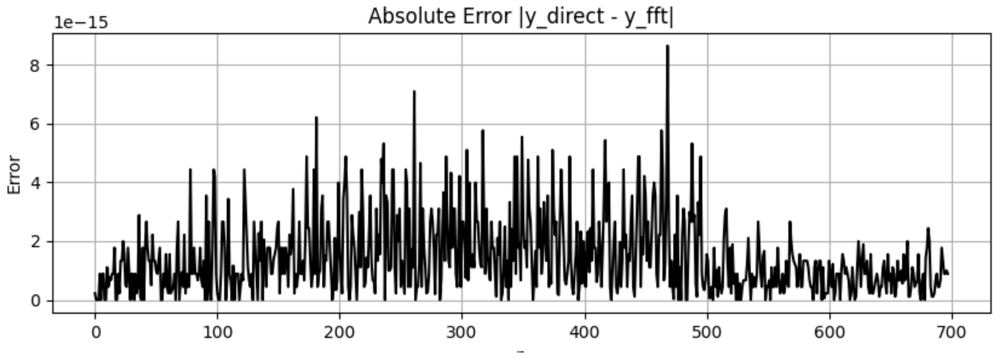

Figure 2: Signals Differences

As we can see from the figures and the former descriptions, the algorithm significantly reduces the computational cost and preserves the signals very accurately, leading to a 10^{-15} magnitude absolute error.

3.2 Problem Two: Frequency Filtering via FFT

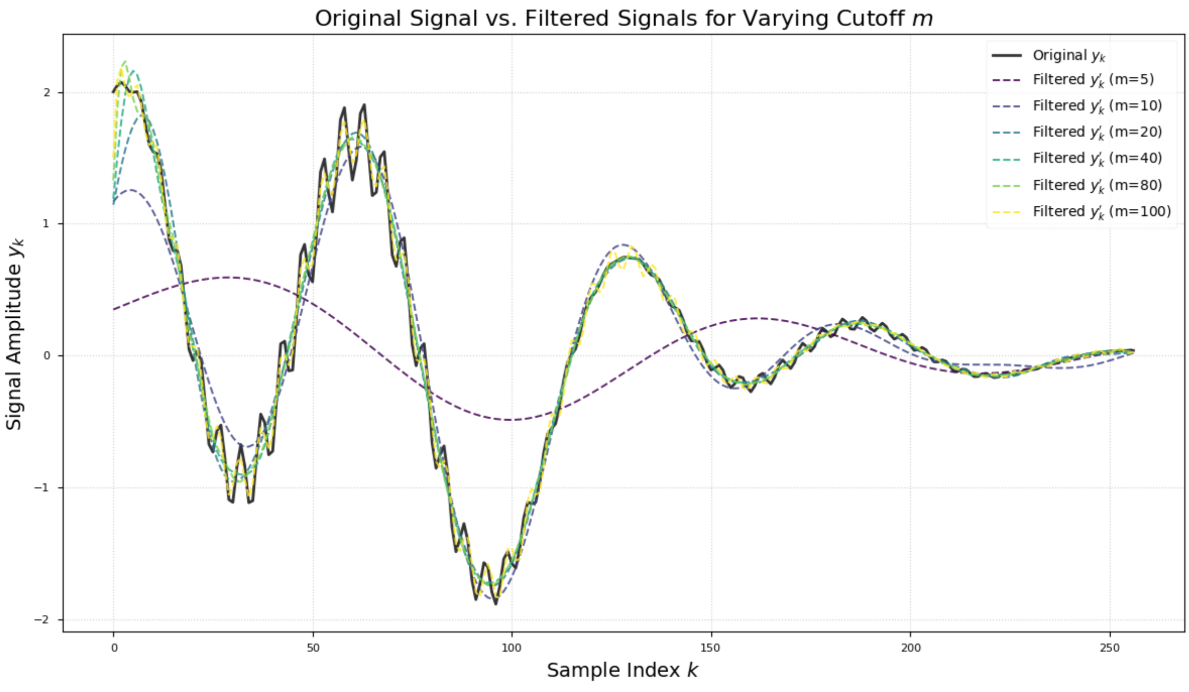

Figure 3: Original Signals v.s. Filtered Signals

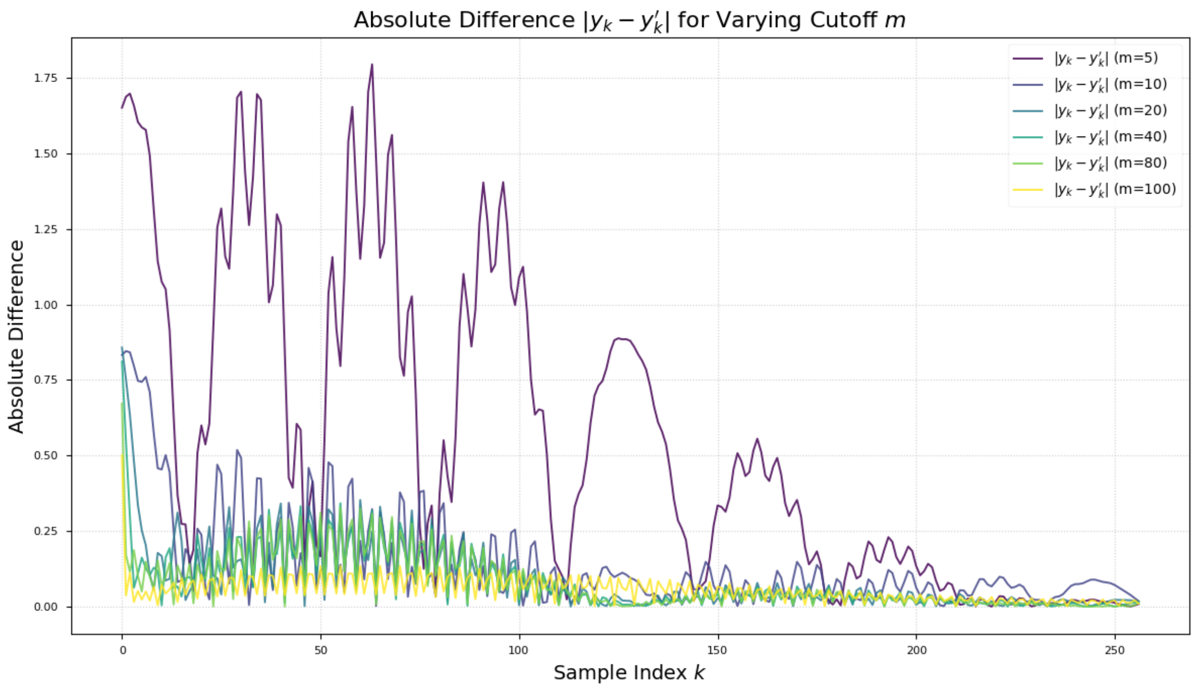

Figure 4: Signals Differences

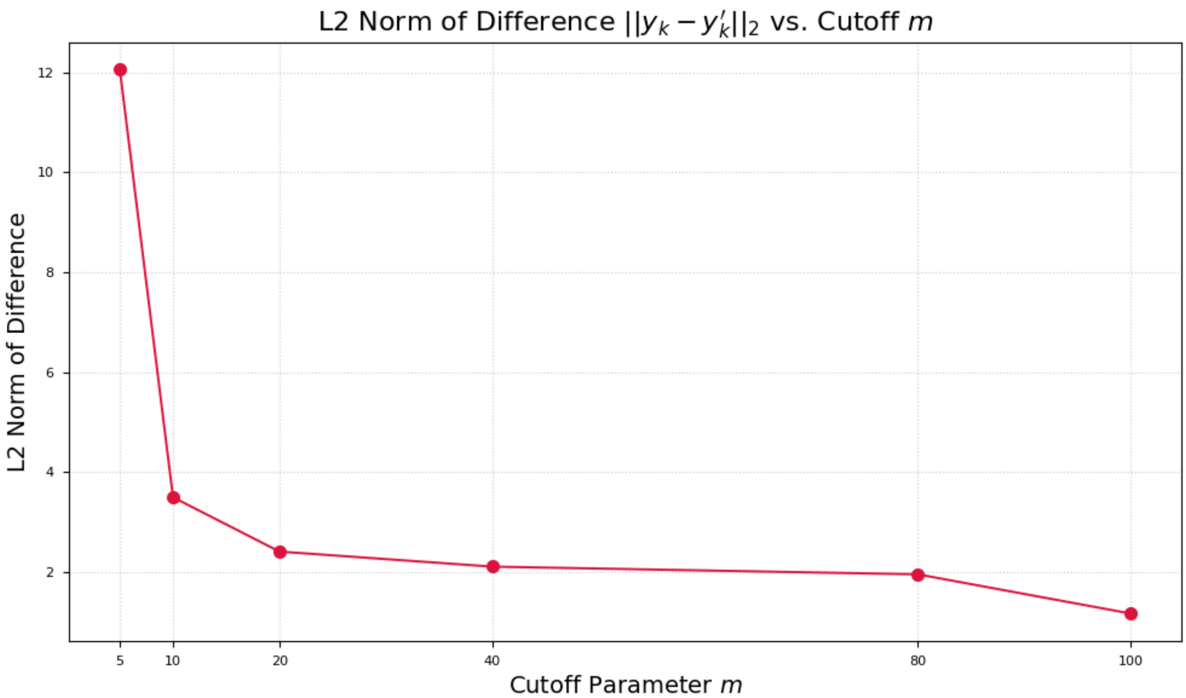

Figure 5: L_2 Norm of Signals Differences

From the figures above, we conclude the followings:

1. **Effect of Removing High-Frequency Components:**

- For large cutoff indices, few high-frequency coefficients are zeroed, so the original signal characteristics are largely preserved and the overall smoothness changes minimally.

- As m decreases, an increasing number of high-frequency components are removed, yielding a progressively smoother waveform and the disappearance of rapid oscillations.

- For very small m values, almost all high-frequency content is filtered out, leaving only the low-frequency envelope. At this stage, the signal amplitude variations are greatly diminished and most fine details are lost.

2. **Signal Smoothness:**

- A smaller m produces a smoother signal because high-frequency terms—responsible for rapid fluctuations—are removed.

- For very small m, the filtered signal approaches a near-constant or slowly varying trend with negligible fast oscillations.

3. **Preservation of Signal Features:**

- With large m, the original oscillatory patterns, peaks, and troughs remain intact.

- With small m, only the low-frequency trend persists, and many high-frequency features are eliminated, resulting in loss of detail.

High-frequency components often correspond to noise, so truncating them acts as a denoising mechanism: as m decreases, noise is progressively removed and the signal becomes cleaner. These observations show that an appropriate choice of m can balance between smoothness and retention of original features. A large m is suitable when preserving detail is important, while a small m is best for smoothing and noise suppression.

References