Chris Jiang 400322193
Sohel Saini 400327615

Discrete Fourier Transform

The Inverse Discrete Fourier Transform (IDFT) function reconstructs the original signal from its frequency domain representation, allowing for a complete round-trip Fourier analysis. A unit test for Fourier and Inverse Fourier Transform (fourierUnitTest) was improved to verify the accuracy of the IDFT by comparing the reconstructed signal with the original input, using a specified tolerance (EPSILON). Additionally, the execution time measurement function (measureExecutionTimes) was introduced to evaluate the computational efficiency of both DFT and IDFT for varying sample sizes, providing performance benchmarks. The main() function was updated to execute these performance tests, ensuring the robustness of the implementation. These enhancements make the program more comprehensive, enabling correctness validation and execution time profiling for Fourier Transform operations.

Digital Filter Design

A random sine wave generator (generateRandomSin) was implemented to create sine waves with random frequencies (25-75 Hz) and amplitudes (3-6), replacing the manually defined sine waves. A multiplication function (multiplySines) was added to perform amplitude modulation by multiplying two sine waves element-wise. The impulse response function (impulseResponseLPF) was significantly improved by implementing a sinc-based low-pass filter (LPF) with a Hann window, ensuring a smooth transition. The finite impulse response (FIR) filter (convolveFIR) was fully implemented using convolution, applying the LPF to the modulated signal. The main() function was updated to integrate these changes, generating two random sine waves, modulating them, determining the higher frequency for filtering, applying FIR filtering, computing the Discrete Fourier Transform (DFT), and saving the data for visualization using GNUplot. These additions make the program more dynamic, efficient, and aligned with the task requirements.

Digital Filtering of Data Streams Divided into Blocks

The impulseResponseLPF and convolveFIR functions were implemented to calculate the impulse response for a low-pass filter and perform convolution, enabling effective audio data filtering. A block-based convolution function, blockConvolveFIR, was introduced to process data in smaller segments, making it more efficient for large datasets or real-time applications. This function uses a state vector to maintain continuity between blocks, ensuring seamless processing. To validate the block convolution approach, unit testing functions like unitTestBlockConvolution were added. These tests compare the block convolution results with the full convolution output, ensuring accuracy and reliability. Together, these enhancements make the filtering process more robust, scalable, and efficient, while providing a way to verify the correctness of the block-based processing logic.