GROUP 1
Chan, Ethan Lester
Dolon, John Michael
Lu, Andre Giancarlo
Teng, Adriel Shanlley

# Integrating Project Proposal: Speeding-up Discrete Fourier Transform and Inverse Discrete Fourier Transform

## Abstract

Discrete Fourier Transform (DFT) is a tool to find the frequency spectrum of a finite-duration signal, by transforming the signal from the time domain to the frequency domain. This transformation technique has many applications in designing filters, manipulating or comparing multiple signals, and other processes in Digital Signal Processing. The Inverse Discrete Fourier Transform reverts signals back from the frequency domain to the time domain. Our project proposal is to implement DFT and IDFT through a program written in C and optimize it using CUDA for faster calculations. The inputs of the program will be a signal (array of floating point values) and the signal frequency. The program will then compute for the DFT and the IDFT of each sample in the signal. The output will input the signal transformed through DFT into the frequency domain, as well as the reverted signal from performing IDFT on the signal in the frequency domain.

## Project Proposal: Description

In digital signal processing, Discrete Fourier Transform (DFT) is an incredibly powerful tool for finding the frequency spectrum of a finite-duration signal (Arar, 2017). Essentially, DFT converts signals from the time domain to the frequency domain without any loss (Sidhu et al, 2023). Given this, it is an important transformation technique that has applications in designing filters, manipulating or comparing multiple signals, and many more processes in digital signal processing. Likewise, the Inverse Discrete Fourier Transform (IDFT) is also an important method to revert signals back from the frequency domain to the time domain. Hence, our plan for our integrating project is to implement DFT and IDFT through a program written in C, and optimize using CUDA for faster calculations with the help of data parallelism.

### A. Inputs

The input to our program will simply be a signal (array of floating point values) and the signal frequency. The process that will be parallelized using SIMT (GPU/CUDA) is the process of computing for the DFT as well as the IDFT of each sample in the signal. The general formula for DFT and IDFT are as follows:

DFT:

$$\boxed{X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}} \quad k = 0, 1, \ldots, N-1}$$

IDFT:

$$\boxed{x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi kn}{N}} \quad n = 0, 1, \ldots, N-1}$$

**B. Existing implementations with the proposed process**

A paper written by Douglas Lyon (2009) showed an existing implementation for DFT and IDFT. The machine used for this was a 1.5 GHz Celeron with 1 GB RAM and a 1.6.11 version of the JVM that was used as the platform for the code. The programming language that was used was Java. The runtime of the DFT as a function of 65,536 array length was 934,375ms, or 93.4375s. The difference of their implementation from our proposed implementation is that ours will be significantly faster since ours will parallelize the computations for the DFT and IDFT such as the summing and multiplying of values with imaginary numbers as exponents.

**C. Plans on how to parallelize your proposed process**

The program starts by allocating each element of our input array to a thread and the program runs on the threads one by one. The program will then split all the threads to blocks where each block will contain 1024 threads. This allows each block to run in parallel with each other, speeding up the process of computing our output array. This results in our program running only 1024 times, but each time will have multiple instances of our computation running at the same time.

**D. Other processes in the project that will not be parallelized**

In the project, the only processes that will not be parallelized are the accepting of inputs and converting of signals into an array.

**E. Output**

The outputs of our project will simply be the input signal transformed through DFT into the frequency domain, as well as the reverted signal from performing IDFT on the signal in the frequency domain.

## Project Proposal: Implementation Platform

The program will be coded using the C programming language with CUDA being integrated into it. However, we will not be using Google Colab to create the project. Instead, we will be using Visual Studio 2022. This is possible because the GPUs of the devices of the group members are all NVIDIA.

## References

Arar, S. (2017, July 20). *An Introduction to the Discrete Fourier Transform.* All About Circuits. Retrieved from https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-discrete-fourier-transform/

Delgutte, B., Greenberg, J. (2005). Chapter 4 - *The Discrete Fourier Transform*. Biomedical Signal and Image Processing. Retrieved from https://web.mit.edu/~gari/teaching/6.555/lectures/ch_DFT.pdf

DFT and IDFT (n.d.). Retrieved from https://www.kth.se/social/upload/50919490f2765474b1890dde/lec2.pdf

Lyon, D. (2009, May). The Discrete Fourier Transform, Part 1. *The Journal of Object Technology 8*(3):17-26. Retrieved from (PDF) The Discrete Fourier Transform, Part 1. (researchgate.net)

Project Nariyuki (n.d.). *How to implement the discrete Fourier transform*. Retrieved June 20, 2024, from https://www.nayuki.io/page/how-to-implement-the-discrete-fourier-transform

Sidhu, T., Bhajla, B., Das, S. (2023). Numerical algorithms for protection and metering devices. *Encyclopedia of Electrical and Electronic Power Engineering* (pp. 45-87). Elsevier. https://doi.org/10.1016/B978-0-12-821204-2.00131-8