Final Project Proposal

Tran Phong Binh*

Department of Computer Science and Information Technology, National Taipei University of Technology, Taipei

May 13, 2021

1 Motivation

Recently I have been studying a linear algebra course by MIT Professor Gilbert Strang, and am intrigued to understand why the Discrete Fourier Transform (DFT) prefers complex exponential function to its real counterpart – sine and cosine, even though they are analytically equivalent. I aim to use this final project to find it out.

2 Goal

My Fourier approximation is defined as:

$$\begin{aligned} f(x) &= a_0 + a_1 cos(\frac{1}{N}2\pi x) + b_1 sin(\frac{1}{N}2\pi x) + \dots \\ &\quad + a_{\frac{N-1}{2}} cos(\frac{\frac{N-1}{2}}{N}2\pi x) + b_{\frac{N-1}{2}} sin(\frac{\frac{N-1}{2}}{N}2\pi x) \\ &= a_0 + \sum_{k=1}^{\frac{N-1}{2}} a_k cos(\frac{k}{N}2\pi x) + b_k sin(\frac{k}{N}2\pi x), \end{aligned} \tag{1}$$

where N is the number of input data; $x, k \in \mathbf{Z}$; $f\colon [0, N-1] \to \mathbf{R}$ is the function that we want to approximate using N orthogonal basis $sin, cos\colon [0, 2\pi] \to \mathbf{R}$; with $a_k, b_k \in \mathbf{R}$ being the co-efficients to be estimated. Here we assume that N

*Student ID: 106590048

is odd (for $\frac{N-1}{2} \in \mathbf{Z}$), and that f is a periodic function with period N (otherwise Fourier Transform is not theoretically correct).

My project is to build a program that computes the co-efficients a_k's and b_k's above given N input data. The algorithm involves only naive multiplications and additions. The goal is to compare the numerical result of this method and that of Fast Fourier Transform (FFT) which uses complex exponential function to see why the latter is favored in real-world applications.

3 Method

To implement this DFT, f will be represented as an array with N elements, which is our input data. We compute a_k as follow:

$$a_k = \frac{\langle f, g_k \rangle}{\langle g_k, g_k \rangle}, \tag{2}$$

where $\langle f, g_k \rangle$ denotes the inner product of two functions, and $g_k \colon \mathbf{Z} \to \mathbf{R}$ ($x \to g_k(x)$) denotes the corresponding function of a_k. For $k = 0$,

$$a_0 = \frac{\sum_{x=0}^{N-1} f(x)1}{N} = \frac{1}{N} \sum_{x=0}^{N-1} f(x). \tag{3}$$

Otherwise,

$$a_k = \frac{\sum_{x=0}^{N-1} f(x)cos(\frac{k}{N}2\pi x)}{\frac{N}{2}} = \frac{2}{N} \sum_{x=0}^{N-1} f(x)cos(\frac{k}{N}2\pi x). \tag{4}$$

Similary, b_k can be computed as

$$b_k = \frac{2}{N} \sum_{x=0}^{N-1} f(x)sin(\frac{k}{N}2\pi x). \tag{5}$$

We are going to accelerate the algorithm by spawning N threads computing each co-efficient independently. The arguments of each threads include the input array f and the output array of co-efficients.

4 Expectation

The algorithm would have to deal with roughly $N \approx 50,000$ jobs, but I think it would work fine under 10 seconds. I also expect the numerical result not to differ much from that of FFT.