MATH 603 - Final Assignment

Aiden Taylor - 30092686

December 5, 2024

Contents

1 Problems **3**

2 Problem 1 **4**

3 Problem 2 **7**

1 Problems

1. Write a computer program to implement the Fast Fourier Transform (FFT).

2. Using the FFT, write a computer program to solve numerically the initial-value problem (IVP) for the heat equation,

$$\begin{cases} u_t = u_{xx} & (t,x) \in [0,T] \times [0,1] \\ u(0,x) = u_0(x) & x \in [0,1] \end{cases}.$$

2 Problem 1

To implement the FFT, we should first revist the Continuous Fourier Transform of some function $f(x)$,

$$F(\omega) = \hat{f}(\omega) = \mathcal{F}[f](\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)e^{-i\omega x},$$

where the function can be recovered as

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\omega)e^{i\omega x}.$$

Now, consider discretizing both the original and frequency domains into n equally spaced points, where

$$\begin{cases} \omega_m = 2\pi m/n, & m = 0, 1, \ldots, n-1, \\ x_k = x_0 + k\Delta x, & k = 0, 1, \ldots, n-1, \end{cases}$$

given that $x_0 = 0$ and $\Delta x = L/(n-1)$. Then, if we let $f_k = f(x_k)$ for $k = 0, 1, \ldots, n-1$, we can define the Discrete Fourier Transform (DFT) as

$$f_m^{\#} = \sum_{k=0}^{n-1} f_k e^{-i\omega_m k}, \quad m = 0, 1, \ldots, n-1,$$

where the discretization of the function can be recovered as

$$f_k = \frac{1}{n}\sum_{m=0}^{n-1} f_k^{\#} e^{i\omega_m k}, \quad k = 0, 1, \ldots, n-1,$$

which we call the Inverse DFT (IDFT). Letting $\xi = e^{i2\pi/n}$, we can instead represent the DFT and IDFT respectively as the following two matrix-vector multiplications,

$$\begin{bmatrix} f_0^{\#} \\ f_1^{\#} \\ f_2^{\#} \\ \vdots \\ f_{n-1}^{\#} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \xi^{-1} & \xi^{-2} & \ldots & \xi^{-(n-1)} \\ 1 & \xi^{-2} & \xi^{-4} & \ldots & \xi^{-2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \xi^{-(n-1)} & \xi^{-2(n-1)} & \ldots & \xi^{-(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix},$$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix} = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \xi^{1} & \xi^{2} & \ldots & \xi^{(n-1)} \\ 1 & \xi^{2} & \xi^{4} & \ldots & \xi^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \xi^{(n-1)} & \xi^{2(n-1)} & \ldots & \xi^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} f_0^{\#} \\ f_1^{\#} \\ f_2^{\#} \\ \vdots \\ f_{n-1}^{\#} \end{bmatrix}.$$

Since both the DFT and IDFT are just $n \times n$ systems, it follows that they both have a computational complexity of $\mathcal{O}(n^2)$. From here, the FFT is derived from noticing redundancies in the computation of the DFT, specifically, from noticing that ξ is periodic and that certain powers of ξ are equal. To illustrate this claim, consider the system of equations generated by the DFT when $n = 4$,

$$\begin{cases} f_0^{\#} &= f_0\xi^0 + f_1\xi^0 + f_2\xi^0 + f_3\xi^0 \\ f_1^{\#} &= f_0\xi^0 + f_1\xi^{-1} + f_2\xi^{-2} + f_3\xi^{-3} \\ f_2^{\#} &= f_0\xi^0 + f_1\xi^{-2} + f_2\xi^{-4} + f_3\xi^{-6} \\ f_3^{\#} &= f_0\xi^0 + f_1\xi^{-3} + f_2\xi^{-6} + f_3\xi^{-9}. \end{cases}$$

If we notice that $\xi^0 = \xi^{-4} = 1$, $\xi^{-2} = \xi^{-6} = -1$, $\xi^{-1} = \xi^{-9} = -i$, and $\xi^{-3} = i$, then we can simplify the above system of equations to,

$$\begin{cases} f_0^{\#} &= (f_0 + f_1) + \xi^0(f_2 + f_3) \\ f_1^{\#} &= (f_0 - f_1) + \xi^{-1}(f_2 - f_3) \\ f_2^{\#} &= (f_0 + f_1) + \xi^{-2}(f_2 + f_3) \\ f_3^{\#} &= (f_0 - f_1) + \xi^{-3}(f_2 - f_3), \end{cases}$$

which has reduced the number of computations from 16 multiplications and 12 additions to 4 multiplications and 12 additions (without and doing any caching or precomputing). This idea can be generalized when n is some positive integer power of 2, i.e. $n = 2^\ell$ where $\ell \in \mathbb{Z}^+$, which instead allows us to represent the DFT as

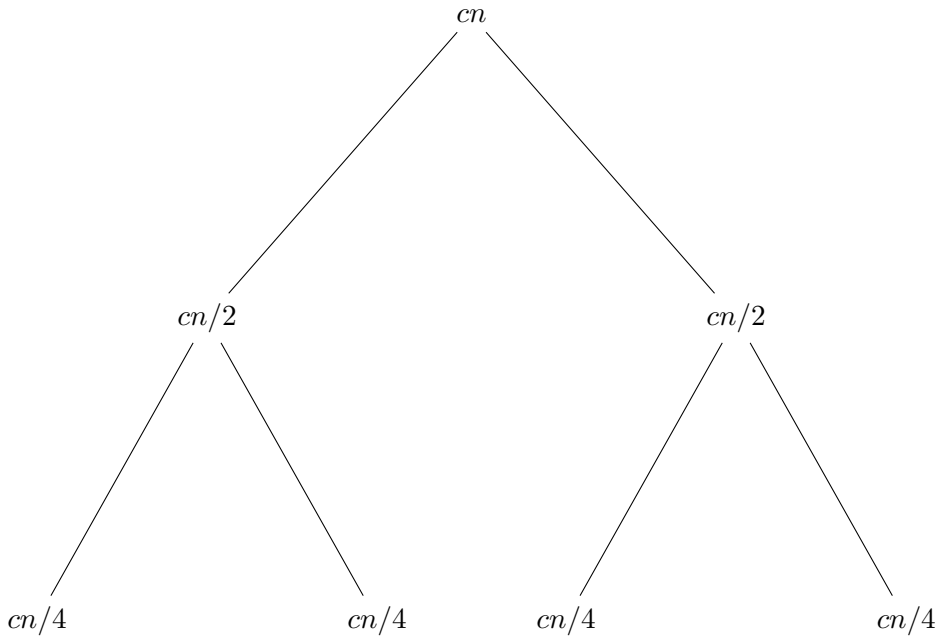
$$f_m^{\#} = \sum_{k=0}^{n-1} f_k\xi^{-mk} = \sum_{k=0}^{\frac{n}{2}-1} f_{2k}\xi^{-m(2k)} + \sum_{k=0}^{\frac{n}{2}-1} f_{2k+1}\xi^{-m(2k+1)}, \tag{1}$$

for $m = 0, 1 \ldots, n-1$, where we are essentially just breaking up the summation into its even and odd indexed summations. If we also notice that

$$\begin{aligned} f_m^{\#} &= \sum_{k=0}^{\frac{n}{2}-1} f_{2k}\xi^{-m(2k)} + \sum_{k=0}^{\frac{n}{2}-1} f_{2k+1}\xi^{-m(2k+1)} \\ &= \sum_{k=0}^{\frac{n}{2}-1} f_{2k}\xi^{-2mk} + \xi^{-m}\sum_{k=0}^{\frac{n}{2}-1} f_{2k+1}\xi^{-2mk} \end{aligned} \tag{2}$$

for $m = 0, 1, \ldots, n-1$, then we can use the idea from (1) on each of the two individual summations in (2). Applying this process recursively until each of the individual summations has just one term, reduces the computational complexity of the DFT from $\mathcal{O}(n^2)$

to $\mathcal{O}(n \log_2 n)$, giving us the FFT. This reduction in complexity can be visualized by the corresponding binary tree generated by the recursive process,



3 Problem 2