1 Introduction

The Fast Fourier Transform is easily one of the most significant algorithms (if not the most) of all time. It is an essential tool for almost all modern technology, including cell service, wifi, image compression, audio compression, radar, and numerous other technological applications. However, its relevance is not limited to technology. The FFT is an essential tool in parts of linear algebra, differential equations, as well as many parts of physics.

2 Historic Context

The first person to discover the FFT was Gauss in the early nineteenth century. He only briefly mentioned it in his notes since he decided to use a different numeric method for his research of celestial bodies, and it was only discovered far later that he had come up with the FFT.

Because Gauss only mentioned the method in passing in an obscure note, Fourier transforms as a whole were not discovered until Joseph Fourier came up with the Fourier transform while studying the heat equation. He invented the Fourier transform as a method to decompose a function into the summation of sines. Fourier additionally proved that any function can be represented in this way, making the method incredibly versatile.

The FFT proper was invented by J.W. Cooly and J.W. Tukey in order to be able to detect underground nuclear weapons tests during the Cold War. An underground nuclear test causes a seismograph to give out a sequence of readings which vary based on the strength and depth of the bomb being tested. However, disentangling the signal to determine the depth and power of a test would take an infeasible time with the DFT (Discrete Fourier Transform), so Cooly and Tukey came up with a way to dramatically reduce the calculation time by cleverly eliminating redundant calculations. It was *creatively* named the Fast Fourier Transform.

Now, whenever a Fourier Transformation is needed, the FFT is almost universally used for its accuracy and speed. [1]

3 Introducing our Example Problem

An interesting and simple example we can look at is polynomial multiplication. This approach should abstract the method from the complexity of the problems it is often used to solve.

The naive approach to solving polynomial multiplication would be to multiply each term in each polynomial by every other term in the other polynomial via the distributive rule. Suppose that we want to evaluate two polynomials of degree $\frac{n}{2}$, we will have $T = O(n^2)$ where T is the runtime complexity of our algorithm. Our goal is to find a clever way using a Fourier transform to accomplish $T = O(n \log_2 n)$. Nota bene: we shall name our nth degree polynomial P.

Recall from earlier in the semester how for any given set of $n+1$ points there is exactly one polynomial of degree n which passes through all of the points.

So we can choose $p_0, ..., p$ points to evaluate each of our functions at in order to arrive at their product, $\hat{c}$ (more on this in Section 6). However, that results in the following matrix vector multiplication

$$\hat{c} = \begin{bmatrix} x_0^0 & x_1^0 & \dots & x_n^0 \\ x_0^1 & x_1^1 & \dots & x_n^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_0^n & x_1^n & \dots & x_n^n \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix}$$

which will also take $O(n^2)$ time to solve.

4 Discrete Fourier Transform

We have, however, gotten very close to reconstructing the Discrete Fourier Transform, which can be defined as follows.

$$\hat{f}_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \hat{f}_j e^{-\frac{2\pi i}{n} jk} \text{ for } k = 0, 1, 2, ..., n-1.$$

Suppose that f is the period representation of our polynomial P over our sampled $n+1$ points. Our equation will convert the coefficients of f to values at our sampled points in $\hat{f}$. This results in a system of equations which can be represented as an $n \times n$ matrix

$$F_n = \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \dots & \omega^n \\ \omega^0 & \omega^2 & \omega^3 & \dots & \omega^{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^n & \omega^{2n} & \dots & \omega^{n^2} \end{bmatrix}$$

where $\omega = e^{-\frac{2\pi i}{n}}$. This matrix is called a Fourier matrix which can be used evaluate the polynomial we have set up with $\hat{f}$. Notice, how our intuition earlier is very similar (except the special ω, but more on those later). However, solving our linear system $F_n f = \hat{f}$ will still take $T = O(n^2)$ time. So, even though we have found a different solution to polynomial multiplication with the Discrete Fourier Transform and some linear algebra we haven't made progress on the problem itself.

5 Roots of Unity

Now many of you are probably wondering what the ω values in the Fourier transformation are. This is where I will tangent, briefly, into the roots of unity. The roots of unity are defined as $\omega, \omega^2, ..., \omega^{n-1}$. So, for some $n \geq 5$ polynomial p^* our $5th$ root of unity will be

$$\omega^5 = e^{-\frac{2\pi i}{5}}.$$

And the value $\omega = e^{-\frac{2\pi i}{n}}$ is called the principle root of unity. Notice that these Roots of Unity are in the complex plane, and for any given function of degree n it is to our advantage to choose the $n+1$ where $n+1$ is a power of 2. This is because our values $e^{-\frac{2\pi i}{n}} = \cos(\frac{2\pi}{n}) - i\sin(\frac{2\pi}{n})$ on the unit circle of the complex plain. (Read into this further [3, 4])

6 Fast Fourier Transform

Now let us return to our attempt at evaluating P. We will take $n+1$ to be some power of 2, since it simplifies things a little bit (the algorithm still works on other values of $n+1$, but this is the simplest family of cases).

So once P is evaluated it will look something like:

$$c_0 + c_1 x + ... + c_{n-1}x^{n-1}.$$

But let us instead represent P as a vector of values at points, like we had been doing before $\hat{P} = [p_0, p_1, ..., p_{n-1}]$. Now let us choose to evaluate our function at the roots of unity. By doing this notice that the odd values evaluate to the negative input of the even values s.t.

$$\hat{P}_e = [p_0, p_2, ...],$$
$$\hat{P}_o = [p_1, p_3, ...]$$

Which means instead of evaluating one degree n polynomial with our DFT, we can evaluate two degree $\frac{n}{2}$ polynomials. But we can only do this if we choose points to evaluate s.t.

$$P(x) = P(-x)$$

Because this allows us to use work solving $P(x)$ to have "already solved" $P(-x)$. And since we choose our roots of unity, they will lie upon the unit circle. Meaning that our prior calculation is really:

$$\hat{p}_e = [c_0, c_2, ...],$$
$$\hat{p}_o = [-c_0, -c_2, ...]$$

which ackomplishes our goal. We can then perform recursive calls on each of these functions until we reach a base case (thence the assuming of n being a power of two earlier).

This means that we now have a time complexity which can be solved using the Master Theorem for recursive functions.

$$T(n) = 2T(\frac{n}{2}) + O(n) = O(n\log_2 n)$$

where the $2T(\frac{n}{2})$ comes from splitting the problem and the $O(n)$ is the vector calculations that we will have to perform on each combining step. (See [6] if you want to learn more about the Master Theorem)

Graphically, it is helpful to think about how points will pair off on the unit circle based around the roots of unity. If we were working with frequencies more directly this would be represented by moments where you are calculating the same frequency in a signal repeatedly (this last intuition is key, I would watch the video by Reducible [3] to get a more clear understanding of what is happening). Which leads to the central theme of the FFT, don't repeat work you have already done.

7 References

This was a rather short introduction to a massive topic in mathematics. I would highly advise examining the following resources in order to get a more comprehensive view on the FFT and related problems.

1. The Story of the Fast Fourier Transform (Versci):
   https://www.versci.com/fft/index.html

2. The Discrete Fourier Transform (Reducible):
   https://www.youtube.com/watch?v=yYEMxqreA10

3. The Fast Fourier Transform (Reducible):
   https://www.youtube.com/watch?v=h7apO7q16V0&t=205s

4. Big Ideas Applied in Math: The Fast Fourier Transform (Ethen N. Epperly):
   https://www.ethanepperly.com/index.php/2021/05/10/big-ideas-in-applied-math-the-fast-fourier-transform/

5. An Intuitive Interpretation Of The Fourier Transform (or The Link Between Fourier Analysis And Linear Algebra):
   https://devincody.github.io/Blog/post/an_intuitive_interpretation_of_the_fourier_transform/

6. Advanced Master Theorem for Divide and Conquer Recurrences:
   https://www.geeksforgeeks.org/advanced-master-theorem-for-divide-and-conquer-recurrences/

7. But what is the Fourier Transform? A Visual Introduction:
   https://www.3blue1brown.com/lessons/fourier-transforms