Fast Fourier Transform

Jiankun Wei

November 27, 2022

Contents

1 Introduction

Fourier Transform is a widely used method in many different areas with countless applications. One obvious application is in Signal Processing where we want to decompose a wave into its elementary sine and cosine waves. [1] Fourier Transform has a lot to say on its own, but to quickly get to the topic of Fast Fourier Transform, I will only focus on the multiplication of polynomials. Normally multiplying polynomials will take $\Theta(n^2)$ as we need to multiply all $n+1$ terms in the first polynomial with all the $n+1$ terms in the second polynomial. [4] But FFT can do this better. In this paper, I will illustrate how the FFT algorithm is developed, implemented, and show that it has $\Theta(n \log(n))$ complexity.

Remark. From now on, $\mathbb{F}$ will denote either $\mathbb{C}$ or $\mathbb{R}$.

2 Interpolating Polynomial

In many areas of Data Science, Physics, Machine Learning, etc. we are often given a dataset $S = \{(x_1, p_1), \ldots, (x_n, p_n)\} \subseteq \mathbb{F}^2$, where x's are the time that measurements are taken and p's are the measurements. And we are asked to find an **Interpolating Polynomial** $\tilde{p} \in P_{n-1}(\mathbb{F})$ which interpolates S. That means for all $i \in \{1, \ldots, n\}, \tilde{p}(x_i) = p_i$.
We can write any polynomial $\tilde{p}$ as $\tilde{p}(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$, and the system of equations

$$\begin{cases} \tilde{p}(x_1) &= p_1 \\ & \vdots \\ \tilde{p}(x_n) &= p_n \end{cases}$$

as a matrix equation:

$$\begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

If such a system has a solution, then an interpolating polynomial exists. Otherwise, the interpolating polynomial does not exist.

2.1 Interlude on Determinant

To analyze this matrix equation, we will resort to the powerful tool of determinants. In any linear algebra book, determinants are defined as the unique multilinear, alternating, normalizing function on rows or on columns. Here is a version of definition of determinants where I modified slightly to account for column interpretation.

Definition 1 (Determinant)**.** [3]
Let $\{e_1, \ldots, e_n\}$ be the standard basis of $\mathbb{F}^n$, Let $M \in \mathrm{MAT}_{n,n}(\mathbb{F})$ and $\{v_1, \ldots, v_n\}$ be its columns. Then the determinant of M is the unique multilinear, alternating, normalizing function f: $(\mathbb{F}^n)^n \to \mathbb{F}$ where det(M) = $f(v_1, \ldots, v_n)$
Muitilinear means that f is linear with respect to every entry when all other entries are fixed.

Alternating means whenever two entries of f swaps, f changes sign.
Normalizing means $f(e_1, \ldots, e_n) = 1$

This definition immediately leads to the following theorem:

Theorem 2 (Column Operations does not Change Determinant). *[3]*
Let $a \in \mathbb{F}$, Let $M \in MAT_{n,n}(\mathbb{F})$ and $\{v_1, \ldots, v_n\}$ be its columns. Let $i, j \in \{1, \ldots, n\}$ where $i \neq j$. Then det(M) $= f(v_1, \ldots, v_n) = f(v_1, \ldots, v_i + a \cdot v_j, \ldots, v_n)$

Proof. Let $M \in \mathrm{MAT}_{n,n}(\mathbb{F})$ and $\{v_1, \ldots, v_n\}$ be its columns. Let $i, j \in \{1, \ldots, n\}$ where $i \neq j$. Then by multilinear,

$$\begin{aligned} f(v_1, \ldots, v_i + a \cdot v_j, \ldots, v_n) &= f(v_1, \ldots, v_i, \ldots, v_n) + f(v_1, \ldots, a \cdot v_j, \ldots, v_n) \\ &= det(M) + a \cdot f(v_1, \ldots, v_j, \ldots, v_n) \\ &= det(M) + a \cdot f(v_1, \ldots, v_j, \ldots, v_j, \ldots, v_n) \end{aligned}$$

Since f is alternating, by swapping the two v_j, we have $f(v_1, \ldots, v_j, \ldots, v_j, \ldots, v_n) =$ -$f(v_1, \ldots, v_j, \ldots, v_j, \ldots, v_n)$, so $f(v_1, \ldots, v_j, \ldots, v_j, \ldots, v_n) = 0$.
Thus, we have det(M) $= f(v_1, \ldots, v_i + a \cdot v_j, \ldots, v_n)$ □

Here is the method of computing determinant:

Definition 3 (ij-th Minor). [3]
Let $M \in \mathrm{MAT}_{n,n}(\mathbb{F})$, the ij-th minor where $i, j \in \{1, \ldots, n\}$ is the matrix $M_{ij} \in \mathrm{MAT}_{n-1,n-1}(\mathbb{F})$ obtained by removing the i-th row and the j-th column from M.

Theorem 4 (Cofactor Expansion). *[3]*
For all $M = (m)_{ij} \in MAT_{n,n}(\mathbb{F})$, for all $i \in \{1, \ldots, n\}$, det(M) $= \sum_{j=1}^{n} (-1)^{i+j} m_{ij} det(M_{ij})$

Finally, I will present the following well celebrated theorem:

Theorem 5 (Determinant and Invertibility). *[3]*
Let $M \in MAT_{n,n}(\mathbb{F})$, $det(M) \neq 0$ if and only if M is invertible.

2.2 Uniqueness of Interpolating Polynomial

Currently, we alreay have a way to uniquely determine polynomial through its coefficients, but this leads to the problem of $\Theta(n^2)$ in multiplication, is there another way that we can multiply polynomial fast? Yes there is. If polynomial p pass through the point (1, 2), and polynomial q pass through the point (1, 4), then we immediately know that their product pq must pass through the point (1, 8). Indeed, graph of polynomials can be viewed as collection of points. So why don't we choose some representing points to denote each polynomials? In this way, the multiplication will be very straightforward![4] But how many points are needed? It turns out, for an $n - 1$ degree polynomial with $n \geq 2$, n points will uniquely determine the polynomial, but of course, its proof relies on determinant.

Lemma 6. .
Let n be a natural number with $n \geq 2$, let $x_1, \dots, x_n \in \mathbb{F}$

$$det \begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{pmatrix} = \prod_{1 \leq i < j \leq n} (x_j - x_i)$$

Proof. I will prove by induction on n
let $x_1, \dots, x_n \in \mathbb{F}$

1. (Base Case) n = 2
   Then
   $$det \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} = 1 \cdot x_2 - 1 \cdot x_1 = x_2 - x_1 = \prod_{1 \leq i < j \leq 2} (x_j - x_i)$$
   So the base case holds.

2. (Inductive Case) Assume the inductive hypothesis that there exists some $k \in \mathbb{N}$ where
   $$det \begin{pmatrix} 1 & x_1 & \dots & x_1^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & \dots & x_k^{k-1} \end{pmatrix} = \prod_{1 \leq i < j \leq k} (x_j - x_i)$$
   I WTS the case $k+1$ holds.
   Let
   $$A = \begin{pmatrix} 1 & x_1 & \dots & x_1^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k+1} & \dots & x_{k+1}^k \end{pmatrix}$$
   Then according to Theorem 2, I want to add $-x_1$ multiplied the l-th column to the $(l+1)$-st column for every $l \in \{1, \dots, k\}$, and get:
   $$A' = (a'_{ij}) = \begin{pmatrix} 1 & x_1 - x_1 & \dots & x_1^k - x_1^{k-1} \cdot x_1 \\ 1 & x_2 - x_1 & \dots & x_2^k - x_2^{k-1} \cdot x_1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k+1} - x_1 & \dots & x_{k+1}^k - x_{k+1}^{k-1} \cdot x_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & (x_2 - x_1) & \dots & x_2^{k-1} \cdot (x_2 - x_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (x_{k+1} - x_1) & \dots & x_{k+1}^{k-1} \cdot (x_{k+1} - x_1) \end{pmatrix}$$
   where we still have det(A') = det(A) by Theorem 4 and since $x_1 \in \mathbb{F}$, so $-x_1 \in \mathbb{F}$
   Now I compute the determinant of A' after row 1
   det(A) = det(A') = $\sum_{j=1}^{k+1} (-1)^{1+j} a'_{1j} det(A'_{1j}) = 1 \cdot det(A'_{11})$, where
   $$A'_{11} = \begin{pmatrix} (x_2 - x_1) & \dots & x_2^{k-1} \cdot (x_2 - x_1) \\ \vdots & \ddots & \vdots \\ (x_{k+1} - x_1) & \dots & x_{k+1}^{k-1} \cdot (x_{k+1} - x_1) \end{pmatrix}$$

since by definition 1, the determinant is multilinear, then

$$\begin{aligned}
det(A'_{11}) &= det\begin{pmatrix} (x_2-x_1) & \dots & x_2^{k-1}\cdot(x_2-x_1) \\ \vdots & \ddots & \vdots \\ (x_{k+1}-x_1) & \dots & x_{k+1}^{k-1}\cdot(x_{k+1}-x_1) \end{pmatrix} \\
&= (x_2-x_1)\dots(x_{k+1}-x_1)\cdot det\begin{pmatrix} 1 & x_2 & \dots & x_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k+1} & \dots & x_{k+1}^{k-1} \end{pmatrix} \\
&= (x_2-x_1)\dots(x_{k+1}-x_1)\cdot det\begin{pmatrix} 1 & x'_1 & \dots & (x'_1)^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x'_k & \dots & (x'_k)^{k-1} \end{pmatrix}
\end{aligned}$$

where the last step is by changing the reference of x_2 to x'_1, …, x_{k+1} to x'_k Now, by inductive hypothesis, we have

$$det\begin{pmatrix} 1 & x'_1 & \dots & (x'_1)^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x'_k & \dots & (x'_k)^{k-1} \end{pmatrix} = \prod_{1\le i<j\le k}(x'_j-x'_i) = \prod_{2\le i<j\le k+1}(x_j-x_i)$$

where the last step is by changing back the reference of x'_1 to x_2, …, x'_k to x_{k+1} plug this fact back into the original equation, we have:

$$\begin{aligned}
det(A'_{11}) &= (x_2-x_1)\dots(x_{k+1}-x_1)\cdot det\begin{pmatrix} 1 & x'_1 & \dots & (x'_1)^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x'_k & \dots & (x'_k)^{k-1} \end{pmatrix} \\
&= (x_2-x_1)\dots(x_{k+1}-x_1)\cdot \prod_{2\le i<j\le k+1}(x_j-x_i) \\
&= \prod_{1\le i<j\le k+1}(x_j-x_i)
\end{aligned}$$

Thus,

$$\begin{aligned}
det(A) &= 1\cdot det(A'_{11}) \\
&= 1\cdot \prod_{1\le i<j\le k+1}(x_j-x_i) \\
&= \prod_{1\le i<j\le k+1}(x_j-x_i)
\end{aligned}$$

So the inductive case holds.

Hence, by principle of induction, I have shown that the formula holds for $n \geq 2$ as needed. □

Now with this lemma, we are finally ready to prove the **main theorem** of this section, for an $n-1$ degree polynomial with $n \geq 2$, n points will uniquely determine the polynomial.

Theorem 7 (Uniqueness of Interpolating Polynomial)**.** .
Let n be a natural number with $n \geq 2$, let $S = \{(x_1, p_1), \ldots, (x_n, p_n)\} \subseteq \mathbb{F}^2$ be n points satisfying for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$, we must have $x_i \neq x_j$, then there is one unique polynomial $\tilde{p} \in P_{n-1}(\mathbb{F})$ which interpolate the dataset S.

Proof. Let n be a natural number with $n \geq 2$, let $S = \{(x_1, p_1), \ldots, (x_n, p_n)\} \subseteq \mathbb{F}^2$. Let $\tilde{p} \in P_{n-1}(\mathbb{F})$ where $\tilde{p}(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. Let's look at the matrix equation at the begining of this section, it says that we can write the system of equations

$$\begin{cases} \tilde{p}(x_1) & = p_1 \\ & \vdots \\ \tilde{p}(x_n) & = p_n \end{cases}$$

as a matrix equation:

$$\begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

where by Lemma6, we have

$$det \begin{pmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{pmatrix} = \prod_{1 \leq i < j \leq n} (x_j - x_i)$$

Since for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$, we must have $x_i \neq x_j$, so for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$, we must have $x_j - x_i \neq 0$, so $\prod_{1 \leq i < j \leq n}(x_j - x_i) \neq 0$. Thus this tells us:

$$det \begin{pmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{pmatrix} \neq 0$$

Now by Theorem 5, we know the matrix $\begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix}$ is invertible, and so it has an inverse.

In particular, the inverse of $\begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix}$ is $\begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix}^{-1}$

Since this matrix is invertible, we know that our matrix equation admits a unique solution and so the vector corresponds to the coefficient of the interpolating polynomial can be unique solved.

In particular,

$$\begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

Thus, since the solution is unique, and coefficients uniquely determines the polynomial, this means that the interpolating polynomial exists and is unique. □

Remark. Notice the requirement of $x_i \neq x_j$. Since polynomials are functions, they must be univalent. So it is impossible for a polynomial to map the same element in domain to different elements in the codomain.

2.3 Overfitting Problem

Now let's consider what will happen if we ask for an interpolating polynomial $\tilde{p} \in P_m(\mathbb{F})$ where $m > n - 1$?

Theorem 8 (Overfitting Problem)**.** .
Let n be a natural number with $n \geq 2$, let $S = \{(x_1, p_1), \dots, (x_n, p_n)\} \subseteq \mathbb{F}^2$ be n points satisfying for all $i, j \in \{1, \dots, n\}$ where $i \neq j$, we must have $x_i \neq x_j$, Let $m > n - 1$, and assume there exists a $\tilde{p} \in P_m(\mathbb{F})$ that interpolates the dataset S. Then there are infinitely many polynomials in $P_m(\mathbb{F})$ that interpolates the dataset S.

Proof. Let $p \in P_m(\mathbb{F})$ be a polynomial, assume $p(x) = a_0 + a_1 x + \cdots + a_m x^m$
then we can easily write the system of equations

$$\begin{cases} p(x_1) & = p_1 \\ & \vdots \\ p(x_n) & = p_n \end{cases}$$

as a matrix equation:

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

Let A $= \begin{bmatrix} 1 & x_1 & \dots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^m \end{bmatrix}$, let x $= \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix}$, let b $= \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$
Thus we have $Ax = b$
Let the polynomial that interpolates the data be $\tilde{p}$, Since A has n rows and $m + 1$ colunms, where $m > n - 1$, so $m + 1 > n$, then A has nontrivial kernel. In particular, there are infinitely many elements in ker(A).
Let $k_0 = \begin{bmatrix} a'_0 \\ \vdots \\ a'_m \end{bmatrix}$ be in ker(A), then $Ak_0 = 0$ where $k_0 \neq 0$.
So let $k_1 = k_0 + x$, let $p_1 \in P_m(\mathbb{F})$ be the polynomial with coefficient given by k_1, then $Ak_1 = A(k_0 + x) = Ak_0 + Ax = 0 + b = b$, so p_1 also interpolates the data. Since $k_0 \neq 0$, then $p_1 \neq \tilde{p}$. Also for each different nonzero element in ker(A), I have a unique p_1.
Thus, we have infinitely many polynomials that interpolate the data. □

Remark. This is particularly problematic because when we want to deduce the value at an unknown position, by Theorem 7 and $m \geq n$, for any value in $\mathbb{F}$ we can find a polynomial to predict it, thus our model is useless.

3 Evaluation

Now Let's consider our original question, we want to multiply two polynomials p and q each of degree n to get a polynomial pq where pq is of degree 2n. By Theorem 7, we know that $2n+1$ points will be sufficient to uniquely determine the polynomial pq. So our plan is to choose $2n+1$ different x values in the domain, evaluate p and q on these $2n+1$ places to get 2 sets of $2n+1$ points. multiply their corresponding y values, and we will get $2n+1$ distinct points (at least the x value are different) for pq, which will determine pq uniquely. This is a good plan as we successfully reduced the time requirement of multiplying polynomials to $\Theta(n)$ as multiplying $2n+1$ points takes $2n+1$ procedures. However, what about evaluating these points? We have $2n+1$ position to evaluate, and for each position, we need to plug it into polynomials with $n+1$ terms in the worst case, so the total steps is $(2n+1)\cdot(n+1) = 2n^2+3n+2$ for each polynomial in the worst case, which is still $\Theta(n^2)$, so we actually did not improve anything. How can we reduce the number of steps needed?

3.1 Odd and Even Functions

The solution, it turns out, is through the properties of Odd and Even Functions.[4] Let's first see what does odd and even functions mean.

Definition 9 (Odd Function)**.** A function f: $\mathbb{F} \to \mathbb{F}$ is odd if for all $x \in \mathbb{F}$, $f(-x) = -f(x)$

Definition 10 (Even Function)**.** A function f: $\mathbb{F} \to \mathbb{F}$ is even if for all $x \in \mathbb{F}$, $f(-x) = f(x)$

Now, by using odd and even function, we now only need half of the points, because if we take a positive position x, we immediately know what the value at position $-x$ is. In particular, we have the follow theorem:

Theorem 11. *[4]*
For all polynomial $p \in P_n(\mathbb{F})$, then there exists even polynomial $q, r \in P_n(\mathbb{F})$ such that $p = q + x \cdot r$.

Proof. Let $p(x) = a_0 + a_1x + \cdots + a_nx^n$

1. If n is even, then

$$\begin{aligned}
p(x) &= a_0 + a_1x + \cdots + a_nx^n \\
&= (a_0 + a_2x^2 + \cdots + a_nx^n) + (a_1x + a_3x^3 + \cdots + a_{n-1}x^{n-1}) \\
&= (a_0 + a_2x^2 + \cdots + a_nx^n) + x \cdot (a_1 + a_3x^2 + \cdots + a_{n-1}x^{n-2})
\end{aligned}$$

Now I take $q(x) = a_0 + a_2x^2 + \cdots + a_nx^n$ and $r(x) = a_1 + a_3x^2 + \cdots + a_{n-1}x^{n-2}$, then $q, r \in P_n(\mathbb{F})$
First, I will show q is even. Let $x \in \mathbb{F}$,

$$\begin{aligned} q(-x) &= a_0 + a_2(-x)^2 + \cdots + a_n(-x)^n \\ &= a_0 + a_2x^2 + \cdots + a_nx^n \\ &= q(x) \end{aligned}$$

since each term is of even power and (-x) to an even power is the same as x to the same power, thus q is even.
Now I will show r is even. Let $x \in \mathbb{F}$,

$$\begin{aligned} r(-x) &= a_1 + a_3(-x)^2 + \cdots + a_{n-1}(-x)^{n-2} \\ &= a_1 + a_3x^2 + \cdots + a_{n-1}x^{n-2} \\ &= r(x) \end{aligned}$$

since each term is of even power and (-x) to an even power is the same as x to the same power, thus r is even.
Therefore, I successfully write $p = q + x \cdot r$ where both q and r are even function.

2. If n is odd, then

$$\begin{aligned} p(x) &= a_0 + a_1x + \cdots + a_nx^n \\ &= (a_0 + a_2x^2 + \cdots + a_{n-1}x^{n-1}) + (a_1x + a_3x^3 + \cdots + a_nx^n) \\ &= (a_0 + a_2x^2 + \cdots + a_{n-1}x^{n-1}) + x \cdot (a_1 + a_3x^2 + \cdots + a_nx^{n-1}) \end{aligned}$$

Now I take $q(x) = a_0 + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ and $r(x) = a_1 + a_3x^2 + \cdots + a_nx^{n-1}$, then $q, r \in P_n(\mathbb{F})$
First, I will show q is even. Let $x \in \mathbb{F}$,

$$\begin{aligned} q(-x) &= a_0 + a_2(-x)^2 + \cdots + a_{n-1}(-x)^{n-1} \\ &= a_0 + a_2x^2 + \cdots + a_{n-1}x^{n-1} \\ &= q(x) \end{aligned}$$

since each term is of even power and (-x) to an even power is the same as x to the same power, thus q is even.
Now I will show r is even. Let $x \in \mathbb{F}$,

$$\begin{aligned} r(-x) &= a_1 + a_3(-x)^2 + \cdots + a_n(-x)^{n-1} \\ &= a_1 + a_3x^2 + \cdots + a_nx^{n-1} \\ &= r(x) \end{aligned}$$

since each term is of even power and (-x) to an even power is the same as x to the same power, thus r is even.
Therefore, I successfully write $p = q + x \cdot r$ where both q and r are even function.

In conclusion, I successfully write $p = q + x \cdot r$ where both q and r are even function. □

Remark. According to the proof, if $p(x) = a_0 + a_1x + \cdots + a_nx^n$ where n is even, then $q(x) = a_0 + a_2x^2 + \cdots + a_nx^n$ and $r(x) = a_1 + a_3x^2 + \cdots + a_{n-1}x^{n-2}$. if I take $y = x^2$, then $q(y) = a_0 + a_2y + \cdots + a_ny^{\frac{n}{2}}$ and $r(y) = a_1 + a_3y + \cdots + a_{n-1}y^{\frac{n-2}{2}}$ which all make sense as n is even.
If n is odd, then $q(x) = a_0 + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ and $r(x) = a_1 + a_3x^2 + \cdots + a_nx^{n-1}$. if I take $y = x^2$, then $q(y) = a_0 + a_2y + \cdots + a_ny^{\frac{n-1}{2}}$ and $r(y) = a_1 + a_3y + \cdots + a_ny^{\frac{n-1}{2}}$ which all make sense as n is odd.
Thus, we can rewrite p such that $p(x) = q(y) + x \cdot r(y)$ where $y = x^2$, so for any chosen x, we have $p(x) = q(x^2) + x \cdot r(x^2)$, and $p(-x) = q(x^2) - x \cdot r(x^2)$

Now with this remark, given any polynomial p, I only need to evaluate half of the points that is positive, and the negative points are immediately known. This leads us to think about using recursion to solve this problem. However, there is one last problem. In the recursive step, I want to appeal to Theorem 11 again to evaluate only half of the points, but now all points are positive as x^2 is guaranteed to be positive under real numbers, so our chain breaks. Is there a set of points that can be taking powers yet still producing positive and negative pairs?

3.2 Complex Numbers and Roots of Unity

The solution turned out to be in the Complex Numbers. And the n-th root of 1 is just what we need.[4] We know that any complex number can be written as $a + bi$ where a is the real part and b is the imaginary part, but, there is a further polar interpretation of complex numbers.

Definition 12. Complex Conjugate[3]
Let $z = a+bi$ be a complex number, then the **Complex Conjugate** of z denote $\overline{z}$ is $\overline{z} = a-bi$

Definition 13. Polar Representation[3]
Let $z = a + bi$ be a complex number , let $r = \sqrt{a^2 + b^2}$, let $\theta \in [0, 2\pi)$ be an angle such that $tan(\theta) = \frac{b}{a}$, then $z = r(cos(\theta) + i \cdot sin(\theta))$ and we call $r(cos(\theta) + i \cdot sin(\theta))$ the **polar representation** of z.
Further, z is also equal to

$$r(cos(\theta + 2k\pi) + i \cdot sin(\theta + 2k\pi)) \text{ for all } k \in \mathbb{Z}$$

and

$$-r(cos(\theta + (2k + 1)\pi) + i \cdot sin(\theta + (2k + 1)\pi)) \text{ for all } k \in \mathbb{Z}$$

With polar representation, we can talk about the roots of complex numbers:

Definition 14 (N-th root of Complex numbers)**.** [3]
for all $n \in \mathbb{N}$, for all $z_0 = r_0(cos(\theta_0) + i \cdot sin(\theta_0)) \in \mathbb{C}$, let $z = r(cos(\theta) + i \cdot sin(\theta)) \in \mathbb{C}$ be an n-th root of z_0, then

$$r = r_0^{\frac{1}{n}}$$

and

$$\theta = \frac{1}{n}\theta_0 + \frac{2k\pi}{n} \text{ for some } k \in \mathbb{Z}, k \in \{0, \ldots, n-1\}$$

Remark. In particular, there are n distinct values for θ, so there are n distinct n-th roots of every complex number.
In our recursive algorithm, since we are keep dividing the set of points into 2 sets. we need the number of points to be some power of 2. So from now on, we will let n be some power of 2.

Now let's look at 1's roots with the help of Euler's formula.

Definition 15 (Euler's Formula)**.** [4]
Let $\theta \in \mathbb{R}$, then

$$e^{i\theta} = cos(\theta) + i \cdot sin(\theta)$$

Theorem 16 (N-th Root of Unity)**.** *[4]*
The n-th root of 1 has the form $e^{\frac{2ki\pi}{n}}$

Proof. Since $1 = 1 + 0i = 1(cos(0) + i \cdot sin(0))$, by Definition 14, the n-th root of 1 are $z = r(cos(\theta) + i \cdot sin(\theta)) \in \mathbb{C}$ where $r = 1^{\frac{1}{n}} = 1$, and $\theta = \frac{1}{n}\theta_0 + \frac{2k\pi}{n}$ for some $k \in \mathbb{Z}, k \in \{0, \ldots, n-1\}$
Then that means $\theta = \frac{1}{n} \cdot 0 + \frac{2k\pi}{n} = \frac{2k\pi}{n}$
Thus, $z = 1 \cdot (cos(\frac{2k\pi}{n}) + i \cdot sin(\frac{2k\pi}{n})) = e^{\frac{2ki\pi}{n}}$ by Euler's Formula. Thus, then n-th root of 1 has the form $e^{\frac{2ki\pi}{n}}$ □

Remark. Now, taking k = 1, we define $w = e^{\frac{2i\pi}{n}}$
Let n be an even integer, if we raise w to some integer power m, what will happen?
$(w)^m = (e^{\frac{2i\pi}{n}})^m = e^{m \cdot \frac{2i\pi}{n}}$
Now consider the n-th power of $(w)^m$, we have $((w)^m)^n = (e^{m \cdot \frac{2i\pi}{n}})^n = e^{n \cdot m \cdot \frac{2i\pi}{n}} = e^{2mi\pi} = cos(2m\pi) + i \cdot sin(2m\pi) = 1$ as m is an integer. So integer powers of w is still n-th root of unity!
In addition, given an n-th root of unity r, -r is also an n-th root of unity as $(-r)^n = (-1)^n r^n = 1$ because n is a power of 2 according to previous remark.

Finally, I am ready to proof the theorem that we have long anticipated for, that the square of n-th root of 1 come in positive and negative pairs.

Theorem 17 (Square of N-th Root of Unity Comes with Positive and Negative Pairs)**.** *Let* $n \geq 4$, *let* $e^{\frac{2ki\pi}{n}}$ *where* $k \in \{0, \ldots, n-1\}$ *be an n-th root of 1, then there exists an* $m \in \{0, \ldots, n-1\}$ *such that* $(e^{\frac{2mi\pi}{n}})^2 = -(e^{\frac{2ki\pi}{n}})^2$

Proof. since $e^{\frac{2ki\pi}{n}}$ is an n-th root of 1, so $(e^{\frac{2ki\pi}{n}})^2 = e^{2 \cdot \frac{2ki\pi}{n}} = e^{\frac{4ki\pi}{n}}$

1. If $k < \frac{3n}{4}$, then I take $m = k + \frac{n}{4}$, then

$$\begin{aligned}
(e^{\frac{2mi\pi}{n}})^2 &= (e^{\frac{2(k+\frac{n}{4})i\pi}{n}})^2 \\
&= (e^{\frac{(2k+\frac{n}{2})i\pi}{n}})^2 \\
&= e^{2\cdot\frac{(2k+\frac{n}{2})i\pi}{n}} \\
&= e^{\frac{(4k+n)i\pi}{n}} \\
&= e^{\frac{4ki\pi}{n}} e^{i\pi} \\
&= e^{\frac{4ki\pi}{n}} (cos(\pi) + i \cdot sin(\pi)) \\
&= -e^{\frac{4ki\pi}{n}}
\end{aligned}$$

2. If $k \geq \frac{3n}{4}$, then I take $m = k - \frac{n}{4}$, then

$$\begin{aligned}
(e^{\frac{2mi\pi}{n}})^2 &= (e^{\frac{2(k-\frac{n}{4})i\pi}{n}})^2 \\
&= (e^{\frac{(2k-\frac{n}{2})i\pi}{n}})^2 \\
&= e^{2\cdot\frac{(2k-\frac{n}{2})i\pi}{n}} \\
&= e^{\frac{(4k-n)i\pi}{n}} \\
&= \frac{e^{\frac{4ki\pi}{n}}}{e^{i\pi}} \\
&= \frac{e^{\frac{4ki\pi}{n}}}{(cos(\pi) + i \cdot sin(\pi))} \\
&= \frac{e^{\frac{4ki\pi}{n}}}{-1} \\
&= -e^{\frac{4ki\pi}{n}}
\end{aligned}$$

In either case, the squares of the n-th root of unity still comes in positive and negative pairs. □

Remark. In addition, as $n \geq 4$ is a power of 2, by redefining $n = \frac{n}{2}$, we still have n-th root of unity.

3.3 DFT Matrix

Now we have already shown that by taking $n \geq 4$ to be power of 2, the n-th root of unity comes in positive and negative pairs, their integer powers are also n-th root of unity, and their squares also comes in positive and negative pairs. Now consider our matrix equation back in section 1. We have:

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

Since we require $n \geq 4$ to be power of 2, and now we need our polynomial to have the same degree as n. This is easy as we can simply take any valid n that is greater than the degree of the polynomial and view the additional term of the polynomial as having coefficient 0. Thus, we have the coefficient vectors: $\begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix}$. We need to fill in the matrix $\begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix}$ by roots of unity so we can compute the values on the right hand side.
we first choose an $n \geq 4$ that is a power of 2, and greater than the degree of the polynomial. Then by Theorem 16, we have $S = \{e^{\frac{2ki\pi}{n}} \mid k \in \{0, \dots, n-1\}\}$ being the n-th root of unity where S contain n points and we need to fill in x_1 to x_n. So we can simply take for all $k \in \{0, \dots, n-1\}$, Let $x_{k+1} = e^{\frac{2ki\pi}{n}}$, and this will satisfy all our requirements.
notice, as we define $w = e^{\frac{2i\pi}{n}}$, then $w^k = e^{\frac{2ki\pi}{n}}$, thus this means we take $x_{k+1} = w^k$.
Therefore, our matrix becomes:

$$\begin{bmatrix} 1 & w^0 & \dots & w^{0\cdot(n-1)} \\ 1 & w^1 & \dots & w^{1\cdot(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & \dots & w^{(n-1)\cdot(n-1)} \end{bmatrix}$$

and we define this matrix to be the famous DFT Matrix.

Definition 18 (DFT Matrix)**.** [5]
The DFT (Discrete Fourier Transform) matrix of size $n \times n$ is the matrix

$$M = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^1 & w^2 & \dots & w^{(n-1)} \\ 1 & w^2 & w^4 & \dots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{(n-1)} & w^{2(n-1)} & \dots & w^{(n-1)^2} \end{bmatrix}$$

Where $w = e^{\frac{2i\pi}{n}}$

4 Interpolation

Now let's look at what we have achieved. So far we are able to take the coefficient representation of a polynomial, and use the DFT Matrix to find several value points that represent it by simply performing the matrix multiplication. Using these value points, we can do the multiplication in $\Theta(n)$. However, we need a way to change the resulting value points back to coefficients. Fortunately, in the proof of Theorem 7, we already have the way to do this, and the new coefficients is given by:

$$\begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

We have the values, and we only need to find

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix}^{-1}$$

That is, we need to find the inverse of the DFT Matrix.

Theorem 19 (Inverse DFT Matrix)**.** *[5]*
The Inverse of DFT matrix of size $n \times n$ is the matrix

$$M^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^{-1} & w^{-2} & \dots & w^{-(n-1)} \\ 1 & w^{-2} & w^{-4} & \dots & w^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{-(n-1)} & w^{-2(n-1)} & \dots & w^{-(n-1)^2} \end{bmatrix}$$

Where $w = e^{\frac{2i\pi}{n}}$

Proof. notice that both the DFT matrix and the Inverse DFT Matrix is symmetric as it is equal to its transpose, so the entries in the j-th column of the matrix is the same as the j-th row of the matrix, for both of the matrices.

1. Now the jk-th entry of the product MM^{-1} is

$$\begin{aligned} (MM^{-1})_{jk} &= \frac{1}{n} \cdot \sum_{l=1}^{n} w^{(j-1)\cdot(l-1)} \cdot w^{-(l-1)\cdot(k-1)} \\ &= \frac{1}{n} \cdot \sum_{l=1}^{n} w^{(l-1)((j-1)-(k-1))} \\ &= \frac{1}{n} \cdot \sum_{l=1}^{n} w^{(l-1)(j-k)} \end{aligned}$$

   (a) If j = k, then $j-k = 0$ so every term in the summation becomes 1, so $\sum_{l=1}^{n} w^{(l-1)(j-k)} = n$, so $(MM^{-1})_{jk} = \frac{1}{n} \cdot \sum_{l=1}^{n} w^{(l-1)(j-k)} = \frac{1}{n} \cdot n = 1$.

   (b) if $j \neq k$, then $\sum_{l=1}^{n} w^{(l-1)(j-k)} = \sum_{l=1}^{n} e^{\frac{2(l-1)(j-k)i\pi}{n}} = \sum_{l=1}^{n} e^{\frac{2l(j-k)i\pi}{n}} e^{\frac{-2(j-k)i\pi}{n}} = e^{\frac{-2(j-k)i\pi}{n}} \cdot \sum_{l=1}^{n} e^{\frac{2l(j-k)i\pi}{n}} = 0$ because we already showed that the n-th root of unity comes in positive and negative pairs and we already assumes n is a power of 2. Thus $(MM^{-1})_{jk} = \frac{1}{n} \cdot \sum_{l=1}^{n} w^{(l-1)(j-k)} = \frac{1}{n} \cdot 0 = 0$.

Thus, the diagonal entries of the product is 1 and off diagonal entries are 0, so $MM^{-1} = I$

2. Now the jk-th entry of the product $M^{-1}M$ is

$$\begin{aligned}(M^{-1}M)_{jk} &= \frac{1}{n}\cdot\sum_{l=1}^{n} w^{-(j-1)\cdot(l-1)}\cdot w^{(l-1)\cdot(k-1)}\\ &= \frac{1}{n}\cdot\sum_{l=1}^{n} w^{(l-1)(-(j-1)+(k-1))}\\ &= \frac{1}{n}\cdot\sum_{l=1}^{n} w^{(l-1)(k-j)}\end{aligned}$$

   (a) If j = k, then $k-j=0$ so every term in the summation becomes 1, so $\sum_{l=1}^{n} w^{(l-1)(k-j)} = n$, so $(M^{-1}M)_{jk} = \frac{1}{n}\cdot\sum_{l=1}^{n} w^{(l-1)(k-j)} = \frac{1}{n}\cdot n = 1$.

   (b) if $j \neq k$, then $\sum_{l=1}^{n} w^{(l-1)(k-j)} = \sum_{l=1}^{n} e^{\frac{2(l-1)(k-j)i\pi}{n}} = \sum_{l=1}^{n} e^{\frac{2l(k-j)i\pi}{n}} e^{\frac{-2(k-j)i\pi}{n}} = e^{\frac{-2(k-j)i\pi}{n}}\cdot\sum_{l=1}^{n} e^{\frac{2l(k-j)i\pi}{n}} = 0$ because we already showed that the n-th root of unity comes in positive and negative pairs and we already assumes n is a power of 2. Thus $(M^{-1}M)_{jk} = \frac{1}{n}\cdot\sum_{l=1}^{n} w^{(l-1)(k-j)} = \frac{1}{n}\cdot 0 = 0$.

Thus, the diagonal entries of the product is 1 and off diagonal entries are 0, so $M^{-1}M = I$

Therefore, I have shown that the inverse DFT Matrix is indeed the inverse of the DFT matrix. □

5 Pseudocode and Complexity

Now we have all the pieces solved to assemble the program. This code based mainly on code provided in the video.[4]

Code 20. [4]

```
import math
 def FFT(list_in, direction) -> list_out:
    """This FFT algorithm will perform the FFT or Inverse FFT
       according to the direction specified.

       direction is either 1 or -1.
       When it is 1, then the algorithm perform FFT
       When it is -1, then the algorithm perform Inverse FFT

       When performing FFT, list_in will be the list of coefficients
       of a polynomial starting from constant term and end with the
       term of highest degree, and list_out will produce a list of y-values
       corresponding to the value points of the polynomial.

       When performing Inverse FFT, list_in will be the list of y-values
       corresponding to the value points of the polynomial, and
       list_out will produce the list of coefficients of the polynomial

       Precondition: the length of list-in must be a power of 2
       """
       n = len(list_in)
       if direction = 1:
           w = math.exp(2 * math.pi * math.complex(0, 1)/n)
       else:
           w = (math.exp(-2 * math.pi * math.complex(0, 1)/n))/n
       even = []
       odd = []
       if n == 2:
           return [list_in[0] + list_in[1], list_in[0] - list_in[1]]
       else:
           for i in range(n):
               if i % 2 == 1:
                   odd.append(i)
               else:
                   even.append(i)
           even_out = FFT(even, direction)
           odd_out = FFT(odd, direction)
           result = []
```

```
for j in range(n/2):
    result[j] = even_out[j] + (w ** j) * odd_out[j]
    result[j + n/2] = even_out[j] - (w ** j) * odd_out[j]
return result
```

[1] Now, Let's analysis the complexity through the Master Theorem.

Theorem 21 (Master Theorem)**.** *[2]*
Given a Divide-And-Conquer Algorithm, let a be the number of recursive calls, let b be the number of pieces we divide into, let f be the cost of splitting and recombining,
If $f \in \Theta(n^d)$, then

$$T(n) = \begin{cases} \Theta(n^d) & \textit{if } a < b^d \\ \Theta(n^d \log_b(n) & \textit{if } a = b^d \\ \Theta(n^{\log_b(a)}) & \textit{if } a > b^d \end{cases}$$

Now, from the code, it is clear that $a = 2$, $b = 2$ and $d = 1$, so $a = b^d$
Thus the running time is $T(n) = \Theta(n^1 \log_2(n))$ which is $\Theta(n \log(n))$[2]

6 Conclusion

In this paper, we have started with the uniqueness of interpolating polynomial, which inspires us to use values instead of coefficients to represent polynomials leading to a very fast solution for multiplication. Then we resort to complex numbers to be able to keep squaring a set of numbers while still having positive and negative pairs which allows us to keep profiting from properties of odd and even functions in recursion. Finally we achieved an algorithm that is $\Theta(n \log(n))$ which is significantly faster than the original $\Theta(n^2)$ algorithm. And the resulting algorithm is the Fast Fourier Transform.

[1] This pseudocode was mainly based on the syntax of Python. I changed the code given in the video to merge the 2 functions into one.

[2] As a convention in CS, log denote log base 2.

References

[1] 3Blue1Brown. *But what is the Fourier Transform? A visual introduction.* URL: https://www.youtube.com/watch?v=spUNpyF58BY. (accessed: 06.25.2022).

[2] CLRS. *Introduction to Algorithms.* Cambridge, Massachusetts: MIT Press, 2009.

[3] John Little David Damiano. *A Course in Linear Algebra.* Orlando, Florida 32887: Harcourt Brace Jovanovich, Inc., 1998.

[4] Reducible. *The Fast Fourier Transform (FFT): Most Ingenious Algorithm Ever?* URL: https://www.youtube.com/watch?v=h7apO7q16V0&t=362s. (accessed: 06.25.2022).

[5] Gilbert Strang. *LECTURE 26: COMPLEX MATRICES; FAST FOURIER TRANSFORM (FFT).* URL: https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/resources/lecture-26-complex-matrices-fast-fourier-transform-fft/. (accessed: 06.25.2022).