Lab 3: The Fast Fourier Transform

Assume that we have a finite signal $x[0], \ldots, x[N-1]$ with n terms that we pad with infinite zeros at the beginning and at the end, and we want to compute the Fourier transform that we have defined as

$$X(\omega) = \sum_{n \in \mathbb{Z}} x[n] e^{-in\omega}.$$

where $\omega \in [-\pi, \pi)$ or $[0, 2\pi)$ for convenience. If we evaluate at the N frequencies $\frac{2\pi j}{N}$ for $j = 0, \ldots, N-1$, we get the values of the Fourier transform in some equispaced points in the frequency interval. The transformation becomes

$$y[j] = X[2\pi j/N] = \sum_{n=0}^{N-1} x[n] e^{-inj2\pi/N}, \qquad 0 \leq j < N. \tag{1}$$

This is called the discrete Fourier transform, that maps values from $\mathbb{C}^N$ $\{x[i]\}_{i=0}^{N-1}$ to $\mathbb{C}^N$ $\{y[j]\}_{j=0}^{N-1}$. We have interpreted it as way of evaluating the Fourier transform of an infinite sequence x, which is a priori defined on the whole interval $[0, 2\pi]$ only at the frequencies corresponding to the N-roots of unity. There is another way of interpreting this discrete Fourier transform as the Fourier transform in the finite group $G = \mathbb{Z}/N\mathbb{Z}$ but we won't take this approach.

THEOREM 1. *The discrete Fourier transform (DFT) has an inverse:*

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} y[k] e^{ikn2\pi/N}, \qquad 0 \leq n < N. \tag{2}$$

Observe that the sequence $x[n]$ that we obtain in the inverse formula is N-periodic (and also for the direct formula), and the sequence we started from was originally compactly supported. For this reason one may interpret the DFT and its inverse as mappings from N-periodic signals to N-periodic signals. There are several ways to interpret the DFT. We outline three of them. The first one is, as we introduced it, a discretization of the usual Fourier transform of sequences. The second one is as a linear mapping from $\mathbb{C}^N$ to $\mathbb{C}^N$. If we denote by $W_N = e^{-2\pi i/N}$ a primitive root of unity, then the DFT is given by

$$y[k] = \sum_{n=0}^{N-1} W_N^{kn} x[n],$$

and the IDFT is given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{-kn} y[k]. \tag{3}$$

Thus, what we need to check to prove Theorem 1 is that $(W_N^{kn})_{k,n}^{-1} = \frac{1}{N}(W_N^{-kn})_{k,n}$ and therefore equation (2) provides a solution to the system of equations (1). This follows from the orthogonality properties of the exponentials. More precisely:

$$\sum_{n=0}^{N-1} W_N^{jn} = \begin{cases} N & \text{if } j = mN \\ 0 & \text{otherwise} \end{cases}$$

This follows from the formula of a geometric progression.

We multiply both sides of (3) by W_N^{nr} and we sum over n and we get

$$\sum_n x[n] W_N^{nr} = \frac{1}{N} \sum_n \sum_k W_N^{n(r-k)} y[k] =$$
$$= \sum_{k=0}^{N-1} \left(\frac{1}{N} \sum_{n=0}^{N-1} W_N^{n(r-k)} \right) y[k] = y[r].$$

We reverse the order of summation and use the orthogonality to get

$$\sum_n x[n] W_N^{nr} = y[r],$$

as desired. Thus we have proved that the inverse of the matrix $M = (W_N^{ij})_{i,j=0}^{N-1}$ is $M^{-1} = \frac{1}{N}\overline{M}$.

The formulas (1) and (2) are a very inefficient way to compute the DFT or the IDFT. The cost of multiplying a $N \times N$ matrix by a vector of length N is about N^2 operations which is prohibitive if N is big.

Finally, one last way of interpreting the DFT is through two different representations of polynomials of degree $N-1$. To begin with any data $x[0], \ldots, x[N-1]$ can be represented as a polynomial $p(z) = \sum_{n=0}^{N-1} x[n] z^n$. But any polynomial p of degree $N-1$ is equally determined by its values at any N different points. In particular p is completely determined from its values at the roots of unity $W_N^0, \ldots, W_N^{N-1}$. The values of the polynomial at these points are exactly the DFT of the coefficients. Thus the DFT is to convert a polynomial from its coefficients to its values at the roots of unity and conversely the IDFT is a conversion from the values of a polynomial to its coefficients. This is not only conceptually clarifying but it also provides a fast way to compute the DFT, that is the Fast Fourier Transform that we present now.

We will see how is the polynomial $A(x) = \sum_{k=0}^{n-1} a_k x^k$ converted from coefficient form to point value form. We assume that the degree

of n is a power of 2. If n is not a power of 2, then make it a power of 2 by padding the polynomial's higher degree coefficients with zeroes.

Let us define

$$A^{\text{even}}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1},$$
$$A^{\text{odd}}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}.$$

Here, $A^{\text{even}}(x)$ contains all even-indexed coefficients of $A(x)$ and $A^{\text{odd}}(x)$ contains all odd-indexed coefficients of $A(x)$.

It follows that

$$A(x) = A^{\text{even}}(x^2) + xA^{\text{odd}}(x^2).$$

So now the problem of evaluating $A(x)$ at the n complex n-th roots of unity, ie. at $W_n^0, W_n^1, \ldots, W_n^{n-1}$ reduces to

- Evaluating the $(n-1)$-degree polynomials $A^{\text{even}}(x^2)$ and $A^{\text{odd}}(x^2)$ at $W_n^0, W_n^1, \ldots, W_n^{n-1}$. (the divide part)
- Combine the two together to get $A(x)$. (the conquer part)

Now observe that $A^{\text{odd}}(x^2)$ and $A^{\text{even}}(x^2)$ have exactly the same value at x and at $-x$. Thus we only need to evaluate $A^{\text{odd}}(x)$ and $A^{\text{even}}(x)$ at the square of the roots of unity. But if n is even the square of the n-roots of unity are exactly the $n/2$-roots of unity. That is, we want to evaluate $A^{\text{odd}}(x)$ and $A^{\text{even}}(x)$ at the points $W_{n/2}^0, W_{n/2}^1, \ldots, W_{n/2}^{n/2-1}$. That means that each of the new polynomials of degree $n/2 - 1$ is evaluated at half the number of points. It is exactly the same problem as we started with, but with half the points and half the degree.

To combine them only requires n multiplications and additions.

So the pseudocode, from [**2**, Chapter 30], for the FFT would be like this

```
1  RECURSIVEFFT(a) {
2  n = length(a)
3  if n == 1 then return a // Base Case
4  w_n = e^{-2πi/n}
5  w = 1
6  a^even = (a_0, a_2, ..., a_{n-2})
7  a^odd = (a_1, a_3, ..., a_{n-1})
8  y^even = RECURSIVEFFT(a^even)
9  y^odd = RECURSIVEFFT(a^odd)
10 for k = 0 to n/2−1
11       y_k = y_k^even + w * y_k^odd
12       y_{k+n/2} = y_k^even − w * y_k^odd
13       w = w * w_n
14 return y;
15 }
```

Exercise 1. *Implement the FFT and the IFFT*

What is the cost of computing an FFT? Let us denote by $T(n)$ the cost (number of multiplications and additions) of computing the FFT

of a vector of length n. If we analyse the recursive formula we observe that the cost satisfies

$$T(n) = T(n/2) + T(n/2) + O(n).$$

The first $T(n/2)$ term comes from evaluating the FFT of the even part, the second comes from evaluating the FFT of the odd part and the $O(n)$ part is when we add the two pieces together. Thus $T(n) = 2T(n/2) + O(n)$. Of course $T(1) = 1$ and by induction we check then that $T(n) = O(n\log(n))$ (or we may apply the master theorem, see [**2**]).

A highly recommended lecture on the FFT by Erik Demaine is found on the MITOpenCourseware.

One inmediate application is the following. Suppose that we want to multiply two polynomials p and q and obtain the coefficients of $h = pq$. We may assume that both are of degree $N-1$ (by padding with zeros if necessary). The cost of the naive multiplicaton is $N \times N$. But we can do something better. Let us pad the polynomials with zeros up to length 2^k with $2^k >= 2N$. In the worst case scenario we only need to choose $2^k < 4N$. Now, we take the FFT of both polynomials. The cost is $O(N \log N)$. We now know the values of these two polynomials of degree $N-1$ in $2N$ points. But we can then know the values of $h(W_{2N}^j) = p(W_{2N}^j)q(W_{2N}^j)$. This only needs $2N$ multiplications. Finally h is of degree at most $2N-2$. Therefore if we take back the IFFT we obtain h. The total cost is $O(N \log N)$ operations!

Remark that multiplying two polynomials $a_0 + a_1 x + \cdots a_n x^n$ and $b_0 + b_1 x + \cdots + b_m$ gives the polynomial $c_0 + c_1 x + \cdots + c_{n+m} x^{n+m}$ with coefficients given by

$$c_k = \sum_{j=0}^{k} a_j b_{k-j},$$

which is exactly the convolution of the sequences a and b (padded with infinite zeros). That is $c = a \star b$.

Exercise 2. *A problem in additive combinatorics: Consider the set formed by the first N primes: $p_1, p_2, \ldots, p_N$. We want to know in how many ways we can choose three primes in this set such that they are three consecutive terms of an arithmetic progression.*

Meaning that, how many triplets (i, j, k) are there such that $1 \leq i < j < k \leq N$ and $p_j - p_i = p_k - p_j$.

So the triplets $(5, 11, 17)$, $(7, 157, 307)$, $(3, 5, 7)$ are valid as they are three consecutive terms of an arithmetic progression. But the triplets $(2, 5, 7)$, $(3, 7, 17)$ are not.

Write a program that computes the number of triplets formed by the first 100.000 primes.

Hint: Consider the polynomial $p(x) = \sum_{i=1}^{M} a_i x^i$ *where* $a_i = 1$ *if* i *is prime and zero otherwise and compute its square* $q = p^2$*. Write the solution in terms of the coefficients of* q*.*

Bibliography

[1] Ronald W. Schafer. Alan V. Oppenheim, *Discrete-time signal processing.*, Prentice Hall, 2010.

[2] Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein., *Introduction to Algorithms.*, MIT Press, 2009.

[3] Sidhant Bansal, *Tutorial on FFT/NTT - The tough made simple.*