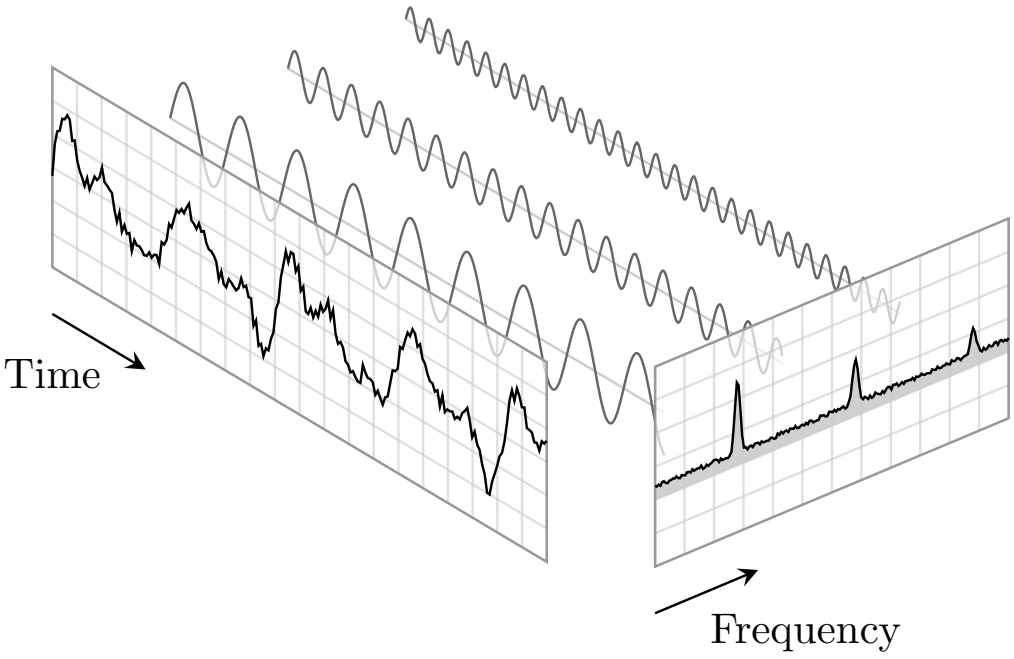
CHAPTER 5

Harmonic Analysis 101

In this lecture, we introduce the quantum Fourier transform, which is $O((\log N)^2)$, i.e., exponentially faster than the classical fast Fourier transform in $O(N \log N)$, but before we do so, we try to explain some harmonic analysis in general.



Harmonic analysis as we need it requires discrete samples and finite fields. The corresponding, perhaps seemingly obscure parts of harmonic analysis have also led to classical breakthroughs, such as multiplication of n-bit integers in time $O(n \log n)$ [Harvey and Van Der Hoeven, 2021] and multiplication of polynomials over finite fields [Harvey and Van Der Hoeven, 2022] in the same time.

1. Discrete Fourier Transform

The discrete Fourier transform maps an N-vector $\mathbf{x}$ complex numbers to an N-vector $\mathbf{X}$ of complex numbers:

$$X_k = \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}}, \tag{5.1}$$

up to a normalization $\frac{1}{\sqrt{N}}$. (This is sometimes called the analysis formula.) Let us assume $N = 2^n$ throughout, where n is a constant.

One way to think of the N-vector $\mathbf{x}$ is to see those as samples of a periodic function with period T, i.e., $f(t) = f(t+T)$. In particular, one would sample f uniformly at points $j\Delta t$, where $\Delta T = T/N$ and $n = 0, 1, ..., N-1$.

Alternatively, one could see discrete Fourier transform as a function on a group G, often a finite Abelian group. For any group G, and especially for cyclic groups $\mathbb{Z}_q$, it may be tempting to identify the group with its elements $g \in G$ and consider $f(g)$ only, or to identify the cyclic groups $\mathbb{Z}_q$ with the set $\{0, \ldots, q-1\}$ and modulo q addition. One could do much better, however, if one considers the group's symmetries. For a very nice introduction to Harmonic analysis on finite groups, see Peyré [2020].

Alternatively, one could see the analysis formula (5.1) as a matrix equation $\mathbf{X} = \mathbf{F}\mathbf{x}$. Thus, a discrete Fourier transform can be expressed as a so-called Vandermonde matrix (Sylvester, 1867),

$$\mathbf{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \cdots & \omega_N^{0\cdot(N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \cdots & \omega_N^{1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \cdots & \omega_N^{(N-1)\cdot(N-1)} \end{bmatrix} \tag{5.2}$$

where $\omega_N^{m,n} = e^{-i2\pi mn/N}$ and the mn is the usual product of the integers.

Notice that:

- because ω depends only on the the product of frequency m, and position n, the DFT $\mathbf{F}$ is symmetric. Notice that it is also unitary: $\mathbf{F}^{-1} = \mathbf{F}^*$ and $|\det(\mathbf{F})| = 1$.
- $\mathbf{X}$ is the inner product of $\mathbf{x}$ with the m-th row of $\mathbf{F}$. Conversely, $\mathbf{f}$ is a linear combination of the columns of $\mathbf{F}$, where the mth column is weighted by X_m.
- the vectors $u_m = \left[e^{\frac{i2\pi mn}{N}} \;\middle|\; n = 0, 1, \ldots, N-1 \right]^{\mathsf{T}}$ form an orthogonal basis over the set of N-dimensional complex vectors.
- $\mathbf{F}^2$ reverses the input, while $\mathbf{F}^4 = \mathbf{I}$. The eigenvalues satisfy: $\lambda^4 = 1$ and thus are the fourth roots of unity: $+1, -1, +i, or -i$.

1.1. The Hadamard Transform. We can define the 1×1 Hadamard transform $H_0 = 1$ as the identity, and then define H_m for $m > 0$ by:

$$H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix}.$$

Other than the normalization, the Hadamard matrices are made up of 1 and -1. Notice that Hadamard

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

is a discrete Fourier transform; indeed $\omega_N = e^{-i\pi} = -1$. Likewise for

$$H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

Notice that classically, we can compute the fast Hadamard transform algorithm in $O(n \log n)$ while performing only sign-flips. Quantumly, the Hadamard transform can be computed in time $O(1)$, in many commonly used gate sets.

1.2. The z-Transform. If you know the z-transform, notice that the X_k can also be seen as evaluation of the z-transform $X(z) = \sum_{j=0}^{N-1} x_j z^{-j}$ at points ω_N^{-j}, i.e., $X_j = X(z)_{z=\omega_N^{-j}}$.

1.3. Examples of Discrete Fourier Transform. Let us be more specific:

$$\mathbf{F}_0 = H_0 = 1$$

$$\mathbf{F}_1 = H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\mathbf{F}_2 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega_3^{2,2} & \omega_3^{2,3} \\ 1 & \omega_3^{3,2} & \omega_3^{3,3} \end{pmatrix}$$

Let us consider one particular DFT:

$$\mathbf{F}_3 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

Let us consider a simple example (cf. Wikipedia):

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2-i \\ -i \\ -1+2i \end{pmatrix}. \tag{5.3}$$

$$X_0 = e^{-i2\pi 0 \cdot 0/4} \cdot 1 + e^{-i2\pi 0 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 0 \cdot 2/4} \cdot (-i) + e^{-i2\pi 0 \cdot 3/4} \cdot (-1+2i) = 2 \tag{5.4}$$

$$X_1 = e^{-i2\pi 1 \cdot 0/4} \cdot 1 + e^{-i2\pi 1 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 1 \cdot 2/4} \cdot (-i) + e^{-i2\pi 1 \cdot 3/4} \cdot (-1+2i) = -2-2i \tag{5.5}$$

$$X_2 = e^{-i2\pi 2 \cdot 0/4} \cdot 1 + e^{-i2\pi 2 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 2 \cdot 2/4} \cdot (-i) + e^{-i2\pi 2 \cdot 3/4} \cdot (-1+2i) = -2i \tag{5.6}$$

$$X_3 = e^{-i2\pi 3 \cdot 0/4} \cdot 1 + e^{-i2\pi 3 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 3 \cdot 2/4} \cdot (-i) + e^{-i2\pi 3 \cdot 3/4} \cdot (-1+2i) = 4+4i \tag{5.7}$$

which yields:

$$\mathbf{X} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -2-2i \\ -2i \\ 4+4i \end{pmatrix}. \tag{5.8}$$

Equivalently:

$$\mathbf{X} = \begin{pmatrix} 1 \\ 2-i \\ -i \\ -1+2i \end{pmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \tag{5.9}$$

$$= \begin{pmatrix} 2 \\ -2-2i \\ -2i \\ 4+4i \end{pmatrix}. \tag{5.10}$$

2. Fast Fourier Transform

2.1. A Cartoon. A straightforward implementation of DFT as a matrix-vector product requires $O(N^2)$ operations. In the so-called fast Fourier transform (Cooley and Tukey, 1965), one requires only $O(N \log_2 N) = O(2^n n)$ operations. As we assume $N = 2^n$, we will present a variant known as the radix-2 decimation in time (DIT) algorithm.

This speedup is achieved by a divide-and-conquer approach, where we consider subsets of the initial sequence, take the DFT of these subsequences, and reconstruct the DFT of the original sequence from

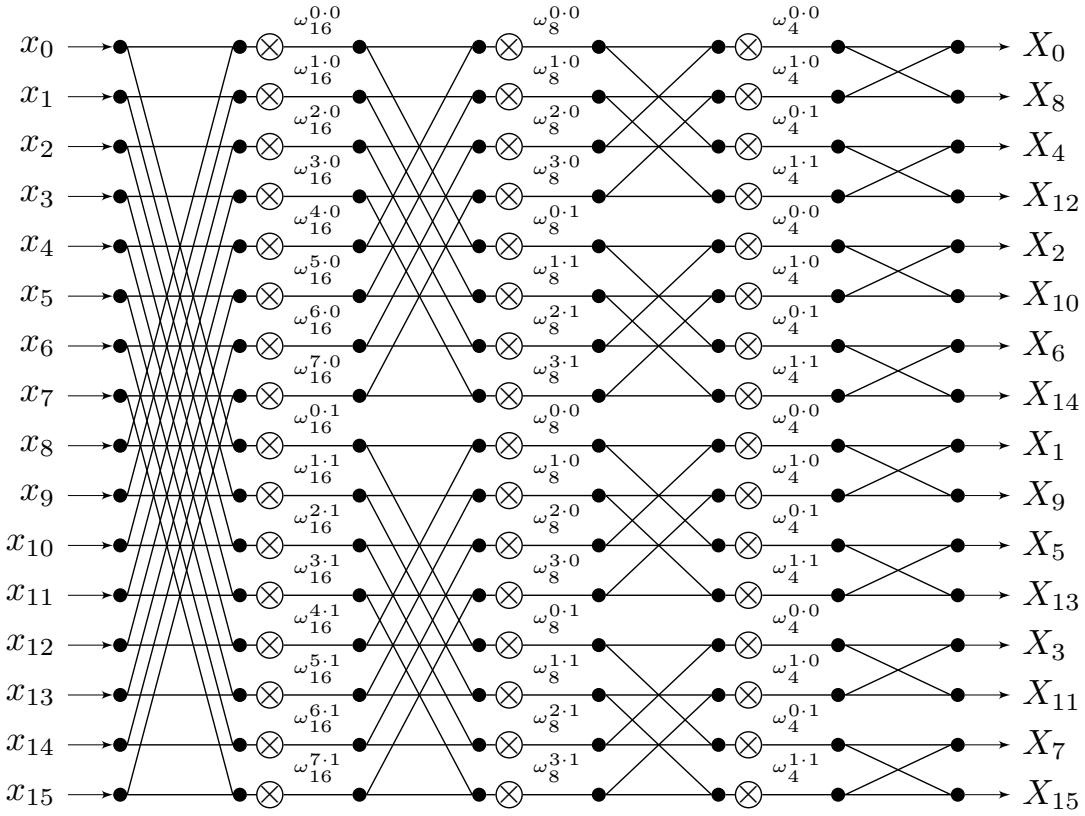
the results on the subsequences. One option is based on the following insight:

$$X_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}} \tag{5.11}$$

$$= \frac{1}{\sqrt{N}} \left(\sum_{\text{even } j} x_j \cdot e^{\frac{2\pi jki}{N}} + \sum_{\text{odd } j} x_j \cdot e^{\frac{2\pi (j-1)ki}{N}} \right) \tag{5.12}$$

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{\text{even } j} x_j \cdot e^{\frac{2\pi (j/2)ki}{N/2}} \right) + \left(\frac{1}{\sqrt{N/2}} \sum_{\text{odd } j} x_j \cdot e^{\frac{2\pi (j-1)/2ki}{N/2}} \right) \tag{5.13}$$

The divide-and-conquer approach can be illustrated on $N = 16$ with the following cartoon.



If you know the z-transform, you should see that $X(z) = \sum_{j=0}^{N-1} x_j z^{-j} = \sum_{l=0}^{r-1} \sum_{j \in \mathbf{I}_l} x_j z^{-j}$ for some partition $\mathbf{I}$ of $\{0, 1, \ldots, N-1\}$ into r subsets, and that one can also normalise the terms. This way, one can define a variety of recursions similar to the one above, as long as the subset are chosen to be similar to the initial sequence in terms of their periodicity. This is very nicely explained in Duhamel and Vetterli [1990].

Alternatively, Camps et al. [2021] sees the Fast Fourier Transform as a certain matrix factorization. This is both important to understand FFT, but also to understand the QFT later. In particular, the $2^n \times 2^n$ DFT matrix $\mathbf{F}_n$ can be factored as:

$$\mathbf{F} = \mathbf{P}_n \mathbf{A}_n^{(0)} \mathbf{A}_n^{(1)} \cdots \mathbf{A}_n^{(n-1)}, \tag{5.14}$$

where,

- $\mathbf{P}_n$ is some permutation matrix
- $\mathbf{A}_n^{(k)} = \mathbf{I}_{n-k-1} \otimes \mathbf{B}_{k+1}$,

- $\mathbf{B}_{k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_k & \mathbf{I}_k \\ \mathbf{\Omega}_k & -\mathbf{\Omega}_k \end{bmatrix}$ for some $k \times k$ identity matrix $\mathbf{I}_k$
- $\mathbf{\Omega}_{2^n}$ is a $2^n \times 2^n$ diagonal matrix:

$$\mathbf{\Omega}_{2^n} := \begin{bmatrix} \omega_{2^{n+1}}^0 & & & \\ & \omega_{2^{n+1}}^1 & & \\ & & \ddots & \\ & & & \omega_{2^{n+1}}^{2^n-1} \end{bmatrix}, \tag{5.15}$$

  where $\omega_{2^{n+1}}$ is $e^{\frac{-2\pi i}{2^{n+1}}}$ as before.
- $\oplus$ now (confusingly) denotes the Kronecker product.

Notice that each matrix $\mathbf{A}_n^{(i)}$ has two non-zero elements on every row. Consequently, the matrix-vector product $\mathbf{A}_n^{(i)}\mathbf{x}$ can be computed in $O(2^n)$ operations, resulting in $O(2^n n)$ operations, when one includes the permutation.

3. Quantum Fourier Transform

In general, $\mathbf{F}$ is an $N \times N$ unitary matrix, and thus we can implement it on a quantum computer as an n-qubit unitary for $N = 2^n$. There, it maps N-dimensional vector of amplitudes to N-dimensional vector of amplitudes. This is called quantum Fourier transform (QFT). Shor [1994], Kitaev [1995] presented the first polynomial, $O(n^2)$ quantum algorithms for QFT over certain finite fields and arbitrary finite Abelian groups, respectively. This is exponentially faster than the classical fast Fourier transform, which takes $O(N \log N)$ steps.

Recall that the DFT is:

$$X_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}}.$$

In contrast, the QFT on an orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ is a linear operator:

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi jki}{N}} |k\rangle.$$

An alternative representation of the QFT utilizes the *product form*:

$$|j_1, j_2, \ldots, j_n\rangle \to \frac{\left(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle\right)\left(|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle\right)}{2^{n/2}},$$

where $|j_1, j_2, \ldots, j_n\rangle$ is a binary representation of a basis state j and $0.j_1 j_2 \cdots j_n$ is a notation for binary fraction $j_1/2 + j_2/4 \cdots j_n/2^{n+1}$. This is actually easy enough to derive:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}} |k\rangle \tag{5.16}$$

$$\frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi jki}{2^n}} |k\rangle \tag{5.17}$$

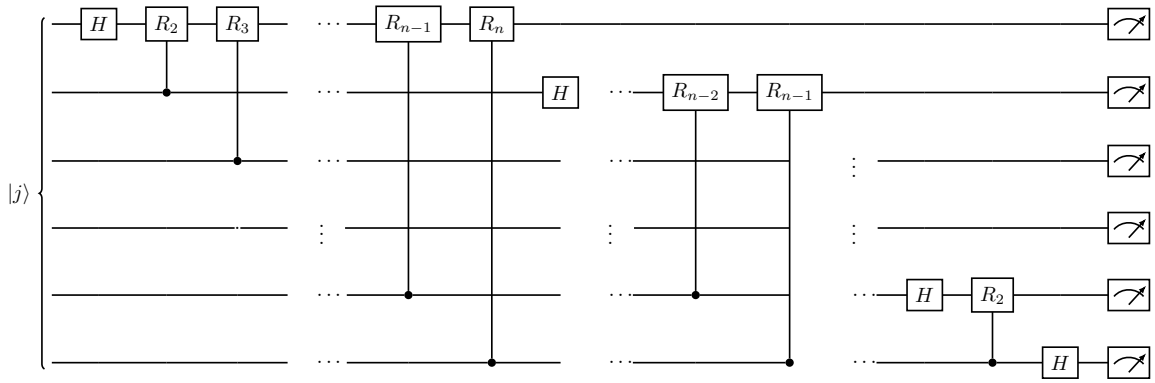
$$\frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi j(\sum_{l=1}^{n} k_l 2^{-l})i} |k_1 k_2 \ldots k_n\rangle \tag{5.18}$$

$$\frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi j k_l 2^{-l} i} |k_l\rangle \tag{5.19}$$

$$\frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi j k_l 2^{-l} i} |k_l\rangle \right] \tag{5.20}$$

$$\frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi j 2^{-l} i} |1\rangle \right] \tag{5.21}$$

A simplistic illustration of the quantum circuit for QFT, omitting swaps at the end and normalization:



Its derivation is very nicely given in Camps et al. [2021]. The key to its understanding is the phase kickback, which we have seen earlier.

3.1. Even Faster QFT. Cleve and Watrous [2000], Hales and Hallgren [2000], and others improved this to $O(n \log n)$ depth, if one allows for some error. This is based on the the realization that R_s for $s \ll \log n$ are very close to the identity and can be omitted.

Bibliography

Daan Camps, Roel Van Beeumen, and Chao Yang. Quantum fourier transform revisited. *Numerical Linear Algebra with Applications*, 28(1):e2331, 2021.

R. Cleve and J. Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536, 2000. doi: 10.1109/SFCS.2000.892140.

Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299, 1990.

Lisa Hales and Sean Hallgren. An improved quantum Fourier transform algorithm and applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 515–525. IEEE, 2000.

David Harvey and Joris Van Der Hoeven. Integer multiplication in time $o(n \log n)$. *Annals of Mathematics*, 193(2):563–617, 2021.

David Harvey and Joris Van Der Hoeven. Polynomial multiplication over finite fields in time $o(n \log n)$. *Journal of the ACM (JACM)*, 69(2):1–40, 2022.

A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem, 1995.

Gabriel Peyré. The discrete algebra of the Fourier transform. 2020. A draft textbook at `https://mathematical-tours.github.io/daft-sources/DAFT-EN.pdf`.

Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.