# Section 6: The Discrete Fourier Transform

May 1, 2019

The central object of discussion in this section is the so-called Discrete Fourier Transform. The DFT is one of four types of Fourier transforms which together form the basis of a much more richer subject generally going under the name of Fourier analysis. In Fourier analysis, one is interested in how functions can be decomposed or approximated by sums of simpler trigonometric functions. Historically, this subject arose from the study of Fourier series (one the other variants of Fourier transforms), and is named after Joseph Fourier, who showed that representing a function as a sum of trigonometric functions greatly simplifies the solution of the heat equation. Nowadays, Fourier analysis plays a central role in many practical applications of science and engineering. Its importance cannot be dismissed by almost every technological device used by mankind today. A quick google search on the applications of Fourier analysis should convince you why. For those who like music (who doesn't?) have a look here.

Nevertheless, this set of notes does not really intend looking into the applications of Fourier analysis in detail. Rather, we will be looking at the subject from a more the computational/numerical analysis point of view. There are in particular two goals in mind: *a)* describe what the DFT *is* in its bare essence mathematically, and *b)* discuss the efficient computation of the DFT through the Fast Fourier Transform algorithm. Keep in mind that to gain a full awareness on the power of Fourier analysis, this piece of text simply does not do justice to the richness of the underlying subject (there are many textbooks that do a better job at this). You will get to learn this however

through the course of time as you inevitebly will encounter this subject again in your other courses.

# 1 Preliminaries

Before we move on to the main topic, it is useful to discuss some preliminaries first.

## 1.1 A quick refresher on complex number

Define: $i := \sqrt{-1}$. A complex number $z = a + bi$ consist of a real part $a$ and an imaginary part $b$. The conjugate $\bar{z} = a - bi$ of a complex $z$ flips the sign of the imaginary part while preserving the sign of the real part. By using Euler's famous formula:

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

we may write a complex number in terms of polar coordinates: $z = re^{i\theta}$, where the two representation are related by $r^2 = a^2 + b^2$ and $\tan(\theta) = \frac{b}{a}$. Complex conjugation $\bar{z} = re^{-i\theta}$ in polar coordinates amounts to changing the sign of the phase angle $\theta$. All of these concepts should be familiar to you.

## 1.2 Matrix algebra with complex numbers

Notice it requires two parameters to describe a complex number. It is possible to write the multiplication of two complex numbers as matrix vector multplication. Compare the following two expressions:

$$zw = (a + bi)(c + di) = (ac - bd) + (ad + bc)i = (c + di)(a + bi) = wz$$

and

$$\begin{bmatrix} ac - bd \\ ad + bc \end{bmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} c & -d \\ d & d \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

and notice the similarities. The above observations generalize to higher dimensions as well. Indeed, suppose that $Z$ is a complex-valued matrix and $w$ is a complex valued vector. Writing $Z = A + iB$ and $w = c + id$ (with $A, B$ and $c, d$ as real matrices and vectors), the following equivalence

$$Zw \equiv \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix}$$

shows that matrix-vector multiplication of complex numbers can be written as matrix-vector multiplies of real numbers.

## 1.3 Unitary matrices

In the complex field, the inner (or dot) product also undergoes a slight modificaion. If $a, b$ are two complex valued vectors, then the dot product is defined as:

$$\langle a, b \rangle := a^* b = \sum_{i=1}^{n} \bar{a}_i b_i$$

The notation $a^*$ denotes the conjugate transpose. In general, if $A$ denotes a $m$ by $n$ matrix then its conjugate $A^*$ is simply its transpose but with all its entries conjugated in addition, e.g.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}^* = \begin{bmatrix} \bar{a}_{11} & \bar{a}_{21} & \bar{a}_{31} \\ \bar{a}_{12} & \bar{a}_{22} & \bar{a}_{32} \end{bmatrix}.$$

A unitary matrix $U$ is a square complex-valued matrix whose conjugate transpose is also his inverse. That is,

$$U^* U = U U^* = I$$

Unitary matrices generalize the notion of orthogonal matrices. In the special case when all entries of the unitary matrix are real, $U$ necessarily becomes an orthogonal matrix.

## 1.4 Trigonometric polynomials and orthogonality

Let:

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$$

denote a degree $n$ polynomial expressed in the monomial basis. Naturally, this polynomial is defined for every complex number $x \in \mathbb{C}$. Suppose we restrict the domain of our polynomial to just the unit circle defined on the complex plane. We can do this by the substitution:

$$x = e^{i2\pi\theta},$$

where $\theta$ is the variable that parametrizes the unit circle. What we obtain is a so-callled trigonometric polynomial:

$$T_n(\theta) = p_n(e^{i2\pi\theta}) = c_0 + c_1 e^{i2\pi\theta} + c_2 e^{i2(2\pi\theta)} + \ldots + c_n e^{in(2\pi\theta)}$$

The name trigonometric polynomial is in reference to Eulers formula $e^{2\pi i\theta} = \cos(2\pi\theta) + i\sin(2\pi\theta)$. When $\theta$ is allowed to range in $\mathbb{R}$, trigonometric polynomials are complex valued periodic functions (with period 1).

Note that trigonometric polynomials are not only generated from polynomials. We can also consider negative exponents:

$$p_{-m,n}(x) = c_{-m}x^{-m} + \ldots + c_{-2}x^{-2} + c_{-1}x^{-1} + c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$$

A restriction to the unit circle would again yield a trigonometric polynomial:

$$T_{-m,n}(\theta) = c_{-m}e^{-im(2\pi\theta)} + \ldots + c_{-2}e^{-i2(2\pi\theta)} + c_{-1}e^{-i2\pi\theta} + c_0 + c_1 e^{i2\pi\theta} + c_2 e^{i2(2\pi\theta)} + \ldots + c_n e^{in(2\pi\theta)}$$

An interesting thing happens when we restrict these functions to the unit circle on the complex plane. With respect to the inner product:

$$\langle a(\theta), b(\theta)\rangle := \int_0^1 \bar{a}(\theta)b(\theta)d\theta$$

The terms $e^{ik(2\pi\theta)}$ for $k \in \mathbb{Z}$ (i.e. integers) are orthonormal. This was not the case when we for example restricted the monomial basis to the unit interval $[0, 1)$ on the real line.

## 2 The Definition of the DFT

Before we get into what the DFT does, let us first define it. Suppose we have a complex-valued vector $x \in \mathbb{C}^n$. You may think of this vector describing some time signal sampled at a specific frequency, even though this context is really not necessary to define the DFT. The DFT is simply the linear transformation which maps $x \in \mathbb{C}^n$ to another vector $\hat{x} \in \mathbb{C}^n$:

$$\hat{x}[k] = \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} e^{-\frac{2\pi i}{n} kl} x[k], \quad k = 0, 1, \ldots, n-1.$$

This transformation is invertible, and the inverse is given by:

$$x[k] = \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} e^{\frac{2\pi i}{n} kl} \hat{x}[k], \quad k = 0, 1, \ldots, n-1.$$

Call:

$$\omega_n = e^{-\frac{2\pi i}{n}}$$

In matrix notation, we may write the transform as:

$$\hat{x} = F_n x = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[n-1] \end{bmatrix}$$

Similarly the inverse transform is given by:

$$x = F_n^{-1} \hat{x} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \bar{\omega}_n & \bar{\omega}_n^2 & \cdots & \bar{\omega}_n^{n-1} \\ 1 & \bar{\omega}_n^2 & \bar{\omega}_n^4 & \cdots & \bar{\omega}_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \bar{\omega}_n^{n-1} & \bar{\omega}_n^{2(n-1)} & \cdots & \bar{\omega}_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} \hat{x}[0] \\ \hat{x}[1] \\ \hat{x}[2] \\ \vdots \\ \hat{x}[n-1] \end{bmatrix}$$

Notice that the inverse DFT formula looks very similar to the DFT itself. It is, in fact, its conjugate transpose: conjugate transpose.

$$F_n^{-1} = F_n^*$$

The DFT is an *unitary* transformation.

# 3 What does the DFT describe?

There are many ways of deriving the DFT matrix. Here, we will take what we have learned already about polynomial interpolation and go from there.

## 3.1 Derivation from the Vandermonde matrix

Recall the Vandermonde matrix associated with the monomial basis:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

We stated back then that if we picked our points $x_1, x_2, \ldots, x_n$ uniquely, i.e. without choosing the same point twice, this matrix would be invertible. Subsequently, solving for the coefficient $a_1, a_2, \ldots, a_n$ would find us the unique polynomial of degree $n-1$ interpolating the values $y_1, y_2, \ldots, y_n$ at the points $x_1, x_2, \ldots, x_n$. In topic of polynomial interpolation the points $x_1, x_2, \ldots, x_n$ were all assumed to lie to some interval on the real line. However, nobody really is stopping us from choosing point on the complex plane!

Now suppose we choose the following points. Consider the equation:

$$x^n = 1$$

This equation has solutions $n$ solutions:

$$x_k = e^{i\frac{2\pi}{n}(k-1)}, \quad k = 1, \ldots n.$$

Plugging in these points, referred to as *roots of unity*, into the Vandermonde matrix yields:

$$V_{n-1}(1, e^{i\frac{2\pi}{n}}, \ldots, e^{i\frac{2\pi}{n}(n-1)}) = \sqrt{n}F_n^*$$

In other words, the DFT matrix will find us the coefficients for the unique interpolating polynomial in the case when the interpolation points are the roots of unity. However, thanks to Euler's magnificent formula, we also have another interpretation for the DFT in terms of trigonometric interpolation.

## 3.2 Trigonometric interpolation and aliasing

Suppose we have a function defined on a circle of length 1 (or equivalently a periodic function on the real line with period 1). This (complex-valued) function is sampled uniformly on the circle at let's say $\theta_k = \frac{k}{n}$ with values $y_k$ for $k = 0, 1, \ldots, n-1$. If we would like to interpolate the following the trigonometric polynomial through the sample/data points:

$$T_{n-1}(\theta) = \sqrt{n}\left[c_0 + c_1 e^{2\pi i\theta} + c_2 e^{2\pi i 2\theta} + \ldots + c_{n-1} e^{2\pi i(n-1)\theta}\right] \tag{1}$$

Then the coefficients $c = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix}^T$ are found by applying the DFT:

$$c = F_n y$$

where $y = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{bmatrix}^T$. Yes. It is true that this is the unique trigonometric polynomial of degree $n-1$ (i.e. a trigonometric polynomial generated from polynomial of degree $n-1$) that passes through all these $n$ points. But there is catch, it is not the only trigonometric poynomials which fits the data exactly. Indeed, plugging in $\theta = \frac{k}{n}$

$$\sqrt{n}\left[c_0 + c_1 e^{2\pi i \frac{k}{n}} + c_2 e^{2\pi i 2\frac{k}{n}} + \ldots + c_{n-1} e^{2\pi i(n-1)\frac{k}{n}}\right]$$

and observing that:

$$e^{2\pi i l \frac{k}{n}} = e^{2\pi i(l+mn)\frac{k}{n}}, \quad m \in \mathbb{Z}$$

and rewriting:

$$\sqrt{n}\left[c_0 e^{2\pi i m_0 n \frac{k}{n}} + c_1 e^{2\pi i(1+m_1 n)\frac{k}{n}} + c_2 e^{2\pi i(2+m_2 n)\frac{k}{n}} + \ldots + c_{n-1} e^{2\pi i(n-1+m_{n-1} n)\frac{k}{n}}\right]$$

shows that the data points could have as well been generated from the signal:

$$\sqrt{n}\left[c_0 e^{2\pi i m_0 n\theta} + c_1 e^{2\pi i(1+m_1 n)\theta} + c_2 e^{2\pi i(2+m_2 n)\theta} + \ldots + c_{n-1} e^{2\pi i(n-1+m_{n-1} n)\theta}\right]$$

where $m_0, m_1, \ldots, m_{n-1} \in \mathbb{Z}$ can be chosen arbitrary.

There is an ambiguity. What he have just stumbled upon is the issue of aliasing: if a continuous signal is sampled information will be lost. Reconstructing the continuous signal from the data points will always give rise to

an ambiguity, i.e. there are abitrary ways of filling in the holes. Think for example when a sinusoidal wave get undersampled. From a practical point of view, it makes sense to associate always the lowest frequency signal to the data points. This means that all the complex exponentials $e^{2\pi ik\theta}$ in (1) with $k > n/2$ should actually be replaced with $e^{-2\pi ik\theta}$

# 4 A fast algorithm for computing the DFT

Computing the DFT of a signal/vector requires evaluating a matrix vector product. We know that this operation generally requires $\mathcal{O}(n^2)$ operations, with $n$ being the size of signal/vector. For relatively small signals, this complexity is manageable, but for most applications with quite large signals, it can become prohibitely expensive to evaluate a DFT. The great news is that the complexity can actually be improved to $\mathcal{O}(n \log n)$ by exploited some special structural properties of the DFT matrix. The fact that things may be computed $\mathcal{O}(n \log n)$ is also the reason why the DFT is so widely used. The plethora of algorithms that give rise to this level of complexity is referred to as Fast Fourier Transform.

Let us now describe the original FFT algorithm, as discovered by J. W. Cooley and John Tukey. Examine the DFT associated with $n = 8$, i.e.

$$\sqrt{8}F_8 x = \begin{bmatrix} \omega_8^{0\cdot 0} & \omega_8^{1\cdot 0} & \omega_8^{2\cdot 0} & \omega_8^{3\cdot 0} & \omega_8^{4\cdot 0} & \omega_8^{5\cdot 0} & \omega_8^{6\cdot 0} & \omega_8^{7\cdot 0} \\ \omega_8^{0\cdot 1} & \omega_8^{1\cdot 1} & \omega_8^{2\cdot 1} & \omega_8^{3\cdot 1} & \omega_8^{4\cdot 1} & \omega_8^{5\cdot 1} & \omega_8^{6\cdot 1} & \omega_8^{7\cdot 1} \\ \omega_8^{0\cdot 2} & \omega_8^{1\cdot 2} & \omega_8^{2\cdot 2} & \omega_8^{3\cdot 2} & \omega_8^{4\cdot 2} & \omega_8^{5\cdot 2} & \omega_8^{6\cdot 2} & \omega_8^{7\cdot 2} \\ \omega_8^{0\cdot 3} & \omega_8^{1\cdot 3} & \omega_8^{2\cdot 3} & \omega_8^{3\cdot 3} & \omega_8^{4\cdot 3} & \omega_8^{5\cdot 3} & \omega_8^{6\cdot 3} & \omega_8^{7\cdot 3} \\ \omega_8^{0\cdot 4} & \omega_8^{1\cdot 4} & \omega_8^{2\cdot 4} & \omega_8^{3\cdot 4} & \omega_8^{4\cdot 4} & \omega_8^{5\cdot 4} & \omega_8^{6\cdot 4} & \omega_8^{7\cdot 4} \\ \omega_8^{0\cdot 5} & \omega_8^{1\cdot 5} & \omega_8^{2\cdot 5} & \omega_8^{3\cdot 5} & \omega_8^{4\cdot 5} & \omega_8^{5\cdot 5} & \omega_8^{6\cdot 5} & \omega_8^{7\cdot 5} \\ \omega_8^{0\cdot 6} & \omega_8^{1\cdot 6} & \omega_8^{2\cdot 6} & \omega_8^{3\cdot 6} & \omega_8^{4\cdot 6} & \omega_8^{5\cdot 6} & \omega_8^{6\cdot 6} & \omega_8^{7\cdot 6} \\ \omega_8^{0\cdot 7} & \omega_8^{1\cdot 7} & \omega_8^{2\cdot 7} & \omega_8^{3\cdot 7} & \omega_8^{4\cdot 7} & \omega_8^{5\cdot 7} & \omega_8^{6\cdot 7} & \omega_8^{7\cdot 7} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

Let us now re-order the columns of $F_8$ by first placing the odd columns next to each other and then all the even columns. The permutation matrix required

for this is:

$$P_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Notice what we get:

$$(\sqrt{8}F_8P_8)(P_8^T x) = \begin{bmatrix} \omega_8^{0\cdot0} & \omega_8^{2\cdot0} & \omega_8^{4\cdot0} & \omega_8^{6\cdot0} & \omega_8^{1\cdot0} & \omega_8^{3\cdot0} & \omega_8^{5\cdot0} & \omega_8^{7\cdot0} \\ \omega_8^{0\cdot1} & \omega_8^{2\cdot1} & \omega_8^{4\cdot1} & \omega_8^{6\cdot1} & \omega_8^{1\cdot1} & \omega_8^{3\cdot1} & \omega_8^{5\cdot1} & \omega_8^{7\cdot1} \\ \omega_8^{0\cdot2} & \omega_8^{2\cdot2} & \omega_8^{4\cdot2} & \omega_8^{6\cdot2} & \omega_8^{1\cdot2} & \omega_8^{3\cdot2} & \omega_8^{5\cdot2} & \omega_8^{7\cdot2} \\ \omega_8^{0\cdot3} & \omega_8^{2\cdot3} & \omega_8^{4\cdot3} & \omega_8^{6\cdot3} & \omega_8^{1\cdot3} & \omega_8^{3\cdot3} & \omega_8^{5\cdot3} & \omega_8^{7\cdot3} \\ \omega_8^{0\cdot4} & \omega_8^{2\cdot4} & \omega_8^{4\cdot4} & \omega_8^{6\cdot4} & \omega_8^{1\cdot4} & \omega_8^{3\cdot4} & \omega_8^{5\cdot4} & \omega_8^{7\cdot4} \\ \omega_8^{0\cdot5} & \omega_8^{2\cdot5} & \omega_8^{4\cdot5} & \omega_8^{6\cdot5} & \omega_8^{1\cdot5} & \omega_8^{3\cdot5} & \omega_8^{5\cdot5} & \omega_8^{7\cdot5} \\ \omega_8^{0\cdot6} & \omega_8^{2\cdot6} & \omega_8^{4\cdot6} & \omega_8^{6\cdot6} & \omega_8^{1\cdot6} & \omega_8^{3\cdot6} & \omega_8^{5\cdot6} & \omega_8^{7\cdot6} \\ \omega_8^{0\cdot7} & \omega_8^{2\cdot7} & \omega_8^{4\cdot7} & \omega_8^{6\cdot7} & \omega_8^{1\cdot7} & \omega_8^{3\cdot7} & \omega_8^{5\cdot7} & \omega_8^{7\cdot7} \end{bmatrix} \begin{bmatrix} x[0] \\ x[2] \\ x[4] \\ x[6] \\ x[1] \\ x[3] \\ x[5] \\ x[7] \end{bmatrix}$$

$$= \left( \begin{bmatrix} \omega_4^{0\cdot0} & \omega_4^{1\cdot0} & \omega_4^{2\cdot0} & \omega_4^{3\cdot0} \\ \omega_4^{0\cdot1} & \omega_4^{1\cdot1} & \omega_4^{2\cdot1} & \omega_4^{3\cdot1} \\ \omega_4^{0\cdot2} & \omega_4^{1\cdot2} & \omega_4^{2\cdot2} & \omega_4^{3\cdot2} \\ \omega_4^{0\cdot3} & \omega_4^{1\cdot3} & \omega_4^{2\cdot3} & \omega_4^{3\cdot3} \end{bmatrix} \begin{bmatrix} \omega_8^0 & & & \\ & \omega_8^1 & & \\ & & \omega_8^2 & \\ & & & \omega_8^3 \end{bmatrix} \begin{bmatrix} \omega_4^{0\cdot0} & \omega_4^{1\cdot0} & \omega_4^{2\cdot0} & \omega_4^{3\cdot0} \\ \omega_4^{0\cdot1} & \omega_4^{1\cdot1} & \omega_4^{2\cdot1} & \omega_4^{3\cdot1} \\ \omega_4^{0\cdot2} & \omega_4^{1\cdot2} & \omega_4^{2\cdot2} & \omega_4^{3\cdot2} \\ \omega_4^{0\cdot3} & \omega_4^{1\cdot3} & \omega_4^{2\cdot3} & \omega_4^{3\cdot3} \end{bmatrix} \right) \begin{bmatrix} x[0] \\ x[2] \\ x[4] \\ x[6] \end{bmatrix}$$

$$- \left( \begin{bmatrix} \omega_4^{0\cdot0} & \omega_4^{1\cdot0} & \omega_4^{2\cdot0} & \omega_4^{3\cdot0} \\ \omega_4^{0\cdot1} & \omega_4^{1\cdot1} & \omega_4^{2\cdot1} & \omega_4^{3\cdot1} \\ \omega_4^{0\cdot2} & \omega_4^{1\cdot2} & \omega_4^{2\cdot2} & \omega_4^{3\cdot2} \\ \omega_4^{0\cdot3} & \omega_4^{1\cdot3} & \omega_4^{2\cdot3} & \omega_4^{3\cdot3} \end{bmatrix} \begin{bmatrix} \omega_8^0 & & & \\ & \omega_8^1 & & \\ & & \omega_8^2 & \\ & & & \omega_8^3 \end{bmatrix} \begin{bmatrix} \omega_4^{0\cdot0} & \omega_4^{1\cdot0} & \omega_4^{2\cdot0} & \omega_4^{3\cdot0} \\ \omega_4^{0\cdot1} & \omega_4^{1\cdot1} & \omega_4^{2\cdot1} & \omega_4^{3\cdot1} \\ \omega_4^{0\cdot2} & \omega_4^{1\cdot2} & \omega_4^{2\cdot2} & \omega_4^{3\cdot2} \\ \omega_4^{0\cdot3} & \omega_4^{1\cdot3} & \omega_4^{2\cdot3} & \omega_4^{3\cdot3} \end{bmatrix} \right) \begin{bmatrix} x[1] \\ x[3] \\ x[5] \\ x[7] \end{bmatrix}$$

In condensed form:

$$(\sqrt{8}F_8P_8)(P_8^T x) = \begin{bmatrix} \sqrt{4}F_4 & \Omega_4(\sqrt{4}F_4) \\ \sqrt{4}F_4 & -\Omega_4(\sqrt{4}F_4) \end{bmatrix} \begin{bmatrix} x_{\text{odd}} \\ x_{\text{even}} \end{bmatrix}, \quad \Omega_4 = \begin{bmatrix} \omega_8^0 & & & \\ & \omega_8^1 & & \\ & & \omega_8^2 & \\ & & & \omega_8^3 \end{bmatrix}$$

The shenanigans we did above is not only applicable to $n = 8$, but for any $n$ divisible by 2. In general, we have:

$$(\sqrt{n}F_nP_n)(P_n^T x) = \begin{bmatrix} \sqrt{n/2}F_{n/2} & \Omega_{n/2}(\sqrt{n/2}F_{n/2}) \\ \sqrt{n/2}F_{n/2} & -\Omega_{n/2}(\sqrt{n/2}F_{n/2}) \end{bmatrix} \begin{bmatrix} x_{\text{odd}} \\ x_{\text{even}} \end{bmatrix} \tag{2}$$

where:

$$\Omega_{n/2} = \begin{bmatrix} \omega_{2n}^0 & & & \\ & \omega_{2n}^1 & & \\ & & \ddots & \\ & & & \omega_{2n}^{n-1} \end{bmatrix}$$

is often referred to as the "twiddle factors". Let us consider the significance of what has just been derived. Suppose for some reason we were already given (by God?) on our plate the DFT's $F_{n/2}x_{\text{odd}}$ and $F_{n/2}x_{\text{even}}$ in our hands. Equation (2) tells us that with little effort we can reconstruct DFT $F_n x$ from those smaller DFTs. In $\mathcal{O}(n)$ operations to be exact. Indeed, $\Omega_{n/2}$ is a diagonal matrix after all.

We can exploit this property to design a fast algorithm to evaluate the DFT. For the special case when $n = 2^N$, we can repeatedly apply (2) until we have divided the vector $x$ into its scalar component. The DFT can be assembled from those basic components. There are $N$ such assemblies in total. Each assembly will take $\mathcal{O}(n)$. So overall, we have $\mathcal{O}(nN)$ or $\mathcal{O}(n \log n)$ operation. This is so much faster than $\mathcal{O}(n^2)$. As you already know, it is even faster than $\mathcal{O}(n^{1+\epsilon})$ for any $\epsilon > 0$.

# 5 Convolution