FOURIER - GENIUS OR TERRORISTE?

Samir Chettri [1]

[1] HUGHES-STX, Kanazawa University, GST, UMBC

In this memo we discuss certain aspects of Fourier analysis. This is by no means a complete document. However, with these notes and some of the references the reader should be able understand key computational aspects of the Fourier transform. In a later memo we will talk about applications of the Fourier transform to image processing.

The life of Joseph

By all accounts Fourier led an adventurous life. Born in 1768 to a master tailor, he was orphaned at the age of 10. Through his intellect he gained a place in the local Benedictine school where he did very well.

When old enough to be conscripted he is said to have wanted to join the French army as a member of its elite artillery corps. He was turned down since he did not have royal blood in him. This incident may have caused him to take up with the revolution that brought "liberty, equality and fraternity" to France.

Naturally, when the new Republic seemed to be in trouble the definition of treachery was extended and changed. During these troubled times no one was safe including Fourier who was arrested, released, and arrested again. He was lucky to get away with his head still attached to his neck (re: guillotine) though he remained a marked man during those years with the label "terroriste" attached to his name.

At various times in his life he held governmental positions under King Louis and Napolean. The fact that he escaped being beheaded is probably a testament to his diplomatic talents. Among his achievements during his "government" period is his contribution to Egyptology (he encouraged Champollion who eventually deciphered the Rosetta stone, which was the key to the heiroglyphics). During this period he is said to have turned down free lunches in return for granting fat government contracts, whence the saying *non existat prandium gratuitum*.

When he turned 21, Fourier lamented that at his age Newton and Pascal had acquired many claims to immortality. At the age of 35 Fourier was still a footnote in the annals of science. This changed with his introduction of the Fourier series which he used to solve the heat equation. A little later he introduced the Fourier integral. At first, his methods were not accepted by Laplace and Lagrange - both of whom were towering figures at the time. However, through his intellect and perseverance he proved himself right and eventually became a grand old man of French sciences.

Introduction to continuous and discrete Fourier transforms

In this section we discuss the continuous and discrete Fourier transforms (FT) in one and two dimensions respectively. Using the arguments presented here the FT can be extended to more than two dimensions. Derivations of these formulae will not be given, though the interested reader is referred to [1].

Continuous domain

Let f be a function of x. The Fourier transform (FT) of $f(x)$ is defined as:

$$\mathcal{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}dx \qquad (1)$$

Note that we are using electrical engineering terminology by defining $j = \sqrt{-1}$.

The inverse FT (IFT) is defined as

$$\mathcal{F}^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ux}du \qquad (2)$$

Both the FT and the IFT form a *Fourier transform pair*. When speaking about Fourier transforms one should define *both* equations.

One thing to note about the FT is that the input function $f(x)$ can be either real or complex and that in general the transformation results in complex numbers. Thus we can represent $F(u)$ as

$$F(u) = R(u) + jI(u)$$

or in complex polar form as

$$F(u) = |F(u)|\, e^{j\phi(u)}.$$

Here $|F(u)| = [R(u)^2 + I(u)^2]^{1/2}$ is the *Fourier spectrum* and $\phi(u) = \tan^{-1}[I(u)/R(u)]$ is the *phase angle*. The *power spectrum* is defined as $P(u) = |F(u)|^2$.

We can now extend these definitions to the two variable case. Let $f(x,y)$ be a function of x and y. The two dimensional FT is defined as:

$$\mathcal{F}\{f(x,y)\} = F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy \tag{3}$$

and the inverse FT (IFT) as

$$\mathcal{F}^{-1}\{F(u,v)\} = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv \tag{4}$$

Our definitions for the spectrum and the phase remain unchanged except that the notation accounts for the fact that we are using two variables instead of one. So the Fourier spectrum is $|F(u,v)| = [R(u,v)^2 + I(u,v)^2]^{1/2}$, $\phi(u,v) = \tan^{-1}[I(u,v)/R(u,v)]$ is the *phase angle*, and the *power spectrum* is defined as $P(u,v) = |F(u,v)|^2$.

Discrete domain

As in the continuous case we can define the discrete FT (DFT) as

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{-kn} \quad 0 \le k \le N-1 \tag{5}$$

with $W_N = \exp\{j2\pi/N\}$.

W_N plays an important role in many further calculations. Thus we immediately state the following important result concerning sums of integer powers of this quantity. For a proof see the exercises.

$$\sum_{n=0}^{N-1} W_N^{kn} = N\delta[k \mathrm{mod} N] = \begin{cases} 0 & \text{for } k \bmod N \neq 0 \\ N & \text{for } k \bmod N = 0 \end{cases} \tag{6}$$

The inverse DFT (IDFT) is defined as [1]

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{kn} \quad 0 \le n \le N-1. \tag{7}$$

The above equation can be simply proved by the following steps:

$$\begin{aligned} \text{RHS} &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{kn} \\ &= \frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{m=0}^{N-1} X[m]W_N^{-km}\right]W_N^{kn} \\ &= \frac{1}{N}\sum_{m=0}^{N-1} X[m]\left[\sum_{k=0}^{N-1} W_N^{(n-m)k}\right] \\ &= \frac{1}{N}\sum_{m=0}^{N-1} X[m]\delta[(n-m)\mathrm{mod}N] \\ &= x[n] \end{aligned}$$

With no great effort we can define the two dimensional DFT as

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \tag{8}$$

with $u = 0,1,2,\ldots,M-1$ and $v = 0,1,2,\ldots,N-1$.

Finally, the two dimensional inverse DFT is given as

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \tag{9}$$

with $x = 0,1,2,\ldots,M-1$ and $y = 0,1,2,\ldots,N-1$.

On a square array, very little changes except the transform and its inverse are made more symmetric. We can write the pair as:

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux+vy)/N} \tag{10}$$

with $u,v = 0,1,2,\ldots,N-1$.

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux+vy)/N} \tag{11}$$

with $x,y = 0,1,2,\ldots,M-1$.

Properties

In this section we present (without proof) some properties of the FT in the continuous and discrete domain. Proofs are left to the student as an exercise.

Continuous domain

1. *Conjugate.* $\mathcal{F}\{f^*(x,y)\} = F^*(u,v)$, where the * indicates the complex conjugation operation.
2. *Symmetry.* If $f(x,y) = f(-x,-y)$ then $F(u,v) = F(-u,-v)$.
3. *Linearity.* $\mathcal{F}\{af_1(x,y) + bf_2(x,y)\} = aF_1(u,v) + bF_2(u,v)$.
4. *Shift.* $\mathcal{F}\{f(x-a,y-b)\} = F(u,v)e^{-j(ua+vb)2\pi}$
5. *Convolution.* $\mathcal{F}\{f(x,y) \odot h(x,y)\} = F(u,v)H(u,v)$ where $\odot$ is the convolution operation. Convolution can be written as $f(x,y) \odot h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\xi,\eta)h(x-\xi,y-\eta)d\xi d\eta$.

[1] Through some matrix algebra, we shall see shortly why the inverse should be so defined.

Discrete domain

1. *Linearity.* DFT $\{ax[n] + by[n]\} = aX[k] + bY[k],\ a, b, \in \mathcal{C}$.
2. *Periodicity.* $X[k] = X[k + N]$.
3. *Circular Shift.* Given $y[n] = x[(n - m) \text{mod} N]$, then $Y[k] = W_N^{-km} X[k],\ m \in \mathcal{Z}$.
4. *Frequency Shift.* $y[n] = W_N^{mn} x[n] \Leftrightarrow Y[k] = X[(k - m) mod N],\ m \in \mathcal{Z}$
5. *Parseval's Theorem* $\sum_{n=0}^{N-1} x[n]y^*[n] = \sum_{k=0}^{N-1} X[k]Y^*[k]$. As a special case of this theorem we have $\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{k=0}^{N-1} |X[k]|^2$.

Computational considerations

In section we will discuss how to compute the DFT. We look at the computational complexity of this method. Then we discuss the Fast Fourier Transform (FFT) method to compute the DFT. The computational advantages of the FFT are discussed and also a Fast IDFT is derived.

How to compute

At this point it will be useful to do an example of the FT and also speak about the computational complexity of DFT. The example is for the one dimensional FT, but due to the separability property of the FT we can compute a two dimensional DFT as two separate one dimensional transforms.

Consider a discrete signal with the following four values $[f(0)\ f(1)\ f(2)\ f(3)] = [2\ 3\ 4\ 4]$. We can now use equation (5) to get

$$F(0) = \frac{1}{4}[f(0) + f(1) + f(2) + f(3)] = 3.25.$$

Similarly we can get $F(1)$ as

$$F(1) = \frac{1}{4} \sum_{x=0}^{3} f(x) e^{-j2\pi x/4} = \frac{1}{4}(-2 + j)$$

After going through the tedium we get $[F]^T = \frac{1}{4}[13 \quad -2 + j \quad -1 \quad -2 - j]$.

We can write the evaluation of $[F]$ as a matrix multiplication of $\frac{1}{N}[w]$ (to be defined) and $[f]$, making for easier visualization of the process. $[w]$ is written as:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j2\pi/4} & e^{(-j2\pi/4)2} & e^{(-j2\pi/4)3} \\ 1 & e^{(-j2\pi/4)2} & e^{(-j2\pi/4)4} & e^{(-j2\pi/4)6} \\ 1 & e^{(-j2\pi/4)3} & e^{(-j2\pi/4)6} & e^{(-j2\pi/4)9} \end{pmatrix}$$

Now define $\omega = e^{-j2\pi/4}$. Hence the matrix $[w]$ can be written (in a considerably simplified notation) as:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{pmatrix}$$

In general we can define $\omega = e^{-j2\pi/N}$, where N is the number of values of $f()$ that we wish to transform. In this case it can be shown that the matrix $[w]$ is

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)^2} \end{pmatrix}$$

To obtain the IDFT normally one would invert the $[w]$ matrix. However it is not necessary. On going through some math we find that the following matrix equation holds:

$$[f] = [w^*][F]$$

In the above equation the * indicates the complex conjugate operation. The significance of this result is that the original functions may be recovered from the transformed version without the need for matrix inversion.

We continue with our example. The $[w]$ matrix can be written as

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix}$$

and $[w^*]$ can be written as

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{pmatrix}$$

On performing the indicated multiplication we recover the original function values.

Note that we perform N^2 multiplications to obtain the DFT of $[f]$. In the next section we will show how to reduce the number of arithmetic operations. The algorithm to do this is called the Fast Fourier Transform.

How to compute faster

The Fast Fourier Transform (FFT) has an interesting history. Apparently first discovered by Cooley and Tukey in 1965, later historical surveys traced the method back to C. F. Gauss in an 1805 work that predates Fouriers work on his famous series! Gauss has said of his method that, "experience will teach the user that this method will greatly lessen

the tedium of mechanical calculation." At any rate, having quick methods to compute the DFT were useless until the advent of the computer age and so the older methods were a solution looking for a problem. The following method of doing the FFT is due to Danielson and Lanczos [2] and goes back to the 40's.

The DFT can be written as

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)\, \omega_N^{ux}$$

where $\omega_N = e(-j2\pi/N)$. We have added the subscript N to reinforce the notion that ω is dependent on N. We make the further assumption that $N = 2^n = 2M$.

Since we assume that we are using powers of 2 for N we can break up the sum into two sums. The first sum uses those values of $f(x)$ for which x is even and the second sum takes $f(x)$ for x odd. Thus the sum is:

$$F(u) = \frac{1}{2}\left[\frac{1}{M}\sum_{x=0}^{M-1} f(2x)\, \omega_{2M}^{u\,2x} + \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)\, \omega_{2M}^{u(2x+1)}\right].$$

Noting that $\omega_{2M}^{2ux} = \omega_M^{ux}$, the above equation simplifies to

$$F(u) = \frac{1}{2}\left[\frac{1}{M}\sum_{x=0}^{M-1} f(2x)\omega_M^{ux} + \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)\omega_M^{ux}\omega_{2M}^{u}\right]. \tag{12}$$

Now let us define the odd (subscript $_o$) and even (subscript $_e$) parts as:

$$F_e(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x)\, \omega_M^{ux}, \tag{13}$$

and

$$F_o(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)\, \omega_M^{ux}. \tag{14}$$

In equations (13) and (14) $0 \le u \le M-1$. Substituting equations (13) and (14) into equation (12) we get

$$F(u) = \frac{1}{2}\left[F_e(u) + F_o(u)\, \omega_{2M}^{u}\right] \tag{15}$$

Noting that $\omega_M^{u+M} = \omega_M^u$ and that $\omega_{2M}^{u+M} = -\omega_{2M}^u$, we can write an equation to get the second part of the series as

$$F(u+M) = \frac{1}{2}\left[F_e(u) - F_o(u)\, \omega_{2M}^{u}\right] \tag{16}$$

In principle one could continue the subdivision process till one reached a single value. The DFT a point is a point itself. Then we work with pairs of values, pairs of pairs of values, pairs of pairs of pairs of values ... using equations (15) and (16) each time. The subdivision process amounts to a reordering of the input sequence of data. In the following paragraph we show this process for an array of 8 values.

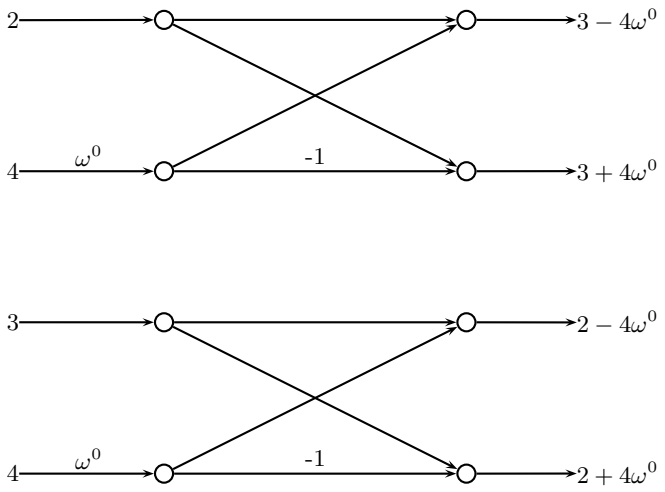

Figure 1: Signal flow graph for 2 point transforms

Example 1: As an example consider the following input sequence [2 3 4 4].

Solution: The reordered data is [2 4 3 4]. Taking the data two points at a time we get the two two point transforms as [6 −2 7 −1]. Now apply the 4 point transform to this data to get the following array [13 $-2+j$ −1 $-2-j$]. Throughout this process we have ignored the scaling factor. Thus the final transformed array is $\frac{1}{4}$[13 $-2+j$ −1 $-2-j$].

This process can be represented in the form of a signal flow graph shown in the following figures. First the two two point transforms are shown in figure 1. In each graph a ∘ represents an adder, any number associated with a line is a multiplier. Thus to get the first number in the transform the number 2 is added to the product of 4 and ω^0. The second number in the transform is obtained by adding 2 to the product of the following numbers 4 $\omega^0(-1)$. Also, the four point transform is shown in 2. With a diagram of the signal flow graph it is easy to visualize the implementation of equations (15) and (16).

Example 2: In this example consider the following array of values that we wish to transform - $[f_0\ f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7] = [2\ 3\ 4\ 5\ 6\ 7\ 8\ 9]$.

Solution: As a first step we divide it into its even and odd values $[f_0\ f_2\ f_4\ f_6]$ and $[f_1\ f_3\ f_5\ f_7]$. So the 8 points have been divided into 2 sets of 4 points. For each set of 4 points group the odd and even points. This gives us $[f_0\ f_4]$, $[f_2\ f_6]$, $[f_1\ f_5]$, and $[f_3\ f_7]$. Finally, we can write our reordered array as $[f_0\ f_4\ f_2\ f_6\ f_1\ f_5\ f_3\ f_7] = [2\ 6\ 4\ 8\ 3\ 7\ 5\ 9]$. Remembering that the first phase takes the data in pairs

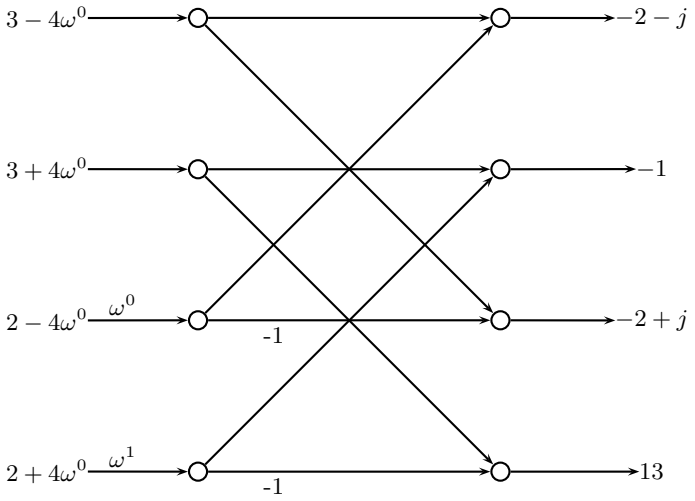

Figure 2: Signal flow graph for 4 point transforms

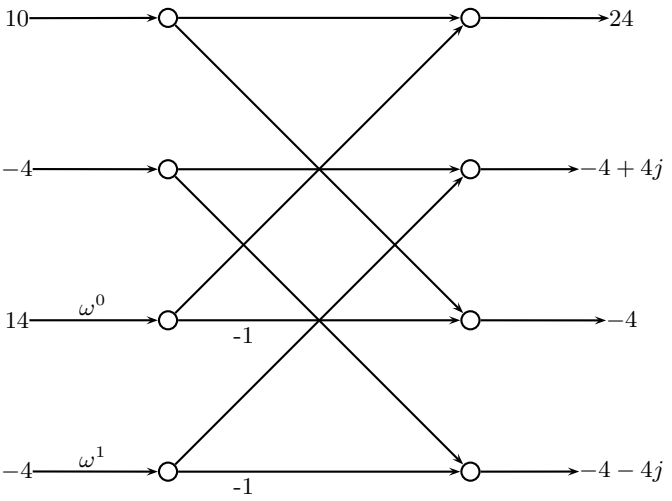

Figure 4: Problem 2: Signal flow graph for 4 point transforms, bottom half of signal.

and that this phase only involves subtractions and additions (i.e., no complex multiplications whatsoever) we get $[8\ -4\ 12\ -4\ 10\ -4\ 14\ -4]$. Next we do the two four point transforms shown in Figures 3 and 4.

Finally, an eight point transform is shown in Figure 5 which gives us the actual FFT. Also note the way in which the figure encodes equations (15) and (16).

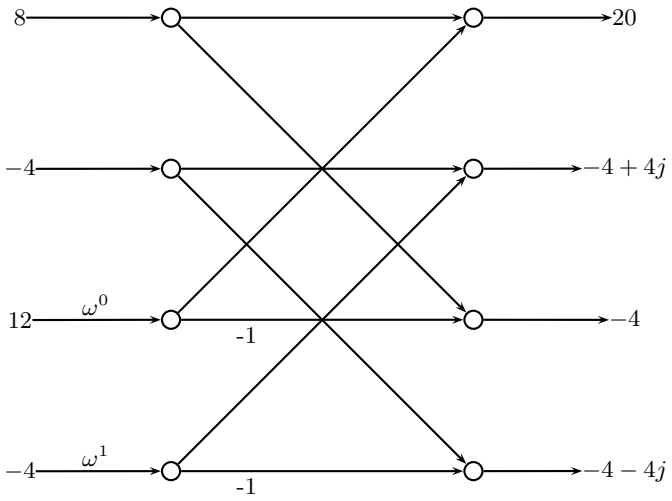

Figure 3: Problem 2: Signal flow graph for 4 point transforms, top half of signal.

The FFT can be shown to have $\mathcal{O}(N \log_2\{N\})$ multiplications. The older algorithm has N^2 multiplications. Thus we have achieved considerable savings by using the Danielson-Lanczos Lemma.

The inverse operation

We rewrite the DFT and IDFT for convenience. They are

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)\ e^{-j2\pi ux/N}$$

and

$$f(x) = \frac{1}{N} \sum_{x=0}^{N-1} F(u)\ e^{+j2\pi ux/N}.$$

Take the complex conjugate on both sides of the above equation. This can be written as

$$\frac{1}{N} f^*(x) = \frac{1}{N} \sum_{x=0}^{N-1} F^*(u)\ e^{-j2\pi ux/N}.$$

The above equation is in the form of the DFT. Hence, we can take the complex conjugate of the transformed variables and FFT it to get $f^*(x)/N$. Take the complex conjugate of $f^*(x)$ and multiply by N to get $f(x)$. The advantage of this is the fact that we can use the same code to do the FFT and the inverse FFT.

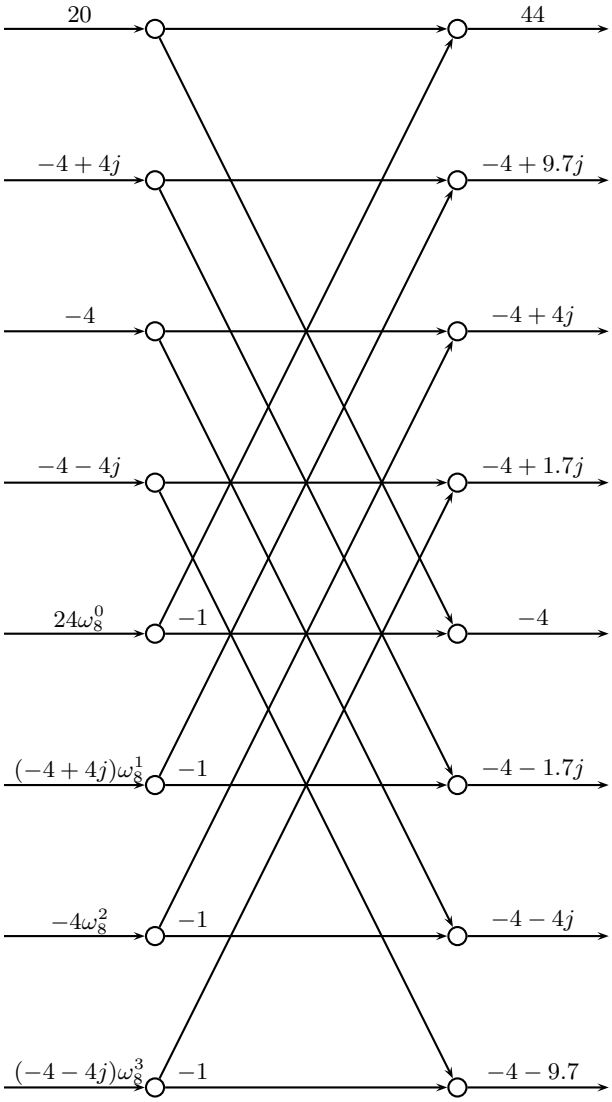

Figure 5: Problem 2: Signal flow graph for 8 point transform.

A matrix factorization view of the FFT

We have already defined the discrete fourier matrix $[w]$. Let us use the notation $\mathbf{W}_N = [w]$ to represent the $N \times N$ DFT matrix, in order to make typography simpler and start with $\mathbf{W}_8$.

$$\mathbf{W}_8 = \frac{1}{8}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_8 & \omega_8^2 & \omega_8^3 & \omega_8^4 & \omega_8^5 & \omega_8^6 & \omega_8^7 \\ 1 & \omega_8^2 & \omega_8^4 & \omega_8^6 & \omega_8^8 & \omega_8^{10} & \omega_8^{12} & \omega_8^{14} \\ 1 & \omega_8^3 & \omega_8^6 & \omega_8^9 & \omega_8^{12} & \omega_8^{15} & \omega_8^{18} & \omega_8^{21} \\ 1 & \omega_8^4 & \omega_8^8 & \omega_8^{12} & \omega_8^{16} & \omega_8^{20} & \omega_8^{24} & \omega_8^{28} \\ 1 & \omega_8^5 & \omega_8^{10} & \omega_8^{15} & \omega_8^{20} & \omega_8^{25} & \omega_8^{30} & \omega_8^{35} \\ 1 & \omega_8^6 & \omega_8^{12} & \omega_8^{18} & \omega_8^{24} & \omega_8^{30} & \omega_8^{36} & \omega_8^{42} \\ 1 & \omega_8^7 & \omega_8^{14} & \omega_8^{21} & \omega_8^{28} & \omega_8^{35} & \omega_8^{42} & \omega_8^{49} \end{pmatrix}, \tag{17}$$

where, $\omega_8 = \exp(-2\pi i/8)$ or in general $\omega_N = \exp(-2\pi i/8)$. This can be factored into $\mathbf{W}_8 = \frac{1}{8}\mathbf{CS}$ where each of the matrices are written as

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

and

$$\mathbf{C} = \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_8^2 & \omega_8^4 & \omega_8^6 & \omega_8 & \omega_8^3 & \omega_8^5 & \omega_8^7 \\ 1 & \omega_8^4 & \omega_8^8 & \omega_8^{12} & \omega_8^2 & \omega_8^6 & \omega_8^{10} & \omega_8^{14} \\ 1 & \omega_8^6 & \omega_8^{12} & \omega_8^{18} & \omega_8^3 & \omega_8^9 & \omega_8^{15} & \omega_8^{21} \\ \hline 1 & \omega_8^8 & \omega_8^{16} & \omega_8^{24} & \omega_8^4 & \omega_8^{12} & \omega_8^{20} & \omega_8^{28} \\ 1 & \omega_8^{10} & \omega_8^{20} & \omega_8^{30} & \omega_8^5 & \omega_8^{15} & \omega_8^{25} & \omega_8^{35} \\ 1 & \omega_8^{12} & \omega_8^{24} & \omega_8^{36} & \omega_8^6 & \omega_8^{18} & \omega_8^{30} & \omega_8^{42} \\ 1 & \omega_8^{14} & \omega_8^{28} & \omega_8^{42} & \omega_8^7 & \omega_8^{21} & \omega_8^{35} & \omega_8^{49} \end{array}\right). \tag{19}$$

The reader should confirm that the product $\frac{1}{8}\mathbf{CS}$ does indeed produce $\mathbf{W}_8$. This can be done by noting that $\mathbf{C}$ is the full $\mathbf{W}_8$ matrix but with even numbered columns, i.e., $0, 2, 4, 6$, first and odd numbered columns, $1, 2, 3, 7$ next and that post multiplication by $\frac{1}{8}\mathbf{S}$ leads to $\mathbf{W}_8$.

Now, $\mathbf{S} = [\delta_0\ \delta_4\ \delta_1\ \delta_5\ \delta_2\ \delta_6\ \delta_3\ \delta_7]$ with

$$\delta_k = [0\ 0\ \ldots\ \underbrace{1}_{k^{th}\text{row}}\ \ldots\ 0]^T, k \in \{0 \ldots 7\},$$

and as indicated in (19) $\mathbf{C}$ can be partitioned into four 4×4 sub–matrices

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{C}_2 \\ \hline \mathbf{C}_3 & \mathbf{C}_4 \end{array}\right)$$

Explicitly,

$$\frac{1}{4}\mathbf{C}_1 = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_8^2 & \omega_8^4 & \omega_8^6 \\ 1 & \omega_8^4 & \omega_8^8 & \omega_8^{12} \\ 1 & \omega_8^6 & \omega_8^{12} & \omega_8^{18} \end{pmatrix}.$$

Noting that $\omega_8^{2n} = \omega_4^n$, gives us

$$\mathbf{W}_4 = \frac{1}{4}\mathbf{C}_1 = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4 & \omega_4^2 & \omega_4^3 \\ 1 & \omega_4^2 & \omega_4^4 & \omega_4^6 \\ 1 & \omega_4^3 & \omega_4^6 & \omega_4^9 \end{pmatrix}.$$

Similarly

$$\begin{aligned} \mathbf{C}_2 &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ \omega_8 & \omega_8^3 & \omega_8^5 & \omega_8^7 \\ \omega_8^2 & \omega_8^6 & \omega_8^{10} & \omega_8^{14} \\ \omega_8^3 & \omega_8^9 & \omega_8^{15} & \omega_8^{21} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega_8 & 0 & 0 \\ 0 & 0 & \omega_8^2 & 0 \\ 0 & 0 & 0 & \omega_8^3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_8^2 & \omega_8^4 & \omega_8^6 \\ 1 & \omega_8^4 & \omega_8^8 & \omega_8^{12} \\ 1 & \omega_8^6 & \omega_8^{12} & \omega_8^{18} \end{pmatrix}, \end{aligned}$$

or,

$$\frac{1}{4}\mathbf{C}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega_8 & 0 & 0 \\ 0 & 0 & \omega_8^2 & 0 \\ 0 & 0 & 0 & \omega_8^3 \end{pmatrix} \mathbf{W}_4 = \mathbf{\Omega}\mathbf{W}_4.$$

Having seen the patterns for $\mathbf{C}_1, \mathbf{C}_2$ we can easily write the corresponding results for the remaining quadrants of $\mathbf{C}$ as $\frac{1}{4}\mathbf{C}_3 = \mathbf{W}_4$ and

$$\frac{1}{4}\mathbf{C}_4 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -\omega_8 & 0 & 0 \\ 0 & 0 & -\omega_8^2 & 0 \\ 0 & 0 & 0 & -\omega_8^3 \end{pmatrix} \mathbf{W}_4 = -\mathbf{\Omega}\mathbf{W}_4.$$

With $\mathbf{C}_i, i \in \{1,2,3,4\}$ being obtained as above, we have

$$\begin{aligned} \mathbf{W}_8 &= \frac{1}{2}\left(\begin{array}{c|c} \mathbf{I}\mathbf{W}_4 & \mathbf{\Omega}\mathbf{W}_4 \\ \hline \mathbf{I}\mathbf{W}_4 & -\mathbf{\Omega}\mathbf{W}_4 \end{array}\right)\mathbf{S} \\ &= \frac{1}{2}\left(\begin{array}{c|c} \mathbf{I} & \mathbf{\Omega} \\ \hline \mathbf{I} & -\mathbf{\Omega} \end{array}\right)\left(\begin{array}{c|c} \mathbf{W}_4 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{W}_4 \end{array}\right)\mathbf{S} \\ &= \frac{1}{2}\mathbf{\Phi}_8\left(\begin{array}{c|c} \mathbf{W}_4 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{W}_4 \end{array}\right)\mathbf{S}. \end{aligned}$$

The notation $\mathbf{\Phi}_8$ may seem strange, but the utility will become evident shortly.

Now consider matrix $\mathbf{M}$ and define

$$\begin{aligned} \mathbf{M}^{(1)} &\equiv \mathbf{M} \\ \mathbf{M}^{(2)} &\equiv \left(\begin{array}{c|c} \mathbf{M} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{M} \end{array}\right) \\ \mathbf{M}^{(3)} &\equiv \left(\begin{array}{c|c|c} \mathbf{M} & \mathbf{0} & \mathbf{0} \\ \hline 0 & \mathbf{M} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{M} \end{array}\right) \end{aligned}$$

Clearly, the above notation has application to our representation of $\mathbf{W}_8$ above. Based on the above definitions we can prove the following properties:

$$\begin{aligned} \left(\mathbf{M}^{(a)}\right)^{(b)} &= \mathbf{M}^{(ab)} \\ \left(\mathbf{M}_1\mathbf{M}_2\right)^{(a)} &= \mathbf{M}_1^{(a)}\mathbf{M}_2^{(a)} \\ \left(\alpha\mathbf{M}\right)^{(a)} &= \alpha\left(\mathbf{M}\right)^{(a)} \end{aligned}$$

Inspecting $\mathbf{W}_8$ and using the above properties leads to the important generalization

$$\mathbf{W}_{2^m} = \frac{1}{2}\mathbf{\Phi}_{2^m}\mathbf{W}_{2^{m-1}}^{(2)}\mathbf{S}_{2^m}, L = 1, 2, \ldots \tag{20}$$

The reader should compare equation (20) with that for $\mathbf{W}_8$ and see if it makes sense.

Now the same technique can be applied repeatedly. Let us see what this means:

$$\mathbf{W}_{2^{m-1}}^{(2)} = \frac{1}{2}\left[\mathbf{\Phi}_{2^{m-1}}\mathbf{W}_{2^{m-2}}^{(2)}\mathbf{S}_{2^{m-1}}\right]^{(2)}, \tag{21}$$

or,

$$\mathbf{W}_{2^m} = \frac{1}{2^2}\mathbf{\Phi}_{2^m}\mathbf{\Phi}_{2^{m-1}}^{(2)}\mathbf{W}_{2^{m-2}}^{(4)}\mathbf{S}_{2^{m-1}}^{(2)}\mathbf{S}_{2^m}. \tag{22}$$

Clearly this process can be repeated to get a complete factorization

$$\mathbf{W}_{2^m} = \frac{1}{2^m}\mathbf{\Phi}_{2^m}\mathbf{\Phi}_{2^{m-1}}^{(2)}\ldots\mathbf{\Phi}_2^{(2^{m-1})}\mathbf{S}_2^{(2^{m-1})}\mathbf{S}_4^{(2^{m-2})}\ldots\mathbf{S}_{2^{m-1}}^{(2)}\mathbf{S}_{2^m}. \tag{23}$$

Note that the $\mathbf{W}_x^{(y)}$ term has completely disappeared because repeated application of equation (21) leads to $\mathbf{W}_1^{(2^m)}$. Remembering that $\mathbf{W}_L^{(a)}$ is an $L \times L$ matrix replicated a times to create a block diagonal $aL \times aL$ matrix we see that

$$\mathbf{W}_1^{(2^m)} = \mathbf{I}_{2^m},$$

i.e., an identity matrix with $2^m \times 2^m$ elements, leading to the disappearance of $\mathbf{W}$ altogether in equation (23).

Let

$$\mathbf{S} = \mathbf{S}_2^{(2^{m-1})}\mathbf{S}_4^{(2^{m-2})}\ldots\mathbf{S}_{2^{m-1}}^{(2)}\mathbf{S}_{2^m}.$$

In the next section we will show that $\mathbf{S}$ is essentially the bit–reversal of the indices of the original input vector.

In order to understand equation (23), let us do a complete factorization or $\mathbf{W}_8$.

$$\mathbf{W}_8 = \frac{1}{2^3}\mathbf{\Phi}_{2^3}\mathbf{\Phi}_{2^2}^{(2)}\mathbf{\Phi}_{2^1}^{(2^2)}\mathbf{S}.$$

We will focus on the computations that occur in the $\mathbf{\Phi}$ matrices while ignoring the bit–reversal for now.

$$\mathbf{\Phi}_{2^3} = \left(\begin{array}{cccc|cccc} 1 & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & \omega_8 & . & . \\ . & . & 1 & . & . & . & \omega_8^2 & . \\ . & . & . & 1 & . & . & . & \omega_8^3 \\ \hline 1 & . & . & . & -1 & . & . & . \\ . & 1 & . & . & . & -\omega_8 & . & . \\ . & . & 1 & . & . & . & -\omega_8^2 & . \\ . & . & . & 1 & . & . & . & -\omega_8^3 \end{array}\right)$$

$$\mathbf{\Phi}_{2^2}^{(2)} = \left(\begin{array}{cc|cc|cc|cc} 1 & . & 1 & . & . & . & . & . \\ . & 1 & . & \omega_4 & . & . & . & . \\ \hline 1 & . & -1 & . & . & . & . & . \\ . & 1 & . & -\omega_4 & . & . & . & . \\ \hline . & . & . & . & 1 & . & 1 & . \\ . & . & . & . & . & 1 & . & \omega_4 \\ \hline . & . & . & . & 1 & . & -1 & . \\ . & . & . & . & . & 1 & . & -\omega_4 \end{array}\right)$$

$$\mathbf{\Phi}_{2^1}^{(2^2)} = \begin{pmatrix} 1 & 1 & . & . & . & . & . & . \\ 1 & -1 & . & . & . & . & . & . \\ . & . & 1 & 1 & . & . & . & . \\ . & . & 1 & -1 & . & . & . & . \\ . & . & . & . & 1 & 1 & . & . \\ . & . & . & . & 1 & -1 & . & . \\ . & . & . & . & . & . & 1 & 1 \\ . & . & . & . & . & . & 1 & -1 \end{pmatrix}$$

The reader is asked to confirm that the above three matrices for $\mathbf{\Phi}_{2^3}, \mathbf{\Phi}_{2^2}^{(2)}, \mathbf{\Phi}_{2^1}^{(2^2)}$ correspond to the butterfly diagrams represented in figures 2, 3, 4 and 5 respectively.

Verbally the algorithm can be described as follows:

- Perform a bit reversal of the original data vector. This is a special kind of shuffle and corresponds to multiplying the $\mathbf{S}$ matrix with the input vector. More on this in the next section.
- Take the vector output by the bit–reversal and pre–multiply by $\mathbf{\Phi}_2^{(2^{m-1})}$ to get a new vector out. Note that the first $\mathbf{\Phi}$ matrix will always consist of 1's and -1's (as in the $\mathbf{\Phi}_{2^1}^{(2^2)}$ case) causing the the output vector to consist of paired sums and differences.
- Move right to left using the appropriate $\mathbf{\Phi}$ in equation (23) and repeat the matrix vector multiplication. Terminate after multiplying with $\mathbf{\Phi}_{2^m}$. Divide by $1/2^m$.

With the matrix factorization, the computational complexity of the algorithm can be easily obtained. All the work is done in the $\mathbf{\Phi}$ multiplications. Note that each $\mathbf{\Phi}$ matrix is $2^m \times 2^m$ but is sparse. Also note that the number of non–zero elements in each $\mathbf{\Phi}$ matrix is the same. For $\mathbf{W}_{2^3}$ the number of elements in each $\mathbf{\Phi}$ matrix is 2×2^3. For general $N = 2^m$, the number of elements is 2×2^m and therefore the number of complex–arithmetic operations to multiply an input vector with one $\mathbf{\Phi}$ matrix is $\mathcal{O}(2^m) = \mathcal{O}(N)$. The total number of $\mathbf{\Phi}$ matrices for $N = 2^m$ is $\log_2(2^m) = m$. Hence the overall complexity of the algorithm is $\mathcal{O}(m \times 2^m) = \mathcal{O}(N \log N)$.

Essentially equation (23) represents the decimation–in–time radix–two FFT.

The curly shuffle

In the FFT described, the first operation is the "shuffle" in which an input array is reordered. The reordering follows a fixed pattern. As with the butterfly diagrams let us start with $\mathbf{W}_8$ and generalize from there. For ease of reference we re–write the $\mathbf{S}$ matrix

$$\mathbf{S} = \mathbf{S}_2^{(2^{m-1})} \mathbf{S}_4^{(2^{m-2})} \ldots \mathbf{S}_{2^{m-1}}^{(2)} \mathbf{S}_{2^m},$$

and for the 8×8 case we have

$$\mathbf{S} = \mathbf{S}_2^{(4)} \mathbf{S}_4^{(2)} \mathbf{S}_8,$$

with

$$\mathbf{S}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{S}_4^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{S}_2^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

For example consider an eight term input sequence given as $[f_0 f_1 f_2 f_3 f_4 f_4 f_6 f_7]$. After the first shuffle, we get two 4 term input sequences (denoted as the lower and upper sequences respectively) $[f_0 f_2 f_4 f_6]$ and $[f_1 f_3 f_5 f_7]$. The second shuffle gives us $[f_0 f_4]$, $[f_2 f_6]$, $[f_1 f_5]$ and $f_3 f_7]$. Thus the new sequence is $[f_0 f_4 f_2 f_6 f_1 f_5 f_3 f_7]$ and this is what is input to two–point computation, the four–point computation etc. The last shuffle is not done because it is the identity matrix. The mapping can be written in tabular form as:

Decimal			Binary		
0	→	0	000	→	000
1	→	4	001	→	100
2	→	2	010	→	010
3	→	6	011	→	110
4	→	1	100	→	001
5	→	5	101	→	101
6	→	3	110	→	011
7	→	7	111	→	111

The left hand side of the of the table shows the mapping in decimal numbers and the right hand side shows the equivalent

binary digits. The remarkable thing about the mapping or the shuffle is that it can be obtained by a bit reversal (or mirror image) as can be seen from the from the fight hand column of the table. Generalizing to data sequences larger than eight, yet powers of two, we expect that the complete shuffle should be represented by a bit reversal. We shall now prove this.

Theorem: Let $n = 2^l$ where $l \in \mathcal{Z}^+$. The complete shuffle is given by:

0	$\rightarrow$	r_0^l
1	$\rightarrow$	r_1^l
2	$\rightarrow$	r_2^l
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
n-1	$\rightarrow$	r_{n-1}^l

Where, r_p^s is the bit reversal of the s bit representation of the number p. For example the $s = 4$ bit representation of $p = 8$ is 1000, then $r_8^4 = 0001$.

Proof: We use the method of induction. We can see that the theorem is trivially true for the case $k = 1$. Assume it is true for $m = k$, i.e., the inductive hypothesis is that the bits are reversed for $m = k$. We now proceed to prove that it is true for $m = k + 1$.

When $m = k+1$, the total number of elements is 2^{k+1}. In the first shuffle the even indexed elements are sent to the lower half of the interval, i.e., 0, 1, 2, 3, ..., $2^k - 1$, while the odd indexed elements to the upper half, i.e., 2^k, $2^k + 1$, $2^k + 2$, ..., $2^{k+1} - 1$.

Consider the lower half. The transfer (shuffle) is achieved by a division by 2. If the bit representation of the indices is $b_{k+1}b_k \dots b_1b_0$, then for even indices, division by two is a right circular shift, i.e., the result is given by $b_0b_{k+1}b_k \dots b_1$.

For odd indices, the shift is to the upper half. In decimal form this is written as:

Decimal		
1	$\rightarrow$	$2^k + 0$
3	$\rightarrow$	$2^k + 1$
5	$\rightarrow$	$2^k + 2$
7	$\rightarrow$	$2^k + 3$
9	$\rightarrow$	$2^k + 4$
$\vdots$	$\vdots$	$\vdots$
$2^{k+} - 1$	$\rightarrow$	$2^{k+1} - 1$

While in binary form it is:

Binary		
000...0001	$\rightarrow$	100...0 + 000...0000
000...0011	$\rightarrow$	100...0 + 000...0001 = 100...001
000...0101	$\rightarrow$	100...0 + 000...0010 = 100...010
000...0111	$\rightarrow$	100...0 + 000...0011 = 100...011
000...1001	$\rightarrow$	100...0 + 000...0100 = 100...100
$\vdots$	$\vdots$	$\vdots$
111...1111	$\rightarrow$	111...1111

What these two tables are really saying is that the odd number $1, 3, 5, \dots, p, \dots, 2^k - 1$ is represented as $p = 2q + 1$ for $q = 0, 1, 2, \dots$. In tabular form this is written as:

Decimal		
$2 \times 0 + 1 = 1$	$\rightarrow$	$2^k + 0$
$2 \times 1 + 1 = 3$	$\rightarrow$	$2^k + 1$
$2 \times 2 + 1 = 5$	$\rightarrow$	$2^k + 2$
$2 \times 3 + 1 = 7$	$\rightarrow$	$2^k + 3$
$2 \times 4 + 1 = 9$	$\rightarrow$	$2^k + 4$
$\vdots$	$\vdots$	$\vdots$
$2 \times q + 1 = p$	$\rightarrow$	2^k + q
$\vdots$	$\vdots$	$\vdots$
$2^{k+} - 1$	$\rightarrow$	$2^k + 2^k - 1$

Thus, to get the number q you subtract 1 from p and divide by 2. Now the number p always ends in 1 because it is odd, i.e., $p = b_{k+1}b_k \dots b_1 1$. So, subtracting unity from it gives us an even number $b_{k+1}b_k \dots b_1 0$ and dividing by 2 gives us a right circular shift $0b_{k+1}b_k \dots b_1$. Now, $2^k = 100 \dots 000$, i.e., $k + 1$ binary digits with 1 in the left most place and all others being 0). Thus adding the two numbers in the above table gives us $1b_{k+1}b_k \dots b_1$ which is the equivalent of a right circular shift. Thus the shuffle is $b_0b_{k+1}b_k \dots b_1$ for even or odd integers.

Now we continue by performing the shuffle on the lower and upper halves that were obtained by the initial shuffle. In this case the bit representation is $b_kb_{k-1} \dots b_1b_0$ and the full shuffle on this reverses the bits by the inductive hypothesis. Hence, the number m gets shifted to $b_0b_1b_2 \dots b_kb_{k+1}$. ■

Two dimensional transforms

In this subsection we show how we can decompose the two dimensional DFT into the product of two one dimensional DFT's. Also, we assume that x indexes $f(x, y)$ along a column while y indexes $f(x, y)$ along a row. Again, for convenience we write the 2D DFT and its inverse below.

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux+vy)/N}.$$

with $u, v = 0, 1, 2, \dots, N - 1$.

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux+vy)/N}. \qquad (24)$$

with $x, y = 0, 1, 2, \ldots, M-1$.

Rewrite the DFT as

$$F(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \left[\sum_{x=y}^{N-1} f(x,y) e^{-j2\pi vy/N} \right] e^{-j2\pi ux/N}$$

or

$$F(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} F(x,v) e^{-j2\pi ux/N}$$

with

$$F(x,v) = \sum_{x=y}^{N-1} f(x,y) e^{-j2\pi vy/N}$$

Hence the two dimensional DFT is obtained by taking the one dimensional transform of the rows of $f(x,y)$ and multiplying by N to get $F(x,v)$. Next we take the FFT along each row of $F(x,v)$ to get $F(u,v)$. This property of the DFT is called *separability* and ensures that we can trivially write a 2-D FFT routine, provided we have a one dimensional FFT routine available.

A word from the man himself

In [2] there is the following quote by Fourier: "These same theorems which enable us to solve the heat equation have immediate applications to questions in general analysis and dynamics whose solutions have long been sought." Amen to that.

Posterity has not forgotten Joseph Fourier - Egyptologist, mathematician, terroriste and public servant.

References

1. D. F. Elliott and D. R. Rao. *Fast Transforms – Algorithms, Analysis, Applications.* Academic Press, Orlando, 1982.
2. T. W.. Korner. *Fourier Analysis.* Cambridge University Press, Cambridge, 1988.