

The Fast Fourier Transform

A. Reda

Electrical Engineering and Computer Science
University of Michigan

August 13, 2007

Fast Fourier Transform

- It is not a transform by itself but rather a family of algorithms used to calculate the the Discrete Fourier Transform of a given signal.
- It expresses an input function in terms of a sum of sinusoidal components by determining the amplitude and phase of each component. However, the DFT is distinguished by the fact that its input function is discrete and finite, which makes the DFT ideal for processing information stored in computers.
- It is one of the most significant algorithms in all of computer science and is a basis for tremendous number of applications, ranging from MP3 encoding to various file formats to store photos.

Applications

Since it was introduced in the 1960's, it had been applied in numerous fields, which include

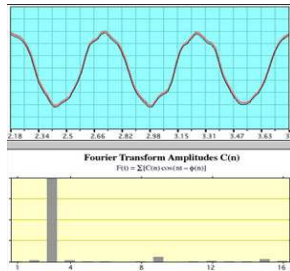
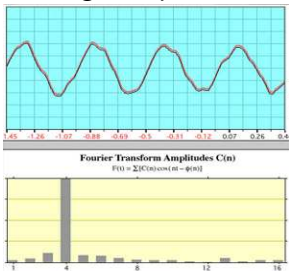
- Music and audio signal processing
- MRI
- Image processing
- Pattern recognition
- Computational chemistry
- Spectral methods for PDEs

And many other instances that could benefit from the conversion of a time-domain input to a frequency-domain output, which makes analysis easier.

Example: The London Police Whistle

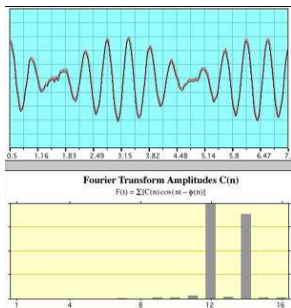


Within the compact whistle are two short pipes which produce two separate high frequencies when blown.



Cont: The London Police Whistle

- When they are blown together, we get:



- Combining two sound waves produces a complex pattern in the time domain, but the FFT clearly shows it as consisting almost entirely of two frequencies.
- In a similar manner, changing time-domain to frequency domain makes signal processing easier to handle.

DFT Definition

- A complex DFT for an N point signal f can be given as follows (where F is the transform at each point)

$$F(n) = \sum_{k=0}^{n-1} f(k) e^{\frac{-i2\pi nk}{N}}$$

- Given a signal with N points, Each of the N results of transformation needs N multiplications, resulting in N^2 operations
- Directly calculating the DFT requires about N^2 operations while an FFT algorithm can compute the same result in only $O(N \log N)$ operations.

Base-2 FFT

- If we let $W = e^{\frac{-i2\pi}{N}}$, then we will have:

$$F(n) = \sum_{k=0}^{n-1} f(k)W^{nk}$$

- It assumes that N is a power of 2, that is it can be expressed as $N = 2^\gamma$
- If we pick a simple case like $N = 4$, it would make $\gamma = 2$. If we were going to determine $F(n)$ for $n = 0, 1, 2, 3$ we will have the following set of equations:

$$F(0) = f(0)W^0 + f(1)W^0 + f(2)W^0 + f(3)W^0$$

$$F(1) = f(0)W^0 + f(1)W^1 + f(2)W^2 + f(3)W^3$$

$$F(2) = f(0)W^0 + f(1)W^2 + f(2)W^4 + f(3)W^6$$

$$F(3) = f(0)W^0 + f(1)W^3 + f(2)W^6 + f(3)W^9$$

Cont.

- In a matrix form, it will be:

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

- This will take in the neighborhood of 16 multiplications and 12 additions to figure out
- To simplify this matrix, lets establish some facts:

$$W^N = W^0 = 1$$

And for integers m and k ,

$$W^k = W^{mN+k}$$

- Using the above information, the matrix can be rewritten as:

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & W^0 & W^0 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

- Now, The role the FFT here will be to factor the $N \times N$ matrix into two $N \times N$ matrices that are easier to handle. If we were going to jump to the result, it would look like:

$$\begin{bmatrix} F(0) \\ F(2) \\ F(1) \\ F(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

- Matrix multiplication shows that the result is similar to the original, except two rows are interchanged, hence we have compensated by switching $F(1)$ and $F(2)$

- To see what we have accomplished, define:

$$\begin{bmatrix} f_1(0) \\ f_1(2) \\ f_1(1) \\ f_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

- So

$$f_1(0) = f(0) + W^0 f(2)$$

$$f_1(1) = f(1) + W^0 f(3)$$

And $W^2 f(2) = -W^0 f(2)$ (from Euler's formula)

$$f_1(2) = f(0) + W^2 f(2) = f(0) - W^0 f(2)$$

$$f_1(3) = f(1) + W^2 f(3) = f(1) - W^0 f(3)$$

Using a total of two multiplications.

- Multiplying the remaining matrix

$$\begin{bmatrix} F(0) \\ F(2) \\ F(1) \\ F(3) \end{bmatrix} = \begin{bmatrix} f_2(0) \\ f_2(2) \\ f_2(1) \\ f_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} f_1(0) \\ f_1(1) \\ f_1(2) \\ f_1(3) \end{bmatrix}$$

Result is

$$f_2(0) = f_1(0) + W^0 f_1(1)$$

$$f_2(2) = f_1(0) - W^0 f_1(1)$$

And both $f_2(1)$ and $f_2(3)$ can be done with one multiplication and two additions

- The entire process of finding $F(0), F(1), F(2), F(3)$ costs four multiplications.
- In the general case where $N = 2^\gamma$, the FFT is a method of factoring the appropriate $N \times N$ matrix to $\gamma \cdot N \times N$ matrices with enough entries equal to zero or one to reduce the multiplication required to find out $F(0), F(1), \dots, F(N - 1)$.

- So there are two issues to address here: finding the **interchanging rows**, and also the **factor matrices**
- Associate each of the integers $n=0,1,2,3$ with the ordered pair (n_1, n_0) where $n = 2n_1 + n_0$, and n_1 is either 0 or 1, and n_2 is either 0 or 1. So we will have the association

$$0 \rightarrow (0,0)$$

$$2 \rightarrow (1,0)$$

$$1 \rightarrow (0,1)$$

$$3 \rightarrow (1,1)$$

- To determine which rows are interchanged in the $N = 2^2$ case, write the bits in the reverse order. This works for γ cases too

$$0 (0,0) \rightarrow (0,0) 0$$

$$1 (0,1) \rightarrow (1,0) 2$$

$$2 (1,0) \rightarrow (0,1) 1$$

$$3 (1,1) \rightarrow (1,1) 3$$

This accurately reflects the interchange.

- Now, let's see the matrix factorization. Define

$$\hat{F}(n_1, n_0) = F(2n_1 + n_0) = F(n)$$

$$\hat{f}(k_1, k_0) = f(2k_1 + k_0) = f(k)$$

- This can be processed as

$$\begin{aligned} F(n) &= F(2n_1 + n_0) = \hat{F}(n_1, n_0) \\ &= \sum_{k=0}^3 f(k) W^{(2n_1+n_0)k} \\ &= \sum_{k_0=0}^1 \sum_{k_1=0}^1 f(2k_1 + k_0) W^{(2n_1+n_0)(2k_1+k_0)} \\ &= \sum_{k_0=0}^1 \sum_{k_1=0}^1 \hat{f}(k_1, k_0) W^{(2n_1+n_0)(2k_1+k_0)} \end{aligned}$$

- Lets consider W , which can be manipulated as follows

$$\begin{aligned}
 W^{(2n_1+n_0)(2k_1+k_0)} &= W^{(2n_1+n_0)2k_1} W^{(2n_1+n_0)k_0} \\
 &= W^{4n_1k_1} W^{2n_0k_1} W^{(2n_1+n_0)k_0} \\
 &= W^{2n_0k_1} W^{(2n_1+n_0)k_0}
 \end{aligned}$$

- Because $N = 4$, and by the property $W^0 = W^{4m+0} = 1$ for any integer m , and in this case $m = n_1k_1$. Therefore:

$$\hat{F}(n_1, n_0) = \sum_{k_0=0}^1 \left[\sum_{k_1=0}^1 \hat{f}(k_1, k_0) W^{2n_0k_1} \right] W^{(2n_1+n_0)k_0}$$

- We will briefly consider the inner sum and the outer sum to get the desired factorization

The inner sum

- If we define:

$$\hat{f}_1(n_0, k_0) = \sum_{k_1=0}^1 \hat{f}(k_1, k_0) W^{2n_0 k_1}$$

For $n_0 = 0, 1; n_1 = 0, 1$, we will have:

$$\hat{f}_1(0, 0) = \hat{f}(0, 0)W^0 + \hat{f}(1, 0)W^0,$$

$$\hat{f}_1(0, 1) = \hat{f}(0, 1)W^0 + \hat{f}(1, 1)W^0,$$

$$\hat{f}_1(1, 0) = \hat{f}(0, 0)W^0 + \hat{f}(1, 0)W^2,$$

$$\hat{f}_1(1, 1) = \hat{f}(0, 1)W^0 + \hat{f}(1, 1)W^2$$

- In matrix form, we have

$$\begin{bmatrix} \hat{f}_1(0, 0) \\ \hat{f}_1(0, 1) \\ \hat{f}_1(1, 0) \\ \hat{f}_1(1, 1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

The outer sum

- If we define:

$$\hat{f}_2(n_0, n_1) = \sum_{k_0=0}^1 \hat{f}_1(n_0, k_0) W^{(2n_1+n_0)k_0}$$

For $n_0 = 0, 1; n_1 = 0, 1$, we have

$$\begin{bmatrix} \hat{f}_1(0, 0) \\ \hat{f}_1(0, 1) \\ \hat{f}_1(1, 0) \\ \hat{f}_1(1, 1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} \hat{f}_1(0, 0) \\ \hat{f}_1(0, 1) \\ \hat{f}_1(1, 0) \\ \hat{f}_1(1, 1) \end{bmatrix}$$

- Therefore, the inner sum and outer sum define the factors of the $N \times N$ matrix

Summary

- So, to summarize what we have written so far:

$$\hat{f}_1(n_0, k_0) = \sum_{k_1=0}^1 \hat{f}(k_1, k_0) W^{2n_0 k_1}$$

and

$$\hat{f}_2(n_0, n_1) = \sum_{k_0=0}^1 \hat{f}_1(n_0, k_0) W^{(2n_1+n_0)k_0}$$

and

$$F(n) = F(2n_1 + n_0) = \hat{F}(n_1, n_0) = \hat{f}_2(n_0, n_1)$$

Where we have compensated for the interchanging of rows in the matrix by reversing the coordinates in $\hat{F}(n_1, n_0)$ and $\hat{f}_2(n_0, n_1)$

Extension

- This process can be extended to other values of N , where $N = 2^\gamma$. For Example, in $N = 2^3$, we will write

$$n = 4n_2 + 2n_1 + n_0 \quad (n_0, n_1, n_2 = 0, 1)$$

and

$$k = kn_2 + kn_1 + k_0 \quad (k_0, k_1, k_2 = 0, 1)$$

- And the above process is repeated by defining

$$F(n) = \hat{F}(n_2, n_1, n_0)$$

... etc, leading to

$$F(n) = \hat{F}(n_2, n_1, n_0) = \hat{f}_2(n_0, n_1, n_2)$$

Generalization

We can generalize the process to develop the base-2 FFT for any value of γ where $N = 2^\gamma$. Denote

$$n = 2^{\gamma-1}n_{\gamma-1} + 2^{\gamma-2}n_{\gamma-2} + \cdots + 2n_1 + n_0$$

$$k = 2^{\gamma-1}k_{\gamma-1} + 2^{\gamma-2}k_{\gamma-2} + \cdots + 2k_1 + k_0$$

We get:

$$\hat{f}_1(n_0, k_{\gamma-2}, k_{\gamma-3}, \cdots, k_0) = \sum_{k_{\gamma-1}=0}^1 \hat{f}(k_{\gamma-1}, k_{\gamma-2}, \cdots, k_0) W^{n_0 2^{\gamma-1} k_{\gamma-1}};$$

$$\hat{f}_2(n_0, n_1, k_{\gamma-3}, \cdots, k_0) = \sum_{k_{\gamma-2}=0}^1 \hat{f}_1(n_0, k_{\gamma-2}, \cdots, k_0) W^{(2n_1+n_0) 2^{\gamma-2} k_{\gamma-2}};$$

Cont. :Generalization

⋮

$$\hat{f}_\gamma(n_0, n_1, n_2, \dots, n_{\gamma-1}) = \sum_{k_0=0}^1 f_{\gamma-1}^\wedge(n_0, n_1, \dots, n_{\gamma-2}, k_0) \\ \times W^{(2^{\gamma-1}n_{\gamma-1} + \dots + 2n_1 + n + 0)k_0},$$





and

$$F(n) = F(2^{\gamma-1}n_{\gamma-1} + \dots + 2n_1 + n + 0) = \hat{F}(n_{\gamma-1}, \cdot, n_1, n_0)$$

$$F(n) = \hat{f}_\gamma(n_0, n_1, n_2, \dots, n_{\gamma-1})$$

Conclusion

- This algorithm can be used in a program that implements the Fast Fourier Transform, and reduces the efficiency of the operation from $N \times N$ to $N \times \gamma/2$, which is a huge improvement, especially in the real life applications where N is a large number.
- The Fast Fourier Transform has made performing Discrete Fourier Transforms feasible, thereby enabling us to use DFT in a plethora of applications all across the digital world.
- Slides located at www.umich.edu/~azarias

-  Peter V Oniel, Advanced Engineering Mathematics, University of Alabama, Wadsworth Publishing, 1991,
-  Daniel Rockmore, FFT-an algorithm the whole family can use, Dartmouth College, 1999
-  E. O. Brigham, The fast fourier transform and its applications, Prentice Hall, 1988
-  <http://hyperphysics.phy-astr.gsu.edu/hbase/sound/london.html>