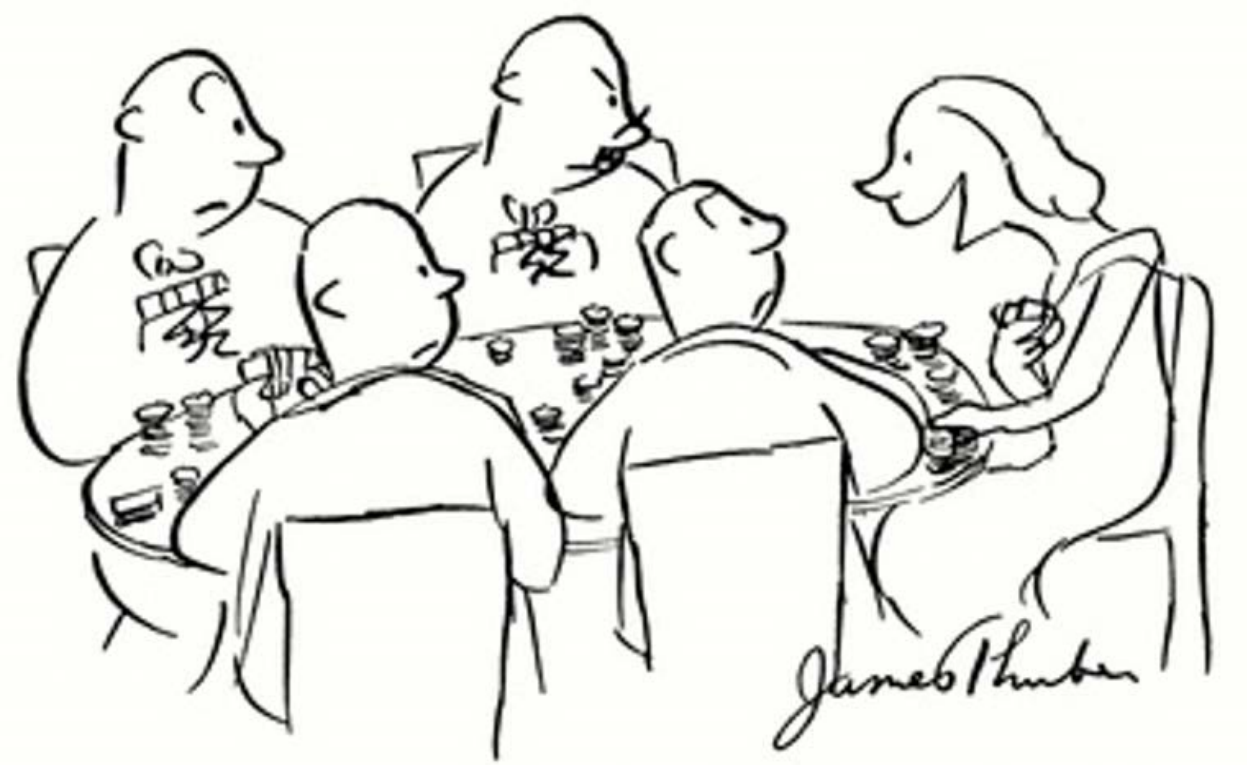


"What do four ones beat?"

EE 311 **January 25, 2019**
Notes – Fifth Day

Chapter 3
• Discrete Time Fourier Transform

The Fourier transform and its inverse are:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$ Fourier Transform

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega$$ Inverse Fourier Transform

The Fourier transform works on a signal continuous in time and produces a signal continuous in frequency.

If we allow the time variable to become discrete we get the discrete-time Fourier transform or DTF.

An easy way to think about the DTF is to perform numerical integration on the Fourier transform integral using rectangles. Each rectangle is of height $f(nT)e^{-jwnT}$ and is of width T. Applying numerical integration to the Fourier transform we get

$$F(\omega) = T\sum_{n=-\infty}^{\infty} f(nT)e^{-j\omega nT}$$

The T out front is either normalized to 1 or just dropped since we are rarely interested in the absolute value of the frequency coefficients but in their value relative to each other.

$$DTF(f(nT)) = \sum_{n=-\infty}^{\infty} f(nT)e^{-j\omega nT}$$ discrete-Time Fourier transform

$$f(nT) = DTF^{-1}\{F(\omega T)\} = \frac{1}{2\pi}\int_{2\pi} F(\omega T)e^{j\omega nT}d\omega$$ inverse DTF

The DTF has properties similar to the Fourier transform and is important since the z-transform is a generalization of the DTF.

• Discrete Fourier Transform

The DFT is defined by taking the equation for the Fourier transform and allowing both the time and frequency variable to become discrete. Since both the time and frequency variable are sampled we will choose to take N samples. In the time domain we choose the sampling period to be T so that the continuous time variable t will be replaced by the discrete time variable nT, where n is the integer counter in the time domain. In the frequency domain the signal will necessarily be periodic with period $\omega_p = 2\pi f_p = 2\pi/T$. Since we are taking N samples in frequency space, $\Delta\omega = \omega_p/N = 2\pi/(NT)$. The discrete frequency variable is $k\Delta\omega = 2\pi k/(NT)$, where k is the integer frequency counter. We replace the integral with a sum over a finite period and $dt \to T$. Making these substitutions in the equation for the Fourier transform produces the following:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt \quad \to \quad F(2\pi k / N) = \sum_{n=0}^{N-1} f(nT) e^{-j\frac{2\pi k}{NT}nT} \cdot T$$

(Note that this is equivalent to replacing the integral with a formula for numerical integration using rectangles of height $f(nT) \cdot e^{-j\frac{2\pi kn}{N}}$ and of width T.)

DFT

$$X[k] = \mathbf{DFT}\left[x[n]\right] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, \quad k = 0,1,2,\ldots,N-1$$

IDFT

$$x[n] = \mathbf{DF}^{-1}\left[X[k]\right] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}, \quad n = 0,1,2,\ldots,N-1$$

The DFT is used mostly because all variables are discrete so it is easily done on a computer.

Notice that in calculating the DFT we have to do N complex multiplications for each value of k. This implies that a 256 point DFT would take $256^2 = 65{,}536$ complex multiplications.

We consider the case that we have three data points. Hence, for $N = 3$, the discrete Fourier transform, (3.42), is given by

$$\begin{aligned} X[0] &= x[0] + x[1] + x[2]; \\ X[1] &= x[0] + x[1]e^{-j2\pi/3} + x[2]e^{-j4\pi/3}; \\ X[2] &= x[0] + x[1]e^{-j4\pi/3} + x[2]e^{-j2\pi/3}. \end{aligned}$$

In the last term in the third equation, $e^{\frac{-j8\pi}{3}} = e^{\frac{-j6\pi}{3}} e^{\frac{-j2\pi}{3}} = e^{\frac{-j2\pi}{3}}$. The inverse discrete Fourier transform, (3.43), is then

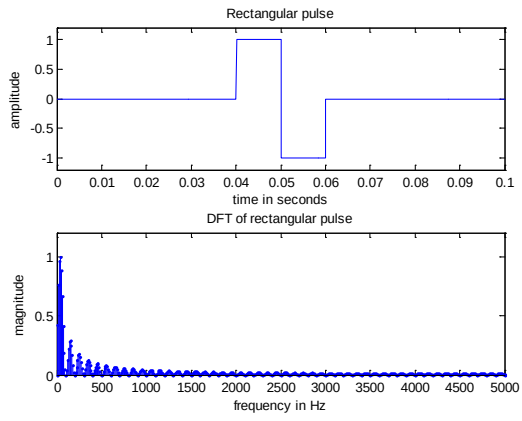
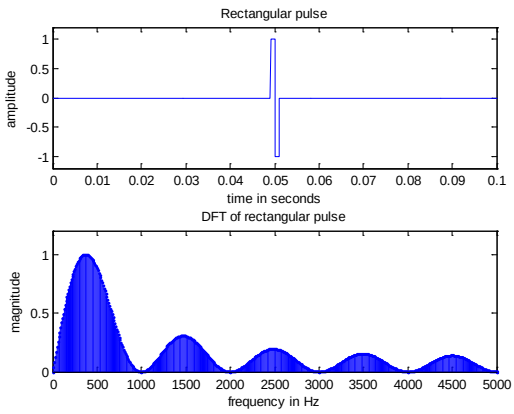
$$\begin{aligned} x[0] &= \frac{1}{3}\left(X[0] + X[1] + X[2]\right); \\ x[1] &= \frac{1}{3}\left(X[0] + X[1]e^{j2\pi/3} + X[2]e^{j4\pi/3}\right); \\ x[2] &= \frac{1}{3}\left(X[0] + X[1]e^{j4\pi/3} + X[2]e^{j2\pi/3}\right). \end{aligned}$$

Note that the calculation of the DFT requires only a sign change and multiplication by a constant to get the IDFT.

MATLAB example

```
%DFTDemo2.m
%Plots a rectangular pulse and its DFT.
fs = 10000;
T = 1/fs;
N = 1000;
n = 1:N;
nT = n*T;
%
m = fix(.01/T);
s = [zeros(1, N/2-m) ones(1, m) -ones(1, m) zeros(1, N/2-m)];

figure(1);clf;
subplot(2, 1, 1);
plot(nT, s);
axis([0 N*T -1.2 1.2]);
xlabel('time in seconds');
ylabel('amplitude');
title('Rectangular pulse');
%
subplot(2, 1, 2);
sDFT = fft(s);
f = 0:fs/N:fs-fs/N;
sDFTNorm = abs(sDFT)/max(abs(sDFT));
stem(f, sDFTNorm, 'MarkerSize', 1);
%axis([0 fs/2 0 1.2]);
axis([0 fs/2 0 1.2]);
xlabel('frequency in Hz');
ylabel('magnitude');
title('DFT of rectangular pulse');
```

	
pulse width = 0.02	pulse width = 0.002

• Fast Fourier transform (FFT)

To show how the FFT can be done we take N to be a power of 2 and break the DFT up into two sequences of even and odd terms.

$$F(k)=\sum_{\substack{n=0\\ n\ even}}^{N-1} f(n)e^{-j\frac{2\pi kn}{N}}+\sum_{\substack{n=0\\ n\ odd}}^{N-1} f(n)e^{-j\frac{2\pi kn}{N}}$$

This equation can be simplified by doing a variable change of n to 2m.

$$F(k)=\sum_{m=0}^{\frac{N}{2}-1} f(2m)e^{-j\frac{2\pi k}{N}2m}+\sum_{m=0}^{\frac{N}{2}-1} f(2m+1)e^{-j\frac{2\pi k}{N}(2m+1)}$$

From this equation we can factor $e^{-j\frac{2\pi k}{N}}$ from the right most term to get

$$F(k)=\sum_{m=0}^{\frac{N}{2}-1} f(2m)e^{-j\frac{2\pi k}{N}2m}+e^{-j\frac{2\pi k}{N}}\sum_{m=0}^{\frac{N}{2}-1} f(2m+1)e^{-j\frac{2\pi k}{N}2m} \qquad \textbf{2.18}$$

To simplify the notation let $G(k)=\sum_{m=0}^{\frac{N}{2}-1} f(2m)e^{-j\frac{2\pi k}{N}2m}$ and $H(k)=\sum_{m=0}^{\frac{N}{2}-1} f(2m+1)e^{-j\frac{2\pi k}{N}2m}$ so that equation 2.18 can be written as

$F(k)=G(k)+e^{-j\frac{2\pi k}{N}}H(k)$ where G(k) and H(k) are both N/2 term DFTs.

An N-term DFT is periodic with a period of N. Equation 2.18 is composed of two N/2-term DFT's and the period of each of these must be N/2. This implies that G(k) = G(k+N/2) and H(k) = H(k+N/2). To take advantage of this we write

$$F(k+N/2)=\sum_{m=0}^{\frac{N}{2}-1} f(2m)e^{-j\frac{2\pi(k+N/2)}{N}2m}+e^{-j\frac{2\pi(k+N/2)}{N}}\sum_{m=0}^{\frac{N}{2}-1} f(2m+1)e^{-j\frac{2\pi(k+N/2)}{N}2m}$$

or, $F(k+N/2)=\sum_{m=0}^{\frac{N}{2}-1} f(2m)e^{-j\frac{2\pi k}{N}2m}\cdot e^{-j2m\pi}+e^{-j\frac{2\pi k}{N}}\cdot e^{-j\pi}\sum_{m=0}^{\frac{N}{2}-1} f(2m+1)e^{-j\frac{2\pi k}{N}2m}\cdot e^{-j2m\pi}$

But $e^{-j2m\pi}=\cos(2m\pi)-j\sin(2m\pi)=1$ since m is an integer and $e^{-j\pi}=-1$. This leads to

$$F(k+N/2)=\sum_{m=0}^{\frac{N}{2}-1} f(2m)e^{-j\frac{2\pi k}{N}2m}-e^{-j\frac{2\pi k}{N}}\sum_{m=0}^{\frac{N}{2}-1} f(2m+1)e^{-j\frac{2\pi k}{N}2m} \qquad \textbf{2.19}$$

or $F(k+N/2)=G(k)-e^{-j\frac{2\pi k}{N}}H(k)$

Thus we see that if we calculate G(k) and H(k) to find F(k) in equation 3.16, we can make a single sign change and find F(k+N/2) using equation 2.19. To find all of the values of F(k) then, it is necessary to calculate all of the values of G(k) and H(k). This amounts to some considerable savings in computation time since G(k) and H(k) are N/2-term DFTs. For example, if N = 16, a straight forward calculation of F(k) using the equation for the DFT would have $N^2 = 256$ complex operations. Calculation of F(k) from G(k) and H(k) would have only $2(N/2)^2 + N/2 = 136$ complex operations. (The N/2 term which is added in this equation is the result of the multiplication of $e^{-j\frac{2\pi k}{N}}$ time H(k)).

We can get further computational savings if we can repeat this process on G(k) and H(k). Thus, if N is a power of 2, N/2 is also a power of 2 and G(k) and H(k) can be divided into even and odd terms just as F(k) was. This division can continue until we are down to transforms of 1-term functions. Thus a 16 term function would be divided into two 8 point functions. These would be divided into four 4 point functions which would lead to eight 2 point functions. Each division would produce fewer complex operations. For an N-point sequence where $N = 2^p$, we can repeat this reduction process p times. Since $p = \log_2(N)$, the total number of complex multiplications is reduced to $(N/2)\log_2(N)$. ($N\log_2(N)$ additions are also required.) For example, if N = 1024, $\log_2(1024) = 10$ and 5,120 complex multiplications are required as opposed to 1,048,576 by brute force.

To visualize this consider the signal flow graph of Figure 3.9. In this figure, the central darkened circle is a summing junction and the signal flow graph shows how F(k) and F(k+N/2) are calculated from G(k) and H(k). Note that the ± sign on the e term must be chosen positive when calculating F(k) and negative when calculating F(k+N/2).

If an 8 point DFT is broken down into two 4 point DFTs, the butterfly signal flow graphs would be calculated as shown in Figure 3.10. In this figure the even terms of f(n) form the first 4 terms of G(k) and the odd terms of f(n) form the first four terms of H(k). If G(k) and H(k) are further broken down into even and odd terms the signal flow graph for F(k) is shown in Figure 2.11.

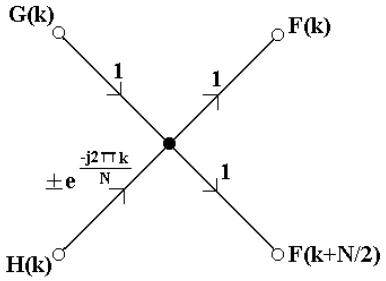


Figure 2.9
Signal flow graph for the calculation of F(k) and F(k+N/2) from G(k) and H(k) according to equations 1 and 2. This signal flow graph is commonly referred to a as a "Butterfly".

Note that in Figure 2.11 the ordering of the original signal, f(n) is skewed. In the first reduction the function was divided into even and odd parts so that the ordering is 0, 2, 4, 6, 1, 3, 5, 7. These 4-point DFTs are further divided into even and odd parts the new ordering becomes 0, 4, 2, 6, 1, 5, 3, 7.

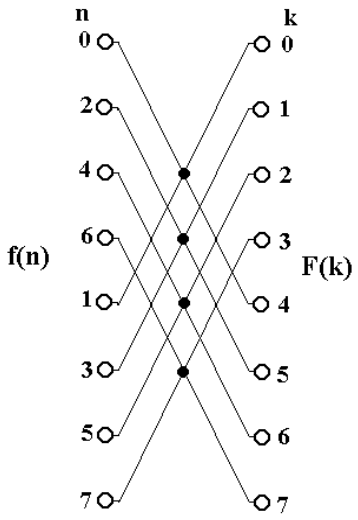


Figure 2.10
The butterfly signal flow graphs for the calculation of an 8-point FFT from two 4-point DFTs.

As it turns out this ordering of the input sequence can be easily determined by writing the sequence in binary notation and reversing the bit orders. Thus we see that

$0 = 000 \rightarrow$	$000 = 0$
$1 = 001 \rightarrow$	$100 = 4$
$2 = 010 \rightarrow$	$010 = 2$
$3 = 011 \rightarrow$	$110 = 6$
$4 = 100 \rightarrow$	$001 = 1$
$5 = 101 \rightarrow$	$101 = 5$
$6 = 110 \rightarrow$	$011 = 3$
$7 = 111 \rightarrow$	$111 = 7$

This process of reordering the input sequence is called “bit twiddling”.

Example
Use MATLAB to calculate the FFT of a wav file and display the data in both time and frequency.

```
%WavFileFFT.m
[x fs] = audioread('rossini.wav');
T = 1/fs;
k = 1:length(x);
figure(1);clf;
subplot(2,1,1)
plot(k*T,x)                %Plot x in time
axis([0 T*length(x) -1.5 1.5])
xlabel('time in seconds');
ylabel('voltage');
title('rossini.wav in Time');
%
X = fft(x);
X = X/max(abs(X));
subplot(2,1,2)
plot(k*fs/length(x), abs(X))  %Plot X in frequency
axis([0 fs/2 0 .5]);
xlabel('frequency in Hz');
ylabel('gain');
title('rossini.wav in frequency');
wavplay(x);
```