ROSE-HULMAN INSTITUTE OF TECHNOLOGY
Department of Electrical and Computer Engineering

EC 300 Signals and Systems — Fall 2000
B.A. Black

The Fourier Transform

by Bruce A. Black

Objectives

- To introduce the fast Fourier transform (FFT) algorithm for efficient computer calculation of Fourier transforms.
- To investigate some of its properties

Background

The Fast Fourier Transform

Suppose `g` is an array of N values representing the time signal $g(k\Delta t)$. The MATLAB command

```
 >> G = fft(g);
```

causes MATLAB to compute the Fourier transform of the time signal $g(k\Delta t)$ and place the results in an array `G`. The array `G` represents a spectrum $G(n\Delta f)$, also of N values. Remember that MATLAB numbers its array elements starting with one. This means that $g(0)$ is stored in array element `g(1)` and $g((N\text{-}1)\Delta t)$ is stored in `g(N)`. Similarly, $G(0)$ is stored in array element `G(1)` and $G((N\text{-}1)\Delta f)$ is stored in `G(N)`. For greatest computational efficiency, N should be a power of two.

If $g(k\Delta t)$, $k = 0, ..., N\text{-}1$ are samples of a time signal, the Fourier transform that MATLAB calculates is given by

$$G(n\Delta f) = \sum_{k=0}^{N-1} g(k\Delta t) e^{-j2\pi nk/N}, \; n = 0, \ldots, N-1. \tag{1.1}$$

Note that Δt represents the time between values of $g(k\Delta t)$. It turns out that the frequency interval Δf between values of $G(n\Delta f)$ is given by

$$\Delta f = \frac{1}{N\Delta t}.$$

Given the array `G`, the MATLAB command

```
    >> g = ifft(G);
```

calculates the *inverse* Fourier transform given by

$$g(k\Delta t) = \frac{1}{N}\sum_{n=0}^{N-1} G(n\Delta f) e^{+j2\pi nk/N}, \; k = 0, \ldots, N-1. \tag{1.2}$$

The Fourier transform given by Eq. (1.1) is called a *discrete* Fourier transform. MATLAB uses the discrete transform because a computer cannot store continuous-time signals. MATLAB uses an efficient algorithm called the Fast Fourier Transform (FFT) to calculate the discrete Fourier transform.

The discrete Fourier transform has properties that are almost identical to those of the familiar continuous Fourier transform. There is one important difference. The spectrum $G(n\Delta f)$ defined in Eq. (1.1) above is periodic in frequency with period $n\Delta f$. This periodicity is a consequence of the discrete-time nature of the time signal $g(k\Delta t)$. One period of the spectrum extends from frequency 0 to frequency $(N-1)\Delta f$. The positive frequency components lie between frequency 0 and frequency $\left(\frac{N}{2}-1\right)\Delta f$. The spectral components from frequency $\left(\frac{N}{2}\right)\Delta f$ to $(N-1)\Delta f$ are repeats of the *negative frequency* components that lie between frequencies $-\left(\frac{N}{2}\right)\Delta f$ and $-\Delta f$ respectively.

Because the spectrum $G(n\Delta f)$ is defined only at discrete values of frequency, the FFT algorithm considers the time function $g(k\Delta t)$ to be periodic with period $N\Delta t$. Consequently, the N-value array `g` you define will be interpreted as one period of an infinite-duration periodic signal. The spectrum $G(n\Delta f)$ defined in Eq. (1.1) is actually the Fourier transform of the periodic signal.

Plotting the Spectrum

If $G(n\Delta f)$ is plotted against frequency, zero hertz will appear on the left of the graph. The positive frequency components will appear to the right of zero, followed farther to the right by the negative frequency components. Because this is contrary to convention, MATLAB provides a function to rearrange the components of the array `G` to place the negative frequency components to the left of zero. The command

```
 >> H = fftshift(G);
```

will create an array `H` that represents a spectrum $H(n\Delta f)$ whose DC component is in the center as expected.

Before plotting `H` (or `G`), remember that these arrays contain complex numbers. The command `plot(H)` will cause MATLAB to plot the imaginary part against the real part! This usually gives an interesting graph, but probably not the one you had in mind. You may obtain the magnitude spectrum by using the command `M = abs(H)`, and the angle spectrum by `a = angle(H)`. Jumps of 2π in the angle spectrum can be removed by `p = unwrap(a)`. Tip: If `H` is real (i.e. the imaginary part is zero or only contains numerical roundoff "noise"), plot it. If `H` is complex-valued, plot `abs(H)` and `angle(H)`.

Pre-Lab

Calculate by hand the conventional Fourier transform of the continuous-time version of each of the waveforms described in Procedure Steps 1 through 7 below. Record the calculations, the results, and plots of the Fourier transforms in your lab notebook.

Procedure

Because of the requirement that the number of samples be a power of two, we will let all of the time signals in this lab project consist of $N = 512$ samples having a total duration of 500 μs. (What does this make Δt? What is Δf?) You can generate a time axis and a frequency axis for your graphs by

```
 >> t = linspace(0,(N-1)*deltat,N);
 >> f = linspace(-(N/2)*deltaf,((N/2)-1)*deltaf,N);
```

Don't forget to do

```
 >> H = fftshift(G);
```

so the spectrum matches the frequency axis you have created.

1. A discrete-time "unit impulse" is defined by the time signal

$$\delta(k\Delta t) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise.} \end{cases}$$

   Let the time signal be a unit impulse. Compute and plot the spectrum. Verify that the computation is correct by evaluating Eq. (1.1) by hand.

   MATLAB hint: If you type
   ```
    >> set(gcf, 'PaperPosition', [0.5,0.5,7.5,10]);
   ```
   before printing, your plots will be better spaced on the page.

2. Let the time signal be $g(k\Delta t) \equiv 1$. Compute and plot the spectrum. Verify that your spectrum is correct by substituting the spectrum you obtain into Eq. (1.2) and showing by hand that you obtain $g(k\Delta t)$ back again.

3. Let the time signal be a single pulse extending from $t = -16\Delta t$ to $t = 16\Delta t$. (Remember, the time signal is interpreted as periodic!) Compute and plot the spectrum. Verify the plausibility of your result by comparing it with the conventional Fourier transform of the continuous-time pulse that you computed in the prelab.

4. Let the time signal be the pulse of Step 3 above, but extending from $t = -32\Delta t$ to $t = 32\Delta t$. Compare its spectrum with the spectrum you obtained in Step 3.

5. Let the time signal be a cosine of amplitude one whose frequency is chosen so that it has exactly 32 cycles in 500 µs. Compute and plot the spectrum. Now verify your result by substituting the spectrum you obtain into Eq. (1.2) and showing that you recover the original cosine. (This is much easier that substituting a cosine into Eq. (1.1) to verify the spectrum.)

6. Let the time signal be a cosine of amplitude one whose frequency is 65 kHz. Compute and plot the spectrum. Compare the result with the spectrum you obtained in Step 5 above.

7. Let the time signal be an "RF pulse" of frequency 64 kHz and duration $64\Delta t$. Compare your result with the conventional Fourier transform of the continuous-time RF pulse that you computed in the prelab.

Report

Make sure that your lab notebook contains the required spectra from Steps 1 through 7. Verify the correctness of your results as requested in Steps 1, 2, and 5. Make the comparisons requested in Steps 3, 4, 6, and 7. Have all members of your lab group sign the lab notebook, and hand the notebook in at the end of lab.