EFC LaBerge
November 23, 2016
FFT_help.docx

This document provides some hints about using the MATLAB function fft and associated functions.

As noted in class, the Fast Fourier Transform (FFT) is just a very efficient algorithm for computing the Discrete Fourier Transform (DFT). The DFT is defined by

$$X[k]=\sum_{n=0}^{N-1}x[n]e^{-j\frac{2\pi kn}{N}} \tag{0.1}$$

where $n=0,1,2,\ldots N-1$ is the time domain index, and is assumed to correspond to the times $t_n=n\Delta t$; N is the number of points in the time domain sample, which is also the number of points in the frequency domain result; $x[n]$ is the value of the n-th time domain sample; $k=0,1,2,\ldots N-1$ is the frequency domain index, and is assumed to correspond to the frequencies $f_k=k\Delta f=\frac{k}{N\Delta t}$, and $X[k]$ is the value of the k-th frequency domain term or sample. Note that (0.1) is just a set of numbers and mathematical operations: the *interpretation* associated with $n\Delta t$ and $k\Delta f$ is dependent on the application, but is *not* explicitly contained in the expression.

The FFT/DFT implicitly assumes that the N point time domain and frequency domain records are periodic. The time domain period is $T=N\Delta t$, while the frequency domain period is $S=N\Delta f=N\frac{1}{N\Delta t}=\frac{1}{\Delta t}$ is the sample frequency.

The FFT/DFT may be used to approximate the analog/analytical Fourier Transform expression

$$X(f)=\int_{-\infty}^{\infty}x(t)e^{-j2\pi ft}\,dt \tag{0.2}$$

under the assumption that $T=N\Delta t$ is very long relative to meaningful changes in $x(t)$ by

$$\begin{aligned}X(k\Delta f)&\approx\left(\sum_{n=0}^{N-1}x(n\Delta t)e^{-j2\pi k\Delta f n\Delta t}\right)\Delta t\\&=\left(\sum_{n=0}^{N-1}x(n\Delta t)e^{-j\frac{2\pi kn\Delta t}{N\Delta t}}\right)\Delta t\\&=X[k]\Delta t\end{aligned} \tag{0.3}$$

When using MATLAB's `fft` function, it is important to note the indices given in (0.1). The index of the first time sample is *assumed* to be at $t=0$. If the physical system is such that the first time sample instead takes place at $t=-M\Delta t$, where M is a positive integer, then the output will be $X_M[k]=e^{-j\frac{2\pi M\Delta tk}{N\Delta t}}X[k]=e^{-j\frac{2\pi Mk}{N}}X[k]$ by the time shift property of Fourier Transforms.

Similarly, the frequency domain results $X[k]$ are assumed to correspond to frequencies $f_k = 0, \Delta f, 2\Delta f, 3\Delta f, \ldots, (N-1)\Delta f$. Because $X[k]$ is periodic in frequency with period $N\Delta f$, this sequence of frequencies is equivalent to

$$\begin{aligned} f_k &= 0, \Delta f, 2\Delta f, \ldots, \left(\frac{N}{2}-1\right)\Delta f, \left(\frac{N}{2}\right)\Delta f, \left(\frac{N}{2}+1\right)\Delta f \ldots, (N-1)\Delta f \\ &= 0, \Delta f, 2\Delta f, \ldots, \left(\frac{N}{2}-1\right)\Delta f, \left(-\frac{N}{2}\right)\Delta f, \left(-\frac{N}{2}+1\right)\Delta f \ldots, 1\Delta f \end{aligned}, \qquad (0.4)$$

that is, the *negative* frequency elements are in the *second* half of the array.

Recognizing that users frequently want to view the `fft` results in the *principle* range of frequencies, $-\frac{S}{2} \le f < \frac{S}{2}$, MATLAB provides a useful function, `fftshift`, that rearranges the `fft` results to correspond to

$$f_k = \left(-\frac{N}{2}\right)\Delta f, \left(-\frac{N}{2}+1\right)\Delta f \ldots, 1\Delta f, 0, \Delta f, 2\Delta f, \ldots, \left(\frac{N}{2}-1\right)\Delta f \qquad (0.5)$$

MATLAB Example

```
% FFT_Example.m
%  EFCL  11/23/2016

pulse = @(t,T) (t>=0)-(t>=T);

fsa=1000;
dt=1/fsa;
t=[-4096:4095]*dt; % an array with 2^M points, but starting at M=-4096
NFFT = length(t);
trecord = NFFT*dt; % total measurement time
fresolution = 1/trecord; % the resolution is 1/(measurement time)
df = fresolution; %...which is also the frequency resolution

tau = 1;
p1 = pulse(t+tau/2,tau); % a centered pulse

funshifted=[0:NFFT-1]*df; % FFTs go from 0 to (N-1)df
f = [-NFFT/2: NFFT/2-1]*df;  % centered version

P1 = fft(p1)*dt; % multiply by dt to scale to be an integral

figure(1);
subplot(3,1,1);
theory_p1 = tau*sinc(f*tau);
plot(funshifted,abs(P1),f,abs(theory_p1),'r:','LineWidth',2);
ylabel('|P_1(f)|');
xlim([0 fsa]);
legend('|FFT of p_1(t)|','Theory');
grid on;

subplot(3,1,2);
P1 = fftshift(P1); % use fftshift to reorder
plot(f,abs(P1),f,abs(theory_p1),'r:','LineWidth',2); % use the shifted version
ylabel('|P_1(f)|');
xlim([-10 10]);
```

```
legend('|FFT of p_1(t)|','Theory');
grid on;

subplot(3,1,3);
% remove erroneous phase shift due to first sample not at t=0
P1adj = exp(j*2*pi*min(t)*f).*P1; % adjust for first sample not at t=0

plot(f,angle(P1adj)/pi,'b',f,angle(theory_p1)/pi,'r:','LineWidth',2);
xlim([-10, 10]);
ylabel('Angle(P1adj)/\pi');
xlabel('Frequency in Hz');
```