CS1031 - Lab 3
Fourier Transform and Spectrum Analysis

February 7, 2016

1 Introduction to the Fast Fourier Transform (FFT)

The Discrete Fourier Transform (DFT) is extremely important in the area of spectrum analysis. It takes a discrete signal in the time domain and transforms that signal into its discrete frequency domain representation; this allows the processing of its frequency components through computer-based algorithms, which wouldn't be able to work with a continuous analog function.[1]

The Fast Fourier Transform (FFT) is just a clever implementation of the DFT, specifically designed to be as fast as possible in terms of processing time; its use has become so widespread that the acronyms FFT (a specific implementation) and DFT (the generic analysis tool) have become inter-exchangeable.

1.1 Basic properties of the FFT

The time-continuous version of the DFT is periodic with period f_s, where f_s is the sampling frequency of the discrete-time signal; it thus makes sense to only define the DFT in the region between 0 and f_s.

Furthermore, it's easy to observe that the DFT is symmetric around the centre point $0.5f_s$, which is called the Nyquist frequency (see Fig. 1). For this reason, very often only half of the DFT is shown (e.g. the portion between $f = 0$ and $f = 0.5f_s$).

For a better understanding of these concepts, please refer to the lecture slides "More on Spectrum".

1.2 FFT in Matlab

The typical syntax for computing the FFT of a signal is FFT(x,N) where x is the signal, x[n], you wish to transform, and N is the number of points

[1] Note that this is different from the Discrete-Time Fourier Transform (DTFT), which takes an discrete signal in the time domain and maps it into a continuous representation in the frequency domain.

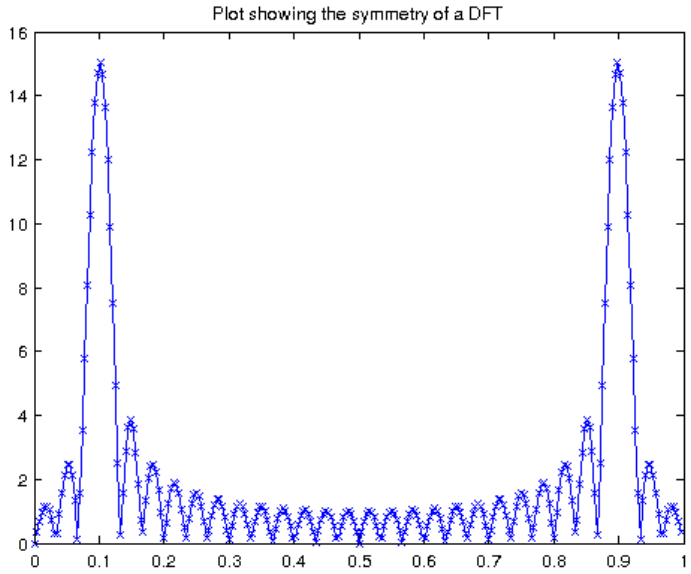

Figure 1: The DFT of a cosine with frequency equal to one tenth of the sampling frequency.

in the FFT. N must be at least as large as the number of samples in x[n]. Remember that from the slides in the lecture

2 Exercises

2.1 Exercise 1: simple FFT plot

To demonstrate the effect of changing the value of N, produce a sine function with frequency 10Hz, 20 periods long. Remember to choose an appropriate sampling frequency f_s (e.g., >=2*$fmax$).

f_s=... put here your sampling frequency

time=... put here the number of seconds required to cover 20 periods of your function

frequency=... put here the frequency of your sine function

x=[0:1/fs:time-1/fs]; this is the x axis of your function

y=... write here your sine function with correct parameters

Define 3 different values for N, e.g. 64, 128 and 256. Then make the FFT for each of the 3 values of N. Use the abs function to find the magnitude of the transform, as we are not concerned with distinguishing between real

and imaginary components. Then use fftshift, and normalise the x axis.

N=... this should be done for 64, 128 and 256;

F=fftshift(....); fill in correctly

newX=-f_s/2:f_s/N:f_s/2-f_s/N; this is the x axis for your spectrum plot

plot(newX,F);

Plot the three different FFT in separate plots.

2.2 Exercise 2: Frequency Components With Noise

A common use of Fourier transforms is to find the frequency components of a signal buried in a noisy time domain signal. The array of values given in array.mat contains a signal made up of a number of sine waves with added random noise. Load the signal load('array.mat'), then plot it (plot(y)) and check whether you can identify the frequency of the sine functions making up the signal. Most probably you cannot due to the noise making it impossible to identify by eye any sinusoidal behaviour.

Now do the FFT of the above signal, plot it and tell which frequencies is the signal made of. (Hints: remember that the result of the FFT needs to be shifted and scaled in order to properly identify the correct frequencies: refer to the code above. The sampling frequency f_s used was 1000 Hz. Use a value of N=1024 for the FFT).

2.3 Exercise 3: Spectrum of Music

In this exercise we will try to obtain the results similar to those shown at the end of the slides "More on Spectrum". From the file 'exercise notes.wav', containing samples from two notes, you will need to: separate the two notes into separate arrays, and find their tune, by examining their spectrum individually. For the FFT use a value of 16,384 for N. Once you identify the fundamental frequency, you can use the document 'frequency of musical notes.pdf' to identify the pitch.

Useful commands:

[$notes, fsampling$]=audioread('filename'): reads the .wav file, putting the sample values in an array called notes, and also returns the sampling frequency used in the variable fsampling.

When you try to separate the two notes, you can play them using the command: sound(array,fsampling), where array is the array containing the sample values you want to play, and fsampling should be the same value obtained by the audioread() function above.