EEE3008 Lab 2: Discrete Fourier Transform and Spectral Analysis

Lab Session:	Tue 29 Oct 2024, 12:00-14:00 (check personal timetable)
Submission:	Before 16:00 Tue 5 Nov 2024
Feedback:	Within 2 weeks of submission deadline

Introduction

This lab examines the computer implementation of the Fourier Transform – the Discrete Fourier Transform (DFT). Matlab uses "fft" to calculate the DFT. You will use this to compute the magnitude spectrum. The experiment continues by examining some of the finer points of spectral analysis, particularly the use of Window functions to improve the ability to detect low level signals in the presence of high level signals.

Experiments – Part A: DFT of Sinewaves & Zero-padding

In this set of experiments, you will look at the DFT (in particular the magnitude) of some simple signals and examine the effect of *zero padding*. In each case the signal will be 100 samples long. You will examine what happens using a DFT with lengths of 100, 200 and 1000 (plus other lengths greater than 1000 if you wish).

1. DFT of a Sinewave

Create a .m file and name it e.g. **zeropadding.m**. This will be the file where you will enter the skeleton script to be used for the exercises to follow. Start by setting up time and frequency domain indices/axes:

```
%set up the time and frequency domain indices/axes
M = 100;          %100 samples
n = [0:M-1];      %array of values to assist
N = 100;          %length of DFT
t = [0:N-1];      %time axis
w = [0:2*pi/N:pi];  %frequency axis up to half sampling frequency
```

Next, set up the signal:

```
%set up the signal
T = 10;                  %sine period
x = sin(2*pi*n/T);       %the first 100 samples
x = [x, zeros(1,N-M)];   %zero pad if necessary
```

Note: The current signal is not zero-padded yet, since N = M.

Add a "plot" or a "stem" function (see the Self-Learning worksheet), in order to view the generated signal, **x**, and execute the script at the command line.

Now append the following lines to look at the signal's Fourier Transform up to the half the sampling frequency:

```
f_domain = abs(fft(x, N));                    % magnitude of dft
dB_f_domain = 20*log10(f_domain);             % dB magnitude
figure(2)                                     %open a new plot window
subplot(2, 1, 1); stem(w, f_domain(1:N/2+1));  % plot magnitude in upper graph
subplot(2, 1, 2); plot(w, dB_f_domain(1:N/2+1))  % plot dB in lower graph
```

Note that the y-axis of dB_f_domain is marked in dB.

Add lines to your code for labels and titles to the plots.

Explain what you see in the spectrum you plotted. How does that compare to what you would expect from theory?

[3 marks]

2. Effect on DFT of different sinewave periods

Keeping M = N = 100 (so still no zero padding yet), repeat the above plots with T = {15, 30, 50}.

Describe how the frequency representation changes as T changes. Why do you think some of these look "sharper" than the others? Suggest a condition relating T and N when the frequency representation is "sharper". **[3 marks]**

3. Zero padding

Now we will look at the effect of padding a signal with zeros. We will put them after the time signal, but padding can occur before the signal or both before and after. The line below takes care of the zero padding (this should already be in your script):

```
x = [x,zeros(1,N-M)];          %zero pad if necessary
```

Keeping M = 100, set N = 200 and T = 10. Examine the signal in the frequency domain.

How does the appearance of the sine wave spectrum change from the case of no padding (N=100), to zero-padding with 100 zeros (N=200, so N-M = 100)? **[2 marks]**

Now try with N = 1000 at T = 10. Comment on the changes you see. **[2 marks]**

Experiments – Part B: Windows and Spectral Analysis

4. Window functions

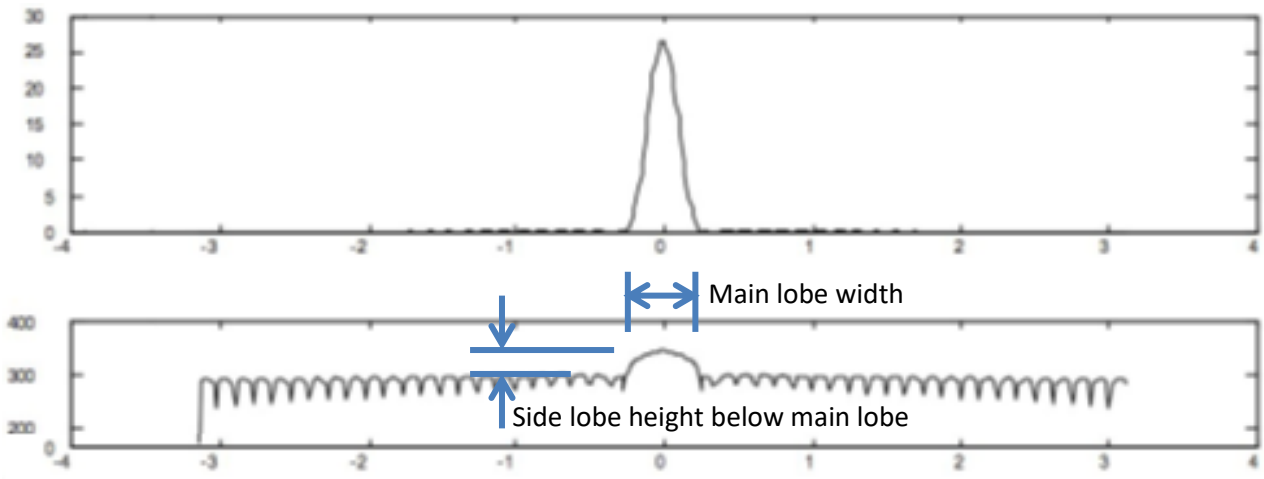
The following script allows you to look at the M-length windows in the frequency and the time domain. Enter the script in a .m file (without any function heading). You can run this script by typing the name of the file in the Matlab console.

```
M = 50;                          %window length
N = 1500;                        %fft length
n = [0:1:N-1];                   %array to assist
n = mod(n-N/2,N);                %circular shift n
w = [-pi:2*pi/N:pi-2*pi/N];      %frequency axis

win = hamming(M);                %window in time domain (e.g. Hamming window)
Fwin = fft(win,N);               %window in frequency domain
mag = abs(Fwin);                 %magnitude of window in freq. domain
mag = [mag zeros(1,N-length(mag))]; %perform circular shift (see also the fftshift command)
magc = mag(n+1);                 %to centre the spectral peak
dB = 20*log10((mag+eps)/max(eps)); %same as above but in dB
dB = [dB zeros(1,N-length(dB))];
dBc = dB(n+1);

%plot the results
subplot(3,1,1); plot(win);       %plot window in time domain
subplot(3,1,2); plot(w,magc);    %plot window in frequency domain
subplot(3,1,3); plot(w,dBc);     %plot window in frequency domain in dB
```

Note: mag and dB are circularly shifted so that the spectral peak (main lobe) occurs in the centre of the plot. We use an fft length N longer than window to bring out the fine detail in the sidelobes.



In Matlab, to measure main lobe width and side lobe height, zoom in the plot and analyse the co-ordinates.

a) Properties of different window functions

Use the script you entered above to examine various windows in the time and frequency domains. Keeping the window length M = 50 and fft length N = 1500, look in particular at the following windows, making good use of the plotting options to get the best view of your data:

```
win = boxcar(M)       % rectangular window
win = triang(M)       % Bartlett window [optional]
win = hanning(M)      % Hann window [optional]
win = hamming(M)      % Hamming window
win = blackman(M)     % Blackman window
```

Consider how different windows look different in the time and frequency domains.
(Hint: concentrate on frequency domain, in dBs, and think about main lobe widths and side lobe levels). Mark on your plots the main lobe and side-lobes.

Describe how the main lobe width and side lobe height change as you change between the following windows: a rectangular (boxcar) window; a Hamming window; a Blackman window.

[3 marks]

b) Effect of changing window length

What happens to the main lobe width and the side lobe height, if you change the length of the Hamming window to M = 25 or M = 100? Suggest a rule relating these values.

[2 marks]

5. Windowed sinewave

Next you will examine a windowed sinewave in the frequency domain. Set up a sinewave **x** as in Part A of this lab using T=10 but with N=1000. Now window the signal **x** by adding the following lines to your script:

```
M = 100;                      % Set window width
win = hamming(M);             % Create Hamming window (column vector)
win = [win', zeros(1, N-M)];  % Zero-pad the window to a row vector of length N
xwin = x.*win;                % Apply window to signal x using element-wise multiplication
```

Look at the windowed sinewave **xwin** in the time and frequency domains in exactly the way you did for **x** in Part 1. Repeat this using the rectangular (boxcar) and Blackman windows.

Plot the time domain and magnitude spectra in dB of the windowed sinewave **xwin**, when windowed using a rectangular, Hamming and Blackman windows.

On your plots, estimate the heights (in dB) of the main lobe and side lobes for the different windows.

How do these relate to the properties of the different windows you saw in Part 4?

[4 marks]

6. Windowing a sum of sine waves and noise

Next we look at two sine waves, and add noise to our signal, to explore how different windows allow us to see more or less detail in the frequency domain.

Modify your script to use a signal with two sine waves of different amplitude and noise, with longer fft length, for example as follows:

```
N=1000; t=[0:N-1]; w=[0:2*pi/N:pi];
M = 200; n=[0:M-1];
T1=10; T2=8;                    % Periods of sin waves
noise=randn(1,M);               % Noise signal
A1=1.0; A2=0.1;                 % Amplitude of sin waves
A3=0;                           % Amplitude of noise (here = 0)
x=A1*sin(2*pi*n/T1) + A2*sin(2*pi*n/T2) + A3*noise;
x = [x, zeros(1, N-M)];         % Zero-pad signal to length N
win=boxcar(M);                  % Build M-point window, and zero pad to N
win = [win', zeros(1, N-M)];
xwin = x.*win;                  % Apply window to signal x
f=20*log10(abs(fft(xwin)));     % Take magnitude of DFT
% Plot time and freq domain
subplot(2,1,1);plot(t(1:M),xwin(1:M));
subplot(2,1,2);plot(w,f(1:N/2+1));
```

Plot and describe what happens using different windows, including (a) rectangular (boxcar), (b) Hamming and (c) Blackman.

[2 marks]

Investigate different amplitudes for the second sinusoid by changing its amplitude A2. For different window types, see how small you can set A2 but still resolve the second frequency peak. Describe how the use of different window types affect the ability to resolve the second spectral peak, as you change A2.

[2 marks]

Now change A3 to a non-zero value, to add some noise to the signal. Observe and plot what happens as you change the noise amplitude A3 to different levels (e.g. A3=0.1, 0.01, 0.001).

The level of “background noise” is sometimes called the “noise floor”. Using the Blackman window, estimate the noise level (“noise floor”) for a few values of A3. Suggest an equation relating the level of the noise floor (in dB) to the noise amplitude A3.

[2 marks]

[Total 25 marks]

Optional Sections (No marks)

7. Try different levels of the sinusoids and noise and different frequencies of the sinusoids. Consider the “resolution” of the spectral estimate, using different windows as the relative amplitudes and frequencies of two sine waves change. Consider the relative advantages and disadvantages of different windows.

8. So far we've been looking at single frames of the spectrum analysis. If we take this spectral analyisis of every window and put them together we can create a spectogram of how the frequencies change over time. For the next piece of code you will need a .wav soundclip saved in your Matlab directory.

An example can be found at: http://www.ee.columbia.edu/~dpwe/sounds/sents/sm1_cln.wav.

```
» [y,fs] = wavread('example.wav');
» [B,f,t] = specgram(y,1024,fs,256,192);
» bmin = max(max(abs(B)))/300;
» imagesc(t,f,20*log10(max(abs(B),bmin)/bmin));
» axis xy;
» xlabel('Time (s)'); ylabel('Frequency (Hz)');
» sound(y,fs);
```