Music 320
Autumn 2010–2011
Homework #5
DFT, Spectrogram, Convolution
140 points
Due in one week (10/28/2010)

Theory Problems

1. (10 pts) [Fourier Theorems] Design a *complex frequency shifter* which accepts a complex input signal and shifts its spectrum up by 100 Hz (adds 100 Hz to all frequency components, positive and negative). Denote the sampling interval by T. (Hint: Consider the dual of the shift theorem.) For what input signals is the output signal real?
   Solution: By the Shift theorem of the DFT,

   $$Y(\omega) = X(\omega - \Delta) \leftrightarrow e^{j\Delta nT} x(n)$$

   Therefore,

   $$y(n) = e^{j2\pi 100nT} x(n)$$

   shifts up the input spectra by 100 Hz.

   Now, let's assume a signal $x(n)$ as:

   $$x(n) = e^{-j2\pi 100nT} r(n)$$

   where $r(n)$ is a real signal.
   Let's use this signal $x(n)$ as the input of our system, i.e.:

   $$y(n) = e^{j2\pi 100nT} e^{-j2\pi 100nT} r(n) = r(n)$$

   So $y(n)$ is real with and input $e^{-j2\pi 100nT} r(n)$ with $r(n)$ a real signal.

   Another approach: Since the spectrum of real signals must be Hermitian, the spectrum of input must be Hermitian centered around -100[Hz].

2. (30 pts) [DFT] Define $\omega_k T \overset{\Delta}{=} 2\pi k/N$. Find the length $N = 8$ DFTs

   $$X_i(k), \quad k = 0, 1, \ldots, 7,$$

   for the following sequences (without using `Matlab`):

(a) $x_1 = [1, 0, 0, 0, 0, 0, 0, 0]$

$$X_1(k) = [1, 1, 1, 1, 1, 1, 1, 1]$$

(b) $x_2 = [0, 1, 0, 0, 0, 0, 0, 0]$

$$X_2(k) = [1, \frac{1-j}{\sqrt{2}}, -j, \frac{-1-j}{\sqrt{2}}, -1, \frac{-1+j}{\sqrt{2}}, j, \frac{1+j}{\sqrt{2}}]$$

(c) $x_3 = [1, 1, 1, 1, 1, 1, 1, 1]$

$$X_3(k) = [8, 0, 0, 0, 0, 0, 0, 0]$$

(d) $x_4 = [2, 1, 0, 0, 0, 0, 0, 1]$

$$X_4(k) = [4, 2+\sqrt{2}, 2, 2-\sqrt{2}, 0, 2-\sqrt{2}, 2, 2+\sqrt{2}]$$

(e) Express $X_4(k)$ in terms of $X_1(k)$ and $X_2(k)$.

$$\begin{aligned} X_1(k) &= 1 \\ X_2(k) &= e^{-j\cdot k\pi/4} \\ X_4(k) &= 2 + e^{-j\cdot k\pi/4} + e^{-j\cdot 7k\pi/4} \end{aligned}$$

Since $e^{-j\cdot 7k\pi/4} = e^{j\cdot k\pi/4} \quad (\forall k \in \mathbf{Z})$,

$$\begin{aligned} X_4(k) &= 2 + e^{-j\cdot k\pi/4} + e^{j\cdot k\pi/4} \\ &= 2 + e^{-j\cdot k\pi/4} + \overline{e^{-j\cdot k\pi/4}} \\ &= 2X_1(k) + X_2(k) + \overline{X_2(k)} \\ &= 2X_1(k) + 2\mathbf{Re}\{X_2(k)\} \end{aligned}$$

Or,

$$\begin{aligned} X_4(\hat{\omega}_k) &= 2 + e^{-j\cdot k\pi/4} + e^{j\cdot k\pi/4} \\ &= 2 + e^{-j\cdot k\pi/4} + e^{j\cdot k\pi/4} \cdot 1 \\ &= 2X_1(\hat{\omega}_k) + X_2(\hat{\omega}_k) + e^{j\cdot k\pi/4} X_1(\hat{\omega}_k) \\ &= (2 + e^{j\cdot k\pi/4}) X_1(\hat{\omega}_k) + X_2(\hat{\omega}_k) \end{aligned}$$

Alternatively, We can express $x_4(n)$ as linear combination of $x_1(n)$ and $x_2(n)$ with some delay. That is,

$$x_4(n) = 2x_1(n) + x_2(n) + x_1(n+1)$$

Therefore, using the linearity and the shift theorem of the DFT,

$$X_4(\hat{\omega}_k) = 2X_1(\hat{\omega}_k) + X_2(\hat{\omega}_k) + e^{j\pi k/4}X_1(\hat{\omega}_k)$$

Meanwhile, you can see that $X_4(k)$ is real since $x_4(n)$ is symmetric, that is, an even function.

3. (15 points) [Convolution] The *impulse* or "unit pulse" signal is defined by

$$\delta(n) \triangleq \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

For example, $\delta = [1, 0, 0, 0]$ for $N = 4$.

(a) Verify that the impulse signal is the *identity element* under convolution using the impulse signal $\delta = [1, 0, 0, 0]$ and the input signal $x = [1, -1, 1, -1]$. That is, show that $x * \delta = x$.
Solution:

$$(x * \delta)(n) \triangleq \sum_{m=0}^{N-1} x(m)\delta(n - m)$$

$$\begin{aligned}
(x * \delta)(0) &= x(0)\delta(0) + x(1)\delta(0-1) + x(2)\delta(0-2) + x(3)\delta(0-3) = 1 \\
(x * \delta)(1) &= x(0)\delta(1) + x(1)\delta(1-1) + x(2)\delta(1-2) + x(3)\delta(1-3) = -1 \\
(x * \delta)(2) &= x(0)\delta(2) + x(1)\delta(2-1) + x(2)\delta(2-2) + x(3)\delta(2-3) = 1 \\
(x * \delta)(3) &= x(0)\delta(3) + x(1)\delta(3-1) + x(2)\delta(3-2) + x(3)\delta(3-3) = -1
\end{aligned}$$

Therefore, since $(x*\delta)(n) = x(n)$, the impulse signal is the *identity element* under convolution.

(b) Show that $x * \text{Shift}_1(\delta) = \text{Shift}_1(x)$, where $\textsc{Shift}_{1,n}(x) \triangleq x(n-1)$.
Solution: $x * \text{Shift}_1(\delta)$:

$$\begin{aligned}
(x * \text{Shift}_1(\delta))_n &= \sum_{m=0}^{3} x(m)\delta((n-1) - m) \\
&= x(n-1) = (\text{Shift}_1(x))_n
\end{aligned}$$

Therefore, $x * \text{Shift}_1(\delta) = \text{Shift}_1(x)$

(c) Find $(x * [1, 1, 0, 0, \ldots])_n$.
(Hint: use linearity of convolution and the preceding results)
Solution: Since $[1, 1, 0, 0, \ldots] = \delta + \text{Shift}_1(\delta)$,

$$\begin{aligned}
x * [1, 1, 0, 0, \ldots] &= x * (\delta + \text{Shift}_1(\delta)) \\
&= (x * \delta) + (x * \text{Shift}_1(\delta)) \\
&= x + \text{Shift}_1(x)
\end{aligned}$$

Therefore, $([1, 1, 0, 0, \ldots])_n = (x + \text{Shift}_1(x))_n$

4. (20 pts) [Convolution] For the signals $x(n) = [0, 0, 1, 0, 0, 0, 1, 0]$ and $h(n) = [1, 1, 1, 1, 1, 0, 0, 0]$ compute:

(a) $y(n) = (x * h)_n$

Solution:

Linear convolution is defined as $y(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)$. Therefore,

$$
\begin{aligned}
y(0) &= 0+0+0+0+0+0+0+0 = 0 \\
y(1) &= 0+0+0+0+0+0+0+0 = 0 \\
y(2) &= 0+0+1+0+0+0+0+0 = 1 \\
y(3) &= 0+0+1+0+0+0+0+0 = 1 \\
y(4) &= 0+0+1+0+0+0+0+0 = 1 \\
y(5) &= 0+0+1+0+0+0+0+0 = 1 \\
y(6) &= 0+0+1+0+0+0+1+0 = 2 \\
y(7) &= 0+0+0+0+0+0+1+0 = 1 \\
y(8) &= 0+0+0+0+0+0+1+0 = 1 \\
y(9) &= 0+0+0+0+0+0+1+0 = 1 \\
y(10) &= 0+0+0+0+0+0+1+0 = 1 \\
y(11) &= 0+0+0+0+0+0+0+0 = 0 \\
y(12) &= 0+0+0+0+0+0+0+0 = 0 \\
y(13) &= 0+0+0+0+0+0+0+0 = 0 \\
y(14) &= 0+0+0+0+0+0+0+0 = 0 \\
y(n) &= 0 \qquad \forall\, n \notin [0, 14]
\end{aligned}
$$

In other words, $y(n) = [0, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0]$

A simpler option (if the students know the shift theorem):

$$
\begin{aligned}
y(n) = (x * h)_n &= (SHIFT_2(\delta) * h)_n + (SHIFT_6(\delta) * h)_n \\
&= [0, 0, 1, 1, 1, 1, 1, 0, 0, 0, \ldots] + [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, \ldots] \\
&= [0, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 0, \ldots]
\end{aligned}
$$

(b) $y(n) = (x \circledast h)_n$

Solution:

Circular convolution is defined as $y(n) = \sum_{m=0}^{N-1} x(m)h(n-m)$. Therefore,

$$
\begin{aligned}
y(0) &= 0+0+0+0+0+0+1+0=1 \\
y(1) &= 0+0+0+0+0+0+1+0=1 \\
y(2) &= 0+0+1+0+0+0+1+0=2 \\
y(3) &= 0+0+1+0+0+0+0+0=1 \\
y(4) &= 0+0+1+0+0+0+0+0=1 \\
y(5) &= 0+0+1+0+0+0+0+0=1 \\
y(6) &= 0+0+1+0+0+0+1+0=2 \\
y(7) &= 0+0+0+0+0+0+1+0=1
\end{aligned}
$$

In other words, $y(n) = [1, 1, 2, 1, 1, 1, 2, 1]$
Alternatively,

$$
\begin{aligned}
y(n) &= (x \circledast h)_n = (SHIFT_2(\delta) \circledast h)_n + (SHIFT_6(\delta) \circledast h)_n \\
&= [0, 0, 1, 1, 1, 1, 1, 0] + [1, 1, 1, 0, 0, 0, 1, 1] \\
&= [1, 1, 2, 1, 1, 1, 2, 1]
\end{aligned}
$$

Lab Assignments

Note: follow the naming conventions of the last Homework

1. (25 pts) [Zero padding] The zero padding factor is the ratio of the length of the zero-padded signal (N) to the length of the original signal(M):

$$\text{zpf} = \frac{N}{M}$$

   where zpf is the zero padding factor, N is the length of the zero-padded signal (also equal to the number of bins in the spectrum), and M is the length of the original signal (or *frame* to be analyzed).

   (a) Create a 1 second sinusoid at 16.0625 Hz using a sampling rate of 128 Hz.

   (b) Using your `plotspec` function from the previous homework (HW 4), plot the magnitude spectrum of the sinusoid. Is the result what you expected?

   (c) Zero-pad your sinusoid by a factor of zpf = 2, and plot its magnitude spectrum. Repeat while increasing the zero padding factor until your spectrum is interpolated enough so that the peak magnitude bin frequency corresponds to the correct frequency of the sinusoid. Turn in your final plot as a Matlab figure file. Specify the value of zpf you used.

Solution:

```
clear all
close all

% a)
%--------------------------------------------
dur=1; %(sec)
fs=128; %(Hz/sec)
A=1;
wo=2*pi*16.0625; %(rad)

dT=1/fs;
t=(0:dT:dur-dT);
SINUSOID=A*sin(wo*t);

plot(t,SINUSOID,'-o')
title('Sinusoid')
xlabel('time(sec)')
ylabel('Amplitude')
grid

% b)
%--------------------------------------------
plotspec (SINUSOID,fs)


% c)
%--------------------------------------------
zpf=2;
SINUSOID_zp=z_pad(SINUSOID,zpf);

figure
plot(SINUSOID_zp)
title('Sinusoid zpf=2')
xlabel('time(sec)')
ylabel('Amplitude')
plotspec (SINUSOID_zp,fs)
title('Spectrum zpf=2')

zpf=16;
SINUSOID_zp=z_pad(SINUSOID,zpf);
```

```
figure
plot(SINUSOID_zp)
title('Sinusoid zpf=16')
xlabel('time(sec)')
ylabel('Amplitude')
plotspec (SINUSOID_zp,fs)
title('Spectrum zpf=16')
```

zero pad Function:

```
function x_zp=z_pad(x,zpf);

N=length(x);
M=N*zpf;
x_zp=[x zeros(1,M-N)];

return
```

2. (40 points) [Short-time Fourier transform] Write a function that generates a spectrogram of an input signal.

```
function myspecgram (x, fs, Nf)

% function myspecgram (x, fs, Nf)
% A function to plot the spectrogram of input signal
% using hann window and zpf = 8
%
% x: input signal (assume a row vector)
% fs: sampling rate of x
% Nf: frame size
%
% Your Name / Lab ##
```

Please note:

(a) Use the hann window and zpf=8.

(b) To produce an image from a matrix, use `image` or `imagesc` function. Try with various colormap and choose one for your plot.

Solution:

```
function X=myspecgram (x, fs, Nf);
```

```
% function myspecgram (x, fs, Nf)
% A function to plot the spectrogram of input signal
% using hann window and zpf = 8
%
% X: Spectrogram
% x: input signal (assume a row vector)
% fs: sampling rate of x
% Nf: frame size

zpf =8;
% force window size to be odd
if rem(Nf,2) == 0
  Nf = Nf + 1;
end
N = length(x);
L = floor(N/Nf);     % number of frame - 1
R = N-L*Nf;          % residue

% pad zeros for the length of x to be integer multiples of Nf
x = [x, zeros(1,(Nf-R))];
L = L+1;             % now L is the number of frame!

% compute hann window
n = 0:(Nf-1);
hann = 1/Nf*(cos((pi/Nf)*(n-(Nf-1)/2))).^2;

% pre-allocate spectrum array
fnum = zpf*(Nf+1)/2;   %% take only half ( it is symmetrical to the other half )
XdB = zeros((1+fnum),L);
Xspec = zeros((1+fnum),L);

k = 1:fnum;
% spectrum array
for i=1:L
  % take one frame and windowing
  xw = hann.*x(((i-1)*Nf+1):((i-1)*Nf+Nf));
  % zeropadding
  if ( zpf > 1 )
    xwz = [xw,zeros(1,(zpf-1)*Nf)];
  else
    xwz = xw;
  end
```

```
  % FFT
  X = fft(xwz);
  Xamp = abs(X(1:(1+fnum)));
  XdB(:,i) = 20*log10(Xamp');
end

% display spectrum
time = 0:Nf/fs:(L-1)*Nf/fs;
freq = [0:(fnum-1)]/(fnum-1)*fs/2;
imagesc(time,freq,XdB);
axis xy;
colormap('copper')
xlabel('Time(sec)')
ylabel('Frecuency(Hz)')
```