Homework #4
Due M 3/04

You must turn in your code as well as output files. Please generate a report that contains the code and output in a single readable format.

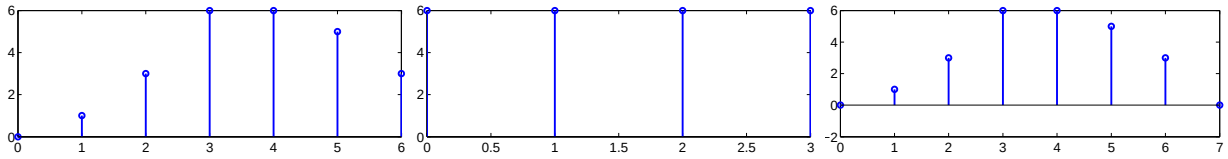
Visit the book website to download companion software, including all the example problems.

`http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118414322.html`

1. (KLT 5.5)

   Solution

   No detailed solutions will be given. Please be sure you can solve these problems by hand. The solutions are presented in Fig 1.



(a) (b) (c)

Figure 1: KLT 5.5

(a) Compute the linear convolution using the flip-and-drag technique.

Case	Summation	Result
$n < 1$	$y(n) = 0$	0
$n - 3 > 3$ $n > 6$	$y(n) = 0$	0
$1 \le n \le 3$	$y(n) = \sum_{k=1}^{n} k$	$[1, 3, 6]$
$1 \le n - 3 < 3$ $4 \le n \le 6$	$y(n) = \sum_{k=n-3}^{3} k$	$[6, 5, 3]$

(b) Use circular convolution technique described in Figure 5.3 on pg 204 of the book.

(c) Repeat (b) after padding $x_1(n)$ and $x_2(n)$ with zeros to length 8 samples (4 zeros padding)

(d) Matlab results plotted in Fig. 1

```
x1 = [1, 1, 1, 1]; % Define x(n)
x2 = [0, 1, 2, 3]; % Define h(n)

%a linear convolution
ylc = conv(x1,x2)

% b circular convolution
Xk = fft(x1);      % Compute X(k)
Hk = fft(x2);      % Compute H(k)
Yk = Xk.*Hk;      % Y(k)=X(k)H(k)
ycc = ifft(Yk)      % Compute and display circular convolution result
```

```
%c linear convolution using padding and circular evaluation
x1p = [x1, 0, 0, 0, 0];
x2p = [x2, 0, 0, 0, 0];
Xk = fft(x1p);      % Compute X(k)
Hk = fft(x2p);      % Compute H(k)
Yk = Xk.*Hk;      % Y(k)=X(k)H(k)
ycl = ifft(Yk)      % Compute and display circular convolution result


ylc =
     0     1     3     6     6     5     3

ycc =
     6     6     6     6

ycl =
         0    1.0    3.0    6.0    6.0    5.0    3.0   -0.0
```

2. (KLT 5.6)

   Solution

   The structure can be obtained by stacking two 8-point DFT sections (Figure 5.6) together and connecting the outputs through the butterfly. The full structure is shown in Fig. 2 where each twiddle term is subscript 16. The ordering of the input signal $x(n)$ can be obtained by the bit-reversal process.

3. (KLT 5.10)

   Solution

   (a) The frequency resolution is computed as

   $$\Delta\omega = \frac{2\pi}{N} = 0.0491 \qquad\qquad \Delta f = \frac{f_s}{N} = 62.5\,\text{Hz}$$

   (b) There are two peaks at $k_1 = f_c/\Delta f = 16$ and at $k_2 = N - k_1$ for the negative copy. See the plot in Fig. 3a.

   (c) The line spectrum exists because this is a periodic signal (shown in Fig. 3b).

   (d) If spectral leakage occurs then the windowing function must be changed to smooth the edge transitions to reduce the widow sidelobe height.

4. (KLT 5.12)

   Solution

```
x = 1:8; % Define x(n)
h = [1, 0, 1, 0, 1, 0, 1, 0]; % Define h(n)

%a linear convolution
ylc = conv(x,h)
```

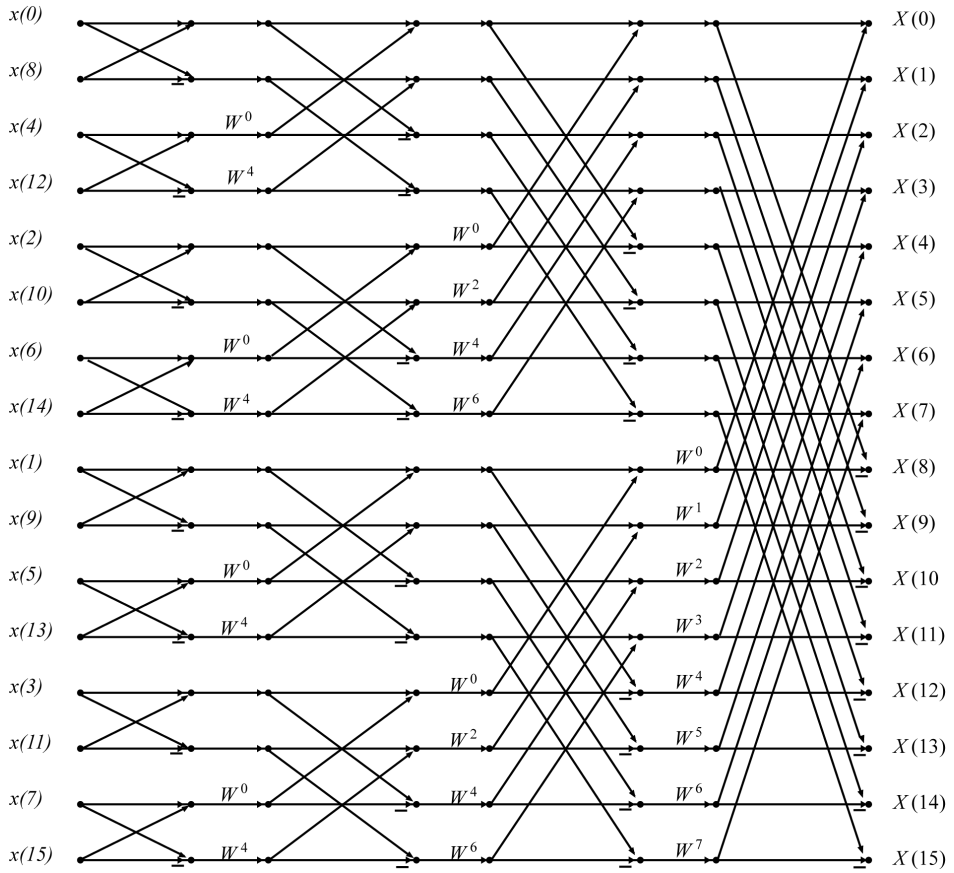
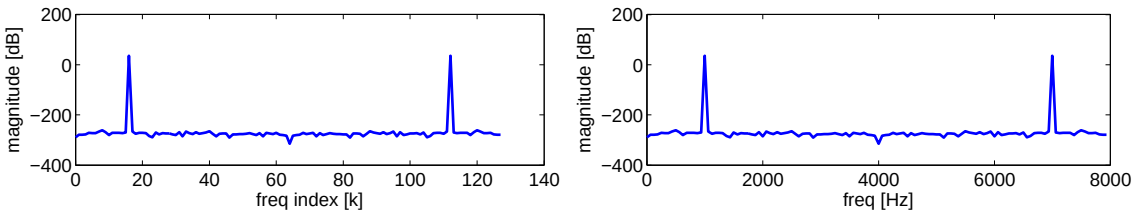


Figure 2: Decimation-in-time FFT signal flow diagram for N=16



(a) (b)

Figure 3: KLT 5.10

```
% b circular convolution
Xk = fft(x);      % Compute X(k)
Hk = fft(h);      % Compute H(k)
Yk = Xk.*Hk;      % Y(k)=X(k)H(k)
ycc = ifft(Yk)      % Compute and display circular convolution result

%c linear convolution using padding and circular evaluation
x1p = [x, zeros(size(x))];
x2p = [h, zeros(size(h))];
Xk = fft(x1p);      % Compute X(k)
Hk = fft(x2p);      % Compute H(k)
```

```
Yk = Xk.*Hk;       % Y(k)=X(k)H(k)
ycl = ifft(Yk)       % Compute and display circular convolution result

%d fast convolution
yff = fftfilt(x1p,x2p)


ylc =
     1     2     4     6     9    12    16    20
     15    18    12    14     7     8     0

ycc =
    16    20    16    20    16    20    16    20

ycl =
    1.0    2.0    4.0    6.0    9.0   12.0   16.0   20.0
    15.0   18.0   12.0   14.0    7.0    8.0         0    0.0

yff =
    1.0    2.0    4.0    6.0    9.0   12.0   16.0   20.0
    15.0   18.0   12.0   14.0    7.0    8.0    0.0    0.0
```

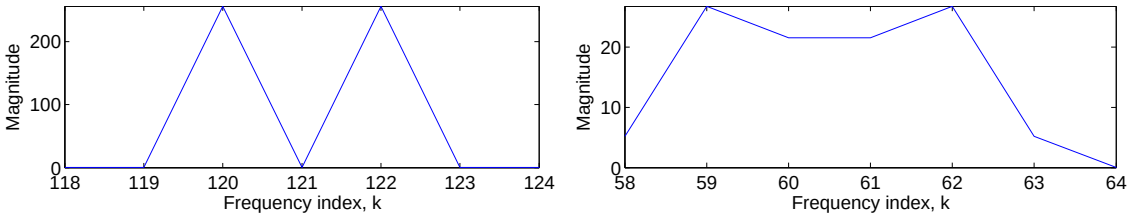
5. (KLT 5.16)

Solution

In order to distinguish the two close sinusoids there must be sufficient resolution and limited spectral leakage. The easiest way to distinguish the two sinusiods is to satisfy the frequency separation condition which results in window length

$$N > \frac{f_s}{\Delta f} = 256.$$

In Fig. 4a the window length is made twice the minimum length for clear separation. In Fig. 4b the window length is set at the minimum but spectral leakage is controlled by using a Blackman window. Notice the two peaks are just able to be distinguished.



(a) (b)

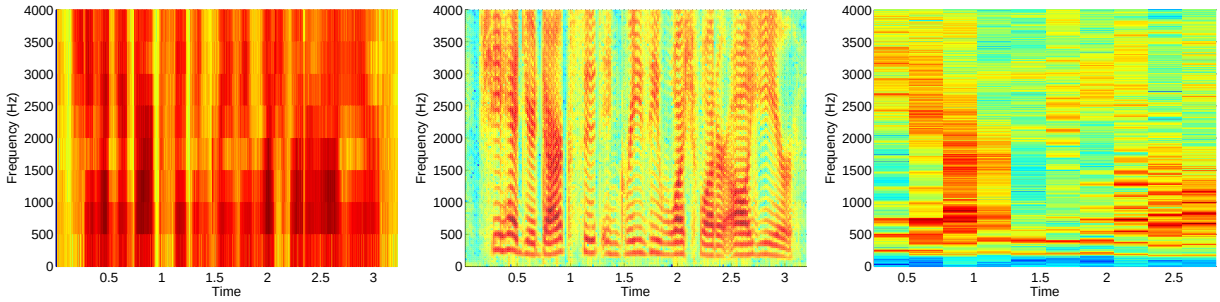
Figure 4: KLT 5.16

6. (KLT 5.17)

Solution

(a) Window Size: When the window size is increased, each short-time Fourier transform (column of spectrogram) is computed on longer speech segments. If the window is too

small (Fig. 5a) then spectral information does not capture the pitch of the speaker. When the window is too long (Fig. 5c) too many harmonics are introduced between changing tone resulting in too many spectral components or less smooth in time and able to track changing pitch.



(a) `window`=16 (b) `window`=256 (c) `window`=4096

Figure 5: KLT 5.17: Varying window size

(b) Overlap: Increasing the window overlap smooths the spectrum. This needs to be viewed on screen to see effects.

(c) FFT Size: Increasing FFT size increases the frequency resolution making the spectrum sharper. This needs to be viewed on screen to see effects.

```
% Speech sampled at 8 kHz, 16 bits
load('timit2.asc');
soundsc(timit2, 8000) % Play the speech

%examine 3 values for each parameter
wvals = 2.^[4 8 12];
oper = [0.25 0.5 0.75];
fvals = [1 2 4];

%% test window size
for win = wvals
    ov = win * 0.5;
    h=figure;
    spectrogram(timit2,win,ov,win,8000,'yaxis');
end

%% test overlap
for ov = oper
    h=figure;
    spectrogram(timit2,256,ov*256,256,8000,'yaxis');
end

%% test fft size
for s = fvals
    h=figure;
    spectrogram(timit2,256,256*0.5,s*256,8000,'yaxis');
end
```