數位信號處理實習

Digital Signal Processing Lab.

蔡偉和

whtsai@ntut.edu.tw
02-27712171 ext. 2257
0933052581
綜科館 **311-4**

Lecture Material:

北科ｉ學園

Grading:

- In-class practice/participation: 50%
- Reports: 50%

Syllabus:

- Part I: Matlab® Simulations on PC
  - Matlab 簡介
  - 實驗一: 描繪訊號
  - 實驗二: Convolution
  - 實驗三: DTFT、DFT 與 FFT
  - 實驗四: 數位 LTI 系統
  - 實驗五: 聲訊處理
  - 實驗六: 濾波器設計
- Part II: C Programming on TMS320C6713 DSK
  - 數位訊號處理器簡介
  - 實驗一: 熟悉 CCS 與 DSK6713
  - 實驗二: DSK6713 記憶體的配置
  - 實驗三: Board Support Library (BSL) 與音訊處理

Part I:

Matlab® Simulations on PC

實驗一：描繪訊號

[範例 1-1] 產生 Fig. 1-1 之弦波訊號。

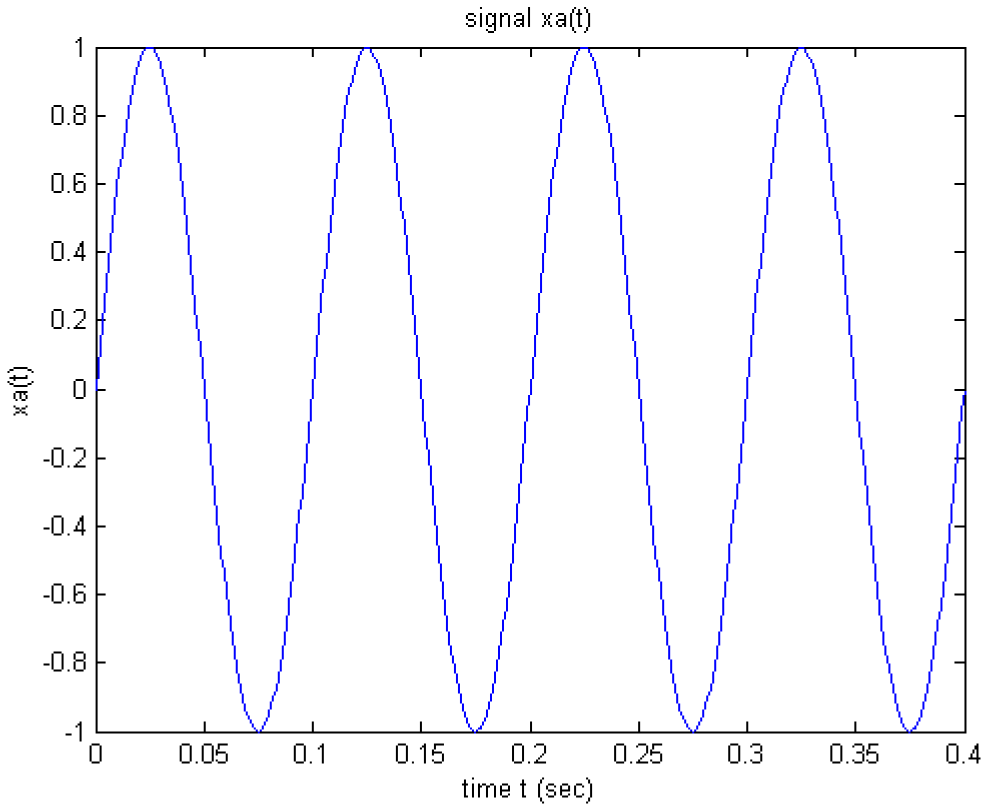


Fig. 1-1

程式 Ex_1_1.m

```
% Generating a single-tone sine signal xa
clear;
f0=10;          % 10 Hz sine wave
dt=0.001;       % resolution
Length=0.4      % Total length =0.4 sec
t=0:dt:Length;
xa=sin(2*pi*f0*t);
plot(t,xa);
xlabel('time t (sec)'); ylabel('xa(t)');
title('signal xa(t)');
```

****

[練習 1-1] 繪出訊號 $\mathrm{sinc}(t) = \dfrac{\sin \pi t}{\pi t}$ 與 $\mathrm{sinc}^2(t)$ for $-3 \le t \le 3$。

****

[範例 1-2] 將 Fig. 1-1 之弦波訊號以 0.01 second 之取樣週期表示成 Fig. 1-2 之弦波離散訊號。

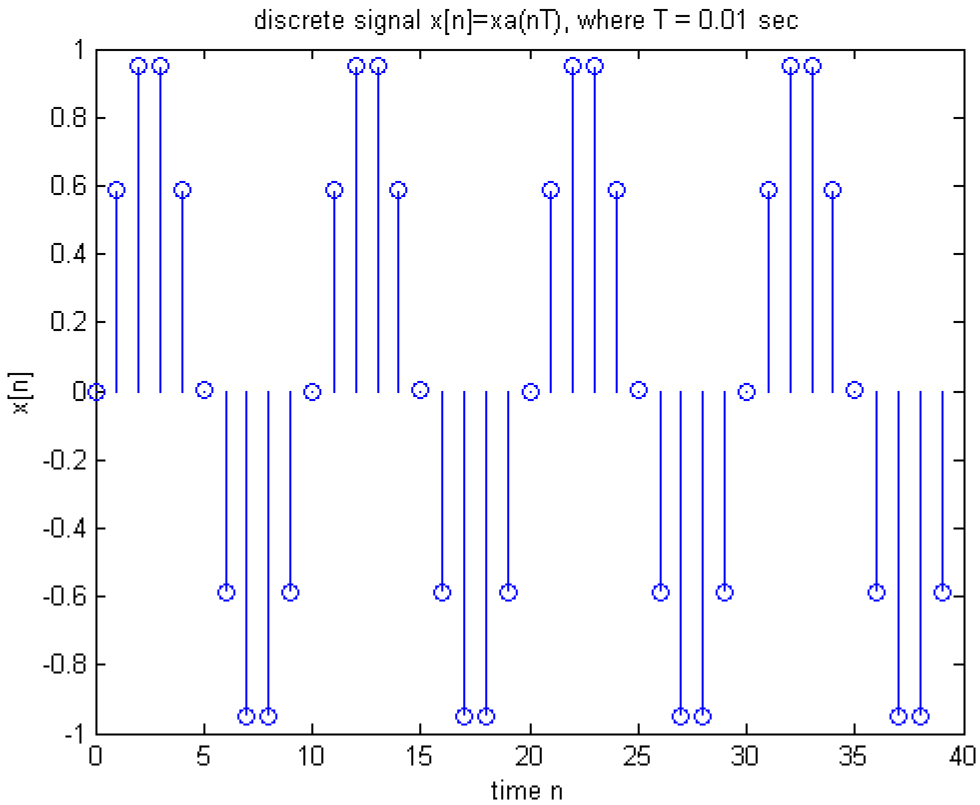


Fig. 1-2

程式 Ex_1_2.m

```
% Generating a discrete-time signal x
clear;
f0=10;           % 10 Hz sine wave
Length=0.4        % Total length =0.4 sec
T=0.01;           % sampling period = 0.01 sec
N=Length/T;
n=0:1:N-1;
x=sin(2*pi*f0*n*T);
stem(n,x);
xlabel('time n'); ylabel('x[n]');
title('discrete signal x[n]=xa(nT), where T = 0.01 sec');
```

****

[練習 1-2] 繪出訊號 $x[n]=\dfrac{\sin w_c n}{\pi n}$, where $w_c = 0.2\pi,\ -30 \le n \le 30$。

[練習 1-3] 繪出一包含 10Hz 與 30Hz 之弦波離散訊號，其中取樣週期為 0.01 second。

****

實驗二：Convolution

[範例 2-1] 繪出兩訊號 $x_1[n]$與 $x_2[n]$之 Convolution $y[n]=\sum_{k=-\infty}^{\infty}x_1[k]x_2[n-k]=\sum_{k=-\infty}^{\infty}x_2[k]x_1[n-k]$

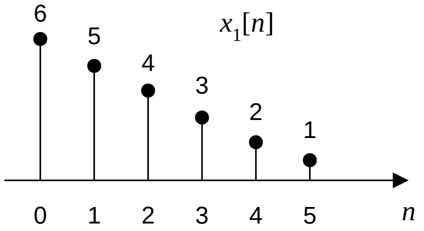


Fig. 2-1-1(a)

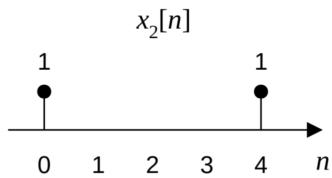


Fig. 2-1-1(b)

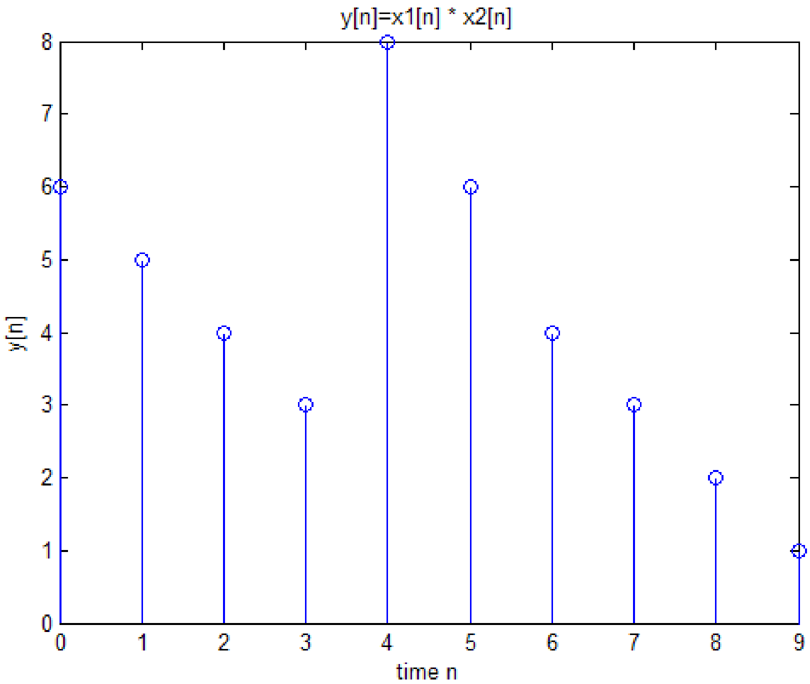


Fig. 2-1-2

程式 Ex_2_1.m

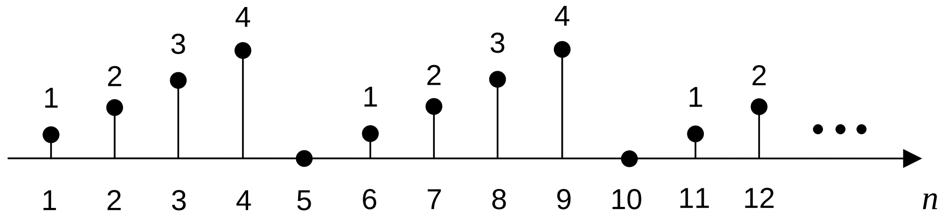
```
% y[n]=x1[n] * x2[n]
clear;
x1=[6 5 4 3 2 1];
x2=[1 0 0 0 1];
y=conv(x1,x2);
n=1:length(y);
stem(n,y);
xlabel('time n'); ylabel('y[n]');
title('y[n]=x1[n] * x2[n]');
```

****

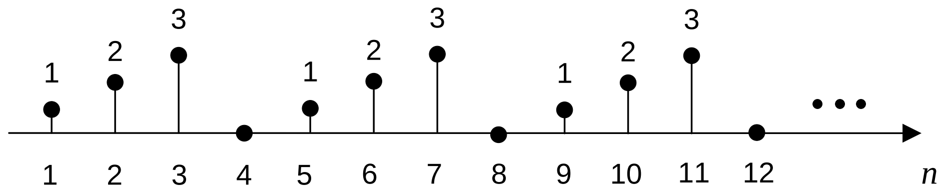
[練習 2-1] 不使用內建函式 conv()而改以迴圈累加方式來實現 Convolution。

[練習 2-2] 若有兩訊號 $x_1[n] = n \% 5, \quad x_2[n] = n \% 4, \quad 1 \le n \le 1000$

$x_1[n]$



$x_2[n]$



試利用矩陣運算方式來實現 Convolution，並比較此方式與上題迴圈累加方式的運算速度差別。

****

實驗三：DTFT、DFT 與 FFT

[範例 3-1] 繪出訊號 $x[n]$之 Discrete-Time Fourier Transform (DTFT) $X(e^{jw})=\sum_{n=-\infty}^{\infty}x[n]e^{-jwn}$ 。

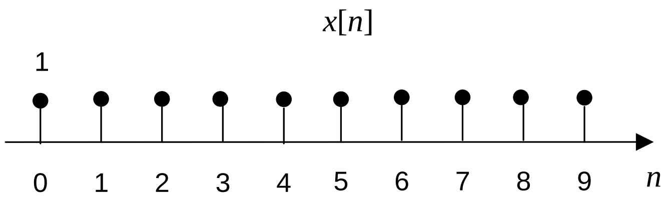


Fig. 3-1-1

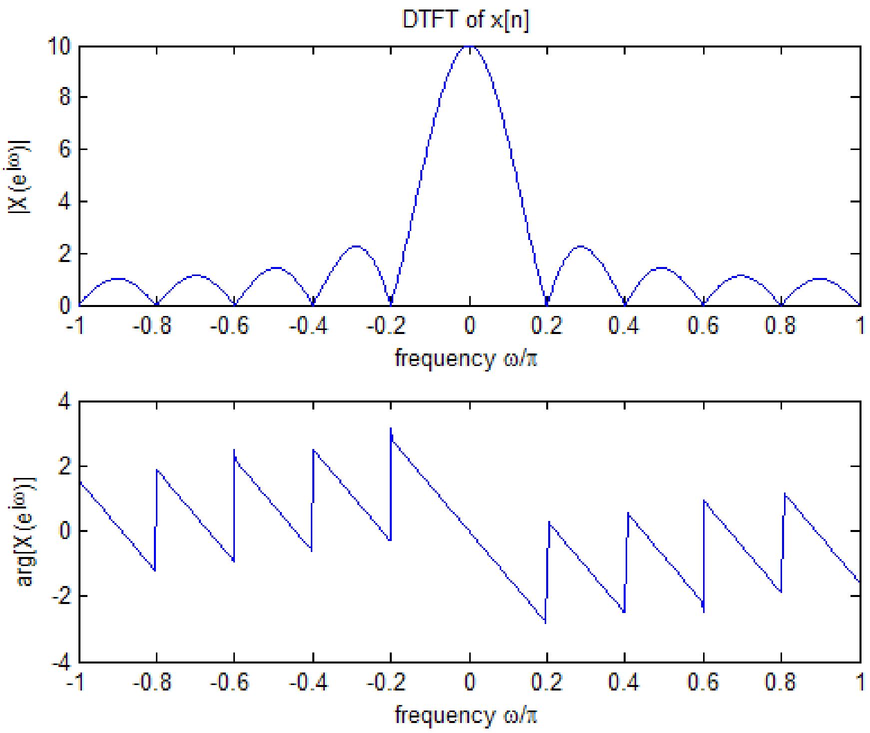


Fig. 3-1-2

程式 Ex_3_1.m

```
% Computing the DTFT of signal x
clear;
x=[1 1 1 1 1 1 1 1 1 1];
n=0:length(x)-1;
K=500;
k=-K:K;
w=pi*k/K;
X=x*exp(-j*n'*w);
magX=abs(X);
angX=angle(X);
title('DTFT of x[n]');
subplot(2,1,1); plot(w/pi,magX);
xlabel('frequency \omega/\pi');    ylabel('|X(e^j^\omega)|');
subplot(2,1,2); plot(w/pi,angX);
xlabel('frequency \omega/\pi');    ylabel('arg(X(e^j^\omega))');
```

****

[練習 3-1] 不使用內建函式 exp()、abs()、與 angle()而改以下列的方式來實現 DTFT:

$X(e^{jw})=\sum_{n=-\infty}^{\infty}x[n]e^{-jwn}=\sum_{n=-\infty}^{\infty}\{x[n]\cos(wn)-jx[n]\sin(wn)\}=X_R(e^{jw})+jX_I(e^{jw})$ 分成實部與虛部

，則振幅為 $\sqrt{X_R^2(e^{jw})+X_I^2(e^{jw})}$ ，相位為 $\tan^{-1}\left[\frac{X_I(e^{jw})}{X_R(e^{jw})}\right]$ 。

****

[範例 3-2] 觀察 Gibbs phenomenon。

我們知道 $x[n]=\frac{\sin w_c n}{\pi n} \quad\leftrightarrow\quad X(e^{jw})=\begin{cases}1, & |w|\le w_c \\ 0, & w_c<|w|<\pi\end{cases}$, $-\infty\le n\le\infty$，但若取有限 n，$-M\le n$ $\le M$，並計算 DTFT，則可發現振幅上有振盪現象，即所謂 Gibbs phenomenon。

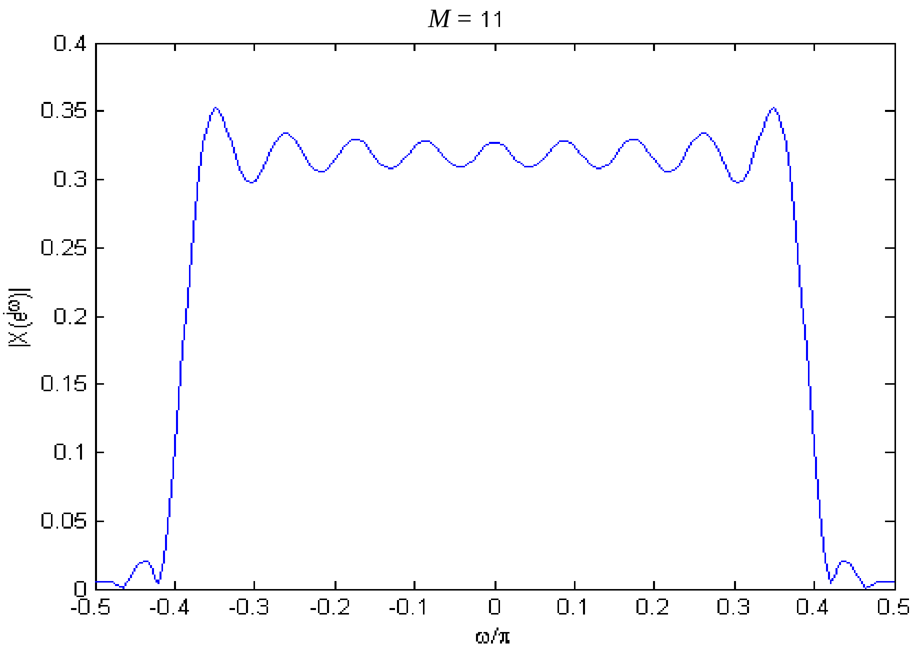


Fig. 3-2

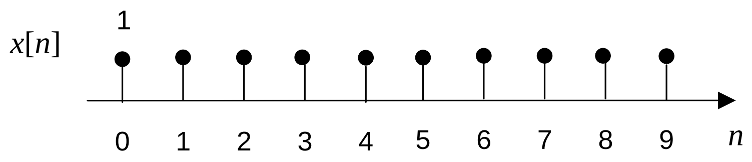
程式 Ex_3_2.m

```
f = linspace(-0.5,0.5,200);    % [-0.5pi, 0.5pi]
wc = 0.25*pi;                 % cutoff frequency

for M=1:5:2000
   n = (-M:M);
   for j=1:length(f)
      X(j) = sum(wc/pi*sinc(wc*n).*exp(-i*2*pi*n*abs(f(j))));
   end
   plot(f,abs(X));
```

```
    xlabel('\omega/\pi');
    ylabel('|X(e^j^\omega)|');
    title(M);
    pause;
end
```

[範例 3-3] 繪出訊號 $x[n]$之 N-point Discrete Fourier Transform (DFT) $X[k]=\sum_{n=0}^{N-1}x[n]e^{-j\frac{2\pi}{N}kn}$，

其中 *N* 分別取 10 與 100，觀察結果可發現 *N* 取 10 時的 DFT 與 Fig. 3-1 中的 DTFT 相差甚大，而當 *N* 取 100 時的 DFT 則較相似於 Fig. 3-1 中的 DTFT。



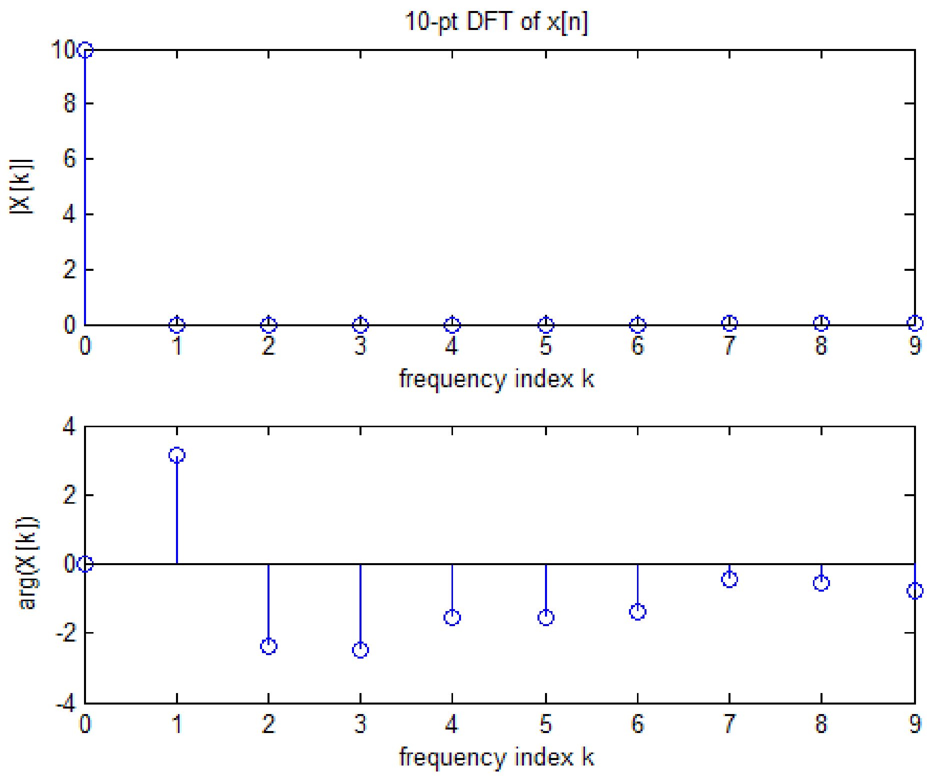


Fig. 3-3

程式 Ex_3_3.m

```
% Computing the DFT of signal x
clear;
x=[1 1 1 1 1 1 1 1 1 1];
n=0:length(x)-1;
N=5;
k=0:N-1;
X=x*exp(-j*2*pi/N*n'*k);
magX=abs(X);
angX=angle(X);
```

```
subplot(2,1,1); stem(k,magX);  xlabel('frequency index k');
ylabel('|X[k]|');
title('5-pt DFT of x[n]');
subplot(2,1,2); stem(k,angX);  xlabel('frequency index k');
ylabel('arg(X[k])');
```

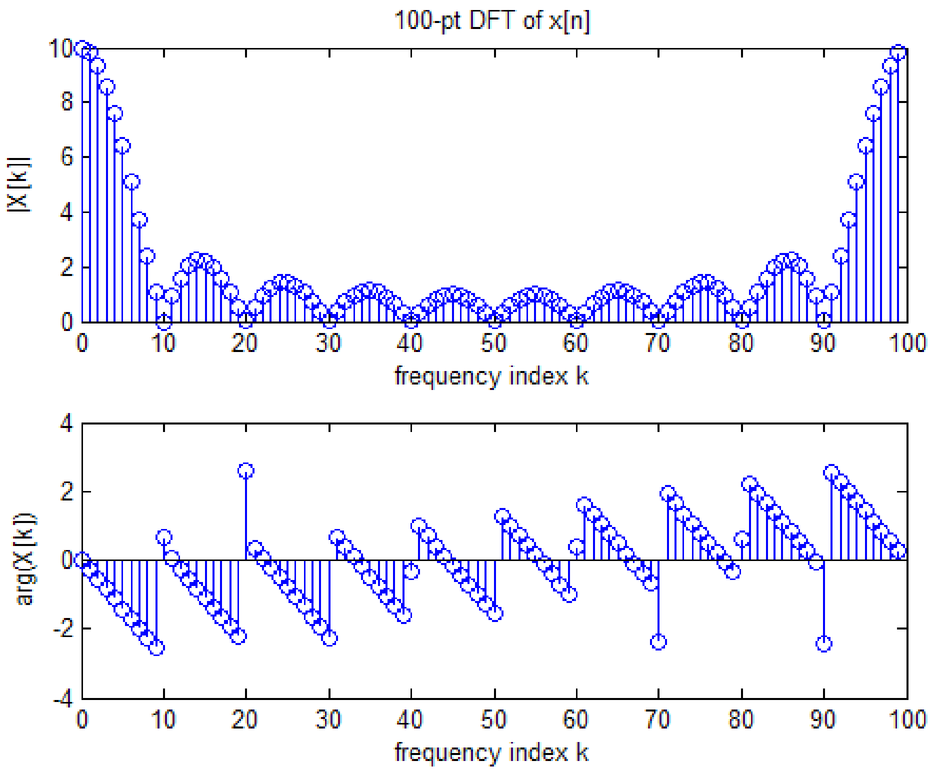


Fig. 3-4

****

[練習 3-2] 取 Fig. 1-2 之離散弦波訊號的一個週期(10 points)進行 DFT，分別繪出 10-point 與 100-point DFT，討論兩者差異。

[練習 3-3] 使用函式 fft()計算練習 3-2 的 DFT，並繪出結果。

[練習 3-4] 在練習 1-3 中曾繪出一包含 10Hz 與 30Hz 之弦波離散訊號，其中取樣週期為 0.01 second，試利用函式 fft()計算此訊號前 10 points 之 DFT。

[練習 3-5] 若上述包含 10Hz 與 30Hz 之弦波離散訊號是透過取樣週期為 0.02 second 所產生，試利用函式 fft()計算此訊號前 10 points 之 DFT，討論此結果與練習 3-4 有何差異。

****

實驗四：數位 LTI 系統

[範例 4-1] 若有一訊號如 Fig. 4-1 所示輸入一系統 y[n] = x[n] - x[n-1] (此為 backward difference system)，其輸出可由兩種方式計算：一為 filtering，另一為 convolution。若以 filtering 方式計算，結果為 [21 1 1]，若以 convolution 方式計算，結果為 [21 1 1 -5]。在程式 Ex_4_1.m 中，我們可驗證：「若 x[n] Ⓝ h[n] = y[n]，則 X[k] · H[k] = Y[k]」。

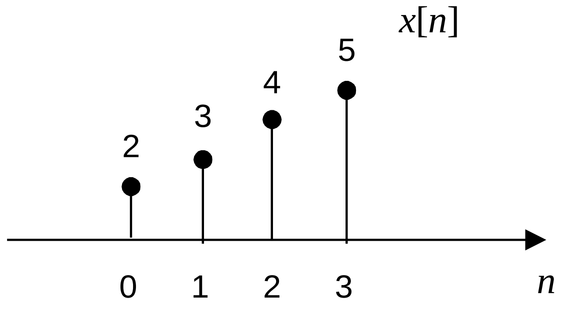


Fig. 4-1

程式 Ex_4_1.m

```
% Backward difference system y[n] = x[n] - x[n-1]
a=[1];
b=[1 -1];
x=[2 3 4 5];
y=filter(b,a,x)      % y = [2 1 1 1]

h=[1 -1];
x=[2 3 4 5];
w=conv(h,x)   % w = [2 1 1 1 -5]

h1=[1 -1 0 0 0];
x1=[2 3 4 5 0];
H=fft(h1)
X=fft(x1)
Z=H.*X
W=fft(w)         % Z = W
```

****

[練習 4-1] 將 Fig. 4-1 之訊號輸入一系統 y[n] = 0.8y[n-1] + x[n] - x[n-1]，試分別利用 filtering 與 convolution 計算輸出結果，並比較兩者在頻域上的差異。

****

[範例 4-2] 將 $H(z)=\dfrac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots}{a_0+a_1z^{-1}+a_2z^{-2}+\cdots}$ 拆解為 $H(z)=\dfrac{(1-d_1z^{-1})(1-d_2z^{-1})\cdots}{(1-c_1z^{-1})(1-c_2z^{-1})\cdots}$，並繪出 pole-zero plot。例如，$H(z)=\dfrac{1-5z^{-1}+6z^{-2}}{1-4.5z^{-1}+2z^{-2}}=\dfrac{(1-3z^{-1})(1-2z^{-1})}{(1-4z^{-1})(1-0.5z^{-1})}$，我們可藉以了解系統是否為穩定。注意，transfer function $H(z)$往往會因數值上的小差異而使系統的特性迥異，例如，

$H_1(z)=\dfrac{1}{1-1.845z^{-1}+0.850586z^{-2}}=\dfrac{1}{(1-0.943z^{-1})(1-0.902z^{-1})}$，其所有 poles 都在單位圓內，但若將數值四捨五入後，

$H_2(z)=\dfrac{1}{1-1.85z^{-1}+0.85z^{-2}}=\dfrac{1}{(1-z^{-1})(1-0.85z^{-1})}$，將發生有些 poles 不在單位圓內，因此可能造成系統不穩定。

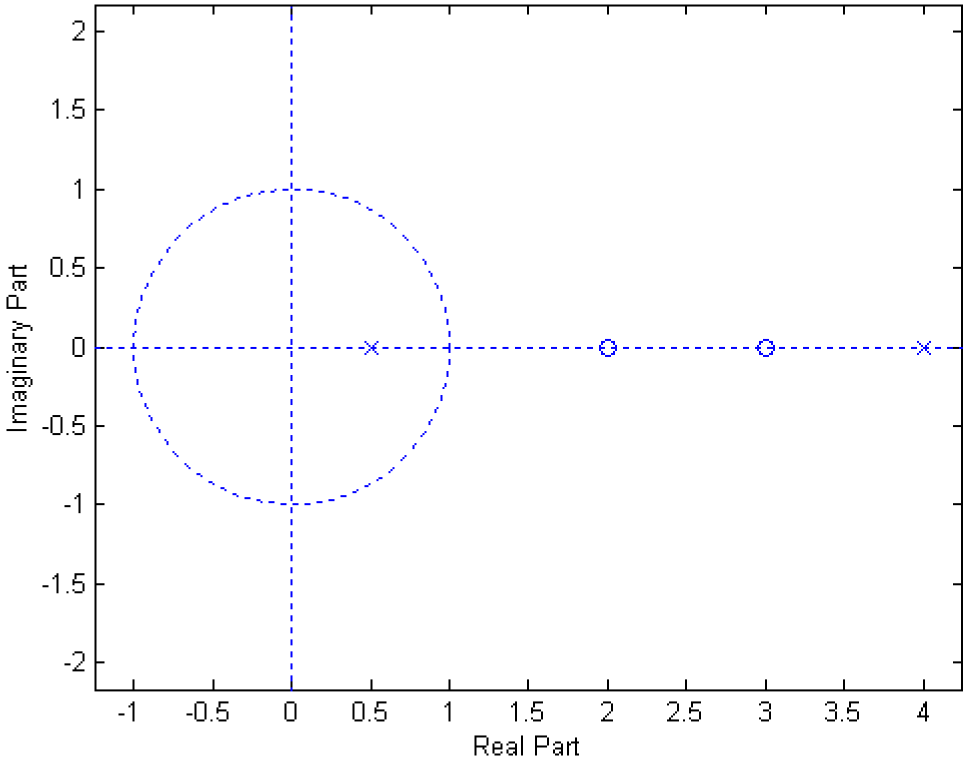


Fig. 4-2

程式 Ex_4_2.m

```
% pole-zero plot of H(z)
num = input('Type in the numerator coefficients (e.g., [1 -5 6]) = ');
den = input('Type in the denominator coefficients (e.g., [1 -4.5 2])
= ');
roots(num)
roots(den)
zplane(num,den)
```

[範例 4-3] 將一頻率 10Hz 之弦波訊號輸入至一 allpass system $H(z)=\dfrac{z^{-1}-0.5}{1-0.5z^{-1}}$ ，可發現輸出訊號仍為 10Hz 之弦波訊號。

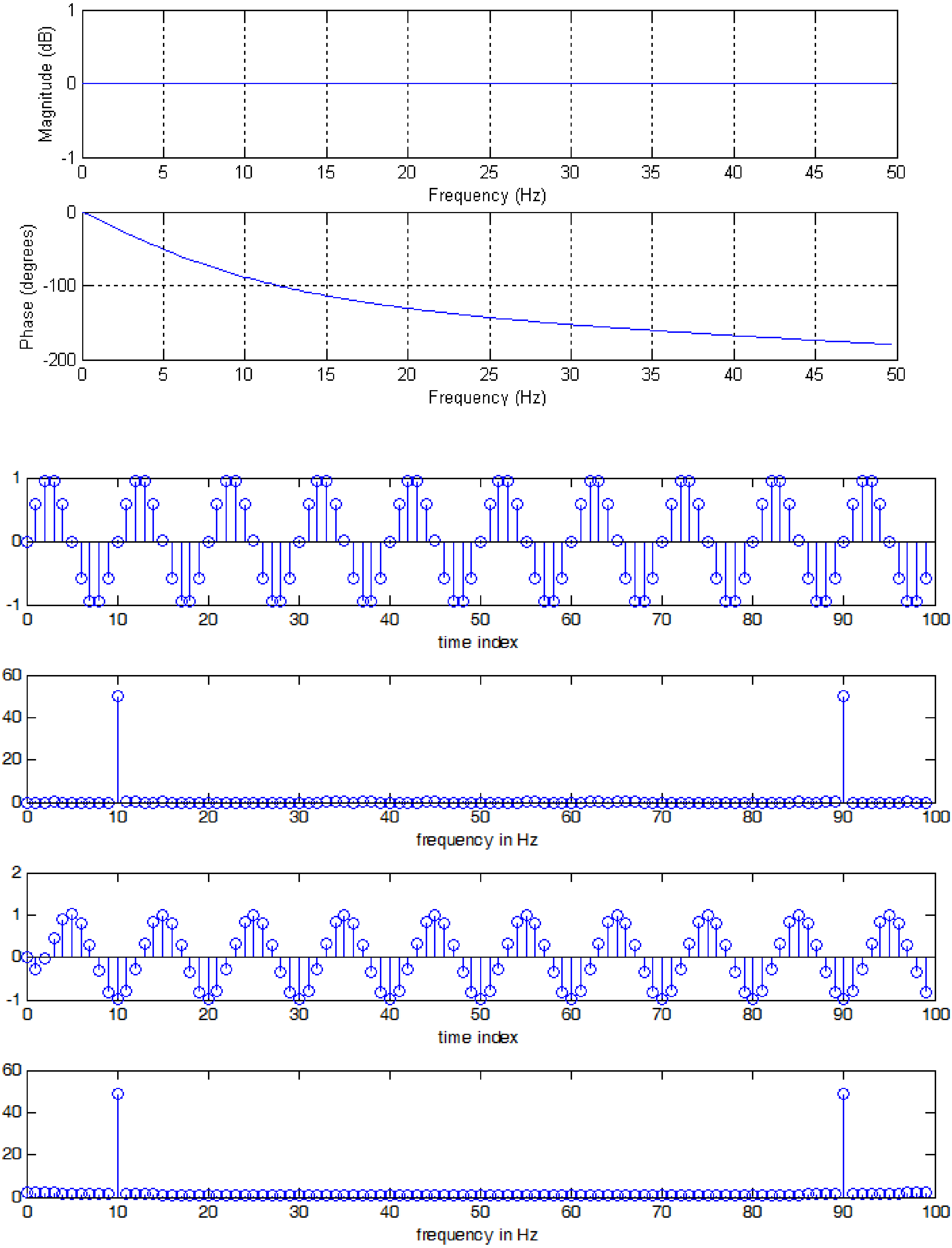


Fig. 4-3

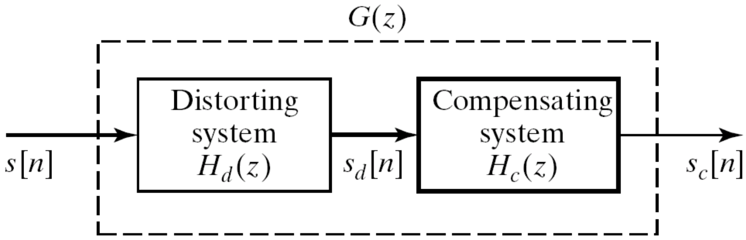
程式 Ex_4_3.m

```
% a 10-Hz sine wave is input to an allpass system
num = [-0.5 1];
den = [1 -0.5];
freqz(num,den,200,100);
pause;
zplane(num,den);
pause;

f0=10;          % 10 Hz sine wave
T=0.01;        % sampling freq. = 100 Hz
N=100;
n=0:1:N-1;
x=sin(2*pi*f0*n*T);
subplot(4,1,1); stem(n,x);
xlabel('time index'); ylabel('x[n]');

f=n/T/N;
subplot(4,1,2); stem(f,abs(fft(x)));
xlabel('frequency in Hz'); ylabel('|X(f)|');


y=filter(num,den,x);
subplot(4,1,3); stem(n,y);
xlabel('time index'); ylabel('y[n]');
subplot(4,1,4); stem(f,abs(fft(y)));
xlabel('frequency in Hz'); ylabel('|Y(f)|');
```

****

[練習 4-2] 設計一如下圖之頻率補償系統 $H_d(z)$(必需為 minimum phase system)，並將一具有 10Hz 與 30Hz 之弦波訊號 *s*[*n*]輸入一 Distorting system

$H_c(z) = \dfrac{1-6.9z^{-1}+13.4z^{-2}-7.2z^{-3}}{1-1.3z^{-1}+0.47z^{-2}-0.035z^{-3}}$，繪出 $s_c[n]$及其 DFT。



****

實驗五：聲訊處理

[範例 5-1] 利用 Matlab 函式錄音，其中取樣頻率設為 16000Hz，聲音經由類比轉數位後儲存為”.wav”檔案。再利用 Matlab 函式讀取音檔，繪出波形與頻譜圖(spectrogram)，並播放聲音。

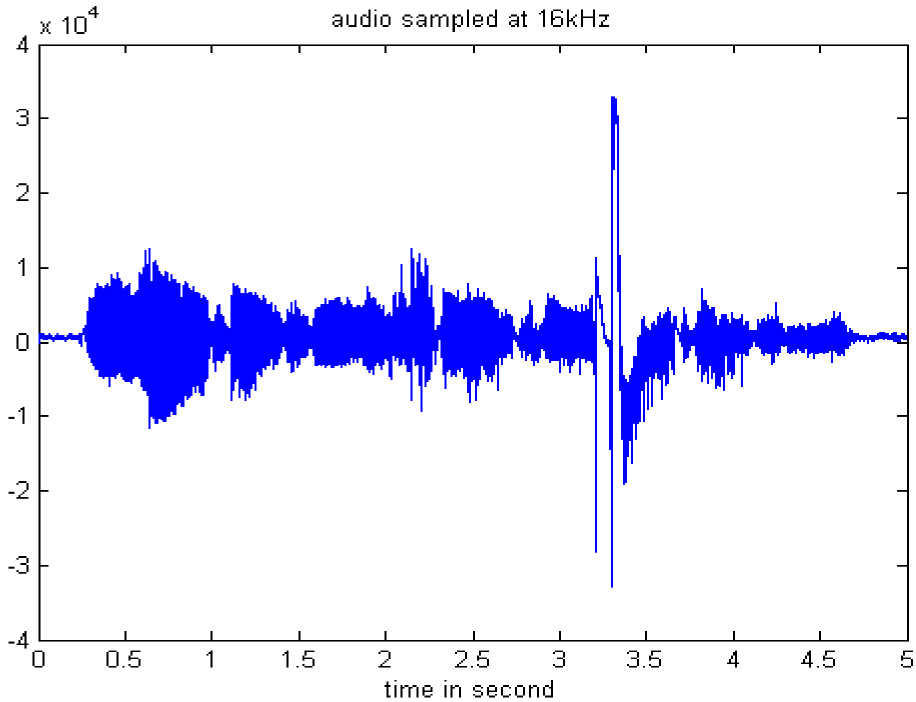


Fig. 5-1-1

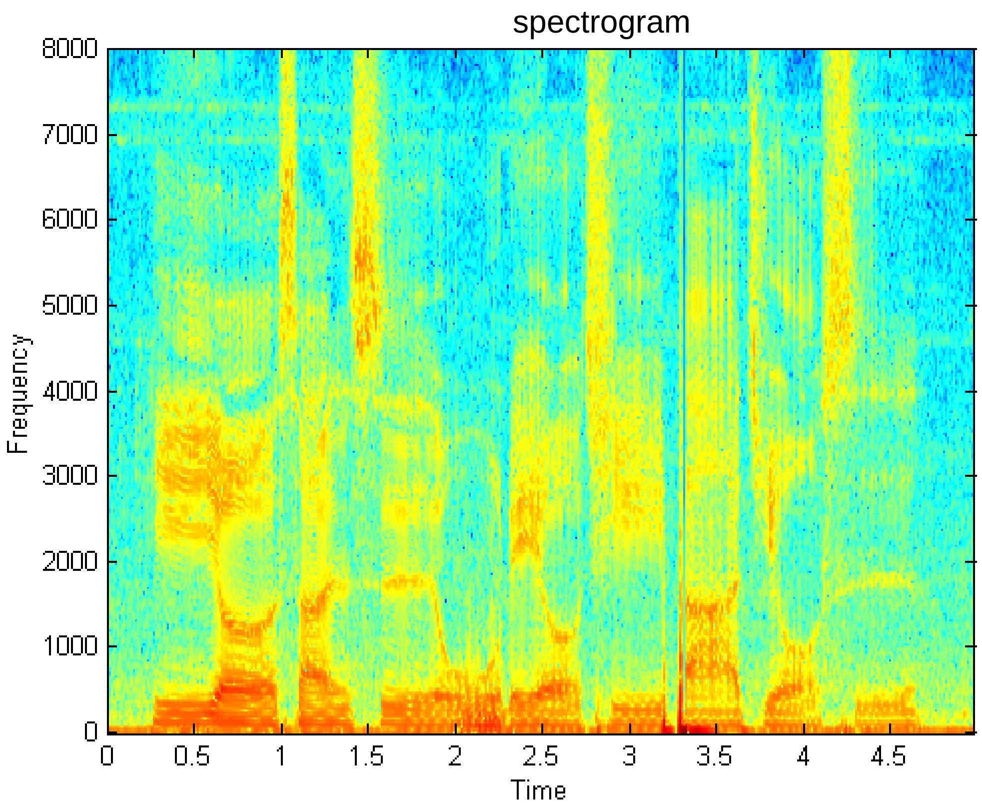


Fig. 5-1-2

程式 Ex_5_1.m

```
% Record and play back 5 seconds of 16-bit audio sampled at 16000 kHz.
Fs=16000;
x = wavrecord(5*Fs, Fs);
wavplay(x,Fs);           % playback
wavwrite(x,Fs,'16kHz.wav');
```

程式 Ex_5_2.m

```
% Read, plot, and play a wav file.
fp=fopen('16kHz.wav','r');
fseek(fp,44,-1)
x=fread(fp,'short');
Fs=16000;
n=0:length(x)-1;
t=n/Fs;
plot(t,x);
xlabel('time in second')
title('audio sampled at 16kHz');
sound(x./32766,Fs,16)
specgram(x,512,Fs,320);
```

[範例 5-2] 利用 Matlab 函式進行 downsampling 與 upsampling。注意在 Matlab 函式中 `downsample` 其實是 decimation (↓)，`upsample` 其實是 sampling rate expansion (↑)；而 `decimate` 才是 downsampling (Lowpass filtering + decimation)，`interp` 才是 upsampling (sampling rate expansion + Lowpass filtering)。

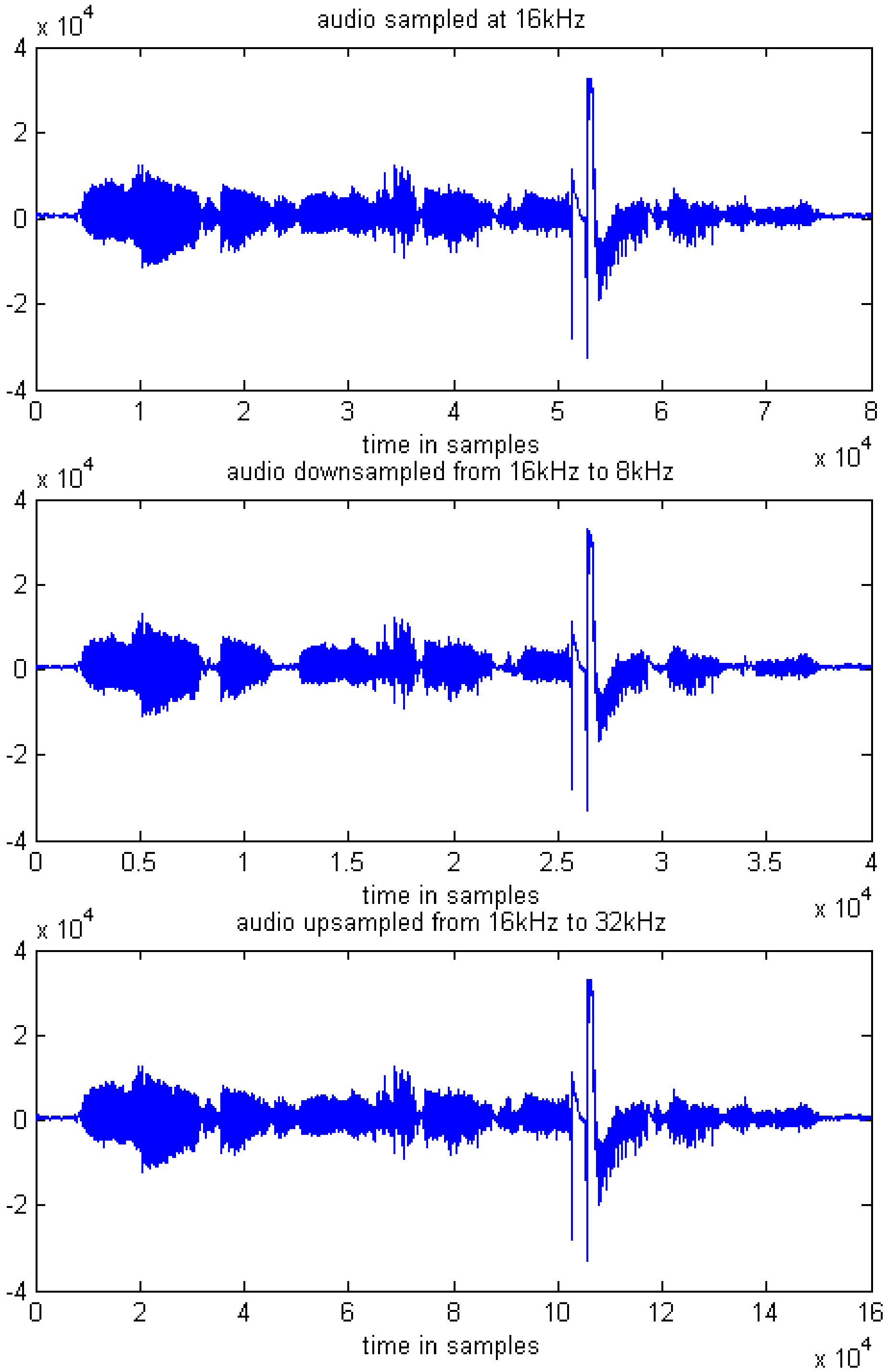
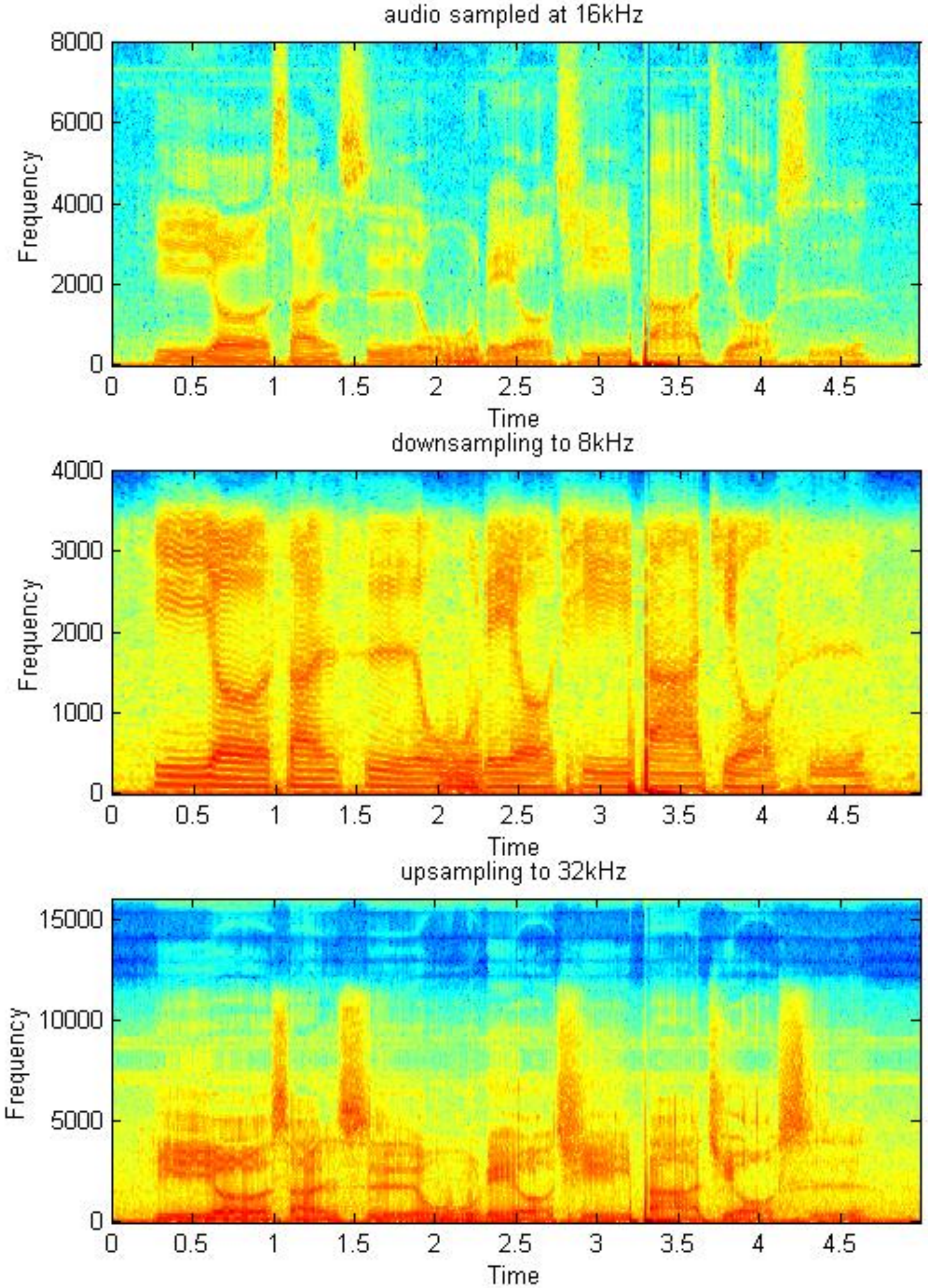


Fig. 5-2-1



audio sampled at 16kHz
Frequency
Time
downsampling to 8kHz
Frequency
Time
upsampling to 32kHz
Frequency
Time

Fig. 5-2-2

程式 Ex_5_3.m

```
fp=fopen('16kHz.wav','r');
fseek(fp,44,-1)
x=fread(fp,'short');
Fs=16000;
subplot(3,1,1); plot(x);
xlabel('time in samples')
title('audio sampled at 16kHz');
sound(x./32766,Fs,16)
pause;
prompt='press any key'

% downsampling
y=decimate(x,2);
subplot(3,1,2); plot(y);
xlabel('time in samples')
title('audio downsampled from 16kHz to 8kHz');
sound(y./32766,Fs/2,16)
pause;
prompt='press any key'

% upsampling
z=interp(x,2);
subplot(3,1,3); plot(z);
xlabel('time in samples')
title('audio upsampled from 16kHz to 32kHz');
sound(z./32766,Fs*2,16)
prompt='press any key'
subplot(3,1,1); specgram(x,512,Fs,320);
title('audio sampled at 16kHz');
subplot(3,1,2); specgram(y,512,Fs/2,320);
title('downsampling to 8kHz');
subplot(3,1,3); specgram(z,512,Fs*2,320);
title('upsampling to 32kHz');
```

****

[練習 5-1] 將聲音訊號分別進行 ↓2 與 ↑2 (不使用 lowpass filters)，繪出其 spectrograms。

****

****

[練習 5-2] 產生一段歌曲訊號 So Mi Mi Fa Re Re Do Re Mi Fa So So So ; So Mi Mi Fa Re Re Do Mi So So Do，並使用 Matlab 函式 sound.m 以取樣頻率 8000Hz 播放出來。

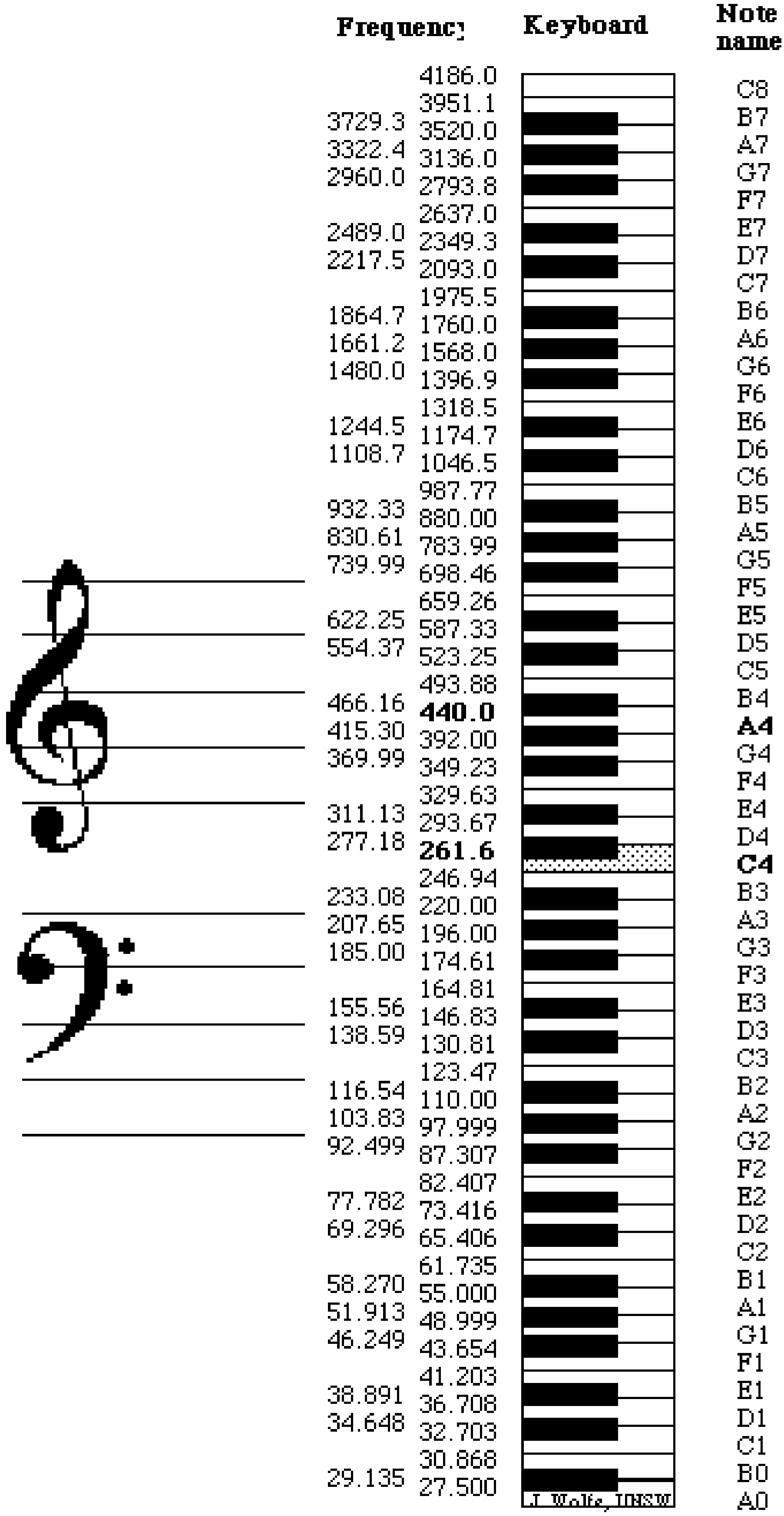


Fig. 5-3

****

實驗六：濾波器設計

[範例 6-1] 利用 Matlab 函式設計一 Butterworth 低通濾波器，cut-off frequency 為 0.4π rad/sec，其中若取樣頻率為 100 Hz，則 cut-off frequency 為 20 Hz。將練習 1-3 之訊號通過此低通濾波器，則可見到訊號中大部分的 30 Hz 成份已受到濾除。

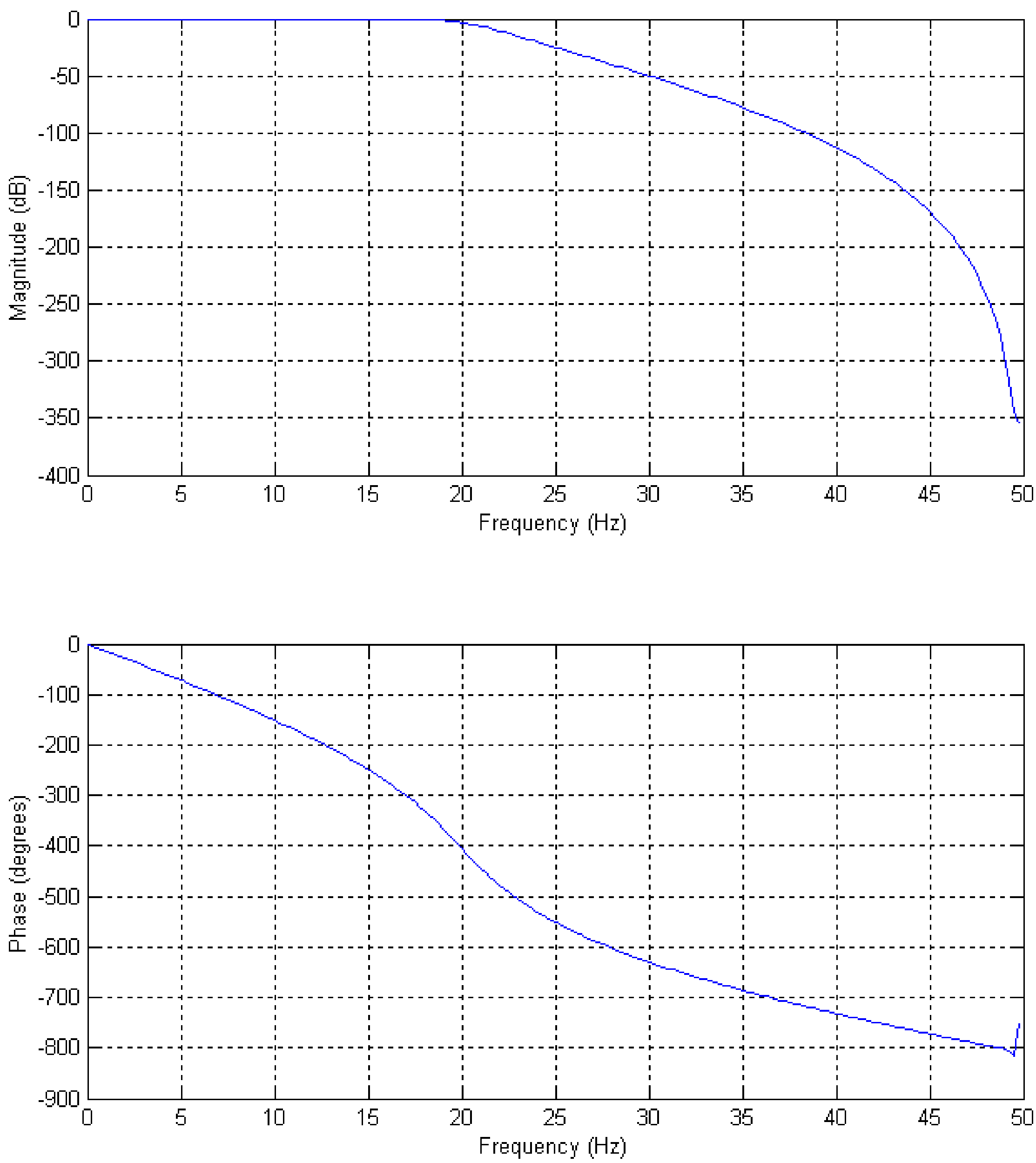


Fig. 6-1-1

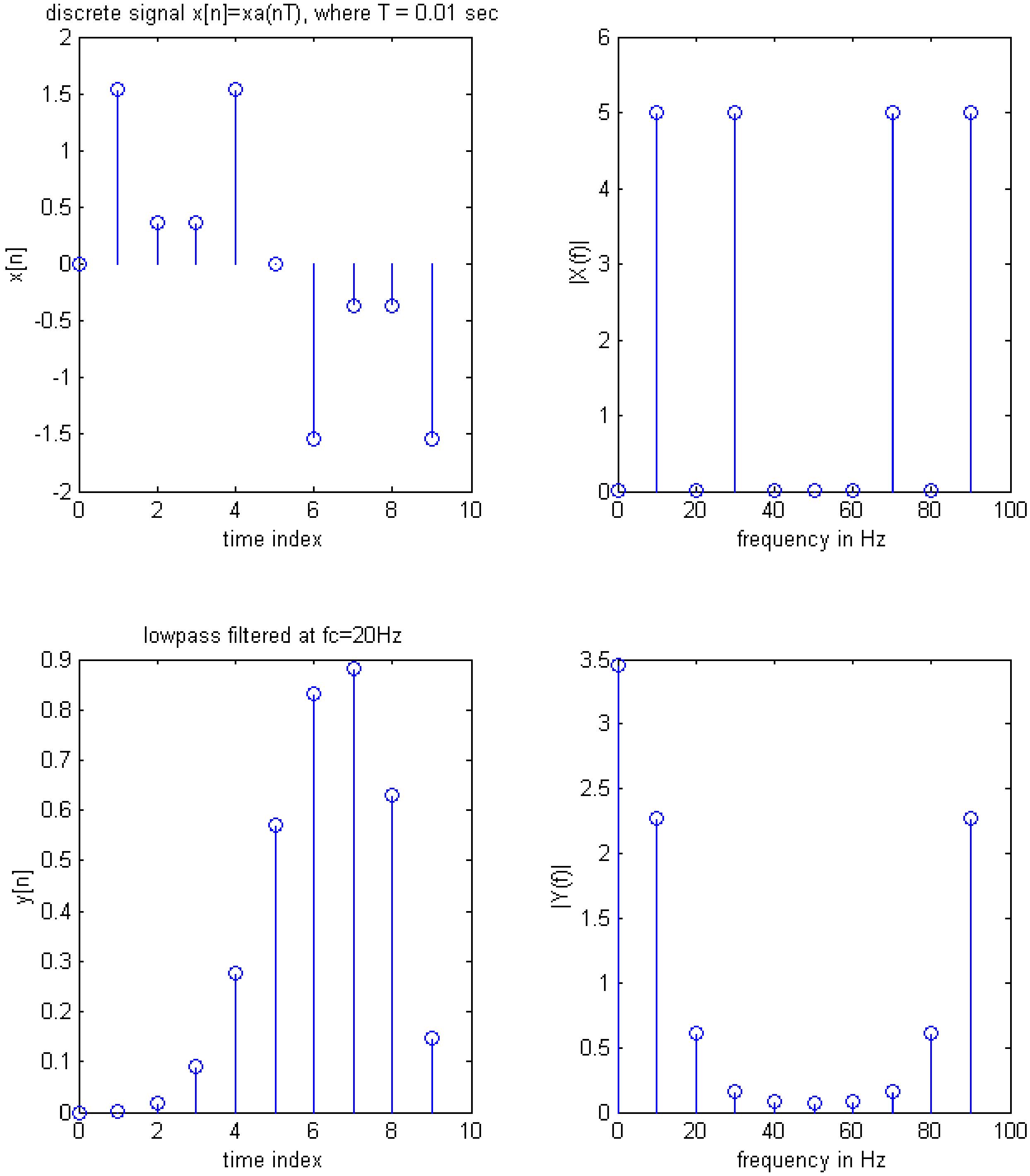

discrete signal x[n]=xa(nT), where T = 0.01 sec
x[n]
time index
|X(f)|
frequency in Hz
lowpass filtered at fc=20Hz
y[n]
time index
|Y(f)|
frequency in Hz

Fig. 6-1-2

程式 Ex_6_1.m

```
% Butterworth lowpass filter
[b,a] = butter(9,0.4,'low');       % cut-off freq. = 0.4 pi = 20 Hz
freqz(b,a,200,100);
pause;

% signal x
f1=10;        % 10 Hz sine wave
f2=30;        % 30 Hz sine wave
T=0.01;       % sampling freq. = 100 Hz
N=10;
n=0:1:N-1;
x=sin(2*pi*f1*n*T)+sin(2*pi*f2*n*T);
subplot(2,2,1); stem(n,x);
xlabel('time index'); ylabel('x[n]');
title('discrete signal x[n]=xa(nT), where T = 0.01 sec');

% DFT of x
f=n/T/N;
subplot(2,2,2); stem(f,abs(fft(x)));
xlabel('frequency in Hz'); ylabel('|X(f)|');

% lowpass filtering
y=filter(b,a,x);
subplot(2,2,3); stem(n,y);
xlabel('time index'); ylabel('y[n]');
title('lowpass filtered at fc=20Hz');

% DFT of y
f=n/T/N;
subplot(2,2,4); stem(f,abs(fft(y)));
xlabel('frequency in Hz'); ylabel('|Y(f)|');
```

[範例 6-2] 利用 Matlab 函式設計一 Chebyshev 低通濾波器，cut-off frequency 為 0.8π rad/sec，其中若取樣頻率為 1000 Hz，則 cut-off frequency 為 400 Hz。將一個具有 10Hz 與 300Hz 頻率之訊號通過此低通濾波器，並進行 decimation by 2，則由於取樣頻率變為 500Hz，其不足訊號最高頻(300Hz)的兩倍，因此發生 aliasing，參考 Fig. 6-2-2 (b)。若將 Chebyshev 低通濾波器之 cut-off frequency 改為 0.4π rad/sec (200 Hz)，則原訊號之 300Hz 成份將被濾除，經 decimation by 2 之後不致發生 aliasing，參考 Fig. 6-2-3 (b)。

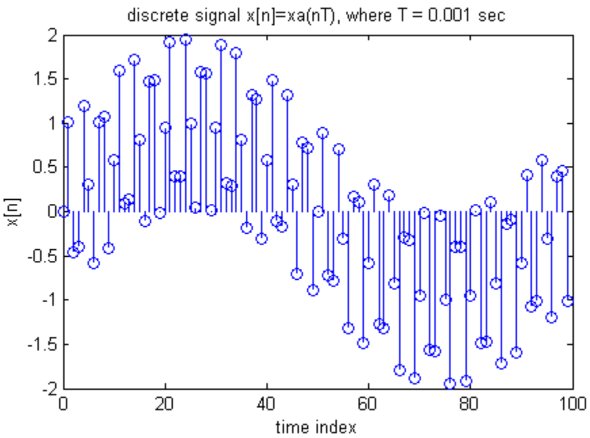


Fig. 6-2-1(a)

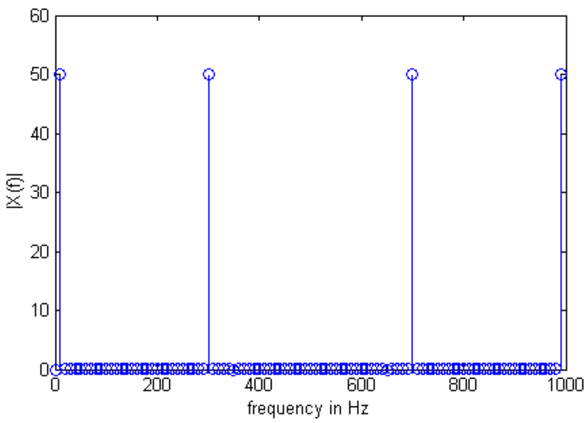


Fig. 6-2-1(b)

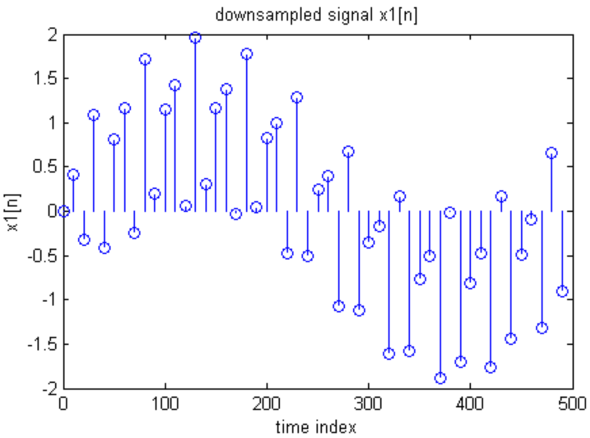


Fig. 6-2-2(a)

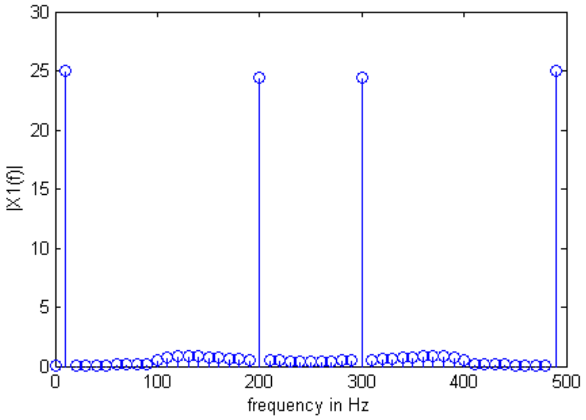


Fig. 6-2-2(b)

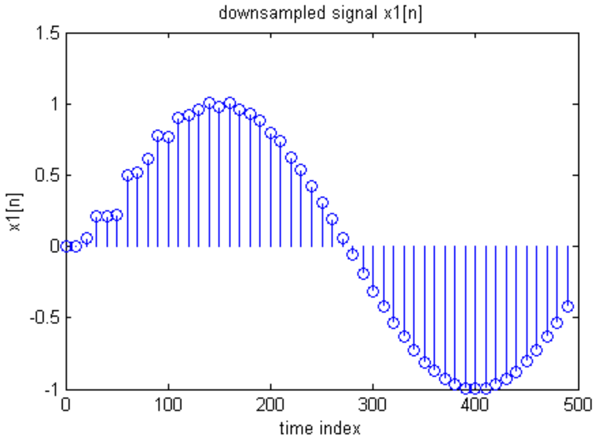


Fig. 6-2-3(a)

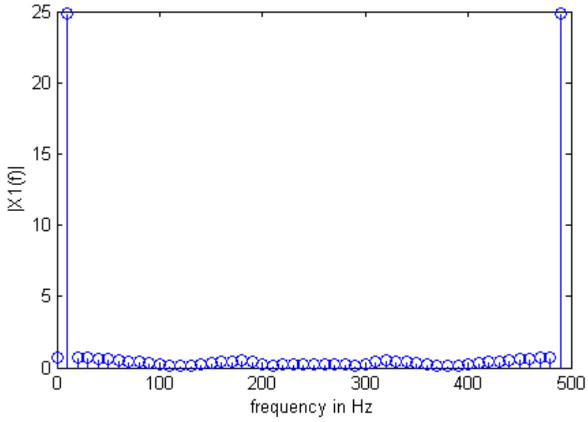
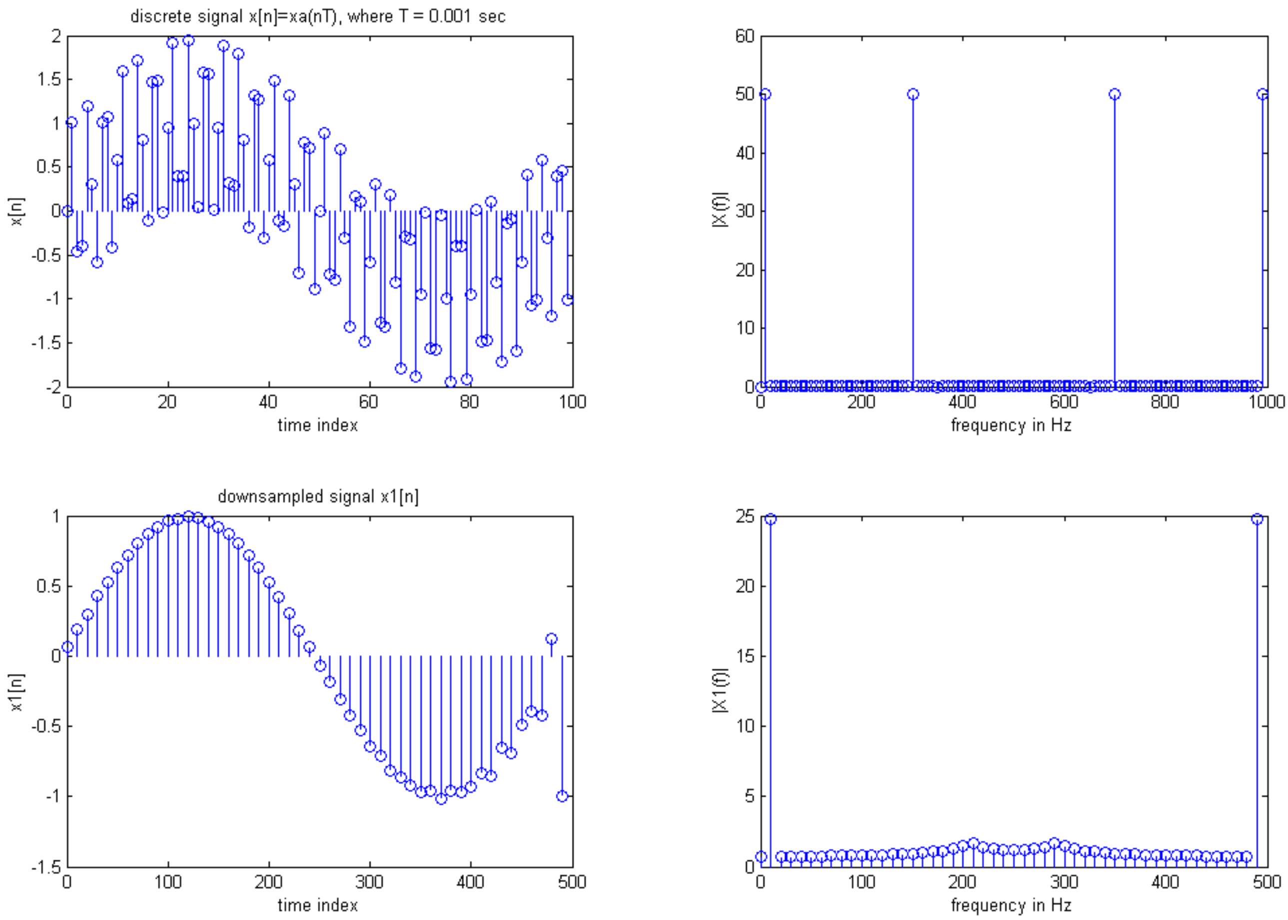


Fig. 6-2-3(b)

程式 P6_2.m

```
% Chebyshev lowpass filter
[b,a] = cheby1(9,0.05,0.4);        % cut-off freq. = 0.8 pi = 400 Hz

% signal x
f1=10;          % 10 Hz sine wave
f2=300;          % 300 Hz sine wave
T=0.001;        % sampling freq. = 1000 Hz
N=100;
n=0:1:N-1;
x=sin(2*pi*f1*n*T)+sin(2*pi*f2*n*T);
subplot(2,2,1); stem(n,x);
xlabel('time index'); ylabel('x[n]');
title('discrete signal x[n]=xa(nT), where T = 0.001 sec');

% DFT of x
f=n/T/N;
subplot(2,2,2); stem(f,abs(fft(x)));
xlabel('frequency in Hz'); ylabel('|X(f)|');

% lowpass filtering & Decimation & DFT
y=filter(b,a,x);
z=downsample(y,2);
n2=0:1:N/2-1;
f=n2/(2*T)/(N/2);
subplot(2,2,3); stem(f,z);
xlabel('time index'); ylabel('x1[n]');
title('downsampled signal x1[n]');
subplot(2,2,4); stem(f,abs(fft(z)));
xlabel('frequency in Hz'); ylabel('|X1(f)|');
```

範例 6-2 也可以用 Matlab 函式 decimate()來完成，參考程式 Ex_6_3.m



程式 Ex_6_3.m

```
% downsampling & DFT
z=decimate(x,2);
n2=0:1:N/2-1;
f=n2/(2*T)/(N/2);
subplot(2,2,3); stem(f,z);
xlabel('time index'); ylabel('x1[n]');
title('downsampled signal x1[n]');
subplot(2,2,4); stem(f,abs(fft(z)));
xlabel('frequency in Hz'); ylabel('|X1(f)|');
```

****

[練習 6-1] 利用 Matlab 函式 upsample.m 並自行設計一 Chebyshev 低通濾波器，將範例 6-2 之訊號進行升取樣兩倍。繪出升取樣後訊號之波形與其 DFT。

****

Part II:

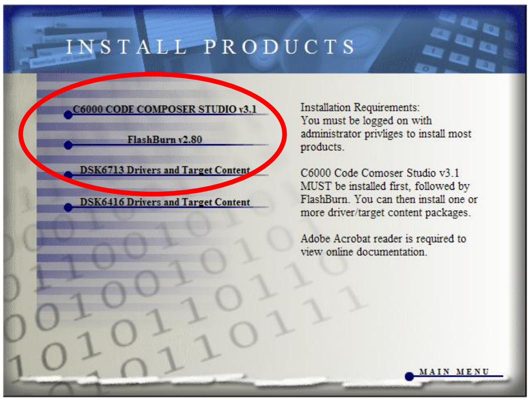
C Programming on TMS320C6713 DSK

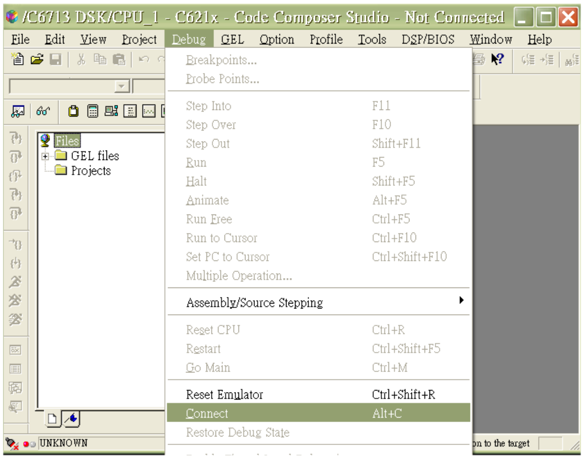
實驗一：熟悉 CCS 與 DSK6713

[工作一]

(1) Setup DSK6713

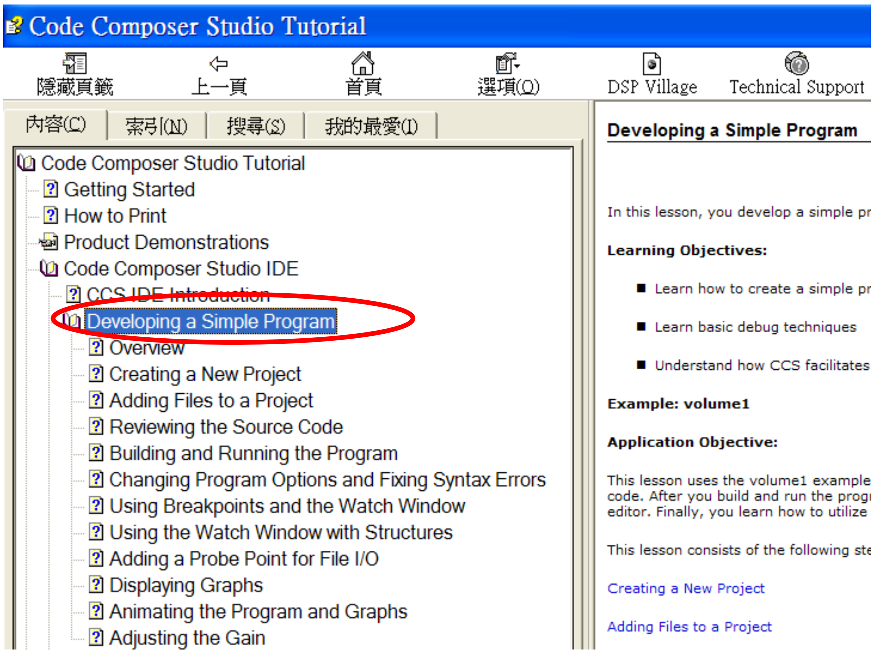
(2) Install Code Composer Studio





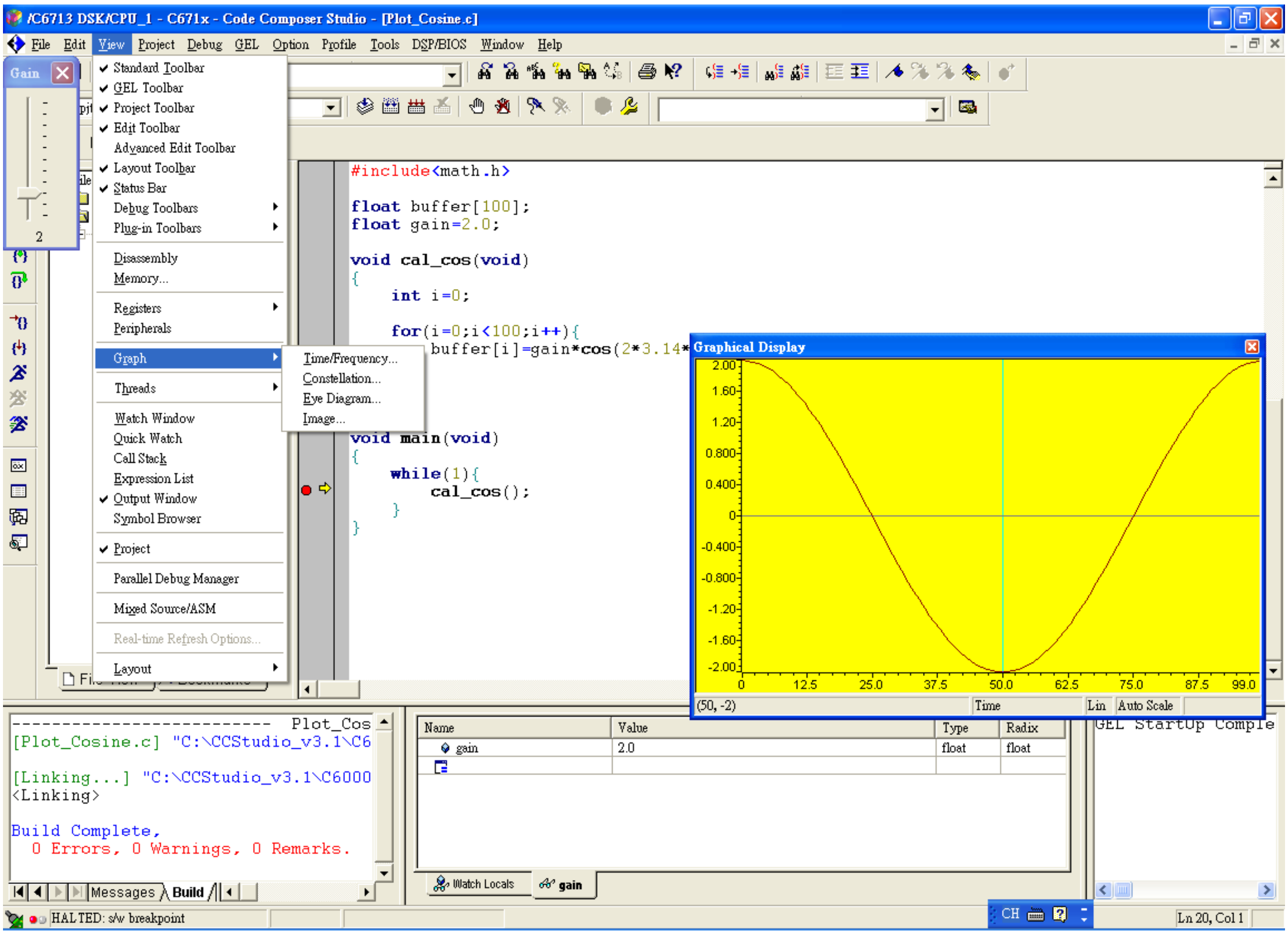
(3) Read CCS Tutorial

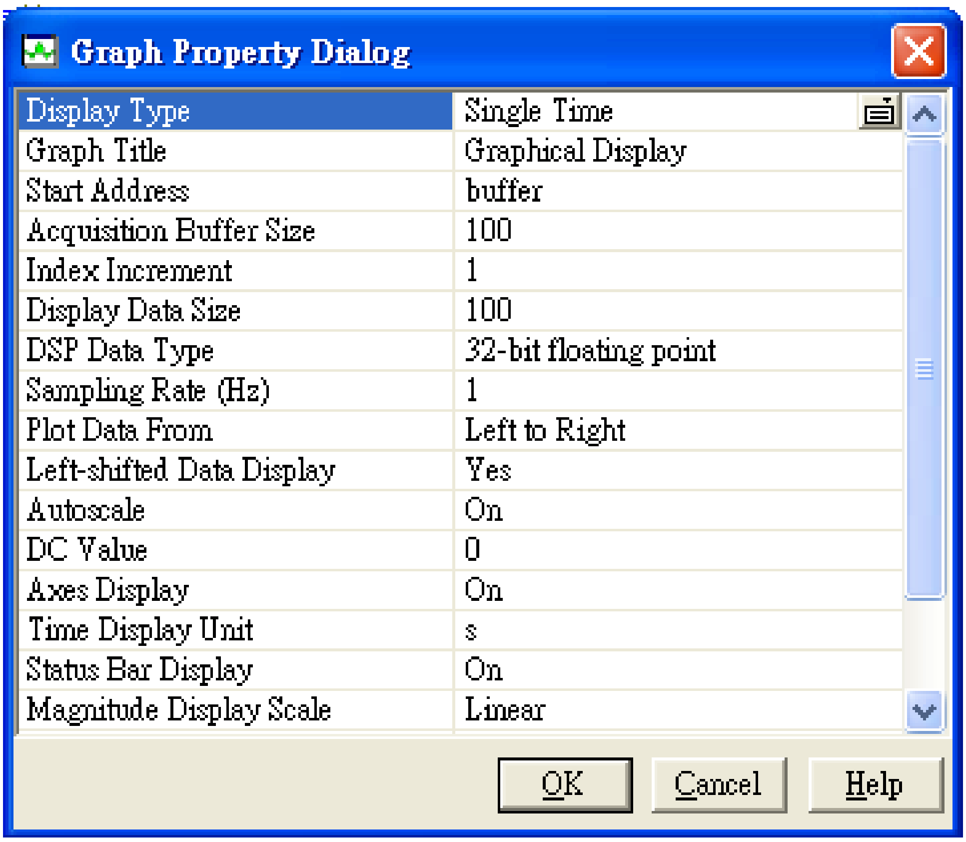
(4) Try CCS Tutorial's examples



[工作二]

撰寫一程式，並利用 CCS 中的 Graph 畫出 cosine 圖





實驗二：DSK6713 記憶體的配置

[工作一]

撰寫一段計算離散傅利葉轉換(discrete Fourier transform; DFT)之程式，並練習 CCS 操作。

N-point DFT 計算公式為 $X[k]=\sum_{n=0}^{N-1}x[n]e^{-j2\pi kn/N},\ k=0,1,\ldots,N-1$

在範例程式中，我們給定 $\mathbf{x}=\begin{bmatrix}x[0]\\x[1]\\x[2]\\x[3]\\x[4]\\x[5]\\x[6]\\x[7]\end{bmatrix}=\begin{bmatrix}10\\1\\2\\3\\4\\3\\2\\1\end{bmatrix}$，其 8-point DFT 計算得 $\mathbf{X}=\begin{bmatrix}X[0]\\X[1]\\X[2]\\X[3]\\X[4]\\X[5]\\X[6]\\X[7]\end{bmatrix}=\begin{bmatrix}26.0\\3.17\\10.0\\8.83\\10.0\\8.83\\10.0\\3.17\end{bmatrix}$。

試改寫程式為：給定 $x[n]=(n \bmod 5)$，其中 $0<n<1000$，計算 1000-point DFT。

Note:

(1) “.prj”專案中需要加入 rts6700.lib (little endian)或 rts6700e.lib (big endian)。

(2) 改變".cmd"檔案中的記憶體安排，例如將程式放在 internal memory 而資料放在 external memory、或將程式與資料全放在 internal memory 等，觀察執行結果。

[工作二]

參考 CCS Help 中有關 `DSPF_dp_cfftr2()`的說明，將範例程式中的 DFT 改成 FFT。

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	192K	0000 0000 – 0002 FFFF
Internal RAM/Cache	64K	0003 0000 – 0003 FFFF
Reserved	24M – 256K	0004 0000 – 017F FFFF
External Memory Interface (EMIF) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	128K	0184 0000 – 0185 FFFF
Reserved	128K	0186 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	512	019C 0000 – 019C 01FF
Device Configuration Registers	4	019C 0200 – 019C 0203
Reserved	256K – 516	019C 0204 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
Reserved	768K	01A4 0000 – 01AF FFFF
GPIO Registers	16K	01B0 0000 – 01B0 3FFF
Reserved	240K	01B0 4000 – 01B3 FFFF
I2C0 Registers	16K	01B4 0000 – 01B4 3FFF
I2C1 Registers	16K	01B4 4000 – 01B4 7FFF
Reserved	16K	01B4 8000 – 01B4 BFFF
McASP0 Registers	16K	01B4 C000 – 01B4 FFFF
McASP1 Registers	16K	01B5 0000 – 01B5 3FFF
Reserved	160K	01B5 4000 – 01B7 BFFF
PLL Registers	8K	01B7 C000 – 01B7 DFFF
Reserved	264K	01B7 E000 – 01BB FFFF
Emulation Registers	256K	01BC 0000 – 01BF FFFF
Reserved	4M	01C0 0000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	16M – 52	0200 0034 – 02FF FFFF
Reserved	720M	0300 0000 – 2FFF FFFF
McBSP0 Data Port	64M	3000 0000 – 33FF FFFF
McBSP1 Data Port	64M	3400 0000 – 37FF FFFF
Reserved	64M	3800 0000 – 3BFF FFFF
McASP0 Data Port	1M	3C00 0000 – 3C0F FFFF
McASP1 Data Port	1M	3C10 0000 – 3C1F FFFF
Reserved	1G + 62M	3C20 0000 – 7FFF FFFF
EMIF CE0†	256M	8000 0000 – 8FFF FFFF
EMIF CE1†	256M	9000 0000 – 9FFF FFFF
EMIF CE2†	256M	A000 0000 – AFFF FFFF
EMIF CE3†	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

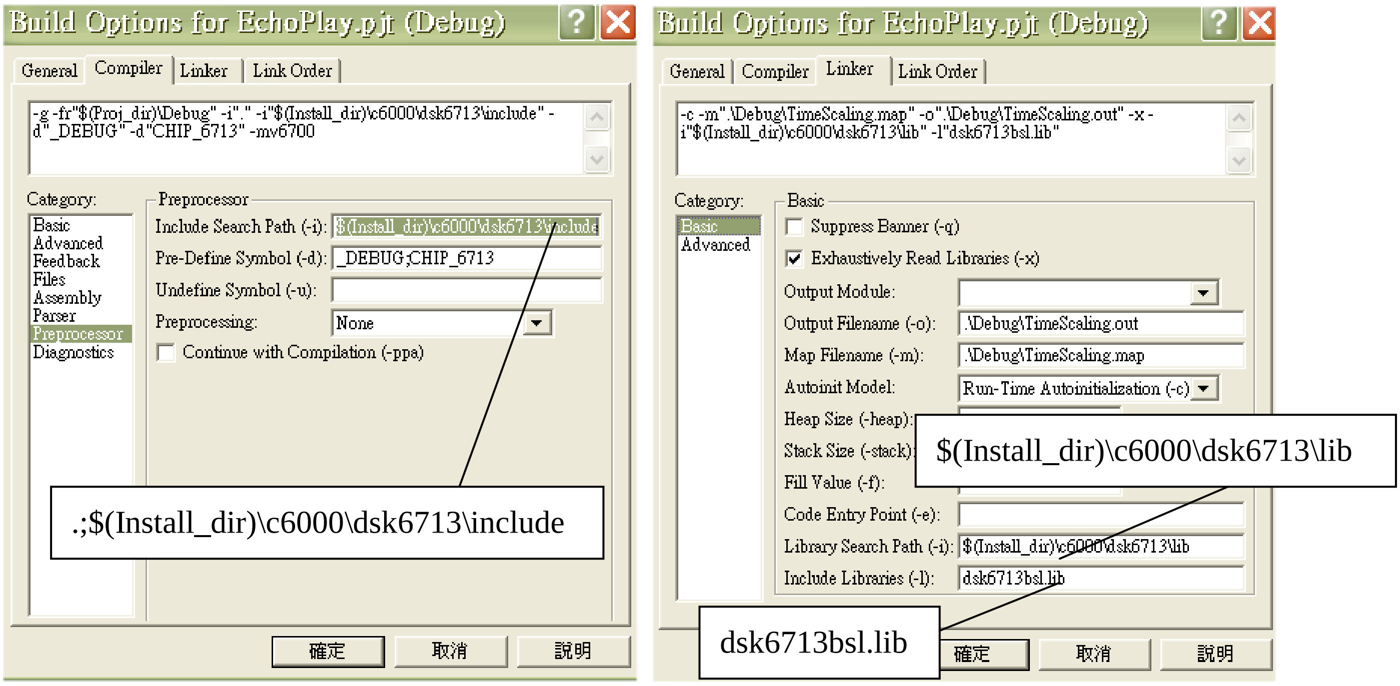
† The number of EMIF address pins (EA[21:2]) limits the maximum addressable memory (SDRAM) to 128MB per CE space.

實驗三 Board Support Library (BSL) 與音訊處理

[工作一] 練習利用 BSL 操作 AIC23

DSK 上除了 DSP 核心以外還包括許多的晶片，本實驗將練習使用其中一個做為類比/數位轉換以及音訊處理之晶片—AIC23。該晶片是透過 McBSP 與 DSP 核心進行溝通，我們需要設定 McBSP 並搭配 EMIF 來操作 AIC23。但考慮直接設定 McBSP 的步驟複雜，我們將採用 DSK 所提供的 library (BSL)則較容易進行。Lab3-1 的範例程式中產生一個 1KHz 的 sine wave，透過 McBSP 送至 AIC23 的輸出端，執行後可以聽到持續五秒的單音符，其中 sine wave 是事先計算好並存於一陣列中。由於 AIC23 codec 的 D/A 是以 48KHz 進行(by default)，因此若 sine wave 陣列內含 48 個取樣值則相當於 1/1000 秒的播放時間，重複播放 5000 次 sine wave 陣列即為五秒。注意使用 BSL 必須在<Build Options>中的 Preprocessor 與 Link 欄位加入某些參數：另外注意範例程式的.prj 中並沒有 include rts6700.lib 也沒有自行定義的.cmd，這是因為其中包含了一個.cdb。

試修改 Lab3-1 的範例程式使得輸出結果可聽到七個音符：Do, Re, Mi, Fa, Sol, La, Si，其中各音符之基頻(fundamental frequency)如 Fig. 5-3 所示。



[工作二] 即時音訊處理

Lab3-2 的範例程式利用"ping-pong buffering"方式將 Line In 訊號不斷送入 DSP 內部，同時經由 Line Out 輸出至喇叭，達到即時的錄放音效果。試將程式修改以產生回音效果，其中回音的產生過程如下圖所示，主要是訊號經過延遲後的疊加結果。Hint: 在 copyData( )中加入你的程式。