The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

Digital Signal Processing (EIE413/EIE4413)
Lab 1: Spectrum Analysis and Filtering

Name: YANG Tianyu
Student ID: 13102841D
Submission date: 4-3-2016

Purpose

This lab explores the implementation of various fundamental concepts in signal processing using Matlab. The lab is divided into two parts: (i) DFT and FFT; and (ii) LTI Systems and Filtering. They serve as the supplement to the discussion in class.

Things to do

For each question in this lab sheet, give your answer right under the question. Submit this lab sheet (with your answers) through the Blackboard to your lecturer before the deadline. In addition, show the result of Q.2.5 to your tutor during the laboratory session.

Equipment

PC with Windows 7 or above
PC with MATLAB 7 or above

Part 1. DFT and FFT

The Discrete Fourier Transform (DFT) allows us to evaluate the Fourier Transform of a signal at N evenly spaced frequencies, where N is the number of data in that signal. Mathematically, the DFT of a signal $x[n]$, where $n = 0, 1, ..., N-1$, is defined as follows:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}$$

where $k = 0, 1, ..., N-1$. Basically, the number of complex multiplications and additions required by an N-point DFT is N^2 and $N(N-1)$, respectively. It becomes a heavy computational burden when N is large. The FFT is then devised to reduce the computational complexity, however it has the limitation that N

must be a power 2. In Matlab, the method of how a DFT is computed is determined by the value of *N*. If *N* is a power of 2, FFT will be used. If *N* is, for instance, a prime number, the traditional DFT will be used. We can examine their computational complexity indirectly by using the commands *tic-toc* in Matlab. These commands report the elapse CPU time between the execution of the commend *tic* and the command *toc*. By using *tic-toc*, we can measure the time taken by FFT for different N as follows:

```
x = rand(1,65536);      % Define a random vector x with 65536 elements
y = rand(1,65213);      % Define a random vector y with 65213 elements
tic                     % Start to count
fft(x);
t1 = toc                % Record the finish time

tic                     % Start to count
fft(y);
t2 = toc                % Record the finish time
```

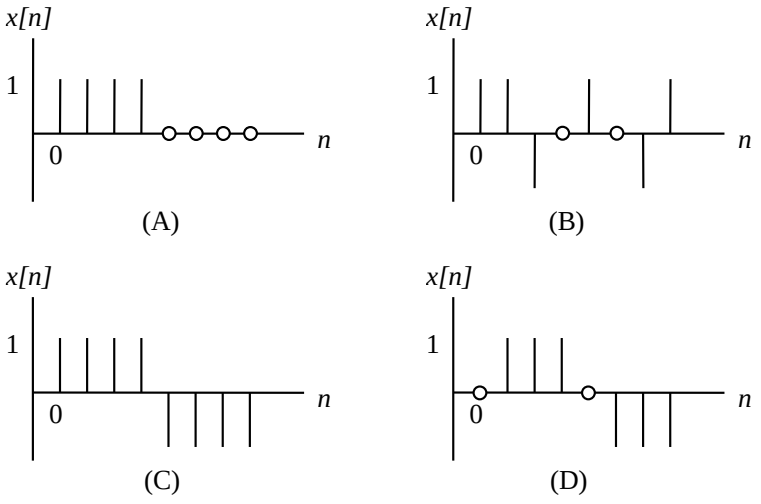
Q.1.1 Try the program above by putting it into a M-file. Show the difference in time taken when calculating the fft of `x` and `y`. (Hint: There can be error when using the commands `tic-toc` for the first time. Try to run the program 2 to 3 times and take the last result as the final result.)

(1)
t1 = 0.0207

t2 = 0.0259
(2)
t1 = 0.0015

t2 = 0.0117

Q.1.2 Find the 8-point FFTs of each of the four signals as shown below:



a) Ignore the slight computer round-off error, which of these FFTs are purely real?

(B) is purely real.

b) Which of these four FFTs are purely imaginary?

(D) is purely imaginary.

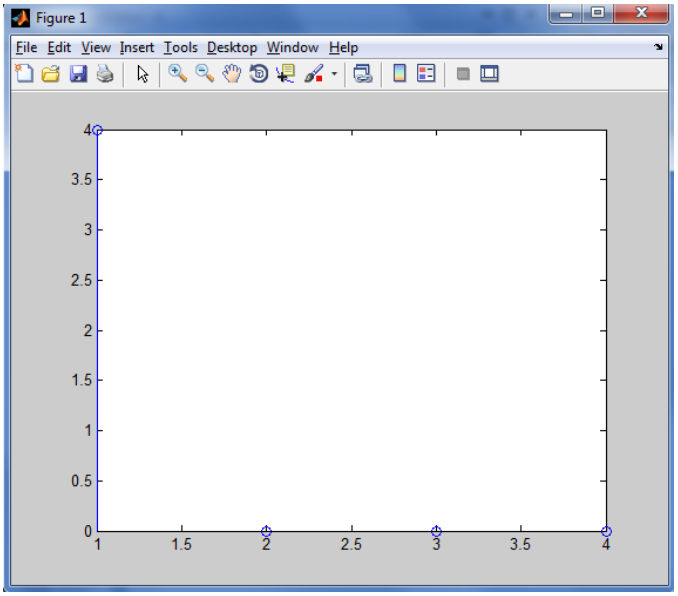
c) Can you derive the rules by which a real signal has a purely real FFT or a purely imaginary FFT? (Find the answer in the course notes.)

If x[n] is even symmetry, then it has a purely real FFT.
If x[n] is odd symmetry, then it has a purely imaginary FFT.

Q.1.3 Let $x[n]$ be defined as follows:

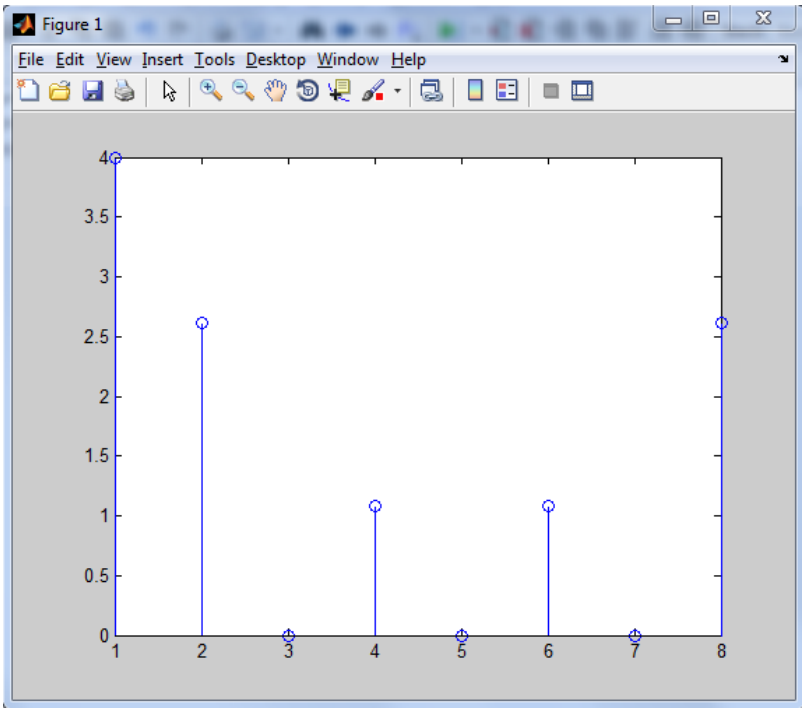
$$x[n] = \begin{cases} 1 & for \quad n = 0,1,2,3 \\ 0 & otherwise \end{cases}$$

a) Compute and plot the magnitude of the spectrum of $x[n]$ by using a 4-point DFT (i.e. using only the first 4 ones of $x[n]$).
Hint: the magnitude of a complex number can be found by the command *abs*.

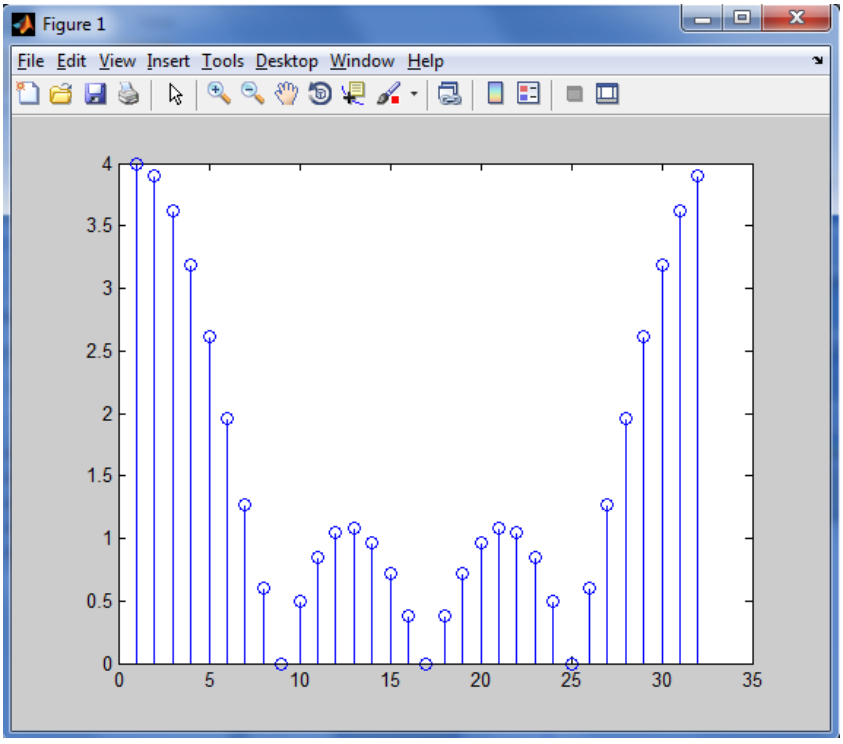


b) Compute and plot the magnitude of the spectrum of $x[n]$ by using 8, 32, 128 and 1024-point DFTs (by padding an appropriate number of zeros). Note how the frequencies are "filled in" as the size of the DFT is increased. Why the spectrum in (a) is so different from the spectrum when 128-point DFT is used?

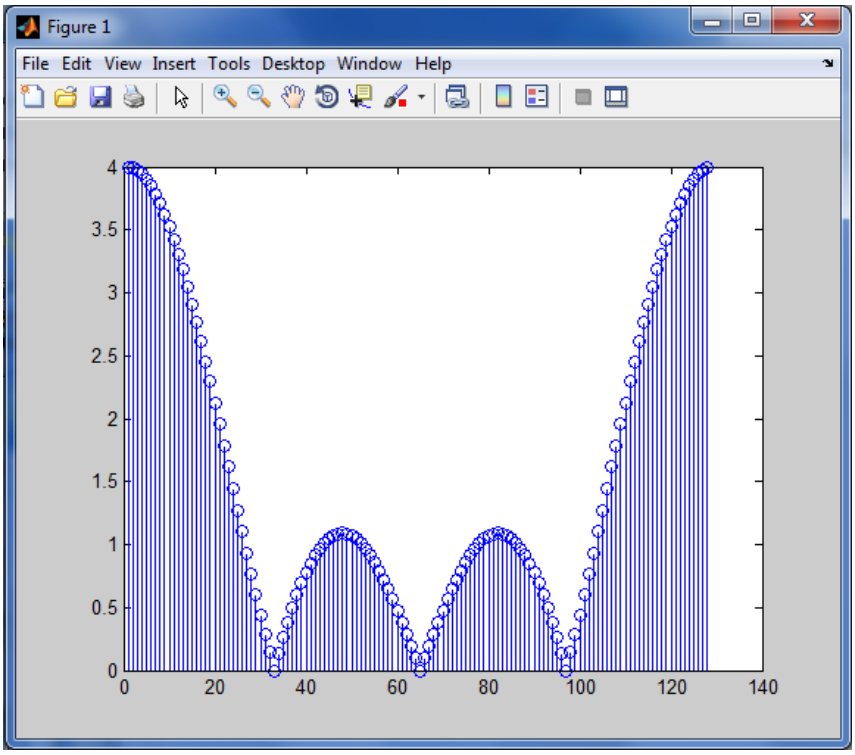
8-point DFTs



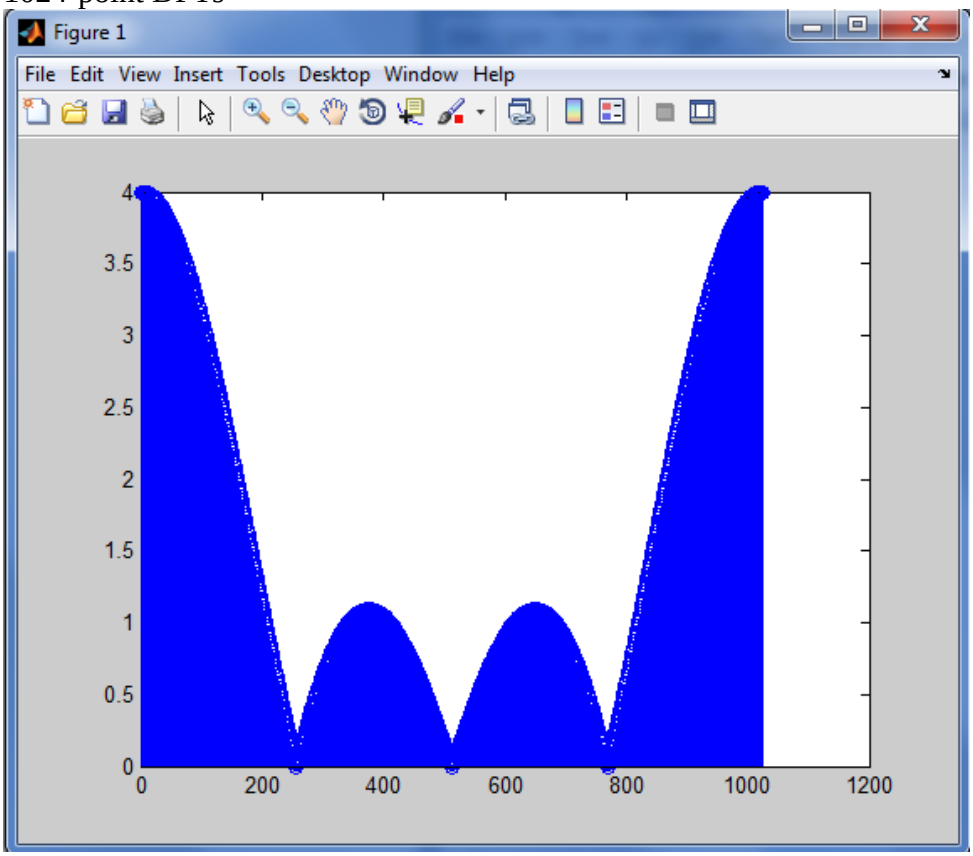
32-point DFTs



128-point DFTs



1024-point DFTs



Answer: Because there is more samples used to do DFT, so the result is more accurate. The graph is more similar to that in Fourier Transform of continuous signals.

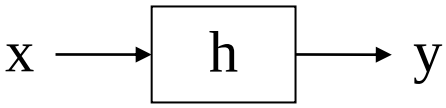
c) If $X_p(\hat{\omega})$ the Fourier transform of x is [n] and if the sampling frequency is 12,800Hz, find $X_p(\hat{\omega})$ at 900 Hz.

Using 128-point DFTs

900Hz/(12800Hz/128)=9

X(8)= 3.013669746062924 - 2.013669746062924i

Part 2. LTI systems and Filtering



Discrete-time linear time-invariant (LTI) systems are usually described by using the linear convolution operator, with input *x*, output *y* and the "impulse response" *h*,

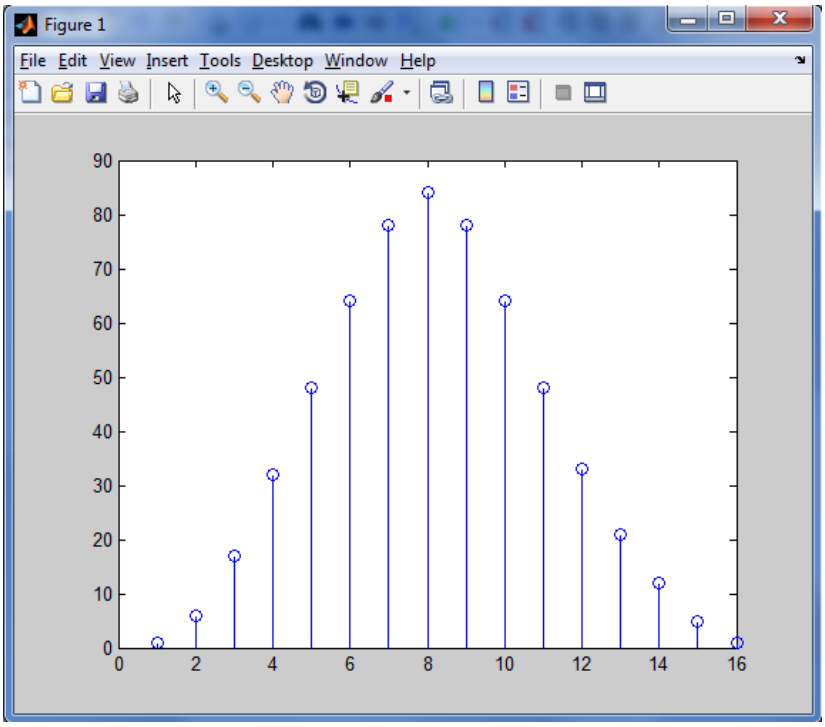
$$y[n]=\sum_{m=0}^{M} x[m]h[n-m] = \sum_{m=0}^{M} x[n-m]h[m].$$

We usually denote the equation above as $y[n] = x[n]*h[n]$. In Matlab, linear convolution can be computed by the command *conv*.

Q.2.1 Let $x[n]$ and $h[n]$ be defined as follows:

$$x[n]=\begin{bmatrix}1 & 2 & 3 & 4 & 5 & 6 & 5 & 4 & 3 & 2 & 1 & 1\end{bmatrix}$$
$$h[n]=\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

a. Calculate the output of the LTI system $y[n]$ using *conv*.



b. What is the length of $y[n]$?

According to the data, the length of y[n] is 16.

There is a very important concept for the LTI system that linear convolution in time domain is equivalent to multiplication in frequency domain. Let us verify it using x, h, and y as discussed in Q.2.1.

Q.2.2 Compute the spectrums of $x[n]$, $h[n]$ and $y[n]$ by using a 16-point DFT (hence zero padding to x and h is required).

a. What is the mean square difference between the magnitude of $X.*H$ and Y? (where X, H and Y are the length-16 DFTs of x, h and y, respectively.). Can it be concluded that $X.*H = Y$?

The mean square difference is 3.944304526105059e-29.

b. Can we use a shorter length DFT? Why?

No, because the shortest length DFT is L+P-1=16 and it need to be the power of 2.

c. How about if we use a longer length DFT (say 32-point DFT, hence even y needs to be zero padded)? Do we still have $X.*H = Y$?

Yes, because the mean square difference is 3.944304526105059e-29 and it is small enough to neglected. So we get X.*H = Y.

Now let's see how we can make use of FFT to speed up the computation of linear convolutions.

Q.2.3 Generate a random sequence ***x***, an FIR filter ***h*** with random coefficients, and compute their convolution as follows:

```
x = randn(1,102500);
h = randn(1,1024);
y = conv(x,h);
```

a. Use the command `tic-toc` to give the computation time required for the above operations.

The time required is 0.043921726298009s.

b. The above linear convolution can also be implemented based on the convolution property of the DFT using the following commands:

```
X = fft(x,2^20);  % Compute the DFT of x by using a 2^20-point FFT
H = fft(h,2^20);
y2 = ifft(X.H);
```

i) What is the mean square difference between `y` and `y2`?

The mean square difference is 1.523648033225253e-28.

ii) Use the command `tic-toc` to give the computation time required for the above operations. Is there any saving in computation time?

The time required is 0.186799696583607s. There is not any saving in computation time.

c. Open a new M-file and write a program that implements the above linear convolution using the convolution by section method as well as the convolution property of the DFT. You may want to follow the procedure below:

- Divide the sequence ***x*** into 100 segments, each with 1025 data.
- Compute the DFT of ***h*** (i.e. ***H***) by using a 2048-point FFT
- For each segment $\boldsymbol{x_r}$, compute its DFT (i.e. $\boldsymbol{X_r}$) by using a 2048-point FFT. Then dot-multiply with ***H*** to generate $\boldsymbol{Y_r}$.
- Compute the inverse FFT of $\boldsymbol{Y_r}$ to generate $\boldsymbol{y_r}$.
- Sum all $\boldsymbol{y_r}$ to get ***y3*** (note that each $\boldsymbol{y_r}$ starts at different point in ***y3***).

i) Show your program in the box below.

```
x = randn(1,102500);
h = randn(1,1024);

i=1;
  while i<=100
        m=[x(i:i+1025),zeros(1,1023)];
        n=[h,zeros(1,1024)];
        X = fft(m,2048);
        H = fft(n,2048);
        y1 = ifft(X.*H);
```

```
        y(i:2048)=y(i:2048)+y1;
        i=i+1025;
end
```

ii) What is the mean square difference between `y` and `y3`?

The mean square difference is 2.286616348614650e+03.

iii) Use the command `tic-toc` to give the computation time required for the above operations. Is there any saving in computation time?

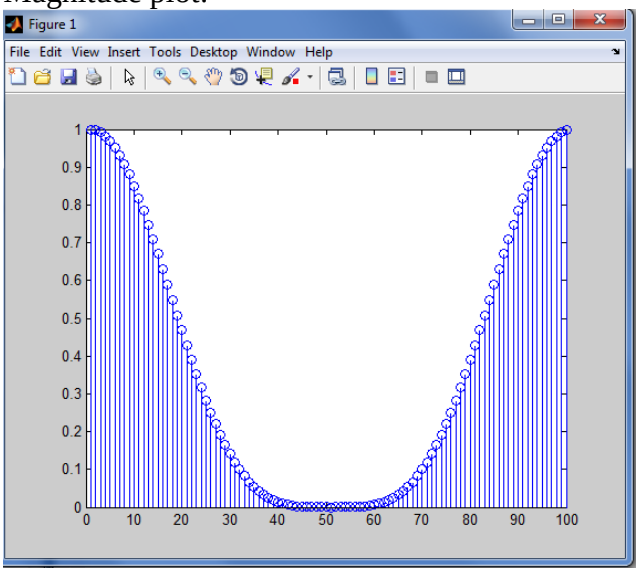
The time required is 1.934837918838832e-04 s. It saves time.

One of the examples of FIR filters is the running average filter. In the following we shall define a running average filter and study its frequency response.

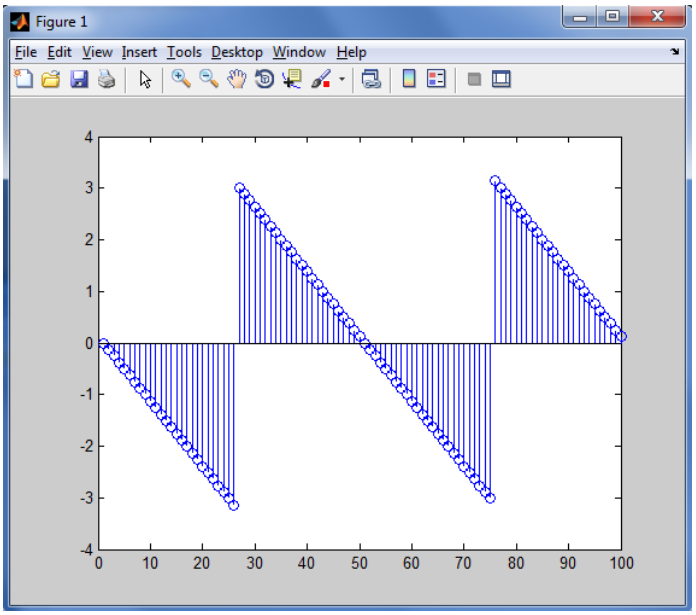
Q.2.4 Define an FIR filter ***h*** with coefficients {1, 4, 6, 4, 1}/16.

a. By using a 100-point DFT, compute and plot the frequency response for ***h*** (magnitude and phase plots).
(Hint: the phase angle of a complex number can be found by the command *angle*.)

Magnitude plot:



Phase plots:



b. If the sampling frequency is 8 kHz, estimate the bandwidth of the filter (check the definition of bandwidth in your course notes).

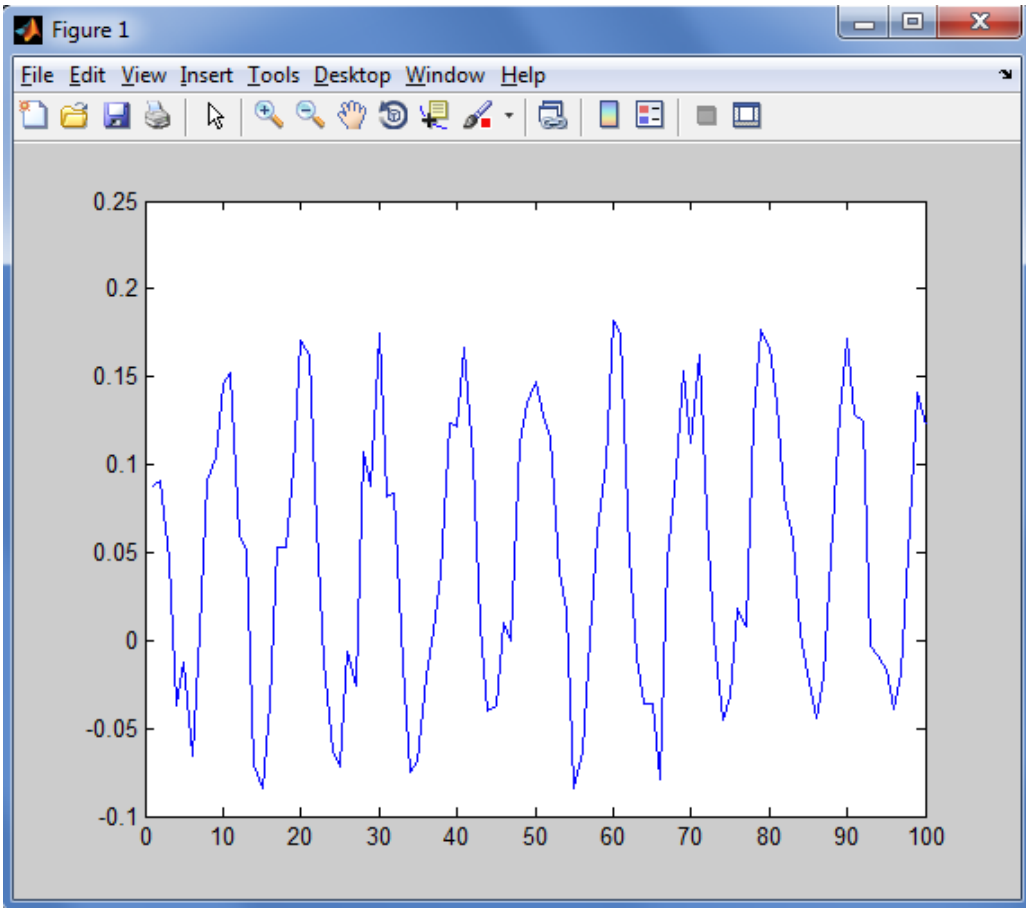
The amplitude of H(14) is approximately 0.71.
8000Hz/100*14=1120Hz

See the effect of filtering an irregular signal with a running average filter. An irregular signal can be generated by the following Matlab statements:

```
i = 0.001:0.001:0.1;                % Generate a vector of 100 elements
x = 0.1*rand(1,100);                % Generate a random signal of 100 elements
s = 0.1*cos(2*pi*i*100);            % Generate a cosine function at 100Hz
xx = x + s;                         % Add the random signal to the cosine function
```

Q.2.5 Generate an irregular signal by putting the above statements into an M-file.

a. Plot the signal generated by the above statements.



b. Feed such signal to the running average filter as discussed in Q.2.4. Note the result. **(Show the result to your tutor)**

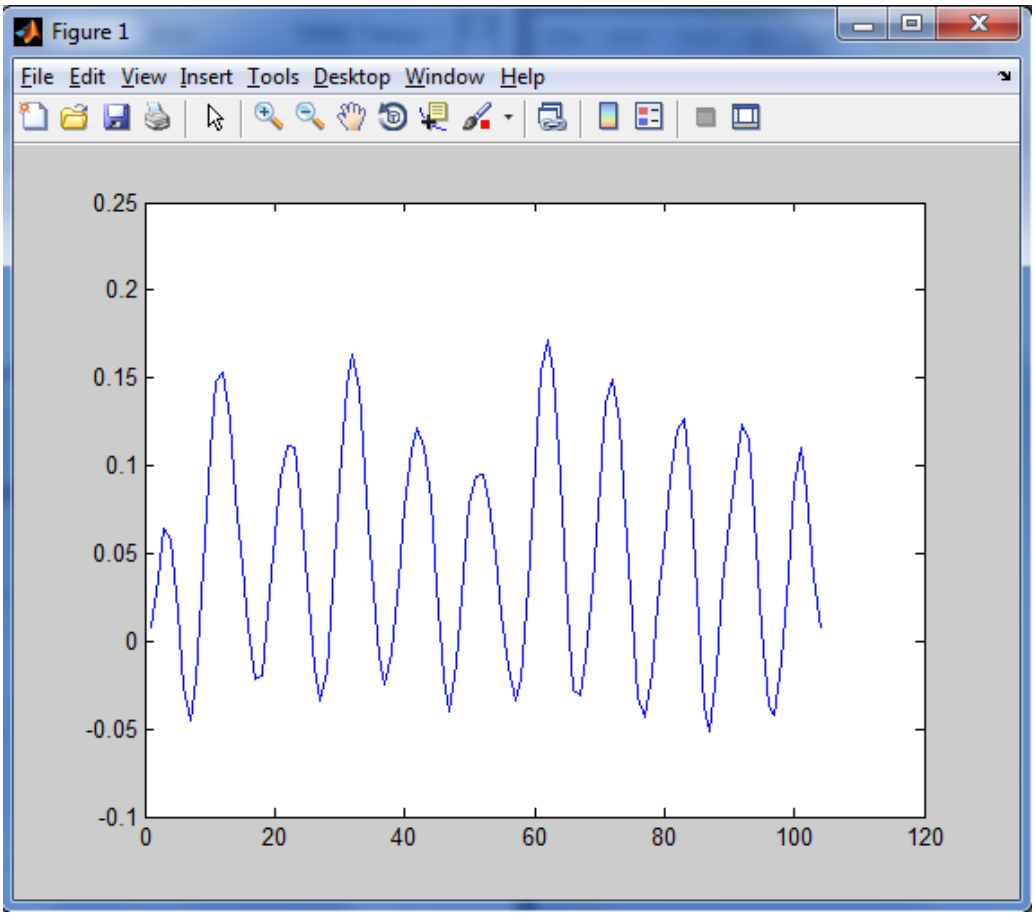


Figure 1
File Edit View Insert Tools Desktop Window Help
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
20
40
60
80
100
120

The End

Appendix A : The Function of MATLAB

A. General

help [instruction]	---	Displays help for [instruction]
help	---	Displays a list of available instructions
dir (or ls)	---	Lists directory
cd	---	Change directory
who (whos)	---	Lists current variables
format	---	Set output (display) format, e.g. format compact
; (after command)	---	Suppresses the display of result

B. Matrix

Row matrix : A = [1,2,3] or A = [1 2 3] Column matrix:
A = [1:1:3] or A = [1, 2, 3]' Rectangular
matrix: A = [1 2 3 4: 5 6 7 8]
To access an element: c = [A (row, column)]; e.g. A(2,3)
To access an row: c = [A (row,:)]; e.g. A(1,:) = the first row of A To access an column: c = [A (:,column)];
Identity matrix: b = eye (n), where n is an integer
Ones matrix: x = ones(m,n) Zeros matrix:
y = zeros(m,n)

Length(x)	---	Shows the length of vector x
Size(A)	---	Shows the size of matrix A in (row, column)

C. Special Variable and Constants

Ans	---	Most recent answer
eps	---	Floating point relative accuracy i, j ---
	Imaginary unit; e.g. c = 2 + 5I inf ---	Infinity
pi	---	π
Nan	---	Not a number

D. Control Flow

if {condition} – else – end

if {condition} – elseif {condition}– else – end

for j = 1:N – end

while {condition} – end

E. Mathematical Operators

+	→ Plus	+	→ Unary plus
-	→ Minus	-	→ Unary minus
*	→ Matrix multiply	*.	→ Array multiply
\	→ Backslash or left matrix divide	/	→ Slash or right matrix divide
.\	→ Left array divide	./	→ Right array divide
^	→ Matrix power	.^	→ Array power

F. Functions

abs	---	Absolute value
angle	---	Phase angle
cos	---	cosine
sin	---	sine
imag	---	Complex imaginary part
real	---	Complex real part

G. Useful Functions

y = fft (x) or y = fft(x, n)	---	FFT of x ; n is number of point
[z, p, k] = buttap(n)	---	butterworth analog prototype
[num, den] = zp2tf (z, p, k)	---	zero-pole to transfer function
[bz, az] = impinvar (num, den, fs)	---	impulse invariant transformation
[zd, pd, kd] = bilinear (z, p, k, fs)	---	bilinear transformation
[h,w] = freqz (num, den, n)	---	Z-transform digital filter frequency response
w = hamming (n)	---	hamming window
stem	---	plot the signal in the impulse form

References

[1] S.K. Mitra, Digital Signal Processing, 3rd Edition, McGraw-Hill Education (Asia), 2009.
[2] J.G. Proakis and D.G. Manolakis, Digital Signal Processing: Principles, Algorithms and Applications, 4th Edition, Pearson International Edition, 2007.
[3] McClellan, Schafer and Yoder, *DSP FIRST: A Multimedia Approach*. Prentice Hall, Upper Saddle River, New Jersey, 1998 Prentice Hall.
[4] *Using Matlab*, The Math Works Inc.