# MATLAB Basics …

*and*

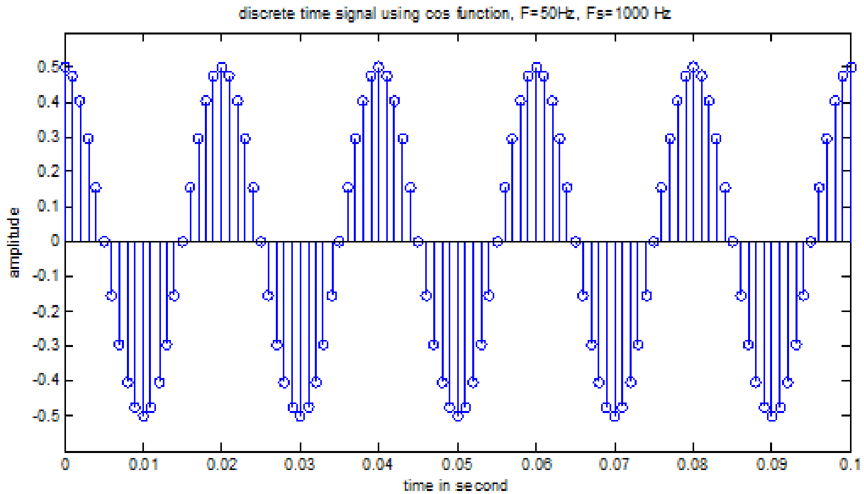
## Lab. Experiment 01

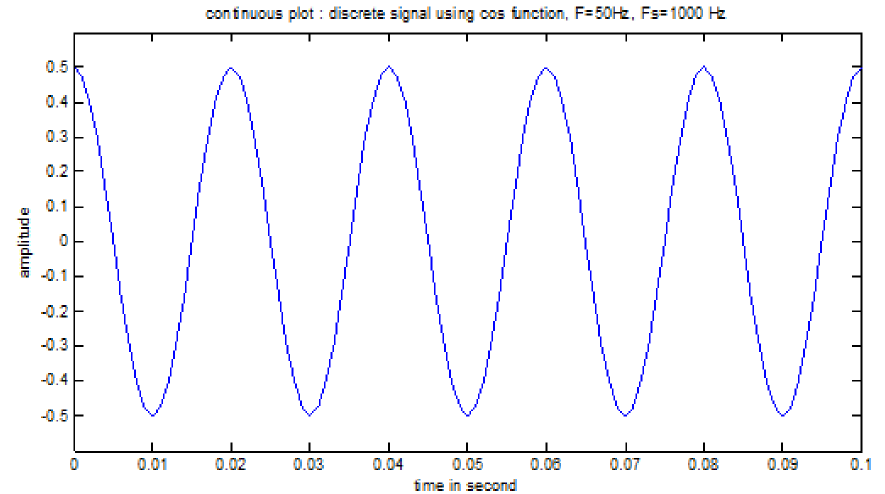
```
% Exp. 1
% Generation and plotting of discrete-time sinusoids

F=50; % frequency of the sinusoidal signal
A=0.5; % amplitude of sinusoid
Fs=1000; % sampling frequency
t = 0:1/Fs:0.1; %sampling between 0 to 0.1 second at interval of 1/Fs

% Sinusoidal signal is generated by using in built function cos or sin
s1 = A*cos(2*pi*F*t); % Generation of sinusoid using cos function
s2 = A*sin(2*pi*F*t); % Generation of sinusoid using sin function
```

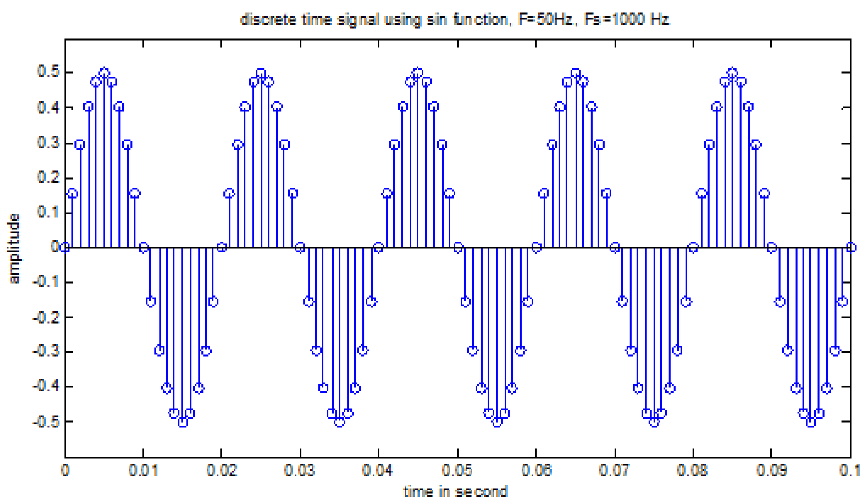
```
% s1 plotted as discrete time signal
subplot (221); % divides graphic window into 2x2 matrix, 1st selected
stem(t,s1);  % plot of discrete samples
axis ([0 0.1 -0.6 0.6]); % defining axis else in default mode
xlabel('time in second'); ylabel('amplitude');
title('discrete time signal using cos function, F=50Hz, Fs=1000 Hz')

% s1 plotted as if a continuous time signal through ‘plot’ function
subplot (222); % 2nd graphic window is selected
plot(t,s1);  % continuous plot of discreet samples
axis ([0 0.1 -0.6 0.6]); xlabel('time in second'); ylabel('amplitude');
title('continuous plot : discrete signal using cos function, F=50Hz, Fs=1000 Hz')
```

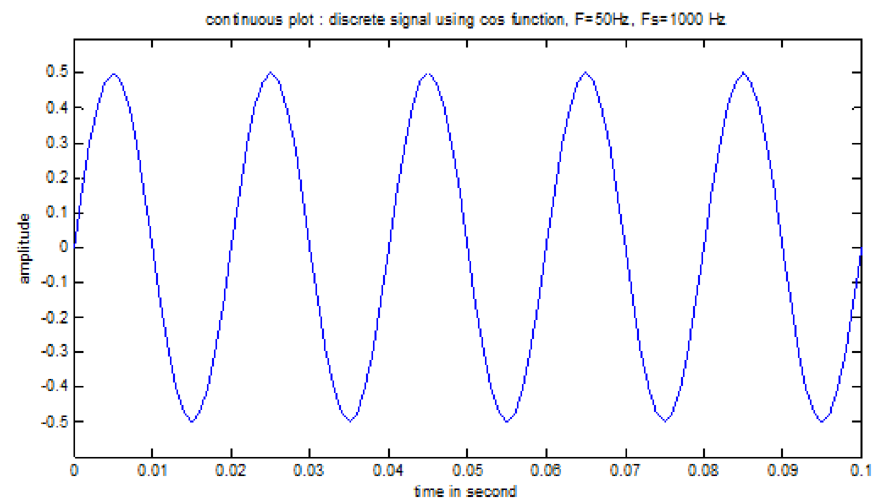
?

```
% s2 plotted as discrete time signal
subplot (223); % 3rd graphic window is selected
stem(t,s2);  % plot of discrete samples
axis ([0 0.1 -0.6 0.6]); xlabel('time in second'); ylabel('amplitude');
title('discrete time signal using sin function, F=50Hz, Fs=1000 Hz')

 % s2 plotted as if a continuous time signal
subplot (224); % 4th graphic window selected
plot(t,s2);  % continuous plot discreet samples
axis ([0 0.1 -0.6 0.6]); xlabel('time in second'); ylabel('amplitude');
title('continuous plot : discrete signal using sin function, F=50Hz, Fs=1000 Hz')
```


discrete time signal using cos function, F=50Hz, Fs=1000 Hz
amplitude
time in second


continuous plot : discrete signal using cos function, F=50Hz, Fs=1000 Hz
amplitude
time in second

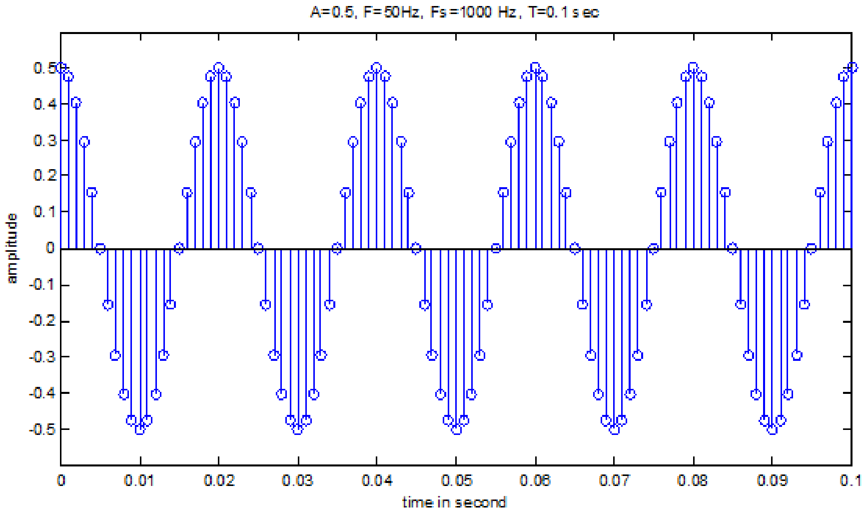

discrete time signal using sin function, F=50Hz, Fs=1000 Hz
amplitude
time in second

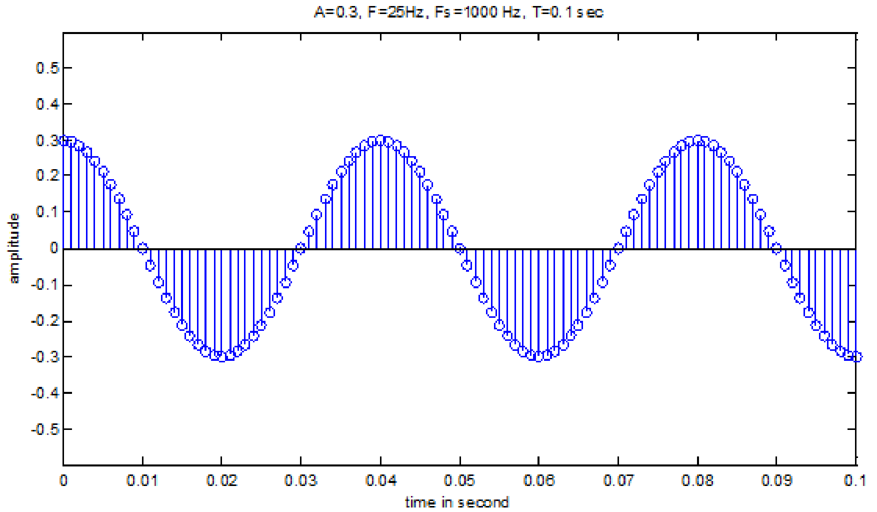

continuous plot : discrete signal using cos function, F=50Hz, Fs=1000 Hz
amplitude
time in second

```matlab
% Generation of function mysin.m which is to be called later in Exp2

function [s, t] = mysin(A, F, Fs, T)
% mysin as a function that takes amplitude (A), frequency of sinusoid (F),
% sampling frequency (Fs), duration of sampling (T) as input. This generates
% sinusoidal signal (s) and sampling time instants (t) at the output

% Generation of sine wave
 t = 0:1/Fs:T; %sampling instances between 0 to T second at interval of 1/Fs
 s = A*cos(2*pi*F*t); % Generation of sinusoid using cos function
 end
```

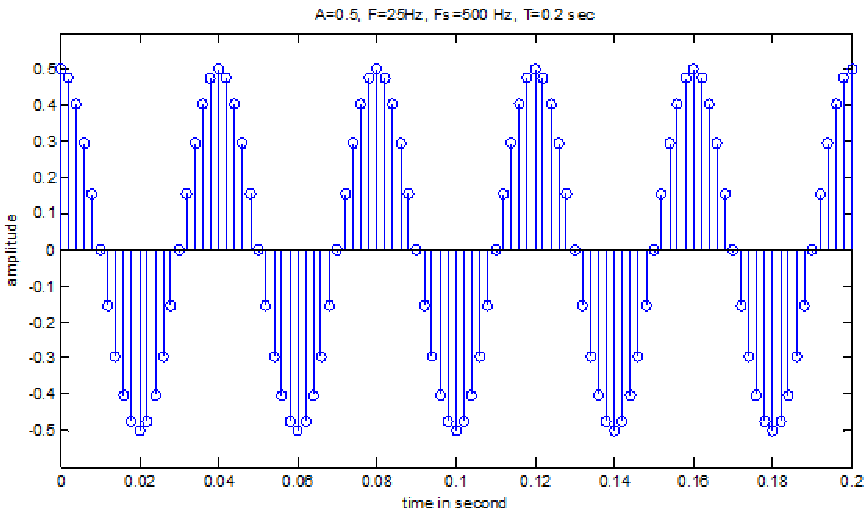
```
% Exp. 2

% Generate and plot sinusoidal signals through function calls mysin.m
[s1, t1] = mysin (0.5, 50, 1000, 0.1); % A=0.5, F=50. Fs=1000, T=0.1
subplot (221);  stem(t1, s1);  axis ([0 0.1 -0.6 0.6]);
xlabel('time in second'); ylabel('amplitude');
title('A=0.5, F=50Hz, Fs=1000 Hz, T=0.1 sec')

% Generating a sinusoid of frequency and amplitude different from s1
 [s2, t2] = mysin (0.3, 25, 1000, 0.1); % A=0.3, F=25. Fs=1000, T=0.1
 subplot (222);  stem(t2, s2);  axis ([0 0.1 -0.6 0.6]);
 xlabel('time in second'); ylabel('amplitude');
 title('A=0.3, F=25Hz, Fs=1000 Hz, T=0.1 sec')
```

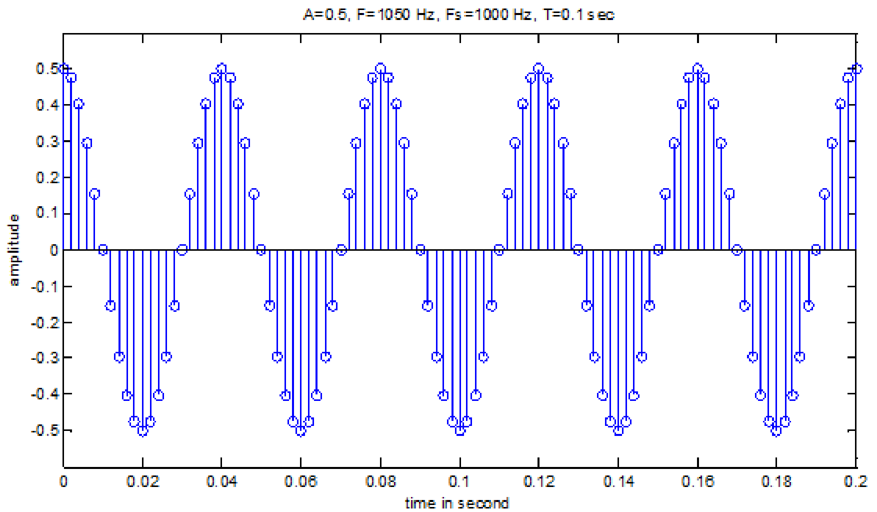
```
% Verifying two discreet time sinusoids are same if ratio of F and Fs are same
 [s3, t3] = mysin (0.5, 25, 500, 0.2); % A=0.5, F=25. Fs=500, T=0.2
 subplot (223);  stem(t3, s3);  axis ([0 0.2 -0.6 0.6]);
 xlabel('time in second'); ylabel('amplitude');
 title('A=0.5, F=25Hz, Fs=500 Hz, T=0.2 sec')

% Verifying two discreet time sinusoids are same if frequencies are F and (F+Fs)
 [s4, t4] = mysin (0.5, 1050, 1000, 0.1); % A=0.5, F=1050. Fs=1000, T=0.1
 subplot (224);  stem(t4, s4);  axis ([0 0.1 -0.6 0.6]);
 xlabel('time in second'); ylabel('amplitude');
 title('A=0.5, F=1050 Hz, Fs=1000 Hz, T=0.1 sec')
```

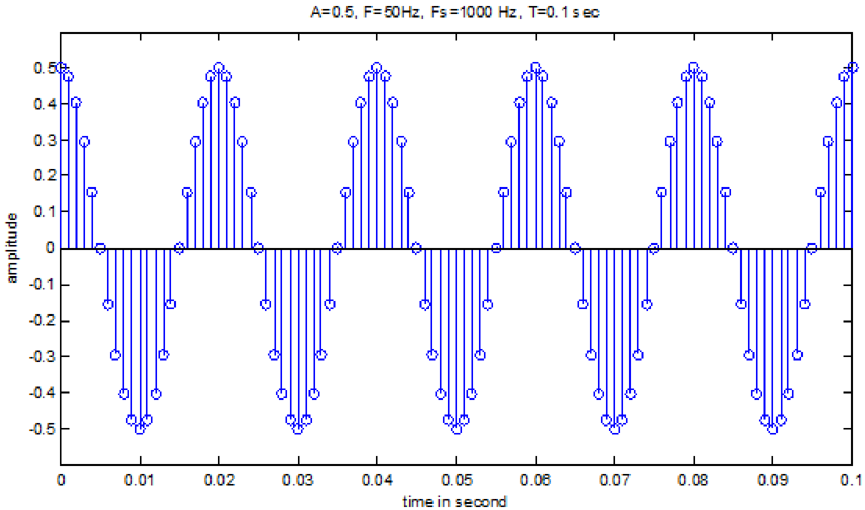


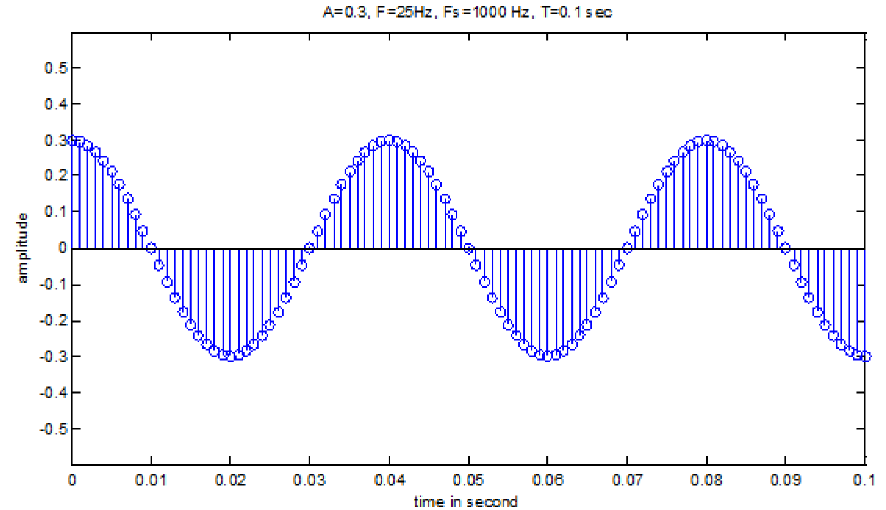




## Anything wrong anywhere?

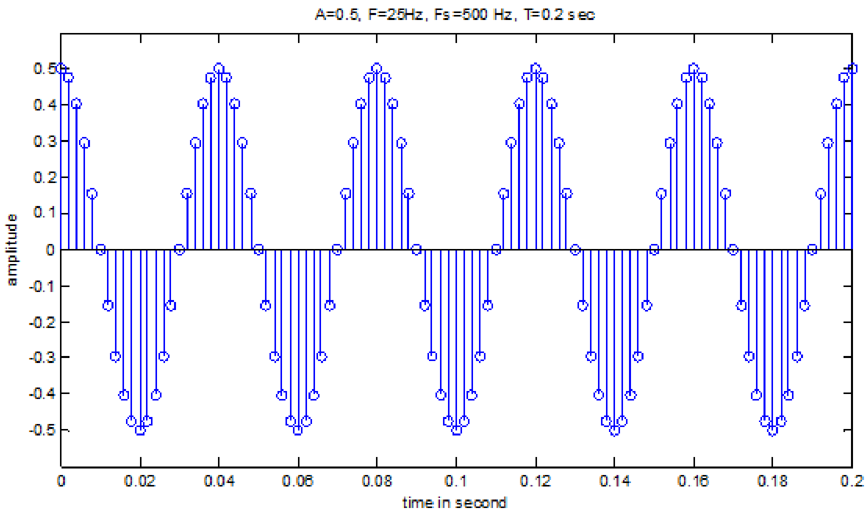


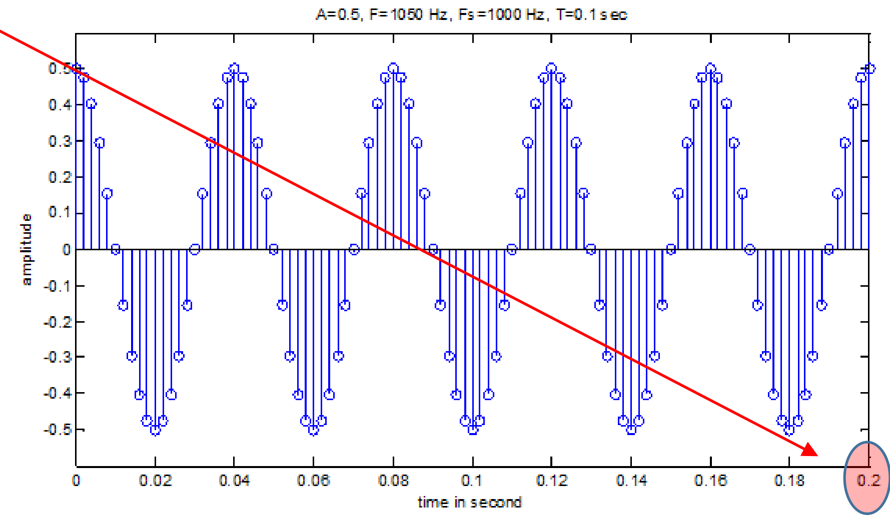



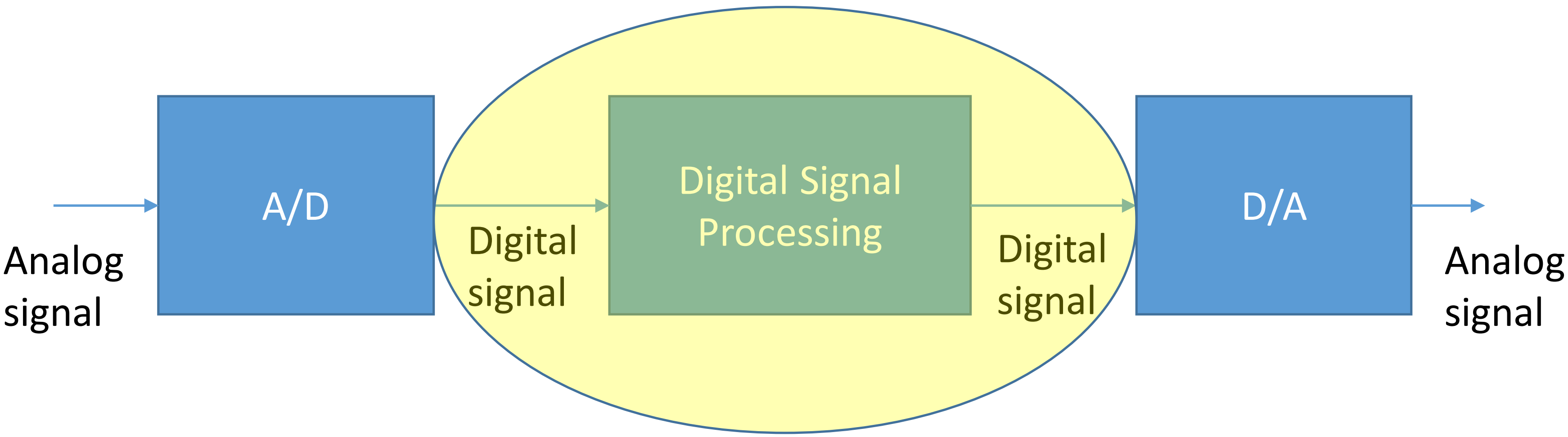



**Anything wrong anywhere?**






$$A\cos(2\pi f t) \rightarrow A\cos\left(\frac{2\pi f n}{F}\right) = A\cos\left(\frac{2\pi (kf) n}{(kF)}\right)$$

F=50. Fs=1000

F=25. Fs=500

$$A\cos(2\pi (f + F) t) \rightarrow A\cos\left(\frac{2\pi (f + F) n}{F}\right)$$

$$= A\cos\left(\frac{2\pi f n}{F} + 2n\pi\right)$$

F=50. Fs=1000

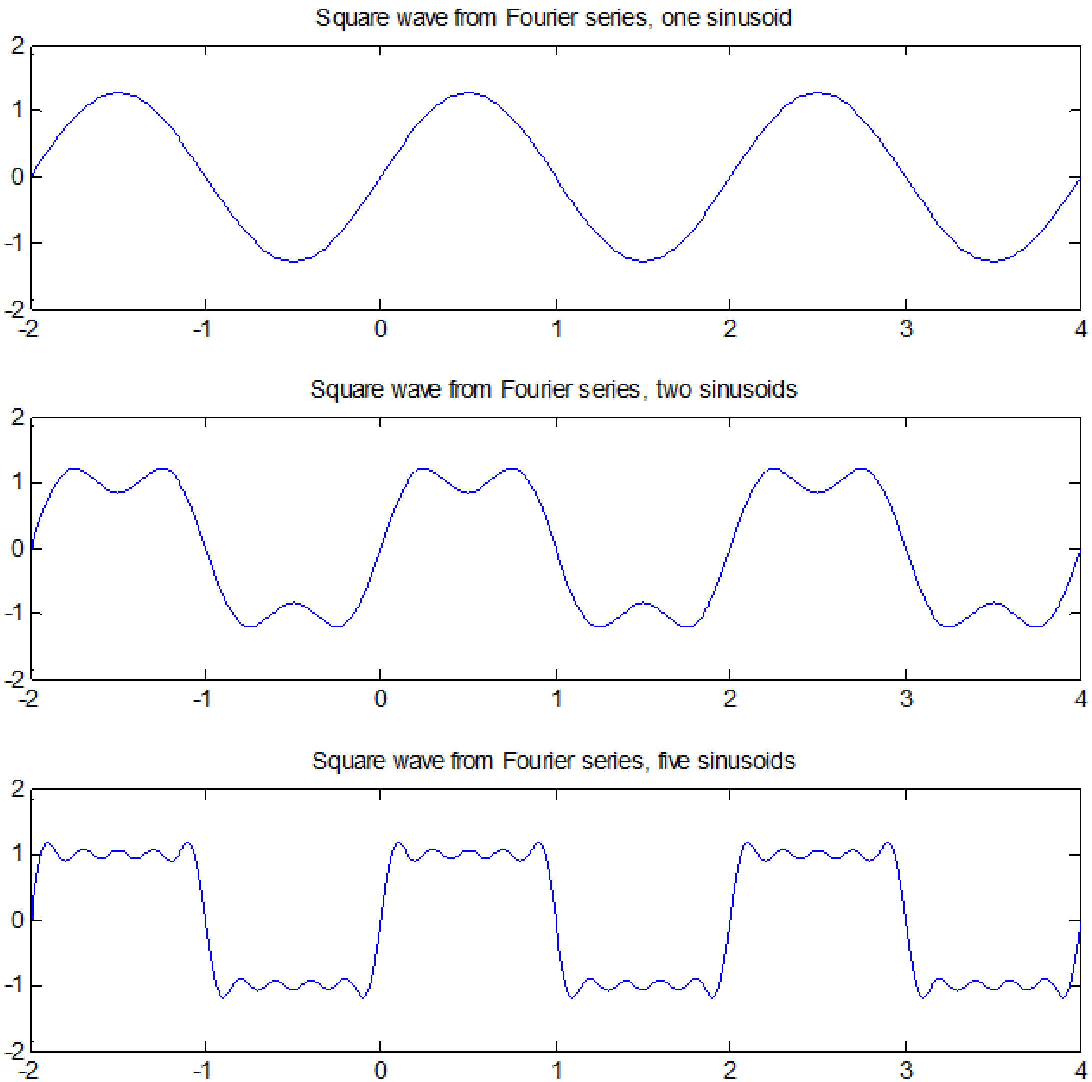
F=1050. Fs=1000

How will $F=950$. $Fs=1000$ look like?

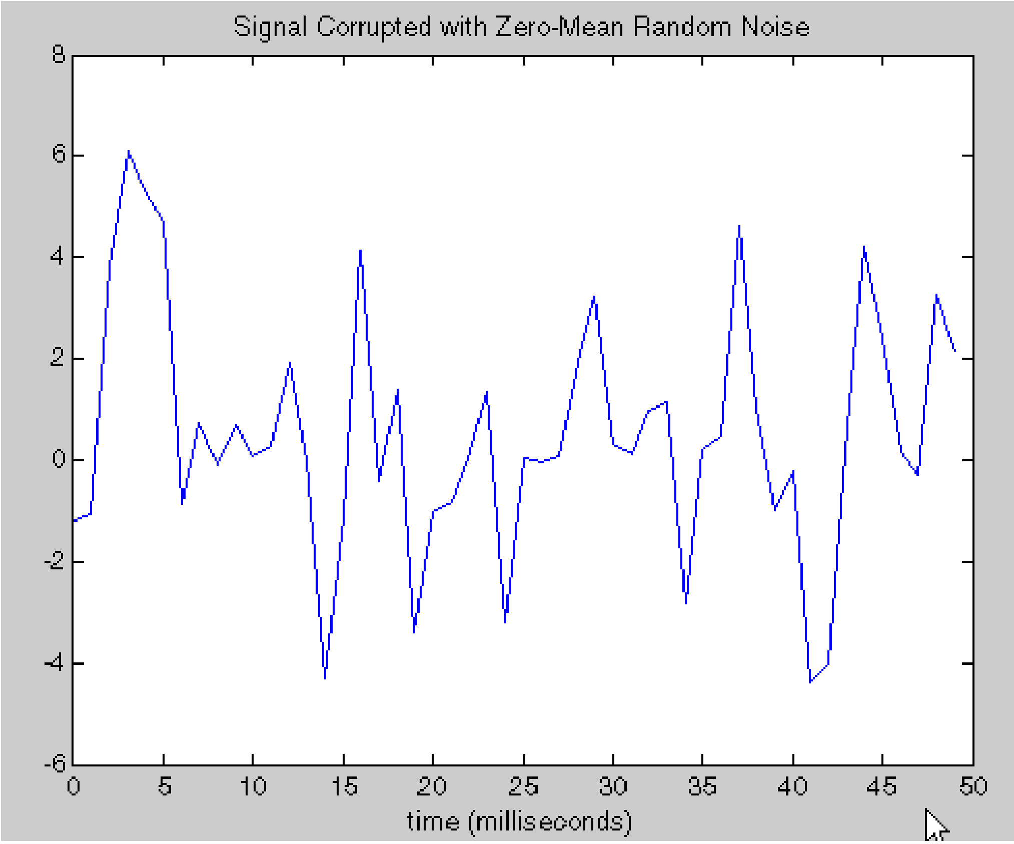
```
% Here we write a function mysquare.m that reconstructs a square wave from its
% Fourier coefficients. T0 = Time period of square wave, n= No. of harmonics used
function [s,t]=mysquare(T0,n)

F0=1/T0;     % Fundamental frequency
t=-1*T0:1/(100*F0):2*T0; % Three cycles generated(-T0 to 2T0),
% Fs = 100*F0 : Note that this is simulation of analog signal in digital platform

s=0;
for i=1:n
    j=2*(i-1)+1;       % Only odd harmonics of sin function are present
    s=s+(4*sin(2*pi*F0*j*t))/(pi*j); % Weighted sum from Fourier expansion
end
```

$$x(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2\pi(2k-1)ft)}{2k-1}$$

```
% Exp. 3
% Use function mysquare.m to see addition of weighted sinusoids give a square wave

[s,t]=mysquare(2,1);   %Time period chosen 2 second, only fundamental
subplot(3,1,1); plot(t,s); % Figure is divided into 3x1 subplots, 1st chosen
title( 'Square wave from Fourier series, one sinusoid' );

[s,t]=mysquare(2,2);   %Time period chosen 2 second, two sinusoids
subplot(3,1,2); plot(t,s); % 2nd of 3x1 subplots chosen
title( 'Square wave from Fourier series, two sinusoids' );

[s,t]=mysquare(2,5);   %Time period chosen 2 second, five sinusoids
subplot(3,1,3); plot(t,s); % 3rd of 3x1 subplots chosen
title('Square wave from Fourier series, five sinusoids' );
```


Square wave from Fourier series, one sinusoid
Square wave from Fourier series, two sinusoids
Square wave from Fourier series, five sinusoids
2
1
0
-1
-2
-2
-1
0
1
2
3
4

```
Fs = 1000;     % Sampling frequency
T = 1/Fs;       % Sample time
L = 1000;       % Length of signal
t = (0:L-1)*T;      % Time vector

% Sum of a 50 Hz and a 120 Hz sinusoid
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
y = x + 2*randn(size(t));    % noise addition
plot(Fs*t(1:50),y(1:50))

title('Signal Corrupted with Zero-Mean
Random Noise')
xlabel('time (milliseconds)')
```



## (a) Sampling of a sinusoidal waveform

**Theory:** The sampling theorem says that an analog signal bandlimited to B Hz, if sampled at Fs > 2B, the analog signal can be completely reconstructed from its samples. To compute the spectrum of an analog signal numerically, the sampled waveform has to be truncated to apply DFT. This truncation or rectangular windowing in time domain causes spectrum spreading. The more the width of the window is chosen the less is spreading. Note also that the DFT involves a sampling in frequency domain.

1. Take an analog waveform $x(t) = 10\cos(2\pi\times10^3 t) + 6\cos(2\pi\times2\times10^3 t) + 2\cos(2\pi\times4\times10^3 t)$.
2. Sample it at Fs = 12 kHz.
3. Obtain DFT of x (t) with N = 64, 128, 256 points and plot the respective magnitude spectra (Fig.1).

- Note the change in spectrum as N is increased.

## (b) Sampling at below Nyquist rate and effect of aliasing

1. Repeat above with Fs = 8 KHz, 5 kHz, 4 kHz.

- Note carefully, that sampling theorem requires $F_s > 2F_{max}$ (without equality sign) for sinusoidal signal.

2. Find out from the spectrum, what are the aliases of the original frequencies present in x (t) when the sampling rate is below the Nyquist rate.

```
>> help fft
   fft Discrete Fourier transform.

   fft(X) is the discrete Fourier transform (DFT) of vector X.

   fft(X,N) is the N-point fft, padded with zeros if X has less
   than N points and truncated if it has more.

   Reference page in Help browser
      doc fft
```

Help

fft ✕ +

Other uses of fft:
comm/fft, ident/fft, dsp/FFT, dsp/dsp.FFT, vision/vision.FFT

Contents



Search Documentation



# fft

Fast Fourier transform

## Syntax

```
Y = fft(x)
Y = fft(X,n)
Y = fft(X,[],dim)
Y = fft(X,n,dim)
```

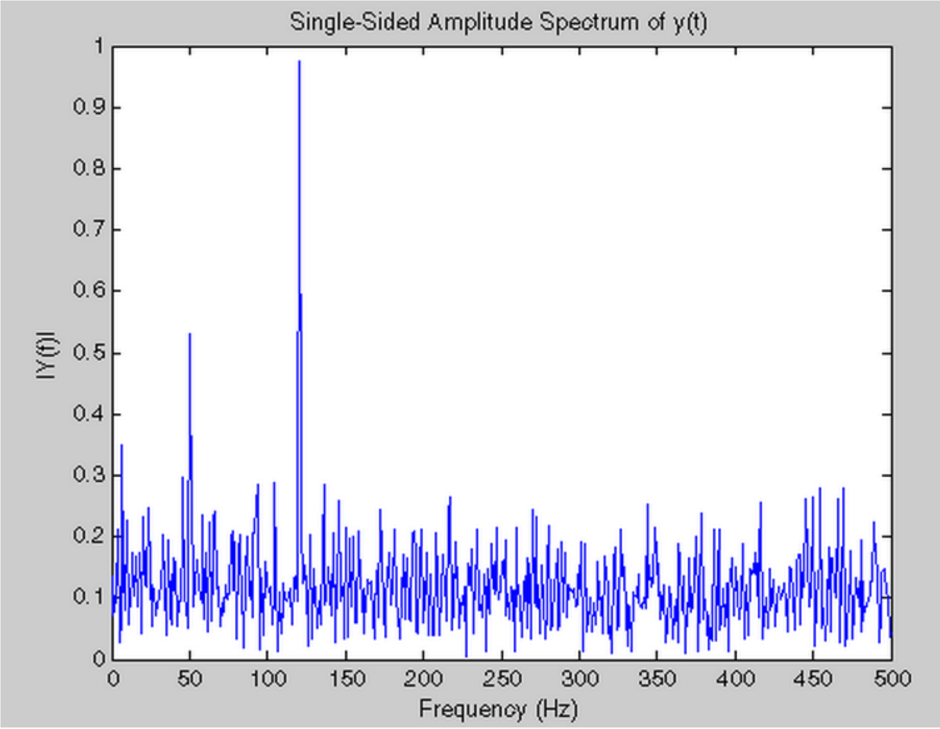
## Definitions

The functions `Y = fft(x)` and `y = ifft(X)` implement the transform and inverse transform pair given for vectors of length *N* by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

```
NFFT = 2^nextpow2(L);
% Next power of 2 from length of y

Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
plot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
```



>> help fftshift

**fftshift** Shift zero-frequency component to center of spectrum.

Y = fftshift(X) rearranges the outputs of fft, fft2, and fftn by moving the zero-frequency component to the center of the array.

It is useful for visualizing a Fourier transform with the zero-frequency component in the middle of the spectrum.

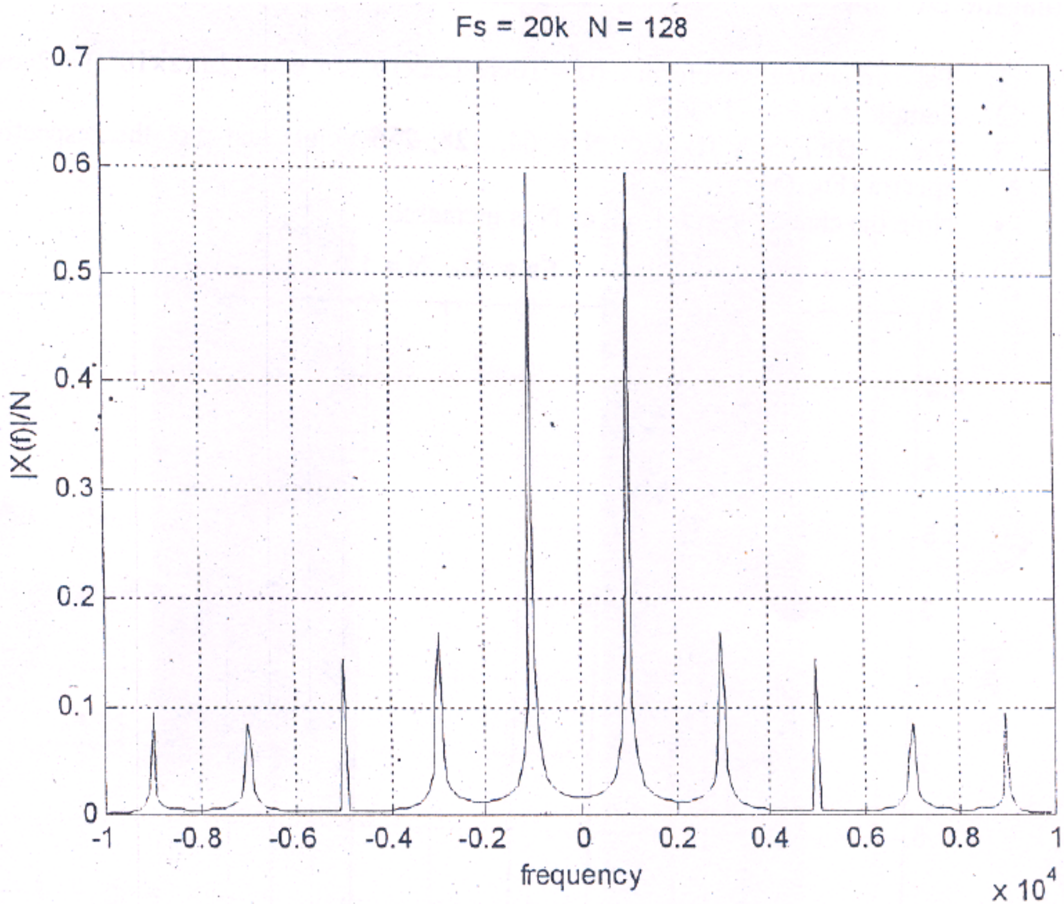
For vectors, fftshift(X) swaps the left and right halves of X.

For matrices, fftshift(X) swaps the first quadrant with the third and the second quadrant with the fourth.

## (c) Spectrum of a square wave

**Theory:** A square wave theoretically has an infinite bandwidth. For practical purposes, the spectrum beyond the $10^{th}$ harmonic can be neglected.

1. Take a square wave with time period T = 1ms (F = 1 kHz).
2. Sample it at Fs = 20 kHz.
3. Obtain DFT of the sampled square wave with N = 256 points and plot the results. (Fig. 2)



```
>> help square
   square Square wave generation.

   square(T) generates a square wave with period 2*Pi for the
   elements of time vector T.  square(T) is like SIN(T), only
   it creates a square wave with peaks of +1 to -1 instead of
   a sine wave.

   square(T,DUTY) generates a square wave with specified duty
   cycle. The duty cycle, DUTY, is the percent of the period
   in which the signal is positive.

   For example, generate a 30 Hz square wave:
       t = 0:.0001:.0625;
       y = square(2*pi*30*t);, plot(t,y)
```

### (d) Interpolation or upsampling

**Theory:** If an analog signal is sampled at a frequency higher than the Nyquist rate, ($F_{s1} > 2F_{max}$) it is possible to interpolate the intermediate L-1 samples or in other words to obtain the samples at $F_{s2} = LF_{s1}$ frequency. This can be simply done by passing the sampled signals through an ideal lowpass filter of cutoff frequency $F_{max}$ and sampling it again at a higher rate. But in discrete time domain this is achieved as shown in Fig.3. Here, $F_{s1}$ = 12 kHz & $F_{s2}$ = 24 kHz (L=2).

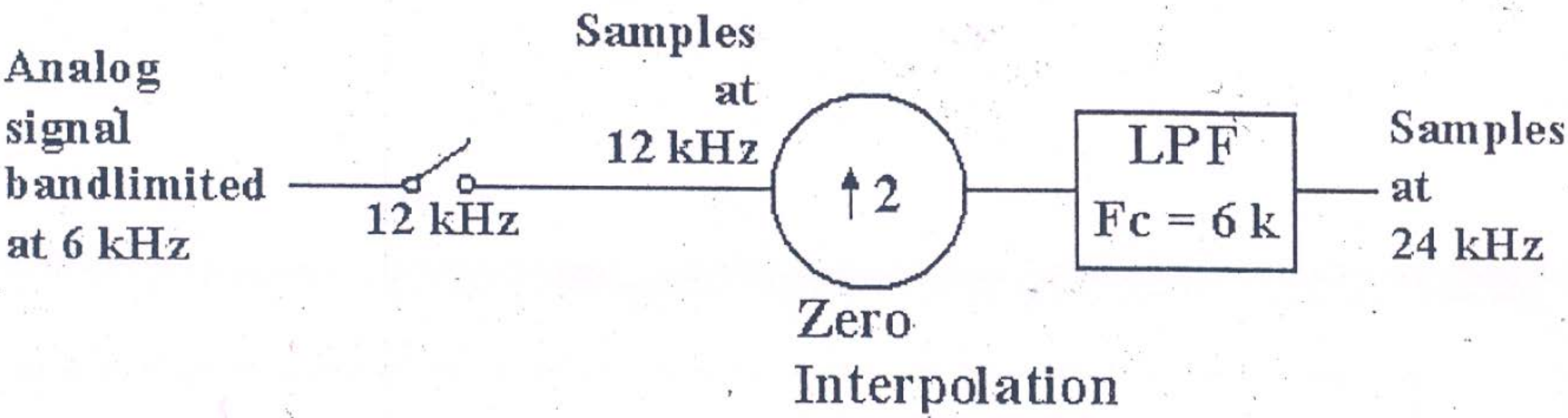



**Fig.3** An example of upsampling

1. Take a lowpass signal of bandwidth 6 kHz.
2. Sample it at $F_{s1}$ = 12 kHz.
3. Insert one zero in between every two samples.

4. Pass it through a lowpass filter of cutoff frequency 6 kHz.

- Note that at step 4 the LPF is a digital filter & the sampling frequency you have to use is Fs2 = 24 kHz.

5. Plot the output of the lowpass filter and compare it with the original signal sampled at Fs = 24 kHz.

- The output of the lowpass filter may differ from the original one by certain delay and scaling factor.

- **MATLAB functions that you may need:**
  `fft,fftshift,conv,square,fir1,fir2,butter,filter,`
  `cheby1,cheby2,ellip`