1. %Sinusoidal sequence:-

```
t=0:0.01:pi;
y=sin(2*pi*t);
subplot(2,2,1);
plot(t,y);
ylabel('Amplitude');
xlabel('e');
title('Sinusoidal Sequence');
```

2. Cosine Sequence:-

```
t=0:0.01:pi;
y=cos(2*pi*t);
subplot(2,2,1);
plot(t,y);
ylabel('Amplitude');
xlabel('f');
title('Cosine Sequence');
```

3. Plot a continuous signal

```
%plot -s 560,300

t = [0:0.01:2];
x = sin(2*pi*t);

plot(t,x,t,zeros(size(t)),'k--'), ylim([-1.1 1.1])
xlabel('t [sec]');
ylabel('x(t)');
```

4. *%plot Signal Continus and Discrete-s 560,420*

```
N = 20;
n = 0:N-1;
x = sin(2*pi/N*n);

subplot(2,1,1);  plot(n,x), axis tight
subplot(2,1,2);  stem(n,x), axis tight



%plot -s 560,200
plot(n,x,'k--'),
hold on stem(n,x,'filled','markersize',4),
 hold off ylim([-1.1 1.1])
```

5. Sine wave

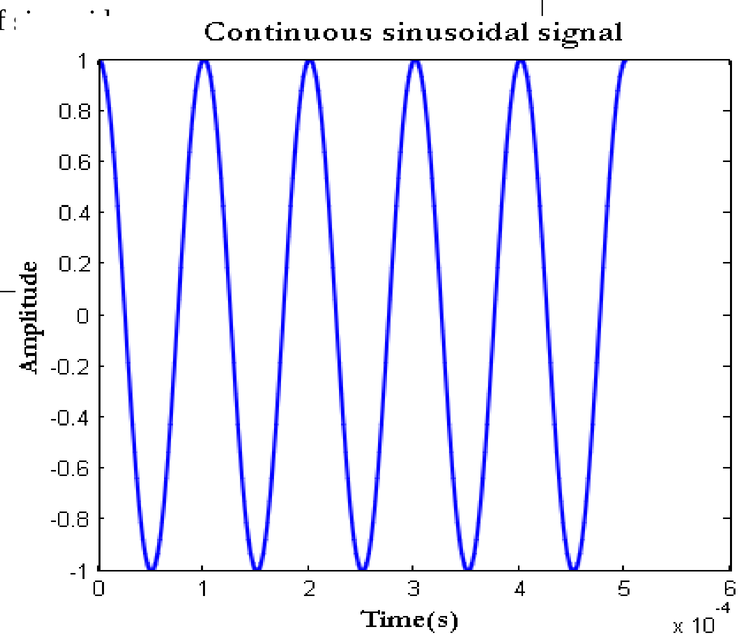
```
%%Time specifications:
   Fs = 8000;                   % samples per second
   dt = 1/Fs;                   % seconds per sample
   StopTime = 0.25;             % seconds
   t = (0:dt:StopTime-dt)';     % seconds
   %%Sine wave:
   Fc = 60;                     % hertz
   x = cos(2*pi*Fc*t);
   % Plot the signal versus time:
   figure;
   plot(t,x);
   xlabel('time (in seconds)');
   title('Signal versus Time');
   zoom xon;
```

6. For baseband signal, the sampling is straight forward. By Nyquist Shannon sampling theorem, for faithful reproduction of a continuous signal in discrete domain, one has to sample the signal at a rate **fs higher than at-least twice the maximum frequency fmfmcontained** in the signal (actually, it is twice the one-sided bandwidth occupied by a real signal. For a baseband signal bandwidth (0 to fm) and maximum frequency fm in a given band are equivalent).

Matlab or any other simulation softwares process everything in digital i.e, discrete in time. Therefore, we cannot generate a real continuous-time signal on it, rather we can generate a "continuous-like" signal by using a very very high sampling rate. When plotted, such signals look like a continuous signal.

Let's generate a simple continuous-like sinusoidal signal with frequency fm=10kHz In order to make it appear as a continuous signal when plotting, a sampling rate of fs=500kHz is used.

```
fs=500e3;                 %Very high sampling rate 500 kHz
f=10e3;                   %Frequency of sinusoid
nCyl=5                    %generate five cycles of
t=0:1/fs:nCyl*1/f;        %time index
x=cos(2*pi*f*t);

plot(t,x)
title('Continuous sinusoidal signal');
xlabel('Time(s)');
ylabel('Amplitude');
```

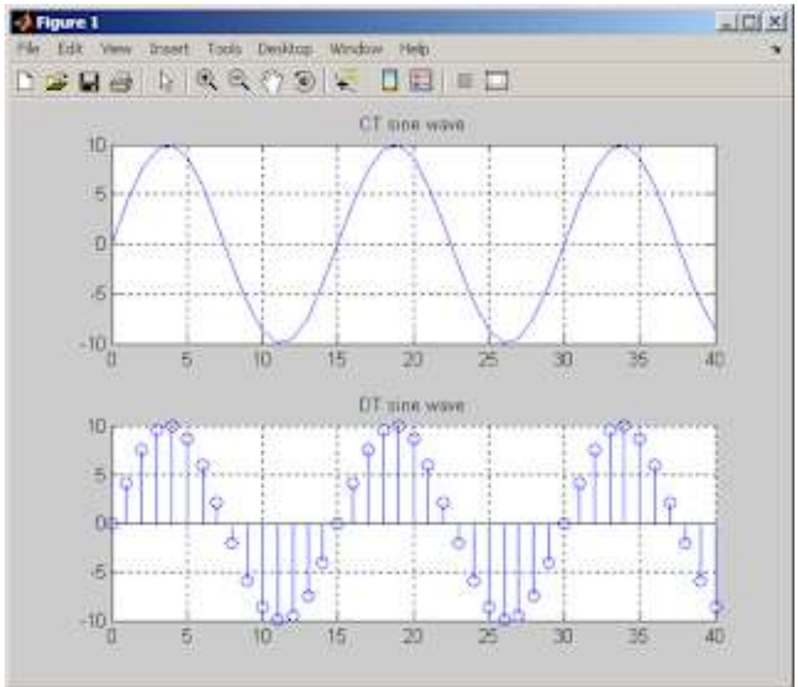


a) **plot(x,y)** to obtain the graph in Continuous time(CT)

b) **stem(x,y)** to obtain the graph in Discrete time (DT)

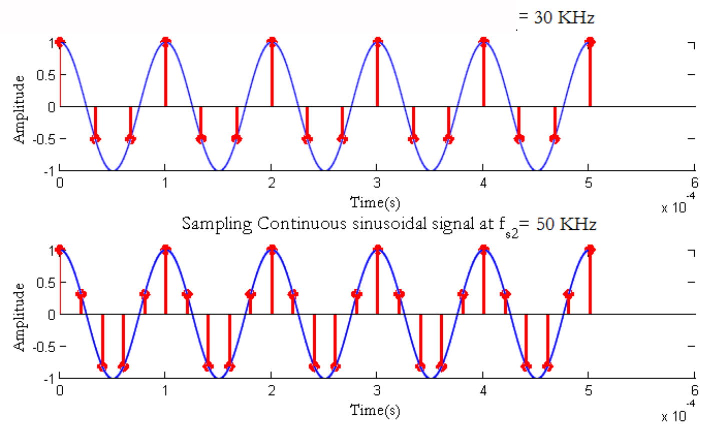
7. **Computing the graph of wave sequences both in CT and DT**

```
clc
clear all
x=0:1:40;
y=10*sin(2*pi*x/15);
subplot(2,1,1)
plot(x,y) %CT
title('CT sine wave')
grid
subplot(2,1,2)
stem(x,y) %DT
title('DT sine wave')
grid
```



8. Pretending the above generated signal as a sinusoidal signal, we would like to convert the signal to discrete-time equivalent by sampling. **By Nyquist Shannon Theorem, the signal has to be sampled at** *at-least* **fs=2∗fm=20kHz Let's sample the signal fs1=30kHz and then at fs1=50kHz for illustration**

```
fs1=30e3; %30kHz sampling rate
t1=0:1/fs1:nCyl*1/f; %time index
x1=cos(2*pi*f*t1);

fs2=50e3; %50kHz sampling rate
t2=0:1/fs2:nCyl*1/f; %time index
x2=cos(2*pi*f*t2);

subplot(2,1,1);
plot(t,x);
hold on;
stem(t1,x1);
subplot(2,1,2);
plot(t,x);
hold on;
stem(t2,x2)
```



Sampling a signal:

9. **To sample a signal in MATLAB, generate a time vector at the appropiate rate, and use this to generate the signal. Plot using the `stem` function.**
For example:

```
% Sample the sinusoid x = sin(2 pi f t), where f = 2 kHz.
% Let x1 be the signal sampled at 10 kHz.
% Let x2 be the signal sampled at 3 kHz.

f = 2000;
T = 1/f;
tmin = 0;
tmax = 5*T;              % Plot 5 cycles
dt1 = 1/10000;
dt2 = 1/3000;
t1 = tmin:dt1:tmax;
t2 = tmin:dt2:tmax;
x1 = sin(2*pi*f*t1);
x2 = sin(2*pi*f*t2);
subplot(211)
```

```
stem(t1,x1);
subplot(212)
stem(t2,x2);
```

10. It is useful to plot the continuous time signal on the same plot. All signals in MATLAB are discrete-time, but they will look like continuous-time signals if the sampling rate is much higher than the Nyquist rate:

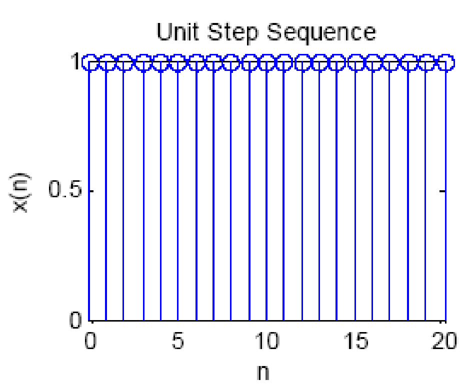
```
% Sample the sinusoid x = sin(2 pi f t), where f = 2 kHz, and plot the
sampled
% signals over the continuous-time signal.
% Let x1 be the signal sampled at 10 kHz.
% Let x2 be the signal sampled at 3 kHz.

f = 2000;
T = 1/f;
tmin = 0;
tmax = 5*T;
dt = T/100;
dt1 = 1/10000;
dt2 = 1/3000;
t = tmin:dt:tmax;
t1 = tmin:dt1:tmax;
t2 = tmin:dt2:tmax;
x = sin(2*pi*f*t);
x1 = sin(2*pi*f*t1);
x2 = sin(2*pi*f*t2);
subplot(211)
plot(t,x,'r');
hold on
stem(t1,x1);
subplot(212)
plot(t,x,'r');
hold on
stem(t2,x2);
```

11. Unit Sample Sequence

```
n=-10:20
ns=[zeros(1,10) 1 zeros(1,20)];
subplot(3,1,1);
stem(n,ns);
title('unit impulse')
```

12. Unit Step sequence:

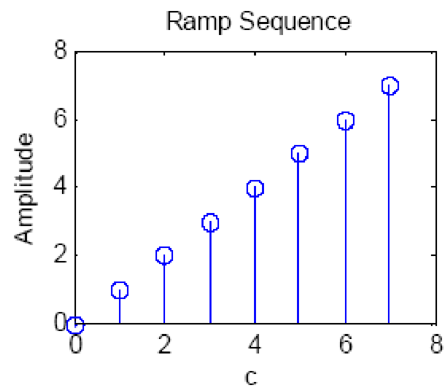
```
N=21;
x=ones(1,N);
```



```
n=0:1:N-1;
subplot(2,2,1);stem(n,x);
xlabel('n');ylabel('x(n)');
title('Unit Step Sequence');
```

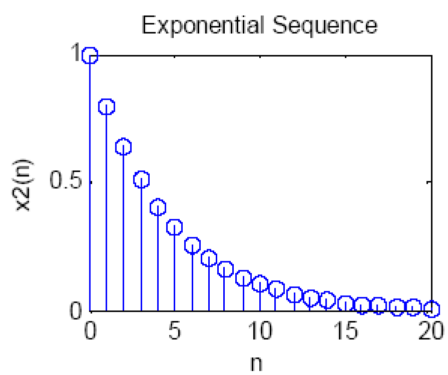
13. Ramp Sequence

```
x=input('enter the length of ramp sequence')
enter the length of ramp sequence
x =7
t=0:7;
subplot(2,2,1);stem(t,t);
xlabel('c');
ylabel('Amplitude');
title(' Ramp Sequence');
```



14. Exponential sequence: -(0<a<1)

```
x2=0.8.^(n);
subplot(2,2,3);stem(n,x2);
xlabel('n');ylabel('x(n)');
title('Exponential Sequence');
```



15. xponential sequence: -(1<a)

```
x2=2.^(n);
subplot(2,2,3);stem(n,x2);
xlabel('n');ylabel('x(n)');
title('Exponential Sequence');
```

16. Exponential can real and imag parts

```
N = 20;
n = 0:2*N-1;
x = exp(-n/N).*exp(1j*2*pi/N*n);
%plot -s 560,350

subplot(2,1,1);     stem(n,real(x),'markersize',4), ylabel('real')
```

```
subplot(2,1,2);     stem(n,imag(x),'markersize',4), ylabel('imag'), xlabel('n')
```

17. 3d plot real and imaginary plot

```
%plot -s 560,420
plot3(n,real(x),
imag(x),'o','markersize',4)
xlabel('n'),
 ylabel('real'),
zlabel('imag')
grid on
```

18. *magnitude and phase*

```
%plot -s 560,420
%% magnitude and phase
subplot(2,1,1);     stem(n,abs(x),'markersize',3), ylabel('amplitude')
subplot(2,1,2);     stem(n,(phase(x)),'markersize',3), ylabel('phase'), xlabel('n')
```

19. phase and angle

```
% see the difference between 'phase' and 'angle'
subplot(2,1,1);     stem(n,phase(x),'markersize',3), ylabel('phase')
subplot(2,1,2);     stem(n,angle(x),'markersize',3), ylabel('angle'), xlabel('n')
```

20. Polar angle coordinate

```
%% polar coordinate
polar(angle(x),abs(x),'o')
% theta in radian
```

21. Plot the following equation

1) $\cos\left(\frac{5}{7}\pi n\right)$

$$N = 14$$

$$k = 5$$

2) $\cos\left(\frac{1}{5}\pi n\right)$

$$N = 10$$
$$k = 1$$

3) Which frequency is higher?

$$\cos\left(\frac{5}{7}\pi n\right) \quad \text{or} \quad \cos\left(\frac{1}{5}\pi n\right)$$

4) $\cos\left(\frac{5}{7}\pi n\right) + \cos\left(\frac{1}{5}\pi n\right)$

$$N = ?$$
$$k = ?$$

5) Which one is a higher frequency?

$$\omega_0 = \pi \quad \text{or} \quad \omega_0 = \frac{3\pi}{2}$$

```
n = 0:7;
x1 = cos(pi*n);
x2 = cos(3/2*pi*n);

subplot(1,2,1), stem(n,x1), axis([0,7,-1.5 1.5])
subplot(1,2,2), stem(n,x2), axis([0,7,-1.5 1.5])
```

```
n = 0:31;
x1 = cos(0*pi*n);
x2 = cos(1/8*pi*n);
x3 = cos(1/4*pi*n);
x4 = cos(1*pi*n); subplot(4,1,1),
stem(n,x1,'filled','markersize',3), axis([0,31,-1.5 1.5]) subplot(4,1,2), stem(n,x2,'filled','markersize',3),
axis([0,31,-1.5 1.5]) subplot(4,1,3), stem(n,x3,'filled','markersize',3), axis([0,31,-1.5 1.5]) subplot(4,1,4),
stem(n,x4,'filled','markersize',3), axis([0,31,-1.5 1.5]
```

```
n = 0:31;
x5 = cos(2*pi*n);
x6 = cos(15/8*pi*n);
```

```
x7 = cos(7/4*pi*n);
x8 = cos(1*pi*n);

subplot(4,1,1), stem(n,x5,'filled','markersize',3), axis([0,31,-1.5 1.5])
subplot(4,1,2), stem(n,x6,'filled','markersize',3), axis([0,31,-1.5 1.5])
subplot(4,1,3), stem(n,x7,'filled','markersize',3), axis([0,31,-1.5 1.5])
subplot(4,1,4), stem(n,x8,'filled','markersize',3), axis([0,31,-1.5 1.5])
```

```
n = 0:31;
x5 = cos(2*pi*n);
x6 = cos(15/8*pi*n);
x7 = cos(7/4*pi*n);
x8 = cos(1*pi*n);

subplot(4,1,1), stem(n,x5,'filled','markersize',3), axis([0,31,-1.5 1.5])
subplot(4,1,2), stem(n,x6,'filled','markersize',3), axis([0,31,-1.5 1.5])
subplot(4,1,3), stem(n,x7,'filled','markersize',3), axis([0,31,-1.5 1.5])
subplot(4,1,4), stem(n,x8,'filled','markersize',3), axis([0,31,-1.5 1.5])
```

Aliasing

In descrete signal, there is identical signals with different frequency.

$$x_1[n] = e^{j((\omega+2\pi)n+\phi)} = e^{j(\omega n+\phi)+j2\pi n} = e^{j(\omega n+\phi)}e^{j2\pi n} = e^{j(\omega n+\phi)} = x_2[n]$$

Any integer multiple of 2π will do

$$x_3[n] = e^{j((\omega+2\pi m)n+\phi)},\ m \in \mathbb{Z}$$

22. Example of aliasing signal

```
N = 15;
k = 1;
n = 1:N-1;

for i = 1:N-1
    xn(i) = cos(2*pi*k*i/N);
    xn2(i) = cos((2*pi*k/N + 2*pi)*i);
end

subplot(2,1,1), stem(n,xn)
subplot(2,1,2), stem(n,xn2)
```

```
%plot -s 560,200

t = linspace(0,10*2*pi,300);
x = sin(t);
plot(t,x), axis tight, xlabel('sec')
```

22. Anything less than 20 points will cause problems:

```
%plot -s 560,200

t = linspace(0,10*2*pi,300);
x = sin(t);
ts = linspace(0,10*2*pi,12);
xs = sin(ts);

plot(t,x,ts,xs,'o--'), axis tight, xlabel('sec')
```

```
%plot -s 560,200

t = linspace(0,10*2*pi,300);
x = sin(t);
ts = linspace(0,10*2*pi,11);
xs = sin(ts);

plot(t,x,ts,xs,'o--'), axis tight, xlabel('sec')
```

```
%plot -s 560,200
t = linspace(0,10*2*pi,300);
x = sin(t);
 ts = linspace(0,10*2*pi,20);
 xs = sin(ts);
 plot(t,x,ts,xs,'o--'),
axis tight, xlabel('sec')
```

.Quantization

```
% A simple sampling and reconstruction model for students
% beginners of Digital Signal Processing
% by Mukhtar Hussain (Email: mukhtarhussain@ciitlahore.edu.pk)
% f - The frequency of analog sinosoid signal
% F - Sampling Rate
% qbits - Number of Quantizations bits
```

```
% A - Amplitude of sinusoid signal
% L - Number of quantization levels based on qbits
% I - Quantization Interval
% sim_time - Simultaion Time
% span - x-axis range of frequency plot 1 & 3 (spectrum scope 1 & 3)
% span1 - x-axis range of frequency plot 2 (spectrum scope 2)
% NFFT - Number of FFT points
                    clc;
                    clear;
                    close all;
                    f = input('Enter the frequency of signal = ');
                    F = input('Enter the sampling frequency = ');
                    A = input('Enter max amplitude of signal = ');
                    qbits = input('Enter the number of quantization bits = ');
                    fc = input('Enter the lowpass filter cutoff frequency = ');
                    L = 2^qbits;
                    I = 2*A/(L-1);
                    % Settings for Spectrum Scope
                    span = 8*F;
                    span1 = 8*F;
                    NFFT = 256;
                    % To run simulink model
                    t = 1/f;
                    sim_time = 10*t;
                    sim('sampling.slx');
```