DSP Assignment 2 (MATLAB)

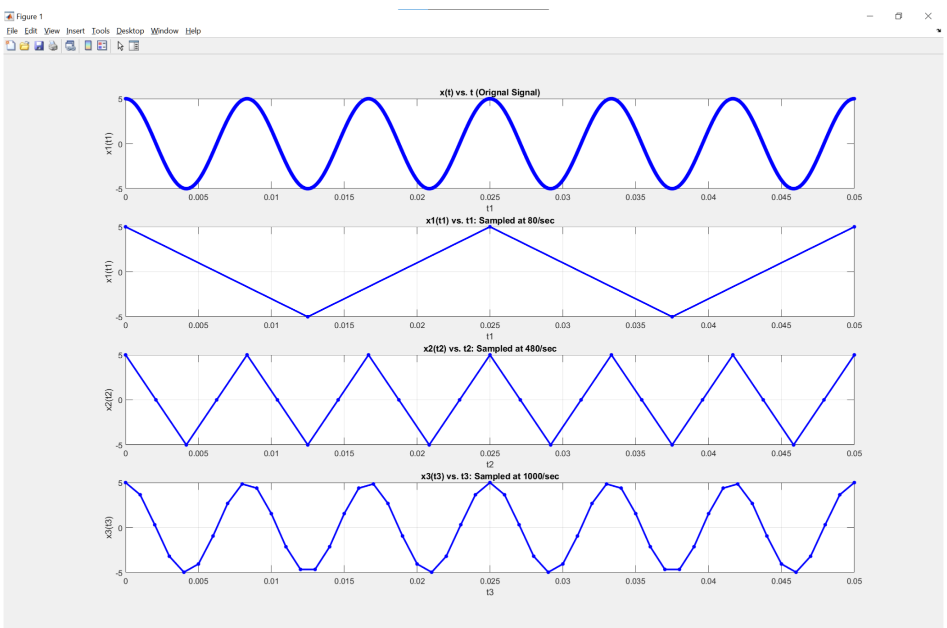
Q.1

```
close;
clc;
clear;
close all;
% Question 1
% Formula for a cosine is cos(2 * pi * t / period).
% So, 240 * pi = 2 * pi / period, so period = 2/240 = 0.00834.
% Thus 10 periods would mean a max t of 10 * 0.00834 = 0.0834.
% Sampling rates are 80, 480 and 1000 samples per second,
% Number of Samples = (elapsed time) / (time per sample)
% numSamples1 = round(0.00834 / (1/80));
% numSamples2 = round(0.00834 / (1/480));
% numSamples3 = round(0.00834 / (1/1000));
fs1 = 80;
dt1 = 1/fs1;
fs2 = 480;
dt2 = 1/fs2;
fs3 = 1000;
dt3 = 1/fs3;
t = (0:0.0000005:0.05);
% That is only 40 and gives a severe undersampling of the signal
% giving a weird shape.
% Make up a t axis going from 0 to 0.1 with numSamples samples.
t1 = (0:dt1:0.05);
t2 = (0:dt2:0.05);
t3 = (0:dt3:0.05);
% t1 = linspace(0, 0.00834, numSamples1);
% t2 = linspace(0, 0.00834, numSamples2);
% t3 = linspace(0, 0.00834, numSamples3);
x = 5 * cos(240 * pi * t);
x1 = 5 * cos(240 * pi * t1);
x2 = 5 * cos(240 * pi * t2);
x3 = 5 * cos(240 * pi * t3);
figure
subplot(4,1,1),plot(t,x,'b.-', 'LineWidth', 2, 'MarkerSize', 15);
title('x(t) vs. t (Original Signal)');
xlabel('t1');
ylabel('x1(t1)')
subplot(4,1,2),plot(t1, x1, 'b.-', 'LineWidth', 2, 'MarkerSize', 15);
grid on;
title('x1(t1) vs. t1: Sampled at 80/sec');
xlabel('t1');
ylabel('x1(t1)');
subplot(4,1,3),plot(t2, x2, 'b.-', 'LineWidth', 2, 'MarkerSize', 15);
grid on;
title('x2(t2) vs. t2: Sampled at 480/sec');
xlabel('t2');
```

```
ylabel('x2(t2)');
subplot(4,1,4),plot(t3, x3, 'b.-', 'LineWidth', 2, 'MarkerSize', 15);
grid on;
title('x3(t3) vs. t3: Sampled at 1000/sec');
xlabel('t3');
ylabel('x3(t3)');
```

According to Nyquist Theorem, fs>=2*f, our sampling frequency must be at least more than twice of the frequency, i.e. more than 240, for our original signal to be preserved, which we can see that happens in the case of 480 and 1000, with the sampling output improving with higher frequency as we can see in the 1000 case. At the same time, the case with 80 as the sampling frequency doesn't really give us a nice output.

Plot:



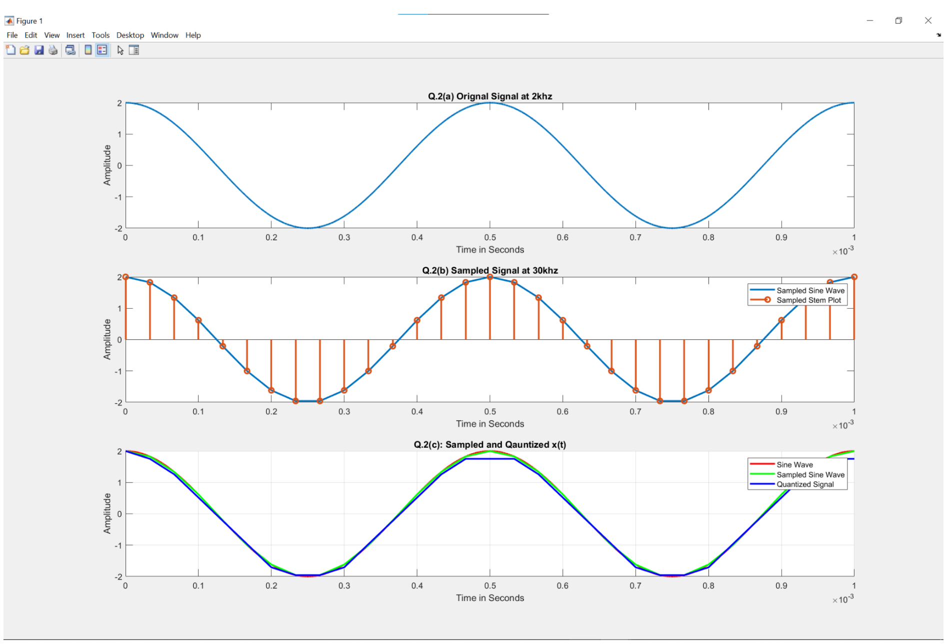
Q.2

```
close;
clc;
clear;
close all;
%Part a

% Original signal

to=0:1/(10000*2000):0.001;
freq = 2000;
a=2;
x_orig=a*cos(2*pi*freq*to);
figure
subplot(3,1,1)
plot(to,x_orig,'linewidth',2)
title("Q.2(a) Original Signal at 2khz")
xlabel('Time in Seconds')
ylabel('Amplitude')
```

```
%Part b
% 30Khz

F_s = 30000;
t = 0:1/F_s:0.001;
a=2;
samp=a*cos(2*pi*freq*t);
subplot(3,1,2);
plot(t,samp,'linewidth',2)
hold on
stem(t,samp,'linewidth',2)
hold off
title("Q.2(b) Sampled Signal at 30khz")
legend("Sampled Sine Wave","Sampled Stem Plot")
xlabel('Time in Seconds')
ylabel('Amplitude')

% Part 3

M = 16; % As given in the question
[q,r] = quantization(samp,M); % Calling the Quantization function
% figure; % Plotting the sampled signal
% stem(t,x,"g")
% disp(q)
subplot(3,1,3)
hold on; % Plotting the original Signal
plot(to,x_orig,'linewidth',2,'color','r')
p =plot(t,samp);
set((p),'linewidth',2,'color','g');
h= plot(t,q); % Plotting the quantised Signal
set((h),'linewidth',2,'color','b');
grid on;
title("Q.2(c): Sampled and Quantized x(t)");
legend("Sine Wave","Sampled Sine Wave","Quantized Signal")
xlabel('Time in Seconds')
ylabel('Amplitude')
hold off;

%Part 4
mat = zeros(1,length(q));
min = min(samp);
for i=1:length(q)
    for j = 1:length(r)
        if i == 1
            mat(i) = 16;
        elseif (q(i) == r(j))
            mat(i) = j;
        end
    end
```

```
end
disp("The quantized levels into as 4-bit representation to get a digital ")
disp("signal at each sampled time can be seen as (Taking levels from 1-16")
disp(mat)
function [q,r] = quantization(initial,M)
N=length(initial);                         % Measure the length of initial
q = zeros(1,N);                            % store the quantization values
diff=(max(initial)-min(initial))/M;          % Gap between 2 levels
r = min(initial):diff:max(initial);         % Updating the levels
for i=1:N
   for j=1:M
      if i == 1 && j == 1
         q(i) = max(initial);
      end
      if initial(i)<=r(j)+diff/2
         if initial(i)>=r(j)-diff/2
            q(i)=r(j);
         end
      elseif i ~= 1
         if initial(i)>=r(j)-diff/2
            q(i) = q(i-1);
         end
      end
   end
end
end
```

Plot:



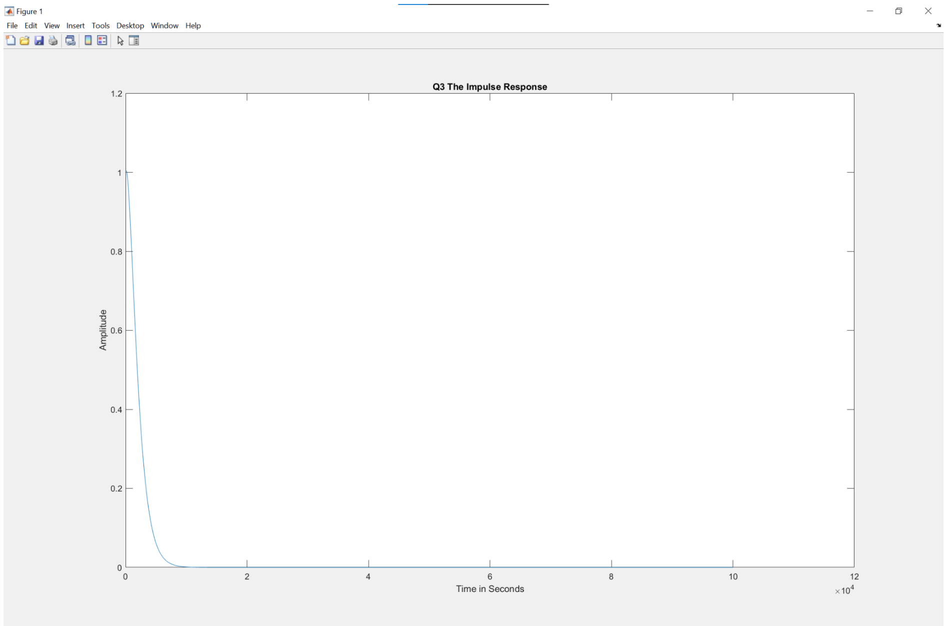
Output:

```
Command Window
The quantized levels into as 4-bit representation to get a digital
signal at each sampled time can be seen as (Taking levels from 1-16
  Columns 1 through 24

    16    16    14    11     8     5     2     1     1     2     5     8    11    14    16    16    16    14    11     8     5     2     1     1

  Columns 25 through 31

     2     5     8    11    14    16    16

fx >>
```

Q. 3

```
close;
clc;
clear;
close all;
% Question 3
% syms z
% t = (0:0.1:100);
syms z
Hs = 1/ (1 - 0.8/z + 0.16/(z^2));
% Ht = (4.*exp((2*t)/5))/5 + dirac(t) + (4*t.*exp((2*t)/5))/25;
% figure;
% plot(Ht)
% % This can be re-written as z^2/(z-0.4)^2
% Ht = tf([1,0,0],[1,-0.8,0.16]);
% % disp(Ht)
% figure;
% impulse(Ht)
% title("Q3 The Impulse Response")
Ht = iztrans(Hs)
% disp(Ht)
n = 0:0.001:100;

% Function from Z Transform of H(z) we'll use this

t = 2*(2/5).^n + (2/5).^n.*(n - 1);
plot(n,t)
xlabel('Time in Seconds')
ylabel('Amplitude')
title("Q3 The Impulse Response")
```

Plot:



Output:

```
Command Window

Ht =

2*(2/5)^n + (2/5)^n*(n - 1)

fx >>
```

Q.4

```
close;
clc;
clear;
close all;
% Question 4
rn = -1 + (1+1).*rand(1000,1);
sum1=0;
for i=1:length(rn)
  sum1=sum1+rn(i);
end
M=sum1/length(rn); %the mean
sum2=0;
for i=1:length(rn)
    sum2=sum2+ (rn(i)-M)^2;
end
V=sum2/length(rn); %The Variance
fprintf("The Random Signal has Mean as "+ M + " and Variance as "+ V);
```

Output:

Command Window

```
fx The Random Signal has Mean as 0.017576 and Variance as 0.32011>>
```