**Problem 1 (Signal reconstruction after sampling):**

Given the following signal



**Task (a)**

Implement a function ss2\_problem1() that has one input parameter n which controls the sampling frequency. The function is required to generate a plot containing two subplots.

function[] = ss2\_problem1(n)

x\_given = linspace(0,1);

y\_given = 4\*sin(2\*pi\*x\_given)+cos((pi/4)+16\*pi\*x\_given);

xs = 0: n:1;

ys = 4\*sin(2\*pi\*xs)+cos((pi/4)+16\*pi\*xs);

subplot(2,1,1)

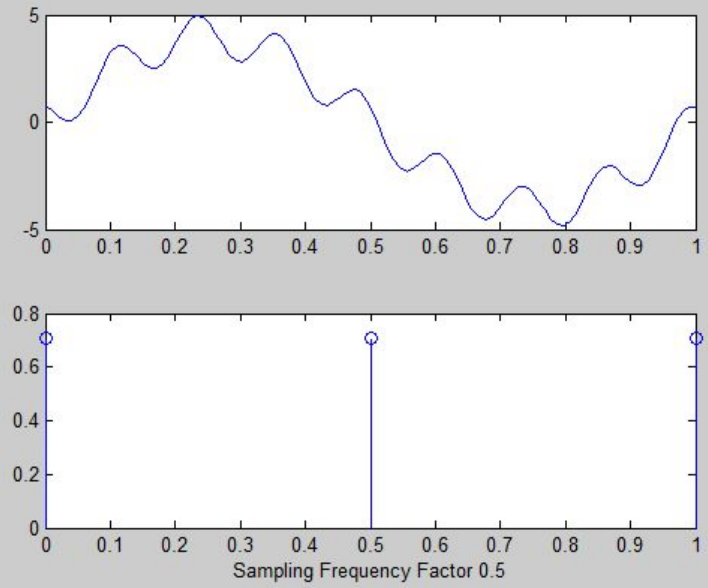
plot(x\_given,y\_given);

subplot(2,1,2);

stem(xs,ys);

xlabel(['Sampling Frequency Factor ',num2str(n),''])

end



**Task(b)**

In the first subplot, plot the signal x ( t ) within the interval t ∈ [0.0025, 1] using a temporal resolution of T = 0.0025.

function[] = ss2\_problem1(n)

x\_given = 0.0025:0.0025:1;

y\_given = 4\*sin(2\*pi\*x\_given)+cos((pi/4)+16\*pi\*x\_given);

xs = 0.0025: n:1;

ys = 4\*sin(2\*pi\*xs)+cos((pi/4)+16\*pi\*xs);

subplot(2,1,1)

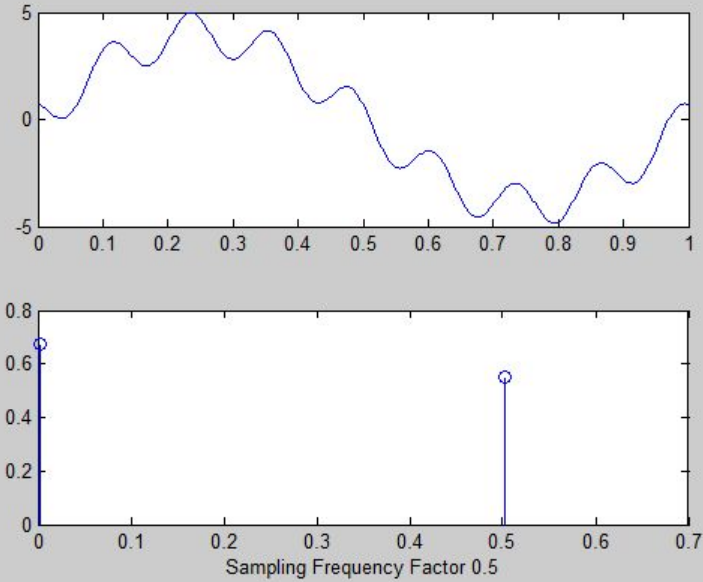
plot(x\_given,y\_given);

subplot(2,1,2);

stem(xs,ys);

xlabel(['Sampling Frequency Factor ',num2str(n),''])

end



**Task(c)**

Sample the signal x(t) and plot the sampled values xk in the first subplot, without deleting the previous plot. The sampling interval Ts is specified by Ts= T · n. That is, the larger the value of n, the lower the sampling frequency. Please ensure that the plots share the same vertical axis (identical scaling) to enable a direct comparison. Hint: The Signal Processing Toolbox provides the helpful function downsample().

function[] = ss2\_problem1(n)

x\_given = .0025:0.0025:1;

y\_given = 4\*sin(2\*pi\*x\_given)+cos((pi/4)+16\*pi\*x\_given);

plot(x\_given,y\_given);

hold on

xs = 0:n:1;

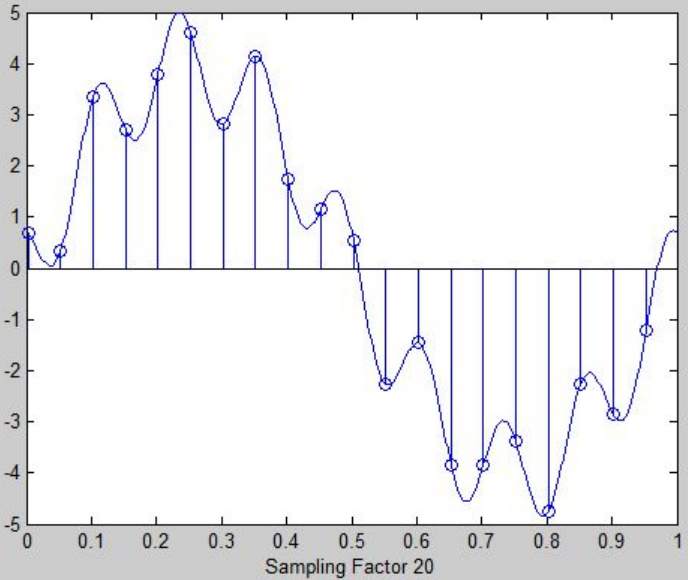
ys = 4\*sin(2\*pi\*xs)+cos((pi/4)+16\*pi\*xs);

stem(xs,ys);

xlabel(['Sampling Factor ',num2str(n),''])

hold off

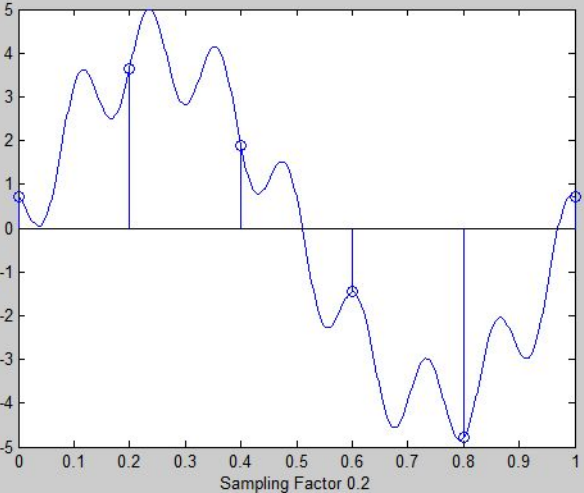
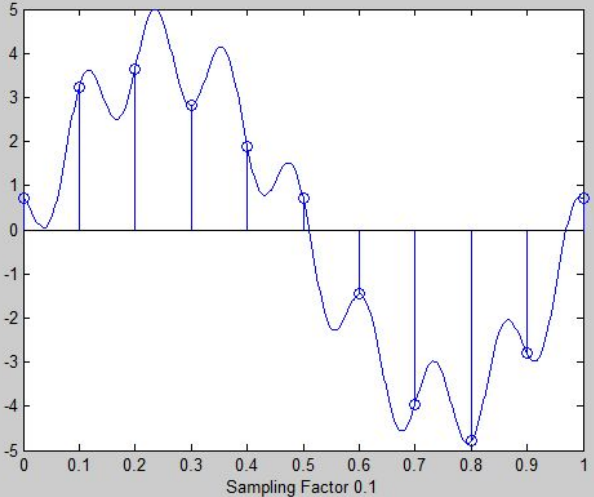
end

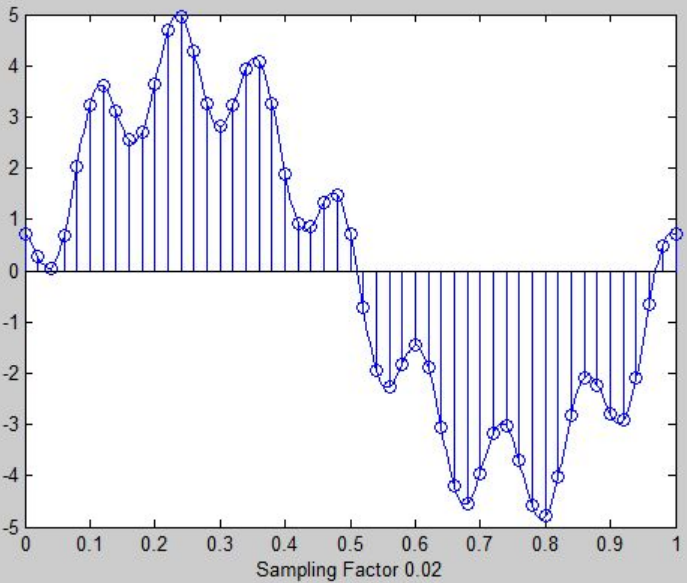
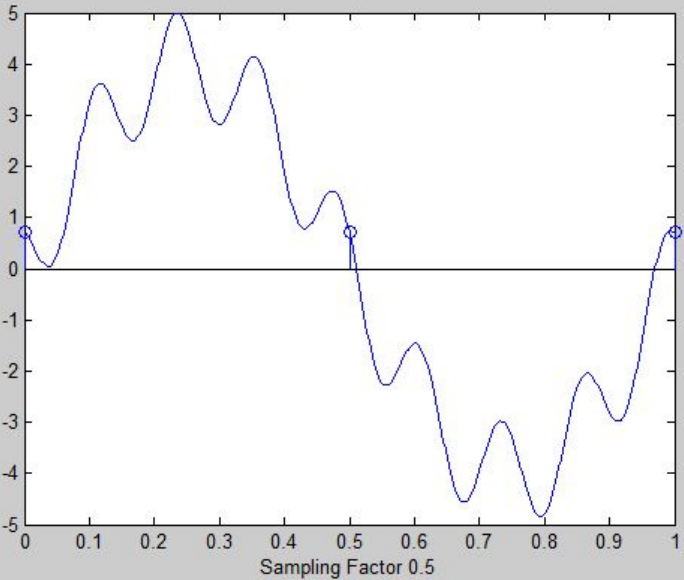


**Task(d)**

Test your MATLAB function by varying the values of the input parameter n. The sampled values show when the sampling theorem is met/ not met. This limit of n is also to be found using theoretical considerations.

We tested the function with different inputs and it seems that the sampling theorem is met.

**Task(e)**

If the sampling theorem is met, the original signal x(t) can be reconstructed from the sampling values x

k(t) based on the following relationship



You can use the MATLAB function sinc() for this purpose (check its definition!). Plot the reconstructed signal ˜x(t) in the second subplot. Hint: To verify your code, you may want to plot the individual summands as well as the original signal x (t).

function[] = SS\_lab1\_e(T)

x=0:0.05:2; %add 20 past and future samples

y=4\*sin(2\*pi\*x)+cos((pi/4)\*16\*pi\*x);

x1 = 0:0.05:2;

y1=4\*sin(2\*pi\*x1)+cos((pi/4)\*16\*pi\*x1);

for l=1:200 %observed interval is [0,0.02]

t=(l­1)\*T/100;%successive sample separation is 0.01T

h=sinc(pi\*(t­x1.\*T)./T);

xr(l)=y1\*h.'; %approximate interpolation of (4.11)

end

subplot(2,1,1);

hold on

plot(x,y);

stem(x1,y1);

xlabel(['Sampling Period is ',num2str(T),''])

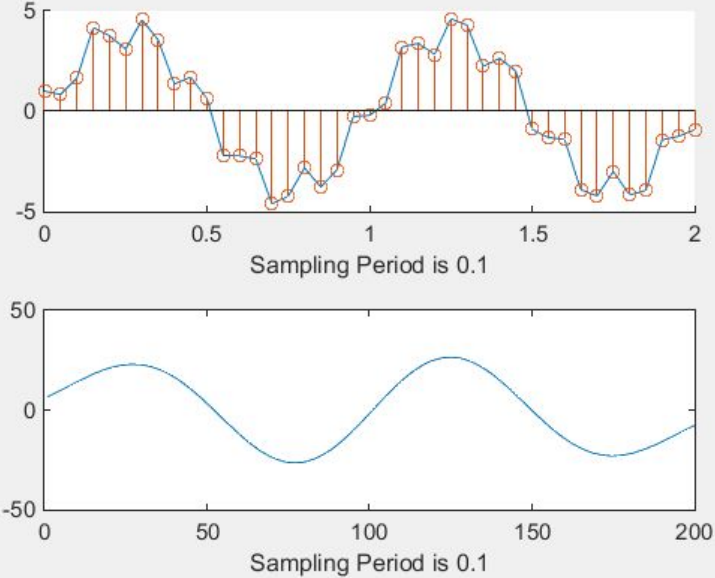
hold off

subplot(2,1,2);

plot(xr);

xlabel(['Sampling Period is ',num2str(T),''])

end



**Task(f)**

If the sampling theorem is satisfied, you should observe a close match between the reconstructed signal and the original signal. Also plot the reconstructed signal for a case where the sampling theorem is violated.

