SIGNALS AND SYSTEMS EXPERIMENT 9

Name-Swastik Panda

Roll Number-23EE01069

B-TECH IN ELECTRICAL ENGINEERING

IMPULSE TRAIN SAMPLING AND RECONSTRUCTION VIA IDEAL LOW-PASS FILTERING

1. Write a function reconstruct() that takes the following inputs: the samples of x(t),

the sampling interval T, the LPF cutoff ωc , a vector containing the values of n for

which the x(nT) are obtained, and a vector t. For each element in t, the function

should output xr(t) using the reconstruction formula given above.

```
function xr_t=reconstruct( samples,T,wc,N,t);

xr_t=zeros(1,length(t));

scaling=wc*T/pi;

for i=1:length(t)

   current_t=t(i);

   sum=0;

   for j=1:length(samples)
```

```
    arg=wc*(current_t-N(j)*T);
    if arg==0
      sum=sum+samples(j);
    else
      sum=sum+samples(j)*scaling*sin(arg)/arg;
    end
  end
  xr_t(i)=sum;
end
end
```

SAMPLING AND RECONSTRUCTING A SINUSOIDAL SIGNAL

1. Write a function sampled cosine(), whose input is a vector containing the values of

n and the sampling interval T. The output is the sequence of samples x(nT), where

x(t) = cos(2πt).

```
    function samples=sampled_cosine(N,T)
samples=zeros(1,length(N));
for i = 1:length(N)
  samples(i)=cos(2*pi*N(i)*T);
```

```
end

end
```

2. Use this function to obtain samples of x(t) over the range −6s ≤ t ≤ 6s with a

sampling frequency ωs = 5π. Thus, the sampling interval is T = 2π/ωs = 0.4s.

```
function samples=sampled_cosine(N,T)

samples=zeros(1,length(N));

for i = 1:length(N)

   samples(i)=cos(2*pi*N(i)*T);

end

end

N1=-6:0.01:6;

saa=sampled_cosine(N1,0.4);

for o=1:length(N1)

   disp(saa(o));

end
```

3. Use the reconstruct() function to obtain xr(t) for t = −6 : 0.001 : 6. Use ωc = ωs/2.

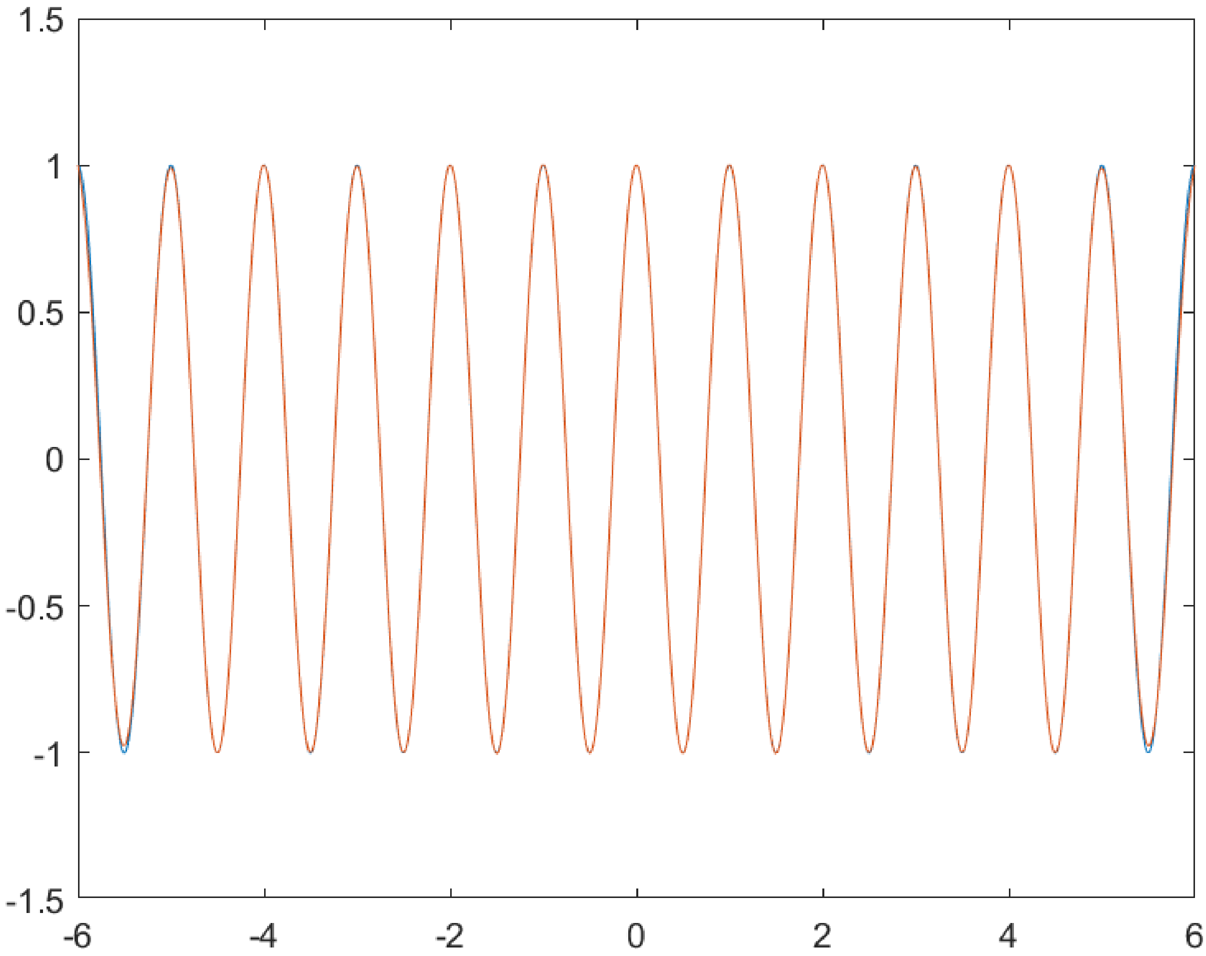
```
t=-6:0.001:6;

rec1=reconstruct(saa,0.4,2.5*pi,N1,t);
```

```
for p=1:length(N1)
  disp(rec1(p));
end
```

4. Plot x(t) and xr(t) against t on the same graph. Do they match?

```
function samples=sampled_cosine(N,T)
samples=zeros(1,length(N));
for i = 1:length(N)
   samples(i)=cos(2*pi*N(i)*T);
end
end
N1=-6/0.4:6/0.4;
saa=sampled_cosine(N1,0.4);
function xr_t=reconstruct( samples,T,wc,N,t);
xr_t=zeros(1,length(t));
scaling=wc*T/pi;
for i=1:length(t)
   current_t=t(i);
   sum=0;
   for j=1:length(samples)
      arg=wc*(current_t-N(j)*T);
      if arg==0
```

```
        sum=sum+samples(j);
      else
        sum=sum+samples(j)*scaling*sin(arg)/arg;
      end
    end
    xr_t(i)=sum;
end
end
t=-6:0.001:6;
rec1=reconstruct(saa,0.4,2.5*pi,N1,t);
for p=1:length(N1)
   disp(rec1(p));
end
plot(t,cos(2*pi*t));
hold on;
plot(t,rec1);
```



ALIASING

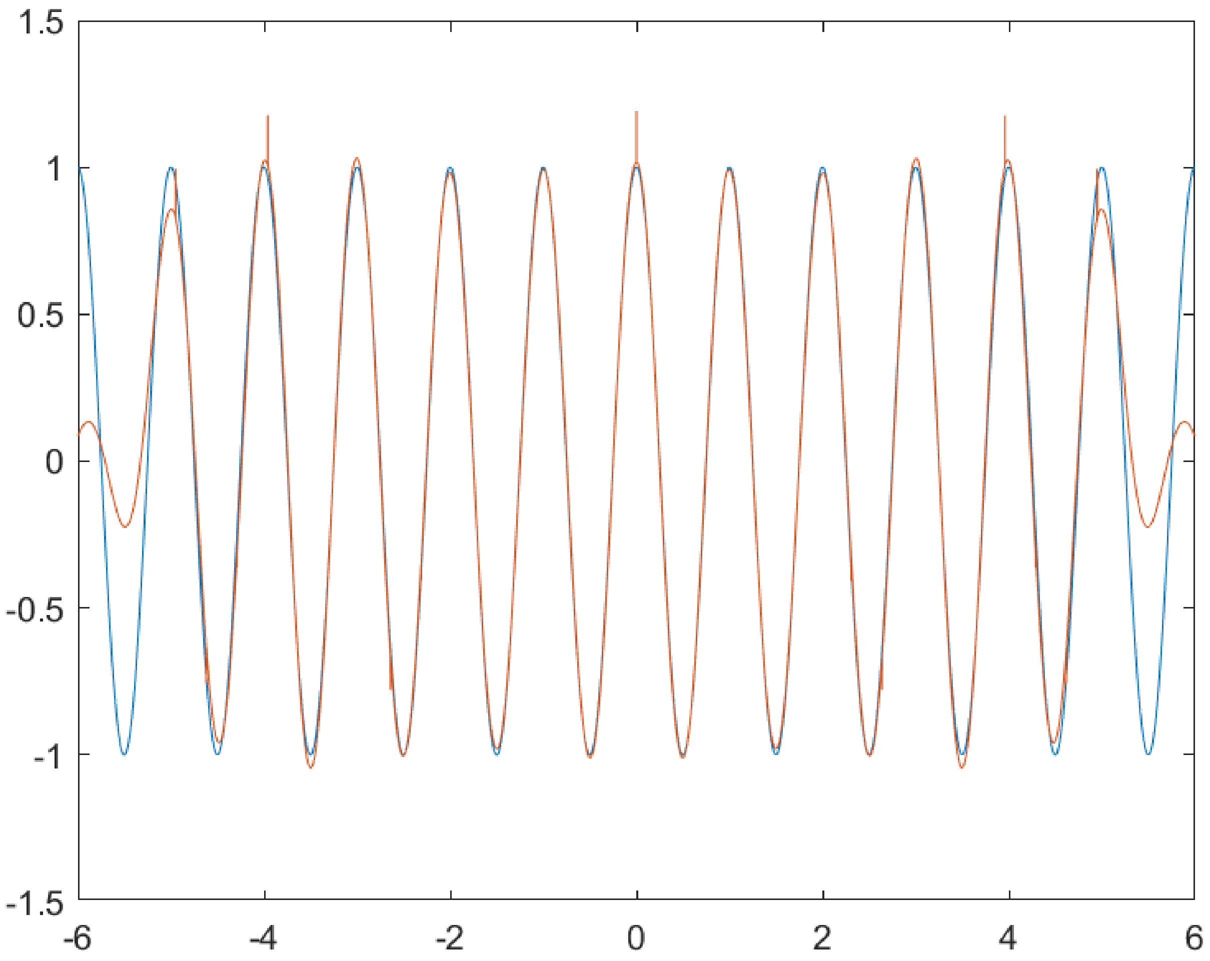
1. Keeping everything else the same as in the previous experiment, increase ωs to 6π

and 8π and compare x(t) and xr(t).

```
function samples=sampled_cosine(N,T)
samples=zeros(1,length(N));
for i = 1:length(N)
   samples(i)=cos(2*pi*N(i)*T);
end
```

```
end
N1=-6/0.4:6/0.4;
saa=sampled_cosine(N1,0.33);
function xr_t=reconstruct( samples,T,wc,N,t);
xr_t=zeros(1,length(t));
scaling=wc*T/pi;
for i=1:length(t)
  current_t=t(i);
  sum=0;
  for j=1:length(samples)
    arg=wc*(current_t-N(j)*T);
    if arg==0
      sum=sum+samples(j);
    else
      sum=sum+samples(j)*scaling*sin(arg)/arg;
    end
  end
  xr_t(i)=sum;
end
end
t=-6:0.001:6;
rec1=reconstruct(saa,0.33,2.5*pi,N1,t);
```

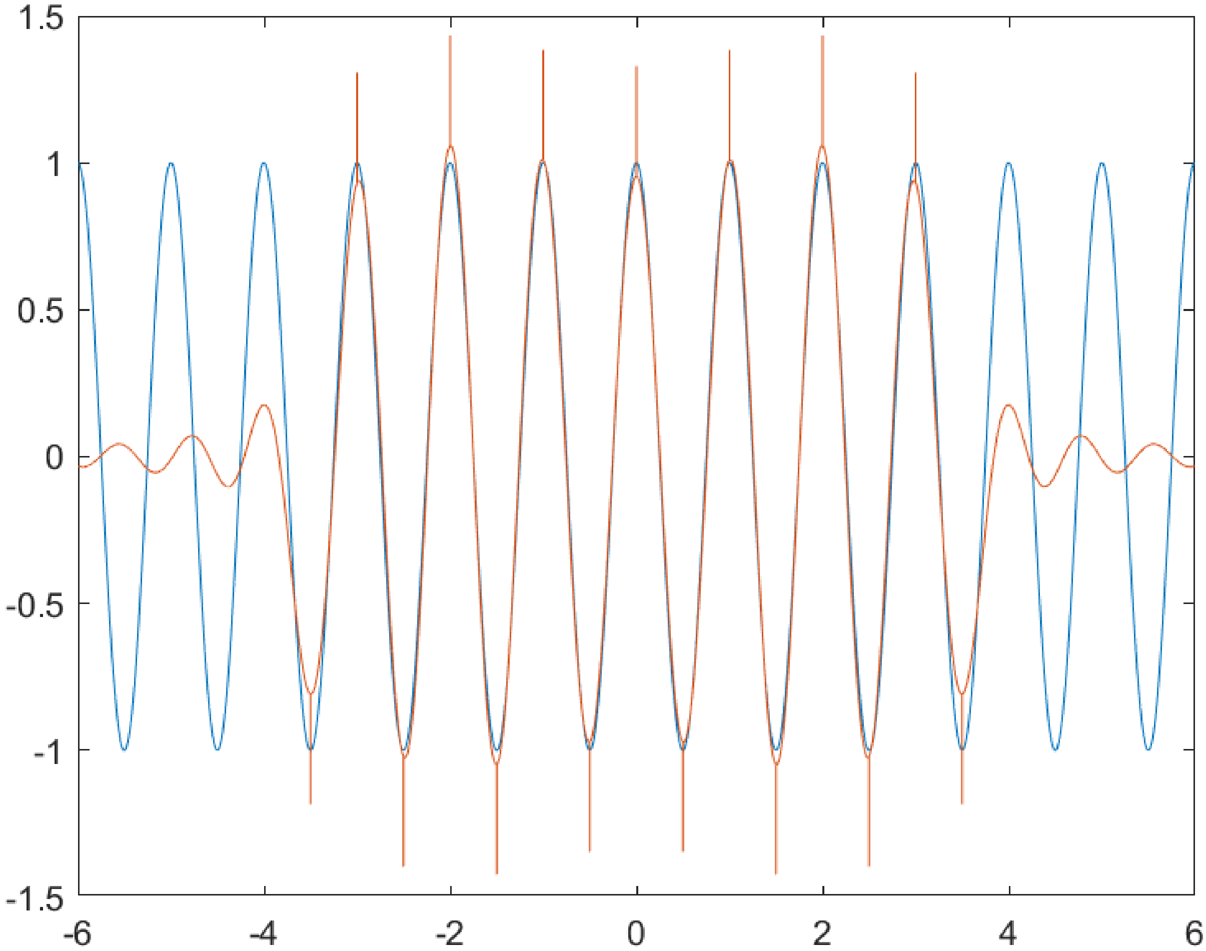
```
for p=1:length(N1)
  disp(rec1(p));
end
plot(t,cos(2*pi*t));
hold on;
plot(t,rec1);
```




```
function samples=sampled_cosine(N,T)
samples=zeros(1,length(N));
for i = 1:length(N)
```

```
    samples(i)=cos(2*pi*N(i)*T);
end
end
N1=-6/0.4:6/0.4;
saa=sampled_cosine(N1,0.25);
function xr_t=reconstruct( samples,T,wc,N,t);
xr_t=zeros(1,length(t));
scaling=wc*T/pi;
for i=1:length(t)
    current_t=t(i);
    sum=0;
    for j=1:length(samples)
        arg=wc*(current_t-N(j)*T);
        if arg==0
            sum=sum+samples(j);
        else
            sum=sum+samples(j)*scaling*sin(arg)/arg;
        end
    end
    xr_t(i)=sum;
end
end
```

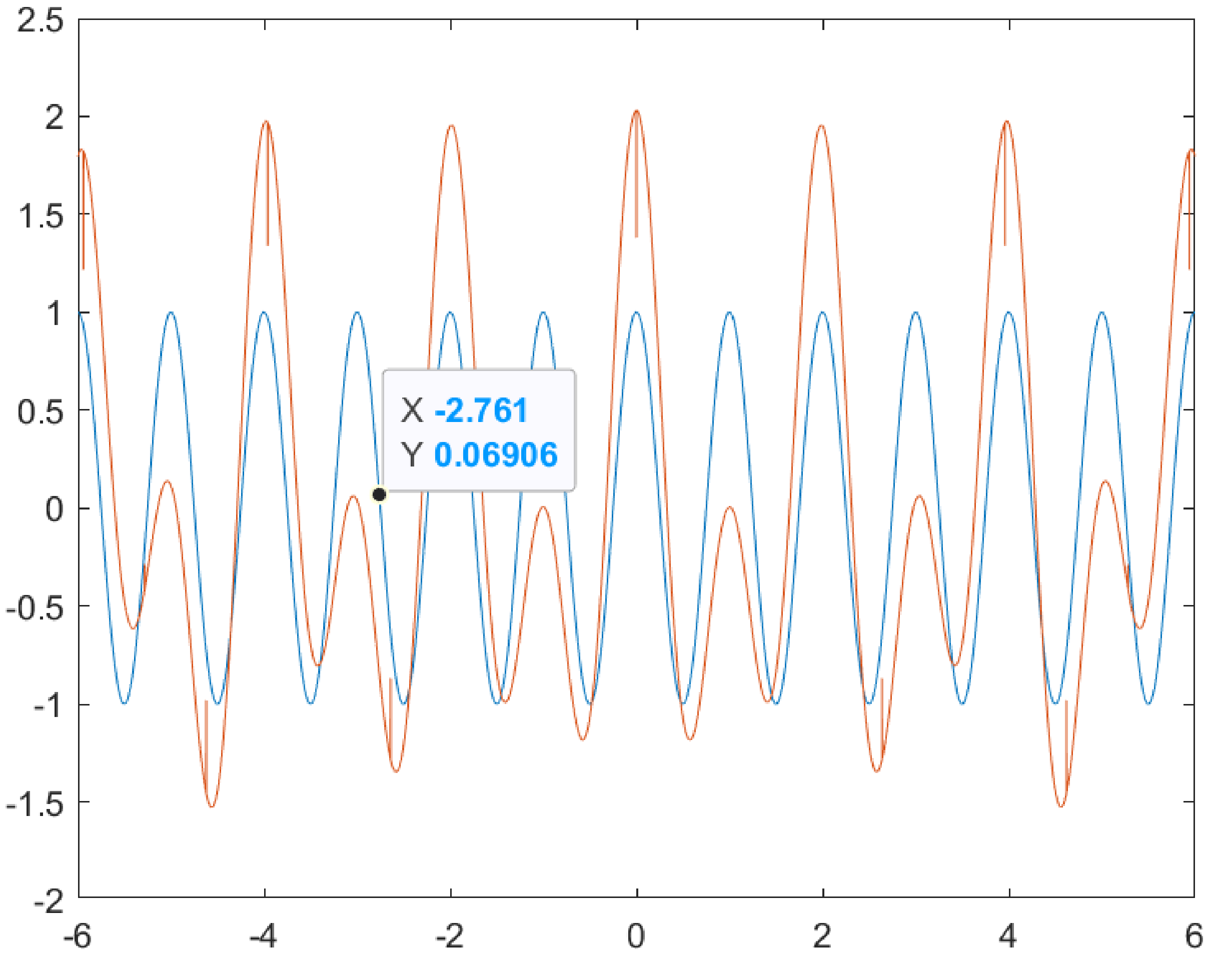
```
t=-6:0.001:6;
rec1=reconstruct(saa,0.25,2.5*pi,N1,t);
for p=1:length(N1)
  disp(rec1(p));
end
plot(t,cos(2*pi*t));
hold on;
plot(t,rec1);
```



2.Now decrease ωs to 3π and compare x(t) and xr(t). Are they the same?

```
function samples=sampled_cosine(N,T)
samples=zeros(1,length(N));
for i = 1:length(N)
   samples(i)=cos(2*pi*N(i)*T);
end
end
N1=-6/0.4:6/0.4;
saa=sampled_cosine(N1,0.66);
function xr_t=reconstruct( samples,T,wc,N,t);
xr_t=zeros(1,length(t));
scaling=wc*T/pi;
for i=1:length(t)
   current_t=t(i);
   sum=0;
   for j=1:length(samples)
      arg=wc*(current_t-N(j)*T);
      if arg==0
         sum=sum+samples(j);
      else
         sum=sum+samples(j)*scaling*sin(arg)/arg;
      end
   end
```

```
   xr_t(i)=sum;
end
end
t=-6:0.001:6;
rec1=reconstruct(saa,0.66,2.5*pi,N1,t);
for p=1:length(N1)
  disp(rec1(p));
end
plot(t,cos(2*pi*t));
hold on;
plot(t,rec1);
```



SAMPLING A NON-BANDLIMITED SIGNALS

1. Write a function sampled rect(), whose input is a vector containing the values of n

and the sampling interval T. The output is the sequence of samples x(nT), where

x(t) = (

1, 1 ≤ t ≤ 2

0, otherwise.

```
function sample1=sampled_rect(N,T)
sample1=zeros(1,length(N));
```

```
for i=1:length(sample1)
   t=N(i)*T;
   if t<=2 && t>=1
      sample1(i)=1;
   else
      sample1(i)=0;
   end
end
end
```

2. What is the Nyquist rate for x(t)? Justify your answer.

The Nyquist Rate is 2Hz.

3. Using the same procedure as outlined in the previous exercises, sample x(t) over

the range 0 ≤ t ≤ 3 with sampling frequencies ωs = 2π/T for T = 0.03, T = 0.01,

and T = 0.005.

4. In each case, compare x(t) and xr(t) by plotting them against t, where

t = 0 : 0.001 : 3.

```
t_range=0:0.001:3;
T_val=[0.03,0.01,0.005];
for i=T_val
   n=0:i:3;
```

```
    sample2=sampled_rect(n,i);
    plot(n,sample2);
    hold on;
    t_cont=(t_range<=2 && t_range>=1);
    plot(t_range,t_cont);
end
```