EEE 321 LAB5 OFFLAB ASSIGNMENT REPORT

Name/Surname: Muhammet Melih GELİK

ID: 22003836

PART 1: Basic Operations

The function g(t) was plotted analytically in Figure 1 below:

$$g(t) = \begin{cases} 2t + 1, & 0 \le t < 0.5 \\ \dfrac{5 - 2t}{2}, & 0.5 \le t < 1 \\ \dfrac{3}{2}, & 1 \le t \le 2 \\ 0, & otherwise \end{cases}$$

The critical points of the function was indicated in the figure as well. The function g(t) is a discontinuous function at the points (0,0) and (2,0).

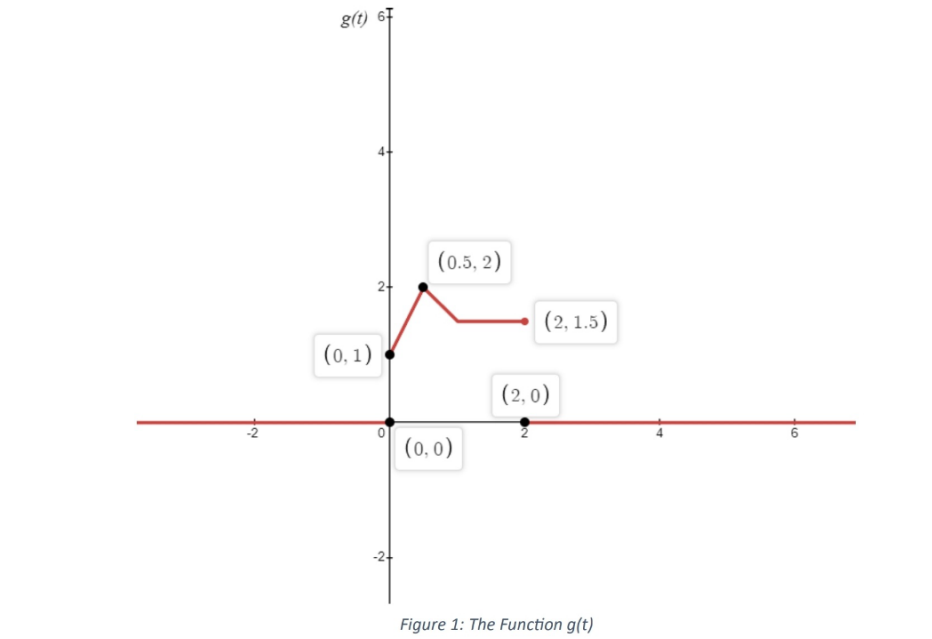

Figure 1: The Function g(t)

The function $g(t)$ can be rewritten by the basic ramp and unit step function as below:

$$g(t) = (u(t) - u(t-1))\left((2r(t)+1) - 3r\left(t - \frac{1}{2}\right)\right) + \frac{3}{2}(u(x-1) - u(x-2))$$

To determine the recoverability of the signal $g(t)$ with the sampling frequency $F_s = \frac{1}{T_s}$, we need to first evaluate its CTFT as $G(j\omega)$

$$G(j\omega) = \int_{-\infty}^{+\infty} g(t)e^{-j\omega t}dt$$

$$= \int_0^{0.5} (2t+1)e^{-j\omega t}dt - \int_{0.5}^{1}\left(3t - \frac{3}{2}\right)e^{-j\omega t}dt + \int_1^2 e^{-j\omega t}dt$$

$$= 2\int_0^{0.5} te^{-j\omega t}dt + \int_0^{0.5} e^{-j\omega t}dt - \left[3\int_{0.5}^{1} te^{-j\omega t}dt - \frac{3}{2}\int_{0.5}^{1} e^{-j\omega t}dt\right] + \int_1^2 e^{-j\omega t}dt$$

$$\int_a^b te^{-j\omega t}dt = -\frac{1}{j\omega}\left[te^{-j\omega t}\right]_a^b - \int_a^b e^{-j\omega t}dt$$

$$= -\frac{1}{j\omega}\left[be^{-j\omega b} - ae^{-j\omega a}\right] + \frac{1}{j\omega}\left[e^{-j\omega b} - e^{-j\omega a}\right]$$

$$G(j\omega) = 2I_1 + I_2 - 3I_3 + \frac{3}{2}I_4 + I_5$$

$$2I_1 = 2\left[-\frac{1}{j\omega}\frac{1}{2}e^{-\frac{j\omega}{2}} + \frac{1}{j\omega}\left(e^{-\frac{j\omega}{2}} - 1\right)\right] = \frac{1}{j\omega}\left(e^{-\frac{j\omega}{2}} - 2\right)$$

$$I_2 = \frac{1}{j\omega}\left(e^{-\frac{j\omega}{2}} - 1\right)$$

$$3I_3 = 3\left[-\frac{1}{j\omega}\left(e^{-j\omega} - \frac{1}{2}e^{-\frac{j\omega}{2}}\right) + \frac{1}{j\omega}\left(e^{-j\omega} - e^{-\frac{j\omega}{2}}\right)\right] = \frac{1}{j\omega}\left(-\frac{3}{2}e^{-\frac{j\omega}{2}}\right)$$

$$\frac{3}{2}I_4 = \frac{3}{2}\left[\frac{1}{j\omega}\left(e^{-j\omega} - e^{-\frac{j\omega}{2}}\right)\right] = \frac{1}{j\omega}\left(\frac{3}{2}e^{-j\omega} - \frac{3}{2}e^{-\frac{j\omega}{2}}\right)$$

$$I_5 = \frac{1}{j\omega}\left(e^{-j2\omega} - e^{-j\omega}\right)$$

$$G(j\omega) = \frac{1}{j\omega}\left[\left(e^{-\frac{j\omega}{2}} - 2\right) + \left(e^{-\frac{j\omega}{2}} - 1\right) + \frac{3}{2}e^{-\frac{j\omega}{2}} + \left(\frac{3}{2}e^{-j\omega} - \frac{3}{2}e^{-\frac{j\omega}{2}}\right) + \left(e^{-j2\omega} - e^{-j\omega}\right)\right]$$

$$= \frac{1}{j\omega}\left[2e^{-\frac{j\omega}{2}} - 3 + \frac{1}{2}e^{-j\omega} + e^{-j2\omega}\right]$$

As the CTFT of the signal $g(t)$ is not band limited, it is not possible to recover the signal $g(t)$ from the sampled one.

PART 2: Sampling and Interpolation Basics

The Sampling operation was given below analytically:

$$\tilde{x}(t) = x_p(t) = x(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

a) Analyzing the system:

i- For linearity:

$$y_1(t) = x_{p1}(t) = x_1(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

and

$$y_2(t) = x_{p2}(t) = x_2(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

Let the input $x_3(t)$ be defined as follows:

$$x_3(t) = x_1(t) + x_2(t)$$

Then, the output of this input is:

$$y_3(t) = x_{p3}(t) = x_3(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

Hence, the system is linear as:

$$y_1(t) + y_2(t) = [x_1(t) + x_2(t)] \sum_{n=-\infty}^{+\infty} \delta(t - nT_s) = y_3(t) = x_3(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

ii- For time invariancy, we can use $x_1(t)$ as input again and this time $x_2(t)$ should be defined as time shifted version of the first input:

$$y_1(t) = x_{p1}(t) = x_1(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

and

$$x_2(t) = x_1(t - t_0)$$

$$y_2(t) = x_{p2}(t) = x_2(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

If we shift the first output now, we obtain:

$$y_1(t - t_0) = x_{p1}(t - t_0) = x_1(t - t_0) \sum_{n=-\infty}^{+\infty} \delta(t - t_0 - nT_s)$$

$$x_1(t - t_0) \sum_{n=-\infty}^{+\infty} \delta(t - t_0 - nT_s) \neq x_2(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s) = x_1(t - t_0) \sum_{n=-\infty}^{+\infty} \delta(t - nT_s)$$

Hence, the system is not time-invariant. It is time-variant system.

b) The Fourier relation between x(t) and sampled signal:

Let's the Fourrer representation of x(t) is the function given in Figure 2 with analytic expression:

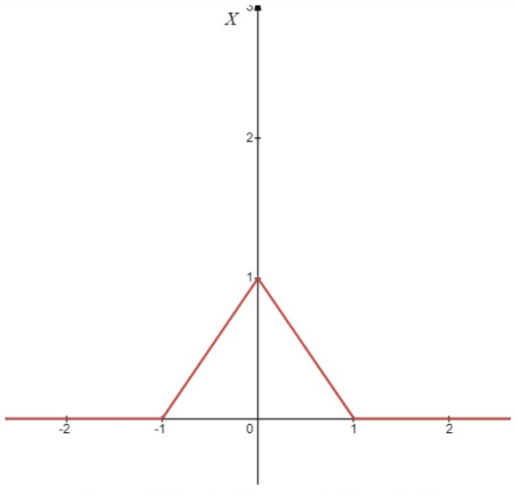

Figure 2: Fourier Representation of x(t)

$$X(j\omega) = \big(u(\omega+1) - u(\omega)\big)[\omega+1] + \big(u(\omega) - u(\omega-1)\big)[1-\omega]$$

When the impulse train is multiplied with the original x(t), for sampling Fourier transform of the impulse train and Fourier transform of the signal the result is convolution integral of two Fourier representation.

$$\tilde{X}(j\omega) = \frac{1}{2\pi}\big(X(j\omega) * P(j\omega)\big)$$

The impulse function, also has the Fourier representation as following:

$$p(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT_s) \longleftarrow \mathcal{F} \longrightarrow P(j\omega) = \frac{2\pi}{T_s} \sum_{n=-\infty}^{+\infty} \delta\left(\omega - \frac{2\pi n}{T_s}\right)$$

Hence, the periodic convolution is:

$$\tilde{X}(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \left[\frac{2\pi}{T_s} \sum_{n=-\infty}^{+\infty} \delta\left(\theta - \frac{2\pi n}{T_s}\right)\right] X(j(\omega - \theta)) d\theta$$

$$= \frac{1}{T_s} \sum_{n=-\infty}^{+\infty} X\left(j\left(\omega - \frac{2\pi n}{T_s}\right)\right) \int_{-\infty}^{+\infty} \delta\left(\theta - \frac{2\pi n}{T_s}\right) d\theta$$

$$\tilde{X}(j\omega) = \frac{1}{T_s} \sum_{n=-\infty}^{+\infty} X\left(j\left(\omega - \frac{2\pi n}{T_s}\right)\right)$$

So, the sampled signal Fourier representation contains Fourier transform of the original signal shifted to $\frac{2\pi n}{T_s}$, $n \in \mathbb{Z}$. the Fourier representation of the sampled signal was given in Figure3.

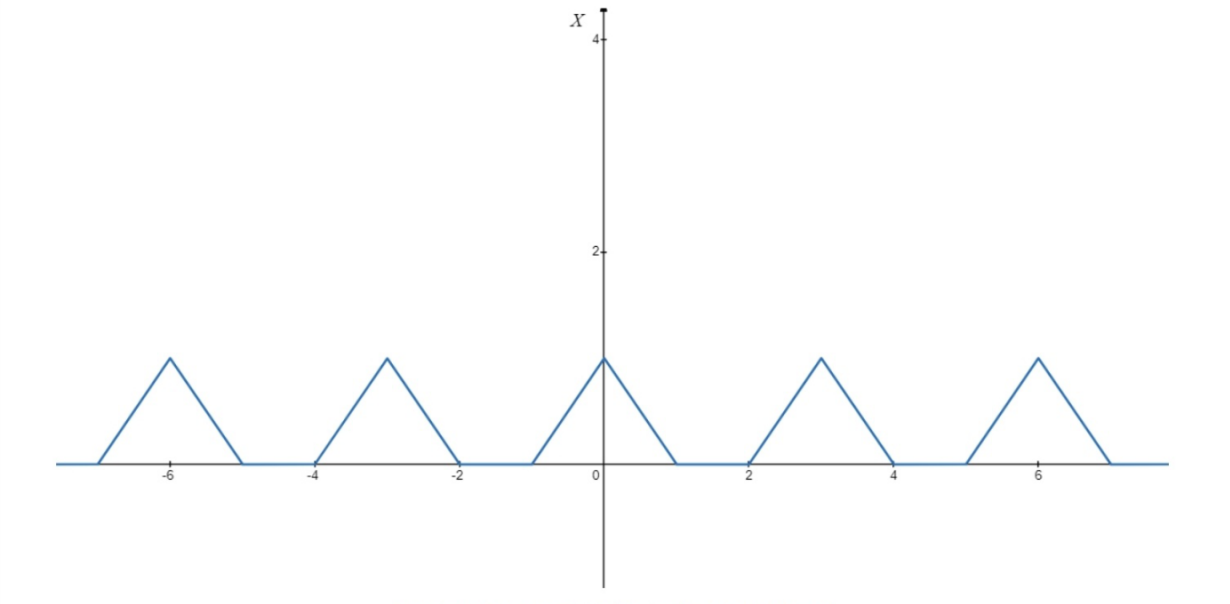

Figure 3: Fourier Transform of Sampled Signal

c) For consistency of the sampling, the expression of recovered signal was given as:

$$x_R(t) = \sum_{n'=-\infty}^{+\infty} x[n']p(t - n'T_s)$$

$$\text{for } t = nT_s, x_R(t) = \sum_{n'=-\infty}^{+\infty} x[n']p(nT_s - n'T_s)$$

$$\text{and if } p(t) = \begin{cases} 1, & t = nT_s; n \in \mathbb{Z} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{then, } \sum_{n'=-\infty}^{+\infty} x[n']p(nT_s - n'T_s) = \begin{cases} x[n], & n = n' \\ 0, & \text{otherwise} \end{cases}$$

Therefore, the recovered signal become consistent.

d) The values of $p_z(t)$, $p_L(t)$ and $p_I(t)$:

i- $t=0$; $p_z(0)=1$, $p_L(0)=1$, $p_I(0)=1$.

ii- $t = kT_s$, $k \in Z - \{0\}$; $p_z(t)=0$, $p_L(t)=0$, $p_I(t)=0$

Hence, the interpolations performed by using $p_z(t)$, $p_L(t)$ and $p_I(t)$ are consisted.

PART 3: Generation of Interpolating Functions in MATLAB

The Interpolation functions were given in figures 4, 5, 6.

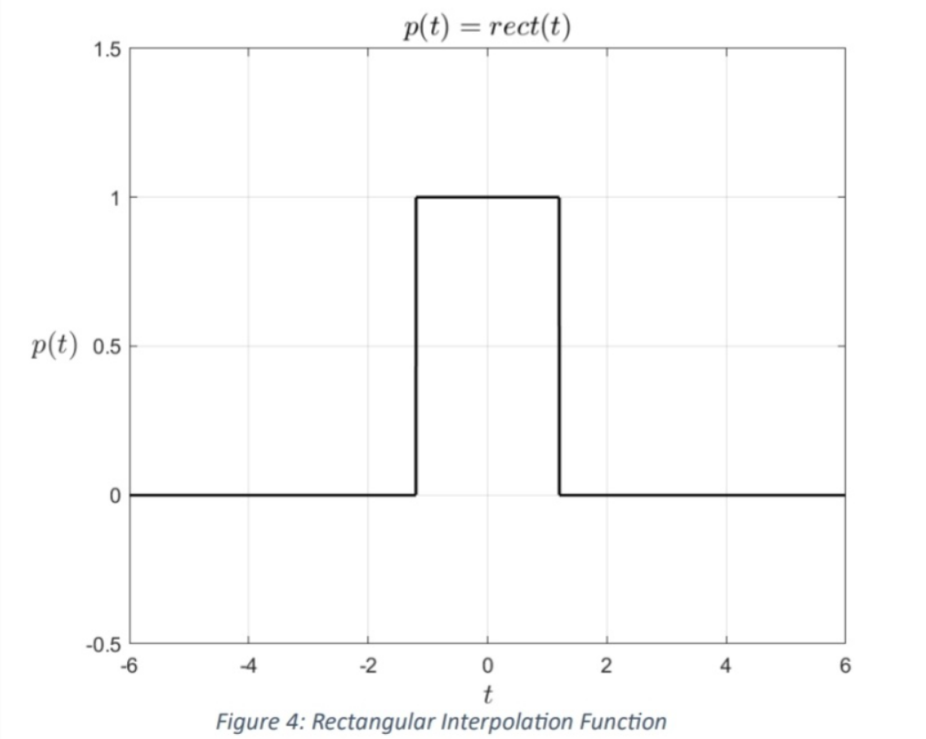

Figure 4: Rectangular Interpolation Function

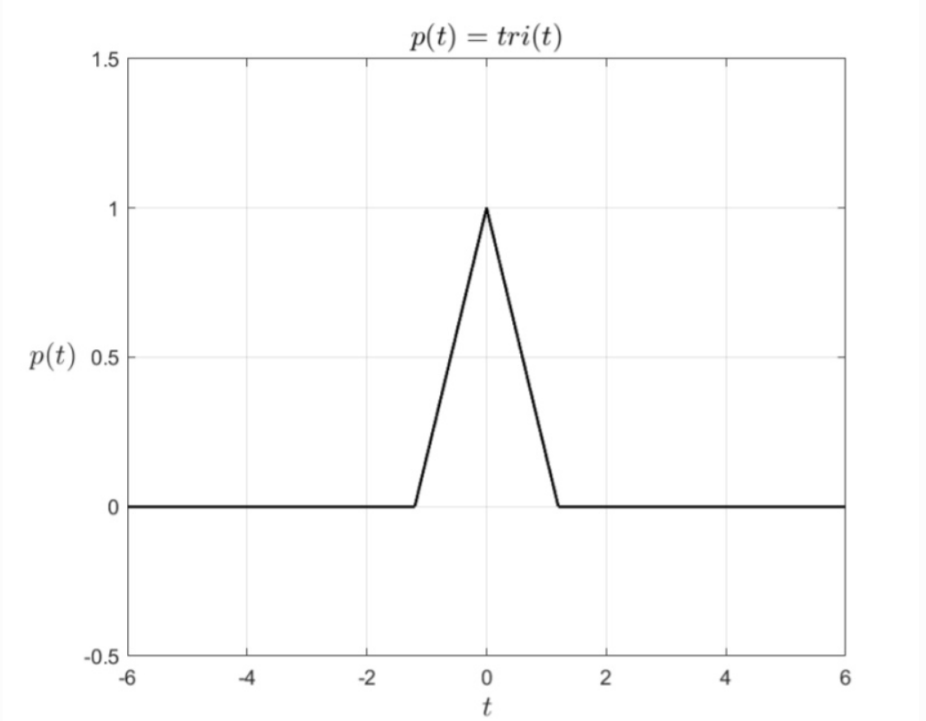

Figure 5: Triangular Interpolation Function

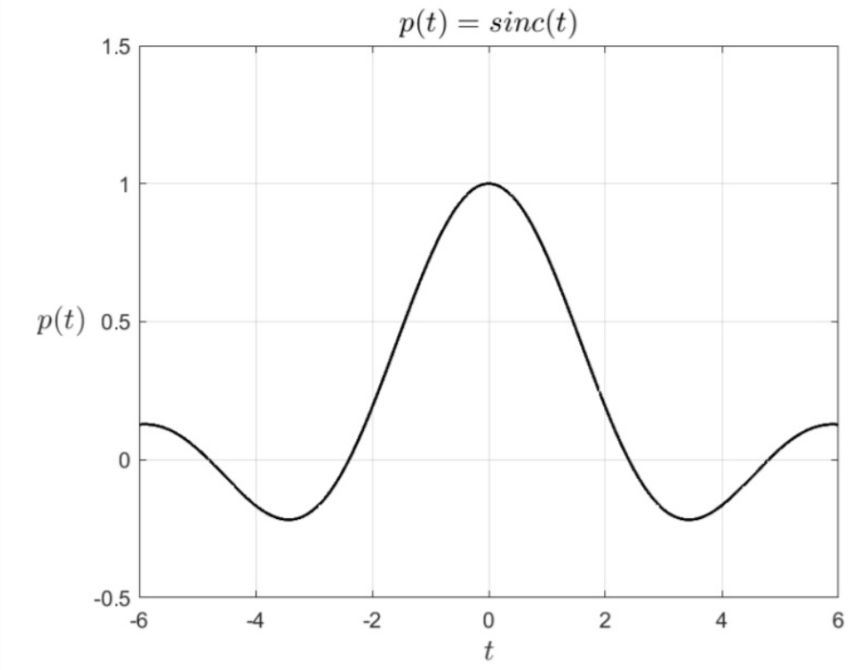

Figure 6: Sinc Interpolation Function

```
%The interpolating signal generator function.
function [p] = generateInterp(type,Ts,dur)
    if dur > Ts
        samp_int = Ts/1000;
        t = -dur:samp_int:dur;
        p = zeros(1,length(t));

        if type == 0
            t_in = -Ts/2:samp_int:Ts/2;
            rect = ones(1,length(t_in));
            bound = (dur-Ts/2)/samp_int;
            p(1,bound+1:bound+length(rect)) = rect;

        elseif type == 1
            t_in1 = -Ts:samp_int:0;
            t_in2 =   0+samp_int:samp_int:Ts;
            trian1 = 1 + t_in1/Ts;
            trian2 = 1 - t_in2/Ts;
            trian  = [trian1 trian2];
            bound = (dur-Ts)/samp_int;
            p(1,bound+1:bound+length(trian)) = trian;

        elseif type == 2
            t_1 = -dur:samp_int:0-samp_int;
            t_2 = samp_int:samp_int:dur;
            sinc1 = sin(pi*t_1)./(pi*t_1);
            sinc2 = sin(pi*t_2)./(pi*t_2);
            sinc = [sinc1 1];
            sinc = [sinc sinc2];
            p = sinc;

        end

    else
        samp_int = Ts/1000;
        t = -dur:samp_int:dur;

        if type == 0
            rect = ones(1,length(t));
            p = rect;

        elseif type == 1
            t_in1 = -dur:samp_int:0;
            t_in2 =   0+samp_int:samp_int:dur;
            trian1 = 1 + 2*t_in1;
            trian2 = 1 - 2*t_in2;
            trian  = [trian1 trian2];
            p = trian;

        elseif type == 2
            t_1 = -dur:samp_int:0-samp_int;
            t_2 = samp_int:samp_int:dur;
            sinc1 = sin(pi*t_1)/(pi*t_1);
            sinc2 = sin(pi*t_2)/(pi*t_2);
            sinc = [sinc1 1 sinc2];
            p = sinc;

        end
    end
end
```

The MATLAB code for sampled signal was given:

```
%PART5: MATLAB Simulation for Interpolation

a = 3 + (7 - 3) * rand;
Ts = 1/(10*a);
dur = 6;

t_n = -dur:Ts:dur;
g_n = zeros(1,length(t_n));

for i=1:length(t_n)

    if t_n(1,i) < 0
        g_n(1,i) = 0;

    elseif t_n(1,i) < 0.5
        g_n(1,i) = 2*t_n(1,i) + 1;

    elseif t_n(1,i) < 1
        g_n(1,i) = 2.5 - t_n(1,i);

    elseif t_n(1,i) < 2
        g_n(1,i) = 1.5;

    else
        g_n(1,i) = 0;

    end

end

stem(t_n,g_n,"k");
grid on;
title('$Sampled\,\,Signal$','Interpreter','latex','FontSize',14);
ylabel('$g(nT_s)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);
ylim([-1 4]);

ax = gca;
ax.YLabel.Rotation = 360;
```

The stem plot of the signal was given in Figure 7.

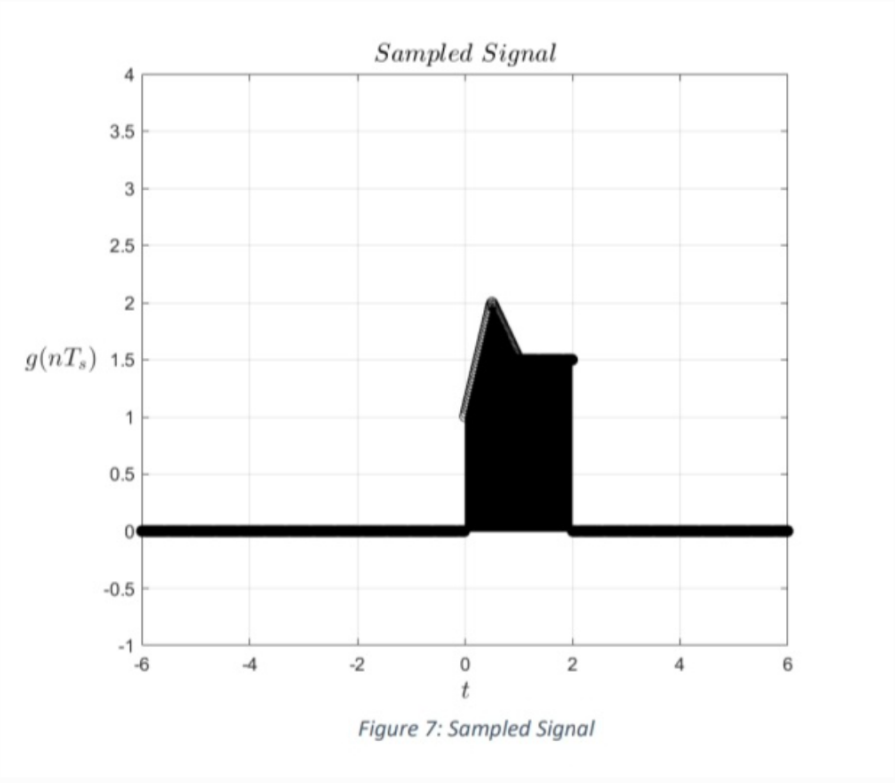

Figure 7: Sampled Signal

PART 4: MATLAB Implementation of Interpolation:

```
%Function generates recovered signal

function [x_r,t_r] = DtoA(type,Ts,dur,Xn)

    p = generateInterp(type,Ts,dur);
    N  = length(Xn);
    len_p = length(p);
    low_b = floor(len_p/2);

    %t = linspace(-dur,dur,length(p));
    %plot(t,p,"k",LineWidth=1.3);

    x_r = zeros(1,10*(N-1)+len_p);

    for i=1:N
        %disp(i)
        x_r(1,1+(i-1)*1000:len_p+(i-1)*1000) ...
      = x_r(1,1+(i-1)*1000:len_p+(i-1)*1000) ...
      + Xn(1,i).*p;

    end

    x_r = x_r(1,low_b+1:low_b+1000*(N-1));
    t_r = linspace(-dur,dur,length(x_r));

end
```

PART 5: MATLAB Simulation for Interpolation

The reconstructed signal plots were given in figures 8, 9, 10.

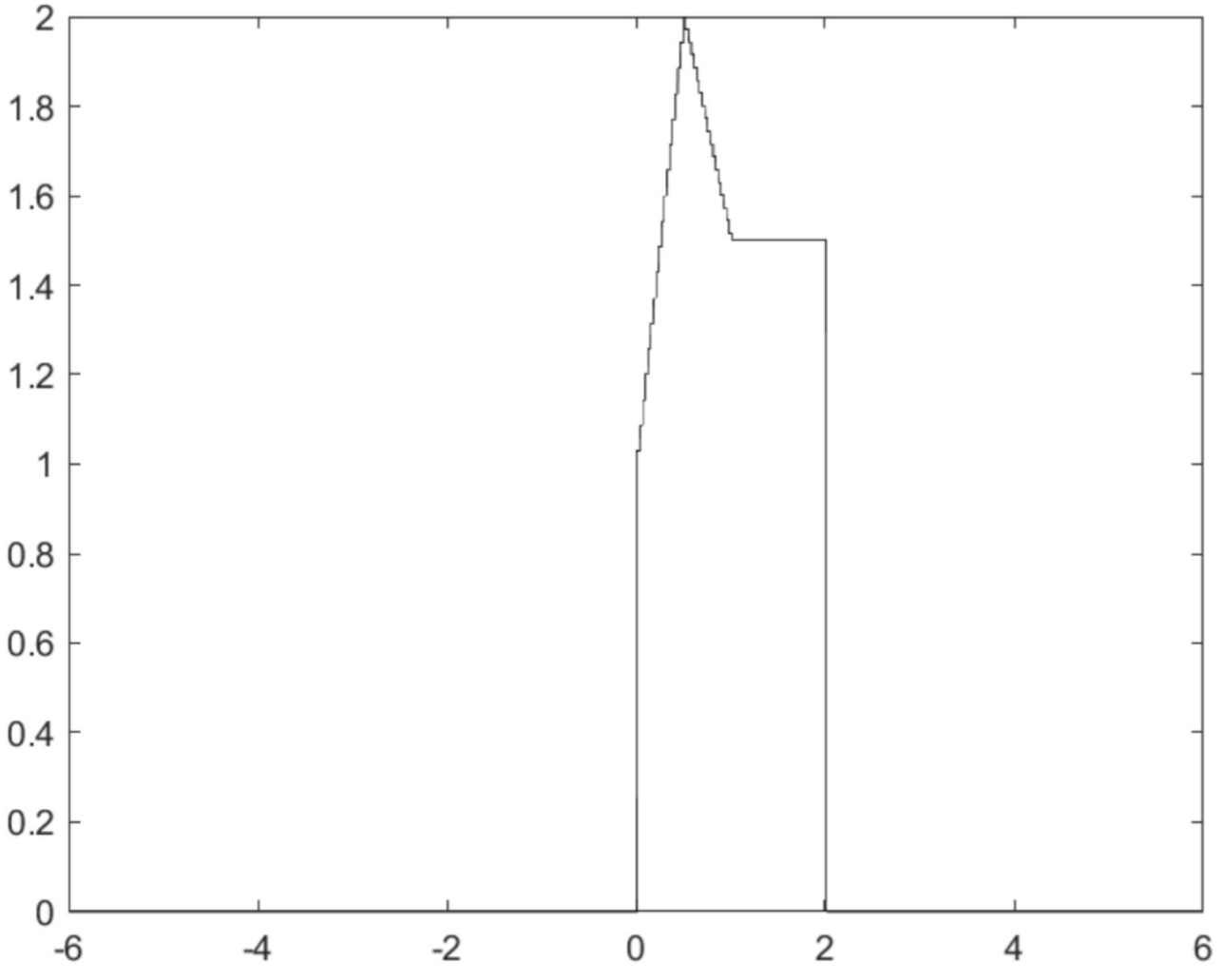

Figure 8: Zero Hold Interpolation

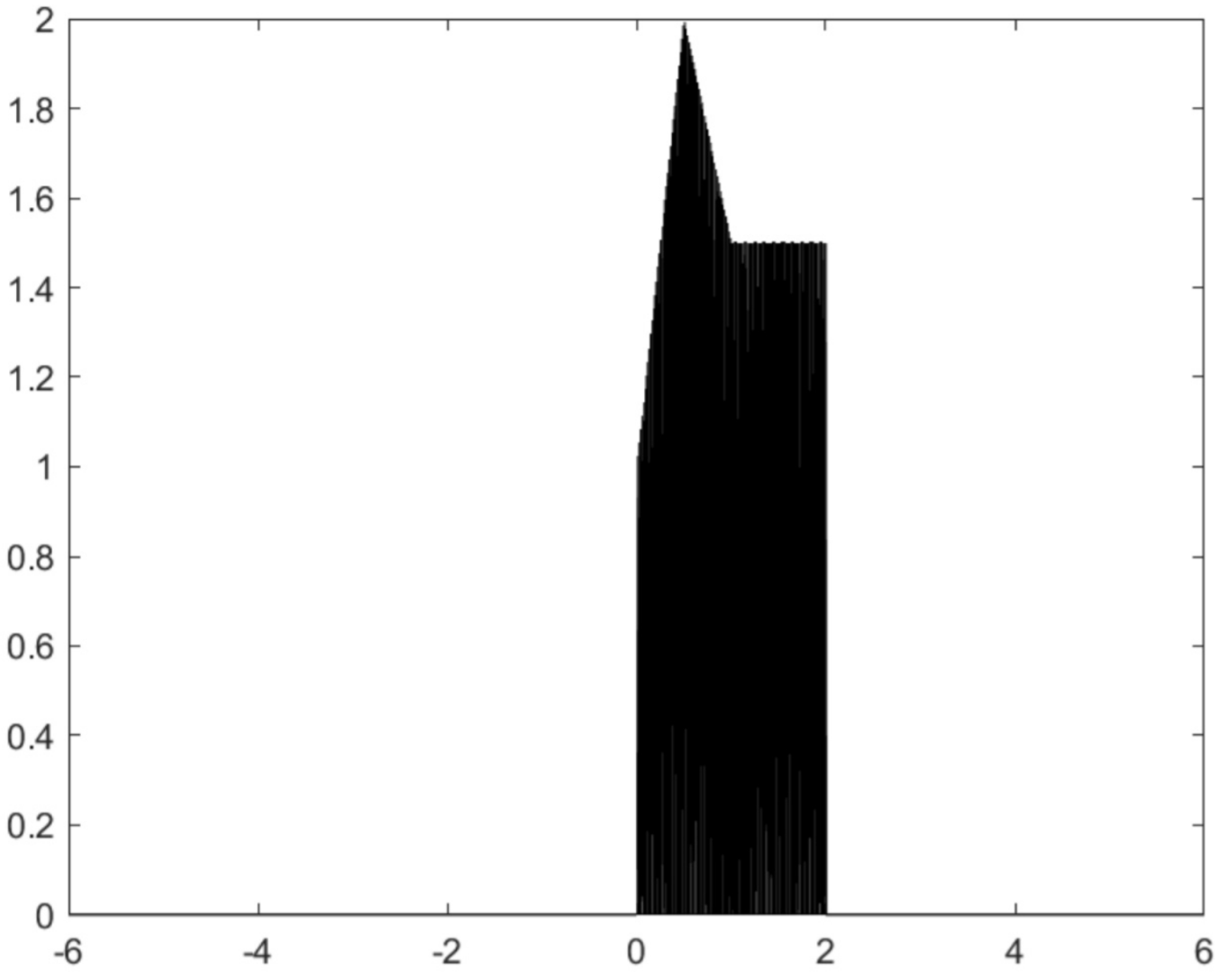

Figure 9: Linear Interpolation

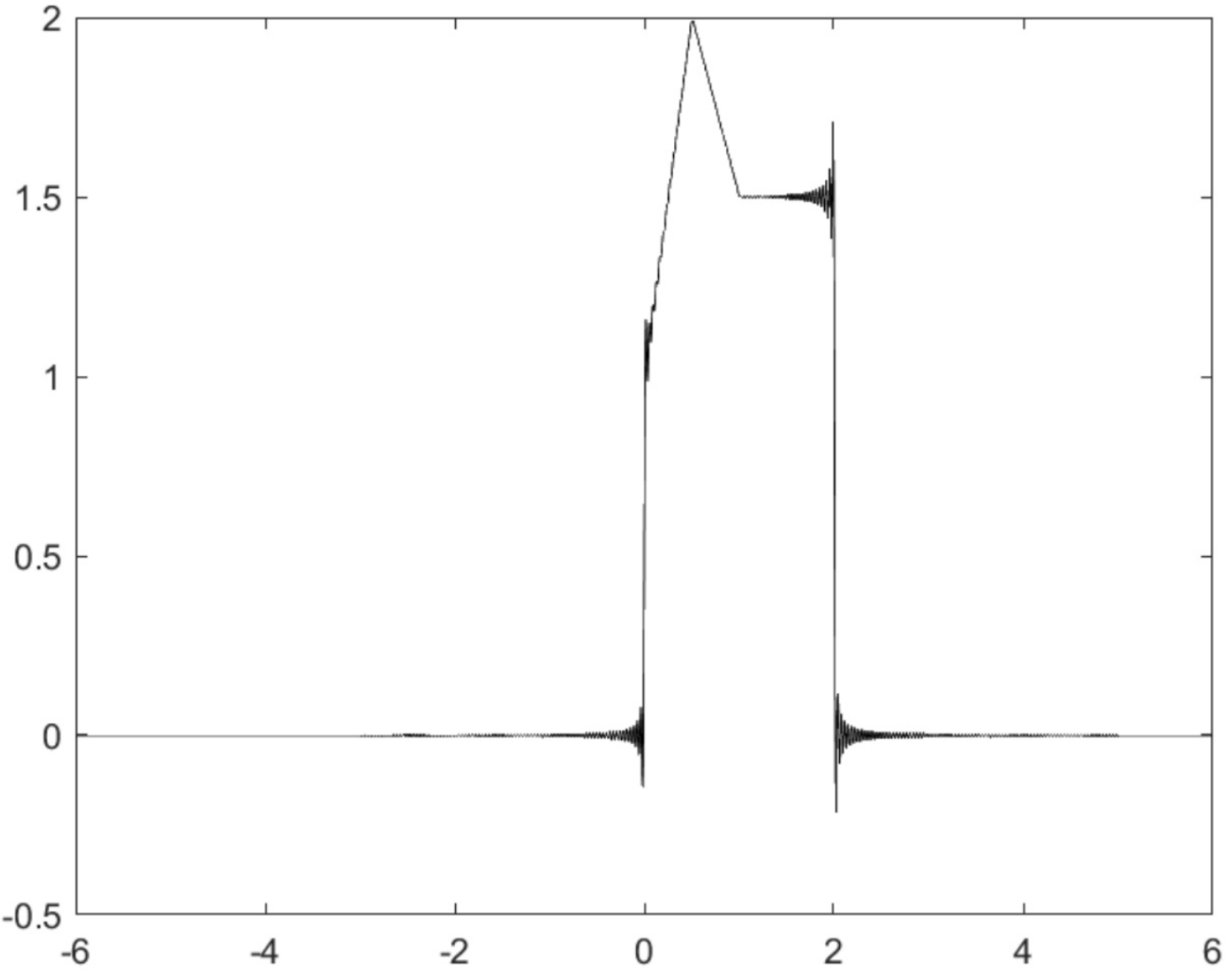

Figure 10: Ideal Interpolation with Sinc Function

When the period of sampling increases, the quality of being identical of the reconstructed signal to the original signal become worse; when we decrease the sampling period, the reconstructed signal starts to approaching to original signal.

PART 6: Reconstruction of a Sum of Cosines

The original signal and different interpolation plots were given in Figure 11.

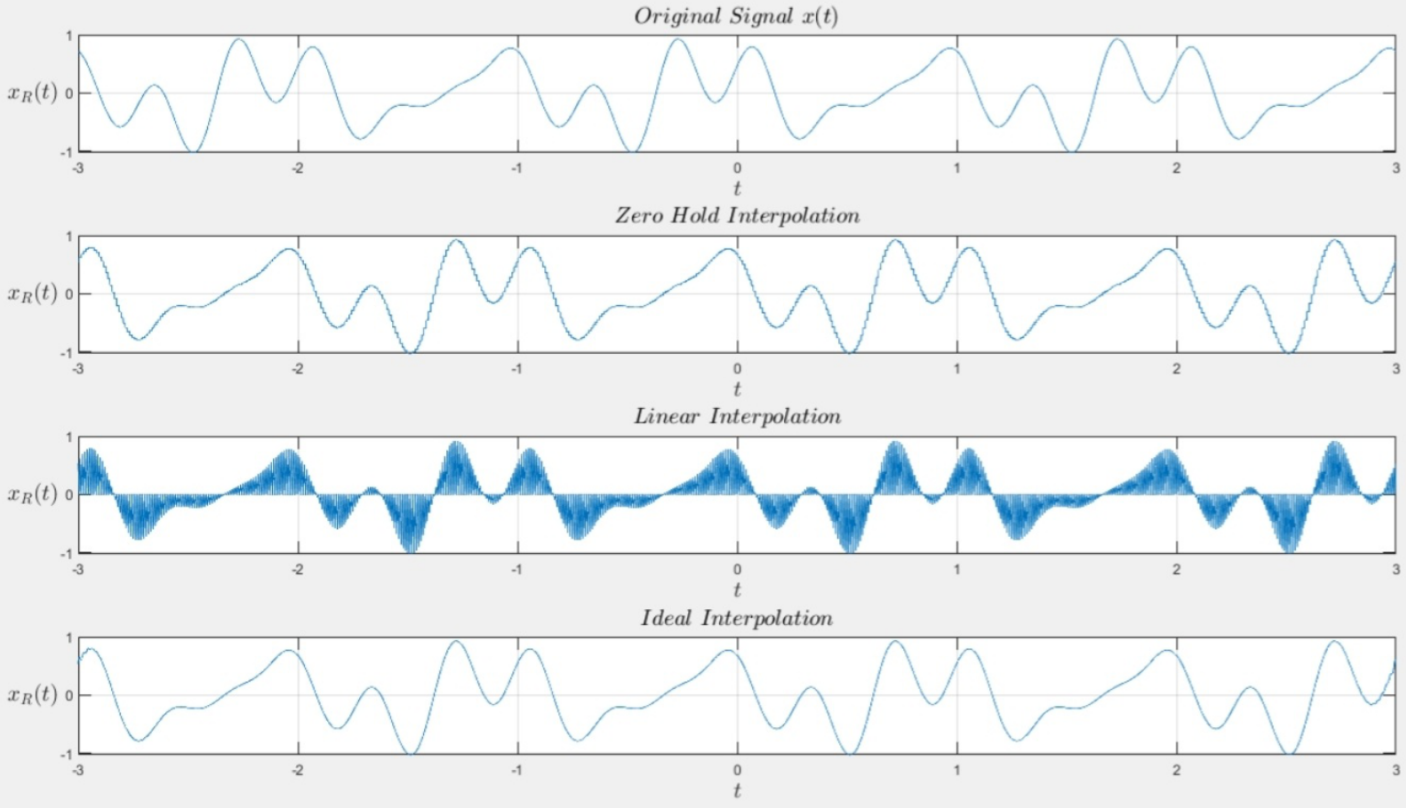

Figure 11: Different Interpolations

When T_s is not in the particular interval that is given in the assignment, the reconstructed signals become deflected. They start not to resemble the original signal. It is because the Fourier Transform starts to alias with itself.

EEE 321 LAB 5 ONLAB ASSIGNMENT REPORT

Name/Surname: Muhammet Melih GELİK

ID: 22003836

Original signal x(t) was plotted as in Figure 12.

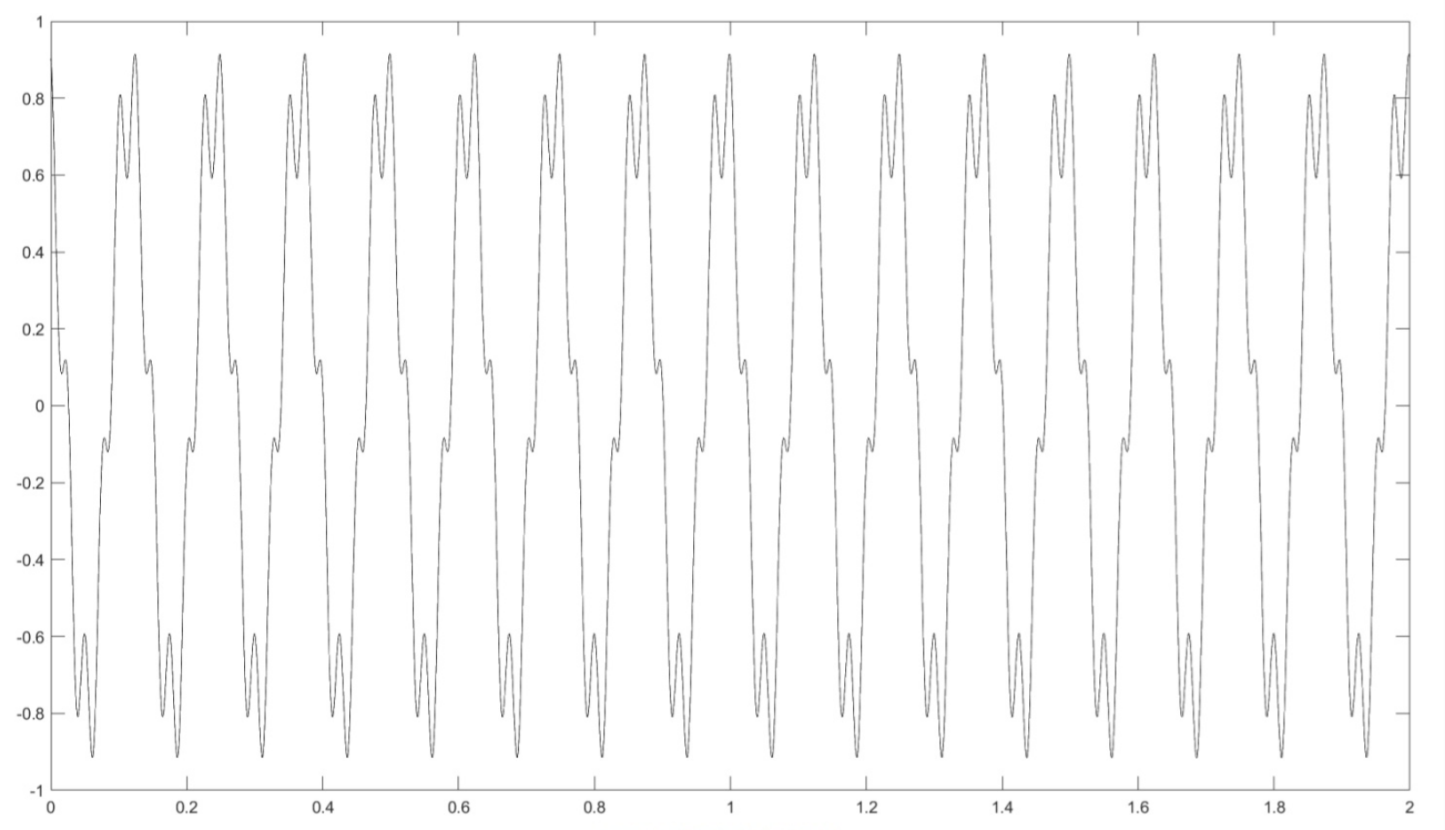

Figure 12: Original Signal x(t)

Discrete signals, $x_{0.5}$, $x_{0.1}$, $x_{0.1}$ were plotted for $\alpha = 1/2$, $\alpha = 1$ and $\alpha = 2$: they are given in Figures 13, 14 and 15 respectively.

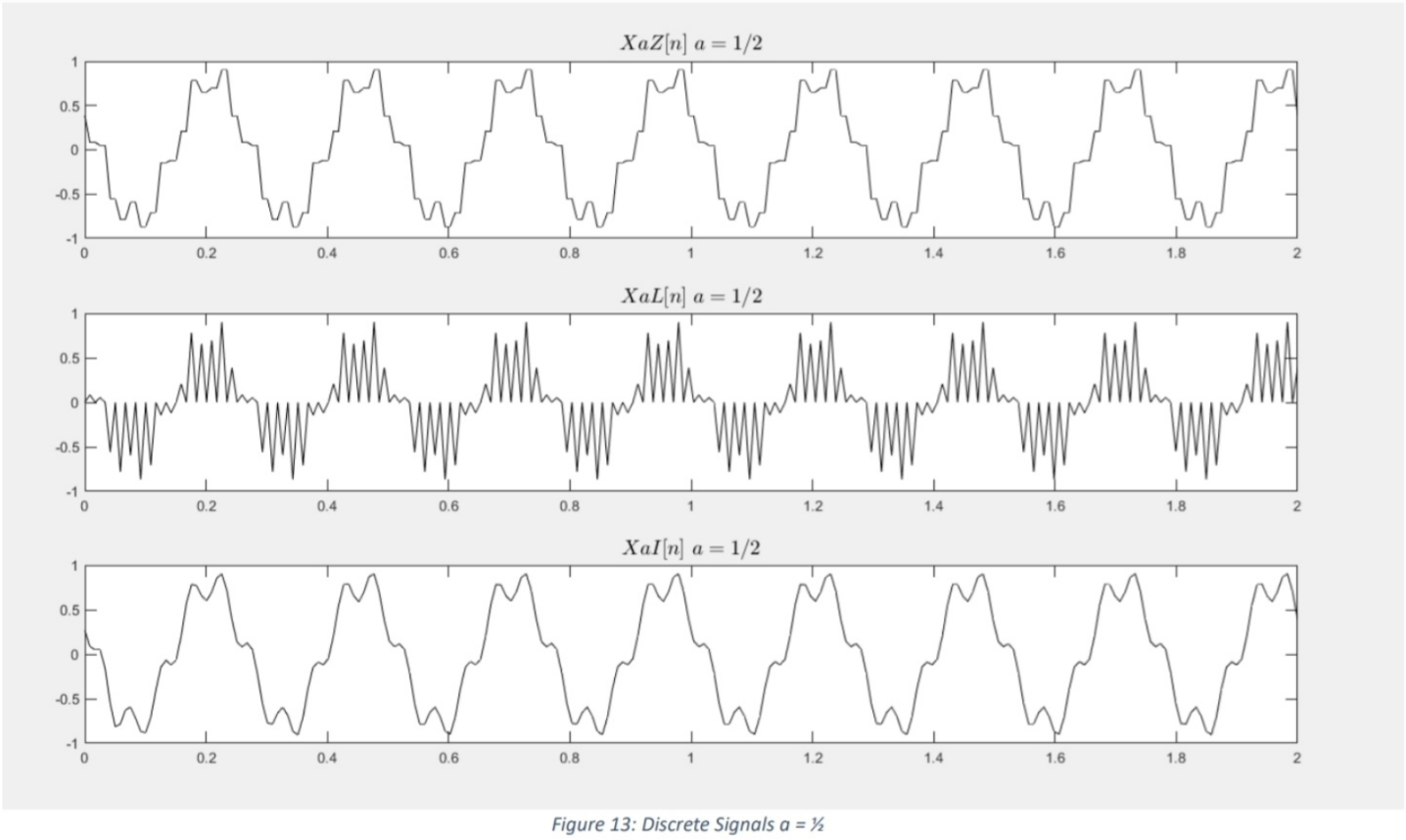

Figure 13: Discrete Signals a = ½

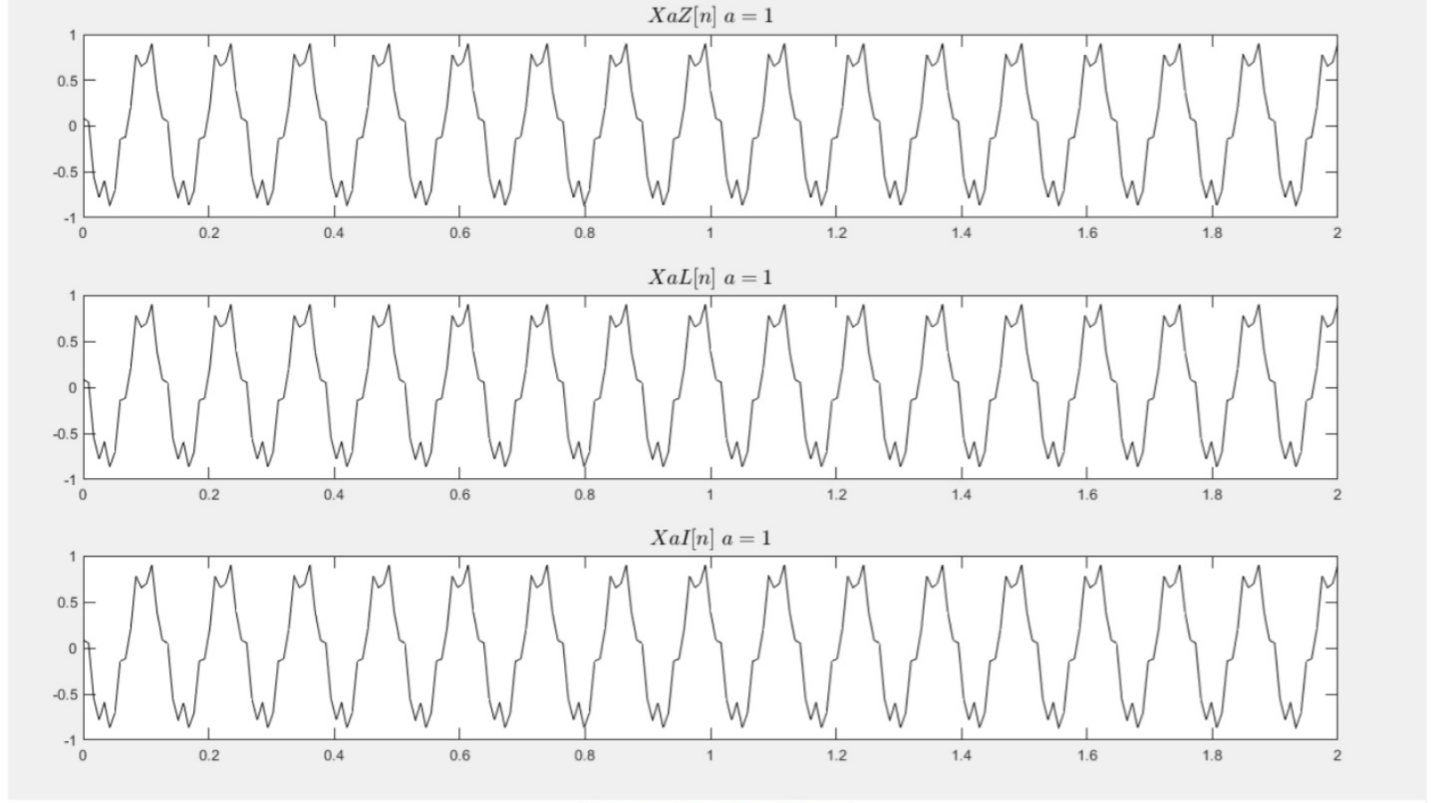

Figure 14: Discrete Signals a = 1

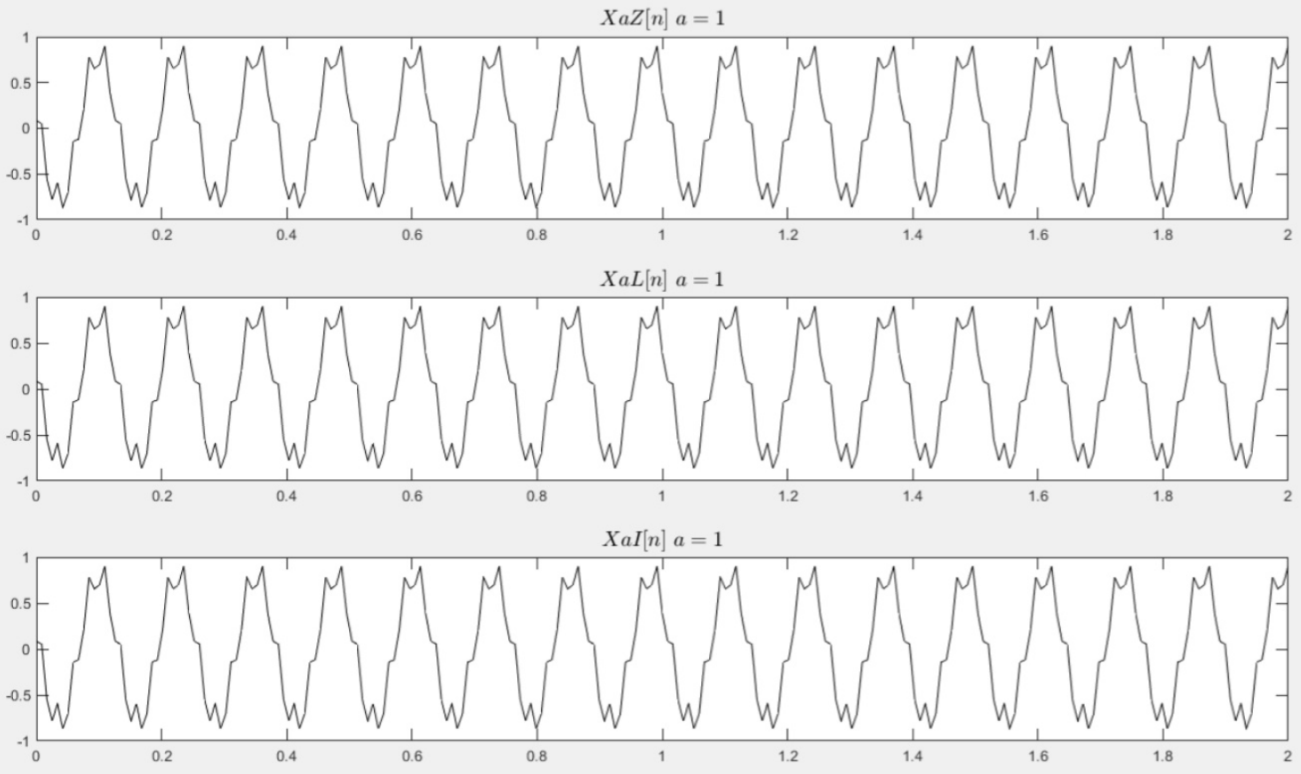

Figure 14: Discrete Signals a = 1

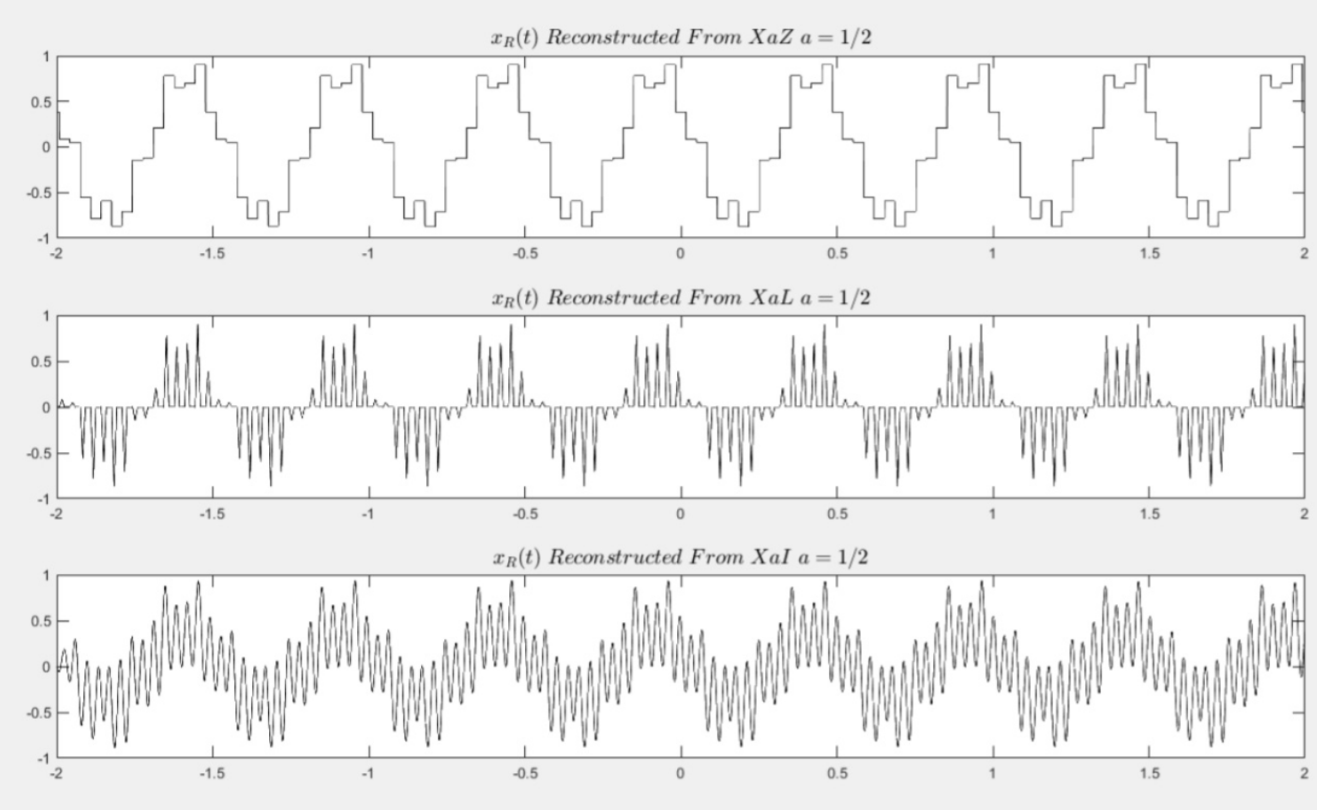

Figure 15: Reconstructed Signals from Discrete Signals

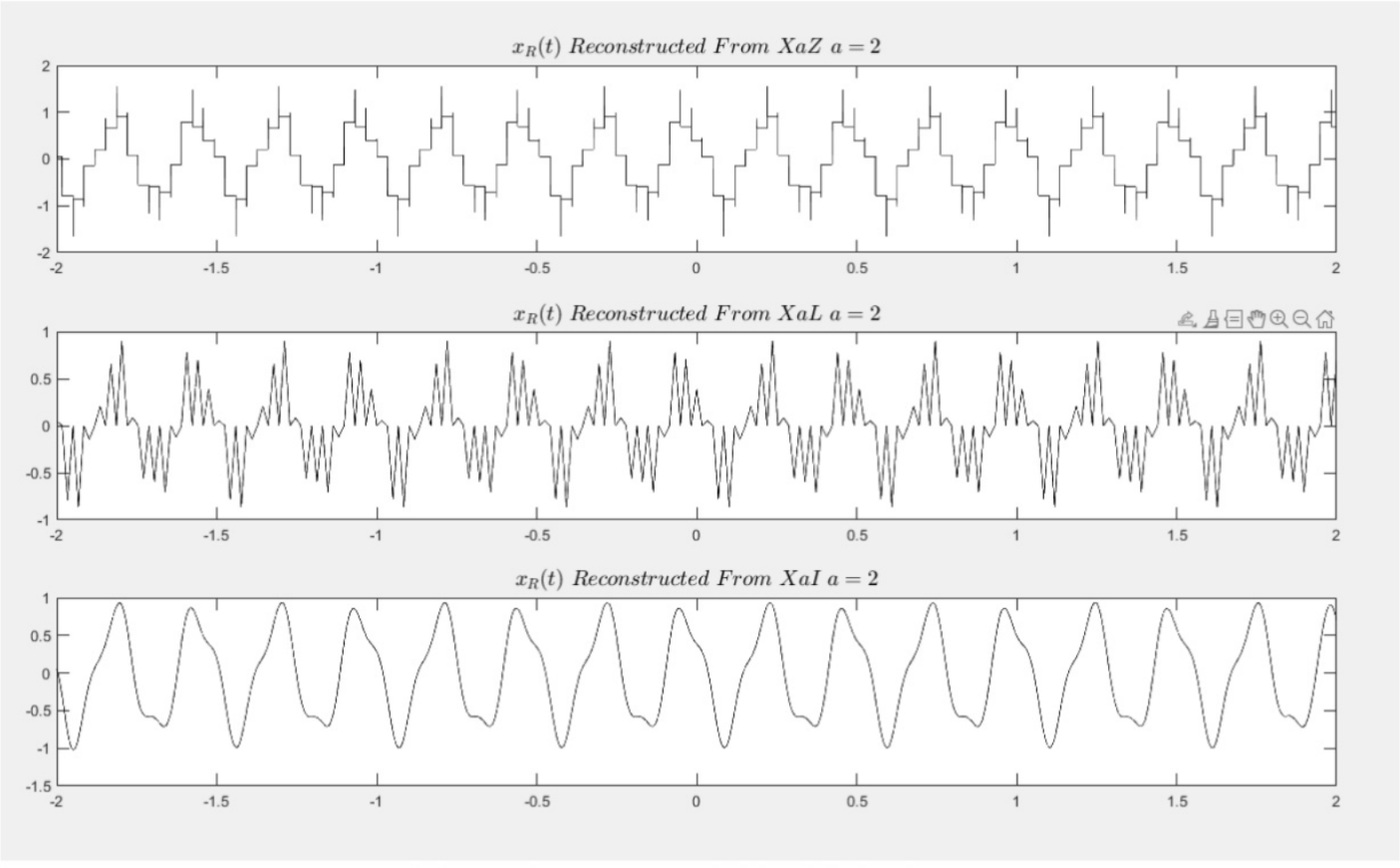

Figure 16: Reconstructed Signals from Discrete Functions a = 2

```
%------------------EEE321 LAB5 OFFLAB CODE---------------------------------
%PART3:Generation of Interpolating Functions in Matlab
ID = 22003836;
dur = mod(ID,10);
Ts = dur/5;

%rectangular interpolation function
p = generateAyberk(0,Ts,dur);
t = linspace(-dur,dur,length(p));

plot(t,p,"k",LineWidth=1.3);
grid on;
ylim([-0.5 1.5]);
title('$p(t) = rect(t)$','Interpreter','latex','FontSize',14);
ylabel('$p(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);

ax = gca;
ax.YLabel.Rotation = 360;

%triangular interpolation function
p = generateAyberk(1,Ts,dur);
t = linspace(-dur,dur,length(p));

plot(t,p,"k",LineWidth=1.3);
grid on;
ylim([-0.5 1.5]);
title('$p(t) = tri(t)$','Interpreter','latex','FontSize',14);
ylabel('$p(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);

ax = gca;
ax.YLabel.Rotation = 360;

%sinc interpolation function
p = generateInterp(2,Ts,dur);
t = linspace(-dur,dur,length(p));

plot(t,p,"k",LineWidth=1.3);
grid on;
ylim([-0.5 1.5]);
title('$p(t) = sinc(t)$','Interpreter','latex','FontSize',14);
ylabel('$p(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);

ax = gca;
ax.YLabel.Rotation = 360;

%PART5: MATLAB Simulation for Interpolation
a = 3;
Ts2 = 1/(10*a);
dur = 6;
```

```
t_n = -dur:Ts2:dur;
g_n = zeros(1,length(t_n));

for i=1:length(t_n)

    if t_n(1,i) < 0
        g_n(1,i) = 0;

    elseif t_n(1,i) < 0.5
        g_n(1,i) = 2*t_n(1,i) + 1;

    elseif t_n(1,i) < 1
        g_n(1,i) = 2.5 - t_n(1,i);

    elseif t_n(1,i) < 2
        g_n(1,i) = 1.5;

    else
        g_n(1,i) = 0;

    end

end
%{
stem(t_n,g_n,"k");
grid on;
title('$Sampled\,\,Signal$','Interpreter','latex','FontSize',14);
ylabel('$g(nT_s)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);
ylim([-1 4]);

ax = gca;
ax.YLabel.Rotation = 360;
%}

g_r = DtoAyberk(1,Ts2,dur,g_n);
t_r = linspace(0,2,length(g_r));
plot(t_r,g_r,"k");

%Part 6:Reconstruction of a Sum of Cosines

ID = 22003836;
D = mod(ID,5);
Ts = 0.005*(D+1);
t = -3:Ts/1000:3;
t_n = -3:Ts:3;
N = length(t_n);
Xn = zeros(1,N);
x_t = 0.5*cos(2*pi*t+pi/5) + ...
      0.3*sin(6*pi*t+2/3) +  ...
      0.4*cos(5*pi*t-0.7*exp(1));
for i=1:N
    Xn(1,i) = 0.5*cos(2*pi*(i*Ts)+pi/5) + ...
              0.3*sin(6*pi*(i*Ts)+2/3) + ...
```

```
            0.4*cos(5*pi*(i*Ts)-0.7*exp(1));
end

[x_r_z,t_z] = DtoA(0,Ts,3,Xn);
[x_r_l,t_l] = DtoA(1,Ts,3,Xn);
[x_r_i,t_i] = DtoA(2,Ts,3,Xn);

subplot(4,1,1);
plot(t,x_t);
grid on;
title('$Original\,\,Signal\,\,x(t)$',...
      'Interpreter','latex','FontSize',14);
ylabel('$x_R(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);
ax = gca;
ax.YLabel.Rotation = 360;

subplot(4,1,2);
plot(t_z,x_r_z);
grid on;
title('$Zero\,\,Hold\,\,Interpolation$',...
      'Interpreter','latex','FontSize',14);
ylabel('$x_R(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);
ax = gca;
ax.YLabel.Rotation = 360;

subplot(4,1,3);
plot(t_l,x_r_l);
grid on;
title('$Linear\,\,Interpolation$',...
      'Interpreter','latex','FontSize',14);
ylabel('$x_R(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);
ax = gca;
ax.YLabel.Rotation = 360;

subplot(4,1,4);
plot(t_i,x_r_i);
grid on;
title('$Ideal\,\,Interpolation$',...
      'Interpreter','latex','FontSize',14);
ylabel('$x_R(t)$','Interpreter','latex','FontSize',14);
xlabel('$t$','Interpreter','latex','FontSize',14);
ax = gca;
ax.YLabel.Rotation = 360;
```

Published with MATLAB® R2023b

```
%----------------------EEE321 LAB5 ONLAB CODE----------------------------
Ts = 1/120;
samp_int = Ts/1000;
t = 0:samp_int:2;
t_n = 0:Ts:2;
x = 0.2*cos(2*pi*40*t)+0.8*cos(2*pi*8*t+0.5);
clf;
plot(t,x,"k");

N = length(t_n);
X = zeros(1,N);
for i = 1:N-1
    X(1,i) = 0.2*cos(2*pi*40*(i*Ts))+0.8*cos(2*pi*8*(i*Ts)+0.5);
end

a = 2;

[XaZ,XaL,XaI] = SampleA2(X,Ts,a);
t_new = linspace(0,2,length(XaZ));

clf;
subplot(3,1,1);
plot(t_new,XaZ,"k");
title('$XaZ[n]\,\,a=2$','Interpreter','latex','FontSize',14);

subplot(3,1,2);
plot(t_new,XaL,"k");
title('$XaL[n]\,\,a=2$','Interpreter','latex','FontSize',14);

subplot(3,1,3);
plot(t_new,XaI,"k");
title('$XaI[n]\,\,a=2$','Interpreter','latex','FontSize',14);

[x_r_zii,t_zii] = DtoA(0,a*Ts,2,XaZ);
[x_r_lii,t_lii] = DtoA(1,a*Ts,2,XaL);
[x_r_iii,t_iii] = DtoA(2,a*Ts,2,XaL);

clf;
subplot(3,1,1);
plot(t_zii,x_r_zii,"k");
title('$x_R(t)\,\,Reconstructed\,\,From\,\,XaZ\,\,a = 2$',...
    'Interpreter','latex','FontSize',14);

subplot(3,1,2);
plot(t_zii,x_r_lii,"k");
title('$x_R(t)\,\,Reconstructed\,\,From\,\,XaL\,\,a = 2$',...
    'Interpreter','latex','FontSize',14);

subplot(3,1,3);
plot(t_zii,x_r_iii,"k");
title('$x_R(t)\,\,Reconstructed\,\,From\,\,XaI\,\, a = 2$',...
    'Interpreter','latex','FontSize',14);
```

Published with MATLAB® R2023b

```
function [XaZ,XaL,XaI]= SampleA2(X,Ts,a)

    Ts_new = a*Ts;
    samp_int = Ts/1000;
    N = length(X);
    dur = 2;

    [x_r_z, t_r] = DtoA(0,Ts,dur,X);
    [x_r_l, t_l] = DtoA(1,Ts,dur,X);
    [x_r_i, t_i] = DtoA(2,Ts,dur,X);

    for i=1:N-1
        if i*Ts_new < 2
            XaZ(1,i) = x_r_z(1,i*a*1000);
        end
    end

    for i=1:N-1
        if i*Ts_new < 2
            XaL(1,i) = x_r_l(1,i*a*1000);
        end
    end

    for i=1:N-1
        if i*Ts_new < 2
            XaI(1,i) = x_r_i(1,i*a*1000);
        end
    end
end
```

Published with MATLAB® R2023b

```
%Function generates recovered signal
function [x_r,t_r] = DtoA(type,Ts,dur,Xn)

    p = generateAyberk(type,Ts,dur);
    N  = length(Xn);
    len_p = length(p);
    low_b = floor(len_p/2);

    %t = linspace(-dur,dur,length(p));
    %plot(t,p,"k",LineWidth=1.3);

    x_r = zeros(1,1000*(N-1)+len_p);

    for i=1:N
        %disp(i)
        x_r(1,1+(i-1)*1000:len_p+(i-1)*1000) ...
      = x_r(1,1+(i-1)*1000:len_p+(i-1)*1000) ...
      + Xn(1,i).*p;

    end

    x_r = x_r(1,low_b+1:low_b+1000*(N-1));
    t_r = linspace(-dur,dur,length(x_r));

end
```

Published with MATLAB® R2023b

```
function p = generateInterp(type,Ts,dur)

t = -dur/2:Ts/1000:dur/2;

    if type == 0
        p = zeros(1,length(t));
        i2 = 1;

        for i=-dur/2:Ts/1000:dur/2

            if (i >= -Ts/2) && (i <= Ts/2)
                p(1,i2) = 1;
            else
                p(1,i2) = 0;
            end

            i2 = i2 + 1;

        end

    elseif type == 1

        p = zeros(1,length(t));
        i2 = 1;

        for i=-dur/2:Ts/1000:dur/2

            if (i >= -Ts/2) && (i < 0)
                p(1,i2) = 1+(2*i)/Ts;
            elseif (i >= 0) && (i < Ts/2)
                p(1,i2) = 1-(2*i)/Ts;
            else
                p(1,i2) = 0;
            end

            i2 = i2 + 1;

        end

    elseif type == 2

        p = zeros(1,length(t));
        i2 = 1;

        for i=-dur/2:Ts/1000:dur/2
            if i == 0
                p(1,i2) = 1;
            else
                p(1,i2) = Ts*(sin((pi*i)/Ts)/(pi*i));
            end

            i2 = i2 + 1;
```

```
        end
    end

end
```

Published with MATLAB® R2023b