Bilkent University

EEE321: Signals and Systems

Lab Assignment 5

Mehmet Emre Uncu - 22003884

Section - 01

Part 1:

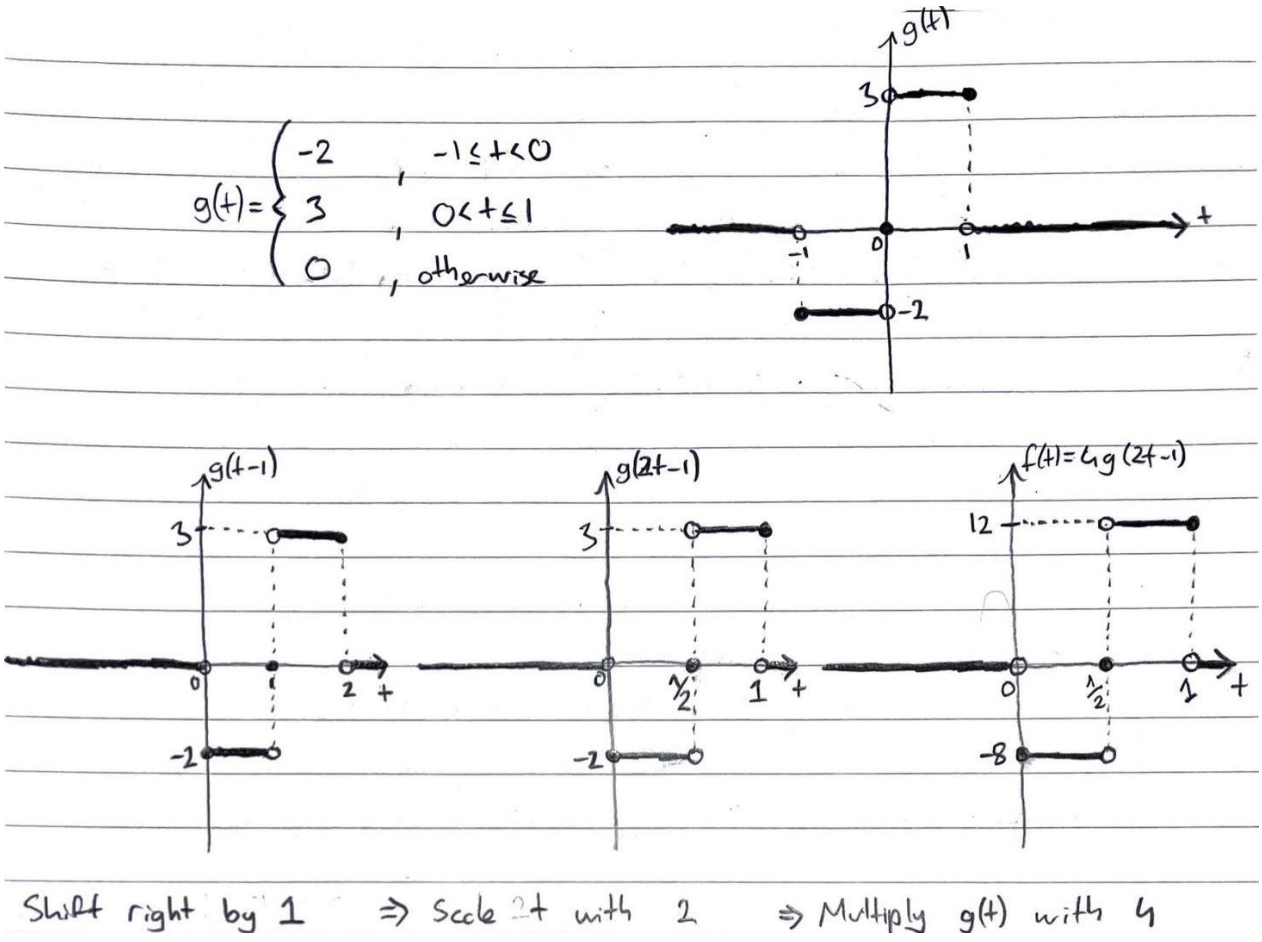


Figure 1: Sketch of g(t) and f(t)

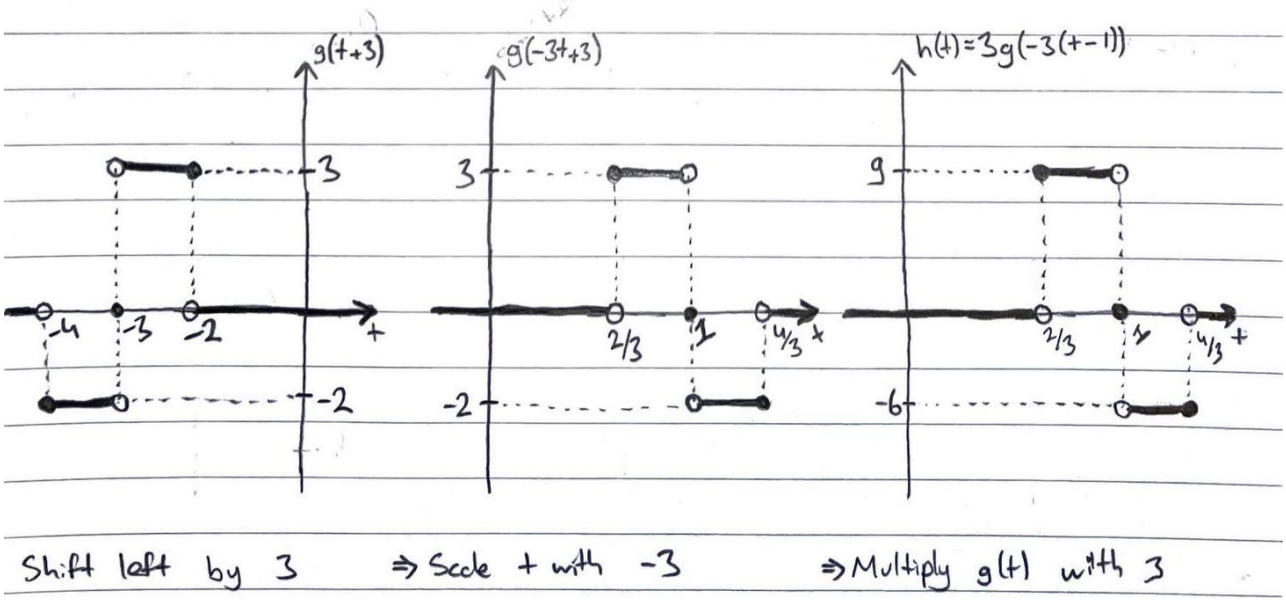


Figure 2: Sketch of h(t)

We can't recover g(t) fully when it is sampled since $G(j\omega) \neq 0$ for $\omega > \omega_c$ thus it is not a band limited signal.

$$G(j\omega) = \int_{-\infty}^{\infty} g(t).e^{-j\omega t}dt = \int_{-1}^{0} -2.e^{-j\omega t}dt + \int_{0}^{1} 3.e^{-j\omega t}dt = \frac{2}{j\omega}e^{j\omega t}\Big|_{-1}^{0} + \frac{-3}{j\omega}e^{j\omega t}\Big|_{0}^{1}$$

$$= \frac{2}{j\omega}(1-e^{j\omega}) - \frac{3}{j\omega}(e^{j\omega}-1) = \frac{5}{j\omega} - \frac{1}{j\omega}\left(e^{j\omega} + \frac{4e^{j\omega}+4e^{-j\omega}}{2}\right) = \frac{5}{j\omega} - \frac{e^{j\omega}}{j\omega} - \frac{4\cos j\omega}{j\omega}$$

Since $G(j\omega)$ contains cos, it is not a band limited signal and can't be fully recovered

Figure 3: Answer for Part 1

Part 2:

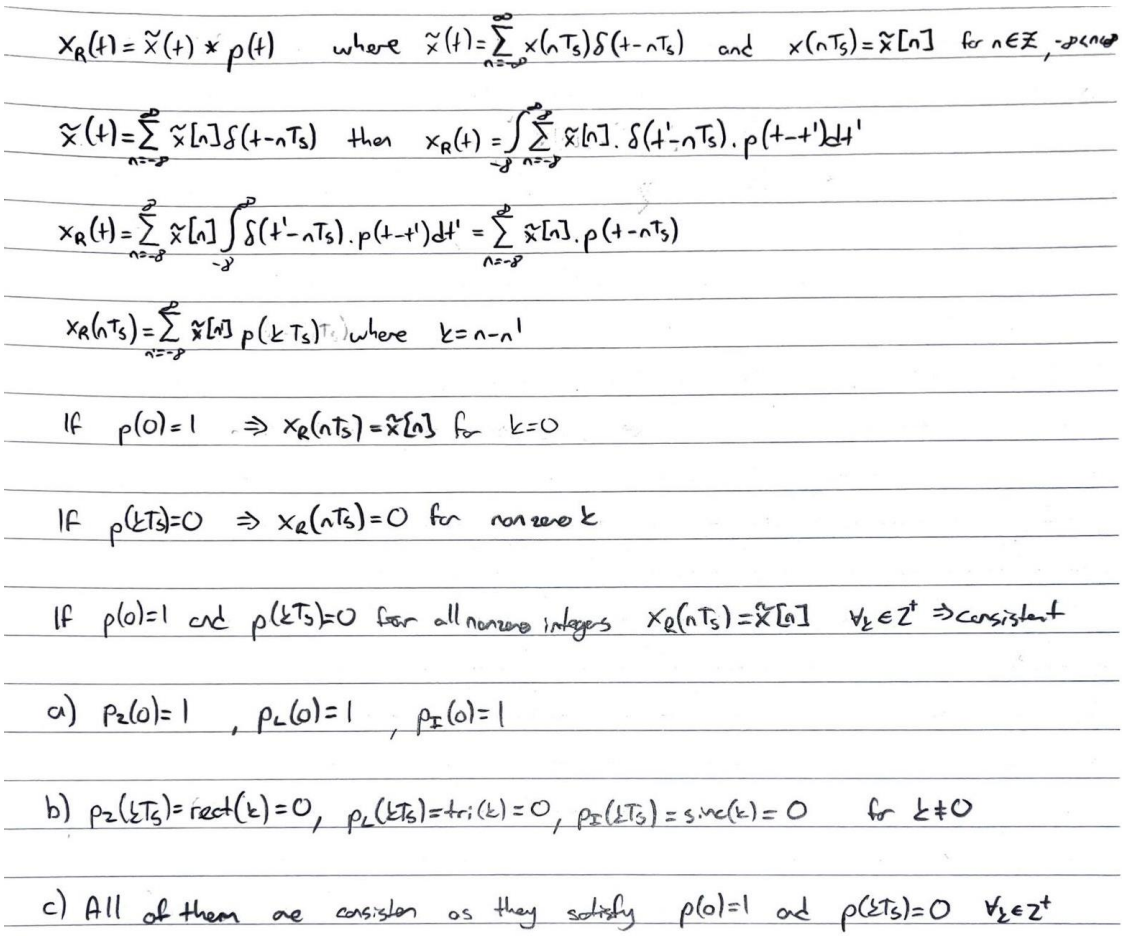


Figure 4: Calculations for Part 2

Part 3:

The generateInterp(type,Ts,dur) function is as follows;

```
function p = generateInterp(type, T_s, dur)
    t = -dur/2:T_s/1200:dur/2-T_s/1200;
    p = zeros(1, length(t));
    if type == 0
        p(-T_s/2 <= t & t < T_s/2) = 1;
    elseif type == 1
        p(-T_s/2 <= t & t <= T_s/2) = 1 - 2*abs(t(-T_s/2 <= t & t <= T_s/2))/T_s;
    elseif type == 2
        p(t==0) = 1;
        p = sin(pi*t/T_s)./(pi*t/T_s);
    end
end
```

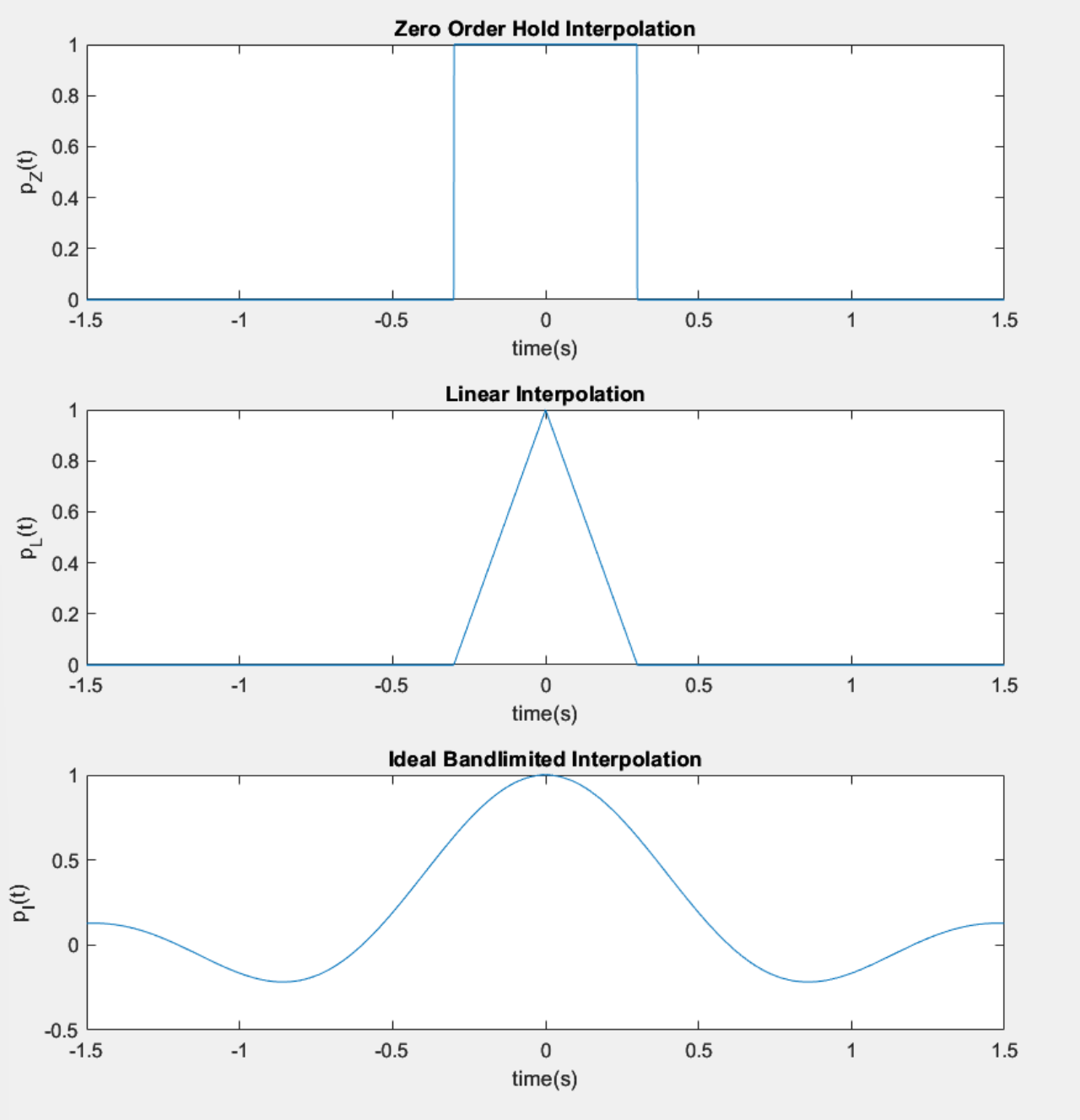


Figure 5. Interpolation Graphs

Part 4:

The DtoA(type, T_s, dur, X_N) function is as follows;

```
function x_R = DtoA(type, T_s, dur, X_N)
    dur = dur*T_s;
    p   = generateInterp(type, T_s, dur);
    x_R = zeros(1, round((dur + (length(X_N) - 1)*T_s)/(T_s/1200)));
    for n = 0:length(X_N) - 1
        x_R(1+(n*1200):round((dur+n*T_s)/(T_s/1200))) =
x_R(1+(n*1200):round((dur+n*T_s)/(T_s/1200)))+X_N(n+1)*p;
    end
end
```

Part 5:

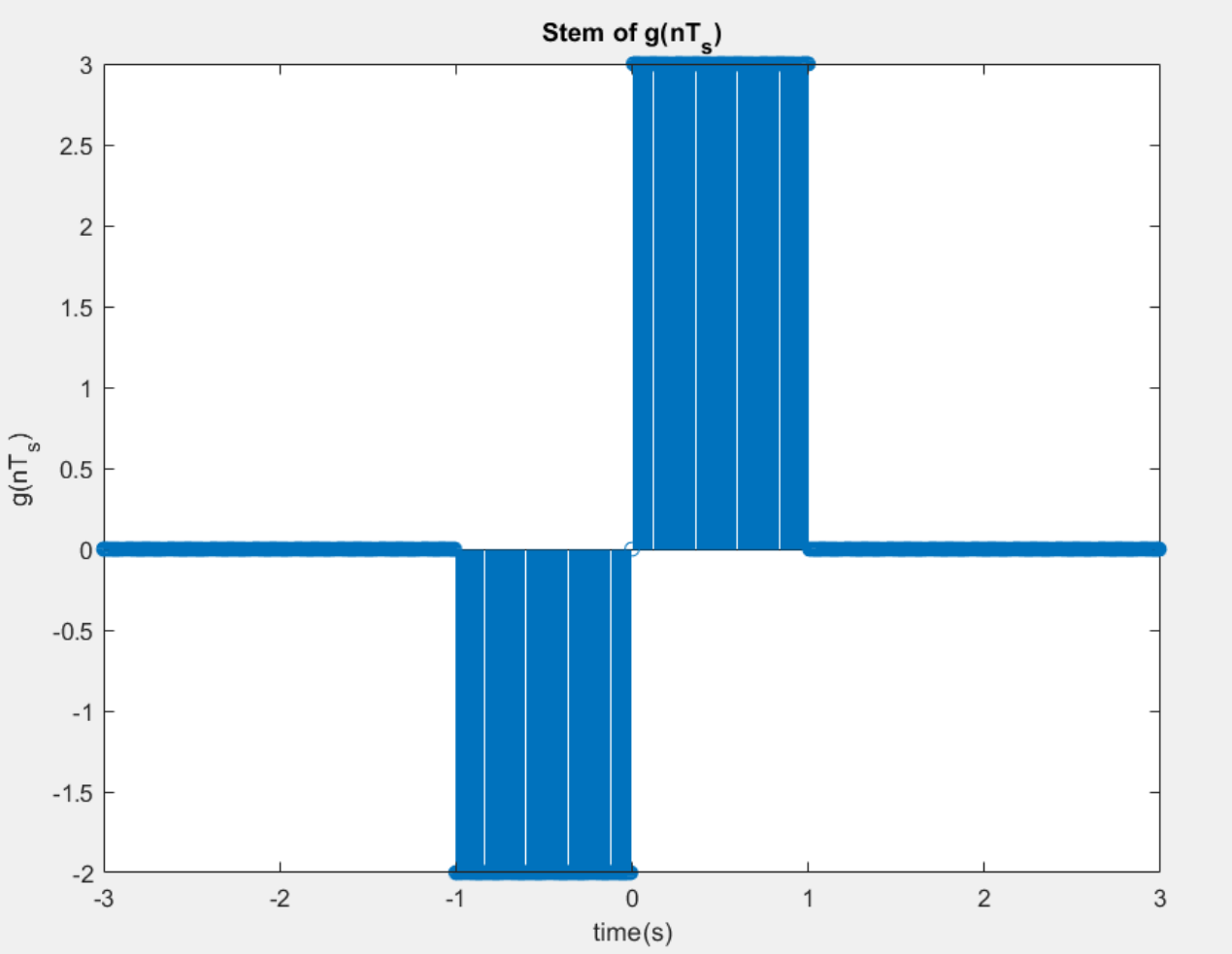
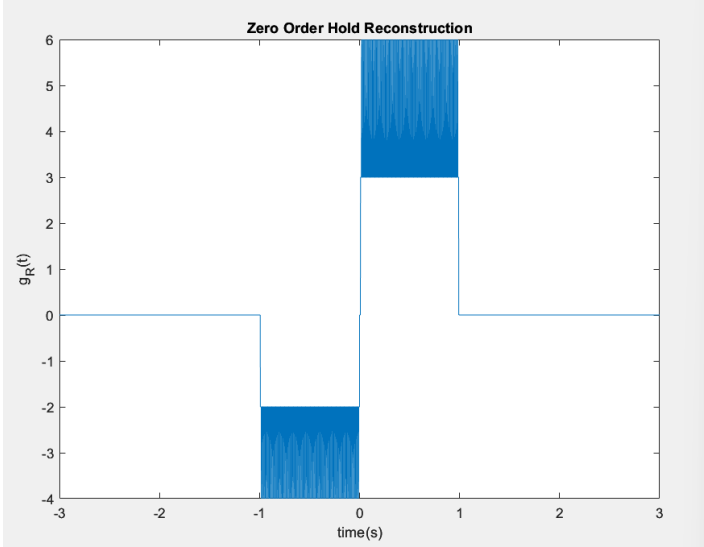
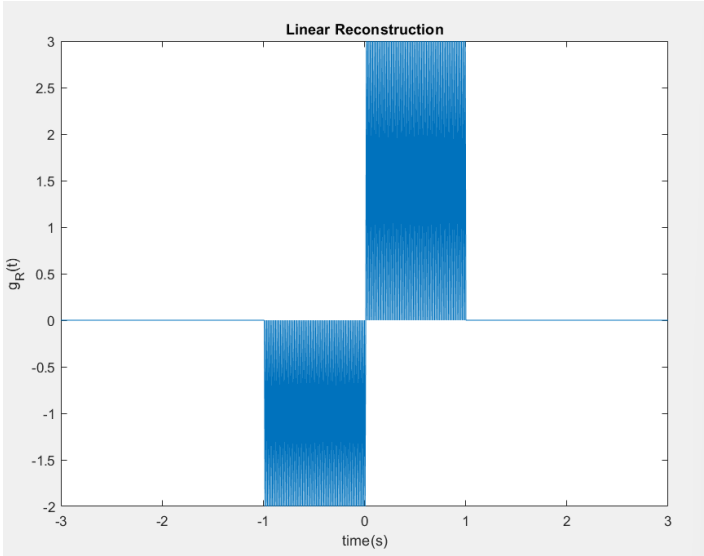


Figure 6: Stem of g(nTs)





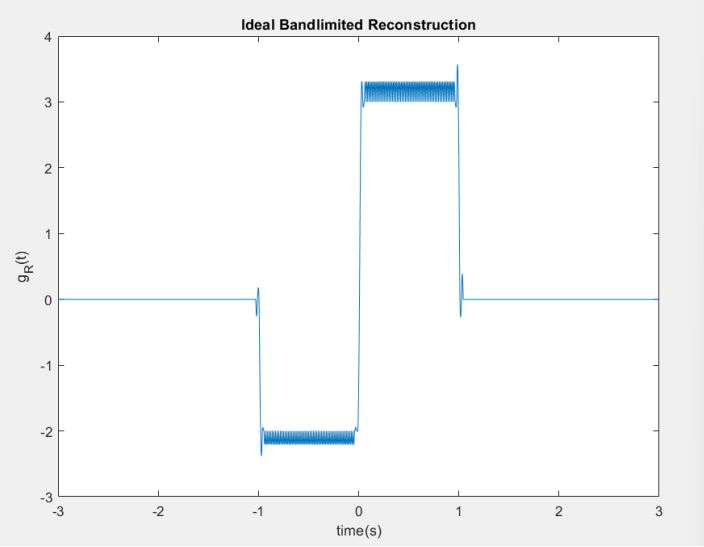


Figure 7: Plots for Part 5

When zero hold reconstruction is examined, it is seen that the values of the function are doubled at some points due to the overlapping of the edges. When linear reconstruction is examined, it is seen that the borders of the resulting graph are the same as the original signal, but the desired image cannot be obtained because the interior of the graph is full of triangular pulses. Finally, when the ideal bandlimited reconstruction was examined, oscillations occurred in the max and min values of the signal, but other than that, the graph resembles the original signal.

As T_s was increased, the reconstruction of the signal began to move further away from the original, as less samples were taken in the same period.

Part 6:

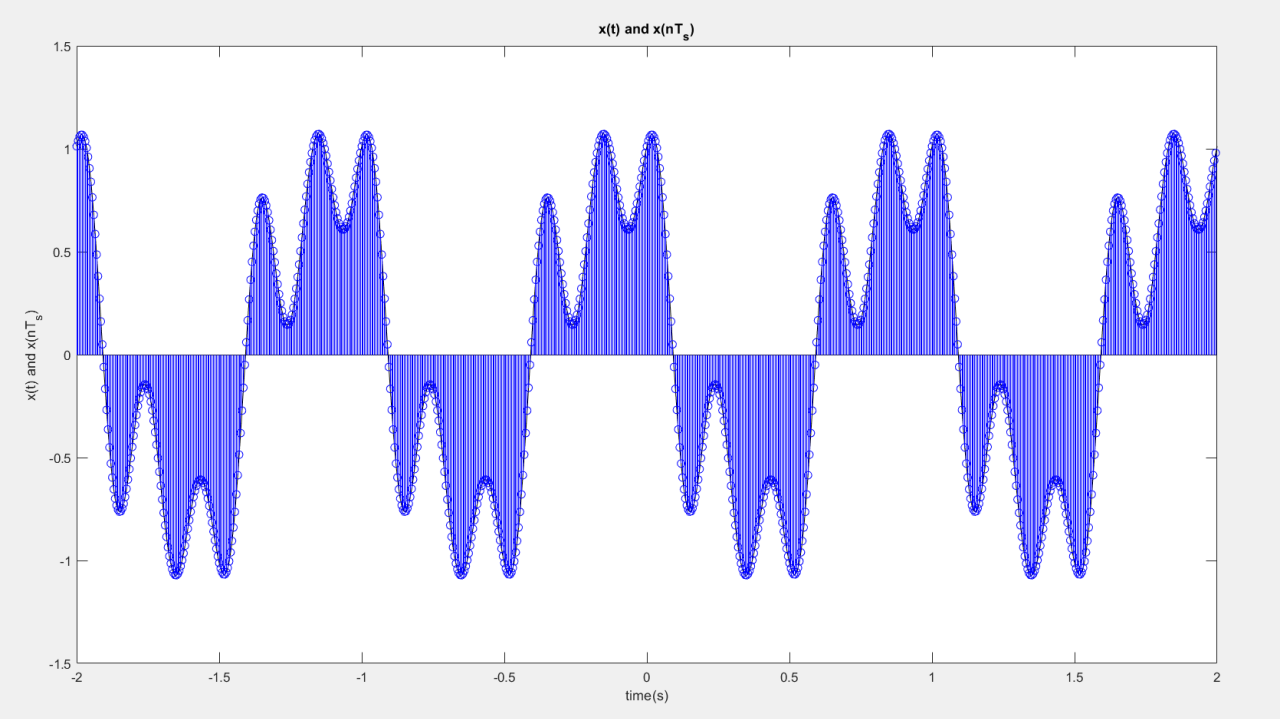


Figure 8: plot of x(t) and stem of $x(nT_s)$

a) When $T_s = 0.005(D_7 + 1)$:

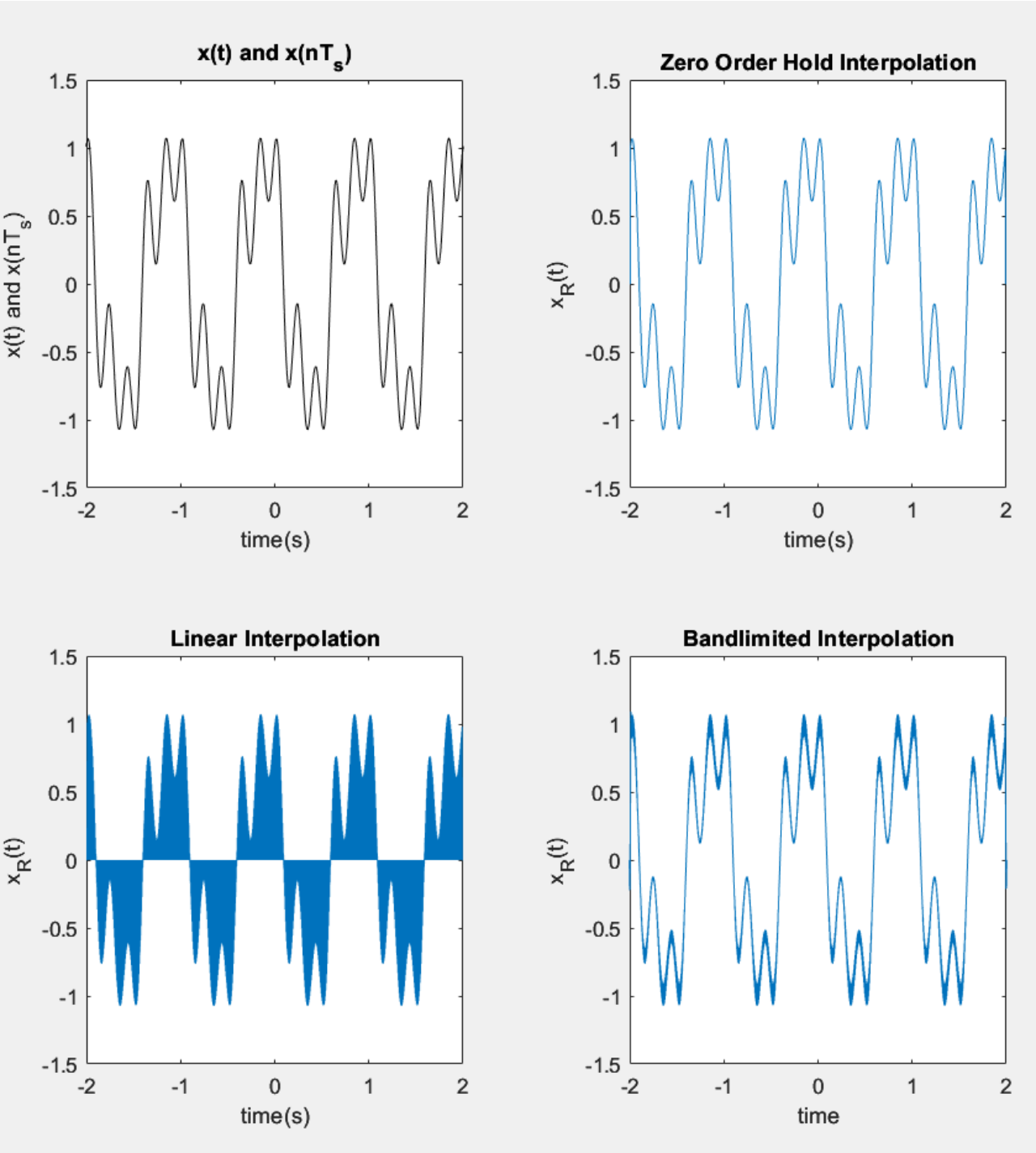


Figure 9: When $T_s = 0.005(D_7 + 1)$:

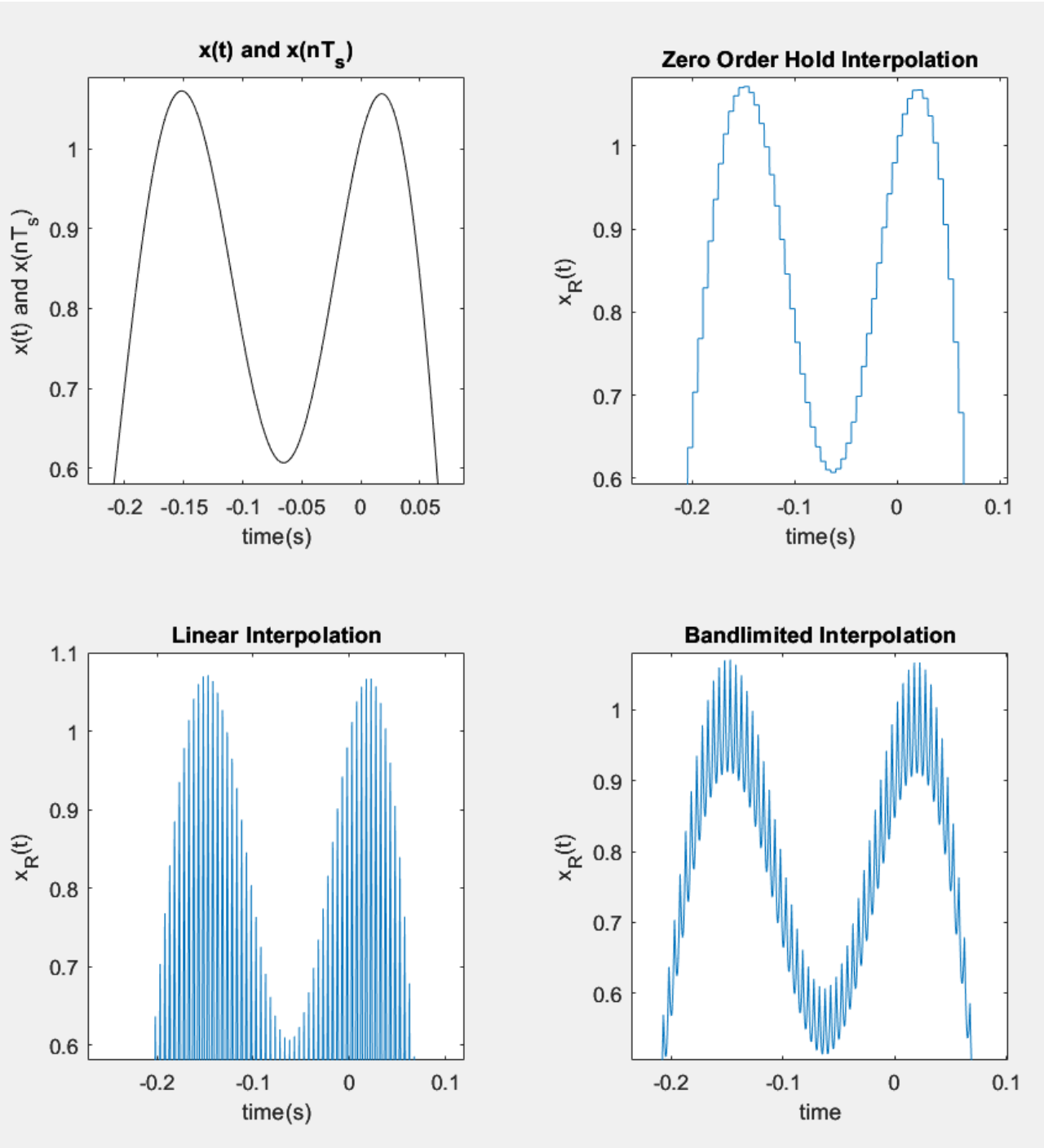


Figure 10: Detailed versions of previous graphs

b) When $T_s = 0.25 + 0.01D_7$:

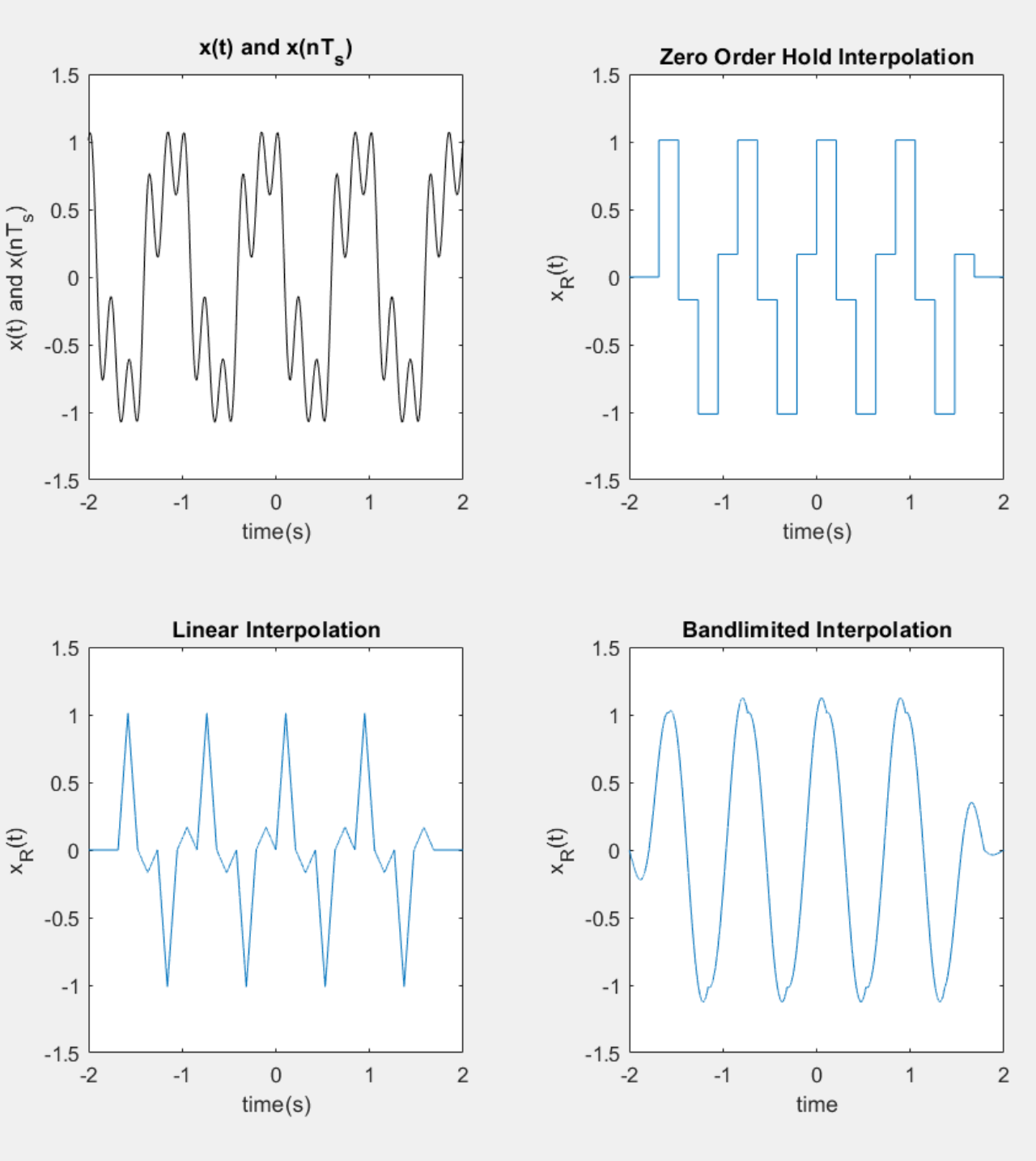


Figure 11: When $T_s = 0.25 + 0.01$:

c) When $T_s = 0.18 + 0.05(D_7 + 1)$:

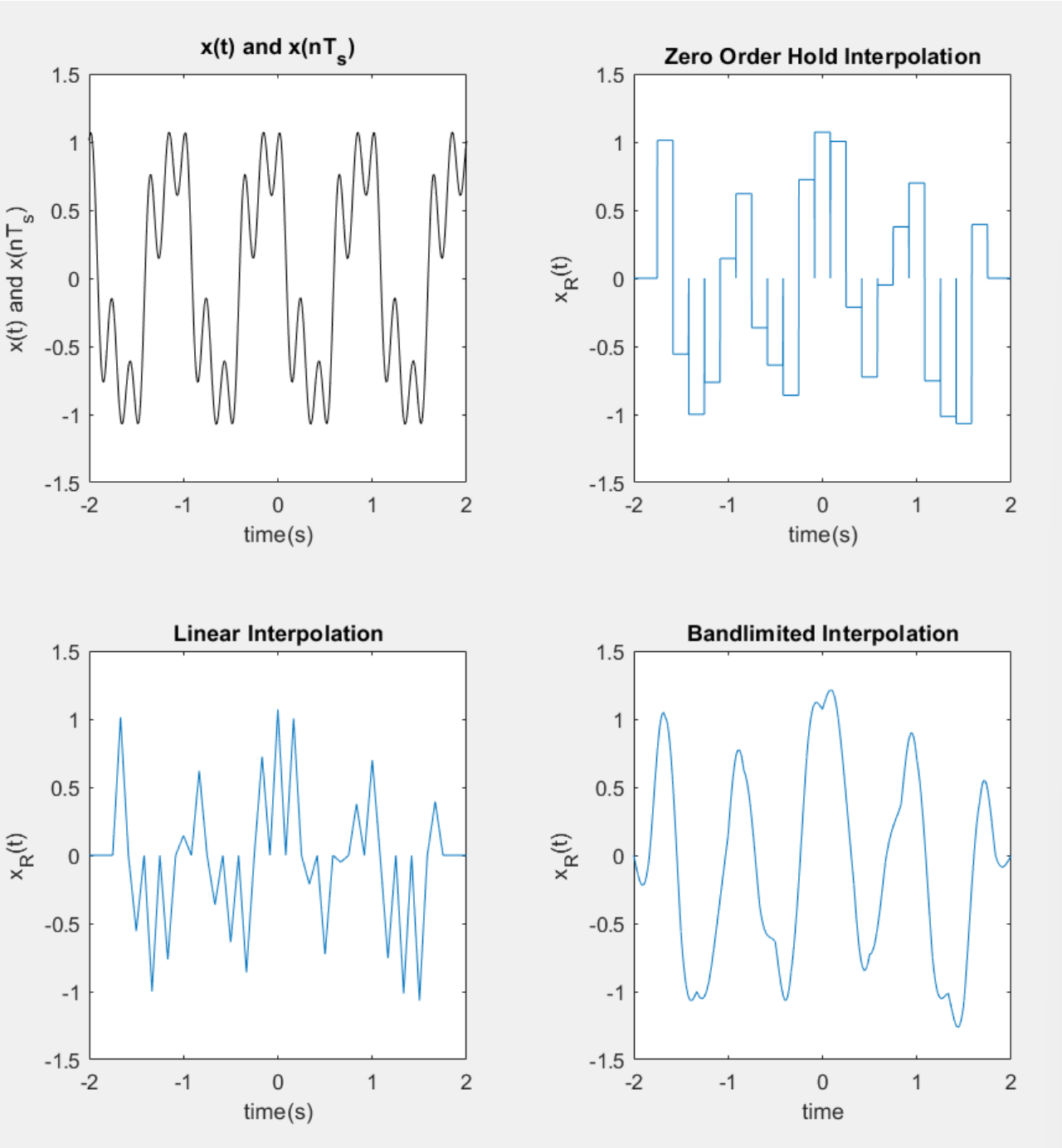


Figure 12: When $T_s = 0.18 + 0.05(D_7 + 1)$:

d) When $T_s = 0.099$:

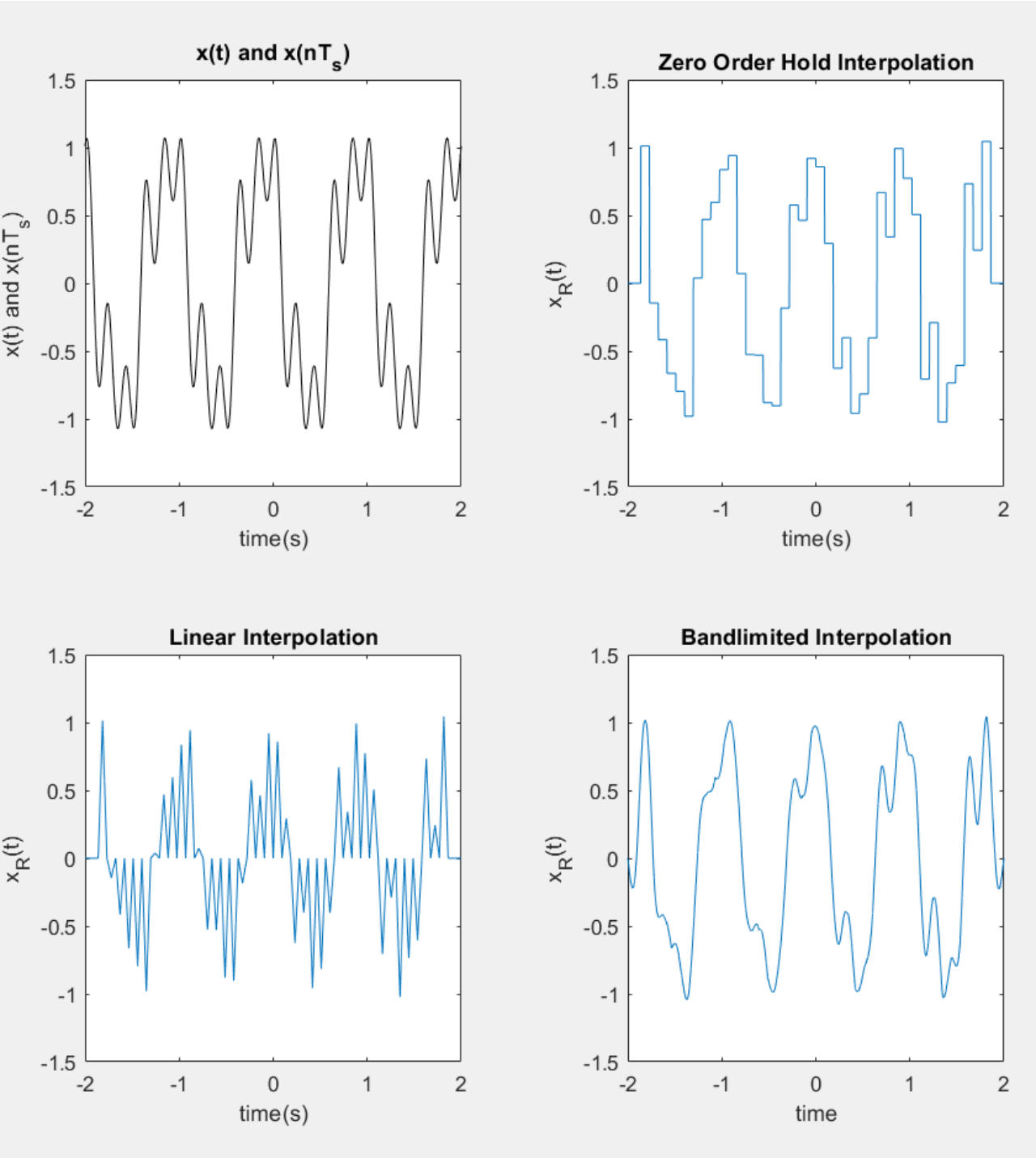


Figure 13: When $T_s = 0.099$

Since the D_7 value is 0, the T_s value is 0.005. When the graphs are examined, it is seen that the boundaries of the linear reconstruction graph are exactly the same as the original signal, but it is filled. Although the zero hold reconstruction graph appears to be the same as the original signal because the T_s value is quite small, it is seen that it consists of corners when the graph is enlarged. As the T_s value increases, these corners become more prominent. When Ideal Bandlimited Reconstruction is examined, a signal very similar to the original signal is obtained as the T_s value decreases. Unlike zero hold reconstruction, the distortion becomes less as the T_s value gradually increases from 0.005. Therefore, although Ideal Bandlimited Reconstruction is more successful except for minimum T_s values, it can be distinguished from the original signal due to the oscillations at the edges. If an infinite number of samples could be taken, the oscillations between two samples could be minimized, and the original signal could be obtained.

While the original signal can be obtained when the T_s value goes to zero, that is, when an infinite number of samples are taken, distortions occur in the reconstructed signal as the T_s value increases. When the T_s value is between 0.01 and 0.1, a reconstructed signal very similar to the original signal is obtained. As the T_s value increases, the distance between the samples increases, and the success of the ideal bandlimited interpolator decreases due to the oscillations occurring in these intervals becoming more evident. When the T_s value exceeds 0.1, distortions increase because the Nyquist criteria cannot be satisfied. Since the maximum frequency of the signal is 5Hz, the T_s value must be less than 0.1 in order to meet the Nyquist criteria. As expected, distortions increase when this value is exceeded.

MATLAB Code:

```
%% PART 3

dur = 3;
T_s = dur/5;
t   = -dur/2:T_s/1200:dur/2-T_s/1200;
tiledlayout(3, 1);
for i = 0:2
    p = generateInterp(i, T_s, dur);
    if i == 0
        nexttile;
        plot(t, p);
        title('Zero Order Hold Interpolation');
        xlabel('time(s)');
        ylabel('p_Z(t)');
    elseif i == 1
        nexttile;
        plot(t, p);
        title('Linear Interpolation');
        xlabel('time(s)');
        ylabel('p_L(t)');
    elseif i == 2
        nexttile;
        plot(t, p);
        title('Ideal Bandlimited Interpolation');
        xlabel('time(s)');
        ylabel('p_I(t)');
    end
end


%% PART 5

a   = randi([2 6],1);
T_s  = 1/(20*a);
dur = 6;
t   = -dur/2:T_s:dur/2-T_s;
g   = zeros(1,length(t));
g(-1 <= t & t <  0) = -2;
g(0  <  t & t <= 1)=3;
tiledlayout(2, 2);
nexttile;
stem(t,g);
title('Stem of g(nT_s)');
xlabel('time(s)');
ylabel('g(nT_s)');
nexttile;
g_RZ = DtoA(0, T_s, dur, g);
plot(linspace(-3, 3, length(g_RZ)), g_RZ);
title('Zero Order Hold Reconstruction');
xlabel('time(s)');
```

```
ylabel('g_R(t)');
nexttile;
g_RL = DtoA(1, T_s, dur, g);
plot(linspace(-3, 3, length(g_RL)), g_RL);
title('Linear Reconstruction');
xlabel('time(s)');
ylabel('g_R(t)');
nexttile;
g_RI = DtoA(2, T_s, dur, g);
plot(linspace(-3, 3, length(g_RI)), g_RI);
ylabel('g_R(t)');
title('Ideal Bandlimited Reconstruction');
xlabel('time(s)');


%% PART 6

D   = 0;
T_s = 0.005*(0+1); % 0.005
dur = 4;
t   = -dur/2:T_s/1200:dur/2-T_s/1200;
st  = -dur/2:T_s:dur/2-T_s;
xt  = 0.25*cos(2*pi*3*t+pi/8)+0.4*cos(2*pi*5*t-1.2)+0.9*cos(2*pi*t+pi/4);
sxt = 0.25*cos(2*pi*3*st+pi/8)+0.4*cos(2*pi*5*st-1.2)+0.9*cos(2*pi*st+pi/4);
figure
plot(t, xt, 'k');
title('x(t) and x(nT_s)');
xlabel('time(s)');
ylabel('x(t) and x(nT_s)');
hold on
stem(st, sxt,'b')
hold off


%% PART 6 PLOTS

D   = 0;
dur = 4;
t   = -dur/2:T_s/1200:dur/2-T_s/1200;
st  = -dur/2:T_s:dur/2-T_s;
xt  = 0.25*cos(2*pi*3*t+pi/8)+0.4*cos(2*pi*5*t-1.2)+0.9*cos(2*pi*t+pi/4);
sxt = 0.25*cos(2*pi*3*st+pi/8)+0.4*cos(2*pi*5*st-1.2)+0.9*cos(2*pi*st+pi/4);
tiledlayout(2, 2);
nexttile;
plot(t, xt, 'k');
title('x(t) and x(nT_s)');
xlabel('time(s)');
ylabel('x(t) and x(nT_s)');
nexttile;
x_0 = DtoA(0, T_s, dur, sxt);
t_0 = linspace(-2, 2, length(x_0));
plot(t_0, x_0);
title('Zero Order Hold Interpolation');
xlabel('time(s)');
ylabel('x_R(t)');
```

```
nexttile;
x_1 = DtoA(1, T_s, dur, sxt);
t_1 = linspace(-2, 2, length(x_1));
plot(t_1, x_1);
title('Linear Interpolation');
xlabel('time(s)');
ylabel('x_R(t)');
nexttile;
x_2 = DtoA(2, T_s, dur, sxt);
t_2 = linspace(-2, 2, length(x_2));
plot(t_2,x_2);
title('Bandlimited Interpolation');
xlabel('time');
ylabel('x_R(t)');


%% PART 6 A

T_s = 0.005*(0+1); % 0.005


%% PART 6 B

T_s = 0.25 + 0.01*0; % 0.25


%% PART 6 C

T_s = 0.18 + 0.005*(0+1); % 0.185


%% PART 6 D

T_s = 0.099;


%% PART 6 D

T_s = 0.001;


%% FUNCTIONS

function p = generateInterp(type, T_s, dur)
    t = -dur/2:T_s/1200:dur/2-T_s/1200;
    p = zeros(1, length(t));
    if type == 0
        p(-T_s/2 <= t & t < T_s/2) = 1;
    elseif type == 1
        p(-T_s/2 <= t & t <= T_s/2) = 1 - 2*abs(t(-T_s/2 <= t & t <= T_s/2))/T_s;
    elseif type == 2
        p(t==0) = 1;
        p = sin(pi*t/T_s)./(pi*t/T_s);
    end
end
```

```
function x_R = DtoA(type, T_s, dur, X_N)
    dur = dur*T_s;
    p   = generateInterp(type, T_s, dur);
    x_R = zeros(1, round((dur + (length(X_N) - 1)*T_s)/(T_s/1200)));
    for n = 0:length(X_N) - 1
        x_R(1+(n*1200):round((dur+n*T_s)/(T_s/1200))) =
x_R(1+(n*1200):round((dur+n*T_s)/(T_s/1200)))+X_N(n+1)*p;
    end
end
```