Report: Signals and Systems Lab Assignment 5

Bilkent University Electrical and Electronics Department
EE321-01

Semih Akkoç - 22103132

November 25, 2023

Part 1:

The graph of $f(t)$ mirrors $g(t)$ but with a displacement to the right by 1 unit, a halving of the signal's frequency, and an amplification of the amplitude by a factor of 4. As for $h(t)$, its plot resembles $g(t)$ but shifted left by 3 units, a reduction in the signal's frequency by threefold, mirrored along the y-axis, and lastly, its amplitude scaled by a factor of 3. These modifications are distinctly evident in the accompanying figure displaying the resulting graphs.

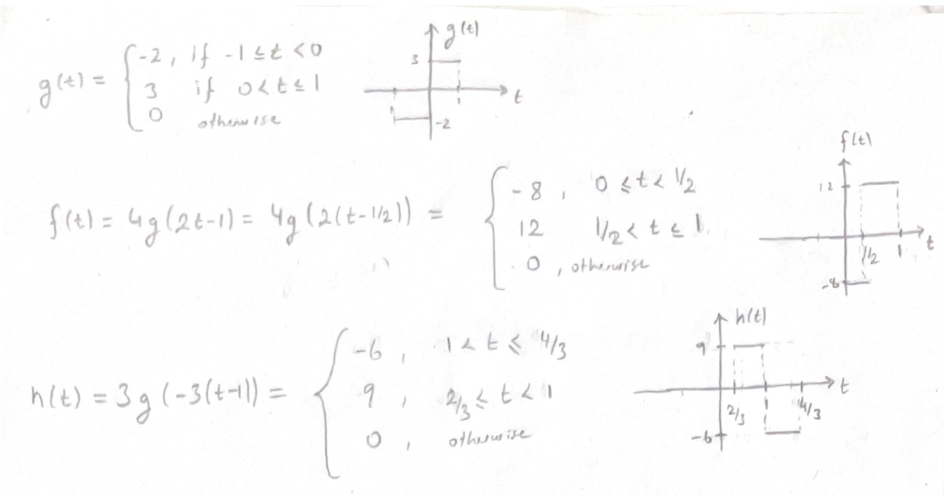


Figure 1: Graphs of $g(t)$, $f(t)$, and $h(t)$.

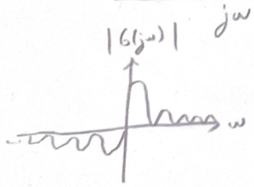
From the next figure, it can seen that whether $g(t)$ is recoverable or not,

To show that $g(t)$ is fully recoverable it needs to satisfy Nyquist rate. So it needs to have a band-limit.

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

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

$$= \frac{5 - 5\cos\omega + j\sin\omega}{j\omega}, \quad \text{to plot } |G(j\omega)| = \sqrt{\frac{\sin^2(\omega)}{\omega^2} + \left(\frac{5 - 5\cos\omega}{\omega}\right)^2}$$



Since in frequency domain $g(t)$ does not have a band-limit it camot be fully recoverable.

Figure 2: $g(t)$ being not recoverable.

Part 2:

The validity of $x_R(t) = \sum_{n'=-\infty}^{\infty} x[n']p(t - n'T_s)$, which ensures the condition $x_R(nT_s) = x(nT_s)$ when $p(0) = 1$ and $p(kT_s) = 0$, has been confirmed. The consistency of certain Interpolation Functions, including zero order, linear, and Ideal functions, is demonstrated in the figure provided.

$$x_R(t) = \bar{x}(t) * p(t) = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(nT_s)\,\delta(t'-nT_s)\,p(t-t')\,dt'$$

$$= \sum_{n=-\infty}^{\infty} x(nT_s) \int_{-\infty}^{\infty} \delta(t'-nT_s)\,p(t-t')\,dt' = \sum_{n=-\infty}^{\infty} x(nT_s)\,p(t-nT_s)$$

since $\bar{x}[n] = x[nT_s]$ we have,

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

$$\Rightarrow x_R(nT_s) = \sum_{n'=-\infty}^{\infty} \bar{x}(n')\,p(nT_s-n'T_s) = \sum_{n'=-\infty} \bar{x}(n')\,p((n-n')T_s)$$

Let $p(0)=1$ and $p(kT_s)=0$ for $k \in \mathbb{Z}-\{0\}$

If $n-n'=0 \Rightarrow p((n-n')T_s) = p(0) = 1 \Rightarrow x_R(nT_s) = \sum_{n'=-\infty}^{\infty} \bar{x}(n') = x(nT_s)$

If $n-n' \neq 0 \Rightarrow p((n-n')T_s) = 0 \Rightarrow x_R(nT_s) = 0$

Therefore we have, $x_R(nT_s) = x(nT_s)$

a) $p_Z(0) = rect(0) = 1$
$p_L(0) = tri(0) = 1$
$p_I(0) = sinc(0) = 1$

b) $p_Z(kT_s) = rect(k) = 0$
$p_L(kT_s) = tri(k) = 0$
$p_I(kT_s) = sinc(k) = 0$
for $k \in \mathbb{Z}-\{0\}$

c) Due to $p_{Z,L,I}(0)=1$ and $p_{Z,L,I}(kT_s)=0$ we can say that 3 interpolation systems are consistent.

Figure 3: Calculations for part 2.

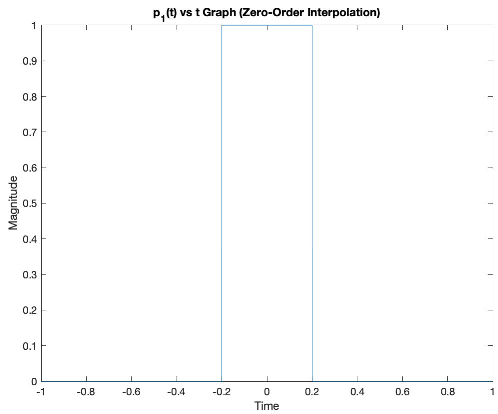
Part 3:

The code for the Part 3:

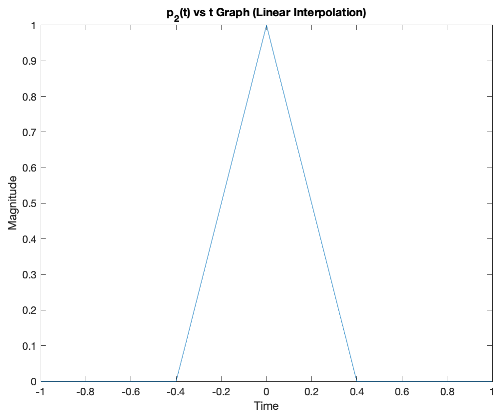
```
dur = mod(22103132, 7);
Ts = dur/5;
t = -dur/2:Ts/500:dur/2-Ts/500;

p1 = generateInterp(0,Ts,dur);
p2 = generateInterp(1,Ts,dur);
p3 = generateInterp(2,Ts,dur);

figure;

plot(t, p1);
xlabel('Time');
ylabel('Magnitude');
title('p_1(t) vs t Graph (Zero-Order Interpolation)');


figure;

plot(t, p2);
xlabel('Time');
ylabel('Magnitude');
title('p_2(t) vs t Graph (Linear Interpolation)');


figure;

plot(t, p3);
xlabel('Time');
ylabel('Magnitude');
title('p_3(t) vs t Graph (Ideal Interpolation)');

function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
        p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
```

```
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```

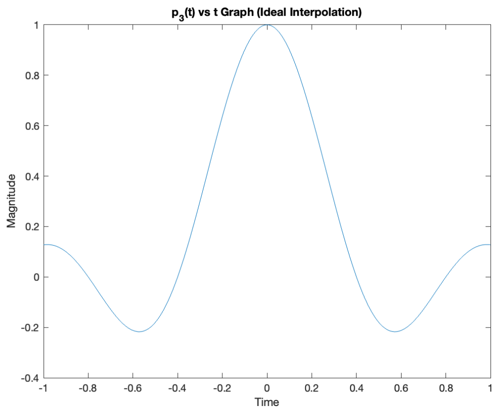
Graphs of the interpolation functions,



(a) Zero-order interpolation.



(b) Linear interpolation.



(c) Ideal interpolation.

Figure 4: Interpolation function graphs.

Part 4:

The code for the Part 4:

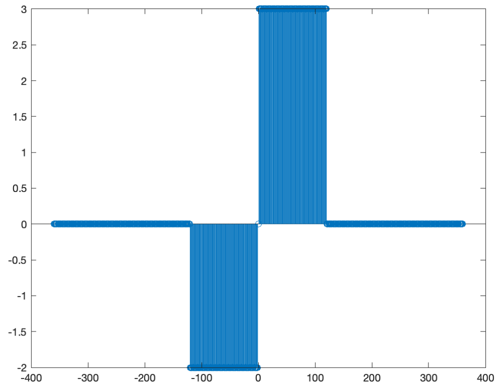
```
function [xR] = DtoA(type,Ts,dur,Xn)
    p = generateInterp(type, Ts, dur);
    l = length(Xn)*500+length(p);
    xR = zeros(1,l );
    for x = 1: length(Xn)
        xR(1+500*(x-1): 500*(x-1) + length(p)) = Xn(x)*p+
            xR(1+500*(x-1): 500*(x-1) +length(p));
    end
    xR = xR(250*length(Xn)+1:end-250*length(Xn));
end


function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
        p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```

Part 5:

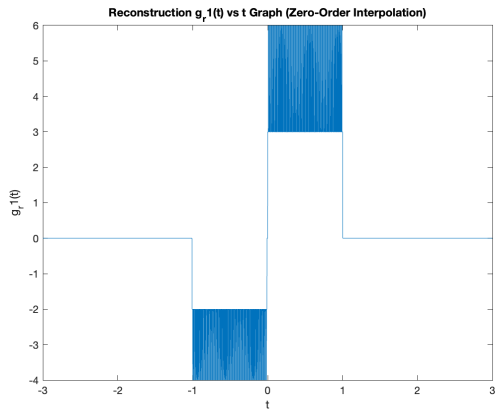
The code for the Part 5:

```
% create g(t)
Ts = 1/(20*randi([2,6]));
t = -3:Ts:3;

g = zeros(1,length(t));
g(2/Ts+1:3/Ts) = -2;
g(3/Ts+1) = 0;
g(3/Ts+2:4/Ts+1) = 3;

% plot g(n*Ts) in stem plot
stem(linspace(-3/Ts, 3/Ts, length(t)), g);

% generation of gr's
dur = 6;

gr1 = DtoA(0, Ts, dur, g);
gr2 = DtoA(1, Ts, dur, g);
gr3 = DtoA(2, Ts, dur, g);

figure;

plot(linspace(-3,3, length(gr1)), gr1);
xlabel('t');
ylabel('g_r1(t)');
title('Reconstruction g_r1(t) vs t Graph (Zero-Order
    Interpolation)');

figure;

plot(linspace(-3,3, length(gr2)), gr2);
xlabel('t');
ylabel('g_r2(t)');
title('Reconstruction g_r2(t) vs t Graph (Linear Interpolation)');

figure;

plot(linspace(-3,3, length(gr3)), gr3);
xlabel('t');
ylabel('g_r3(t)');
```

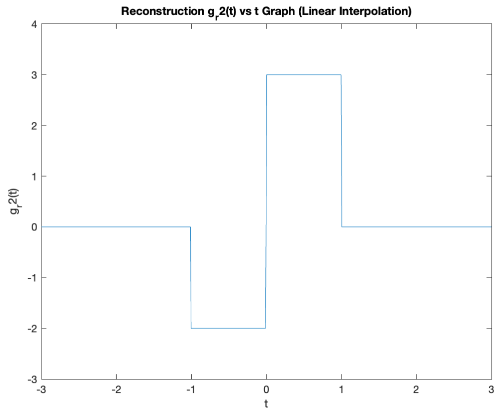
```
title('Reconstruction g_r3(t) vs t Graph (Ideal Interpolation)');

function [xR] = DtoA(type,Ts,dur,Xn)
    p = generateInterp(type, Ts, dur);
    l = length(Xn)*500+length(p);
    xR = zeros(1,l );
    for x = 1: length(Xn)
        xR(1+500*(x-1): 500*(x-1) + length(p)) = Xn(x)*p+
            xR(1+500*(x-1): 500*(x-1) +length(p));
    end
    xR = xR(250*length(Xn)+1:end-250*length(Xn));
end


function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
        p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```



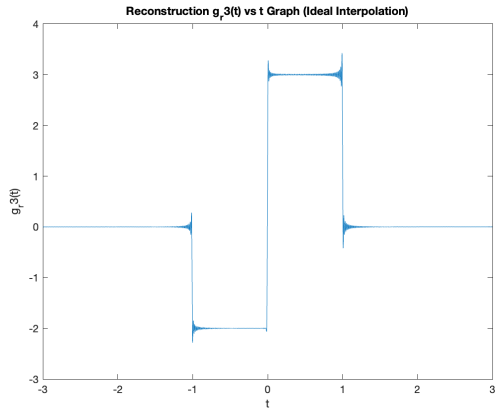
(a) $g(nT_s)$ stem plot of the signal.



(b) Recovered version of $g(t)$ by using zero-order interpolation.



(c) Recovered version of $g(t)$ by using linear interpolation.



(d) Recovered version of $g(t)$ by using ideal interpolation.

Figure 5: Part 5 graphs.

As the signal lacks band limitation, encompassing all frequencies, the ideal interpolation method did not yield the expected outcome. As T_s increases, the ideal interpolation gets worse and worse since the sampling rate decreases and it does not recover the function fully.

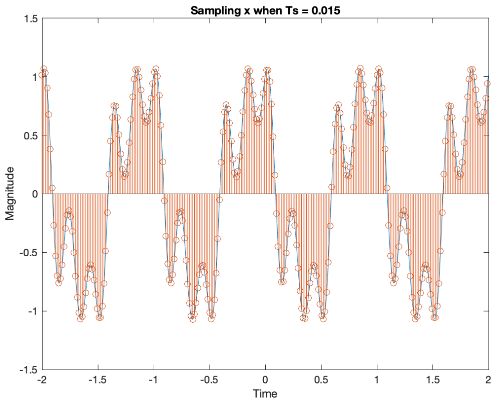
Part 6:

The code for the Part 6 for $T_s = 0.015$:

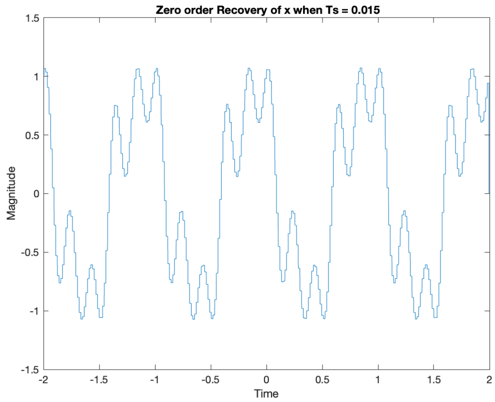
```
D = mod(22103132, 7);
Ts = 0.005*(D+1);
t_continuous = -2:Ts/1000:2;
t_sampling = -2:Ts:2;


x=0.25*cos(2*pi*3*t_continuous+pi/8)+0.4*
cos(2*pi*5*t_continuous-1.2)+0.9*cos(2*pi*t_continuous+pi/4);
Xn=0.25*cos(2*pi*3*t_sampling+pi/8)+0.4*
cos(2*pi*5*t_sampling-1.2)+0.9*cos(2*pi*t_sampling+pi/4);


plot(t_continuous,x);
hold on;
stem(t_sampling, Xn);
title("Sampling x when Ts = 0.015");
ylabel("Magnitude");
xlabel("Time");
hold off;

t_continuous = -2:Ts/500:2-Ts/500;

figure;
xR = DtoA(0, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Zero order Recovery of x when Ts = 0.015");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(1, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Linear Recovery of x when Ts = 0.015");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(2, Ts, 4, Xn);
plot(t_continuous, xR);
```

```
title(" Ideal Recovery of x when Ts = 0.015");
ylabel("Magnitude");
xlabel("Time");


function [xR] = DtoA(type,Ts,dur,Xn)
    p = generateInterp(type, Ts, dur);
    l = length(Xn)*500+length(p);
    xR = zeros(1,l );
    for x = 1: length(Xn)
        xR(1+500*(x-1): 500*(x-1) + length(p)) = Xn(x)*p+
            xR(1+500*(x-1): 500*(x-1) +length(p));
    end
    xR = xR(250*length(Xn)+1:end-250*length(Xn));
end


function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
            p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```

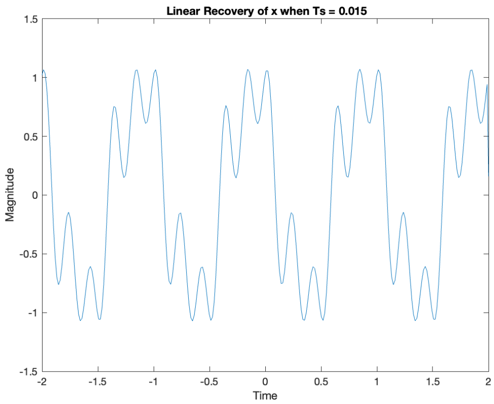
Graphs of the recovered signal for $T_s = 0.015$:



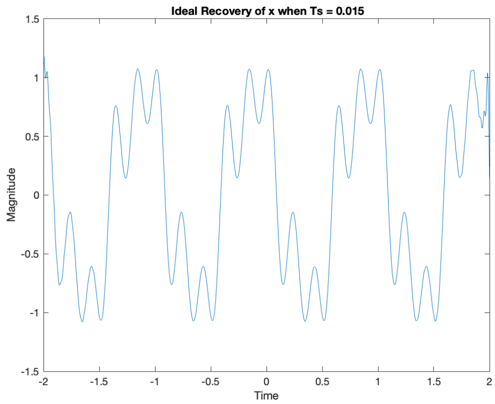
(a) Sampling of the signal.



(b) Zero-order recovery of the signal.



(c) Linear recovery of the signal.



(d) Ideal recovery of the signal.

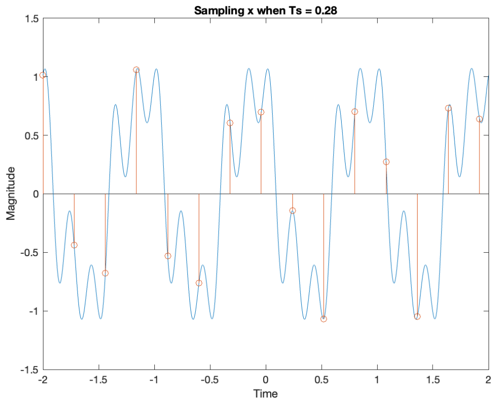
Figure 6: Graphs of the recovered signal.

The code for the Part 6 for $T_s = 0.28$:

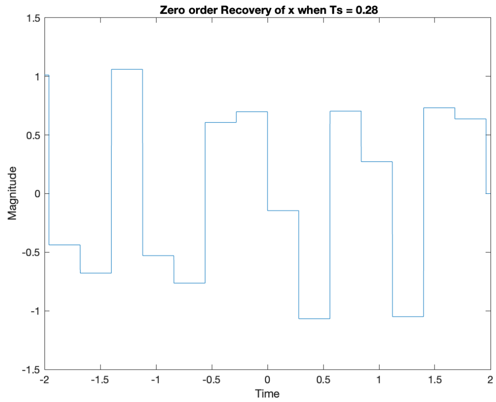
```
D = mod(22103132, 7);
Ts = 0.25+0.01*(D+1);
t_continuous = -2:Ts/1000:2;
t_sampling = -2:Ts:2;


x=0.25*cos(2*pi*3*t_continuous+pi/8)+0.4*
cos(2*pi*5*t_continuous-1.2)+0.9*cos(2*pi*t_continuous+pi/4);
Xn=0.25*cos(2*pi*3*t_sampling+pi/8)+0.4*
cos(2*pi*5*t_sampling-1.2)+0.9*cos(2*pi*t_sampling+pi/4);



plot(t_continuous,x);
hold on;
stem(t_sampling, Xn);
title("Sampling x when Ts = 0.28");
ylabel("Magnitude");
xlabel("Time");
hold off;

t_continuous = -2:Ts/500:2-Ts/500;

figure;
xR = DtoA(0, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Zero order Recovery of x when Ts = 0.28");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(1, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Linear Recovery of x when Ts = 0.28");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(2, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Ideal Recovery of x when Ts = 0.28");
```

```
ylabel("Magnitude");
xlabel("Time");


function [xR] = DtoA(type,Ts,dur,Xn)
    p = generateInterp(type, Ts, dur);
    l = length(Xn)*500+length(p);
    xR = zeros(1,l );
    for x = 1: length(Xn)
        xR(1+500*(x-1): 500*(x-1) + length(p)) = Xn(x)*p+
            xR(1+500*(x-1): 500*(x-1) +length(p));
    end
    xR = xR(250*length(Xn)+1:end-250*length(Xn));
end


function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
        p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```

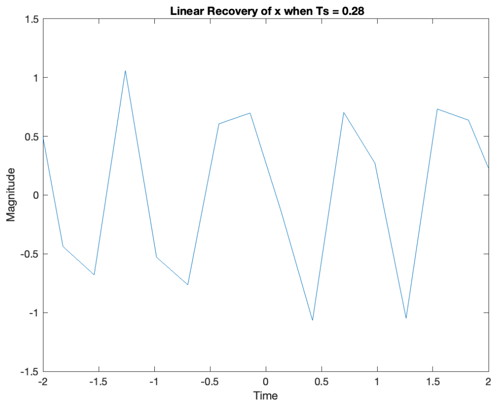
Graphs of the recovered signal for $T_s = 0.28$:



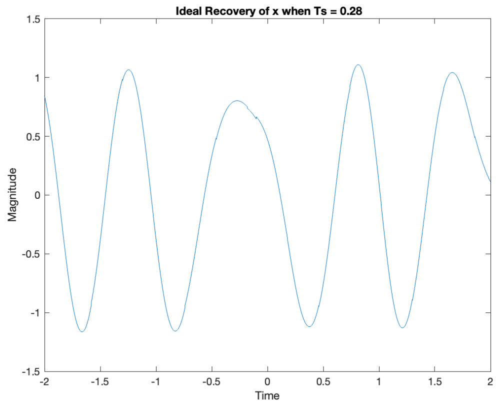
(a) Sampling of the signal.



(b) Zero-order recovery of the signal.



(c) Linear recovery of the signal.



(d) Ideal recovery of the signal.

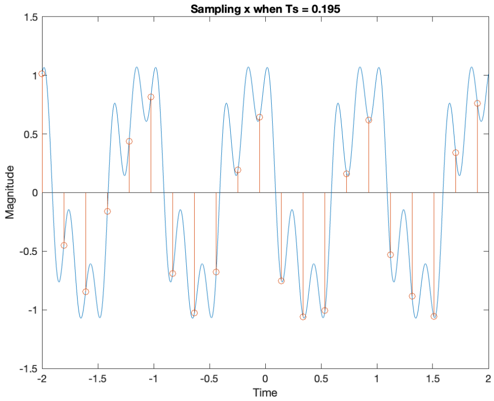
Figure 7: Graphs of the recovered signal.

The code for the Part 6 for $T_s = 0.195$:

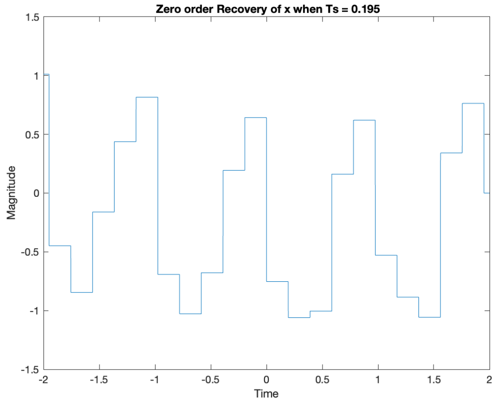
```
D = mod(22103132, 7);
Ts = 0.18+0.005*(D+1);
t_continuous = -2:Ts/1000:2;
t_sampling = -2:Ts:2;


x=0.25*cos(2*pi*3*t_continuous+pi/8)+0.4*
cos(2*pi*5*t_continuous-1.2)+0.9*cos(2*pi*t_continuous+pi/4);
Xn=0.25*cos(2*pi*3*t_sampling+pi/8)+0.4*
cos(2*pi*5*t_sampling-1.2)+0.9*cos(2*pi*t_sampling+pi/4);


plot(t_continuous,x);
hold on;
stem(t_sampling, Xn);
title("Sampling x when Ts = 0.195");
ylabel("Magnitude");
xlabel("Time");
hold off;

t_continuous = -2:Ts/500:2-Ts/500;

figure;
xR = DtoA(0, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Zero order Recovery of x when Ts = 0.195");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(1, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Linear Recovery of x when Ts = 0.195");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(2, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Ideal Recovery of x when Ts = 0.195");
ylabel("Magnitude");
```

```
xlabel("Time");


function [xR] = DtoA(type,Ts,dur,Xn)
    p = generateInterp(type, Ts, dur);
    l = length(Xn)*500+length(p);
    xR = zeros(1,l );
    for x = 1: length(Xn)
        xR(1+500*(x-1): 500*(x-1) + length(p)) = Xn(x)*p+
            xR(1+500*(x-1): 500*(x-1) +length(p));
    end
    xR = xR(250*length(Xn)+1:end-250*length(Xn));
end


function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
        p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```

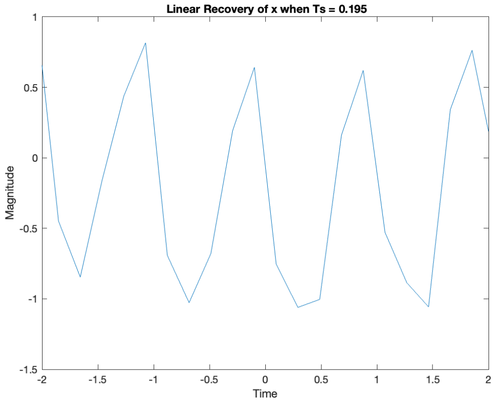
Graphs of the recovered signal for $T_s = 0.195$:



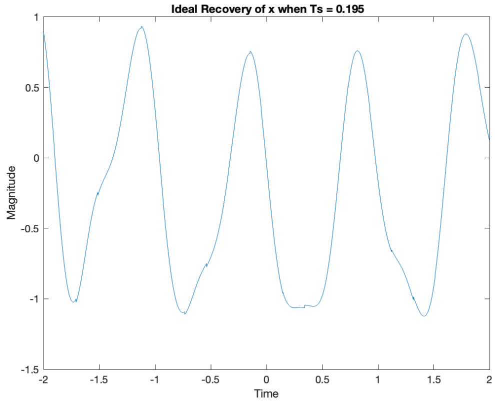
(a) Sampling of the signal.



(b) Zero-order recovery of the signal.



(c) Linear recovery of the signal.



(d) Ideal recovery of the signal.

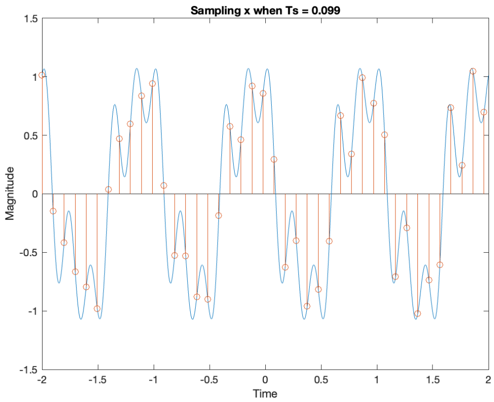
Figure 8: Graphs of the recovered signal.

The code for the Part 6 for $T_s = 0.099$:

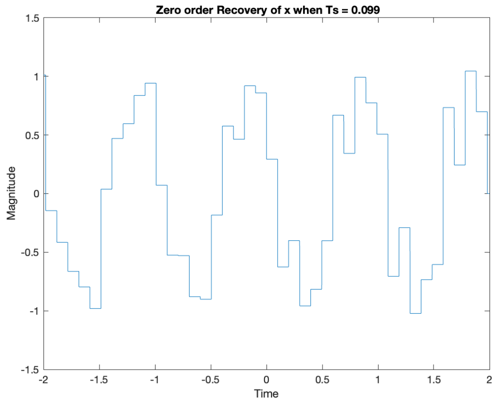
```
D = mod(22103132, 7);
Ts = 0.099;
t_continuous = -2:Ts/1000:2;
t_sampling = -2:Ts:2;


x=0.25*cos(2*pi*3*t_continuous+pi/8)+0.4*
cos(2*pi*5*t_continuous-1.2)+0.9*cos(2*pi*t_continuous+pi/4);
Xn=0.25*cos(2*pi*3*t_sampling+pi/8)+0.4*
cos(2*pi*5*t_sampling-1.2)+0.9*cos(2*pi*t_sampling+pi/4);



plot(t_continuous,x);
hold on;
stem(t_sampling, Xn);
title("Sampling x when Ts = 0.099");
ylabel("Magnitude");
xlabel("Time");
hold off;

t_continuous = -2:Ts/500:2-Ts/500;

figure;
xR = DtoA(0, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Zero order Recovery of x when Ts = 0.099");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(1, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Linear Recovery of x when Ts = 0.099");
ylabel("Magnitude");
xlabel("Time");

figure;
xR = DtoA(2, Ts, 4, Xn);
plot(t_continuous, xR);
title(" Ideal Recovery of x when Ts = 0.099");
```

```
ylabel("Magnitude");
xlabel("Time");


function [xR] = DtoA(type,Ts,dur,Xn)
    p = generateInterp(type, Ts, dur);
    l = length(Xn)*500+length(p);
    xR = zeros(1,l );
    for x = 1: length(Xn)
        xR(1+500*(x-1): 500*(x-1) + length(p)) = Xn(x)*p+
            xR(1+500*(x-1): 500*(x-1) +length(p));
    end
    xR = xR(250*length(Xn)+1:end-250*length(Xn));
end


function [p]=generateInterp(type,Ts,dur)
    t = -dur/2 : Ts/500 : dur/2 -Ts/500;
    % zero-order
    if type == 0
        p =zeros(1,length(t));
        p(t>=-1/2*Ts & t < 1/2*Ts) = 1;

    % linear
    elseif type == 1
        p = zeros(1, length(t));
        p(t>-Ts & t <Ts) = 1-abs(t(t>-Ts & t < Ts))/Ts;

    % ideal
    elseif type == 2
        p = sin(pi*t/Ts)./ (pi* t/Ts);
        p(t==0) = 1;
    end
end
```

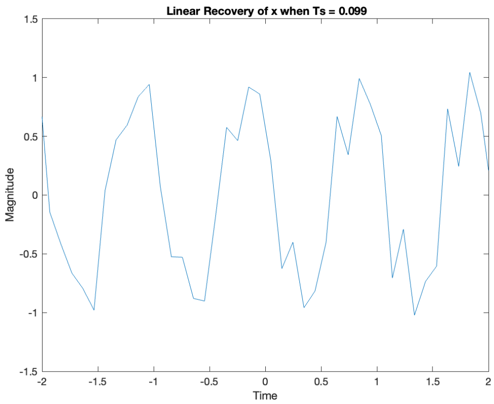
Graphs of the recovered signal for $T_s = 0.099$:



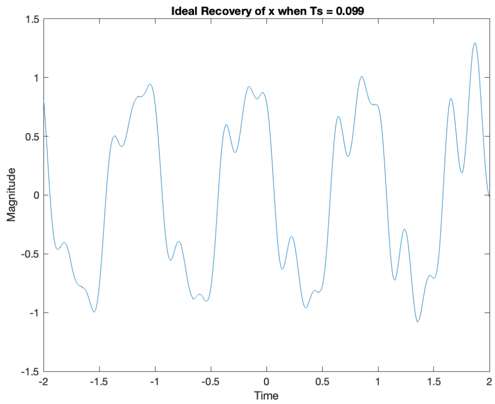
(a) Sampling of the signal.



(b) Zero-order recovery of the signal.



(c) Linear recovery of the signal.



(d) Ideal recovery of the signal.

Figure 9: Graphs of the recovered signal.

It is obvious that as T_s decreases, the recovery of the signal gets better. Using T_s below the Nyquist rate guarantees full signal recovery in these interpolation methods. A decreased T_s yields finer data recovery concerning the signal, leading to enhanced reconstructed signals. The sequence $T_{s_a} < T_{s_d} < T_{s_c} < T_{s_b}$ indicates the effectiveness of interpolations in the order of Part A > Part D > Part C > Part B.