A Questions

a)

Given a continuous time signal

$$x_c(t) = \begin{cases} e^{-at}\cos(\Omega_1 t) & t \geq 0, a > 0 \\ 0 & t < 0 \end{cases} \tag{1}$$

Driven an expression for the spectrum $X_c(j\Omega)$. Plot $x_c(t)$ and $X_c(j\Omega)$ in Matlab.

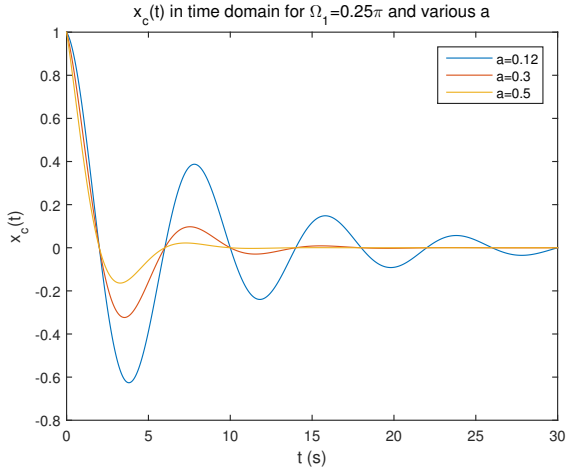

Figure 1: $x_c(t)$ for various a

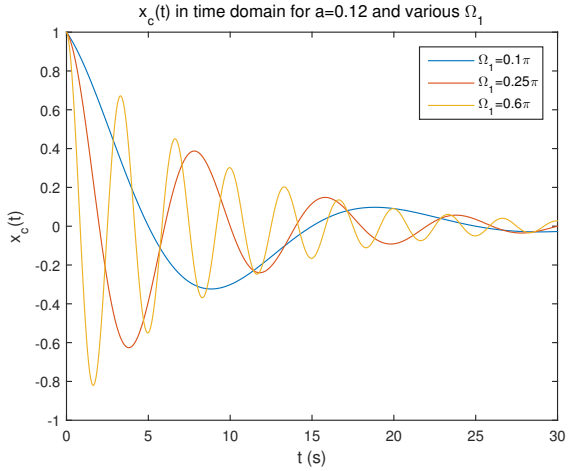

Figure 2: $x_c(t)$ for various Ω_1

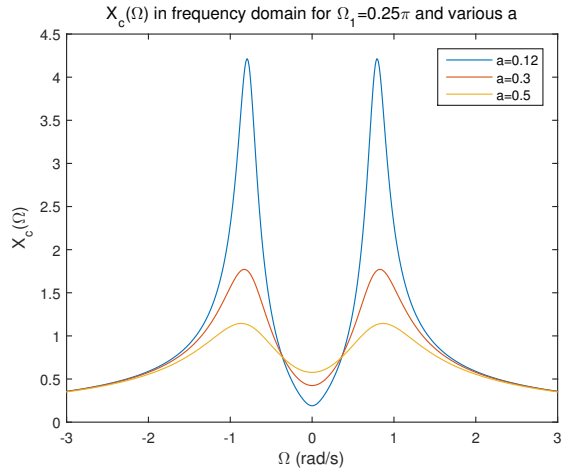

Figure 3: $X_c(\Omega)$ for various a

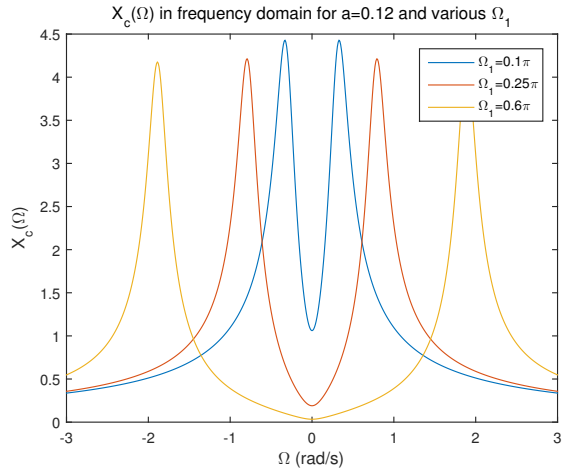

Figure 4: $X_c(\Omega)$ for various Ω_1

Take Fourier transform on Eq.1 to get the spectrum of the original signal.

$$
\begin{aligned}
X_c(j\Omega) &= \int_0^\infty e^{-at}\cos(\Omega_1 t)e^{-j\Omega t}dt \quad \text{(since the original signal only has value when } t \geq 0) \\
&= \int_0^\infty e^{-at}\frac{e^{j\Omega_1 t}+e^{-j\Omega_1 t}}{2}e^{-j\Omega t}dt \\
&= \frac{1}{2}\int_0^\infty e^{(-a+j\Omega_1-j\Omega)t}+e^{(-a-j\Omega_1-j\Omega)t}dt \\
&= \frac{1}{2}\ \frac{1}{-a+j\Omega_1-j\Omega}e^{(-a+j\Omega_1-j\Omega)t}\Big|_0^\infty + \frac{1}{2}\ \frac{1}{-a-j\Omega_1-j\Omega}e^{(-a-j\Omega_1-j\Omega)t}\Big|_0^\infty \\
&= \frac{1}{2}[\frac{-1}{-a+j(\Omega_1-\Omega)}+\frac{-1}{-a-j(\Omega_1+\Omega)}] \\
&= \frac{1}{2}\frac{a+j(\Omega_1+\Omega)+a-j(\Omega_1-\Omega)}{(-a+j(\Omega_1-\Omega))(-a-j(\Omega_1+\Omega))} \\
&= \frac{a+j\Omega}{a^2+2aj\Omega+\Omega_1^2-\Omega^2} \\
&= \frac{a+j\Omega}{(a+j\Omega)^2+\Omega_1^2}
\end{aligned}
$$

As a increases, signal decays more rapidly in time domain while the magnitudes become smaller and the peaks move slightly farther away from the symmetry axis in frequency domain.

When Ω_1 grows, signal fluctuates more sharply in time domain. In terms of frequency domain, the spectrum spread wider and the peaks move farther away from the symmetry axis. The magnitude decreases marginally.

b)

Let the sample version of $x_c(t)$ be $x[n] = x_c(nT)$. Calculate the DTFT $X(e^{j\omega})$ for the sampled signal.

The DTFT of discrete time signal $x[n]$ is

$$\begin{aligned}
X(e^{j\omega}) &= \sum_{-\infty}^{\infty} x[n]e^{-j\omega n} \\
&= \sum_{0}^{\infty} e^{-anT}\cos(\Omega_1 nT)e^{-j\omega n} \\
&= \sum_{0}^{\infty} e^{-anT}\frac{e^{j\Omega_1 nT}+e^{-j\Omega_1 nT}}{2}e^{-j\omega n} \\
&= \frac{1}{2}\sum_{0}^{\infty}(e^{-anT+j\Omega_1 nT-j\omega n}+e^{-anT-j\Omega_1 nT-j\omega n}) \\
&= \frac{1}{2}(\frac{1}{1-e^{-aT+j(\Omega_1 T-\omega)}}+\frac{1}{1-e^{-aT-j(\Omega_1 T+\omega)}}) \\
&= \frac{1}{2}\frac{2-e^{-aT-j(\Omega_1 T+\omega)}-e^{-aT+j(\Omega_1 T-\omega)}}{(1-e^{-aT+j(\Omega_1 T-\omega)})(1-e^{-aT-j(\Omega_1 T+\omega)})} \\
&= \frac{1-e^{-aT}\cos(\Omega_1 T)e^{-j\omega}}{1-2e^{-aT}\cos(\Omega_1 T)e^{-j\omega}+e^{-2aT}e^{-2j\omega}}
\end{aligned}$$

c)

Let $a = 0.12$, $\Omega_1 = 0.25\pi$ and let the sampling frequency F=4.8Hz. Plot $X_c(j\Omega)$ and $X(e^{j\omega})$ in Matlab.Do necessary scalings to match them up. Explain the command freqz in Matlab.

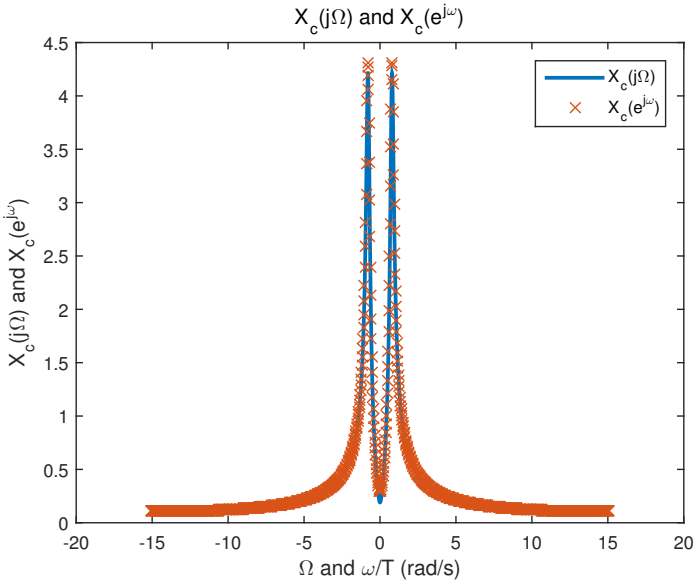

Figure 5: $X_c(j\Omega)$ and $X(e^{j\omega})$

According to the lecture slides $1-9/23$ and $1-18/23$, the process of recovering $X_c(j\Omega)$ from $X(e^{j\omega})$ depends on sampling period T.

$$X(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c(j(\Omega + k\Omega_T)) \tag{2}$$

In order to match the spectrum of continuous time signal, we will scale the magnitude of $X(e^{j\omega})$ by a factor of T, in this case, $T = \frac{1}{4.8}$. Since the spectrum of $X(e^{j\omega})$ has period of 2π, we need to convert the discrete ω into continuous Ω by $\Omega = \frac{\omega}{T}$.

In terms of the mode used in this question, three arguments are passed to the freqz function.
1. the coefficient before $e^{-jn\omega}$ of the numerator of $X(e^{j\omega})$
2. the coefficient before $e^{-jn\omega}$ of the denominator of $X(e^{j\omega})$
3. the ω range

d)

A simple first order digital lowpass filter is given by

$$H_{LP}(z) = \frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}} \tag{3}$$

Show that

$$|H_{LP}(e^{j\omega})|^2 = \frac{(1-\alpha)^2(1+\cos\omega)}{2(1+\alpha^2-2\alpha\cos\omega)} \tag{4}$$

For which values of α is the filter stable? Plot $|H_{LP}(e^{j\omega})|$ and $\angle H_{LP}(e^{j\omega})$ for some values of α. Comment on the results.

Substitute $z = e^{j\omega}$ into Eq.4, yields

$$\begin{aligned}
|H_{LP}(e^{j\omega})|^2 &= \left|\frac{1-\alpha}{2}\frac{1+e^{-j\omega}}{1-\alpha e^{-j\omega}}\right|^2 \\
&= \frac{(1-\alpha)^2}{4}\left|\frac{1+\cos\omega-j\sin\omega}{1-\alpha(\cos\omega-j\sin\omega)}\right|^2 \\
&= \frac{(1-\alpha)^2}{4}\frac{(1+\cos\omega)^2+(-\sin\omega)^2}{(1-\alpha\cos\omega)^2+(-\alpha\sin\omega)^2} \\
&= \frac{(1-\alpha)^2}{4}\frac{2+2\cos\omega}{1-2\alpha\cos\omega+\alpha^2} \\
&= \frac{(1-\alpha)^2(1+\cos\omega)}{2(1+\alpha^2-2\alpha\cos\omega)}
\end{aligned}$$

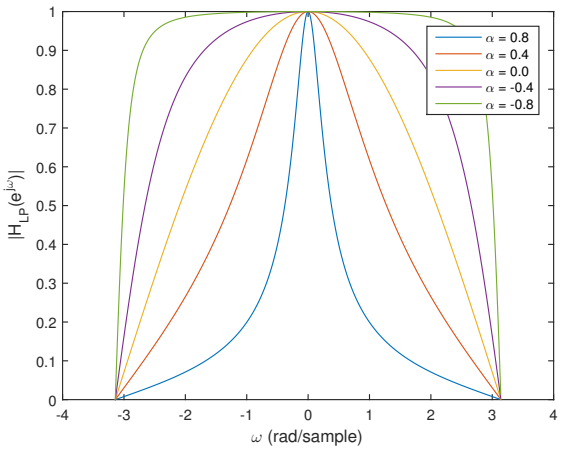

Figure 6: Plot of $|H_{LP}(e^{j\omega})|$

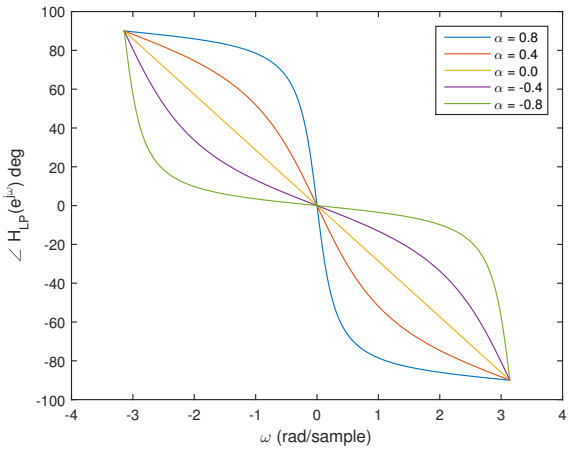

Figure 7: Plot of $\angle H_{LP}(e^{j\omega})$

The pole of the filter $z = \alpha$ should be strictly inside the unit circle, i.e.

$$|\alpha| < 1 \tag{5}$$

For simplicity, only positive ω will be discussed, because frequency domain plots are symmetric about $\omega = 0$.

For bigger α, e.g. 0.8 comparing with 0.4 or - 0.8, the magnitude decreases more quickly at lower frequencies and more slowly at higher frequencies. However, for smaller α, the magnitude decreases more slowly at lower frequencies and decreases more quickly at higher frequencies. When α increases, the low-pass filter becomes more capable to reject high frequency components.

For bigger α, the phase decreases more slowly at lower frequencies and more quickly at higher frequencies. However, for smaller α, the phase decreases more quickly at lower frequencies and decreases more slowly at higher frequencies. However, for larger α, huger phase shift is inevitable.

B Implementation of digital anti-aliasing filters on a DSP

B.4 Lab tasks

a)

Increase the length of the signal to 256, Figure 8 and 9 are the signal in time domain and frequency domain. As can be seen in Figure 9, the spectrum achieve a peak value at $f = 0.125$Hz which equals to $\Omega_1 = 0.25\pi$ in code SPWS1.c.

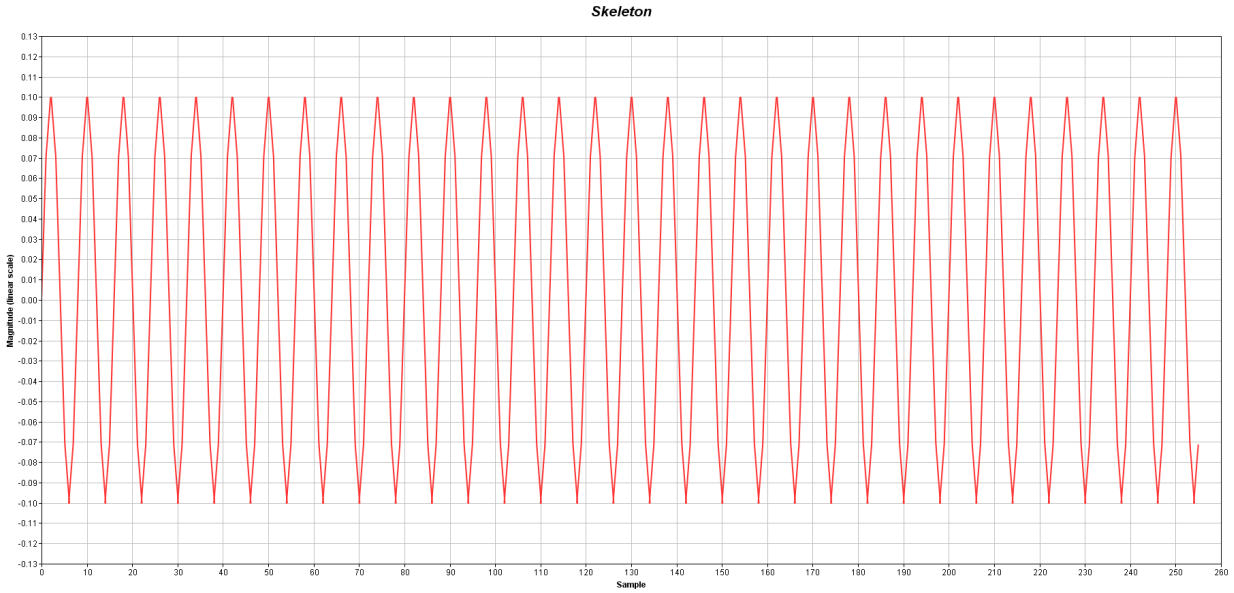

Figure 8: skeleton function in time domain

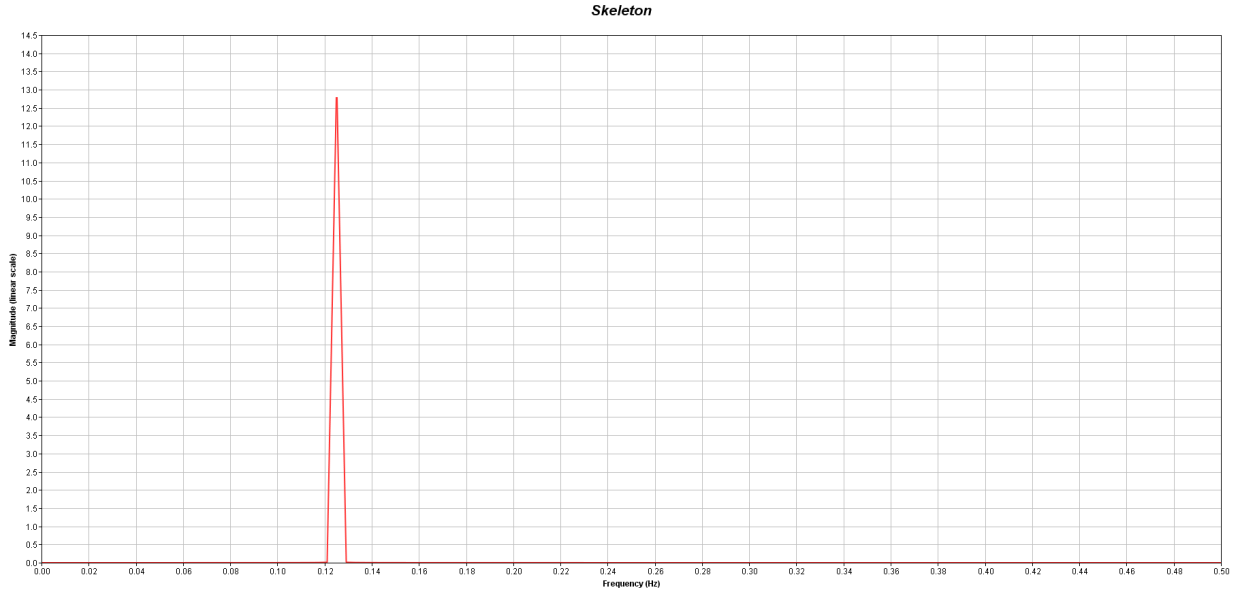

Figure 9: skeleton function in frequency domain

c)

Figure 10, 12 and 13 are plotted by MATLAB. Figure 11, 14 and 15 are plotted by CCES.

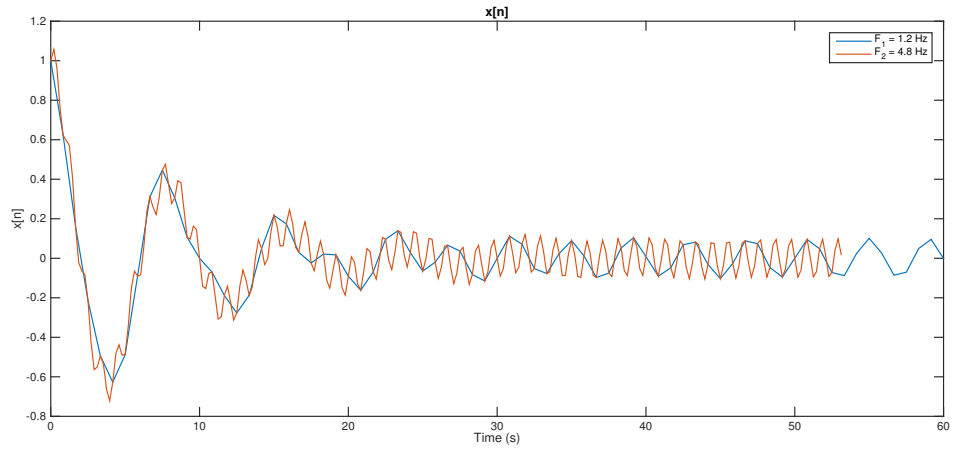

Figure 10: $x[n]$ in time domain (MATLAB)

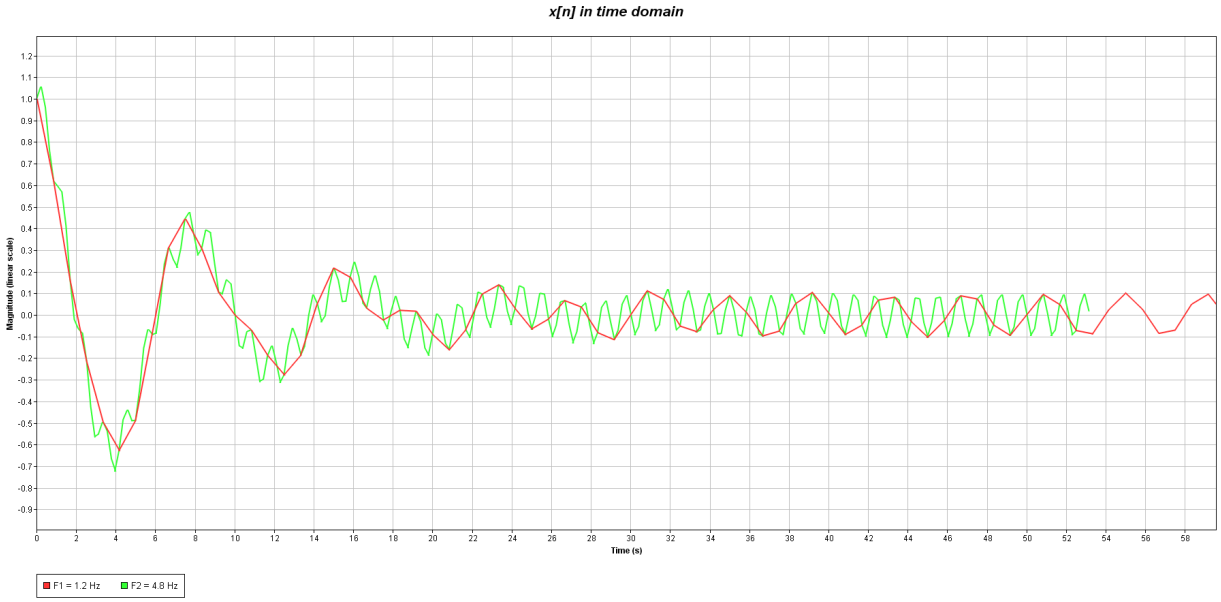

Figure 11: $x[n]$ in time domain (CCES)

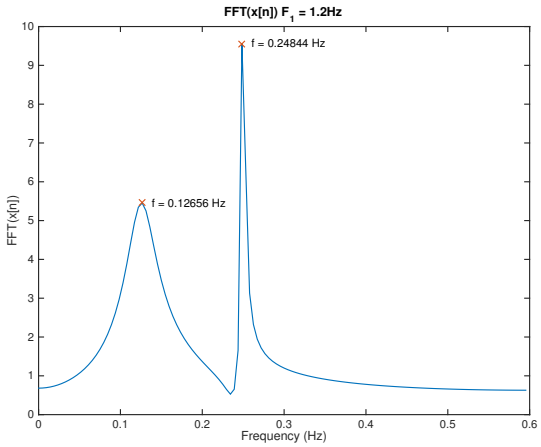

Figure 12: $x[n]$ (1.2Hz) (MATLAB)

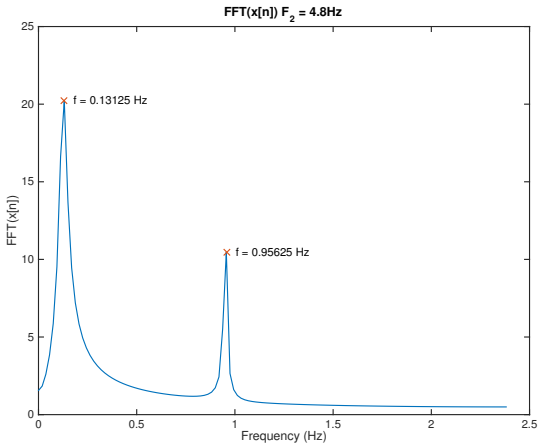

Figure 13: $x[n]$ (4.8Hz) (MATLAB)

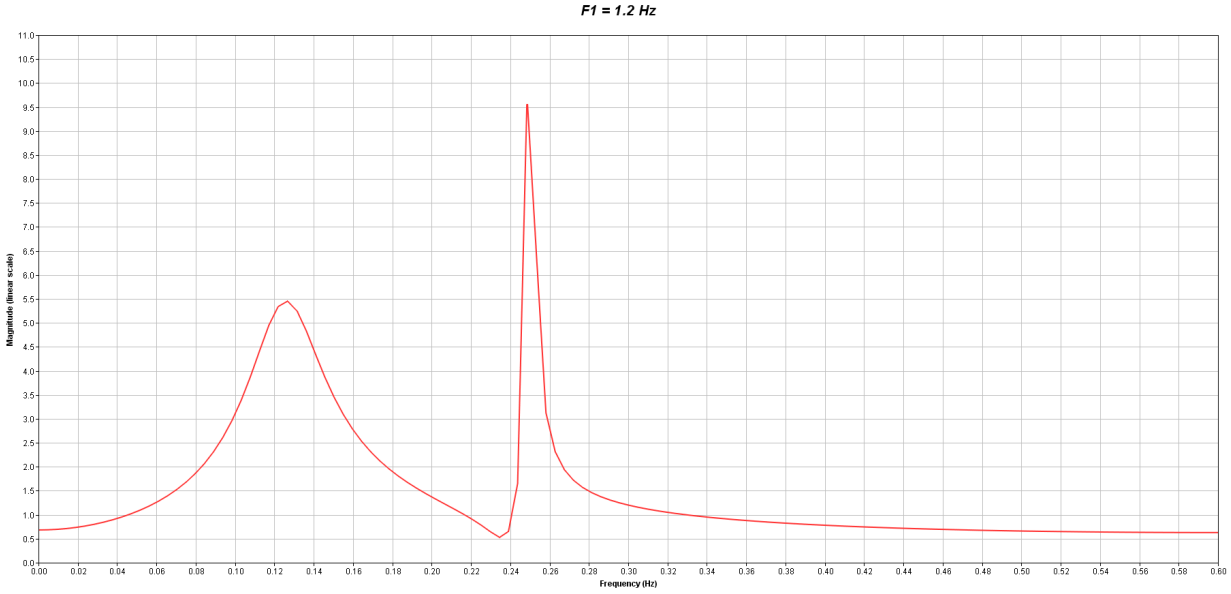

Figure 14: $x[n]$ (1.2Hz) in frequency domain (CCES)

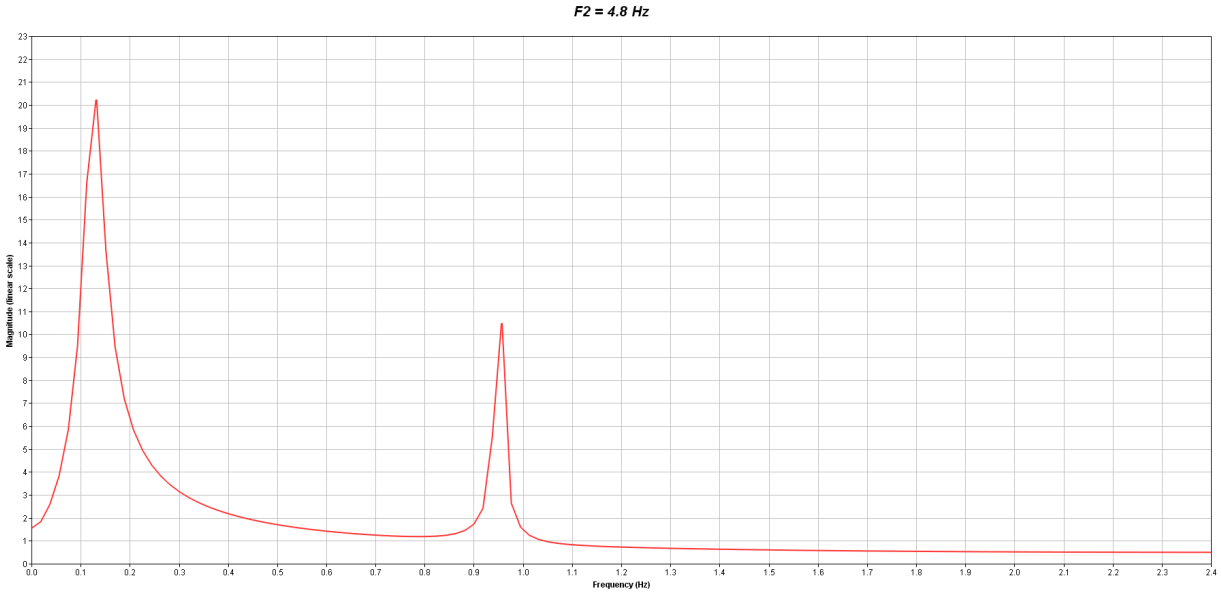

Figure 15: $x[n]$ (4.8Hz) in frequency domain (CCES)

In FFT plots, peaks near $F = \frac{\Omega_1}{2\pi} = \frac{0.25\pi}{2\pi} = 0.125$ Hz and $F = \frac{\Omega_2}{2\pi} = \frac{1.9\pi}{2\pi} = 0.95$ Hz are expected. Figure 13 and Figure 15 (sampling frequency $F_2 = 4.8$Hz) are in agreement with this expectation. However, an unexpected peak appears at $f \approx 0.24844$ Hz in Figure 12 and Figure 14 (sampling frequency $F_1 = 1.2$Hz). Interestingly, we find $\frac{0.24844+0.95}{2}$ Hz ≈ 0.6 Hz $= \frac{F_1}{2}$, which means this unexpected peak and the frequency of $x_2(t)$ are symmetric around $f = \frac{F_1}{2}$ (folding frequency).

In time domain, as is shown in Figure 10 and Figure 11, when sampling frequency $F_2 = 4.8$Hz, higher frequency (1.9π rad/s) fluctuation is kept; when sampling frequency $F_1 = 1.2$Hz, the oscillation frequency decreases which means higher frequency component becomes distorted.

In conclusion, $F_2 = 4.8$Hz is a proper sampling frequency such that all the information of the original signal is obtained without aliasing and folding.

d)

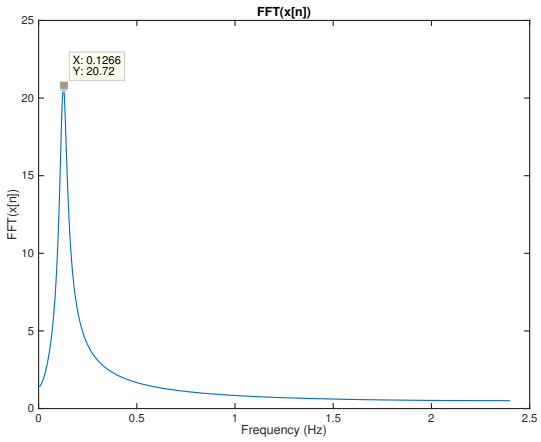

Figure 16: Plot of $X_1(F)$

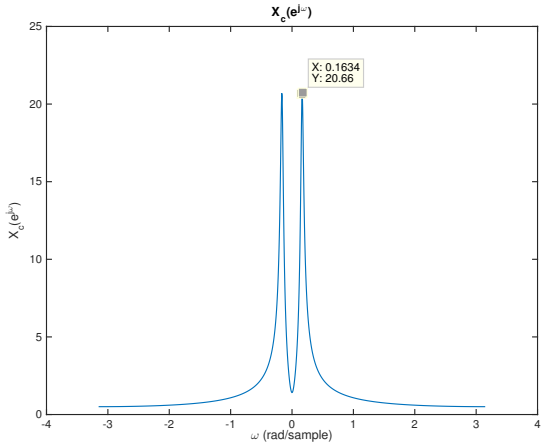

Figure 17: Plot of $X_1(e^{j\omega})$

As can be seen from the FFT plot (Figure 16), $F_{max} = 0.1266$ Hz, $\omega_{max} = \Omega_{max} T_S = \frac{0.1266}{4.8} \times 2\pi = 0.1657$ rad/sample.
As is shown in the DTFT $X(e^{j\omega})$ plot (Figure 17), $\omega_{max} = 0.1634$ rad/sample.

Two values of ω_{max} are consistent, we choose $\omega_{max} =$ **0.165 rad/sample**.

The frequency of the sinusoidal signal $x_2[n]$: $\omega_2 = \frac{1.9\pi}{4.8}$ rad/sample.

(i) a first order filter

From Eq.4, we obtain:

$$|H_{LP}(e^{j\omega})| = \frac{|1-\alpha|}{\sqrt{2}} \frac{\sqrt{1+\cos\omega}}{\sqrt{1+\alpha^2-2\alpha\cos\omega}} \tag{6}$$

According to the design specification, $|H_{LP}(e^{j\omega_{max}})| = 0.95$ at $\omega_{max} = 0.165$ rad/sample, i.e.

$$|H_{LP}(e^{j\omega_{max}})| = \frac{|1-\alpha_1|}{\sqrt{2}} \frac{\sqrt{1+\cos\omega_{max}}}{\sqrt{1+\alpha_1^2-2\alpha_1\cos\omega_{max}}} = 0.95 \tag{7}$$

Eq.7 can be solved by MATLAB (assuming $-1 < \alpha_1 < 1$): $\alpha_1 =$ **0.597991**.

At the frequency ω_2, **the 1st order filter does not satisfy the design specification**, because

$$|H_{LP}(e^{j\omega_2})| = \frac{|1-\alpha_1|}{\sqrt{2}} \frac{\sqrt{1+\cos\omega_2}}{\sqrt{1+\alpha_1^2-2\alpha_1\cos\omega_2}} = 0.33 > 0.25 \tag{8}$$

(ii) a second order filter

From Eq.4, we obtain:

$$|H_2(e^{j\omega})| = |H_{LP}(e^{j\omega})|^2 = \frac{(1-\alpha)^2(1+\cos\omega)}{2(1+\alpha^2-2\alpha\cos\omega)} \tag{9}$$

According to the design specification, $|H_2(e^{j\omega_{max}})| = 0.95$ at $\omega_{max} = 0.165$ rad/sample, i.e.

$$|H_2(e^{j\omega_{max}})| = |H_{LP}(e^{j\omega_{max}})|^2 = \frac{(1-\alpha_2)^2(1+\cos\omega_{max})}{2(1+\alpha_2^2-2\alpha_2\cos\omega_{max})} = 0.95 \tag{10}$$

Eq.10 can be solved by MATLAB (assuming $-1 < \alpha_2 < 1$): $\alpha_2 = \mathbf{0.470126}$.

At the frequency ω_2, **the 2nd order filter satisfies the design specification**, because

$$|H_2(e^{j\omega_2})| = |H_{LP}(e^{j\omega_2})|^2 = \frac{(1-\alpha_2)^2(1+\cos\omega_2)}{2(1+\alpha_2^2-2\alpha_2\cos\omega_2)} = 0.20 < 0.25 \tag{11}$$

```
clear;

omega_max = 0.165;
F_s = 4.8;
omega_2 = 1.9 * pi / F_s;

syms unknown;
assume(-1<unknown<1);
eqn = abs(1 - unknown) / sqrt(2) * sqrt(1 + cos(omega_max)) / sqrt(1 + unknown^2 - 2 *
    unknown * cos(omega_max)) == 0.95;
result = solve(eqn, unknown);
alpha_1st = double(result);

syms unknown;
assume(-1<unknown<1);
eqn = (1 - unknown)^2 / 2 * (1 + cos(omega_max)) / (1 + unknown^2 - 2 * unknown * cos(
    omega_max)) == 0.95;
result = solve(eqn, unknown);
alpha_2nd = double(result);

gain_1st = abs(1 - alpha_1st) / sqrt(2) * sqrt(1 + cos(omega_2)) / sqrt(1 + alpha_1st^2 -
    2 * alpha_1st * cos(omega_2));
gain_2nd = (1 - alpha_2nd)^2 / 2 * (1 + cos(omega_2)) / (1 + alpha_2nd^2 - 2 * alpha_2nd *
     cos(omega_2));

fprintf('1st order alpha = %f\n', alpha_1st);
fprintf('2nd order alpha = %f\n', alpha_2nd);

fprintf('1st order gain at omega_2 = %f > 0.25\n', gain_1st);
fprintf('2nd order gain at omega_2 = %f < 0.25\n', gain_2nd);
```

e)

First order filter time domain representation

$$
\begin{aligned}
Y(z) &= X(z)H_{LP}(z) \\
Y(z) &= X(z)\frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}} \\
Y(z)(1-\alpha z^{-1}) &= X(z)\frac{1-\alpha}{2}(1+z^{-1}) \\
Y(z) - \alpha Y(z)z^{-1} &= \frac{1-\alpha}{2}X(z) + \frac{1-\alpha}{2}X(z)z^{-1} \\
y[n] - \alpha y[n-1] &= \frac{1-\alpha}{2}x[n] + \frac{1-\alpha}{2}x[n-1] \text{ (inverse Z transform)} \\
y[n] &= \frac{1-\alpha}{2}x[n] + \frac{1-\alpha}{2}x[n-1] + \alpha y[n-1]
\end{aligned}
$$

Second order filter time domain representation

$$
\begin{aligned}
Y(z) &= X(z)(H_{LP}(z))^2 \\
Y_1(z) &:= X(z)H_{LP}(z) \\
Y(z) &= Y_1(z)H_{LP}(z) \\
y_1[n] &= \frac{1-\alpha}{2}x[n] + \frac{1-\alpha}{2}x[n-1] + \alpha y_1[n-1] \text{ (inverse Z transform)} \\
y[n] &= \frac{1-\alpha}{2}y_1[n] + \frac{1-\alpha}{2}y_1[n-1] + \alpha y[n-1]
\end{aligned}
$$

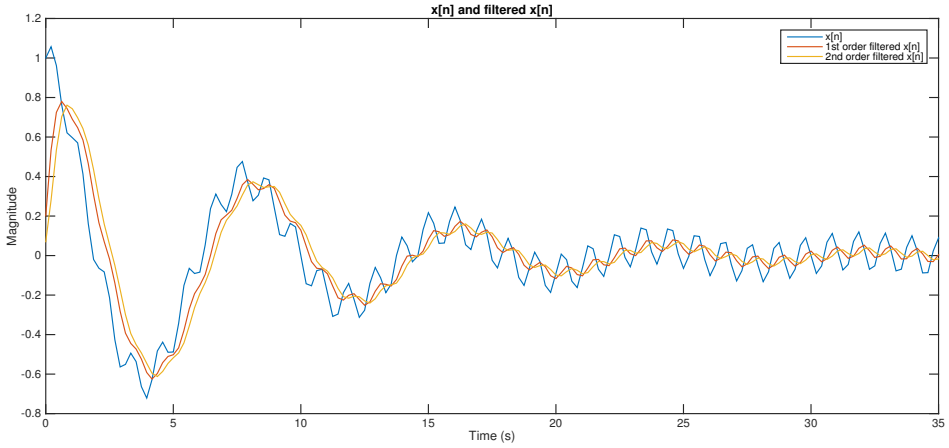

Figure 18: $x[n]$ and filtered $x[n]$ in time domain (MATLAB)

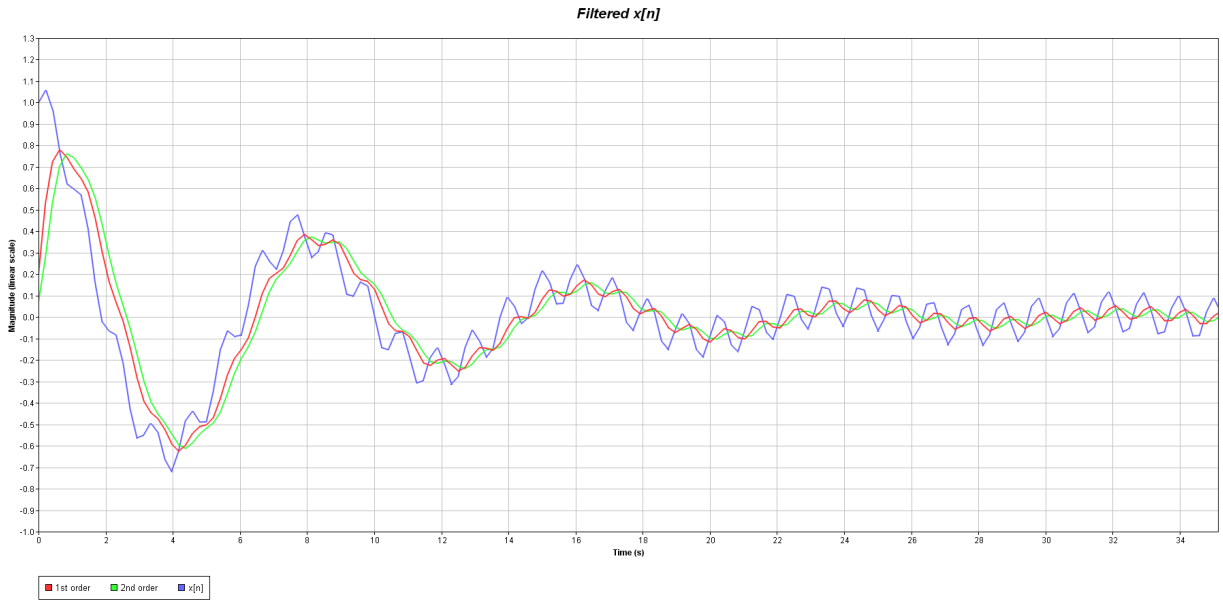

Figure 19: $x[n]$ and filtered $x[n]$ in time domain (CCES)

As can bee seen from Figure 18 and Figure 19, the magnitude of the fast oscillation period of the original signal has a peak-to-peak value 0.19 roughly. While the output signal of the first order filter is around 0.06 (larger than 25% of 0.19). The output from the second order filter yields a 0.04 peak-to-peak value (21% of 0.19) which satisfies the gain condition.

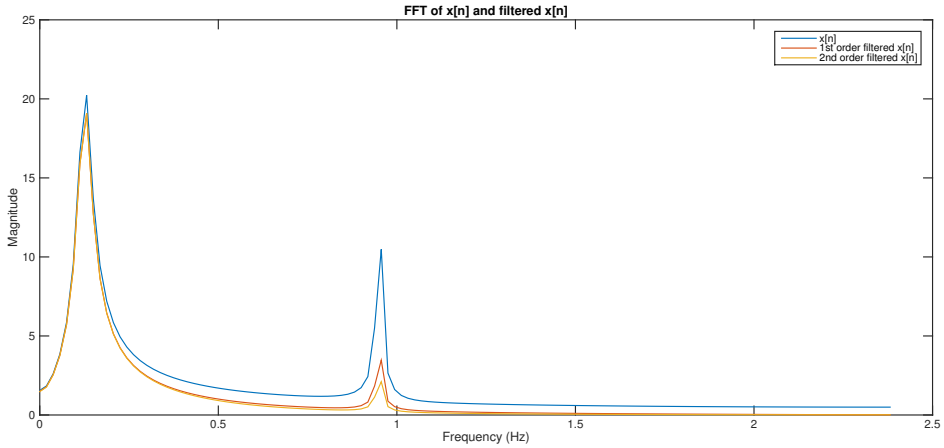

Figure 20: $x[n]$ and filtered $x[n]$ in frequency domain (MATLAB)

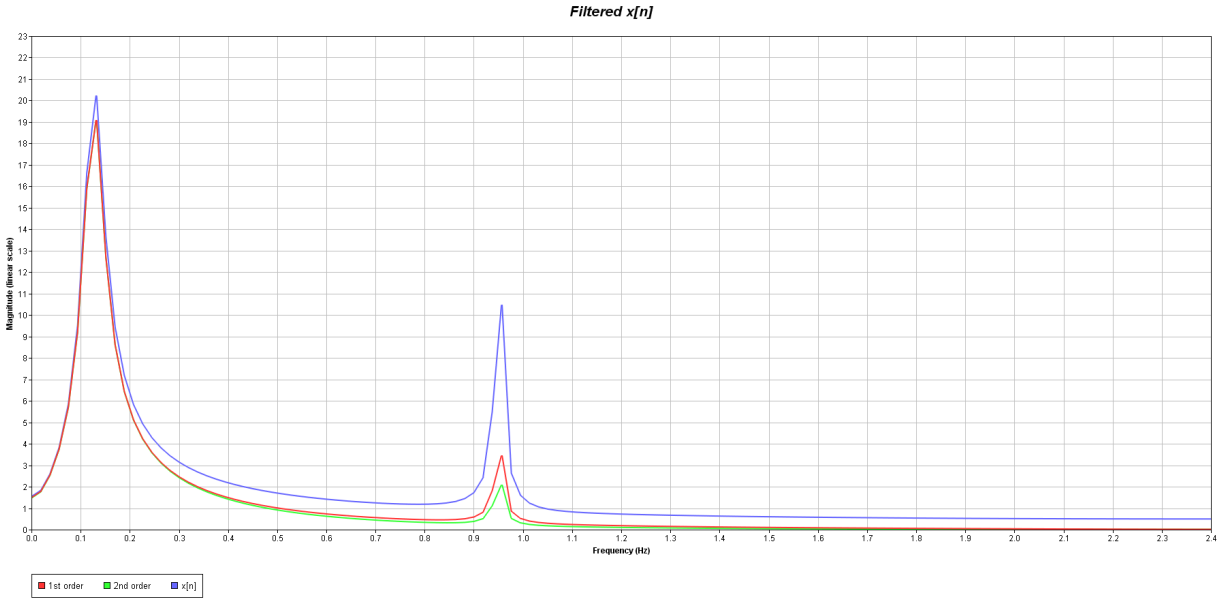

Figure 21: $x[n]$ and filtered $x[n]$ in frequency domain (CCES)

Figure 20 and Figure 21 show analogous results. Looking at the frequency around 0.95Hz, the magnitudes are 10.5, 3.5 and 2 respectively. Apparently,the second order filter successfully reduces the original signal down to about 20% at frequency around 0.95Hz. Looking at frequency around 0.125Hz, the magnitudes are 20.1, 19.1 and 19.1 respectively. That is, the outputs from both filters achieve 95% magnitude value around 0.125Hz.

In conclusion, we eventually remove the component produced by $x_2[n]$ and keep the component induced by $x_1[n]$.

f)

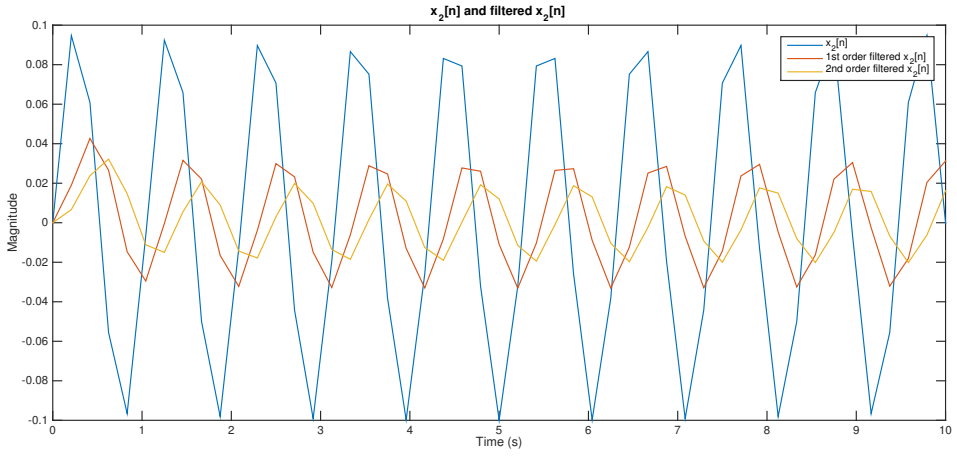

Figure 22: $x_2[n]$ and filtered $x_2[n]$ in time domain (MATLAB)

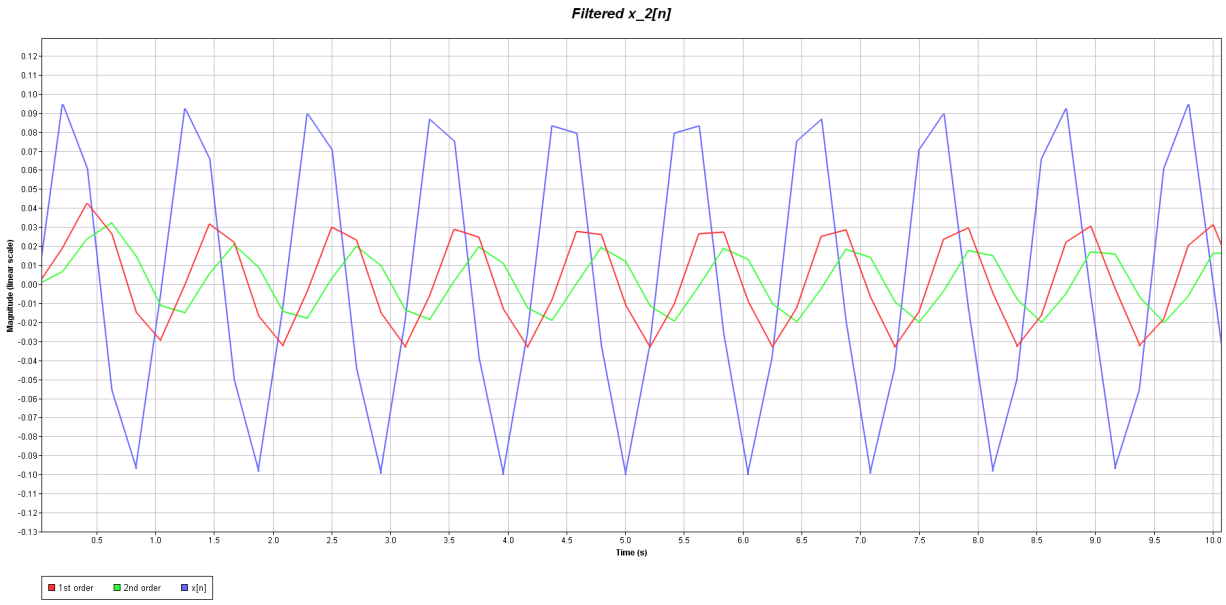

Figure 23: $x_2[n]$ and filtered $x_2[n]$ in time domain (CCES)

From Figure 22 and Figure 23, we can verify the gain by measuring the peak-to-peak value of each output.The original signal $x_2[n]$ has a peak-to-peak value 0.19 while the output of a second-order filter has 0.04 peak to peak value (21% of 0.19) roughly. So the gain meets the requirement.

Second-order filter has better performance on gain attenuation, but at the expense of inducing larger phase shift. Time delay is more dramatic in second-order filter than the first-order filter.

g)

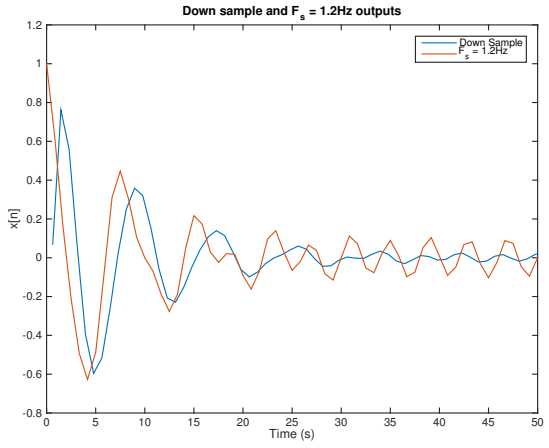

Figure 24: Down sample in time domain

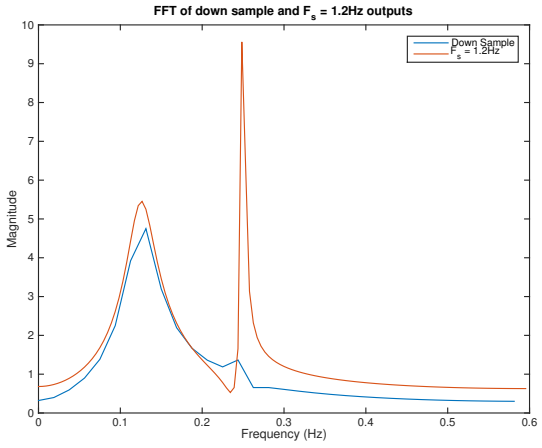

Figure 25: Down sample in frequency domain

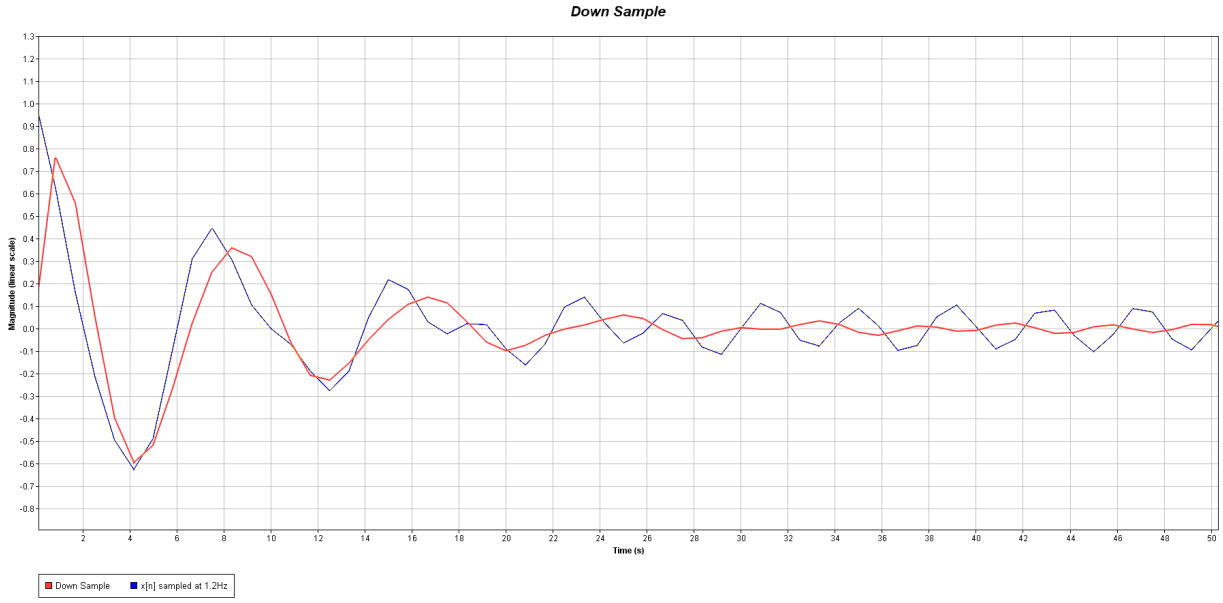

Figure 26: Down sample in time domain (CCES)

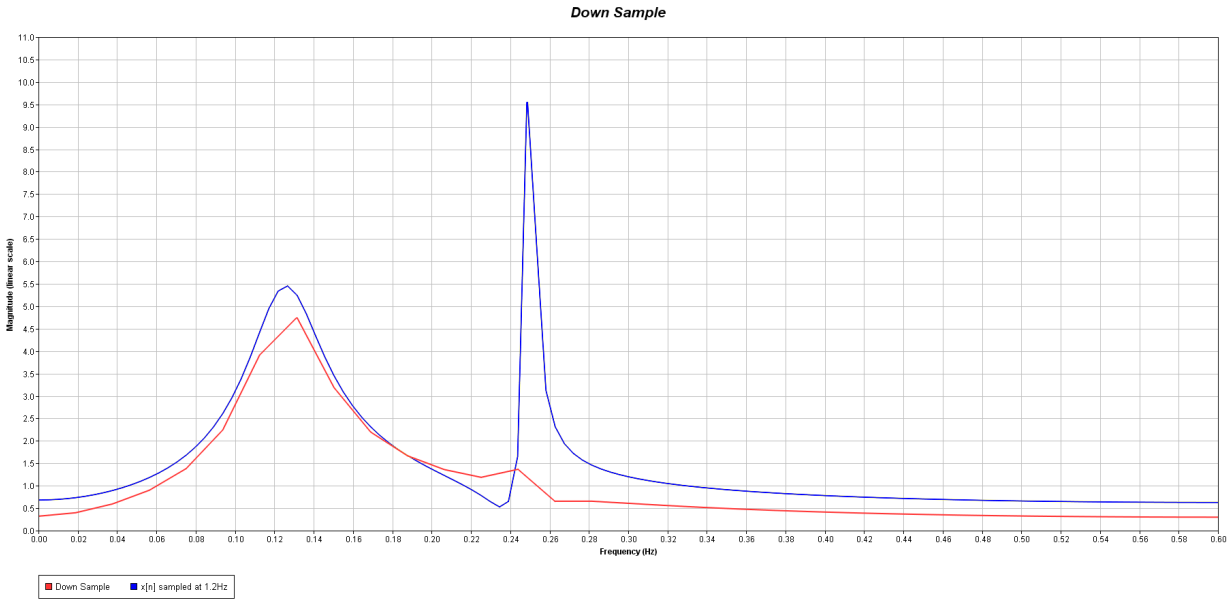

Figure 27: Down sample in frequency domain (CCES)

Based on our observation, in time domain, after down sampling $y[n]$, the peak-to-peak value of $y_{ds}[n]$ shrinks significantly which means we have removed the high frequency component in $y[n]$.

In frequency domain, seeing from Figure 25 and 27 , after down sampling, the folded peak around 0.248Hz disappears. Sampling frequency 4.8Hz is faster than twice the highest frequency of interest ($2 \times 1.9\pi$ rad/s). The original signal is scarcely distorted after sampling. The second-order filter works as a digital anti-aliasing filter, high frequency components (e.g. at 0.95Hz) are attenuated effectively. After down sampling, de facto sampling frequency becomes 1.2Hz. 1.2Hz is still higher than twice of 0.25π rad/s, hence no distortion occurs.

Appendix

SPWS1.c

```
#include <stdio.h>
#include <math.h>

#define N       256
#define PI      3.1415926

float a = 0.12;
float Omega_1 = 0.25 * PI;
float Omega_2 = 1.9 * PI;

float alpha_1st = 0.597991;
float alpha_2nd = 0.470126;

float t_12[N];
float t_48[N];
float t_ds[N/4];

float x_12Hz[N];
float x_2[N];
float x_48Hz[N];

float y_1st[N];
float y_2nd_intermediate[N];
float y_2nd[N];
float y_2_1st[N];
float y_2_2nd_intermediate[N];
float y_2_2nd[N];
float y_ds[N/4];

// print the values of an array in order to compare with the MATLAB variables.
int print_array(const float *array, int length) {
    for (int i = 0; i < length; i++) {
        printf("%f\t", *(array+i));
    }
    printf("\n");
    return 0;
}

int main(void) {
    int i;

    float T_1 = 1 / 1.2;
    float T_2 = 1 / 4.8;

    float t;
    for (i = 0; i < N; i++) {
        t = T_1 * i;
        x_12Hz[i] = exp(-a * t) * cos(Omega_1 * t) + 0.1 * sin(Omega_2 * t);
        // In this case, we do not think type casting is useful primarily because the
            arguments and returns value of exp(), sin(), cos() function are all double
            floating point.
        t_12[i] = t;

        t = T_2 * i;
        x_2[i] = 0.1 * sin(Omega_2 * t);
        x_48Hz[i] = exp(-a * t) * cos(Omega_1 * t) + x_2[i];
        t_48[i] = t;
```

```
    }
    // print_array(x_12Hz, N);
    // print_array(x_48Hz, N);
    printf("Signal Generated.\n");

    // y 1st order
    y_1st[0] = (1 - alpha_1st) / 2 * x_48Hz[0];
    for (i = 1; i < N; i++) {
        y_1st[i] = (1 - alpha_1st) / 2 * (x_48Hz[i] + x_48Hz[i-1]) + alpha_1st * y_1st[i-1];
    }
    // print_array(y_1st, N);

    // y 2nd order
    y_2nd_intermediate[0] = (1 - alpha_2nd) / 2 * x_48Hz[0];
    for (i = 1; i < N; i++) {
        y_2nd_intermediate[i] = (1 - alpha_2nd) / 2 * (x_48Hz[i] + x_48Hz[i-1]) + alpha_2nd
            * y_2nd_intermediate[i-1];
    }
    y_2nd[0] = (1 - alpha_2nd) / 2 * y_2nd_intermediate[0];
    for (i = 1; i < N; i++) {
        y_2nd[i] = (1 - alpha_2nd) / 2 * (y_2nd_intermediate[i] + y_2nd_intermediate[i-1]) +
             alpha_2nd * y_2nd[i-1];
    }
    // print_array(y_2nd, N);
    printf("e) done.\n");

    // y 1st order
    y_2_1st[0] = (1 - alpha_1st) / 2 * x_2[0];
    for (i = 1; i < N; i++) {
        y_2_1st[i] = (1 - alpha_1st) / 2 * (x_2[i] + x_2[i-1]) + alpha_1st * y_2_1st[i-1];
    }
    // print_array(y_2_1st, N);

    // y 2nd order
    y_2_2nd_intermediate[0] = (1 - alpha_2nd) / 2 * x_2[0];
    for (i = 1; i < N; i++) {
        y_2_2nd_intermediate[i] = (1 - alpha_2nd) / 2 * (x_2[i] + x_2[i-1]) + alpha_2nd *
            y_2_2nd_intermediate[i-1];
    }
    y_2_2nd[0] = (1 - alpha_2nd) / 2 * y_2_2nd_intermediate[0];
    for (i = 1; i < N; i++) {
        y_2_2nd[i] = (1 - alpha_2nd) / 2 * (y_2_2nd_intermediate[i] + y_2_2nd_intermediate[i
            -1]) + alpha_2nd * y_2_2nd[i-1];
    }
    // print_array(y_2_2nd, N);
    printf("f) done.\n");

    // down sample
    for (i = 0; i < N/4; i++) {
        y_ds[i] = y_2nd[i*4];
        t_ds[i] = t_48[i*4];
    }
    // print_array(y_ds, N/4);
    printf("g) done.\n");

    return 0;
}
```

A a)

```
%% Question a
clear;
close all;
t = 0:0.01:30;
Omega_1 = 0.25 * pi;

a = 0.12;
x_c = exp(-a * t) .* cos(Omega_1 * t);
figure;
plot(t, x_c);

a = 0.3;
x_c = exp(-a * t) .* cos(Omega_1 * t);
hold on;
plot(t, x_c);

a = 0.5;
x_c = exp(-a * t) .* cos(Omega_1 * t);
hold on;
plot(t, x_c);

xlabel('t (s)');
ylabel('x_c(t)');
title('x_c(t) in time domain for \Omega_1=0.25\pi and various a');
legend('a=0.12', 'a=0.3', 'a=0.5');

clear;
Omega = -3:0.01:3;
Omega_1 = 0.25 * pi;

a = 0.12;
X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);
figure;
plot(Omega, X_c);

a = 0.3;
X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);
hold on;
plot(Omega, X_c);

a = 0.5;
X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);
hold on;
plot(Omega, X_c);

xlabel('\Omega (rad/s)');
ylabel('X_c(\Omega)');
title('X_c(\Omega) in frequency domain for \Omega_1=0.25\pi and various a');
legend('a=0.12', 'a=0.3', 'a=0.5');

clear;
t = 0:0.01:30;
a = 0.12;

Omega_1 = 0.1 * pi;
x_c = exp(-a * t) .* cos(Omega_1 * t);
figure;
```

```
plot(t, x_c);

Omega_1 = 0.25 * pi;
x_c = exp(-a * t) .* cos(Omega_1 * t);
hold on;
plot(t, x_c);

Omega_1 = 0.6 * pi;
x_c = exp(-a * t) .* cos(Omega_1 * t);
hold on;
plot(t, x_c);

xlabel('t (s)');
ylabel('x_c(t)');
title('x_c(t) in time domain for a=0.12 and various \Omega_1');
legend('\Omega_1=0.1\pi', '\Omega_1=0.25\pi', '\Omega_1=0.6\pi');

clear;
Omega = -3:0.01:3;
a = 0.12;

Omega_1 = 0.1 * pi;
X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);
figure;
plot(Omega, X_c);

Omega_1 = 0.25 * pi;
X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);
hold on;
plot(Omega, X_c);

Omega_1 = 0.6 * pi;
X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);
hold on;
plot(Omega, X_c);

xlabel('\Omega (rad/s)');
ylabel('X_c(\Omega)');
title('X_c(\Omega) in frequency domain for a=0.12 and various \Omega_1');
legend('\Omega_1=0.1\pi', '\Omega_1=0.25\pi', '\Omega_1=0.6\pi');
```

A c)

```
clear;
close all;

a = 0.12;
Omega_1 = 0.25 * pi;
T = 1 / 4.8;

Omega = -pi:0.005:pi;

X_c = (a + 1j * Omega) ./ ( (a + 1j * Omega).^2 + Omega_1^2 );
X_c = abs(X_c);

figure;
plot(Omega, X_c, 'LineWidth', 2);

num = [1 -exp(-a*T)*cos(Omega_1*T) 0];
```

```
den = [1 -2*exp(-a*T)*cos(Omega_1*T) exp(-2*a*T)];
h = freqz(num, den, Omega);

hold on;
plot(Omega/T, T*abs(h), 'x');

title('X_c(j\Omega) and X_c(e^{j\omega})');
xlabel('\Omega and \omega/T (rad/s)');
ylabel('X_c(j\Omega) and X_c(e^{j\omega})');
legend('X_c(j\Omega)', 'X_c(e^{j\omega})');
```

A d)

```
clear;
close all;
omega = -pi:0.01:pi;

z = exp(1j*omega);

index = 0;
for alpha = 0.8:-0.4:-0.8
    H_LP = (1 - alpha) / 2 * (1 + 1./z) ./ (1 - alpha./z);

    magnitude = abs(H_LP);
    phase = angle(H_LP);

    figure(1);
    plot(omega, magnitude);
    hold on;

    figure(2);
    plot(omega, rad2deg(phase));
    hold on;

    index = index + 1;
    legendInfo{index} = ['\alpha = ', sprintf('%0.1f', alpha)];
end

figure(1);
xlabel('\omega (rad/sample)');
ylabel('|H_{LP}(e^{j\omega})|');
legend(legendInfo);

figure(2);
xlabel('\omega (rad/sample)');
ylabel('\angle H_{LP}(e^{j\omega}) deg');
legend(legendInfo);
```

B.4 c) time domain

```
clear;
close all;

a = 0.12;
Omega_1 = 0.25 * pi;
Omega_2 = 1.9 * pi;
F_1 = 1.2;
F_2 = 4.8;

sample = 0:255;

T_1 = 1 / F_1;
T_2 = 1 / F_2;

t_1 = T_1 * sample;
x_1 = exp(-a * t_1) .* cos(Omega_1 * t_1) + 0.1 * sin(Omega_2 * t_1);

t_2 = T_2 * sample;
x_2 = exp(-a * t_2) .* cos(Omega_1 * t_2) + 0.1 * sin(Omega_2 * t_2);

plot(t_1, x_1);
hold on;
plot(t_2, x_2);
xlabel('Time (s)');
ylabel('x[n]');
title('x[n]');
legend('F_1 = 1.2 Hz', 'F_2 = 4.8 Hz');
axis([0 60 -0.8 1.2]);
set (gcf, 'Position', [500 500 1000 420]);
```

B.4 c) frequency domain

```
clear;
close all;

a = 0.12;
Omega_1 = 0.25 * pi;
Omega_2 = 1.9 * pi;
F_1 = 1.2;
F_2 = 4.8;

sample = 0:255;
N = length(sample);

T_1 = 1 / F_1;
T_2 = 1 / F_2;

t_1 = T_1 * sample;
x_1 = exp(-a * t_1) .* cos(Omega_1 * t_1) + 0.1 * sin(Omega_2 * t_1);

t_2 = T_2 * sample;
x_2 = exp(-a * t_2) .* cos(Omega_1 * t_2) + 0.1 * sin(Omega_2 * t_2);

X_1 = fft(x_1);
X_1 = fftshift(X_1);
X_1 = X_1(N/2+1:N);
X_1 = abs(X_1);
f = (N/2:N-1) * F_1 / N - F_1 / 2;
plot(f, X_1);
xlabel('Frequency (Hz)');
```

```
ylabel('FFT(x[n])');
title('FFT(x[n]) F_1 = 1.2Hz');

% peaks
pos = find(diff(sign(diff(X_1)))<0) + 1;
for index=pos
        text(f(index) + 0.02 * max(f), X_1(index), ['f = ' num2str(f(index)) ' Hz']);
end
hold on;
plot(f(pos), X_1(pos), 'x');

X_2 = fft(x_2);
X_2 = fftshift(X_2);
X_2 = X_2(N/2+1:N);
X_2 = abs(X_2);
f = (N/2:N-1) * F_2 / N - F_2 / 2;
figure;
plot(f, X_2);
xlabel('Frequency (Hz)');
ylabel('FFT(x[n])');
title('FFT(x[n]) F_2 = 4.8Hz');

% peaks
pos = find(diff(sign(diff(X_2)))<0) + 1;
for index=pos
        text(f(index) + 0.02 * max(f), X_2(index), ['f = ' num2str(f(index)) ' Hz']);
end
hold on;
plot(f(pos), X_2(pos), 'x');
```

B.4 d) ω_{max} obtained from DTFT

```
clear;
close all;

a = 0.12;
Omega_1 = 0.25 * pi;
T = 1 / 4.8;

Omega = -pi:0.005:pi;

num = [1 -exp(-a*T)*cos(Omega_1*T) 0];
den = [1 -2*exp(-a*T)*cos(Omega_1*T) exp(-2*a*T)];
h = freqz(num, den, Omega);

plot(Omega, abs(h));

title('X_1(e^{j\omega})');
xlabel('\omega (rad/sample)');
ylabel('X_1(e^{j\omega})');
```

B.4 d) ω_{max} obtained from FFT

```
clear;
close all;

N = 2048;
F_s = 4.8;

t = 0:1/F_s:1/F_s*(N-1);

F = 0.25 * pi / 2 / pi;
```

```
x = exp(-0.12 * t) .* cos(2 * pi * F * t);

X = fft(x);
X = fftshift(X);

X = X(N/2+1:N);
X = abs(X);
f = (N/2:N-1) * F_s / N - F_s / 2;
plot(f, X);
title('FFT(x_1[n])');
xlabel('Frequency (Hz)');
ylabel('FFT(x_1[n])');

% peaks
pos = find(diff(sign(diff(X)))<0) + 1;
for index=pos
        text(f(index) + 0.02 * max(f), X(index), ['f = ' num2str(f(index)) ' Hz']);
end
hold on;
plot(f(pos), X(pos), 'x');
```

B.4 e) & f) & g)

```
clear;
close all;

a = 0.12;
Omega_1 = 0.25 * pi;
Omega_2 = 1.9 * pi;
F_s = 4.8;
F_s_12Hz = 1.2;
alpha_1st = 0.597991;
alpha_2nd = 0.470126;

sample = 0:255;
N = length(sample);

t = 1 / F_s * sample;
x_2 = 0.1 * sin(Omega_2 * t);
x = exp(-a * t) .* cos(Omega_1 * t) + x_2;

t_12Hz = 1 / F_s_12Hz * sample;
x_12Hz = exp(-a * t_12Hz) .* cos(Omega_1 * t_12Hz) + 0.1 * sin(Omega_2 * t_12Hz);

y_1st = zeros(1, N);
y_2nd_intermediate = zeros(1, N);
y_2nd = zeros(1, N);
y_2_1st = zeros(1, N);
y_2_2nd_intermediate = zeros(1, N);
y_2_2nd = zeros(1, N);

y_1st(1) = (1 - alpha_1st) / 2 * x(1);
for index = 2:N
    y_1st(index) = (1 - alpha_1st) / 2 * (x(index) + x(index-1)) + alpha_1st * y_1st(index
        -1);
end

y_2nd_intermediate(1) = (1 - alpha_2nd) / 2 * x(1);
for index = 2:N
    y_2nd_intermediate(index) = (1 - alpha_2nd) / 2 * (x(index) + x(index-1)) + alpha_2nd *
        y_2nd_intermediate(index-1);
end
```

```
y_2nd(1) = (1 - alpha_2nd) / 2 * y_2nd_intermediate(1);
for index = 2:N
    y_2nd(index) = (1 - alpha_2nd) / 2 * (y_2nd_intermediate(index) + y_2nd_intermediate(
        index-1)) + alpha_2nd * y_2nd(index-1);
end


y_2_1st(1) = (1 - alpha_1st) / 2 * x_2(1);
for index = 2:N
    y_2_1st(index) = (1 - alpha_1st) / 2 * (x_2(index) + x_2(index-1)) + alpha_1st * y_2_1st
        (index-1);
end

y_2_2nd_intermediate(1) = (1 - alpha_2nd) / 2 * x_2(1);
for index = 2:N
    y_2_2nd_intermediate(index) = (1 - alpha_2nd) / 2 * (x_2(index) + x_2(index-1)) +
        alpha_2nd * y_2_2nd_intermediate(index-1);
end
y_2_2nd(1) = (1 - alpha_2nd) / 2 * y_2_2nd_intermediate(1);
for index = 2:N
    y_2_2nd(index) = (1 - alpha_2nd) / 2 * (y_2_2nd_intermediate(index) +
        y_2_2nd_intermediate(index-1)) + alpha_2nd * y_2_2nd(index-1);
end

y_ds = y_2nd(1:4:N);

X = fft(x);
X = fftshift(X);
X = X(N/2+1:N);
X = abs(X);

Y_1st = fft(y_1st);
Y_1st = fftshift(Y_1st);
Y_1st = Y_1st(N/2+1:N);
Y_1st = abs(Y_1st);

Y_2nd = fft(y_2nd);
Y_2nd = fftshift(Y_2nd);
Y_2nd = Y_2nd(N/2+1:N);
Y_2nd = abs(Y_2nd);

N_ds = N / 4;
Y_ds = fft(y_ds);
Y_ds = fftshift(Y_ds);
Y_ds = Y_ds(N_ds/2+1:N_ds);
Y_ds = abs(Y_ds);

X_12Hz = fft(x_12Hz);
X_12Hz = fftshift(X_12Hz);
X_12Hz = X_12Hz(N/2+1:N);
X_12Hz = abs(X_12Hz);


f = (N/2:N-1) * F_s / N - F_s / 2;

F_s_ds = F_s/4;
f_ds = (N_ds/2:N_ds-1) * F_s_ds / N_ds - F_s_ds / 2;

f_12Hz = (N/2:N-1) * F_s_12Hz / N - F_s_12Hz / 2;

% (e) time domain
plot(t, x);
```

```
hold on;
plot(t, y_1st);
hold on;
plot(t, y_2nd);
xlabel('Time (s)');
ylabel('Magnitude');
title('x[n] and filtered x[n]');
legend('x[n]', '1st order filtered x[n]', '2nd order filtered x[n]');
axis([0 35 -0.8 1.2]);
set (gcf, 'Position', [500 500 1000 420]);

% (e) frequency domain
figure;
plot(f, X);
hold on;
plot(f, Y_1st);
hold on;
plot(f, Y_2nd);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('FFT of x[n] and filtered x[n]');
legend('x[n]', '1st order filtered x[n]', '2nd order filtered x[n]');
set (gcf, 'Position', [500 500 1000 420]);

% (f) x2[n] - time domain
figure;
plot(t, x_2);
hold on;
plot(t, y_2_1st);
hold on;
plot(t, y_2_2nd);
xlabel('Time (s)');
ylabel('Magnitude');
title('x_2[n] and filtered x_2[n]');
legend('x_2[n]', '1st order filtered x_2[n]', '2nd order filtered x_2[n]');
axis([0 10 -0.1 0.1]);
set (gcf, 'Position', [500 500 1000 420]);

% (g) down sample and sampling frequency = 1.2Hz - time domain
figure;
plot(t(4:4:N), y_ds);
hold on;
plot(t_12Hz, x_12Hz);
xlabel('Time (s)');
ylabel('x[n]');
title('Down sample and F_s = 1.2Hz outputs');
legend('Down Sample', 'F_s = 1.2Hz');
axis([0 50 -0.8 1.2]);

% (g) down sample and sampling frequency = 1.2Hz - frequency domain
figure;
plot(f_ds, Y_ds);
hold on;
plot(f_12Hz, X_12Hz);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('FFT of down sample and F_s = 1.2Hz outputs');
legend('Down Sample', 'F_s = 1.2Hz');
```