E

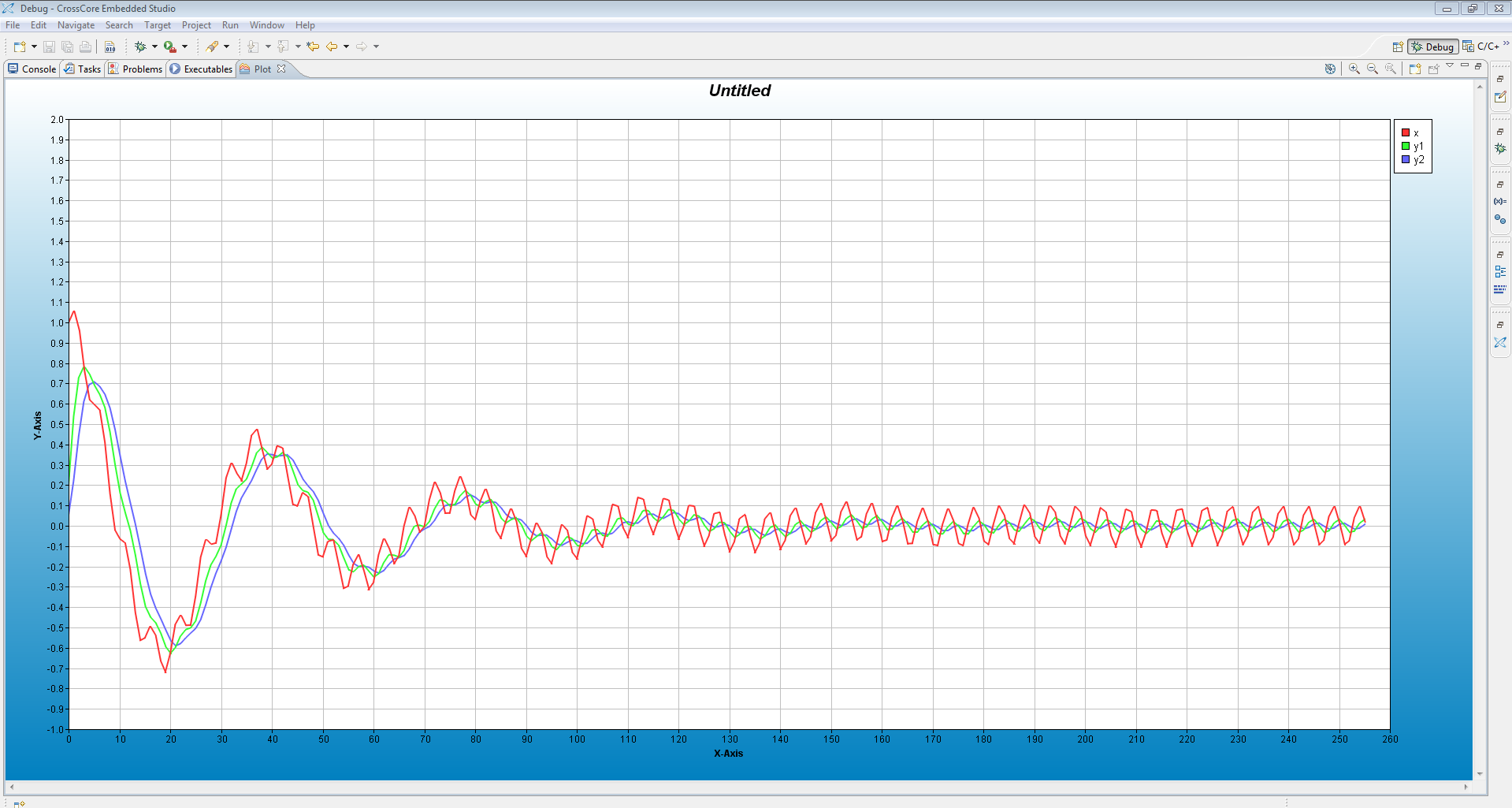
For the signal x[n], x[n]=x(nt)==

=

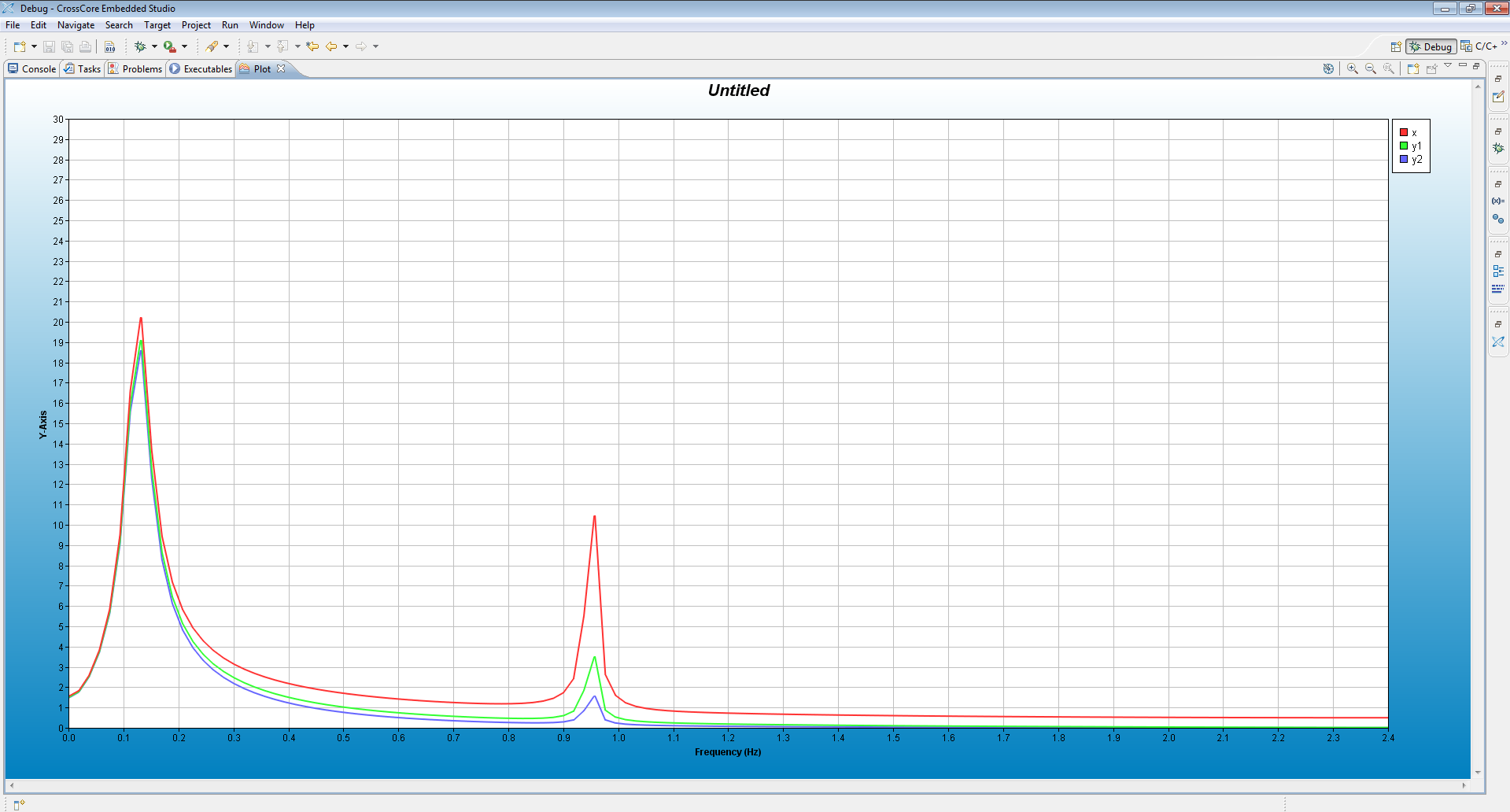
As we know:t=1/4.8

X[n]=

After the signal go through two kinds of low pass filters, we get:



Time domain



Frequency domain

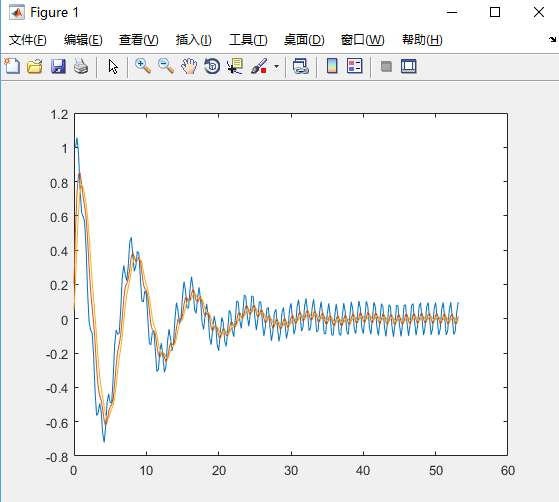
In the two images above, x (the red one) means x[n], the sequence without filters, y1 (the green one) means signal go through first order filter, and y2 (the blue one) means signal go through second order filter. From the diagram we can see at the frequency of 0.95Hz,

the gain=1.7÷10.5=0.162, which is lower than 0.25, thus it suits the filter we designed.

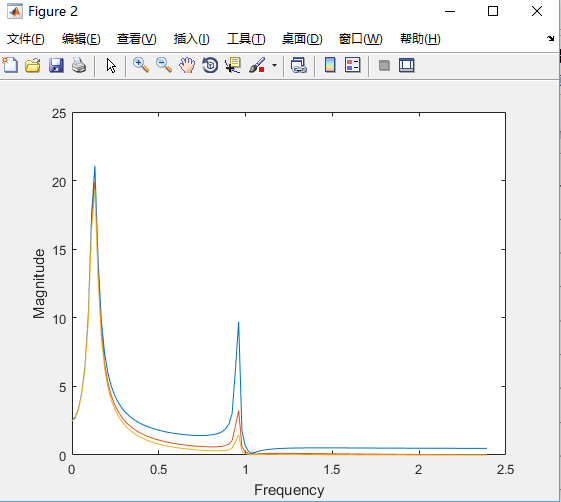
ii)

As we talked about in (i), the gain at the 0.95Hz has been removed. But the gain at 0.125Hz is nearly equal 1.

We use Matlab to stimulate the signal. The code and the figures are showed below.



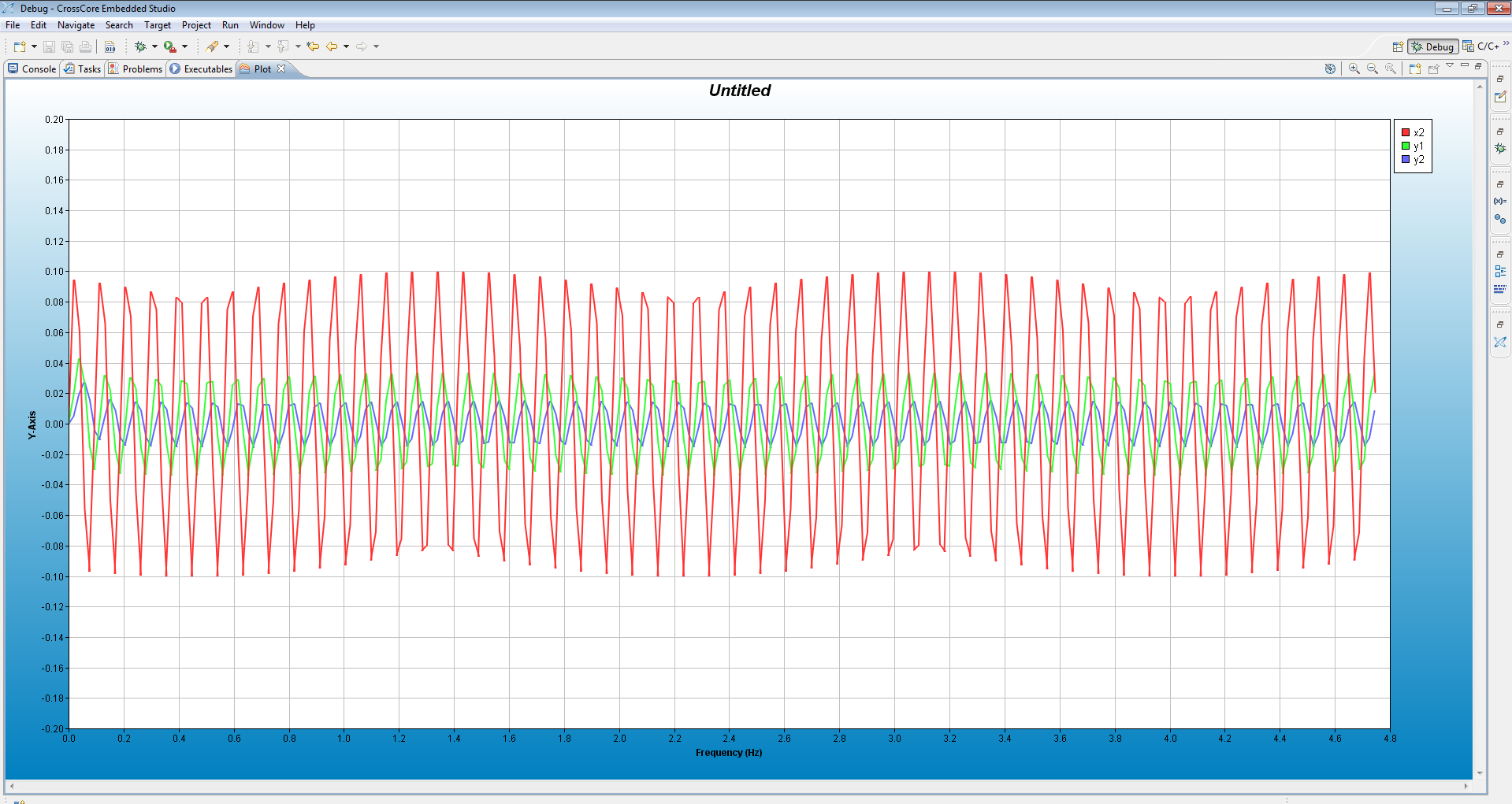
Time domain



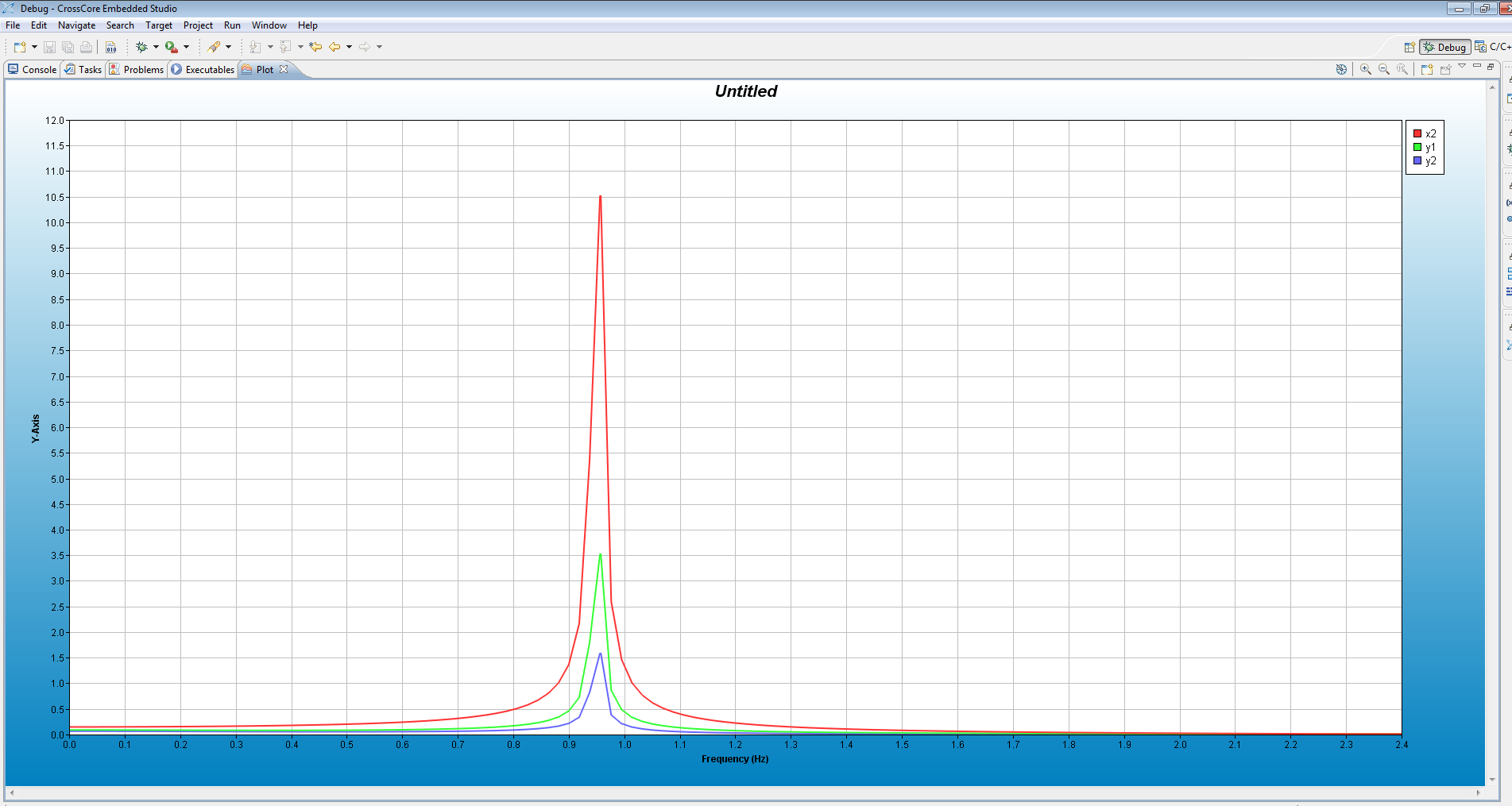
Frequency domain

F

After signal [n] go through two kinds of filters, we get:



Time domain



Frequency domain

In the two images above, x (the red one) means[n], the sequence without filters, y1 (the green one) means signal go through first order filter, and y2 (the blue one) means signal go through second order filter. From the diagram we can see at the frequency of 0.95Hz,

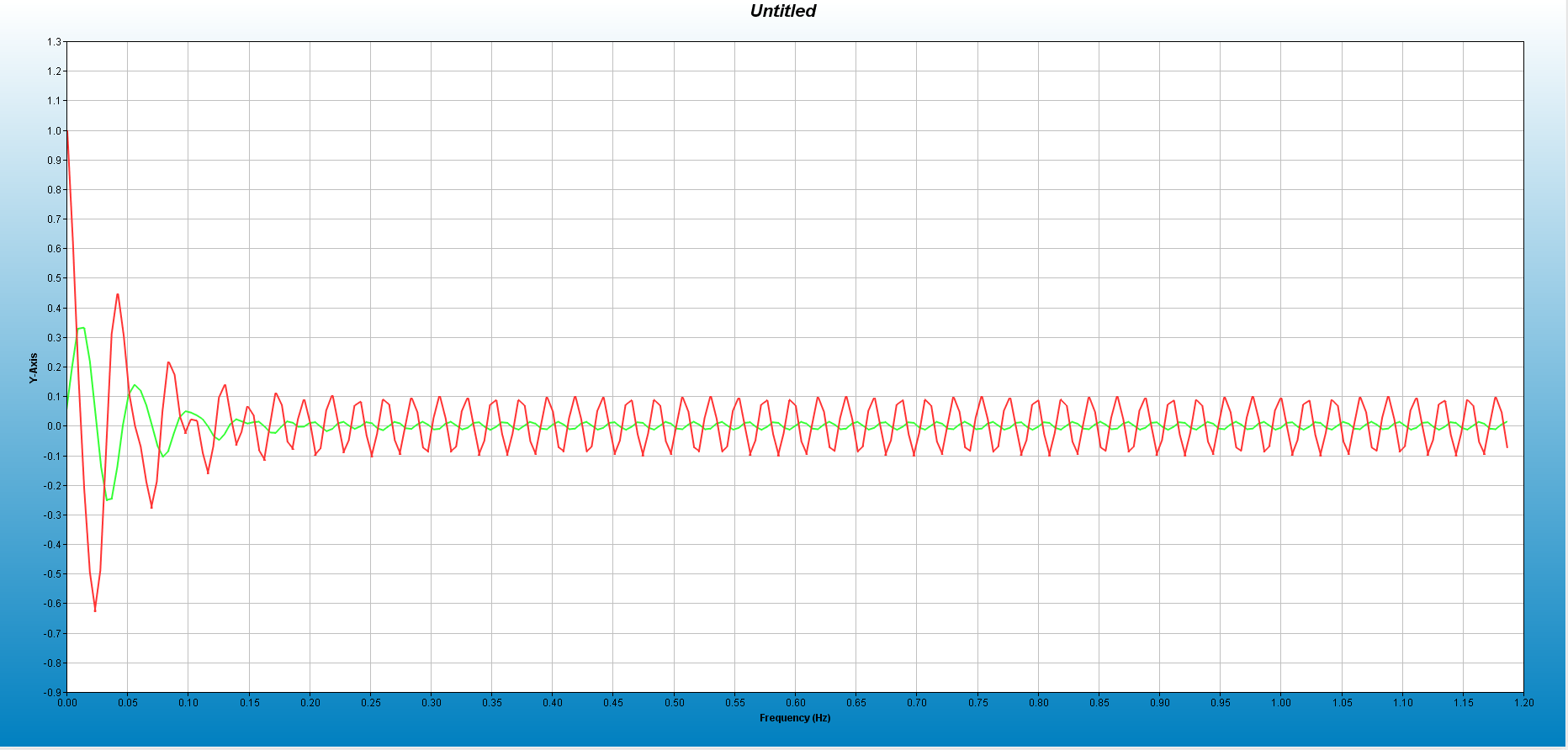
the gain=3.1÷21.0=0.148, which is lower than 0.25, thus it suits the filter we designed. After the signal go through the second order filter, it has 90 degrees delay.

The code is attached below.

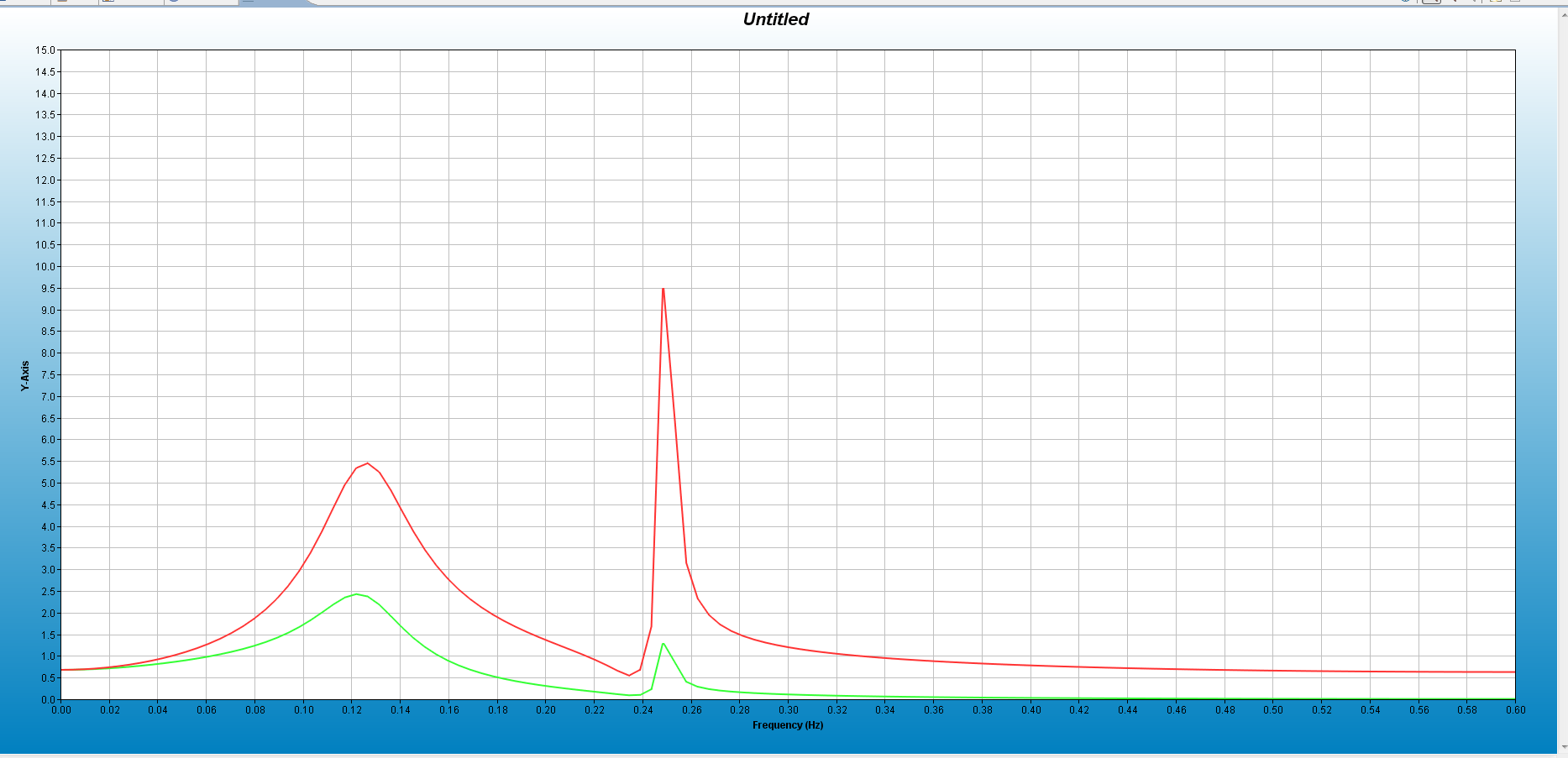
|  |
| --- |
|  |
|  | #include <math.h>  #include <stdio.h> |
|  |  |
|  | // Globals |
|  | #define N 256 |
|  | #define PI 3.1415 |
|  |  |
|  | float x[N]; |
|  | float x2[N]; |
|  | float y1[N]; |
|  | float y2[N]; |
|  |  |
|  | int main(void) |
|  | { |
|  |  |
|  | int i; |
|  | float omega1 = 0.25 \* PI, omega2 = 1.9 \* PI; |
|  | float T2 = 1/4.8; |
|  | float T = 1.0; |
|  | float a=0.12; |
|  | float alpha1 = 0.593, alpha2 = 0.464; |
|  |  |
|  |  |
|  | //x[0] = exp(-a\*0\*T2)\*cos(omega1\*0\*T2) + 0.1\*sin(omega2\*0\*T2); |
|  | x2[0] = 0.1\*sin(omega2\*0\*T2); |
|  | y1[0] = ((1-alpha1)/2)\*x2[0]; |
|  | y2[0] = ((1-alpha2)/2)\*y1[0]; |
|  | for (i = 0; i < N; i++) |
|  | { |
|  | //x[i] = exp(-a\*i\*T2)\*cos(omega1\*i\*T2) + 0.1\*sin(omega2\*i\*T2); |
|  | x2[i] = 0.1\*sin(omega2\*i\*T2); |
|  | y1[i] = ((1-alpha1)/2)\*x2[i]+((1-alpha1)/2)\*x2[i-1]+alpha1\*y1[i-1]; |
|  | y2[i] = ((1-alpha2)/2)\*y1[i]+((1-alpha2)/2)\*y1[i-1]+alpha2\*y2[i-1]; |
|  | } |
|  |  |
|  | /\*for (i = 0; i < N; i++) |
|  | { |
|  | printf("x[%d] = %f\n", i, x[i]); |
|  | } |
|  | \*/ |
|  | printf("Done.\n"); |
|  |  |
|  | return 0; |
|  | } |

G

After the signal go through the second order low pass filter, we get:



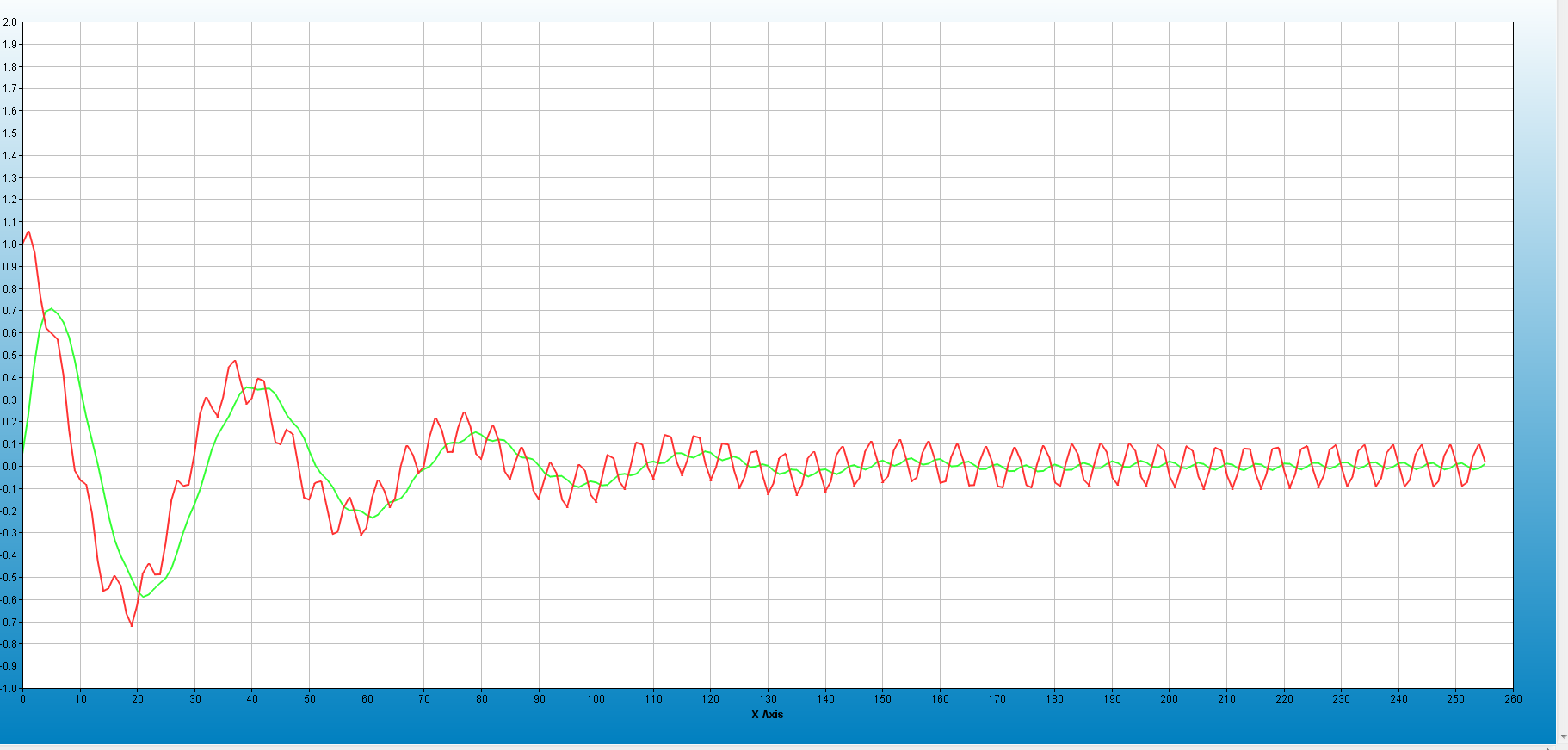
Time domain



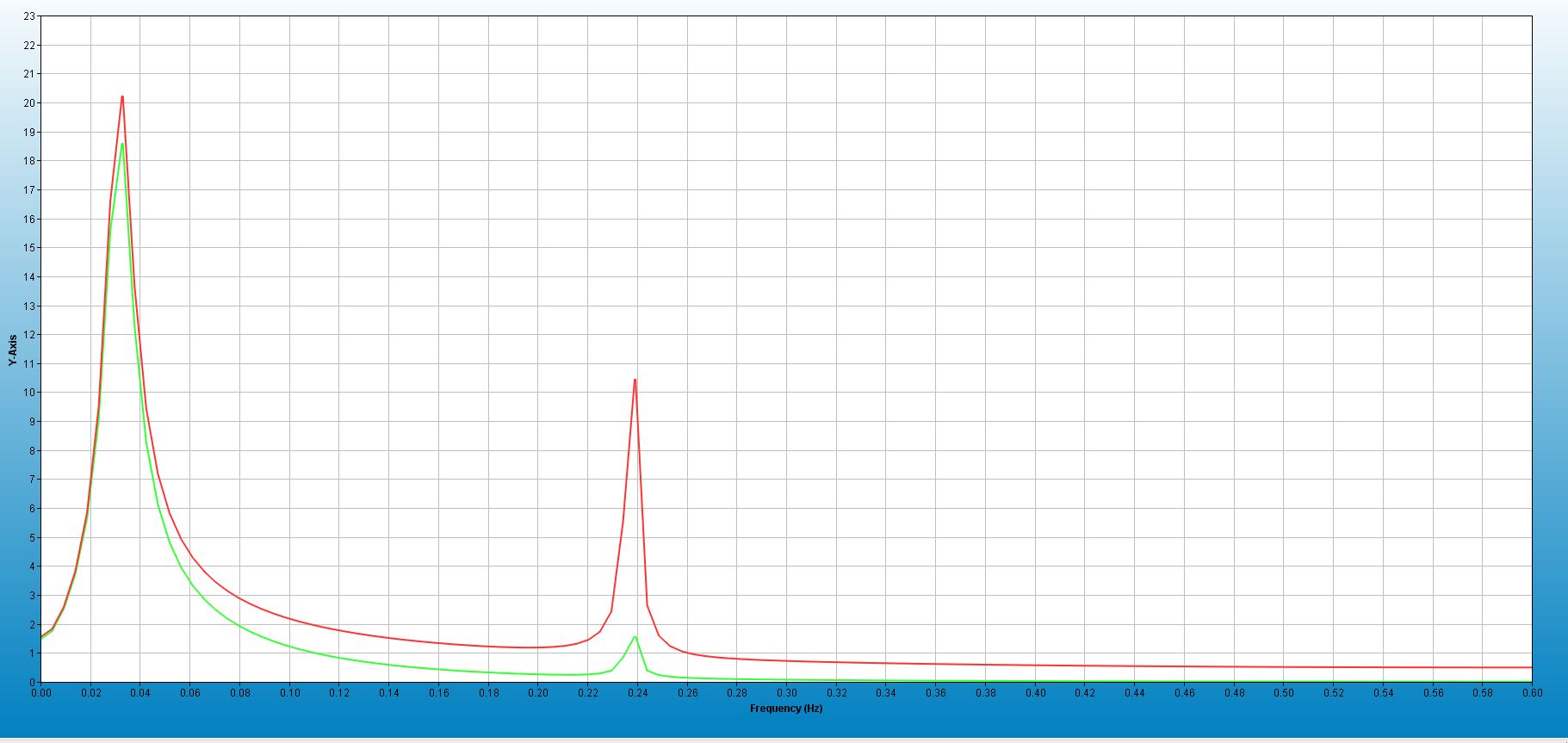
Frequency domain

In the two images above, x (the red one) means x[n], the sequence without filters, y1 (the green one) means signal go through the second order filter.

Then we plot x(t) at 1.2Hz,



Time domain



Frequency domain

After y[n] go through the filter, at about 0.25Hz, the gain =2.3÷19.0=0.121; Then at 0.13Hz,

The gain =5.0÷11.0=0.455. For x(t), when it go through the second order filter, at 0.24Hz,

The gain= 1.5÷10.5=0.143; at 0.03Hz, the gain=18.5÷20.3=0.911. From the analyze we see that the filter act well when signal is x(t), it is a low pass filter. For the signal is y[n], it is also act as a low pass filter, but it attenuates about a half at 0.13Hz.

The code of how to generate and plot y[n] is attached below.

|  |
| --- |
|  |
|  | #include <math.h>  #include <stdio.h> |
|  | #include <complex.h> |
|  |  |
|  | // Globals |
|  | #define N 256 |
|  | #define PI 3.1415 |
|  |  |
|  | float x[N]; |
|  | float x2[N]; |
|  | float y1[N]; |
|  | float y2[N]; |
|  | float n1[N]; |
|  |  |
|  | int main(void) |
|  | { |
|  |  |
|  | int i; |
|  |  |
|  | float omega1 = 0.25 \* PI, omega2 = 1.9 \* PI; |
|  | float T2 = 1/4.8; |
|  | float T = 1/1.2;; |
|  | float a=0.12; |
|  | float alpha1 = 0.593, alpha2 = 0.464; |
|  |  |
|  |  |
|  | x[0] = exp(-a\*0\*T2)\*cos(omega1\*0\*T2) + 0.1\*sin(omega2\*0\*T2); |
|  | //x2[0] = 0.1\*sin(omega2\*0\*T2); |
|  | y1[0] = ((1-alpha1)/2)\*x2[0]; |
|  | y2[0] = ((1-alpha2)/2)\*y1[0]; |
|  | for (i = 0; i < N; i++) |
|  | { n1[i]=4\*i; |
|  | x[i] = exp(-a\*i\*T2)\*cos(omega1\*i\*T2) + 0.1\*sin(omega2\*i\*T2); |
|  | //x2[i] = 0.1\*sin(omega2\*i\*T2); |
|  | y1[i] = ((1-alpha1)/2)\*x[i]+((1-alpha1)/2)\*x[i-1]+alpha1\*y1[i-1]; |
|  | y2[i] = ((1-alpha2)/2)\*y1[i]+((1-alpha2)/2)\*y1[i-1]+alpha2\*y2[i-1]; |
|  | } |
|  |  |
|  | /\*for (i = 0; i < N; i++) |
|  | { |
|  | printf("x[%d] = %f\n", i, x[i]); |
|  | } |
|  | \*/ |
|  | printf("Done.\n"); |
|  |  |
|  | return 0; |
|  | } |

Appendix:

Matlab code in question e.

Code:

clc

clear all

close all

T1 = 1/1.2;

T2 = 1/4.8;

omega1 = 0.25 \* pi;

omega2 = 1.9 \* pi;

T = 1.0;

i = 0;

a = 0.12;

alpha1 = 0.593;

alpha2 = 0.464;

x(1) = exp(-a\*0\*T2)\*cos(omega1\*0\*T2) + 0.1\*sin(omega2\*0\*T2);

y1(1) = ((1-alpha1)/2)\*x(1);

y2(1) = ((1-alpha2)/2)\*y1(1);

for n = 2:1:256

%only use fs=4.8Hz

x(n) = exp(-a\*i\*T2)\*cos(omega1\*i\*T2) + 0.1\*sin(omega2\*i\*T2);

y1(n) = ((1-alpha1)/2)\*x(n)+((1-alpha1)/2)\*x(n-1)+alpha1\*y1(n-1);

y2(n) = ((1-alpha2)/2)\*y1(n)+((1-alpha2)/2)\*y1(n-1)+alpha2\*y2(n-1);

i = i+1;

end

sample = 0:1:255;

plot(T2\*sample, x);

hold on

plot(T2\*sample, y1);

plot(T2\*sample, y2);

figure;

fs1 = 1.2;

fs2 = 4.8;

N=255;

X\_mags = abs(fft(x))

Y1\_mags = abs(fft(y1));

Y2\_mags = abs(fft(y2));

fax\_bins = [0 : N-1]; %frequency axis in bins

N\_2 = ceil(N/2);

plot(fax\_bins(1:N\_2)\*fs2/N, X\_mags(1:N\_2))

hold on;

plot(fax\_bins(1:N\_2)\*fs2/N, Y1\_mags(1:N\_2))

plot(fax\_bins(1:N\_2)\*fs2/N, Y2\_mags(1:N\_2))

xlabel('Frequency')

ylabel('Magnitude');