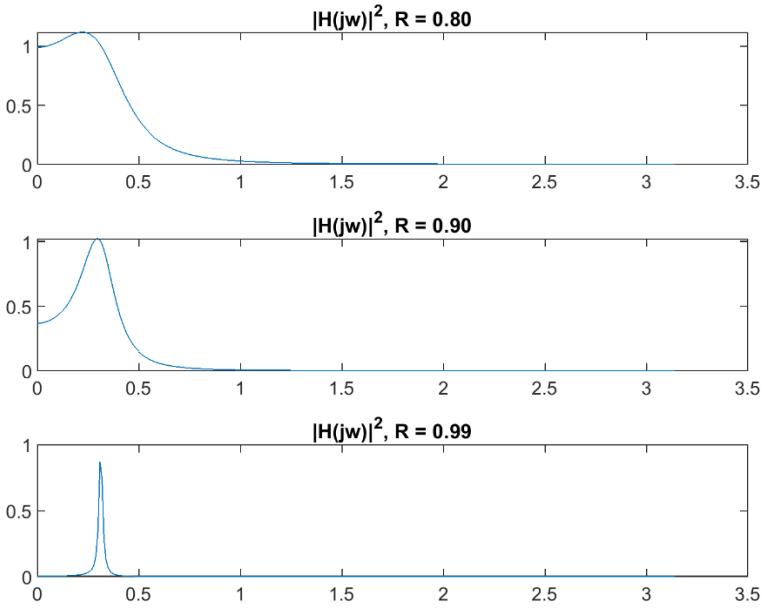
" بسمه تعالی "

# گزارش سوم آزمایشگاه DSP – زهرا لطیفی – ۹۹۲۳۰۶۹

### بخش ۳-۱-الف)

```
w1 = 0:0.01:pi;
R = [0.8, 0.9, 0.99];
f0 = 500;
fs = 10000;
w0 = (2*pi*f0)/fs;
for i = 1:1:3
    G = (1-R(i)).*(1-2*R(i)*cos(2*w0) + R(i)^2)^0.5;
    H = (G^2) ./ ((1-2*R(i)*cos(w1-w0) + R(i)^2).*(1-2*R(i)*cos(w1+w0) +
R(i)^2));
    subplot(3, 1, i);
    plot(w1, H);
    title(sprintf("|H(jw)|^2, R = %.2f", R(i)));
end
```

در این بخش طبق ضوابط داده شده در دستورکار، |H(jw)|^2 را برای R های برابر ۰.۸، ۰.۹، ۰.۹۹ در بازه فرکانسی ۰ تا پی رسم کردیم.

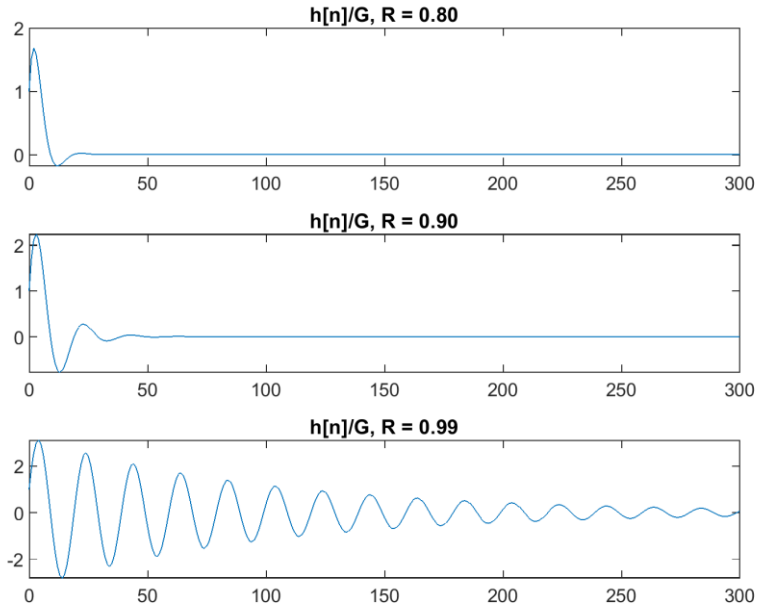



شکل ۱

**بخش ۳-۱-ب)**

```
n = 0:1:300;
for i = 1:1:3
    hn = (G/sin(w0)).*(R(i).^n).*sin(w0*n+w0);
    subplot(3, 1, i);
    plot(n, hn/G);
    title(sprintf("h[n]/G, R = %.2f", R(i)));
end
```

در این بخش h[n]/G را به ازای Rهای مذکور در بازه n بین ۰ و ۳۰۰ رسم کردیم.



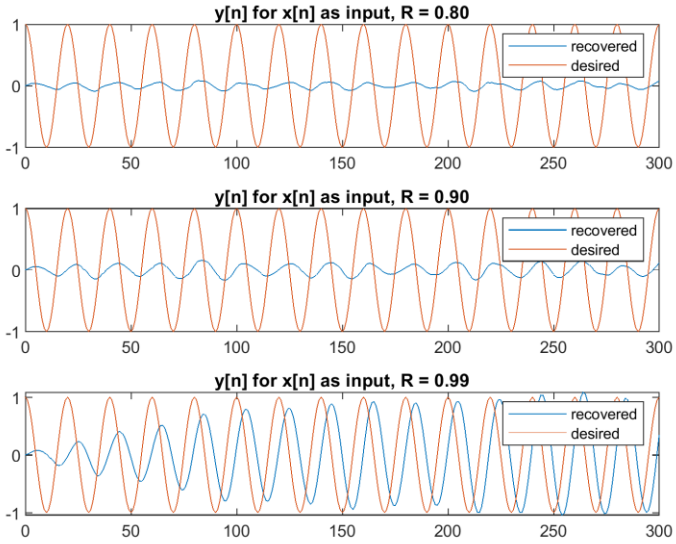

شکل ۲

**بخش ۳-۱-ج)**

```
n = 0:1:300;
v = randn(1, 301);
s = cos(w0*n);
x = s + v;


for i = 1:1:3
    a1 = -2*R(i)*cos(w0);
    a2 = R(i).^2;
    w1 = 0;
    w2 = 0;
    y = zeros(1, 301);
    for j = 1:1:301
        y(j) = -a1*w1 - a2*w2 +G*x(j);
        w2 = w1;
        w1 = y(j);
```

```
    end
    subplot(3, 1, i);
    plot(n, y);
    hold on;
    plot(n, s);
    title(sprintf("y[n] for x[n] as input, R = %.2f", R(i)));
    legend("recovered", "desired");
end
```

در این بخش نویز رندومی ایجاد کرده با سیگنال کسینوسی جمع کردیم و سپس مراحل خواسته شده را برروی آن اعمال کردیم و خروجی را با سیگنال بدون نویز به ازای هر R مقایسه کردیم. هرچه فیلتر sharpتر باشد، بیشتر طول می‌کشد تا به حالت مانا برسیم. در R=0.99 بیش از همه طول کشیده تا به حالت مانا برسیم اما سیگنال بسیار دقیق‌تر recover شده است.



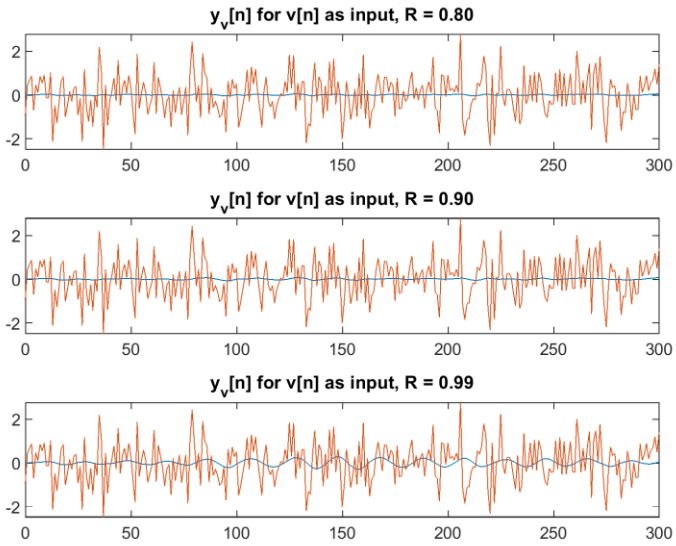

*شکل ۳*

**بخش ۳-۱-د)**

```
figure();
NRR = zeros(1, 3);
for i = 1:1:3
    a1 = -2*R(i)*cos(w0);
    a2 = R(i).^2;
    w1 = 0;
    w2 = 0;
    yv = zeros(1, 301);
    for j = 1:1:301
        yv(j) = -a1*w1 - a2*w2 +G*v(j);
        w2 = w1;
        w1 = yv(j);
    end
    subplot(3, 1, i);
    plot(n, yv);
    hold on;
    plot(n, v);
```

```
    title(sprintf("y_v[n] for v[n] as input, R = %.2f", R(i)));
    NRR(i) = (std(yv)/std(v))^2;
end
```

در این بخش سیگنال نویز را به طور جداگانه از فیلتر عبور داده و خروجی را با سیگنال نویز اصلی مقایسه کردیم.
نویز فیلتر شده بیشتر از نویز اصلی سینوسی است، زیرا فیلتر اعمال شده روی نویز دارای پاسخ فرکانسی است که بر فرکانس‌های خاص نسبت به سایر فرکانس‌ها تأکید دارد. فیلتر به گونه‌ای طراحی شده است که یک بریدگی در فرکانس «w0» داشته باشد، به این معنی که فرکانس‌های اطراف «w0» را کاهش می‌دهد و به فرکانس‌های دور از «w0» اجازه می‌دهد با تضعیف کمتری عبور کنند. نویز اصلی «v» دارای طیف وسیعی از فرکانس‌ها، از جمله فرکانس‌های اطراف «w0» است. هنگامی که این نویز از فیلتر عبور می‌کند، اجزای نویز در فرکانس‌های نزدیک به "w0" به طور قابل توجهی کاهش می‌یابد، در حالی که اجزای موجود در فرکانس‌های دیگر کمتر تحت تاثیر قرار می‌گیرند. این منجر به یک سیگنال نویز می‌شود که دارای اجزای فرکانس بالای کمتری است و بیشتر سینوسی به نظر می‌رسد، زیرا سیگنال‌های سینوسی با یک جزء فرکانس مشخص می‌شوند.
«NRR» (نسبت کاهش نویز) محاسبه‌شده در کد اندازه‌گیری می‌کند که چقدر انحراف استاندارد نویز فیلترشده «yv» در مقایسه با نویز اصلی «v» کاهش یافته است. "NRR" بالاتر نشان می دهد که فیلتر به طور موثر قدرت نویز را در فرکانس‌های اطراف"w0" کاهش داده است، که منجر به سیگنال تمیزتر می‌شود که بیشتر از سیگنال نویزدار اصلی شبیه یک سینوسی است. این مسئله به ویژه در کاربردهایی مفید است که حفظ یا تقویت برخی از اجزای فرکانس و کاهش سایر اجزای فرکانس مهم است.




*شکل ۴*

**بخش ۳-۱-ه)**

```
for i = 1:1:3
    fprintf('NRR(%.2f) = %.5f\n', R(i), NRR(i));
end
```

با استفاده از دستور std متلب، NRR را به ازای هر R محاسبه کردیم. نتایج به شرح زیر بود:

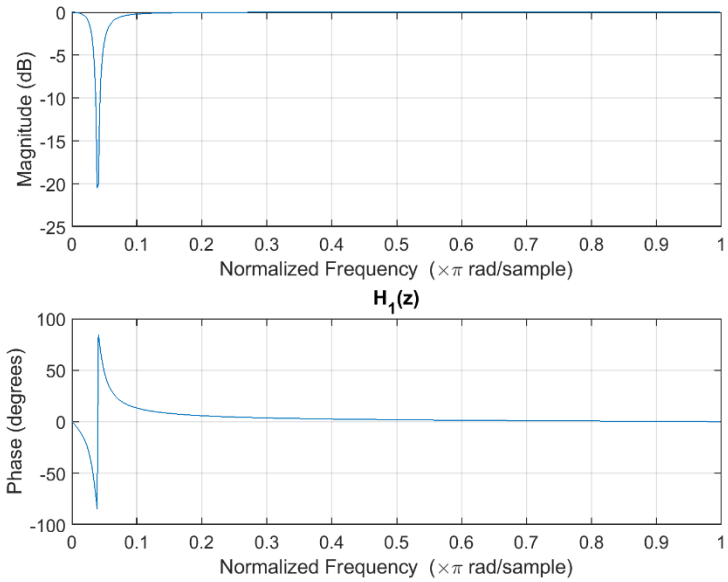
```
NRR(0.80) = 0.00044
NRR(0.90) = 0.00105
NRR(0.99) = 0.01822
```

که منطبق بر مقادیر تئوری هم بود.

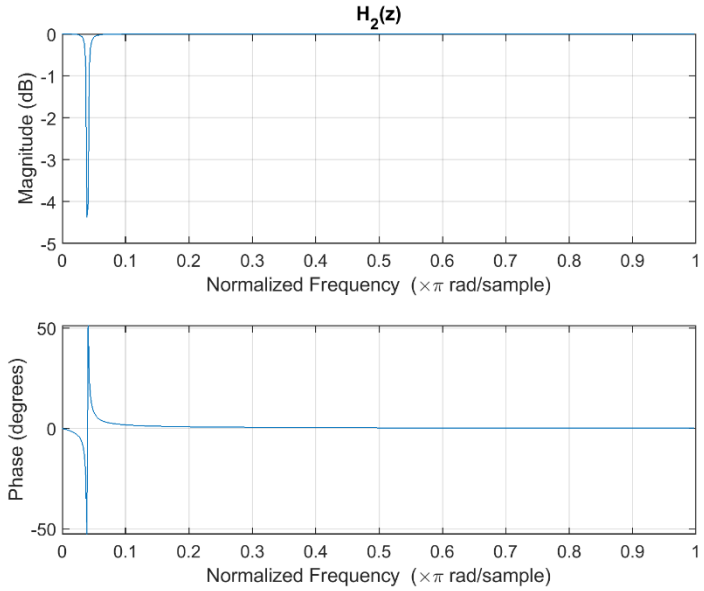
**بخش ۳-۲-الف)**

```
b1 = [0.969531, -1.923772, 0.969531];
a1 = [1, -1.923772, 0.939063];
freqz(b1, a1);
title("H_1(z)");

b2 = [0.996088, -1.976468, 0.996088];
a2 = [1, -1.976468, 0.992177];
freqz(b2, a2);
title("H_2(z)");
```

در این بخش تحقق کانونی دو فیلتر تعریف شده را با دستور freqz رسم کردیم:




شکل ۵




شکل ۶

**بخش ۳-۲-ب)**

```
fs = 400;
Ts = 1/fs;
H1 = tf(b1, a1, Ts);
stepinfo(H1, 'SettlingTimeThreshold', 0.01)

H2 = tf(b2, a2, Ts);
stepinfo(H2, 'SettlingTimeThreshold', 0.01)
```

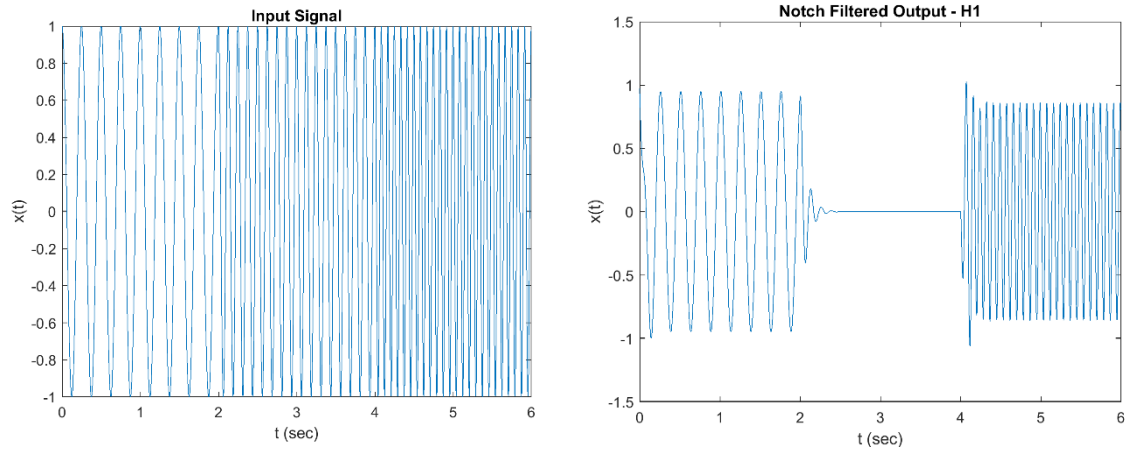
زمان نشست ۱ درصد هر دو فیتر را محاسبه کردیم:

```
Ans1 = struct with fields:
         RiseTime: 0.0025
     SettlingTime: 0.3700
      SettlingMin: 0.9297
      SettlingMax: 1.1581
        Overshoot: 15.8145
       Undershoot: 0
             Peak: 1.1581
         PeakTime: 0.0900

Ans2 = struct with fields:
         RiseTime: 0
     SettlingTime: 2.9150
      SettlingMin: 0.9510
      SettlingMax: 1.0539
        Overshoot: 5.4004
       Undershoot: 0
             Peak: 1.0539
         PeakTime: 0.0925
```

**بخش ۳-۲-ج و د)**

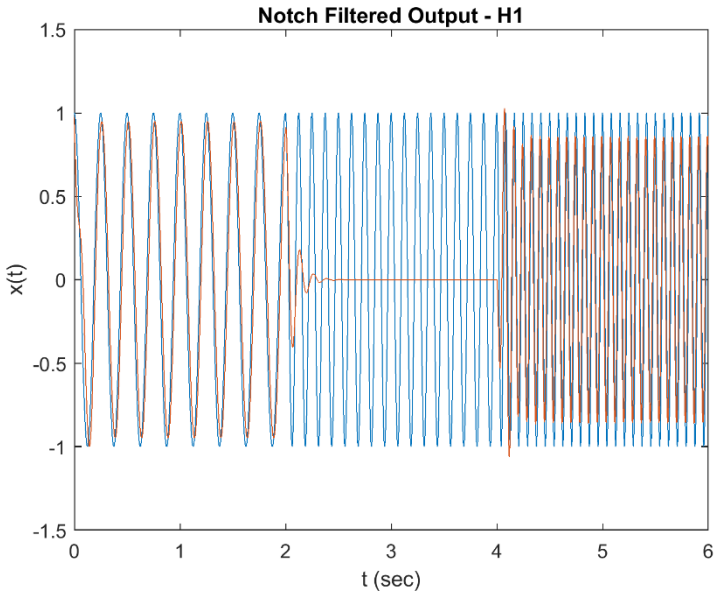
```
f1 = 4;
f2 = 8;
f3 = 12;
t1 = 0:Ts:2-Ts;
t2 = 2:Ts:4-Ts;
t3 = 4:Ts:6-Ts;
t = [t1, t2, t3];
x1 = cos(2*pi*f1*t1);
x2 = cos(2*pi*f2*t2);
x3 = cos(2*pi*f3*t3);
x = [x1, x2, x3];
figure();
plot(t, x);
title("Input Signal");
xlabel('t (sec)');
ylabel('x(t)');

figure();
plot(t, filter(b1, a1, x));
title("Notch Filtered Output - H1");
xlabel('t (sec)');
ylabel('x(t)');
```

سیگنال تعریف شده را در بازه‌های زمانی داده شده رسم کردیم و سیگنال نمونه برداری شده بین ۰ تا ۶ ثانیه را از Notch فیلتر ۱ عبور دادیم و خروجی و ورودی را مجزا رسم کردیم:




*شکل ۷*

شاهدیم که فرکانس میانی (f2) تقریبا به طور کامل حذف شده و f3 و f1 هم دیگر دامنه واحد ندارند و تضعیف شده دامنه‌هایشان. ثابت زمانی پاسخ گذرا منطبق بر بخش ب هست و همینطور به اندازه طول فیلتر (۳ واحد) شیفت فاز (تاخیر) داریم که در تصویر زیر قابل مشاهده است:

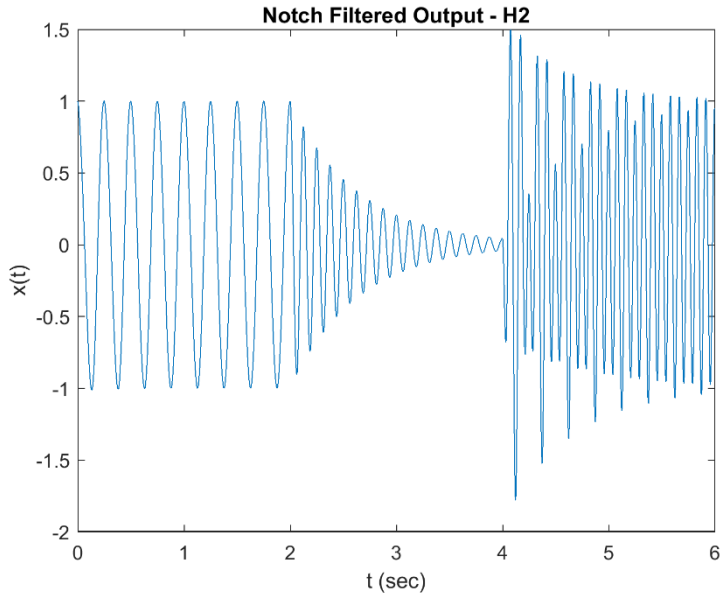



*شکل ۸*

**بخش ۳-۲-ه)**

```
figure();
plot(t, filter(b2, a2, x));
title("Notch Filtered Output - H2");
xlabel('t (sec)');
ylabel('x(t)');
```

همان مراحل را با Notch فیلتر دوم تکرار می‌کنیم. این بار خروجی به شکل زیر است:




شکل ۹

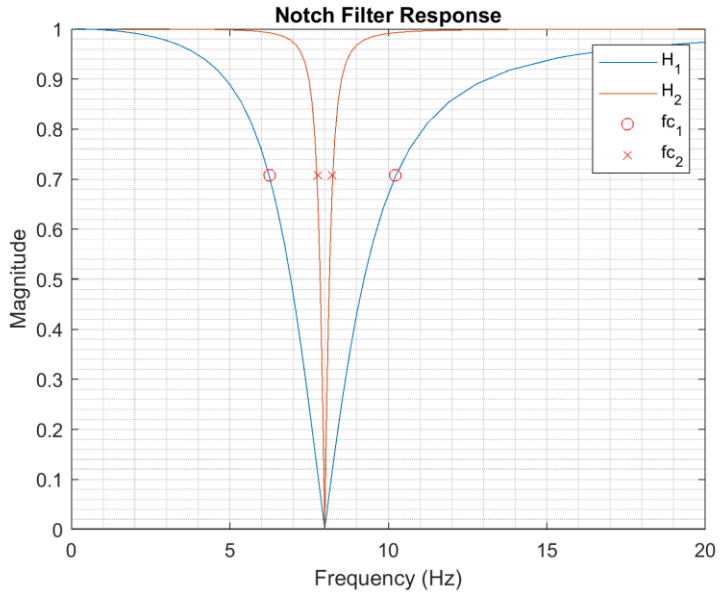
شاهدیم که تغییرات بر روی f2 و f3 ایجاد شده اند و فیلتر اثری بر f1 نگذاشته است.

**بخش ۳-۲-و و ز)**

```
[mag1, phase1, w1] = bode(H1, {0, 20*2*pi});
mag1 = squeeze(mag1);
figure();
plot(w1/(2*pi), mag1);
title('Notch Filter Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
[mag2, phase2, w2] = bode(H2, {0, 20*2*pi});
mag2 = squeeze(mag2);
hold on;


plot(w2/(2*pi), mag2);
grid on;
grid minor;
hold on;


ind = find(abs(mag1) < sqrt(1/2), 1, 'first');
fL1 = w1(ind)/(2*pi);
ind = find(abs(mag1) < sqrt(1/2), 1, 'last');
fH1 = w1(ind)/(2*pi);
plot([fL1 fH1], sqrt(1/2).*ones(1,2), 'ro');
hold on;


ind = find(abs(mag2) < sqrt(1/2), 1, 'first');
fL2 = w2(ind)/(2*pi);
ind = find(abs(mag2) < sqrt(1/2), 1, 'last');
fH2 = w2(ind)/(2*pi);
```

```
plot([fL2 fH2], sqrt(1/2).*ones(1,2), 'rx');
legend("H_1", "H_2", "fc_1", "fc_2");
```

در این قسمت پاسخ فرکانسی دو فیلتر Notch را رسم کرده و fL و fH هرکدام که فرکانس‌های قطع بالا و پایین هستند را یافته و برروی شکل مشخص کردیم. همینطور پهنای باند هر دو را محاسبه‌کردیم که مطابق انتظار بود:



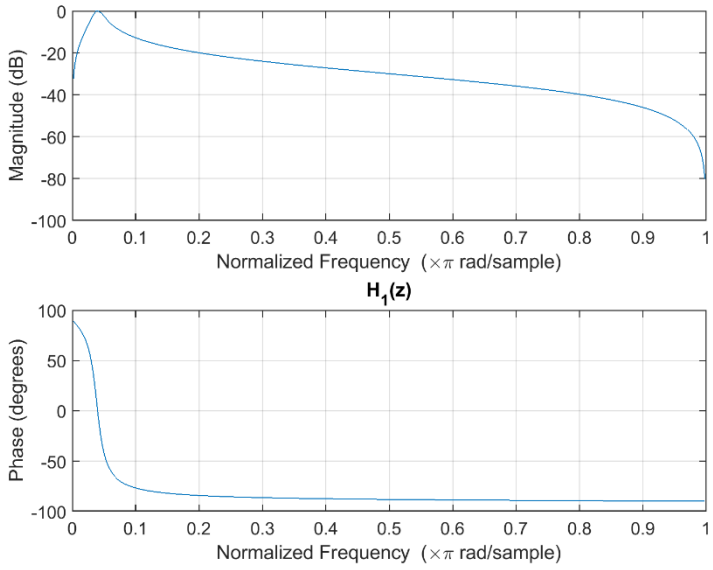

شکل ۱۰

```
bw1 = fH1 - fL1
```
```
bw1 = 3.9579
```
```
bw2 = fH2 - fL2
```
```
bw2 = 0.4606
```

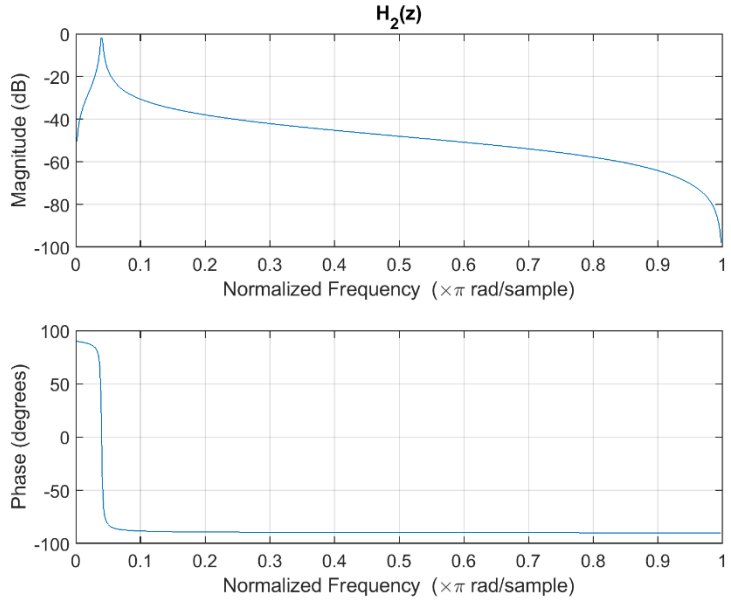
**بخش ۳-۲-ح)**

```
b1 = [0.030469, 0, -0.030469];
a1 = [1, -1.923772, 0.939063];
freqz(b1, a1);
title("H_1(z)");
```
```
b2 = [0.003912, 0, -0.003912];
a2 = [1, -1.976468, 0.992177];
freqz(b2, a2);
title("H_2(z)");
```

ابتدا پاسخ دامنه و فاز دو peak فیلتر تعریف شده را رسم کردیم.




شکل ۱۱




شکل ۱۲

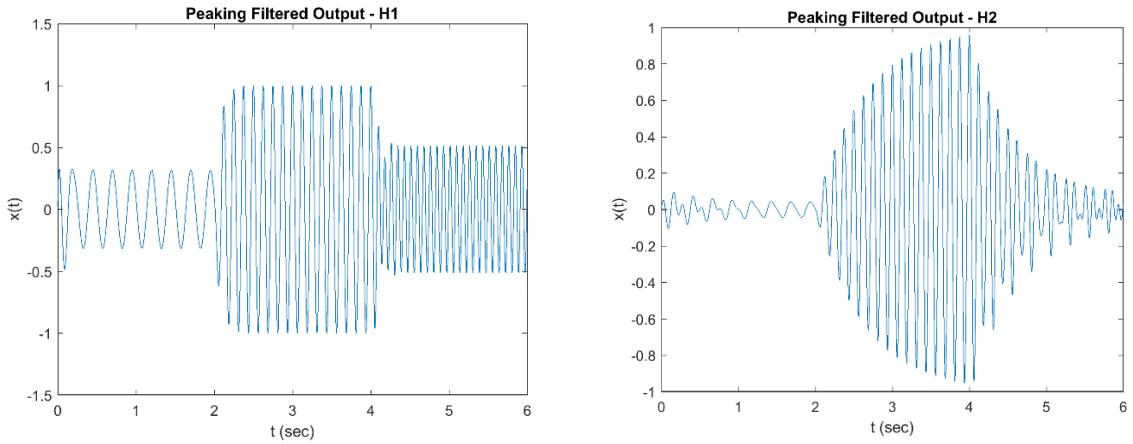
```
fs = 400;
Ts = 1/fs;
H1 = tf(b1, a1, Ts);
stepinfo(H1, 'SettlingTimeThreshold', 0.01)

H2 = tf(b2, a2, Ts);
stepinfo(H2, 'SettlingTimeThreshold', 0.01)
```

سپس زمان نشست ۱ درصد را با دستور step info گزارش کردیم:

```
Ans1 = struct with fields:
        RiseTime: 0.0025
    SettlingTime: 0.3700
     SettlingMin: -0.1581
     SettlingMax: 0.0703
       Overshoot: Inf
      Undershoot: Inf
            Peak: 0.3558
        PeakTime: 0.0250


Ans2 = struct with fields:
        RiseTime: 0
    SettlingTime: 2.9150
     SettlingMin: -0.0540
     SettlingMax: 0.0490
       Overshoot: Inf
      Undershoot: Inf
            Peak: 0.0596
        PeakTime: 0.0300
```

```
figure();
plot(t, filter(b1, a1, x));
title("Peaking Filtered Output - H1");
xlabel('t (sec)');
ylabel('x(t)');

figure();
plot(t, filter(b2, a2, x));
title("Peaking Filtered Output - H2");
xlabel('t (sec)');
ylabel('x(t)');
```

سیگنال خروجی حاصل از اعمال این دو peak فیلتر بر همان سیگنال x قبلی را رسم کردیم:

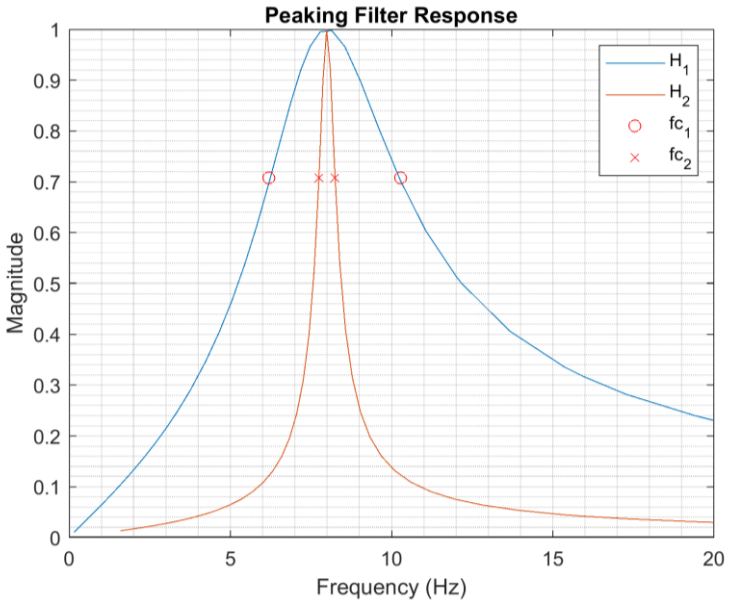



شکل ۱۳

دقیقا شاهدیم که نقش مکمل برای ۲ فیلتر Notch را دارند. H1 فرکانس‌های f1 و f3 را تضعیف کرده و H2 هم تقریبا تنها بر f2 اثر گذاشته.

نهایتا پاسخ فرکانسی و پهنای باند را هم بدست آوردیم:

```
[mag1, phase1, w1] = bode(H1, {0, 2*pi*20});
mag1 = squeeze(mag1);
figure();
plot(w1/(2*pi), mag1);
title('Peaking Filter Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
[mag2, phase2, w2] = bode(H2, {0, 2*pi*20});
mag2 = squeeze(mag2);
hold on;


plot(w2/(2*pi), mag2);
grid on;
grid minor;
hold on;


ind = find(abs(mag1) >= sqrt(1/2), 1, 'first');
fL1 = w1(ind-1)/(2*pi);
ind = find(abs(mag1) >= sqrt(1/2), 1, 'last');
fH1 = w1(ind+1)/(2*pi);
plot([fL1 fH1], sqrt(1/2).*ones(1,2), 'ro');
hold on;


ind = find(abs(mag2) >= sqrt(1/2), 1, 'first');
fL2 = w2(ind-1)/(2*pi);
ind = find(abs(mag2) >= sqrt(1/2), 1, 'last');
fH2 = w2(ind)/(2*pi);
plot([fL2 fH2], sqrt(1/2).*ones(1,2), 'rx');
legend("H_1", "H_2", "fc_1", "fc_2");
```




شکل ۱۴

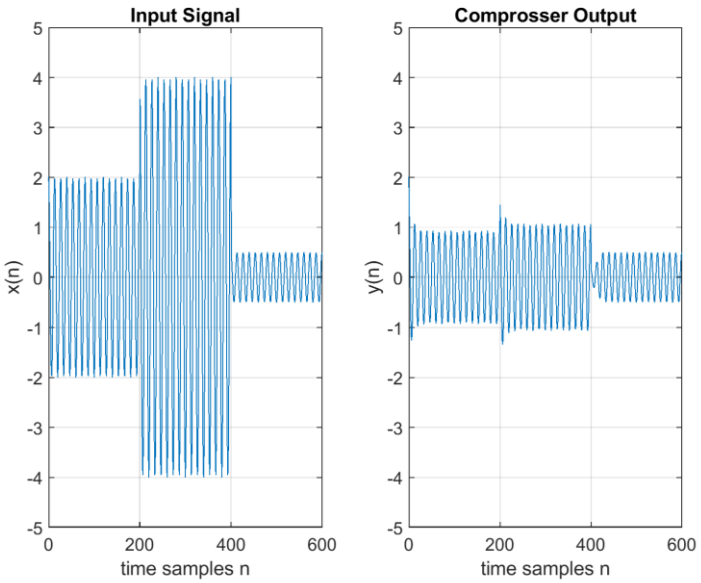
```
bw1 = fH1 - fL1
```
```
bw1 = 4.0857
```
```
bw2 = fH2 - fL2
```
```
bw2 = 0.4887
```

**بخش ۳-۳-الف و ب)**

```
f = 0.15/2;
A1 = 2;
A2 = 4;
A3 = 0.5;
t1 = 0:1:200;
t2 = 201:1:400;
t3 = 401:1:600;
t = [t1, t2, t3];
x1 = A1*cos(2*pi*f*t1);
x2 = A2*cos(2*pi*f*t2);
x3 = A3*cos(2*pi*f*t3);
x = [x1, x2, x3];


lambda = 0.9;
rho = 0.2;
c = zeros(1, 601);
c(1) = 0.5;
for n = 2:600
    c(n) = lambda*c(n-1) + (1-lambda)*abs(x(n));
end


G = zeros(1, 601);
for n = 1:601
    if c(n) >= c(1)
       G(n) = (c(n)/c(1)).^(rho-1);
    else
       G(n) = 1;
    end
    y(n) = G(n)*x(n);
end


figure();
subplot(1, 2, 1);
plot(t, x);
ylim([-5 5])
title("Input Signal");
xlabel('time samples n');
ylabel('x(n)');
grid on;


subplot(1,2,2);
plot(t, y);
```

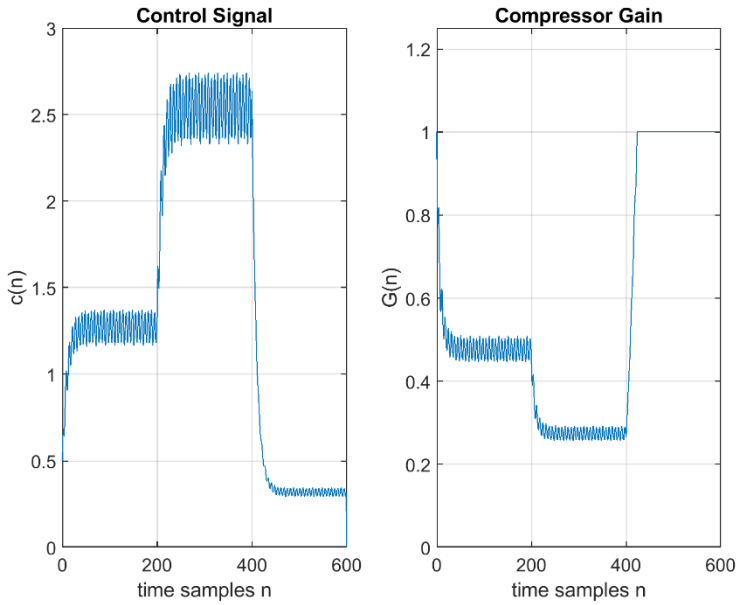
```
ylim([-5 5])
title("Comprosser Output");
xlabel('time samples n');
ylabel('y(n)');
grid on;
```

سینوسی ورودی را طبق خواست صورت سوال ساخته و از یک فشرده‌ساز با پارامترهای $\lambda = 0.9$، $c0 = 0.5$ و $\rho = 0.2$ استفاده کرده و سیگنال کنترلی cn و بهره Gn و خروجی $yn$ را نشان دادیم.




شکل ۱۵

```
figure();
subplot(1,2,1);
plot(t, c);
ylim([0 3])
title("Control Signal");
xlabel('time samples n');
ylabel('c(n)');
grid on;


subplot(1,2,2);
plot(t, G);
ylim([0 1.25])
title("Compressor Gain");
xlabel('time samples n');
ylabel('G(n)');
grid on;
```

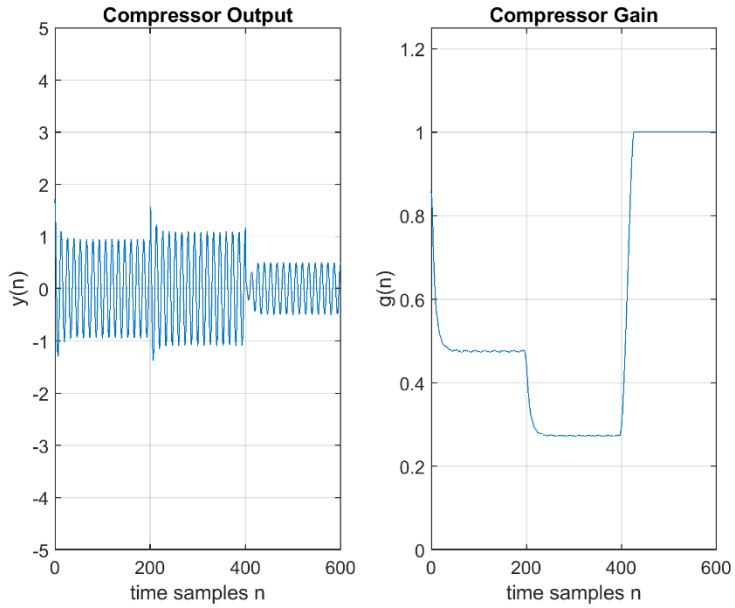



شکل ۱۶

این بار از هموارسازی ۷ نقطه‌ای برای هموارکردن بهره غیرخطی استفاده کردیم که یک فیلتر پایین گذر یا همان moving average بود.

```
g = movmean(G, 7);
y = g.*x;


figure();
subplot(1,2,1);
plot(t, y);
ylim([-5 5])
title("Compressor Output");
xlabel('time samples n');
ylabel('y(n)');
grid on;


subplot(1,2,2);
plot(t, g);
ylim([0 1.25])
title("Compressor Gain");
xlabel('time samples n');
ylabel('g(n)');
grid on;
```

خروجی و بهره را رسم کردیم:

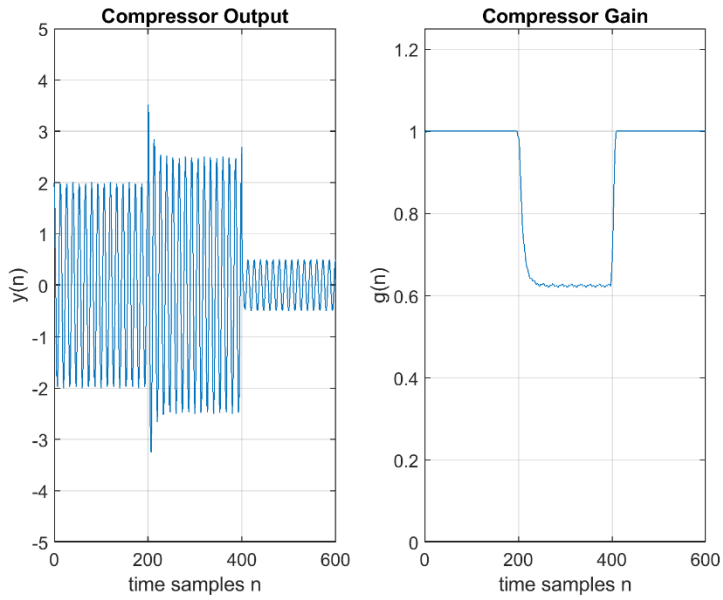



شکل ۱۷

```
L = 7;
g = zeros(1, 601);
lambda = 0.9;
rho = 0.1;
c = zeros(1, 601);
c(1) = 1.5;
for n = 2:600
    c(n) = lambda*c(n-1) + (1-lambda)*abs(x(n));
end


G = zeros(1, 601);
for n = 1:601
    if c(n) < c(1)
       G(n) = 1;
    else
       G(n) = (c(n)/c(1)).^(rho-1);
    end
end
g = movmean(G, L);
y = g.*x;




figure();
subplot(1,2,1);
plot(t, y);
ylim([-5 5])
title("Compressor Output");
xlabel('time samples n');
ylabel('y(n)');
grid on;
```

```
subplot(1,2,2);
plot(t, g);
ylim([0 1.25])
title("Compressor Gain");
xlabel('time samples n');
ylabel('g(n)');
grid on;
```

همین فرآیند را این بار با $\rho = 0.1$ و هموارساز انجام دادیم:



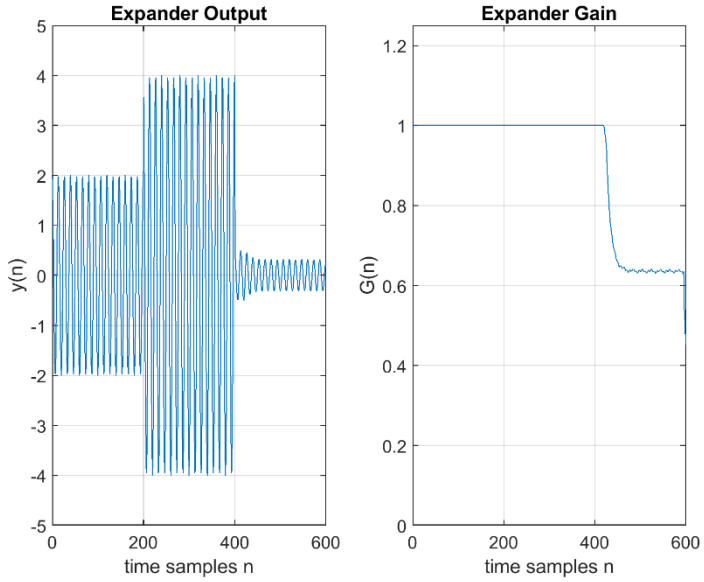

شکل ۱۸

برای Expander ۲ با مشخصات متفاوت هم انجام دادیم:

```
lambda = 0.9;
rho = 2; %and then rho = 10;
c = zeros(1, 601);
c(1) = 0.5;
for n = 2:600
    c(n) = lambda*c(n-1) + (1-lambda)*abs(x(n));
end
G = zeros(1, 601);
for n = 1:601
    if c(n) >= c(1)
       G(n) = 1;
    else
       G(n) = (c(n)/c(1)).^(rho-1);
    end
end
g = movmean(G, L);
y = g.*x;
figure();
subplot(1,2,1);
plot(t, y);
ylim([-5 5])
title("Expander Output");
```

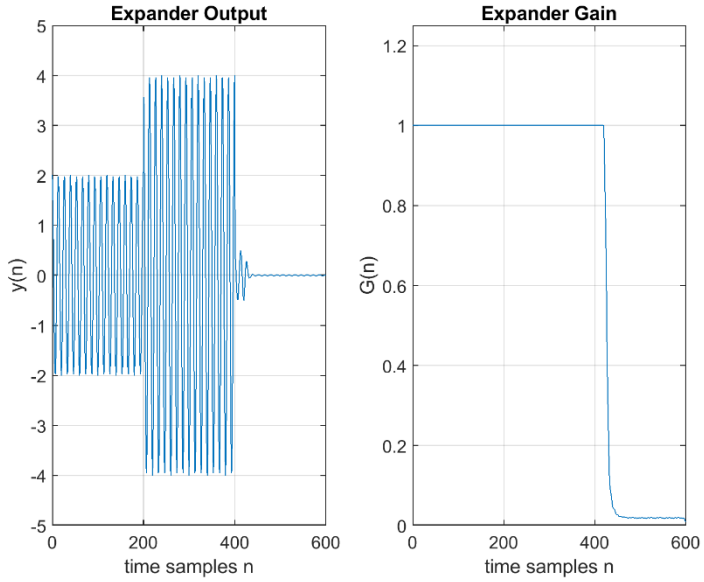
```
xlabel('time samples n');
ylabel('y(n)');
grid on;
subplot(1,2,2);
plot(t, g);
ylim([0 1.25])
title("Expander Gain");
xlabel('time samples n');
ylabel('G(n)');
grid on;
```

خروجی با rho = 2:




شکل ۱۹

خروجی با rho = 10:



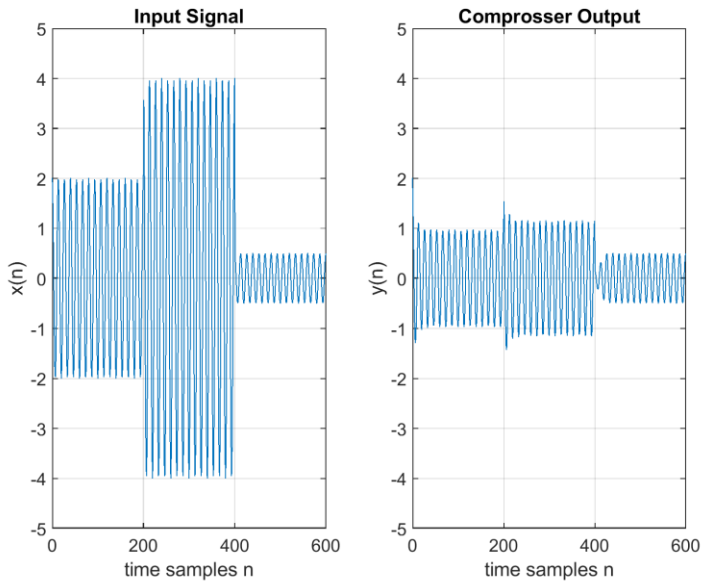

شکل ۲۰

بخش ۳-۳-ج)

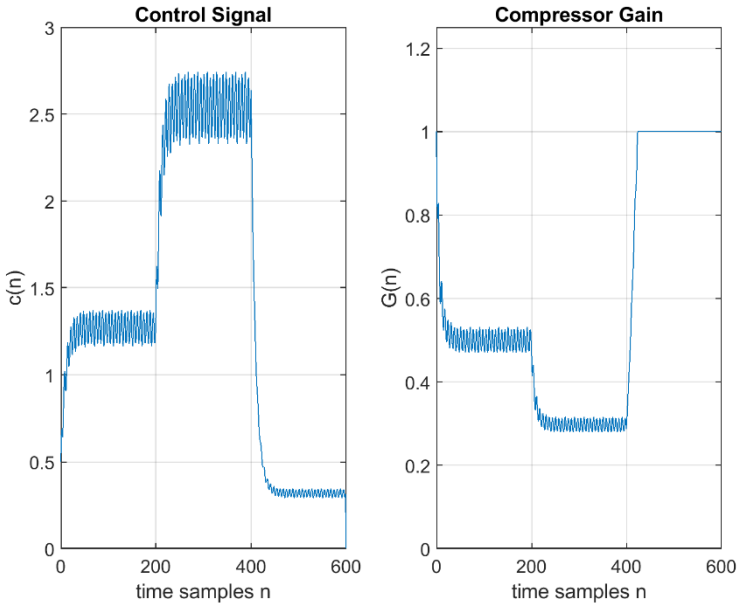
```
lambda = 0.9;
rho = 0.25;
c = zeros(1, 601);
c(1) = 0.5;
for n = 2:600
    c(n) = lambda*c(n-1) + (1-lambda)*abs(x(n));
end


G = zeros(1, 601);
for n = 1:601
    if c(n) >= c(1)
       G(n) = (c(n)/c(1)).^(rho-1);
    else
       G(n) = 1;
    end
    y(n) = G(n)*x(n);
end


figure();
subplot(1, 2, 1);
plot(t, x);
ylim([-5 5])
title("Input Signal");
xlabel('time samples n');
ylabel('x(n)');
grid on;


subplot(1,2,2);
plot(t, y);
ylim([-5 5])
title("Comprosser Output");
xlabel('time samples n');
ylabel('y(n)');
grid on;

figure();
subplot(1,2,1);
plot(t, c);
ylim([0 3])
title("Control Signal");
xlabel('time samples n');
ylabel('c(n)');
grid on;


subplot(1,2,2);
plot(t, G);
ylim([0 1.25])
title("Compressor Gain");
```

```
xlabel('time samples n');
ylabel('G(n)');
grid on;
```

نمودارهای فشرده‌ساز را این بار با $\rho = ۰.۲۵$ و سایر پارامترهای قبلی انجام دادیم.




شکل ۲۱



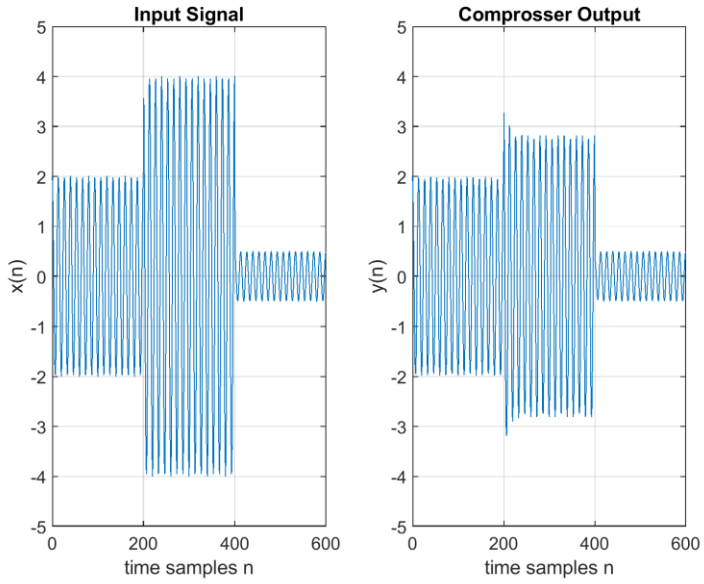

شکل ۲۲

**بخش ۳-۳-د)**

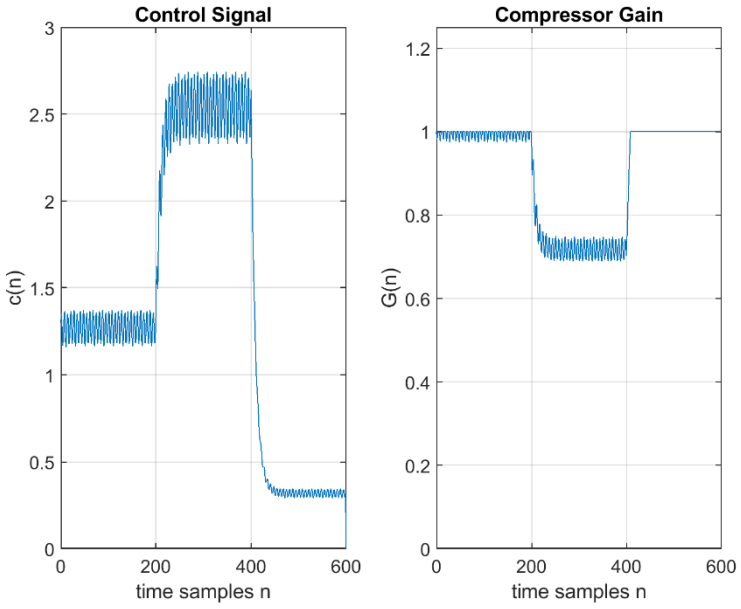
```
lambda = 0.9;
rho = 0.5;
c = zeros(1, 601);
c(1) = 1.3;
for n = 2:600
    c(n) = lambda*c(n-1) + (1-lambda)*abs(x(n));
```

```
end


G = zeros(1, 601);
for n = 1:601
    if c(n) >= c(1)
       G(n) = (c(n)/c(1)).^(rho-1);
    else
       G(n) = 1;
    end
    y(n) = G(n)*x(n);
end


figure();
subplot(1, 2, 1);
plot(t, x);
ylim([-5 5])
title("Input Signal");
xlabel('time samples n');
ylabel('x(n)');
grid on;


subplot(1,2,2);
plot(t, y);
ylim([-5 5])
title("Comprosser Output");
xlabel('time samples n');
ylabel('y(n)');
grid on;

figure();
subplot(1,2,1);
plot(t, c);
ylim([0 3])
title("Control Signal");
xlabel('time samples n');
ylabel('c(n)');
grid on;


subplot(1,2,2);
plot(t, G);
ylim([0 1.25])
title("Compressor Gain");
xlabel('time samples n');
ylabel('G(n)');
grid on;
```

نمودارهای فشرده‌ساز را این بار با c0 = ۱.۳، ρ = ۰.۵ و λ = ۰.۹ رسم کردیم:




شکل ۲۳




شکل ۲۴

در مجموع کمپرسور با پارامترهای داده شده λ=0.9 ، rho = 0.5 و c(1) = 1.3 برای کاهش دامنه سیگنال ورودی x طراحی شده است. پارامتر lambda مقدار فشرده سازی اعمال شده را کنترل می‌کند، rhoشیب منحنی فشرده سازی را تعیین می‌کند و c(1) سطح آستانه‌ای را که فشرده‌سازی بالاتر از آن شروع می‌شود، تعیین می‌کند.

در این حالت، فقط قسمت میانی سیگنال فشرده می‌شود زیرا سطح آستانه c(1) = 1.3 مقدار بالایی تنظیم شده است. این بدان معنی است که تنها بخش‌هایی از سیگنال که از این آستانه فراتر می‌روند فشرده می‌شوند. از آنجایی که سیگنال ورودی x یک موج سینوسی با دامنه پیک ۱ است، بیشتر انرژی آن حول این مقدار پیک متمرکز می‌شود.

پارامتر لامبدا در ۰.۹ نشان می‌دهد که مقدار قابل توجهی از حافظه در کمپرسور وجود دارد، به این معنی که هنگام تعیین میزان فشرده‌سازی سطح فعلی، تاریخچه بیشتری از سطوح سیگنال گذشته را در نظر می‌گیرد. این منجر به منحنی فشرده‌سازی صاف‌تر می‌شود و از تغییرات ناگهانی در افزایش جلوگیری می‌کند.

پارامتر rho = 0.5 هم شیب متوسطی را برای منحنی فشرده‌سازی تعیین می‌کند، به این معنی که با افزایش سطح سیگنال بالاتر از آستانه، با نرخی متناسب با فاصله آن از آستانه فشرده می‌شود.