



دانشگاه صنعتی شریز

دانشکده مهندسی برق و الکترونیک

گزارش کار پروژه درس کنترل صنعتی (PID)

دانشجو:

نیما جهان بازفرد (400113020)

استاد درس:

جناب آقای دکتر مختار شاصادقی

آذر 1403

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

فهرست مطالب

عنوان	صفحه
بخش اول(طراحی PID به صورت حلقه باز)	
۱-۱ به روش ZN	۲
۱-۲ به روش CC	۱۰
۱-۳ به روش CHR-SERVO	۱۹
۱-۴ به روش CHR-REGULATORY	۲۹
بخش دوم(طراحی PID به صورت حلقه بسته)	
۲-۱ به روش ZN	۳۹
۲-۲ به روش ZN همراه با ورودی مرجع وزن دار	۴۹
۲-۳ به روش ZN نوسان میرا	۵۸
۲-۴ به روش PR	۶۸
بخش سوم(طراحی PID بر اساس حد بهره و حد فاز-ZN تعمیم یافته)	
۳-۱ دستیابی به حد بهره 6dB	۷۹
۳-۲ دستیابی به حد فاز deg 45	۸۸
۳-۳ بهینه سازی برای دست یابی به حد فاز deg 45 و حد بهره 6 dB	۹۸
بخش چهارم(بررسی و مقایسه و انجام عمل integral anti windup برای بخش ۳-۳ و مقایسه و بررسی آن)	
۴-۱ بررسی کلی تاثیرات عوامل خارجی بر پاسخ	۱۰۹
۴-۲ تشکیل جدول مقایسه و بررسی تاثیر حد فاز و حد بهره(پاسخ پله، اغتشاش پله)	۱۱۳
۴-۳ نمای کلی از سیمولینک کلیه ای طراحی ها	۱۱۶

مقدمه

قبل از شروع گزارشکار این پروژه باید چند نکته ذکر شود. اولین مورد در بخش آخر پروژه قبلی که مدل

سازی سیستم های با رفتار انگرالی بود در صورتی که در بخش سیمولینک توابع تبدیل مدل شده در $\frac{1}{S}$

ضرب شده ولی به طور واضح این مورد در گزارشکار ذکر نشده است ، پس این مورد باید ذکر می شد که

توابع مدل شده بر اساس ورودی ضربه باید در انتهایا در $\frac{1}{S}$ ضرب می شدند. دومین مورد این است که بجای

$$\frac{63}{(S+0.5)(S+2)(S+4)} \text{ استفاده} \quad \frac{21}{(S+1)(S+3)(S+7)} \text{ سیستم}$$

شده است. مورد بعدی در مورد بخش بندی این گزارشکار این گونه می باشد که در ابتدای امر کد های

مربوطه به هر بخش ابتدای کار قرار داده میشود که هر بخش کد با کمک کامنت های قرار داده شده به

طور واضح مشخص است که چکاری انجام می دهد . در ادامه کد ها توضیحات مربوط به خروجی کد های

زده شده و همچنین خروجی سیستم در شرایط مختلف (ورودی پله،اغتشاش،نویز،تغییر پارامتر و اشباع

محرك) نشان داده شده و توضیحات مربوط به آن داده میشود. قابل ذکر است که مورد تغییر پارامتر در

$$\frac{21}{(S+0.5)(S+2)(S+4)} \text{ در نظر گرفته شده و همچنین برای بررسی تاثیر}$$

نویز بر سیستم هم برای همه ی طرای ها نویز سینوسی با فرکانس 60 هرتز و با دامنه ی 0.1 در نظر

گرفته شده است. و مورد بعدی این می باشد که حد فاز و حد بهره‌ی پلنت مورد بررسی به ترتیب برابر با
۰.۶ dB و 2.04 deg می باشد.

بخش اول

طراحی PID به صورت حلقه باز

ZN به روشنی 1-1

بعد از تغییر سیستم و مدلسازی بر اساس G5 که بهترین عملکرد را بین مدل ها داشت این گونه ادامه می

دھیم .

```
%ZN OPEN LOOP
k=15.7452;
T=2.37;
T_d=0.5788;
alpha=T_d/T;
%PID CONTROLER PARAMERTERS
kp=1.2/(k*alpha);
Ti=2*T_d;
Td=Ti/4;
%KP=0.3121
%Ti=1.1576
%Td=0.2894
s=tf('s');
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
g=63/((s+0.5)*(s+2)*(s+4));
hold on
nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'r')
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=42.05 deg
%GMcg=25.8672 dB

%calculating of characteristics of system response to pulse input

t = out.zop(:, 1);
y = out.zop(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);
```

```

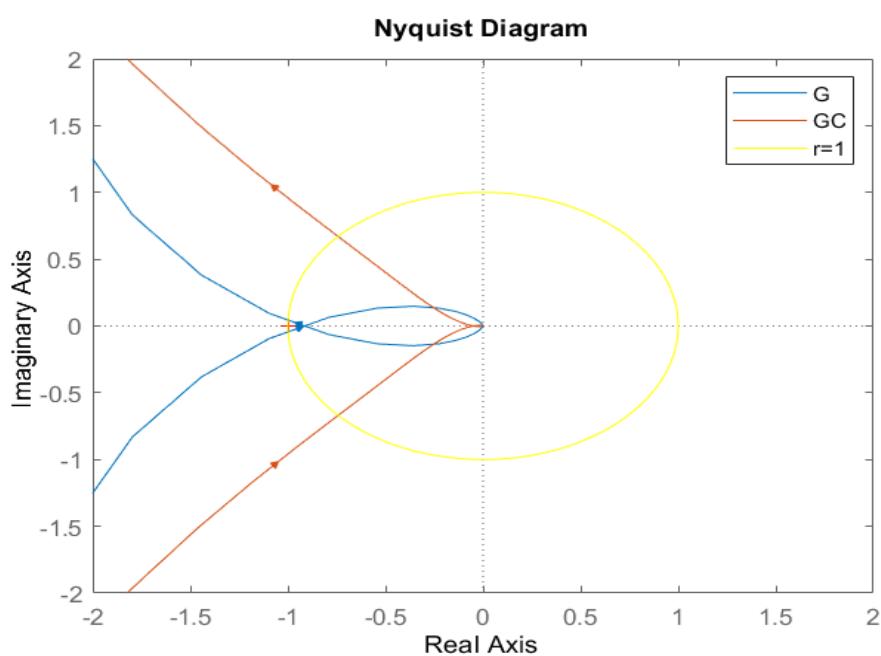
% calculating touching points
t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
    (line_value - y(idx)) ./ (y(idx+1) - y(idx));
y_intersect = line_value * ones(size(t_intersect)); % value of output in
touching points
%drawing lines
plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
1.2);

% showing touching points
plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
    'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
'UniformOutput', false)]};
legend(legend_entries{:});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.62
%Ts%2=6.1
%overshoot%=43.7%
%d=0.08
%IAE=1.2990

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبهٔ ضرایب PID می‌کنیم که با مقایسهٔ دایاگرام‌های نایکوپیست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به 45 درجه نزدیک شده است(42) و همچنین بهره سیستم جبران شده بر حسب dB خیلی بیشتر شده است(25.87).

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه ای موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه ای طراحی ها در یک جدول این موارد آورده می شود.

مشخصات گفته شده از روی این نمودار بدست آمده است:

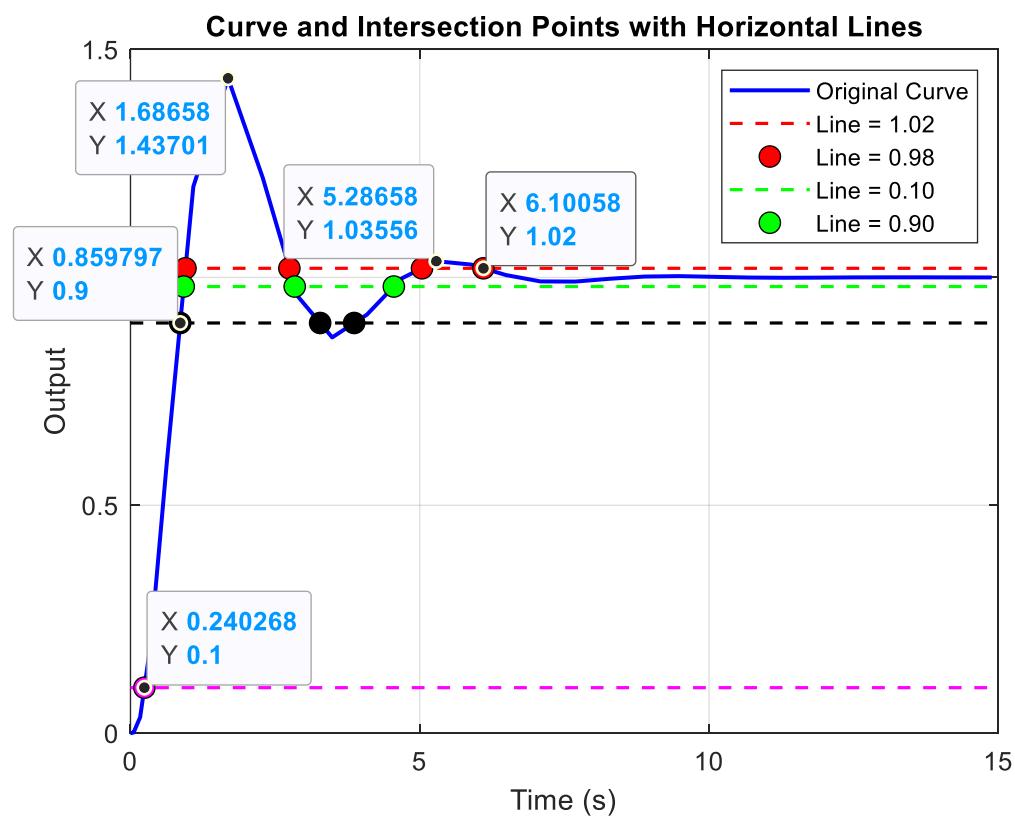
$$Tr=0.62$$

$$Ts\%2=6.1$$

$$overshoot\% = 43.7\%$$

$$d=0.08$$

$$IAE=1.2990$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ۱۰ اعمال شده است):

$T_s(5\%)=7.45$

$T_s(10\%)=6.03$

$overshoot\% = 228\%$

$d=0.08$

$IAE=4.4827$

```
t = out.zop1(:, 1);

y = out.zop1(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth', 1.2);

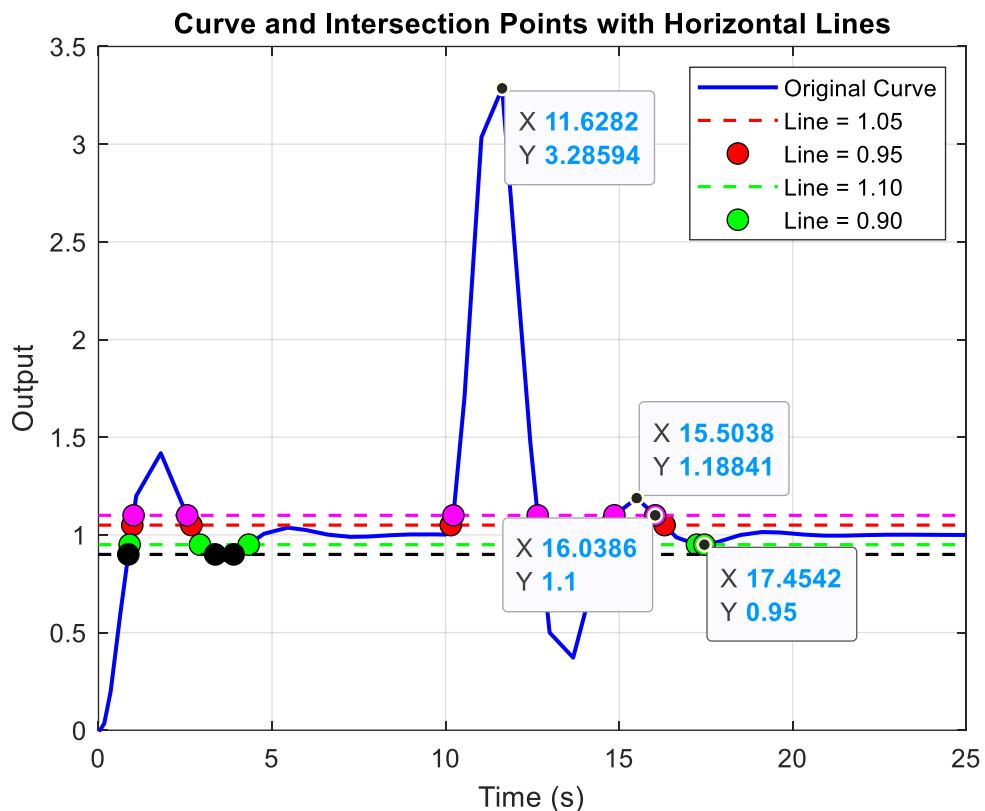
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines, 'UniformOutput', ...
    false)]};
legend(legend_entries{::});
grid on;
```

```

hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-1.2990;
%Ts5%=7.45
%Ts10%=6.03
%overshoot%=228%
%d=0.08
%IAE=4.4827

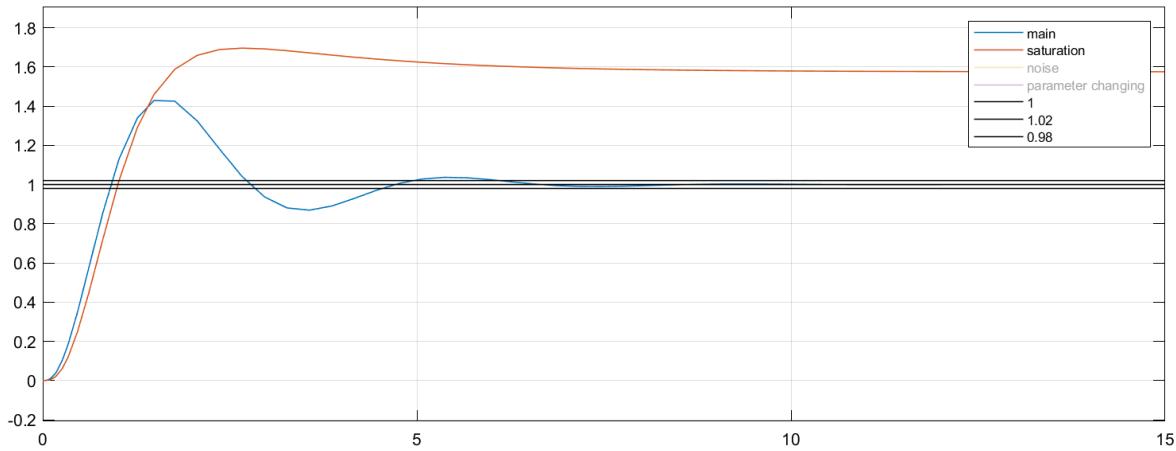
```



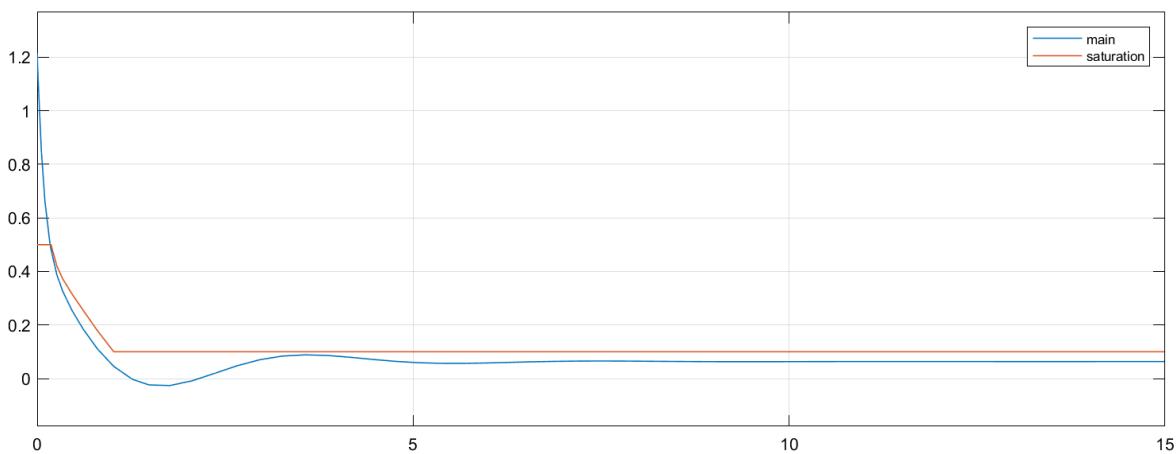
در ادامه ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می شود(نویز ، تغییر پارامترو...):

اشباع محرک:

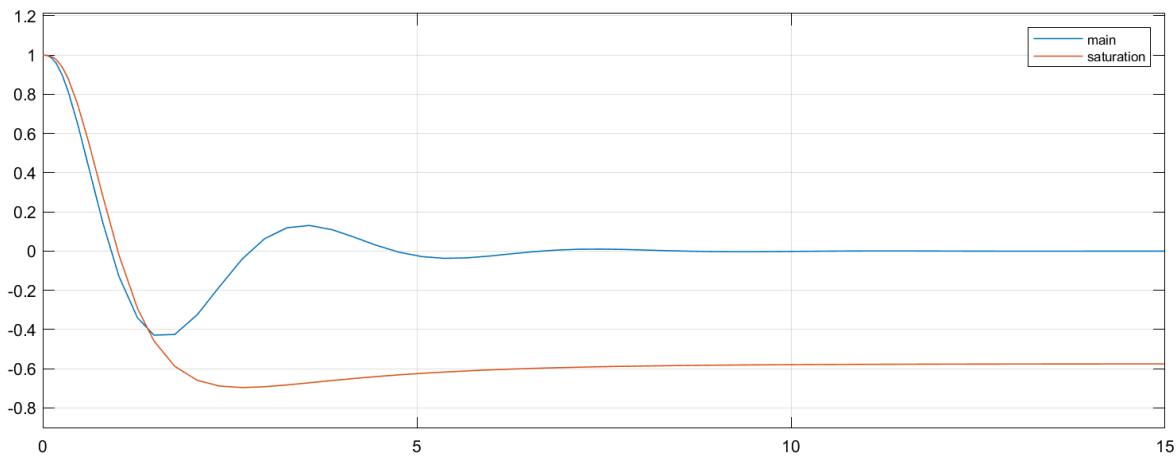
در این بخش به مقایسه ی خروجی ها بر اثر اشباع و خروجی اصلی می پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطای خطا شده است(خطای صفر نمی شود). در ادامه به مقایسه ی سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



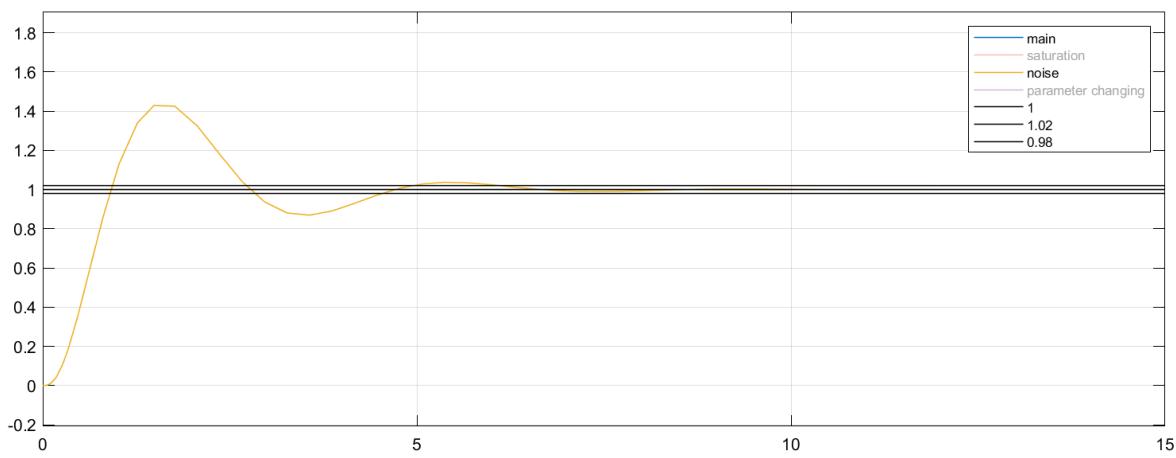
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطای شود ادامه با مقایسه ی سیگنال های خطا داریم:



همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای انباشته شده صفر نمی شود.

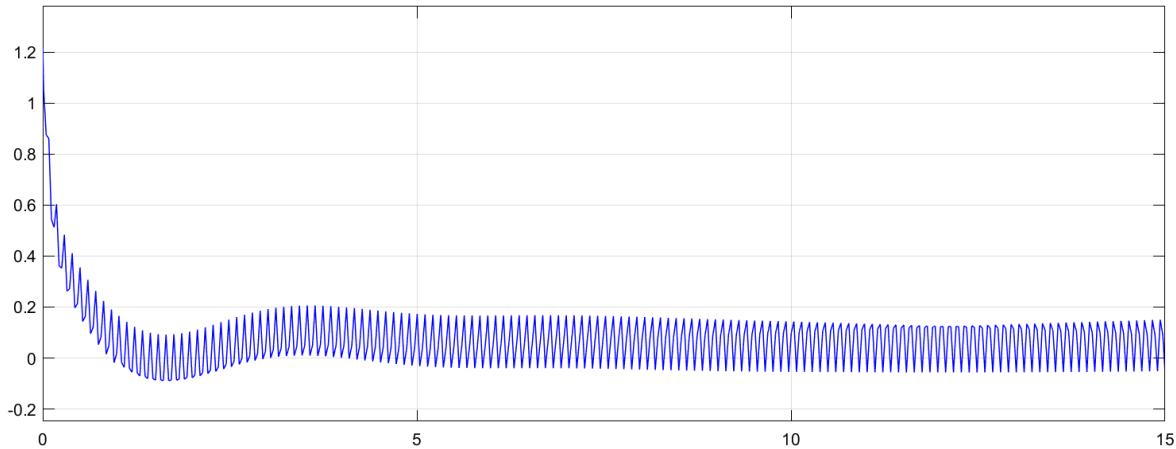
اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:



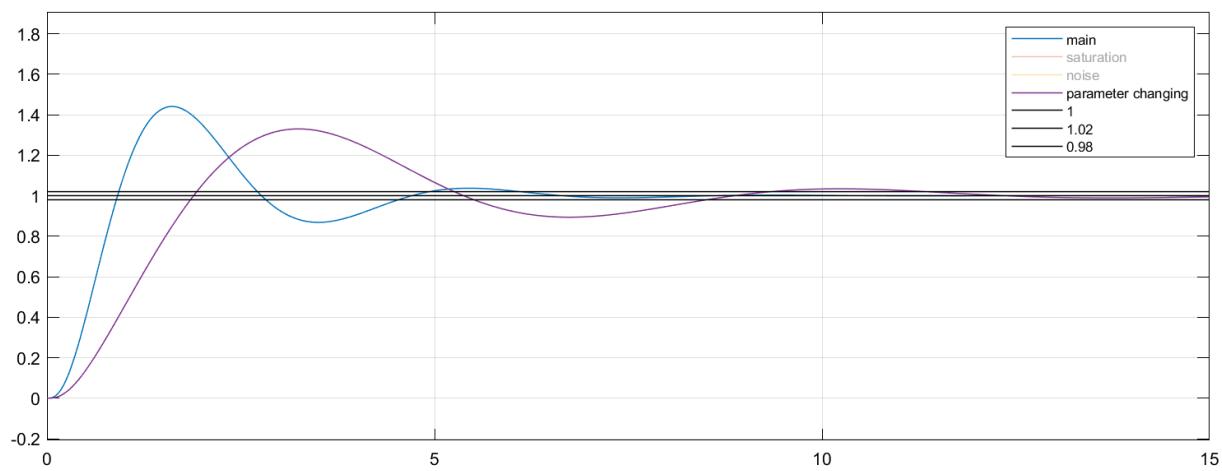
همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم

تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم که:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و کاهش ماکزیمم فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

CC به روش 2-1

```
%CC OPEN LOOP
k=15.7452;
T=2.37;
T_d=0.5788;
alpha=T_d/T;
%PID CONTROLER PARAMERTERS
kp=(1/k)*((1.35/alpha)+0.27);
Ti=T_d*((2.5+(0.5*alpha))/(1+(0.6*alpha)));
Td=T_d*(0.37/(1+0.2*alpha));
%KP=0.3682
%Ti=1.3237
%Td=0.2042
s=tf('s');
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
g=63/((s+0.5)*(s+2)*(s+4));
hold on
nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y, 'r')
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.4 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=35.47 deg
%GMcg=25.59 dB

t = out.cop(:, 1);
y = out.cop(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);
```

```

% showing touching points
plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
      'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

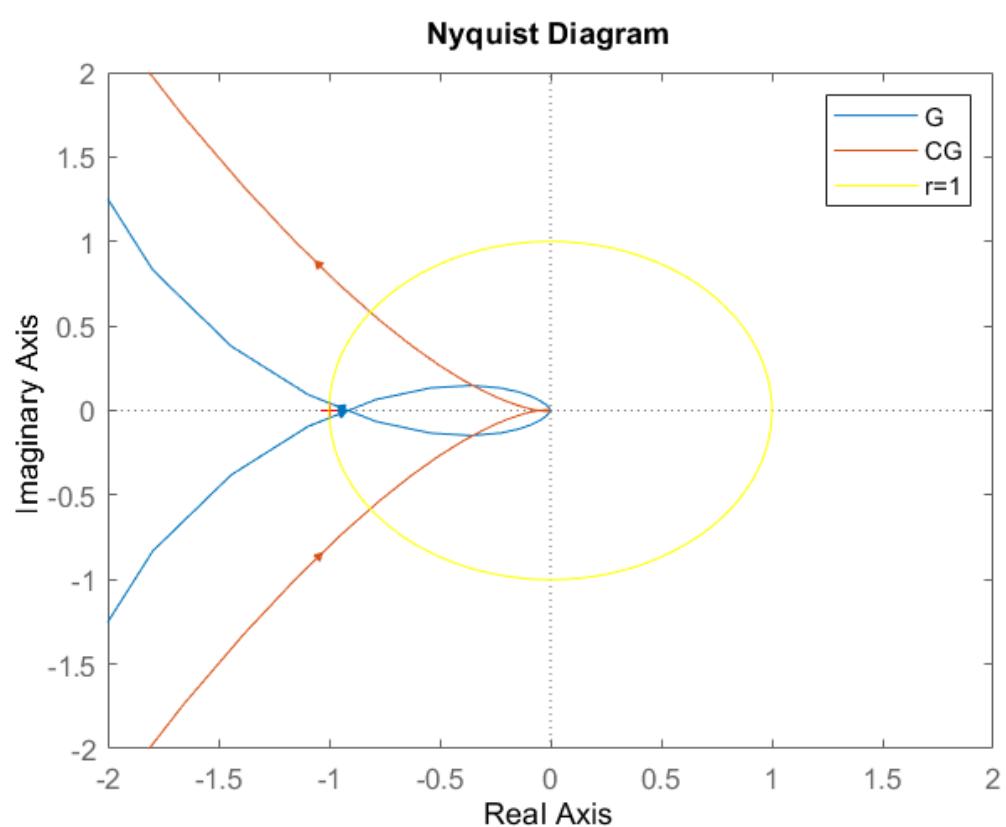
xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
'UniformOutput', false)]};
legend(legend_entries{:, :});
grid on;
hold off;

error=1-y;
absIntegralError = trapz(t, abs(error));

%Tr=0.56
%Ts%2=9.85
%overshoot%=57.38%
%d=0.34
%IAE=1.9798

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبه‌ی ضرایب PID می‌کنیم که با مقایسه‌ی دایاگرام‌های نایکوییست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به 45 درجه نزدیک شده است(35.5) و همچنین بهره سیستم جبران شده بر حسب dB خیلی بیشتر شده است(25.6). در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه طراحی ها در یک جدول این موارد آورده می شود. مشخصات گفته شده از روی این نمودار بدست آمده است:

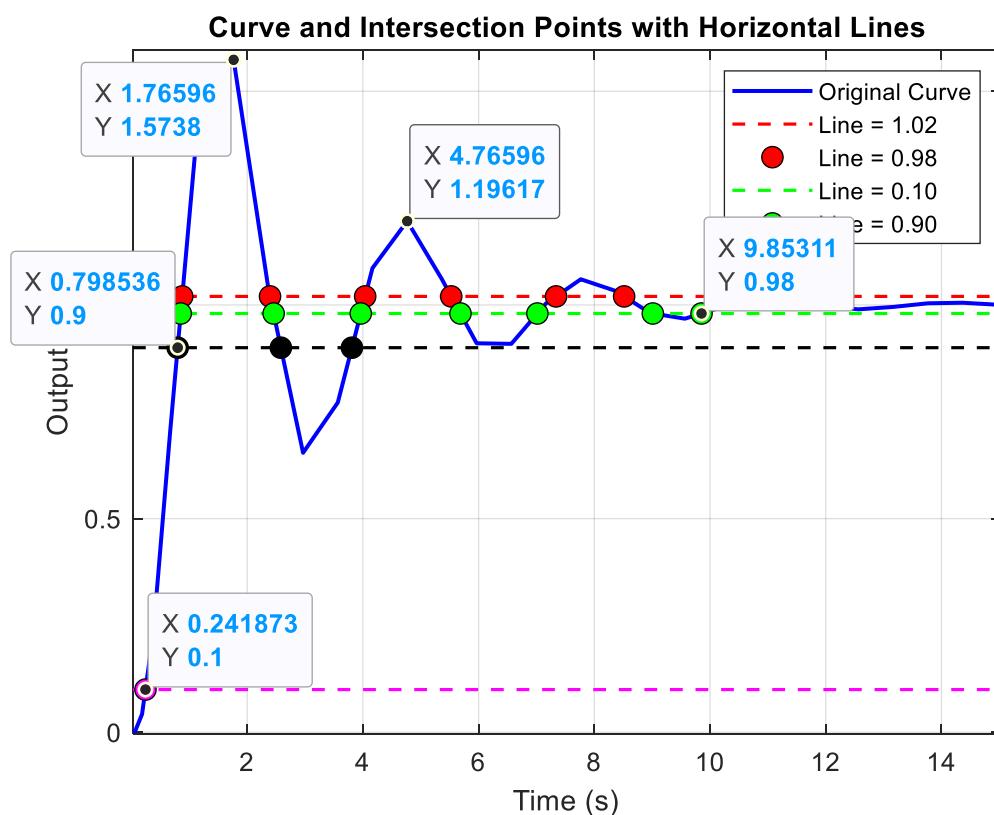
$Tr=0.56$

$Ts\%2=9.85$

$overshoot\% = 57.38\%$

$d=0.34$

$IAE=1.9798$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه طراحی ها این مشخصات در جدول انتهای

گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه

ی 15 اعمال شده است):

Ts5%=10.89

Ts10%=9.4

overshoot%=227%

d=0.27

IAE=5.34

```
t = out.cop1(:, 1);

y = out.cop1(:, 2);

plot(t, y '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

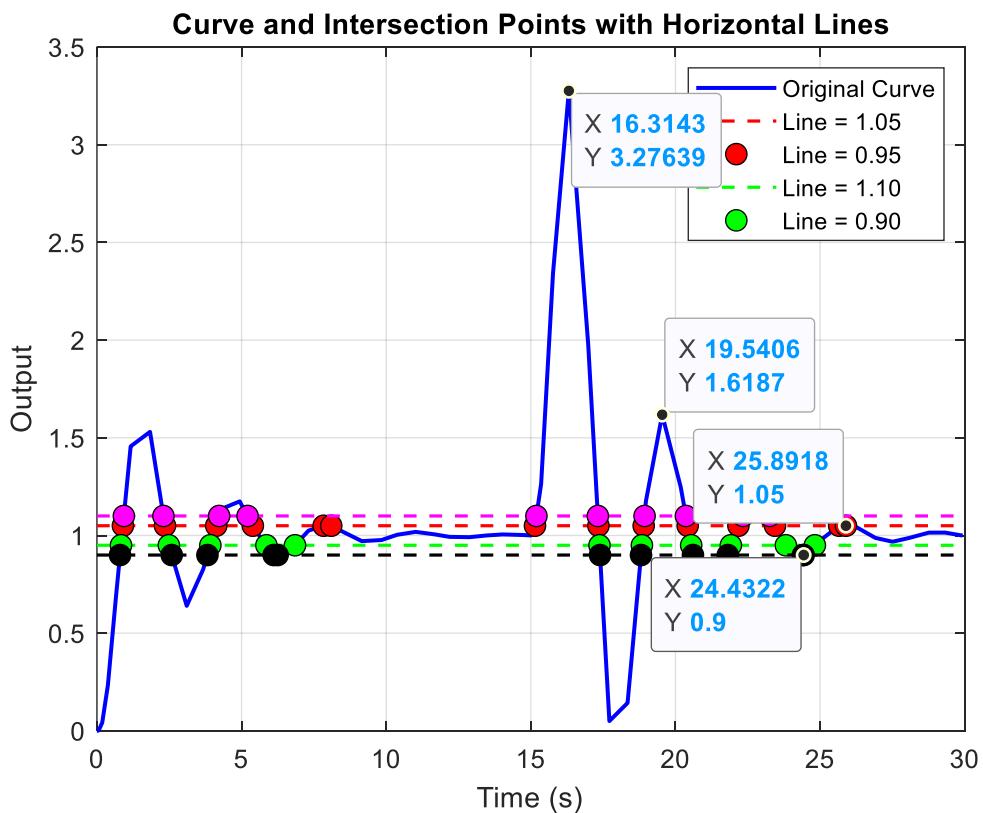
% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    %touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-1.9798;
```

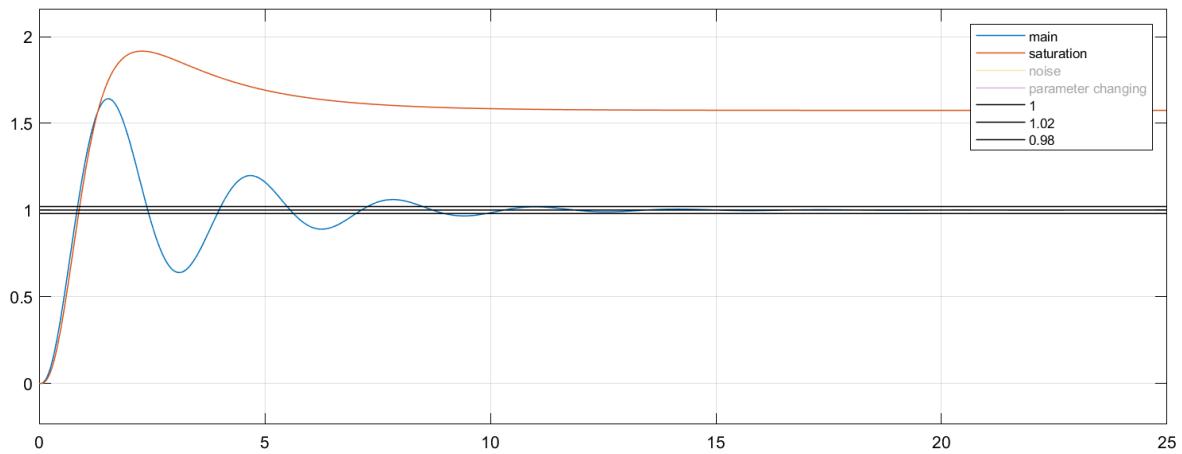
```
%Ts5%=10.89
%Ts10%=9.4
%overshoot%=227%
%d=0.27
%IAE=5.34
```



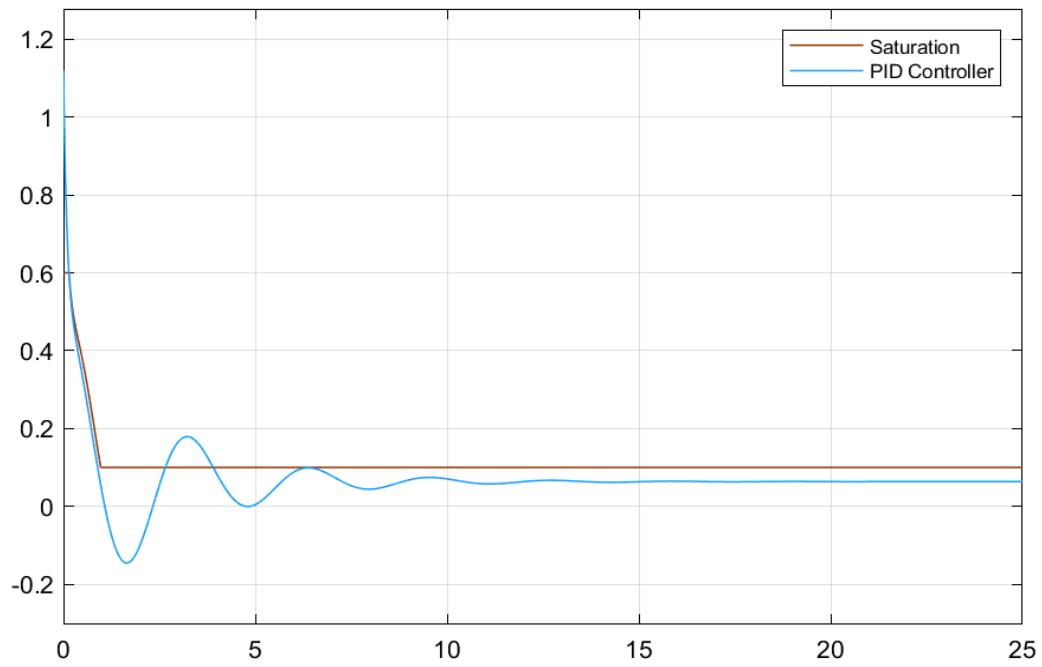
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامتر و...):

اشباع محرک:

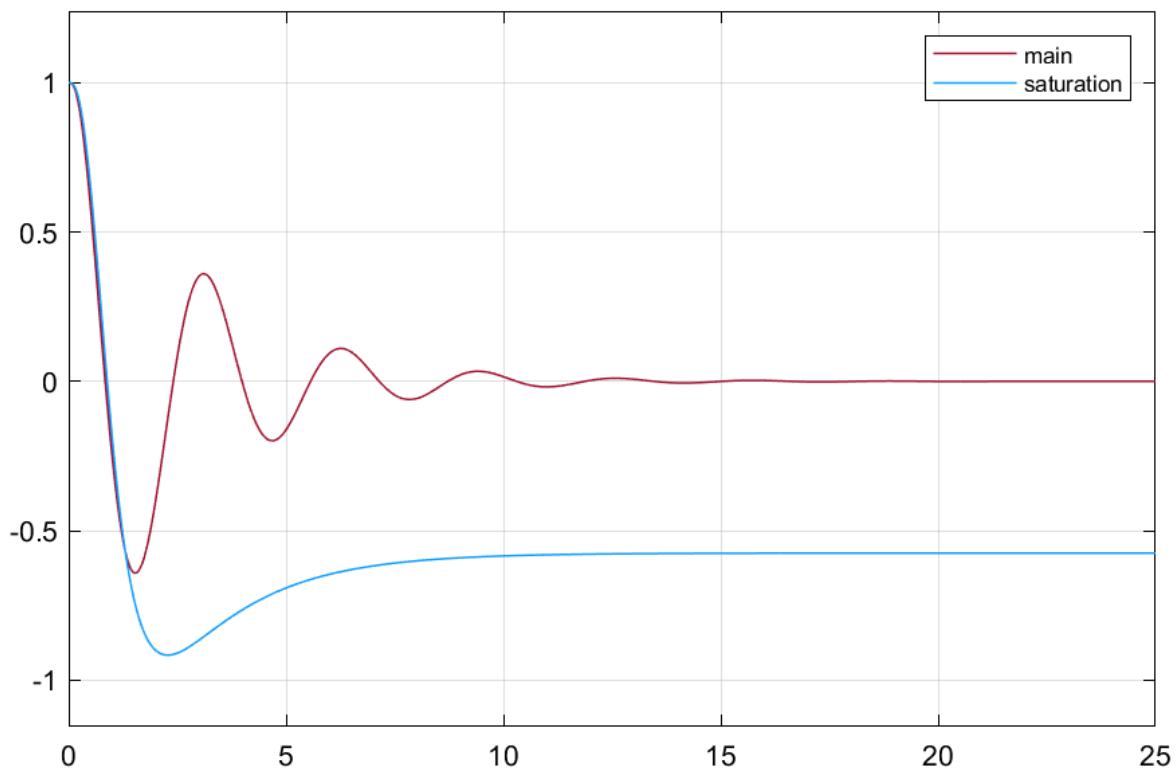
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطا شده است(خطا صفر نمی شود). در ادامه به مقایسه ی سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



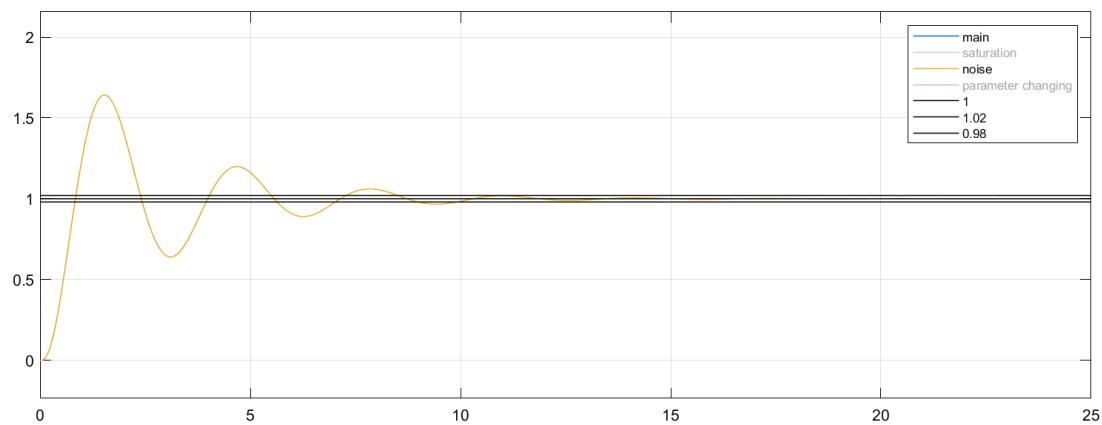
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطأ می شود ادامه با مقایسه ی سیگنال های خطأ داریم:



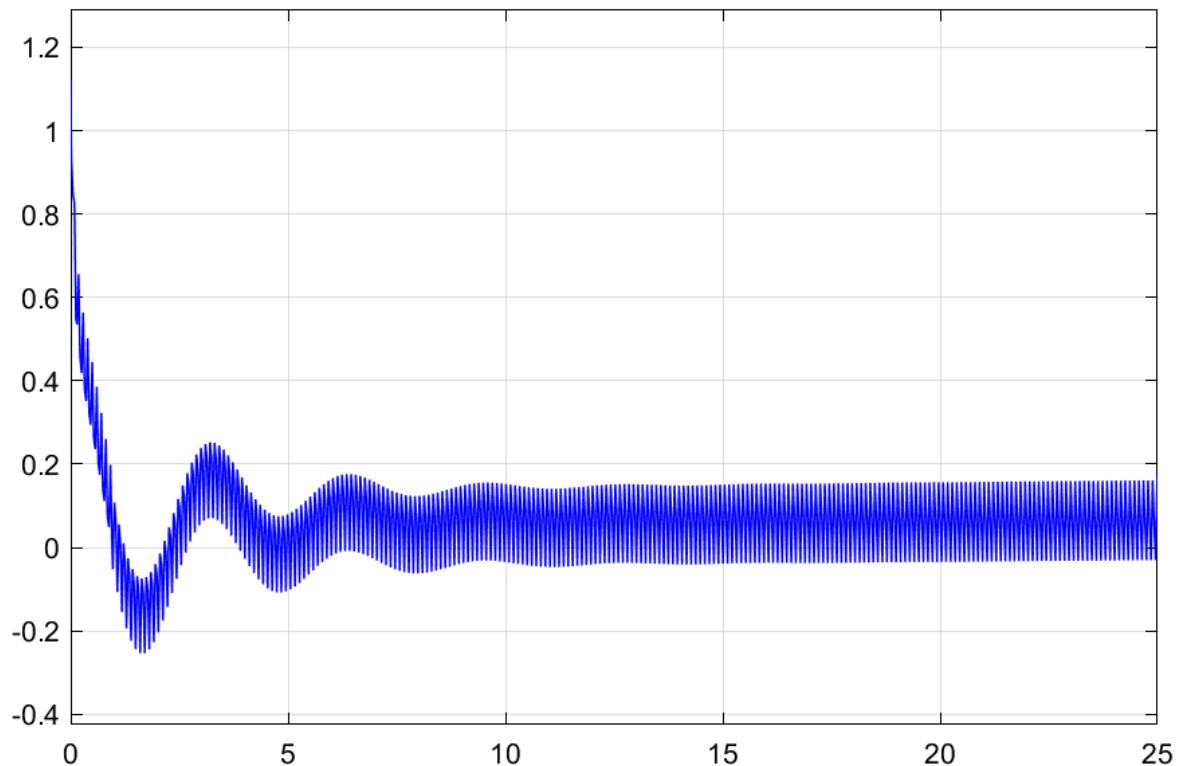
همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای انباشته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

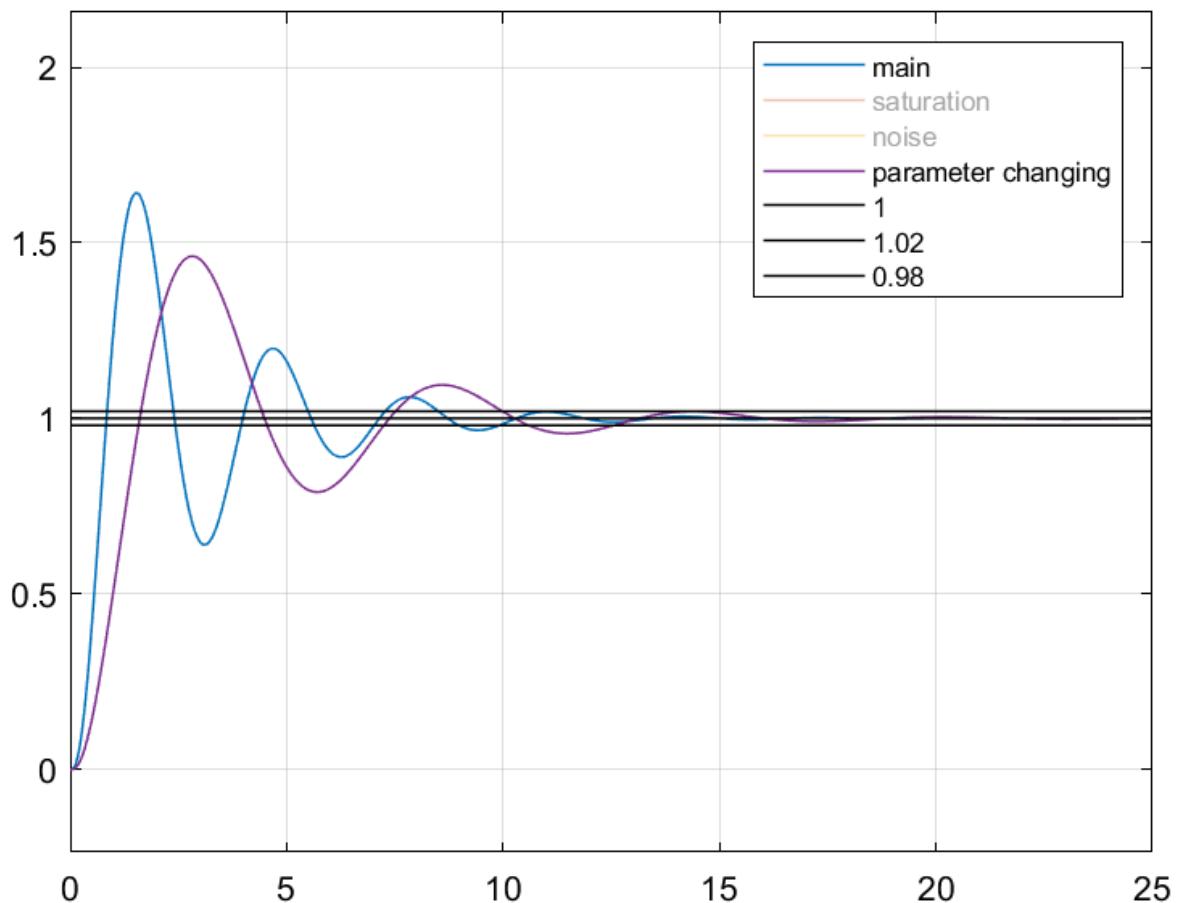


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و کاهش ماقزیم فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

CHR-SERVO 3-1 به روش

```
%CHR_SERV OPEN LOOP
k=15.7452;
T=2.37;
T_d=0.5788;
alpha=T_d/T;

%PID CONTROLER PARAMERTERS
kp=0.6/(k*alpha);
Ti=T_d;
Td=T_d*0.5;
%KP=0.1560
%Ti=0.5788
%Td=0.2894
s=tf('s');
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
g=63/((s+0.5)*(s+2)*(s+4));
hold on
```

```

nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'r');
legend('G','GC','r=1');
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.4 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=16.8863 deg
%GMcg=31.5971 dB


t = out.chos(:, 1);
y = out.chos(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:, :});
grid on;
hold off;

```

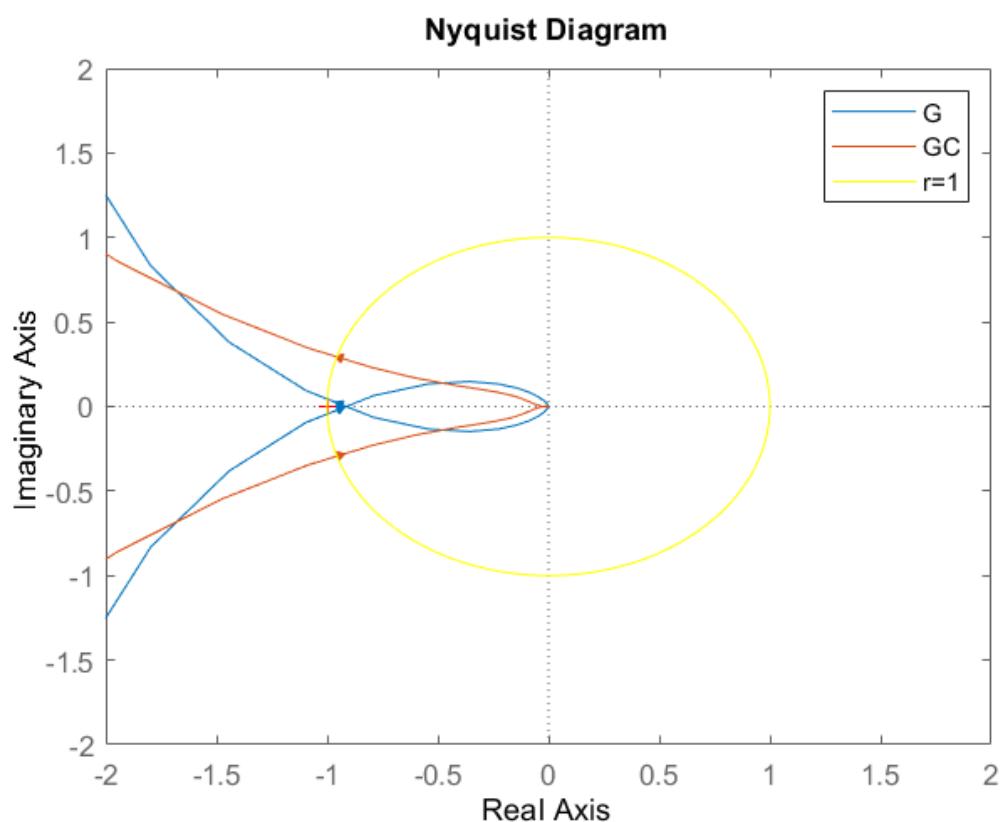
```

error=1-y;
absIntegralError = trapz(t, abs(error));

%parameters for step
%Tr=1.24
%Ts%2=5.16
%overshoot%=9.8%
%d=-
%IAE=1.1537

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبه‌ی ضرایب PID می‌کنیم که با مقایسه‌ی دایاگرام‌های نایکوییست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به ۴۵ درجه نزدیک شده است(31.6) و همچنین بهره سیستم جبران شده بر حسب dB خیلی بیشتر شده است(16.9).

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه‌ی موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه‌ی طراحی‌ها در یک جدول این موارد آورده می‌شود.

مشخصات گفته شده از روی این نمودار بدست آمده است:

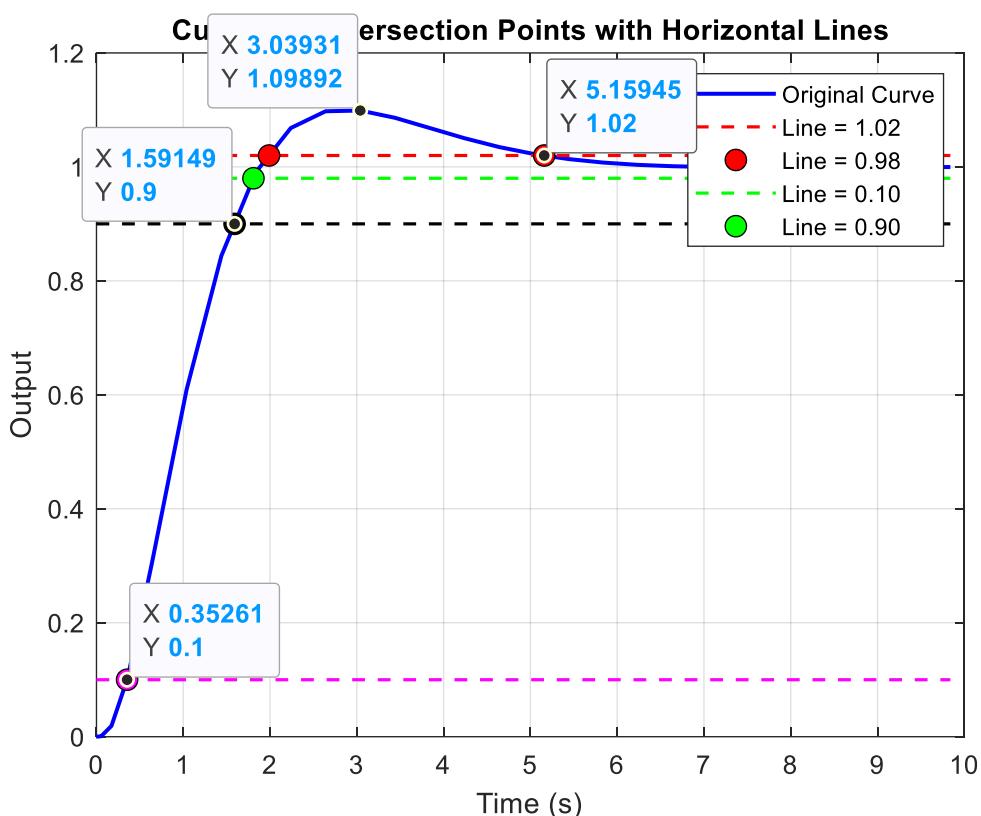
$T_r = 1.24$

$T_{s2\%} = 5.16$

overshoot% = 9.8%

$d = -$

$IAE = 1.1537$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ی طراحی ها این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ی 8 اعمال شده است):

$T_{s5\%} = 6.16$

$T_{s10\%} = 5.94$

overshoot% = 382%

d=-

IAE=11.08

```
t = out.chos1(:, 1);

y = out.chos1(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

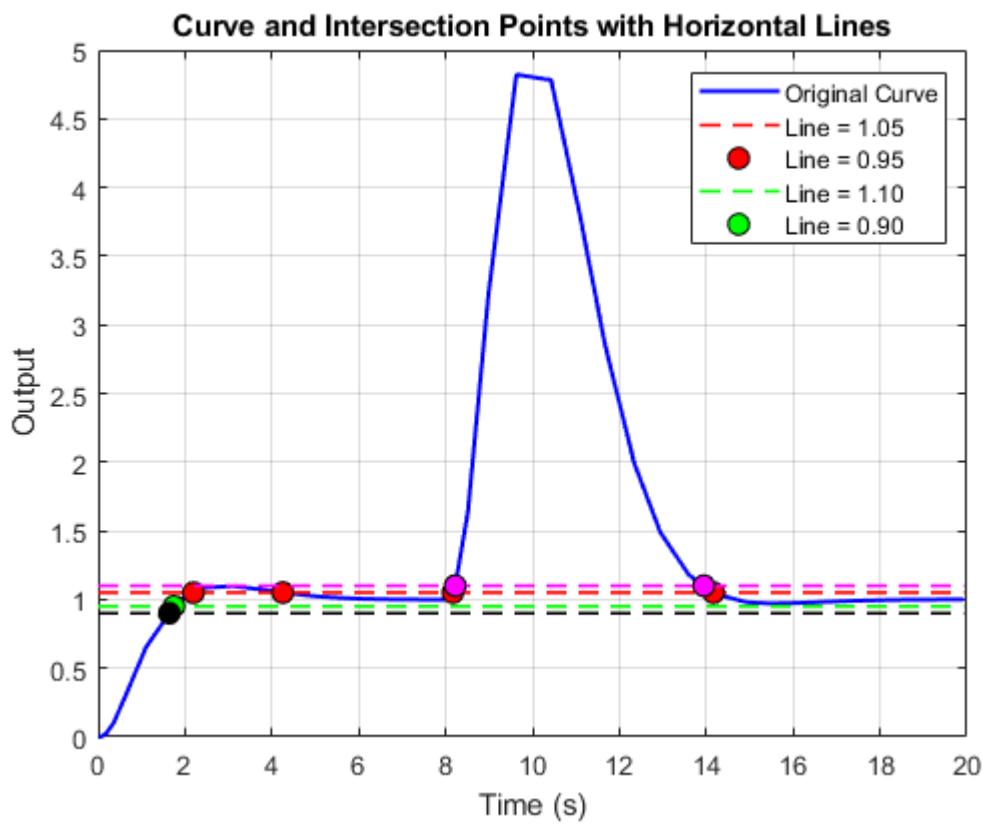
horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

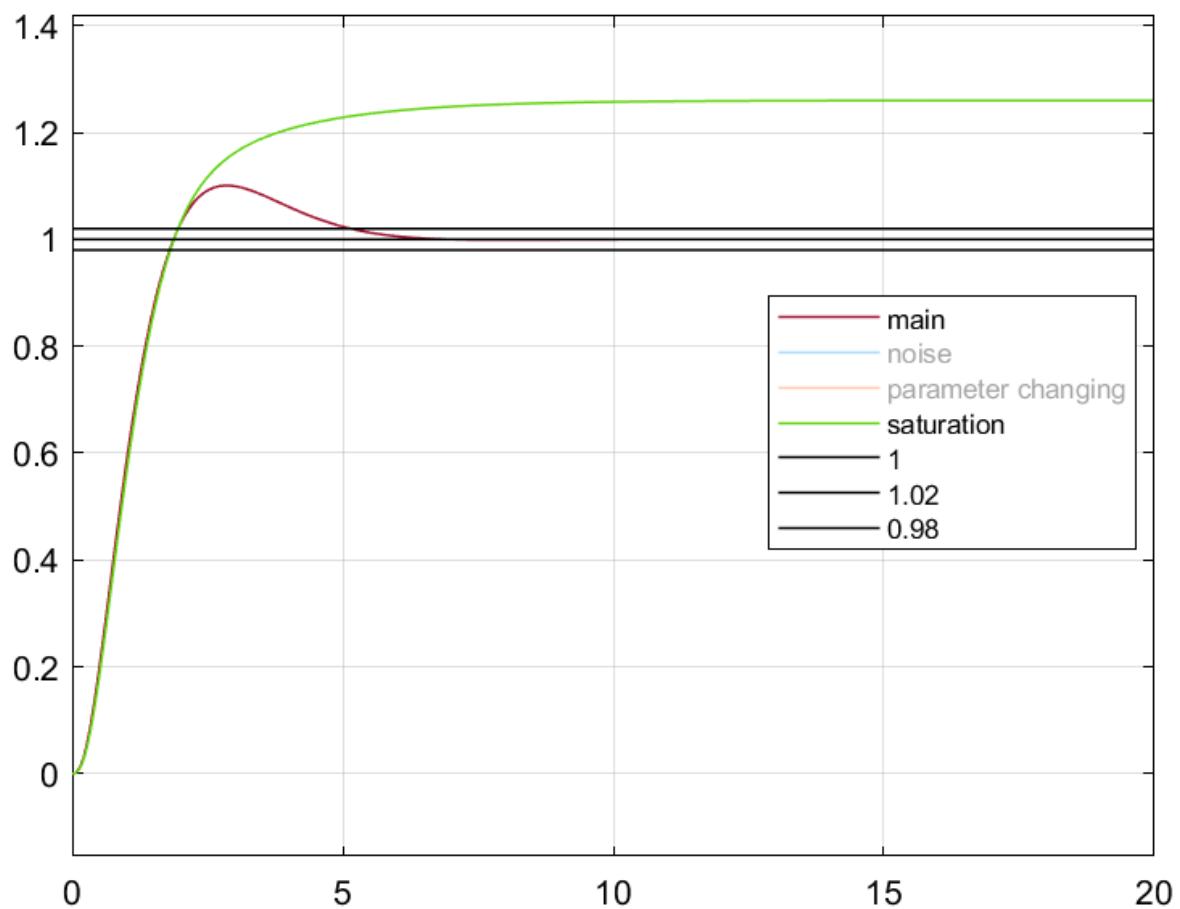
xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-1.1537;
%Ts5%=6.16
%Ts10%=5.94
%overshoot%=382%
%d=-
%IAE=11.08
```



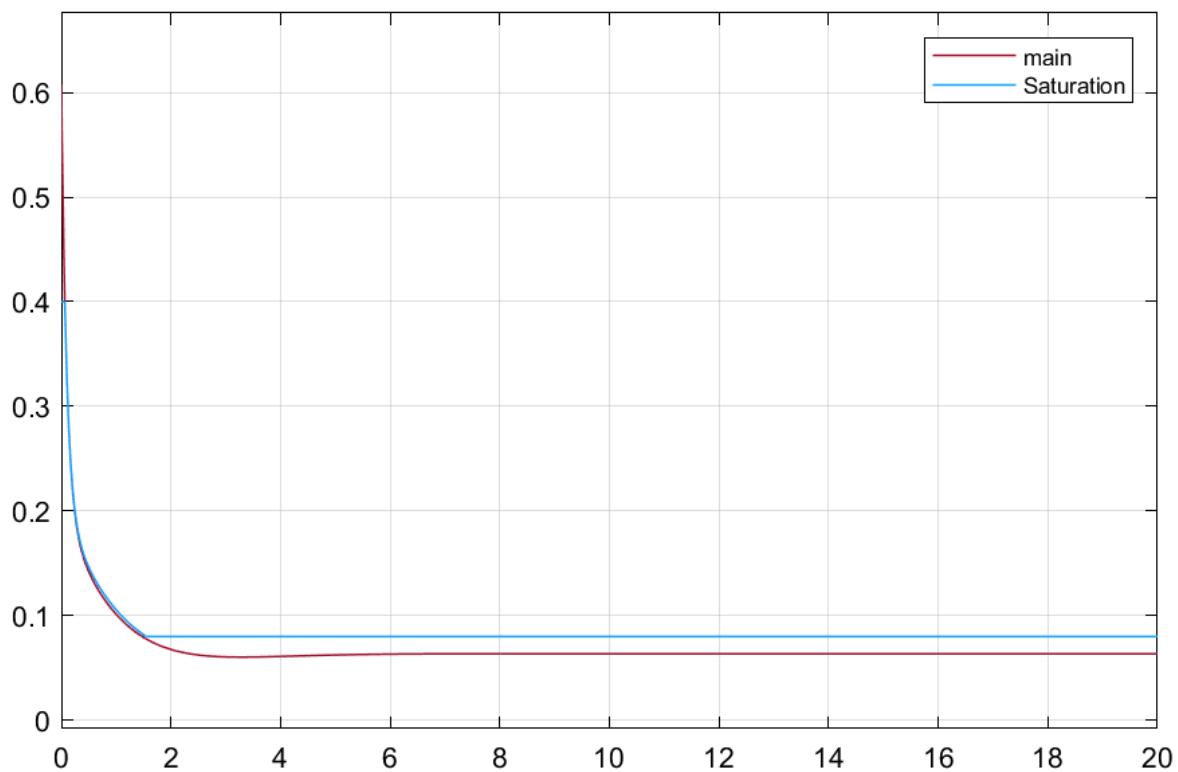
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامترو...):

اشباع محرک:

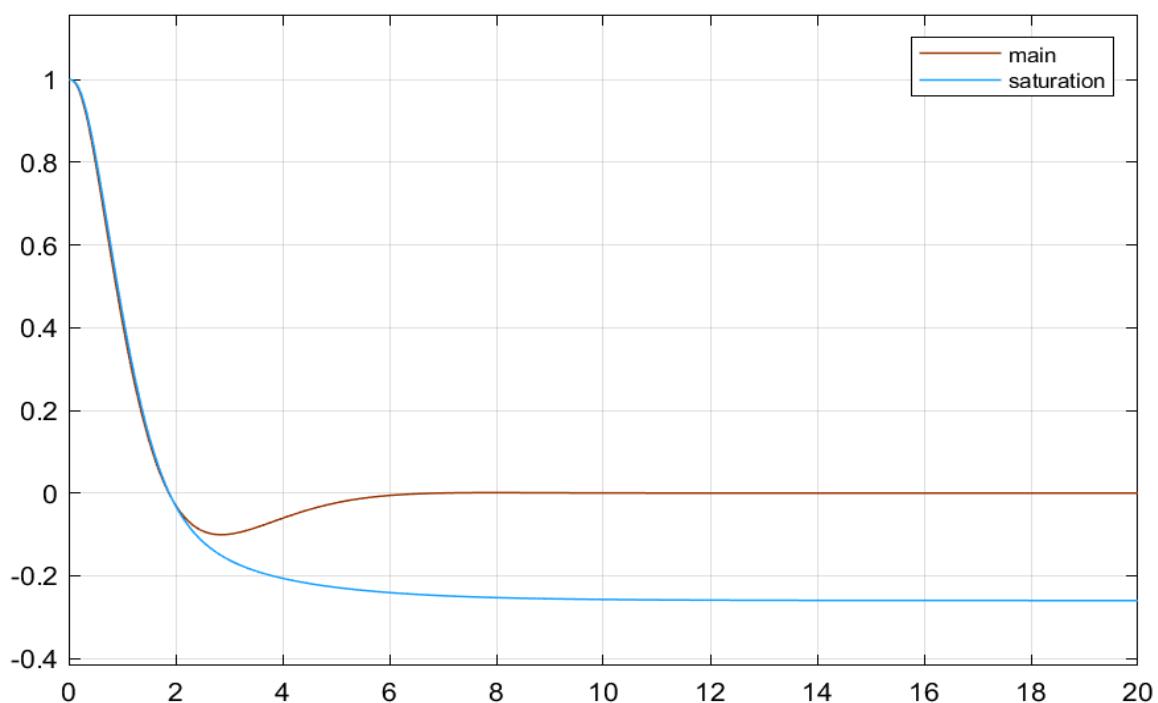
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطأ شده است(خطا صفر نمی شود). در ادامه به مقایسه ای سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



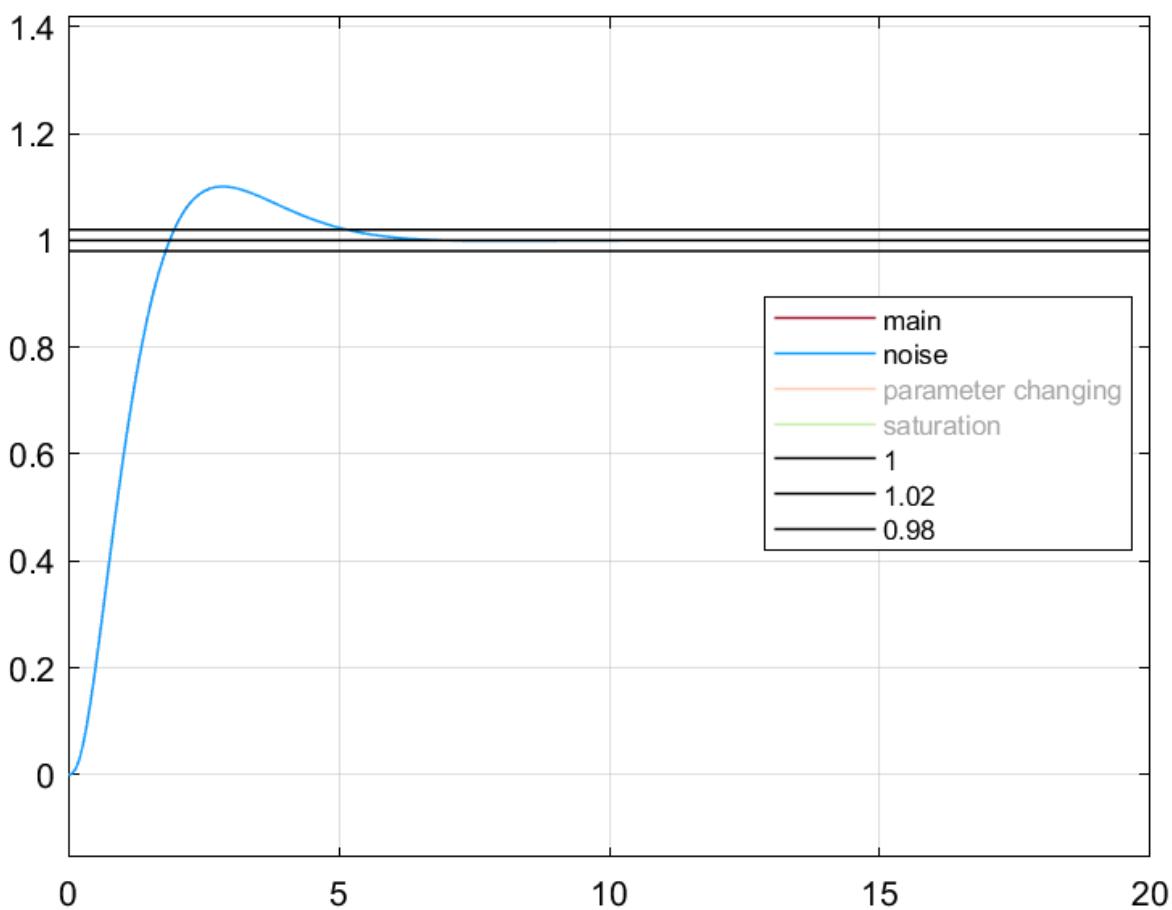
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطای شود ادامه با مقایسه ی سیگنال های خطای داریم:



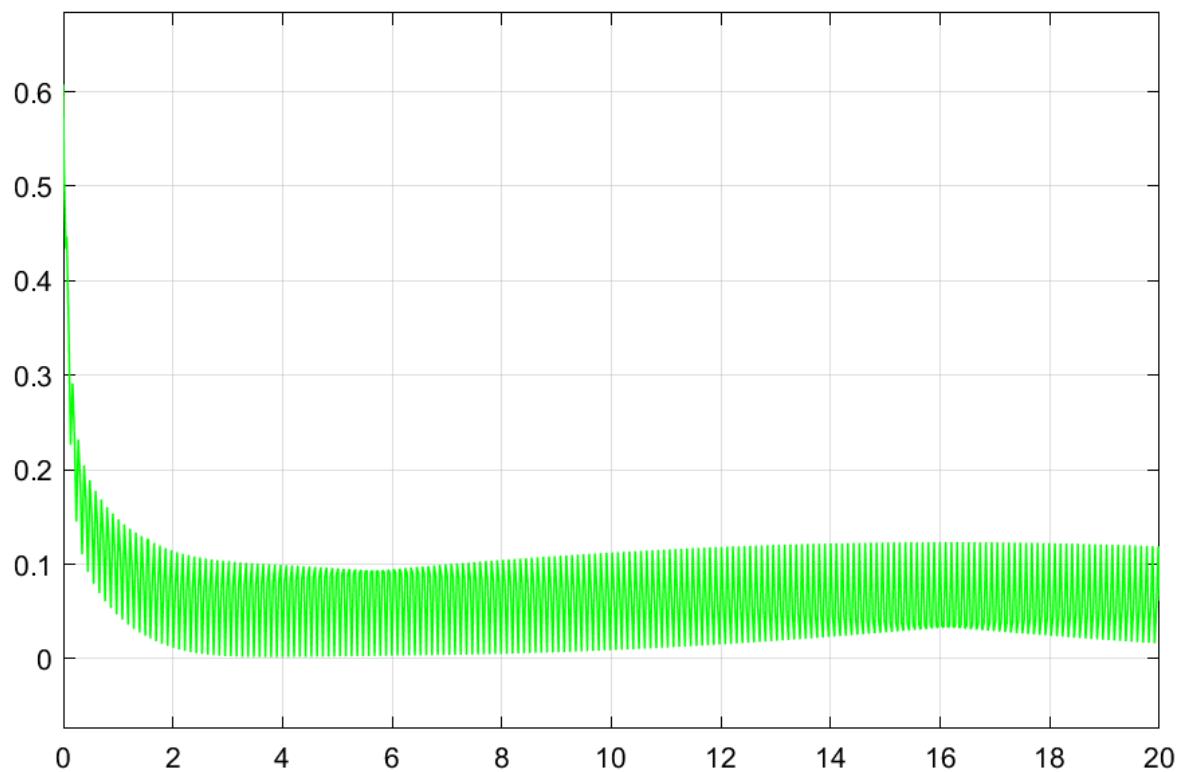
همانطور که قابل مشاهده است با مقایسه‌ی سیگنال‌های خطأ می‌توان متوجه شد که به دلیل وجود بلوک اشباع خطأ انباسته شده صفر نمی‌شود.

اثر نویز:

با مقایسه‌ی خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

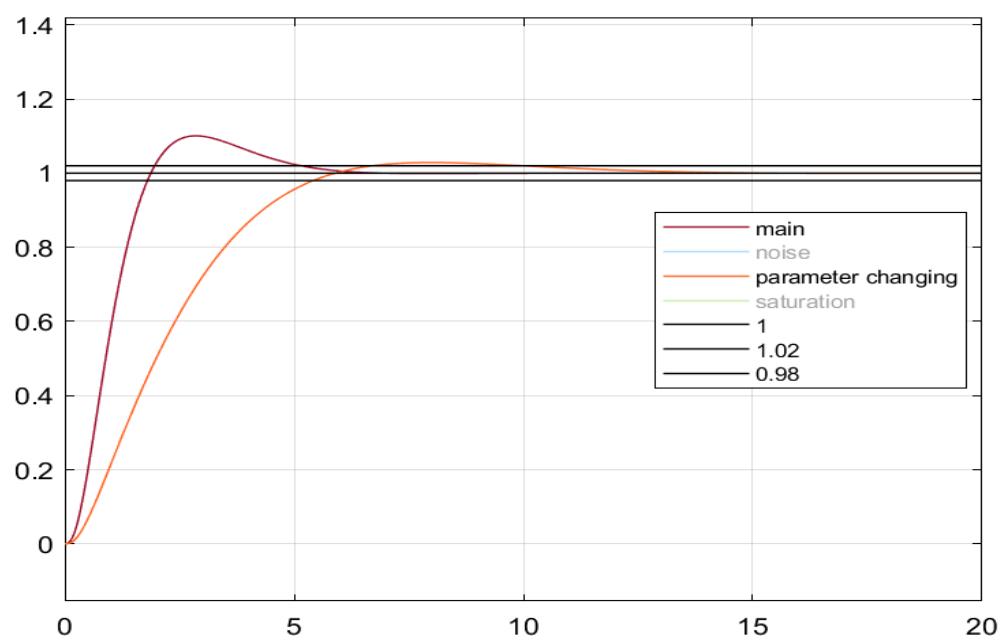


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می‌باشد که نویز فرکانس بالا می‌باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و می‌توان گفت که تاثیر بسیار کمی دارد. می‌دانیم که اثر نویز با بهره‌ی کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می‌گذرد (به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می‌توان از روی سیگنال فرمان که در ادامه نشان داده می‌شود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و کاهش ماکریم فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

CHR-REGULATORY 4-1 به روش

```
%CHR_REG OPEN LOOP
k=15.7452;
T=2.37;
T_d=0.5788;
alpha=T_d/T;
%PID CONTROLER PARAMERTERS
kp=0.95/(k*alpha);
Ti=T_d*2.38;
Td=T_d*0.42;
%KP=0.2471
%Ti=1.3775
%Td=0.2431
s=tf('s');
c=kp+((kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1)));
g=63/((s+0.5)*(s+2)*(s+4));
hold on
nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'y');
legend('G','GC','r=1');
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.4 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=44.94 deg
%GMcg=28.92 dB

t = out.chor(:, 1);
y = out.chor(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
```

```

idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

% calculating touching points
t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
    (line_value - y(idx)) ./ (y(idx+1) - y(idx));
y_intersect = line_value * ones(size(t_intersect)); % value of output in
touching points
%drawing lines
plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
1.2);

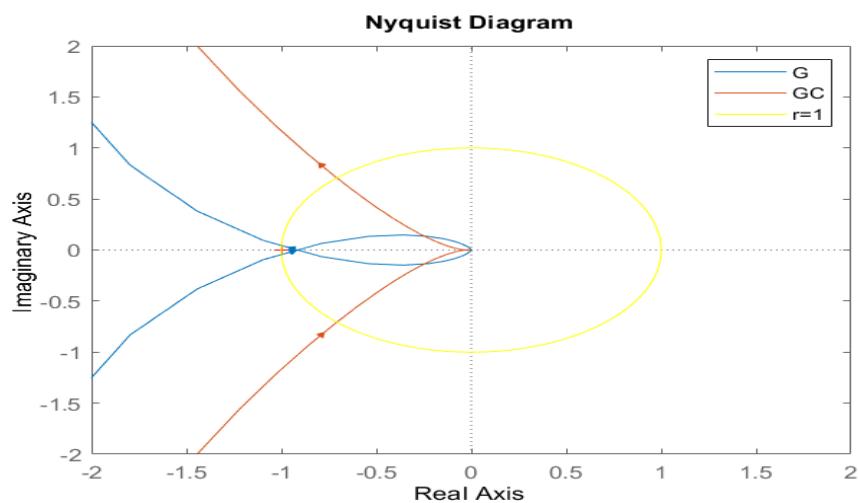
% showing touching points
plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
    'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
'UniformOutput', false)]};
legend(legend_entries{:, :});
grid on;
hold off;
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.69
%Ts%2=10.58
%overshoot%=55%
%d=0.22
%IAE=2.0213

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبه‌ی ضرایب PID می‌کنیم که با مقایسه

ی دایاگرام‌های نایکوپیست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به 45 درجه نزدیک شده است(44.94) و همچنین بهره سیستم جبران شده بر حسب dB خیلی بیشتر شده است(28.92).

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه ای موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه ای طراحی ها در یک جدول این موارد آورده می شود.
مشخصات گفته شده از روی این نمودار بدست آمده است:

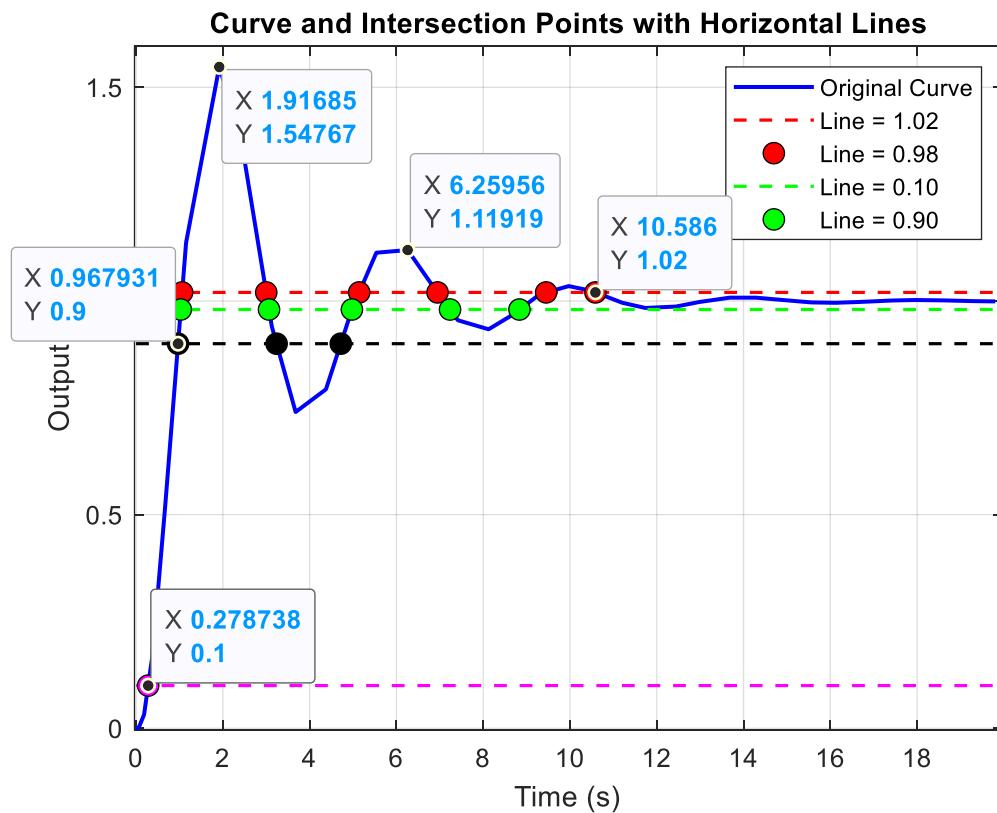
$$Tr=0.69$$

$$Ts\%2=10.58$$

$$overshoot\% = 55\%$$

$$d=0.22$$

$$IAE=2.0213$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ای طراحی ها این مشخصات در جدول انتهای

گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه

ی 20 اعمال شده است):

Ts5%=12.25

Ts10%=10.25

overshoot%=272%

d=0.234

IAE=7.44

```
t = out.chorl(:, 1);

y = out.chorl(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

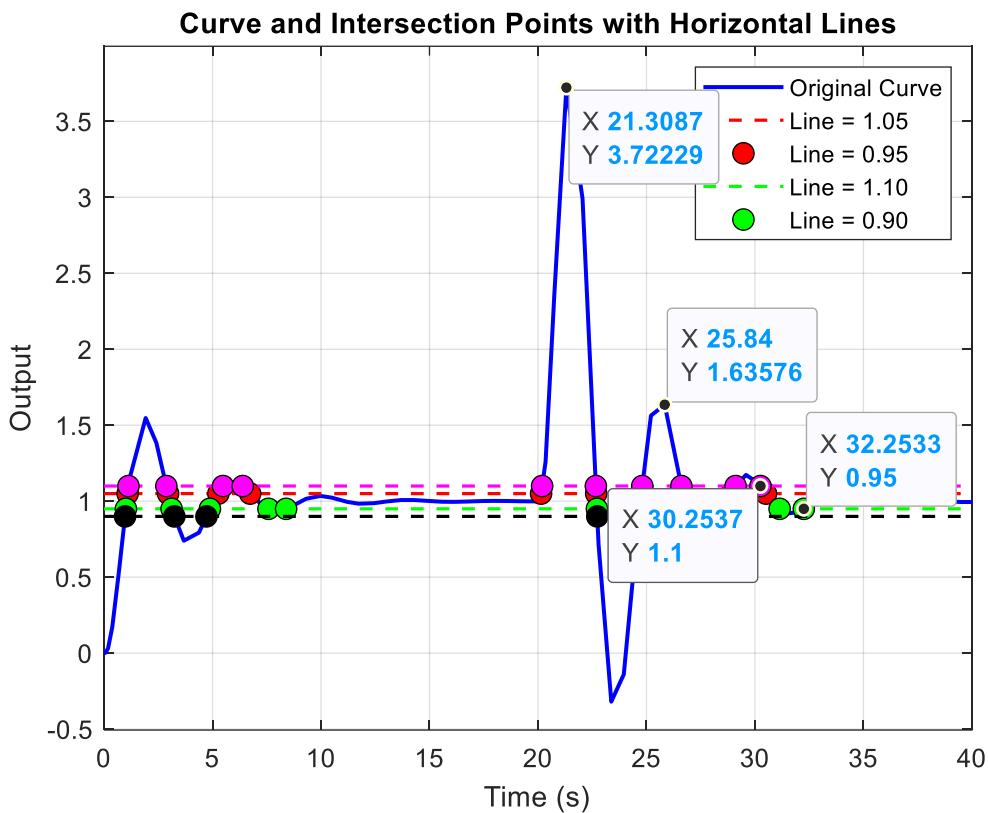
% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = [{['Original Curve']}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)];
legend(legend_entries{::});
grid on;
hold off;
```

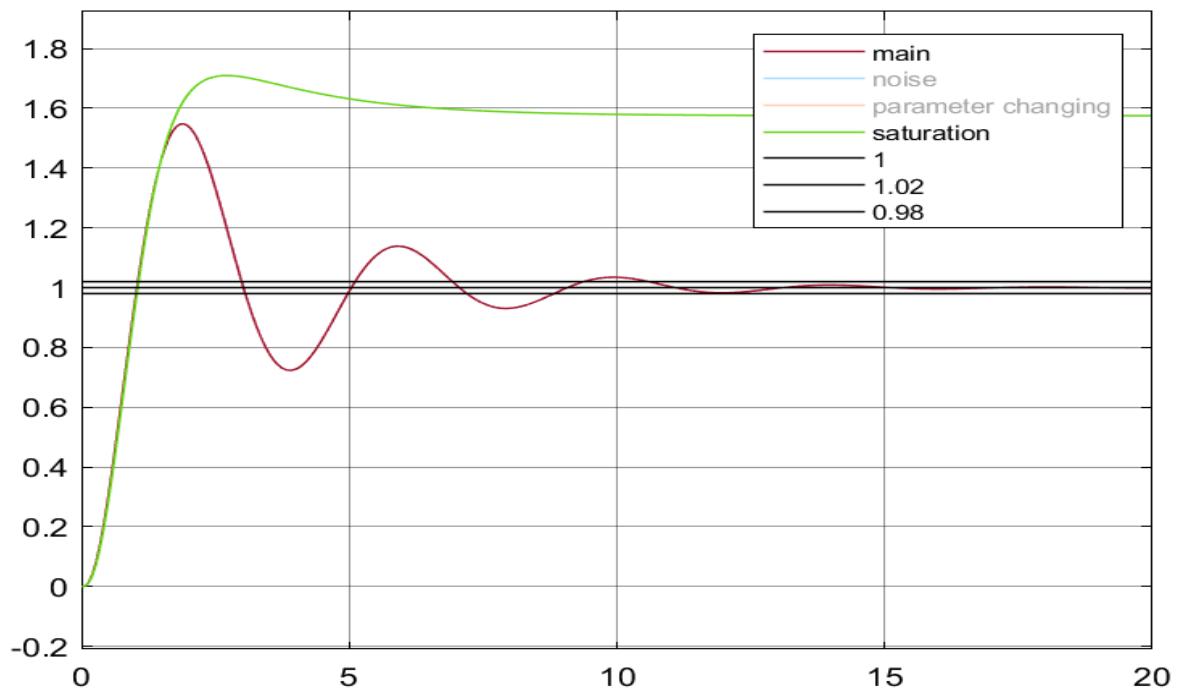
```
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-2.0213;
%Ts5%=12.25
%Ts10%=10.25
%overshoot%=272%
%d=0.234
%IAE=7.44
```



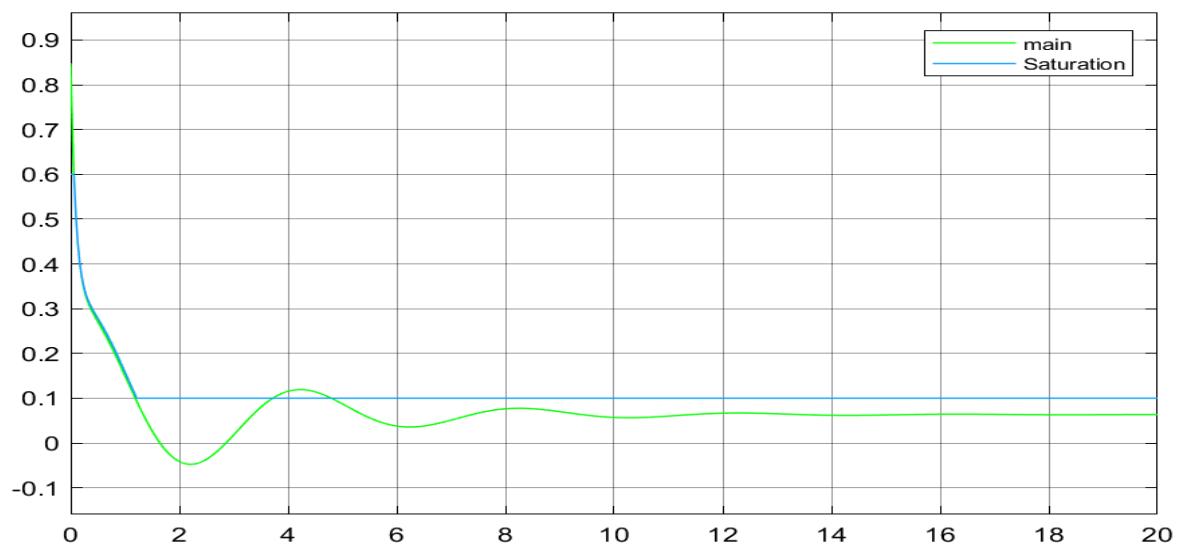
در ادامه ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می شود(نویز ، تغییر پارامترو...):

اشباع محرک:

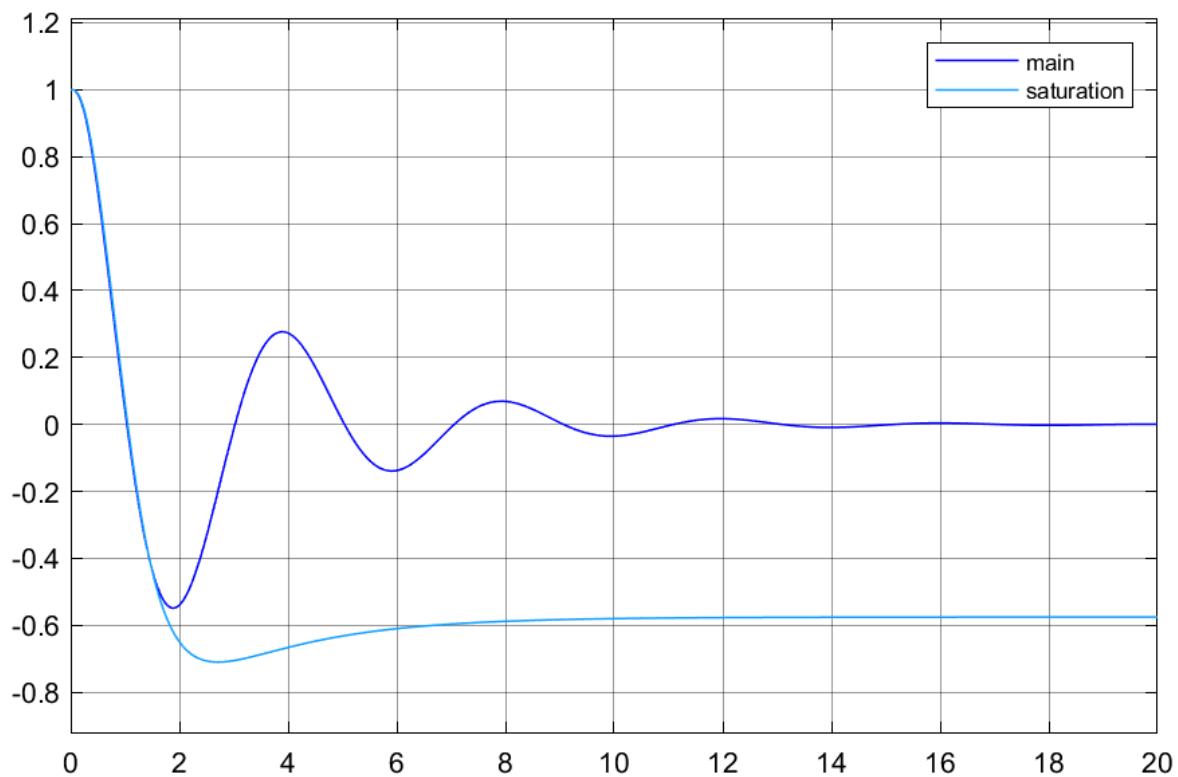
در این بخش به مقایسه ی خروجی ها بر اثر اشباع و خروجی اصلی می پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطا شده است(خطا صفر نمی شود). در ادامه به مقایسه ی سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



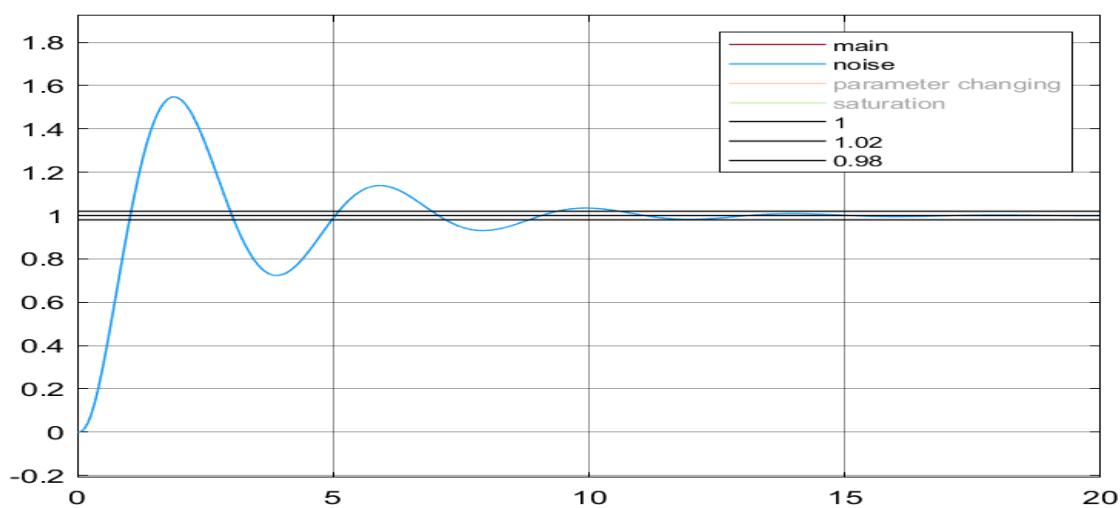
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطأ می شود ادامه با مقایسه ی سیگنال های خطأ داریم:



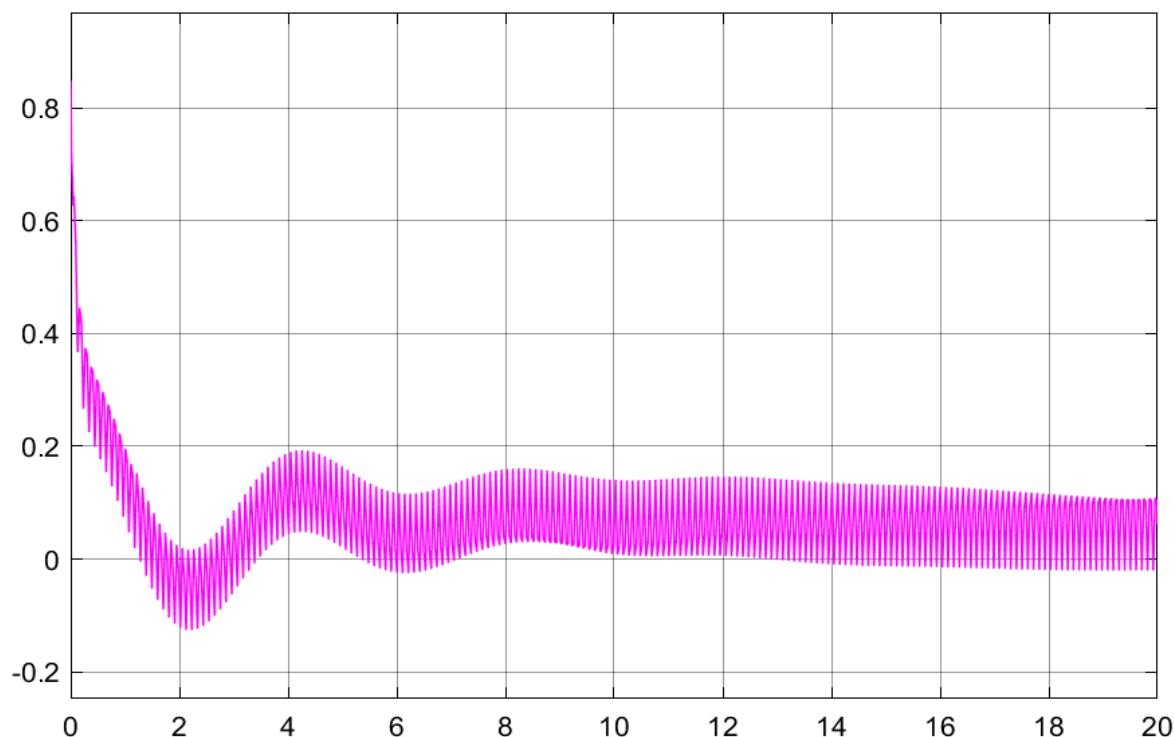
همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای انباشته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

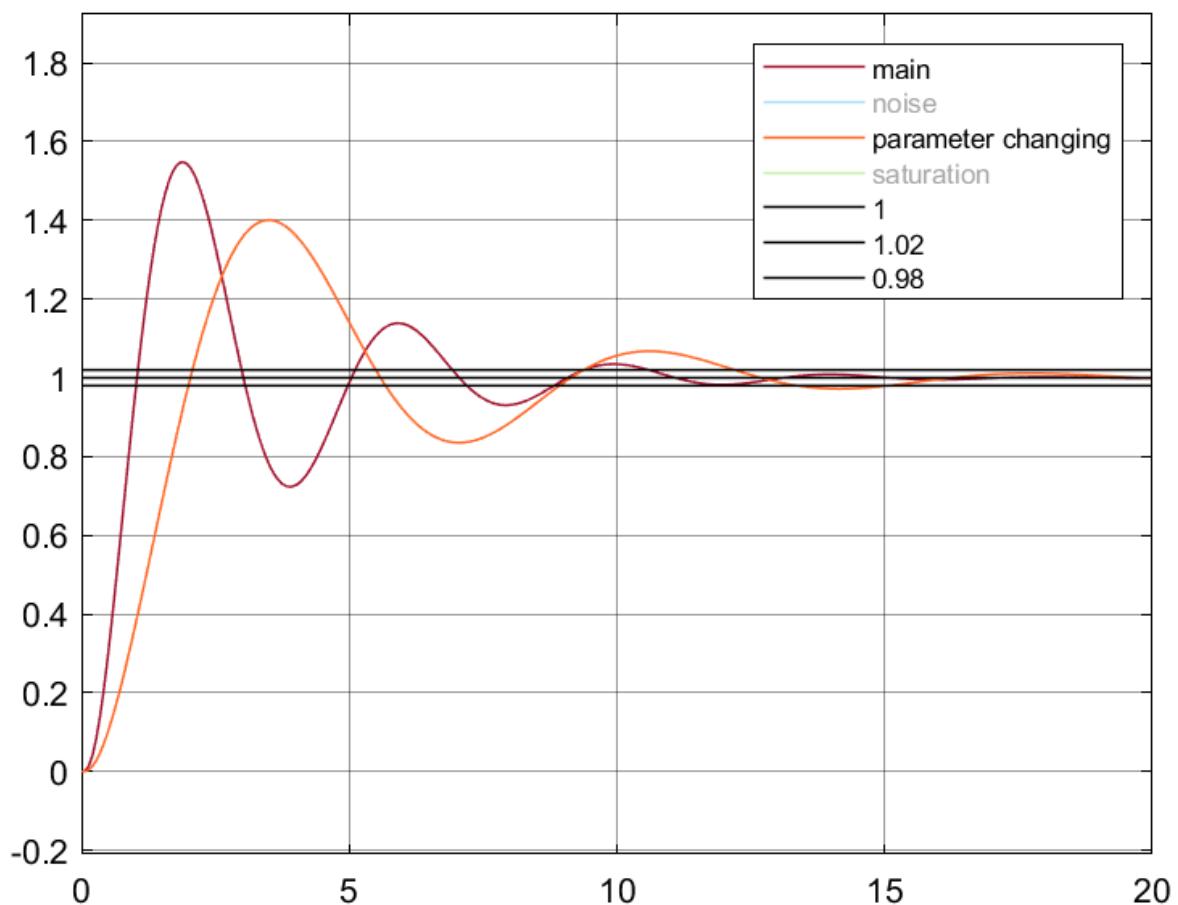


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و کاهش ماکزیمم فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

بخش دوم

طراحی PID به صورت حلقه بسته

ZN به روش 1-2

```

s=tf('s');
g=63/((s+0.5)*(s+2)*(s+4));
%sisotool(g)
ku=1.085;
Tu=2.29;
%PID controller
kp=0.6*ku;
Ti=Tu/2;
Td=Tu/8;
%KP=0.6510
%Ti=1.1450
%Td=0.2863

c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
hold on
nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'y');
legend('G','GC','r=1');
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=35.1161 deg
%GMcg=19.5548 dB


t = out.zpc(:, 1);
y = out.zpc(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

```

```

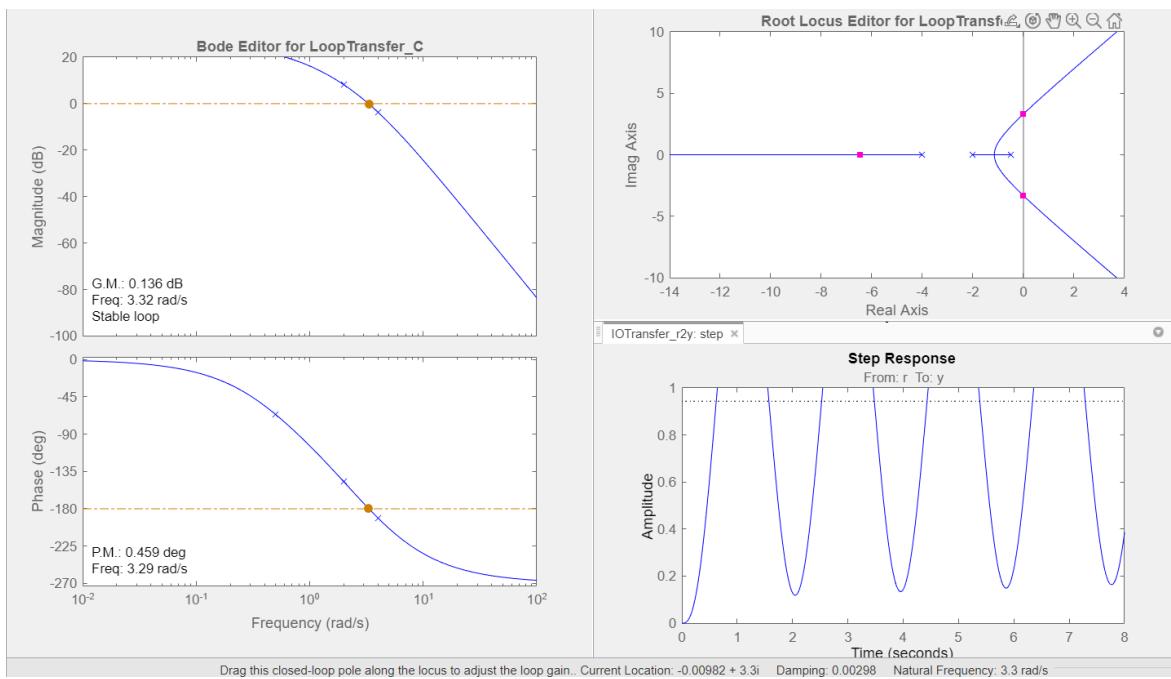
% showing touching points
plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
      'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = [ {'Original Curve'}, ...
                    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
                    'UniformOutput', false)];
legend(legend_entries{:});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.40
%Ts%2=4
%overshoot%=45.9%
%d=0.076
%IAE=0.8461

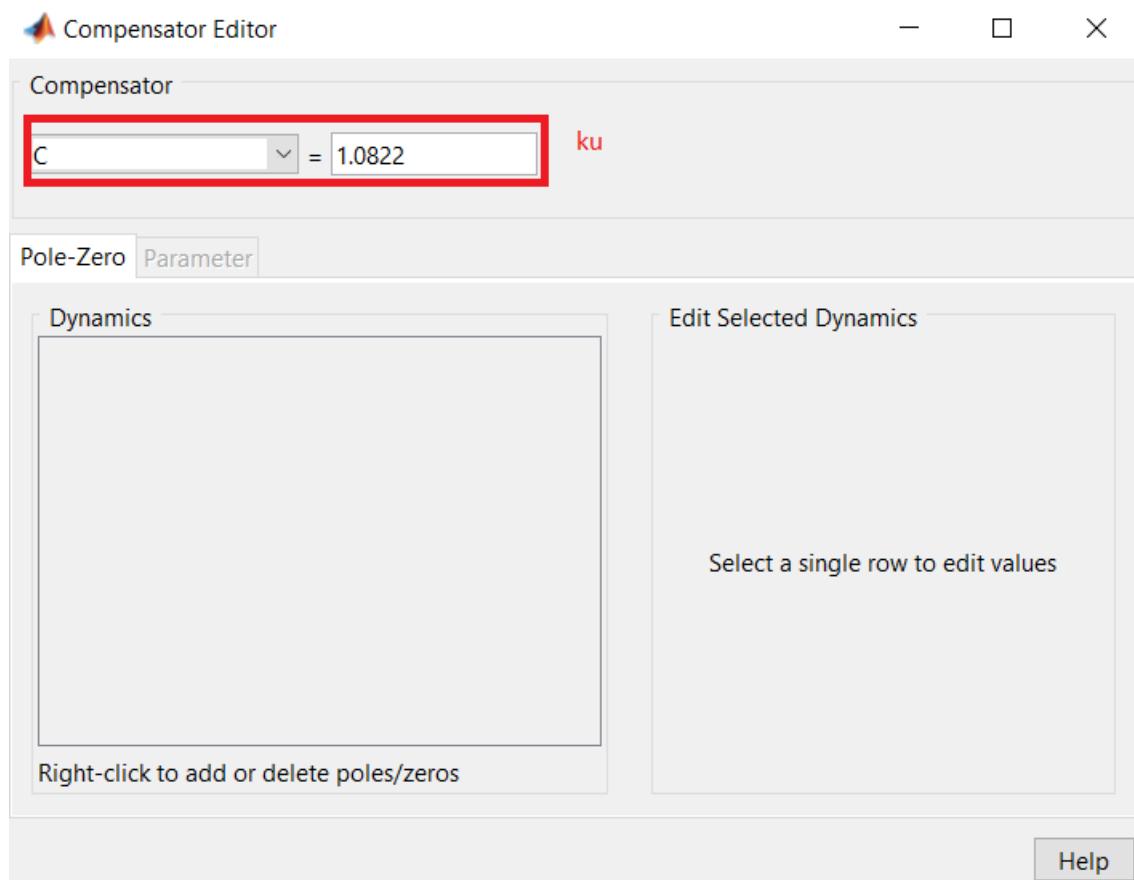
```

در این روش برای بدست آوردن مقادیر ضرایب PID نیازی به پارامتر های مدل نیست در ابتدای کار با استفاده از یک کنترل کننده KP بهره را آنچنان تغییر میدیم که پاسخ خروجی نوسان پایدار شود که با ضریب بهره ای به دست آمده که آن را ku قرار می دهیم و دوره تناوب نوسان پایدار را Tu می نامیم و در ادامه ضرایب را از روی این مقادیر بدست می آوریم.(استفاده کردن از sisotool)

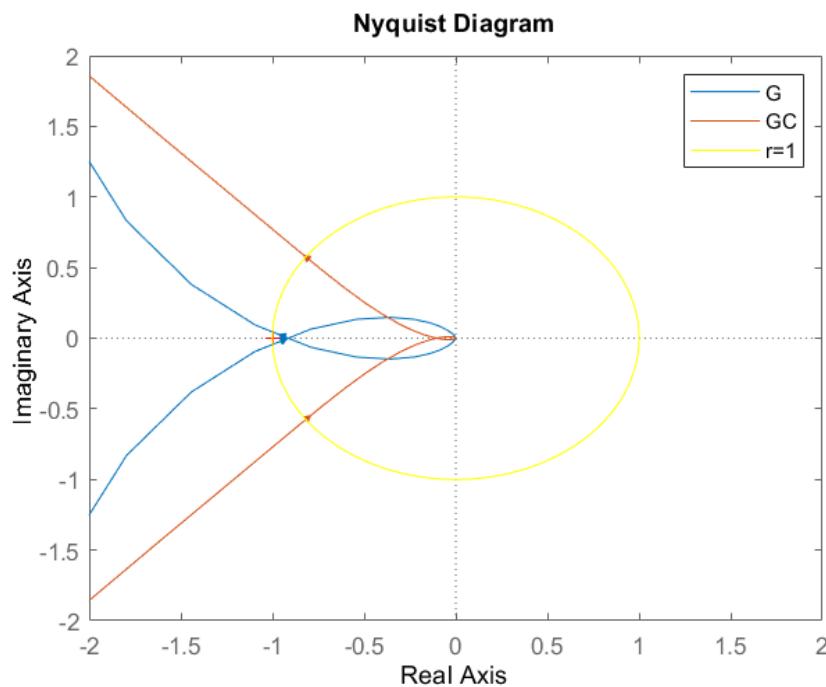
قرار دادن قطب ها روی محور موهومی برای دست یابی به نوسان پایدار:



بدست آوردن T_u اروی پاسخ پله و بدست آوردن ضریب ku



در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبهٔ ضرایب PID می‌کنیم که با مقایسهٔ دایاگرام‌های نایکوپیسیت داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به 45 درجه نزدیک شده است(35.1) و همچنین بهره سیستم جبران شده بر حسب dB خیلی بیشتر شده است(19.6).

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همهٔ موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همهٔ طراحی‌ها در یک جدول این موارد آورده می‌شود.
مشخصات گفته شده از روی این نمودار بدست آمده است:

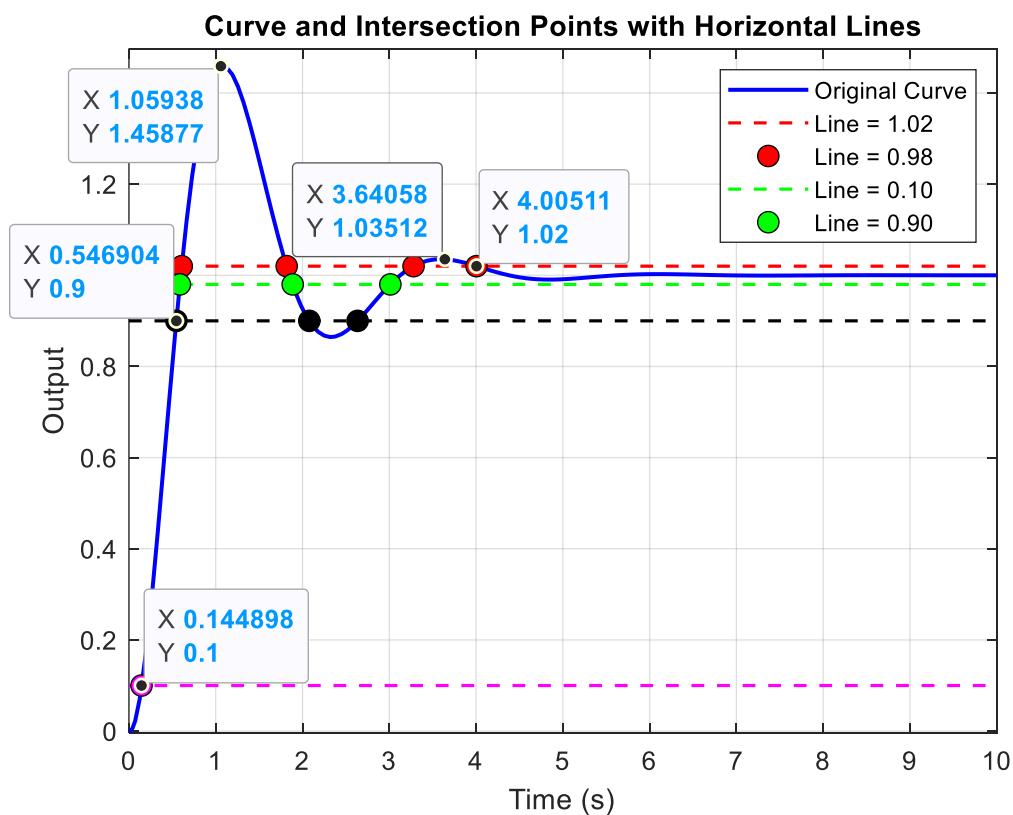
$$Tr=0.40$$

$$Ts\%2=4$$

$$overshoot\% = 45.9\%$$

$$d=0.076$$

$$IAE=0.8461$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ی طراحی ها این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ۷ اعمال شده است):

$$T_{s5\%}=4.1$$

$$T_{s10\%}=3.09$$

$$\text{overshoot\%}=139\%$$

$$d=0.046$$

$$\text{IAE}=1.7800$$

```
t = out.zpc1(:, 1);
y = out.zpc1(:, 2);
```

```

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

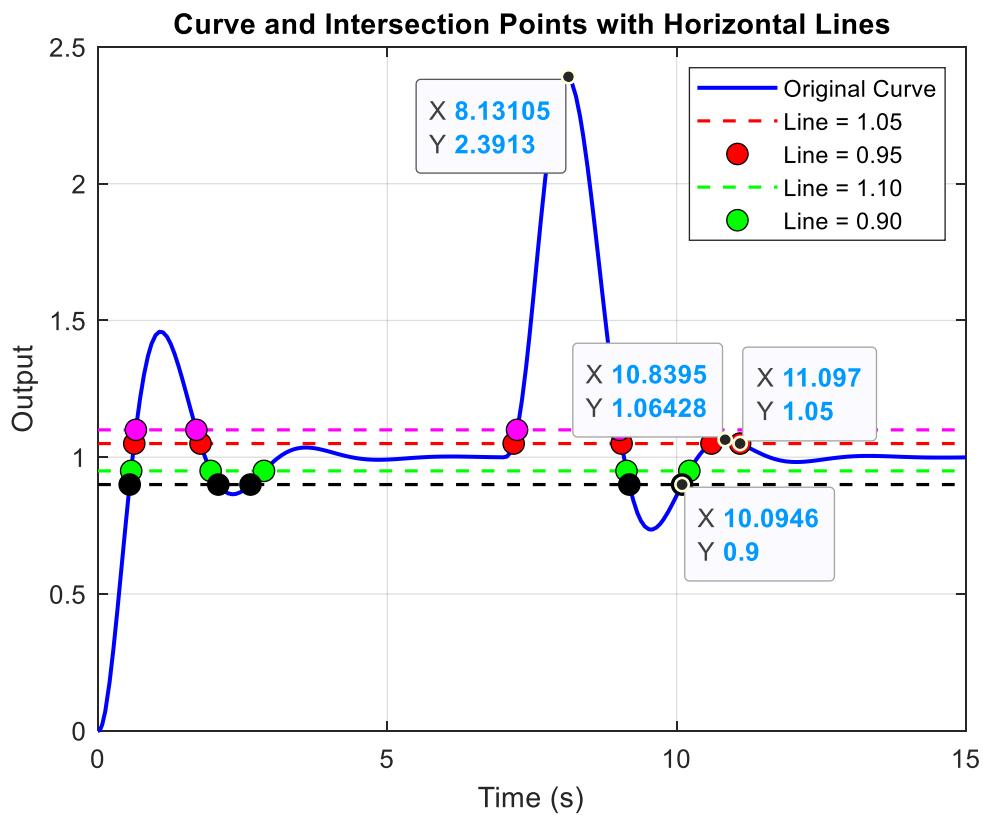
% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-0.8461;
%Ts5%=4.1
%Ts10%=3.09
%overshoot%=139%
%d=0.046
%IAE=1.7800

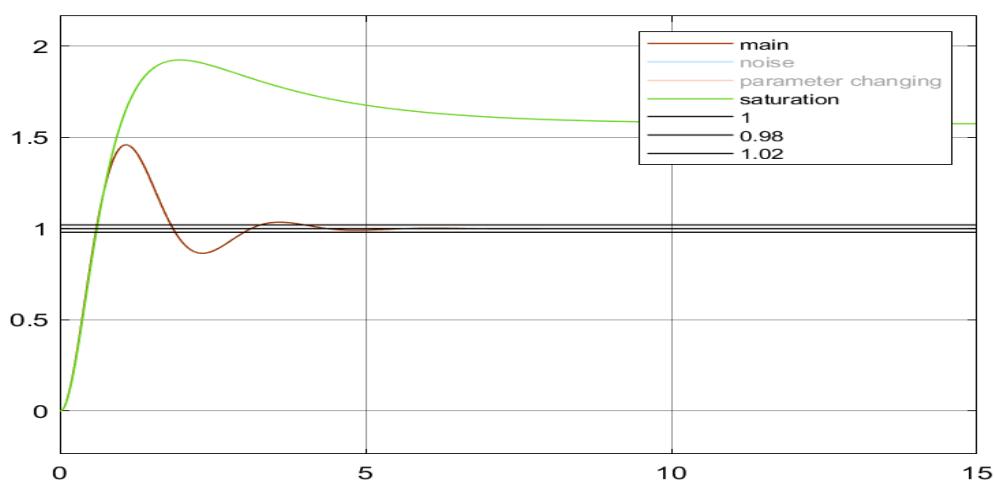
```



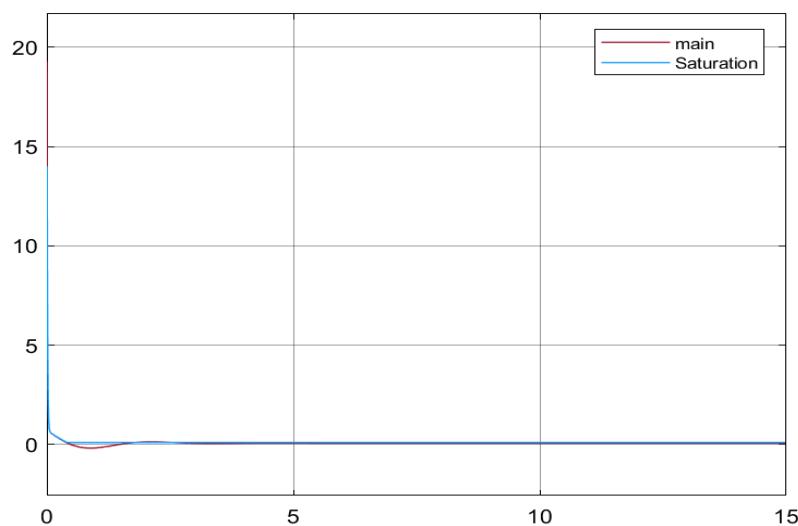
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامترو...):

اشباع محرک:

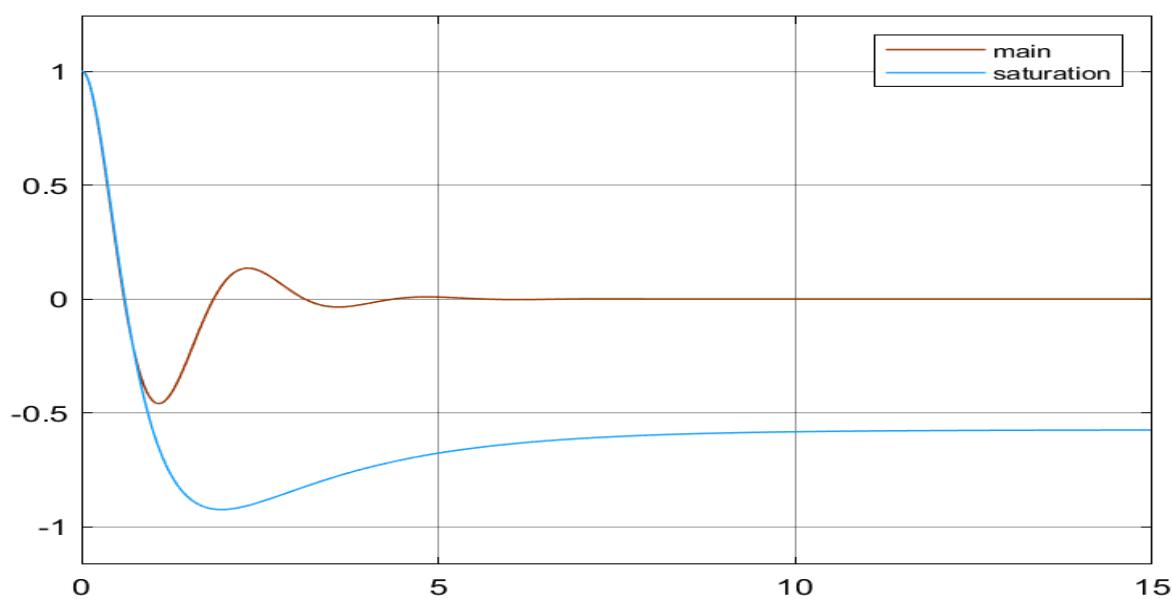
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطای خطا نمی شود). در ادامه به مقایسه ای سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



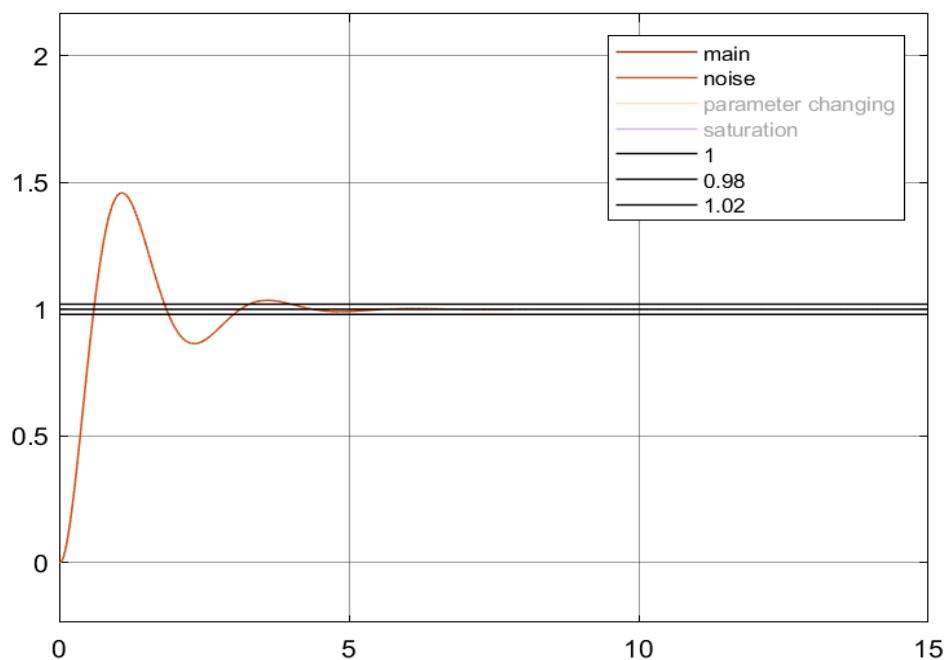
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطای خطا می شود ادامه با مقایسه ای سیگنال های خطا داریم:



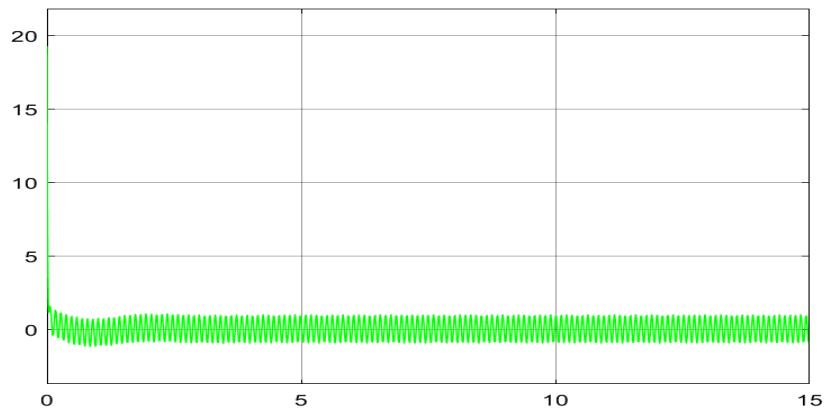
همانطور که قابل مشاهده است با مقایسه‌ی سیگنال‌های خطأ می‌توان متوجه شد که به دلیل وجود بلوک اشباع خطأ انباسته شده صفر نمی‌شود.

اثر نویز:

با مقایسه‌ی خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

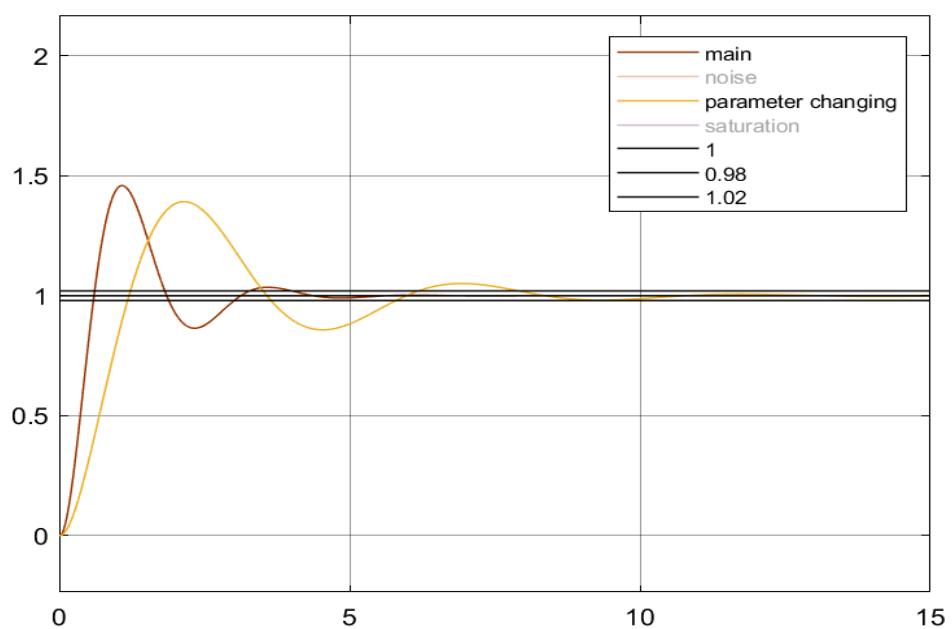


همانطور که مشخص است هر دو روی یکدیگر افتاده‌اند و این مورد به دلیل این می‌باشد که نویز فرکانس بالا می‌باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و می‌توان گفت که تاثیر بسیار کمی دارد. می‌دانیم که اثر نویز با بهره‌ی کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می‌گذرد (به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می‌توان از روی سیگنال فرمان که در ادامه نشان داده می‌شود متوجه شد:



اثر عدم قطعیت:

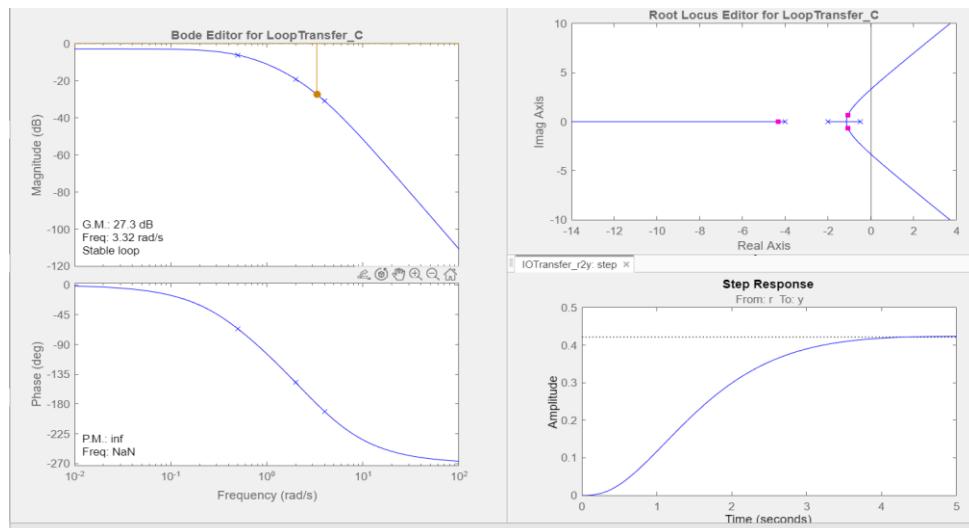
در ادامه با مقایسه خروجی ها داریم:



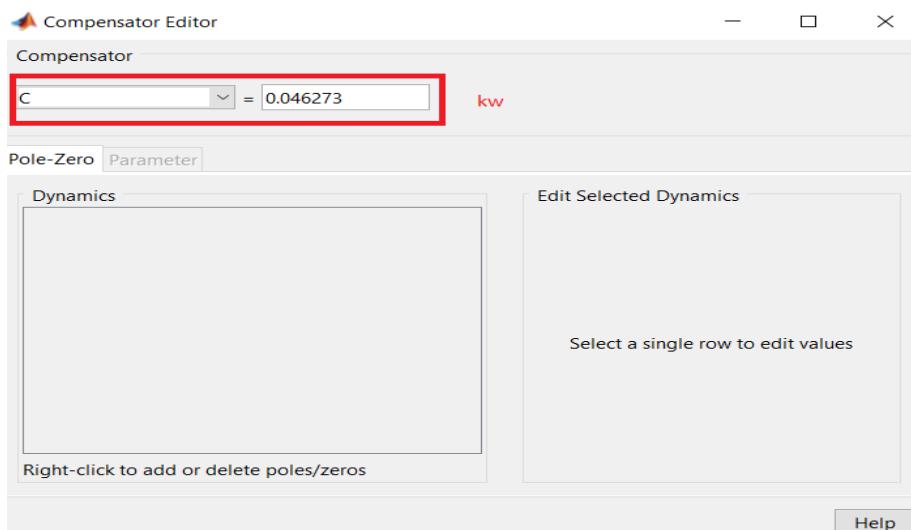
تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و کاهش ماکزیمم فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

2-2 به روش ZN همراه با ورودی مرجع وزن دار

در این بخش علاوه بر مراحل طی شده در بخش قبل برای بدست آوردن مقادیر k_u و T_u برای بدست آوردن وزن جهت ورودی مرجع وزن دار باید k_w را این گونه بدست آوریم که ضریب بهره ای که باعث می شود پاسخ پله میرایی بحرانی شود و داریم :



و برای ضریب k_w داریم:



در ادامه روند را مانند روش های قبل طی می کنیم.

```
s=tf('s');
g=63/((s+0.5)*(s+2)*(s+4));
sisotool(g)
ku=1.085;
```

```

Tu=2.29;
%PID controller
kp=0.6*ku;
Ti=Tu/2;
Td=Tu/8;
%KP=0.6510
%Ti=1.1450
%Td=0.2863
kw=0.045;
a=kw/kp;
%a=0.0691
Gc=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
Gff=kp*a+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
G=(Gff*Gp)/(1+(Gp*Gc)-(Gff*Gp));
hold on
nyquist(Gp)
nyquist(G)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'y');
legend('Gp','Geq','r=1');
[GMg_1,PMg,~,~]=margin(Gp);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(G);
GMcg=20*log10(GMcg_1);
%PMcg=78 deg
%GMcg=26.51 dB

t = out.zpc(:, 1);
y = out.zpc(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    %touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points

```

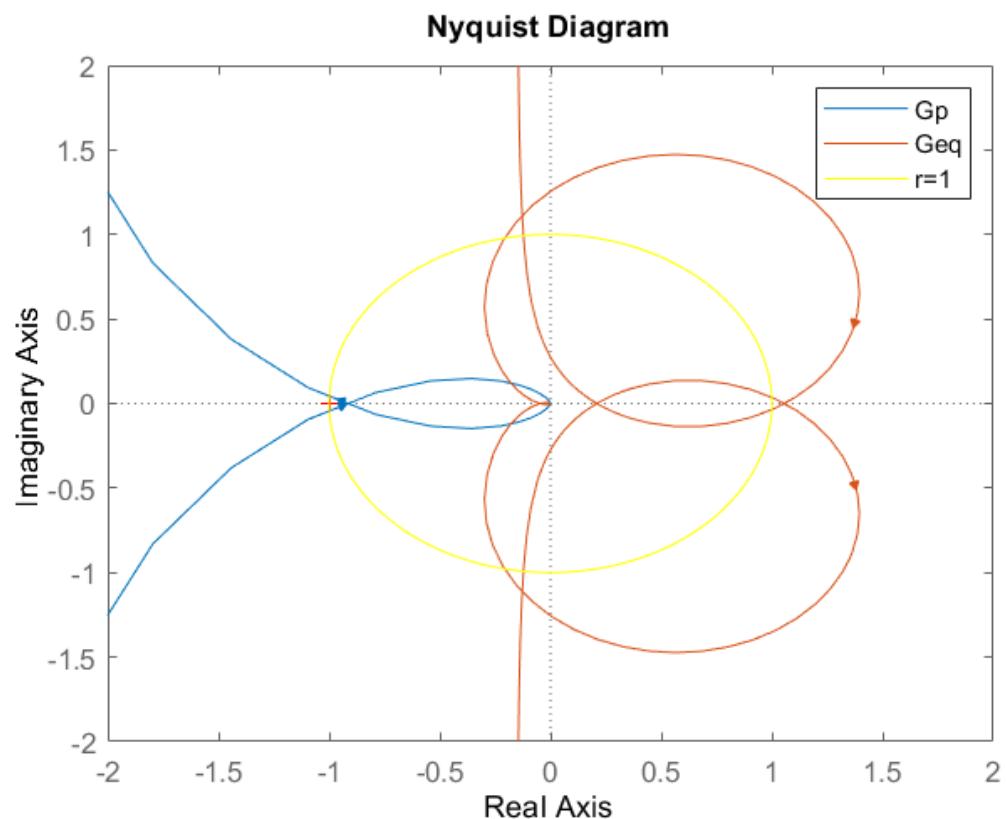
```

plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
      'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
'UniformOutput', false)]};
legend(legend_entries{:, :});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=2.225
%Ts%2=2.78
%overshoot%==
%d=-
%IAE=1.1838

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبهٔ ضرایب PID می‌کنیم که با مقایسهٔ دایاگرام‌های نایکوپیست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به 90 درجه نزدیک شده است(78) و همچنین بهره سیستم جبران شده بر حسب dB خیلی بیشتر شده است(26.5).

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه ای موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه ای طراحی ها در یک جدول این موارد آورده می شود. مشخصات گفته شده از روی این نمودار بدست آمده است:

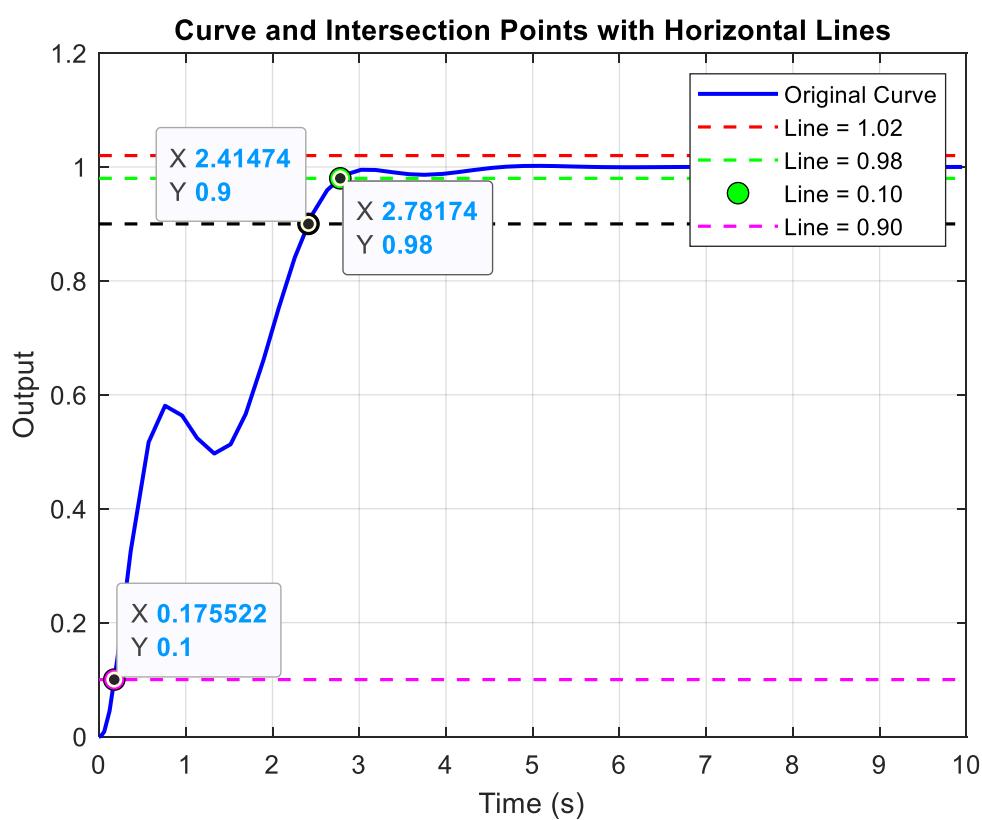
$$Tr=2.225$$

$$Ts\%2=2.78$$

$$overshoot\%=-$$

$$d=-$$

$$IAE=1.1838$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش (اغتشاش در ثانیه ۵ اعمال شده است):

Ts5%=2.43

Ts10%=2.32

overshoot%=143.5%

d=-

IAE=1.7669

```
t = out.zcwl(:, 1);

y = out.zcwl(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
1.2);

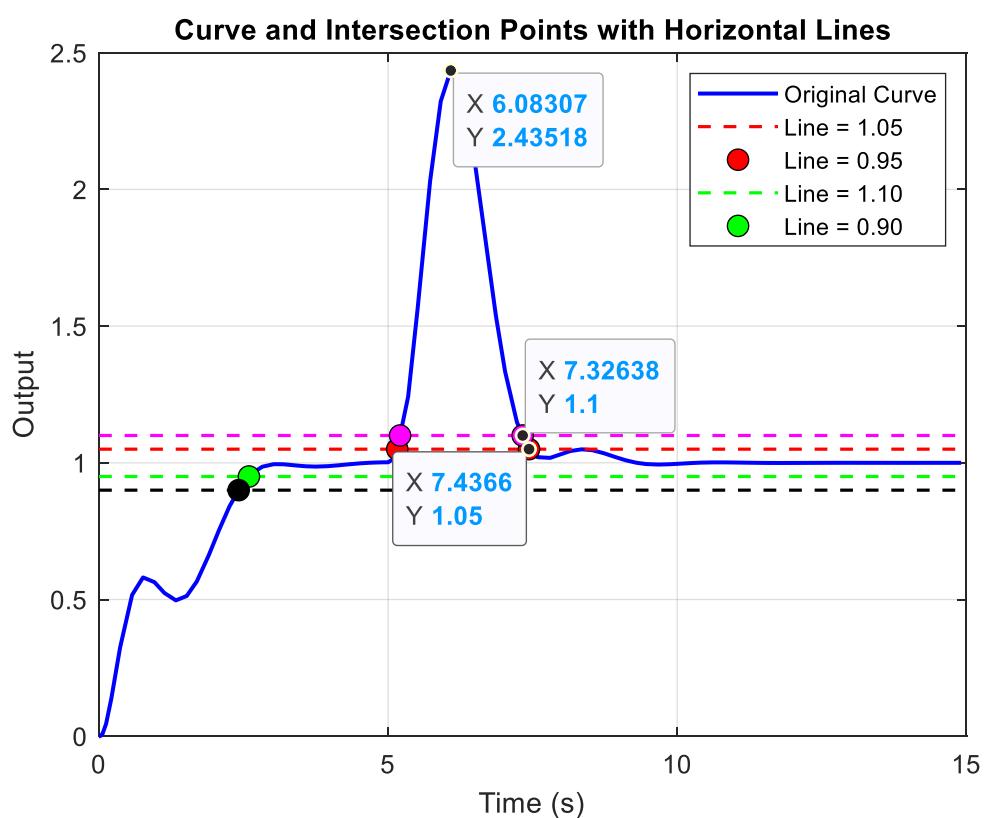
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
```

```

title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
'UniformOutput', false)]};
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-1.1838;
%Ts5%=2.43
%Ts10%=2.32
%overshoot%=143.5%
%d=-
%IAE=1.7669

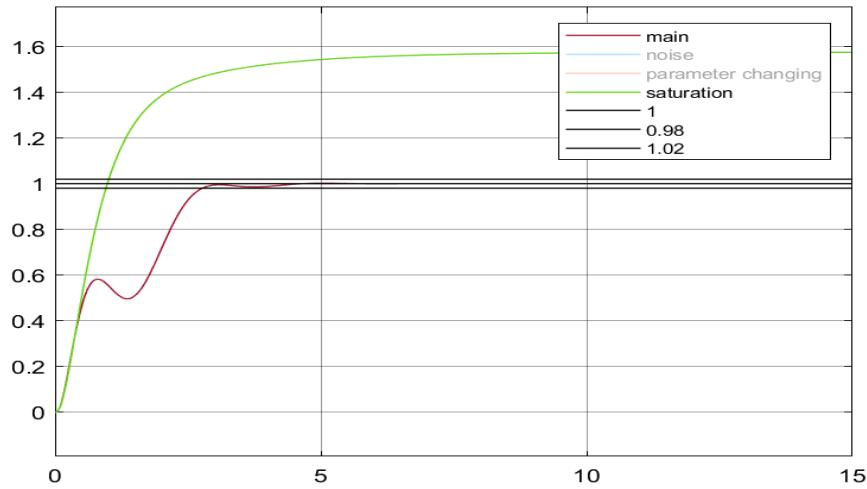
```



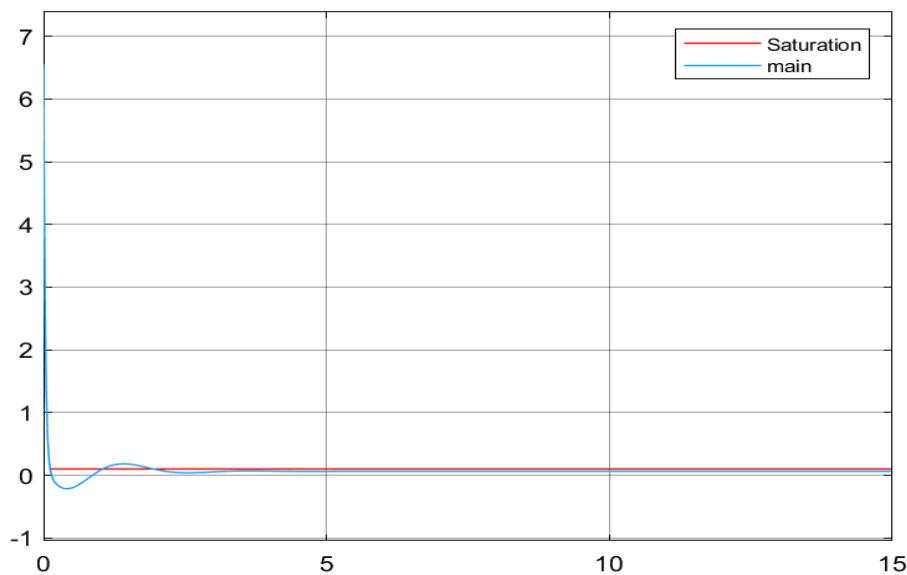
در ادامه‌ی این بخش خرچی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامتر و...):

اشباع محرک:

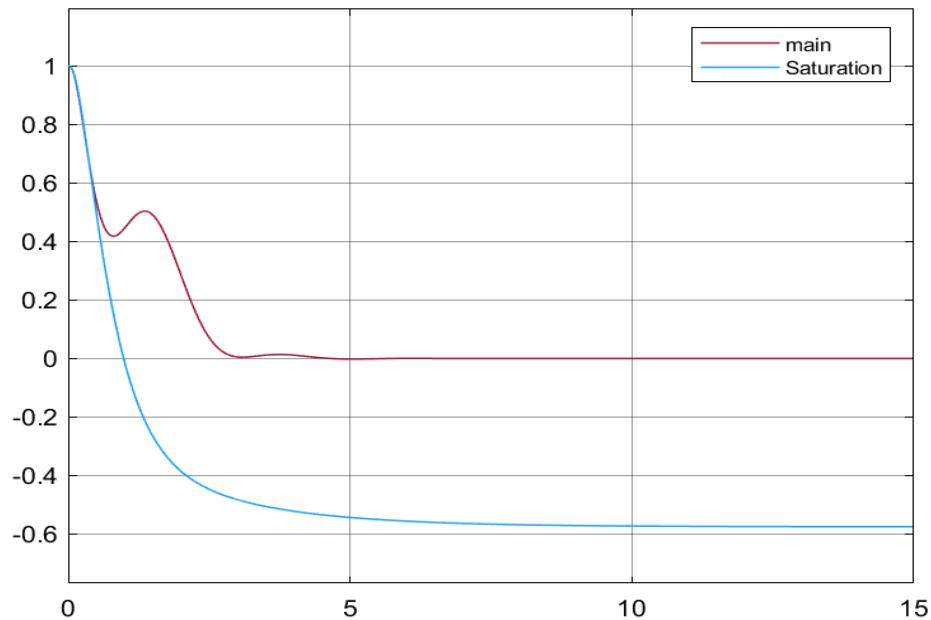
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



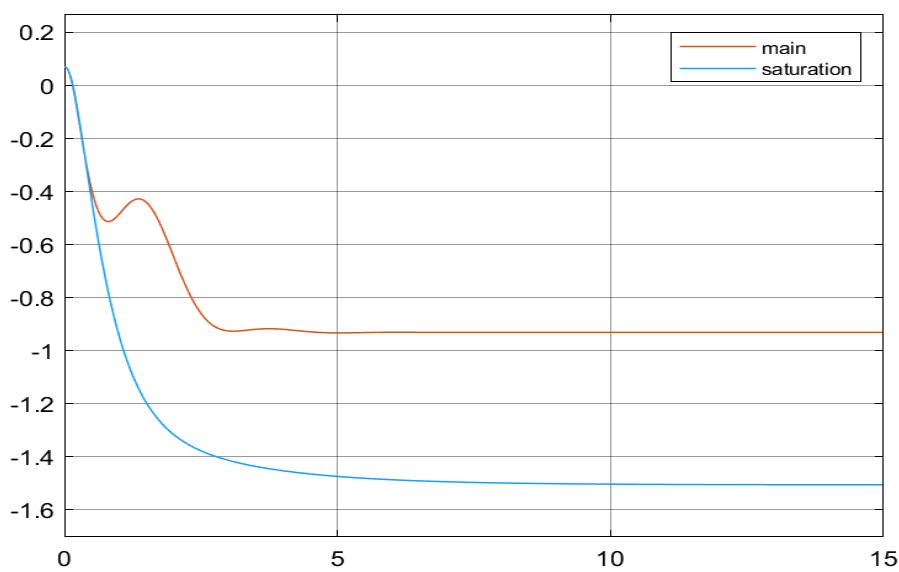
همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطأ شده است(خطأ صفر نمی شود)، در ادامه به مقایسه ای سیگنال های فرمان میپردازیم که نشان دهنده ای نوع عملکرد بلوک اشباع می باشد.



همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطأ می شود ادامه با مقایسه ای سیگنال های خطأ داریم:
سیگنال خطای اعمالی به DI :



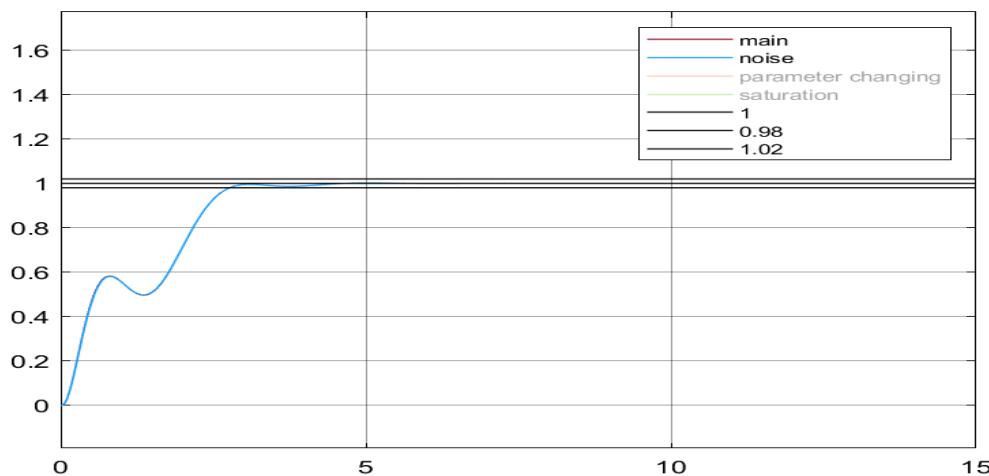
سیگنال خطای اعمالی به P :



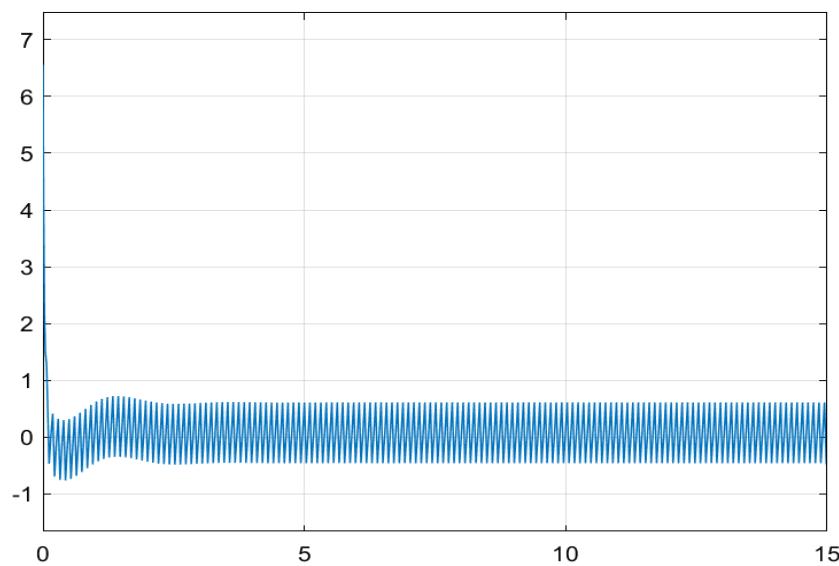
همانطور که قابل مشاهده است با مقایسه سیگنال های خطا میتوان متوجه شد که به دلیل وجود بلوک اشباع خطا انباشته شده صفر نمی شود. (البته خطای اعمالی به ترم P نیز به دلیل وجود ترم وزن دار نیز صفر نمی شود).

اثر نویز:

با مقایسه خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

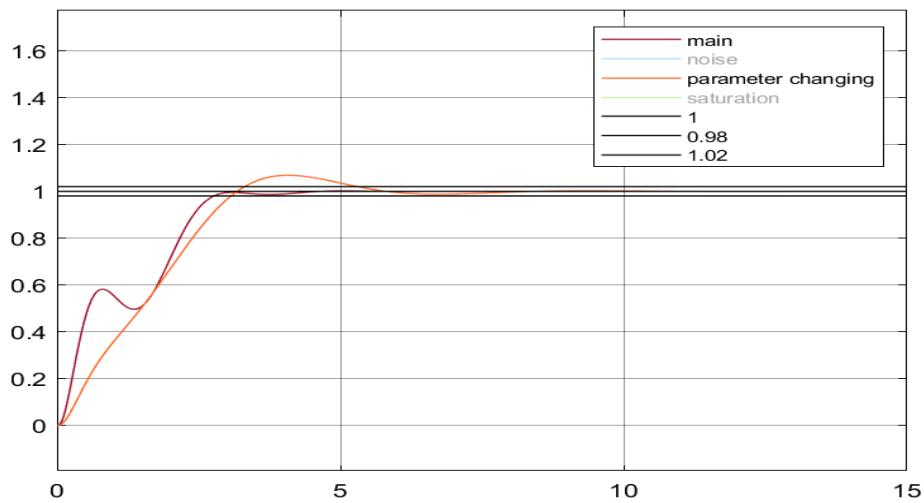


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذارد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:

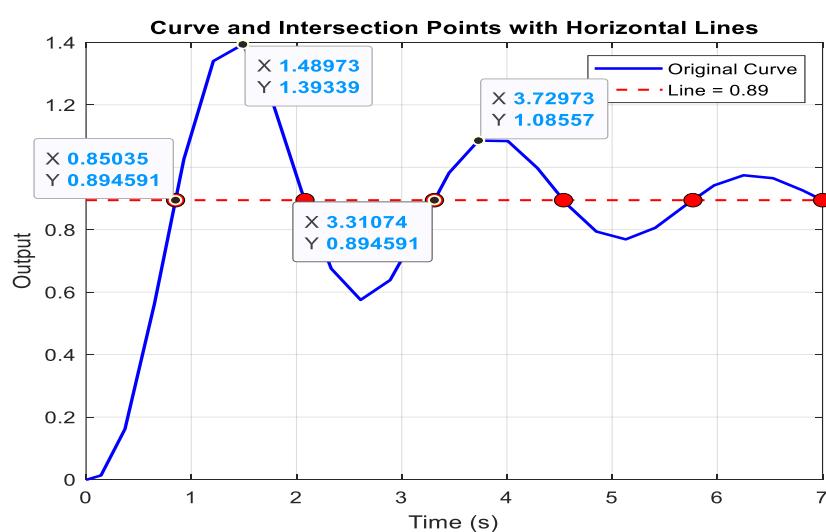


تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و کاهش ماکزیمم فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

3-2 به روش ZN نوسان میرا

در این روش ضریب kp را به گونه ای تغییر می دهیم که پاسخ پله با نسبت افت 0.25 به دست آید. با بدست آوردن این ضریب که نام آن را $Kdmp$ و همچنین دوره تناوب این پاسخ را $Tdmp$ گذاشته ایم.

که با $Kdmp=0.545$ داریم:



در ادامه روند رو مثل قبل طی می کنیم:

کد مربوطه:

```
%for finding kdmp(d=0.25) and Tdmp
n=1;
t = out.znn(:, 1);
y = out.znn(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

k=mean(y(end-n+1:end));

horizontal_lines = [k];
colors = ['r'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:});
grid on;
hold off;
kdmp=0.545;
Tdmp=2.44;
%PID controller parameters
kp=1.1*kdmp;
Ti=Tdmp/3.6;
Td=Tdmp/9;
%kp=0.5995
%Ti=0.6778
%Td=0.2711
s=tf('s');
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
g=63/((s+0.5)*(s+2)*(s+4));
hold on
```

```

nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'r')
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=22.64 deg
%GMcg=20.39 dB


t = out.znn(:, 1);
y = out.znn(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

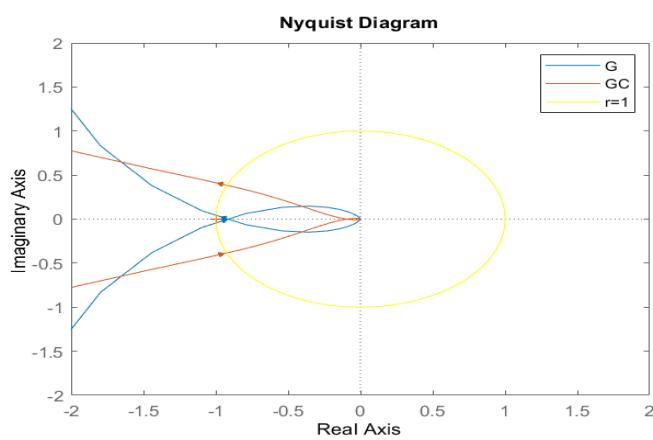
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.4

```

```
%Ts%2=3.53
%overshoot%=43.18%
%d=0.14
%IAE=0.8309
```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبهٔ ضرایب PID می‌کنیم که با مقایسهٔ دایاگرام‌های نایکوپیست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده بیشتر شده و به حدود 22.64 درجه نزدیک شده استو همچنین بهره سیستم جبران شده برحسب dB خیلی بیشتر شده است(20.39).

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همهٔ موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همهٔ طراحی‌ها در یک جدول این موارد آورده می‌شود.

مشخصات گفته شده از روی این نمودار بدست آمده است:

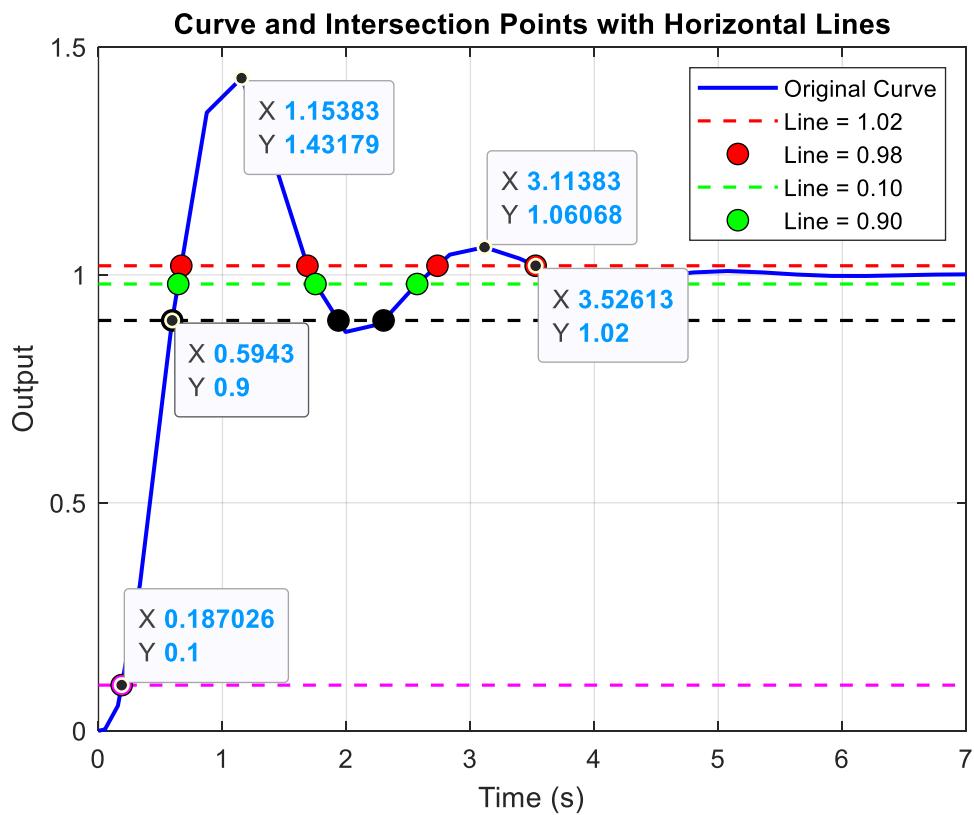
$Tr=0.4$

$Ts\%2=3.53$

$overshoot\% = 43.18\%$

$d=0.14$

$IAE=0.8309$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ای طراحی ها این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ۷ اعمال شده است):

$$T_{s5\%}=4.22$$

$$T_{s10\%}=3.95$$

$$\text{overshoot\%}=157\%$$

$$d=-$$

$$\text{IAE}=2.4369$$

```
t = out.znn1(:, 1);
y = out.znn1(:, 2);
```

```

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

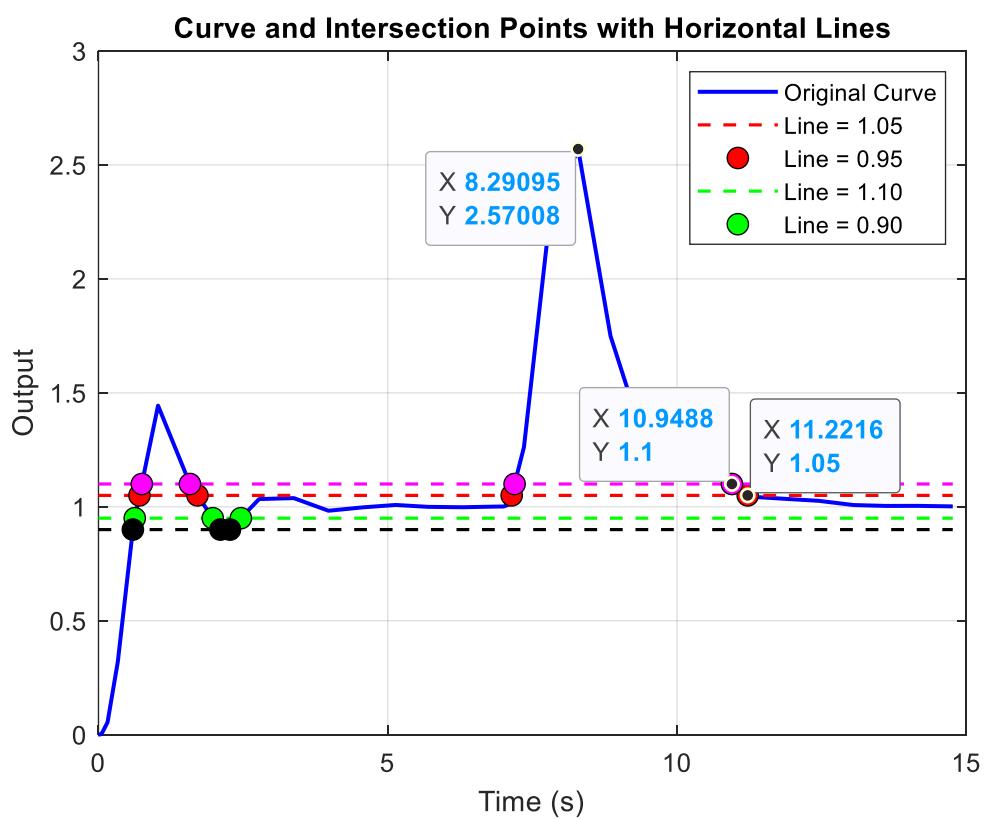
% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-0.8309;
%Ts5%=4.22
%Ts10%=3.95
%overshoot%=157%
%d=-
%IAE=2.4369

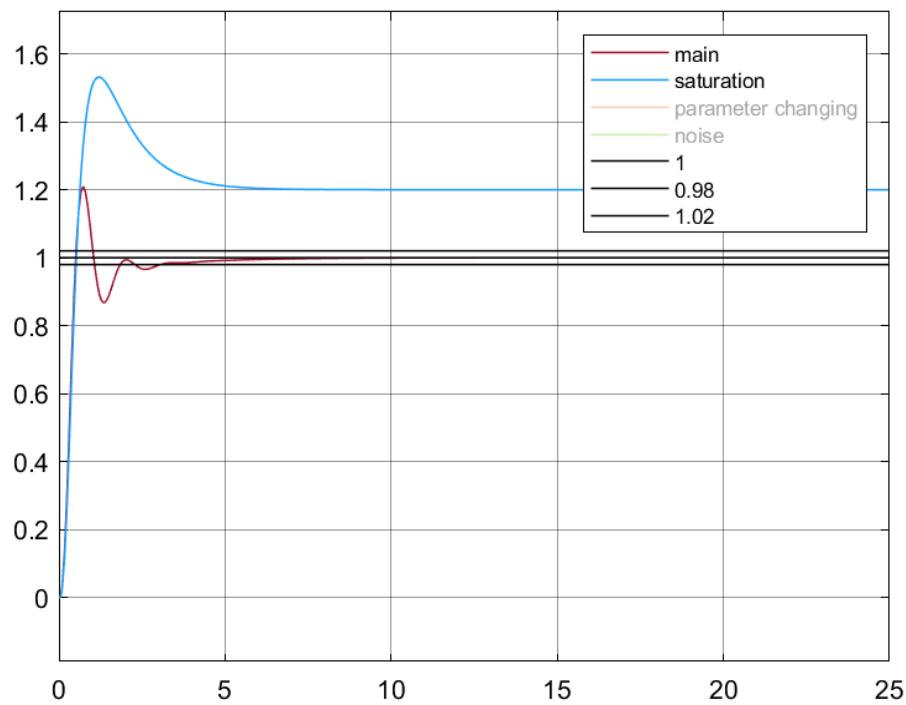
```



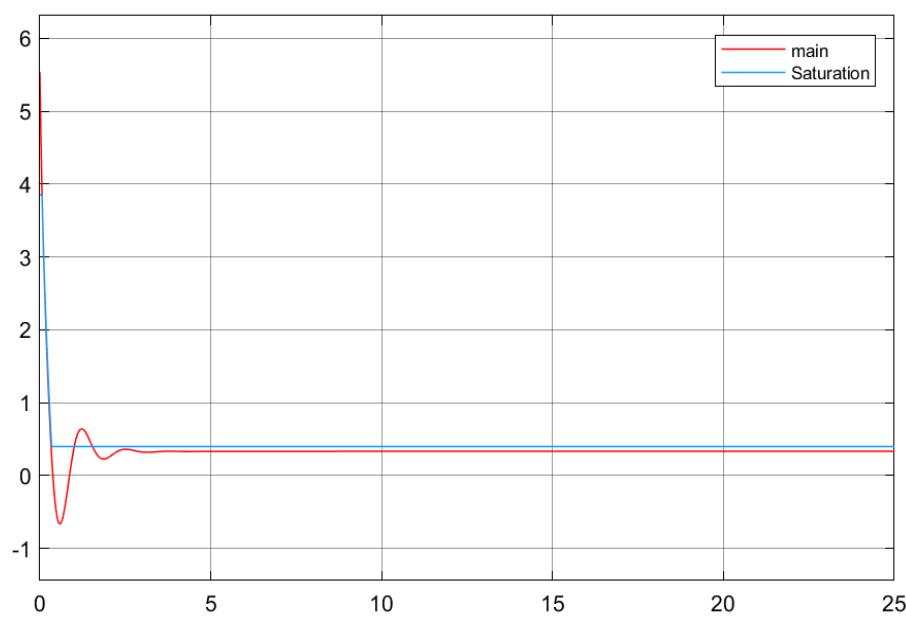
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامتر و...):

اشباع محرک:

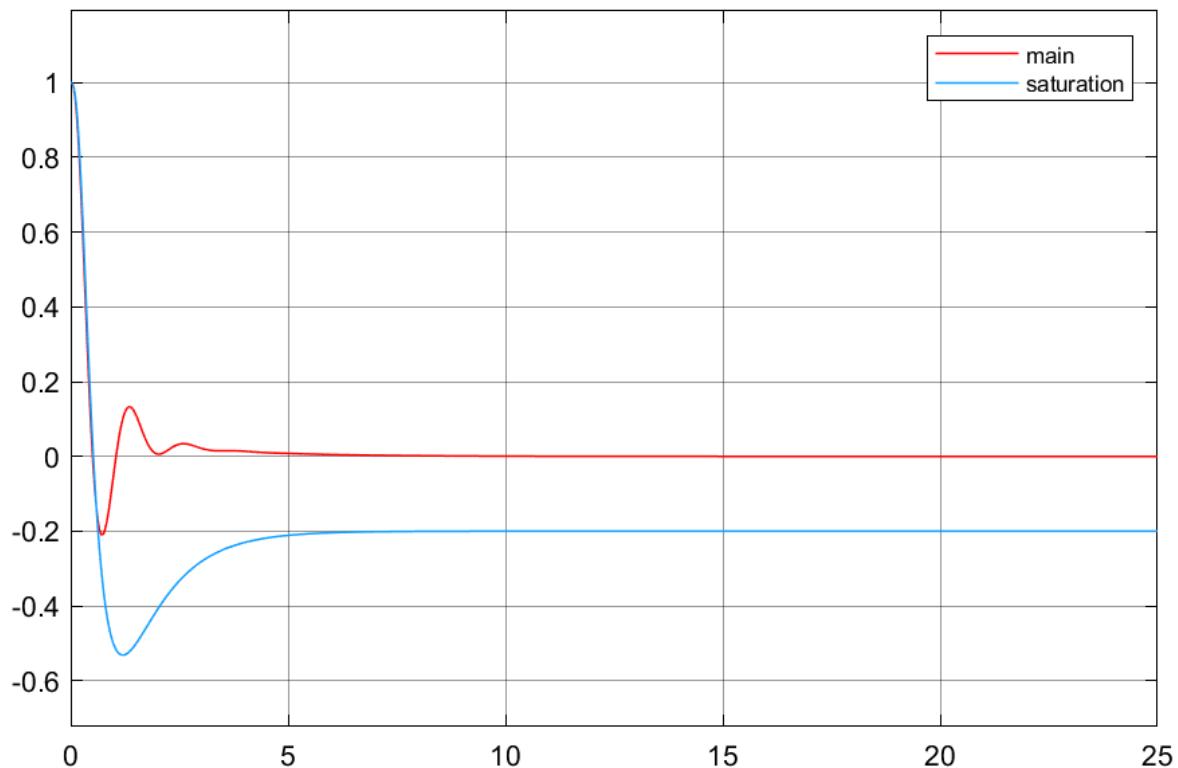
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطأ شده است(خطا صفر نمی شود). در ادامه به مقایسه ای سیگنال های فرمان میپردازیم که نشان دهنده ای نوع عملکرد بلوک اشباع می باشد.



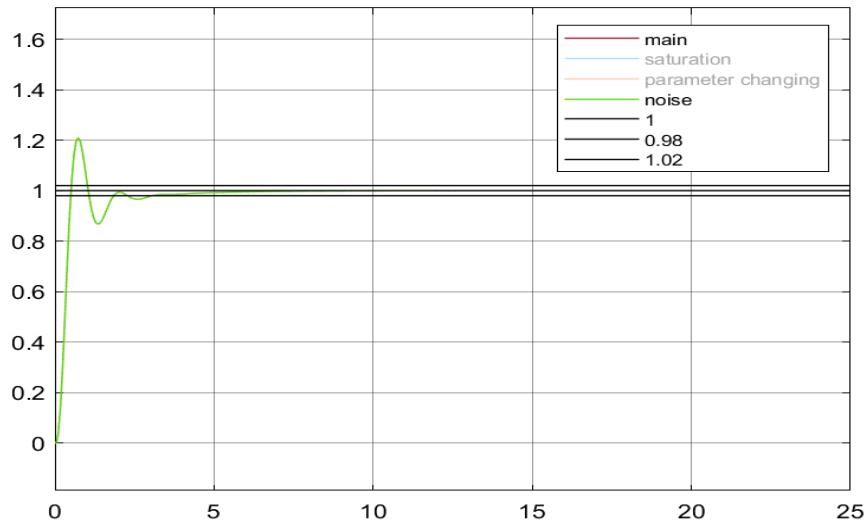
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطای می شود ادامه با مقایسه ای سیگنال های خطای داریم:



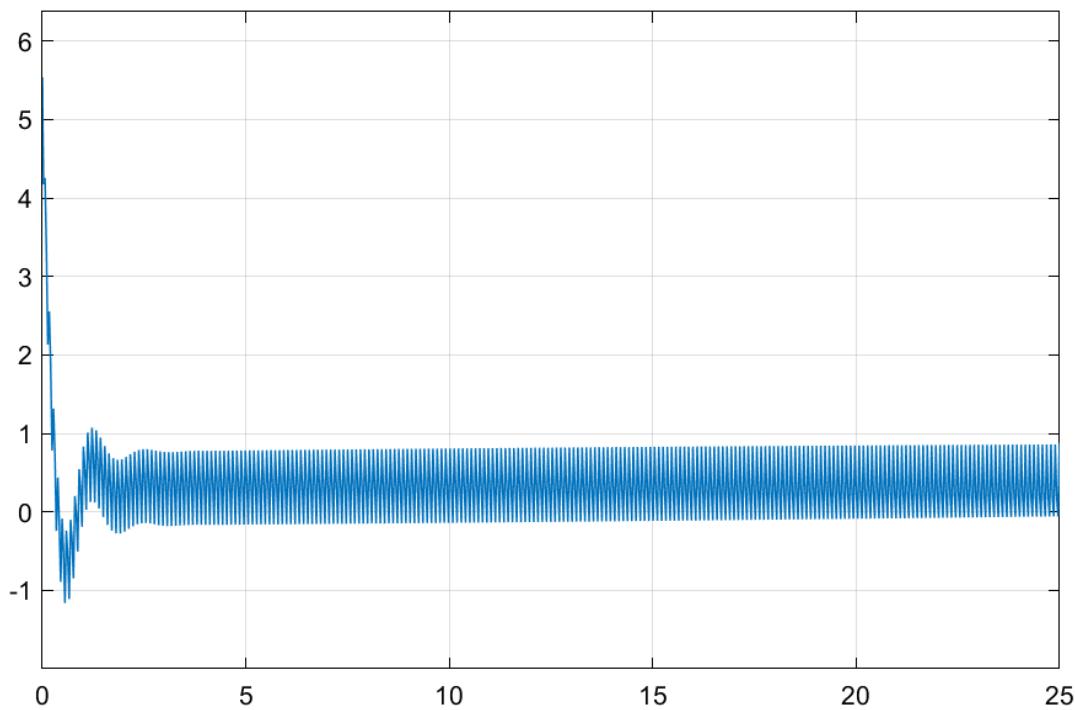
همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای انباشته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

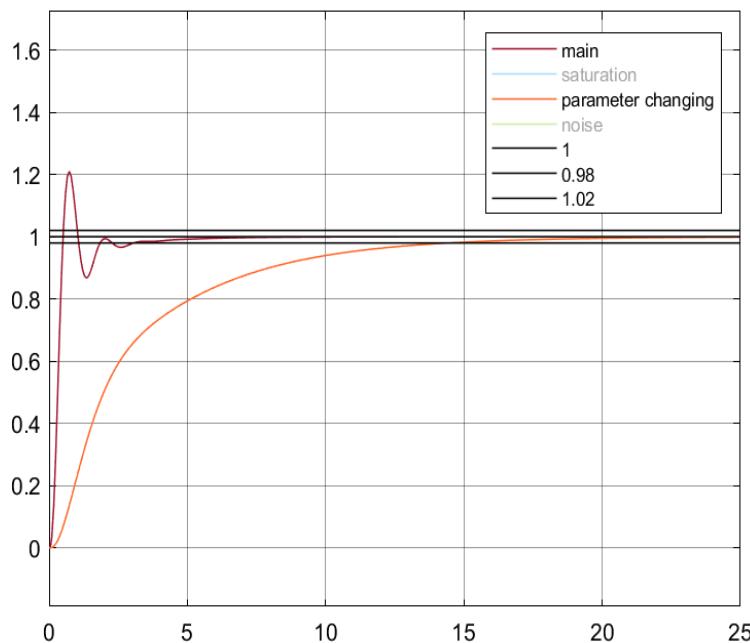


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و با از بین رفتن فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

4-2 به روش PR

در این روش ضریب T_d ، T_i ، k_p را به گونه ای تغییر می دهیم که به ازای هر کدام پاسخ پلهنوسان پایدار شود. با بدست آوردن این ضرایب که نام آنها را به ترتیب T_{du} ، T_{iu} ، K_u می نامیم.

که داریم:

```
% kp & Ti & Td for stable oscillation
ku=1.085;
Tdu=1.285;
Ti=0.1;
%ku/2 & 3.3Ti & 0.3Tdu
kp=ku/2;
Ti=3.3*Tiu;
Td=0.3*Tdu;
%kp=0.5425 & Ti=0.33 & Td=0.3855
s=tf('s');
g=63/((s+0.5)*(s+2)*(s+4));
```

```

c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
hold on
nyquist(g)
nyquist(c*g)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end
hold on
axis([-2 2 -2 2])
plot(x,y,'y');
legend('G','GC','r=1');

[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=4.6338 deg
%GMcg=-2.34 dB

t = out.PR(:, 1);
y = out.PR(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{::});
grid on;

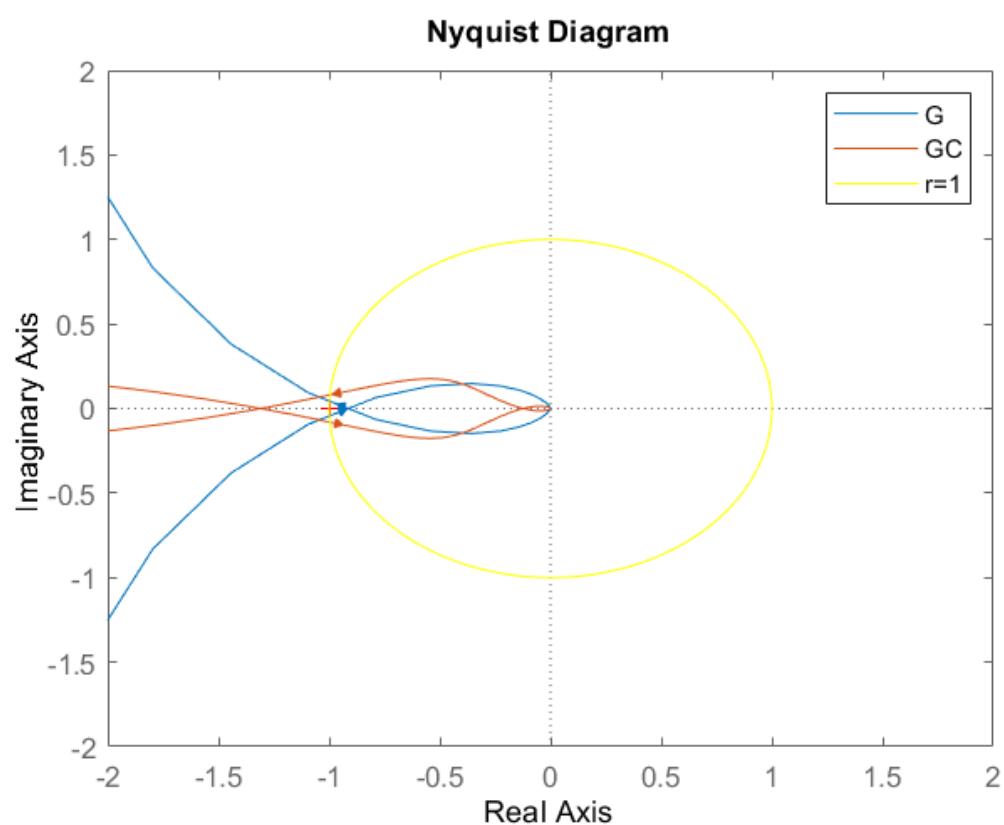
```

```

hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.42
%Ts%2=2.32
%overshoot%=24.6%
%d=0.08
%IAE=0.6199

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبهٔ ضرایب PID می‌کنیم که با مقایسهٔ دایاگرام‌های نایکوییست داریم که:



همانطور که می‌دانیم زمانی که نقطهٔ ۱ در سمت چپ دیاگرام قطبی (برای فرکانس مثبت) قرار بگیر سیستم (حلقهٔ بسته) پایدار می‌باشد و همانطور که مشخص است سیستم جبران شده نیز پایدار می‌باشد و دارای بهرهٔ -2.34 dB و دارای حد فاز 4.63° باشد. که نشان دهنده این است که

حد بهره بسیار کم شده و همچنین حد فاز تقریبا ثابت باقی مانده است. تاثیرات حد فاز و حد بهره در انتهای گزارشکار مورد بررسی قرار می گیرد.

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه طراحی ها در یک جدول این موارد آورده می شود. مشخصات گفته شده از روی این نمودار بدست آمده است:

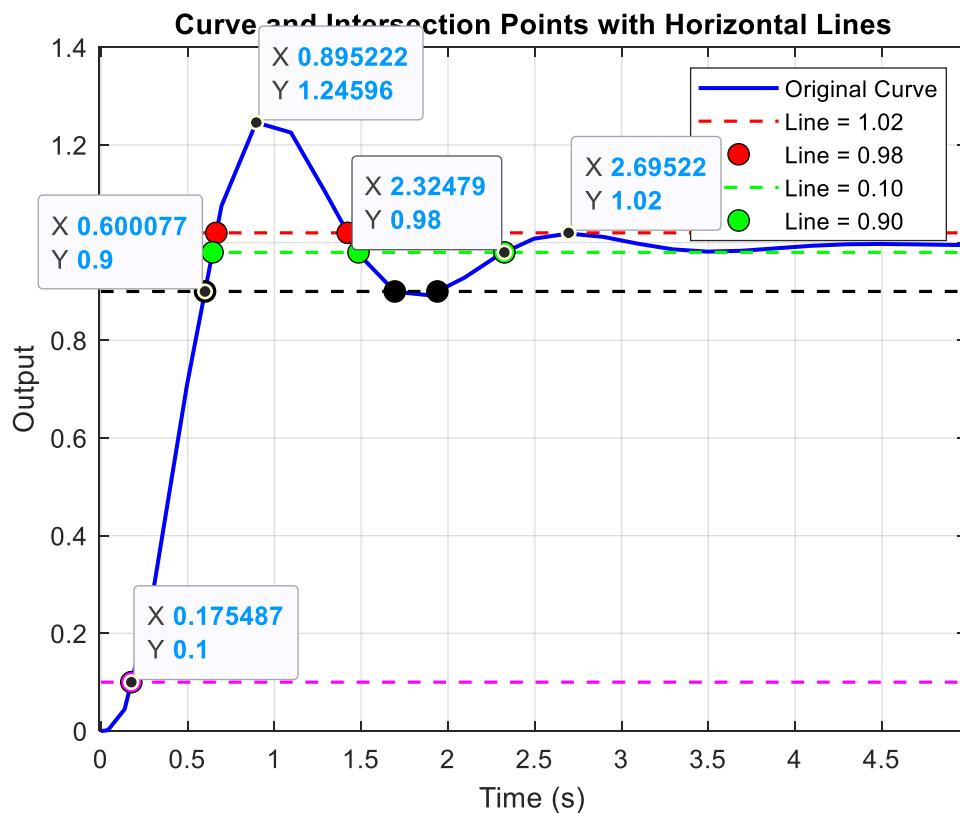
$$Tr=0.42$$

$$Ts\%2=2.32$$

$$overshoot\% = 24.6\%$$

$$d=0.08$$

$$IAE=0.6199$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ی طراحی ها این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ی 5 اعمال شده است):

Ts5%=10.7

Ts10%=8.86

overshoot%=163%

d=-

IAE=5.55

```
t = out.PR1(:, 1);

y = out.PR1(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

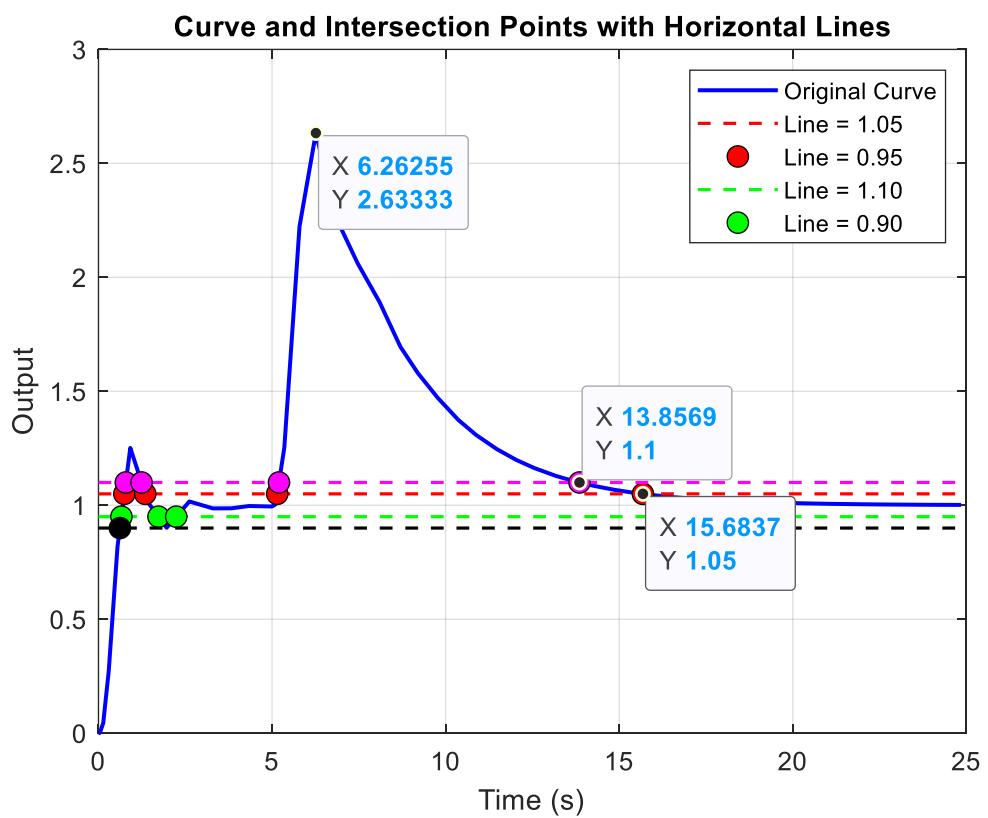
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = [{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
```

```

'UniformOutput', false)];
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-0.6199;
%Ts5%=10.7
%Ts10%=8.86
%overshoot%=163%
%d=-
%IAE=5.55

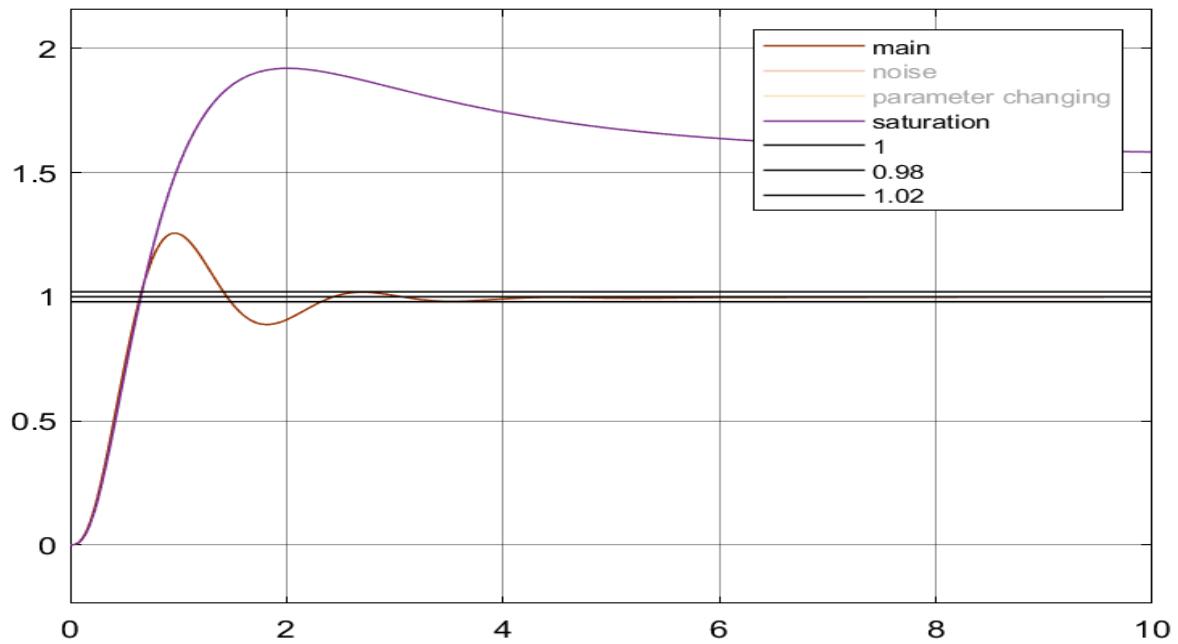
```



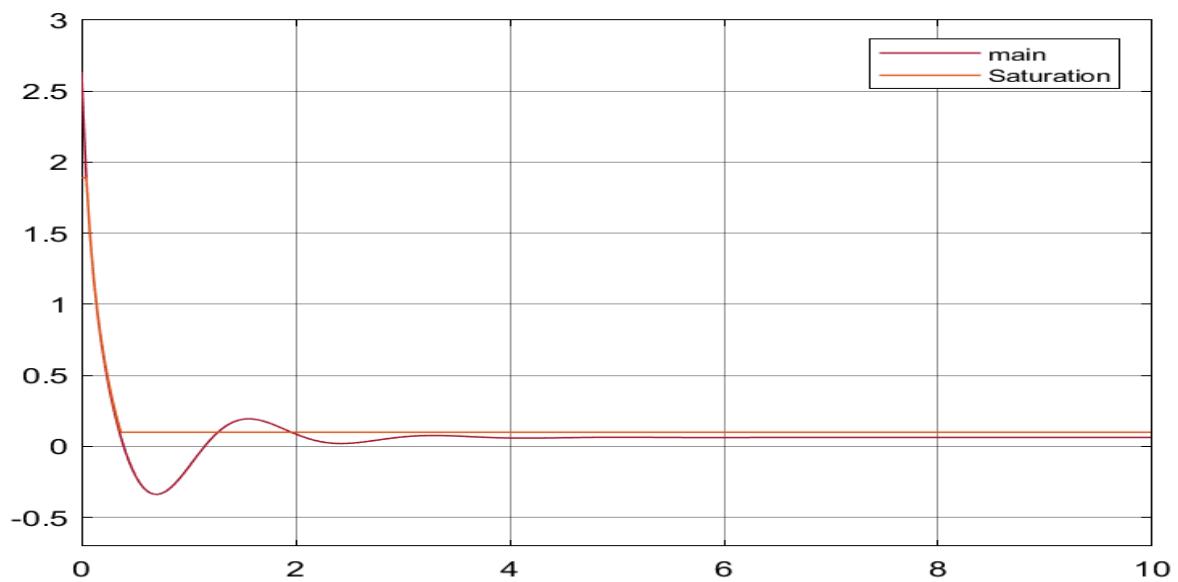
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامتر و...):

اشباع محرک:

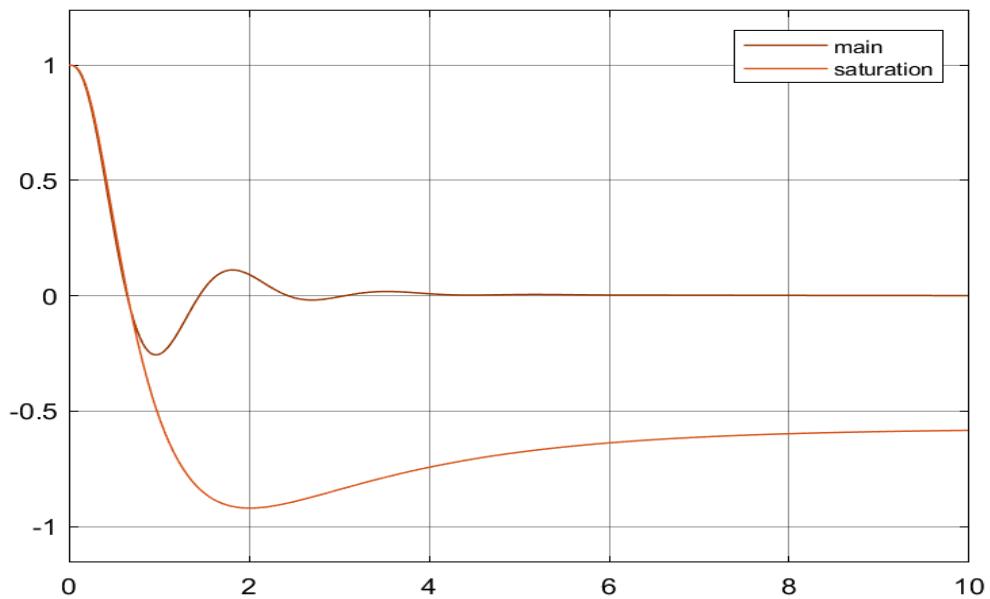
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطأ شده است(خطأ صفر نمی شود). در ادامه به مقایسه ای سیگنال های فرمان میپردازیم که نشان دهنده ای نوع عملکرد بلوک اشباع می باشد.



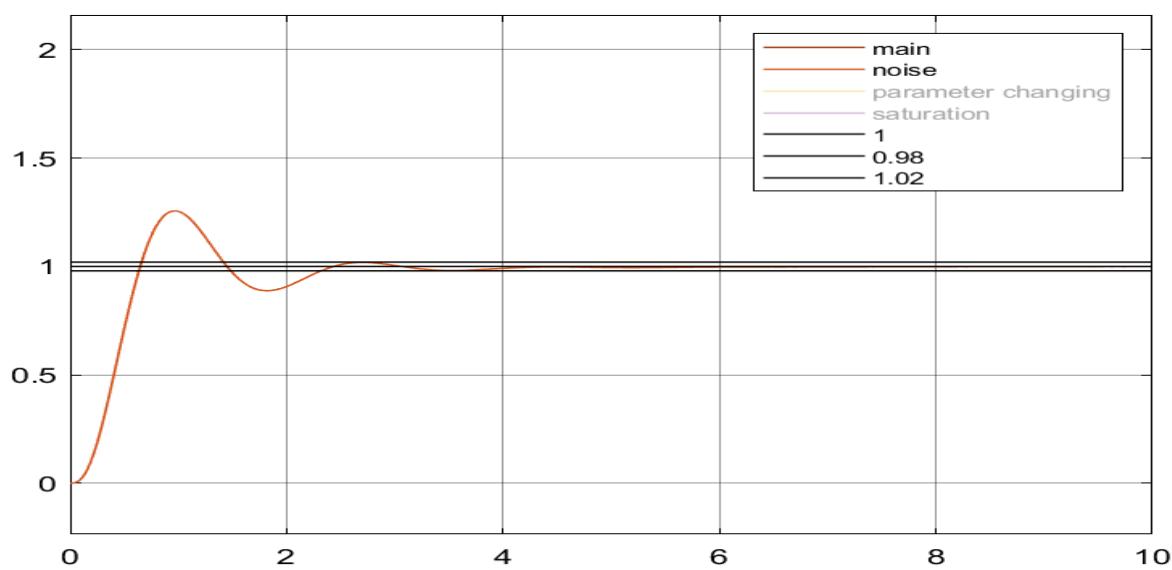
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطأ می شود ادامه با مقایسه ای سیگنال های خطأ داریم:



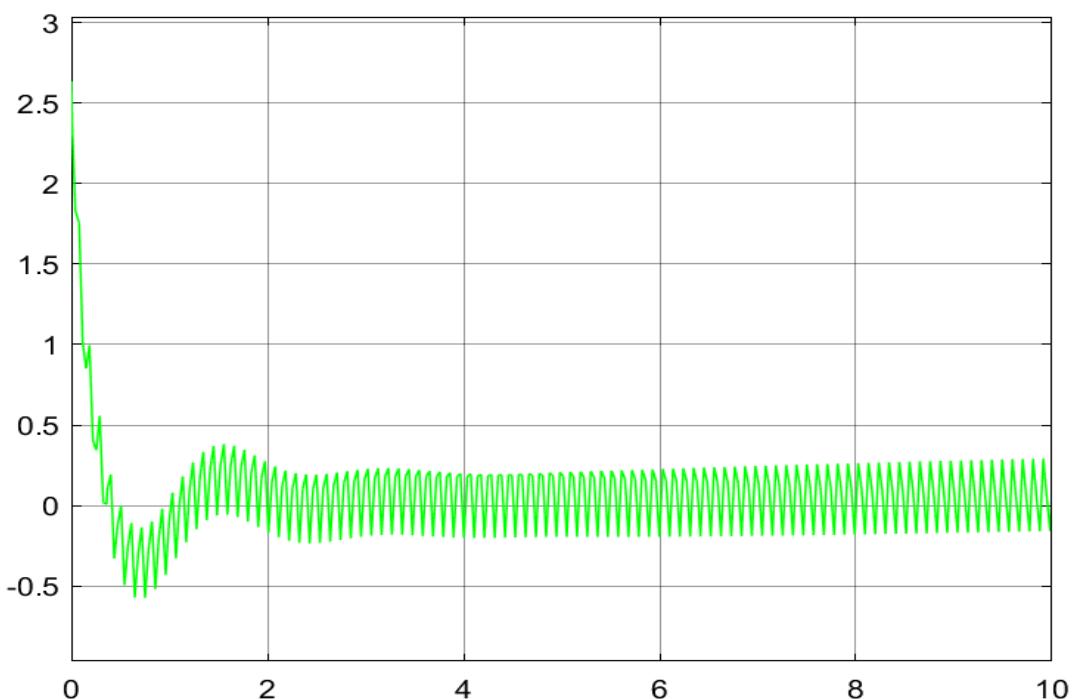
همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای انباشته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

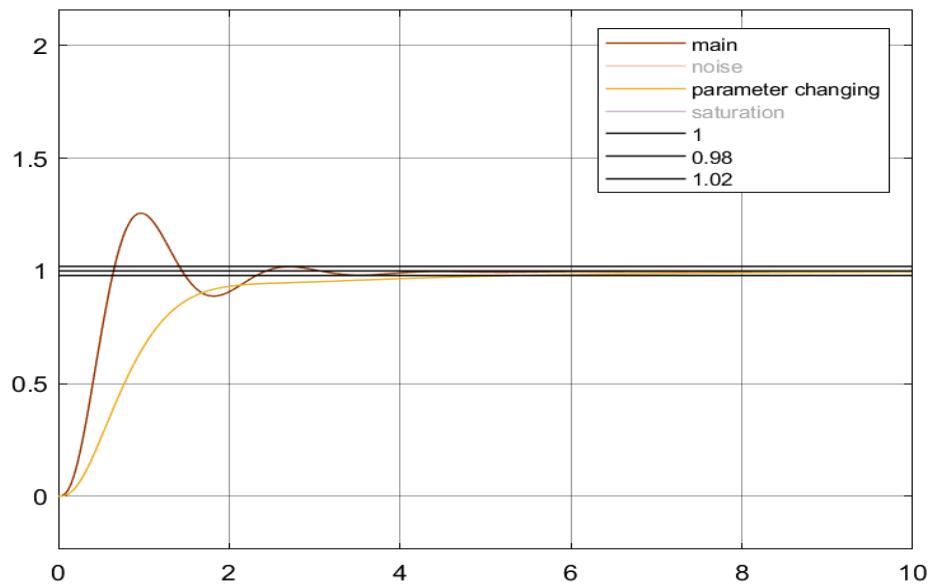


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و با از بین رفتن فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

بخش سوم

طراحی PID بر اساس حد بهره و حد فاز-ZN-

تعمیم یافته

1-3 دستیابی به حد بهره 6 dB

با استخراج معادلات لازم با طراحی PID بر اساس دست یابی به حد بهره 6 dB ، معادلات را اینگونه داریم که:

نکته: با توجه به اینکه کنترل کننده PID که به صورت $c(s) = kp + \frac{kp}{Ti.s} + \frac{kp.Td.s}{\frac{Td}{10}s + 1}$ باشد اما معادلات

از روی فرم ایده آل به دست آمده است ($c(s) = kp + \frac{kp}{Ti.s} + kp.Td.s$) اما تقریبا می توان گفت مقادیر

بدست آمده برای PID از روی هردوی معادلات هیچ تفاوتی وجود ندارد.

$$: c(s) = kp + \frac{kp}{Ti.s} + \frac{kp.Td.s}{\frac{Td}{10}s + 1}$$

معادلات به دست آمده از روی

$$Td = \sqrt{\frac{1}{4.w^2 - \frac{w^2}{100}}}$$

$$kp = \frac{\frac{rB}{rA}}{1 + \frac{Td^2}{10}.w^2}$$

$$\frac{\frac{1}{10}}{1 + \frac{Td^2}{100}.w^2}$$

$$Ti = 4.Td$$

$$: c(s) = kp + \frac{kp}{Ti.s} + kp.Td.s$$

معادلات به دست آمده از روی

$$kp = \frac{rB}{rA}$$

$$Td = \frac{1}{2.w}$$

$$Ti = \frac{2}{w}$$

که در آن w فرکانس برخورد دیاگرام نایکوییست سیستم جبران نشده با محور حقیقی و rA محل برخورد دیاگرام نایکوییست سیستم جبران نشده با محور حقیقی و rB محل برخورد دیاگرام نایکوییست سیستم جبران شده با محور حقیقی می باشد.

همانطور که در کد هم یک نکته به صورت کامنت آمده است که با kp بدست آمده از این روابط به حد بهره مورد نظر نمیرسیم و با سعی و خطا و تغییر آن به این امر دست می یابیم.

داریم:

```
rA=0.914;

rB=0.5012; %6dB
w=3.37;

%finding kp & Ti & Tp for PID controller
kp=rB/rA;
Td=1/(2*(w));
Ti=2/w;
%kp=0.5484
%Td=0.1484
%Ti=0.5935

%with kp=5484 we can achieve gain margin=6 dB so by changing kp=0.39 we can
kp=0.39;
s=tf('s');
g=63/((s+0.5)*(s+2)*(s+4));
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
cg=c*g;
hold on
nyquist(g)
nyquist(cg)

r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'r');

[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=3.52 deg
%GMcg=6.05 dB
```

```

t = out.g(:, 1);
y = out.g(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

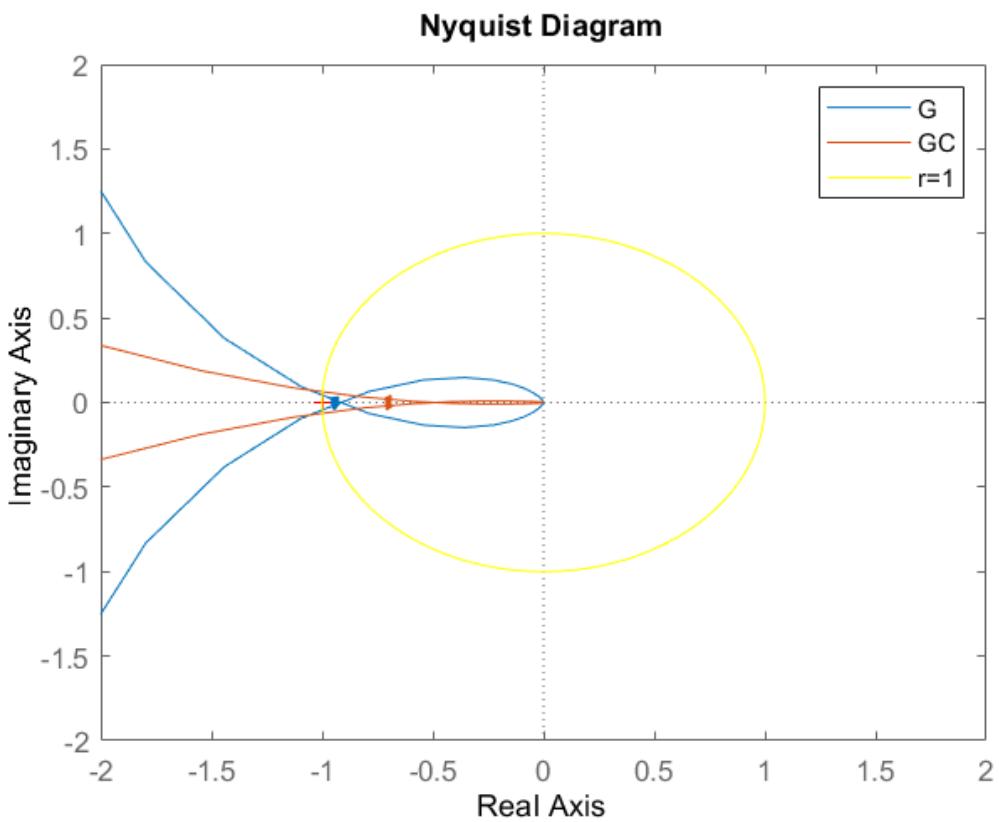
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.61
%Ts%2=5.91
%overshoot%=41.25%
%d=0.14
%IAE=1.1894

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبه ی ضرایب PID می کنیم که با

مقایسه ی دایاگرام های نایکوپیست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده تقریباً ثابت مانده است و برابر 3.52° استو همچنین بهره سیستم جبران شده مورد نظر که برای دست یابی به آن تلاش شده است برابر با 6 dB است که تقریباً برابر با 6 dB مورد نظر است.

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه‌ی موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه‌ی طراحی‌ها در یک جدول این موارد آورده می‌شود.
مشخصات گفته شده از روی این نمودار بدست آمده است:

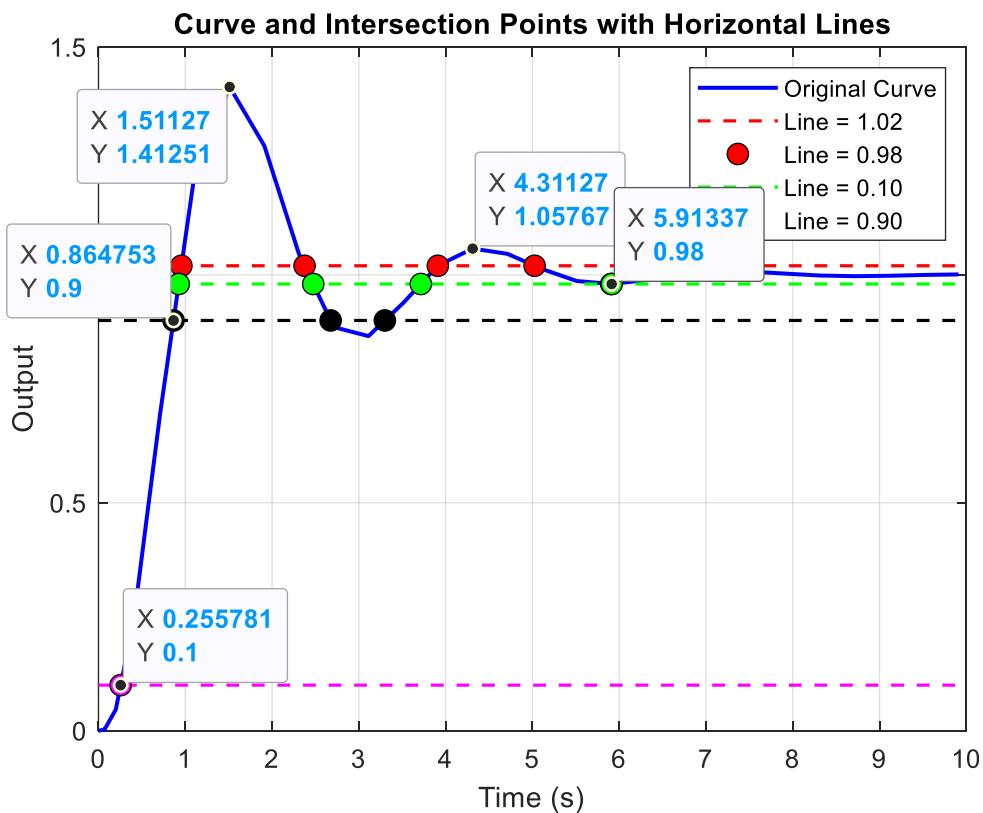
$$Tr=0.61$$

$$Ts\%2=5.91$$

$$\text{overshoot}\% = 41.25\%$$

$$d=0.14$$

$$\text{IAE}=1.1894$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ای طراحی ها این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ی 7 اعمال شده است):

$$Ts5\% = 5.63$$

$$Ts10\% = 5.34$$

$$overshoot\% = 247\%$$

$$d = 0.13$$

$$IAE = 4.26$$

```
t = out.g1(:, 1);
y = out.g1(:, 2);
```

```

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

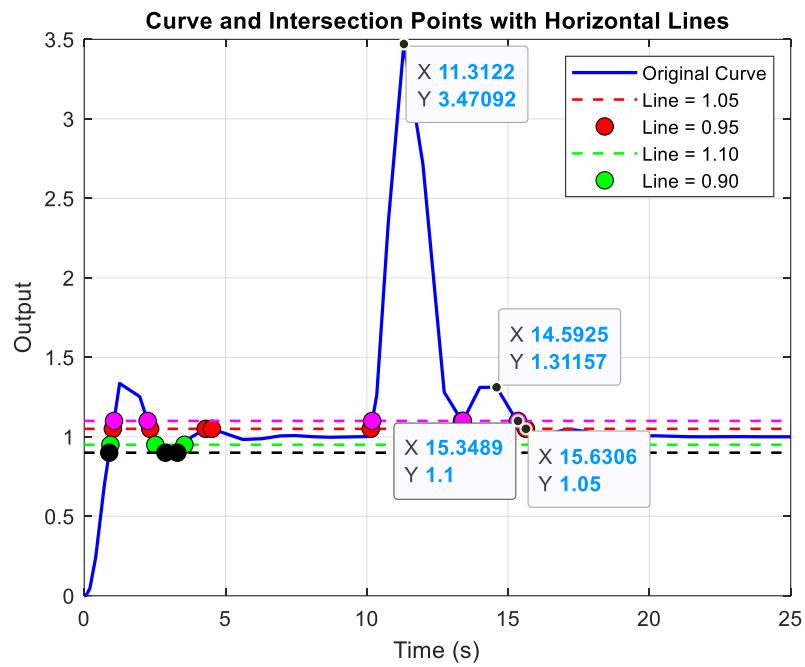
% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-1.1894;
%Ts5%=5.63
%Ts10%=5.34
%overshoot%=247%
%d=0.13
%IAE=4.26

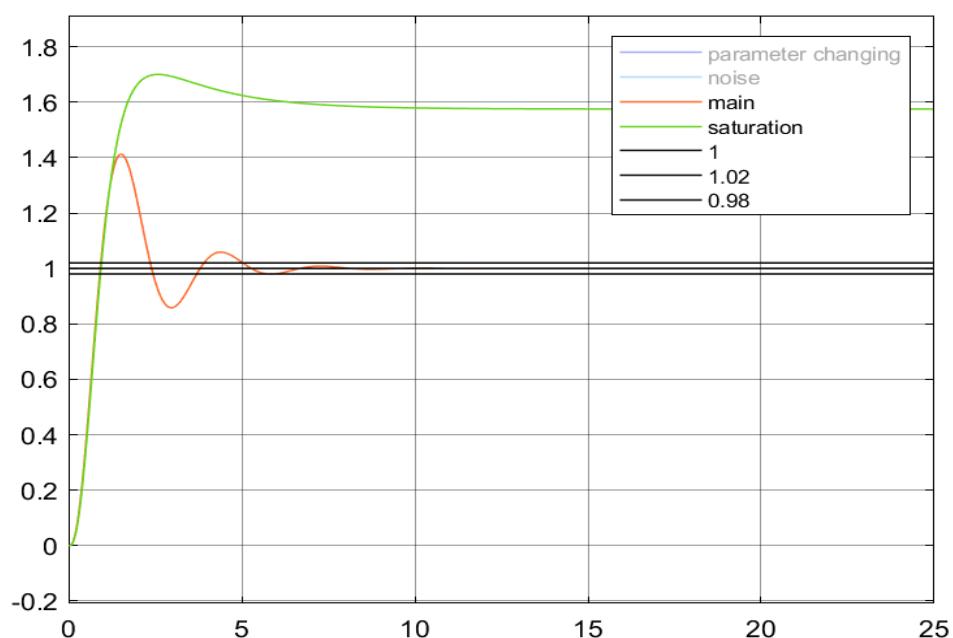
```



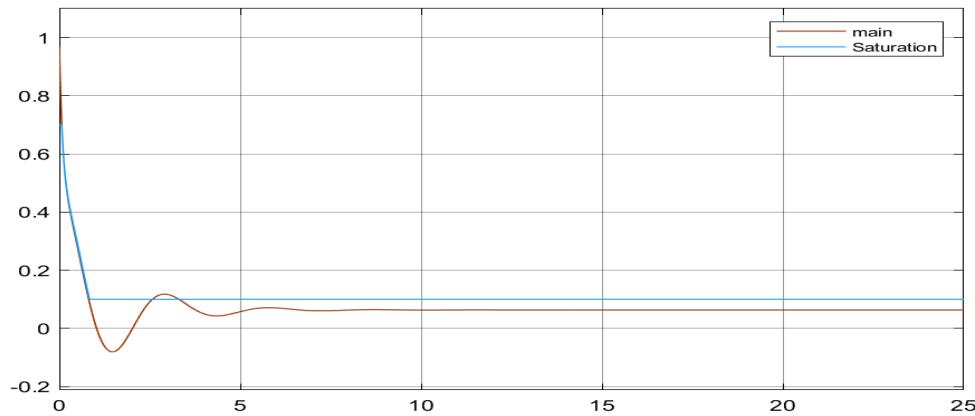
در ادامه ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می شود(نویز ، تغییر پارامترو...):

اشباع محرک:

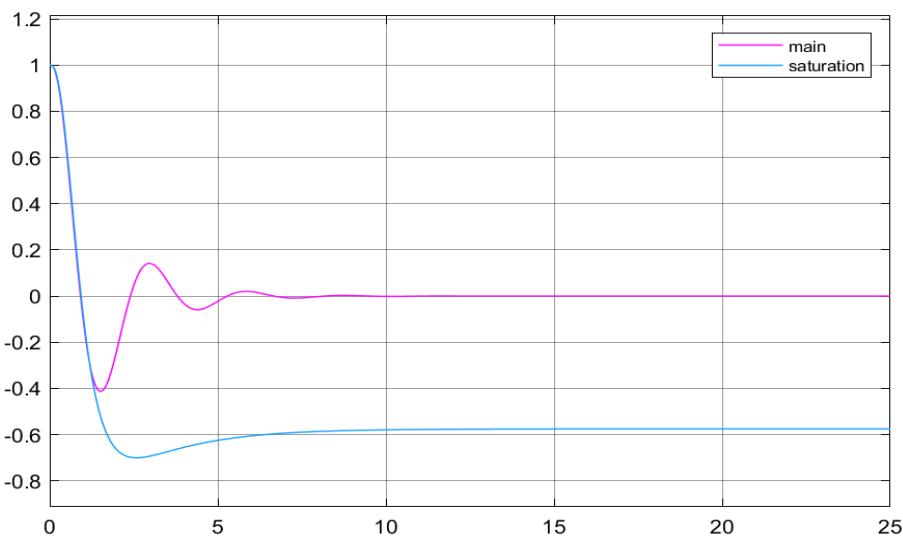
در این بخش به مقایسه ی خروجی ها بر اثر اشباع و خروجی اصلی می پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطای صفر نمی شود. در ادامه به مقایسه ی سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



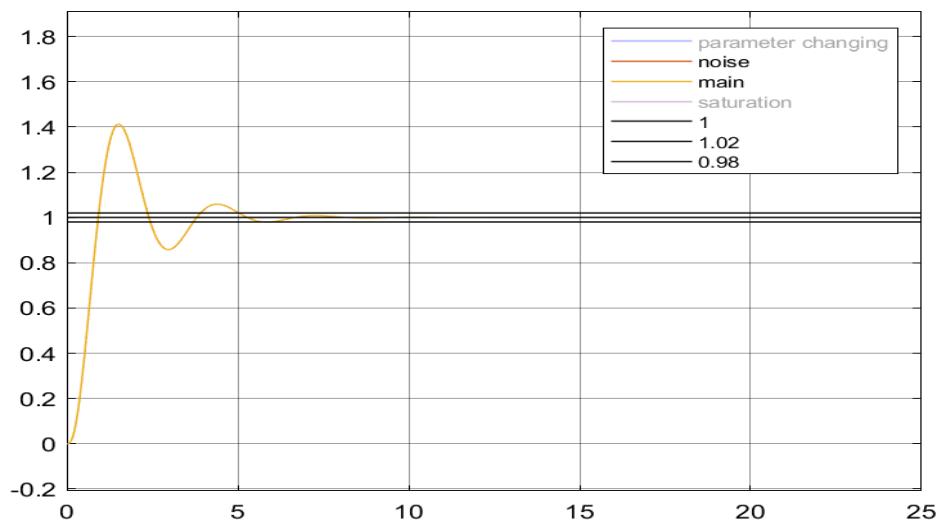
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطای صفر نمی شود ادامه با مقایسه ی سیگنال های خطای داریم:



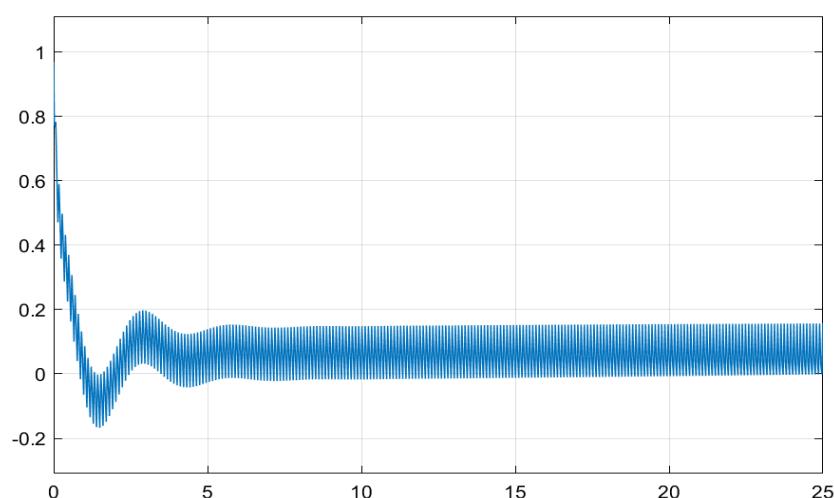
همانطور که قابل مشاهده است با مقایسه ی سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای اینباشته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

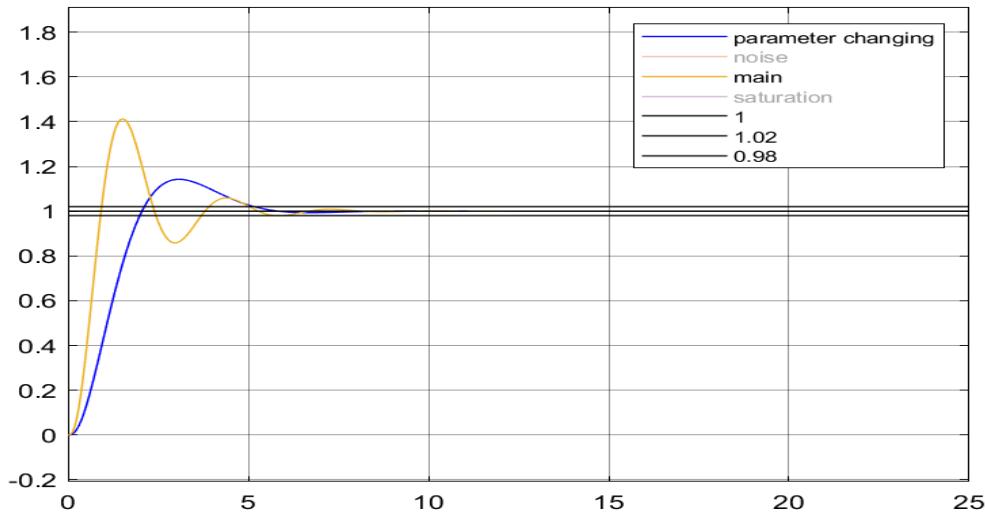


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و با از بین رفتن فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

2-3 دستیابی به حد فاز 45 deg

با استخراج معادلات لازم با طراحی PID بر اساس دست یابی به حد فاز 45 deg ، معادلات را اینگونه داریم که:

نکته: با توجه به اینکه کنترل کننده PID به صورت $c(s) = kp + \frac{kp}{Ti.s} + \frac{kp.Td.s}{10s+1}$ باشد اما معادلات

از روی فرم ایده آل به دست آمده است ($c(s) = kp + \frac{kp}{Ti.s} + kp.Td.s$) اما تقریبا می توان گفت مقادیر

بدست آمده برای PID از روی هردوی معادلات هیچ تفاوتی وجود ندارد.

$$c(s) = kp + \frac{kp}{Ti.s} + \frac{kp.Td.s}{10s+1}$$

معادلات به دست آمده از روی

$$kp = \frac{\cos(\varphi B - \varphi A)}{1 + \frac{Td^2}{10} \cdot w^2}$$

$$\frac{1}{1 + \frac{Td^2}{100} \cdot w^2}$$

$$\frac{-1}{4.Td.w} + \frac{Td.w}{\frac{Td^2}{100}w^2 + 1} = \tan(\varphi B - \varphi A) \cdot \frac{\frac{Td^2 \cdot w^2}{100} + 1}{\frac{Td^2}{10}w^2 + 1}$$

$$Ti = 4.Td$$

$$c(s) = kp + \frac{kp}{Ti.s} + kp.Td.s$$

$$kp = \cos(\varphi B - \varphi A)$$

$$Td = \frac{\tan(\varphi B - \varphi A)}{2w} + \sqrt{\frac{\tan(\varphi B - \varphi A)^2}{4w^2} + \frac{1}{4w^2}}$$

$$Ti = 4.Td$$

که در آن w فرکانس برخورد دیاگرام نایکوپیست سیستم جبران نشده با دایره به شعاع ۱ و φA زاویه بردار محل برخورد دیاگرام نایکوپیست سیستم جبران نشده با دایره به شعاع ۱ و φB زاویه برخورد دیاگرام نایکوپیست سیستم جبران شده با دایره به شعاع ۱ می باشد.

همانطور که در کد هم یک نکته به صورت کامنت آمده است که با kp بدست آمده از این روابط به حد فاز مورد نظر نمیرسیم و با سعی و خطأ و تغییر آن به این امر دست می یابیم.

در ادامه داریم:

```
%point of touch Re=-0.45 & Im=-0.893
QA=2.04;
QB=45;
w=3.23;

%finding kp & Ti & Tp for PID controller
```

```

kp=cosd(QB-QA);
Td=(tand(QB-QA)/(2*w))+((sqrt(((tand(QB-QA)/w)^2)+(1/(w^2))))/2);
Ti=4*Td;
%kp=0.7318
%Td=0.3515
%Ti=1.4061

%with this kp we cant achieve phase margin=45 deg so by kp=0.54704 we can
kp=0.54704;
s=tf('s');
g=63/((s+0.5)*(s+2)*(s+4));
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
cg=c*g;
hold on
nyquist(g)
nyquist(cg)

r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'r')
plot(x,y,'y');
legend('G','GC','r=1')
[GMg_1,PMg,~,~]=margin(g);
GMg=20*log10(GMg_1);
%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1,PMcg,~,~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=45 deg
%GMcg=19.39 dB

t = out.p(:, 1);
y = out.p(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    %touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i),
'LineWidth',1.2);

```

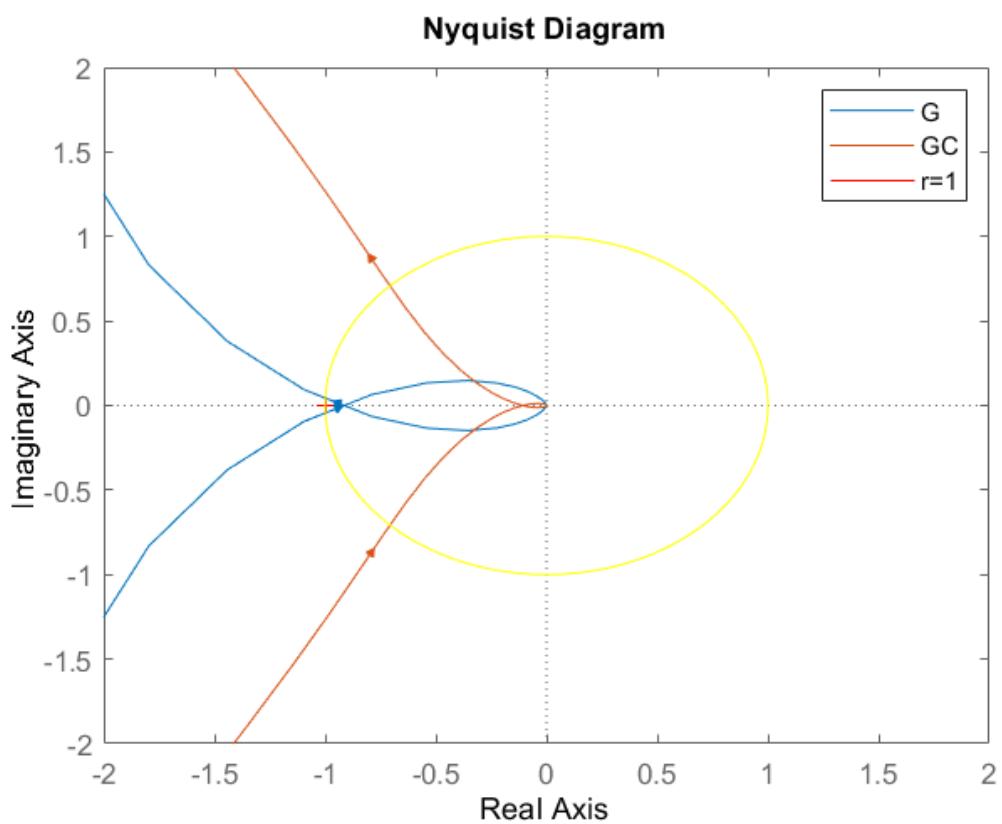
```

% showing touching points
plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
      'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:, :});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
%Tr=0.4
%Ts%2=2.99
%overshoot%=54%
%d=0.04
%IAE=0.9259

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبه‌ی ضرایب PID می‌کنیم که با مقایسه‌ی دایاگرام‌های نایکوپیست داریم که:



همانطور هم که در کد نشان داده شده فاز سیستم جبران شده که برای آن تلاش شده است برابر با 45 درجه می باشد و همچنین بهره سیستم جبران شده برابر با 19.38 dB است که نسب به سیستم جبران نشده خیلی بیشتر شده است.

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همه ای موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همه ای طراحی ها در یک جدول این موارد آورده می شود. مشخصات گفته شده از روی این نمودار بدست آمده است:

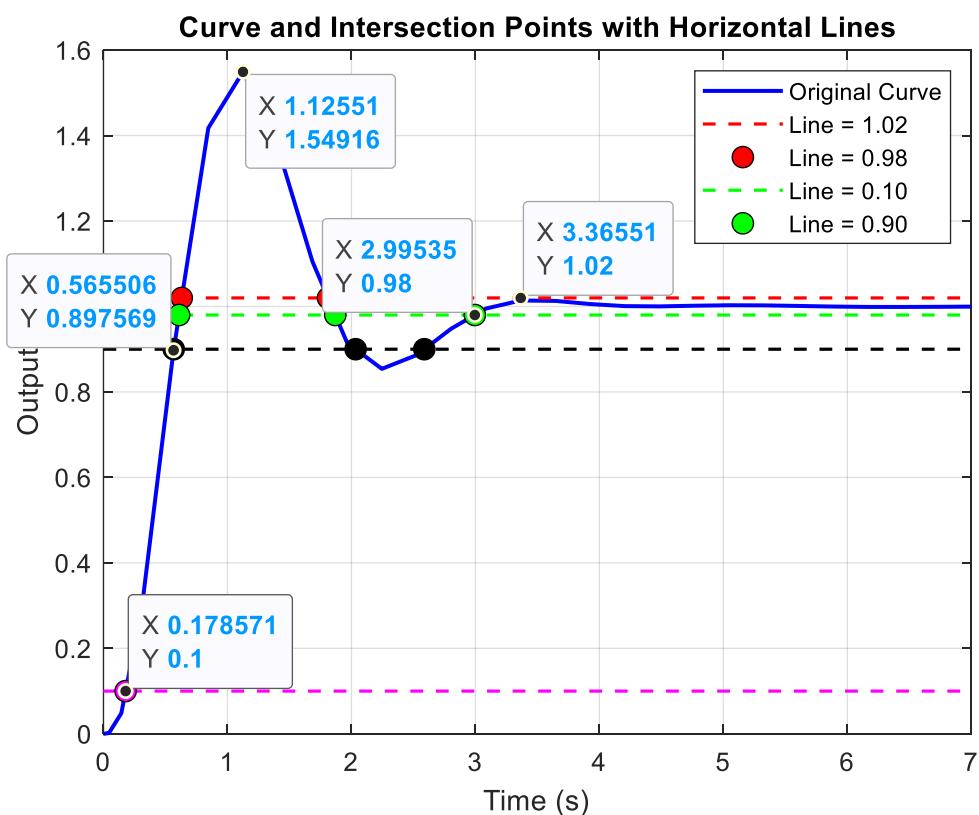
$$Tr=0.4$$

$$Ts\%2=2.99$$

$$overshoot\% = 54\%$$

$$d=0.04$$

$$IAE=0.9259$$



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ۷ اعمال شده است):

Ts5%=3.9

Ts10%=3.29

overshoot%=151%

d=0.013

IAE=1.9568

```
t = out.p1(:, 1);

y = out.p1(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

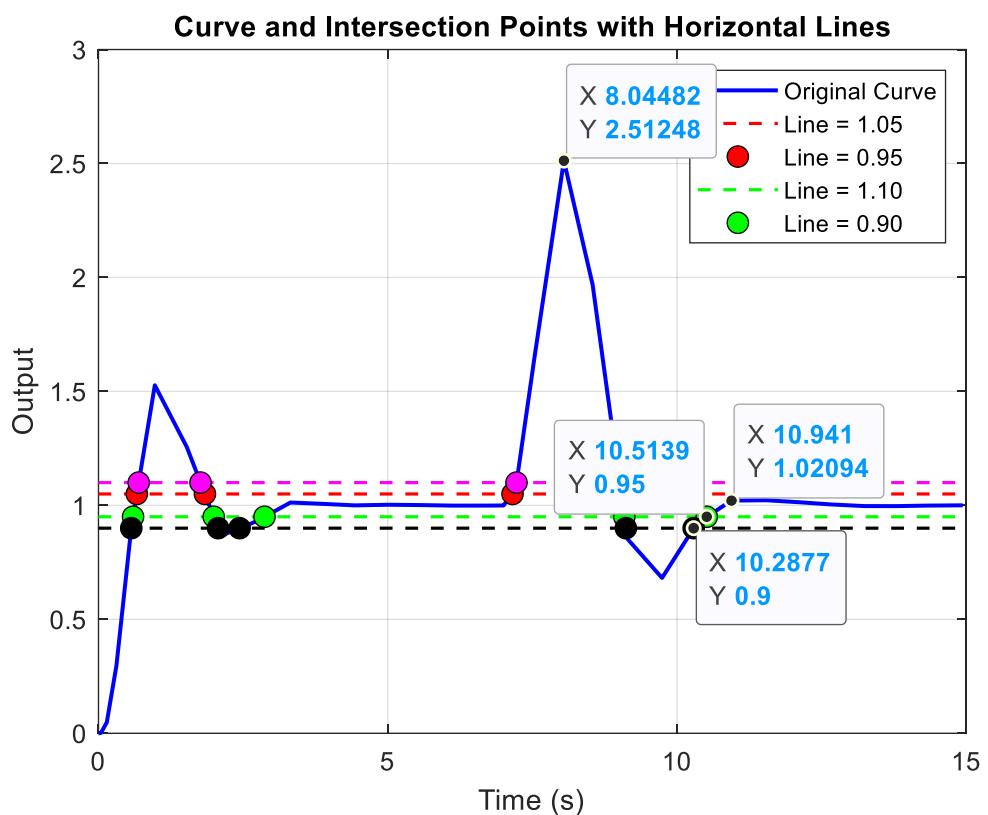
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
```

```

legend_entries = [ {'Original Curve'}, ...
                    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
                    'UniformOutput', false)];
legend(legend_entries{:});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-0.9259;
%Ts5%=3.9
%Ts10%=3.29
%overshoot%=151%
%d=0.013
%IAE=1.9568

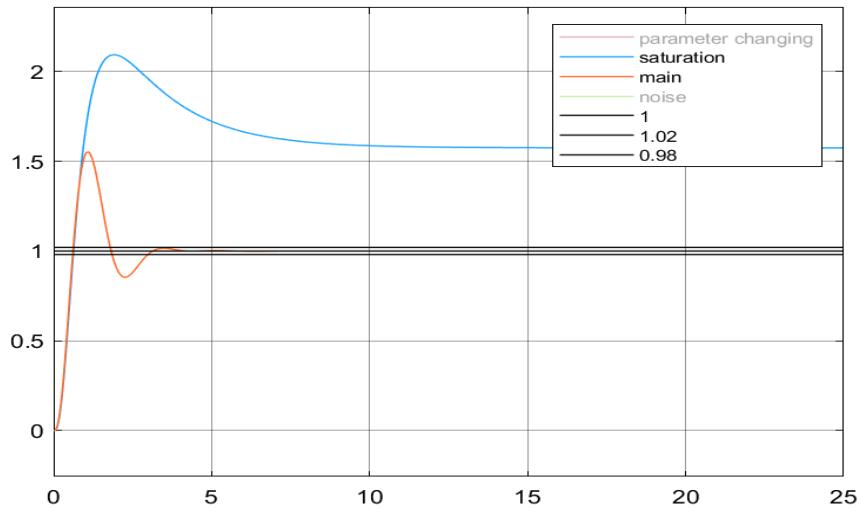
```



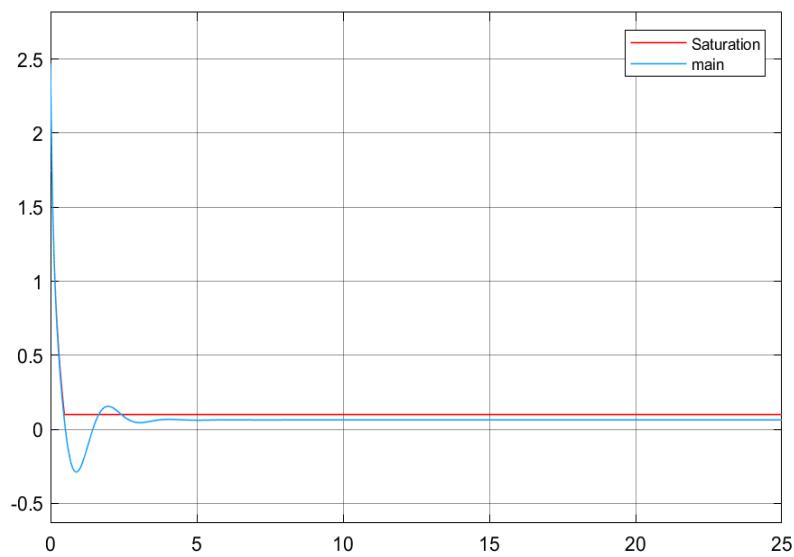
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامترو...):

اشباع محرک:

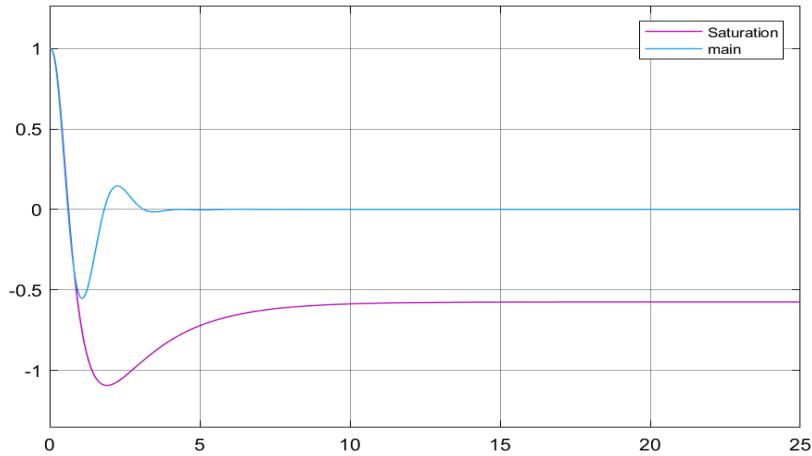
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطأ شده است(خطأ صفر نمی شود). در ادامه به مقایسه ای سیگنال های فرمان میپردازیم که نشان دهنده ای نوع عملکرد بلوک اشباع می باشد.



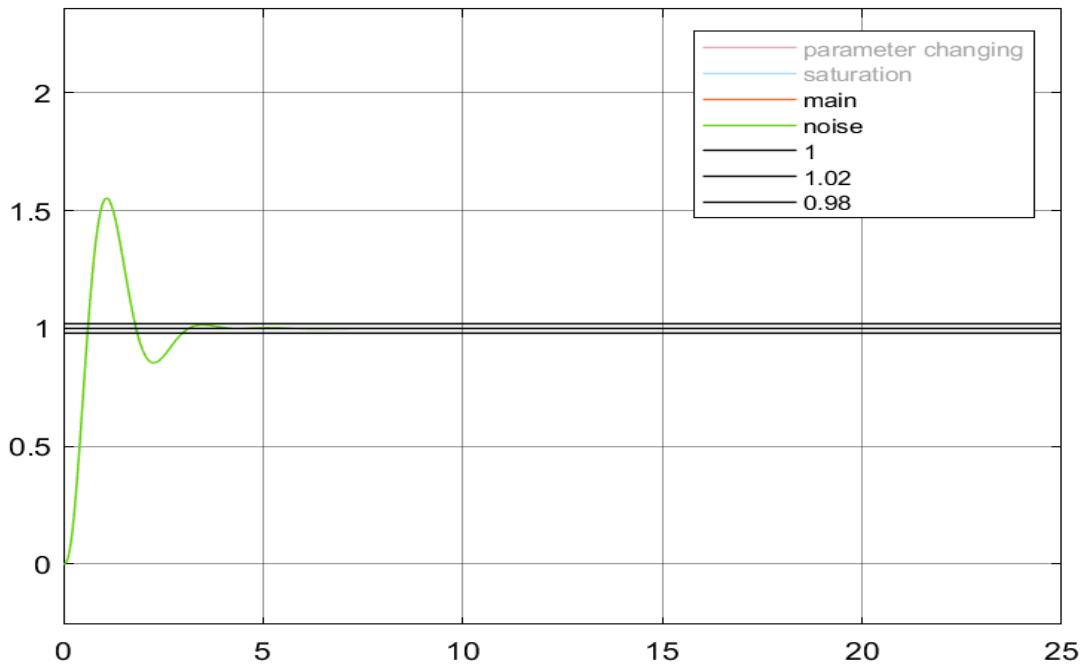
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطأ می شود ادامه با مقایسه ای سیگنال های خطأ داریم:



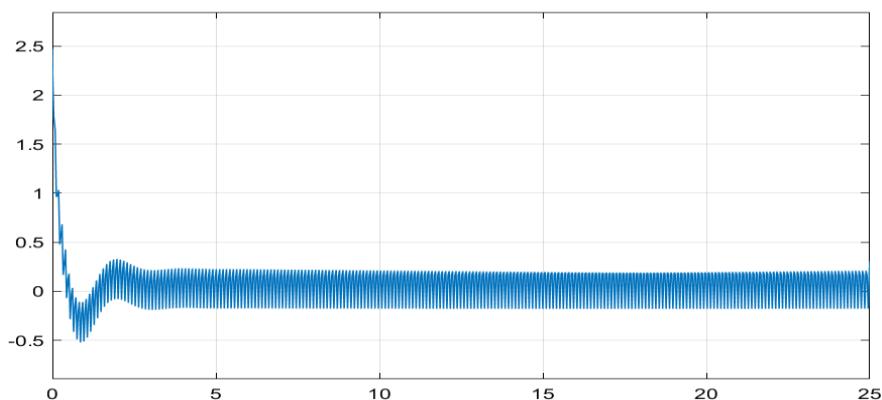
همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطأ میتوان متوجه شد که به دلیل وجود بلوک اشباع خطأ انباشته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

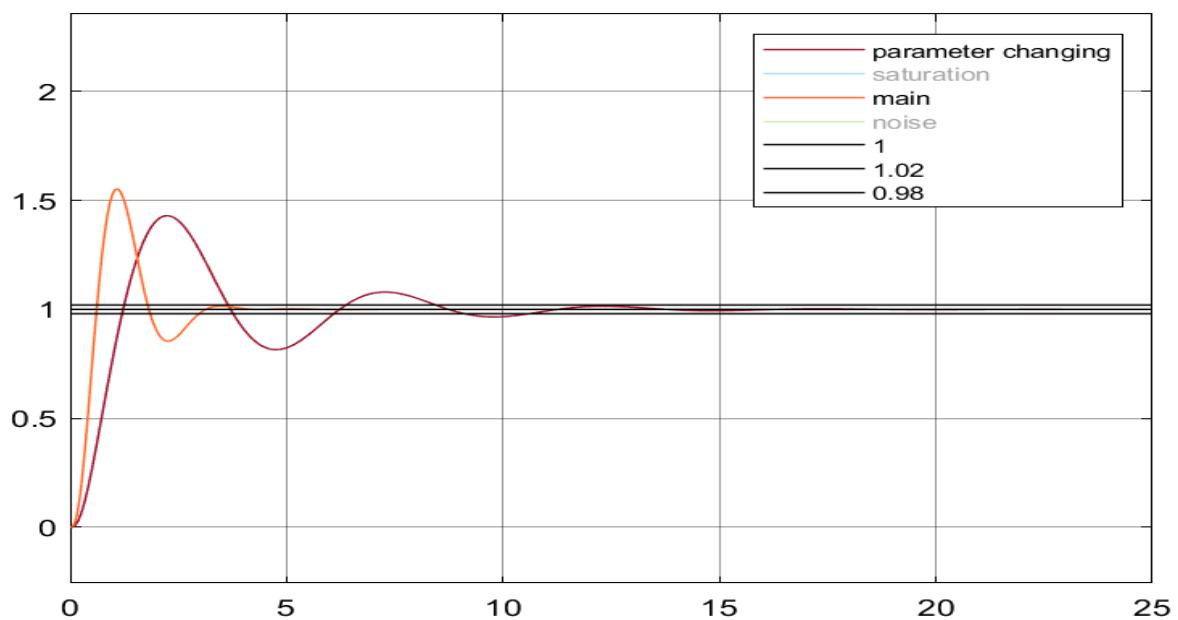


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و با از بین رفتن فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

3-3 بهینه سازی برای دست یابی به حد فاز $deg\ 45$ و حد بهره $6\ dB$

در این بخش تلاش این بوده است که با بهینه سازی هرودی فاز $deg\ 45$ و حد بهره $6\ dB$ را به طور همزمان داشته باشیم. به دلیل اینکه این دو مورد در فرکانس های متفاوت صورت می گیرند رسیدن دقیق به این دو مقدار ممکن نمی باشد و تلاش این است با توابع خطأ و توابع بهینه سازی به این دو مقدار نزدیک شویم(به دلیل ضعیف بودن CPU سیستم توابعی همچون ژنتیک را نتوانستم run بگیرم). در این بخش با کد اعمالی و با سعی و خطأ و انتخاب بازه های مناسب برای رسیدن به این دو هدف ثمره \deg حد فاز $7.4541\ dB$ و حد بهره $37.83\ deg$ صورت پذیرفته است. در ادامه کد بهینه سازی را داریم:

```
% Desired Gain Margin and Phase Margin
GM_desired = 2; % Desired Gain Margin in 1/|gc(jw)|
PM_desired = 45; % Desired Phase Margin in degrees

% Optimization bounds
lb = [0.001, 0.001, 0.001]; % Lower bounds for Kp, Ti, Td
ub = [10, 10, 10]; % Upper bounds for Kp, Ti, Td
% Optimization options using Particle Swarm Optimization (PSO)
options = optimoptions('particleswarm', 'SwarmSize', 50, 'MaxIter', 100,
'UseParallel', true);

% Optimization function
opt_func = @(x) cost_function(x(1), x(2), x(3), GM_desired, PM_desired);

% PSO optimization
[x_opt, fval_opt] = particleswarm(opt_func, 3, lb, ub, options);

% Display optimized parameters and results
disp(['Optimized Kp: ', num2str(x_opt(1))]);
disp(['Optimized Ti: ', num2str(x_opt(2))]);
disp(['Optimized Td: ', num2str(x_opt(3))]);

% Compute Gain Margin and Phase Margin with optimized parameters
[GM_opt, PM_opt] = compute_margins(x_opt(1), x_opt(2), x_opt(3));

disp(['Optimized Gain Margin (1/|gc(jw)|): ', num2str(GM_opt)]);
disp(['Optimized Phase Margin (degrees): ', num2str(PM_opt)]);
disp(['GM Error: ', num2str(abs(GM_opt - GM_desired))]);
disp(['PM Error: ', num2str(abs(PM_opt - PM_desired))]);

% Define the optimized PID controller
s = tf('s');
PID_opt = x_opt(1) + (x_opt(1) / (x_opt(2) * s)) + ((x_opt(1) * x_opt(3) * s) /
(((x_opt(3) / 10) * s) + 1));

% Closed-loop transfer function
G = 63 / ((s + 0.5) * (s + 2) * (s + 4)); % Example plant
```

```

L_opt = PID_opt * G;

% Local Functions
function cost = cost_function(Kp,Ti,Td, GM_desired, PM_desired)
    % Compute Gain Margin and Phase Margin
    [GM, PM] = compute_margins(Kp,Ti,Td);

    % Compute normalized errors
    GM_error = abs((GM - GM_desired) / GM_desired);
    PM_error = abs((PM - PM_desired) / PM_desired);

    % Total cost
    cost = GM_error^2 + PM_error^2;
end

function [GM, PM] = compute_margins(Kp,Ti,Td)
    % Define the plant (example system)
    s = tf('s');
    G = 63 / ((s + 0.5) * (s + 2) * (s + 4)); % Example plant

    % Define PID controller
    PID = Kp + (Kp / (Ti * s)) + ((Kp * Td * s) / (((Td / 10) * s) + 1));

    % Open-loop transfer function
    L = PID * G;

    % Compute Gain Margin and Phase Margin
    [GM, PM,~,~] = margin(L);
end

```

در ادامه روند را مثل قبل ادامه می دهیم:

```

kp= 0.14267;

Ti= 0.14637;

Td=2.7491 ;

s=tf('s');
g=63/((s+0.5)*(s+2)*(s+4));
c=kp+(kp/(Ti*s))+((kp*Td*s)/(((Td/10)*s)+1));
cg=c*g;
hold on
nyquist(g)
nyquist(cg)
r=1;
for j=0:360
    x(j+1)=r*cos((pi/180)*j);
    y(j+1)=r*sin((pi/180)*j);
end

hold on
axis([-2 2 -2 2])
plot(x,y,'r')
plot(x,y,'y');

```

```

legend('G', 'GC', 'r=1')

[GMg_1, PMg, ~, ~]=margin(g);
GMg=20*log10(GMg_1);

%PMg=2.04 deg
%GMg=0.6 dB
[GMcg_1, PMcg, ~, ~]=margin(c*g);
GMcg=20*log10(GMcg_1);
%PMcg=37.83 deg
%GMcg=7.4541 dB

t = out.gp(:, 1);
y = out.gp(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

horizontal_lines = [1.02, 0.98, 0.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

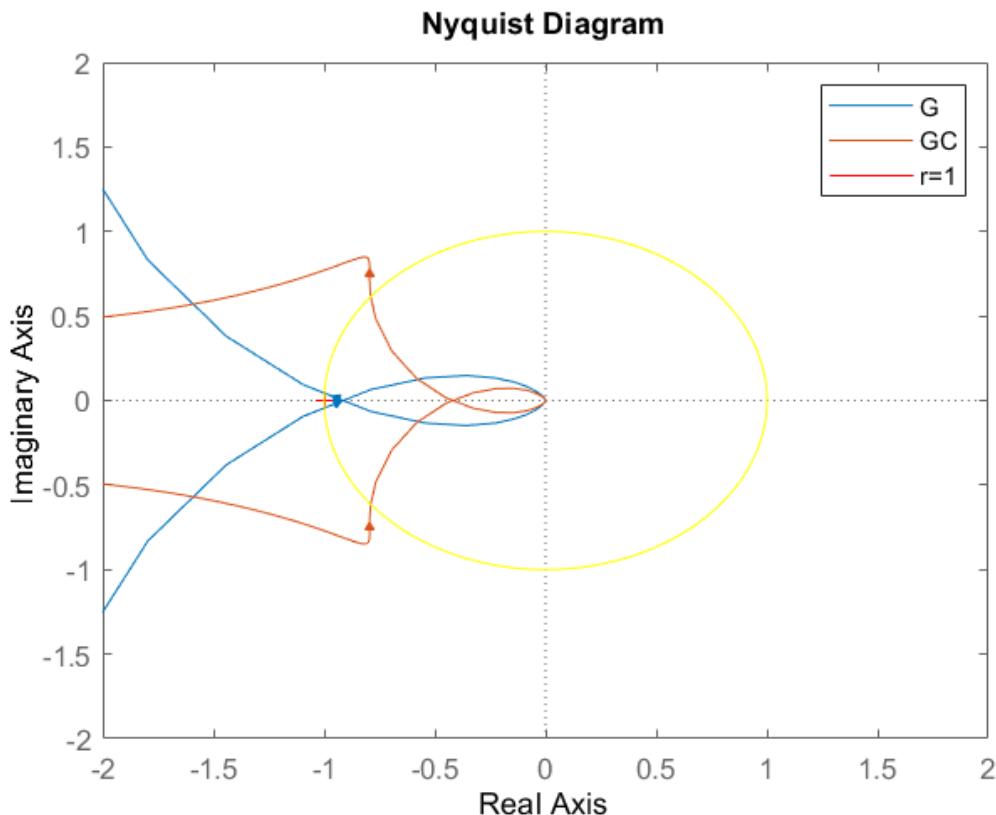
    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{::});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));

%Tr=0.83
%Ts%2=4.32
%overshoot%=-
%d=-
%IAE=3.05

```

در ابتدای کد با داشتن مقادیر سیستم مدل شده شروع به محاسبهٔ ضرایب PID می‌کنیم که با مقایسهٔ دایاگرام‌های نایکوپیست داریم که:



همانطور که مشخص است فاز سیستم جبران شده به 45 درجه نزدیک شده و حد بهرهٔ سیستم جبران شده هم نیز به بهرهٔ 6 دسی بل بسیار نزدیک شده که این نشان دهندهٔ عملکرد بسیار خوبه بهینه سازی و ضرایب تعیین شده برای PID می‌باشد.

در ادامه مشخصات پاسخ پله سیستم حلقه بسته به دست آمده است که همهٔ موارد به صورت کامنت در کد آمده است که در پایان گزارش کار برای همهٔ طراحی‌ها در یک جدول این موارد آورده می‌شود. مشخصات گفته شده از روی این نمودار بدست آمده است:

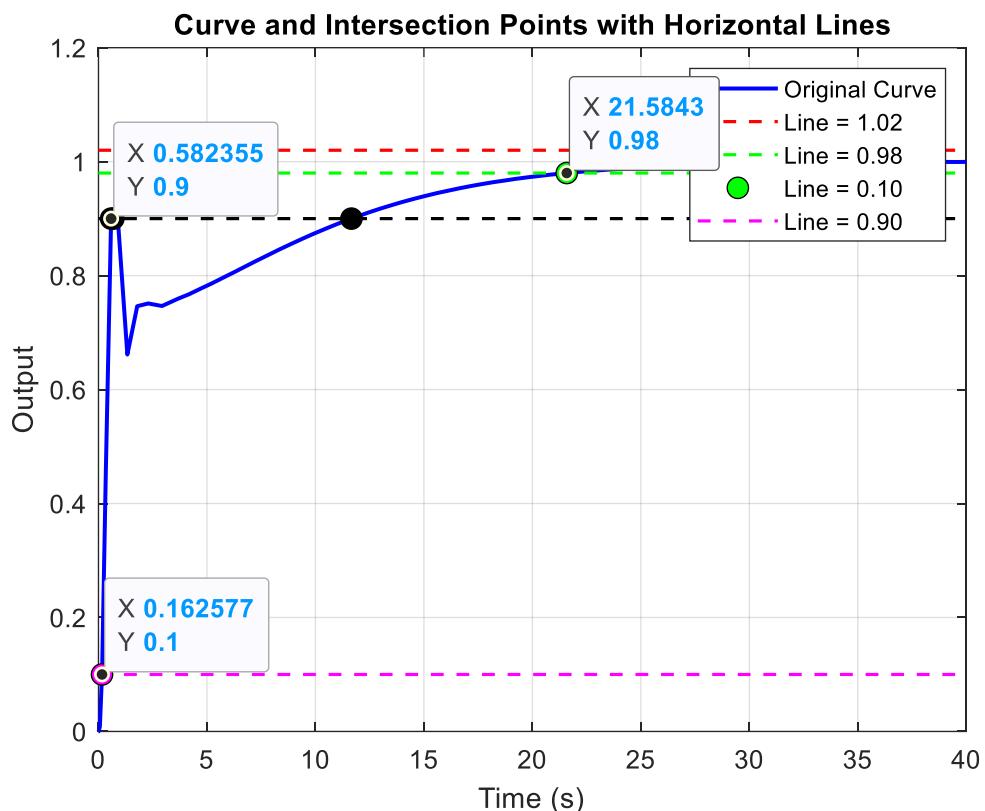
$$Tr=0.83$$

$$Ts\%2=4.32$$

overshoot%=-

d=-

IAE=3.05



در ادامه کد مربوط به استخراج مشخصات پاسخ سیستم به ورودی اغتشاش آورده می شود که مشخصات استخراج شده به صورت کامنت آورده شده و برای همه ی طراحی ها این مشخصات در جدول انتهای گزارشکار آورده می شود. نمودار مربوط به مشخصات استخراج شده برای ورودی اغتشاش(اغتشاش در ثانیه ی 7 اعمال شده است):

Ts5%=34.22

Ts10%=30.82

overshoot%=355%

d=-

IAE=47.81

```
t = out.pg1(:, 1);

y = out.pg1(:, 2);

plot(t, y, '-b', 'LineWidth', 1.5);
hold on;

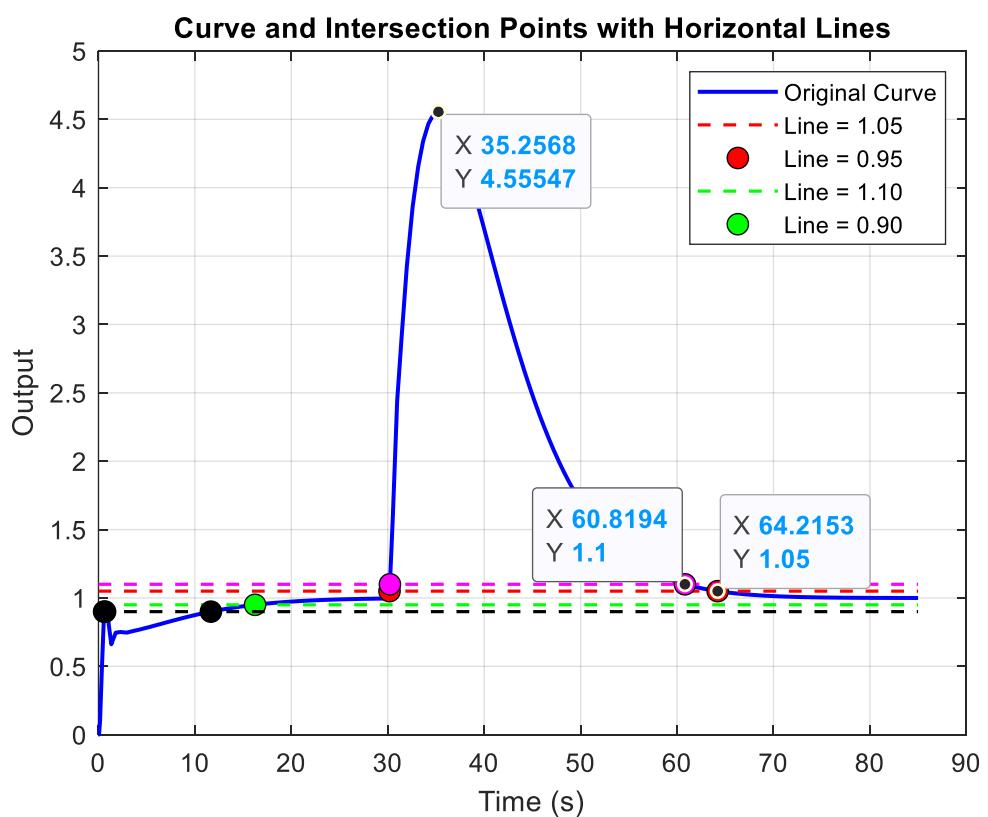
horizontal_lines = [1.05, 0.95, 1.1, 0.9];
colors = ['r', 'g', 'm', 'k'];

% finding and showing points
for i = 1:length(horizontal_lines)
    line_value = horizontal_lines(i);
    %finding points
    idx = find((y(1:end-1) - line_value) .* (y(2:end) - line_value) <= 0);

    % calculating touching points
    t_intersect = t(idx) + (t(idx+1) - t(idx)) .* ...
        (line_value - y(idx)) ./ (y(idx+1) - y(idx));
    y_intersect = line_value * ones(size(t_intersect)); % value of output in
    touching points
    %drawing lines
    plot(t, line_value * ones(size(t)), '--', 'Color', colors(i), 'LineWidth',
    1.2);

    % showing touching points
    plot(t_intersect, y_intersect, 'o', 'MarkerSize', 8, ...
        'MarkerFaceColor', colors(i), 'MarkerEdgeColor', 'k');
end

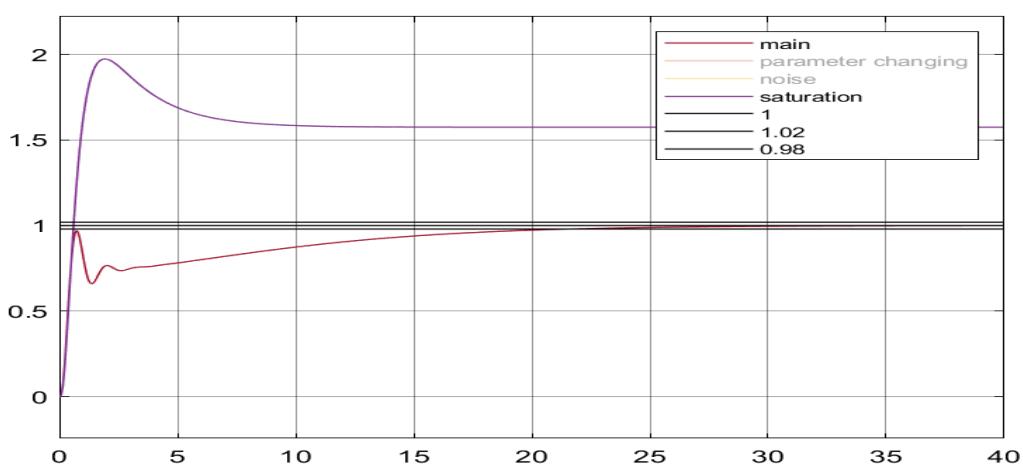
xlabel('Time (s)');
ylabel('Output');
title('Curve and Intersection Points with Horizontal Lines');
legend_entries = {[{'Original Curve'}, ...
    arrayfun(@(x) sprintf('Line = %.2f', x), horizontal_lines,
    'UniformOutput', false)]};
legend(legend_entries{:, :});
grid on;
hold off;
%calculating IAE
error=1-y;
absIntegralError = trapz(t, abs(error));
IAE=absIntegralError-3.05;
%Ts5%=34.22
%Ts10%=30.82
%overshoot%=355%
%d=-
%IAE=47.81
```



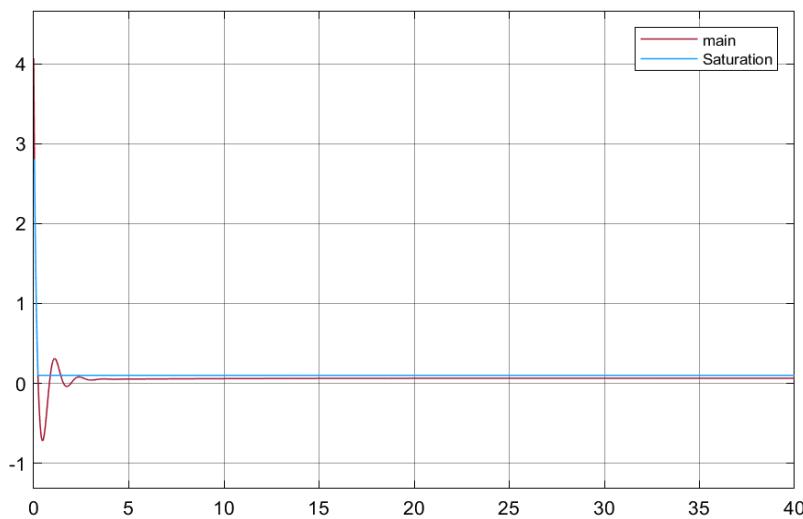
در ادامه‌ی این بخش خرجی سیستم حلقه بسته در شرایط مختلف نشان داده می‌شود(نویز ، تغییر پارامتر و...):

اشباع محرک:

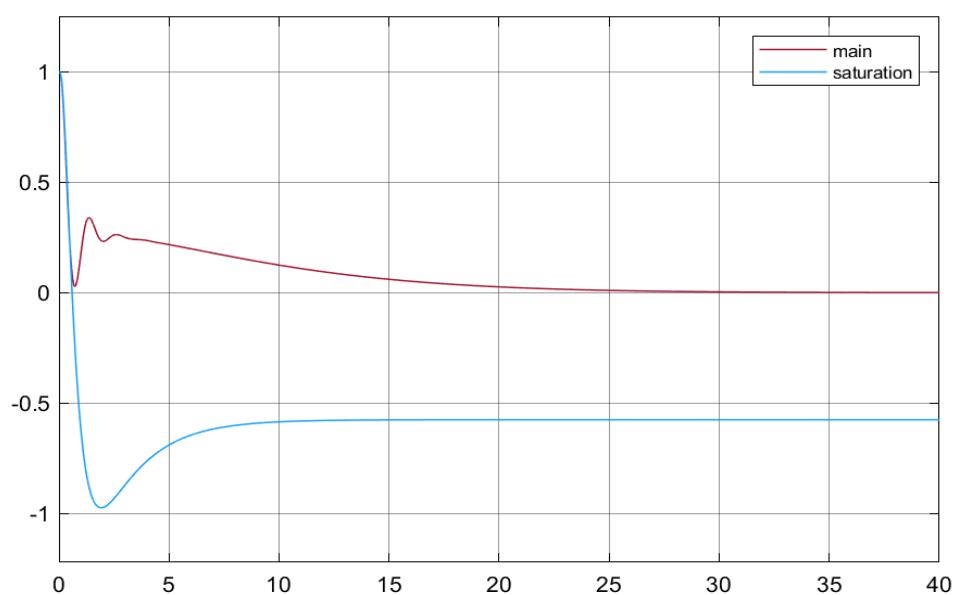
در این بخش به مقایسه‌ی خروجی‌ها بر اثر اشباع و خروجی اصلی می‌پردازیم:



همانطور که مشخص می باشد افزودن بلوک اشباع جوری که سیگنال فرمان به اشباع رود باعث می شود که سیستم از سیگنال فرمان ، فرمان نپذیرد و باعث انباشته شدن خطای صفر نمی شود). در ادامه به مقایسه ی سیگنال های فرمان میپردازیم که نشان دهنده ی نوع عملکرد بلوک اشباع می باشد.



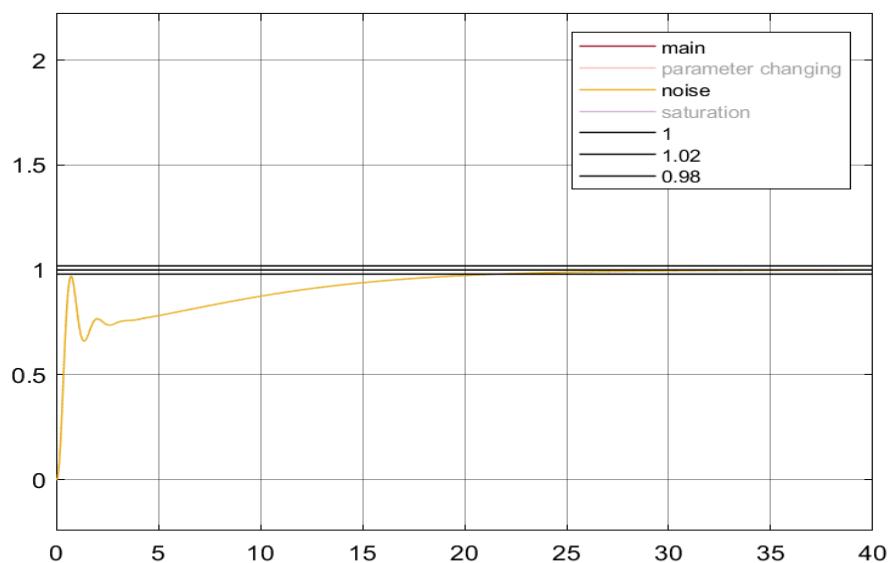
همانطور که قابل ملاحظه می باشد با توجه به بلوک اشباع سیگنال فرمان به اشباع رفته باعث انباشته شدن خطای صورت می شود ادامه با مقایسه ی سیگنال های خطای داریم:



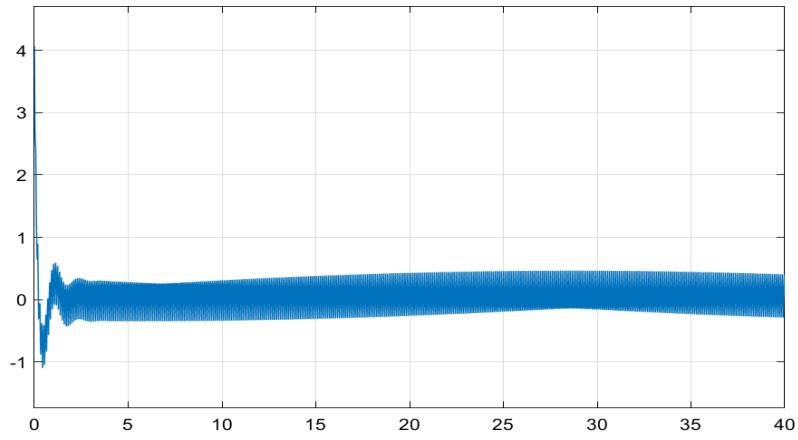
همانطور که قابل مشاهده است با مقایسه ای سیگنال های خطای میتوان متوجه شد که به دلیل وجود بلوک اشباع خطای انباسته شده صفر نمی شود.

اثر نویز:

با مقایسه ای خروجی سیستم با اثر نویز و بدون اثر نویز داریم:

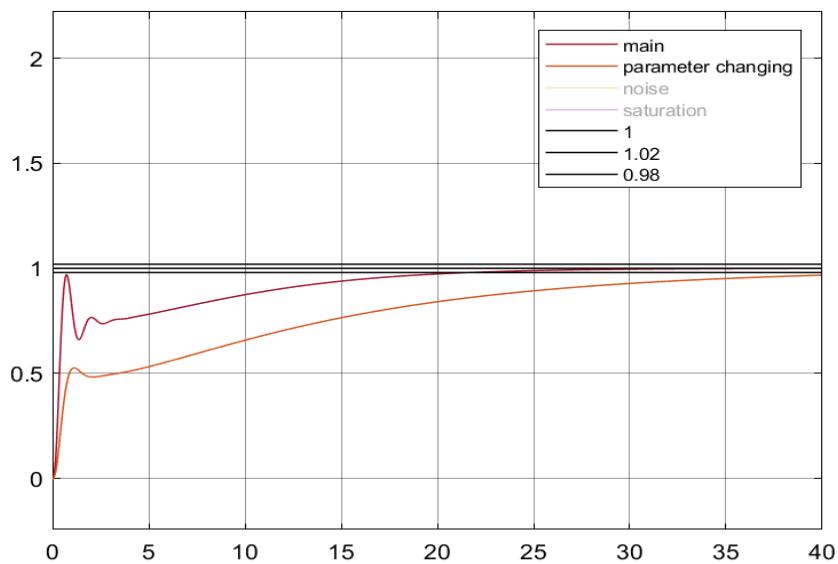


همانطور که مشخص است هر دو روی یکدیگر افتاده اند و این مورد به دلیل این می باشد که نویز فرکانس بالا می باشد و سیستم مورد نظر فرکانس پایین و نویز تاثیری ندارد روی سیستم و میتوان گفت که تاثیر بسیار کمی دارد. می دانیم که اثر نویز با بهره ای کنترل کننده رابطه مستقیم دارد و این رابطه مستقیم تاثیر زیادی روی سیگنال فرمان می گذرد(به خصوص وجود ترم مشتق گیر و استفاده فیلتر پایین گذر در آن). این مورد را می توان از روی سیگنال فرمان که در ادامه نشان داده میشود متوجه شد:



اثر عدم قطعیت:

در ادامه با مقایسه خروجی ها داریم:



تغییر پارامتر مشخصا باعث ایجاد تغییر در پاسخ گذرا که افزایش زمان خیزش و با از بین رفتن فراجهش و همچنین افزایش زمان نشست می شود ولی با وجود فیدبک و کنترل کننده به حالت ماندگار می رسد.

بخش چهارم

بررسی و مقایسه

4-1 بررسی کلی تاثیرات عوامل خارجی بر پاسخ

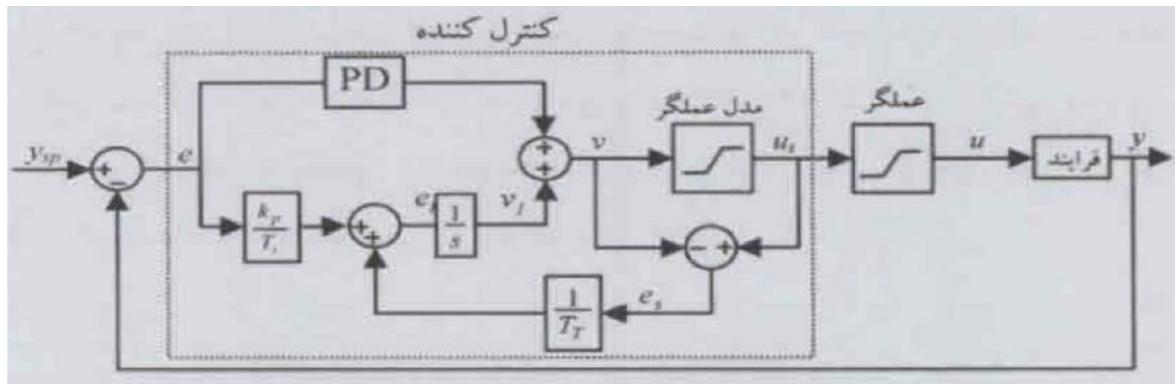
نویز:

با توجه به خروجی های سیستم که در بخش های قبل نشان داده شد کاملاً واضح بود به دلیل اینکه نویز فرکانس بالا می باشد و سیستم خود فرکانس پایین می باشد ، نویز تاثیر چندانی بر روی پاسخ خروجی ندارد مگر اینکه توسط کنترل کننده آنچنان تقویت شود که بر روی خروجی تاثیر بگذارد که تقویت فرکانس رابطه‌ی مستقیم با بهره کنترل کننده دارد که برای PID بهره‌ی کنترل کننده در فرکانس بی نهایت برابر با $K_p(1+N)$ می باشد که از خروجی سیگنال‌ها فرمان همگی بخش‌ها کاملاً مشخص است که هر کدام ضریب K_p بزرگ‌تری دارند ،؟ نویز اثر بیشتری بر سیگنال فرمان می‌گذارد.

اشباع سیگنال تحریک:

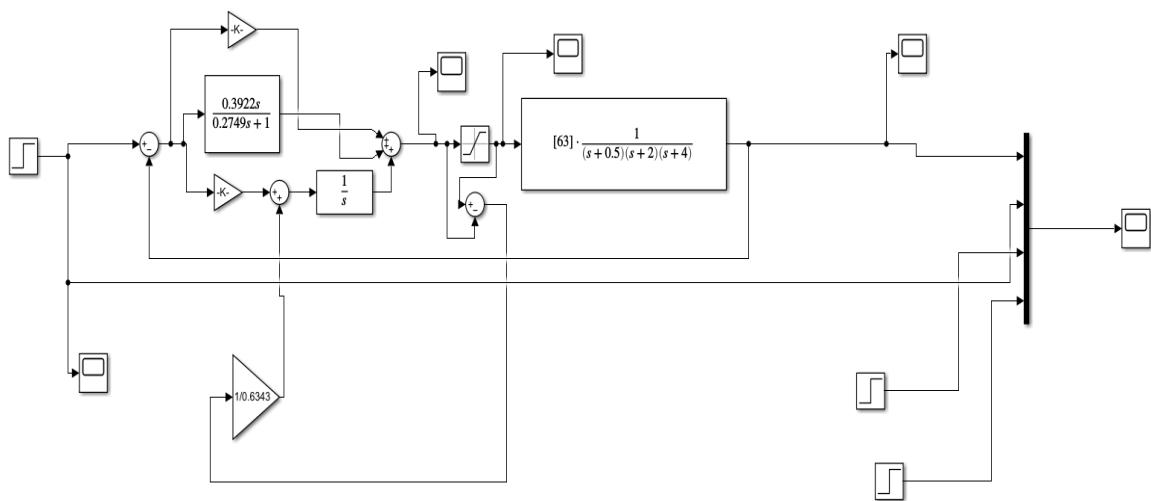
بدلیل استفاده کنترل کننده PID و وجود ترم انتگرال گیر به نوبه‌ی خود کار را بسیار خراب می‌کند زمانی که سیگنال فرمان به اشباع می‌رود. اینگونه که باعث جمع خطاهای شده و باعث می‌شود که سیگنال فرمان از اشباع خارج نشود و باعث می‌شود سیستم مورد کنترل دیگر از کنترل کننده فرمان نپذیرد که این نشان دهنده‌ی باز شدن حلقه‌ی فیدبک می‌باشد. برای رفع این مشکل از عمل integral anti windup با استفاده شود. برای زیر بخش آخر بخش سوم اینکار انجام شده و نتیجه به اینگونه شده است. با مقایسه با استفاده نکردن از این عمل:

با پیاده سازی چنین ساختاری برای عمل integral anti windup با:

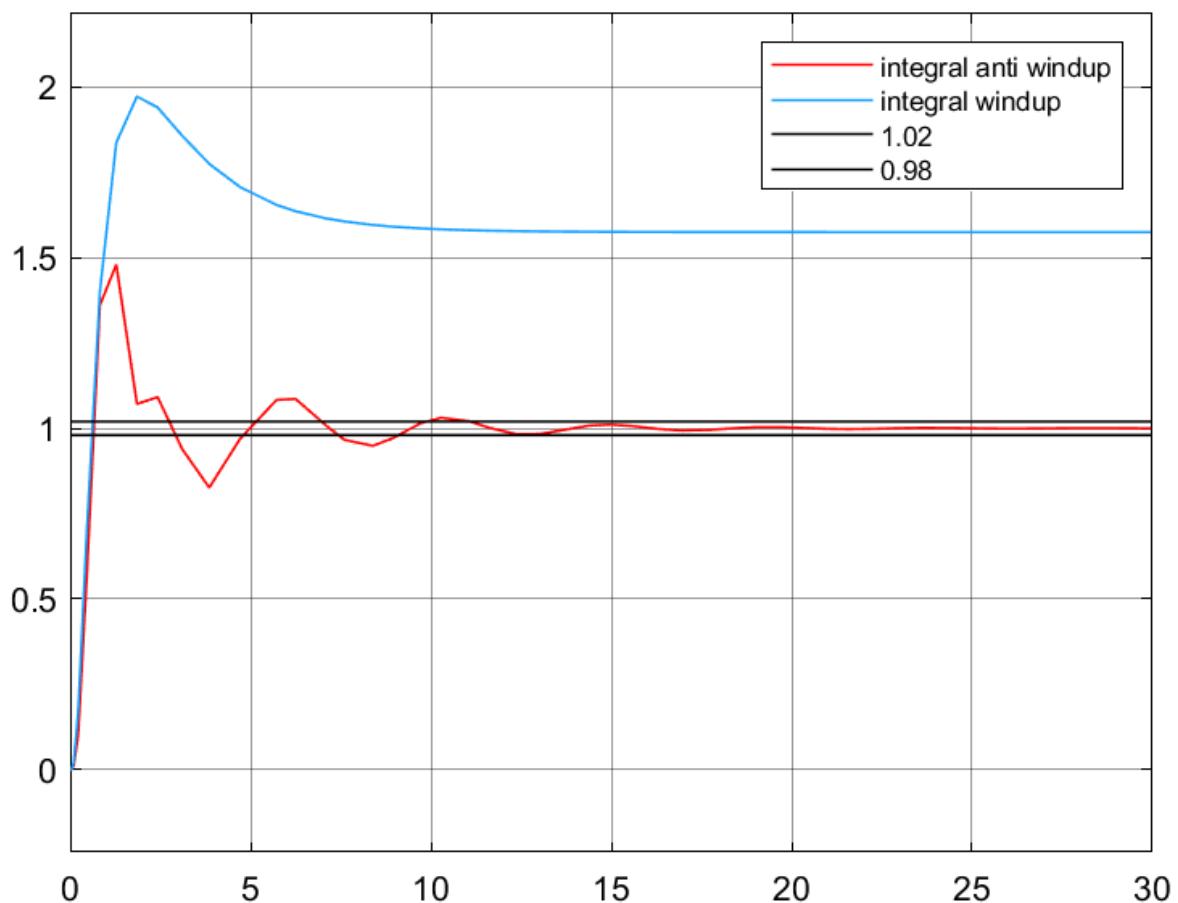


که مقدار T_t از رابطه i $T_t = \sqrt{T_d T_i}$ داریم: $T_t = 0.6343$ همانطور که در ابتدا گفته شد این عمل برای بخش 3-3 در حال انجام است(طراحی کننده با دست یابی به حد بهره 6 دسی بل و حد فاز 45 درجه)

که برای ساختار کلی داریم:



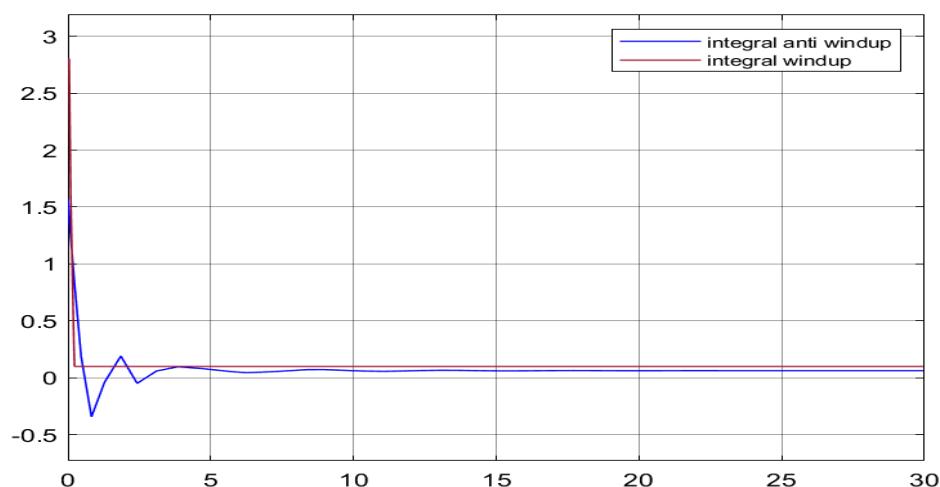
مقایسه ای خروجی با استفاده از عمل integral anti windup و بدون استفاده از این عمل:



Offset=0

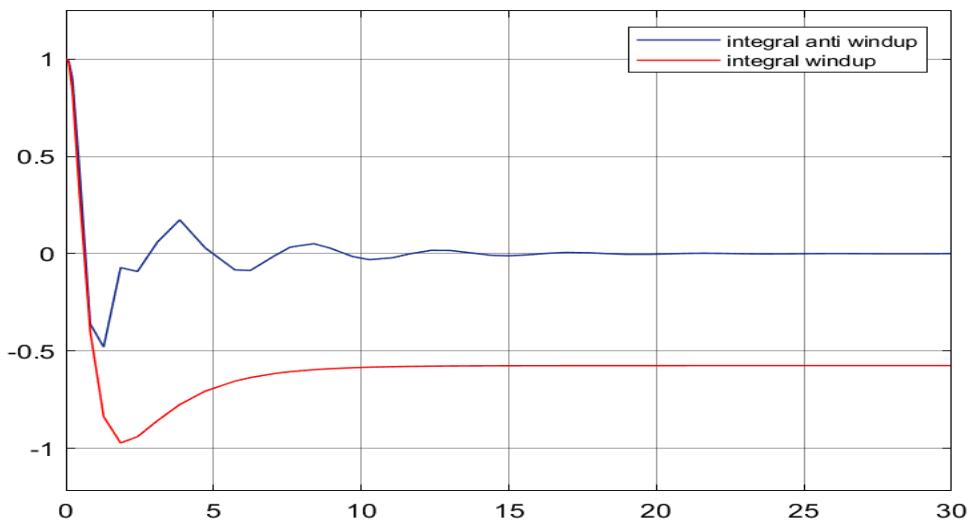
کاملاً قابل مشاهده است که عمل integral anti windup باعث به اشباع نرفتن سیگنال کنترل شده و خروجی به حالت ماندگار خود می رسد.

مقایسه‌ی سیگنال‌های فرمان:



با مقایسه‌ی سیگنال‌های فرمان هم می‌توان به تلاش عمل integral anti windup برای به اشباع نرفتن سیگنال کنترل را نیز متوجه شد.

سیگنال‌های خطأ:



عدم قطعیت:

با توجه به خروجی‌های بخش‌های قبل کاملاً قابل مشاهده بود که عدم قطعیت پاسخ گذرا را متفاوت می‌کرد ولی به حالت دام می‌رسید(به دلیل حلقه بسته بودن و استفاده از کنترل کننده) این متفاوت بودن پاسخ گذرا با نحوه‌ی تعیین مقادیر کنترل کننده (حد فاز و حد بهره) کاملاً رابطه دارد که در قسمت بعد به آن پرداخته می‌شود.

اغتشاش:

حذف اغتشاش مهم ترین مورد در سیستم‌های کنترلی است که حذف سریع تر آن و کم کردن اثر آن جزو اهداف کنترلی می‌باشد. در بخش‌های قبل با توجه به پاسخ‌های مشاهده شده در اثر ورودی اغتشاش هر کدام سرعت متفاوتی برای حف اغتشاش داشتند، همگی به دلیل وجود انتگرال گیر اغتشاش را حذف می‌کردند ولی مهم آن است که کدام فرجهش کم تر در اثر اعمال اغتشاش و سرعت بیشتر برای رسیدن به حالت ماندگار داشته است. که در قسمت بعد به آن پرداخته می‌شود.

4-2 تشکیل جدول مقایسه و بررسی تاثیر حد فاز و حد بهره(پاسخ پله، اغتشاش پله)

پاسخ پله:

	Tr	Ts(%2)	Overshoot(%)	d	IAE	Gain margin(dB)	Phase margin(deg)
ZN-ol	0.6	6.1	43.7	0.08	1.2990	25.87	42.05
CC	0.6	9.85	57.38	0.34	1.9798	25.59	35.47
CHR-ser	1.2	5.16	9.8	-	1.1537	31.6	16.89
CHR-reg	0.7	10.58	55	0.22	2.0213	28.92	44.94
ZN-cl	0.4	4	45.9	0.08	0.8461	19.5548	35.1161
ZN-cl-weighted	2.2	2.78	-	-	1.1838	26.51	78
ZN-dmp	0.4	3.53	43.18	0.14	0.8309	20.39	22.64
PR	0.4	2.32	24.6	0.08	0.6199	-2.34	4.63
Gain margin(6dB)	0.6	5.91	41.25	0.14	1.1894	6	3.52
Phase margin(45 deg)	0.4	2.99	54	0.04	0.9259	19.39	45
Gain(6dB) & phase(45deg)	0.8	4.32	-	-	3.05	7.4541	37.83

ورودی اغتشاش:

	Ts5%	Ts10%	Overshoot%	d	IAE	Gain margin(dB)	Phase margin(deg)
ZN-ol	7.5	6.03	228	0.08	4.4827	25.87	42.05
CC	10.89	9.4	227	0.27	5.34	25.59	35.47
CHR-ser	6.16	5.94	382	-	11.08	31.6	16.89
CHR-reg	12.25	10.25	272	0.23	7.44	28.92	44.94
ZN-cl	4.1	3.09	139	0.05	1.78	19.5548	35.1161
ZN-cl-weighted	2.4	2.3	143.5	-	1.7669	26.51	78
ZN-dmp	4.2	3.95	157	-	2.43	20.39	22.64
PR	10.7	8.86	163	-	5.55	-2.34	4.63
Gain margin(6dB)	5.63	5.34	247	0.13	4.26	6	3.52
Phase margin(45 deg)	3.9	3.29	151	0.01	1.96	19.39	45
Gain(6dB) & phase(45deg)	34.22	30.82	355	-	47.81	7.4541	37.83

در حالت کلی بهترین موارد برای ما زمان خیزش ، زمان نشت ، ماکریم فراجهش ، نسبت افت و IAE کمتر است.

با مقایسه‌ی کلی که از مشخصات سیستم (هم پاسخ پله و هم اغتشاش پله) با کم بودن زمان خیزش و زمان نشست مقدار IAE کاهش پیدا می‌کند. حد فاز و حد بهره ارتباط مستقیمی با این مشخصات دارند. به گونه‌ای که داریم:

حد فاز:

- (1) هرچه حد فاز زیاد باشد، پایداری سیستم بیشتر می‌باشد یعنی نوسانات هم کمتر می‌باشد و بالعکس.
- (2) هرچه حد فاز زیاد باشد، زمان نشست کم می‌باشد و بالعکس.
- (3) هرچه حد فاز زیاد باشد زمان خیزش کاهش می‌یابد و بالعکس.
- (4) هرچه حد فاز زیاد باشد، فراجهش کاهش می‌یابد و بالعکس.
- (5) هرچه حد فاز بیشتر شود، نسبت افت افزایش می‌یابد و بالعکس.

حد بهره:

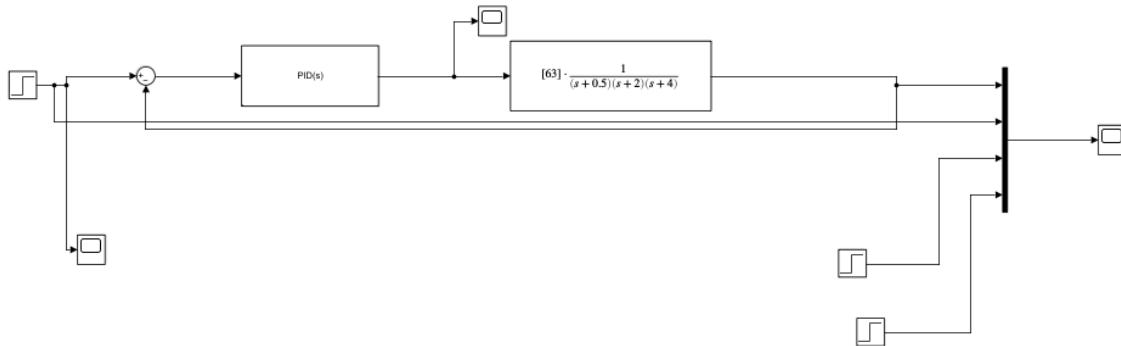
- (1) حد بهره‌ی بالا باعث افزایش پایداری می‌باشد و بالعکس.
- (2) حد بهره‌ی بالا باعث کاهش زمان خیزش می‌شود و بالعکس.
- (3) افزایش حد بهره باعث کاهش زمان نشست می‌شود و بالعکس.
- (4) با افزایش حد بهره مقدار فراجهش افزایش می‌یابد و بالعکس.
- (5) با افزایش حد بهره نسبت افت کاهش می‌یابد و بالعکس.

همانطور که از جداول مشخص می‌باشد کاملاً از موارد بالا تبعیت می‌کنند.

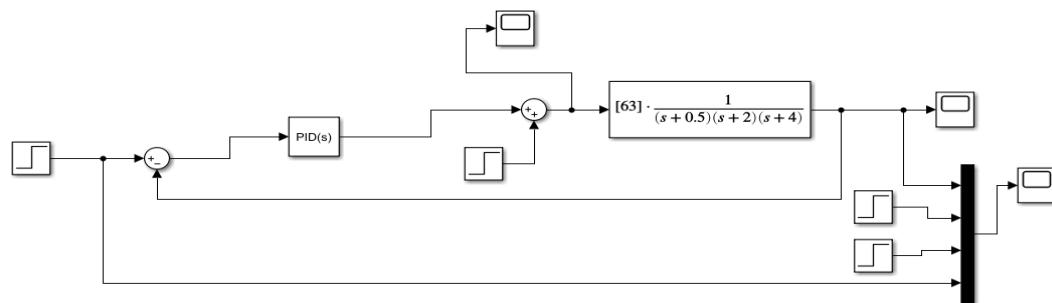
*از جداول مقایسه نمی‌توان به راحتی متوجه شد که کدام طراحی عملکرد بهتری داشته و باید طبق نیاز بین مواردی که عملکرد خوبی داشته اند انتخاب انجام داد (چه از لحاظ حذف اغتشاش و چه از لحاظ پاسخ گذاری اولیه سیستم)

3-4 نمای کلی از سیمولینک کلیه‌ی طراحی‌ها

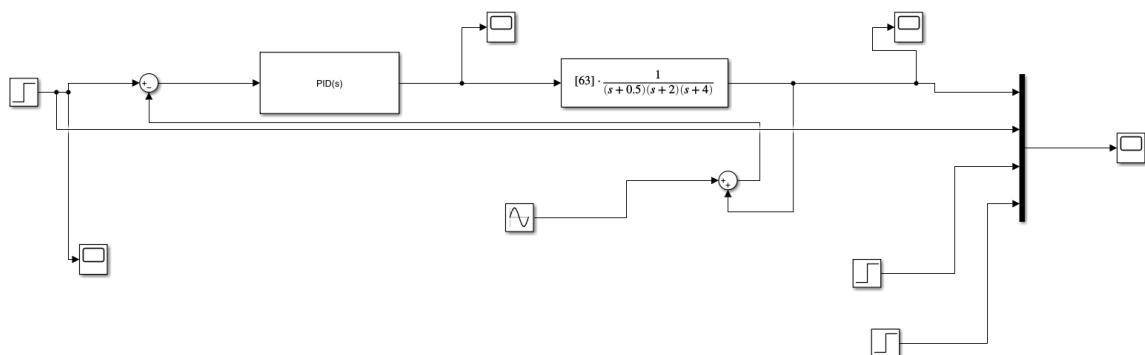
ورودی پله:



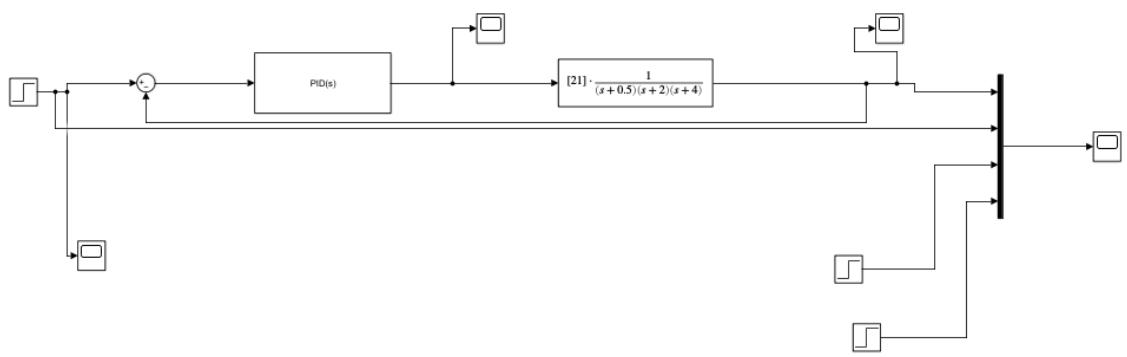
ورودی اغتشاش:



اثر نویز:



اثر تغییر پارامتر:



اثر اشباع:

