به نام نور

تمرین شماره 5 طراحی کنترلر

دانشجو:ریحانه نیکوبیان

شماره دانشجویی:99106747

سال تحصیلی:1402

1)

۱- سیستم زیر را در نظر بگیرید.

$$G_{(s)} = \frac{-s+3}{(s+1)(s+2)(s^2+2s+4)}$$

الف – این سیستم را با تابع تبدیل رسته اول همراه با تاخیر زمانی تقریب بزنید. به این منظور از تابع get_fod در دو حالت "فرکانس و بهره بحرانی" و "تابع تبدیل" و روش "بهینه سازی" به کمک تابع opt_app استفاده کنید. پاسخ پله سیستم اصلی را با توابع تبدیل تقریبی بدست آمده مقایسه کرده (رسم هر چهار نمودار در یک شکل) و بهترین تقریب را انتخاب کنید.

ب – با استفاده از نتیجه قسمت قبل، به روش‌های زیر کنترلر PID مناسب برای سیستم طراحی کنید.

ZN , refined ZN , modified ZN , CC , CC-revisited , AH , frequency based AH ,
CHR(set point, 0% & 20% overshoot) , WJC ,
Optimum PID (set point, PID & PI-D , ISTE)

برای هرکدام از کنترلرهای طراحی شده، پاسخ سیستم مدار بسته به فرمان پله واحد را رسم کرده و با یکدیگر مقایسه کنید.

الف)

با استفاده از کد q1_a و کدهای از پیش نوشته شده برای توابع get_fod، opt_app توابع را به سه تابع زیر تقریب می زنیم:

روش get_fod مبنای فرکانسی

```
                0.375
exp(-1.45*s) * ------------
               0.9587 s + 1
```

روش get_fod مبنای تابع تبدیل

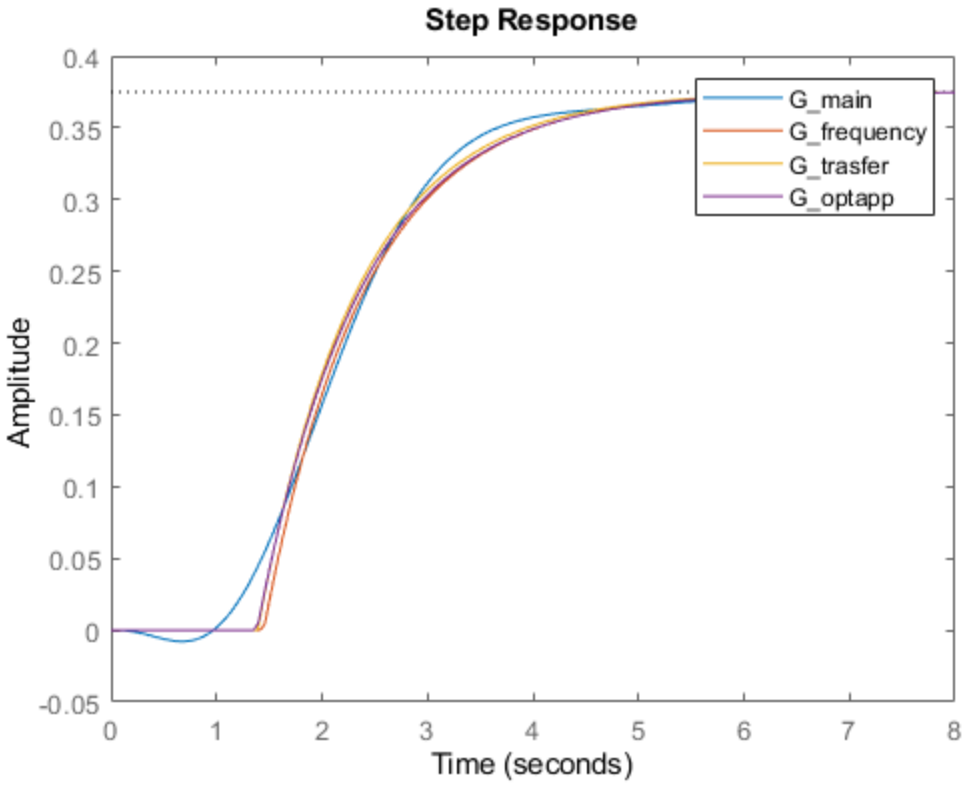
```
                0.375
exp(-1.39*s) * ------------
               0.9428 s + 1
```

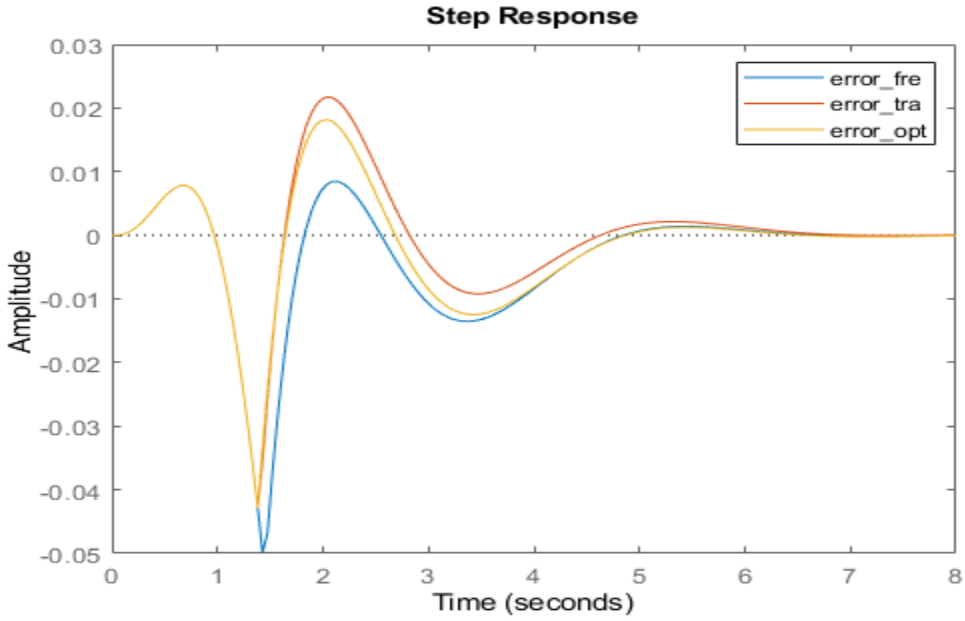
روش opt_app

```
                0.383
exp(-1.38*s) * ---------
                s + 1.021
```

تقریب زدن توابع با FOPDT

```
clc;clear;close all
s=tf('s');
G_main=(-s+3)/((s+1)*(s+2)*(s^2+2*s+4));
step(G_main);
hold on
legend 'show'

[K,L,T]=get_fod(G_main,0);  % frequency
G_frequency=K*exp(-L*s)/(T*s+1);
step(G_frequency);
hold on

[K,L,T]=get_fod(G_main,1);  % Transfer function
G_trasfer=K*exp(-L*s)/(T*s+1);
step(G_trasfer);
hold on

G_optapp=opt_app(G_main,0,1,1);   % opt_app
step(G_optapp);

figure
error_fre=G_frequency-G_main;
error_tra=G_trasfer-G_main;
error_opt=G_optapp-G_main;
step(error_fre,error_tra,error_opt);
legend 'show'
```


Step Response
G_main
G_frequency
G_trasfer
G_optapp
Amplitude
Time (seconds)
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
1
2
3
4
5
6
7
8


Step Response
error_fre
error_tra
error_opt
Amplitude
Time (seconds)
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0
1
2
3
4
5
6
7
8

تابع get_fod

```
function [K,L,T]=get_fod(G1,method)

K=real(dcgain(G1));
if nargin==2 & method==0
   [Kc,Pm,wc,wcp]=margin(G1); ikey=0; L=1.6*pi/(3*wc); T=0.5*Kc*K*L;
   if isfinite(Kc),
       x0=[L;T];
       while ikey==0,
          ww1=wc*x0(1); ww2=wc*x0(2);
          FF=[K*Kc*(cos(ww1)-ww2*sin(ww1))+1+ww2^2; sin(ww1)+ww2*cos(ww1)];
          J=[-K*Kc*wc*sin(ww1)-K*Kc*wc*ww2*cos(ww1),-K*Kc*wc*sin(ww1)+2*wc*ww2;
              wc*cos(ww1)-wc*ww2*sin(ww1), wc*cos(ww1)];
          x1=x0-inv(J)*FF;
          if norm(x1-x0)<1e-8, ikey=1; else, x0=x1; end
       end
       L=x0(1); T=x0(2);
   end
elseif nargin==2 & method==1
   [nn,dd]=tfdata(G1,'v'); [n1,d1]=tf_derv(nn,dd); [n2,d2]=tf_derv(n1,d1);
   K1=n1(end)/d1(end); K2=n2(end)/d2(end); Tar=-K1/K; T=sqrt(K2/K-Tar^2); L=Tar-T;
end
L=real(L); T=real(T);
%
function [e,f]=tf_derv(b,a)
f=conv(a,a); na=length(a); nb=length(b);
e1=conv((nb-1:-1:1).*b(1:end-1),a);
e2=conv((na-1:-1:1).*a(1:end-1),b);
maxL=max(length(e1),length(e2));
e=[zeros(1,maxL-length(e1)) e1]-[zeros(1,maxL-length(e2)) e2];
```

تابع opt_app

```
function G_r=opt_app(G_Sys,r,k,key,G0)
GS=tf(G_Sys); num=GS.num{1}; den=GS.den{1};
Td=totaldelay(GS); GS.ioDelay=0; GS.InputDelay=0; GS.OutputDelay=0;
if nargin<5
    n0=[1,1];
    for i=1:k-2
        n0=conv(n0,[1,1]);
    end
    G0=tf(n0,conv([1,1],n0));
end
beta=G0.num{1}(k+1-r:k+1);alpha=G0.den{1};Tau=1.5*Td;
x=[beta(1:r),alpha(2:k+1)];
if abs(Tau)<1e-5
    Tau=0.5;
end
dc=dcgain(GS);
if key==1
    x=[x,Tau];
end
y=opt_fun(x,GS,key,r,k,dc);
x=fminsearch('opt_fun',x,[],GS,key,r,k,dc);
alpha=[1,x(r+1:r+k)]; beta=x(1:r+1);
if key==0
    Td=0;
end
beta(r+1)=alpha(end)*dc;
if key==1
    Tau=x(end)+Td;
else
    Tau=0;
end
G_r=tf(beta,alpha,'ioDelay',Tau);
```

توضیح کد قسمت الف:

بعد از تعریف تابع اصلی ، برای تقریب زدن به روش تابع تبدیل و فرکانسی تابع get_fod که قبل تر نوشتیم فراخوانی می شود.ورودی این تابع ارگمان اول تابع اصلی و ارگان دوم روش تقریب زدن است که اگر صفر باشد به روش فرکانسی و اگر یک باشد به روش تابع تبدیل تقریب می زند. خروجی این تابع K، L، T به همان ترتیب نوشته شده است.و بعد به شکل روبرو تابع را می نویسیم:

$$G_n(s) = \frac{k\mathrm{e}^{-Ls}}{Ts+1}.$$

در قسمت بعدی برای تقریب زدن تابع به روش opt_appاین تابع را فراخوانی می کنیم که ارگمان های ورودی به ترتیب تابع تبدیل اصلی، رسته صورت، رسته مخرج، و key است که اگر صفر باشد بدون تاخیر و اگر یک باشد همراه با تاخیر تقریب می زند.این تابع ، تابع تبدیل داده شده را به تابع تبدیلی که رسته صورت و مخرج اش را مشخص کردیم تقریب زده می شود.خروجی، تابع تبدیل تقریب زده شده است.

هر 3 تقریب تقریبا به هم نزدیک و فاصله کمی با تابع تبدیل اصلی دارند.اما به نظر می رسد با توجه به اختلاف ها با سیستم اصلی تابعی که به روش فرکانسی تقریب زده شده دقت بهتری دارد. همین تابع را به عنوان مدل تقریبی سیستم انتخاب می کنیم.

G_model=

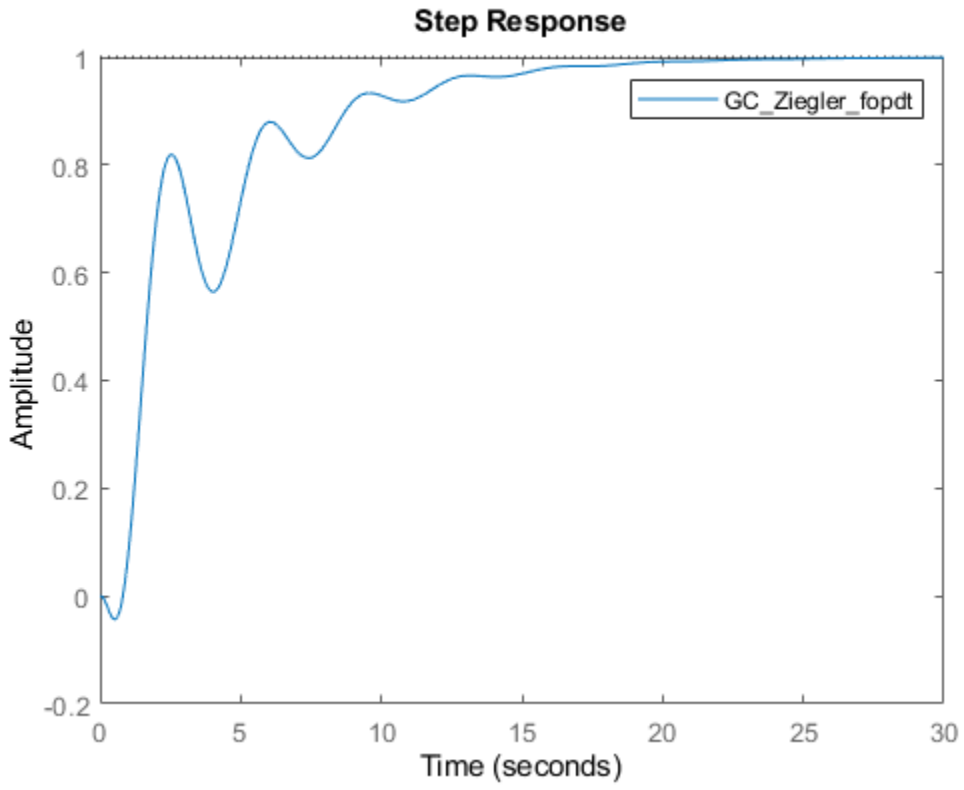
$$\exp(-1.45s) * \frac{0.375}{0.9587\,s + 1}$$

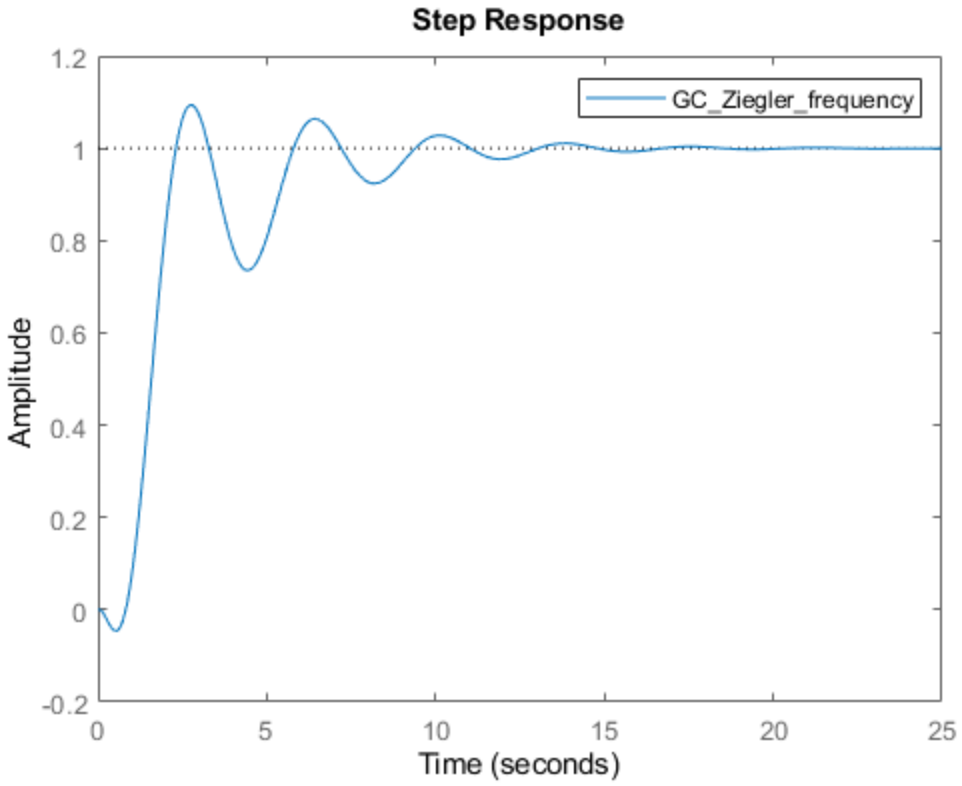
ب)کد زده شده برای سوال 2 و نتایج به شرح زیر است:

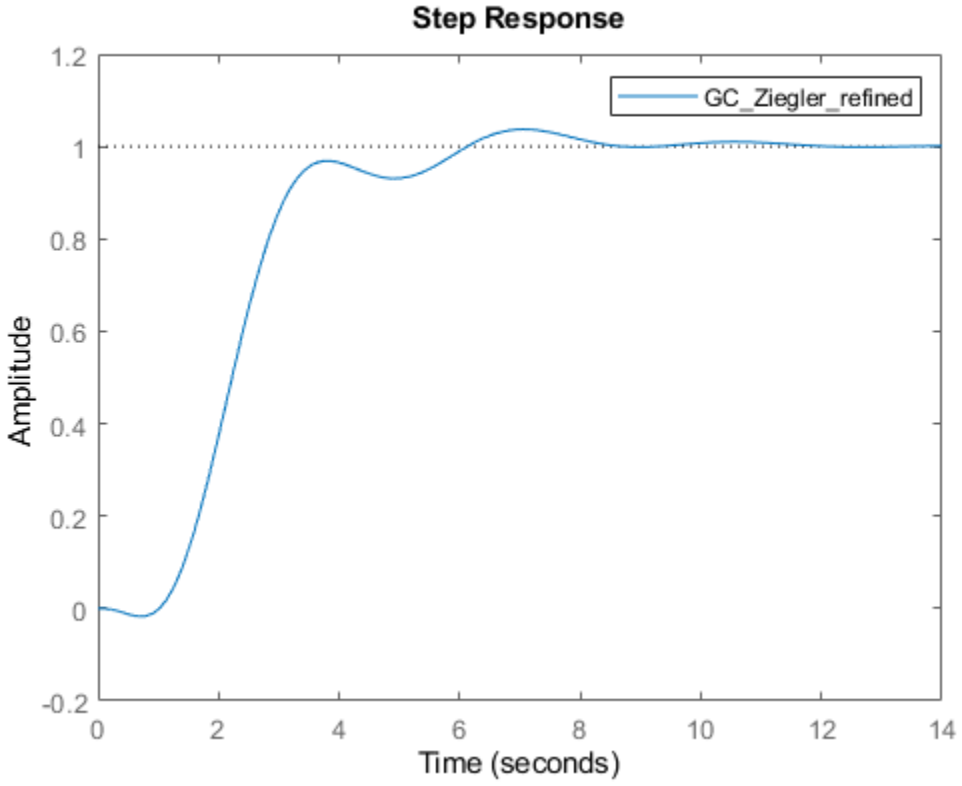
```
clc;clear;close all
s=tf('s');
G_main=(-s+3)/((s+1)*(s+2)*(s^2+2*s+4));

[K,L,T]=get_fod(G_main,0);  % frequency
G_model=K*exp(-L*s)/(T*s+1);
[Kc,~,wc,~]=margin(G_main);  %margin
Tc=2*pi/wc;
N=10;
rb=0.6;
pb=10;

[G_Ziegler_fopdt,H_Sys,Kp,Ti,Td]=ziegler_nic(3,[K,L,T,N]); %FOPDT
GC_Ziegler_fopdt=feedback(G_main*G_Ziegler_fopdt,1);
step(GC_Ziegler_fopdt);
legend 'show'
figure

[G_Ziegler_frequency,H_Sys,Kp,Ti,Td]=ziegler_nic(3,[Kc,Tc,N]); %Frequency
GC_Ziegler_frequency=feedback(G_main*G_Ziegler_frequency,1);
step(GC_Ziegler_frequency);
legend 'show'
figure


[G_Ziegler_refined,H_Sys,Kp,Ti,Td,beta]=rziegler_nic([K,L,T,N,Kc,Tc]); %Refined
GC_Ziegler_refined=feedback(G_main*G_Ziegler_refined,H_Sys);
step(GC_Ziegler_refined);
legend 'show'
figure


[G_Ziegler_modified,H_Sys,Kp,Ti,Td]=ziegler_nic(3,[Kc,Tc,rb,pb,N]); %modified
GC_Ziegler_modified=feedback(G_main*G_Ziegler_modified,1);
step(GC_Ziegler_modified);
legend 'show'
figure


[G_Cohen,H_Sys,Kp,Ti,Td]=cohen_pid(3,1,[K,L,T,N]); %cohen
GC_Cohen=feedback(G_main*G_Cohen,1);
step(GC_Cohen);
legend 'show'
figure

[G_Cohen_rev,H_Sys,Kp,Ti,Td]=cohen_pid(3,2,[K,L,T,N]); %revisited cohen
GC_Cohen_rev=feedback(G_main*G_Cohen_rev,1);
step(GC_Cohen_rev);
legend 'show'
figure
```

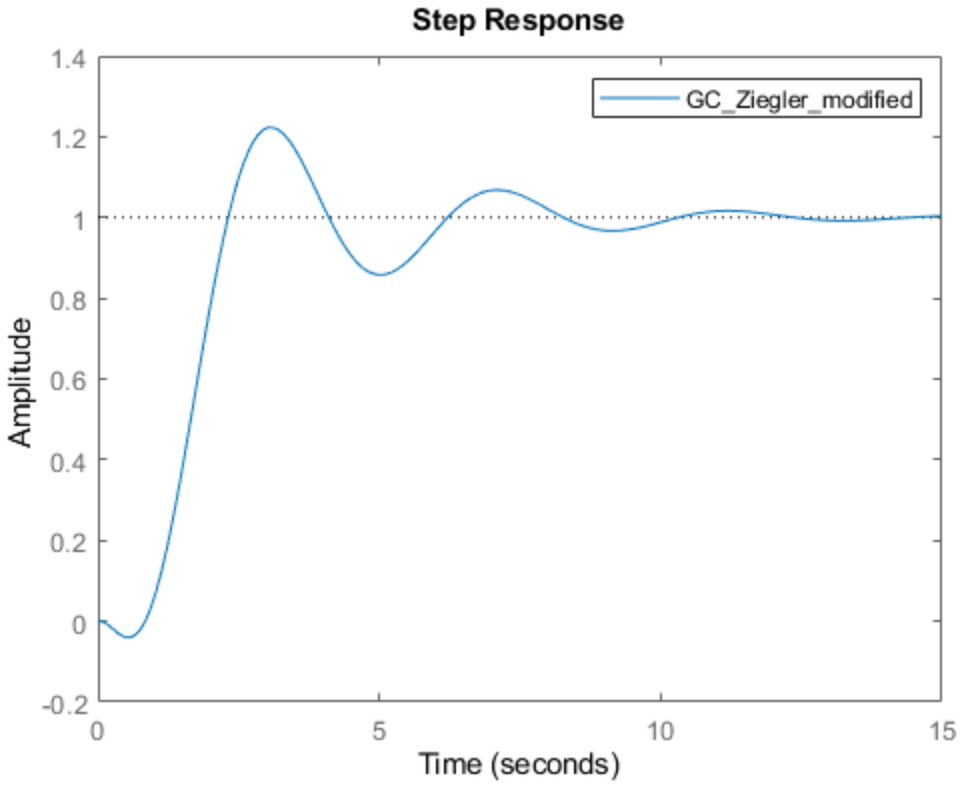
```
[G_Astrom,H_Sys,Kp,Ti,Td]=astrom_hagglund(2,1,[K,L,T,N]) ; %astrom_haglund
GC_Astrom=feedback(G_main*G_Astrom,1);
step(GC_Astrom);
legend 'show'
figure



[G_Astrom_frequency,H_Sys,Kp,Ti,Td]=astrom_hagglund(2,2,[K,Kc,Tc,N]) ; %astrom_haglund_frequency
GC_Astrom_frequency=feedback(G_main*G_Astrom_frequency,1);
step(GC_Astrom_frequency);
legend 'show'
figure


[Gc_chr,H_Sys,Kp,Ti,Td]=chr_pid(3,1,[K,L,T,N,0]); %chr_without_overshoot
GC_chr=feedback(G_main*Gc_chr,1);
step(GC_chr);
legend 'show'
figure



[Gc_chr_overshoot,H_Sys,Kp,Ti,Td]=chr_pid(3,1,[K,L,T,N,1]); %chr_with_overshoot
GC_chr_overshoot=feedback(G_main*Gc_chr_overshoot,1);
step(GC_chr_overshoot);
legend 'show'
figure


[G_wjc,Kp,Ti,Td]=wjcpid([K,L,T,N]); %wjcpid
GC_wjc=feedback(G_main*G_wjc,1);
step(GC_wjc);
legend 'show'
figure


[G_optimum, Kp, Ti, Td, H] = Optimum(3, 1, [K,L,T,N,2]) ; % optimum PID
GC_optimum=feedback(G_main*G_optimum,1);
step(GC_optimum);
legend 'show'
figure


[G_optimum_Dfeedback, Kp, Ti, Td, H] = Optimum(4, 1, [K,L,T,N,2]) ; % optimum PI_D
GC_optimum_Dfeedback=feedback(G_main*G_optimum_Dfeedback,H);
step(GC_optimum_Dfeedback);
legend 'show'
figure

[G_optimum_frequency, Kp, Ti, Td, H] = Optimum(3, 1, [K,L,T,N,Kc,Tc,K*Kc]) ; % optimum PID
frequency
```

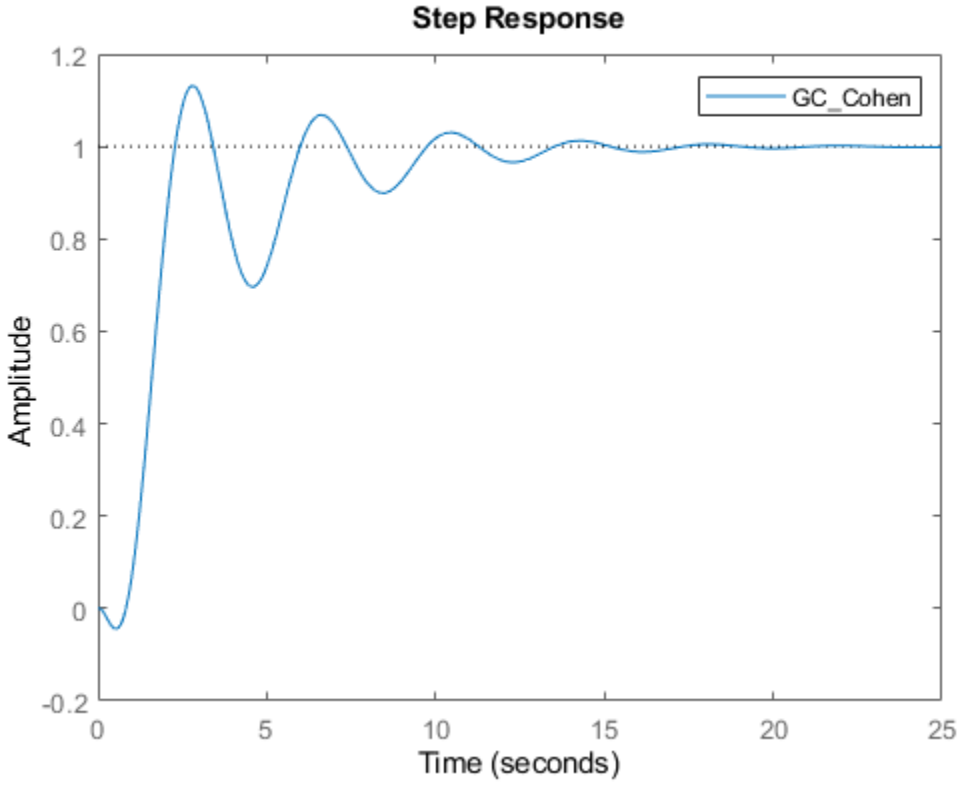
```
GC_optimum_frequency=feedback(G_main*G_optimum_frequency,1);
step(GC_optimum_frequency);
legend 'show'
figure

[G_optimum_Dfeedback_frequency, Kp, Ti, Td, H] = Optimum(4, 1, [K,L,T,N,Kc,Tc,K*Kc]) ; % optimum
PI_D frequency
GC_optimum_Dfeedback_frequency=feedback(G_main*G_optimum_Dfeedback_frequency,1);
step(GC_optimum_Dfeedback_frequency);
legend 'show'
figure


step(GC_Ziegler_fopdt,GC_Ziegler_frequency,GC_Ziegler_modified,GC_Ziegler_refined,GC_Cohen,GC_Coh
en_rev, ...
    GC_Astrom,GC_Astrom_frequency,GC_chr,GC_chr_overshoot,GC_wjc,GC_optimum,GC_optimum_Dfeedback,
...
    GC_optimum_Dfeedback_frequency,GC_optimum_frequency);
legend 'show'
```

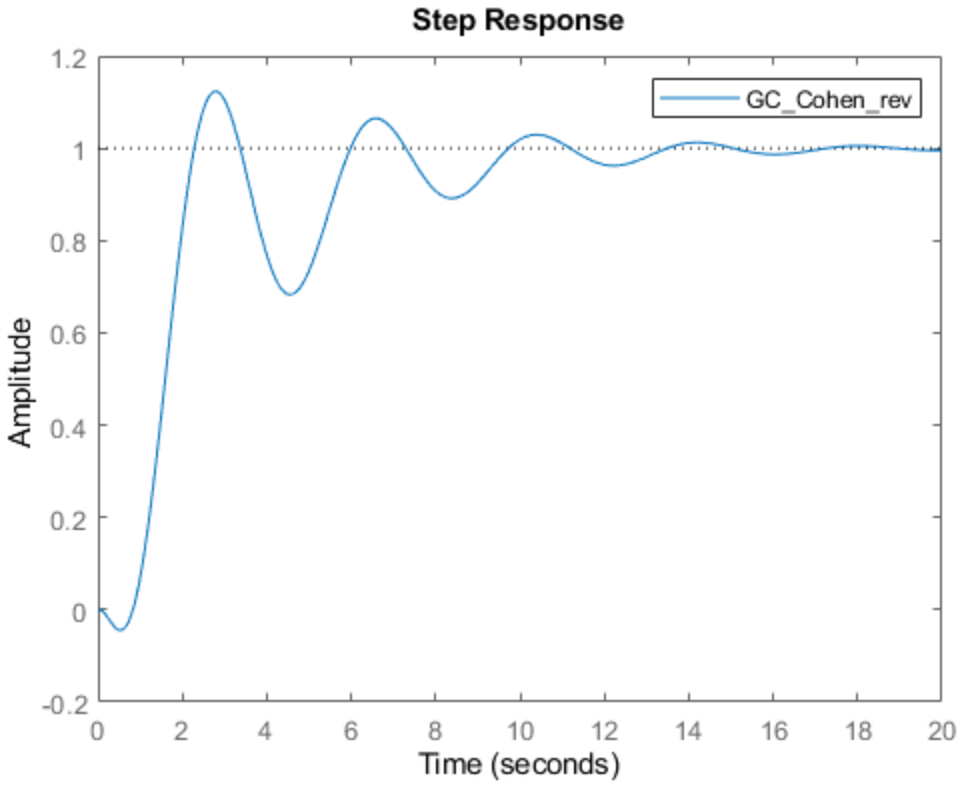


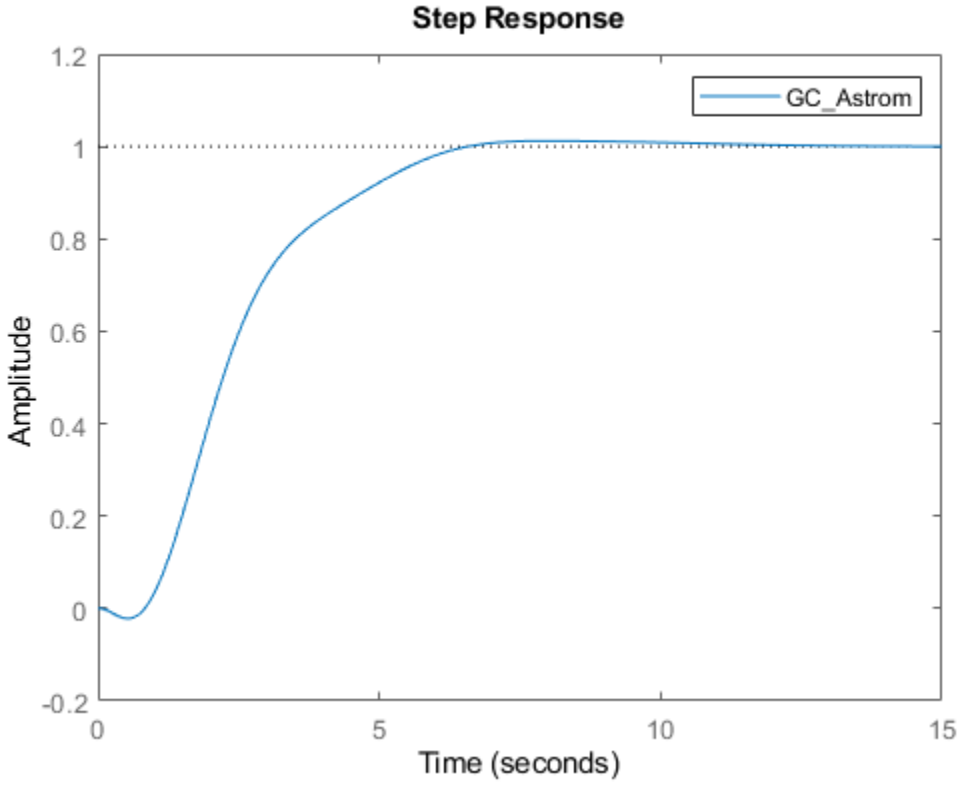

Step Response
GC_Ziegler_frequency
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25

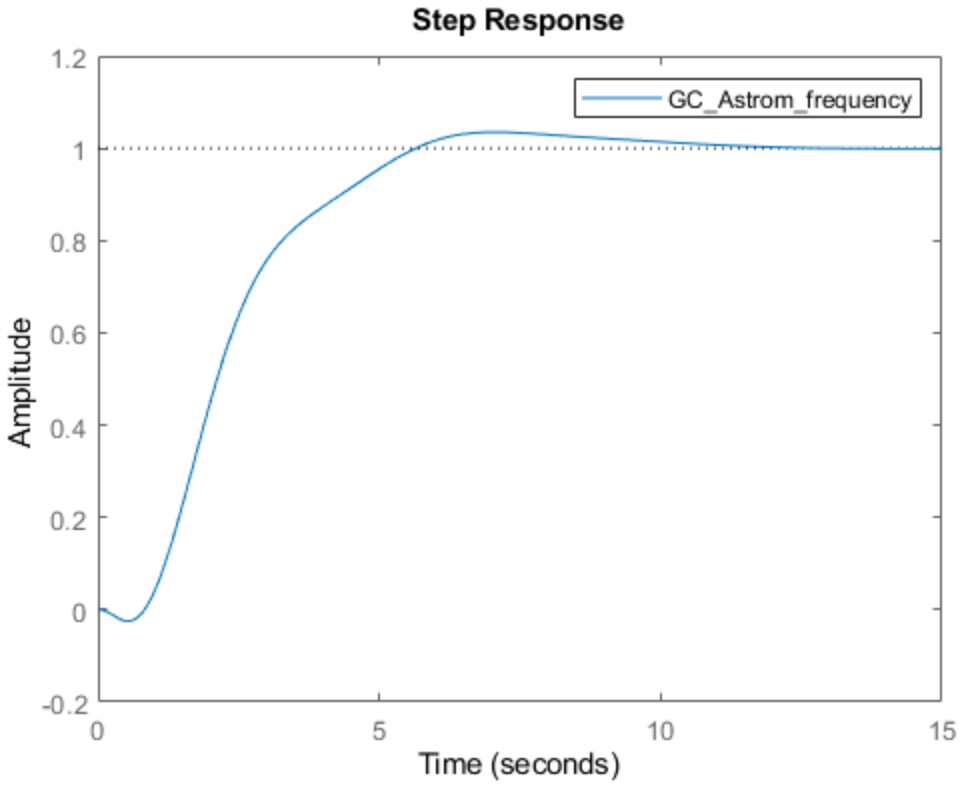

Step Response
GC_Ziegler_refined
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14

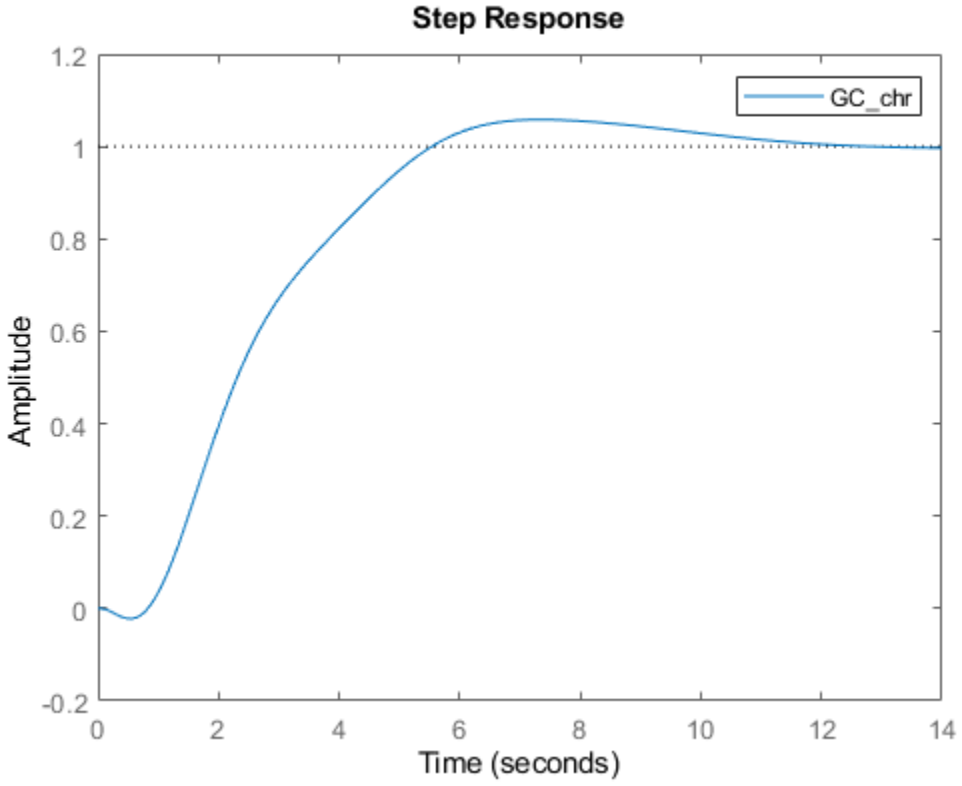

Step Response
GC_Ziegler_modified
Amplitude
Time (seconds)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15

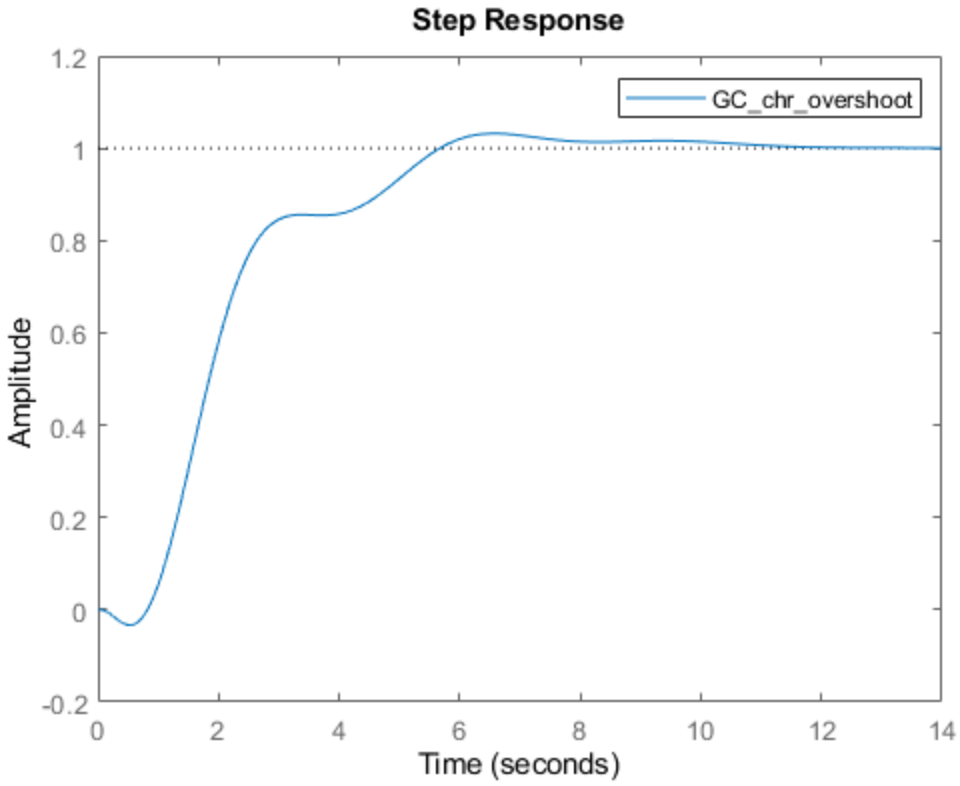

Step Response
GC_Cohen
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25

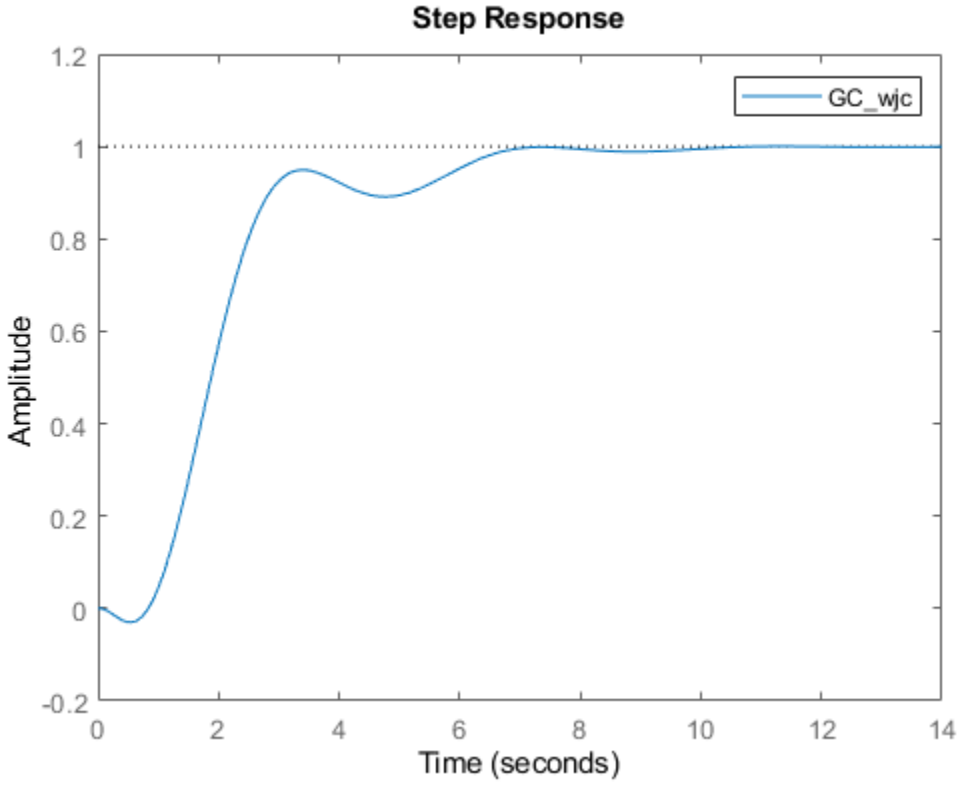

Step Response
GC_Cohen_rev
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14
16
18
20

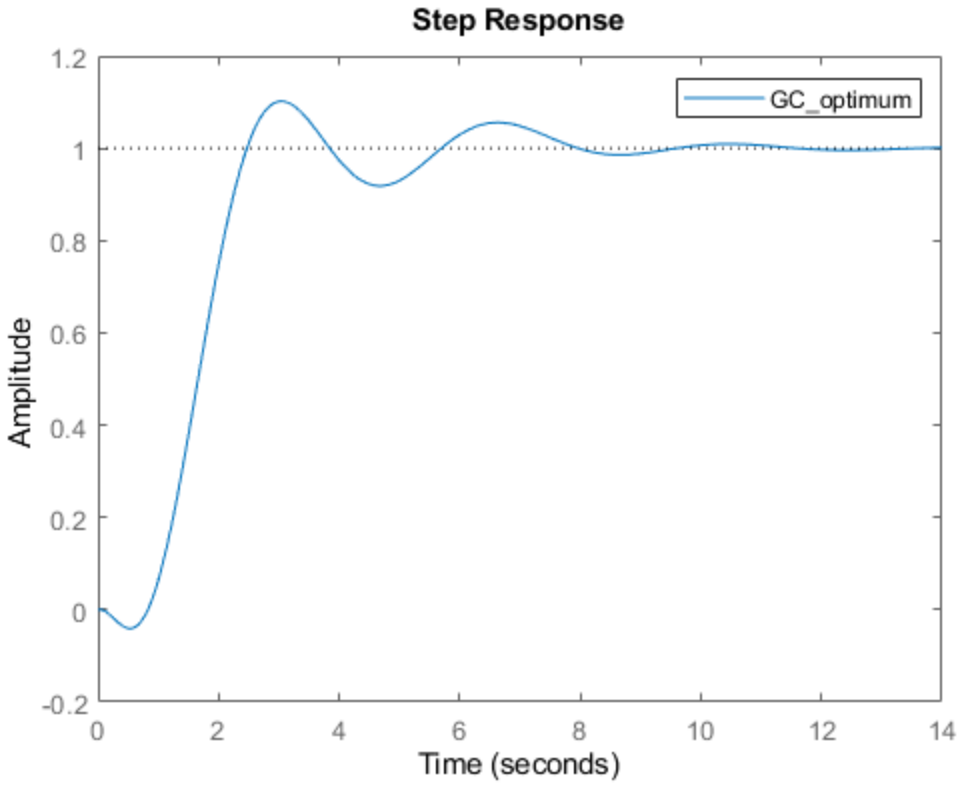

Step Response
GC_Astrom
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15

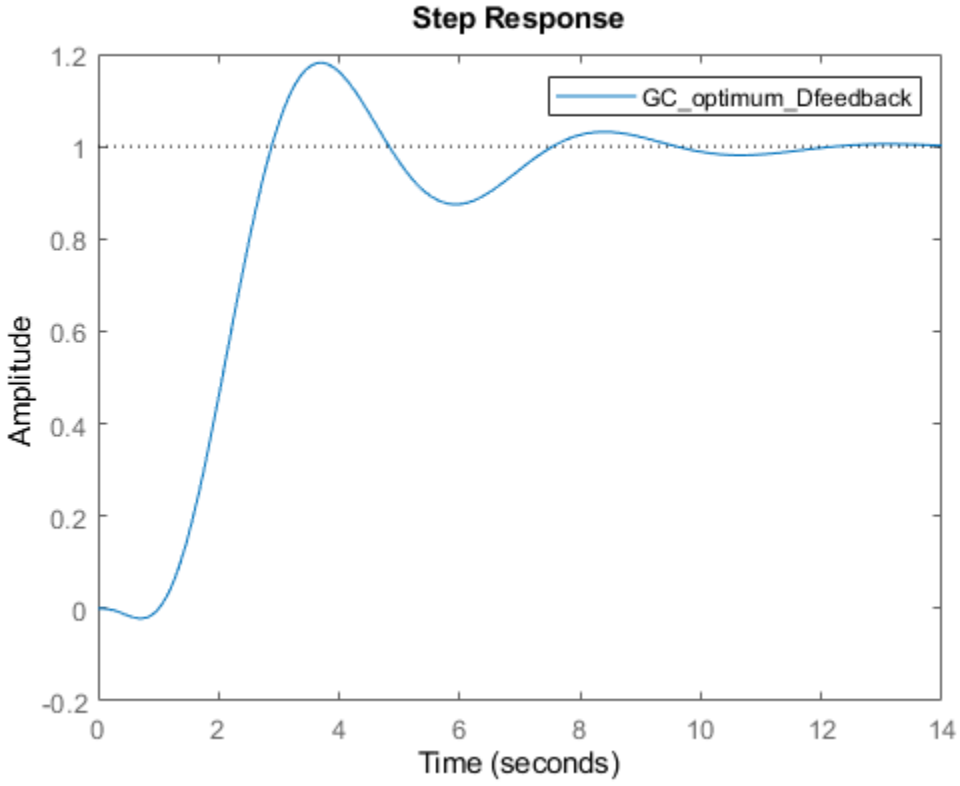

Step Response
GC_Astrom_frequency
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15

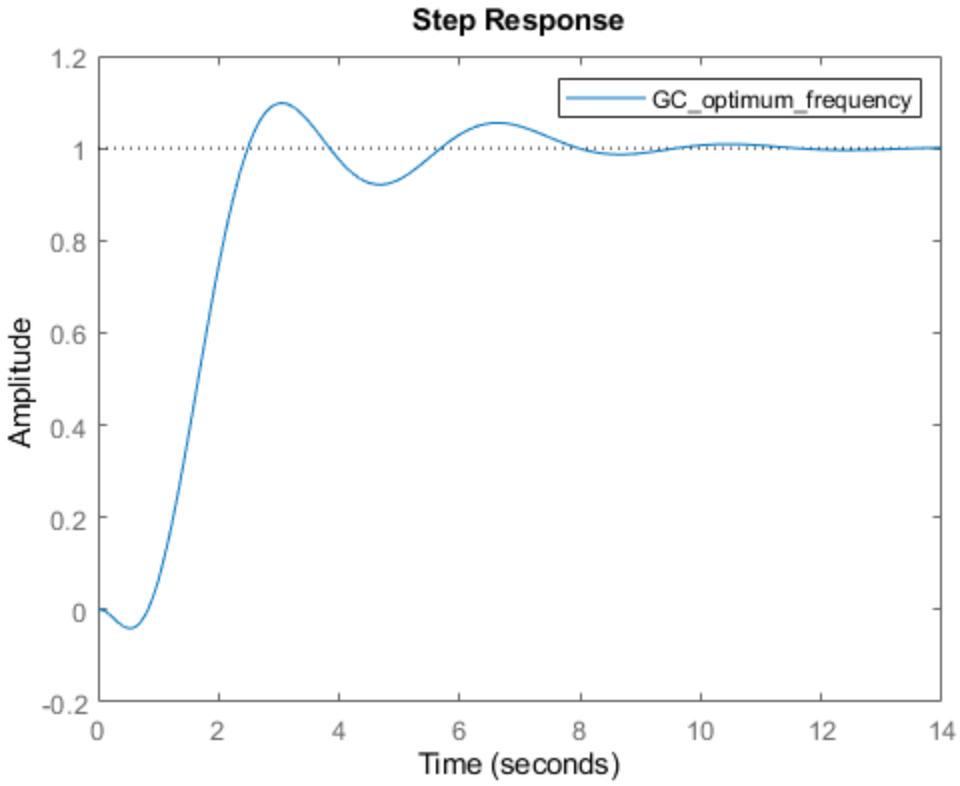

Step Response
GC_chr
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14

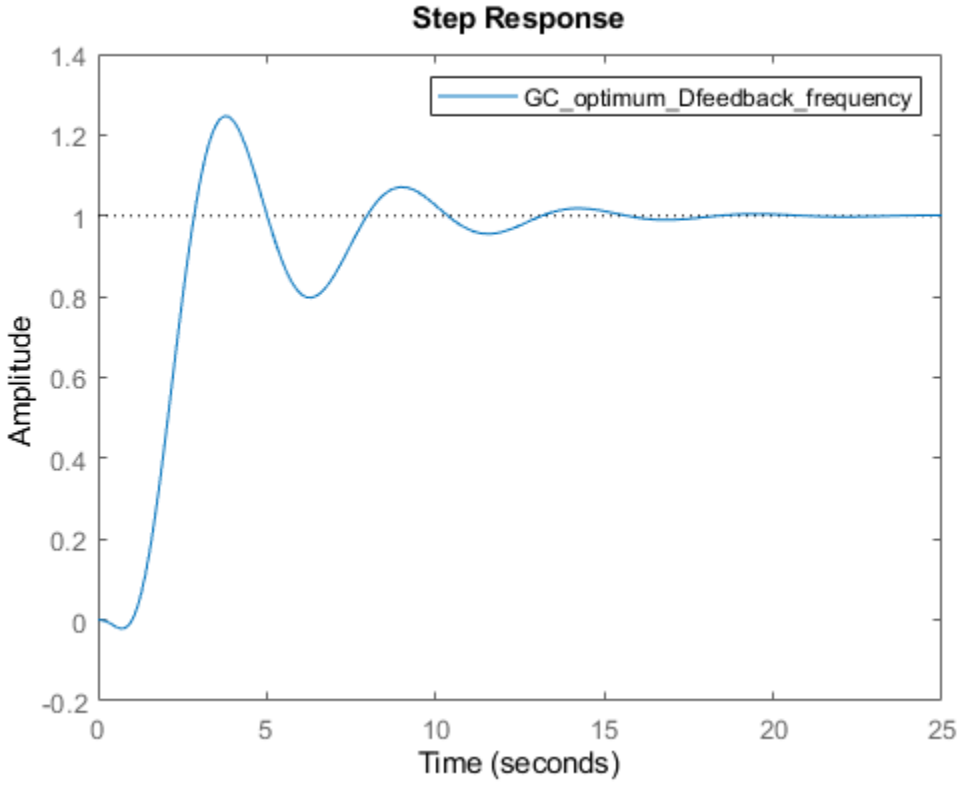

Step Response
GC_chr_overshoot
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14

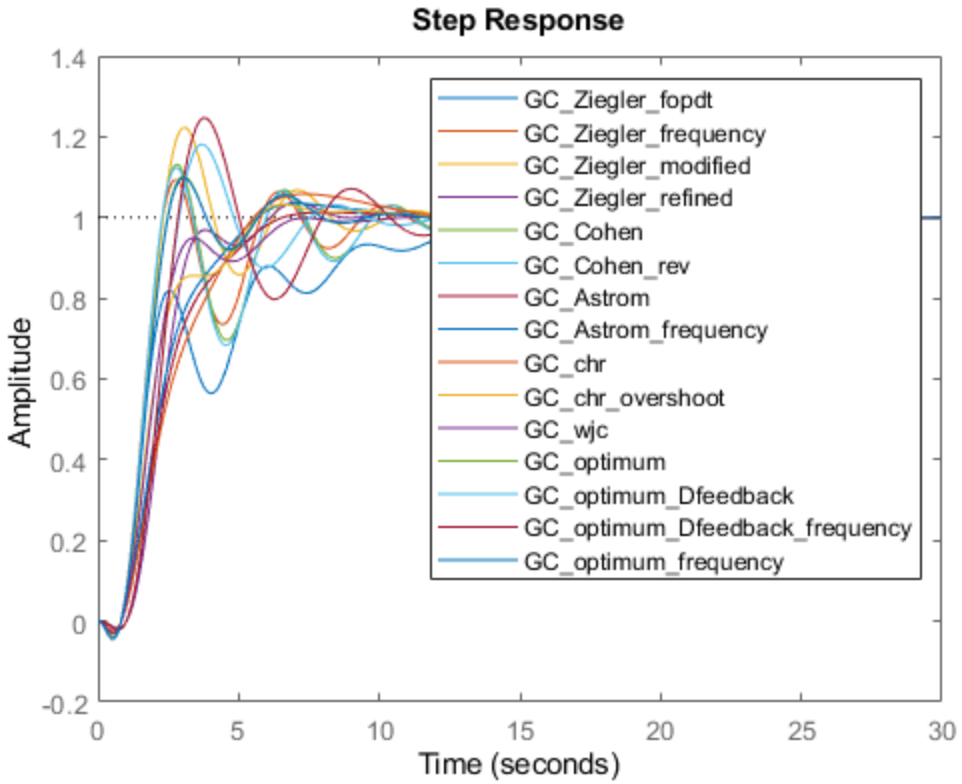

Step Response
GC_wjc
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14


Step Response
GC_optimum
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14


Step Response
GC_optimum_Dfeedback
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14


Step Response
GC_optimum_frequency
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14

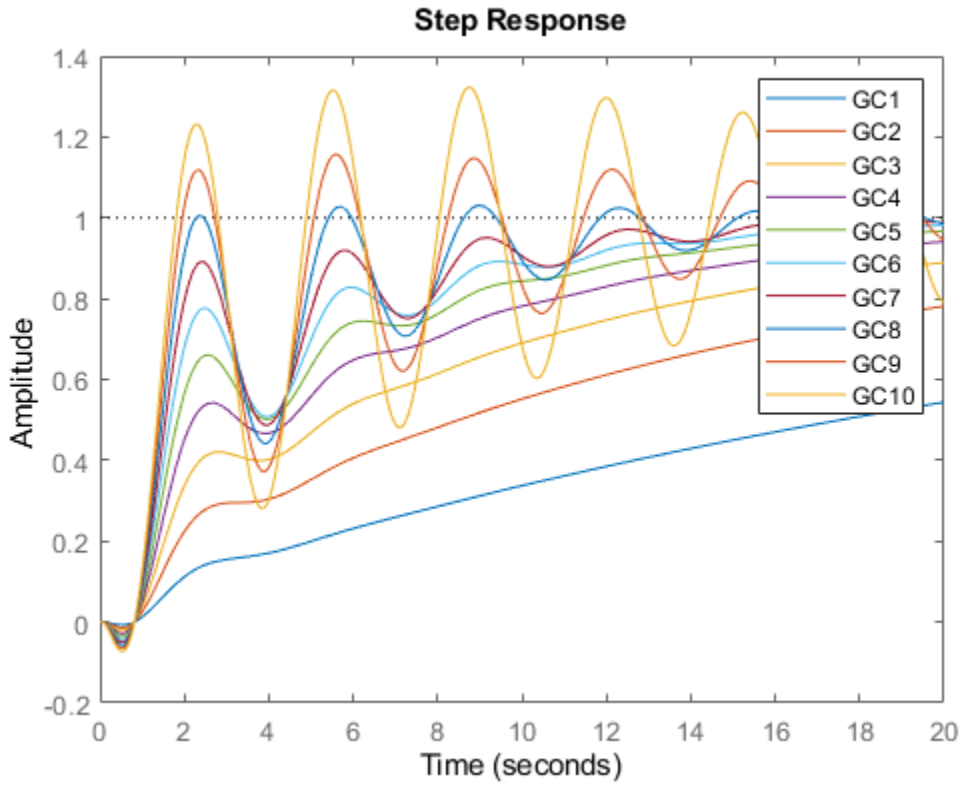

Step Response
GC_optimum_Dfeedback_frequency
Amplitude
Time (seconds)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25

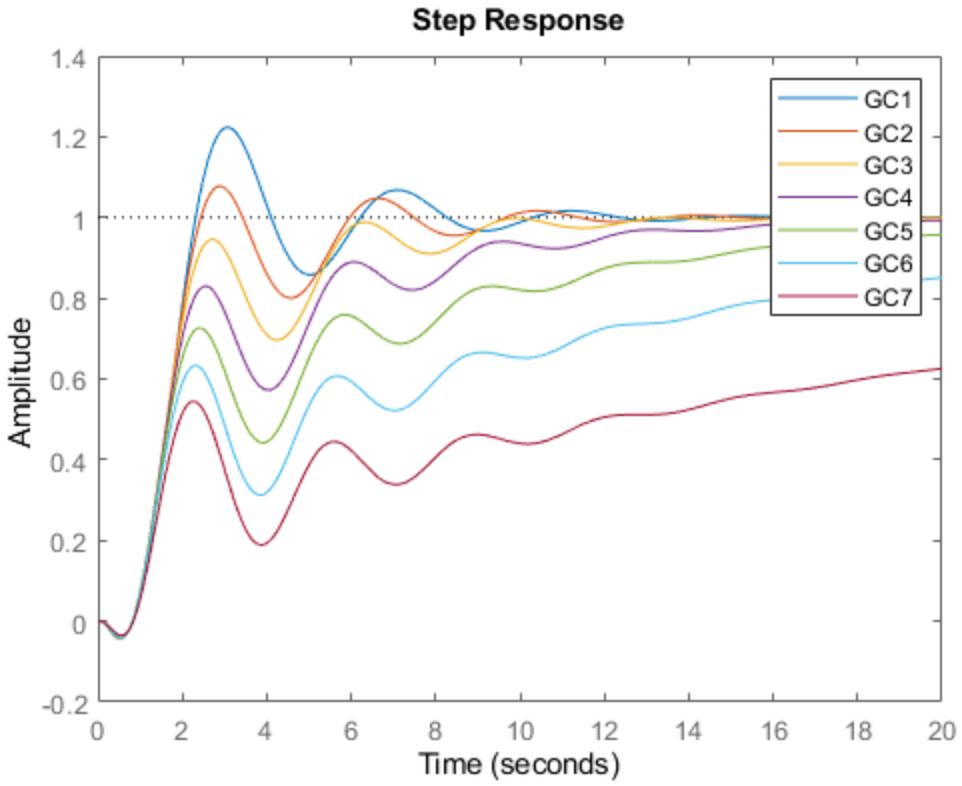


همچنین برای این سوال طراحی کنترلر refined ziegler یک کد جدا زده شده تا ضرایب rb pb به دست بیاید:

```
clc;clear;close all
s=tf('s');
G_main=(-s+3)/((s+1)*(s+2)*(s^2+2*s+4));

[K,L,T]=get_fod(G_main,0);  % frequency
G_model=K*exp(-L*s)/(T*s+1);
[Kc,~,wc,~]=margin(G_main);
Tc=2*pi/wc;
pb=45;
N=10;
legend 'show'
name={'GC1', 'GC2', 'GC3', 'GC4', 'GC5', 'GC6', 'GC7', 'GC8', 'GC9', 'GC10'};
i=1;
for rb=0.1:0.1:1

  [Gc_Sys,H_Sys,Kp,Ti,Td]=ziegler_nic(3,[Kc,Tc,rb,pb,N]);
  Gc=feedback(Gc_Sys*G_main,1);
  step(Gc,20);
  legend (name)
  hold on
  i=i+1;
end
```

```
figure
rb=0.6;
for pb=10:10:70

  [Gc_Sys,H_Sys,Kp,Ti,Td]=ziegler_nic(3,[Kc,Tc,rb,pb,N]);
  Gc=feedback(Gc_Sys*G_main,1);
  step(Gc,20);
  legend (name)
  hold on
  i=i+1;
end

% rb=0.6
%pb=10
```





Published with MATLAB® R2021b

از نمودار اول rb=0.6 را انتخاب می کنیم.بعد pbرا تغییر می دهیم از10 درجه تا 70 درجه. مطابق نمودار زاویه 10 درجه بهترین انتخاب است.اورشوت زیادی ندارد و سرعت بهتری دارد.سپس همین اعداد را در اسکریپت q2b استفاده می کنیم.

برسی کنترلرها:

همه این حالت ها چون در مسیر پیشرو انتگرال گیر دارند خطای ماندگار صفر است.همچنین همه حالت ها پایدار است.

زیگلرنیولز به روش تابع تبدیل: اورشوت ندارد، شکل پاسخ مناسب نیست و زمان نشستs15.9 است.

زیگلز نیکولز به روش فرکانسی:اورشوت 9.3%،زمان نشست 12.3s و زمان برخواست حدود 1s.شکل پاسخ بد نیست و نسبت به کنترلر قبل این کنترلر بهتر است.

Refined_ziegler_nikols:اورشوت:3.7%، زمان نشست 7.8s و زمان برخواست 1.7s است.به نظر می رسد این کنترلر نیز مناسب باشد و کاملا واضح است از دو کنترلر قبل وضعیت بهتری دارد.(سرعت بهتر و اورشوت مناسب)

Modifie_ziegler_nikols: اورشوت 22% ، زمان نشست9.78s و زمان برخواست 1.07s است.شکل پاسخ قابل قبول است اما سرعت این کنترلر نسبت به حالت قبل کمتر و اورشوت بیشتری دارد.

Cohen_cohn: اورشوت 13%، زمان نشست 13s و زمان برخواست 1ثانیه است.همانطور که از شکل پاسخ پیداست نوسانی است و این کنترلر از این نظر مناسب نیست. همچنین سرعتش کند است.

Cohen_rev:همانطور که از شکل پاسخ پیداست با حالت قبل تفاوت کمی دارد. زمان نشست و برخواست تقریبا ثابت و اورشوت 12.3% شده است.بنابراین هرچند از کنترلر قبلی بهتر شده اما همچنان مناسب نیست.

استروم هاگلند:اورشوت 1.2% زمان نشست 6s و زمان برخواست 3.46s است.شکل پاسخ هم قابل قبول است. همانطور که پیداست این کنترلر نسبت به کنترلرهای قبلی سرعت بیشتر و اورشوت پایین تری دارد و برتری این کنترلر نسبت به کنترلرهای قبلی واضح است.

استروم هاگلند به روش فرکانسی:اورشوت 3.49%، زمان نشست 9.18s و زمان برخواست 3.13s است.بنابراین استروم هاگلند به روش تابع تبدیل جواب بهتری داشت.اورشوت پایین تر و سرعت بیشتری داشت.

Chr_without overshoot:اورشوت 5.8%و زمان نشست 10.6s و زمان برخواست 3.4s است.کنترلر سرعت پایینی دارد و اورشوت هرچند قرار بود صفر باشد اما مقداری اورشوت دارد.

Chr_with overshoot: اورشوت 3.2%و زمان نشست 7.47s و زمان برخواست 3.56s است.این کنترلر نسبت به کنترلر قبلی پاسخ بهتری دارد . اورشوت کمتر و سرعت بیشتر. و با اینکه این کنترلربا اورشوت بیشتری تخمین زده شده ولی اورشوت پایین تری دارد.

Wjc:این کنترلر اورشوت ندارد و زمان نشست s6.47 و زمان برخواست 1.7s است. این کنترلر نیز سرعت خوبی دارد و نشان میدهد این روش هم یکی از روش های مناسب طراحی می تواند باشد.

Optimum_pid_transferfunction:اورشوت % 10.2 و زمان نشست 7.56s و زمان برخواست 1.18s است.این روش نسبتا سرعت مناسبی دارد و اورشوت هم قابل تحمل است و شکل پاسخ مناسبی دارد.

Optimum_pi_d_transferfunction: اورشوت 18.2% و زمان نشست s9 و زمان برخواست s1.3 است.این کنترلر نسبت به کنترلر قبل اورشوت زیادی دارد و سرعت هم کمتر شده و در کل کنترلر قبلی مناسب تر به نظر می رسد.

Optimum_pid_FREQUENCY: اورشوت 9.7%و زمان نشست 7.56Sو زمان برخواست 1.18Sاست.این کنترلر بسیار شبیه دو حالت قبل است و کمی فقط اورشوت کمتر شده پس در مقایسه با ان می شود گفت کنترلر بهتری است.

Optimum_pi_d_FREQUENCY: اورشوت 24.7% و زمان نشست 12.6S و زمان برخواست 1.3s است.این حالت اورشوت نسبتا بیشتری دارد و پاسخ کندی دارد. بنابراین مناسب نیست و در این 4 حالت بهترین حالت Optimum_pid_FREQUENCY بود.

با توجه به برسی کنترلر ها به نظر می رسد کنترلرASTROM بهترین حالت باشد. زمان نشست کمتری دارد،شکل پاسخ مناسب است و اورشوت کمی می زند.کنترلرOptimum_pid_FREQUENCY هم در درجه بعدی کنترلر خوبی است.

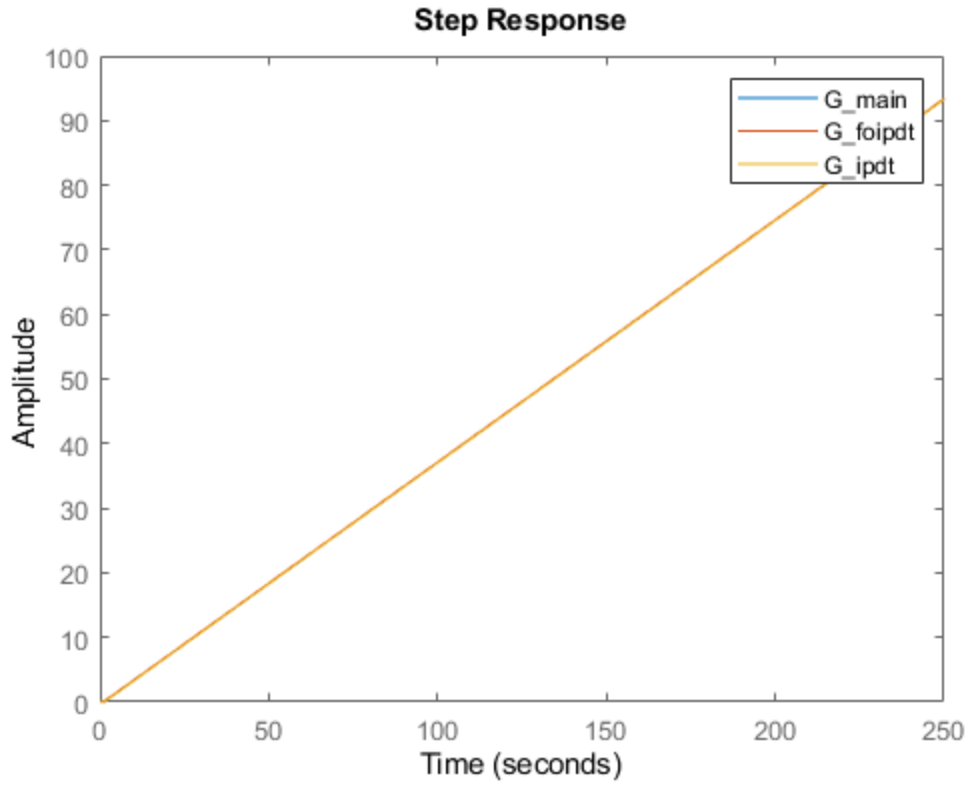
(2

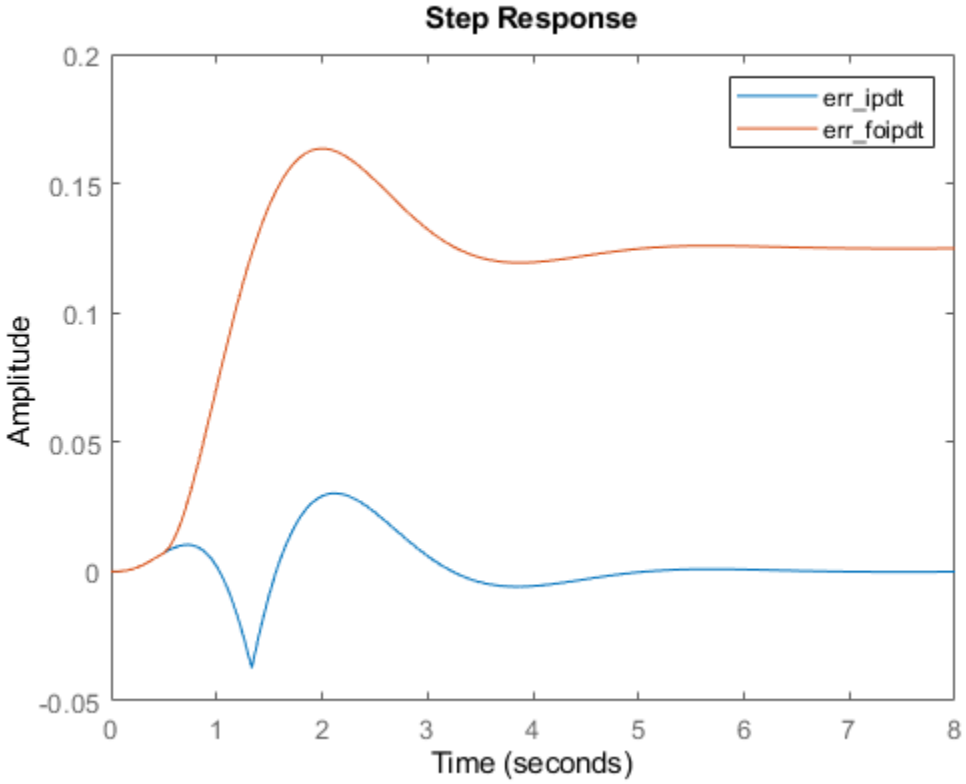
۲- برای سیستم زیر به روش‌های گفته شده در درس، کنترلر PID و PD طراحی کرده و برای هر کدام پاسخ سیستم مدار بسته به فرمان پله واحد را رسم کنید و نتایج را با یکدیگر مقایسه نمایید.

$$G_{(s)} = \frac{-s+3}{s(s+2)(s^2+2s+4)}$$

چون تابع تبدیل انتگرال گیردر خود دارد بنابراین به یکی از دو مدل foipdt یا ipdt باید تقریب بزنیم. برای مدل ipdt از تابع get_ipd و برای مدل foipdt تابع تبدیل را در یک sضرب کرده و بعد با opt_app به صورت یک تابع تبدیل رسته یک با دیلی تقریب می زنیم و بعد انتگرال گیر را به مدل تقریبی اضافه می کنیم.کدها به صورت زیر است:

```
clc;clear;close all
s=tf('s');
G_main=(-s+3)/((s)*(s+2)*(s^2+2*s+4));
step(G_main)
legend 'show'
hold on
[Kv,L] = get_ipd(G_main);
G_ipdt=Kv*exp(-L*s)/s;
G1=G_main*s;
R=opt_app(G1,0,1,1);
k=R.num{1}(2)/R.den{1}(2);
l=R.ioDelay;
T=1/R.den{1}(2);
G_foipdt=k*exp(-l*s)/(s*(T*s+1));

step(G_foipdt,G_ipdt);

figure;

err_foipdt=G_foipdt-G_main;
err_ipdt=G_ipdt-G_main;
step(err_ipdt,err_foipdt)
legend 'show'
```

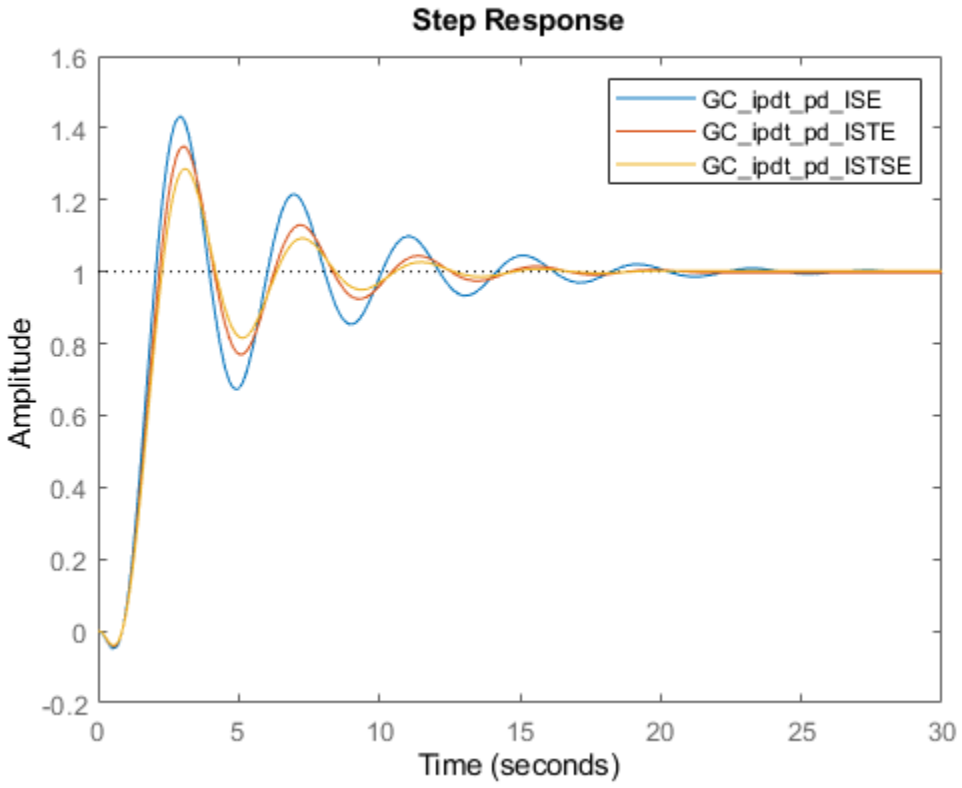

Step Response
G_main
G_foipdt
G_ipdt
Amplitude
Time (seconds)
0
10
20
30
40
50
60
70
80
90
100
0
50
100
150
200
250

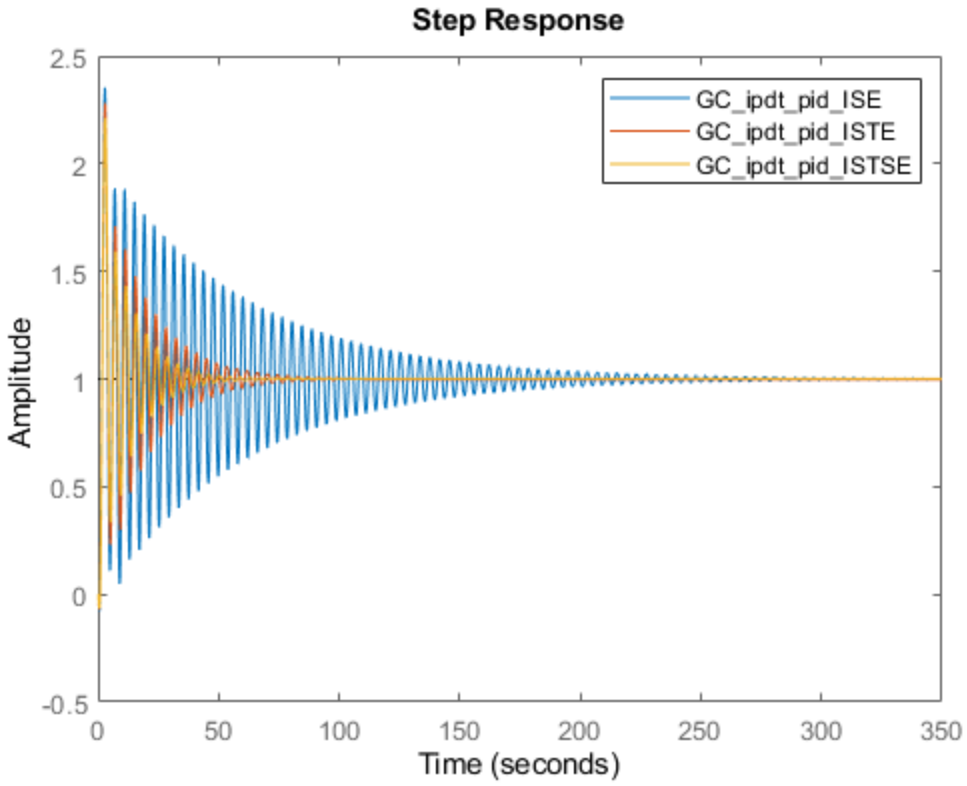

Step Response
err_ipdt
err_foipdt
Amplitude
Time (seconds)
-0.05
0
0.05
0.1
0.15
0.2
0
1
2
3
4
5
6
7
8

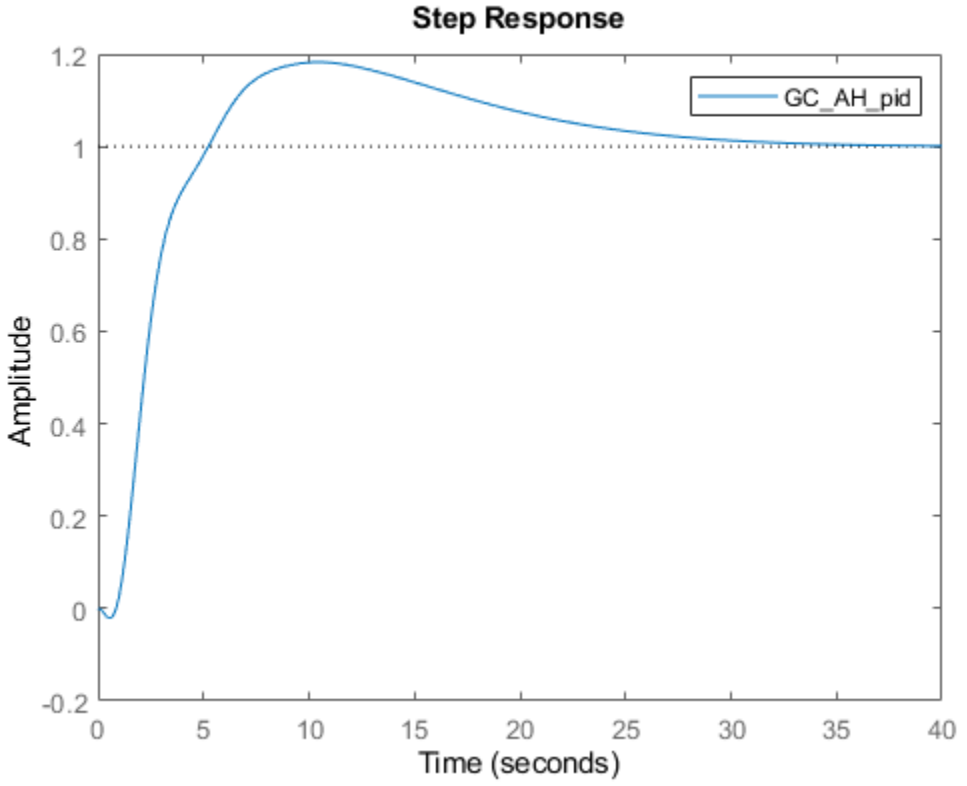
همانطور که مشاهده می شود مدلipdt *تقریب بهتری از تابع تبدیل اصلی است.پس برای این مدل کنترلر طراحی می کنیم.*

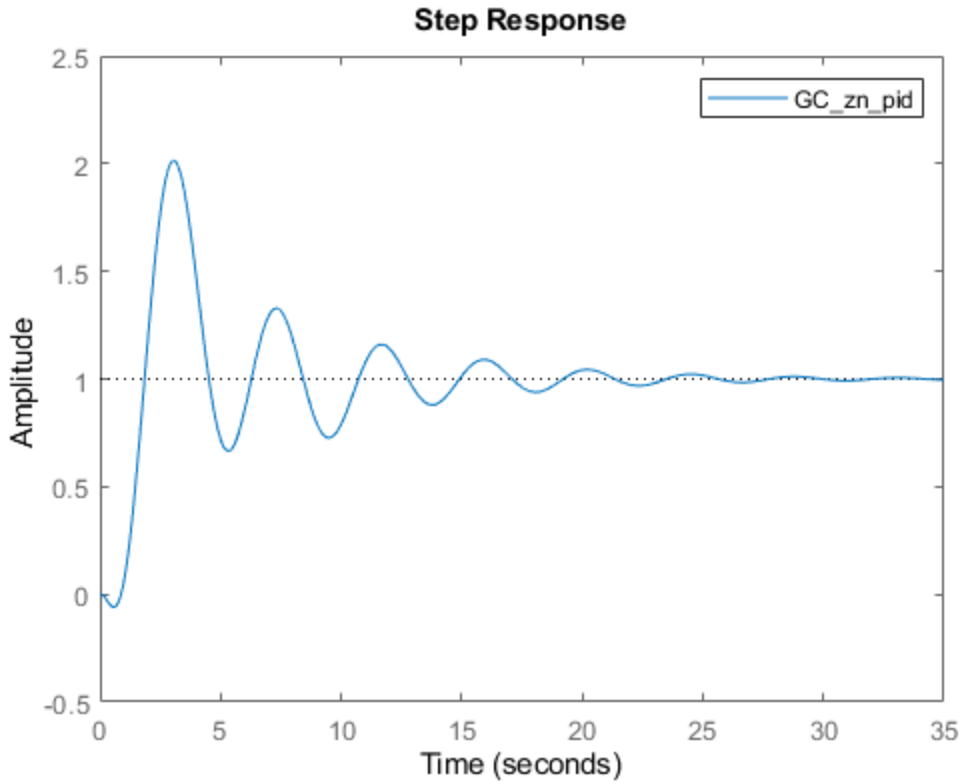
```
clc;clear;close all
s=tf('s');
G_main=(-s+3)/((s)*(s+2)*(s^2+2*s+4));

[Kv,L] = get_ipd(G_main);
G_ipdt=Kv*exp(-L*s)/s;
N=10;

[G_ipdt_pd_ISE,Kp,Ti,Td]=ipdtctrl(1,1,Kv,L,N); %ipdt pd ISE
GC_ipdt_pd_ISE=feedback(G_main*G_ipdt_pd_ISE,1);

[G_ipdt_pd_ISTE,Kp,Ti,Td]=ipdtctrl(1,2,Kv,L,N); %ipdt pd ISTE
GC_ipdt_pd_ISTE=feedback(G_main*G_ipdt_pd_ISTE,1);

[G_ipdt_pd_ISTSE,Kp,Ti,Td]=ipdtctrl(1,3,Kv,L,N); %ipdt pd ISTSE
GC_ipdt_pd_ISTSE=feedback(G_main*G_ipdt_pd_ISTSE,1);

step(GC_ipdt_pd_ISE,GC_ipdt_pd_ISTE,GC_ipdt_pd_ISTSE);
legend 'show'
figure

[G_ipdt_pid_ISE,Kp,Ti,Td]=ipdtctrl(2,1,Kv,L,N); %ipdt pid ISE
GC_ipdt_pid_ISE=feedback(G_main*G_ipdt_pid_ISE,1);

[G_ipdt_pid_ISTE,Kp,Ti,Td]=ipdtctrl(2,2,Kv,L,N); %ipdt pid ISTE
GC_ipdt_pid_ISTE=feedback(G_main*G_ipdt_pid_ISTE,1);

[G_ipdt_pid_ISTSE,Kp,Ti,Td]=ipdtctrl(2,3,Kv,L,N); %ipdt pid ISTSE
GC_ipdt_pid_ISTSE=feedback(G_main*G_ipdt_pid_ISTSE,1);

step(GC_ipdt_pid_ISE,GC_ipdt_pid_ISTE,GC_ipdt_pid_ISTSE);
legend 'show'
figure


[G_AH_pid,H_Sys,Kp,Ti,Td] = AH_idt(2,[Kv,L,N]);  %pid AH
GC_AH_pid=feedback(G_AH_pid*G_main,1);

step(GC_AH_pid);
legend 'show'
figure

[Gc_Zeigler,H_Sys,Kp,Ti,Td] = ziegler_itd(3,[Kv,L,N]); %pid ZN
GC_zn_pid=feedback(Gc_Zeigler*G_main,1);
```

```
step(GC_zn_pid);
legend 'show'
```




Step Response
GC_ipdt_pid_ISE
GC_ipdt_pid_ISTE
GC_ipdt_pid_ISTSE
Amplitude
Time (seconds)
2.5
2
1.5
1
0.5
0
-0.5
0
50
100
150
200
250
300
350


Step Response
GC_AH_pid
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25
30
35
40



کنترلر*ipdt_pd*:در این حالت بهترین کنترلر مربوط به مدل *ISTSE* است. اورشوت پایین تر و سرعت بیشتری دارد.اورشوت با این کنترلر *28.6%* و زمان نشست *s12* است.

کنترلر*ipdt_pid*:پاسخ سیستم با این کنترلر ها نوسان زیاد و اورشوت بالایی دارد و اصلا کنترلر مناسبی نیست.

کنترلر*pid_AH*: اورشوت 18.3% و زمان نشست *s27.8* و زمان برخواست *2.7s*است.این کنترلر سرعت بسیار کمی دارد.

کنترلر*Zn_pid*:این کنترلر حدود100 درصد اورشوت می زند و اصلا کنترلر مناسبی برای این سیستم نیست.

بهترین کنترلر در این حالت *ipdt_pd* مدل *ISTSE* است.