Laboratorium Kontrol dan Otomasi
Departemen Teknik Elektro ITS
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

MATLAB Training

Advance Level Module

BAB 1. Modelling

1.1. Pendahuluan

Dalam dunia teknik pengaturan diperlukan kemampuan memodelkan suatu sistem dinamik dalam bentuk persamaan matematika. Pemodelan sistem adalah teknik pendekatan sistem dalam bentuk persamaan matematis berdasarkan input dan output sistem tersebut. Model matematis ini diharapkan dapat menggambarkan perilaku (merepresentasikan) dinamika sistem yang didekati, khususnya jika diberi komponen baru, misalnya kontroler. Beberapa model matematika yang umum digunakan antara lain: model persamaan differensial, model fungsi alih, dan model state space. Pada bab ini dibahas pemodelan sistem menggunakan model fungsi alih yang akan diselesaikan dengan bantuan MATLAB. Fungsi MATLAB yang akan dimanfaatkan adalah toolbox System Identification.

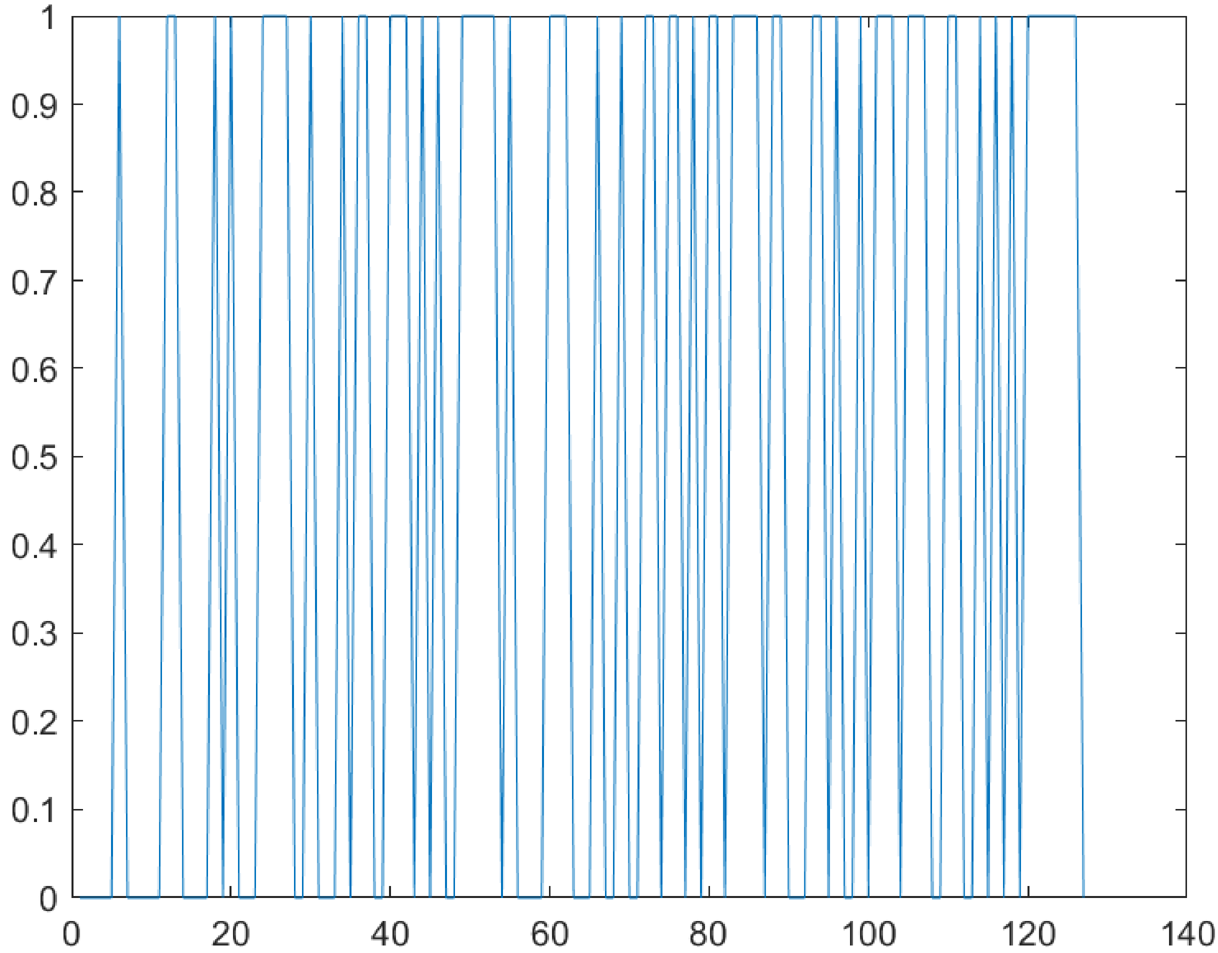
1.2. Pengambilan Data

Seperti yang sudah tertera pada pendahuluan, pemodelan memerlukan data berupa input-output sistem.Ada beberapa metode untuk mendeklarasi input-output sistem di matlab.

1.2.1. PRBS Signal Input

Pseudorandom Binary Sequence (PRBS) adalah sinyal deterministik periodik yang nilainya bergantian pada dua nilai. Sinyal PRBS periodik dengan periode maksimum $2^n - 1$, dengan n adalah orde PRBS. Pada MATLAB, kita dapat membangkitkan sinyal PRBS sebagai berikut

```
O = 7
N = 2^O-1
input = prbs(O,N)
plot(input)
```



Sinyal PRBS ini digunakan sebagai masukan dari sistem, sebagai sinyal uji, melalui input PRBS kita akan mengetahui bagaimana respon sistem dengan masukan berbeda antara 1 dan 0. Sinyal keluaran sistem dengan masukan PRBS inilah yang nantinya akan kita gunakan untuk memodelkan sistem tersebut. Selanjutnya kita akan menggunakan SIMULINK untuk mengambil output sistem blackbox dengan input PRBS.

```
%import data hasil uji
PRBS_input = out.PRBS_input
PRBS_output = out.PRBS_output
```

1.2.2. Import Data

Apabila data input-output sistem sudah diketahui, maka tidak perlu lagi menggunakan sinyal uji input PRBS, cukup melakukan import data ke workspace MATLAB.

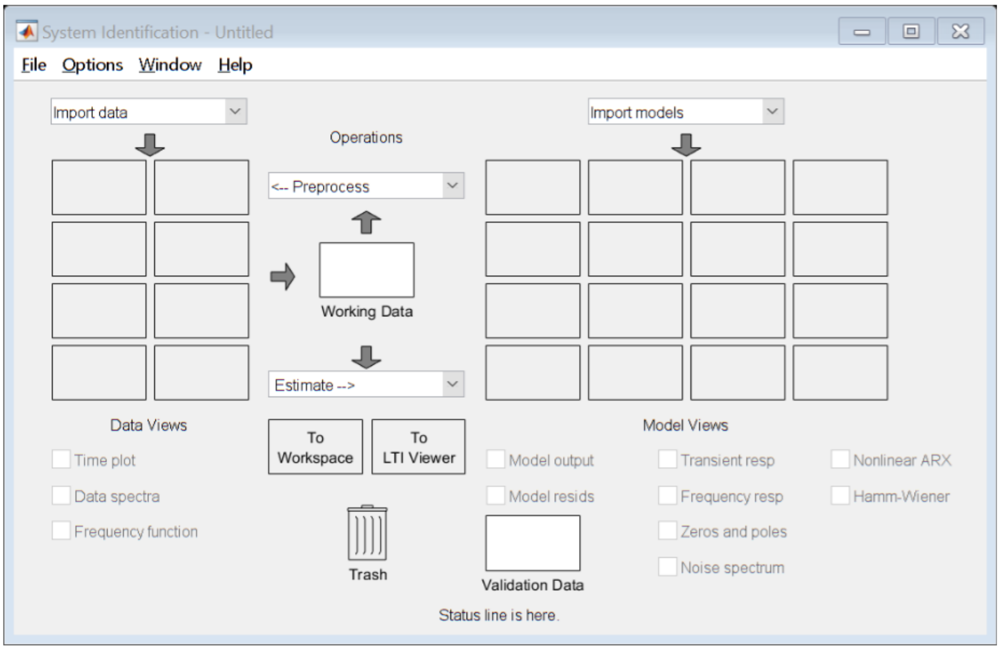
```
%import data file .txt
load('blackboxsystem_training_step.txt')
datasistem = blackboxsystem_training_step
time = datasistem(:,1)
input = datasistem(:,2)
output = datasistem(:,3)
```

Terdapat beberapa cara lain untuk import data, misalnya import data dari file .xls dan .csv

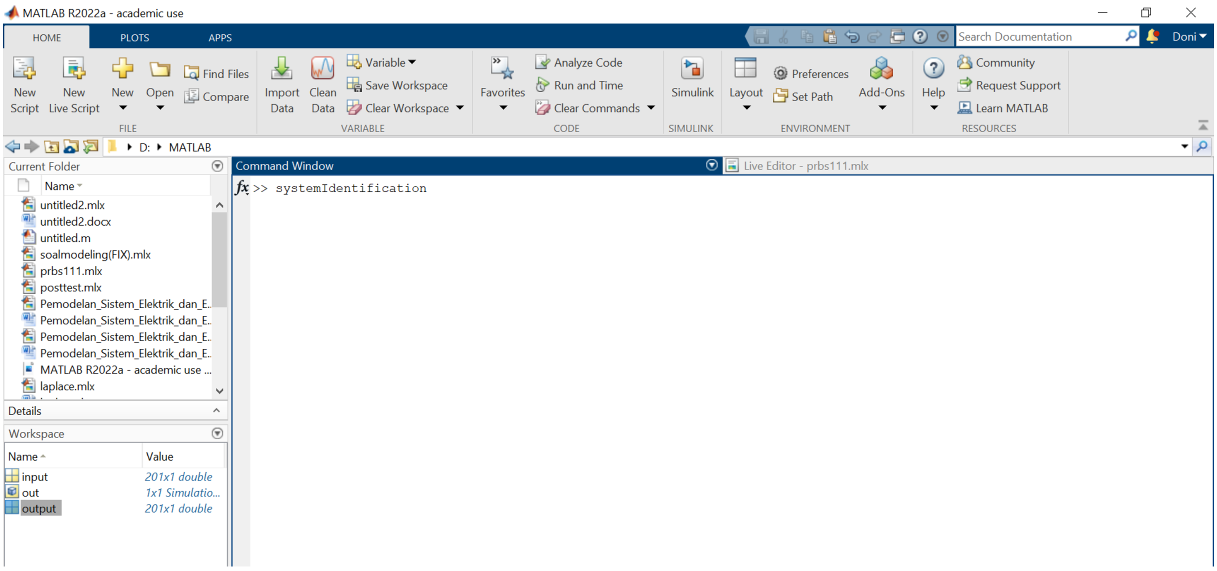
1.3. System Identification Toolbox

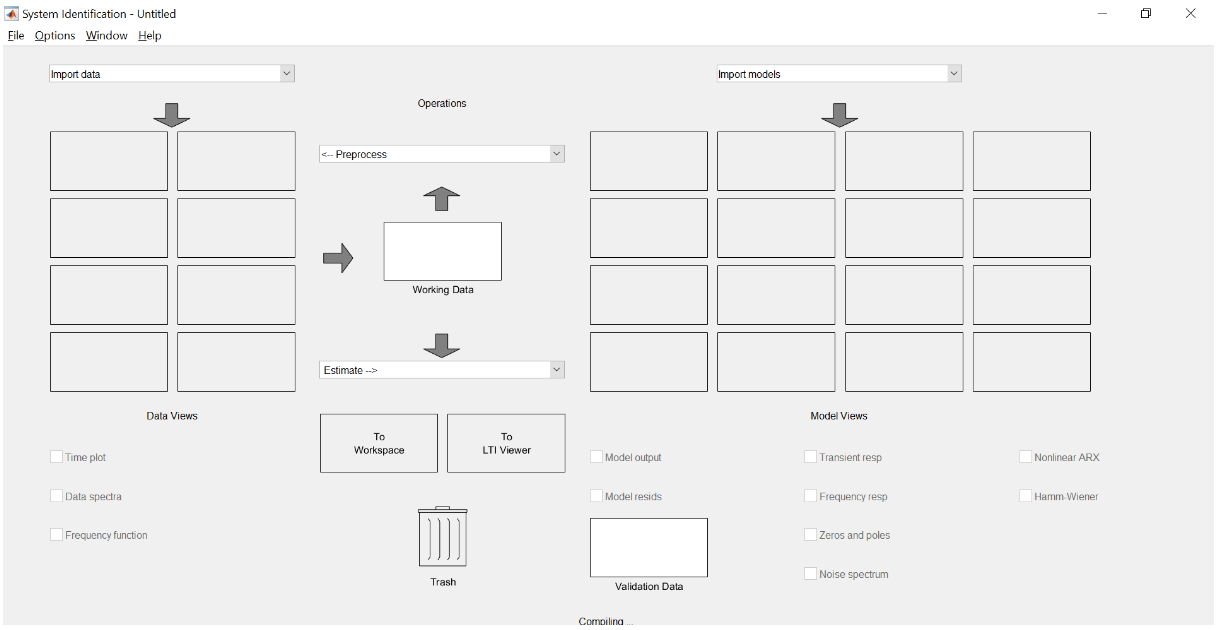
Setelah mendapat data sistem dan memasukkannya ke workspace MATLAB, selanjutnya kita dapat mulai memodelkan sistem. Pertama akan dipelajari cara mendapatkan fungsi alih sistem dengan menggunakan toolbox System Identification

```
systemIdentification
%deklarasikan hasil transfer function dari toolbox dalam variabel sys
s = tf('s')
```

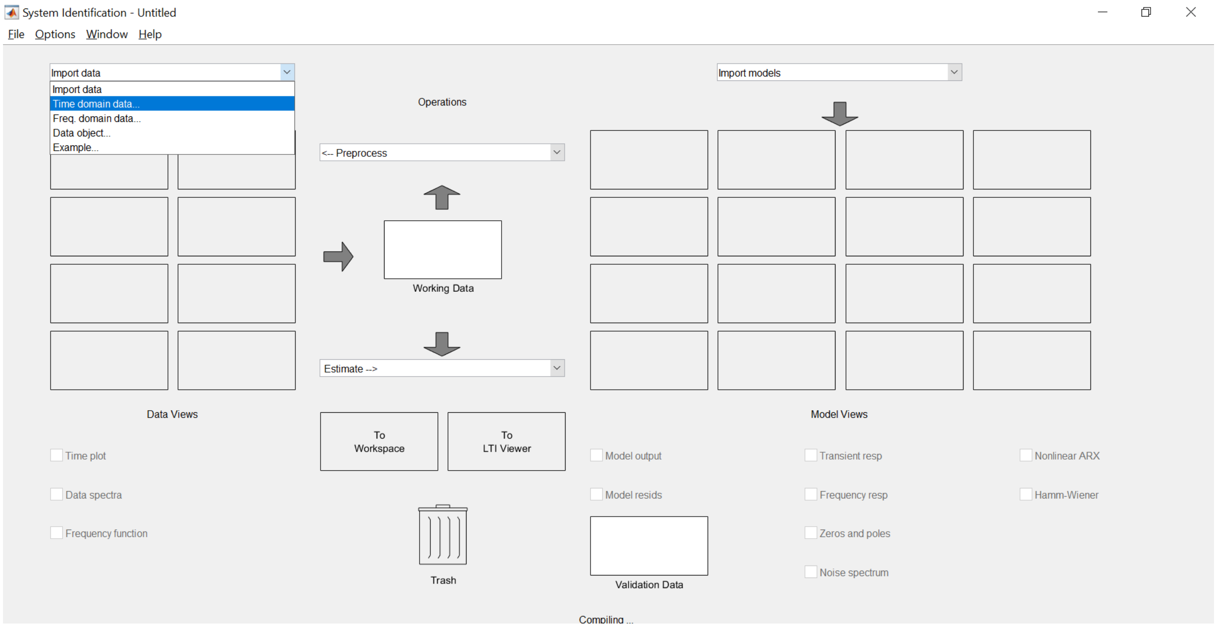


Toolbox ini dapat diakses menggunakan perintah 'systemIdentification' pada command window yang nantinya akan memunculkan sebuah window baru yang berisi fitur-fitur pada System Identification Toolbox. Dan sebelum menggunakan toolbox tersebut pastikan sudah memiliki data input dan output yang akan diidentifikasi.

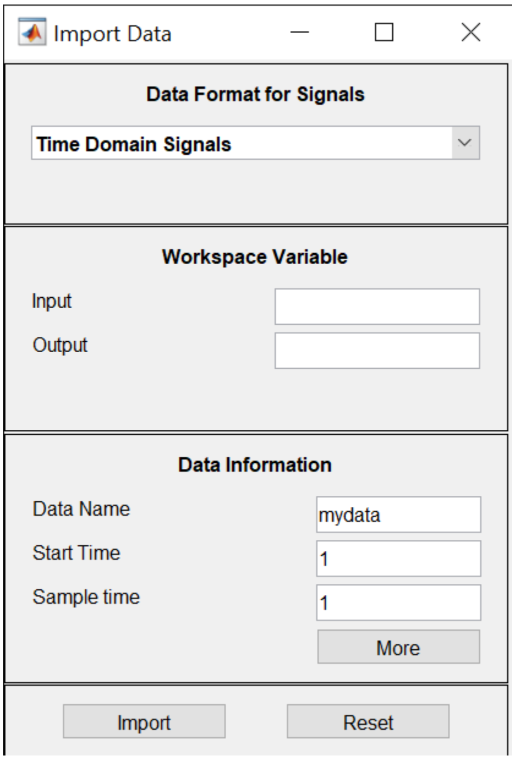


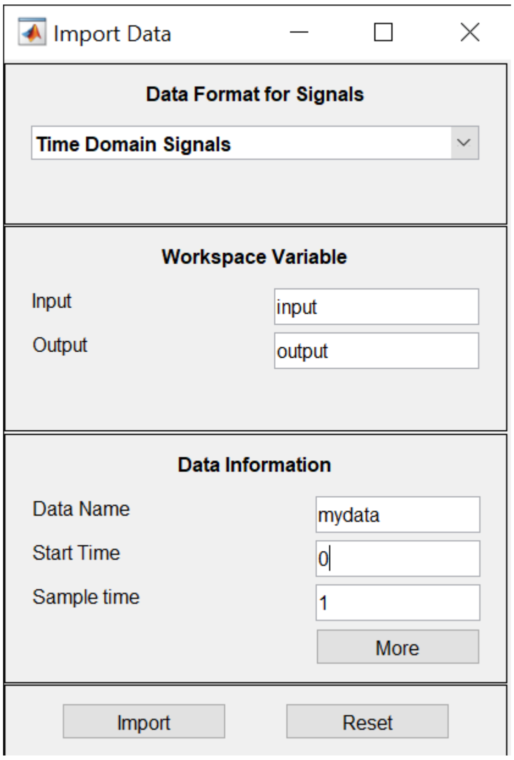


Untuk mengidentifikasi sistem dapat dilakukan dengan beberapa cara, untuk contoh penggunaannya kali ini akan dilakukan dengan menggunakan time domain data.

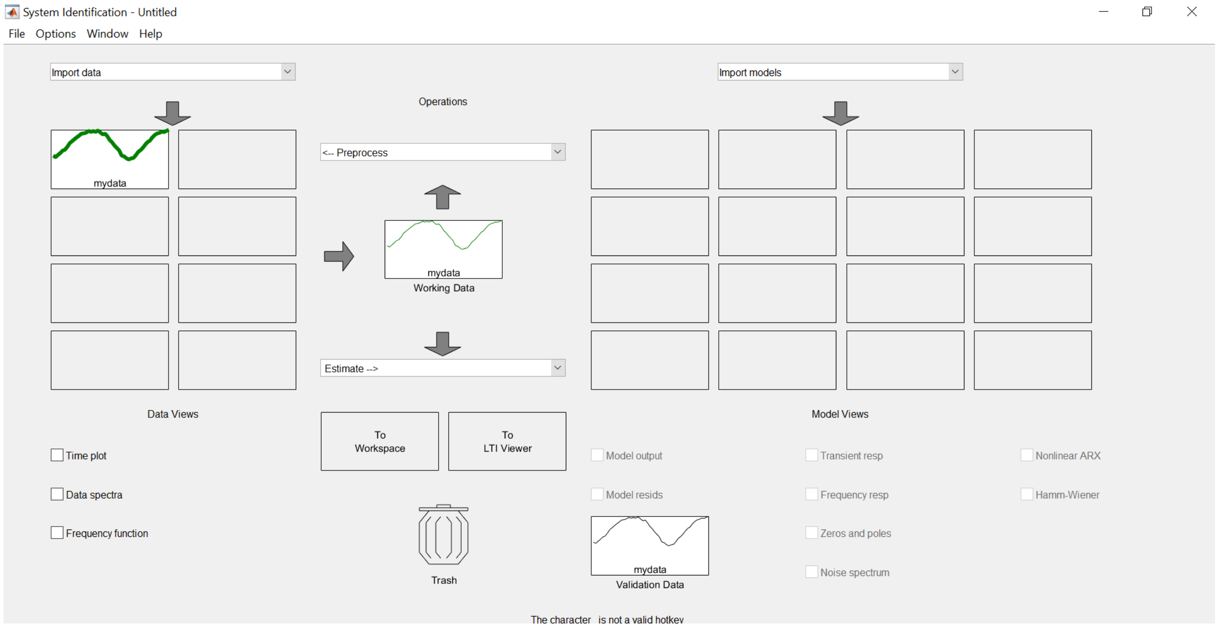


Setelah dipilih time domain data maka akan muncul window seperti berikut, lalu tuliskan nama input dan output yang telah didapatkan.

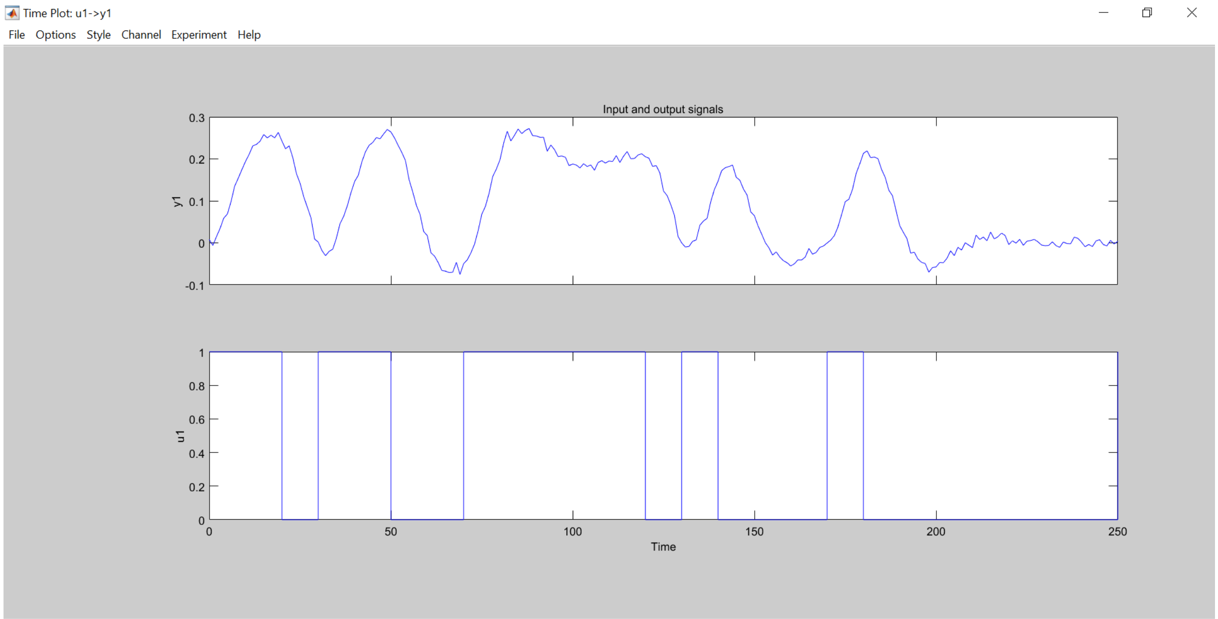




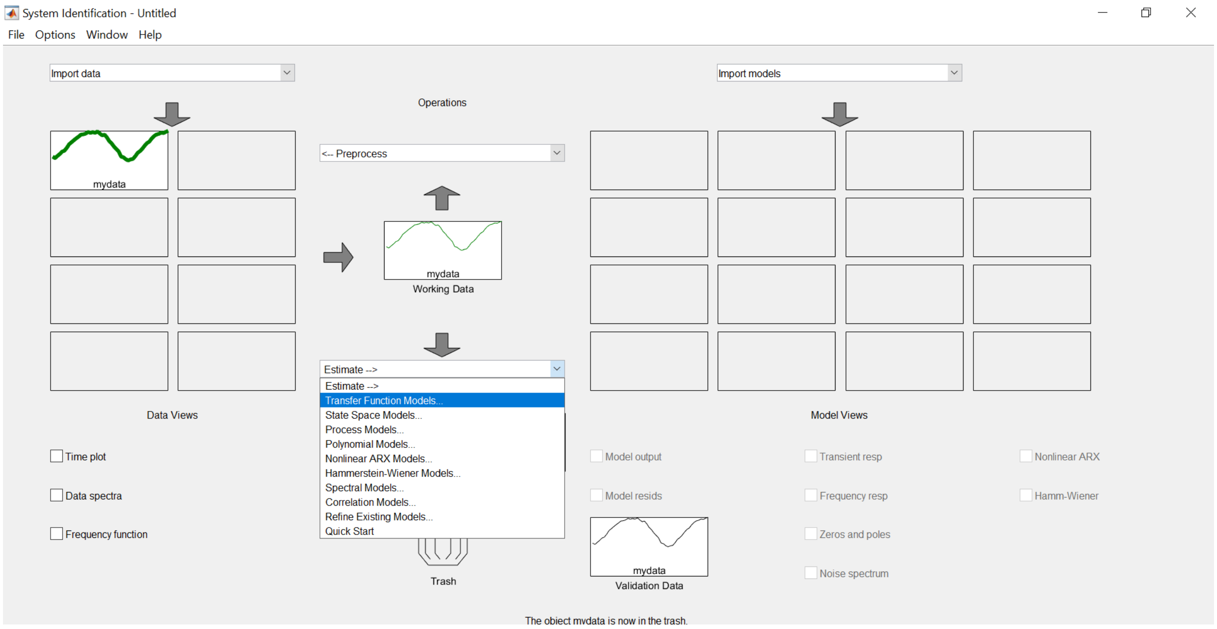
Setelah di import maka akan terlihat gambar pada bagian kiri yang menandakan bahwa data yang dimasukkan tadi telah berhasil diimport.



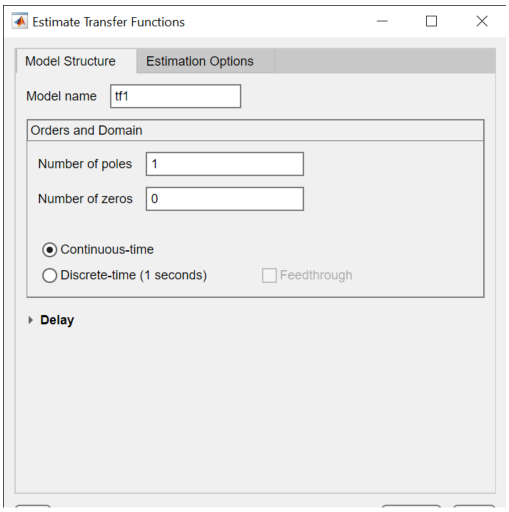
Untuk melihat input output yang telah diimport dapat dilakukan melalui perintah time plot yang terdapat pada bagian kiri bawah sehingga dapat menampilkan

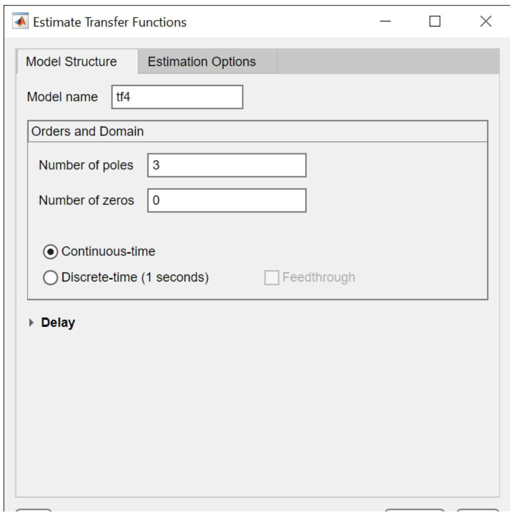


Untuk mengolah data atau mengestimasi plant maka pilih pada bagian Estimate dan pilih Transfer Function Models dikarenakan pada pembahasan kali ini akan berfokus pada model transfer function.

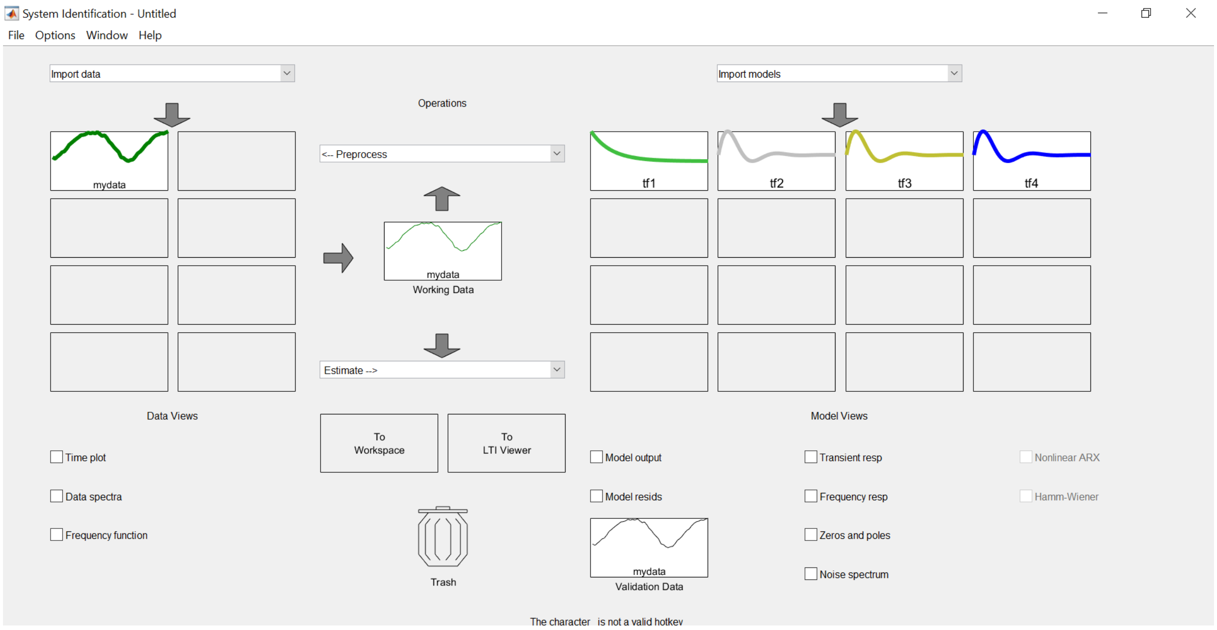


Pada window estimate transfer function akan kita coba untuk mengestimasi plant dengan mengubah-ubah jumlah pole dan zero yang memungkinkan dan klik estimate pada bagian pojok kanan bawah.

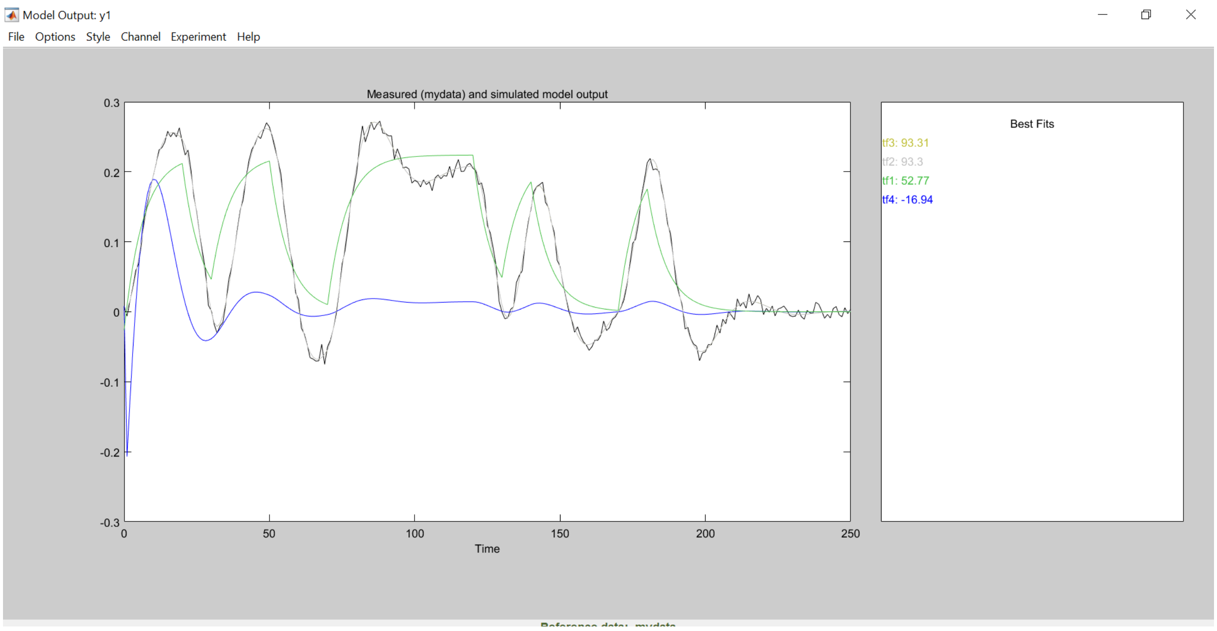




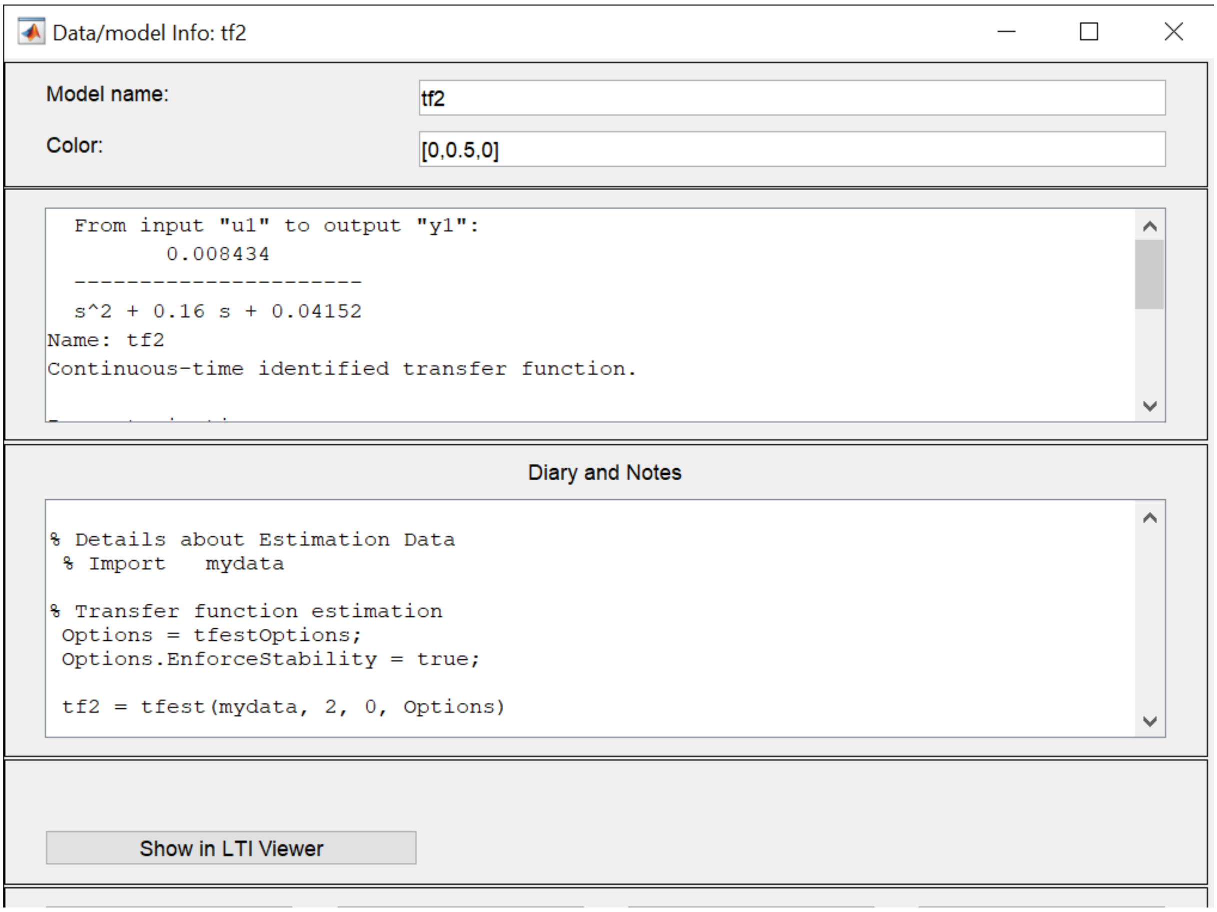
Setelah beberapa kali dilakukan estimasi, grafik atau hasil estimasi akan terlihat pada bagian kanan dan untuk membandingkan nilai estimasi mana yang paling mendekati plant sesungguhnya maka kita pilih Model output



Model output ini akan membandingkan masing-masing estimasi dengan real plant yang kita inputkan tadi dan memberikan persentase seberapa cocok output plant estimasi dengan plant sesungguhnya. Dalam hal ini tf3 mendapatkan persentase 93.31 sehingga nilai dari transfer function tf3 dapat kita gunakan sebagai model plant.



Untuk nilai spesifik dari plant estimasi tadi dapat kita lihat dengan cara melakukan double klik pada grafik yang ingin dilihat.



1.4. Metode Lainnya

Terdapat pula metode lain yang dapat digunakan untuk mencari fungsi alih sistem, diantaranya metode Ziegler-Nichols, metode Broida, dan metode Strejc.

1.4.1. Viteckova

Dengan metode ini, sistem didekati dengan model orde 1 atau orde 2. Data yang perlu diamati yaitu waktu saat respon mencapai 33% dan 70% dari nilai steady-state (), gain (K), dan reference (R). Untuk mencari nilai steady-state dari sistem dengan noise, digunakan perintah sebagai berikut

```
smoothing = sgolayfilt(output,1,17) %syntax untuk filtering data
threshold = 0.98*smoothing(end) %nilai diatas 98& dari data terakhir
```

```
ind = find(smoothing > threshold)
Css = mean(output(ind))
```

Untuk mempermudah pencarian nilai dan , dilakukan simulasi melalui SIMULINK.

```
%t33
V33 = 0.33*Css
C33 = find(output >= V33)
C33(1,1)
t33 = time(C33(1,1)) %waktu yang didapatkan adalah waktu terdekat dari value yang dicari
%untuk mendapatkan nilai yang lebih akurat dapat menggunakan cursor measurement pada SIMULINK
t33 = 0.113

%t70
V70 = 0.7*Css
C70 = find(output >= V70)
C70(1,1)
t70 = time(C70(1,1))
t70 = 0.346
R = 1
K = Css/R
```

Viteckova Orde 1

$$G_{v1}(s) = \frac{Ke^{(-T_{dv1})s}}{T_{v1}s+1}, \text{ dimana}$$

$$T_{dv1} = 1.498t_{33} - 0.498t_{70}$$

$$T_{v1} = 1.245(t_{70} - t_{33})$$

```
Tdv1 = (1.498*t33) - (0.498*t70)
Tv1 = 1.245*(t70-t33)
Gv1 = (K*exp(-Tdv1*s))/((Tv1*s)+1)
```

Viteckova Orde 2

$$G_{v2}(s) = \frac{Ke^{(-T_{dv2})s}}{(T_{v2}s+1)^2}, \text{ dimana}$$

$$T_{dv1} = 1.937t_{33} - 0.937t_{70}$$

$$T_{v1} = 0.794(t_{70} - t_{33})$$

```
Tdv2 = (1.937*t33) - (0.937*t70)
Tv2 = 0.794*(t70-t33)
Gv2 = (K*exp(-Tdv2*s))/(((Tv2*s)+1)^2)
```

1.4.2. Latzel

Metode Latzel memerlukan data berupa waktu saat respon mencapai 10%, 50%, dan 90% dari nilai steady-state (C_{ss}). Dengan persamaan sebagai berikut

$$G_{L(s)} = \frac{K}{(T_L s+1)^n}, \text{ dimana}$$

$$T_L = a_{10}t_{10} + a_{50}t_{50} + a_{90}t_{90}$$

Nilai a dan n didapat dari nilai μ beserta tabelnya,

$$\mu = \frac{t_{10}}{t_{90}}$$

Pilih nilai μ hasil perhitungan yang paling dekat dengan nilai μ pada tabel

μ_a	n	α_{10}	α_{50}	α_{90}	μ_a	n	α_{10}	α_{50}	α_{90}
0,137	2	1,880	0,596	0,257	0,456	11	0,142	0,094	0,065
0,174	2,5	1,245	0,460	0,216	0,472	12	0,128	0,086	0,060
0,207	3	0,907	0,374	0,188	0,486	13	0,116	0,079	0,056
0,261	4	0,573	0,272	1,150	0,499	14	0,106	0,073	0,053
0,304	5	0,411	0,214	0,125	0,512	15	0,097	0,068	0,050
0,340	6	0,317	0,176	0,108	0,523	16	0,090	0,064	0,047
0,370	7	0,257	0,150	0,095	0,533	17	0,084	0,060	0,045
0,396	8	0,215	0,130	0,085	0,543	18	0,078	0,057	0,042
0,418	9	0,184	0,115	0,077	0,552	19	0,073	0,054	0,040
0,438	10	0,161	0,103	0,070	0,561	20	0,069	0,051	0,039

```
V10 = 0.1*Css

C10 = find(output >= V10)

C10(1,1)

t10 = time(C10(1,1)) %waktu yang didapatkan adalah waktu
terdekat dari value yang dicari

%untuk mendapatkan nilai yang lebih akurat dapat menggunakan
cursor measurement pada SIMULINK

t10 = 0.113

V50 = 0.5*Css

C50 = find(output >= V50)

C50(1,1)

t50 = time(C50(1,1))

t50 = 0.204

V90 = 0.9*Css

C90 = find(output >= V90)

C90(1,1)

t90 = time(C90(1,1))

t90 = 0.67

miu = t10/t90
```

```
n = 2
a10 = 1.88
a50 = 0.596
a90 = 0.257

TL = (a10*t10)+(a50*t50)+(a90*t90)
GL = K/(((TL*s)+1)^n)
```

1.4.3. Ziegler-Nichols

Metode Ziegler-Nichols memerlukan nilai L (waktu delay) dan T (selisih antara waktu yang diperlukan sistem untuk pertama kali mencapai Css dengan L) dengan persamaan fungsi alihnya

$$G_Z(s) = \frac{K}{Ts+1}e^{-Ls}$$

```
L = 0
VtT = Css
CtT = find(output >= VtT)
CtT(1,1)
tT = time(CtT(1,1)) %waktu yang didapatkan adalah waktu terdekat dari value yang dicari
%untuk mendapatkan nilai yang lebih akurat dapat menggunakan cursor measurement pada SIMULINK
tT = 1.2
T = tT - L
GZ = (K*exp((-L)*s))/(T*s+1)
```

1.4.4. Strejc

Metode Strejc memiliki sedikit kesamaan dengan metode Ziegler-Nichols. Dalam hal ini metode Strejc membutuhkan nilai , yang nilainya sama dengan L, dan nilai , yang nilainya sama dengan T.

```
TU = L
TN = T
```

Metode Strejc memiliki dua kondisi berdasarkan nilai τ, jika nilai $\tau < 0$ maka persamaan fungsi alihnya sebagai berikut

$$G_{ST1}(s) = \frac{K}{(\tau_{ST1}s+1)(\tau_{ST2}s+1)}$$

Sedangkan jika nilai $\tau \geq 0$ maka persamaan fungsi alihnya sebagai berikut

$$G_{ST2}(s) = \frac{K}{(\tau_{ST}s+1)^n}, \text{ dimana}$$

$$\tau_{ST} = \frac{t_i}{n-1}$$

t_i = waktu yang dibutuhkan untuk mencapai nilai y_i

Nilai y_i dan t_i bisa diketahui dari tabel berikut yang bergantung pada nilai $\tau = \frac{T_U}{T_N}$

n	2	3	4	5	6	7	8	9	10
τ	0,104	0,218	0,319	0,41	0,493	0,57	0,642	0,709	0,773
y_i	0,264	0,327	0,359	0,371	0,384	0,394	0,401	0,407	0,413

```
tau = TU/TN
n = 2
yi = 0.264
Ci = find(output >= yi)
Ci(1,1)
ti = time(Ci(1,1)) %waktu yang didapatkan adalah waktu terdekat dari value yang dicari
```

```
%untuk mendapatkan nilai yang lebih akurat dapat menggunakan cursor measurement pada SIMULINK
ti = 0.259
tauST = ti/(n-1)
GST2 = K/(((tauST*s)+1)^n)
```

1.4.5. Broida

Metode Broida memiliki persamaan model alih fungsi sebagai berikut

$$G_B(s) = \frac{K}{Ts+1}e^{-\tau s}, \text{ dimana}$$

$$\tau = 2.8t_{28} - 1.8t_{40}$$

$$T = 5.5(t_{40} - t_{28})$$

```
V28 = 0.28*Css
C28 = find(output > V28)
C28(1,1)
t28 = time(C28(1,1))
t28 = 0.085

V40 = 0.4*Css
C40 = find(output > V40)
C40(1,1)
t40 = time(C40(1,1))
t40 = 0.145
```

```
tau = (2.8*t28)-(1.8*t40)
T = 5.5*(t40-t28)
GB = (K*exp((tau)*s))/(T*s+1)
```

1.4.6. Validasi

Pada bab ini, kita coba bandingkan akurasi fungsi alih dari masing-masing metode yang dicoba sebelumnya dengan data sebenarnya. Digunakan metode mean square error dengan syntax 'mse'

```
t = 0:0.1:127
[yGv1,t] = step(Gv1,t)
[yGv2,t] = step(Gv2,t)
[yGL,t] = step(GL,t)
[yGZ,t] = step(GZ,t)
[yGB,t] = step(GB,t)
[yGST2,t] = step(GST2,t)
```

```
MSE_Gv1=mse(yGv1,output)
MSE_Gv2=mse(yGv2,output)
MSE_GL=mse(yGL,output)
MSE_GZ=mse(yGZ,output)
MSE_GB=mse(yGB,output)
MSE_GST2=mse(yGST2,output)
```

BAB 2. Karakteristik Sistem

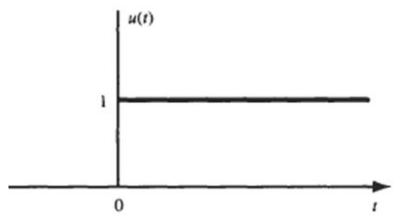
2.1. Analisis Sistem

Untuk menganalisa dan mendesain sistem pengaturan, diperlukan pengetahuan mengenai karakteristik sistem. Karakteristik suatu sistem dapat diketahui dengan mengamati respon sistem tersebut terhadap sinyal uji tertentu. Pada bagian ini akan dibahas mengenai karakteristik sistem dan macam-macam sinyal uji yang lazim digunakan dalam sistem pengaturan. Disebut pula sebagai spesifikasi performansi sistem. Selain itu pada bab ini akan dibahas kestabilan, controllability, dan observability.

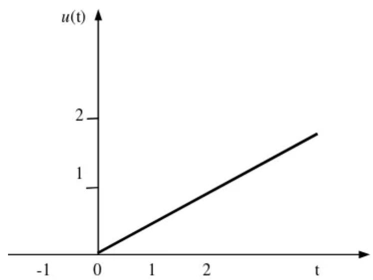
2.2. Sinyal Uji

Sinyal uji merupakan sinyal input yang biasa digunakan untuk menguji atau melakukan analisis terhadap suatu sistem. Untuk memudahkan dalam melakukan analisis terhadap suatu respon, digunakan beberapa sinyal uji dengan fungsi waktu sederhana. Berikut merupakan beberapa sinyal uji yang biasanya digunakan

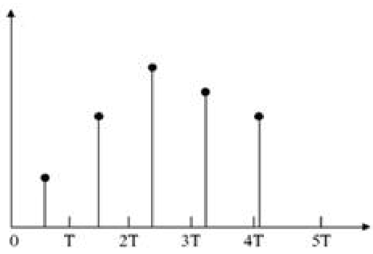
1. Sinyal step : berguna untuk menguji respon terhadap gangguan yang muncul tiba-tiba, dan juga melihat kemampuan sistem kontrol dalam memposisikan respon.



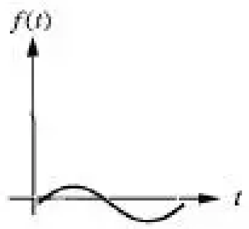
2. Sinyal ramp : fungsi berubah bertahap terhadap waktu, berguna untuk melihat kemampuan sistem kontrol dalam melacak target yang bergerak dengan kecepatan konstan.



3. Sinyal impuls : berguna untuk menguji respon terhadap gangguan sesaat yang muncul tiba-tiba dan untuk menguji sistem yang responnya berubah dalam selang waktu yang sangat singkat.

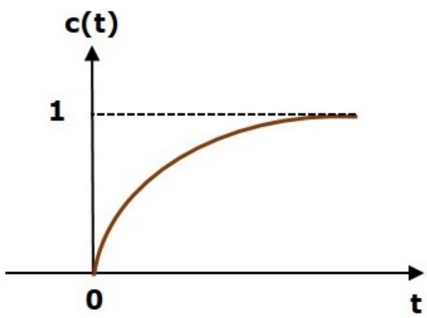


4. Sinyal sinusoidal : berguna untuk menguji respon sistem yang menerima input berupa sinyal sinusoidal.



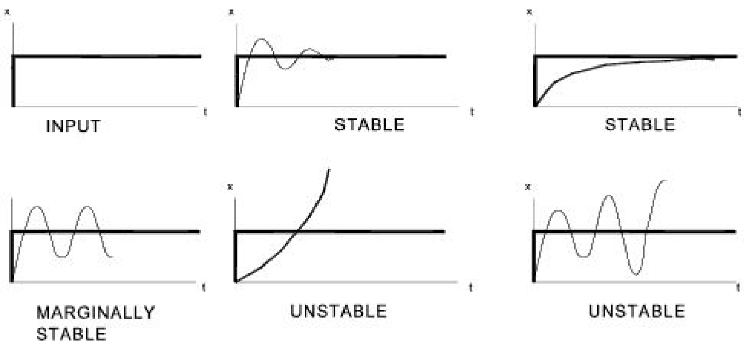
2.3. Analisis Root Locus

Sistem dikatakan stabil apabila dalam keadaan steady-state dihasilkan bounded output untuk setiap bounded input.



Gambar di atas adalah contoh respon step sistem yang stabil karena dalam keadaan steady-state nilai amplitude dan frequency outputnya konstant berada di antara 0 dan 1 (bounded output). Selain itu unit step yang merupakan sinyal input bernilai 1 untuk setiap waktu (bounded input).

Untuk dapat membedakan sistem stable dan sistem unstable dapat melihat gambar berikut



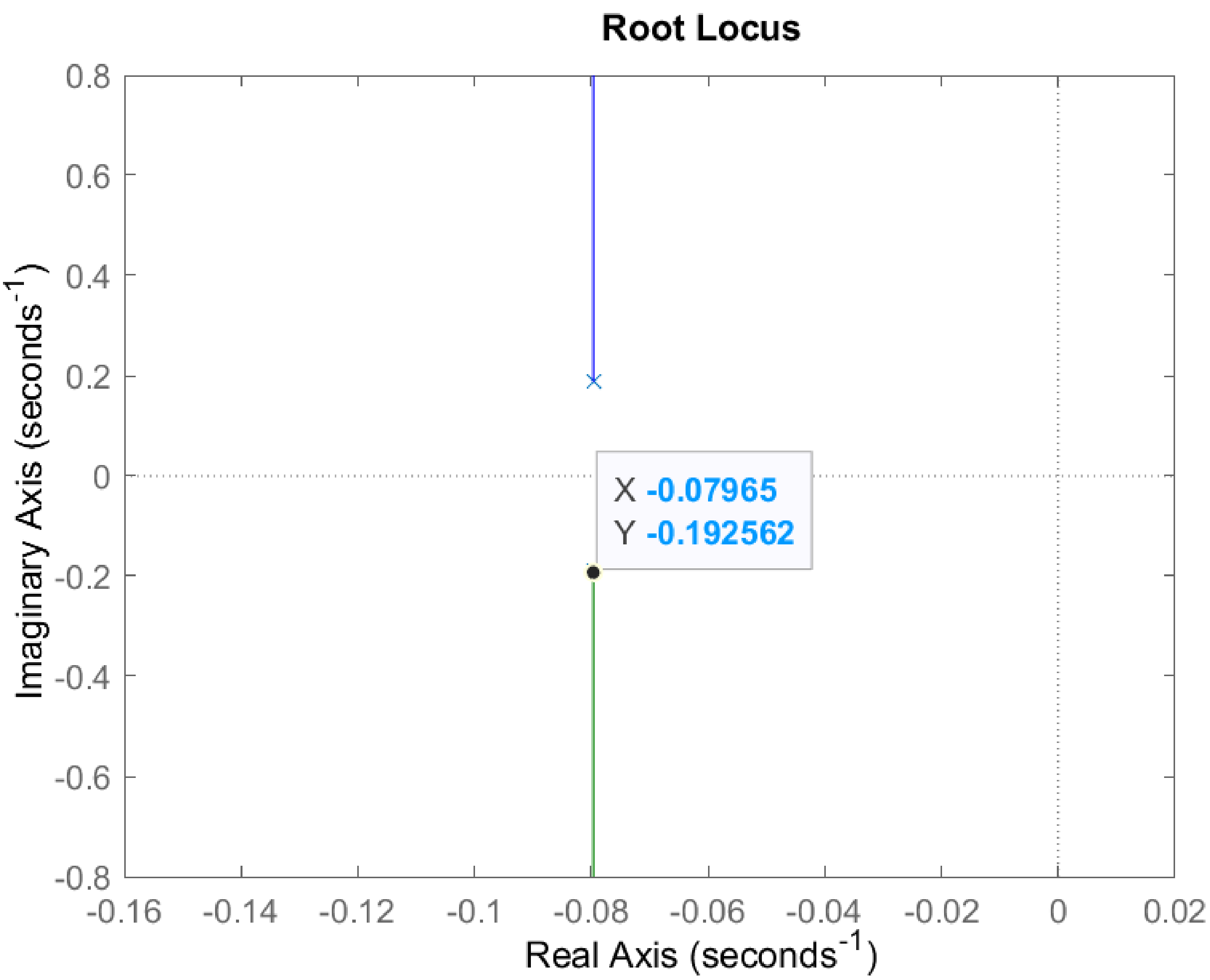
Root locus sendiri plot lokasi dari semua akar-akar persamaan karakteristik dari suatu sistem. Dengan menggunakan root locus ini akan dapat ditentukan stabilitas dari suatu sistem. Secara singkatnya berdasarkan metode ini, suatu sistem dikatakan stabil apabila letak akar-akarnya berada pada sebelah kiri sumbu imajiner. Terdapat 2 macam kestabilan antara lain:

1. Absolutely Stable System
   Sistem dikatakan absolutely stable apabila open loop transfer function dan closed loop transfer function semua pole-nya berada pada sisi kiri bidang ‘s’.
2. Marginally Stable System
   Sistem dikatakan marginally stable apabila open loop transfer function dan closed loop transfer function terdapat pole pada sumbu imajiner.

Sebagai contoh dengan suatu transfer function

$$G(s) = \frac{0.008519}{s^2+0.1593s+0.0418}$$

```
num = [0.008519]

den = [1 0.1593 0.0418]

G=tf(num,den)

rlocus(G)
```



2.4. Analisis State Space

Didapatkan bentuk persamaan transfer function

$$G(s) = \frac{0.008519}{s^2+0.1593s+0.0418}$$

Bentuk umum persamaan state space

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Definisikan numerator dan denumerator dari transfer function

```
num = [0.008519]

den = [1 0.1593 0.0418]

G=tf(num,den)
```

Untuk mendapatkan nilai A, B, C, D dapat menggunakan tf2ss untuk mengubah bentuk TF ke SS

```
[A,B,C,D] = tf2ss(num,den)
```

Stability

Sistem dikatakan stabil apabila semua nilai eigen λ_n dari matriks **A** terletak di sebelah kiri sumbu imajiner.

Matriks A dari persamaan state space yaitu:

$$A = \begin{matrix} -0.1593 & -0.0418 \\ 1 & 0 \end{matrix}$$

Untuk mencari nilai eigen dari matriks A

```
lambda=eig(A)
```

Nantinya jika didapatkan nilai dari λ_n berada pada sebelah kiri sumbu imajiner maka dapat disimpulkan bahwa sistem stabil, begitupun sebaliknya jika λ_n berada pada sebelah kanan sumbu imajiner maka dapat disimpulkan bahwa sistem tidak stabil

Controllability

Suatu sistem controllable jika matriks $M_C = [B|AB|A^2B|...|A^{n-1}B|]$ mempunyai rank sama dengan n. Berdasarkan persamaan state space yang diperoleh maka matriks M_C sebagai berikut:

$$M_C = [B|AB]$$

Dengan memasukkan matriks A dan B dan melakukan perhitungan dengan MATLAB

```
Mc=[B A*B]

rank(Mc)
```

Nantinya akan didapatkan rank pada perhitungan tersebut, jika rank sama dengan n dapat disimpulkan sistem controllable

Observability

Suatu sistem observable jika matriks M_o, mempunyai rank n.

$$\left(M_o\right)^T = \left[\, C \quad CA \quad CA^2 \quad ... \quad CA^{n-1} \,\right]$$

Berdasarkan persamaan state space yang diperoleh maka matriks sebagai berikut:

$$M_o = [\, C \quad CA \,]$$

Dengan memasukkan matriks A dan C dan melakukan perhitungan dengan MATLAB

```
Mo=[C;CA]

rank(Mo)
```

Nantinya akan didapatkan rank pada perhitungan tersebut, jika rank sama dengan n dapat disimpulkan sistem observable

2.5. Analisis Domain Waktu

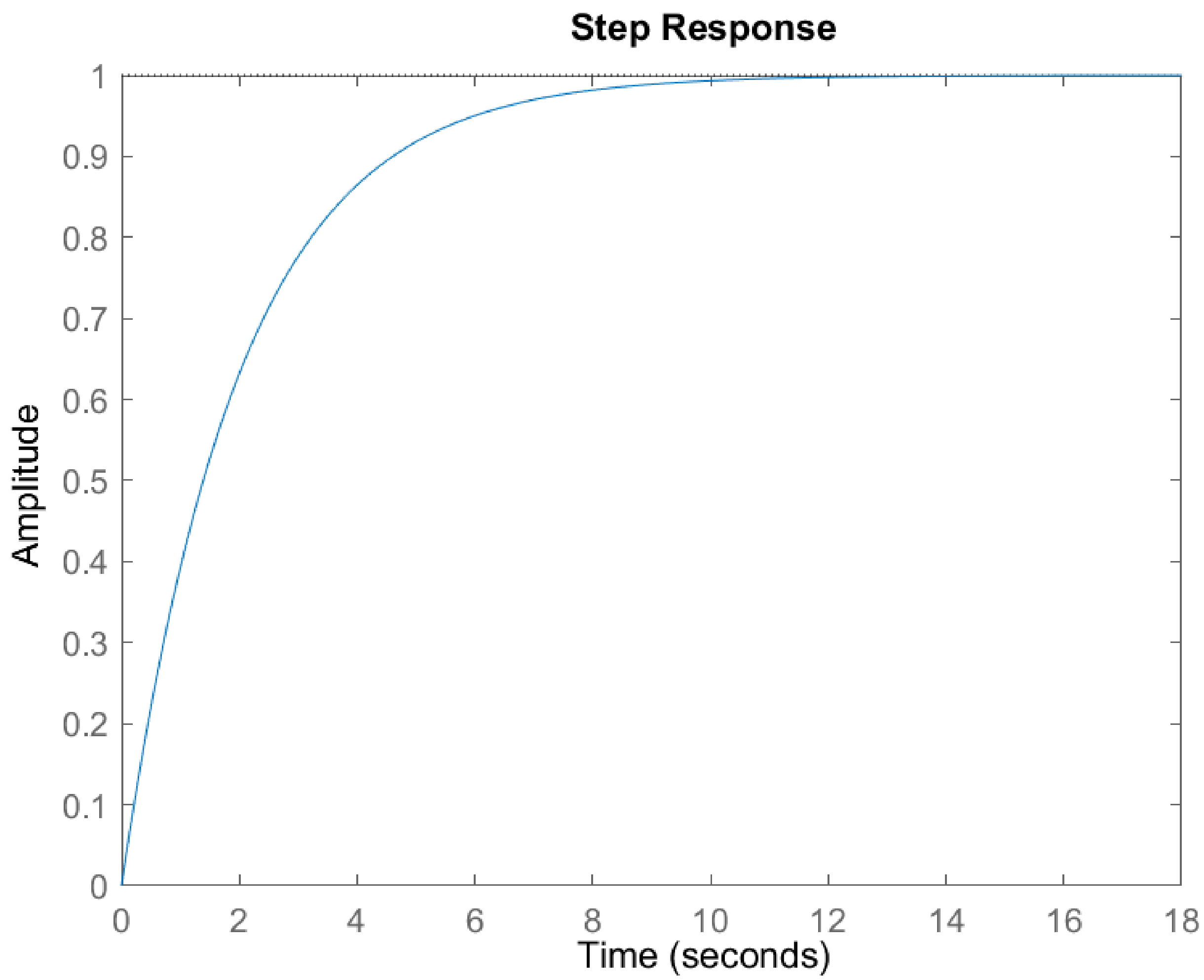
Time Domain memberikan gambaran ketika terjadinya perubahan suatu sistem dinamis terhadap waktu ketika sistem tersebut diberikan input tertentu. Hal tersebut menghasilkan karakteristik-karakteristik dari sinyal terhadap waktu yang didapatkan ketika melakukan observasi terhadap sinyal yang dihasilkan. Karakteristik respon waktu yang dihasilkan oleh sinyal step dibagi menjadi dua yaitu karakteristik respon transien dan karakteristik respon steady state. Berikut merupakan beberapa karakteristik yang didapat ketika melakukan analisis terhadap domain.

2.5.1. Karakteristik Respon Orde 1

Untuk dapat mengetahui karakteristik respon orde 1, terlebih dahulu perlu diketahui persamaan transfer function dari plant orde 1 yaitu

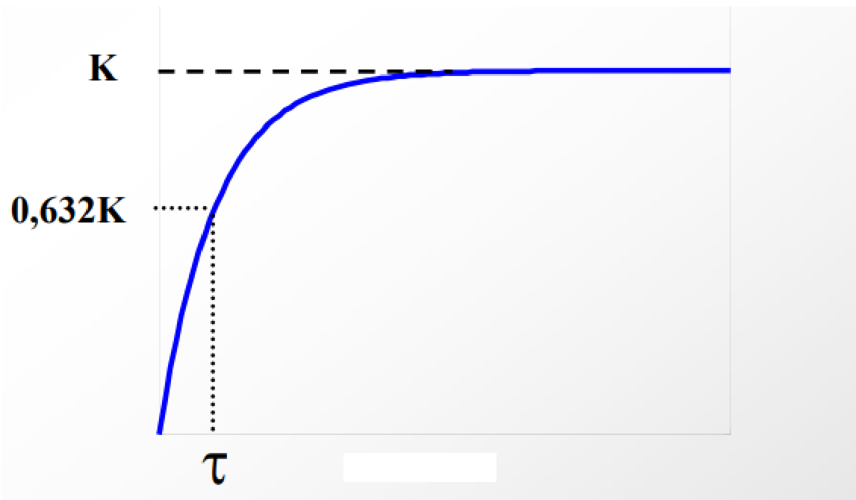
$$\frac{C(s)}{R(s)} = \frac{K}{\tau s+1}$$

```
k=1;

tau=2;

g=tf(k,[tau 1])

step(g)
```



1. Karakteristik respon transien orde 1

Sebelumnya perlu diketahui bahwa terdapat suatu konstanta waktu τ yang merupakan waktu untuk respon dari t=0 sampai mencapai 63,2% dari respon steady state.



a. Settling Time (ts)
Settling time merupakan ukuran waktu yang menyatakan bahwa respon sistem telah masuk pada daerah stabil
ts(±5%) ≈ 3τ

```
ts=3*tau %%settling time +-5%
```

ts(±2%) ≈ 4τ

```
ts=4*tau %%settling time +-2%
```

ts(±0.5%) ≈ 5τ

```
ts=5*tau %%settling time +-0.5%
```

b. Rise Time (tr)
Rise time merupakan waktu yang dibutuhkan oleh suatu sistem respon untuk naik dari 5% ke 95%, 10% ke 90%, atau dalam buku lain juga menyebutkan dari 0% ke 100% dari nilai respon steady state.
tr(10%-90%) ≈ τ ln9

```
tr=tau*log(9) %%rise time 10%-90%
```

tr(5%-95%) ≈ τ ln19

```
tr=tau*log(19) %%rise time 5%-95%
```

c. Delay Time (td)
Delay time atau waktu tunda yaitu waktu yang dibutuhkan respon mulai dari t=0 sampai respon mencapai 50% dari respon steady state. Besarnya faktor delay dari respon ini biasanya berdasarkan akibat dari proses sampling.
td = τ ln2

```
td=tau*log(2) %%delay time
```

2. Karakteristik respon steady state orde 1
a. Error Relatif pada Kondisi Steady State
Error merupakan kesalahan pengukuran yang terjadi antara set point dengan respon keluaran yang dihasilkan. Error ini dapat dihitung dengan
$\varepsilon_{ss} = \frac{X_{ss} - Y_{ss}}{X_{ss}} \times 100\%$, dengan

$$Y_{ss} = \lim_{t \to \infty} y(t) = \lim_{s \to 0} s\, Y_s = \lim_{s \to 0} s \left(\frac{K}{s(\tau s+1)}\right) = K$$

$$X_{ss} = \lim_{t \to \infty} x(t) = \lim_{s \to 0} s\, X_s = \lim_{s \to 0} s \left(\frac{1}{s}\right) = 1$$

Sehingga, $\varepsilon_{ss} = (1 - K) \times 100\%$

```
error=(1-k)*100 %%error dalam persentase
```

Pada matlab ini juga dapat menampilkan semua karakteristik dari plant orde tersebut dengan menggunakan

```
stepinfo(g)
```

```
ans = struct with fields:
        RiseTime: 4.3940
   TransientTime: 7.8241
    SettlingTime: 7.8241
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 21.0917
```

2.5.2. Karakteristik Respon Orde 2

Pada orde 2 memiliki beberapa perbedaan dalam segi persamaan dan karakteristik dari orde 1. Persamaan umum dari plant orde 2 adalah sebagai berikut

$$\frac{C(s)}{R(s)} = \frac{K\omega_n^{\ 2}}{s^2 + 2\xi\omega_n s + \omega_n^{\ 2}}$$

K : gain overall
ξ : rasio redaman
ω_n: frekuensi tak teredam
Dari persamaan tersebut dapat dilihat redaman yang terjadi pada suatu plant
a. Under damped
Redaman kurang terjadi dikarenakan nilai dari ξ kurang dari 1 atau 0 < ξ < 1, hal ini berarti pole yang dimiliki oleh plant tersebut bernilai konjugat komplek

$$p1 = -\xi\omega_n + j\omega_n\sqrt{1 - \xi^2}$$

$$p2 = -\xi\omega_n - j\omega_n\sqrt{1 - \xi^2}$$

b. Critically damped

Critically damped berarti plant tersebut teredam dengan maksimal atau dan berarti memiliki akar kembar yaitu $p1 = p2 = -\omega_n$

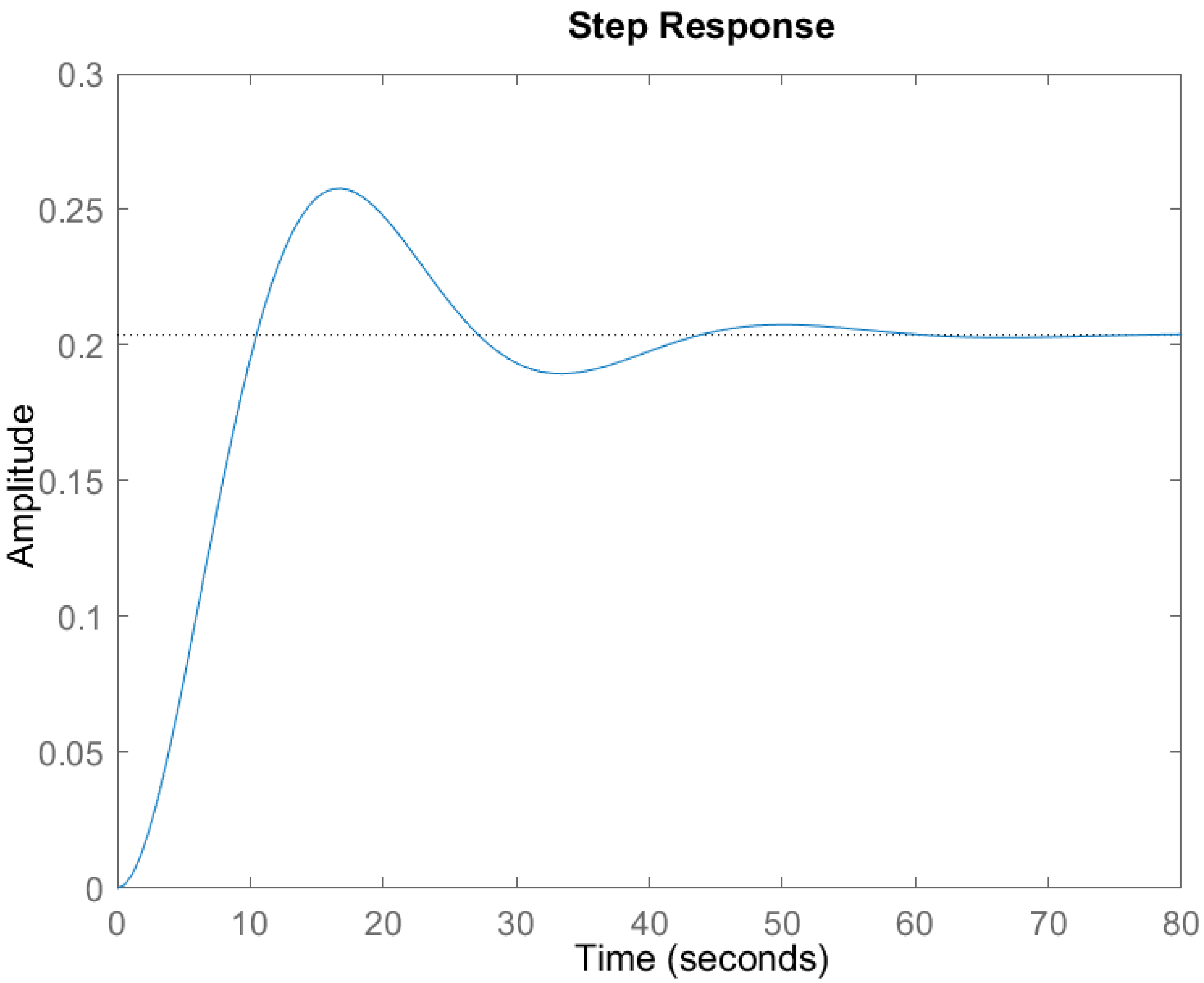
c. Over damped

Redaman berlebih terjadi dikarenakan nilai dari yang berarti polenya bernilai real dan berbeda

$$p1 = -\xi\omega_n + \omega_n\sqrt{\xi^2 - 1}$$

$$p2 = -\xi\omega_n - \omega_n\sqrt{\xi^2 - 1}$$

```
k=0.20380382775;
wn=0.204450483;
e=0.38958; %underdamped 0<e<1
h=tf(k*wn^2,[1 2*e*wn wn^2])
step(h)
```




```
%Mencari pole
p1=-e*wn+wn*sqrt(1-e^2)*i
p2=-e*wn-wn*sqrt(1-e^2)*i
```

Selanjutnya mengenai karakteristik respon dari orde 2 yang sebelumnya juga terdapat pada orde 1, terdapat beberapa tambahan pada karakteristik dan juga persamaan tiap karakteristik yang berbeda dengan orde 1. Berikut penjelasan lebih lanjut mengenai karakteristik respon yang terdapat pada orde 2

1. Karakteristik respon transien orde 2
a. Settling Time (ts)
Settling time merupakan ukuran waktu yang menyatakan bahwa respon sistem telah masuk pada daerah stabil

$ts(\pm\ 5\%) \approx 3\tau \approx \frac{3}{\sigma} \approx \frac{3}{\xi\omega_n}$

```
ts5=3/(e*wn) %settling time +-5%
```

$$ts(\pm\ 2\%) \approx 4\tau \approx \frac{4}{\sigma} \approx \frac{4}{\xi\omega_n}$$

```
ts2=4/(e*wn) %settling time +-2%
```

b. Rise Time (tr)

Rise time merupakan waktu yang dibutuhkan oleh suatu sistem respon untuk naik dari 5% ke 95%, 10% ke 90%, atau dalam buku lain juga menyebutkan dari 0% ke 100% dari nilai respon steady state.

$$tr = \frac{1}{\omega_d} tan^{-1}(\frac{\omega_d}{-\sigma}) = \frac{\pi - \beta}{\omega_d}$$

Dengan $\omega_d = \omega_n \sqrt{1 - \xi^2}$ dan $\beta = tan^{-1} \frac{\omega_d}{\xi\omega_n}$

```
wd=wn*sqrt(1-e^2)
tr2=(pi-beta)/wd  %rise time
```

c. Peak Time (tp)

Waktu puncak yaitu waktu yang diperlukan oleh suatu respon untuk mencapai puncak overshoot pertama.

$$tp = \frac{\pi}{\omega_d}$$

```
tp2=pi/wd %peak time
```

d. Maksimum Overshoot (Mp)

Maksimum overshoot merupakan nilai puncak dari suatu respon sistem. Persentase overshoot maksimum bernilai

$$\%Mp = \frac{c(tp) - c(\infty)}{c(\infty)} \times 100\%$$

Maksimum overshoot ini dapat terjadi ketika $t = tp = \frac{\pi}{c}$

$$Mp = c(tp) - 1$$

$$Mp = e^{-(\frac{\sigma}{\omega_d})\pi} = e^{-(\frac{\xi}{\sqrt{1-\xi^2}})\pi}$$

```
mp=exp(-e*pi/sqrt(1-e^2)) %maksimum overshoot
```

2. Karakteristik respon steady state orde 2

a. Error Relatif pada Kondisi Steady State

Error merupakan kesalahan pengukuran yang terjadi antara set point dengan respon keluaran yang dihasilkan. Error ini dapat dihitung dengan

$$\varepsilon_{ss} = \frac{2\xi}{\omega_n}$$

```
error2=(2*e)/wn*100 %error dalam persentase
```

```
stepinfo(h)
```

```
ans = struct with fields:
        RiseTime: 7.0860
   TransientTime: 41.1321
    SettlingTime: 41.1321
     SettlingMin: 0.1895
     SettlingMax: 0.2578
       Overshoot: 26.4732
      Undershoot: 0
            Peak: 0.2578
        PeakTime: 16.7671
```

```
damp(h)
```

```
        Pole                Damping       Frequency      Time Constant
                                        (rad/seconds)      (seconds)

-7.96e-02 + 1.88e-01i     3.90e-01       2.04e-01         1.26e+01
-7.96e-02 - 1.88e-01i     3.90e-01       2.04e-01         1.26e+01
```

2.6. Analisis Domain Frekuensi

Domain frekuensi merupakan representasi dari dekomposisi suatu sinyal menjadi komponen-komponen sinusoidal atau eksponensial kompleks. Domain frekuensi berfungsi untuk menganalisis data, fokusnya pada analisis fungsi matematis atau sinyal yang berhubungan dengan frekuensi. Respon frekuensi yang didapatkan dari suatu sistem dapat dianalisis dengan dua cara yaitu dengan plot bode atau melalui diagram nyquist.

2.6.1. Bode

Diagram bode atau plot bode merupakan suatu grafik frekuensi dari suatu respon sistem. Diagram bode ini dinyatakan dalam dua diagram yang terpisah yaitu diagram besaran (magnitude) dan diagram sudut fase. Untuk menganalisa kestabilan sistem menggunakan Bode dapat dilihat dari gain margin dan phase margin. Berikut syarat-syarat kestabilan dari suatu sistem ditinjau dari gain margin GM dan phase margin PM:

- Jika GM > 1 dan PM > 0 maka sistem tersebut stabil
- Jika GM = 1 dan PM = 0 maka sistem tersebut stabil marginal
- Jika GM < 1 dan PM < 0 maka sistem tersebut tidak stabil
- GM tak hingga akan menghasilkan sistem yang selalu stabil

Setelah diketahui syarat-syarat kestabilan di atas, sistem akan diuji dengan sinyal step untuk sistem close loop
Berikut script MATLAB untuk menentukan gain margin dan phase margin:
Contoh 1:
Transfer function plant open-loop:

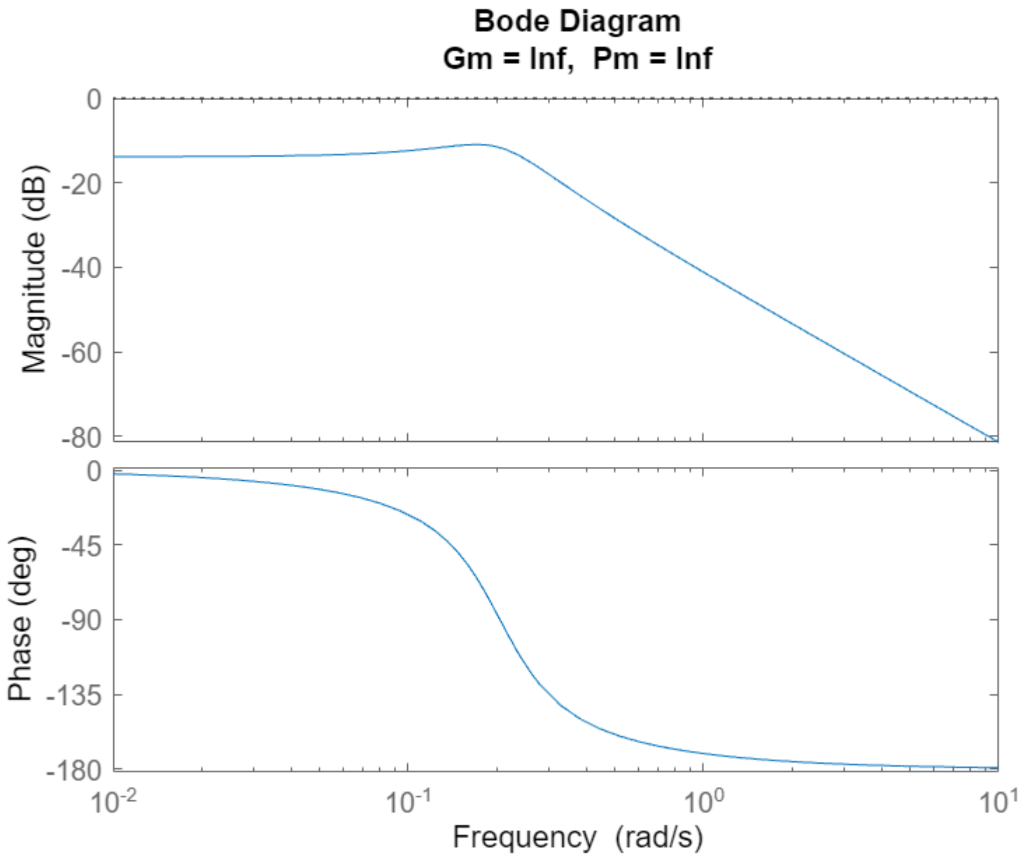
$$G(s) = \frac{0.008519}{s^2+0.1593s+0.0418}$$

Definisikan numerator dan denumerator dari transfer function

```
num = [0.008519];
den = [1 0.1593 0.0418];
G = tf(num,den)
```

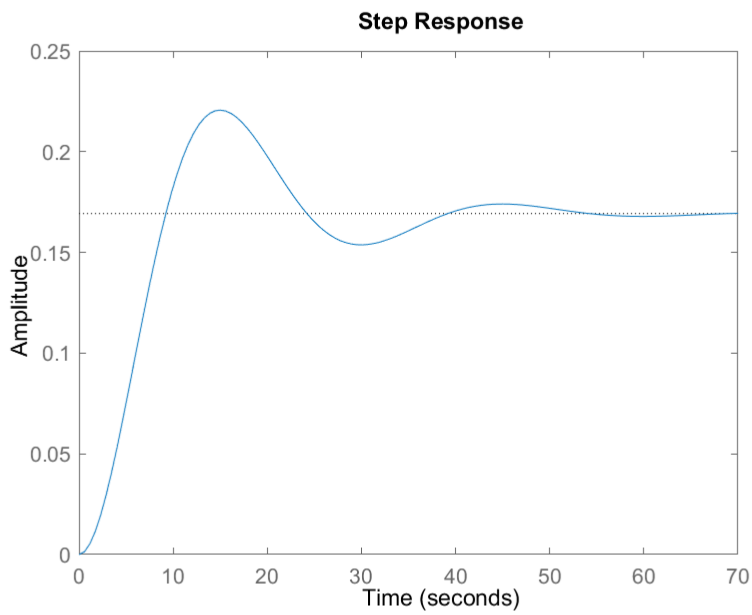
Menentukan gain margin dan phase margin

```
margin(G)
```



Berdasarkan GM dari bode diagram di atas, sistem dikatakan stabil

```
step(feedback(G,1))
```

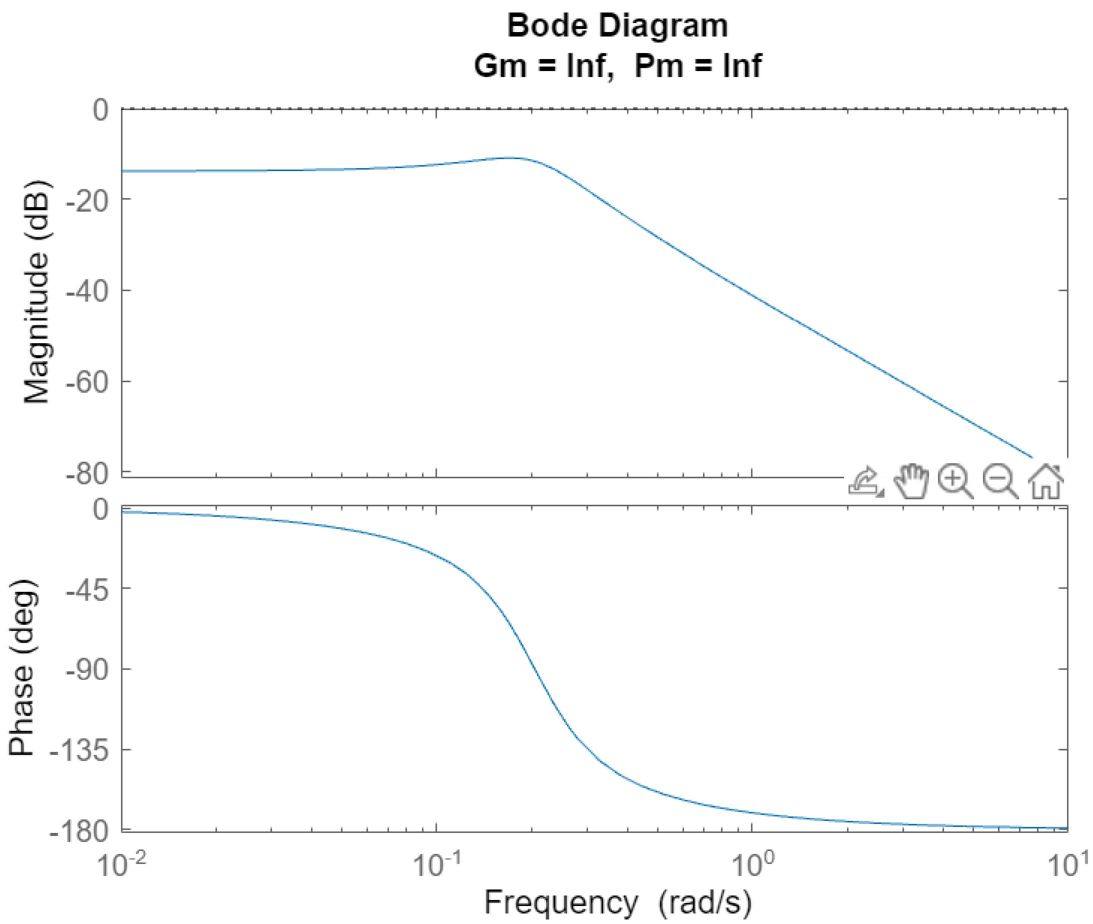


Contoh 2
Transfer function plant open-loop:
Definisikan numerator dan denumerator dari transfer function

```
num= [1];
den= [1 1];
P = tf(num,den)
```

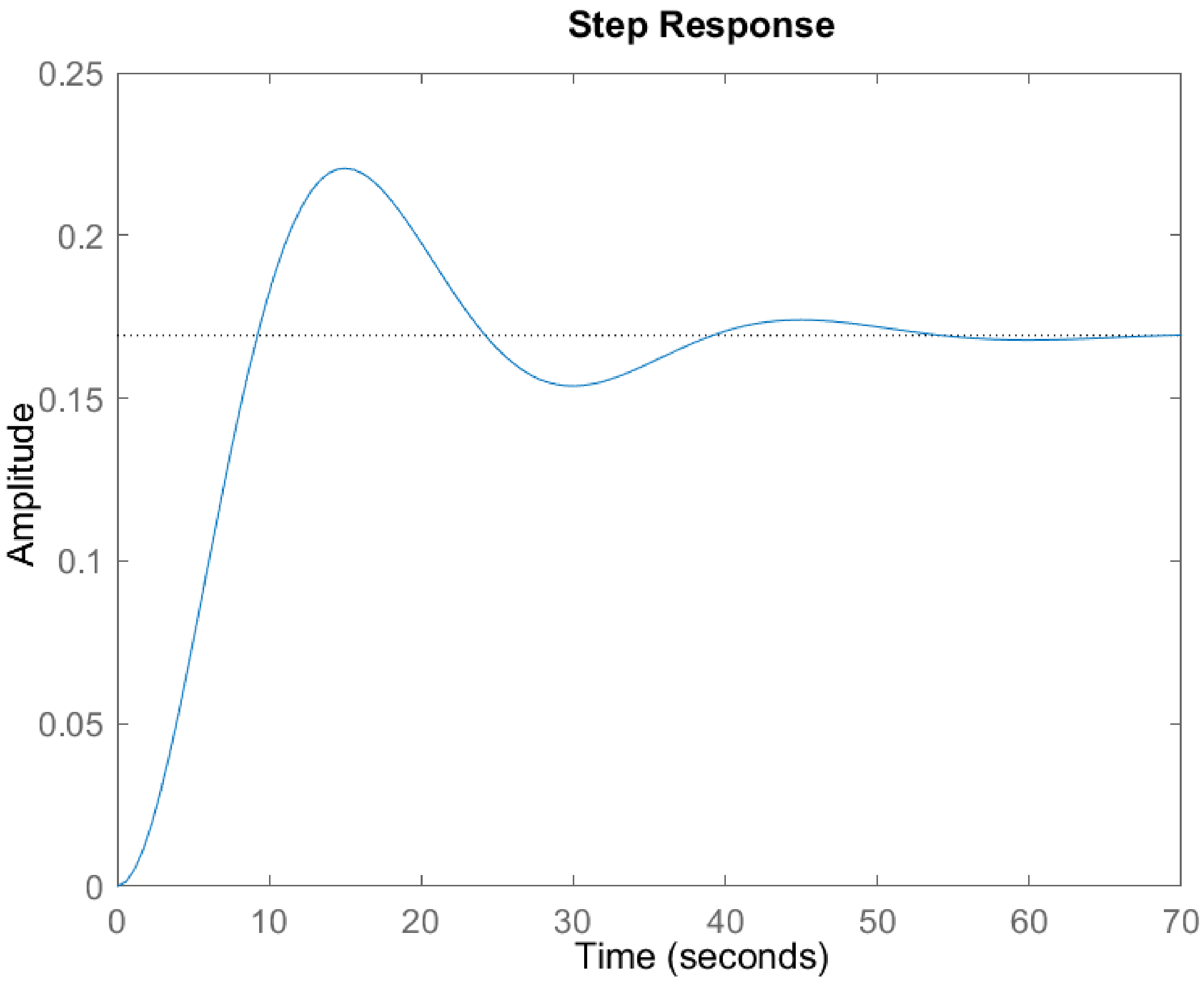
Menentukan gain margin dan phase margin

```
margin(P)
```



Berdasarkan **GM** dari bode diagram di atas, sistem dikatakan stabil

```
step(feedback(P,1))
```

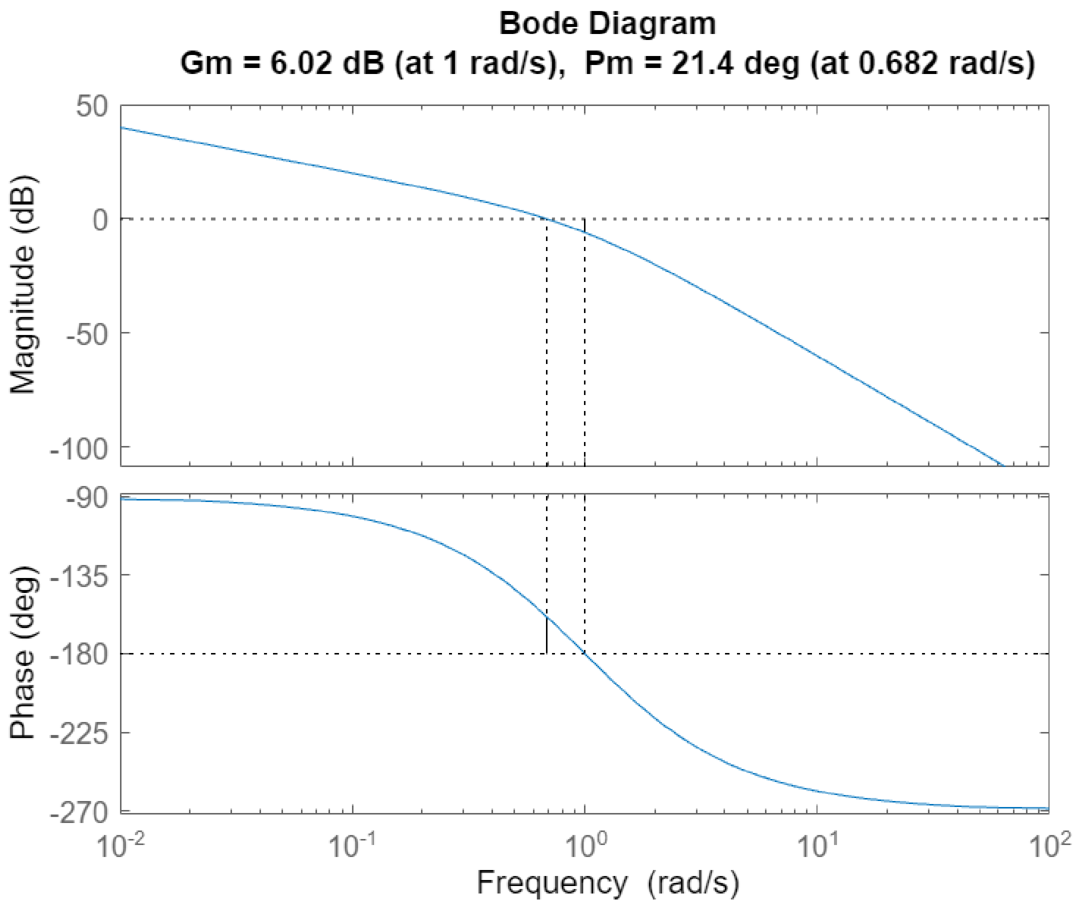


Contoh 3
Transfer function plant open-loop:
Definisikan numerator dan denumerator dari transfer function

```
a = [1];
b = [1 2 1 0];
H = tf(a,b)
```

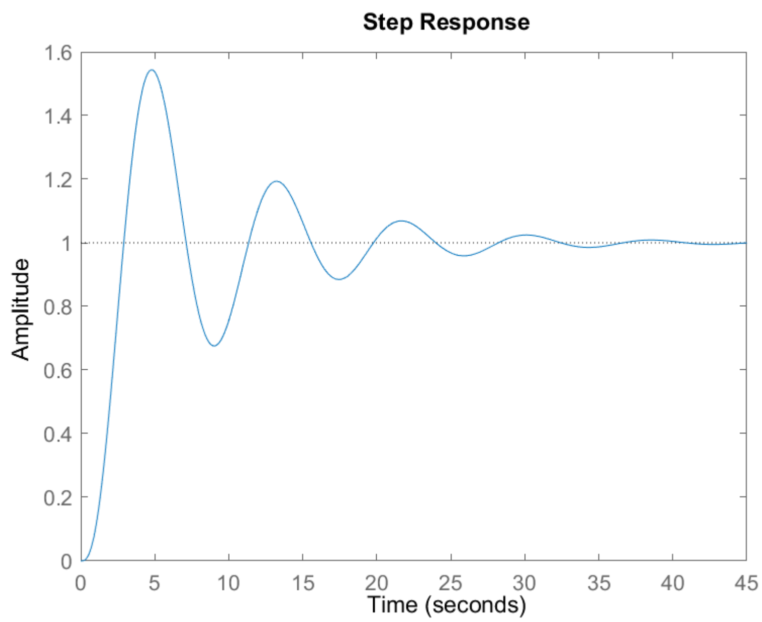
Menentukan gain margin dan phase margin

```
margin(H)
```



Karena GM > 1 dan PM > 0 maka sistem tersebut stabil

```
step(feedback(H,1))
```



Selain dengan keempat syarat tadi, kita juga bisa menggunakan fungsi allmargin di MATLAB. Untuk melihat kestabilan, bisa dilihat di bagian Stable jika 1 berarti sistem stabil dan jika 0 berarti sistem tidak stabil atau stabil marginal.

```
allmargin(G)

allmargin(P)

allmargin(H)
```

2.6.2. Nyquist

Plot Nyquist berfungsi untuk memprediksi kestabilan dan performansi dari sistem loop tertutup dengan mengamati karakteristik dari sistem loop terbuka. Kriteria nyquist ini digunakan untuk mendesain spesifikasi tanpa memperhatikan kestabilan loop terbuka atau dengan kata lain kriteria nyquist digunakan untuk menentukan kestabilan loop tertutup saat gambaran plot bode memberikan hasil yang tidak meyakinkan atau membingungkan.

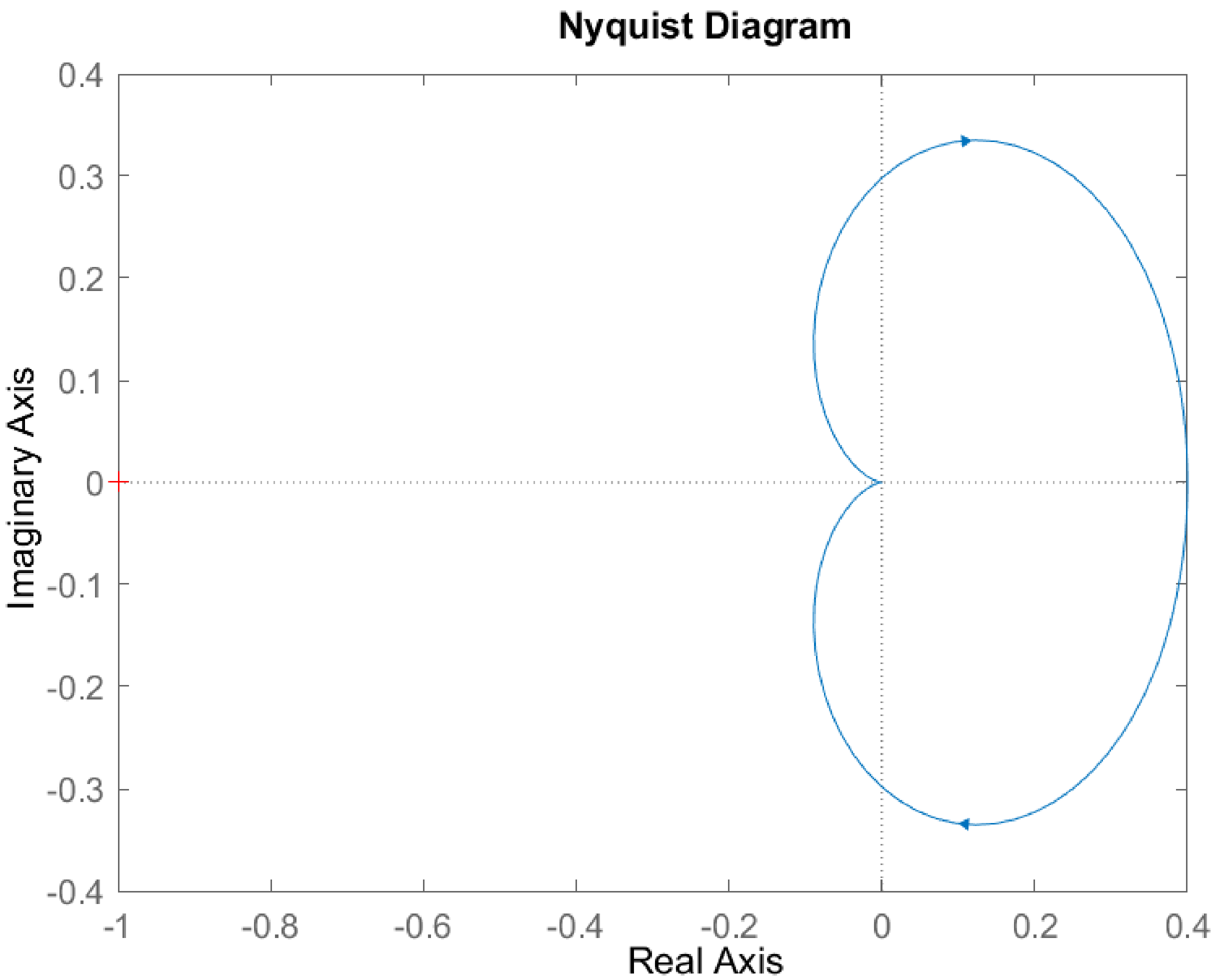
Dalam menentukan kestabilan suatu sistem dengan menggunakan diagram nyquist terdapat beberapa kemungkinan diantaranya adalah

1. Tidak melingkupi titik -1+j0. Sistem stabil jika tidak ada pole yang terletak di sebelah kanan sumbu J.
2. Ada satu atau lebih yang melingkupi titik -1+j0 berlawanan arah jarum jam. Sistem stabil jika jumlah pengelilingan titik itu sama dengan jumlah pole yang di sebelah kanan j.
3. Ada satu atau lebih yang melingkupi titik -1+j0 searah jarum jam. Sistem tersebut tidak stabil

Contoh 1

Transfer function : $\frac{2}{s^2+3s+5}$

```
G=tf(2,[1 3 5])

nyquist(G)
```



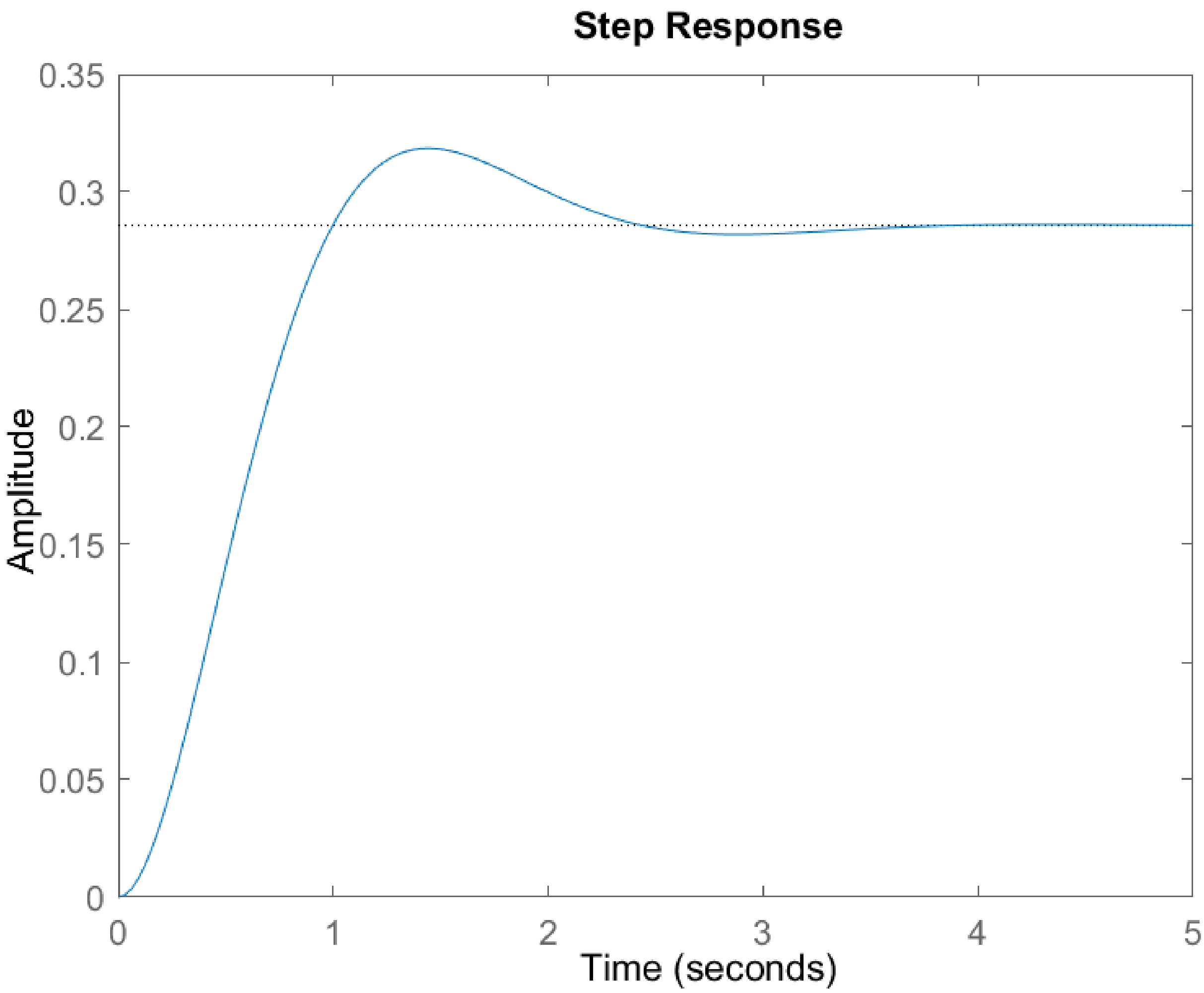
Dari diagram nyquist tersebut dapat dilihat bahwa tidak ada yang mengelilingi titik -1+j0, maka diagram tersebut masuk ke kategori 1 dan sekarang tinggal dicari pole nya

```
pole(G)
```

```
ans = 2×1 complex
 -1.5000 + 1.6583i
 -1.5000 - 1.6583i
```

Dari pole yang didapatkan dapat disimpulkan bahwa plant stabil karena semua pole berada disebelah kiri sumbu imajiner, sekarang tinggal kita plot untuk close loop

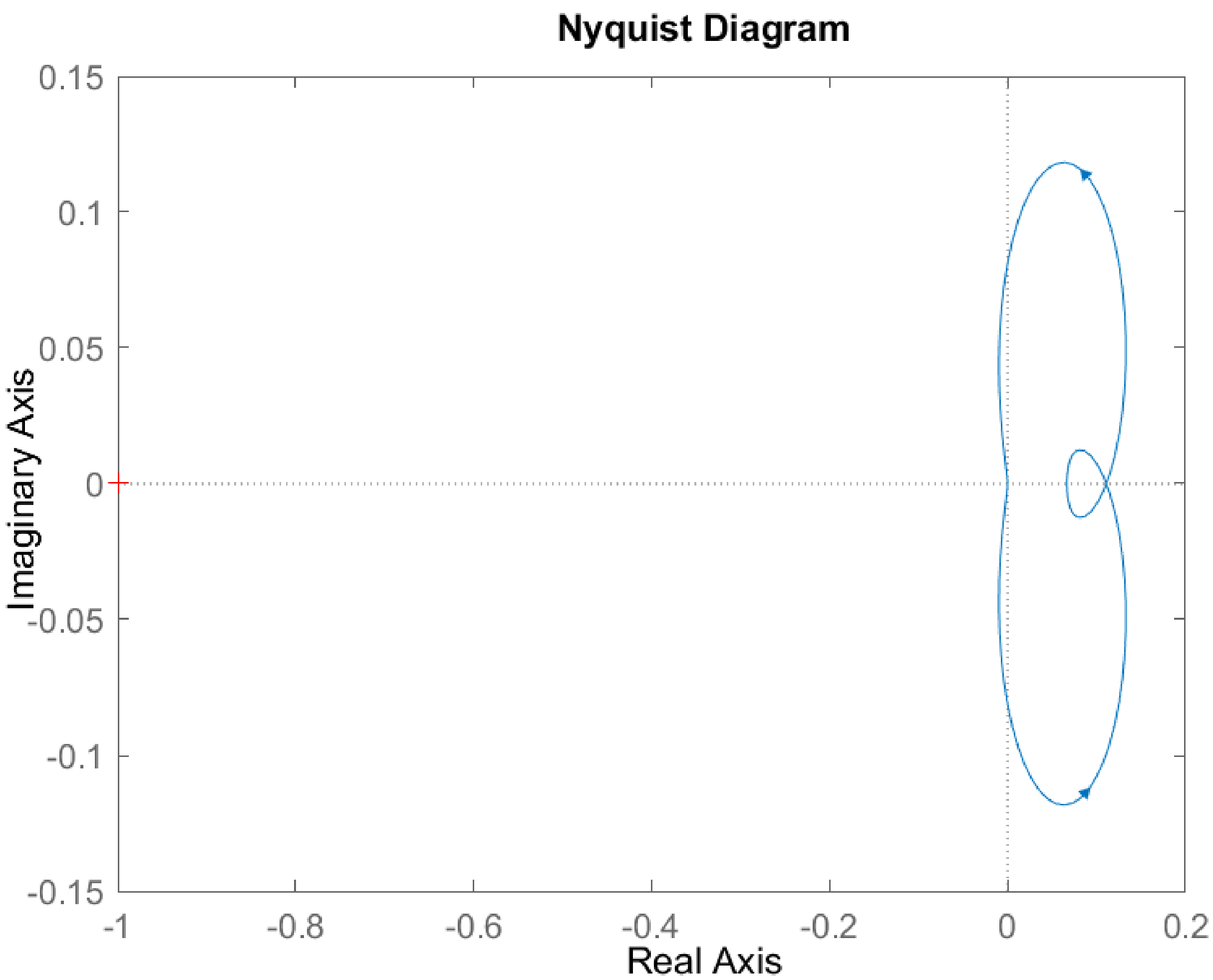
```
step(feedback(G,1))
```



Contoh 2

Transfer function : $\frac{1}{s^3+2s^2+3s+15}$

```
P=tf(1,[1 2 3 15])
nyquist (P)
```



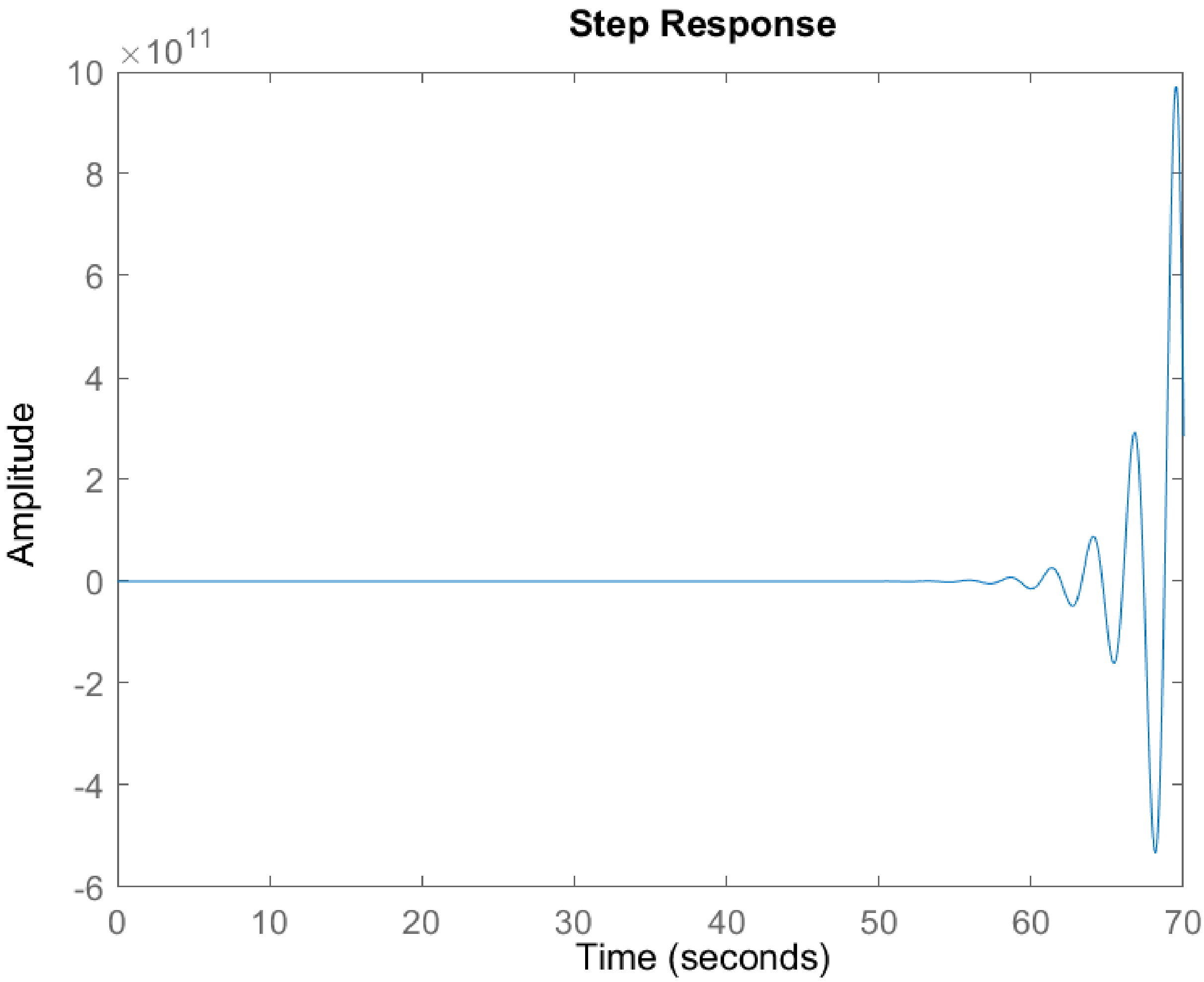
Dari diagram nyquist tersebut dapat dilihat bahwa tidak ada yang mengelilingi titik -1+j0, maka diagram tersebut masuk ke kategori 1 dan sekarang tinggal dicari pole nya

```
pole(P)
```

```
ans = 3×1 complex
 -2.8212 + 0.0000i
  0.4106 + 2.2690i
  0.4106 - 2.2690i
```

Dari pole yang didapatkan dapat disimpulkan bahwa plant tidak stabil karena terdapat 2 pole yang berada di sebelah kanan sumbu imajiner, sekarang tinggal kita buktikan dengan melakukan plot untuk close loop

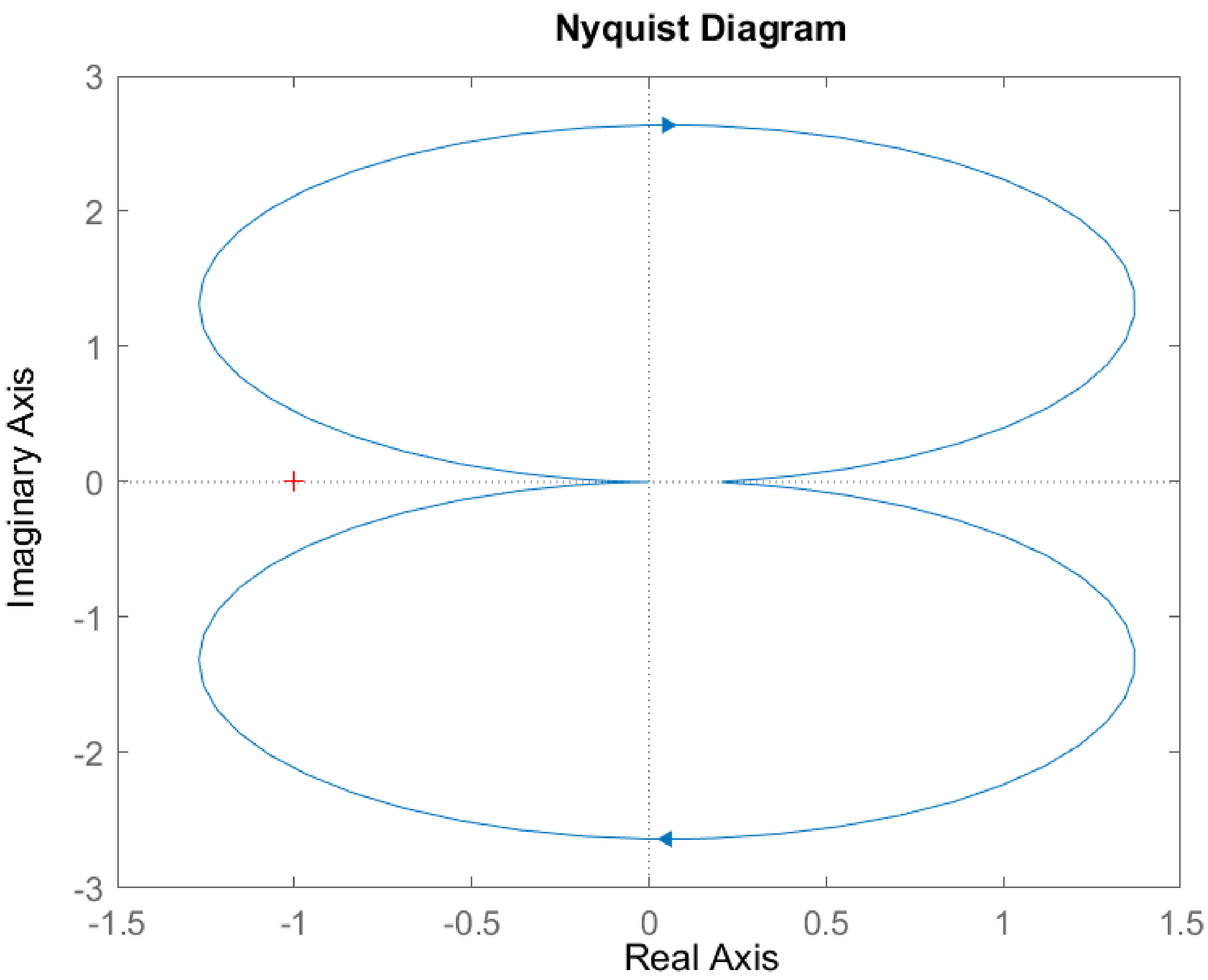
```
step(feedback(P,1))
```



Contoh 3

Transfer function : $\frac{0.008519}{s^2+0.1593s+0.0418}$

```
H=tf(0.00859,[1 0.01593 0.0418])

nyquist (H)
```



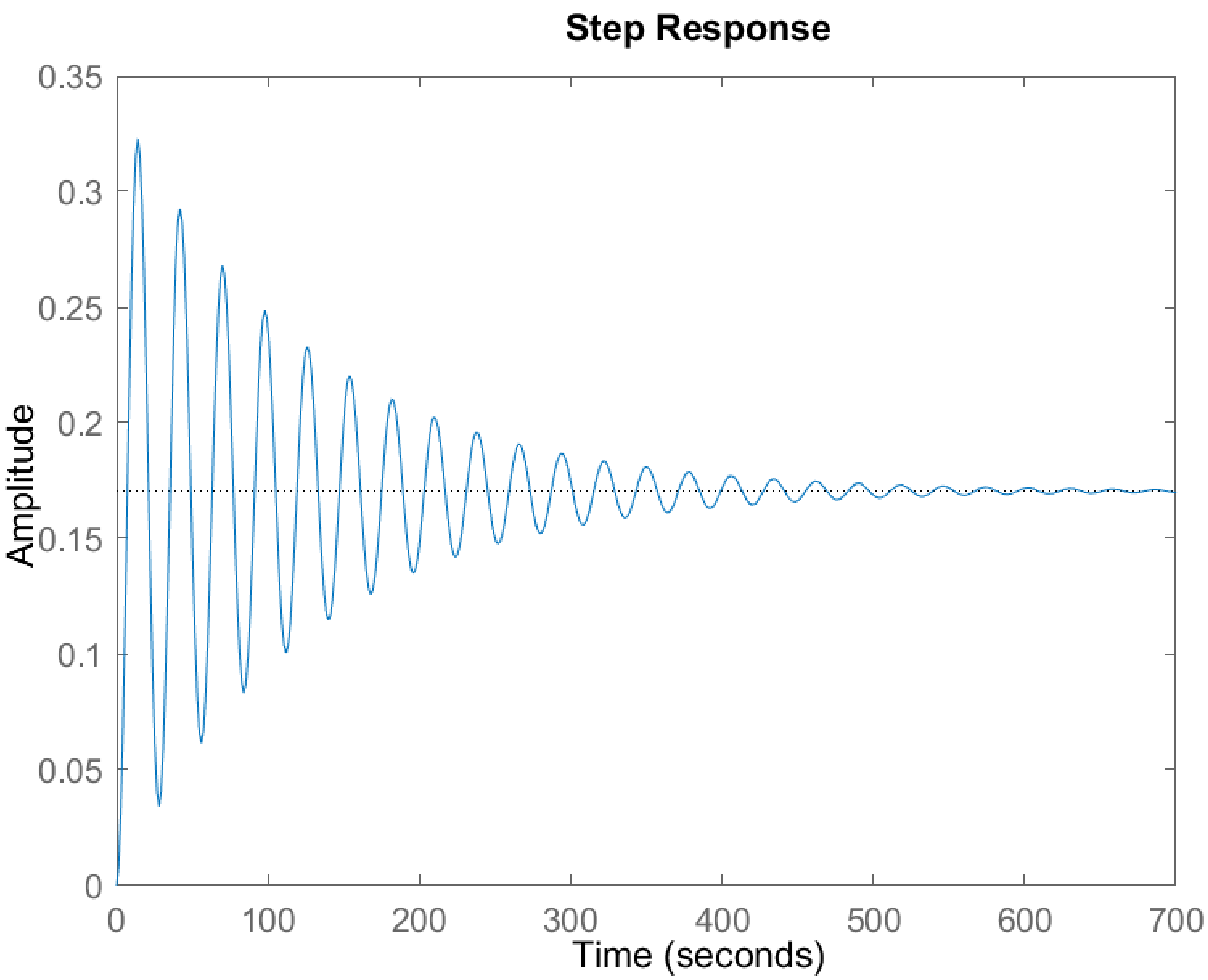
Dari diagram nyquist tersebut dapat dilihat bahwa tidak ada yang mengelilingi titik -1+j0, maka diagram tersebut masuk ke kategori 1 dan sekarang tinggal dicari pole nya

```
pole(H)
```

```
ans = 2×1 complex
 -0.0080 + 0.2043i
 -0.0080 - 0.2043i
```

Dari pole yang didapatkan dapat disimpulkan bahwa plant stabil karena semua pole berada disebelah kiri sumbu imajiner, sekarang tinggal kita plot untuk close loop

```
step(feedback(H,1))
```

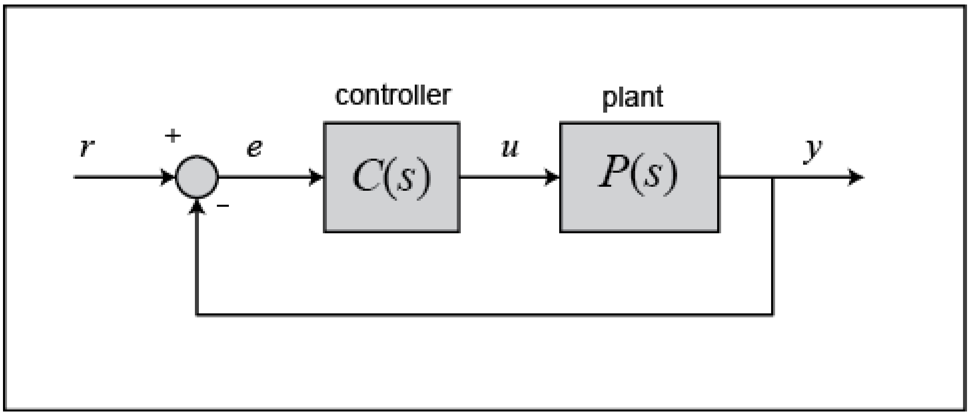

Step Response
Amplitude
Time (seconds)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0
100
200
300
400
500
600
700

BAB 3. Kontroller PID

3.1. Kontroler PID

3.1.1. Pengenalan Kontroler PID

Berikut adalah sistem closed-loop unity-feedback dengan plant sistem adalah P(s) dan controller C(s).



Sinyal u merupakan output dari PID controller yang berperan sebagai control input bagi plant. Persamaan control input adalah sebagai berikut:

$$u(t) = K_p e(t) + K_i \int e(t) + K_d \frac{de(t)}{dt}$$

Variabel e adalah sinyal error yang merupakan selisih antara input/referensi r dan output y. Sinyal error e diinputkan ke PID controller. Di dalam PID controller, proportional gain K_p dikalikan dengan magnitudo sinyal error e, dijumlahkan dengan integral gain K_i dikalikan dengan integral error, dijumlahkan dengan K_d dikalikan dengan turunan error.

Sinyal kontrol u yang dihasilkan akan masuk ke plant dan didapatkan output y. Output y di-umpan balik dan dibandingkan dengan sinyal referensi r untuk mendapatkan sinyal error e yang baru.

Transfer function PID controller dalam domain laplace:

$$\frac{U(s)}{E(s)} = K_p + K_i \frac{1}{s} + K_d s$$

Script untuk menghasilkan PID transfer function:

```
Kp = 1;

Ki = 1;

Kd = 1;

C = pid(Kp,Ki,Kd)
```

```
C =

              1
  Kp + Ki * --- + Kd * s
              s

  with Kp = 1, Ki = 1, Kd = 1

Continuous-time PID controller in parallel form.
```

3.1.2. Karakteristik Performa Kontroler PID

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Small Change	Decrease	Decrease	No Change

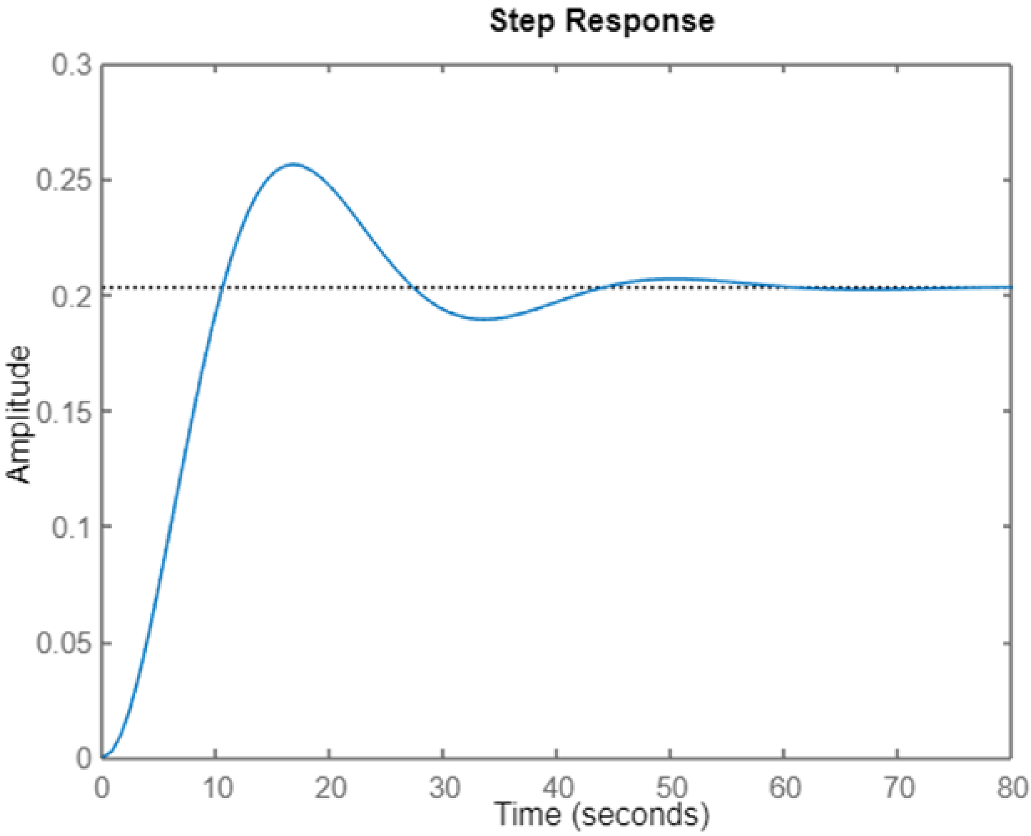
3.1.3. Contoh Permasalahan

Transfer function plant yang digunakan

$$P(s) = \frac{0.008434}{s^2+0.16s+0.04152}$$

Respon open loop sistem:

```
s = tf('s');
P = 0.008434/(s^2 + 0.16*s + 0.04152);
step(P,80)
```

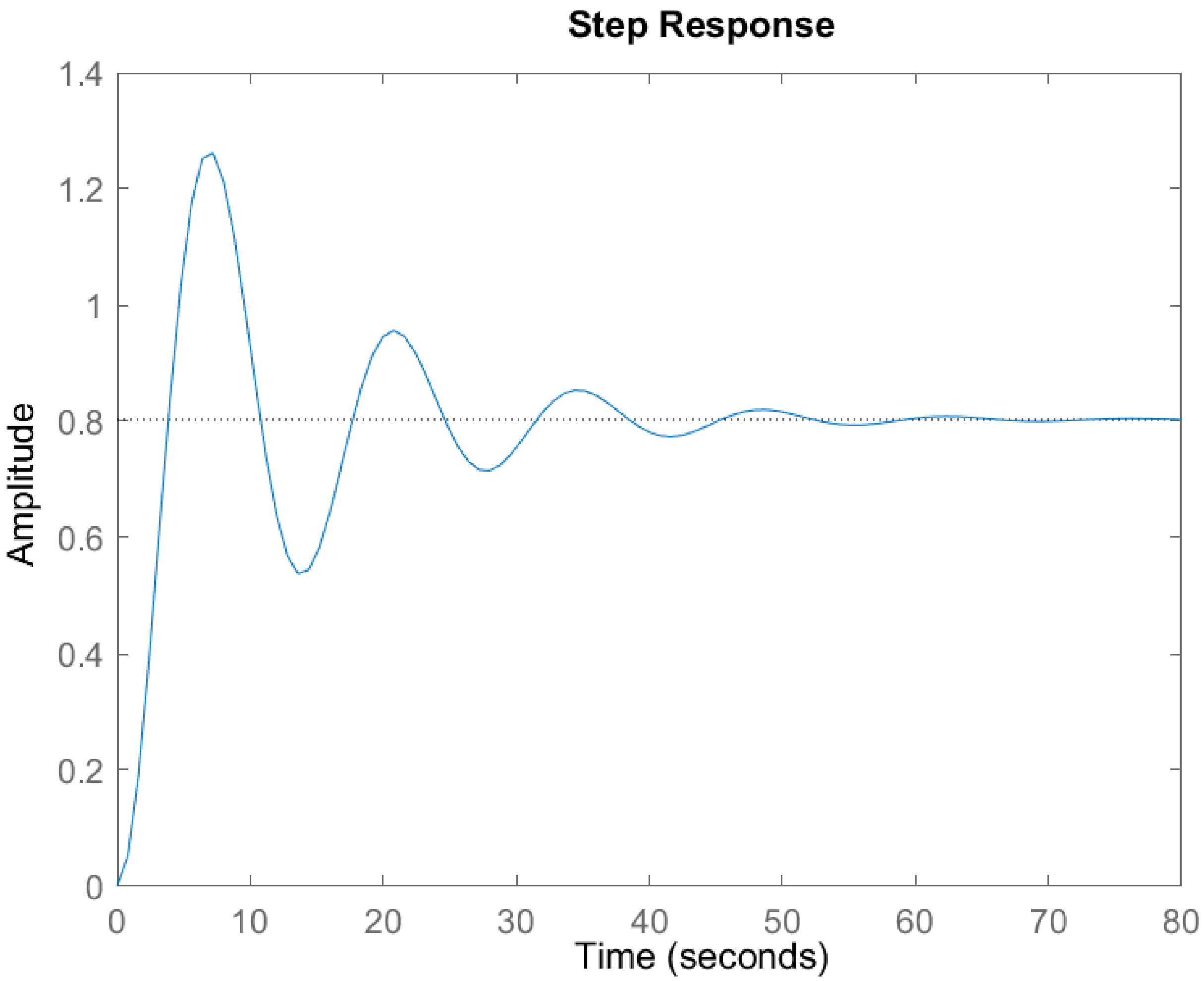


Nilai steady state dari step response diatas adalah sekitar 0.203 menunjukkan error steady state sekitar 98%. Selain itu settling time (±2%) disekitar 41 detik dan rise time (10%-90%) disekitar 7 detik.
Selanjutnya akan dipelajari pengaruh masing-masing parameter kombinasi P, PD, PI, PID pada sistem di atas.

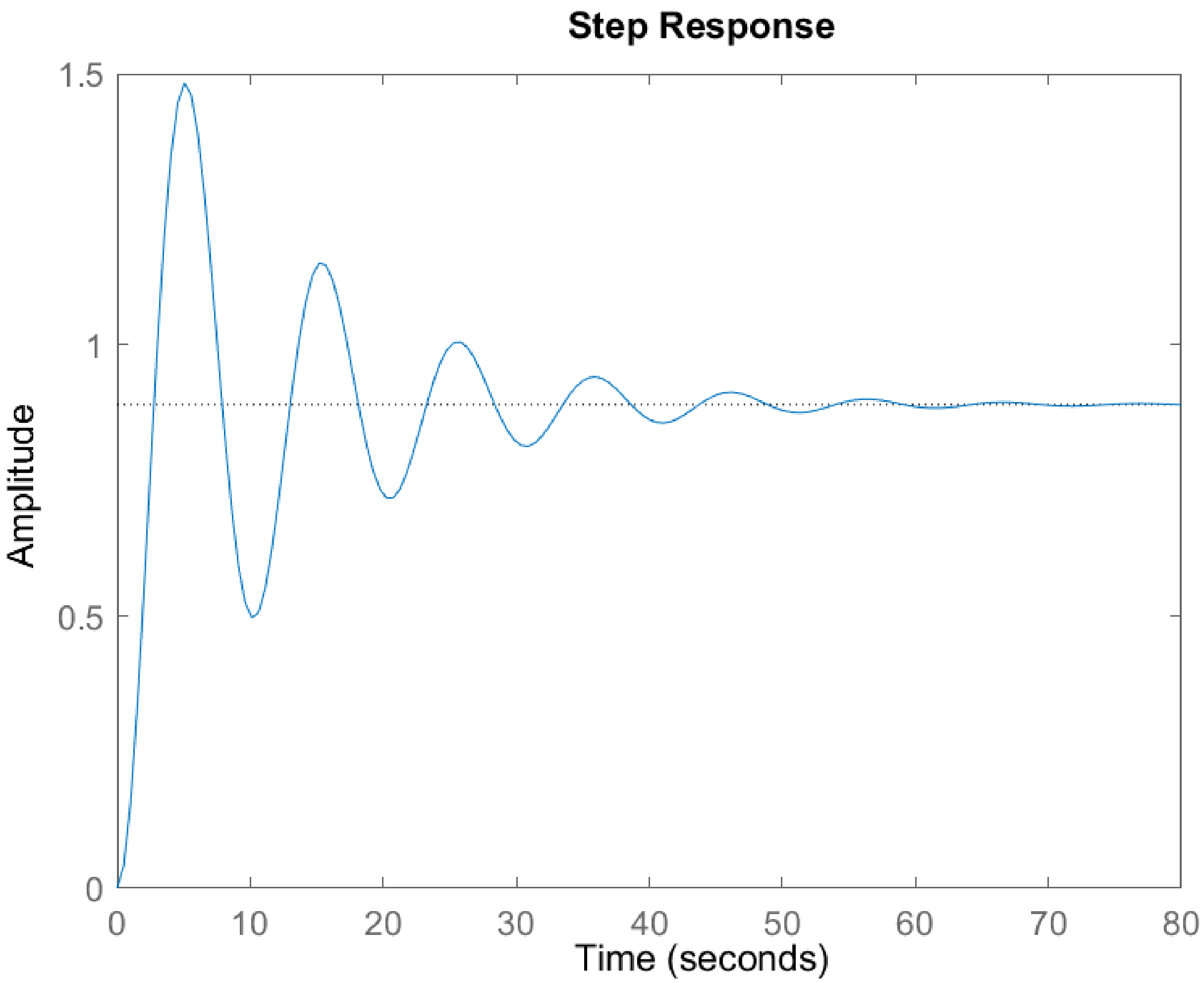
3.1.4. Proportional Control

Berdasarkan tabel di atas, parameter kontroler proporsional K_p akan mengurangi rise time dan error steady-state serta meningkatkan overshoot. Transfer function loop tertutup untuk sistem unity-feedback dengan kontroler proporsional adalah sebagai berikut:

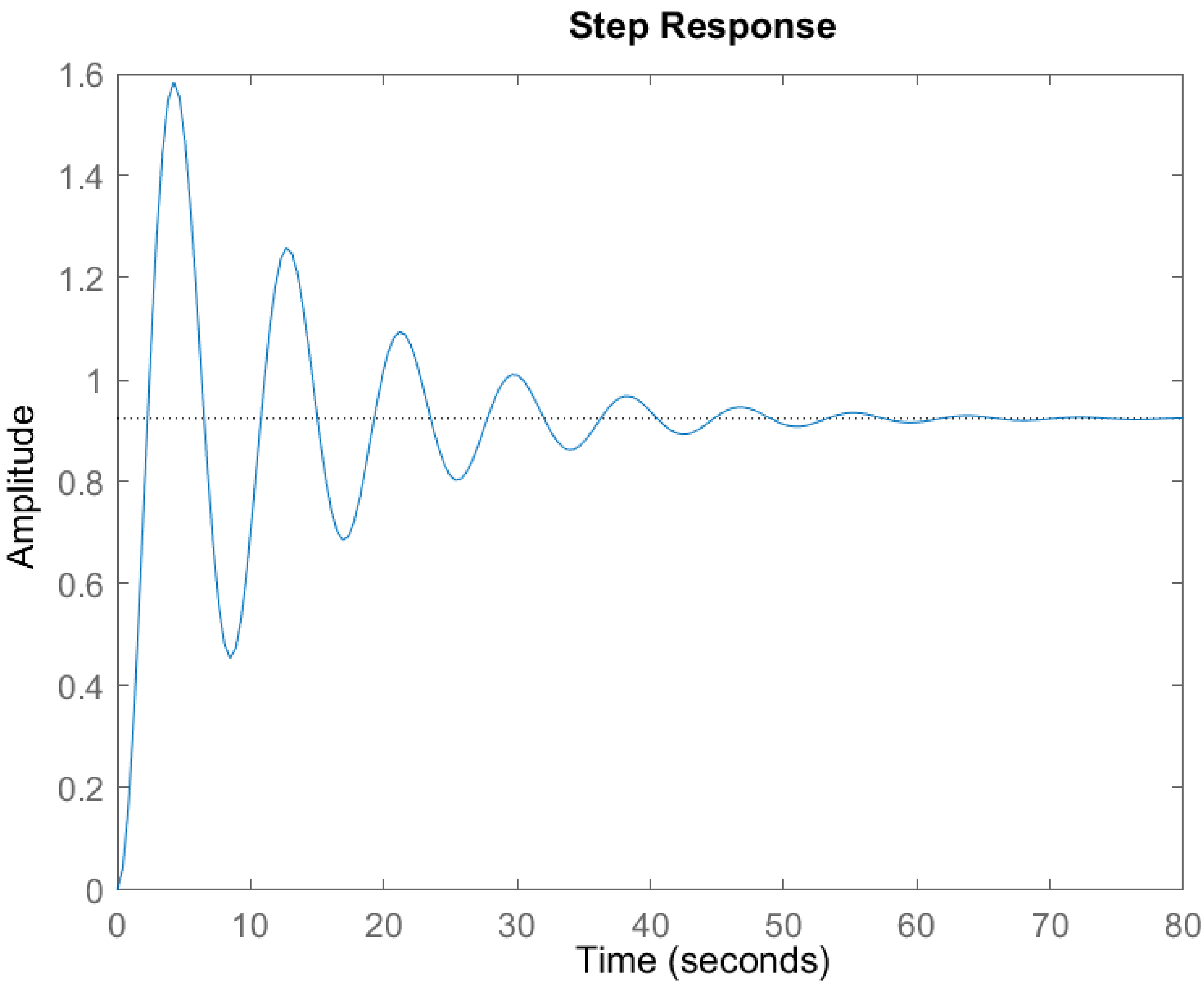
```
Kp = 20;

C = pid(Kp);

T = feedback(C*P,1);

step(T,80)
```




```
Kp = 40;
C = pid(Kp);
T = feedback(C*P,1);
step(T,80)
```




```
Kp = 60;
C = pid(Kp);
T = feedback(C*P,1);
step(T,80)
```

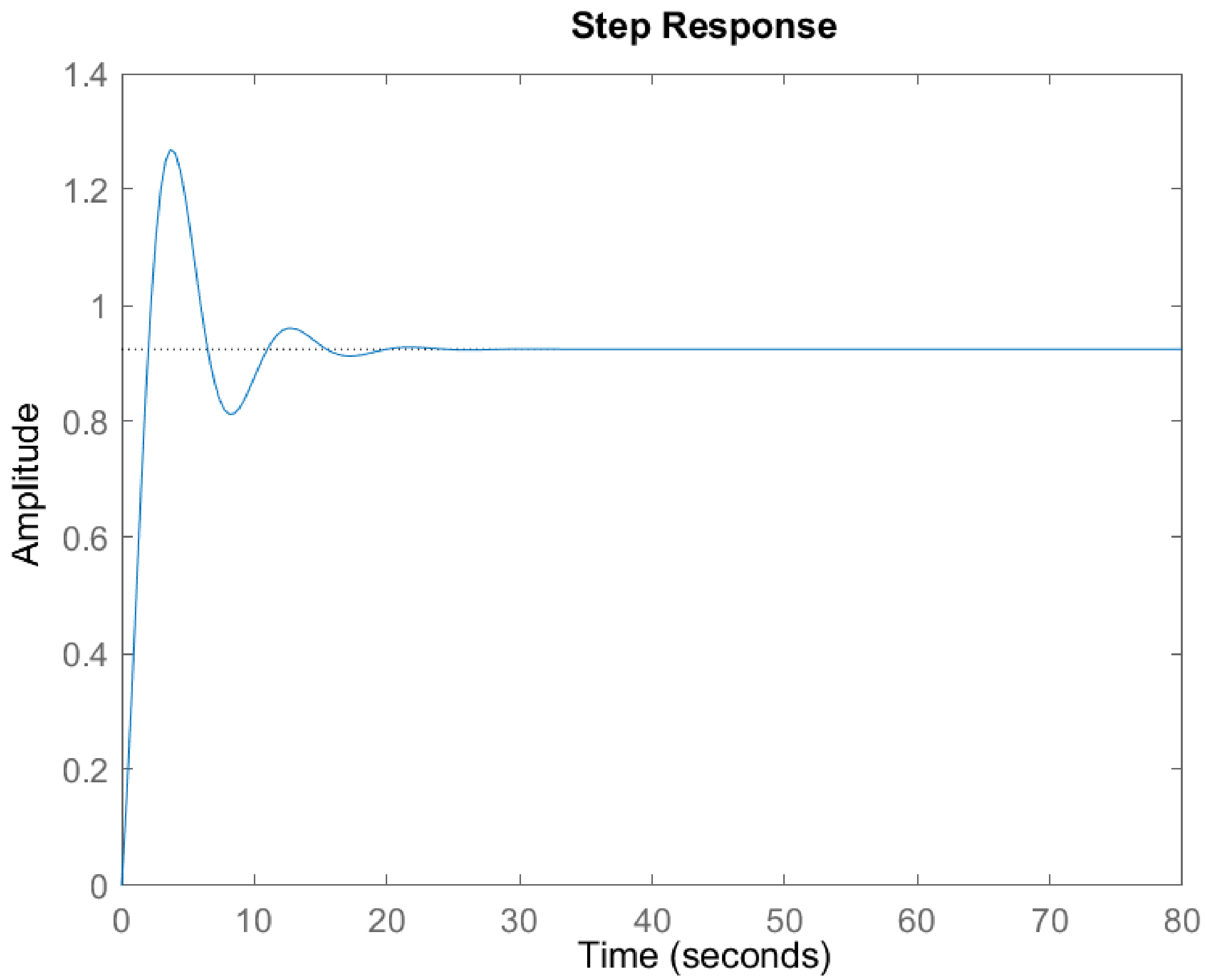


Dapat dilihat dari plot step response bahwa kenaikan K_p mengurangi rise time dan steady-state error, serta meningkatkan overshoot.

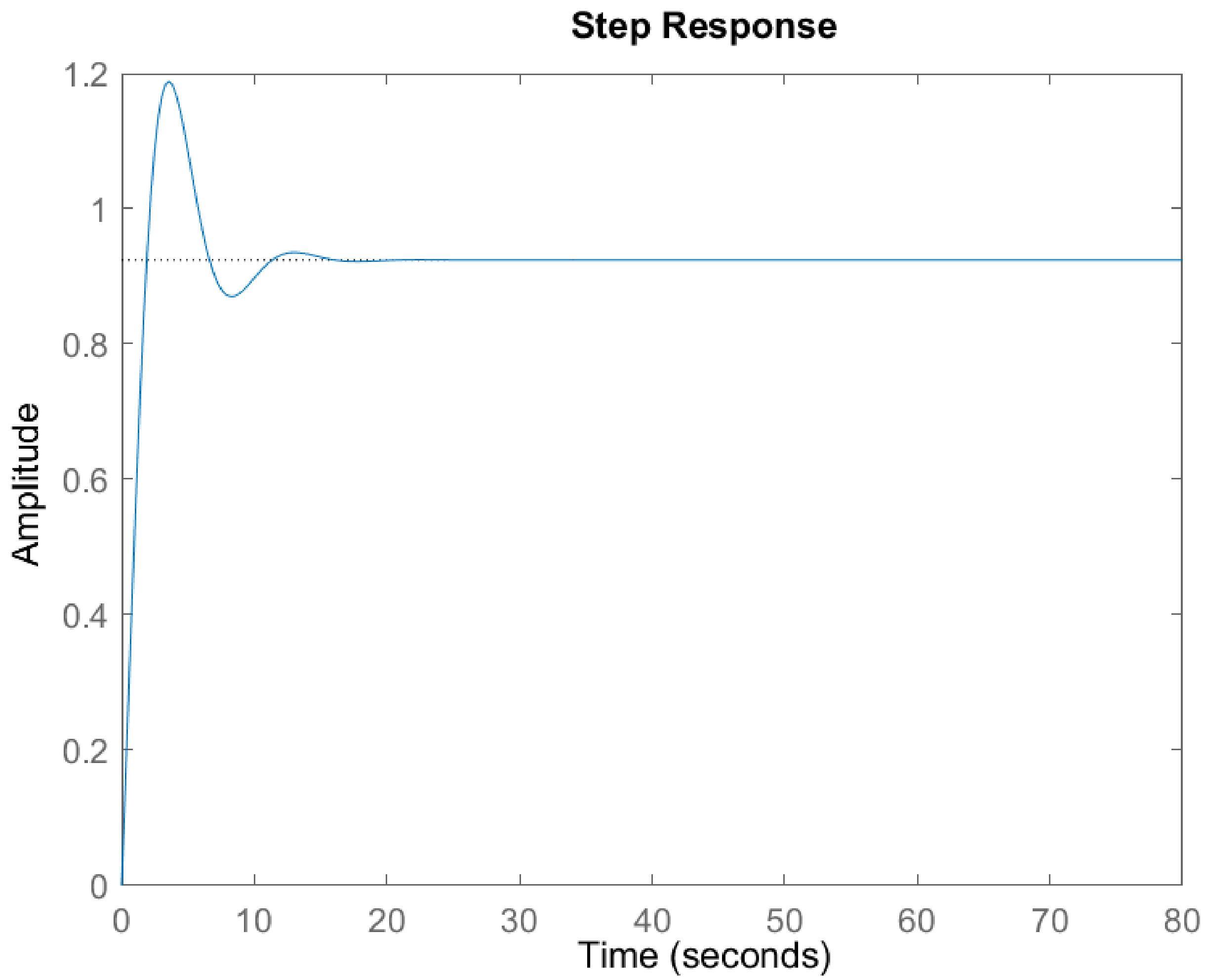
3.1.5. Proportional-Derivative Control

Berdasarkan tabel di atas, penambahan parameter kontroler derivative K_d akan mengurangi overshoot dan settling time. Transfer function loop tertutup untuk sistem unity-feedback dengan kontroler proporsional adalah sebagai berikut:

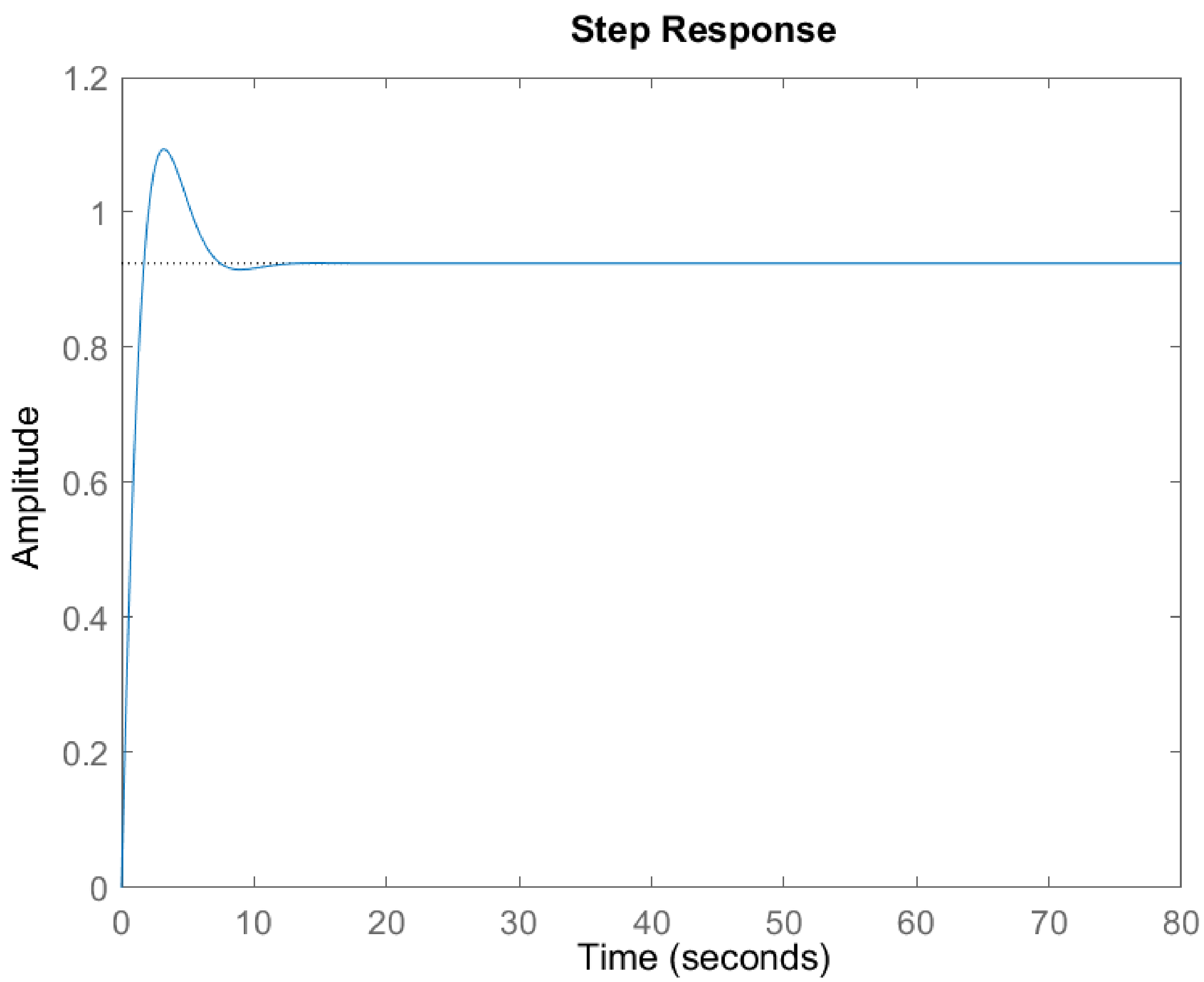
```
Kp = 60;
Kd = 40;
C = pid(Kp,0,Kd);
T = feedback(C*P,1);
step(T,80)
```




```
Kp = 60;
Kd = 60;
C = pid(Kp,0,Kd);
T = feedback(C*P,1);
step(T,80)
```




```
Kp = 60;
Kd = 100;
C = pid(Kp,0,Kd);
T = feedback(C*P,1);
step(T,80)
```



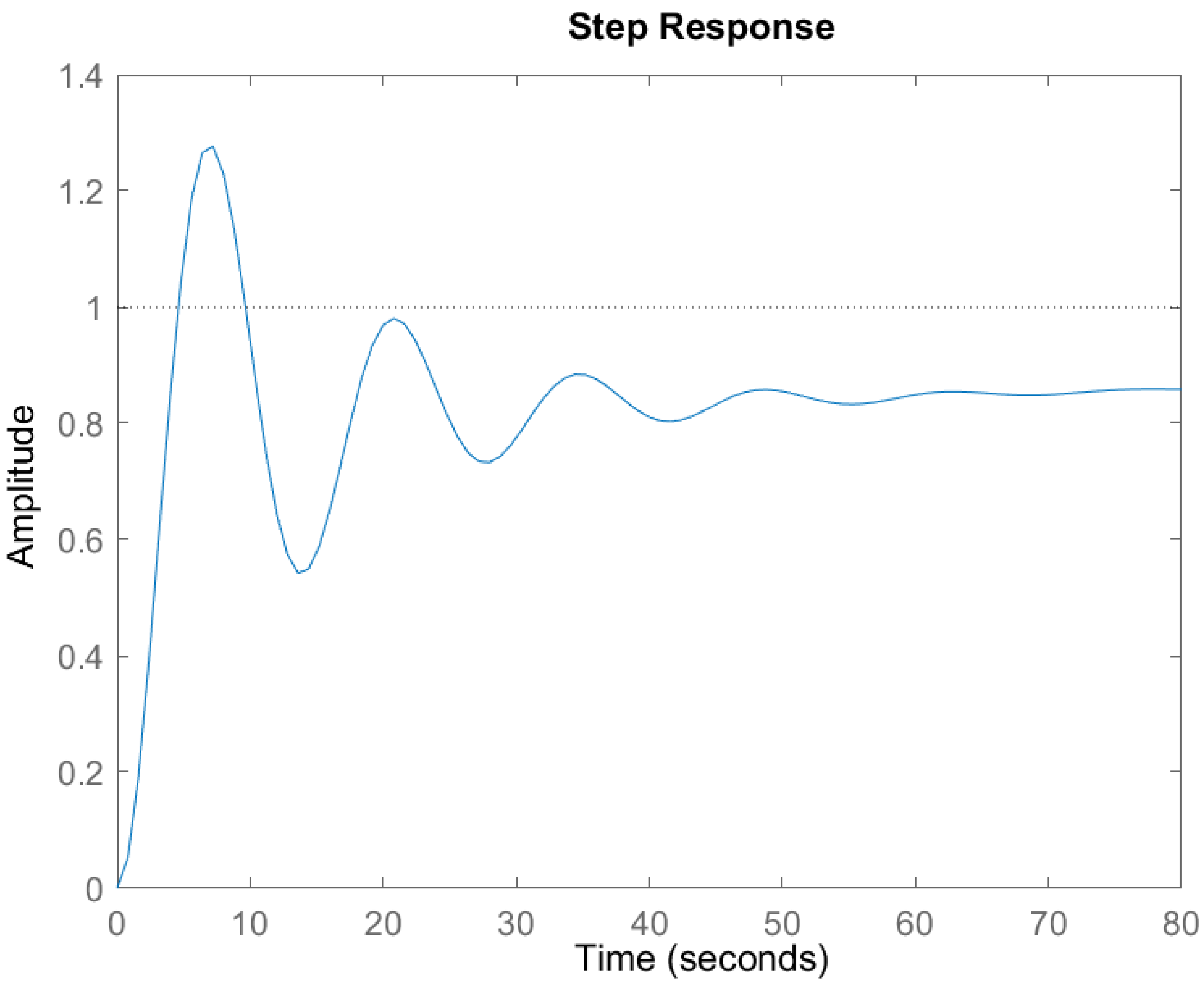
Terlihat dari plot respon step di atas, penambahan nilai parameter K_d pada kontroler PD dengan K_p konstan akan mereduksi overshoot dan settling time, namun tidak mempengaruhi rise time dan error steady state.

3.1.6. Proportional-Integral Control

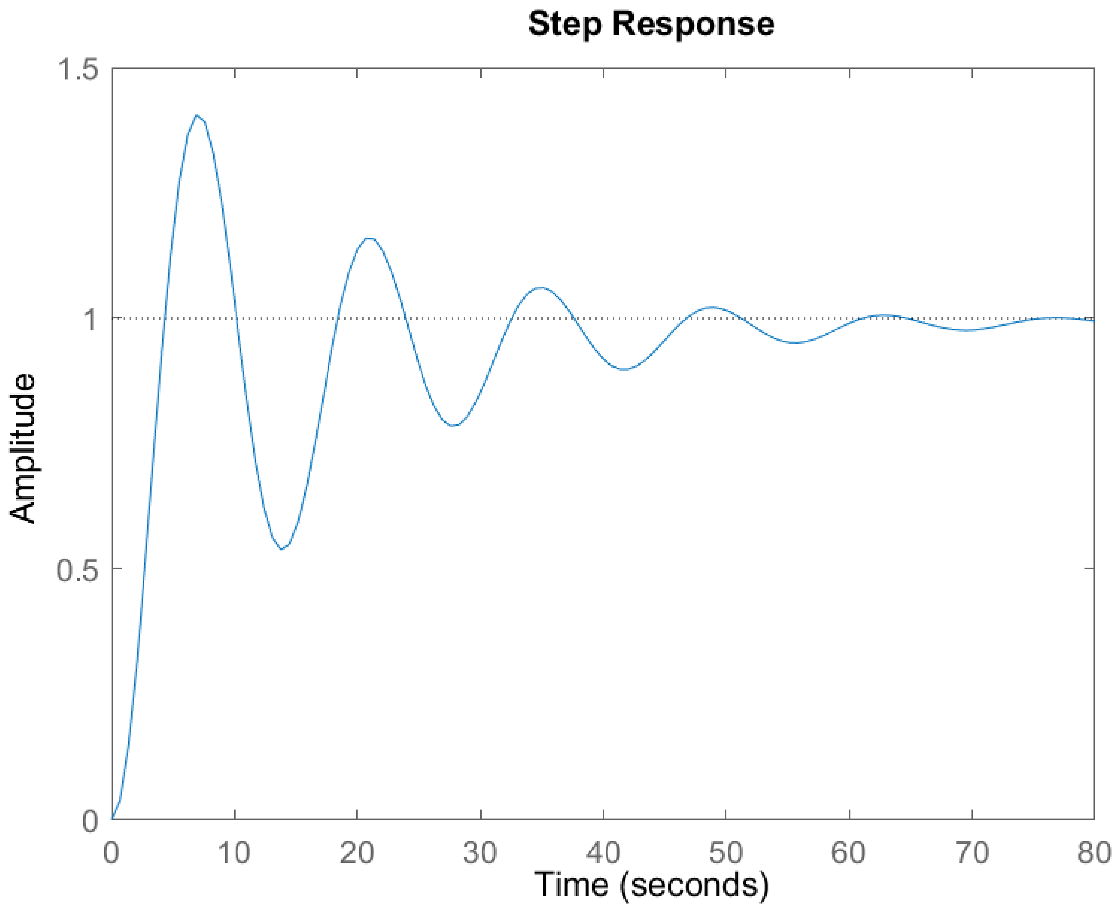
Berdasarkan tabel, penambahan parameter kontroler integral K_i cenderung akan menurunkan rise time, meningkatkan overshoot dan settling time, serta mereduksi error steady state. Karena efek parameter kontroler integral hampir mirip seperti kontroler proporsional, sehingga perlu mereduksi nilai K_p yang awalnya 60 menjadi lebih kecil.

```
Kp = 20;
Ki = 0.1;
C = pid(Kp,Ki,0);
```

```
T = feedback(C*P,1);
step(T,80)
```




```
Kp = 20;
Ki = 1;
C = pid(Kp,Ki,0);
T = feedback(C*P,1);
step(T,80)
```



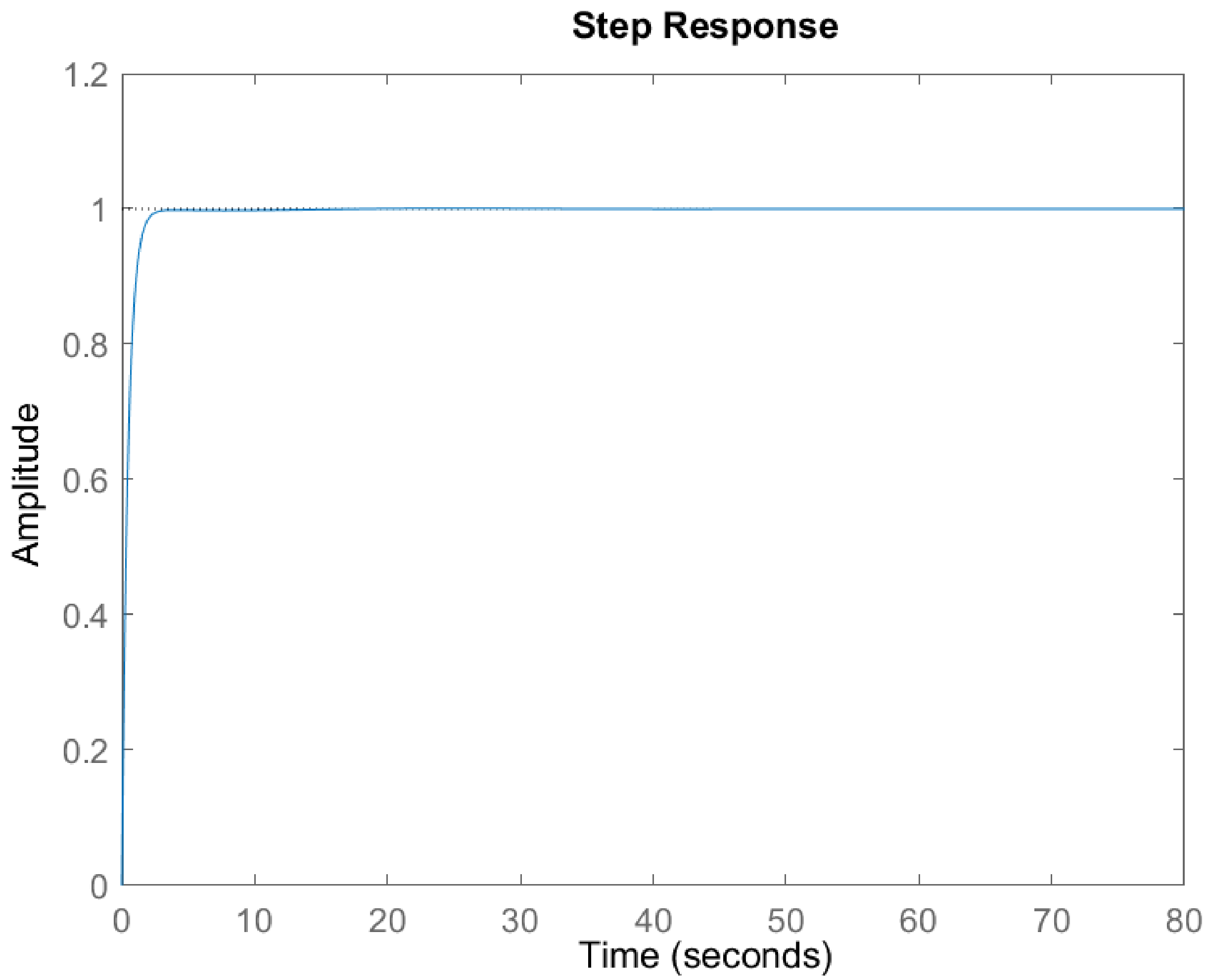
Dapat dilihat dari plot respon step di atas, penambahan kontroler integral akan membuat sistem berosilasi. Hal ini dikarenakan efek penambahan kontroler integral hampir mirip seperti efek penambahan kontroler proporsional sehingga akan mengakibatkan efek double. Selain itu dapat dilihat juga kedua kontroler ini dapat mereduksi error steady state.

3.1.7. Proportional-Integral-Derivative Control

Tujuan dalam penggunaan kontroler PID pada umumnya adalah untuk mendapatkan respon sistem yang bagus yaitu respon sistem yang tanpa overshoot, rise time cepat, dan tanpa error steady state. Setelah beberapa kali dilakukan tuning parameter PID didapatkan nilai parameter menghasilkan respon yang diinginkan adalah $K_p = 40$, $K_i = 10$, $K_d = 250$

```
Kp = 40;

Ki = 10;

Kd = 250;
```

```
C = pid(Kp,Ki,Kd);
T = feedback(C*P,1);
step(T,80)
```




```
stepinfo(T)
```

```
ans = struct with fields:
    RiseTime          : 1.0319
    TransientTime     : 1.8485
    SettlingTime      : 1.8485
    SettlingMin : 0.9001
    SettlingMax : 0.9985
    Overshoot         : 0
    Undershoot        : 0
    Peak        : 0.9985
    PeakTime          : 12.9623
```

Dapat dilihat berdasarkan respon sistem dan stepinfo, didapatkan output tanpa overshoot, rise time lebih cepat, dan tanpa error steady state.
Untuk mendapatkan respon sistem yang diinginkan maka perlu melakukan tuning parameter K_p, K_i, K_d dari kontroler PID. Terdapat beberapa metode untuk melakukan tuning parameter yang akan dibahas di sub bab berikutnya.

3.2. Desain kontroler PID metode Ziegler Nichols

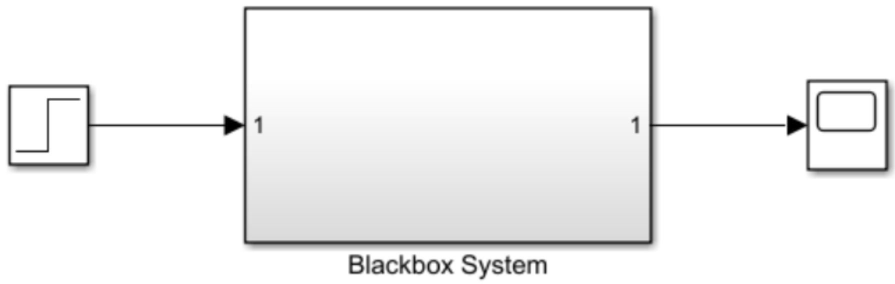
3.2.1. Ziegler-Nichols tuning 1st Method

Berikut langkah-langkah tuning metode Ziegler-Nichols metode kedua:

1. Modelkan persamaan sistem dengan metode Ziegler-Nichols dengan menarik tangent line pada inflection point sehingga didapatkan waktu delay *L* dan konstanta waktu *T*
2. Setelah didapatkan *L* dan konstanta waktu *T*, lakukan perhitungan parameter PID berdasarkan tabel di bawah ini
3. Fine tuning

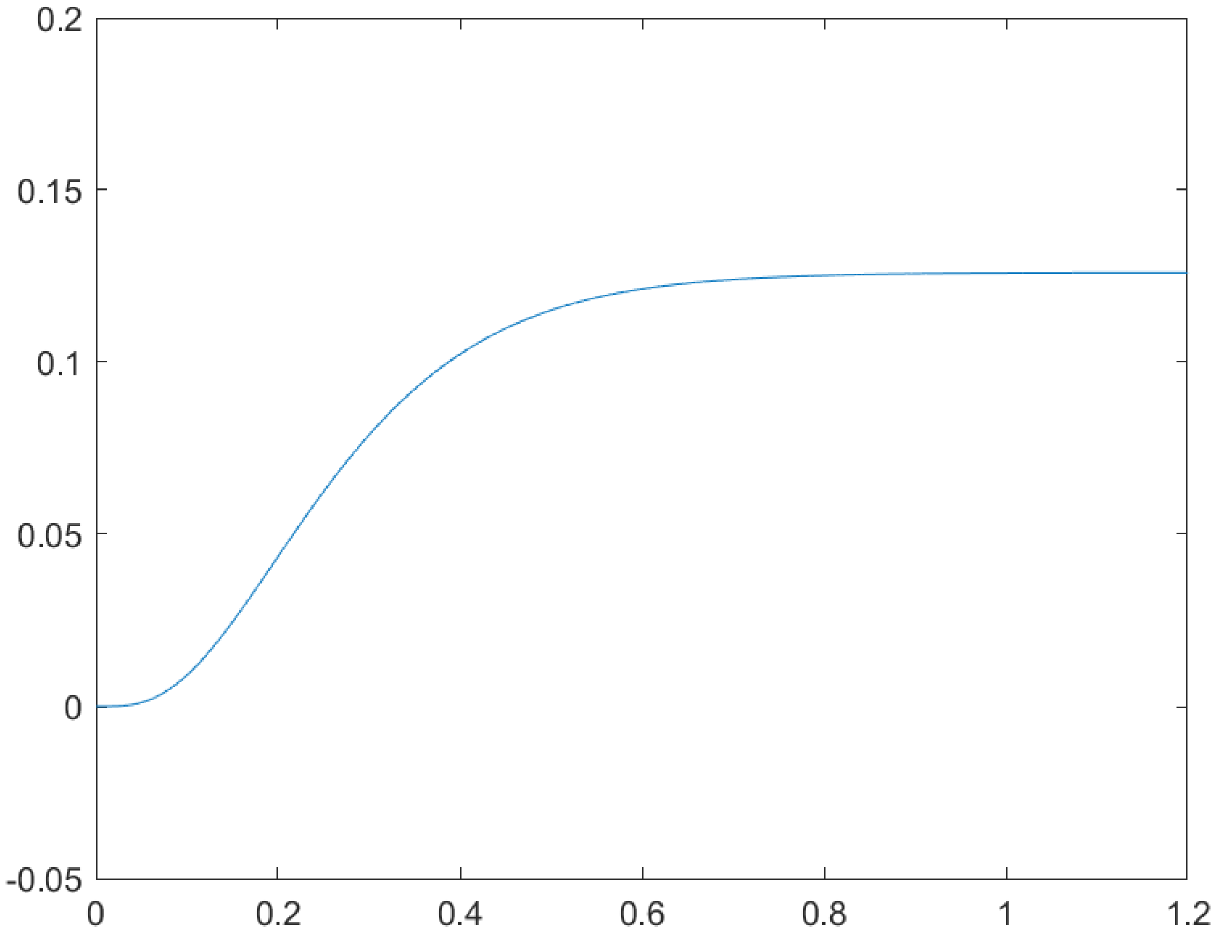
Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Contoh soal (1)
Telah di-generate sebuah data step response sistem dengan time sampling 0.00001, berikut blackbox system yang digunakan untuk generate data.



Berikut plot data-nya:

```
load('timeandoutput.mat') % load data
plot(t,Y)
axis([0 1.2 -0.05 0.2]);
```

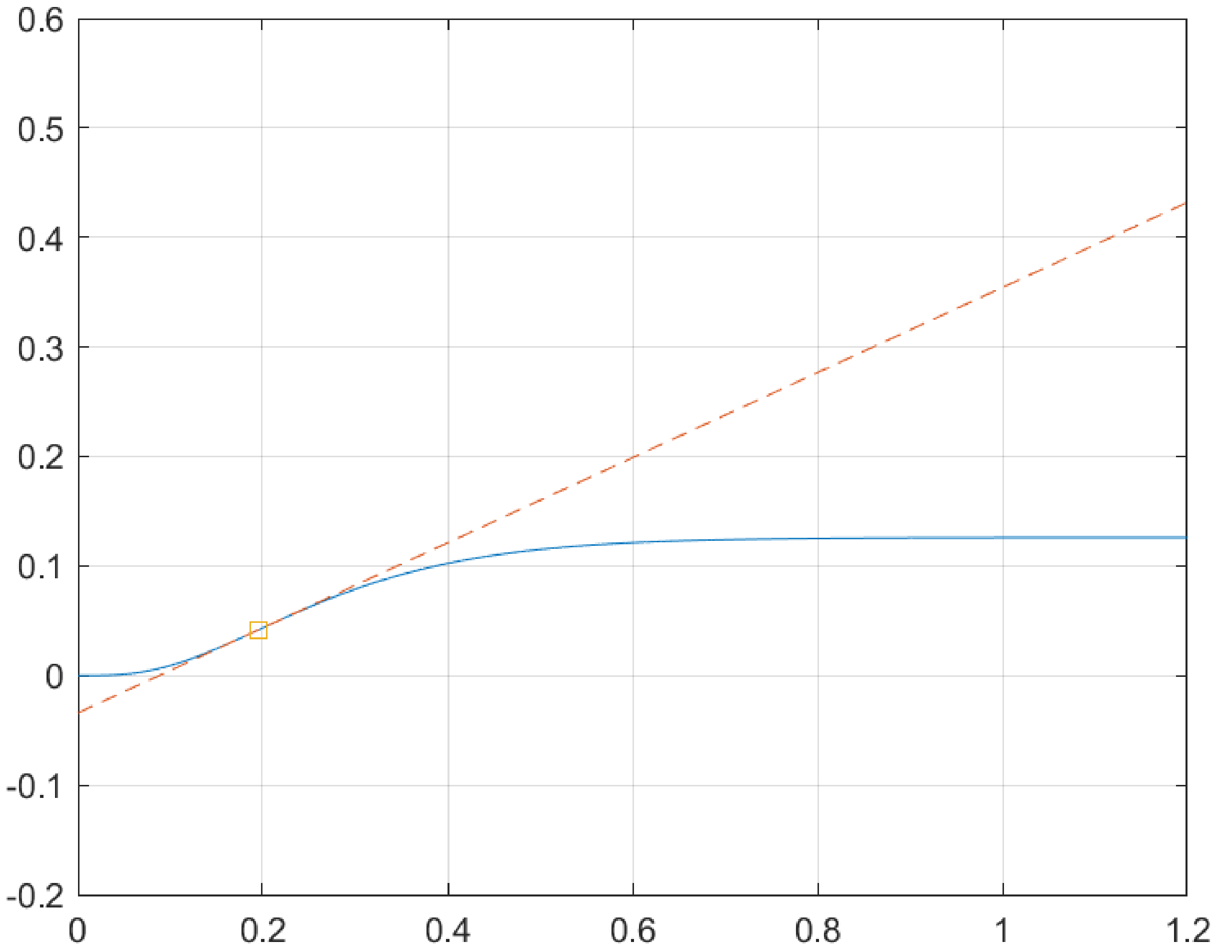


1) Modelkan persamaan sistem dengan metode Ziegler-Nichols dengan menarik tangent line pada inflection point sehingga didapatkan waktu delay *L* dan konstanta waktu *T*

Cari gradien tangent lain, dimana merupakan garis singgung pada plot step response dengan nilai maksimal:

```
M = gradient(Y,t); % Gradien masing masing titik pada step response
[m i] = max(M); % Max gradien m pada data ke i
c = Y(i)-(m*t(i)); % Menghitung pergeseran data
```

```
plot(t,Y,t,(m*t)+c,'--',t(i),Y(i),'s'); % plot step respon,
garis singgung, serta tanda titik inflection point

grid on;

axis([0 1.2 -0.2 0.6]);
```



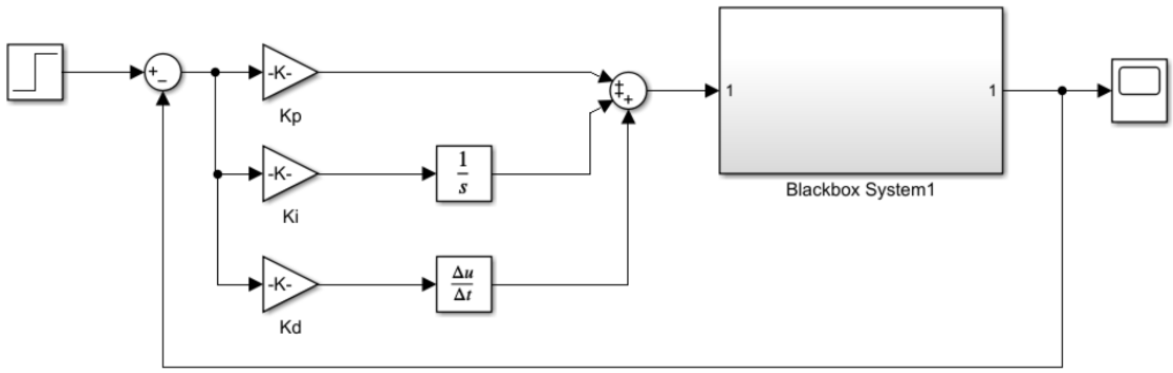

```
L = -c/m % Delay constant

T = ((Y(end)-c)/m)-L % Time constant
```

2) Setelah didapatkan *L* dan konstanta waktu *T*, lakukan perhitungan parameter PID berdasarkan tabel di bawah ini

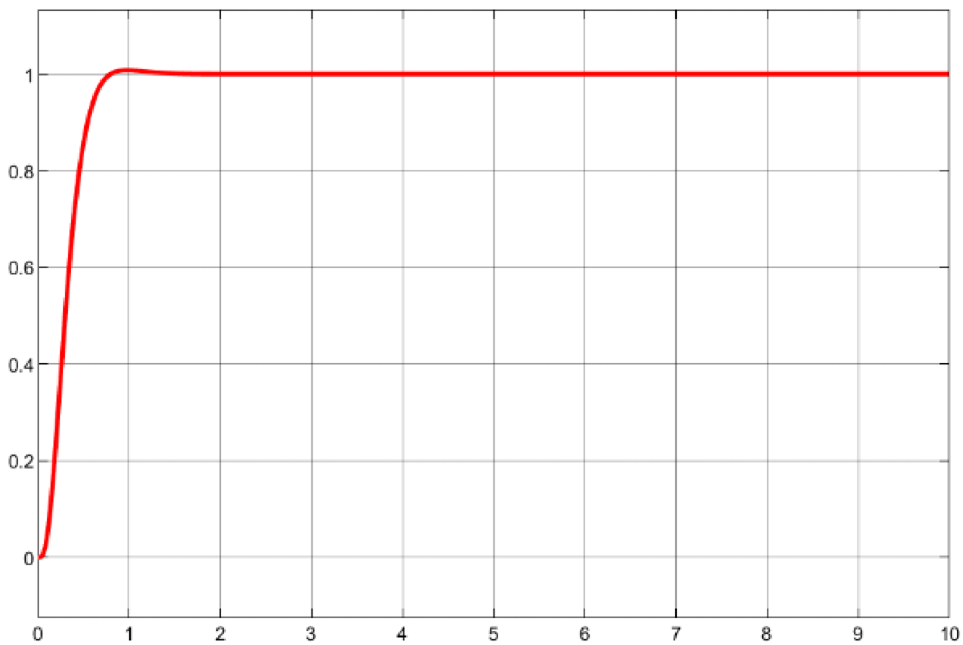
```
Kp = 1.2*(T/L)

Kp = 4.4202
```

```
Ti = 2*L;

Ki = Kp/Ti

Ki = 25.0978

Td = 0.5*L;

Kd = Kp*Td

Kd = 0.1946
```

Berikut diagram blok closed-loop system dengan plant blackbox dan diberi kontroler PID hasil tuning ZN



Step respon sistem:



3) Fine tuning
Pada plant ini, step respon sudah menghasilkan grafik yang bagus, sehingga fine tuning tidak perlu dilakukan.

3.2.2. Ziegler-Nichols tuning 2nd Method

Metode ini
Berikut langkah-langkah tuning metode Ziegler-Nichols metode kedua:

1. Set K_i dan K_d ke 0
2. Ubah nilai K_p dan cari nilai kritis (disebut K_{cr}) yang menghasilkan output berupa osilasi terus menerus atau mencapai kestabilan marginal yang berarti pole-pole berada tepat di sumbu imajiner
3. Tentukan periode osilasi output tersebut P_{cr}
4. Lakukan perhitungan parameter PID berdasarkan tabel di bawah

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

5. Fine tuning

Catatan: jika output tidak menghasilkan osilasi atau sistem tidak mencapai kestabilan marginal maka metode ini tidak dapat digunakan.

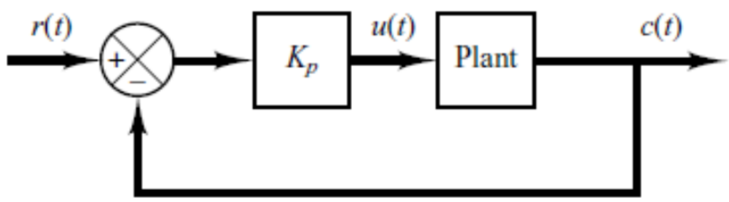
Contoh soal (1)

Transfer function plant:

$$G_1(s) = \frac{1}{s(s+1)(s+5)}$$

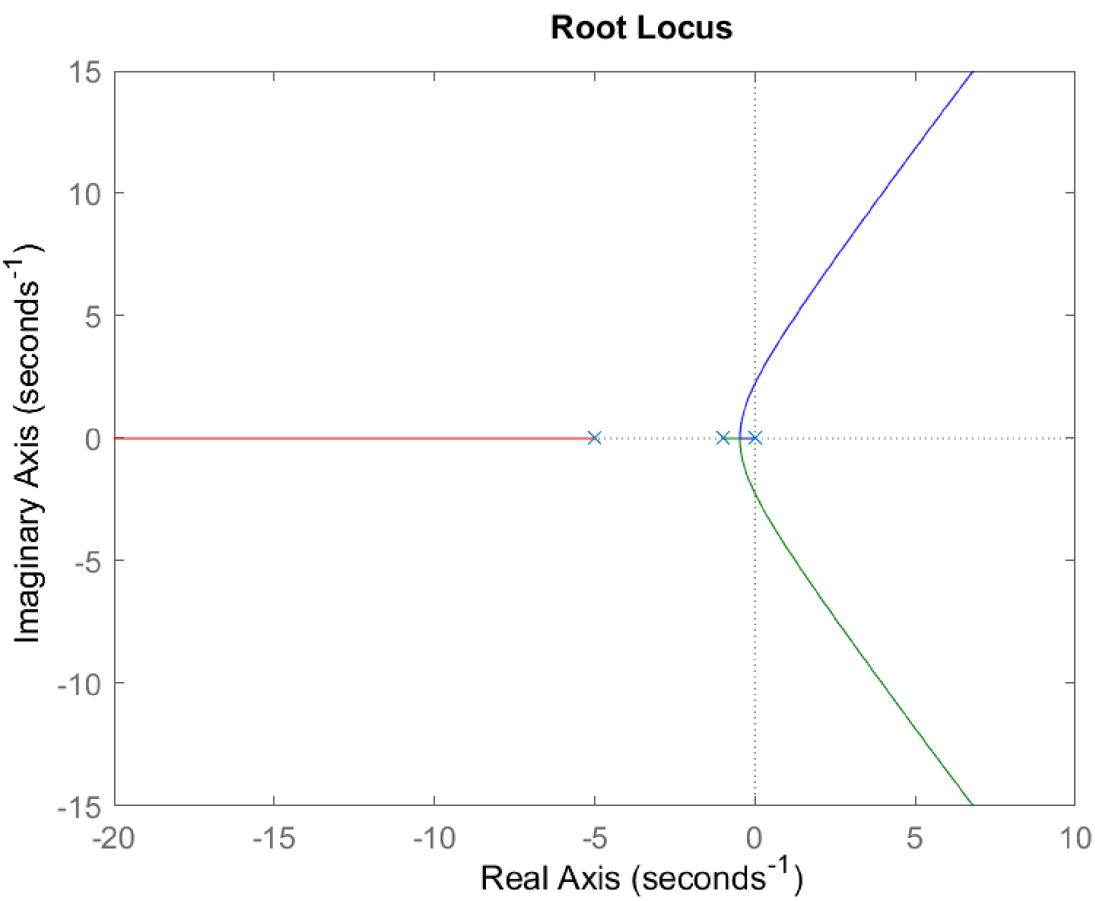
```
s = tf('s');
G1 = 1/(s*(s+1)*(s+5));
H1 = 1;
```

1) Set K_i and K_d ke 0, Sehingga closed loop system menjadi



2) Cari nilai kritis K_{cr}

```
rlocus(G1*H1)
```



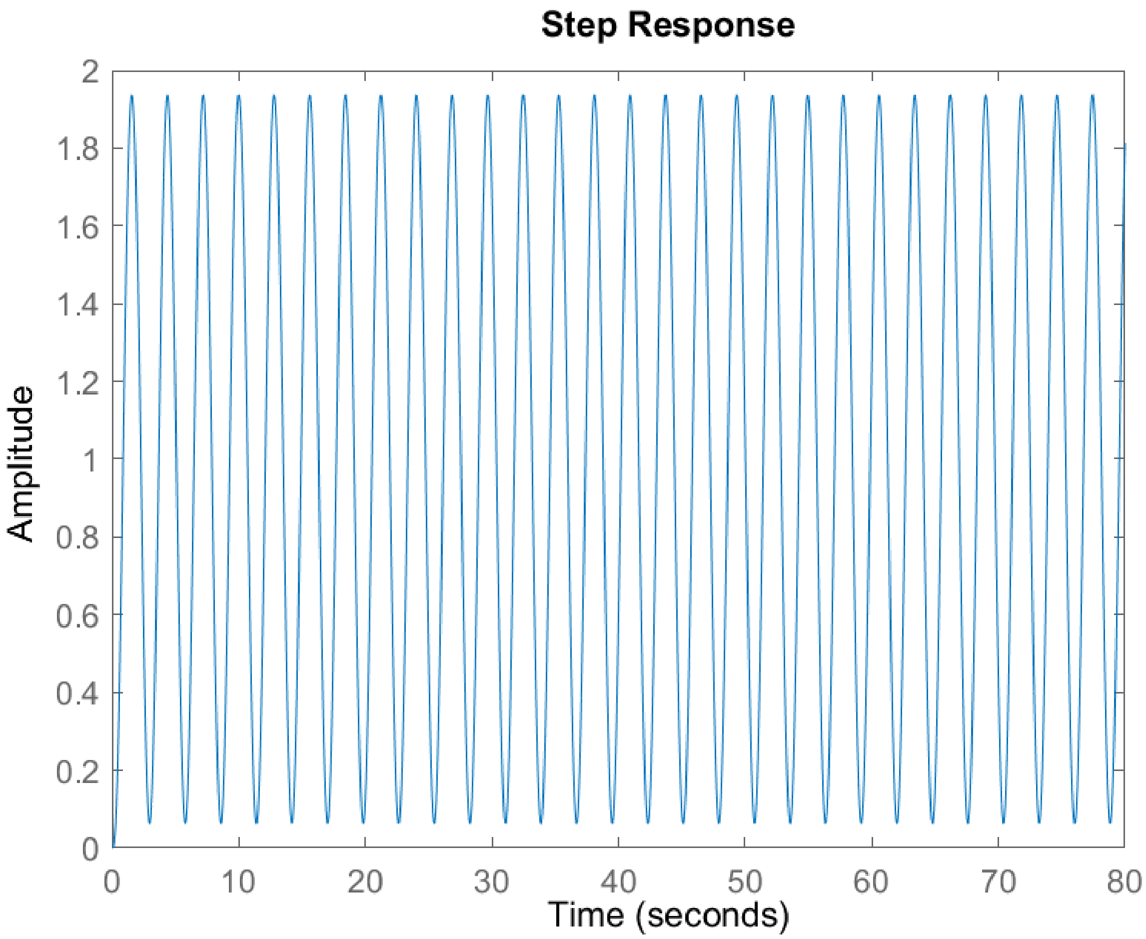
Didapatkan nilai gain critical value:

System: G1
Gain: 30
Pole: -1.8e-07 + 2.24i
Damping: 8.07e-08
Overshoot (%): 100
Frequency (rad/s): 2.24

Selain itu juga terdapat nilai frequency dalam satuan rad/s.

Melakukan pengecekan step response apakah sistem marginally stable dengan gain 30:

```
Kcr = 30;
feedback(Kcr*G1,1)
step(feedback(Kcr*G1,1))
```



Terlihat step response sistem berosilasi sehingga bisa dikatakan sistem stabil marginal

(Cara 2) Mencari K_{cr} dengan routh table

s^3	1	5
s^2	6	K_P
s^1	$\frac{30-K_P}{6}$	
s^0	$6K_P$	

3) Tentukan periode osilasi output P_{cr}

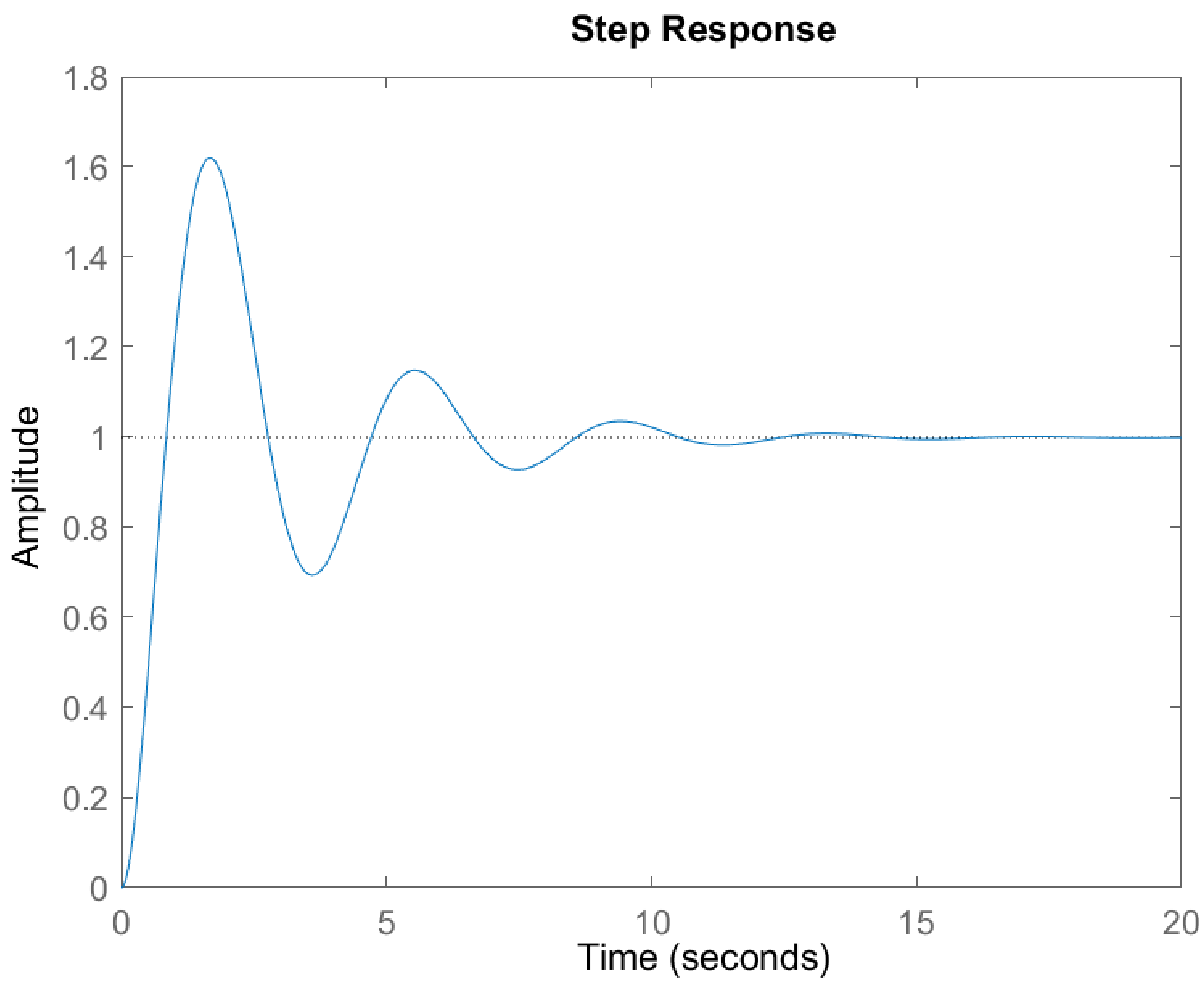
Dari langkah nomor 2 didapatkan frequency 2.24 rad/s, sehingga untuk mencari periode harus dikonversi dulu

```
w = 2.24;
f = w/(2*pi)
Pcr = 1/f
```

4) Lakukan perhitungan berdasarkan tabel

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

```
Kp = 0.6*Kcr
Ti = 0.5*Pcr;
Ki = Kp/Ti
Td = 0.125*Pcr;
Kd = Kp*Td
C = pid(Kp,Ki,Kd);
step(feedback(C*G1,1),20)
```

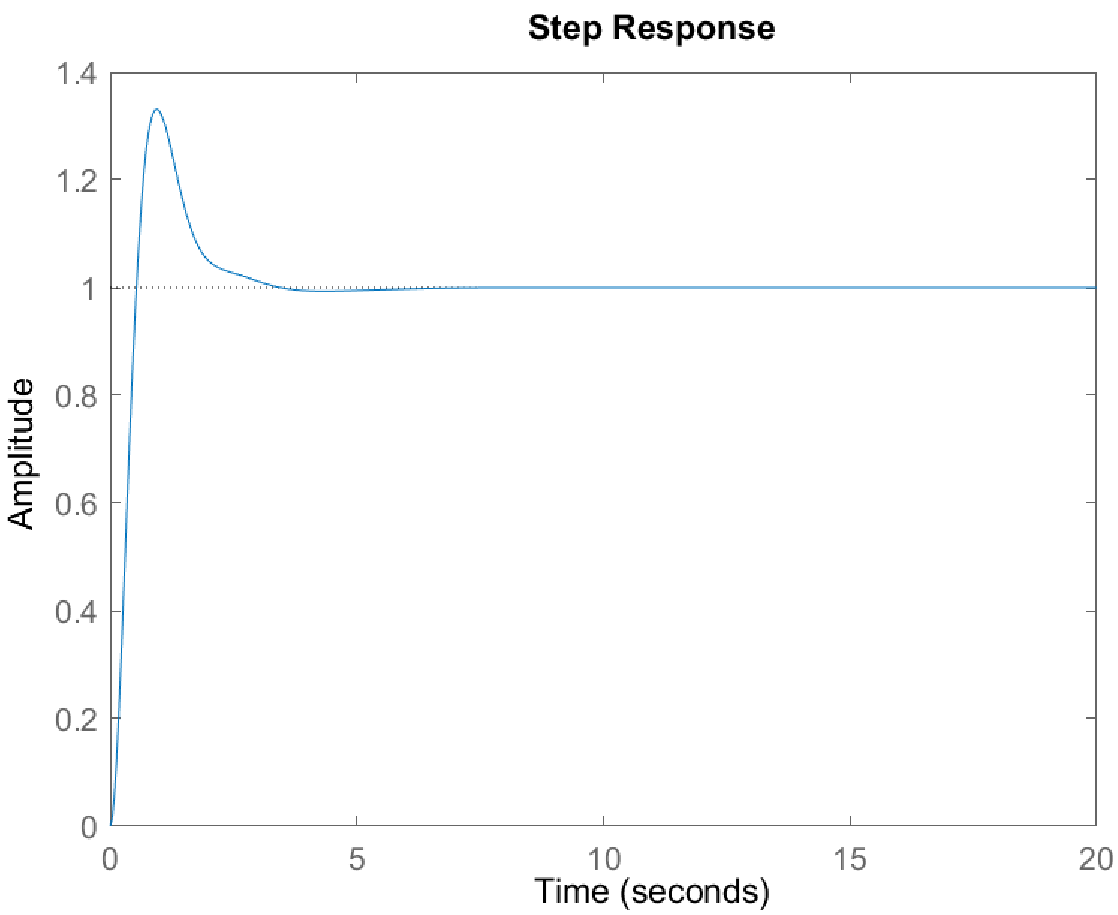


5) Fine tuning
Step response masih kurang baik dalam hal overshoot. ZN belum menjamin step response optimal, ZN hanya memberi parameter sebagai acuan awal. Lakukan fine tuning untuk mendapatkan yang paling bagus.

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Small Change	Decrease	Decrease	No Change

Selain itu, bisa dilakukan cara berikut:
1) Tambahkan proportional control untuk memperbaiki rise time
2) Tambahkan derivative control untuk mengurangi overshoot
3) Tambahkan integral control untuk mengurangi steady state error

```
Kp = 30;
Ki = 18;
Kd = 18;
C = pid(Kp,Ki,Kd);
step(feedback(C*G1,1),20)
```

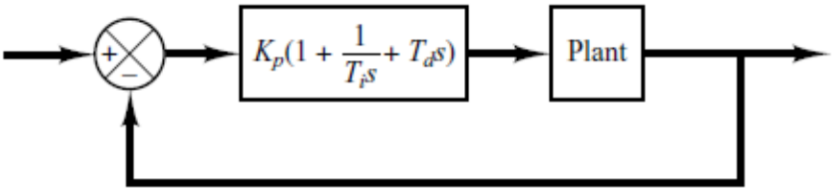


Contoh soal (2)

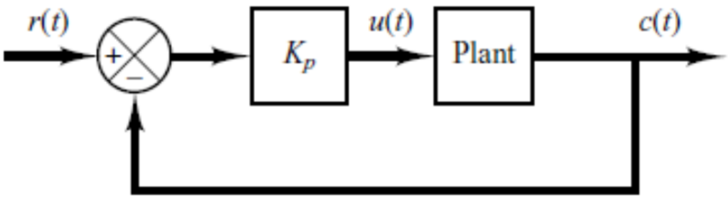
Transfer function plant:

$$G_1(s) = \frac{(s+2)(s+)}{s(s+1)(s+5)}$$

```
s = tf('s');
G2 = ((s+2)*(s+3))/(s*(s+1)*(s+5));
H2 = 1;
```



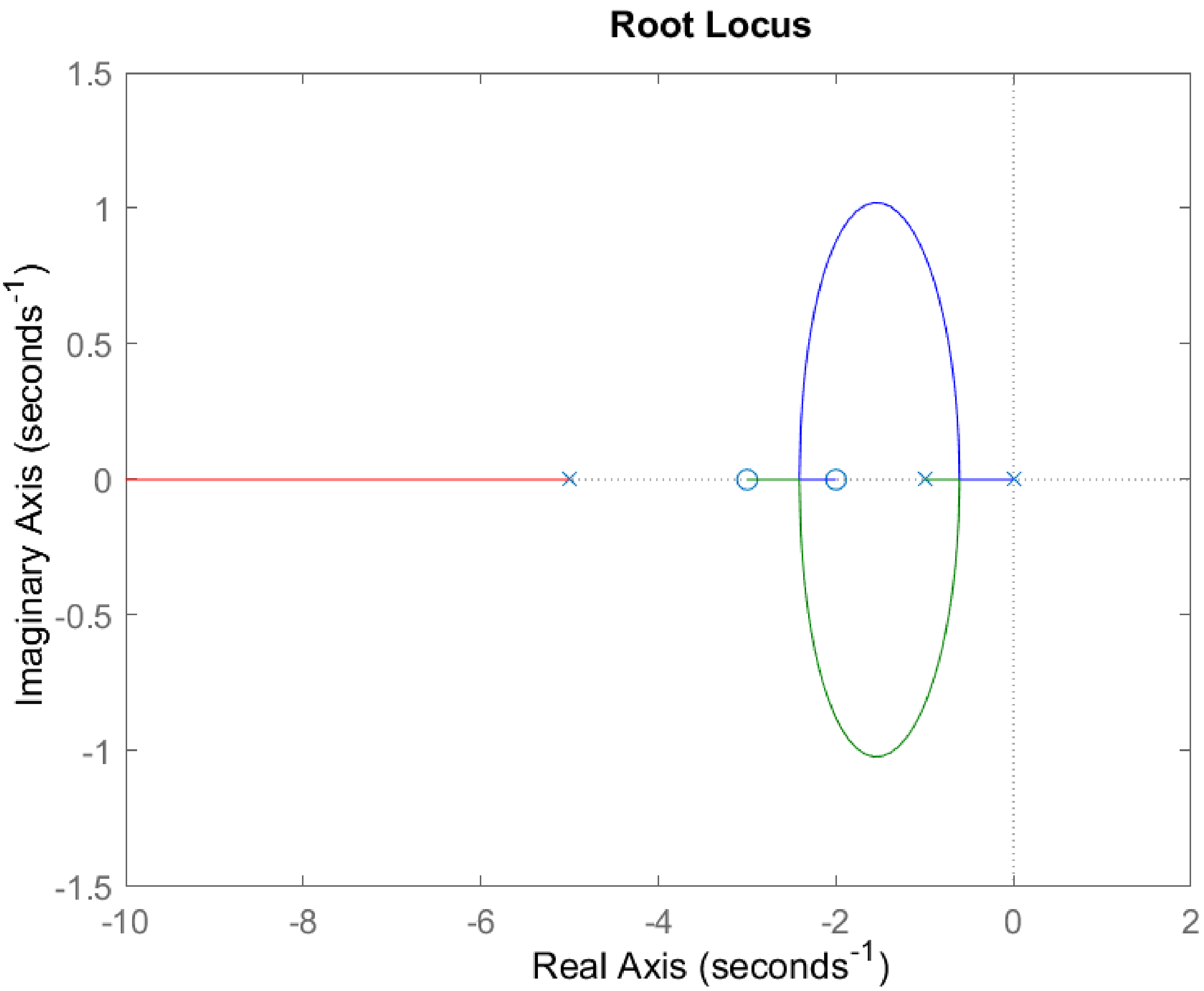
1) Set K_i dan K_d menjadi 0, Sehingga closed loop system menjadi



2) Cari nilai kritis K_{cr}

(Cara 1) Mencari K_{cr} dengan root locus plot

```
rlocus(G2*H2)
```



Tidak ditemukan kestabilan marginal untuk sistem ini karena pole-pole selalu berada di kiri sumbu imajiner sehingga metode Ziegler-Nichols tidak dapat digunakan

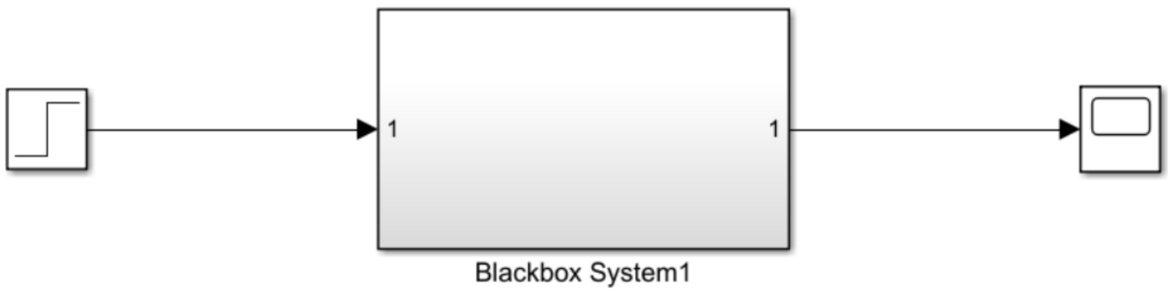
(Cara 2) Mencari dengan susunan kestabilan Routh:

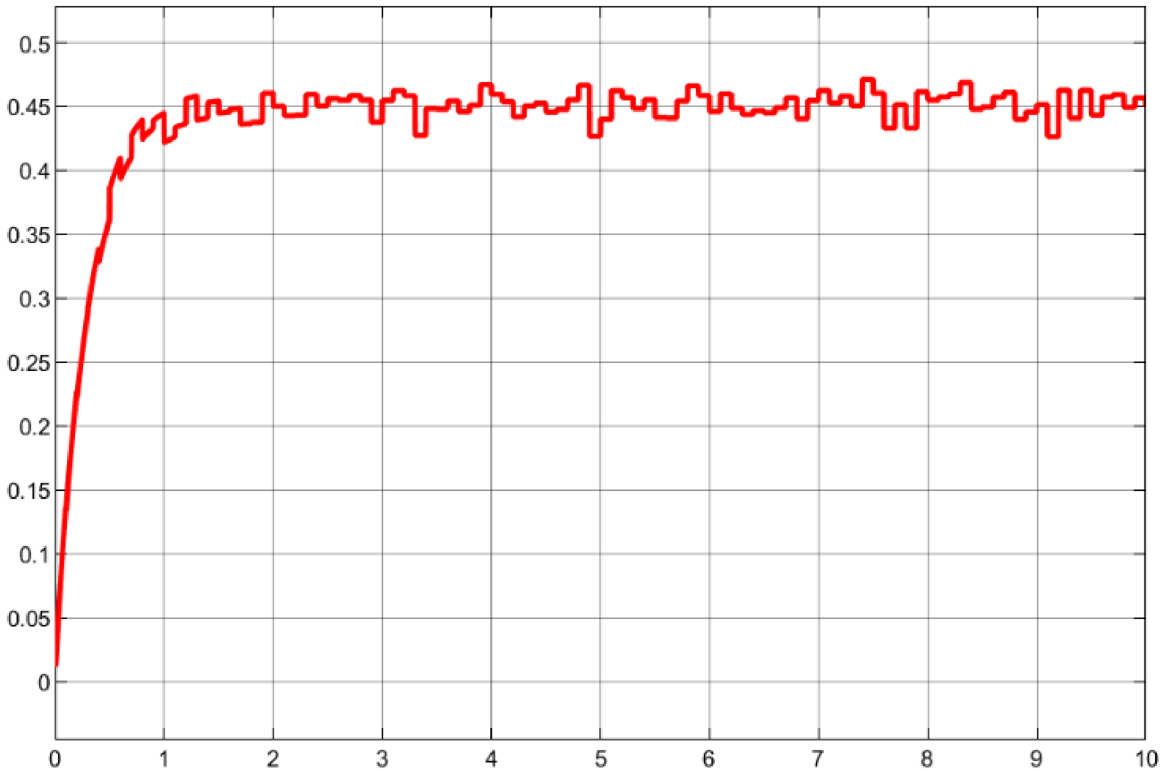
s^3	1	$5 + 5K_P$
s^2	$6 + 6K_P$	$6K_P$
s^1	$\frac{30+29K_p+5K_p^2}{6}$	0
s^0	$6K_P$	

Sistem stabil untuk semua nilai positif dan tidak ditemukan kestabilan marginal sehingga metode Ziegler-Nichols tidak dapat digunakan.

3.3. Desain Kontroller PID

Pada subbab ini akan dirancang kontroler PID berdasarkan model sistem yang didapat pada subbab PRBS modeling. Berikut respon open loop dari sistem blackbox yang digunakan:





Kontroler PID untuk sistem orde kedua tanpa delay

Prosedur desain kontroler PID dengan cara analitik

Transfer function kontroler PID, perbandingan sinyal kontrol dan sinyal error dalam domain s

$$\frac{U(s)}{E(s)} = K_p(1 + \frac{1}{T_i s} + T_d s)$$

1) Menentukan fungsi alih plant orde dua

$$TF = \frac{K\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$$

2) Menentukan spesifikasi performansi respon orde pertama yang diinginkan (Dikarenakan sistem hasil desain merupakan sistem orde pertama tanpa overshoot dan zero offset)

3) Menentukan parameter PID , , dengan rumus-rumus sebagai berikut

$$K_p = \frac{2\zeta}{\tau^* \omega_n K}$$

$$T_i = \frac{2\zeta}{\omega_n}$$

$$T_d = \frac{1}{2\zeta\omega_n}$$

dimana,
ζ = damping ratio plant

ω_n = natural frequency plant

K = gain plant

τ^* = time constant sistem hasil desain

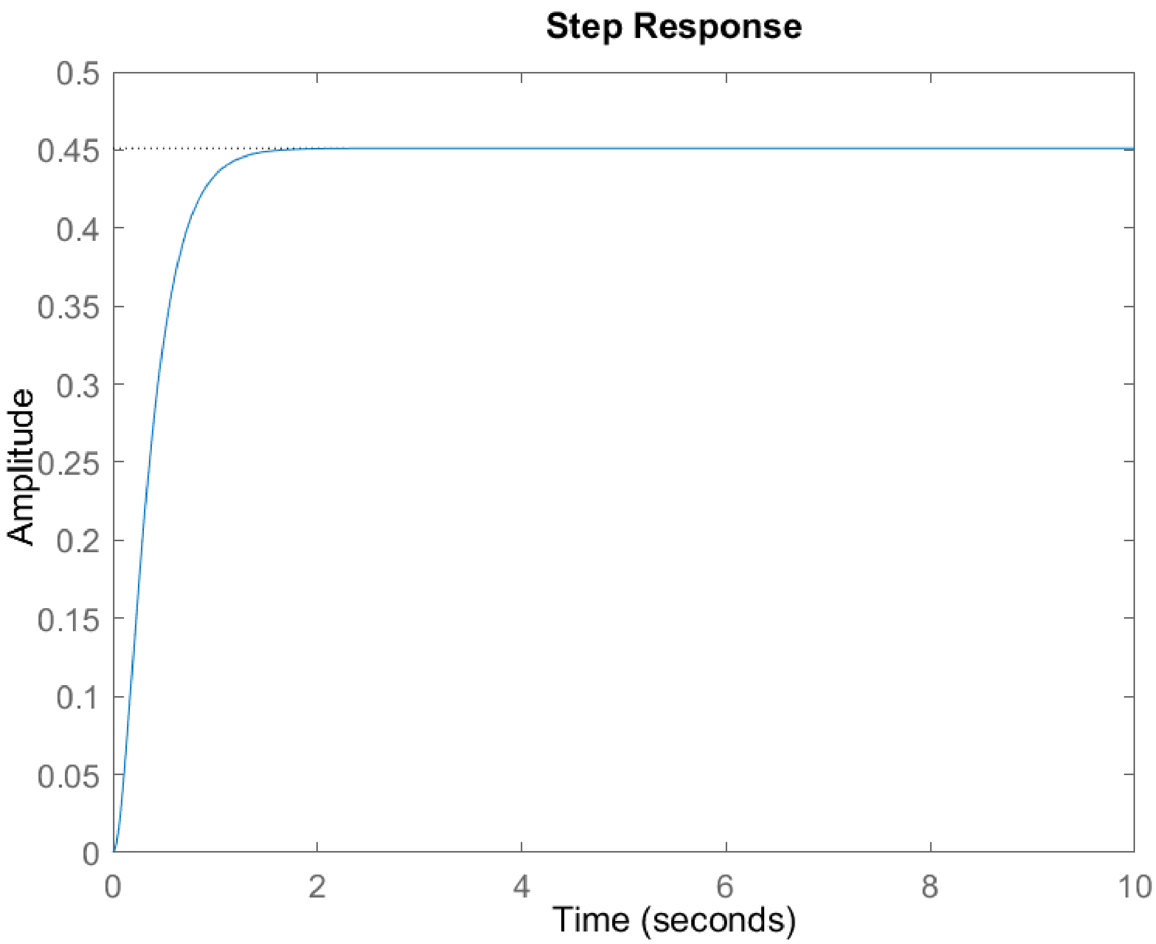
Contoh Soal

Transfer Function yang digunakan adalah sebagai berikut

$$P_1(s) = \frac{12.7}{s^2+10.9s+28.15}$$

Step response open loop dari sistem tersebut dapat dicari dengan

```
s = tf('s');
P1 = 12.7/(s^2 + 10.9*s + 28.15);
step(P1,10)
```



Berdasarkan step response di atas, dapat dicari karakteristik sistem:

```
stepinfo(P1)
```

```
ans = struct with fields:
        RiseTime: 0.6585
   TransientTime: 1.1572
    SettlingTime: 1.1572
     SettlingMin: 0.4074
     SettlingMax: 0.4511
       Overshoot: 0
      Undershoot: 0
            Peak: 0.4511
        PeakTime: 2.6181
```

Akan didesain kontroler PID sedemikian hingga keluaran sistem hasil mempunyai ts (±2%) sekitar 0.5 detik, Ess = 0 (Zero offset) dan tidak memiliki overshoot.

Menentukan parameter plant K, ω_n, ζ

Untuk mencari parameter plant, dapat menggunakan fungsi matlab:

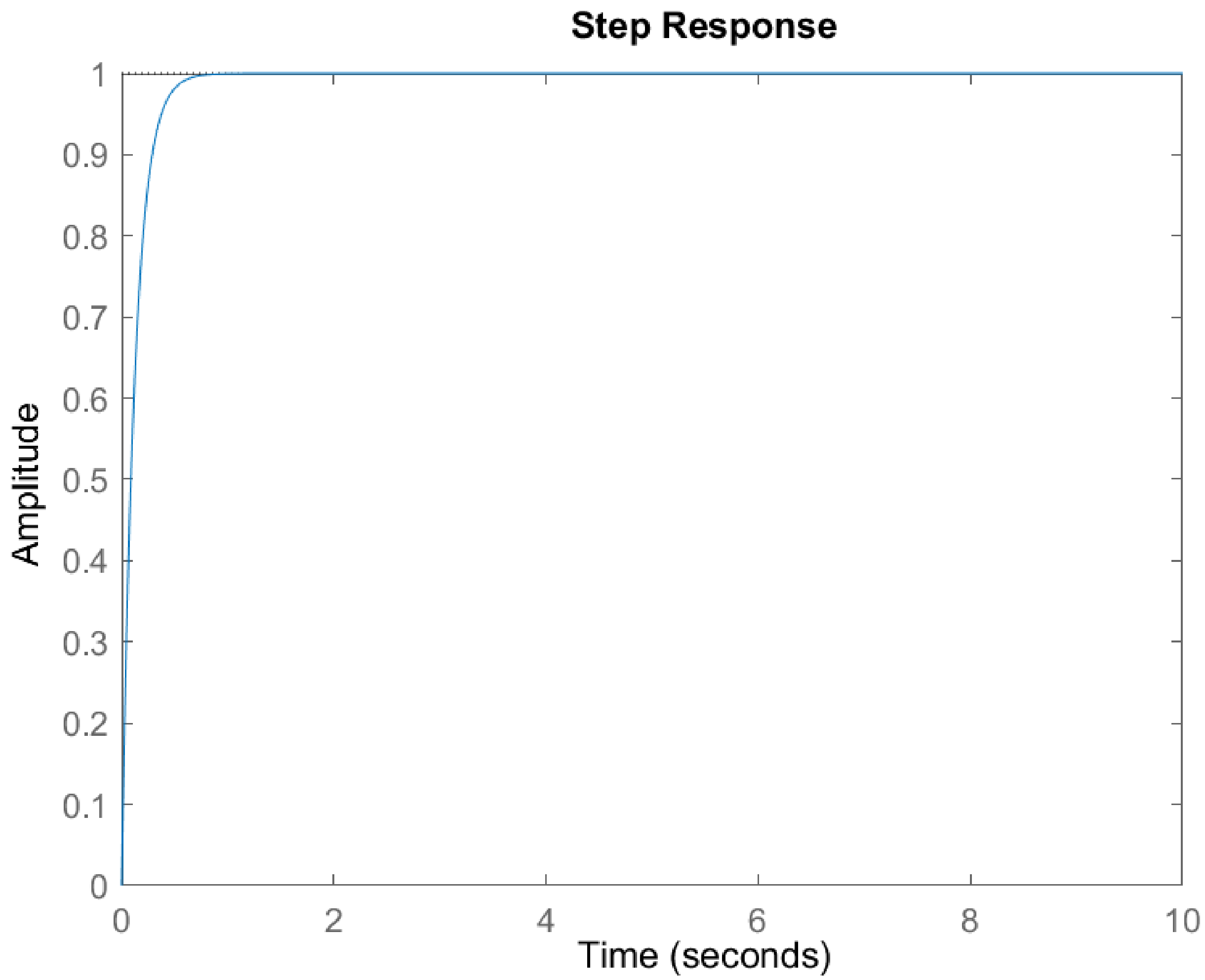
```
wn = sqrt(28.15)
zeta = 10.9/(2*wn)
K = 12.7/wn^2
Val = (K*wn^2)/(s^2+2*zeta*wn*s+wn^2)
```

Disubstitusikan parameter kontroler PID kedalam persamaan

```
ts = 0.5;
tau_star = ts/4; % ts(2%) = 4*tau
Kp = 2*zeta/(tau_star*wn*K)
Ti = 2*zeta/wn;
Td = 1/(2*zeta*wn);
Ki = Kp/Ti
Kd = Kp*Td
C = pid(Kp,Ki,Kd);
```

Selanjutnya dicari step response untuk closed loop transfer function

```
CLTF1 = feedback(P1*C,1);
step(CLTF1,10)
```



Berdasarkan step response sistem hasil desain di atas, dapat dicari karakteristik sistem:

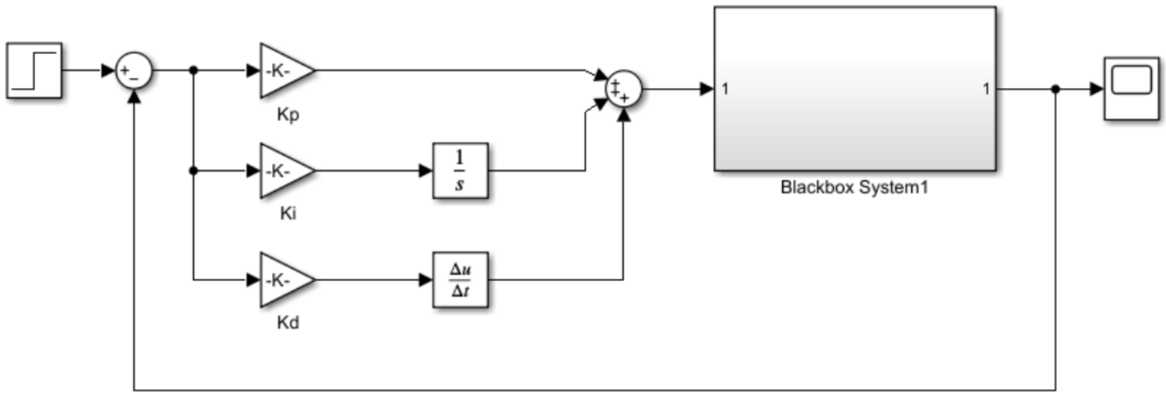
```
stepinfo(CLTF1)
```

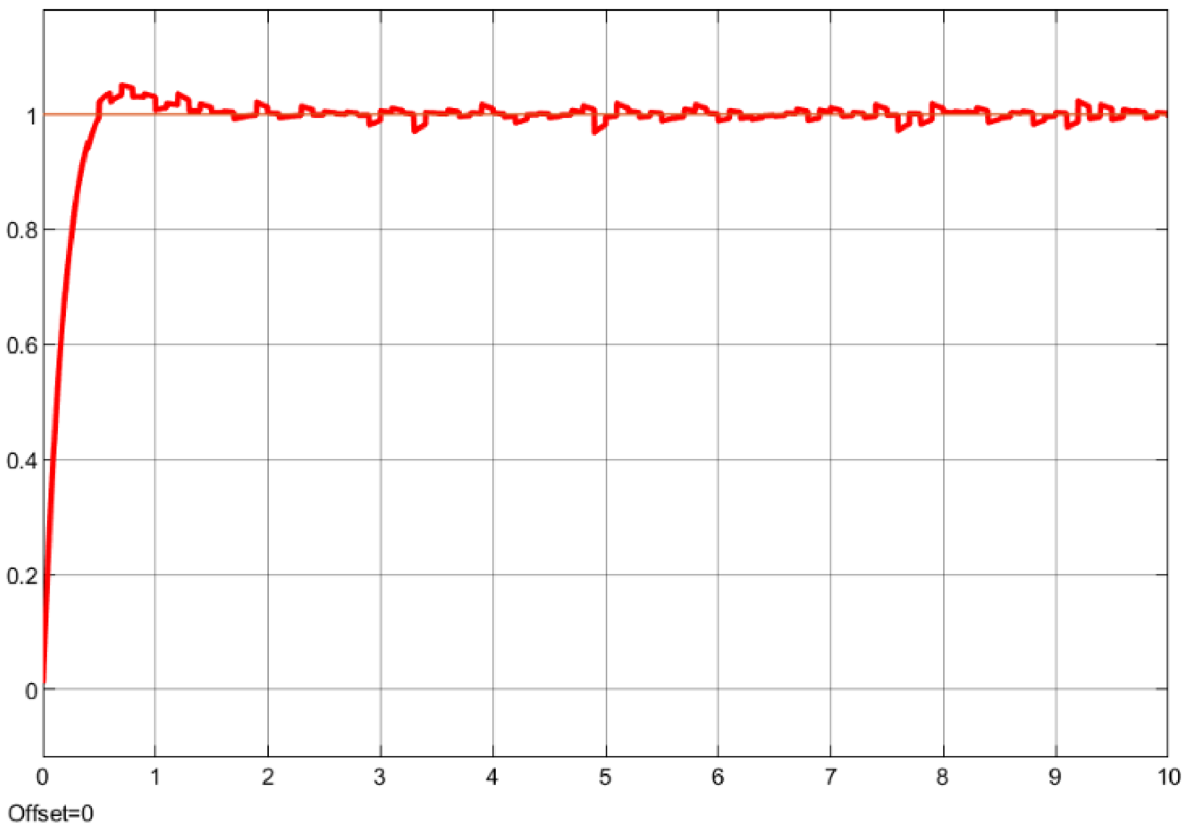
```
ans = struct with fields:
        RiseTime: 0.2746
   TransientTime: 0.4890
    SettlingTime: 0.4890
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 1.3182
```

Berdasarkan plot step response dan step info, sistem sudah memenuhi kriteria desain.

Implementasi di simulink dengan model system black box yang digunakan pada bab 1:

Closed-loop response:


-K-
Kp
-K-
Ki
-K-
Kd
1
1
Blackbox System1


1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
Offset=0