



Daftar Isi

Daftar Isi	1
MODUL 2 <i>Modelling</i> dan Karakteristik Sistem.....	3
Bab 1 Identifikasi atau <i>Modelling</i> Sistem.....	3
1.1 Pengambilan Data	3
1.1.1 PRBS <i>Signal Input</i>	3
1.1.2 <i>Import Data</i>	4
1.2 <i>System Identification Toolbox</i>	5
1.3 Metode <i>Black Box</i> (ARX)	10
1.3.1 Pengambilan Data Sistem <i>Real</i>	11
1.3.2 Identifikasi Sistem dengan ARX.....	12
1.4 Metode Lainnya.....	14
1.4.1 Metode Viteckova	14
1.4.2 Metode Latzel	17
1.4.3 Metode Ziegler-Nichols.....	19
1.4.4 Metode Strejc	19
1.4.5 Metode Broida	20
1.4.6 Perbandingan Metode	21
BAB 2 Karakteristik Sistem	22
2.1 Sinyal Uji.....	22
2.1.1 Sinyal Impuls	22
2.1.2 Sinyal Step	23
2.1.3 Sinyal Ramp Tunggal atau Periodik	23
2.1.4 Sinyal Persegi	24
2.1.5 Sinyal Sinusoidal	24
2.2 Analisis <i>Root Locus</i>	25
2.3 Analisis <i>State Space</i>	30
2.3.1 Stability.....	30
2.3.2 <i>Controllability</i>	31
2.3.3 <i>Observability</i>	31
2.4 Analisis Domain Waktu.....	32
2.4.1 Karakteristik Respon Orde 1.....	32
2.4.2 Karakteristik Respon Orde 2.....	35



2.5 Analisis Domain Frekuensi	39
2.5.1 Bode.....	39
2.5.2 Nyquist.....	44

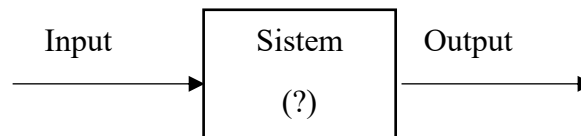


MODUL 2 *Modelling* dan Karakteristik Sistem

Pada modul ini akan dibahas mengenai identifikasi atau *modelling* serta karakteristik sistem. Kedua hal tersebut sangat bermanfaat bagi pemahaman *engineer* terutama *control system engineer*. Secara sekilas, pemodelan dapat digunakan untuk mendapatkan model matematis dari suatu sistem yang merepresentasikan dinamika sistem yang didekati dan karakteristik dapat digunakan untuk menganalisis suatu sistem sebelum dilakukan desain suatu *control system* bagi sistem tersebut.

Bab 1 Identifikasi atau *Modelling* Sistem

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 nilai input dan output sistem tersebut. Model matematis ini diharapkan dapat menggambarkan perilaku atau merepresentasikan dinamika sistem yang didekati, khususnya jika diberi komponen baru seperti kontroler. Beberapa model matematika yang umum digunakan antara lain: model persamaan diferensial, model fungsi alih atau *transfer function*, 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*.



Ilustrasi dari identifikasi sistem yaitu seperti diagram di atas, dimana sistem yang tidak diketahui modelnya akan menghasilkan suatu output berdasarkan input yang diberikan. Kita akan mencari model dari sistem tersebut berdasarkan input-output yang dihasilkan.

1.1 Pengambilan Data

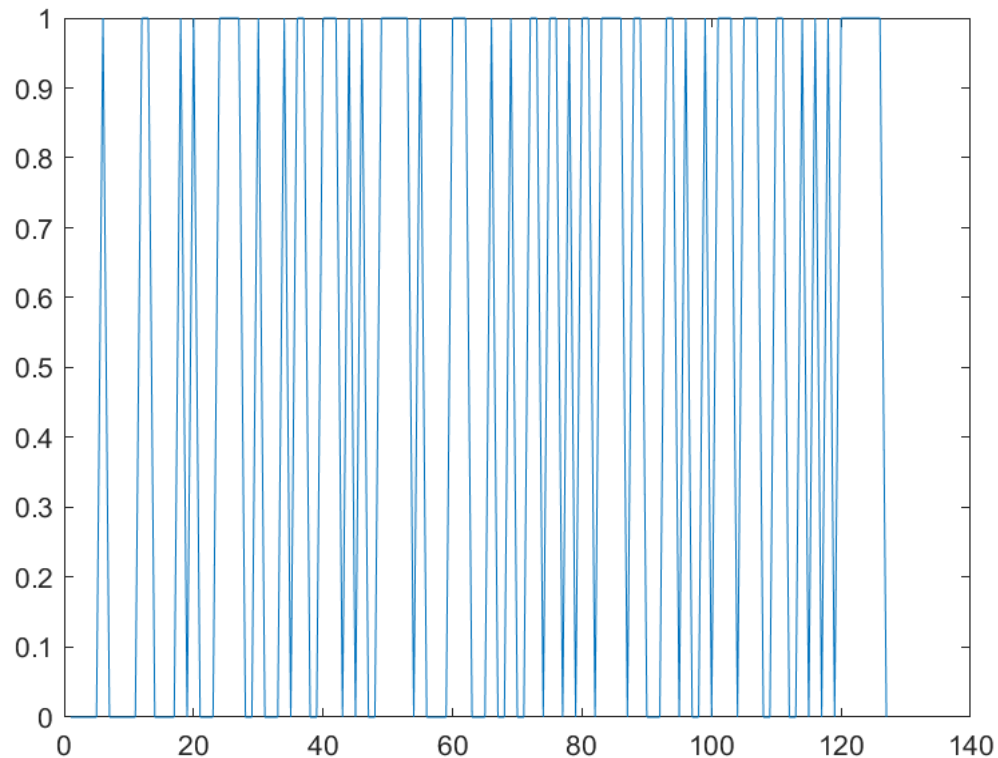
Pemodelan memerlukan data berupa input-output sistem yang ingin diidentifikasi. Ada beberapa metode untuk mendeklarasikan input-output sistem di MATLAB.

1.1.1 PRBS *Signal Input*

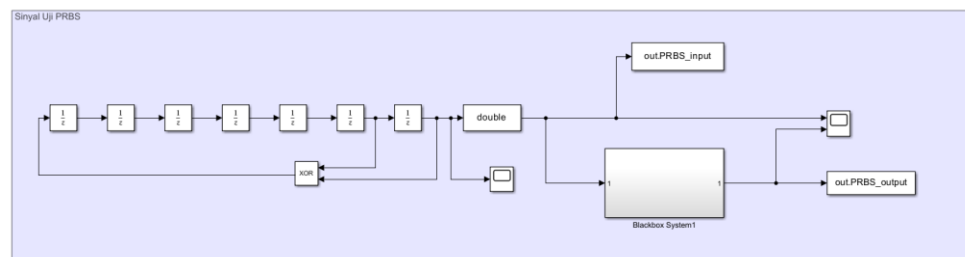
Pseudorandom Binary Sequence (PRBS) adalah sinyal deterministik periodik yang nilainya bergantian pada dua nilai. Sinyal PRBS periodik memiliki periode maksimum yaitu $2^n - 1$, dengan n adalah orde PRBS. Pada MATLAB, kita dapat membangkitkan sinyal PRBS sebagai berikut

```
O = 7 %orde PRBS
N = 2^O-1 %periode maksimum
input = prbs(O,N)
plot(input) %plot sinyal PRBS
```

Apabila *script* tersebut dijalankan maka menghasilkan plot sebagai berikut



Secara *default*, nilai yang dihasilkan yaitu 0 dan 1. Sinyal PRBS ini digunakan sebagai masukan atau input dari sistem sebagai sinyal uji, melalui input PRBS kita akan mengetahui bagaimana respon sistem dengan masukan berbeda dan berubah secara tiba-tiba antara 0 dan 1. 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* atau sistem yang akan diidentifikasi dengan input PRBS secara simulasi.



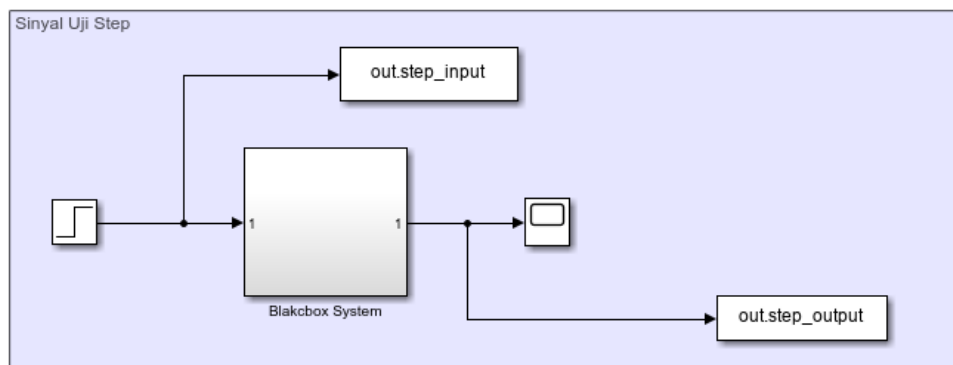
Pada diagram di atas, terdapat *Blackbox System 1* yang merupakan sistem yang akan diidentifikasi. Blok *out.PRBS_input* dan *out.PRBS_output* merupakan jenis blok “To Workspace” pada SIMULINK yang digunakan untuk menyimpan kedua nilai tersebut pada workspace MATLAB. Sehingga nantinya kita dapat memodelkan sistem melalui MATLAB berdasarkan nilai yang diinputkan pada workspace.

1.1.2 Import Data

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

```
%import data file .txt  
load('blackboxsystem_training_step.txt')  
datasistem = blackboxsystem_training_step  
time = datasistem(:,1) %kolom pertama sebagai waktu  
input = datasistem(:,2) %kolom kedua sebagai nilai input  
output = datasistem(:,3) %kolom ketiga sebagai nilai output
```

Data tersebut sebenarnya didapat dari simulasi SIMULINK juga yaitu dengan menggunakan sinyal uji *step* seperti pada diagram di bawah.

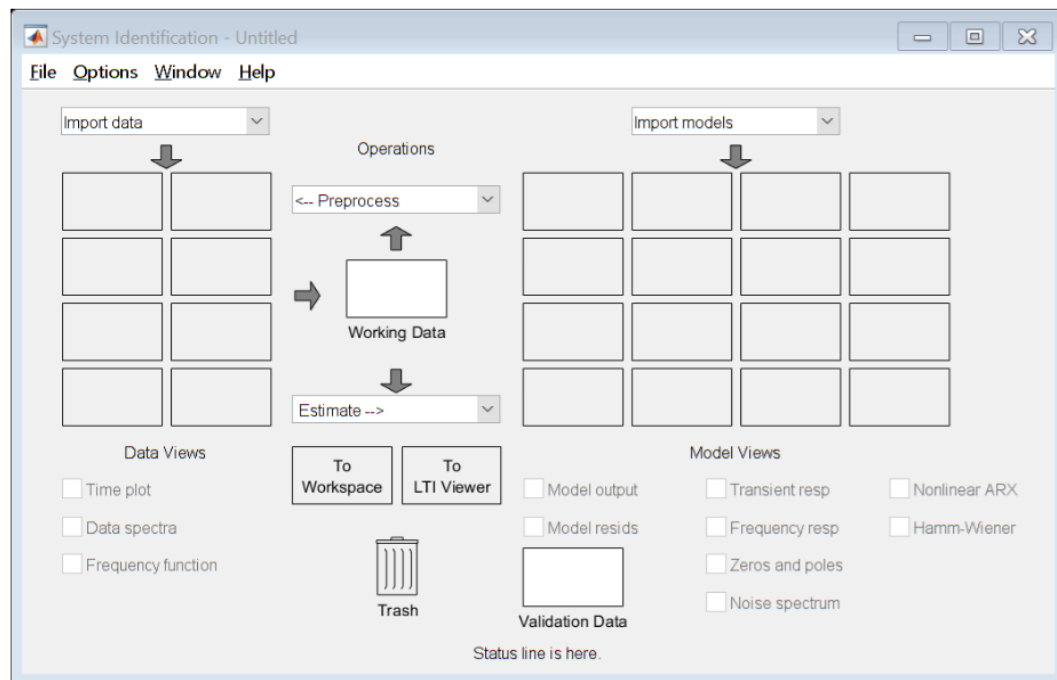


Tetapi data dari input-output telah disimpan pada suatu *file* berbentuk .txt. Kita juga dapat menggunakan bentuk *file* lain seperti .xls (excel), .csv, .mat (data MATLAB).

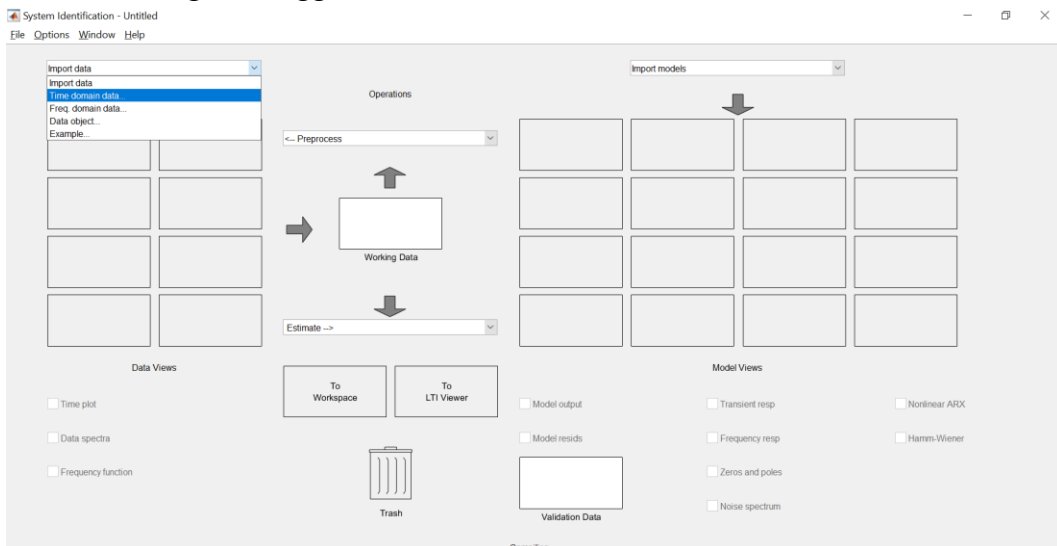
1.2 System Identification Toolbox

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**

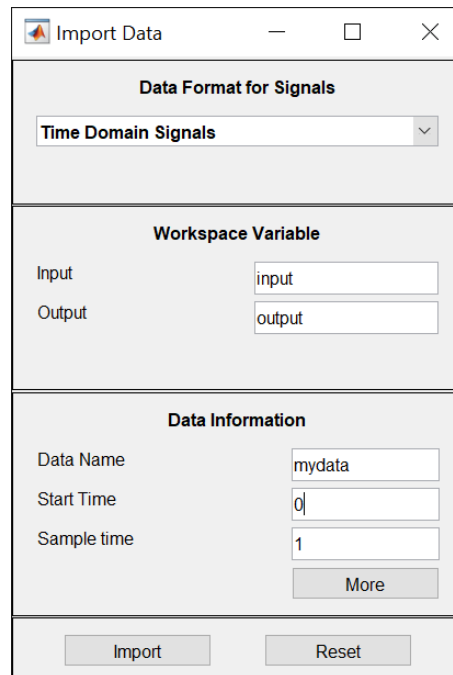
```
%membuka toolbox  
systemIdentification  
  
%deklarasikan hasil transfer function dari toolbox dalam variabel  
sys  
s = tf('s')
```



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 pada workspace MATLAB. 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 (variabel pada workspace MATLAB) input dan output yang telah didapatkan beserta *start time* (waktu awal dari data yang dapat dilihat pada variabel *time*).



Import Data

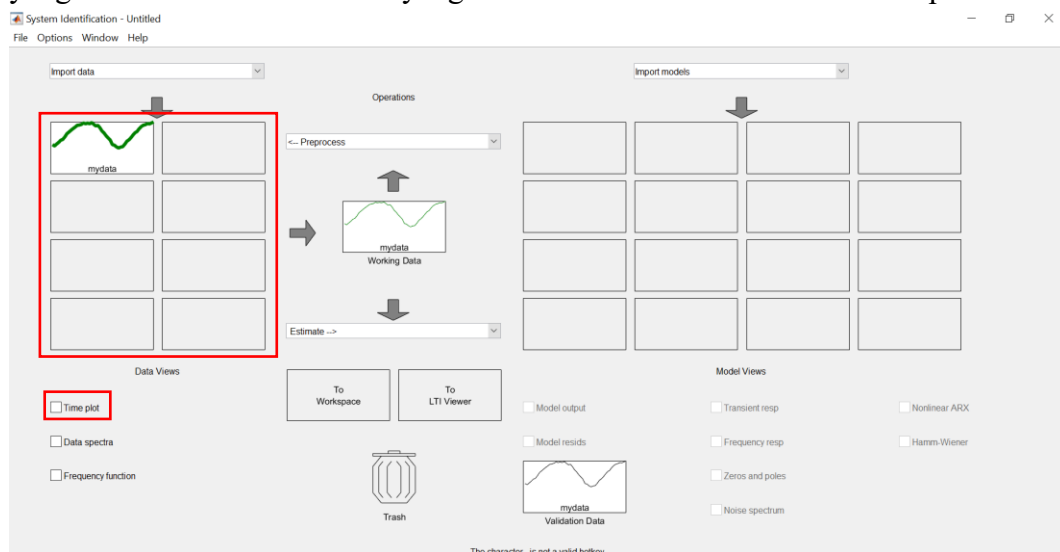
Data Format for Signals
Time Domain Signals

Workspace Variable
Input: input
Output: output

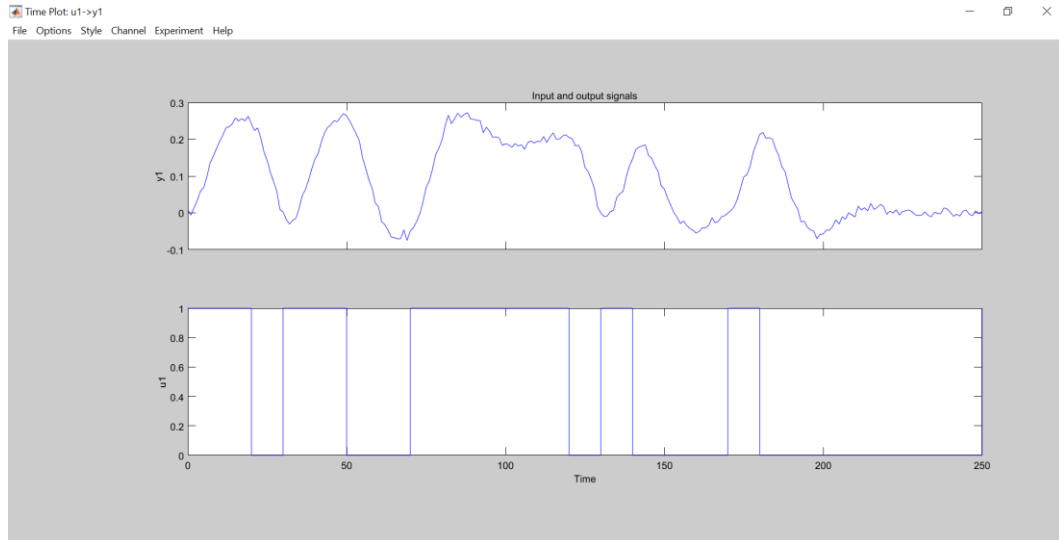
Data Information
Data Name: mydata
Start Time: 0
Sample time: 1
More

Import Reset

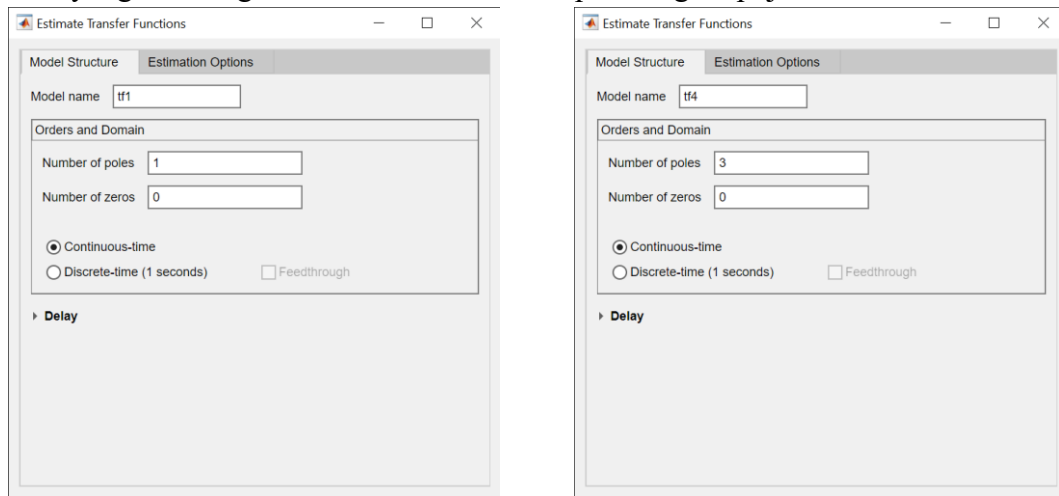
Setelah diimport maka akan terlihat gambar pada bagian kiri (gambar di bawah) yang menandakan bahwa data yang dimasukkan tadi telah berhasil diimport.



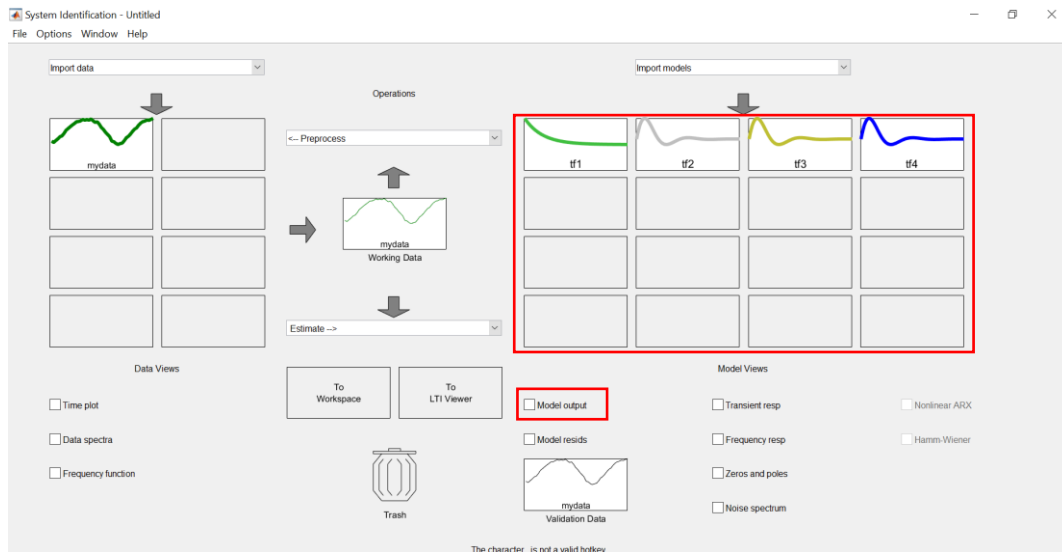
Untuk melihat input output yang telah diimport dapat dilakukan melalui perintah 'Time plot' yang terdapat pada bagian kiri bawah sehingga dapat menampilkan plot dari input dan output terhadap waktu sebagai berikut



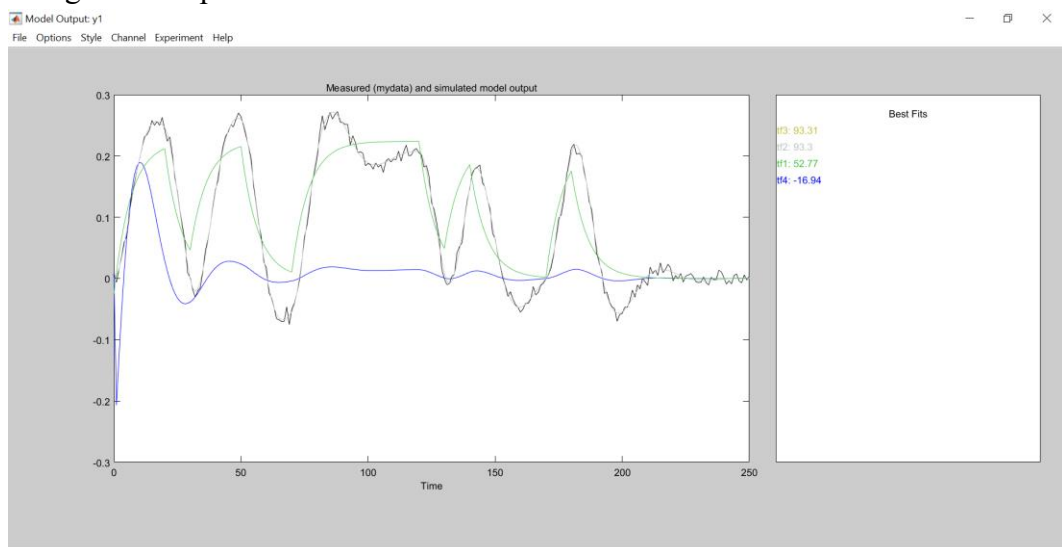
Untuk mengolah data atau mengestimasi sistem maka pilih pada bagian **Estimate** dan pilih **Transfer Function Models** dikarenakan pada pembahasan kali ini akan berfokus pada model transfer function. Pada window yang muncul akan kita coba untuk mengestimasi model dari sistem dengan mengubah-ubah jumlah 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. Untuk membandingkan nilai estimasi mana yang paling mendekati plant atau sistem sesungguhnya maka kita pilih 'Model output'.



Masing-masing estimasi akan dibandingkan dengan *real plant* (data input-output) yang kita inputkan tadi dan memberikan persentase seberapa cocok output model estimasi dengan sistem sesungguhnya. Dalam hal ini tf3 mendapatkan persentase tertinggi yaitu 93.31% sehingga nilai dari transfer function tf3 dapat kita gunakan sebagai model plant.



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



Data/model Info: tf2

Model name:

tf2

Color:

[0.25,0.75,0.25]

From input "u1" to output "y1":

12.7

$s^2 + 10.9 s + 28.15$

Name: tf2

Continuous-time identified transfer function.

Parameterization:

Number of poles: 2 Number of zeros: 0

Diary and Notes

```
% Details about Estimation Data
% Import    mydata

% Transfer function estimation
Options = tfestOptions;
Options.Display = 'on';
Options.EnforceStability = true;

tf2 = tfest(mydata, 2, 0, Options)
```

Show in LTI Viewer

Present

Export

Delete

Close

Help

Selanjutnya kita dapat mengeksport hasil estimasi tersebut ke workspace MATLAB dengan memencet tombol Export pada window tersebut. Hasil akan disimpan dalam bentuk objek *idtf* yaitu berupa TF atau *Transfer Function*.

1.3 Metode *Black Box* (ARX)

Kita juga dapat memodelkan dinamika sistem dari data input-output sistem dengan menggunakan metode *Black Box*. Berbeda dengan sebelumnya, data yang digunakan harus dalam bentuk **diskrit** dengan waktu sampling tertentu (T_s). Terdapat beberapa variasi metode *Black Box* antara lain

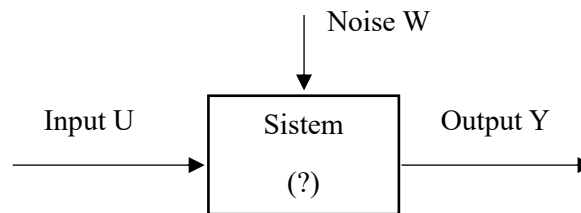
1. Model ARMA (*Auto Regressive and Moving Average*)
2. Model ARIMA (ARMA dengan Integrator)
3. Model sARIMA (Seasonal ARIMA)
4. Model ARMAX (ARMA dengan eXogenous input)

Perbedaan penggunaan antara model yaitu ARMA digunakan untuk *modelling* variasi output saja tanpa input (output dengan *noise* sistem), ARIMA untuk *modelling* variasi output yang memiliki *trend* (tidak stasioner), sARIMA untuk *modelling* data periodik, ARMAX untuk *modelling* output dari sistem dengan input yang diberikan. Pada sub bab ini akan dibahas mengenai pemodelan sistem dengan

menggunakan ARMAX karena diinginkan fungsi alih atau transfer function antara output terhadap input sistem. Bentuk umum model ARMAX(p,s,q) yaitu

$$Y(k) = \sum_{i=1}^p a_i Y(k-i) + \sum_{j=0}^s b_j U(k-j) + \sum_{n=0}^q c_n W(k-n)$$

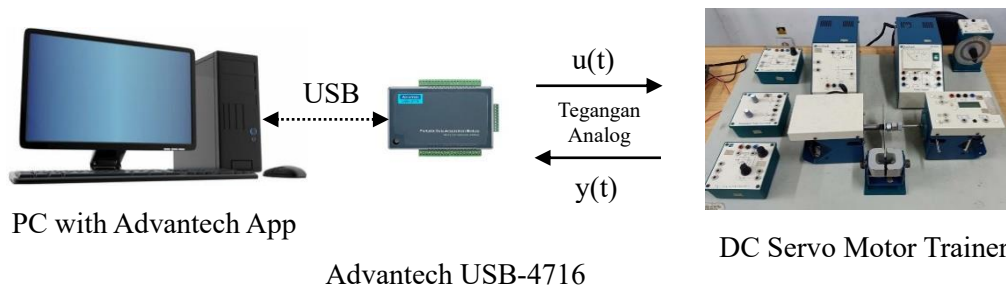
dimana Y adalah output, U adalah input, dan W adalah *noise* sistem dengan asumsi bahwa W adalah iid *zero mean Gaussian white noise*.



Dikarenakan pada fungsi alih tidak terdapat *noise* sistem maka kita hanya menggunakan input dan output saja. Model ini sering disebut ARX(p,s) dengan p adalah derajat AR (output) dan s adalah derajat komponen exogenous (input). Sehingga diagram sistem akan menjadi seperti diagram pada pendahuluan modul ini.

1.3.1 Pengambilan Data Sistem *Real*

Sistem yang akan diidentifikasi pada bagian ini yaitu sistem DC Servo Motor Trainer. Ilustrasi dari sistem tersebut yaitu pada gambar di bawah.



Data dari motor DC tersebut diambil dengan menggunakan perangkat yang bernama Advantech USB-4716. Data berupa respon sistem *open loop* terhadap sinyal uji *step* hingga *steady state*. Perangkat tersebut tersambung dengan sistem motor DC serta sebuah PC. Input ditentukan berupa unit step sebesar 7V dengan waktu sampling (T_s) sebesar 0.001 detik. Data tersebut dapat diimport pada workspace MATLAB melalui perintah sebagai berikut

```
%import data file .mat
load('data_servo_dcmotor.mat')

t = data_dc(:,1) %kolom pertama sebagai waktu
```

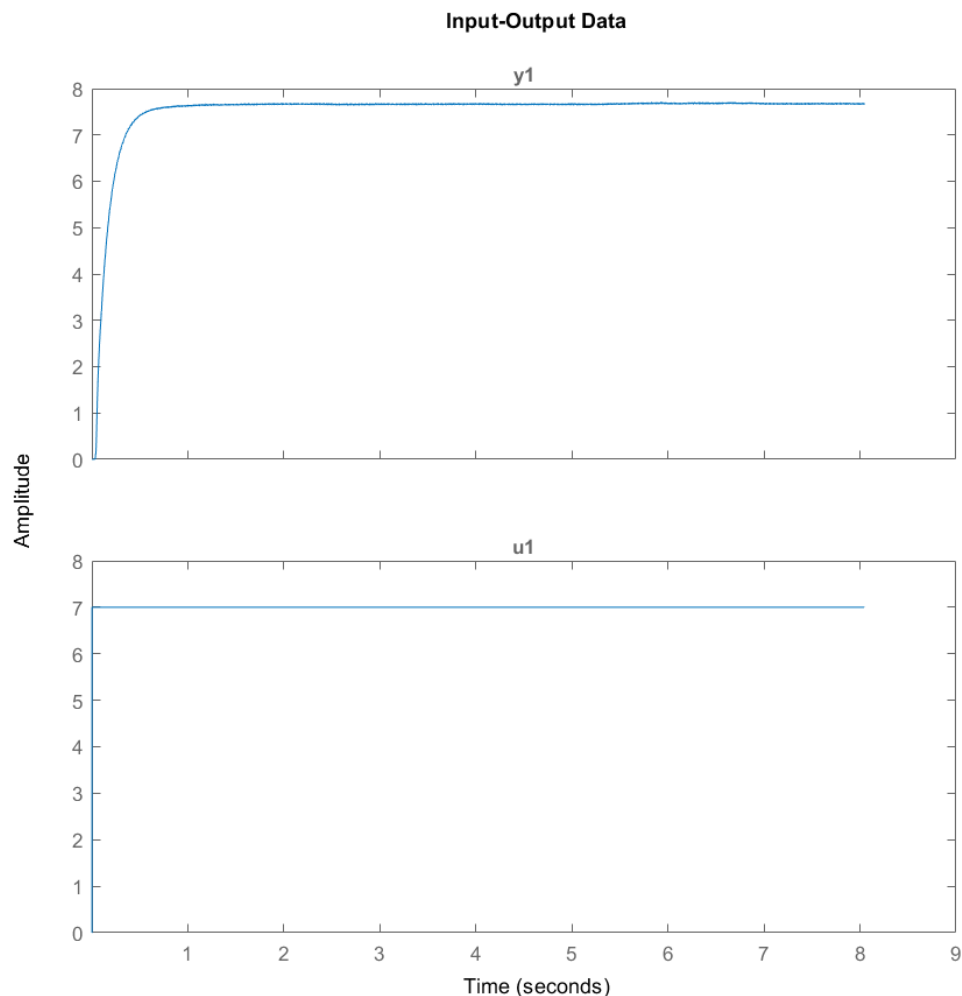


```
u = data_dc(:,2) %kolom kedua sebagai nilai input  
y = data_dc(:,3) %kolom ketiga sebagai nilai output
```

1.3.2 Identifikasi Sistem dengan ARX

Setelah melakukan import data input-output dari sistem pada workspace MATLAB maka kita dapat memodelkan data tersebut dengan metode ARX. Kita harus merubah data sebelumnya menjadi sebuah objek atau variabel *iddata* terlebih dahulu melalui perintah yaitu

```
Ts = 0.001 %waktu sampling  
data = iddata(y,u,Ts)  
  
idplot(data) %plot
```



Dari variabel *data* tersebut selanjutnya dapat dimodelkan dengan ARX(p,s) dengan perintah sederhana sebagai berikut

```
sys_1 = arx(data,[2 1 0]) %modelling ARX(2,1)
```



```
sys_2 = arx(data,[3 1 0]) %modelling ARX(3,1)  
sys_3 = arx(data,[3 2 0]) %modelling ARX(3,2)
```

Variabel *sys_1*, *sys_2*, *sys_3* terbentuk dalam polinomial atau objek *idpoly*. Kita selanjutnya dapat mengubahnya menjadi TF atau *Transfer Function*

```
%konversi polinomial ke TF
```

```
G1 = idtf(sys_1)  
G2 = idtf(sys_2)  
G3 = idtf(sys_3)
```

Dari ketiga model ARX yang telah digenerate, kita dapat menguji model mana yang terbaik berdasarkan beberapa kriteria yaitu seperti fit percent, MSE, maupun AIC-BIC sebagai berikut

```
compare(data,sys_1,sys_2,sys_3) %plot fit dengan data
```

```
%MSE
```

```
MSE_G1 = sys_1.Report.Fit.MSE  
MSE_G2 = sys_2.Report.Fit.MSE  
MSE_G3 = sys_3.Report.Fit.MSE
```

```
%AIC
```

```
AIC_G1 = sys_1.Report.Fit.AIC  
AIC_G2 = sys_2.Report.Fit.AIC  
AIC_G3 = sys_3.Report.Fit.AIC
```

```
%BIC
```

```
BIC_G1 = sys_1.Report.Fit.BIC  
BIC_G2 = sys_2.Report.Fit.BIC  
BIC_G3 = sys_3.Report.Fit.BIC
```

Semakin besar persen fit dan semakin kecil MSE menandakan bahwa model ARX yang dihasilkan semakin mendekati sistem (data yang digunakan). Sedangkan, AIC dan BIC melakukan perhitungan kesesuaian model dengan menggunakan pendekatan penalti terhadap kompleksitas model. Sehingga semakin kecil AIC dan BIC maka semakin bagus.



1.4 Metode Lainnya

Terdapat pula metode lain yang dapat digunakan untuk mencari fungsi alih sistem, diantaranya metode Viteckova, Latzel, Ziegler-Nichols, Strejc, dan Broida. Pada bagian ini akan digunakan data output dengan sinyal uji step pada bagian *Import Data* sebelumnya.

1.4.1 Metode 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* (C_{ss}), 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 nilai pada data terakhir
ind = find(smoothing > threshold)
Css = mean(output(ind))
```

Setelah dijalankan maka didapat nilai C_{ss} yaitu sebesar 0.4510 seperti pada nilai yang *tergenerate* pada workspace MATLAB.

Workspace	
Name ^	Value
ans	5
blackboxsystem...	1271x3 double
blackboxsystem...	128x2 double
C33	1269x1 double
C70	1267x1 double
Css	0.4510
datasistem	1271x3 double
ind	1213x1 double
input	1271x1 double
K	0.4510
N	127
O	7

Untuk pencarian nilai t_{33} (waktu saat mencapai 33% dari C_{ss}) dan t_{70} (waktu saat 70% dari C_{ss}), dapat dilakukan menggunakan command pada MATLAB ataupun dilakukan simulasi melalui SIMULINK.

```
%t33 dan t70 secara MATLAB, waktu yang didapatkan adalah waktu terdekat dari value yang dicari
V33 = 0.33*Css
C33 = find(output >= V33)
C33(1,1)
```




```
t33 = time(C33(1,1))
```

```
V70 = 0.7*Css
```

```
C70 = find(output >= V70)
```

```
C70(1,1)
```

```
t70 = time(C70(1,1))
```

%t33 dan t70 secara SIMULINK dengan menggunakan cursor measurement (nilai lebih akurat)

```
t33 = 0.113
```

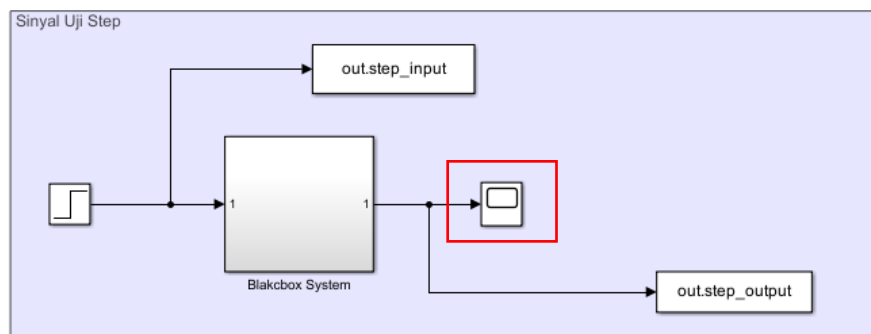
```
t70 = 0.346
```

```
R = 1
```

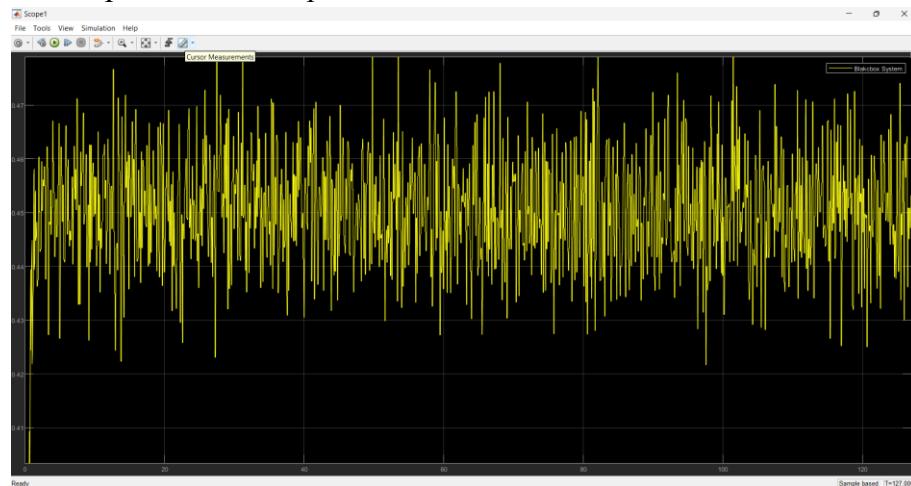
```
K = Css/R
```

Berikut adalah langkah untuk mencari nilai t_{33} dan t_{70} dengan menggunakan SIMULINK.

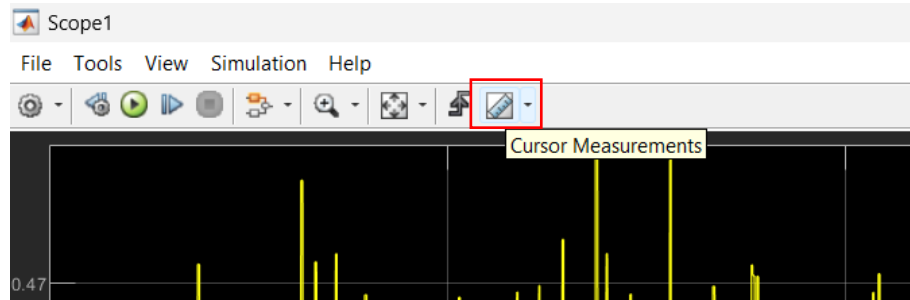
1. Sambungkan output sistem dengan sebuah scope



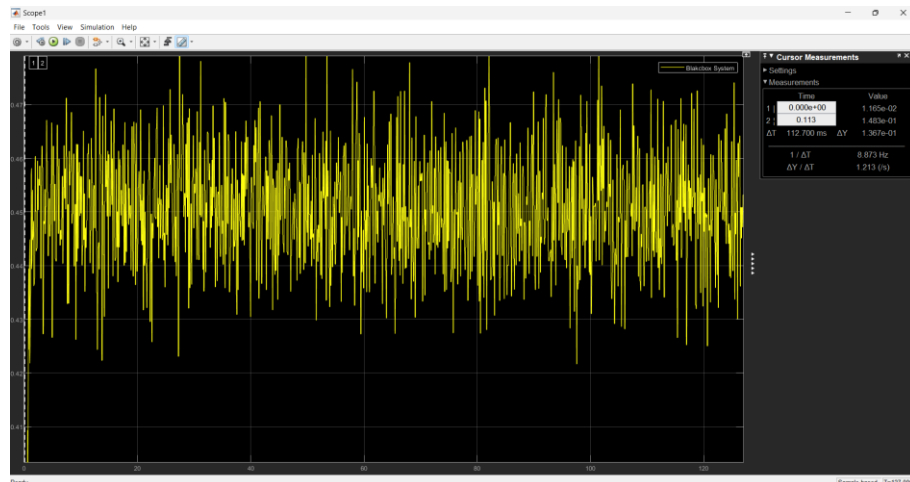
2. Jalankan simulasi untuk mencari respon sistem dengan sinyal uji berupa unit step serta buka *scope*



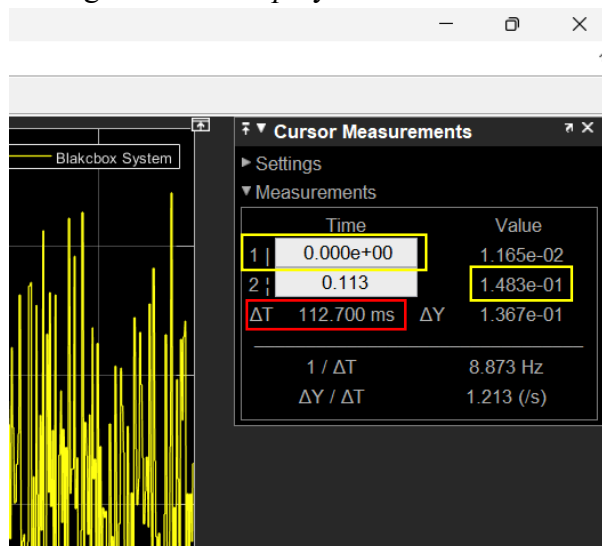
- Pilih “Cursor Measurements” pada bagian atas *scope*



- Pindahkan Cursor 1 ke nilai awal (t_0) dan Cursor 2 ke nilai yang ingin dicari yaitu 33% dari C_{ss} sebesar 0.14883 dan 70% dari C_{ss} sebesar 0.3157



Maka waktu yang dibutuhkan untuk mencapai nilai tersebut dapat diketahui pada bagian kanan *scope* yaitu ΔT



Pastikan waktu dari Cursor 1 bernilai 0 dan nilai dari Cursor 2 sesuai dengan yang dicari (33% maupun 70% C_{ss}) seperti pada bagian kotak kuning di atas.



Metode Viteckova terdapat 2 jenis yang akan dibahas berdasarkan orde yang digunakan yaitu Viteckova Orde 1 dan Orde 2 sebagai berikut

- **Viteckova Orde 1**

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

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

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

$$T_{dv1} = (1.498 \cdot t_{33}) - (0.498 \cdot t_{70})$$

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

$$G_{v1} = (K \cdot \exp(-T_{dv1} \cdot s)) / ((T_{v1} \cdot s) + 1)$$

- **Viteckova Orde 2**

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

$$T_{dv2} = 1.937t_{33} - 0.937t_{70}$$

$$T_{v2} = 0.794(t_{70} - t_{33})$$

$$T_{dv2} = (1.937 \cdot t_{33}) - (0.937 \cdot t_{70})$$

$$T_{v2} = 0.794 \cdot (t_{70} - t_{33})$$

$$G_{v2} = (K \cdot \exp(-T_{dv2} \cdot s)) / (((T_{v2} \cdot s) + 1)^2)$$

1.4.2 Metode 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_a 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	0,150	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

%t10, t50, t90 dengan MATLAB

```
V10 = 0.1*Css
C10 = find(output >= V10)
C10(1,1)
t10 = time(C10(1,1))

V50 = 0.5*Css
C50 = find(output >= V50)
C50(1,1)
t50 = time(C50(1,1))

V90 = 0.9*Css
C90 = find(output >= V90)
C90(1,1)
t90 = time(C90(1,1))
```

%t10, t50, t90 dengan Cursor Measurements SIMULINK

```
t10 = 0.113
t50 = 0.204
t90 = 0.67

miu = t10/t90
```

%nilai n, a10, a50, a90 berdasarkan miu pada tabel



$$n = 2$$

$$a_{10} = 1.88$$

$$a_{50} = 0.596$$

$$a_{90} = 0.257$$

$$TL = (a_{10} \cdot t_{10}) + (a_{50} \cdot t_{50}) + (a_{90} \cdot t_{90})$$

$$GL = K / (((TL \cdot s) + 1)^n)$$

1.4.3 Metode Ziegler-Nichols

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

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

%mencari tT (T ditambah L) dengan MATLAB

VtT = C_{ss}

CtT = find(output >= VtT)

CtT(1,1)

tT = time(CtT(1,1))

tT = 1.2 %SIMULINK

L = 0 %diasumsikan tanpa delay karena sangat kecil

T = tT - L

GZ = (K*exp((-L)*s))/(T*s+1)

1.4.4 Metode Strejc

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

$$T_U = L$$

$$T_N = T$$



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

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

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

dengan t_i adalah 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

$\tau = T_U/T_N$

%berdasarkan tabel yang paling mendekati

$n = 2$

$y_i = 0.264$

$C_i = \text{find}(\text{output} \geq y_i)$

$C_i(1,1)$

$t_i = \text{time}(C_i(1,1))$ %MATLAB

$t_i = 0.259$ %SIMULINK

$\tau_{ST} = t_i/(n-1)$

$G_{ST2} = K/((\tau_{ST}s+1)^n)$

1.4.5 Metode Broida

Metode Broida memerlukan data berupa waktu saat respon mencapai 28% dan 40% dari nilai *steady-state* (C_{ss}). Model ini memiliki persamaan model alih fungsi sebagai berikut

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

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

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



%nilai t28 dan t40 dengan MATLAB

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

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

%nilai t28 dan t40 dengan Cursor Measurements SIMULINK

```
t28 = 0.085  
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 Perbandingan Metode

Pada bab ini, kita coba bandingkan akurasi fungsi alih dari masing-masing metode yang dicoba sebelumnya dengan data sebenarnya. Digunakan metode *Mean Square Error* atau MSE dengan syntax '**mse**'

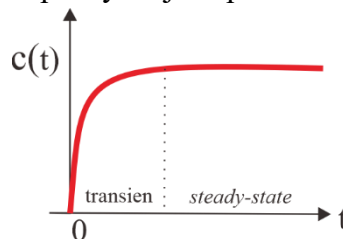
```
t = 0:0.1:127  
[yGv1,t] = step(Gv1,t)  
[yGv2,t] = step(Gv2,t)  
[yGL,t] = step(GL,t)  
[yGZ,t] = step(GZ,t)  
[yGB,t] = step(GB,t)  
[yGST2,t] = step(GST2,t)  
  
MSE_Gv1=mse(yGv1,output)
```



```
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

Pengetahuan mengenai karakteristik sistem diperlukan sebelum kita menganalisis dan mendesain sistem pengaturan. 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 karena karakteristik sistem menunjukkan ciri-ciri khusus dari respon output sistem. Tetapi, perlu dipahami terlebih dahulu karakteristik sistem terbagi menjadi dua yaitu karakteristik respon waktu dan frekuensi. Karakteristik respon waktu merupakan karakteristik respon terhadap perubahan waktu yang terbagi menjadi 2 yaitu transien dan *steady-state* (keadaan tunak) seperti pada gambar. Pada gambar tersebut sinyal $c(t)$ merupakan respon sistem terhadap sinyal uji step.



Karakteristik transien yaitu sistem pada saat memiliki respon yang masih berubah atau sementara ketika terjadi perubahan sinyal input dan *steady state* yaitu sistem pada saat memiliki respon yang cenderung tidak berubah atau menetap terhadap besar sinyal input yang sama (tunak). Sedangkan karakteristik respon frekuensi merupakan karakteristik respon sistem terhadap perubahan frekuensi. Selain itu pada bab ini akan dibahas kestabilan, *controllability*, dan *observability*.

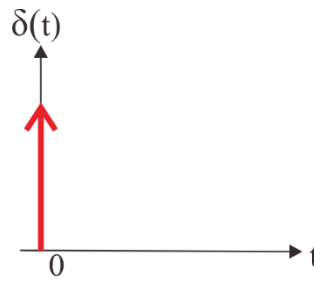
2.1 Sinyal Uji

Sinyal uji merupakan sinyal input yang biasa digunakan untuk menguji respon 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

2.1.1 Sinyal Impuls

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. Respon yang didapat menunjukkan

karakteristik respon transien yang memberikan gambaran internal sistem (karakteristik internal).

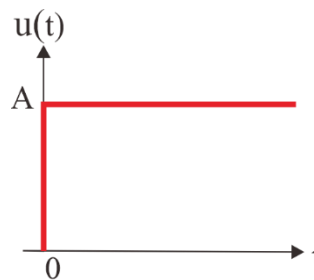


$$\lim_{t \rightarrow 0} \delta(t) = \infty$$

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

2.1.2 Sinyal Step

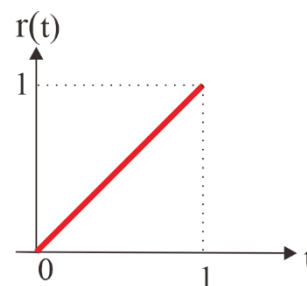
Sinyal step berguna untuk menguji respon terhadap gangguan yang muncul tiba-tiba, dan juga melihat kemampuan sistem kontrol dalam memposisikan respon. Respon yang didapat menunjukkan karakteristik respon transien (internal) dan *steady state* (eksternal).



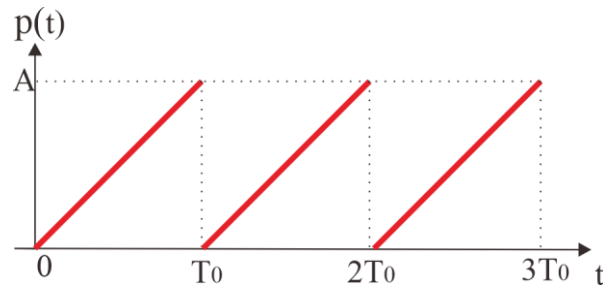
$$u(t) = \begin{cases} A, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

2.1.3 Sinyal Ramp Tunggal atau Periodik

Sinyal ramp merupakan fungsi berubah bertahap terhadap waktu. Sinyal ini berguna untuk melihat kemampuan sistem kontrol dalam melacak target yang bergerak dengan kecepatan konstan. Sehingga sinyal uji ini cocok digunakan untuk sistem yang memiliki sifat sebagai *tracking system*.



$$r(t) = \begin{cases} t, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

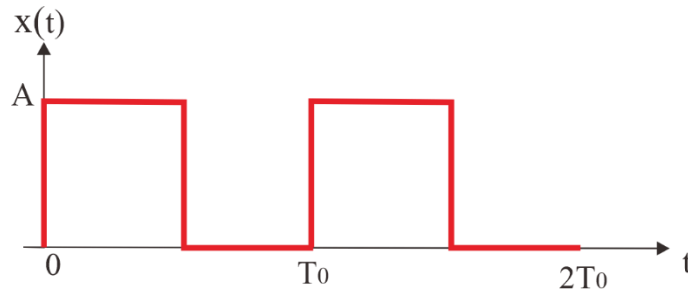


$$p(t) = \begin{cases} \frac{A}{T_0}, & 0 \leq t \leq T_0 \\ 0, & t < 0 \end{cases}$$

$$p(t) = p(t + kT_0), \quad k = 0, 1, 2, \dots$$

2.1.4 Sinyal Persegi

Sinyal persegi berguna untuk menguji respon sistem yang menerima input berubah secara tiba-tiba seperti step tetapi periodik bergantian (tidak satu nilai saja). Respon yang didapat digunakan untuk mendapatkan karakteristik respon frekuensi.

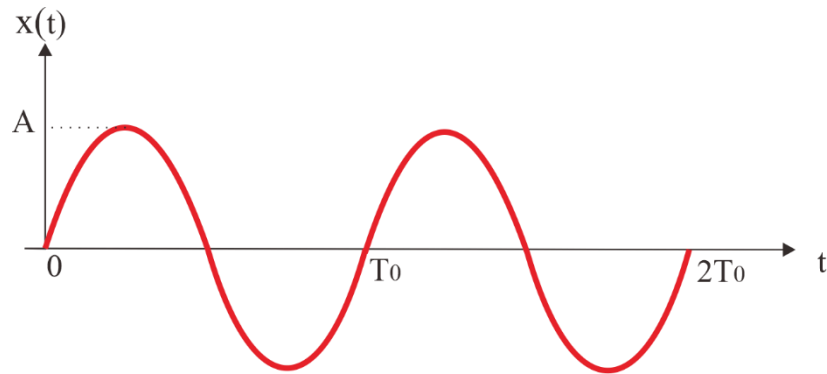


$$x(t) = \begin{cases} A, & 0 \leq t \leq \frac{T_0}{2} \\ 0, & \frac{T_0}{2} \leq t < T_0 \end{cases}$$

$$x(t) = x(t + kT_0), \quad k = 0, 1, 2, \dots$$

2.1.5 Sinyal Sinusoidal

Sinyal sinusoidal berguna untuk menguji respon sistem yang menerima input berupa sinyal sinusoidal dimana nilainya selalu berubah setiap saat dan periodik. Respon yang didapat digunakan untuk mendapatkan karakteristik respon frekuensi.

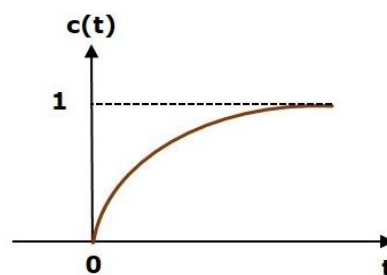


$$x(t) = A \sin(\omega_0 t)$$

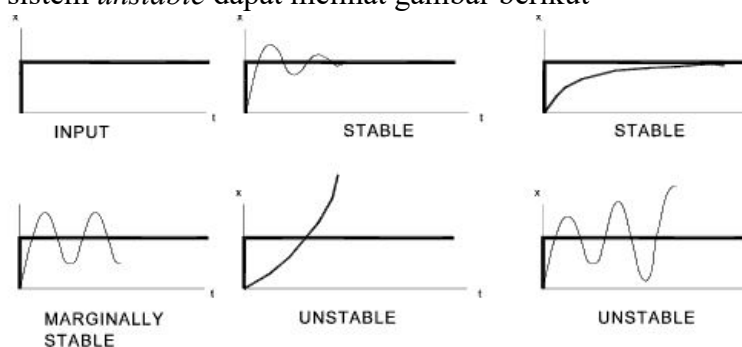
$$x(t) = x(t + kT_0), \quad k = 0, 1, 2, \dots$$

2.2 Analisis Root Locus

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



Gambar di atas adalah contoh respon step sistem yang stabil karena dalam keadaan *steady-state* nilai amplitudo dan frekuensi outputnya konstan berada di antara 0 dan 1 (*bounded output*). Selain itu, sinyal input berupa unit step 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. Terdapat 2 macam kestabilan dari suatu sistem yaitu

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' atau bagian *real* bernilai negatif.

2. *Marginally Stable System*

Sistem dikatakan *marginally stable* apabila *open loop transfer function* dan *closed loop transfer function* **terdapat** pole pada sumbu imajiner.

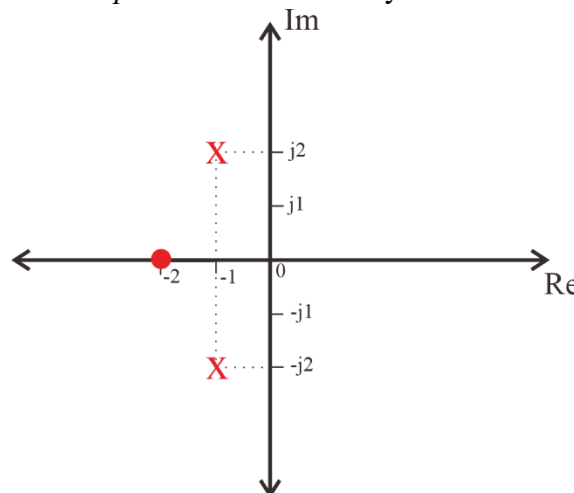
Bidang 's' sendiri merupakan bidang kartesian dimana sumbu x atau horizontal merupakan bilangan *real* dan sumbu y atau vertikal merupakan bilangan *imaginer*. Bidang 's' digunakan untuk menggambarkan letak dari *pole* dan *zero* suatu sistem. Misalkan kita memiliki suatu sistem dengan *transfer function* berikut

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

Kita dapat memfaktorkan persamaan tersebut menjadi

$$G(s) = \frac{s + 1}{(s + 1 + j2)(s + 1 - j2)}$$

Pole merupakan akar-akar dari penyebut dan *Zero* merupakan akar-akar dari pembilang dari bentuk rasional *transfer function* tersebut. Sehingga *zero* dari sistem adalah $s = -1$ dan *pole* dari sistem adalah $s = -1 \pm j2$. Bidang 's' yang menggambarkan letak dari *pole* dan *zero* sistem yaitu

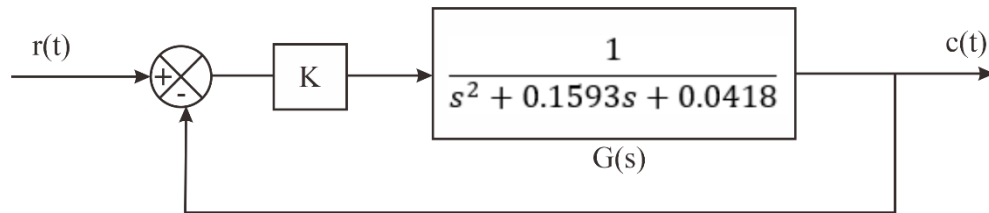


Simbol 'X' melambangkan *pole* dari sistem dan simbol '●' melambangkan *zero* dari sistem. Dapat dipahami pula bahwa sistem *open loop* tersebut *absolutely stable* karena semua *pole* berada pada sisi kiri bidang 's' atau bagian *real* dari *pole* bernilai negatif semua.

Berikut adalah contoh analisis kestabilan suatu sistem dengan menggunakan *root locus* dimana memiliki *transfer function* sebagai berikut

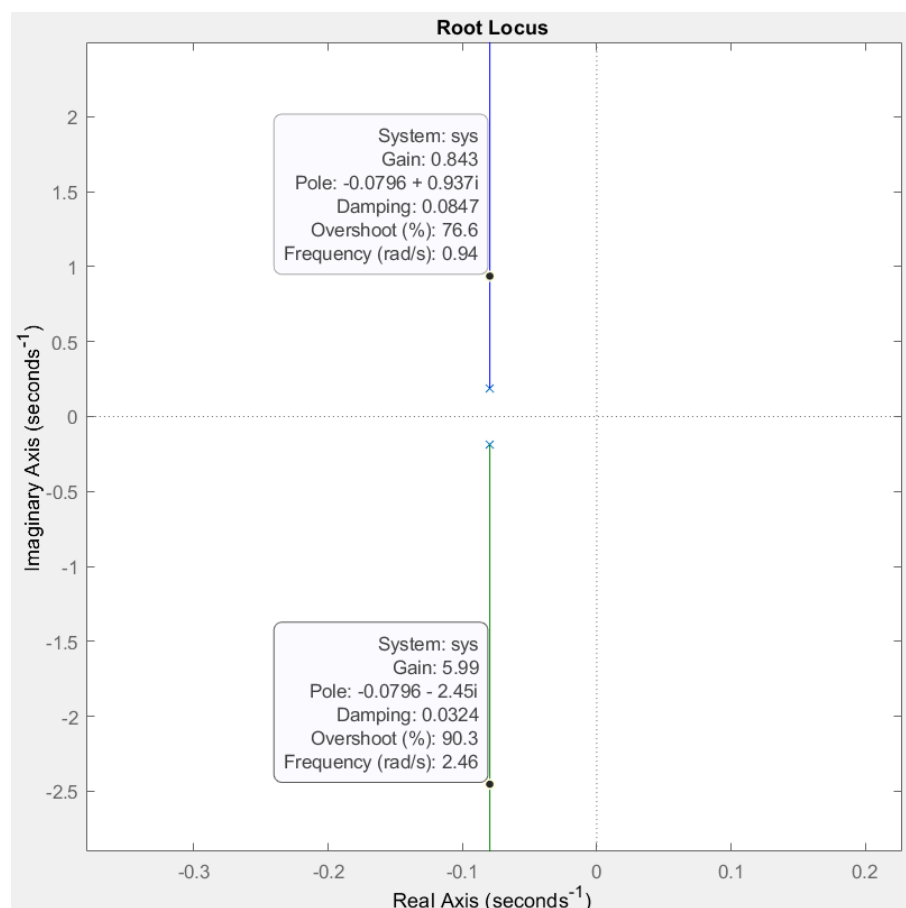
$$G(s) = \frac{1}{s^2 + 0.1593s + 0.0418}$$

Diagram blok *closed loop* dari sistem dengan *transfer function* tersebut dengan $r(t)$ adalah input dan $c(t)$ adalah output yaitu



Kita selanjutnya dapat menentukan gain K dimana dapat membuat sistem stabil dengan menggunakan plot *root locus*. Perintah berikut dapat digunakan pada MATLAB untuk menentukan plot *root locus* dari suatu sistem dimana secara *default* akan menggambarkan plot untuk $KG(s)$

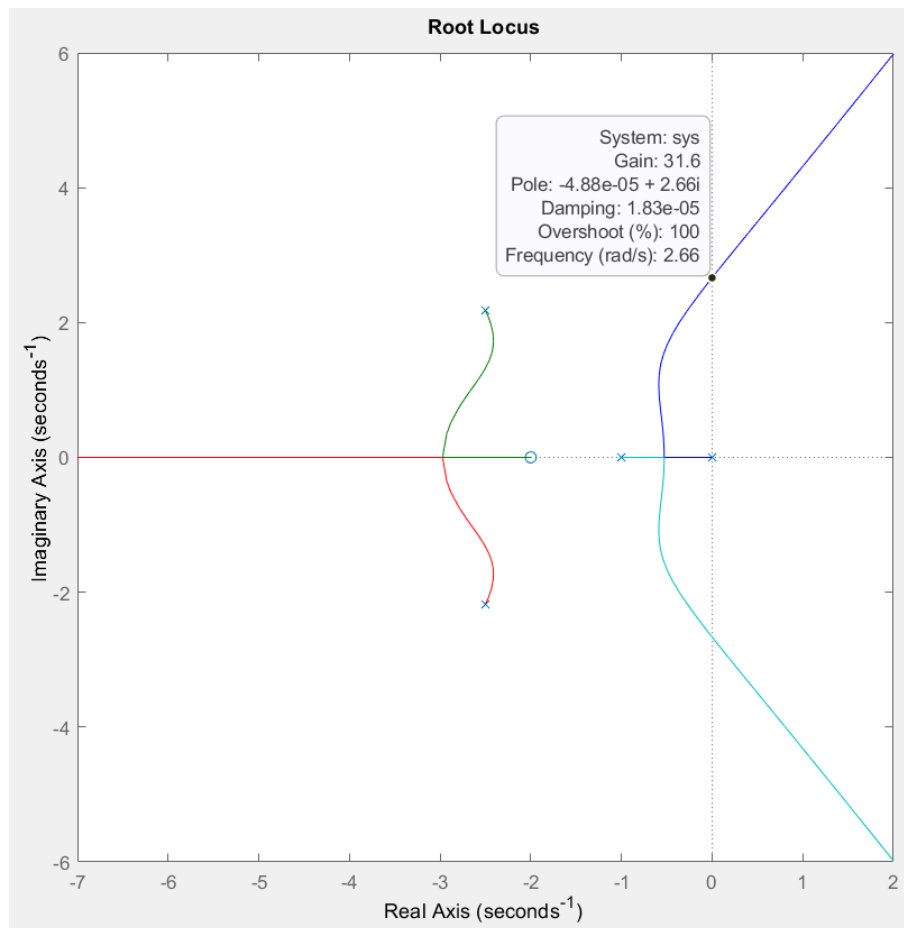
```
num = [1] %pembilang
den = [1 0.1593 0.0418] %penyebut
rlocus(num,den) %plot root locus KG(s)
```



Gain K yang dapat membuat stabil yaitu titik-titik pada plot *root locus* yang berada pada sisi kiri bidang 's'. Apabila kita klik suatu titik pada plot maka akan muncul berbagai informasi seperti pada gambar di atas termasuk **gain K**. Seperti sebelumnya, simbol 'X' melambangkan *pole* dari sistem dan simbol '●' melambangkan *zero* dari sistem. Selanjutnya akan kita coba untuk sistem lain dengan *transfer function* yaitu

$$G(s) = \frac{(s + 2)}{s(s + 1)(s^2 + 5s + 11)}$$

```
num = [1 2] %pembilang
den = conv(conv([1 0],[1 1]),[1 5 11]) %penyebut
rlocus(num,den) %plot root locus KG(s)
```



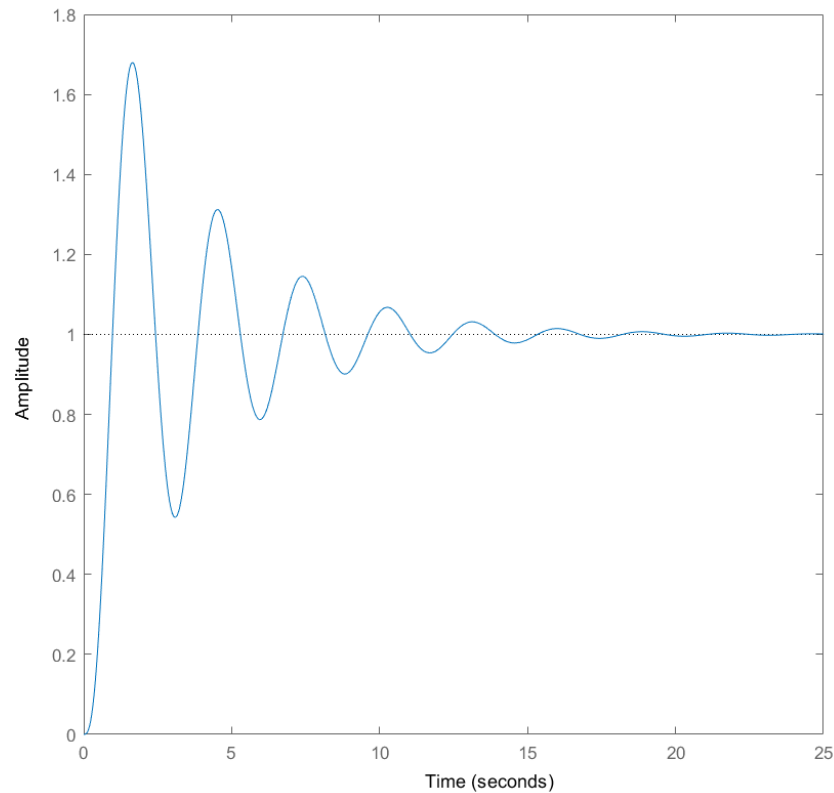
Dapat dilihat bahwa semua *pole* berada pada sisi kiri bidang 's' sehingga sistem stabil. Gain K yang berada pada titik sumbu imajiner tersebut yaitu sebesar 31.6 dapat membuat sistem *marginally stable*. Sehingga apabila diinginkan gain yang membuat sistem *absolute stable* maka kita memiliki gain K pada titik-titik sebelah kiri sumbu imajiner atau $K < 31.6$. Berikut adalah pembuktiannya

```
G = tf(num,den) %TF sistem

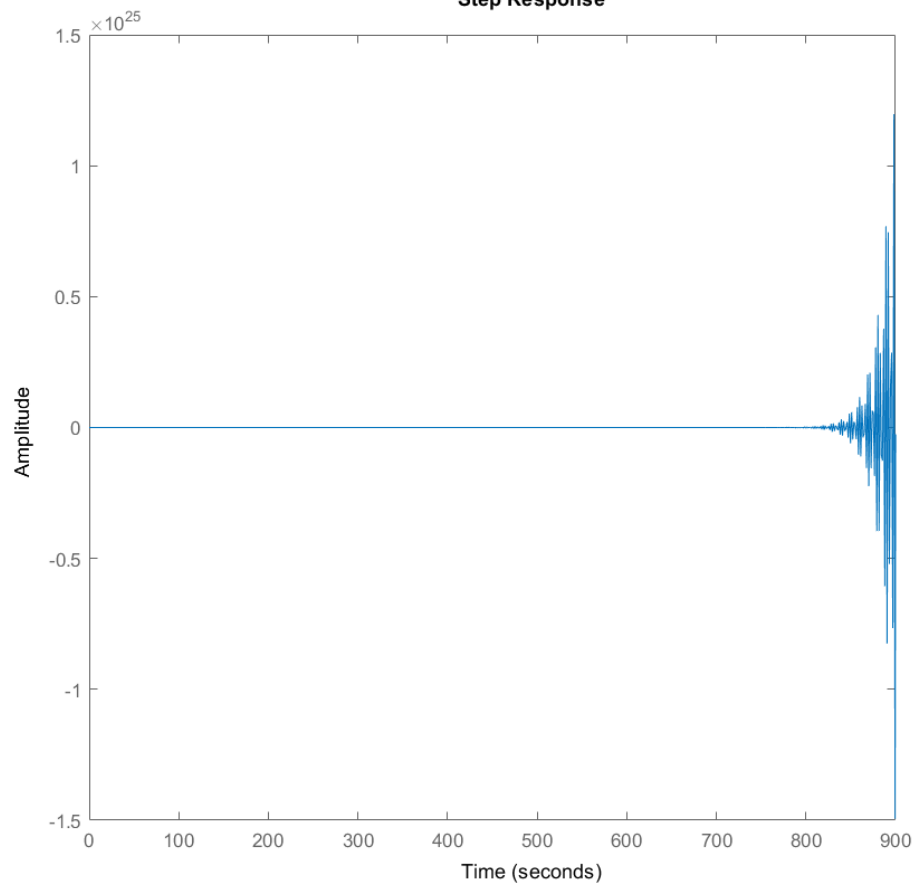
step(feedback(20*G,1)) %gain K < 31.6
step(feedback(35*G,1)) %gain K > 31.6
```



Step Response



Step Response





Gambar atas merupakan respon sistem terhadap sinyal input step dengan gain K sebesar 20 dan gambar bawah dengan gain K sebesar 35. Dapat dilihat bahwa sistem akan stabil apabila menggunakan gain kurang dari gain marginal atau 31.6.

2.3 Analisis State Space

Seperti yang sudah diketahui bahwa dinamika sistem dapat direpresentasikan dalam bentuk persamaan differensial maupun *transfer function*. Kita juga dapat merepresentasikan sistem dalam notasi vektor-matriks. Sistem dengan persamaan differensial **orde- n** dapat dinyatakan dengan persamaan differensial vektor-matriks **orde pertama**. Jika vektor dengan elemen sebanyak n adalah himpunan variabel *state* maka persamaan differensial vektor-matriks adalah persamaan *state*. *State* sendiri merupakan variabel yang digunakan untuk menggambarkan perilaku sistem pada waktu tertentu. Contohnya yaitu sistem motor DC dimana terdapat 3 variabel *state* yaitu posisi, kecepatan sudut, maupun percepatan sudut dari poros motor. Bentuk umum dari persamaan *state space* yaitu

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

dimana \mathbf{x} adalah vektor *state*, \mathbf{y} adalah vektor output, \mathbf{u} adalah vektor input atau kontrol, \mathbf{A} adalah matriks *state* atau sistem, \mathbf{B} adalah matriks input, \mathbf{C} adalah matriks output, dan \mathbf{D} adalah matriks *feedforward*. Dapat dilihat pada persamaan pertama, sistem dengan orde berapapun dapat direpresentasikan dalam bentuk persamaan differensial orde 1 saja. Kita dapat mengubah bentuk *transfer function* menjadi *state space*. Persamaan *state space* biasanya digunakan untuk mengecek *stability*, *controllability*, dan *observability* dari suatu sistem.

Misalkan diketahui sistem dengan bentuk persamaan *transfer function* yaitu

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

Untuk mendapatkan nilai matriks \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} atau mengubah bentuk TF ke SS dapat digunakan *syntax tf2ss* sebagai berikut

```
num = [0.008519]
den = [1 0.1593 0.0418]
G=tf(num,den) %TF sistem

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

2.3.1 Stability

Sistem dikatakan stabil apabila semua nilai eigen λ_n dari matriks \mathbf{A} terletak di sebelah kiri sumbu imajiner (nilai *realnya* negatif). Matriks \mathbf{A} dari persamaan *state space* yaitu:



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

Untuk mencari nilai *eigen* dari matriks A digunakan *syntax eig*

```
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.

2.3.2 Controllability

Sistem dikatakan *controllable* pada waktu t_0 apabila vektor kontrol (sinyal kontrol) memungkinkan untuk membawa sistem dari *state* awal $x(t_0)$ sembarang menuju nilai *state* lain dalam interval waktu terbatas. Suatu sistem *controllable* jika matriks *controllability* atau $M_c = [B|AB|A^2B|\dots|A^{n-1}B]$ mempunyai rank sama dengan n . Berdasarkan persamaan *state space* ($n = 2$) yang diperoleh pada sistem di atas 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 dari matriks tersebut, jika rank sama dengan n dapat disimpulkan sistem *controllable*. Kita dapat menghitung matriks M_c dari persamaan *state space* secara langsung melalui *syntax* berikut

```
Mc = ctrb(A,B) %matriks Mc
```

2.3.3 Observability

Sistem dikatakan *observable* pada waktu t_0 apabila dapat ditentukan nilai dari *state* dari observasi output dengan sistem dalam keadaan *state* $x(t_0)$ dalam interval waktu terbatas. Suatu sistem *observable* jika matriks *observability* atau M_o , mempunyai rank n .

$$M_o = \begin{bmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{n-1} \end{bmatrix}$$

Berdasarkan persamaan *state space* yang diperoleh maka matriks M_o sebagai berikut:



$$M_o = \begin{bmatrix} C \\ CA \end{bmatrix}$$

Dengan memasukkan matriks A dan C dan melakukan perhitungan dengan MATLAB

```
Mo = [C;C*A]
rank(Mo)
```

Nantinya akan didapatkan rank dari matriks tersebut, jika rank sama dengan n dapat disimpulkan sistem *observable*. Kita juga dapat menghitung matriks M_o dari persamaan *state space* secara langsung melalui *syntax* berikut

```
Mo = obsv(A,B) %matriks Mc
```

2.4 Analisis Domain Waktu

Time Domain memberikan gambaran terjadinya perubahan dinamika sistem 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. Seperti pada bab sebelumnya, 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.

2.4.1 Karakteristik Respon Orde 1

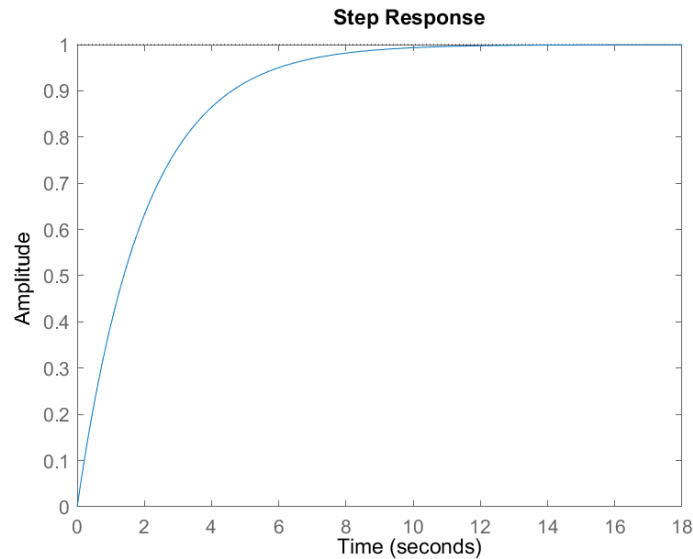
Untuk dapat mengetahui karakteristik respon orde 1, terlebih dahulu perlu diketahui persamaan *transfer function* dari *plant* atau sistem orde 1 yaitu

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

Sebagai contoh kita akan menguji sistem orde 1 dengan $K = 1$ dan τ atau konstanta waktu sebesar 2 dengan sinyal *unit step*

```
k=1;
tau=2;
g=tf(k,[tau 1]) %transfer function sistem

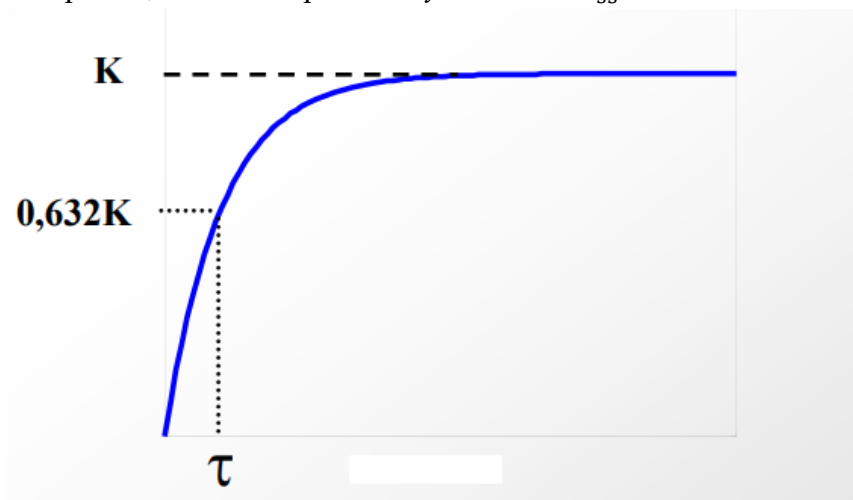
step(g) %respon uji step
```

Dari kurva respon sistem terhadap sinyal uji *unit step* didapatkan karakteristik sebagai berikut

1. Karakteristik Respon Transien

Sebelumnya perlu diketahui bahwa terdapat suatu konstanta waktu τ yang merupakan waktu untuk respon dari $t = 0$ atau waktu awal sampai mencapai 63,2% dari respon *steady state* atau C_{ss} .



a) Settling Time (t_s)

Settling time merupakan ukuran waktu yang menyatakan bahwa respon sistem telah masuk pada daerah stabil

- $t_s(\pm 5\%) \approx 3\tau$

$$t_s = 3 \cdot \tau \quad \% \text{settling time } \pm 5\%$$

- $t_s(\pm 2\%) \approx 4\tau$

$$t_s = 4 \cdot \tau \quad \% \text{settling time } \pm 2\%$$

- $t_s(\pm 0.5\%) \approx 5\tau$



$$t_s = 5 \cdot \tau \quad \% \text{settling time } \pm 0.5\%$$

b) *Rise Time (t_r)*

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*.

- $t_r(10\% - 90\%) \approx \tau \ln(9)$

$$t_r = \tau \cdot \log(9) \quad \% \text{rise time } 10\%-90\%$$

- $t_r(5\% - 95\%) \approx \tau \ln(19)$

$$t_r = \tau \cdot \log(19) \quad \% \text{rise time } 5\%-95\%$$

c) *Delay Time (t_d)*

Delay time atau waktu tunda yaitu waktu yang dibutuhkan respon mulai dari $t = 0$ atau waktu awal sampai respon mencapai 50% dari respon *steady state*. Besarnya faktor *delay* dari respon ini biasanya berdasarkan akibat dari proses sampling.

- $t_d \approx \tau \ln(2)$

$$t_d = \tau \cdot \log(2) \quad \% \text{delay time}$$

2. Karakteristik Respon *Steady State*

Karakteristik respon *steady state* yaitu *error* relative pada saat *steady state*. *Error* merupakan kesalahan atau selisih yang terjadi antara *set point* dengan respon keluaran yang dihasilkan. Error ini dapat dihitung dengan

$$\varepsilon_{ss} = \frac{X_{ss} - Y_{ss}}{X_{ss}} \times 100\%, \text{ dengan}$$

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

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

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

$$\text{error} = (1 - k) \cdot 100 \quad \% \text{error dalam persentase}$$

Pada MATLAB juga dapat menampilkan beberapa karakteristik dari sistem orde 1 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
```

2.4.2 Karakteristik Respon Orde 2

Pada orde 2 memiliki beberapa perbedaan dalam segi *transfer function* dan karakteristik dari orde 1. Persamaan umum dari sistem atau *plant* orde 2 adalah sebagai berikut

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

dimana K adalah *gain overall*, ξ adalah rasio redaman, dan ω_n adalah frekuensi tak teredam atau natural. Terdapat 3 keadaan redaman yang dapat terjadi pada suatu sistem

- *Under Damped*

Redaman kurang terjadi ketika nilai dari ξ kurang dari 1 atau $0 < \xi < 1$. Hal ini berarti *pole* yang dimiliki oleh sistem tersebut bernilai kompleks konjugat

$$p_{1,2} = -\xi\omega_n \pm j\omega_n\sqrt{1-\xi^2}$$

- *Critically Damped*

Redaman kritis terjadi ketika sistem teredam dengan maksimal atau $\xi = 1$. Hal ini berarti *pole* yang dimiliki oleh sistem merupakan akar kembar yaitu

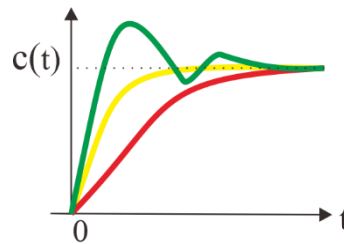
$$p_1 = p_2 = -\omega_n$$

- *Over Damped*

Redaman berlebih terjadi dikarenakan nilai dari $\xi > 1$. Hal ini berarti *pole* yang dimiliki oleh sistem bernilai *real* dan berbeda

$$p_{1,2} = -\xi\omega_n \pm \omega_n\sqrt{\xi^2-1}$$

Secara sekilas, ketiga kondisi redaman tadi dapat digambarkan oleh gambar di bawah

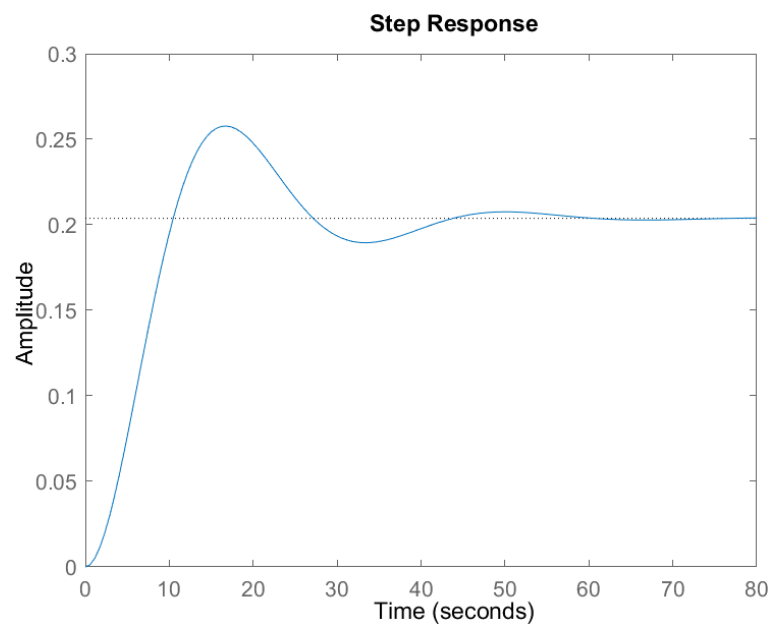


Kurva berwarna hijau merupakan kondisi *under damped*, kurva kuning merupakan kondisi *critically damped*, dan kurva merah merupakan kondisi *over damped*. Suatu sistem akan dicoba untuk memahami lebih dalam terkait karakteristik respon orde 2 sebagai berikut

```
k=0.20380382775;
wn=0.204450483;
e=0.38958; %underdamped 0<e<1

h=tf(k*wn^2,[1 2*e*wn wn^2]) %transfer function sistem

step(h) %respon uji step
```



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

Berikut penjelasan lebih lanjut mengenai karakteristik respon yang terdapat pada orde 2

1. Karakteristik Respon Transien



Terdapat 5 karakteristik transien yang dapat dipahami yaitu *settling time*, *delay time*, *rise time*, *peak time*, *overshoot* maksimum. Berikut adalah penjelasan serta persamaan untuk mencari nilai karakteristik tersebut

a) *Settling Time* (t_s)

Settling time merupakan ukuran waktu yang menyatakan bahwa respon sistem telah masuk pada daerah stabil.

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

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

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

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

b) *Delay Time* (t_d)

Delay time merupakan waktu yang diperlukan oleh respon untuk mencapai 50% dari respon *steady state* untuk waktu pertama.

- $t_d \approx \frac{1+0.7\xi}{\omega_n}$

```
td2 = (1+0.7*e)/wn %delay time
```

c) *Rise Time* (t_r)

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*.

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

dengan $\omega_d = \omega_n \sqrt{1 - \xi^2}$ dan $\beta = \tan^{-1} \left(\frac{\omega_d}{\xi \omega_n} \right)$

```
wd = wn*sqrt(1-e^2)
```

```
beta = atan(wd/(e*wn))
```

```
tr2 = (pi-beta)/wd %rise time
```

d) *Peak Time* (t_p)

Peak time atau waktu puncak yaitu waktu yang diperlukan oleh respon sistem untuk mencapai puncak *overshoot* pertama.



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

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

e) *Overshoot* Maksimum (M_p)

Overshoot maksimum merupakan nilai puncak maksimum dari respon sistem. *Overshoot* maksimum terjadi ketika respon mencapai waktu puncak atau $t = t_p$

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

Apabila dituliskan dalam persentase maka menjadi

$$\%M_p = \frac{M_p}{c(\infty)} \times 100\%$$

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

2. Karakteristik Respon *Steady State*

Karakteristik respon *steady state* yaitu *error* relatif pada saat *steady state*. *Error* merupakan kesalahan atau selisih 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
```

Pada MATLAB juga dapat menampilkan beberapa karakteristik dari sistem orde 2 tersebut dengan menggunakan

```
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

2.5 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.

2.5.1 Bode

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

1. Jika $GM > 1$ dan $PM > 0$ maka sistem tersebut stabil
2. Jika $GM = 1$ dan $PM = 0$ maka sistem tersebut stabil marginal
3. Jika $GM < 1$ dan $PM < 0$ maka sistem tersebut tidak stabil
4. GM tak hingga akan menghasilkan sistem yang selalu stabil

Setelah diketahui syarat-syarat kestabilan di atas, sistem juga akan diuji dengan sinyal step dalam *closed loop system*.

1. Contoh 1

Diketahui *transfer function* sebuah sistem *open loop* sebagai berikut

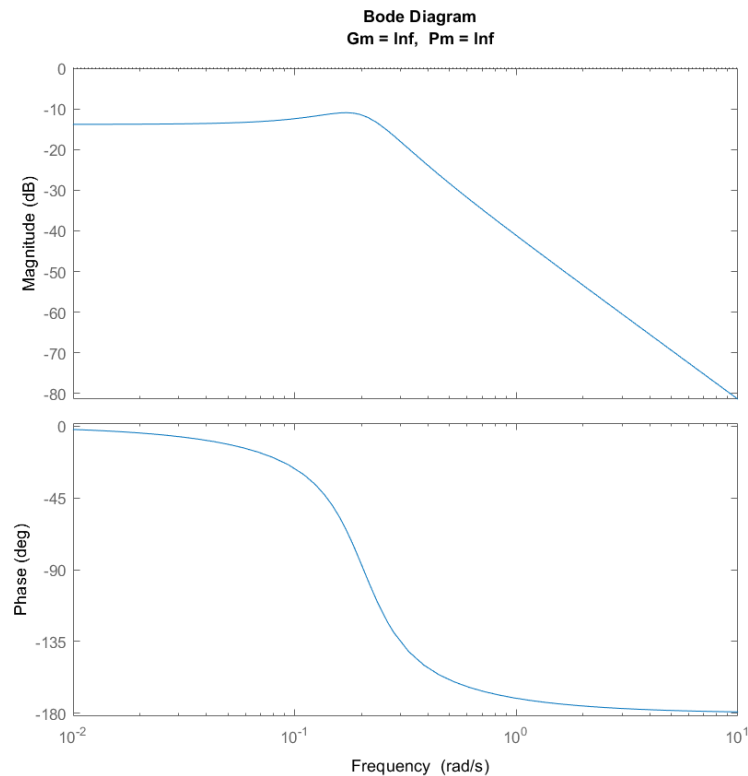
$$G(s) = \frac{0.008159}{s^2 + 0.1593s + 0.0418}$$

Tentukan kestabilan dari sistem dengan menggunakan plot Bode!

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

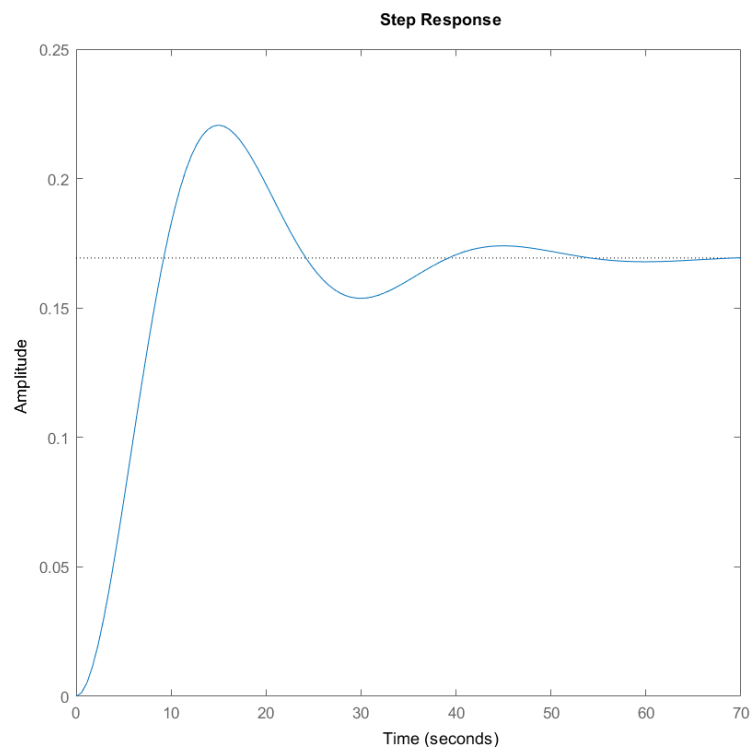
Gain Margin (GM) dan *Phase Margin* (PM) dapat dicari dengan menggunakan perintah berikut

```
margin(G)
```



Berdasarkan GM dari diagram Bode di atas (tak hingga) maka sistem dikatakan stabil. Selanjutnya akan diuji *unity feedback closed loop* sistem dengan sinyal input *unit step*

```
step(feedback(G,1)) %respon closed loop
```





Terbukti bahwa sistem stabil dengan respon seperti kurva di atas.

2. Contoh 2

Diketahui *transfer function* sebuah sistem *open loop* sebagai berikut

$$G(s) = \frac{1}{s + 1}$$

Tentukan kestabilan dari sistem dengan menggunakan plot Bode!

```
%transfer function sistem
```

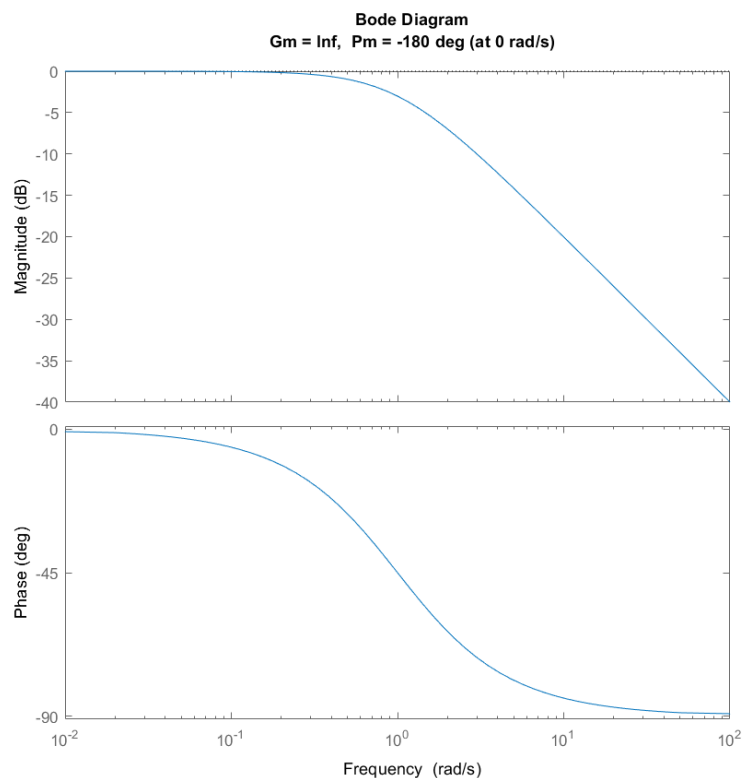
```
num = [1];
```

```
den = [1 1];
```

```
P = tf(num,den)
```

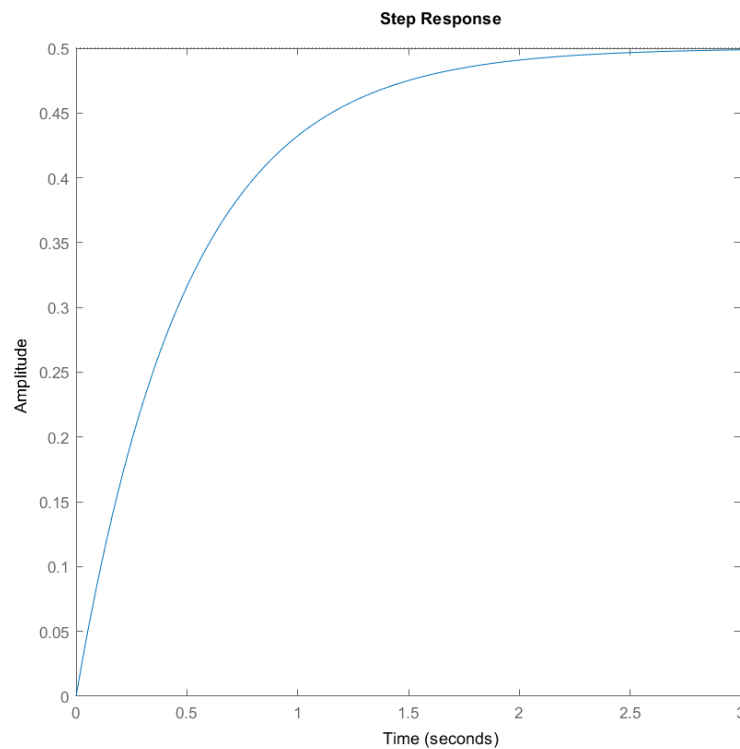
```
%plot Bode, GM, dan PM
```

```
margin(P)
```



Berdasarkan GM (tak hingga) dari diagram Bode di atas maka sistem dikatakan stabil. Selanjutnya akan diuji *unity feedback closed loop* sistem dengan sinyal input *unit step*

```
step(feedback(P,1)) %respon closed loop
```



Terbukti bahwa sistem stabil dengan respon seperti kurva di atas.

3. Contoh 3

Diketahui *transfer function* sebuah sistem *open loop* sebagai berikut

$$G(s) = \frac{1}{s^3 + 2s^2 + s}$$

Tentukan kestabilan dari sistem dengan menggunakan plot Bode!

```
%transfer function sistem
```

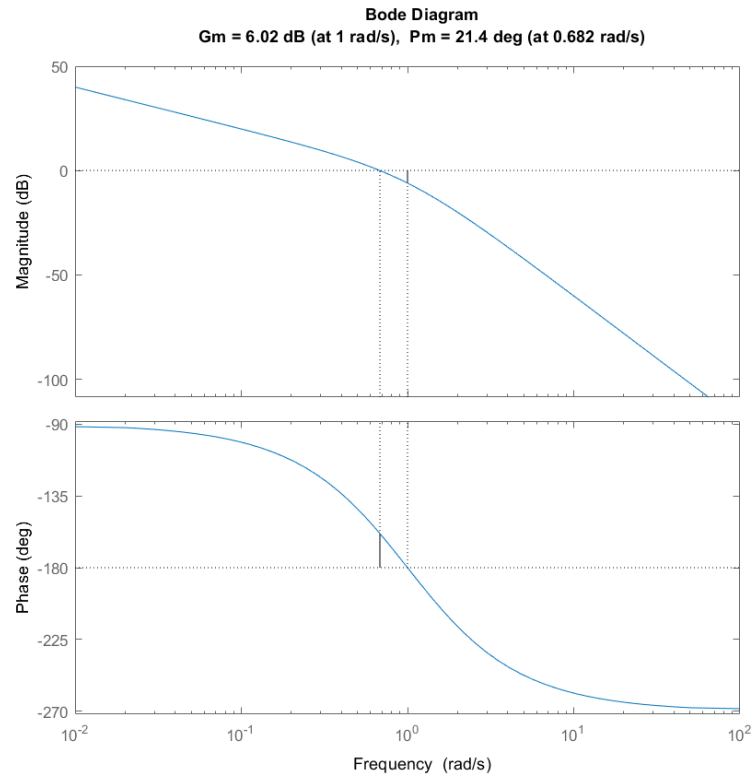
```
num = [1];
```

```
den = [1 2 1 0];
```

```
H = tf(num,den)
```

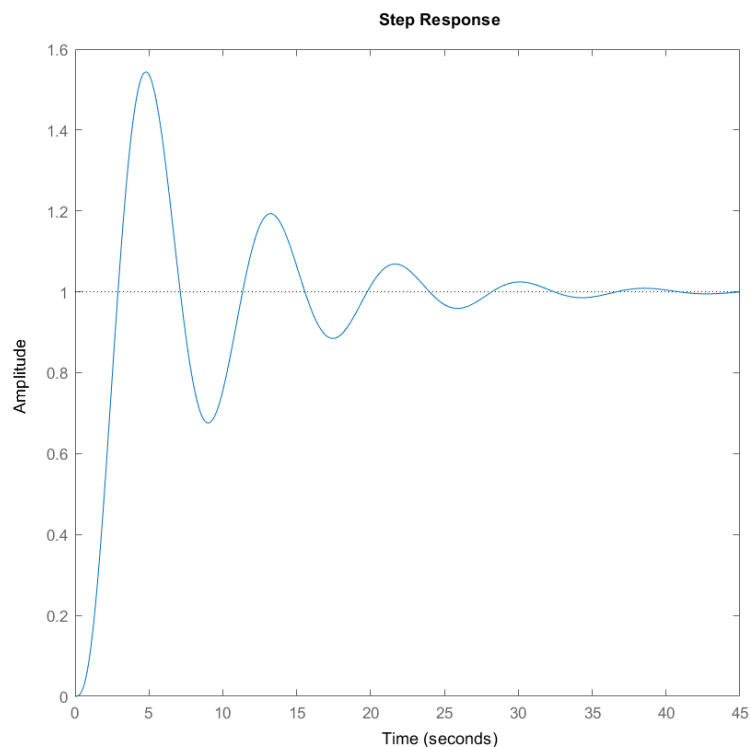
```
%plot Bode, GM, dan PM
```

```
margin(H)
```



Berdasarkan GM (>1) dan PM (>0) dari diagram Bode di atas maka sistem dikatakan stabil. Selanjutnya akan diuji *unity feedback closed loop* sistem dengan sinyal input *unit step*

```
step(feedback(H,1)) %respon closed loop
```





Terbukti bahwa sistem stabil dengan respon seperti kurva di atas.

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)
```

2.5.2 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 atau imajiner.
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 sumbu j atau imajiner.
3. Ada satu atau lebih yang melingkupi titik $-1+j0$ searah jarum jam. Sistem tersebut tidak stabil.

Setelah diketahui syarat-syarat kestabilan di atas, sistem juga akan diuji dengan sinyal step dalam *closed loop system*.

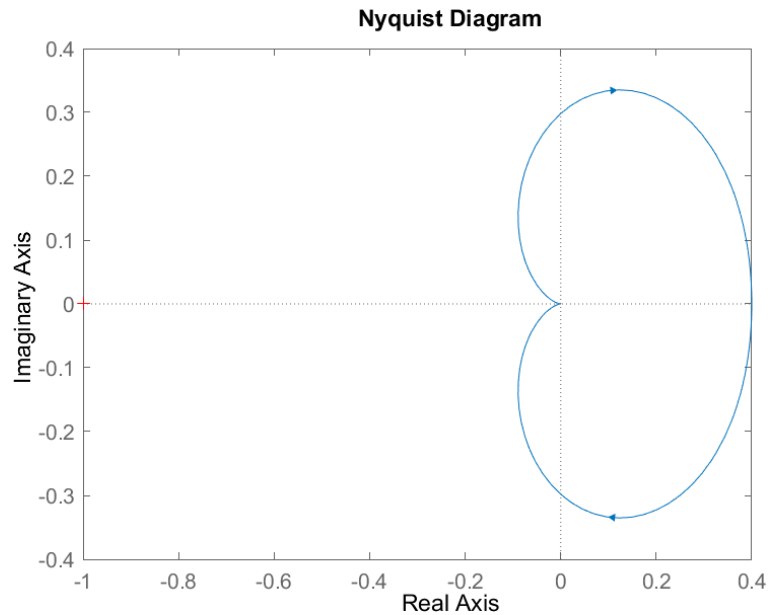
1. Contoh 1

Diketahui *transfer function* sebuah sistem *open loop* sebagai berikut

$$G(s) = \frac{2}{s^2 + 3s + 5}$$

Tentukan kestabilan dari sistem dengan menggunakan plot Nyquist!

```
G = tf(2,[1 3 5]) %transfer function  
nyquist(G) %plot nyquist
```

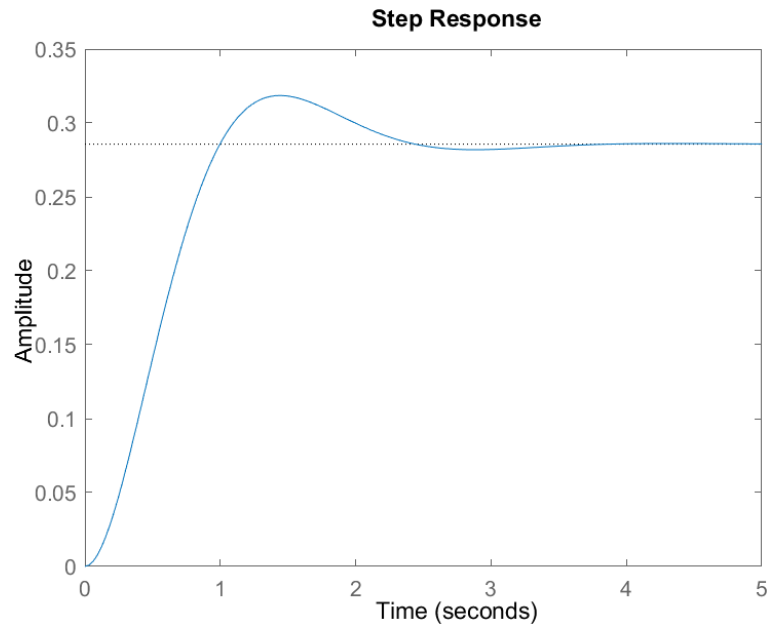
Dari diagram Nyquist tersebut dapat dilihat bahwa plot tidak mengelilingi titik $-1+j0$, maka diagram tersebut masuk ke kategori 1 dan sekarang tinggal dicari pole nya

```
pole(G)
```

```
ans = 2x1 complex  
-1.5000 + 1.6583i  
-1.5000 - 1.6583i
```

Dari pole yang didapatkan dapat disimpulkan bahwa sistem stabil karena semua pole berada di sebelah kiri sumbu imajiner (bagian *real* negatif). Selanjutnya akan diuji *unity feedback closed loop* sistem dengan sinyal input *unit step*

```
step(feedback(G,1)) %respon closed loop
```



Terbukti bahwa sistem stabil dengan respon seperti kurva di atas.

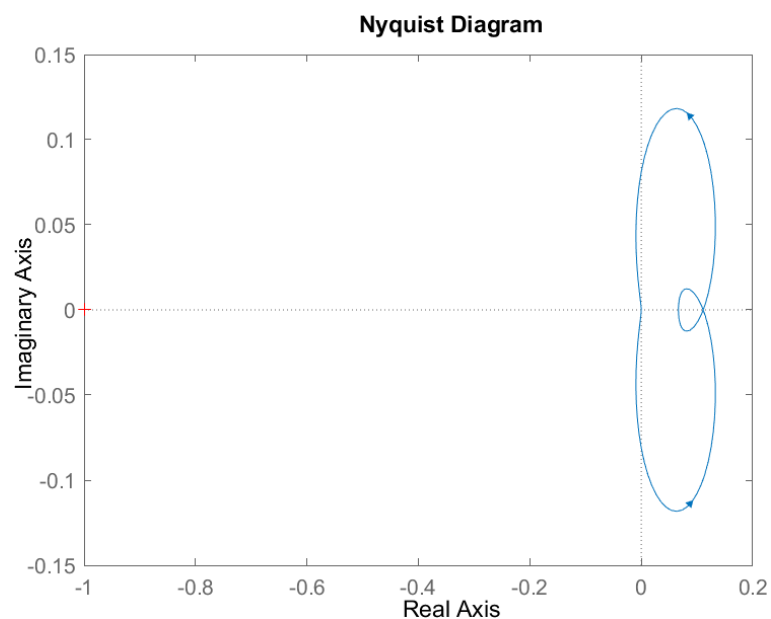
2. Contoh 2

Diketahui *transfer function* sebuah sistem *open loop* sebagai berikut

$$G(s) = \frac{1}{s^3 + 2s^2 + 3s + 15}$$

Tentukan kestabilan dari sistem dengan menggunakan plot Nyquist!

```
P = tf(1,[1 2 3 15]) %transfer function  
nyquist(P) %plot nyquist
```





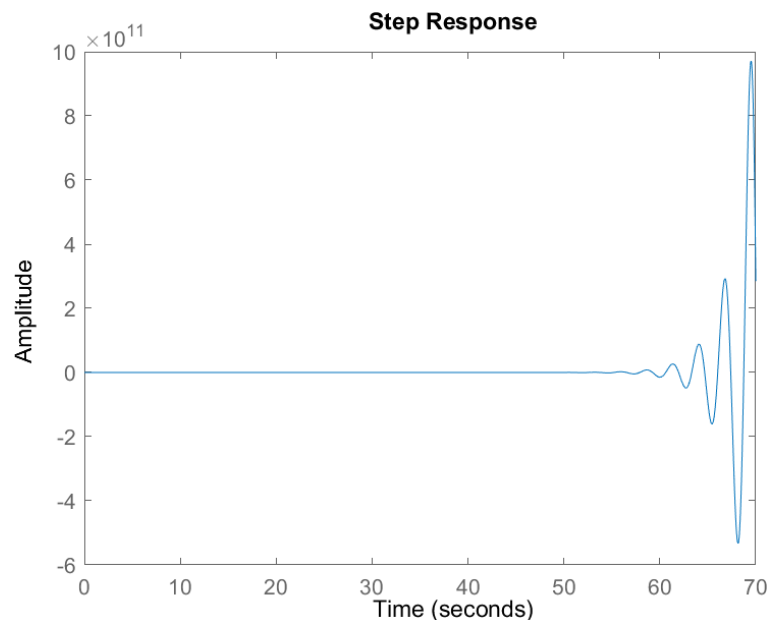
Dari diagram Nyquist tersebut dapat dilihat bahwa plot tidak mengelilingi titik $-1+j0$, maka diagram tersebut masuk ke kategori 1 dan sekarang tinggal dicari pole nya

```
pole(P)
```

```
ans = 3x1 complex  
-2.8212 + 0.0000i  
0.4106 + 2.2690i  
0.4106 - 2.2690i
```

Dari pole yang didapatkan dapat disimpulkan bahwa sistem tidak stabil karena terdapat 2 pole yang berada di sebelah kanan sumbu imajiner (bagian *real* positif). Selanjutnya akan diuji *unity feedback closed loop* sistem dengan sinyal input *unit step*

```
step(feedback(P,1)) %respon closed loop
```



Terbukti bahwa sistem tidak stabil dengan respon seperti kurva di atas.

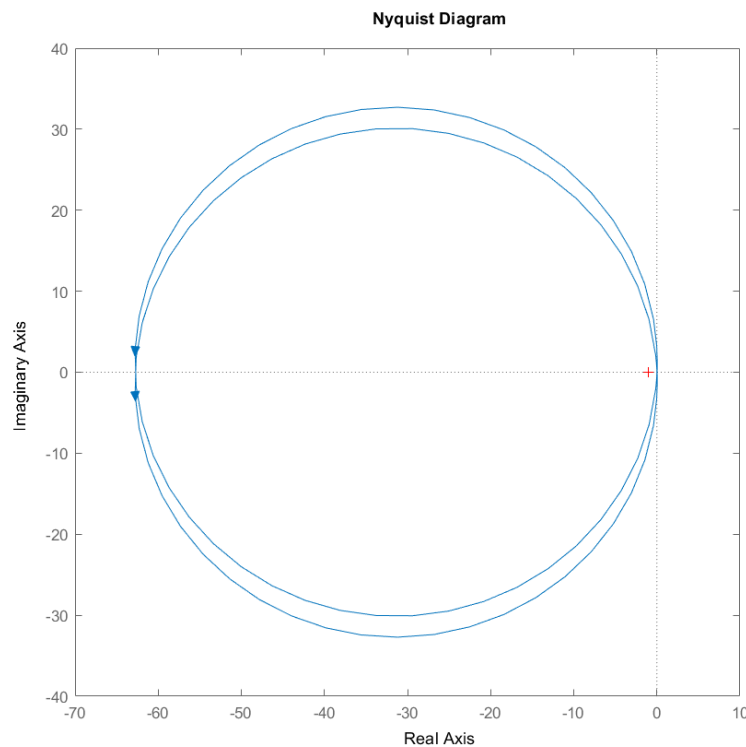
3. Contoh 3

Diketahui *transfer function* sebuah sistem *open loop* sebagai berikut

$$G(s) = \frac{s + 0.0519}{s^2 - 0.01593s + 1.5913}$$

Tentukan kestabilan dari sistem dengan menggunakan plot Nyquist!

```
H = tf([1 0.0519],[1 -0.01593 1.5913]) %transfer  
function  
nyquist(H) %plot nyquist
```



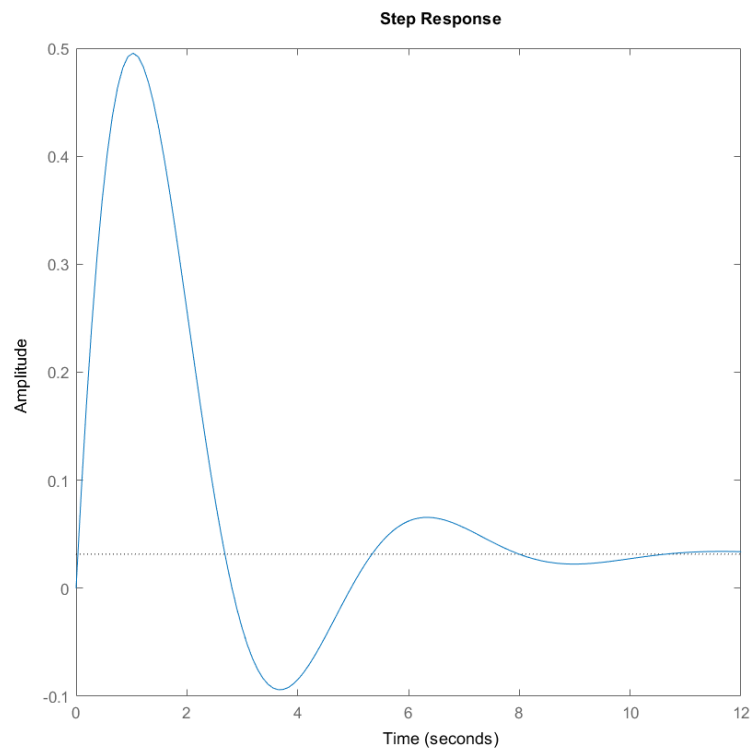
Dari diagram Nyquist tersebut dapat dilihat bahwa plot mengelilingi titik $-1+j0$ berlawanan arah jarum jam sebanyak 2 kali (dilihat dari arah panah), maka diagram tersebut masuk ke kategori 2 dan sekarang tinggal dicari pole nya

```
pole(H)
```

```
ans = 2x1 complex  
    0.0080 + 1.2614i  
    0.0080 - 1.2614i
```

Dari pole yang didapatkan dapat disimpulkan bahwa sistem stabil karena terdapat 2 pole yang berada di sebelah kanan sumbu imajiner (bagian *real* positif) atau sebanyak putaran plot berlawanan arah jarum jam. Selanjutnya akan diuji *unity feedback closed loop* sistem dengan sinyal input *unit step*

```
step(feedback(H,1)) %respon closed loop
```



Terbukti bahwa sistem stabil dengan respon seperti kurva di atas.