https://www.halvorsen.blog

Frequency Response with MATLAB Examples

Control Design and Analysis

Hans-Petter Halvorsen

Contents

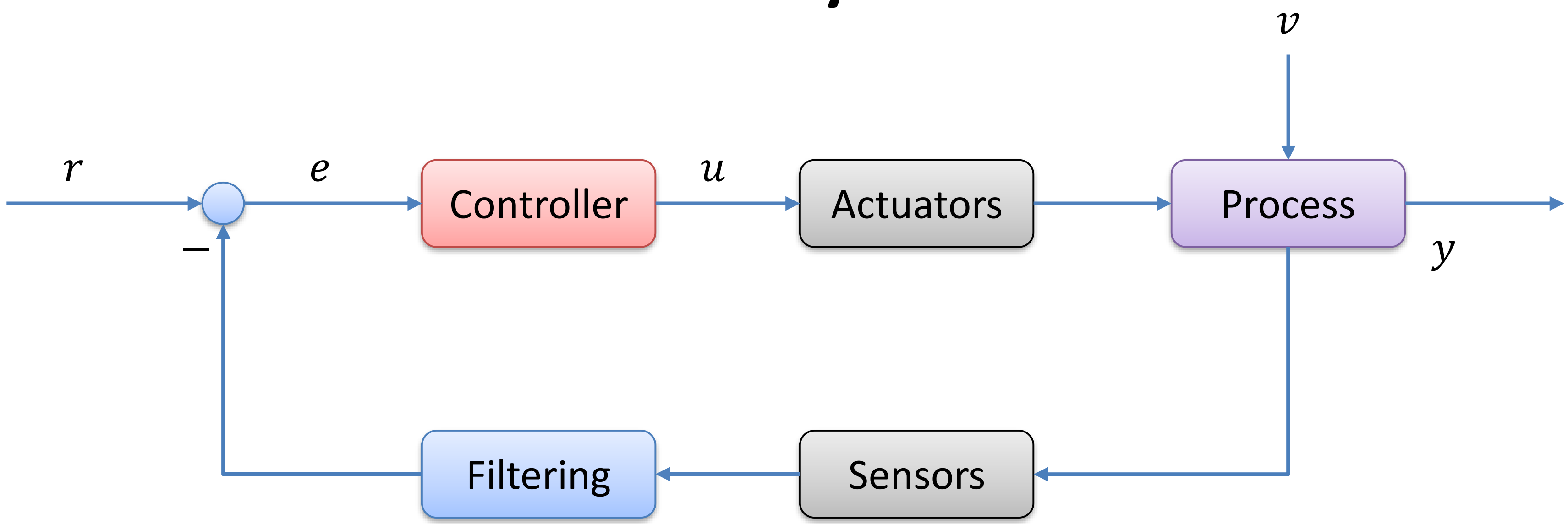
- Introduction to PID Control
- Introduction to Frequency Response
- Frequency Response using Bode Diagram
- Introduction to Complex Numbers (which Frequency Response Theory is based on)
- Frequency Response from Transfer Functions
- Frequency Response from Input/output Signals
- PID Controller Design and Tuning (Theory)
- PID Controller Design and Tuning using MATLAB
- Stability Analysis using MATLAB
- Stability Analysis of Feedback Systems
- Stability Analysis of Feedback Systems – a Practical Example
- The Bandwidth of the Control System
- Practical PI Controller Example

https://www.halvorsen.blog

PID Control

Hans-Petter Halvorsen

Control System



r – Reference Value, SP (Set-point), SV (Set Value)
y – Measurement Value (MV), Process Value (PV)
e – Error between the reference value and the measurement value ($e = r - y$)
v – Disturbance, makes it more complicated to control the process
u - Control Signal from the Controller

The PID Algorithm

$$u(t) = K_p e + \frac{K_p}{T_i}\int_0^t e d\tau + K_p T_d \dot{e}$$

Where u is the controller output and e is the control error:

$$e(t) = r(t) - y(t)$$

r is the Reference Signal or Set-point

y is the Process value, i.e., the Measured value

Tuning Parameters:

K_p Proportional Gain

T_i Integral Time [sec.]

T_d Derivative Time [sec.]

The PI Algorithm

$$u(t) = K_p e + \frac{K_p}{T_i}\int_0^t e d\tau$$

Where u is the controller output and e is the control error:

$$e(t) = r(t) - y(t)$$

r is the Reference Signal or Set-point

y is the Process value, i.e., the Measured value

Tuning Parameters:

K_p Proportional Gain

T_i Integral Time [sec.]

PI(D) Algorithm in MATLAB

- We can use the *pid()* function in MATLAB
- We can define the PI(D) transfer function using the *tf()* function in MATLAB
- We can also define and implement a discrete PI(D) algorithm

Discrete PI Controller Algorithm

We start with:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i}\int_0^t e d\tau$$

In order to make a discrete version using, e.g., Euler, we can derive both sides of the equation:

$$\dot{u} = \dot{u}_0 + K_p \dot{e} + \frac{K_p}{T_i} e$$

If we use Euler Forward we get:

$$\frac{u_k - u_{k-1}}{T_s} = \frac{u_{0,k} - u_{0,k-1}}{T_s} + K_p \frac{e_k - e_{k-1}}{T_s} + \frac{K_p}{T_i} e_k$$

Then we get:

$$u_k = u_{k-1} + u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

Where

$$e_k = r_k - y_k$$

We can also split the equation above in 2 different parts by setting:

$$\Delta u_k = u_k - u_{k-1}$$

This gives the following PI control algorithm:

$$e_k = r_k - y_k$$

$$\Delta u_k = u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

$$u_k = u_{k-1} + \Delta u_k$$

This algorithm can easily be implemented in MATLAB

Discrete PI Controller Algorithm

Discrete PI control algorithm:

$$e_k = r_k - y_k$$

$$\Delta u_k = u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

$$u_k = u_{k-1} + \Delta u_k$$

PID Controller –Transfer Function

We have:

$$u(t) = K_p e + \frac{K_p}{T_i}\int_0^t e d\tau + K_p T_d \dot{e}$$

Laplace gives:

$$H_{PID}(s) = \frac{u(s)}{e(s)} = K_p + \frac{K_p}{T_i s} + K_p T_d s$$

or:

$$H_{PID}(s) = \frac{u(s)}{e(s)} = \frac{K_p(T_d T_i s^2 + T_i s + 1)}{T_i s}$$

PI Controller –Transfer Function

We have:

$$u(t) = K_p e + \frac{K_p}{T_i}\int_0^t e d\tau$$

Laplace gives:

$$H_{PI}(s) = \frac{u(s)}{e(s)} = K_p + \frac{K_p}{T_i s} = \frac{K_p T_i s + K_p}{T_i s} = \frac{K_p(T_i s + 1)}{T_i s}$$

Finally:

$$H_{PI}(s) = \frac{u(s)}{e(s)} = \frac{K_p(T_i s + 1)}{T_i s}$$

Define PI Transfer function in MATLAB

$$H_{PI}(s) = \frac{u(s)}{e(s)} = \frac{K_p(T_i s + 1)}{T_i s}$$


```
clear, clc

% PI Controller Transfer function
Kp = 0.52;
Ti = 18;

num = Kp*[Ti, 1];
den = [Ti, 0];

Hpi = tf(num,den)

...
```

PI Controller – State space model

Given:

$$u(s) = K_p e(s) + \frac{K_p}{T_i s} e(s)$$

We set $z = \frac{1}{s} e \Rightarrow sz = e \Rightarrow \dot{z} = e$

This gives:

$$\dot{z} = e$$

$$u = K_p e + \frac{K_p}{T_i} z$$

Where

$$e = r - y$$

PI Controller – Discrete State space model

Using Euler:

$$\dot{z} \approx \frac{z_{k+1} - z_k}{T_s}$$

Where T_s is the Sampling Time.

This gives:

$$\frac{z_{k+1} - z_k}{T_s} = e_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

Finally:

$$e_k = r_k - y_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

$$z_{k+1} = z_k + T_s e_k$$

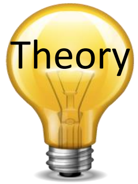
PI Controller – Discrete State space model implemented in MATLAB

```
clear, clc
...

for i=1:N

...

  e = r - y;
  u = Kp*e + z;
  z = z + dt*Kp*e/Ti;

  ...

end

plot(...)
```

https://www.halvorsen.blog

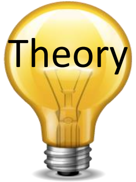
Frequency Response

Hans-Petter Halvorsen

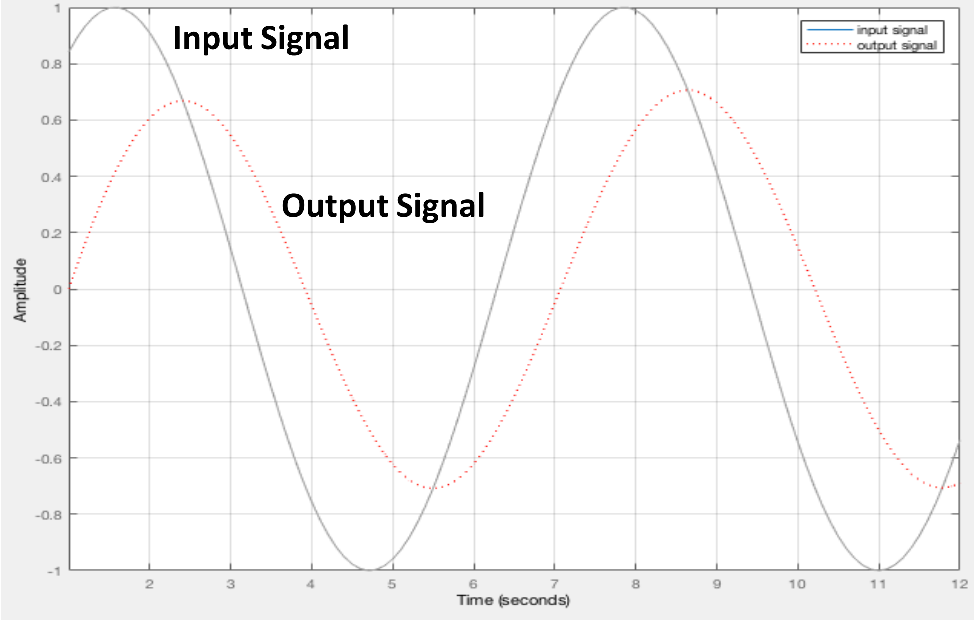


Frequency Response

- The frequency response of a system is a frequency dependent function which expresses how a sinusoidal signal of a given frequency on the system input is transferred through the system. Each frequency component is a sinusoidal signal having certain amplitude and a certain frequency.
- The frequency response is an important tool for analysis and design of signal filters and for analysis and design of control systems.
- The frequency response can be found experimentally or from a transfer function model.
- The frequency response of a system is defined as the steady-state response of the system to a sinusoidal input signal. When the system is in steady-state, it differs from the input signal only in amplitude/gain (A) and phase lag (ϕ).



Frequency Response



Input Signal

$$u(t) = U \cdot \ sin\omega t$$

Amplitude

Frequency

Dynamic System

$$H(s)$$

Output Signal

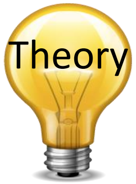
$$y(t) = \underbrace{UA}_{Y} \ sin(\omega t + \phi)$$

Gain

Phase Lag

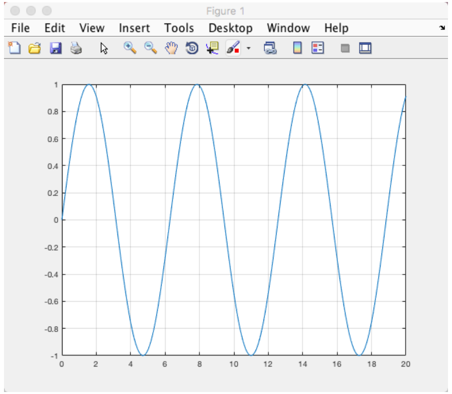
The frequency response of a system expresses how a sinusoidal signal of a given frequency on the system input is transferred through the system. The only difference is the gain and the phase lag.

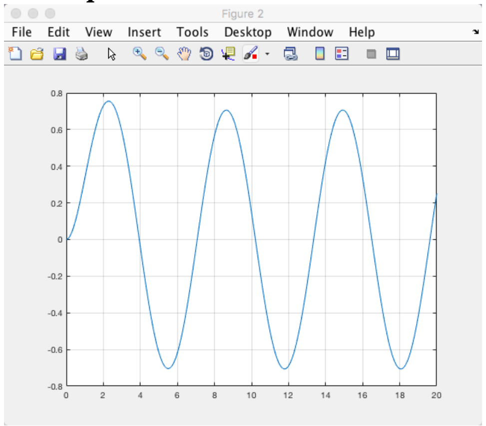
Frequency Response - Definition



$$u(t) = U \cdot \ sin\omega t$$

$$y(t) = \underbrace{UA}_{Y} \ sin(\omega t + \phi)$$





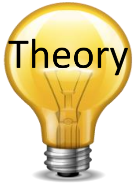
$$\omega = 1\, rad/s$$

$$\omega = 1\, rad/s$$

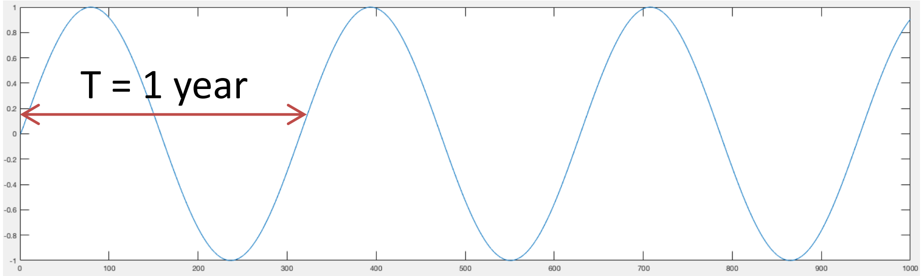
and the same for Frequency 2, 3, 4, 5, 6, etc.

- The frequency response of a system is defined as the **<u>steady-state</u> response** of the system to a **sinusoidal** input signal.
- When the system is in steady-state, it differs from the input signal only in **amplitude/gain** (A) (Norwegian: “forsterkning”) and **phase lag** (φ) (Norwegian: “faseforskyvning”).

Frequency Response - Simple Example



Outside Temperature



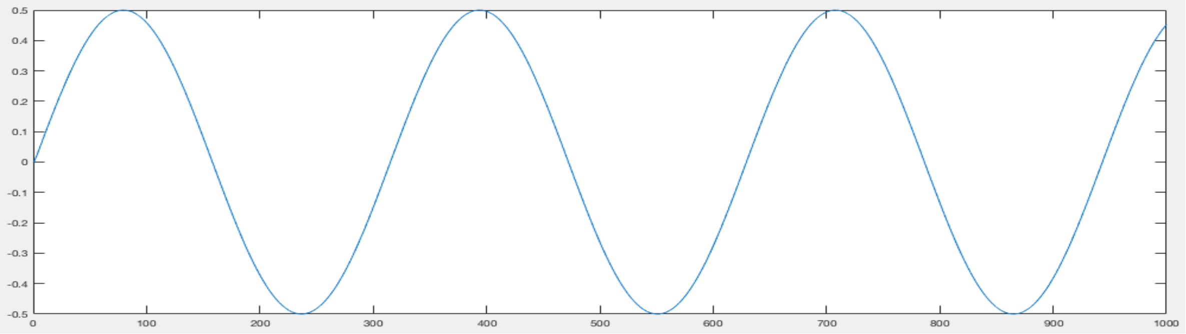
frequency 1 (year)

Dynamic System

frequency 1 (year)

Inside Temperature

Note! Only the gain and phase are different

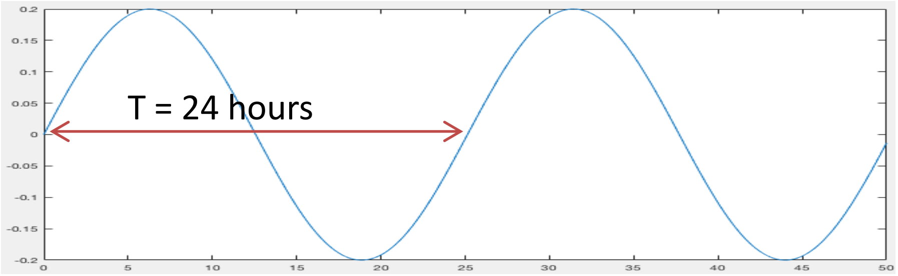


Assume the outdoor temperature is varying like a sine function during a year (frequency 1) or during 24 hours (frequency 2). Then the indoor temperature will be a sine as well, but with different gain. In addition it will have a phase lag.

Frequency Response - Simple Example

Theory

<u>Outside</u> Temperature

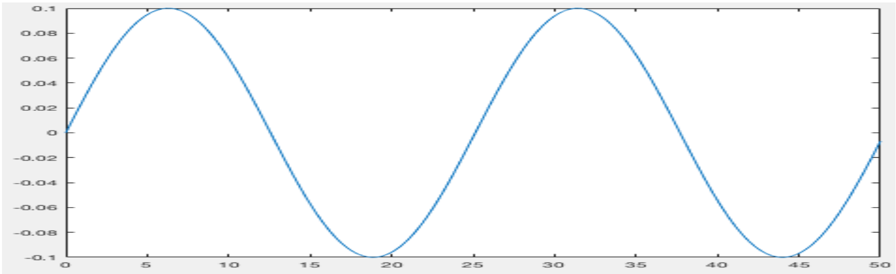


frequency 2 (24 hours)

Dynamic System

frequency 2 (24 hours)

<u>Inside</u> Temperature



Note! Only the gain and phase are different

Assume the outdoor temperature is varying like a sine function during a year (frequency 1) or during 24 hours (frequency 2). Then the indoor temperature will be a sine as well, but with different gain. In addition it will have a phase lag.

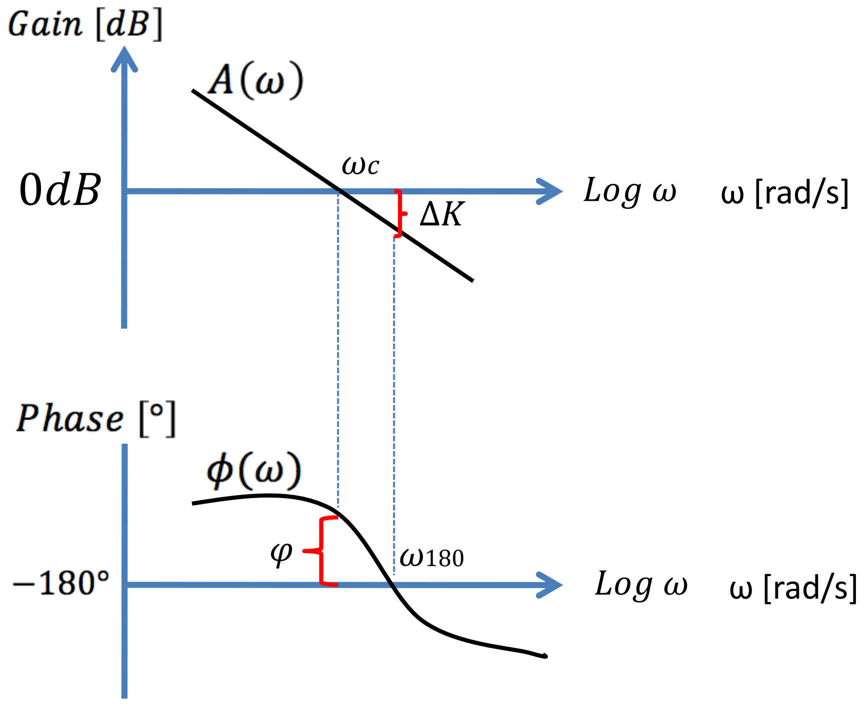
https://www.halvorsen.blog

Frequency Response using Bode Diagram

Hans-Petter Halvorsen

Bode Diagram

You can find the Bode diagram from experiments on the physical process or from the transfer function (the model of the system). A simple sketch of the Bode diagram for a given system:

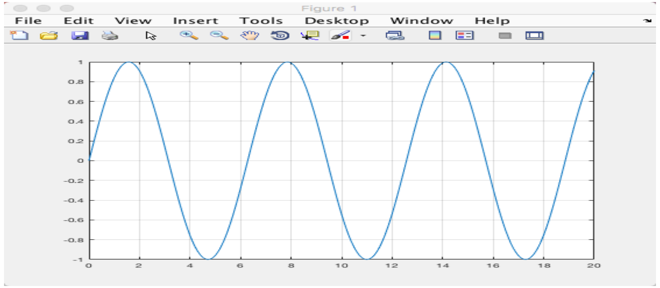


The Bode diagram gives a simple Graphical overview of the Frequency Response for a given system. A Tool for Analyzing the Stability properties of the Control System.

With MATLAB you can easily create Bode diagram from the Transfer function model using the bode() function

Bode Diagram from experiments

1. Find Data for different frequencies

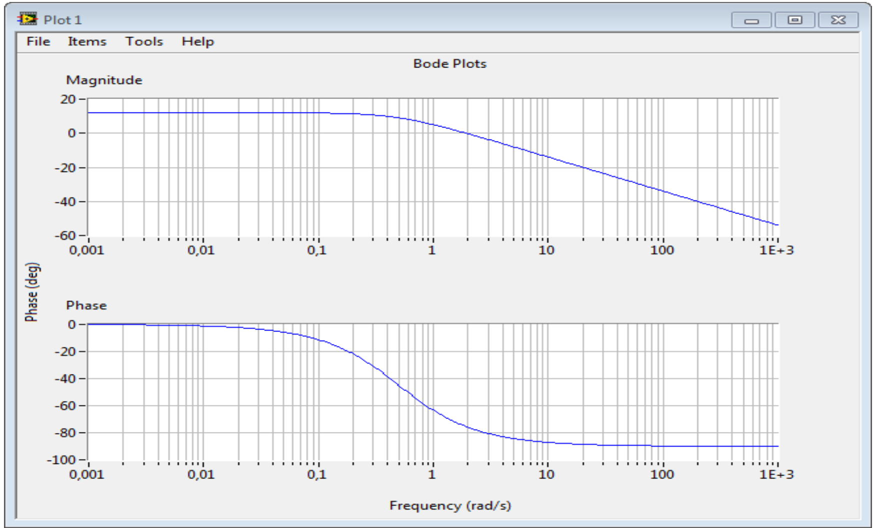


...

2. We find A and ϕ for each of the frequencies, e.g.:

ω	$A(\omega)$	$\phi(\omega)$
0.1	11.9	-11.3
0.16	11.6	-17.7
0.25	11.1	-26.5
0.4	9.9	-38.7
0.625	7.8	-51.3
2.5	-2.1	-78.6

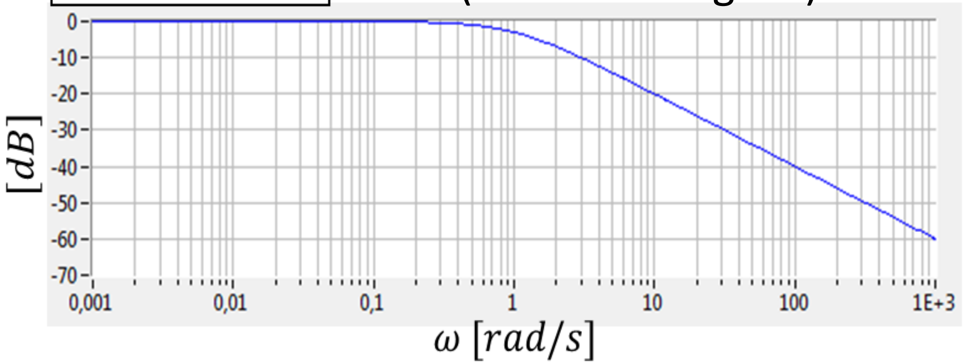
3. Based on that we can plot the Frequency Response in a so-called Bode Diagram:



Bode Diagram

The x-scale is **logarithmic**

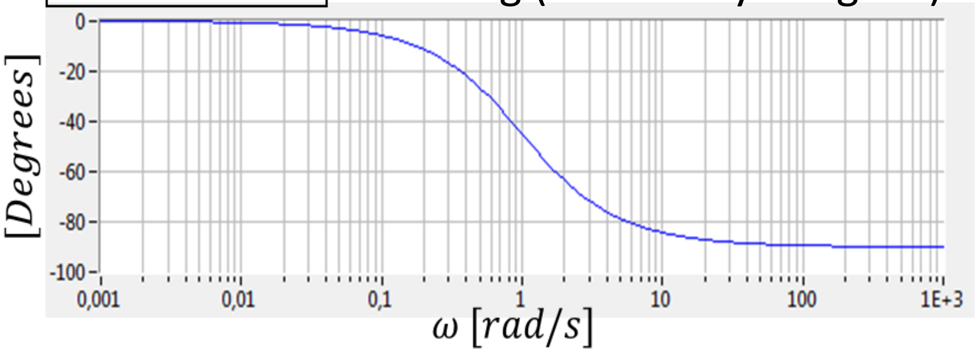
$\boxed{A(\omega) = |H(j\omega)|}$ Gain ("Forsterkningen")



Note! The y-scale is in $[dB]$

$$x\ [dB] = 20 log_{10} x$$

$\boxed{\phi(\omega) = \angle H(j\omega)}$ Phase lag ("Faseforkyvningen")



The y-scale is in $[degrees]$

$$2\pi\ rad = 360°$$

$$\boxed{d\ [degrees] = r[radians] \cdot \left(\frac{180}{\pi}\right)}$$

$$\boxed{r[radians] = d[degrees] \cdot \left(\frac{\pi}{180}\right)}$$

Normally, the unit for frequency is Hertz [Hz], but in frequency response and Bode diagrams we use radians ω [rad/s]. The relationship between these are as follows:

$$\omega = 2\pi f$$

Frequency Response – MATLAB

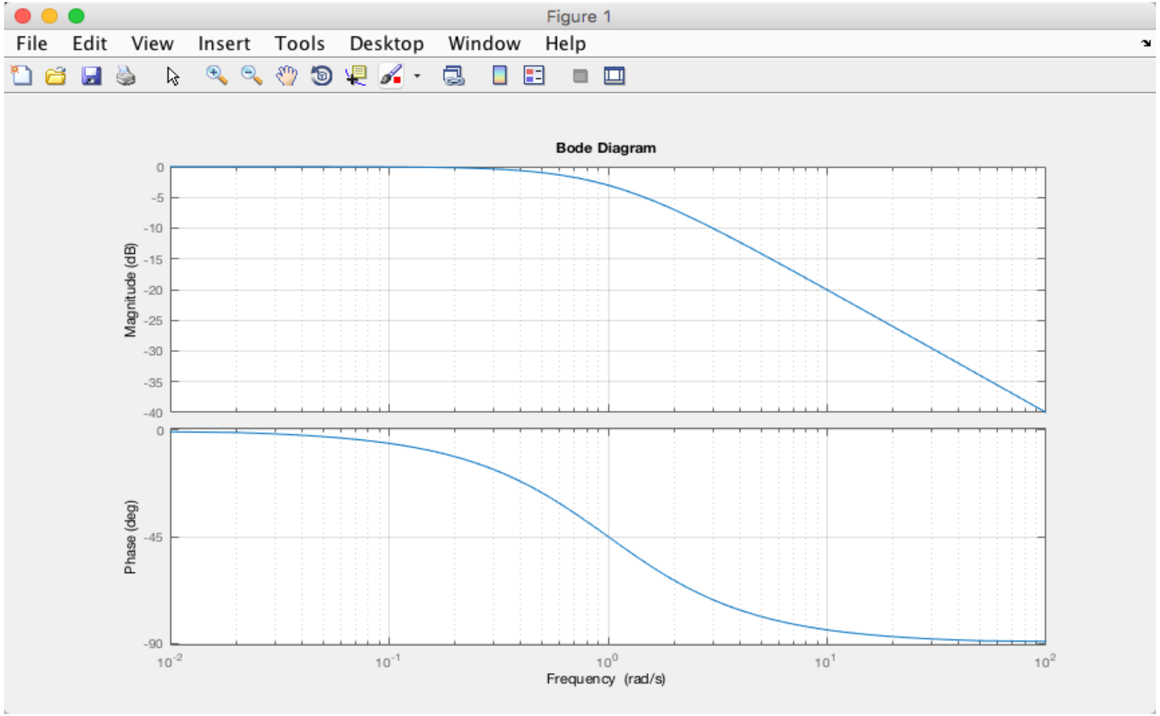
Example

Transfer Function:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{s+1}$$

MATLAB Code:

```
clear
clc
close all

% Define Transfer function
num=[1];
den=[1, 1];
H = tf(num, den)

% Frequency Response
bode(H);
grid on
```



The frequency response is an important tool for analysis and design of signal filters and for analysis and design of control systems.

Frequency Response – MATLAB

Example

Transfer Function:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{s+1}$$

Instead of Plotting the Bode Diagram we can also use the bode function for calculation and showing the data as well:

```
freq_data =

    0.0100   -0.0004   -0.5729
    0.1000   -0.0432   -5.7106
    1.0000   -3.0103   -45.0000
    2.0000   -6.9897   -63.4349
    3.0000   -10.0000  -71.5651
    5.0000   -14.1497  -78.6901
    10.0000  -20.0432  -84.2894
    100.0000 -40.0004  -89.4271
```

```
clear
clc
close all

% Define Transfer function
num = [1];
den = [1, 1];
H = tf(num, den)

% Frequency Response
bode(H);
grid on

% Get Freqquency Response Data
wlist = [0.01, 0.1, 1, 2 ,3 ,5 ,10, 100];
[mag, phase, w] = bode(H, wlist);

for i=1:length(w)
  magw(i) = mag(1,1,i);
  phasew(i) = phase(1,1,i);
end

magdB = 20*log10(magw); % Convert to dB

freq_data = [wlist; magdB; phasew]'
```

Bode Diagram – MATLAB Example

Example

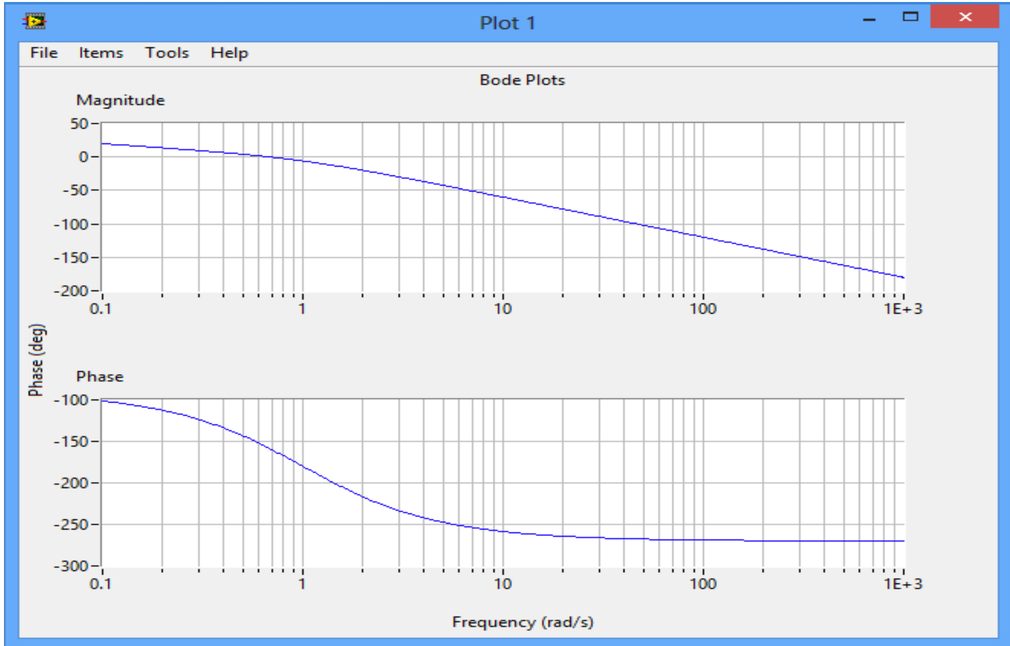
$$H(s) = \frac{1}{s(s+1)^2} = \frac{1}{s^3 + 2s^2 + s}$$

MATLAB Code:

```
clear, clc

% Transfer function
num=[1];
den1=[1,0];
den2=[1,1]
den3=[1,1]
den = conv(den1,conv(den2,den3));
H = tf(num, den)

% Bode Diagram
bode(H)
subplot(2,1,1)
grid on
subplot(2,1,2)
grid on
```

or:

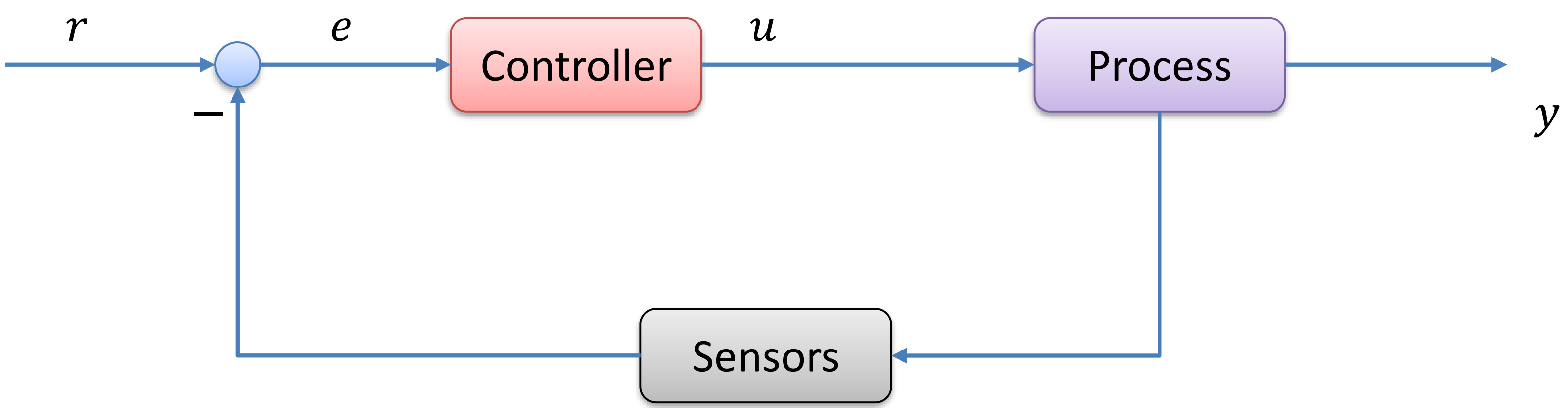
```
clear, clc

% Transfer function
num=[1];
den=[1,2,1,0];
H = tf(num, den)

% Bode Diagram
bode(H)
subplot(2,1,1)
grid on
subplot(2,1,2)
grid on
```



Example

We will use the following system as an example:

$$H_c = K_p \qquad H_p = \frac{1}{s(s+1)}$$



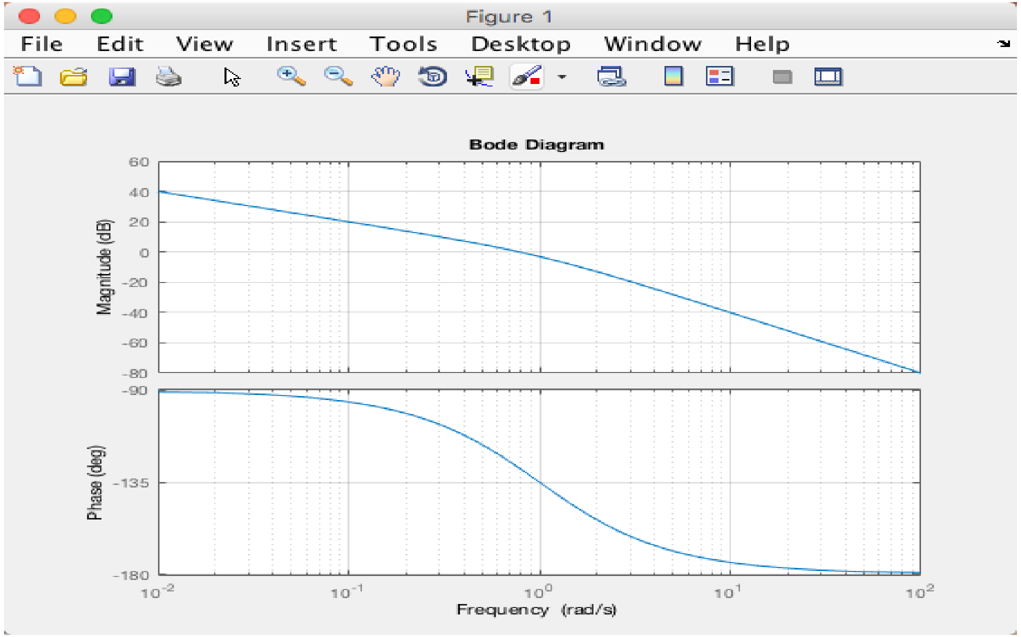
$$H_m = \frac{1}{3s+1}$$

Bode Diagram – MATLAB Example

MATLAB Code:

```
clear
clc

num = 1;
den = [1,1,0];

Hp = tf(num,den)

bode(Hp)
grid on
```

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



https://www.halvorsen.blog

Complex Numbers

Background Theory for Frequency Response

Hans-Petter Halvorsen

Complex Numbers

A Complex Number is given by:

$$z = a + jb$$

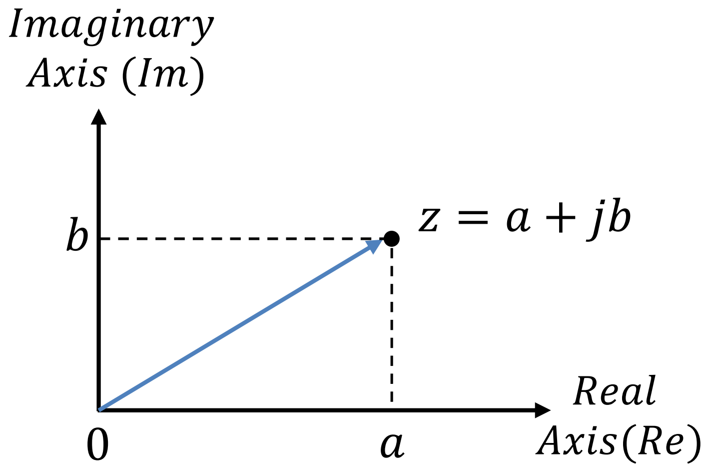
Where

$$j = \sqrt{-1}$$

We have that:

$$a = Re(z)$$

$$b = Im(z)$$



Complex Numbers

$j = \sqrt{-1}$

Polar form:

$$z = re^{j\theta}$$

Where:

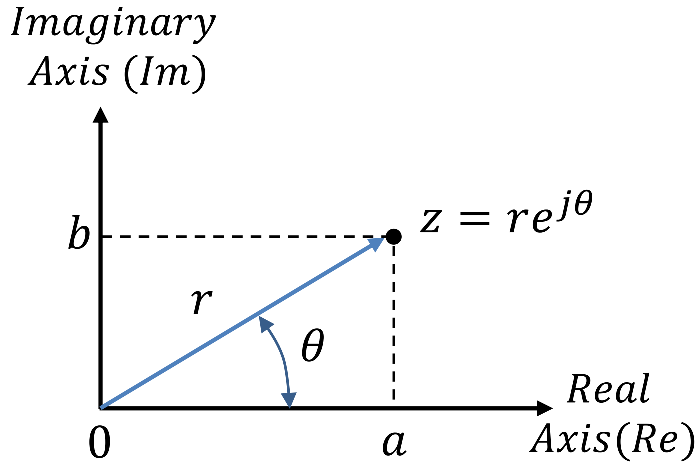
$$r = |z| = \sqrt{a^2 + b^2}$$

$$\theta = atan\frac{b}{a}$$

Note!

$a = r\cos\theta$

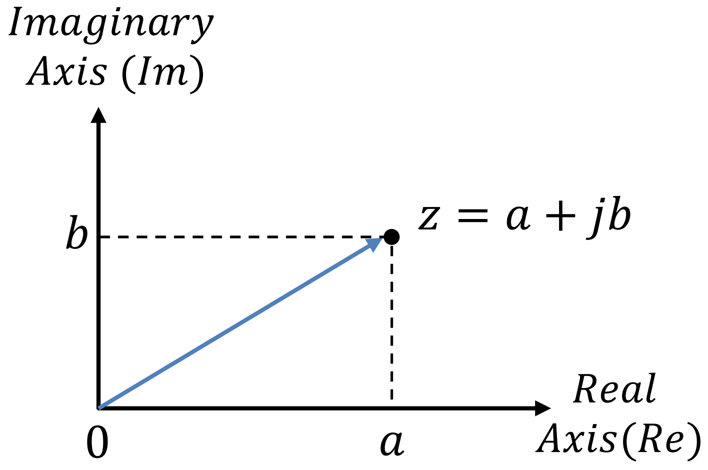
$b = r\sin\theta$



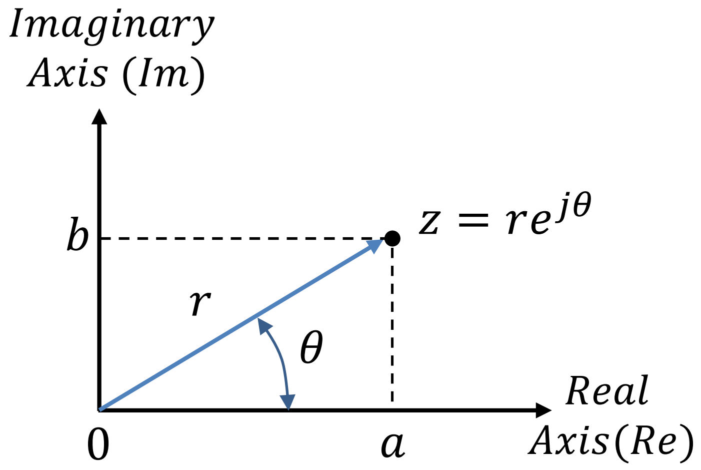
Complex Numbers

$j = \sqrt{-1}$

1 <u>Rectangular form</u> of a complex number



2 <u>Exponential/polar form</u> of a complex number



Length ("Gain"):

$$r = |z| = \sqrt{a^2 + b^2}$$

Angle ("Phase"):

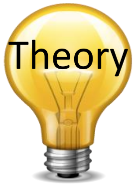
$$\theta = atan\frac{b}{a}$$

https://www.halvorsen.blog

Frequency response from Transfer function

Hans-Petter Halvorsen

Manually find the Frequency Response from the Transfer Function



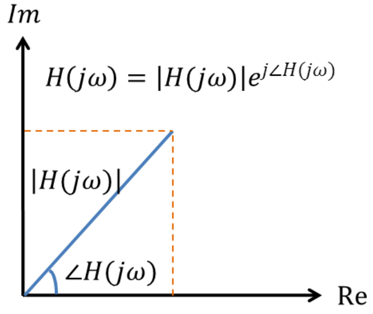
For a transfer function:

$$H(S) = \frac{y(s)}{u(s)}$$

$s = j\omega$

We have that:

$$H(j\omega) = |H(j\omega)|e^{j\angle H(j\omega)}$$



Where $H(j\omega)$ is the frequency response of the system, i.e., we may find the frequency response by setting $s = j\omega$ in the transfer function. Bode diagrams are useful in frequency response analysis.

The Bode diagram consists of 2 diagrams, the Bode magnitude diagram, $A(\omega)$ and the Bode phase diagram, $\phi(\omega)$.

The **Gain** function:

$$A(\omega) = |H(j\omega)|$$

The **Phase** function:

$$\phi(\omega) = \angle H(j\omega)$$

The $A(\omega)$-axis is in decibel (dB), where the decibel value of x is calculated as: $x[dB] = 20log_{10}x$

The $\phi(\omega)$-axis is in degrees (not radians!)

Mathematical expressions for $A(\omega)$ and $\phi(\omega)$

We find the Mathematical expressions for $A(\omega)$ and $\phi(\omega)$ by setting $s = j\omega$ in the transfer function given by:

$$H(s) = \frac{y(s)}{u(s)} = \frac{K}{Ts + 1}$$

The Frequency Response (we replace s with $j\omega$) then becomes:

$$H(j\omega) = \frac{K}{Tj\omega + 1} = \frac{K}{\underbrace{1}_{Re} + j\underbrace{T\omega}_{Im}}$$

Polar form:

$$H(j\omega) = \frac{K}{\sqrt{1^2 + (T\omega)^2}e^{j\,arctan\left(\frac{T\omega}{1}\right)}}$$

$$= \frac{K}{\sqrt{1 + (T\omega)^2}} e^{j[-arctan(T\omega)]}$$

Finally:

$$H(j\omega) = \frac{K}{\sqrt{1 + (T\omega)^2}} e^{j[-arctan(T\omega)]}$$

cont. next page

cont. from previous page

The Gain function becomes:

$$A(\omega) = |H(j\omega)| = \frac{K}{\sqrt{1 + (T\omega)^2}}$$

Or in $[dB]$ (used in the Bode Plot):

$$A(\omega)_{dB} = |H(j\omega)|_{dB} = 20logK - 20log\sqrt{1 + (T\omega)^2}$$

The Phase function becomes ($[rad]$):

$$\phi(\omega) = \angle H(j\omega) = arg\ \ H(j\omega) = -arctan(T\omega)$$

Or in degrees[°] (used in the Bode plot):

$$\phi(\omega) = \angle H(j\omega) = -arctan(T\omega) \cdot \frac{180}{\pi}$$

Note: $2\pi\ rad = 360°$

Example

Transfer function:	$A(\omega)$ og $\phi(\omega)$:
$H(s) = \frac{1}{s+1}$	$\lvert H(j\omega)\rvert_{dB} = 20log1 - 20log\sqrt{(\omega)^2+1}$ $\angle H(j\omega) = -\arctan(\omega)$
$H(s) = \frac{4}{2s+1}$	$\lvert H(j\omega)\rvert_{dB} = 20log4 - 20log\sqrt{(2\omega)^2+1}$ $\angle H(j\omega) = -\arctan(2\omega)$
$H(S) = \frac{5}{(s+1)(10s+1)}$	$\lvert H(j\omega)\rvert_{dB} = 20log5 - 20log\sqrt{(\omega)^2+1} - 20log\sqrt{(10\omega)^2+1}$ $\angle H(j\omega) = -\arctan(\omega) - \arctan(10\omega)$
$H(S) = \frac{1}{s(s+1)^2}$	$\lvert H(j\omega)\rvert_{dB} = -20log\sqrt{(\omega)^2} - 2x20log\sqrt{(\omega)^2+1}$ $= 20log\omega - 40log\sqrt{(\omega)^2+1}$ $\angle H(j\omega) = -90 - 2\arctan(\omega)$
$H(s) = \frac{3.2e^{-2s}}{3s+1}$	$\lvert H(j\omega)\rvert_{dB} = 20log3.2 - 20log\sqrt{(3\omega)^2+1}$ $\angle H(j\omega) = -2\omega - \arctan(3\omega)$
$H(S) = \frac{5s+1}{(2s+1)(10s+1)}$	$\lvert H(j\omega)\rvert_{dB} = 20log\sqrt{(5\omega)^2+1} - 20log\sqrt{(2\omega)^2+1} - 20log\sqrt{(10\omega)^2+1}$ $\angle H(j\omega) = \arctan(5\omega) - \arctan(2\omega) - \arctan(10\omega)$

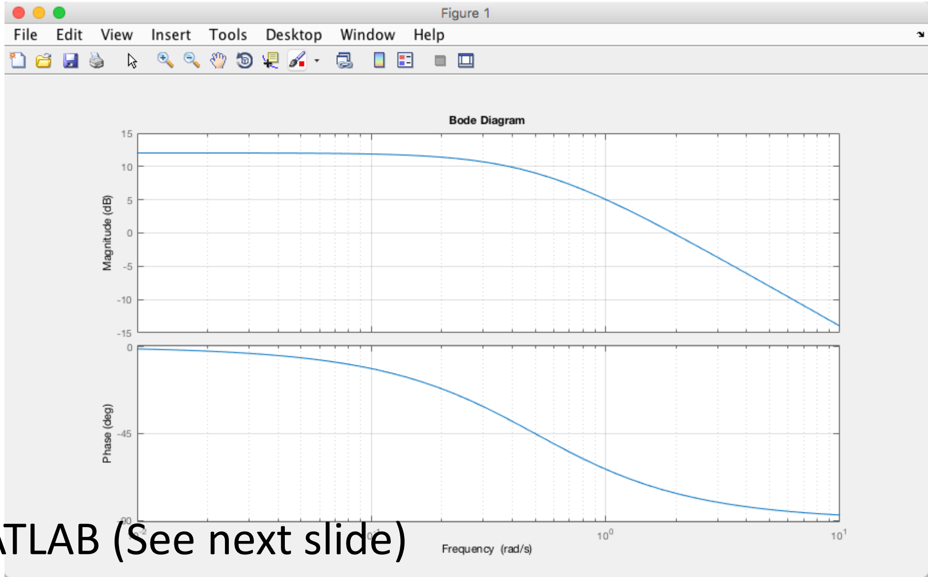
Note! In order to find the phase in degrees, we have to multiply with: $\frac{180}{\pi}$

Manually find the Frequency Response from the Transfer Function

Given the following transfer function:

$$H(S) = \frac{4}{2s+1}$$

Bode Plot:



The mathematical expressions for $A(\omega)$ and $\phi(\omega)$ become:

$$|H(j\omega)|_{dB} = \underline{20log4 - 20log\sqrt{(2\omega)^2+1}}$$

$$\angle H(j\omega) = \underline{-\arctan(2\omega)}$$

These equations can easily be implemented in MATLAB (See next slide)

MATLAB Code

```
clear
clc

% Transfer function
num=[4];
den=[2, 1];
H = tf(num, den)

% Bode Plot
figure(1)
bode(H)
grid on

% Margins and Phases for given Frequencies

% Alt 1: Use bode function directly
disp('----- Alternative 1 -----')
w = [0.1, 0.16, 0.25, 0.4, 0.625, 2.5, 10];

[magw, phasew] = bode(H, w);

for i=1:length(w)
  mag(i) = magw(1,1,i);
  phase(i) = phasew(1,1,i);
end

magdB = 20*log10(mag); %convert to dB
mag_data = [w; magdB]
phase_data = [w; phase]
```

```
clear
clc

w = [0.1, 0.16, 0.25, 0.4, 0.625, 2.5, 10];

% Alt 2: Use Mathematical expressions for H and <H
disp('----- Alternative 2 -----')
gain = 20*log10(4) - 20*log10(sqrt((2*w).^2+1));
phase = -atan(2*w);
phasedeg = phase * 180/pi; %convert to degrees

mag_data2 = [w; gain]
phase_data2 = [w; phasedeg]

figure(2)
subplot(2,1,1)
semilogx(w,gain)
grid on

subplot(2,1,2)
semilogx(w,phasedeg)
grid on
```

Transfer functions with Time delay

Transfer functions with Time delay

A general transfer function for a 1.order system with time delay is:

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

Frequency Response functions for gain and phase margin becomes:

$$A(\omega)[dB] = 20log(K) - 20log\sqrt{(T\omega)^2+1}$$
$$\phi(\omega) = -arctan(T\omega) - \omega \cdot \tau$$

Or $\phi(\omega)$ in degrees:

$$\phi(\omega)[degrees] = [-\mathrm{arctan}(T\omega) - \omega \cdot \tau]\frac{180}{\pi}$$

Transfer functions with Time delay in MATLAB

Different ways to implement a time delay in MATLAB:

Alt 1

```
K = 3.5;
T = 22;
Tau = 2;

num = [K];
den = [T, 1];
H1 = tf(num, den);

s = tf('s')
H2 = exp(-Tau*s);

H = H1 * H2

bode(H)
```

Alt 2

```
K = 3.5;
T = 22;
Tau = 2;

num = [K];
den = [T, 1];
H1 = tf(num, den);

H = set(H1,'inputdelay',Tau)

bode(H);
```

Alt 3

```
K = 3.5;
T = 22;
Tau = 2;

s = tf('s');
H = K*exp(-Tau*s)/(T*s+1)

bode(H);
```

Alt 4: Use Pade approximation

```
K = 3.5;
T = 22;
Tau = 2;

num = [K];
den = [T, 1];
H1 = tf(num, den);

N=5;
H2 = pade(Tau, N)

[num_pade,den_pade] = pade(T,N)
Hpade = tf(num_pade,den_pade);

H = series(H1, Hpade);

bode(H);
```

1. order system with Time delay

Example

Given the following transfer function:

$$H(s) = \frac{3.2e^{-2s}}{3s+1}$$

Poles:

$$p_1 = -\frac{1}{3} = -0.33$$

Zeros:

None

The mathematical expressions for $A(\omega)$ and $\phi(\omega)$:

$$|H(j\omega)|_{dB} = \underline{20log3.2 - 20log\sqrt{(3\omega)^2+1}}$$

$$\angle H(j\omega) = \underline{-2\omega - \arctan(3\omega)}$$

Or in degrees:$\angle H(j\omega) = (-2\omega - \arctan(3\omega)) \cdot \frac{180}{\pi}$

Break frequency:

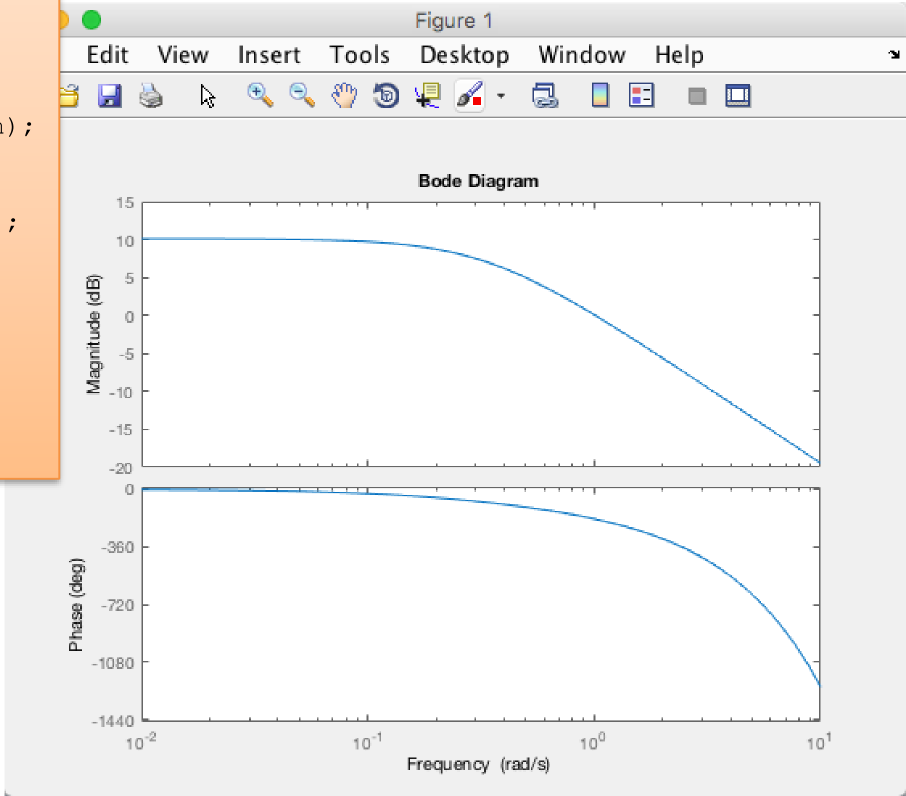
$$\omega = \frac{1}{T} = \frac{1}{3} = 0.33\, rad/s$$

```
clear, clc

s = tf('s');
K=3.2;
T=3;

H1=tf(K/(T*s+1));

delay = 2;
H = set(H1,'inputdelay',delay)
bode(H);

p = pole(H)
z = zero (H)
```

```
H =

                   3.2
  exp(-2*s) * -------
               3 s + 1

Continuous-time transfer function.

p =   -0.3333


z =   Empty matrix: 0-by-1
```

or:

```
clear, clc

s=tf('s');

K=3.2;
T=3;
Tau = 2;

num = K;
den = [T, 1];
H1 = tf(num, den);

s = tf('s')
H2 = exp(-Tau*s);

H = H1 * H2

bode(H);

p = pole(H)
z = zero (H)
```

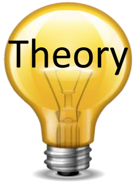


https://www.halvorsen.blog

Frequency response from Input/Output Signals

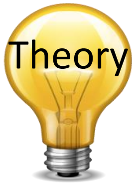
Hans-Petter Halvorsen

Frequency Response from sinusoidal input and output signals



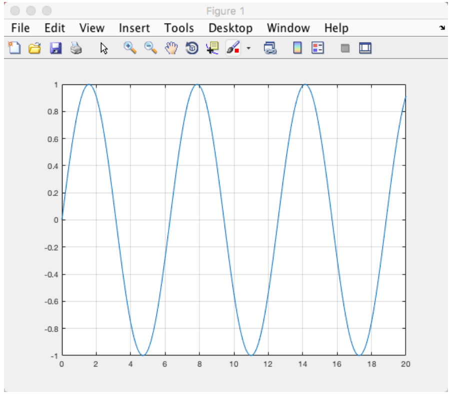
We can find the frequency response of a system by exciting the system (either the real system or a model of the system) with a sinusoidal signal of amplitude A and frequency ω $[rad/s]$ (Note: $\omega = 2\pi f$) and observing the response in the output variable of the system.

Frequency Response - Definition

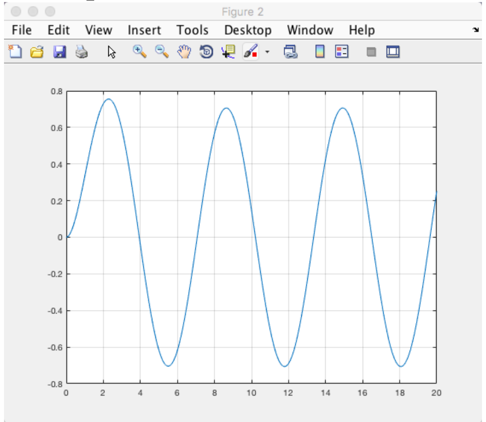


$$u(t) = U \cdot \ sin\omega t$$

$$y(t) = \underbrace{UA}_{Y}\ sin(\omega t + \phi)$$



$$\omega = 1\,rad/s$$



$$\omega = 1\,rad/s$$

and the same for Frequency 2, 3, 4, 5, 6, etc.

- The frequency response of a system is defined as the **steady-state response** of the system to a **sinusoidal** input signal.
- When the system is in steady-state, it differs from the input signal only in **amplitude/gain** (A) (Norwegian: “forsterkning”) and **phase lag** (φ) (Norwegian: “faseforskyvning”).

Frequency Response from sinusoidal input and output signals

Theory

The input signal is given by:

$$u(t) = U \cdot \ sin\omega t$$

The steady-state output signal will then be:

$$y(t) = \underbrace{UA}_{Y} \ sin(\omega t + \phi)$$

The gain is given by:

$$A = \frac{Y}{U}$$

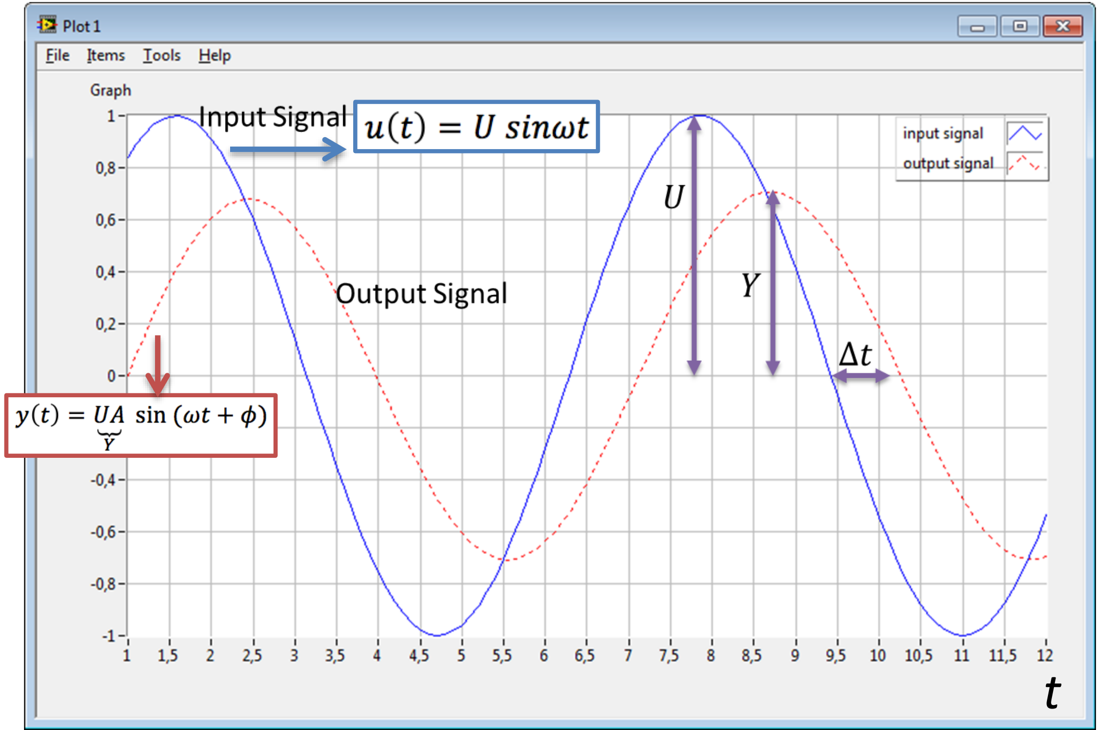
The phase lag is given by:

$$\phi = -\omega \Delta t \ [rad]$$

Frequency Response from sinusoidal input and output signals

Theory

You will get plots like this for <u>each</u> frequency:



The gain is given by:

$$A = \frac{Y}{U}$$

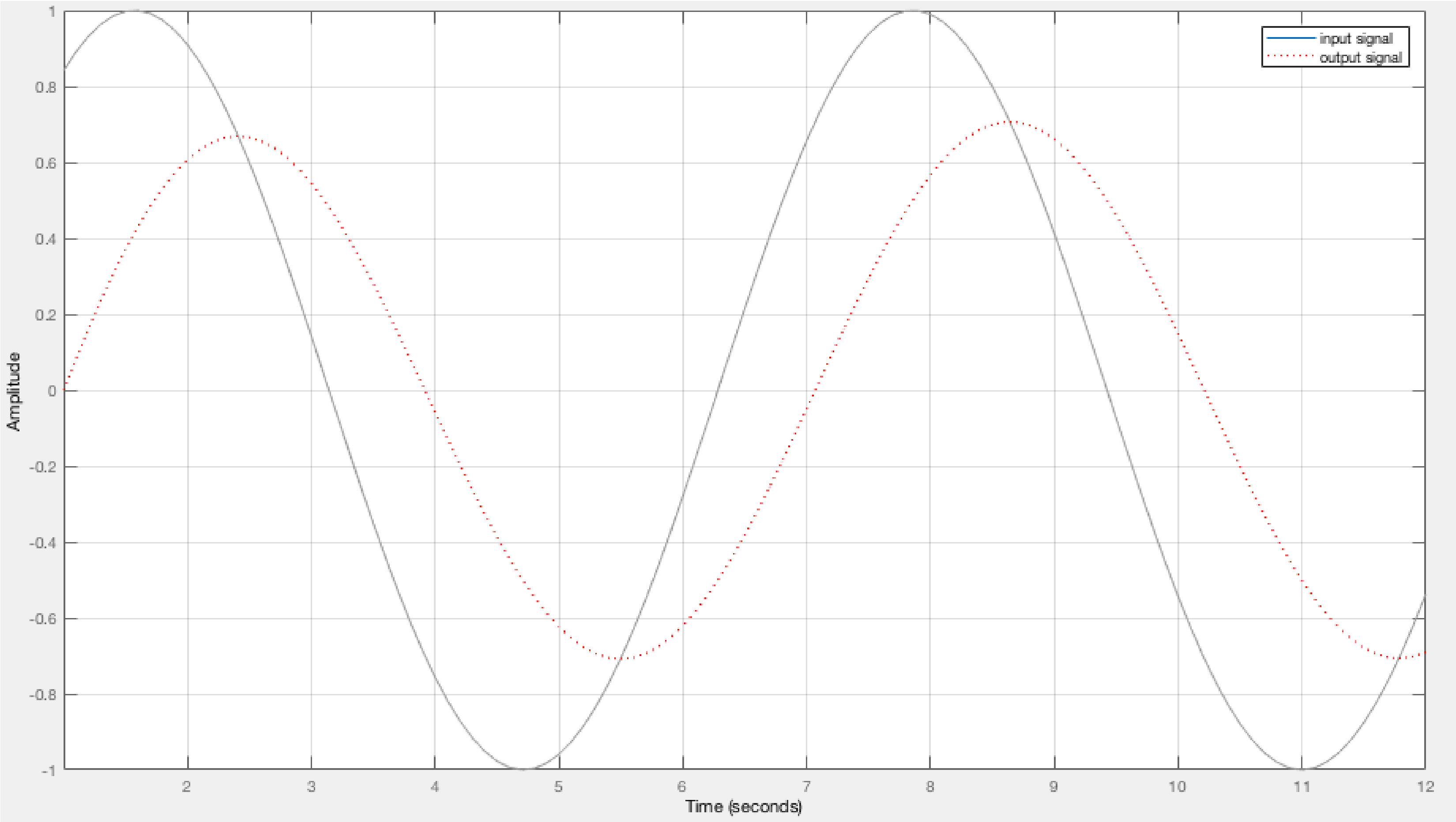
The phase lag is given by:

$$\phi = -\omega \Delta t \; [rad]$$

Find the gain (A) and the phase (ϕ) for the given frequency from the plot

Example

$\omega = 1\ rad/s$

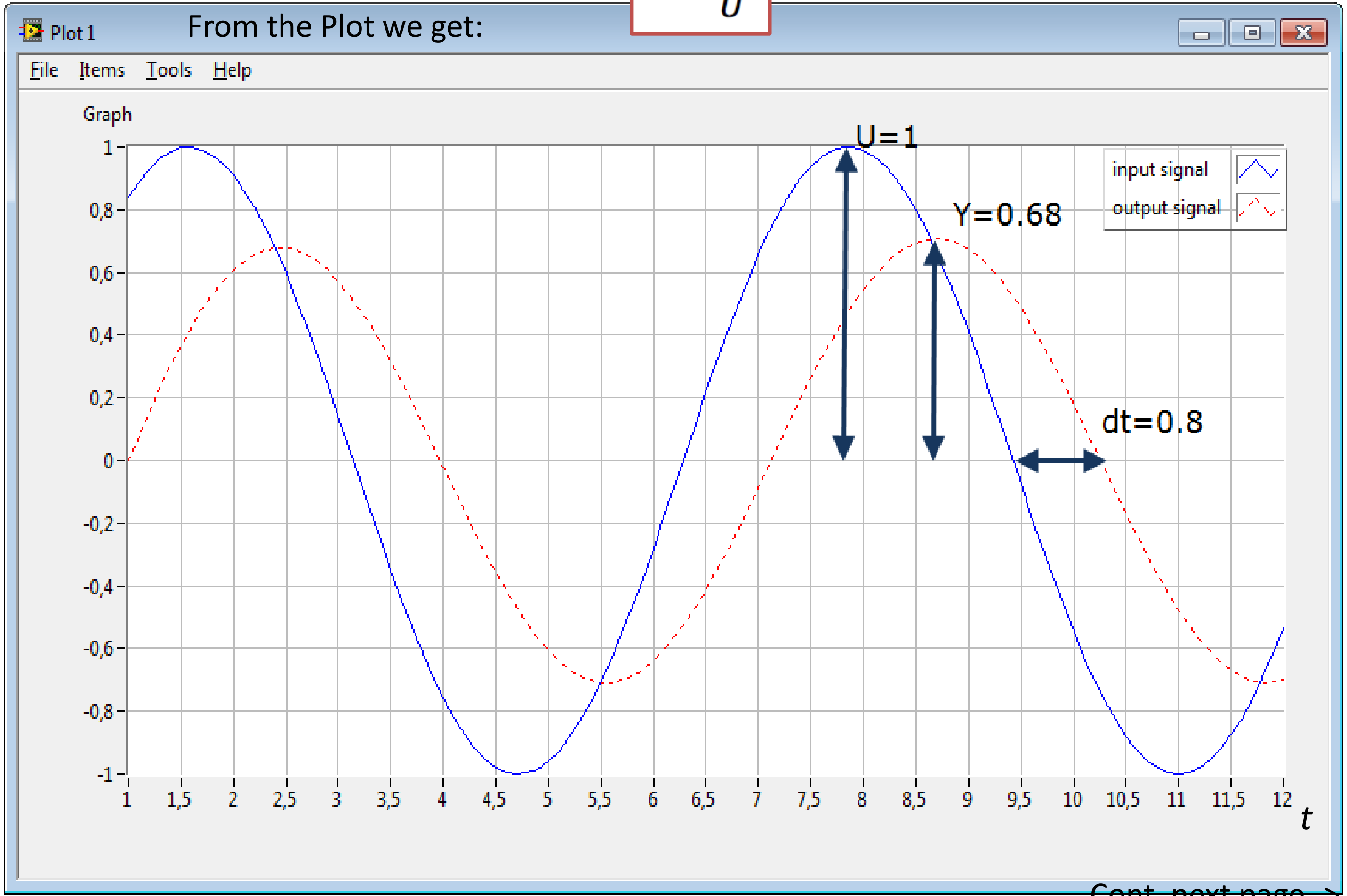


Solutions

$$A = \frac{Y}{U}$$

$$\phi = -\omega \Delta t \; [rad]$$

From the Plot we get:



Cont. next page ->

Solutions

This gives the following:

1 **Amplitude gain:**

$$A = \frac{Y}{U} = \frac{0.68}{1} = \underline{0.68}$$

Or in dB:

$$A\ [dB] = 20 \log 0.68 = \underline{-3.35\ dB}$$

2 **Phase lag:**

$$\phi = -\omega \Delta t = -1 \cdot 0.8 = \underline{-0.8\ rad}$$

Or in degrees ($2\pi\ [rad] = 360°$):

$$\phi\ [degrees] = \frac{180}{\pi} \cdot (-0.8) = \underline{-45.9°}$$

Conversion to dB

$$A\ [dB] = 20log(A)$$ or the other way: $$A = 10^{\frac{A[dB]}{20}}$$

Example:

$$A = 0.68$$
$$A\ [dB] = 20log(A)=20log(0.68) \approx -3.35dB$$

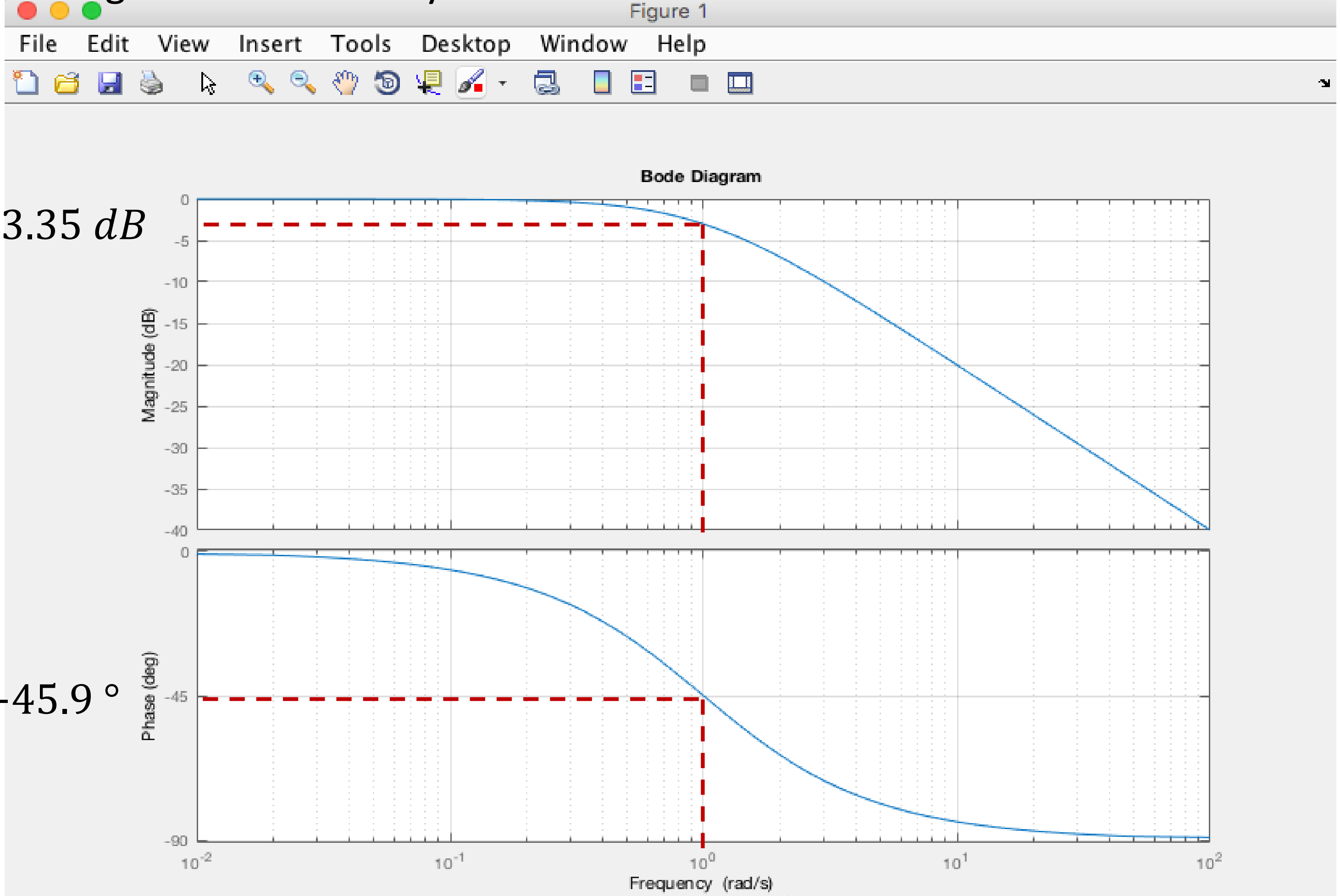
Or the other way:

$$A[dB] = -3.35dB$$
$$A\ [dB] = 10^{\frac{-3.35}{20}} \approx 0.68$$

From the Bode diagram we can verify that our calculations are correct:

$A = -3.35\ dB$

$\phi = -45.9\ °$



$\omega = 1\ rad/s$

The following MATLAB Code is used to create the Plot:

```
clear, clc

K = 1;
T = 1;
num = [K];
den = [T, 1];
H = tf(num, den);

figure(1)
bode(H), grid on

% Define input signal
t = [1: 0.1 : 12];
w = 1;
U = 1;
u = U*sin(w*t);
figure(2)
plot(t, u)

% Output signal
hold on
lsim(H, ':r', u, t)
grid on
hold off
legend('input signal', 'output signal')
```

We use the following transfer function:

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

The Frequency used:

$$\omega = 1\ rad/s$$

https://www.halvorsen.blog

PID Controller Design/Tuning

Hans-Petter Halvorsen

PID Controller Design

A lot of PID Tuning methods exist, e.g.,

- Skogestad's method
- Ziegler-Nichols’ methods
- Trial and Error Methods
- PID Tuning functionality built into MATLAB
- Auto-tuning built into commercial PID controllers
- …

Skogestad's method

- The Skogestad's method assumes you apply a step on the input (u) and then observe the response and the output (y), as shown below.
- If we have a model of the system (which we have in our case), we can use the following Skogestad's formulas for finding the PI(D) parameters directly.

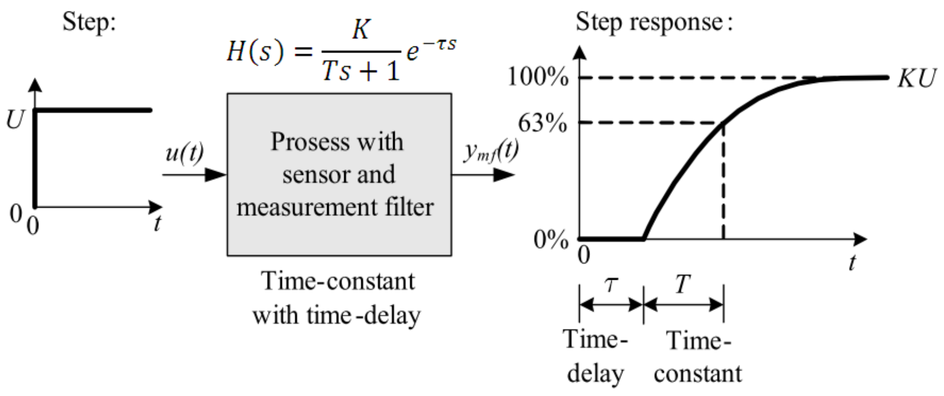


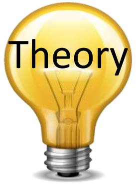
Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010.

Process type	$H_{psf}(s)$ (process)	K_p	T_i	T_d
Integrator + delay	$\frac{K}{s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	0
Time-constant + delay	$\frac{K}{Ts+1}e^{-\tau s}$	$\frac{T}{K(T_C+\tau)}$	$\min\,[T,\, c\,(T_C+\tau)]$	0
Integr + time-const + del.	$\frac{K}{(Ts+1)s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	T
Two time-const + delay	$\frac{K}{(T_1s+1)(T_2s+1)}e^{-\tau s}$	$\frac{T_1}{K(T_C+\tau)}$	$\min\,[T_1,\, c\,(T_C+\tau)]$	T_2
Double integrator + delay	$\frac{K}{s^2}e^{-\tau s}$	$\frac{1}{4K(T_C+\tau)^2}$	$4\,(T_C+\tau)$	$4\,(T_C+\tau)$

T_c is the time-constant of the control system which the user must specify

Originally, Skogestad defined the factor $c = 4$. This gives good set-point tracking. But the disturbance compensation may become quite sluggish. To obtain faster disturbance compensation, you can use $c = 1.5$

Ziegler–Nichols Frequency Response method



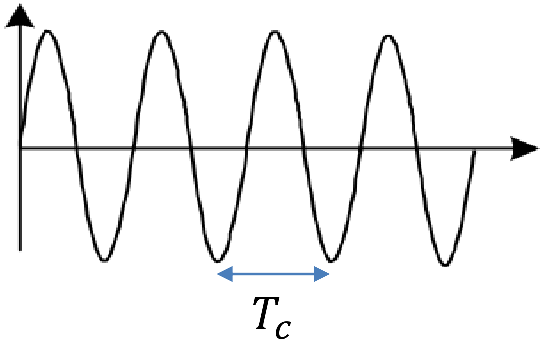
Assume you use a P controller only $T_i = \infty, T_d = 0$. Then you need to find for which K_p the closed loop system is a marginally stable system ($\omega_c = \omega_{180}$). This K_p is called K_c (critical gain). The T_c (critical period) can be found from the damped oscillations of the closed loop system. Then calculate the PI(D) parameters using the formulas below.

Controller	K_p	T_i	T_d
P	$0.5K_c$	∞	0
PI	$0.45K_c$	$\frac{T_c}{1.2}$	0
PID	$0.6K_c$	$\frac{T_c}{2}$	$\frac{T_c}{8}$

K_c- Critical Gain

T_c- Critical Period

Marginally stable system:



$$\omega_c = \omega_{180}$$

$$0 < \lim_{t\to\infty} y(t) < \infty$$

$$T_c = \frac{2\pi}{\omega_{180}}$$

https://en.wikipedia.org/wiki/Ziegler–Nichols_method

https://www.halvorsen.blog

Controller Design/Tuning using MATLAB

Hans-Petter Halvorsen

Controller Design/Tuning using MATLAB

- Frequency Design and Analysis
- *pidtune()* MATLAB function
- PID Tuner (Interactive Tools)
- …

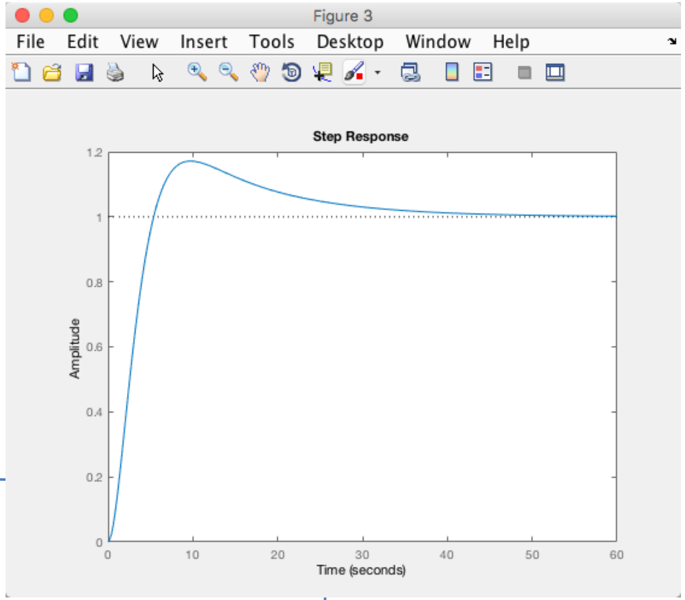
Validate with simulations!

pidtune() MATLAB function

Example

```
clear, clc

%Define Process
num = 1;
den = [1,1,0];
Hp = tf(num,den)

% Find PI Controller
[Hpi,info] = pidtune(Hp,'PI')

%Bode Plots
figure(1)
bode(Hp)
grid on

figure(2)
bode(Hpi)
grid on

% Feedback System
T = feedback(Hpi*Hp, 1);
figure(3)
step(T)
```

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

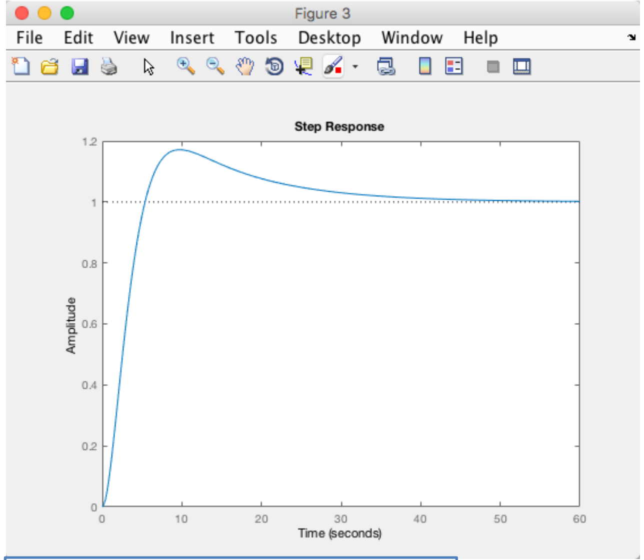



```
Hpi =

             1
  Kp + Ki * ---
             s

  with Kp = 0.333, Ki = 0.023

Continuous-time PI controller in parallel form.


info =

                Stable: 1
    CrossoverFrequency: 0.3237
           PhaseMargin: 60.0000
```

$$K_p = 0.33$$

$$T_i = \frac{1}{K_i} = \frac{1}{0.023} \approx 43.5s$$

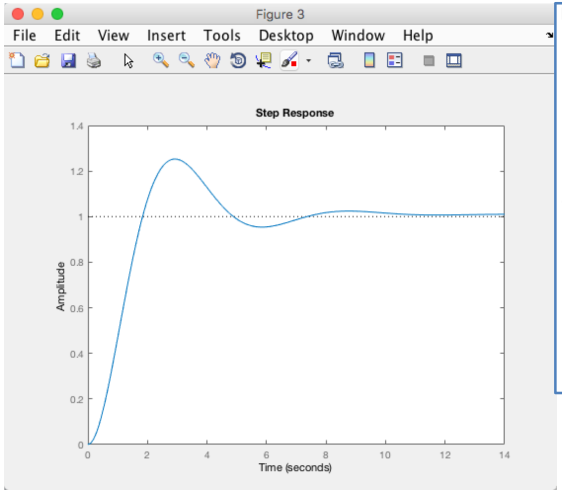
pidtune() MATLAB function




```
Hpi =

             1
  Kp + Ki * ---
             s

  with Kp = 0.333, Ki = 0.023

Continuous-time PI controller in parallel form.


info =

                   Stable: 1
       CrossoverFrequency: 0.3237
              PhaseMargin: 60.0000
```

To improve the response time, you can set a higher target crossover frequency than the result that *pidtune()* automatically selects, 0.32. Increase the crossover frequency to 1.0.

```
[Hpi,info] = pidtune(Hp,'PI', 1.0)
```




```
Hpi =

             1
  Kp + Ki * ---
             s

  with Kp = 1.41, Ki = 0.0247

Continuous-time PI controller in parallel form.


info =

                   Stable: 1
       CrossoverFrequency: 1
              PhaseMargin: 43.9973
```

The new controller achieves the higher crossover frequency, but at the cost of a reduced phase margin.

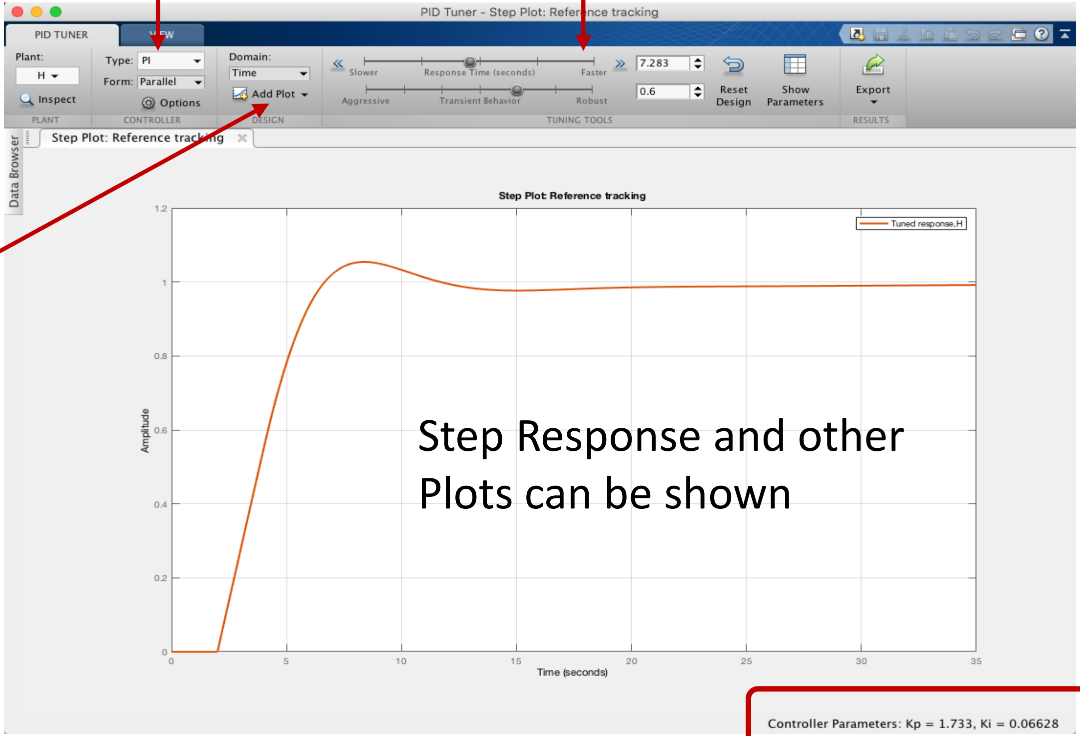
MATLAB PID Tuner

Define Controller Type

Tuning

Define your Process

Add Additional Plots



Step Response and other Plots can be shown

PID Parameters

https://www.halvorsen.blog

Stability Analysis using MATLAB

Hans-Petter Halvorsen

Stability Analysis

How do we figure out that the Feedback System is stable before we test it on the real System?

1. Poles
2. Frequency Response/Bode
3. Simulations (Step Response)

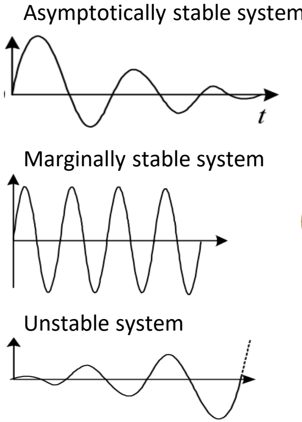
We will do all these things using MATLAB

Stability Analysis

3 Time domain

$T(s)$

Tracking transfer function



$$\lim_{t\to\infty} y(t) = k$$

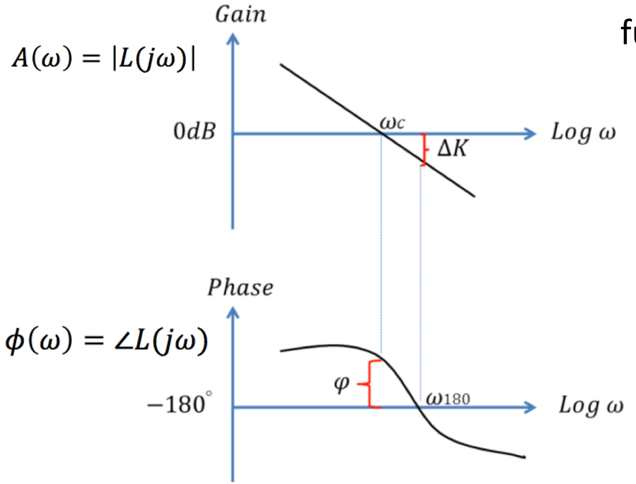
$$0 < \lim_{t\to\infty} y(t) < \infty$$

$$\lim_{t\to\infty} y(t) = \infty$$

2 Frequency domain

$L(s)$

Loop transfer function



Asymptotically stable system: $\omega_c < \omega_{180}$

Marginally stable system: $\omega_c = \omega_{180}$

Unstable system: $\omega_c > \omega_{180}$

1 The Complex domain

Poles

$T(s)$

Tracking transfer function

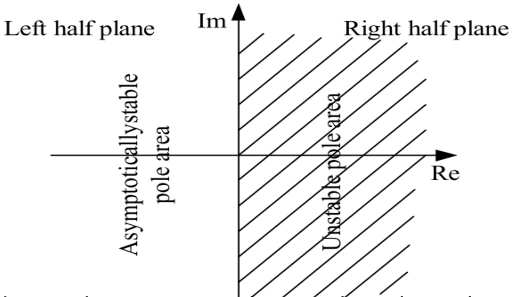
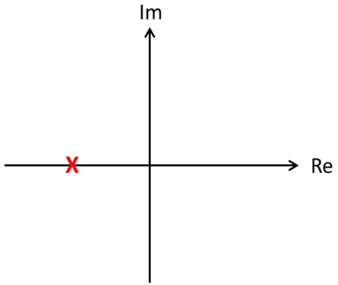


Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010.

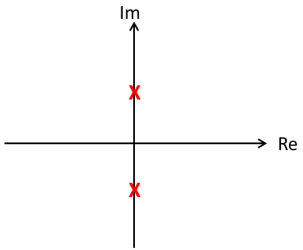
Poles

1 Asymptotically stable system:



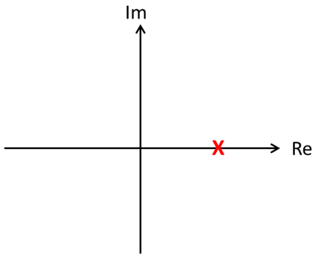
Each of the poles of the transfer function lies strictly in the left half plane (has strictly negative real part).

2 Marginally stable system:

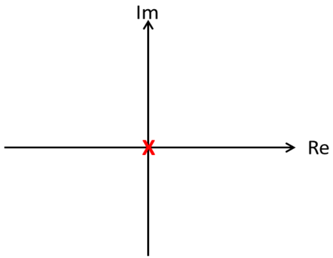


One or more poles lies on the imaginary axis (have real part equal to zero), and all these poles are distinct. Besides, no poles lie in the right half plane.

3 Unstable system:



At least one pole lies in the right half plane (has real part greater than zero).



Or: There are multiple and coincident poles on the imaginary axis.

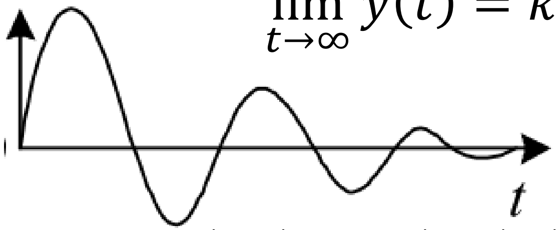
Example: double integrator $H(s) = \frac{1}{s^2}$

Theory

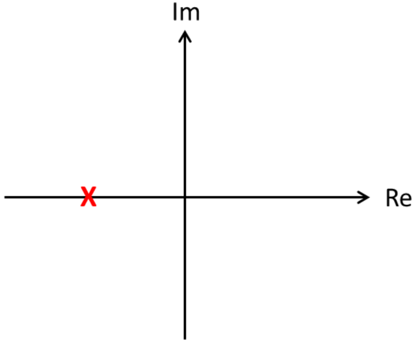
Stability Analysis

Asymptotically stable system:

$$\lim_{t\to\infty} y(t) = k$$

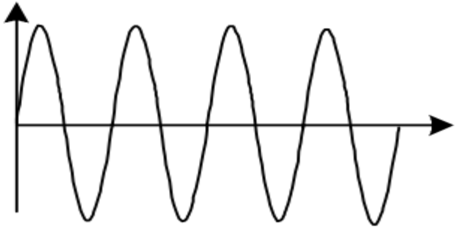


Figures: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010.

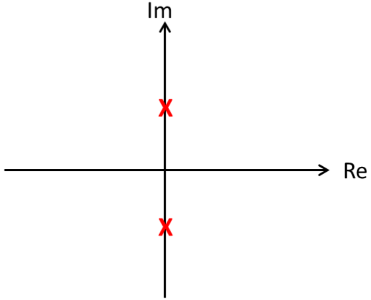


Each of the poles of the transfer function lies strictly in the left half plane (has strictly negative real part).

Marginally stable system:



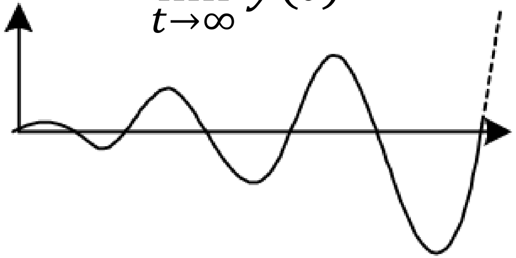
$$0 < \lim_{t\to\infty} y(t) < \infty$$

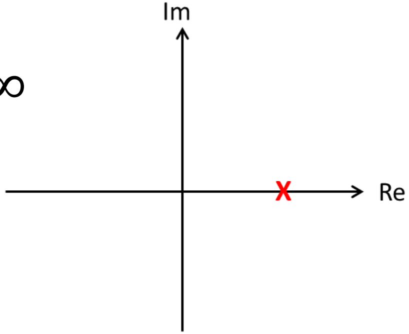


One or more poles lies on the imaginary axis (have real part equal to zero), and all these poles are distinct. Besides, no poles lie in the right half plane.

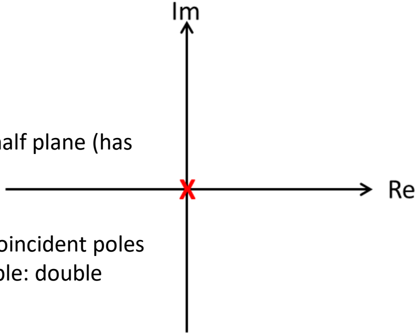
Unstable system:

$$\lim_{t\to\infty} y(t) = \infty$$





At least one pole lies in the right half plane (has real part greater than zero).



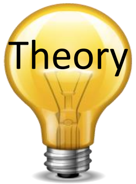
Or: There are multiple and coincident poles on the imaginary axis. Example: double integrator $H(s) = \frac{1}{s^2}$

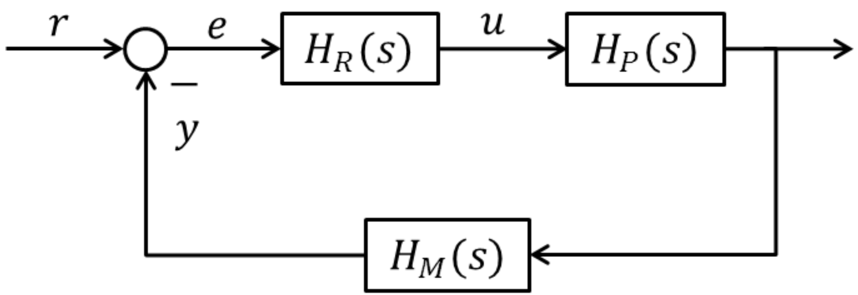
https://www.halvorsen.blog

Stability Analysis of Feedback Systems

Hans-Petter Halvorsen

Stability Analysis of Feedback Systems





1 **Loop Transfer Function** ("Sløyfetransferfunksjonen"):

$$L(s) = H_R H_P H_M$$

```
Hr = ...
Hp = ...
Hm = ...
L = series(series(Hr, Hp), Hm)
```

2 **The Tracking Function** ("Følgeforholdet"):

$$T(s) = \frac{y(s)}{r(s)} = \frac{H_R H_P H_M}{1 + H_R H_P H_M} = \frac{L(s)}{1 + L(s)}$$

```
L = ...
T = feedback(L, 1)
```

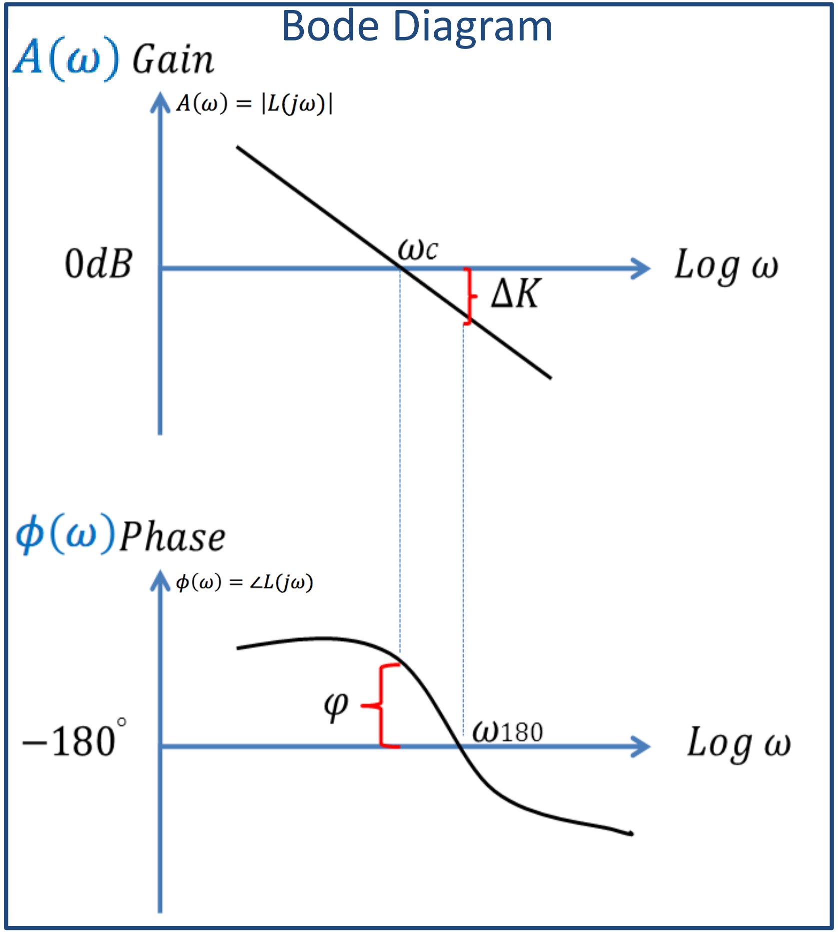
3 **The Sensitivity Function** ("Sensitivitetsfunksjonen"):

$$S(s) = \frac{e(s)}{r(s)} = \frac{1}{1 + L(s)} = 1 - T(s)$$

```
T = ...
S = 1-T
```

Frequency Response and Stability Analysis

Theory



ω_c and ω_{180} are called the crossover-frequencies ("kryssfrekvens")

$A(\omega) = |L(j\omega)|$

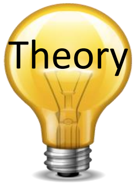
ΔK is the gain margin (GM) of the system ("Forsterkningsmargin"). How much the loop gain can increase before the system becomes unstable

$\phi(\omega) = \angle L(j\omega)$

ϕ is the phase margin (PM) of the system ("Fasemargin"). How much phase shift the system can tolerate before it becomes unstable.

Asymptotically stable system: $\omega_c < \omega_{180}$
Marginally stable system: $\omega_c = \omega_{180}$
Unstable system: $\omega_c > \omega_{180}$

Frequency Response and Stability Analysis



The definitions are as follows:

Gain Crossover-frequency - ω_c :

$$|L(j\omega_c)| = 1 = 0dB$$

Phase Crossover-frequency - ω_{180} :

$$\angle L(j\omega_{180}) = -180^o$$

Gain Margin - GM (ΔK):

$$GM\ [dB] = -|L(j\omega_{180})|\ [dB]$$

Phase margin PM (φ):

$$PM = 180^o + \angle L(j\omega_c)$$

ω_{180} is the gain margin frequency, in radians/second. A gain margin frequency indicates where the model phase crosses -180 degrees.

GM (ΔK) is the gain margin of the system.

ω_c is phase margin frequency, in radians/second. A phase margin frequency indicates where the model magnitude crosses 0 decibels.

PM (φ) is the phase margin of the system.

We have that:

1. Asymptotically stable system: $\omega_c < \omega_{180}$
2. Marginally stable system: $\omega_c = \omega_{180}$
3. Unstable system: $\omega_c > \omega_{180}$

https://www.halvorsen.blog

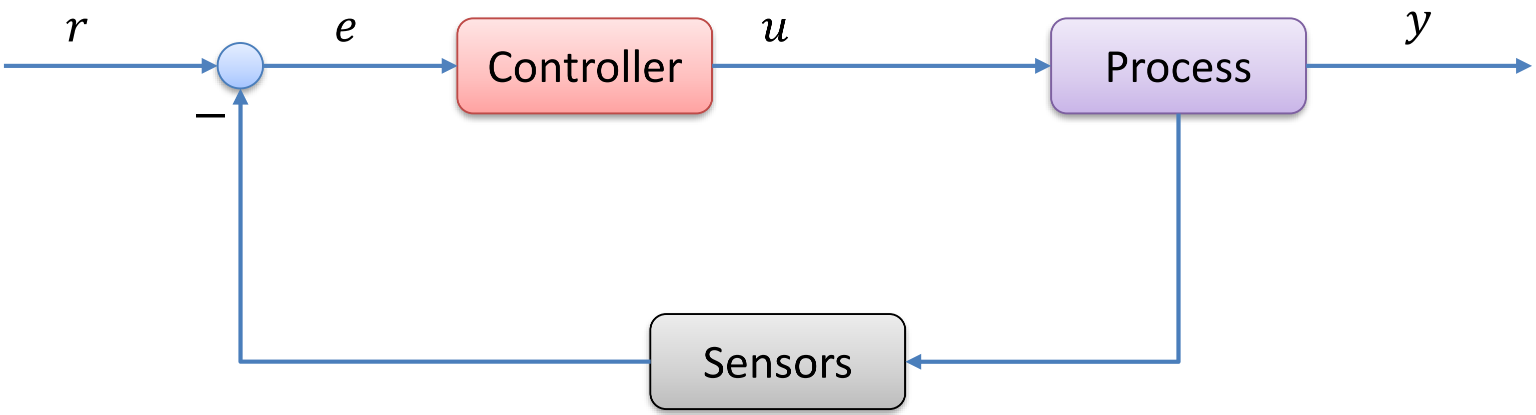
Stability Analysis of Feedback System - Example

Hans-Petter Halvorsen

Example

We will use the following system as an example:

$$H_c = K_p \qquad H_p = \frac{1}{s(s+1)}$$



$$H_m = \frac{1}{3s+1}$$

Analysis of the Feedback System

Loop transfer function: $\boldsymbol{L(s)}$

We need to find the Loop transfer function $L(s)$ using MATLAB.

The Loop transfer function is defined as:

$$L(s) = H_c H_p H_m$$

We will use the built-in function *series()* in MATLAB.

Tracking transfer function: $\boldsymbol{T(s)}$

We need to find the Tracking transfer function $T(s)$ using MATLAB.

The Tracking transfer function is defined as:

$$T(s) = \frac{y(s)}{r(s)} = \frac{H_c H_p H_m}{1+H_c H_p H_m} = \frac{L(s)}{1+L(s)}$$

We will use the built-in function *feedback()* in MATLAB.

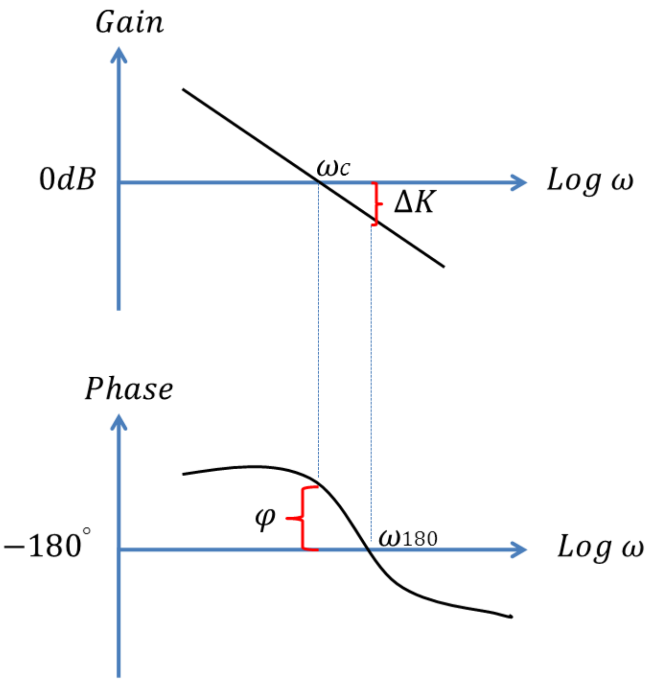
Sensitivity transfer function: $\boldsymbol{S(s)}$

We need to find the Sensitivity transfer function $S(s)$ using MATLAB.

The Sensitivity transfer function is defined as:

$$S(s) = \frac{e(s)}{r(s)} = \frac{1}{1+L(s)} = 1 - T(s)$$

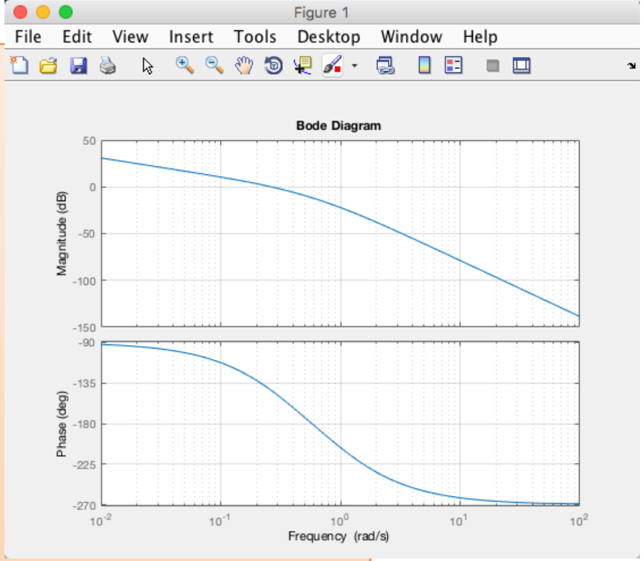
Stability Analysis



- Plot the Bode plot for the system using e.g., the bode() function in MATLAB
- Find the crossover-frequencies (ω_{180}, ω_c) and stability margins GM ($A(\omega)$), PM ($\phi(\omega)$) of the system ($L(s)$) from the Bode plot.
- Plot also Bode diagram where the crossover-frequencies, GM and PM are illustrated. Tip! Use the *margin()* function in MATLAB.
- Use also the *margin()* function in order to find values for $\omega_{180}, \omega_c, A(\omega), \phi(\omega)$ directly.
- You should compare and discuss the results.
- How much can you increase K_p before the system becomes unstable?

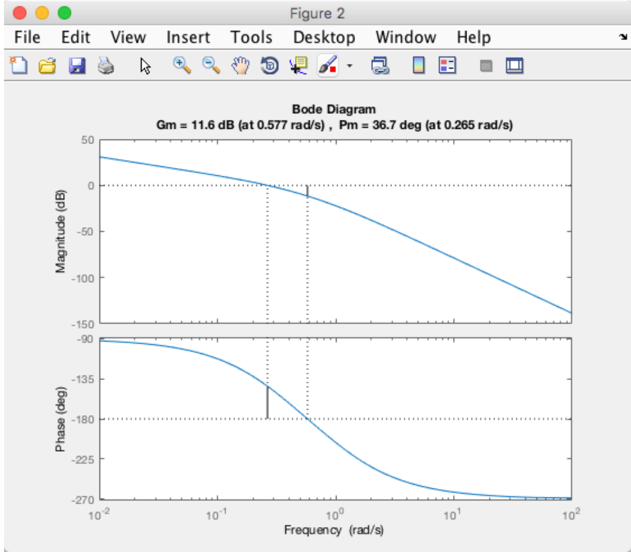
MATLAB Code:

```
clear, clc, clf

% The Process Transfer function Hp(s)
num_p=[1];
den1=[1, 0];
den2=[1, 1];
den_p = conv(den1,den2);
Hp = tf(num_p, den_p)

% The Measurement Transfer function Hm(s)
num_m=[1];
den_m=[3, 1];
Hm = tf(num_m, den_m)

% The Controller Transfer function Hr(s)
Kp = 0.35;
Hr = tf(Kp)

% The Loop Transfer function
L = series(series(Hr, Hp), Hm)

% Bode Diagram
figure(1)
bode(L),grid on

figure(2)
margin(L)

[gm, pm, w180, wc] = margin(L);

wc
w180
gm
gmdB = 20*log10(gm)
pm
```



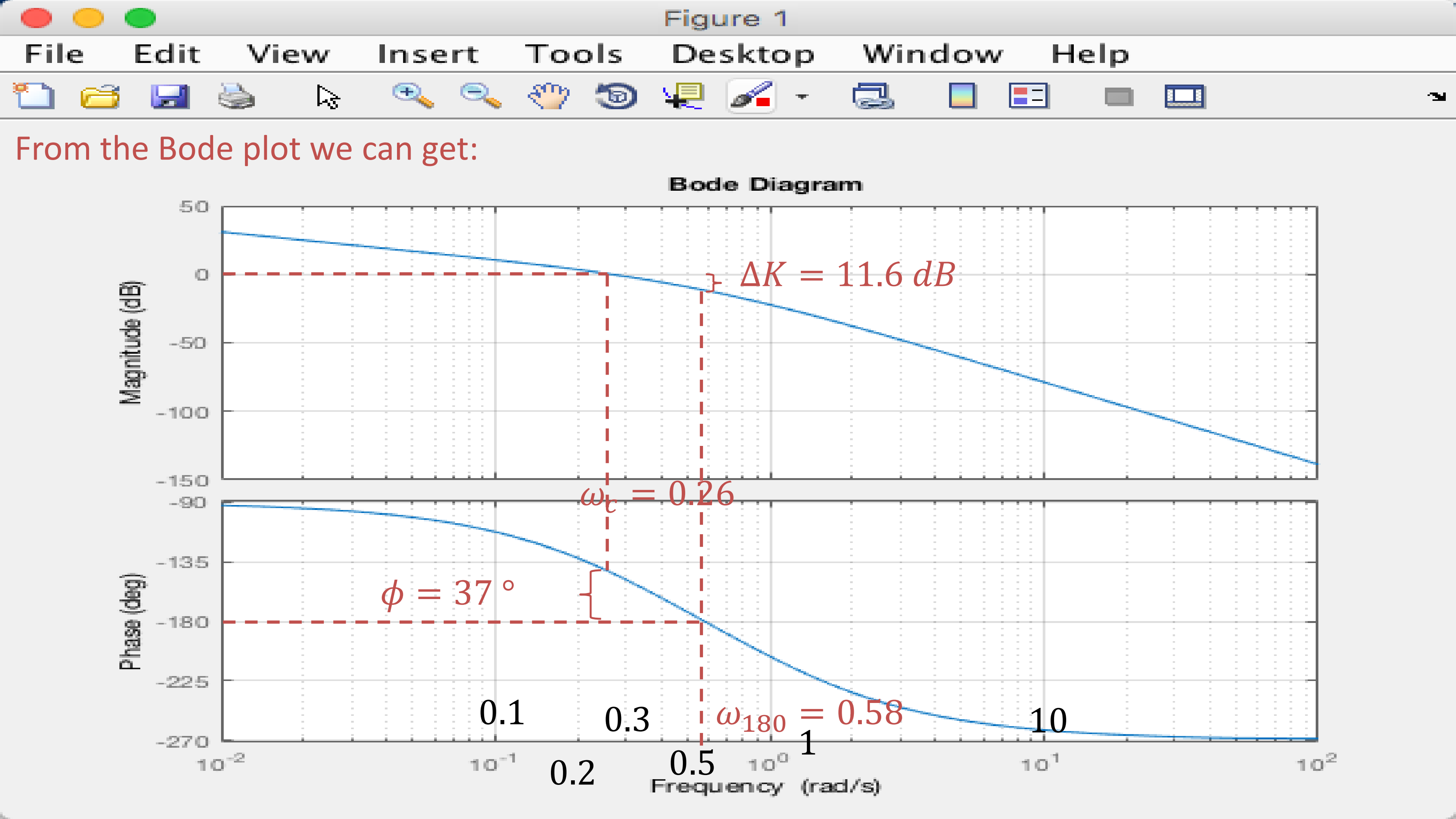
bode(L)

margin(L)




```
wc     =   0.2649 rad/s
w180   =   0.5774 rad/s
gm     =   3.8095
gmdB   =   11.6174 dB
pm     =   36.6917 degrees
```

From the Bode plot we can get:



Stable vs. Unstable System

- We will find and use different values of K_p where we get a marginally stable system, an asymptotically stable system and an unstable system.
- We will Plot the time response for the tracking function using the *step()* function in MATLAB for all these 3 categories. How can we use the step response to determine the stability of the system?
- We will find ω_{180}, ω_c, $A(\omega)$ and $\phi(\omega)$ in all 3 cases. We will see how we use ω_c and ω_{180} to determine the stability of the system.
- We will find and plot the poles and zeros for the system for all the 3 categories mentioned above. We will see how we can we use the poles to determine the stability of the system.
- Bandwidth: We will plot the Loop transfer function $L(s)$, the Tracking transfer function $T(s)$ and the Sensitivity transfer function $S(s)$ in a Bode diagram for the system for all the 3 categories mentioned above.

Stable System

$K_p = 0.35$

For what K_p becomes the system marginally stable?

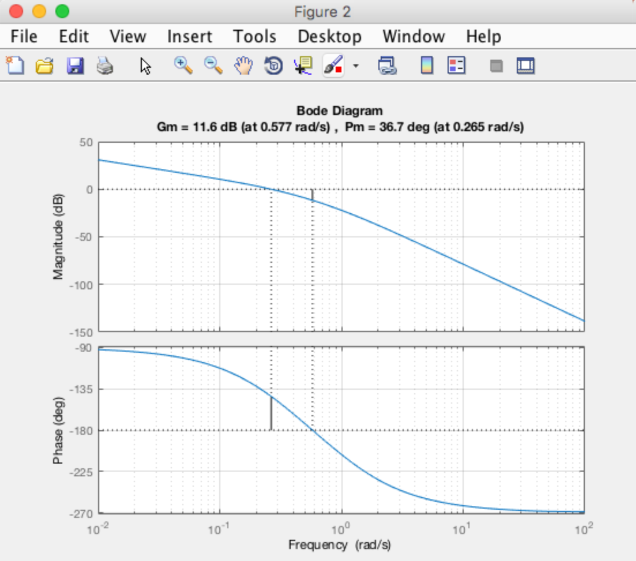
$K_{pm} = 0.35\ \times\ \Delta K = 0.35\ \times\ 3.8 \approx 1{,}32$

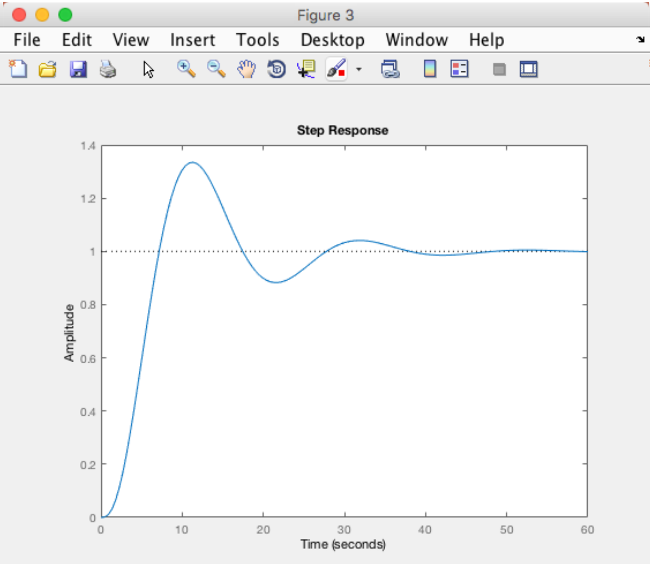
```
wc     =   0.2649 rad/s
w180   =   0.5774 rad/s
gm     =   3.8095
gmdB   =   11.6174 dB
pm     =   36.6917 degrees
```

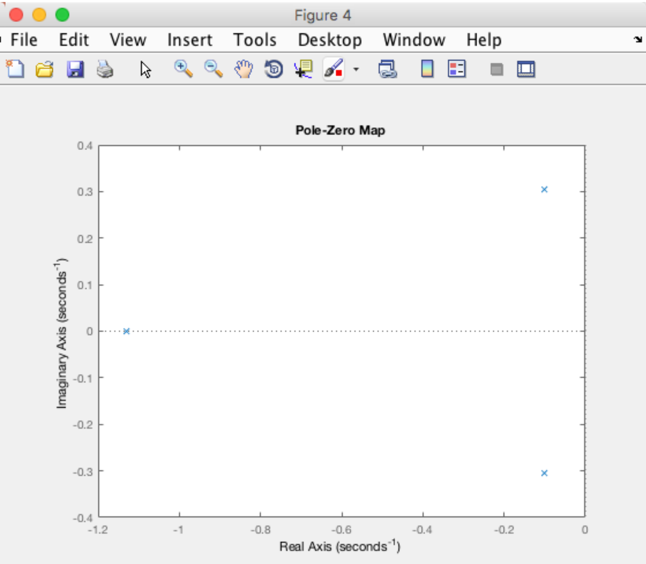
$\omega_c < \omega_{180}$

$\lim_{t \to \infty} y(t) = 1$

Poles in the left half plane







Marginally Stable System

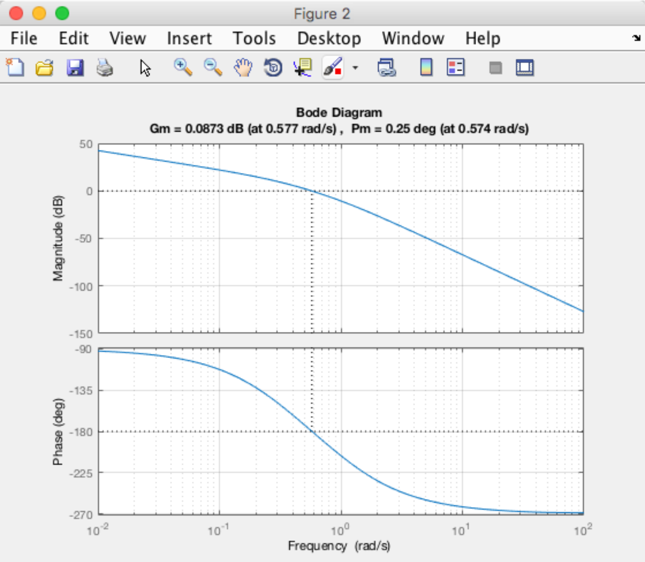
```
wc    = 0.5744 rad/s
w180  = 0.5774 rad/s
gm    = 1.0101
gmdB  = 0.0873 dB
pm    = 0.2500 degrees
```

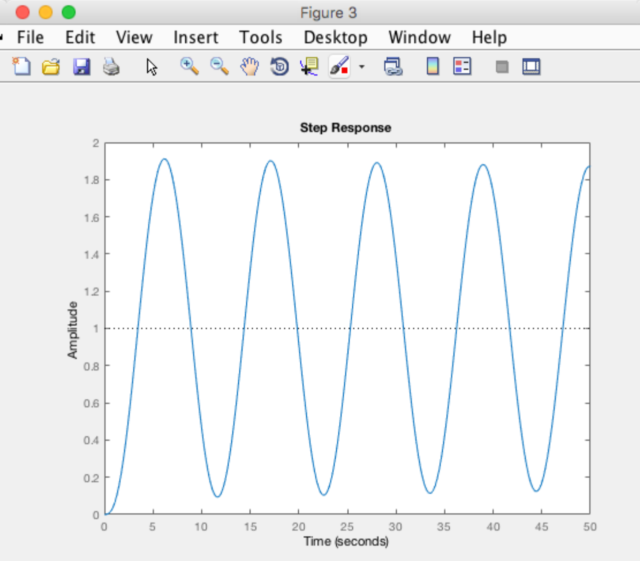
$$K_p = 1.32$$

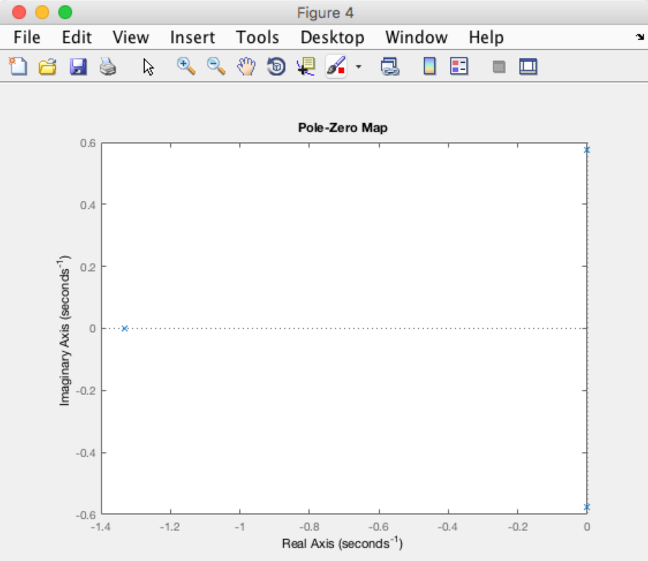
$$\omega_c = \omega_{180}$$

$$0 < \lim_{t \to \infty} y(t) < \infty$$

Poles at the imaginary axis







Unstable System

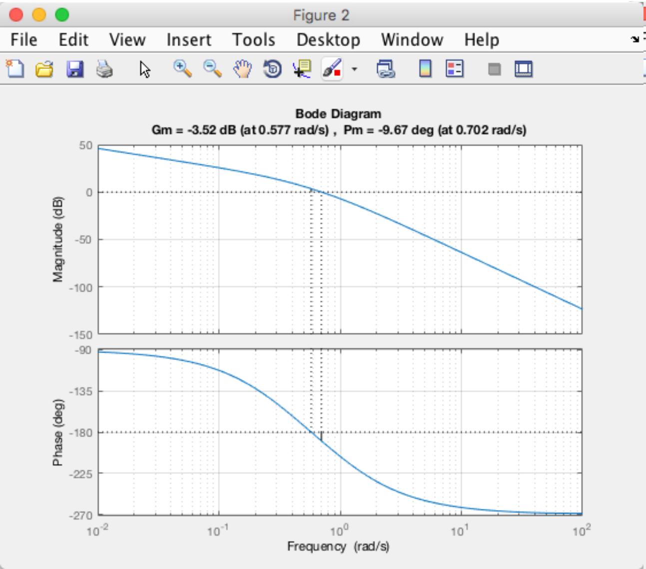
```
wc    =  0.7020
w180  =  0.5774
gm    =  0.6667
gmdB  = -3.5218
pm    = -9.6664
```

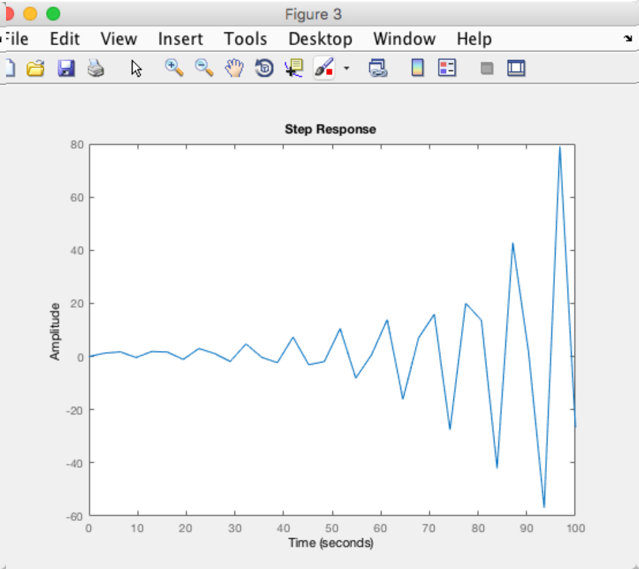
$K_p = 2$

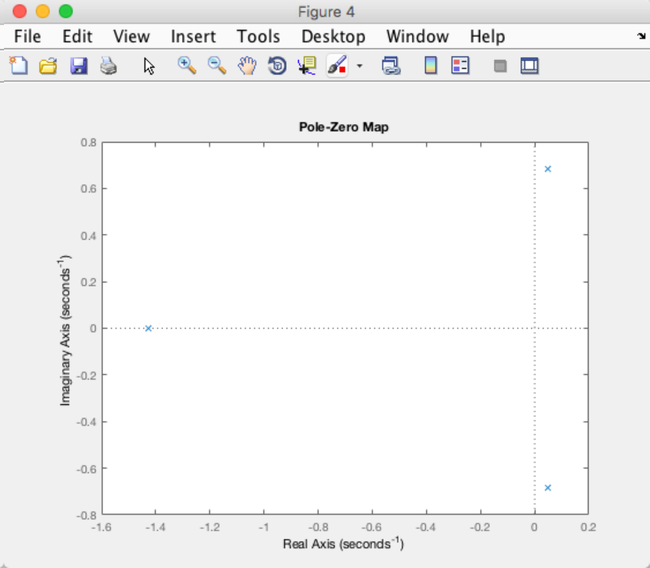
$\omega_c > \omega_{180}$

$$\lim_{t\to\infty} y(t) = \infty$$

Poles in the right half plane







MATLAB Code

```
clear, clc, clf

% The Process Transfer function Hp(s)
num_p=[1];
den1=[1, 0];
den2=[1, 1];
den_p = conv(den1,den2);
Hp = tf(num_p, den_p)

% The Measurement Transfer function Hm(s)
num_m=[1];
den_m=[3, 1];
Hm = tf(num_m, den_m)

% The Controller Transfer function Hr(s)
Kp = 0.35; % Stable System
%Kp = 1.32; % Marginally Stable System
%Kp = 2; % Unstable System
Hr = tf(Kp)

% The Loop Transfer function
L = series(series(Hr, Hp), Hm)
% Tracking transfer function
T=feedback(L,1);
% Sensitivity transfer function
S=1-T;

...
```

```
...

% Bode Diagram
figure(1)
bode(L), grid on

figure(2)
margin(L), grid on
[gm, pm, w180, wc] = margin(L);

wc
w180
gm
gmdB = 20*log10(gm)
pm

% Simulating step response for control system
(tracking transfer function)
figure(3)
step(T)

% Poles
pole(T)
figure(4)
pzmap(T)

% Bandwidth
figure(5)
bodemag(L,T,S), grid on
```

Theory

"Golden rules" of Stability Margins for a Control System

Reference: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010.

Gain Margin (GM): (Norwegian: "Forsterkningsmargin")

$$2\ (6dB) < \Delta K < 4\ (12dB)$$

Phase Margin (PM): (Norwegian: "Fasemargin")

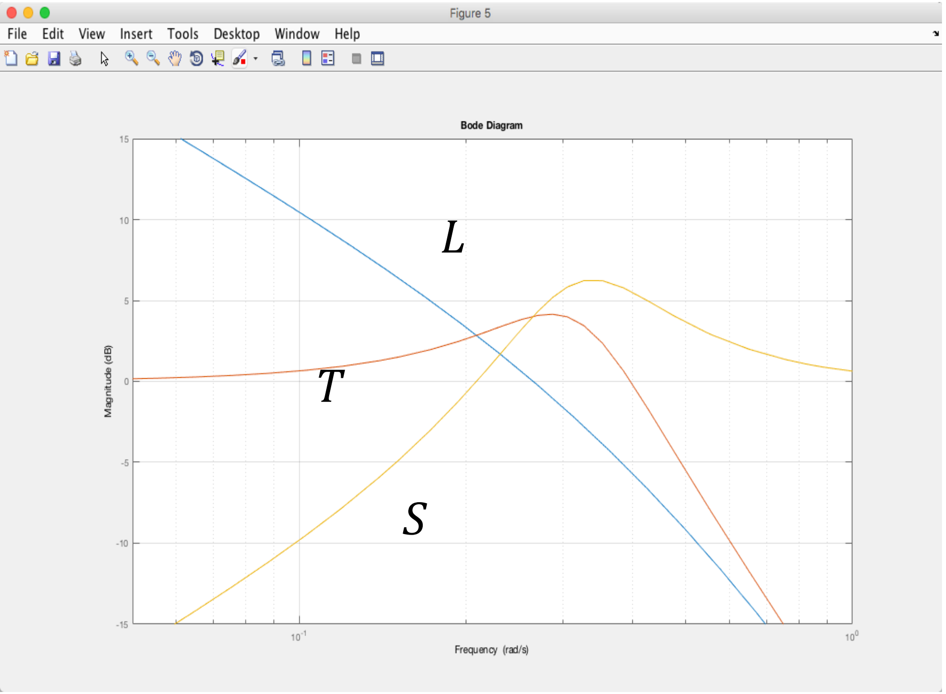
$$30^\circ < \varphi < 60^\circ$$

https://www.halvorsen.blog

The Bandwidth of the Control System

Hans-Petter Halvorsen

The Bandwidth of the Control System

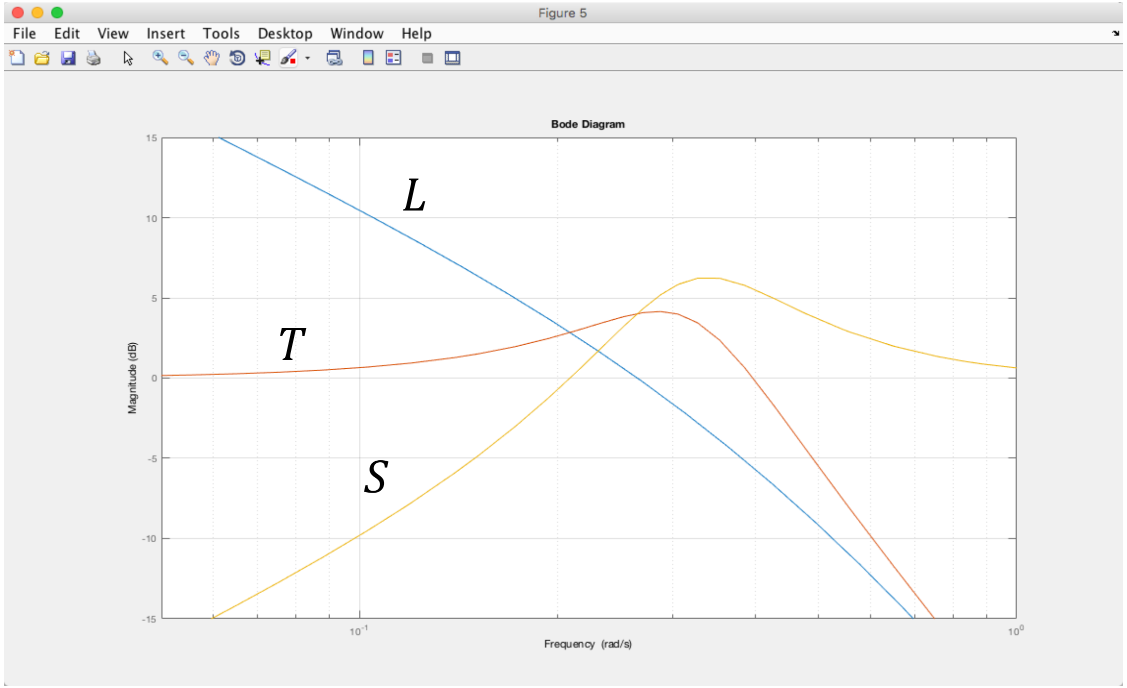


- You should plot the Loop transfer function $L(s)$, the Tracking transfer function $T(s)$ and the Sensitivity transfer function $S(s)$ in the same Bode diagram.
- Use, e.g., the *bodemag()* function in MATLAB (only the gain diagram is of interest in this case, not the phase diagram).
- Use the values for K_p and T_i found in a previous Tasks.
- You need to find the different bandwidths $\boldsymbol{\omega_t, \omega_c, \omega_s}$ (see the sketch below).

Theory

The Bandwidth of the Control System

The bandwidth of a control system is the frequency which divides the frequency range of good tracking and poor tracking.

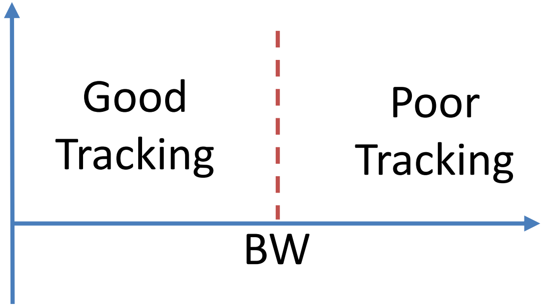


3 different Bandwidth definitions:

$$\omega_c \rightarrow L(j\omega) \rightarrow 0dB$$

$$\omega_t \rightarrow T(j\omega) \rightarrow -3dB$$

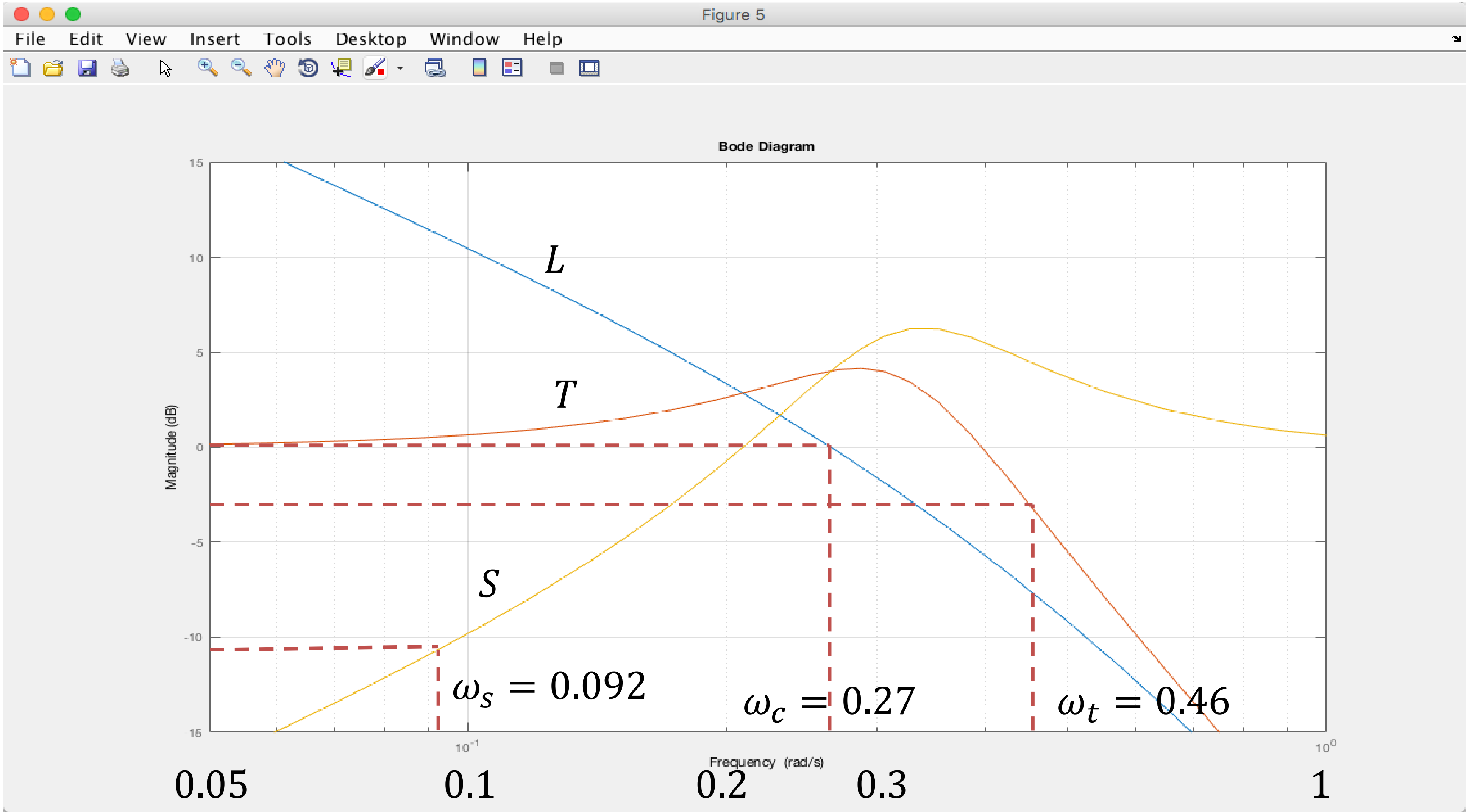
$$\omega_s \rightarrow S(j\omega) \rightarrow -11dB$$



Good Set-point Tracking: $|S(j\omega)| \ll 1, |T(j\omega)| \approx 1, |L(j\omega)| \gg 1$

$K_p = 0.35$

Good Set-point Tracking: $|S(j\omega)| \ll 1, |T(j\omega)| \approx 1, |L(j\omega)| \gg 1$



https://www.halvorsen.blog

PI Controller Example

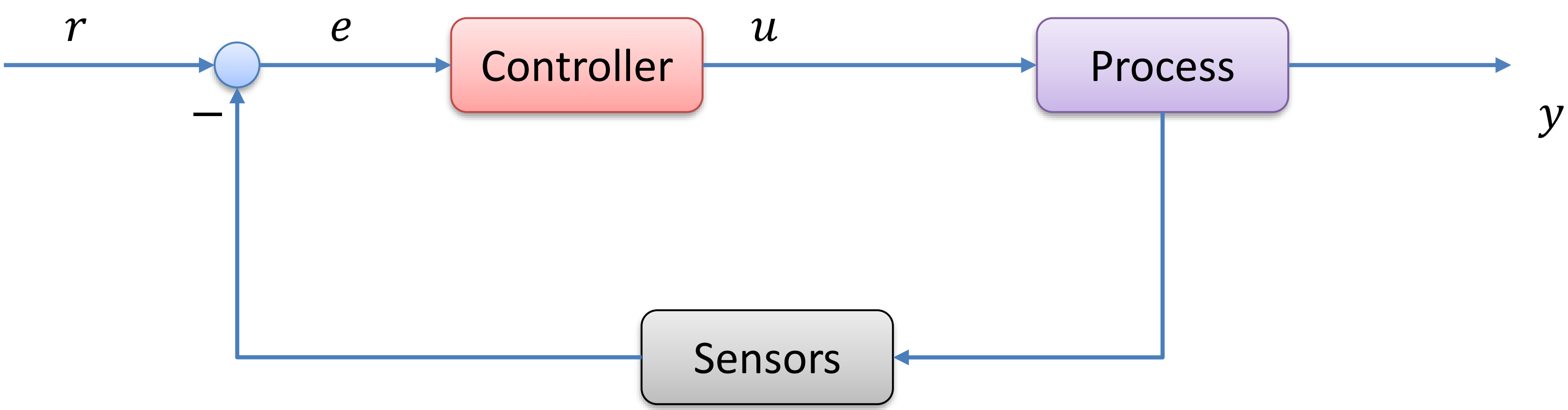
Hans-Petter Halvorsen

PI Controller - Example

We will use the following system as an example:

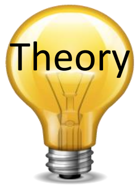
$$H_c = \frac{K_p(T_i s + 1)}{T_i s}$$

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



$$H_m = \frac{1}{3s+1}$$

Ziegler–Nichols Frequency Response method



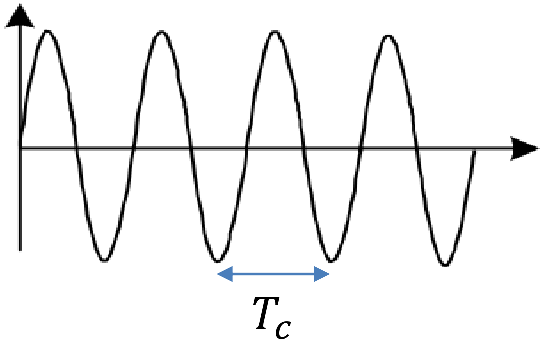
Assume you use a P controller only $T_i = \infty, T_d = 0$. Then you need to find for which K_p the closed loop system is a marginally stable system ($\omega_c = \omega_{180}$). This K_p is called K_c (critical gain). The T_c (critical period) can be found from the damped oscillations of the closed loop system. Then calculate the PI(D) parameters using the formulas below.

Controller	K_p	T_i	T_d
P	$0.5K_c$	∞	0
PI	$0.45K_c$	$\frac{T_c}{1.2}$	0
PID	$0.6K_c$	$\frac{T_c}{2}$	$\frac{T_c}{8}$

K_c- Critical Gain

T_c- Critical Period

Marginally stable system:



$$\omega_c = \omega_{180}$$

$$0 < \lim_{t \to \infty} y(t) < \infty$$

$$T_c = \frac{2\pi}{\omega_{180}}$$

https://en.wikipedia.org/wiki/Ziegler–Nichols_method

PI Controller using Ziegler–Nichols

Ziegler–Nichols (PI Controller):

$$K_p = 0.45K_c$$

$$T_i = \frac{T_c}{1.2}$$

From previous Simulations:

$$K_c = 1.32$$

$$T_c = \frac{2\pi}{\omega_{180}} = \frac{2\pi}{0.58}$$

This gives the following PI Parameters:

$$K_p = 0.45K_c = 0.45 \cdot 1.32 \approx 0.6$$

$$T_i = \frac{T_c}{1.2} = \frac{\frac{2\pi}{0.58}}{1.2} \approx 9s$$

Skogestad's method

- The Skogestad's method assumes you apply a step on the input (u) and then observe the response and the output (y), as shown below.
- If we have a model of the system (which we have in our case), we can use the following Skogestad's formulas for finding the PI(D) parameters directly.

Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010.

Process type	$H_{psf}(s)$ (process)	K_p	T_i	T_d
Integrator + delay	$\frac{K}{s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	0
Time-constant + delay	$\frac{K}{Ts+1}e^{-\tau s}$	$\frac{T}{K(T_C+\tau)}$	$\min\,[T,\, c\,(T_C+\tau)]$	0
Integr + time-const + del.	$\frac{K}{(Ts+1)s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	T
Two time-const + delay	$\frac{K}{(T_1s+1)(T_2s+1)}e^{-\tau s}$	$\frac{T_1}{K(T_C+\tau)}$	$\min\,[T_1,\, c\,(T_C+\tau)]$	T_2
Double integrator + delay	$\frac{K}{s^2}e^{-\tau s}$	$\frac{1}{4K(T_C+\tau)^2}$	$4\,(T_C+\tau)$	$4\,(T_C+\tau)$

T_c is the time-constant of the control system which the user must specify

We set, e.g., $T_C = 5\ s$ and $c = 1.5$:

$$K_p = \frac{1}{K(T_c+\tau)} = \frac{1}{1(10+0)} = \frac{1}{5} = 0.2$$

$$T_i = c(T_c+\tau) = 1.5(5+0) = 7.5s$$

Our Process:

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

$$K = 1, T = 1, \tau = 0$$

MATLAB

Ziegler–Nichols and Skogestad’s Formulas:

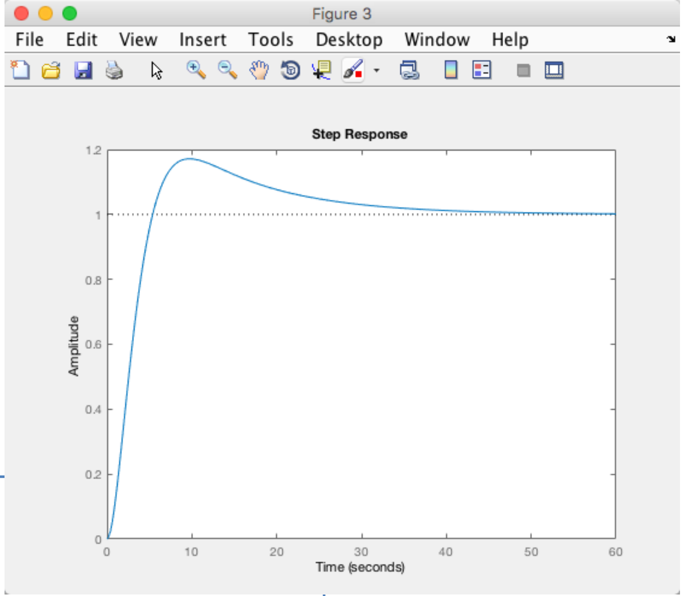
```
% Ziegler-Nicols Method

Kc = 1.32; % Critical Gain
Tc = 2*pi/w180; % Tc - Critical Period

Kp = 0.45 * Kc
Ti = Tc/1.2

% Skogestad's Method
Tc = 5; % time-constant of the control system which the user must specify
c = 1.5;

% H=K*e(-Tau*s)/(T*s+1)*s
Kp = 1/K*(Tc)
Ti = c*(Tc+Tau)
```

pidtune() MATLAB function

Example

```
clear, clc

%Define Process
num = 1;
den = [1,1,0];
Hp = tf(num,den)

% Find PI Controller
[Hpi,info] = pidtune(Hp,'PI')

%Bode Plots
figure(1)
bode(Hp)
grid on

figure(2)
bode(Hpi)
grid on

% Feedback System
T = feedback(Hpi*Hp, 1);
figure(3)
step(T)
```

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




```
Hpi =

             1
  Kp + Ki * ---
             s

  with Kp = 0.333, Ki = 0.023

Continuous-time PI controller in parallel form.


info =

                Stable: 1
    CrossoverFrequency: 0.3237
           PhaseMargin: 60.0000
```

$$K_p = 0.33$$

$$T_i = \frac{1}{K_i} = \frac{1}{0.023} \approx 43.5s$$

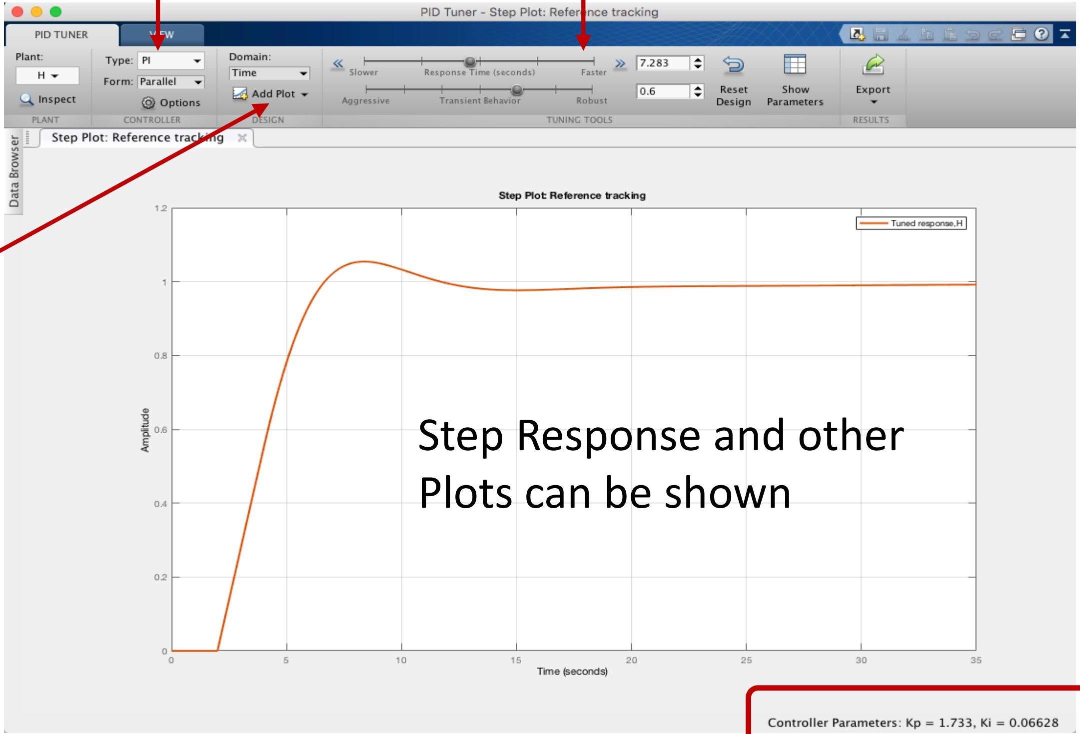
MATLAB PID Tuner

Define Controller Type

Tuning

Define your Process

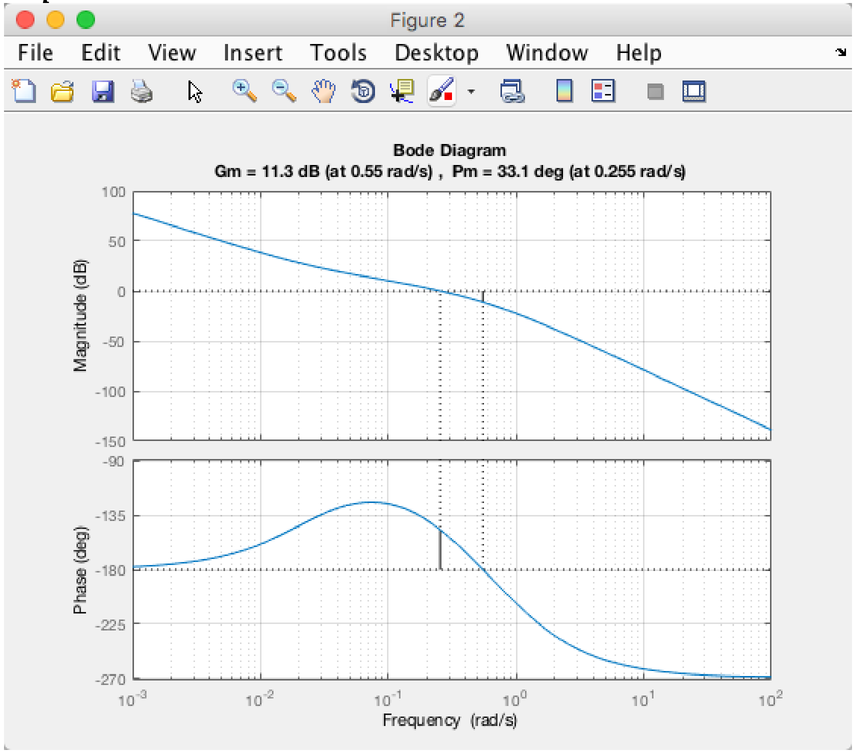
Add Additional Plots

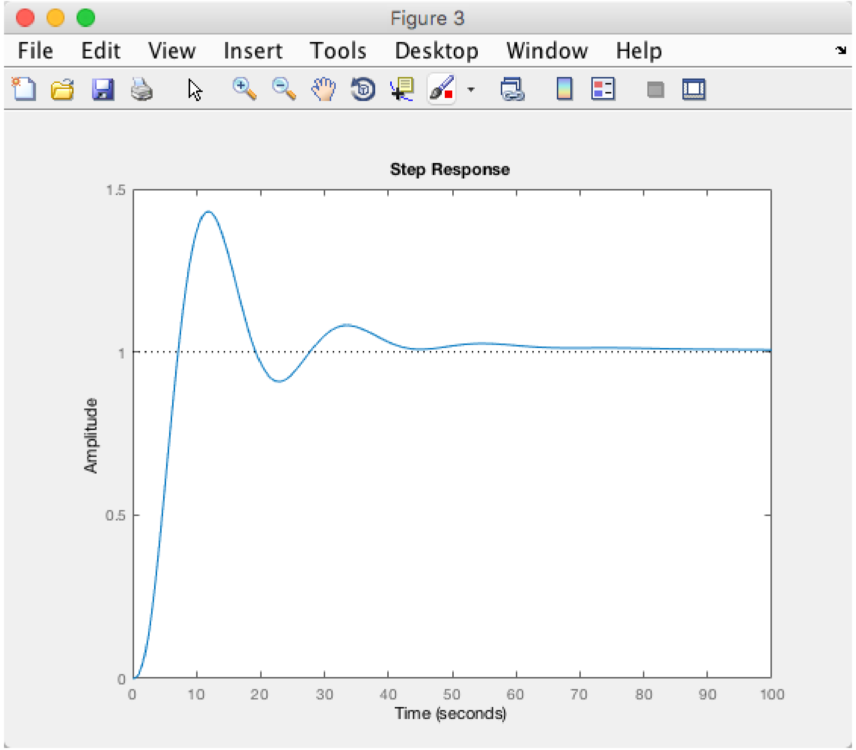


PID Parameters

MATLAB Simulations

$K_p = 0.33, T_i = 43.5s$





$GM = 11.3dB, PM = 33.1°$

MATLAB Simulations

```
clear, clc, clf

% The Process Transfer function Hp(s)
num_p=[1];
den1=[1, 0];
den2=[1, 1];
den_p = conv(den1,den2);
Hp = tf(num_p, den_p);

% The Measurement Transfer function Hm(s)
num_m=[1];
den_m=[3, 1];
Hm = tf(num_m, den_m);

% The PI Controller Transfer function Hc(s)
%Kp = 0.6; Ti = 9; % Ziegler?Nichols
%Kp = 0.2; Ti = 7.5; % Skogestad
Kp = 0.33; Ti = 43.5; % MATLAB pidtune() function

num = Kp*[Ti, 1];
den = [Ti, 0];
Hc = tf(num,den);

% The Loop Transfer function
L = series(series(Hc, Hp), Hm);
% Tracking transfer function
T=feedback(L,1);
% Sensitivity transfer function
S=1-T;

...
```

```
...

% Bode Diagram
figure(1)
bode(L), grid on

figure(2)
margin(L), grid on

[gm, pm, w180, wc] = margin(L);

wc
w180
gm
gmdB = 20*log10(gm)
pm

% Simulating step response for control system
(tracking transfer function)
figure(3)
step(T)

% Poles
pole(T)
figure(4)
pzmap(T)

% Bandwidth
figure(5)
bodemag(L,T,S), grid on
```

References

1. D. Ruscio, System Theory - State Space Analysis and Control Theory, Lecture Notes, 2015
2. F. Haugen, Advanced Dynamics and Control: TechTeach, 2010.
3. R. C. Dorf and R. H. Bishop, Modern Control Systems. Eleventh Edition: Pearson Prentice Hall.
4. https://www.halvorsen.blog

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog