# 01211433 VISION AND CONTROL OF INDUSTRIAL ROBOTS

ดร.วโรดม ตู้จินดา

ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์

ม.เกษตรศาสตร์

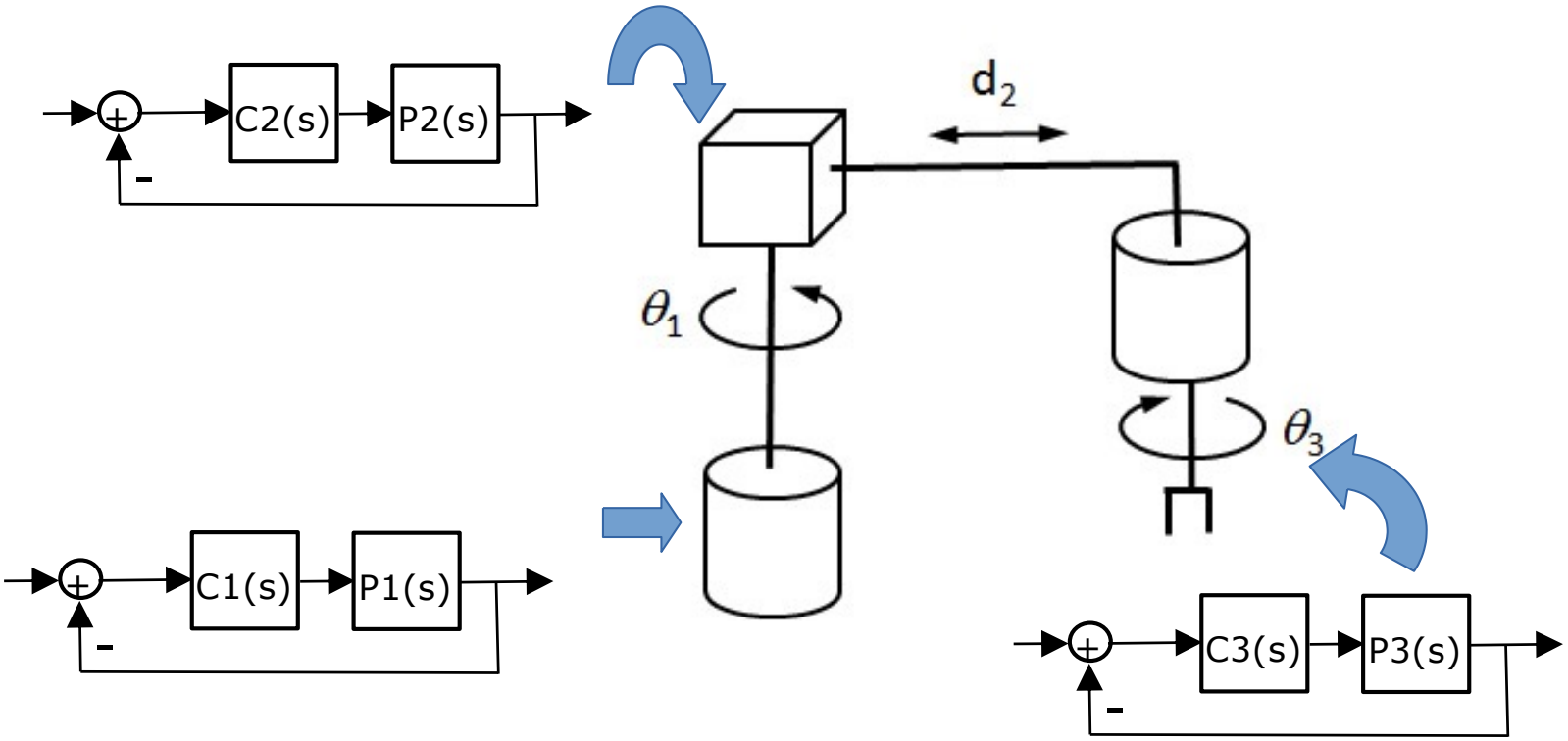
# Lecture 2 : Independent Joint Dynamics & Feedback Control Basics

พลวัตข้อต่ออิสระและพื้นฐานการควบคุมป้อนกลับ

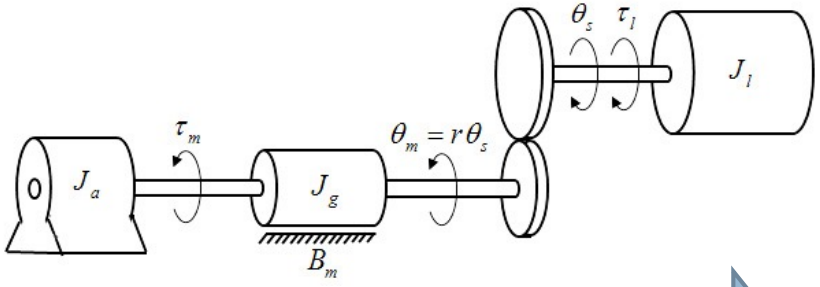
# หัวข้อบรรยาย

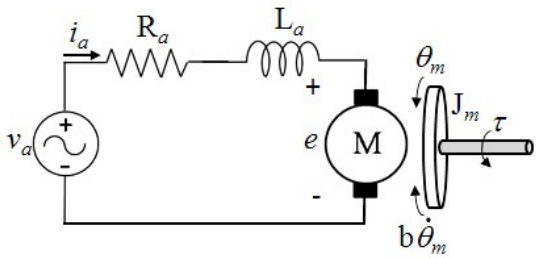
- ฟังก์ชันถ่ายโอนของข้อต่ออิสระ
- ฟังก์ชันถ่ายโอนวงเปิดและวงปิด
- คุณสมบัติการป้อนกลับ
  - เสถียรภาพ (stability)
  - สมรรถนะ (performance)
    - การตามรอย (tracking)
    - การลดทอนการรบกวน (disturbance attenuation)
- เงื่อนไขบังคับเชิงพีชคณิต (Algebraic Constraints)

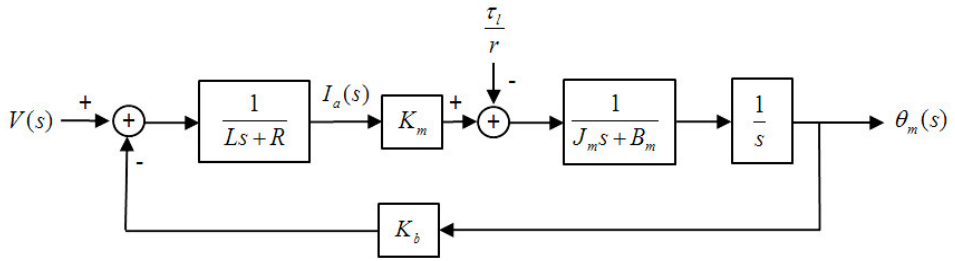
# พื้นฐานการควบคุมแบบข้อต่ออิสระ

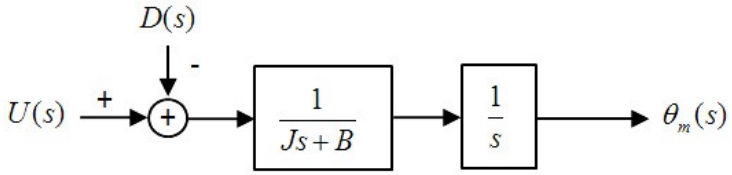


# ฟังก์ชันถ่ายโอนของข้อต่ออิสระ




รายละเอียดใน ceb_m1.ipynb

$$P(s) = \frac{1}{s(Js+B)}$$

# Python Libraries used

**Requirement :** Python 3 with Python control systems library

Execute the commands below to install Python control systems library in Colab

```
In [ ]: !pip install slycot
```

```
In [ ]: !pip install control
```

For Colab user

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import control as ctl
```

# Install ControlSystems package for Julia


# Links

- Python :
  - https://github.com/dewdotninja/control_python
- Julia :
  - https://dewdotninja.github.io/julia/control/julia_control.html

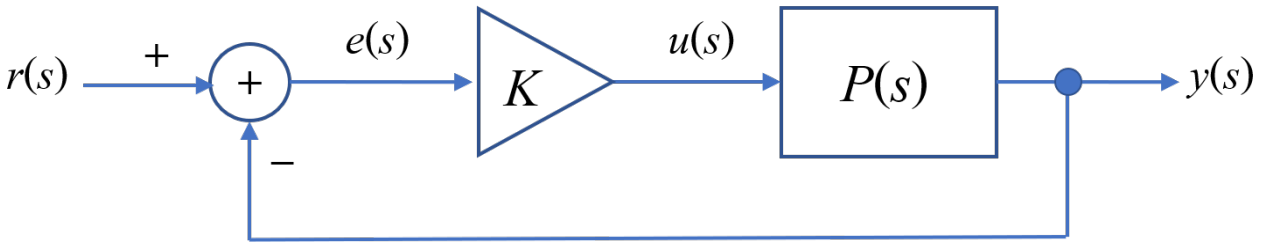
# Module 2 : Feedback Properties

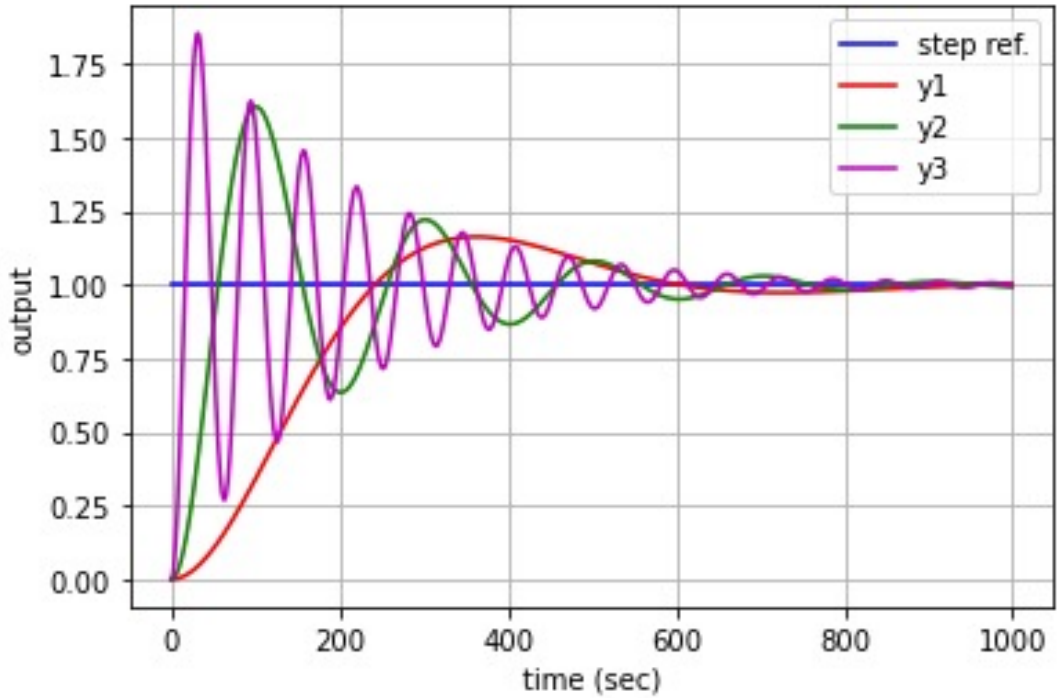
## ceb_m2.ipynb

**Module Key Study Points**

- The benefiets of feedback
- Step response and tracking performance
- Stability judgment from transfer function pole locations
- Root-locus plot
- Disturbance attenuation performance

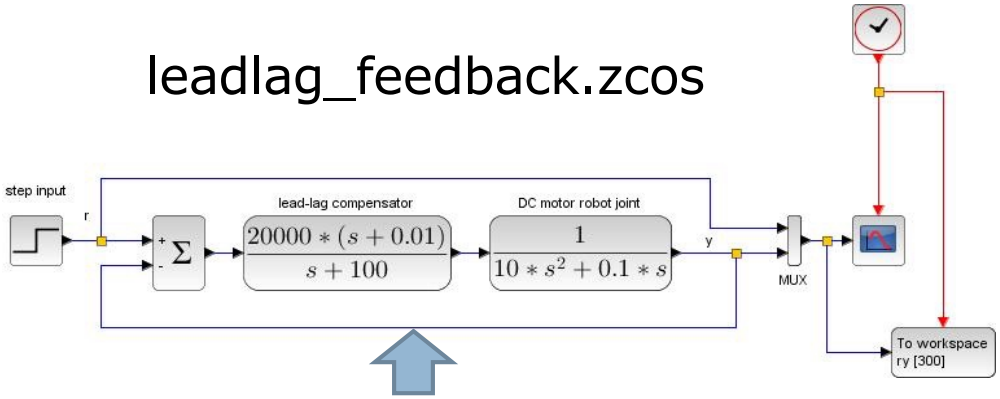
# การป้อนกลับสัดส่วน



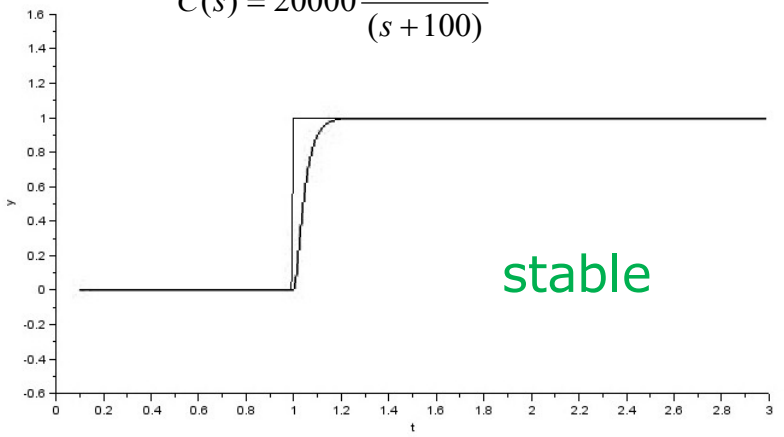
# ตัวควบคุมเชิงเส้น (linear controllers)

leadlag_feedback.zcos



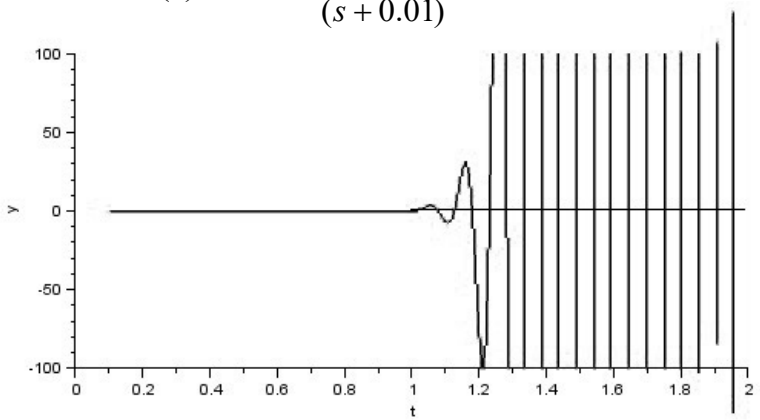

$$C(s) = K\frac{(s+z)}{(s+p)}$$

$$C(s) = 20000\frac{(s+0.01)}{(s+100)}$$




stable

$$C(s) = 20000\frac{(s+100)}{(s+0.01)}$$

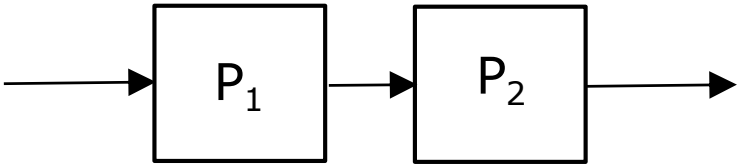



Unstable!

Python version in ceb_m2.ipynb

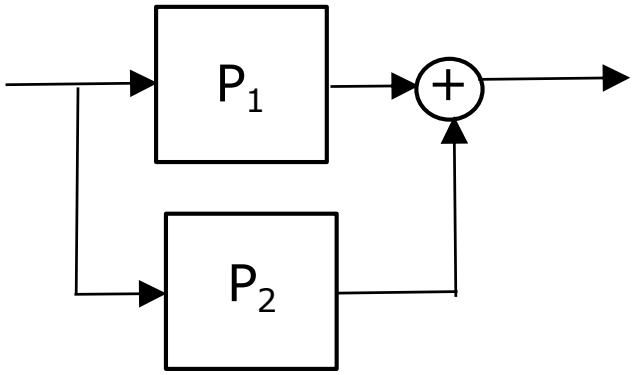
# การลดทอนแผนภาพบล็อก (block diagram reduction)

Series



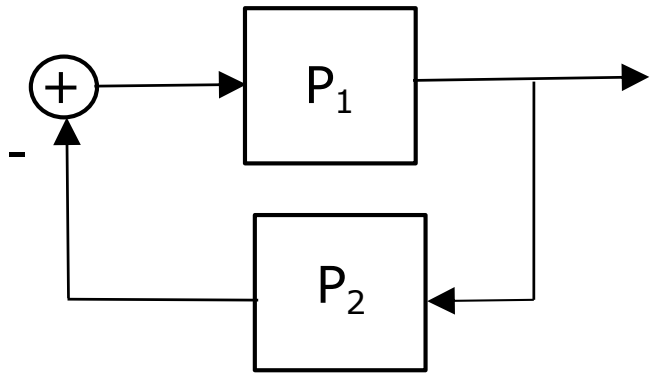

$$P = P_1 P_2$$

Parallel



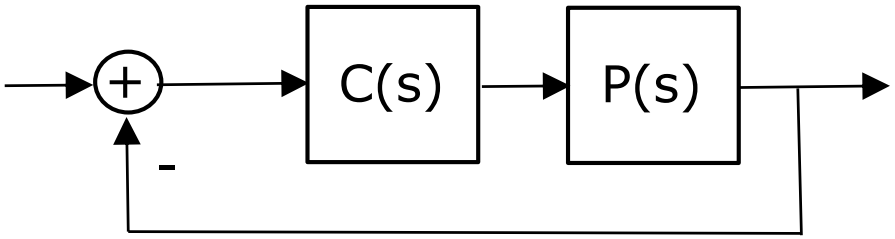

$$P = P_1 + P_2$$

Feedback (negative)




$$P = \frac{P_1}{1 + P_1 P_2}$$

# นิยามฟังก์ชันถ่ายโอนวงเปิดและวงปิด




*ฟังก์ชันวง :* Loop t.f. $L(s) = C(s)P(s)$

*ฟังก์ชันความไว :* Sensitivity t.f. $S(s) = \dfrac{1}{1+L(s)}$

*ฟังก์ชันเติมเต็มความไว :*

Complementary sensitivity t.f. $T(s) = \dfrac{L(s)}{1+L(s)}$

# การสร้างโดย Python Control Library

```
In [5]: s=ctl.tf('s')
        P = 1/(10*s**2+0.1*s)
        C = 20000*(s+0.01)/(s+100)
        L = C*P
        L
```

Out[5]:
$$\frac{2 \times 10^4 s + 200}{10s^3 + 1000s^2 + 10s}$$

```
In [6]: S = 1/(1+L)
        S
```

Out[6]:
$$\frac{10s^3 + 1000s^2 + 10s}{10s^3 + 1000s^2 + 2.001 \times 10^4 s + 200}$$

```
In [9]: T=1-S
        T
```

Out[9]:
$$\frac{2 \times 10^4 s + 200}{10s^3 + 1000s^2 + 2.001 \times 10^4 s + 200}$$

Exercise:

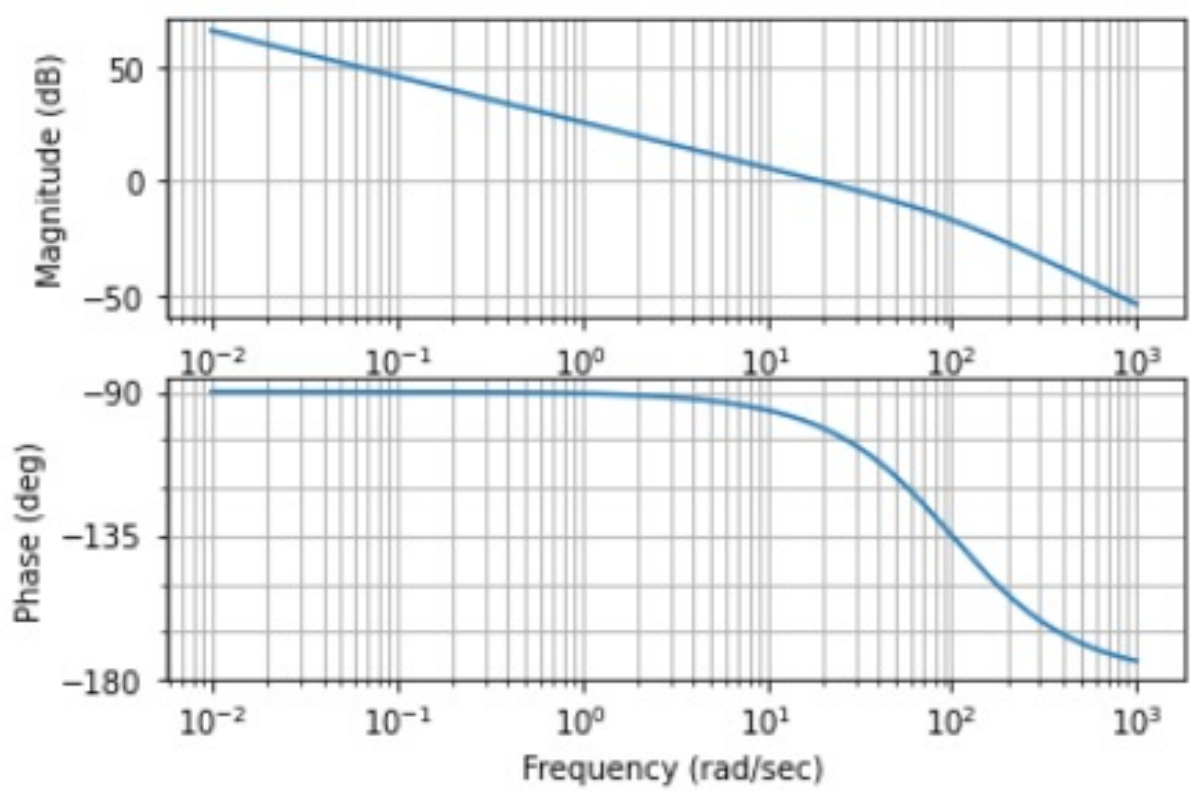
พล็อตแผนภาพโบเดของ

L, S, T

# ตัวอย่าง Bode plot ของ L(s)

Python

```
In [21]: _,_,_ = ctl.bode_plot(L,dB=True,omega_limits=(0.01,1000))
```



# การสร้างโดย Julia

```
using ControlSystems
```

```
TransferFunction{Continuous, ControlSystems.SisoRational{Float64}}
     20000.0s + 200.0
--------------------------
10.0s^3 + 1000.1s^2 + 10.0s

Continuous-time transfer function model
```

```
begin
    s = tf("s")
    P = 1/(10s^2+0.1s)
    C = 20000(s+0.01)/(s+100)
    L = C*P
end
```

```
S = TransferFunction{Continuous, ControlSystems.SisoRational{Float64}}
         10.0s^3 + 1000.1s^2 + 10.0s
    -----------------------------------
    10.0s^3 + 1000.1s^2 + 20010.0s + 200.0

    Continuous-time transfer function model
```

```
S = 1/(1+L)
```

```
S₁ = TransferFunction{Continuous, ControlSystems.SisoRational{Float64}}
                    1.0s^2 + 100.0s
    ---------------------------------------------
    1.0s^2 + 100.00000000000009s + 2000.0000000000018

    Continuous-time transfer function model
```

```
S₁ = minreal(S)
```

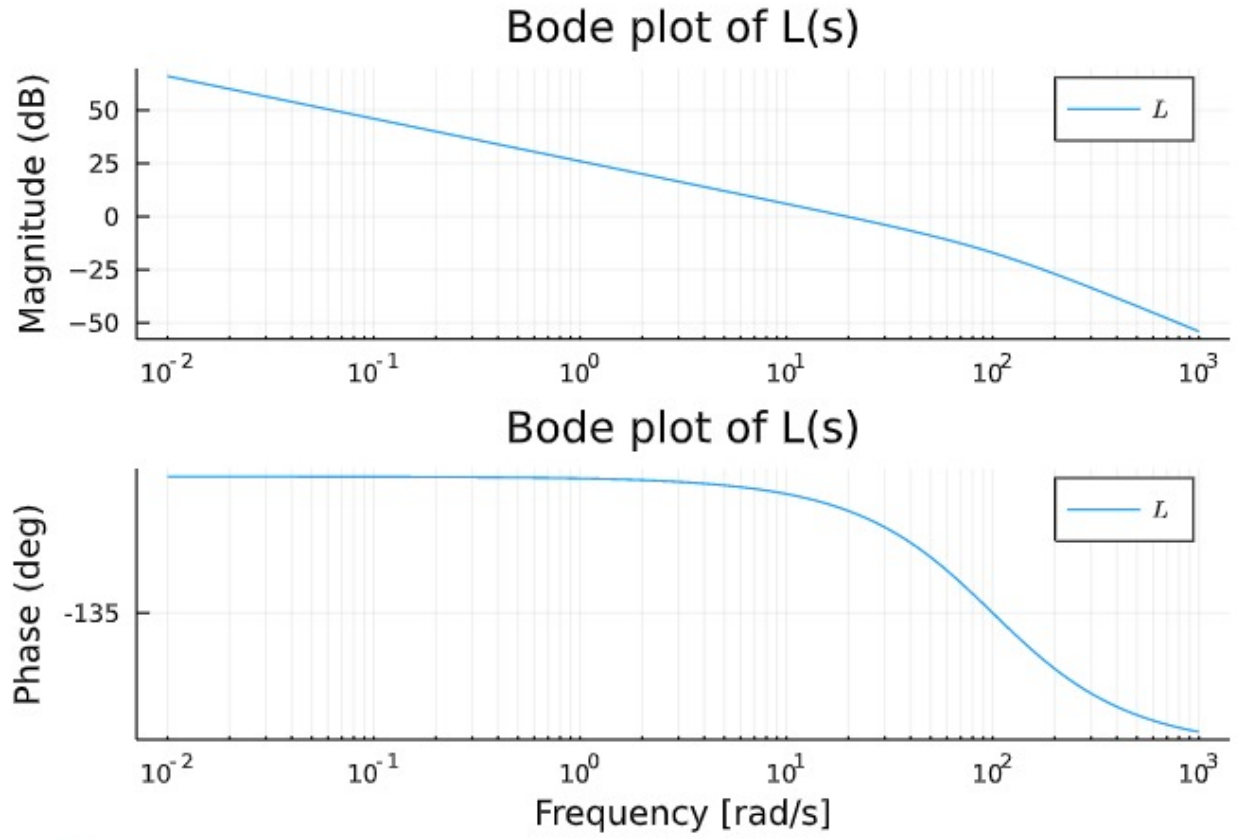
Exercise:

พล็อตแผนภาพโบเดของ

L, S, T

# ตัวอย่าง Bode plot ของ L(s)

Julia




```
begin
    setPlotScale("dB")
    w = exp10.(LinRange(-2,3,2000))
    fig = bodeplot(L,w; title="Bode plot of L(s)", label="\$L\$")
end
```

# เสถียรภาพของระบบป้อนกลับ

- คุณสมบัติการป้อนกลับ
  - เสถียรภาพ (stability)
  - สมรรถนะ (performance)
    - การตามรอย (tracking)
    - การลดทอนการรบกวน (disturbance attenuation)

# โพล (poles) และซีโร (zeros) ของฟังก์ชันถ่ายโอน

Roots ของ numerator คือ zeros

Ex. $P(s) = \dfrac{s+0.5}{10s^2+0.1s+1}$

Roots ของ denominator คือ poles

## Python

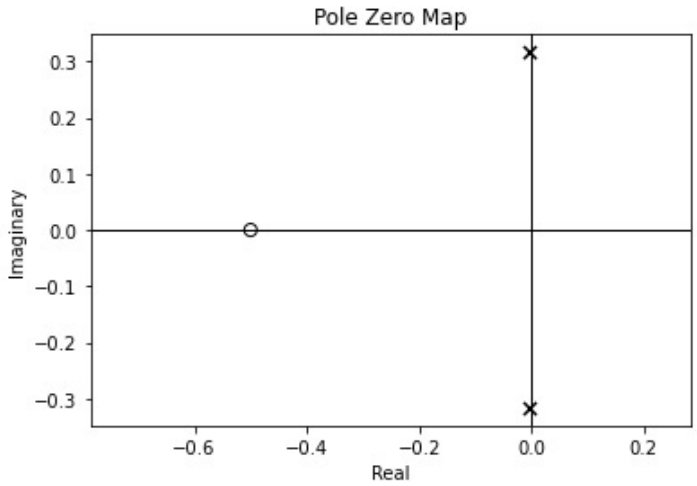
```
In [2]: s=ctl.tf('s')
        P = (s+0.5)/(10*s**2+0.1*s+1)
        P
```

Out[2]:
$$\frac{s+0.5}{10s^2+0.1s+1}$$

```
In [3]: num = [1, 0.5]
        den = [10, 0.1, 1]
        P = ctl.tf(num,den)
        P
```

Out[3]:
$$\frac{s+0.5}{10s^2+0.1s+1}$$

```
In [4]: ctl.pzmap(P)
Out[4]: (array([-0.005+0.31618824j, -0.005-0.31618824j]), array([-0.5]))
```



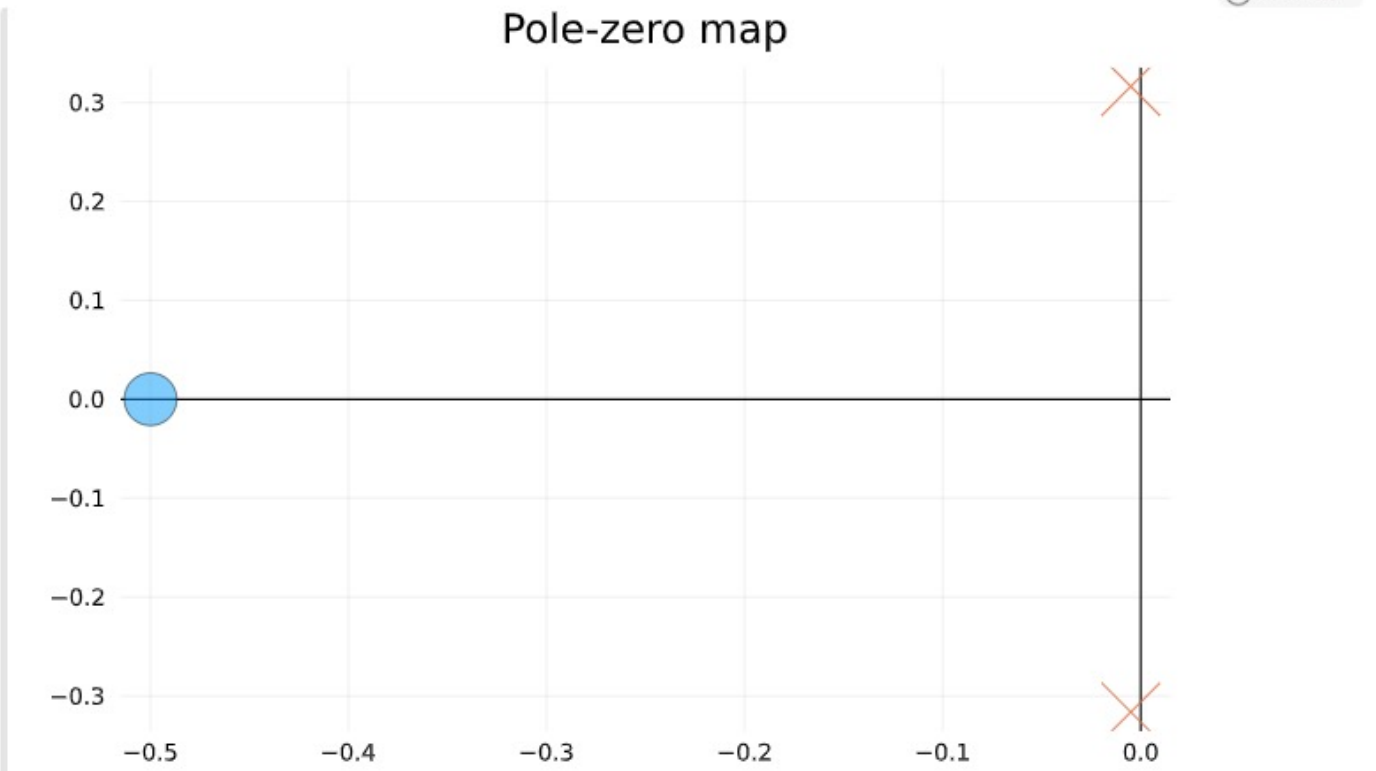
# โพล (poles) และซีโร (zeros) ของฟังก์ชันถ่ายโอน

Julia

```
TransferFunction{Continuous, ControlSystems.SisoRational{Float64}}
     1.0s + 0.5
--------------------
10.0s^2 + 0.1s + 1.0

Continuous-time transfer function model
```

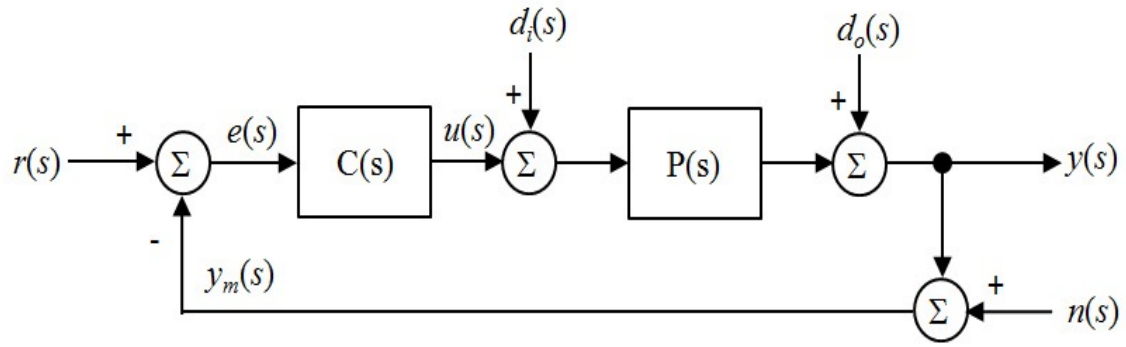
```
begin
    s = tf("s")
    P₁ = (s+0.5)/(10s^2 + 0.1s + 1)
end
```

5.1 ms




```
pzmap(P₁)
```

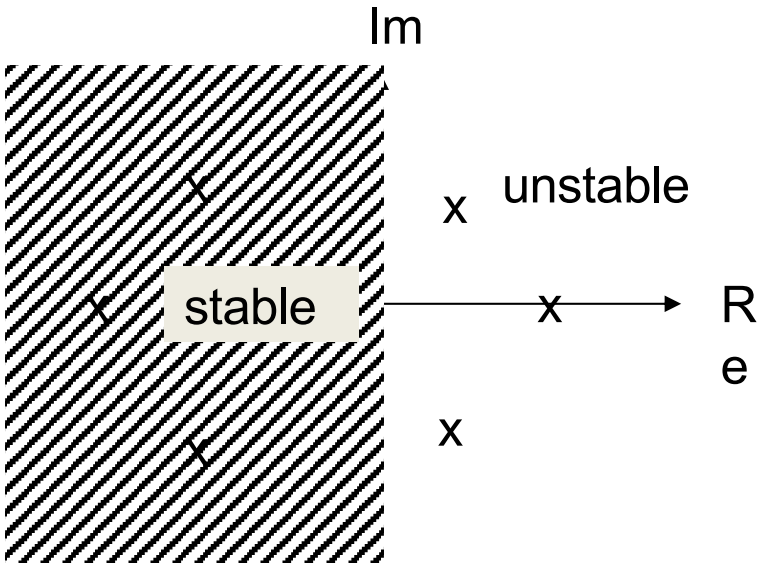
# การวิเคราะห์เสถียรภาพของระบบป้อนกลับ SISO




ตรวจสอบ poles ของฟังก์ชันถ่ายโอนวงปิด S หรือ T ว่าอยู่ทางด้านซ้ายของระนาบเชิงซ้อนหรือไม่?

Stable: โพลทุกตัวต้องอยู่ด้านซ้ายของระนาบ

Unstable: มีโพลอย่างน้อยหนึ่งตัวอยู่ทางด้านขวาของระนาบ



# ตัวอย่าง (Scilab)

```
-->s=poly(0,'s');

-->P=syslin('c',1/(10*s^2+0.1*s));

-->C=syslin('c',20000*(s+0.01)/(s+100));

-->L=C*P;

-->T=L/(1+L)
 T  =

          2000
    --------------
                   2
    2000 + 100s + s

-->roots(T.den)
 ans  =

  - 72.36068
  - 27.63932
```

Stable

```
-->P=syslin('c',1/(10*s^2+0.1*s));

-->C=syslin('c',20000*(s+100)/(s+0.01));

-->L=C*P;

-->T=L/(1+L)
 T  =

             200000 + 2000s
    -------------------------------
                                  2    3
    200000 + 2000.0001s + 0.02s + s

-->roots(T.den)
 ans  =

    23.618224 + 60.616881i
    23.618224 - 60.616881i
  - 47.256447
```

Unstable

# ตัวอย่าง (Python)

```
In [1]: import control as ctl

In [7]: s = ctl.tf("s")
        P = 1/(10*s**2 + 0.1*s)
        C1 = 20000*(s+0.01)/(s+100)
        L = C1*P
        T = L/(1+L)
        T = ctl.minreal(T)
        T

        4 states have been removed from t
```

Out[7]:

$$\frac{2000}{s^2 + 100s + 2000}$$

```
In [8]: ctl.pole(T)
Out[8]: array([-72.36067977, -27.63932023])
```

```
In [9]: s = ctl.tf("s")
        P = 1/(10*s**2 + 0.1*s)
        C2 = 20000*(s+100)/(s+0.01)
        L = C2*P
        T = L/(1+L)
        T = ctl.minreal(T)
        T

        3 states have been removed from the model
```

Out[9]:

$$\frac{2000s + 2 \times 10^5}{s^3 + 0.02s^2 + 2000s + 2 \times 10^5}$$

```
In [10]: ctl.pole(T)
Out[10]: array([ 23.61822354+60.61688114j,  23.61822354-60.61688114j,
                -47.25644708 +0.j        ])
```

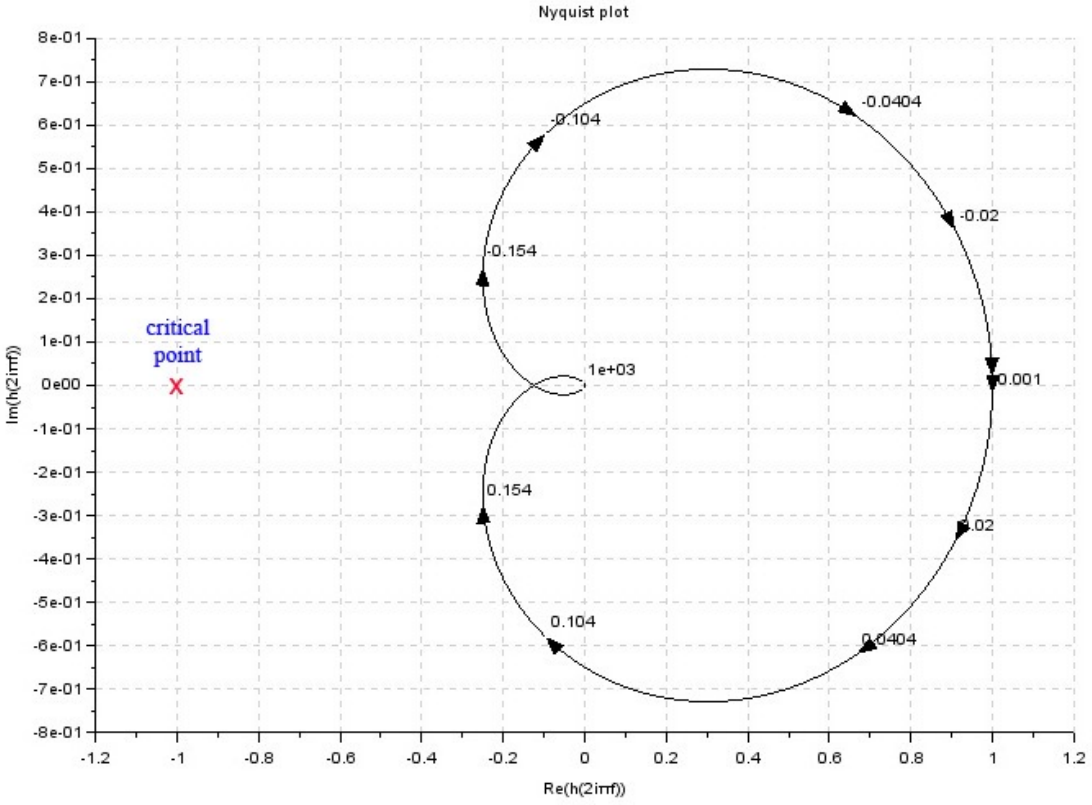
Stable

Unstable

# การตรวจสอบเสถียรภาพจาก Nyquist Plot

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

```
-->P = syslin('c',1/(1+s)^3)
 P  =
            1
    ---------------
                2    3
    1 + 3s + 3s + s
-->nyquist(P)
```




use ctl.nyquist_plot(P) in Python

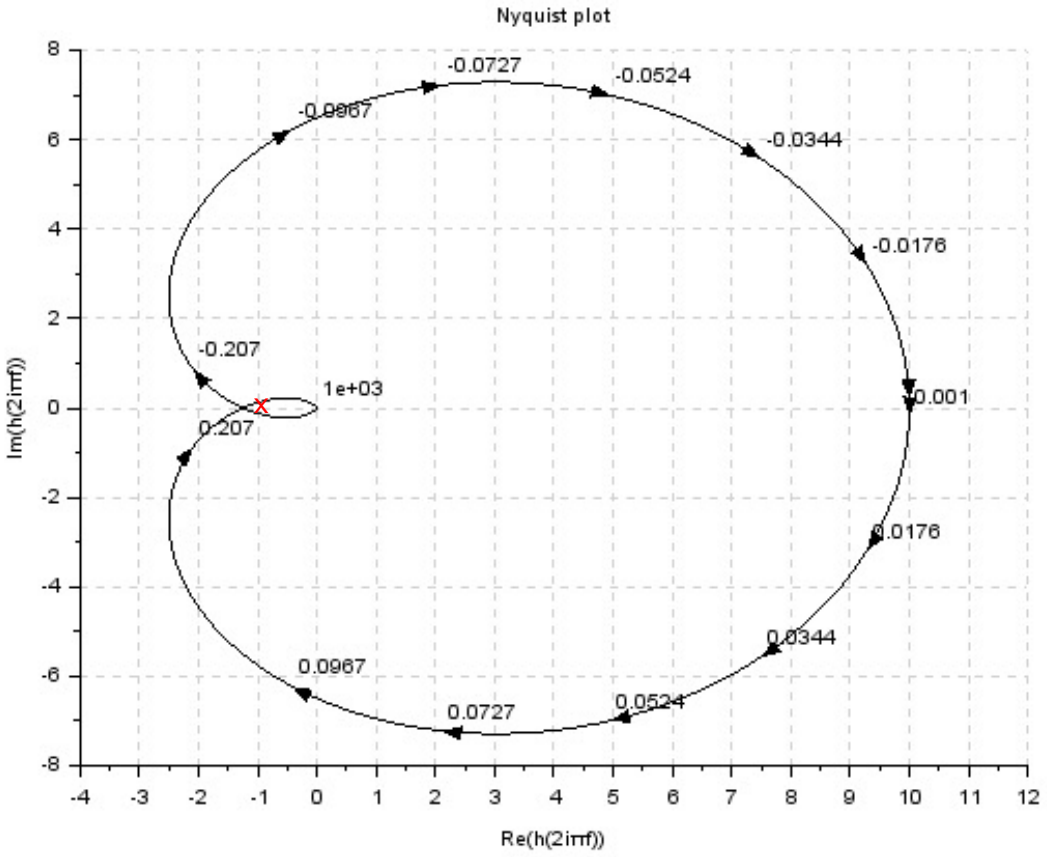
In Julia

```
w = LinRange(1,100,1000)
nyquistplot(P,w)
```

# Nyquist Plot ของระบบไม่เสถียร

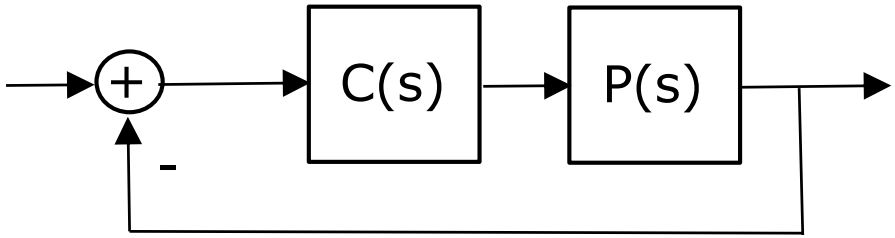
$$P(s) = \frac{1}{(s+1)^3}$$

```
-->P = syslin('c',1/(1+s)^3)
 P  =
             1
    ---------------
                2    3
    1 + 3s + 3s + s
-->nyquist(10*P)
```



# Exercise 2.1 :

กำหนดให้ $P(s) = \frac{1}{(s+1)^3}$




$C(s) = 7 + \frac{1}{s}$

ระบบป้อนกลับ stable หรือไม่?

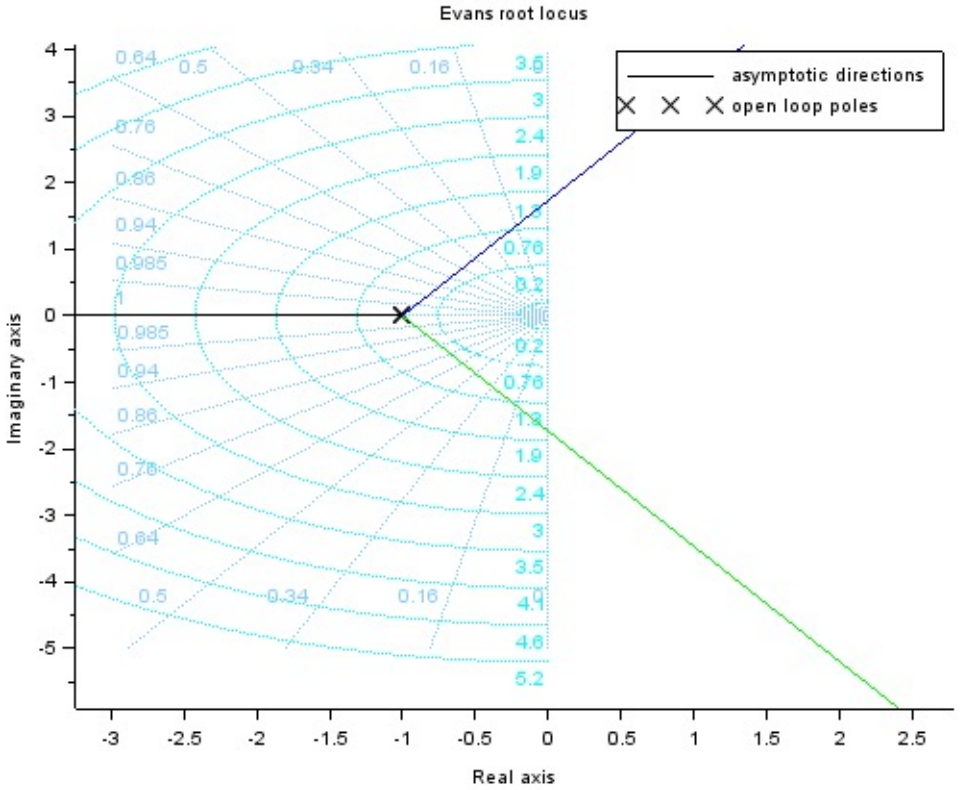
วิธีการที่ทำได้:

1. ตรวจสอบโพลของ T (หรือ S)
2. จำลองผลตอบสนองของระบบวงปิด เช่นผลตอบสนองขั้นบันได
3. ตรวจสอบ nyquist plot ของ L(s)

# วิธี Root Locus

## Scilab code

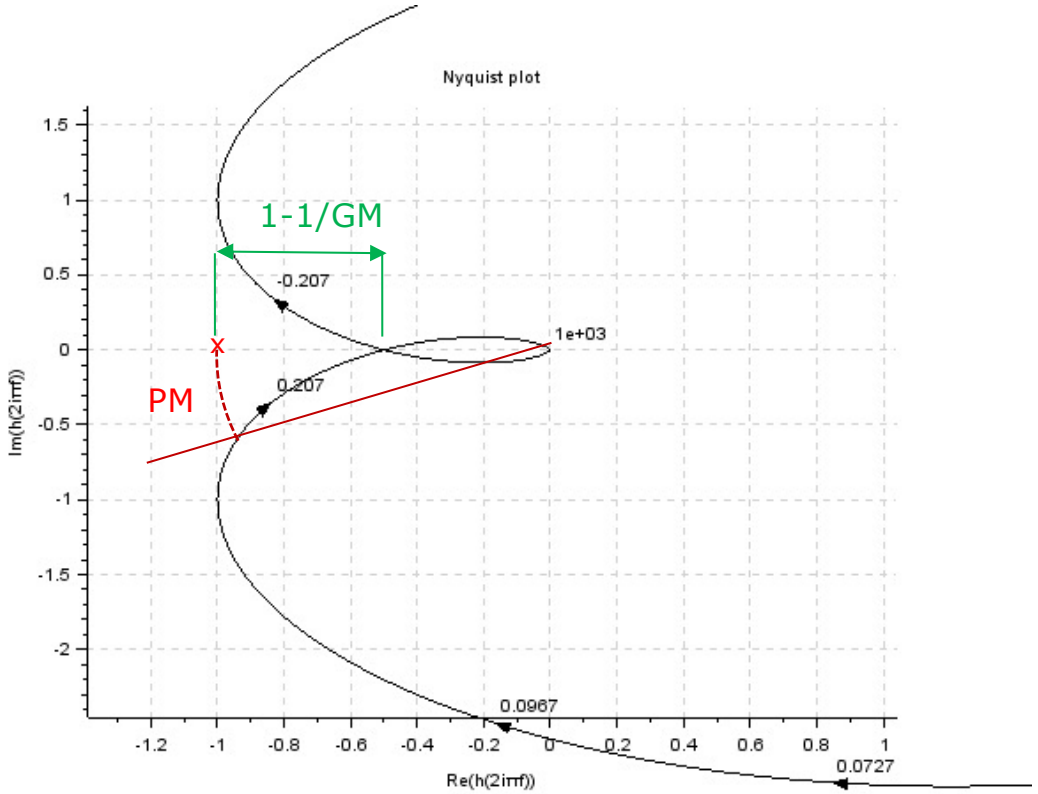
```
-->evans(P)

-->sgrid

-->[kmax,s]=kpure(P)
 s   =

    1.7320508i
 kmax   =

    8.
```



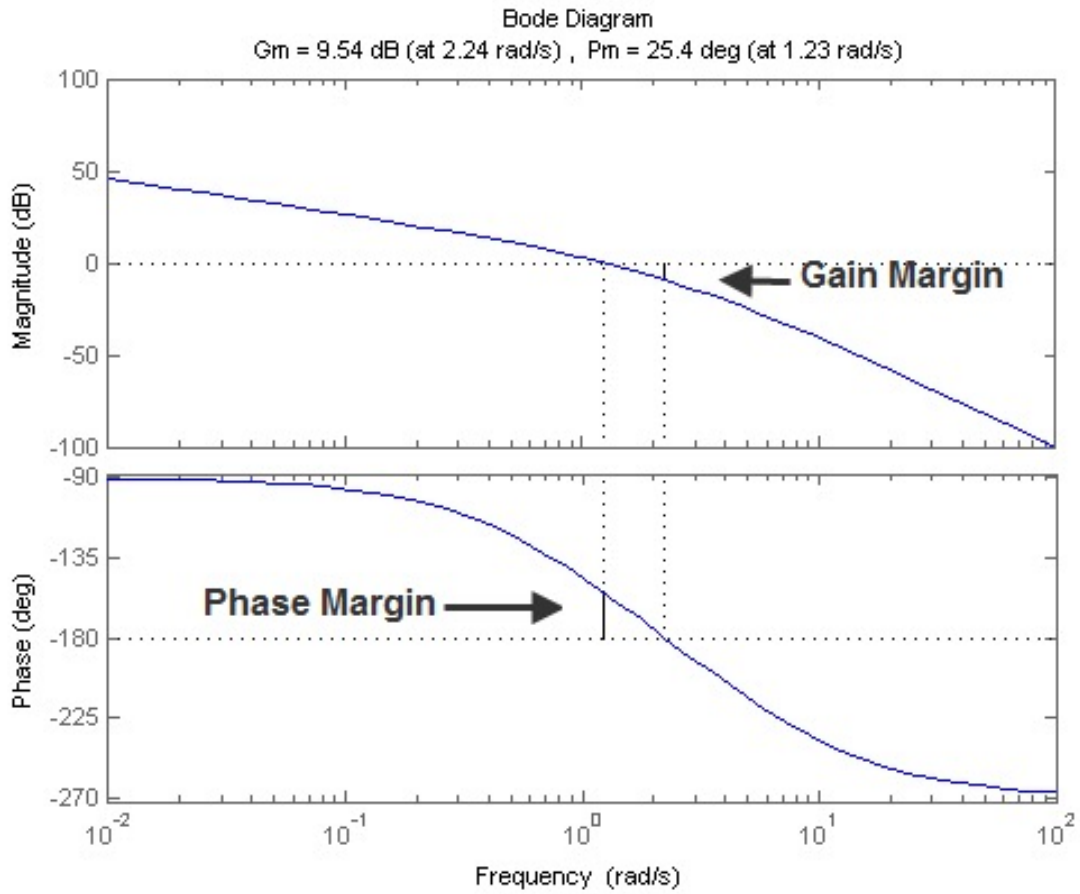

เมื่อเพิ่มอัตราขยายมากกว่า 8 จะทำให้ระบบป้อนกลับเสียเสถียรภาพ

# ค่าเผื่อเสถียรภาพ (stability margins)

- ค่าเผื่ออัตราขยาย (gain margin)
- ค่าเผื่อเฟส (phase margin)



# Gain and phase margin in Bode plot

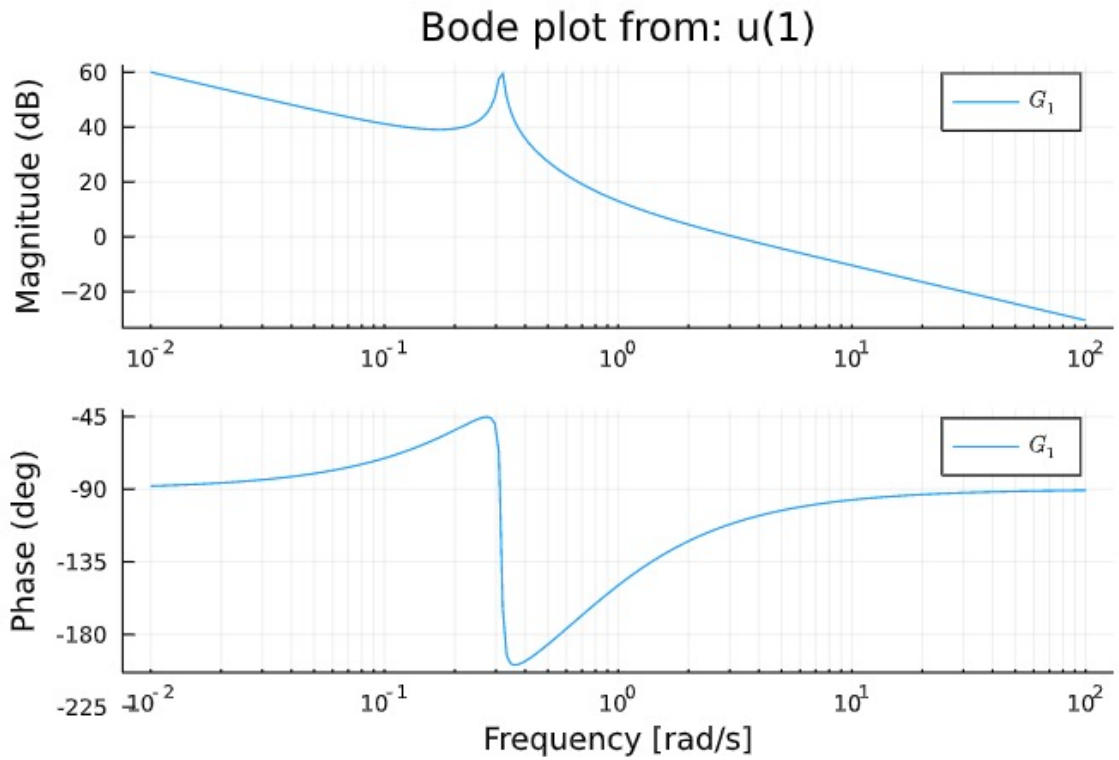



https://la.mathworks.com/help/sldo/ug/frequency-domain-design-requirements-gui.html

# margin() ใน Julia

```
(1×1 Matrix{Float64}:, 1×1 Matrix{Float64}:, 1×1 Matrix{Float64}:, 1×1 Matrix{Float64}:
 0.327748              0.00199509            3.13828               69.1389
```

```
begin
    #s = tf("s")
    Psys = (s+0.5)/(10s^2 + 0.1s + 1)
    Csys = 30 + 20/s
    Lsys = Csys*Psys
    wgm, g_margin,wpm_ph_margin = margin(Lsys)
end
```

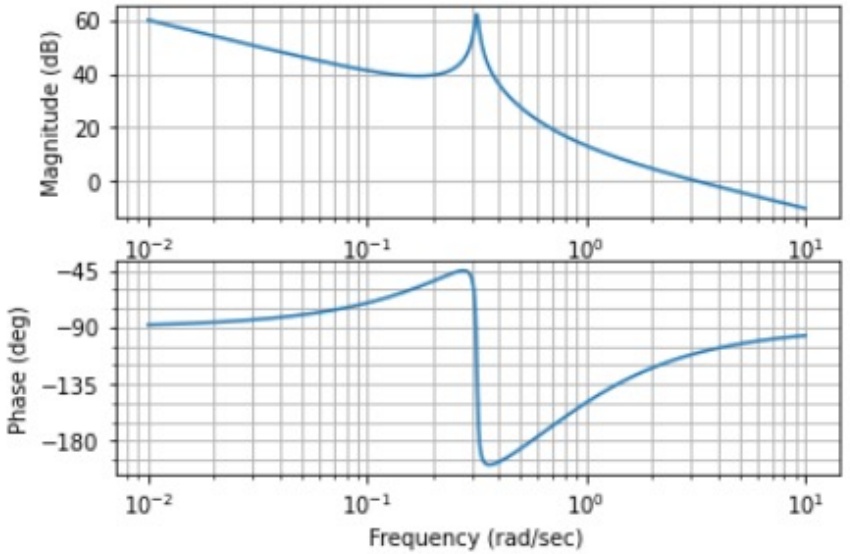



```
bodeplot(Lsys)
```

Question :
Is this system stable or not?

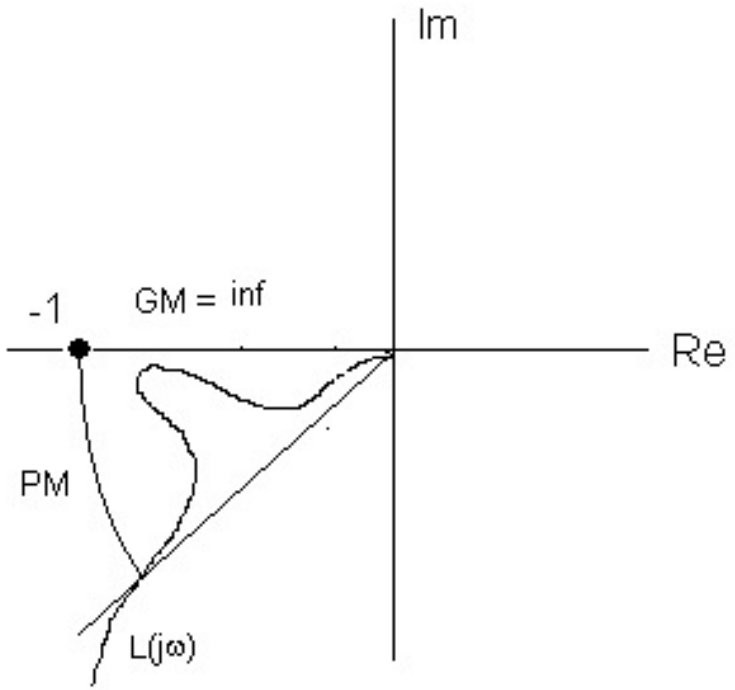
# control.margin() ใน Python

```
In [24]: Psys = (s+0.5)/(10*s**2 + 0.1*s + 1)
         Csys = 30 + 20/s
         Lsys = Csys*Psys
         ctl.margin(Lsys) # gm, pm, wgm, wpm

Out[24]: (0.06179206595928491, 69.13404027808036, 0.5623808592559825, 3.137531095981345)

In [29]: _,_,_ =ctl.bode_plot(Lsys,dB=True)
```

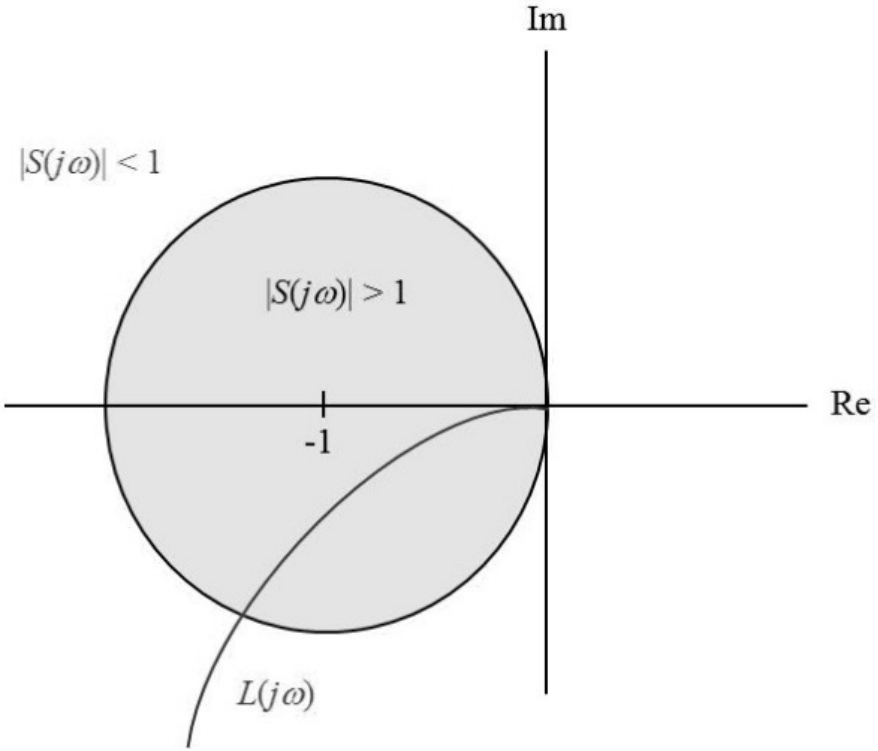



Question :
why gain margin differs in Python and Julia?

# ข้อควรระวังเกี่ยวกับค่าเผื่อเสถียรภาพ



# เงื่อนไขบังคับด้านเสถียรภาพ

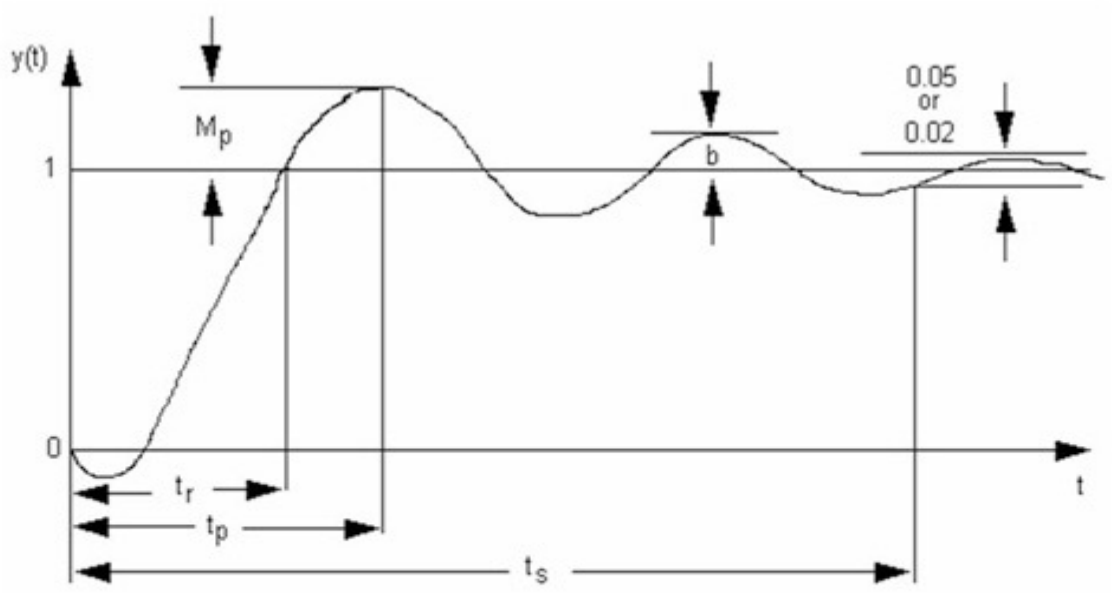



สรุป: ไม่ต้องการให้ |S(jw)| มีค่ายอดสูง

# สมรรถนะของระบบป้อนกลับ

- คุณสมบัติการป้อนกลับ
  - เสถียรภาพ (stability)
  - สมรรถนะ (performance)
    - การตามรอย (tracking)
    - การลดทอนการรบกวน (disturbance attenuation)

# สมรรถนะการตามรอย (tracking performance)



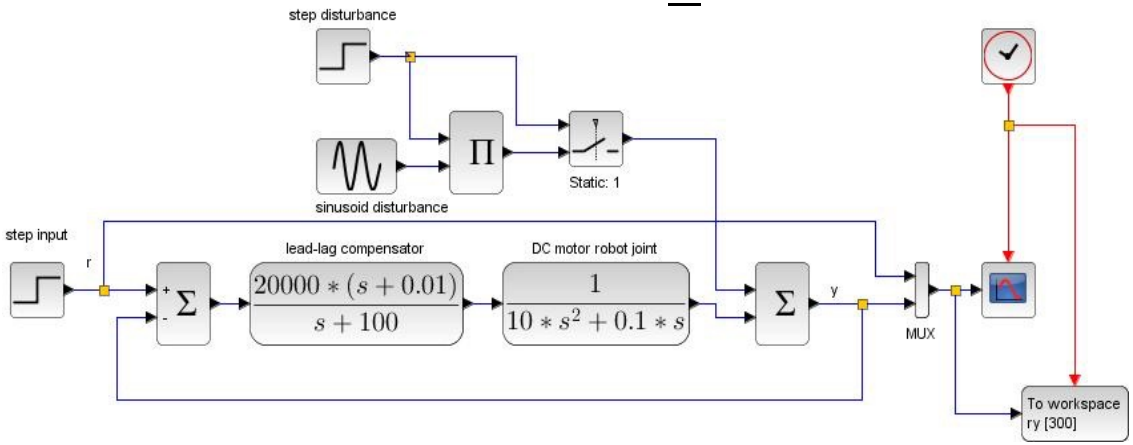

$t_r$ = rise time
$t_p$ = peak time
$t_s$ = settling time

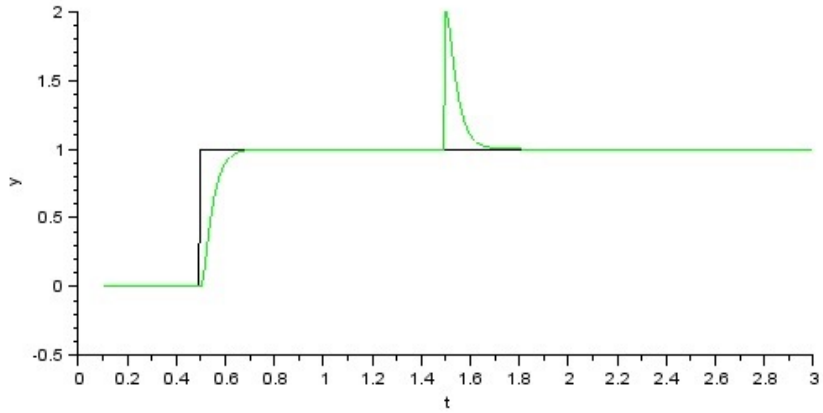
$M_p$ = percent overshoot
d = Mp /b (decay ratio)

ผลตอบสนองต่อสัญญาณขั้นบันได (step response)

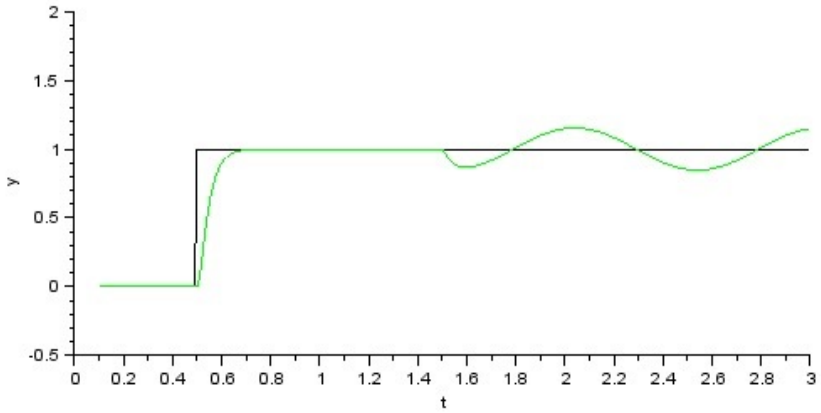
# สมรรถนะการลดทอนการรบกวน (disturbance attenuation performance)

feedback_wdist.zcos




ผลตอบสนองต่อการรบกวนขั้นบันได
ที่เอาต์พุตของพลานต์

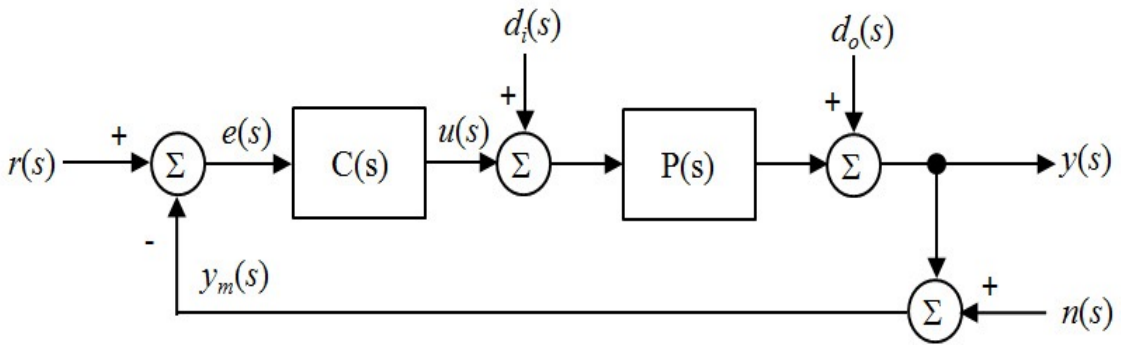



ผลตอบสนองต่อการรบกวนรูปคลื่นไซน์
ที่เอาต์พุตของพลานต์

# Homework # 2

จำลองผลตอบสนองการขจัดการรบกวนรูปคลื่นไซน์ที่เอาต์พุต
ของระบบป้อนกลับในสไลด์ที่ผ่านมา โดยใช้ Python หรือ Julia
(เพื่อความง่าย ให้รูปคลื่นไซน์เริ่มต้นที่ $t = 0$)

# แผนภาพทั่วไปของระบบป้อนกลับ SISO

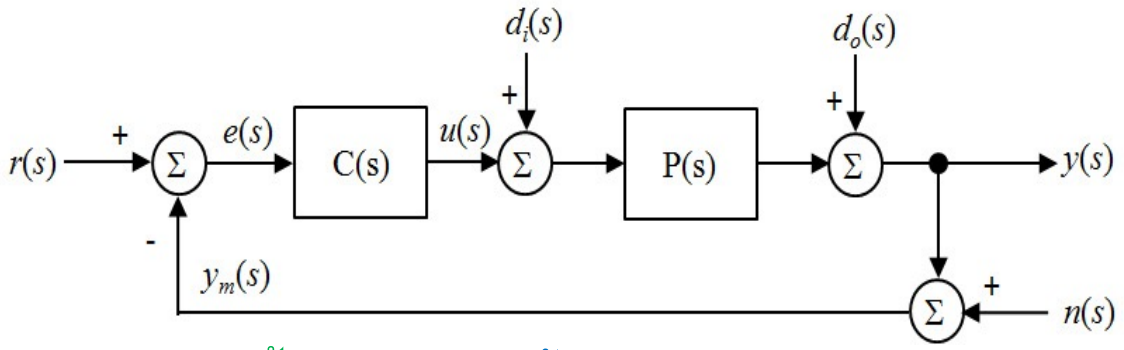



$$y(s) = S(s)d_o(s) + T(s)(r(s) - n(s)) - S(s)P(s)d_i(s)$$

$$e(s) = S(s)(r(s) - n(s) - d_o(s) - P(s)d_i(s))$$

$$u(s) = C(s)S(s)(r(s) - n(s) - d_o(s)) + T(s)d_i(s)$$

# เงื่อนไขบังคับด้านสมรรถนะ




$$y(s) = S(s)d_o(s) + T(s)(r(s) - n(s)) - S(s)P(s)d_i(s)$$

$$e(s) = S(s)(r(s) - n(s) - d_o(s) - P(s)d_i(s))$$

ตามรอย น้อย

สรุป: ต้องการให้ S(s) และ T(s) มีค่าน้อย

# เงื่อนไขบังคับเชิงพีชคณิต (Algebraic Constraints)

$$S(s) = \frac{1}{1 + L(s)} \qquad T(s) = \frac{L(s)}{1 + L(s)}$$

$$S + T = 1$$