2021.10 Y. Minami

Python-Control *Cheat Sheet*

```
from control.matlab import *
import numpy as np
```

1 System representation

Transfer Function

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

```
s = tf('s')
P = 4/(s**2 + 2*s + 3)
```

State-Space Equation

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = x(t)$$

```
A = [[0, 1],[-4, -5]]
B = [[0], [1]]
C = np.eye(2)
D = np.zeros([2, 1])
P = ss(A, B, C, D)
```

▶ Controllability and Reachability Matrix

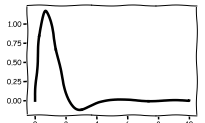
```
Vc = ctrb(P.A, P.B)
Vo = obsv(P.A, P.C)
```

2 Time response

Impulse response

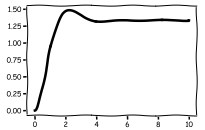
```
T = np.arange(0, 10, 0.01)
y, t = impulse(P, T)
```

0, 0.01, 0.02,...,9.99



Step response

```
T = np.arange(0, 10, 0.01)
y, t = step(P, T)
```

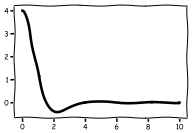


▶ Step response characteristics

Info = stepinfo(sys)

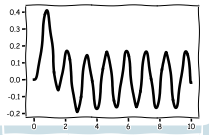
Initial condition response

```
T = np.arange(0, 10, 0.01)
X0 = [0, -1]
y, t = initial(P, T, X0)
```



Forced response

```
T = np.arange(0, 10, 0.01)
U = np.sin(5*T)
y, t, x0 = lsim(P, U, T)
```



3 Frequency response

Bode diagram $P(j\omega) = \alpha(\omega)e^{j\beta(\omega)}$

$10^{-2} \sim 10^{2}$

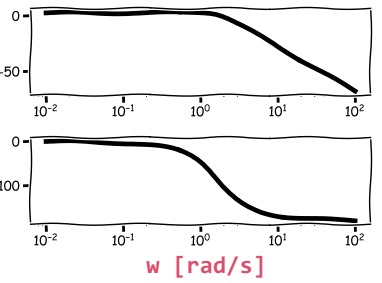
```
gain, phase, w = bode(P, logspace(-2,2))
```

Gain $20\log_{10}|\alpha(\omega)|$

```
20*np.log10(gain) [dB]
```

Phase $\beta(\omega) \times \frac{180}{\pi}$

```
phase*180/np.pi [deg]
```

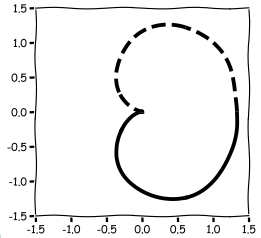


▶ Frequency response at multiple angular frequencies

gain, phase, w = freqresp(sys, [omega])

Nyquist diagram $P(j\omega) = x(\omega) + jy(\omega)$

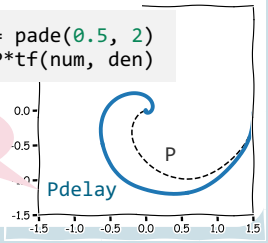
```
x, y, w = nyquist(P, logspace(-2,2,1000))
```



Pade approximation

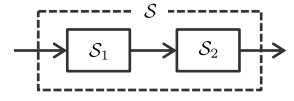
```
num, den = pade(0.5, 2)
Pdelay = P*tf(num, den)
```

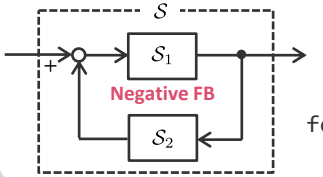
Delay = **0.5**[s]
2nd order appr.



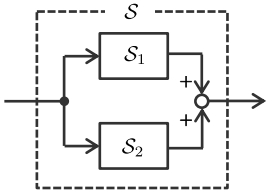
4 Block diagram

series(S1, S2)





parallel(S1, S2)



feedback(S1, S2, sign=-1)

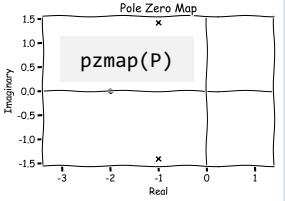
▶ minreal(sys)

5 Stability and Robustness

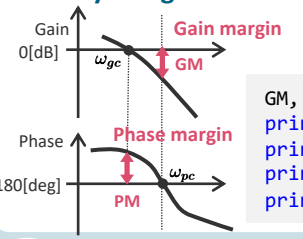
Pole & Zero

```
P.pole()
P.zero()
```

```
pzmap(P)
```

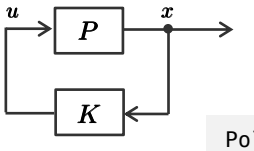


Stability margin




```
GM, PM, wpc, wgc = margin(P)
print('GM=', 20*np.log10(GM))
print('PM=', PM)
print('wpc=', wpc)  Phase crossover
print('wgc=', wgc)  Gain crossover
```

6 Controller design



$$P : \dot{x}(t) = Ax(t) + Bu(t)$$

$$K : u(t) = Fx(t)$$

Pole placement

```
Pole = [-1, -2]
F = -acker(P.A, P.B, Pole)
```

```
F = -place(P.A, P.B, Pole)
```

LQ optimal control

$$J = \int_0^\infty \{x^\top Qx + u^\top Ru\}\, dt$$

```
Q = [ [100, 0], [0, 1]]
R = 1
F, _, _ = lqr(P.A, P.B, Q, R)
F = -F
```

▶ solves the continuous-time algebraic Riccati equation

(X, L, G) = care(A, B, Q, R)

Mixed sensitivity design

```
from control import mixsyn
K, _, gamma = mixsyn(Sys, w1=WS , w2=WU, w3=WT)
```

7 Digitalization

Zero Order Hold

```
ts = 0.2
Pd = c2d(P, ts, method='zoh')
```

Tustin Transformation

```
ts = 0.2
Pd = c2d(P, ts, method='tustin')
```