Advanced Dynamical Systems Control
動的システム制御論
#3

ver. 0.05

禁無断転載

Do not redistribute

Schedule
授業の予定

- Brief review of “modern control theory”
  現代制御総復習
- Basic idea of sample-data control
  サンプル値制御理論の考え方
- Continuous-time systems and discrete-time systems
  連続時間システムと離散時間システム
- Stability of discrete-time linear systems
  離散時間線形システムの安定性
- Multi-rate sampling systems
  マルチレートサンプリング系
- Design example of sample-data control systems
  サンプル値制御系の設計例
- Quantization errors and their solution
  量子化誤差とその対策
- Implementation of sample-data systems
  サンプル値制御系の実装

Reachability

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k]$$

Definition:

A linear system is called *reachable*
if for any $x_0,\ x_f \in \mathbb{R}^n$ there exist a $T > 0$ and $u : [0,\ T] \to \mathbb{R}$
such that if $x(0) = x_0$ then the corresponding solution satisfies $x(T) = x_f$

$\mathscr{R}_n = [B\ AB\ A^2B\ \cdots\ A^{n-1}B]$: Reachability matrix ($A \in \mathbb{R}^{n\times n}$)

$(A,\ B)$ is reachable $\Leftrightarrow \mathrm{rank}\mathscr{R}_n = n$

$\Leftrightarrow \mathrm{rank}[zI - A\ \ B] = n\ (\forall z \in \mathbb{C})$

Note:
In continuous-time linear systems case, Reachability = Controllability

Observability

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k]$$

A linear system is called *observable*
if for every $T > 0$ it is possible to determine the state of the system $x(T)$
through measurements of $y(t)$ and $u(t)$ on the interval $[0,\ T]$

$$\mathcal{O}_n = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$: Observability matrix ($A \in \mathbb{R}^{n\times n}$)

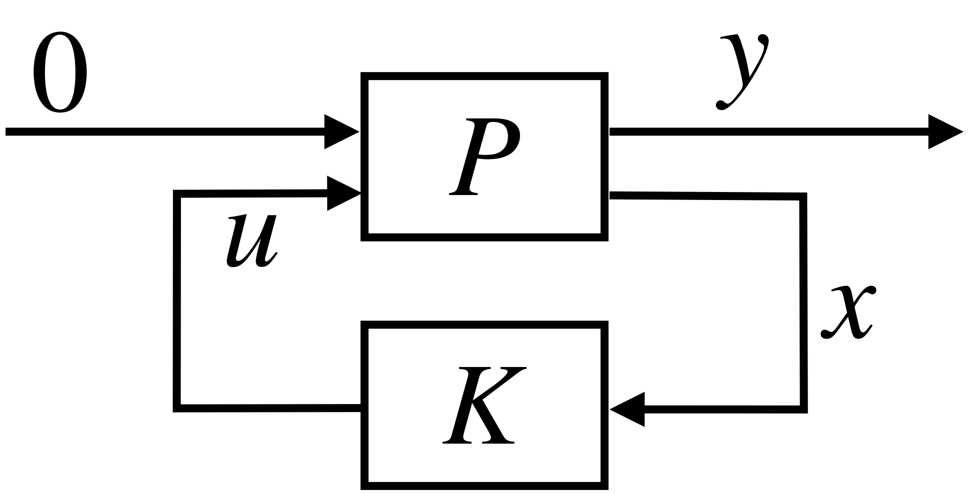
$(C,\ A)$ is reachable $\Leftrightarrow \mathrm{rank}\mathcal{O}_n = n$

$$\Leftrightarrow \mathrm{rank} \begin{bmatrix} zI - A \\ B \end{bmatrix} = n \ (\forall z \in \mathbb{C})$$

Pole assignment

One of the design method of control systems to be stable

Example



$$P: \begin{cases} x[k+1] = \begin{bmatrix} 3.5 & 2 \\ -3 & -1.5 \end{bmatrix} x[k] + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u[k] \\ y[k] = [1 \ \ -1] x[k] \end{cases}$$

$$K: \ u[k] = -Fx[k]$$

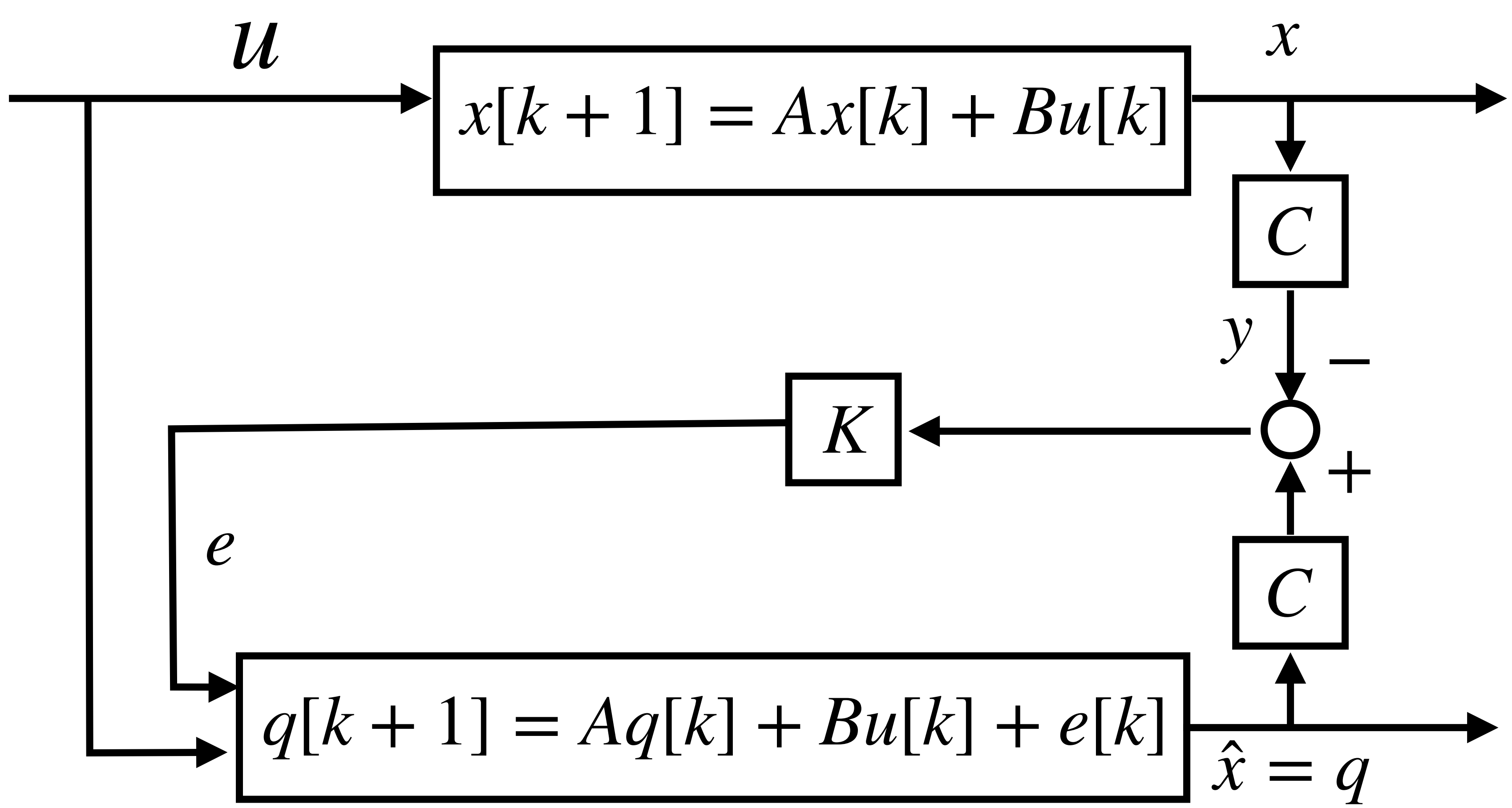
poles of P : 1.5, 0.5 (unstable)

To be the FB system stable where poles are $0.8e^{\pm j}$

$$F = [0.4449 \ 0.2457]$$

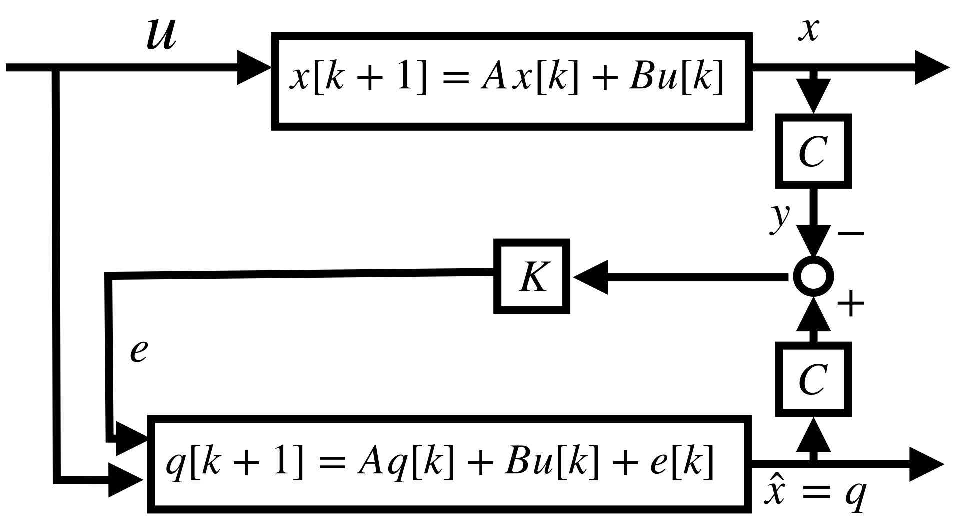
Observer

Estimate the states of the system



Observer

Full order observer



$$\epsilon[k+1] = \hat{x}[k+1] - x[k+1]$$

$$= Aq[k] + Bu[k]$$
$$-K(y[k] - Cq[k]) - (Ax[k] + Bu[k])$$

$$= A(q[k] - x[k]) - K(Cx[k] - Cq[k])$$

$$= (A + KC)(q[k] - x[k])$$

$$= (A + KC)\epsilon[k]$$

$$= (A + KC)\epsilon[k]$$

To be the poles of $A + KC$ stable, the state can be estimated.

To use the same algorithm as controller design,
we can consider $A^T + C^T K$ instead.

Code Example

MATLAB

```
clear

A=[
    3.5 2;
    -3 -1.5
    ];
B=[2;1];
C=[1 -1];
sys1=ss(A,B,C,0,0.01);

% pole placement with [0.8exp(-i) 0.8exp(i)]
F=place(A,B,0.8*[exp(-1i) exp(1i)]);
sys2=ss(A-B*F,B,C,0,0.01);

% pole placement with [0.5exp(-i) 0.5exp(i)]
F2=place(A,B,0.5*[exp(-1i) exp(1i)]);
sys3=ss(A-B*F2,B,C,0,0.01);

figure(1)
subplot(3,1,1)
step(sys1);
subplot(3,1,2)
step(sys2)
subplot(3,1,3)
step(sys3)

ob=[C;C*A]; %observability matrix
disp(ob)

% pole placement for observer gain
K=place(A',C',[0.4, 0.6]);
```

```
t=0:0.01:1;
x=zeros(2,102); % true state
x(:,1)=[1;0];
q=zeros(2,102); % estimated state
y=zeros(1,101);
u=zeros(1,101);
for k=1:100
    y(k)=C*x(:,k);
    u(k)=-F*q(:,k);
    x(:,k+1)=A*x(:,k)+B*u(k);
    q(:,k+1)=A*q(:,k)+B*u(k)-K'*(C*q(:,k)-y(k));
end
e=x-q; % estimation error

figure(2)
subplot(2,1,1)
stairs(t(1:51),x(:,1:51)')
hold on
stairs(t(1:51),q(:,1:51)')
legend('x1','x2','q1','q2')
xlabel('t')
grid on
hold off

subplot(2,1,2)
stairs(t(1:51),e(:,1:51)')
xlabel('t')
grid on
```

Code Example

Python (Jupyter Notebook)

```
# !pip install control

import numpy as np
from control.matlab import *
import matplotlib.pyplot as plt
```

```
A=np.array([[3.5, 2], [-3, -1.5]])
B=np.array([ [2] , [1] ] )
C=np.array([[ 1 , -1 ]])
sys1=ss(A,B,C,0,0.01);
print(sys1)
```

```
F1=place(A, B, [0.8*np.exp(-1j) ,  0.8*np.exp(1j)] );
sys2=ss(A-B*F1, B, C, 0, 0.01);
print(sys2)

F2=place(A, B, [0.5*np.exp(-1j) ,  0.5*np.exp(1j)] );
sys3=ss(A-B*F2, B, C, 0, 0.01);
print(sys3)
```

```
t=np.arange(0,0.2,0.01)

y1,t1=step(sys1,t)
y2,t2=step(sys2,t)
y3,t3=step(sys3,t)

fig, ax = plt.subplots(3,1)
ax[0].plot(t1,y1)
ax[0].set_xlim(0,0.2)
ax[1].plot(t2,y2)
ax[1].set_xlim(0,0.2)
ax[2].plot(t3,y3)
ax[2].set_xlim(0,0.2)
```

```
ob=np.array([C,np.dot(C,A)])
print(ob)
```

```
K=place(A.T,C.T,[0.4, 0.6]);
print(K)
```

```
y=[]
u=[]
x=np.array([[1],[0]])
q=np.array([[0],[0]])

for k in range(100):
  y.append(np.dot(C,x[:,k].reshape(2,1)))
  u.append(np.dot(-F1,q[:,k].reshape(2,1)))
  x=np.append(x,np.dot(A,x[:,k].reshape(2,1))+np.dot(B,u[k]), axis=1)
  q=np.append(q,np.dot(A,q[:,k].reshape(2,1))+np.dot(B,u[k])-np.dot(K.T,(np.dot(C,q[:,k].reshape(2,1))-y[k])) ,axis=1)
```

```
e=x-q
t=np.arange(0,1.01,0.01)
```

```
fig2, ax2 = plt.subplots(2,1)
ax2[0].plot(t, x.T, label='x')
ax2[0].plot(t, q.T, label='q')
ax2[0].set_xlim(0,0.5)
ax2[0].legend(loc='lower right',ncol=2)
ax2[0].set_ylabel('x,q')
ax2[1].plot(t,e.T)
ax2[1].set_xlim(0,0.5)
ax2[1].set_xlabel('t')
ax2[1].set_ylabel('e')
```