国家精品课程/ 国家精品资源共享课程/ 国家级精品教材
国家级十一(二)五规划教材/ 教育部自动化专业教学指导委员会牵头规划系列教材

控制系统计算机辅助设计

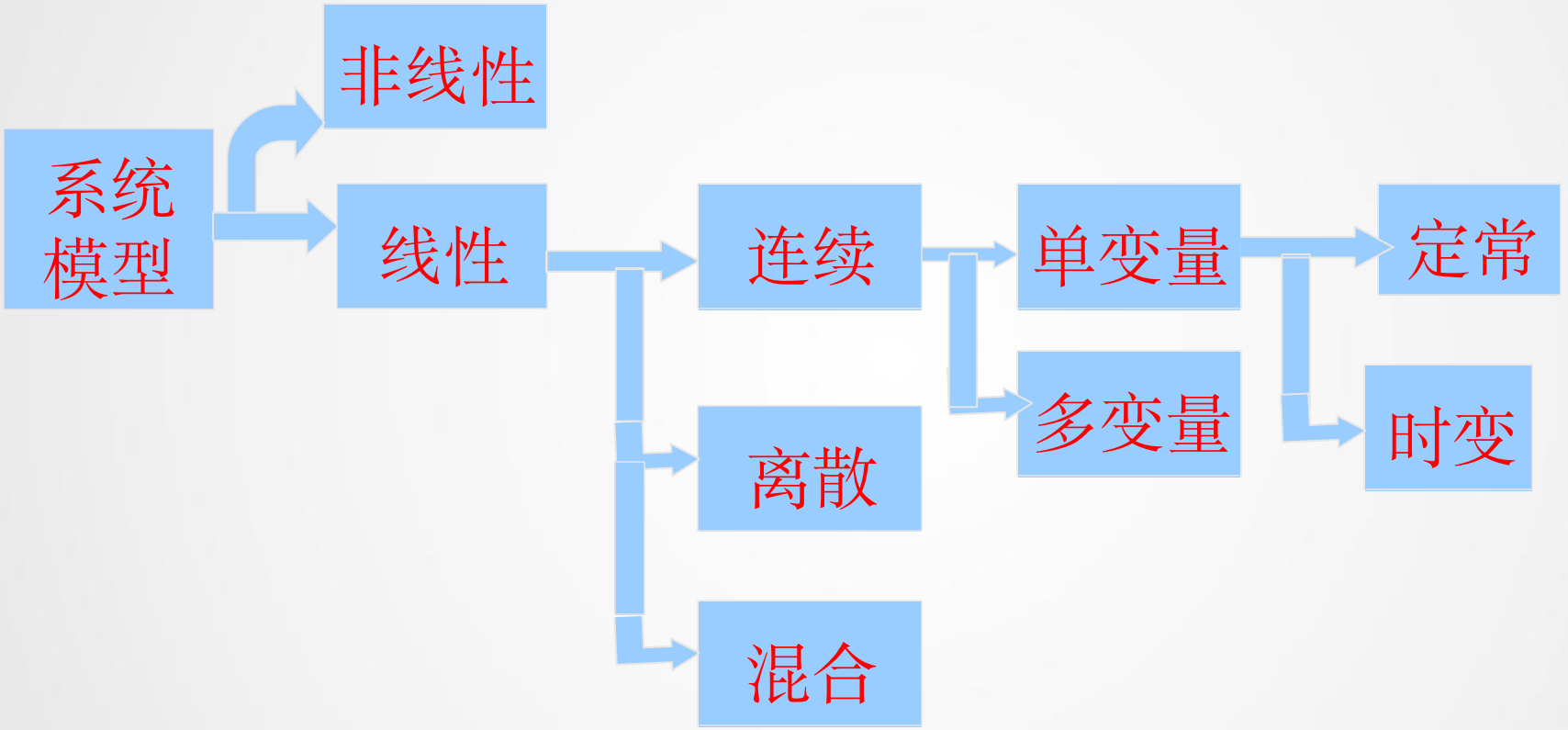
第4章 线性控制系统的数学模型

主讲：修贤超

线性系统的数学模型

- 4.1 线性连续系统模型及MATLAB表示
- 4.2 线性离散时间系统的数学模型
- 4.3 线性模型的相互转换
- 4.4 方框图描述系统的化简
- 4.5 线性系统的模型降阶
- 4.6 线性系统的模型辨识

系统数学模型的分类



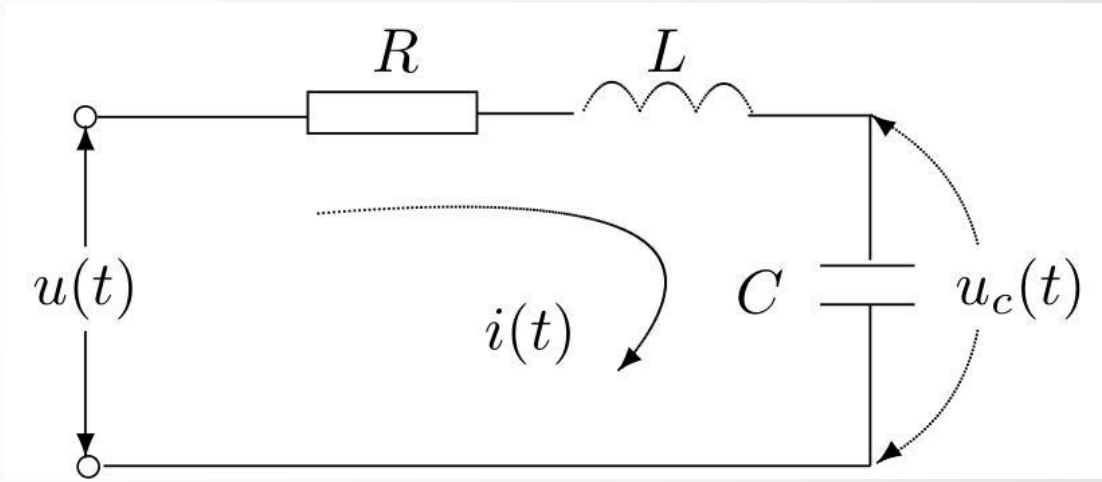
- 传递函数、状态方程、时间延迟、采样周期

例4-1 电路的数学模型

- RLC电路的微分方程模型
  - Kirchhoff定律

$$Ri(t)+L\frac{\mathrm{d}i(t)}{\mathrm{d}t}+u_c(t)=u(t)$$

$$i(t)=C\frac{\mathrm{d}u_c(t)}{\mathrm{d}t}$$



  - 变换成单一的微分方程

$$LC\frac{\mathrm{d}^2u_c(t)}{\mathrm{d}t^2}+RC\frac{\mathrm{d}u_c(t)}{\mathrm{d}t}+u_c(t)=u(t)$$

线性连续系统模型及MATLAB表示

- 线性系统的常系数线性常微分方程模型

$$a_1\frac{\mathrm{d}^n y(t)}{\mathrm{d}t^n}+a_2\frac{\mathrm{d}^{n-1} y(t)}{\mathrm{d}t^{n-1}}+\cdots+a_n\frac{\mathrm{d}y(t)}{\mathrm{d}t}+a_{n+1}y(t)$$
$$=b_1\frac{\mathrm{d}^m u(t)}{\mathrm{d}t^m}+b_2\frac{\mathrm{d}^{m-1} u(t)}{\mathrm{d}t^{m-1}}+\cdots+b_m\frac{\mathrm{d}u(t)}{\mathrm{d}t}+b_{m+1}u(t)$$

a_i, b_i　　$m \leqslant n$

- n为阶次，　　为常数，　　物理可实现
- 线性定常系统 LTI（linear time invariant）
  - 线性时不变模型、 LTI模型

传递函数的理论基础——Laplace变换

Pierre-Simon Laplace (1749- 1827)

法国数学家 Laplace变换 t 域到 s域

- 定义

$$\mathscr{L}[f(t)] = \int_0^\infty f(t)\mathrm{e}^{-st}\mathrm{d}t = F(s)$$

- Laplace变换的一条重要性质

若 $y(0) = \dot{y}(0) = \ddot{y}(0) = \cdots = y^{(n-1)}(0) = 0$

则

$$\mathscr{L}\left[\frac{\mathrm{d}^n y(t)}{\mathrm{d}t^n}\right] = s^n\mathscr{L}[y(t)] = s^n Y(s)$$

微分方程到传递函数转换举例

- RLC电路的微分方程

$$LC\frac{\mathrm{d}^2u_c(t)}{\mathrm{d}t^2}+RC\frac{\mathrm{d}u_c(t)}{\mathrm{d}t}+u_c(t)=u(t)$$

- RLC电路的微分方程 $u_c(0)=0,\quad \left.\frac{\mathrm{d}u(t)}{\mathrm{d}t}\right|_{t=0}=0$

- Laplace变换 $\mathscr{L}\left[\frac{\mathrm{d}^n y(t)}{\mathrm{d}t^n}\right]=s^n\mathscr{L}[y(t)]=s^nY(s)$

$$(LCs^2+RCs+1)U_c(s)=U(s)\ \rightarrow\ G(s)=\frac{U_c(s)}{U(s)}=\frac{1}{LCs^2+RCs+1}$$

传递函数的MATLAB表示

- 传递函数即放大倍数 $G(s)=Y(s)/U(s)$
- 传递函数的一般表示

$$G(s)=\frac{b_1 s^m+b_2 s^{m-1}+\cdots+b_m s+b_{m+1}}{a_1 s^n+a_2 s^{n-1}+a_3 s^{n-2}+\cdots+a_n s+a_{n+1}}$$

- 传递函数的输入方法

num=[$b_1,b_2,\cdots,b_m,b_{m+1}$];

den=[$a_1,a_2,\cdots,a_n,a_{n+1}$]; G=tf(num,den);

传递函数的MATLAB表示

- 传递函数即放大倍数 $G(s)=Y(s)/U(s)$
- 传递函数的一般表示

$$G(s)=\frac{b_1 s^m+b_2 s^{m-1}+\cdots+b_m s+b_{m+1}}{a_1 s^n+a_2 s^{n-1}+a_3 s^{n-2}+\cdots+a_n s+a_{n+1}}$$

- 传递函数的输入方法

num=[b_1,b_2,$\cdots$,b_m,b_{m+1}];

den=[a_1,a_2,$\cdots$,a_n,a_{n+1}]; G=tf(num,den);

例4-2 传递函数输入举例

- 传递函数模型
  - 数学模型

$$G(s)=\frac{12s^3+24s^2+12s+20}{2s^4+4s^3+6s^2+2s+2}$$

  - MATLAB输入语句

```
>> num=[12 24 12 20];
   den=[2 4 6 2 2]; G=tf(num,den)
```

- 在MATLAB环境中建立一个变量 G

例4-3 另外一种传递函数输入方法

- 如何处理如下的传递函数？

$$G(s)=\frac{3(s^2+3)}{(s+2)^3(s^2+2s+1)(s^2+5)}$$

- 定义算子 s=tf('s')，再输入传递函数

```
>> s=tf('s');
   G=3*(s^2+3)/(s+2)^3/(s^2+2*s+1)/(s^2+5)
```

- 应该根据给出传递函数形式选择输入方法

例4-4 复杂传递函数的输入

- 输入混合运算的传递函数模型

$$G(s)=\frac{s^3+2s^2+3s+4}{s^3(s+2)[(s+5)^2+5]}$$

显然用第一种方法麻烦，所以

```
>> s=tf('s');
   G=(s^3+2*s^2+3*s+4)/(s^3*(s+2)*((s+5)^2+5))
```

- 不同方法有不同的适用范围

MATLAB的传递函数对象

- 传递函数对象属性：新版本 get(tf)

```
    Numerator: {}
  Denominator: {}
     Variable: 's'
      IODelay: []
   InputDelay: [01 double]
  OutputDelay: [01 double]
           Ts: 0
     TimeUnit: 'seconds'
    InputName: {01 cell}
    InputUnit: {01 cell}
   InputGroup: [11 struct]
   OutputName: {01 cell}
   OutputUnit: {01 cell}
  OutputGroup: [11 struct]
        Notes: [01 string]
     UserData: []
         Name: ''
 SamplingGrid: [11 struct]
```

例4-5 传递函数属性修改

- 延迟传递函数 $G(s)\mathrm{e}^{-3s}$，即 $\tau=3$

```
>> G.ioDelay=3
   set(G,'ioDelay',3)
```

- 若假设复域变量为 p，则

```
set(G,'Variable','p'); G.Variable='p';
```

- 直接赋值方法

```
>> s=tf('s');
   G=(s^3+2*s^2+3*s+4)/(s^3*(s+2)*((s+5)^2+5))*exp(-3*s)
```

多变量系统传递函数矩阵模型

- 传递函数矩阵

$$\boldsymbol{G}(s)=\begin{bmatrix} g_{11}(s) & g_{12}(s) & \cdots & g_{1p}(s) \\ g_{21}(s) & g_{22}(s) & \cdots & g_{2p}(s) \\ \vdots & \vdots & \ddots & \vdots \\ g_{q1}(s) & g_{q2}(s) & \cdots & g_{qp}(s) \end{bmatrix}$$

$g_{ij}(s)$

- 为第 i 输出对第 j 输入的传递函数
- 可以先定义子传递函数，再由矩阵定义 $\boldsymbol{G}(s)$

例4-6 多变量模型

- 多变量传递函数矩阵模型

$$G(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\ \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$$

- 输入方法

```
>> G=[tf(0.1134,[1.78 4.48 1]), tf(0.924,[2.07 1]);
      tf(0.3378,[0.361 1.09 1]), tf(-0.318,[2.93 1])];
G.ioDelay=[0.72 0; 0.3, 1.29]
```

另一种输入方法

- 先输入各个子传递函数

$$G(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\ \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$$

- 再按普通矩阵输入的方法输入传递函数矩阵

```
>> g11=tf(0.1134,[1.78 4.48 1],'ioDelay',0.72);
   g12=tf(0.924,[2.07 1]);
   g21=tf(0.3378,[0.361 1.09 1],'ioDelay',0.3);
   g22=tf(-0.318,[2.93 1],'ioDelay',1.29);
   G=[g11, g12; g21, g22]
```

传递函数参数提取

- 由于使用单元数组，直接用 G.num 不行
- 有两种方法可以提取参数

```
>> [num,den]=tfdata(G,'v')
```

```
num=G.num{1}; den=G.den{1};
```

- 这样定义的优点：可以直接描述多变量系统
  - 第 i 输出对第 j 输入的传递函数

```
num=G.num{i,j}; den=G.den{i,j};
```

例4-8 状态方程举例

- 例4-1的RLC电路模型

$$Ri(t)+L\frac{\mathrm{d}i(t)}{\mathrm{d}t}+u_c(t)=u(t)\ \rightarrow\ \frac{\mathrm{d}i(t)}{\mathrm{d}t}=-\frac{R}{L}i(t)-\frac{1}{L}u_c(t)+\frac{1}{L}u(t)$$

$$i(t)=C\frac{\mathrm{d}u_c(t)}{\mathrm{d}t}\ \rightarrow\ \frac{\mathrm{d}u_c(t)}{\mathrm{d}t}=\frac{1}{C}i(i)$$

- 选择 $x_1(t)=i(t),\ x_2(t)=u_c(t)$

$$x_1'(t)=-Rx_1(t)/L-x_2(t)/L+u(t)/L$$

$$x_2'(t)=x_1(t)$$

$$\begin{bmatrix}x_1'(t)\\x_2'(t)\end{bmatrix}=\begin{bmatrix}-R/L & -1/L\\1 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\x_2(t)\end{bmatrix}+\begin{bmatrix}1/L\\0\end{bmatrix}u(t)$$

线性系统的状态方程模型

- 非线性状态方程模型

$$\begin{cases} \dot{x}_i = f_i(x_1, x_2, \cdots, x_n, u_1, \cdots, u_p), & i = 1, \cdots, n \\ y_i = g_i(x_1, x_2, \cdots, x_n, u_1, \cdots, u_p), & i = 1, \cdots, q \end{cases}$$

- 状态方程的重要元素
  - 状态变量 x_i，阶次 n，输入和输出
  - 非线性函数：$f_i(\cdot), g_i(\cdot)$
  - 一般非线性系统的状态方程描述
- 状态方程模型是微分方程的一种表现形式

线性状态方程

- 一般线性状态方程模型
  - 数学形式

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) + \boldsymbol{D}(t)\boldsymbol{u}(t) \end{cases}$$

  - 时变状态方程
- 线性时不变模型(linear time invariant, LTI)

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{cases}$$

常规状态方程的局限性

- 一些特殊线性模型不存在状态方程
  - 如果物理不可实现
  - 严格正则系统的逆系统
- 需要引入下面的方程，称为描述符系统

$$\begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{cases}$$

  - 对物理可实现系统而言， $\boldsymbol{E}$ 为单位矩阵
  - 这种模型是MATLAB下的标准状态方程模型

线性时不变模型的MATLAB描述

- MATLAB 输入方法

$$\begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{cases}$$

$G=\mathrm{ss}(A,B,C,D);$

$G=\mathrm{ss}(A,B,C,D,E);$

  - 系数矩阵 A 矩阵是方程，为 $n \times n$ 矩阵
  - B 为 $n \times p$ 矩阵，C 为 $q \times n$ 矩阵，D 为 $q \times p$ 矩阵
  - 可以直接处理多变量模型
  - 给出(A,B,C,D)矩阵即可
- 注意维数的兼容性

例4-9 状态方程的输入

- 状态方程模型

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -12 & -17.2 & -16.8 & -11.9 \\ 6 & 8.6 & 8.4 & 6 \\ 6 & 8.7 & 8.4 & 6 \\ -5.9 & -8.6 & -8.3 & -6 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1.5 & 0.2 \\ 1 & 0.3 \\ 2 & 1 \\ 0 & 0.5 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 2 & 0.5 & 0 & 0.8 \\ 0.3 & 0.3 & 0.2 & 1 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

- MATLAB 模型输入

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
      6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5];
   C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D)
```

```
>> G=ss(A,B,C,0)
```

带时间延迟的状态方程

- 数学模型

$$\begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t-\boldsymbol{\tau}_{\mathrm{i}}) \\ \boldsymbol{z}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t-\boldsymbol{\tau}_{\mathrm{i}}), \quad \boldsymbol{y}(t) = \boldsymbol{z}(t-\boldsymbol{\tau}_{\mathrm{o}}) \end{cases}$$

- MATLAB输入语句

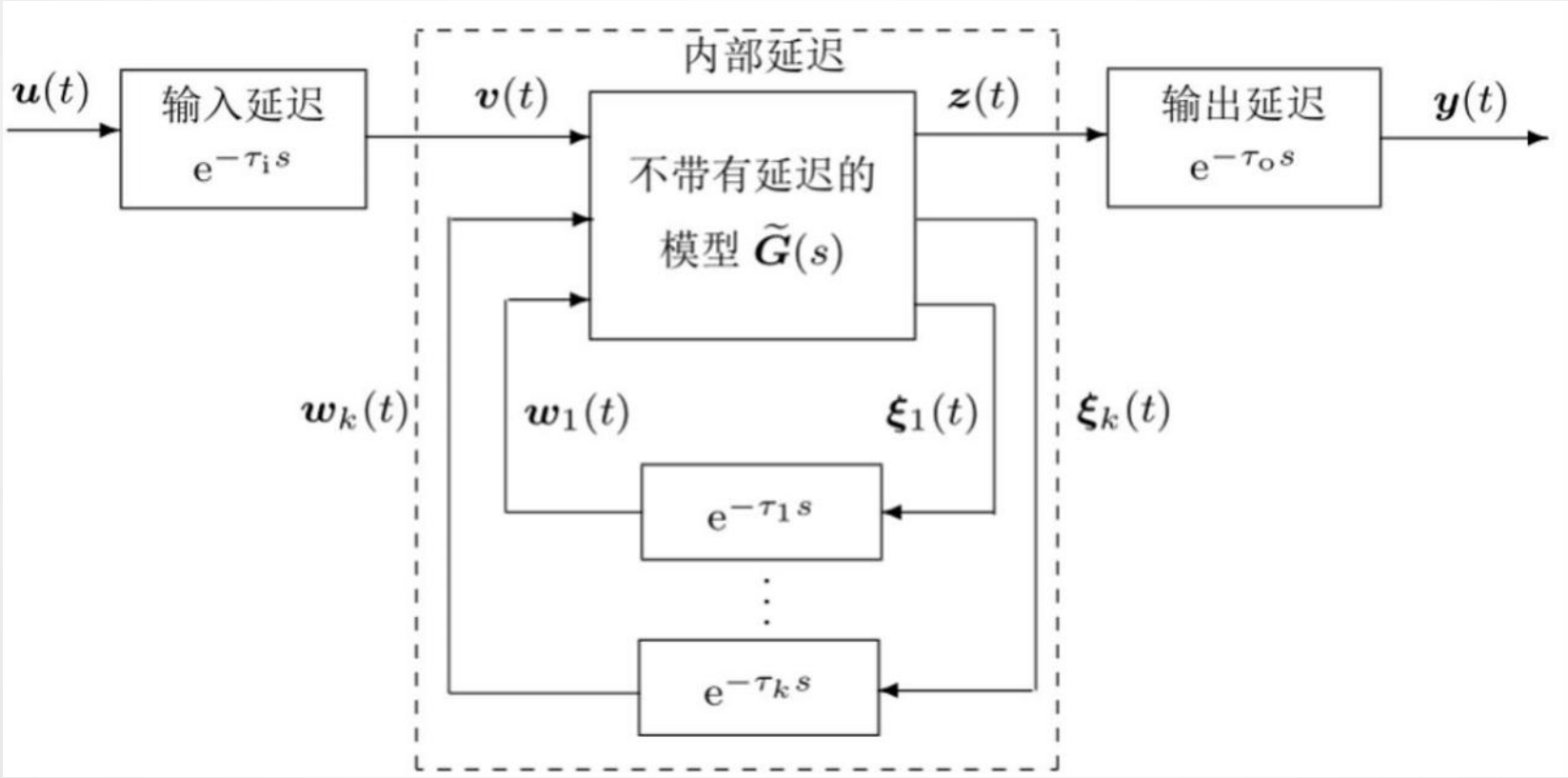
G=ss($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,$\boldsymbol{D}$,'InputDelay',$\boldsymbol{\tau}_{\mathrm{i}}$,'OutputDelay',$\boldsymbol{\tau}_{\mathrm{o}}$)

G=ss($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,$\boldsymbol{D}$,$\boldsymbol{E}$,'InputDelay',$\boldsymbol{\tau}_{\mathrm{i}}$,'OutputDelay',$\boldsymbol{\tau}_{\mathrm{o}}$)

- 其他延迟属性：ioDelay

带有内部延迟的状态方程

- 框图描述，更一般的状态方程模型



数学模型与输入方法

- 数学描述

$$\begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_1\boldsymbol{v}(t) + \boldsymbol{w}(t-\boldsymbol{\tau}) \\ \boldsymbol{z}(t) = \boldsymbol{C}_1\boldsymbol{x}(t) + \boldsymbol{D}_{11}\boldsymbol{v}(t) + \boldsymbol{D}_{12}\boldsymbol{w}(t-\boldsymbol{\tau}) \\ \boldsymbol{\xi}(t) = \boldsymbol{C}_2\boldsymbol{x}(t) + \boldsymbol{D}_{21}\boldsymbol{v}(t) + \boldsymbol{D}_{22}\boldsymbol{w}(t-\boldsymbol{\tau}) \end{cases}$$

- 其中 $\boldsymbol{w}_j(t) = \boldsymbol{\xi}_j(t-\tau_j),\ j = 1, 2, \cdots, k$ 内部延迟

  $\boldsymbol{v}(t) = \boldsymbol{u}(t-\tau_{\mathrm{i}}),\ \boldsymbol{y}(t) = \boldsymbol{z}(t-\tau_{\mathrm{o}})$ $\boldsymbol{\tau} = [\tau_1, \tau_2, \cdots, \tau_k]$

- MATLAB 输入方法

$[\boldsymbol{A}, \boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{D}_{11}, \boldsymbol{D}_{12}, \boldsymbol{D}_{21}, \boldsymbol{D}_{22}, \boldsymbol{E}, \boldsymbol{\tau}] =$ `getdelaymodel(`G`,'mat')`

- $\boldsymbol{v}(t)$, $\boldsymbol{u}(t)$为内部信号，提取模型

线性系统的零极点模型

- 零极点模型是因式型传递函数模型

$$G(s)=K\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}$$

  - 零点 z_i、极点 p_i 和增益 K

- 零极点模型的MATLAB表示

```
z=[z1; z2; ···; zm];    s=zpk('s');
p=[p1; p2; ···; pn];    G=...
G=zpk(z, p, K);
```

例4-10 零极点模型的输入

- 已知零极点模型

$$G(s)=\frac{6(s+5)(s+2+\mathrm{j}2)(s+2-\mathrm{j}2)}{(s+4)(s+3)(s+2)(s+1)}$$

- MATLAB 输入方法

```
>> P=[-1; -2; -3; -4];
   Z=[-5; -2+2i; -2-2i]; G=zpk(Z,P,6)
```

- [redacted]种输入方法；

```
G=6*(s+5)*(s+2+2i)*(s+2-2i)/((s+1)*(s+2)*(s+3)*(s+4))
```

线性系统的数学模型

- 4.1 线性连续系统模型及MATLAB表示
- 4.2 线性离散时间系统的数学模型
- 4.3 线性模型的相互转换
- 4.4 方框图描述系统的化简
- 4.5 线性系统的模型降阶
- 4.6 线性系统的模型辨识

离散系统的差分方程模型

- 差分方程表示

$$a_1 y(t+n) + a_2 y(t+n-1) + \cdots + a_n y(t+1) + a_{n+1} y(t)$$

$$= b_0 u(t+n) + b_1 u(k+n-1) + \cdots + b_{n-1} u(t+1) + b_n u(t)$$

- z变换

$$\mathscr{Z}[y(t+k)] = z^k \mathscr{Z}[y(t)]$$

$$H(z) = \frac{b_0 z^n + b_1 z^{n-1} + \cdots + b_{n-1} z + b_n}{a_1 z^n + a_2 z^{n-1} + \cdots + a_n z + a_{n+1}}$$

离散传递函数模型

- 数学表示（z 变换代替Laplace变换）

$$H(z)=\frac{b_0z^n+b_1z^{n-1}+\cdots+b_{n-1}z+b_n}{a_1z^n+a_2z^{n-1}+\cdots+a_nz+a_{n+1}}$$

- MATLAB表示（采样周期T）

num=[b_0,b_1,$\cdots$,b_{n-1},b_n];

den=[a_1,a_2,$\cdots$,a_n,a_{n+1}]; H=tf(num,den,'Ts',T);

- 算子输入方法： z=tf('z',T)

例4-8 离散传递函数输入

离散传递函数，采样周期 T=0.1

$$H(z)=\frac{6z^2-0.6z-0.12}{z^4-z^3+0.25z^2+0.25z-0.125}$$

- MATLAB输入方法

```
>> z=tf('z',0.1);
   H=(6*z^2-0.6*z-0.12)/(z^4-z^3+0.25*z^2+0.25*z-0.125)
```

- 另一种输入方法

```
>> num=[6 -0.6 -0.12]; den=[1 -1 0.25 0.25 -0.125];
   H=tf(num,den,'Ts',0.1)
```

离散延迟系统与输入

- 数学模型

$$H(z)=\frac{b_0z^n+b_1z^{n-1}+\cdots+b_{n-1}z+b_n}{a_1z^n+a_2z^{n-1}+\cdots+a_nz+a_{n+1}}z^{-d}$$

  - 延迟为采样周期的整数倍 dT
  - MATLAB输入方法

```
H.ioDelay=d
set(H,'ioDelay',d)
```

滤波器型描述方法

- 滤波器型离散模型

$$\widehat{H}\left(z^{-1}\right)=\frac{b_0+b_1z^{-1}+\cdots+b_{n-1}z^{-n+1}+b_nz^{-n}}{a_1+a_2z^{-1}+\cdots+a_nz^{-n+1}+a_{n+1}z^{-n}}$$

- 分子、分母除以 z^n
- 记 $q=z^{-1}$，则

$$\widehat{H}(q)=\frac{b_0+b_1q+\cdots+b_{n-1}q^{n-1}+b_nq^n}{a_1+a_2q+\cdots+a_nq^{n-1}+a_{n+1}q^n}$$

例4-9 零极点模型输入

- 零极点型模型可以用zpk函数输入
- 零极点型传递函数，T=0.1

$$H(z)=\frac{(z-1/2)(z-1/2+\mathrm{j}/2)(z-1/2-\mathrm{j}/2)}{120(z+1/2)(z+1/3)(z+1/4)(z+1/5)}$$

- MATLAB输入方法

```
>> z=[1/2; 1/2+1i/2; 1/2-1i/2]; p=[-1/2; -1/3; -1/4; -1/5];
   H=zpk(z,p,1/120,'Ts',0.1)
```

- 本例有复数极点，不适合用方法 z=zpk('z')

离散状态方程模型

- 离散状态方程数学形式

$$\begin{cases} \boldsymbol{E}\boldsymbol{x}[(k+1)T] = \boldsymbol{F}\boldsymbol{x}(kT) + \boldsymbol{G}\boldsymbol{u}(kT) \\ \boldsymbol{y}(kT) = \boldsymbol{C}\boldsymbol{x}(kT) + \boldsymbol{D}\boldsymbol{u}(kT) \end{cases}$$

  - 注意矩阵兼容性
  - T 为采样周期
  - MATLAB表示方法

$$H=\mathtt{ss}(\boldsymbol{F},\boldsymbol{G},\boldsymbol{C},\boldsymbol{D},\mathtt{'Ts'},T);$$

$$H=\mathtt{ss}(\boldsymbol{F},\boldsymbol{G},\boldsymbol{C},\boldsymbol{D},\boldsymbol{E},\mathtt{'Ts'},T);$$

离散延迟系统的状态方程

- 数学模型

$$\begin{cases} \boldsymbol{x}[(k+1)T] = \boldsymbol{F}\boldsymbol{x}(kT) + \boldsymbol{G}\boldsymbol{u}[(k-d)T] \\ \boldsymbol{y}(kT) = \boldsymbol{C}\boldsymbol{x}(kT) + \boldsymbol{D}\boldsymbol{u}[(k-d)T] \end{cases}$$

- MATLAB表示方法

$$H=\texttt{ss}(\boldsymbol{F},\boldsymbol{G},\boldsymbol{C},\boldsymbol{D},\texttt{'Ts'},T,\texttt{'ioDelay'},d);$$

- 离散系统也有内部延迟模型

系统数学模型举例小结

- 系统模型的分类方法
  - 自动控制原理与现代控制理论课程涉及的数学模型只是系统模型中很窄的几类模型
- 给出了连续系统建模的实例
  - 列出系统的微分方程模型
  - 由微分方程导入系统传递函数的概念——传递函数是系统的增益，是 s 的函数

传递函数输入小结

- 传递函数的定义与两种输入方法
  - 调用 tf 函数直接输入
  - 定义 *s* 算子，再输入传递函数
- 可以提取传递函数的分子与分母 tfdata
- 多变量传递函数矩阵的输入
- 有了传递函数模型, 则可以将其输入到MATLAB以后则可以对其分析与设计了

状态方程与零极点输入小结

- 线性系统的模型表示方法
  - 传递函数模型 tf
  - 状态方程模型的输入 ss
  - 零极点模型 zpk
- 常规模型可能遇到困难，需要描述符系统
- 带有内部延迟的状态方程模型

离散系统模型

- 离散系统也有各种线性模型
  - 传递函数 tf
  - 状态方程 ss
  - 零极点模型 zpk
- 离散系统的采样周期 T
- 离散系统的延迟模型

Q & A

感谢您的聆听和反馈

国家精品课程/ 国家精品资源共享课程/ 国家级精品教材
国家级十一(二)五规划教材/ 教育部自动化专业教学指导委员会牵头规划系列教材

控制系统计算机辅助设计

第4章 线性控制系统的数学模型

主讲：修贤超

线性系统的数学模型

- 4.1 线性连续系统模型及MATLAB表示
- 4.2 线性离散时间系统的数学模型
- 4.3 线性模型的相互转换
- 4.4 方框图描述系统的化简
- 4.5 线性系统的模型降阶
- 4.6 线性系统的模型辨识

系统模型的相互转换

- 前面介绍的各种模型之间的相互等效变换
- 主要内容
  - 连续模型和离散模型的相互转换
  - 系统传递函数的获取
  - 控制系统的状态方程实现
  - 状态方程的最小实现
  - 传递函数与符号表达式的相互转换

连续模型和离散模型的相互转换

- 连续状态方程的解析解

$$\boldsymbol{x}(t) = \mathrm{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}(t_0) + \int_{t_0}^{t} \mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau$$

- 采样周期 T
- 选择 $t_0 = kT, t = (k+1)T$

$$\boldsymbol{x}[(k+1)T] = \mathrm{e}^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \int_{kT}^{(k+1)T} \mathrm{e}^{\boldsymbol{A}[(k+1)T-\tau]}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau$$

数学推导与MATLAB命令

- 离散化

$$\boldsymbol{x}[(k+1)T] = \mathrm{e}^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \int_0^{\mathrm{T}} \mathrm{e}^{\boldsymbol{A}\tau}\mathrm{d}\tau\ \boldsymbol{B}\boldsymbol{u}(kT)$$

$$\boldsymbol{x}[(k+1)T] = \boldsymbol{F}\boldsymbol{x}(kT) + \boldsymbol{G}\boldsymbol{u}(kT)$$

- 离散化等效关系 $\boldsymbol{F} = \mathrm{e}^{\boldsymbol{A}T}$, $\boldsymbol{G} = \int_0^{\mathrm{T}} \mathrm{e}^{\boldsymbol{A}\tau}\mathrm{d}\tau\ \boldsymbol{B}$
- 还可以采用Tustin变换（双线性变换） $s = \dfrac{2(z-1)}{T(z+1)}$
- MATLAB函数直接求解 G_1=c2d(G,T)

例4-10 状态方程的离散化

- 双输入模型，T=0.1

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -12 & -17.2 & -16.8 & -11.9 \\ 6 & 8.6 & 8.4 & 6 \\ 6 & 8.7 & 8.4 & 6 \\ -5.9 & -8.6 & -8.3 & -6 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1.5 & 0.2 \\ 1 & 0.3 \\ 2 & 1 \\ 0 & 0.5 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 2 & 0.5 & 0 & 0.8 \\ 0.3 & 0.3 & 0.2 & 1 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

- 输入模型、变换

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
      6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5];
   C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   G=ss(A,B,C,0); T=0.1; Gd=c2d(G,T)
```

例4-11 延迟传递函数的离散化

- 时间延迟系统的离散化 $G(s)=\dfrac{1}{(s+2)^3}\mathrm{e}^{-2s}$
  - MATLAB求解

```
>> s=tf('s'); G=1/(s+2)^3; G.ioDelay=2;
```

  - 零阶保持器变换

```
>> G1=c2d(G,0.1)
```

  - 变换结果 T=0.1

$$G_{\mathrm{ZOH}}(z)=\frac{0.0001436z^2+0.0004946z+0.0001064}{z^3-2.456z^2+2.011z-0.5488}z^{-20}$$

其他离散化方法

- Tustin变换 `>> G2=c2d(G,0.1,'tustin')`
- 数学表示

$$G_{\text{Tustin}}(z)=\frac{9.391\times10^{-5}z^3+0.0002817z^2+0.0002817z+9.391\times10^{-5}}{z^3-2.455z^2+2.008z-0.5477}z^{-20}$$

- 其他转换方法
  - FOH 一阶保持器
  - matched 单变量系统零极点不变
  - imp 脉冲响应不变准则

离散模型连续化

- 对前面的变换求逆 —— 数学公式
  - 状态方程的变换

$$\boldsymbol{A}=\frac{1}{T}\ln\boldsymbol{F},\quad \boldsymbol{B}=(\boldsymbol{F}-\boldsymbol{I})^{-1}\boldsymbol{A}\boldsymbol{G}$$

  - Tustin反变换

$$z=(1+sT/2)/(1-sT/2)$$

- MATLAB求解 (无需T)　　G_1=d2c(G)

例4-12 系统的连续化

- 对前面的连续状态方程模型离散化，
  - 对结果再连续化

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;...
       6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5];
   C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D); Gd=c2d(G,T);
   G1=d2c(Gd)
```

- 可以基本上还原连续模型

系统传递函数的获取

- 已知状态方程

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{cases}$$

- 两端Laplace变换

$$\begin{cases} s\boldsymbol{I}\boldsymbol{X}(s) = \boldsymbol{A}\boldsymbol{X}(s) + \boldsymbol{B}\boldsymbol{U}(s) \\ \boldsymbol{Y}(s) = \boldsymbol{C}\boldsymbol{X}(s) + \boldsymbol{D}\boldsymbol{U}(s) \end{cases}$$

- 则 $\boldsymbol{X}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{U}(s)$

系统传递函数的变换

- 因此可以得出传递函数

$$\boldsymbol{G}(s)=\boldsymbol{Y}(s)\boldsymbol{U}^{-1}(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}$$

  - 难点 $(s\boldsymbol{I}-\boldsymbol{A})^{-1}$
  - 基于Leverrie–Fadeev算法能得出更好结果
- 由零极点模型，直接展开分子分母
- 用MATLAB统一求解 $G_1=\mathtt{tf}(G)$

例4-13 多变量系统的变换

- 多变量模型，求传递函数矩阵

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
      6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5];
   C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D); G1=tf(G)
```

$$G(s)=\begin{bmatrix} \dfrac{3.5s^3-144.1s^2-20.69s-0.8372}{s^4+s^3+0.35s^2+0.05s+0.0024} & \dfrac{0.95s^3-64.13s^2-9.161s-0.374}{s^4+s^3+0.35s^2+0.05s+0.0024} \\ \dfrac{1.15s^3-36.32s^2-6.225s-0.1339}{s^4+s^3+0.35s^2+0.05s+0.0024} & \dfrac{0.85s^3-15.71s^2-2.619s-0.04559}{s^4+s^3+0.35s^2+0.05s+0.0024} \end{bmatrix}$$

控制系统的状态方程实现

- 由传递函数到状态方程的转换
  - 不同状态变量选择，结果不唯一
  - 默认变换方式，采用MATLAB函数 $G_1=\mathrm{ss}(G)$
- 通用的G模型
  - G 可以是传递函数、状态方程和零极点模型
  - 适用于有延迟的、离散的或多变量模型
  - 可以将延迟传递函数模型转成内部延迟
  - G可以是非正则系统，得出描述符状态方程模型

例4-14 多变量系统

- 连续多变量模型

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\ \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$$

- 状态方程获取

```
>> g11=tf(0.1134,[1.78 4.48 1],'ioDelay',0.72);
   g12=tf(0.924,[2.07 1]);
   g21=tf(0.3378,[0.361 1.09 1],'ioDelay',0.3);
   g22=tf(-0.318,[2.93 1],'ioDelay',1.29);
   G=[g11, g12; g21, g22]; G1=ss(G)
```

例4-15 非正则系统的实现

- 系统模型

$$G(s)=\frac{12s^3+24s^2+12s+20}{2s^4+4s^3+6s^2+2s+2}$$

- 逆系统模型的状态方程实现

$$G(s)=\frac{2s^4+4s^3+6s^2+2s+2}{12s^3+24s^2+12s+20}$$

```
>> num=[12 24 12 20]; den=[2 4 6 2 2];
   G=1/tf(num,den), G1=ss(G)
```

均衡实现 （**balanced realization**）

- 由一般状态方程输入输出关系显著程度不明显，需要进一步变换
- 均衡实现是一种很有用的方式
- 用MATLAB直接求解 $[G_{\mathrm{b}},\boldsymbol{g},\boldsymbol{T}]=\mathtt{balreal}(G)$
- 得出均衡实现的模型
- 得出排序的 Gram 矩阵

例4-17 均衡实现举例

- 原系统模型 $\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-1 & 0\\ 0 & -2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}10^{-6}\\ 10^{6}\end{bmatrix}u$

$$y=\begin{bmatrix}10^{6} & 10^{-6}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

- 引入 $z_1=10^{6}x_1,\quad z_2=10^{-6}$
- 内部坐标变换

$$\begin{bmatrix}\dot{z}_1\\ \dot{z}_1\end{bmatrix}=\begin{bmatrix}-1 & 0\\ 0 & -2\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix}+\begin{bmatrix}1\\ 1\end{bmatrix}u,\ y=\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix}$$

例4-18 状态方程的最小实现

观察传递函数模型

$$G(s)=\frac{5s^3+50s^2+155s+150}{s^4+11s^3+41s^2+61s+30}$$

- 未见有何特殊
- 求取零极点模型

```
>> G=tf([5 50 155 150],[1 11 41 61 30]);
   zpk(G)
```

最小实现的计算

- 得出结果

$$G(s)=\frac{5(s+3)(s+2)(s+5)}{(s+5)(s+3)(s+2)(s+1)}$$

- 最小实现方法
  - 相同位置的零极点，可以对消
  - 问题：多变量系统、状态方程如何处理？
  - MATLAB解决方法

```
>> G1=minreal(G)
```

G_{m}=minreal(G)

G_{m}=minreal(G,ϵ)

例4-19 多变量模型的最小实现

- 多变量模型

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -6 & -1.5 & 2 & 4 & 9.5 \\ -6 & -2.5 & 2 & 5 & 12.5 \\ -5 & 0.25 & -0.5 & 3.5 & 9.75 \\ -1 & 0.5 & 0 & -1 & 1.5 \\ -2 & -1 & 1 & 2 & 3 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 6 & 4 \\ 5 & 5 \\ 3 & 4 \\ 0 & 2 \\ 3 & 1 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 2 & 0.75 & -0.5 & -1.5 & -2.75 \\ 0 & -1.25 & 1.5 & 1.5 & 2.25 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

- 不能直接看出是否最小实现

最小实现计算

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -6 & -1.5 & 2 & 4 & 9.5 \\ -6 & -2.5 & 2 & 5 & 12.5 \\ -5 & 0.25 & -0.5 & 3.5 & 9.75 \\ -1 & 0.5 & 0 & -1 & 1.5 \\ -2 & -1 & 1 & 2 & 3 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 6 & 4 \\ 5 & 5 \\ 3 & 4 \\ 0 & 2 \\ 3 & 1 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 2 & 0.75 & -0.5 & -1.5 & -2.75 \\ 0 & -1.25 & 1.5 & 1.5 & 2.25 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

- MATLAB求解

```
>> A=[-6,-1.5,2,4,9.5; -6,-2.5,2,5,12.5; -5,0.25,-0.5,3.5,9.75;
      -1, 0.5, 0, -1, 1.5;  -2, -1, 1, 2, 3];
   B=[6,4; 5,5; 3,4; 0,2; 3,1]; D=zeros(2);
   C=[2,0.75,-0.5,-1.5,-2.75; 0,-1.25,1.5,1.5,2.25];
   G=ss(A,B,C,D); G1=minreal(G)
```

- 数学表示

$$\begin{cases} \dot{\hat{\boldsymbol{x}}}(t) = \begin{bmatrix} -2.4125 & 1.1729 & -0.17022 \\ -0.73946 & 0.12333 & -0.37256 \\ -0.65067 & 1.6766 & -1.7108 \end{bmatrix} \hat{\boldsymbol{x}}(t) + \begin{bmatrix} 6.4843 & 4.0942 \\ 5.1517 & 3.7888 \\ 3.227 & 5.5572 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 0.84235 & 0.073798 & 0.048876 \\ 0.25085 & 0.36129 & 0.46861 \end{bmatrix} \hat{\boldsymbol{x}}(t) \end{cases}$$

线性系统的数学模型

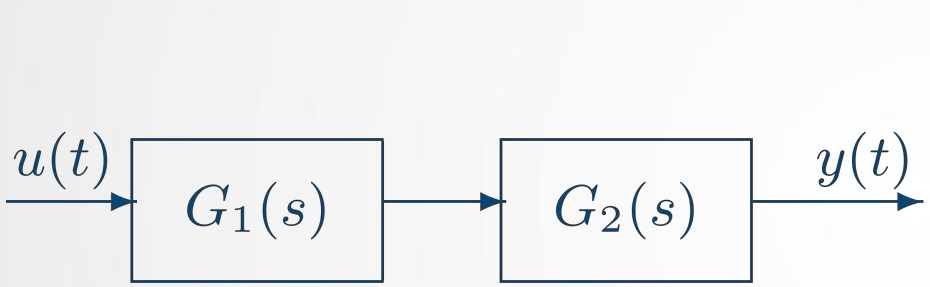
- 4.1 线性连续系统模型及MATLAB表示
- 4.2 线性离散时间系统的数学模型
- 4.3 线性模型的相互转换
- 4.4 方框图描述系统的化简
- 4.5 线性系统的模型降阶
- 4.6 线性系统的模型辨识

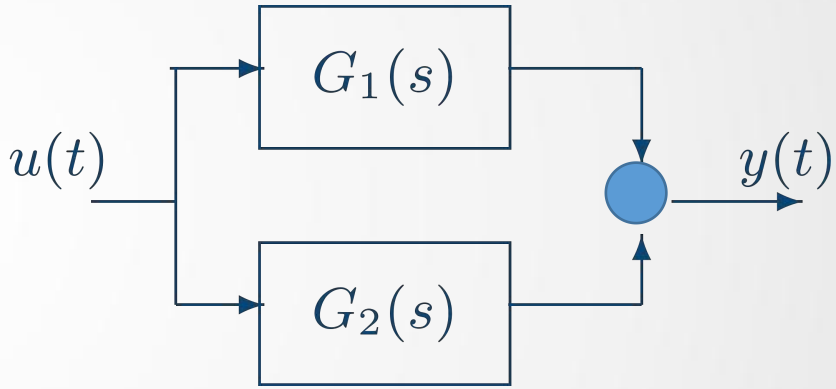
方框图描述系统的化简

- 单环节模型前面已经介绍了，实际系统为多个环节互连
- 如何解决互连问题，获得等效模型？
- 主要内容
  - 控制系统的典型连接结构
  - 节点移动时的等效变换
  - 复杂系统模型的简化
  - 基于信号流图的代数化简方法

控制系统的典型连接结构

- 系统串、并联





- 数学等效方法
  - 串联传递函数 $G(s) = G_2(s)G_1(s)$
  - 并联传递函数 $G(s) = G_1(s) + G_2(s)$

串、并联状态方程模型

- 串联系统的状态方程

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ B_2C_1 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2D_1 \end{bmatrix} u \\ y = [D_2C_1, C_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D_2D_1u \end{cases}$$

- 并联系统的状态方程

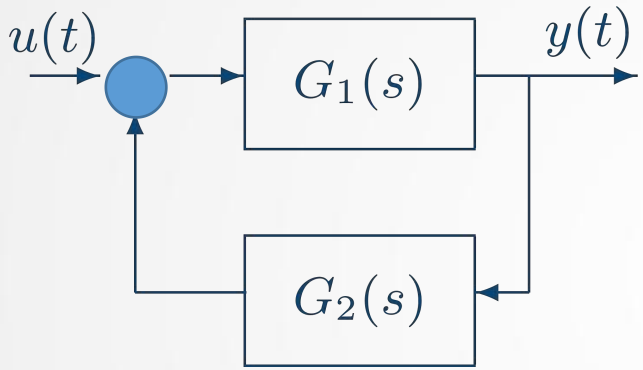
$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \\ y = [C_1, C_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + (D_1 + D_2)u \end{cases}$$

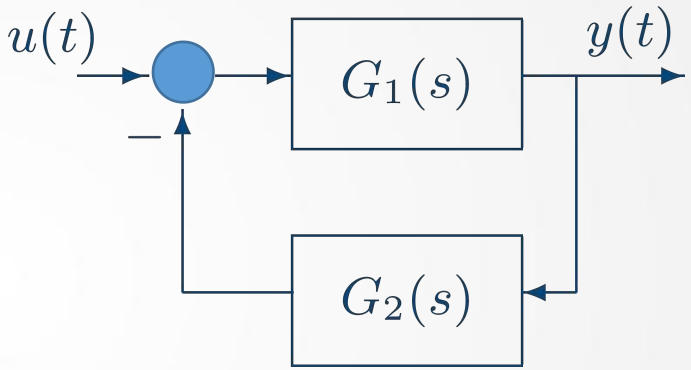
串、并联系统的MATLAB求解

- 若一个模型为传递函数、另一个为状态方程如何处理？
  - 将二者变换成同样结构再计算
- 基于MATLAB的计算方法
  - 串联 $G=G_2*G_1$ 注意次序：多变量系统
  - 并联 $G=G_2+G_1$
  - 优点，无须事先转换，可以直接处理多变量系统

系统的反馈连接

- 反馈结构





- 数学等效关系
  - 负反馈 $G(s)=\dfrac{G_1(s)}{1+G_1(s)G_2(s)}$ 正反馈 $G(s)=\dfrac{G_1(s)}{1-G_1(s)G_2(s)}$
  - 多变量系统 $\boldsymbol{G}(s)=[\boldsymbol{I}\pm\boldsymbol{G}_1(s)\boldsymbol{G}_2(s)]^{-1}\boldsymbol{G}_1(s)$

状态方程的反馈等效方法

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_1 - B_1 Z D_2 C_1 & -B_1 Z C_2 \\ B_2 Z C_1 & A_2 - B_2 D_1 Z C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 Z \\ B_2 D_1 Z \end{bmatrix} u \\ y = [Z C_1, \ -D_1 Z C_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + (D_1 Z) u \end{cases}$$

- 其中 $Z = (I + D_1 D_2)^{-1}$
- 若 $D_1 = D_2 = 0$

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_1 & -B_1 C_2 \\ B_2 C_1 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} u \\ y = [C_1, \ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{cases}$$

反馈连接的MATLAB求解

- LTI 模型

  G=feedback(G_1,G_2);

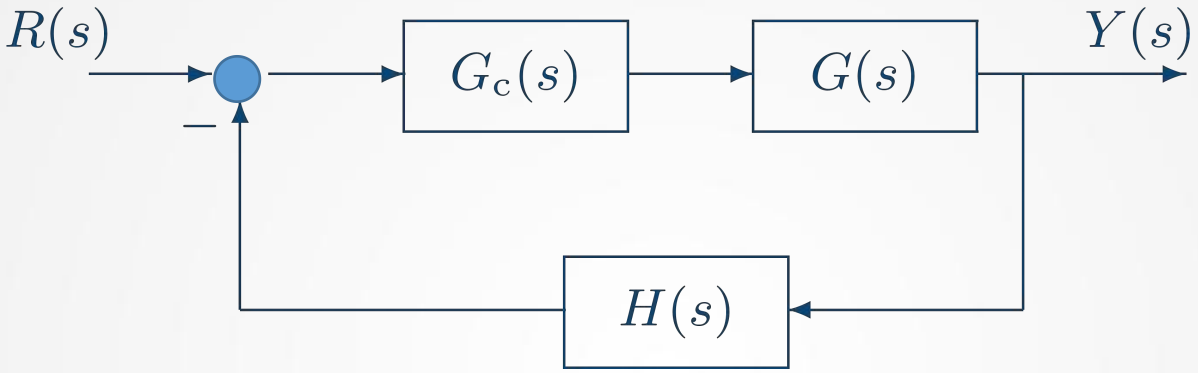
  G=feedback(G_1,G_2,1);

- 符号运算实现

```
function H=feedbacksym(G1,G2,key)
if nargin==2; key=-1; end,
H=G1/(sym(1)-key*G1*G2); H=simplify(H);
```

例4-17 典型反馈控制

典型反馈



$$G(s)=\frac{12s^3+24s^2+12s+20}{2s^4+4s^3+6s^2+2s+2} \qquad G_c(s)=\frac{5s+3}{s},\quad H(s)=\frac{1000}{s+1000}$$

```
>> s=tf('s');
   G=(12*s^3+24*s^2+12*s+20)/(2*s^4+4*s^3+6*s^2+2*s+2);
   Gc=(5*s+3)/s; H=1000/(s+1000); GG=feedback(G*Gc,H)
>> syms G Gc H; GG=feedbacksym(G*Gc,H)
```

例4-18 多变量系统处理

例4-18 状态空间受控对象模型

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -12 & -17.2 & -16.8 & -11.9 \\ 6 & 8.6 & 8.4 & 6 \\ 6 & 8.7 & 8.4 & 6 \\ -5.9 & -8.6 & -8.3 & -6 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1.5 & 0.2 \\ 1 & 0.3 \\ 2 & 1 \\ 0 & 0.5 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 2 & 0.5 & 0 & 0.8 \\ 0.3 & 0.3 & 0.2 & 1 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6; ...
      6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5];
   C=[2,0.5,0,0.8; 0.3,0.3,0.2,1]; G=ss(A,B,C,0)
```

多变量系统的连接

- 控制器为对角矩阵

$$G_c(s) = \begin{bmatrix} (2s+1)/s & 0 \\ 0 & (5s+2)/s \end{bmatrix}$$

- 反馈环节为单位矩阵
- 闭环系统模型计算

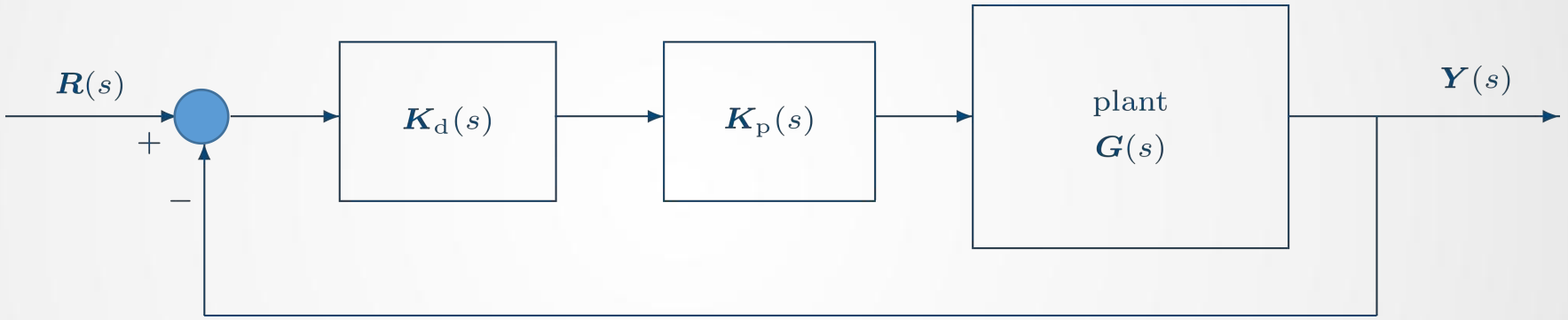
```
>> s=tf('s'); g11=(2*s+1)/s;g22=(5*s+2)/s;
   Gc=[g11 0;0 g22]; H=eye(2); GG=feedback(G*Gc,H)
```

带有延迟的闭环系统模型处理

- 传递函数模型是不能处理延迟的
  - 开环模型 $$G(s)=\frac{1}{s+2}\mathrm{e}^{-3s}$$
  - 闭环传递函数 $$G_c(s)=\frac{\frac{1}{s+2}\mathrm{e}^{-3s}}{1+\frac{1}{s+2}\mathrm{e}^{-3s}}=\frac{1}{s+2+\mathrm{e}^{-3s}}\mathrm{e}^{-3s}$$
- 需要引入带有内部延迟的状态方程模型

例4-19 内部延迟模型的应用

- 多变量闭环系统



$$G(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\ \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$$

模型化简方法

- 控制器模型

$$\boldsymbol{K}_{\mathrm{p}}(s)=\begin{bmatrix}-0.4136 & 2.6537\\ 1.133 & -0.3257\end{bmatrix}$$

$$\boldsymbol{K}_{\mathrm{d}}(s)=\begin{bmatrix}3.8582+1.0640/s & 0\\ 0 & 1.1487+0.8133/s\end{bmatrix}$$

- MATLAB求解

```
>> g11=tf(0.1134,[1.78 4.48 1],'ioDelay',0.72);
   g12=tf(0.924,[2.07 1]);
   g21=tf(0.3378,[0.361 1.09 1],'ioDelay',0.3);
   g22=tf(-0.318,[2.93 1],'ioDelay',1.29); G=[g11, g12; g21, g22];
>> s=tf('s'); Kp=[-0.4136,2.6537; 1.133,-0.3257];
   Kd=[3.8582+1.0640/s, 0; 0, 1.1487+0.8133/s];
   H=eye(2); G1=feedback(G*Kp*Kd,H)
```

例4-20 计算机控制系统

- 典型计算机控制系统

$$G(s)=\frac{2}{s(s+2)},\ G_c(z)=\frac{9.1544(z-0.9802)}{z-0.8187},\ T=0.2\,\mathrm{s}$$

- 应该先统一域，再化简
  - 统一成离散模型

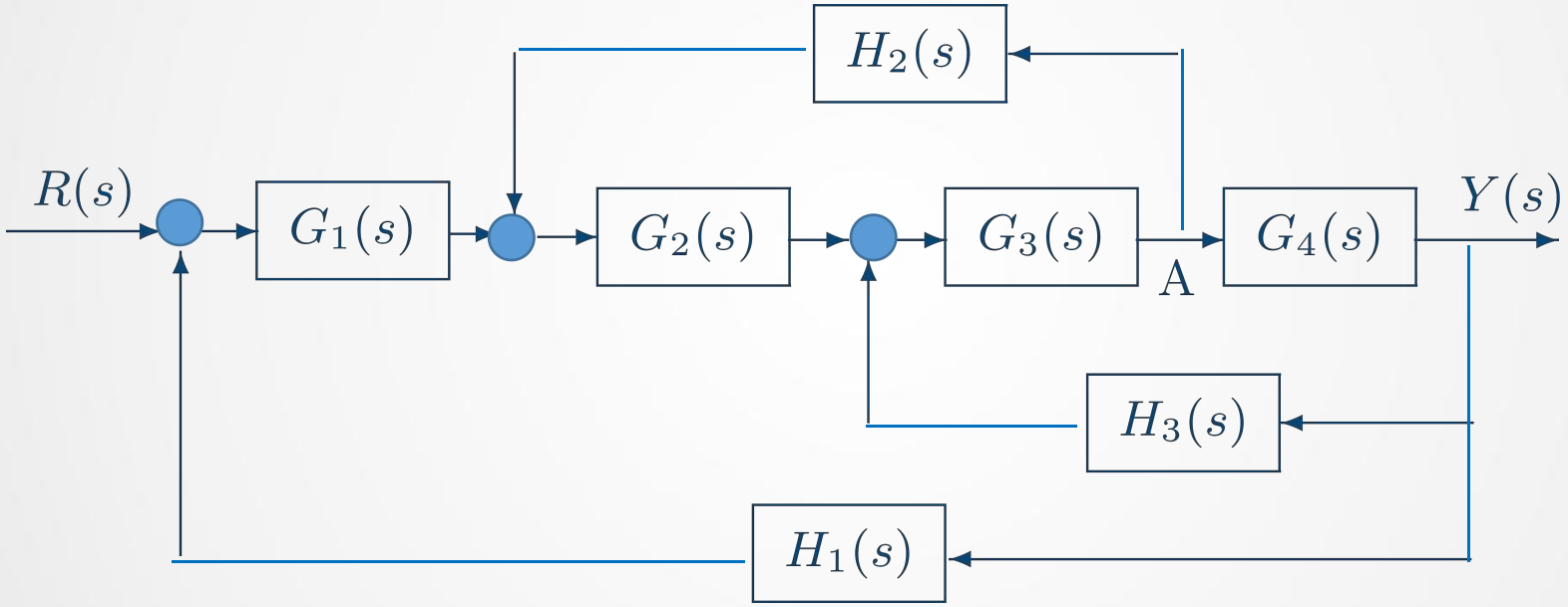
```
>> s=tf('s'); T=0.2; G=2/s/(s+2);
   z=tf('z',T); Gc=9.1544*(z-0.9802)/(z-0.8187);
   G1=feedback(c2d(G,T)*Gc,1)
```

  - 统一成连续模型

```
>> G2=feedback(G*d2c(Gc),1)
```

节点移动时的等效变换

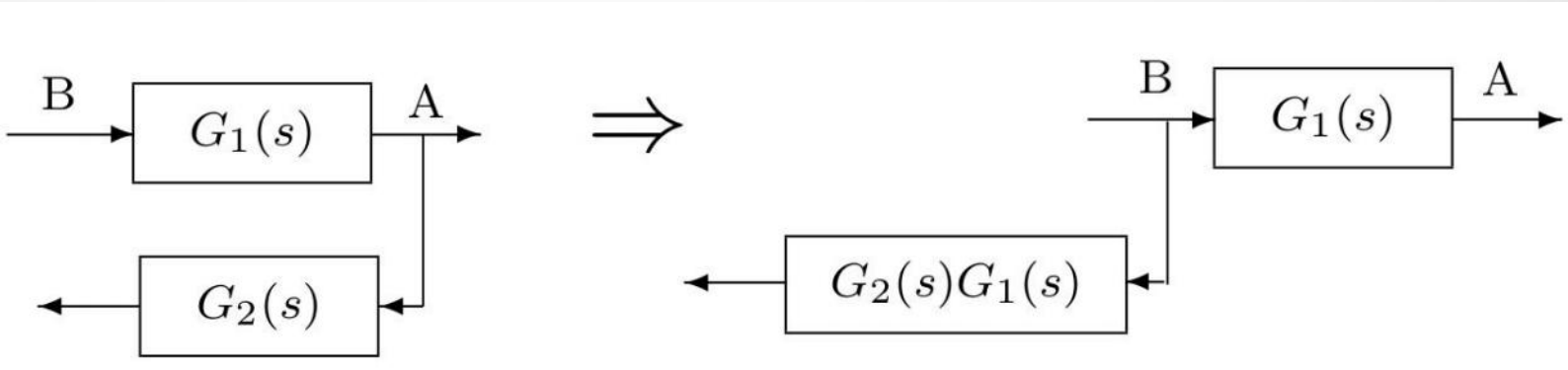
- 考虑模型



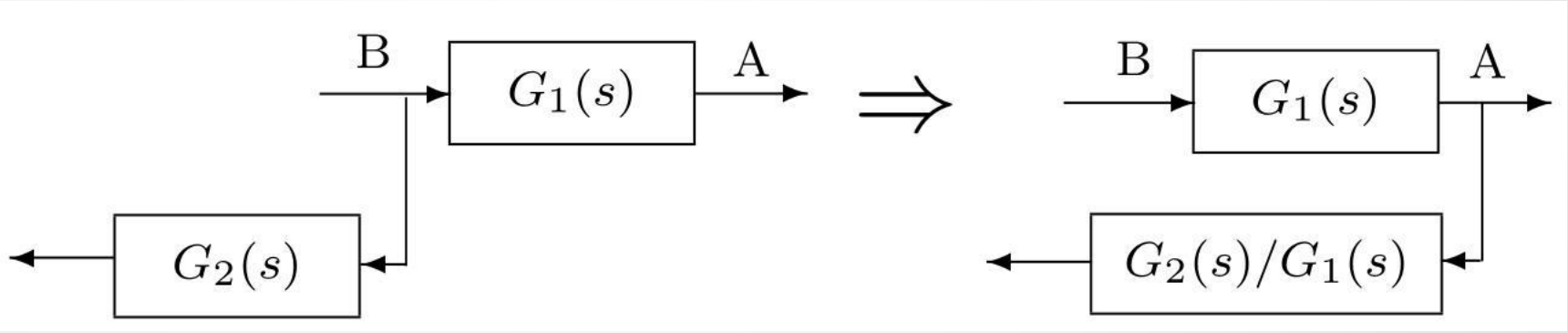
- 难点：A点在回路间，移至输出端

节点移动的等效处理

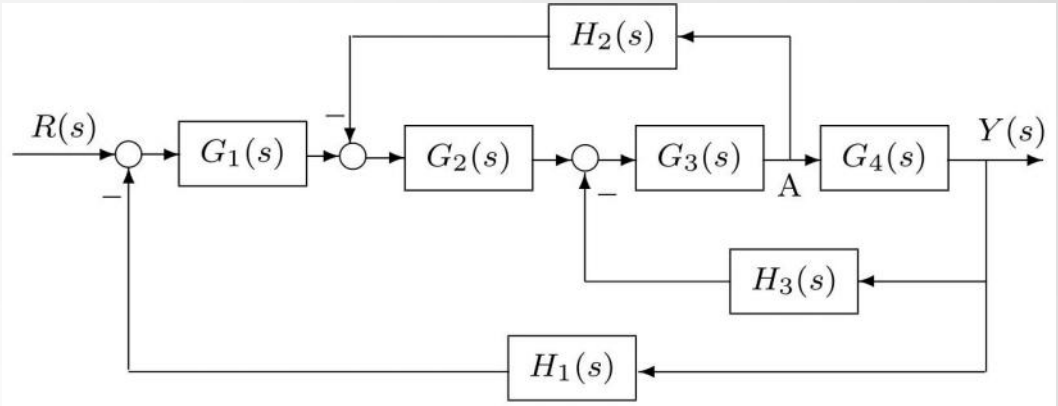
- 前向移动



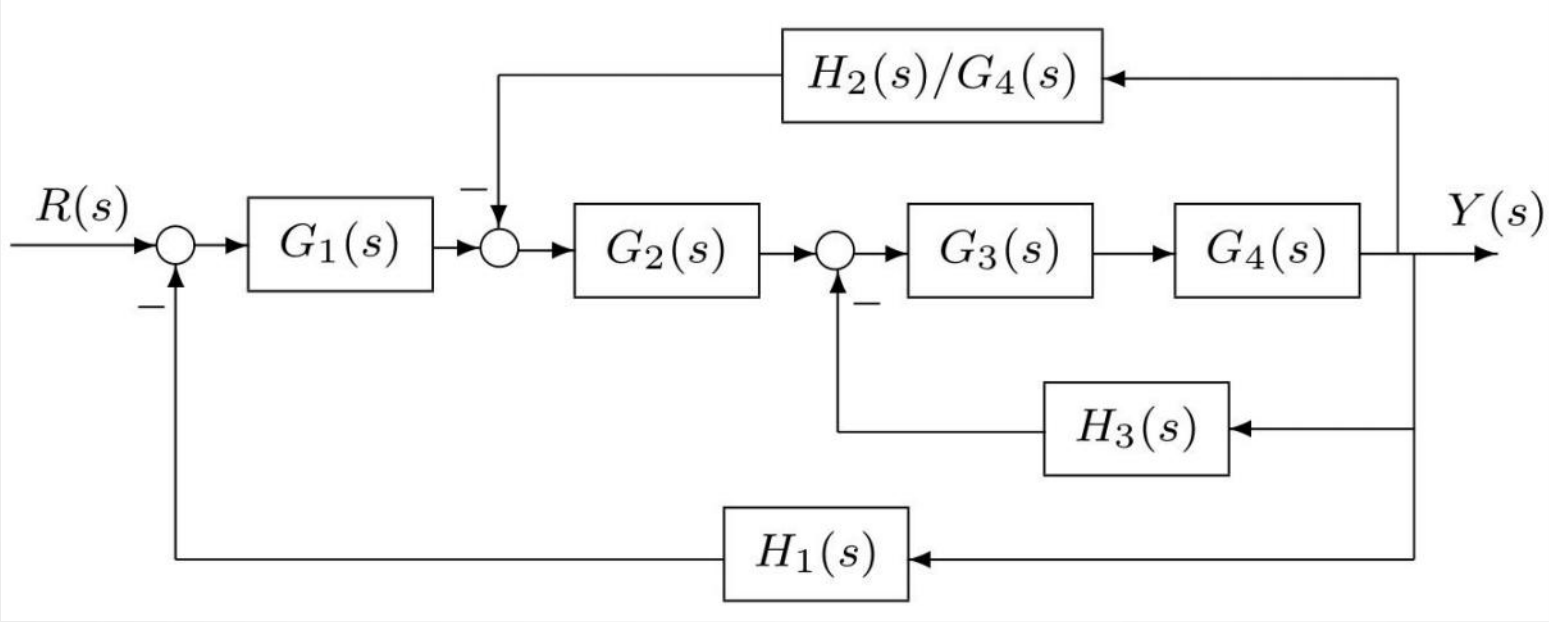
- 后向移动



例4-21 复杂模型化简

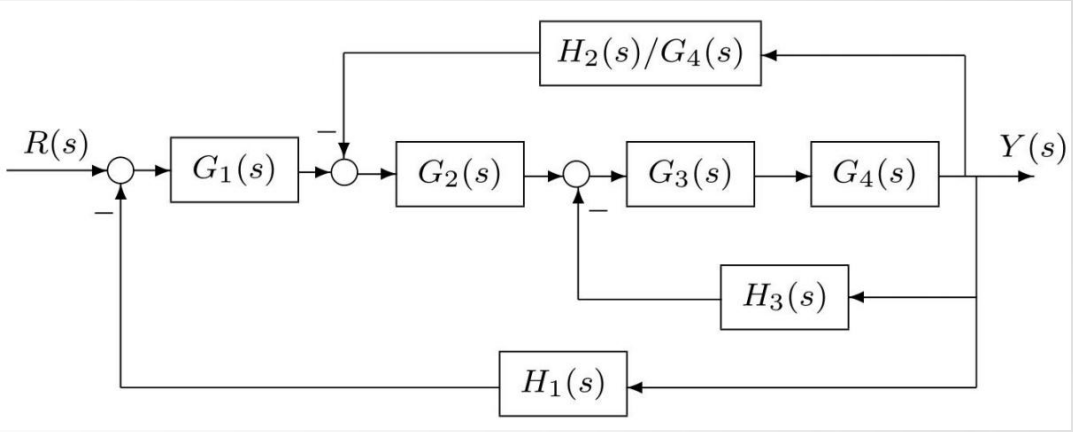


原系统可以移动



- 新支路模型 $H_2(s)/G_4(s)$

等效模型的化简



- 从 $R(s)$ 到 $Y(s)$ 的等效模型计算

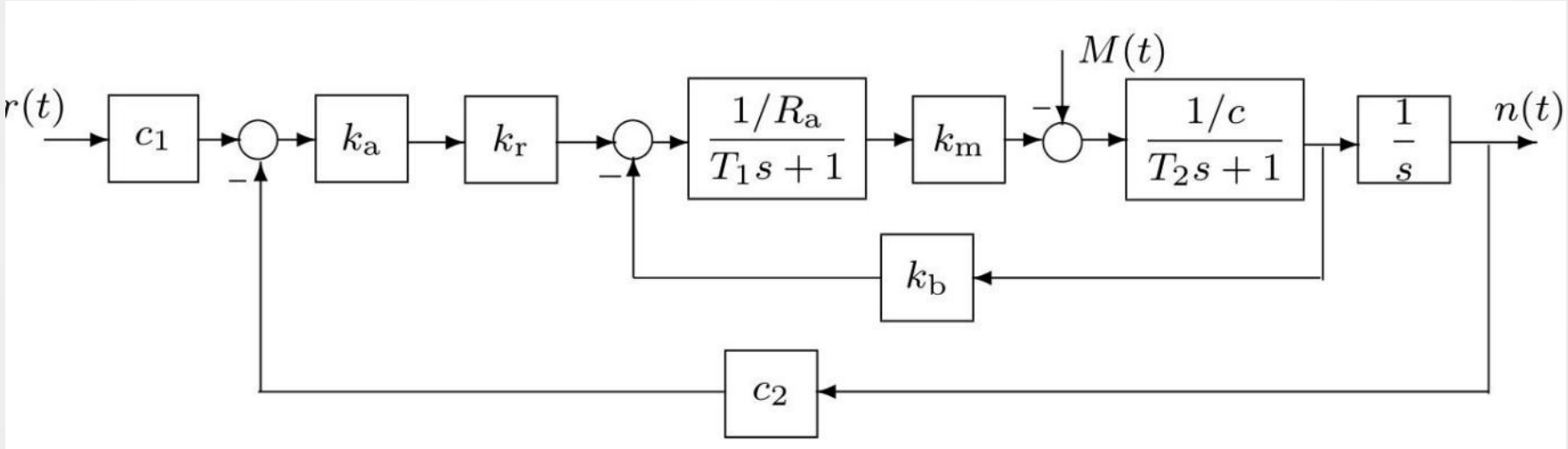
```
>> syms G1 G2 G3 G4 H1 H2 H3
   c1=feedbacksym(G4*G3,H3);
   c2=feedbacksym(c1*G2,H2/G4);
   G=feedbacksym(c2*G1,H1); pretty(G)
```

- 结果的数学表示

$$G(s)=\frac{G_2G_4G_3G_1}{1+G_4G_3H_3+G_3G_2H_2+G_2G_4G_3G_1H_1}$$

例4-22 直流电机拖动系统

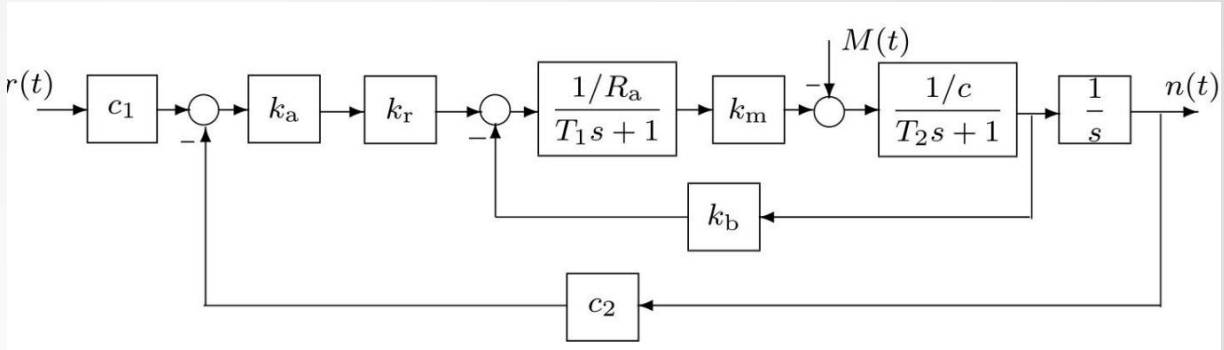
- 双输入系统



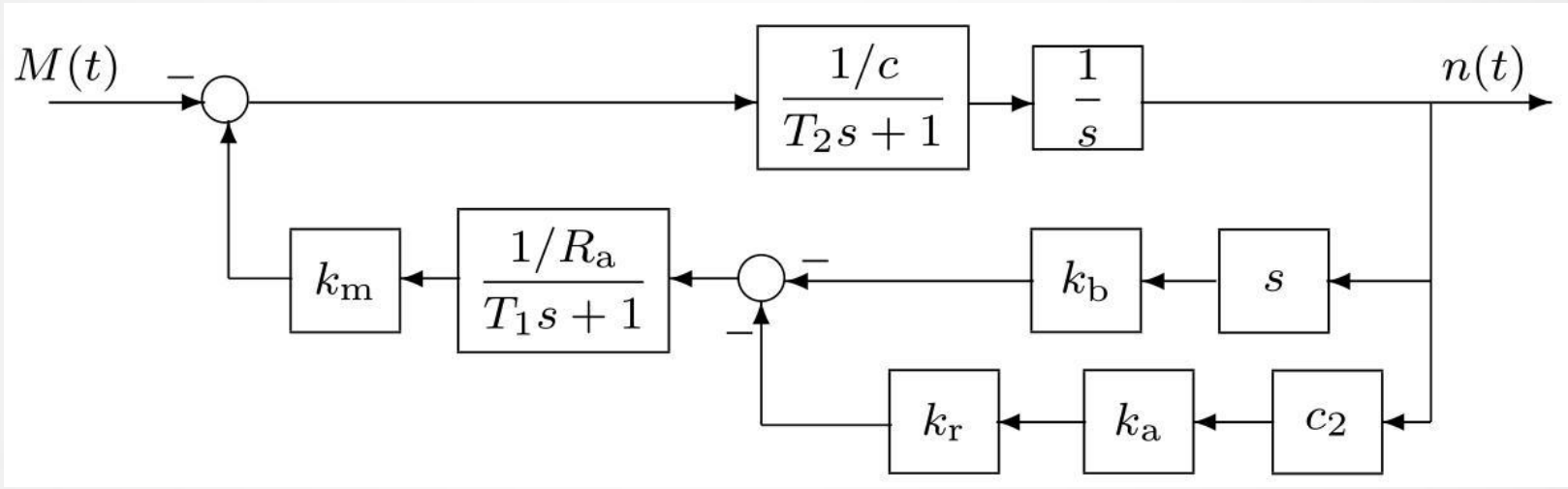
- 输入 $r(t)$ 单独激励

```
>> syms Ka Kr c1 c2 c Ra T1 T2 Km Kb s
Ga=feedbacksym(1/Ra/(T1*s+1)*Km*1/c/(T2*s+1),Kb);
G1=c1*feedbacksym(Ka*Kr*Ga/s,c2); G1=collect(G1,s)
```

另一个传递函数



- $M(t)$ 信号单独输入重新绘图



- 得出另一个传递函数

```
>> G2=-feedbacksym(1/c/(T2*s+1)/s, ...
     Km/Ra/(T1*s+1)*(Kb*s+c2*Ka*Kr));
   G2=collect(simplify(G2),s)
```

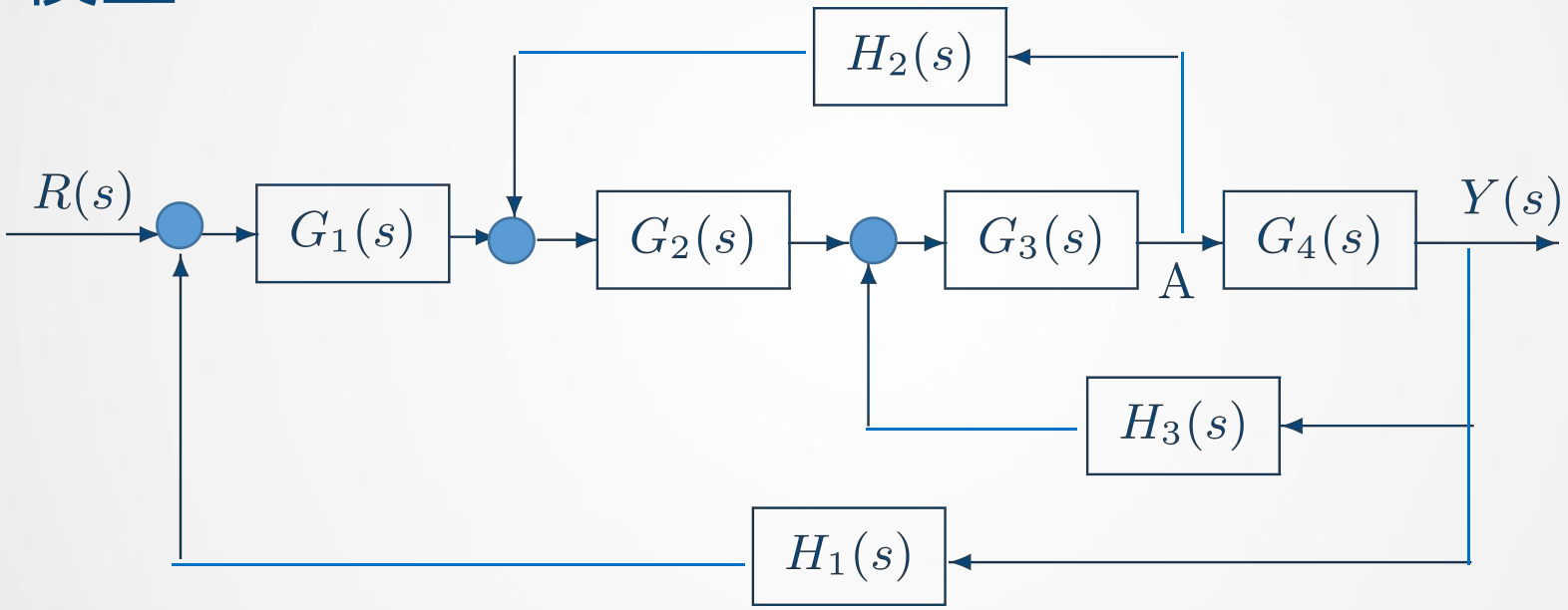
结果的数学形式

- 传递函数矩阵

$$\boldsymbol{G}^{\mathrm{T}}(s)=\begin{bmatrix}\dfrac{c_1k_{\mathrm{m}}k_{\mathrm{a}}k_{\mathrm{r}}}{R_{\mathrm{a}}cT_1T_2s^3+(R_{\mathrm{a}}cT_1+R_{\mathrm{a}}cT_2)s^2+(k_{\mathrm{m}}k_{\mathrm{b}}+R_{\mathrm{a}}c)s+k_{\mathrm{a}}k_{\mathrm{r}}k_{\mathrm{m}}c_2}\\ -\dfrac{(T_1s+1)R_{\mathrm{a}}}{R_{\mathrm{a}}cT_1T_2s^3+(R_{\mathrm{a}}cT_1+R_{\mathrm{a}}cT_2)s^2+(k_{\mathrm{m}}k_{\mathrm{b}}+R_{\mathrm{a}}c)s+k_{\mathrm{a}}k_{\mathrm{r}}k_{\mathrm{m}}c_2}\end{bmatrix}$$

节点移动时的等效变换

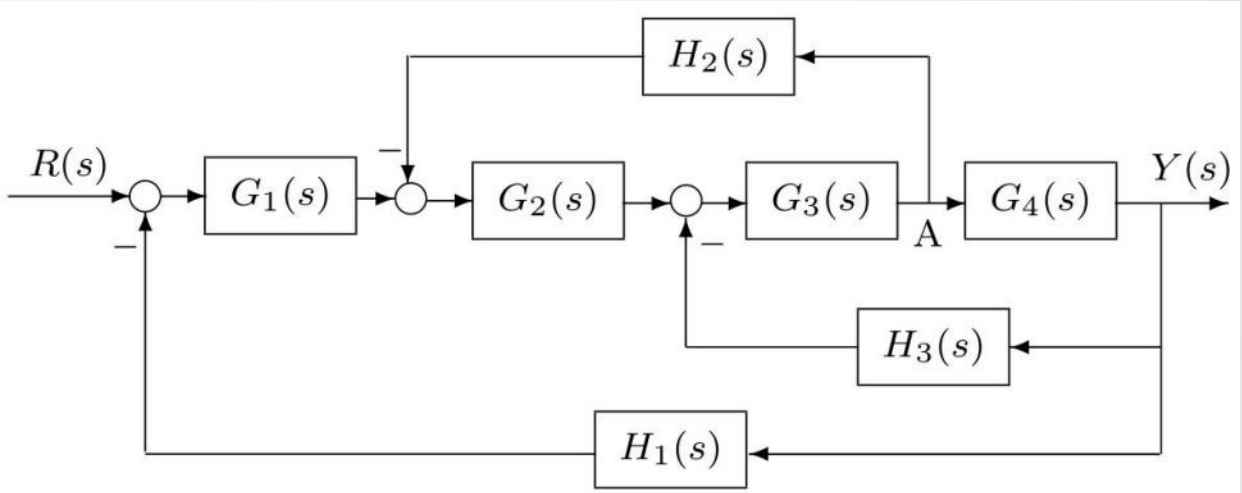
- 考虑模型

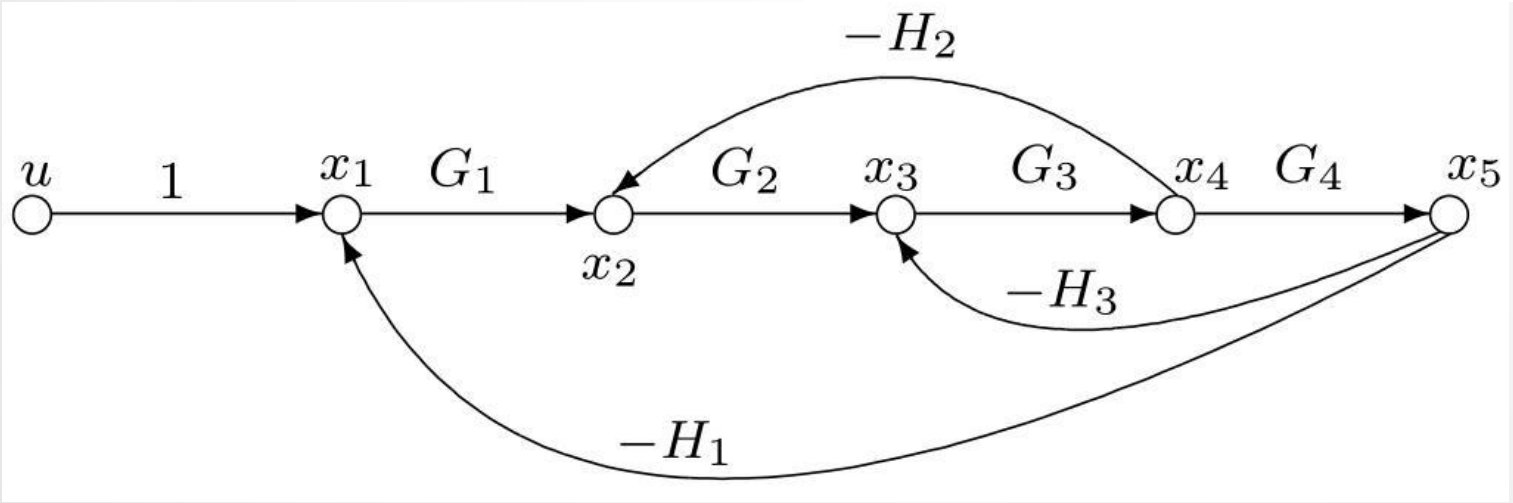


- 难点：需要手工移动节点，容易出错，需要更简单的方法

例4-23 框图的信号流图表示

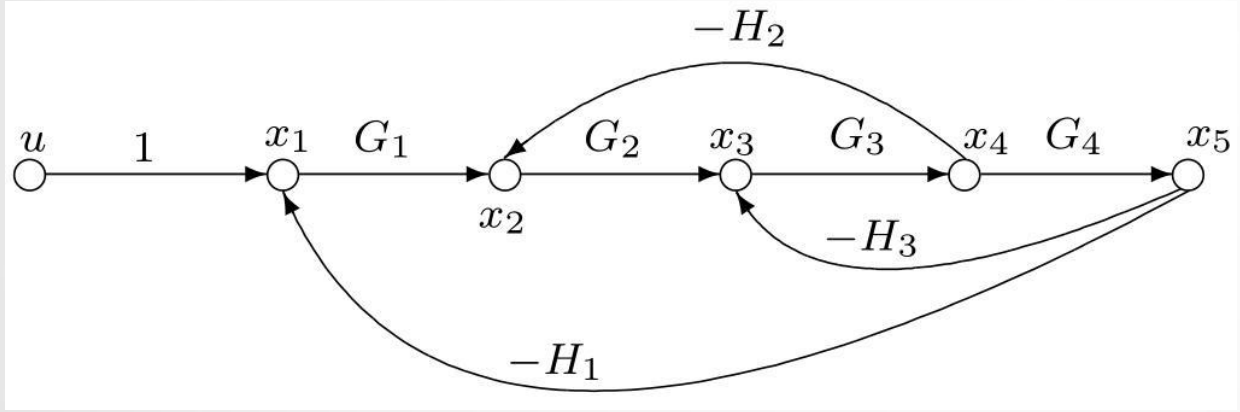
- 信号流图
  - 节点、通路、传递函数





写出节点方程

- 以流入节点的信号为准写出方程



$$
\begin{cases}
x_1 = u - H_1 x_5 \\
x_2 = G_1 x_1 - H_2 x_4 \\
x_3 = G_2 x_2 - H_3 x_5 \\
x_4 = G_3 x_3 \\
x_5 = G_4 x_4
\end{cases}
$$

由节点方程写出矩阵形式

- 直接写出

$$\begin{cases} x_1 = u - H_1 x_5 \\ x_2 = G_1 x_1 - H_2 x_4 \\ x_3 = G_2 x_2 - H_3 x_5 \\ x_4 = G_3 x_3 \\ x_5 = G_4 x_4 \end{cases}$$

$$\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{X} + \boldsymbol{P}\boldsymbol{U}$$

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 & -H_1 \\ G_1 & 0 & 0 & -H_2 & 0 \\ 0 & G_2 & 0 & 0 & -H_3 \\ 0 & 0 & G_3 & 0 & 0 \\ 0 & 0 & 0 & G_4 & 0 \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

由节点方程推导出传递函数矩阵

- 节点方程
  - 期望的传递函数 X/U
  - 精确写法 $G = XU^{-1}$

$$X = QX + PU \quad \rightarrow \quad (I - Q)X = PU$$

$$G = XU^{-1} = (I - Q)^{-1}P$$

  - 利用MATLAB提供的符号运算可以直接求解，无需预处理

例4-24 框图的直接化简

- 节点方程的矩阵形式

$$\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{X} + \boldsymbol{P}\boldsymbol{U}$$

- 其中

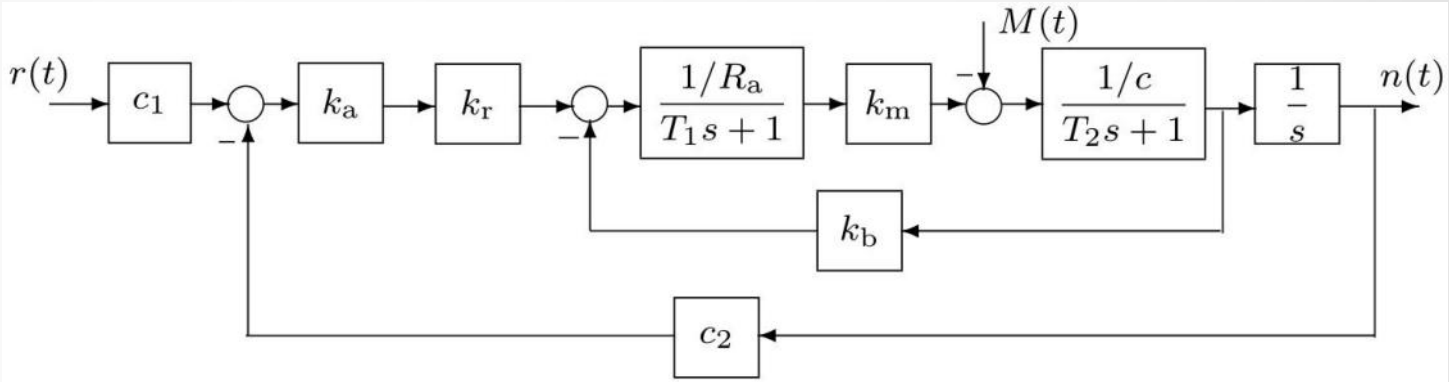
$$\boldsymbol{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 & -H_1 \\ G_1 & 0 & 0 & -H_2 & 0 \\ 0 & G_2 & 0 & 0 & -H_3 \\ 0 & 0 & G_3 & 0 & 0 \\ 0 & 0 & 0 & G_4 & 0 \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

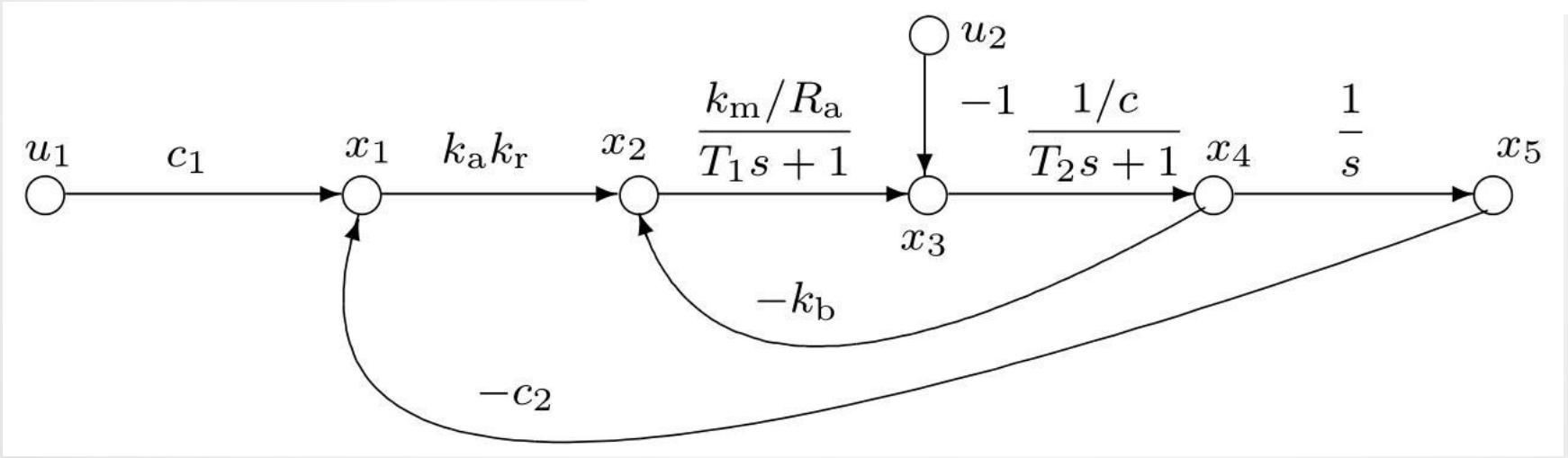
- MATLAB求解

```
>> syms G1 G2 G3 G4 H1 H2 H3
   Q=[0 0 0 0 -H1; G1 0 0 -H2 0; 0 G2 0 0 -H3;
      0 0 G3 0 0; 0 0 0 G4 0];
   P=[1 0 0 0 0]'; G=inv(eye(5)-Q)*P
```

例4-25 重新求解电力拖动系统模型问题

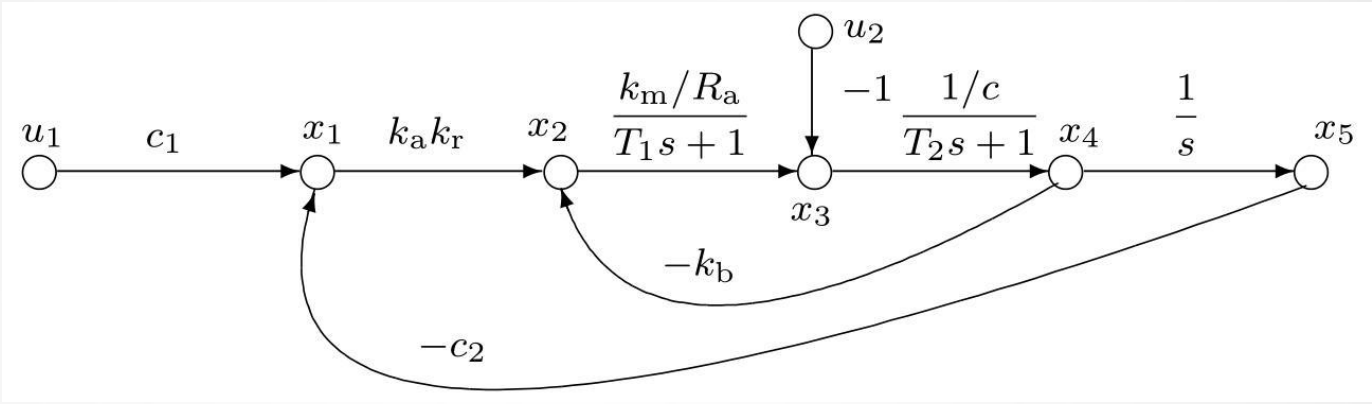
- 方框图信号流图
  - 双输入信号
  - 无需预处理





传递函数计算

- 信号流图



- 节点方程

$$
\begin{cases}
x_1 = c_1 u_1 - c_2 x_5 \\
x_2 = k_a k_r x_1 - k_b x_4 \\
x_3 = \dfrac{k_m/R_a}{T_1 s+1} x_2 - u_2 \\
x_4 = \dfrac{1/c}{T_2 s+1} x_3 \\
x_5 = \dfrac{1}{s} x_4
\end{cases}
$$

$$
\boldsymbol{Q} = \begin{bmatrix}
0 & 0 & 0 & 0 & -c_2 \\
k_a k_r & 0 & 0 & -k_b & 0 \\
0 & \dfrac{k_m/R_a}{T_1 s+1} & 0 & 0 & 0 \\
0 & 0 & \dfrac{1/c}{T_2 s+1} & 0 & 0 \\
0 & 0 & 0 & \dfrac{1}{s} & 0
\end{bmatrix},
\qquad
\boldsymbol{P} = \begin{bmatrix}
c_1 & 0 \\
0 & 0 \\
0 & -1 \\
0 & 0 \\
0 & 0
\end{bmatrix}
$$

MATLAB求解

- 相关方程

$$\boldsymbol{X}=\boldsymbol{Q}\boldsymbol{X}+\boldsymbol{P}\boldsymbol{U}$$

$$\boldsymbol{G}=\boldsymbol{X}\boldsymbol{U}^{-1}=(\boldsymbol{I}-\boldsymbol{Q})^{-1}\boldsymbol{P}$$

$$\boldsymbol{Q}=\begin{bmatrix} 0 & 0 & 0 & 0 & -c_2 \\ k_{\mathrm{a}}k_{\mathrm{r}} & 0 & 0 & -k_{\mathrm{b}} & 0 \\ 0 & \dfrac{k_{\mathrm{m}}/R_{\mathrm{a}}}{T_1 s+1} & 0 & 0 & 0 \\ 0 & 0 & \dfrac{1/c}{T_2 s+1} & 0 & 0 \\ 0 & 0 & 0 & \dfrac{1}{s} & 0 \end{bmatrix},\ \boldsymbol{P}=\begin{bmatrix} c_1 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- MATLAB直接求解

```
>> syms Ka Kr c1 c2 c Ra T1 T2 Km Kb s
   Q=[0 0 0 0 -c2; Ka*Kr 0 0 -Kb 0; 0 Km/Ra/(T1*s+1) 0 0 0
      0 0 1/c/(T2*s+1) 0 0; 0 0 0 1/s 0];
   P=[c1 0; 0 0; 0 -1; 0 0; 0 0]; W=inv(eye(5)-Q)*P; W(5,:)
```

模型转换小结

- 本章涉及的线性模型统称为LTI模型
- 不同的模型是可以相互转换，如
  - 连续化与离散化 c2d, d2c
  - 转换成 tf, zpk，则调用相应函数即可

状态方程实现小结

- 直接转换函数
  - 多变量系统、连续离散系统的统一处理 tf
  - 特例——描述符系统、内部延迟系统
- 将LTI模型转换成状态方程是不唯一的
- 系统非均衡实现 balreal
- 系统的最小实现 minreal

系统典型连接小结

- 三种典型连接
  - 串联连接 *，多变量系统注意顺序
  - 并联连接 +
  - 反馈连接 feedback，feedbacksym
- 可以处理LTI模型，也可以处理符号运算
- 特殊情况
  - 带有延迟的模型，可以自动处理为内部延迟ss
  - 不同域的先统一域再运算

方框图化简小结

- 需要手工处理
  - 节点的前向移动或后向移动
  - 复杂问题可能需要手工重新画图
  - 将图形化成明显的串并联、反馈结构
  - 嵌套使用 +,*, feedback， feedbacksym函数
- 需要更容易使用的工具

方框图代数化简小结

- 无需框图结构的手工处理
- 代数化简的步骤
  - 需要手工将框图表示成信号流图
  - 由信号流图可以写出节点方程
  - 由节点方程进行代数运算，可以得出化简模型
- 代数化简的优点
  - 手工处理比前面的方法更易于检验
  - 可以一次性得出所有节点的传递函数矩阵

Q & A

感谢您的聆听和反馈

国家精品课程/ 国家精品资源共享课程/ 国家级精品教材
国家级十一(二)五规划教材/ 教育部自动化专业教学指导委员会牵头规划系列教材

控制系统计算机辅助设计

第4章 线性控制系统的数学模型

主讲：修贤超

线性系统的数学模型

- 4.1 线性连续系统模型及MATLAB表示
- 4.2 线性离散时间系统的数学模型
- 4.3 线性模型的相互转换
- 4.4 方框图描述系统的化简
- 4.5 线性系统的模型降阶
- 4.6 线性系统的模型辨识

线性系统的模型降阶

- 模型降阶与最小实现
- 传递函数的经典降阶方法
  - 基于Pade近似的降阶方法与实现
  - Routh降阶方法
- 状态方程的降阶
  - 均衡实现的降阶方法
  - 最优Hankel范数的方法

最小实现与模型降阶

- 最小实现是数学意义下的严格化简（没有近似）
- 模型降阶是用低阶模型去逼近高阶模型
  - 最早由Edward J. Davison提出(1966)
- 本节介绍的方法
  - 传递函数的 Padé与 Routh 算法
  - 状态空间的降阶算法

模型降阶的需求

- 已知原始模型

$$G(s)=\frac{b_1 s^m+b_2 s^{m-1}+\cdots+b_m s+b_{m+1}}{a_1 s^n+a_2 s^{n-1}+a_3 s^{n-2}+\cdots+a_n s+a_{n+1}}$$

$k<n$

- 寻求降阶模型

$$G_{r/k}(s)=\frac{\beta_1 s^r+\beta_2 s^{r-1}+\cdots+\beta_{r+1}}{\alpha_1 s^k+\alpha_2 s^{k-1}+\cdots+\alpha_k s+\alpha_{k+1}}$$

- 为简单起见假设 $\alpha_{k+1}=1$

时间矩量的计算

- 展开原模型

$$G(s) = c_0 + c_1 s + c_2 s^2 + \cdots$$

- 其中时间矩量可以递推求出

$$c_0 = b_{k+1}, \quad \text{and} \quad c_i = b_{k+1-i} - \sum_{j=0}^{i-1} c_j a_{n+1-i+j}, \quad i = 1, 2, \cdots$$

- 若已知状态方程模型

$$c_i = \frac{1}{i!} \frac{\mathrm{d}^i G(s)}{\mathrm{d}s^i}\bigg|_{s=0} = -\boldsymbol{C}\boldsymbol{A}^{-(i+1)}\boldsymbol{B}, \quad i = 0, 1, \cdots$$

Padé降阶的思想

- 时间矩量的MATLAB求解

```
function M=timmomt(G,k)
G=ss(G); C=G.c; B=G.b; iA=inv(G.a);
iA1=iA; M=zeros(1,k);
for i=1:k, M(i)=-C*iA1*B; iA1=iA*iA1; end
```

- Padé 降阶思想：保留前 $r+k+1$ 个时间矩量

$$\sum c_i s^i = \frac{\sum \beta_i s^i}{\sum \alpha_i s^i} \quad \Rightarrow \quad \sum c_i s^i \sum \alpha_i s^i = \sum \beta_i s^i$$

Padé降阶模型系数推到

- 对比系数，则

$$
\begin{cases}
\beta_{r+1} = c_0 \\
\beta_r = c_1 + \alpha_k c_0 \\
\qquad\qquad \vdots \\
\beta_1 = c_r + \alpha_k c_{r-1} + \cdots + \alpha_{k-r+1} c_0 \\
0 = c_{r+1} + \alpha_k c_r + \cdots + \alpha_{k-r} c_0 \\
0 = c_{r+2} + \alpha_k c_{r+1} + \cdots + \alpha_{k-r-1} c_0 \\
\qquad\qquad \vdots \\
0 = c_{k+r} + \alpha_k c_{k+r-1} + \cdots + \alpha_2 c_{r+1} + \alpha_1 c_r
\end{cases}
$$

Padé降阶公式

- 这样可以得出

$$\begin{bmatrix} c_r & c_{r-1} & \cdots & . \\ c_{r+1} & c_r & \cdots & . \\ \vdots & \vdots & \ddots & \vdots \\ c_{k+r-1} & c_{k+r-2} & \cdots & c_r \end{bmatrix} \begin{bmatrix} \alpha_k \\ \alpha_{k-1} \\ \vdots \\ \alpha_1 \end{bmatrix} = - \begin{bmatrix} c_{r+1} \\ c_{r+2} \\ \vdots \\ c_{k+r} \end{bmatrix}$$

$$\begin{bmatrix} \beta_{r+1} \\ \beta_r \\ \vdots \\ \beta_1 \end{bmatrix} = \begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_r & c_{r-1} & \cdots & c_0 \end{bmatrix} \begin{bmatrix} 1 \\ \alpha_k \\ \vdots \\ \alpha_{k-r+1} \end{bmatrix}$$

- 可以得出降阶模型的分子、分母系数

Padé降阶的求解函数

- Padé 降阶求解函数

```
function G_r=pademod(G_Sys,r,k)
c=timmomt(G_Sys,r+k+1); G_r=pade_app(c,r,k);

function Gr=pade_app(c,r,k)
w=-c(r+2:r+k+1)';
vv=[c(r+1:-1:1)'; zeros(k-1-r,1)];
W=rot90(hankel(c(r+k:-1:r+1),vv));
V=rot90(hankel(c(r:-1:1)));
x=[1 (W\w)']; dred=x(k+1:-1:1)/x(k+1);
y=[c(1) x(2:r+1)*V'+c(2:r+1)];
nred=y(r+1:-1:1)/x(k+1); Gr=tf(nred,dred);
```

例4-26 Padé 近似举例

- 原始模型 $G(s)=\dfrac{s^3+7s^2+11s+5}{s^4+7s^3+21s^2+37s+30}$
- Padé 近似
  - ```
    >> G=tf([1,7,11,5],[1,7,21,37,30]);
       Gr=pademod(G,1,2), step(G,Gr,'--')
    ```
  - ```
    >> bode(G,Gr,'--')
    ```
- 用二阶模型近似原始的四阶模型

$$G_{\mathrm{r}}(s)=\frac{0.8544s+0.6957}{s^2+1.091s+4.174}$$

例4-27 Padé降阶的反例

- 原始模型

$$G(s)=\frac{0.067s^5+0.6s^4+1.5s^3+2.016s^2+1.55s+0.6}{0.067s^6+0.7s^5+3s^4+6.67s^3+7.93s^2+4.63s+1}$$

$$=\frac{(s+5.92)(s+1.221)(s+0.897)(s^2+0.9171s+1.381)}{(s+2.805)(s+1.856)(s+1.025)(s+0.501)(s^2+4.261s+5.582)}$$

- 稳定模型，但降阶模型不稳定

```
>> num=[0.067,0.6,1.5,2.016,1.66,0.6];
   den=[0.067 0.7 3 6.67 7.93 4.63 1]; G=tf(num,den);
   zpk(G), Gr=pademod(G,1,3); zpk(Gr)
```

Routh 降阶方法与实例

- Routh算法（较烦琐，从略）

```
function G_r=routhmod(G_Sys,nr)
num=G_Sys.num{1}; den=G_Sys.den{1}; n0=length(den); n1=length(num);
a1=den(end:-1:1); b1=[num(end:-1:1) zeros(1,n0-n1-1)];
for k=1:n0-1,
   k1=k+2; alpha(k)=a1(k)/a1(k+1); beta(k)=b1(k)/a1(k+1);
   for i=k1:2:n0-1,
      a1(i)=a1(i)-alpha(k)*a1(i+1); b1(i)=b1(i)-beta(k)*a1(i+1);
end, end
nn=[]; dd=[1]; nn1=beta(1); dd1=[alpha(1),1]; nred=nn1; dred=dd1;
for i=2:nr,
   nred=[alpha(i)*nn1, beta(i)]; dred=[alpha(i)*dd1, 0];
   n0=length(dd); n1=length(dred); nred=nred+[zeros(1,n1-n0),nn];
   dred=dred+[zeros(1,n1-n0),dd];
   nn=nn1; dd=dd1; nn1=nred; dd1=dred;
end
G_r=tf(nred(nr:-1:1),dred(end:-1:1));
```

例4-28 Routh降阶

- 仍考虑稳定模型

$$G(s)=\frac{0.067s^5+0.6s^4+1.5s^3+2.016s^2+1.55s+0.6}{0.067s^6+0.7s^5+3s^4+6.67s^3+7.93s^2+4.63s+1}$$

- MATLAB求解

```
>> num=[0.067,0.6,1.5,2.016,1.66,0.6];
   den=[0.067 0.7 3 6.67 7.93 4.63 1];
   G=tf(num,den); Gr=zpk(routhmod(G,3)), step(G,Gr,'--')
```

```
>> bode(G,Gr,'--')
```

- Routh算法降阶后能保证降阶模型稳定性

状态方程模型的降阶算法

- 均衡实现模型的降阶算法

$$\begin{bmatrix}\dot{\boldsymbol{x}}_1\\ \dot{\boldsymbol{x}}_2\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}+\begin{bmatrix}\boldsymbol{B}_1\\ \boldsymbol{B}_2\end{bmatrix}u,\quad y=[\boldsymbol{C}_1\ \ \boldsymbol{C}_2]\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}+Du$$

令 $\dot{\boldsymbol{x}}_2=0$ ↓ 并消去x_2

$$\begin{cases}\dot{\boldsymbol{x}}_1=\left(\boldsymbol{A}_{11}-\boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21}\right)\boldsymbol{x}_1+\left(\boldsymbol{B}_1-\boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{B}_2\right)u\\ y=\left(\boldsymbol{C}_1-\boldsymbol{C}_2\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21}\right)\boldsymbol{x}_1+\left(D-\boldsymbol{C}_2\boldsymbol{A}_{22}^{-1}\boldsymbol{B}_2\right)u.\end{cases}$$

G_{r}=`modred(`G`,elim)`

其他降阶模型计算

- MATLAB求解函数
  - Schur降阶

$$G_{\mathrm{r}}=\mathtt{schmr}(G,1,k)$$

  - 最优Hankel范数降阶

$$G_{\mathrm{r}}=\mathtt{ohklmr}(G,1,k)$$

- 控制系统工具箱函数

例4-34 状态方程降阶举例

- 原始模型

$$G(s)=\frac{1+8.8818s+29.9339s^2+67.087s^3+80.3787s^4+68.6131s^5}{1+7.6194s+21.7611s^2+28.4472s^3+16.5609s^4+3.5338s^5+0.0462s^6}$$

- 降阶模型与比较

```
>> num=[68.6131,80.3787,67.087,29.9339,8.8818,1];
   den=[0.0462,3.5338,16.5609,28.4472,21.7611,7.6194,1];
   G=ss(tf(num,den));
   G1=zpk(schmr(G,1,3)), G2=zpk(ohklmr(G,1,3))
   step(G,G1,'--',G2,':')
>> bode(G,G1,'--',G2,':')
```

时间延迟模型的 Padé 近似

- 纯延迟的Padé近似方法

$$P_{k,\tau}(s)=\frac{1-\tau s/2+p_2(\tau s)^2-p_3(\tau s)^3+\cdots+(-1)^{n+1}p_n(\tau s)^k}{1+\tau s/2+p_2(\tau s)^2+p_3(\tau s)^3+\cdots+p_n(\tau s)^k}$$

- Padé近似函数 $[n,d]=\mathrm{pade}(\tau,k)$

$$G_1=\mathrm{pade}(G,k)$$

- 多变量系统的Padé近似

$$G_1=\mathrm{pade}(\boldsymbol{G},n_{\mathrm{i}},n_{\mathrm{o}},k)$$

不同分子分母的Padé近似

- 算法：逼近 $\mathrm{e}^{-\tau s} = 1 - \frac{1}{1!}\tau s + \frac{1}{2!}\tau^2 s^2 - \frac{1}{3!}\tau^3 s^3 + \cdots$
- 编写 MATLAB 函数

```
function [n,d]=paderm(tau,r,k)
c(1)=1; for i=2:r+k+1, c(i)=-c(i-1)*tau/(i-1); end
Gr=pade_app(c,r,k); n=Gr.num{1}(k-r+1:end);
d=Gr.den{1};
```

- 其中 r/k 任意选择
- 可以选择 $0/k$，以避免非最小相位模型

例4-29 纯延迟模型的近似

纯延迟模型 $G(s)=\mathrm{e}^{-s}$

- MATLAB求解

```
>> tau=1; [n1,d1]=pade(tau,3); G1=tf(n1,d1)
   [n2,d2]=paderm(tau,1,3); G2=tf(n2,d2)
```

- 拟合结果

$$G_1(s)=\frac{-s^3+12s^2-60s+120}{s^3+12s^2+60s+120}$$

$$G_2(s)=\frac{-6s+24}{s^3+6s^2+18s+24}$$

例4-30 已知带有延迟的线性模型

带有延迟的线性模型 $G(s)=\dfrac{3s+1}{(s+1)^3}\mathrm{e}^{-2s}$

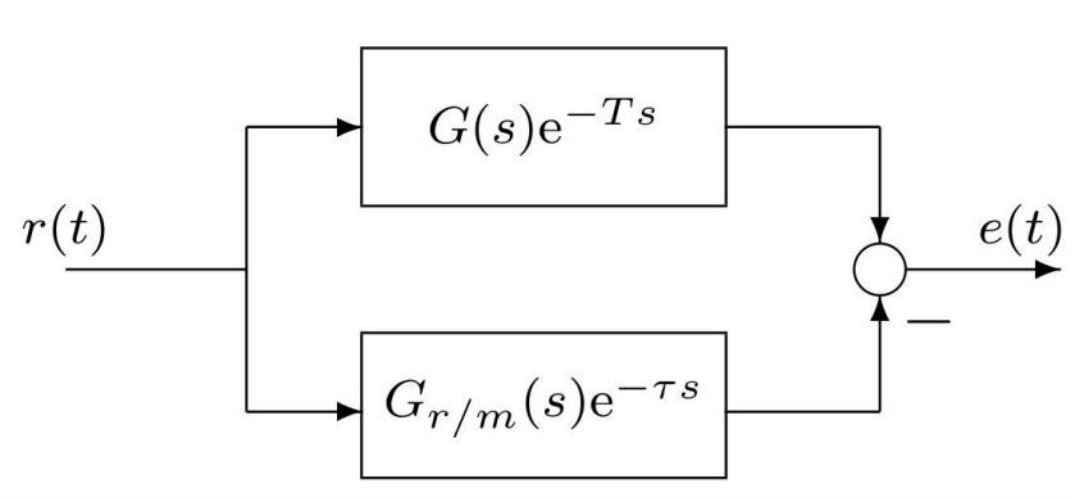
- 可以得出近似模型

```
>> G=tf([3 1],[1 3 3 1]);
   [n,d]=paderm(2,1,3); Gr=G*tf(n,d);
   G.ioDelay=2; Gr1=pade(G,2)
   step(G,Gr,'--',Gr1,':')
```

```
>> bode(G,Gr,'--',Gr1,':')
```

什么是最好的降阶模型?

- 客观的最优降阶指标提出



- 误差最小指标
- 性能指标定义

$$\sigma_h^2 = \int_0^\infty h^2(t)\mathrm{d}t = \int_0^\infty w^2(t)e^2(t)\mathrm{d}t$$

次最优降阶的MATLAB实现

- 决策变量

$$\boldsymbol{\theta} = (\alpha_1, \cdots, \alpha_m, \beta_1, \cdots, \beta_{r+1}, \tau)$$

- 目标函数的计算

$$J = \min_{\boldsymbol{\theta}} \left[\int_0^{\infty} w^2(t) \hat{e}^2(t, \boldsymbol{\theta}) \mathrm{d}t \right]$$

- MATLAB实现见教材，核心为fminsearch函数
  - 调用格式　G_r=opt_app(G,r,m,key,G_0)

例4-31 复杂模型的最优降阶

- 前面介绍的模型

$$G(s)=\frac{1+8.8818s+29.9339s^2+67.087s^3+80.3787s^4+68.6131s^5}{1+7.6194s+21.7611s^2+28.4472s^3+16.5609s^4+3.5338s^5+0.0462s^6}$$

- 次最优降阶

```
>> num=[68.6131,80.3787,67.087,29.9339,8.8818,1];
   den=[0.0462,3.5338,16.5609,28.4472,21.7611,7.6194,1];
   G=tf(num,den); Gr=zpk(opt_app(G,2,3,0))
   step(G,Gr,'--')
```

```
>> bode(G,Gr,'--')
```

例4-32 高阶全极点模型

- 高阶模型

$$G(s)=\frac{432}{(5s+1)(2s+1)(0.7s+1)(s+1)(0.4s+1)}$$

- 最优降阶

```
>> s=tf('s');
   G=432/(5*s+1)/(2+s+1)/(0.7*s+1)/(s+1)/(0.4*s+1);
   Gr=zpk(opt_app(G,0,2,1))
   step(G,Gr,'--')
```

例4-33 非最小相位系统

- 系统模型

$$G(s)=\frac{10s^3-60s^2+110s+60}{s^4+17s^2+82s^2+130s+100}$$

- 模型降阶

```
>> G=tf([10 -60 110 60],[1 17 82 130 100]);
   Gr=opt_app(G,1,2,1); Gr1=opt_app(G,1,2,0);
   step(G,Gr,'--',Gr1,':')
```

线性系统的数学模型

- 4.1 线性连续系统模型及MATLAB表示
- 4.2 线性离散时间系统的数学模型
- 4.3 线性模型的相互转换
- 4.4 方框图描述系统的化简
- 4.5 线性系统的模型降阶
- 4.6 线性系统的模型辨识

系统辨识

- 系统模型建立的方法
  - 由数学、物理规则推导
  - 由已知实测数据获得系统模型的方法
- 实测数据
  - 时域响应数据、频率响应数据
- 本节主要内容
  - 离散系统辨识方法
  - 辨识模型阶次的选择、辨识信号生成
  - 连续系统辨识、多变量系统辨识
  - 离散系统在线辨识简介

离散系统的模型辨识

- 离散传递函数模型

$$G\left(z^{-1}\right)=\frac{b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_nz^{-n}}z^{-d}$$

- z^{-1}是一步延迟的算子
- 对应的差分方程模型

$$y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_ny(t-n)$$

$$=b_1u(t-d)+b_2u(t-d-1)+\cdots+b_mu(t-d-m+1)+\varepsilon(t)$$

- $\varepsilon(t)$ 是辨识模型的残差信号

系统辨识的算法基础

- 已知实测信号
  - 输入　$\boldsymbol{u} = [u(1), u(2), \cdots, u(M)]^{\mathrm{T}}$
  - 输出　$\boldsymbol{y} = [y(1), y(2), \cdots, y(M)]^{\mathrm{T}}$
- 由数据可以得出

$$
\begin{aligned}
y(1) &= -a_1 y(0) - \cdots - a_n y(1-n) + b_1 u(1-d) + \cdots + b_m u(2-m-d) + \varepsilon(1) \\
y(2) &= -a_1 y(1) - \cdots - a_n y(2-n) + b_1 u(2-d) + \cdots + b_m u(3-m-d) + \varepsilon(2) \\
&\vdots \\
y(M) &= -a_1 y(M-1) - \cdots - a_n y(M-n) + b_1 u(M-d) + \\
&\qquad \cdots + b_m u(M-m-d+1) + \varepsilon(M)
\end{aligned}
$$

- 线性代数方程　$\boldsymbol{y} = \boldsymbol{\Phi\theta} + \boldsymbol{\varepsilon}$

最小二乘系统辨识

- 矩阵形式 $\boldsymbol{y}=\boldsymbol{\Phi\theta}+\boldsymbol{\varepsilon}$

$$\boldsymbol{\Phi}=\begin{bmatrix} y(0) & \cdots & y(1-n) & u(1-d) & \cdots & u(m-d) \\ y(1) & \cdots & y(2-n) & u(2-d) & \cdots & u(1+m-d) \\ \vdots & & \vdots & \vdots & & \vdots \\ y(M-1) & \cdots & y(M-n) & u(M-d) & \cdots & u(M+1-m-d) \end{bmatrix}$$

$$\boldsymbol{\theta}^{\mathrm{T}}=[-a_1,-a_2,\cdots,-a_n,b_1,\cdots,b_m],\boldsymbol{\varepsilon}^{\mathrm{T}}=[\varepsilon(1),\cdots,\varepsilon(M)]$$

- 定义残差最小指标 $\min\limits_{\boldsymbol{\theta}}\sum\limits_{i=1}^{M}\varepsilon^2(i)$
- 最小二乘解 $\boldsymbol{\theta}=[\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

辨识问题的MATLAB直接求解

- 系统辨识工具箱直接求解
  - 已知时域数据

$$T=\texttt{arx}([\boldsymbol{y},\boldsymbol{u}],\ [m,n,d])$$

  - 已知时域数据对象

$$T=\texttt{arx}(\texttt{dat},\ [m,n,d])$$

- T 为结构体变量，T.a, T.b, tf(T)
- 当然由前面的公式也能直接求解

例4-37 系统辨识计算

实测数据

文件 c4dat1.mat

t	$u(t)$	$y(t)$	t	$u(t)$	$y(t)$	t	$u(t)$	$y(t)$
0	1.4601	0	1.6	1.4483	16.411	3.2	1.056	11.871
0.1	0.8849	0	1.7	1.4335	14.336	3.3	1.4454	13.857
0.2	1.1854	8.7606	1.8	1.0282	15.746	3.4	1.0727	14.694
0.3	1.0887	13.194	1.9	1.4149	18.118	3.5	1.0349	17.866
0.4	1.413	17.41	2	0.7463	17.784	3.6	1.3769	17.654
0.5	1.3096	17.636	2.1	0.9822	18.81	3.7	1.1201	16.639
0.6	1.0651	18.763	2.2	1.3505	15.309	3.8	0.8621	17.107
0.7	0.7148	18.53	2.3	0.7078	13.7	3.9	1.2377	16.537
0.8	1.3571	17.041	2.4	0.8111	14.818	4	1.3704	14.643
0.9	1.0557	13.415	2.5	0.8622	13.235	4.1	0.7157	15.086
1	1.1923	14.454	2.6	0.8589	12.299	4.2	1.245	16.806
1.1	1.3335	14.59	2.7	1.183	11.6	4.3	1.0035	14.764
1.2	1.4374	16.11	2.8	0.9177	11.607	4.4	1.3654	15.498
1.3	1.2905	17.685	2.9	0.859	13.766	4.5	1.1022	14.679
1.4	0.841	19.498	3	0.7122	14.195	4.6	1.2675	16.655
1.5	1.0245	19.593	3.1	1.2974	13.763	4.7	1.0431	16.63

如何辨识系统模型?

- 基于MATLAB的求解（选择阶次组合4,4,1）

```
>> load c4dat1
   T=arx([y u],[4 4 1])
```

- 离散传递函数模型提取

```
>> G=tf(T), G.Ts=0.1
```

- 数学形式

$$H(z)=\frac{4.83\times10^{-8}z^3+6z^2-0.5999z-0.1196}{z^4-z^3+0.25z^2+0.25z-0.125}$$

辨识效果评价

- 样本点的输入方法

```
>> U=iddata(y,u,0.1);
   T=arx(U,[4,4,1]); H=tf(T)
```

- 如何评价？
  - 用已知输入信号去激励辨识模型
  - 比较输出与实际输出信号
- MATLAB求解

```
>> t=0:0.1:4.7; lsim(G,u,t);
   hold on; plot(t,y,'o')
```

由解方程的方法直接辨识

- 给出辨识语句 $y = \Phi\theta + \varepsilon$

```
>> Phi=[[0;y(1:end-1)] [0;0;y(1:end-2)],...
        [0;0;0; y(1:end-3)] [0;0;0;0;y(1:end-4)],...
        [0;u(1:end-1)] [0;0;u(1:end-2)],...
        [0;0;0; u(1:end-3)] [0;0;0;0;u(1:end-4)]];
    T=Phi\y; T'
    Gd=tf(ans(5:8),[1,-ans(1:4)],'Ts',0.1)
```

- 数学模型

$$G(z) = \frac{-5.824 \times 10^{-7} z^3 + 6z^2 - 0.5999z - 0.1196}{z^4 - z^3 + 0.25z^2 + 0.25z - 0.125}$$

系统辨识的图形用户界面

- 启动界面 systemIdentification
  - 输入样本点
  - 选择阶次
  - 直接辨识
- 无需给出命令、函数调用

辨识模型的阶次选择

- 为什么上个例子选择4,4,1?
- Akaike准则（Akaike‘s information criterion）

$$\mathrm{AIC} = \lg\left\{\det\left[\frac{1}{M}\sum_{i=1}^{M}\epsilon(i,\boldsymbol{\theta})\epsilon^{\mathrm{T}}(i,\boldsymbol{\theta})\right]\right\} + \frac{k}{M}$$

- MATLAB求解函数 v=`aic`(H)
- 根据该准则的变化情况选择合适阶次
  - 后面通过例子演示

例4-38 阶次选择

- 前面的例子，由已知数据辨识模型
  - 尝试不同的延迟常数 $d = 0, 1, 2$
  - 尝试不同的阶次组合 $1 \leqslant m, n \leqslant 7$
  - 用循环结构得出AIC表, 从表中观测应该如何选择阶次

```
>> U=iddata(y,u,0.1);
   for n=1:7, for m=1:7
      T=arx(U,[n,m,0]); TAic0(n,m)=aic(T);
      T=arx(U,[n,m,1]); TAic1(n,m)=aic(T);
      T=arx(U,[n,m,2]); TAic2(n,m)=aic(T);
   end, end
```

AIC表

- 可行的阶次
  - [4,5,0]
  - [4,4,1]
  - [4,3,2]
- AIC变化趋势

延迟步数为 $d=0$							
n	$m=1$	2	3	4	5	6	7
1	1.3487	1.3738	−0.23458	−0.63291	−1.0077	−1.5346	−2.61
2	1.2382	1.1949	−2.0995	−2.3513	−4.9058	−5.2429	−7.4246
3	1.0427	1.0427	−2.8743	−3.4523	−5.4678	−5.6186	−7.7328
4	1.0223	1.0345	−7.8505	−10.504	−20.729	−20.942	−20.946
5	1.0079	1.0287	−10.025	−13.396	−20.941	−20.982	−21.002
6	1.0293	1.0575	−13.658	−18.931	−20.944	−21.002	−21.125
7	0.98503	1.0261	−16.607	−20.701	−20.976	−20.996	−21.088
延迟步数为 $d=1$							
1	1.484	−0.25541	−0.66303	−1.0494	−1.57	−2.6414	−3.4085
2	1.346	−2.1263	−2.3685	−4.9326	−5.2359	−7.4658	−7.6678
3	1.0658	−2.8886	−3.4758	−5.4795	−5.6407	−7.7744	−7.9316
4	1.0329	−7.8839	−10.53	−20.733	−20.973	−20.984	−20.9737
5	1.0043	−10.034	−13.406	−20.971	−21.002	−21.037	−21.0356
6	1.023	−13.694	−18.965	−20.982	−21.037	−21.148	−21.1105
7	0.9909	−16.6423	−20.7387	−21.0160	−21.0324	−21.1105	−21.1115
延迟步数为 $d=2$							
1	−0.29215	−0.70464	−1.0849	−1.6057	−2.6827	−3.415	−3.5863
2	−2.1672	−2.4101	−4.9737	−5.2763	−7.477	−7.7083	−10.2034
3	−2.929	−3.5109	−5.5163	−5.6663	−7.8124	−7.9722	−10.5894
4	−7.9075	−10.57	−20.775	−21.013	−21.026	−21.015	−20.9850
5	−10.07	−13.438	−21.011	−21.036	−21.079	−21.077	−21.0617
6	−13.71	−18.991	−21.023	−21.078	−21.184	−21.149	−21.1646
7	−16.6792	−20.7794	−21.0574	−21.0736	−21.1488	−21.1444	−21.1393

例4-39 连续系统辨识——一个反例

- 由模型生成数据，再由数据辨识模型
  - 连续系统模型 $$G(s)=\frac{s^3+7s^2+11s+5}{s^4+7s^3+21s^2+37s+30}$$
  - 辨识离散模型再转换成连续模型
  - 数据生成与辨识

```
>> G=tf([1,7,11,5],[1,7,21,37,30]);
   t=[0:.1:8]'; u=sin(t); y=lsim(G,u,t);
   U=arx([y u],[4 4 1]);
   G1=tf(U); G1.Ts=0.1; G2=d2c(G1)
```

离散系统辨识信号的生成

- 问题：什么样信号激励系统，辨识效果最好？
- 有丰富频率信息的信号最好，如 PRBS
- 伪随机二进制序列 (pseudo-random binary sequence)
  - 频率丰富
  - 值为 ± 1，可重复构建
  - MATLAB直接生成 u=idinput(k,'prbs')
  - 正弦信号、阶跃信号不宜作为输入，频率单一

例4-40 PRBS信号波形及相关函数分析

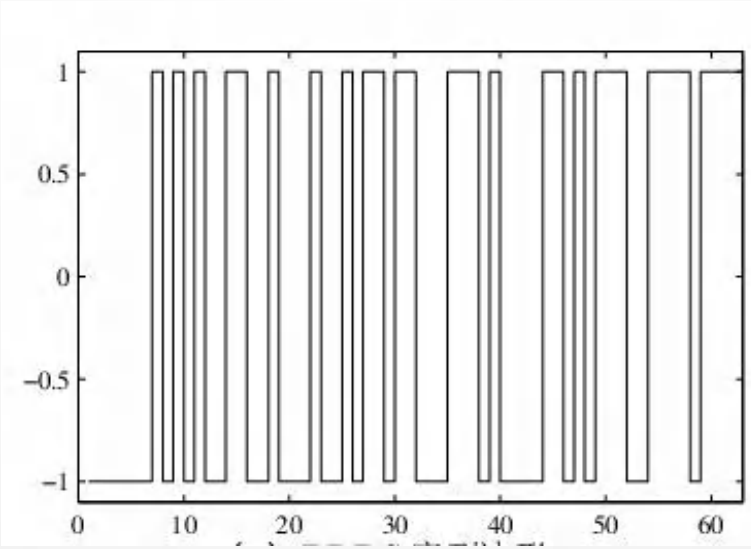
- 生成63个点的PRBS信号

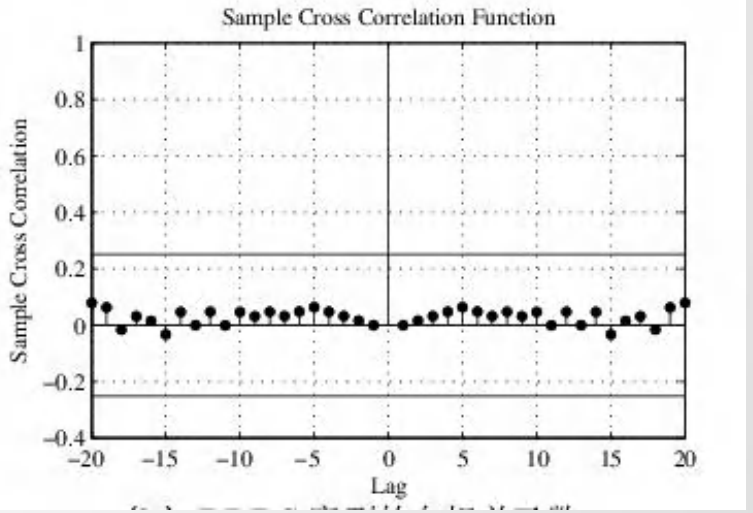
```
>> u=idinput(63,'PRBS'); t=[0:.1:6.2]'; stairs(u)
   set(gca,'XLim',[0,63],'YLim',[-1.1 1.1])
```

- 自相关函数

```
>> crosscorr(u,u)
```

- 不宜采用plot()函数绘制





例4-41 连续系统的辨识

- 仍考虑前面的例子

$$G(s)=\frac{s^3+7s^2+11s+5}{s^4+7s^3+21s^2+37s+30}$$

  - 生成样本点信息，由PRBS信号激励
  - 离散方法，再转换成连续模型

```
>> G=tf([1,7,11,5],[1,7,21,37,30]);
   t=[0:.2:6]'; u=idinput(31,'PRBS');
   y=lsim(G,u,t); U=arx([y u],[4 4 1]);
   G1=tf(U); G1.Ts=0.2; G2=d2c(G1)
```

例4-42 连续系统频域辨识方法

- 由模型生成频域响应数据，由数据辨识模型

$$G(s)=\frac{s^3+7s^2+11s+5}{s^4+7s^3+21s^2+37s+30}$$

- MATLAB函数 invfreqs
- 频域辨识方法

```
>> G=tf([1,7,11,5],[1,7,21,37,30]);
   w=logspace(-2,2); H=frd(G,w); h=H.ResponseData;
   [n,d]=invfreqs(h(:),w,4,4); Gd=tf(n,d)
```

多变量离散系统的辨识

- 多变量系统的差分方程模型

$$\boldsymbol{A}(z^{-1})\boldsymbol{y}(t) = \boldsymbol{B}(z^{-1})\boldsymbol{u}(t-\boldsymbol{d}) + \boldsymbol{\varepsilon}(t)$$

- 离散传递函数矩阵模型

$$\begin{cases} \boldsymbol{A}(z^{-1}) = \boldsymbol{I}_{p\times q} + \boldsymbol{A}_1 z^{-1} + \cdots + \boldsymbol{A}_{n_\mathrm{a}} z^{-n_a} \\ \boldsymbol{B}(z^{-1}) = \boldsymbol{I}_{p\times q} + \boldsymbol{B}_1 z^{-1} + \cdots + \boldsymbol{B}_{n_b} z^{-n_\mathrm{b}} \end{cases}$$

例4-43 多变量系统辨识

- 原始模型

$$G(z)=\begin{bmatrix} \dfrac{0.5234z-0.1235}{z^2+0.8864z+0.4352} & \dfrac{3z+0.69}{z^2+1.084z+0.3974} \\ \dfrac{1.2z-0.54}{z^2+1.764z+0.9804} & \dfrac{3.4z-1.469}{z^2+0.24z+0.2848} \end{bmatrix}$$

- 模型输入

```
>> u1=idinput(31,'PRBS'); t=0:.1:3; u2=u1(end:-1:1);
   g11=tf([0.5234, -0.1235],[1, 0.8864, 0.4352],'Ts',0.1);
   g12=tf([3, 0.69],[1, 1.084, 0.3974],'Ts',0.1);
   g21=tf([1.2, -0.54],[1, 1.764, 0.9804],'Ts',0.1);
   g22=tf([3.4, 1.469],[1, 0.24, 0.2848],'Ts',0.1);
   G=[g11, g12; g21, g22];
```

- 模型辨识

```
>> y=lsim(G,[u1 u2],t); na=4*ones(2); nb=na; nc=ones(2);
   U=iddata(y,[u1,u2],0.1); T=arx(U,[na nb nc])
```

```
>> H=tf(T); g11=H(1,1)
```

模型降阶方法小结

- 给出了模型降阶的思想
  - 介绍了Pade降阶思想与函数 pademod
  - 介绍了Routh降阶函数 routhmod
- 介绍了鲁棒控制工具箱中的降阶函数
  - 均衡实现的降阶函数 modred
  - Schur降阶算法 schmr
  - 最优Hankel范数降阶算法 ohklmr
- 需要探讨更好的算法，包括可处理延迟的方法

次最优模型降阶小结

- 引入客观最优性能指标，将降阶问题转换成数值最优化问题
- 利用MATLAB提供的最优化工具直接求解
- 由于延迟项存在，引入Pade近似，所以最优降阶更精确地定义为次最优
- MATLAB函数 opt_app

系统辨识小结

- 为什么要系统辨识？
- 由实验数据如何辨识系统的数学模型
  - 最小二乘求解，矩阵左除
  - 直接辨识函数 arx, tf
- 系统辨识界面 idnt

系统辨识进阶小结

- 如果想合理选择阶次，则需要用AIC准则衡量
  - AIC准则的定义域近似 aic
  - 对采用的使用的函数 arx, tf
- 激励信号在辨识中至关重要，应选择频率信息丰富的信号，如PRBS，idinput，iddata函数
- 连续系统的频域响应辨识 invfreqs
- 函数arx可以用于多变量系统的辨识

Q & A

感谢您的聆听和反馈