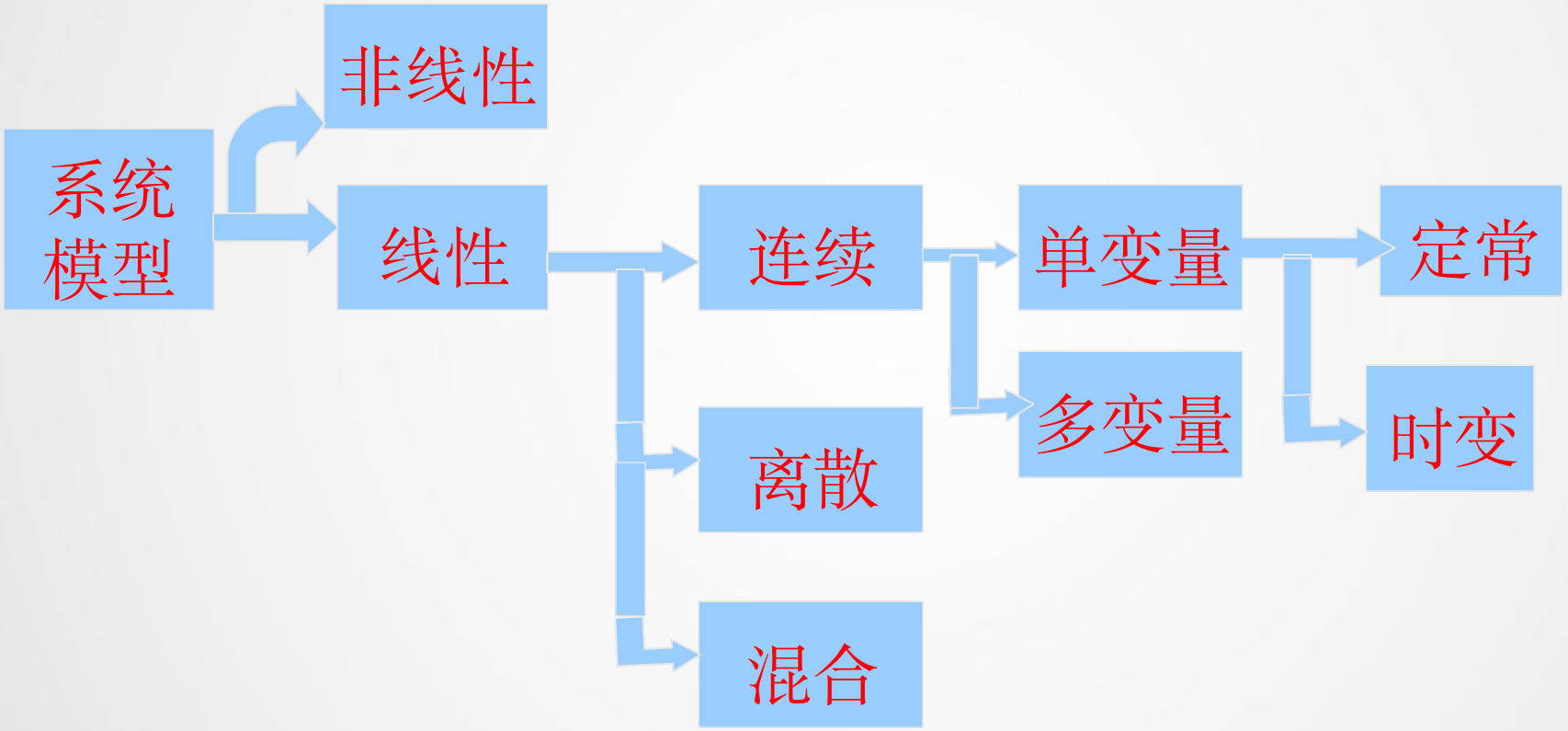
国家精品课程/ 国家精品资源共享课程/ 国家级精品教材
国家级十一(二)五规划教材/ 教育部自动化专业教学指导委员会牵头规划系列教材

控制系统计算机辅助设计

第6章: 非线性系统的建模与仿真

主讲：修贤超

系统数学模型的分类



非线性系统的建模与仿真

- 6.1 Simulink建模的基础知识
- 6.2 Simulink建模与仿真
- 6.3 控制系统的Simulink建模与仿真实例
- 6.4 非线性系统分析与仿真
- 6.5 子系统与模块封装技术
- 6.6 S-函数编写及其应用

Simulink建模基础

- 在Simulink出现之前，MATLAB仿真功能弱
  - 借助ACSL等仿真语言进行仿真
  - 接口不好，需要数据文件交互
  - 描述系统采用语言描述，缺乏框图支持，易错
- Simulink的出现与进展
  - 1990年MathWorks推出了SimuLAB，取代ACSL
  - 1992年更名Simulink
  - 2007年，支持Simscape、多领域物理建模
  - 2012年，2012b，全新的建模界面，使用方便，新版本功能更强

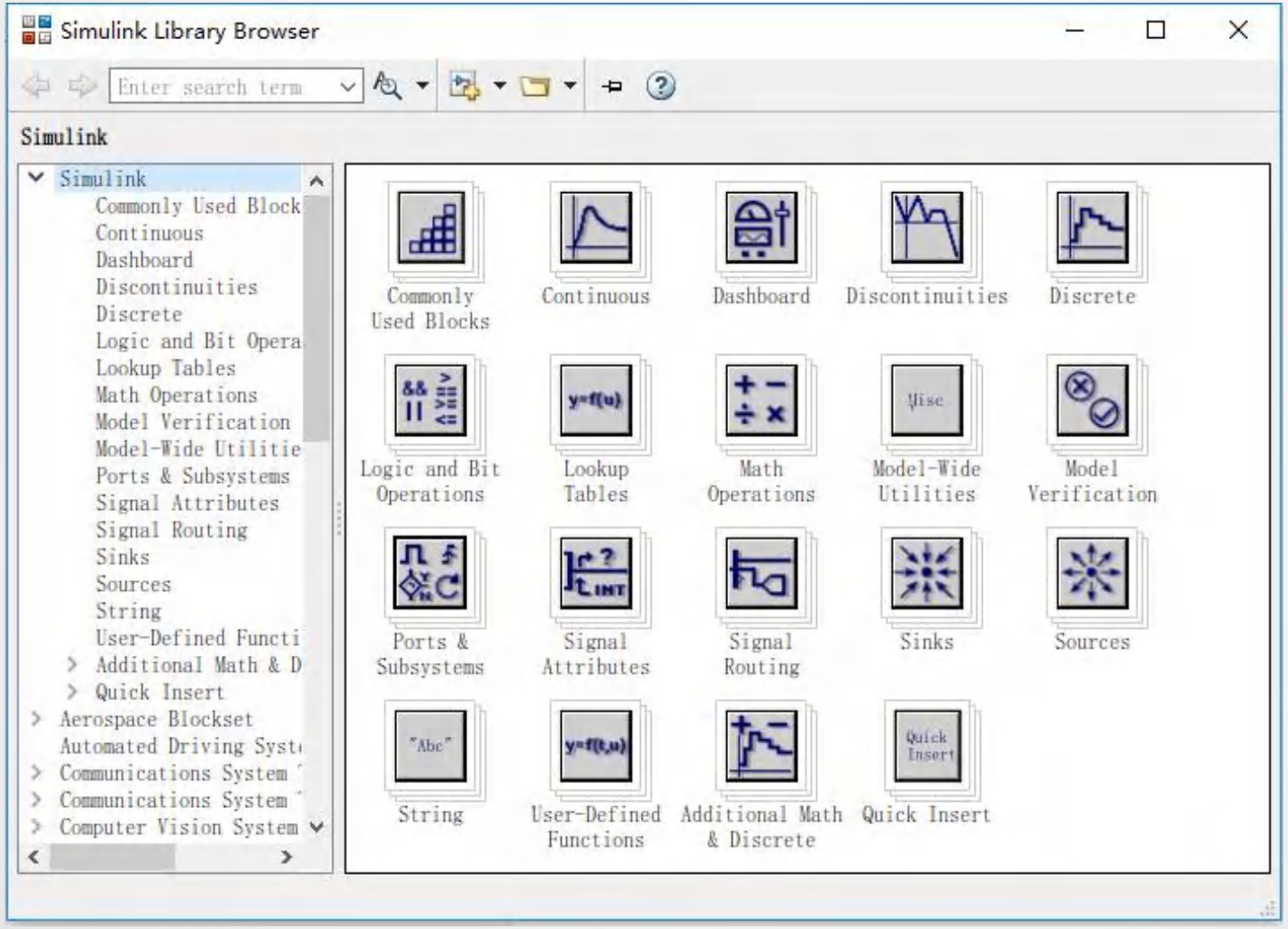
Simulink的常用模块简介

- 启动Simulink
  - 命令行式

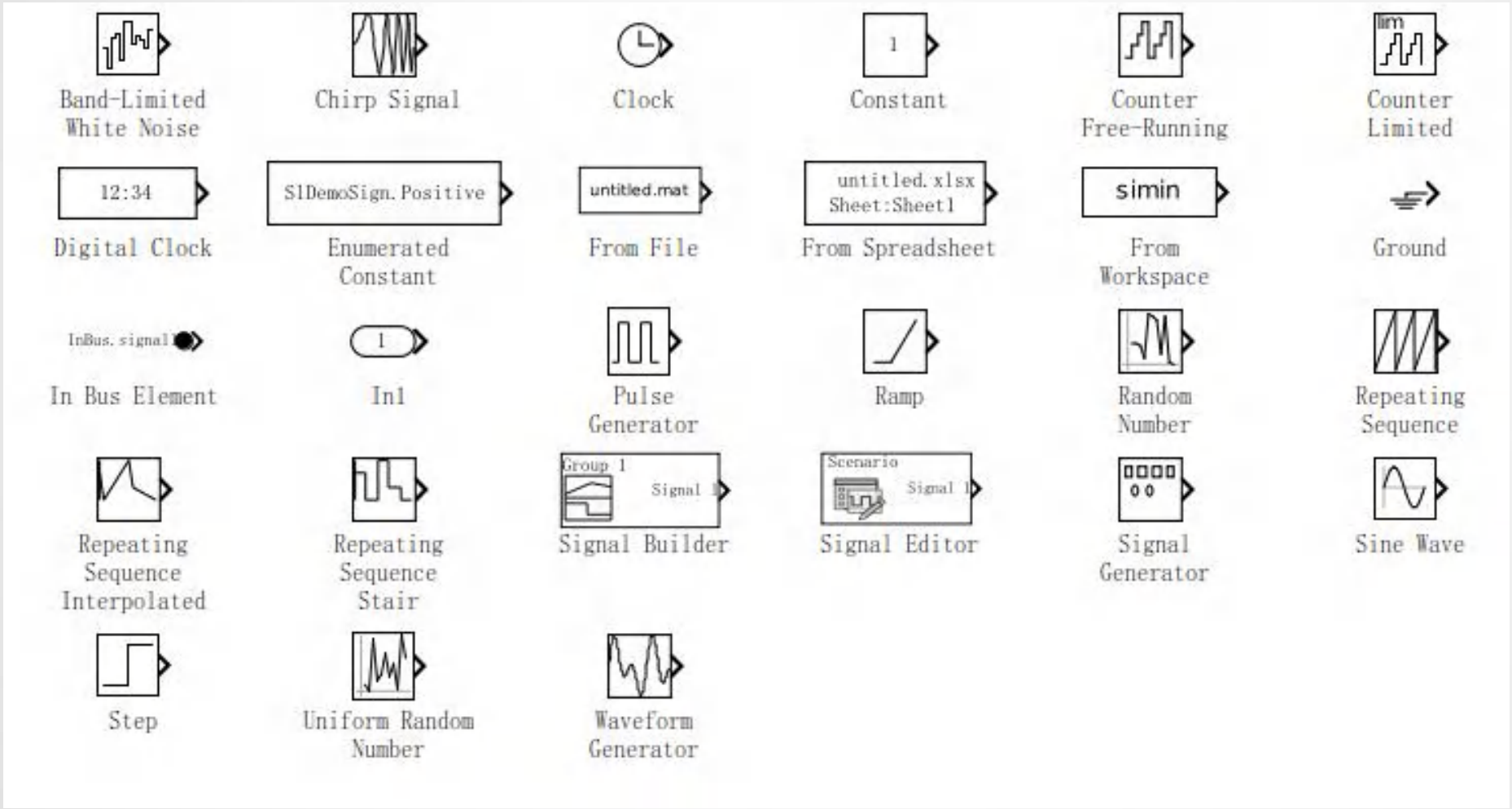
```
>> open_system('simulink')
```

  - 双击图标——不同版本的Simulink图标略有不同
- 相关模块简介
  - 输入模块组、输出模块组
  - 其他相关模块：连续、离散、非线性等
  - 专业模块：Simscape、SimMechanics等

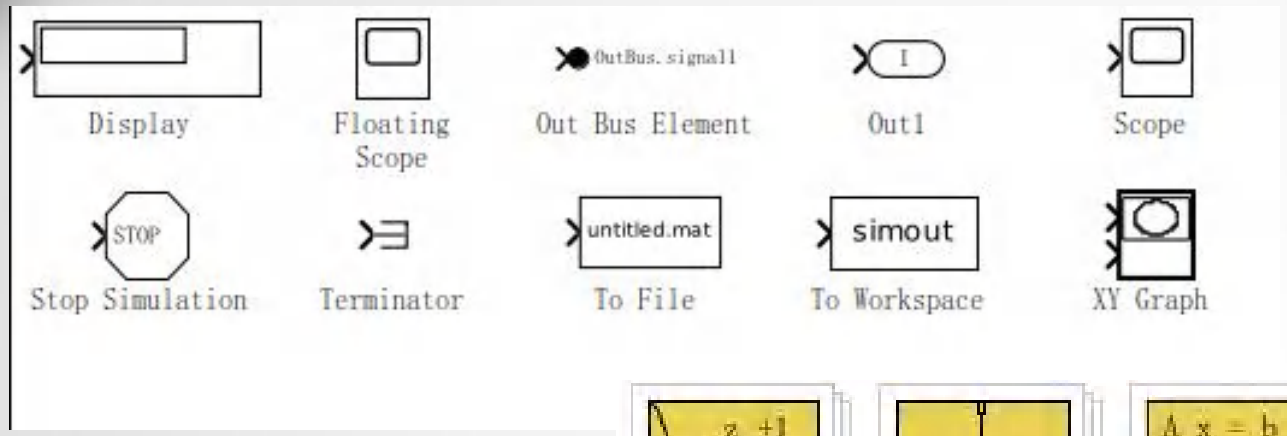
现有的模块库

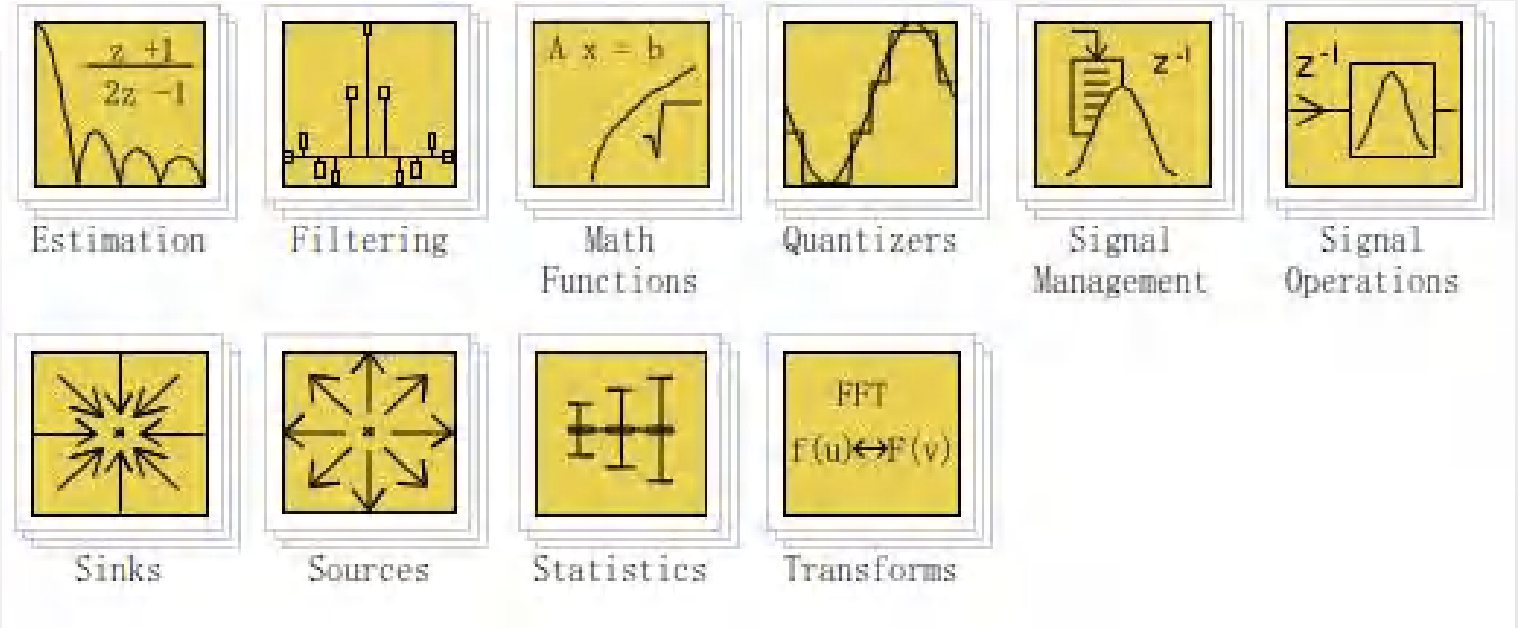

Simulink Library Browser
Enter search term
Simulink
Simulink
Commonly Used Block
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Opera
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilitie
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functi
Additional Math & D
Quick Insert
Aerospace Blockset
Automated Driving Syst
Communications System
Communications System
Computer Vision System
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model-Wide Utilities
Model Verification
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
String
User-Defined Functions
Additional Math & Discrete
Quick Insert

输入模块组 Sources

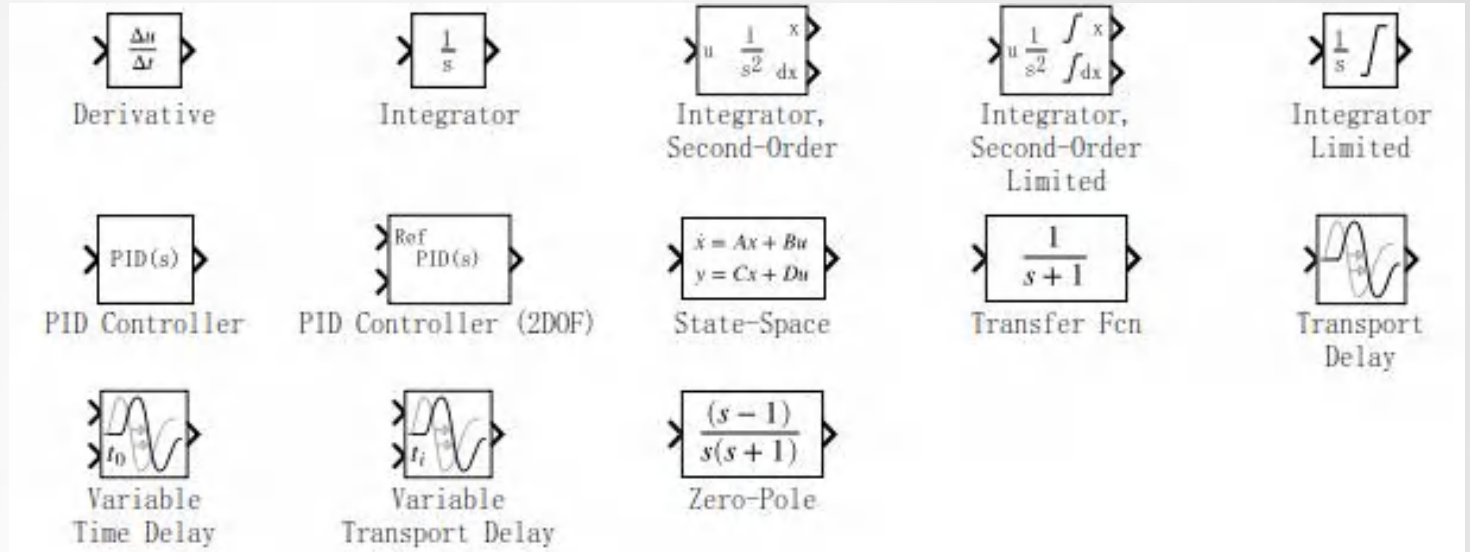


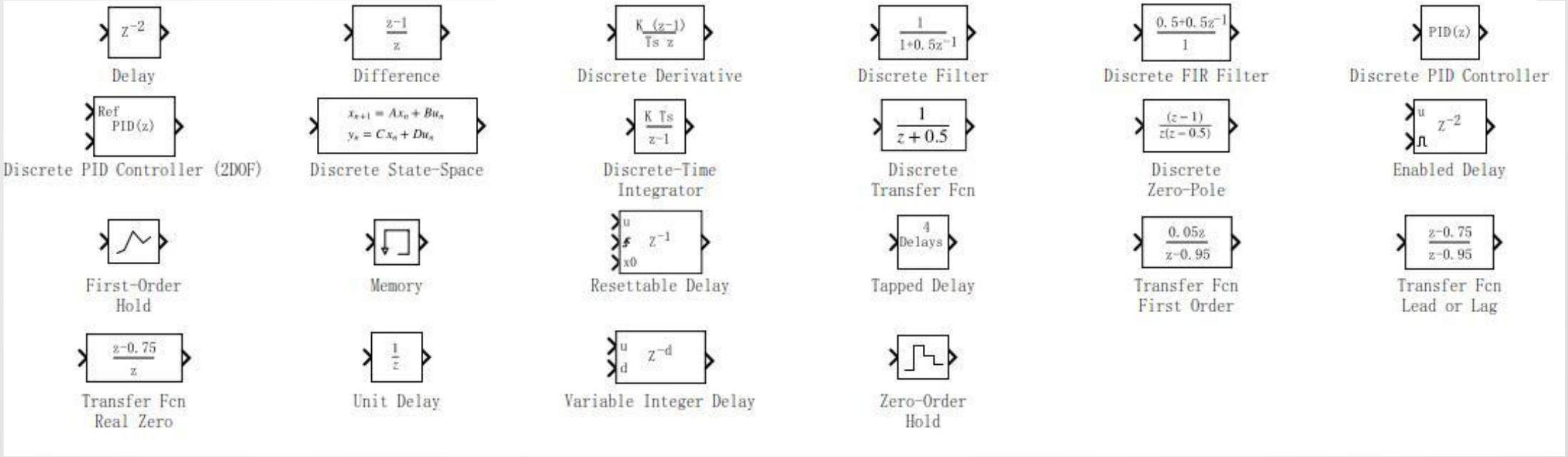
输出模块组


Display
Floating Scope
OutBus.signal1
Out Bus Element
1
Out1
Scope
STOP
Stop Simulation
Terminator
untitled.mat
To File
simout
To Workspace
XY Graph

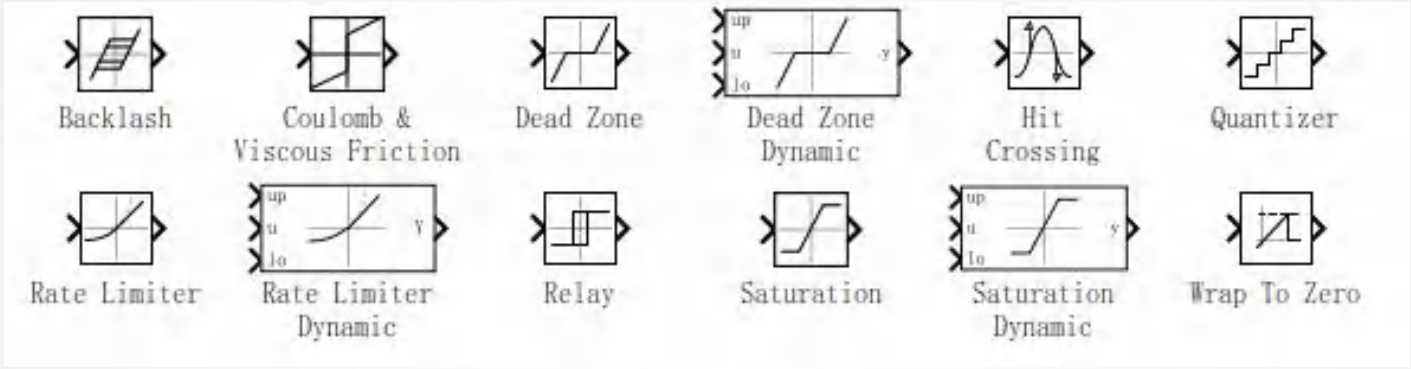

Estimation
Filtering
A x = b
Math Functions
Quantizers
Signal Management
Signal Operations
Sinks
Sources
Statistics
FFT
Transforms

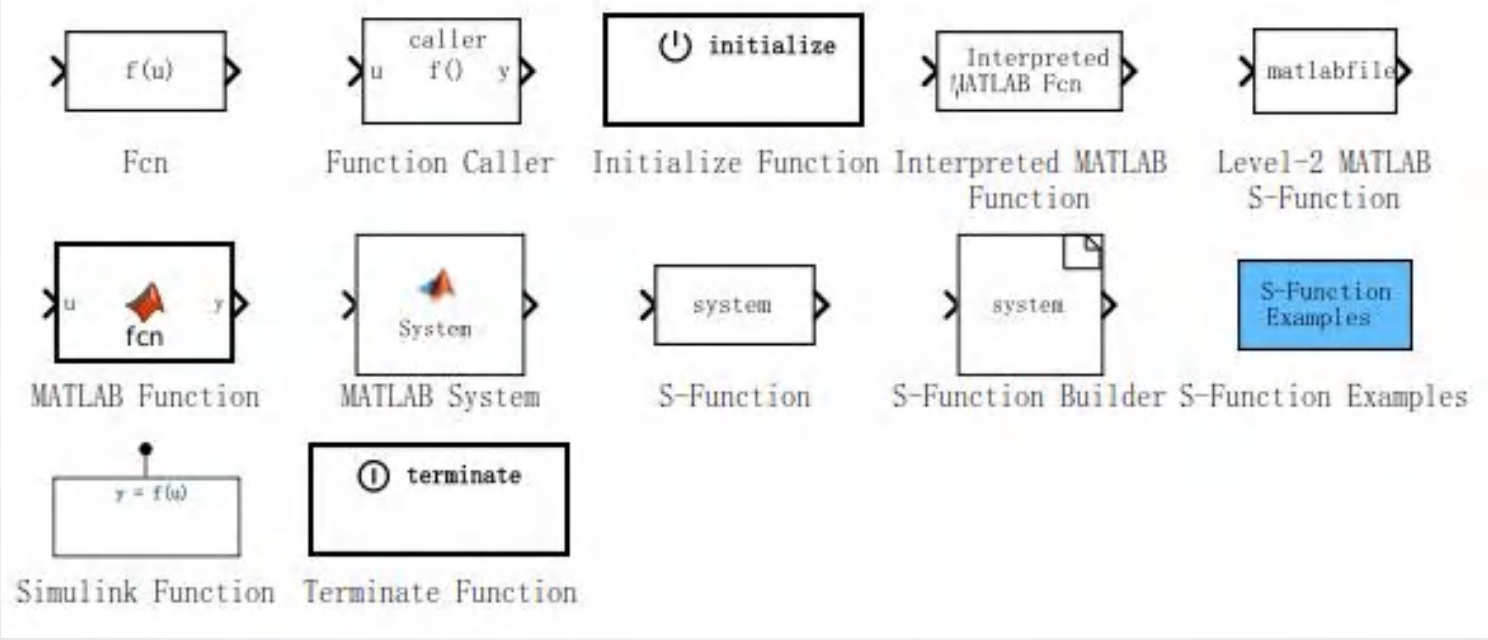
线性模块组


Derivative
Integrator
Integrator, Second-Order
Integrator, Second-Order Limited
Integrator Limited
PID Controller
PID Controller (2DOF)
State-Space
Transfer Fcn
Transport Delay
Variable Time Delay
Variable Transport Delay
Zero-Pole


Delay
Difference
Discrete Derivative
Discrete Filter
Discrete FIR Filter
Discrete PID Controller
Discrete PID Controller (2DOF)
Discrete State-Space
Discrete-Time Integrator
Discrete Transfer Fcn
Discrete Zero-Pole
Enabled Delay
First-Order Hold
Memory
Resettable Delay
Tapped Delay
Transfer Fcn First Order
Transfer Fcn Lead or Lag
Transfer Fcn Real Zero
Unit Delay
Variable Integer Delay
Zero-Order Hold

非线性模块与自定义模块组





Toolboxes and Blocksets

- Aerospace Blockset
- Automated Driving System Toolbox
- Communications System Toolbox
- Communications System Toolbox HDL Support
- Computer Vision System Toolbox
- Control System Toolbox
- Data Acquisition Toolbox
- DSP System Toolbox
- DSP System Toolbox HDL Support
- Embedded Coder
- Fuzzy Logic Toolbox
- HDL Coder
- HDL Verifier
- Image Acquisition Toolbox
- Instrument Control Toolbox
- MIMO FOTF Toolbox
- Model Predictive Control Toolbox
- Neural Network Toolbox
- OPC Toolbox
- Phased Array System Toolbox
- Powertrain Blockset
- Report Generator
- RF Blockset
- Robotics System Toolbox
- Robust Control Toolbox
- SimEvents
- Simscape
- Simulink 3D Animation
- Simulink Coder
- Simulink Control Design
- Simulink Design Optimization
- Simulink Design Verifier
- Simulink Desktop Real-Time
- Simulink Extras
- Simulink Real-Time
- Simulink Requirements
- Simulink Test
- Stateflow
- System Identification Toolbox
- Vehicle Network Toolbox
- Vision HDL Toolbox
- Recently Used

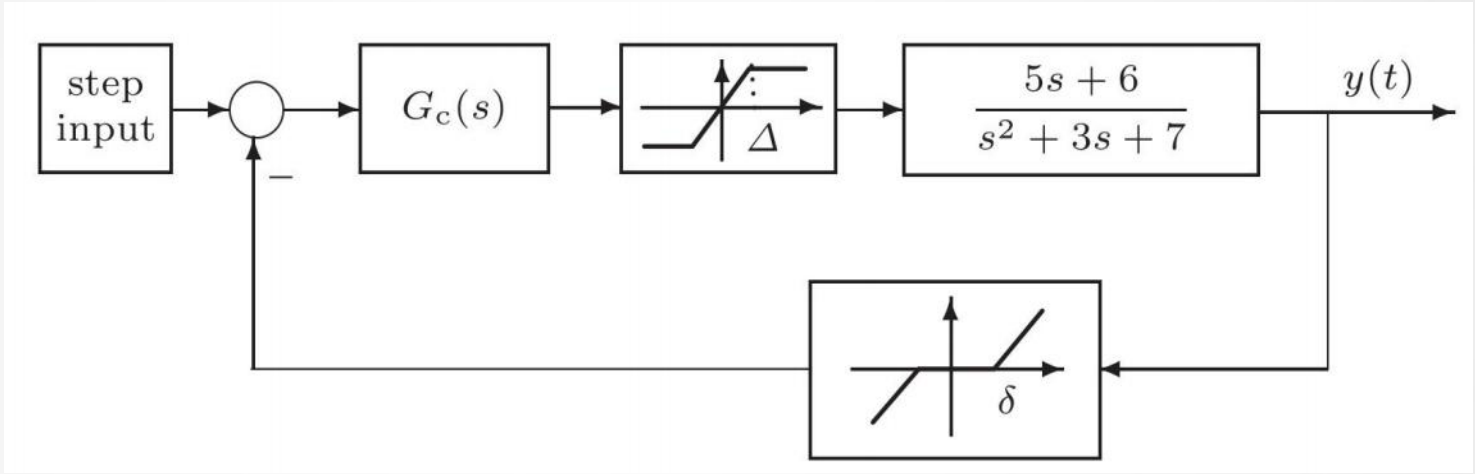
非线性系统的建模与仿真

- 6.1 Simulink建模的基础知识
- 6.2 Simulink建模与仿真
- 6.3 控制系统的Simulink建模与仿真实例
- 6.4 非线性系统分析与仿真
- 6.5 子系统与模块封装技术
- 6.6 S-函数编写及其应用

例6-1 非线性系统建模与仿真

- 非线性系统建模与仿真

$$G_c(s) = 3 + 2/s$$
$$\Delta = 2, \delta = 0.1$$



- 所需模块：
  - 传递函数、饱和、死区、加法器、输入、输出
- 模块修饰、模块连线、仿真参数、启动仿真
- 修改参数

6.2 仿真举例——微分方程建模

- 微分方程的建模：Rössler方程

$$\begin{cases} x'(t) = -y(t) - z(t) \\ y'(t) = x(t) + ay(t) \\ z'(t) = b + [x(t) - c]z(t) \end{cases}$$

$$a = b = 0.2,\ c = 5.7$$

$$x(0) = y(0) = z(0) = 0$$

- 建模方法
  - 底层建模——关键信号定义：积分器使用
  - 利用MATLAB函数模块 Fcn

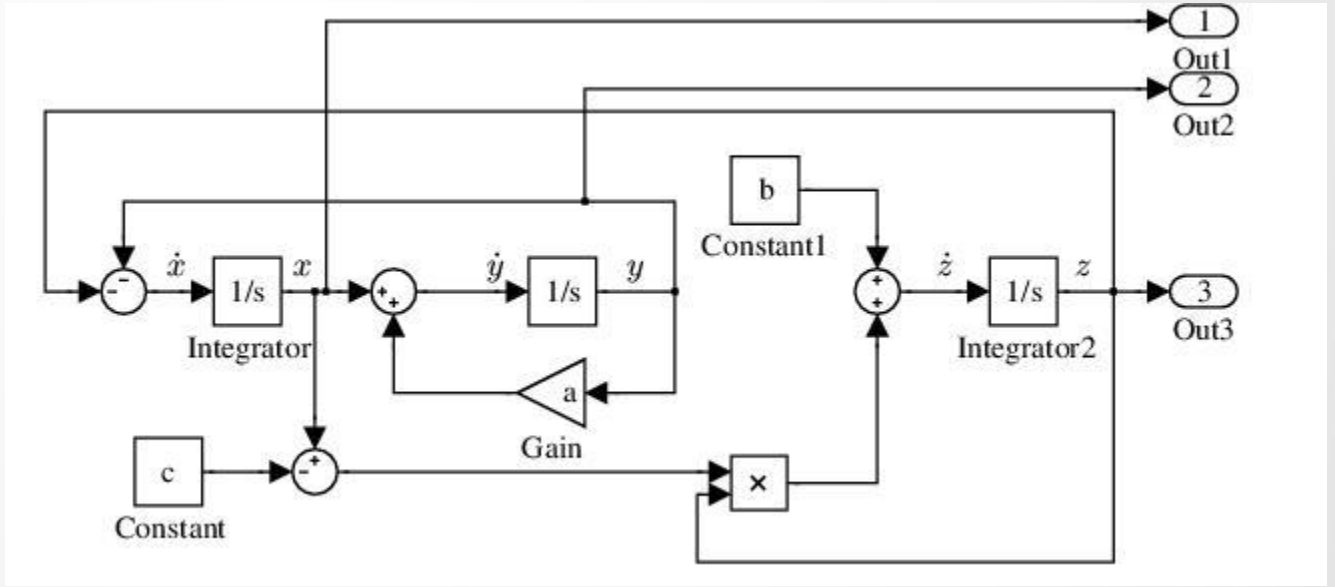
微分方程最底层的建模方法

- 底层模块建模
  - 先构建关键信号

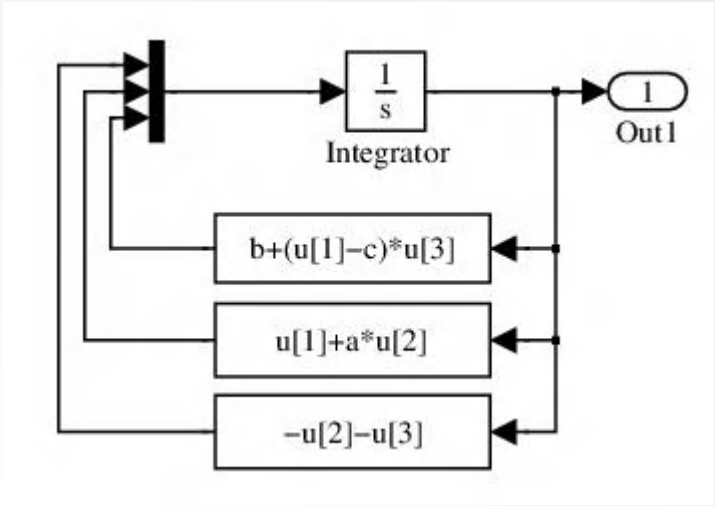
$$x, x', y, y', z, z'$$

  - 由关键信号连线，画出微分方程
  - 微分方程的求解：仿真方法

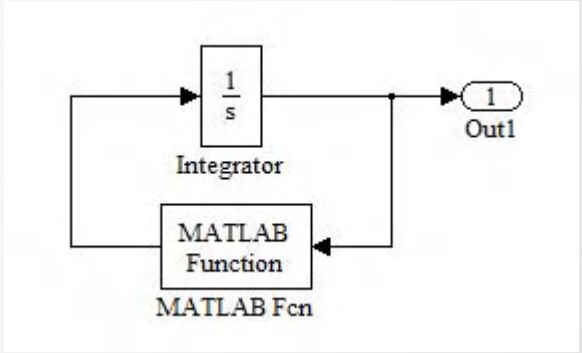
$$\begin{cases} x'(t) = -y(t) - z(t) \\ y'(t) = x(t) + ay(t) \\ z'(t) = b + [x(t) - c]z(t) \end{cases}$$



简单建模方法



- 简化模型
  - 为什么需要简化？
  - 利用Fcn模块——局限性
  - 模型修饰——向量信号、数据类型、路数等
- 更简单的描述：
  - MATLAB Fcn模块——模型：c6mross1.mdl




```
function y=c6mross(x)
a=0.2; b=0.2; c=5.7;
y=[-x(2)-x(3); x(1)+a*x(2); b+[x(1)-c]*x(3)];
```

微分方程的求解

- MATLAB求解

```
>> a=0.2; b=0.2; c=5.7; x0=[0; 0; 0];
   f=@(t,x)[-x(2)-x(3); x(1)+a*x(2); b+[x(1)-c]*x(3)];
   [t,x]=ode45(f,[0,10],x0); plot(t,x)
```

- Simulink建模优势：模块化建模
- 两种方法
  - 界面方法：输出端子——返回 tout, yout
  - 语句方法

$$[t,x,y]=\texttt{sim(model,}[t_0,t_\mathrm{f}]\texttt{,options)}$$

非线性系统的建模与仿真

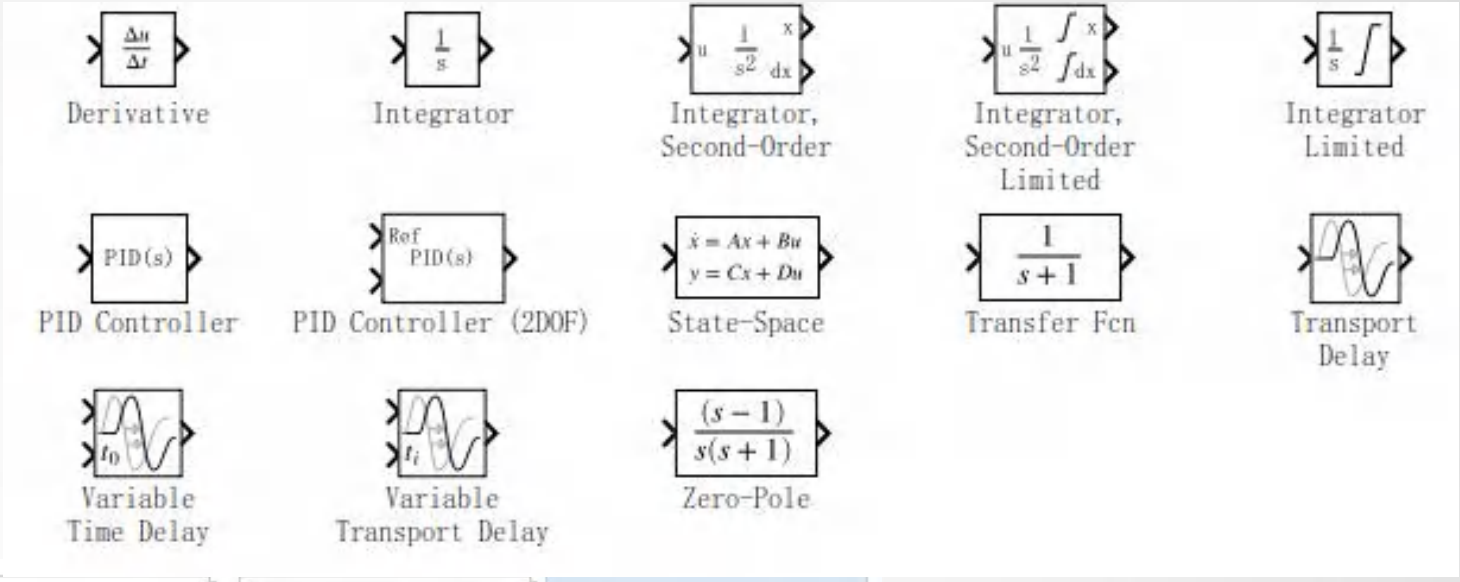
- 6.1 Simulink建模的基础知识
- 6.2 Simulink建模与仿真
- 6.3 控制系统的Simulink建模与仿真实例
- 6.4 非线性系统分析与仿真
- 6.5 子系统与模块封装技术
- 6.6 S-函数编写及其应用

控制系统的建模与仿真举例

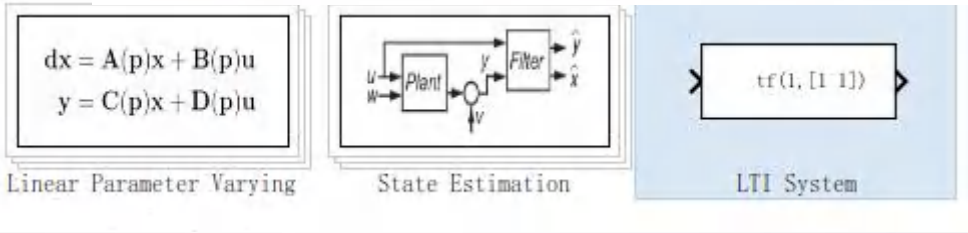
- Simulink 提供的工具可以把控制系统画出来
- 这里将介绍各种控制系统的建模仿真方法
  - 多变量控制系统
  - 计算机控制系统
  - 时变系统
  - 多采样速率系统
  - 延迟系统与变延迟系统
  - 切换系统
  - 随机输入系统

线性模块库

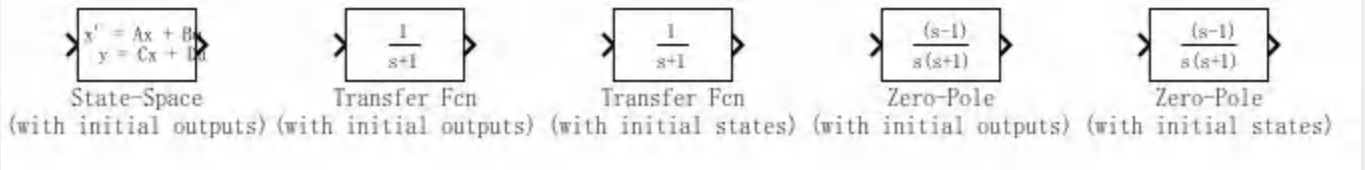
- 常规模块库



- 控制系统工具箱



- 非零初值系统的建模
  - Simulink Extras -> Additional Linear



例6-3多变量时间延迟系统的仿真

- 多变量系统

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

- MATLAB仿真方法——局限性
- Simulink仿真

MATLAB求解方法

- 补偿系统的阶跃响应

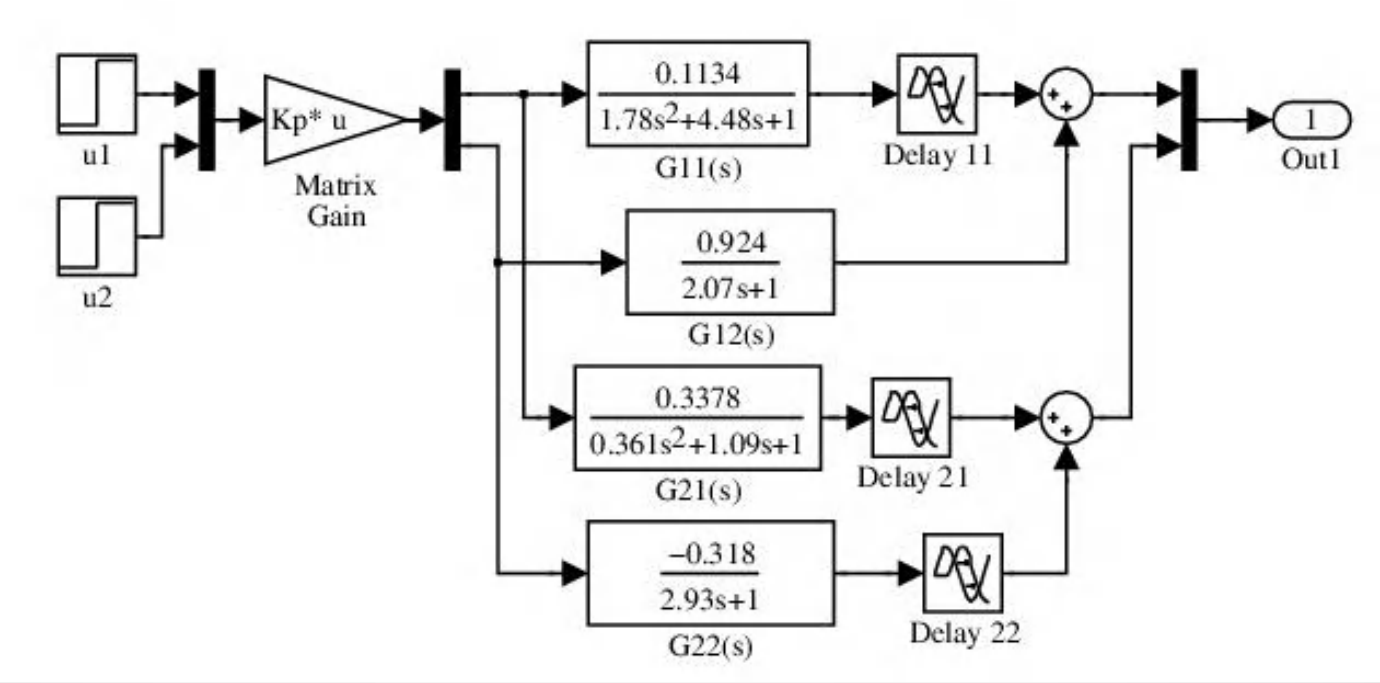
```
>> g11=tf(0.1134,[1.78 4.48 1],'ioDelay',0.72);
   g21=tf(0.3378,[0.361 1.09 1],'ioDelay',0.3);
   g12=tf(0.924,[2.07 1]);
   g22=tf(-0.318,[2.93 1],'ioDelay',1.29);
   G=[g11, g12; g21, g22]; G=ss(G);
   Kp=[0.1134,0.924; 0.3378,-0.318]; step(G*Kp,15)
```

- 局限性——如果有非线性环节，无能为力

Simulink仿真方法

- 早期版本建模方法
  - 模型文件： c6mmimo.mdl

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



使用LTI模块的建模仿真方法

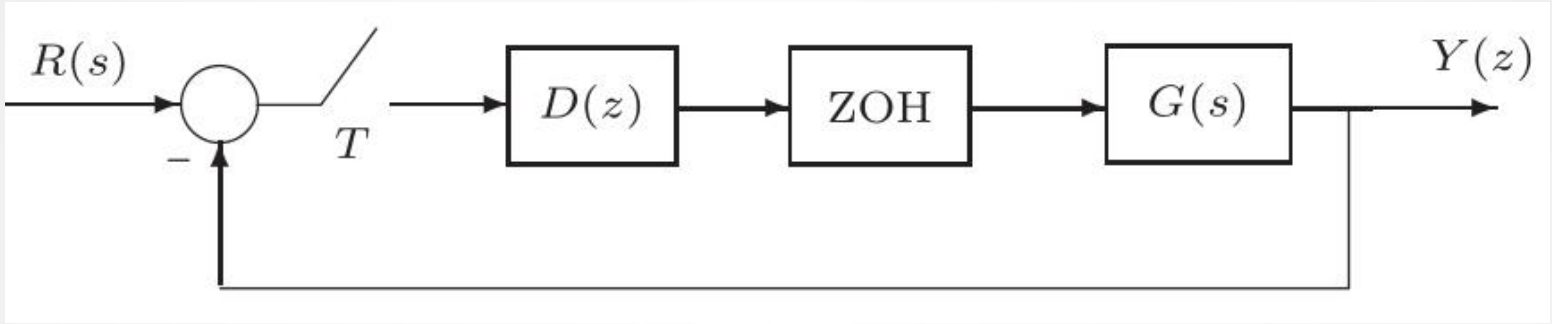
- 简化建模—— c6mmimo0.mdl
- 不同方法比较

```
>> u1=1; u2=0; [t1,a,y1]=sim('c6mmimo',15);
   u1=0; u2=1; [t2,a,y2]=sim('c6mmimo',15);
   [y,t]=step(G*Kp,15);
>> subplot(221), plot(t,y(:,1,1),':',t1,y1(:,1))
   subplot(222), plot(t,y(:,1,2),':',t2,y2(:,1))
   subplot(223), plot(t,y(:,2,1),':',t1,y1(:,2))
   subplot(224), plot(t,y(:,2,2),':',t2,y2(:,2))
```

- 对复杂线性系统模型，尽量采用新版的LTI模块

例6-5 计算机控制系统仿真

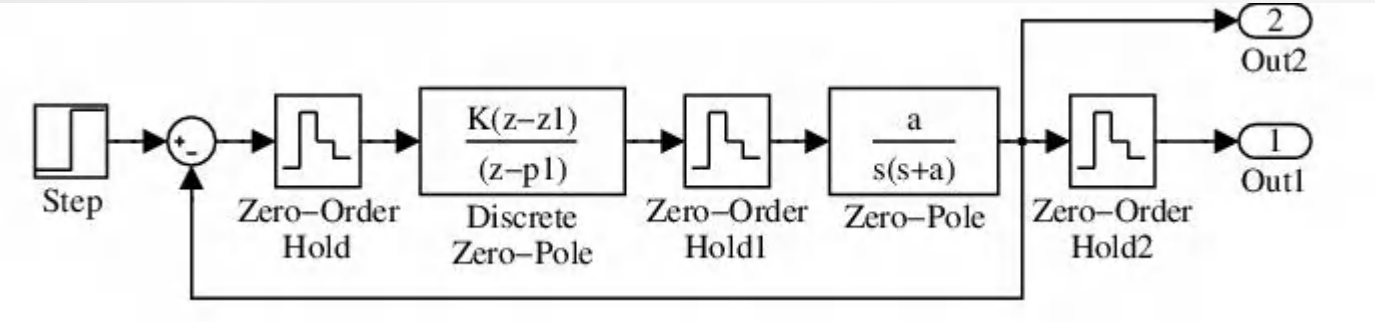
- 计算机控制系统



$$G(s)=\frac{a}{s(s+1)},\quad D(z)=\frac{1-\mathrm{e}^{-T}}{1-\mathrm{e}^{-0.1T}}\frac{z-\mathrm{e}^{-0.1T}}{z-\mathrm{e}^{-T}}$$

- 所需模块
  - 连续传递函数、离散传递函数、零阶保持器
  - 模型名：c6mcompc.mdl

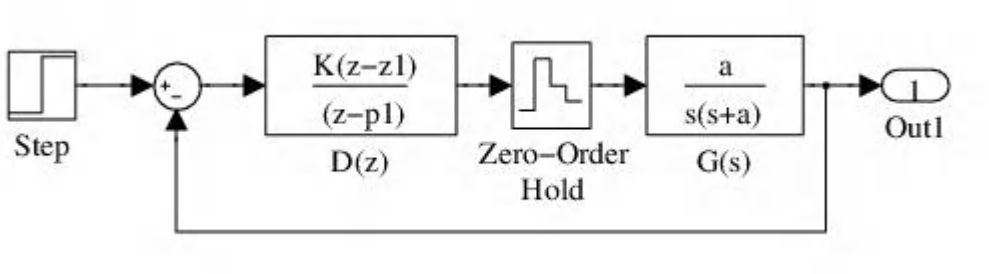
Simulink模型



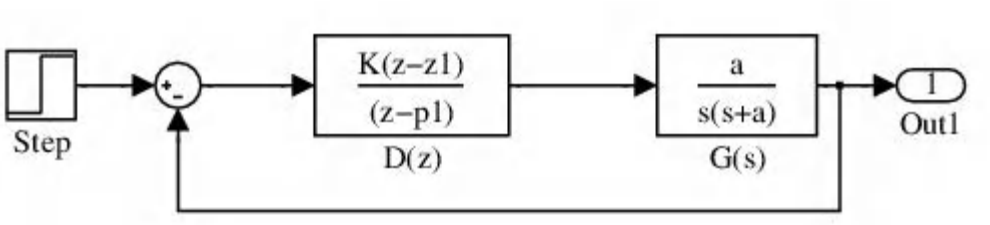

```
>> T=0.2; a=0.1; z1=exp(-0.1*T); p1=exp(-T);
   K=(1-p1)/(1-z1); [t,x,y]=sim('c6mcompc',20);
   plot(t,y(:,2)); hold on; stairs(t,y(:,1))
>> T=1; z1=exp(-0.1*T); p1=exp(-T); K=(1-p1)/(1-z1);
   [t,x,y]=sim('c6mcompc',20); plot(t,y(:,2));
   hold on; stairs(t,y(:,1))
>> T=0.2; z1=exp(-0.1*T); p1=exp(-T); K=(1-p1)/(1-z1);
   Dz=zpk(z1,p1,K,'Ts',T); G=zpk([],[0;-a],a);
   Gz=c2d(G,T); GG=zpk(feedback(Gz*Dz,1)), step(GG)
```

Simulink模型的化简

- Simulink模型进一步化简
  - c6mcomc1



  - c6mcomc2



- 参数的预置
  - File -> Model properties

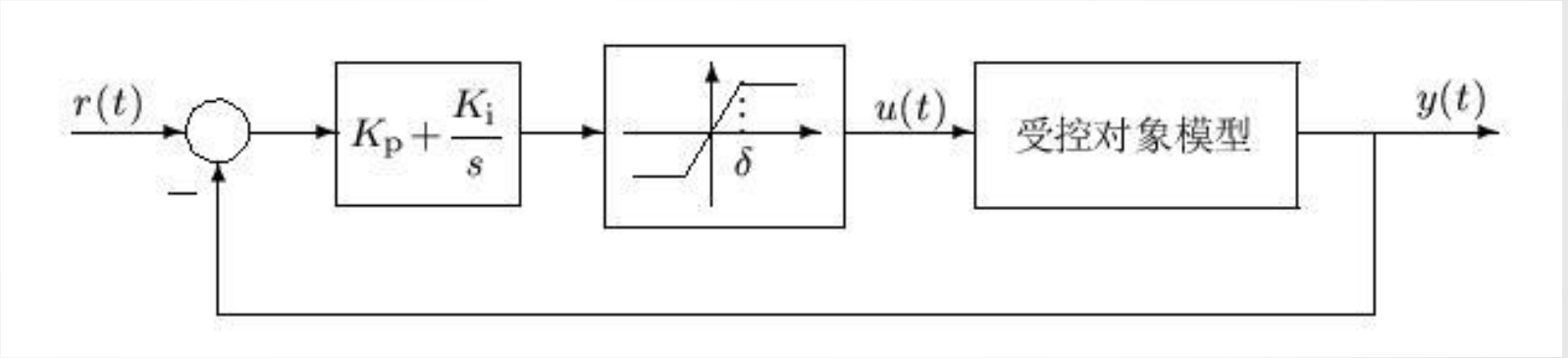
```
T=0.2; z1=exp(-0.1*T); p1=exp(-T); K=(1-p1)/(1-z1);
```

例6-6 时变系统仿真

- 受控对象为微分方程

$$\ddot{y}(t)+\mathrm{e}^{-0.2t}\dot{y}(t)+\mathrm{e}^{-5t}\sin(2t+6)y(t)=u(t)$$

- PI 控制器

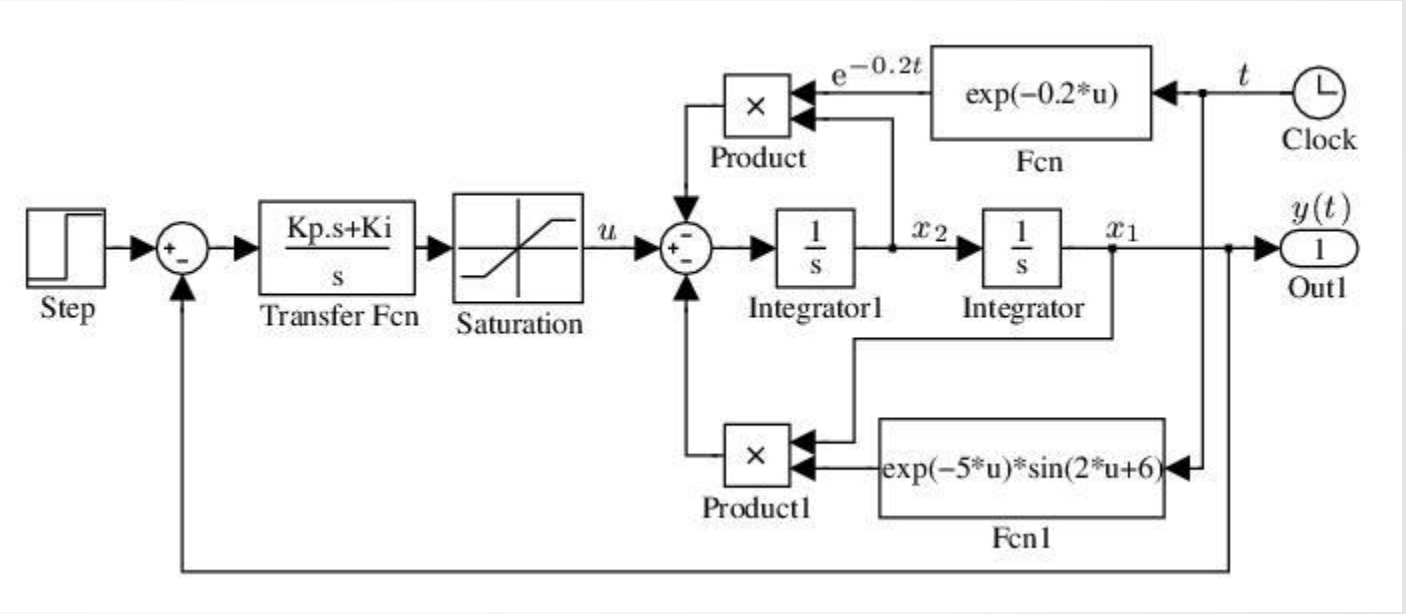


- 改写受控对象模型

$$\ddot{y}(t)=-\mathrm{e}^{-0.2t}\dot{y}(t)-\mathrm{e}^{-5t}\sin(2t+6)y(t)+u(t)$$

Simulink建模

- 时变受控对象 $\ddot{y}(t) = -\mathrm{e}^{-0.2t}\dot{y}(t) - \mathrm{e}^{-5t}\sin(2t+6)y(t) + u(t)$
- 模型：c6mtimv.mdl




```
>> opt=simset('RelTol',1e-8); Kp=200; Ki=10;
   [t,x,y]=sim('c6mtimv',10,opt); plot(t,y)
```

例6-9 切换系统的仿真

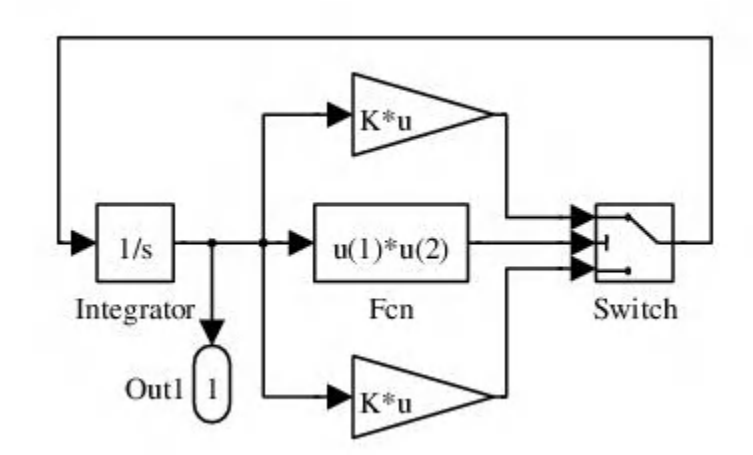
- 状态方程（自治系统） $\dot{\boldsymbol{x}} = \boldsymbol{A}_i \boldsymbol{x},\ x_1(0) = x_2(0) = 5$

$$\boldsymbol{A}_1 = \begin{bmatrix} 0.1 & -1 \\ 2 & 0.1 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 0.1 & -2 \\ 1 & 0.1 \end{bmatrix}$$

- 不稳定矩阵，切换条件
  - II、IV象限： $x_1 x_2 < 0 \rightarrow \boldsymbol{A}_1$
  - I、III象限： $x_1 x_2 \geqslant 0 \rightarrow \boldsymbol{A}_2$
  - Simulink模型：c6mswi1.mdl——矩阵自动调入

Simulink建模方法

- 过零检测



- 相平面图

```
>> plot(tout,yout),
```

```
>> plot(yout(:,1),yout(:,2))
```

- 切换系统稳定

Simulink小结

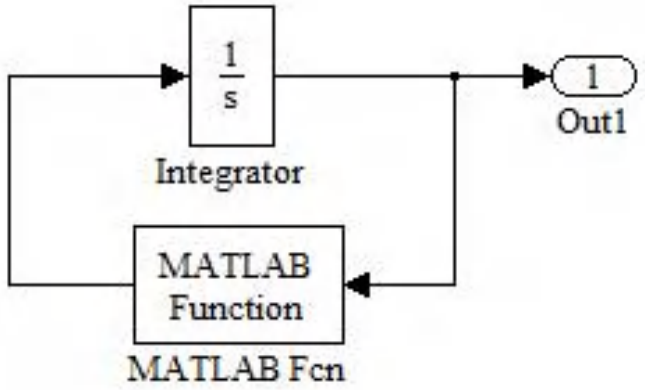
- 简单介绍了Simulink发展背景
- 介绍了一些常用的模块
  - 输入模块
  - 输出模块
  - 线性连续模块、线性离散模块
  - 非线性模块

Simulink例子小结

- Simulink建模的步骤
  - 打开空白模型窗口
  - 找出所需模块并复制到窗口
  - 修改参数
  - 连线，构造整个模型，使用技巧
- 模型的仿真
  - 控制参数修改
  - 仿真结果的检验——不同的算法、相对误差限

微分方程Simulink建模小结

- 用Simulink画出微分方程
  - 关键信号构造——利用积分器搭建
  - 向量化的积分器
  - 自定义函数描述微分方程
    - 统一的描述框架
- 微分方程求解——sim函数



控制系统建模仿真小节

- 介绍了不同系统结构的Simulink建模方法
- 对线性系统模块建议采用LTI模块
- 尤其指出了随机输入信号传统仿真方法的错误

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6章: 非线性系统的建模与仿真

主讲：修贤超

非线性系统的建模与仿真

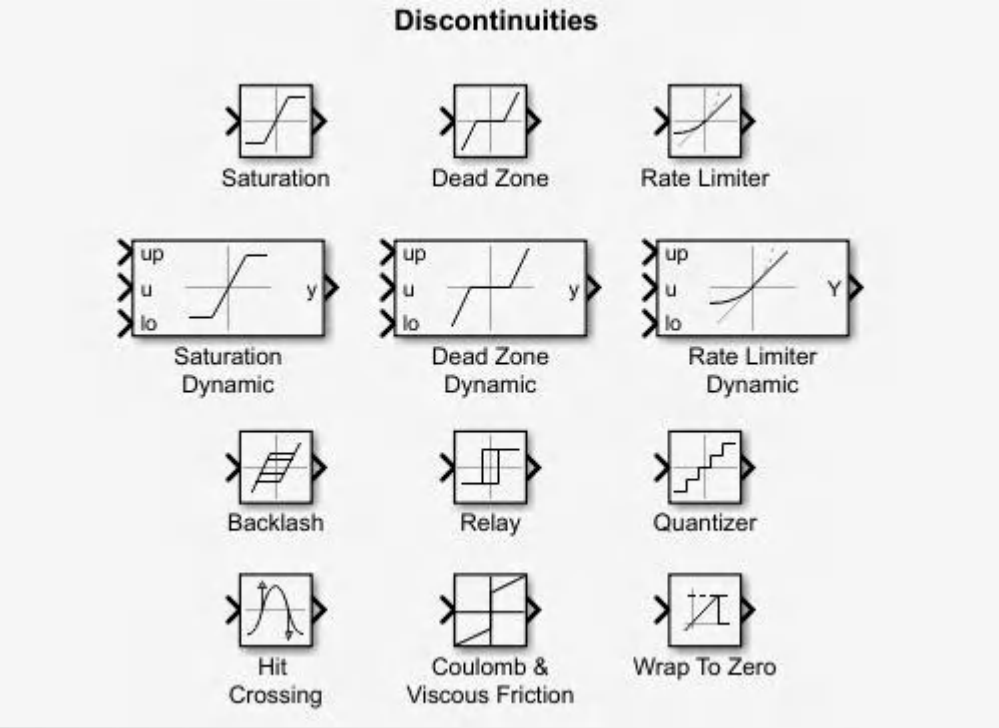
- 6.1 Simulink建模的基础知识
- 6.2 Simulink建模与仿真
- 6.3 控制系统的Simulink建模与仿真实例
- 6.4 非线性系统分析与仿真
- 6.5 子系统与模块封装技术
- 6.6 S-函数编写及其应用

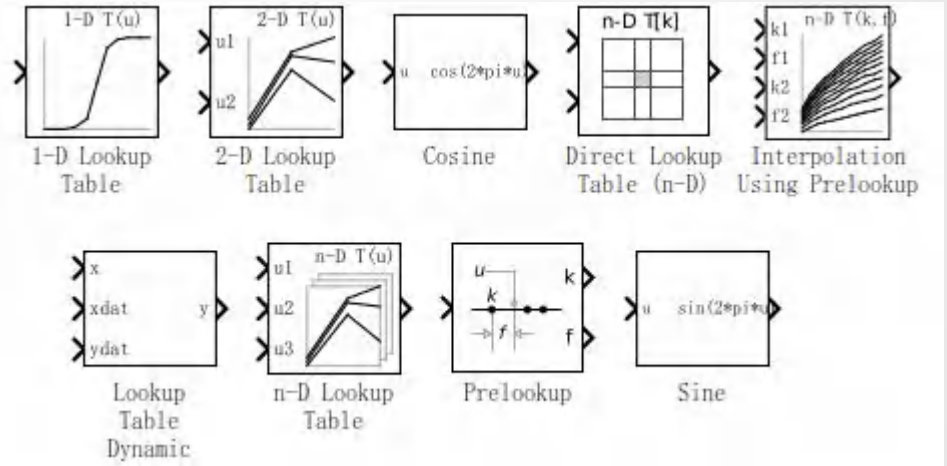
一般非线性系统仿真

- MATLAB控制系统工具箱直接用于线性时不变系统的分析与仿真
- 时变系统需要Simulink建模
- 现有非线性模块集的不足
- 一般静态非线性环节的建模
  - 单值非线性与多值非线性
  - 记忆非线性环节

现有非线性模块库

- Discontinuity模块组——模块有限难以描述任意非线性



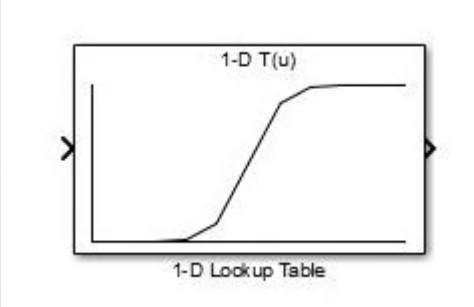


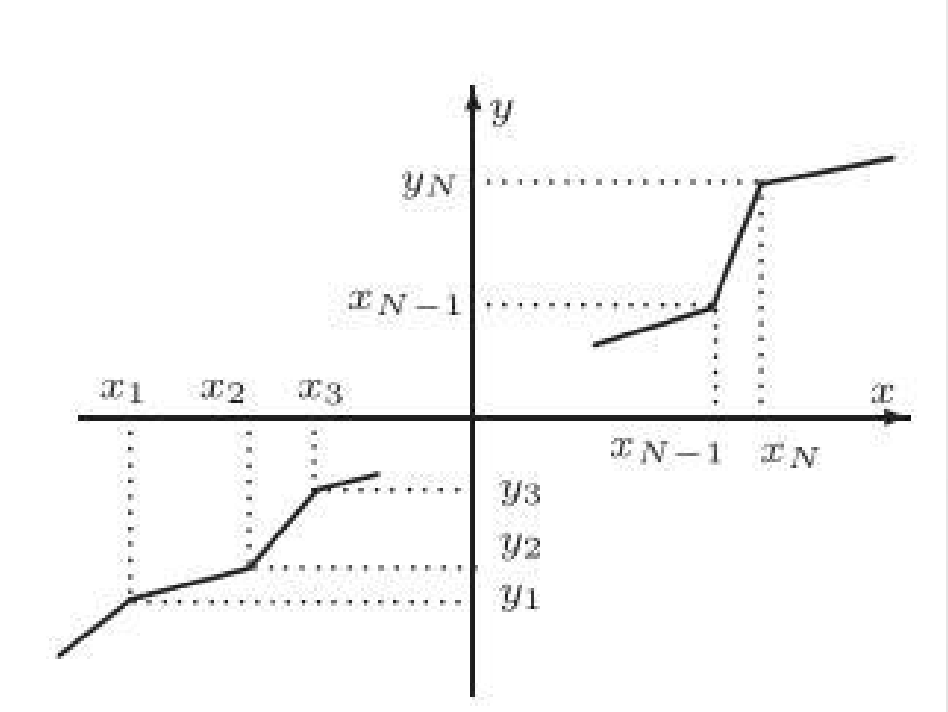
一般静态非线性特性的建模

- 静态非线性：$y=f(u)$
- 单值非线性环节
  - 转折点

    $\mathrm{xx}=[x_0,x_1,x_2,\cdots,x_N,x_{N+1}];$

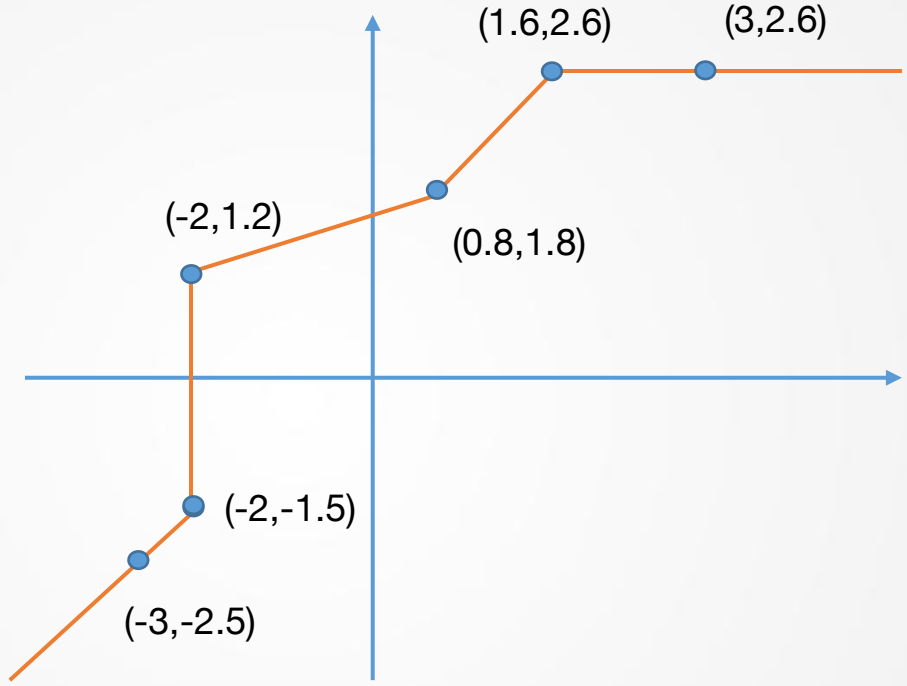
    $\mathrm{yy}=[y_0,y_1,y_2,\cdots,y_N,y_{N+1}];$
  - 一维查表模块





例6-10 单值非线性环节的建模

- 单值非线性
  - 找出转折点

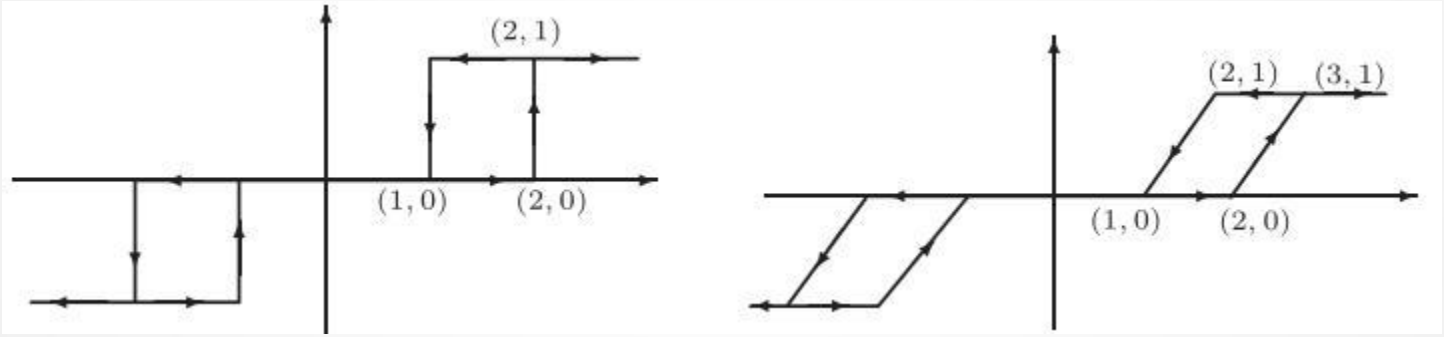


  - 利用查表包括实现
  - 如何处理直上直下的那条线

```
>> xx=[-3,-2,-2+eps,0.8,1.6,3]; yy=[-2.5 -1.5 1.2 1.8 2.6 2.6];
```

多值非线性环节建模

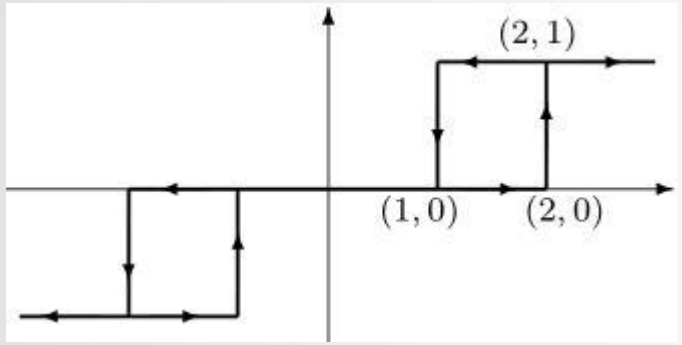
- 多值非线性在实际系统中存在——磁滞
- 双值非线性举例

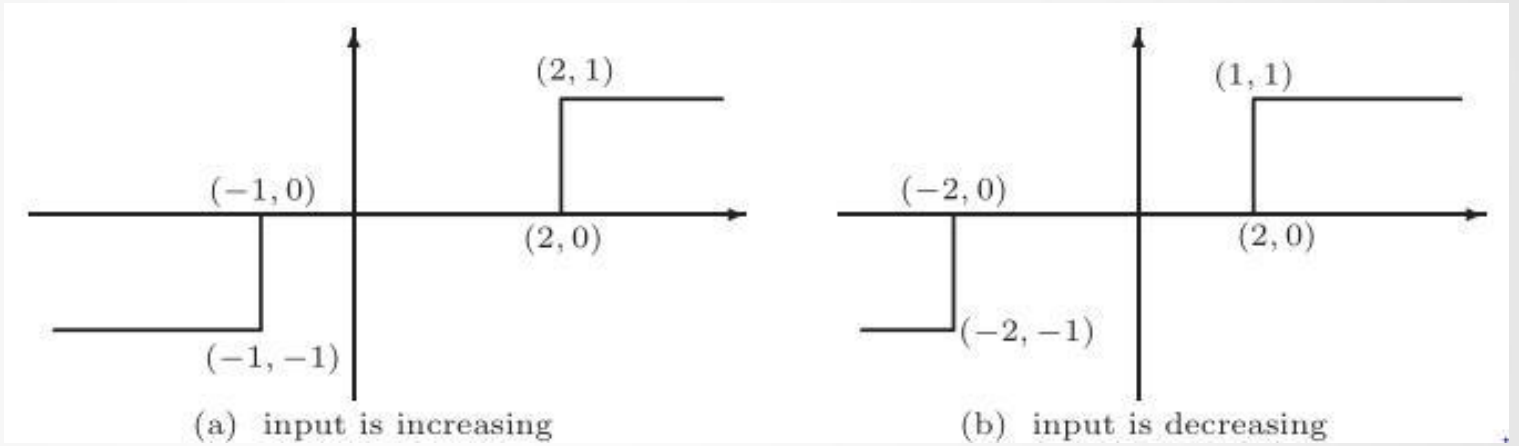


- 如何表示？想法：
  - 分成两个单值非线性
  - 识别输入信号上升或下降：开关、比较

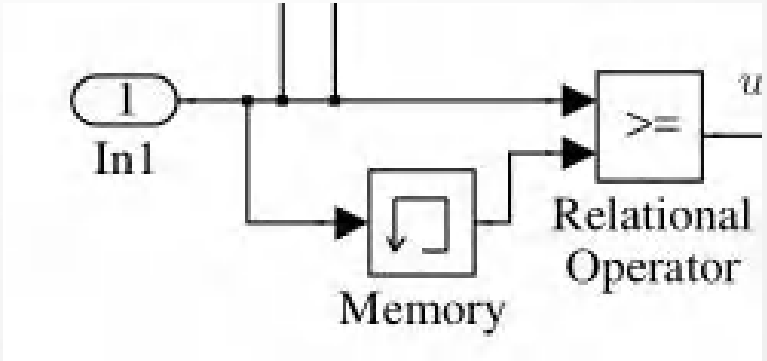
例6-11 多值非线性的实现

- 分解成两个单值非线性



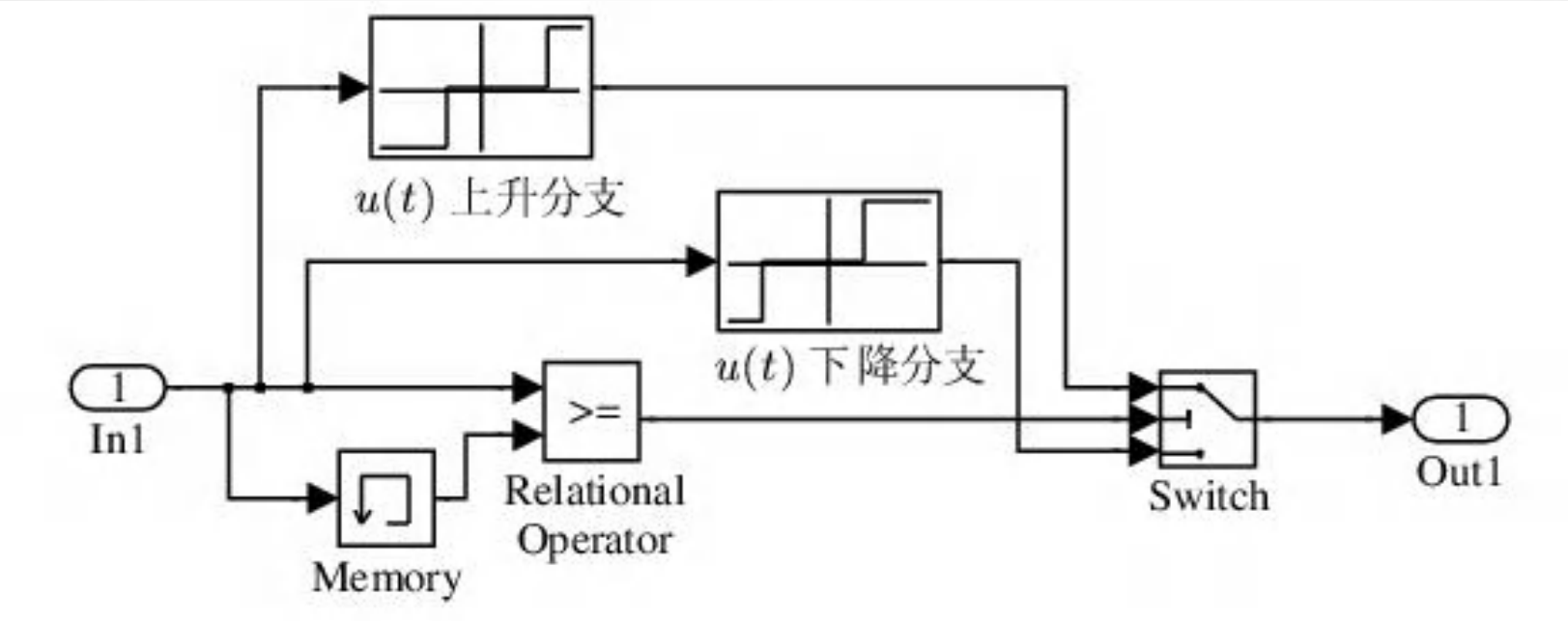


- 判定输入信号的增减
  - 得出前一步信号值
  - 和当前值比较



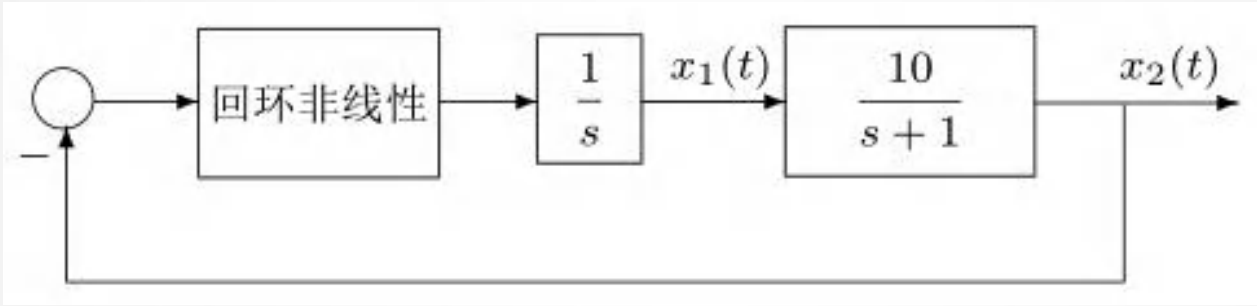
Simulink如何实现?

- Simulink实现模型

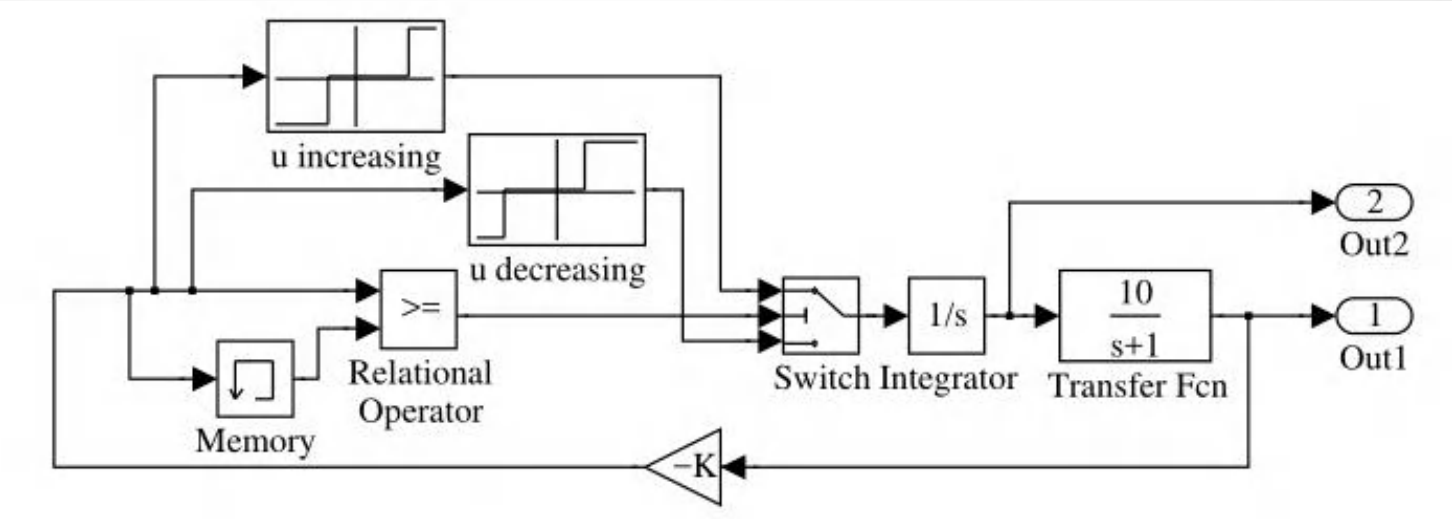


例6-12 非线性系统的极限环

- 闭环系统模型



- Simulink模型



闭环系统的仿真研究与极限环绘制

- Simulink模型名 c6mlimcy.mdl
- 非线性系统的仿真
  - 直接仿真
    - 终止仿真时间为40

```
>> [t,x,y]=sim('c6mlimcy',40); plot(t,y)
```

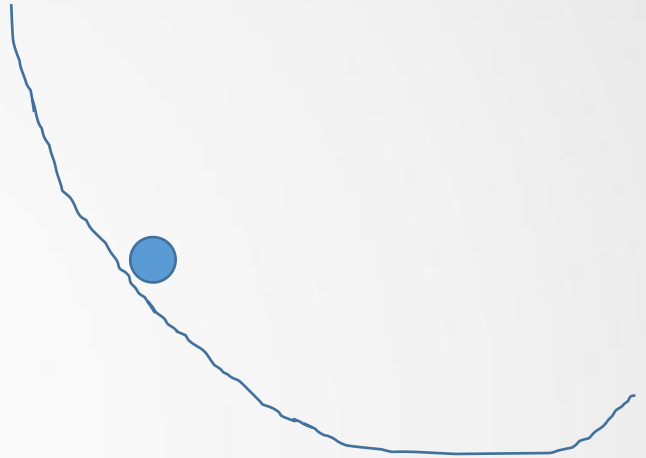
  - 极限环绘制

```
>> plot(y(:,1),y(:,2))
```

非线性系统的线性化

- 线性系统理论成熟
  - 非线性不严重可以用线性系统近似
  - 非线性系统在工作点附近用线性系统近似
- 非线性系统线性化的数学理论与实现
  - 工作点的计算
  - 线性化的数学表示
  - 基于MATLAB的直接计算
- 复杂线性系统模型的化简

平衡点的计算



- 非线性状态方程模型

$$\dot{x}_i(t) = f_i(t, \boldsymbol{x}, \boldsymbol{u}), \quad i = 1, 2, \cdots, n$$

- 平衡点的定义

$$\boldsymbol{y} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) = 0$$

- MATLAB直接求解

$$[\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}, \boldsymbol{x}_{\mathrm{d}}] = \mathtt{trim}(\mathtt{model}, \boldsymbol{x}_0, \boldsymbol{u}_0)$$

线性化的数学理论

- 状态方程的Taylor级数展开

$$\Delta\dot{x}_i = \sum_{j=1}^{n} \left.\frac{\partial f_i(t,\boldsymbol{x},\boldsymbol{u})}{\partial x_j}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} \Delta x_j + \sum_{j=1}^{p} \left.\frac{\partial f_i(t,\boldsymbol{x},\boldsymbol{u})}{\partial u_j}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} \Delta u_j$$

$$\boldsymbol{z}(t) = \Delta\boldsymbol{x}(t),\ \boldsymbol{v}(t) = \Delta u(t)$$

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{A}_{\mathrm{l}}\boldsymbol{z}(t) + \boldsymbol{B}_{\mathrm{l}}\boldsymbol{v}(t)$$

$$\boldsymbol{A}_{\mathrm{l}} = \begin{bmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \cdots & \partial f_n/\partial x_n \end{bmatrix}, \quad \boldsymbol{B}_{\mathrm{l}} = \begin{bmatrix} \partial f_1/\partial u_1 & \cdots & \partial f_1/\partial u_p \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial u_1 & \cdots & \partial f_n/\partial u_p \end{bmatrix}$$

线性化的MATLAB求解

- 有了Simulink模型就可以直接线性化

$$G=\texttt{linearize(model,op)}$$

- 早期版本命令

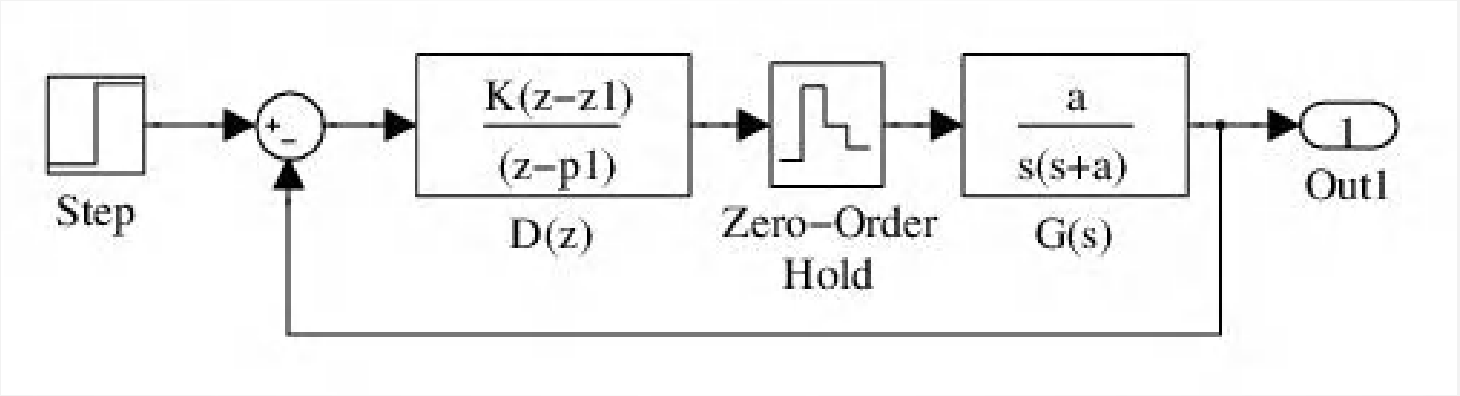
$$[A,B,C,D]=\texttt{linmod2(model,}x_0\texttt{,}u_0\texttt{)}$$

$$[A,B,C,D]=\texttt{linmod(model,}x_0\texttt{,}u_0\texttt{)}$$

$$[A,B,C,D]=\texttt{dlinmod(model,}x_0\texttt{,}u_0\texttt{)}$$

例6-13 离散系统的线性化

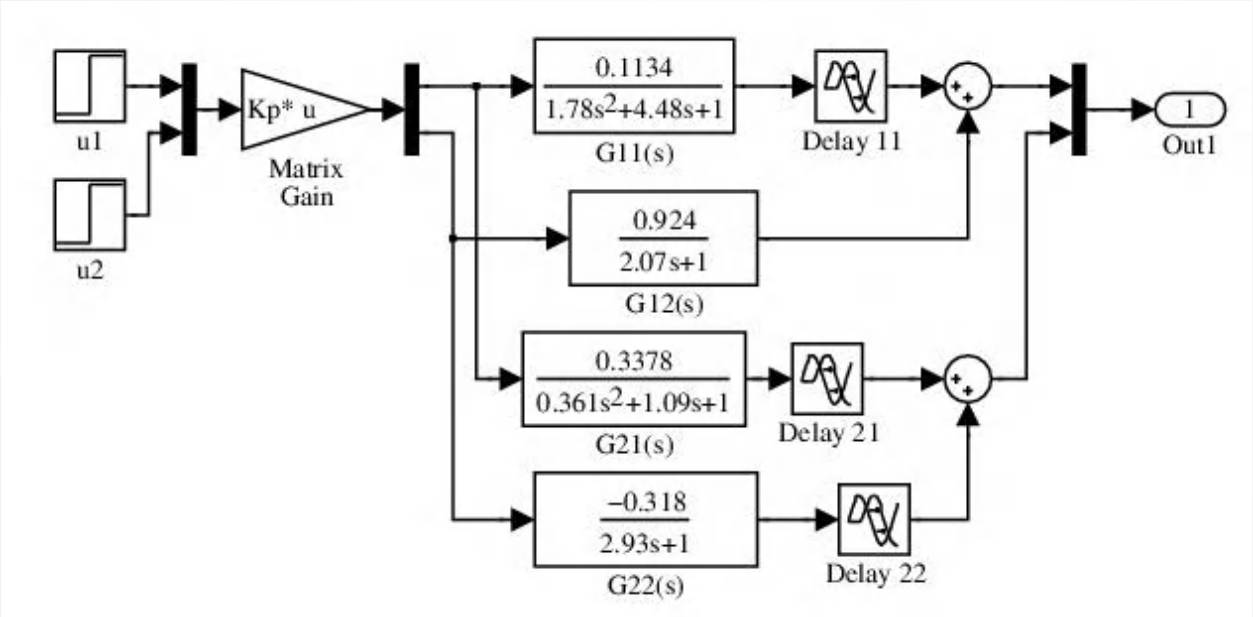
- 提取整个系统的线性模型



- G=zpk(linearize('c6mcomp2'))
- [A,B,C,D]=dlinmod('c6mcomp2'); G=zpk(ss(A,B,C,D,'Ts',0.2))

例6-14 连续系统的线性化

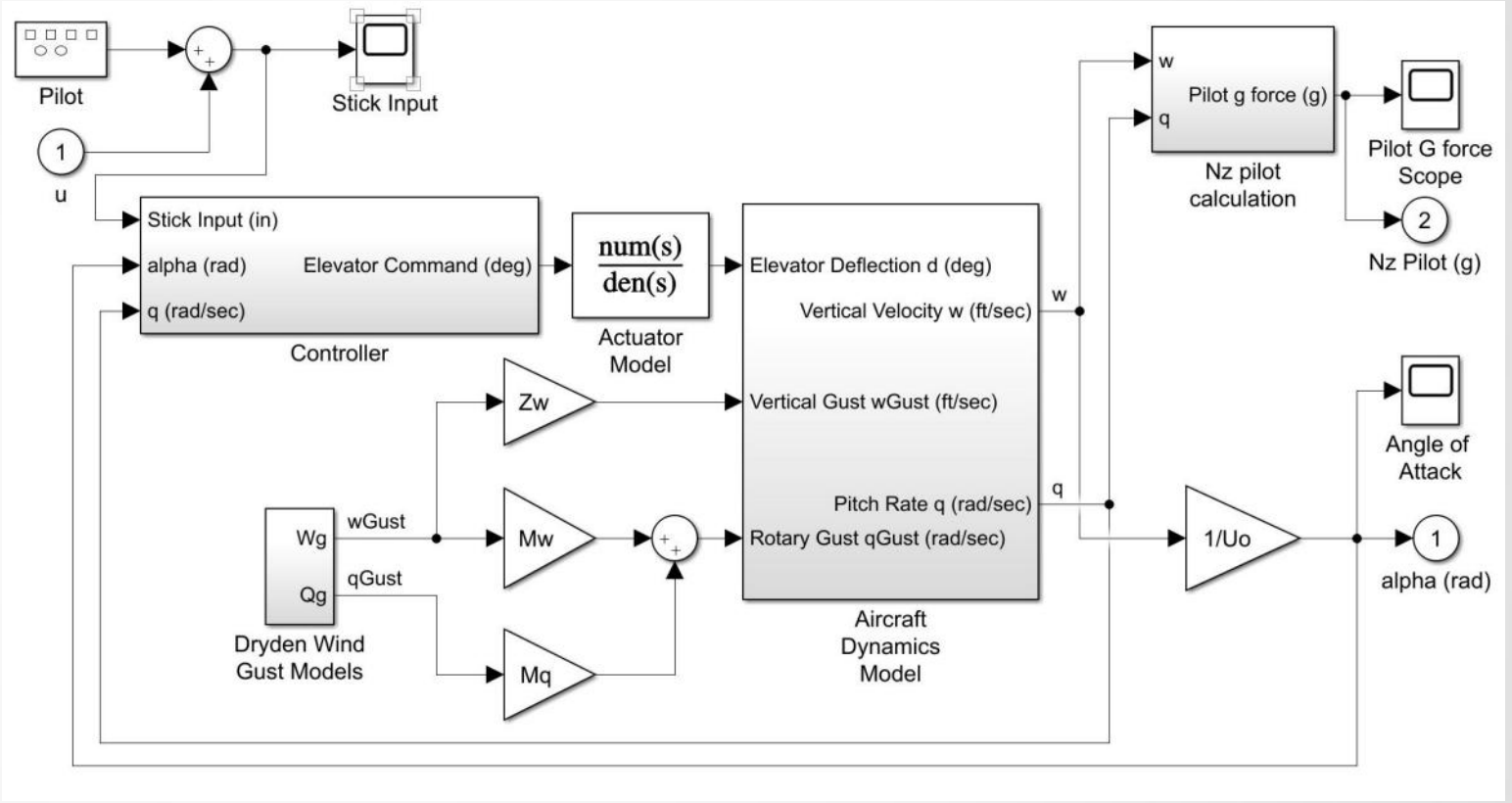
- 提取整个系统的线性模型



- Kp=[0.1134,0.924;0.3378,-0.318]; G=linearize('c6mmdly1’)
- [A,B,C,D]=linmod('c6mmdly1')

MATLAB提供了现成的F14 Simulink模型

- 仿真模型名称 f14



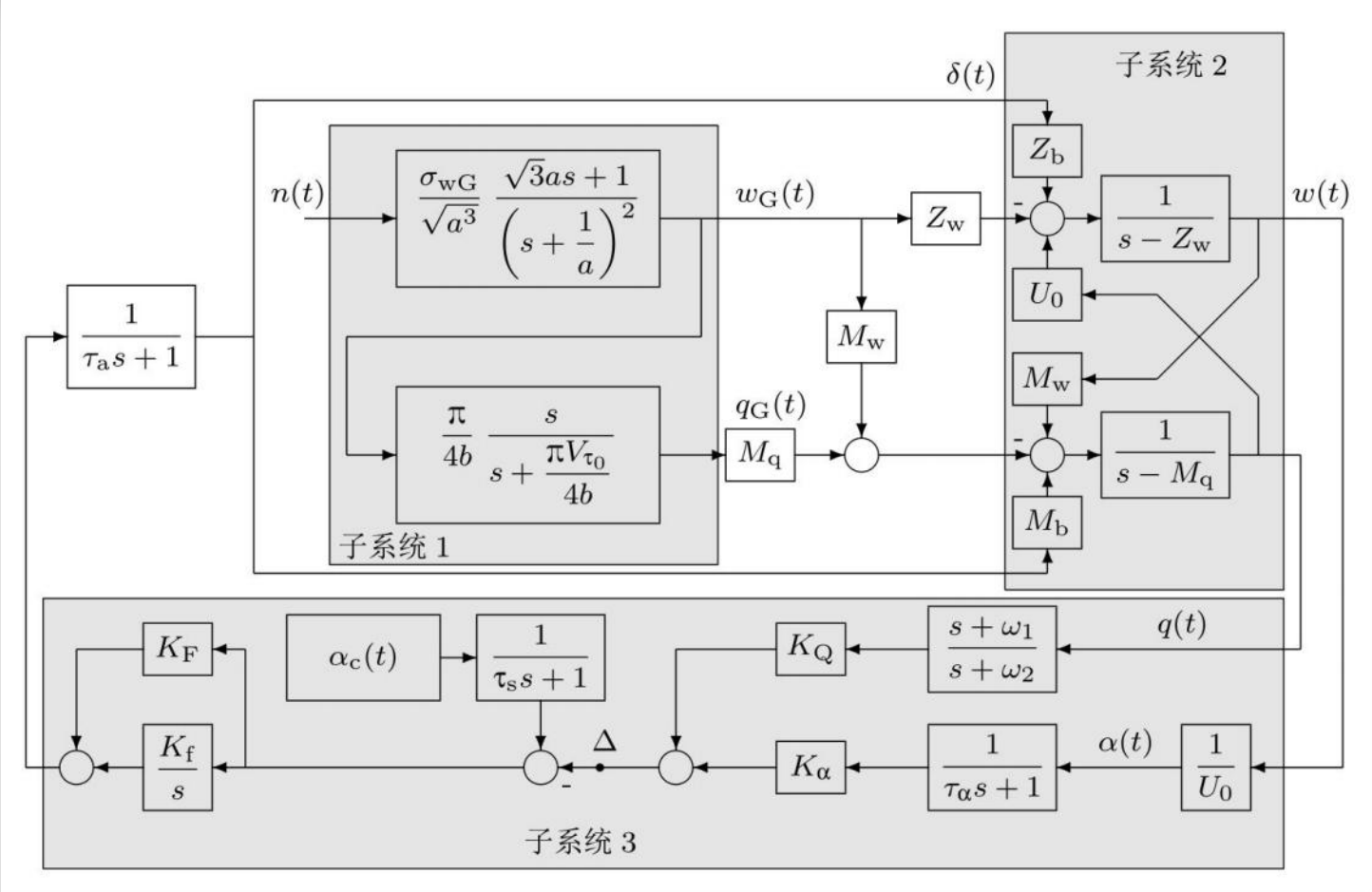
- 线性化命令与结果

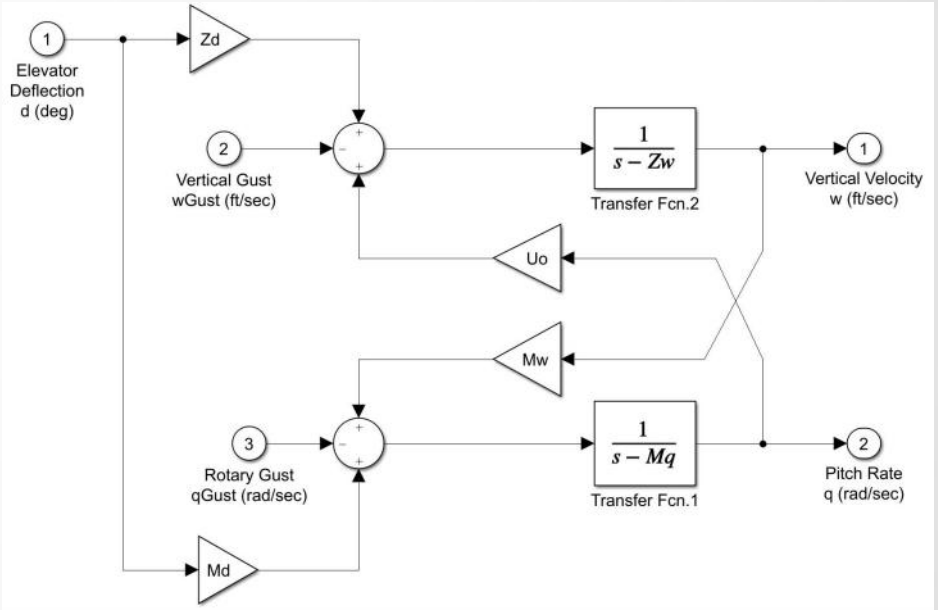
```
>> G=linearize('f14'), zpk(G)
```

非线性系统的建模与仿真

- 6.1 Simulink建模的基础知识
- 6.2 Simulink建模与仿真
- 6.3 控制系统的Simulink建模与仿真实例
- 6.4 非线性系统分析与仿真
- 6.5 子系统与模块封装技术
- 6.6 S-函数编写及其应用

例6-20 F14战斗机模型的子系统表示





例6-21 子系统的建立——PID控制器模块

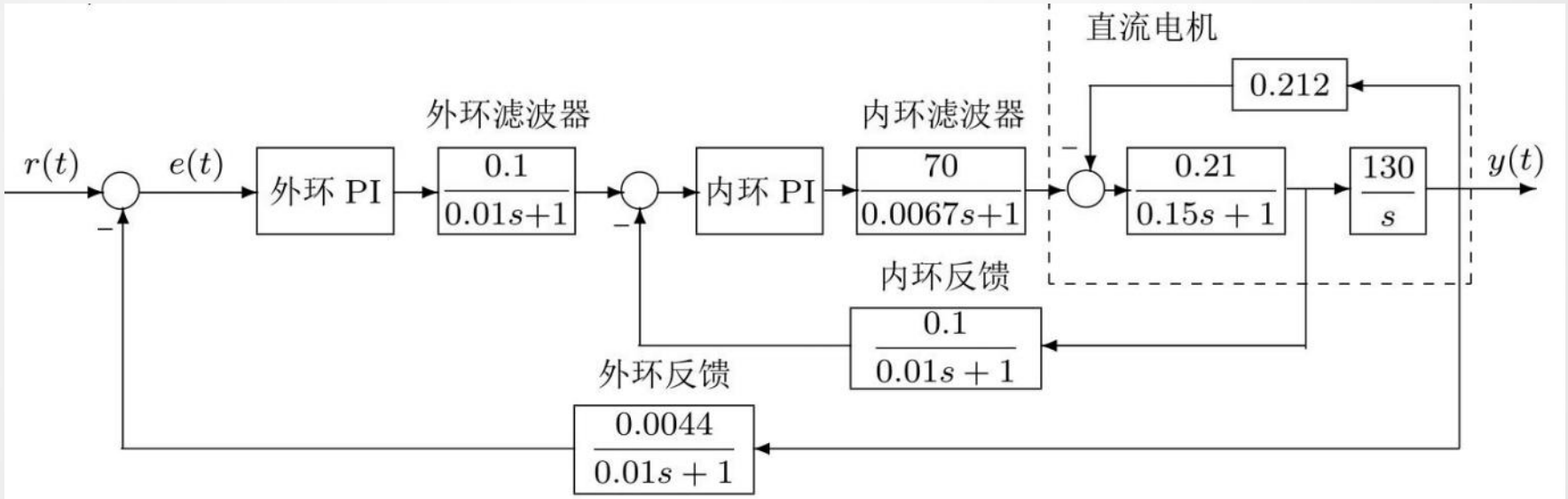
- 新版本有通用PID模块，这里只用于演示
- PID控制器数学模型

$$U(s) = K_{\mathrm{p}}\left(1 + \frac{1}{T_{\mathrm{i}}s} + \frac{sT_{\mathrm{d}}}{1 + sT_{\mathrm{d}}/N}\right)E(s)$$

- 如何建模：比例、积分器、微分器、加法器
  - 连线构造子系统
  - 子系统菜单

子系统的局限性

- 双闭环直流调速系统



- 能不能用PID子系统？

子系统的封装

- 子系统的局限性
  - 将其变换成相互独立的模块，互不影响
  - 在同一个系统中可以使用若干系统的封装模块
  - 参数可以独立设置
- 封装模块的设计
  - PID控制器为例
  - 图标设计、内部程序处理与设置

例6-23 PID控制器子系统的封装

- PID控制器的数学表达式

$$U(s) = K_{\mathrm{p}}\left(1 + \frac{1}{T_{\mathrm{i}}s} + \frac{sT_{\mathrm{d}}}{1 + sT_{\mathrm{d}}/N}\right)E(s)$$

- 现成的PID子系统——封装步骤
  - Edit -> Mask Subsystems
  - 设计图标
  - 设计参数对话框——选择变量名
  - 设计初始化

模块封装的图标设计

- 图标处理（Drawing commands）
  - 画曲线 (新版本支持后者)

```
plot(cos(0:.1:2*pi),sin(0:.1:2*pi))
```

```
t=0:.1:2*pi; plot(cos(t),sin(t));
plot(-0.4+0.1*cos(t),0.2+0.1*sin(t)); plot(0.4+0.1*cos(t),0.2+0.1*sin(t));
t=0:.1:pi; plot(0.6*cos(t),-0.2-0.4*sin(t));
```

  - 写文字

```
disp('PID\nController')
```

  - 加图像

```
image(imread('tiantan.jpg'))
```

参数对话框设计

- 4个参数 —— 设计参数对话框
  - Kp —— 比例系数
  - Ti —— 积分系数
  - Td —— 微分系数
  - N —— 滤波常数
- 对这种简单模块，参数将直接写入模块，无需进一步处理
- PID模块是可重用的

非线性系统的建模与仿真

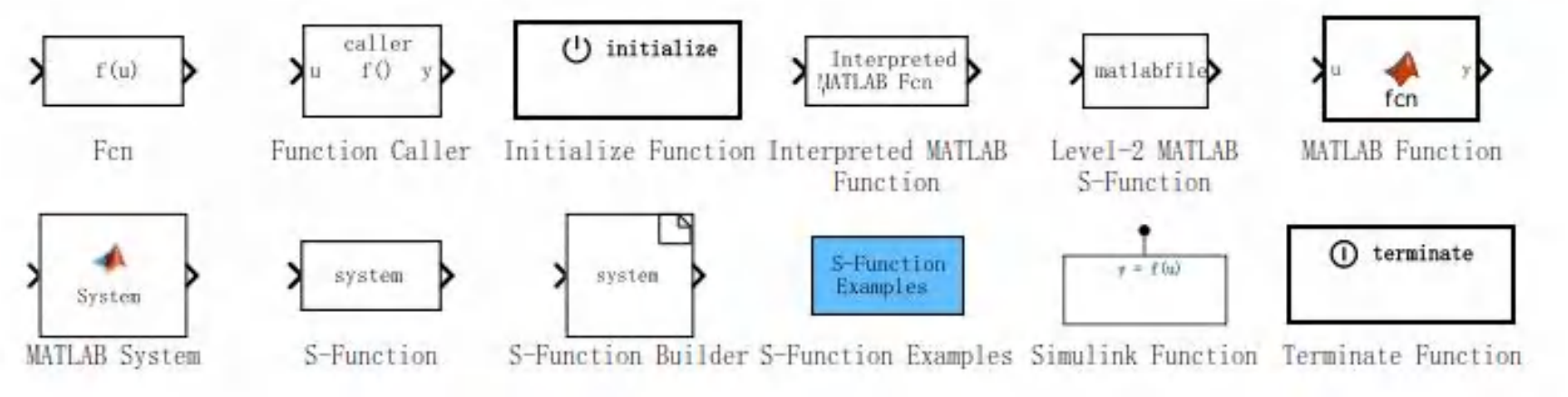
- 6.1 Simulink建模的基础知识
- 6.2 Simulink建模与仿真
- 6.3 控制系统的Simulink建模与仿真实例
- 6.4 非线性系统分析与仿真
- 6.5 子系统与模块封装技术
- 6.6 S-函数编写及其应用

M-函数与S-函数

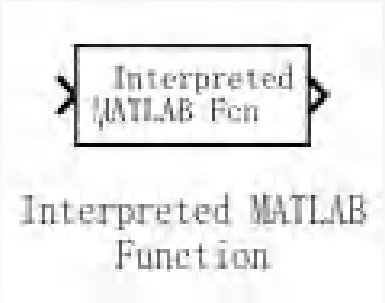
- 为什么需要M-函数或S-函数
  - 底层模块搭建繁琐
  - MATLAB语言、C、Ada等
- M-函数与S-函数
  - M-模块：静态关系 $y = f(u)$
  - S-函数——系统函数
    - 动态关系——状态方程、任意复杂系统
- S-函数的编程结构与框架
- S-函数举例

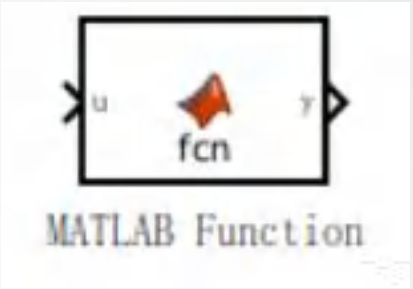
自定义函数模块组

- User-defined Functions
  - 弥补底层模块搭建的不足
  - 两类模块——静态与动态



M-函数





- 描述函数的静态关系
- M-函数模块
  - User-Defined Functions
- 饱和非线性的M-描述编程

```
function y=satur_non(x)
if abs(x)>=3, y=2*sign(x); else, y=2/3*x; end
```

- M-函数的局限性
  - 不能带有附加参数：宽度、斜率

为什么需要S-函数

- 例6-23 韩京清研究员提出的微分-跟踪器

$$\begin{cases} x_1(k+1) = x_1(k) + Tx_2(k) \\ x_2(k+1) = x_2(k) + T\mathrm{fst}(x_1(k), x_2(k), u(k), r, h) \end{cases}$$

$$\delta = rh, \quad \delta_0 = \delta h, \quad b = x_1 - u + hx_2, \quad a_0 = \sqrt{\delta^2 + 8r|b|}$$

$$a = \begin{cases} x_2 + b/h, & |b| \leqslant \delta_0 \\ x_2 + 0.5(a_0 - \delta)\mathrm{sign}(b), & |b| > \delta_0 \end{cases}$$

$$\mathrm{fst} = \begin{cases} -ra/\delta, & |a| \leqslant \delta \\ -r\mathrm{sign}(a), & |a| > \delta \end{cases}$$

S-函数(状态方程)的基本结构

- S-函数入口语句

  function [sys,x_0,str,ts]=fun(t,x,u,flag,p_1,p_2,$\cdots$)

- S-函数的基本框架

```
switch flag
case 0, [sys,x0,str,ts] = mdlInitializeSizes(T);
case 1, sys = mdlDeravitivess(x,u,r,h,T);
case 2, sys = mdlUpdates(x,u,r,h,T);
case 3, sys = mdlOutputs(x);
end
```

S-函数的执行过程

- flag变量的自动取值
  - 初始化：flag=0, 输入、输出路数、状态数等
  - flag=3, 计算输出信号
  - flag=1, 2，分别更新状态：连续、离散状态方程
- S-函数用模块表示

```
switch flag
case 0, [sys,x0,str,ts] = mdlInitializeSizes(T);
case 1, sys = mdlDeravitivess(x,u,r,h,T);
case 2, sys = mdlUpdates(x,u,r,h,T);
case 3, sys = mdlOutputs(x);
end
```

S-函数的初始化

- 由simsizes读入模板，成员变量
  - .NumContStates，S-函数中连续状态的个数
  - .NumDiscStates，表示离散状态的个数
  - .NumInputs，模块的输入路数
  - .NumOutputs，表示模块的输出路数
  - .DirFeedthrough，输出方程是否显含 u
  - .NumSampleTimes，采样周期的个数
- 由simsizes返回
- 计算初值、采样周期等

其他函数结构

- 系统的状态 $\boldsymbol{x} = [\boldsymbol{x}_{\mathrm{c}}, \boldsymbol{x}_{\mathrm{d}}]^{\mathrm{T}}$
  - 连续状态更新 $\dot{\boldsymbol{x}}_{\mathrm{c}}(t) = \boldsymbol{F}(t, \boldsymbol{x}, \boldsymbol{u})$

```
function sys = mdlDeravitivess(x,u,r,h,T);
```

  - 离散状态更新 $\boldsymbol{x}_{\mathrm{d}}(k+1) = \boldsymbol{G}(t, \boldsymbol{x}(k), \boldsymbol{u}(k))$

```
function sys = mdlUpdates(x,u,r,h,T);
```

- 输出计算 $\boldsymbol{y}(t) = \boldsymbol{H}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))$

```
function sys = mdlOutputs(x);
```

例6-25 微分跟踪器编程实现

- 韩京清研究员的微分-跟踪器

$$\begin{cases} x_1(k+1) = x_1(k) + Tx_2(k) \\ x_2(k+1) = x_2(k) + T\mathrm{fst}(x_1(k), x_2(k), u(k), r, h) \end{cases}$$

$$\delta = rh, \quad \delta_0 = \delta h, \quad b = x_1 - u + hx_2, \quad a_0 = \sqrt{\delta^2 + 8r|b|}$$

$$a = \begin{cases} x_2 + b/h, & |b| \leqslant \delta_0 \\ x_2 + 0.5(a_0 - \delta)\mathrm{sign}(b), & |b| > \delta_0 \end{cases} \qquad \mathrm{fst} = \begin{cases} -ra/\delta, & |a| \leqslant \delta \\ -r\mathrm{sign}(a), & |a| > \delta \end{cases}$$

- S函数准备
  - 模块的输出方程 $y = x$
  - 状态个数、输入输出路数、附加变量 (r, h, T)

主程序的设计

- 主函数

```
function [sys,x0,str,ts]=han_td(t,x,u,flag,r,h,T)
switch flag
case 0, [sys,x0,str,ts] = mdlInitializeSizes(T);
case 2, sys = mdlUpdates(x,u,r,h,T);
case 3, sys = mdlOutputs(x);
case {1, 4, 9}, sys = [];
otherwise, error(['Unhandled flag = ',num2str(flag)]);
end;
```

- 输出方程

```
function sys = mdlOutputs(x), sys=x;
```

初始化函数编写

```
function [sys,x0,str,ts] = mdlInitializeSizes(T)
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 2;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0; 0];
str = [];
ts = [T 0];
```

离散状态更新程序

- 没有连续状态，无需编程
- 离散状态更新

$$\begin{cases} x_1(k+1) = x_1(k) + Tx_2(k) \\ x_2(k+1) = x_2(k) + T\mathrm{fst}(x_1(k), x_2(k), u(k), r, h) \end{cases}$$

- 支持函数

```
function sys = mdlUpdates(x,u,r,h,T)
sys(1,1)=x(1)+T*x(2);
sys(2,1)=x(2)+T*fst2(x,u,r,h);
```

支持函数编写

数学公式

$$\delta = rh, \quad \delta_0 = \delta h, \quad b = x_1 - u + hx_2, \quad a_0 = \sqrt{\delta^2 + 8r|b|}$$

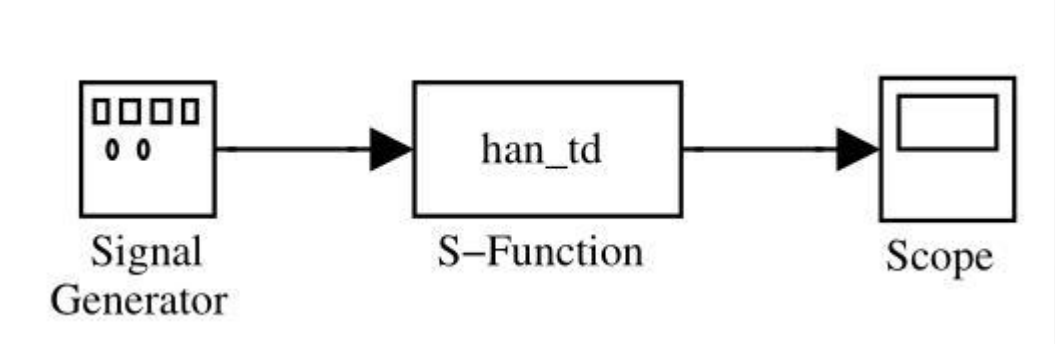
$$a = \begin{cases} x_2 + b/h, & |b| \leqslant \delta_0 \\ x_2 + 0.5(a_0 - \delta)\mathrm{sign}(b), & |b| > \delta_0 \end{cases} \qquad \mathrm{fst} = \begin{cases} -ra/\delta, & |a| \leqslant \delta \\ -r\mathrm{sign}(a), & |a| > \delta \end{cases}$$

程序实现

```
function f=fst2(x,u,r,h)
delta=r*h; delta0=delta*h; b=x(1)-u+h*x(2);
a0=sqrt(delta*delta+8*r*abs(b));
a=x(2)+b/h*(abs(b)<=delta0)+0.5*(a0-delta)*sign(b)*(abs(b)>delta0);
f=-r*a/delta*(abs(a)<=delta)-r*sign(a)*(abs(a)>delta);
```

模块的使用举例

- 用正弦信号去激励S-函数模块
- 得出跟踪信号与其导数信号
- 仿真模型 c6msf2



非线性系统线性化小结

- 线性化的作用
  - 非线性系统 —— trim, linearize
    - 在平衡点附近作线性近似
    - 平衡点的概念与计算
    - 线性化的直接计算
  - 线性系统
    - 线性系统的模型化简

子系统小结

- 子系统的作用
  - 将大系统按功能分成相互关联的子系统
  - 化整为零，模型更具结构化，更易于处理
- 子系统构造
  - 以PID控制器为例演示了子系统建模的全过程
- 子系统面临的问题

子系统封装小结

- 对子系统操作可以封装
  - 图标设计
  - 内部模型与代码处理

S-函数小结

- 为什么需要M-函数与S-函数
- M-函数的编程实现与局限性
- S-函数的结构与执行机制
  - Switch case标准框架 —— 适用于所有S-函数
  - 初始化 flag=0
  - 计算模块的输出, flag=3
  - 状态更新——离散与连续状态, flag=1,2

Q & A

感谢您的聆听和反馈