自动控制原理Ⅱ：线性系统分析与设计课内实验

系统分析部分

概述：本部分实验主要内容

- 定量分析：状态空间表达式的解
- 定性分析：能控性与能观性及与之其相关的状态空间模型变换
- 定性分析：稳定性

概述：重点&预备知识

- 重点
  - 系统解的获取，即系统响应曲线的绘制
  - 线性定常系统能控性、能观性的判定，及其相关的坐标变换
  - 线性定常系统稳定性判定
- 预备知识
  - MATLAB基础：矩阵运算、曲线绘制、等常规操作
  - 线性系统基础：系统的解、能控性、能观性、稳定性、等理论知识

课内实验：分析部分

2.1 状态空间表达式的解（系统响应曲线绘制）

2.2 线性定常系统的能控性/能观性判定及结构分解

2.3 线性定常系统的稳定性判定

状态空间表达式的解

- 用途：绘制系统响应曲线，测试系统某些响应，验证控制器性能
- 关键：获得不同时刻t时的解x(t)，形成系列(t,x(t))对（即点），逐点连线
- 方法：

➢ 计算状态转移矩阵

$$x(t) = \Phi(t)x(0) + \int_0^t \Phi(t-\tau)Bu(\tau)d\tau$$

➢ 利用MATLAB内嵌函数

dsolve

Ordinary differential equation and system solver

ode23

Solve nonstiff differential equations; low order method

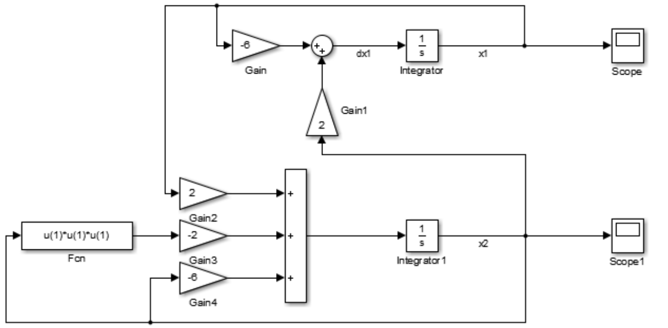
deval

Evaluate solution of differential equation problem

See Also

bvp4c | bvp5c | dde23 | ddensd | ddesd | ode113 | ode15i | ode15s | ode23 | ode23s | ode23t | ode23tb | ode45

➢ 利用Simulink搭建结构模型




```
ode45 (Dormand-Prince)
discrete (no continuous states)
ode45 (Dormand-Prince)
ode23 (Bogacki-Shampine)
ode113 (Adams)
ode15s (stiff/NDF)
ode23s (stiff/Mod. Rosenbrock)
ode23t (mod. stiff/Trapezoidal)
ode23tb (stiff/TR-BDF2)
```

状态空间表达式的解

- 计算状态转移矩阵

$$x(t) = \Phi(t)x(0) + \int_0^t \Phi(t-\tau)Bu(\tau)d\tau$$

expm()函数、ilaplace()函数

功能：计算状态转移矩阵

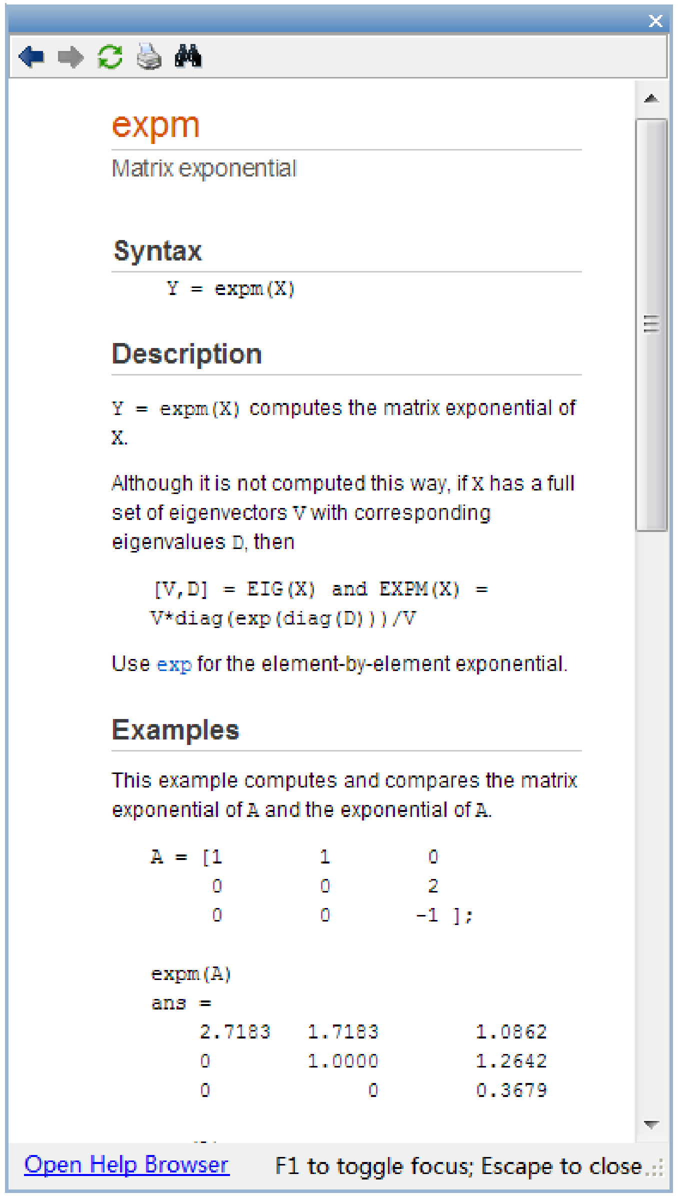
格式：

eAt = expm(A*t)

eAt = ilaplace(FS,s,t)

其中，s，t为定义的符号变量；FS为预解矩阵

$$FS = (s\boldsymbol{I} - \boldsymbol{A})^{-1}$$



expm

Matrix exponential

Syntax

```
Y = expm(X)
```

Description

`Y = expm(X)` computes the matrix exponential of `X`.

Although it is not computed this way, if `X` has a full set of eigenvectors `V` with corresponding eigenvalues `D`, then

```
[V,D] = EIG(X) and EXPM(X) = V*diag(exp(diag(D)))/V
```

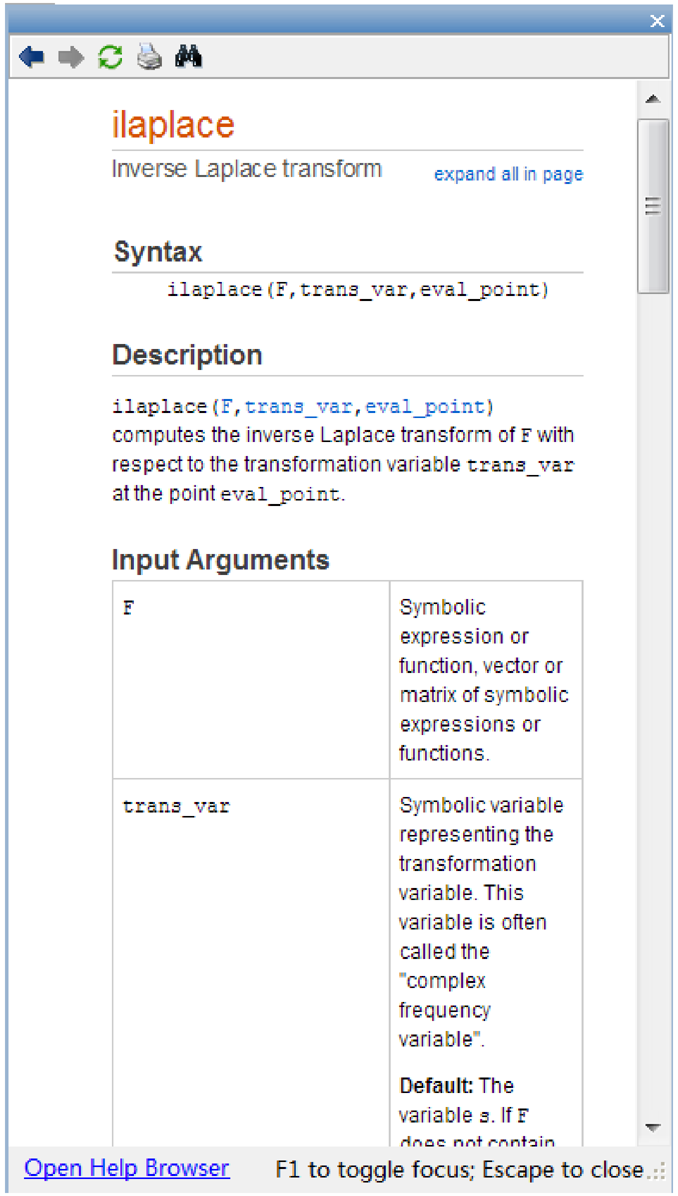
Use exp for the element-by-element exponential.

Examples

This example computes and compares the matrix exponential of A and the exponential of A.

```
A = [1      1      0
     0      0      2
     0      0     -1 ];

expm(A)
ans =
     2.7183    1.7183      1.0862
     0         1.0000      1.2642
     0              0      0.3679
```

Open Help Browser   F1 to toggle focus; Escape to close



ilaplace

Inverse Laplace transform   expand all in page

Syntax

```
ilaplace(F,trans_var,eval_point)
```

Description

`ilaplace(F,trans_var,eval_point)` computes the inverse Laplace transform of `F` with respect to the transformation variable `trans_var` at the point `eval_point`.

Input Arguments

`F`	Symbolic expression or function, vector or matrix of symbolic expressions or functions.
`trans_var`	Symbolic variable representing the transformation variable. This variable is often called the "complex frequency variable". **Default:** The variable `s`. If `F` does not contain

Open Help Browser   F1 to toggle focus; Escape to close

状态空间表达式的解

例2-1：计算状态转移矩阵，绘制零输入响应

$$
\begin{cases}
\dot{x}(t) = \begin{bmatrix} -3 & 1 \\ 1 & -3 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \\
u(t) = 0 \\
x(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}
\end{cases}
$$

$$\boldsymbol{x}(t) = \boldsymbol{\Phi}(t)\boldsymbol{x}(0)$$

$$\boldsymbol{x}(t) = \boldsymbol{\Phi}(t)\boldsymbol{x}(0) + \int_0^t \boldsymbol{\Phi}(t-\tau)\boldsymbol{B}\boldsymbol{u}(\tau)d\tau$$

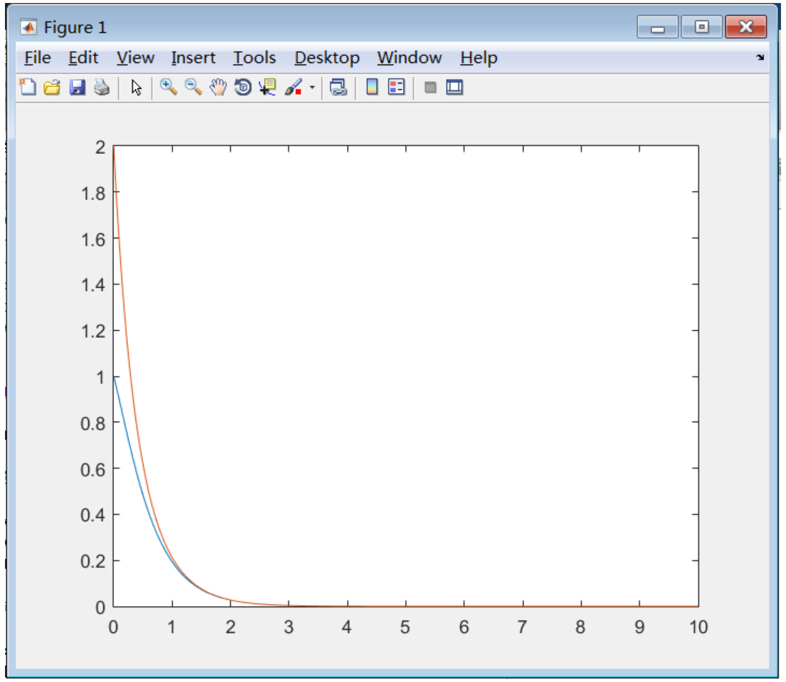
思考：绘制零状态响应？

Editor - ex2_1.m

```matlab
%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-07
%%% Example 2-1
%%% 计算状态转移矩阵，绘制零输入响应
%%%%%%%%%%%%%%%%

clc
clear all

A = [-3 1; 1 -3];

% 计算状态转移矩阵
syms s t                  % MATLAB符号运算，定义符号
eAt1 = expm(A*t)          % 方法1
FS = inv(s*eye(2)-A);     % 预解矩阵
eAt2 = ilaplace(FS,s,t)   % 方法2

% 绘制零输入响应
x0 = [1; 2];              % 定义初值
Time = 0:0.01:10;         % 定义时间区间和间隔，即曲线横坐标数据
xTime = [];               % 存放求得的xt，即曲线纵坐标数据
for t = Time
    eAt1 = expm(A*t);
    xt = eAt1*x0;
    xTime = [xTime xt];
end
plot(Time,xTime) % 绘制响应曲线
```

Command Window

```
eAt1 =

[ exp(-2*t)/2 + exp(-4*t)/2, exp(-2*t)/2 - exp(-4*t)/2]
[ exp(-2*t)/2 - exp(-4*t)/2, exp(-2*t)/2 + exp(-4*t)/2]


eAt2 =

[ exp(-2*t)/2 + exp(-4*t)/2, exp(-2*t)/2 - exp(-4*t)/2]
[ exp(-2*t)/2 - exp(-4*t)/2, exp(-2*t)/2 + exp(-4*t)/2]

fx >>
```



状态空间表达式的解

- 利用MATLAB内嵌函数

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n$$
$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n$$
$$\vdots$$
$$\dot{x}_n = a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n$$

dsolve()函数

功能：求解微分方程

格式：

r = dsolve(‘eqs’, ‘conds’, ‘v’)

其中，eqs为方程；conds为约束条件(如初值)；v为自变量

dsolve

Ordinary differential equation and system solver

expand all in page

Syntax

```
S = dsolve(eqn)
S = dsolve(eqn,cond)
S = dsolve(eqn,cond,Name,Value)
Y = dsolve(eqns)
Y = dsolve(eqns,conds)
Y = dsolve(eqns,conds,Name,Value)
[y1,...,yN] = dsolve(eqns)
[y1,...,yN] = dsolve(eqns,conds)
[y1,...,yN] = dsolve(eqns,conds,Name,Value)
```

Description

`S = dsolve(eqn)` solves the ordinary differential equation `eqn`. Here `eqn` is a symbolic equation containing `diff` to indicate derivatives. Alternatively, you can use a string with the letter `D` indicating derivatives. For example, `syms y(x); dsolve(diff(y) == y + 1)` and `dsolve('Dy = y + 1','x')` both solve the equation `dy/dx = y + 1` with respect to the variable `x`. Also, `eqn` can be an array of such equations or strings.

`S = dsolve(eqn,cond)` solves the ordinary differential equation `eqn` with the initial or boundary condition `cond`.

`S = dsolve(eqn,cond,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

`Y = dsolve(eqns)` solves the system of ordinary differential equations `eqns` and returns a structure array that contains the solutions. The number of fields in the structure array corresponds to the number of independent variables in the system.

`Y = dsolve(eqns,conds)` solves the system of ordinary differential equations `eqns` with the initial or boundary conditions `conds`.

`Y = dsolve(eqns,conds,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

`[y1,...,yN] = dsolve(eqns)` solves the system of ordinary differential equations `eqns` and assigns the solutions to the variables `y1,...,yN`.

`[y1,...,yN] = dsolve(eqns,conds)` solves the system of ordinary differential equations `eqns` with the initial or boundary conditions `conds`.

`[y1,...,yN] = dsolve(eqns,conds,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

Open Help Browser

F1 to toggle focus; Escape to close

状态空间表达式的解

例2-2：求解微分方程，并绘制零输入曲线

$$\begin{cases} \dot{x}(t)=\begin{bmatrix} -3 & 1 \\ 1 & -3 \end{bmatrix}x(t)+\begin{bmatrix} 1 \\ 0 \end{bmatrix}u(t) \\ u(t)=0 \\ x(0)=\begin{bmatrix} 1 \\ 2 \end{bmatrix} \end{cases}$$

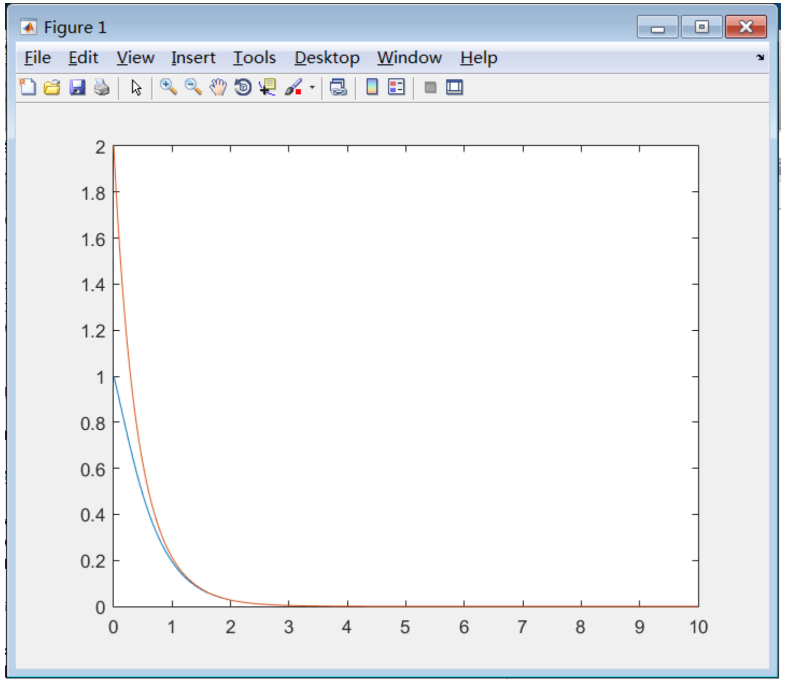
Editor - ex2_2.m

ex2_1.m ex2_2.m

```
%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-07
%%% Example 2-2
%%% 求解齐次常微分方程，绘制零输入响应
%%%%%%%%%%%%%%%%

clc
clear all

A = [-3 1; 1 -3];
x0 = [1; 2];

% 求解齐次常微分方程
r = dsolve('Dx1=-3*x1+1*x1,Dx2 = x1-3*x2','x1(0) = 1,x2(0)=2');
x1 = r.x1
x2 = r.x2

% 绘制零输入响应
Time = 0:0.01:10;        % 定义时间区间和间隔，即曲线横坐标数据
xTime = [];              % 存放求得的xt，即曲线纵坐标数据
for t = Time
    xt = [exp(-2*t); exp(-2*t)+exp(-3*t)]  % 利用计算出来的[x1;x2]
    xTime = [xTime xt];
end
plot(Time,xTime) % 绘制响应曲线
```

Command Window

```
x1 =

exp(-2*t)


x2 =

exp(-2*t) + exp(-3*t)

fx >>
```



思考：零状态响应？

状态空间表达式的解

- ❑ 利用MATLAB内嵌函数

step()函数，initial ()函数，lsim()函数

功能：计算LTI系统的单位阶跃响应、零输入响应、任意输入响应

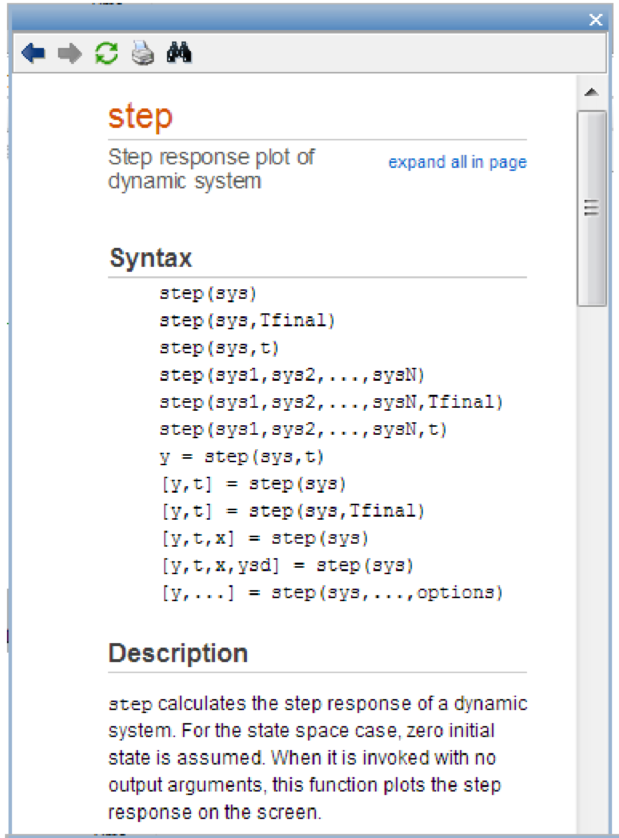
格式：

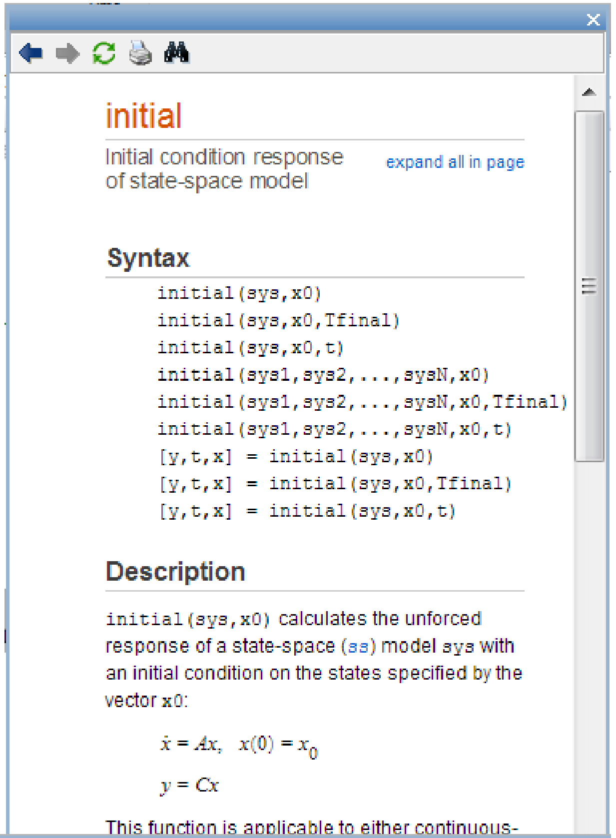
step(A,B,C,D)

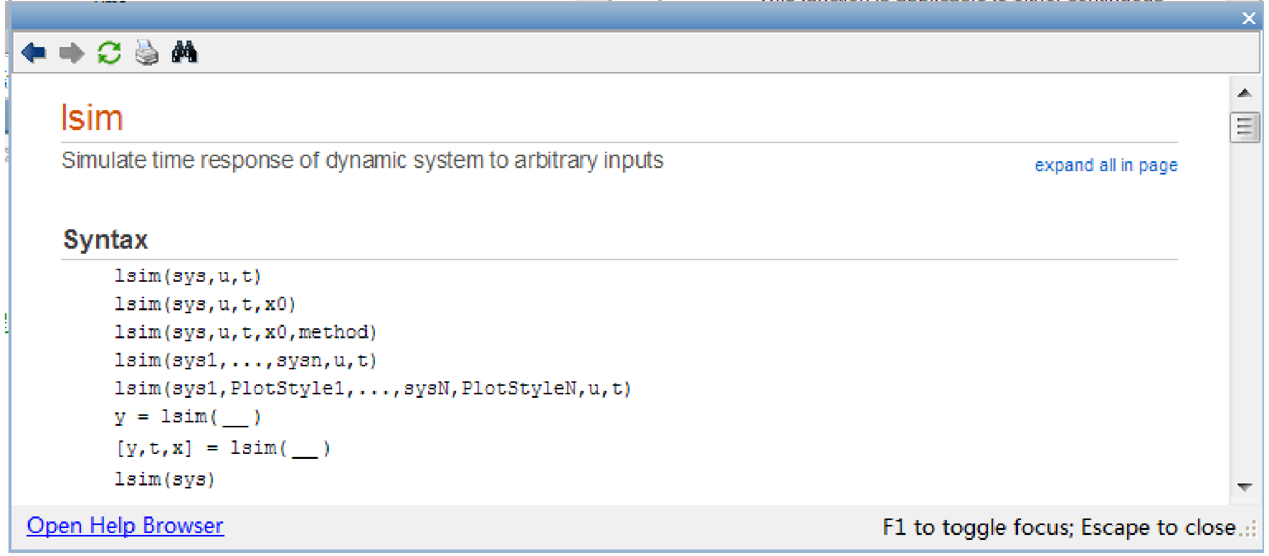
initial(A,B,C,D,t,x0)

lsim(A,B,C,D,u,t,x0)

其中，x0为初始条件；u为输入







状态空间表达式的解

例2-3：求解系统

- ✓ 单位阶跃响应
- ✓ 零输入响应
- ✓ 零状态响应

$$
\begin{cases}
\dot{x}(t)=\begin{bmatrix}-3 & 1\\ 1 & -3\end{bmatrix}x(t)+\begin{bmatrix}1\\0\end{bmatrix}u(t)\\
y(t)=[1\quad 3]x(t)\\
1.\ u(t)=1\\
2.\ x(0)=\begin{bmatrix}1\\2\end{bmatrix},u(t)=0\\
3.\ x(0)=\begin{bmatrix}0\\0\end{bmatrix},u(t)=2
\end{cases}
$$

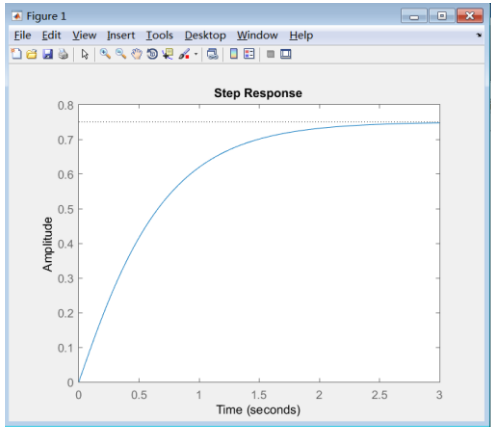
思考：全响应

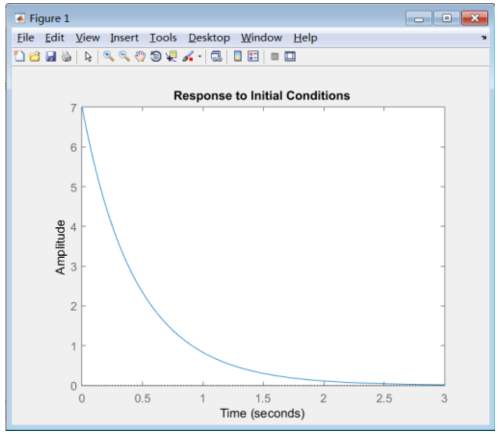
$$4.\ x(0)=\begin{bmatrix}1\\2\end{bmatrix},u(t)=\sin(t)$$

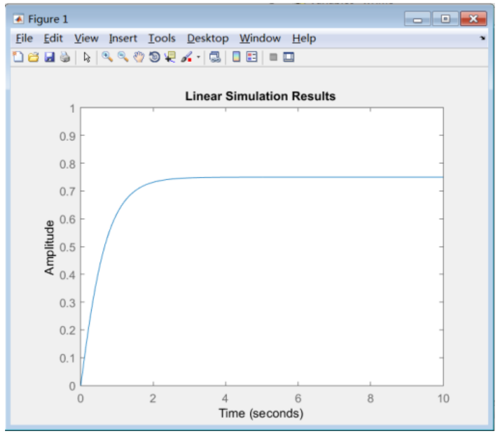
Editor - ex2_3.m

ex2_1.m | ex2_2.m | ex2_3.m

```
%%%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-07
%%% Example 2-3
%%% 求解LTI系统的单位阶跃响应、零状态响应、零输入响应
%%%%%%%%%%%%%%%%%%

clc
clear all

A = [-3 1; 1 -3];
B = [1; 0];
C = [1 3];
D = 0;

% 单位阶跃响应
step(A,B,C,D);

% 零状态响应
x0 = [1; 2];
initial(A,B,C,D,x0)

% 零状态响应
x0 = [0; 0];
t = 0:0.01:10;
u = ones(1,size(t,2));
lsim(A,B,C,D,u,t,x0)
```



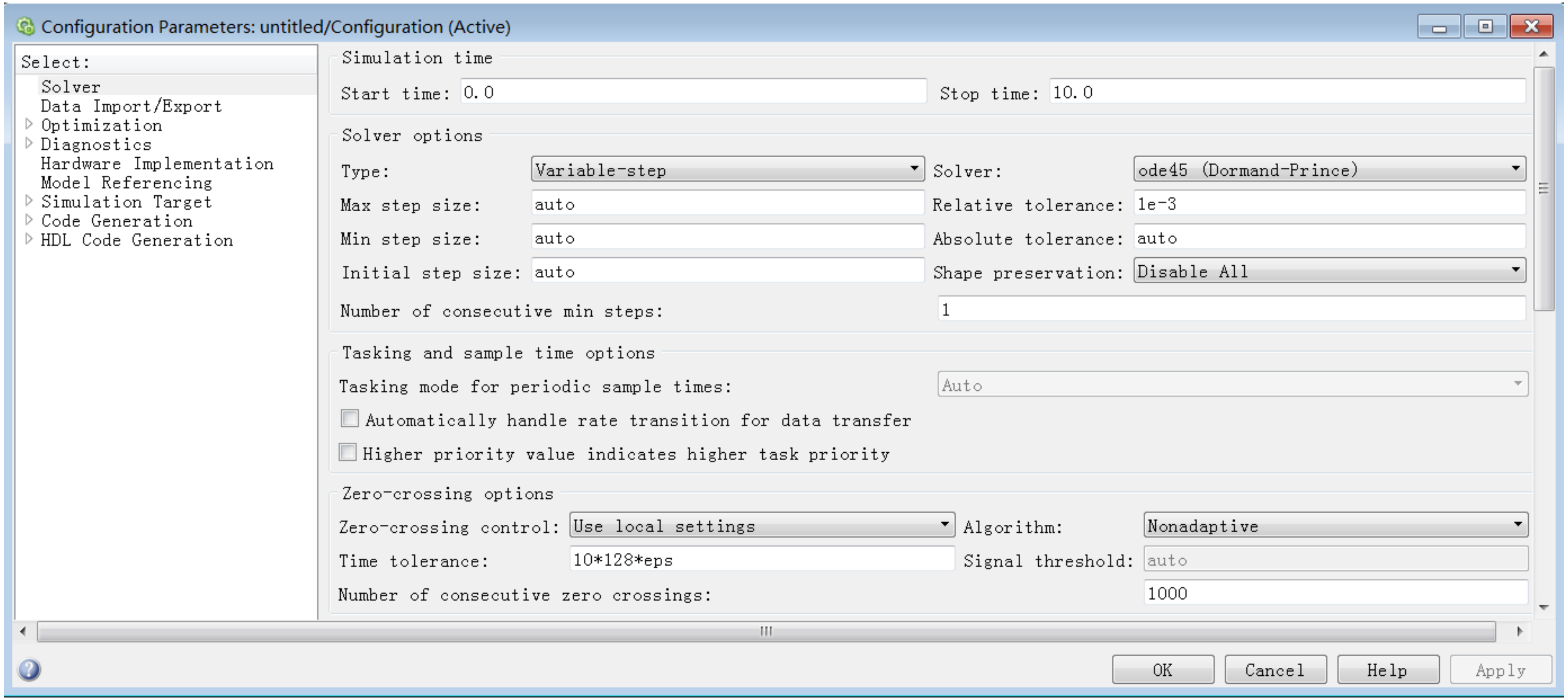


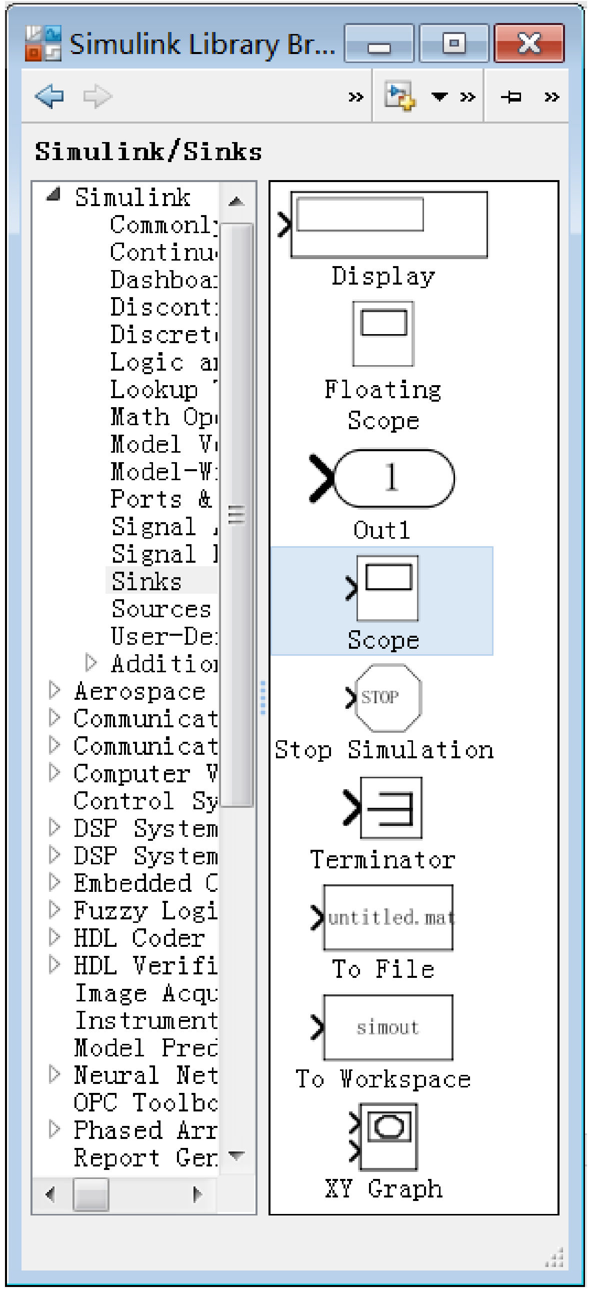


状态空间表达式的解

❑ 利用Simulink搭建结构模型

优点：模型框图化、丰富求解器、多种结果显示





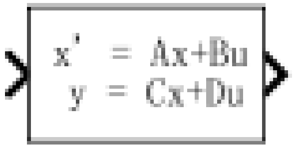
状态空间表达式的解

例2-4：求解系统

✓ 单位阶跃响应

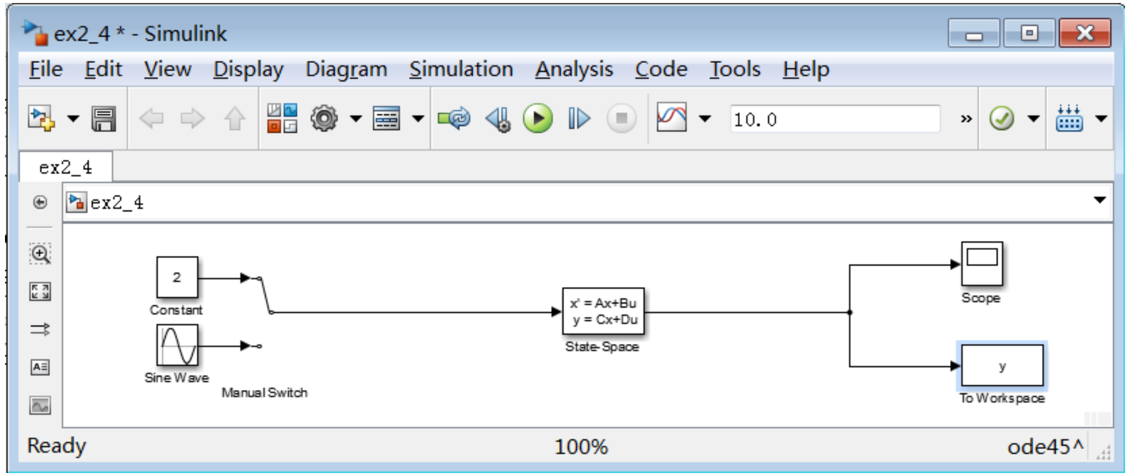
✓ 零输入响应

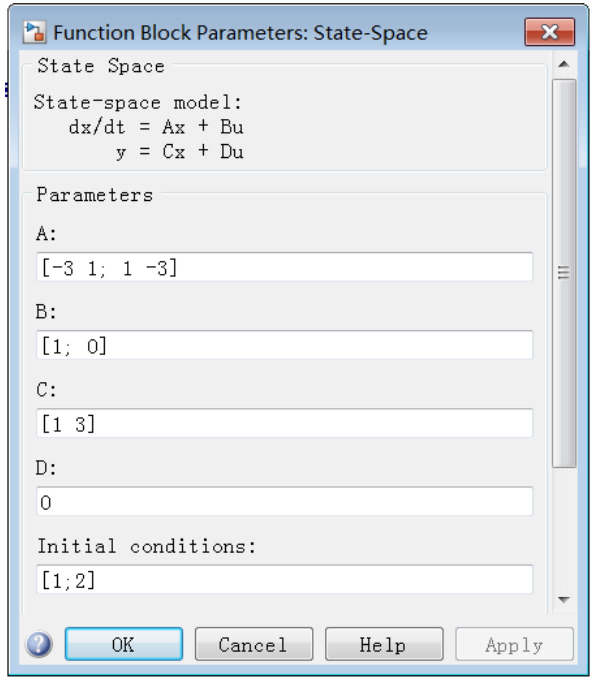
✓ 零状态响应

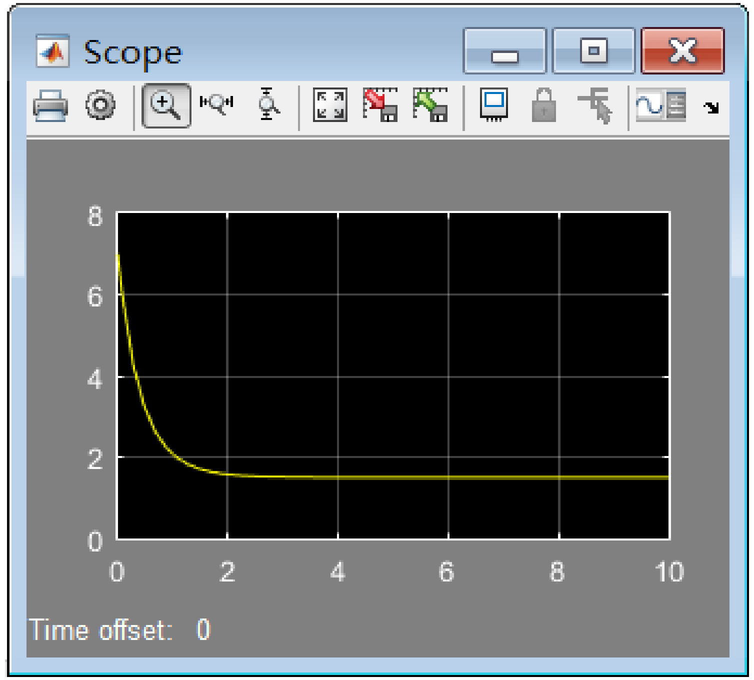


State-Space

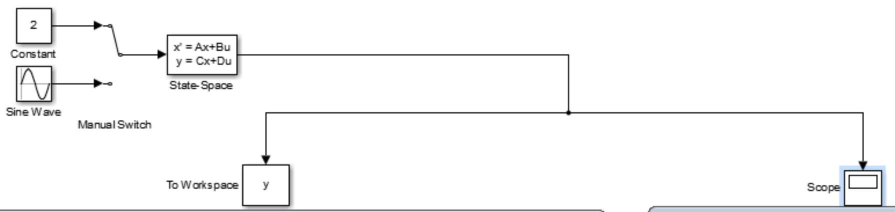
$$
\begin{cases}
\dot{x}(t)=\begin{bmatrix}-3 & 1\\ 1 & -3\end{bmatrix}x(t)+\begin{bmatrix}1\\0\end{bmatrix}u(t)\\
y(t)=[1 \quad 3]x(t)\\
1.\ u(t)=1\\
2.\ x(0)=\begin{bmatrix}1\\2\end{bmatrix}, u(t)=0\\
3.\ x(0)=\begin{bmatrix}0\\0\end{bmatrix}, u(t)=2\\
4.\ x(0)=\begin{bmatrix}1\\2\end{bmatrix}, u(t)=\sin(t)
\end{cases}
$$

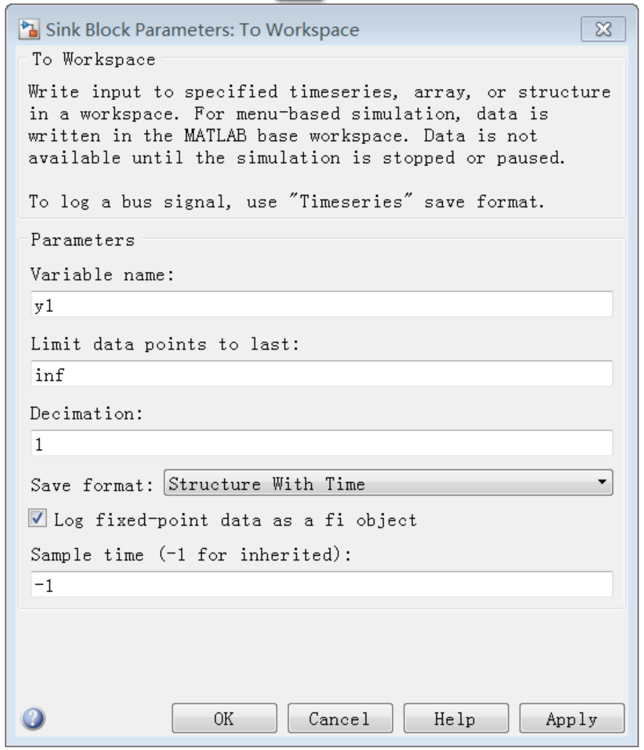


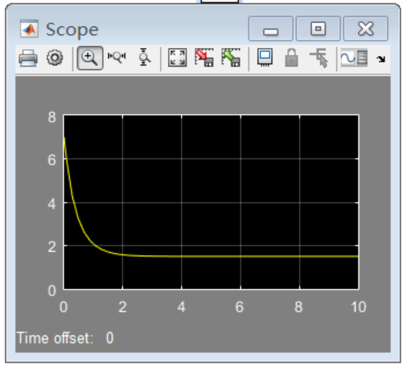


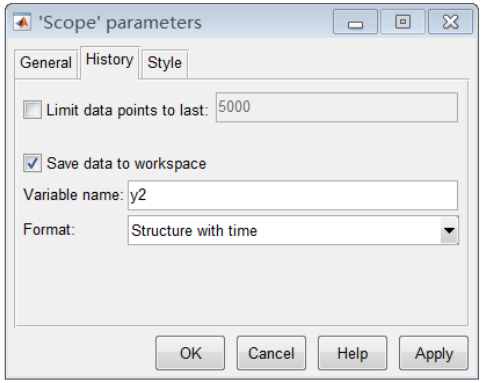


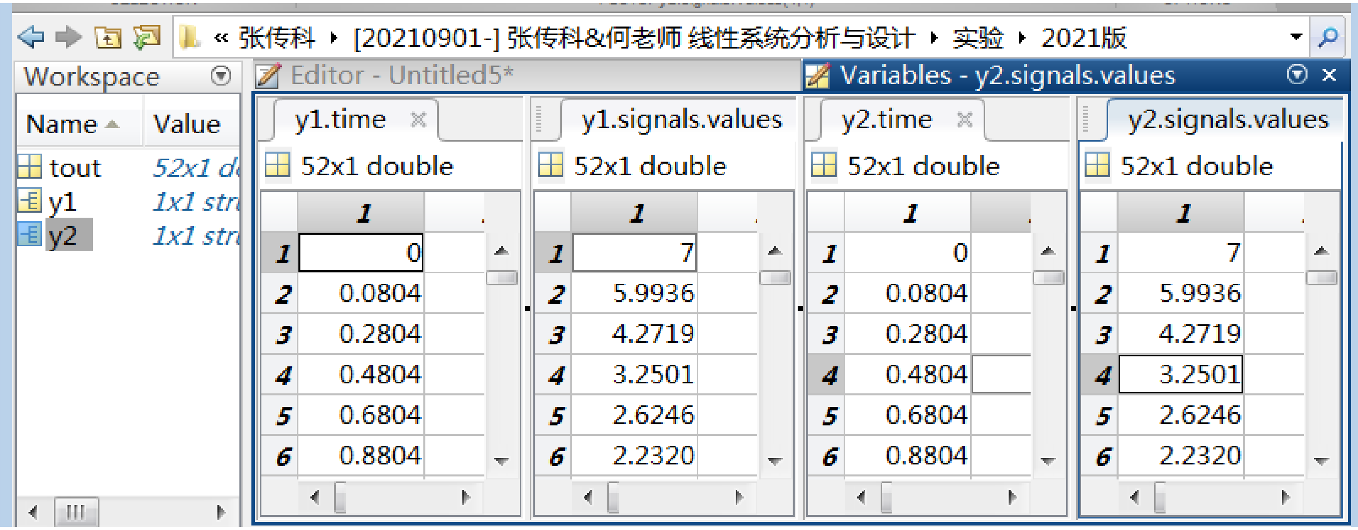
状态空间表达式的解





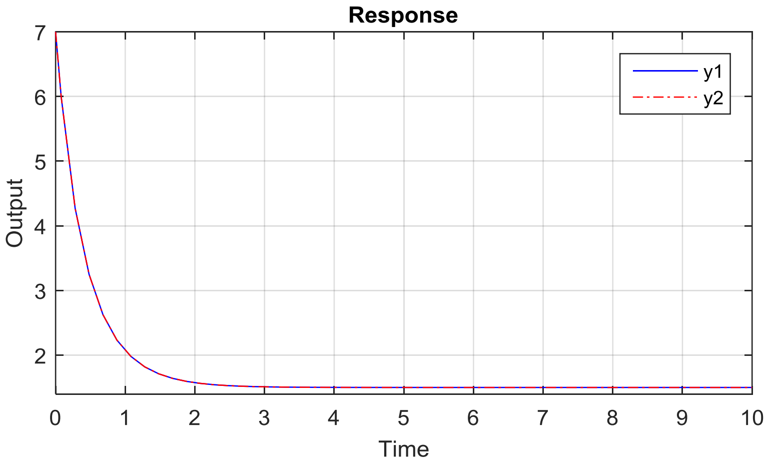






	y1.time	y1.signals.values	y2.time	y2.signals.values
1	0	7	0	7
2	0.0804	5.9936	0.0804	5.9936
3	0.2804	4.2719	0.2804	4.2719
4	0.4804	3.2501	0.4804	3.2501
5	0.6804	2.6246	0.6804	2.6246
6	0.8804	2.2320	0.8804	2.2320

```
close all

figure(1)
set(gca,'FontSize',12,'FontName','Times New Roman')
plot(y1.time,y1.signals.values(:,1),'b-',...
    y2.time,y2.signals.values(:,1),'r-.')
xlabel('Time')
ylabel('Output')
title('Response')
axis([0 10 1.4 7])
grid on
legend('y1','y2')
```



课内实验：分析部分

2.1 状态空间表达式的解（系统响应曲线绘制）

2.2 线性定常系统的能控性/能观性判定及结构分解

2.3 线性定常系统的稳定性判定

线性定常系统的能控性和能观性判定及结构分解

❑ 秩判据

$$\mathrm{rank}\left(\left[B, AB, \cdots, A^{n-1}B\right]\right) = n\,?$$

ctrb(A,B)

Yes → 能控，可化为能控标准型

No → 不完全能控，可按能控型分解

ctrbf(A,B)

$$\mathrm{rank}\left(\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}\right) = n\,?$$

obsv(A,C)

Yes → 能观，可化为能观标准型

No → 不完全能观，可按能观性分解

obsvf(A,B)

线性定常系统的能控性和能观性判定及结构分解

❑ 能控性、能观性判定

$$\mathrm{rank}\left(\left[B, AB, \cdots, A^{n-1}B\right]\right) = n\,?$$

$$\mathrm{rank}\left(\left[C^T, (CA)^T, \cdots, \left(CA^{n-1}\right)^T\right]^T\right) = n\,?$$

ctrb()函数、obsv()函数

功能：求能控性/能观性矩阵

格式：

Qc = ctrb(A,B)

Qo = obsv(A,C)

例2-5：判断能控能观性

$$\begin{cases} \dot{\boldsymbol{x}} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 1 & 2 \\ 0 & 0 \\ 2 & 1 \end{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 2 & 0 & 4 & 2 \end{bmatrix} \boldsymbol{x} \end{cases}$$

Command Window

```
ans =

系统能控


ans =

系统能观

>>
```

Editor - ex2_5.m

ex2_3.m ex2_5.m

```
%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-07
%%% Example 2-5
%%% 能控能观判断
%%%%%%%%%%%%%%%%
clc
clear

A = [4 1 0 0; 0 4 1 0; 0 0 4 1; 0 0 0 4];
B = [0 0; 1 2; 0 0; 0 2];
C = [1 0 2 0; 2 0 4 2];

n = size(A,1);
Qc = ctrb(A,B);
Qo = obsv(A,C);

if rank(Qc)==n
    str = '系统能控'
else
    str = '系统不完全能控'
end

if rank(Qo)==n
    str = '系统能观'
else
    str = '系统不完全能观'
end
```

线性定常系统的能控性和能观性判定及结构分解

❑ 按能控性分解

$$\mathrm{rank}\left(\left[B, AB, \cdots, A^{n-1}B\right]\right) = m < n$$

ctrbf()函数

功能：按能控性分解系统

格式：

[Ab,Bb,Cb,T,k] = ctrbf(A,B,C)

$$\begin{bmatrix}x_c\\ x_{uc}\end{bmatrix} \xleftarrow{\substack{x_c\in R^m\\ \text{教材}}} x \xrightarrow{\substack{x_c\in R^m\\ \text{MATLAB}}} \begin{bmatrix}x_{uc}\\ x_c\end{bmatrix}$$

ctrbf

Compute controllability staircase form

Syntax

```
[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)
ctrbf(A,B,C,tol)
```

Description

If the controllability matrix of (A, B) has rank $r \le n$, where n is the size of A, then there exists a similarity transformation such that

$$\bar{A} = TAT^T,\quad \bar{B} = TB,\quad \bar{C} = CT^T$$

where T is unitary, and the transformed system has a *staircase* form, in which the uncontrollable modes, if there are any, are in the upper left corner.

$$\boxed{\bar{A} = \begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix},\quad \bar{B} = \begin{bmatrix} 0 \\ B_c \end{bmatrix},\quad \bar{C} = [C_{nc}\, C_c]}$$

where (A_c, B_c) is controllable, all eigenvalues of A_{uc} are uncontrollable, and $C_c(sI - A_c)^{-1}B_c = C(sI - A)^{-1}B$.

`[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)` decomposes the state-space system represented by `A`, `B`, and `C` into the controllability staircase form, `Abar`, `Bbar`, and `Cbar`, described above. `T` is the similarity transformation matrix and `k` is a vector of length n, where n is the order of the system represented by `A`. Each entry of `k` represents the number of controllable states factored out during each step of the transformation matrix calculation. The number of nonzero elements in `k` indicates how many iterations were necessary to calculate `T`, and `sum(k)` is the number of states in A_c, the controllable portion of `Abar`.

`ctrbf(A,B,C,tol)` uses the tolerance `tol` when calculating the controllable/uncontrollable subspaces. When the tolerance is not specified, it defaults to `10*n*norm(A,1)*eps`.

Open Help Browser

F1 to toggle focus; Escape to close

线性定常系统的能控性和能观性判定及结构分解

❑ 按能观性分解

$$\operatorname{rank}\left(\left[C^T,(CA)^T,\cdots,\left(CA^{n-1}\right)^T\right]^T\right)=m<n$$

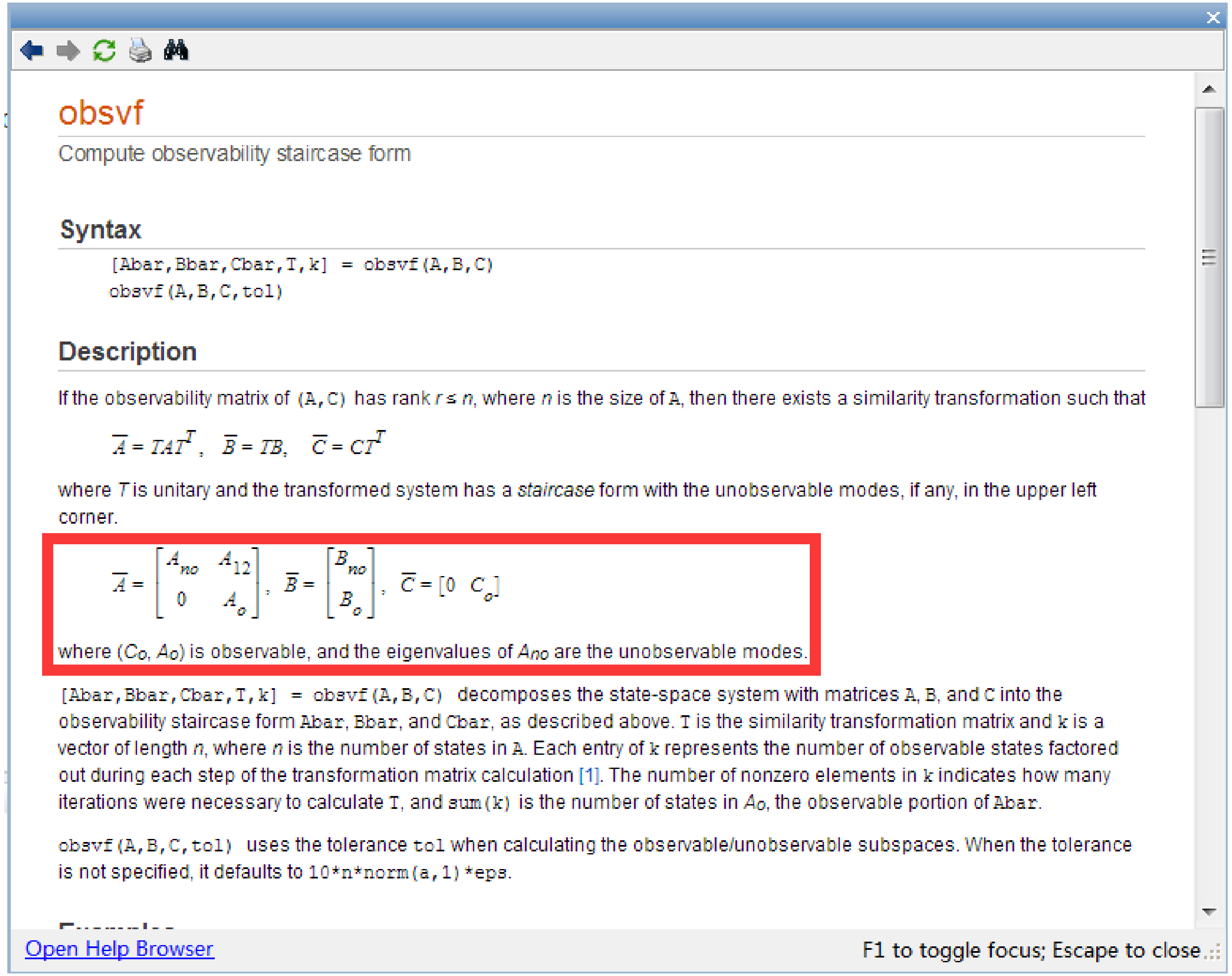
obsvf()函数

功能：按能观性分解系统

格式：

[Ab,Bb,Cb,T,k] = obvsf(A,B,C)

$$\begin{bmatrix} x_o \\ x_{uo} \end{bmatrix} \xleftarrow[\text{教材}]{x_o\in R^m} x \xrightarrow[\text{MATLAB}]{x_o\in R^m} \begin{bmatrix} x_{uo} \\ x_o \end{bmatrix}$$



obsvf

Compute observability staircase form

Syntax

```
[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)
obsvf(A,B,C,tol)
```

Description

If the observability matrix of (A,C) has rank $r \le n$, where n is the size of A, then there exists a similarity transformation such that

$$\bar{A}=TAT^T,\quad \bar{B}=TB,\quad \bar{C}=CT^T$$

where T is unitary and the transformed system has a *staircase* form with the unobservable modes, if any, in the upper left corner.

$$\bar{A}=\begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix},\quad \bar{B}=\begin{bmatrix} B_{no} \\ B_o \end{bmatrix},\quad \bar{C}=\begin{bmatrix} 0 & C_o \end{bmatrix}$$

where (C_o, A_o) is observable, and the eigenvalues of A_{no} are the unobservable modes.

[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C) decomposes the state-space system with matrices A, B, and C into the observability staircase form Abar, Bbar, and Cbar, as described above. T is the similarity transformation matrix and k is a vector of length n, where n is the number of states in A. Each entry of k represents the number of observable states factored out during each step of the transformation matrix calculation [1]. The number of nonzero elements in k indicates how many iterations were necessary to calculate T, and sum(k) is the number of states in A_o, the observable portion of Abar.

obsvf(A,B,C,tol) uses the tolerance tol when calculating the observable/unobservable subspaces. When the tolerance is not specified, it defaults to 10*n*norm(a,1)*eps.

Open Help Browser

F1 to toggle focus; Escape to close

线性定常系统的能控性和能观性判定及结构分解

例2-6：判断能控能观性，若否，则结构分解

$$\begin{cases} \dot{x} = \begin{bmatrix} 3 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 1 & 2 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 2 \end{bmatrix} x \end{cases}$$

Command Window

```
系统不完全能控

A1 =

    4.0000   -0.0000    0.0000    0.0000
   -0.0000    3.0000   -0.2298   -0.9732
   -0.9732    0.0000    3.9472   -0.2236
    0.2298    0.0000   -0.2236    3.0528

B1 =

    0.0000    0.0000
   -0.0000   -0.0000
    0.7435   -0.4595
   -1.2030   -1.9465

C1 =

   -1.0000    1.0000    0.0000    0.0000
   -2.0000    2.0000   -0.2298   -0.9732
```

Command Window

```
系统不完全能观

A2 =

    4.0000    0.7071    0.1958   -0.6795
         0    3.5000    0.8179   -0.2846
         0    0.1385    2.8502   -0.1873
         0   -0.4804    0.5199    3.6498

B2 =

    1.0000         0
    0.0000    0.0000
   -0.9609   -1.9218
   -0.2769   -0.5538

C2 =

         0    0.0000    0.3916   -1.3589
         0    0.0000   -0.1777   -2.9947
```

Editor - ex2_6.m

ex2_3.m　ex2_5.m　ex2_6.m

```matlab
clc
clear

A = [3 1 0 0; 0 3 0 0; 0 0 4 1; 0 0 0 4];
B = [0 0; 1 2; 1 0; 0 0];
C = [1 0 0 1; 2 1 0 2];

n = size(A,1);
Qc = ctrb(A,B);
Qo = obsv(A,C);

if rank(Qc)==n
    '系统能控'
else
    '系统不完全能控'
    [A1,B1,C1,T,k] = ctrbf(A,B,C)
end

if rank(Qo)==n
    '系统能观'
else
    '系统不完全能观'
    [A2,B2,C2,T,k] = obsvf(A,B,C)
end
```

思考：如何提取能控/观子系统、不能控/观子系统？
　　　如何实现按能控和能观分解？

课内实验：分析部分

2.1 状态空间表达式的解（系统响应曲线绘制）

2.2 线性定常系统的能控性/能观性判定及结构分解

2.3 线性定常系统的稳定性判定

线性定常系统的稳定性判定

- 李氏判据（间接法、直接法）

间接法：

$$\mathrm{Re}\left(\lambda_i\right) < 0?$$

$$\forall \lambda_i \in \left\{\lambda \mid \det(\lambda I - A) = 0\right\}$$

eig(A)

Yes → 系统渐近稳定

No → 系统不稳定

直接法：
（存在递减的正能量）

$$\dot{x}(t) = Ax(t)$$

$$V(x) = x^T(t)Px(t)$$

$$\dot{V}(x) = x^T(t)\left(A^T P + PA\right)x(t)$$

$$= x^T(t)\left(-Q\right)x(t)$$

先选正能量(P=I)，再判断能量导数(Q vs 0?)

$$-Q = A^T P + PA$$

先确定能量递减(Q=I)，再判断能量(P vs 0?)

$$A^T P + PA = -Q \longrightarrow P$$ **P = lyap(A,Q)**

一步到位找递减的正能量

$$P > 0,\ A^T P + PA < 0$$ **LMI（LMI toolbox、YALMIP)**

线性定常系统的稳定性判定

❑ 间接法

$$\mathrm{Re}\left(\lambda_i\right) < 0\,?$$

$$\forall \lambda_i \in \left\{\lambda \mid \det(\lambda I - A) = 0\right\}$$

例3-1：判断如下系统的稳定性

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} -3 & -6 & -2 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

eig()函数

功能：求矩阵特征根

格式：

Lambda = eig(A)

思考：如何用程序实现输出最后结论？

Editor - C:\Users\hp\Desktop\张传科\[20210901-] 张

ex2_3.m ex2_5.m ex3_1.m +

```
%%%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-10
%%% Example 3-1
%%% 稳定性判定，间接法
%%%%%%%%%%%%%%%%%%
clc
clear

A = [-3 -6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];

Lembda = eig(A)

```

Command Window

```
Lembda =

  -1.3544 + 1.7825i
  -1.3544 - 1.7825i
  -0.1456 + 0.4223i
  -0.1456 - 0.4223i
```

线性定常系统的稳定性判定

❑ 直接法：正能量是否衰减？

选P=I(正能量)

$$-Q = A^T P + PA$$

判Q > 0?(能量衰减？)

矩阵运算、定号判断

例3-2：判断如下系统的稳定性

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} -3 & -6 & -2 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Editor - C:\Users\hp\Desktop\张传科\[20210901-] 引

ex3_1.m　ex3_2.m

```matlab
%%%%%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-10
%%% Example 3-2
%%% 稳定性判定，直接法
%%%%%%%%%%%%%%%%%%%%
clc
clear

A = [-3 -6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];

P = eye(size(A,1));
Q = -P*A-A'*P;

det1 = det(Q(1,1));
det2 = det(Q(1:1,2:2));
det3 = det(Q(1:1,3:3));
det4 = det(Q);

Det = [det1; det2; det3; det4];

if min(Det) > 0
    '系统稳定'
end
```

Command Window

```
ans =

系统稳定

fx >>
```

思考：还可以怎么判断矩阵定号？

线性定常系统的稳定性判定

- 直接法：衰减能量是否为正？

选 $Q=-A^TP-PA=I$ **(能量衰减)**

算P

判P > 0?(正能量？)

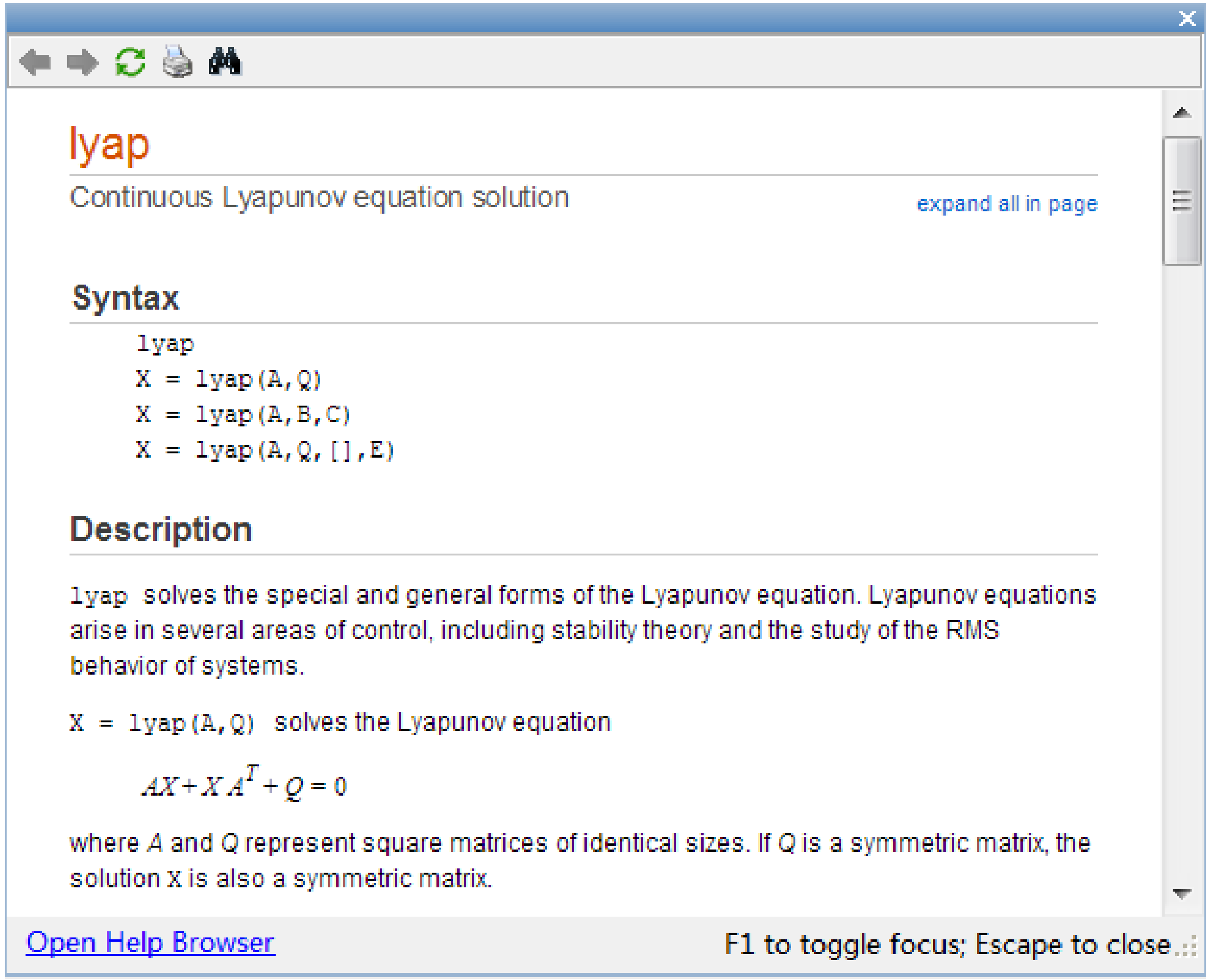
lyap()函数

功能：求李亚普洛夫方程

格式：

P = lyap(A,Q)

其中，A为系统矩阵



lyap

Continuous Lyapunov equation solution

expand all in page

Syntax

```
lyap
X = lyap(A,Q)
X = lyap(A,B,C)
X = lyap(A,Q,[],E)
```

Description

`lyap` solves the special and general forms of the Lyapunov equation. Lyapunov equations arise in several areas of control, including stability theory and the study of the RMS behavior of systems.

`X = lyap(A,Q)` solves the Lyapunov equation

$$AX+XA^T+Q=0$$

where *A* and *Q* represent square matrices of identical sizes. If *Q* is a symmetric matrix, the solution `X` is also a symmetric matrix.

Open Help Browser

F1 to toggle focus; Escape to close

线性定常系统的稳定性判定

例3-3：判断如下系统的稳定性

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix}=\begin{bmatrix}-3 & -6 & -2 & -1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}$$

Command Window

```
ans =

系统稳定

fx >>
```

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix}=\begin{bmatrix}-3 & 6 & -2 & -1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}$$

Command Window

```
ans =

系统不稳定

fx >>
```

Editor - C:\Users\hp\Desktop\张传科\[20210901-]

ex3_1.m ex3_2.m ex3_3.m

```
%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2021-10-10
%%% Example 3-3
%%% 稳定性判定，直接法
%%%%%%%%%%%%%%%
clc
clear

A = [-3 -6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];
A = [-3 6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];

Q = eye(size(A,1));
P = lyap(A,Q);

det1 = det(P(1,1));
det2 = det(P(1:1,2:2));
det3 = det(P(1:1,3:3));
det4 = det(P);

Det = [det1; det2; det3; det4];

if min(Det) > 0
    '系统稳定'
else
    '系统不稳定'
end
```

思考：else部分稳定成立？

线性定常系统的稳定性判定

- 直接法：正的衰减能量可直接找到？

$$\begin{cases} P > 0 \Rightarrow \text{正能量} \\ A^T P + PA < 0 \Rightarrow \text{能量衰减} \end{cases}$$

LMI toolbox，MATLAB自带
YALMIP toolbox，需加载

功能：求解线性矩阵不等式

LMI toolbox 自学

线性定常系统的稳定性判定

例3-4：判断如下系统的稳定性

$$\dot{x}(t)=\begin{bmatrix}1 & 2 & 4\\ 1 & 1 & 1\\ 0 & 2 & 1\end{bmatrix}x(t)$$

$$P>0,\ \ A^T P+PA<0$$

```
P = sdpvar(n,n,'symmetric');
% P为n维*n维对称矩阵
Q = sdpvar(n,m,'full');
% Q为n维*m维矩阵
```

思考：code中其它代码的意义、为什么有if else部分成立？

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Chuan-Ke Zhang
%%% 2018-01-19
%%% 验证系统的稳定性程序
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear

%%%  系统参数
A = [1 2 4; 1 1 1; 0 2 1];

%%% 描述待求的LMI
P = sdpvar(3,3,'symmetric'); % 给出待求矩阵
Fcond = [P>0,A'*P+P*A<0]; % 列出所有待求LMI

%%% 求解LMI
ops = sdpsettings('verbose',0,'solver','sedumi'); % 设置求解环境
diagnostics = solvesdp(Fcond,[],ops); % 迭代求解
[m p] = checkset(Fcond); % 返回求解结果
tmin = min(m); % 验证是否满足

if tmin > 0
    disp('System is stable') % 结论输出
else
    disp('System is unstable')  % 结论输出
end
```

本部分实验小结

- MATLAB使用
  - 如何绘制曲线（获得仿真数据、画图）
  - 如何加载新的toolbox，使用YALMIP求解LMI

- 系统分析
  - 如何用MATLAB获取状态空间模型的解
  - 如何用MATLAB判断系统的能控性、能观性和结构分解
  - 如何用MATLAB判断系统的稳定性

本部分实验练习题

- 作业一：
  - 选择两组初值[-0.1,0.1,0.2][1,2,3]
  - 绘制如下系统的系统响应曲线
  - 绘制如下系统的状态轨迹

the following representation of Chua's circuit systems:

$$\begin{cases} \dot{x}_1(t) = a\left[x_2(t) - h\left(x_1(t)\right)\right] \\ \dot{x}_2(t) = x_1(t) - x_2(t) + x_3(t) \\ \dot{x}_3(t) = -bx_2(t) \\ p(t) = x_1(t) \end{cases} \tag{23}$$

with the nonlinear characteristics of Chua's diode

$$h(x) = m_1 x_1(t) + \frac{1}{2}(m_0 - m_1)\left(|x_1(t) + c| - |x_1(t) - c|\right) \tag{24}$$

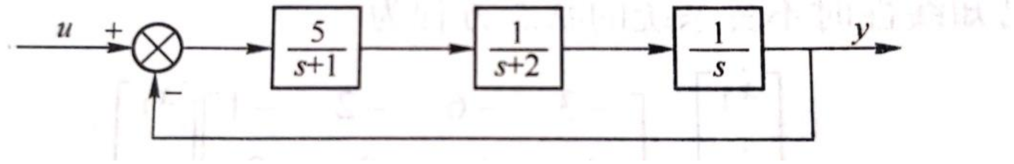
and parameters $a = 9$, $b = 14.28$, $c = 1$, $m_0 = -(1/7)$, $m_1 = 2/7$, and $c = 1$.

- 判断如下系统的能控能观性，若不完全能控且不完全能观，求其能控能观子系统

$$\begin{cases} \dot{x} = \begin{bmatrix} -4 & 1 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 1 & 3 \\ 5 & 7 \\ 4 & 3 \\ 0 & 0 \\ 1 & 6 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 3 & 1 & 0 & 5 & 0 & 0 \\ 1 & 4 & 0 & 2 & 0 & 0 \end{bmatrix} x \end{cases}$$

本部分实验练习题

- 作业一：
  - 判断如下系统的稳定性



- 作业二：
  - 尝试利用MATLAB完成教材第2章习题，2-3、2-4、2-6（至少完成1题）
  - 尝试利用MATLAB完成教材第3章习题，3-7、3-8、3-11、3-12、3-13（至少完成1题）
  - 尝试利用MATLAB完成教材第4章习题，4-3（至少完成1题）

谢谢！