

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

➤ 利用MATLAB内嵌函数

➤ 利用Simulink搭建的模型（重点）

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

ode23

Solve nonstiff differential equations; low order method

dsolve

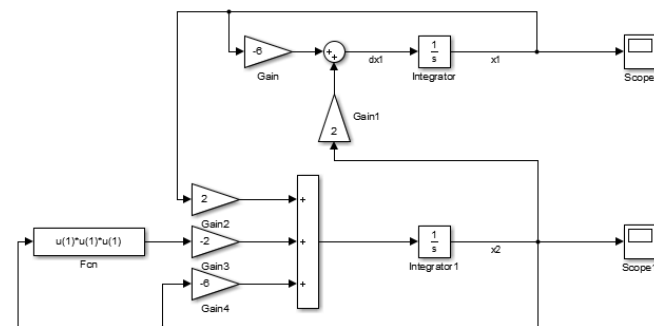
Ordinary differential equation and system solver

deval

Evaluate solution of differential equation problem

See Also

[bvp4c](#) | [bvp5c](#) | [dde23](#) | [ddensd](#) | [ddeas](#) | [ode113](#) | [ode15i](#) |
[ode15s](#) | [ode23](#) | [ode23s](#) | [ode23t](#) | [ode23tb](#) | [ode45](#)



```
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

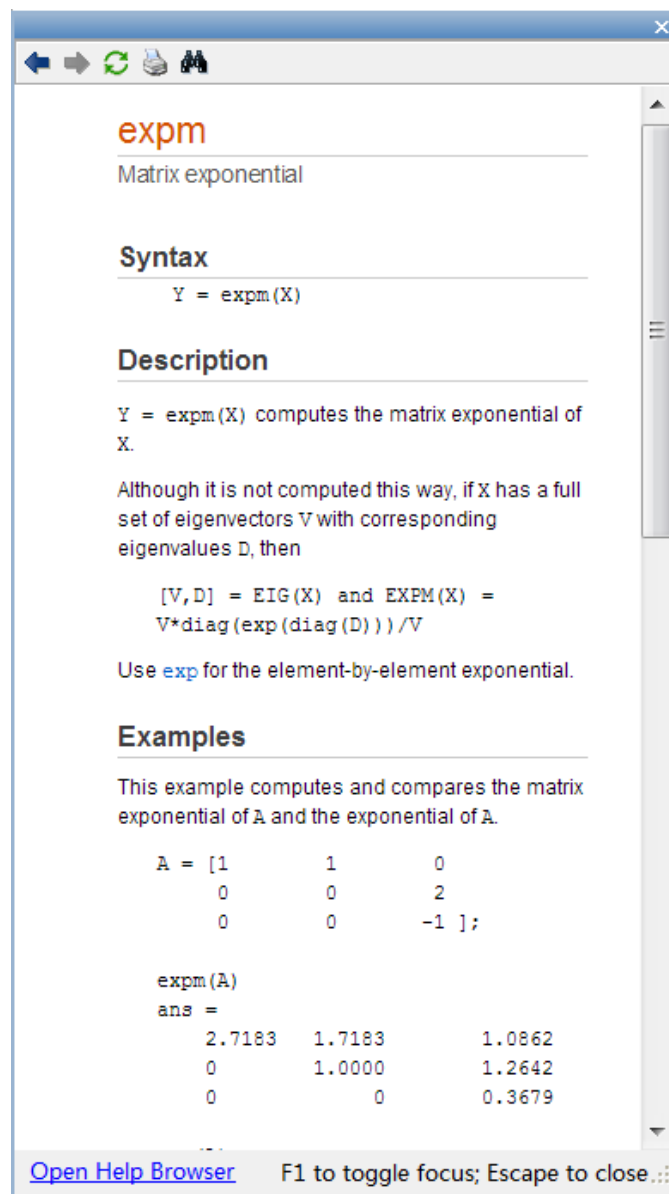
$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) + \int_0^t \Phi(t-\tau)\mathbf{B}u(\tau)d\tau$$

expm()函数、ilaplace()函数
(复习教材知识用，实际不这么求解)

功能：计算状态转移矩阵
格式：

$$\mathbf{e}^{At} = \text{expm}(\mathbf{A} * t)$$
$$\mathbf{e}^{At} = \text{ilaplace}(\mathbf{F}\mathbf{S}, s, t)$$

其中，s，t为定义的符号变量；
FS为预解矩阵 $\mathbf{F}\mathbf{S} = (s\mathbf{I} - \mathbf{A})^{-1}$



expm
Matrix exponential

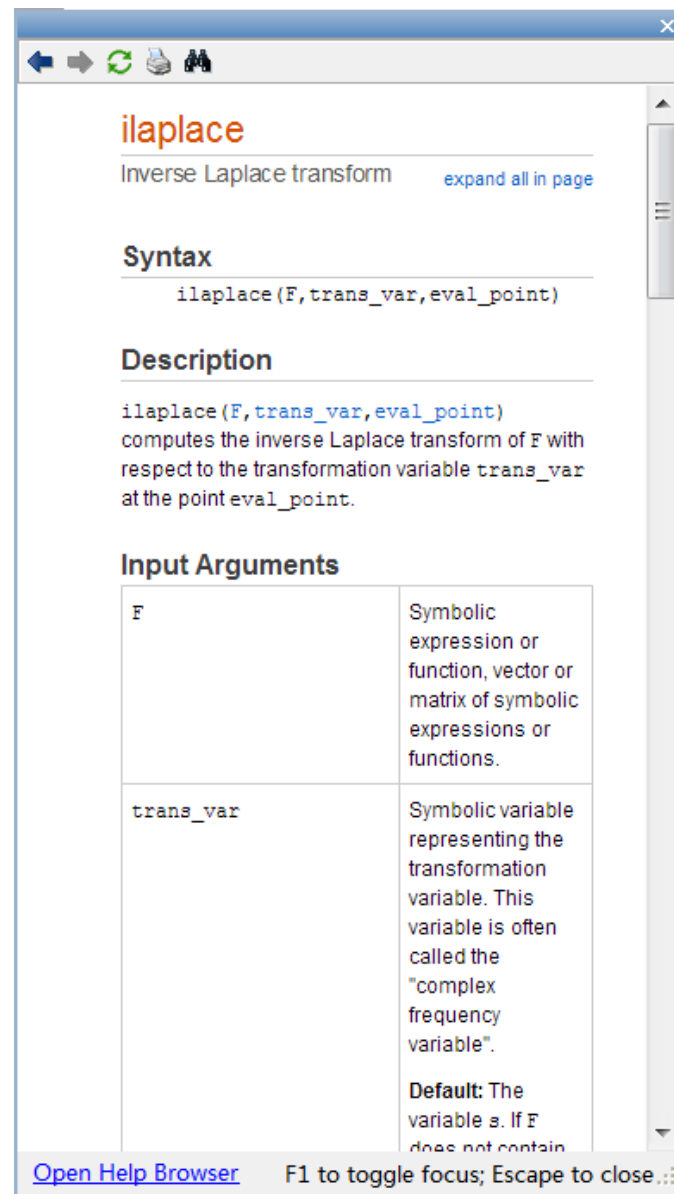
Syntax
`Y = expm(X)`

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

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

$$x(t) = \Phi(t)x(0)$$

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

思考：绘制零状态响应？
(选做，是不是比较麻烦)

```
Editor - ex2_1.m
ex2_1.m  x +
1      %%%%%%%%%%%
2      %% Chuan-Ke Zhang
3      %% 2021-10-07
4      %% Example 2-1
5      %% 计算状态转移矩阵，绘制零输入响应
6      %%%%%%%%%%%
7
8      clc
9      clear all
10
11     A = [-3 1; 1 -3];
12
13     % 计算状态转移矩阵
14     syms s t          % MATLAB符号运算，定义符号
15     eAt1 = expm(A*t)  % 方法1
16     FS = inv(s*eye(2)-A); % 预解矩阵
17     eAt2 = ilaplace(FS, s, t) % 方法2
18
19     % 绘制零输入响应
20     x0 = [1; 2];      % 定义初值
21     Time = 0:0.01:10; % 定义时间区间和间隔，即曲线横坐标数据
22     xTime = [];       % 存放求得的xt，即曲线纵坐标数据
23     for t = Time
24         eAt1 = expm(A*t);
25         xt = eAt1*x0;
26         xTime = [xTime xt];
27     end
28     plot(Time, xTime) % 绘制响应曲线
29
```

```
Command Window

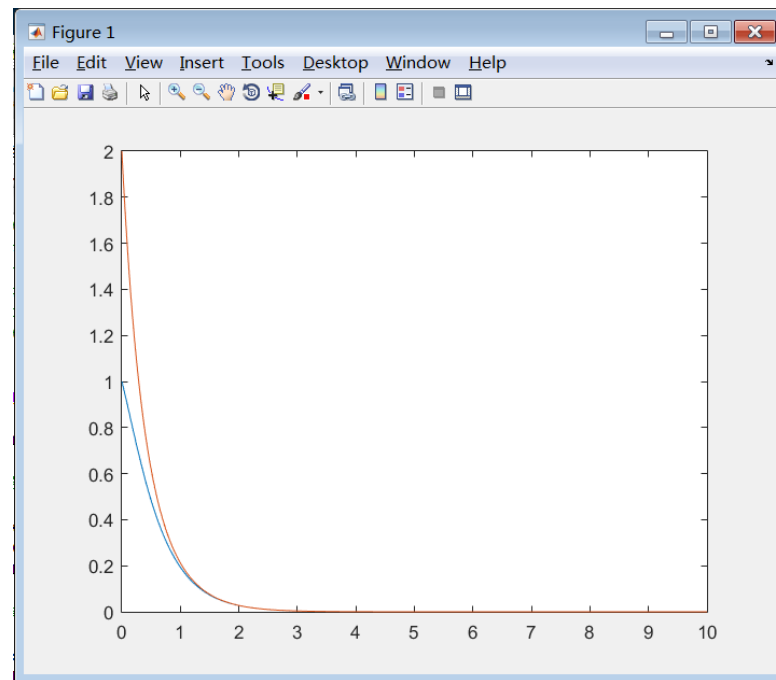
eAt1 =

[ 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

step()函数，initial()函数，lsim()函数
(了解即可，功能太单一了)

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

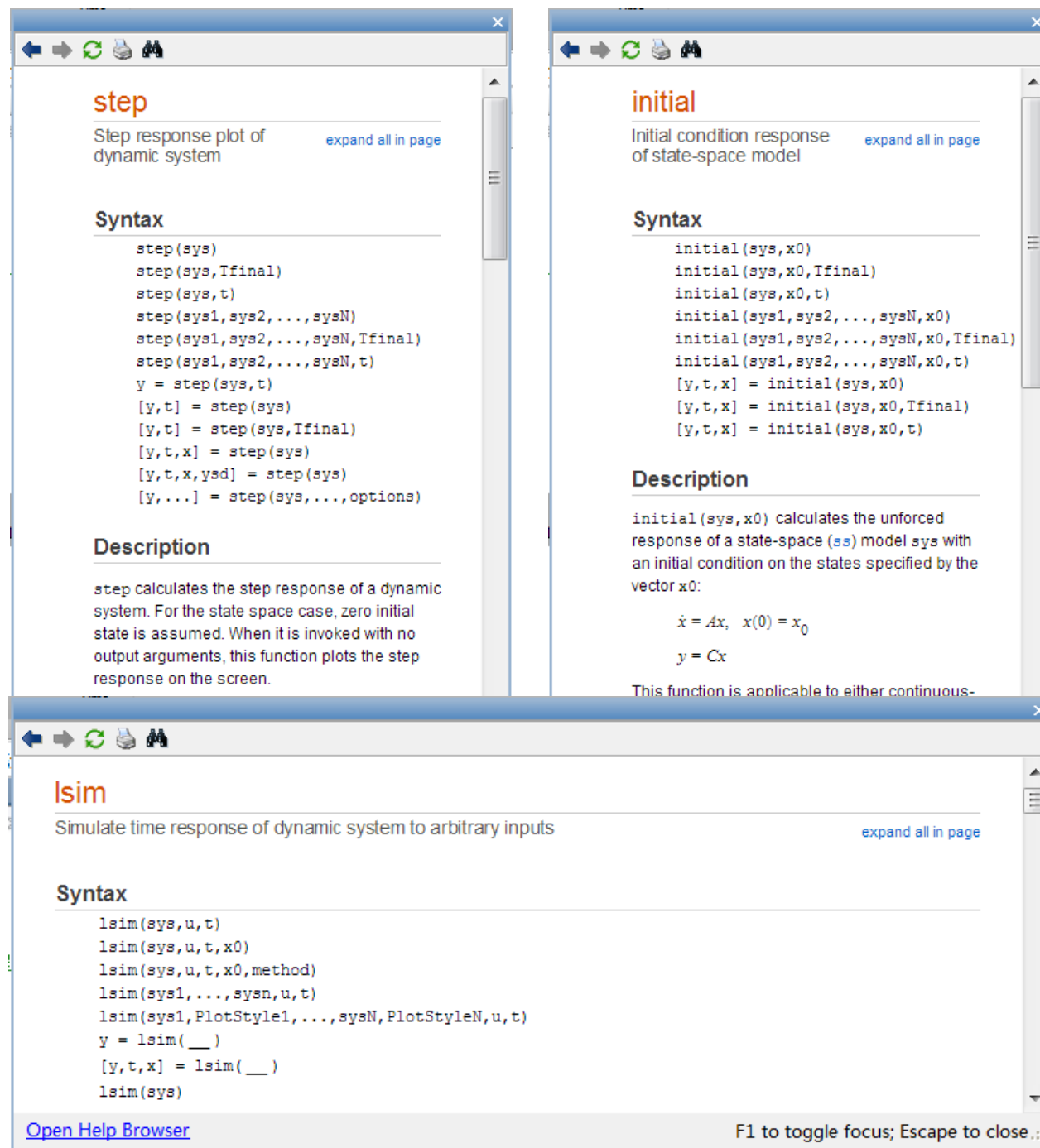
格式：

`step(A,B,C,D)`

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

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

其中， x_0 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

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

$$y(t) = [1 \quad 3]x(t)$$

1. $u(t) = 1$

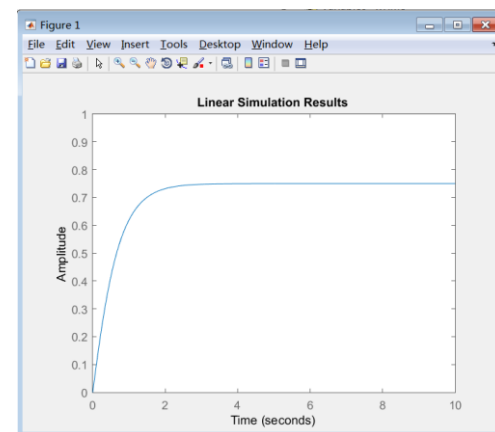
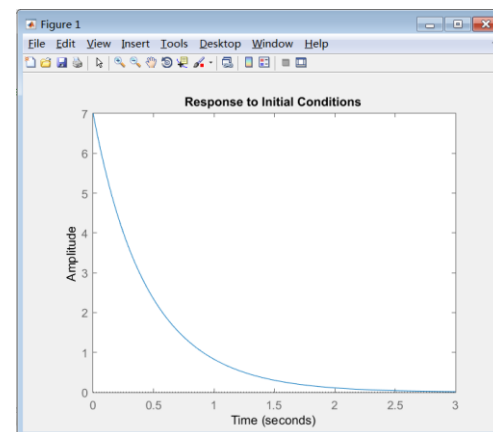
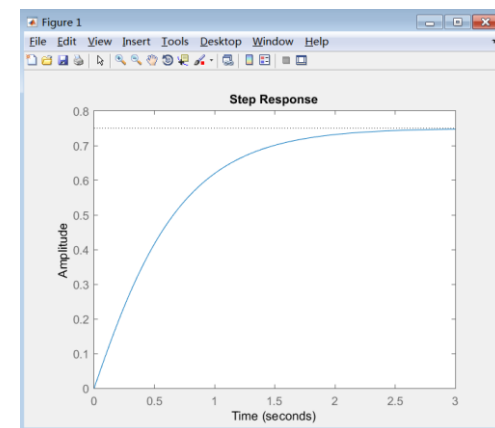
2. $x(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, u(t) = 0$

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

思考：全响应（选做）

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  +
1  %%%%%%%%%%%%%%
2  %% Chuan-Ke Zhang
3  %% 2021-10-07
4  %% Example 2-3
5  %% 求解LTI系统的单位阶跃响应、零状态响应、零输入响应
6  %%%%%%%%%%%%%%
7
8  clc
9  clear all
10
11  A = [-3 1; 1 -3];
12  B = [1; 0];
13  C = [1 3];
14  D = 0;
15
16  % 单位阶跃响应
17  step(A,B,C,D);
18
19  % 零状态响应
20  x0 = [1; 2];
21  initial(A,B,C,D,x0)
22
23  % 零状态响应
24  x0 = [0; 0];
25  t = 0:0.01:10;
26  u = ones(1,size(t,2));
27  lsim(A,B,C,D,u,t,x0)
```



状态空间表达式的解

□ 利用MATLAB内嵌函数

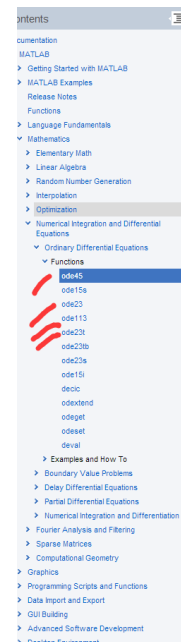
$$\begin{cases} \dot{x}(t) = f(x, u) \\ u(t) = \text{具体形式} \\ x(0) = \text{具体数值} \end{cases}$$

ode系列函数

功能：求解微分方程

格式：求解命令 + 函数定义

```
- [t,y] = solver(@func,time,initial):  
% solver 求解器, 可选 ode23, ode45, 等  
% func 待求解的方程  
% time 仿真时间区间, e.g. [1,200]  
% initial 初始条件, [x10, x20, ..., ]  
% [t,y] 返回数值, 一一对应的数据
```



Search Documentation

MATLAB > Mathematics > Numerical Integration and Differential Equations > Ordinary Differential Equations

ode45

Solve nonstiff differential equations; medium order method

[expand all in page](#)

Syntax

```
[T,Y] = solver(odefun,tspan,y0)  
[T,Y] = solver(odefun,tspan,y0,options)  
[T,Y,TE,YE,IE] = solver(odefun,tspan,y0,options)  
sol = solver(odefun,[t0 tf],y0,...)
```

This page contains an overview of the solver functions: ode23, ode45, ode113, ode15s, ode23s, ode23t, and ode23tc. You can call any of these solvers by substituting the placeholder, `solvrz`, with any of the function names.

Arguments

The following table describes the input arguments to the solvers.

odefun	A function handle that evaluates the right side of the differential equations. All solvers solve systems of equations in the form $y' = f(t,y)$ or problems that involve a mass matrix, $M(t)y' = f(t,y)$. The ode23s solver can solve only equations with constant mass matrices. ode15s and ode23t can solve problems with a mass matrix that is singular, i.e., differential-algebraic equations (DAEs).
tspan	A vector specifying the interval of integration, $[t_0, t_f]$. The solver imposes the initial conditions at $tspan(1)$, and integrates from $tspan(1)$ to $tspan(end)$. To obtain solutions at specific times (all increasing or all decreasing), use <code>tspan = [t0,t1,...,tf]</code> . For <code>tspan</code> vectors with two elements $[t_0, t_f]$, the solver returns the solution evaluated at every integration step. For <code>tspan</code> vectors with more than two elements, the solver returns solutions evaluated at the given time points. The time values must be in order, either increasing or decreasing.
	Specifying <code>tspan</code> with more than two elements does not affect the internal time steps that the solver uses to traverse the interval from $tspan(1)$ to $tspan(end)$. All solvers in the ODE suite obtain output values by means of continuous extensions of the basic formulas. Although a solver does not necessarily step precisely to a time point specified in <code>tspan</code> , the solutions produced at the specified time points are of the same order of accuracy as the solutions computed at the internal time points.
	Specifying <code>tspan</code> with more than two elements has little effect on the efficiency of computation, but for large systems, affects memory management.
y0	A vector of initial conditions.
options	Structure of optional parameters that change the default integration properties. This is the fourth input argument. <code>[t,y] = solver(odefun,tspan,y0,options)</code> You can create options using the <code>odeset</code> function. See odeset for details.

The following table lists the output arguments for the solvers.

T	Column vector of time points.
Y	Solution array. Each row in Y corresponds to the solution at a time returned in the corresponding row of T.
TE	The time at which an event occurs.
YE	The solution at the time of the event.
IE	The index i of the event function that vanishes.

```
- function dx = func(t,x) % 定义要求的方程  
    dx = f(x)  
% 例如 dx = [-3 1; 1 -3]*x; % 定义状态方程 dot x =A x
```

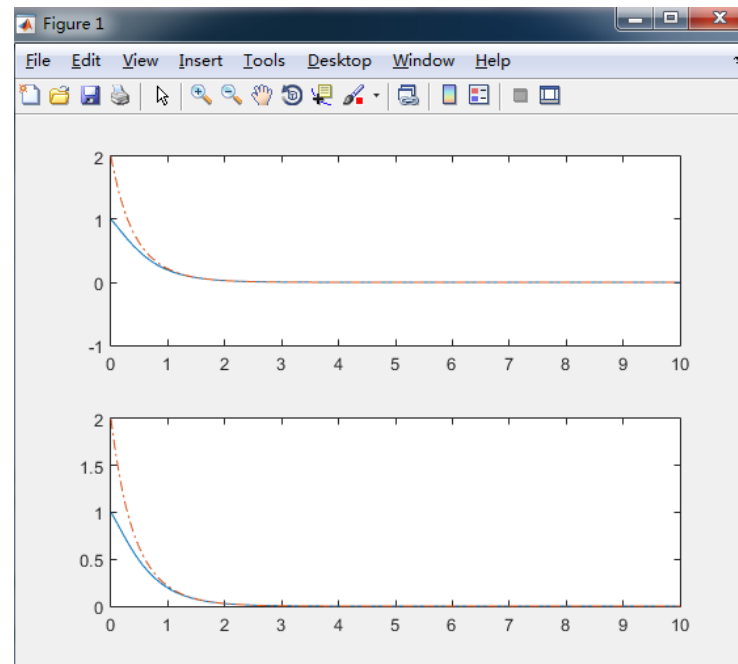
状态空间表达式的解

例2-2：求解微分方程，并绘制零输入曲线

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

```
%%%%%%%%%%%%%%  
%%% Chuan-Ke Zhang  
%%% Example 2-2  
%%% 求齐次常微分方程，绘制零输入响应  
%%%%%%%%%%%%%%
```

```
function main  
  
[I1,Y1] = ode23(@func, [0 10], [1 2]); % 求解方程 func，时间范围[0,10]，初始条件x0 = [1; 2]  
[I2,Y2] = ode45(@func, [0 10], [1 2]); % 求解方程 func，时间范围[0,10]，初始条件x0 = [1; 2]  
figure(1)  
subplot(2,1,1)  
plot(I1,Y1(:,1),'-r', I1,Y1(:,2),'-b') % 利用数据绘制曲线  
subplot(2,1,2)  
plot(I2,Y2(:,1),'-r', I2,Y2(:,2),'-b') % 利用数据绘制曲线
```



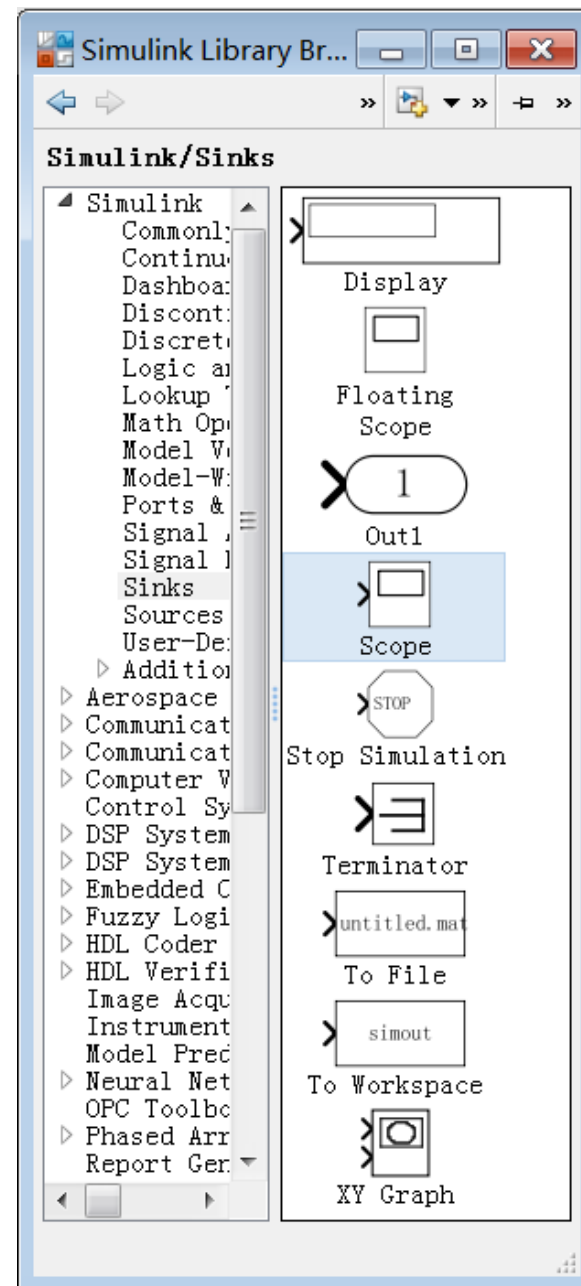
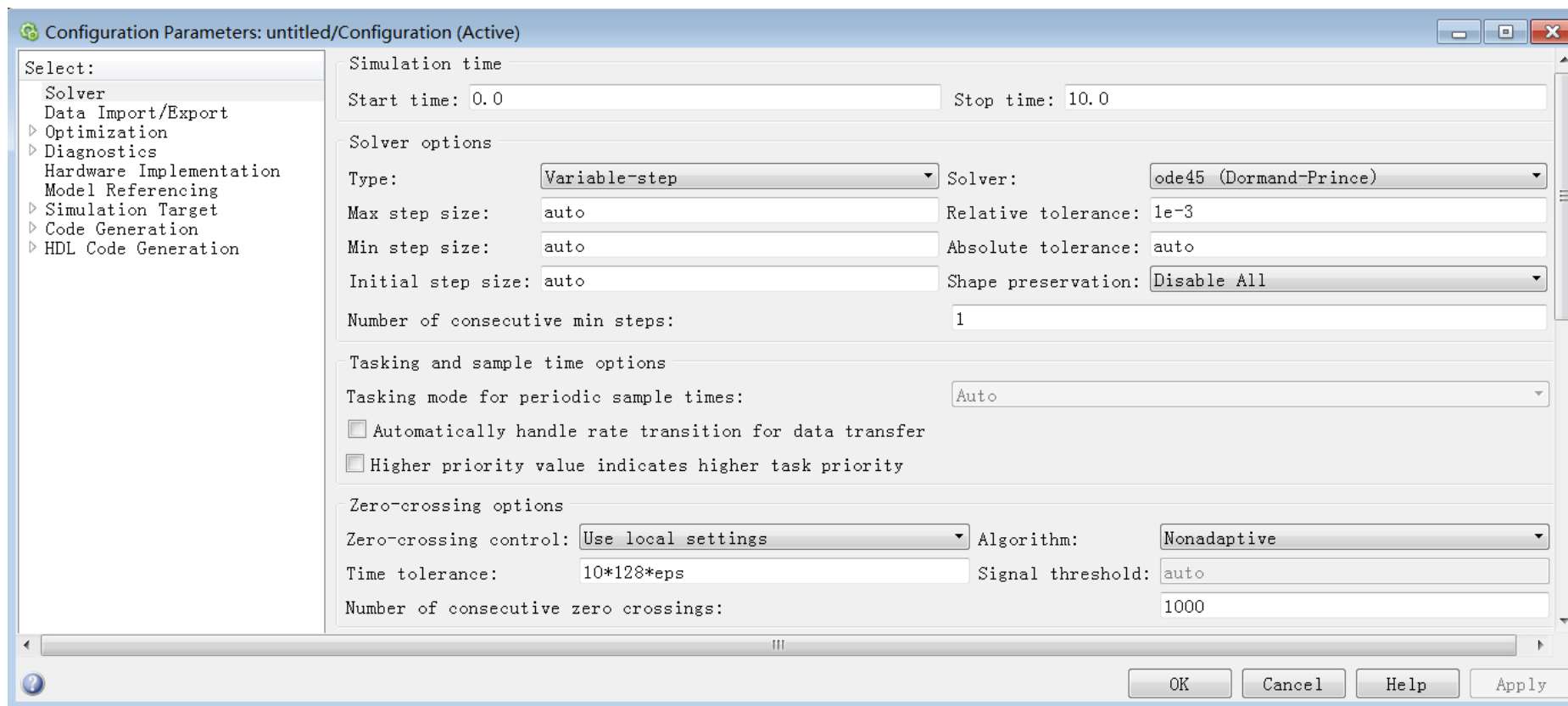
思考：零状态响应？

```
function dx = func(t,x) % 定义要求的方程  
  
dx = [-3 1; 1 -3]*x; % 定义状态方程
```

状态空间表达式的解

□ 利用 Simulink 搭建结构模型（重点）

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

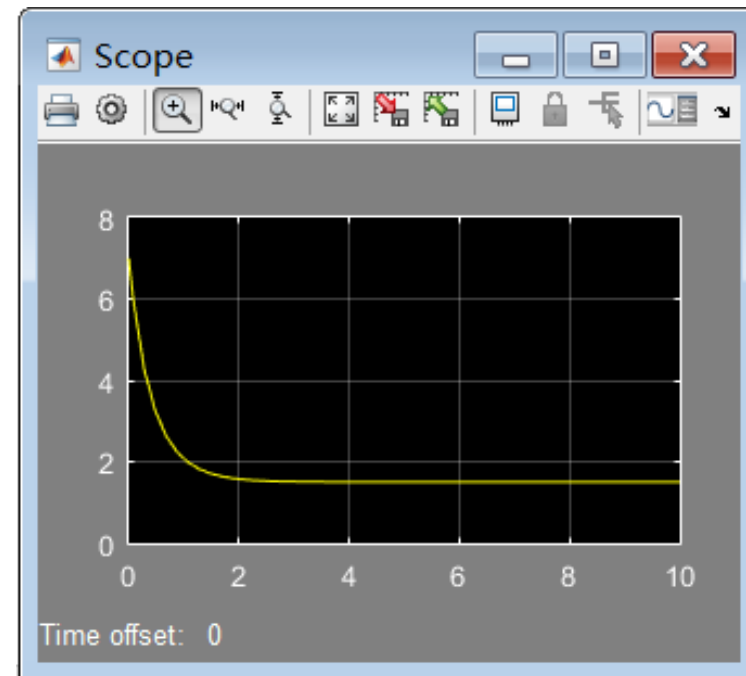
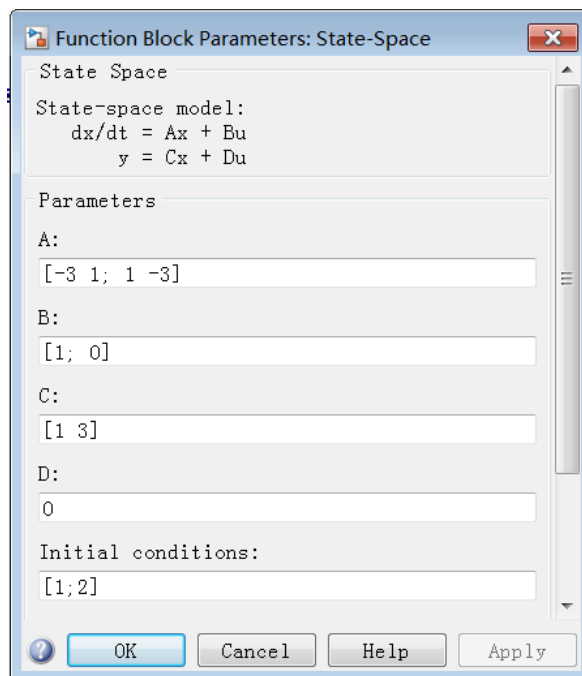
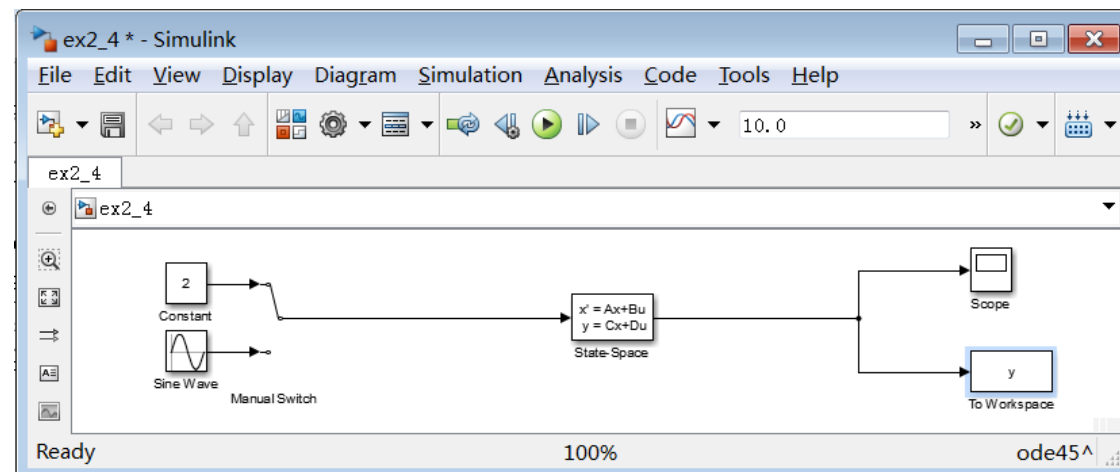
- ✓ 单位阶跃响应
- ✓ 零输入响应
- ✓ 零状态响应

$$\begin{cases} \dot{x}' = Ax + Bu \\ y = Cx + Du \end{cases}$$

State-Space

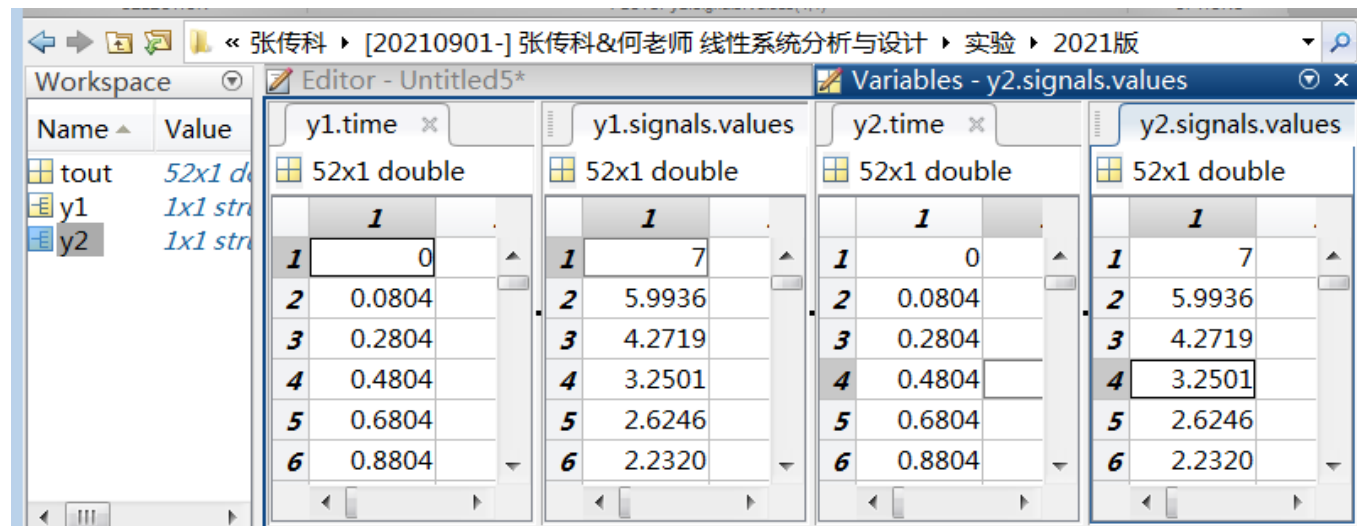
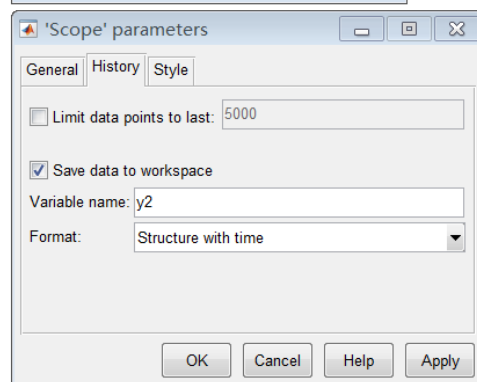
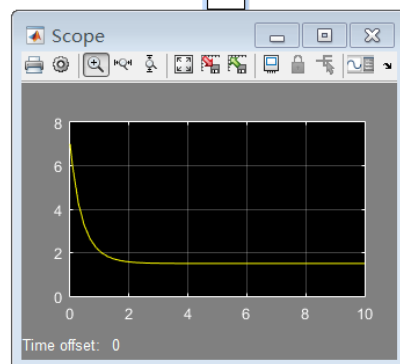
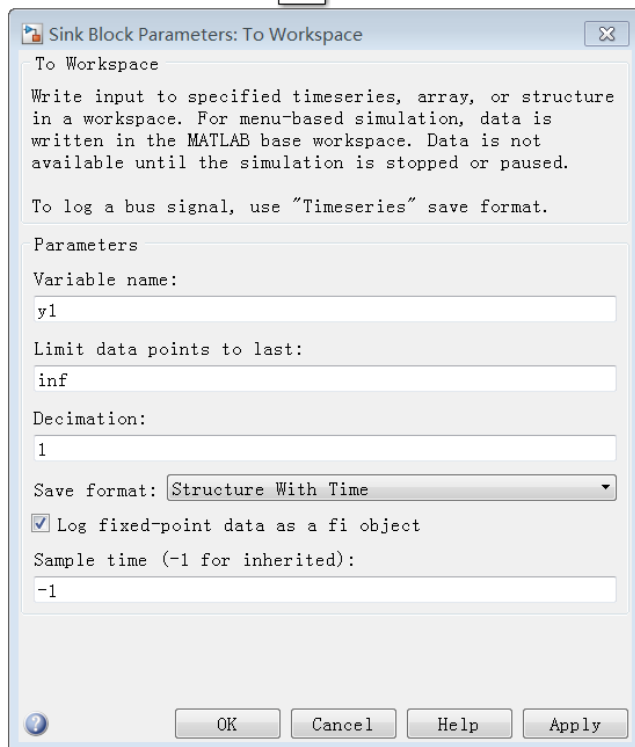
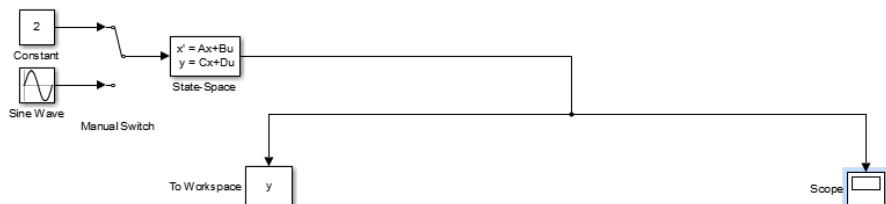
$$\begin{cases} \dot{x}(t) = \begin{bmatrix} -3 & 1 \\ 1 & -3 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \\ y(t) = [1 \quad 3]x(t) \end{cases}$$

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



状态空间表达式的解

怎么把simulink仿真数据导入工作空间



```
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')
```

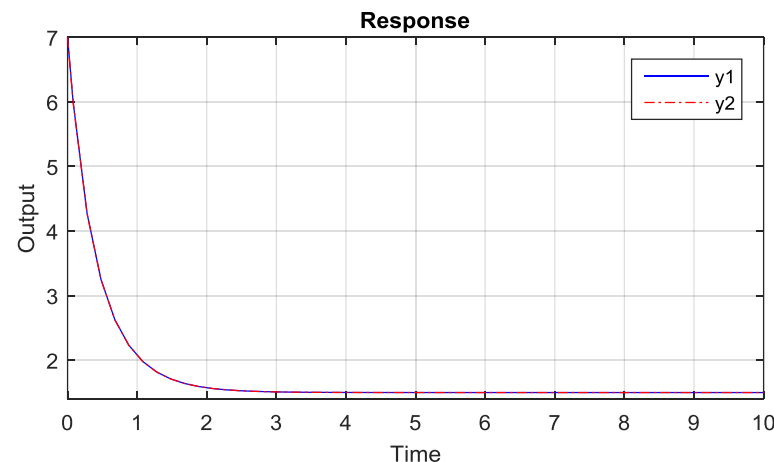
```
ylabel('Output')
```

```
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

$$\text{rank} \left(\begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \right) = n ?$$

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)

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

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

$$\text{rank}\left(\begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}\right) = n?$$

$$\text{rank}\left(\begin{bmatrix} C^T & (CA)^T & \cdots & (CA^{n-1})^T \end{bmatrix}^T\right) = n?$$

ctrb()函数、obsv()函数

功能：求能控性/能观性矩阵

格式：

$$Q_c = \text{ctrb}(A, B)$$

$$Q_o = \text{obsv}(A, C)$$

例2-5：判断能控能观性

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

Command Window

ans =

系统能控

ans =

系统能观

fx >>

```
Editor - ex2_5.m
ex2_3.m x ex2_5.m x +
1 %%%%%%%%%%%%%%
2 %%% Chuan-Ke Zhang
3 %%% 2021-10-07
4 %%% Example 2-5
5 %%% 能控能观判断
6 %%%%%%%%%%%%%%
7 -
8 - clear
9
10 - A = [4 1 0 0; 0 4 1 0; 0 0 4 1; 0 0 0 4];
11 - B = [0 0; 1 2; 0 0; 0 2];
12 - C = [1 0 2 0; 2 0 4 2];
13
14 - n = size(A,1);
15 - Qc = ctrb(A,B);
16 - Qo = obsv(A,C);
17
18 - if rank(Qc)==n
19 -     str = '系统能控'
20 - else
21 -     str = '系统不完全能控'
22 - end
23
24 - if rank(Qo)==n
25 -     str = '系统能观'
26 - else
27 -     str = '系统不完全能观'
28 - end
```

线性定常系统的能控性和能观性判定及结构分解

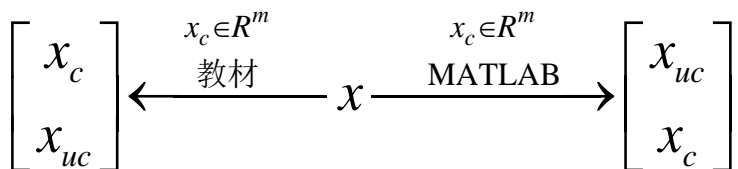
□ 按能控性分解

$$\text{rank}\left(\begin{bmatrix} B, AB, \dots, A^{n-1}B \end{bmatrix}\right) = m < n$$

ctrbf()函数

功能：按能控性分解系统
格式：

$$[Ab, Bb, Cb, T, k] = \text{ctrbf}(A, B, C)$$



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 $r \leq n$, 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.

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

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

$[Abar, Bbar, Cbar, T, k] = \text{ctrbf}(A, B, C)$ 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 $\text{sum}(k)$ is the number of states in A_c , the controllable portion of $Abar$.

$\text{ctrbf}(A, B, C, \text{tol})$ uses the tolerance tol when calculating the controllable/uncontrollable subspaces. When the tolerance is not specified, it defaults to $10 \cdot n \cdot \text{norm}(A, 1) \cdot \text{eps}$.

[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

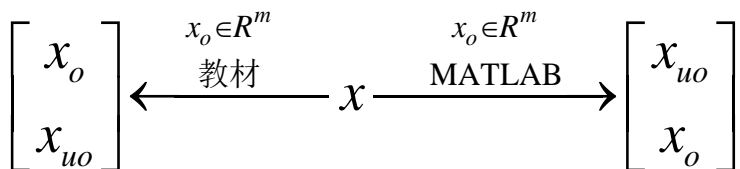
□ 按能观性分解

$$\text{rank} \left(\begin{bmatrix} C^T, (CA)^T, \dots, (CA^{n-1})^T \end{bmatrix}^T \right) = m < n$$

obsvf()函数

功能：按能观性分解系统
格式：

$$[Ab, Bb, Cb, T, k] = \text{obsvf}(A, B, C)$$



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 $r \leq n$, 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.

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

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

$[Abar, Bbar, Cbar, T, k] = \text{obsvf}(A, B, C)$ 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 $\text{sum}(k)$ is the number of states in A_o , the observable portion of $Abar$.

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

[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

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

Command Window	Command Window
系统不完全能控	系统不完全能观
A1 =	A2 =
4.0000 -0.0000 0.0000 0.0000	4.0000 0.7071 0.1958 -0.6795
-0.0000 3.0000 -0.2298 -0.9732	0 3.5000 0.8179 -0.2846
-0.9732 0.0000 3.9472 -0.2236	0 0.1385 2.8502 -0.1873
0.2298 0.0000 -0.2236 3.0528	0 -0.4804 0.5199 3.6498
B1 =	B2 =
0.0000 0.0000	1.0000 0
-0.0000 -0.0000	0.0000 0.0000
0.7435 -0.4595	-0.9609 -1.9218
-1.2030 -1.9465	-0.2769 -0.5538
C1 =	C2 =
-1.0000 1.0000 0.0000 0.0000	0 0.0000 0.3916 -1.3589
-2.0000 2.0000 -0.2298 -0.9732	0 0.0000 -0.1777 -2.9947

```
Editor - ex2_6.m
ex2_3.m x ex2_5.m x ex2_6.m x +
7 - clc
8 - clear
9
10 - A = [3 1 0 0; 0 3 0 0; 0 0 4 1; 0 0 0 4];
11 - B = [0 0; 1 2; 1 0; 0 0];
12 - C = [1 0 0 1; 2 1 0 2];
13
14 - n = size(A,1);
15 - Qc = ctrb(A,B);
16 - Qo = obsv(A,C);
17
18 - if rank(Qc)==n
19 -     '系统能控'
20 - else
21 -     '系统不完全能控'
22 -     [A1,B1,C1,I,k] = ctrbf(A,B,C)
23 - end
24
25 - if rank(Qo)==n
26 -     '系统能观'
27 - else
28 -     '系统不完全能观'
29 -     [A2,B2,C2,I,k] = obsvf(A,B,C)
30 - 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:

$$\begin{aligned} & \text{Re}(\lambda_i) < 0? \\ & \forall \lambda_i \in \{\lambda \mid \det(\lambda I - A) = 0\} \\ & \text{eig}(A) \end{aligned}$$

Yes



系统渐近稳定

No



系统不稳定

直接法:

（存在递减的正能量）

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

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

$$\begin{aligned} \dot{V}(x) &= x^T(t)(A^T P + PA)x(t) \\ &= x^T(t)(-Q)x(t) \end{aligned}$$

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?)

$$A^T P + PA = -Q \longrightarrow P \quad \mathbf{P = lyap(A, Q)}$$

一步到位找递减的正能量

$$P > 0, A^T P + PA < 0 \quad \mathbf{LMI \text{ (LMI toolbox、YALMIP)}}$$

线性定常系统的稳定性判定

□ 间接法

$$\operatorname{Re}(\lambda_i) < 0?$$
$$\forall \lambda_i \in \{\lambda \mid \det(\lambda I - A) = 0\}$$

eig()函数

功能：求矩阵特征根

格式：

$$\text{Lambda} = \text{eig}(A)$$

思考：如何用程序实现
输出最后结论？

例3-1：判断如下系统的稳定性

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

```
Editor - C:\Users\hp\Desktop\张传科\20210901-张
ex2_3.m x ex2_5.m x ex3_1.m x +
1 %%%%%%%%%%%
2 %%% Chuan-Ke Zhang
3 %%% 2021-10-10
4 %%% Example 3-1
5 %%% 稳定性判定，间接法
6 %%%%%%%%%%%
7 - clc
8 - clear
9
10 - A = [-3 -6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];
11
12 - Lambda = eig(A)
13
```

```
Command Window

Lambda =

-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-] 5
ex3_1.m x ex3_2.m x +
1 %%%%%%%%%%%
2 %%% Chuan-Ke Zhang
3 %%% 2021-10-10
4 %%% Example 3-2
5 %%% 稳定性判定，直接法
6 %%%%%%%%%%%
7 -
8 - clear
9
10 - A = [-3 -6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];
11 |
12 - P = eye(size(A,1));
13 - Q = -P*A-A'*P;
14
15 - det1 = det(Q(1,1));
16 - det2 = det(Q(1:1,2:2));
17 - det3 = det(Q(1:1,3:3));
18 - det4 = det(Q);
19
20 - Det = [det1; det2; det3; det4];
21
22 - if min(Det) > 0
23 -     '系统稳定'
24 - 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 $Q = -A^T P - PA = I$ (能量衰减)
算P

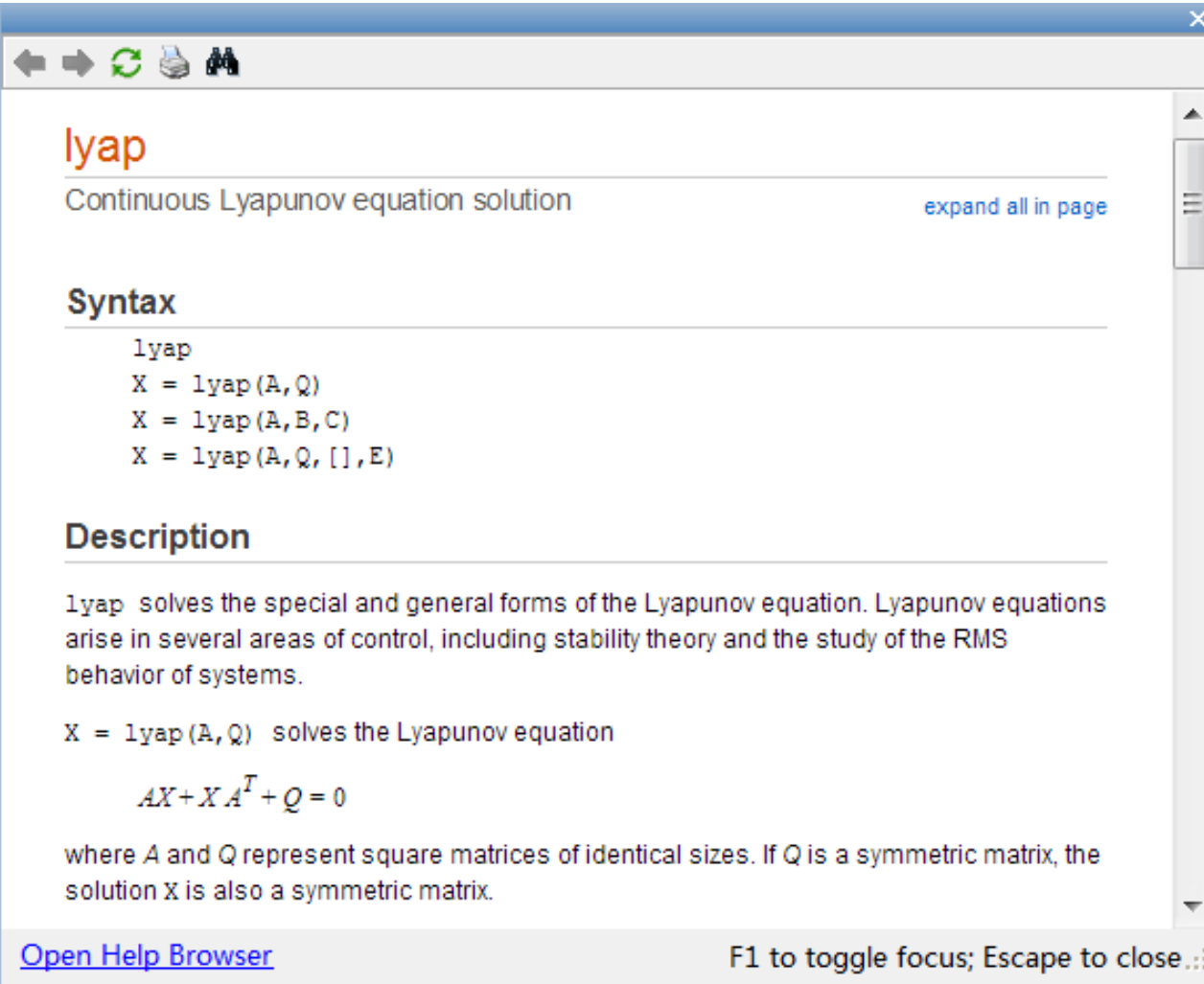
判 $P > 0$? (正能量?)

lyap()函数

功能：求李亚普洛夫方程
格式：

$$P = \text{lyap}(A, Q)$$

其中，A为系统矩阵



The image shows a MATLAB Help Browser window for the `lyap` function. The window has a title bar with standard OS controls and a toolbar with navigation icons. The main content area is titled `lyap` and `Continuous Lyapunov equation solution`. It includes a `Syntax` section with the following code snippets:

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

Below the syntax is a `Description` section. It states: "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." It then shows the equation $AX + XA^T + Q = 0$ and explains that A and Q are square matrices of identical sizes, and if Q is symmetric, the solution X is also symmetric. At the bottom, there is a link to "Open Help Browser" and a footer note: "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

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

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

Command Window

```
ans =  
  
系统稳定
```

fx >>

Command Window

```
ans =  
  
系统不稳定
```

fx >>

```
Editor - C:\Users\hp\Desktop\张传科\20210901-] 5
ex3_1.m x ex3_2.m x ex3_3.m x +
1 %%%%%%%%%%%
2 %%% Chuan-Ke Zhang
3 %%% 2021-10-10
4 %%% Example 3-3
5 %%% 稳定性判定，直接法
6 %%%%%%%%%%%
7 - clc
8 - clear
9
10 - A = [-3 -6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];
11 - A = [-3 6 -2 -1; 1 0 0 0; 0 1 0 0; 0 0 1 0];
12
13 - Q = eye(size(A,1));
14 - P = lyap(A,Q);
15
16 - det1 = det(P(1,1));
17 - det2 = det(P(1:1,2:2));
18 - det3 = det(P(1:1,3:3));
19 - det4 = det(P);
20
21 - Det = [det1; det2; det3; det4];
22
23 - if min(Det) > 0
24 -     '系统稳定'
25 - else
26 -     '系统不稳定'
27 - end
--
```

思考：else部分为什么成立？

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



Baidu search results for 'YALMIP'. The search bar shows 'YALMIP'. Below the search bar, there are tabs for '网页' (Web), '资讯' (News), '贴吧' (Tieba), and '知道' (Zhidao). The results show approximately 128,000 related results. A link to 'YALMIP' is highlighted, with a description: '查看此网页的中文翻译, 请点击 A question on the YALMIP forum squares solutions which really a https://yalmip.github.io/'. Below this, there are recommendations for 'cplex matlab' and 'matlab trace'. At the bottom, there is a link to 'Yalmip使用学习 - 简书' with the date '2018年1月23日' and the text 'yalmip学习 0.'.



Baidu search results for 'matlab 加载toolbox'. The search bar shows 'matlab 加载toolbox'. Below the search bar, there are tabs for '网页' (Web), '资讯' (News), '贴吧' (Tieba), '知道' (Zhidao), and '视频' (Video). The results show approximately 2,720,000 related results. A link to '给Matlab添加工具箱Toolbox的方法' is highlighted, with a description: '2013年11月20日 - 虽然庞大的Matlab已经有的要求,常常需要自己添加Toolbox。下面以 https://blog.csdn.net/u0127364...'. Below this, there is a link to 'Matlab添加toolbox - CongliYin的博' with the date '2017年8月10日' and the text '由于科研需要,为matlab添具包,它专门用于简化最先进的黎曼优化算法 https://blog.csdn.net/sinat_20...'. At the bottom, there is a link to 'MATLAB2016添加工具箱toolbox方' with the date '2016年11月22日' and the text '仅供参考'.

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

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

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

```
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？

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
2 %% Chuan-Ke Zhang  
3 %% 2018-01-19  
4 %% 验证系统的稳定性程序  
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
6  
7 - clc; clear  
8  
9 %% 系统参数  
10 - A = [1 2 4; 1 1 1; 0 2 1];  
11  
12 %% 描述待求的LMI  
13 - P = sdpvar(3,3,'symmetric'); % 给出待求矩阵  
14 - Fcond = [P>0,A'*P+P*A<0]; % 列出所有待求LMI  
15  
16 %% 求解LMI  
17 - ops = sdpsettings('verbose',0,'solver','sedumi'); % 设置求解环境  
18 - diagnostics = solvesdp(Fcond,[],ops); % 迭代求解  
19 - [m p] = checkset(Fcond); % 返回求解结果  
20 - tmin = min(m); % 验证是否满足  
21  
22 - if tmin > 0  
23 -     disp('System is stable') % 结论输出  
24 - else  
25 -     disp('System is unstable') % 结论输出  
26 - 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：

- 尝试利用MATLAB完成教材第2章习题，2-3、2-4、2-6（至少尝试1题）
- 尝试利用MATLAB完成教材第4章习题，4-3（至少尝试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（至少尝试1题）

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（状态响应+输出响应）曲线
- 绘制如下系统的状态轨迹

the following representation of Chua's circuit systems:

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

with the nonlinear characteristics of Chua's diode

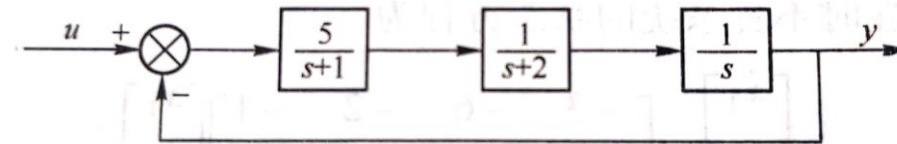
$$h(x) = m_1 x_1(t) + \frac{1}{2}(m_0 - m_1)(|x_1(t) + c| - |x_1(t) - c|) \quad (24)$$

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

本部分实验练习题

□ 作业二：（要求要做的，验收和报告包含的题）

➤ 判断如下系统的稳定性



本部分实验练习题

□ 作业二：

➤ 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

$$\left\{ \begin{array}{l} \dot{x} = \begin{bmatrix} -4 & 1 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 1 & 3 \\ 5 & 7 \\ 4 & 3 \\ 0 & 0 \\ 1 & 6 \\ 0 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 3 & 1 & 0 & 5 & 0 & 0 \\ 1 & 4 & 0 & 2 & 0 & 0 \end{bmatrix} x \end{array} \right.$$

谢谢！