第二讲

常微分方程数值求解

—— MATLAB 求解

Matlab 解初值问题函数

- 用 Maltab自带函数 解初值问题

  - 求解析解：dsolve
  - 求数值解：

    ode45、ode23、

    ode113、ode23t、ode15s、

    ode23s、ode23tb

符号求解

符号求解　　dsolve

dsolve 求解析解

- **求解析解：dsolve**

```
y=dsolve('eq1','eq2', ... ,'cond1','cond2', ... ,'v')
```

其中 y 为输出的解， eq1、eq2、... 为方程，cond1、cond2、... 为初值条件，v 为自变量。

例 1：求微分方程 $\frac{dy}{dx}+2xy=xe^{-x^2}$ 的通解，并验证。

```
sol=dsolve('Dy+2*x*y=x*exp(-x^2)','x')
```

```
syms x
diff(sol)+2*x*sol - x*exp(-x^2)  % 验证
```

dsolve 的使用

- 几点说明
  - 微分方程中用 D 表示对 自变量 的导数，如：

    Dy ⟶ y'；　D2y ⟶ y''；　D3y ⟶ y'''

  - 如果省略初值条件，则表示求通解；
  - 如果省略自变量，则默认自变量为 *t*

```
dsolve('Dy=2*x','x');      % dy/dx = 2x
dsolve('Dy=2*x');          % dy/dt = 2x
```

  - 若找不到解析解，则提出警告，并返回空解。

dsolve 的使用

- 使用符号方程
  - 导数：diff，如 diff(y)，diff(y,2)
  - 等号：==
  - 必须声明应变量与自变量!

例 1：求微分方程 $\frac{dy}{dx}+2xy=xe^{-x^2}$ 的通解，并验证。

```
syms y(x)
sol=dsolve(diff(y)+2*x*y==x*exp(-x^2))
diff(y)+2*x*y - x*exp(-x^2)
```

dsolve 举例

例 2： 求微分方程 $xy'+y-e^x=0$在初值条件 $y(1)=2e$ 下的特解，并画出解函数的图形。

```
sol=dsolve('x*Dy+y-exp(x)=0', 'y(1)=2*exp(1)', 'x');
ezplot(sol);
```

```
syms y(x)
sol=dsolve(diff(y)*x+y-exp(x)==0, y(1)==2*exp(1));
ezplot(sol);
```

```
syms y(x)
sol=dsolve(diff(y)*x+y-exp(x)==0, y(1)==2*exp(sym(1)));
ezplot(sol);
```

dsolve 举例

例3： 求微分方程组 $\begin{cases} \dfrac{dx}{dt}+5x+y=e^t \\ \dfrac{dy}{dt}-x-3y=0 \end{cases}$ 在初值条件 $\begin{cases} x|_{t=0}=1 \\ y|_{t=0}=0 \end{cases}$ 下的特解，并画出解函数的图形。

```
[x,y]=dsolve('Dx+5*x+y=exp(t)','Dy-x-3*y=0', ...
                    'x(0)=1', 'y(0)=0', 't')
ezplot(x,y,[0,1.3]);
```

注：解微分方程组时，如果所给的输出个数与方程个数相同，则方程组的解按词典顺序输出；如果只给一个输出，则输出的是一个包含解的结构（structure）类型的数据。

dsolve 举例

例：

```
[x,y]=dsolve('Dx+5*x=0', 'Dy-3*y=0', ...
             'x(0)=1',  'y(0)=1', 't')
```

```
sol = dsolve('Dx+5*x=0', 'Dy-3*y=0', ...
             'x(0)=1',  'y(0)=1', 't')
```

这里返回的 sol 是一个 结构类型 的数据

```
sol.x   % 查看解函数 x(t)
sol.y   % 查看解函数 y(t)
```

dsolve的输出个数只能为一个或与方程个数相等

dsolve 举例

- 使用符号方程

```
syms   x(t)   y(t)
sol=dsolve(diff(x)+5*x==0, diff(y)-3*y==0, ...
                  x(0)==1, y(0)==1)
```

dsolve 举例

例 4： 求微分方程 $\frac{d^2y}{dt^2} = -a^2 y$ 的特解，初值条件为

$$y(0) = 1, y'(\pi / a) = 0$$

其中 a 是符号常量

```
dsolve('D2y=-a^2*y','y(0)=1','Dy(pi/a)=0')
```

```
syms y(t)  a
dy = diff(y);
sol=dsolve(diff(y,2)==-a^2*y, y(0)==1, dy(pi/a)==0)
```

数值求解

ode45、ode23、

ode113、ode23t、ode15s、

ode23s、ode23tb

数值求解

[T, Y] = solver(odefun, tspan, y0)

其中 y0 为初值条件，tspan为求解区间；Matlab在数值求解时自动对求解区间进行分割，T (列向量) 中返回的是分割点的值(自变量)，Y (数组) 中返回的是这些分割点上的近似解，其列数等于应变量的个数。

solver 为Matlab的ODE求解器（可以是 ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb）

没有一种算法可以有效地解决所有的 ODE 问题，因此 MATLAB 提供了多种ODE求解器，对于不同的ODE，可以调用不同的求解器。

Matlab的ODE求解器

求解器	ODE类型	特点	说明
ode45	非刚性	单步法；4，5 阶 R-K 方法；累计截断误差为 $(\triangle x)^3$	大部分场合的首选方法
ode23	非刚性	单步法；2，3 阶 R-K 方法；累计截断误差为 $(\triangle x)^3$	使用于精度较低的情形
ode113	非刚性	多步法；Adams算法；高低精度均可到 $10^{-3}\sim10^{-6}$	计算时间比 ode45 短
ode23t	适度刚性	采用梯形算法	适度刚性情形
ode15s	刚性	多步法；Gear’s 反向数值微分；精度中等	若 ode45 失效时，可尝试使用
ode23s	刚性	单步法；2 阶Rosebrock 算法；低精度	当精度较低时，计算时间比 ode15s 短
ode23tb	刚性	梯形算法；低精度	当精度较低时，计算时间比ode15s短

参数说明

[T, Y] = solver(odefun,tspan,y0)

odefun 为函数句柄，代表显式常微分方程，可以通过匿名函数定义，或在函数文件中定义，然后通过函数句柄调用。

例：求 $\begin{cases} \dfrac{dy}{dx} = -2y + 2x^2 + 2x \\ y(0) = 1 \end{cases}$ 的数值解，求解区间为 [0,0.5]

```
f = @(x,y) -2*y+2*x^2+2*x;
[x,y] = ode45(f, [0,0.5],1);
```

注：也可以在 tspan 中指定对求解区间的分割，如：

```
[x,y] = ode45(f, [0:0.1:0.5],1);    % x=[0:0.1:0.5]
```

数值求解举例

如果需求解的问题是高阶常微分方程，则需将其化为一阶常微分方程组，此时必须用函数文件来定义该常微分方程组。

例： 求解 Ver der Pol 初值问题

$$\begin{cases} \dfrac{d^2y}{dt^2}-\mu(1-y^2)\dfrac{dy}{dt}+y=0 \\ \mu=7,\ \ y(0)=1,\quad y'(0)=0 \end{cases}$$

令

$$\begin{cases} x_1=y \\ x_2=\dfrac{dy}{dt} \end{cases} \Rightarrow \begin{cases} \dfrac{dx_1}{dt}=x_2 \\ \dfrac{dx_2}{dt}=\mu(1-x_1^2)x_2-x_1 \\ \mu=7 \\ x_1(0)=1,\quad x_2(0)=0, \end{cases}$$

数值求解举例

- 先编写函数文件 verderpol.m

```
function dx=verderpol(t,x) % 必须是两个输入和一个输出
global mu;  % 其它参数只能通过全局变量传递
dx=[x(2); mu*(1-x(1)^2)*x(2) - x(1)]; % 必须是列向量
```

- 然后编写脚本文件 vanderpol_main.m

```
clear;
global mu;
mu=7;
y0=[1; 0];
[t,x] = ode45(@verderpol, [0,40], y0);
plot(t, x(:,1));
```