第十讲：微分方程数值计算方法

刘保东
山东大学, 计算机科学与技术学院
Email:baodong@sdu.edu.cn

本讲主要内容

1. 一阶常微分方程初值问题
2. 一阶常微分方程数值解法
3. 一阶常微分方程组的数值解法
4. 高阶常微分程的数值解法
5. 利用 MATLAB 求解微分方程

一、一阶常微分方程初值问题

- 种群增长、种群竞争模型、传染病扩散等问题往往归结为关于某个或某些状态变量的常微分方程或方程组的问题, 如状态变量是关于时间变化的, 则其定解问题称为常微分方程初值问题; 若状态变量是关于空间变化的, 则其定解问题称为常微分方程边值问题.
- 由常微分方程的定解理论可知, 绝大部分常微分方程的定解问题无法给出具体的解析解表达式.
- 设计具有足够精度的数值计算格式,借助于计算机编程或数学软件, 并求得其近似解成为必然.
- 数值计算技术是目前大规模或复杂**科技与工程计算**的关键技术.

1. 基本定义

称形如

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t,y), \quad t \in (a,b] \tag{1}$$

$$y(a) = \alpha, \tag{2}$$

为**一阶常微分方程的初值问题**, 其中, 区间 $[a,b]$ 为求解区间.

2. 解的存在、唯一性定理

定理:若函数 $f(t,y)$ 关于 y 满足Lipshitz 条件,则常微方程初值问题(1)-(2)的解存在且唯一.

二、常微分方程数值解法

1. 关于微分方程的解析解与数值解

- 利用数学理论推导的方法求得的微分方程解称为**解析解或精确解**, 其基本特点是得到一个状态变量关于自变量的函数关系表达式, 如$y = y(t)$.
- 利用数值计算方法求连续问题的近似解称为**数值解**. 数值解的基本特点是利用数值计算方法和计算机编程得到一系列**离散点上的近似解**.
- 为区别起见, 若用 $y = y(t)$ 表示解析解
  - 以 $y(t_i)$ 表示在 $t = t_i$ 时刻的精确值;
  - 以 y_i 表示在 $t = t_i$, $i = 1, 2, \cdots, N$ 时刻的数值解.

2. 常微分方程数值解法的基本步骤:

- **STEP 1: 区间离散化:** 先把区间 $[a, b]$ 离散成一系列离散点

$$a = t_0 < t_1 < t_2 < \ldots < t_N = b;$$

记 $h_k = t_{k+1} - t_k, k = 0, 1, 2, \cdots, N-1$, 称为**网格步长**.

  - 若 $h_k, k = 0, 1, 2, \cdots, N-1$ 不完全相同, 则称相应的数值计算方法为**变步长**方法, 任意网格点的坐标可以写为 $t_{k+1} = t_k + h_k, k = 0, 1, 2, \cdots, N-1$.
  - 若网格剖分均匀, 则记 $h = \frac{b-a}{N}$ 为网格步长, 此时任意网格点可以表示为 $t_k = a + kh$, $k = 0, 1, 2, \cdots, N$..

- **STEP 2: 构造数值计算格式:**把微分方程近似为差分方程, 即把求解析解问题化为求在各离散点上的数值近似值$\{y_i \approx y(t_i)\}_{i=0}^{N}$ 的问题.
- **STEP 3: 编程求解**.

I、欧拉格式(Euler Formula):常见的有三种构造方法

1 Taylor 法:

- 利用Taylor 多项式展开. 若 $y \in C^2[a,b]$, 则

$$\begin{aligned} y(t_{k+1}) &= y(t_k + h_k) \\ &= y(t_k) + y'(t_k)h_k + \frac{y''(\xi_k)}{2}h_k^2. \end{aligned}$$

- 由 $y \in C^2[a,b]$, 则 $y''(\xi_k)$ 有界, 又由微分方程(1)知 $y'(t_k) = f(t_k, y(t_k))$, 从而有

$$y(t_{k+1}) = y(t_k) + f(t_k, y(t_k))h_k + O(h_k^2).$$

- 若$\max\limits_{k} h_k$ 足够小, 则 $O(h_k^2)$ 是高阶无穷小量, 称为**局部截断误差**项. 忽略 $O(h_k^2)$ 项得:

$$y(t_{k+1}) \approx y(t_k) + f(t_k, y(t_k))h_k.$$

- 用$y(t_k)$ 表示微分方程的精确解, 用y_k 表示数值近似解, $k = 0, 1, \cdots, N$.
- **数值计算格式— 欧拉格式:**

$$\begin{cases} y_0 = \alpha \\ y_{k+1} = y_k + f(t_k, y_k)h_k, \quad k = 0, 1, 2, \cdots, N-1. \end{cases} \tag{3}$$

- 由于初值α 已知, 则据(3) 可利用计算机编程逐步求出 $y_1, y_2, \cdots, y_N$.

2. 数值微分方法：

若 $y \in C^2[a,b]$, 则利用导数的定义可得

$$\frac{y(t_{k+1}) - y(t_k)}{h_k} \approx y'(t_k) = f(t_k, y(t_k)) \tag{4}$$

以差商替代微商, 也可容易地导出的相同的数值计算公式.

3. 数值积分法：

由微分方程易得 $\mathrm{d}y = f(t,y)\mathrm{d}t$, 两端分别在子区间 $[t_k, t_{k+1}]$ 上求定积分

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} \mathrm{d}y = \int_{t_k}^{t_{k+1}} f(t,y)\mathrm{d}t, \tag{5}$$

方程右端采用左矩形公式近似,亦得相同的结果.

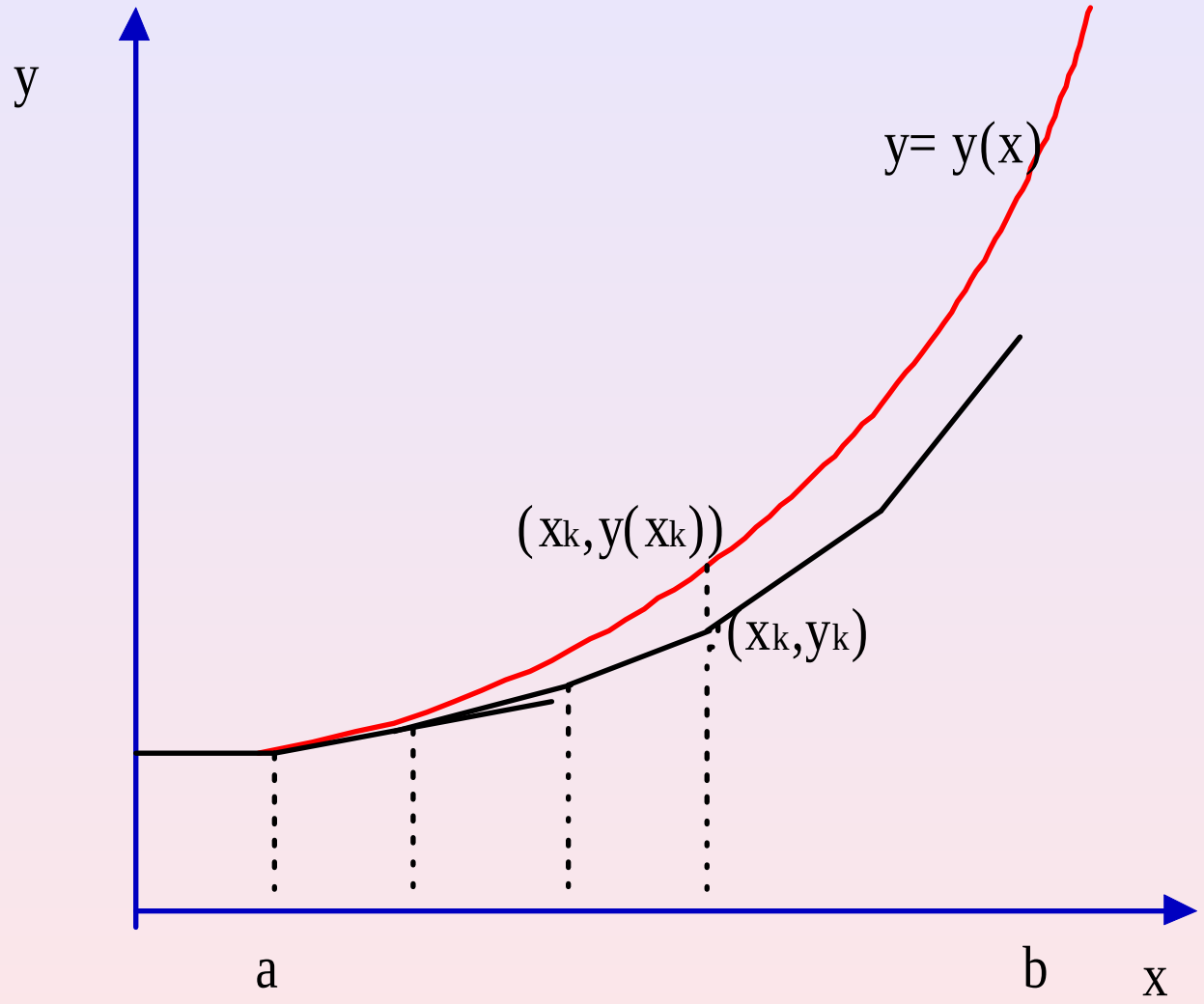
几点注释：

1. 若区间剖分为均匀的，令剖分步长$h = \frac{b-a}{N}$，则上述欧拉格式可简写为

$$\begin{cases} y_0 = \alpha \\ y_{k+1} = y_k + hf(t_k, y_k), \ k = 0, 1, 2, \cdots, N-1. \end{cases}$$

2. 前述欧拉方法在计算 y_{k+1} 时, 右端项均为已知项, 故该格式是一种显式数值计算方法(有时也叫**欧拉显式格式或向前的欧拉格式**).
3. 利用欧拉格式求微分方程数值解, 计算量小, 编程简单, 但计算精度很低. 实际计算时一般不采用.

欧拉格式的几何解释



欧拉隐式格式

若对方程(4) 左端的导数项采用向后差商

$$\frac{y(t_{k+1})-y(t_k)}{h} \approx \left.\frac{\mathrm{d}y}{\mathrm{d}t}\right|_{t=t_{k+1}} = f(t_{k+1}, y(t_{k+1}))$$

或在方程(5) 右端采用右矩形公式, 则形成**向后的欧拉格式— 欧拉隐式格式**。

$$\begin{cases} y_0 = \alpha \\ y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}), k = 0, 1, 2, \cdots, N-1 \end{cases}$$

II.梯形格式:

类似地,对方程(5) 右端积分项采用梯形公式近似, 则可导出下列计算格式

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, y_{k+1})] \tag{6}$$

这一差分格式称作**梯形公式**.

- 无论是向后的欧拉格式或梯形格式, 由于格式右端均含有 $f(t_{k+1}, y_{k+1})$, 而 y_{k+1}是一个未知量, 故它们本质上都是隐式格式.
- 隐式问题求解是数学上的一个不动点(如求 $x = g(x)$)问题, 可采用迭代方法求解, 但计算量大.

注释：

- 影响数值解计算精度的因素主要有: **截断误差和舍入误差**. 截断误差主要由格式设计方法产生, 舍入误差主要来自计算机的浮点运算. 两种误差都会随计算工作量的增加而产生累积误差.
- 梯形格式的计算精度要比单纯的欧拉格式要高, 其局部截断误差为 $O(h^3)$.
- 评价数值计算格式优劣的主要指标是：误差, 格式的稳定性, 数值解的收敛性等.
- 有关数值计算的详细理论介绍可查阅数值分析、计算方法、微分方程数值解等图书资料.

如何提高数值解计算精度？

1. 加密网格, 适当减小离散网格步长 h, 有助于改善计算结果的精确度.
2. 注意到, 数值计算格式中每一个状态值 y_{k+1} 的大小都与上一个状态 y_k 有关, 而每一步计算都会产生一个截断误差和舍入误差, 当网格步长变得很小时, 会导致状态点无限制地增加, 因而会造成因计算工作量无限制增大而造成误差的累积. 同时, 因计算点的增加, 也会引起计算速度的减慢.
3. 要解决误差累积和计算速度减慢的问题, 一个好的思想是采用变步长方法. 当问题的解变动较为剧烈时, 应采用小步长, 而当解变动较为平缓时, 应采用大步长.
4. 改进数值计算格式, 尽量采用高精度的格式或算法.如Runge-Kutta方法、Adams方法等.

III.修正格式—预估校正格式

- 先用显式欧拉方法求得一个初步的近似值 $\bar{y}_{k+1}$, 称之为预估值;
- 用预估的 $\bar{y}_{k+1}$ 代替(6) 式右端的y_{k+1}, 直接计算得到校正值 y_{k+1}.

$$\text{预估：} \quad \bar{y}_{k+1} = y_k + hf(t_k, y_k)$$

$$\text{校正:} \quad y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, \bar{y}_{k+1})]$$

称作基于梯形公式的**预估-校正格式**. 可直接写为：

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k))] \qquad (7)$$

IV. 龙格—库塔方法

基本思想: 考察(7), 它亦可改写成

$$\begin{cases} y_{i+1} = y_i + \frac{h}{2}(K_1 + K_2) \\ K_1 = f(t_i, y_i) \\ K_2 = f(t_{i+1}, y_i + hK_1) \\ i = 1, 2, \cdots, N-1 \end{cases}$$

1. K_1 和K_2 可以看作是解曲线在t_i 与t_{i+1} 两个点的斜率值;
2. 上式右端第二项即可理解为解曲线在t_i 与t_{i+1} 两个点的斜率值K_1 和K_2 的算术平均
3. 启示: 如果设法在$[t_i, t_{i+1}]$ 内多估算几个点的斜率值, 然后将它们取加权平均作为平均斜率, 则有可能构造出具有高精度的计算格式.

Runge-Kutta方法的一般形式:

$$y_{i+1} = y_i + \sum_{j=1}^{M} c_j K_j \tag{8}$$

其中

$$\begin{aligned} K_1 &= hf(t_i, y_i) \\ K_j &= hf(t_i + \alpha_j h, y_i + \sum_{l=1}^{j-1} b_{jl} K_l), j = 2, 3, \cdots, M \end{aligned}$$

$c_j, \alpha_j, b_{jl}, l = 1, 2, \cdots, j-1, j = 1, 2, \cdots, M$, 为系数项;

格式构造:

以二阶格式为例说明其构造过程. 考虑 $M = 2$ 的情形,则二阶龙格库塔格式为

$$y_{i+1} = y_i + c_1 K_1 + c_2 K_2 \tag{9}$$

其中:

$$\begin{aligned} K_1 &= hf(t_i, y_i) \\ K_2 &= hf(t_i + \alpha_2 h, y_i + b_{21} K_1), \end{aligned}$$

由二元泰勒展开式得:

$$\begin{aligned} & f(t_i + \alpha_2 h, y_i + b_{21} K_1) \\ &= f(t_i, y_i) + [\alpha_2 h f_t + b_{21} K_1 f_y] + O(h^2) \\ &= f(t_i, y_i) + [\alpha_2 h f_t + b_{21} h f_y f] + O(h^2) \end{aligned}$$

则代入格式(9)得

$$\begin{aligned} y_{i+1} &= y_i + c_1 K_1 + c_2 K_2 \\ &= y_i + (c_1 + c_2) h f + c_2 \alpha_2 h^2 f_t + c_2 b_{21} h^2 f_y f + O(h^3) \end{aligned}$$

又由Taylor 展开式得

$$\begin{aligned} y_{i+1} &= y(t_i + h) = y(t_i) + h y'(t_i) + \frac{h^2}{2} y'' + O(h^3) \\ &= y(t_i) + hf + \frac{h^2}{2} f' + O(h^3) \\ &= y(t_i) + hf + \frac{h^2}{2} [f_t + f_y f] + O(h^3) \end{aligned}$$

比较二式可得

$$\begin{aligned} c_1 + c_2 &= 1 \\ c_2\alpha_2 &= 1/2 \\ c_2 b_{21} &= 1/2 \end{aligned}$$

该方程组有无穷多解.

(1) 中点格式(Midpoint Method):

—$c_1 = 0, c_2 = 1, \alpha_2 = 1/2, b_{21} = 1/2$

$$\begin{cases} y_0 = \alpha \\ y_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)), i = 1, 2, \cdots, N-1 \end{cases}$$

(2) 修正欧拉格式(Modified Euler Method):

—$c_1 = 1/2, c_2 = 1/2, \alpha_2 = 1, b_{21} = 1$

$$\begin{cases} y_0 = \alpha \\ y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_i + hf(t_i, y_i))], i = 1, 2, \cdots, N-1. \end{cases}$$

(3). 四阶龙格-库塔方法

最常用的四阶经典的龙格-库塔公式:

$$
\begin{cases}
y_0 = \alpha, \\
y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\
K_1 = f(t_i, y_i), \\
K_2 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}K_1), \\
K_3 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}K_2), \\
K_4 = f(t_i + h, y_i + hK_3) \\
i = 1, 2, \cdots, N-1.
\end{cases}
$$

四阶经典的龙格-库塔MATLAB 参考程序：

```
function[x,y]=rk4(dyfun,xs,y0,h)
%功能：用四阶Runge-Kutta格式求解常微分方程
%初值问题: dy/dx=f(x,y),y(x0)=y0
% dyfun为f(x,y), xs为求解区间[x0,xn],
% y0为初值，h为剖分步长，x 返回节点，y返回节点上的数值解
x=xs(1):h:xs(2);
y(1)=y0;
for n=1:length(x)-1
k1=feval(dyfun,x(n),y(n));
k2=feval(dyfun,x(n)+h/2,y(n)+h/2*k1);
k3=feval(dyfun,x(n)+h/2,y(n)+h/2*k2);
k4=feval(dyfun,x(n+1),y(n)+h*k3);
y(n+1)=y(n)+h*(k1+2*k2+2*k3+k4)/6;
end
x=x'; y=y';
```

例1 用经典的四阶龙格-库塔方法计算：

$$\begin{cases} y' = y - \frac{2x}{y} & (0 \le x \le 1) \\ y_0 = 1 \end{cases}$$

取步长为0.2, 且与准确值比较.

MATLAB求解程序

```
clear;
dyfun=inline('y-2*x/y');
 [x,y]=rk4(dyfun,[0,1],1,0.2);
 [x,y]
ans =
0         1.0000
0.2000 1.1832
0.4000 1.3417
0.6000 1.4833
0.8000 1.6125
1.0000 1.7321
```

结果比较

x_n	y_n	精确解$y(x_n)$
0	1	1
0.2	1.183 229	1.183216
0.4	1.341 667	1.341 641
0.6	1.483 281	1.483240
0.8	1.612 514	1.612 452
1.0	1.732 142	1.732 051

1. 一阶常微分方程组的数值计算方法

只要把 y 和 f 理解为向量, 则前面所研究的各种算法即可推广应用到一阶常微分方程组的情形. 以如下最简单的常微分方程组为例

$$\begin{cases} y' = f(x, y, z), \\ z' = g(x, y, z), \\ y(x_0) = y_0, z(x_0) = z_0. \end{cases}$$

预估-校正格式

令$x_n = x_0 + nh, n = 1, 2, \cdots,$ 以y_n, z_n表示节点x_n 上的近似解, 则其基于梯形公式的预估-校正格式具有形式:

$$
\begin{aligned}
\text{预估:} \bar{y}_{n+1} &= y_n + hf(x_n, y_n, z_n) \\
\bar{z}_{n+1} &= z_n + +hg(x_n, y_n, z_n) \\
\text{校正:} \bar{y}_{n+1} &= y_n + \frac{h}{2}[f(x_n, y_n, z_n) + f(x_n, \bar{y}_{n+1}, \bar{z}_{n+1})] \\
\bar{z}_{n+1} &= z_n + \frac{h}{2}[g(x_n, y_n, z_n) + g(x_n, \bar{y}_{n+1}, \bar{z}_{n+1})]
\end{aligned}
$$

相应的四阶龙格一库塔格式为

$$
\begin{cases}
y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\
z_{n+1} = z_n + \frac{h}{6}(L_1 + 2L_2 + 2L_3 + L_4) \\
K_1 = f(x_n, y_n, z_n) \\
L_1 = g(x_n, y_n, z_n) \\
K_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1, z_n + \frac{h}{2}L_1) \\
L_2 = g(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1, z_n + \frac{h}{2}L_1) \\
K_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2, z_n + \frac{h}{2}L_2) \\
L_3 = g(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2, z_n + \frac{h}{2}L_2) \\
K_4 = f(x_{n+1}, y_n + hK_3, z_n + hL_3) \\
L_4 = g(x_{n+1}, y_n + hK_3, z_n + hL_3)
\end{cases}
$$

四、高阶常微分方程的数值计算方法

高阶微分方程的初值问题, 原则上总可以归结为一阶方程组来求解.

- 以二阶常微分方程为例：

$$\begin{cases} y''(x) = f(x, y, y'), x_0 \le x \le x_n \\ y(x_0) = y_0, y'(x_0) = z_0 \end{cases}$$

- 令$z = y'$, 化为一阶方程组求解:

$$\begin{cases} y'(x) = z, , x_0 \le x \le x_n \\ z'(x) = f(x, y, z), \\ y(x_0) = y_0, z(x_0) = z_0 \end{cases}$$

则问题即转化为一阶常微分方程组的问题.

1. 微分方程(组) 的解析解

MATLAB 利用函数dsolve 求微分方程(组) 的解析解(符号解).

其调用格式为:

dsolve(eq1,eq2,$\cdots$,eqn,cond1,cond2,$\cdots$,condn,v)

- eqn—表示第n 个方程，形式为$f_n(x,\alpha)=p$ 或$f_n(x,\alpha)$ (此时默认右端项为0). 如, 微分方程$\frac{\mathrm{d}^2y}{\mathrm{d}x^2}=0$ 应表达为: $D2y=0$.
- 微分方程表示：用字母D 表示求微分, 如$D, D2, D3$ 等表示求一阶、二阶、三阶微分, D后所跟的字母为因变量, 依此类推.
- 自变量可以指定，缺省为t.

- codn—-表示第n 个边界条件，输入规则同微分方程; condn 可以是含等号或不等号的符号或字符串, 并且有诸如$y(a) = b, Dy(a) = b$ 的形式.
- v—表示自变量，默认（缺省）时$v = t$.如Dy 表示$\frac{\mathrm{d}y}{\mathrm{d}t}$, Dny 表示$\frac{\mathrm{d}^n y}{\mathrm{d}t^n}$
- 如条件的个数少于微分方程的个数，在符号解中可能会出现类似于求不定积分时的积分常量C_1, C_2等

例2 求$\frac{du}{dt} = 1 + u^2$ 的通解.

解 命令及求解结果为

```
dsolve('Du=1+u^2','t')
ans=
    tan(t-C1)
```

例3 求下列微分方程的特解

$$\begin{cases} \frac{\mathrm{d}^2y}{\mathrm{d}x^2} + 4\frac{\mathrm{d}y}{\mathrm{d}x} + 29y = 0 \\ y(0) = 0, y'(0) = 15 \end{cases}$$

解

MATLAB及求解结果

```
y=dsolve('D2y+4*Dy+29*y=0','y(0)=0,Dy(0)=15','x')
y=
    3*exp(-2*x)*sin(5*x)
```

例3 求下列微分方程组的通解

$$\begin{cases} \frac{dx}{dt} = 2x - 3y + 3z \\ \frac{dy}{dt} = 4x - 5y + 3z \\ \frac{dz}{dt} = 4x - 4y + 2z \end{cases}$$

解

MATLAB及求解结果

```
[x,y,z]=dsolve('Dx=2*x-3*y+3*z','Dy=4*x-5*y+3*z',
        ... 'Dz=4*x-4*y+2*z');
x=simple(x) % 化简方程式
y=simple(y) % 化简方程式
z=simple(z) % 化简方程式
```

求解结果为:

x=

-(-C1-C2*exp(-3*t)+C2-C3*exp(-3*t))*exp(2*t)

y=

-(C1*exp(-4*t)-C1-C2*exp(-4*t)-C2*exp(-3*t)+C2-C3+C3*exp(-3*t))*exp(2*t)

z=

(-C1*exp(-4*t)+C1-C2+C2*exp(-4*t)+C3)*exp(2*t)

2. 利用MATLAB 求微分方程(组)的数值解

- 在 MATLAB 中用于求解常微分方程的数值解的函数较多, 常用的函数如表1 所示.
- 每一个函数都有自己的适用对象, 因此在具体使用时应先了解方程的形式和刚性性质, 再确定使用哪个函数求解.
- 如希望详细了解有关信息, 读者可在 MATLAB 启动页面中点击命令窗口的“Getting Started”, 然后查看目录:
  MATLAB \Mathematics\Functions\Nonlinear Numerical Methods
  然后在此目录下查看“Ordinary Differential Equations”子目录, 即可了解相关函数调用方式, 相关算法及应用示例.

MATLAB 中用于求解 ODE (Ordinary Differential Equations)问题的函数

表1：MATLAB 的微分方程求解函数

求解函数	适用类型	求解算法	注释
ode45	非刚性微分方程	龙格库塔方法	Runge-Kutta(4,5)格式, 适用于了解不多的初学者
ode23	非刚性微分方程	龙格库塔方法	Rounge-Kutta(2,3)格式, 适用中等刚性微分方程求解
ode113	非刚性微分方程	阿当姆斯方法	多步法
ode15s	刚性微分方程和 微分代数方程	NDF(BDF)	采用数值微分(NDF)和 向后微分(BDF)方法求解
ode23s	刚性微分方程	Rosenbrock	修正的Rosenbrock 方法
ode23t	适度刚性微分方程和 微分代数方程	梯形方法	适用于无数值衰减问题
ode23tb	刚性微分方程	TR-BDF2	利用带有梯形格式的隐式 Rounge-Kutta方法和二阶向后 微分(BDF)公式求解
ode15i	全隐式微分方程	BDF	采用向后差分(BDF)方法 求解$f(y, y', t) = 0$形式的微分方程

关于刚性微分方程

- 在常微分方程求解过程中，了解所研究的问题是否属于刚性问题是十分重要的一步.
- 一个问题被称之为是刚性(Stiffness)的是指，其解产生过程变化很慢，同时存在变化很快的解.直观理解就是问题的解在某个自变量的小邻域内范围内，函数值的变化非常剧烈，或者说函数曲线变得非常陡峭. 因此解决这类问题必须采用小步长计算. 所谓刚性问题通常是针对微分方程组而言的.
- 对初学者或者对微分方程了解不多的读者来说，选择一个可以兼顾刚性和非刚性问题，而且处理效率和可信度较高的函数，是一个不错的选择，MATLAB 中 ode45函数就具有这一特性. 以下以 ode45 函数为例，说明函数的调用方法.

MATLAB 求解函数调用方法: 以ODE45 函数调用为例

- **调用格式：** [t,y]=ode45(odefun,tspan,y0)
- **参数说明：**
  - odefun 为用以表示微分方程的右端项 $f(t, y)$ 的函数句柄或 inline 函数;
  - tspan 表示自变量的变化范围, 用区间[t0,ts]表示, t0 表示自变量的初始时间, ts 表示自变量取值的上限值, 若上限值为无穷大时可取一个足够大的正实数替代;
  - 若 tspan 为一个向量形式, 如 [t0, t1,t2,...,tN], 则表示给定网格剖分点, 求微分方程在这些给定点的数值解, 系统默认t0 为初始点.
  - y0 是微分方程的初始值向量, 其元素的个数等于微分方程的阶数, 特别地若方程为一阶常微分方程, 则 y0 为一个数值标量即在 t0 点的初始值; 输出参数 t, y 为系统自行设定的计算网格点以及网格点上的数值计算结果, 为向量或矩阵形式.

例4 求解描述振荡器的经典的 van der Pol 微分方程

$$\begin{cases} \frac{\mathrm{d}^2 y}{\mathrm{d}t^2} - \mu(1-y^2)\frac{\mathrm{d}y}{\mathrm{d}t} + y = 0, \\ y(0) = 1,\ y'(0) = 0. \end{cases}$$

解 首先把二阶常微分方程化成常微分方程组.设 $y_1 = y$, $y_2 = y'$. 则原问题转化为常微分方程组的形式:

$$\begin{cases} y_1' = y_2, \\ y_2' = \mu(1-y_1^2)y_2 - y_1, \\ y_1(0) = 1, y_2(0) = 0. \end{cases}$$

- 取 $\mu = 1$, 创建描述该微分方程组的函数文件(vdp1.m):
  function dy = vdp1(t,y)
  dy=zeros(2,1);
  dy(1)=y(2);
  dy(2)=1*(1-y(1)^2)*y(2)-y(1);
  end
- 调用 MATLAB 求解函数 ode45. 对于初值 $y(0) = 1, y'(0) = 0$, 取求解区间上限为 ts=20.

在 MATLAB 命令窗口下运行

```
[t,Y] = ode45(@vdp1,[0 20],[1 0]);
plot(t,Y(:,1),'k-o')
xlabel('t'),ylabel('y'),title('van der Pol Equation')
```

求解结果的可视化, 如下图所示.

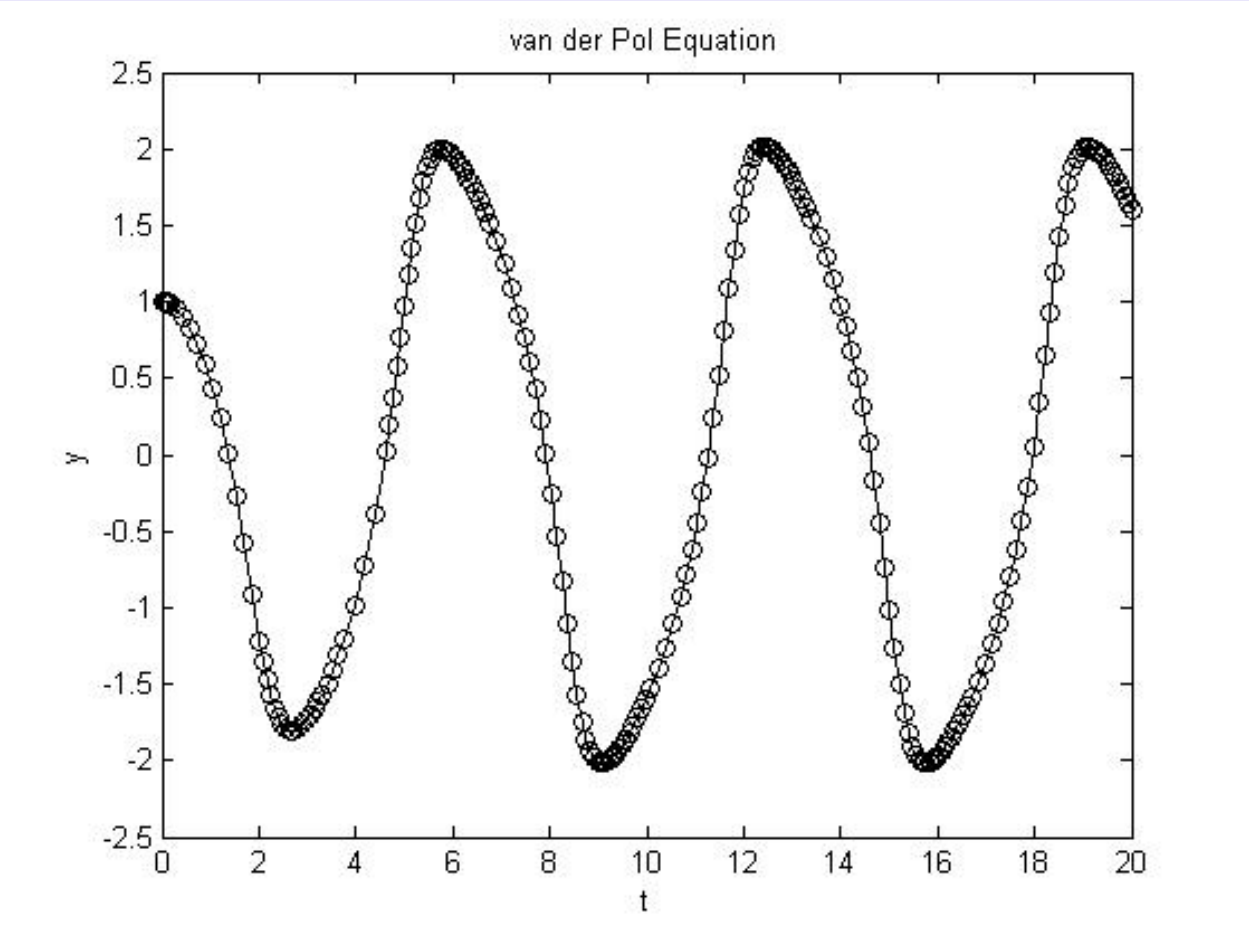


Figure: van der Pol 方程数值计算结果示意图

例5 求解如下常微分方程组

$$\begin{cases} y_1' = y_2 y_3 \\ y_2' = y_1 y_3 \\ y_3' = 0.51 y_1 y_2 \\ y_1(0) = 0, y_2(0) = 1, y_3(0) = 1 \end{cases}$$

建立rigid.m 函数文件

```
function dy=rigid(t,y)
dy=zeros(3,1);
dy(1)=y(2)*y(3);
dy(2)=-y(1)*y(3);
dy(3)=-0.51*y(1)*y(2);
end
```

取t_0=0,t_f=12,在MATLAB 命令窗口输入

```
[T,y]=ode45('rigid',[0 12],[0 1 1]);
plot(T,y(:,1),'-',T,y(:,2),'*',T,y(:,3),'+');
legend('y1','y2','y3')
xlabel('t')
```

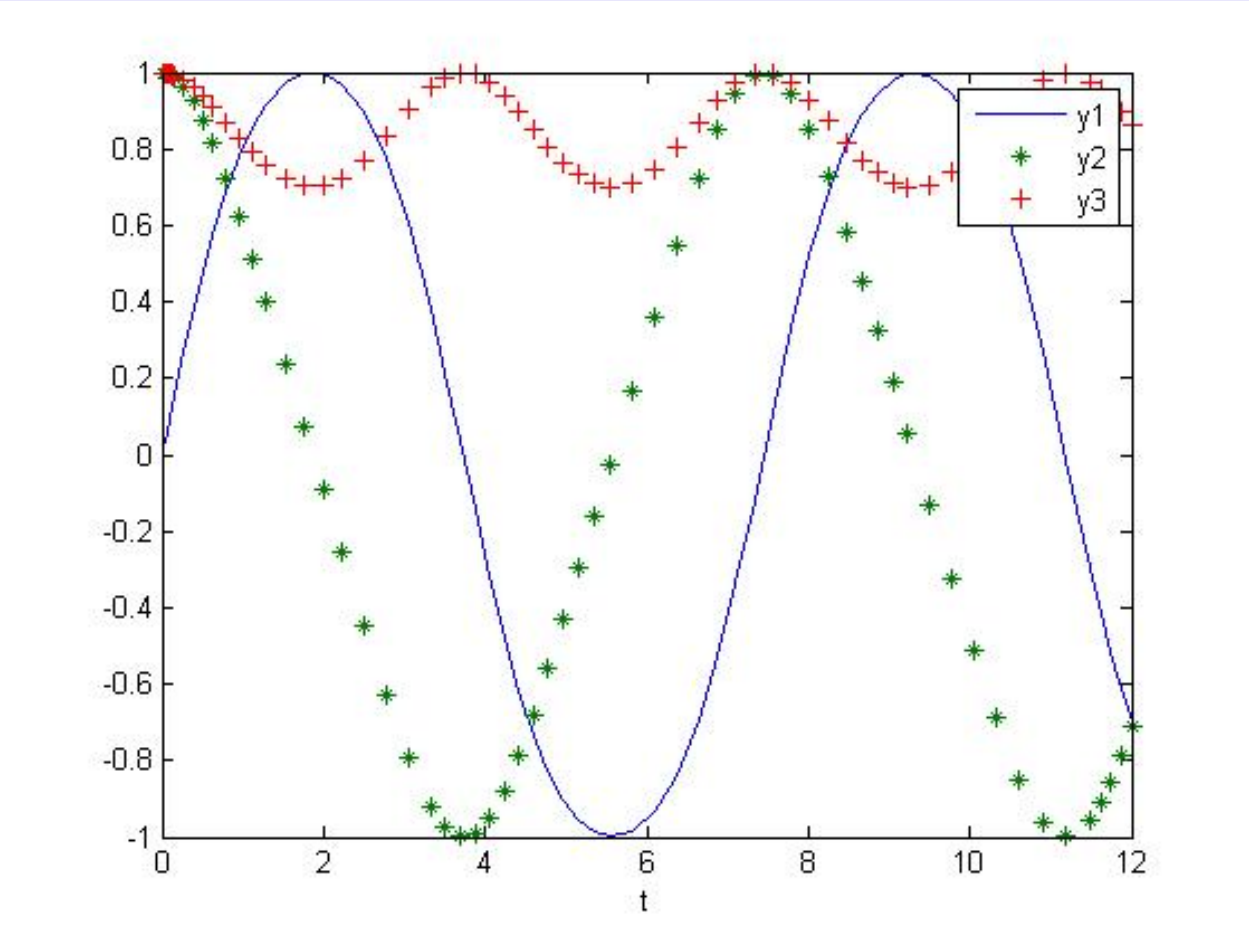


Figure: 数值计算结果示意图

3. 利用 MATLAB 求解偏微分方程

- 利用 MATLAB 求解偏微分方程的最简单的做法就是直接利用PDE 工具箱求解
- 调用方式为直接在命令窗口下键入 pdetool 即可进入人机交互界面
- 用户可根据实际需要选择相应的 PDE(Partial Differential Equations) 类型及参数(定义域、边界条件、初始条件等), 选择求解区域并生成所需要的矩形或三角形网格. 然后直接求解即可.

五、建模案例：再论导弹追踪问题

首先要将此问题表达为等价的微分方程组，现在建立此问题的参数方程.

- 设在任意时刻 t，乙舰的坐标为 $Q(X(t), Y(t))$, 导弹的坐标为 $P(x(t), y(t))$.
- 设导弹的速度恒为ω，则

$$\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2 = \omega^2 \tag{10}$$

- 由于导弹头始终对准乙舰，故导弹头的速度向量平行于乙舰与导弹头位置的差向量，即：

$$\begin{bmatrix} \frac{\mathrm{d}x}{\mathrm{d}t} \\ \frac{\mathrm{d}y}{\mathrm{d}t} \end{bmatrix} = \lambda \begin{bmatrix} X - x \\ Y - y \end{bmatrix}, \lambda > 0. \tag{11}$$

- 故

$$\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2+\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2=\lambda^2[(X-x)^2+(Y-y)^2] \tag{12}$$

- 将(10)代入(12),得

$$\lambda=\frac{\omega}{\sqrt{(X-x)^2+(Y-y)^2}} \tag{13}$$

- 将(13)代入(11),得

$$\begin{cases}\frac{dx}{dt}=\frac{\omega}{\sqrt{(X-x)^2+(Y-y)^2}}(X-x)\\ \frac{dy}{dt}=\frac{\omega}{\sqrt{(X-x)^2+(Y-y)^2}}(Y-y)\end{cases} \tag{14}$$

- 因乙舰以速度 v_0 沿直线 $x=1$ 运动，设 $v_0=1$，则 $\omega=5$.有

$$X=1,\quad Y=t$$

- 初始条件的确定: $t=0$时，甲舰位于原点，故

$$x(0)=0,\quad y(0)=0$$

- 因此导弹运动轨迹的参数方程为:

$$\begin{cases} \frac{dx}{dt}=\frac{5}{\sqrt{(1-x)^2+(t-y)^2}}(1-x) \\ \frac{dy}{dt}=\frac{5}{\sqrt{(1-x)^2+(t-y)^2}}(t-y) \\ x(0)=0, y(0)=0 \end{cases} \tag{15}$$

利用MATLAB 求解

建立M 函数文件eq2.m 如下

```
function dy=eq2(t,y)
dy=zeros(2,1);
dy(1)=5*(1-y(1))\sqrt((1-y(1)^2)+(t-y(2))^2);
dy(2)=5*(t-y(2))\sqrt((1-y(1)^2)+(t-y(2))^2);
end
```

在命令窗口下,输入

```
[t,y]=ode45('eq2',[0 2],[0 0]);
plot(y(:,1),y(:,2),'*')
xlabel('x'),xlabel('y')
```

计算结果的可视化