

# 第七讲 常微分方程数值解法

## (Numerical Methods for Ordinary Differential Equations)

# 一、引言

- 在许多实际问题中，往往不能找出所需要的函数关系，但是根据问题所提供的情况，有时可以列出含有要找的函数及其导数的关系式，这样的关系式就是所谓的微分方程。

例 一曲线通过点  $(1, 2)$ ，且在该曲线上任意点  $M(x, y)$  处的切线斜率为横坐标的两倍，求这曲线的方程。

# 什么是微分方程

- 表示未知函数、未知函数的导数及自变量之间的关系方程。（未知函数的导数必须出现）
  - 如果其中的未知函数只与一个自变量有关，则称为常微分方程；
  - 如果未知函数是两个或两个以上自变量的函数，并且在方程中出现偏导数，则称为偏微分方程。

判断下列方程是否为微分方程：

$$x^2 + xy + y^2 = 0$$

$$x + y'' = 0$$

$$3y' = c$$

# 微分方程的阶

- 微分方程中所出现的未知函数的最高阶导数的阶数。

$$\frac{dy}{dx} = 2x$$

$$x^2 y''' + xy'' - 4y' = 3x$$

$$y^4 - 2y''' - 12y' + 5y = \sin 2x$$

# 微分方程的一般形式

## 1、一阶微分方程

$$y' = f(x, y) \text{ 或 } F(x, y, y') = 0$$

## 2、二阶微分方程

$$y'' = f(x, y, y') \text{ 或 } F(x, y, y', y'') = 0$$

# 为什么研究常微分方程？

- 微分方程有着深刻而生动的实际背景
  - 在自然界中，很多时候为了刻画客观对象的运动规律或变化规律，经常需要描述变量之间的函数关系。
  - 但针对实际问题，我们通常很难直接找到这种函数关系，却容易建立起变量所满足的微分方程。
  - 如果方程可求解，则可以得到描述客观对象运动规律或变化规律的函数关系。

## 初值问题

求一阶微分方程  $F(x, y, y') = 0$  满足初值条件  $y|_{x=x_0} = y_0$  的特解的问题，称为一阶微分方程的初值问题。

记为

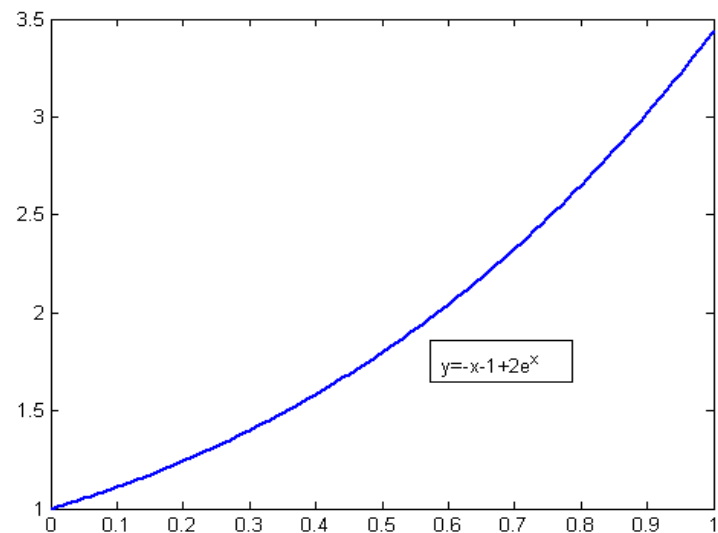
$$\begin{cases} F(x, y, y') = 0 \\ y|_{x=x_0} = y_0 \end{cases}$$



例如：

$$\begin{cases} y' = x + y, & x \in [0, 1] \\ y(0) = 1 \end{cases}$$

其解析解为： $y = -x - 1 + 2e^x$



## 本章重点讨论定解问题(初值问题)

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases} \quad \leftarrow \text{定解条件 (初始条件)}$$

是否能够找到定解问题的解取决于  $f(x, y)$

仅有极少数的方程可以通过“常数变易法”、“可分离变量法”等特殊方法求得初等函数形式的解，绝大部分方程至今无法理论求解。

如

$$y' = \sin(xy^2), \quad y' = \sqrt{1 + x^{\sin y}}, \quad y' = e^{\sqrt{x^2 + xy}} \quad \text{等等}$$

只要函数 $f(x, y)$ 适当光滑连续，  
且关于 $y$ 满足李普希兹(Lipschitz)条件，  
即存在常数 $L$ ，使得

$$|f(x, y) - f(x, \bar{y})| \leq L|y - \bar{y}|$$

由常微分方程理论知：

初值问题的解必存在且唯一。

# 数值解法含义

- 所谓数值解法, 就是设法将常微分方程离散化, 建立差分方程, 给出解在一些离散点上的近似值。
- 微分方程的数值解: 设方程问题的解 $y(x)$ 的存在区间是 $[a, b]$ , 令 $a = x_0 < x_1 < \dots < x_n = b$ , 其中 $h_k = x_{k+1} - x_k$ , 如是等距节点 $h = (b - a) / n$ ,  $h$ 称为步长。
- 由于 $y(x)$ 的解析表达式不容易得到或根本无法得到, 我们用数值方法求得 $y(x)$ 在每个节点 $x_k$ 上 $y(x_k)$ 的近似值, 用 $y_k$ 表示, 即 $y_k \approx y(x_k)$ , 这样 $y_0, y_1, \dots, y_n$ 称为微分方程的数值解。

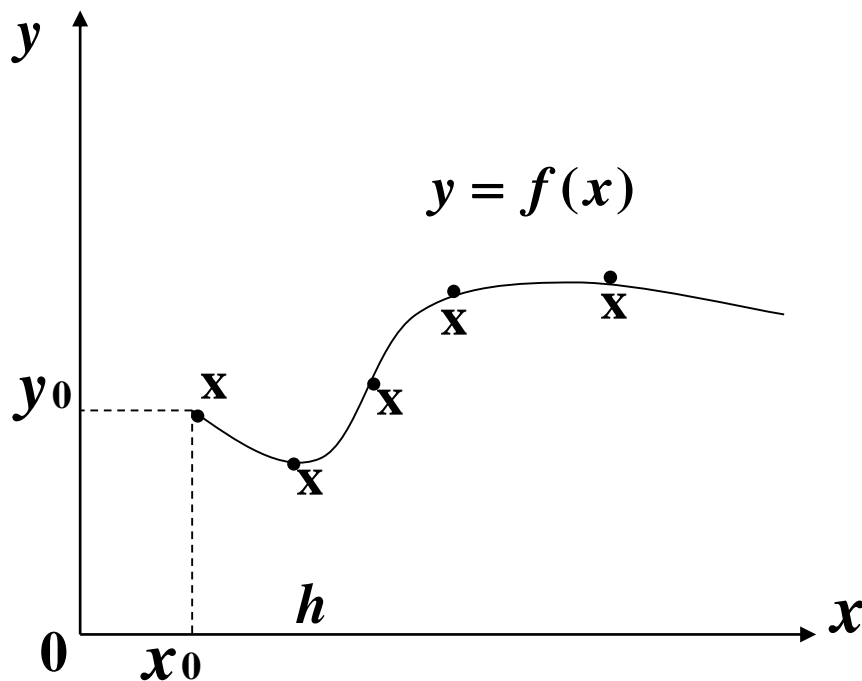
本章：求微分方程的数值解，即解函数 $y(x)$ 在一些离散点

$$x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1}$$

上的值

的近似解

$$\left. \begin{array}{cccc} y(x_1) & y(x_2) & y(x_n) & y(x_{n+1}) \\ y_1 & y_2 & \cdots & y_n & y_{n+1} \end{array} \right\} y_n \approx y(x_n)$$



初值  
问题的  
常见  
解法

单步法:

利用前一个单步的信息(一个点), 在  
 $y=f(x)$  上找下一点 $y_i$ ,

有欧拉法, 龙格-库塔法。

预测校正法:

多步法, 利用一个以上的前点信息求  
 $f(x)$ 上的下一个 $y_i$ ,

常用迭代法, 如改进欧拉法, 阿当姆斯法。

# 微分方程的数值解法需要解决的主要问题

**(1)** 如何将微分方程离散化，并建立求其数值解的迭代公式？

基本方法有：有限差分法（数值微分）、有限体积法（数值积分）、有限元法（函数插值）等等

**(2)** 如何估计迭代公式的局部截断误差与整体误差？

**(3)** 如何保证迭代公式的稳定性与收敛性？

# 数值解的思想

(1) 将连续变量  $x \in [a, b]$  离散为

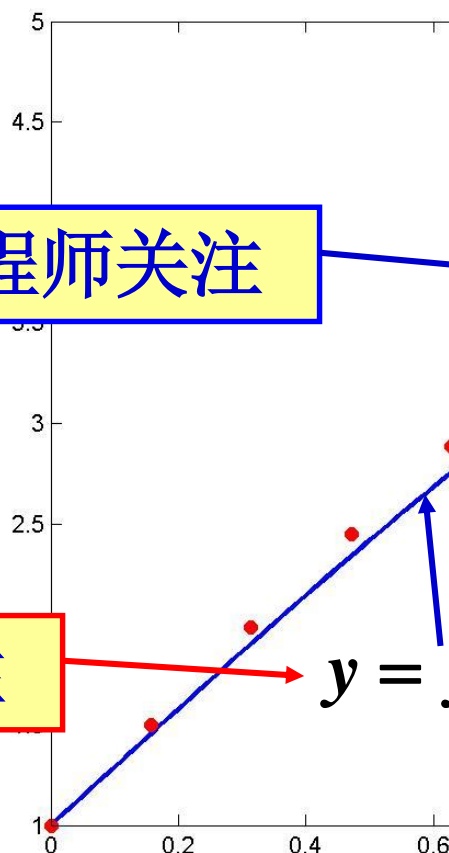
$$a = x_0 < x_1 < \dots < x_n = b$$

(2) 用代数的方法求解

$$y_k \approx y(x_k)$$

工程师关注

数学界关注



如果找不到解函数

数学界还关注:

解的存在性

解的唯一性

解的光滑性

解的振动性

解的周期性

解的稳定性

解的混沌性

.....



## 二、欧拉方法

- 18世纪最杰出的数学家之一，13岁时入读巴塞尔大学，15岁大学毕业，16岁获得硕士学位。

- 1727年-1741年（20岁-34岁）在彼得堡科学院从事研究工作，在分析学、数论、力学方面均有出色成就，并应俄国政府要求，解决了不少地图学、造船业等实际问题。

- 24岁晋升物理学教授。

- 1735年（28岁）右眼失明。



**Euler(瑞士)**  
**1707 ~ 1783**

- 1741年 – 1766（34岁-59岁）任德国科学院物理数学所所长，任职25年。在行星运动、刚体运动、热力学、弹道学、人口学、微分方程、曲面微分几何等研究领域均有开创性的工作。
- 1766年应沙皇礼聘重回彼得堡，在1771年（64岁）左眼失明。
- Euler是数学史上最多产的数学家，平均以每年800页的速度写出创造性论文。他去世后，人们用35年整理出他的研究成果74卷。

- 一. 欧拉 (*Euler*) 格式

设节点为  $x_i = a + ih$  ( $i = 0, 1, 2, \dots, n$ ) 式中:  $h = \frac{b-a}{n}$

方法一: *Taylor* 展开法

$$y(x_{i+1}) = y(x_i) + y'(x_i)(x_{i+1} - x_i) + \frac{y''(\xi_i)}{2!}(x_{i+1} - x_i)^2$$

$$= y(x_i) + hf(x_i, y(x_i)) + \frac{h^2}{2} y''(\xi_i) \text{ 或}$$

可忽略

$$y_{i+1} = y_i + hf(x_i, y_i) \quad (i = 0, 1, 2, \dots, n-1)$$

*Euler* 显格式, 可循环求解。

方法二: 数值微分法——实质还是 *Taylor* 展开法 (略)

### 方法三：数值积分法

将方程 $\frac{dy}{dx} = f(x, y)$ 两端于 $[x_i, x_{i+1}]$ 积分

$$\int_{x_i}^{x_{i+1}} y' dx = \int_{x_i}^{x_{i+1}} f(x, y(x)) dx$$

$$y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x, y(x)) dx \quad \text{式3}$$

对右端积分采用左矩形积公式，得

$$y(x_{i+1}) = y(x_i) + hf(x_i, y_i)$$

$$y_{i+1} = y_i + hf(x_i, y_i)$$

左矩形

补充:

梯形公式

$$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a)+f(b)]$$

左矩形公式

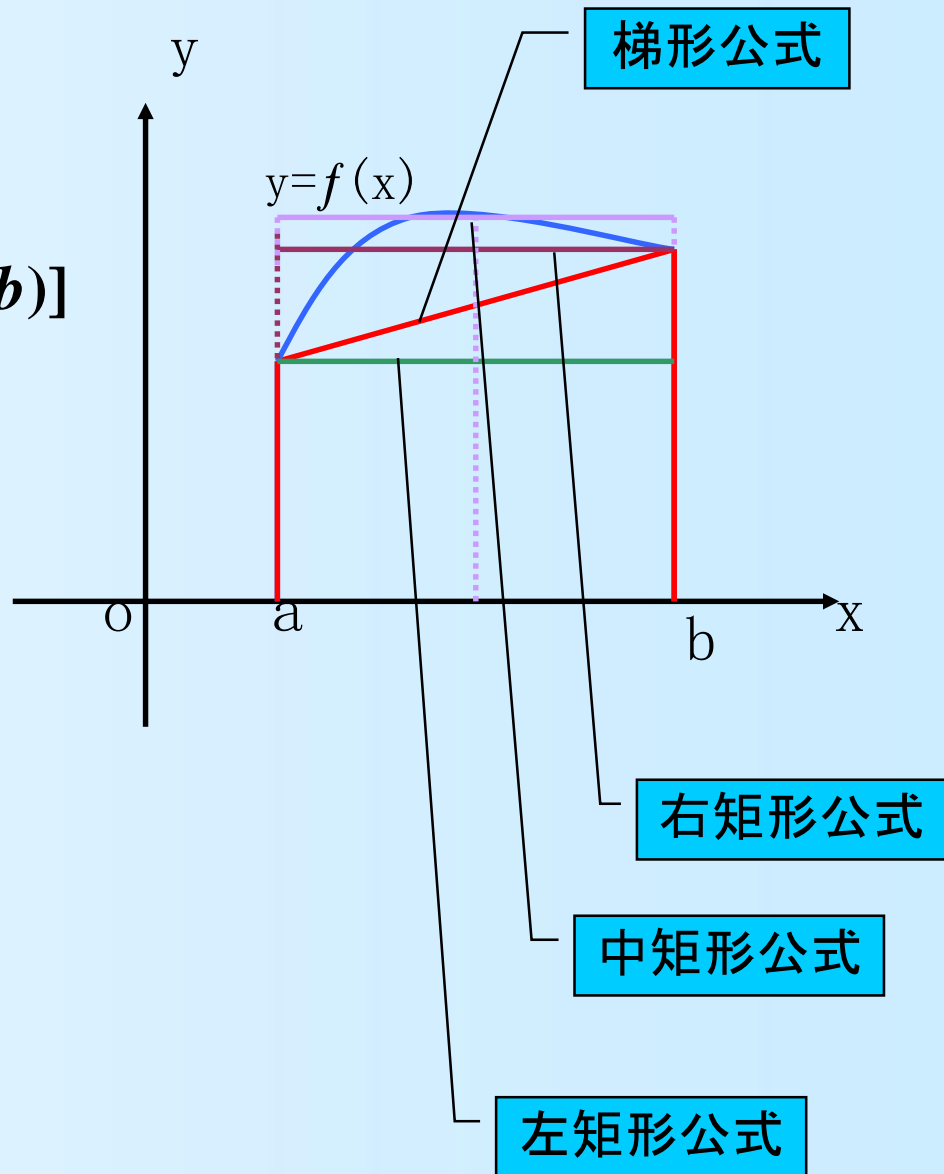
$$\int_a^b f(x)dx \approx (b-a)f(a)$$

右矩形公式

$$\int_a^b f(x)dx \approx (b-a)f(b)$$

中矩形公式

$$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right)$$



方法四：几何方法→ *Euler*法的几何意义

过 $(x_0, y(x_0)) = (x_0, y_0)$ 作切线 $y = y_0 + k(x - x_0)$

斜率 $k = y'(x_0) = f(x_0, y(x_0))$

$$k = f(x_0, y_0)$$

$y = y_0 + f(x_0, y_0)(x - x_0)$ 与 $x = x_1$ 求交点，

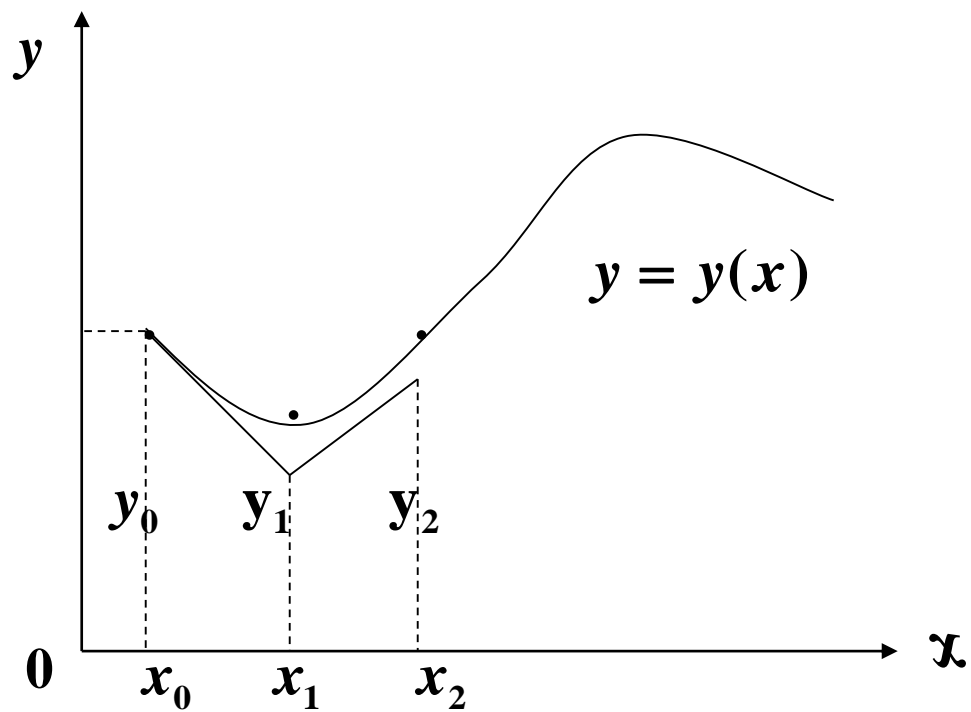
纵坐标记为 $y_1$ ，则 $y_1 = y_0 + hf(x_0, y_0)$

过 $(x_1, y_1)$ 以 $f(x_1, y_1)$ 为斜率作直线

$y = y_1 + f(x_1, y_1)(x - x_1)$ 与 $x = x_2$ 求交点，

纵坐标记为 $y_2$ ，则 $y_2 = y_1 + hf(x_1, y_1)$

.....





➤ 欧拉公式: /\* Euler's Method \*/

向前差商近似导数  $\rightarrow y'(x_0) \approx \frac{y(x_1) - y(x_0)}{h}$

$$y(x_1) \approx y(x_0) + hy'(x_0) = y_0 + hf(x_0, y_0) \underline{\underline{\text{记为}}} y_1$$

$$y_{i+1} = y_i + hf(x_i, y_i) \quad (i = 0, \dots, n-1)$$

➤ 隐式欧拉法 /\* implicit Euler method \*/

向后差商近似导数  $\rightarrow y'(x_1) \approx \frac{y(x_1) - y(x_0)}{h}$

$\rightarrow y(x_1) \approx y_0 + h f(x_1, y(x_1))$

$$y_{i+1} = y_i + h f(x_{i+1}, y_{i+1}) \quad (i = 0, \dots, n-1)$$

由于未知数  $y_{i+1}$  同时出现在等式的两边，不能直接得到，故称为隐式 /\* implicit \*/ 欧拉公式，而前者称为显式 /\* explicit \*/ 欧拉公式。

一般先用显式计算一个初值，再迭代求解。

例：用欧拉法解初值问题

$$\begin{cases} \frac{dy}{dx} = 1 - xy & (0 < x < 1) \\ y(0) = 0 \end{cases}$$

取步长 $h = 0.2$ 。计算过程保留4位小数。

解：因为 $f(x, y) = 1 - xy$ ,  $h = 0.2$ ,

欧拉公式为：

$$\begin{aligned} y(x_{i+1}) &\approx y_{i+1} = y_i + hf(x_i, y_i) \\ &= y_i + 0.2(1 - x_i y_i) \quad (i = 0, 1, 2, 3, 4, 5) \end{aligned}$$

当 $i=0$ ，即 $x_0=0$ 时，

$$y_1=0+0.2(1-0\times 0)=0.2000$$

当 $i=1$ ，即 $x_1=0.2$ 时，

$$y_2=0.2+0.2(1-0.2\times 0.2)=0.3920$$

当 $i=2$ ，即 $x_2=0.4$ 时，

$$y_3=0.392+0.2(1-0.4\times 0.392)=0.56064$$

当 $i=3$ ，即 $x_3=0.6$ 时，

$$y_4=0.56064+0.2(1-0.6\times 0.56064)=0.6933632$$

当 $i=4$ ，即 $x_4=0.8$ 时，

$$y_5=0.6933632+0.2(1-0.8\times 0.6933632)=0.782425088$$

列表计算如下：

$i$	0	1	2	3	4	5
$x_i$	0	0.2	0.4	0.6	0.8	1
$y_i$	0.0000	0.200	0.3920	0.5606	0.6934	0.7824

例：利用Euler方法求初值问题

$$\begin{cases} y' = \frac{1}{1+x^2} - 2y^2, & 0 \leq x \leq 2 \\ y(0) = 0 \end{cases}$$

的数值解，此问题的精确解是 $y(x) = x/(1+x^2)$ 。

解：此时的Euler公式为

$$\begin{cases} y_{i+1} = y_i + h(\frac{1}{1+x_i^2} - 2y_i^2) \\ y_0 = 0, i = 0, 1, 2, \dots \end{cases}$$

分别取步长 $h = 0.2, 0.1, 0.05$ ，计算结果如下表：

$h$	$x_i$	$y_i$	$y(x_i)$	$y(x_i)-y_i$
$h=0.2$	0.00	0.00000	0.00000	0.00000
	0.40	0.37631	0.34483	-0.03148
	0.80	0.54228	0.48780	-0.05448
	1.20	0.52709	0.49180	-0.03529
	1.60	0.46632	0.44944	-0.01689
	2.00	0.40682	0.40000	-0.00682
$h=0.1$	0.00	0.00000	0.00000	0.00000
	0.40	0.36085	0.34483	-0.01603
	0.80	0.51371	0.48780	-0.02590
	1.20	0.50961	0.49180	-0.01781
	1.60	0.45872	0.44944	-0.00928
	2.00	0.40419	0.40000	-0.00419
$h=0.05$	0.00	0.00000	0.00000	0.00000
	0.40	0.35287	0.34483	-0.00804
	0.80	0.50049	0.48780	-0.01268
	1.20	0.50073	0.49180	-0.00892
	1.60	0.45425	0.44944	-0.00481
	2.00	0.40227	0.40000	-0.00227

## 梯形格式

在(式3)中，对右端积分采用梯形求积公式得：

$$y(x_{i+1}) \approx y(x_i) + \frac{h}{2} [f(x_i, y(x_i)) + f(x_{i+1}, y(x_{i+1}))]$$

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_{i+1})]$$

显、隐式两种算法的平均

$$(i = 0, 1, 2, \dots, n-1) \quad (\text{式4})$$

(式4)是隐格式。实用上

- 1、化为显格式
- 2、用迭代法求  $y_{i+1}$ ，初值用 *Euler* 显格式确定。

$$y_{i+1} = y_i + h f(x_i, y_i) \quad (i = 0, \dots, n-1)$$

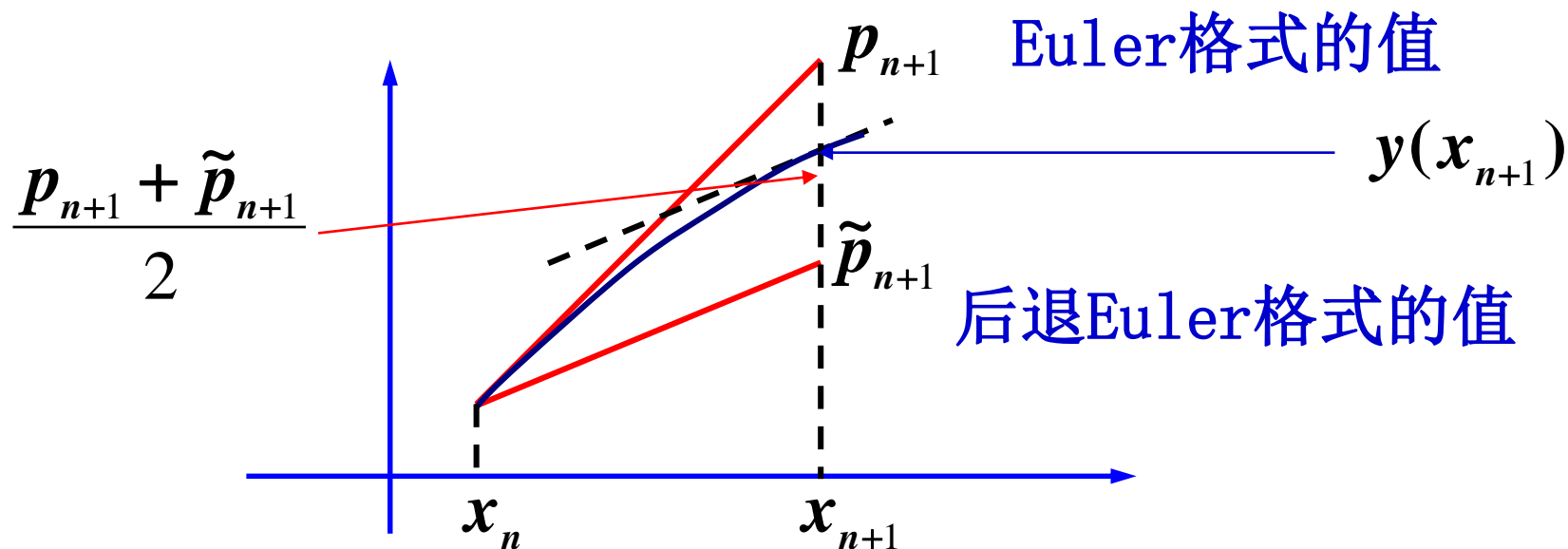
Euler格式

$$y_{i+1} = y_i + h f(x_{i+1}, y_{i+1})$$

后退Euler格式

隐式格式

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_{i+1})] \quad (i = 0, \dots, n-1)$$





 中点欧拉公式 /\* midpoint formula \*/

中心差商近似导数  $\rightarrow y'(x_1) \approx \frac{y(x_2) - y(x_0)}{2h}$

$$y(x_2) \approx y(x_0) + 2h f(x_1, y(x_1))$$

$$y_{i+1} = y_{i-1} + 2h f(x_i, y_i) \quad i = 1, \dots, n-1$$

### 三. *Euler*预估—校正法

/\* predictor-corrector method \*/

方 法		
显式欧拉	简单	精度低
隐式欧拉	稳定性最好	精度低, 计算量大
梯形公式	精度提高	计算量大
中点公式	精度提高, 显式	多一个初值, 可能影响精度

将*Euler*格式和梯形格式综合使用可得到  
改进的欧拉格式。

改进欧拉法 /\* modified Euler's method \*/

*Step 1:* 先用显式欧拉公式作预测，算出  $\bar{y}_{i+1} = y_i + h f(x_i, y_i)$

*Step 2:* 再将  $\bar{y}_{i+1}$  代入隐式梯形公式的右边作校正，得到

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, \bar{y}_{i+1})]$$

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_i + h f(x_i, y_i))] \quad (i = 0, \dots, n-1)$$

$$\begin{cases} \overline{y_{i+1}} = y_i + hf(x_i, y_i) \\ y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, \overline{y_{i+1}})] \\ y_0 = \alpha \end{cases} \quad (\text{式5})$$

$$(i = 0, 1, 2, \dots, n-1)$$

当 $y_i$ 已知时，有第一式预估出初值 $\overline{y_{i+1}}$ ，再代入第二式反复校正（迭代）。

改进的*Euler*方法也可以写成

$$\begin{cases} y_{i+1} = y_i + \frac{h}{2}(K_1 + K_2) \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + h, y_i + hK_1) \\ y_0 = \alpha, i = 0, 1, 2, \dots, n-1 \end{cases}$$

例：求初值问题

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

的数值解，取步长  $h = 0.1$ 。（精确解为  $y(x) = (1 + 2x)^{1/2}$ ）

解：

(1) 利用 *Euler* 方法

$$\begin{cases} y_{i+1} = 1.1y_i - 0.2x_i / y_i \\ y_0 = 1, i = 0, 1, 2, \dots, 9 \end{cases}$$

(2) 利用改进 *Euler* 方法

$$\begin{cases} y_{i+1} = y_i + 0.05(K_1 + K_2) \\ K_1 = y_i - 2x_i / y_i \\ K_2 = y_i + 0.1K_1 - \frac{2(x_i + 0.1)}{y_i + 0.1K_1} \\ y_0 = 1, i = 0, 1, 2, \dots, 9 \end{cases}$$

计算结果如下:

$i$	$x_i$	Euler方法 $y_i$	改进Euler法 $y_i$	精确解 $y(x_i)$
0	0	1	1	1
1	0.1	1.1	1.095909	1.095445
2	0.2	1.191818	1.184096	1.183216
3	0.3	1.277438	1.266201	1.264991
4	0.4	1.358213	1.343360	1.341641
5	0.5	1.435133	1.416402	1.414214
6	0.6	1.508966	1.485956	1.483240
7	0.7	1.580338	1.552515	1.549193
8	0.8	1.649783	1.616476	1.612452
9	0.9	1.717779	1.678168	1.673320
10	1	1.784770	1.737869	1.732051

# 一些基本概念



## 基本概念

---

### 1. 显式方法:

计算 $y_{n+1}$ , 不需解方程, 可以直接求出.  
如Euler方法.

### 2. 隐式方法:

计算 $y_{n+1}$ , 需要解方程, 一般采用迭代法.  
如后退Euler方法, 梯形方法.





### 3. 单步方法:

---

计算 $y_{n+1}$ 时, 只用到 $y_n$ 一点的值.

如: 向前, 向后Euler方法, 梯形方法.

### 4. 多步方法:

计算 $y_{n+1}$ 时, 需要用到多点的值, 若计算 $y_{n+1}$ 需要用到 $y_n, y_{n-1}, \dots, y_{n-k+1}$ 等 $k$ 点的值, 相应的方法称为 $k$ 步方法.

如: Euler两步方法.  $y_{n+1} = y_{n-1} + 2hf(x_n, y_n)$



## 5. 局部截断误差

离散化微分方程得到数值方法时, 舍去的部分称为局部截断误差. 它等于在第 $n$ 步为准确解情况下 (即:  $y_n = y(x_n)$ ), 由数值方法计算 $y_{n+1}$ ,  $y(x_{n+1}) - y_n$  称为局部截断误差.

已有结果:            局部截断误差

Euler方法:             $O(h^2)$

后退Euler方法:         $O(h^2)$

两步Euler方法:         $O(h^3)$

梯形方法:             $O(h^3)$



## 6. 整体截断误差

---

在无舍入误差的情况下, 由数值方法求出微分方程的近似解:

$$Y_1, Y_2, \dots, Y_N.$$

称  $\varepsilon_n = y(x_n) - Y_n$  为数值方法的**整体截断误差**.

可见, 整体截断误差是局部截断误差在每步的积累.



可以证明：

方法	整体截断误差
Euler方法	$O(h)$
后退Euler方法	$O(h)$
两步Euler方法	$O(h^2)$
梯形方法	$O(h^2)$

整体截断误差的阶比局部截断误差的阶低一阶。



## 7. 数值方法的阶

若数值方法整体截断误差的阶为 $O(h^p)$ , 则称该数值方法的阶为 $p$ .

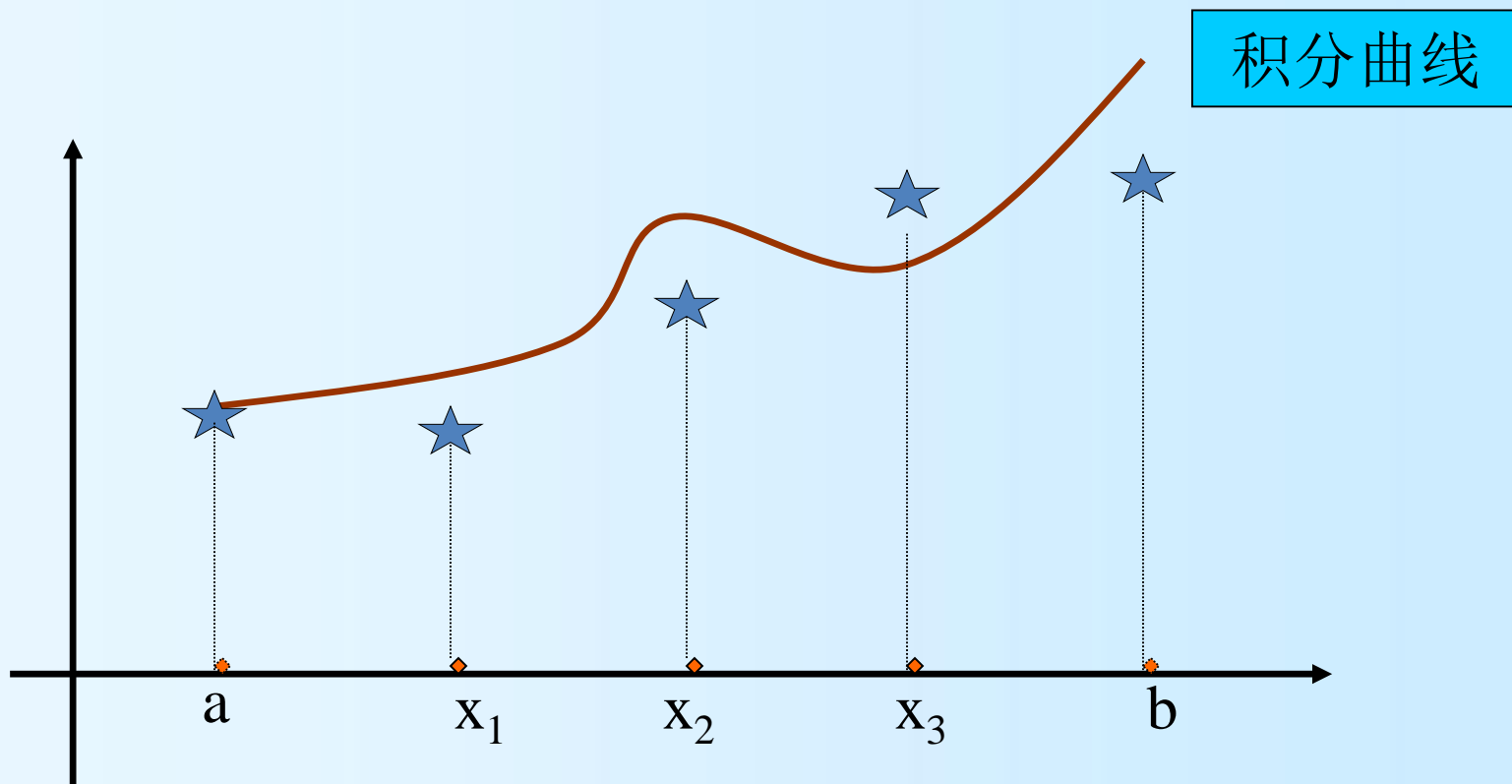
方法	方法的阶
Euler方法	1阶
后退Euler方法	1阶
两步Euler方法	2阶
梯形方法	2阶

### 三、龙格—库塔方法

数值解法的含义：将 $[a,b]$ 分成 $n$ 等份，得 $n+1$ 个节点

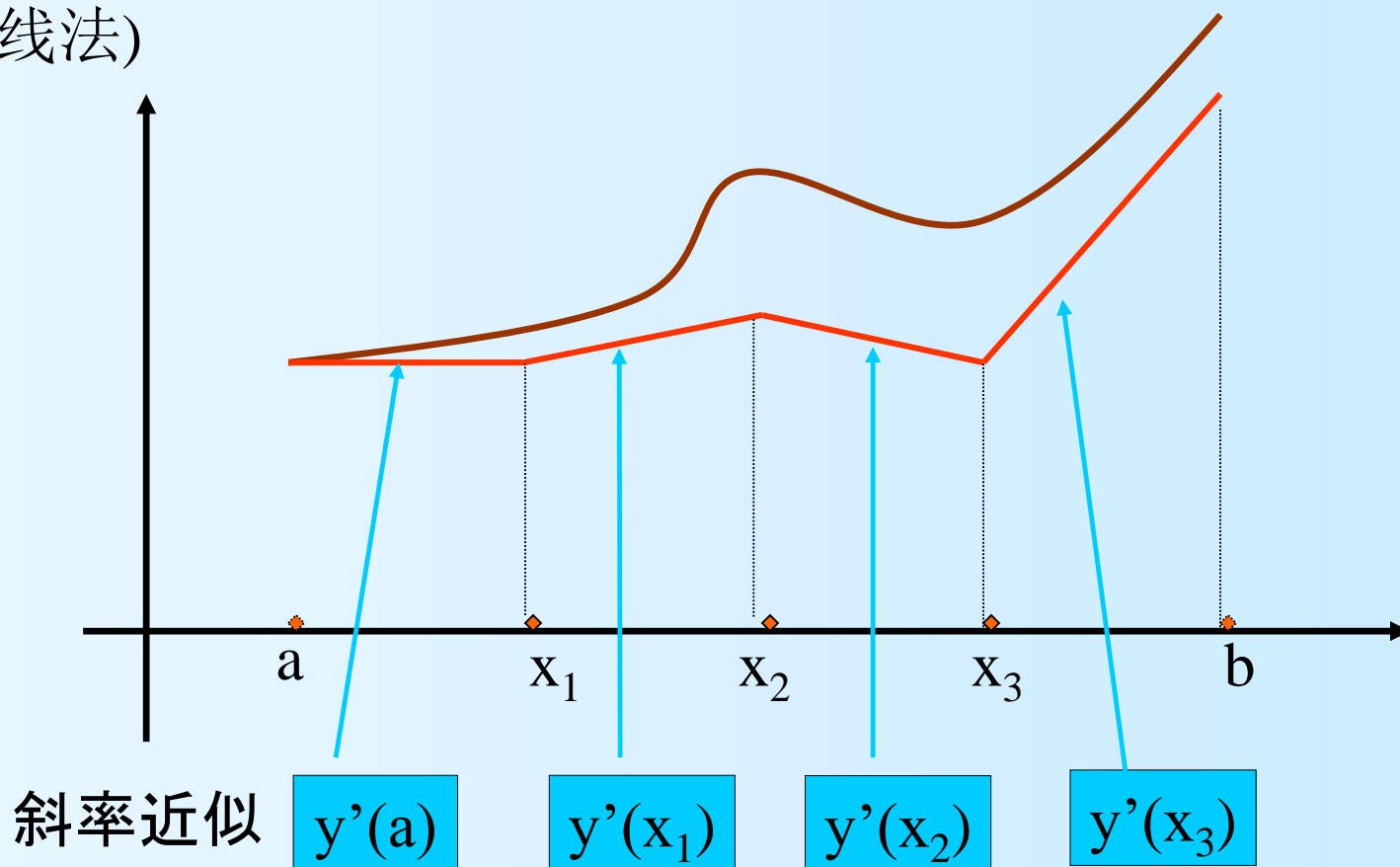
$$a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$$

计算 $y(x_k)$  ( $k>0$ )的近似值 $y_k$ 。



## *Euler*法(折线法)

折线  
法的  
几何  
构造  
I



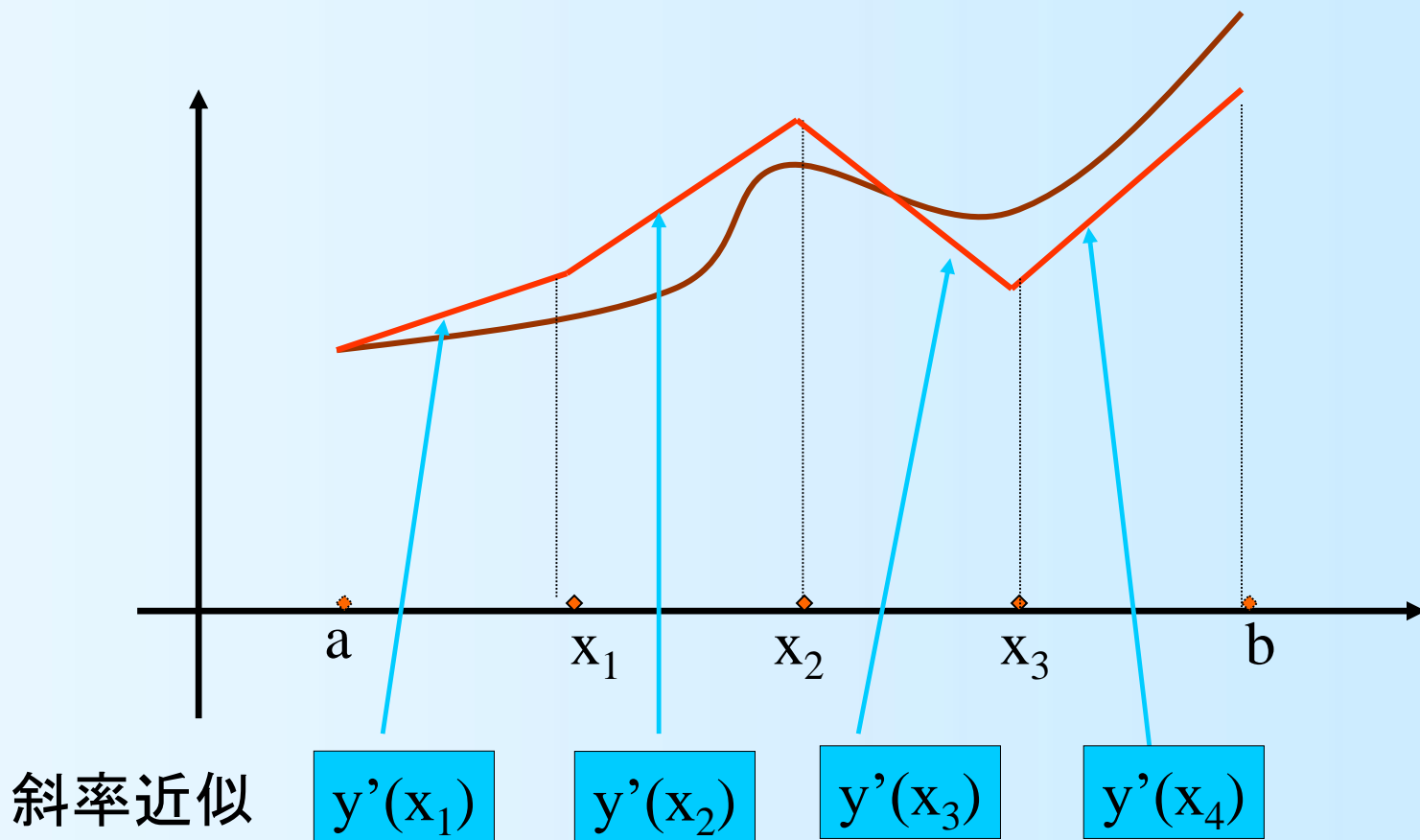
前进*Euler*法的计算公式

$$y_{i+1} = y_i + hf(x_i, y_i) \\ (i = 0, 1, 2, \dots, n-1)$$

显式公式



折线法的几何构造 II



后退Euler法的公式

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1})$$
$$(i = 0, 1, 2, \dots, n-1)$$

隐式公式

- 2阶龙格—库塔格式

**单步法**：即利用前一个节点的函数值 $y_i$ ，计算后一个节点的函数值 $y_{i+1}$ 。

目的：建立高精度的单步递推格式。

单步递推法的**基本思想**是从 $(x_i, y_i)$ 点出发，以**某一斜率**沿直线达到 $(x_{i+1}, y_{i+1})$ 点。欧拉法及其各种变形所能达到的最高精度为2阶。

考察改进的欧拉法,

$$\begin{cases} \bar{y}_k = y_{k-1} + hf(x_{k-1}, y_{k-1}) \\ y_k = y_{k-1} + \frac{h}{2}[f(x_{k-1}, y_{k-1}) + f(x_k, \bar{y}_k)] \end{cases}$$

可将其改写为:

$$\begin{cases} y_{i+1} = y_i + h \left[ \frac{1}{2} K_1 + \frac{1}{2} K_2 \right] \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + h, y_i + hK_1) \\ y_0 = y(x_0) \end{cases}$$

斜率一定取 $K_1$   $K_2$   
的平均值吗?

步长一定是一个 $h$ 吗?

将改进欧拉法推广为:

$$\begin{cases} y_{i+1} &= y_i + h[\lambda_1 K_1 + \lambda_2 K_2] \\ K_1 &= f(x_i, y_i) \\ K_2 &= f(x_i + ph, y_i + phK_1) \end{cases}$$

首先希望能确定系数  $\lambda_1$ 、 $\lambda_2$ 、 $p$ ，使得到的算法格式有2阶精度，即在  $y_i = y(x_i)$  的前提假设下，使得

$$R_i = y(x_{i+1}) - y_{i+1} = O(h^3)$$

*Step 1:* 将  $K_2$  在  $(x_i, y_i)$  点作 Taylor 展开

$$\begin{aligned} K_2 &= f(x_i + ph, y_i + phK_1) \\ &= f(x_i, y_i) + phf_x(x_i, y_i) + phK_1f_y(x_i, y_i) + O(h^2) \\ &= y'(x_i) + phy''(x_i) + O(h^2) \end{aligned}$$

$$y''(x) = \frac{d}{dx} f(x, y) = f_x(x, y) + f_y(x, y) \frac{dy}{dx} = f_x(x, y) + f_y(x, y)f(x, y)$$

*Step 2:* 将  $K_2$  代入第1式, 得到

$$\begin{aligned} y_{i+1} &= y_i + h \left\{ \lambda_1 y'(x_i) + \lambda_2 [y'(x_i) + phy''(x_i) + O(h^2)] \right\} \\ &= y_i + (\lambda_1 + \lambda_2)hy'(x_i) + \lambda_2 ph^2 y''(x_i) + O(h^3) \end{aligned}$$

*Step 3:* 将  $y_{i+1}$  与  $y(x_{i+1})$  在  $x_i$  点的泰勒展开作比较

$$y_{i+1} = y_i + (\lambda_1 + \lambda_2)hy'(x_i) + \lambda_2 ph^2 y''(x_i) + O(h^3)$$

$$y(x_{i+1}) = y(x_i) + hy'(x_i) + \frac{h^2}{2} y''(x_i) + O(h^3)$$

要求  $R_i = y(x_{i+1}) - y_{i+1} = O(h^3)$ ，则必须有：

$$\lambda_1 + \lambda_2 = 1, \quad \lambda_2 p = \frac{1}{2}$$

这里有3个未知数，2个方程。

存在无穷多个解。所有满足上式的格式统称为2阶龙格 - 库塔格式。

注意到， $p=1$ ， $\lambda_1 = \lambda_2 = \frac{1}{2}$  就是改进的欧拉法。

- 高阶龙格—库塔格式

问题: 为获得更高的精度, 应该如何进一步推广?

$$\left\{ \begin{array}{l} y_{i+1} = y_i + h[\lambda_1 K_1 + \lambda_2 K_2 + \dots + \lambda_m K_m] \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + \alpha_2 h, y_i + \beta_{21} h K_1) \\ K_3 = f(x_i + \alpha_3 h, y_i + \beta_{31} h K_1 + \beta_{32} h K_2) \\ \dots\dots \\ K_m = f(x_i + \alpha_m h, y_i + \beta_{m1} h K_1 + \beta_{m2} h K_2 + \dots + \beta_{m\ m-1} h K_{m-1}) \end{array} \right.$$

其中  $\lambda_i$  ( $i = 1, \dots, m$ ),  $\alpha_i$  ( $i = 2, \dots, m$ ) 和  $\beta_{ij}$  ( $i = 2, \dots, m; j = 1, \dots, i-1$ ) 均为待定系数, 确定这些系数的步骤与前面相似。

### 3阶龙格-库塔法

$$\left\{ \begin{array}{lcl} y_{i+1} & = & y_i + \frac{h}{6}(K_1 + 4K_2 + K_3) \\ K_1 & = & f(x_i, y_i) \\ K_2 & = & f(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1) \\ K_3 & = & f(x_i + h, y_i - hK_1 + 2hK_2) \end{array} \right.$$



最常用为4阶经典龙格-库塔法

*/\* Classical Runge-Kutta Method \*/ :*

$$\left\{ \begin{array}{lcl} y_{i+1} & = & y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 & = & f(x_i, y_i) \\ K_2 & = & f(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1) \\ K_3 & = & f(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2) \\ K_4 & = & f(x_i + h, y_i + hK_3) \end{array} \right.$$

注:

👉 龙格-库塔法的主要运算在于计算  $K_i$  的值, 即计算  $f$  的值。*Butcher* 于1965年给出了计算量与可达到的最高精度阶数的关系:

每步须算 $K_i$ 的个数	2	3	4	5	6	7	$n \geq 8$
可达到的最高精度	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^4)$	$O(h^5)$	$O(h^6)$	$O(h^{n-2})$

👉 由于龙格-库塔法的导出基于泰勒展开, 故精度主要受解函数的光滑性影响。对于光滑性不太好的解, 最好采用低阶算法而将步长 $h$ 取小。

例：使用高阶*R-K*方法计算初值问题

$$\begin{cases} y' = y^2 & 0 \leq x \leq 0.5 \\ y(0) = 1 \end{cases} \quad \text{取 } h = 0.1.$$

解：（1）使用三阶*R-K*方法

$$n = 1 \text{ 时 } K_1 = y_0^2 = 1$$

$$K_2 = (y_0 + \frac{0.1}{2} K_1)^2 = 1.1025$$

$$K_3 = (y_0 + 0.1(2K_2 - K_1))^2 = 1.2555$$

$$y_1 = y_0 + \frac{0.1}{6} (K_1 + 4K_2 + K_3) = 1.1111$$

其余结果如下:

$i$	$x_i$	$k_1$	$k_2$	$k_3$	$y_i$
1.0000	0.1000	1.0000	1.1025	1.2555	1.1111
2.0000	0.2000	1.2345	1.3755	1.5945	1.2499
3.0000	0.3000	1.5624	1.7637	2.0922	1.4284
4.0000	0.4000	2.0404	2.3423	2.8658	1.6664
5.0000	0.5000	2.7768	3.2587	4.1634	1.9993

(2) 如果使用四阶*R-K*方法

$$n = 1 \text{ 时 } K_1 = y_0^2 = 1$$

$$K_2 = (y_0 + \frac{0.1}{2} K_1)^2 = 1.1025$$

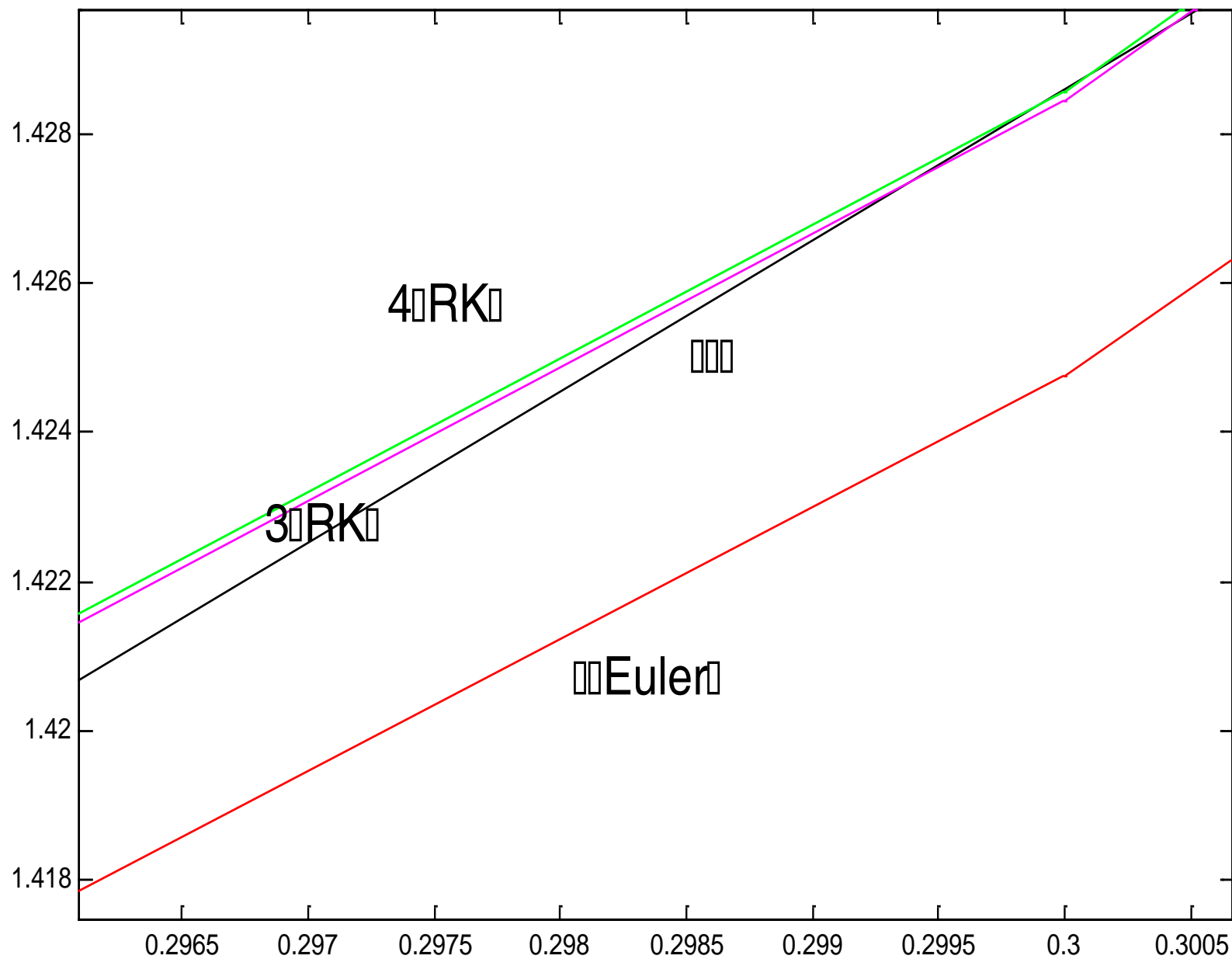
$$K_3 = (y_0 + \frac{0.1}{2} K_2)^2 = 1.1133$$

$$K_4 = (y_0 + 0.1 K_3)^2 = 1.2351$$

$$y_1 = y_0 + \frac{0.1}{6} (K_1 + 2K_2 + 2K_3 + K_4) = 1.1111$$

其余结果如下:

$i$	$x_i$	$k_1$	$k_2$	$k_3$	$k_4$	$y_i$
1.0000	0.1000	1.0000	1.1025	1.1133	1.2351	1.1111
2.0000	0.2000	1.2346	1.3756	1.3921	1.5633	1.2500
3.0000	0.3000	1.5625	1.7639	1.7908	2.0423	1.4286
4.0000	0.4000	2.0408	2.3428	2.3892	2.7805	1.6667
5.0000	0.5000	2.7777	3.2600	3.3476	4.0057	2.0000



# 龙格—库塔方法

## 三阶Runge-Kutta方法

$$\left\{ \begin{array}{l} y_{i+1} = y_i + \frac{h}{6}(K_1 + 4K_2 + K_3) \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1) \\ K_3 = f(x_i + h, y_i + h(2K_2 - K_1)) \\ y_0 = y(x_0) \end{array} \right.$$

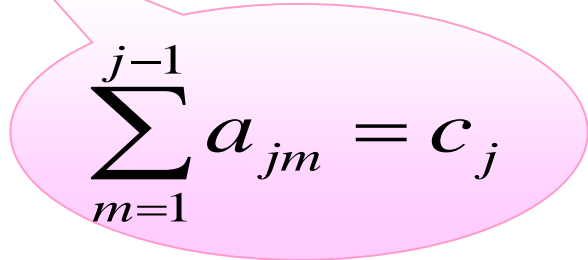


## 四阶(经典)*Runge-Kutta*方法

$$\left\{ \begin{array}{l} y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_i, y_i) \\ K_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right) \\ K_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2\right) \\ K_4 = f(x_i + h, y_i + hK_3) \\ y_0 = y(x_0) \end{array} \right.$$

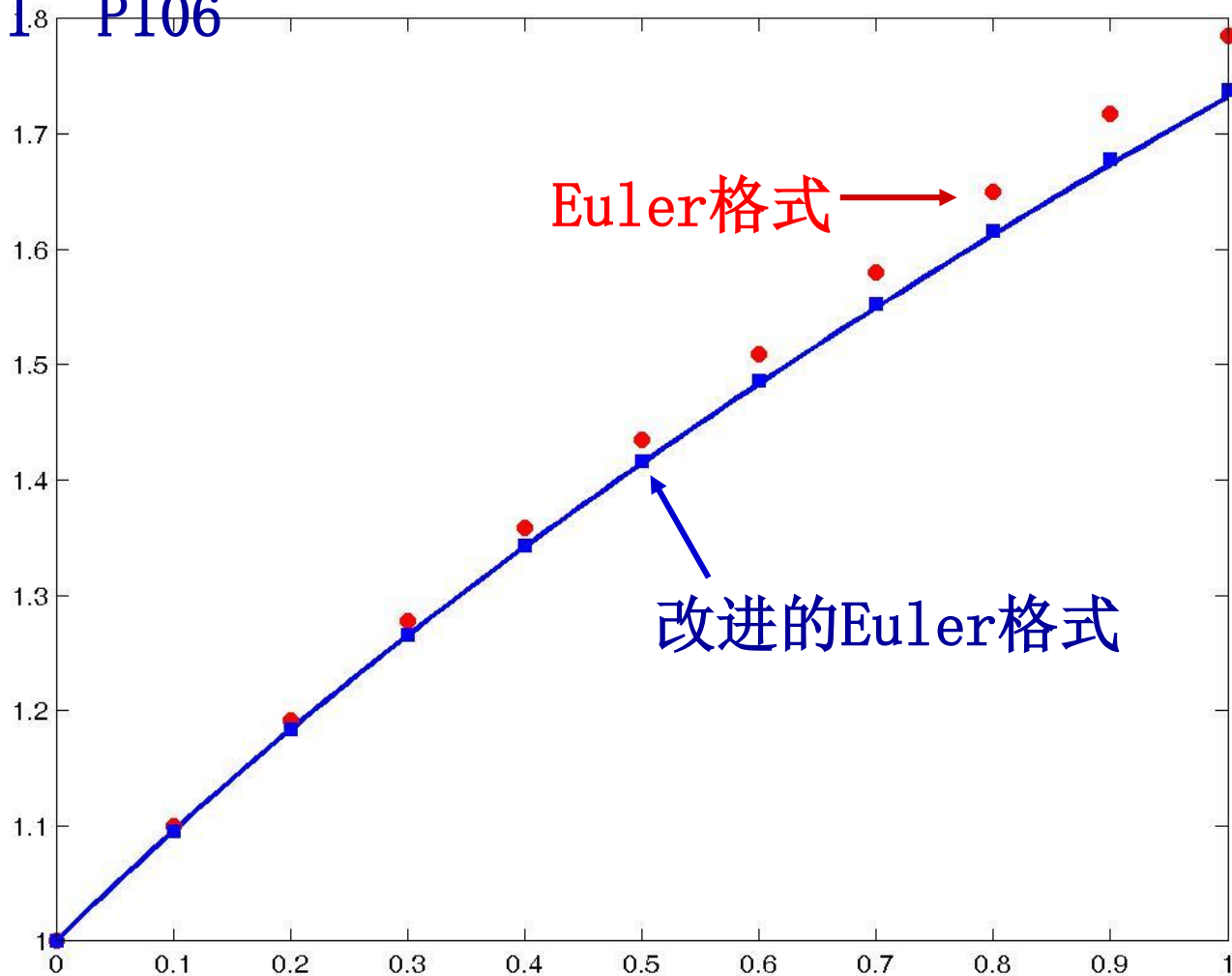
## n级显式Runge-Kutta方法

$$\left\{ \begin{array}{l} y_{i+1} = y_i + h \sum_{j=1}^n b_j k_j, \\ k_1 = f(x_i, y_i), \\ k_j = f(x_i + c_j h, y_i + h \sum_{m=1}^{j-1} a_{jm} k_m), j = 2, 3, \dots, n \end{array} \right.$$


$$\sum_{m=1}^{j-1} a_{jm} = c_j$$

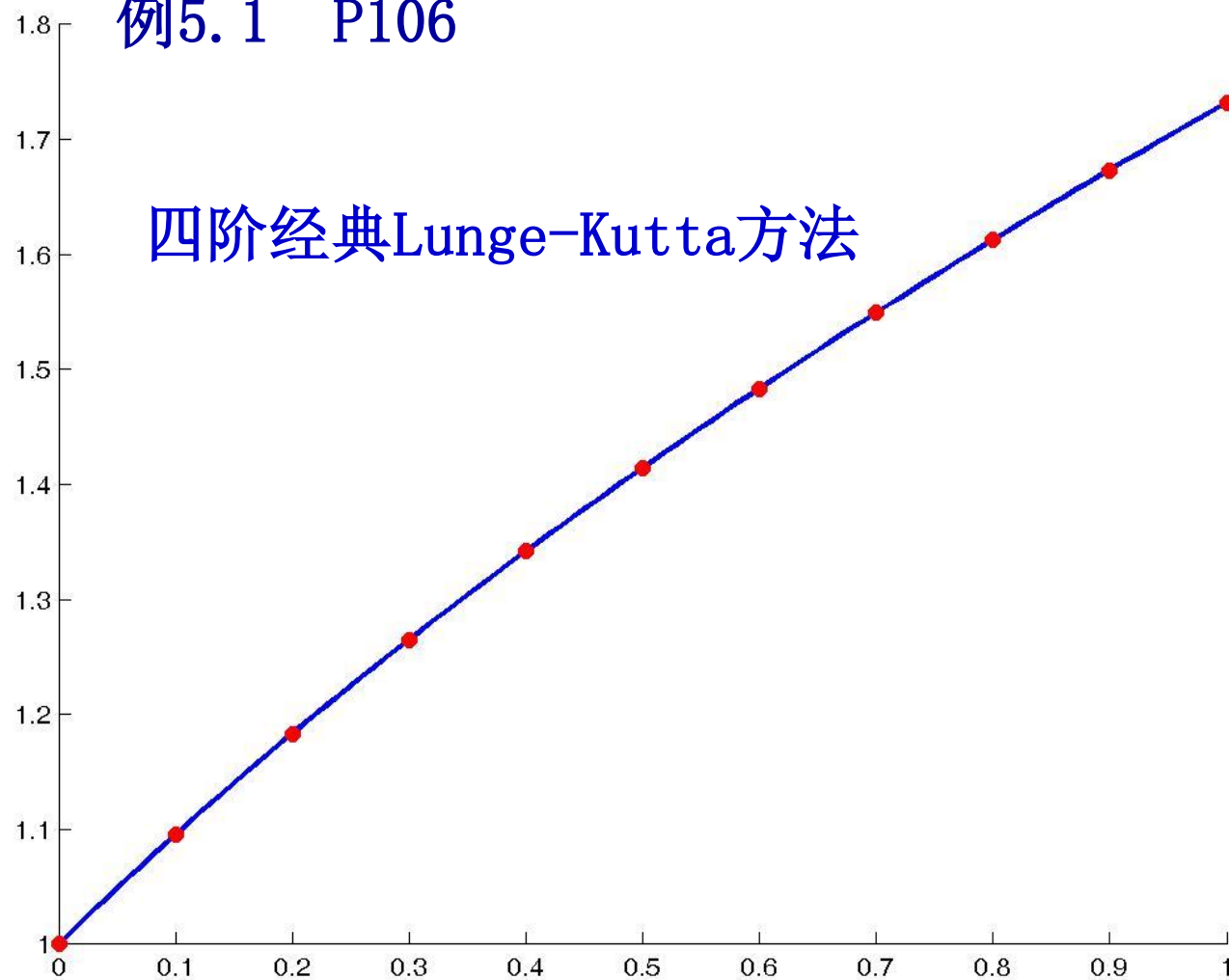
# 几种方法的数值计算

例5.1 P106



## 例5.1 P106

四阶经典Lunge-Kutta方法



# 几种方法的结果与误差

x	精确值	Euler方法	改进Euler方法	四阶Lunge-Kutta	Euler方法误差	改进Euler误差	四阶Lunge-Kutta误差
0	1.0000000	1.0000000	1.0000000	1.0000000	0.0000000	0.0000000	0.0000000
0.1	1.0954451	1.1000000	1.0959091	1.0954455	0.0045549	0.0004640	0.0000004
0.2	1.1832160	1.1918182	1.1840966	1.1832167	0.0086022	0.0008806	0.0000008
0.3	1.2649111	1.2774378	1.2662014	1.2649122	0.0125268	0.0012903	0.0000012
0.4	1.3416408	1.3582126	1.3433602	1.3416424	0.0165718	0.0017194	0.0000016
0.5	1.4142136	1.4351329	1.4164019	1.4142156	0.0209194	0.0021884	0.0000020
0.6	1.4832397	1.5089663	1.4859556	1.4832422	0.0257266	0.0027159	0.0000025
0.7	1.5491933	1.5803382	1.5525141	1.5491965	0.0311449	0.0033208	0.0000031
0.8	1.6124515	1.6497834	1.6164748	1.6124553	0.0373319	0.0040232	0.0000038
0.9	1.6733201	1.7177793	1.6781664	1.6733247	0.0444593	0.0048463	0.0000046
1	1.7320508	1.7847708	1.7378674	1.7320564	0.0527200	0.0058166	0.0000056

## 参考程序-Euler

```
x=0:0.01:1;
y=sqrt(1+2.*x);
a=0.0;b=1.0;n=10;
h=(b-a)/n;
x0=a:h:b;y0(1)=1.0;
for k=1:10
    y0(k+1)=y0(k)+h*(y0(k)-2*x0(k)/y0(k));
end
for i=1:10
    y1(1)=1.0;
    y1(i+1)=y1(i)+h*(y1(i)-2*x0(i)/y1(i));
    y1(i+1)=y1(i)+h*(y1(i)-2*x0(i)/y1(i)+y1(i+1)-
                    2*x0(i+1)/y1(i+1))/2;
end
plot(x,y,'b');
hold on;
plot(x0,y0,'or');
hold on;
plot(x0,y1,'*');
```

## 参考程序-Lunge\_Kutta

```
x=0:0.01:1;  
y=sqrt(1+2.*x);  
a=0.0;b=1.0;n=10;  
h=(b-a)/n;  
x0=a:h:b;  
y0(1)=1.0;  
for k=1:10  
    k1=y0(k)-2*x0(k)/y0(k);  
    k2=y0(k)+h*k1/2-(2*x0(k)+h)/(y0(k)+h*k1/2);  
    k3=y0(k)+h*k2/2-(2*x0(k)+h)/(y0(k)+h*k2/2);  
    k4=y0(k)+h*k3-2*(x0(k)+h)/(y0(k)+h*k3);  
    y0(k+1)=y0(k)+h*(k1+2*k2+2*k3+k4)/6;  
end  
hold on;  
plot(x,y,'b');  
hold on;  
plot(x0,y0,'or');
```

## 四、收敛性和稳定性



# 单步方法的收敛性与稳定性

/\* Convergency and Stability \*/

## ➤ 收敛性 /\* Convergency \*/

**定义** 若某算法对于任意固定的  $x = x_i = x_0 + i h$ , 当  $h \rightarrow 0$  (同时  $i \rightarrow \infty$ ) 时有  $y_i \rightarrow y(x_i)$ , 则称该算法是收敛的。

**例:** 就初值问题  $\begin{cases} y' = \lambda y \\ y(0) = y_0 \end{cases}$  考察欧拉显式格式的收敛性。

**解：**该问题的精确解为  $y(x) = y_0 e^{\lambda x}$

欧拉公式为  $y_{i+1} = y_i + h\lambda y_i = (1 + \lambda h)y_i$

对任意固定的  $x = x_i = i h$ ，有

$$\lim_{h \rightarrow 0} (1 + \lambda h)^{1/\lambda h} = e$$

$$y_i = y_0 (1 + \lambda h)^{x_i/h} = y_0 [(1 + \lambda h)^{1/\lambda h}]^{\lambda x_i}$$

$$\rightarrow y_0 e^{\lambda x_i} = y(x_i)$$

## ➤ 稳定性 /\* Stability \*/

例：考察初值问题  $\begin{cases} y'(x) = -30y(x) \\ y(0) = 1 \end{cases}$  在区间  $[0, 0.5]$  上的解. 分别用欧拉显、隐式格式和改进的欧拉格式计算数值解。

节点 $x_i$	欧拉显式	欧拉隐式	改进欧拉法	精确解 $y = e^{-30x}$
0.0	1.0000	1.0000	1.0000	1.0000
0.1	-2.0000	$2.5000 \times 10^{-1}$	2.5000	$4.9787 \times 10^{-2}$
0.2	4.0000	$6.2500 \times 10^{-2}$	6.2500	$2.4788 \times 10^{-3}$
0.3	-8.0000	$1.5625 \times 10^{-2}$	$1.5626 \times 10^1$	$1.2341 \times 10^{-4}$
0.4	$1.6000 \times 10^1$	$3.9063 \times 10^{-3}$	$3.9063 \times 10^1$	$6.1442 \times 10^{-6}$
0.5	$-3.2000 \times 10^1$	$9.7656 \times 10^{-4}$	$9.7656 \times 10^1$	$3.0590 \times 10^{-7}$

**定义** 若某算法在计算过程中任一步产生的误差在以后的计算中都**逐步衰减**，则称该算法是**绝对稳定的** ***/\*absolutely stable \*/***。

一般分析时为简单起见，只考虑 ***/\* test equation \*/***

常数，可以是复数

$$y' = \lambda y$$

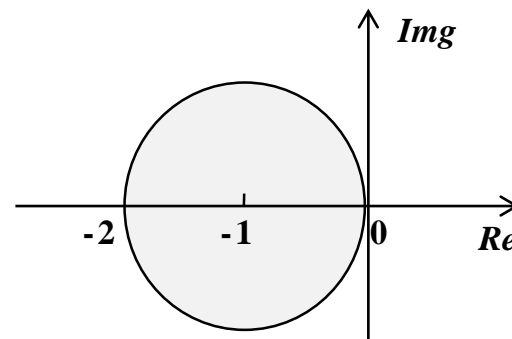
当步长取为  $h$  时，将某算法应用于上式，并假设只在初值产生误差  $\varepsilon_0 = y_0 - \bar{y}_0$ ，则若此误差以后逐步衰减，就称该算法相对于  $\overline{h} = \lambda h$  绝对稳定， $\overline{h}$  的全体构成绝对稳定区域。

我们称算法 **A** 比算法 **B** 稳定，就是指 **A** 的绝对稳定区域比 **B** 的大。

例：考察显式欧拉法  $y_{i+1} = y_i + h\lambda y_i = (1 + h\lambda)^{i+1} y_0$

$$\varepsilon_0 = y_0 - \bar{y}_0 \rightarrow \bar{y}_{i+1} = (1 + \bar{h})^{i+1} \bar{y}_0$$

$$\rightarrow \varepsilon_{i+1} = y_{i+1} - \bar{y}_{i+1} = (1 + \bar{h})^{i+1} \varepsilon_0$$



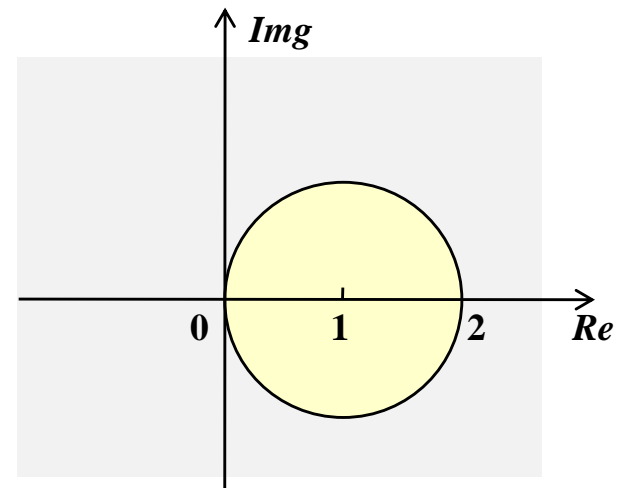
由此可见，要保证初始误差  $\varepsilon_0$  以后逐步衰减， $\bar{h} = \lambda h$

必须满足：  $|1 + \bar{h}| < 1$

例：考察隐式欧拉法  $y_{i+1} = y_i + h\lambda y_{i+1}$

$$y_{i+1} = \left( \frac{1}{1 - \bar{h}} \right) y_i$$

$$\rightarrow \varepsilon_{i+1} = \left( \frac{1}{1 - \bar{h}} \right)^{i+1} \varepsilon_0$$



可见绝对稳定区域为： $|1 - \bar{h}| > 1$

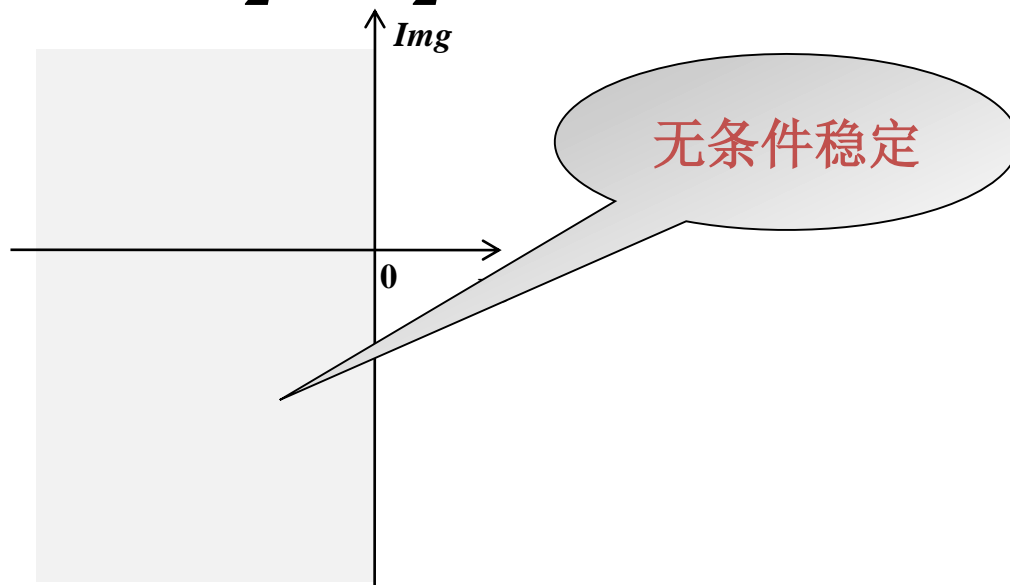
注：一般来说，隐式欧拉法的绝对稳定性比同阶的显式法的好。

## 例：隐式龙格-库塔法

$$\begin{cases} y_{i+1} = y_i + h[\lambda_1 K_1 + \dots + \lambda_m K_m] \\ K_j = f(x_i + \alpha_j h, y_i + \beta_{j1} h K_1 + \dots + \beta_{jm} h K_m) \\ (j = 1, \dots, m) \end{cases}$$

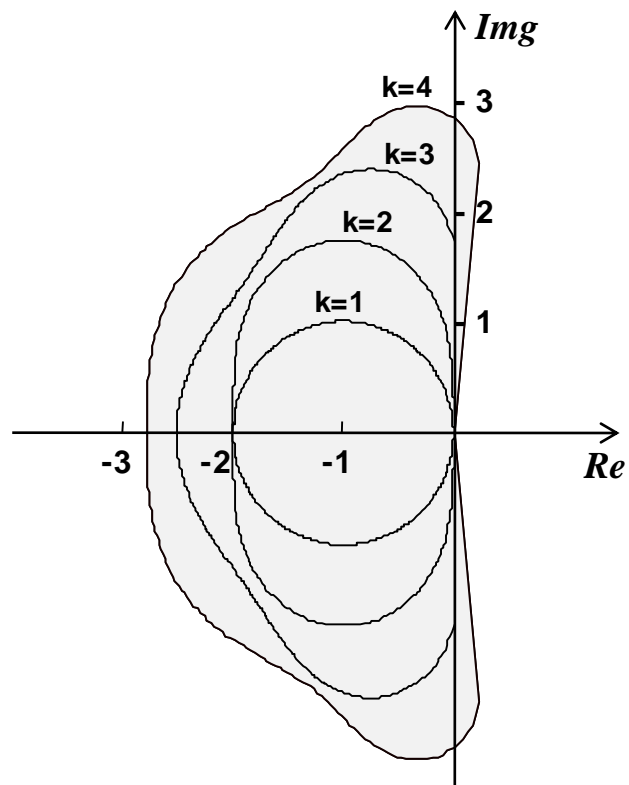
其中2阶方法  $\begin{cases} y_{i+1} = y_i + hK_1 \\ K_1 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1) \end{cases}$  的绝对稳定

区域为





而显式 1~4 阶方法的绝对稳定区域为



## 五、亚当姆斯方法

- 一般线性多步法

- 由于在计算 $y_{n+1}$ 时, 已经知道 $y_n, y_{n-1}, \dots$ , 及 $f(x_n, y_n), f(x_{n-1}, y_{n-1}), \dots$ , 利用这些值构造出精度高、计算量小的差分公式就是线性多步法。

- 即用若干节点处的  $y$  及  $y'$  值的线性组合来近似  $y(x_{i+1})$ 。其通式可写为:

$$f_j = f(x_j, y_j)$$

$$y_{i+1} = \alpha_0 y_i + \alpha_1 y_{i-1} + \dots + \alpha_k y_{i-k} + h(\beta_{-1} f_{i+1} + \beta_0 f_i + \beta_1 f_{i-1} + \dots + \beta_k f_{i-k})$$

当  $\beta_{-1} \neq 0$  时, 为隐式公式;  $\beta_{-1} = 0$  则为显式公式。

## ➤ 基于数值积分的构造法



将  $y' = f(x, y)$  在  $[x_i, x_{i+1}]$  上积分，得到

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x)) dx$$

只要近似地算出右边的积分  $I_k \approx \int_{x_i}^{x_{i+1}} f(x, y(x)) dx$ ，则可通过  $y_{i+1} = y_i + I_k$  近似  $y(x_{i+1})$ 。而选用不同近似式  $I_k$ ，可得到不同的计算公式。



## 亚当姆斯(Adams)方法


### (一)显式Adams方法

对所讨论的微分方程  $y' = f(x, y)$  两边从  $x_n$  到  $x_{n+1}$  积分:

$$\int_{x_n}^{x_{n+1}} y'(x) dx = \int_{x_n}^{x_{n+1}} f(x, y(x)) dx$$

得到:  $y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \quad (5.1)$

给定步长  $h$ , 若我们已经求出初值问题(1.1)的解  $y(x)$  在等距点  $x_m = x_0 + mh$  ( $x_0 = a, m=0, 1, \dots, n$ ) 处的近似值  $y_0, y_1, \dots, y_n$ .



我们以  $f_m = f(x_m, y_m)$ ,  $m=0, 1, \dots, n$

作为  $f(x_m, y(x_m))$  的近似值, 用经过  $k+1$  个点

$$(x_n, f_n), (x_{n-1}, f_{n-1}), \dots, (x_{n-k}, f_{n-k})$$

的插值多项式  $P_k(x)$  ( $k \leq n$ ) 作为  $f(x, y(x))$  在  $x_n$  与  $x_{n+1}$  之间的近似式(外插!), 并将(5.1)式换成

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} P_k(x) dx. \quad (5.2)$$

若取  $P_k(x)$  为 Newton 后差插值多项式

- 亚当姆斯显式公式 /\* Adams explicit formulae \*/

利用  $k+1$  个节点上的被积函数值  $f_i, f_{i-1}, \dots, f_{i-k}$  构造  $k$  阶牛顿后插多项式  $N_k(x_i + th)$ ,  $t \in [0, 1]$ , 有

$$\int_{x_i}^{x_{i+1}} f(x, y(x)) dx = \int_0^1 N_k(x_i + th) h dt + \int_0^1 R_k(x_i + th) h dt$$

Newton插值余项

$$\longrightarrow y_{i+1} = y_i + h \int_0^1 N_k(x_i + th) dt \quad /* \text{显式计算公式} */$$

$$\text{局部截断误差为: } R_i = y(x_{i+1}) - y_{i+1} = h \int_0^1 R_k(x_i + th) dt$$

例：  $k=1$  时有

$$N_1(x_i + th) = f_i + t \nabla f_i = f_i + t(f_i - f_{i-1})$$


$$\rightarrow y_{i+1} = y_i + h \int_0^1 [f_i + t(f_i - f_{i-1})] dt = y_i + \frac{h}{2}(3f_i - f_{i-1})$$

$$R_i = h \int_0^1 \frac{d^2 f(\xi_x, y(\xi_x))}{dx^2} \frac{1}{2!} th(t+1)h dt = \frac{5}{12} h^3 y'''(\xi_i)$$



注：一般有  $\mathbf{R}_i = \mathbf{B}_k \mathbf{h}^{k+2} \mathbf{y}^{(k+2)}(\xi_i)$ ，其中  $\mathbf{B}_k$  与  $\mathbf{y}_{i+1}$  计算公式中  $f_i, \dots, f_{i-k}$  各项的系数均可查表得到。

[illegible]



$k=1$ 时, 我们得到二步公式, 是二阶方法.

---

$$y_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1})$$

当 $k=3$ 时, 得到四步公式, 是四阶方法.

$$y_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}).$$

多步方法不是自开始的, 需要用另外的方法求出前几个值.



常用的是  $k = 3$  的4阶亚当姆斯显式公式

$$y_{i+1} = y_i + \frac{h}{24} (55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3})$$



## (二) 隐式Adams方法

---

因为, 显式Adams方法用 $k+1$ 个点:

$$(x_n, f_n), (x_{n-1}, f_{n-1}), \dots, (x_{n-k}, f_{n-k})$$

的插值公式 $P_k(x)$ 代替 $(x_n, x_{n+1})$ 上的  $f(x, y(x))$ , 因而是外推.

为提高精度, 取 $k+1$ 个点:

$$(x_{n+1}, f_{n+1}), (x_n, f_n), \dots, (x_{n-k+1}, f_{n-k+1})$$

构造的插值公式 $P_k(x)$ 代替 $(x_n, x_{n+1})$ 上的  $f(x, y(x))$ , 为内插.

- 亚当姆斯隐式公式 /\* Adams implicit formulae \*/

利用  $k+1$  个节点上的被积函数值  $f_{i+1}, f_i, \dots, f_{i-k+1}$  构造  $k$  阶牛顿前插多项式。与显式多项式完全类似地可得到一系列隐式公式，并有  $\mathbf{R}_i = \mathbf{B}_k \tilde{\mathbf{h}}^{k+2} \mathbf{y}^{(k+2)}(\eta_i)$ ，其中  $\tilde{\mathbf{B}}_k$  与  $f_{i+1}, f_i, \dots, f_{i-k+1}$  的系数亦可查表得到。

$k$	$f_{i+1}$	$f_i$	$f_{i-1}$	$f_{i-2}$	$\dots$	$\tilde{B}_k$
0	1					$-\frac{1}{2}$
1	$\frac{1}{2}$	$\frac{1}{2}$				$-\frac{1}{12}$
2	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$			$-\frac{1}{24}$
3	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$		$-\frac{19}{720}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

小于  $B_k$



常用的是  $k = 3$  的4阶亚当姆斯隐式公式

$$y_{i+1} = y_i + \frac{h}{24}(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2})$$

较同阶显式稳定



k=3时, 得到三步公式, 方法是四阶的:

---

$$y_{n+1} = y_n + \frac{h}{24} [9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}]$$

一般, **k**步的显式Adams方法具有k阶精度,

**k**步的隐式Adams方法具有k+1阶精度.

## ■ 亚当姆斯预测-校正系统/\* Adams predictor-corrector system \*/

Step 1: 用Runge-Kutta 法计算前  $k$  个初值;

Step 2: 用Adams 显式计算预测值;

Step 3: 用同阶Adams 隐式计算校正值。

注意：三步所用公式的精度必须相同。通常用经典Runge-Kutta 法配合4阶Adams 公式。

4阶Adams显式公式的截断误差为  $y(x_{i+1}) - \bar{y}_{i+1} = \frac{251}{720} h^5 y^{(5)}(\xi_i)$

4阶Adams隐式公式的截断误差为  $y(x_{i+1}) - y_{i+1} = -\frac{19}{720} h^5 y^{(5)}(\eta_i)$

当  $h$  充分小时, 可近似认为  $\xi_i \approx \eta_i$ , 则:  $\frac{y(x_{i+1}) - \bar{y}_{i+1}}{y(x_{i+1}) - y_{i+1}} \approx -\frac{251}{19}$

→ 
$$\left. \begin{aligned} y(x_{i+1}) &\approx \bar{y}_{i+1} + \frac{251}{270} (y_{i+1} - \bar{y}_{i+1}) \\ y(x_{i+1}) &\approx y_{i+1} - \frac{19}{270} (y_{i+1} - \bar{y}_{i+1}) \end{aligned} \right\} \begin{array}{l} \text{外推技术} \\ /* extrapolation */ \end{array}$$



# 第五章习题

P.139. 第2题(计算 $0 < x < 0.5$ );第6题第(2)题;第9题

补充:用Taylor展开法构造

$$\begin{cases} y' = e^{-x^2} \\ y(0) = e \end{cases}$$

的二阶、三阶数值解