§7 常微分方程数值解法

- 1 Euler方法及其变形
- 2 Runge-Kutta 方法
- 3 线性多步法
- 4 预估校正公式

第七章　常微分方程数值解法

/* Numerical Methods for Ordinary Differential Equations */

考虑一阶常微分方程的初值问题 /* Initial-Value Problem */:

$$\begin{cases} \dfrac{dy}{dx} = f(x,y) \quad x \in [a,b] \\ y(a) = y_0 \end{cases}$$

只要 $f(x,y)$ 在$[a,b] \times R^1$ 上连续，且关于 y 满足 *Lipschitz* 条件，即存在与 x, y 无关的常数 L，使

$$|f(x,y_1) - f(x,y_2)| \le L|y_1 - y_2|$$

对任意定义在 $[a, b]$ 上的 $y_1(x)$ 和 $y_2(x)$ 都成立，则上述 IVP存在唯一解。

第七章　常微分方程数值解法

/* Numerical Methods for Ordinary Differential Equations */

考虑一阶常微分方程的初值问题 /* Initial-Value Problem */:

$$\begin{cases} \dfrac{dy}{dx} = f(x,y) \quad x \in [a,b] \\ y(a) = y_0 \end{cases}$$

本章任务：计算出解函数 $y(x)$ 在一系列节点

$a = x_0 < x_1 < \cdots < x_n = b,\ y_n \approx y(x_n)\ (n = 1, \ldots, N)$

处的近似值 。

➢ 常微分方程数值方法的基本思想

微分方程数值解法，其实质是求出方程的解$y(x)$在一系列离散点上的近似值。则微分方程数值解的基本思想是：求解区间和方程离散化。

➢ 求解区间离散化

将求解区间$[a,b]$离散化，是在$[a,b]$上插入一系列的分点$\{x_k\}$，使$a = x_0 < x_1 < \ldots < x_n < x_{n+1} < \ldots < x_N = b$

记$h_n = x_{n+1} - x_n, (n = 0,1,\ldots,N-1)$称为步长.

一般取$\boldsymbol{h_n = h}$(常数), $h = \frac{b-a}{N}$.

节点$x_n = x_0 + nh, (n = 0,1,2,\ldots,N)$称为等步长节点 。

➢ 将微分方程离散化

将微分方程离散化，通常有下述方法：

（1）差商逼近法

用适当的差商逼近导数值。

（2）数值积分法

基本思想是先将问题转化为积分方程

$$y(x_m) - y(x_n) = \int_{x_n}^{x_m} f\big(x, y(x)\big)dx\ ,\ \ (y(x_0) = y_0)$$

然后将上式右端采用数值积分离散化，

从而获得原初值问题的一个离散差分格式。

（3）Taylor展开法

§ 1 欧拉方法 /* Euler's Method */

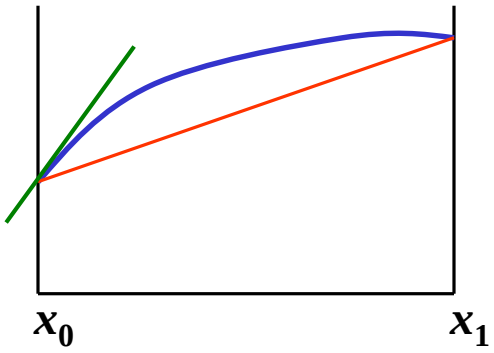
➢ 欧拉公式：

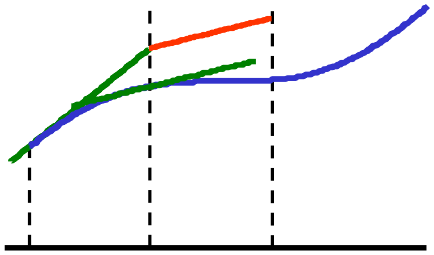
向前差商近似导数 ➡ $y'(x_0) \approx \dfrac{y(x_1)-y(x_0)}{h}$

$$y(x_1) \approx y(x_0)+hy'(x_0) = y_0 + h\,f(x_0, y_0) \xlongequal{\text{记为}} y_1$$

$$y_{n+1} = y_n + h\,f(x_n, y_n) \quad (n = 0, \ldots, N-1)$$

欧拉方法亦称为欧拉折线法 /* Euler's polygonal arc method*/





例 求初值问题

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases} \qquad (0 < x < 1)$$

解：Euler公式的具体形式为

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

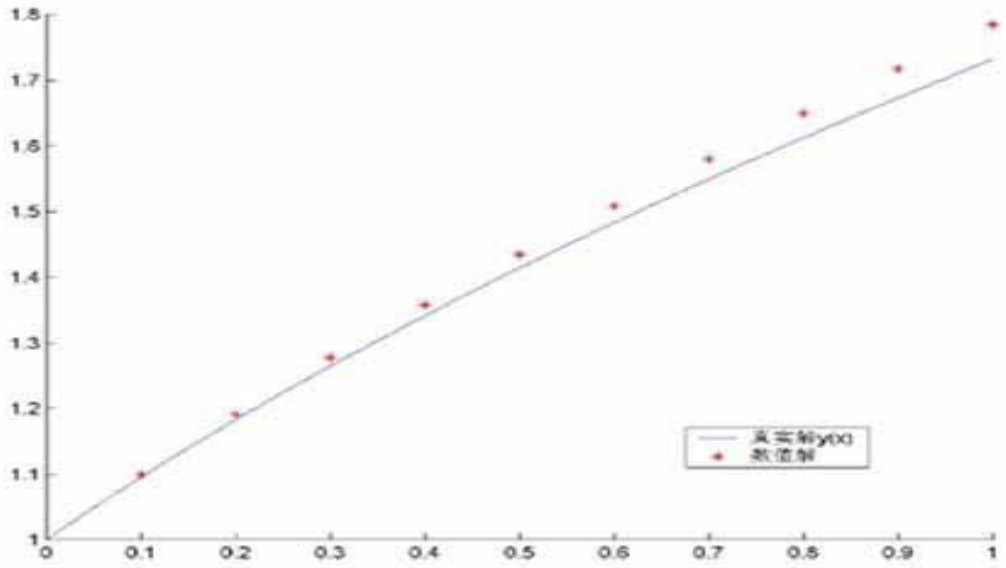
取步长$h = 0.1$。其中x_n是节点，y_n是节点上的近似值$y(x_n)$是精确值，结果见下表：

x_n	y_n	$y(x_n)$	x_n	y_n	$y(x_n)$
0.1	1.1000	1.0954	0.6	1.5090	1.4832
0.2	1.1918	1.1832	0.7	1.5803	1.5492
0.3	1.2774	1.2649	0.8	1.6498	1.6125
0.4	1.3582	1.2416	0.9	1.7178	1.6733
0.5	1.4351	1.4142	1.0	1.7848	1.7321

例 求初值问题

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases} \qquad (0 < x < 1)$$

解：

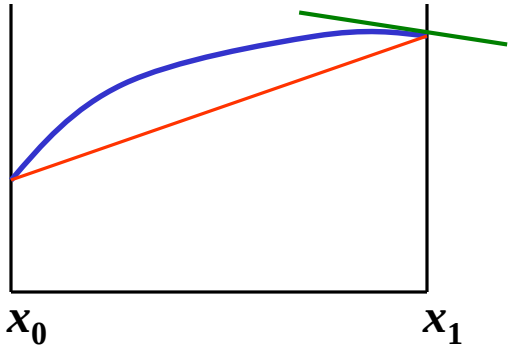


从图中可以看出，灰色连续的曲线就是初值问题的解析解 $y = \sqrt{1 + 2x}$的曲线。而星点则为数值解。

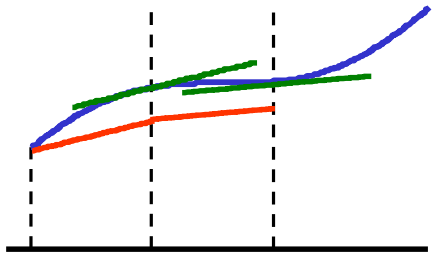
➢ 隐式欧拉法 /* implicit Euler method */

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

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



$$y_{n+1} = y_n + h\, f(x_{n+1}, y_{n+1}) \quad (n = 0, \ldots, N-1)$$



由于未知数 y_{n+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。即

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n)$$

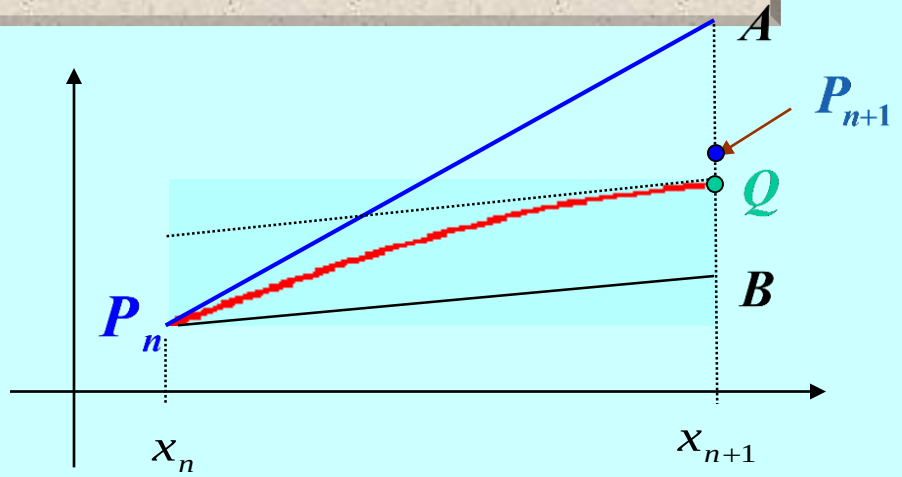
$$y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)}) \quad k = 0,1,2,\cdots$$

如果迭代过程收敛，则某步后 $y_{n+1}^{(k)}$ 就可以作为 y_{n+1}，从而进行下一步的计算。

梯形公式 /* trapezoid formula */ — 显、隐式两种算法的平均

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (n = 0, \ldots, N-1)$$

梯形方法的平均化思想
可以借助几何直观说明，见右：

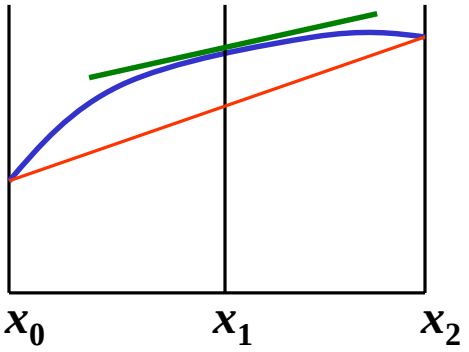


两步欧拉公式 /* midpoint formula */

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

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

$$y_{n+1} = y_{n-1} + 2h\, f(x_n, y_n) \quad n = 1, \ldots, N-1$$



需要2个初值 y_0 和 y_1 来启动递推过程，这样的算法称为两步法 /* double-step method */，而前面的三种算法都是单步法/* single-step method */。

➢ 改进欧拉法 /* modified Euler's method */

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

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

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})]$$

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f\left(x_{n+1}, y_n + h\, f(x_n, y_n)\right)\right] \quad (n = 0, \ldots, N-1)$$

$$y_p = y_n + hf(x_n, y_n),\ y_c = y_n + hf(x_{n+1}, y_p)$$

$$y_{n+1} = \frac{1}{2}(y_p + y_c)$$

注：此法亦称为预测-校正法 /* predictor-corrector method */。可以证明该算法具有 2 阶精度，同时可以看到它是个单步递推格式，比隐式公式的迭代求解过程简单。后面将看到，它的稳定性高于显式欧拉法。

例　求初值问题

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases} \qquad (0 < x < 1)$$

解：改进的Euler公式为

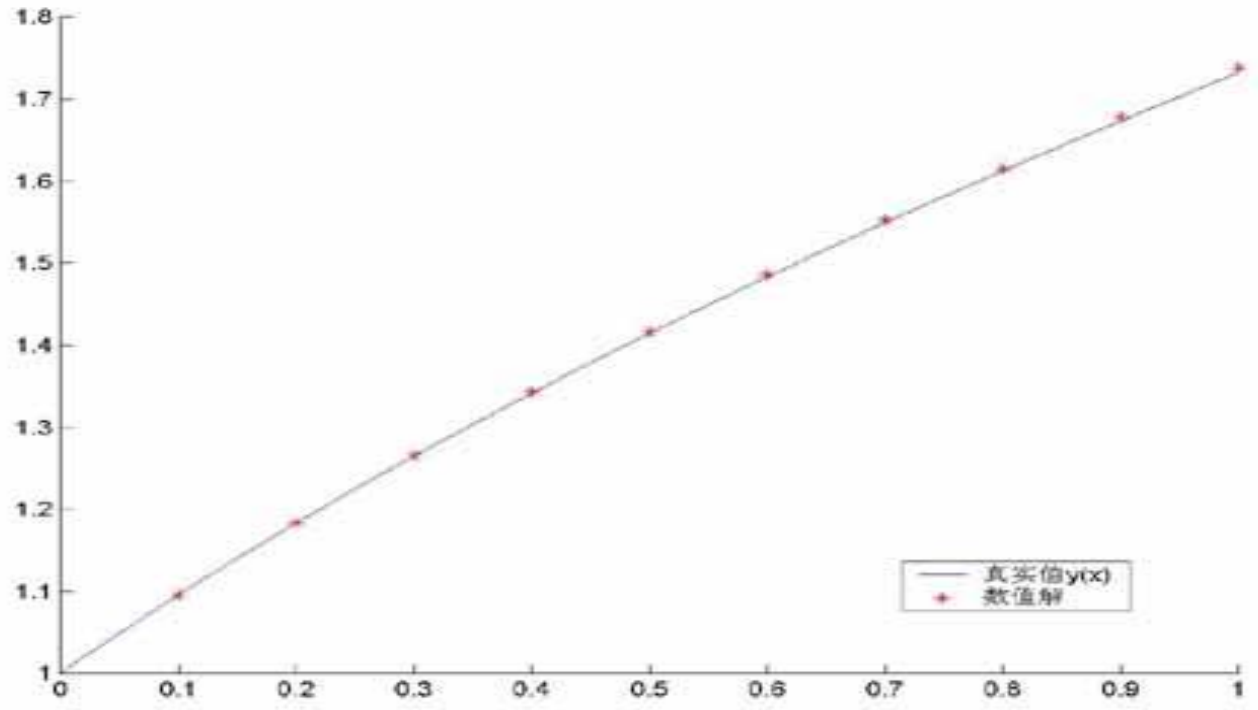
$$\begin{cases} y_p = y_n + h\left(y_n - \dfrac{2x_n}{y_n}\right) \\ y_c = y_n + h\left(y_p - \dfrac{2x_{n+1}}{y_p}\right) \\ y_{n+1} = \dfrac{1}{2}(y_p + y_c) \end{cases}$$

我们仍取步长h=0.1，

计算结果见右表：

x_n	y_n	$y(x_n)$
0.1	1.0959	1.0954
0.2	1.1841	1.1832
0.3	1.2662	1.2649
0.4	1.3434	1.3416
0.5	1.4164	1.4142
0.6	1.4860	1.4832
0.7	1.5525	1.5492
0.8	1.6153	1.6125
0.9	1.6782	1.6733
1.0	1.7379	1.7321

Matlab作图显示



从表和图可以看出，改进的**Euler**法的精度提高了不少。

➢ 局部截断误差和方法的阶

初值问题的单步法可用一般形式表示为:

φ 称为增量函数

$$y_{n+1} = y_n + h\varphi(x_n, y_n, y_{n+1}, h)$$

φ 含有 y_{n+1} 时，方法是隐式的，
φ 不含有 y_{n+1} 时，方法是显式的。

例如对欧拉法有 $\varphi(x_n, y_n, h) = f(x_n, y_n)$，

隐式欧拉法有 $\varphi(x_n, y_n, y_{n+1}, h) = f(x_{n+1}, y_{n+1})$

从 x_0 开始计算，如果考虑每一步产生的误差，直到 x_n 则有误差 $e_n = y(x_n) - y_n$，称为该方法在 x_n 的整体截断误差，分析和求得整体截断误差是复杂的。为此，我们仅考虑从 x_n 到 x_{n+1} 的局部情况，并假设 x_n 之前的计算没有误差，即 $y_n = y(x_n)$，下面给出单步法的局部截断误差概念。

定义 设 $y(x)$ 是初值问题的准确解，称

$$T_{n+1} = y(x_{n+1}) - y(x_n) - h\varphi(x_n, y(x_n), h)$$

为显式单步法的局部截断误差。

即 在假设 $y_n = y(x_n)$，即第 n 步计算是精确的前提下，考虑的截断误差 $T_{n+1} = y(x_{n+1}) - y_{n+1}$ 称为局部截断误差 /* local truncation error */。

定义 若某算法的局部截断误差为 $O(h^{p+1})$，则称该算法有 p 阶精度。

☞ 欧拉法的局部截断误差：

$$T_{n+1} = y(x_{n+1}) - y_{n+1} = [y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + O(h^3)] - [y_n + hf(x_n, y_n)]$$

$$= \frac{h^2}{2}y''(x_n) + O(h^3)$$ 欧拉法具有 1 阶精度。

☞ 两步欧拉法的局部截断误差：

$$T_{n+1} = y(x_{n+1}) - y_{n+1} = O(h^3)$$ 即两步欧拉公式具有 2 阶精度。

例 求隐式欧拉格式 $y_{n+1}=y_n+hf(x_{n+1},y_{n+1})$ 的局部截断误差。

$$T_{n+1}=y(x_{n+1})-y(x_n)-hf(x_{n+1},y(x_{n+1}))$$

$$=hy'(x_n)+\frac{h^2}{2}y''(x_n)+O(h^3)-h[y'(x_n)+hy''(x_n)+O(h^2)]$$

$$=-\frac{h^2}{2}y''(x_n)+O(h^3)$$ 具有 1 阶精度。

例 求梯形格式 $y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1})]$ 的局部截断误差。

$$T_{n+1}=y(x_{n+1})-y(x_n)-\frac{h}{2}[y'(x_n)+y'(x_{n+1})]$$

$$=hy'(x_n)+\frac{h^2}{2}y''(x_n)+\frac{h^3}{3!}y'''(x_n)$$

$$-\frac{h}{2}[y'(x_n)+y'(x_n)+hy''(x_n)+\frac{h^2}{2}y'''(x_n)]+O(h^4)$$

$$=-\frac{h^3}{12}y'''(x_n)+O(h^4)$$ 具有 2 阶精度。

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
两步欧拉公式	精度提高, 显式	多一个初值, 可能影响精度

§ 2 龙格 - 库塔法 /* Runge-Kutta Method */

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_n, y_n) 点出发，以某一斜率沿直线达到 (x_{n+1}, y_{n+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，可以将其改写为：

$$\begin{cases} y_{n+1} = y_n + h\left[\frac{1}{2}K_1 + \frac{1}{2}K_2\right] \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + h, y_n + hK_1) \end{cases}$$

显然，k_1，k_2是在点 x_n，x_{n+1} 处的斜率。以上公式用到的是两个点的斜率的加权平均，它为构造算法提供了新的途径。**Runge-Kutta**方法就是这种思想的体现和发展。

将改进欧拉法推广为：

$$\begin{cases} y_{n+1} = y_n + h[\lambda_1 K_1 + \lambda_2 K_2] \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + ph, y_n + phK_1) \end{cases} \qquad \lambda_1 + \lambda_2 = 1, \quad 0 < p \le 1$$

首先希望能确定系数 λ_1、λ_2、p，使得到的算法格式有2阶精度，即在 $y_n = y(x_n)$ 的前提假设下，使得

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

Step 1: 将 K_2 在 (x_n, y_n) 点作 Taylor 展开

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

Step 2: 将 K_2 代入第1式，得到

$$\begin{aligned} y_{n+1} &= y_n + h\{\lambda_1 y'(x_n) + \lambda_2 [y'(x_n) + phy''(x_n) + O(h^2)]\} \\ &= y_n + (\lambda_1 + \lambda_2) hy'(x_n) + \lambda_2 ph^2 y''(x_n) + O(h^3) \end{aligned}$$

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

$$y_{n+1} = y_n + (\lambda_1 + \lambda_2)hy'(x_n) + \lambda_2 ph^2 y''(x_n) + O(h^3)$$

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

要求 $T_{n+1} = y(x_{n+1}) - y_{n+1} = O(h^3)$，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。

Q: 为获得更高的精度，应该如何进一步推广？

➢ Runge-Kutta方法的一般形式

$$y_{n+1} = y_n + h\sum_{i=1}^{L} \lambda_i k_i$$

$$k_1 = f(x_n, y_n)$$

$$k_2 = f(x_n + c_2 h, y_n + c_2 h k_1)$$

$$k_3 = f(x_n + c_3 h, y_n + c_3 h(a_{31}k_1 + a_{32}k_2))$$

$$\cdots\cdots\cdots\cdots$$

$$k_i = f(x_n + c_i h, y_n + c_i h\sum_{j=1}^{i-1} a_{ij}k_j) \quad i = 2,3,\cdots,L$$

其中$\sum_{i=1}^{L}\lambda_i = 1$，$c_i \le 1$，$\sum_{j=1}^{i-1} a_{ij} = 1$均为待定系数，确定这些系数的步骤与前面相似。

➢ 最常用为四级4阶经典龙格-库塔法 /* Classical Runge-Kutta Method */ ：

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1) \\ K_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2) \\ K_4 = f(x_n + h, y_n + hK_3) \end{cases}$$

我们仍用前面的例子来看看四阶Runge-Kutta方法的效果。

$$\begin{cases} y' = y - \dfrac{2x}{y} \quad (0 < x < 1) \\ y(0) = 1 \end{cases}$$

例　求初值问题 $\begin{cases} y' = y - \dfrac{2x}{y} & (0 < x < 1) \\ y(0) = 1 \end{cases}$

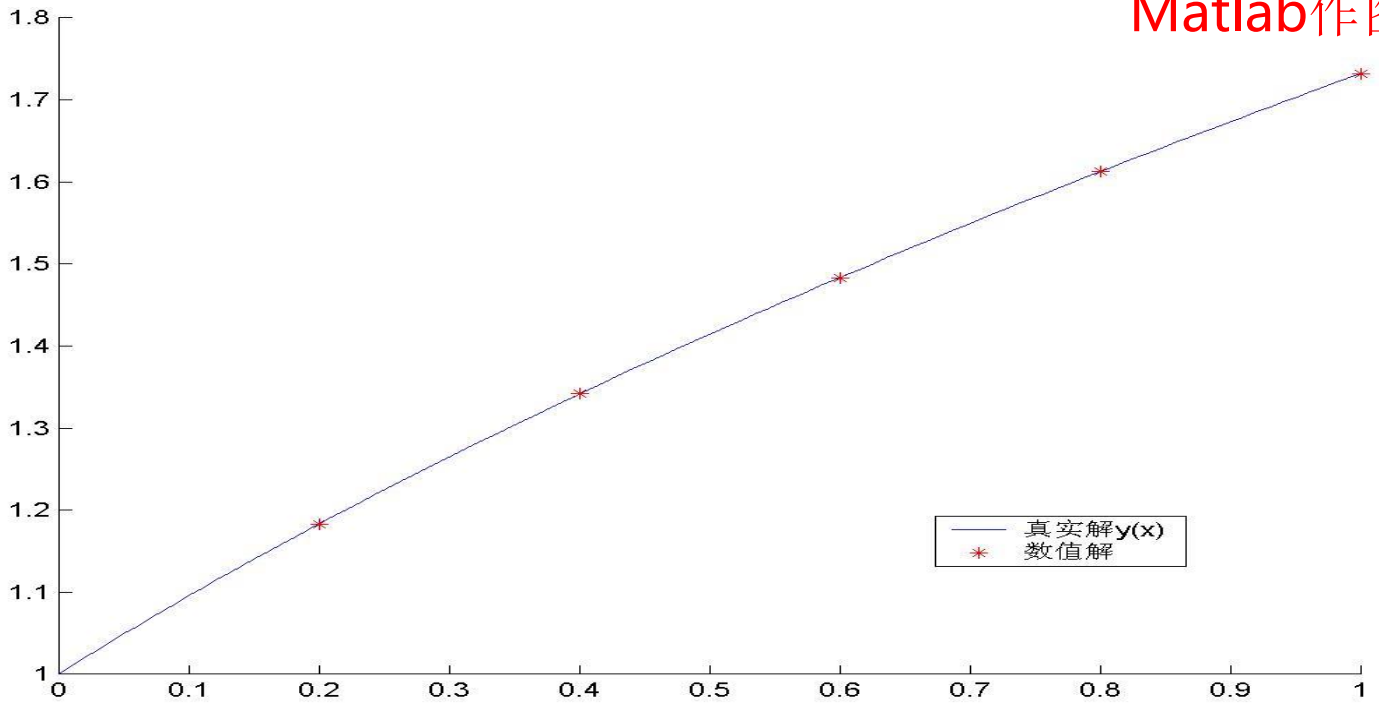
解: 对于本题，经典的四阶Runge-Kutta方法具有以下形式：

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = y_n - \dfrac{2x_n}{y_n} \\ k_2 = y_n + \dfrac{h}{2}k_1 - \dfrac{2x_n + h}{y_n + \dfrac{h}{2}k_1} \\ k_3 = y_n + \dfrac{h}{2}k_2 - \dfrac{2x_n + h}{y_n + \dfrac{h}{2}k_2} \\ k_4 = y_n + hk_3 - \dfrac{2(x_n + h)}{y_n + hk_3} \end{cases}$$

这里，我们取步长h=0.2，下面是计算结果（符号的意义同前）

x_n	y_n	$y(x_n)$
0.2	1.1832	1.1832
0.4	1.3417	1.3416
0.6	1.4833	1.4832
0.8	1.6125	1.6125
1.0	1.7321	1.7321

Matlab作图



从上图可以看出，每一个数值解都“准确”的落在了真实解的曲线上。与改进的Euler法所做出来的图比较，似乎精确性没有很明显的提高，但是不要忘了在用Runge-Kutta方法时，我们取的步长是h=0.2。实际上，Runge-Kutta方法的精确性是要高很多。

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。

§3 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 $x = x_n = x_0 + n\,h$，当 $h \to 0$ (同时 $n \to \infty$) 时有 $y_n \to y(x_n)$，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。

$\lim\limits_{h\to 0}(1+\lambda h)^{1/\lambda h} = e$

解：该问题的精确解为 $y(x) = y_0 e^{\lambda x}$

欧拉公式为 $y_{n+1} = y_n + h\lambda\, y_n = (1+\lambda h) y_n$ ➡ $y_n = (1+\lambda h)^n y_0$

对任意固定的 $x = x_n = n\,h$，有

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

$$\to y_0 e^{\lambda x_n} = y(x_n)$$ ✓

定理 对于一个p阶的显式单步法，若满足如下条件

（1）增量函数φ关于y满足Lipschitz条件，即存在常数$L_\varphi > 0$，使

$$|\varphi(x, y, h) - \varphi(x, \bar{y}, h)| \le L_\varphi |y - \bar{y}|, \quad \forall y, \bar{y} \in R$$

成立；

（2）微分方程的初值是精确的.

则该方法收敛，其整体截断误差为:

$$|e_n| = |y(x_n) - y_n| = O(h^p)。$$

注：1、判断显式单步格式的收敛性，归结为验证增量函数φ是否满足Lipschitz 条件。

2、单步格式的整体截断误差由初值误差及局部截断误差决定，整体截断误差比局部截断误差的阶数低一阶。

3、要构造高精度的计算方法，只需设法提高局部截断误差的阶即可。

➢ 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	2.5000×10^{-1}	2.5000	4.9787×10^{-2}
0.2	4.0000	6.2500×10^{-2}	6.2500	2.4788×10^{-3}
0.3	-8.0000	1.5625×10^{-2}	1.5626×10^{1}	1.2341×10^{-4}
0.4	1.6000×10^{1}	3.9063×10^{-3}	3.9063×10^{1}	6.1442×10^{-6}
0.5	-3.2000×10^{1}	9.7656×10^{-4}	9.7656×10^{1}	3.0590×10^{-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，只考虑试验方程 /* test equation */

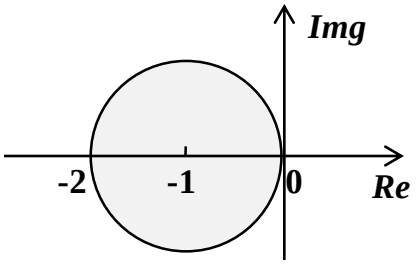
$$y' = \lambda y \quad \mathrm{Re}(\lambda) < 0$$

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 $y_{i+1}=y_i+h\lambda y_i=(1+\bar{h})^{i+1}y_0$

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

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

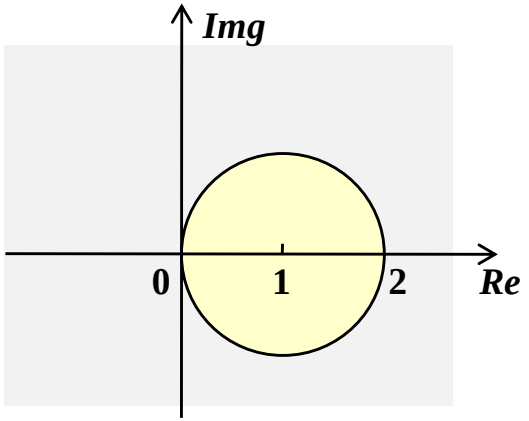


由此可见，要保证初始误差ε_0以后逐步衰减，$\bar{h}=\lambda h$
必须满足：$|1+\bar{h}|<1$

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

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

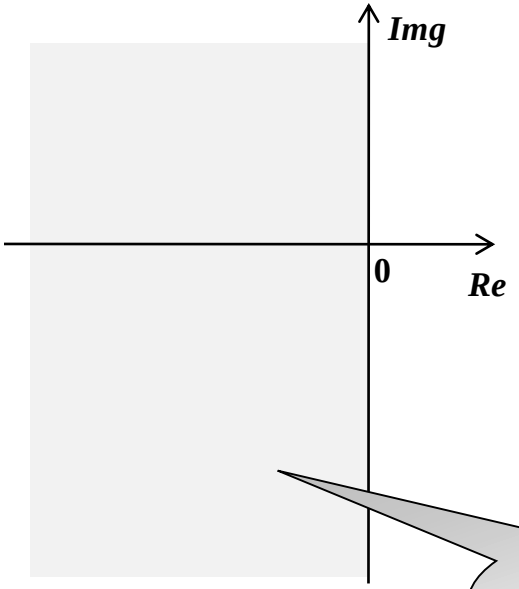
可见绝对稳定区域为：$|1-\bar{h}|>1$



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。

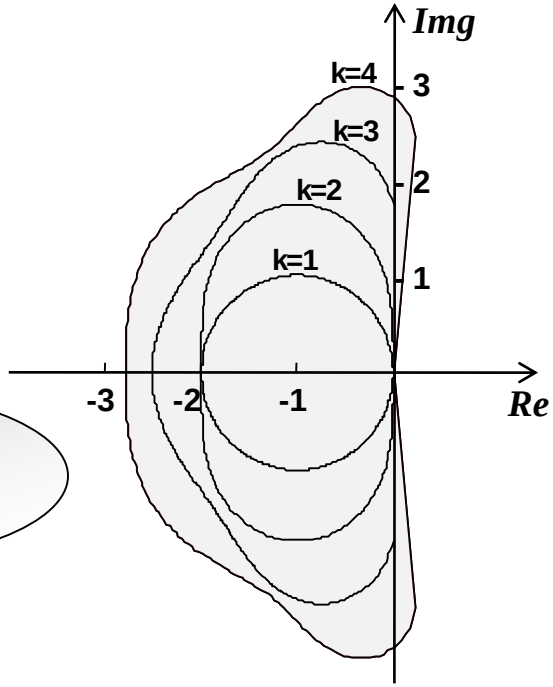
例：隐式龙格-库塔法
$$\begin{cases} y_{i+1} = y_i + h[\lambda_1 K_1 + \ldots + \lambda_m K_m] \\ K_j = f(x_i + \alpha_j h,\ y_i + \beta_{j1} h K_1 + \ldots + \beta_{jm} h K_m) \\ \quad (j = 1, \ldots, m) \end{cases}$$

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



无条件稳定

而显式 1~ 4 阶方法的绝对稳定区域为



§4 线性多步法 /* Multistep Method */

用若干节点处的 y 及 y' 值的线性组合来近似 $y(x_{n+1})$。

其通式可写为：

$$y_{n+1}=\alpha_0 y_n+\alpha_1 y_{n-1}+\ldots+\alpha_k y_{n-k}+h(\beta_{-1} f_{n+1}+\beta_0 f_n+\beta_1 f_{n-1}+\ldots+\beta_k f_{n-k})$$

- 基于数值积分的构造法

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

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

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

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 $f_n, f_{n-1}, \cdots, f_{n-k}$ 构造 k 阶牛顿后插多项式 $N_k(x_n+th)$, $t\in[0,1]$, 有

$$\int_{x_n}^{x_{n+1}} f(x,y(x))dx=\int_0^1 N_k(x_n+th)hdt+\int_0^1 R_k(x_n+th)hdt$$

$$\Rightarrow\ y_{n+1}=y_n+h\int_0^1 N_k(x_n+th)dt \qquad /* \text{显式计算公式} */$$

局部截断误差为: $T_{n+1}=y(x_{n+1})-y_{n+1}=h\int_0^1 R_k(x_n+th)dt$

例: $k=1$ 时有 $N_1(x_n+th)=f_n+t\nabla f_n=f_n+t(f_n-f_{n-1})$

$$\Rightarrow\ y_{n+1}=y_n+h\int_0^1[f_n+t(f_n-f_{n-1})]dt=y_n+\frac{h}{2}(3f_n-f_{n-1})$$

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

注：一般有$T_{n+1}=B_k h^{k+2} y^{(k+2)}(\xi_n)$，其中$B_k$与$y_{n+1}$计算公式中 $f_n, \cdots, f_{n-k}$ 各项的系数均可查表得到。

k	f_n	f_{n-1}	f_{n-2}	f_{n-3}	$\cdots$	B_k
0	1					$\frac{1}{2}$
1	$\frac{3}{2}$	$-\frac{1}{2}$				$\frac{5}{12}$
2	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$			$\frac{3}{8}$
3	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$		$\frac{251}{720}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

常用的是 $k=3$ 的4阶亚当姆斯显式公式

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

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_{n+1}，f_n，…，f_{n-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$T_{n+1}=\tilde{B}_k h^{k+2} y^{(k+2)}(\eta_n)$，其中$\tilde{B}_k$与$f_{n+1}$，$f_n$，…，$f_{n-k+1}$的系数亦可查表得到。

k	f_{n+1}	f_n	f_{n-1}	f_{n-2}	…	$\tilde{B}_k$
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	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$		$-\frac{19}{720}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

小于B_k

常用的是 $k=3$ 的4阶亚当姆斯隐式公式

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

较同阶显式稳定

➢ 基于泰勒展开的构造法

$$y_{n+1}=\alpha_0 y_n+\alpha_1 y_{n-1}+\ldots+\alpha_k y_{n-k}+h(\beta_{-1} f_{n+1}+\beta_0 f_n+\beta_1 f_{n-1}+\ldots+\beta_k f_{n-k})$$

将通式中的右端各项 $y_{n-1}, \cdots, y_{n-k}; f_{n+1}, f_{n-1}, \cdots, f_{n-k}$ 分别在 x_n 点作泰勒展开，与精确解 $y(x_{n+1})$ 在 x_n 点的泰勒展开作比较。通过令同类项系数相等，得到足以确定待定系数 $\alpha_0, \cdots, \alpha_k; \beta_{-1}, \beta_0, \cdots, \beta_k$ 的等式，则可构造出线性多步法的公式。

例： 设 $y_{n+1}=\alpha_0 y_n+\alpha_1 y_{n-1}+\alpha_2 y_{n-2}+h(\beta_0 y'_n+\beta_1 y'_{n-1}+\beta_2 y'_{n-2}+\beta_3 y'_{n-3})$

确定式中待定系数α_0, α_1, α_2, β_0, β_1, β_2, β_3, 使得公式具有4阶精度。

解：

$$y_{n-1}=y_n-hy'_n+\tfrac{1}{2}h^2y''_n-\tfrac{1}{6}h^3y'''_n+\tfrac{1}{24}h^4y^{(4)}_n+O(h^5)$$

$$y_{n-2}=y_n-2hy'_n+2h^2y''_n-\tfrac{4}{3}h^3y'''_n+\tfrac{2}{3}h^4y^{(4)}_n+O(h^5)$$

$$y'_{n-1}=y'_n-hy''_n+\tfrac{1}{2}h^2y'''_n-\tfrac{1}{6}h^3y^{(4)}_n+O(h^4)$$

$$y'_{n-2}=y'_n-2hy''_n+2h^2y'''_n-\tfrac{4}{3}h^3y^{(4)}_n+O(h^4)$$

$$y'_{n-3}=y'_n-3hy''_n+\tfrac{9}{2}h^2y'''_n-\tfrac{9}{2}h^3y^{(4)}_n+O(h^4)$$

$$y(x_{n+1})=y_n+hy'_n+\tfrac{1}{2}h^2y''_n+\tfrac{1}{6}h^3y'''_n+\tfrac{1}{24}h^4y^{(4)}_n+O(h^5)$$ /* $y(x_n)=y_n$ */

$$\alpha_0+\alpha_1+\alpha_2=1$$

$$h(-\alpha_1-2\alpha_2+\beta_0+\beta_1+\beta_2+\beta_3)=h$$

$$h^2(\tfrac{1}{2}\alpha_1+2\alpha_2-\beta_1-2\beta_2-3\beta_3)=\tfrac{1}{2}h^2$$

$$h^3(-\tfrac{1}{6}\alpha_1-\tfrac{4}{3}\alpha_2+\tfrac{1}{2}\beta_1+2\beta_2+\tfrac{9}{2}\beta_3)=\tfrac{1}{6}h^3$$

$$h^4(\tfrac{1}{24}\alpha_1+\tfrac{2}{3}\alpha_2-\tfrac{1}{6}\beta_1-\tfrac{4}{3}\beta_2-\tfrac{9}{2}\beta_3)=\tfrac{1}{24}h^4$$

7 个未知数

5 个方程

- 令 $\alpha_1 = \alpha_2 = 0$ ➡ *Adams* 显式公式
- 以 y'_{n+1} 取代 y'_{n-3}，并取 $\alpha_1 = \alpha_2 = 0$ ➡ *Adams* 隐式公式
- 以 y_{n-3} 取代 y'_{n-3}，则可导出另一组4 阶显式算法，其中包含了著名的米尔尼 /* Milne */ 公式

$$y_{n+1} = y_{n-3} + \frac{4h}{3}(2y'_n - y'_{n-1} + 2y'_{n-2})$$

其局部截断误差为 $T_{n+1} = \frac{14}{45}h^5 y^{(5)}(\xi_n)$，$\xi_n \in (x_n, x_{n+1})$

注：上式也可通过数值积分导出，即将 $y' = f(x,y)$ 在区间 $[x_{n-3}, x_{n+1}]$ 上积分，得到 $y(x_{n+1}) = y(x_{n-3}) + \int_{x_{n-3}}^{x_{n+1}} f(x, y(x))dx$，再过 f_n, f_{n-1}, f_{n-2} 做 f 的插值多项式即可。

- 辛甫生 /* Simpson */ 公式 $y_{n+1} = y_{n-1} + \frac{h}{3}(y'_{n+1} + 4y'_n + y'_{n-1})$

☞ *Milne-Simpson* 系统的缺点是稳定性差，为改善稳定性，考虑另一种隐式校正公式：

$$y_{n+1}=\alpha_0 y_n+\alpha_1 y_{n-1}+\alpha_2 y_{n-2}+h(\beta_{-1}y'_{n+1}+\beta_0 y'_n+\beta_1 y'_{n-1})$$

要求公式具有4 阶精度。通过泰勒展开，可得到 5 个等式，从中解出 6 个未知数，则有 1 个自由度。

哈明 /* Hamming */ 用α_1 的不同数值进行试验，发现当$\alpha_1 = 0$ 时，公式的稳定性较好，即：

$$y_{n+1}=\frac{1}{8}(9y_n-y_{n-2})+\frac{3h}{8}(y'_{n+1}+2y'_n-y'_{n-1})$$

其局部截断误差为$T_{n+1}=-\frac{1}{40}h^5 y^{(5)}(\xi_n),\ \xi_n\in(x_n,x_{n+1})$

注：哈明公式不能用数值积分方法推导出来。

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 $y(x_{n+1})-\bar{y}_{n+1}=\frac{251}{720}h^5y^{(5)}(\xi_n)$

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

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

$$\left.\begin{aligned} y(x_{n+1})&\approx \bar{y}_{n+1}+\frac{251}{270}(y_{n+1}-\bar{y}_{n+1})\\ y(x_{n+1})&\approx y_{n+1}-\frac{19}{270}(y_{n+1}-\bar{y}_{n+1}) \end{aligned}\right\}$$

Adams 4th-Order predictor-corrector Algorithm

To approximate the the solution of the initial-value problem

$$y' = f(x, y), \quad a \le x \le b, \quad y(a) = y_0$$

At (N+1) equally spaced numbers in the interval [a, b].

Input: endpoints a, b; integer N; initial value $y0$.

Output: approximation y at the (N+1) values of x.

Step 1 Set $h = (b - a) / N$; $x_0 = a$; $y_0 = y0$; Output (x_0, y_0);

Step 2 For n = 1, 2, 3

Compute y_n using classical Runge-Kutta method; Output (x_n , y_n);

Step 3 For n = 4, …, N do steps 4-10

Step 5 $p_{n+1} = y_n + h(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3}) / 24$; /* predict */

Step 6 $m_{n+1} = p_{n+1} + 251(c_n - p_n) / 270$; /* modify */

Step 7 $c_{n+1} = y_n + h(9 f(x_{n+1}, m_{n+1}) + 19 f_n - 5 f_{n-1} + f_{n-2}) / 24$; /* correct */

Step 8 $y_{n+1} = c_{n+1} - 19(c_{n+1} - p_{n+1}) / 270$; /* modify the final value */

Step 9 Output (x_{n+1} , y_{n+1});

Step 10 For j = 0, 1, 2, 3

Set $x_n = x_{n+1}$; $y_n = y_{n+1}$; /* Prepare for next iteration */

Step 11 STOP.

§5 微分方程组与高阶方程 /* Systems of Differential Equations and Higher-Order Equations */

- 一阶微分方程组

IVP的一般形式为：

$$\begin{cases} y_1'(x) = f_1(x, y_1(x), \dots, y_m(x)) \\ \cdots\cdots\cdots \\ y_m'(x) = f_m(x, y_1(x), \dots, y_m(x)) \end{cases}$$

初值 $y_1(x_0) = y_1^0,\ y_2(x_0) = y_2^0, \dots, y_m(x_0) = y_m^0$

将问题记作向量形式，令：$\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix},\ \vec{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix},\ \vec{y}_0 = \begin{pmatrix} y_1^0 \\ \vdots \\ y_m^0 \end{pmatrix}$

$$\Rightarrow \begin{cases} \vec{y}'(x) = \vec{f}(x, \vec{y}) \\ \vec{y}(x_0) = \vec{y}_0 \end{cases}$$

➢ 高阶微分方程

$$\begin{cases} y^{(n)} = f(x, y, y', \cdots, y^{(n-1)}) \\ y(x_0) = a_0, \quad y'(x_0) = a_1, \cdots, y^{(n-1)}(x_0) = a_{n-1} \end{cases}$$

化作一阶微分方程组求解。

引入新变量 $y_1 = y, \quad y_2 = y', \cdots, \quad y_n = y^{(n-1)}$

$$\Rightarrow \begin{cases} y_1' = y_2 \\ \vdots \\ y_{n-1}' = y_n \\ y_n' = f(x, y_1, \cdots, y_n) \end{cases}$$

初值条件为：

$$y_1(x_0) = a_0$$
$$y_2(x_0) = a_1$$
$$\cdots$$
$$y_n(x_0) = a_{n-1}$$

§6 边值问题的数值解 /* Boundary-Value Problems */

2 阶常微分方程边值问题

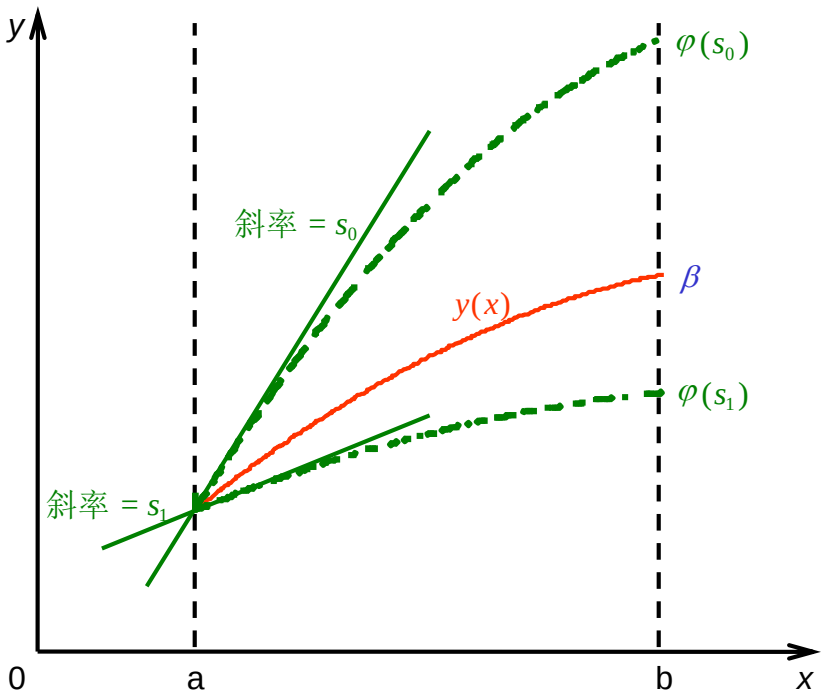
$$\begin{cases} y'' = f(x, y, y') & x \in (a, b) \\ y(a) = \alpha, \ \ y(b) = \beta \end{cases}$$

- 打靶法 /* shooting method */

先猜测一个初始斜率 $y'(a) = s$，通过解初值问题

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = a \\ y'(a) = s \end{cases} \Rightarrow y(b) = \varphi(s)$$

找出s^*使得$\varphi(s^*) = \beta$，即把问题转化为求方程 $\varphi(s) - \beta = 0$ 的根。



➢ 有限差分法 /* finite difference method */

将求解区间$[a, b]$等分为N份，取节点$x_i = a + ih$ $(i = 0, \ldots, N)$，在每一个节点处将y'和y''离散化。

泰勒展开

$$y''(x) = \frac{\dfrac{y(x+h)-y(x)}{h} - \dfrac{y(x)-y(x-h)}{h}}{h} - \frac{h^2}{12}y^{(4)}(\xi)$$

$$= \frac{y(x+h)-2y(x)+y(x-h)}{h^2} + O(h^2)$$

$$y'(x) = \frac{y(x+h)-y(x-h)}{2h} + O(h^2)$$

$$\begin{cases} \dfrac{y_{i+1}-2y_i+y_{i-1}}{h^2} = f\left(x_i, y_i, \dfrac{y_{i+1}-y_{i-1}}{2h}\right) & i = 1, \ldots, N-1 \\ y_0 = \alpha, \ y_N = \beta \end{cases}$$