



第七章 常微分方程的数值解法

肖 明

微电子科学与技术学院

邮箱: xiaom37@mail.sysu.edu.cn



- 1 欧拉法
- 2 **改进的欧拉法**
- 3 **龙格—库塔方法**
- 4 阿达姆斯方法
- 5 算法的稳定性和收敛性



回顾-Euler法

研究一阶常微分方程的初值问题的数值解

已知初值问题的一般形式为:

$$\begin{cases} \frac{dy}{dx} = f(x, y) \\ y(x_0) = y_0 \end{cases} \quad a \leq x \leq b \quad (1)$$

用差商近似导数

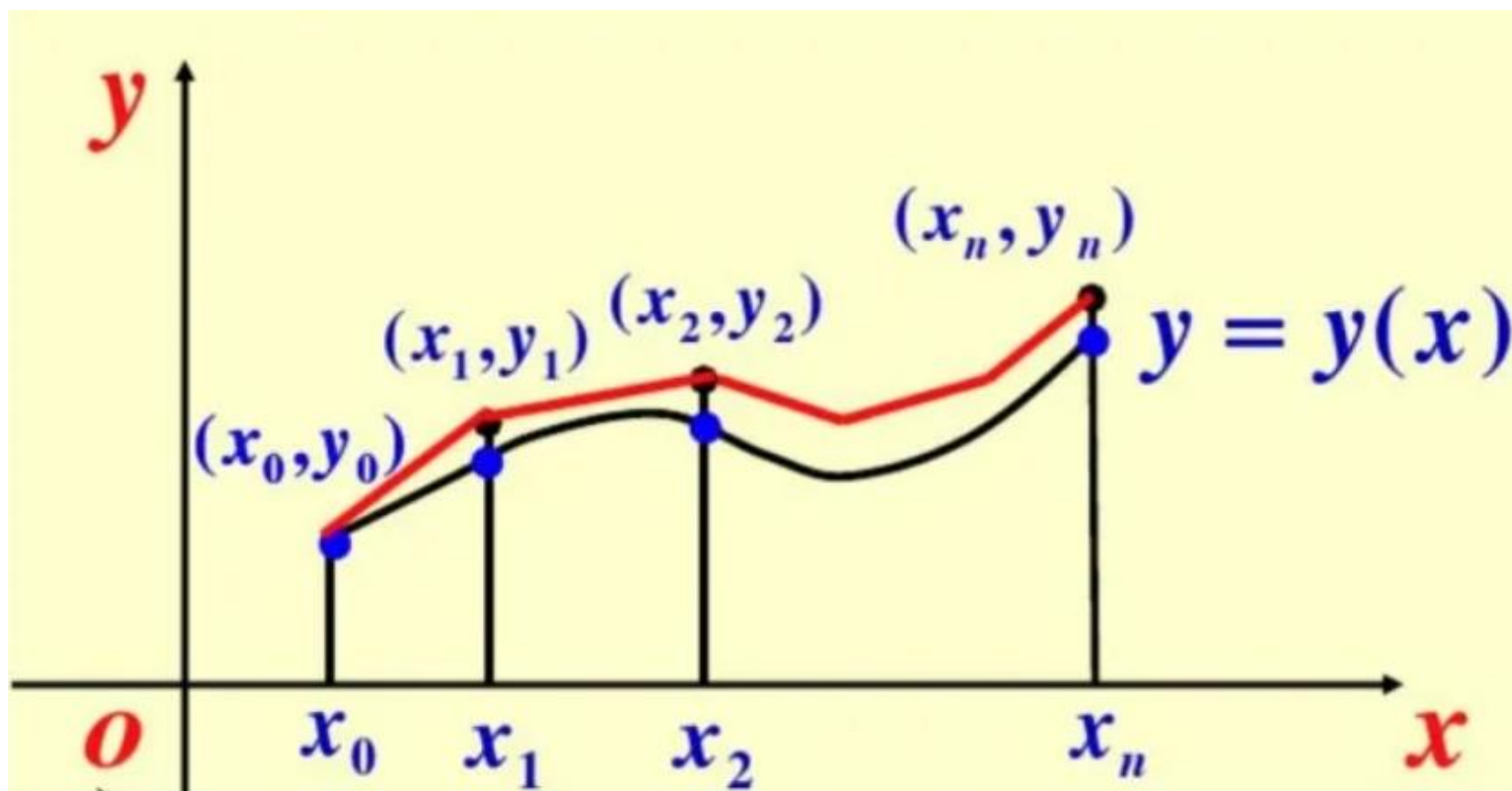
$$\frac{y_{n+1} - y_n}{h} \approx \frac{dy}{dx}$$

问题转化为

$$\begin{cases} y_{n+1} = y_n + hf(x_n, y_n) \\ y_0 = y(x_0) \end{cases} \quad (n = 0, 1, 2, 3, \dots)$$

-Euler方法的迭代公式

回顾-Euler法的几何意义



欧拉法的几何意义：用一条折线近似代替积分曲线



Euler法的误差估计

1、局部截断误差。

在一步中产生的误差而非累积误差：

$$R_{n+1} = y(x_{n+1}) - \tilde{y}_{n+1}$$

其中 $\tilde{y}_{n+1}$ 是当 $y_n = y(x_n)$ (精确等!) 时由 *Euler* 法求出的值, 即 y_n 无误差!

将 $y(x_{n+1})$ 在 x_n 点 *Taylor* 展开:

$$y(x_{n+1}) = y(x_n + h) = y(x_n) + hf(x_n, y(x_n)) + \frac{h^2}{2} y''(\xi) \quad x_n < \xi < x_{n+1}$$



Euler法的误差估计

$$\tilde{y}_{n+1} = y(x_n) + hf(x_n, y(x_n))$$

则 $R_{n+1} = y(x_{n+1}) - \tilde{y}_{n+1} = \frac{h^2}{2} y''(\xi) \quad x_n < \xi < x_{n+1}$

令 $M = \max_{a \leq x \leq b} |y''(x)|$, $y(x)$ 充分光滑, 则:

$$|R_{n+1}| \leq M \frac{h^2}{2} = O(h^2)$$



Euler法的误差估计

2、整体截断误差

定义： 在不考虑舍入误差的情况下，考虑每一步局部截断误差的影响而求得的 y_{n+1} 与 $y(x_{n+1})$ 的误差 $e_{n+1} = y(x_{n+1}) - y_{n+1}$

由微分方程解的存在唯一性,自然假定 $f(x, y)$ 充分光滑, 即

$$|y''(x)| \leq M \Rightarrow |R_{n+1}| = \frac{h^2}{2} |y''(\xi)| \leq \frac{h^2}{2} M$$

或满足 *Lipschitz* 条件: $|f(x_n, y(x_n)) - f(x_n, y_n)| \leq L |y(x_n) - y_n|$

第 n 步的总体截断误差记为 $e_n = y(x_n) - y_n$, 则对 $n+1$ 步:

$$|e_{n+1}| = |y(x_{n+1}) - y_{n+1}| \leq |y(x_{n+1}) - \tilde{y}_{n+1}| + |\tilde{y}_{n+1} - y_{n+1}| = |R_{n+1}| + |\tilde{y}_{n+1} - y_{n+1}|$$

以下估计 $|\tilde{y}_{n+1} - y_{n+1}|$



Euler法的误差估计

$$\begin{aligned} |\tilde{y}_{n+1} - y_{n+1}| &= |y(x_n) + hf(x_n, y(x_n)) - y_n - hf(x_n, y_n)| \\ &\leq |y(x_n) - y_n| + h|f(x_n, y(x_n)) - f(x_n, y_n)| \\ (\text{由} Lipschitz \text{条件}) &\leq (1 + hL)|y(x_n) - y_n| = ((1 + hL)|e_n|) \end{aligned}$$

$$\therefore |e_{n+1}| \leq |R_{n+1}| + (1 + hL)|e_n| \quad \text{对一切} n \text{ 成立。} \quad \boxed{e_{n+1} = y(x_{n+1}) - y_{n+1}}$$

反复递推得

$$\begin{aligned} |e_{n+1}| &\leq \frac{h^2}{2} M + (1 + hL) \left[\frac{h^2}{2} M + (1 + hL) |e_{n-1}| \right] \\ &\leq \frac{h^2}{2} M \sum_{k=0}^n (1 + hL)^k \end{aligned}$$



Euler法的误差估计

$$= \frac{h^2}{2} M \frac{(1+hL)^{n+1} - 1}{1+hL - 1}$$

$$= \frac{hM}{2L} [(1+hL)^{n+1} - 1]$$

Euler方法是一阶方法,故精度不高.

$$(n+1)h \leq b-a \Rightarrow (1+hL)^{n+1}$$

$$\leq (1+hL)^{\frac{b-a}{h}} = (1+hL)^{\frac{1}{hL}L(b-a)} < e^{L(b-a)}$$

$$\therefore |e_{n+1}| \leq \frac{hM}{2L} [e^{L(b-a)} - 1] = O(h)$$

Euler 方法以 $O(h)$ 速率收敛: $h \rightarrow 0, e_N \rightarrow 0$.



不同形式的欧拉法



向后Euler法

用向后差商: $y'(x_{n+1}) \approx \frac{y(x_{n+1}) - y(x_n)}{h}$

则隐式算法:
$$\begin{cases} y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \\ y(x_0) = y_0 \end{cases}$$

与Euler法结合可得迭代公式, 事实上是逐步显化。

$$\begin{cases} y_{n+1}^{(0)} = y_n + hf(x_n, y_n) \\ y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)}) \end{cases} \quad k = 0, 1, 2, \dots$$

需要进行迭代计算

局部截断误差 $R_{n+1} = O(h^2)$, 整体截断误差: $O(h)$.

$$|y_{n+1}^{(k+1)} - y_{n+1}^{(k)}| = h \left| f(x_{n+1}, y_{n+1}^{(k)}) - f(x_{n+1}, y_{n+1}^{(k-1)}) \right| \leq hL |y_{n+1}^{(k)} - y_{n+1}^{(k-1)}|$$

当 $0 < hL < 1$ 时收敛

利用数值积分将微分方程离散化得梯形公式:

$$\begin{aligned}y(x_{n+1}) - y(x_n) &= \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \\&\approx \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \\ \Rightarrow y_{n+1} &= y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})]\end{aligned}$$

梯形方法为隐式算法

与Euler法结合, 形成迭代算法 对 $n = 0, 1, 2, \dots$

$$\begin{cases} y_{n+1}^{(0)} = y_n + hf(x_n, y_n) \\ y_{n+1}^{(k+1)} = y_n + \frac{h}{2} \left(f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(k)}) \right) \quad k = 0, 1, 2, \dots \end{cases}$$

需要进行迭代计算

梯形公式比欧拉法精度高, 但计算量较大 (需要使用迭代计算)



改进的Euler法

预测 $\tilde{y}_{k+1} = y_k + hf(x_k, y_k)$

校正 $y_{k+1} = y_k + \frac{h}{2}(f(x_k, y_k) + f(x_{k+1}, \tilde{y}_{k+1}))$

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

隐式法

实际计算中只迭代一次，这样建立的预测—校正系统称作**改进的欧拉公式**。

$$\text{作等价变换: } \begin{cases} y_{n+1} = y_n + \frac{h}{2}(K_1 + K_2) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + h, y_n + hK_1) \\ y(x_0) = y_0 \end{cases}$$



改进的Euler法---典型例题

求解初值问题(步长 $h = 0.1$)

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

解

$$f(x, y) = y - 2x / y$$

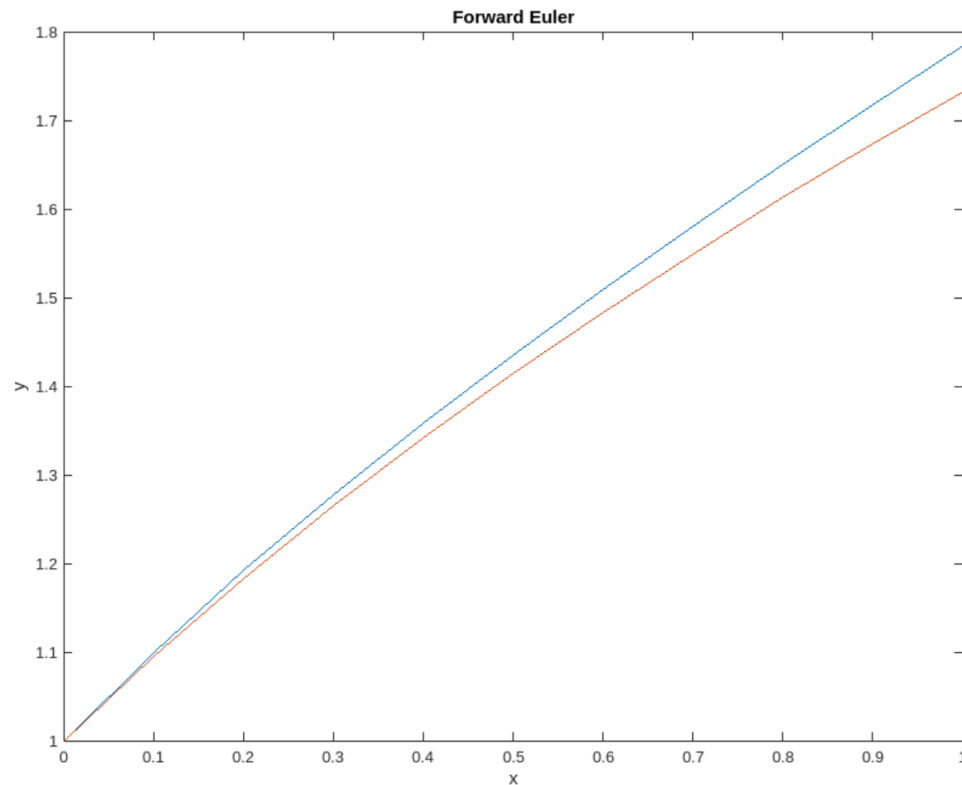
作等价变换:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(K_1 + K_2) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + h, y_n + hK_1) \\ y(x_0) = y_0 \end{cases}$$

```
function result = forward(a, b, h, y)
    n = (b-a)/h;
    x0 = a;
    x1 = a;
    y0 = y;
    result(1,1) = x0;
    result(2,1) = y0;

    for m = 0:n-1
        x1 = x1 + h;
        f0 = y0 - 2*x0/y0;
        y1 = y0 + h*f0;
        x0 = x1;
        y0 = y1;
        result(1, m+2) = x0;
        result(2, m+2) = y0;
    end
end
```

$$\begin{cases} y_{n+1} = y_n + hK_1 \\ K_1 = f(x_n, y_n) \\ y(0) = 1 \end{cases}$$



精确解

向后Euler 法—程序设计

```
function result = back_euler(a, b, h, y)
    n = (b-a)/h;
    x0 = a;
    x1 = a;
    y0 = y;
    result(1,1) = x0;
    result(2,1) = y0;

    for m = 0:n-1
        x1 = x1 + h;
        f0 = y0 - 2*x0/y0;
        d = y0 + h*f0;
        y1 = calculate(y0, x1, d, h);
        %result = calculate(x1, d, h);
        x0 = x1;
        y0 = y1;
        result(1, m+2) = x0;
        result(2, m+2) = y0;
    end
end
```

```
function result = calculate(y0, x1, y1, h)

    acc = -6;
    now = 0.0;
    z1 = y1;

    while now >= -6

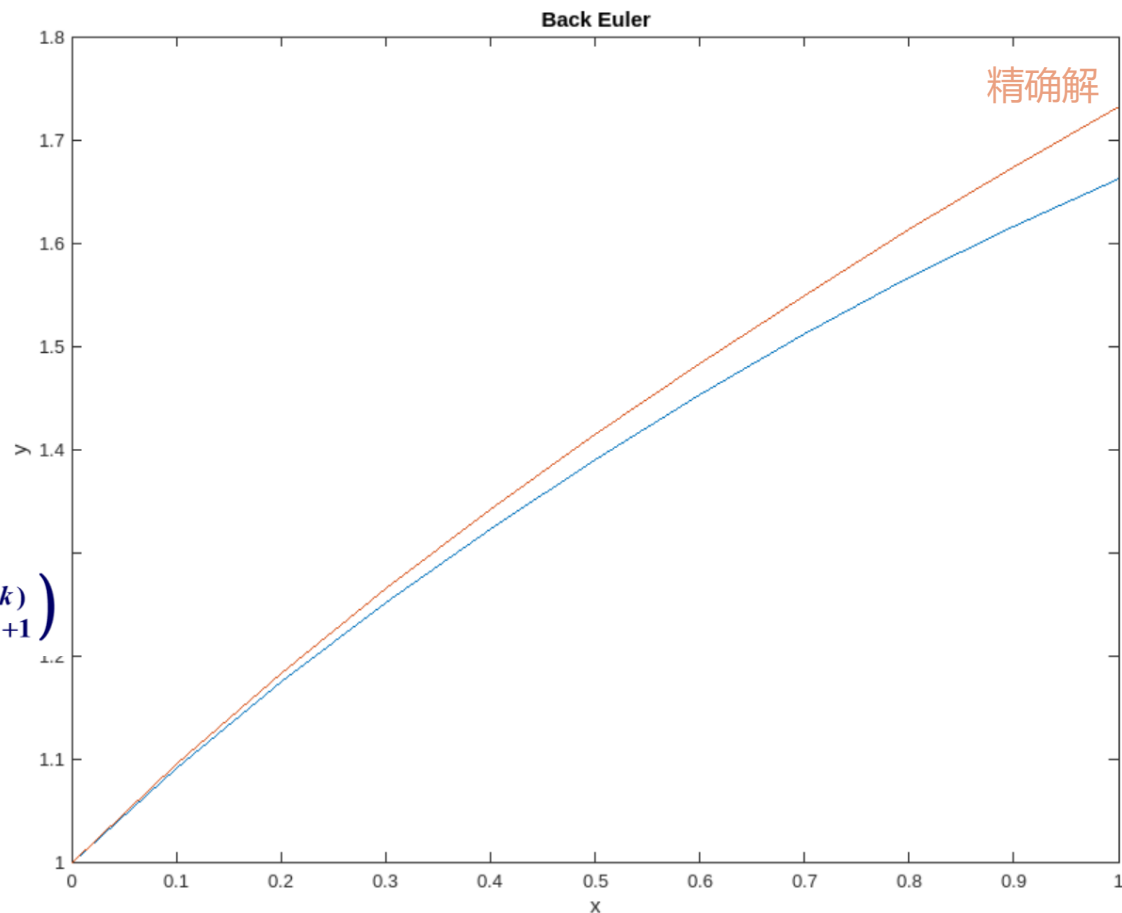
        z0 = z1;
        f0 = z0 - 2*x1/z0;
        z1 = y0 + h*f0;
        now = log10(abs(z1-z0));

    end
    result = z1;
end
```

按照步长逐步计算

$$\begin{cases} y_{n+1}^{(0)} = y_n + hf(x_n, y_n) \\ y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)}) \end{cases}$$

迭代求解直到收敛满足



Euler 梯形法—程序设计



```
function result = trapi_eular(a, b, h, y)
    n = (b-a)/h;
    x0 = a;
    x1 = a;
    y0 = y;
    result(1,1) = x0;
    result(2,1) = y0;

    for m = 0:n-1
        x1 = x1 + h;
        f0 = y0 - 2*x0/y0;
        d = y0 + h*f0;
        y1 = calculate(y0, x1, d, h, f0);
        x0 = x1;
        y0 = y1;
        result(1, m+2) = x0;
        result(2, m+2) = y0;
    end
end
```

```
function result = calculate(y0, x1, y1, h, f
    acc = -6;
    now = 0.0;
    z1 = y1;

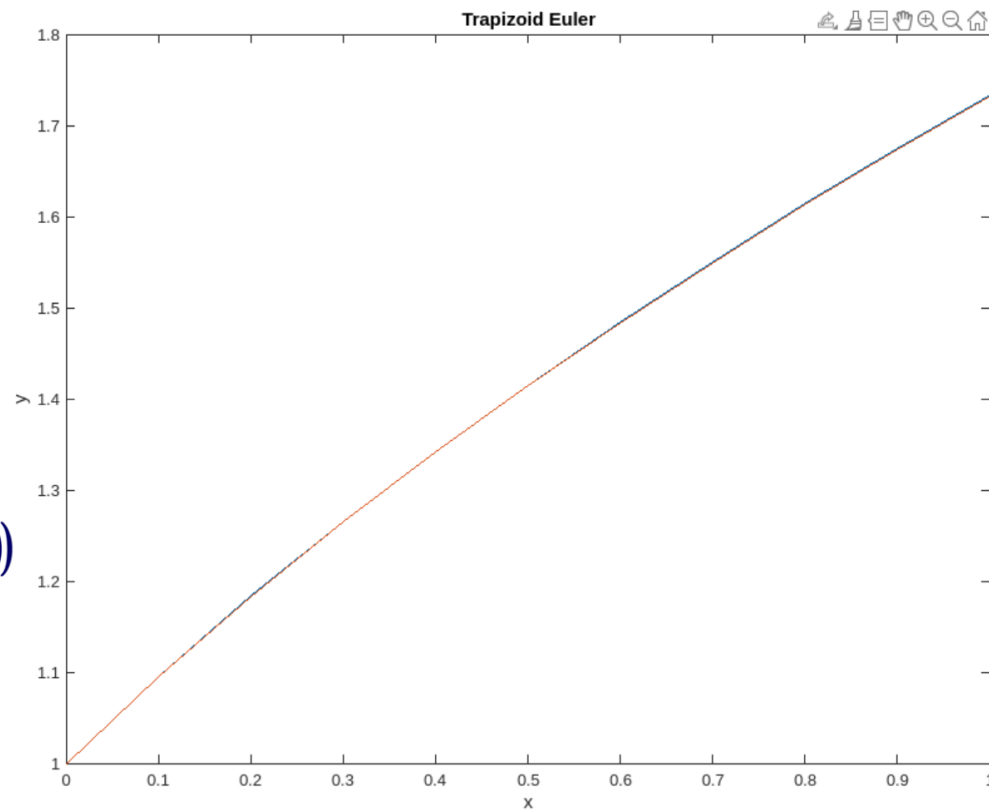
    while now >= acc
        z0 = z1;
        f1 = z0 - 2*x1/z0;
        z1 = y0 + h/2*(f0+f1);
        now = log10(abs(z1-z0));
    end

    result = z1;
end
```

按照步长逐步计算

$$\begin{cases} y_{n+1}^{(0)} = y_n + hf(x_n, y_n) \\ y_{n+1}^{(k+1)} = y_n + \frac{h}{2} (f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(k)})) \end{cases}$$

迭代求解直到收敛满足





改进的Euler法---程序设计

```
function result = mod_euler(a, b, h, y)
```

```
    n = (b-a)/h;
```

```
    x0 = a;
```

```
    x1 = a;
```

```
    y0 = y;
```

```
    result(1,1) = x0;
```

```
    result(2,1) = y0;
```

```
    for m = 0:n-1
```

```
        x1 = x1 + h;
```

```
        f0 = y0 - 2*x0/y0;
```

```
        d = y0 + h*f0;
```

```
        f1 = d - 2*x1/d;
```

```
        y1 = y0 + h/2*(f0+f1);
```

```
        x0 = x1;
```

```
        y0 = y1;
```

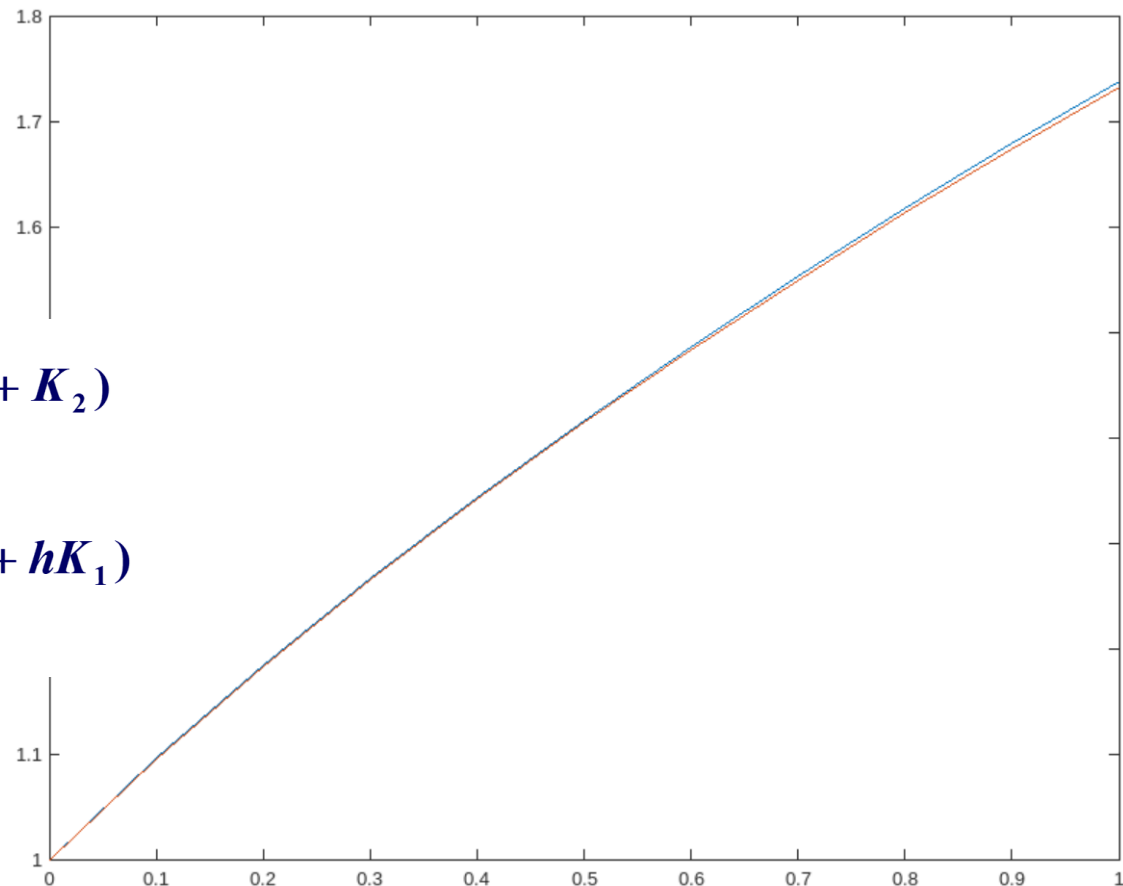
```
        result(1, m+2) = x0;
```

```
        result(2, m+2) = y0;
```

```
    end
```

```
end
```

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(K_1 + K_2) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + h, y_n + hK_1) \\ y(x_0) = y_0 \end{cases}$$



解的表达式: $y = \sqrt{1+2x}$

结果比较

精确解

$y[0] \rightarrow 1$
 $y[0.1] \rightarrow 1.09545$
 $y[0.2] \rightarrow 1.18322$
 $y[0.3] \rightarrow 1.26491$
 $y[0.4] \rightarrow 1.34164$
 $y[0.5] \rightarrow 1.41421$
 $y[0.6] \rightarrow 1.48324$

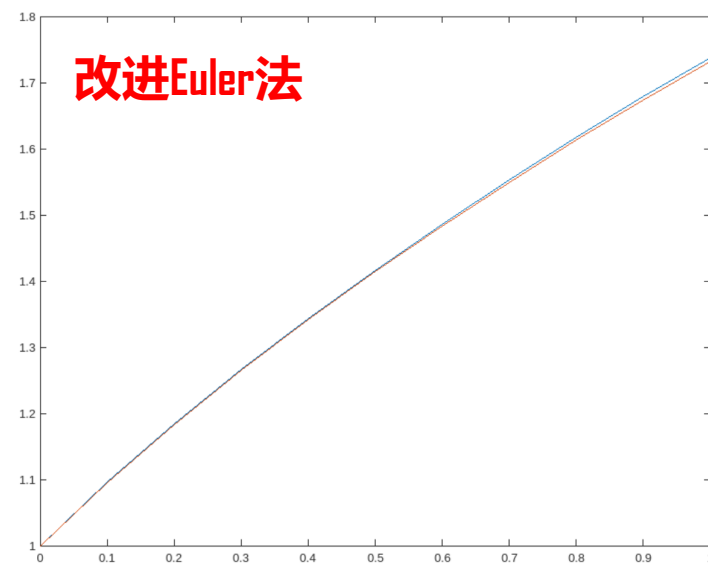
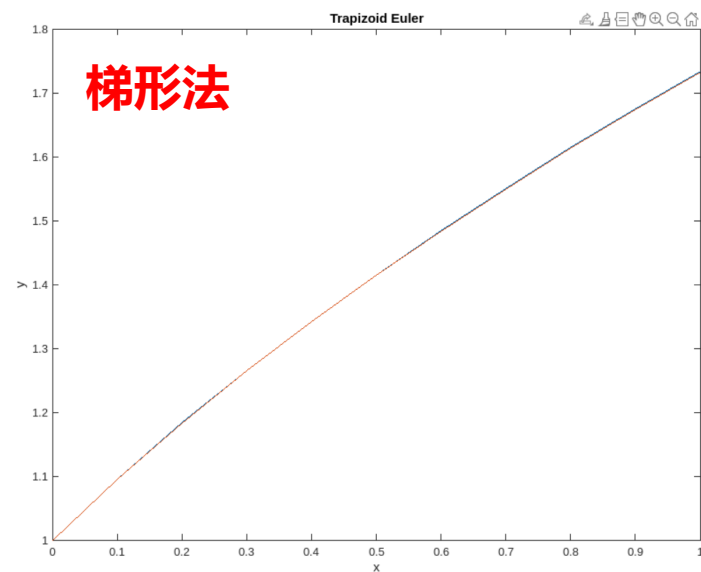
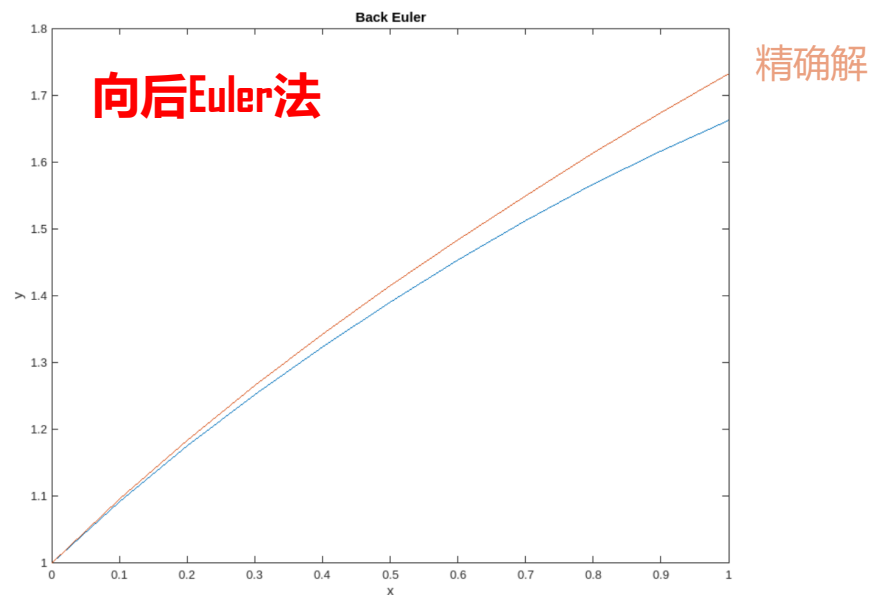
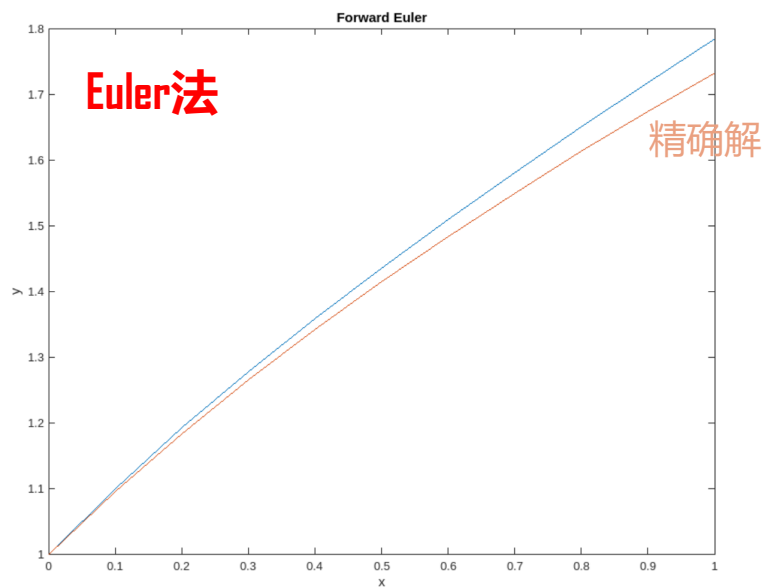
Euler近似解

0	1.
0.1	1.1
0.2	1.19182
0.3	1.27744
0.4	1.35821
0.5	1.43513
0.6	1.50897

改进Euler近似解

0	1.
0.1	1.09774
0.2	1.18757
0.3	1.27129
0.4	1.35013
0.5	1.42499
0.6	1.49657

四种Euler法的对比



四种Euler法的对比---数值积分角度

$$\frac{y_{n+1} - y_n}{x_{n+1} - x_n} = f(x_n, y_n)$$

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

$$y' = f(x, y), x \in [x_0, b], \quad (1)$$

$$y(x_0) = y_0 \quad (2)$$

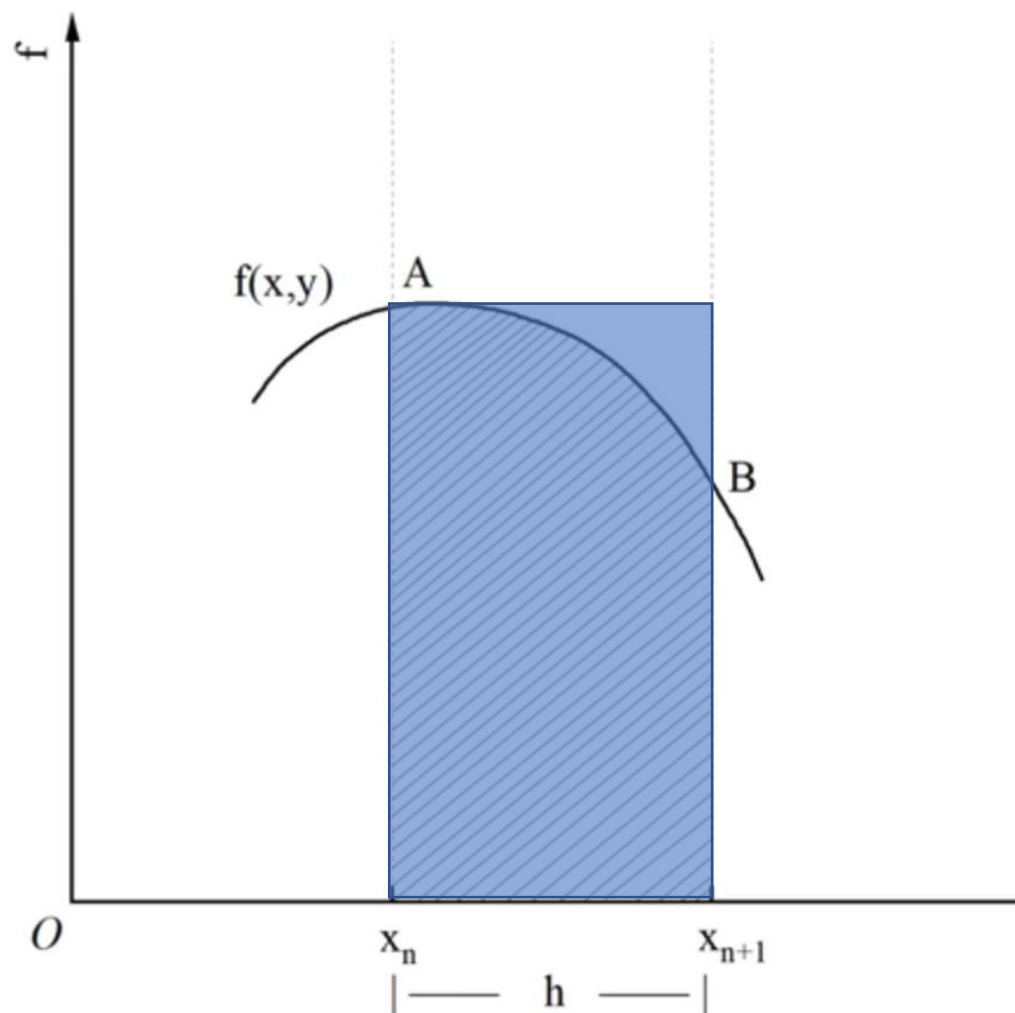
对微分方程(1)从 x_n 到 x_{n+1} 积分, 得

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(t, y(t)) dt \quad (4)$$

右端积分用**左矩形公式** $hf(x_n, y(x_n))$ 近似,
再用 y_{n+1} 代替 $y(x_{n+1})$, y_n 代替 $y(x_n)$,
即可得到

$$y_{n+1} = y_n + hf(x_n, y_n) \quad (3)$$

Euler法的数值积分



四种Euler法的对比---数值积分角度

$$\frac{y_{n+1} - y_n}{x_{n+1} - x_n} = f(x_n, y_n)$$

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

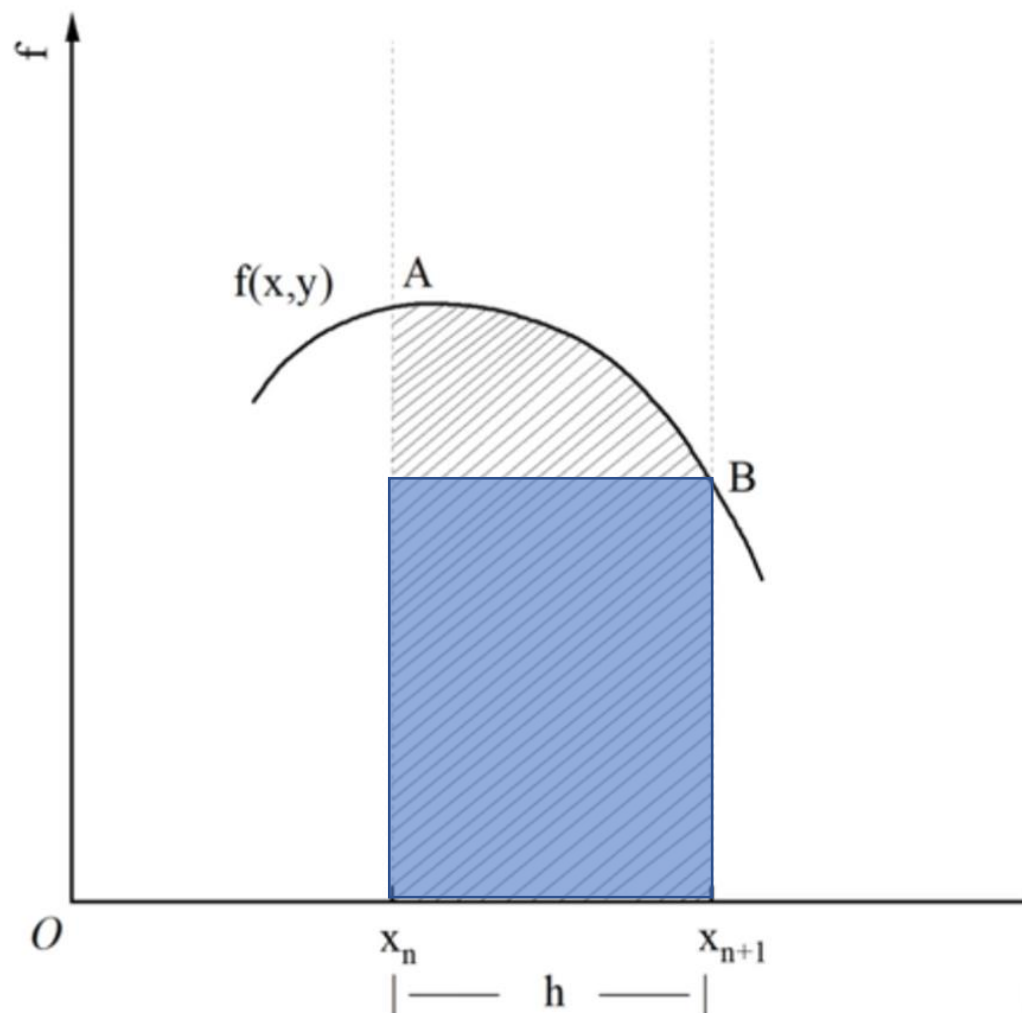
对微分方程(1)从 x_n 到 x_{n+1} 积分, 得

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(t, y(t)) dt \quad (4)$$

右端积分用**右矩形公式** $hf(x_{n+1}, y(x_{n+1}))$ 近似,
再用 y_{n+1} 代替 $y(x_{n+1})$, y_n 代替 $y(x_n)$,
即可得到

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \quad (5)$$

向后Euler法的数值积分





四种Euler法的对比---数值积分角度

$$\frac{y_{n+1} - y_n}{x_{n+1} - x_n} = f(x_n, y_n)$$

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

对微分方程(1)从 x_n 到 x_{n+1} 积分, 得

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(t, y(t)) dt \quad (4)$$

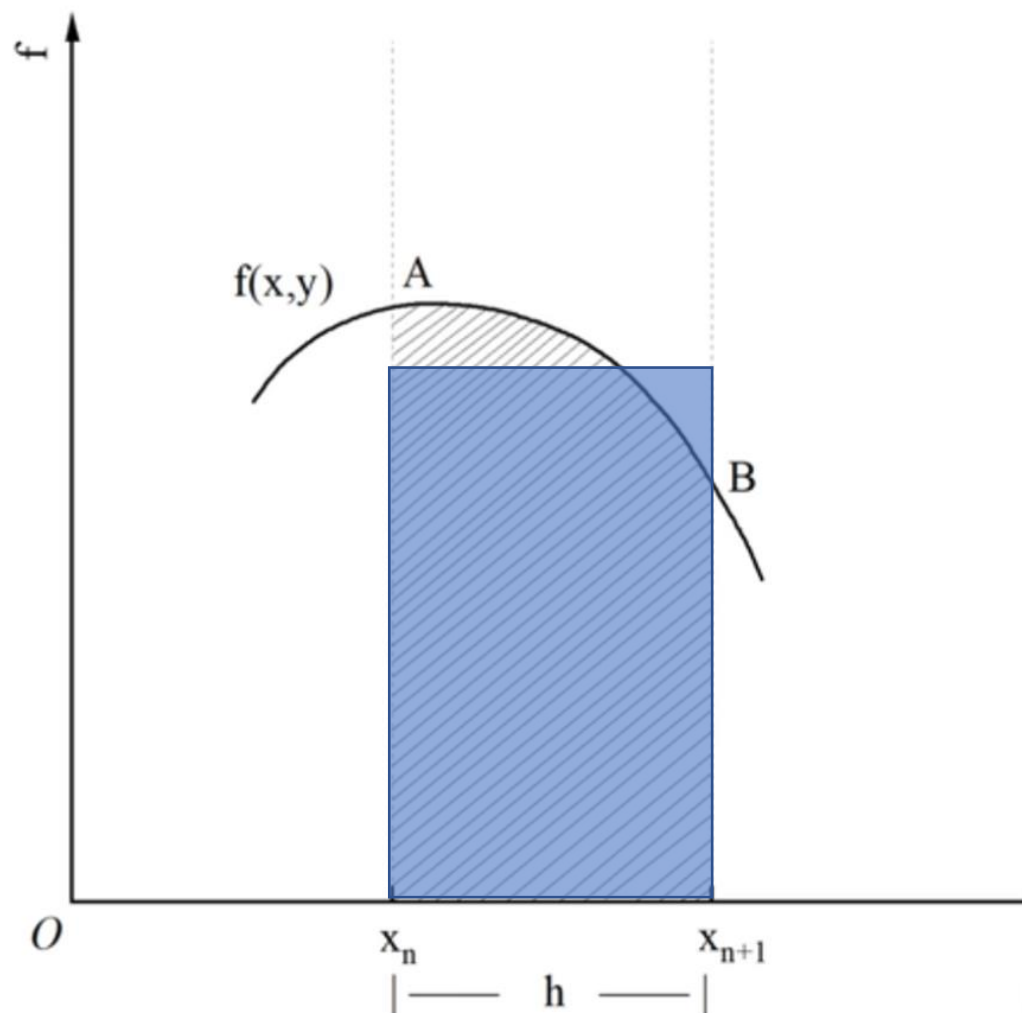
右端积分用**梯形求积公式**

$$\frac{h}{2} [f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))]$$

近似, 再用 y_{n+1} 代替 $y(x_{n+1})$, y_n 代替 $y(x_n)$, 即可得到

$$y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (6)$$

梯形Euler法的数值积分



四种Euler法的对比---数值积分角度

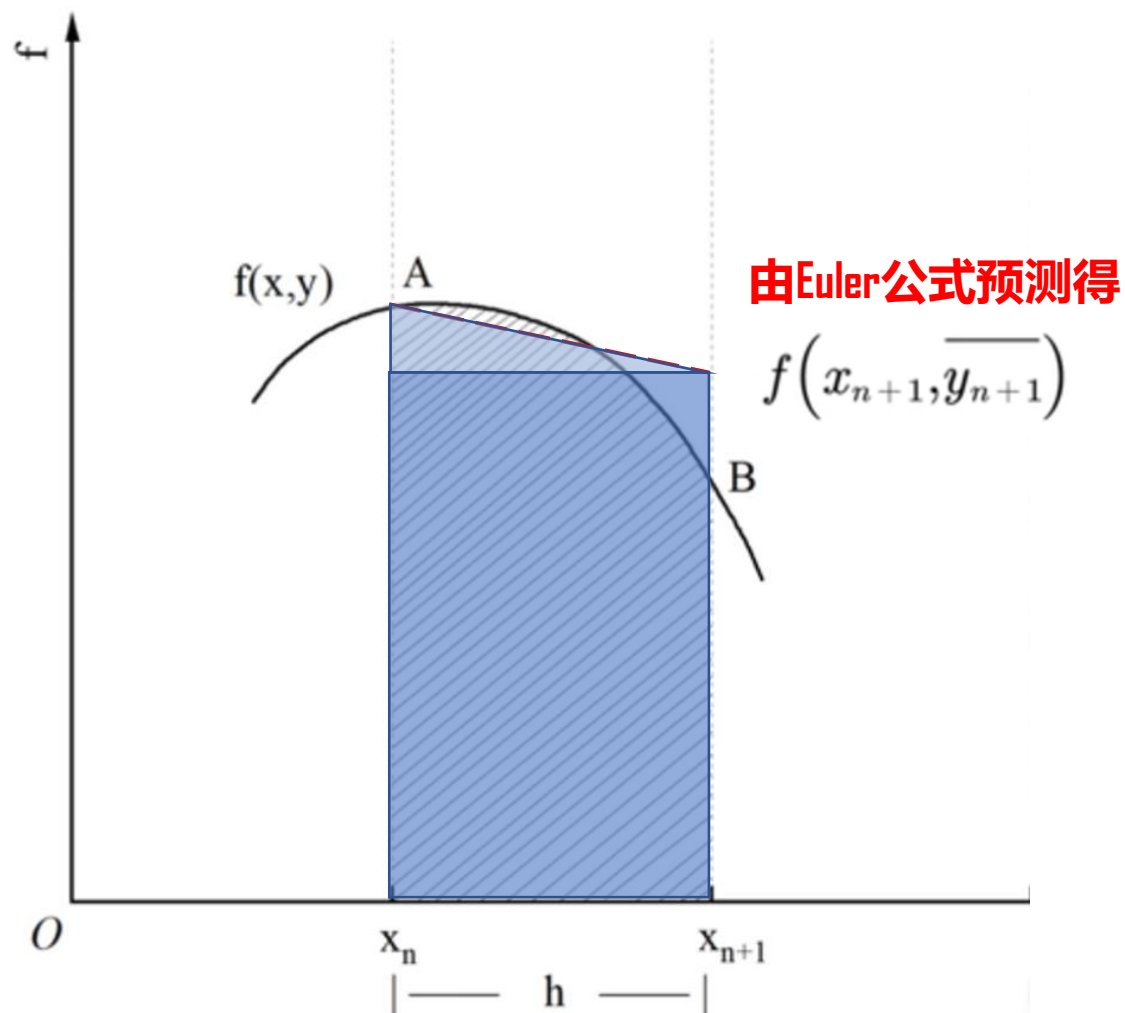
Euler法的数值积分

$$y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (6)$$

梯形公式 (6) 算法较为复杂

$$\begin{cases} \text{预测 } \overline{y}_{n+1} = y_n + hf(x_n, y_n), \\ \text{矫正 } y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, \overline{y}_{n+1})]. \end{cases}$$

$$\begin{cases} y_p = y_n + hf(x_n, y_n), \\ y_c = y_n + hf(x_{n+1}, y_p), \\ y_{n+1} = \frac{1}{2} (y_p + y_c). \end{cases}$$





各种Euler法的对比

- **向前欧拉法**: 以当前点的值为初值, 当前点的导数为导数, 计算下一个步长的点的值
- **向后欧拉法**: 以当前点的值为初值, 当下一个步长的点的导数为导数, 计算下一个步长的点的值, 但需要用最新的下一个点的值来计算导数, 不断迭代直到收敛
- **梯形方法**: 以当前点的值为初值, 当前点的导数和下一个步长的点的导数平均数为导数, 计算下一个步长的点的值, 但需要用最新的下一个点的值来计算导数, 不断迭代直到收敛
- **改进欧拉法**: 以当前点的值为初值, 当前点的导数和下一个步长的点的导数平均数为导数, 计算下一个步长的点的值, 不需要进行迭代计算, 只用计算一步即可

计算方法	计算量	计算精度	稳定性
显示欧拉法（向前）	小	低	差
隐式欧拉法（向后）	大	中	差
梯形欧拉法	大	高	好
改进欧拉法	中	高	好



龙格-库塔法

$$\begin{aligned} y(x_{n+1}) &= y(x_n + h) \\ &= y(x_n) + hy'(x_n) + \frac{h^2}{2!} y''(x_n) + \dots + \frac{h^r}{r!} y^{(r)}(x_n) + o(h^r) \end{aligned}$$

理论上讲上式阶数越高,精确度越高,但是计算量过大。

欧拉方法:

$$\begin{cases} y_{n+1} = y_n + hK_1 \\ K_1 = f(x_n, y_n) \end{cases}$$

改进欧拉法:

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

对许多实际问题来说, 欧拉公式与改进欧拉公式精度还不能满足要求, 为此从另一个角度来分析这两个公式的特点, 从而探索一条构造高精度方法的途径.



回顾

共同的特点是:

用 $f(x, y)$ 在某点上的值的线性组合来计算 y_{n+1}

基本想法:

- (1) 避免计算函数 $f(x, y)$ 的偏导数
- (2) 提高方法的精确度

Euler公式: 每步计算一次 $f(x, y)$ 的值, 为一阶方法。

改进Euler公式: 需计算两次 $f(x, y)$ 的值, 二阶方法。



龙格-库塔法的构建原理

初值问题的单步法可用一般形式表示为

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

其中多元函数 φ 与 $f(x, y)$ 有关，当 φ 含有 y_{n+1} 时，方法是隐式的，若不含 y_{n+1} 则为显式方法，所以显式单步法可表示为

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

$\varphi(x, y, h)$ 称为**增量函数**，例如对欧拉法有

$$\varphi(x_n, y_n, h) = f(x, y).$$

它的局部截断误差已给出，对一般显式单步法则可如下定义.



龙格-库塔法的构建原理

定义： 设 $y(x)$ 是初值问题的准确解，若存在最大整数 p 使显式单步法的局部截断误差满足

$$R_{n+1} = y(x+h) - y(x) - h\varphi(x, y, h) = O(h^{p+1})$$

则称方法具有 p 阶精度.

上节给出了显式单步法的表达式其局部截断误差为 $O(h^{p+1})$ ，对欧拉法 $R_{n+1}=O(h^2)$ ，即方法为 $p=1$ 阶，若用改进欧拉法，它可表为

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

其中增量函数为

$$\varphi(x_n, y_n, h) = \frac{1}{2} [f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))]$$



龙格-库塔法的构建原理

它比欧拉法的 $\varphi(x_n, y_n, h) = f(x_n, y_n)$, 增加了计算一个右函数 f 的值, 有望 $p=2$. 若要使得到的公式阶数 p 更大, φ 就必须包含更多的 f 值. 实际上根据

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

若要使公式阶数提高, 就必须使右端积分的数值求积公式精度提高, 它必然要增加求积节点, 为此可将的右端用求积公式表示为

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) dx \approx h \sum_{i=1}^r c_i f(x_n + \lambda_i h, y(x_n + \lambda_i h)).$$

牛顿-柯特斯公式



龙格-库塔法的构建原理

一般说来, 点数 r 越多, 精度越高, 上式右端相当于增量函数 $\varphi(x_n, y_n, h)$, 为得到便于计算的显式方法, 可类似于改进欧拉法, 将公式表示为

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

其中

$$\varphi(x_n, y_n, h) = \sum_{i=1}^r c_i K_i$$

$$K_1 = f(x_n, y_n),$$

$$K_i = f(x_n + \lambda_i h, y_n + h \sum_{j=1}^{i-1} \mu_{ij} K_j), i = 2, \dots, r,$$

这里 c_i, λ_i, μ_{ij} 均为常数. 上式称为 r 级显式龙格-库塔(Runge-Kutta)法, 简称R-K方法.



龙格-库塔法的构建

设近似公式为:

$$\begin{cases} y_{n+1} = y_n + h \sum_{i=1}^r c_i K_i \\ K_1 = f(x_n, y_n) \\ K_i = f(x_n + \lambda_i h, y_n + h \sum_{j=1}^{i-1} \mu_{ij} K_j) \end{cases} \quad i = 2, \dots, r,$$

这里 c_i, λ_i, μ_{ij} 均为常数.

上式称为 r 级Runge – Kutta法, 简称R – K方法.

当 $r=1$, $\varphi(x_n, y_n, h)=f(x_n, y_n)$ 时, 就是欧拉法, 此时方法的阶为 $p=1$. 当 $r=2$ 时, 改进欧拉法是其中一种, 下面将证明其阶 $p=2$. 要使公式具有更高的阶 p , 就要增加点数 r . 下面就 $r=2$ 推导R-K方法. 并给出 $r=3, 4$ 时的常用公式, 其推导方法与 $r=2$ 时类似, 只是计算较复杂.



龙格-库塔法的构造

二阶显式R-K方法

$$\begin{cases} y_{n+1} = y_n + h(c_1 K_1 + c_2 K_2), \\ K_1 = f(x_n, y_n), \\ K_2 = f(x_n + \lambda_2 h, y_n + \mu_{21} h K_1). \end{cases}$$

其中 $c_1, c_2, \lambda_2, \mu_{21}$ 均为待定常数，希望适当选取这些系数，使公式阶数 p 尽量高. 根据局部截断误差定义，推导出局部截断误差为

$$\begin{aligned} R_{n+1} = & y(x_{n+1}) - y(x_n) \\ & - h[c_1 f(x_n, y_n) + c_2 f(x_n + \lambda_2 h, y_n + \mu_{21} h f_n)] \end{aligned}$$

这里 $y_n = y(x_n), y_{n+1} = y(x_{n+1})$. 为得到 R_{n+1} 的阶 p ，要将上式各项在 (x_n, y_n) 处做泰勒展开。



龙格-库塔法的构造

$$y(x_{n+1}) = y_n + hy'_n + \frac{h^2}{2} y''_n + \frac{h^3}{3!} y'''_n + O(h^4),$$

其中

$$\begin{cases} y'_n = f(x_n, y_n) = f_n, \\ y''_n = \frac{d}{dx} f(x_n, y(x_n)) = f'_x(x_n, y_n) + f_n f'_y(x_n, y_n), \\ y'''_n = f''_{xx} + 2f_n f''_{xy} + f_n^2 f''_{yy} + f'_y[f'_x + f_n f'_y]. \end{cases}$$

$$\begin{aligned} & f(x_n + \lambda_2 h, y_n + \mu_{21} h f_n) \\ &= f_n + f'_x(x_n, y_n) \lambda_2 h + f'_y(x_n, y_n) \mu_{21} h f_n + O(h^2). \end{aligned}$$

将以上结果代入上式，则有

$$R_{n+1} = y(x_{n+1}) - y(x_n) - h(c_1 K_1 + c_2 K_2)$$



龙格-库塔法的构造

$$\begin{aligned} R_{n+1} &= y(x_{n+1}) - y(x_n) - h[c_1 f(x_n, y_n) + c_2 f(x_n + \lambda_2 h, y_n + \mu_{21} h f_n)] \\ &= h f_n + \frac{h^2}{2} [f'_x + f'_y f_n] + O(h^3) \\ &\quad - h[c_1 f_n + c_2 (f_n + f'_x \lambda_2 h + f'_y \mu_{21} h f_n)] + O(h^3) \\ &= (1 - c_1 - c_2) f_n h + \left(\frac{1}{2} - c_2 \lambda_2\right) f'_x(x_n, y_n) h^2 \\ &\quad + \left(\frac{1}{2} - c_2 \mu_{21}\right) f'_y(x_n, y_n) f_n h^2 + O(h^3). \end{aligned}$$

要使公式具有 $p=2$ 阶，必须使

$$1 - c_1 - c_2 = 0, \frac{1}{2} - c_2 \lambda_2 = 0, \frac{1}{2} - c_2 \mu_{21} = 0.$$

即
$$c_2 \lambda_2 = \frac{1}{2}, c_2 \mu_{21} = \frac{1}{2}, c_1 + c_2 = 1.$$

显然，解不唯一。



龙格-库塔法的构造

若令 $c_2=a \neq 0$ ，则得

这样得到的公式称为**二阶R-K方法**。

如取 $a=1/2$ ，则 $c_1=c_2=1/2$ ， $\lambda_2=\mu_{21}=1$ 。这就是**改进的欧拉公式**

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

由此可以看出在改进的欧拉公式中相当于取 (x_n, y_n) ， (x_{n+1}, y_{n+1}) 两点处**斜率的平均值**，近似代替平均斜率，其精度比欧拉公式提高了。



龙格-库塔法的构造

如取 $a=1$, 则 $c_1=0, c_2=1, \lambda_2=\mu_{21}=1/2$. 得计算公式

$$\begin{cases} y_{n+1} = y_n + hK_2, \\ K_1 = f(x_n, y_n), \\ K_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2} K_1). \end{cases}$$

称为中点公式(变形的欧拉公式), 相当于数值积分的中矩形公式.也可以表示为

$$y_{n+1} = y_n + hf(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n)).$$



龙格-库塔法的构造

依次推导，就可以给出 $r = 3, 4$ 时的 $R-K$ 常用公式，

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(K_1 + 4K_2 + K_3), \\ K_1 = f(x_n, y_n), \\ K_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right), \\ K_3 = f(x_n + h, y_n - hK_1 + 2hK_2). \end{cases}$$

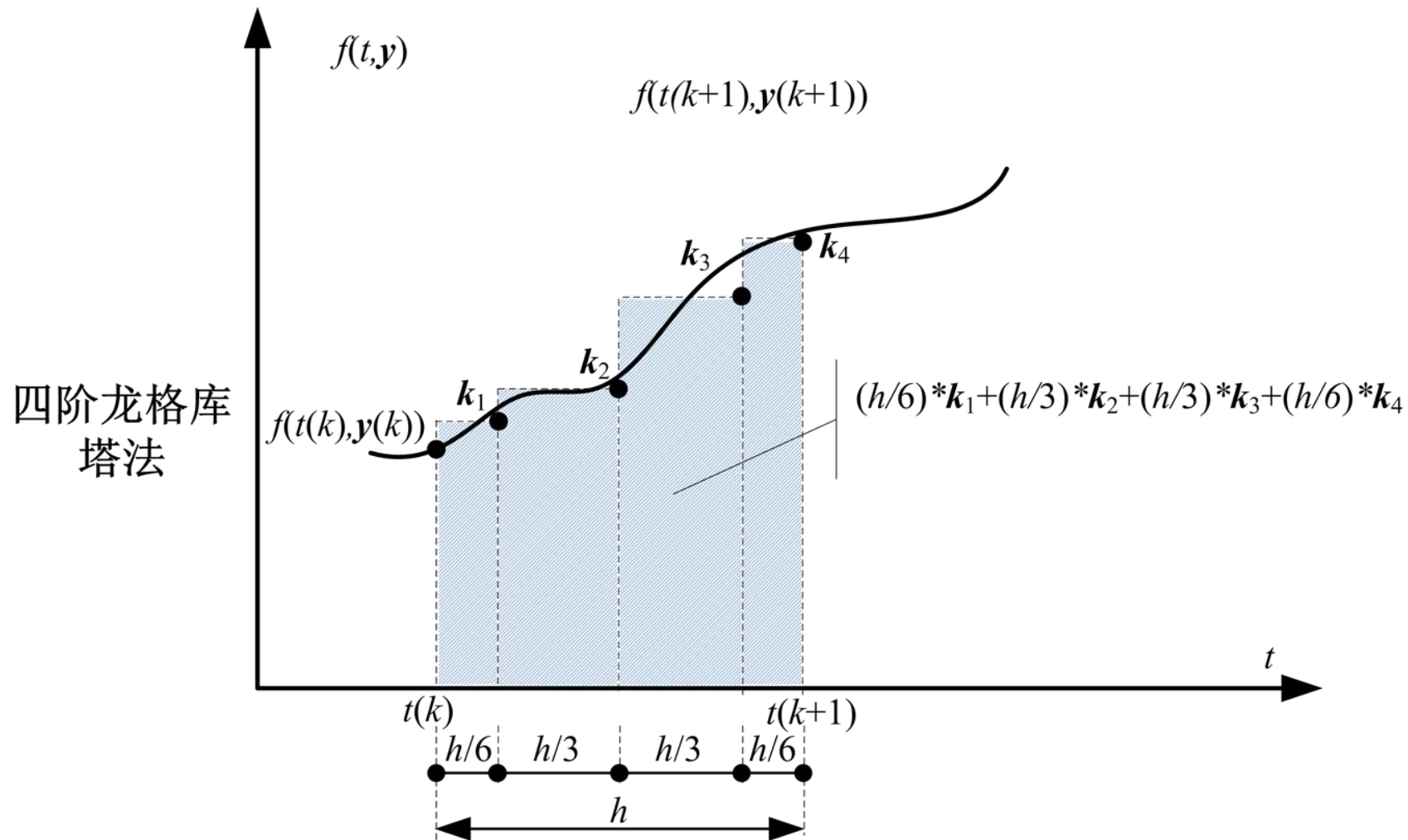
称为三阶R-K方法.

四阶经典R-K公式

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

四阶R-K方法的每一步需要计算四次函数值 f ，可以证明其局部截断误差为 $O(h^5)$.

龙格-库塔法的几何意义



龙格-库塔法的典型例题

例 1

求(3阶、4阶 $R-K$ 公式)解初值问题(步长 $h = 0.1$)

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

解

$$f(x, y) = y - 2x / y$$

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(K_1 + 4K_2 + K_3), \\ K_1 = f(x_n, y_n), \\ K_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right), \\ K_3 = f(x_n + h, y_n - hK_1 + 2hK_2). \end{cases}$$



龙格-库塔法的构建原理

结果比较

精确解

$y[0] \rightarrow 1$
$y[0.1] \rightarrow 1.09545$
$y[0.2] \rightarrow 1.18322$
$y[0.3] \rightarrow 1.26491$
$y[0.4] \rightarrow 1.34164$
$y[0.5] \rightarrow 1.41421$
$y[0.6] \rightarrow 1.48324$

改进Euler近似解

0	1.
0.1	1.09774
0.2	1.18757
0.3	1.27129
0.4	1.35013
0.5	1.42499
0.6	1.49657

3阶R-K近似解

0	1.
0.1	1.09544
0.2	1.18322
0.3	1.26491
0.4	1.34165
0.5	1.41422
0.6	1.48326

- 改进欧拉法使用一次迭代和梯形计算公式，可以实现比欧拉法更高的计算精度和稳定性。
- 龙格-库塔法为单步法高精度计算提供了解决方案，其中三阶和四阶公式为常用的求解常微分方程的解法。



肖明

中山大学



谢谢大家！

肖 明

微电子科学与技术学院

邮箱: xiaom37@mail.sysu.edu.cn

电话: 1781722706

办公室: 中山大学珠海校区公共实验楼2楼A214室