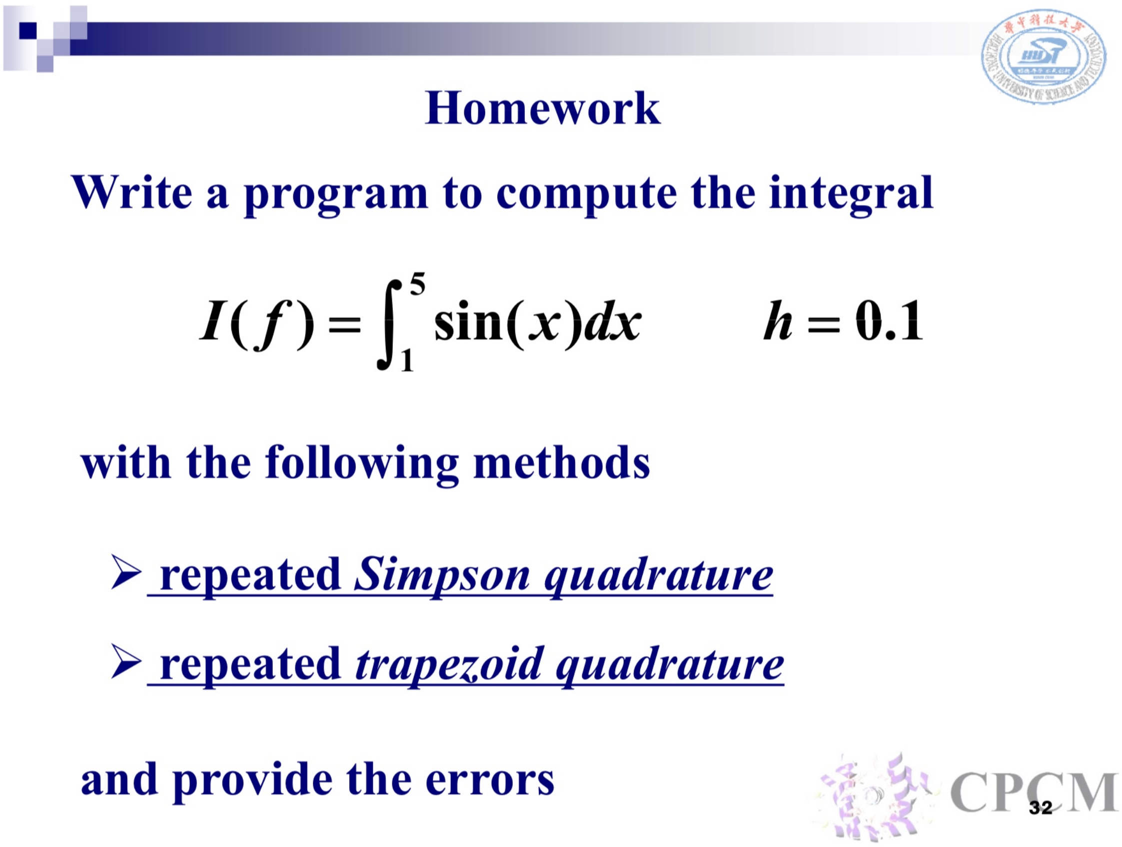
Differentiation and Integration

曹玉清 天眷1701班 U201710185

一、Repeated Simpson quadrature and Repeated trapezoid quadrature

1、问题描述



2、主要算法与公式

Repeated trapezoid quadrature

$$h=\frac{b-a}{n},\ x_i=a+ih,\ i=0,\cdots,n$$

$$\int_{x_i}^{x_{i+1}} f(x)=\frac{h}{2}\big(f(x_i)+f(x_{i+1})\big)$$

$$\boxed{\begin{aligned}T_n(f)&=\sum_{i=0}^{n-1}\left(\frac{h}{2}(f(x_i)+f(x_{i+1})\right)\\&=\frac{h}{2}\left(f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right)\end{aligned}}$$

CPCM

Repeated *Simpson* quadrature

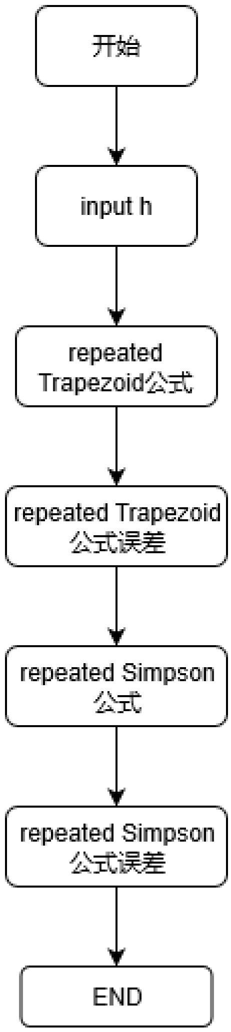
$$h=\frac{b-a}{n},x_i=a+ih,i=0,\cdots,n;n=2m$$

$$\int_{x_{2i}}^{x_{2i+2}} f(x)=\frac{2h}{6}\big(f(x_{2i})+4f(x_{2i+1})+f(x_{2i+2})\big)$$

$$\boxed{S_n(f)=\frac{h}{3}\left(f(a)+4\sum_{i=0}^{m-1}f(x_{2i+1})+2\sum_{i=1}^{m-1}f(x_{2i})+f(b)\right)}$$

CPCM

3、流程图



3、源代码

```
program ST
implicit none
    real*8 :: sum_fxk,h,sum_odd,sum_even
    real*8 :: trape_40,simp_40
    real*8 :: trape_20,simp_20
    real*8 :: errorTrape,errorSimp
    integer :: k

    h = 0.1
    sum_fxk = 0.0
    do k = 1,39
        sum_fxk = sum_fxk + sin(1.0 + k*h)
    end do
    trape_40 = 1/20.0 * ( sin(1.0) + 2*sum_fxk + sin(5.0) )

    sum_fxk = 0.0
    do k = 1,19
        sum_fxk = sum_fxk + sin(1.0 + k*h)

    end do
```

```
    trape_20 = 1/20.0 * ( sin(1.0) + 2*sum_fxk + sin(5.0) )
    errorTrape = 1/3.0 * (trape_40 -trape_20)

    print "(a)", "using repeated trapezoid quadrature:"
    print "(f8.6)", trape_40
    print "(a)", "repeated trapezoid quadrature error analysis:"
    print *, errorTrape
    print "(a)", "--------------------------------------------------"

    sum_odd = 0.0
    sum_even = 0.0
    k = 1
    do while (k<=39)
        sum_odd = sum_odd + sin(1.0+k*h)
        k = k+2
    end do
    k = 2
    do while (k<=39)
        sum_even = sum_even + sin(1.0+k*h)
        k = k+2
    end do
    simp_40 = 1/30.0 * ( sin(1.0) + 4*sum_odd + 2*sum_even +sin(5.0) )

    sum_odd = 0.0
    sum_even = 0.0
    k = 1
    do while (k<=19)
        sum_odd = sum_odd + sin(1.0+k*h)
        k = k+2
    end do
    k = 2
    do while (k<=19)
        sum_even = sum_even + sin(1.0+k*h)
        k = k+2
    end do
    simp_20 = 1/30.0 * ( sin(1.0) + 4*sum_odd + 2*sum_even +sin(5.0) )
    errorSimp = 1/15.0 * (simp_40 - simp_20)

    print "(a)", "using repeated Simpson quadrature:"
    print "(f8.6)", simp_40
    print "(a)", "repeated Simpson quadrature error analysis:"
    print *, errorSimp

end program ST
```

4、运行时结果

```
E:\cyq\cyq essays\Phy HUST\计算物理\hw\cyq\5\code>ST.exe
using repeated trapezoid quadrature:
0.256426
repeated trapezoid quadrature error analysis:
 -0.405863670384824l8
-------------------------------------------------
using repeated Simpson quadrature:
0.256640
repeated Simpson quadrature error analysis:
  -8.246582804623237OE-002
```

5、误差分析与画图

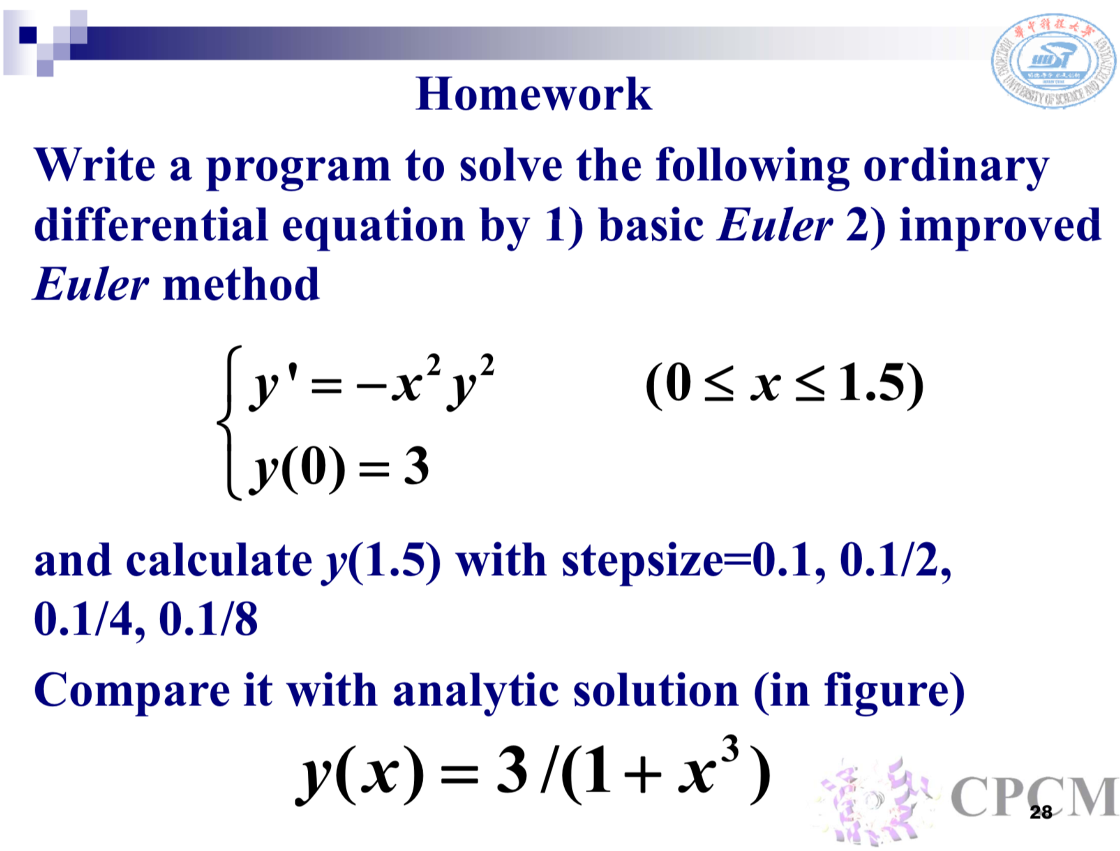
repeated trapezoid quadrature的绝对误差是 0.405864

repeated Simpson quadrature的绝对误差是 0.082466

Ordinary Differential Equation

一、Basic Euler and improved Euler method

1、问题描述



2、主要算法与公式

Error for basic Euler method

$$y(x_{n+1}) - y_{n+1}^{(h)} = O(h^2) \approx ch^2$$ step size h

$$y(x_{n+1}) - y_{n+1}^{(h/2)} = 2O\left((h/2)^2\right) \approx 2c\left(h/2\right)^2$$

step size h/2

$$\frac{y(x_{n+1}) - y_{n+1}^{(h/2)}}{y(x_{n+1}) - y_{n+1}^{(h)}} \approx \frac{1}{2}$$

$$y(x_{n+1}) = 2y_{n+1}^{(h/2)} - y_{n+1}^{(h)}$$

$$\begin{cases} y(x_{n+1}) - y_{n+1}^{(h/2)} \approx y_{n+1}^{(h/2)} - y_{n+1}^{(h)} \\ y(x_{n+1}) - y_{n+1}^{(h)} \approx 2(y_{n+1}^{(h/2)} - y_{n+1}^{(h)}) \end{cases}$$

CPCM

Error for basic Euler method

This is only the error of one rectangle integration

$$y(x_{n+1}) - y_{n+1}^{(h)} \approx 2(y_{n+1}^{(h/2)} - y_{n+1}^{(h)})$$

The total error for y_{n+1} (n+1 rectangles) is

$$E(h) = y(x_{n+1}) - y_{n+1}^{(h)} = \sum_{i=1}^{n+1} 2(y_i^{(h/2)} - y_i^{(h)})$$

CPCM

Error for improved Euler method

$$y(x_{n+1}) - y_{n+1}^{(h)} = O(h^3) \approx ch^3$$ **step size h**

$$y(x_{n+1}) - y_{n+1}^{(h/2)} = 2O\left((h/2)^2\right) \approx 2c\left(h/2\right)^3$$

step size h/2

$$\frac{y(x_{n+1}) - y_{n+1}^{(h/2)}}{y(x_{n+1}) - y_{n+1}^{(h)}} \approx \frac{1}{4}$$

$$y(x_{n+1}) = \frac{4y_{n+1}^{(h/2)} - y_{n+1}^{(h)}}{3}$$

$$\begin{cases} y(x_{n+1}) - y_{n+1}^{(h/2)} \approx \dfrac{y_{n+1}^{(h/2)} - y_{n+1}^{(h)}}{3} \\ y(x_{n+1}) - y_{n+1}^{(h)} \approx \dfrac{4}{3}\left(y_{n+1}^{(h/2)} - y_{n+1}^{(h)}\right) \end{cases}$$

Error for improve Euler method

This is only the error of one trapezoid integration

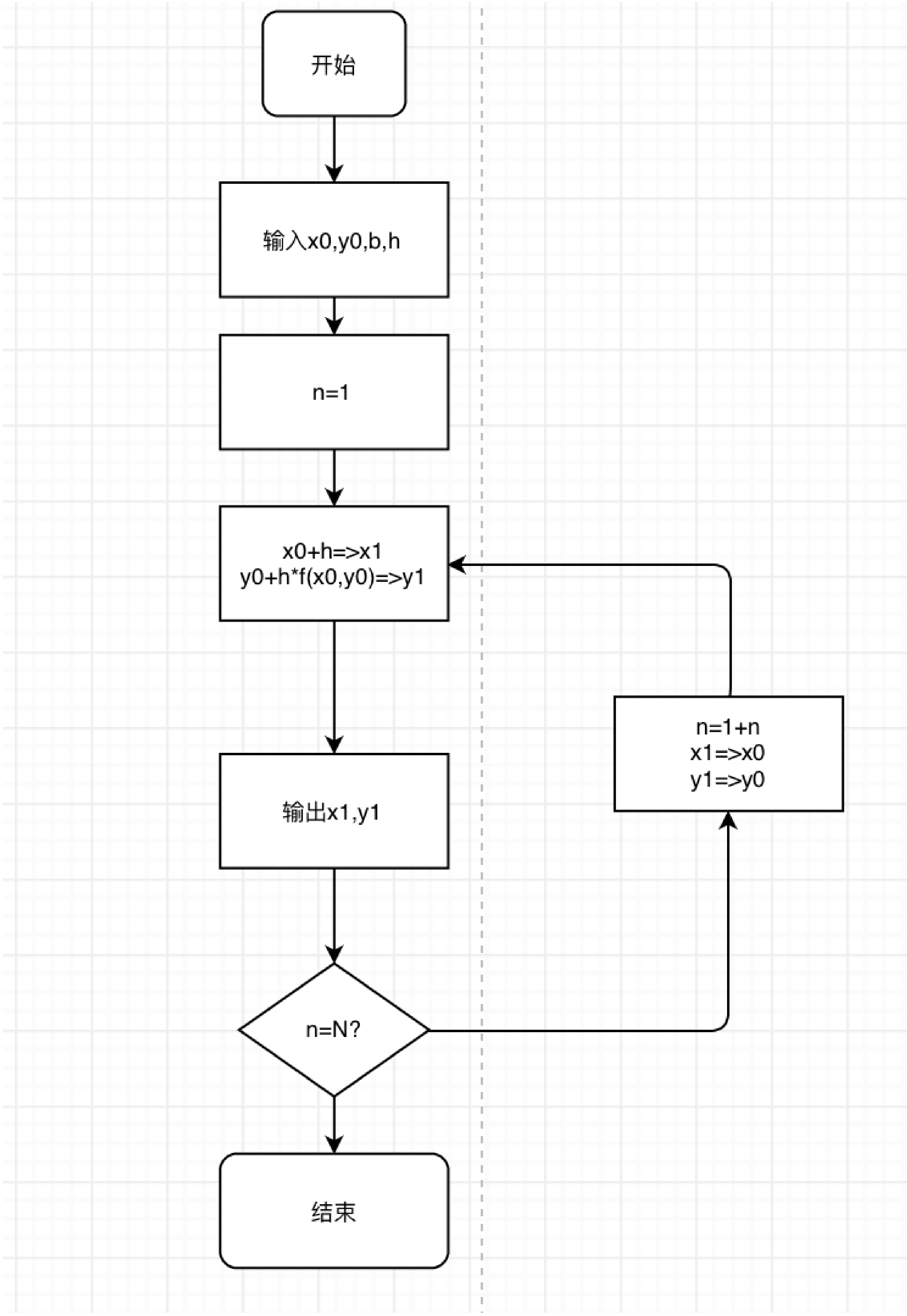
$$y(x_{n+1}) - y_{n+1}^{(h)} \approx \frac{4}{3}\left(y_{n+1}^{(h/2)} - y_{n+1}^{(h)}\right)$$

The total error for y_{n+1} (n+1 trapezoids) is

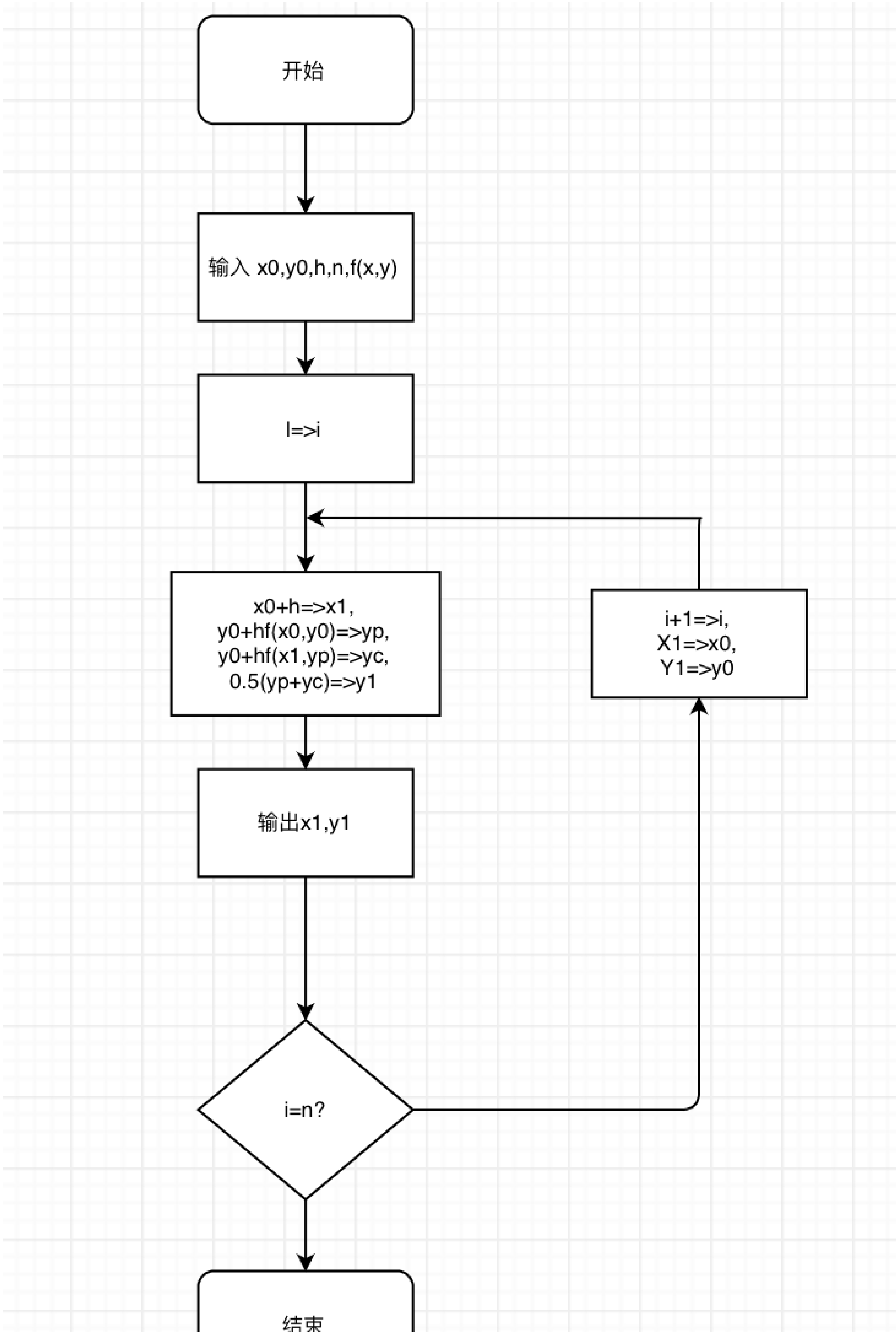
$$E(h) = y(x_{n+1}) - y_{n+1}^{(h)} = \sum_{i=1}^{n+1} \frac{4}{3}\left(y_i^{(h/2)} - y_i^{(h)}\right)$$

3、流程图

1）Basic Euler method


开始
输入x0,y0,b,h
n=1
x0+h=>x1
y0+h*f(x0,y0)=>y1
输出x1,y1
n=N?
结束
n=1+n
x1=>x0
y1=>y0

2) improved Euler method


开始
输入 x0,y0,h,n,f(x,y)
I=>i
x0+h=>x1,
y0+hf(x0,y0)=>yp,
y0+hf(x1,yp)=>yc,
0.5(yp+yc)=>y1
i+1=>i,
X1=>x0,
Y1=>y0
输出x1,y1
i=n?
结束

3、源代码

```
program RK
implicit none

    real*8 :: fxy

    real*8 :: h
    real*8 :: K1,K2,K3,K4
    real*8 :: xn
    integer :: interval
    real*8 :: y1(16),y2(31),y3(61),y4(121)

    h = 0.1
    y1(1) = 3.0
    do interval = 1,15
        xn = 0.0 + interval * h
        K1 = fxy( xn,y1(interval) )
        K2 = fxy( xn+h/2.0, y1(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y1(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y1(interval)+h*K3)
        y1(interval+1) = y1(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(16f8.3)", y1(:)
    print "(a)", "-----------------------------------------------------------------"

    h = 0.1/2.0
    y2(1) = 3.0
    do interval = 1,30
        xn = 0.0 + interval * h
        K1 = fxy( xn,y2(interval) )
        K2 = fxy( xn+h/2.0, y2(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y2(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y2(interval)+h*K3)
        y2(interval+1) = y2(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(31f8.3)", y2(:)
    print "(a)", "-----------------------------------------------------------------"

    h = 0.1/4.0
    y3(1) = 3.0
    do interval = 1,60
        xn = 0.0 + interval * h
        K1 = fxy( xn,y3(interval) )

        K2 = fxy( xn+h/2.0, y3(interval)+h/2.0*K1 )
```

```fortran
        K3 = fxy( xn+h/2.0, y3(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y3(interval)+h*K3)
        y3(interval+1) = y3(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(31f8.3)", y3(:)
    print "(a)", "------------------------------------------------------------------"

    h = 0.1/8.0
    y4(1) = 3.0
    do interval = 1,120
        xn = 0.0 + interval * h
        K1 = fxy( xn,y4(interval) )
        K2 = fxy( xn+h/2.0, y4(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y4(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y4(interval)+h*K3)
        y4(interval+1) = y4(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(31f8.3)", y4(:)
    print "(a)", "------------------------------------------------------------------"

end program RK

function fxy(x,y)
implicit none
    real*8 :: x,y
    real*8 :: fxy
    fxy = - x**2 * y**2
end function
```

4、运行时结果

```
E:\cyq\cyq essays\Phy HUST\计算物理\hw\cyq\5\code>RK.exe
when stepsize = 0.1, y= :
   3.000   2.979   2.924   2.822   2.669   2.469   2.235   1.985   1.736   1.501   1.288   1.100   0.939   0.802   0.686   0.589
------------------------------------------------------------------
when stepsize = 0.1, y= :
   3.000   2.997   2.990   2.977   2.954   2.921   2.877   2.820   2.750   2.667   2.572   2.467   2.354   2.234   2.110   1.984
  1.859   1.735   1.615   1.500   1.390   1.287   1.190   1.100   1.016   0.938   0.867   0.801   0.741   0.686   0.635
------------------------------------------------------------------
when stepsize = 0.1, y= :
   3.000   3.000   2.999   2.997   2.994   2.990   2.984   2.976   2.966   2.954   2.939   2.921   2.900   2.877   2.850   2.820
  2.786   2.749   2.710   2.667   2.621   2.572   2.521   2.467   2.411   2.354   2.294   2.234   2.172   2.110   2.047
   1.984   1.921   1.859   1.797   1.735   1.675   1.615   1.557   1.500   1.444   1.390   1.338   1.287   1.238   1.190   1.144
  1.100   1.057   1.016   0.976   0.938   0.902   0.867   0.833   0.801   0.770   0.741   0.713   0.686   0.660
------------------------------------------------------------------
when stepsize = 0.1, y= :
   3.000   3.000   3.000   3.000   2.999   2.999   2.998   2.997   2.996   2.994   2.992   2.990   2.987   2.984   2.980   2.976
  2.971   2.966   2.960   2.954   2.947   2.939   2.930   2.921   2.911   2.900   2.889   2.877   2.864   2.850   2.835
   2.820   2.803   2.786   2.768   2.749   2.730   2.710   2.689   2.667   2.644   2.621   2.597   2.572   2.547   2.521   2.494
  2.467   2.439   2.411   2.383   2.354   2.324   2.294   2.264   2.234   2.203   2.172   2.141   2.110   2.079   2.047
   2.016   1.984   1.953   1.921   1.890   1.859   1.827   1.796   1.766   1.735   1.705   1.675   1.645   1.615   1.586   1.557
  1.528   1.500   1.472   1.444   1.417   1.390   1.364   1.338   1.312   1.287   1.262   1.238   1.214   1.190   1.167
   1.144   1.122   1.100   1.078   1.057   1.036   1.016   0.996   0.976   0.957   0.938   0.920   0.902   0.884   0.867   0.850
  0.833   0.817   0.801   0.786   0.770   0.756   0.741   0.727   0.713   0.699   0.686   0.673
------------------------------------------------------------------
```

二、Runge-Kutta method

1、问题描述

Homework

Write a program to solve the following ordinary differential equation by four-order *Runge-Kutta* method

$$\begin{cases} y' = -x^2 y^2 & (0 \le x \le 1.5) \\ y(0) = 3 \end{cases}$$

and calculate $y(1.5)$ with stepsize=0.1, 0.1/2, 0.1/4, 0.1/8

Compare it with analytic solution (in figure)

$$y(x) = 3/(1+x^3)$$

2、主要算法与公式

Runge-Kutta Method with 4-order precision

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

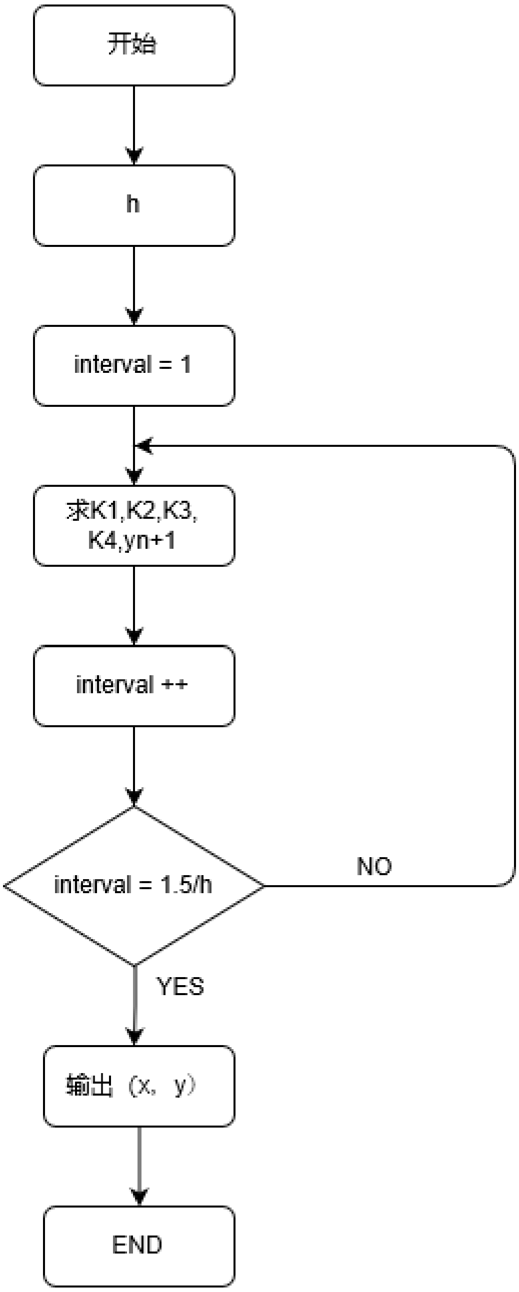
CPCM

Runge-Kutta Method with m-order precision

$$\begin{cases} F(h, x, y, f) = c_1K_1 + c_2K_2 + \cdots + c_mK_m \\ K_1 = f(x, y) \\ K_2 = f(x + a_2h, y + b_{21}hK_1) \\ \vdots \\ K_m = f(x + a_mh, y + h\sum_{i=1}^{m-1} b_{mi}K_i) \end{cases}$$

CPCM

3、流程图



3、源代码

```
program RK
implicit none

    real*8 :: fxy

    real*8 :: h
    real*8 :: K1,K2,K3,K4
    real*8 :: xn
    integer :: interval
    real*8 :: y1(16),y2(31),y3(61),y4(121)

    h = 0.1
    y1(1) = 3.0

    do interval = 1,15
```

```
        xn = 0.0 + interval * h
        K1 = fxy( xn,y1(interval) )
        K2 = fxy( xn+h/2.0, y1(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y1(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y1(interval)+h*K3)
        y1(interval+1) = y1(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(16f8.3)", y1(:)
    print "(a)", "------------------------------------------------------------------"

    h = 0.1/2.0
    y2(1) = 3.0
    do interval = 1,30
        xn = 0.0 + interval * h
        K1 = fxy( xn,y2(interval) )
        K2 = fxy( xn+h/2.0, y2(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y2(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y2(interval)+h*K3)
        y2(interval+1) = y2(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(31f8.3)", y2(:)
    print "(a)", "------------------------------------------------------------------"

    h = 0.1/4.0
    y3(1) = 3.0
    do interval = 1,60
        xn = 0.0 + interval * h
        K1 = fxy( xn,y3(interval) )
        K2 = fxy( xn+h/2.0, y3(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y3(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y3(interval)+h*K3)
        y3(interval+1) = y3(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

    print "(a)", "when stepsize = 0.1, y= :"
    print "(31f8.3)", y3(:)
    print "(a)", "------------------------------------------------------------------"

    h = 0.1/8.0
    y4(1) = 3.0
    do interval = 1,120
        xn = 0.0 + interval * h
        K1 = fxy( xn,y4(interval) )
        K2 = fxy( xn+h/2.0, y4(interval)+h/2.0*K1 )
        K3 = fxy( xn+h/2.0, y4(interval)+h/2.0*K2 )
        K4 = fxy( xn+h, y4(interval)+h*K3)
        y4(interval+1) = y4(interval) + h/6.0 * ( K1+2*K2+2*K3+K4 )
    end do

```

```fortran
    print "(a)", "when stepsize = 0.1, y= :"
    print "(31f8.3)", y4(:)
    print "(a)", "-----------------------------------------------------------------"

end program RK

function fxy(x,y)
implicit none
    real*8 :: x,y
    real*8 :: fxy
    fxy = - x**2 * y**2
end function
```

4、运行时结果

```
E:\cyq\cyq essays\Phy HUST\计算物理\hw\cyq\5\code>RK.exe
when stepsize = 0.1, y= :
   3.000   2.979   2.924   2.822   2.669   2.469   2.235   1.985   1.736   1.501   1.288   1.100   0.939   0.802   0.686   0.589
-----------------------------------------------------------------
when stepsize = 0.1, y= :
   3.000   2.997   2.990   2.977   2.954   2.921   2.877   2.820   2.750   2.667   2.572   2.467   2.354   2.234   2.110   1.984
  1.859   1.735   1.615   1.500   1.390   1.287   1.190   1.100   1.016   0.938   0.867   0.801   0.741   0.686   0.635
-----------------------------------------------------------------
when stepsize = 0.1, y= :
   3.000   3.000   2.999   2.997   2.994   2.990   2.984   2.976   2.966   2.954   2.939   2.921   2.900   2.877   2.850   2.820
  2.786   2.749   2.710   2.667   2.621   2.572   2.521   2.467   2.411   2.354   2.294   2.234   2.172   2.110   2.047
   1.984   1.921   1.859   1.797   1.735   1.675   1.615   1.557   1.500   1.444   1.390   1.338   1.287   1.238   1.190   1.144
  1.100   1.057   1.016   0.976   0.938   0.902   0.867   0.833   0.801   0.770   0.741   0.713   0.686   0.660
-----------------------------------------------------------------
when stepsize = 0.1, y= :
   3.000   3.000   3.000   3.000   2.999   2.999   2.998   2.997   2.996   2.994   2.992   2.990   2.987   2.984   2.980   2.976
  2.971   2.966   2.960   2.954   2.947   2.939   2.930   2.921   2.911   2.900   2.889   2.877   2.864   2.850   2.835
   2.820   2.803   2.786   2.768   2.749   2.730   2.710   2.689   2.667   2.644   2.621   2.597   2.572   2.547   2.521   2.494
  2.467   2.439   2.411   2.383   2.354   2.324   2.294   2.264   2.234   2.203   2.172   2.141   2.110   2.079   2.047
   2.016   1.984   1.953   1.921   1.890   1.859   1.827   1.796   1.766   1.735   1.705   1.675   1.645   1.615   1.586   1.557
  1.528   1.500   1.472   1.444   1.417   1.390   1.364   1.338   1.312   1.287   1.262   1.238   1.214   1.190   1.167
   1.144   1.122   1.100   1.078   1.057   1.036   1.016   0.996   0.976   0.957   0.938   0.920   0.902   0.884   0.867   0.850
  0.833   0.817   0.801   0.786   0.770   0.756   0.741   0.727   0.713   0.699   0.686   0.673
-----------------------------------------------------------------
```