Ordinary Differential Equation

报告主要内容

> **Write a program to solve the following ordinary differential equation by**
>
> - basic Euler method
> - improved Euler method
> - four-order Runge-Kutta method
>
> $$\begin{cases} y' = -x^2 y^2 \\ y(0) = 3 \end{cases} \quad x \in [0, 1.5]$$
>
> **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)**
>
> $$y(x) = \frac{3}{1+x^3}$$

主程序

```
program main
    use ODE
    implicit none
    real :: x0=0.0,y0=3.0,a=0.0,b=1.5,h=0.1
    integer :: n,i,j
    character(len=512) :: filename


    print *,'The analytic solution of y(1.5)=
',analytic_solution(1.5)
    print *

    print *,'Basic Euler Method'
    print '(5x,a,f3.1,3x,a,f3.1,3x,a,f3.1,3x,a,f3.1)','Set
initial  x0= ',x0,'y0= ',y0,'a= ',a,'b= ',b
```

```
    do j=0,3
        h=0.1/2**j
        print '(5x,a,f6.4)','Set h= ',h
        call init(x0,y0,a,b,h)
        n=size(x)-1
        call Basic_Euler
        print '(8x,a,f10.8)','y(1.5)= ',y(n)
        write(filename, *) j
        filename='BEM_'//Trim(AdjustL(filename))//'.txt'
        open(101,file=filename)
        write(101,'(f3.1,f10.8)') (x(i),y(i),i=0,n)
        close(101)
        deallocate(x,y)
    end do
    print *

    print *,'Improved Euler Method'
    print '(5x,a,f3.1,3x,a,f3.1,3x,a,f3.1,3x,a,f3.1)','Set
initial  x0= ',x0,'y0= ',y0,'a= ',a,'b= ',b
    do j=0,3
        h=0.1/2**j
        print '(5x,a,f6.4)','Set h= ',h
        call init(x0,y0,a,b,h)
        n=size(x)-1
        call Improved_Euler
        print '(8x,a,f10.8)','y(1.5)= ',y(n)
        write(filename, *) j
        filename='IEM_'//Trim(AdjustL(filename))//'.txt'
        open(101,file=filename)
        write(101,'(f3.1,f10.8)') (x(i),y(i),i=0,n)
        close(101)
        deallocate(x,y)
    end do
    print *

    print *,'Four Order Runge-Kutta Method'
    print '(5x,a,f3.1,3x,a,f3.1,3x,a,f3.1,3x,a,f3.1)','Set
initial  x0= ',x0,'y0= ',y0,'a= ',a,'b= ',b
    do j=0,3
        h=0.1/2**j
        print '(5x,a,f6.4)','Set h= ',h
        call init(x0,y0,a,b,h)
        n=size(x)-1
        call Four_Order_Runge_Kutta
        print '(8x,a,f10.8)','y(1.5)= ',y(n)
        write(filename, *) j
        filename='RKM_'//Trim(AdjustL(filename))//'.txt'
        open(101,file=filename)
```

```
        write(101,'(f3.1,f10.8)') (x(i),y(i),i=0,n)
        close(101)
        deallocate(x,y)
    end do
    print *

end program main
```

求解ODE方程的关键方法写在ODE模块中:

```
module ODE
    implicit none
    private
    public ::
x,y,init,Basic_Euler,Improved_Euler,Four_Order_Runge_Kutta,analy
tic_solution

    real,allocatable :: x(:),y(:)
    real :: x0,y0,a,b,h
    integer :: n,i

contains

    function f(x,y)
        implicit none
        real :: f,x,y
        f=-x*x*y*y
    end function f

    function analytic_solution(x) result(f)
        implicit none
        real :: f,x
        f=3.0/(1+x*x*x)
    end function

    subroutine init(x0_,y0_,a_,b_,h_)
        implicit none
        real :: x0_,y0_,a_,b_,h_
        x0=x0_
        y0=y0_
        a=a_
        b=b_
        h=h_
        n=int((b-a)/h)
        allocate(x(0:n),y(0:n))
        x=(/ (a+i*h,i=0,n) /)
```

```
        y=0
        y(0)=y0
    end subroutine init

    subroutine Basic_Euler()
        implicit none
        do i=1,n
            y(i)=y(i-1)+h*f(x(i-1),y(i-1))
        end do
    end subroutine Basic_Euler

    subroutine Improved_Euler()
        implicit none
        real :: y_
        do i=1,n
            y_=y(i-1)+h*f(x(i-1),y(i-1))
            y(i)=y(i-1)+h/2*(f(x(i-1),y(i-1))+f(x(i),y_))
        end do
    end subroutine Improved_Euler

    subroutine Four_Order_Runge_Kutta()
        implicit none
        real :: k1,k2,k3,k4
        do i=1,n
            k1=f(x(i-1),y(i-1))
            k2=f(x(i-1)+h/2,y(i-1)+h/2*k1)
            k3=f(x(i-1)+h/2,y(i-1)+h/2*k2)
            k4=f(x(i-1)+h,y(i-1)+h*k3)
            y(i)=y(i-1)+h/6*(k1+2*k2+2*k3+k4)
        end do
    end subroutine Four_Order_Runge_Kutta
end module ODE
```

其中`init`方法是用来初始化:

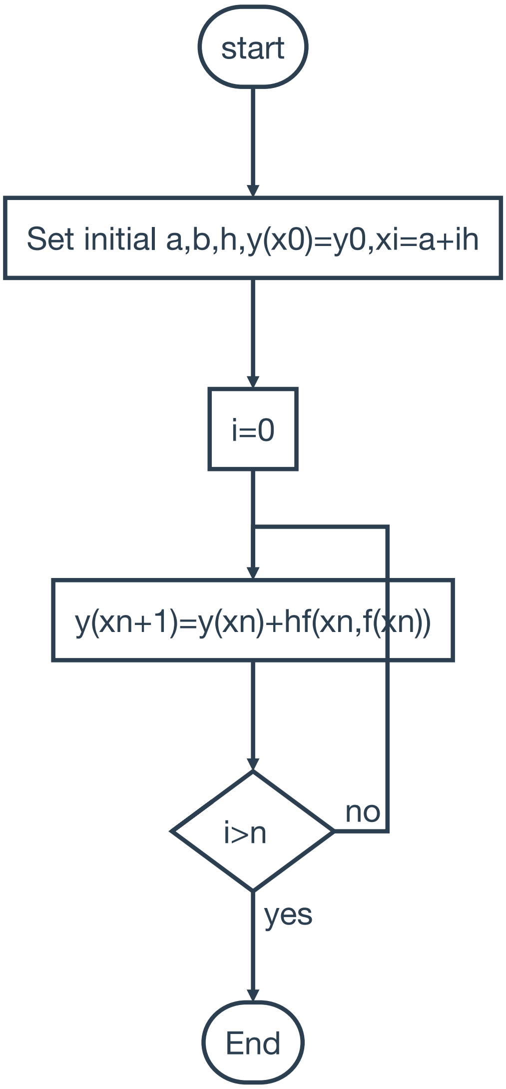
```
    subroutine init(x0_,y0_,a_,b_,h_)
        implicit none
        real :: x0_,y0_,a_,b_,h_
        x0=x0_
        y0=y0_
        a=a_
        b=b_
        h=h_
        n=int((b-a)/h)
        allocate(x(0:n),y(0:n))
        x=(/ (a+i*h,i=0,n) /)
        y=0
        y(0)=y0
    end subroutine init
```

`analytic_solution`是数值解:

```
    function analytic_solution(x) result(f)
        implicit none
        real :: f,x
        f=3.0/(1+x*x*x)
    end function
```

Basic Euler Method

流程图:



原理:

$$\begin{cases} y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n)) + O(h^2) \\ y(x_0) = y_0 \end{cases}$$

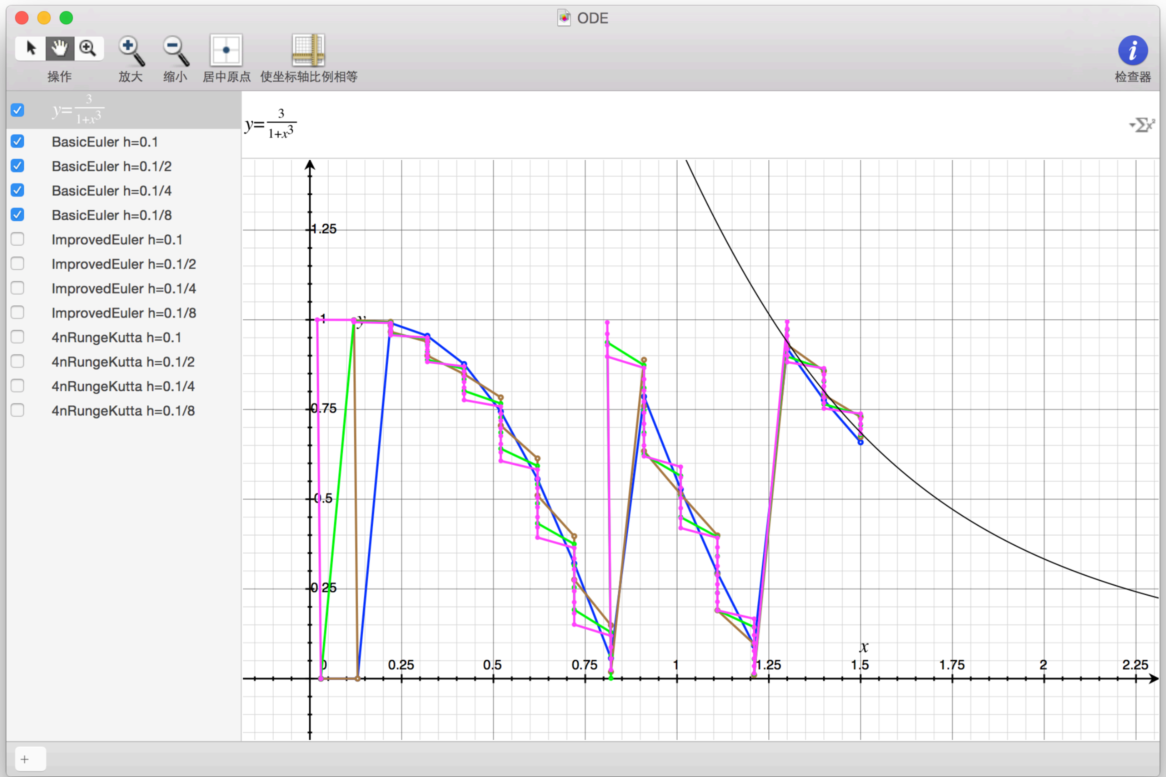
代码:

```
subroutine Basic_Euler()
    implicit none
    do i=1,n
        y(i)=y(i-1)+h*f(x(i-1),y(i-1))
    end do
end subroutine Basic_Euler
```

输出结果：

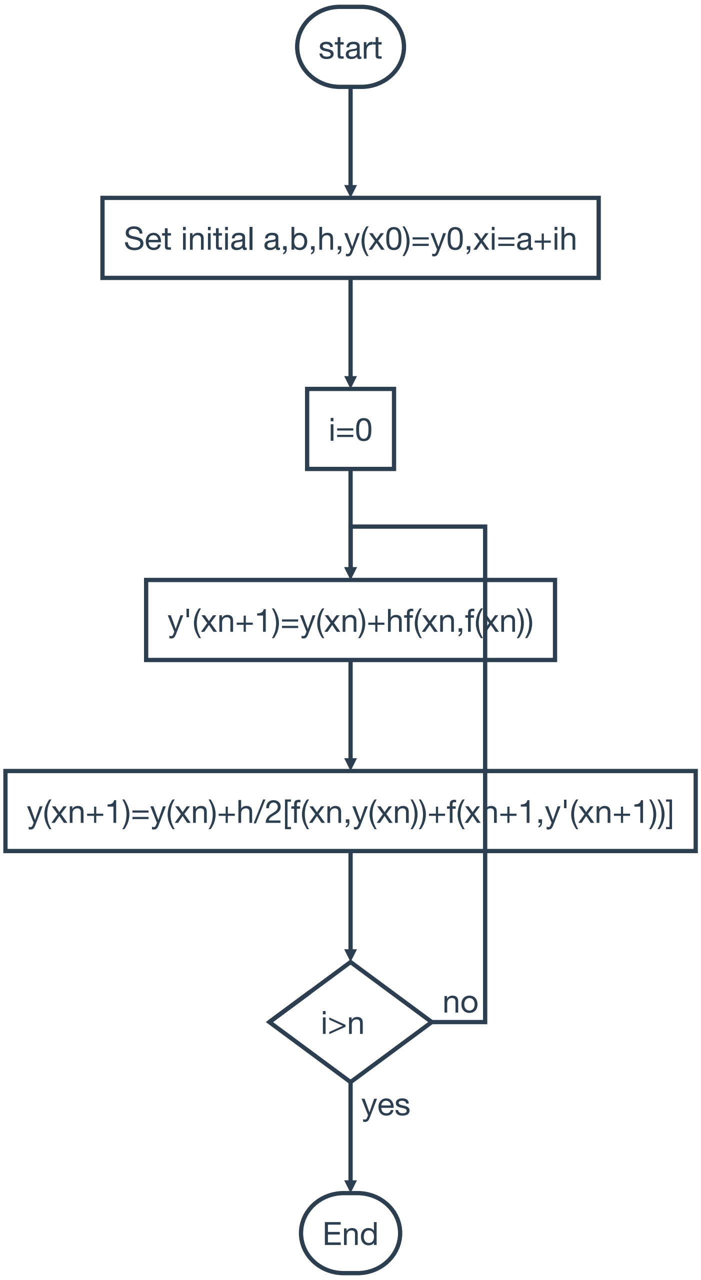
```
Eulars-MacBook-Pro:5 eular$ gfortran -o a 1.f90 && ./a
 The analytic solution of y(1.5)=   0.685714304

 Basic Euler Method
     Set initial  x0= 0.0   y0= 3.0   a= 0.0   b= 1.5
     Set h= 0.1000
        y(1.5)= 0.65864694
     Set h= 0.0500
        y(1.5)= 0.67318219
     Set h= 0.0250
        y(1.5)= 0.67967767
     Set h= 0.0125
        y(1.5)= 0.68275088

Eulars-MacBook-Pro:5 eular$ 
```

图例：



Improved Euler Method

流程图:



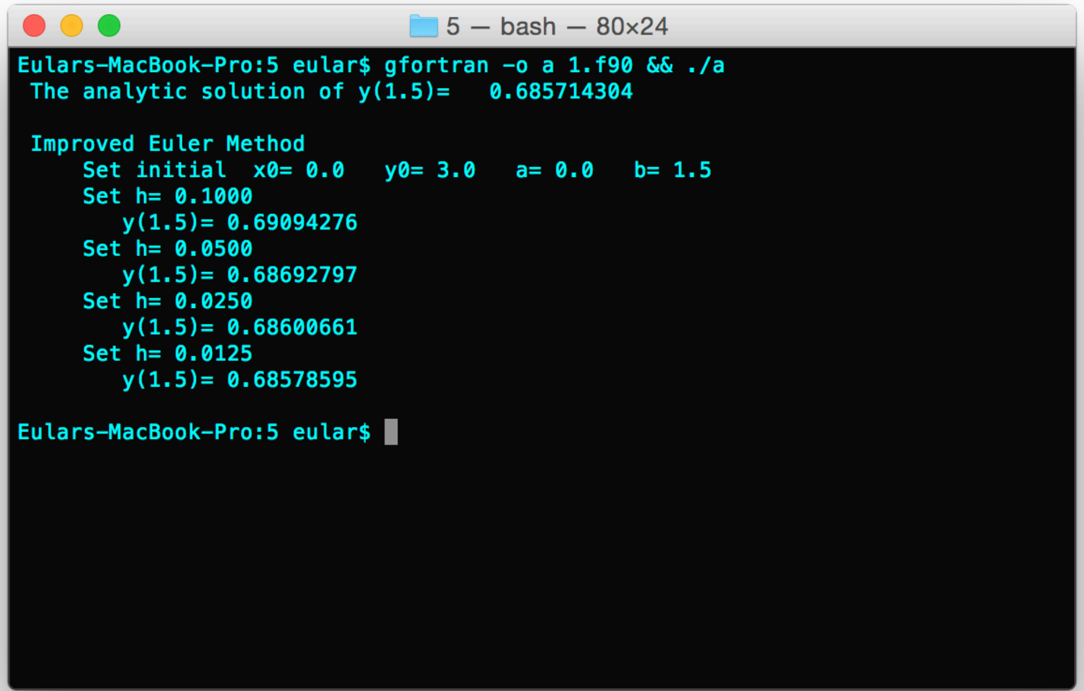
原理:

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n) \\ y(x_{n+1}) = y(x_n) + \dfrac{h}{2}[f(x_n, y(x_n)) + f(x_{n+1}, \bar{y}_{n+1})] + O(h^2) \\ y(x_0) = y_0 \end{cases}$$

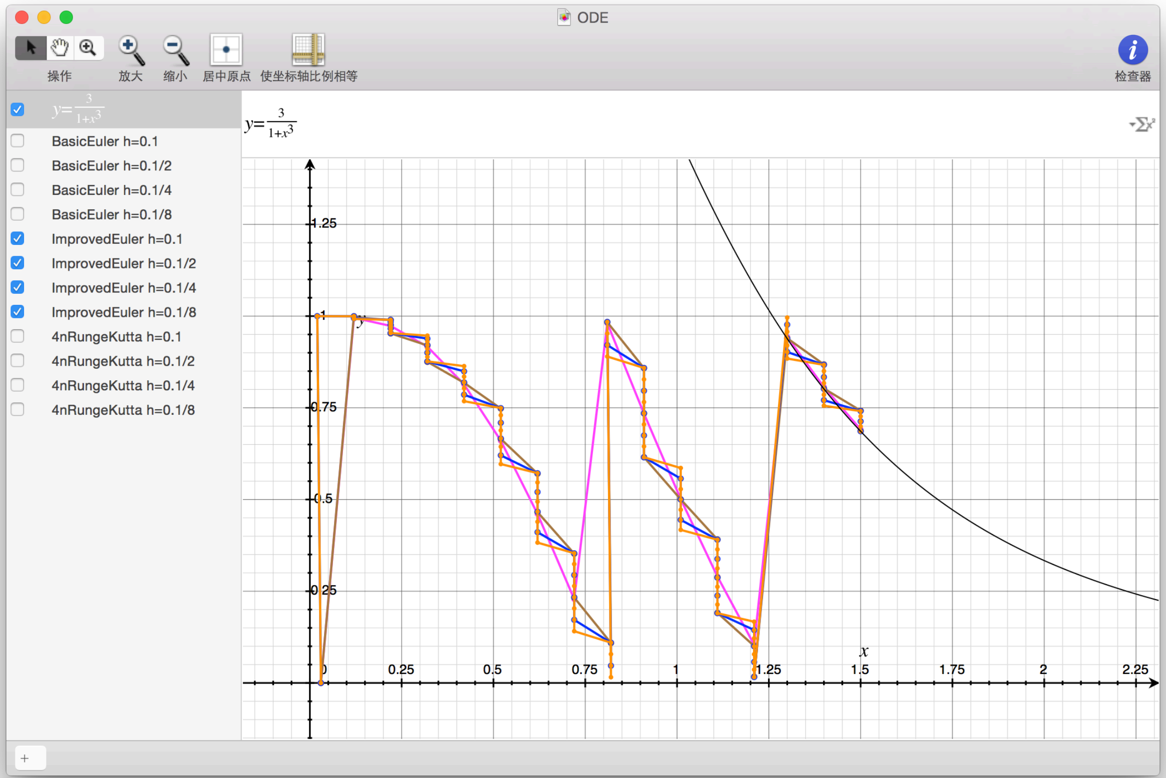
代码:

```
subroutine Improved_Euler()
    implicit none
    real :: y_
    do i=1,n
        y_=y(i-1)+h*f(x(i-1),y(i-1))
        y(i)=y(i-1)+h/2*(f(x(i-1),y(i-1))+f(x(i),y_))
    end do
end subroutine Improved_Euler
```

输出结果:

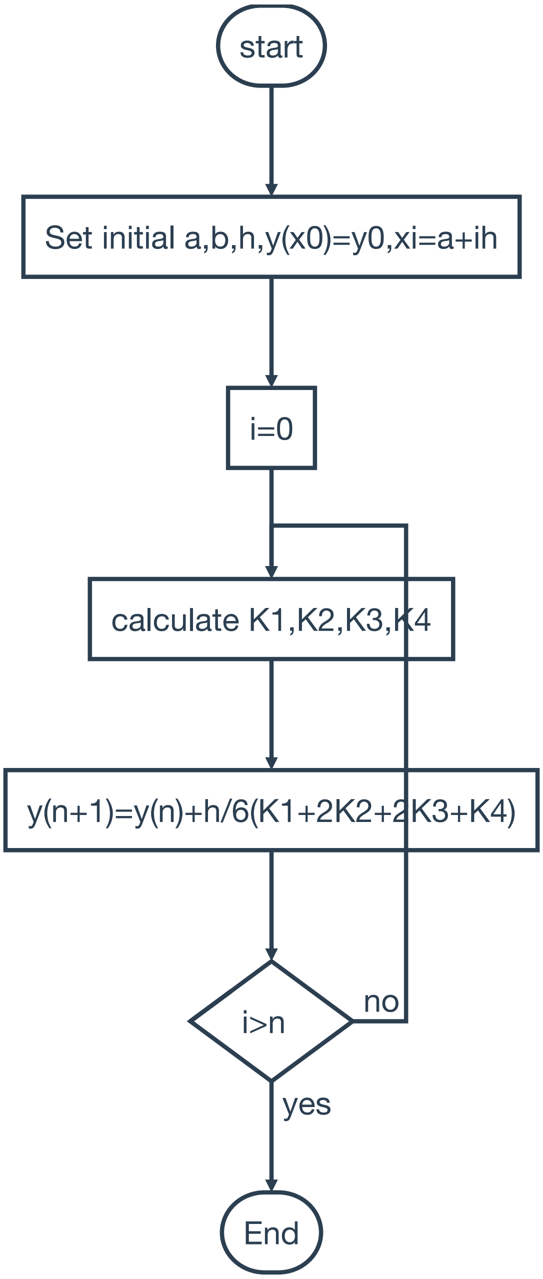


图例:



Four Order Runge-Kutta Method

流程图:



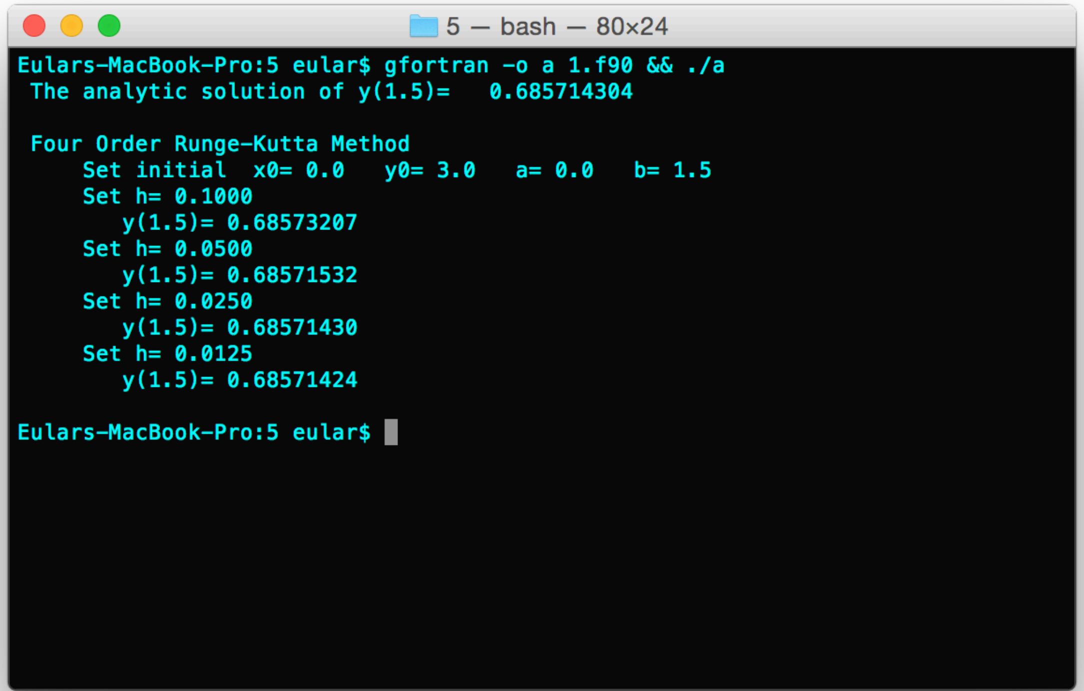
原理:

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

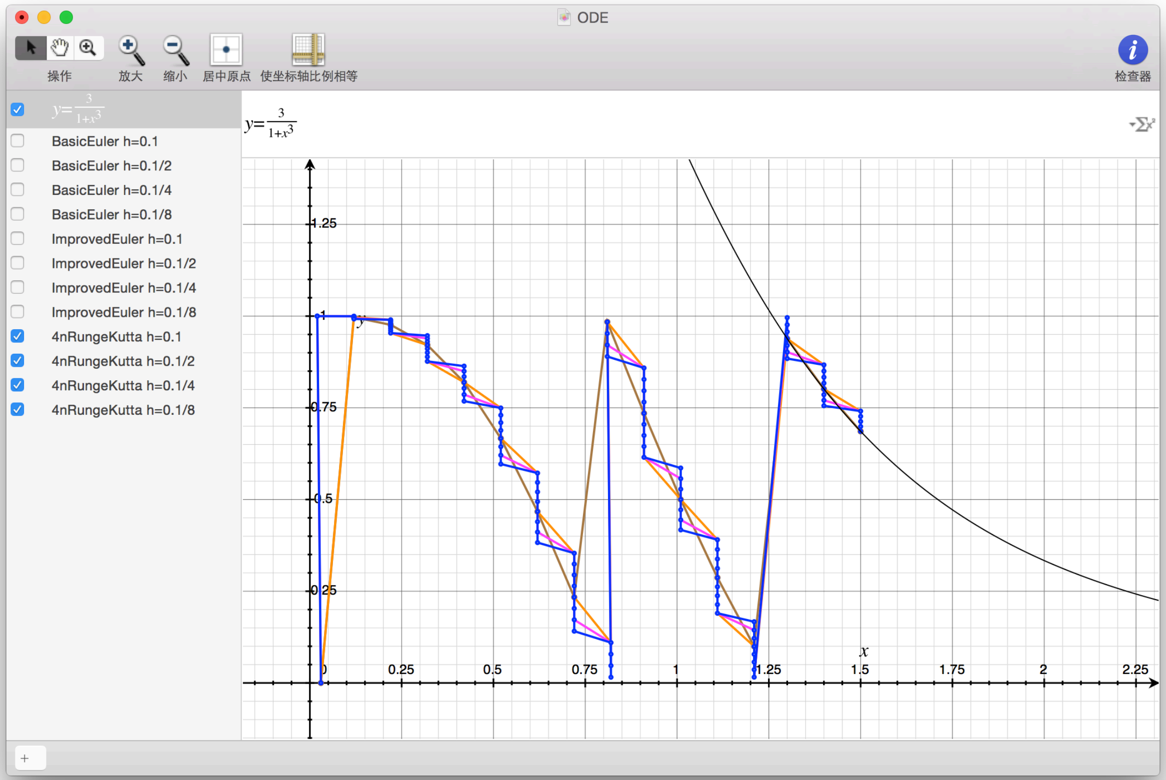
代码:

```
subroutine Four_Order_Runge_Kutta()
    implicit none
    real :: k1,k2,k3,k4
    do i=1,n
        k1=f(x(i-1),y(i-1))
        k2=f(x(i-1)+h/2,y(i-1)+h/2*k1)
        k3=f(x(i-1)+h/2,y(i-1)+h/2*k2)
        k4=f(x(i-1)+h,y(i-1)+h*k3)
        y(i)=y(i-1)+h/6*(k1+2*k2+2*k3+k4)
    end do
end subroutine Four_Order_Runge_Kutta
```

输出结果:

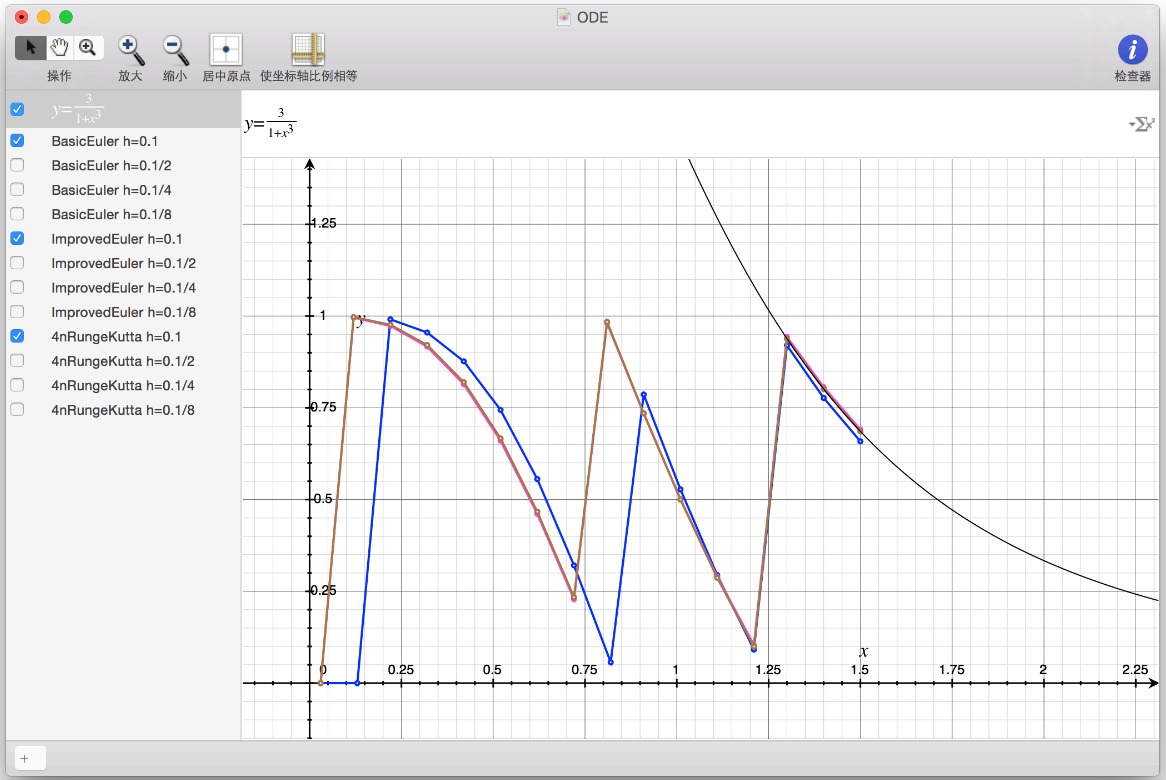


图例:

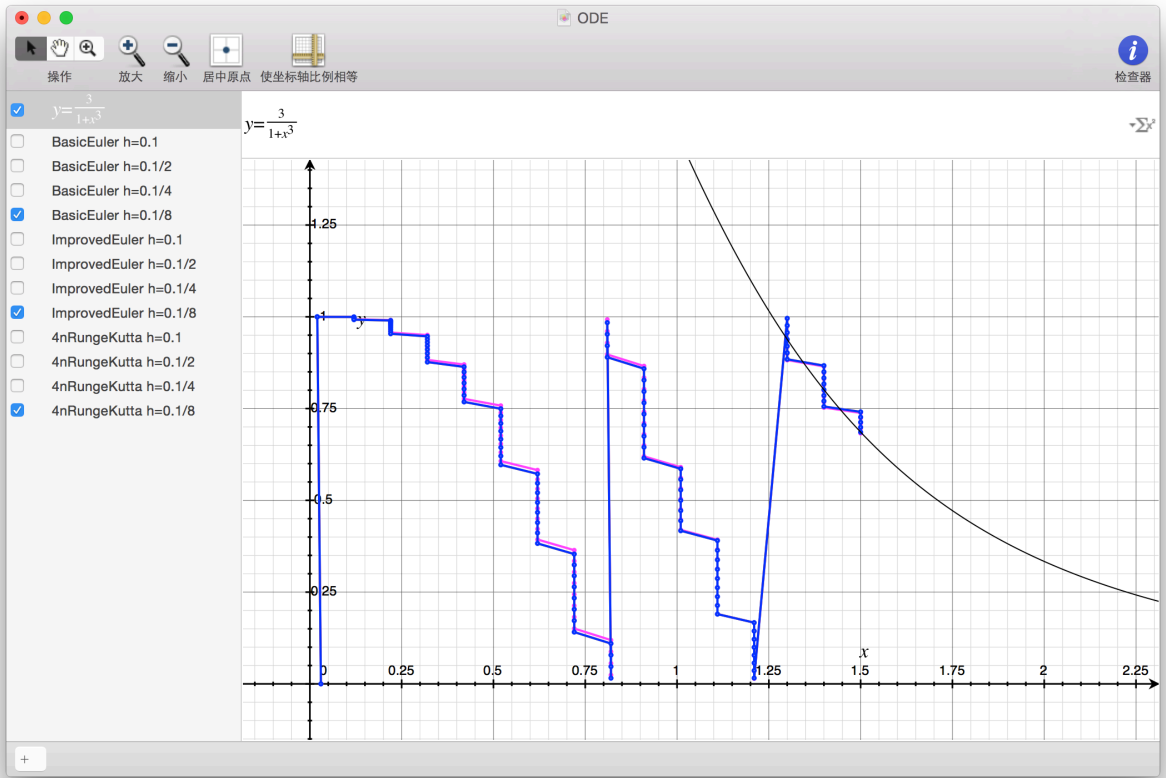

ODE
操作
放大
缩小
居中原点
使坐标轴比例相等
检查器
BasicEuler h=0.1
BasicEuler h=0.1/2
BasicEuler h=0.1/4
BasicEuler h=0.1/8
ImprovedEuler h=0.1
ImprovedEuler h=0.1/2
ImprovedEuler h=0.1/4
ImprovedEuler h=0.1/8
4nRungeKutta h=0.1
4nRungeKutta h=0.1/2
4nRungeKutta h=0.1/4
4nRungeKutta h=0.1/8
$y=\frac{3}{1+x^3}$
1.25
1
0.75
0.5
0.25
0
0.25
0.5
0.75
1
1.25
1.5
1.75
2
2.25
x
y

三种方法结果对比

`h=0.1`时


ODE
操作
放大
缩小
居中原点
使坐标轴比例相等
检查器
BasicEuler h=0.1
BasicEuler h=0.1/2
BasicEuler h=0.1/4
BasicEuler h=0.1/8
ImprovedEuler h=0.1
ImprovedEuler h=0.1/2
ImprovedEuler h=0.1/4
ImprovedEuler h=0.1/8
4nRungeKutta h=0.1
4nRungeKutta h=0.1/2
4nRungeKutta h=0.1/4
4nRungeKutta h=0.1/8
$y=\frac{3}{1+x^3}$

`h=0.1/8`时


ODE
操作
放大
缩小
居中原点
使坐标轴比例相等
检查器
BasicEuler h=0.1
BasicEuler h=0.1/2
BasicEuler h=0.1/4
BasicEuler h=0.1/8
ImprovedEuler h=0.1
ImprovedEuler h=0.1/2
ImprovedEuler h=0.1/4
ImprovedEuler h=0.1/8
4nRungeKutta h=0.1
4nRungeKutta h=0.1/2
4nRungeKutta h=0.1/4
4nRungeKutta h=0.1/8
$y=\frac{3}{1+x^3}$

> 可以看出，当h较大时，三种方法的差别还是很大的，当h逐渐减小时，三种方法的结果已基本相同。