实验报告

1 Euler 法和改进欧拉法

1. 实验题目

实习题 1：

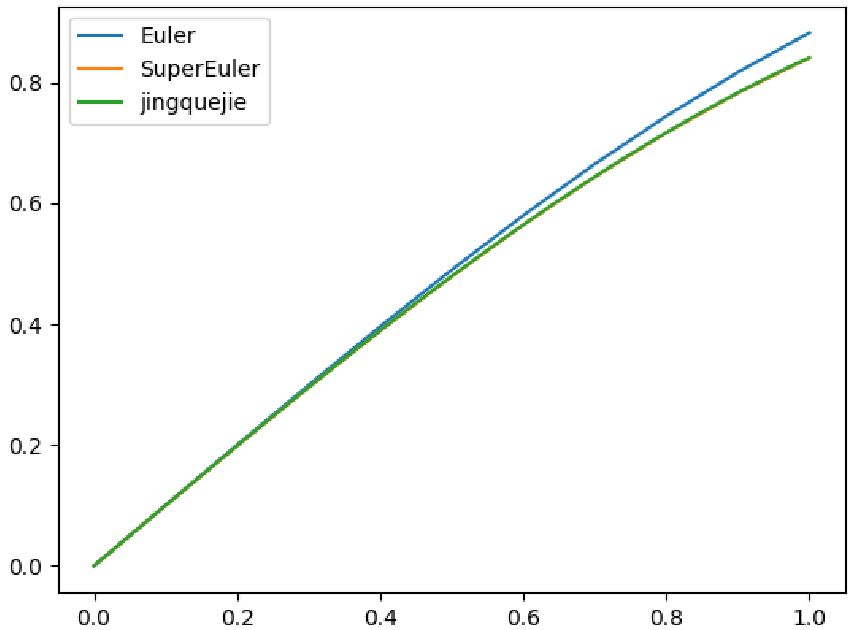
将 $u''=-u(0\le t\le 1)$, $u(0)=0$, $u'(0)=1$ 化成一阶方程组，并用 Euler 法和改进的 Euler 法求解，步长 h = 0.1, 0.05。

(1) 画出 Euler 法，Euler 改进法以及精确解的数值结果图形；

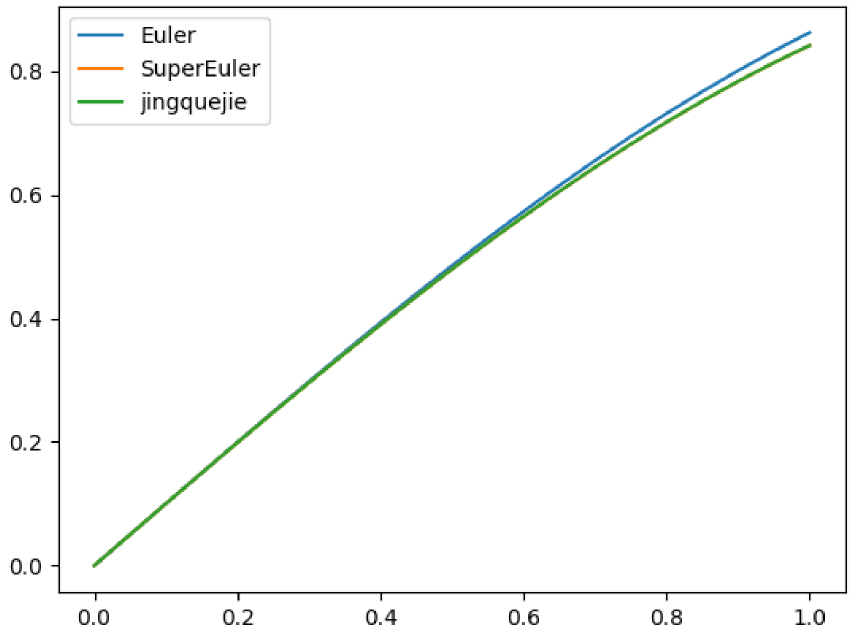
(2) 列表对比 Euler 法，Euler 改进法以及精确解，判断哪种方法的精度更高。

2. 实验结果

(1) 步长h=0.1



（1） 步长h=0.05



3. 代码实现

```python
# import module
import numpy as np
import matplotlib
from math import *
matplotlib.rcParams['axes.unicode_minus'] =False
import matplotlib.pyplot as plt


def jingquejie(a,b):
    '''
    input :
    a: 区间左端点
    b: 区间右端点
    output :

    '''
    x=np.linspace(a,b,100)
    y=np.sin(x)
    plt.plot(x,y)


#Euler法
def Euler(a,b,Mo,Uo,h):
    '''
    input:
    a,b :区间左右端点
```

```python
    Mo,Uo: 初值
    h: 步长
    '''
    #节点个数
    num=int((b-a)/h+1)
    U=np.zeros((num,))
    M=np.zeros((num,))
    U[0]=Uo
    M[0]=Mo
    for n in range (num-1):
        if n>=1:
           M[n]=M[n-1]-h*U[n-1]
        U[n+1]=U[n]+h*M[n]

    x=np.linspace(a,b,num)
    plt.plot(x,U)


#改进欧拉法
def SuperEuler(a,b,Mo,Uo,h):
    '''
    input :
    a,b:区间左右端点
    Mo,Uo: 初值
    h:步长
    '''
    num=int((b-a)/h+1)
    U=np.zeros((num,))
    M=np.zeros((num,))
    U[0]=Uo
    M[0]=Mo
    L=np.zeros((num,2))
    L[0,0]=M[0]
    L[0,1]=U[0]
    A=np.array([[2,h],[-h,2]])
    #L[n+1,:]=[M[n+1],U[n+1]]
    #A*L[n+1,:].T=C*L[n,:].T
    C=A.T
    for n in range (num-1):
        L[n+1,:]=np.linalg.solve(A, np.dot(C,L[n,:].T)).T
    U=L[:,1]
    x=np.linspace(a,b,num)
    plt.plot(x,U)


def main():
    Mo=1
    Uo=0
```

```
    h=0.05
    a=0
    b=1
    Euler(a,b,Mo,Uo,h)
    SuperEuler(a,b,Mo,Uo,h)
    jingquejie(a,b)
    plt.legend(["Euler","SuperEuler","jingquejie"])
    plt.show()

main()
```

2 龙格库塔法

1. 实验题目

实验题 2

用四级四阶 Runge Kutta 法计算初值问题：

$$u' = 4tu^{\frac{1}{2}}, \qquad 0 \le t \le 2,$$

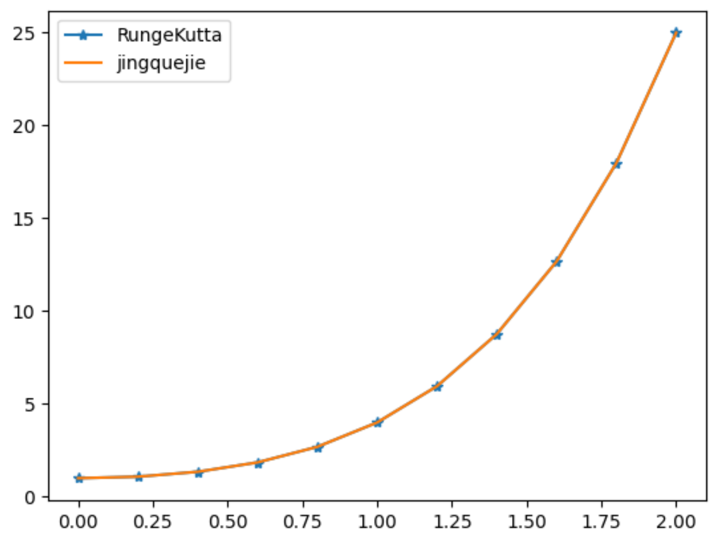
$$u(0)=1,$$

取 h=0.2，0.4，0.5.

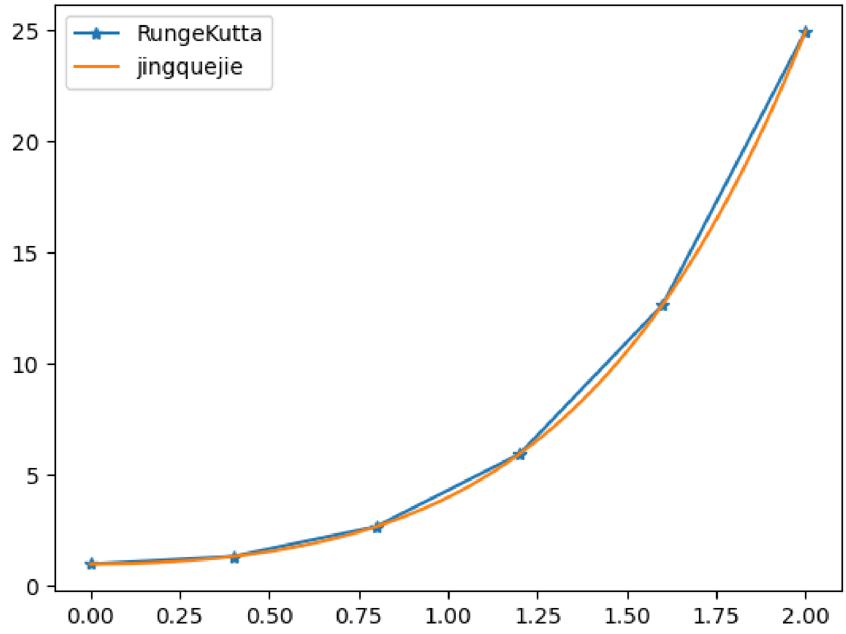
精确解为 $u(t) = (1+t^2)^2$

2. 实验结果

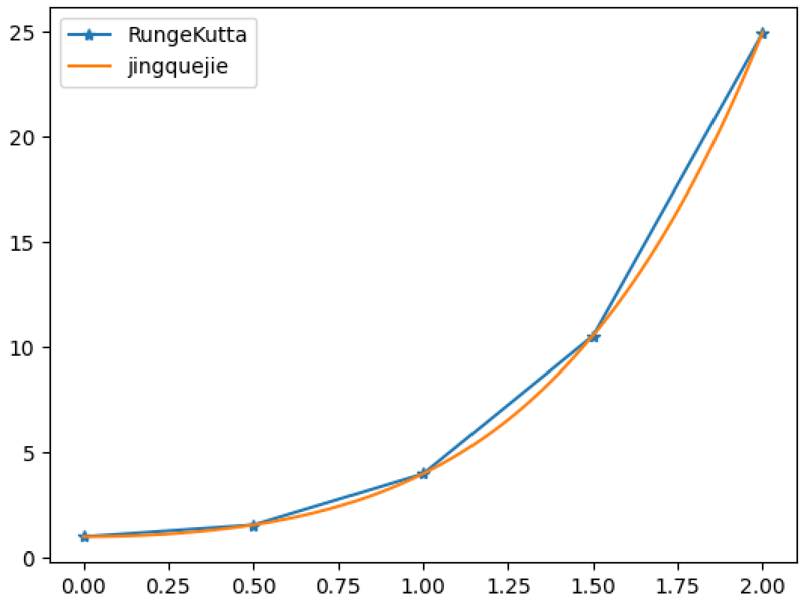
(1) $h = 0.2$



i. h=0.4



i. h= 0.5



3. 代码实现

```
# import module
import numpy as np
import matplotlib
from math import *
matplotlib.rcParams['axes.unicode_minus'] =False
import matplotlib.pyplot as plt
```

```python
def function(t,u):
    '''
    inpu
    t: 微分方程中的变量
    u: 微分方程的变量
    '''
    du=4*t*sqrt(u)
    return du
def CALk(t,Un,h):
    '''
    input :
    t:
    Un: U在t(n)处的取值
    h: 步长
    '''
    K=np.zeros((4,))
    K[0]=function(t,Un)
    K[1]=function(t+h/2,Un+h/2*K[0])
    K[2]=function(t+h/2,Un+h/2*K[1])
    K[3]=function(t+h,Un+h*K[2])
    return K

def RungeKutta(a,b,h,Uo):
    '''
    input
    a,b: 区间的做右端点
    h: 步长
    Uo: U的初值
    '''
    num=int((b-a)/h)+1
    U=np.zeros((num,))
    U[0]=Uo
    for n in range(num-1):
        t=a+n*h
        K=CALk(t,U[n],h)
        U[n+1]=U[n]+h/6*(K[0]+2*K[1]+2*K[2]+K[3])
    x=np.linspace(a,b,num)
    plt.plot(x,U,'-*')


def jingquejie(a,b):
    '''
    input
    a,b:区间做右端点
    '''
    x=np.linspace(a,b,100)
    y=(1+x*x)*(1+x*x)
    plt.plot(x,y)
```

```python
def main():
    a=0
    b=2
    h=0.5
    Uo=1
    RungeKutta(a,b,h,Uo)
    jingquejie(a,b)
    plt.legend(["RungeKutta","jingquejie"])
    plt.show()

main()
```