实　验　报　告

实验地点		学生姓名	
实验日期	2021 年 11 月 30 日　第 7、8 节	学院	数学与统计学院
实验课程	数值逼近	学号	
实验项目	常微分方程数值解	成绩	

一、实验目的或要求

编写常用的常微分方程数值解方法

二、实验过程记录

（零）待求解方程

方程

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x} = f(x,y) = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases}$$

```
def f(x,y):
    return y-2*x/y
```

（一）前 Euler 法

1、程序代码

欧拉法

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

```
def euler(fun,y0,a,b,h):
    n = int(np.ceil((b-a)/h))
    y = np.zeros(n+1)
    x = np.arange(a,b+0.1,0.1)
    y[0] = y0
    for i in range(n):
        y[i+1] = y[i]+h*fun(x[i],y[i])
    out = {'X':x,'Y':y}
    return out
```

2、代码测试

```
# test
a = euler(f,1,0,1,0.1)
a
```

```
{'X': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 'Y': array([1.        , 1.1       , 1.19181818, 1.27743783, 1.3582126 ,
        1.43513292, 1.50896625, 1.58033824, 1.64978343, 1.71777935,
        1.78477083])}
```

（二）隐式 Euler 法

1、程序代码

隐式欧拉格式:

欧拉格式是用差商的值来近似点(x_n, y_n)处的导数值，若用差商的值来近似点(x_{n+1}, y_{n+1})处的导数值，则可以得到隐式欧拉格式

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

隐式欧拉格式只有在$f(x, y)$容易将两个参数分离的情况，才比较容易使用.

```
def hide_euler(fun,y0,a,b,h):
    n = int(np.ceil((b-a)/h))+1
    eu = euler(fun,y0,a,b,h)
    y = eu['Y']
    x = eu['X']
    y_old = np.zeros(n)
    times = 0
    while(np.linalg.norm(y_old-y)>0.1 and times<500):
        y_old = y.copy()
        for i in range(n-1):
            y[i+1] = y[i]+h*(fun(x[i+1],y[i+1]))
        #print(np.linalg.norm(y_old-y))
        times+=1
    out = {'X':x,'Y':y}
    return out
```

2、代码测试

```
# test
ha = hide_euler(f,1,0,1,0.1)
ha
```

```
{'X': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 'Y': array([1.        , 1.09181818, 1.17743783, 1.2582126 , 1.33513292,
        1.40896625, 1.48033824, 1.54978343, 1.61777935, 1.68477083,
        1.75118871])}
```

（三）梯形方法

1、实验代码

梯形方法:
为了得到更加准确的估计，可以考虑将欧拉格式和隐式欧拉格式取平均值，得到

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

上述式子称为梯形公式，为了计算y_{n+1}常用以下迭代式

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

当$|y_{n+1}^{(k+1)} - y_{n+1}^{(k)}| < \varepsilon$时，取$y_{n+1} \approx y_{n+1}^{(k+1)}$.

```
def trapezoid(fun,y0,a,b,h):
    n = int(np.ceil((b-a)/h))+1
    eu = euler(fun,y0,a,b,h)
    y = eu['Y']
    x = eu['X']
    y_old = np.zeros(n)
    times = 0
    while(np.linalg.norm(y_old-y)>1 and times<100):
        y_old = y.copy()
        for i in range(n-1):
            yy = y.copy()
            y[i+1] = y[i]+(h/2)*(fun(x[i+1],y[i+1])+fun(x[i],yy[i]))
        #print(np.linalg.norm(y_old-y))
        times+=1
    out = {'X':x,'Y':y}
    return out
```

2、代码测试

```
# test
at = trapezoid(f,1,0,1,0.1)
at
```

```
{'X': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 'Y': array([1.        , 1.09590909, 1.18438953, 1.26711005, 1.34524979,
        1.41969469, 1.49114658, 1.56018901, 1.62733006, 1.69303203,
        1.7577335 ])}
```

（四）改进 Euler 法

$$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_n, y_n + hf(x_n, y_n))), \quad n = 0,1,\ldots$$

1、实验代码

```
def euler_pro(fun,y0,a,b,h):
    n = int(np.ceil((b-a)/h))+1
    y = np.zeros(n)
    x = np.arange(a,b+0.1,0.1)
    y[0] = y0
    for i in range(n-1):
        y[i+1] = y[i]+(h/2)*(fun(x[i],y[i])+f(x[i+1],y[i]+h*f(x[i],y[i])))
    out = {'X':x,'Y':y}
    return out
```

2、代码测试

```
# test
apro = euler_pro(f,1,0,1,0.1)
apro
```

```
{'X': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 'Y': array([1.        , 1.09590909, 1.18409657, 1.26620136, 1.34336015,
        1.41640193, 1.4859556 , 1.55251409, 1.61647478, 1.67816636,
        1.7378674 ])}
```

（五）二级龙格库塔法

$$
\begin{aligned}
y_{n+1} &= y_n + h\alpha_1 k_1 + h\alpha_2 k_2 \\
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + h\lambda_2, y_n + h\mu_2 k_1)
\end{aligned}
$$

1、实验代码

```
def runge_kutta2(fun,y0,a,b,h,alpha1,alpha2,lambda2,mu2):
    if alpha1+alpha2!=1 or alpha2*lambda2!=1/2 or alpha2*mu2!=1/2:
        print('不满足条件')
        return 0
    n = int(np.ceil((b-a)/h))+1
    y = np.zeros(n)
    x = np.arange(a,b+0.1,0.1)
    y[0] = y0
    for i in range(n-1):
        k1=fun(x[i],y[i])
        k2=fun(x[i]+h*lambda2,y[i]+h*mu2*k1)
        y[i+1] = y[i]+h*alpha1*k1+h*alpha2*k2
    out = {'X':x,'Y':y}
    return out
```

2、代码测试

```
# test
# 1 alpha1=0,alpha2=1,lambda2=mu2=1/2
ark1 = runge_kutta2(f,1,0,1,0.1,0,1,1/2,1/2)
ark1
# 1 alpha1=1/2,alpha2=1/2,lambda2=mu2=1
ark2 = runge_kutta2(f,1,0,1,0.1,1/2,1/2,1,1)
ark2
```

```
{'X': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 'Y': array([1.        , 1.09590909, 1.18409657, 1.26620136, 1.34336015,
        1.41640193, 1.4859556 , 1.55251409, 1.61647478, 1.67816636,
        1.7378674 ])}
```

（六）4 级龙格库塔法

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$k_1 = f(x_n, y_n)$$
$$k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$
$$k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$
$$k_4 = f(x_n + h, y_n + hk_3)$$

1、实验代码

```
def runge_kutta4(fun,y0,a,b,h):
    n = int(np.ceil((b-a)/h))+1
    y = np.zeros(n)
    x = np.arange(a,b+0.1,0.1)
    y[0] = y0
    for i in range(n-1):
        k1=fun(x[i],y[i])
        k2=fun(x[i]+h/2,y[i]+h*k1/2)
        k3=fun(x[i]+h/2,y[i]+h*k2/2)
        k4=fun(x[i]+h,y[i]+h*k3)
        y[i+1] = y[i]+h*(k1+2*k2+2*k3+k4)/6
    out = {'X':x,'Y':y}
    return out
```

2、代码测试

```
# test
ark4 = runge_kutta4(f,1,0,1,0.1)
ark4
```

```
{'X': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 'Y': array([1.        , 1.09544553, 1.18321675, 1.26491223, 1.34164235,
        1.41421558, 1.48324222, 1.54919645, 1.61245535, 1.67332466,
        1.73205637])}
```

（七）结果展式

1、实验代码

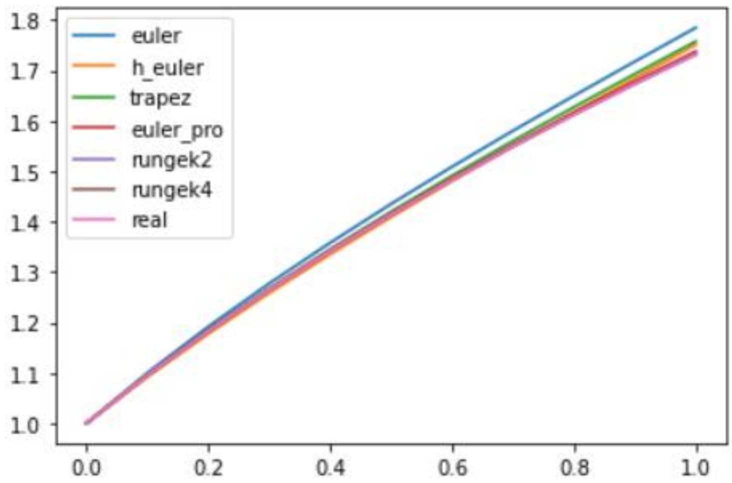
Display

```
import matplotlib.pyplot as plt
from scipy.integrate import odeint
a = euler(f,1,0,1,0.1)
ha = hide_euler(f,1,0,1,0.1)
at = trapezoid(f,1,0,1,0.1)
apro = euler_pro(f,1,0,1,0.1)
ark2 = runge_kutta2(f,1,0,1,0.1,0,1,1/2,1/2)
ark4 = runge_kutta4(f,1,0,1,0.1)
# real
def real(x):
    return np.sqrt(2*x + 1)

plt.plot(a['X'],a['Y'],label='euler')
plt.plot(ha['X'],ha['Y'],label='h_euler')
plt.plot(at['X'],at['Y'],label='trapez')
plt.plot(apro['X'],apro['Y'],label='euler_pro')
plt.plot(ark2['X'],ark2['Y'],label='rungek2')
plt.plot(ark4['X'],ark4['Y'],label='rungek4')
plt.plot(np.arange(0,1+0.1,0.1),real(np.arange(0,1+0.1,0.1)),label='real')
plt.legend(loc='best')

plt.show()
```

2、结果



其中，粉红色线为真实值。通过观察可以得到，对于此方程，本文所用的 6 种数值方法的效果都不错，其中 2 级龙格库塔法效果最好。

三、实验结果报告及总结

隐式欧拉法与梯形方法由于需要第 n+1 步的信息，因此需要首先用欧拉法进行第一步的初始化，之后再进行多步迭代，时间复杂度较高。

实验结果反思及讨论：
教师对报告的最终评价和意见: 年　　月　　日