**欧拉法的ODE初值问题求解**

**082210115 徐安璞 2025/3/4**

一、问题重述

对于设定的步长，用欧拉法求解初值问题：



的数值解，并与精确解比较。

二、欧拉方法介绍

欧拉法（Euler’s Method）是一种数值方法，用于求解初值问题形式的常微分方程（ODE），其核心思想是利用微分方程的具体斜率，通过迭代的方法逐步逼近真实的解。

欧拉法的思想其实是基于泰勒展开的一个近似，在一个小的设定步长内，使用已知点的斜率来估算下一个点的值。通过的公式即为下面这个式子：  


其中：是当前点的近似点，为当前点的导数，是步长，控制着数值积分的精度和稳定性。

通过不断重复上述的步骤，可以从迭代计算到目标区间的终点.

三、欧拉方法的几何解释

* 在每一个点上，我们使用微分方程提供的斜率作为线性逼近的方向。
* 通过沿着斜率的方向前进一个小步长，得到下一个点的近似值。
* 由于我们只是使用了一个当前点的信息，逼近的曲线实际上是一系列小线段的连接用来近似真实的解

四、课本作业的编程实现

4.1源代码

main.py

import numpy as np  
import matplotlib  
import pandas as pd  
matplotlib.use('TkAgg') # 设置后端为 TkAgg  
import matplotlib.pyplot as plt  
  
  
"""本程序使用了一些简单的步骤实现了完成课本程序的复现，本程序使用了类似于全局变量的操作，使得用户可以更改函数、步长、初值进行其他ODE方程的求解。"""  
  
# 解析解定义  
def exact\_solution(x):  
 return (1 / 2) \* (x \* x + 2) \* (np.exp(-x))  
  
#欧拉法  
def euler\_method(f, x0, y0, h, x\_end):  
 *"""使用欧拉法求解ODE：y'=x\*np.exp(-x)-y"""* x\_values = np.arange(x0, x\_end + h, h) # 生成x值  
 y\_values = np.zeros(len(x\_values)) # 预分配y值  
 y\_values[0] = y0 # 初值  
 errors = np.zeros(len(x\_values))#误差  
 # 循环输出  
 for i in range(1, len(x\_values)):  
 y\_values[i] = y\_values[i - 1] + h \* f(x\_values[i - 1], y\_values[i - 1])  
 y\_exact = exact\_solution(x\_values[i])  
 errors[i] = np.abs(y\_values[i] - y\_exact)  
  
  
 return x\_values, y\_values,errors  
  
# 定义ODE：dy/dx=x\*np.e(-x)-y  
def f(x, y):  
 return x \* np.exp(-x) - y  
  
#设置初值  
x0,y0=0,1  
x\_end=1  
step\_size=[0.1,0.05,0.01]  
  
#画图  
plt.figure(figsize=(8,6))  
  
#绘制解析解  
x\_exact=np.linspace(x0,x\_end,100)  
y\_exact=exact\_solution(x\_exact)  
plt.plot(x\_exact,y\_exact,label="Exact Solution",linestyle="dashed",color='black')  
  
# 使用欧拉法求数值解  
colors=['b','g','r']  
  
#创建excel  
with pd.ExcelWriter("euler\_results.xlsx", engine="openpyxl") as writer:  
 for h,color in zip(step\_size,colors):  
 #计算数值解  
 x\_vals,y\_vals,errors=euler\_method(f,x0,y0,h,x\_end)  
  
 #画图  
 plt.plot(x\_vals,y\_vals,marker='o',linestyle='-',color=color,label=f"Euler h={h}")  
  
 #存入DataFrame  
 df=pd.DataFrame({  
 "x":x\_vals,  
 "y\_exact":exact\_solution(x\_vals),  
 "y\_numerical":y\_vals,  
 "error":errors  
 })  
  
 #写入Excel，每种步长的数据存入不同的sheet  
 df.to\_excel(writer,sheet\_name=f"h={h}",index=False)  
  
 #打印结果  
 print(f"\n步长 h={h} 的计算结果：")  
 print(df)  
  
#设置图例  
plt.xlabel("x")  
plt.ylabel("y")  
plt.legend()  
plt.title("Euler Method Approximation with Different Step Sizes")  
plt.grid()  
plt.show()  
  
print("\n数据已成功保存到 euler\_results.xlsx，每个步长数据存入不同 Sheet")

4.2代码分析

通过python强大的绘图和excel表生成的库，我将所有的结果放在了一个excel表中，并且通过步长的不同放在了不同的sheet中，完整表现了不同步长带来的精确度的不同。

本代码也可以同步应用在其他常微分方程的求解中，十分方便的通过修改内容来改变结果的输出。具有强大的用户友好型。

4.3代码结果

4.3.1图像

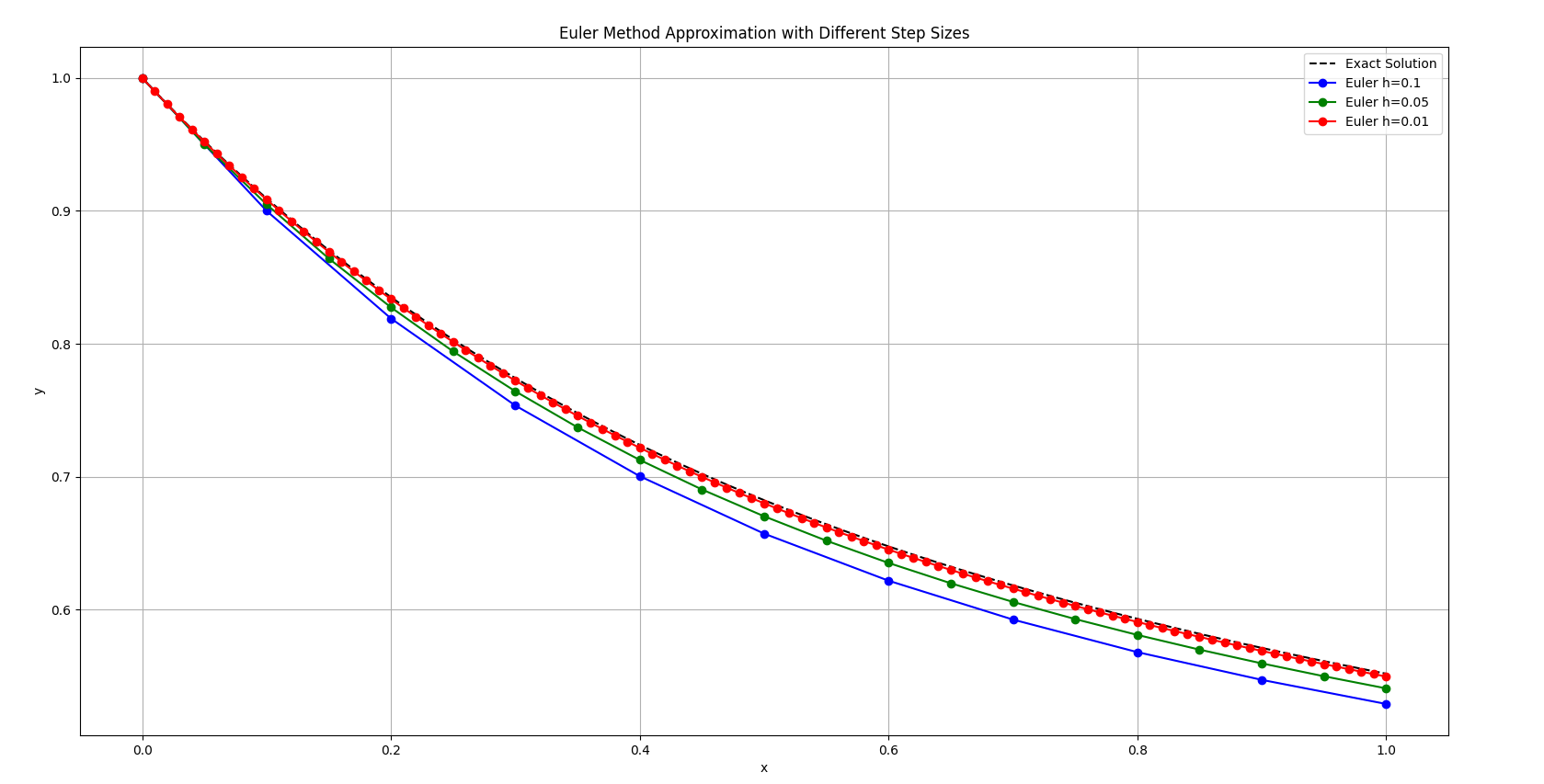


图1 欧拉法不同步长与精确解的逼近程度

4.3.2数据表格

图片包含 文本

AI 生成的内容可能不正确。

图2 步长为0.1时的数据

文本

AI 生成的内容可能不正确。

图三 步长为0.05时的数据

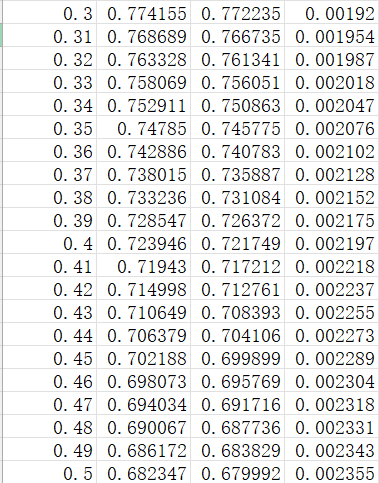


图4 步长为0.01时的数据（截取0.3-0.5部分）

可见当步长越小的时候，计算的精度存在一定程度的提升。

五、对欧拉法的认识

5.1优点

1、简单易实现，只需要一个简单的迭代公式，便可以编程实现一个较为精确的结果。

2、计算速度快，同样的，一个计算公式便可以估算大致的数值解。

5.2缺点

1、精度较低，由于只使用当前点的信息，没有考虑未来点的变化，导致误差较大。

2、步长敏感：过大的步长会导致严重的误差，而步长过小，会严重增加计算量。

3、稳定性问题：当微分方程的解存在过大的变化时，可能结果会不稳定》

5.3欧拉法的改进

1、中点格式

2、梯形法：使用当前点和下一个点的平均斜率。

3、龙格库塔方法