数值分析期末上机

By 211870125 陈睿硕

§1 问题

Q1 应用经典的四阶 Runge-Kutta 方法数值求解如下问题

$$\begin{cases} y'' + \frac{2}{x}y' - \frac{6}{x^2}y = 5 - 6x + 7x^2, 1 < x < 2 \\ y(1) = \frac{1}{2}, y(2) = 4 + 4\ln 2 \end{cases}$$

该问题的真解为 $y = x^2 - x^3 + \frac{1}{2}x^4 + x^2 \ln x$。
分别取步长 $h = \frac{1}{200}, \frac{1}{100}, \frac{1}{50}, \frac{1}{25}, \frac{1}{10}, \frac{1}{5}, \frac{1}{4}$ 进行计算，根据数值结果给出误差收敛阶，对计算结果进行分析。

§2 算法思路

I 算法：经典的四阶 Runge-Kutta 方法

即 4 级四阶 Runge-Kutta 方法：

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_1) \\ K_3 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_2) \\ K_4 = f(x_n + h, y_n + hK_3) \end{cases}$$

II 对Q1的解答

事实上，对于这个问题，我们也可以使用高阶微分方程的四阶 Runge-Kutta 方法，但是我们这里发现，我们可以将题目式子化简为

$$\begin{cases} (y' + \frac{3}{x}y)' = \frac{1}{x}(y' + \frac{3}{x}y) + 5 - 6x + 7x^2, 1 < x < 2 \\ y(1) = \frac{1}{2}, y(2) = 4 + 4\ln 2 \end{cases}$$

进而使用常微分方程方法，可以进一步化简为

$$\begin{cases} y' = -\frac{3}{x}y + 6x - 6x^2 + \frac{7}{2}x^3 + 5x\ln x, 1 < x < 2 \\ y(1) = \frac{1}{2}, y(2) = 4 + 4\ln 2 \end{cases}$$

于是我们可以直接令四阶经典 Runge-Kutta 方法中的 $f(x, y) = -\frac{3}{x}y + 6x - 6x^2 + \frac{7}{2}x^3 + 5x\ln x$。使用 python 实现算法，见代码 1。计算出在 x=2 处经典的四阶 Runge-Kutta 方法的误差（绝对值）和 h 的关系如图 1。

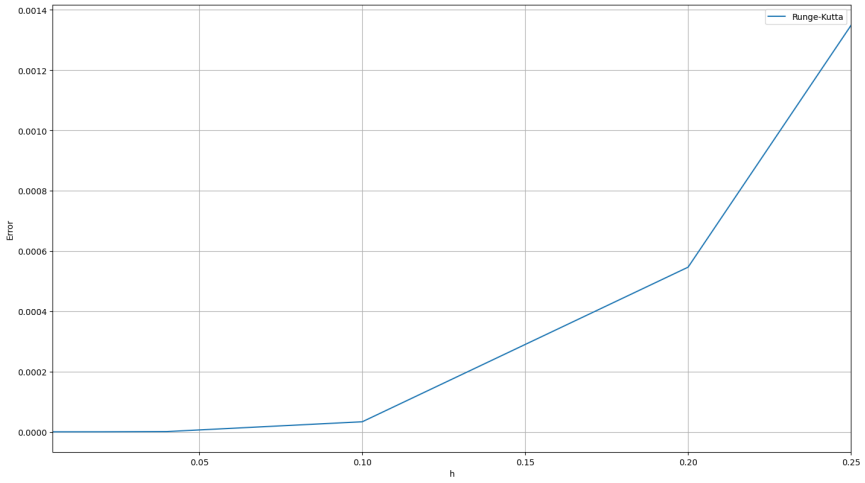

图 1: x=2 处误差（绝对值）和 h 的关系

得到 x=2 处的误差与 h 值的对应如表 1。

h	Error(x=2)
0.005	2.02958539e-10
0.01	3.25197380e-09
0.02	5.21786614e-08
0.04	8.39503355e-07
0.1	3.33185076e-05
0.2	5.46120318e-04
0.25	1.34812243e-03

表 1: x=2 处的误差

为了分析其误差收敛阶，我们对 h 与 x=2 处的误差两者的对数做线性拟合，拟合结果为 $\ln error = 4.0152541352629365 \ln h + -1.0543858312344068$, 如图 2，使用 python 的实现见代码 2。

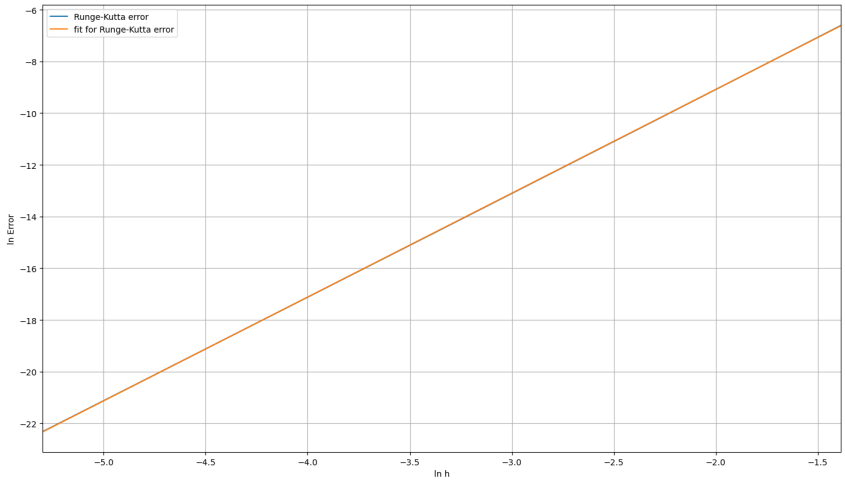

图 2: 误差（绝对值）对数和 h 对数的拟合

§3 结果分析

I 对Q1的分析

可以看到，图 2中拟合程度较高，所以我们基本可以判断这个方法的误差收敛阶应该为 4。事实上，这个结果也与我们理论学习中得到的结论相符合。

§4 附录：程序代码

代码 1: Final exam1.py

```
import numpy as np
from matplotlib import pyplot as plt

def y(x):
    return x**2-x**3+1/2*x**4+x**2*np.log(x)

def f(x, y):
    return (-3/x)*y+6*x-6*x**2+7/2*x**3+5*x*np.log(x)


def runge_kutta(x, y0, f):
    y = np.zeros(x.shape)
    y[0] = y0
    for i in range(len(x) - 1):
        k1 = f(x[i], y[i])
        k2 = f(x[i] + h/2, y[i] + h/2 * k1)
```

```
        k3 = f(x[i] + h/2, y[i] + h/2 * k2)
        k4 = f(x[i] + h, y[i] + h * k3)
        y[i+1] = y[i] + h * (k1 + 2*k2 + 2*k3 + k4) / 6
    return y

h_list = [1/200,1/100,1/50,1/25,1/10,1/5,1/4]
error_x_eq_2=np.zeros(len(h_list))
y0 = 1/2
for i in range(len(h_list)):
    h=h_list[i]
    n=int(1/h)
    x = np.linspace(1, 2, n+1)
    y_cal=runge_kutta(x, y0, f)
    error_x_eq_2[i]=np.abs(y_cal[-1]-y(2))
print(error_x_eq_2)

plt.figure(figsize=(12, 8))
plt.plot(h_list, error_x_eq_2, label='Runge-Kutta')
plt.legend(loc='best')
plt.xlabel('h')
plt.ylabel('Error')
plt.xlim((1/200,1/4))
plt.grid(True)
plt.show()
```

代码 2: Final exam2.py

```
import numpy as np
from matplotlib import pyplot as plt

def y(x):
    return x**2-x**3+1/2*x**4+x**2*np.log(x)

def f(x, y):
    return (-3/x)*y+6*x-6*x**2+7/2*x**3+5*x*np.log(x)


def runge_kutta(x, y0, f):
    y = np.zeros(x.shape)
    y[0] = y0
    for i in range(len(x) - 1):
        k1 = f(x[i], y[i])
```

```
        k2 = f(x[i] + h/2, y[i] + h/2 * k1)
        k3 = f(x[i] + h/2, y[i] + h/2 * k2)
        k4 = f(x[i] + h, y[i] + h * k3)
        y[i+1] = y[i] + h * (k1 + 2*k2 + 2*k3 + k4) / 6
    return y

h_list = [1/200,1/100,1/50,1/25,1/10,1/5,1/4]
error_x_eq_2=np.zeros(len(h_list))
y0 = 1/2
for i in range(len(h_list)):
    h=h_list[i]
    n=int(1/h)
    x = np.linspace(1, 2, n+1)
    y_cal=runge_kutta(x, y0, f)
    error_x_eq_2[i]=np.abs(y_cal[-1]-y(2))
print(error_x_eq_2)

h_list_log=np.log(h_list)
error_x_eq_2_log=np.log(error_x_eq_2)

coefficients = np.polyfit(
    h_list_log, error_x_eq_2_log, 1)
print(
    f"对于拟合的函数式为 ln(error) = {coefficients[0]}ln(h) + {
       coefficients[1]}")
error_log_fit=coefficients[0]*h_list_log+coefficients[1]

plt.figure(figsize=(12, 8))
plt.plot(h_list_log, error_x_eq_2_log, label='Runge-Kutta error')
plt.plot(h_list_log, error_log_fit, label='fit for Runge-Kutta error')
plt.legend(loc='best')
plt.xlabel('ln h')
plt.ylabel('ln Error')
plt.xlim((np.log(1/200),np.log(1/4)))
plt.grid(True)
plt.show()
```