數值方法 作業 5

工科系 115 E14116401 張瑋哲

第一題：

```
PS D:\ForClass\1132\1132Numerical\HW5> & C:/ProgramData/anaconda3/python.exe d:/ForClass/1132/1132Numerical/HW5/1All_Cal.py
      t     Euler    Taylor     Exact  Error (Euler)  Error (Taylor)
0   1.0  0.000000  0.000000  0.000000       0.000000        0.000000
1   1.1  0.100000  0.105000  0.105160       0.005160        0.000160
2   1.2  0.209917  0.220919  0.221243       0.011325        0.000324
3   1.3  0.330471  0.348612  0.349121       0.018651        0.000509
4   1.4  0.462354  0.488954  0.489682       0.027328        0.000728
5   1.5  0.606285  0.642883  0.643875       0.037590        0.000993
6   1.6  0.763041  0.811438  0.812753       0.049711        0.001315
7   1.7  0.933475  0.995787  0.997494       0.064019        0.001707
8   1.8  1.118537  1.197252  1.199439       0.080902        0.002187
9   1.9  1.319293  1.417344  1.420116       0.100823        0.002772
10  2.0  1.536943  1.657795  1.661282       0.124338        0.003487
PS D:\ForClass\1132\1132Numerical\HW5>
```

圖 1、第一題計算結果

```python
import numpy as np
import pandas as pd

# 定義微分方程 f(t, y) = 1 + y/t + (y/t)^2
def f(t, y):
    return 1 + y/t + (y/t)**2

# f 的導數 f'(t, y) = f_t + f_y * f(t, y)
def f_prime(t, y):
    f_t = -y/t**2 - 2*y**2/t**3
    f_y = 1/t + 2*y/t**2
    return f_t + f_y * f(t, y)

# exact sol y(t) = t*tan(ln(t))
def exact_solution(t):
    return t * np.tan(np.log(t))

# (A) euler:
#  y_i+1 = y_i + h * f(t_i, y_i)
def euler(t0, y0, h, tn):
    t_values = np.arange(t0, tn+ h, h)
    y_values = [y0]

    for i in range(len(t_values) - 1):
        t = t_values[i]
        y = y_values[-1]
        y_next = y + h * f(t, y)
        y_values.append(y_next)

    return t_values, y_values
```

```python
# (B) Taylor's method of order 2:
#  y_i+1 = y_i + h * f(t_i, y_i) + (h^2 / 2) * f'(t_i, y_i)
def taylor_order2(t0, y0, h, tn):
    t_values = np.arange(t0, tn + h, h)
    y_values = [y0]

    for i in range(len(t_values) - 1):
        t = t_values[i]
        y = y_values[-1]
        y_next = y + h * f(t, y) + (h**2 / 2) * f_prime(t, y)
        y_values.append(y_next)

    return t_values, y_values
```

圖 2、第一題計算式

```python
# 主程式
def main():
    # INIT
    t0 = 1.0
    y0 = 0.0
    h = 0.1
    tn = 2.0

    # Euler
    t_euler, y_euler = euler(t0, y0, h, tn)

    # Taylor
    t_taylor, y_taylor = taylor_order2(t0, y0, h, tn)

    # exact
    y_exact = [exact_solution(t) for t in t_euler]

    # errors
    error_euler = [abs(y_euler[i] - y_exact[i]) for i in range(len(y_euler))]
    error_taylor = [abs(y_taylor[i] - y_exact[i]) for i in range(len(y_taylor))]

    results = {
        't': t_euler,
        'Euler': y_euler,
        'Taylor': y_taylor,
        'Exact': y_exact,
        'Error (Euler)': error_euler,
        'Error (Taylor)': error_taylor
    }

    df = pd.DataFrame(results)
    df = df.round(6)
    print(df)

if __name__ == "__main__":
    main()
```

圖 3、第一題結果顯示程式碼

第二題：

```
PS D:\ForClass\1132\1132Numerical\HW5> & C:/ProgramData/anaconda3/python.exe d:/ForClass/1132/1132Numerical/HW5/2.Cal_h=0.05.py
數值解與精確解比較：
       t    u1_rk4  u1_exact  u1_error    u2_rk4  u2_exact  u2_error
0   0.00  1.333333  1.333333  0.000000  0.666667  0.666667  0.000000
1   0.05  1.721880  1.912059  0.190178 -0.499599 -0.909077  0.409477
2   0.10  1.726915  1.793063  0.066148 -0.832598 -1.032002  0.199405
3   0.15  1.617161  1.601967  0.015194 -0.890373 -0.961459  0.071086
4   0.20  1.481687  1.423902  0.057785 -0.861042 -0.874681  0.013639
5   0.25  1.348945  1.267646  0.081299 -0.807505 -0.795221  0.012284
6   0.30  1.227063  1.131577  0.095487 -0.750341 -0.724999  0.025342
7   0.35  1.117478  1.012999  0.104480 -0.695886 -0.663060  0.032826
8   0.40  1.019525  0.909409  0.110117 -0.645732 -0.608214  0.037518
9   0.45  0.931977  0.818630  0.113347 -0.599934 -0.559389  0.040545
10  0.50  0.853541  0.738788  0.114753 -0.558092 -0.515658  0.042435
11  0.55  0.783017  0.668275  0.114743 -0.519706 -0.476225  0.043482
12  0.60  0.719337  0.605710  0.113627 -0.484290 -0.440411  0.043880
13  0.65  0.661560  0.549909  0.111651 -0.451407 -0.407635  0.043772
14  0.70  0.608868  0.499860  0.109007 -0.420673 -0.377404  0.043269
15  0.75  0.560547  0.454695  0.105852 -0.391754 -0.349296  0.042459
16  0.80  0.515980  0.413671  0.102309 -0.364365 -0.322954  0.041411
17  0.85  0.474633  0.376158  0.098475 -0.338259 -0.298076  0.040183
18  0.90  0.436043  0.341614  0.094428 -0.313226 -0.274409  0.038817
19  0.95  0.399812  0.309583  0.090229 -0.289089 -0.251739  0.037351
20  1.00  0.365600  0.279675  0.085925 -0.265698 -0.229888  0.035810

u1 的最大誤差: 1.901784e-01
u2 的最大誤差: 4.094772e-01
PS D:\ForClass\1132\1132Numerical\HW5>
```

圖 4、h=0.05, 0≦t≦1, RK4 檢驗結果

```
PS D:\ForClass\1132\1132Numerical\HW5> & C:/ProgramData/anaconda3/python.exe d:/ForClass/1132/1132Numerical/HW5/2.Cal_h=0.1.py
數值解與精確解比較（所有結果）：
      t        u1_rk4  u1_exact      u1_error        u2_rk4  u2_exact      u2_error
0   0.0  1.333333e+00  1.333333  0.000000e+00  6.666670e-01  0.666667  0.000000e+00
1   0.1 -3.052437e+00  1.793063  4.845500e+00  8.989305e+00 -1.032002  1.002131e+01
2   0.2 -2.384780e+01  1.423902  2.527170e+01  5.119270e+01 -0.874681  5.206739e+01
3   0.3 -1.301652e+02  1.131577  1.312968e+02  2.692692e+02 -0.724999  2.699942e+02
4   0.4 -6.802315e+02  0.909409  6.811409e+02  1.399369e+03 -0.608214  1.399977e+03
5   0.5 -3.531300e+03  0.738788  3.532038e+03  7.258242e+03 -0.515658  7.258757e+03
6   0.6 -1.831280e+04  0.605710  1.831340e+04  3.763496e+04 -0.440411  3.763540e+04
7   0.7 -9.495133e+04  0.499860  9.495183e+04  1.951319e+05 -0.377404  1.951322e+05
8   0.8 -4.923065e+05  0.413671  4.923069e+05  1.011722e+06 -0.322954  1.011722e+06
9   0.9 -2.552514e+06  0.341614  2.552514e+06  5.245579e+06 -0.274409  5.245579e+06
10  1.0 -1.323428e+07  0.279675  1.323428e+07  2.719729e+07 -0.229888  2.719729e+07

u1 的最大誤差: 1.323428e+07
u2 的最大誤差: 2.719729e+07
PS D:\ForClass\1132\1132Numerical\HW5>
```

圖 5、h=0.1, 0≦t≦1, RK4 檢驗結果之一（h=0.05 之程式修改參數）

```
PS D:\ForClass\1132\1132Numerical\HW5> & C:/ProgramData/anaconda3/python.exe d:/ForClass/1132/1132Numerical/HW5/2.Cal_h=0.1test2.py
Runge-Kutta方法 (h=0.1) 與精確解的比較:
    t        u1_num        u1_ex         err1        u2_num        u2_ex         err2
--------------------------------------------------------------------------------
  0.00      1.333333     1.333333     0.00e+00      0.666667     0.666667     1.11e-16
  0.10     -3.052437     1.793063     4.85e+00      8.989305    -1.032002     1.00e+01
  0.20    -23.847795     1.423902     2.53e+01     51.192704    -0.874681     5.21e+01
  0.30   -130.165202     1.131577     1.31e+02    269.269193    -0.724999     2.70e+02
  0.40   -680.231485     0.909409     6.81e+02   1399.368584    -0.608214     1.40e+03
  0.50  -3531.299585     0.738788     3.53e+03   7258.241839    -0.515658     7.26e+03
  0.60 -18312.795052     0.605710     1.83e+04 37634.955483    -0.440411     3.76e+04
  0.70 -94951.331907     0.499860     9.50e+04 195131.871735    -0.377404     1.95e+05
  0.80 -492306.465639     0.413671     4.92e+05 1011721.872078    -0.322954     1.01e+06
  0.90 -2552513.623867     0.341614     2.55e+06 5245578.826590    -0.274409     5.25e+06
  1.00 -13234278.789168     0.279675     1.32e+07 27197287.206587    -0.229888     2.72e+07
PS D:\ForClass\1132\1132Numerical\HW5>
```

圖 6、h=0.1, 0≦t≦1, RK4 檢驗結果之二（另外一種程式碼）

使用了不同程式碼來計算 h=0.1 之數值解，都產生了相同數值但卻非常發散的數值，根據查詢網路資料，發現 RK4 這種算法有一定的「剛性」存在(也就是穩定度)，且根據初始條件的係數矩陣可求得特徵值。

h=0.1 時的最大特徵值(e^{-39t}時)產生的 λh 值大於其穩定邊界，所以計算會產生發散；但 h=0.05 時，最大特徵值計算出的 λh 則位於穩定邊界內，故誤差相對可接受，但此算法需要更細的步長來降低誤差值。