

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №6

Выполнил: Орехов Сергей Владимирович

Р32151

Преподаватель: Машина Екатерина Алексеевна

Санкт-Петербург

2023

Цель работы

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Листинг методов:

import math  
import sys  
  
import numpy as np  
  
  
def euler(f, x0, y0, h, xn):  
 X = []  
 Y = []  
 X.append(x0)  
 Y.append(y0)  
 while x0 < xn:  
 y0 = y0 + h \* f(x0, y0)  
 x0 += h  
 X.append(x0)  
 Y.append(y0)  
 return X, Y  
  
  
def fourth\_order\_runge\_kutta(f, x0, y0, h, xn):  
 X, Y = [], []  
 X.append(x0)  
 Y.append(y0)  
 while x0 < xn:  
 k1 = h \* f(x0, y0)  
 k2 = h \* f(x0 + h / 2, y0 + k1 / 2)  
 k3 = h \* f(x0 + h / 2, y0 + k2 / 2)  
 k4 = h \* f(x0 + h, y0 + k3)  
 x0 += h  
 y0 = y0 + 1 / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4)  
 X.append(x0)  
 Y.append(y0)  
 return X, Y  
  
  
def milna(f, x, y, h, xn, eps):  
 xnn = x + h \* 4  
 X, Y = fourth\_order\_runge\_kutta(f, x, y, h, xnn)  
 F = []  
 for i in range(4):  
 F.append(f(X[i], Y[i]))  
  
 # Prediction part:  
  
 while X[-1] < xn:  
 X.append(X[-1] + h)  
 Y.append(Y[-4] + 4 \* h / 3 \* (2 \* F[-3] - F[-2] + 2 \* F[-1]))  
 Fi = f(X[-1], Y[-1])  
 y\_correct = Y[-2] + h / 3 \* (F[-2] + 4 \* F[-1] - Fi)  
  
 # Correction part:  
 while abs(y\_correct - Y[-1]) >= eps:  
 Y[-1] = y\_correct  
 Fi = f(X[-1], y\_correct)  
 y\_correct = Y[-2] + h / 3 \* (F[-2] + 4 \* F[-1] - Fi)  
 Y[-1] = y\_correct  
 F[-1] = Fi  
 return X, Y  
  
  
def runge\_rule(method, f, x0, y0, h, xn, p, eps=None):  
 eps\_array = []  
 if eps is None:  
 eps = 0.01  
 X, Y1 = method(f, x0, y0, h, xn)  
 Y2 = method(f, x0, y0, h / 2, xn)[1]  
 else:  
 X, Y1 = method(f, x0, y0, h, xn, eps)  
 Y2 = method(f, x0, y0, h / 2, xn, eps)[1]  
  
 for i in range(1, int(math.ceil((xn - x0) / h))):  
 res = abs(Y1[i] - Y2[i]) / (2 \*\* p - 1)  
 eps\_array.append(res)  
  
 if len(eps\_array) == 0:  
 sys.exit("Bad interval! Expected more elements")  
 is\_optimal = max(eps\_array) < eps  
 eps\_array = np.array(eps\_array)  
 return eps\_array.mean(), "h is OK" if is\_optimal else "h is NOT OK"

Пример работы:

Пример 1:

choose function: from 0 to 4

1

'0' - Euler; '1' - Runge Kutta; '2' - Milna

1

Enter x0

1

Enter y0

1

Enter h

0.1

Enter xn

3

X Y original Y from method Runge Rule = Epsilon\_mean Rate Runge Rule

0 1.0 1.0 1.000000 0.253357 h is NOT OK

1 1.1 3.3 1.215512 -/- -/-

2 1.2 3.6 1.464208 -/- -/-

3 1.3 3.9 1.749575 -/- -/-

4 1.4 4.2 2.075473 -/- -/-

5 1.5 4.5 2.446162 -/- -/-

6 1.6 4.8 2.866354 -/- -/-

7 1.7 5.1 3.341255 -/- -/-

8 1.8 5.4 3.876619 -/- -/-

9 1.9 5.7 4.478804 -/- -/-

10 2.0 6.0 5.154839 -/- -/-

11 2.1 6.3 5.912490 -/- -/-

12 2.2 6.6 6.760342 -/- -/-

13 2.3 6.9 7.707879 -/- -/-

14 2.4 7.2 8.765587 -/- -/-

15 2.5 7.5 9.945052 -/- -/-

16 2.6 7.8 11.259079 -/- -/-

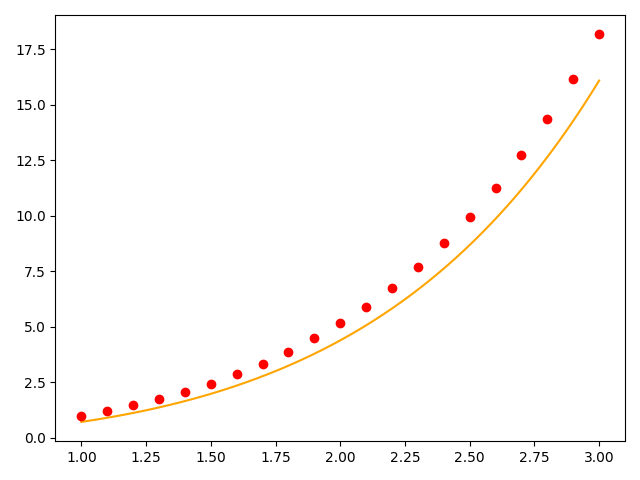
17 2.7 8.1 12.721821 -/- -/-

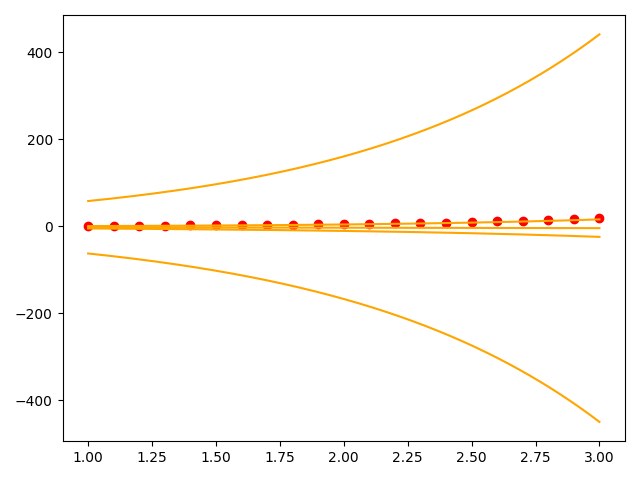
18 2.8 8.4 14.348917 -/- -/-

19 2.9 8.7 16.157654 -/- -/-

20 3.0 9.0 18.167134 -/- -/-

Process finished with exit code 0





Пример 2:

choose function: from 0 to 4

2

'0' - Euler; '1' - Runge Kutta; '2' - Milna

2

Enter x0

2

Enter y0

2

Enter h

0.1

Enter xn

4

Enter Epsilon

0.01

X Y original Y from method Runge Rule = Epsilon\_mean Rate Runge Rule

0 2.0 2.000000 2.000000 0.036163 h is NOT OK

1 2.1 0.535226 1.909916 -/- -/-

2 2.2 0.761934 1.808226 -/- -/-

3 2.3 1.021025 1.698176 -/- -/-

4 2.4 1.304999 1.582986 -/- -/-

5 2.5 1.604578 1.435016 -/- -/-

6 2.6 1.909072 1.282819 -/- -/-

7 2.7 2.206835 1.134790 -/- -/-

8 2.8 2.485792 0.993373 -/- -/-

9 2.9 2.734003 0.860399 -/- -/-

10 3.0 2.940255 0.737240 -/- -/-

11 3.1 3.094640 0.624601 -/- -/-

12 3.2 3.189096 0.522662 -/- -/-

13 3.3 3.217884 0.431193 -/- -/-

14 3.4 3.177976 0.349648 -/- -/-

15 3.5 3.069329 0.277259 -/- -/-

16 3.6 2.895032 0.213115 -/- -/-

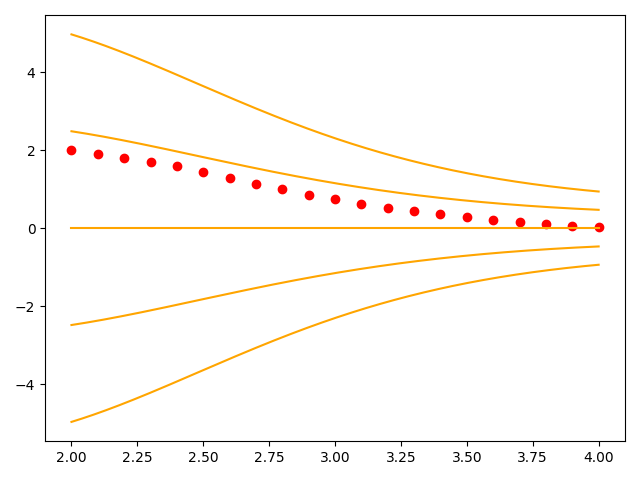
17 3.7 2.661313 0.156232 -/- -/-

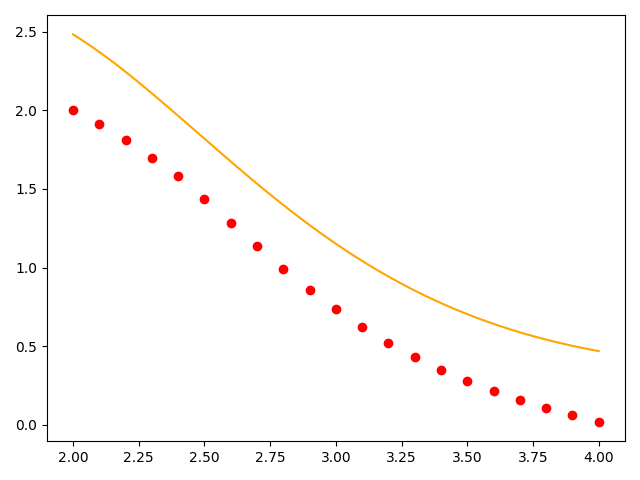
18 3.8 2.377394 0.105606 -/- -/-

19 3.9 2.055213 0.060249 -/- -/-

20 4.0 1.709000 0.019215 -/- -/-

Process finished with exit code 0





Вывод

Реализация численных методов решения обыкновенных дифференциальных уравнений самая сложная задача, которую необходимо было решать в рамках данного курса, на мой взгляд.