Санкт-Петербургский государственный политехнический университет им. Петра Великого

Кафедра “Прикладная математика”

Отчет по лабораторной работе №8
Языки программирования в биоинформатике

Работу выполнили студент группы № 5030102/10401
Дубовицкий Владислав Александрович
Преподаватель
Козлов Константин Николаевич

Санкт-Петербург, 2024

Оглавление

Условие

Требуется запрограммировать Модифицированный метод Эйлера. Программа должна работать для произвольной размерности системы уравнений.

Функция правой части системы и начальное условие подаются на вход программе. Вычисления должны производиться с пошаговым контролем точности по правилу Рунге. Если на текущем шаге точность не достигается, то шаг уменьшается в 2 раза, если достигнутая погрешность меньше заданной в 64 раза, то шаг увеличивается в 2 раза.

Входные данные

На вход подаются несколько строк, в которых:

- начало промежутка t_0
- конец промежутка T
- начальный шаг h_0
- максимальное число вызовов функции правой части N_x
- желаемая точность eps
- число уравнений
- следующие n+3 строк определяют функцию правой части на Python
- последняя строка содержит n чисел - начальное условие

```
t0 = 1.5
T = 2.5
h0 = 0.1
N_x = 10000
eps = 0.0001
n = 3
def fs(t, v, kounter):
#
#
  A = np.array([[-0.4, 0.02, 0], [0, 0.8, -0.1], [0.003, 0, 1]])
  kounter[0] += 1
  return np.dot(A, v)
```

Рисунок 1. “Входные данные”

Выходные данные

Программа печатает в консоль следующие столбцы, одна строчка соответствует одному шагу интегрирования:

1. значение t
2. значение шага h
3. оценка Рунге R
4. истраченое число вычислений правой части N
5. значения функций решений

Алгоритм

```
while t < T and kounter[0] < N_x:
    v_First = euler_Modf(t,v,h)
    v_Second = euler_Modf(t,v,h/2)
    v_Second = euler_Modf(t + h/2, v_Second, h/2)

    R = np.linalg.norm(v_First - v_Second) / (pow(2,2) - 1)

    if R > eps:
        h /= 2

    elif R < (eps / 64):
        h *= 2

    else:
        v = v_First
        t += h

    if t + h > T:
        h = T - t
```

Рисунок 2. “Алгоритм»

Результат

```
Eps =  0.0001
    1.500000     0.100000 0                    0     1.000000     1.000000     2.000000
    1.600000     0.100000  8.45154e-05         6     0.962820     1.061398     2.210309
    1.700000     0.100000  9.33737e-05        12     0.927221     1.125613     2.442690
    1.750000     0.050000  1.28171e-05        24     0.909992     1.158775     2.568019
    1.800000     0.050000  1.34742e-05        30     0.893138     1.192634     2.699768
    1.850000     0.050000  1.41654e-05        36     0.876652     1.227187     2.838267
    1.900000     0.050000  1.48925e-05        42     0.860527     1.262426     2.983862
    1.950000     0.050000  1.56573e-05        48     0.844756     1.298342     3.136916
    2.000000     0.050000  1.64617e-05        54     0.829333     1.334924     3.297812
    2.050000     0.050000  1.73079e-05        60     0.814253     1.372157     3.466951
    2.100000     0.050000  1.81979e-05        66     0.799508     1.410026     3.644757
    2.150000     0.050000  1.91341e-05        72     0.785092     1.448511     3.831672
    2.200000     0.050000  2.01189e-05        78     0.771001     1.487590     4.028165
    2.250000     0.050000  2.11548e-05        84     0.757227     1.527236     4.234726
    2.300000     0.050000  2.22444e-05        90     0.743766     1.567420     4.451871
    2.350000     0.050000  2.33906e-05        96     0.730612     1.608110     4.680143
    2.400000     0.050000  2.45962e-05       102     0.717758     1.649267     4.920111
    2.450000     0.050000  2.58644e-05       108     0.705200     1.690849     5.172377
    2.500000     0.050000  2.71984e-05       114     0.692932     1.732810     5.437568
```

Рисунок 3. «Пример результатов вычисления»

Графики

Изменение шага по отрезку для разных значений заданной точности

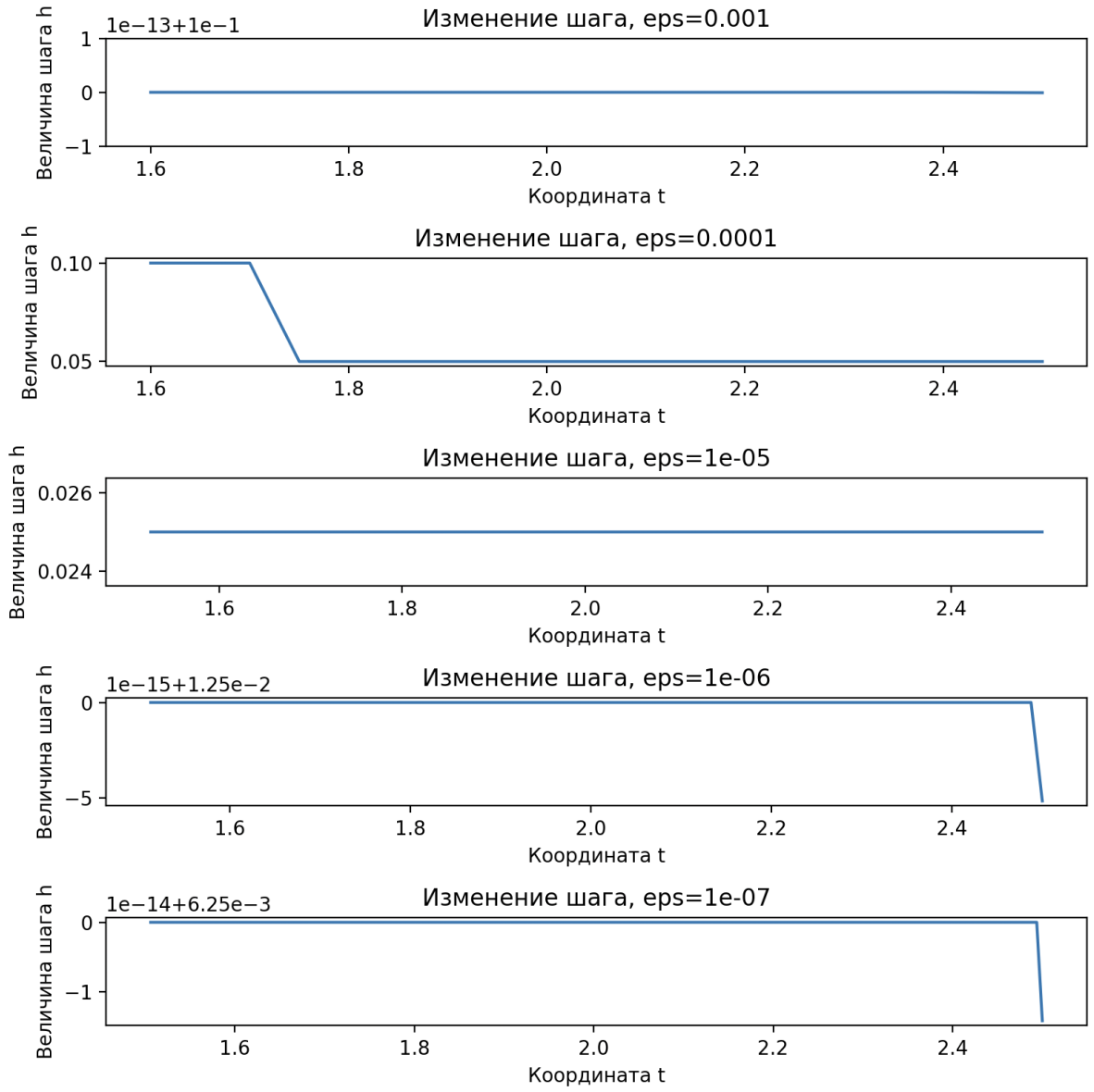

Рисунок 4. “График изменения шага по отрезку для разных значений заданной точности”

График зависимости минимального шага от заданной точности

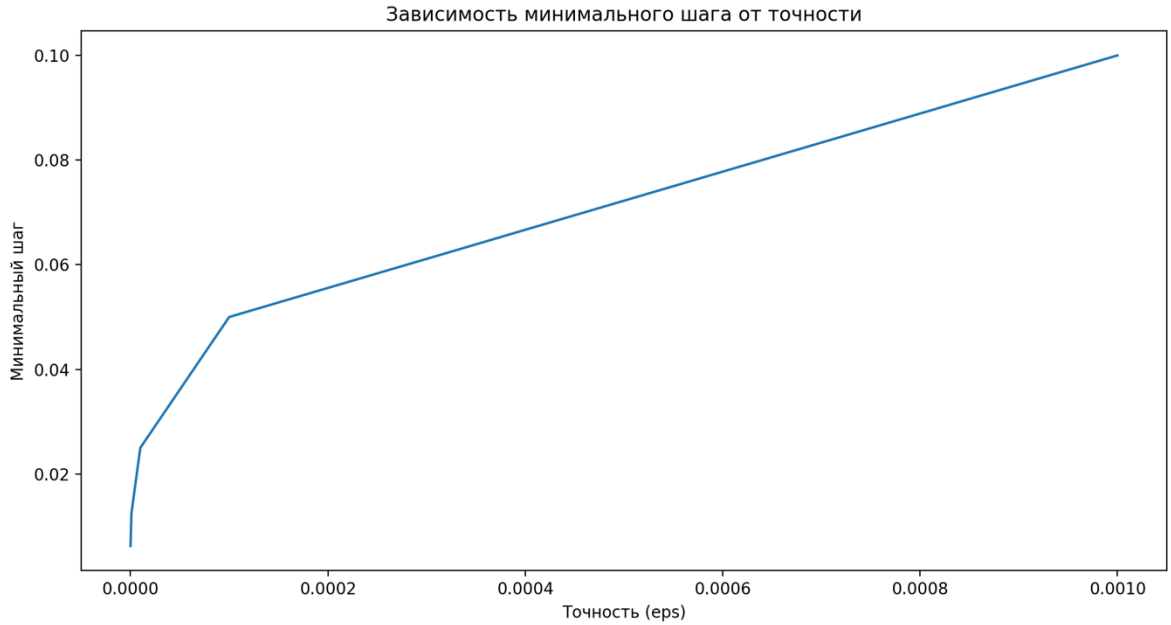

Рис. 5 «График зависимости минимального шага от заданной точности”

График зависимости числа шагов от заданной точности

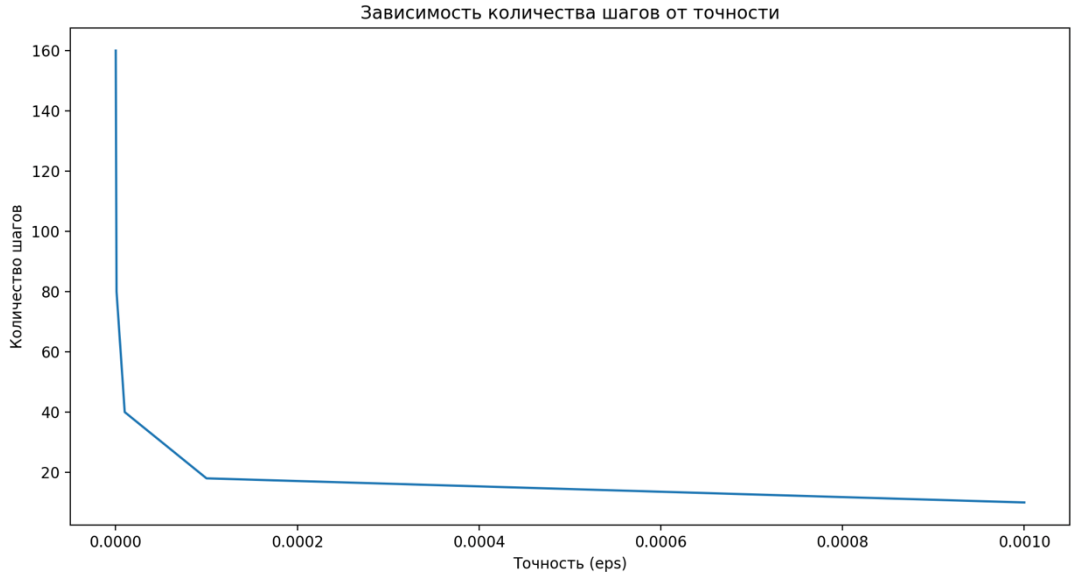

Рис. 6 “График зависимости числа шагов от заданной точности”

Графики изменения решения для разных значений заданной точности

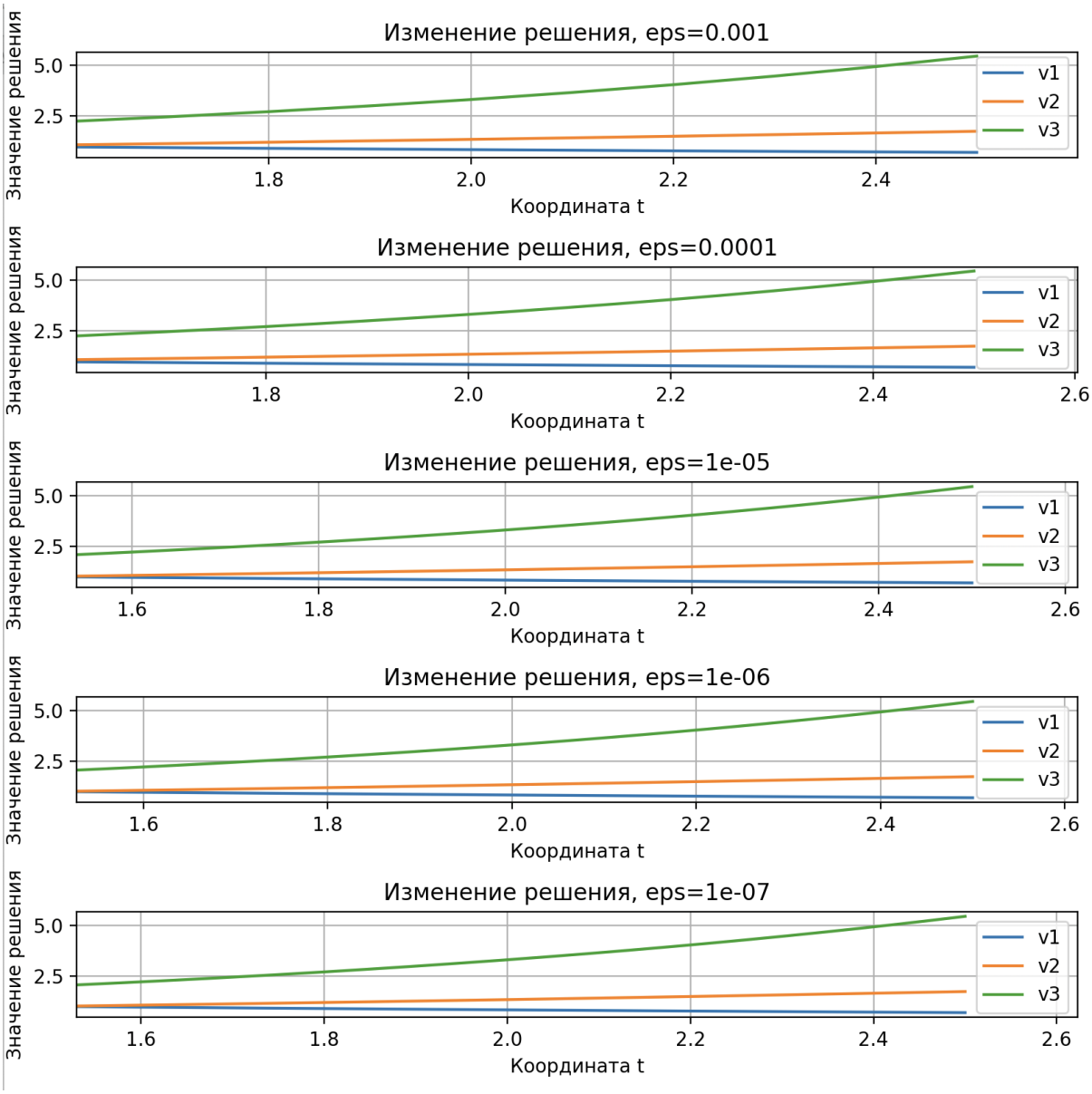

Рис 7. “Графики изменения решения для разных значений заданной точности”

Код

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

def modified_euler_method(rhs_func, initial_condition, t0, T, h0, N_x,
eps):
    t = t0
    h = h0
    v = np.array(initial_condition)
    kounter = [0]
    results = []
    steps = []
    solutions = []
    coord = []

    print("{:12.6f} {:12.6f} {:12s} {:12d} {:12.6f} {:12.6f}
{:12.6f}".format(
        t, h, "0", kounter[0], *v))

    def euler_Modf(t,v,h):
        v_hat = rhs_func(t, v, kounter)
        v_tilde = rhs_func(t + h, v + h * v_hat, kounter)
        return  v + (h / 2) * (v_hat + v_tilde)



    while t < T and kounter[0] < N_x:
        v_First = euler_Modf(t,v,h)
        v_Second = euler_Modf(t,v,h/2)
        v_Second = euler_Modf(t + h/2, v_Second, h/2)

        R = np.linalg.norm(v_First - v_Second) / (pow(2,2) - 1)

        if R > eps:
            h /= 2

        elif R < (eps / 64):
            h *= 2

        else:
            v = v_First
            t += h
            steps.append(h)
            solutions.append(v.copy())
            coord.append(t)

            print("{:12.6f} {:12.6f} {:12.5e} {:12d} {:12.6f} {:12.6f}
{:12.6f}".format(
                t, h, R, kounter[0], *v))

        if t + h > T:
```

```
            h = T - t

        results.append((t, h, R, kounter[0], *v))

    return results, steps, solutions, coord

t0 = 1.5
T = 2.5
h0 = 0.1
N_x = 10000
eps = 0.0001
n = 3
eps_count = [0.001, 0.0001, 0.00001, 0.000001, 0.0000001]

kounter_mas = []

function_code = []
for i in range(n+3):
    line = input()
    function_code.append(line)

func = '\n'.join(function_code)
exec(func)

#def fs(t, v, kounter):
#    A = np.array([[-0.4, 0.02, 0], [0, 0.8, -0.1], [0.003, 0, 1]])
#    kounter[0] += 1
#    return np.dot(A, v)

initial_condition = [1, 1, 2]

results, steps, solutions, coord = modified_euler_method(fs,
initial_condition, t0, T, h0, N_x, eps)
```