Санкт-Петербургский политехнический университет
Высшая школа прикладной математики и вычислительной физики, ФизМех

Направление подготовки
«01.03.02 Прикладная математика и информатика»
Специальность «Биоинформатика»

Лабораторная работа №8
тема "Решение задачи Коши"

Выполнила студент гр. 5030102/10401:

Губанов Алексей

Преподаватель:

Козлов Константин Николаевич

Санкт-Петербург

2024

Содержание

Формулировка задания

Необходимо создать программу, реализующую метод Хёйна для решения задачи Коши для систем обыкновенных дифференциальных уравнений (ОДУ) произвольной размерности. Программа должна позволять пользователю задавать функцию-правую часть системы ОДУ и начальные условия.

Программа должна вычислять решение ОДУ с пошаговым контролем точности по правилу Рунге. Если на текущем шаге точность не достигнута, то шаг уменьшается вдвое. Если достигнутая погрешность меньше заданной в 64 раза, то шаг увеличивается вдвое.

По полученным результатам программа должна построить следующие графики с использованием библиотеки Matplotlib:

- Изменение шага по отрезку для разных значений заданной точности
- Зависимость минимального шага от заданной точности
- Зависимость числа шагов от заданной точности
- Решение ОДУ для разных значений заданной точности

Алгоритм метода Хойна

```
k1 = fs(t, v, call_counter)
k2 = fs(t + h, v + h * k1, call_counter)
v1 = v + (h / 2) * (k1 + k2)
k2 = fs(t + h / 2, v + h / 2 * k1, call_counter)
v2 = v + (h / 4) * (k1 + k2)
v2 = heun_step(t + h / 2, v2, h / 2)
```

Запуск функции

Вывод организован в виде столбцов, где каждая строка соответствует одному шагу интегрирования:

1. Значение t.
2. Значение шага h.
3. Оценка по методу Рунге R.
4. Истраченное количество вычислений правой части N.
5. Значения функций-решений.

Пример выполнения программы

```
t_0 = 1.5
T = 2.5
h_0 = 0.1
N_x = 10000
eps = 0.001
n = 3
```

[107] ✓ 0.1s

```
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_conditions = [1, 1, 2]
```

[108] ✓ 0.2s

1.500000	0.100000	0	0	1.000000	1.000000	2.000000
1.600000	0.100000	8.45154e-05	5	0.962820	1.061398	2.210309
1.700000	0.100000	9.33737e-05	10	0.927221	1.125613	2.442690
1.800000	0.100000	1.03174e-04	15	0.893145	1.192637	2.699459
1.900000	0.100000	1.14015e-04	20	0.860540	1.262439	2.983178
2.000000	0.100000	1.26009e-04	25	0.829352	1.334956	3.296678
2.100000	0.100000	1.39277e-04	30	0.799532	1.410090	3.643086
2.200000	0.100000	1.53956e-04	35	0.771031	1.487698	4.025858
2.300000	0.100000	1.70196e-04	40	0.743801	1.567592	4.448812
2.400000	0.100000	1.88162e-04	45	0.717797	1.649522	4.916167
2.500000	0.100000	2.08039e-04	50	0.692976	1.733175	5.432587

Графики

Рисунок 1. График зависимости числа шагов от точности

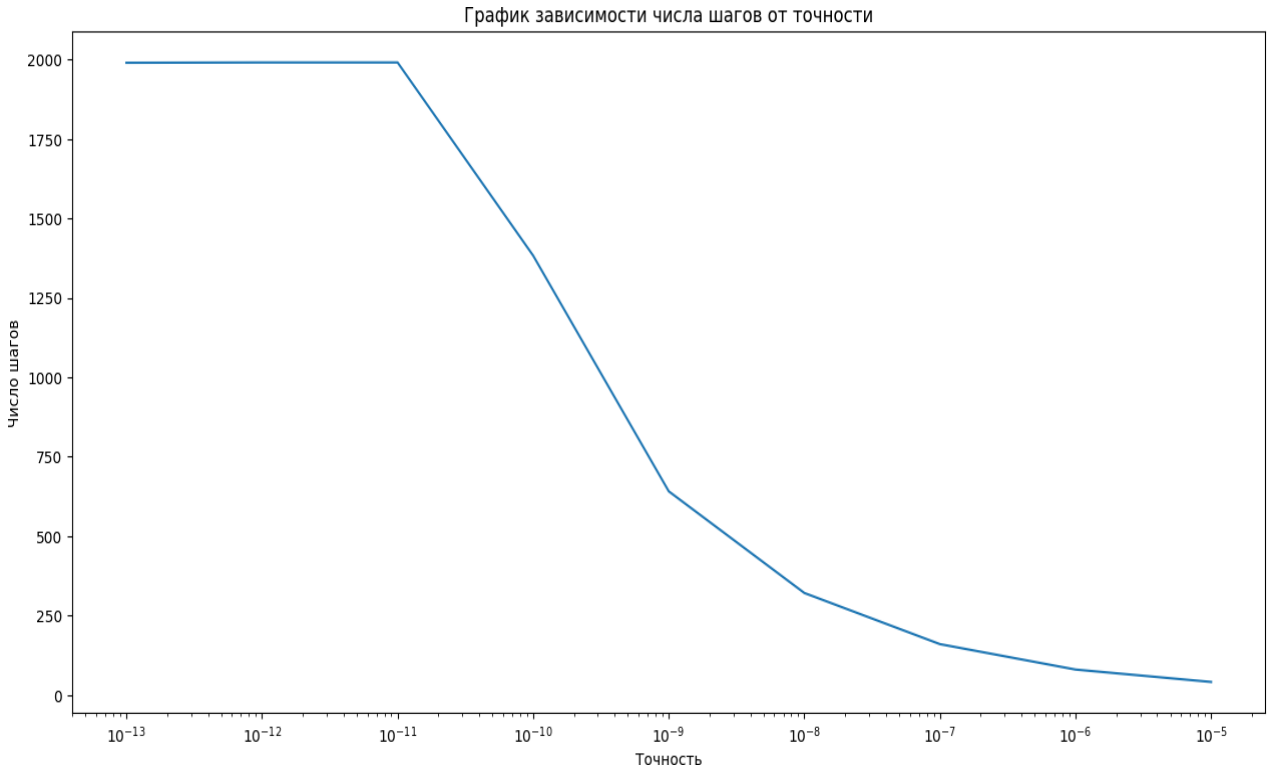

Рисунок 2. Зависимость минимального шага от точности

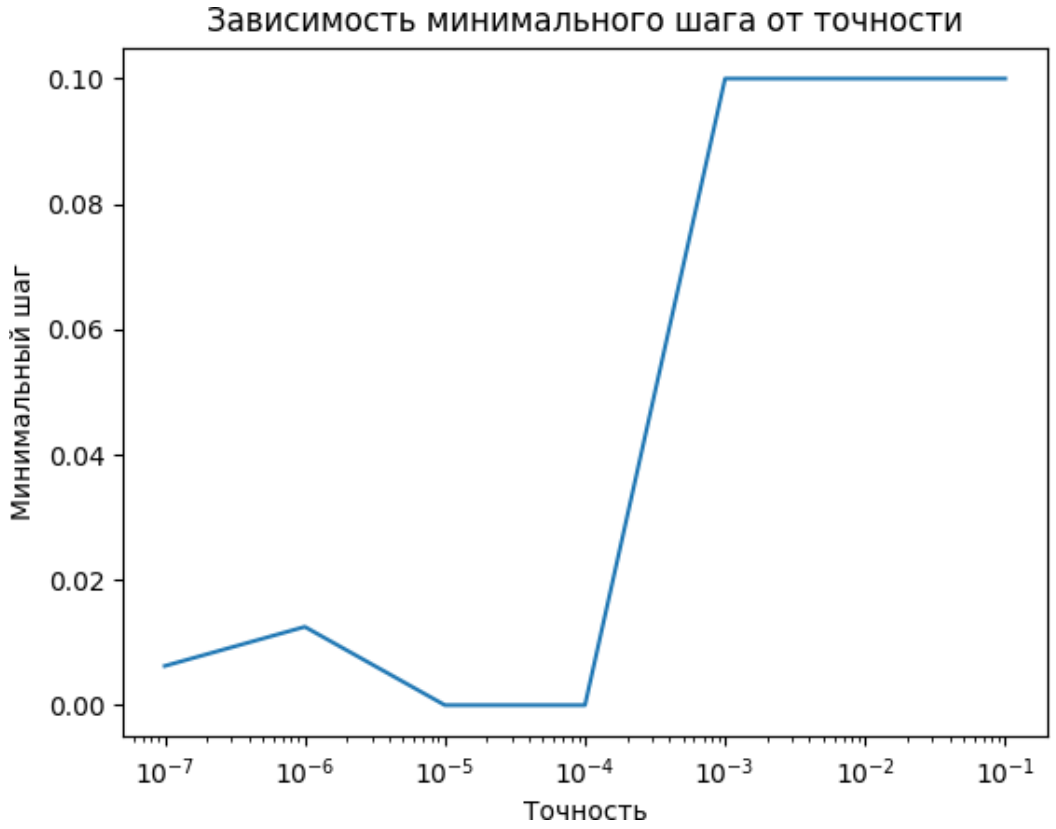

Рисунок 3. Изменение шага по отрезку для разных значений заданной точностью

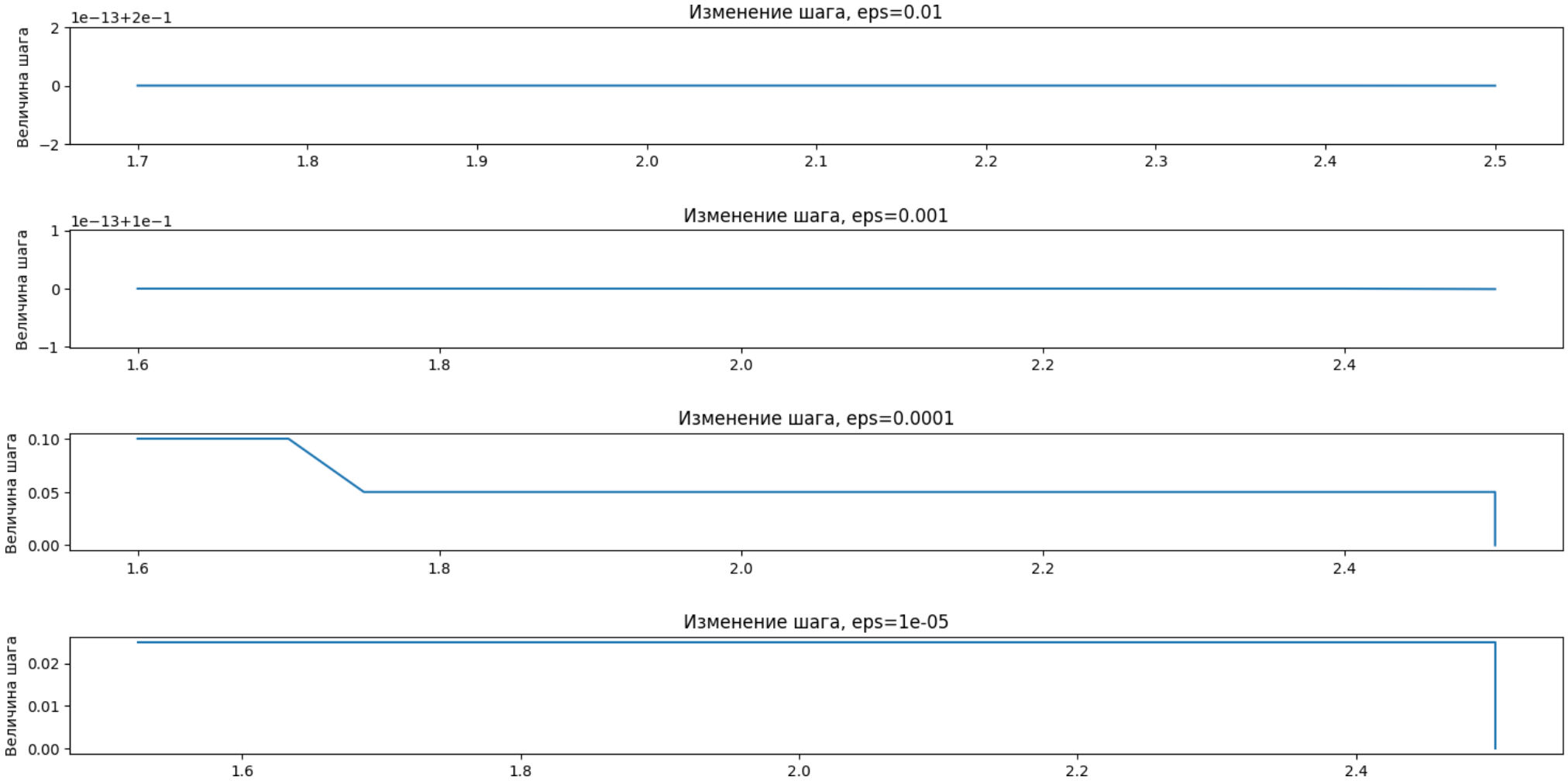

Рисунок 4. Решение для разных значений заданной точности