Санкт-Петербургский Политехнический Университет

Физико-Механический институт

Кафедра прикладной математики и вычислительной физики

Лабораторная работа №8

Вариант №4

Выполнил студент группы 5030102/10401
Бурмакова Дарья Александровна

Проверил
Козлов Константин Николаевич

31 мая 2024 г.

1 Введение

Метод Рунге-Кутта 4-го порядка (RK4) — это численный метод решения обыкновенных дифференциальных уравнений (ОДУ) вида

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

Метод RK4 обеспечивает высокую точность за счёт вычисления промежуточных шагов на каждом шаге интегрирования.

2 Формула метода

Метод Рунге-Кутта 4-го порядка использует следующие формулы для вычисления y_{n+1} на основе y_n:

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\
k_4 &= f(t_n + h, y_n + hk_3).
\end{aligned}$$

Затем новое значение y_{n+1} вычисляется как:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

где h — шаг интегрирования, t_n — текущее значение времени, y_n — текущее значение функции.

3 Пошаговый алгоритм

Алгоритм метода Рунге-Кутта 4-го порядка можно описать следующим образом:

1. Задать начальные условия: $t_0 = t_{\text{нач}}$ и $y_0 = y(t_{\text{нач}})$.
2. Выбрать шаг интегрирования h.
3. Для каждого шага n:
   (a) Вычислить промежуточные значения:

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\
k_4 &= f(t_n + h, y_n + hk_3).
\end{aligned}$$

(b) Обновить значение функции:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

(c) Перейти к следующему шагу: $t_{n+1} = t_n + h$.

4. Повторять шаги, пока не будет достигнуто конечное время $t_{\text{кон}}$.

4 Постановка задачи

Требуется запрограммировать метод Рунге-Кутты 3-го порядка решения задачи Коши для ОДУ. 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.

5 Пример выполнения и результаты

Полученные результаты указаны на следующих страницах.

```
1.5
2.5
0.1
10000
0.0001
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)
1 1 2
     1.500000     0.100000            0            0     1.000000     1.000000     2.000000
     1.700000     0.200000  1.07326e-08           12     0.962810     1.061402     2.210651
     2.100000     0.400000  3.85407e-07           24     0.893118     1.192625     2.700706
     2.500000     0.400000  1.55356e-05           36     0.770977     1.487538     4.029964

Process finished with exit code 0
```

Рис. 1: Пример работы программы

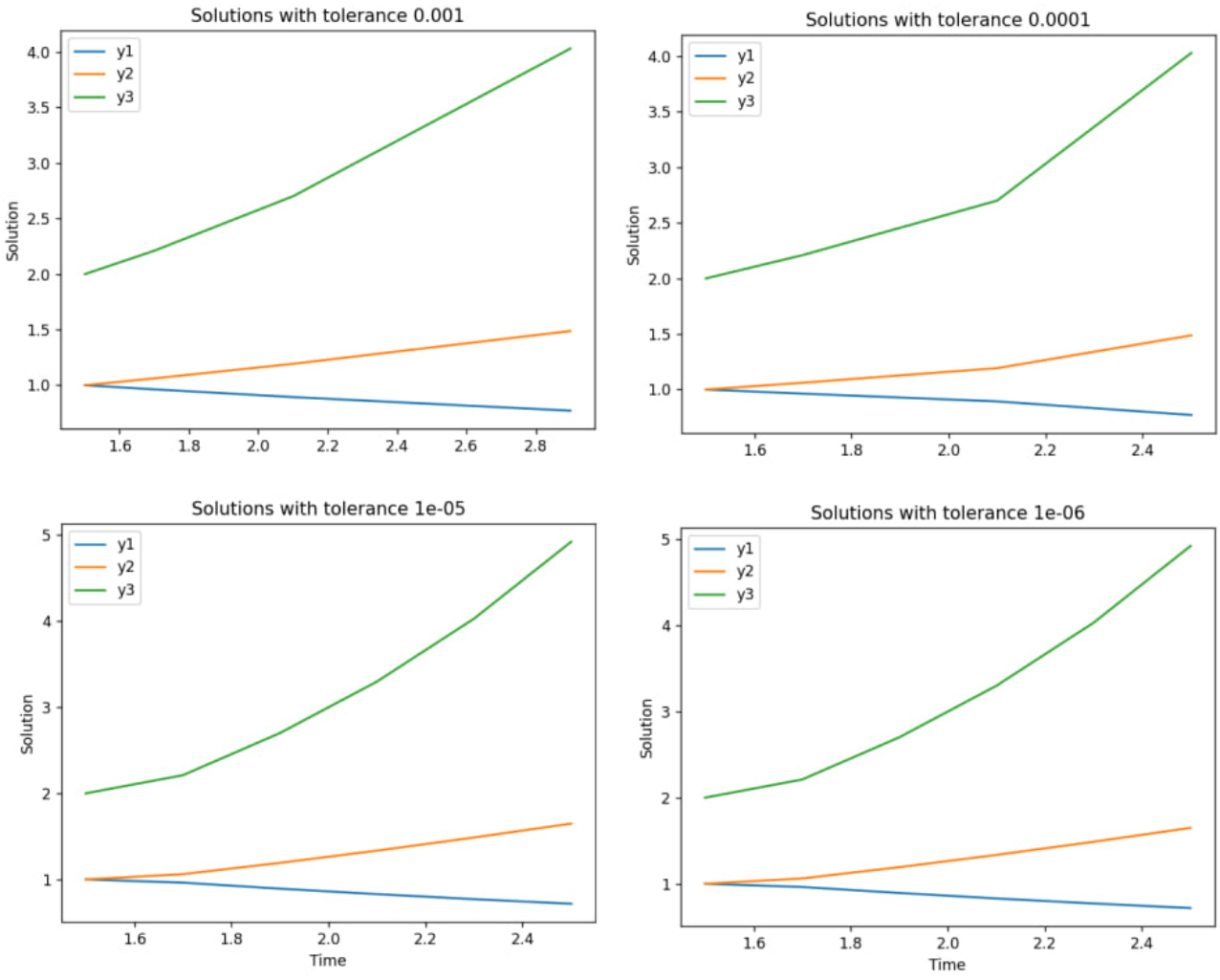

Рис. 2: Полученные решения для различных точностей

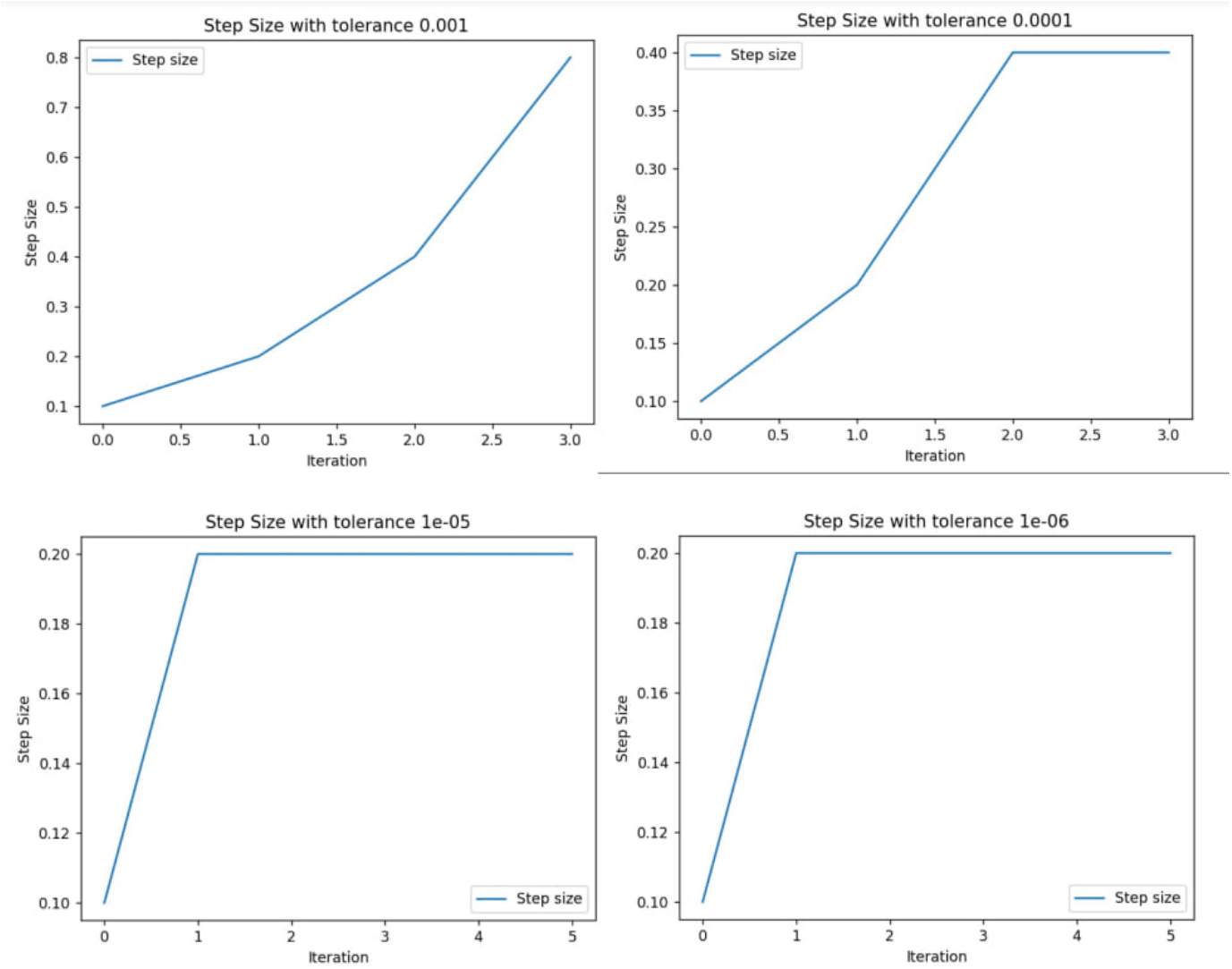

Рис. 3: Размеры шагов для различных точностей

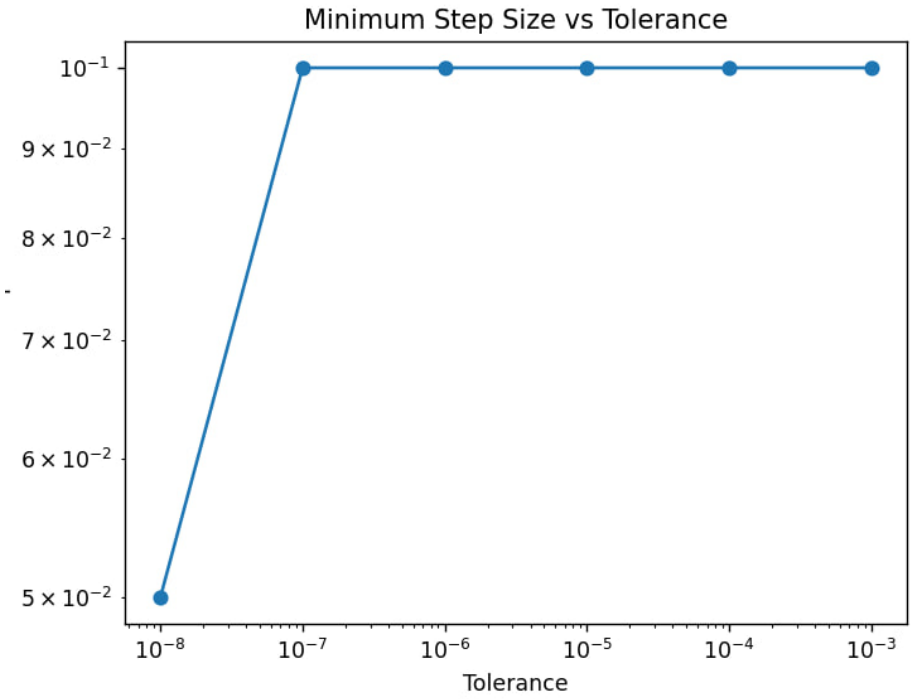

Рис. 4: Размеры минимальных шагов для различных точностей

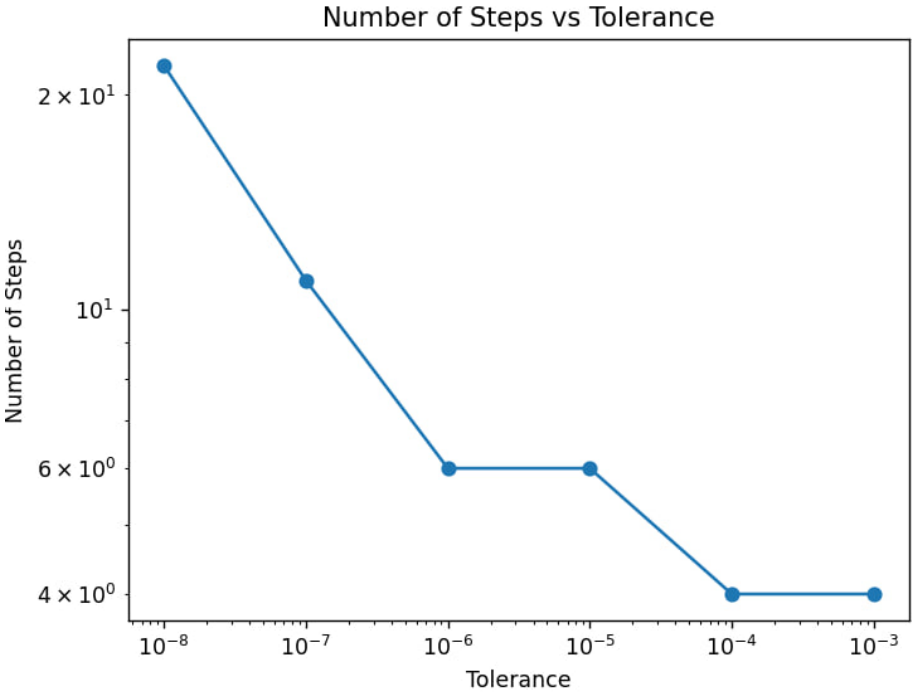

Рис. 5: Количество шагов для различных точностей