«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия  
Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №6

Вариант 1

Студент:

Алхимовици А.

Р3210

Преподаватель:

Наумова Н. А.

Санкт-Петербург, 2025 г.

# **Цель работы**:

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

# **Программная реализация**

# **Блок схемы**

Метод Эйлера:

A diagram of a algorithm

AI-generated content may be incorrect.

Метод Рунге-Кутта 4-го порядка:

A diagram of a mathematical equation

AI-generated content may be incorrect.

Многошаговый метод Адамса:

A diagram of a flowchart

AI-generated content may be incorrect.

**Листинг программы**

<https://github.com/senya-2011/Vu4Math/tree/main/lab6>

|  |
| --- |
| def solve\_adams4(f, x0, y0, xn, h):  n\_steps = int((xn - x0) / h)  xs = [x0 + i\*h for i in range(n\_steps + 1)]  ys = [0.0] \* (n\_steps + 1)  ys[0] = y0   if n\_steps < 3:  xs\_rk, ys\_rk = solve\_rk4(f, x0, y0, xn, h)  return xs\_rk, ys\_rk   xs\_init, ys\_init = solve\_rk4(f, x0, y0, x0 + 3\*h, h)  for i in range(4):  ys[i] = ys\_init[i]   def Fi(i):  return f(xs[i], ys[i])    for i in range(3, n\_steps):   P = ys[i] + (h/24) \* (  55\*Fi(i) - 59\*Fi(i-1) + 37\*Fi(i-2) - 9\*Fi(i-3)  )   ys[i+1] = ys[i] + (h/24) \* (  9 \* f(xs[i+1], P) + 19\*Fi(i) - 5\*Fi(i-1) + Fi(i-2)  )  return xs, ys  def max\_error\_adams(f\_exact, xs, ys):   max\_err = 0.0  for xi, yi in zip(xs, ys):  y\_ex = f\_exact(xi)  err = abs(yi - y\_ex)  if err > max\_err:  max\_err = err  return max\_err  def solve\_euler(f, x0, y0, xn, h):  n\_steps = int((xn - x0) / h)  xs = [x0 + i\*h for i in range(n\_steps + 1)]  ys = [0.0] \* (n\_steps + 1)  ys[0] = y0  for i in range(n\_steps):  xi = xs[i]  yi = ys[i]  ys[i+1] = yi + h \* f(xi, yi)  return xs, ys  def runge\_error(f, x0, y0, xn, h, method\_func, p):  \_, ys\_h = method\_func(f, x0, y0, xn, h)  y\_end\_h = ys\_h[-1]  \_, ys\_h2 = method\_func(f, x0, y0, xn, h/2)  y\_end\_h2 = ys\_h2[-1]  factor = 2\*\*p - 1  if factor == 0:  return None  return abs(y\_end\_h - y\_end\_h2) / factor  def solve\_rk4(f, x0, y0, xn, h):  n\_steps = int((xn - x0) / h)  xs = [x0 + i\*h for i in range(n\_steps + 1)]  ys = [0.0] \* (n\_steps + 1)  ys[0] = y0  for i in range(n\_steps):  xi = xs[i]  yi = ys[i]  k1 = f(xi, yi)  k2 = f(xi + h/2, yi + h\*k1/2)  k3 = f(xi + h/2, yi + h\*k2/2)  k4 = f(xi + h, yi + h\*k3)  ys[i+1] = yi + (h/6) \* (k1 + 2\*k2 + 2\*k3 + k4)  return xs, ys  def runge\_error(f, x0, y0, xn, h, method\_func, p):  \_, ys\_h = method\_func(f, x0, y0, xn, h)  y\_end\_h = ys\_h[-1]  \_, ys\_h2 = method\_func(f, x0, y0, xn, h/2)  y\_end\_h2 = ys\_h2[-1]  factor = 2\*\*p - 1  if factor == 0:  return None  return abs(y\_end\_h - y\_end\_h2) / factor |

## **Примеры и результаты работы программы**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# **Выводы**:

В лабораторной работе изучены и применены численные методы решения ОДУ: Эйлера, Рунге-Кутты 4-го порядка и Адамса. Алгоритмы реализованы на Python с использованием правила Рунге для оценки погрешности в одношаговых методах. Точность и эффективность методов сравнены графически, что наглядно показало их особенности. Практика углубила понимание численных подходов к решению ОДУ и важность выбора подходящего метода для конкретной задачи.