«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.

P3210

Преподаватель:

Наумова Н. А.

Санкт-Петербург, 2025 г.

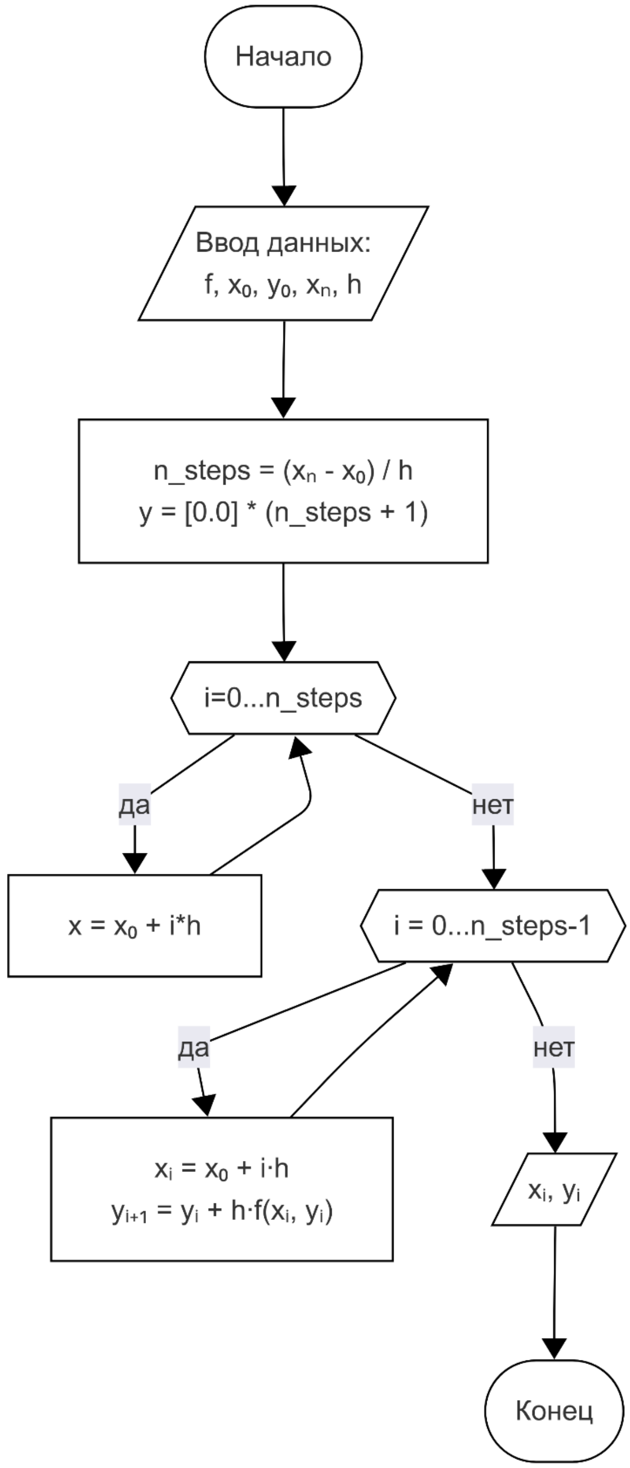
Цель работы:

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

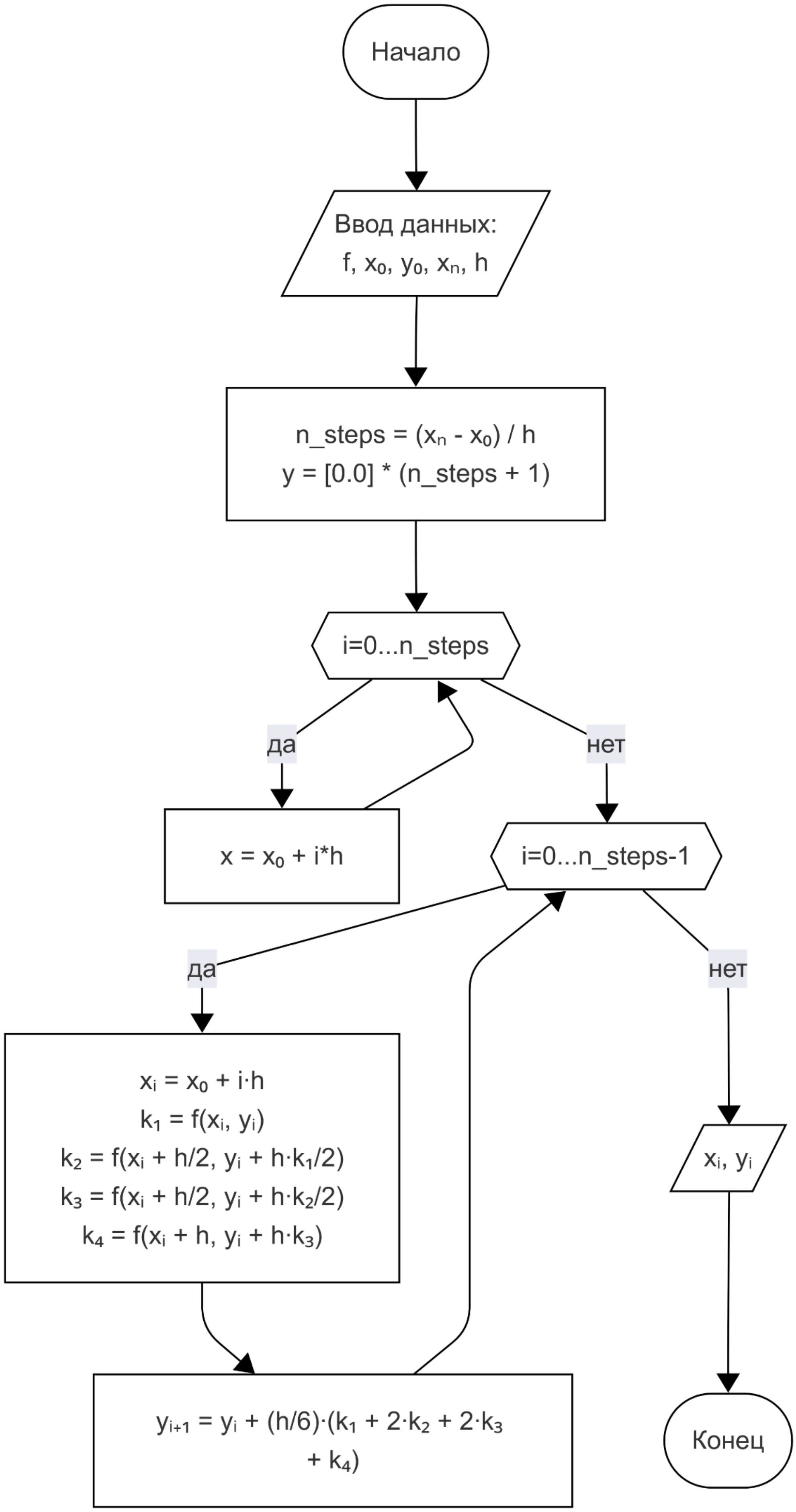
Программная реализация

Блок схемы

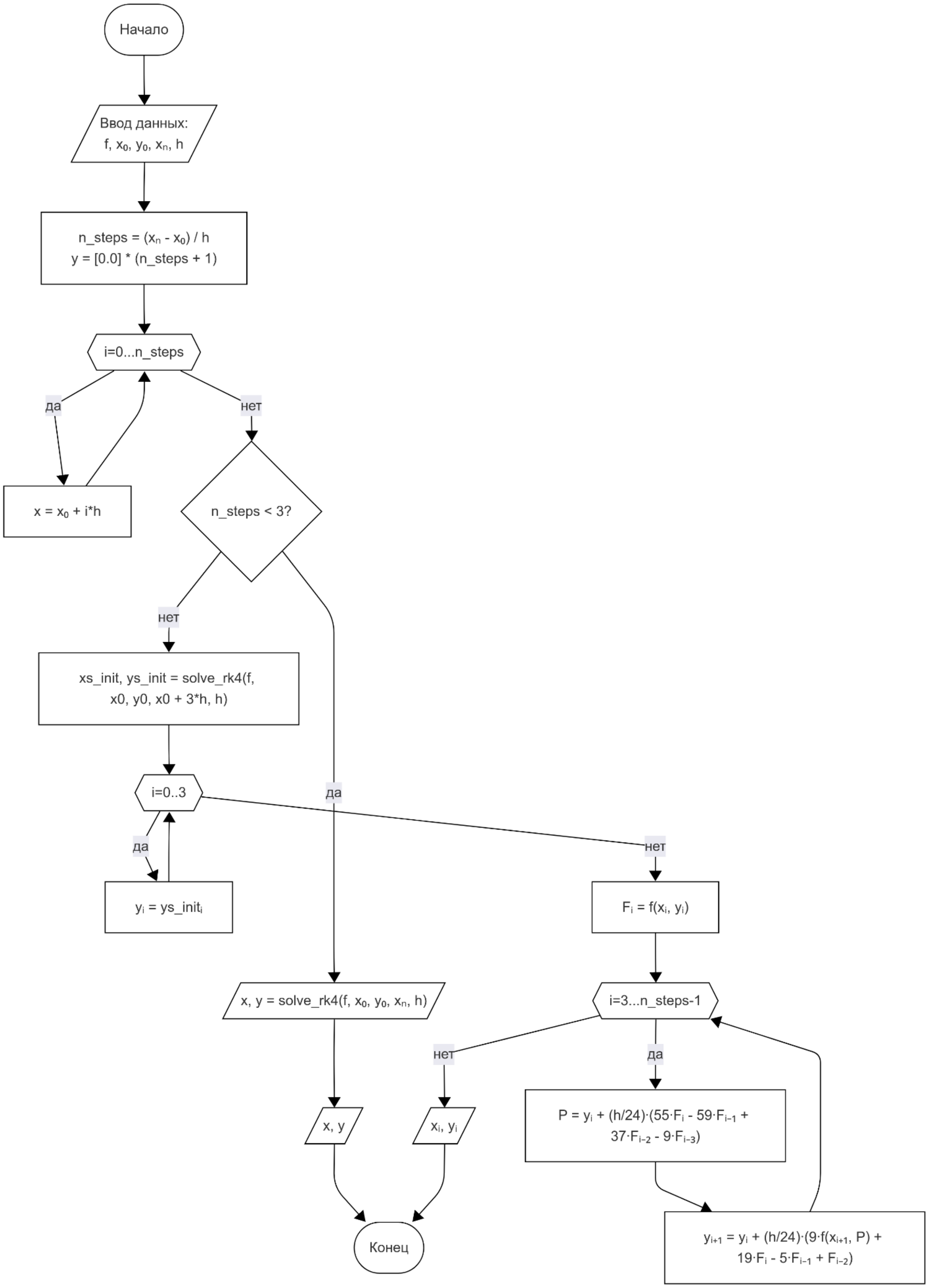
Метод Эйлера:



Метод Рунге-Кутта 4-го порядка:



Многошаговый метод Адамса:



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

Результаты вычислений

Новая задача | Скачать отчёт (TXT)

Выбранная ОДУ

y' = y + (1 + x)y^2

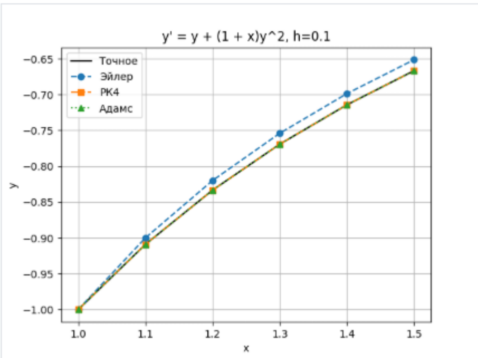
Параметры

x_0 = 1.000000, y_0 = -1.000000, x_n = 1.500000, h = 0.100000, eps = 1e-06

Оценки погрешностей методов

Эйлер	0.0080138646
РК4	0.0000002174
Адамс	0.0000271618

Общий график всех методов



Общая таблица

x	Эйлер	РК4	Адамс	Exact
1.000000	-1.000000	-1.000000	-1.000000	-1.000000

x	Эйлер	РК4	Адамс	Exact
1.000000	-1.000000	-1.000000	-1.000000	-1.000000
1.100000	-0.900000	-0.909093	-0.909093	-0.909091
1.200000	-0.819900	-0.833337	-0.833337	-0.833333
1.300000	-0.753998	-0.769234	-0.769234	-0.769231
1.400000	-0.698640	-0.714289	-0.714267	-0.714286
1.500000	-0.651360	-0.666670	-0.666640	-0.666667

Эйлер РК4 Адамс

Эйлер: график и таблица

Погрешность метода: 0.0080138646

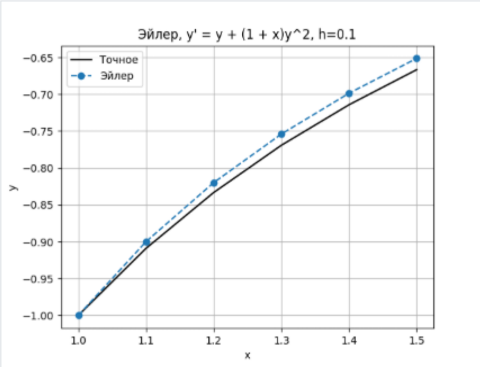


Таблица для Эйлер

x	y_num	y_exact
1.000000	-1.000000	-1.000000
1.100000	-0.900000	-0.909091
1.200000	-0.819900	-0.833333
1.300000	-0.753998	-0.769231
1.400000	-0.698640	-0.714286
1.500000	-0.651360	-0.666667

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.