Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №6

Вариант 10

Студент: Крикунов Олег Евгеньевич

P3267

Преподаватель: Машина Екатерина Алексеевна

Оценка: __________

Подпись преподавателя: __________

1. Цели работы

Изучить численные методы решения однородных дифференциальных уравнений и реализовать их в программе.

2. Описание метода, расчётные формулы

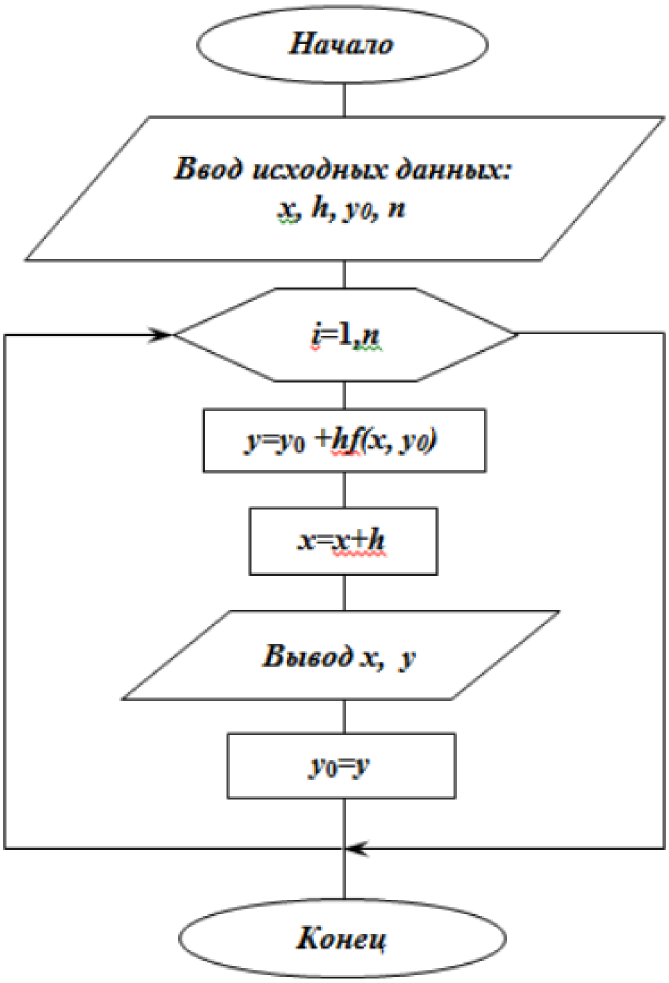


Рис.1 Блок-схема метода Эйлера

$$y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2),$$

$$k_1 = h \cdot f(x_i, y_i) \quad k_2 = h \cdot f(x_i + h, y_i + k_1)$$

Формулы (11), (12) представляют **метод Рунге – Кутты II порядка**

а) этап прогноза:

$$y_i^{прогн} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$$

б) этап коррекции:

$$y_i^{корр} = y_{i-2} + \frac{h}{3}(f_{i-2} + 4f_{i-1} + f_i^{прогн})$$

$$f_i^{прогн} = f(x_i, y_i^{прогн})$$

3. Программная часть

Листинг программы:

Метод Эйлера:

```
#ifndef COMPUTATIONAL_MATHEMATICS_EULERMETHOD_H
#define COMPUTATIONAL_MATHEMATICS_EULERMETHOD_H

typedef std::function<double(double, double)> ODEFunction;

std::vector<double> EulerMethod(ODEFunction f, double x0, double y0, double h, int steps) {
    std::vector<double> y_values(steps + 1);
    y_values[0] = y0;
    double x = x0;
    for (int i = 1; i <= steps; i++) {
        y_values[i] = y_values[i - 1] + h * f(x, y_values[i - 1]);
        x += h;
    }
    return y_values;
}

#endif
```

Метод Рунге-Кутта 4-го порядка:

```
#ifndef COMPUTATIONAL_MATHEMATICS_RUNGEKUTTAMETHOD_H
#define COMPUTATIONAL_MATHEMATICS_RUNGEKUTTAMETHOD_H

std::vector<double> RungeKuttaMethod(ODEFunction f, double x0, double y0, double h, int steps)
{
    std::vector<double> y_values(steps + 1);
    y_values[0] = y0;
    double x = x0;
    for (int i = 1; i <= steps; i++) {
        double k1 = h * f(x, y_values[i - 1]);
        double k2 = h * f(x + h / 2, y_values[i - 1] + k1 / 2);
        double k3 = h * f(x + h / 2, y_values[i - 1] + k2 / 2);
        double k4 = h * f(x + h, y_values[i - 1] + k3);
        y_values[i] = y_values[i - 1] + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        x += h;
    }
    return y_values;
}

#endif
```

Метод Милне:

```
#ifndef COMPUTATIONAL_MATHEMATICS_MILNEMETHOD_H
#define COMPUTATIONAL_MATHEMATICS_MILNEMETHOD_H

std::vector<double> MilneMethod(ODEFunction f, double x0, double y0, double h, int steps) {
    std::vector<double> y_values(steps + 1);
    std::vector<double> predictor(steps + 1);
    std::vector<double> corrector(steps + 1);

    std::vector<double> RungeKutta_values = RungeKuttaMethod(f, x0, y0, h, 3);
    for (int i = 0; i < 4; i++) {
        y_values[i] = RungeKutta_values[i];
    }

    for (int i = 3; i < steps; i++) {
```

```
        double x = x0 + i * h;
        predictor[i + 1] = y_values[i - 3] + (4 * h / 3) * (2 * f(x, y_values[i]) - f(x - h,
y_values[i - 1]) +
                                                                  2 * f(x - 2 * h, y_values[i - 2]));
        corrector[i + 1] = y_values[i - 1] +
                            (h / 3) * (f(x + h, predictor[i + 1]) + 4 * f(x, y_values[i]) + f(x
- h, y_values[i - 1]));
        y_values[i + 1] = corrector[i + 1];
    }
    return y_values;
}


#endif
```

Header-файл с функциями:

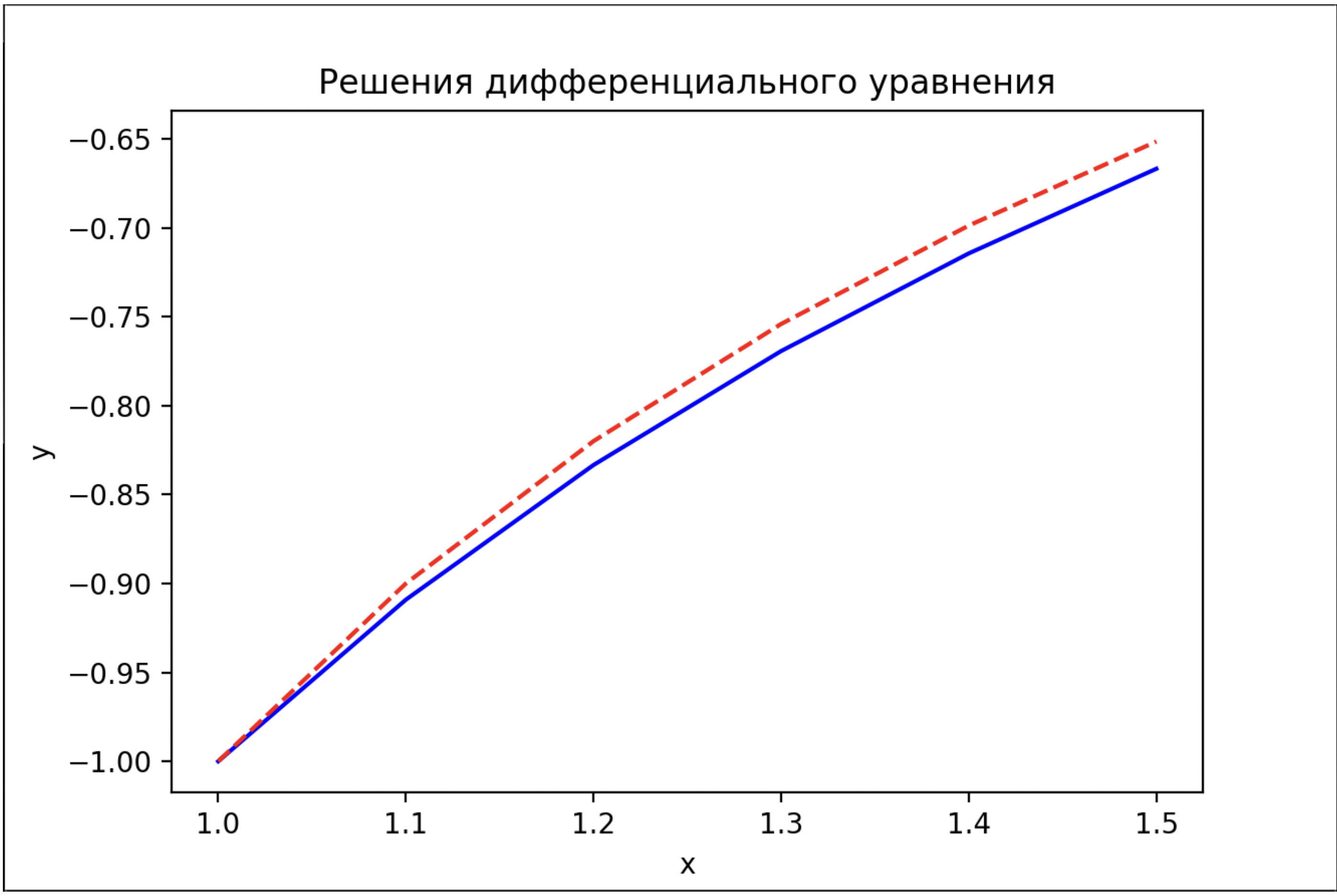
```
#ifndef COMPUTATIONAL_MATHEMATICS_FUNCTIONS_H
#define COMPUTATIONAL_MATHEMATICS_FUNCTIONS_H

double F1(double x, double y) {
    return y + (1 + x) * pow(y, 2);
}

double F2(double x, double y) {
    return y * log(x + 1);
}

double SolutionF1(double x) {
    return -(pow(2.71, x)) / (x * pow(2.71, x));
}

double SolutionF2(double x) {
    return exp((x * (x + 2)) / 2);
}


#endif
```

Правило Рунге:

```
#ifndef COMPUTATIONAL_MATHEMATICS_RUNGERULE_H
#define COMPUTATIONAL_MATHEMATICS_RUNGERULE_H

double RungeRule(const std::vector<double>& y_h, const std::vector<double>& y_half_h) {
    double max_error = 0.0;
    size_t n = y_half_h.size() / 2;
    for (size_t i = 0; i < n; i++) {
        double error = abs(y_h[i] - y_half_h[2 * i]) / 15.0;
        if (error > max_error) {
            max_error = error;
        }
    }
    return max_error;
}

#endif
```

4. Примеры и результаты работы программы

```
Выберите уравнение:
1) y' = y + (1 + x) * y^2
2) y' = y * log(x + 1)
1
Выберите метод решения уравнения
1) Метод Эйлера
2) Метод Рунге-Кутта
3) Метод Милне
1
Введите начальное условие y0: -1
Введите начальное значение x0: 1
Введите конечное значение xN: 1.5
Введите шаг h: 0.1

Метод Эйлера:
x = 1.000000, y = -1.000000
x = 1.100000, y = -0.900000
x = 1.200000, y = -0.819900
x = 1.300000, y = -0.753998
x = 1.400000, y = -0.698640
x = 1.500000, y = -0.651360
Оценка точности методом Рунге для метода Эйлера: 0.000551
```



5. Вывод:

В ходе выполнения работы я познакомился с численными методами решения ОДУ. Реализовали в программе метод Эйлера, Рунге-Кутта 4-го порядка, метод Милне. Построили графики.