Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №6
Вычислительная математика

Вариант: №17

Группа	P3208
Студент	Щетинин С.В.
Преподаватель	Машина Е.А.

Цель работы:

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Программная реализация:

```
#include <iostream>
#include <vector>
#include <cmath>
#include <functional>
#include <iomanip>
#include <fstream>


std::vector<double> generateXs(double x0, double
xn, int n) {
    std::vector<double> xs(n + 1);
    double h = (xn - x0) / n;
    for (int i = 0; i <= n; ++i) {
        xs[i] = x0 + h * i;
    }
    return xs;
}

std::vector<double> eulerMethod(const
std::function<double(double, double)>& f, const
std::vector<double>& xs, double y0) {
    std::vector<double> ys(xs.size());
    ys[0] = y0;
    double h = xs[1] - xs[0];

    for (size_t i = 1; i < xs.size(); ++i) {
        double xi = xs[i - 1];
        double yi = ys[i - 1];
        ys[i] = yi + h * f(xi, yi);
    }
    return ys;
}
```

```
std::vector<double> rungeKuttaMethod(const
std::function<double(double, double)>& f, const
std::vector<double>& xs, double y0) {
    std::vector<double> ys(xs.size());
    ys[0] = y0;
    double h = xs[1] - xs[0];

    for (size_t i = 1; i < xs.size(); ++i) {
        double xi = xs[i - 1];
        double yi = ys[i - 1];
        double k1 = h * f(xi, yi);
        double k2 = h * f(xi + h / 2, yi + k1 /
2);
        double k3 = h * f(xi + h / 2, yi + k2 /
2);
        double k4 = h * f(xi + h, yi + k3);
        ys[i] = yi + (k1 + 2 * k2 + 2 * k3 + k4)
/ 6;
    }
    return ys;
}

std::vector<double> adamsMethod(const
std::function<double(double, double)>& f, const
std::vector<double>& xs, double y0) {
    std::vector<double> ys = rungeKuttaMethod(f,
std::vector<double>(xs.begin(), xs.begin() + 4),
y0);
    double h = xs[1] - xs[0];

    for (size_t i = 4; i < xs.size(); ++i) {
        double xi1 = xs[i - 1];
        double xi2 = xs[i - 2];
        double xi3 = xs[i - 3];
        double xi4 = xs[i - 4];
        double yi1 = ys[i - 1];
        double yi2 = ys[i - 2];
        double yi3 = ys[i - 3];
        double yi4 = ys[i - 4];
```

```
            double f1 = f(xi1, yi1);
            double f2 = f(xi2, yi2);
            double f3 = f(xi3, yi3);
            double f4 = f(xi4, yi4);

            double df = f1 - f2;
            double d2f = f1 - 2 * f2 + f3;
            double d3f = f1 - 3 * f2 + 3 * f3 - f4;

            double y = yi1 + h * f1 + (h * h / 2) *
df + (5 * std::pow(h, 3) / 12) * d2f + (3 *
std::pow(h, 4) / 8) * d3f;
            ys.push_back(y);
        }
        return ys;
}

void write_points(const std::vector <double> &x,
const std::vector <double> &y, std::ofstream &
out) {
        for (auto &p : x) {
            out << p << " ";
        }
        out << "\n";
        for (auto &p : y) {
            out << p << " ";
        }
        out << "\n";
}

void runAndPrintMethod(const std::string&
methodName, const std::function<double(double,
double)>& f, const std::vector<double>& xs,
double y0,
                       const
std::function<double(double, double)>& exactY,
const std::function<std::vector<double>(const
std::function<double(double, double)>&, const
std::vector<double>&, double)>& method,
                       std::ofstream &out) {
```

```
    std::vector<double> ys = method(f, xs, y0);
    write_points(xs, ys, out);
    double maxError = 0.0;

    std::cout << "+--------------------+\n";
    std::cout << "|    x    |    y    |\n";
    std::cout << "+--------------------+\n";
    for (size_t i = 0; i < xs.size(); ++i) {
        double x = xs[i];
        double y = ys[i];
        double exact = exactY(x, 0);
        double error = std::abs(y - exact);
        if (error > maxError) {
            maxError = error;
        }
        std::cout << "| " << std::fixed <<
std::setw(7) << std::setprecision(5) << x << " |
" << std::setw(7) << y << " |\n";
    }
    std::cout << "+--------------------+\n";
    std::cout << methodName << " - max error: "
<< maxError << "\n";
}

int main() {
    std::cout << "1. y' = x\n";
    std::cout << "2. y' = e^x\n";
    std::cout << "3. y' = x^2\n";
    std::cout << "Select function: ";
    int mode;
    std::cin >> mode;

    std::cout << "Input n: ";
    int n;
    std::cin >> n;

    std::cout << "Input x0: ";
    double x0;
    std::cin >> x0;
```

```
    std::cout << "Input xn: ";
    double xn;
    std::cin >> xn;

    auto xs = generateXs(x0, xn, n);

    std::function<double(double, double)> f;
    std::function<double(double, double)> exactY;

    switch (mode) {
        case 1:
            f = [](double x, double y) { return
x; };
            exactY = [](double x, double) {
return x * x / 2 + 1; };
            break;
        case 2:
            f = [](double x, double y) { return
std::exp(x); };
            exactY = [](double x, double) {
return std::exp(x) - 1; };
            break;
        case 3:
            f = [](double x, double y) { return x
* x; };
            exactY = [](double x, double) {
return x * x * x / 3 + 5; };
            break;
        default:
            std::cout << "Invalid input\n";
            return 1;
    }

    double y0 = exactY(x0, 0);
    std::vector <double> ex;

    for (auto x : xs) {
        ex.push_back(exactY(x, 0));
    }
```

```
    std::ofstream out("../script/df");
    runAndPrintMethod("Euler Method", f, xs, y0,
exactY, eulerMethod, out);
    runAndPrintMethod("Runge-Kutta Method", f,
xs, y0, exactY, rungeKuttaMethod, out);
    runAndPrintMethod("Adams Method", f, xs, y0,
exactY, adamsMethod, out);
    write_points(xs, ex, out);

    out.close();

    system(DRAW_GRAPH);

    return 0;
}
```

Пример работы программы:

```
 +------------------+
|   x   |   y   |
+------------------+
| 0.00000 | 1.00000 |
| 1.00000 | 1.00000 |
| 2.00000 | 2.00000 |
| 3.00000 | 4.00000 |
| 4.00000 | 7.00000 |
| 5.00000 | 11.00000 |
+------------------+
Euler Method - max error: 2.50000
+------------------+
|   x   |   y   |
+------------------+
| 0.00000 | 1.00000 |
| 1.00000 | 1.50000 |
| 2.00000 | 3.00000 |
| 3.00000 | 5.50000 |
| 4.00000 | 9.00000 |
| 5.00000 | 13.50000 |
+------------------+
Runge-Kutta Method - max error: 0.00000
+------------------+
|   x   |   y   |
```

```
+------------------+
| 0.00000 | 1.00000 |
| 1.00000 | 1.50000 |
| 2.00000 | 3.00000 |
| 3.00000 | 5.50000 |
| 4.00000 | 9.00000 |
| 5.00000 | 13.50000 |
+------------------+
```
Adams Method - max error: 0.00000