**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

**РЕШЕНИЕ ОДУ**

**(Вариант 10)**

*Выполнил студент 3 курса МОиАИС*

*Соколов Арсений*

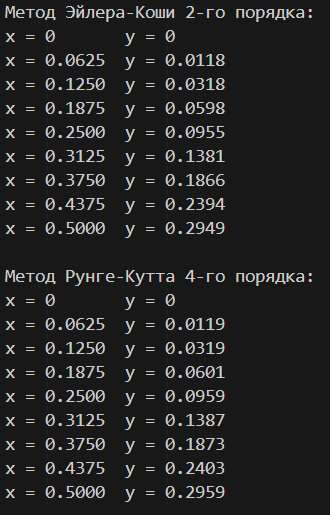
**Задание 1:**

Решить уравнение 1 методом Эйлера-Коши 2-го порядка точности и методом Рунге-Кутта 4-го порядка точности.

**Дано:**

y = sin(2x + y) - 0,3y; y(*a*) = 0; a = 0; b = 0,5;

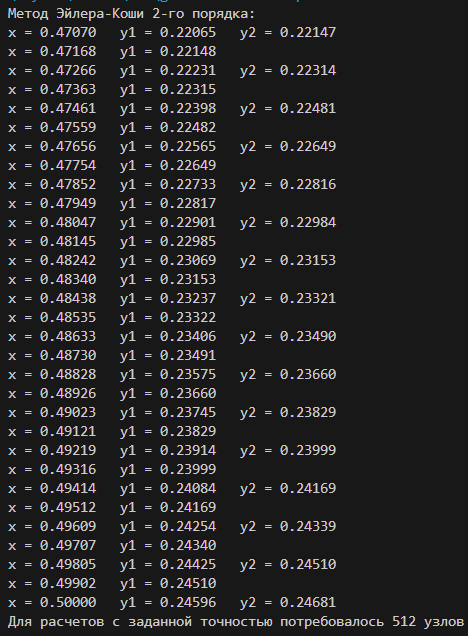
**Решение:**

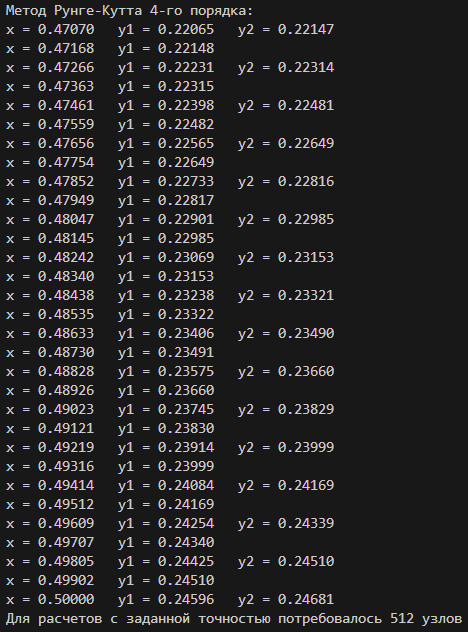
****

(см. приложение 1)

**Это метод двойного пересчета?**

Исправлено:





(добавлена проверка точности методом двойного перерасчета см код в Приложении 1)

**Задание 2:**

Решить уравнение 2 (ДУ второго порядка) методами Адамса 3-го порядка точности и 4-го порядка точности.

Точность вычислений контролировать методом двойного пересчета.

**Дано:**

y” = cos(1,5x + y); y(*a*) = 0; y(*a*) = 1; a = 0; b = 0,5

**Решение:**

Приведем ОДУ 2-го порядка к ОДУ 1-го порядка

Выполним замену yz, z = f(x, y)

Имеем приближения y0 , z0 , методом Эйлера-Коши найдем y1 , y2  и методом Рунге-Кутта z1 , z2 (см. приложение 2), тогда:

**Все расписать подробно для первых четырех точек**

Исправлено:

**y0**= 0

**z0**= y0= 1

x0 = 0

Методом Эйлера-Коши найдем y1:

**y1**​ = y0​ + h z0

Методом Рунге-Кутта найдем z1:

x1 = a + h

k1 = h *f*(x1, y0),

k2 = h *f*(x1 + h/2, y0 + k1/2)

k3 = h *f*(x1 + h/2, y0 + k2/2)

k4 = h *f*(x1 + h, y0 + k3)

**z1**​= z0 + 1/6 \* (k1​ + 2k2​ + 2k3​ + k4​​)

Методом Эйлера-Коши найдем y2:

**y2**​ = y1​ + h z1

Методом Рунге-Кутта найдем z2:

x2 = a + 2 h

k1 = h *f*(x2, y0)

k2 = h *f*(x2 + h/2, y0 + k1/2)

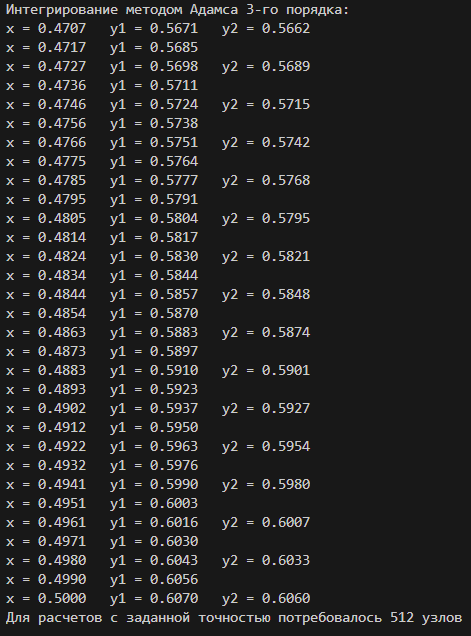
k3 = h *f*(x2 + h/2, y0 + k2/2)

k4 = h *f*(x2 + h, y0 + k3)

**z2​**= z1 + 1/6 (k1​ + 2k2​ + 2k3​ + k4​​)

(добавлено описание нахождения первых 4 точек)

**Интегрирование методом Адамса 3-го порядка:**



Аналогичным образом найдем y3  и z3:

**y3**​ = y2​ + h z2

x3 = a + 3 h

k1 = h *f*(x3, y0)

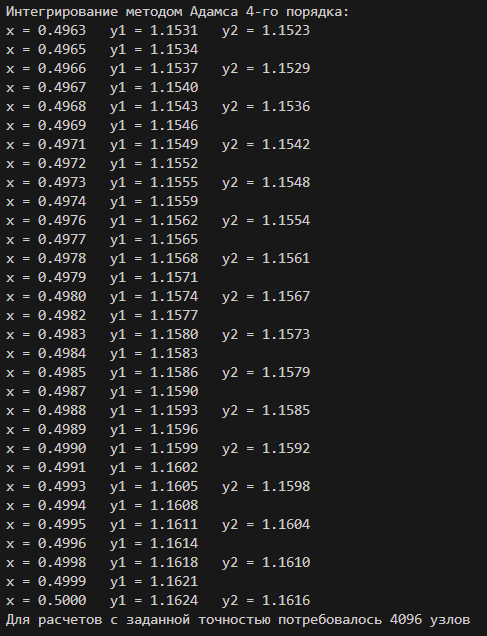
k2 = h *f*(x3 + h/2, y0 + k1/2)

k3 = h *f*(x3 h + h/2, y0 + k2/2)

k4 = h *f*(x3 h + h, y0 + k3)

**z3​**= z2 + 1/6 (k1​ + 2k2​ + 2k3​ + k4​​)

(добавлено описание нахождения 5-ой и 6-ой точек)

**Интегрирование методом Адамса 4-го порядка:**

**ПРИЛОЖЕНИЕ 1**

***Программа вычисления значений для первого задания***

#include <iostream>

#include <iomanip>

#include <cmath>

#include <map>

// Определение правой части уравнения y' = sin(2x + y) - 0.3y

double f(double x, double y) {

    return sin(2 \* x + y) - 0.3 \* y;

}

// Метод Эйлера-Коши 2-го порядка точности

std::map<double, double> EulerCauchy(double a, double b, int n, double y0) {

    double h = (b - a) / n; // Шаг интегрирования

    double x = a;

    double y = y0;

    std::map<double, double> EC;

    while (x < b) {

        x += h;

        double y\_temp = y + h \* f(x, y);

        y = y + (h \* (f(x, y) + f(x + h, y\_temp))/ 2);

        EC[x] = y;

    }

    return EC;

}

// Метод Рунге-Кутта 4-го порядка точности

std::map<double, double> RungeKutta(double a, double b, int n, double y0) {

    double h = (b - a) / n; // Шаг интегрирования

    double x = a;

    double y = y0;

    std::map<double, double> RK;

    while (x < b) {

        x += h;

        double k1 = h \* f(x, y);

        double k2 = h \* f(x + h / 2, y + k1 / 2);

        double k3 = h \* f(x + h / 2, y + k2 / 2);

        double k4 = h \* f(x + h, y + k3);

        y = y + (k1 + 2\*k2 + 2\*k3 + k4) / 6;

        RK[x] = y;

    }

    return RK;

}

bool checkAccuracy(std::map<double, double>& y1, std::map<double, double>& y2, double eps) {

    for (const auto& [x, y] : y1) {

        if (y2.find(x) != y2.end()) {

            if (fabs(y - y2[x]) >= eps) {

                return false;

            }

        }

    }

    return true;

}

int main() {

    double a = 0;

    double b = 0.5;

    int n = 2;

    double y0 = 0; // Начальное условие y(0) = 0

    double eps = 0.001; // Точность

    std::map<double, double> notChecked;

    std::map<double, double> checked;

    std::cout << std::fixed << std::setprecision(5) << std::showpoint;

    // std::cout << "Метод Эйлера-Коши 2-го порядка:" << std::endl;

    // notChecked = EulerCauchy(a, b, n, y0);

    // do {

    //     checked = notChecked;

    //     n \*= 2;

    //     notChecked = EulerCauchy(a, b, n, y0);

    // } while (!checkAccuracy(checked, notChecked, eps));

    std::cout << "Метод Рунге-Кутта 4-го порядка:" << std::endl;

    notChecked = RungeKutta(a, b, n, y0);

    do {

        checked = notChecked;

        n \*= 2;

        notChecked = RungeKutta(a, b, n, y0);

    } while (!checkAccuracy(checked, notChecked, eps));

    int count = 0;

    for (const auto& [x, y] : notChecked) {

        if (count > n - 32) {

            std::cout << "x = " << x << "   y1 = " << y;

            if (checked[x]){

                std::cout << "   y2 = " << checked[x] << std::endl;

            }

            else {

                std::cout << std::endl;

            }

        }

        count++;

    }

    std::cout << "Для расчетов с заданной точностью потребовалось " << count << " узлов" << std::endl;

    std::cout << std::endl;

    return 0;

}

**ПРИЛОЖЕНИЕ 2**

***Программа вычисления значений для второго задания***

#include <iostream>

#include <iomanip>

#include <cmath>

#include <map>

double f(double x, double y) {

    return cos(1.5 \* x + y);

}

double RungeKutta(double x, double y, double y0, double h) {

    double k1 = h \* f(x, y);

    double k2 = h \* f(x + h / 2, y + k1 / 2);

    double k3 = h \* f(x + h / 2, y + k2 / 2);

    double k4 = h \* f(x + h, y + k3);

    return y0 + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);

}

// Метод Адамса 3-го порядка точности

std::map<double, double> thirdOrderAdams(double a, double b, int n, double y0, double z0) {

    double h = (b - a) / n; // Шаг интегрирования

    std::map<double, double> Y;

    std::map<double, double> Z;

    Y[a] = y0;

    Z[a] = z0;

    Z[a + h] = RungeKutta(a, Y[a], Z[a], h);

    Y[a + h] = Y[a] + h \* Z[a];

    Z[a + 2 \* h] = RungeKutta(a + h, Y[a + h], Z[a + h], h);

    Y[a + 2 \* h] = Y[a + h] + h \* Z[a + h];

    for (int i = 3; i <= n; i++) {

        Z[a + i \* h] = Z[a + (i - 1) \* h] + h \* (23 \* f(a + (i - 1) \* h, Y[a + (i - 1) \* h]) - 16 \* f(a + (i - 2) \* h, Y[a + (i - 2) \* h]) + 5 \* f(a + (i - 3) \* h, Y[a + (i - 3) \* h])) / 12;

        Y[a + i \* h] = Y[a + (i - 1) \* h] + h \* (23 \* Z[a + (i - 1) \* h] - 16 \* Z[a + (i - 2) \* h] + 5 \* Z[a + (i - 3) \* h]) / 12;

    }

    return Y;

}

// Метод Адамса 4-го порядка точности

std::map<double, double> forthOrderAdams(double a, double b, int n, double y0, double z0) {

    double h = (b - a) / n; // Шаг интегрирования

    std::map<double, double> Y;

    std::map<double, double> Z;

    Y[a] = y0;

    Z[a] = z0;

    Z[a + h] = RungeKutta(a, Y[a], Z[a], h);

    Y[a + h] = Y[a] + h \* Z[a];

    Z[a + 2 \* h] = RungeKutta(a + h, Y[a + h], Z[a + h], h);

    Y[a + 2 \* h] = Y[a + h] + h \* Z[a + h];

    Z[a + 3 \* h] = RungeKutta(a + 2 \* h, Y[a + 2 \* h], Z[a + 2 \* h], h);

    Y[a + 3 \* h] = Y[a + 2 \* h] + h \* Z[a + 2 \* h];

    for (int i = 4; i <= n; i++) {

        Z[a + i \* h] = Z[a + (i - 1) \* h] + h \* (55 \* f(a + (i - 1) \* h, Y[a + (i - 1) \* h]) - 59 \* f(a + (i - 2) \* h, Y[a + (i - 2) \* h]) + 37 \* f(a + (i - 3) \* h, Y[a + (i - 3) \* h]) + 9 \* f(a + (i - 4) \* h, Y[a + (i - 4) \* h])) / 24;

        Y[a + i \* h] = Y[a + (i - 1) \* h] + h \* (55 \* Z[a + (i - 1) \* h] - 59 \* Z[a + (i - 2) \* h] + 37 \* Z[a + (i - 3) \* h] + 9 \* Z[a + (i - 4) \* h]) / 24;

    }

    return Y;

}

bool checkAccuracy(std::map<double, double>& y1, std::map<double, double>& y2, double eps) {

    for (const auto& [x, y] : y1) {

        if (y2.find(x) != y2.end()) {

            if (fabs(y - y2[x]) >= eps) {

                return false;

            }

        }

    }

    return true;

}

int main() {

    double a = 0;

    double b = 0.5;

    double y0 = 0; // Начальное условие y(0) = 0

    double z0 = 1; // Начальное условие y'(0) = z = 1

    double eps = 0.001; // Точность

    std::map<double, double> notChecked;

    std::map<double, double> checked;

    std::cout << std::fixed << std::setprecision(4) << std::showpoint;

    int n = 16;

    std::cout << "Интегрирование методом Адамса 3-го порядка:" << std::endl;

    notChecked = thirdOrderAdams(a, b, n, y0, z0);

    do {

        checked = notChecked;

        n \*= 2;

        notChecked = thirdOrderAdams(a, b, n, y0, z0);

    } while (!checkAccuracy(checked, notChecked, eps));

    // std::cout << "Интегрирование методом Адамса 4-го порядка:" << std::endl;

    // notChecked = forthOrderAdams(a, b, n, y0, z0);

    // do {

    //     checked = notChecked;

    //     n \*= 2;

    //     notChecked = forthOrderAdams(a, b, n, y0, z0);

    // } while (!checkAccuracy(checked, notChecked, eps));

    int count = -1;

    for (const auto& [x, y] : notChecked) {

        if (count > n - 32) {

            std::cout << "x = " << x << "   y1 = " << y;

            if (checked[x]){

                std::cout << "   y2 = " << checked[x] << std::endl;

            }

            else {

                std::cout << std::endl;

            }

        }

        count++;

    }

    std::cout << "Для расчетов с заданной точностью потребовалось " << count << " узлов" << std::endl;

    std::cout << std::endl;

    return 0;

}