**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт: №8 «Информационные технологии   
и прикладная математика»   
Кафедра: 806 «Вычислительная математика   
и программирование»**

Лабораторная работа № 4   
по курсу «Численные методы»

Группа: М8О-308Б-21

Студентка: Шевлякова С. С.

Преподаватель: Ревизников Д. Л.

Дата: 22.05.2024

Москва, 2024

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc158983147)

[2 Задание 3](#_Toc158983148)

[3 Теория 4](#_Toc158983149)

[4 Ход лабораторной работы 5](#_Toc158983150)

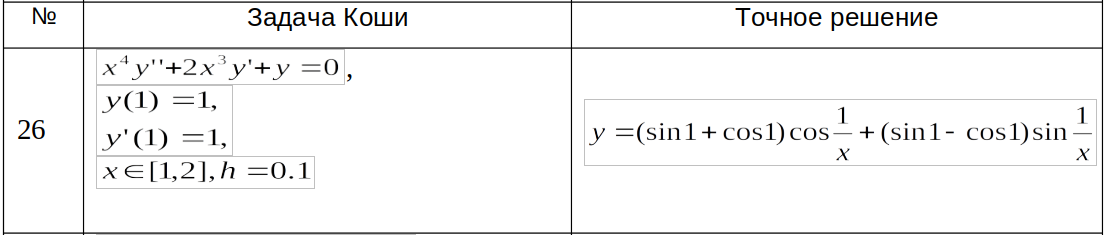
[5 Выводы 6](#_Toc158983151)

# **Тема**

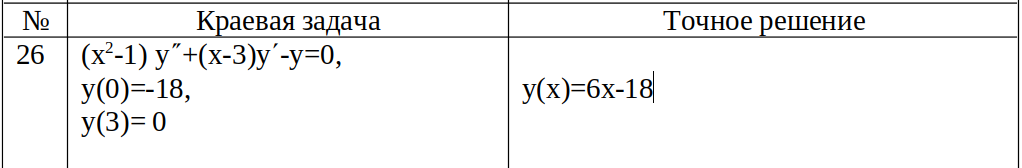
Методы решения начальных и краевых задач для обыкновенных дифференциальных уравнений (ОДУ) и систем ОДУ

# **Задание**

4.1. Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки h . С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

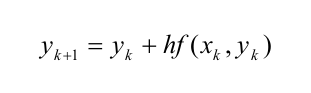


4.2. Реализовать метод стрельбы и конечно-разностный метод решения краевой задачи для ОДУ в виде программ. С использованием разработанного программного обеспечения решить краевую задачу для обыкновенного дифференциального уравнения 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

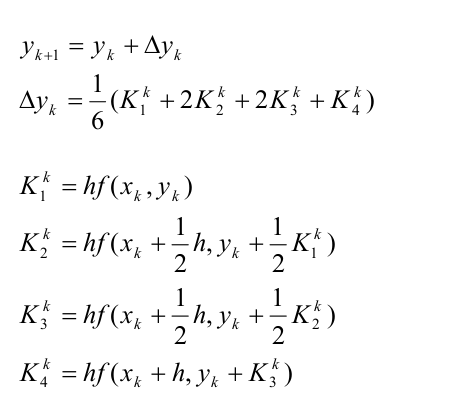


# **Теория**

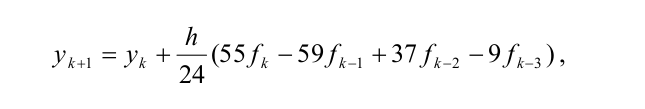
**Метод Эйлера** играет важную роль в теории численных методов решения ОДУ, хотя и не часто используется в практических расчетах из-за невысокой точности. Вывод расчетных соотношений для этого метода может быть произведен несколькими способами: с помощью геометрической интерпретации, с использованием разложения в ряд Тейлора, конечно разностным методом (с помощью разностной аппроксимации производной), квадратурным способом (использованием эквивалентного интегрального уравнения).



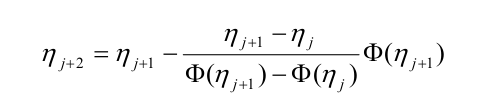
Все рассмотренные выше явные методы являются вариантами методов **Рунге-Кутты.** Семейство явных методов Рунге-Кутты 4-го порядка записывается в виде совокупности формул:



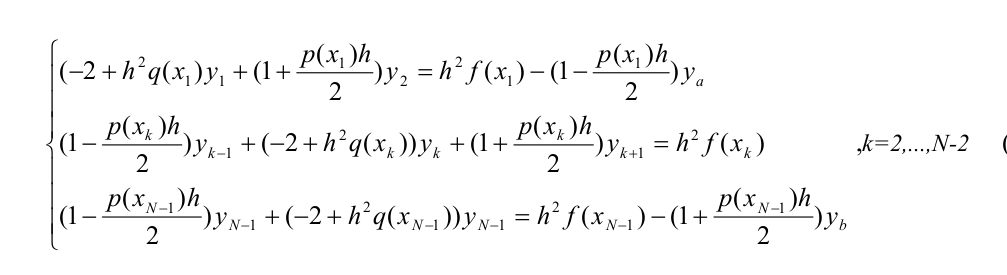
При использовании интерполяционного многочлена 3-ей степени построенного по значениям подынтегральной функции в последних четырех узлах получим **метод Адамса** четвертого порядка точности:



Суть **метода стрельбы** заключена в многократном решении задачи Коши для приближенного нахождения решения краевой задачи.



**Конечно-разностный метод** решения краевой задачи получим систему линейных алгебраических уравнений с трехдиагональной матрицей коэффициентов, которую можно решить с помощью метода прогонки:



# **Х****од лабораторной работы**

Код был реализован на языке C++

4.1) Методы Эйлера, Рунге-Кутты и Адамса 4-го порядка:

**pair<vector<double>, vector<double>> method\_euler\_recalculation(double x\_start, double x\_finish, double h, double y0, double z0) {**

**int n = (x\_finish - x\_start) / h;**

**vector<double> X(n + 1), Y(n + 1), Y\_tmp(n + 1), Z(n + 1), Z\_tmp(n + 1);**

**for (int i = 0; i <= n; ++i) {**

**X[i] = x\_start + i \* h;**

**}**

**Y[0] = y0;**

**Z[0] = z0;**

**for (int i = 1; i <= n; ++i) {**

**vector<double> F\_tmp = f(X[i - 1], Y[i - 1], Z[i - 1]);**

**Y\_tmp[i] = Y[i - 1] + h \* F\_tmp[0];**

**Z\_tmp[i] = Z[i - 1] + h \* F\_tmp[1];**

**vector<double> F = f(X[i], Y\_tmp[i], Z\_tmp[i]);**

**Y[i] = Y[i - 1] + h \* (F\_tmp[0] + F[0]) / 2;**

**Z[i] = Z[i - 1] + h \* (F\_tmp[1] + F[1]) / 2;**

**}**

**return {X, Y};**

**}**

**vector<vector<double>> method\_runge\_kutta\_4(vector<double> (\*f)(double, double, double), double x\_start, double x\_finish, double h, double y0, double z0, int iter = -1) {**

**int n;**

**if (iter == -1) {**

**n = (x\_finish - x\_start) / h;**

**} else**

**n = iter;**

**vector<double> X(n + 1), Y(n + 1), Z(n + 1);**

**for (int i = 0; i <= n; ++i) {**

**X[i] = x\_start + i \* h;**

**}**

**Y[0] = y0;**

**Z[0] = z0;**

**for (int i = 1; i <= n; ++i) {**

**vector<double> K\_1 = num\_vector(f(X[i - 1], Y[i - 1], Z[i - 1]), h);**

**vector<double> K\_2 = num\_vector(f(X[i - 1] + h / 2, Y[i - 1] + K\_1[0] / 2, Z[i - 1] + K\_1[1] / 2), h);**

**vector<double> K\_3 = num\_vector(f(X[i - 1] + h / 2, Y[i - 1] + K\_2[0] / 2, Z[i - 1] + K\_2[1] / 2), h);**

**vector<double> K\_4 = num\_vector(f(X[i - 1] + h, Y[i - 1] + K\_3[0], Z[i - 1] + K\_3[1]), h);**

**Y[i] = Y[i - 1] + 1.0 / 6 \* (K\_1[0] + 2 \* K\_2[0] + 2 \* K\_3[0] + K\_4[0]);**

**Z[i] = Z[i - 1] + 1.0 / 6 \* (K\_1[1] + 2 \* K\_2[1] + 2 \* K\_3[1] + K\_4[1]);**

**}**

**return {X, Y, Z};**

**}**

**pair<vector<double>, vector<double>> method\_adams(double x\_start, double x\_finish, double h, double y0, double z0) {**

**int n = (x\_finish - x\_start) / h;**

**vector<double> X(n + 1), Y(n + 1), Z(n + 1);**

**for (int i = 0; i <= n; ++i) {**

**X[i] = x\_start + i \* h;**

**}**

**vector<vector<double>> res\_runge\_kutta = method\_runge\_kutta\_4(f, x\_start, x\_finish, h, y0, z0, 4);**

**Y[0] = res\_runge\_kutta[1][0];**

**Y[1] = res\_runge\_kutta[1][1];**

**Y[2] = res\_runge\_kutta[1][2];**

**Y[3] = res\_runge\_kutta[1][3];**

**Z[0] = res\_runge\_kutta[2][0];**

**Z[1] = res\_runge\_kutta[2][1];**

**Z[2] = res\_runge\_kutta[2][2];**

**Z[3] = res\_runge\_kutta[2][3];**

**for (int i = 4; i <= n; ++i) {**

**vector<double> F\_k = f(X[i - 1], Y[i - 1], Z[i - 1]);**

**vector<double> F\_k\_1 = f(X[i - 2], Y[i - 2], Z[i - 2]);**

**vector<double> F\_k\_2 = f(X[i - 3], Y[i - 3], Z[i - 3]);**

**vector<double> F\_k\_3 = f(X[i - 4], Y[i - 4], Z[i - 4]);**

**Y[i] = Y[i - 1] + h / 24 \* (55 \* F\_k[0] - 59 \* F\_k\_1[0] + 37 \* F\_k\_2[0] - 9 \* F\_k\_3[0]);**

**Z[i] = Z[i - 1] + h / 24 \* (55 \* F\_k[1] - 59 \* F\_k\_1[1] + 37 \* F\_k\_2[1] - 9 \* F\_k\_3[1]);**

**}**

**return {X, Y};**

**}**

4.2) Метод стрельбы, конечно-разностный метод:

**vector<vector<double>> shooting(double x\_start, double x\_finish, double h, double y0, double y1, double n0, double n1, double eps) {**

**double ni = n0, ni\_1 = n1;**

**double phi\_ni = phi(x\_start, x\_finish, h, y0, ni, y1);**

**double phi\_ni\_1 = phi(x\_start, x\_finish, h, y0, ni\_1, y1);**

**while (abs(phi\_ni\_1) > eps) {**

**double ni\_2 = ni\_1 - (ni\_1 - ni) / (phi\_ni\_1 - phi\_ni) \* phi\_ni\_1;**

**ni = ni\_1;**

**ni\_1 = ni\_2;**

**phi\_ni = phi\_ni\_1;**

**phi\_ni\_1 = phi(x\_start, x\_finish, h, y0, ni\_1, y1);**

**}**

**return method\_runge\_kutta\_4(f, x\_start, x\_finish, h, y0, ni\_1);**

**}**

**vector<vector<double>> finite\_difference(double x\_start, double x\_finish, double y0, double y1, double h) {**

**int n = (x\_finish - x\_start) / h + 1;**

**vector<double> X(n);**

**for (int i = 0; i < n; ++i) {**

**X[i] = x\_start + i \* h;**

**}**

**Matrix A(n, n), B(n, 1);**

**A(0, 0) = h;**

**A(0, 1) = 0;**

**for (int i = 1; i < n - 1; ++i) {**

**A(i, i - 1) = 1 - p(X[i]) \* h / 2;**

**A(i, i) = -2 + h \* h \* q(X[i]);**

**A(i, i + 1) = 1 + p(X[i]) \* h / 2;**

**}**

**A(n - 1, n - 2) = 0;**

**A(n - 1, n - 1) = h;**

**B(0, 0) = h \* y0;**

**for (int i = 1; i < n - 1; ++i) {**

**B(i, 0) = h \* h \* f2(X[i]);**

**}**

**B(n - 1, 0) = h \* y1;**

**Matrix C = A.run\_through\_method(B);**

**vector<double> Y;**

**for (int i = 0; i < C.GetRows(); ++i) {**

**Y.push\_back(C(i, 0));**

**}**

**return {X, Y};**

**}**

# **Выводы**

В ходе лабораторной работы были реализованы методы Эйлера (явный и модифицированный), метод Адамса 4-го порядка, Рунге-Кутта 4-го порядка, метод стрельбы и конечно-разностный метод. Результаты работы алгоритмов оценены с помощью сравнения с точным решением, а также методом Рунге-Ромберга.