Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики

**Дисциплина «Вычислительная математика»**

**Лабораторная работа №5**

**Решение краевой задачи методом пристрелки**

Выполнил:

Съестов Дмитрий Вячеславович

Группа Р3217

Преподаватель:

Калёнова Ольга Вячеславовна

Санкт-Петербург, 2017 г.

**Описание метода**

Метод стрельбы не решает краевую задачу для уравнения Эйлера в исходной постановке, а сводит ее к последовательности более простых задач, а именно — задач Коши с особым образом сформулированными начальными условиями.

Рассмотрим краевую задачу вида с краевыми условиями

Также возьмём два произвольных значения пристрелочного параметра, равного производной искомой функции на левом конце отрезка: .

Чтобы решить ОДУ второго порядка численным методом, приведём его к системе двух ОДУ первого порядка:

Используя численный метод, найдём значения функции на правом конце отрезка при значениях пристрелочного параметра z1 и z2.

Затем найдём уравнение линейной зависимости и найдём значение z, соответствующее заданному значению y(b).

Наконец, используя численный метод, решим полученную систему из двух задач Коши с условием

**Листинг программы**

using System;

using AssertLibrary;

using UnaryFunc = System.Func<double, double>;

using TernaryFunc = System.Func<double, double, double, double>;

namespace vychmat5

{

internal static class ShootingMethod

{

public const int STEP\_COUNT = 3;

//Возвращает уравнение прямой, проходящей через две точки

private static UnaryFunc getLineEquation(double x1, double y1, double x2, double y2)

{

if (y1 == y2) return x => y1;

double linearFunc(double x) => (x - x1) / (x2 - x1) \* (y2 - y1) + y1;

Assert.IsEqual(y1, linearFunc(x1));

Assert.IsEqual(y2, linearFunc(x2));

return linearFunc;

}

//Возвращает значение Y на правой границе отрезка

private static double getLastY(TernaryFunc f, double x0, double y0, double z0, double xn, double h)

{

Assert.IsNotNull(f);

int stepCount = (int)((xn - x0) / h);

double xi = x0, yi = y0, zi = z0;

for (int i = 1; i <= stepCount; i++)

{

double deltaY = h \* zi;

zi += h \* f(xi, yi, zi);

yi += deltaY;

xi += h;

}

return yi;

}

public static UnaryFunc Solve(TernaryFunc f, double a, double b, double ya, double yb, double z1, double z2, double precision)

{

if (f == null) throw new ArgumentNullException("Функция не определена");

if (b <= a) throw new ArgumentException("b должно быть больше a");

double step = (b - a) / STEP\_COUNT;

//Находим Y в точке b при значениях пристрелочного параметра Z1 и Z2

double try1 = getLastY(f, a, ya, z1, b, step);

double try2 = getLastY(f, a, ya, z2, b, step);

//Находим линейную зависимость Z = f( Y(b) )

UnaryFunc lineFunc = getLineEquation(try1, z1, try2, z2);

//Находим такое значение Z, что Y(b) = Yb

double z = lineFunc(yb);

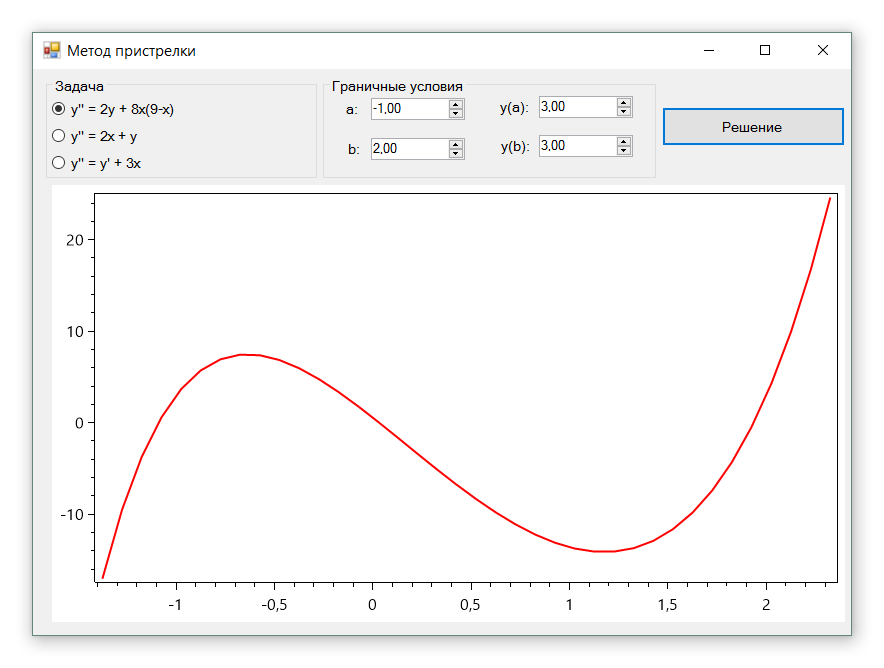
return EulerMethod.Solve(f, a, ya, z, b, precision);

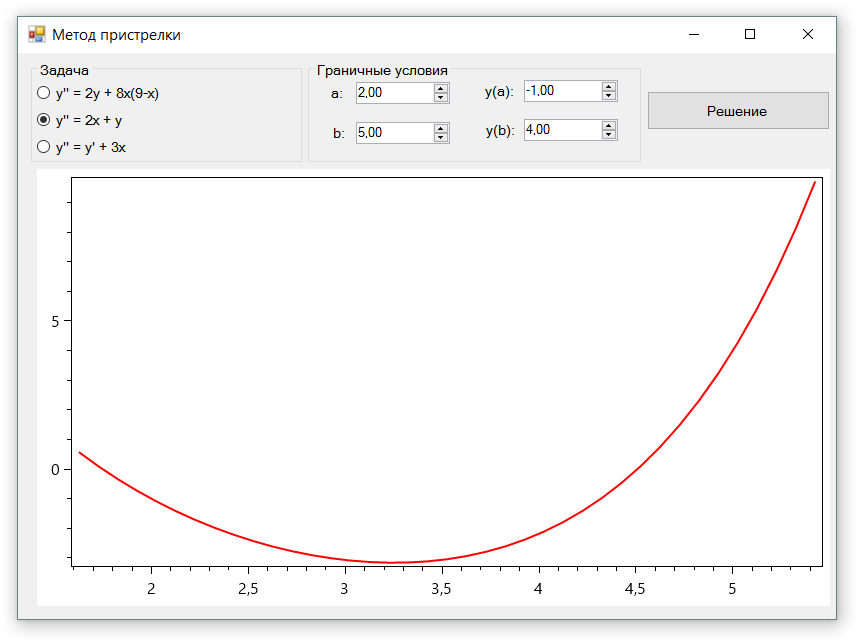
}

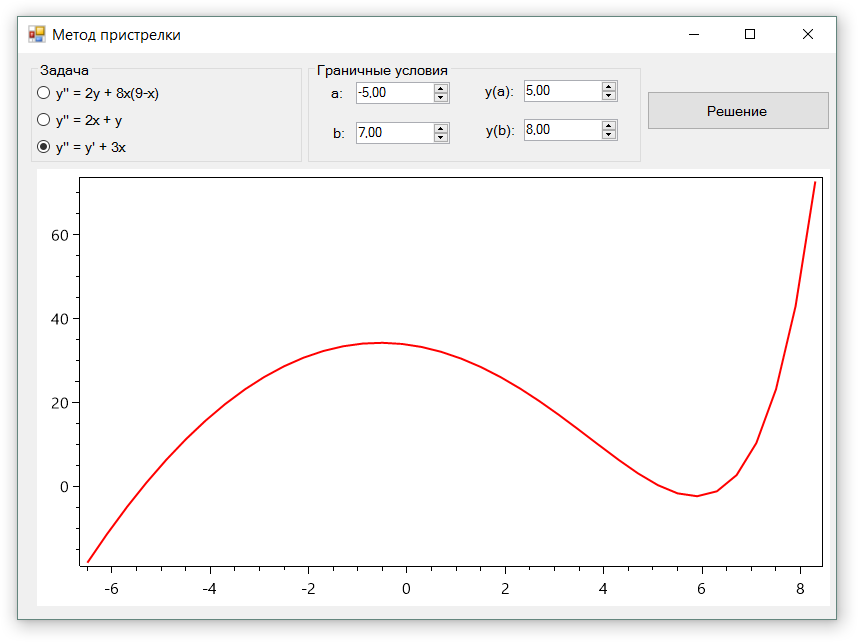
}

}

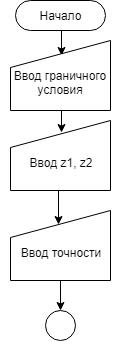
**Тестовые данные**

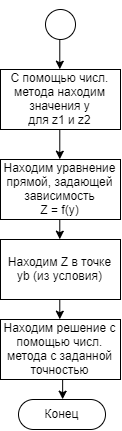
****

****

****

**Блок-схема**

****

****

**Вывод**

Метод пристрелки не даёт решения задачи сам по себе, поэтому его точность зависит от численного метода, используемого для решения задачи Коши. Однако он даёт меньшую точность, чем метод параллельной стрельбы.

Как и большинство численных методов решения краевых задач, метод пристрелки разработан для уравнений второго порядка.