Университет ИТМО

Факультет: ПИиКТ

Дисциплина: Вычислительная математика

Лабораторная работа №5

Вариант “2”

Выполнили: **Кудлаков Роман**

Группа: **P3231**

Преподаватель: **Перл Ольга Вячеславовна**

Санкт-Петербург, 2021

Описание метода

Метод прогонки – метод для решения дифференциальных уравнений второго порядка. С его помощью можно найти решение на отрезке [a, b] неоднородоного обыкновенного дифференциального уравнения второго порядка вида

$$A(x)y'' + p(x)y' - q(x)y = f(x)$$

Которое удовлетворяет граничным условиям

$$\alpha_0 y(a) + \beta_0 y'(a) = A,$$
$$\alpha_1 y(b) + \beta_1 y'(b) = B,$$
$$\text{где } |\alpha_0| + |\beta_0| \neq 0,$$
$$|\alpha_1| + |\beta_1| \neq 0$$

Алгоритм решения

1. Вводятся дифференциальное уравнение, которое надо решить, и граничные условия

2. Формируется система ДУ первого порядка для прямого хода и ДУ первого порядка для обратного хода по следующим правилам:

a) Если $\beta_0 \neq 0$

Прямой ход:

$$\begin{cases} Z_1' = -Z_1^2 - p(x)Z_1 + q(x), \qquad Z_1(a) = -\dfrac{\alpha_0}{\beta_0} \\ Z_2' = -Z_2 * [Z_1 + p(x)] + f(x), \qquad Z_2(a) = \dfrac{A}{\beta_0} \end{cases}$$
$$a \le x \le b$$

Обратный ход:

$$y' = Z_1(x)y + Z_2(x), \qquad a \le x \le b; \qquad y(b) = \frac{B - \beta_1 Z_2(b)}{\alpha_1 + \beta_1 Z_1(b)}$$

b) Если $\alpha_0 \neq 0$

Прямой ход:

$$\begin{cases} Z_1' = -Z_1^2 q(x) + Z_1 p(x) + 1, \qquad Z_1(a) = -\dfrac{\beta_0}{\alpha_0} \\ Z_2' = -Z_1(x) * [Z_2 q(x) + f(x)], \qquad Z_2(a) = -\dfrac{A}{\alpha_0} \end{cases}$$
$$a \le x \le b$$

Обратный ход:

$$Z_1(x)y' = y - Z_2(x), \qquad a \le x \le b; \qquad y(b) = \frac{BZ_1(b) + \beta_1 Z_2(b)}{\beta_1 + \alpha_1 Z_1(b)}$$

3. Решается задача Коши в прямом ходе

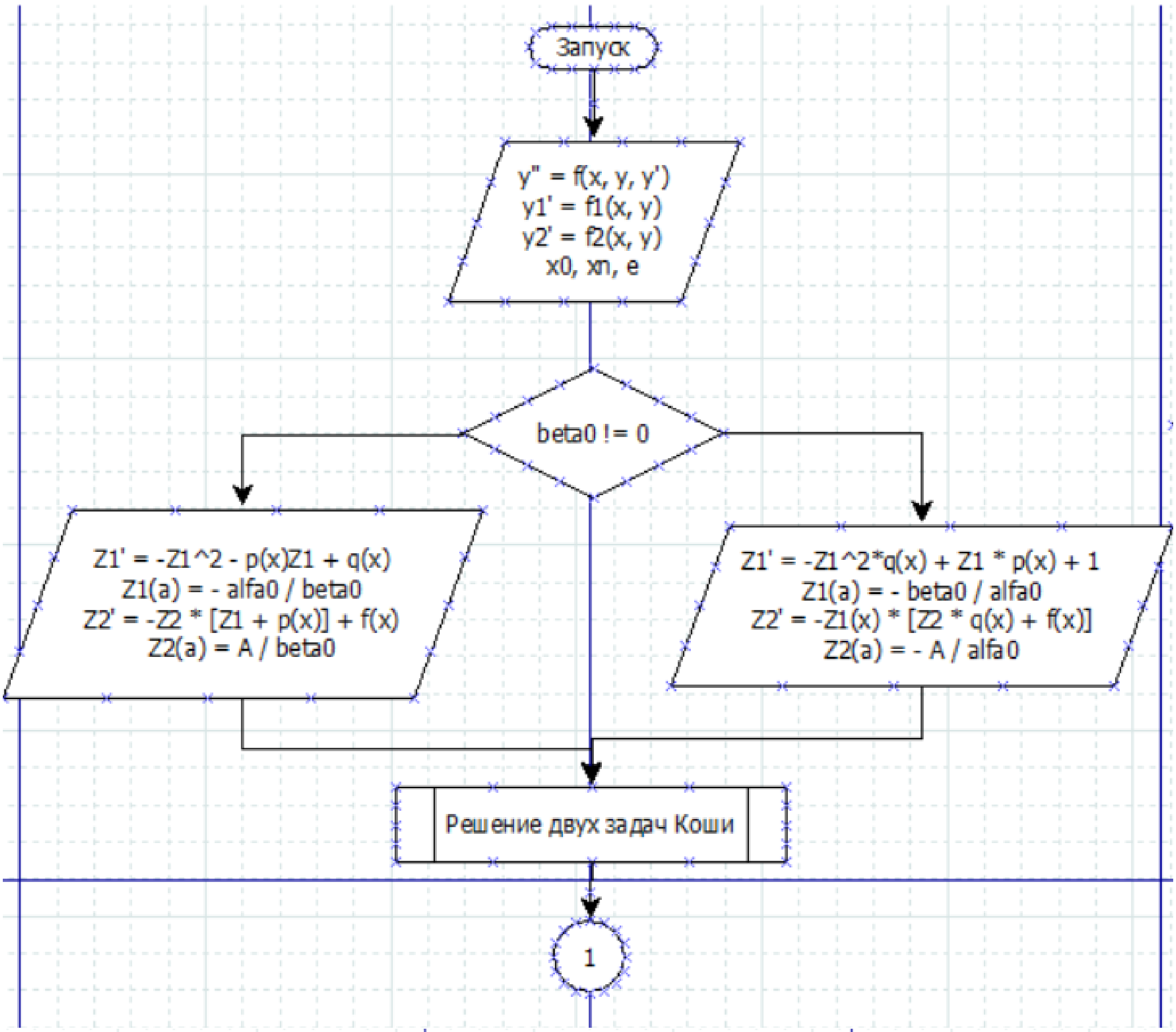
4. По найденным значениям $Z_1(x), Z_2(x)$ решается задача Коши в обратном ходе

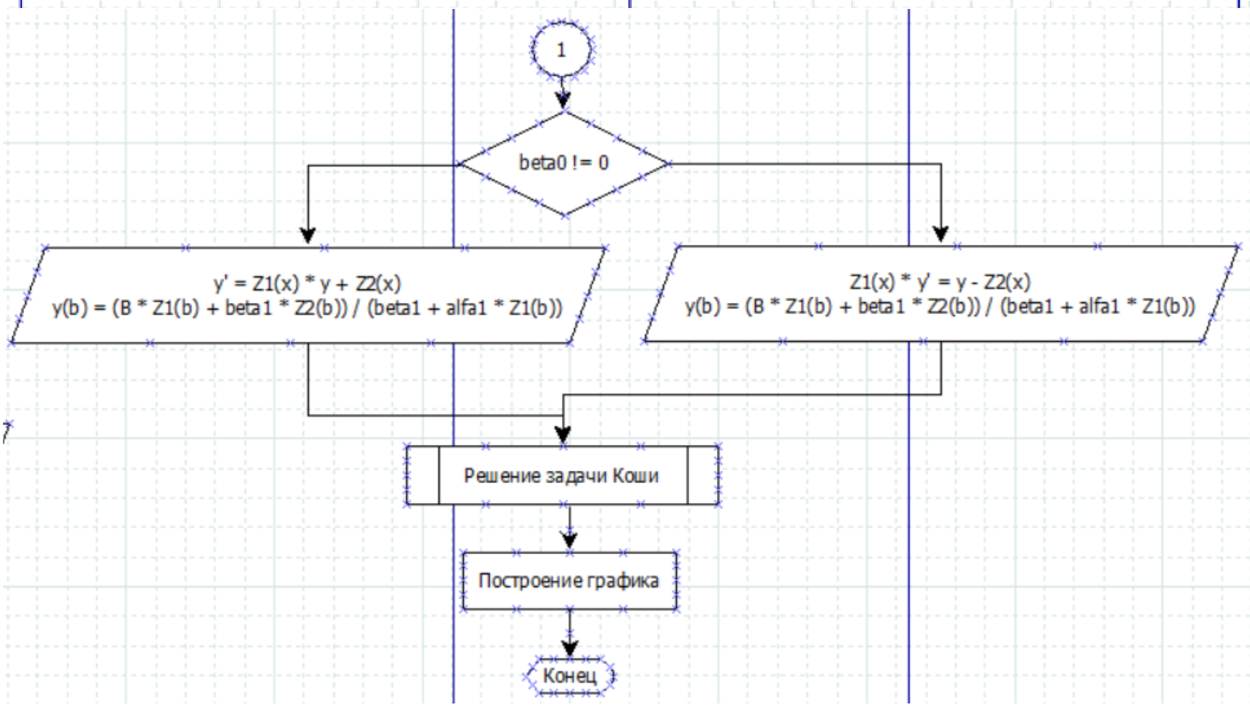
5. Строится график по результатам обратного хода

Листинг кода

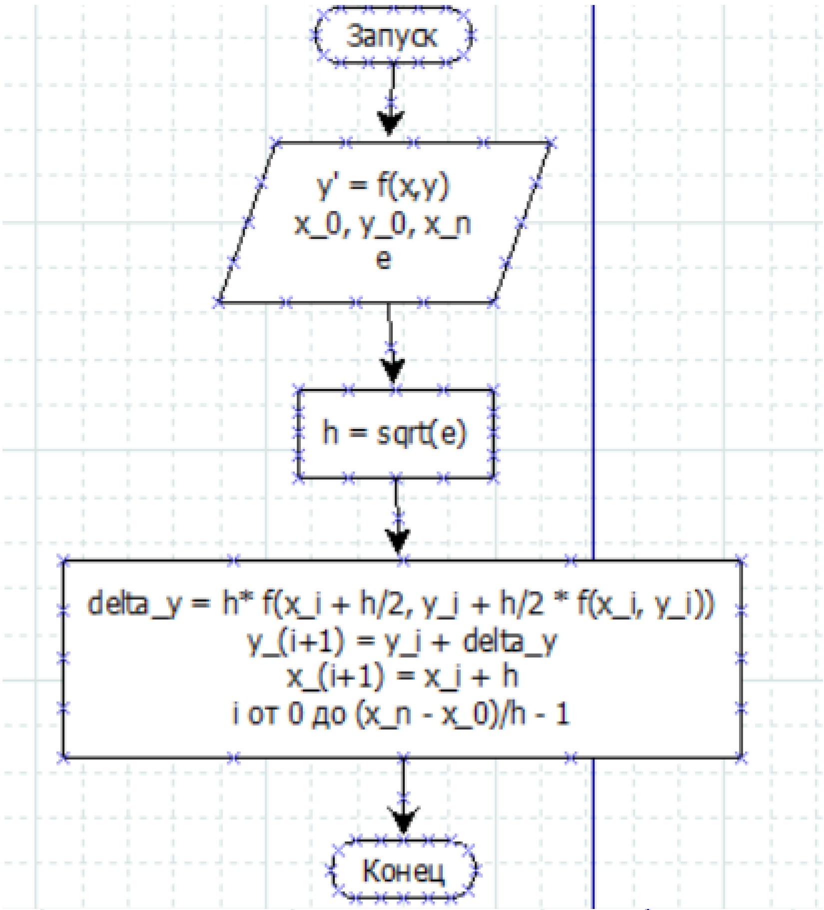
```
def advanced_euler_method (func, x0, y0, xn, step, sign):
    x = [x0]
    y = [y0]
    current_x = x0
    current_y = y0
    while current_x < xn:
        current_res_func = func.subs([('x', current_x), ('y', current_y)]).evalf()
        delta_y = step * func.subs([('x', current_x + step/2), ('y', current_y +
step/2 * current_res_func)]).evalf()
        current_y += delta_y * sign
        current_x += step
        x.append(current_x)
        y.append(current_y)
    return y


def straight_turn(system, interval, start_values, step):
    z1_points = [start_values[0]]
    z2_points = [start_values[1]]
    current_x = interval[0] + step
    while current_x < interval[1]:
        new_point1 = advanced_euler_method(system[0], current_x, z1_points[-1],
current_x + step, step, 1)
        new_point2 = advanced_euler_method(system[1].subs('z1', z1_points[-1]),
current_x, z2_points[-1],
                                           current_x + step, step, 1)
        z1_points.append(new_point1[-1])
        z2_points.append(new_point2[-1])
        current_x += step

    return z1_points, z2_points


def reverse_turn(func, interval, z1_points, z2_points, yn, step):
    x = [interval[1]]
    y = [yn]
    current_x = x[-1] - step
    i = 2
    while current_x > interval[0]:
        new_y = advanced_euler_method(func.subs([('z1', z1_points[-i]), ('z2',
z2_points[-i])]),
                                      current_x, y[-1], current_x + step, step, -
1)
        x.append(current_x)
        y.append(new_y[-1])
        current_x -= step
        i += 1

    return x, y
```

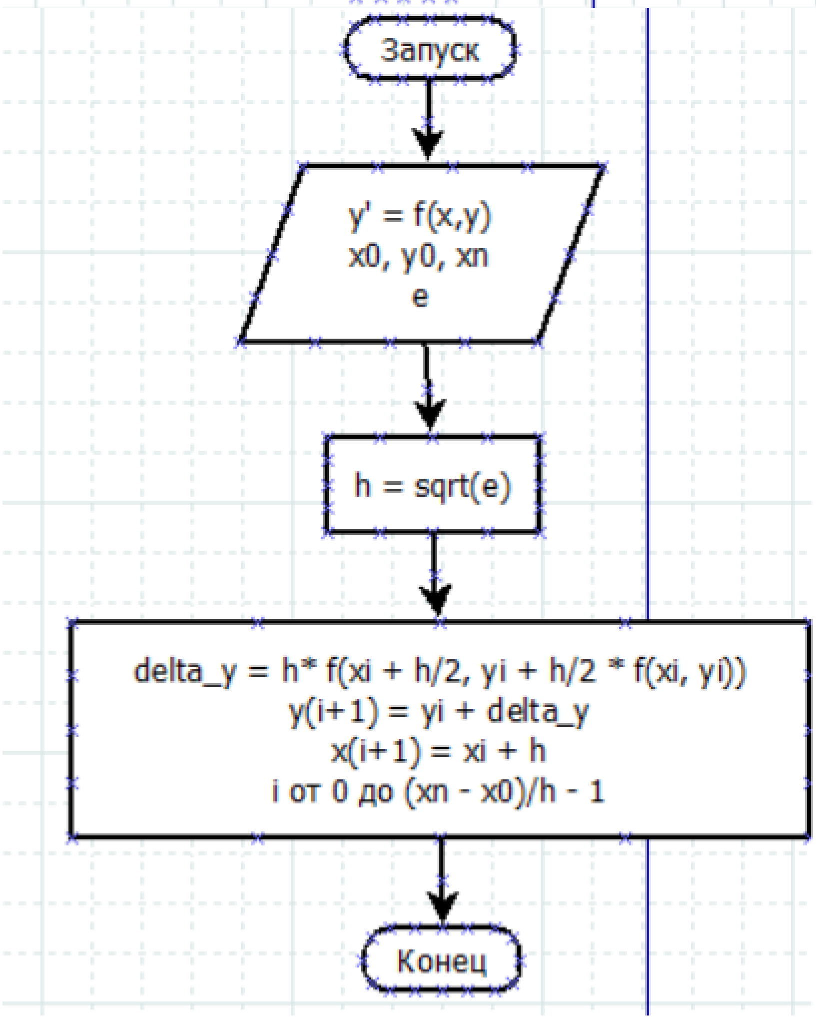
Блок-схема





Решение задачи Коши


Запуск
y' = f(x,y)
x_0, y_0, x_n
e
h = sqrt(e)
delta_y = h* f(x_i + h/2, y_i + h/2 * f(x_i, y_i))
y_(i+1) = y_i + delta_y
x_(i+1) = x_i + h
i от 0 до (x_n - x_0)/h - 1
Конец

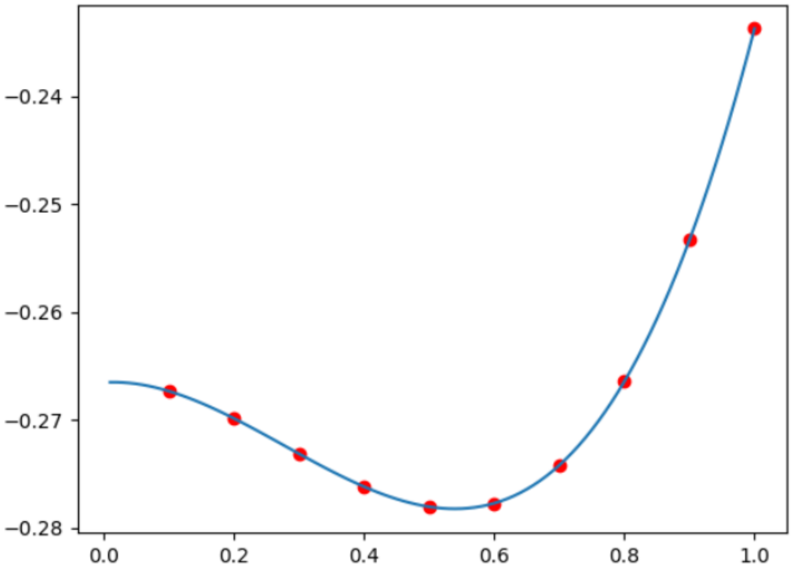

Запуск
y' = f(x,y)
x0, y0, xn
e
h = sqrt(e)
delta_y = h* f(xi + h/2, yi + h/2 * f(xi, yi))
y(i+1) = yi + delta_y
x(i+1) = xi + h
i от 0 до (xn - x0)/h - 1
Конец

Тесты и результаты

Тест 1.

```
Choose one of differential equations:
1) y'' + (x+1)y' - 2y = 2
2) y'' = x + y
3) y'' + 2xy' - 2y = 2x^2
2) y' - x^2 - 2y = 0
3) y' =  sin(x)
4) y' = 1 + e^(-x) * cos(pi * x) - y
2
Input start point
x >= 0
Input end point
x <= 1
Input precision
3
Choose one of the pairs of equations:
1) y(0.0) - y'(0.0) = -1
   y(1.0) = 4
2) y(0.0) = 0
   y(1.0) = 0
3) y'(0.0) = 0
   y(1.0) + y'(1.0) = 0
2) y' - x^2 - 2y = 0
3) y' =  sin(x)
4) y' = 1 + e^(-x) * cos(pi * x) - y
3
```

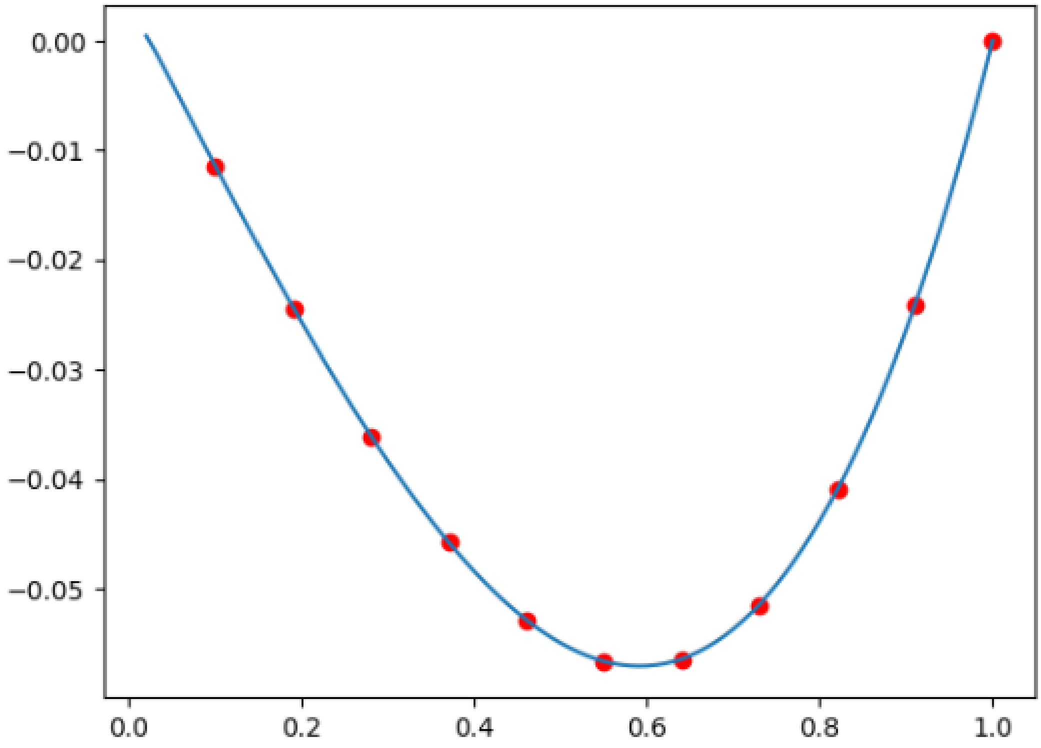
Точное решение: $y(x) = -x + e^{-x-1} + e^{x-1} - e^{-x}$



Тест 2.

```
Choose one of differential equations:
1) y'' + (x+1)y' - 2y = 2
2) y'' = x + y
3) y'' + 2xy' - 2y = 2x^2
2
Input start point
x >= 0
Input end point
x <= 1
Input precision
3
Choose one of the pairs of equations:
1) y(0.0) - y'(0.0) = -1
   y(1.0) = 4
2) y(0.0) = 0
   y(1.0) = 0
3) y'(0.0) = 0
   y(1.0) + y'(1.0) = 0
2
```

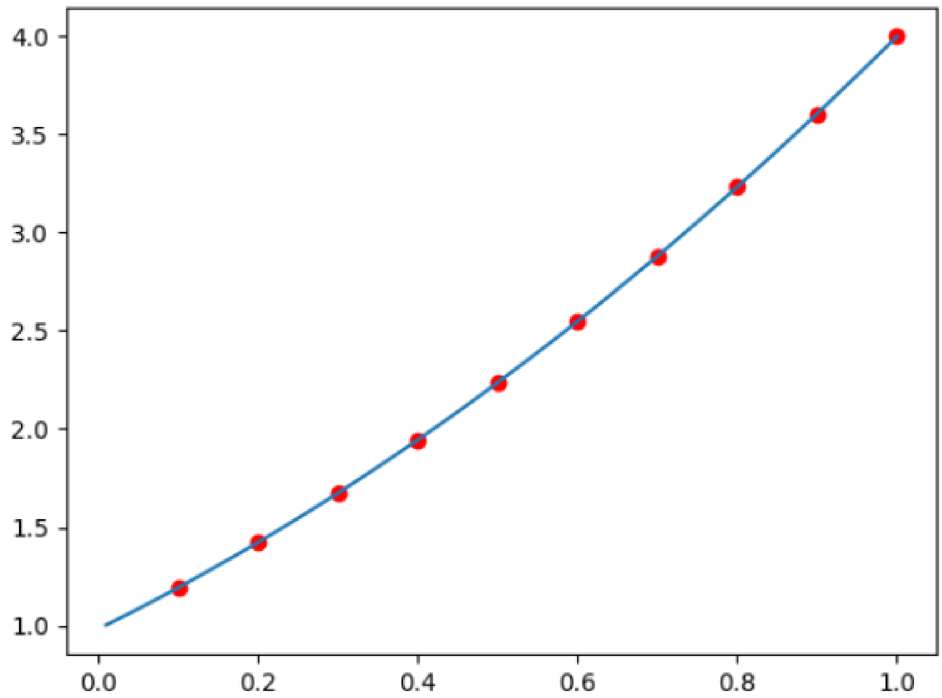
Точное решение: $y(x) = \frac{e^2 * x - x + e^{1-x} - e^{x+1}}{1-e^2}$



Тест 3

```
Choose one of differential equations:
1) y'' + (x+1)y' - 2y = 2
2) y'' = x + y
3) y'' + 2xy' - 2y = 2x^2
1
Input start point
x >= 0
Input end point
x <= 1
Input precision
3
Choose one of the pairs of equations:
1) y(0.0) - y'(0.0) = -1
   y(1.0) = 4
2) y(0.0) = 0
   y(1.0) = 0
3) y'(0.0) = 0
   y(1.0) + y'(1.0) = 0
1
```

Точное решение: $y(x) = (x+1)^2$



Вывод.

Дифференциальное уравнение (дальше ДУ) – уравнение, в которое входят производные функции и также могут входить сама функция, независимая переменная x и параметры. Найти решение ДУ – значит найти множество всех n раз дифференцируемых функций, которые удовлетворяют заданному уравнению. Обычно такое множество имеет вид $y = f(x, C)$, где C – произвольная постоянная. Также такое множество называют общим решением ДУ.

Краевая задача – задача о нахождении решения заданного дифференциального уравнения или их системы, удовлетворяющего краевым условиям в концах интервала или на границе области. Решить задачу означает найти функцию или множество функций или доказать, что задача решений не имеет.

Суть метода прогонки состоит в последовательном исключении неизвестных из трехдиагональной матрицы. Его отличием от других методов решения краевых задач является наличие двух ходов: прямого и обратного во время решения ДУ. По сравнению с методом стрельбы в методе прогонки изначально нужно разбить отрезок на n равных по длине частей, в то время как в методе стрельбы используются значения только в тех точках, которые были получены при помощи метода решения нелинейного уравнения. Также метод прогонки на более большей длине отрезка сделает меньшее количество итераций, нежели метод стрельбы, так как количество итераций для метода прогонки – 3*n, а для метода стрельбы k*n, где k – количество попыток пристрелок. Однако метод прогонки может быть использован только для линейных ДУ, а метод стрельбы как для линейных, так и для нелинейных.