Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №5:

«Сравнительный анализ методов численного решения краевой задачи для линейного дифференциального уравнения второго порядка.

Студент ИУ9-61Б (Группа) — Е.А. Матвеев (И.О. Фамилия)

Проверила — А. Б. Домрачева (И.О. Фамилия)

2023 г.

Содержание

1 Цель

Целью данной работы является изучение и сравнение двух методов решения краевой задачи для дифференциальных уравнений (далее ДУ) : метода прогонки и метода стрельбы.

2 Постановка задачи

Дано: ДУ $y'' + p(x)y' + q(x)y = f(x)$, краевые условия: $y(0) = a$, $y(1) = b$, набор точек $\{x_i\}_1^n \in [0; 1]$

Требуется: Найти частное решение данного ДУ в точках $\{x_i\}_1^n$, отвечающее краевым условиям двумя методами: методом прогонки и методом стрельбы. Полученные результаты сравнить с аналитическим решение и друг с другом.

Для достижения цели работы были поставлены следующие задачи:

1. Изучить теоретическую основу обоих методов.
2. Реализовать алгоритмы методов на языке программирования python3.
3. Рассчитать значения погрешностей, сравнить результаты.

3 Теоретические сведения

В обоих методах осуществляется переход от ДУ на отрезке, к системе уравнений относительно значений функции в наборе из n точек равномерно, распределенных по этому отрезку.

В методе прогонки осуществляется переход к дрехдиагональной СЛАУ, в методе стрельбы - последовательному нахождению двух массивов, однако первые элементы второго массива инициализируются случайным образом, что вносит непредсказуемость в результат.

Погрешность результата вычисляется как $max_{0 \leq i \leq n} |y(x_i) - y_i|$, где y_i - значение функции в точке x_i, полученное в результате работы вычислительного алгоритма, а $y(x)$ - аналитическое решение задачи.

Для обоих методов справедливо следующее утверждение: при увеличении количества точек (n), для которых вычисляется значение функции, методическая погрешность убывает.

4 Реализация

На листинге 1 представлена реализация метода прогонки и метода стрельбы.

```
import numpy as np
import pandas as pd
pd.options.display.float_format = '{:.10f}'.format

p = -4
q = 0

c1 = (3.56+301/1024)/4

def analytic_answer(x):
    return -1/28 * x**7 - \
            1/16 * x**6 - \
            3/32 * x**5 - \
            15/128 * x**4 - \
            15/128 *  x**3 - \
            45/512* x**2 - \
            301/1024 *x + \
            c1* np.e**(4*x) + \
            (-c1)
def f(x):
    return x**6+1

x0 = 0
y0 = analytic_answer(x0)
xn = 1
yn = analytic_answer(xn)
y0, yn

def create_diagonals():
    top = []
    mid = []
    low = []
    res = []

    mid.append(h*h*q-2)
    top.append(1+h/2*p)

    x1 = x0 + (xn-x0)*h*1
    f1 = analytic_answer(x1)
    res.append(h*h*f1 - y0*(1-h/2*p))

    for i in range(2, n-1):
        low.append(1-h/2*p)
        mid.append(h*h*q-2)
        top.append(1+h/2*p)

        xi = x0 + (xn-x0)*h*i
        fi = f(xi)
        res.append(h*h*fi)

    low.append(1-h/2*p)
    mid.append(h*h*q-2)

    xn_1 = x0 + (xn-x0)*h*(n-1)
    fn_1 = analytic_answer(xn_1)
    res.append(h*h*fn_1 - yn*(1+h/2*p))

    return [0] + low, mid, top + [0], res

def solve_stripe(low, mid, top, res):
```

```
    n = len(mid)
    low = np.array(low)
    mid = np.array(mid)
    top = np.array(top)
    res = np.array(res)

    alpha = np.zeros((n,))
    beta = np.zeros((n,))
    alpha[0] = -top[0] / mid[0]
    beta[0] = res[0] / mid[0]

    for i in range(1, n):
        alpha[i] = -top[i]/(low[i]*alpha[i-1] + mid[i])
        beta[i] = (res[i] - low[i]*beta[i-1])/(low[i]*alpha[i-1] + mid[i])

    x = np.zeros((n,))
    x[n-1] = beta[n - 1]

    for i in range(n-1,0,-1):
        x[i-1] = alpha[i-1]*x[i] + beta[i-1]

    return list(x)

def create_matrix(low, mid, top):
    mat = np.diag(mid)
    for i in range(1,len(mid)):
        mat[i-1, i] = top[i-1]
        mat[i, i-1] = low[i]
    return mat


low, mid, top, res = create_diagonals()
m = create_matrix(low,mid,top)
res_my = np.array(
    [y0] + list(solve_stripe(low, mid, top, res)) + [yn]
)


res_true = np.array(
    [y0] + list(np.vectorize(analytic_answer)(np.linspace(0.1,1,9,False))) + [yn]
)
def shooting(Oh):
    y_0 = np.empty(n+1)
    y_1 = np.empty(n+1)

    y_0[0] = y0
    y_0[1] = y0 + Oh
    y_1[0] = 0
    y_1[1] = Oh

    for i in range(1, n):
        y_0[i+1] = (f(x0+i*h)*h**2 + (2-q*h**2)*y_0[i] - (1-p*h/2)*y_0[i-1]) / (1 + p*h/2)
        y_1[i+1] = ((2-q*h**2)*y_1[i] - (1-p*h/2)*y_1[i-1]) / (1 + p*h/2)

    if abs(y_1[n]) < 0.001:
        return gun(Oh+1)
    else:
        c1 = (yn - y_0[n]) / y_1[n]
    return [y_0[i] + c1 * y_1[i] for i in range(n+1)]
```

Листинг 1: Реализация метода стерльбы и метода прогонки

5 Тестирование

Для тестирования было взято ДУ:

$$y'' - 4y' = x^6 + 1;$$

Граничные условия:

$$y(0) = 0; y(1) = 50.8329095465$$

Аналитическое решенение задачи высчитывается следующим образом:
*c1 = (3.56+301/1024)/4; y(x) = -1/28 * x**7 - 1/16 * x**6 - 3/32 * x**5 - 15/128 * x**4 - 15/128 * x**3 - 45/512* x**2 - 301/1024 *x + c1* np.e**(4*x) + (-c1)*

Численное решение будет находится для 11-ти точек: 0, 0.1, 0.2 ... 1.

На рисунке 1 представлены результаты, полученные методом прогонки и методом стрельбы. Для каждой точки найдено отклонение полученного результата от вычисленного аналитически.

	xi	analytic ans	run ans	run abs diff	sh ans	sh abs diff
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.1	0.44346302469014753	0.4265244659773178	0.016938558712829732	0.42547509255389093	0.0179879321362566
2	0.2	1.1173277845172533	1.0718544527519214	0.04547333176533197	1.0761877438847274	0.041140040632525965
3	0.3	2.134912717743825	2.052350232913827	0.08256248482999817	2.064757520880982	0.0701551968628431
4	0.4	3.6652775562538014	3.535603015656685	0.12967454059711647	3.560121298875363	0.10515625737843814
5	0.5	5.9606649643361855	5.773033389770972	0.18763157456521373	5.815718165866936	0.14494679846924985
6	0.6	9.39747982320503	9.141874263442402	0.25560555976262833	9.211808778854293	0.18567104435073745
7	0.7	14.537495000421936	14.208218773949545	0.3292762264723912	14.319027898335328	0.21846710208660802
8	0.8	22.219271032532138	21.821706152210258	0.3975648803218803	21.99382719005688	0.22544384247525784
9	0.9	33.694698065979345	33.25771401960134	0.4369840463780079	33.52180292763921	0.17289513834013803
10	1.0	50.832909546512695	50.832909546512695	0.0	50.832909546512695	0.0

Рисунок 1 — Таблица результатов, полученных методом прогонки и методом стрельбы.

Погрешность результата метода прогонки составила 0.4369840463780079, а метода стрельбы 0.22544384247525784.

6 Вывод

При выполнении лабораторной работы был изучен и реализован в программном коде два метода решения краевой задачи ДУ 2ого порядка: метода прогонки и метода стрельбы.

Несмотря на то, что в методе стрельбы присутствует использование случайной величины, на каждой точке из набора $\{x_i\}_1^n$ он дает меньшее отклонение от теоретического результата, что делает его более предпочтительным для решения краевой задачи для ДУ второго порядка.