|  |  |
| --- | --- |
|  | **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 Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**«Метод прогонки для решения краевой задачи для линейного дифференциального уравнения второго порядка»**

**ПО КУРСУ:**

***«Численные методы»***

Студент *Балтаева М. Б.*

Преподаватель *Домрачева А. Б.*

*Москва, 2024 г.*

Содержание

[ЦЕЛЬ 3](#_Toc163166588)

[ПОСТАНОВКА ЗАДАЧИ 4](#_Toc163166589)

[РЕАЛИЗАЦИЯ 5](#_Toc163166590)

[ТЕСТИРОВАНИЕ 6](#_Toc163166591)

[ВЫВОД 7](#_Toc163166592)

# ЦЕЛЬ

Целью данной лабораторной работы является изучение метода прогонки для решения краевой задачи для дифференциальных уравнений второго рода. Требуется найти частное решение, отвечающее краевым условиям.

# ПОСТАНОВКА ЗАДАЧИ

Дано: ДУ ,

Краевые условия: у(0) = а, у(1) = b, x

Задача:

1. Написать и отладить процедуру для решения трехдиагональной линейной системы методом прогонки.
2. Решить аналитически задачу Коши
3. Найти численное решение краевой задачи для того же уравнения с краевыми условиями у(0) = а, у(1) = b при n = 10
4. Найти погрешность численного решения

# РЕАЛИЗАЦИЯ

Листинг main.py

import numpy as np  
  
def fi(x):  
 return 5.0  
  
def analytic\_func(x):  
 return np.exp(4.0 \* x) / 4.0 + np.exp(3.0 \* x) / 3.0 + 5.0 / 12.0  
  
def solve(pi, qi, a, b, n):  
 h = 1.0 / n  
  
 xy = [(i \* h, analytic\_func(i \* h)) for i in range(n + 1)]  
  
 ai = 1.0 - (h / 2.0) \* pi  
 bi = h\*\*2 \* qi - 2.0  
 ci = 1.0 + (h / 2.0) \* pi  
  
 d\_res = [h\*\*2 \* fi(xy[i][0]) for i in range(n)]  
 d\_res[1] -= a \* ai  
 d\_res[n-1] -= b \* ci  
  
 alpha = np.zeros(n)  
 betta = np.zeros(n)  
  
 y = bi  
 alpha[1] = -ci / y  
 betta[1] = d\_res[1] / y  
  
 for i in range(2, n):  
 y = bi + ai \* alpha[i - 1]  
 alpha[i] = -ci / y  
 betta[i] = (d\_res[i] - ai \* betta[i - 1]) / y

y\_ch = np.zeros(n + 1)  
 y\_ch[0] = a  
 y\_ch[n-1] = betta[n-1]  
 y\_ch[n] = b  
 for i in range(n - 2, 0, -1):  
 y\_ch[i] = alpha[i] \* y\_ch[i + 1] + betta[i]  
 return xy, y\_ch  
  
def main():  
 pi = -7.0  
 qi = 12.0  
 a = analytic\_func(0.0)  
 b = analytic\_func(1.0)  
 n = 10  
  
 xy, y\_ch = solve(pi, qi, a, b, n)  
  
 pogr = max(abs(xy[i][1] - y\_ch[i]) / y\_ch[i] for i in range(n))  
  
 print(" x Analytic Solution Numerical Solution Relative Error")  
 for i in range(n + 1):  
 print(f"{xy[i][0]:.6f} {xy[i][1]:.6f} {y\_ch[i]:.6f} {(abs(xy[i][1] - y\_ch[i]) / y\_ch[i]):.6f}")  
  
 print(f"Максимальная относительная погрешность: {pogr:.6f}")  
  
main()

# ТЕСТИРОВАНИЕ

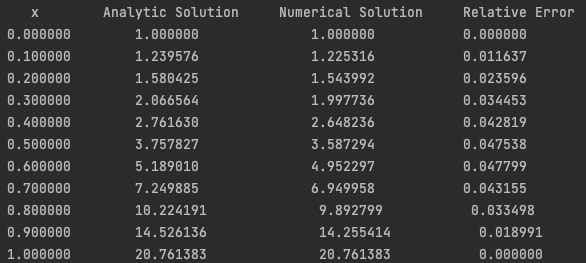
Для тестирование было взято ДУ:

Граничные условия y(0) = 1, y(1) = 20.761383149348617

Аналитическое решение задачи:

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

Ниже представлены результаты, полученные методом прогонки, Для каждой точки найдена относительная погрешность полученного результата от вычисленного аналитически.



Максимальная относительная погрешность составила: 0.047799

# ВЫВОД

При выполнении лабораторной работы был изучен и реализован в программном коде метод прогонки для краевой задачи ДУ 2-ого порядка.

Была выведена относительная погрешность, так как решение - экспонециальная функция и она растет очень быстро и абсолютная погрешность будет большая.

Заметим, что функция на краях имеет меньшую погрешность, а в середине она увеличивается. При увеличении количества разбиений (n), для которых вычисляется значение функции, методическая погрешность убывает.