Лабораторная работа №7. Метод стрельбы

Денис Слынко, Б01-818

2021

Задача:

Найти решение краевой задачи для одномерного стационарного уравнения теплопроводности

$$\begin{cases} \frac{d}{dx}((x+1)\frac{du}{dx}) - e^x u = -e^{-x^2}, \\ u_x(0) = 0, \\ -2u_x(0) = u(1). \end{cases}$$

в одиннадцати равноудалённых точках отрезка $[0, 1]$ с относительной точностью 0.0001.

1 Результаты вычислений

Вычисления численного решения задачи Штурма-Лиувилля проводились с помощью метода трёхдиагональной прогонки. Полученные численные значения искомой функции в точках отрезка $[0, 1]$:

0.35233929; 0.35233929; 0.34793276; 0.34014776; 0.32988073; 0.31792073; 0.304964; 0.29162312; 0.27843357; 0.26585962; 0.2543005.

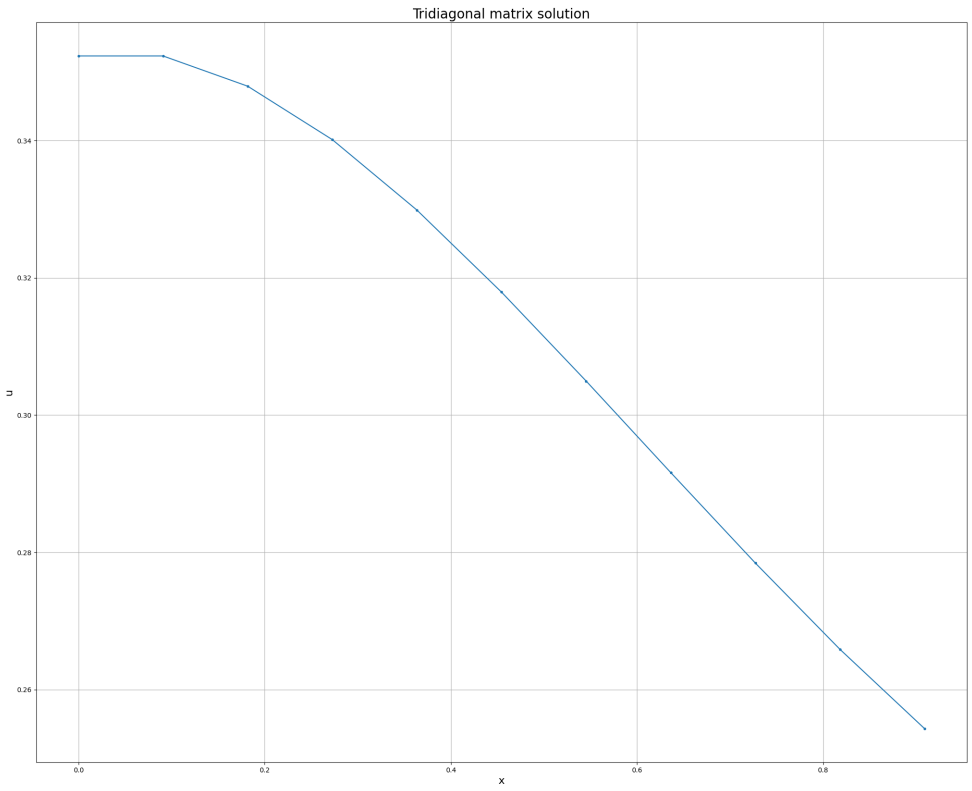

Рис. 1: Решение методом трёхдиагональной прогонки для 11 равноудалённых точек отрезка $[0, 1]$

2 Код программы

// main.py

```
#!/usr/bin/python3.8

import matplotlib.pyplot as plt
import numpy as np
```

```
import subprocess
import sys
import os
import equation


def create_plots_dir(name="Plots"):
    if not os.path.isdir(os.path.join(os.path.dirname(sys.argv[0]),
        name)):
        subprocess.check_output(f"mkdir {name}", shell=True)


N = 11
h = (equation.Equation.b - equation.Equation.a) / float(N)

M, d = equation.get_matrices(N, h)

p, r = equation.get_auxiliary_matrices(M, d)

u = np.zeros(N)
u[N - 1] = r[N - 1]
for i in range(N - 2, -1, -1):
    u[i] = r[i] - p[i] * u[i + 1]
x = np.arange(0.0, N * h, step=h, dtype=float)

print(f"The solution for {N} points")
print(u)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x, u, marker='o', markersize=3)
ax.set_title("Tridiagonal matrix solution", fontsize=20)
ax.set_xlabel("x", fontsize=16)
ax.set_ylabel("u", fontsize=16)
ax.grid()

fig.set_figheight(20)
fig.set_figwidth(25)

dirname = "Plots"
```

```
create_plots_dir(dirname)
fig.savefig(os.path.join(dirname, "plot.png"))
```

// equation.py

```
import numpy as np


class Equation:
    epsilon = 0.0001
    a = 0.0
    b = 1.0

    @staticmethod
    def k(x):
        return x + 1

    @staticmethod
    def q(x):
        return np.exp(x)

    @staticmethod
    def f(x):
        return np.exp(-1.0 * x * x)


def get_matrices(N, h):
    M = np.zeros(N * N).reshape(N, N)
    d = np.zeros(N)

    M[0][0] = -1.0
    M[0][1] = 1.0
    M[N - 1][N - 2] = -1.0 * Equation.k(1.0)
    M[N - 1][N - 1] = h + Equation.k(1.0)
    for i in range(1, N - 1):
        M[i][i - 1] = Equation.k((float(i) - 0.5) * h) / h / h
        M[i][i] = -1.0 * (
                    Equation.k((float(i) - 0.5) * h) +
                        Equation.k((float(i) + 0.5) * h)) / h / h - \
                  Equation.q(i * h)
```

```
        M[i][i + 1] = Equation.k((float(i) + 0.5) * h) / h / h
        d[i] = -1.0 * Equation.f(i * h)

    return M, d


def get_auxiliary_matrices(M, d):
    if M.shape[0] != M.shape[1] or M.shape[0] != d.shape[0]:
        raise ValueError("invalid matrices")

    N = d.shape[0]
    p = np.zeros(N)
    r = np.zeros(N)

    p[0] = M[0][1] / M[0][0]
    r[0] = d[0] / M[0][0]

    for i in range(1, N - 1):
        p[i] = M[i][i + 1] / (M[i][i] - M[i][i - 1] * p[i - 1])
        r[i] = (d[i] - M[i][i - 1] * r[i - 1]) / (M[i][i] - M[i][i -
            1] * p[i - 1])
    r[N - 1] = (d[N - 1] - M[N - 1][N - 2] * r[N - 2]) / (M[N - 1][N
        - 1] - M[N - 1][N - 2] * p[N - 2])

    return p, r
```

3 Литература

1 Демченко В.В. Вычислительный практикум по прикладной математике: Учебное пособие. - М.: МФТИ, 2007. - 196 с. ISBN 5-7417-0186-8