Лабараторная Работа № 2
Метод сеток решения краевой задачи для ОДУ

Царик Виталий
3-й курс 2-я группа

4 декабря 2019 г.

1 Постановка задачи

Условие

Дана линейная краевая задача. Необходимо построить для краевой задачи разностную схему второго порядка аппроксимации на минимальном шаблоне и с помощью метода прогонки при $h = 0.05$ найти её численное решение. Оценить погрешность полученного численного решения с помощью правила Рунге. Обосновать применимость метода прогонки для решения разностной задачи. Построить график численного решения задачи.

Задача

$$\begin{cases} u''(x) - \dfrac{1+x}{2}u'(x) - \cos\dfrac{x}{2}u(x) = -\left(\dfrac{x}{2}+1\right), -1 < x < 1 \\ u'(-1) = 0 \\ u'(1) = 1 \end{cases} \tag{1}$$

2 Краткие теоретические сведения

Построим разностную схему

$$\begin{cases} y_{\overline{x}x} - \dfrac{1+x}{2}y_{\overset{\circ}{x}} - \cos\dfrac{x}{2}y = -\left(\dfrac{x}{2}+1\right) \\ y_x(-1) = A_0 y(-1) + B_0 \\ y_{\overline{x}}(1) = A_1 y(1) + B_1 \end{cases}$$

Найдём коэффициенты A_0, B_0, чтобы получить 2-й порядок аппроксимации

$$
\begin{aligned}
\nu_n(-1) &= u_x(-1) - (A_0 u(-1) + B_0) = \\
&= u'(-1) + \frac{h}{2}u''(-1) + O(h^2) - (A_0 u(-1) + B_0) = \\
&\left[\begin{array}{l} u''(x) = \frac{1+x}{2}u'(x) + \cos\frac{x}{2}u(x) - \left(\frac{x}{2}+1\right) \\ u'(-1) = 0 \end{array}\right] \\
&= \frac{h}{2}\left(\cos\left(-\frac{1}{2}\right)u(-1) - \frac{1}{2}\right) + O(h^2) - (A_0 u(-1) + B_0)
\end{aligned}
$$

$$
\begin{cases}
A_0 = \frac{h}{2}\cos\frac{1}{2} \\
B_0 = -\frac{h}{4}
\end{cases}
$$

Аналогично, найдём A_1, B_1

$$
\begin{aligned}
\nu_n(1) &= u_{\overline{x}}(1) - (A_1 u(1) + B_1) = \\
&= u'(1) - \frac{h}{2}u''(1) + O(h^2) - (A_1 u(1) + B_1) = \\
&\left[\begin{array}{l} u''(x) = \frac{1+x}{2}u'(x) + \cos\frac{x}{2}u(x) - \left(\frac{x}{2}+1\right) \\ u'(1) = 1 \end{array}\right] \\
&= 1 - \frac{h}{2}\left(1 + \cos\frac{1}{2}u(1) - \frac{3}{2}\right) + O(h^2) - (A_1 u(1) + B_1)
\end{aligned}
$$

$$
\begin{cases}
A_1 = -\frac{h}{2}\cos\frac{1}{2} \\
B_1 = 1 + \frac{h}{4}
\end{cases}
$$

Итого

$$
\begin{cases}
y_{\overline{x}x} - \frac{1+x}{2}y_{\overset{\circ}{x}} - \cos\frac{x}{2}y = -\left(\frac{x}{2}+1\right) \\
y_x(-1) = \frac{h}{2}\cos\frac{1}{2}y(-1) - \frac{h}{4} \\
y_{\overline{x}}(1) = -\frac{h}{2}\cos\frac{1}{2}y(1) + 1 + \frac{h}{4}
\end{cases}
\tag{2}
$$

Индексная форма

$$\begin{cases} \dfrac{y_{i+1}-2y_i+y_{i-1}}{h^2} - \dfrac{1+x_i}{2}\dfrac{y_{i+1}-y_{i-1}}{2h} - \cos\dfrac{x_i}{2}y_i = -\left(\dfrac{x_i}{2}+1\right), i=\overline{1,N-1} \\ \dfrac{y_1-y_0}{h} = \dfrac{h}{2}\cos\dfrac{1}{2}y_0 - \dfrac{h}{4} \\ \dfrac{y_N-y_{N-1}}{h} = -\dfrac{h}{2}\cos\dfrac{1}{2}y_N + 1 + \dfrac{h}{4} \end{cases} \tag{3}$$

Итого, имеем систему из $N+1$ уравнений с трёхдиагональной матрицей

$$\begin{cases} -\left(\dfrac{1}{h}+\dfrac{h}{2}\cos\dfrac{1}{2}\right)y_0 + \dfrac{1}{h}y_1 = -\dfrac{h}{4} \\ \left(\dfrac{1}{h^2}+\dfrac{1+x_i}{4h}\right)y_{i-1} - \left(\dfrac{2}{h^2}+\cos\dfrac{x_i}{2}\right)y_i + \left(\dfrac{1}{h^2}-\dfrac{1+x_i}{4h}\right)y_{i+1} = -\left(\dfrac{x_i}{2}+1\right), \\ i=\overline{1,N-1} \\ -\dfrac{1}{h}y_{N-1} + \dfrac{1}{h}y_N = -\dfrac{h}{2}\cos\dfrac{1}{2}y_N + 1 + \dfrac{h}{4} \end{cases} \tag{4}$$

Будем решать её методом прогонки

$$\begin{cases} a_i = -\left(\dfrac{1}{h^2}+\dfrac{1+x_i}{4h}\right), i=\overline{1,N-1} \\ a_N = \dfrac{1}{h} \\ b_0 = -\dfrac{1}{h} \\ b_i = -\left(\dfrac{1}{h^2}-\dfrac{1+x_i}{4h}\right), i=\overline{1,N-1} \\ c_0 = -\left(\dfrac{1}{h}+\dfrac{h}{2}\cos\dfrac{1}{2}\right) \\ c_i = -\left(\dfrac{2}{h^2}+\cos\dfrac{x_i}{2}\right), i=\overline{1,N-1} \\ c_N = \dfrac{1}{h}+\dfrac{h}{2}\cos\dfrac{1}{2} \\ f_0 = -\dfrac{h}{4} \\ f_i = -\left(\dfrac{x_i}{2}+1\right), i=\overline{1,N-1} \\ f_N = 1+\dfrac{h}{4} \end{cases} \tag{5}$$

3 Листинг программы

Листинг 1: файл main.py

```
from math import cos

import numpy as np
import matplotlib.pyplot as plt

from matrix_solver import solve_tridiag

h = 0.05
[A, B] = [-1, 1]


def init_data(h):
    x = np.arange(A, B + h, h)
    n = len(x)

    a = np.empty(n - 1)
    av = np.vectorize(lambda x: -(1 / (h * h) + (1 + x) / (4 * h)))
    a[:-1] = av(x[1:-1])
    a[-1] = 1 / h

    b = np.empty(n - 1)
    b[0] = -1 / h
    bv = np.vectorize(lambda x: -(1 / (h * h) - (1 + x) / (4 * h)))
    b[1:] = bv(x[1:-1])

    c = np.empty(n)
    c[0] = -(1 / h + h / 2 * cos(1 / 2))
    cv = np.vectorize(lambda x: -(2 / (h * h) + cos(x / 2)))
    c[1:-1] = cv(x[1:-1])
    c[-1] = 1 / h + h / 2 * cos(1 / 2)

    f = np.empty(n)
    f[0] = -h / 4
    fv = np.vectorize(lambda x: -(x / 2 + 1))
    f[1:-1] = fv(x[1:-1])
    f[-1] = 1 + h / 4

    return a, b, c, f, x


def runge_rule(y1, y2, order=2):
    return np.amax((abs(y1 - y2))) / (2 ** order - 1)


def solve(h):
    a, b, c, f, x = init_data(h)
    y = solve_tridiag(a, b, c, f)
    return x, y


```

```python
if __name__ == '__main__':
    x1, y1 = solve(h)
    x2, y2 = solve(h * 2)
    print(runge_rule(y1[::2], y2))

    plt.plot(x1, y1)
    plt.show()
```

Листинг 2: файл matrix_solver.py

```python
import numpy as np


def solve_tridiag(a, b, c, f):
    n = len(f)
    beta = np.zeros(n)
    alpha = np.zeros(n)
    x = np.zeros(n)

    alpha[0] = 0
    beta[0] = 0
    for i in range(n - 1):
        din = c[i] - a[i] * alpha[i]
        alpha[i + 1] = b[i] / din
        beta[i + 1] = (f[i] + a[i] * beta[i]) / din

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

    return np.array(x)
```

4 Результаты

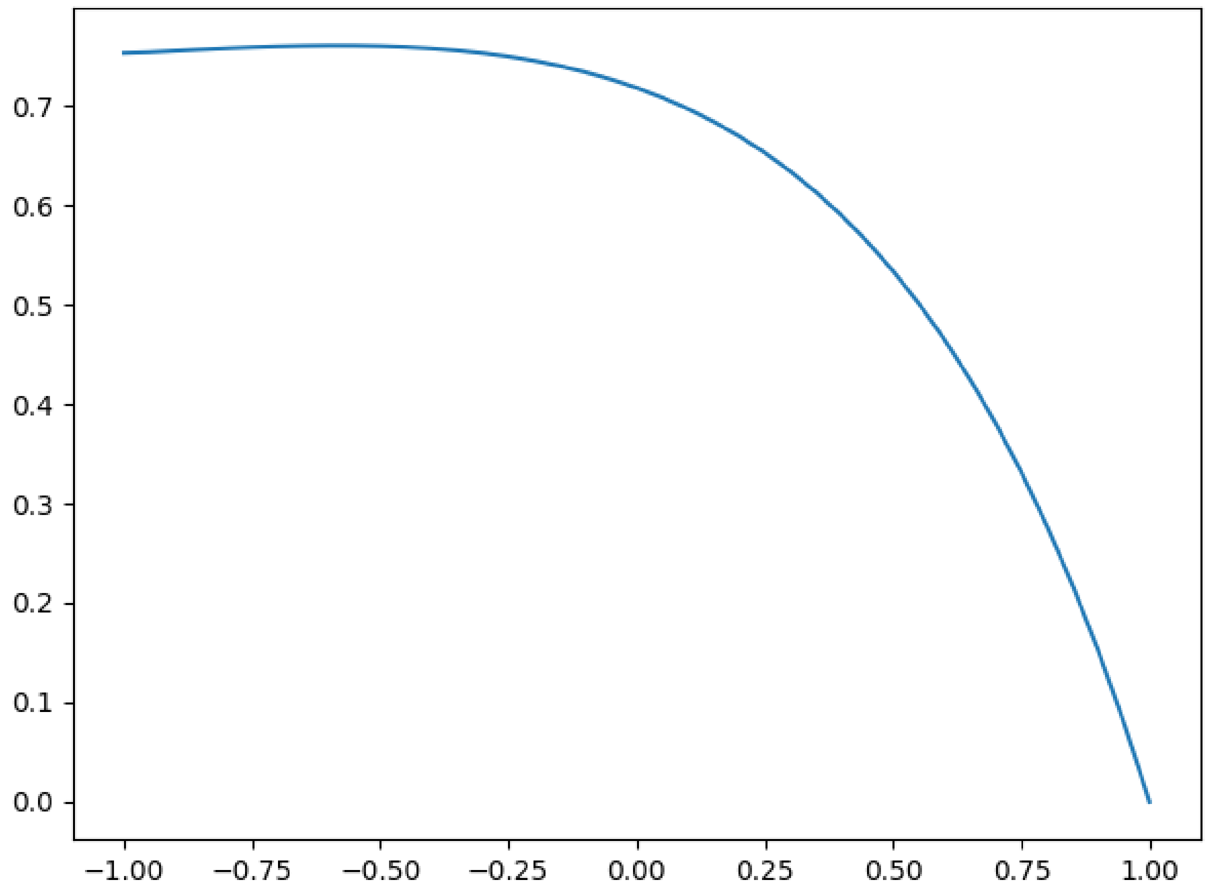

Рис. 1: График численного решения задачи при $h = 0.05$

Погрешность: 0.0185