Лабораторная работа №3

«Метод сеток решения граничной задачи для ОДУ»

Вариант 10

Выполнил студент 3 курса 2 группы ФПМИ

Сараев Владислав Максимович

Минск, 2020

Постановка задачи

Дана линейная граничная задача. Необходимо построить для граничной задачи разностную схему второго порядка аппроксимации на минимальном шаблоне и с помощью метода прогонки с шагом $h = 0.01$ найти ее численное решение. Проверить выполняются ли достаточные условия корректности и устойчивости прогонки. Сравнить найденное численное решение с точным решением $u(x) = \frac{1}{x+1}$. В одной системе координат построить график функции $u(x)$ и график полученного численного решения.

Граничная задача:

$$\begin{cases} u'' - 3(x+1)^2 u' - \dfrac{2}{(x+1)^2} u = 3 \\ u(0) = 1 \\ u(1) - 2u'(1) = 1 \end{cases}$$

Краткие теоретические сведения

Имеем разностную задачу

$$\begin{cases} u''(x) + p(x)u'(x) - q(x)u(x) = -f(x) \\ u(a) = \sigma_0 \\ u'(b) = -\sigma_1 u(b) + \mu_1 \end{cases}$$

Выбрав равномерную сетку узлов $\bar{\omega}_h$, разностное уравнение запишем в виде

$$y_{\bar{x}x} + p y_{\dot{x}} - qy = -f$$

Т.к. невязка разностного уравнения на решении $u(x)$ дифференциального уравнения равна $\psi_h(x) = y_{\bar{x}x} + p y_{\dot{x}} - qy + f = O(h^2)$, то разностное уравнение аппроксимирует дифференциальное со вторым порядком.

Рассмотрим второе граничное условие

$$u'(b) = -\sigma_1 u(b) + \mu_1$$

Разностное условие будем искать в виде

$$y_{\bar{x}} = -\bar{\sigma}_1 u(b) + \bar{\mu}_1$$

Сеточные коэффициенты $\bar{\sigma}_1$ и $\bar{\mu}_1$ введены, т.к. разностное граничное условие $y_{\bar{x}} = -\sigma_1 u(b) + \mu_1$ аппроксимирует дифференциальное $u'(b) = -\sigma_1 u(b) + \mu_1$ с первым порядком, а нам необходимо получить второй порядок аппроксимации.

Тогда

$$\nu_h(b) = u_{\bar{x}}(b) + \bar{\sigma}_1 u(b) - \bar{\mu}_1 =$$

$$= \left(-\left(\sigma_1 + \frac{p(b)\sigma_1 h}{2} + \frac{h}{2} q(b) \right) + \bar{\sigma}_1 \right) u(b)$$

$$+ \left(\left(\mu_1 + \frac{h}{2} p(b)\mu_1 + \frac{f(b)h}{2} \right) - \bar{\mu}_1 \right) + O(h^2)$$

Отсюда следует, что, выбрав

$$\bar{\sigma}_1 = \sigma_1 + \frac{p(b)\sigma_1 h}{2} + \frac{h}{2} q(b), \qquad \bar{\mu}_1 = \mu_1 + \frac{h}{2} p(b)\mu_1 + \frac{f(b)h}{2}$$

получим разностное граничное условие

$$y_{\bar{x}}(b) = -\left(\sigma_1 + \frac{p(b)\sigma_1 h}{2} + \frac{h}{2} q(b) \right) u(b) + \left(\mu_1 + \frac{h}{2} p(b)\mu_1 + \frac{f(b)h}{2} \right)$$

Получили схему второго порядка аппроксимации

$$\begin{cases} y_{\bar{x}x} + py_{\dot{x}} - qy = -f \\ y(a) = \sigma_0 \\ y_{\bar{x}}(b) = -\left(\sigma_1 + \dfrac{p(b)\sigma_1 h}{2} + \dfrac{h}{2}q(b)\right)u(b) + (\mu_1 + \dfrac{h}{2}p(b)\mu_1 + \dfrac{f(b)h}{2}) \end{cases}$$

Система в индексной форме имеет вид

$$\begin{cases} -y_{j-1}\left(\dfrac{1}{h^2} - \dfrac{p_j}{2h}\right) + y_j\left(\dfrac{2}{h^2} + q_j\right) - y_{j+1}\left(\dfrac{1}{h^2} + \dfrac{p_j}{2h}\right) = -f_j \\ y_0 = \sigma_0 \\ -\dfrac{y_{N-1}}{h} + y_N\left(\dfrac{1}{h} + \sigma_1 + \dfrac{p(b)\sigma_1 h}{2} + \dfrac{h}{2}q(b)\right) = \mu_1\left(1 + \dfrac{h}{2}p(b)\right) + \dfrac{f(b)h}{2} \end{cases}$$

Метод прогонки:

Имеем трехдиагональную систему

$$\begin{aligned} c_0 y_0 - b_0 y_1 &= f_0, \\ -a_1 y_0 + c_1 y_1 - b_1 y_2 &= f_1, \\ &\cdots\cdots\cdots\cdots \\ -a_i y_{i-1} + c_i y_i - b_i y_{i+1} &= f_i, \\ &\cdots\cdots\cdots\cdots \\ -a_N y_{N-1} + c_N y_N &= f_N. \end{aligned}$$

Верхняя двухдиагональная матрица

$$\begin{aligned} y_0 - \alpha_1 y_1 &= \beta_1, \\ y_1 - \alpha_2 y_2 &= \beta_2, \\ &\cdots\cdots\cdots\cdots \\ y_i - \alpha_{i+1} y_{i+1} &= \beta_{i+1}, \\ &\cdots\cdots\cdots\cdots \\ y_N &= \beta_{N+1}. \end{aligned}$$

Коэффициенты для прямой прогонки вычисляются по формулам:

$$\alpha_1 = \frac{b_0}{c_0},\ \beta_1 = \frac{f_0}{c_0},$$

$$\alpha_{i+1} = \frac{b_i}{c_i - a_i\alpha_i},\ \beta_{i+1} = \frac{f_i + a_i\beta_i}{c_i - a_i\alpha_i},\ i = 1, 2, \ldots, N-1,$$

$$\beta_{N+1} = \frac{f_N + a_N\beta_N}{c_N - a_N\alpha_N};$$

Для обратной $y_N = \beta_{N+1}$, $y_i = \alpha_{i+1}y_{i+1} + \beta_{i+1}$, $i = N-1, \ldots, 1, 0$

Если элементы a_{ij}, $1 \le i, j \le n$, некоторой матрицы A удовлетворяют условиям

$$|a_{ii}| \ge \sum_{j=1,\, j \ne i}^{n} |a_{ij}|, \; 1 \le i \le n$$

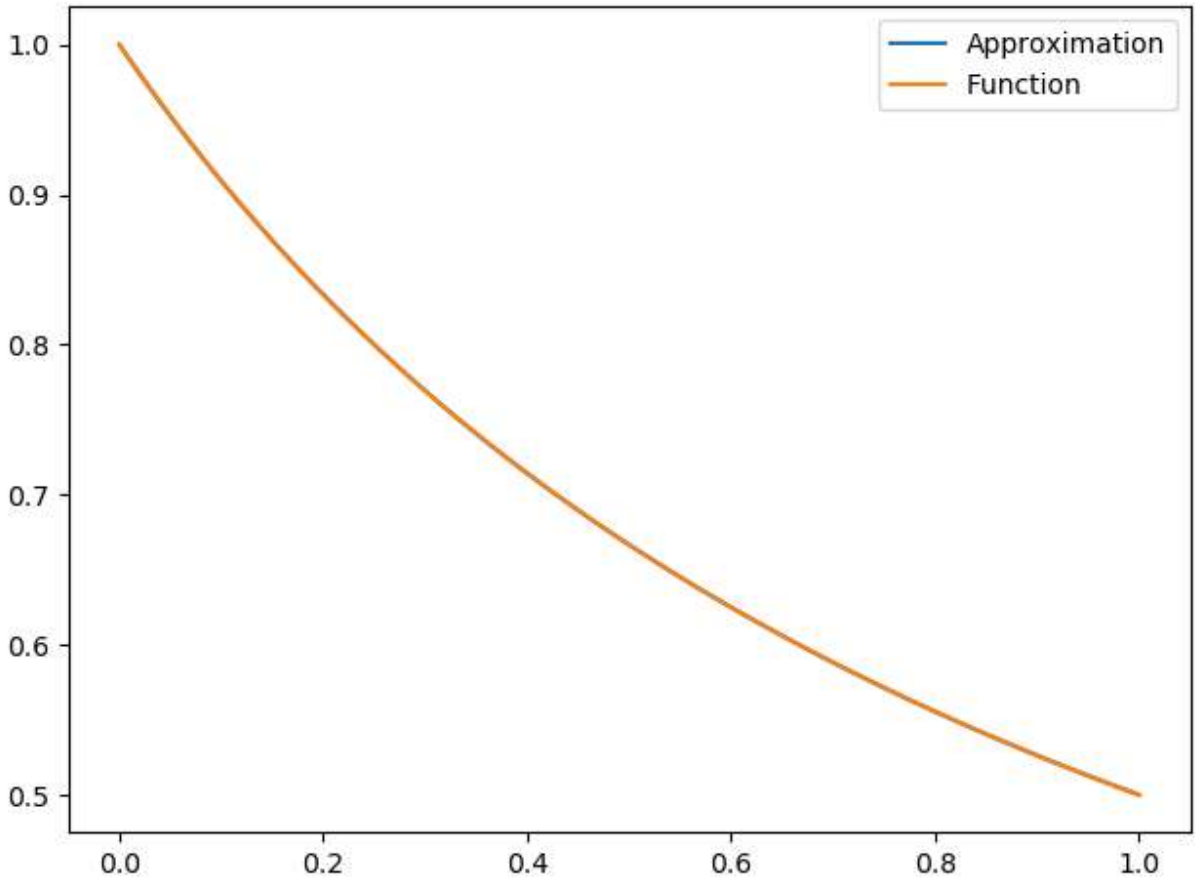
то говорят, что матрица A обладает свойством диагонального преобладания по строкам. Диагональное преобладание является достаточным условием корректности прогонки.

Листинг

```
# coding=utf-8
import numpy as np
from matplotlib import pyplot as plot
import time
from math import fabs


# Искомая функция
def u(x):
    return 1 / (x + 1)


# Функция при u'
def p(x):
    return -3 * (x + 1) ** 2


# Функция при u
def q(x):
    return 2 / (x + 1) ** 2


# Правая часть
def f(x):
    return -3


def check(matrix):
    for i in range(1, len(matrix[1]) - 1):
        if fabs(matrix[0][i - 1]) + fabs(matrix[2][i]) - fabs(matrix[1][i]) >
0:
            raise ValueError("Incorrect matrix")


def tridiagonal_matrix_algorithm(matrix, f):
    check(matrix)
    n = len(f)
    matrix[2][0] /= matrix[1][0]
    f[0] /= matrix[1][0]
    matrix[1][0] = 1
    for i in range(1, n - 1):
        coeff = matrix[1][i] - matrix[2][i - 1] * matrix[0][i - 1]
        matrix[2][i] /= coeff
        f[i] = (f[i] - f[i - 1] * matrix[0][i - 1]) / coeff
        matrix[1][i] = 1
        matrix[0][i - 1] = 0
    f[n - 1] = (f[n - 1] - f[n - 2] * matrix[0][n - 2]) / (matrix[1][n - 1] -
matrix[2][n - 2] * matrix[0][n - 2])
    matrix[1][n - 1] = 1

    for i in range(n - 2, -1, -1):
        f[i] -= f[i + 1] * matrix[2][i]
        matrix[2][i] = 0


# y(x0)
y0 = 1
# шаг
h = 0.01
# сигма 1
sigma1 = -0.5
# ню 1
```

```
nu1 = -0.5
# границы
L = 0
R = 1
# узлы
dots = np.linspace(L, R, int((R - L) / h) + 1)
# значения функций в узлах
q_values = [q(x) for x in dots]
p_values = [p(x) for x in dots]
f_values = [f(x) for x in dots]

# создание матрицы из трех диагоналях
matrix = np.array([np.zeros(len(dots) - 2), np.zeros(len(dots) - 1),
np.zeros(len(dots) - 2)])
# создание правой части
b = np.zeros(len(dots) - 1)

# заполнение матрицы и правой части
matrix[1][0] = -2 / (h ** 2) - q_values[1]
matrix[2][0] = 1 / (h ** 2) + p_values[1] / (2 * h)
b[0] = -f_values[1] - 1 / (h ** 2) + p_values[1] / (2 * h)

for i in range(1, len(b) - 1):
    matrix[0][i - 1] = 1 / (h ** 2) - p_values[i + 1] / (2 * h)
    matrix[1][i] = -2 / (h ** 2) - q_values[i + 1]
    matrix[2][i] = 1 / (h ** 2) + p_values[i + 1] / (2 * h)
    b[i] = -f_values[i + 1]

N = len(b) - 1
matrix[0][N - 1] = -1 / h
matrix[1][N] = 1 / h + sigma1 + h / 2 * p_values[N + 1] * sigma1 + h *
q_values[N + 1] / 2
b[N] = nu1 + h * f_values[N + 1] / 2 + h * p_values[N + 1] * nu1 / 2

try:
    # метод прогонки
    tridiagonal_matrix_algorithm(matrix, b)

    # полученное решение
    y = [y0]
    for i in b:
        y.append(i)
    u_values = [u(x) for x in dots]

    # вывод результата
    plot.plot(dots, y, label="Approximation")
    plot.plot(dots, u_values, label="Function")
    plot.legend()
    plot.show()
    print("Max delta: " + str(max([abs(y[i] - u_values[i]) for i in
range(len(y))])))
except ValueError as e:
    print(e.message)
```

Результаты



$\max_{j=0,N}(|u(t_j) - y_j|)$ = 2.3369184327104442e-05

Условия для корректности метода прогонки соблюдаются.

Выводы

При решении граничной задачи, с помощью разностных схем можно получить СЛАУ с трехдиагональной матрицей, которая быстро решается с помощью метода прогонки. Если использовать разностные схемы второго порядка, то полученное решения является достаточно точным.