Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА **ИУК4 «Программное обеспечение ЭВМ, информационные технологии»**

Лабораторная работа №2

«Численное решение стационарных задач теплопроводности»

ДИСЦИПЛИНА: «Моделирование»

Выполнил: студент гр. ИУК4-62Б ______________ (Калашников А.С.)
(подпись) (Ф.И.О.)

Проверил: ______________ (Никитенко У.В.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель работы: сформировать практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов применения метода конечных разностей.

Постановка задачи

Найти приближенное решение краевой задачи методом конечных разностей с заданной точностью tol и построить его график для:

$$\frac{-1}{x+3}u'' - xu' + ln(2+x)u = 1 - x/2,$$

$$u'(1) + 0.5u(1) = u'(-1),$$

$$u'(1) + 0.5u(1) = 0$$

ПОРЯДОК РЕШЕНИЯ

1. Используя встроенные функции/библиотеки PYTHON/MATLAB etc, получить “точное” решение задачи в узлах основной сетки, обозначим его Yex.

2. Составить разностную схему второго порядка точности.

3. Для реализации алгоритма метода прогонки следует создать модуль с процедурой, параметрами которой должны являться порядок системы, массивы коэффициентов системы уравнений и коэффициенты правой части.

4. Для вычисления решения задачи с заданной точностью произвести расчет с начальным шагом h, затем уменьшить шаг вдвое. Вывести на экран в виде таблицы два соседних приближенных решения и сравнить результаты.

5. Если заданная точность не достигнута, то продолжить уменьшение шага. Построить график найденного решения и указать шаг, при котором заданная точность достигается.

Результаты выполнения работы

Представим дифференциальное уравнение в виде $u'' = p(x)u' + q(x)u + r(x)$:

$$u'' = \frac{xu' + ln(2+x)u + 0.5x - 1}{\frac{-1}{x+3}}$$

$$= -(x+3)xu' - (x+3)ln(2+x)u - (x+3)(0.5x+1)$$

При помощи формул центральной разности. Тогда первая и вторая производные в точках заданной сетки примут вид:

$$u'' = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} = \frac{u_{k-1} - 2u_k + u_{k+1}}{h^2}$$

$$u' \approx \frac{u(x+h) - u(x-h)}{2h} = \frac{u_{k+1} - u_{k-1}}{2h}$$

Подставим их в исходное уравнение:

$$\frac{u_{k-1} - 2u_k + u_{k+1}}{h^2} = p\frac{u_{k+1} - u_{k-1}}{2h} + qu_k + r$$

Помножим обе части на h^2:

$$u_{k-1} - 2u_k + u_{k+1} = p * \frac{u_{k+1} - u_{k-1}}{2}h + qu_k h^2 + rh^2$$

$$\left(1 + \frac{p}{2}h^2\right)u_{k-1} - (2 + qh^2)u_k + \left(1 - \frac{p}{2}h^2\right)u_{k+1} \ = rh^2$$

или

$$a(x_k)u_{k-1} - b(x_k)u_k + c_k(x)u_{k+1} = rh^2$$

Выпишем граничные условия для точки $k = 0$:

$$u'(1) + 0.5u(1) = 0$$

Подставим приближённое значение:

$$0.5u_n - \frac{u_{n+1} - u_{n-1}}{2h} = 0$$

Так как значение $i = -1$ не представлено в сетке, выразим его через существующие узлы:

$$u_{n-1} = u_{n+1} + 2h(0 - 0.5u_n)$$

Подставим полученное значение в уравнение

$$\left(1 + \tfrac{p_n}{2}h^2 + 1 - \tfrac{p_n}{2}h^2\right)u_{n-1} + \left((2 + q_0h^2) - 2 * 0.5 * h\right)u_n = -r_nh^2 - 2 * (0) * h * \left(1 + \tfrac{p_n}{2}h^2\right)$$

или

$$(a_n + c_n)u_{n-1} + (b_n - 2h * 0.5)u_n = -r_nh^2 - 2(0)ha_n$$

По левой границе решим с помощью метода Неймана

$$p_1 u_0 + \ q_1 u_1 + r_1 u_2 = 2g(x_1)h^2$$

$$p_1(u_1 - \ u_a^{(1)} h) + \ q_1 u_1 + r_1 u_2 = 2g(x_1)h^2$$

$$u_1(p_1 + \ q_1) + r_1 u_2 = 2g(x_1)h^2 + p_1 u_a^{(1)} h$$

Составим систему из полученных уравнений:

$$-(a_0 + c_0)u_0 + (-b_0 - 2h * 0.5)u_1 = r_0 h^2 - 2(0)ha_0$$

$$a_1 u_0 - b_1 u_1 + c_1 u_2 = r_1 h^2$$

$$\dots$$

$$a_{n-1} u_{n-2} - b_{n-1} u_{n-1} + c_{n-1} u_n = r_{n-1} h^2$$

$$(a_n + c_n)u_{n-1} + (-b_n - 2h * 0.5)u_n = r_n h^2 - 2(0)ha_n$$

Таким образом, разностная схема примет вид:

$$\begin{pmatrix} -(a_0 + c_0) & (-b_0 - 2h * 0.5) & & & & \\ a_1 & b_1 & c_1 & & 0 & \\ & & & \dots & \dots & \\ & 0 & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & (a_n + c_n) & (b_n - 2h * 0.5) \end{pmatrix} \begin{pmatrix} u_0 \\ \dots \\ u_n \end{pmatrix} =$$

$$= \begin{pmatrix} -r_0 h^2 - 2(0)ha_0 \\ r_1 h^2 \\ \dots \\ r_{n-1} h^2 \\ r_n h^2 - 2(0)ha_n \end{pmatrix}$$

Для «точного» решения воспользуемся методом solve_bvp библиотеки scipy. Уменьшая каждый раз шаг в 2 раза, начиная с шага $h = 0.05$, получим следующее решение (см. рис. 1).

Для полученного решения построим график, а также график решения, полученного встроенным методом (см. рис 3).Значение, удовлетворяющее точности $tol = 0.001$, было получено при шаге $h = 9.765 * 10^{-6}$.

Шаг	Погрешность
0.0500000000	0.85600
0.0250000000	0.46200
0.0125000000	0.24000
0.0062500000	0.12300
0.0031250000	0.06200
0.0015625000	0.03100
0.0007812500	0.01600
0.0003906250	0.00800

Рисунок 1 – Вывод таблицы

```
Точное значение -1:          2.9275108886203056
Полученное значение -1:      2.9196962819673296

Точное значение 1:           0.5141924214519076
Полученное значение 1:       0.5141948874463271

Точность:                    0.01
Шаг:                         0.000390625
```

Рисунок 2 – Вывод значений

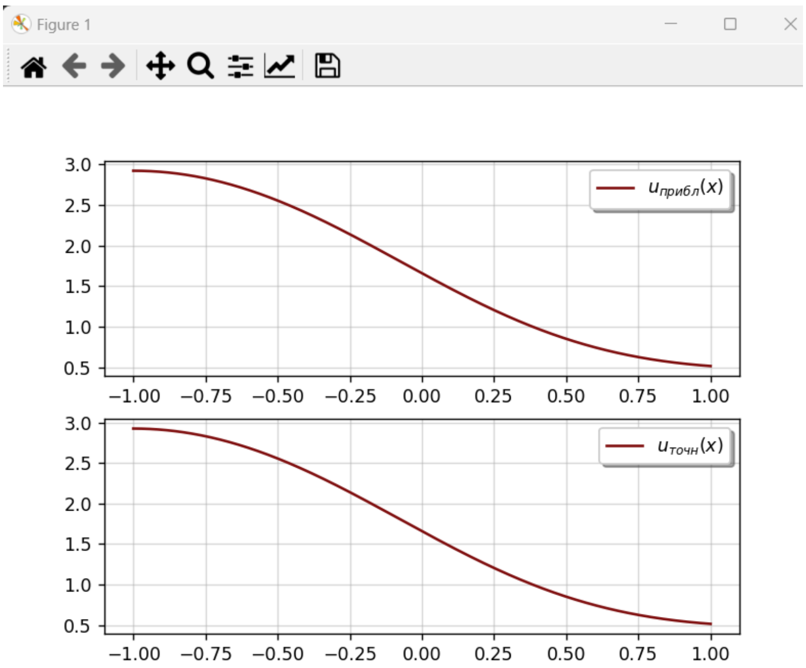

Рисунок 3 – Вывод графика

Вывод: в ходе выполнения работы были сформированы практические навыки анализа моделей с использованием специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов применения метода конечных разностей.

ПРИЛОЖЕНИЯ

Листинг программы

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_bvp, solve_ivp
import scipy.linalg as la

def g(x):
    return -(1)/(x+3)

def task(a2, a1, a0, g, robin_A, robin_B, rng, h):
    n = int((rng[1]-rng[0])/h) + 1

    pf = lambda x: 1 - a1(x)/a2(x) * h / 2
    qf = lambda x: -(2 - a0(x)/a2(x) * h**2)
    rf = lambda x: 1 + a1(x)/a2(x) * h / 2
    df = lambda x: g(x)/a2(x) * h**2

    xs = np.linspace(rng[0], rng[1], n)
    A = np.zeros((n, n))
    B = np.zeros(n)

    A[n-1, n-2] = robin_A[1, 1] * (pf(xs[n-1]) + rf(xs[n-1]))
    A[n-1, n-1] = robin_A[1, 1] * qf(xs[n-1]) - 2 * h * robin_A[1, 0]
    B[n-1] = df(xs[n-1]) * robin_A[1, 1] - 2 * h * robin_B[1] * pf(xs[n-1])

    #xn = xs[n-1]
    #A[n-1, n-1] = robin_A[1, 1] * qf(xn) - 2 * h * robin_A[1, 0]
    #A[n-1, n-2] = robin_A[1, 1] * (pf(xn) + rf(xn))
    #B[n-1] = df(xn) * robin_A[1, 1] - 2 * h * robin_B[1] * pf(xn)

    #pf = lambda x: 2*a2(x) - a1(x)*h
    #qf = lambda x: -4*a2(x) + 2*a0(x)*h**2
    #rf = lambda x: 2*a2(x) + a1(x)*h
    #df = lambda x: 2*g(x) * h**2

    A[0, 1] = qf(xs[0]) + rf(xs[0])
    A[0, 0] = pf(xs[0])
    B[0] = df(xs[0]) - rf(xs[0])*0*h

    for i in range(1, n - 1):
         A[i, i - 1] = pf(xs[i])
         A[i, i] = qf(xs[i])
         A[i, i + 1] = rf(xs[i])
         B[i] = df(xs[i])

    u = np.linalg.solve(A, B)

    return xs, u
```

```
def exact_task(rng):
    a = -1
    b = 1

    def fun(x, y):
        return np.vstack((y[1], (x * y[1] - np.log(2+x) * y[0] - (x/2) +
1)/g(x)))

    def bc(ya, yb):
        return np.array([ya[1], 1*yb[1] + 0.5*yb[0]])


    res = solve_bvp(fun, bc, [a, b], [[-1, -1], [1, 1]], tol=1e-9)

    return res.x, res.y[0], res

if __name__ == "__main__":
    h = 0.1
    rng = [-1, 1]

    tol = 1e-2
    cnt = 3
    err = tol + 1

    x_ex, y_ex, res = exact_task(rng)

    x, y = task(a2 = lambda x: -1/(x+3),
            a1 = lambda x: -1 * x,
            a0 = lambda x: np.log(2+x),
            g  = lambda x: 1-(x/2),
            robin_A = np.array([[-1, 1], [0.5, 1]]),
            robin_B = np.array([0, 0]),
            rng=rng,
            h=h)


    print(f"|{'-'*30}-{'-   '*30}|")
    print(f"|{'Шаг':>29} |{'Погрешность':>29} |")
    print(f"|{'-'*30}|{'-'*30}|")

    while err >= tol:
        x, y = task(a2 = lambda x: -1/(x+3),
            a1 = lambda x: -1 * x,
            a0 = lambda x: np.log(2+x),
            g  = lambda x: 1-(x/2),
            robin_A = np.array([[-1, 1], [0.5, 1]]),
            robin_B = np.array([0, 0]),
            rng=rng,
            h=h)
        err = round(max(abs(np.array([res.sol(i)[0] for i in x]) - y)), cnt)
        h /= 2
        print(f'|{round(h, 10):29.10f} |{err:29.5f} |')

    print(f"|{'-'*30}_{'-'*30}|")

    print(f'\n{f"Точное значение {rng[0]}:":40} {res.sol(rng[0])[0]}')
    print(f'{f"Полученное значение {rng[0]}:":40} {y[0]}')
```

```
print(f'\n{f"Точное значение {rng[1]}:":40} {res.sol(rng[1])[0]}')
print(f'{f"Полученное значение {rng[1]}:":40} {y[-1]}')

print(f'\n{f"Точность:":40} {tol}')
print(f'{f"Шаг:":40} {h}')

plt.subplot(2, 1, 1)
plt.plot(x, y, color='#FF1010', label='$u_{прибл}(x)$')
plt.grid(alpha=0.5)
plt.legend(framealpha=1, shadow=True)

plt.subplot(2, 1, 2)
plt.plot(res.x, res.y[0], color='#FF1010', label='$u_{точн}(x)$')
plt.grid(alpha=0.5)
plt.legend(framealpha=1, shadow=True)
plt.show()
```