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 ________________ (Карельский М.К.)
(Подпись)

Проверил: ________________ (Никитенко У.В.)
(Подпись)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2023

Цель: 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.

Задачи: построить разностные схемы для предложенного уравнения. Оценить точность аппроксимации. Оценить устойчивость и сходимость. Выбрать среду для проведения расчетов и вычислительного эксперимента. Написать программу, реализующую решение разностной задачи. Оценить результаты расчетов. Визуализировать результаты.

Задание:

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

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.

Вариант 5

$$-\frac{6+x}{7+3x}u'' - \left(1-\frac{x}{2}\right)u' + \left(1+\frac{1}{2}\cos(x)\right)u = 1-\frac{x}{3}$$
$$u'(-1) - 2u(-1) = 0$$
$$u'(1) = 0$$

Решение:

$$a_2(x) = -\frac{6+x}{7+3x}$$
$$a_1(x) = -\left(1-\frac{x}{2}\right)$$
$$a_0(x) = 1+\frac{1}{2}\cos(x)$$

$$g(x) = 1 - \frac{x}{3}$$
$$a_2(x)u'' + a_1(x)u' + a_0(x)u = g(x)$$

Имея шаг h, найдем p, q, r, d:

$$p(x) = 1 - \frac{a_1(x)h}{2a_2(x)}$$
$$q(x) = -\left(2 - \frac{a_0(x)h^2}{a_2(x)}\right)$$
$$r(x) = 1 + \frac{a_1(x)h}{2a_2(x)}$$
$$d(x) = \frac{g(x)h}{2a_2(x)}$$

Запишем коэффициенты граничных условий в форме Робина и Неймана:

$$r = (-2 \quad 1 \quad 0)$$
$$n = 0$$

Зададим матрицы A и B:
Из граничного условия в форме Робина:

$$A_{0,0} = r_1\big(p(x_0) + r(x_0)\big)$$
$$A_{0,1} = r_1 q(x_0) - 2hr_0$$
$$B_0 = d(x_0)r_1 - 2hr_2 p(x_0)$$

Из граничного условия в форме Неймана:

$$A_{n-1,n-1} = q(x_{n-1}) + r(x_{n-1})$$
$$A_{n-1,n-2} = p(x_{n-1})$$
$$B_{n-1} = d(x_{n-1}) - r(x_{n-1})nh$$

Оставшиеся значения при $i \in [1, n-1)$:

$$A_{i,i-1} = p(x_i)$$
$$A_{i,i} = q(x_i)$$
$$A_{i,i+1} = r(x_i)$$
$$B_i = d(x_i)$$

Производим расчет решения:

Шаг	Погрешность
0.0050000000	0.00700
0.0025000000	0.00300
0.0012500000	0.00200
0.0006250000	0.00100
0.0003125000	0.00000

```
Точное значение в точке -1:        0.3468395787522008
Полученное значение в точке -1:    0.3464309899915504

Точное значение в точке 1:         0.5661147770977418
Полученное значение в точке 1:     0.5660667593683236

Точность:                          0.001
Значение шага:                     0.0003125
```

Рис. 1. Решение

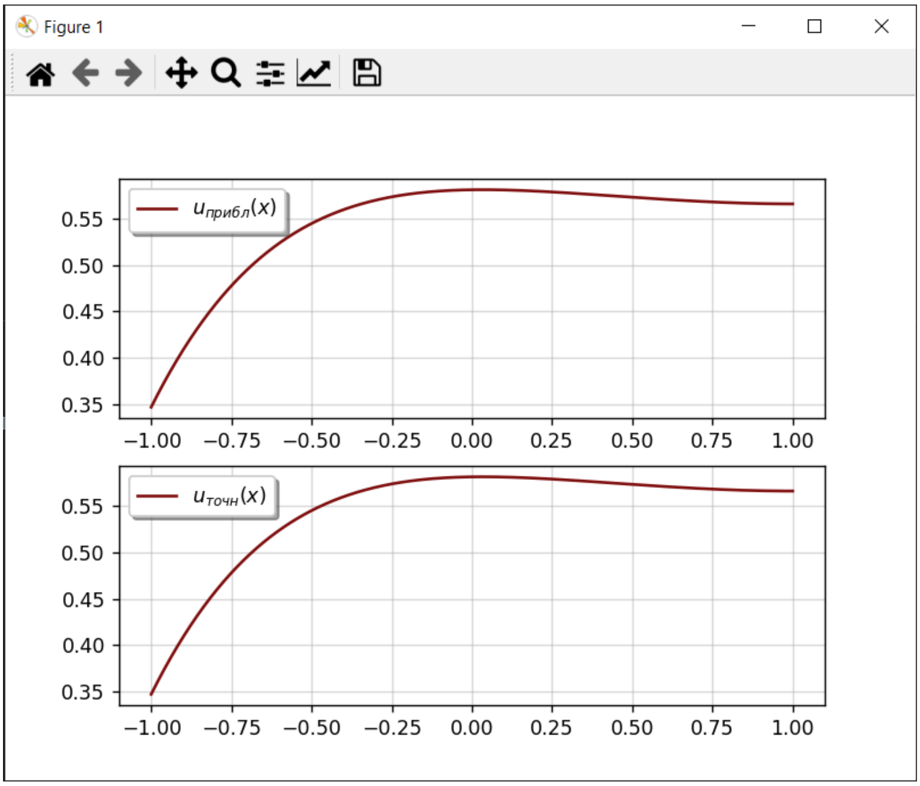

Рис. 2. Полученные графики

Вывод: в ходе выполнения лабораторной работы были получены 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.

ПРИЛОЖЕНИЯ

Листинг:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_bvp

def task(a2, a1, a0, g, robin_A, robin_B, neumann, 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[0, 0] = robin_A[1] * (pf(xs[0]) + rf(xs[0]))
    A[0, 1] = robin_A[1] * qf(xs[0]) - 2 * h * robin_A[0]
    B[0] = df(xs[0]) * robin_A[1] - 2 * h * robin_B * pf(xs[0])

    A[n-1, n-1] = qf(xs[n-1]) + rf(xs[n-1])
    A[n-1, n-2] = pf(xs[n-1])
    B[n-1] = df(xs[n-1]) - rf(xs[n-1])*neumann*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 g(x):
        return -(6 + x)/(7 + 3*x)

    def fun(x, y):
        return np.vstack((y[1], ((1 - x/2)*y[1] - (1 + 0.5*np.cos(x))*y[0] + (1
- x/3))/g(x)))

    def bc(ya, yb):
        return np.array([ya[1] - 2*ya[0], yb[1]])
```

```
        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 = 0.01
        rng = [-1, 1]

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

        x_ex, y_ex, res = exact_task(rng)

        x, y = task(a2 = lambda x: -(6 + x)/(7 + 3*x),
                    a1 = lambda x: -(1-x/2),
                    a0 = lambda x: 1 + 0.5*np.cos(x),
                    g  = lambda x: 1 - x/3,
                    robin_A = np.array([-2, 1]),
                    robin_B = 0,
                    neumann = 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: -(6 + x)/(7 + 3*x),
                    a1 = lambda x: -(1-x/2),
                    a0 = lambda x: 1 + 0.5*np.cos(x),
                    g  = lambda x: 1 - x/3,
                    robin_A = np.array([-2, 1]),
                    robin_B = 0,
                    neumann = 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 в точке {rng[0]}:":40} {res.sol(rng[0])[0]}')
        print(f'{f"Полученное значение в точке {rng[0]}:":40} {y[0]}')

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

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

```
print(f'{f"Значение шага:":40} {h}')

plt.subplot(2, 1, 1)
plt.plot(x, y, color='#801010', 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='#801010', label='$u_{точн}(x)$')
plt.grid(alpha=0.5)
plt.legend(framealpha=1, shadow=True)
plt.show()
```