|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 c 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**

**Решение:**

Имея шаг h, найдем p, q, r, d:

Запишем коэффициенты граничных условий в форме Робина и Неймана:

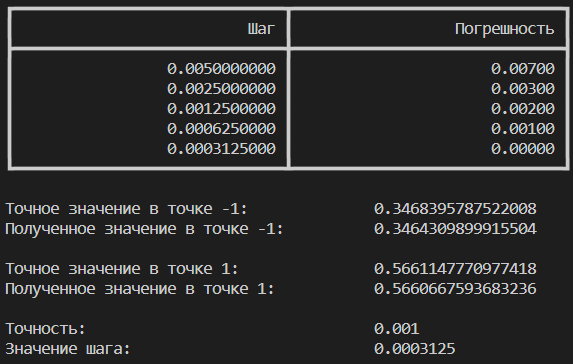
Зададим матрицы A и B:

Из граничного условия в форме Робина:

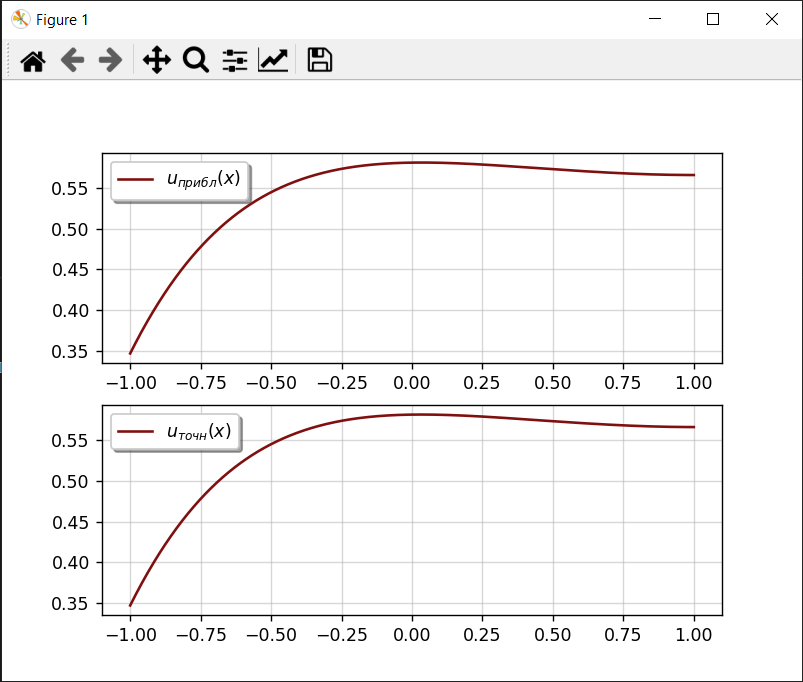
Из граничного условия в форме Неймана:

Оставшиеся значения при :

Производим расчет решения:



**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()