ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

им. А. Н. ТИХОНОВА

Труханов Александр Ильич
БИВ172

ЛАБОРАТОРНАЯ РАБОТА №1
По курсу «Математический компьютерный практикум»
по направлению 09.03.01 Информатика и вычислительная техника
студента образовательной программы бакалавриата
«Информатика и вычислительная техника»

Руководитель:
Круглик Станислав Александрович

Москва 2019 г.

Оглавление

1. Исходные данные варианта

Найти $u(0.5, 0.5)$ путем численного решения уравнения теплопроводности $u_t = u_{xx} + tsh(x)$ (0), $u(0, x) = 0, u(t, 0) = 2t, u(t, 1) = 3t^2$ в области $(t, x) \in [0, 0.5] \times [0, 1]$ с помощью неявной разностной схемы.

2. Решение задачи

2.1. Аналитическое решение

На данный момент у несчастью не удалось получить аналитическое решение данного уравнения.

2.2. Численное решение

Для получения численного решения был применена неявная разностная схема. Опишем алгоритм применения разностной схемы для данного уравнения.

Зададим сетку на области $(t, x) \in [0, 0.5] \times [0, 1]$ с шагом h_x по оси x и шагом h_t по оси t.

Тогда:

$$x_i = x_0 + h_x(i - 1), x_0 = 0 \ (1)$$
$$t_j = t_0 + h_t(j - 1),\ t_0 = 0 \ (2)$$
$$u_i^j = u(x_i, t_j) \ (3)$$
$$f_i^j = t_j sh(x_i) \ (4)$$

Используя уравнения (1), (2), (3) и (4) перпешем уравнение (0):

$$\frac{u_i^{j+1} - u_i^j}{h_t} = \frac{u_{i-1}^{j+1} - 2u_i^{j+1} + u_{i+1}^{j+1}}{{h_x}^2} + f_i^{j+1} \ (5)$$

Преобразуем уравнение (5) в следующий вид:

$$A_i u_{i-1}^{j+1} + B_i u_i^{j+1} + C_i u_{i+1}^{j+1} = F_i \ (6)$$
$$A_i = \frac{1}{{h_x}^2}$$
$$B_i = -\frac{2}{{h_x}^2} - \frac{1}{h_t}$$
$$C_i = \frac{1}{{h_x}^2}$$
$$F_i = -\frac{u_i^j}{h_t} - f_i^{j+1}$$

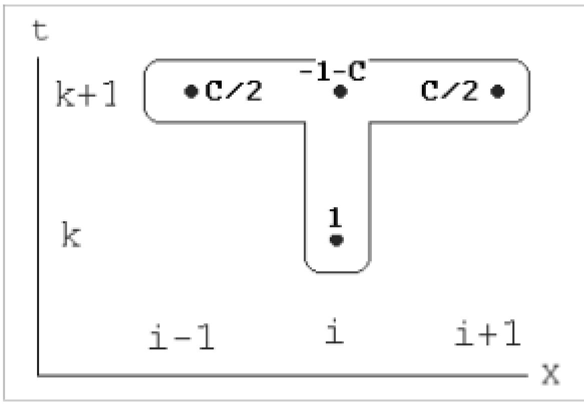

Рис. 1. Шаблон неявной схемы

Пусть максимальное значение i – i_{max}, а максимально значение j – j_{max}.

У нас известны значения функции во всех точках по краям (кроме $u(0.5, x)$ нашей области $(t, x) \in [0, 0.5] \times [0, 1]$ $(u(0, x) = 0, u(t, 0) = 2t, u(t, 1) = 3t^2)$, а поскольку для всех внутренних точек справедливо уравнение (6), то записав его для каждого слоя $j + 1$, мы получим систему из $i_{max} - 2$ уравнений с $i_{max} - 2$ неизвестными (u_1^{j+1} и $u_{i_{max}}^{j+1}$) уже заранее известны из граничных условий. Решая данную систему уравнений, мы получим u_i^{j+1}, где $i = \overline{2, i_{max} - 1}$, а поскольку u_1^{j+1} и $u_{i_{max}}^{j+1}$ нам уже заранее известных из граничных условий, мы вычислили все значения функции на слое $j + 1$. Последовательно повторяя это для всех слоев начиная со второго слоя (поскольку первый уже известен из граничного условия $u(0, x) = 0$), мы получим значения функции во всех точках сетки.

2.3. Листинг решения

Численное решение, а также построение всех необходмых графиков было релазиовано с помощью Python и сторонних библиотек Numpy и Matplotlib.

Листинг фнукций, реализующих аналитическое решение неявную разностную схему (файл TCLab.py):

```
import numpy as np



def solve_analytically(t, x, n=500):
    """
    Возврщает значение функции теплопроводности, заданной ДУ:
    du/dt = d2u / dx2 + t*sh(x)
    С начальными условиями:
    u(0, x) = 0
    u(t, 0) = 2*t
    u(t, 1) = 3*t^2
    В области (t, x) с [0; 0.5] x [0; 1]

    :param t:
    :param x:
```

```
        :param n: количество членов в разложении
        :return: значение функции в точке (t, x)
        """
        U = (3 * t ** 2 - 2 * t) * x + 2 * t

        a = lambda n: 2 * (-1) ** (n + 1) * np.pi * n * np.sinh(1) / (1 + (np.pi * n) ** 2)
+ \
                      12 * (-1) ** n / np.pi / n * (1 - 1 / np.pi / n) + 12 / (np.pi * n) **
2
        b = lambda n: 4 / (np.pi * n) ** 2 * ((-1) ** n - 1) - 4 / np.pi / n
        Cn = lambda n: a(n) * t + (b(n) - a(n) / (np.pi * n) ** 2) * \
                       (1 - np.exp(-(np.pi * n) ** 2 * t) / (np.pi * n) ** 2)
        v = 0
        for i in range(1, n + 1):
            v += Cn(i) * np.sin(np.pi * i * x)

        return U + v



    def solve_implicit_schema(x: tuple, t: tuple, hx: float, ht: float):
        """
        Принимает на вход границы области, на которой необходимо найти решение
        уравнения теплопроводности, создает сетку, решает уравнение с помощью
        неявной разностной схемы и возвращает матрицу значений функций в узлах
        сетки.

        :param x: границы области по координате x
        :param t: границы области по координате t
        :param hx: шаг сетки по координате x
        :param ht: шаг сетки по координате t
        :return: двумерный массив u[t, x]
        """
        x0, xmax = x
        t0, tmax = t

        nmax, kmax = int((tmax - t0) / ht) + 1, int((xmax - x0) / hx) + 1

        u = np.zeros((nmax, kmax))
```

```
    u[0, :] = 0
    u[:, 0] = np.linspace(t0, tmax, nmax).T * 2
    u[:, -1] = np.linspace(t0, tmax, nmax).T ** 2 * 3

    f = lambda k, n: (t0 + (k-1) * ht) * np.sinh(x0 + (k-1) * hx)

    for n in range(1, nmax):
        A = np.eye(kmax - 2, k=-1) / hx ** 2
        B = np.eye(kmax - 2) * (-2 / hx ** 2 - 1 / ht)
        C = np.eye(kmax - 2, k=1) / hx ** 2
        equations_koef = A + B + C

        F = np.zeros(kmax - 2)
        F[0] = -u[n - 1, 1] / ht - f(1, n) - u[n, 0] / hx ** 2
        F[kmax - 3] = -u[n - 1, kmax - 2] / ht - f(kmax - 2, n) - u[n, kmax - 1] / hx **
2
        for k in range(1, kmax - 3):
            F[k] = -u[n - 1, k + 1] / ht - f(k + 1, n)
        ucurr = np.linalg.solve(equations_koef, F.T)
        u[n, 1:-1] = ucurr.T
    return u


def get_u_xt(u, xborders: tuple, tborders: tuple, x: float, t: float):
    """
    Примимает на вход матрицу значений функции в узлах сетки, границы
    области решения и координаты точки, значение функции в которой
    нас интересует. Возвращает значение функции в интересующей нас точке.

    :param u: матрица значений функции в узлах сетки
    :param xborders: границы области решения по координате x
    :param tborders: границы области решения по координате t
    :param x: координата x
    :param t: координата t
    :return: значение функции u(x, t)
    """
    x0, xmax = xborders
    t0, tmax = tborders
```

```
        hx, ht = (xmax - x0) / (u.shape[1] - 1), (tmax - t0) / (u.shape[0] - 1)
        i, j = int(np.floor((x - x0) / hx)), int(np.floor((t - t0) / ht))
        return u[j, i]
```

Листинг скрипта, запускащего расчет аналитического и численного решений, а также построение необходимых графиков (файл LabScript.py):

```
import numpy as np
import matplotlib.pyplot as plt
import TCLab


x = (0, 1)
t = (0, 0.5)
hx = 0.01
ht = 0.01


# x_test = 0.7
tt = np.arange(t[0], t[1] + ht, ht)
u_analytic = np.array([TCLab.solve_analytically(tt, x_test) for x_test in np.arange(x[0], x[1]
+ hx, hx)]).T
u_euler_implicit = TCLab.solve_implicit_schema(x, t, hx, ht)


print('Implicit:', TCLab.get_u_xt(u_euler_implicit, x, t, 0.5, 0.5))
print('Analytic:', TCLab.get_u_xt(u_analytic, x, t, 0.5, 0.5))


plt.imshow(u_euler_implicit)
plt.colorbar()


plt.xlabel(u'x')
plt.ylabel(u't')
plt.title(u'Уравнение теплопроводности численное решение')
plt.grid(True)
plt.show()


plt.imshow(u_analytic)
plt.colorbar()


plt.xlabel(u'x')
plt.ylabel(u't')
plt.title(u'Уравнение теплопроводности аналитическое решение')
```

```
plt.grid(True)

plt.show()
```

3. Полученные результаты

По результатам работы было получено 2 графика:

- Численное решение.
- Зависимость значения u(0.5, 0.5) от шага сетки.

$$h_x = 0.01, h_t = 0.01$$

$$u(0.5, 0.5) = 0.641699$$

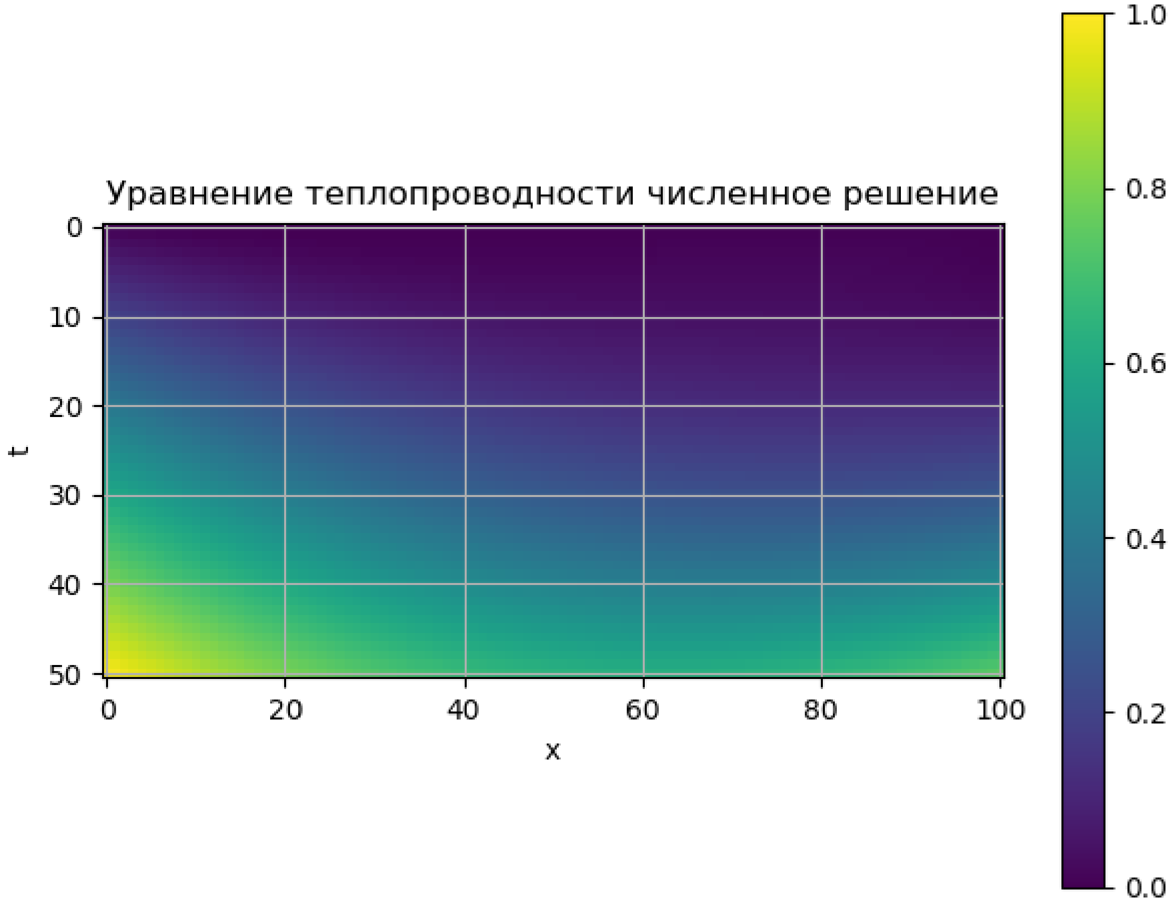

Рис. 2. Численное решение

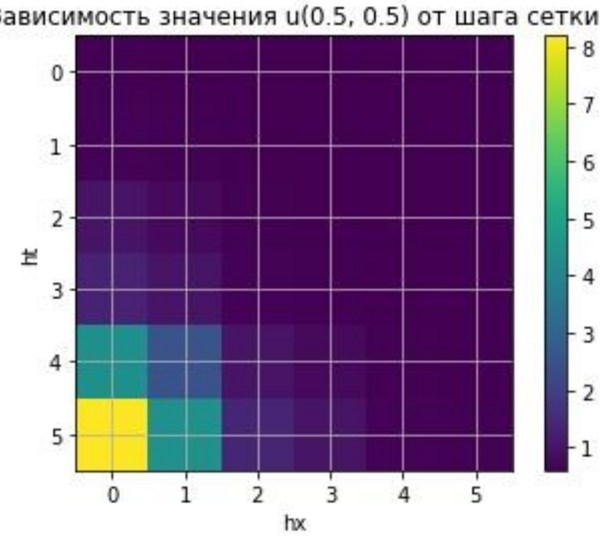

Рис. 3. Зависимость значения функции u(0.5, 0.5) от шага сетки

Рис. 4. Зависимость значения функции u(0.5, 0.5) от h_x, $h_t = 0.01$

Рис. 5. Зависимость значение функции u(0.5, 0.5) от h_t, $h_x = 0.01$