МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
Высшего образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Институт информационных технологий, математики и механики Кафедра дифференциальных уравнений, математического и численного анализа

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

«Численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных»

Выполнил: студент группы 381706-2
Тихомирова М. А.
_________________________ Подпись

Научный руководитель:
Старший преподаватель каф. ДУМЧА
______________________ Эгамов А. И.

Нижний Новгород
2020.

Содержание

1. Введение

Рассмотрим следующую задачу: дан некоторый тонкий однородный стержень с теплоизолированными концами длины l, имеющий некоторую начальную температуру, значения которой в конкретных точках стержня задаются следующей функцией:

$$\boxed{\varphi(x) = \frac{1}{l} + \varphi_1 \cos\left(\frac{\pi x}{l}\right) + \varphi_2 \cos\left(\frac{2\pi x}{l}\right),}$$

где φ_1 и φ_2 – некоторые константы. На процесс изменения температуры стержня осуществляется некое воздействие, например, через стержень пропускается электрический ток или он помещается в электромагнитное поле и т.п. В нашем случае, на изменение температуры будет влиять следующая управляющая функция:

$$\boxed{b(x) = b_0 + b_1\, cos\left(\frac{\pi x}{l}\right) + b_2\, cos\left(\frac{2\pi x}{l}\right),}$$

где b_0, b_1, b_2 – некоторые константы. В данной лабораторной работе будет рассмотрена математическая модель этого процесса. А именно, реализована программа, получающая на вход параметры модели и параметры расчета и реализующая расчет и отрисовку полученных результатов на графике.

2. Постановка задачи

Определим начальные условия:

На множестве $Q = [0, l] \times [0, T]$, $l > 0$, $T > 0$, найти функцию $y(x, t)$ – температуру стержня – непрерывно дифференцируемую по t и дважды непрерывно дифференцируемую по x - решением уравнения:

$$\boxed{y'_t(x,t) = a^2 y''_{xx}(x,t) + u(x,t)} \quad (1)$$

удовлетворяющее (концы тепло-изолированы) однородным граничным условиям второго рода:

$$\boxed{y'_x(0,t) = y'_x(l,t) = 0} \quad (2)$$

и начальному условию

$$\boxed{y(x, 0) = \varphi(x)} \quad (3)$$

где а – константа, $\varphi(x) > 0$, задает начальное распределение температуры, дважды непрерывно дифференцируема на отрезке [0, l], удовлетворяет условиям согласования (3) и условию

$$\boxed{\int_0^l \varphi(x)\,dx = 1} \quad (4)$$

Непрерывная функция $u(x, t)$ – управление с обратной связью, которое представляется в одном из вариантов:

$$\boxed{u(x,t) = b(x)y(x,t)} \quad (5)$$

$$\boxed{u(x,t) = b(x)y(x,t) - y(x,t)\int_0^l b(x)y(x,t)\,dx} \quad (6)$$

Наложение условия дважды непрерывной дифференцируемости функции на отрезке [0, l] объясняется следующим. Во-первых, чтобы решать поставленную задачу одним из самых известных методов – методом Фурье и, чтобы полученный ряд – решение задачи – можно было дифференцировать по t и дважды дифференцировать по x. А во-вторых, как правило, элементарные функции обладают свойством бесконечного дифференцирования, хотя, конечно, для существования решения исходных задач условия на функцию $\varphi(x)$ можно ослабить.

Приведем необходимые для решения теоремы, для определенности обозначим решение: $w(x, t)$

Теорема 1. Задача (1) - (4) при $u(x, t) \equiv 0$, имеет решение:

$$w_0(x,t) = \sum_{i=0}^{+\infty} \varphi_i \exp(-\lambda_i^2 t)\, \nu_i(x),$$

где собственные числа λ_i и полная, ортогональная на отрезке [0, l] система косинусов [2] $\nu_i(x)$, удовлетворяющая условию (2):

$$\lambda_0 = 0, \nu_0(x) = 1;\ \lambda_i = \frac{\pi i}{l};\ \nu_i(x) = \cos\frac{\pi i x}{l} \equiv \cos\lambda_i x, i = \overline{1, +\infty}$$

Доказательство: [2].

Теорема 2. Решение нелинейной задачи (1) – (4), (6) выражается через решение линейной задачи (1) – (5) функцию $w(x, t)$:

$$y(x,t) = \frac{w(x,t)}{\int_0^l w(x,t)\,dx}$$

Доказательство: [2].

В данной работе необходимо выполнить следующие задачи:

1. Написать программу на языке программирования высокого уровня с дружеским интерфейсом для численного решения задачи (1) - (3) и вывода функции $y(x, T)$ в графическом виде.

2. На одном и том же рисунке вывести оси координат, график функции $\varphi(x)$ – синим цветом, $y(x, T)$ – красным цветом.

3. Учесть в оконном меню программы возможность изменения:
   1. длины стержня l; времени T;

2. шага h в разностной схеме по координате x, шага τ в разностной схеме по координате t;

3. константы b_0, b_1, b_2, φ_0, φ_1.

4. Вывести строку прогресса.
5. Полученную функцию $w(x, T)$ нужно разделить на $I = \int_0^l w(x,T)dx$, который

нужно посчитать по формуле Симпсона для значений последнего слоя (при t = T), и вывести полученный график функции:

$$\boxed{\frac{w(x,T)}{\int_0^l w(x,T)dx}}$$

на экран светло-зеленым цветом.

6. Поскольку в идеале красный и зеленый график должны совпадать, желательно сделать так, чтобы зеленый график выводился на экран только при дополнительном нажатии «горячей клавиши», например, «пробел».

3. Руководство пользователя

При запуске программы пользователю необходимо:

- Задать параметры модели – длина стержня, время, шаг по x, шаг по времени.
- Задать константы φ_0, φ_1, b_0, b_1, b_2.

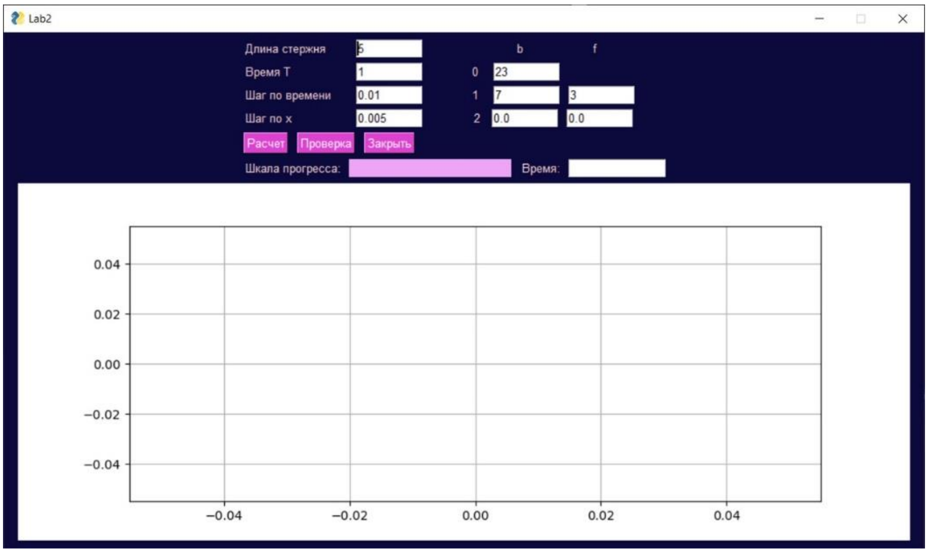

Рисунок 1. Ввод данных.

- Нажать «Расчет». При нажатии на кнопку выведется график φ(x) синим цветом и y(x,T) – красным цветом. При расчете будет заполняться шкала прогресса, а после вывеется время.

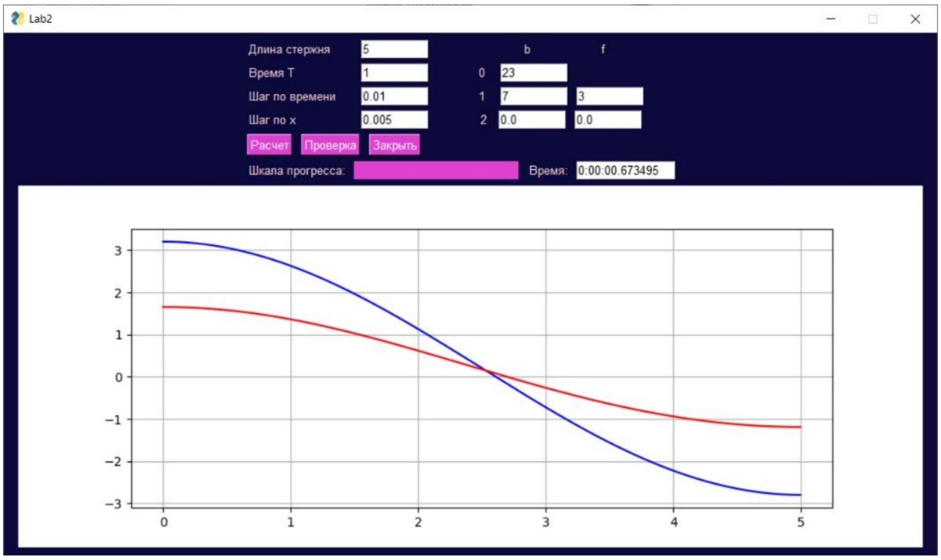

Рисунок 2. При нажатии кнопку "Расчет" на график выводится решение линейной задачи

- При нажатии на кнопку «Проверка» выведется график w(x,T) / I. График будет выведен зеленым цветом.

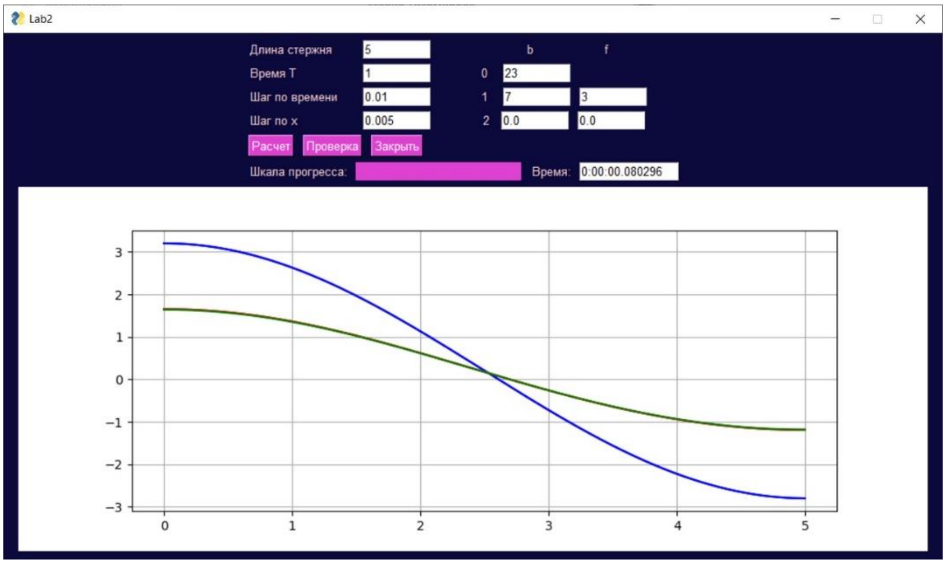

Рисунок 3. При нажатии кнопку "Проверка".

4. Описание алгоритмов

Алгоритм решения задачи:

1. Определить нулевой слой для будущей разностной схемы из:

$$\boxed{y(x,0) = \varphi(x)}$$

2. Перед вычислением каждого следующего слоя по формуле Симпсона посчитать интеграл в управлении (5) для значений последнего известного j слоя: W

$$\boxed{I_j = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_3 + \cdots + 2y_{K-2} + 4y_{K-1} + y_K), n = \frac{l}{h},}$$

предполагается, что n – четное.

3. Составить неявную разностную схему с погрешностью $O(\tau + h^2)$ для уравнения (1), (6), учитывая I_j.

$$\boxed{\frac{y_k^{n+1} - y_k^n}{\tau} = \frac{y_{k+1}^{n+1} - 2y_k^{n+1} + y_{k-1}^{n+1}}{h^2} + u_k^n}$$

4. Учесть, что в данном случае условие устойчивости не доказано. Рекомендуется попробовать: $\frac{a^2\tau}{h^2} < \frac{1}{4}$, параметр а взять равным 1.

5. Составить трехточечные разностные производные первого порядка для краевых условий y_0', y_n' с погрешностью второго порядка.

$$\boxed{\frac{y_k^{n+1} - y_0^{n+1}}{h} = \frac{y_k^{n+1} - y_{k-1}^{n+1}}{h} = 0}$$

Преобразуя получаем:

$$\boxed{\frac{y_1^{n+1} - y_0^{n+1}}{h} - \frac{h}{2}\frac{y_0^{n+1} - y_0^n}{\tau} + \frac{h}{2}u_0^n = 0}$$

$$\boxed{\frac{y_k^{n+1} - y_{k-1}^{n+1}}{h} - \frac{h}{2}\frac{y_k^{n+1} - y_k^n}{\tau} + \frac{h}{2}u_0^n = 0}$$

6. Полученную систему линейных уравнений (5) привести к трех-диагональной и решить методом прогонки, оформив ее решение в виде подпрограммы на языке программирования высокого уровня.

Пример программной реализации можно найти в приложениях.

5. Проверки корректности

Ниже приводятся свойства решения поставленной задачи, отслеживание правильности построения решения могут вызывать значительные сложности.

1. На концах отрезка в силу (2) график функции численного решения имеет горизонтальные касательные.
2. Из Теоремы 1 следует, что см. рисунок 4, площадь фигуры, где график функции $\varphi(x)$ выше, чем $y(x, T)$ равна площади фигуры, где функция $\varphi(x)$ ниже, чем $y(x, T)$, то есть, $S_1 = S_2$.

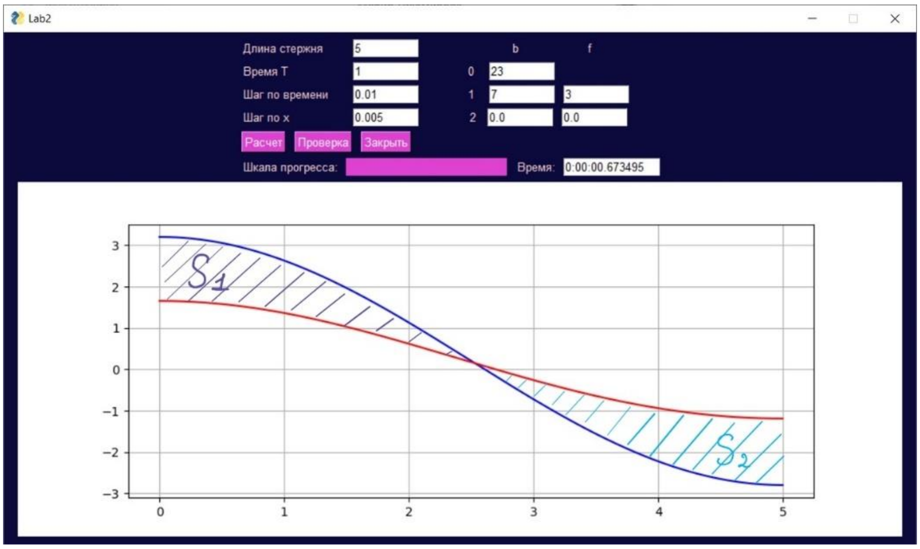

Рисунок 4. Следствие Теоремы 1.

3. При замене функции $\mathrm{b}(x)$на $b(x) + c_0$, где c_0 – некоторая константа, функция $y(x, T)$ не изменится (см. рисунок 4, 5).

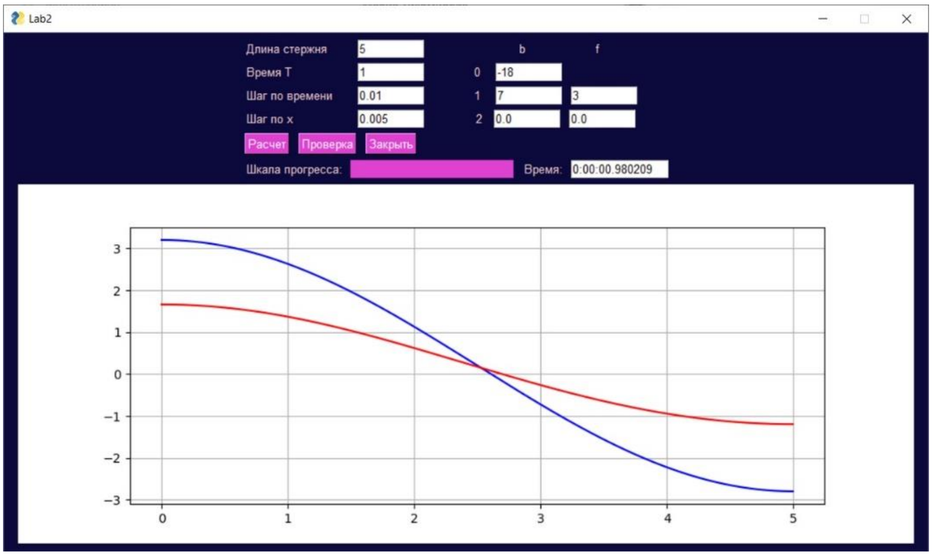

Рисунок 5. Функция с параметром b0 = -18.

4. Из Теоремы 2 следует, что зеленый график будет находиться «близко» к красному, в чем можно убедиться, глядя на рисунок 3.

6. Заключение

В данной лабораторной работе реализовано:

1. Написана программа с дружеским интерфейсом для численного решения задачи (1) - (3) и вывода функции $y(x, T)$ в графическом виде.

2. Выводится график функции $\varphi(x)$ – синим цветом, $y(x, T)$ – красным цветом.
3. Учтена возможность изменения:
   a. длины стержня l; времени T;
   b. шага h в разностной схеме по координате x, шага τ в разностной схеме по координате t;
   c. константы b_0, b_1, b_2, φ_0, φ_1.
2. Выведена строка прогресса.
3. Полученная функция $w(x, T)$ разделена на $I = \int_0^l w(x,T)dx$, посчитанный по формуле Симпсона для значений последнего слоя (при t = T), и выведен полученный график функции:

$$\boxed{\frac{w(x,T)}{\int_0^l w(x,T)dx}}$$

на экран зеленым цветом.

4. Зеленый график выводится на экран только при дополнительном нажатии кнопки «Проверка».

7. Список литературы

1. Эгамов А.И. Лабораторная работа «Численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных»: учебно-мет. пособие / А.И. Эгамов. – Нижний Новгород: Изд-во ННГУ, 2019. - 15с

2. А.Н. Тихонов, А.А. Самарский. Уравнения математической физики. – М.: Наука, 1979. 799с.

8. Приложение

Метод прогонки

```
def TridiagonalAlgorithm(a, b, c, f, count):
    A = []
    B = []
    res = [0] * count
    A.append(-c[0]/b[0])
    B.append(f[0]/b[0])
    for i in range(1, count):
        A.append(-c[i] / (a[i] * A[i - 1] + b[i]))
        B.append((f[i] - a[i] * B[i - 1]) / (a[i] * A[i - 1] + b[i]))
    res[count-1] = B[count - 1]
    for i in range(count - 2, -1, -1):
        res[i] = (A[i] * res[i + 1] + B[i])
    return res
```

Вычисление интеграла методом Симпсона

```
def Get_I(h, f):
    res = (h/3)*(f[0] + f[len(f) - 1])
    for i in range(1, len(f) - 1, 2):
        res += (h/3)*(4*f[i] + 2*f[i + 1])
    return res
```

Вычисление значений функции и заполнение нулевого слоя сетки

```
    for i in range(0, N):
        f_value.append(func(i*h, L, f1, f2))
        b_value.append(bfunc(i*h, L, b0, b1, b2))
        s1[0].append(f_value[i])
        s2[0].append(f_value[i])
```

Заполнение матрицы коэффициентов для метода прогонки

```
    a = [0.0]
    b = [1.0]
    c = [-1.0]
    for i in range(1, N - 1):
        a.append(tau / (h * h))
```

```
        b.append(-1 - 2*tau / (h * h))
        c.append(tau / (h * h))
    a.append(-1.0)
    b.append(1.0)
    c.append(0.0)
```

Вычисление последующих слоев сетки

```
    for i in range(1, T):

        y = []

        for j in range(0, N):
            y.append(b_value[j] * s1[i - 1][j])

        I = Get_I(h, y)
        f = [0]
        f2 = [0]
        s1.append([])
        s2.append([])
```

Вычисляем правую часть системы для прогонки

```
        for j in range(1, N - 1):
            f.append(-s1[i - 1][j] * ((b_value[j] - I) * tau * tau  + 1.0))
            f2.append(-s2[i - 1][j] * (b_value[j] * tau * tau + 1.0))
        f.append(0)
        f2.append(0)
```

Метод прогонки для системы из B

```
        res = TridiagonalAlgorithm(a, b, c, f, N)
        for j in range(0, N):
            s1[i].append(res[j])

        # Метод прогонки для системы из A


        res2 = TridiagonalAlgorithm(a, b, c, f2, N)
        for j in range(0, N):
            s2[i].append(res2[j])
```