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

Отчет по лабораторной работе

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

Выполнил:
студент группы 381706-2
Крюков Дмитрий Алексеевич

Проверил:
Ассистент кафедры дифференциальных уравнений, математического и численного анализа
Морозов Кирилл Евгеньевич

Нижний Новгород
2020

Содержание

Постановка задачи

Дан тонкий однородный стержень с теплоизолированными концами длины l. На процесс изменения температуры стержня осуществляется некое воздействие для достижения определённых целей, например, через стержень пропускается электрический ток или он помещается в электромагнитное поле (индукционный нагрев) и т. п. Построим математическую модель этого процесса.

Найти функцию – температуру стержня – непрерывно дифференцируемую по t и дважды непрерывно дифференцируемую по x – решение уравнения

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

, удовлетворяющее (концы теплоизолированы) однородным граничным условиям второго рода

$$y'_x(0,t) = y'_x(l,t) = 0$$

и начальному условию

$$y(x,0) = \varphi(x)$$

Непрерывная функция u(x) – управление с обратной связью)

$$u(x,t) = b(x)y(x,t) - y(x,t)\int_0^l b(x)y(x,t)dx$$

Исследование задачи

Нам необходимо построить разностную схему для решения задачи. Первй слой разностной схемы заполним значениями функции

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

Для вычисления каждого следующего слоя необходимо найти интеграл для значений последнего по формуле Симпсона:

$$I_j = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \ldots + 2y_{n-4} + 4y_{n-3} + 2y_{n-2} + 4y_{n-1} + y_n)$$

После чего составим явную разностную схему

$$\frac{y_i^{j+1} - y_i^j}{\tau} = \frac{y_{i-1}^{j+1} - 2y_i^{j+1} + y_{i+1}^{j+1}}{h^2} + x_i, \;\; i = 1, 2, ..., N-1, \;\; j = 0, 1, ..., M-1.$$

Представив краевые условия в виде разностных производных

$$\frac{y_N^{j+1} - y_{N-1}^{j+1}}{h} = 0$$

Получим СЛАУ вида:

$$\begin{cases} y_0^{j+1} = 0, \\ \frac{\tau}{h^2} y_{i-1}^{j+1} - \left(1 + \frac{2\tau}{h^2}\right) y_i^{j+1} + \frac{\tau}{h^2} y_{i+1}^{j+1} = -\left(y_i^j + \tau x_i\right) \\ y_N^{j+1} = y_{N-1}^{j+1} + ht_{j+1}, \end{cases}$$

Которую можно решить методом прогонки.

Заполнив таким образом все слои разностной схемы найдем решение уравнения

Программная реализация

В данной программе реализовано численное решение начально-краевой задачи для уравнения:

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

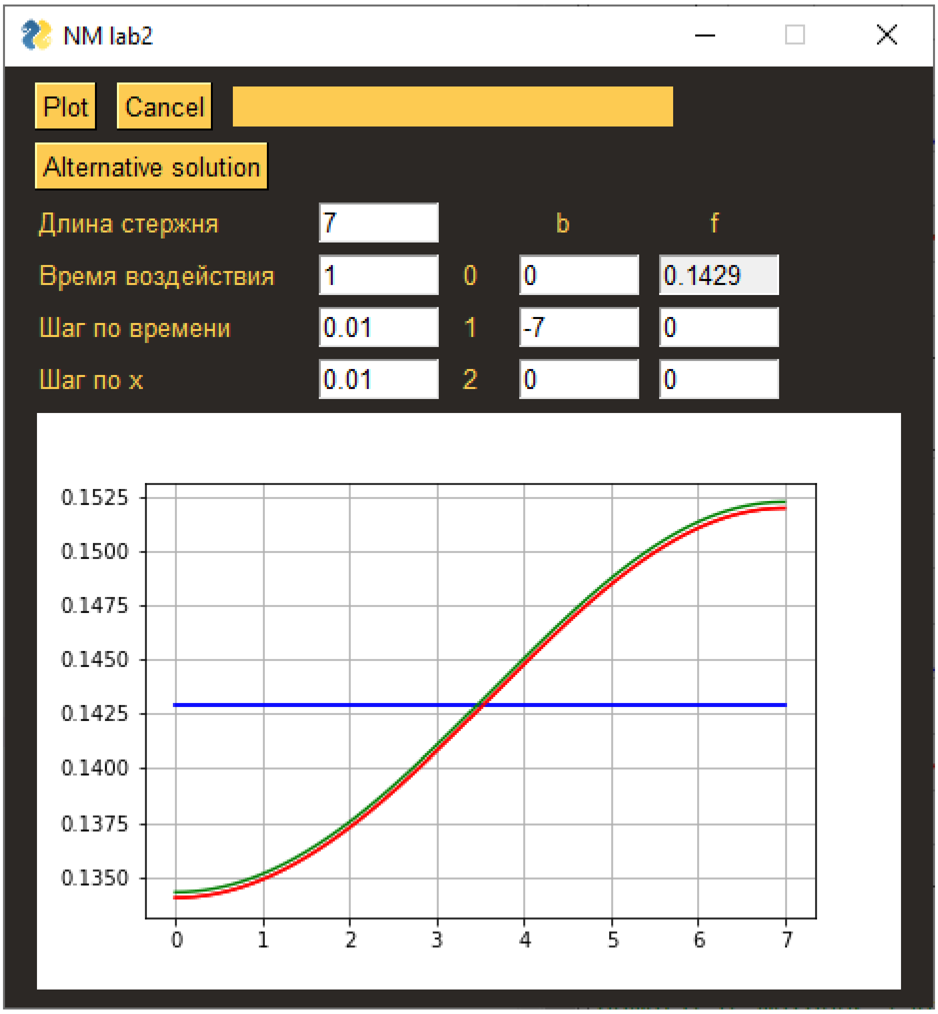
$$y(x,0) = \varphi(x) \quad u(x,t) = b(x)y(x,t)$$

Программа выполнена на языке python, листинг основных функция смотреть в приложении

Программа представляет собой оконное приложение в котором присутствуют поля для ввода:

- Длинны стержня
- Время воздействия на стержень
- Шага изменения сетки по времени и по координате x
- Коэффициенты функций b и f

Для проведения расчетов и постройки графиков нажмите кнопку plot, прогресс построения выводится вверху приложения рядом с кнопками plot и cancel. Красным цветом обозначается конечная температура, синим обозначается начальная температура стержня. Кнопка Alternative solution отображает решение из части А(обозначен зеленым), график которого совпадает с решением из части В

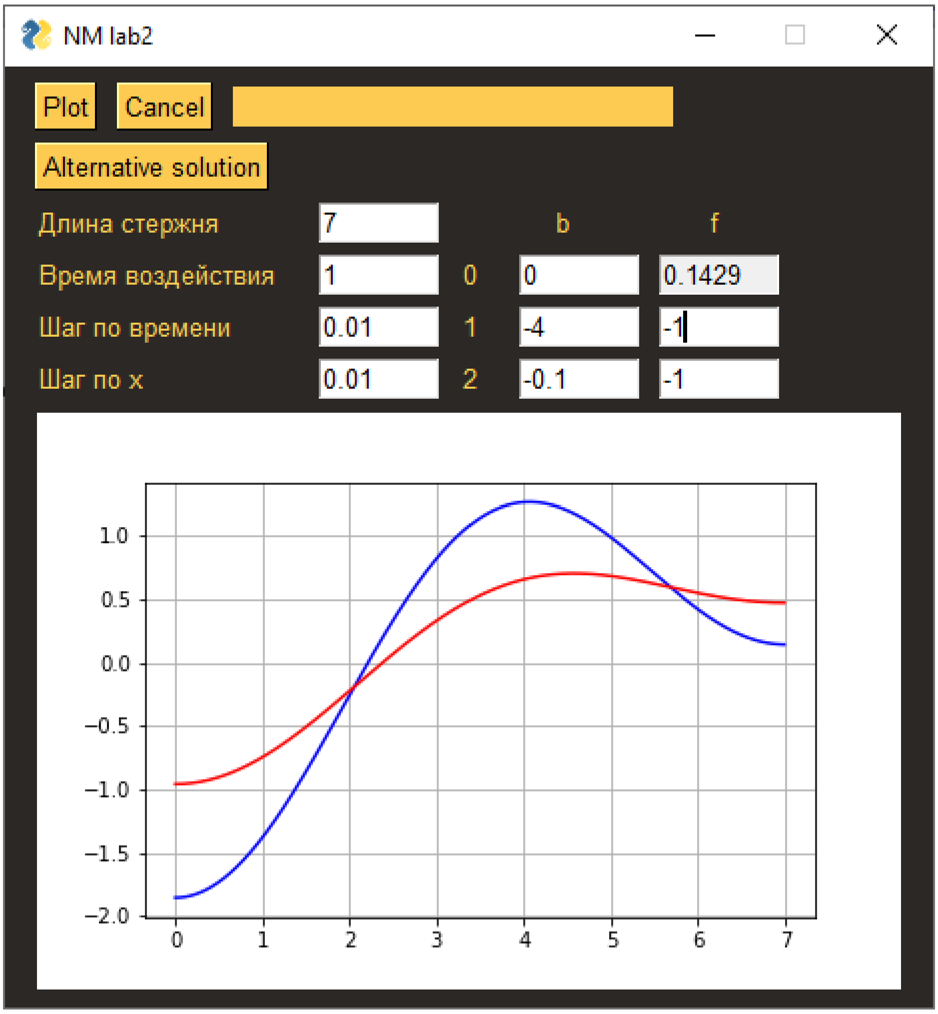


Для выхода из приложения нажмите на кнопку cancel

Пример работы

Приведем пример работы программы и применим методы самоконтроля из [1]

Построим систему:

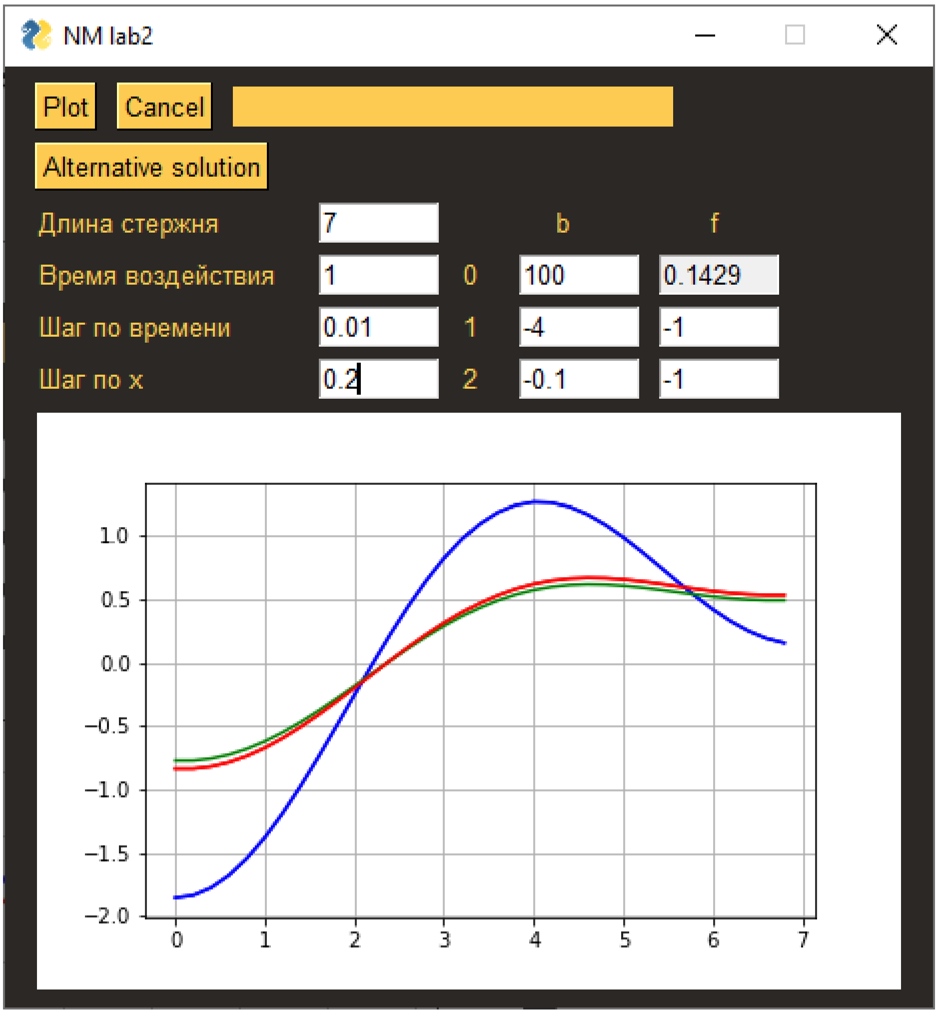


Как видно из решения на концах отрезка график функции численного решения имеет горизонтальные касательные т. к. концы стержня изолированы

В силу теоремы из [1, с. 6], площадь фигуры, где график функции $\varphi(x)$ выше, чем $y(x,T)$ равна площади фигуры, где функция $\varphi(x)$ ниже, чем $y(x,T)$

При замене функции $b(x)$ на $b(x) + c_0$, где c_0 – некоторая константа, функция $y(x, T)$ не изменится.

Из Теоремы 5 [1 с. 7] следует, что зеленый график должен будет находиться «близко» к красному.



Заключение

В ходе работы разработан алгоритм получения численного решения начально-краевой задачи для интегро-дифференциального уравнения в частных производных. Написана программа на языке программирования python с дружеским интерфейсом для решения задачи и вывода функции на экран в графическом виде, проведена проверка корректности работы программы,

Литература

1. Эгамов А.И. Лабораторная работа «Численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных»: учебно-мет. пособие. – Нижний Новгород: Изд-во ННГУ, 2019. - 15с.
2. А.А.Самарский П.Н.Вабищевич Е.А. Самарская. Задачи и упражнения по численным методам: Учебное пособие. — М.: Эдиториал УРСС, 2000. - 208 с.

Приложение

Функция численного интегрирования

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

Метод прогонки

```
def tridiagAlg(a, b, c, func, count):#
    A = []
    B = []
    res = [0] * count
    A.append(-c[0]/b[0])
    B.append(func[0]/b[0])
    for i in range(1, count):
       A.append(-c[i] / (a[i] * A[i - 1] + b[i]))
       B.append((func[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 значений функции и заполнение нулевого слоя сетки

```
for i in range(0, count_N):
    func_val.append(func(i*delta_x, _len, f1, f2))
    bfunc_val.append(bfunc(i*delta_x, _len, b0, b1, b2))
```

```
        slices1[0].append(func_val[i])
        slices2[0].append(func_val[i])
```

Заполнение матрицы коэффициентов для метода прогонки

```
    coeff_a = [0.0]
    coeff_b = [1.0]
    coeff_c = [-1.0]
    for i in range(1, count_N - 1):
        coeff_a.append(delta_t / (delta_x * delta_x))
        coeff_b.append(-1 - 2*delta_t / (delta_x * delta_x))
        coeff_c.append(delta_t / (delta_x * delta_x))
    coeff_a.append(-1.0)
    coeff_b.append(1.0)
    coeff_c.append(0.0)
```

Вычисление последующих слоев сетки

```
    for i in range(1, count_T):
        y_func = []
        for j in range(0, count_N):
            y_func.append(bfunc_val[j] * slices1[i - 1][j])
        I = integrate(delta_x, y_func)
        fu = [0]
        fu2 = [0]
        slices1.append([])
        slices2.append([])
```

Вычисляем правую часть системы для прогонки

```
                for j in range(1, count_N - 1):
                    fu.append(-slices1[i - 1][j] * ((bfunc_val[j] - I)
* delta_t * delta_t  + 1.0))
                    fu2.append(-slices2[i - 1][j] * (bfunc_val[j] *
delta_t * delta_t + 1.0))
                fu.append(0)
                fu2.append(0)
```

Метод прогонки для системы из B

```
                res = tridiagAlg(coeff_a, coeff_b, coeff_c, fu,
count_N)
                for j in range(0, count_N):
                    slices1[i].append(res[j])
```

Метод прогонки для системы из A

```
                res2 = tridiagAlg(coeff_a, coeff_b, coeff_c, fu2,
count_N)
                for j in range(0, count_N):
                    slices2[i].append(res2[j]
```