МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Институт информационных технологий, математики и механики
Кафедра алгебры, геометрии и дискретной математики

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

Выполнила:
студент группы 381706-2
Ясакова Анастасия Евгеньевна
____________ Подпись

Проверил:
Морозов Кирилл Евгеньевич
____________ Подпись

Нижний Новгород

2020

Оглавление

Введение

Дифференциальное уравнение в частных производных (частные случаи также известны как уравнения математической физики, УМФ) — дифференциальное уравнение, содержащее неизвестные функции нескольких переменных и их частные производные.
Существует два вида методов решения данного типа уравнений:

- аналитический, при котором результат выводится различными математическими преобразованиями;
- численный, при котором полученный результат соответствует действительному с заданной точностью, но который требует много рутинных вычислений и поэтому выполним только при помощи вычислительной техники (ЭВМ).

Поскольку нахождение аналитического решения даже простого уравнения в сложной области не всегда возможно, то было разработано множество методов решения уравнений математической физики. Некоторые из них основываются
на аппроксимации дифференциального оператора некоторыми выражениями, другие сводят задачу к проекционной или вариационной и решают её.

Постановка Задачи

Описание управляемого процесса

Рассмотрим управляемый процесс нагревания стержня: 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. 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 − решение уравнения

$$y_t'(x,t) = \alpha^2 y_{xx}''(x,t) + u(x,t)$$

удовлетворяющее (концы теплоизолированы) однородным граничным условиям второго рода

$$y_x'(0,\ t) = y_x'\ (l,\ t) = 0$$

и начальному условию

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

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

$$\int_0^l \varphi(x)dx = 1$$

Непрерывная функция $u(x,\ t)$ − управление с обратной связью

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

где $b(x)$– управляющая функция, непрерывная на отрезке $[0,\ l]$.

Наложение условия дважды непрерывной дифференцируемости на отрезке $[0,\ l]$ объясняется следующим. Во-первых, чтобы решать поставленную задачу одним из самых известных методов – методом Фурье и, чтоб полученный ряд – решение задачи – можно было дифференцировать по t и дважды дифференцировать по x.

Задача

Часть А.

1. Составить неявную разностную схему с погрешностью $O(\tau + h)$ для уравнения.
2. Учесть условие устойчивости $\frac{a^2\tau}{h^2} < 1/2$.
3. Определить нулевой слой для будущей разностной схемы.
4. Составить трехточечные разностные производные первого порядка для краевых условий: y_0' , y_n' с погрешностью второго порядка.
5. Разработать алгоритм получения численного решения задачи.
6. Полученную систему линейных уравнений привести к трехдиагональной матрице и решить методом прогонки , оформив решение в виде подпрограммы.
7. Написать программу на языке программирования высокого уровня с дружеским интерфейсом для численного решения задачи и вывода функции $w(x,T\ )$ на экран в графическом виде. В качестве начальной функции рекомендуется взять $\varphi(x) = \frac{1}{l} + \varphi_1\cos(\frac{\pi x}{l})$, в качестве функции $b(x) = b_1\cos(\frac{\pi x}{l})$, где φ_1, b_1 – некие константы.
8. На одном и том же рисунке вывести оси координат, график функции $\varphi(x)$ - синим цветом; график функции - красным цветом.
9. Учесть в оконном меню программы возможность изменения:

- длины стержня l; времени Т;
- шага h в разностной схеме по координате x;
- шага τ в разностной схеме по координате t;
- констант b_0, b_1, φ_1.

Часть В.

1. Определить нулевой слой для будущей разностной схемы.
2. Перед вычислением каждого следующего слоя по формуле Симпсона посчитать интеграл в управлении для значений последнего известного j слоя;

$$I_l = \frac{h}{6}(y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + y_n)$$

предполагается, что n − четное.

3. Составить неявную разностную схему с погрешностью $O(\tau + h)$ для уравнения, учитывая I_l.
4. Учесть, что в данном случае условие устойчивости не доказано. Рекомендуется попробовать: $\frac{a^2\tau}{h^2} < 1/4$.
5. Составить трехточечные разностные производные первого порядка для краевых условий: y_0' , y_l' с погрешностью второго порядка.
6. Разработать алгоритм получения численного решения задачи.
7. Полученную систему линейных уравнений привести к трехдиагональной матрице и решить методом прогонки, оформив ее решение в виде подпрограммы на языке программирования высокого уровня.
8. Написать программу на языке программирования высокого уровня с дружеским интерфейсом для численного решения задачи и вывода функции $y(x,T)$ на экран в графическом виде. В качестве начальной функции рекомендуется взять $\varphi(x) = \frac{1}{l} + \varphi_1\cos(\frac{\pi x}{l}) + \varphi_2\cos(\frac{2\pi x}{l})$, в качестве функции $b(x) = b_0 + b_1\cos(\frac{\pi x}{l}) + b_2\cos(\frac{2\pi x}{l})$, где $b0$, $b1$, $b2$, φ1 , φ2 – некие константы.
9. На одном и том же рисунке вывести оси координат, график функции $\varphi(x)$- синим цветом; график функции $y(x,T)$ - красным цветом.
10. Учесть в оконном меню программы возможность изменения:
    - длины стержня l; времени T
    - шага h в разностной схеме по координате x;
    - шага τ в разностной схеме по координате t;
    - константы $b0$, $b1$,φ1.
11. Вывести на экран время выполнения данной работы и строку прогресса.
12. Полученную функцию $w(x,T)$ в части А нужно разделить на $I = \int_0^l w(x,T)dx$, который нужно посчитать по формуле Симпсона для значений последнего известного слоя (при $t=T$), и вывести полученный график функции $\frac{w(x,T)}{\int_0^l w(x,T)dx}$ на экран светло-зеленым цветом.
13. 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».

Исследование задачи

$$\mathrm{y_t}'(\mathrm{x},\mathrm{t}) = \alpha^2 y_{\mathrm{xx}}''(\mathrm{x},\mathrm{t}) + u(x,t)$$

Распишем производные:

$$\frac{\partial y}{\partial t}\xrightarrow[t=t_{\mathrm{l}}]{}\frac{y(x,t_{\mathrm{l+1}}) - y(x,t_{\mathrm{l}})}{\Delta t}$$

$$\frac{\partial^2 y}{\partial x^2}\xrightarrow[x=x_{\mathrm{p}}]{}\frac{y(x_{\mathrm{p+1}},t) - 2y(x_{\mathrm{p}},t) - y(x_{\mathrm{p-1}},t)}{\Delta x^2}$$

Подставим в уравнение:

$$\frac{y(x_{\mathrm{p}},t_{\mathrm{l+1}}) - y(x_{\mathrm{p}},t_{\mathrm{l}})}{\Delta t} - a^2\,\frac{y(x_{\mathrm{p+1}},t_{\mathrm{l+1}}) - 2y(x_{\mathrm{p}},t_{\mathrm{l+1}}) - y(x_{\mathrm{p-1}},t_{\mathrm{l+1}})}{\Delta x^2} = u(x_{\mathrm{p}},t_{\mathrm{l+1}})$$

Перенеся известные величины в правую часть, умножив на Δt и сгруппировав коэффициенты, приведём СЛАУ к окончательному виду:

$$\frac{-a^2\Delta t}{\Delta x^2}y(x_{\mathrm{p-1}},t_{\mathrm{l+1}}) + (1 + 2\frac{a^2\Delta t}{\Delta x^2})y(x_{\mathrm{p}},t_{\mathrm{l+1}}) + \frac{-a^2\Delta t}{\Delta x^2}y(x_{\mathrm{p+1}},t_{\mathrm{l+1}}) = y(x_{\mathrm{p}},t_{\mathrm{l}}) + \Delta t u(x_{\mathrm{p}},t_{\mathrm{l+1}})$$

Вид матрицы коэффициентов для конечных точек разностной сетки определяется граничными условиями и выводится отдельно. Наличие диагонального преобладания у матрицы коэффициентов гарантирует устойчивость метода прогонки при решении им данной СЛАУ.

При реализации программы будем считать $a = 1$.

Метод прогонки

Метод прогонки является частным случаем метода Гаусса и применяется к системам с трёх-пятидиагональной матрицей. Такая система получается при построении кубического сплайна.

Уравнение (2) можно решить методом прогонки, представив в виде системы линейных алгебраических уравнений вида $Ax = F$, где вектор x соответствует вектору $\{c_i\}$, вектор F поэлементно равен правой части уравнения (2).

Трёхдиагональная матрица выглядит следующим образом:

$$A = \begin{pmatrix} C_1 & B_1 & 0 & 0 & \cdots & 0 & 0 \\ A_2 & C_2 & B_2 & 0 & \cdots & 0 & 0 \\ 0 & A_3 & C_3 & B_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & B_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & A_n & C_n \end{pmatrix},$$

где $A_i = h_i, i = 2, \dots, n, B_i = h_{i+1}, i = 1, \dots, n-1$ и $C_i = 2(h_i + h_{i+1}), i = 1, \dots, n$

Данный метод основан на предположении, что искомые неизвестные связаны рекуррентным соотношением: $x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1} \ \ i = 1, \dots, n-1$

Используя это соотношение, выразим x_{i-1} и x_i через x_{i+1} и подставим в i-е уравнение:

$$(A_i a_i a_{i+1} + \ C_i a_{i+1} + \ B_i)x_{i+1} + A_i a_i \beta_{i+1} + A_i \beta_i + C_i \beta_{i+1} - \ \ F_i = 0$$

где F_i – правая часть i-го уравнения. Это соотношение будет выполняться независимо от решения, если потребовать

$$A_i a_i a_{i+1} + \ C_i a_{i+1} + \ B_i = 0$$

$$A_i a_i \beta_{i+1} + A_i \beta_i + C_i \beta_{i+1} - \ \ F_i = 0$$

Отсюда следует:

$$\alpha_{i+1} = \frac{-B_i}{A_i \alpha_i + C_i} \quad \beta_{i+1} = \frac{F_i - A_i \beta_i}{A_i \alpha_i + C_i}$$

Из первого уравнения получим:

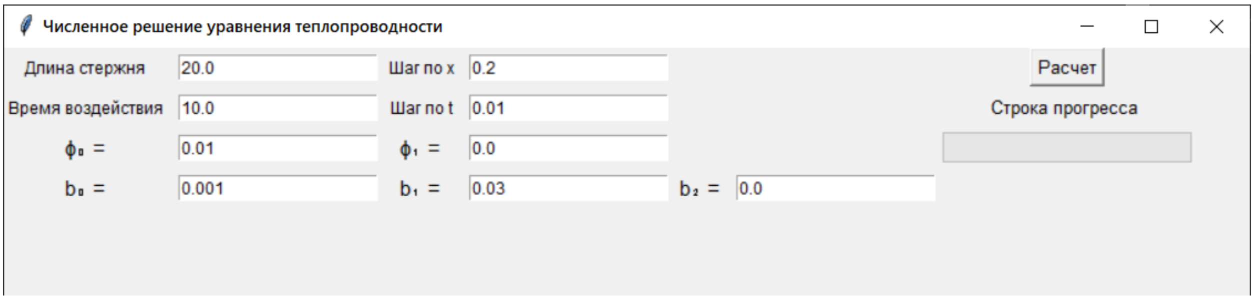
$$\alpha_2 = \frac{-B_1}{C_1}, \ \ \beta_2 = \frac{F_1}{C_1}$$

После нахождения коэффициентов α и β получим решение системы:

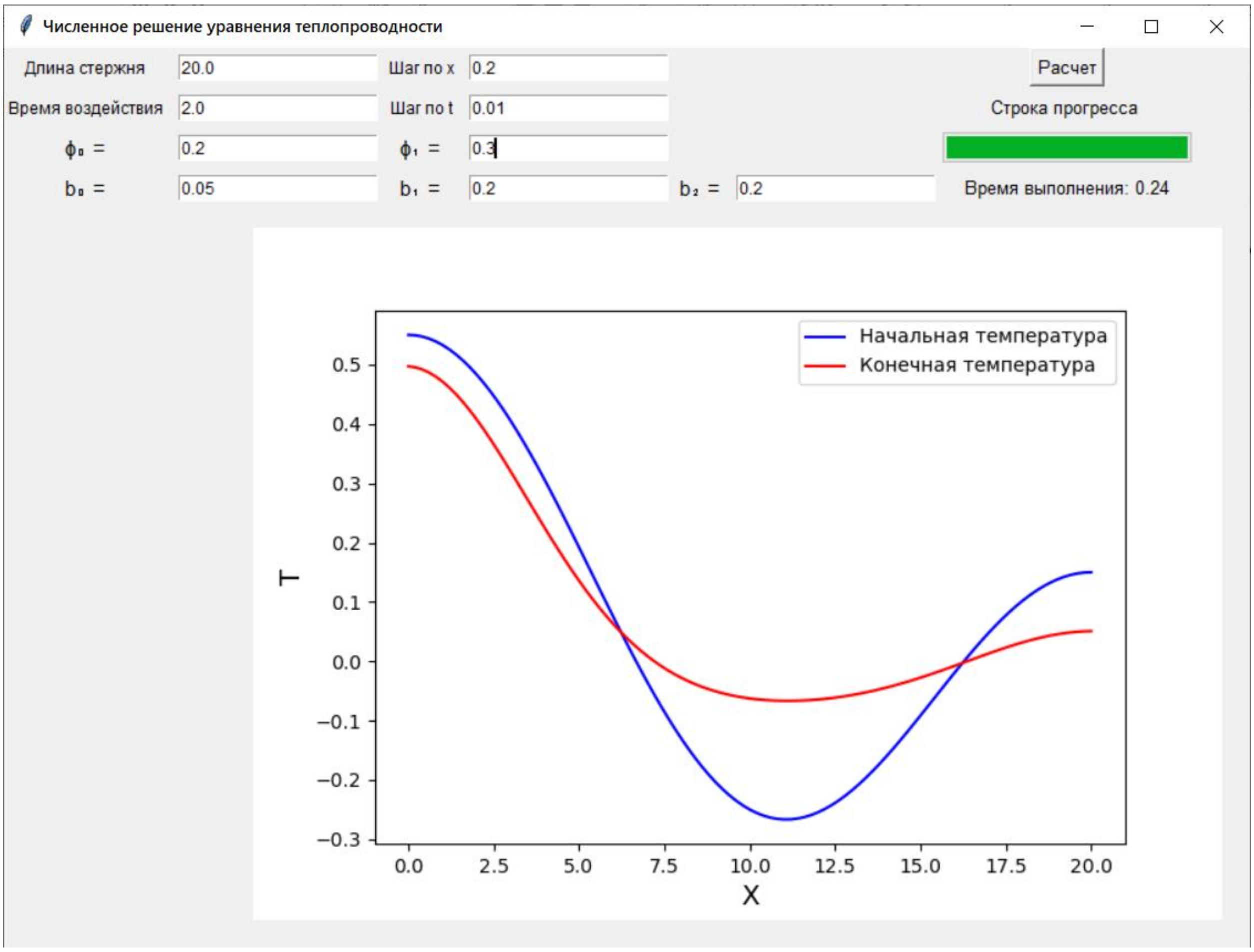
$$x_n = \frac{F_n - A_n \beta_n}{C_n + A_n \alpha_n} = c_n$$

Руководство пользователя

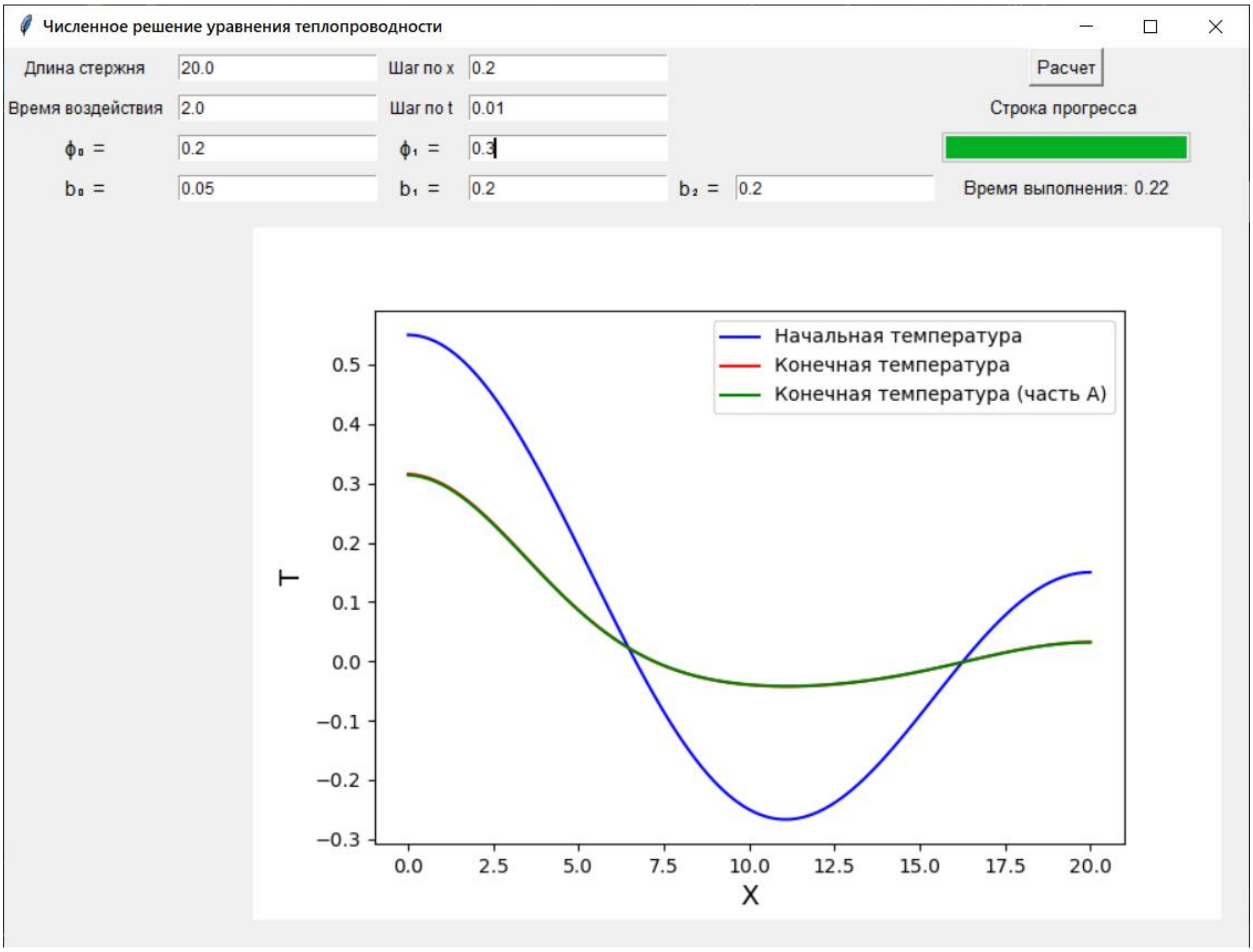
1. При запуске пользователю будет предложено ввести длину стержня, время воздействия, шаг по x, шаг по t и параметры.



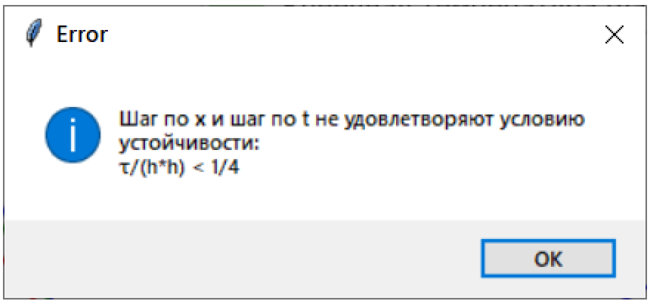
2. После ввода появится графики начальной температуры и конечной температуры.



3. Чтобы посмотреть конечную температуру, которую получили в части 1, нужно нажать пробел.



4. Можно обновить все параметры и снова построить графики, прежние очистятся сами собой.
5. При вводе шагов, нарушающих условие устойчивости, появится ошибка.



Руководство программиста

Программа написана на Python с помощью библиотек matplotlib для визуализации графиков и tkinter для реализации интерфейса.

В функции `run_through_method` реализован метод прогонки.

```
def run_through_method(A, B, C, F):
    size = len(F)
    i = 1;
    a = [1] * (size - 1);
    b = [1] * (size - 1);
    x = [1] * size
    a[0] = -C[0] / B[0]
    b[0] = F[0] / B[0]
    while (i < size  - 1):
        a[i] = -C[i] / (A[i] * a[i - 1] + B[i])
        b[i] = (F[i] - A[i] * b[i - 1]) / (A[i] * a[i - 1] + B[i])
        i = i + 1

    x[size - 1] = (F[size - 1] - A[size - 1] * b[size - 2]) /
                    (B[size - 1] + A[size - 1] * a[size - 2])
    i = size - 2
    while (i > -1):
        x[i] = a[i] * x[i + 1] + b[i]
        i = i - 1
    return x
```

В функции `decision` реализовано решение части B.

```
def decision(mpb, window, l, T, h_t, h_x, b0, b1, b2, f0, f1):
    if (h_t / (h_x * h_x) >= 0.25):
        tk.messagebox.showinfo(title="Error", message="Шаг по x и
      шаг по t не удовлетворяют условию устойчивости:\nτ/(h*h) < 1/4")
        return
    time1 = time.time()
    max = int(2 * T / h_t)
    mpb["maximum"] = max # для строки прогресса
    mpb["value"] = 0
    window.update_idletasks()
    n = int(l / h_x) + 1
    m = int(T / h_t)
    # коэффициенты для метода прогонки
    A = [0] * n
    B = [0] * n
    C = [0] * n
    f_ = [0] * n
    # начальные функции
    b = [0] * n
    f = [0] * n
    x = [0] * n
    # часть 1
    j = 0
    while (j < m):
        i = 0
        while (i < n):
```

```
            if (j == 0):
                x[i] = i * h_x
                f[i] = 1 / l + f0 * math.cos(math.pi * x[i] / l)
                             + f1 * math.cos(2 * math.pi * x[i] / l)
                f_[i] = f[i]
                b[i] = b0 + b1 * math.cos(math.pi * x[i] / l)
                          + b2 * math.cos(2 * math.pi * x[i] / l)
            else:
                f_[i] = w[i]
            B[i] = 1 + 2 * h_t / (h_x * h_x) - h_t * b[i]
            C[i] = -h_t / (h_x * h_x)
            A[i] = -h_t / (h_x * h_x)
            i = i + 1
        # подсчет коэффициентов для 0 и n-1
        A[0] = 0
        B[0] = 1 + h_t / (h_x * h_x) - h_t * b[0]
        C[0] = -h_t / (h_x * h_x)
        A[n - 1] = -h_t / (h_x * h_x)
        B[n - 1] = 1 + h_t / (h_x * h_x) - h_t * b[n - 1]
        C[n - 1] = 0
        w = run_through_method(A, B, C, f_)
        mpb["value"] = mpb["value"] + 1
        window.update_idletasks()
        j = j + 1
    # подсчет интеграла
    w_I = w[0] + w[n-1]
    j = 1
    while(j < n - 1):
        if (j%2 == 1):
            w_I = w_I + 4 * w[j]
        else:
            w_I = w_I + 2 * w[j]
        j = j + 1
    w_I = w_I * h_x / 3
    for i in range(n):
        w[i] = w[i] / w_I
    j = 0
    y = [0] * n
    # часть 2
    I = 0
    while (j < m):
        i = 0
        while (i < n):
            if (j == 0):
                f[i] = 1 / l + f0 * math.cos(math.pi * x[i] / l)
                             + f1 * math.cos(2 * math.pi * x[i] / l)
                f_[i] = f[i]
                b[i] = b0 + b1 * math.cos(math.pi * x[i] / l)
                          + b2 * math.cos(2 * math.pi * x[i] / l)
            else:
                    f_[i] = y[i]
            B[i] = 1 + 2 * h_t / (h_x * h_x) - h_t * b[i] + h_t * I
            C[i] = -h_t / (h_x * h_x)
            A[i] = -h_t / (h_x * h_x)
            i = i + 1
        A[0] = 0
        B[0] = 1 + h_t / (h_x * h_x) - h_t * b[0] + h_t* I
        C[0] = -h_t / (h_x * h_x)
```

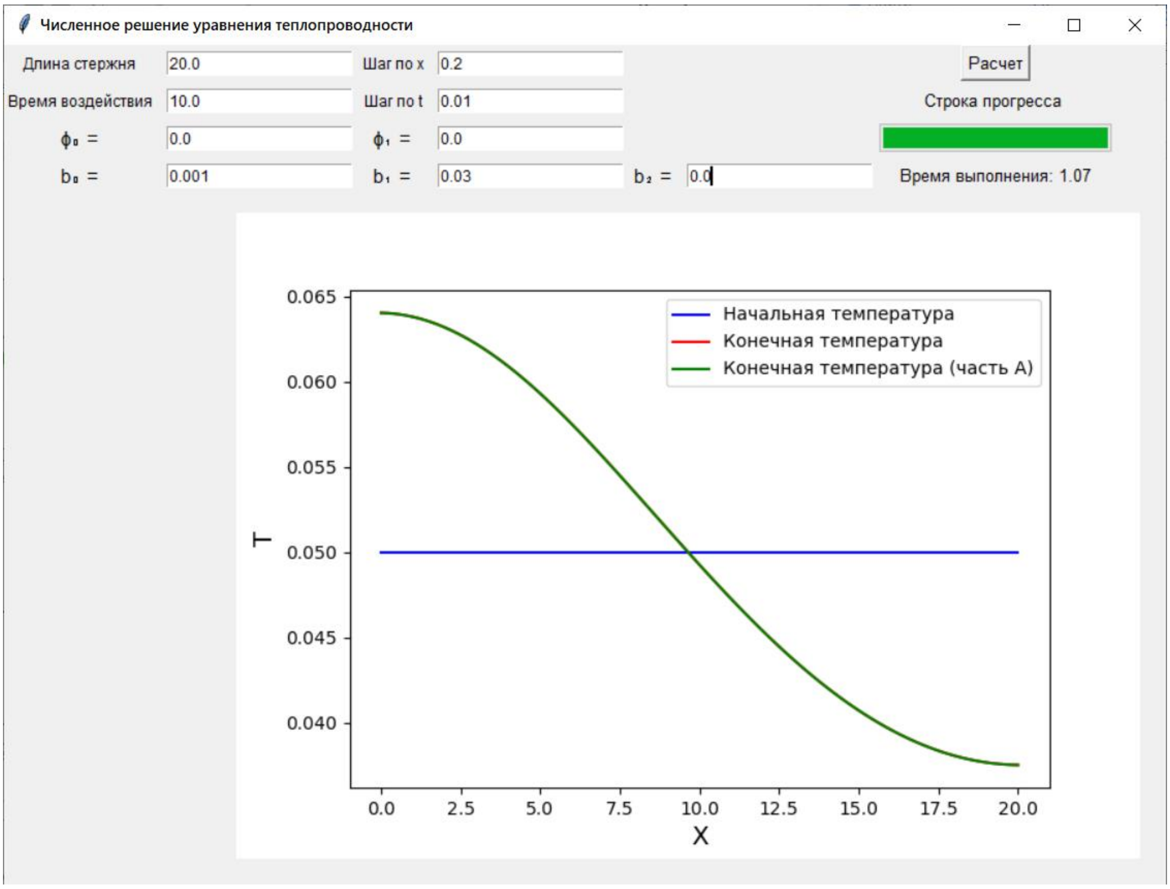
```
        A[n - 1] = -h_t / (h_x * h_x)
        B[n - 1] = 1 + h_t / (h_x * h_x) - h_t * b[n - 1] + h_t * I
        C[n - 1] = 0
        y = run_through_method(A, B, C, f_)
        # подсчет интеграла
        I = y[0] * b[0]+ y[n - 1] * b[n - 1]
        k = 1
        while (k < n - 1):
            if (k%2 == 1):
                I = I + 4 * y[k] * b[k]
            else:
                I = I + 2 * y[k] * b[k]
            k = k + 1
        I = I * h_x / 3
        mpb["value"] = mpb["value"] + 1
        window.update_idletasks()
        j = j + 1

    time2 = time.time()
    time_ = time2 - time1
    str_ = "Время выполнения: " + str(round(time_, 2))
    label = Label(text = str_, font = "Arial 10")
    label.grid(row=3, column=6)
    window.update_idletasks()

    global green, _x, a, canvas
    green = w
    _x = x
    fig = Figure(figsize=(7,5)) # размер графика
    a = fig.add_subplot(111)
    a.plot(x, f, label = "Начальная температура", color='blue')
    a.plot(x, y, label = "Конечная температура", color='red')
    a.set_ylabel("T", fontsize=14)
    a.set_xlabel("X", fontsize=14)
    a.legend()
    canvas = FigureCanvasTkAgg(fig, master=window)
    canvas.get_tk_widget().place(relx=0.2, rely=0.2) #сдвиг по окну
    canvas.draw()
    id = fig.canvas.mpl_connect('key_press_event', continue_)
                                                # реакция на нажатие пробела
```

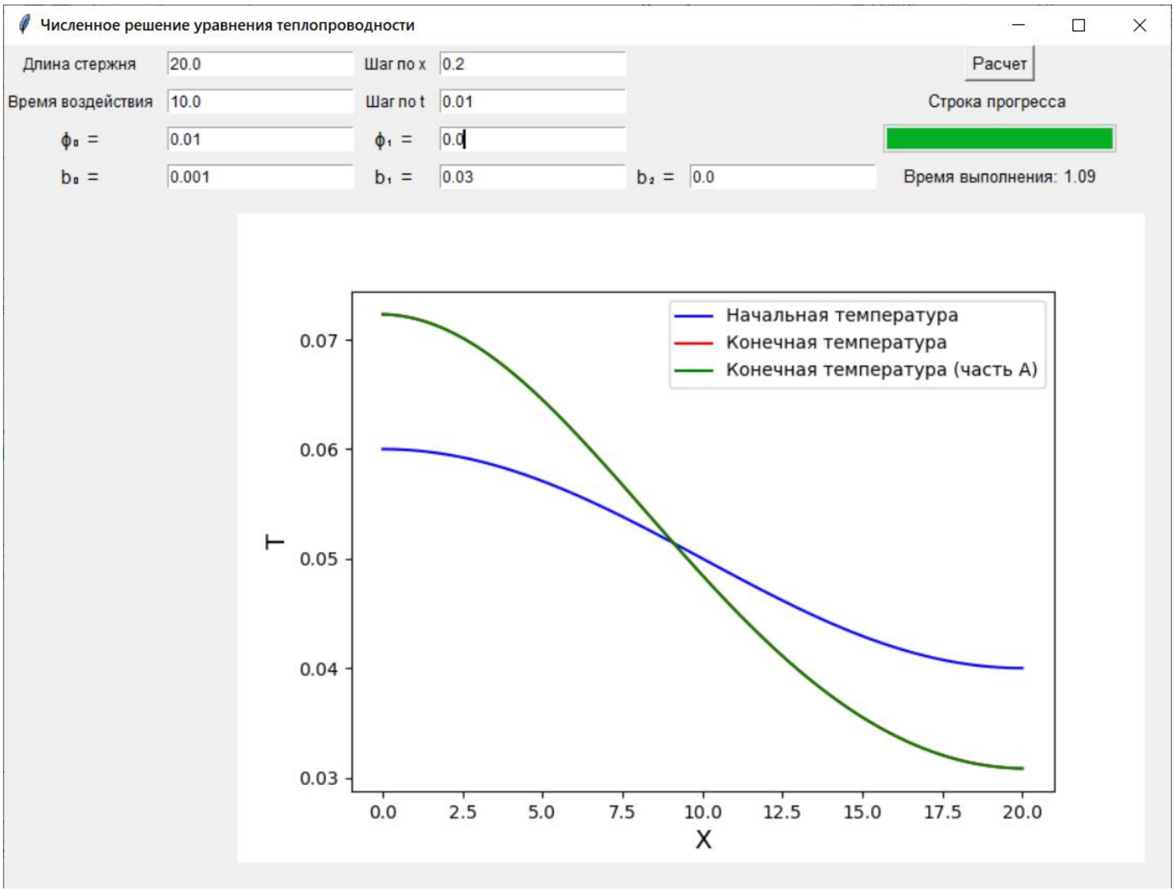
В функции `continue_` реализовано построение конечной температуры (часть А).

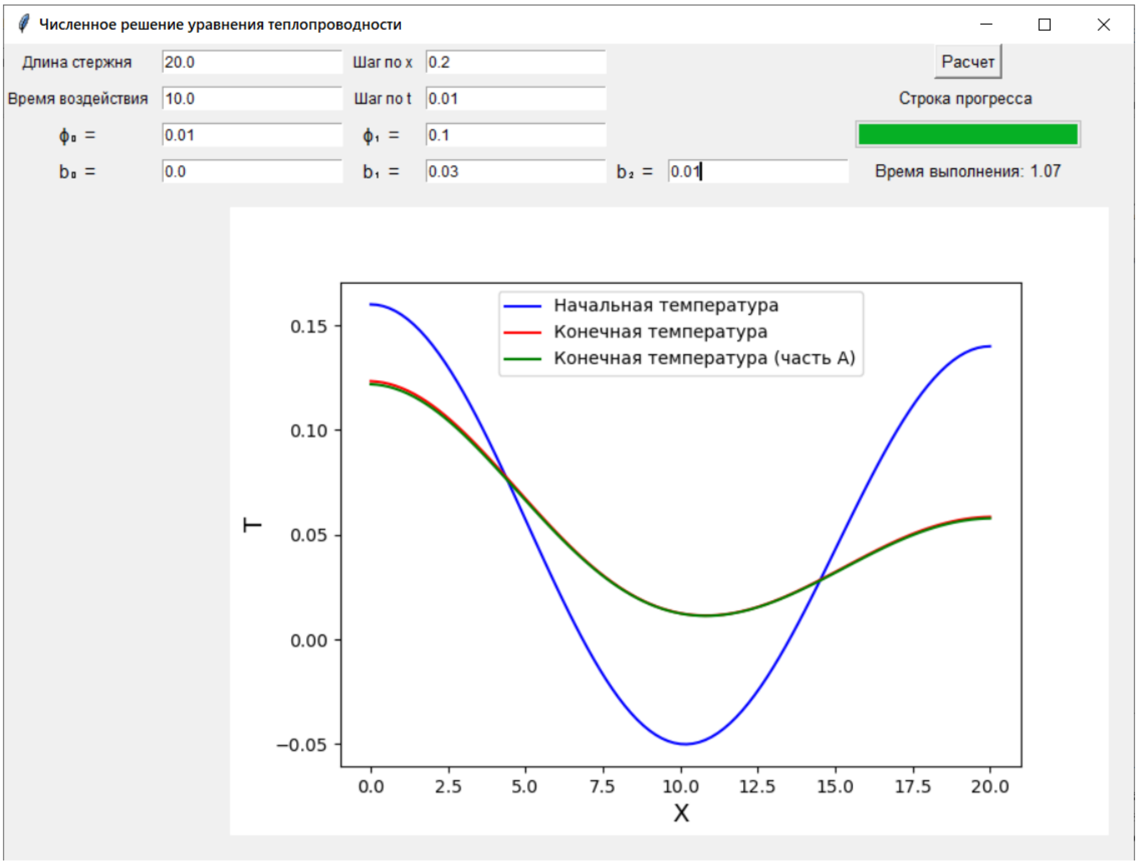
```
def continue_(event):
    global green, _x, a, canvas
    a.plot(_x, green, label = "Конечная температура (часть А)",
color='green')
    a.legend()
    canvas.draw()
```

В основной части написана реализация интерфейса программы.

Пример работы







Заключение

В процессе работы была изучена задача численного решения начально-краевой задачи для интегро-дифференциального уравнения в частных производных (уравнения теплопроводности).