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 381706-1
Денисов В.Л.

Проверил:
ст. преп. каф. ДУМЧА ИИТММ
Эгамов А.И.

Нижний Новгород
2020.

Содержание

1. Введение

Лабораторная работа направлена на изучение вопроса численного решения начально-краевой задачи для интегро-дифференциального уравнения в частных производных.

Дифференциальные уравнения – один из наиболее важных инструментов математического моделирования. Основа большинства физических законов сформулирована в их терминах.

Аналитическое решение наиболее интересных дифференциальных уравнений, как правило, невозможно. В связи с этим возникает задача численного решения дифференциальных уравнений.

В рамках данной предлагается к рассмотрению управляемый процесс нагревания стержня: дан тонкий однородный стержень с теплоизолированными концами длины l.

На процесс изменения температуры стержня осуществляется некое воздействие, например, через стержень пропускается электрический ток. Математическая модель этого процесса будет рассмотрена в теоретической части данной работы.

Для разработки программного комплекса будет использован язык программирования C#, который предоставляет удобные инструменты для написания программ, а также позволяет применять его вместе с .NET Framework, предлагающим API-интерфейс WPF для создания настольных графических программ, имеющих неплохой дизайн и интерактивность.

2. Постановка задачи

В рамках лабораторной работы ставится задача реализации программного комплекса, который позволит находить численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных.

Программный комплекс должен обладать следующими возможностями:

- Получение входных данных для нахождения решения, а именно: длина стержня l, время воздействия T, величина шага h по пространству, величина шага τ по времени, параметры φ_1, φ_2 функции $\varphi(x)$ начального распределения температуры, параметры b_0, b_1, b_2 управляющей функции $b(x)$.
- Вычисление решения начально-краевой задачи для интегро-дифференциального уравнения в частных производных. При этом должно использоваться 2 варианта функций управления с обратной связью.
- Вывод на экран полученного решения в графическом виде.
- Нахождение решения для других параметров начальных функций без перезапуска программы.

Программный комплекс будет предоставлять:

1. Пользовательский интерфейс для работы с программой.
2. Необходимый набор функций, для нахождения численного решения начально-краевой задачи для интегро-дифференциального уравнения в частных производных.

3. Теоретическая часть

3.1 Описание управляемого процесса

Как было отмечено во Введении, здесь будет рассмотрена математическая модель процесса нагревания стержня.

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

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

удовлетворяющее (концы теплоизолированы) однородным граничным условиям второго рода:

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

и начальному условию:

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

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

$$\int_0^l \varphi(x)dx = 1 \tag{4}$$

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

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

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

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

3.2 Решение задачи

Решение задачи будем рассматривать на примере начальной функции $\varphi(x) = 1/l + \varphi_1 \cos(\pi x/l) + \varphi_2 \cos(2\pi x/l)$ и управляющей функции $b(x) = b_0 + b_1 \cos(\pi x/l) + b_2 \cos(2\pi x/l)$, где l – длина стержня.

Приступим к решению с составления неявной разностной схемы с погрешностью $O(\tau + h^2)$ для уравнений (1) и (6) (этот подход обозначается как «Часть Б»).

Построим в области Q равномерную сетку:

$\omega_{h\tau} = \{(x_i, t_j): x_i = ih, t_j = j\tau,$ где $i = \overline{0,n},\ j = \overline{0,m},\ h = l/n,\ \tau = T/m\}$, где n и m – целые числа, задающие соответственно число точек по пространству и по времени.

Краевую задачу аппроксимируем при помощи замены дифференциальных операторов на следующие разностные операторы:

$$(y''_{xx})_{i,j} = \frac{y_{i-1,j} - 2y_{i,j} + y_{i+1,j}}{h^2} \tag{7}$$

$$(y'_t)_{i,j} = \frac{y_{i,j} - y_{i,j-1}}{\tau} \tag{8}$$

Указанные уравнения подставляем в (1) и получаем (далее будем использовать $y_{i,j+1}$ для удобства теоретических выкладок):

$$\frac{y_{i,j+1} - y_{i,j}}{\tau} = a^2 \frac{y_{i-1,j+1} - 2y_{i,j+1} + y_{i+1,j+1}}{h^2} + u_{i,j} \tag{9}$$

где $u_{i,j} = b_i y_{i.j} - y_{i.j} \int_0^l b(x) y(x,t) dx$, а интеграл для каждого слоя находится по методу Симпсона.

Сделаем замену $r = \frac{a^2\tau}{h^2}$ и получим неявную сеточную схему (заметим, что в нашем случае принимается коэффициент $a = 1$):

$$r y_{i-1,j+1} - (1+2r) y_{i,j+1} + r y_{i+1,j+1} = -y_{i.j} - \tau u_{i,j} \tag{10}$$

Значения $y_{i.j+1}$ можно найти методом прогонки.

Пусть $A_i = C_i = r$, $B_i = -(1+2r)$, $F_i = -y_{i.j} - \tau u_{i,j}$

Тогда неявная сеточная схема приобретёт вид:

$$A_i y_{i-1,j+1} + B_i y_{i,j+1} + C_i y_{i+1,j+1} = F_i \tag{11}$$

Однако для решения полученной системы, следует учесть еще граничные условия (2):

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

Заменим эти дифференциальные операторы на центральные разностные производные с погрешностью второго порядка:

$$\begin{gathered} \frac{y_1 - y_{-1}}{2h} = 0 \\ \frac{y_{n+1} - y_{n-1}}{2h} = 0 \end{gathered} \tag{12}$$

Таким образом, для нулевого слоя неявная сеточная схема приобретает вид:

$$B_0 y_{0,j+1} + (A_0 + C_0) y_{1,j+1} = F_0 \tag{13}$$

А для последнего слоя:

$$(A_n + C_n) y_{n-1,j+1} + B_n y_{n,j+1} = F_n \tag{14}$$

Получаем ленточную матрицу вида:

$$M = \begin{pmatrix} B_0 & A_0 + C_0 & 0 & \dots & 0 & 0 & 0 \\ A_1 & B_1 & C_1 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & A_{n-1} & B_{n-1} & C_{n-1} \\ 0 & 0 & 0 & \dots & 0 & A_n + C_n & B_n \end{pmatrix} \quad (15)$$

Для системы алгебраических уравнений вида $My = F$. Именно эту систему мы и будем решать методом прогонки. Подробно останавливаться на этом методе не будем, так как он рассматривался в одной из прошлых лабораторных работ.

Решение этой системы можно найти при помощи заполнения нулевого слоя следующим образом $y_{i,0} = \varphi_i$ и последующим использованием этих данных для дальнейших вычислений.

В случае если мы будем рассматривать уравнения (1) и (5), мы сможем получить решение этой же задачи путем деления найденной функции $w(x,t)$ (аналог функции $y(x,t)$, получаемой при рассмотрении (1) и (6)) на интеграл $I = \int_0^l w(x,t)$, вычисляемый для последнего слоя (этот подход обозначается как «Часть А»).

3. Руководство пользователя и пример работы с программой

При запуске программы перед пользователем появляется интерфейс управления.

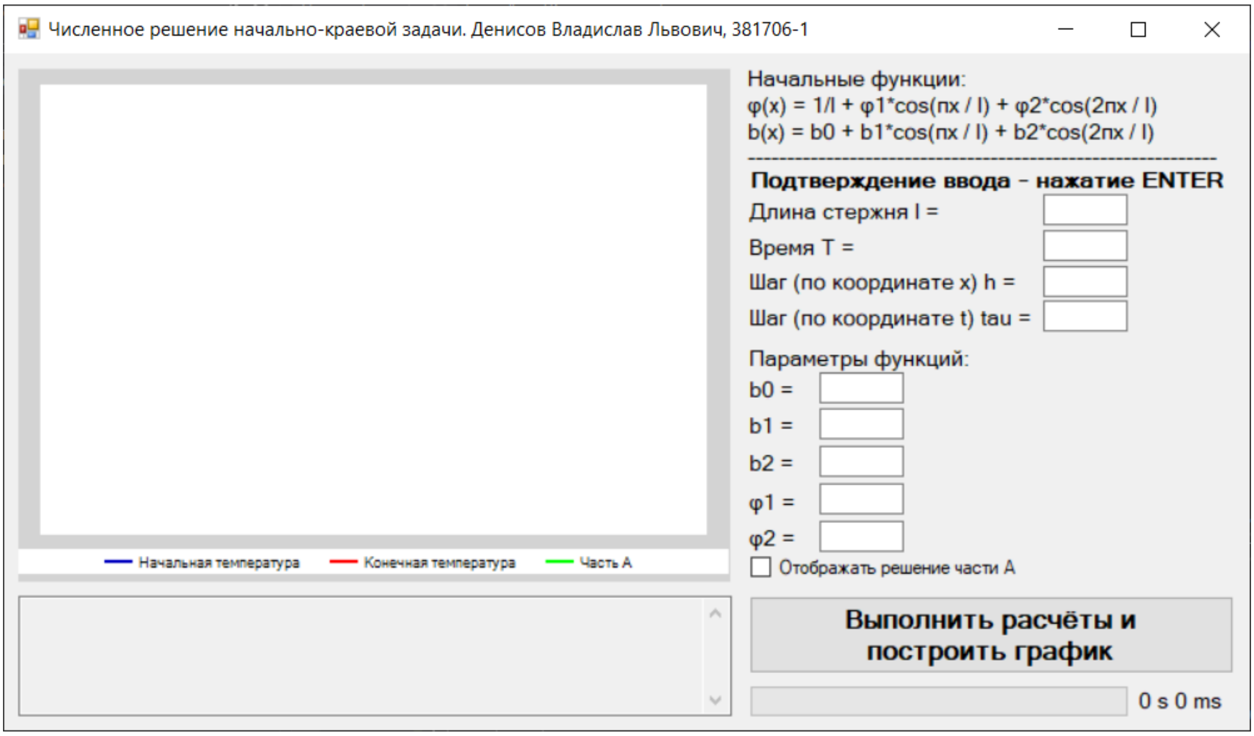

Рисунок 1 Первый запуск программы

Рассмотрим пример работы с программой.

Сначала заполним все поля, интересующими нас значениями. Это можно сделать путем ввода значения в соответствующее поле и нажатия на клавишу “Enter” на клавиатуре для подтверждения.

Все действия сопровождаются информационными сообщениями в специальном окне лога. Поэтому в случае ввода недопустимого значения программа сообщит об этом и даст возможность повторить действие.

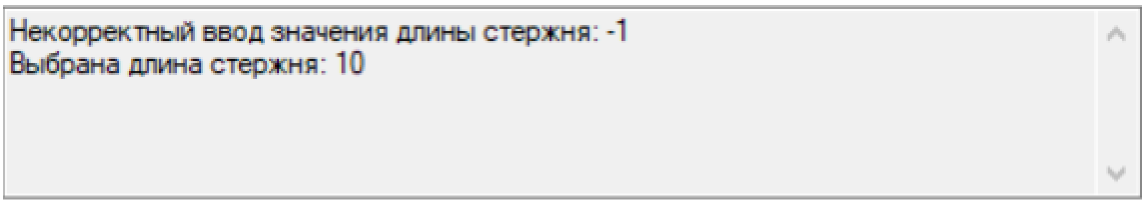

Рисунок 2 Ввод недопустимого значения

После заполнения всех полей следует нажать на кнопку «Выполнить расчёты и построить график». Результат можно увидеть на Рисунке 3.

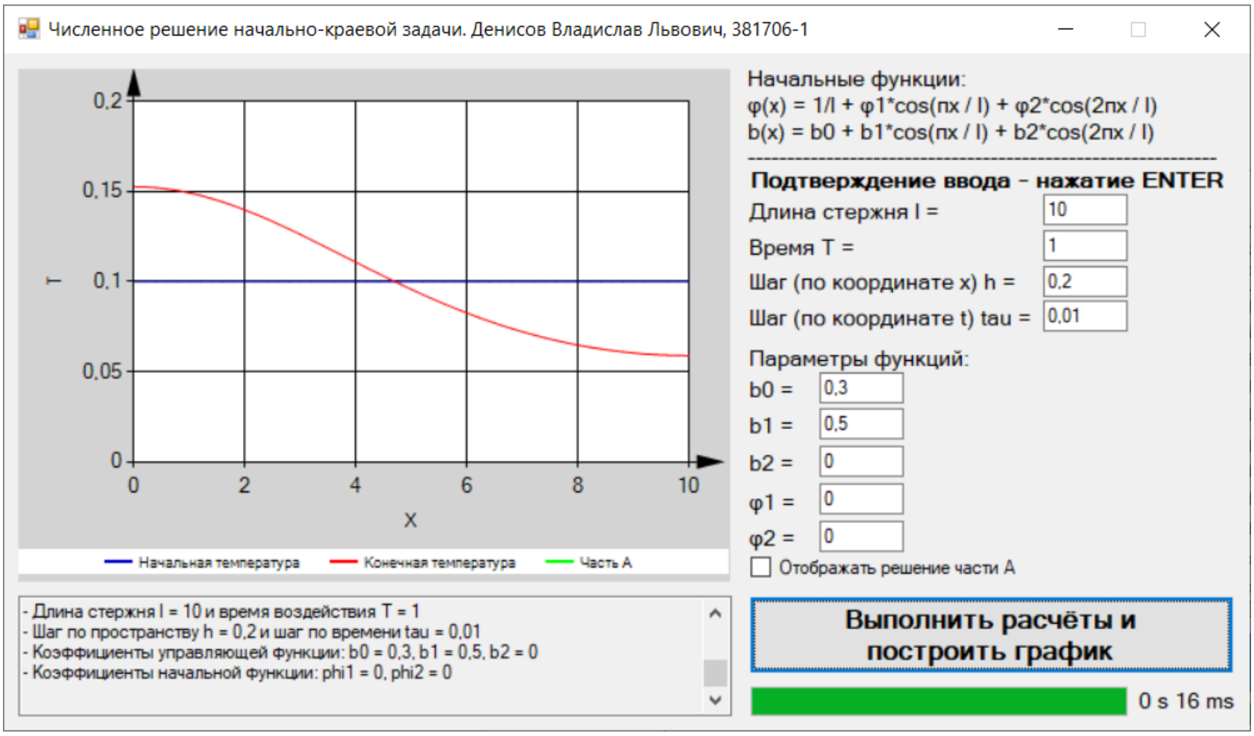

Рисунок 3 Построение решения задачи

При желании можно отобразить решение данной задачи для части А (см. подробности в разделе Решение задачи). Демонстрация на Рисунке 4.

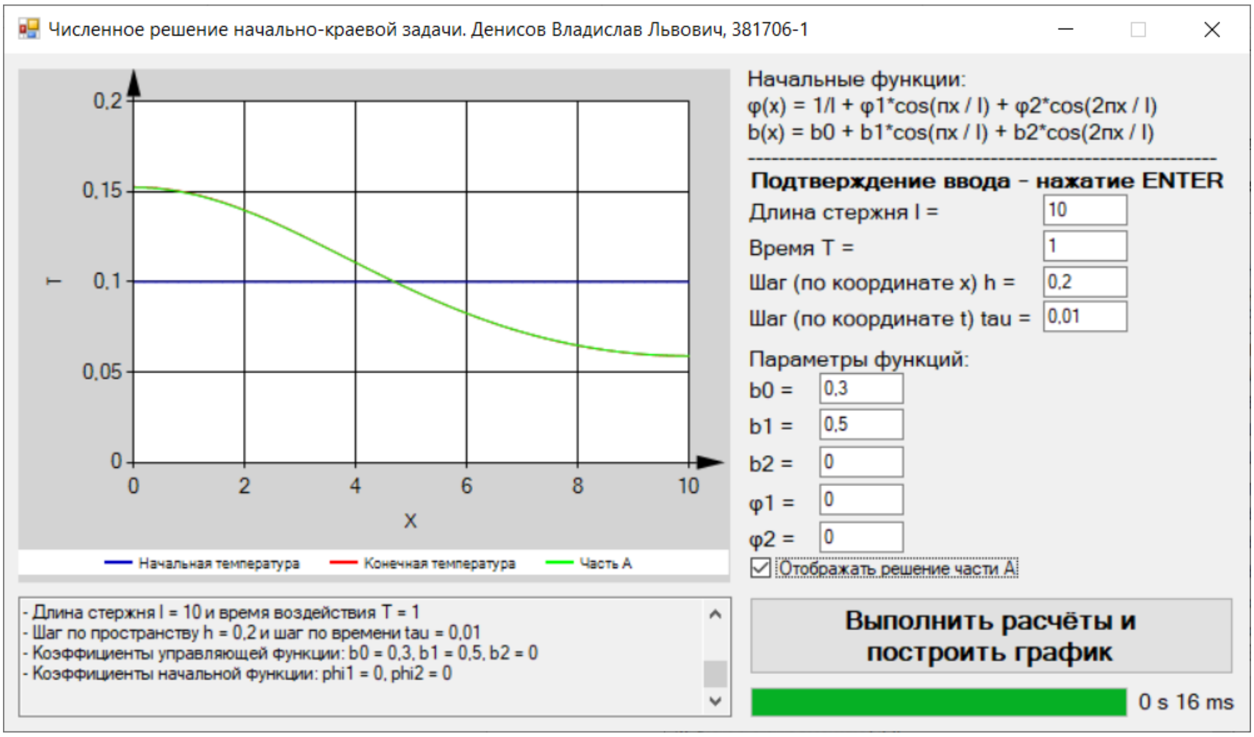

Рисунок 4 Режим отображения решения включая часть А

В идеале оба решения будут совпадать, что мы и наблюдаем на последних двух изображениях.

При желании можно построить решение для других начальных параметров. Для этого достаточно ввести данные заново и снова нажать на кнопку «Выполнить расчёты и построить график». Старое решение автоматически будет удалено и затем отобразятся новые результаты.

Пример для других параметров на Рисунке 5.

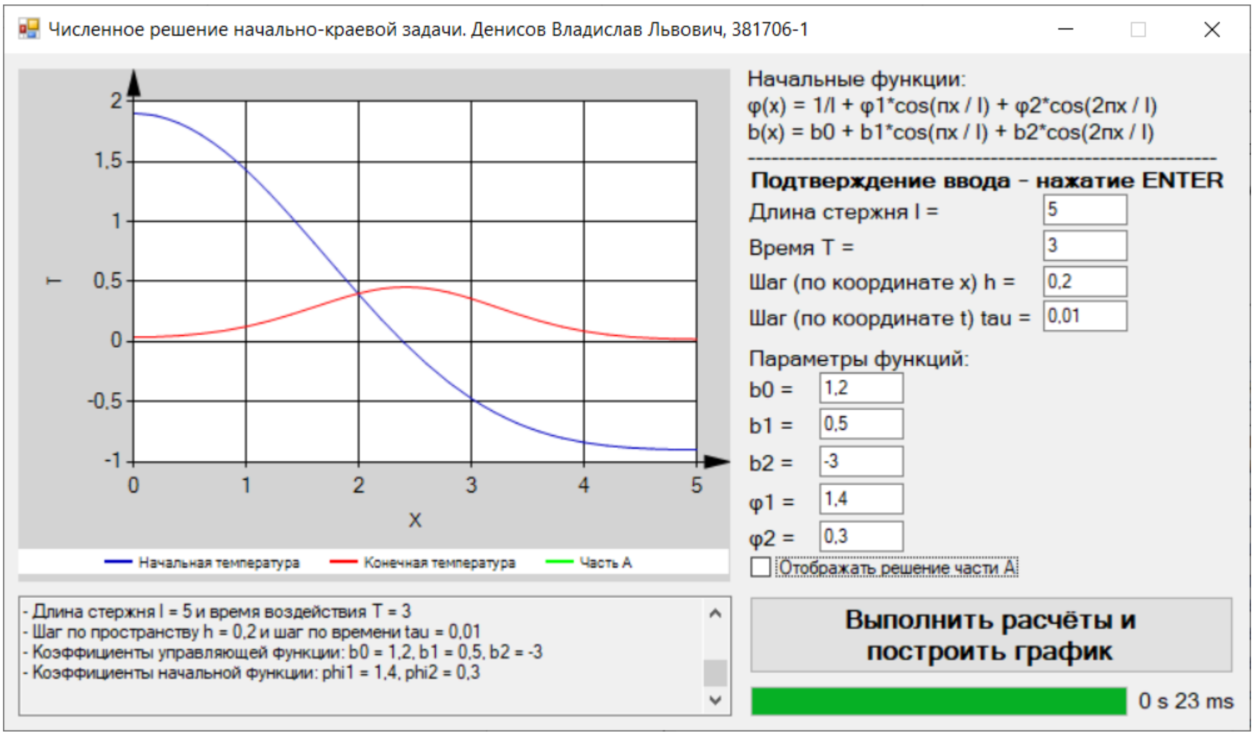

Рисунок 5 Построение решения задачи (другой случай)

4. Заключение

В результате лабораторной работы разработан программный комплекс, который позволяет найти численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных.

Программа выполняет поиск решения с учетом начальных данных, которые задаются пользователем вручную. Затем оно отображается на экране в графическом виде.

При этом, доступно сравнение полученного решения для двух различных функций управления с обратной связью путем установки соответствующего параметра в графическом интерфейсе.

Поддерживается возможность нахождения решения для других входных данных без перезапуска программы.

Цели, поставленные в лабораторной работе, успешно достигнуты.

5. Литература

1. Эгамов А.И. Лабораторная работа «Численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных: учебно-метод. пособие / А.И. Эгамов. – Нижний Новгород: Изд-во ННГУ, 2019. – 15 с.
2. Лабораторный практикум по численным методам: учеб.-метод. пособие / АлтГУ, Мат. фак., Каф. теорет. кибернетики и прикл. математики ; [сост.: В. В. Журавлева, С. С. Кузиков]. - Барнаул : Изд-во АлтГУ, 2015. - 31 с
3. Крайнов А.Ю., Миньков Л.Л. Численные методы решения задач тепло- и массопереноса : учеб. пособие.– Томск : STT, 2016. – 92 с.

6. Приложение

В данном разделе находится листинг основных функций, которые используются в программе.

```
class Thermal
{
	public double[,] grid;  // сетка для нахождения решения
	public double[,] grid_part_a;  // сетка для нахождения решения части А
	public double[] b;  // значения функции b для каждого слоя
	public double[] phi;  // значения фукнции фи для каждого слоя
	public double phi1, phi2;  // коэффициенты для функции phi
	public double b0, b1, b2;  // коэффициенты для функции b
	public double T;  // время воздействия
	public double L;  // длина стержня
	public double tau;  // величина шага по времени tau
	public double h;  // величина шага по длине стержня x
	public double coeff = 1.0;  // a в уравнении (1) в методичке
	public int TCount;  // число шагов по времени tau
	public int LCount;  // число шагов по длине стержня x

	// Функция phi(x) - начальное распределение температуры
	public double function_phi(double x)
	{
		return 1.0 / L + phi1 * Math.Cos(Math.PI * x / L) + phi2 *
			Math.Cos(2.0 * Math.PI * x / L);
	}

	// Функция b(x) - управляющая функция
	public double function_b(double x)
	{
		return b0 + b1 * Math.Cos(Math.PI * x / L) + b2 * Math.Cos(2.0 * Math.PI *
			x / L);
	}

	// Метод Симпсона для вычисления интеграла в части Б
	public double SimpsonMethod(int j)
	{
		double value = b[0] * grid[0, j];

		for (int i = 1; i < LCount - 1; i++)
		{
			if (i % 2 == 0)
				value += 2.0 * b[i + 1] * grid[i + 1, j];
			else
				value += 4.0 * b[i] * grid[i, j];
		}
		value += b[LCount - 1] * grid[LCount - 1, j];
		value = value * h / 3.0;

		return value;
	}

	// Метод Симпсона для вычисления интеграла в части А
	public double SimpsonMethod_W(double[,] w, int j)
	{
		double value = w[0, j];

		for (int i = 1; i < LCount - 1; i++)
		{
			if (i % 2 == 0)
```

```
                value += 2.0 * w[i + 1, j];
            else
                value += 4.0 * w[i, j];
        }
        value += w[LCount - 1, j];
        value = value * h / 3.0;

        return value;
    }

    // Метод прогонки для 3-х диагональной матрицы
    public double[] TridiagonalMatrixAlgorithm(double A, double B, double C, double
        AL, double C0, double[] F)
    {
        double[] y = new double[LCount];
        double[] alpha = new double[LCount];
        double[] beta = new double[LCount];

        alpha[0] = -1.0 * C0 / B;
        beta[0] = F[0] / B;

        for (int i = 1; i < LCount - 1; i++)
        {
            alpha[i] = -1.0 * C / (A * alpha[i - 1] + B);
            beta[i] = (F[i] - A * beta[i - 1]) / (A * alpha[i - 1] + B);
        }

        y[LCount - 1] = (F[LCount - 1] - AL * beta[LCount - 2]) / (AL *
                        alpha[LCount - 2] + B);

        for (int i = LCount - 2; i >= 0; i--)
            y[i] = alpha[i] * y[i + 1] + beta[i];

        return y;
    }

    // Основной алгоритм решения задачи, включающий вызов вспомогательных функций в
    требуемом порядке
    public void Algorithm(ref ProgressBar progressBar)
    {
        TCount = Convert.ToInt32(T / tau);
        LCount = Convert.ToInt32(L / h);
        if (LCount % 2 == 1) LCount++;

        progressBar.Minimum = 0;
        progressBar.Maximum = 2 * LCount + (TCount) * (2 * LCount);
        progressBar.Step = 1;

        grid = new double[LCount, TCount];
        grid_part_a = new double[LCount, TCount];
        b = new double[LCount];
        phi = new double[LCount];

        // Инициализация необходимых переменных
        double[] y;
        double[] y_part_a;
        double[] F = new double[LCount];
        double[] F_part_a = new double[LCount];

        // Первоначальная инициализация функций по интервалам сетки
        for (int i = 0; i < LCount; i++)
        {
            phi[i] = function_phi(i * h);
            b[i] = function_b(i * h);
```

```
                grid[i, 0] = phi[i];
                grid_part_a[i, 0] = phi[i];

                progressBar.PerformStep();
            }

            // Инициализация коэффициентов для метода прогонки
            double r = coeff * coeff * tau / (h * h);  // Выполняем замену для удобства
            double A = r;
            double B = -1.0 - 2.0 * r;
            double C = r;
            double AL = 2.0 * r;
            double C0 = 2.0 * r;

            // Решение задачи
            for (int j = 0; j < TCount - 1; j++)
            {
                double integral = SimpsonMethod(j);
                for (int i = 0; i < LCount; i++)
                {
                    F[i] = -1.0 * grid[i, j] * (1.0 + tau * b[i] - tau *
                        integral);
                    F_part_a[i] = -1.0 * grid_part_a[i, j] * (1.0 + tau * b[i]);

                    progressBar.PerformStep();
                }

                y = TridiagonalMatrixAlgorithm(A, B, C, AL, C0, F);
                y_part_a = TridiagonalMatrixAlgorithm(A, B, C, AL, C0, F_part_a);

                for (int i = 0; i < LCount; i++)
                {
                    grid[i, j + 1] = y[i];
                    grid_part_a[i, j + 1] = y_part_a[i];

                    progressBar.PerformStep();
                }
            }

            // Нахождения решения при помощи части А
            double square = SimpsonMethod_W(grid_part_a, TCount - 1);
            for (int i = 0; i < LCount; i++)
            {
                grid_part_a[i, TCount - 1] = grid_part_a[i, TCount - 1] / square;

                progressBar.PerformStep();
            }
        }  // function Algorithm
    }  // Class Thermal
```