МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
Высшего образования

«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Направление:
Фундаментальная информатика и информационные технологии

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

«ЧИСЛЕННОЕ РЕШЕНИЕ НАЧАЛЬНО-КРАЕВОЙ ЗАДАЧИ ДЛЯ ИНТЕГРО-ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ В ЧАСТНЫХ ПРОИЗВОДНЫХ»

Выполнил: студент группы 381706-1
Соколов А. Д.

_____Подпись

Принял:
Эгамов А. И.

_____Подпись

Нижний Новгород
2020.

1. Введение

В данной лабораторной работе мы рассматриваем 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. В ходе работы над ней нам нужно построить математическую модель процесса, вывести способ вычисления температуры стержня и реализовать это в специальной программе.

2. Постановка задачи

Целью данной лабораторной работы является нахождение решения данного дифференциального уравнения:

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

Областью определения данного уравнения являются следующие области:

$$x \in [0,l]; t \in [0,T]$$

Непрерывная функция $u(x,t)$ – управление с обратной связью, которое представляется для нашей лабораторной работы в виде:

$$u(x,t) = b(x)y(x-1,t) - y(x-1,t)\int_0^l b(x)y(x-1,t)\,dx$$

Также, для получения частных решений данного дифференциального уравнения, мы должны дополнить его начальными и краевыми условиями.

Функция начального распределения температуры:

$$\varphi(x) = \varphi_0 + \varphi_1 Cos\left(\frac{\pi x}{l}\right) + \varphi_2 Cos\left(\frac{2\,\pi x}{l}\right)$$

Управляющая функция:

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

Краевые условия:

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

Начальные условия:

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

На выходе необходимо получить изменение начальной кривой с течением времени, которое описывается данным дифференциальным уравнением.

3. Описание алгоритма

$y'_t(x,t)$ – типичное отыскание производной по определению. Отношение прироста функции ко времени.

$y''_{xx}(x,t)$– уравнение, использующие в себе несколько «слоев» решений. В данном случае, для нахождения решения следующего слоя мы будем использовать информацию не только о текущей точке X, но и о ее двух соседях.

$$y'_t(x,t) = \frac{y(i,j) - y(i,j-1)}{\tau}$$

$$y''_{xx}(x,t) = \frac{y(i-1,j) - 2 * y(i,j) + y(i+1,j)}{h^2}$$

Интеграл рассчитаем по формуле Симпсона:

$$\int_0^l b(x)dx \approx \frac{h}{6}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_3 + \cdots + 2y_{K-2} + 4y_{K-1} + y_K)$$

Раз мы разбиваем сетку с шагом h, то интеграл будем считать отдельно для каждого такого шага. Это увеличит точность расчетов, т.к. обычно шаг небольшой.

Итак, теперь у нас есть абсолютно все для численного решения данного ДУ. Просто подставим наши выкладки в основную формулу, но пока не будем раскрывать значения функций $b(x)$. После некоторого сокращения и приведения подобных имеем следующее:

$$-\tau * y(i-1,j) + (2\tau + h^2)y(i,j) - \tau * y(i+1,j) = \tau * h^2(y(x,t-1) * b(x) - y(x,t-1) * \int_0^l b(x)dx))$$

Для начального состояния:

$$(2\tau + h^2)y(i,j) - 2\tau * y(i+1,j) = \tau * h^2(y(x,t-1) * b(x) - y(x,t-1) * \int_0^l b(x)dx))$$

Для конечного состояния:

$$2\tau * y(i-1,j) - (2\tau + h^2)y(i,j) = \tau * h^2(y(x,t-1) * b(x) - y(x,t-1) * \int_0^l b(x)dx))$$

Таким образом получим систему, где в правой части находятся уже известные нам переменные. Соответственно, на каждом новом шаге можно просто рассчитать новые коэффициенты из значений старого слоя.

А вот для нового слоя мы будем иметь целую систему уравнений. В правой части которой и будут посчитанные константы.

$$d_i = \tau * h^2 * (y(x, t-1) * b(x) - y(x, t-1) * \int_0^l b(x)dx))$$

Теперь составим матрицу из этих уравнений для последующего решения методом прогонки, изученный нами в ходе первой лабораторной работы «Интерполяция сплайнов».

$$\left(\begin{array}{cccccc|c} (2\tau + h^2) & (-2\tau) & 0 & 0 & 0 & 0 & d_0 \\ (-\tau) & (2\tau + h^2) & (-\tau) & 0 & 0 & 0 & d_2 \\ 0 & (-\tau) & (2\tau + h^2) & 0 & 0 & 0 & d_3 \\ & \vdots & & \ddots & \vdots & & \vdots \\ 0 & 0 & 0 & (2\tau + h^2) & (-\tau) & 0 & d_{l-2} \\ 0 & 0 & 0 & (-\tau) & (2\tau + h^2) & (-\tau) & d_{l-1} \\ 0 & 0 & 0 & 0 & (-2\tau) & (2\tau + h^2) & d_l \end{array}\right)$$

Зная решение нулевого слоя (начальное условие), сможем построить первый. И так, итеративно будем рассчитывать слой, который нужен пользователю. То есть, постоянно добавляя в правую часть системы решение прошлого слоя, можно найти решение следующего слоя.

4. Руководство пользователя

После запуска программы пользователь должен ввести параметры системы:

1. длины стержня l; времени T
2. шага h в разностной схеме по координате x;
3. шага τ в разностной схеме по координате t;
4. константы b_0 , b_1 , b_2, φ_1, φ_2

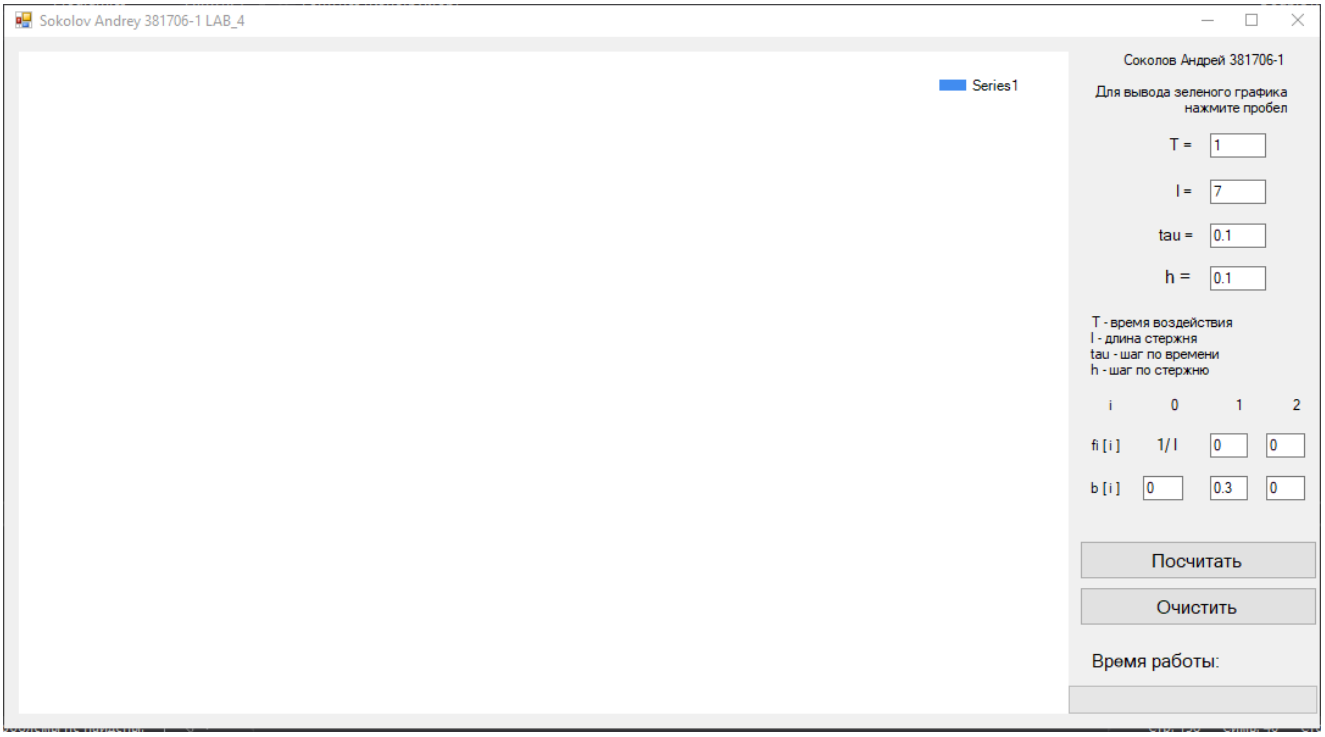

Рисунок 1. Программа с параметрами по умолчанию

После нажатия кнопки «посчитать» на координатной плоскости появится два графика

Синий – начальное распределение температуры

Красный – конечное распределение температуры в момент времени T

При нажатии кнопки пробел будет выведен зелёный график – конечное распределение температуры при решении линейной задачи.

Также на экран будет выведено время, затраченное на вычисление температуры. Время, затраченное на отображение графика на экран не учитывается.

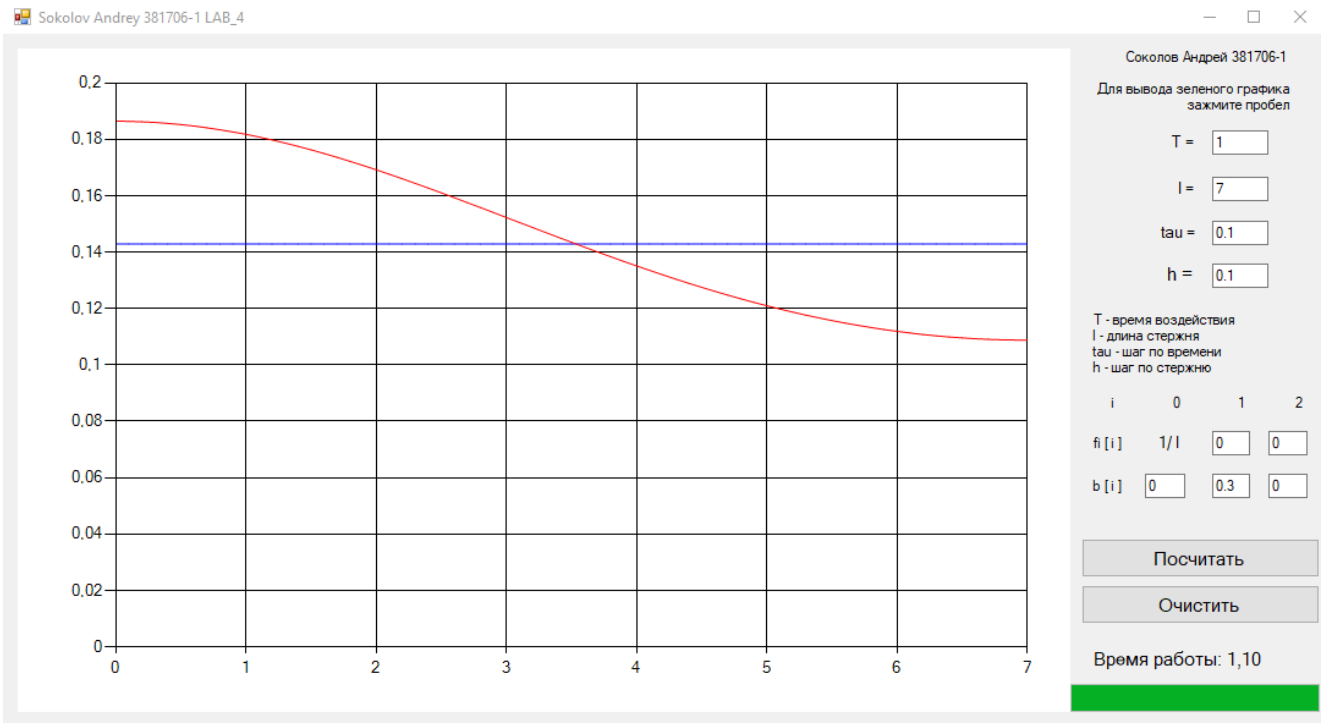

Рисунок 2. Результат работы программы

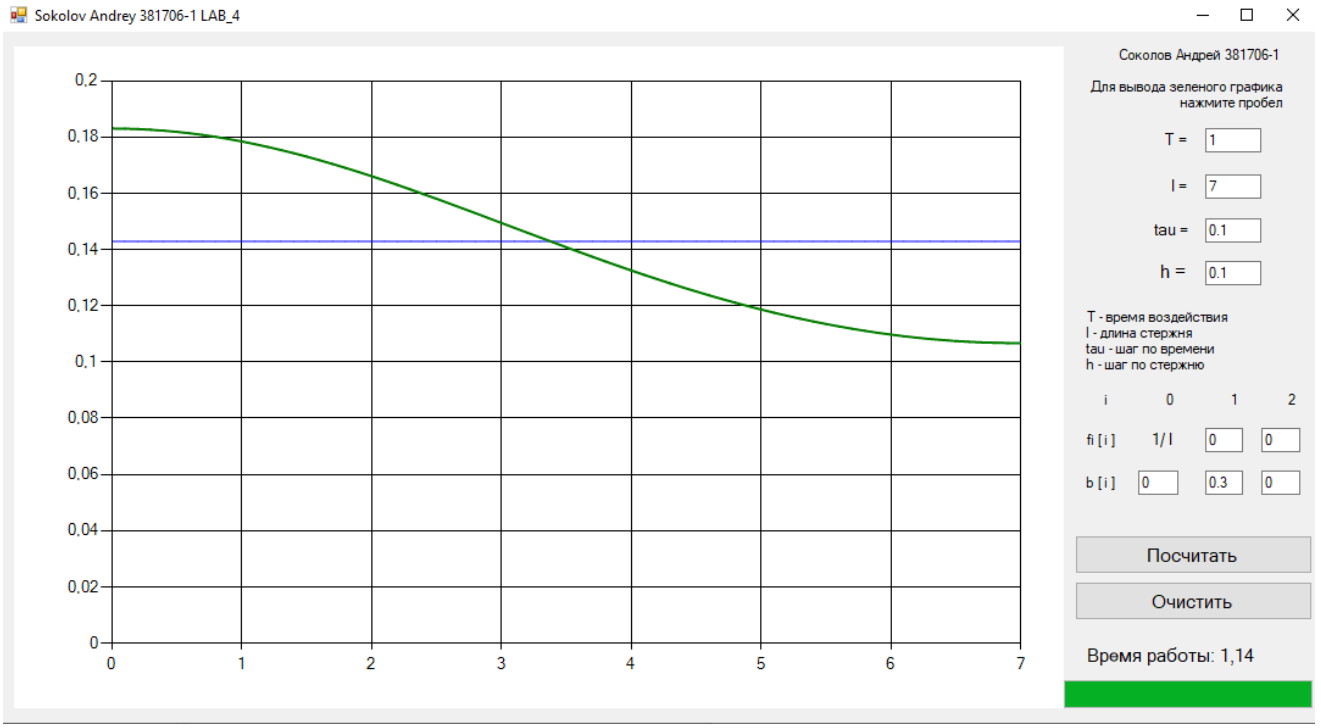

Рисунок 3. Вывод зеленого графика по нажатию кнопки пробел

5. Заключение

В данной работе была реализована программа, производящая вычисление температуры стержня заданной длины под нагревом. Была исследована поставленная задача, подробно выведен метод ее решения с помощью численных методов. Была реализована программа с отображением графиков температур и возможностью задать параметры динамической системы.

6. Список литературы

1. Самарский А.А. Введение в численные методы – М.: Наука, 1989. – 271с.
2. Эгамов А.И. Лабораторная работа «Численное решение начально-краевой задачи для интегро-дифференциального уравнения в частных производных» – 15с.

7. Приложение

В коде не приводятся сервисные стейтменты для вычисления время работы, шкалы прогресса и т. п.

```
// коэффициенты функции начального распределения
double phi_a;
double phi_b;
double phi_c;

// коэффициенты функции воздействия на стержень
double b_a;
double b_b;
double b_c;

int T; // время воздействия на стержень
int l; // длина стержня
double h; // длина шага по x
double tau; // длина шага по времени
int xSteps; // количество шагов по x
int tSteps; // количество шагов по времени

double[,] yRes; // массив температур в каждый момент времени в каждой точке по x


// Функция начального распределения температуры
double FuncPhi(double x)
{
        return phi_a + phi_b * Math.Cos(Math.PI * x / l) + phi_c * Math.Cos(2.0 * Math.PI
* x / l);
}

// Функция, определяющая воздействие на стержень
double FuncB(double x)
{
        return b_a + b_b * Math.Cos(Math.PI * x / l) + b_c * Math.Cos(2.0 * Math.PI * x /
l);
}

// Вычисление интеграла методом Симпсона в формуле управления
private double SimpsonMethod(int t)
{
        double x = h;
        double res = FuncB(0) * yRes[0, t];
        for (int i = 1; i < xSteps - 1; ++i)
        {
                res += FuncB(x) * ((i % 2 == 0) ? 2 : 4) * yRes[i, t];
                x += h;
        }
        res += FuncB(x) * yRes[xSteps - 1, t];
        return res * (h / 3);
}

// Решенне СЛАУ трехдиагональной матрицы методом прогонки
public void Solve(double[] downDiagonal, double[] midDiagonal, double[] upDiagonal,
double[] d, int t)
{
        int length = midDiagonal.Length;

        if (length == downDiagonal.Length + 1 && length == upDiagonal.Length + 1 && length
== d.Length)
        {
                double[] alpha = new double[length - 1];
                double[] beta = new double[length - 1];

                alpha[0] = -upDiagonal[0] / midDiagonal[0];
                beta[0] = d[0] / midDiagonal[0];
```

```
                for (int i = 1; i < length - 1; i++)
                {
                        double gamma = midDiagonal[i] + downDiagonal[i - 1] * alpha[i -
1];
                        alpha[i] = -upDiagonal[i] / gamma;
                        beta[i] = (d[i] - downDiagonal[i - 1] * beta[i - 1]) / gamma;
                }
                yRes[length - 1, t] = (d[length - 1] - downDiagonal[length - 2] *
beta[length - 2]) / (midDiagonal[length - 1] + downDiagonal[length - 2] * alpha[length -
2]);

                for (int i = length - 2; i >= 0; i--)
                {
                        yRes[i, t] = alpha[i] * yRes[i + 1, t] + beta[i];
                }
        }
}


// функция вычисления температуры (нажатие кнопки вычислить)
private void Run_Click(object sender, EventArgs e)
{
        yRes = new double[xSteps, tSteps];
        var timer = System.Diagnostics.Stopwatch.StartNew();
        double x = 0;

        // Начальное распределение температуры
        for (int idx = 0; idx < xSteps; ++idx)
        {
                yRes[idx, 0] = FuncPhi(x);
                x += h;
        }
        // инициализация коэффициентов трехдиагональной матрицы
        double[] upDiagonal = new double[xSteps - 1];
        double[] midDiagonal = new double[xSteps];
        double[] downDiagonal = new double[xSteps - 1];

        for (int idx = 0; idx < xSteps - 1; ++idx)
        {
                if (idx == 0)
                        upDiagonal[idx] = -2 * tau;
                else
                        upDiagonal[idx] = -1 * tau;
        }
        for (int idx = 0; idx < xSteps; ++idx)
        {
                midDiagonal[idx] = h * h + 2 * tau;
        }
        for (int idx = 0; idx < xSteps - 1; ++idx)
        {
                if (idx == xSteps - 2)
                        downDiagonal[idx] = -2 * tau;
                else
                        downDiagonal[idx] = -1 * tau;
        }


        // Вычисление температуры
        for (int t = 1; t < tSteps; ++t)
        {
                x = 0;
                // Вычисление вектора правой части СЛАУ - d
                double[] d = new double[xSteps];
                double simpson = SimpsonMethod(t - 1);
                for (int idx = 0; idx < xSteps; ++idx)
                {
                        d[idx] = * h * h * tau *
(yRes[idx, t - 1] * FuncB(x) - * yRes[idx, t - 1] simpson);
```

```
                x += h;
            }

            // Решение СЛАУ методом прогонки
            Solve(downDiagonal, midDiagonal, upDiagonal, d, t);

            }
        }
```