МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет**

**им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Кафедра дифференциальных уравнений, математического и численного анализа**

**ЛАБОРАТОРНАЯ РАБОТА**

**«Численное решение начально-краевой задачи для**

**интегро-дифференциального уравнения в частных производных»**

**Выполнил:** студент группы 381706-2

Танский Юрий Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Эгамов Альберт Исмаилович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2020

**Оглавление**

[**Введение** 3](#_Toc40845123)

[**Описание управляемого процесса** 4](#_Toc40845124)

[**Описание вычислительной задачи** 5](#_Toc40845125)

[**Руководство программиста** 8](#_Toc40845126)

[**Руководство пользователя** 9](#_Toc40845127)

[**Практическое обоснование** 12](#_Toc40845128)

[**Заключение** 13](#_Toc40845129)

[**Список литературы** 14](#_Toc40845130)

[**Листинг основных методов** 15](#_Toc40845131)

# **Введение**

Для того чтобы описать физический процесс, необходимо, кроме самого уравнения, определяющего закон его развития, задать начальное состояние параметров этого процесса *(начальные условия)* и режим на границе той области, в которой он происходит *(граничные условия).* Это позволяет выбрать из многих возможностей интересующую нас реализацию физического процесса (в силу этого иногда граничные и начальные условия называют *условиями однозначности* решения).

Начальные и граничные условия в своей совокупности называются *краевыми условиями.* Соответствующим образом поставленная задача, включающая в себя дифференциальное уравнение (или систему дифференциальных уравнений) и краевые условия, называется *краевой задачей.*

# **Описание управляемого процесса**

Рассмотрим в качестве примера управляемый процесс нагревания однородного стержня длины с теплоизолированными концами.

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 найдем непрерывно дифференцируемую по и дважды непрерывно дифференцируемую по функцию – температуру стержня, являющуюся решением уравнения

|  |  |  |
| --- | --- | --- |
|  |  |  |

и удовлетворяющую однородным граничным условиям второго рода

|  |  |
| --- | --- |
|  |  |

и начальному условию

|  |  |
| --- | --- |
|  |  |

где – константа, – задает начальное распределение температуры, дважды непрерывно дифференцируема и удовлетворяющая условию:

|  |  |
| --- | --- |
|  |  |

непрерывная функция – управление с обратной связью, представляющаяся в виде

|  |  |
| --- | --- |
|  |  |

или

|  |  |
| --- | --- |
|  |  |

где – непрерывная на управляющая функция.

# **Описание вычислительной задачи**

Для того, чтобы выполнить численное решение начально-краевой задачи для

интегро-дифференциального уравнения в частных производных, проведем следующую цепочку действий:  
Составим неявную разностную схему с погрешностью



Определим нулевой слой для будущей разностной схемы из функции



В качестве функции b(x) используем функцию



Где b0, b1, b2 , ϕ1 , ϕ2 – некие константы, которые задаются в начале работы алгоритма.  
Перед вычислением каждого следующего слоя находим интеграл в (6) для значений последнего известного слоя по формуле Симпсона:

,

– количество шагов по , предполагается чётным.

Составим неявную разностную схему с погрешностью :

|  |  |
| --- | --- |
|  | **(7)** |

Необходимо проверить, что для обеспечения устойчивости разностной схемы.

Составим трехточечные разностные производные первого порядка для краевых условий с погрешностью второго порядка. В виде разностных производных краевые условия выглядят следующим образом:

Уравнение (1) преобразуем к виду

и, подставив вторую производную в выражение

получаем

|  |  |
| --- | --- |
|  | **(8)** |

Для правой границы аналогично получаем

|  |  |
| --- | --- |
|  | **(9)** |

(7)-(9) представляют собой систему из уравнения. Теперь нужно привести эту систему к трехдиагональному виду.

Для части A:

Для части B:

Осталось решить систему методом прогонки:

Представим систему в виде . Для удобства записи опустим верхние индексы (решаем систему для фиксированного слоя).

Идея метода прогонки – следующее предположение:

|  |  |
| --- | --- |
|  | **(10)** |

Выразив и через и подставив в исходный вид системы, получаем

что будет выполняться независимо от в случае

⇒

Так как

Теперь можно найти все прогоночные коэффициенты.

Последняя компонента решения:

Остальные находим из (10).

Чтобы получить из решения части A решение части B, нужно разделить полученную функцию на ее интеграл от до, вычисленный по формуле Симпсона

# **Руководство программиста**

В качестве среды разработки использовался программный продукт Visual Studio 2019, программа написана на языке C# с использованием включенного в него пакета Windows Forms (.Net Framework)

Для работоспособности программного кода необходимо подключить следующие стандартные библиотеки, включенные в пакет C# VS 2019:  
using System;  
using System.Windows.Forms;

using System.Globalization;  
Дополнительные внешние компоненты в программе не используются.

Параметры алгоритма:

* double L – длина стержня
* double τ - шаг по времени
* double T – время воздействия
* double h – шаг, используемые при расчете по длине стержня
* double b0, b1, b2 - параметры управляющей функции
* ϕ1, ϕ2 – константы, параметры начального распределения температуры

Ниже приведено описание основных методов, используемых при реализации алгоритма.  
private void CalculateResult()

Является основным методом, вызываемым при работе программы. В ней вычисляется нулевой и последующие слои для будущей разностной схемы, также метод использует процедуры

double IntegrateA

double IntegrateB

Для вычисления интегралов по формуле Симпсона

# **Руководство пользователя**

При начале работы программы пользователю открывается интуитивно понятный интерфейс (рис.1)

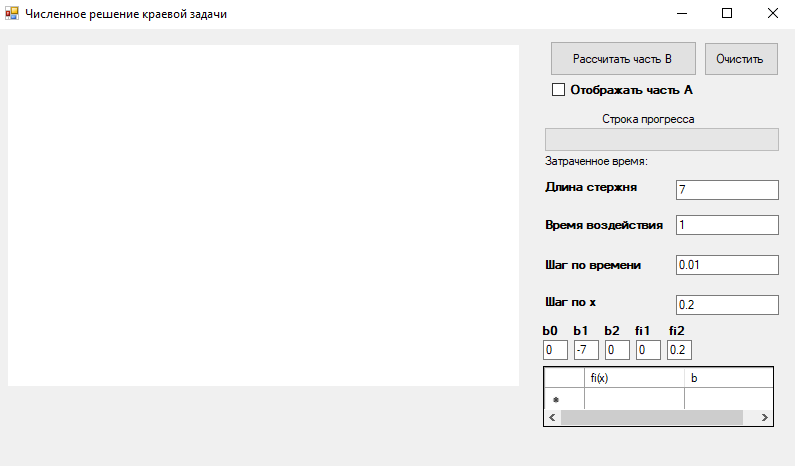


Рисунок 1. Начало работы программы.

Параметры алгоритма заполняются по умолчанию при старте работы программы, но можно задать свои параметры. В случае, если пользователь неверно задаст один из параметров, программа отловит ошибку при вводе входных данных и выдаст соответствующее сообщение (рис.2)

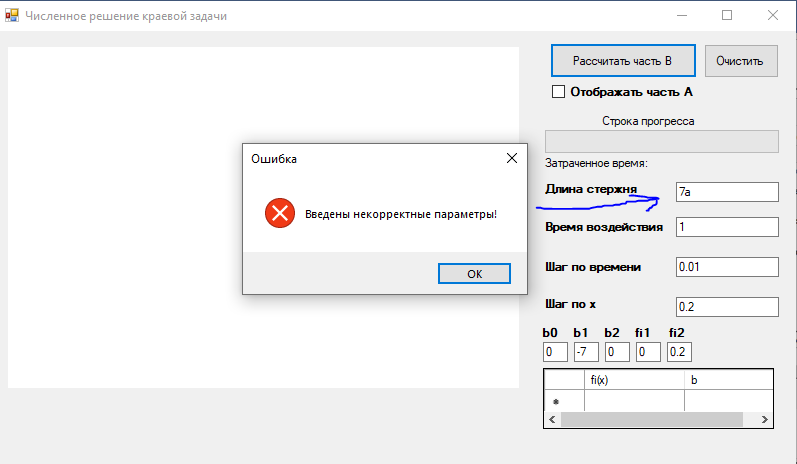


Рисунок 2. Неверный ввод параметров.

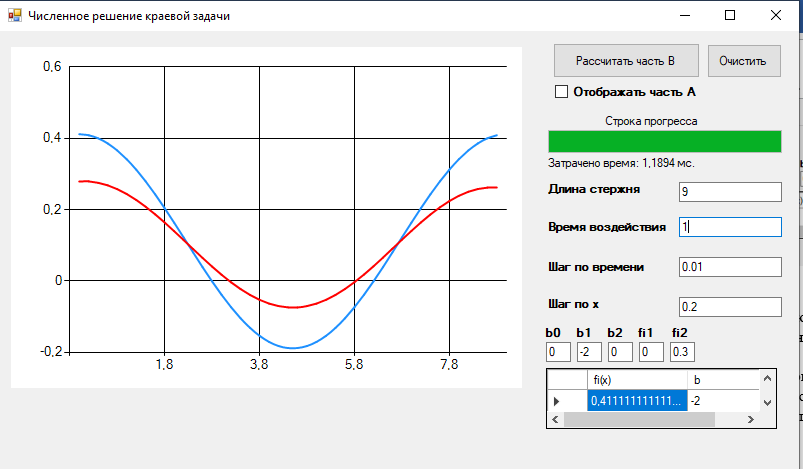
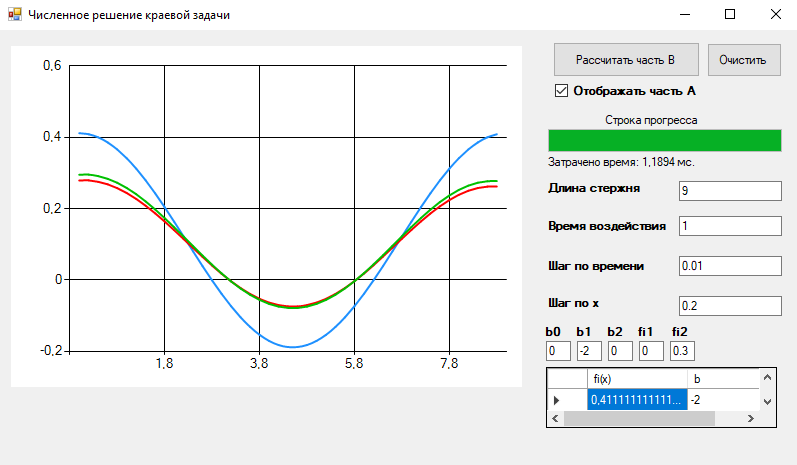
После того, как все входные параметры будут введены, пользователю необходимо нажать кнопку “Рассчитать часть B”, после чего программа начнет исполнение алгоритма и отобразит на графике результат.  
Также в боковой панеле формы отображается индикатор, показывающий в реальном времени прогресс выполнения алгоритма. По окончанию работы алгоритма пользователь увидит, сколько по времени (в мс.) происходило выполнение алгоритма.  
график функции ϕ(x) выводится синим цветом  
график функции y(x,T ) - красным цветом (рис.3)  


Рисунок 3. Демонстрация работы алгоритма.

Пользователь может вывести график функции w(x,T), разделенной на интеграл от этой функции по формуле Симпсона. Для этого после выполнения алгоритма пользователю необходимо нажать на соответствующий чек-бокс “Отображать часть А” (рис.4)  
  
Рисунок 4. Отображение решения задачи части A

# **Практическое обоснование**

1. На концах отрезка в силу график функции численного решения имеет горизонтальные касательные (рис. 3)

2. Площадь фигуры, где график функции ϕ(x) выше, чем y(x,T) равна площади фигуры, где функция ϕ(x) ниже, чем y(x,T) , то есть, S1 = S2 (рис. 3)

3. При замене функции b(x) на b(x) + b0 , где b0‒ некоторая константа, функция y(x,T ) не изменяется. В качестве b0 возьмем значение 5. (рис.5)

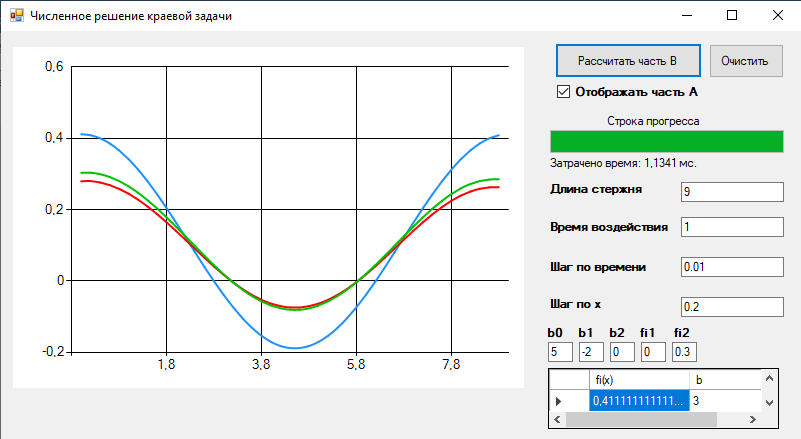


Рисунок 5. Подстановка константы b0

4. Зеленый график находится близко к красному графику (рис. 4, 5)

# **Заключение**

Был разработан алгоритм численного решения начально-краевой задачи для интегро-дифференциального уравнения, описывающего управляемый процесс нагревания тонкого однородного стержня с теплоизолированными концами.

Был проведен анализ полученного результата, в котором мы убедились, что алгоритм удовлетворяет теоремам, применяемым к решению начально-краевой задачи.

Для удобства работы был разработан понятный, “дружеский” интерфейс для использования алгоритма численного решения начально-краевой задачи.

# **Список литературы**

1. Хайрер Э., Нёрсетт С., Ваннер Г. «Решение обыкновенных дифференциальных уравнений. Нежесткие задачи», М., Мир, 1990, 512с.

2. А. А. Самарский “Введение в численные методы”. Издание третье., Москва, “Лань”, 2005, 288с.

# **Листинг основных методов**

private void CalculateResult()

{

CountOfStepsX = Convert.ToInt32(L / h);

CountOfStepsT = Convert.ToInt32(t / tau);

resultB = new double[CountOfStepsT, CountOfStepsX];

resultA = new double[CountOfStepsT, CountOfStepsX];

double[] beta = new double[CountOfStepsX];

double[] alpha = new double[CountOfStepsX];

double[] b\_local\_array = new double[CountOfStepsX];

double[] x = new double[CountOfStepsX];

progressBar1.Maximum = CountOfStepsT;

var timer = System.Diagnostics.Stopwatch.StartNew();

for (int i = 0; i < CountOfStepsX; ++i)

{

resultB[0,i] = fi(i \* h);

resultA[0,i] = fi(i \* h);

b\_local\_array[i] = bi(i \* h);

}

double[] a = new double[CountOfStepsX];

double[] b = new double[CountOfStepsX];

double[] c = new double[CountOfStepsX];

double coef = 1;

a[0] = 0.0;

b[0] = 1.0;

c[0] = -1.0;

a[1] = tau \* coef \* coef / Math.Pow(h, 2);

b[1] = -2.0 \* tau \* coef \* coef / Math.Pow(h,2) - 1.0;

c[1] = tau \* coef \* coef / Math.Pow(h,2);

a[2] = -1.0; b[2] = 1.0; c[2] = 0.0;

for (int j = 1; j < CountOfStepsT; ++j)

{

double integral = IntegrateB(b\_local\_array, j - 1);

for (int i = 1; i < CountOfStepsX - 1; i++)

{

SolveAlphaBeta(ref alpha, ref beta, j, i, b\_local\_array, integral);

}

beta[0] = beta[CountOfStepsX - 1] = 0;

alpha[0] = alpha[CountOfStepsX - 1] = 0;

RunMatrixAlg(a, b, c, beta, ref x);

for (int i = 0; i < CountOfStepsX; i++)

resultB[j,i] = x[i];

RunMatrixAlg(a, b, c, alpha, ref x);

for (int i = 0; i < CountOfStepsX; i++)

resultA[j,i] = x[i];

progressBar1.Value++;

}

result\_test\_A = new double[CountOfStepsX];

double Aintegral = IntegrateA( CountOfStepsX - 1);

for (int i = 0; i < CountOfStepsX; i++)

result\_test\_A[i] = resultA[CountOfStepsT - 1,i] / Aintegral;

progressBar1.Value = CountOfStepsT;

timer.Stop();

double time = (timer.Elapsed).TotalMilliseconds;

label5.Text = "Затрачено время: " + Convert.ToString(time) + " мс.";

}

private double IntegrateB(double[] b\_local\_array, int j)

{

double integral = resultB[j,0] \* b\_local\_array[0];

for (int i = 1; i < CountOfStepsX - 1; i += 2)

{

integral += 4.0 \* resultB[j,i] \* b\_local\_array[i];

integral += 2.0 \* resultB[j,i + 1] \* b\_local\_array[i + 1];

}

integral += resultB[j,CountOfStepsX - 1] \* b\_local\_array[CountOfStepsX - 1];

integral = integral \* h / 3.0;

return integral;

}

double IntegrateA(int j)

{

double integral = resultA[j,0];

for (int i = 1; i < CountOfStepsX - 1; i += 2)

{

integral += 4.0 \* resultA[j,i];

integral += 2.0 \* resultA[j,i + 1];

}

integral += resultA[j,CountOfStepsX - 1];

integral = integral \* h / 3.0;

return integral;

}