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

Институт информационных технологий, математики и механики

Кафедра дифференциальных уравнений, математического и численного анализа

ОТЧЕТ

по учебной практике
на тему:

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

Выполнила:
студентка группы 381706-1
Кукушкина Ксения Олеговна

_____
(подпись)

Проверил:
старший преп. каф. ДУМЧА
Эгамов Альберт Исмаилович

_____
(подпись)

Нижний Новгород
2020

Содержание

1. Введение.....3
2. Теоретическое обоснование.....4
3. Описание программы.....6
3.1 Руководство программиста.....6
3.2 Руководство пользователя.....8
4. Доказательство корректности.....11
5. Вычислительный эксперимент.....14
6. Вывод.....15
7. Список литературы.....16
Приложение.....17

1. Введение

Решение начально-краевой задачи для дифференциального уравнения состоит в нахождении его решения, удовлетворяющего некоторым начальным и граничным условиям (задают поведение дифференциального уравнения в начальный момент времени и на границе рассматриваемой области соответственно).

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

Задача: на множестве $Q = [0, l] \times [0, T], l > 0, T > 0$ найти непрерывно дифференцируемую по t и дважды непрерывно дифференцируемую по x функцию $y(x, t)$ – температуру стержня, являющуюся решением уравнения

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

и удовлетворяющую однородным граничным условиям второго рода

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

и начальному условию

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

где a – константа, $\varphi(x) > 0$ – дважды непрерывно дифференцируемая на отрезке функция, задающая начальное распределение температуры и удовлетворяющая условиям согласования (3) и условию

$$\int_0^l \varphi(x)\, dx = 1\,, \quad \textbf{(4)}$$

непрерывная функция $u(x, t)$ – управление с обратной связью, представляющаяся в виде

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

или

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

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

2. Теоретическое обоснование

Определим нулевой слой будущей разностной схемы из (3). В качестве начальной функции берем

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

Перед вычислением каждого следующего слоя находим интеграл в (6) для значений последнего известного j слоя по формуле Симпсона:

$$I_j = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_3 + \cdots + 2y_{K-2} + 4y_{K-1} + y_K),$$

$K = \frac{l}{h}$ – количество шагов по x, предполагается чётным.

Составим неявную разностную схему с погрешностью $O(\tau + h^2)$:

$$\frac{y_k^{n+1} - y_k^n}{\tau} = \frac{y_{k+1}^{n+1} - 2y_k^{n+1} + y_{k-1}^{n+1}}{h^2} + u_k^n. \quad \textbf{(7)}$$

Необходимо проверить, что $\frac{\tau}{h^2} < \frac{1}{4}$ 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:

$$\frac{y_1^{n+1} - y_0^{n+1}}{h} = \frac{y_K^{n+1} - y_{K-1}^{n+1}}{h} = 0$$

Уравнение (1) преобразуем к виду

$$\frac{\partial^2 y}{\partial x^2} = \frac{\partial y}{\partial \tau} - u(x,\tau)$$

и, подставив вторую производную в выражение

$$\frac{y_1 - y_0}{h} = \frac{\partial y(o,\tau)}{\partial x} + \frac{h}{2}\left(\frac{\partial^2 y}{\partial x^2}\right)_{x=0} + O(h^2),$$

получаем

$$\frac{y_1^{n+1} - y_0^{n+1}}{h} - \frac{h}{2}\frac{y_0^{n+1} - y_0^n}{\tau} + \frac{h}{2}u_0^n = 0. \quad \textbf{(8)}$$

Для правой границы аналогично получаем

$$\frac{y_K^{n+1} - y_{K-1}^{n+1}}{h} + \frac{h}{2}\frac{y_K^{n+1} - y_K^n}{\tau} - \frac{h}{2}u_0^n = 0. \quad \textbf{(9)}$$

(7)-(9) представляют собой систему из $K + 1$ уравнения. Теперь нужно привести эту систему к трехдиагональному виду.

Для части A:

$$\begin{cases} (2\tau + h^2)y_0^{n+1} + (-2\tau)y_1^{n+1} = h^2 y_k^n(1 + \tau b_k) \\ (-\tau)y_{k-1}^{n+1} + (2\tau + h^2)y_k^{n+1} + (-\tau)y_{k+1}^{n+1} = h^2 y_k^n(1 + \tau b_k), \quad k = \overline{1, K-1} \\ (-2\tau)y_{K-1}^{n+1} + (2\tau + h^2)y_K^{n+1} = h^2 y_k^n(1 + \tau b_k) \end{cases}$$

Для части B:

$$\begin{cases} (2\tau + h^2)y_0^{n+1} + (-2\tau)y_1^{n+1} = h^2 y_k^n\big(1 + \tau(b_k - I_n)\big) \\ (-\tau)y_{k-1}^{n+1} + (2\tau + h^2)y_k^{n+1} + (-\tau)y_{k+1}^{n+1} = h^2 y_k^n\big(1 + \tau(b_k - I_n)\big), \quad k = \overline{1, K-1} \\ (-2\tau)y_{K-1}^{n+1} + (2\tau + h^2)y_K^{n+1} = h^2 y_k^n\big(1 + \tau(b_k - I_n)\big) \end{cases}$$

Осталось решить систему методом прогонки:

Представим систему в виде $A_k y_{k-1}^{n+1} + B_k y_k^{n+1} + C_k y_{k+1}^{n+1} = F_k$. Для удобства записи опустим верхние индексы (решаем систему для фиксированного слоя).

Идея метода прогонки – следующее предположение:

$$y_k = \alpha_{k+1} y_{k+1} + \beta_{k+1}, \quad k = \overline{K-1, 0} \tag{10}$$

Выразив y_k и y_{k-1} через y_{k+1} и подставив в исходный вид системы, получаем

$$(A_k\alpha_k\alpha_{k+1} + B_k\alpha_{k+1} + C_k)y_{k+1} + A_k\alpha_k\beta_{k+1} + A_k\beta_k + B_k\beta_{k+1} - F_k = 0\,,$$

что будет выполняться независимо от y в случае

$$\begin{cases} A_k\alpha_k\alpha_{k+1} + B_k\alpha_{k+1} + C_k = 0 \\ A_k\alpha_k\beta_{k+1} + A_k\beta_k + B_k\beta_{k+1} - F_k = 0 \end{cases} \Rightarrow \begin{cases} \alpha_{k+1} = \frac{-C_k}{A_k\alpha_k + B_k} \\ \beta_{k+1} = \frac{F_k - A_k\beta_k}{A_k\alpha_k + B_k} \end{cases}$$

Так как $A_0 = 0$,

$$\begin{cases} \alpha_1 = \dfrac{-C_0}{B_0} \\ \beta_1 = \dfrac{F_0}{B_0} \end{cases}$$

Теперь можно найти все прогоночные коэффициенты.

Последняя компонента решения:

$$y_K = \frac{F_K - A_K\beta_K}{B_K + A_K\alpha_k}$$

Остальные находим из (10).

Чтобы получить из решения части A решение части B, нужно разделить полученную функцию на ее интеграл от 0 до l, вычисленный по формуле Симпсона [1, с. 7 - 8].

3. Описание программы

3.1 Руководство программиста

Каждая функция, производящая вычисления, имеет доступ у следующим полям, считанным из формы:

- double L – длина стержня
- double T – время наблюдения
- int hnum – количество шагов по x
- int tnum – количество шагов по времени
- double b0, b1, b2 – параметры управляющей функции
- double phi1, phi2 – параметры начального распределения температуры

и рассчитываемым программно:

- double t – размер шага по времени
- double h – размер шага по x
- double[,] res – сеточная функция, результат вычислений

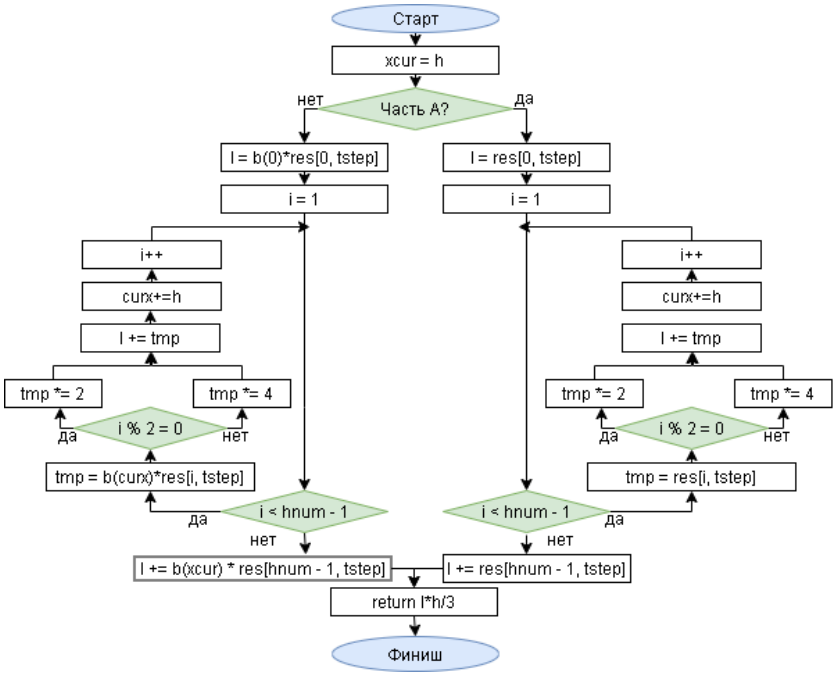
Основные функции:

double phi(double x) – рассчитывает значение начального распределения в заданной точке;

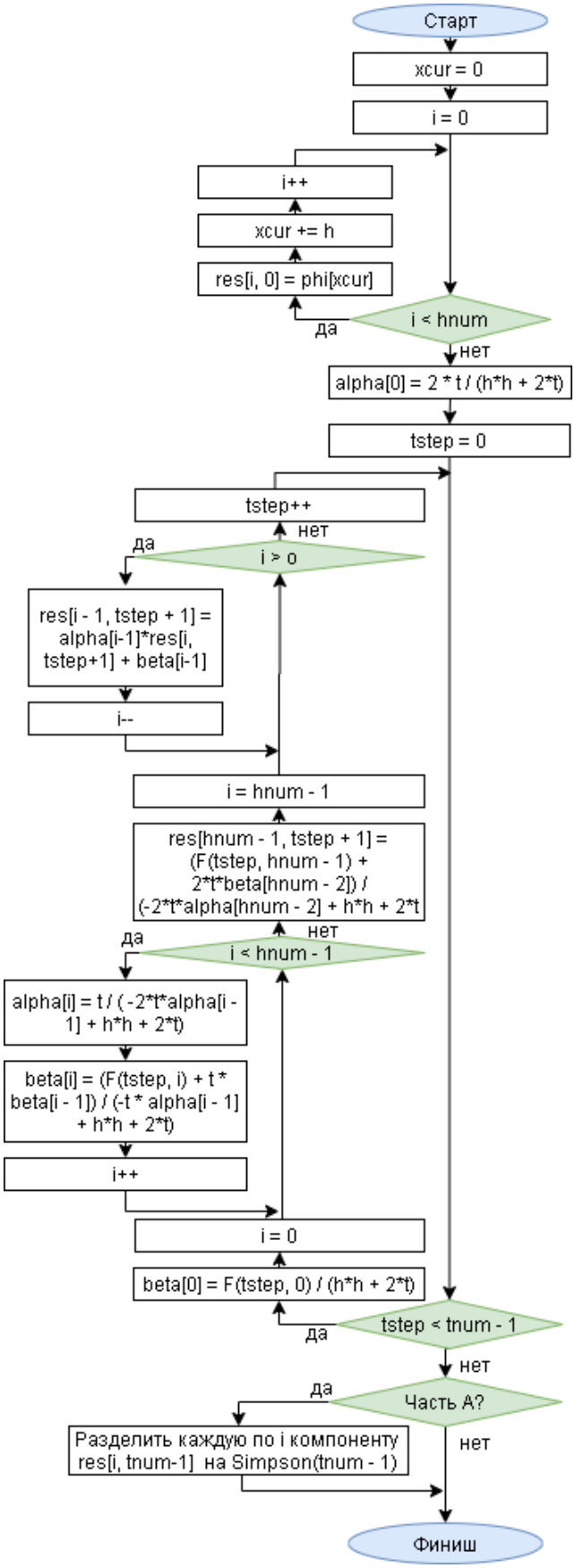
double b(double x) – рассчитывает значение управляющей функции в заданной точке;

double F(int tstep, int k) – рассчитывает значение правой части k-того уравнения на tstep слое;

double Simpson(int tstep) – рассчитывает интеграл по формуле Симпсона:



void Triagonal() – получение решения:



3.2 Руководство пользователя

После запуска программы пользователю предлагается ввести длину стержня, время изменения температуры, количество точек, в которых производятся расчеты, а также параметры начального распределения температуры и управления с обратной связью. Переход к следующему полю ввода может осуществляться с помощью мыши или посредством нажатия клавиш *Tab* и *Enter*. Внутри полей предусмотрен ввод цифр, запятой (для полей, принимающих дробные значения; точка автоматически преобразуется в запятую), а также удаление символов.

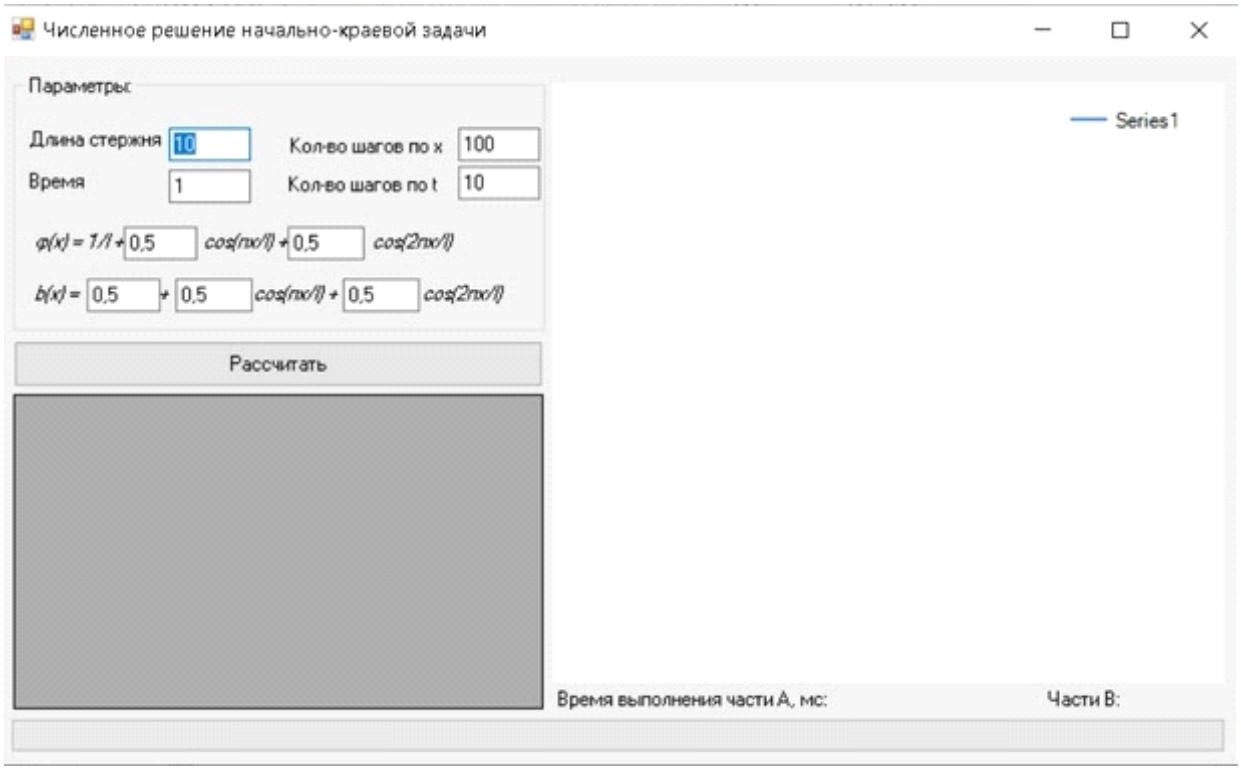


Рисунок 3. 1. Стартовый экран

Когда все параметры заданы, можно приступить к решению задачи. Для этого нужно нажать кнопку *“Рассчитать”*. Прогресс вычисления выводится в нижней части формы.

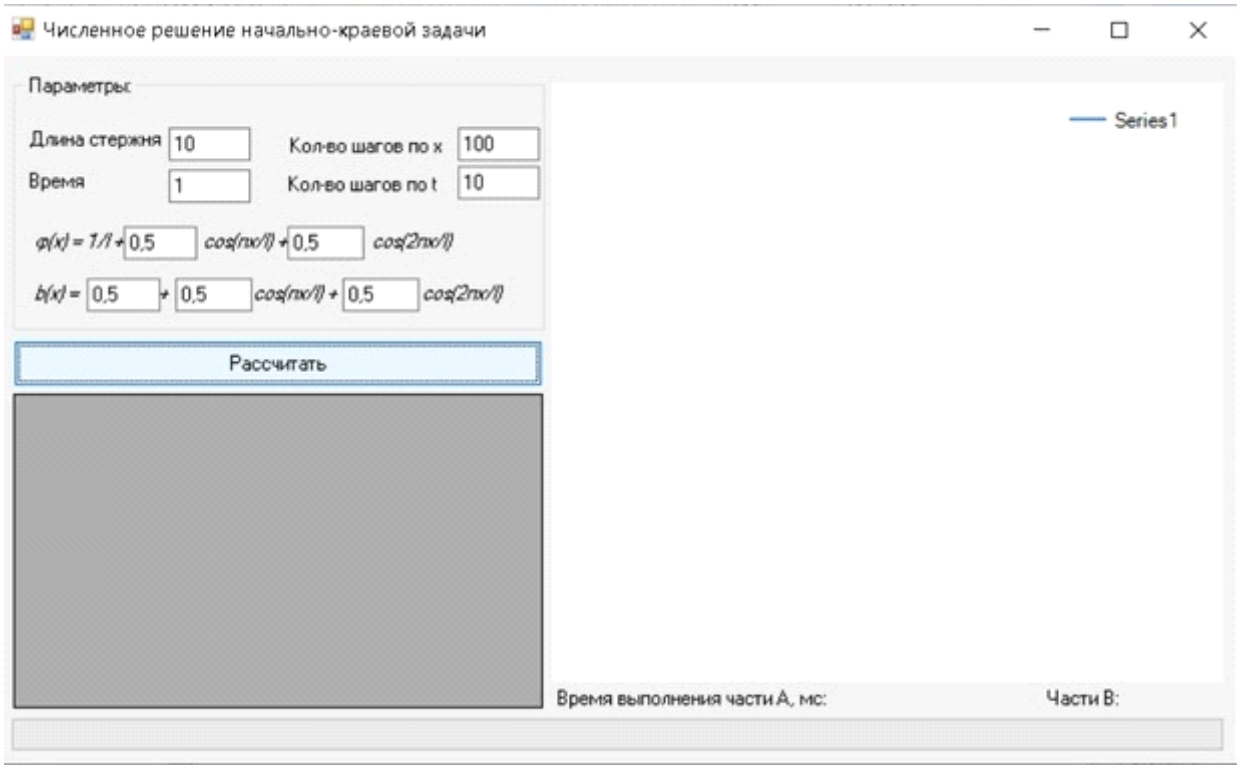


Рисунок 3. 2. Начало расчетов

В результате выполнения программы строится график начального (*“Входные данные”*) и конечного (*“Часть B”*) распределения температуры и выводится время выполнения программы в миллисекундах.

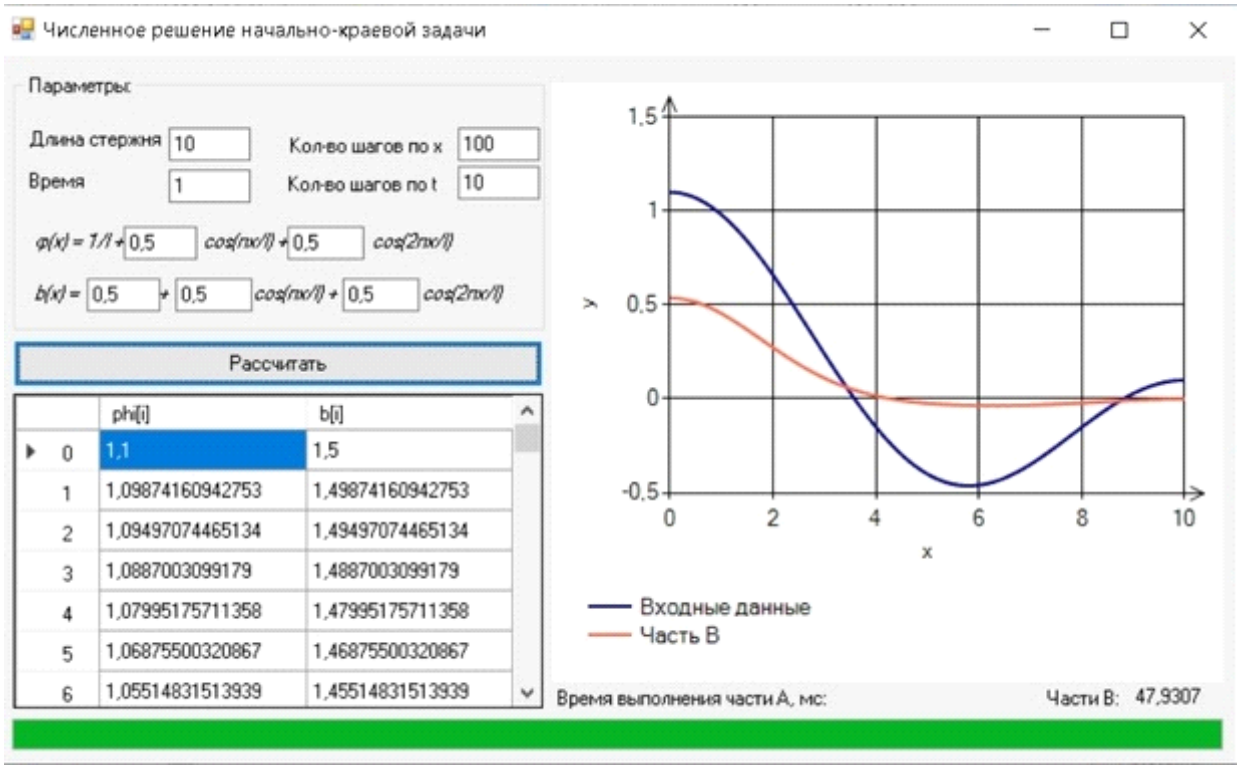


Рисунок 3. 3. Решение части B

По нажатию клавиши *Space* начинается решение с использованием части А. График решения (*“Часть А”*) совпадает с графиком *“Часть В”*, время выполнения так же выводится под графиком.

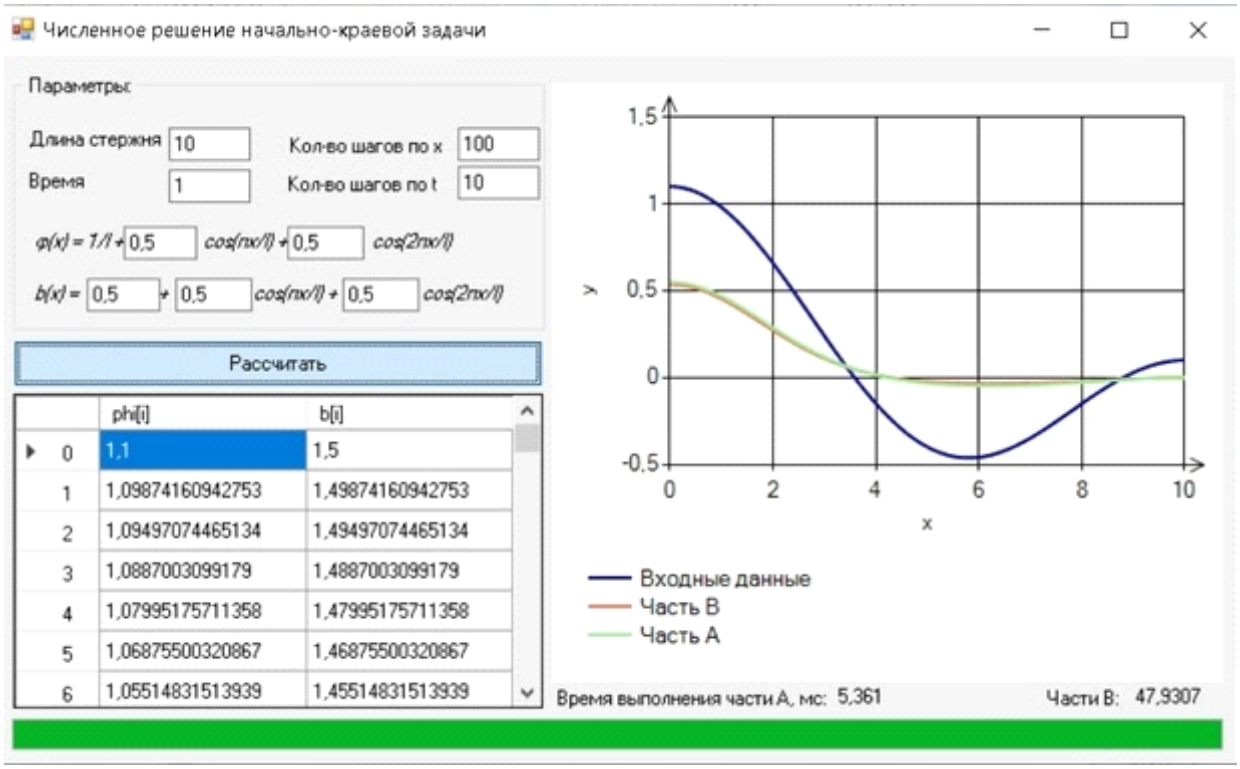


Рисунок 3. 4. Решение с использованием части А

4. Доказательство корректности

Проведем доказательство корректности согласно [2, с. 11].

Для примера возьмем следующие случаи:

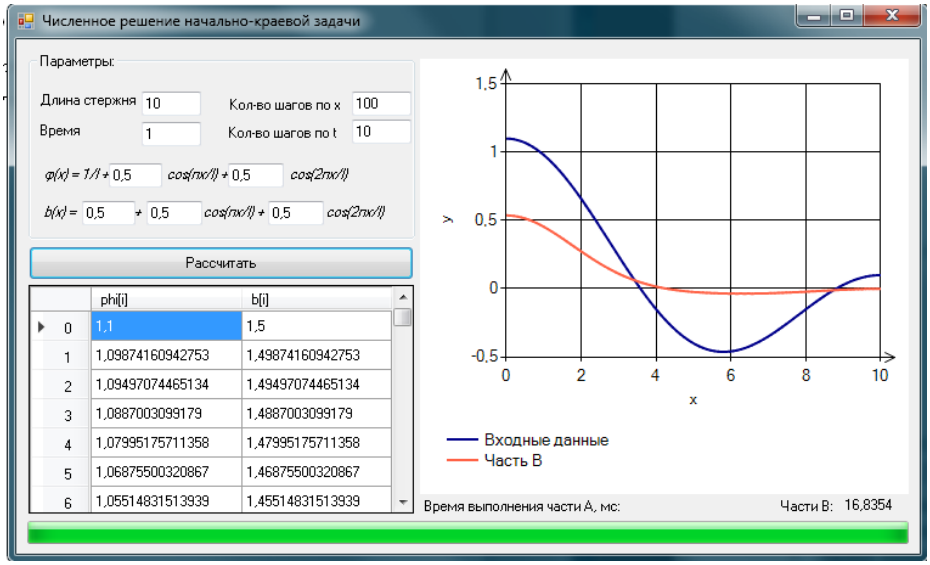


Рисунок 4. 1. Пример 1

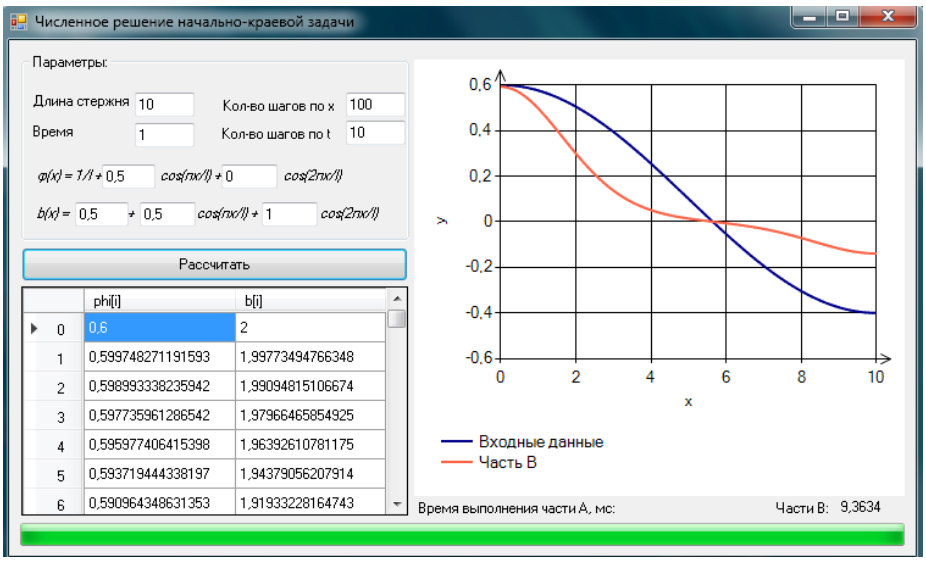


Рисунок 4. 2. Пример 2

На концах отрезка график решения имеет горизонтальные касательные (в силу краевых условий); площадь фигуры между графиками, в которой $y(x) > \varphi(x)$, равна площади фигуры, в которой $y(x) < \varphi(x)$.

Проверим, что при изменении b_0 график не изменится:

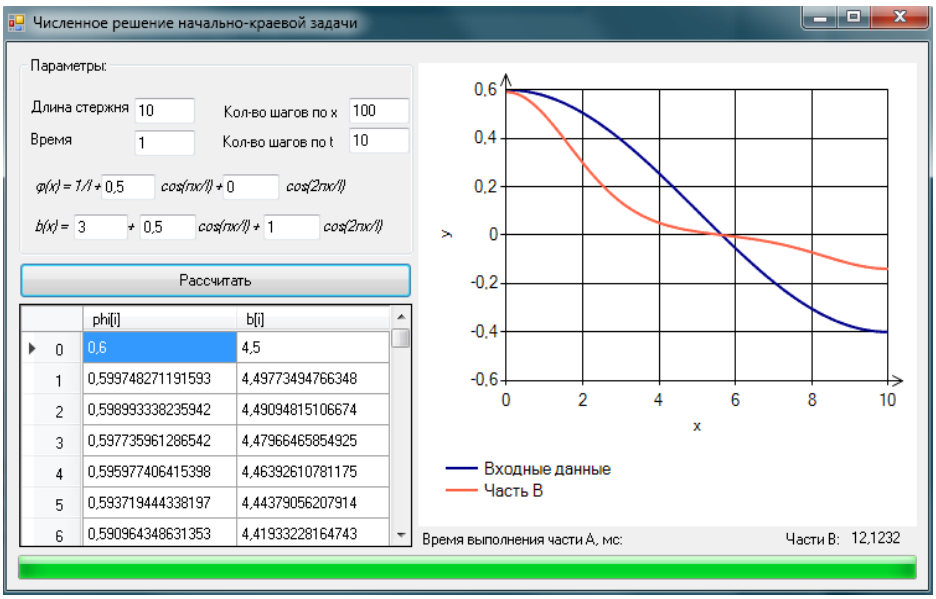


Рисунок 4. 3. Пример 1. Изменение b0

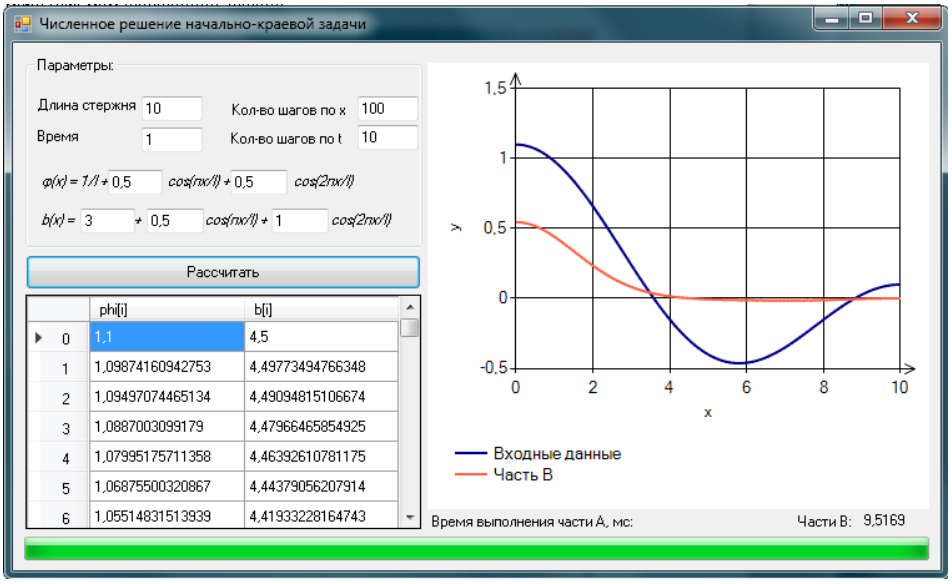


Рисунок 4. 4. Пример 2. Изменение b0

И, наконец, проверим совпадение решения части B и решения с использованием результата части A:

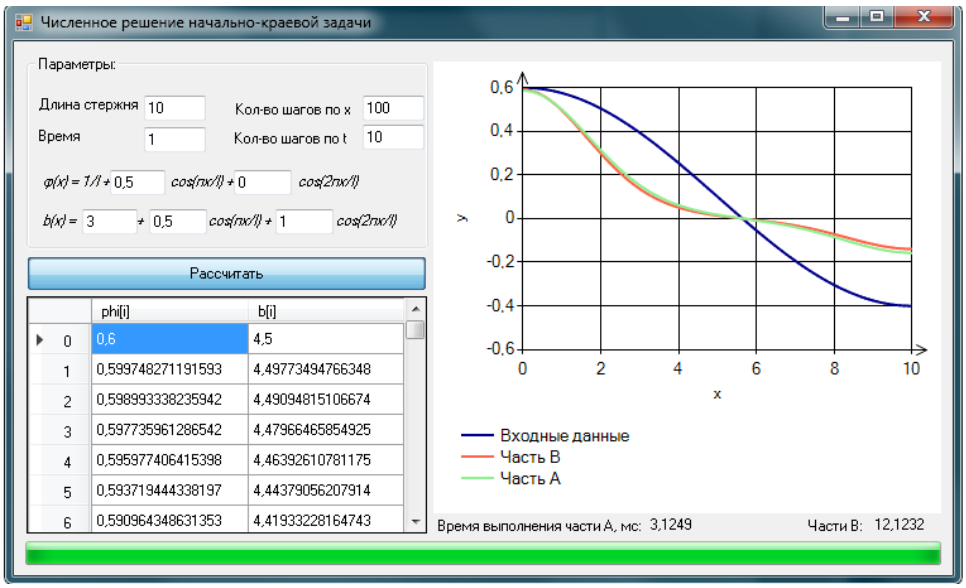


Рисунок 4. 5. Пример 1. Совпадение графиков

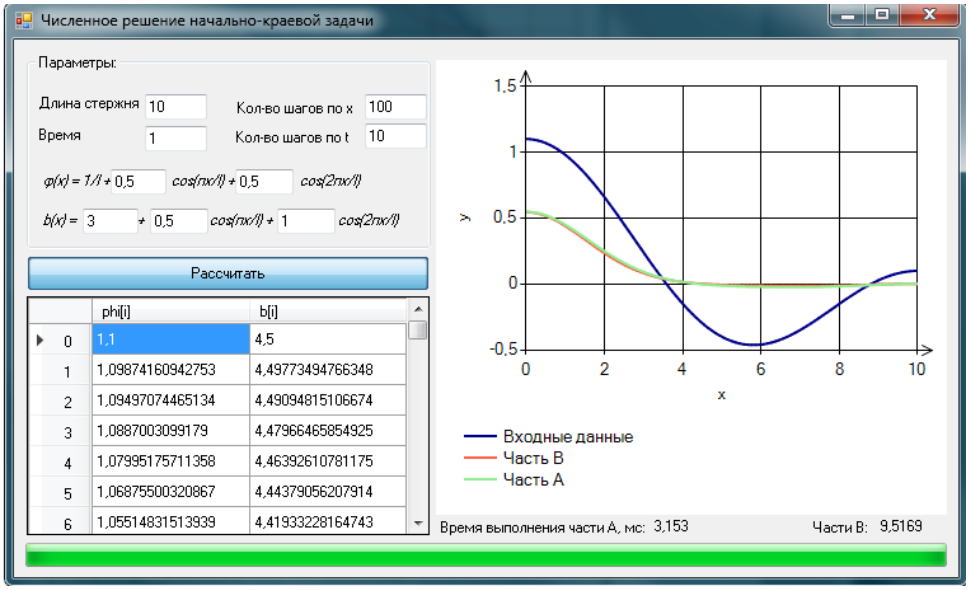


Рисунок 4. 6. Пример 2. Совпадение графиков

5. Вычислительный эксперимент

Рассмотрим время работы программы на тех же данных, что и в главе 4. Будем изменять количество шагов по времени и по пространству и замерять затраченное время в миллисекундах.

Таблица 1. Пример 1. Время работы

шагов по t	**Время выполнения, мс**			
шагов по x	*200*		*500*	
100	69	218	165	1971
500	320	3938	931	12949
1000	1888	21082	5667	21254
2000	неустойчиво		3885	42014

Таблица 2. Пример 2. Время работы

шагов по t	**Время выполнения, мс**			
шагов по x	*200*		*500*	
100	54	358	165	1975
500	268	1751	893	10369
1000	521	3581	1797	20773
2000	неустойчиво		4182	42202

Решение части А занимает гораздо меньше времени – получаем ускорение более чем в 10 раз.

6. Вывод

Разработан алгоритм численного решения начально-краевой задачи для интегро-дифференциального уравнения, описывающего управляемый процесс нагревания тонкого однородного стержня с теплоизолированными концами. Корректность полученного решения подтверждена экспериментально.

Реализована программа с дружественным интерфейсом, позволяющая задавать длину стержня, время воздействия, параметры начального распределения температур и управляющей функции с обратной связью, а также количество шагов по времени и количество точек измерения температуры.

Проведен вычислительный эксперимент, который показал десятикратное ускорение при решении задачи через линейное уравнение по сравнению с нелинейным. Так как численно решения близки, целесообразнее решать линейную задачу.

.

7. Список литературы

1. Самарский А.А.. Введение в численные методы. – СПб.: Лань, 2005. 288с.

2. 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.

Приложение

```
// Расчет начального распределения температур
double phi(double x)
{
  return (1.0 / L) + phi1 * Math.Cos((Math.PI * x) / L) + phi2 * Math.Cos((2 * Math.PI * x) / L);
}



// Расчет значения управляющей функции
double phi(double x)
{
  return part * b0 + b1 * Math.Cos((Math.PI * x) / L) + b2 * Math.Cos((2 * Math.PI * x) / L);
}




// Расчет значения правой части k-того уравнения на tstep слое
double F(int tstep, int k)
{
  if (Часть B)
    return h * h * result[i, tstep] * (t * (b(h * i) - Simpson(tstep)) + 1);
  else
    return h * h * result[i, tstep] * (t * (b(h * i) + 1);
}



// Расчет интеграла по формуле Симпсона
double Simpson(int tstep)
{
   double I;
   double tmp;
   double xcur = h;
   if (Часть B)
   {
     I = b(0) * res[0, tstep];
     for (int i = 1; i < hnum - 1; ++i)
     {
       tmp = b(xcur) * coeff * res [i, tstep];
       if (i % 2 == 0) tmp *= 2;
       else tmp* = 4;
       I += tmp;
       xcur += h;
     }
     I += b(xcur) * res[hnum - 1, tstep];
   }
   else
   {
     I = res[0, tstep];
     for (int i = 1; i < hnum - 1; i++)
     {
       tmp = res[i, tstep];
```

```
        if (i % 2 == 0) tmp *= 2;
        else tmp* = 4;
        I += tmp;
        xcur += h;
      }
      I += res[hnum - 1, tstep];
    }
    return I * h / 3;

}



// Метод прогонки
 void Triagonal()
 {
    double[] alpha = new double[hnum];
    double[] beta = new double[hnum];
    double xcur = 0;

    // Заполнение нулевого слоя
    for (int i = 0; i < hnum; k++)
    {
      res [k, 0] = phi(xcur);
      xcur += h;
    }

    // Для каждого слоя
    for (int tstep = 0; tstep < tnum - 1; tstep++)
    {
      // Вычислить прогоночные коэффициенты
      alpha[0] = - t * (-2 * t / h * h + 2 * t);
      beta[0] = F(tstep, 0) / h * h + 2 * t;
      for (int i = 1; i < hnum - 1; i++)
      {
        alpha[i] = t / (-t * alpha[k - 1] + B);
        beta[i] = (F(tstep, i) + t * beta[i - 1]) / (-t * alpha[i - 1] + h * h + 2 * t);
      }
      // Вычислить решение
      res[hnum - 1, tstep + 1] = (F(tstep, hnum – 1) + 2 * t * beta[hnum - 2]) / (-2 * t *
alpha[hnum - 2] + h * h + 2 * t);
      for (int i = hnum - 1; i > 0; i--)
      {
        res[i - 1, tstep + 1] = alpha[i - 1] * res[i, tstep + 1] + beta[i - 1];
      }
    }
```

```
  if (часть A)
  {
    // разделить каждую компоненту на интеграл
    double I = Simpson(tnum - 1);
    for (int i = 0; i < hnum; i++)
      res[i, tnum - 1] /= I;
  }
}
```