

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

**Институт информационных технологий, математики и механики**

**Кафедра: Дифференциальных уравнений,  
математического и численного анализа**

Направление подготовки: «Фундаментальная информатика и информационные  
технологии»

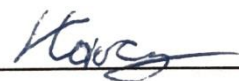
Профиль подготовки: «Инженерия программного обеспечения»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
БАКАЛАВРА**

**Тема:**

**"Разработка программных средств  
для решения краевых задач стационарного уравнения теплопроводности  
методом баланса"**

**Выполнил:** студент гр. 381606-1  
Касмазюк Никита Романович



Подпись

**Научный руководитель:**  
к.ф.-м.н., доцент  
Стронгина Наталья Романовна



Подпись

Нижний Новгород  
2020

## Оглавление

Введение.....	3
Глава I. Краевые задачи для стационарного уравнения теплопроводности.	
Методы численного решения.....	4
1.1. Основные понятия.....	4
1.2. Постановки задач, их физический смысл.....	5
1.3. Построение разностной схемы методом баланса.....	7
1.4. Метод прогонки для решения СЛАУ и его свойства.....	10
1.5. Решение разностных схем для смешанных краевых задач.....	11
1.6. Вопросы аппроксимации, устойчивости, сходимости.....	12
Глава II. Решение краевых задач с гладкими и разрывными коэффициентами	
методом баланса.....	15
2.1. Тестовая модельная задача с гладкими коэффициентами, пример для тестирования и его аналитическое решение.....	15
2.2. Разностная схема с аппроксимацией граничных условий оператором 1 порядка.....	16
2.2.1. Тестовая задача.....	16
2.2.2. Пример для тестирования.....	17
2.2.3. Обоснование аппроксимации граничных условий.....	18
2.2.4. Существование и единственность решения разностной схемы методом прогонки.....	20
2.3. Разностная схема с улучшенной аппроксимацией граничных условий.....	20
2.3.1. Тестовая модельная задача.....	20
2.3.2. Пример для тестирования.....	21
2.3.3. Обоснование аппроксимации граничных условий.....	22
2.3.4. Существование и единственность решения разностной схемы методом прогонки.....	24

2.3.5 Контрольная таблица для передачи параметров тестовой задачи.....	25
2.4. Решение основной модельной задачи с гладкими коэффициентами	
методом баланса.....	26
2.4.1. Постановка задачи.....	26
2.4.2. Схема с аппроксимацией граничных условий оператором 1 порядка.....	27
2.4.3. Схема с улучшенной аппроксимацией.....	28
2.4.4. Обоснование аппроксимации граничных условий.....	29
2.4.5. Контрольная таблица для передачи параметров основной задачи.....	33
2.5. Решение задачи с разрывными коэффициентами методом баланса.....	35
2.5.1. Постановка задачи и ее физический смысл.....	35
2.5.2. Математическая модель задачи.....	36
2.5.3. Описание сетки и разностной схемы.....	39
2.5.4 Информационная модель.....	42
2.5.5. Отладочный пример.....	42
2.5.6. Контрольная таблица для передачи параметров разрывной задачи.....	44
Глава III. Программная реализация.....	50
3.1. Основные принципы.....	50
3.2. Высокоуровневая архитектура.....	50
3.3. Использование программы.....	51
Заключение.....	59
Литература.....	60
Приложение 1. Протоколы отладки программы.....	62
Приложение 2. Решение модельных задач с заданной погрешностью (точностью).....	80
Приложение 3. Код программы.....	85

## Введение

Актуальность темы выпускной квалификационной работы обусловлена тем, что такие уравнения краевых задач актуальны при изучении свойств полупроводников и наноструктурированных материалов, когда исследуемый показатель измеряется, а температура образца в точке замера должна быть вычислена [1-4].

Целью моей выпускной квалификационной работы является создание программного средства, с помощью которого можно решать краевые задачи при разных коэффициентах и граничных условий, а так же решать задачи с нагревом наноструктуры между двумя материалами с разными свойствами.

Для реализации поставленной цели необходимо выполнить следующие задачи:

- рассмотреть различные виды краевых задач для стационарного уравнения теплопроводности;
- изучить метод для решения краевых задач;
- изучить вопросы сходимости, аппроксимации и устойчивости решения задачи и разностной схемы;
- проанализировать физический смысл краевых задач стационарного уравнения теплопроводности;
- разработать программу, которая будет решать краевые задачи с гладкими и разрывными коэффициентами с возможностью ввода параметров и граничных условий;
- проанализировать полученные практические результаты и сравнить их с теоретическими.

В ходе выпускной квалификационной работы были использованы следующие методы: интегро-интерполяционный метод (метод баланса) построения разностных консервативных схем, метод прогонки для решения системы линейных уравнений.

Практическая значимость проделанной работы определяется тем, что была поставлена и разобрана не только абстрактная задача, а реальная прикладная задача с реальными значениями. Ко всему прочему, были изучены материалы по физической составляющей поставленной прикладной задачи. Получен опыт использования таких методов, как интегро-интерполяционный метод (метод баланса) и метод прогонки. Принципиальную значимость имеет тот факт, что такие уравнения позволяют изучать теплопроводимость у новых материалов.

# Глава I. Краевые задачи для стационарного уравнения теплопроводности. Методы численного решения

## 1.1. Основные понятия

Как известно, при математической формулировке многих естественнонаучных и технических задач возникают системы линейных и нелинейных дифференциальных уравнений в частных производных, точное решение которых невозможно получить в аналитическом виде. В таком случае необходимо прибегать к тем или иным численным методам, позволяющим найти приближенное решение дифференциальной задачи. Для того чтобы построить приближенное решение, необходимо заменить исходную задачу, то есть основное уравнение и соответственные граничные условия, некоторой конечномерной задачей.

В процессе построения приближенного решения мы воспользуемся разностной схемой, для решения которой мы введем понятие сетки.

*Разностная схема* – это конечная система алгебраических уравнений, поставленная в соответствие какой-либо дифференциальной задаче, содержащей дифференциальное уравнение и дополнительные условия (например, краевые условия и/или начальное распределение). Таким образом, разностные схемы применяются для сведения дифференциальной задачи, имеющей континуальный характер, к конечной системе уравнений, численное решение которых принципиально возможно на вычислительных машинах.

Решение разностной схемы называется приближенным решением дифференциальной задачи. Есть разные способы для построения разностных схем: метод баланса, метод неопределенных коэффициентов, метод конечных разностей и т.д. Так, например, метод баланса используется обычно для решения уравнений с разрывными коэффициентами, а метод конечных разностей удобно использовать для уравнений с гладкими коэффициентами.

Пусть задана исходная дифференциальная задача, где есть область  $m$ -мерного пространства; заданная функция; линейный дифференциальный оператор. Предполагается, что дополнительные условия (типа начальных и граничных) учтены оператором и правой частью. В общем случае уравнение может быть многомерным. Существенным является требование линейности оператора. Для построения разностной схемы, прежде всего, вводится сетка.

*Сеткой*, вводимой на области, называется конечное множество точек, принадлежащих области, плотность распределения которых характеризуется параметром – шагом сетки. В общем случае – вектор, причем определена величина длины вектора. Обычно сетка выбирается так, что множество стремится заполнить всю область. *Функция, определенная в точках сетки* называется сеточной функцией. После введения сетки дифференциальный оператор в уравнении следует заменить разностным оператором, правую часть – сеточной функцией. В результате получим систему разностных уравнений. Эта система называется разностной схемой или разностной задачей. Пусть решение задачи принадлежит линейному нормированному пространству. Сеточные функции являются элементами линейного нормированного пространства (пространство сеточных функций) с нормой. По существу, имеем семейство линейных нормированных пространств, зависящих от параметра.

Системы линейных уравнений можно решать разными методами. Существуют два вида методов: Первые методы – прямые. Это методы, которые решают задачу за конечное число арифметических действий. Вторые методы – итерационные. Они генерируют последовательность, каждый элемент которой может рассматриваться как приближение. К прямым методам относятся, например, метод Крамера, метод прогонки. Заметим, что метод прогонки используют для решения СЛАУ в виде трехдиагональной матрицы.

С методологической точки зрения всегда полезно помнить, что своевременное исследование любой естественнонаучной или технической проблемы в рамках вычислительного эксперимента сводится к реализации триады модель – алгоритм – программа. Прежде всего, основные законы, управляющие рассматриваемым процессом, формулируются в виде математической модели, т.е. в виде системы дифференциальных уравнений. Затем создается вычислительный алгоритм, предназначенный для приближенного численного решения упомянутой системы с помощью той или иной вычислительной техники. Далее создается реализация численного метода в программе для компьютера [1]. Таким образом, цель нашей работы – изучить основные законы и сформулировать их в виде математической модели, разобраться в вычислительном алгоритме для данной модели, подобрать наиболее оптимальный метод решения и реализовать полученный численный метод в программе.

## 1.2. Постановки задач, их физический смысл

Рассмотрим задачу

$$\begin{cases} (k(x)U'(x))' - q(x)U(x) = -f(x), \text{ при } x \in [0,1] \\ -k(0)U'(0) = -\mu_1 * (U(0) - \Theta_1) \\ -k(1)U'(1) = \mu_2 * (U(1) - \Theta_2) \end{cases} \quad (1.1)$$

где  $k(x)$ ,  $q(x)$ ,  $f(x)$ ,  $\mu_1$ ,  $\mu_2$ ,  $\Theta_1$ ,  $\Theta_2$  заданы, причем  $k(x) > 0$ ,  $q(x) \geq 0$ ,  $x \in [0,1]$ ,  $\mu_1 > 0$ ,  $\mu_2 > 0$ .

Это третья краевая задача для стационарного уравнения теплопроводности, заданного на отрезке. Значение  $U(x)$  показывает температуру стержня в точке  $x$ . В общем случае именно значение  $U(x)$  в каждой точке  $x \in [0,1]$  необходимо вычислить.

Функция  $U(x)$  описывает стационарное распределение температур на тонком однородном в своем поперечном сечении стержне единичной длины. Левый конец стержня соответствует точке  $x = 0$ , правый – точке  $x = 1$ . В поставленной задаче  $\Theta_1$  – число, показывающее температуру окружающей среды на левом конце стержня,  $\mu_1$  – коэффициент передачи тепла от стержня в окружающую среду на левом конце. Аналогично,  $\Theta_2$  – число, показывающее температуру окружающей среды на правом конце стержня,  $\mu_2$  – коэффициент передачи тепла от стержня в окружающую среду справа. Коэффициент  $k(x)$  – коэффициент теплопроводности стержня в точке  $x$ .

Функцию  $\omega(x) = -k(x) * U'(x)$  называют функцией теплового потока. Значение  $\omega(x)$  показывает тепловой поток через поперечное сечение стержня в точке  $x \in [0,1]$ . Заметим, что в рассмотренной задаче каждое граничное условие описывает тепловой поток через торец стержня и теплообмен стержня с окружающей средой по закону Ньютона.

Иногда граничные условия задачи (1.1) будем записывать так:

$$-kU'(0) + \mu_1 * U(0) = \mu_1 * \Theta_1$$

$$-kU'(1) - \mu_2 * U(1) = \mu_2 * \Theta_2$$

Одновременно с задачей (1.1) будем рассматривать задачу (1.2)

$$\begin{cases} (k(x)U'(x))' - q(x)U(x) = -f(x), \text{ при } x \in [0,1] \\ U(0) = \mu_1 \\ U(1) = \mu_2 \end{cases} \quad (1.2)$$

Это первая краевая задача для стационарного уравнения теплопроводности, заданного на отрезке. В данной задаче граничные условия описывают температуру на концах стержня.

Если в уравнении (1.1) одно из граничных условий заменить граничным условием задачи (1.2), то такая задача будет называться смешанной краевой задачей.

Уравнения (1.1) и (1.2) получены на основе закона сохранения тепла, записанного в интегральной форме [1]. Уравнения (1.1) и (1.2) и смешанные краевые задачи могут описывать температуру образцов, сделанных из новых материалов, и используются при изучении свойств этих материалов [6].

### 1.3. Построение разностной схемы методом баланса

Без ограничения общности считаем, что задачи (1.1), (1.2) и смешанные краевые задачи поставлены корректно и имеют единственное решение [2]. Чтобы решать такие задачи численно, используем метод баланса и построим разностные схемы. Рассмотрим построение схем на примере первой краевой задачи (1.2). Для построения возьмем равномерную сетку с шагом  $h = \frac{1}{n}$ , узлами основной сетки  $x_i = ih$ ,  $i=0, n$ , узлами вспомогательной сетки  $x_{i+1/2} = (i + 0,5)h$ ,  $i=0, n-1$ .

Проинтегрируем дифференциальное уравнение задачи (1.2) на каждом из участков от  $x_{i-1/2}$  до  $x_{i+1/2}$ , где  $i=0, n-1$

Тогда

$$\omega_{i+1/2} - \omega_{i-1/2} - \int_{x_{i-0,5}}^{x_{i+0,5}} q(x)U(x)dx + \int_{x_{i-0,5}}^{x_{i+0,5}} f(x)dx = 0 \quad (1.3)$$

Заменим  $\int_{x_{i-0,5}}^{x_{i+0,5}} q(x)U(x)dx$  другим значением, а именно  $U(x_i) * \int_{x_{i-0,5}}^{x_{i+0,5}} q(x)dx$  (значение  $U(x)$  берется в средней точке).

Введем обозначения для индексов  $i=1, n-1$ .

$$\phi_i = \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} f(x)dx \quad (1.4)$$

$$d_i = \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} q(x)dx \quad (1.5)$$

Разделим уравнение (1.3) на  $h$ , воспользуемся обозначениями (1.4), (1.5) и получим новую систему уравнений

$$\frac{\omega_{i+\frac{1}{2}} - \omega_{i-\frac{1}{2}}}{h} - d_i U_i + \phi_i = 0 \quad (1.6)$$

где  $i=1, n-1$ .

Для построения схемы методом баланса используется следующее свойство:



$$U_i - U_{i-1} \cong \omega_{i-1/2} \int_{x_{i-1}}^{x_i} \frac{1}{k(x)} * dx$$

Введем новые обозначения для индекса  $i=1, n$ .

$$a_i = \left( \frac{1}{h} * \int_{x_{i-1}}^{x_i} \frac{dx}{k(x)} \right)^{-1} \quad (1.7)$$

Получим, что

$$\omega_{i-1/2} \approx a_i \frac{U_i - U_{i-1}}{h} = a_i U_{\dot{x},i} \quad (1.8)$$

$$\omega_{i+1/2} \approx a_{i+1} U_{x,i} \quad (1.9)$$

Подставляя формулы (1.8) и (1.9) в уравнение (1.6), получим разностное уравнение, содержащее значения искомой функции в точках  $x_i, x_{i\pm 1}$ :

$$\frac{1}{h} (a_{i+1} U_{x,i} - a_i U_{\dot{x},i}) - d_i U_i + \phi_i = 0$$

Или в сокращенной записи

$$(aU_{\dot{x}})_{x,i} - d_i U_i + \phi_i = 0 \quad (1.10)$$

Записывая уравнение (1.10) во всех точках сетки, где оно определено, то есть при  $i=1, n-1$ , мы получаем систему из  $n-1$  линейных уравнений, но с  $n+1$  неизвестных. Еще два уравнения получаем из аппроксимации граничных условий:

$$U(0) = \mu_1$$

$$U(1) = \mu_2$$

Используя новые обозначения ( $V_i = U_i$ ) и объединяя уравнения, получим разностную схему в виде СЛАУ для решения первой краевой задачи (1.2)

$$(aV_{\dot{x}})_{x,i} - d_i V_i + \phi_i = 0$$

$$V_0 = \mu_1 \quad (1.11)$$

$$V_1 = \mu_2.$$

где  $i = 1, n-1$ .

Для третьей краевой задачи (1.1) и смешанных краевых задач есть два подхода к аппроксимации граничных условий. Один подход использует дифференциальный оператор 1 порядка, другой – использует метод баланса. Такой аппроксимация называется улучшенной и имеет второй порядок. Эти подходы будут рассмотрены во главах 2 и 3 на примере модельных задач.

Разностная схема в виде СЛАУ для третьей краевой задачи (1.1) полученная с помощью аппроксимации граничных условий оператором 1 порядка имеет вид

$$\begin{aligned}
 (aV_{\bar{x}})_{x,i} - d_i V_i + \phi_i &= 0 \\
 -\left(\frac{k}{k+\gamma_1 h}\right) V_1 + V_0 &= \frac{\theta_1 \gamma_1 h}{k+\gamma_1 h} \\
 -\left(\frac{k}{k+\gamma_2 h}\right) V_{n-1} + V_n &= \frac{\theta_2 \gamma_2 h}{k+\gamma_2 h}
 \end{aligned} \tag{1.12}$$

где  $i = 1, n-1$ .

Разностная схема в виде СЛАУ для третьей задачи (1.1) полученная с помощью улучшенной аппроксимации граничных условий имеет вид

$$\begin{aligned}
 (aV_{\bar{x}})_{x,i} - d_i V_i + \phi_i &= 0 \\
 -\left(\frac{k}{k+\gamma_1 h + \frac{3}{2}h^2}\right) V_1 + V_0 &= \frac{\theta_1 \gamma_1 h + 6h^2}{k+\gamma_1 h + \frac{3}{2}h^2} \\
 -\left(\frac{k}{k+\gamma_2 h + \frac{3}{2}h^2}\right) V_{n-1} + V_n &= \frac{\theta_2 \gamma_2 h + 6h^2}{k+\gamma_2 h + \frac{3}{2}h^2}
 \end{aligned} \tag{1.13}$$

где  $i = 1, n-1$ .

В схемах (1.11), (1.12) и (1.13) коэффициенты могут быть найдены точно по формулам (1.5), (1.6), (1.10) либо вычислены приближенно по формуле трапеций на участках гладкости подынтегральных функций. Разностные схемы (1.11), (1.12) и (1.13), как системы линейных алгебраических уравнений представлены на рисунках (1.1) (1.2) (1.3).

$$\begin{bmatrix} \frac{1}{h^2} & -\frac{1}{h^2} + d_1 & \frac{a_2}{h^2} & \cdots & 0 & 0 & 0 \\ \frac{a_1}{h^2} & -\frac{a_1+a_2}{h^2} + d_1 & \frac{a_2}{h^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1}+a_n}{h^2} + d_{n-1} & \frac{a_n}{h^2} \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \mu_1 \\ -\phi_1 \\ \vdots \\ -\phi_{n-1} \\ \mu_2 \end{bmatrix}$$

Рис. 1.1 СЛАУ для разностной схемы (1.11) в виде трехдиагональной матрицы

$$\begin{bmatrix} \frac{1}{h^2} & -\frac{k}{k+\gamma_1 h} & \frac{a_2}{h^2} & \cdots & 0 & 0 & 0 \\ \frac{a_1}{h^2} & -\frac{a_1+a_2}{h^2} + d_1 & \frac{a_2}{h^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1}+a_n}{h^2} + d_{n-1} & \frac{a_n}{h^2} \\ 0 & 0 & 0 & \cdots & 0 & -\frac{k}{k+\gamma_2 h} & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \frac{\theta_1 \gamma_1 h}{k+\gamma_1 h} \\ -\phi_1 \\ \vdots \\ -\phi_{n-1} \\ \frac{\theta_2 \gamma_2 h}{k+\gamma_2 h} \end{bmatrix}$$

Рис. 1.2 СЛАУ для разностной схемы (1.12) в виде трехдиагональной матрицы

$$\begin{bmatrix} 1 & -\frac{a_1}{a_1+\gamma_1 h+\frac{h^2}{2}d_0} & 0 & \dots & 0 & 0 & 0 \\ \frac{a_1}{h^2} & -\frac{a_1+a_2}{h^2}+d_1 & \frac{a_2}{h^2} & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & & \\ 0 & 0 & 0 & \dots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1}+a_n}{h^2}+d_{n-1} & \frac{a_n}{h^2} \\ 0 & 0 & 0 & & 0 & -\frac{a_n}{a_n+\gamma_2 h+\frac{h^2}{2}d_2} & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \frac{\phi_0 \frac{h^2}{2} + \theta_1 \gamma_1 h}{a_1+\gamma_1 h+\frac{h^2}{2}d_0} \\ -\phi_1 \\ \vdots \\ -\phi_{n-1} \\ \frac{\phi_n \frac{h^2}{2} + \theta_2 \gamma_2 h}{a_n+\gamma_2 h+\frac{h^2}{2}d_2} \end{bmatrix}$$

Рис 1.3 СЛАУ для разностной схемы (1.13) в виде трехдиагональной матрицы

## 1.4. Метод прогонки для решения СЛАУ и его свойства

Для решения разностной схемы будем использовать метод прогонки. Метод прогонки представляет собой прямой метод, предназначенный для решения СЛАУ с трехдиагональной матрицей. Такие СЛАУ можно записать в виде:

$$\begin{bmatrix} 1 & -\gamma_1 & 0 & \dots & 0 & 0 & 0 \\ A_1 & -C_1 & B_1 & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & A_{n-1} & -C_{n-1} & B_{n-1} \\ 0 & 0 & 0 & & 0 & -\gamma_2 & 1 \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{n-1} \\ U_n \end{bmatrix} = \begin{bmatrix} \mu_1 \\ -\phi_1 \\ \vdots \\ -\phi_{n-1} \\ \mu_2 \end{bmatrix}$$

Метод прогонки работает в два этапа [1]. Первый этап метода называется прямым ходом, на этом этапе будут подсчитаны коэффициенты  $\alpha_i$  и  $\beta_i$ , где  $i = 1, n$ :

$$\alpha_1 = \gamma_1, \beta_1 = \mu_1,$$

$$\alpha_{i+1} = \frac{\beta_i}{C_i - A_i \alpha_i},$$

$$\beta_{i+1} = \frac{\phi_i + A_i \beta_i}{C_i - A_i \alpha_i},$$

где  $i=1, n-1$ . Коэффициенты вычисляются последовательно, начиная с  $i = 1$  и заканчивая  $i = n - 1$ .

Второй этап называется обратным ходом прогонки. На этом этапе вычисляется искомый вектор, начиная с последнего компонента

$$V_n = \frac{-\chi_2 * \beta_n - \mu_2}{1 - \alpha_n * \chi_2}$$

и затем последовательно, по убыванию индекса, заканчивая компонентом с индексом 0:

$$V_i = \alpha_{i+1} V_{i+1} + \beta_{i+1}, \text{ где } i = n-1, 0.$$

Рассмотрим вопросы о существовании единственности решения, вычислительной устойчивости и трудоемкости метода прогонки.

Ответ на вопрос о существовании единственности решения СЛАУ дают следующие теоремы:

**Теорема 1.** Пусть в системе (1) все  $A_i \neq 0$ ,  $B_i \neq 0$ ,  $|C_i| > |A_i| + |B_i|$ , где  $i = 1, n-1$ .  $|\chi_1| \leq 1$ ,  $|\chi_2| \leq 1$ . Тогда система (1) при любой правой части, то есть любых  $\mu_1, \mu_2, \phi_i$  имеет единственное решение и его можно отыскать с помощью метода прогонки.

**Теорема 2.** Пусть в системе (1) все  $A_i \neq 0$ ,  $B_i \neq 0$ ,  $|C_i| \geq |A_i| + |B_i|$ , где  $i = 1, n-1$ .  $|\chi_1| \leq 1$ ,  $|\chi_2| < 1$ . Тогда система (1) при любой правой части, то есть любых  $\mu_1, \mu_2, \phi_i$  имеет единственное решение и его можно отыскать с помощью метода прогонки.

Эти теоремы называют теоремами о применимости прогонки.

**Определение 1.** Метод называется вычислительно устойчивым, если вычислительная погрешность, возникшая на некотором шаге, больше не возрастает.

**Теорема 3.** При выполнении условий Теоремы 1 или Теоремы 2 метод прогонки является вычислительно устойчивым.

**Утверждение 1.** Для системы (1), имеющей размерность  $(n+1) \times (n+1)$ , для того, чтобы решение было найдено, необходимо выполнить  $8 \cdot n - 1$  арифметических действий.

Таким образом, этот метод эффективнее, чем метод Гаусса, поскольку для системы (1), имеющей размерность  $(n+1) \times (n+1)$ , для того, чтобы решение было найдено методом Гаусса, необходимо выполнить порядка  $(n+1)^3$  арифметических действий.

## 1.5. Решений разностных схем для смешанных краевых задач

На основе Теоремы 1, Теоремы 2, Теоремы 3 о применимости прогонки и ее вычислительной устойчивости и на основе формул коэффициентов (1.4), (1.5), (1.7) решается вопрос о существовании единственности решения разностных схем и возможности отыскания этих решений методом прогонки.

**Теорема 4.** При любом выборе числа разбиений  $n \geq 2$  решение разностной схемы (1.9) для первой краевой задачи (1.2) с коэффициентами (1.4) (1.5) (1.7) существует единственно и может быть найдено методом прогонки. Методом прогонки в этом случае является вычислительно устойчивым.

**Теорема 5.** При любом выборе числа разбиений  $n \geq 2$  решение разностной схемы (1.10) для первой краевой задачи (1.1) с коэффициентами (1.4) (1.5) (1.7) существует единственно и может быть найдено методом прогонки. Методом прогонки в этом случае является вычислительно устойчивым.

**Теорема 6.** При любом выборе числа разбиений  $n \geq 2$  решение разностной схемы (1.11) для первой краевой задачи (1.1) с коэффициентами (1.4) (1.5) (1.7) существует единственно и может быть найдено методом прогонки. Методом прогонки в этом случае является вычислительно устойчивым.

## 1.6. Вопросы аппроксимации, устойчивости, сходимости

**Определение 2.** *Погрешностью решения задачи (1.1) с помощью некоторой разностной схемы называют разность их точных решений в узлах сетки:*

$$z_i = U_i - V_i, \text{ где } i = 0, n.$$

Здесь  $U$  - точное решение задачи (1.1),  $V$  - точное решение схемы.

*Погрешность* это вектор  $z$ , для измерения этого вектора используют норму:  $\|z\|$ . Если при сгущении сетки погрешность стремится к 0,

$$\lim_{n \rightarrow +\infty} \|z\| = 0.$$

то говорят, что решение схемы сходится к решению задачи, или кратко - схема сходится.

**Определение 3.** Если на достаточно густых сетках для погрешности  $z$  верна оценка  $\|z\| \leq Mh^k$ , где  $h > 0$  - шаг сетки и  $k > 0$ ,  $M > 0$  - константы, не зависящие от  $h$ , говорят, что схема имеет порядок сходимости  $k$  и решение схемы сходится к решению задачи с порядком  $k$ .

Так как точные решения дифференциальных уравнений, как правило, неизвестны, и поэтому погрешность  $z$  неизвестна, для изучения сходимости разностных схем используем невязки. Для уравнений произвольного вида *невязкой* называют разность левой и правой частей уравнения.

**Определение 4.** *Погрешностью аппроксимации задачи разностной схемой называют невязку разностной схемы, при условии, что в нее подставили точное решение дифференциальной задачи. Погрешность аппроксимации есть вектор  $\Psi = (\Psi_1, \Psi_2, \dots, \Psi_n) \in R^{n+1}$ , у которого столько же компонент, сколько уравнений содержит разностная схема.*

**Определение 5.** Если на достаточно густых сетках верна оценка  $\|\Psi\| \leq mh^k$ , где  $h > 0$  - шаг сетки и  $k > 0$ ,  $m > 0$  - константы, не зависящие от  $h$ , то говорят, что разностная схема аппроксимирует дифференциальную задачу с порядком  $k$ .

**Определение 6.** Если на достаточно густых сетках верна оценка  $\|z\| \leq C\|\Psi\|$ , где  $C > 0$  - константа, не зависящая от  $h$ , то говорят, что разностная схема устойчива.

**Теорема 7.** Если одновременно выполняются неравенства 7 и 8 и в них использована одна и та же норма погрешности аппроксимации, то

$$\|z\| \leq C\|\Psi\| \leq Cmh^k = Mh^k \quad (1.14)$$

где  $h > 0$  - шаг сетки и  $k > 0$ ,  $M > 0$  - константы, не зависящие от  $h$ . Неравенство (1.14) означает, что схема имеет порядок сходимости  $k$  и *решение схемы сходится* к решению задачи с порядком  $k$ .

Таким образом, если разностная схема устойчива, аппроксимация с порядком  $k$  влечет сходимость с тем же порядком.

Рассмотрим погрешность на примере первой краевой задачи (1.2) и разностной схеме (1.11)

$$\begin{aligned} a_{i+1} * \frac{V_{i+1} - V_i}{h^2} - a_i * \frac{V_i - V_{i-1}}{h^2} - d_i V_i + f_i &= \Psi_i, i=1, n-1 \\ V_0 - \mu_1 &= \Psi_0 \\ V_n - \mu_2 &= \Psi_n \end{aligned} \quad (1.15)$$

Компоненты погрешности  $z$  и компоненты погрешности аппроксимации  $\psi$  связаны системой уравнений.

$$\begin{aligned} a_{i+1} * \frac{z_{i+1} - z_i}{h^2} - a_i * \frac{z_i - z_{i-1}}{h^2} - d_i z_i + f_i &= \Psi_i, i=1, n-1 \\ z_0 - \mu_1 &= \Psi_0 \\ z_n - \mu_2 &= \Psi_n \end{aligned} \quad (1.16)$$

Система уравнений (1.16) имеет такую же матрицу, как и разностная схема первой краевой задачи. Но при этом в левой части уравнения (1.16) вместо решения  $V$  записывается погрешность  $z$ , а в правой части системы уравнений (1.16) - записана погрешность  $\Psi$ .

**Теорема 8.** Разностная схема (1.11) устойчива:  $\max |z_i| \leq \max |\Psi_i|$ , где максимум берется по  $i = 0, n$ . Доказательство в учебной литературе.

**Теорема 9.** Разностная схема имеет  $k$ -й порядок точности (или сходится с порядком  $k$ ), если  $\|V_h - U\| \leq mh^k$ , где  $h > 0$ ,  $m > 0$  – константы, не зависящие от  $h$ ,  $V_h$  – решение разностной задачи (1.11),  $U$  – решение исходной задачи (1.2).

**Определение 7.** Разностная схема (1.11) называется *корректной*, если:

- 1) Ее решение существует и единственно при любых правых частях
- 2) Существует постоянная  $M_2 > 0$ , не зависящая от  $h$  и такая, что при любых  $\phi_h \in \beta_h$  справедлива оценка  $\|V_h\|_h \leq M_2 \|\phi_h\|_h$ .

**Теорема 10.** Пусть дифференциальная задача (1.2) поставлена корректно, разностная схема (1.11) является корректной и аппроксимирует исходную задачу (1.2). Тогда решение разностной задачи (1.11) сходится к решению исходной задачи (1.2), причем порядок точности совпадает с порядком аппроксимации.

## Глава II. Решение краевых задач с гладкими и разрывными коэффициентами методом баланса

### 2.1. Тестовая модельная задача с гладкими коэффициентами, пример для тестирования и его аналитическое решение

Рассмотрим тестовую модельную задачу

$$\begin{cases} kU''(x) - qU(x) = -f \text{ при } x \in [0,1] \\ -kU'(0) = -\gamma_1 * (U(0) - \Theta_1) \\ -kU'(1) = \gamma_2 * (U(1) - \Theta_2) \end{cases} \quad (2.1.1)$$

где  $k > 0$  - положительное число,  $q \geq 0$  - неотрицательное число,  $f$  - любое вещественное число, числа  $\gamma_1 > 0, \gamma_2 > 0$ .

Для тестирования будем использовать следующий пример

$$\begin{cases} U'(x) - 3U(x) = -12 \text{ при } x \in [0,1]. \\ -U'(0) = -10(U(0) - 25) \\ -U'(1) = 10(U(1) - 25) \end{cases} \quad (2.1.2)$$

Найдем аналитическое решение. Точное решение задачи (2.1.2) ищем в виде:

$$U(x) = A \cosh(\alpha x) + B \sinh(\beta x) + C$$

Получим  $\beta = \pm \sqrt[2]{3}$  и  $\alpha = \pm \sqrt[2]{3}$ ,  $C = 4$ ,  $A, B$  – как решение СЛАУ.

$$\begin{bmatrix} 10 & -\sqrt[2]{3} \\ -\sqrt[2]{3} \operatorname{sh}(\sqrt[2]{3}) - 10 \operatorname{ch}(\sqrt[2]{3}) & -\sqrt[2]{3} \operatorname{ch}(\sqrt[2]{3}) - 10 \operatorname{sh}(\sqrt[2]{3}) \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 210 \\ -210 \end{bmatrix}$$

Таким образом,

$$U(x) = 18,7311 \operatorname{ch}(\sqrt[2]{3}x) - 13,0996 \operatorname{sh}(\sqrt[2]{3}x) + 4 \quad (2.1.3)$$

Формула (2.1.3) записана с погрешностью, потому что числовые коэффициенты перед гиперболическими функциями записаны с погрешностью  $0.5 \cdot 10^{-4}$ . Решение задачи (2.1.2) на графике (рис. 2.1).



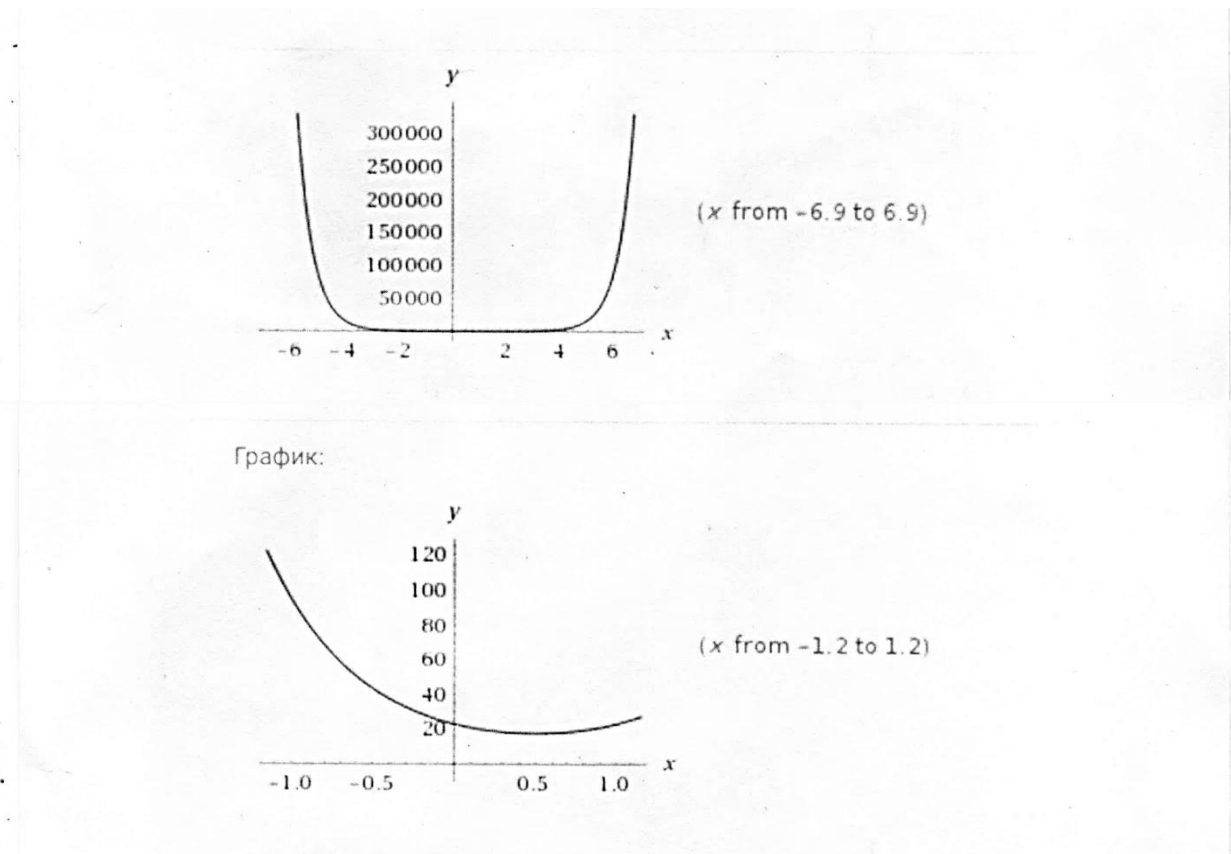


Рис. 2.1 Функция (2.3)

## 2.2. Разностная схема с аппроксимацией граничных условий оператором 1 порядка

### 2.2.1. Тестовая модельная задача

Рассмотрим тестовую модельную задачу (2.1.1). Напомним, что она представляет собой стационарное уравнение теплопроводности с краевыми условиями третьего рода:

$$\begin{cases} kU''(x) - qU(x) = -f \text{ при } x \in [0,1] \\ -kU'(0) = -\gamma_1 * (U(0) - \Theta_1) \\ -kU'(1) = \gamma_2 * (U(1) - \Theta_2) \end{cases}$$

где  $k > 0$  - положительное число,  $q \geq 0$  - неотрицательное число,  $f$  -любое вещественное число, числа  $\gamma_1 > 0, \gamma_2 > 0$ .

Разностная схема, полученная методом баланса, с аппроксимацией граничных условий оператором 1 порядка, на сетке с числом разбиений  $n$  имеет вид

$$\begin{aligned}
a_{i+1} * \frac{V_{i+1} - V_i}{h^2} - a_i * \frac{V_i - V_{i-1}}{h^2} - d_i V_i + f_i &= 0, i=1, n-1 \\
- \left( \frac{k}{k + \gamma_2 h} \right) V_{n-1} + V_n &= \frac{\theta_2 \gamma_2 h}{k + \gamma_2 h} \\
- \left( \frac{k}{k + \gamma_1 h} \right) V_1 + V_0 &= \frac{\theta_1 \gamma_1 h}{k + \gamma_1 h}
\end{aligned} \tag{2.2.1}$$

где шаг сетки составляет  $h = \frac{b-a}{n}$ , узлы основной сетки  $x_i = a + ih$ ,  $i=0, n$ , узлы вспомогательной сетки  $x_{i+1/2} = a + (i + 0,5)h$ ,  $i=0, n-1$ .

В системе уравнений (2.4) формулы для расчёта коэффициентов имеют вид:

$$\begin{aligned}
a_i &= \left( \frac{1}{h} * \int_{x_{i-1}}^{x_i} \frac{dx}{k(x)} \right)^{-1} & i=1, n \\
\phi_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} f(x) d(x) & i=1, n-1 \\
d_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} q(x) d(x) & i=1, n-1
\end{aligned} \tag{2.2.2}$$

Разностная схема с аппроксимацией граничных условий оператором 1 порядка на сетке с числом разбиений  $n$  представляет собой СЛАУ с трехдиагональной матрицей и записывается в следующем виде:

$$\begin{bmatrix}
1 & -\frac{k}{k + \gamma_1 h} & 0 & \dots & 0 & 0 & 0 \\
\frac{a_1}{h^2} & -\frac{a_1 + a_2}{h^2} + d_1 & \frac{a_2}{h^2} & \dots & 0 & 0 & 0 \\
& \vdots & & \ddots & & \vdots & \\
0 & 0 & 0 & \dots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1} + a_n}{h^2} + d_{n-1} & \frac{a_n}{h^2} \\
0 & 0 & 0 & & 0 & -\frac{k}{k + \gamma_1 h} & 1
\end{bmatrix}
\begin{bmatrix}
V_0 \\
V_1 \\
\vdots \\
V_{n-1} \\
V_n
\end{bmatrix}
=
\begin{bmatrix}
\frac{\theta_1 \gamma_1 h}{k + \gamma_1 h} \\
-\phi_1 \\
\vdots \\
-\phi_{n-1} \\
\frac{\theta_2 \gamma_2 h}{k + \gamma_2 h}
\end{bmatrix}$$

### 2.2.2. Пример для тестирования

Для тестирования программы используется описанный ранее пример (2.1.2):

$$\begin{cases}
U''(x) - 3U(x) = -12 \text{ при } x \in [0, 1]. \\
-U'(0) = -10(U(0) - 25) \\
-U'(1) = 10(U(1) - 25)
\end{cases}$$

Разностная схема, полученная методом баланса, с аппроксимацией граничных условий оператором 1 порядка, на сетке с числом разбиений  $n$  в примере, предложенном для тестирования программы, имеет вид

$$\frac{V_{i+1}-V_i}{h^2} - \frac{V_i-V_{i-1}}{h^2} - 3V_i + 12 = 0, i=1, n-1$$

$$-(\frac{1}{1+10h}) V_{n-1} + V_n = \frac{250h}{1+10h} \quad (2.2.3)$$

$$-(\frac{1}{1+10h}) V_1 + V_0 = \frac{250h}{1+10h}$$

где шаг сетки составляет  $h = \frac{b-a}{n}$ , узлы основной сетки  $x_i = a+ih$ ,  $i=0, n$ , узлы вспомогательной сетки  $x_{i+1/2} = a + (i + 0,5)h$ ,  $i=0, n-1$ .

Разностная схема с аппроксимацией граничных условий оператором 1 порядка на сетке с числом разбиений  $n$  представляет собой СЛАУ с трехдиагональной матрицей и записывается в следующем виде:

$$\begin{bmatrix} 1 & -\frac{1}{1+10h} & 0 & \dots & 0 & 0 & 0 \\ \frac{1}{h^2} & -\frac{1+1}{h^2} + 3 & \frac{1}{h^2} & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & \frac{1}{h^2} & -\frac{1+1}{h^2} + 3 & \frac{1}{h^2} \\ 0 & 0 & 0 & \dots & 0 & -\frac{1}{1+10h} & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \frac{250h}{1+10h} \\ -12 \\ \vdots \\ \vdots \\ -12 \\ \frac{250h}{1+10h} \end{bmatrix}$$

### 2.2.3.Обоснование аппроксимации граничных условий

Обоснование аппроксимации граничных условий проведем для тестовой модельной задачи с параметрами (2.1.2).

Граничные условия задачи (2.1.2) имеют вид

$$-U'(0) + 10U(0) = 250$$

$$-U'(1) - 10U(1) = -250$$

Рассмотрим аппроксимацию граничных условий на правом конце стержня:

$$-U'(x_n) - 10U(x_n) = -250$$

$$U'(x_n) \sim \frac{U_n - U_{n-1}}{x_n - x_{n-1}}$$

$$U'(x_n) \sim \frac{U_n - U_{n-1}}{h}$$

$$\frac{U_n - U_{n-1}}{h} + 10 U_n = 250$$

Заменим  $U$  на  $V$ , так как обозначение  $U$  используется для точного решения дифференциального уравнения. Для записи разностной схемы будем использовать обозначение  $V$ .

$$\frac{V_n - V_{n-1}}{h} + 10 V_n = 250 \quad (2.2.4)$$

Преобразуем уравнение (2.2.4)

$$\frac{-1}{h} * V_{n-1} + \left(\frac{1+10h}{h}\right) V_n = 250$$

и запишем граничное условие так, чтобы оно соответствовало каноническому виду.

$$-\left(\frac{1}{1+10h}\right) * V_{n-1} + V_n = \frac{250h}{1+10h}$$

Рассмотрим аппроксимацию граничных условий на левом конце стержня:

$$-U'(x_0) + 10U(x_0) = 250$$

$$U'(x_0) \sim (U_1 - U_0) / (x_1 - x_0)$$

$$U'(x_0) \sim (U_1 - U_0) / h$$

$$(U_1 - U_0) / h - 10 U_0 = -250$$

Заменим  $U$  на  $V$ , так как обозначение  $U$  используется для точного решения дифференциального уравнения. Для записи разностной схемы будем использовать обозначение  $V$ .

$$\frac{V_1 - V_0}{h} + 10 V_0 = -250 \quad (2.2.5)$$

Преобразуем уравнение (2.2.5)

$$\frac{1}{h} * V_1 + \left(\frac{-1-10h}{h}\right) V_0 = -250$$

и запишем граничное условие так, чтобы оно соответствовало каноническому виду.

$$\left(\frac{1}{-1-10h}\right) * V_1 + V_0 = \frac{250h}{-1-10h}$$

Трёхдиагональная матрица в каноническом виде должна содержать уравнения

$$y_0 - k_1 y_1 = \mu_1 \quad (2.2.6)$$

$$y_n - k_2 y_{n-1} = \mu_2$$

Для тестовой модельной задачи с параметрами (2.1.2) коэффициенты таких уравнений (2.2.6) (СЛАУ разностной схемы) имеют вид

$$k_1 = \frac{1}{1+10h} ; \mu_1 = \frac{250h}{1+10h} \quad (2.2.7)$$

$$k_2 = \frac{1}{1+10h} ; \mu_2 = \frac{250h}{1+10h}$$

Именно такие коэффициенты используются в разностной схеме (2.2.3).

## 2.2.4. Существование и единственность решения разностной схемы методом прогонки

Покажем, как решается этот вопрос на примере тестовой модельной задачи с параметрами (2.1.2). Существование и единственность разностной схемы (2.2.3) вытекает из Теоремы 2 о применимости прогонки, см. Главу 1.

**Теорема 4.** При любом выборе числа разбиений  $n \geq 2$  решение разностной схемы (2.2.3) для третьей краевой задачи с параметрами (2.1.2) существует единственно и может быть найдено методом прогонки. Методом прогонки в этом случае является вычислительно устойчивым.

## 2.3. Разностная схема с улучшенной аппроксимацией граничных условий

### 2.3.1. Тестовая модельная задача

Рассмотрим тестовую модельную задачу (2.1.1). Напомним, что она представляет собой стационарное уравнение теплопроводности с краевыми условиями третьего рода.

$$\begin{cases} kU''(x) - qU(x) = -f \text{ при } x \in [0,1] \\ -kU'(0) = -\gamma_1 * (U(0) - \Theta_1) \\ -kU'(1) = \gamma_2 * (U(1) - \Theta_2) \end{cases}$$

где  $k > 0$  - положительное число,  $q \geq 0$  - неотрицательное число,  $f$  -любое вещественное число.  $\gamma_1 > 0, \gamma_2 > 0$ .

Разностная схема, полученная методом баланса с улучшенной аппроксимацией граничных условий (2 порядка), на сетке с числом разбиений  $n$  имеет вид:

$$\begin{aligned} a_{i+1} * \frac{V_{i+1} - V_i}{h^2} - a_i * \frac{V_i - V_{i-1}}{h^2} - d_i V_i + f_i &= 0, i=1, n-1 \\ -\left(\frac{k}{k + \mu_1 h + \frac{d_0}{2} h^2}\right) V_1 + V_0 &= \frac{\theta_1 \mu_1 h + \frac{\Phi_0}{2} h^2}{k + \mu_1 h + \frac{d_0}{2} h^2} \\ -\left(\frac{k}{k + \mu_2 h + \frac{d_n}{2} h^2}\right) V_{n-1} + V_n &= \frac{\theta_2 \mu_2 h + \frac{\Phi_n}{2} h^2}{k + \mu_2 h + \frac{d_n}{2} h^2} \end{aligned} \quad (2.3.1)$$

где шаг сетки составляет  $h = \frac{b-a}{n}$ , узлы основной сетки  $x_i = a + ih$ ,  $i=0, n$ , узлы вспомогательной сетки  $x_{i+1/2} = a + (i + 0,5)h$ ,  $i=0, n-1$ .

В системе уравнений (2.3.1) формулы для расчёта коэффициентов имеют вид:

$$\begin{aligned} a_i &= \left( \frac{1}{h} * \int_{x_{i-1}}^{x_i} \frac{dx}{k(x)} \right)^{-1} & i=1, n \\ \phi_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} f(x) d(x) & i=1, n-1 \\ d_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} q(x) d(x) & i=1, n-1 \end{aligned} \quad (2.3.2)$$

Разностная схема с улучшенной аппроксимацией граничных условий (2 порядка) на сетке с числом разбиений  $n$  представляет собой СЛАУ с трехдиагональной матрицей и записывается в следующем виде:

$$\begin{bmatrix} 1 & -\frac{k}{k+\gamma_1 h + \frac{3h^2}{2}} & 0 & \dots & 0 & 0 & 0 \\ \frac{a_1}{h^2} & -\frac{a_1+a_2}{h^2} + d_1 & \frac{a_2}{h^2} & \dots & 0 & 0 & 0 \\ & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1}+a_n}{h^2} + d_{n-1} & \frac{a_n}{h^2} \\ 0 & 0 & 0 & \dots & 0 & -\frac{k}{k+\gamma_2 h + \frac{3h^2}{2}} & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \frac{6h^2 + \theta_1 \gamma_1 h}{k + \gamma_1 h + \frac{3h^2}{2}} \\ -\phi_1 \\ \vdots \\ -\phi_{n-1} \\ \frac{6h^2 + \theta_2 \gamma_2 h}{k + \gamma_2 h + \frac{3h^2}{2}} \end{bmatrix}$$

### 2.3.2. Пример для тестирования

Для тестирования программы используется описанный ранее пример (2.1.2):

$$\begin{cases} U''(x) - 3U(x) = -12 \text{ при } x \in [0, 1]. \\ -U'(0) = -10(U(0) - 25) \\ -U'(1) = 10(U(1) - 25) \end{cases}$$

Разностная схема, полученная методом баланса с улучшенной аппроксимацией граничных условий (2 порядка) на сетке с числом разбиений  $n$  в примере, предложенном для тестирования программы, имеет вид

$$\begin{aligned} \frac{V_{i+1} - V_i}{h^2} - \frac{V_i - V_{i-1}}{h^2} - 3V_i + 12 &= 0, \quad i=1, n-1 \\ -\left(\frac{1}{1+10h+\frac{3}{2}h^2}\right)V_{n-1} + V_n &= \frac{250h+6h^2}{1+10h+\frac{3}{2}h^2} \\ -\left(\frac{1}{1+10h+\frac{3}{2}h^2}\right)V_1 + V_0 &= \frac{250h+6h^2}{1+10h+\frac{3}{2}h^2} \end{aligned} \quad (2.3.3)$$

где шаг сетки составляет  $h = \frac{b-a}{n}$ , узлы основной сетки  $x_i = a + ih$ ,  $i=0, n$ , узлы вспомогательной сетки  $x_{i+1/2} = a + (i + 0,5)h$ ,  $i=0, n-1$ .

Разностная схема с улучшенной аппроксимацией граничных условий (2 порядка) на сетке с числом разбиений  $n$  представляет собой СЛАУ с трехдиагональной матрицей и записывается в следующем виде:

$$\begin{bmatrix} 1 & -\frac{1}{1+10h+\frac{3h^2}{2}} & 0 & \dots & 0 & 0 & 0 \\ \frac{1}{h^2} & -\frac{1+1}{h^2} + 3 & \frac{1}{h^2} & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & \frac{1}{h^2} & -\frac{1+1}{h^2} + 3 & \frac{1}{h^2} \\ 0 & 0 & 0 & \dots & 0 & -\frac{1}{1+10h+\frac{3h^2}{2}} & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \frac{6h^2+250h}{1+10h+\frac{3h^2}{2}} \\ -12 \\ \vdots \\ -12 \\ \frac{6h^2+250h}{1+10h+\frac{3h^2}{2}} \end{bmatrix}$$

### 2.3.3. Обоснование аппроксимации граничных условий

Проведем это обоснование для тестовой модельной задачи с параметрами (2.1.2).

Построим аппроксимацию граничного условия на правом конце стержня методом баланса. Для этого используем дифференциальное уравнение задачи (2.1.2).

$$U''(x) - 3U(x) = -12$$

Проинтегрируем уравнение (2.2) на  $x \in [x_{n-1/2}; x_n]$

$$\int_{x_{n-1/2}}^{x_n} U'(x) dx - \int_{x_{n-1/2}}^{x_n} 3U(x) dx = \int_{x_{n-1/2}}^{x_n} -12 dx$$

и рассмотрим часть уравнения после знака равно:

$$\int_{x_{n-1/2}}^{x_n} -12 dx = 12 * h/2$$

А для модельных задач общего вида вводим коэффициент  $\phi_n$ .

$$\text{Тогда } \int_{x_{n-1/2}}^{x_n} -12 dx = \phi_n * h/2.$$

а поставленном нами случае  $\phi_n = 12$

Рассмотрим одно из частей уравнение до знака равенства  $\int_{x_{n-1/2}}^{x_n} 3U(x) dx$

В поставленном нами случае получаем

$$\int_{x_n - \frac{1}{2}}^{x_n} 3U(x)dx \sim U(x) * 3 * h/2$$

где коэффициент  $d_n=3$ .

Рассмотрим последнюю часть уравнения (2.1.2)  $\int_{x_n - \frac{1}{2}}^{x_n} U'(x)dx$

Используя определение теплового потока, производную температуры и граничное условие на левом конце стержня получим:

$$-U'(x_{n-1/2}) = (U(x_n) - U(x_{n-1}))/h$$

Объединим выведенные нами уравнения и получим:

$$250 - 10U(x_n) - (U(x_n) - U(x_{n-1}))/h - U(x_n) * 3 * h/2 = -12h/2 \quad (2.3.4)$$

Запишем уравнения (2.3.4) в таком виде, который соответствует канонической записи СЛАУ.

$$-1/(10h+1+3/2h^2)V_{n+1} + V_n = (6h^2+250h)/(10h+1+3/2h^2) \quad (2.3.5)$$

Построим аппроксимацию граничного условия на левом конце стержня методом баланса.

Для этого рассмотрим дифференциальное уравнение с параметрами (2.1.2)

$$U''(x) - 3U(x) = -12$$

и проинтегрируем его на  $x \in [x_0; x_{1+1/2}]$

$$\int_{x_0}^{x_1 + \frac{1}{2}} U'(x)dx - \int_{x_0}^{x_1 + \frac{1}{2}} 3U(x)dx = \int_{x_0}^{x_1 + \frac{1}{2}} -12dx$$

Рассмотрим часть уравнения после знака равно:

$$\int_{x_0}^{x_1 + \frac{1}{2}} -12dx = 12 * h/2$$

а для модельных задач общего вида вводим коэффициент  $\phi_n$ .

$$\text{Тогда } \int_{x_0}^{x_1 + \frac{1}{2}} -12dx = \phi_0 * h/2.$$

а в поставленном нами случае  $\phi_0 = 12$

Рассмотрим одно из частей уравнение до знака равенства  $\int_{x_0}^{x_1 + \frac{1}{2}} 3U(x)dx$

а в поставленном нами случае получаем



$$\int_{x_0}^{x_1+\frac{1}{2}} 3U(x)dx \sim U(x) * 3 * h/2$$

где  $d_0=3$ .

Рассмотрим последнюю часть уравнения (2.1.2)  $\int_{x_0}^{x_1+\frac{1}{2}} U'(x)dx$

Используя определение теплового потока, производную температуры и граничное условие на левом конце стержня получим:

$$-U'(x_0)=(U(x_1)-U(x_0))/h$$

Объединим выведенные нами уравнения и получим:

$$(U(x_1)-U(x_0))/h - (-250+10U(x_1)) - U(x_1) * 3 * h/2 = -12h/2 \quad (2.3.6)$$

Запишем уравнения (2.3.6) в таком виде, который соответствует канонической записи СЛАУ.

$$-1/(10h+1+3/2h^2)V_1+V_0 = (6h^2+250h)/(10h+1+3/2h^2) \quad (2.3.7)$$

Доказано, что построенные выше аппроксимации (2.3.5), (2.3.7) построенные методом баланса имеют 2 порядок. Такая аппроксимация называется улучшенной и имеет вид:

$$250-10U(x_n)- (U(x_n)-U(x_{n-1}))/h - U(x_n) * 3 * h/2 = -12h/2 \quad (2.3.8)$$

$$(U(x_1)-U(x_0))/h - (-250+10U(x_1)) - U(x_1) * 3 * h/2 = -12h/2$$

В каноническом виде улучшенная аппроксимация записывается следующим образом

$$-1/(10h+1+3/2h^2)V_{n+1}+V_n = (6h^2+250h)/(10h+1+3/2h^2) \quad (2.3.9)$$

$$-1/(10h+1+3/2h^2)V_1+V_0 = (6h^2+250h)/(10h+1+3/2h^2)$$

Именно такие условия (2.3.9) использованы в схеме (2.3.3).

## 2.3.4. Существование и единственность решения разностной схемы методом прогонки

Покажем, как решается этот вопрос на примере тестовой модельной задачи с параметрами (2.1.2). Существование и единственность разностной схемы (2.3.3) вытекает из Теоремы 2 о применимости прогонки, см. Главу 1.

**Теорема 4.** При любом выборе числа разбиений  $n \geq 2$  решение разностной схемы (2.3.3) для третьей краевой задачи с параметрами (2.1.2) существует единственно и может

быть найдено методом прогонки. Методом прогонки в этом случае является вычислительно устойчивым.

### 2.3.5. Контрольная таблица для передачи параметров тестовой задачи

При подготовке программной реализации численных методов решения краевых задач предложено использовать контрольные таблицы передачи параметров в процедуру метода прогонки при решении основной модельной задачи (2.1.2). С аппроксимацией граничных условий 1 порядка и улучшенной аппроксимацией граничных условий разработана следующая таблица:

Таблица 2.1

Соответствие обозначений математической модели (трехдиагональная СЛАУ) и программной реализации метода прогонки. Тестовая модельная задача

Обозначения в математической модели	Обозначения в программе	
	Аппроксимация граничных условий оператором 1 порядка	Улучшенная аппроксимация граничных условий (2 порядка)
$V_0, \dots, V_n$	$y[0], \dots, y[n]$	$y[0], \dots, y[n]$
$\mu_1, \mu_2$	$F[0] = \frac{250h}{1+10h} = F[n]$	$F[0] = \frac{250h+6h^2}{1+10h+\frac{3}{2}h^2} = F[n]$
$k_1, k_2$	$C[0] = -\frac{1}{1+10h} = A[n]$	$C[0] = -\frac{1}{1+10h+\frac{3}{2}h^2} = A[n]$
$\phi_1, \dots, \phi_{n-1}$	$F[1], \dots, F[n-1]$	$F[1], \dots, F[n-1]$
$A_1, \dots, A_{n-1}$	$A[1], \dots, A[n-1]$	$A[1], \dots, A[n-1]$
$B_1, \dots, B_{n-1}$	$C[1], \dots, C[n-1]$	$C[1], \dots, C[n-1]$
$C_1, \dots, C_{n-1}$	$B[1], \dots, B[n-1]$	$B[1], \dots, B[n-1]$
1,1 (элементы гл. значений)	$B[0] = 1 = B[n]$	$B[0] = 1 = B[n]$
0,0 (Не исп.)	$C[n] = 0 = A[0]$	$C[n] = 0 = A[0]$
$\alpha_i, \beta_i$ при $i = 1, n$	$\alpha_i = \text{Alfa}[i-1]; \beta_i = \text{Betta}[i-1]$	$\alpha_i = \text{Alfa}[i-1]; \beta_i = \text{Betta}[i-1]$

Описание программы приведено в приложении.

## 2.4. Решение основной модельной задачи с гладкими коэффициентами методом баланса

### 2.4.1. Постановка задачи.

Рассмотрим задачу (1.1) – третью краевую задачу для стационарного уравнения теплопроводности. Напомним, что она имеет вид:

$$\begin{cases} (k(x)U'(x))' - q(x)U(x) = -f(x), & \text{при } x \in [0,1] \\ -k(0)U'(0) = -\mu_1 * (U(0) - \Theta_1) \\ -k(1)U'(1) = \mu_2 * (U(1) - \Theta_2) \end{cases}$$

где  $k(x)$ ,  $q(x)$ ,  $f(x)$ ,  $\mu_1$ ,  $\mu_2$ ,  $\Theta_1$ ,  $\Theta_2$  заданы, причем  $k(x) > 0$ ,  $q(x) \geq 0$ ,  $x \in [0,1]$ ,  $\mu_1 > 0$ ,  $\mu_2 > 0$ .

Для тестирования программы используем модельную задачу (1.1) со следующим параметрами:

$$\begin{cases} (k(x)U'(x))' - q(x)U(x) = -f(x), & \text{при } x \in [0,1] \\ k(x) = x(1-x) + 1 \\ q(x) = \left(x - \frac{1}{2}\right)\left(x - \frac{1}{2}\right) \\ f(x) = xx(1-x)(1-x) * 10 \\ U(0) = 10(U(0) - 25) \\ U(1) = -10(U(1) - 25) \end{cases} \quad (2.4.1)$$

Задачу (2.4.1) будем называть основной модельной задачей. Решение задачи (2.4.1) будем искать только численным методом.

### 2.4.2. Схема с аппроксимацией граничных условий оператором 1 порядка

Рассмотрим основную модельную задачу (2.4.1).

Разностная схема полученная методом баланса, с аппроксимацией граничных условий оператором 1 порядка на сетке с числом разбиений  $n$  для задачи (2.4.1) имеет вид

$$\begin{aligned} a_{i+1} * \frac{V_{i+1} - V_i}{h^2} - a_i * \frac{V_i - V_{i-1}}{h^2} - d_i V_i + f &= 0, \quad i=1, n-1 \\ -\left(\frac{1}{1+10h}\right) V_{n-1} + V_n &= \frac{250h}{1+10h} \\ -\left(\frac{1}{1+10h}\right) V_1 + V_0 &= \frac{250h}{1+10h} \end{aligned} \quad (2.4.2)$$

где шаг сетки составляет  $h = \frac{b-a}{n}$ , узлы основной сетки  $x_i = a + ih$ ,  $i=0, n$ , узлы вспомогательной сетки  $x_{i+1/2} = a + (i + 0,5)h$ ,  $i=0, n-1$ .

В системе уравнений (2.4.2) формулы для расчёта коэффициентов имеют вид:

$$\begin{aligned} a_i &= \left( \frac{1}{h} * \int_{x_{i-1}}^{x_i} \frac{dx}{x(1-x)+1} \right)^{-1} & i=1, n \\ \phi_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} \left( x - \frac{1}{2} \right) \left( x - \frac{1}{2} \right) * dx & i=1, n-1 \\ d_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} xx(1-x)(1-x) * 10 * dx & i=1, n-1 \end{aligned} \quad (2.4.3)$$

Разностная схема с аппроксимацией граничных условий оператором первого порядка на сетке с числом разбиений  $n$  представляет собой СЛАУ с трехдиагональной матрицей и записывается в следующем виде:

$$\begin{bmatrix} 1 & -\frac{1}{1+10h} & 0 & \dots & 0 & 0 & 0 \\ \frac{a_1}{h^2} & -\frac{a_1+a_2}{h^2} + d_1 & \frac{a_2}{h^2} & \dots & 0 & 0 & 0 \\ & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1}+a_n}{h^2} + d_{n-1} & \frac{a_n}{h^2} \\ 0 & 0 & 0 & \dots & 0 & -\frac{1}{1+10h} & 1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \\ V_n \end{bmatrix} = \begin{bmatrix} \frac{250h}{1+10h} \\ -\phi_1 \\ \vdots \\ -\phi_{n-1} \\ \frac{250h}{1+10h} \end{bmatrix}$$

### 2.4.3. Схема с улучшенной аппроксимацией граничных условий

Рассмотрим основную модельную задачу (2.4.1).

Разностная схема, полученная методом баланса с улучшенной аппроксимацией граничных условий на сетке с числом разбиений  $n$  для задачи (2.4.1), имеет вид

$$\begin{aligned} a_{i+1} * \frac{V_{i+1} - V_i}{h^2} - a_i * \frac{V_i - V_{i-1}}{h^2} - d_i V_i + f &= 0, \quad i=1, n-1 \\ -\left( \frac{k_1}{a_1 + 10h + \frac{h^2}{2} d_n} \right) V_{n-1} + V_n &= \frac{250h + \phi_n * \frac{h^2}{2}}{k_1 + 10h + \frac{h^2}{2} d_n} \\ -\left( \frac{k_1}{k_1 + 10h + \frac{h^2}{2} d_0} \right) V_1 + V_0 &= \frac{250h + \phi_0 * \frac{h^2}{2}}{k_1 + 10h + \frac{h^2}{2} d_0} \end{aligned} \quad (2.4.4)$$

где шаг сетки составляет  $h = \frac{b-a}{n}$ , узлы основной сетки  $x_i = a + ih$ ,  $i=0, n$ , узлы вспомогательной сетки  $x_{i+1/2} = a + (i + 0,5)h$ ,  $i=0, n-1$ .

В системе уравнений (2.4.4) формулы для расчёта коэффициентов имеют вид:

$$a_i = \left( \frac{1}{h} * \int_{x_{i-1}}^{x_i} \frac{dx}{x(1-x)+1} \right)^{-1} \quad i=1, n$$

$$\begin{aligned}
\phi_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} \left(x - \frac{1}{2}\right) \left(x - \frac{1}{2}\right) * dx & i=1, n-1 \\
d_i &= \frac{1}{h} * \int_{x_{i-0,5}}^{x_{i+0,5}} (xx(1-x)(1-x) * 10) dx & i=1, n-1 \\
\phi_0 &= \frac{2}{h} * \int_0^{x_{0,5}} f(x) dx \\
d_0 &= \frac{2}{h} * \int_0^{x_{0,5}} q(x) dx \\
\phi_n &= \frac{2}{h} * \int_{x_{n-0,5}}^{x_1} f(x) dx \\
d_n &= \frac{2}{h} * \int_{x_{n-0,5}}^{x_1} q(x) dx
\end{aligned} \tag{3.5}$$

В программной реализации коэффициенты (2.4.5) вычисляются приближенно, методом трапеции. В литературе [2] показано, что замена коэффициентов на их приближенные значения, вычисленные по формуле трапеций, не влияют на порядок погрешности аппроксимаций и порядок сходимости схемы. Так как в задаче (2.4.1) все коэффициенты непрерывны по  $x$ , на участке  $[0,1]$ .

Разностная схема с улучшенной аппроксимацией граничных условий на сетке с числом разбиений  $n$  представляет собой СЛАУ с трехдиагональной матрицей и записывается в следующем виде:

$$\begin{bmatrix}
1 & -\frac{a_1}{a_1+10h+\frac{h^2}{2}d_0} & 0 & \dots & 0 & 0 & 0 \\
\frac{a_1}{h^2} & -\frac{a_1+a_2}{h^2} + d_1 & \frac{a_2}{h^2} & \dots & 0 & 0 & 0 \\
& \vdots & \ddots & & \vdots & & \\
0 & 0 & 0 & \dots & \frac{a_{n-1}}{h^2} & -\frac{a_{n-1}+a_n}{h^2} + d_{n-1} & \frac{a_n}{h^2} \\
0 & 0 & 0 & & 0 & -\frac{a_n}{a_n+10h+\frac{h^2}{2}d_2} & 1
\end{bmatrix}
\begin{bmatrix}
V_0 \\
V_1 \\
\vdots \\
V_{n-1} \\
V_n
\end{bmatrix}
=
\begin{bmatrix}
\frac{\phi_0 \frac{h^2}{2} + 250h}{a_1+10h+\frac{h^2}{2}d_0} \\
-\phi_1 \\
\vdots \\
-\phi_{n-1} \\
\frac{\phi_n \frac{h^2}{2} + 250h}{a_n+10h+\frac{h^2}{2}d_2}
\end{bmatrix}$$

#### 2.4.4. Обоснование аппроксимации граничных условий на основе метода баланса

Рассмотрим улучшенную аппроксимацию на левом конце стержня, для этого рассмотрим основное уравнение задачи (1.1)

$$(k(x)U'(x))' - q(x)U(x) = -f(x), \text{ при } x \in [0;1]$$

Интегрируем на участке  $[x_0; x_{1/2}]$

$$\int_{x_0}^{x_{1/2}} (k(x)U'(x))' dx - \int_{x_0}^{x_{1/2}} q(x)U(x) dx = - \int_{x_0}^{x_{1/2}} f(x) dx \tag{2.4.6}$$

Введем новое обозначение  $\phi_0 = \frac{2}{h} \int_{x_0}^{x_1/2} f(x) dx$

Рассмотрим часть уравнения (2.4.6) после равенства и запишем ее с учетом нового обозначения

$$-\int_{x_0}^{x_1/2} f(x) dx = -\phi_0 * \frac{h}{2}$$

Введем еще одно обозначение  $d_0 = \frac{2}{h} \int_{x_0}^{x_1/2} q(x) dx$

Тогда второе слагаемое уравнения (2.4.6) приблизительно вычислим и запишем в виде

$$\int_{x_0}^{x_1/2} q(x) U(x) dx = U(x_0) \int_{x_0}^{x_1/2} q(x) dx = U(x_0) * \frac{h}{2} * d_0 = U_0 * \frac{h}{2} * d_0$$

Запишем первое слагаемое с помощью первообразной

$$\int_{x_0}^{x_1/2} (k(x) U'(x))' dx = k(x_1/2) U'(x_1/2) - k(x_0) U'(x_0) = -\omega_1/2 + \omega_0 = \omega_0 - \omega_1/2$$

Это уравнение описывает баланс тепла на участке  $[x_0; x_1/2]$ . Это уравнение можно заменить другим

$$\omega_0 - \omega_1/2 - U_0 * \frac{h}{2} * d_0 = -\phi_0 * \frac{h}{2}$$

Вычислим поток  $\omega_1/2$ . Для этого заменим  $U'(x)$

$$U'(x) = \frac{U_{x+\frac{1}{2}} - U_x}{h/2}$$

$$U'(x) = \frac{\omega(x)}{k(x)}$$

и проинтегрируем это равенство на  $[x_0; x_1]$

$$\int_{x_0}^{x_1} U'(x) dx = - \int_{x_0}^{x_1} \frac{\omega(x)}{k(x)} dx$$

Произведем замену  $\int_{x_0}^{x_1} \frac{\omega(x)}{k(x)} dx$  на  $\omega(x_1/2) \int_{x_0}^{x_1} \frac{dx}{k(x)}$

Тогда

$$U(x_1) - U(x_0) = - \int_{x_0}^{x_1} \frac{\omega(x)}{k(x)} dx = \omega(x_1/2) \int_{x_0}^{x_1} \frac{dx}{k(x)}$$

$$\omega\left(\frac{x_1}{2}\right) = \frac{U(x_1) - U(x_0)}{\int_{x_0}^{x_1} \frac{dx}{k(x)}} = \frac{U_1 - U_0}{h * \frac{1}{h} * \int_{x_0}^{x_1} \frac{dx}{k(x)}} = \frac{U_1 - U_0}{h} * a_1$$

Из граничного условия на левом конце стержня выразим тепловой поток  $\omega_0$

$$\omega_0 = -10(U_0 - 25)$$

и подставляя формулы для потоков в уравнение, получим

$$-10(U_0 - 25) - \frac{U_0 - U_1}{h} * a_1 - U_0 * \frac{h}{2} * d_0 = -\phi_0 * \frac{h}{2}$$

Получим улучшенную аппроксимацию на левом конце стержня.

$$-10U_0 + 250 - \frac{U_1 - U_0}{h} * a_1 - U_0 * \frac{h}{2} * d_0 = -\phi_0 * \frac{h}{2}$$

$$-10U_0 h + 250h - a_1 U_1 - a_1 U_0 - U_0 * \frac{h * h}{2} * d_0 = -\phi_0 * \frac{h * h}{2}$$

$$U_0(-10h - a_1 - \frac{h * h}{2} * d_0) + a_1 U_1 = -\phi_0 * \frac{h * h}{2} - 250h$$

$$U_0(-10h - a_1 - \frac{h * h}{2} * d_0) - a_1 U_1 = \phi_0 * \frac{h * h}{2} + 250h$$

$$U_0 - \left( \frac{a_1}{10h - a_1 - \frac{h * h}{2} * d_0} \right) U_1 = \frac{\phi_0 * \frac{h * h}{2} + 250h}{10h - a_1 - \frac{h * h}{2} * d_0} \quad (2.4.7)$$

Уравнение (2.4.7) будет использовано в разностной схеме.

Рассмотрим улучшенную аппроксимацию на правом конце стержня, для этого рассмотрим основное уравнение задачи (1.1)

$$(k(x)U'(x))' - q(x)U(x) = -f(x), \text{ при } x \in [0; 1]$$

Интегрируем на участке  $[x_{n-1/2}; x_n]$

$$\int_{x_{n-1/2}}^{x_n} (k(x)U'(x))' dx - \int_{x_{n-1/2}}^{x_n} q(x)U(x) dx = \int_{x_{n-1/2}}^{x_n} f(x) dx \quad (2.4.8)$$

$$\text{Введем новое обозначение } f_n = \frac{2}{h} \int_{x_{n-1/2}}^{x_n} f(x) dx$$

Рассмотрим часть уравнения (2.4.8) после равенства и запишем ее с учетом нового обозначения

$$-\int_{x_{n-1/2}}^{x_n} f(x) dx = -\phi_n * \frac{h}{2}$$

$$\text{Введем еще одно обозначение } d_n = \frac{2}{h} \int_{x_{n-1/2}}^{x_n} q(x) dx$$

Тогда второе слагаемое уравнения (3.7) приблизительно вычислим и запишем в виде

$$\int_{x_{n-1/2}}^{x_n} q(x)U(x) dx = U(x_n) \int_{x_{n-1/2}}^{x_n} q(x) dx = U(x_n) * \frac{h}{2} * d_n = U_n * \frac{h}{2} * d_n$$

Запишем первое слагаемое с помощью первообразной



$$\int_{x_{n-1/2}}^{x_n} (k(x)U'(x))' dx = k(x_n)U'(x_n) - k(x_{n-1/2})U'(x_{n-1/2}) = -\omega_n + \omega_{n-1/2} = \omega_n - 1/2 - \omega_n$$

Это уравнение описывает баланс тепла на участке  $[x_{n-1/2}; x_n]$ . Это уравнение можно заменить другим

$$\omega_n - 1/2 - \omega_n - U_n * \frac{h}{2} * dn = \phi n * \frac{h}{2}$$

Вычислим поток  $\omega_{n-1/2}$ . Для этого заменим  $U'(x)$

$$U'(x) = \frac{U_x - U_{x-1/2}}{h/2}$$

$$U'(x) = \frac{\omega(x)}{k(x)}$$

Проинтегрируем это равенство на  $[x_{n-1/2}; x_n]$

$$\int_{x_{n-1/2}}^{x_n} U'(x) dx = - \int_{x_{n-1/2}}^{x_n} \frac{\omega(x)}{k(x)} dx$$

и произведем замену  $\int_{x_{n-1/2}}^{x_n} \frac{\omega(x)}{k(x)} dx$  на  $\omega(x_{n-1/2}) \int_{x_{n-1/2}}^{x_n} \frac{dx}{k(x)}$

Тогда

$$U(x_n) - U(x_{n-1/2}) = - \int_{x_{n-1/2}}^{x_n} \frac{\omega(x)}{k(x)} dx = \omega(x_{n-1/2}) \int_{x_{n-1/2}}^{x_n} \frac{dx}{k(x)}$$

$$\omega\left(x_{n-\frac{1}{2}}\right) = \frac{U(x_n) - U(x_{n-1/2})}{\int_{x_{n-1/2}}^{x_n} \frac{dx}{k(x)}} = \frac{U_n - U_{n-1/2}}{h * \frac{1}{h} * \int_{x_{n-1/2}}^{x_n} \frac{dx}{k(x)}} = \frac{U_n - U_{n-1/2}}{h} * an$$

Из граничного условия на левом конце стержня выразим тепловой поток  $\omega_n$

$$\omega_n = 10(U_n - 25)$$

Подставляя формулы для потоков в уравнение, получим

$$-10(U_n - 25) + \left(\frac{U_{n-1} - U_n}{h} * a_1\right) - U_n * \frac{h}{2} * dn = -\phi n * \frac{h}{2}$$

Получим улучшенную аппроксимацию на левом конце стержня.

$$-10U_n + 250 - \frac{U_{n-1} - U_n}{h} * an - U_n * \frac{h}{2} * dn = -\phi n * \frac{h}{2}$$

$$-10U_n h + 250h - anU_n + anU_{n-1} - U_n * \frac{h * h}{2} * dn = -\phi n * \frac{h * h}{2}$$

$$U_n(-10h - an * \frac{h * h}{2} * dn) + anU_{n-1} = -\phi n * \frac{h * h}{2} - 250h$$

$$U_0(-10h - a_1 - \frac{h^*h}{2} * d_0) - a_n U_1 = \phi n * \frac{h^*h}{2} + 250h$$

$$U_n - (\frac{a_n}{10h - a_n - \frac{h^*h}{2} * d_n}) U_1 = \frac{\phi n * \frac{h^*h}{2} + 250h}{10h - a_n - \frac{h^*h}{2} * d_n} \quad (2.4.9)$$

Уравнение (3.8) будет использовано в разностной схеме.

### 2.4.5. Контрольная таблица для передачи параметров основной задачи

При подготовке программной реализации численных методов решения краевых задач предложено использовать контрольные таблицы передачи параметров в процедуру метода прогонки при решении основной модельной задачи (2.4.1). С аппроксимацией граничных условий 1 порядка и улучшенной аппроксимацией граничных условий разработана следующая таблица:

Таблица 2.2

Соответствие обозначений математической модели (трехдиагональная СЛАУ) и программной реализации метода прогонки. Основная модельная задача. Основная сетка с числом разбиений  $n$

Обозначения в математической модели	Обозначения в программе	
	Аппроксимация граничных условий оператором 1 порядка	Улучшенная аппроксимация граничных условий (2 порядка)
$V_0, \dots, V_n$	$y[0], \dots, y[n]$	$y[0], \dots, y[n]$
$\mu_1, \mu_2$	$F[0] = \frac{250h}{1+10h} = F[n]$	$F[0] = \frac{250h+6h^2}{1+10h+\frac{3}{2}h^2} = F[n]$
$k_1, k_2$	$C[0] = -\frac{1}{1+10h} = A[n]$	$C[0] = -\frac{1}{1+10h} = A[n]$
$\phi_1, \dots, \phi_{n-1}$	$F[1], \dots, F[n-1]$	$F[1], \dots, F[n-1]$
$A_1, \dots, A_{n-1}$	$A[1], \dots, A[n-1]$	$A[1], \dots, A[n-1]$
$B_1, \dots, B_{n-1}$	$C[1], \dots, C[n-1]$	$C[1], \dots, C[n-1]$
$C_1, \dots, C_{n-1}$	$B[1], \dots, B[n-1]$	$B[1], \dots, B[n-1]$
1,1 (элементы гл. значений)	$B[0] = 1 = B[n]$	$B[0] = 1 = B[n]$
0,0 (Не исп.)	$C[n] = 0 = A[0]$	$C[n] = 0 = A[0]$
$\alpha_i, \beta_i$ при $i = 1, n$	$\alpha_i = Alfa[i-1]; \beta_i = Betta[i-1]$	$\alpha_i = Alfa[i-1]; \beta_i = Betta[i-1]$

Описание программы приведено в приложении.

Таблица 2.3

Соответствие обозначений математической модели (трехдиагональная СЛАУ) и программной реализации метода прогонки. Основная модельная задача. Контрольная сетка с числом разбиений  $2n$

В теории	В программе	
	Аппроксимация граничных условий оператором 1 порядка	Улучшенная аппроксимация граничных условий (2 порядка)
$V_0, \dots, V_n$	$y[0], \dots, y[n]$	$y[0], \dots, y[n]$
$\mu_1, \mu_2$	$F[0] = \frac{250h}{1+10h} = F[n]$	$F[0] = \frac{250h+6h^2}{1+10h+\frac{3}{2}h^2} = F[n]$
$k_1, k_2$	$C[0] = -\frac{1}{1+10h} = A[n]$	$C[0] = -\frac{1}{1+10h+\frac{3}{2}h^2} = A[n]$
$\phi_1, \dots, \phi_{n-1}$	$F[1], \dots, F[n-1]$	$F[1], \dots, F[n-1]$
$A_1, \dots, A_{n-1}$	$A[1], \dots, A[n-1]$	$A[1], \dots, A[n-1]$
$B_1, \dots, B_{n-1}$	$C[1], \dots, C[n-1]$	$C[1], \dots, C[n-1]$
$C_1, \dots, C_{n-1}$	$B[1], \dots, B[n-1]$	$B[1], \dots, B[n-1]$
1,1 (элементы гл. значений)	$B[0] = 1 = B[n]$	$B[0] = 1 = B[n]$
0,0 (Не исп.)	$C[n] = 0 = A[0]$	$C[n] = 0 = A[0]$
$\alpha_i, \beta_i$ при $i = 1, n$	$\alpha_i = \text{Alfa}[i-1]; \beta_i = \text{Betta}[i-1]$	$\alpha_i = \text{Alfa}[i-1]; \beta_i = \text{Betta}[i-1]$
$V_{1_0}, \dots, V_{1_{2n}}$	$\text{Tochn}[0], \dots, \text{Tochn}[2n],$	$\text{Tochn}[0], \dots, \text{Tochn}[2n],$

## 2.5. Решение задачи с разрывными коэффициентами методом баланса

### 2.5.1. Постановка задачи, ее физический смысл

Рассмотрим задачу

$$\begin{cases} \left( (k(x)U'(x))' - q(x)U(x) = -f(x), \text{ при } x \in [0, l] \right. \\ \left. \begin{aligned} -k(0)U'(0) + \mu_1 * U(0) &= 296.15 \\ k(l)U'(l) + \mu_2 * U(l) &= 296.15 \end{aligned} \right. \end{cases} \quad (2.5.1)$$

$$k(x) = \begin{cases} k_1, \text{ если } x \in [0, \xi) \\ k_2, \text{ если } x \in [\xi, l] \end{cases} \quad f(x) = \begin{cases} 0, \text{ если } x \notin [\xi - \varepsilon, \xi + \varepsilon] \\ f_1, \text{ если } x \in [\xi - \varepsilon, \xi + \varepsilon] \end{cases}$$

где  $k_1, k_2, q(x), f_1, \mu_1, \mu_2$ , заданы, причем  $k(x) > 0, q(x) = 0, x \in [0, l], \mu_1 > 0, \mu_2 > 0$ . А точка  $\xi$  – точка разрыва функции  $k(x)$ , точки  $\xi - \varepsilon, \xi + \varepsilon$  – точки разрыва функции  $f(x)$ .

Это третья краевая задача для стационарного уравнения теплопроводности, заданного на отрезке. Значение  $U(x)$  показывает температуру стержня в точке  $x$ .

Функция  $U(x)$  описывает стационарное распределение температур на тонком однородном в своем поперечном сечении стержне единичной длины. Левый конец стержня соответствует точке  $x = 0$ , правый – точке  $x = l$ . В поставленной задаче  $\mu_1$  – коэффициент передачи тепла от стержня в окружающую среду на левом конце. Аналогично,  $\mu_2$  – коэффициент передачи тепла от стержня в окружающую среду справа. Коэффициент  $k(x)$  – коэффициент теплопроводности стержня в точке  $x$ . Коэффициент  $f(x)$  – коэффициент нагрева. Коэффициент теплопроводности имеет одно значение до точки  $\xi$  и другое значение после этой точки. А нагрев происходит только на участке от  $\xi - \varepsilon$  до  $\xi + \varepsilon$ .

Исследуется образец цилиндрической формы с поперечным сечением  $S$ , часть образца слева изготовлена из кварца, длина участка 0.5 см. С другой стороны образец изготовлен из фианита, длина данного участка составляет 40 нм. Между этих двух участков образца находится кластер золота длиной 2 нм. Опыты проводятся с использованием монохромного лазера (т.е. излучение идет на одной длине волны  $\lambda = 650$  нм). Максимальная мощность лазера  $M = 1$  Ватт на пятно диаметром 100 микрон. Золотым кластером поглощается 10% мощности излучения, остальная мощность уходит в поглотитель, расположенный за образцом. Мощность лазера можно корректировать. Поглощаемая кластером золота мощность преобразуется в тепло, распространяющееся в обе стороны образца. Коэффициенты теплопроводности  $k_1$  кварцевого стекла = 1.38 Вт/(мК) и фианита  $k_2$

варьируется от 3 до 6Вт/(мК). Нужно найти температуру образца при проведении эксперимента в разных точках.

## 2.5.2 Математическая модель задачи

Рассматривается тонкий однородный в поперечном сечении образец (стержень) длины  $l$ , составленный из двух прозрачных материалов. Левому торцу образца соответствует координата  $x = a$ , правому торцу – координата  $x = b$ , вся длина  $l = b - a$ .

Точка контакта двух материалов имеет координату  $x = \xi$ , длина образца слева от точки контакта составляет  $l_1 = \xi - a$ , длина образца справа от точки контакта равна  $l_2 = b - \xi$ .

В точке контакта материалов находится наноструктурный слой, поглощающий излучение монохромного лазера мощности  $F$  и тем самым нагревающий весь образец.

Обозначим через  $U(x)$  температуру стержня в точке  $x = [a, b]$ . Стационарное (не зависящее от времени) распределение температуры на стержне будем описывать дифференциальным уравнением

$$(k(x)U'(x))' = -f(x)$$

Коэффициент теплопроводности образца  $k(x)$  имеет вид

$$k(x) = \begin{cases} k_1, & \text{если } x \in [a, \xi) \\ k_2, & \text{если } x \in (\xi, b] \end{cases}$$

где  $k_1, k_2$  – коэффициент теплопроводности каждого из материалов.

Чтобы описать тепловое воздействие излучения на точку контакта материалов в математической модели, вводим положительные параметры  $\varepsilon_1, \varepsilon_2$ .

Это длины участков слева и справа от точки контакта, на которых функция плотности источников тепла  $f(x)$  принимает ненулевые значения:

$$f(x) = \begin{cases} 0, & \text{если } x \notin [\xi - \varepsilon_1, \xi + \varepsilon_2] \\ f_1, & \text{если } x \in [\xi - \varepsilon_1, \xi] \\ f_2, & \text{если } x \in [\xi, \xi + \varepsilon_2] \end{cases}$$

Излучение, преобразуемое в тепло, составляет  $KPT * F$ , где  $KPT$  – коэффициент преобразования излучения в тепло,  $KPT > 0$ .

Поэтому значения  $f_1, f_2$  должны соответствовать следующим условиям:

$$\int_a^b f(x)dx = F * KPT$$

Так как

$$\int_a^b f(x)dx = \int_{\xi-\varepsilon_1}^{\xi+\varepsilon_2} f(x)dx = \int_{\xi-\varepsilon_1}^{\xi} f_1 dx + \int_{\xi}^{\xi+\varepsilon_2} f_2 dx = f_1 * \varepsilon_1 + f_2 * \varepsilon_2$$

получим условие

$$f_1 * \varepsilon_1 + f_2 * \varepsilon_2 = F * \text{KPT}$$

В рамках математической модели считаем, что тепловое воздействие слева и справа от точки контакта одинаково, поэтому значения  $f_1, f_2$  определим по формулам

$$f_1 = \text{KPT} * F / (2 * \varepsilon_1)$$

$$f_2 = \text{KPT} * F / (2 * \varepsilon_2)$$

В случае первой краевой задачи записываем граничные условия

$$U(a) = \mu_1$$

$$U(b) = \mu_2$$

где  $\mu_1, \mu_2$  – температура окружающей среды на левом и правом торце образца

В случае третьей краевой задачи записываем граничные условия

$$k(a) * U'(a) = \Theta_1 (U(a) - \mu_1)$$

$$k(b) * U'(b) = \Theta_2 (U(b) - \mu_2)$$

где  $\mu_1, \mu_2$  – температура окружающей среды на левом и правом торце образца

$\Theta_1, \Theta_2$  – коэффициент теплообмена с окружающей средой на левом и правом торце образца

Таким образом, математическая модель есть первая краевая задача вида

$$(k(x)U'(x))' = -f(x)$$

$$U(a) = \mu_1$$

$$U(b) = \mu_2$$

или третья краевая задача вида

$$(k(x)U'(x))' = -f(x)$$

$$k(a) * U'(a) = \Theta_1 (U(a) - \mu_1)$$

$$-k(b) * U'(b) = \Theta_2 (U(b) - \mu_2)$$

Точек разрыва коэффициентов уравнения – три:  $x = \xi$  (центральная),  $x = \xi - \varepsilon_1$  (левая),  $x = \xi + \varepsilon_2$  (правая).

В точках разрыва коэффициентов задачи ставятся условия сопряжения: их называют условиями непрерывности температуры  $U(x)$  и условиями непрерывности теплового потока  $W(x) = k(x)U'(x)$ .

Условия непрерывности температуры  $U(x)$  имеют вид

$$\lim_{x \rightarrow \xi - 0} U(x) = \lim_{x \rightarrow \xi + 0} U(x)$$

$$\lim_{x \rightarrow (\xi - \varepsilon_1) - 0} U(x) = \lim_{x \rightarrow (\xi - \varepsilon_1) + 0} U(x)$$

$$\lim_{x \rightarrow (\xi + \varepsilon_2) - 0} U(x) = \lim_{x \rightarrow (\xi + \varepsilon_2) + 0} U(x)$$

Условия непрерывности теплового потока  $W(x) = k(x)U'(x)$  имеют вид

$$\lim_{x \rightarrow \xi - 0} (-k(x)U'(x)) = \lim_{x \rightarrow \xi + 0} (-k(x)U'(x))$$

$$\lim_{x \rightarrow (\xi - \varepsilon_1) - 0} (-k(x)U'(x)) = \lim_{x \rightarrow (\xi - \varepsilon_1) + 0} (-k(x)U'(x))$$

$$\lim_{x \rightarrow (\xi + \varepsilon_2) - 0} (-k(x)U'(x)) = \lim_{x \rightarrow (\xi + \varepsilon_2) + 0} (-k(x)U'(x))$$

Первая и третья краевые задачи с условиями сопряжения поставлены математически корректно: решение таких задач существует, единственно и непрерывно зависит от параметров задачи.

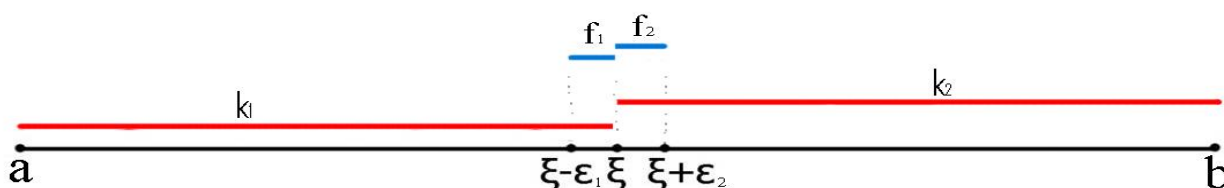


рис. 2.5.1. Распределение коэффициентов

### 2.5.3 Описание сетки и разностной схемы

Левый край сетки находится в точке  $a$ . Правый край –  $b$ . Вся сетка разделена на 4 отрезка в зависимости от точек разрыва. На каждом участке свое число разбиений:

$n_1$  – число разбиений от левого торца до левой точки разрыва



$n_2$  – разбиений от левой до центральной точки разрыва

$n_3$  – разбиений от центральной до правой точки разрыва

$n_4$  – разбиений от правой точки разрыва до правого торца

$$n_1 \geq 2, n_2 \geq 2, n_3 \geq 2, n_4 \geq 2$$

Каждому отрезку соответствует свой шаг:

$$h_1 = \frac{l_1 - \varepsilon_1}{n_1} \quad \text{– шаг сетки от левого торца до левой точки разрыва}$$

$$h_2 = \frac{\varepsilon_1}{n_2} \quad \text{– шаг сетки от левой до центральной точки разрыва}$$

$$h_3 = \frac{\varepsilon_2}{n_3} \quad \text{– шаг сетки от центральной до правой точки разрыва}$$

$$h_4 = \frac{l_2 - \varepsilon_2}{n_4} \quad \text{– шаг сетки от правой точки разрыва до правого торца}$$

Разностная схема, полученная методом баланса для первой краевой задачи имеет вид:

Граничное условие на левом торце образца:

$$V_0 = \mu_1$$

Уравнения на отрезке №1:

$$\frac{k_1}{h_1^2} * (V_{i+1} - 2V_i + V_{i-1}) = 0, \text{ где } i = 1, \dots, n_1 - 1$$

Уравнения для «левой точки разрыва»:

$$\frac{k_1}{\frac{1}{2}(h_1 + h_2)} \left( \frac{V_{i+1} - V_i}{h_2} - \frac{V_i - V_{i-1}}{h_1} \right) = -\frac{h_2 * f_1}{h_2 + h_1}, \text{ где } i = n_1$$

Уравнения на отрезке №2:

$$\frac{k_1}{h_2^2} * (V_{i+1} - 2V_i + V_{i-1}) = -f_1, \text{ где } i = n_1 + 1, \dots, n_1 + n_2 - 1$$

Уравнения для «центральной точки разрыва»:

$$\frac{1}{\frac{1}{2}(h_2 + h_3)} \left( k_1 \frac{V_{i+1} - V_i}{h_3} - k_2 \frac{V_i - V_{i-1}}{h_2} \right) = -\frac{h_2 * f_1 + h_3 * f_2}{h_2 + h_3}, \text{ где } i = n_1 + n_2$$

Уравнения на отрезке №3:

$$\frac{k_2}{h_3^2} * (V_{i+1} - 2V_i + V_{i-1}) = -f_2, \text{ где } i = n_1 + n_2 + 1, \dots, n_1 + n_2 + n_3 - 1$$

Уравнения для «правой точки разрыва»:

$$\frac{k_2}{\frac{1}{2}(h_3+h_4)} \left( \frac{V_{i+1}-V_i}{h_4} - \frac{V_i-V_{i-1}}{h_3} \right) = -\frac{h_3*f_2}{h_3+h_4}, \text{ где } i = n_1 + n_2 + n_3$$

Уравнения на отрезке №4:

$$\frac{k_2}{h_4^2} * (V_{i+1} - 2V_i + V_{i-1}) = 0, \text{ где } i = n_1 + n_2 + n_3 + 1, \dots, n_1 + n_2 + n_3 + n_4 - 1$$

Граничное условие на правом торце образца:

$$V_{n_1+n_2+n_3+n_4} = \mu_2$$

Разностная схема, полученная методом баланса для третьей краевой задачи имеет вид:

Граничное условие на левом торце образца:

$$-\left(\frac{k_1}{k_1+\Theta_1 h_1}\right) V_1 + V_0 = \frac{\Theta_1 \mu_1 h_1 + \phi_0 * \frac{h_1^2}{2}}{k_1 + \Theta_1 h_1}$$

$$\text{где } \phi_0 = \frac{2}{h_1} * \int_a^{x_a+0.5} f(x) dx$$

Уравнения на отрезке №1:

$$\frac{k_1}{h_1^2} * (V_{i+1} - 2V_i + V_{i-1}) = 0, \text{ где } i = 1, \dots, n_1-1$$

Уравнения для «левой точки разрыва»:

$$\frac{k_1}{\frac{1}{2}(h_1+h_2)} \left( \frac{V_{i+1}-V_i}{h_2} - \frac{V_i-V_{i-1}}{h_1} \right) = -\frac{h_2*f_1}{h_2+h_1}, \text{ где } i = n_1$$

Уравнения на отрезке №2:

$$\frac{k_1}{h_2^2} * (V_{i+1} - 2V_i + V_{i-1}) = -f_1, \text{ где } i = n_1+1, \dots, n_1 + n_2 - 1$$

Уравнения для «центральной точки разрыва»:

$$\frac{1}{\frac{1}{2}(h_2+h_3)} \left( k_1 \frac{V_{i+1}-V_i}{h_3} - k_2 \frac{V_i-V_{i-1}}{h_2} \right) = -\frac{h_2*f_1+h_3*f_2}{h_2+h_3}, \text{ где } i = n_1 + n_2$$

Уравнения на отрезке №3:

$$\frac{k_2}{h_3^2} * (V_{i+1} - 2V_i + V_{i-1}) = -f_2, \text{ где } i = n_1 + n_2 + 1, \dots, n_1 + n_2 + n_3 - 1$$

Уравнения для «правой точки разрыва»:

$$\frac{k_2}{\frac{1}{2}(h_3+h_4)} \left( \frac{V_{i+1}-V_i}{h_4} - \frac{V_i-V_{i-1}}{h_3} \right) = -\frac{h_3*f_2}{h_3+h_4}, \text{ где } i = n_1 + n_2 + n_3$$

Уравнения на отрезке №4:

$$\frac{k_2}{h_4^2} * (V_{i+1} - 2V_i + V_{i-1}) = 0, \text{ где } i = n_1 + n_2 + n_3 + 1, \dots, n_1 + n_2 + n_3 + n_4 - 1$$

Граничное условие на правом торце образца:

$$-\left(\frac{k_2}{k_2+\Theta_2 h_4}\right) V_{n_1+n_2+n_3+n_4-1} + V_{n_1+n_2+n_3+n_4} = \frac{\Theta_2 \mu_2 h_4 + \phi_n * \frac{h_4^2}{2}}{k_2 + \Theta_2 h_4}$$

$$\text{где } \phi_n = \frac{2}{h_4} * \int_{x_{b-0,5}}^{x_b} f(x) dx$$

## 2.5.4. Информационная модель

Параметры эксперимента

$\xi$  – координата точки контакта материалов №1 и №2, любое вещ. число

$l_1, l_2$  – длины участков, сделанных из материалов №1 и №2,  $l_1 > 0, l_2 > 0$ ;

$k_1, k_2$  – коэффициент теплопроводности материалов №1 и №2,  $k_1 > 0, k_2 > 0$ ;

$F$  – мощность излучения (лазера),  $F \geq 0$ ;

$KPT$  – коэффициент преобразования излучения в тепло,  $KPT > 0$ ;

$\epsilon_1, \epsilon_2$  – длины участков теплового воздействия слева и справа от точки контакта

$$0 < \epsilon_1 < l_1, 0 < \epsilon_2 < l_2$$

Для первой краевой задачи:

$\mu_1, \mu_2$  – температура окружающей среды на левом и правом торце образца, любое вещ. число

Для третьей краевой задачи:

$\mu_1, \mu_2$  – температура окружающей среды на левом и правом торце образца, любое вещ. число

$\Theta_1, \Theta_2$  – коэффициент теплообмена с окружающей средой на левом и правом торце образца,  $\Theta_1 > 0, \Theta_2 > 0$  (или другое согласованное с презентацией обозначение)

Число участков кусочно-равномерной сетки:

$n_1$  – от левого торца до левой точки разрыва

$n_2$  – от левой до центральной точки разрыва

$n_3$  – от центральной до правой точки разрыва

$n_4$  – от правой точки разрыва до правого торца

$n_1 \geq 2, n_2 \geq 2, n_3 \geq 2, n_4 \geq 2$

### 2.5.5. Отладочный пример

Для отладки программы использован следующий пример:

Параметры эксперимента

$\xi = 0.45$

$l_1 = 0.45$

$l_2 = 0.3$

$k_1 = 2$

$k_2 = 6$

$F = 3$

$KPT = 1$

$\varepsilon_1 = 0.15$

$\varepsilon_2 = 0.2$

Для первой краевой задачи:

$\mu_1 = 200$

$\mu_2 = 200$

Число участков кусочно-равномерной сетки:

$n_1 = 5 \quad n_2 = 5 \quad n_3 = 5 \quad n_4 = 5$

Расчет параметров задачи:

$l = 0.45 + 0.3 = 0.75$

$a = 0$  координата левого торца,  $a = \xi - 11$

$b = 0.75$  координата правого торца,  $b = \xi + 12$

$d1 = 0.3$  – вещественная координата левой точки разрыва  $d1 = \xi - \varepsilon1$

$d2 = 0.65$  – вещественная координата правой точки разрыва  $d2 = \xi + \varepsilon2$

$d = 0.45$  – вещественная координата центральной точки разрыва  $d = \xi$

$F = 3$ ,  $KPT = 1$

$\varepsilon1=0.15$ ,  $\varepsilon2=0.2$  – длины участков теплового воздействия слева и справа от точки контакта

$f1$  – плотность источника тепла слева от центральной точки разрыва,  
 $f1 = KPT \cdot F / (2 \cdot \varepsilon1) = 1 \cdot 3 / 0.3 = 10$

$f2$  – плотность источника тепла от центральной точки разрыва,  
 $f2 = KPT \cdot F / (2 \cdot \varepsilon2) = 1 \cdot 3 / 0.4 = 7.5$

Расчет параметров сетки:

$h1 = 0.06$  шаг от левого торца до левой точки разрыва

$h2 = 0.03$  шаг от левой до центральной точки разрыва

$h3 = 0.04$  шаг от центральной до правой точки разрыва

$h4 = 0.02$  шаг от правой точки разрыва до правого торца

Индекс-координаты крайних точек и точек разрыва:

$0$  – индекс-координата левого торца

$n\_left = 5$  – индекс-координата левой точки разрыва

$n\_centr = 10$  – индекс-координата центральной точки разрыва

$n\_right = 15$  – индекс-координата правой точки разрыва

$n = 20$  – индекс-координата правого торца

Число узлов сетки совпадает с числом строк в таблице результатов и с числом точек на графике:  $N\_Total = 21$ .

## 2.5.6. Контрольная таблица для передачи параметров разрывной задачи

Таблица 2.5.1

Соответствие обозначений математической модели (трехдиагональная СЛАУ) и программной реализации метода прогонки. Первая краевая задача с разрывными коэффициентами.

Обозначения в методе прогонки	В общем случае
$V_0, \dots, V_n$	$y[0], \dots, y[n]$
$\mu_1, \mu_2$	$F[0] = \mu_1; F[n] = \mu_2$
$\kappa_1, \kappa_2$	$C[0] = 0; A[n] = 0$
$\phi_1, \dots, \phi_{n-1}$	$F[1], \dots, F[n\_left-1] = 0$ $F[n\_left] = -\frac{h_2 * f_1}{h_2 + h_1}$ $F[n\_left + 1], \dots, F[n\_centr - 1] = -f_1$ $F[n\_centr] = -\frac{h_2 * f_1 + h_3 * f_2}{h_2 + h_3}$ $F[n\_centr + 1], \dots, F[n\_right - 1] = -f_2$ $F[n\_right] = -\frac{h_3 * f_2}{h_3 + h_4}$ $F[n\_right + 1], \dots, F[n - 1] = 0$

$A_1, \dots, A_{n-1}$	$A[1], \dots, A[n\_left-1] = \frac{k_1}{h_1 * h_1}$ $A[n\_left] = \frac{k_1}{0,5(h_1+h_2)h_1}$ $A[n\_left + 1], \dots, A[n\_centr - 1] = \frac{k_1}{h_2 * h_2}$ $A[n\_centr] = \frac{k_1}{0,5(h_2+h_3)h_2}$ $A[n\_centr + 1], \dots, A[n\_right - 1] = \frac{k_2}{h_3 * h_3}$ $A[n\_right] = \frac{k_2}{0,5(h_3+h_4)h_3}$ $A[n\_right + 1], \dots, A[n - 1] = \frac{k_2}{h_4 * h_4}$
$B_1, \dots, B_{n-1}$	$C[1], \dots, C[n\_left-1] = \frac{k_1}{h_1 * h_1}$ $C[n\_left] = \frac{k_1}{0,5(h_1+h_2)h_2}$ $C[n\_left + 1], \dots, C[n\_centr - 1] = \frac{k_1}{h_2 * h_2}$ $C[n\_centr] = \frac{k_2}{0,5(h_2+h_3)h_3}$ $C[n\_centr + 1], \dots, C[n\_right - 1] = \frac{k_2}{h_3 * h_3}$ $C[n\_right] = \frac{k_2}{0,5(h_3+h_4)h_4}$ $C[n\_right + 1], \dots, C[n - 1] = \frac{k_2}{h_4 * h_4}$
$-C_1, \dots, -C_{n-1}$	$B[1], \dots, B[n\_left-1] = -\frac{k_1+k_1}{h_1 * h_1}$ $B[n\_left] = -\left(\frac{k_1}{0,5(h_1+h_2)h_1} + \frac{k_1}{0,5(h_1+h_2)h_2}\right)$ $B[n\_left + 1], \dots, B[n\_centr - 1] = -\frac{k_1+k_1}{h_2 * h_2}$ $B[n\_centr] = -\left(\frac{k_1}{0,5(h_2+h_3)h_2} + \frac{k_2}{0,5(h_2+h_3)h_3}\right)$ $B[n\_centr + 1], \dots, B[n\_right - 1] = -\frac{k_2+k_2}{h_3 * h_3}$ $B[n\_right] = -\left(\frac{k_2}{0,5(h_3+h_4)h_3} + \frac{k_2}{0,5(h_3+h_4)h_4}\right)$ $B[n\_right + 1], \dots, B[n - 1] = -\frac{k_2+k_2}{h_4 * h_4}$

1,1 (элементы гл. значений)	$B[0] = 1 = B[n]$
0,0 (Не исп.)	$C[n] = 0 = A[0]$
$\alpha_i, \beta_i$ при $i = 1, n$	$\alpha_i = \text{Alfa}[i-1]; \beta_i = \text{Betta}[i-1]$



Соответствие обозначений математической модели (трехдиагональная СЛАУ) и программной реализации метода прогонки. Третья краевая задача с разрывными коэффициентами.

Обозначения в методе прогонки	В общем случае
$V_0, \dots, V_n$	$y[0], \dots, y[n]$
$\mu_1, \mu_2$	$F[0] = \frac{\Theta_1 \mu_1 h_1 + \phi_0 * \frac{h_1^2}{2}}{k_1 + \Theta_1 h_1 + \frac{h_1^2}{2} d_0}; F[n] = \frac{\Theta_2 \mu_2 h_4 + \phi_n * \frac{h_4^2}{2}}{k_2 + \Theta_2 h_4}$ <p>где <math>\phi_0 = \frac{2}{h_1} * \int_a^{x_{a+0,5}} f(x) dx</math></p> <p><math>\phi_n = \frac{2}{h_4} * \int_{x_{b-0,5}}^{x_b} f(x) dx</math></p>
$\kappa_1, \kappa_2$	$C[0] = -\left(\frac{k_1}{k_1 + \Theta_1 h_1}\right);$ $A[n] = -\left(\frac{k_2}{k_2 + \Theta_2 h_4}\right)$
$\phi_1, \dots, \phi_{n-1}$	$F[1], \dots, F[n\_left-1] = 0$ $F[n\_left] = -\frac{h_2 * f_1}{h_2 + h_1}$ $F[n\_left + 1], \dots, F[n\_centr - 1] = -f_1$ $F[n\_centr] = -\frac{h_2 * f_1 + h_3 * f_2}{h_2 + h_3}$ $F[n\_centr + 1], \dots, F[n\_right - 1] = -f_2$ $F[n\_right] = -\frac{h_3 * f_2}{h_3 + h_4}$ $F[n\_right + 1], \dots, F[n - 1] = 0$

$A_1, \dots, A_{n-1}$	$A[1], \dots, A[n\_left-1] = \frac{k_1}{h_1 * h_1}$ $A[n\_left] = \frac{k_1}{0,5(h_1+h_2)h_1}$ $A[n\_left + 1], \dots, A[n\_centr - 1] = \frac{k_1}{h_2 * h_2}$ $A[n\_centr] = \frac{k_1}{0,5(h_2+h_3)h_2}$ $A[n\_centr + 1], \dots, A[n\_right - 1] = \frac{k_2}{h_3 * h_3}$ $A[n\_right] = \frac{k_2}{0,5(h_3+h_4)h_3}$ $A[n\_right + 1], \dots, A[n - 1] = \frac{k_2}{h_4 * h_4}$
$B_1, \dots, B_{n-1}$	$C[1], \dots, C[n\_left-1] = \frac{k_1}{h_1 * h_1}$ $C[n\_left] = \frac{k_1}{0,5(h_1+h_2)h_2}$ $C[n\_left + 1], \dots, C[n\_centr - 1] = \frac{k_1}{h_2 * h_2}$ $C[n\_centr] = \frac{k_2}{0,5(h_2+h_3)h_3}$ $C[n\_centr + 1], \dots, C[n\_right - 1] = \frac{k_2}{h_3 * h_3}$ $C[n\_right] = \frac{k_2}{0,5(h_3+h_4)h_4}$ $C[n\_right + 1], \dots, C[n - 1] = \frac{k_2}{h_4 * h_4}$
$-C_1, \dots, -C_{n-1}$	$B[1], \dots, B[n\_left-1] = -\frac{k_1 + k_1}{h_1 * h_1}$ $B[n\_left] = -\left(\frac{k_1}{0,5(h_1+h_2)h_1} + \frac{k_1}{0,5(h_1+h_2)h_2}\right)$ $B[n\_left + 1], \dots, B[n\_centr - 1] = -\frac{k_1 + k_1}{h_2 * h_2}$ $B[n\_centr] = -\left(\frac{k_1}{0,5(h_2+h_3)h_2} + \frac{k_2}{0,5(h_2+h_3)h_3}\right)$ $B[n\_centr + 1], \dots, B[n\_right - 1] = -\frac{k_2 + k_2}{h_3 * h_3}$ $B[n\_right] = -\left(\frac{k_2}{0,5(h_3+h_4)h_3} + \frac{k_2}{0,5(h_3+h_4)h_4}\right)$ $B[n\_right + 1], \dots, B[n - 1] = -\frac{k_2 + k_2}{h_4 * h_4}$

1,1 (элементы гл. значений)	$B[0] = 1 = B[n]$
0,0 (Не исп.)	$C[n] = 0 = A[0]$
$\alpha_i, \beta_i$ при $i = 1, n$	$\alpha_i = \text{Alfa}[i-1]; \beta_i = \text{Betta}[i-1]$

## Глава III. Программная реализация

### 3.1. Основные принципы

Так как большинство задач математической физики не представляется возможным решить аналитически, существуют численные методы, которые данную проблему могут разрешить. С этой целью была написана программа, реализующая метод баланса с получением разностной схемы для краевой задачи уравнения теплопроводности и метод прогонки для разрешения разностной схемы.

Программа написана на языке C++, с использованием технологии Windows Forms для .NET Framework. Эта технология представляет собой набор управляемых библиотек, которые позволяют создавать интеллектуальные клиентские приложения, которые отображают сведения, запрашивают ввод данных от пользователей.

Написанная мной программа поддерживает только операционную систему Windows, на которой установлен .NET Framework. Она представляет собой файл с расширением .exe, который не может быть запущен на других системах. Хотя существуют специальные приложения, которые позволяют запускать приложения Windows на других операционных системах.

### 3.2. Высокоуровневая архитектура

В написанной мной программе решаются различные задачи. Для каждой задачи свои данные, но они решаются с помощью одинаковых методов. Поэтому при написании своей программы я использовал объектно-ориентированный подход, а именно реализовал несколько классов и организовал иерархию между ними.

Сначала реализован общий класс задач, в котором все функции были абстрактными. Далее два класса, которые отвечают за тестовую и основную задачи. В них уже были описаны функции для взятия интеграла и для функций, по которым будут браться интегралы. Далее классы конкретно под каждую задачу, в зависимости от типа задачи и выбранной схемы.

Перечислим реализованные классы:

- CommonTask – общий класс задач
- TestTask – тестовая задача в общем
- MainTask – основная задача в общем

- PhysicalTask – задача с разрывными коэффициентами в целом
- TestTaskUsual – тестовая задача с разностной схемой 1 порядка
- TestTaskImproved – тестовая задача с улучшенной аппроксимацией разностной схемы
- MainTaskUsual – основная задача с разностной схемой 1 порядка
- MainTaskImproved – основная задача с улучшенной аппроксимацией разностной схемы
- FirstPhysicalTask – первая краевая задача с разрывными коэффициентами
- ThirdPhysicalTask – третья краевая задача с разрывными коэффициентами.

Эти классы были иерархически связаны друг с другом. (рис. 3.1)

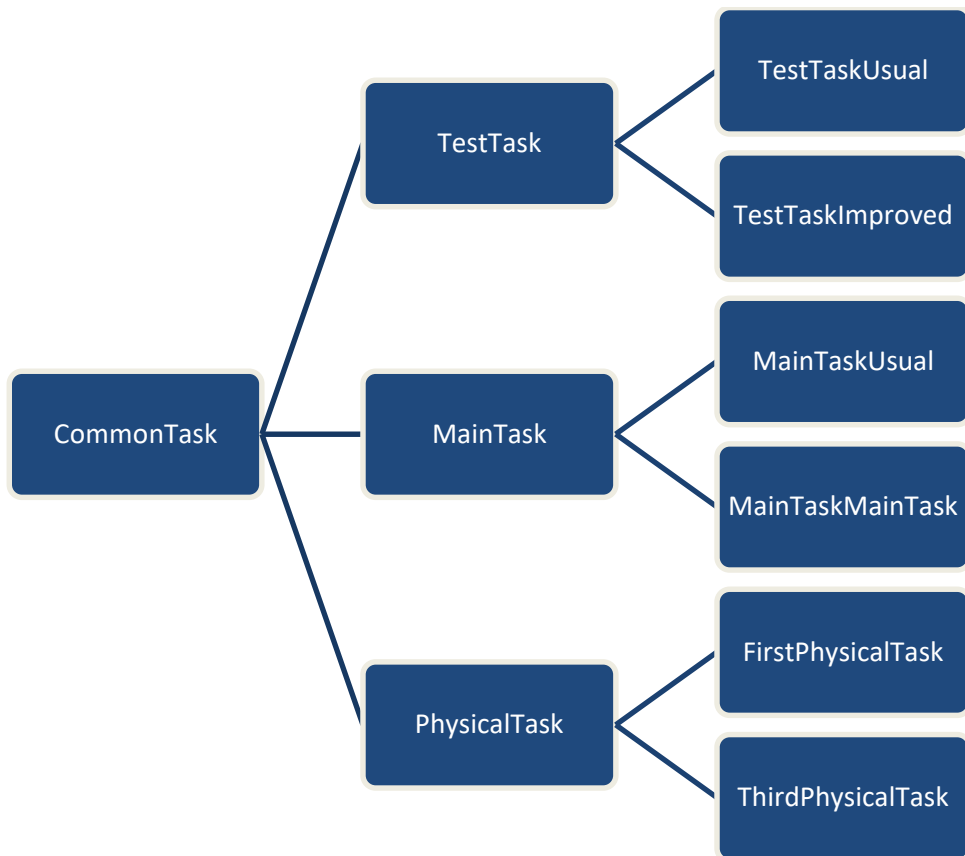


рис. 3.1. Иерархия классов в программе

### 3.3. Использование программы

В данном разделе рассказываются все функции программы и показывается её работоспособность.

В работе рассматривалась краевая задача для отыскания установившейся температуры стержня, на границах которого продолжается теплообмен с окружающей средой. Математическая постановка такой задачи предложена в [1].

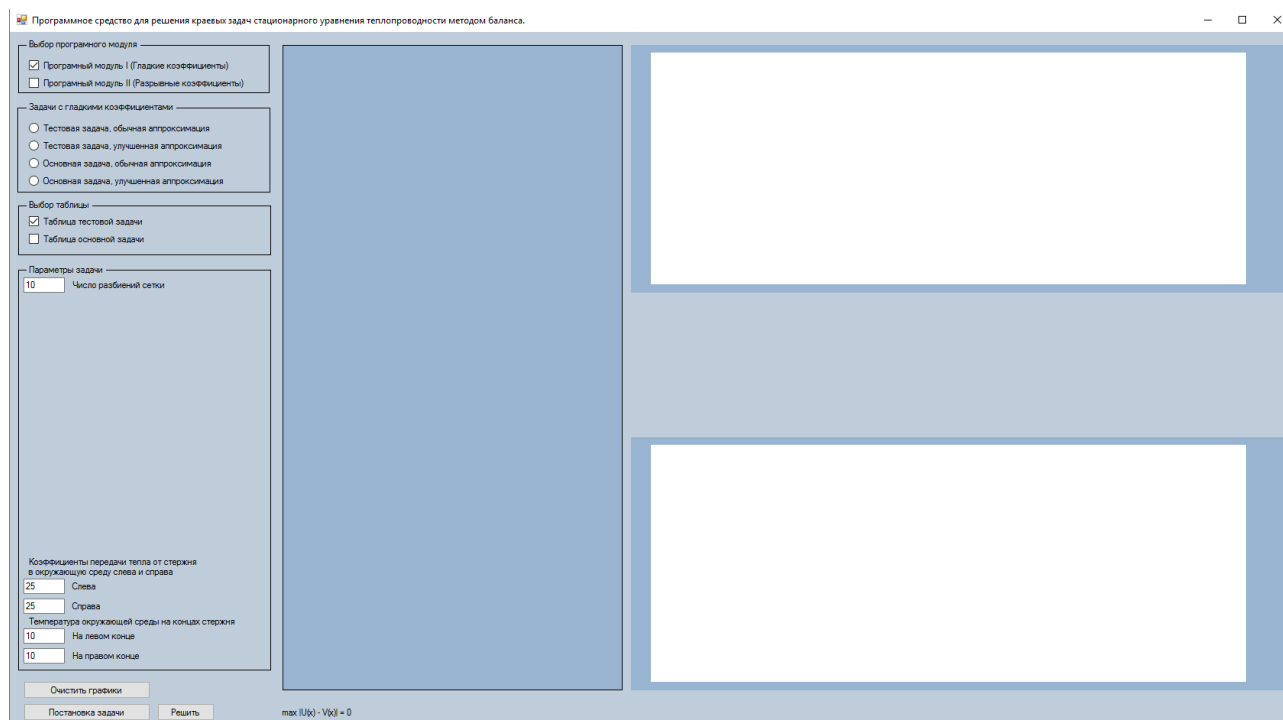


рис. 3.2. Окно программного модуля I

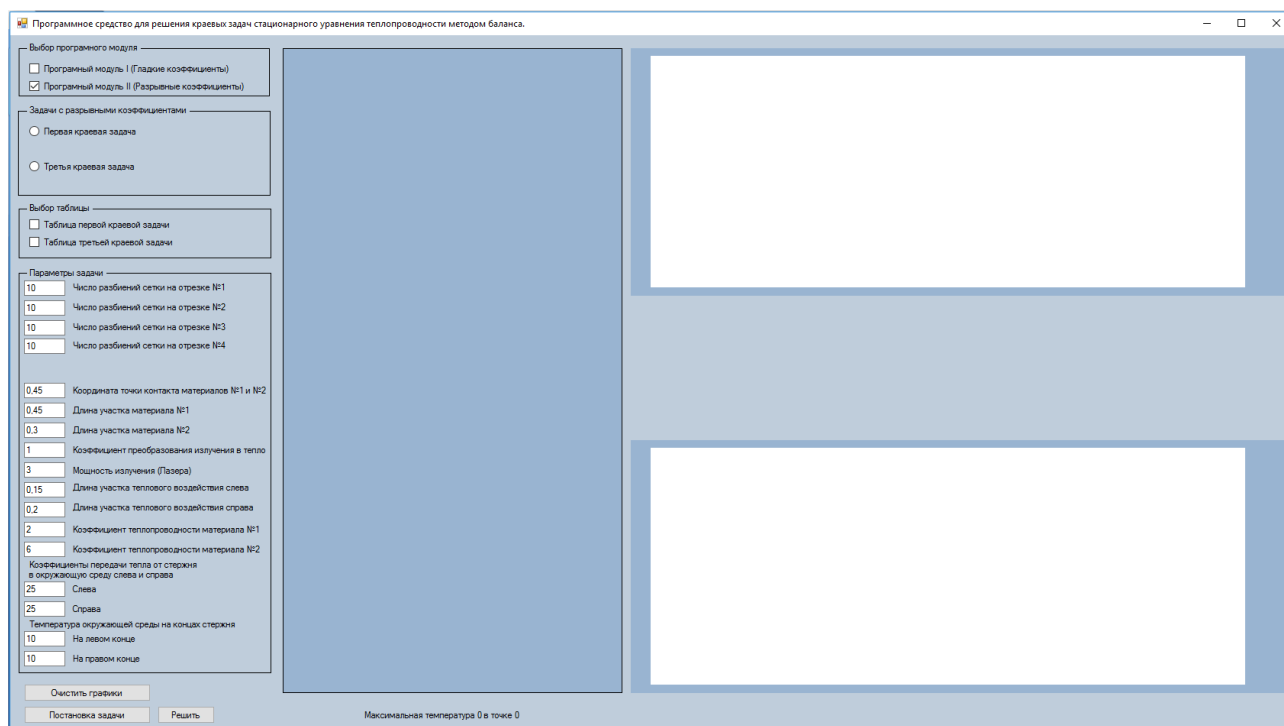


рис. 3.3. Окно программного модуля II

1. Запускается исполнительный файл программы SearchTemperature.exe;
2. Выбираем программный модуль I или II (рис. 3.4).

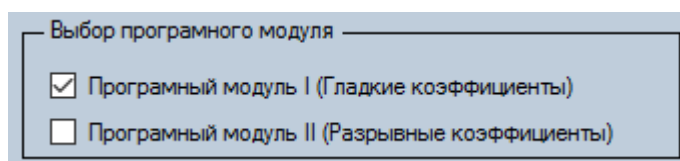


рис. 3.4. Выбор программного модуля

Программный модуль I (рис. 3.2) – модуль для решения задач с гладкими коэффициентами. В данном модуле реализованы следующие задачи:

- тестовая задача: третья краевая задача с гладкими коэффициентами решаемая методом баланса аппроксимацией граничных условий с помощью разностного оператора;
- тестовая задача: третья краевая задача с гладкими коэффициентами решаемая методом баланса улучшенной аппроксимацией граничных условий;
- основная задача: третья краевая задача с гладкими коэффициентами решаемая методом баланса аппроксимацией граничных условий с помощью разностного оператора;
- основная задача: третья краевая задача с гладкими коэффициентами решаемая методом баланса улучшенной аппроксимацией граничных условий;

1. Выбираем одну из четырех задач (рис. 3.5)

Задачи с гладкими коэффициентами

- ☐ Тестовая задача, обычная аппроксимация
- ☐ Тестовая задача, улучшенная аппроксимация
- ☐ Основная задача, обычная аппроксимация
- ☐ Основная задача, улучшенная аппроксимация

рис. 3.5. Задачи I модуля

2. Выбираем таблицу, в которой будут содержаться полученные значения температуры для тестовой или основной задачи (рис. 3.6)

Выбор таблицы

- ☒ Таблица тестовой задачи
- ☐ Таблица основной задачи

рис. 3.6. Выбор таблицы I модуля

3. Устанавливаем необходимые параметры метода (рис. 3.6). Вводим число разбиений на сетке, на каждом из четырех отрезков. Необходимо ввести так же координату точки контакта материалов №1 и №2, длины участков материалов №1 и №2, коэффициент преобразования излучения в тепло и мощность лазера, длины участков теплового воздействия слева и справа, коэффициенты теплопроводности обоих материалов, а так же температуру и коэффициенты передачи тепла на обоих торцах. Для третьей краевой задачи так же необходимо установить значения коэффициентов передачи тепла от стержня в окружающую среду слева и справа и температуру окружающей среды на левом и правом конце стержня. Для первой краевой только температуру окружающей среды на левом и правом конце стержня.



Параметры задачи

Число разбиений сетки

Коэффициенты передачи тепла от стержня  
в окружающую среду слева и справа

Слева

Справа

Температура окружающей среды на концах стержня

На левом конце

На правом конце

рис. 3.7. Параметры задачи I модуля

4. Когда все параметры установлены, то по нажатию кнопки «Решить» (рис. 3.8) будут получены значения температуры в каждом узле сетки. После нажатия кнопки «Решить» на экране можно увидеть что-то подобное (рис 3.9).

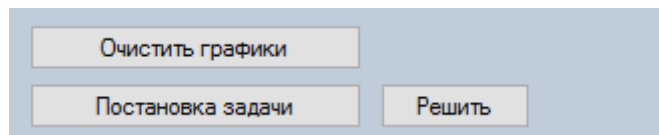


рис. 3.8. Кнопки действия

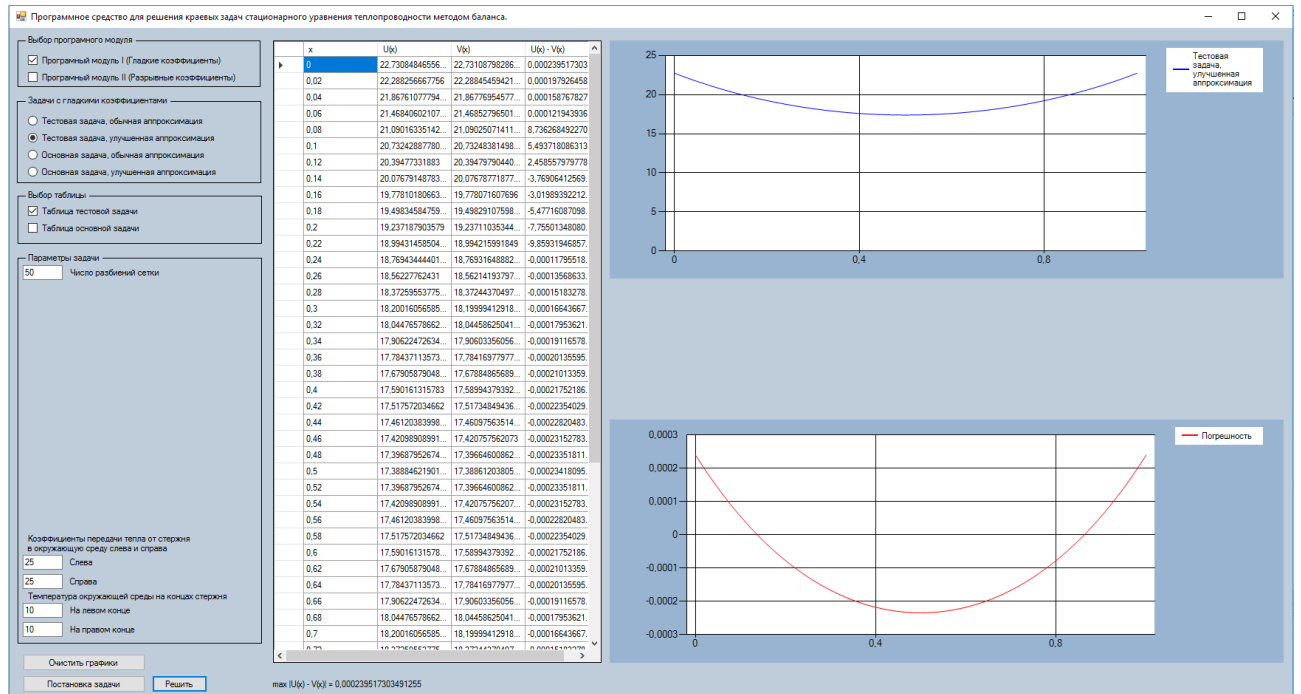


рис. 3.9. «Готовый расчет I модуля»

5. Под таблицей мы видим значение наибольшей погрешности (рис. 3.10): разность значений в каждом узле на обычной и удвоенной сетке в случае основной задачи и разность полученных значений методом и значений полученных аналитическим решением.

$$\max |U(x) - V(x)| = 0.000239517303491255$$

рис. 3.10. Наибольшая погрешность

Программный модуль II – модуль для решения задач с разрывными коэффициентами. В данном модуле реализованы следующие задачи:

- первая краевая задача с разрывными коэффициентами;
- третья краевая задача с разрывными коэффициентами решаемая методом баланса улучшенной аппроксимацией граничных условий.

6. Выбираем одну из двух задач (рис. 3.11)

Задачи с разрывными коэффициентами

☐ Первая краевая задача

☐ Третья краевая задача

рис. 3.11. Задачи II модуля

7. Выбираем таблицу, в которой будут содержаться полученные значения температуры для первой или третьей краевой задачи (рис. 3.12)

Выбор таблицы

☐ Таблица первой краевой задачи

☐ Таблица третьей краевой задачи

рис. 3.12. Выбор таблицы II модуля

8. Устанавливаем необходимые параметры метода (рис. 3.13). Есть возможность выбирать число разбиений на сетке. Для основной задачи есть возможность ввода значений коэффициентов передачи тепла от стержня в окружающую среду слева и справа и температуру окружающей среды на левом и правом конце стержня. Для тестовой задачи эти значения менять нельзя, т.к. для нее было найдено точное аналитическое решение.

Параметры задачи	
<input type="text" value="10"/>	Число разбиений сетки на отрезке №1
<input type="text" value="10"/>	Число разбиений сетки на отрезке №2
<input type="text" value="10"/>	Число разбиений сетки на отрезке №3
<input type="text" value="10"/>	Число разбиений сетки на отрезке №4
<input type="text" value="0,45"/>	Координата точки контакта материалов №1 и №2
<input type="text" value="0,45"/>	Длина участка материала №1
<input type="text" value="0,3"/>	Длина участка материала №2
<input type="text" value="1"/>	Коэффициент преобразования излучения в тепло
<input type="text" value="3"/>	Мощность излучения (Лазера)
<input type="text" value="0,15"/>	Длина участка теплового воздействия слева
<input type="text" value="0,2"/>	Длина участка теплового воздействия справа
<input type="text" value="2"/>	Коэффициент теплопроводности материала №1
<input type="text" value="6"/>	Коэффициент теплопроводности материала №2
Коэффициенты передачи тепла от стержня в окружающую среду слева и справа	
<input type="text" value="25"/>	Слева
<input type="text" value="25"/>	Справа
Температура окружающей среды на концах стержня	
<input type="text" value="10"/>	На левом конце
<input type="text" value="10"/>	На правом конце

рис. 3.13. Параметры задачи II модуля

9. Когда все параметры установлены, то по нажатию кнопки «Решить» (рис. 3.8) будут получены значения температуры в каждом узле сетки. После нажатия кнопки «Решить» на экране можно увидеть что-то подобное (рис 3.12).

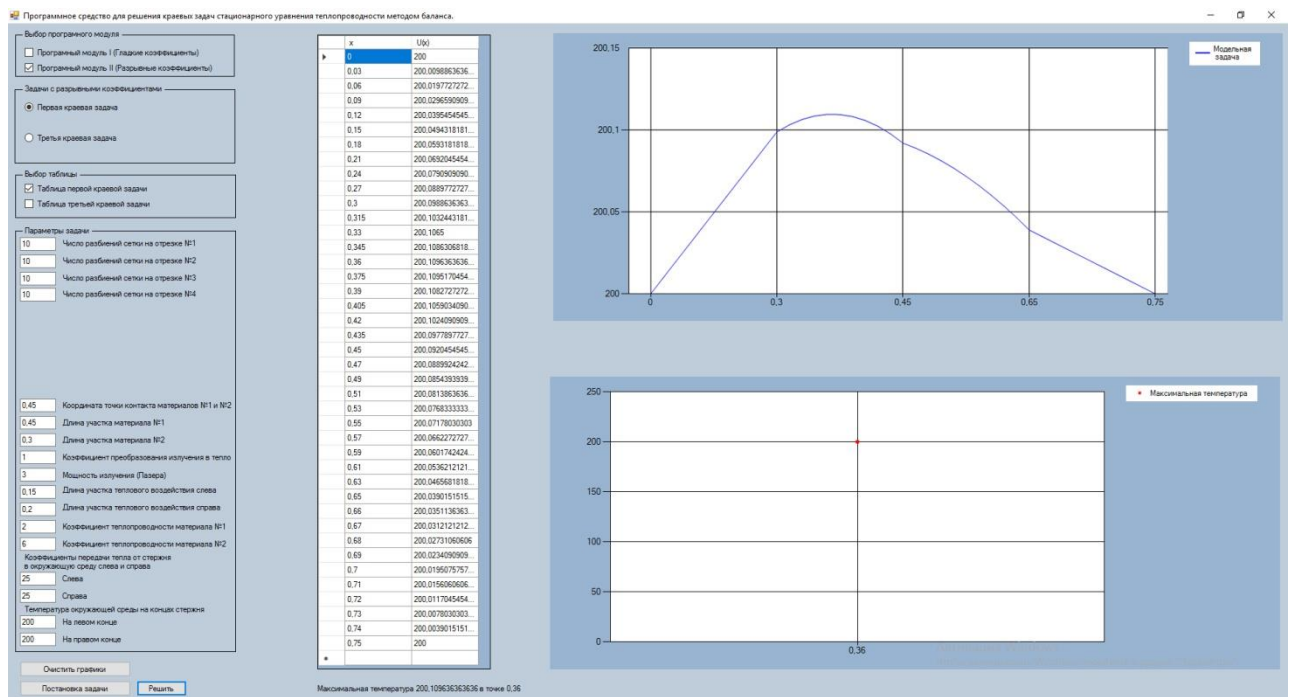


рис. 3.12. «Готовый расчет II модуля»

10. Под таблицей мы видим значение наибольшей температуры и координату узла в которой температура максимальна (рис. 3.13).

Максимальная температура 200,1096363636 в точке 0,36

рис. 3.13. Максимальная температура в точке

## Заключение

Была достигнута поставленная цель, а именно разработана и протестирована программа, решающая третью краевую задачу с гладкими коэффициентами и первую и третью краевые задачи с разрывными коэффициентами, методом баланса аппроксимацией граничных условий разностным оператором и улучшенной аппроксимацией построена консервативная разностная схема. Степень сходимости в программе соответствует теоретическим результатам. Реализованная программа может быть использована для изучения теплопроводности новых материалов.

В ходе достижения цели были выполнены все поставленные задачи:

Рассмотрены и изучены различные виды краевых задач. Третья краевая задача с гладкими коэффициентами и первая, третья краевые задачи с разрывными коэффициентами. Разобран метод баланса, которым получена консервативная разностная схема. Граничные условия аппроксимированы двумя способами: с помощью оператора 1-го порядка и с помощью схемы улучшенной аппроксимации (второй порядок), которая так же была получена методом баланса.

Теоретические результаты и практические совпали. Тем самым можно сделать вывод, что численный метод в программе реализован правильно.

## Литература

1. Тихонов А.Н., Самарский А.А. Уравнения математической физики. – М. Наука, 5-е изд., 1977. – 735 с.
2. Самарский А.А. Теория разностных схем. – М.: Наука, 1983. – 616 с.
3. Самарский А.А., Вабищевич П.Н. Численные методы решения обратных задач математической физики. – М.: Издательство ЛКИ, 2014. – 480 с.
4. Вержбицкий В.М. Численные методы. Математический анализ и обыкновенные дифференциальные уравнения. – Учебное пособие. – М.: Высшая школа, 2001. – 382 с.
5. Liskin D.A., Filatov D.O. Plasmon resonance induced photoconductivity of ZrO<sub>2</sub>(Y) films with embedded Au nanoparticles. / Liskin D.A., Filatov D.O., Gorchkov O.N., Gorchkov A.P., Antonov I.N., Shenina M.E., Zubkov S.Y., Sinutkin D.S. // Journal of Physics: Conference Series. – 2017. – Tome 816, – article identification number 012010.
6. Милюкова О.Ю., Тишкин В.Ф. Численный метод решения уравнений теплопроводности с разрывным коэффициентом на основе многосеточного метода [Электронный ресурс] // Препринты ИПМ им. М.В. Келдыша. – 2013. – №64. – 19 с. Режим доступа: [http://www.keldysh.ru/papers/2013/prep2013\\_64.pdf](http://www.keldysh.ru/papers/2013/prep2013_64.pdf), свободный.
7. Разностные схемы в задачах газовой динамики на неструктурированных сетках / Под ред. проф. В.Н. Емельянова, д.ф.м.н. К.Н. Волкова. – М.: ФИЗМАТЛИТ, 2015. – 416 с.
8. Кузнецов Г.В., Шеремет М.А. Математическое моделирование сложного теплопереноса в замкнутой прямоугольной области [Электронный ресурс] // Теплофизика и аэромеханика. – 2009. – Том 16. – №1. – С. 123–133. Режим доступа: <http://www.sibran.ru/upload/iblock/8e4/8e48d57952515037b88fae611a12b8aa.pdf>, свободный.
9. Краткий физико-технический справочник / Под общ. ред. К.П. Яковлева. – М.: Физматгиз, 1960.
10. Кузнецов Ю.А., Перова В.И., Стронгина Н.Р. Подготовка и защита выпускных квалификационных и научно-исследовательских работ. – Нижний Новгород: Нижегородский госуниверситет, 2013. – 51 с
11. Официальный сайт программы Microsoft Visual Studio. – URL: <https://visualstudio.microsoft.com>

12. Зиборов В.В. Microsoft Visual C++ 2010 в среде .NET. Библиотека программиста. – СПб: Питер, 2012. – 320 с.
13. Хортон А. Visual C++ 2010: полный курс. – М.: ООО «И.Д. Вильямс», 2011. – 1216 с.
14. Страуструп Б. Язык программирования C++: специальное издание. – М.: Издательство «Бином», 2019. – 1136 с.



## Приложение 1. Протоколы отладки программы

### Тестовая задача. Аппроксимация граничных условий (оператор 1 порядка)

Отладка проведена на равномерной сетке с числом разбиений  $n = 10$ .

```
Полная выдача .txt

a = 0
b = 1
h = 0.1
x[0] = 0
x[1] = 0.1
x[2] = 0.2
x[3] = 0.3
x[4] = 0.4
x[5] = 0.5
x[6] = 0.6
x[7] = 0.7
x[8] = 0.8
x[9] = 0.9
x[10] = 1

Xma1oe[0] = 0.05
Xma1oe[1] = 0.15
Xma1oe[2] = 0.25
Xma1oe[3] = 0.35
Xma1oe[4] = 0.45
Xma1oe[5] = 0.55
Xma1oe[6] = 0.65
Xma1oe[7] = 0.75
Xma1oe[8] = 0.85
Xma1oe[9] = 0.95

alfa[1] = 1
alfa[2] = 1
alfa[3] = 1
alfa[4] = 1
alfa[5] = 1
alfa[6] = 1
alfa[7] = 1
alfa[8] = 1
alfa[9] = 1
alfa[10] = 1

fi[0] = 0
fi[1] = 12
fi[2] = 12
fi[3] = 12
fi[4] = 12
fi[5] = 12
fi[6] = 12
fi[7] = 12
fi[8] = 12
fi[9] = 12
fi[10] = 0

d[0] = 0
d[1] = 3
d[2] = 3
d[3] = 3
d[4] = 3
d[5] = 3
d[6] = 3
d[7] = 3
d[8] = 3
d[9] = 3
d[10] = 0

A[0] = 0
A[1] = 100
A[2] = 100
A[3] = 100
A[4] = 100
A[5] = 100
```

Полная выдача .txt

A[6] =100  
A[7] =100  
A[8] =100  
A[9] =100  
A[10] =-0.5

B[0] =1  
B[1] =-203  
B[2] =-203  
B[3] =-203  
B[4] =-203  
B[5] =-203  
B[6] =-203  
B[7] =-203  
B[8] =-203  
B[9] =-203  
B[10] =1

C[0] =-0.5  
C[1] =100  
C[2] =100  
C[3] =100  
C[4] =100  
C[5] =100  
C[6] =100  
C[7] =100  
C[8] =100  
C[9] =100  
C[10] =0

F[0] =12.5  
F[1] =-12  
F[2] =-12  
F[3] =-12  
F[4] =-12  
F[5] =-12  
F[6] =-12  
F[7] =-12  
F[8] =-12  
F[9] =-12  
F[10] =12.5

alfa[0] =0.5  
alfa[1] =0.653595  
alfa[2] =0.72653  
alfa[3] =0.767183  
alfa[4] =0.79188  
alfa[5] =0.807676  
alfa[6] =0.818114  
alfa[7] =0.82516  
alfa[8] =0.829986  
alfa[9] =0.833323  
alfa[10] =-0

betta[0] =12.5  
betta[1] =8.24837  
betta[2] =6.07987  
betta[3] =4.75644  
betta[4] =3.86155  
betta[5] =3.21581  
betta[6] =2.72907  
betta[7] =2.35094  
betta[8] =2.05084  
betta[9] =1.80902  
betta[10] =22.979

y[0] =22.979  
y[1] =20.9579

Полная выдача .txt

y[2] =19.4456  
y[3] =18.3967  
y[4] =17.7797  
y[5] =17.576  
y[6] =17.7797  
y[7] =18.3967  
y[8] =19.4456  
y[9] =20.9579  
y[10] =22.979

x	U(x)	V(x)	U(x) - V(x)
0.000000	22.978962	22.731088	-0.247874
0.100000	20.957925	20.732484	-0.225441
0.200000	19.445625	19.237110	-0.208515
0.300000	18.396694	18.199994	-0.196700
0.400000	17.779664	17.589944	-0.189720
0.500000	17.576023	17.388612	-0.187411
0.600000	17.779664	17.589944	-0.189720
0.700000	18.396694	18.199994	-0.196700
0.800000	19.445625	19.237110	-0.208515
0.900000	20.957925	20.732484	-0.225441
1.000000	22.978962	22.731088	-0.247874

max |U(x) - V(x)| = 0.247874 для i=0,n

## Тестовая задача. Улучшенная аппроксимация граничных условий (2 порядка)

Отладка проведена на равномерной сетке с числом разбиений  $n = 10$ .

Проверка сходимости.txt

$n = 10$

$a = 0.00000000$

$b = 1.00000000$

$h = 0.10000000$

$x[0] = 0.00000000$

$x[1] = 0.10000000$

$x[2] = 0.20000000$

$x[3] = 0.30000000$

$x[4] = 0.40000000$

$x[5] = 0.50000000$

$x[6] = 0.60000000$

$x[7] = 0.70000000$

$x[8] = 0.80000000$

$x[9] = 0.90000000$

$x[10] = 1.00000000$

$Xmaloe[0] = 0.05000000$

$Xmaloe[1] = 0.15000000$

$Xmaloe[2] = 0.25000000$

$Xmaloe[3] = 0.35000000$

$Xmaloe[4] = 0.45000000$

$Xmaloe[5] = 0.55000000$

$Xmaloe[6] = 0.65000000$

$Xmaloe[7] = 0.75000000$

$Xmaloe[8] = 0.85000000$

$Xmaloe[9] = 0.95000000$

$aalfa[1] = 1.00000000$

$aalfa[2] = 1.00000000$

$aalfa[3] = 1.00000000$

$aalfa[4] = 1.00000000$

$aalfa[5] = 1.00000000$

$aalfa[6] = 1.00000000$

$aalfa[7] = 1.00000000$

$aalfa[8] = 1.00000000$

Проверка сходимости.txt

```
aalfa[9] = 1.000000000  
aalfa[10] = 1.000000000
```

```
fi[0] = 0.000000000  
fi[1] = 12.000000000  
fi[2] = 12.000000000  
fi[3] = 12.000000000  
fi[4] = 12.000000000  
fi[5] = 12.000000000  
fi[6] = 12.000000000  
fi[7] = 12.000000000  
fi[8] = 12.000000000  
fi[9] = 12.000000000  
fi[10] = 0.000000000
```

```
d[0] = 0.000000000  
d[1] = 3.000000000  
d[2] = 3.000000000  
d[3] = 3.000000000  
d[4] = 3.000000000  
d[5] = 3.000000000  
d[6] = 3.000000000  
d[7] = 3.000000000  
d[8] = 3.000000000  
d[9] = 3.000000000  
d[10] = 0.000000000
```

```
A[0] = 0.000000000  
A[1] = 100.000000000  
A[2] = 100.000000000  
A[3] = 100.000000000  
A[4] = 100.000000000  
A[5] = 100.000000000  
A[6] = 100.000000000  
A[7] = 100.000000000  
A[8] = 100.000000000
```

A[9] = 100.00000000  
A[10] = -0.49627792

B[0] = 1.00000000  
B[1] = -203.00000000  
B[2] = -203.00000000  
B[3] = -203.00000000  
B[4] = -203.00000000  
B[5] = -203.00000000  
B[6] = -203.00000000  
B[7] = -203.00000000  
B[8] = -203.00000000  
B[9] = -203.00000000  
B[10] = 1.00000000

C[0] = -0.49627792  
C[1] = 100.00000000  
C[2] = 100.00000000  
C[3] = 100.00000000  
C[4] = 100.00000000  
C[5] = 100.00000000  
C[6] = 100.00000000  
C[7] = 100.00000000  
C[8] = 100.00000000  
C[9] = 100.00000000  
C[10] = 0.00000000

F[0] = 12.43672457  
F[1] = -12.00000000  
F[2] = -12.00000000  
F[3] = -12.00000000  
F[4] = -12.00000000  
F[5] = -12.00000000  
F[6] = -12.00000000  
F[7] = -12.00000000  
F[8] = -12.00000000

Проверка сходимости.txt

F[9] = -12.00000000

F[10] = 12.43672457

alfa[0] = 0.49627792

alfa[1] = 0.65200861

alfa[2] = 0.72569394

alfa[3] = 0.76669122

alfa[4] = 0.79157211

alfa[5] = 0.80747535

alfa[6] = 0.81797942

alfa[7] = 0.82506850

alfa[8] = 0.82992270

alfa[9] = 0.83327965

betta[0] = 12.43672457

betta[1] = 8.18709249

betta[2] = 6.02840666

betta[3] = 4.71392938

betta[4] = 3.82640368

betta[5] = 3.18662371

betta[6] = 2.70475014

betta[7] = 2.33061236

betta[8] = 2.03381881

betta[9] = 1.79473339

y[0] = 22.72511721

y[1] = 20.73111119

y[2] = 19.23903849

y[3] = 18.20413696

y[4] = 17.59535953

y[5] = 17.39444288

y[6] = 17.59535953

y[7] = 18.20413696

y[8] = 19.23903849

y[9] = 20.73111119

y[10] = 22.72511721

Проверка сходимости.txt

x	U(x)	V(x)	U(x) - V(x)
0.00000000	22.72511721	22.73108798	0.00597077
0.10000000	20.73111119	20.73248381	0.00137263
0.20000000	19.23903849	19.23711035	-0.00192814
0.30000000	18.20413696	18.19999413	-0.00414283
0.40000000	17.59535953	17.58994379	-0.00541573
0.50000000	17.39444288	17.38861204	-0.00583085
0.60000000	17.59535953	17.58994379	-0.00541573
0.70000000	18.20413696	18.19999413	-0.00414283
0.80000000	19.23903849	19.23711035	-0.00192814
0.90000000	20.73111119	20.73248381	0.00137263
1.00000000	22.72511721	22.73108798	0.00597077

$\max |U(x) - V(x)| = 0.0059707691$  для  $i=0,n$



## Основная задача. Аппроксимация граничных условий (оператор 1 порядка)

Отладка проведена на равномерной сетке с числом разбиений  $n = 10$ .

```
n = 10
a = 0.00000000
b = 1.00000000
h = 0.10000000

x[0] = 0.00000000
x[1] = 0.10000000
x[2] = 0.20000000
x[3] = 0.30000000
x[4] = 0.40000000
x[5] = 0.50000000
x[6] = 0.60000000
x[7] = 0.70000000
x[8] = 0.80000000
x[9] = 0.90000000
x[10] = 1.00000000

Xmaloe[0] = 0.05000000
Xmaloe[1] = 0.15000000
Xmaloe[2] = 0.25000000
Xmaloe[3] = 0.35000000
Xmaloe[4] = 0.45000000
Xmaloe[5] = 0.55000000
Xmaloe[6] = 0.65000000
Xmaloe[7] = 0.75000000
Xmaloe[8] = 0.85000000
Xmaloe[9] = 0.95000000

aalfa[1] = 0.95693780
aalfa[2] = 0.88888889
aalfa[3] = 0.84388186
aalfa[4] = 0.81632653
aalfa[5] = 0.80321285
aalfa[6] = 0.80321285
aalfa[7] = 0.81632653
aalfa[8] = 0.84388186
```

```
alpha[9] = 0.88888889
alpha[10] = 0.95693780
```

```
fi[0] = 0.00000000
fi[1] = 0.09256250
fi[2] = 0.25706250
fi[3] = 0.43456250
fi[4] = 0.56506250
fi[5] = 0.61256250
fi[6] = 0.56506250
fi[7] = 0.43456250
fi[8] = 0.25706250
fi[9] = 0.09256250
fi[10] = 0.00000000
```

```
d[0] = 0.00000000
d[1] = 0.16250000
d[2] = 0.09250000
d[3] = 0.04250000
d[4] = 0.01250000
d[5] = 0.00250000
d[6] = 0.01250000
d[7] = 0.04250000
d[8] = 0.09250000
d[9] = 0.16250000
d[10] = 0.00000000
```

```
A[0] = 0.00000000
A[1] = 95.69377990
A[2] = 88.88888889
A[3] = 84.38818565
A[4] = 81.63265306
A[5] = 80.32128514
A[6] = 80.32128514
A[7] = 81.63265306
A[8] = 84.38818565
```

A[9] =88.88888889

A[10] =-0.50000000

B[0] =1.00000000

B[1] =-184.74516879

B[2] =-173.36957454

B[3] =-166.06333872

B[4] =-161.96643820

B[5] =-160.64507028

B[6] =-161.96643820

B[7] =-166.06333872

B[8] =-173.36957454

B[9] =-184.74516879

B[10] =1.00000000

C[0] =-0.50000000

C[1] =88.88888889

C[2] =84.38818565

C[3] =81.63265306

C[4] =80.32128514

C[5] =80.32128514

C[6] =81.63265306

C[7] =84.38818565

C[8] =88.88888889

C[9] =95.69377990

C[10] =0.00000000

F[0] =12.50000000

F[1] =-0.09256250

F[2] =-0.25706250

F[3] =-0.43456250

F[4] =-0.56506250

F[5] =-0.61256250

F[6] =-0.56506250

F[7] =-0.43456250

F[8] =-0.25706250

F[9] = -0.09256250  
F[10] = 12.50000000

alfa[0] = 0.50000000  
alfa[1] = 0.64930611  
alfa[2] = 0.72966407  
alfa[3] = 0.78126110  
alfa[4] = 0.81801881  
alfa[5] = 0.84601487  
alfa[6] = 0.86830836  
alfa[7] = 0.88660725  
alfa[8] = 0.90196378  
alfa[9] = 0.91511161

beta[0] = 12.50000000  
beta[1] = 8.73834807  
beta[2] = 6.71833776  
beta[3] = 5.43010875  
beta[4] = 4.52020720  
beta[5] = 3.83061458  
beta[6] = 3.27873292  
beta[7] = 2.81659327  
beta[8] = 2.41444241  
beta[9] = 2.05325058

y[0] = 24.93643670  
y[1] = 24.87287339  
y[2] = 24.84887339  
y[3] = 24.84778467  
y[4] = 24.85427219  
y[5] = 24.85769854  
y[6] = 24.85427219  
y[7] = 24.84778467  
y[8] = 24.84887339  
y[9] = 24.87287339  
y[10] = 24.93643670

x	U(x)	V(x)	U(x) - V(x)
0.00000000	24.93643670	24.92562430	-0.01081240
0.10000000	24.87287339	24.86304930	-0.00982409
0.20000000	24.84887339	24.83983886	-0.00903453
0.30000000	24.84778467	24.83932440	-0.00846027
0.40000000	24.85427219	24.84616014	-0.00811205
0.50000000	24.85769854	24.84970311	-0.00799543
0.60000000	24.85427219	24.84616014	-0.00811205
0.70000000	24.84778467	24.83932440	-0.00846027
0.80000000	24.84887339	24.83983886	-0.00903453
0.90000000	24.87287339	24.86304930	-0.00982409
1.00000000	24.93643670	24.92562430	-0.01081240

$\max |U(x) - V(x)| = 0.0108123974$  для  $i=0,n$

## Основная задача. Улучшенная аппроксимация граничных условий (2 порядка)

Отладка проведена на равномерной сетке с числом разбиений  $n = 10$ .

```
n = 10
a = 0.00000000
b = 1.00000000
h = 0.10000000

x[0] = 0.00000000
x[1] = 0.10000000
x[2] = 0.20000000
x[3] = 0.30000000
x[4] = 0.40000000
x[5] = 0.50000000
x[6] = 0.60000000
x[7] = 0.70000000
x[8] = 0.80000000
x[9] = 0.90000000
x[10] = 1.00000000

Xmaloe[0] = 0.05000000
Xmaloe[1] = 0.15000000
Xmaloe[2] = 0.25000000
Xmaloe[3] = 0.35000000
Xmaloe[4] = 0.45000000
Xmaloe[5] = 0.55000000
Xmaloe[6] = 0.65000000
Xmaloe[7] = 0.75000000
Xmaloe[8] = 0.85000000
Xmaloe[9] = 0.95000000

aalfa[1] = 0.95693780
aalfa[2] = 0.88888889
aalfa[3] = 0.84388186
aalfa[4] = 0.81632653
aalfa[5] = 0.80321285
aalfa[6] = 0.80321285
aalfa[7] = 0.81632653
aalfa[8] = 0.84388186
```

```
alpha[9]=0.88888889
alpha[10]=0.95693780
```

```
fi[0]=0.01128125
fi[1]=0.09256250
fi[2]=0.25706250
fi[3]=0.43456250
fi[4]=0.56506250
fi[5]=0.61256250
fi[6]=0.56506250
fi[7]=0.43456250
fi[8]=0.25706250
fi[9]=0.09256250
fi[10]=0.01128125
```

```
d[0]=0.22625000
d[1]=0.16250000
d[2]=0.09250000
d[3]=0.04250000
d[4]=0.01250000
d[5]=0.00250000
d[6]=0.01250000
d[7]=0.04250000
d[8]=0.09250000
d[9]=0.16250000
d[10]=0.22625000
```

```
A[0]=0.00000000
A[1]=95.69377990
A[2]=88.88888889
A[3]=84.38818565
A[4]=81.63265306
A[5]=80.32128514
A[6]=80.32128514
A[7]=81.63265306
A[8]=84.38818565
```

A[9] = 88.88888889

A[10] = -0.48871504

B[0] = 1.00000000

B[1] = -184.74516879

B[2] = -173.36957454

B[3] = -166.06333872

B[4] = -161.96643820

B[5] = -160.64507028

B[6] = -161.96643820

B[7] = -166.06333872

B[8] = -173.36957454

B[9] = -184.74516879

B[10] = 1.00000000

C[0] = -0.48871504

C[1] = 88.88888889

C[2] = 84.38818565

C[3] = 81.63265306

C[4] = 80.32128514

C[5] = 80.32128514

C[6] = 81.63265306

C[7] = 84.38818565

C[8] = 88.88888889

C[9] = 95.69377990

C[10] = 0.00000000

F[0] = 12.76770930

F[1] = -0.09256250

F[2] = -0.25706250

F[3] = -0.43456250

F[4] = -0.56506250

F[5] = -0.61256250

F[6] = -0.56506250

F[7] = -0.43456250

F[8] = -0.25706250



F[9] = -0.09256250  
F[10] = 12.76770930

alfa[0] = 0.48871504  
alfa[1] = 0.64422425  
alfa[2] = 0.72682522  
alfa[3] = 0.77947396  
alfa[4] = 0.81680522  
alfa[5] = 0.84514715  
alfa[6] = 0.86766512  
alfa[7] = 0.88611840  
alfa[8] = 0.90158637  
alfa[9] = 0.91481813

beta[0] = 12.76770930  
beta[1] = 8.85562438  
beta[2] = 6.78198475  
beta[3] = 5.46897319  
beta[4] = 4.54576411  
beta[5] = 3.84828501  
beta[6] = 3.29138975  
beta[7] = 2.82588950  
beta[8] = 2.42138912  
beta[9] = 2.05849514

y[0] = 24.91112801  
y[1] = 24.84764668  
y[2] = 24.82368881  
y[3] = 24.82261687  
y[4] = 24.82910862  
y[5] = 24.83253537  
y[6] = 24.82910862  
y[7] = 24.82261687  
y[8] = 24.82368881  
y[9] = 24.84764668  
y[10] = 24.91112801

x	U(x)	V(x)	U(x) - V(x)
0.00000000	24.91112801	24.91259250	0.00146449
0.10000000	24.84764668	24.85005924	0.00241256
0.20000000	24.82368881	24.82687013	0.00318132
0.30000000	24.82261687	24.82636406	0.00374719
0.40000000	24.82910862	24.83320182	0.00409320
0.50000000	24.83253537	24.83674495	0.00420958
0.60000000	24.82910862	24.83320182	0.00409320
0.70000000	24.82261687	24.82636406	0.00374719
0.80000000	24.82368881	24.82687013	0.00318132
0.90000000	24.84764668	24.85005924	0.00241256
1.00000000	24.91112801	24.91259250	0.00146449

$\max |U(x) - V(x)| = 0.0042095816$  для  $i=0,n$

## Приложение 2. Решение модельных задач с заданной погрешностью (точностью).

*Погрешностью* решения тестовой задачи на сетке с числом разбиений  $n$  называют максимальный модуль разности между точным решением и численным решением задачи в узлах сетки:

$$\max |U(x_i) - V(x_i)| \text{ для } i = 0; n \quad (\text{ПЗ-1})$$

*Точностью* решения модельной задачи на сетке с числом разбиений  $n$  называют максимальный модуль разности между численным решением, полученным на сетке с числом разбиений  $n$  и с численным решением, полученным на сетке с числом разбиений  $2n$ , задачи в узлах сетки:

$$\max |V(x_i) - V_2(x_i)| \text{ для } i = 0; n \quad (\text{ПЗ-2})$$

Пусть для тестовой задачи задана *погрешность*  $E_1$ . Это означает: нужно подобрать сетку с таким числом разбиений  $n$ , чтобы

$$\max |U(x_i) - V(x_i)| \leq E_1 \quad (\text{ПЗ-3})$$

Пусть для основной задачи задана *точность*  $E_2$ . Это означает: нужно подобрать сетку с таким числом разбиений  $n$ , чтобы

$$\max |V(x_i) - V_2(x_i)| \leq E_2 \quad (\text{ПЗ-4})$$

Далее приводятся результаты расчетов, в которых: модельные задачи решены с заданной погрешностью и точностью, и проведена проверка программы путем проверки порядка сходимости: проверяем соответствие порядка убывания погрешности (точности) порядку, заявленному в теореме о сходимости.

Для тестовой задачи задана погрешность  $E_1 = 0,5 * 10^{-4}$ .

Для основной задачи задана точность  $E_2 = 0,5 * 10^{-4}$ .

В таблицах ПЗ-1, ПЗ-2 показана погрешность решения тестовой задачи на сетках с разным числом разбиений, в таблицах ПЗ-3, ПЗ-4 показана точность решения основной задачи на сетках с разным числом разбиений.

Для каждой модельной задачи – для тестовой и основной – использована аппроксимация граничных условий оператором 1 порядка и улучшенная аппроксимация граничных условий (2 порядка).

Таблица ПЗ-1

Тестовая задача. Аппроксимация граничных условий (оператор 1 порядка)

$\max |U(x_i) - V(x_i)| = 0.247874$  для  $i = 0$ ; 10, число разбиений  $n = 10$ .

$\max |U(x_i) - V(x_i)| = 0.124633$  для  $i = 0$ ; 20, число разбиений  $n = 20$ .

$\max |U(x_i) - V(x_i)| = 0.062486$  для  $i = 0$ ; 40, число разбиений  $n = 40$ .

$\max |U(x_i) - V(x_i)| = 0.0312848$  для  $i = 0$ ; 80, число разбиений  $n = 80$ .

$\max |U(x_i) - V(x_i)| = 0.0156528$  для  $i = 0$ ; 160, число разбиений  $n = 160$ .

$\max |U(x_i) - V(x_i)| = 0.00782898$  для  $i = 0$ ; 320, число разбиений  $n = 320$ .

$\max |U(x_i) - V(x_i)| = 0.00391513$  для  $i = 0$ ; 640, число разбиений  $n = 640$ .

$\max |U(x_i) - V(x_i)| = 0.00313221$  для  $i = 0$ ; 800, число разбиений  $n = 800$ .

Из таблицы ПЗ-1 видно:

- 1) при удвоении числа участков погрешность падает примерно в 2 раза;
- 2) при увеличении числа участков в 10 раз, например,  $n=80$  и  $n=800$ , погрешность упала в 10 раз.

Таблица ПЗ-2

Тестовая задача. Улучшенная аппроксимация граничных условий (2 порядка)

$\max |U(x_i) - V(x_i)| = 0.005970769141413$  для  $i = 0$ ; 10, число разбиений  $n = 10$ .

$\max |U(x_i) - V(x_i)| = 0.001496041587558$  для  $i = 0$ ; 20, число разбиений  $n = 20$ .

$\max |U(x_i) - V(x_i)| = 0.000374220546675$  для  $i = 0$ ; 40, число разбиений  $n = 40$ .

$\max |U(x_i) - V(x_i)| = 0.000093568283791$  для  $i = 0$ ; 80, число разбиений  $n = 80$ .

$\max |U(x_i) - V(x_i)| = 0.000023392893318$  для  $i = 0$ ; 160, число разбиений  $n = 160$ .

$\max |U(x_i) - V(x_i)| = 0.000005848274679$  для  $i = 0$ ; 320, число разбиений  $n = 320$ .

$\max |U(x_i) - V(x_i)| = 0.000003742899981$  для  $i = 0$ ; 400, число разбиений  $n = 400$ .

Из таблицы ПЗ-2 видно:

- 1) при удвоении числа участков погрешность падает примерно в 4 раза.

2) при увеличении числа участков в 10 раз, например,  $n=40$  и  $n=400$ , погрешность упала в 100 раз.

Основная задача. Аппроксимация граничных условий (оператор 1 порядка)

$\max |V(x_i) - V_2(x_i)| = 0.0108123974$  для  $i = 0; 10$ , число разбиений  $n = 10$ .

$\max |V(x_i) - V_2(x_i)| = 0.0060558901$  для  $i = 0; 20$ , число разбиений  $n = 20$ .

$\max |V(x_i) - V_2(x_i)| = 0.0031844819$  для  $i = 0; 40$ , число разбиений  $n = 40$ .

$\max |V(x_i) - V_2(x_i)| = 0.0016304076$  для  $i = 0; 80$ , число разбиений  $n = 80$ .

$\max |V(x_i) - V_2(x_i)| = 0.0008246098$  для  $i = 0; 160$ , число разбиений  $n = 160$ .

$\max |V(x_i) - V_2(x_i)| = 0.0003183481$  для  $i = 0; 400$ , число разбиений  $n = 400$ .

Из таблицы ПЗ-3 видно:

- 1) при удвоении числа участков погрешность падает примерно в 2 раза.
- 2) при увеличении числа участков в 10 раз, например,  $n=40$  и  $n=400$ , погрешность упала в 10 раз.

Основная задача. Улучшенная аппроксимация граничных условий (2 порядка)

$\max |V(x_i) - V_2(x_i)| = 0.0042095816$  для  $i = 0; 10$ , число разбиений  $n = 10$ .

$\max |V(x_i) - V_2(x_i)| = 0.0010634913$  для  $i = 0; 20$ , число разбиений  $n = 20$ .

$\max |V(x_i) - V_2(x_i)| = 0.0002671986$  для  $i = 0; 40$ , число разбиений  $n = 40$ .

$\max |V(x_i) - V_2(x_i)| = 0.0000669614$  для  $i = 0; 80$ , число разбиений  $n = 80$ .

$\max |V(x_i) - V_2(x_i)| = 0.0000167603$  для  $i = 0; 160$ , число разбиений  $n = 160$ .

$\max |V(x_i) - V_2(x_i)| = 0.0000026835$  для  $i = 0; 400$ , число разбиений  $n = 400$ .

Из таблицы ПЗ-4 видно:

- 1) при удвоении числа участков погрешность падает примерно в 4 раза.
- 2) при увеличении числа участков в 10 раз, например,  $n=40$  и  $n=400$ , погрешность упала в 100 раз.

Таким образом, для схем с аппроксимацией граничных условий оператором 1 порядка погрешность (точность) убывает с первым порядком. Для схем с улучшенной аппроксимацией граничных условий (2 порядка) погрешность (точность) убывает со вторым порядком. Полученные результаты подтверждают корректную работу программы.

Каждая из модельных задач с помощью улучшенной аппроксимацией, решена с заданной погрешностью (точностью), а именно, погрешность  $E_1 = 0,5 * 10^{-4}$  и точность  $E_2 = 0,5 * 10^{-4}$ . Для этого потребовалась сетка с числом разбиений  $n = 160$  или более.

Дополнительные расчеты показывают, что для решений модельных задач с помощью улучшенной аппроксимации граничных условий, с указанной выше погрешностью (точностью), а именно, погрешностью  $E_1 = 0,5 * 10^{-4}$  и точностью  $E_2 = 0,5 * 10^{-4}$ , нужно брать число разбиений  $n \geq 140$ .

Для достижения указанной выше погрешности (точности) для решения модельных задач с помощью аппроксимации граничных условий оператором 1 порядка, а именно погрешности  $E_1 = 0,5 * 10^{-4}$  и точности  $E_2 = 0,5 * 10^{-4}$ , нужно брать число разбиений  $n \geq 1000$ .

Таблица ПЗ-5

Число разбиений необходимые для достижения заданной погрешности (точности)

$$E_1 = 0,5 * 10^{-4} (E_2 = 0,5 * 10^{-4})$$

Тип модельной задачи	Аппроксимация граничных условий оператором 1 порядка	Улучшенная аппроксимация граничных условий (2 порядка)
Тестовая задача	$n \geq 1000$	$n \geq 140$
Обычная задача	$n \geq 1000$	$n \geq 140$

Используя теорему о сходимости разностной схемы и свойства порядка сходимости, показанные при проверке программы, можно оценить, что для достижения погрешности  $E_3 = 0,5 * 10^{-6}$  и точности  $E_4 = 0,5 * 10^{-6}$  в модельных задачах с улучшенной аппроксимацией граничных условий нужно брать число разбиений  $n \geq 1400$ . Для модельных задач с аппроксимацией граничных условий оператором 1 порядка нужно брать число разбиений  $n \geq 100000$ .

Таким образом, разностные схемы с улучшенной аппроксимацией граничных условий требует значительно меньших затрат памяти и обладает большей вычислительной скоростью, чем разностные схемы с аппроксимацией граничных условий оператором 1 порядка.

## Приложение 3. Код программы.

### CommonTask.h

```
#pragma once
#include <math.h>
#include <fstream>
#include <iostream>

using namespace std;

class Zadacha {
protected:
    bool active;
public:

    virtual double fa(double x) = 0;
    virtual double fd(double x) = 0;
    virtual double ffi(double x) = 0;

    virtual double IntegFA (double a, double b) = 0;
    virtual double IntegFD (double a, double b) = 0;
    virtual double IntegFFI (double a, double b) = 0;

    virtual void TestClick (int n) = 0;
    virtual void TestVivodvfail (int n) = 0;
};

class TestovayaZadacha: public Zadacha {
public:

    double fa(double x) {
        return 1;
    }
    double fd(double x) {
        return 3;
    }
    double ffi(double x) {
        return 12;
    }

    double IntegFA (double a, double b) {
        double Integral = ((fa(a) + fa(b)) / 2.0)*(b-a);
        return Integral;
    }
    double IntegFD (double a, double b) {
        double Integral = ((fd(a) + fd(b)) / 2.0)*(b-a);
        return Integral;
    }
    double IntegFFI (double a, double b) {
        double Integral = ((ffi(a) + ffi(b)) / 2.0)*(b-a);
        return Integral;
    }

    virtual void TestClick (int n) = 0;
    virtual void TestVivodvfail (int n) = 0;
};
```



```

class OsnovnayaZadacha: public Zadacha {
public:
    double fa(double x)
    {
        return (x*(1-x))+1;
    }

    double fd(double x)
    {
        return (x-0.5)*(x-0.5);
    }

    double ffi(double x)
    {
        return (x*x)*((1-x)*(1-x))*(10);
    }

    double IntegFA (double a, double b) {
        double Integral = ((fa(a) + fa(b)) / 2.0)*(b-a);
        return Integral;
    }
    double IntegFD (double a, double b) {
        double Integral = ((fd(a) + fd(b)) / 2.0)*(b-a);
        return Integral;
    }
    double IntegFFI (double a, double b) {
        double Integral = ((ffi(a) + ffi(b)) / 2.0)*(b-a);
        return Integral;
    }

    virtual void TestClick (int n) = 0;
    virtual void TestVivodvfail (int n) = 0;
};

class PricladnayaZadacha: public Zadacha {
public:
    double fa(double x)
    {
        if (x <= 0.5)
            return 1.38;
        else
            return 3;
    }

    double fd(double x)
    {
        return 0;
    }

    double ffi(double x)
    {
        if ((x <= 0.500001) && (x >= 0.499999))
            //if ((x <= 0.500000) && (x >= 0.500000))
            return 100000;
        else
            return 0;
    }

```

```

    }

    double IntegFA (double a, double b) {
        double Integral = ((fa(a) + fa(b)) / 2.0)*(b-a);
        return Integral;
    }
    double IntegFD (double a, double b) {
        double Integral = ((fd(a) + fd(b)) / 2.0)*(b-a);
        return Integral;
    }
    double IntegFFI (double a, double b) {
        double Integral = ((ffi(a) + ffi(b)) / 2.0)*(b-a);
        return Integral;
    }

    virtual void TestClick (int n) = 0;
    virtual void TestVivodvfail (int n) = 0;

};

```

## TestTaskUsual.h

```

#pragma once
#include <math.h>
#include <fstream>
#include <iostream>
#include "Zadacha.h"
#include <iomanip>
using namespace std;

class TestZadacha: public TestovayaZadacha {
public:
    double MAX;
    double bet1, bet2;
    double h, max[1000];
    double x[1000];
    double xi[1000];
    double mu1, mu2;
    double a, b;
    double *alfa, *d, *fi;
    ofstream fout;
    double *A;
    double *B;
    double *C;
    double *F;
    double Tochn[1000];
    double e;
    double *allfa;
    double y[1000];
    double *betta;

    TestZadacha (double _bet1, double _bet2, double _mu1, double _mu2) {
        MAX = 0;
        bet1 = _bet1, bet2 = _bet2;
        mu1 = _mu1, mu2 = _mu2;
        a = 0, b = 1;
    }

    void TestClick (int n) {

        double *alfa = new double[3*n];
        double *allfa = new double[3*n];
    }

```

```

double *betta = new double[3*n];
double *d = new double[3*n];
double *fi = new double[3*n];
double *A = new double[3*n];
double *B = new double[3*n];
double *C = new double[3*n];
double *F = new double[3*n];

double detA, detB, det, coren, Aa, Bb;
int n1 = 0;
int n2 = n;
coren = sqrt(3);

det = (10 * (-coren * cosh(coren) - 10*sinh(coren))) - (-coren*(-coren *
sinh(coren) - 10*cosh(coren)));
detB = (10 * -210) - (210*(-coren * sinh(coren) - 10*cosh(coren)));
detA = (210 * (-coren * cosh(coren) - 10*sinh(coren))) - (-coren * -210);

Aa = detA/det;
Bb = detB/det;

h = (b - a)/n;

for (int i = n1; i <= n2; i++)
{
    x[i] = a + i*h;
    Tochn[i] = Aa * cosh(sqrt(3)*x[i]) + Bb * sinh(sqrt(3)*x[i]) + 4;
    max[i] = 0;
}

for (int i = n1; i <= n2 - 1; i++) {
    xi[i] = a + (i + 0.5)*h;
}

for (int i = n1+1; i <= n2; i++)
{
    alfa[i] = (1/h) * IntegFA(x[i-1], x[i]);
    alfa[i] = 1/alfa[i];
}

cout << endl;
for (int i = n1+1; i <= n2 - 1; i++)
{
    d[i] = (1/h) * IntegFD(xi[i-1], xi[i]);
}

for (int i = n1+1; i <= n2 - 1; i++)
{
    fi[i] = (1/h) * IntegFFI(xi[i-1], xi[i]);
}

fi[n1] = 0;
d[n1] = 0;

fi[n2] = 0;
d[n2] = 0;

// ----- Метод прогонки ----- //

A[0] = 0;
A[n] = -1/(1+10*h);
C[n] = 0;
B[0] = 1;
B[n] = 1;
C[0] = -1/(1+10*h);

```

```

    for (int i = 1; i <= n - 1; i++)
        A[i] = alfa[i]/(h*h);

    for (int i = 1; i <= n - 1; i++)
        B[i] = -((alfa[i])/(h*h) + (alfa[i+1])/(h*h) + d[i]);

    for (int i = 1; i <= n - 1; i++)
        C[i] = alfa[i+1]/(h*h);

    for(int i = 1; i <= n-1; i++)
        F[i] = -fi[i];

    F[0] = (250*h)/(1+10*h);
    F[n] = (250*h)/(1+10*h);

    allfa[0]=-C[0]/B[0];
    betta[0]=F[0]/B[0];

    for(int i = 1; i <= n - 1; i++)
    {
        e=A[i]*allfa[i-1]+B[i];
        allfa[i]=-C[i]/e;
        betta[i]=(F[i]-A[i]*betta[i-1])/e;
    }

    y[n] = (F[n] - A[n] * betta[n - 1]) / (B[n] + A[n] * allfa[n - 1]);

    for (int i = n - 1; i >= 0; i--)
    {
        y[i] = allfa[i] * y[i + 1] + betta[i];
    }

    max[0] = Tochn[0] - y[0];

    for(int i = 0; i <= n; i++)
    {
        max[i] = Tochn[i] - y[i];
    }

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

    if (abs(max[i]) > MAX)
    {
        MAX = abs(max[i]);
    }

    delete A;
    delete alfa;
    delete d;
    delete fi;
    delete B;
    delete C;
    delete F;
    delete allfa;
    delete betta;
}

void TestVivodvfail (int n) {
    fout.open("ResultTestTaskUsual.txt");
    fout.setf(ios::fixed);

```

```

        fout << setw(8) << "x" << " | " << " U(x)" << " | " << " V(x)" << "
| " << "U(x) - V(x)" << endl;
        for(int i = 0; i <= n; i++)
        {
            fout << x[i] << " | " << y[i] << " | " << Tochn[i] << " | " << max[i]
<< endl;
        }
        fout << fixed;
        fout.precision(15);
        fout << "max |U(x) - V(x)| = " << MAX << " on i=0,n" << endl;
        fout.precision(10);
        fout.close();
    }

};

```

## TestTaskImproved.h

```

#pragma once
#include <math.h>
#include <fstream>
#include <iostream>
#include "Zadacha.h"
#include <iomanip>

using namespace std;

class TestZadachaUluch: public TestovayaZadacha {
public:
    double MAX;
    double bet1, bet2;
    double h, max[1000];
    double x[1000];
    double xi[1000];
    double mu1, mu2;
    double a, b;
    double *alfa, *d, *fi;
    ofstream fout;
    double *A;
    double *B;
    double *C;
    double *F;
    double Tochn[1000];
    double e;
    double *allfa;
    double y[1000];
    double *betta;

    TestZadachaUluch (double _bet1, double _bet2, double _mu1, double _mu2) {
        MAX = 0;
        bet1 = _bet1, bet2 = _bet2;
        mu1 = _mu1, mu2 = _mu2;
        a = 0, b = 1;
    }

    void TestClick (int n) {
        double detA, detB, det, coren, Aa, Bb;
        double *alfa = new double[3*n];
        double *d = new double[3*n];
        double *fi = new double[3*n];
    }

```

```

        double *A = new double[3*n];
        double *B = new double[3*n];
        double *C = new double[3*n];
        double *F = new double[3*n];
        double *allfa = new double[3*n];
        double *betta = new double[3*n];
    coren = sqrt(3);

    det = (10 * (-coren * cosh(coren) - 10*sinh(coren))) - (-coren*(-coren * sinh(coren)
- 10*cosh(coren)));
    detB = (10 * -210) - (210*(-coren * sinh(coren) - 10*cosh(coren)));
    detA = (210 * (-coren * cosh(coren) - 10*sinh(coren))) - (-coren * -210);

    Aa = detA/det;
    Bb = detB/det;

    h = (b - a)/n;

    for (int i = 0; i <= n; i++)
    {
        x[i] = a + i*h;
        Tochn[i] = Aa * cosh(sqrt(3)*x[i]) + Bb * sinh(sqrt(3)*x[i]) + 4;
        max[i] = 0;
    }

    for (int i = 0; i <= n - 1; i++) {
        xi[i] = a + (i + 0.5)*h;
    }

    for (int i = 1; i <= n; i++)
    {
        alfa[i] = (1/h) * IntegFA(x[i-1], x[i]);
        alfa[i] = 1/alfa[i];
    }
    cout << endl;
    for (int i = 1; i <= n - 1; i++)
    {
        d[i] = (1/h) * IntegFD(xi[i-1], xi[i]);
    }

    for (int i = 1; i <= n - 1; i++)
    {
        fi[i] = (1/h) * IntegFFI(xi[i-1], xi[i]);
    }

    fi[0] = 0;
    d[0] = 0;

    fi[n] = 0;
    d[n] = 0;

    // ----- Метод прогонки ----- //

    A[0] = 0;
    A[n] = -1/(1+10*h+(1.5*h*h));
    C[n] = 0;
    B[0] = 1;
    B[n] = 1;
    C[0] = -1/(1+10*h+(1.5*h*h));

    for (int i = 1; i <= n - 1; i++)
        A[i] = alfa[i]/(h*h);

```

```

for (int i = 1; i <= n - 1; i++)
    B[i] = -((alfa[i])/(h*h) + (alfa[i+1])/(h*h) + d[i]);

for (int i = 1; i <= n - 1 ; i++)
    C[i] = alfa[i+1]/(h*h);

for(int i = 1; i <= n-1; i++)
    F[i] = -fi[i];

F[0] = (250*h+6*h*h)/(1+10*h+(1.5*h*h));
F[n] = (250*h+6*h*h)/(1+10*h+(1.5*h*h));

allfa[0]=-C[0]/B[0];
betta[0]=F[0]/B[0];

for(int i = 1; i <= n - 1; i++)
{
    e=A[i]*allfa[i-1]+B[i];
    allfa[i]=-C[i]/e;
    betta[i]=(F[i]-A[i]*betta[i-1])/e;
}

y[n] = (F[n] - A[n] * betta[n - 1]) / (B[n] + A[n] * allfa[n - 1]);

for (int i = n - 1; i >= 0; i--)
{
    y[i] = allfa[i] * y[i + 1] + betta[i];
}

max[0] = Tochn[0] - y[0];

for(int i = 0; i <= n; i++)
{
    max[i] = Tochn[i] - y[i];
}

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

if (abs(max[i]) > MAX)
{
    MAX = abs(max[i]);
}

delete A;
delete alfa;
delete d;
delete fi;
delete B;
delete C;
delete F;
delete allfa;
delete betta;

}

void TestVivodvfail (int n) {
    fout.open("ResultTestTaskImproved.txt");
    fout.setf(ios::fixed);

```

```

        fout << setw(8) << "x" << " | " << " U(x)" << " | " << " V(x)" << "
| " << "U(x) - V(x)" << endl;
        for(int i = 0; i <= n; i++)
        {
            fout << x[i] << " | " << y[i] << " | " << Tochn[i] << " | " << max[i]
<< endl;
        }
        fout << fixed;
        fout.precision(15);
        fout << "max |U(x) - V(x)| = " << MAX << " on i=0,n" << endl;
        fout.precision(10);
        fout.close();
    }

};

```

## MainTaskUsual.h

```

#pragma once
#include <math.h>
#include <fstream>
#include <iostream>
#include "Zadacha.h"
#include <iomanip>
using namespace std;

class OsnZadachaOb: public OsnovnayaZadacha {
public:
    double MAX;
    double bet1, bet2;
    double h, max[1000];
    double x[1000];
    double xi[1000];
    double mu1, mu2;
    double a, b;
    double *alfa, *d, *fi;
    ofstream fout;
    double *A;
    double *B;
    double *C;
    double *F;
    double Tochn[1000];
    double e;
    double *allfa;
    double y[1000];
    double *betta;

    OsnZadachaOb (double _bet1, double _bet2, double _mu1, double _mu2) {
        MAX = 0;
        bet1 = _bet1, bet2 = _bet2;
        mu1 = _mu1, mu2 = _mu2;
        a = 0, b = 1;
    }

    void TestClick (int n) {
        double *alfa = new double[3*n];
        double *d = new double[3*n];
        double *fi = new double[3*n];
        double *A = new double[3*n];
        double *B = new double[3*n];
        double *C = new double[3*n];
        double *F = new double[3*n];
    }

```



```

double *allfa = new double[3*n];
double *betta = new double[3*n];

h = (b - a)/n;

for (int i = 0; i <= n; i++)
{
    x[i] = a + i*h;
    max[i] = 0;
}

for (int i = 0; i <= n - 1; i++) {
    xi[i] = a + (i + 0.5)*h;
}

for (int i = 1; i <= n; i++)
{
    alfa[i] = (1/h) * IntegFA(x[i-1], x[i]);
    alfa[i] = 1/alfa[i];
}

cout << endl;
for (int i = 1; i <= n - 1; i++)
{
    d[i] = (1/h) * IntegFD(xi[i-1], xi[i]);
}

for (int i = 1; i <= n - 1; i++)
{
    fi[i] = (1/h) * IntegFFI(xi[i-1], xi[i]); //
}

fi[0] = 0;
d[0] = 0;

fi[n] = 0;
d[n] = 0;

// ----- Метод прогонки ----- //

A[0] = 0;
A[n] = -1/(1+mu1*h);
C[n] = 0;
B[0] = 1;
B[n] = 1;
C[0] = -1/(1+mu2*h);

for (int i = 1; i <= n - 1; i++)
    A[i] = alfa[i]/(h*h);

for (int i = 1; i <= n - 1; i++)
    B[i] = -((alfa[i])/(h*h) + (alfa[i+1])/(h*h) + d[i]);

for (int i = 1; i <= n - 1; i++)
    C[i] = alfa[i+1]/(h*h);

for(int i = 1; i <= n-1; i++)
    F[i] = -fi[i];

F[0] = (bet1*mu1*h)/(1+mu1*h);
F[n] = (bet2*mu2*h)/(1+mu2*h);

allfa[0]=-C[0]/B[0];
betta[0]=F[0]/B[0];

```

```

for(int i = 1; i <= n - 1; i++)
{
    e=A[i]*allfa[i-1]+B[i];
    allfa[i]=-C[i]/e;
    betta[i]=(F[i]-A[i]*betta[i-1])/e;
}

y[n] = (F[n] - A[n] * betta[n - 1]) / (B[n] + A[n] * allfa[n - 1]);

for (int i = n - 1; i >= 0; i--)
{
    y[i] = allfa[i] * y[i + 1] + betta[i];
}
//

// погрешность
int n1 = n;
n = n * 2;
h = (b - a)/n;

for (int i = 0; i <= n; i++)
{
    x[i] = a + i*h;
    max[i] = 0;
}

for (int i = 0; i <= n - 1; i++) {
    xi[i] = a + (i + 0.5)*h;
}

for (int i = 1; i <= n; i++)
{
    alfa[i] = (1/h) * IntegFA(x[i-1], x[i]);
    alfa[i] = 1/alfa[i];
}

cout << endl;
for (int i = 1; i <= n - 1; i++)
{
    d[i] = (1/h) * IntegFD(xi[i-1], xi[i]);
}

for (int i = 1; i <= n - 1; i++)
{
    fi[i] = (1/h) * IntegFFI(xi[i-1], xi[i]);
}

fi[0] = 0;
d[0] = 0;

fi[n] = 0;
d[n] = 0;

// ----- Метод прогонки ----- //

A[0] = 0;
A[n] = -1/(1+mu1*h);
C[n] = 0;
B[0] = 1;
B[n] = 1;
C[0] = -1/(1+mu2*h);

```

```

for (int i = 1; i <= n - 1; i++)
    A[i] = alfa[i]/(h*h);

for (int i = 1; i <= n - 1; i++)
    B[i] = -((alfa[i])/(h*h) + (alfa[i+1])/(h*h) + d[i]);

for (int i = 1; i <= n - 1; i++)
    C[i] = alfa[i+1]/(h*h);

for(int i = 1; i <= n-1; i++)
    F[i] = -fi[i];

F[0] = (bet1*mu1*h)/(1+mu1*h);
F[n] = (bet2*mu2*h)/(1+mu2*h);

allfa[0]=-C[0]/B[0];
betta[0]=F[0]/B[0];

for(int i = 1; i <= n - 1; i++)
{
    e=A[i]*allfa[i-1]+B[i];
    allfa[i]=-C[i]/e;
    betta[i]=(F[i]-A[i]*betta[i-1])/e;
}

Tochn[n] = (F[n] - A[n] * betta[n - 1]) / (B[n] + A[n] * allfa[n - 1]);

for (int i = n - 1; i >= 0; i--)
{
    Tochn[i] = allfa[i] * Tochn[i + 1] + betta[i];
}
//
// конец погрешности

// MAX
max[0] = Tochn[0] - y[0];

n = n1;
h = (b - a)/n;
for (int i = 0; i <= n; i++)
    x[i] = a + i*h;

for(int i = 0; i <= n; i++)
{
    max[i] = Tochn[i*2] - y[i];
}

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

if (abs(max[i]) > MAX)
{
    MAX = abs(max[i]);
}

delete A;
delete alfa;
delete d;
delete fi;
delete B;

```

```

        delete C;
        delete F;
        delete allfa;
        delete betta;
    }

    void TestVivodvfail (int n) {
        fout.open("ResultMainTaskUsual.txt");
        fout.setf(ios::fixed);
        fout << setw(8) << "x" << " | " << " U(x)" << " | " << " U(2x)" << "
| " << "U(x) - U(2x)" << endl;
        for(int i = 0; i <= n; i++) {
            fout << x[i] << " | " << y[i] << " | " << Tochn[i*2] << " | " <<
max[i] << endl;
        }
        fout << fixed;
        fout.precision(15);
        fout << "max |U(x) - U(2x)| = " << MAX << " on i=0,n" << endl;
        fout.precision(10);
        fout.close();
    }
};

```

## MainTaskImproved.h

```

#pragma once
#include <math.h>
#include <fstream>
#include <iostream>
#include "Zadacha.h"
#include <iomanip>
using namespace std;

class OsnZadachaU1: public OsnovnayaZadacha {
public:
    double MAX;
    double bet1, bet2;
    double h, max[1000];
    double x[1000];
    double xi[1000];
    double mu1, mu2;
    double a, b;
    double *alfa, *d, *fi;
    ofstream fout;
    double *A;
    double *B;
    double *C;
    double *F;
    double Tochn[1000];
    double e;
    double *allfa;
    double y[1000];
    double *betta;

    OsnZadachaU1 (double _bet1, double _bet2, double _mu1, double _mu2) {
        MAX = 0;
        bet1 = _bet1, bet2 = _bet2;
        mu1 = _mu1, mu2 = _mu2;
        a = 0, b = 1;
    }

    void TestClick (int n) {

```

```

double *alfa = new double[3*n];
double *d = new double[3*n];
double *fi = new double[3*n];
double *A = new double[3*n];
double *B = new double[3*n];
double *C = new double[3*n];
double *F = new double[3*n];
double *allfa = new double[3*n];
double *betta = new double[3*n];

h = (b - a)/n;

for (int i = 0; i <= n; i++)
{
    x[i] = a + i*h;
    max[i] = 0;
}

for (int i = 0; i <= n - 1; i++) {
    xi[i] = a + (i + 0.5)*h;
}

for (int i = 1; i <= n; i++)
{
    alfa[i] = (1/h) * IntegFA(x[i-1], x[i]);
    alfa[i] = 1/alfa[i];
}

cout << endl;
for (int i = 1; i <= n - 1; i++)
{
    d[i] = (1/h) * IntegFD(xi[i-1], xi[i]);
}

for (int i = 1; i <= n - 1; i++)
{
    fi[i] = (1/h) * IntegFFI(xi[i-1], xi[i]); //
}

fi[0] = (2/h) * IntegFFI(x[0], xi[0]);
d[0] = (2/h) * IntegFD(x[0], xi[0]);

fi[n] = (2/h) * IntegFFI(xi[n-1], x[n]);
d[n] = (2/h) * IntegFD(xi[n-1], x[n]);

// ----- Метод прогонки ----- //

A[0] = 0;
A[n] = -alfa[1]/(mu1*h+alfa[1] + (h*h)/2*d[0]);
C[n] = 0;
B[0] = 1;
B[n] = 1;
C[0] = -alfa[n]/(mu2*h+alfa[n] + (h*h)/2*d[n]);

for (int i = 1; i <= n - 1; i++)
    A[i] = alfa[i]/(h*h);

for (int i = 1; i <= n - 1; i++)
    B[i] = -((alfa[i])/(h*h) + (alfa[i+1])/(h*h) + d[i]);

for (int i = 1; i <= n - 1; i++)
    C[i] = alfa[i+1]/(h*h);

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

```

```

F[i] = -fi[i];

F[0] = (fi[0] * (h*h)/2 + bet1*mu1*h)/(mu1*h + alfa[1] + (h*h)/2 * d[0]);
F[n] = (fi[n] * (h*h)/2 + bet2*mu2*h)/(mu2*h + alfa[n] + (h*h)/2 * d[n]);

allfa[0]=-C[0]/B[0];
betta[0]=F[0]/B[0];

for(int i = 1; i <= n - 1; i++)
{
    e=A[i]*allfa[i-1]+B[i];
    allfa[i]=-C[i]/e;
    betta[i]=(F[i]-A[i]*betta[i-1])/e;
}

y[n] = (F[n] - A[n] * betta[n - 1]) / (B[n] + A[n] * allfa[n - 1]);

for (int i = n - 1; i >= 0; i--)
{
    y[i] = allfa[i] * y[i + 1] + betta[i];
}
//
// погрешность
int n1 = n;
n = n * 2;
h = (b - a)/n;

for (int i = 0; i <= n; i++)
{
    x[i] = a + i*h;
    max[i] = 0;
}

for (int i = 0; i <= n - 1; i++) {
    xi[i] = a + (i + 0.5)*h;
}

for (int i = 1; i <= n; i++)
{
    alfa[i] = (1/h) * IntegFA(x[i-1], x[i]);
    alfa[i] = 1/alfa[i];
}

for (int i = 1; i <= n - 1; i++)
{
    d[i] = (1/h) * IntegFD(xi[i-1], xi[i]);
}

for (int i = 1; i <= n - 1; i++)
{
    fi[i] = (1/h) * IntegFFI(xi[i-1], xi[i]);
}

fi[0] = (2/h) * IntegFFI(x[0], xi[0]);
d[0] = (2/h) * IntegFD(x[0], xi[0]);

fi[n] = (2/h) * IntegFFI(xi[n-1], x[n]);
d[n] = (2/h) * IntegFD(xi[n-1], x[n]);

// ----- Метод прогонки ----- //

```

```

A[0] = 0;
A[n] = -alfa[1]/(mu1*h+alfa[1] + (h*h)/2*d[0]);
C[n] = 0;
B[0] = 1;
B[n] = 1;
C[0] = -alfa[n]/(mu2*h+alfa[n] + (h*h)/2*d[n]);

for (int i = 1; i <= n - 1; i++)
    A[i] = alfa[i]/(h*h);

for (int i = 1; i <= n - 1; i++)
    B[i] = -((alfa[i])/(h*h) + (alfa[i+1])/(h*h) + d[i]);

for (int i = 1; i <= n - 1 ; i++)
    C[i] = alfa[i+1]/(h*h);

for(int i = 1; i <= n-1; i++)
    F[i] = -fi[i];

F[0] = (fi[0] * (h*h)/2 + mu1*bet1*h)/(mu1*h + alfa[1] + (h*h)/2 * d[0]);
F[n] = (fi[n] * (h*h)/2 + mu2*bet2*h)/(mu2*h + alfa[n] + (h*h)/2 * d[n]);

allfa[0]=-C[0]/B[0];
betta[0]=F[0]/B[0];

for(int i = 1; i <= n - 1; i++)
{
    e=A[i]*allfa[i-1]+B[i];
    allfa[i]=-C[i]/e;
    betta[i]=(F[i]-A[i]*betta[i-1])/e;
}

Tochn[n] = (F[n] - A[n] * betta[n - 1]) / (B[n] + A[n] * allfa[n - 1]);

for (int i = n - 1; i >= 0; i--)
{
    Tochn[i] = allfa[i] * Tochn[i + 1] + betta[i];
}
//
// конец погрешности

// MAX
max[0] = Tochn[0] - y[0];

n = n1;
h = (b - a)/n;
for (int i = 0; i <= n; i++)
    x[i] = a + i*h;

for(int i = 0; i <= n; i++)
{
    max[i] = Tochn[i*2] - y[i];
}

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

```

```

        if (abs(max[i]) > MAX)
        {
            MAX = abs(max[i]);
        }
        // Конец MAX
        delete A;
        delete alfa;
        delete d;
        delete fi;
        delete B;
        delete C;
        delete F;
        delete allfa;
        delete betta;
    }

    void TestVivodvfail (int n) {

        fout.open("ResultMainTaskImproved.txt");
        fout.setf(ios::fixed);
        fout << setw(8) << "x" << " | " << " U(x)" << " | " << " U(2x)" << "
| " << "U(x) - U(2x)" << endl;
        for(int i = 0; i <= n; i++)
        {
            fout << x[i] << " | " << y[i] << " | " << Tochn[i*2] << " | " <<
max[i] << endl;
        }
        fout << fixed;
        fout.precision(15);
        fout << "max |U(x) - U(2x)| = " << MAX << " on i=0,n" << endl;
        fout.precision(10);
        fout.close();
    }

};

```

## FirstPhysicalTask.h

```

#pragma once
#include <math.h>
#include <fstream>
#include <iostream>
#include "Zadacha.h"
#include <iomanip>

class PricladZadacha: public PricladnayaZadacha {
public:
    double MAX;
    double h, max[1000];
    double x[1000];
    double xi[1000];
    double mu1, mu2;
    double alfa[1000], d[0100], fi[1000];
    ofstream fout;
    double A[1000];
    double B[1000];
    double C[1000];
    double F[1000];
    double Tochn[1000];
    double e;
    double allfa[1000];
    double y[1000];

```



```

double betta[1000];
double l1, l2, psi, Ff, eps1, eps2, KRT, k1, k2;
int n1, n2, n3, n4;

PricladZadacha (double _mu1, double _mu2, double _l1, double _l2, double _KRT, double
_eps2, double _eps1, double _Ff, double _psi, double _k1, double _k2, int _n1, int _n2, int
_n3, int _n4) {
    MAX = 0;
    mu1 = _mu1, mu2 = _mu2;
    l1 = _l1, l2 = _l2;
    eps1 = _eps1, eps2 = _eps2;
    Ff = _Ff, KRT = _KRT;
    psi = _psi;
    k1 = _k1, k2 = _k2;
    n1 = _n1;
    n2 = _n2;
    n3 = _n3;
    n4 = _n4;
}

void TestVivodvfail(int n) {
    fout.open("ResultPhysicalTaskFirst.txt");
    fout.setf(ios::fixed);
    //fout << setw(8) << "x" << " | " << " U(x)" << " | " << " V(x)" << "
| " << "U(x) - V(x)" << endl;
    fout << setw(8) << "x" << " | " << " U(x)" << " | " << endl;
    for (int i = 0; i <= n1+n2+n3+n4; i++)
    {
        //fout << x[i] << " | " << y[i] << " | " << Tochn[i] << " | " << max[i]
<< endl;
        fout << x[i] << " | " << y[i] << " | " << endl;
    }
    fout << fixed;
    fout.precision(15);
    //fout << "max |U(x) - V(x)| = " << MAX << " on i=0,n" << endl;
    fout.precision(10);
    fout.close();
}

void TestClick (int n) {

double l = l1 + l2;
double a = psi - l1;
double b = psi + l2;
double c1 = psi - eps1;
double c2 = psi + eps2;

double f1 = (KRT * Ff)/(2*eps1);
double f2 = (KRT * Ff)/(2*eps2);

double h1 = (l1 - eps1)/n1;
double h2 = (eps1)/n2;
double h3 = (eps2)/n3;
double h4 = (l2 - eps2)/n4;

int n_left = n1;
int n_centr = n1 + n2;
int n_right = n1 + n2 + n3;
int n_s = n1 + n2 + n3 + n4;
int N_Total = n_s + 1;

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

```

```

{
    //Считаем координаты x
    if (i == 0)
        x[i] = a;
    if ((i >= 1) && (i <= n_left - 1))
        x[i] = x[i-1] + h1;
    if (i == n_left)
        x[i] = c1;
    if ((i >= n_left + 1) && (i <= n_centр - 1))
        x[i] = x[i-1] + h2;
    if (i == n_centр)
        x[i] = psi;
    if ((i >= n_centр + 1) && (i <= n_right - 1))
        x[i] = x[i-1] + h3;
    if (i == n_right)
        x[i] = c2;
    if ((i >= n_right + 1) && (i <= n_s))
        x[i] = x[i-1] + h4;
    if (i == N_Total)
        x[i] = b;
}

```

```

cout.setf(ios::fixed);
cout << fixed;
cout.precision(15);
for (int i = 0; i <= n_s; i++)
    cout << "x [" << i << "]" << x[i] << endl;

```

```

cout << "h1 = " << h1 << endl;
cout << "h2 = " << h2 << endl;
cout << "h3 = " << h3 << endl;
cout << "h4 = " << h4 << endl;

```

```

cout << "f1 = " << f1 << endl;
cout << "f2 = " << f2 << endl;

```

```

A[0] = 0;
A[n_s] = 0;
C[n_s] = 0;
B[0] = 1;
B[n_s] = 1;
C[0] = 0;

```

```

    for (int i = 1; i <= n_s - 1; i++)
    {
        if ((i >= 1) && (i <= n_left - 1))
            A[i] = k1/(h1*h1);
        if (i == n_left)
            A[i] = k1/((h2+h1)*0.5*h1);
        if ((i >= n_left + 1) && (i <= n_centр - 1))
            A[i] = k1/(h2*h2);
        if (i == n_centр)
            A[i] = k1/((h2+h3)*0.5*h2);
        if ((i >= n_centр + 1) && (i <= n_right - 1))
            A[i] = k2/(h3*h3);
        if (i == n_right)
            A[i] = k2/((h4+h3)*0.5*h3);
        if ((i >= n_right + 1) && (i <= n_s - 1))
            A[i] = k2/(h4*h4);
    }

```

```

    for (int i = 0; i <= n_s - 1; i++)

```

```

{
    cout << "A [" << i << "] = " << A[i] << endl;
}

for (int i = 1; i <= n_s; i++)
{
    if ((i >= 1) && (i <= n_left - 1))
        C[i] = k1/(h1*h1);
    if (i == n_left)
        C[i] = k1/((h2+h1)*0.5*h2);
    if ((i >= n_left + 1) && (i <= n_centr - 1))
        C[i] = k1/(h2*h2);
    if (i == n_centr)
        C[i] = k2/((h2+h3)*0.5*h3);
    if ((i >= n_centr + 1) && (i <= n_right - 1))
        C[i] = k2/(h3*h3);
    if (i == n_right)
        C[i] = k2/((h4+h3)*0.5*h4);
    if ((i >= n_right + 1) && (i <= n_s - 1))
        C[i] = k2/(h4*h4);
}

for (int i = 0; i <= n_s; i++)
{
    cout << "B [" << i << "] = " << B[i] << endl;
}

for (int i = 1; i <= n_s ; i++)
{
    if ((i >= 1) && (i <= n_left - 1))
        B[i] = -(k1+k1)/(h1*h1);
    if (i == n_left)
        B[i] = -( (k1/((h2+h1)*0.5*h1)) + ( k1/( (h2+h1) *0.5*h2) ) );
    if ((i >= n_left + 1) && (i <= n_centr - 1))
        B[i] = -(k1+k1)/(h2*h2);
    if (i == n_centr)
        B[i] = -( (k1/((h2+h3)*0.5*h2)) + ( k2/( (h2+h3) *0.5*h3) ) );
    if ((i >= n_centr + 1) && (i <= n_right - 1))
        B[i] = -(k2+k2)/(h3*h3);
    if (i == n_right)
        B[i] = -( (k2/((h3+h4)*0.5*h3)) + ( k2/( (h3+h4) *0.5*h4) ) );
    if ((i >= n_right + 1) && (i <= n_s - 1))
        B[i] = -(k2+k2)/(h4*h4);
}

for (int i = 0; i <= n_s; i++)
{
    cout << "C [" << i << "] = " << C[i] << endl;
}

for(int i = 1; i <= n_s; i++)
{
    if ((i >= 1) && (i <= n_left - 1))
        F[i] = 0;
    if (i == n_left)
        F[i] = -(h2*f1)/(h2+h1);
    if ((i >= n_left + 1) && (i <= n_centr - 1))
        F[i] = -f1;
    if (i == n_centr)
        F[i] = -(h2*f1+h3*f2)/(h3+h2);
    if ((i >= n_centr + 1) && (i <= n_right - 1))
        F[i] = -f2;
    if (i == n_right)
        F[i] = -(h3*f2)/(h3+h4);
}

```

```

        if ((i >= n_right + 1) && (i <= n_s - 1))
            F[i] = 0;
    }

    F[0] = mu1;
    F[n_s] = mu2;

    for (int i = 0; i <= n_s; i++)
    {
        cout << "F [" << i << "]" << F[i] << endl;
    }

    allfa[0] = -C[0]/B[0];
    betta[0] = F[0]/B[0];

    for(int i = 1; i <= n_s - 1; i++)
    {
        e = A[i]*allfa[i-1] + B[i];
        allfa[i] = -C[i]/e;
        betta[i] = (F[i] - A[i]*betta[i-1])/e;
    }

    y[n_s] = (F[n_s] - A[n_s] * betta[n_s - 1]) / (B[n_s] + A[n_s] * allfa[n_s -
1]);

    for (int i = n_s - 1; i >= 0; i--)
    {
        y[i] = allfa[i] * y[i + 1] + betta[i];
    }

    for (int i = 0; i <= n_s; i++)
    {
        cout << "y [" << i << "]" << y[i] << endl;
    }
}

};

```

### ThirdPhysicalTask.h

```

#pragma once
#include <math.h>
#include <fstream>
#include <iostream>
#include "Zadacha.h"
#include <iomanip>

class PricladZadachaThree: public PricladnayaZadacha {
public:
    double MAX;
    double h, max[1000];
    double x[1000];
    double xi[1000];
    double mu1, mu2;
    double alfa[1000], d[0100], fi[1000];
    ofstream fout;
    double A[1000];
    double B[1000];
    double C[1000];
    double F[1000];

```

```

double Tochn[1000];
double e;
double allfa[1000];
double y[1000];
double betta[1000];
double l1, l2, psi, Ff, eps1, eps2, KRT, k1, k2, bet1, bet2;
int n1, n2, n3, n4;

PricladZadachaThree (double _mu1, double _mu2, double _l1, double _l2, double _KRT,
double _eps2, double _eps1, double _Ff, double _psi, double _k1, double _k2, int _n1, int
_n2, int _n3, int _n4, double _bet1, double _bet2) {
    MAX = 0;
    mu1 = _mu1, mu2 = _mu2;
    l1 = _l1, l2 = _l2;
    eps1 = _eps1, eps2 = _eps2;
    Ff = _Ff, KRT = _KRT;
    psi = _psi;
    k1 = _k1, k2 = _k2;
    n1 = _n1;
    n2 = _n2;
    n3 = _n3;
    n4 = _n4;
    bet1 = _bet1;
    bet2 = _bet2;
}

void TestVivodvfail(int n) {
    fout.open("ResultPhysicalTaskThird.txt");
    fout.setf(ios::fixed);
    //fout << setw(8) << "x" << " | " << " U(x)" << " | " << " V(x)" << "
| " << "U(x) - V(x)" << endl;
    fout << setw(8) << "x" << " | " << " U(x)" << " | " << endl;
    for (int i = 0; i <= n1+n2+n3+n4; i++)
    {
        //fout << x[i] << " | " << y[i] << " | " << Tochn[i] << " | " << max[i]
<< endl;
        fout << x[i] << " | " << y[i] << " | " << endl;
    }
    fout << fixed;
    fout.precision(15);
    //fout << "max |U(x) - V(x)| = " << MAX << " on i=0,n" << endl;
    fout.precision(10);
    fout.close();
}

void TestClick (int n) {
    double l = l1 + l2;
    double a = psi - l1;
    double b = psi + l2;
    double c1 = psi - eps1;
    double c2 = psi + eps2;

    double f1 = (KRT * Ff)/(2*eps1);
    double f2 = (KRT * Ff)/(2*eps2);

    double h1 = (l1 - eps1)/n1;
    double h2 = (eps1)/n2;
    double h3 = (eps2)/n3;
    double h4 = (l2 - eps2)/n4;

    int n_left = n1;
    int n_centra = n1 + n2;

```

```

int n_right = n1 + n2 + n3;
int n_s = n1 + n2 + n3 + n4;
int N_Total = n_s + 1;

for (int i = 0; i <= n_s; i++)
{
    //Считаем координаты x
    if (i == 0)
        x[i] = a;
    if ((i >= 1) && (i <= n_left - 1))
        x[i] = x[i-1] + h1;
    if (i == n_left)
        x[i] = c1;
    if ((i >= n_left + 1) && (i <= n_centр - 1))
        x[i] = x[i-1] + h2;
    if (i == n_centр)
        x[i] = psi;
    if ((i >= n_centр + 1) && (i <= n_right - 1))
        x[i] = x[i-1] + h3;
    if (i == n_right)
        x[i] = c2;
    if ((i >= n_right + 1) && (i <= n_s))
        x[i] = x[i-1] + h4;
    if (i == N_Total)
        x[i] = b;
}

```

```

A[0] = 0;
A[n_s] = -k1/(mu1*h1+k1);
C[n_s] = 0;
B[0] = 1;
B[n_s] = 1;
C[0] = -k2/(mu2*h4+k2);

```

```

for (int i = 1; i <= n_s - 1; i++)
{
    if ((i >= 1) && (i <= n_left - 1))
        A[i] = k1/(h1*h1);
    if (i == n_left)
        A[i] = k1/((h2+h1)*0.5*h1);
    if ((i >= n_left + 1) && (i <= n_centр - 1))
        A[i] = k1/(h2*h2);
    if (i == n_centр)
        A[i] = k1/((h2+h3)*0.5*h2);
    if ((i >= n_centр + 1) && (i <= n_right - 1))
        A[i] = k2/(h3*h3);
    if (i == n_right)
        A[i] = k2/((h4+h3)*0.5*h3);
    if ((i >= n_right + 1) && (i <= n_s - 1))
        A[i] = k2/(h4*h4);
}

```

```

for (int i = 1; i <= n_s; i++)
{
    if ((i >= 1) && (i <= n_left - 1))
        C[i] = k1/(h1*h1);
    if (i == n_left)
        C[i] = k1/((h2+h1)*0.5*h2);
    if ((i >= n_left + 1) && (i <= n_centр - 1))
        C[i] = k1/(h2*h2);
    if (i == n_centр)
        C[i] = k2/((h2+h3)*0.5*h3);
}

```

```

        if ((i >= n_centra + 1) && (i <= n_right - 1))
            C[i] = k2/(h3*h3);
        if (i == n_right)
            C[i] = k2/((h4+h3)*0.5*h4);
        if ((i >= n_right + 1) && (i <= n_s - 1))
            C[i] = k2/(h4*h4);
    }

    for (int i = 1; i <= n_s ; i++)
    {
        if ((i >= 1) && (i <= n_left - 1))
            B[i] = -(k1+k1)/(h1*h1);
        if (i == n_left)
            B[i] = -( (k1/((h2+h1)*0.5*h1)) + ( k1/( (h2+h1) *0.5*h2) ) );
        if ((i >= n_left + 1) && (i <= n_centra - 1))
            B[i] = -(k1+k1)/(h2*h2);
        if (i == n_centra)
            B[i] = -( (k1/((h2+h3)*0.5*h2)) + ( k2/( (h2+h3) *0.5*h3) ) );
        if ((i >= n_centra + 1) && (i <= n_right - 1))
            B[i] = -(k2+k2)/(h3*h3);
        if (i == n_right)
            B[i] = -( (k2/((h3+h4)*0.5*h3)) + ( k2/( (h3+h4) *0.5*h4) ) );
        if ((i >= n_right + 1) && (i <= n_s - 1))
            B[i] = -(k2+k2)/(h4*h4);
    }

    for(int i = 1; i <= n_s; i++)
    {
        if ((i >= 1) && (i <= n_left - 1))
            F[i] = 0;
        if (i == n_left)
            F[i] = -(h2*f1)/(h2+h1);
        if ((i >= n_left + 1) && (i <= n_centra - 1))
            F[i] = -f1;
        if (i == n_centra)
            F[i] = -(h2*f1+h3*f2)/(h3+h2);
        if ((i >= n_centra + 1) && (i <= n_right - 1))
            F[i] = -f2;
        if (i == n_right)
            F[i] = -(h3*f2)/(h3+h4);
        if ((i >= n_right + 1) && (i <= n_s - 1))
            F[i] = 0;
    }

    F[0] = (mu1*bet1*h1)/(mu1*h1 + k1);
    F[n_s] = (mu2*bet2*h4)/(mu2*h4 + k2);

    allfa[0]=-C[0]/B[0];
    betta[0]=F[0]/B[0];

    for(int i = 1; i <= n_s - 1; i++)
    {
        e=A[i]*allfa[i-1]+B[i];
        allfa[i]=-C[i]/e;
        betta[i]=(F[i]-A[i]*betta[i-1])/e;
    }

    y[n_s] = (F[n_s] - A[n_s] * betta[n_s - 1]) / (B[n_s] + A[n_s] * allfa[n_s -
1]);

    for (int i = n_s - 1; i >= 0; i--)
    {

```

```
        y[i] = allfa[i] * y[i + 1] + betta[i];
    }
};
```