Полное решение задачи теплопроводности

Исходное уравнение и обезразмеривание

Рассмотрим двумерное уравнение теплопроводности:

$$\frac{\partial T}{\partial t} = a\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

Введем характерные величины:

- L - характерная длина
- T_0 - характерная температура
- t_0 - характерное время

Определим безразмерные величины:

$$T^* = \frac{T}{T_0}, \quad t^* = \frac{t}{t_0}, \quad x^* = \frac{x}{L}, \quad y^* = \frac{y}{L}$$

Подставим в исходное уравнение:

$$\frac{\partial (T_0 T^*)}{\partial (t_0 t^*)} = a\left(\frac{\partial^2 (T_0 T^*)}{\partial (L x^*)^2} + \frac{\partial^2 (T_0 T^*)}{\partial (L y^*)^2}\right)$$

Упростим:

$$\frac{T_0}{t_0}\frac{\partial T^*}{\partial t^*} = a\frac{T_0}{L^2}\left(\frac{\partial^2 T^*}{\partial (x^*)^2} + \frac{\partial^2 T^*}{\partial (y^*)^2}\right)$$

Разделим обе части на $\frac{T_0}{t_0}$:

$$\frac{\partial T^*}{\partial t^*} = \frac{at_0}{L^2}\left(\frac{\partial^2 T^*}{\partial (x^*)^2} + \frac{\partial^2 T^*}{\partial (y^*)^2}\right)$$

Возьмем $t_0 = \frac{L^2}{a}$, тогда:

$$\frac{\partial T^*}{\partial t^*} = \frac{\partial^2 T^*}{\partial (x^*)^2} + \frac{\partial^2 T^*}{\partial (y^*)^2}$$

Тогда получаем итоговое обезразмеренное уравнение теплопроводности:

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}$$

где все переменные T, t, x, y являются безразмерными.

Граничные и начальные условия

Рассмотрим это уравнение со следующими граничными и начальными условиями:

Начальное условие:

$$T(x, y, 0) = 0, \quad x, y \in (0, 1)$$

Граничные условия (при $t \geq 0$):

По координате $x \in [0, 1]$:

$$T(x, 0, t) = 0.4, \quad T(x, 1, t) = 0.1$$

По координате $y \in [0, 1]$:

$$T(0, y, t) = 0.2, \quad T(1, y, t) = 0$$

Разностная схема

Разобьем область $[0, 1] \times [0, 1]$ на N равных частей по каждой координате, так что $N_x = N_y = N$. Шаги сетки:

$$h_x = h_y = h = \frac{1}{N}$$

Временной шаг обозначим через τ.

Запишем полностью неявную разностную схему:

$$\frac{T_{i,j}^{k+1} - T_{i,j}^{k}}{\tau} = \frac{T_{i+1,j}^{k+1} - 2T_{i,j}^{k+1} + T_{i-1,j}^{k+1}}{h^2} + \frac{T_{i,j+1}^{k+1} - 2T_{i,j}^{k+1} + T_{i,j-1}^{k+1}}{h^2}$$

где:

- $i = 0, 1, \ldots, N$ - индекс по координате x
- $j = 0, 1, \ldots, N$ - индекс по координате y
- k - индекс временного слоя

Метод Гаусса-Зейделя

Для решения СЛАУ, возникающей на каждом временном слое, применяем метод Гаусса-Зейделя. Представим матрицу системы в виде суммы:

$$A = L + D + U$$

где D - диагональная, L - нижняя треугольная, U - верхняя треугольная матрицы.

Тогда итерационный процесс метода Гаусса-Зейделя записывается в виде:

$$(L + D)T^{k+1} = b - UT^k$$

где используются уже вычисленные компоненты нового приближения T^{k+1} на текущей итерации.

Алгоритм метода Гаусса-Зейделя на C++

```
// Параметры сетки
double h_x = 1.0 / n;          // шаг по x
double h_y = 1.0 / m;          // шаг по y
double tay = 1.0 / time_steps; // шаг по времени

// Коэффициенты разностной схемы
double c_x = -1.0 / (h_x * h_x);  // коэффициент для соседей по x
double c_y = -1.0 / (h_y * h_y);  // коэффициент для соседей по y
double d = (1.0 / tay) + 2.0 * (1.0 / (h_x * h_x) + 1.0 / (h_y * h_y));
```

```
// Итерационный процесс Гаусса-Зейделя
for (int j = 1; j < m; j++) {          // по строкам (y)
    for (int i = 1; i < n; i++) {      // по столбцам (x)
        int k = (i - 1) + (j - 1) * (n - 1);  // линейный индекс

        // Правая часть уравнения
        double sum = /* правая часть */;

        // Учет соседних узлов
        if (i > 1) sum -= c_x * u[k - 1];         // левый сосед
        if (i < n - 1) sum -= c_x * u[k + 1];     // правый сосед
        if (j > 1) sum -= c_y * u[k - (n - 1)];  // нижний сосед
        if (j < m - 1) sum -= c_y * u[k + (n - 1)];  // верхний сосед

        // Обновление значения
        double new_val = sum / d;
        u[k] = new_val;
    }
}
```

Результаты расчета

На Рисунке 1 представлено стационарное распределение температуры $T(x, y)$, полученное в результате численного решения уравнения

теплопроводности методом Гаусса-Зейделя.

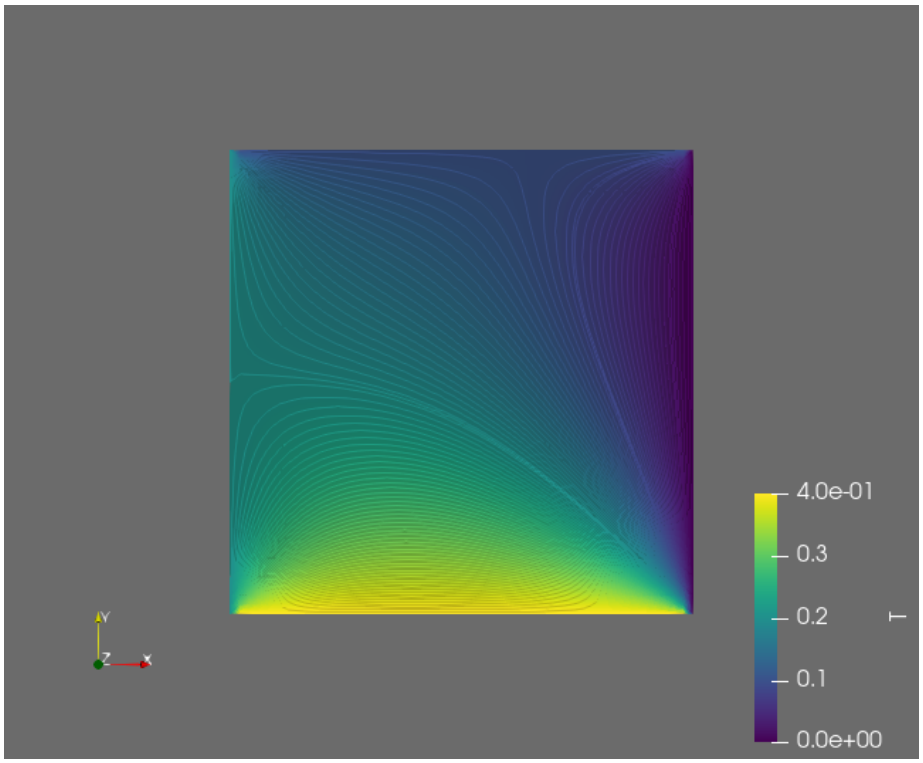

Рис. 1: Стационарное распределение температуры $T(x, y)$, полученное методом Гаусса-Зейделя

Использование библиотеки AMGCL

Общее описание

Для сравнения эффективности методов решения СЛАУ дополнительно применяется библиотека AMGCL, реализующая алгебраические многосеточные методы. Матрица системы хранится в формате

CRS (Compressed Row Storage), что позволяет эффективно работать с разреженными матрицами, характерными для разностных схем. Библиотека AMGCL обеспечивает высокую скорость сходимости за счет использования многосеточного подхода и оптимизирована для решения крупных разреженных систем уравнений.

Формирование матрицы в формате CRS

Для хранения разреженной матрицы системы используется формат CRS (Compressed Row Storage). Функция заполнения матрицы реализована следующим образом:

```
void Temp(int n, int m, std::vector<size_t>& addr,
          std::vector<size_t>& cols, std::vector<double>& vals) {
    size_t n2 = n * n;
    double h = 1.0 / (double)(n - 1);
    double tau = 1.0 / (double)(m - 1);
    addr.clear();
    addr.reserve(n2 + 1);
    addr.push_back(0);

    cols.clear();
    cols.reserve(n2 * 5);
    vals.clear();
    vals.reserve(n2 * 5);
```

```
    for (int j = 0, k = 0; j < n; ++j) {
        for (int i = 0; i < n; ++i, ++k) {
            if (i == 0 || j == 0 || i == n - 1 || j == n - 1) {
                cols.push_back(k);
                vals.push_back(1);
            } else {
                cols.push_back(k - n);
                vals.push_back(-1.0 / (h * h));
                cols.push_back(k - 1);
                vals.push_back(-1.0 / (h * h));
                cols.push_back(k);
                vals.push_back(1.0 / tau + 4.0 / (h * h));
                cols.push_back(k + 1);
                vals.push_back(-1.0 / (h * h));
                cols.push_back(k + n);
                vals.push_back(-1.0 / (h * h));
            }
            addr.push_back(cols.size());
        }
    }
}
```

Данная функция формирует матрицу системы, где граничные узлы обрабатываются отдельно, а для внутренних узлов задается пятидиагональная структура, соответствующая разностной схеме для уравнения теплопроводности.

Структура решателя AMGCL

Для работы с библиотекой AMGCL реализован класс-обёртка, предоставляющий интерфейс для решения СЛАУ:

```
#ifndef AMGCL_SOLVER_HPP
#define AMGCL_SOLVER_HPP

#include "CsrMatrix.hpp"
#include <map>
#include <memory>
#include <string>

class AmgclSolver {
public:
    AmgclSolver(size_t maxit = 1000, double eps = 1e-8);
    ~AmgclSolver();
    AmgclSolver(size_t maxit, double tolerance,
                std::initializer_list<std::pair<std::string, std::string
    void solve(const std::vector<double>& rhs, std::vector<double>& ret)
    void set_matrix(const CsrMatrix& mat);

private:
    size_t maxit_;
    double tolerance_;
    std::map<std::string, std::string> params_;
```

```
    struct Impl;
    std::unique_ptr<Impl> pimpl_;
};

#endif
```

Класс инкапсулирует настройки решателя (максимальное количество итераций, точность) и параметры AMGCL, используя идиому Pimpl для сокрытия деталей реализации библиотеки.

Визуализация результатов AMGCL

На Рисунке 2 представлено распределение температуры $T(x, y)$ в установившемся режиме, полученное в результате численного решения с использованием библиотеки AMGCL.

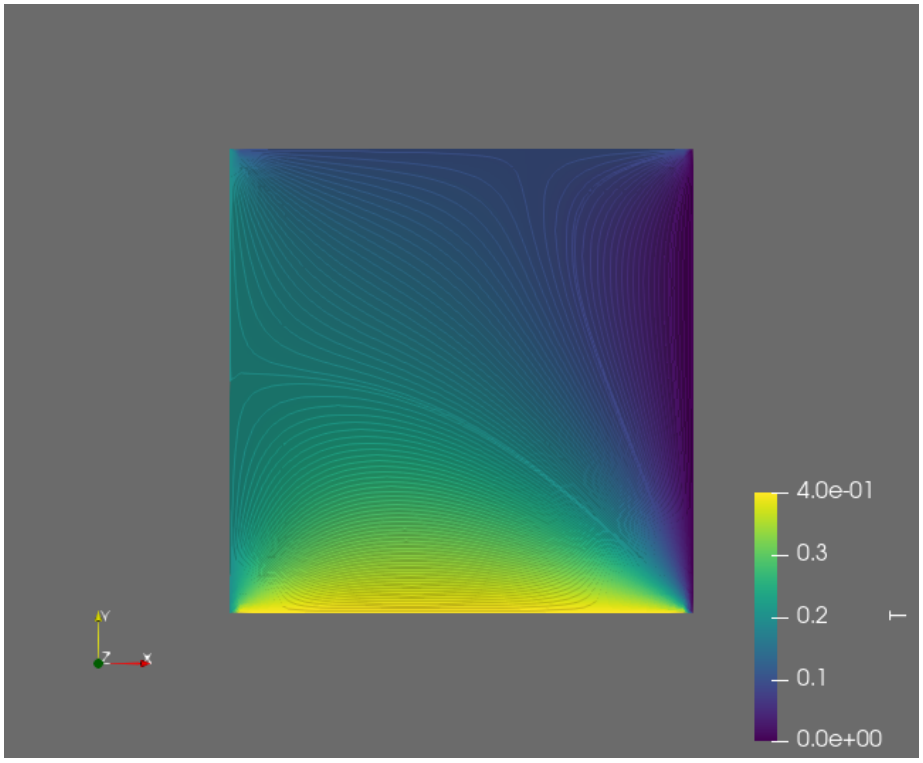

Рис. 2: Распределение температуры $T(x, y)$, полученное с использованием AMGCL

Сравнение скорости сходимости методов

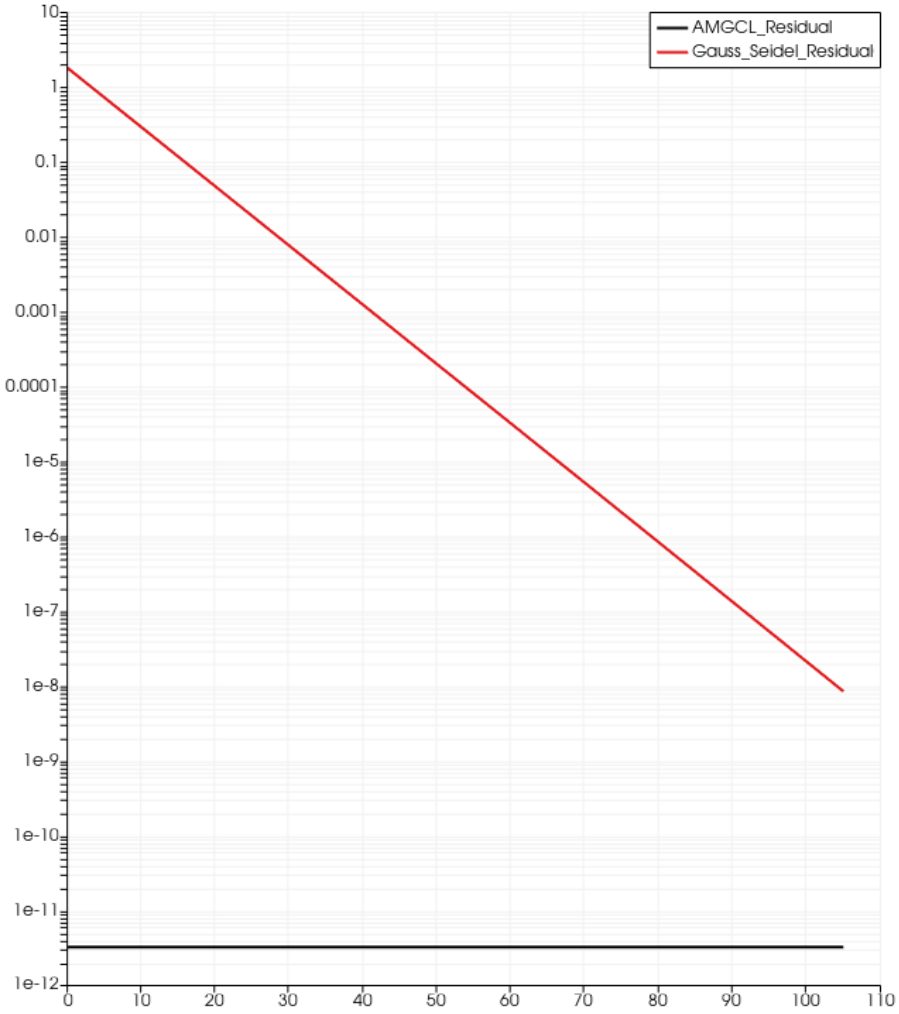

Рис. 3: Сравнение скорости сходимости методов Гаусса-Зейделя и AMGCL

Сравнение решателей и релаксаторов

Сравнив различные виды решателей и релаксаторов для задачи с размером сетки $N = 75$, были получены следующие результаты по времени решения (в миллисекундах):

Решатель	**Релаксатор**	**Время (мс)**
richardson	ilut	492
gmres	iluk	494
fgmres	iluk	501
gmres	damped_jacobi	502
fgmres	damped_jacobi	510
idrs	damped_jacobi	510
bicgstab	ilut	516
idrs	ilut	517
idrs	iluk	525
idrs	gauss_seidel	529
bicgstab	damped_jacobi	532
richardson	gauss_seidel	533
fgmres	gauss_seidel	535
richardson	ilu0	535
richardson	iluk	535
idrs	ilup	536
fgmres	ilu0	537
idrs	ilu0	537
fgmres	ilup	538
richardson	ilup	538
idrs	spai0	539
fgmres	ilut	540
idrs	spai1	540
richardson	chebyshev	540
fgmres	spai1	541
fgmres	chebyshev	541
idrs	chebyshev	543
gmres	ilut	544
richardson	spai0	544
richardson	spai1	545
gmres	ilup	546
gmres	spai0	550
gmres	spai1	553
gmres	ilu0	556
gmres	gauss_seidel	559
fgmres	spai0	559
bicgstab	spai0	567
bicgstab	iluk	568

Модификация задачи с условиями Неймана

Математическая постановка

Рассмотрим модифицированную задачу теплопроводности с теплоизолированными боковыми стенками, что соответствует граничным условиям Неймана.

Уравнение теплопроводности остается без изменений:

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}$$

Начальное условие:

$$T(x, y, 0) = 0, \quad x, y \in (0, 1)$$

Граничные условия (при $t \geq 0$):

- На верхней и нижней границах - условия Дирихле:

$$T(x, 0, t) = 0.4, \quad T(x, 1, t) = 0.1$$

- На боковых границах - условия Неймана (теплоизоляция):

$$\frac{\partial T}{\partial x}(0, y, t) = 0, \quad \frac{\partial T}{\partial x}(1, y, t) = 0$$

Реализация в коде

В программной реализации условия Неймана на боковых границах задаются через разностные соотношения:

```
else if (i == 0) {
    // Условие Неймана на левой стенке: dT/dx = 0
    col.push_back(k);
    val.push_back(-1.0);
    col.push_back(k + 1);
    val.push_back(1.0);
} else if (i == n - 1) {
    // Условие Неймана на правой стенке: dT/dx = 0
    col.push_back(k - 1);
    val.push_back(-1.0);
    col.push_back(k);
    val.push_back(1.0);
}
```

Визуализация решения и анализ распределения температуры

Визуализация стационарного распределения

На Рисунке 3 представлено стационарное распределение температуры $T(x, y)$ для задачи с теплоизолированными боковыми стенками.

Особенностью данного распределения является его независимость от координаты x в установившемся режиме, что демонстрирует вырождение двумерной задачи в одномерную.

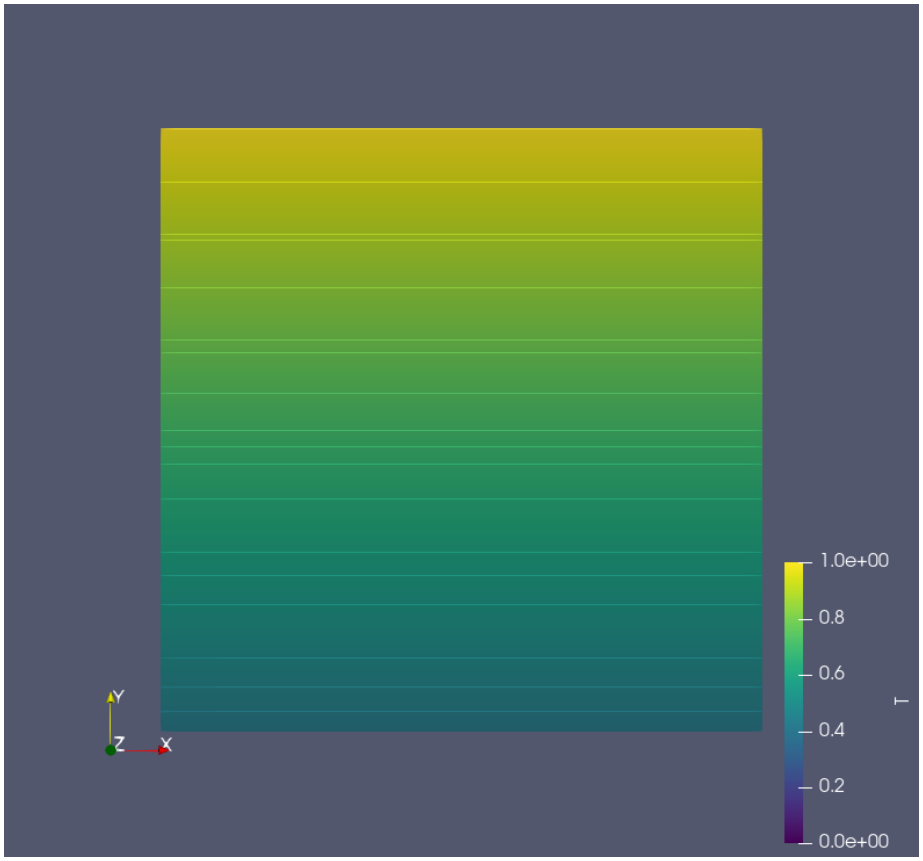

Рис. 4: Стационарное распределение температуры для задачи с условиями Неймана на боковых границах

Анализ с помощью Plot Over Line

Для количественного анализа проведем сечения распределения температуры по линиям $y = \text{const}$. На Рисунке 4 представлены графики

зависимости температуры от координаты x на различных уровнях y.

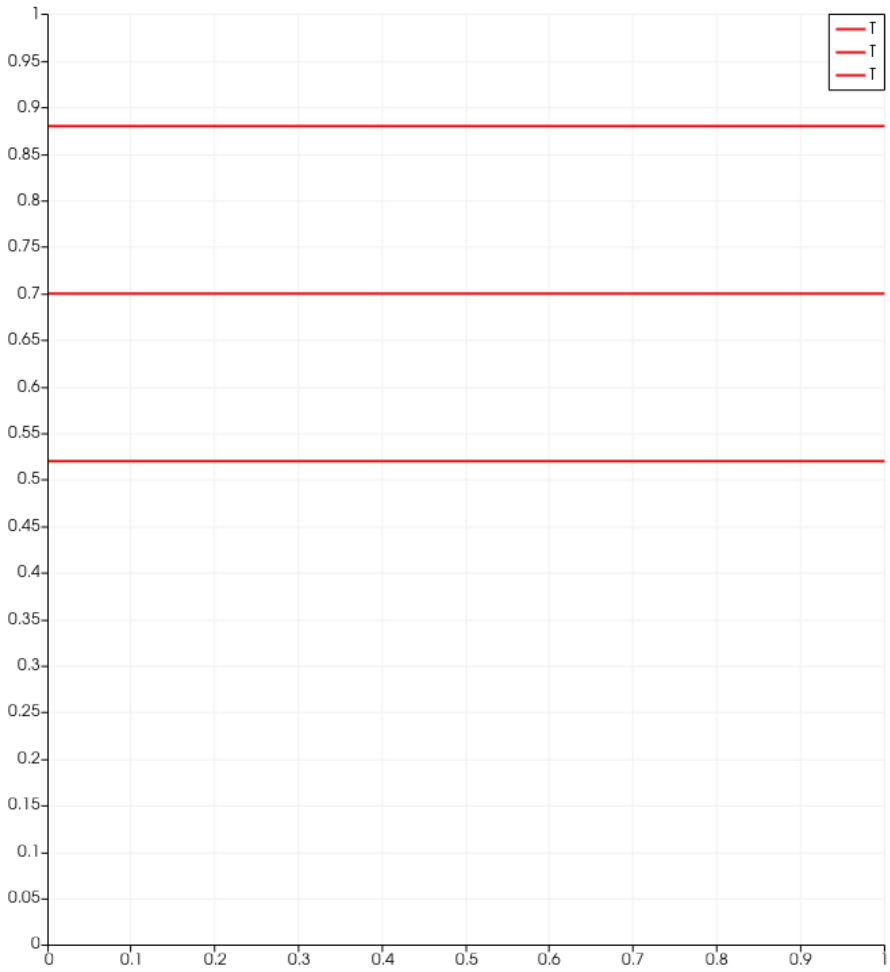

Рис. 5: Распределение температуры по координате x на различных уровнях y

Сведение к одномерной задаче

Как следует из анализа графиков, распределение температуры практически не зависит от координаты x, что подтверждает возможность сведения исходной двумерной задачи к одномерной. В установившемся режиме уравнение теплопроводности принимает вид:

$$\frac{d^2T}{dy^2} = 0$$

с граничными условиями:

$$T(0) = 0.4, \quad T(1) = 0.1$$

Теплоизоляция боковых стенок приводит к установлению одномерного температурного поля, изменяющегося только по вертикальной координате, что полностью согласуется с результатами численного моделирования, представленными на графиках.