МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | Факультет прикладной
математики и информатики

Кафедра прикладной математики

Курсовой проект

по дисциплине «Численные методы»

Группа ПМ-92 АРТЮХОВ РОМАН

Вариант 22

Преподаватели СОЛОВЕЙЧИК ЮРИЙ ГРИГОРЬЕВИЧ

ПАТРУШЕВ ИЛЬЯ ИГОРЕВИЧ

Новосибирск, 2021

Задача (вариант 22):

МКЭ для двумерной краевой задачи для эллиптического уравнения в декартовой системе координат. Базисные функции линейные на треугольниках. Краевые условия всех типов. Коэффициент диффузии λ разложить по квадратичным базисным функциям. Матрицу СЛАУ генерировать в разреженном строчном формате. Для решения СЛАУ использовать МСГ или ЛОС с неполной факторизацией.

1. Постановка задачи

Эллиптическая краевая задача для функции u определяется дифференциальным уравнением:

$$-div(\lambda \cdot gradu) + \gamma u = f \ ,$$

заданным в некоторой области Ω с границей $S = S_1 \cup S_2 \cup S_3$ и краевыми условиями:

$$u\Big|_{S_1} = u_g$$

$$\lambda \frac{\partial u}{\partial n}\Big|_{S_2} = \theta$$

$$\lambda \frac{\partial u}{\partial n}\Big|_{S_3} + \beta\left(u\Big|_{S_3} - u_\beta\right) = 0$$

В декартовой системе координат это уравнение может быть записано в виде:

$$-\frac{\partial}{\partial x}\left(\lambda \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\lambda \frac{\partial u}{\partial y}\right) + \gamma u = f$$, в которых

$u\Big|_{S_i}$ - значение искомой функции μ на границе S_i

$\frac{\partial u}{\partial n}\Big|_{S_i}$ - значение на S_i производной функции u по направлению внешней нормали к поверхности $S_i, i = 1,2,3.$

λ - коэффициент диффузии.

2. Теоретическая часть

a. Вариационная постановка в форме уравнения Галеркин

В операторной форме исходное уравнение можно переписать в форме $Lu = f$, где L - оператор, действующий в Гильбертовом пространстве H Нам нужно найти приближение к элементу u, соответствующее заданному элементу f.

Потребуем, чтобы невязка $\psi_i\Big|_{S_1} = 0$ дифференциального уравнения была ортогональна некоторому пространству Φ функций u, которое мы будем называть пространством пробных функций, т.е.

$$\int_\Omega \left(-div\left(\lambda \cdot gradu_n\right) + \gamma u_n\right)\psi_i d\Omega = \int_\Omega f\psi_i d\Omega$$

Применяя формулу Грина (интегрирование по частям для многомерного случая), перепишем уравнение в виде:

$$\int_{\Omega} \lambda \cdot gradu_n \cdot grad\psi_i d\Omega + \int_{\Omega} \gamma u_n \psi_i d\Omega - \int_{S} \lambda \frac{\partial u}{\partial n} \psi_i dS = \int_{\Omega} f \psi_i d\Omega$$

Учитывая $S = S_1 \cup S_2 \cup S_3$: $\int_{S} \lambda \frac{\partial u}{\partial n} \psi_i dS = \int_{S_1} \lambda \frac{\partial u}{\partial n} \psi_i dS + \int_{S_2} \lambda \frac{\partial u}{\partial n} \psi_i dS + \int_{S_3} \lambda \frac{\partial u}{\partial n} \psi_i dS$

Теперь учтем заданные краевые условия.

Поскольку $\psi_i \Big|_{S_1} = 0$ значит интеграл $\int_{S_1} \lambda \frac{\partial u}{\partial n} \psi_i dS = 0$,то интегральное соотношение примет вид:

$$\int_{\Omega} \lambda \cdot gradu_n \cdot grad\psi_i d\Omega + \int_{\Omega} \gamma u_n \psi_i d\Omega - \int_{S_2} \theta \psi_i dS + \int_{S_3} \beta \left(u - u_\beta \right) \psi_i dS = \int_{\Omega} f \psi_i d\Omega$$

Исходя из того, что $u_n = \sum_{i=1}^{n} q_i \psi_i$ перепишем уравнение в виде:

$$\sum_{j=1}^{n} q_j \int_{\Omega} \lambda \cdot grad\psi_j \cdot grad\psi_i d\Omega + \sum_{j=1}^{n} q_j \int_{\Omega} \gamma \psi_j \psi_i d\Omega + \sum_{j=1}^{n} q_j \int_{S_3} \beta \psi_j \psi_i dS =$$

$$= \int_{\Omega} f \psi_i d\Omega + \int_{S_2} \theta \psi_i dS + \int_{S_3} \beta u_\beta \psi_i dS$$

Исходная задача рассматривается в декартовой системе координат, то

$$gradu = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right)$$

$$gradu \cdot grad\upsilon = \frac{\partial u}{\partial x} \frac{\partial \upsilon}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial \upsilon}{\partial y}$$

Отсюда получаем уравнение в виде:

$$\sum_{j=1}^{n} q_j \int_{\Omega} \lambda \cdot \left(\frac{\partial \psi_j}{\partial x} \frac{\partial \psi_i}{\partial x} + \frac{\partial \psi_j}{\partial y} \frac{\partial \psi_i}{\partial y} \right) dxdy + \sum_{j=1}^{n} q_j \int_{\Omega} \gamma \psi_j \psi_i dxdy + \sum_{j=1}^{n} q_j \int_{S_3} \beta \psi_j \psi_i dxdy$$

$$= \int_{\Omega} f \psi_i dxdy + \int_{S_2} \theta \psi_i dxdy + \int_{S_3} \beta u_\beta \psi_i dxdy$$

b. Конечноэлементная дискретизация

На каждом конечном элементе Ω_k - треугольнике эти функции будут совпадать с функциями $L_1(x,y), L_2(x,y), L_3(x,y)$, такими, что

$$L_i(x,y) = 1 в\ вершине (x_i, y_i) и\ нулю\ во\ всех\ остальных\ i = \overline{1,3}$$

Любая линейная на Ω_k, функция представима в виде линейной комбинации этих базисных линейных функций, коэффициентами будут значения функции в каждой из вершин треугольника Ω_k. Таким образом, на каждом конечном элементе нам понадобятся 3 узла – вершины треугольника.

Получаем:

$$\psi_1 = L_1(x,y)$$
$$\psi_2 = L_2(x,y)$$
$$\psi_3 = L_3(x,y)$$

При вычислении интегралов от произведений вида $L_i L_j$ по треугольнику Ω_k или по любому его ребру Γ можно использовать формулы:

$$\int_{\Omega_k} (L_1)^{\upsilon_1} (L_2)^{\upsilon_2} (L_3)^{\upsilon_3} d\Omega_k = \frac{\upsilon_1!\upsilon_2!\upsilon_3!}{(\upsilon_1+\upsilon_2+\upsilon_3+2)!} 2mes\Omega_k$$

$$\int_{\Gamma} (L_i)^{\upsilon_i} (L_j)^{\upsilon_j} dS = \frac{\upsilon_i!\upsilon_j!}{(\upsilon_i+\upsilon_j+1)!} mes\Gamma, \quad i \neq j$$

где $mes\Omega_k = \frac{1}{2}|\det D|$ — это площадь треугольника

$D = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix}$ - матрица координат его вершин.

Учитывая построение L - функций, получаем следующие соотношения:

$$\begin{cases} L_1 + L_2 + L_3 = 1 \\ L_1 x_1 + L_2 x_2 + L_3 x_3 = x \\ L_1 y_1 + L_2 y_2 + L_3 y_3 = y \end{cases}$$

Получаем систему:

$$\begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \cdot \begin{pmatrix} L_1 \\ L_2 \\ L_3 \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

Находим коэффициенты линейных функций

$$L_i = a_0^i + a_1^i x + a_2^i y, \quad i = \overline{1,3}$$

$$\begin{pmatrix} L_1 \\ L_2 \\ L_3 \end{pmatrix} = \begin{pmatrix} a_0^1 + a_1^1 + a_2^1 \\ a_0^2 + a_1^2 + a_2^2 \\ a_0^3 + a_1^3 + a_2^3 \end{pmatrix} = D^{-1} f = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

$$D^{-1} = \frac{1}{|\det D|} \begin{pmatrix} x_2 y_3 - x_3 y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3 y_1 - x_1 y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1 y_2 - x_2 y_1 & y_1 - y_2 & x_2 - x_1 \end{pmatrix}$$

c. Переход к локальным матрицам

Чтобы получить выражения для локальных матриц жёсткости G и массы M каждого конечного элемента Ω_k, перейдем к решению локальной задачи на каждом конечном элементе. Полученное уравнение для области Ω представим в виде суммы интегралов по областям Ω_k без учёта краевых условий.

$$\int_{\Omega_k} \lambda \cdot \left(\frac{\partial \psi_j}{\partial x} \frac{\partial \psi_i}{\partial x} + \frac{\partial \psi_j}{\partial y} \frac{\partial \psi_i}{\partial y} \right) dxdy + \int_{\Omega_k} \gamma \psi_j \psi_i dxdy = \int_{\Omega_k} f \psi_i dxdy$$

Локальная матрица будет представлять собой сумму матриц жёсткости и массы и будет иметь размерность 3x3.

i. Построение матрицы жёсткости

Рассмотрим первый член в вышеуказанном выражении для k-го конечного элемента:

$$\int_{\Omega_k} \lambda \cdot \left(\frac{\partial \psi_j}{\partial x} \frac{\partial \psi_i}{\partial x} + \frac{\partial \psi_j}{\partial y} \frac{\partial \psi_i}{\partial y} \right) dxdy$$

Учитывая, что $\psi_j = L_j$, $\psi_i = L_i$, получаем:

$$G_{ij} = \lambda \left(a_1^i a_1^j + a_2^i a_2^j \right) \frac{|\det D|}{2}, \quad i, j = \overline{0,2}$$

В поставленной задаче требуется разложить λ по квадратичным базисным функциям: $\lambda = \sum_{t=0}^{5} \lambda_t \varphi_t$, где λ_t – значение коэффициента λ в соответствующих узлах, φ_t - квадратичные базисные функции, которые определяются следующим образом:

$$\varphi_0 = L_1\left(2L_1 - 1\right)$$
$$\varphi_1 = L_2\left(2L_2 - 1\right)$$
$$\varphi_2 = L_3\left(2L_3 - 1\right)$$
$$\varphi_3 = 4L_1L_2$$
$$\varphi_4 = 4L_2L_3$$
$$\varphi_5 = 4L_1L_3$$

Таким образом, $G_{ij} = \sum_{t=0}^{5} \lambda_t \left(\alpha_1^i \alpha_1^j + \alpha_2^i \alpha_2^j\right) \int_{\Omega_k} \varphi_t d\Omega_k, \quad i, j = \overline{0,2}$

Интегралы от базисных функций равны:

$$\int_{\Omega_k} \varphi_0 d\Omega_k = \int_{\Omega_k} \varphi_1 d\Omega_k = \int_{\Omega_k} \varphi_2 d\Omega_k = \int_{\Omega_k} 2L_i^2 - L_i d\Omega_k = 0$$

$$\int_{\Omega_k} \varphi_3 d\Omega_k = \int_{\Omega_k} \varphi_4 d\Omega_k = \int_{\Omega_k} \varphi_5 d\Omega_k = \int_{\Omega_k} 4L_iL_j d\Omega_k = \frac{1}{3} mes\Omega_k = \frac{1}{6}\left|\det D\right|$$

Учитывая интегралы получим:

$$G_{ij} = \frac{\alpha_1^i \alpha_1^j + \alpha_2^i \alpha_2^j}{6}\left|\det D\right| \sum_{t=3}^{5} \lambda_t, \quad i, j = \overline{0,2}$$

Где $\sum_{t=3}^{5} \lambda_t$ - сумма значений коэффициента на серединах трех сторон конечного элемента.

ii. Построение матрицы массы

Рассмотрим второй член в выражении для k-го конечного элемента:

$$\int_{\Omega_k} \gamma \psi_j \psi_i dxdy$$

Учитывая, что $\psi_j = L_j$, $\psi_i = L_i$, получаем:

$$\begin{cases} M_{ij} = \gamma \int_{\Omega_k} L_i L_j d\Omega_k = \gamma \frac{1}{12} mes\Omega_k = \gamma \frac{1}{24}\left|\det D\right|, \ i \neq j \\ M_{ij} = 2 \cdot \left(\gamma \frac{1}{24}\left|\det D\right|\right), \quad i = j \end{cases}$$

iii. Построение вектора правой части

Рассмотрим правую часть выражения для k-го конечного элемента:

$$\int_{\Omega_k} f \psi_i dxdy$$

f можно представить в виде: $f = f_1 L_1 + f_2 L_2 + f_3 L_3$

f_i - значение в вершинах треугольника.

$\psi_i = L_i$

Получим:

$$\int_{\Omega_k} f_m L_m L_i = f_m \int_{\Omega_k} L_m L_i d\Omega_k$$

$$F_i = \sum_{m=1}^{3} f_m \int_{\Omega_k} L_m L_i d\Omega_k = \sum_{m=1}^{3} f_m \frac{1}{12} mes\Omega_k = \sum_{m=1}^{3} f_m \frac{|\det D|}{24}, \quad i = \overline{0,2}$$

iv. Сборка глобальной матрицы и глобального вектора правой части.

В качестве базисных функций ψ_i берутся финитные функции, отличные от нуля лишь на нескольких конечных элементах. Поэтому большинство интегралов будут равны нулю. Ненулевыми интегралы $\int_{\Omega_l} \lambda \cdot grad\psi_j \cdot grad\psi_i d\Omega$, $\int_{\Omega_l} \gamma \psi_j \psi_i d\Omega$, $\int_{\Omega_l} f \psi_i d\Omega$, будут в том случае, если базисные функции ψ_i и ψ_j являются ненулевыми на конечном элементе Ω_l.

$\Omega_l \ (l = 1,...,L)$, L – количество конечных элементов.

При формировании глобальной матрицы из локальных, полученные суммированием матриц жесткости и массы, учитываем соответствие локальной и глобальной нумерации каждого конечного элемента. Зная глобальные номера узлов конечного элемента, определяем и то, какие элементы глобальной матрицы изменятся при учете текущего конечного элемента. Аналогично с правой частью. При учете текущего локального вектора изменяется те элементы глобального вектора правой части, номера которых совпадают с глобальными номерами узлов, присутствующих в этом конечном элементе.

d. Учет краевых условий

i. Учет первых краевых условий

Для учета первых краевых условий, в глобальной матрице и глобальном векторе находим соответствующую глобальному номеру краевого узла строку, и ставим на этой строке вместо диагонального элемента единицу, а вместо элемента с таким номером в вектор правой части – значение краевого условия.

ii. Учет вторых и третьих краевых условий

Рассмотрим краевые условия второго и третьего рода

$$\lambda \frac{\partial u}{\partial n}\bigg|_{S_2} = \theta$$

$$\lambda \frac{\partial u}{\partial n}\bigg|_{S_3} + \beta\left(u\big|_{S_3} - u_\beta\right) = 0$$

Отсюда получаем, что для учета краевых условий необходимо вычислить интегралы:

$$I_1 = \int_{S_2} \theta \psi_i dxdy$$

$$I_2 = \int_{S_3} \beta u_\beta \psi_i dxdy$$

$$I_3 = \int_{S_3} \beta \psi_j \psi_i dxdy$$

Краевые условия второго и третьего рода задаются на ребрах, т.е. определяются двумя узлами, лежащими на ребре.

Параметр β на S_3 будем считать постоянным

Параметр μ_β будем раскладывать по двум базисным функциям, определенным на этом ребре.

$$u_\beta = u_{\beta 0}\varphi_0 + u_{\beta 1}\varphi_1$$

φ_i - линейные базисные функции, которые имеют также свои глобальные номера во всей расчетной области

$u_{\beta i}$ - значения функции u_β в узлах ребра.

Функцию θ раскладываем аналогичным образом.

Тогда интегралы примет вид: ..

$$I_1 = \int_{S_2} \left(\theta_0\varphi_0 + \theta_1\varphi_1\right)\varphi_i dxdy$$

$$I_2 = \int_{S_3} \beta\left(u_{\beta 0}\varphi_0 + u_{\beta 1}\varphi_1\right)\varphi_i dxdy$$

$$I_3 = \int_{S_3} \beta\varphi_i\varphi_j dxdy$$

Для учета вклада вторых и третьих краевых условий рассчитываются две матрицы размерностью 2×2

Интегралы считаем по ребру, следовательно вычислять будем по формуле:

$$\int_\Gamma \left(L_i\right)^{\upsilon_i}\left(L_j\right)^{\upsilon_j} dS = \frac{\upsilon_i!\upsilon_j!}{\left(\upsilon_i+\upsilon_j+1\right)!} mes\Gamma, \quad i \neq j$$

$mes\Gamma$ - длина ребра, $mes\Gamma = \sqrt{\left(x_2-x_1\right)^2 + \left(y_2-y_1\right)^2}$

Интегралы, посчитанные по приведенным формулам, будут равны:

$$I_1 = \begin{pmatrix} \int\limits_{S_2} L_1^2 dxdy & \int\limits_{S_2} L_1 L_2 dxdy \\ \int\limits_{S_2} L_2 L_1 dxdy & \int\limits_{S_2} L_2^2 dxdy \end{pmatrix} \cdot \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \frac{1}{6} mesS_2 \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

$$I_2 = \beta \frac{1}{6} mesS_3 \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} u_{\beta 0} \\ u_{\beta 1} \end{pmatrix}$$

$$I_3 = \beta \frac{1}{6} mesS_3 \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

Добавляя эту матрицу в левую часть, на места, соответствующие номерам узлов, получаем учет третьих краевых условий.

При расчете θ и $\beta\left(u|_{S_3} - u_\beta\right)$ должно учитываться направление нормали

$$\lambda \frac{\partial u}{\partial m}\bigg|_S = \lambda \cdot gradu \cdot \vec{n}$$

Если рассматривать нормаль к наклонной стороне области, то для каждой из двух точек ребра, в которых рассматриваются нормали, значения производных решения по обеим координатам будет ненулевыми, если, производная самой функции по какой-либо координате не будет нулевой.

Helper.cs

```
namespace Project
{
    public struct Matrix
    {
        public int N;
        public int maxIter;
        public double epsilon;
        public uint[] ig, jg;
        public double[] di, gg, pr, x, absolut_x;

        //* Умножение матрицы на вектор
        public double[] mult(double[] x) {
            var y = new double[x.Length];

            for (uint i = 0, jj = 0; i < x.Length; i++) {
                y[i] = di[i] * x[i];

                for (uint j = ig[i]; j < ig[i + 1]; j++, jj++) {
                   y[i]      += gg[jj] * x[jg[jj]];
                   y[jg[jj]] += gg[jj] * x[i];
                }
            }
            return y;
        }
    }

    public static class Helper
    {
        //* Сумма матриц
        public static double[][] SummMatrix(double[][] Mat1, double[][] Mat2)
        {
            var Mat3 = new double[3][];
            for (uint i = 0; i < 3; i++) Mat3[i] = new double[3];

            for (uint i = 0; i < Mat1.Length; i++)
                for (uint j = 0; j < Mat1.Length; j++)
                    Mat3[i][j] = Mat1[i][j] + Mat2[i][j];
            return Mat3;
        }

        //* Середина ребра (между двумя узлами)
        public static double[] MidPoints(double[] point1, double[] point2)
        {
            var midpoint = new double[2];
            midpoint[0] = (point1[0] + point2[0]) / 2.0;
            midpoint[1] = (point1[1] + point2[1]) / 2.0;
            return midpoint;
        }

        //* Вычисление скалярного произведение
        public static double scalar(double[] array1, double[] array2) {
            double res = 0;
            for (uint i = 0; i < array1.Length; i++)
                res += array1[i] * array2[i];
            return res;
        }
    }
}
```

Data.cs

```
using static System.Diagnostics.Debug;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Project
{
    public class Data
    {

    // ************ Данные задачи ************ //
        protected (double gamma, double betta)[] materials ; /// Значения гамма и бетта

        protected uint       countNode     ; /// Количество узлов
        protected uint       countFinitEl  ; /// Количество конечных элементов
        protected uint       countAreas    ; /// Количество областей
        protected uint       countKrayCond ; /// Количество краевых условий

        protected double[][]  nodes         ; /// Координаты узлов
        protected uint[][]    finitElements ; /// Координаты конечных элементов
        protected uint[]      areaFinitEl   ; /// номера областей в которых расположен кон. эл. по порядку
        protected uint[][]    boards        ; /// Условия на границах
        protected uint[]      nodesArea     ; /// Область для каждого узла

        //? Для составления списка связанностей
        public uint[]         numNodes      ; /// Номера узлов

    // ************ Конструктор DATA (чтение данных) ************ //
         public Data(string Path) { Assert(Read(Path), "Несуществующий файл или неккоректные данные!"); }

        //* Чтение данных
        private bool Read(string path)
        {
            bool isCorr = true;

            isCorr &= readParams(path + "param.txt");

            resize(); // Выделение памяти

            isCorr &= readfiles    (path + "xy.txt",      ref nodes                        );
            isCorr &= readMaterial (path + "area.txt",    ref materials, ref areaFinitEl   );
            isCorr &= readfiles    (path + "board.txt",   ref boards                       );
            isCorr &= readNumNodes (path + "finitel.txt", ref numNodes, ref finitElements  );
            isCorr &= readNodesArea(path + "XY_area.txt", ref nodesArea                    );

            // Отсортировать массив board по номеру краевого условия
            boards = boards.OrderByDescending(l => l[3]).ToArray();

            return isCorr;
        }

        private bool readfiles(string path, ref double[][] array)
        {
            if (!File.Exists(path)) return false;
            string[] lines = File.ReadAllLines(path);

            for (uint i = 0; i < lines.Length; i++) {
                var values = lines[i].Split(" ").Select(n => double.Parse(n)).ToArray();
                for (uint j = 0; j < values.Length; j++)
                    array[i][j] = values[j];
            }
            return true;
```

```
    }

    private bool readfiles(string path, ref uint[][] array)
    {
        if (!File.Exists(path)) return false;
        string[] lines = File.ReadAllLines(path);

        for (uint i = 0; i < lines.Length; i++) {
            var values = lines[i].Split(" ").Select(n => uint.Parse(n)).ToArray();
            for (uint j = 0; j < values.Length; j++)
                array[i][j] = values[j];
        }
        return true;
    }

    private bool readNodesArea(string path, ref uint[] array)
    {
        if (!File.Exists(path)) return false;
        array = File.ReadAllText(path).Split(" ").Select(n => uint.Parse(n)).ToArray();
        return true;
    }

    private bool readNumNodes(string path, ref uint[] numNodes, ref uint[][] finEl)
    {
        if (!File.Exists(path)) return false;
        string[] lines = File.ReadAllLines(path);

        List<uint> listTemp = new List<uint>();

        for (int i = 0; i < lines.Length; i++)
        {
            var values = lines[i].Split(" ").Select(n => uint.Parse(n)).ToArray();
            for (int j = 0; j < values.Length; j++) {
                if (!listTemp.Contains(values[j]))
                    listTemp.Add(values[j]);
                finEl[i][j] = values[j];
            }
        }
        // Заполнение массива для списка связанностей
        numNodes = listTemp.OrderBy(n => ((uint)n)).ToArray();
        return true;
    }

    private bool readMaterial(string path, ref (double gamma, double betta)[] material, ref uint[] area)
    {
        if (!File.Exists(path)) return false;
        StreamReader file = new StreamReader(path);
        for (int i = 0; i < countAreas; i++) {
            double[] line = file.ReadLine().Split(" ").Select(n => double.Parse(n)).ToArray();
            material[i] = (line[0], line[1]);
        }
        areaFinitEl = file.ReadLine().Split(" ").Select(n => uint.Parse(n)).ToArray();
        file.Close();
        return true;
    }

    private bool readParams(string path)
    {
        if (!File.Exists(path)) return false;
        bool isCorr = true;
        StreamReader file = new StreamReader(path);
        string[] param = file.ReadLine().Split(" ");

        isCorr &= uint.TryParse(param[0], out countNode);
        isCorr &= uint.TryParse(param[1], out countFinitEl);
        isCorr &= uint.TryParse(param[2], out countAreas);
        isCorr &= uint.TryParse(param[3], out countKrayCond);
```

```
            file.Close();
            return isCorr;
        }

        private void resize()
        {
            nodes         = new double           [countNode]    [];
            finitElements = new uint             [countFinitEl] [];
            materials     = new (double, double) [countAreas]     ;
            areaFinitEl   = new uint             [countFinitEl]   ;
            boards        = new uint             [countKrayCond][];
            numNodes      = new uint             [countNode]      ;
            for (uint i = 0; i < countNode    ; i++) nodes[i]         = new double[2];
            for (uint i = 0; i < countFinitEl ; i++) finitElements[i] = new uint[3];
            for (uint i = 0; i < countKrayCond; i++) boards[i]        = new uint[5];
        }

        //* Данные задачи
        public string dataAll()
        {
            System.Console.OutputEncoding = Encoding.GetEncoding(65001);
            string margin = String.Join("", Enumerable.Repeat("-", 50));
            StringBuilder output = new StringBuilder();

            output.Append($"{margin}\n");
            output.Append($"(Количество узлов) countNode                = {countNode}    \n" +
                          $"(Количество конечных элементов) countFinitEl = {countFinitEl} \n" +
                          $"(Количество областей) countAreas             = {countAreas}   \n" +
                          $"(Количество граничных условий)               = {countKrayCond}\n");

            output.Append($"{margin}\n");
            output.Append("(Координаты узлов) nodes: \n");
            for (int i = 0; i < countNode; i++)
                output.Append($"({nodes[i][0]}, " +
                              $"{nodes[i][1]}) -> область {nodesArea[i]}\n");

            output.Append($"{margin}\n");
            output.Append("(Координаты конечных элементов) finitElements: \n");
            for (int i = 0; i < countFinitEl; i++)
                output.Append($"({finitElements[i][0]}, " +
                              $"{finitElements[i][1]}, "  +
                              $"{finitElements[i][2]}) -> область {areaFinitEl[i]}\n");

            output.Append($"{margin}\n");
            output.Append("(гамма и бетта для каждой области) materials: \n");
            for (int i = 0; i < countAreas; i++)
                output.Append($"{i}: \u03B3 = {materials[i].gamma}, \u03B2 = {materials[i].betta})\n");

            output.Append($"{margin}\n");
            output.Append("Данные граничных условий: \n");
            for (int i = 0; i < countKrayCond; i++)
                output.Append($"({boards[i][0]}, " +
                               $"{boards[i][1]}, " +
                               $"{boards[i][2]}, " +
                               $"{boards[i][3]}, " +
                               $"{boards[i][4]})\n");

            return output.ToString();
        }

    }
}
```

Function.cs

```
using static System.Math;

namespace Project
{
    public static class Function
    {
        public static uint NumberFunc;

        //* Функция u(x,y)
        public static double Absolut(double x, double y, uint area)
        {
            switch(NumberFunc)
            {
                case 1: /// test1
                return area switch
                {
                    0 => x,
                    _ => 0,
                };

                case 2: /// test2-SPLIT
                return area switch
                {
                    0 => 10*x + 10*y,
                    _ => 0,
                };

                case 3: /// test3-STUDY/test1
                return area switch
                {
                    0 => y*y,
                    1 => 20*y - 19,
                    _ => 0,
                };

                case 4: /// test3-STUDY/test2
                return area switch
                {
                    0 => x + 6*y - 2,
                    1 => x + 6*y - 2,
                    _ => 0,
                };

                case 5: /// test4-Decomposition
                return area switch
                {
                    0 => x,
                    _ => 0,
                };

            }
            return 0;
        }

        //* Функция f(x,y)
        public static double Func(double x, double y, uint area)
        {
            switch(NumberFunc)
            {
                case 1: /// test1
                return area switch
                {
                    0 => 2*x,
                    _ => 0,
```

```
            };

            case 2: /// test2-SPLIT
            return area switch
            {
                0 => 20*x + 20*y,
                _ => 0,
            };

            case 3: /// test3-STUDY/test1
            return area switch
            {
                0 => -20,
                1 => 0,
                _ => 0,
            };

            case 4: /// test3-STUDY/test2
            return area switch
            {
                0 => 5*x + 30*y - 10,
                1 => 0,
                _ => 0,
            };

            case 5: /// test4-Decomposition
            return area switch
            {
                0 => 2*x - 1,
                _ => 0,
            };

        }
        return 0;
    }

    //* Функция lambda(x,y)
    public static double Lambda(double x, double y, uint area)
    {
        switch(NumberFunc)
        {
            case 1: /// test1
            return area switch
            {
                0 => 1,
                _ => 0,
            };

            case 2: /// test2-SPLIT
            return area switch
            {
                0 => 4,
                _ => 0,
            };

            case 3: /// test3-STUDY/test1
            return area switch
            {
                0 => 10,
                1 => 1,
                _ => 0,
            };

            case 4: /// test3-STUDY/test2
            return area switch
            {
                0 => 1,
                1 => 1,
```

```
                _ => 0,
            };

            case 5: /// test4-Decomposition
            return area switch
            {
                0 => x,
                _ => 0,
            };
        }
        return 0;
    }

    //* Функция первого краевого условия
    public static double Func_First_Kraev(double x, double y, uint area)
    {
        switch(NumberFunc)
        {
            case 1: /// test1
            return area switch
            {
                0 => x,
                _ => 0,
            };

            case 2: /// test2-SPLIT
            return area switch
            {
                0 => 50 + 10*y,
                _ => 0,
            };

            case 3: /// test3-STUDY/test1
            return area switch
            {
                0 => y*y,
                _ => 0,
            };

            case 4: /// test3-STUDY/test2
            return area switch
            {
                0 => 6*y + 2,
                _ => 0,
            };

            case 5: /// test4-Decomposition
            return area switch
            {
                0 => x,
                _ => 0,
            };
        }
        return 0;
    }

    //* Функция второго краевого условия
    public static double Func_Second_Kraev(double x, double y, uint area, uint lam_area)
    {
        switch(NumberFunc)
        {
            case 1: /// test1
            return area switch
            {
                0 => 1,
                1 => -1,
                _ => 0,
            };
```

```
            case 2: /// test2-SPLIT
            return area switch
            {
                0 => -40,
                1 => 40,
                _ => 0,
            };

            case 3: /// test3-STUDY/test1
            return area switch
            {
                0 => 20,
                1 => 0,
                _ => 0,
            };

            case 4: /// test3-STUDY/test2
            return area switch
            {
                0 => -6,
                1 => -1,
                2 => 6,
                _ => 0,
            };

            case 5: /// test4-Decomposition
            return area switch
            {
                0 => x,
                1 => -x,
                _ => 0,
            };
        }
        return 0;
    }

    //* Функция третьего краевого условия
    public static double Func_Third_Kraev(double x, double y, uint area, uint lam_area)
    {
        switch(NumberFunc)
        {
            case 1: /// test1
            return area switch
            {
                0 => x,
                _ => 0,
            };

            case 2: /// test2-SPLIT
            return area switch
            {
                0 => 10*x + 2,
                _ => 0,
            };

            case 3: /// test3-STUDY/test1
            return area switch
            {
                0 => 20*y - 27,
                _ => 0,
            };

            case 4: /// test3-STUDY/test2
            return area switch
            {
                0 => 6*y + 2.1,
                _ => 0,
```

```
                };

                case 5: /// test4-Decomposition
                return area switch
                {
                    0 => x,
                    _ => 0,
                };
            }
            return 0;
        }


    }
}
```

LOS.cs

```
using static Project.Helper;
using static System.Math;
using System;
using System.Linq;

namespace Project
{
    public class LOS
    {
        private Matrix matrix;    /// Структура СЛАУ

    // ************ Коструктор LOS ************ //
        public LOS(Matrix matrix, int maxIter, double eps) {
            this.matrix         = matrix;
            this.matrix.maxIter = maxIter;
            this.matrix.epsilon = eps;
        }

        //* Решение СЛАУ
        public Matrix solve(bool isLog = true) {
            var r      = new double[matrix.N];
            var z      = new double[matrix.N];
            var multLr = new double[matrix.N];
            var Lr     = new double[matrix.N];
            var p      = new double[matrix.N];
            double alpha, betta, Eps;
            int iter = 0;

            double[] L = Enumerable.Range(0, matrix.N).Select(i => 1.0 / matrix.di[i]).ToArray();

            double[] multX = matrix.mult(matrix.x);
            for (int i = 0; i < r.Length; i++) {
                r[i] = L[i] * (matrix.pr[i] - multX[i]);
                z[i] = L[i] * r[i];
            }
            double[] multZ = matrix.mult(z);
            for (int i = 0; i < p.Length; i++)
                p[i] = L[i] * multZ[i];

            do
            {
                betta = scalar(p, p);
                alpha = scalar(p, r) / betta;
                for (int i = 0; i < matrix.x.Length; i++) {
                    matrix.x[i] += alpha * z[i];
                    r[i]        -= alpha * p[i];
                    Lr[i]       = L[i] * r[i];
                }

                multLr = matrix.mult(Lr);
                for (int i = 0; i < Lr.Length; i++)
                    multLr[i] = L[i] * multLr[i];
                betta = -scalar(p, multLr) / betta;
                for (int i = 0; i < z.Length; i++) {
                    z[i] = L[i] * r[i] + betta * z[i];
                    p[i] = multLr[i] + betta * p[i];
                }
                Eps = scalar(r, r);

                iter++;
                if (isLog) printLog(iter, Eps);
            } while (iter < matrix.maxIter
                && Eps > matrix.epsilon);
```

```
            return matrix;
        }

        //* Вывод невязки и количества итераций
        private void printLog(int Iter, double Eps) {
            Console.WriteLine($"Iteration = {Iter}\t\t" +
                              $"Discrepancy = {Eps}");
        }
    }
}
```

FEM.cs

```
using static System.Console;
using static System.Math;
using static Project.Function;
using static Project.Helper;
using System.Text;
using System.Linq;
using System.IO;
using System;
using System.Collections.Generic;

namespace Project
{
    public class FEM : Data
    {
        protected string Path; /// Путь к папке с задачей
        private Matrix matrix; /// Структура СЛАУ

        // ************ Коструктор FEM ************ //
        public FEM(string Path, uint Num) : base(Path) { this.Path = Path; Function.NumberFunc = Num; }

        public void solve()
        {
            portrait();                             //? Составление списка связанностей и массивов ig[] и jg[]
            global();                               //? Составление глобальный матрицы
            LOS los = new(matrix, 10000, 9e-030);   //? Создание метода LOS
            matrix = los.solve(false);              //? Решение СЛАУ методом ЛОС (диагональный)
            AbsolutSolve();                         //? Абсолютное решение СЛАУ
            WriteMatrix();                          //? Запись матрицы и вектора решения СЛАУ
            WriteTable();                           //? Запись таблички с решением и погрешностью

        }

        //* Составление портрета
        private void portrait()
        {
            // Составление списка связанностей
            var list = new uint[countNode][];

            var listI = new HashSet<uint>();
            for (uint i = 0; i < numNodes.Length; i++)
            {
                uint value = numNodes[i];
                for (uint k = 0; k < countFinitEl; k++)
                {
                    if (finitElements[k].Contains(value))
                        for (uint p = 0; p < 3; p++)
                            if (finitElements[k][p] < value)
                                listI.Add(finitElements[k][p]);
                }
                list[i] = listI.OrderBy(n => ((uint)n)).ToArray();
                listI.Clear();
            }
```

```
        // Заполнение ig[]
        matrix.ig = new uint[countNode + 1];
        matrix.ig[0] = matrix.ig[1] = 0;
        for (uint i = 1; i < countNode; i++)
            matrix.ig[i + 1] = (matrix.ig[i] + (uint)list[i].Length);

        // Заполнение jg[]
        matrix.jg = new uint[matrix.ig[countNode]];
        int jj = 0;
        for (uint i = 0; i < countNode; i++)
            for (uint j = 0; j < list[i].Length; j++, jj++)
                matrix.jg[jj] = list[i][j];

        // Размерность глобальной матрицы
        matrix.N = (int)countNode;
        resize(); // Выделение памяти
    }

    //* Построение глоабльной матрицы
    private void global()
    {
        // Для каждого конечного элемента
        for (uint index_fin_el = 0; index_fin_el < countFinitEl; index_fin_el++)
        {
            // Составим локальную матрицу и локальный вектор
            (double[][] local_matrix, double[] local_f) = local(index_fin_el);

            // Занесение в глобальную
            EntryMatInGlobalMatrix(local_matrix, finitElements[index_fin_el]);
            EntryVecInGlobalMatrix(local_f, finitElements[index_fin_el]);
        }

        // Для каждого условия на границе
        for (uint index_kraev_cond = 0; index_kraev_cond < countKrayCond; index_kraev_cond++)
        {
            uint[] curr_kraev = boards[index_kraev_cond];   // Данные краевого условия
            uint[] Node = { curr_kraev[1], curr_kraev[2] }; // Ребро на котором задано условие
            if (curr_kraev[3] == 1)
                First_Kraev(curr_kraev);
            else if (curr_kraev[3] == 2)
            {
                double[] corr_vec = Second_Kraev(curr_kraev);
                EntryVecInGlobalMatrix(corr_vec, Node);
            }
            else
            {
                (double[][] corr_mat, double[] corr_vec) = Third_Kraev(curr_kraev);
                EntryMatInGlobalMatrix(corr_mat, Node);
                EntryVecInGlobalMatrix(corr_vec, Node);
            }
        }
    }

    //* Построение локальной матрицы и вектора
    private (double[][], double[]) local(uint index_fin_el)
    {
        double[] local_f = build_F(index_fin_el);    // Построение локального вектора
        double[][] M = build_M(index_fin_el);        // Построение матрица массы
        double[][] G = build_G(index_fin_el);        // Построение матрица жесткости
        double[][] local_matrix = SummMatrix(G, M);  // Локальная матрицы (G + M)

        return (local_matrix, local_f);
    }

    //* Занесение матрицы в глоабальную матрицу
    private void EntryMatInGlobalMatrix(double[][] mat, uint[] index)
    {
```

```
        for (uint i = 0, h = 0; i < mat.GetUpperBound(0) + 1; i++)
        {
            uint ibeg = index[i];
            for (uint j = i + 1; j < mat.GetUpperBound(0) + 1; j++)
            {
                uint iend = index[j];
                uint temp = ibeg;

                if (temp < iend)
                    (iend, temp) = (temp, iend);

                h = matrix.ig[temp];
                while (matrix.jg[h++] - iend != 0);
                matrix.gg[--h] += mat[i][j];
            }
            matrix.di[ibeg] += mat[i][i];
        }
    }

    //* Занесение вектора в глолбальный вектор
    private void EntryVecInGlobalMatrix(double[] vec, uint[] index)
    {
        for (uint i = 0; i < vec.Length; i++)
            matrix.pr[index[i]] += vec[i];
    }

    //* Построение вектора правой части
    private double[] build_F(uint index_fin_el)
    {
        uint area_fin_el = areaFinitEl[index_fin_el];        // Область в которой расположек к.э.
        double detD = Abs(ComputeDet(index_fin_el)) / 24.0;  // Вычисление detD

        var f = new double[3];                               // Вычисление f - на узлах к.э.
        for (uint i = 0; i < f.Length; i++)
            f[i] = detD * Func(
                 nodes[finitElements[index_fin_el][i]][0],
                 nodes[finitElements[index_fin_el][i]][1],
                 area_fin_el
            );

        var local_f = new double[3];                         // Вычисление локального вектора
        local_f[0] = 2 * f[0] + f[1] + f[2];
        local_f[1] = 2 * f[1] + f[0] + f[2];
        local_f[2] = 2 * f[2] + f[1] + f[0];

        return local_f;
    }

    //* Построение матрицы масс
    private double[][] build_M(uint index_fin_el)
    {
        var M_matrix = new double[3][];                                  // Матрица масс
        for (uint i = 0; i < 3; i++) M_matrix[i] = new double[3];

        uint area_fin_el = areaFinitEl[index_fin_el];                    // Область в которой расположек к.э.
        double gamma = materials[area_fin_el].gamma;                     // Значение гаммы в этой области

        double value = gamma * Abs(ComputeDet(index_fin_el)) / 24.0;     // Значение матрицы массы

        for (uint i = 0; i < M_matrix.GetUpperBound(0) + 1; i++)         // Заполнение матрицы масс
            for (uint j = 0; j < M_matrix.GetUpperBound(0) + 1; j++)
                M_matrix[i][j] = i == j ? 2 * value
                                        : value;
        return M_matrix;
    }

    //* Построение матрицы жесткости
    private double[][] build_G(uint index_fin_el)
```

```
    {
        var G_matrix = new double[3][];                             // Матрица жесткости
        for (uint i = 0; i < 3; i++) G_matrix[i] = new double[3];

        double[] Node1 = nodes[finitElements[index_fin_el][0]];     // Координаты 1 узла i - конечного элемента
        double[] Node2 = nodes[finitElements[index_fin_el][1]];     // Координаты 2 узла i - конечного элемента
        double[] Node3 = nodes[finitElements[index_fin_el][2]];     // Координаты 3 узла i - конечного элемента
        double[] Mid12 = MidPoints(Node1, Node2);                   // Координаты середины ребра 1-2
        double[] Mid13 = MidPoints(Node1, Node3);                   // Координаты серидины ребра 1-3
        double[] Mid23 = MidPoints(Node2, Node3);                   // Координаты середины ребра 2-3

        uint area_fin_el = areaFinitEl[index_fin_el];               // Область в которой расположек к.э.
        double[,] a = ComputeA(index_fin_el);                       // Вычисление а-компонент

        double lambda = Lambda(Mid12[0], Mid12[1], area_fin_el) +
                        Lambda(Mid13[0], Mid13[1], area_fin_el) +
                        Lambda(Mid23[0], Mid23[1], area_fin_el);    // Подсчет лямбда разложения

        double multip = lambda / (6.0 * Abs(ComputeDet(index_fin_el)));

        for (uint i = 0; i < G_matrix.GetUpperBound(0) + 1; i++)    // Заполнение матрицы жесткости
            for (uint j = 0; j < G_matrix.GetUpperBound(0) + 1; j++)
                G_matrix[i][j] = multip * (a[i, 0] * a[j, 0] + a[i, 1] * a[j, 1]);

        return G_matrix;
    }

    //* Первое краевое условие
    private void First_Kraev(uint[] kraev)
    {
        // Ставим вместо диагонального эл. единицу
        matrix.di[kraev[1]] = 1;
        matrix.di[kraev[2]] = 1;

        // В вектор правой части ставим значение краевого условия
        matrix.pr[kraev[1]] = Func_First_Kraev(nodes[kraev[1]][0], nodes[kraev[1]][1], kraev[4]);
        matrix.pr[kraev[2]] = Func_First_Kraev(nodes[kraev[2]][0], nodes[kraev[2]][1], kraev[4]);

        // Зануляем в строке все стоящие элементы кроме диагонального и сразу делаем симметричной
        for (uint k = 1; k < 3; k++)
        {

            // Зануление в нижнем треугольнике
            for (uint i = matrix.ig[kraev[k]]; i < matrix.ig[kraev[k] + 1]; i++)
            {
                if (matrix.di[matrix.jg[i]] != 1)
                    matrix.pr[matrix.jg[i]] -= matrix.gg[i] * matrix.pr[kraev[k]];
                matrix.gg[i] = 0;
            }

            // Зануление в верхнем треугольнике, но т.к. делаем симметричную "зануление в нижнем"
            for (uint i = kraev[k] + 1; i < countNode; i++)
            {
                uint lbeg = matrix.ig[i];
                uint lend = matrix.ig[i + 1];
                for (uint p = lbeg; p < lend; p++)
                    if (matrix.jg[p] == kraev[k])
                    {
                        if (matrix.di[i] != 1)
                            matrix.pr[i] -= matrix.gg[p] * matrix.pr[kraev[k]];
                        matrix.gg[p] = 0;
                    }
            }
        }
    }

    //* Второе краевое условие
    private double[] Second_Kraev(uint[] kraev)
```

```
    {
        var corr_vec = new double[2];                               // Корректирующий вектор

        uint[] Node = { kraev[1], kraev[2] };                       // Ребро

        double betta = materials[kraev[0]].betta;                   // Значение бетты
        double multip = ComputeMesG(nodes[Node[0]], nodes[Node[1]]) / 6.0;

        for (uint i = 0, j = 1; i < corr_vec.Count(); i++, j--)     // Заполнение вектора
            corr_vec[i] = multip * (2 * Func_Second_Kraev(nodes[Node[i]][0], nodes[Node[i]][1], kraev[4], kraev[0]) +
                                       Func_Second_Kraev(nodes[Node[j]][0], nodes[Node[j]][1], kraev[4], kraev[0]));
        return corr_vec;
    }

    //* Третье краевое условие
    private (double[][], double[]) Third_Kraev(uint[] kraev)
    {
        var corr_vec = new double[2];                               // Корректирующий вектор
        var corr_mat = new double[2][];                             // Корректирующая матрица
        for (uint i = 0; i < 2; i++) corr_mat[i] = new double[2];

        uint[] Node = { kraev[1], kraev[2] };                       // Ребро
        double betta = materials[kraev[0]].betta;                   // Значение Betta

        double multip = (betta * ComputeMesG(nodes[Node[0]], nodes[Node[1]])) / 6.0;

        for (uint i = 0, k = 1; i < corr_vec.Count(); i++, k--)
        {  // Заполнение вектора и матрицы
            corr_vec[i] = multip * (2 * Func_Third_Kraev(nodes[Node[i]][0], nodes[Node[i]][1], kraev[4], kraev[0]) +
                                       Func_Third_Kraev(nodes[Node[k]][0], nodes[Node[k]][1], kraev[4], kraev[0]));

            for (uint j = 0; j < corr_mat.Count(); j++)
                corr_mat[i][j] = i == j ? 2 * multip
                                        : multip;
        }
        return (corr_mat, corr_vec);
    }

    //* Подсчет компонента detD
    private double ComputeDet(uint index_fin_el)
    {
        double[] Node1 = nodes[finitElements[index_fin_el][0]]; // Координаты 1 узла i - конечного элемента
        double[] Node2 = nodes[finitElements[index_fin_el][1]]; // Координаты 2 узла i - конечного элемента
        double[] Node3 = nodes[finitElements[index_fin_el][2]]; // Координаты 3 узла i - конечного элемента

        return (Node2[0] - Node1[0]) * (Node3[1] - Node1[1]) -
               (Node3[0] - Node1[0]) * (Node2[1] - Node1[1]);
    }

    //* Подсчет компонента mes по ребру G
    private double ComputeMesG(double[] Node1, double[] Node2)
    {
        return Sqrt(Pow((Node2[0] - Node1[0]), 2) +
                    Pow((Node2[1] - Node1[1]), 2));
    }

    //* Подсчет компонентов a
    private double[,] ComputeA(uint index_fin_el)
    {
        var a = new double[3, 2];
        double[] Node1 = nodes[finitElements[index_fin_el][0]]; // Координаты 1 узла i - конечного элемента
        double[] Node2 = nodes[finitElements[index_fin_el][1]]; // Координаты 2 узла i - конечного элемента
        double[] Node3 = nodes[finitElements[index_fin_el][2]]; // Координаты 3 узла i - конечного элемента

        // Заполнение a
        a[0, 0] = Node2[1] - Node3[1];
        a[1, 0] = Node3[1] - Node1[1];
        a[2, 0] = Node1[1] - Node2[1];
```

```
        a[0, 1] = Node3[0] - Node2[0];
        a[1, 1] = Node1[0] - Node3[0];
        a[2, 1] = Node2[0] - Node1[0];
        return a;
    }

    //* Абсолютное решение СЛАУ
    private void AbsolutSolve()
    {
        for (uint i = 0; i < countNode; i++)
            matrix.absolut_x[i] = Absolut(nodes[i][0], nodes[i][1], nodesArea[i]);
    }

    //* resize массивов глобальной матрицы
    private void resize()
    {
        matrix.di = new double[matrix.N];
        matrix.pr = new double[matrix.N];
        matrix.x = new double[matrix.N];
        matrix.absolut_x = new double[matrix.N];
        matrix.gg = new double[matrix.jg.Length];
    }

    //* Запись глобальной матрицы
    private void WriteMatrix()
    {
        Directory.CreateDirectory(Path + "matrix");
        Directory.CreateDirectory(Path + "output");
        File.WriteAllText(Path + "matrix/kuslau.txt", $"{matrix.N} {matrix.maxIter} {matrix.epsilon}");
        File.WriteAllText(Path + "matrix/ig.txt", String.Join(" ", matrix.ig));
        File.WriteAllText(Path + "matrix/jg.txt", String.Join(" ", matrix.jg));
        File.WriteAllText(Path + "matrix/di.txt", String.Join("\n", matrix.di));
        File.WriteAllText(Path + "matrix/gg.txt", String.Join("\n", matrix.gg));
        File.WriteAllText(Path + "matrix/pr.txt", String.Join("\n", matrix.pr));
        File.WriteAllText(Path + "output/x.txt", String.Join("\n", matrix.x));
        File.WriteAllText(Path + "output/x_absolut.txt", String.Join("\n", matrix.absolut_x));
    }

    //* Запись таблички с погрешностью
    private void WriteTable()
    {
        (double[] SubX, double norma) = Norm(matrix.absolut_x, matrix.x);

        StringBuilder table = new StringBuilder();
        string margin = String.Join("", Enumerable.Repeat("-", 23));

        table.Append(String.Join("", Enumerable.Repeat("-", 97)) + "\n");
        table.Append($"|X`{" ",-20} | X{" ",-20} | |X` - X|{" ",-13} | ||X` - X||  {" ",-9} |\n");
        table.Append($"|" + margin + "|" + margin + "|" + margin + "|" + margin + "|\n");

        for (uint i = 0; i < countNode; i++)
        {
            table.Append($"|{String.Format("{0,-23}", matrix.absolut_x[i])}" +
                         $"|{String.Format("{0,-23}", matrix.x[i])}" +
                         $"|{SubX[i].ToString("E3")}{String.Format("{0,-13}", "")}|");
            if (countNode / 2 == i)
                table.Append($"{norma.ToString("E3")}{String.Format("{0,-13}", "")}|");
            else
                table.Append($"{String.Format("{0,-23}", " ")}|");
            table.Append("\n");
        }
        table.Append(String.Join("", Enumerable.Repeat("-", 97)) + "\n");
        File.WriteAllText(Path + "output/table.txt", table.ToString());
    }

    //* Расчет погрешности и нормы решения
    private (double[], double) Norm(double[] x_abs, double[] x)
    {
```

```
            double norm = 0;
            double[] norm_arr = new double[x.Length];

            for (int i = 0; i < x.Length; i++)
            {
                norm_arr[i] = Abs(x_abs[i] - x[i]);
                norm += Pow(norm_arr[i], 2);
            }
            return (norm_arr, Sqrt(norm));
        }

        //* Вывод матрицы на консоль
        private void PrintMatrix()
        {
            WriteLine("Matrix: ");
            PrintArray(matrix.di, "di        = [");
            PrintArray(matrix.gg, "gg        = [");
            PrintArray(matrix.ig, "ig        = [");
            PrintArray(matrix.jg, "jg        = [");
            PrintArray(matrix.pr, "pr        = [");
        }

        //* Вывод массива на консоль
        private void PrintArray<T>(T[] array, string design = "array = [")
        {
            Write(design);
            for (int i = 0; i < array.Length; i++)
                Write(array[i] + " ");
            WriteLine("]");
        }
    }
}
```

Program.cs

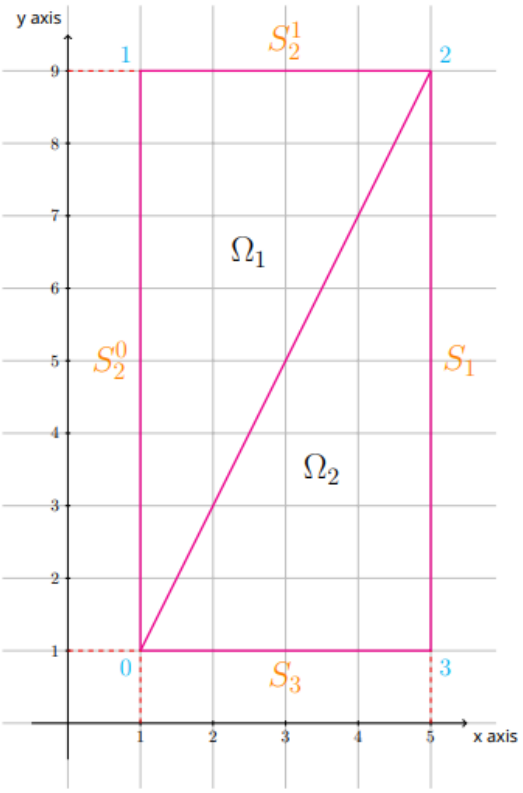
```
using static System.Console;
using Project;

FEM task = new("file/test1/", 1);               //! Обычный тест с одной областью в начале координат
//FEM task = new("file/test2-SPLIT/split1/", 2);  //! Тест с разбиением #1
//FEM task = new("file/test2-SPLIT/split2/", 2);  //! Тест с разбиением #2
//FEM task = new("file/test2-SPLIT/split3/", 2);  //! Тест с разбиением #3
//FEM task = new("file/test2-SPLIT/split4/", 2);  //! Тест с разбиением #4
//FEM task = new("file/test3-STUDY/study1/", 3);  //! Тест с книги МКЭ #1
//FEM task = new("file/test3-STUDY/study2/", 4);  //! Тест с книги МКЭ #2
//FEM task = new("file/test4-Decomposition/", 5); //! Тест с книги МКЭ #2

task.solve();                                   //! Запуск решения задачи
//WriteLine(task.dataAll());                    //! Вывод всех данных задачи
```

Данные задачи:

$$\begin{cases} u(x,y) = 10x + 10y \\ f(x,y) = 20x + 20y \\ \lambda = 4 \\ \gamma = 2 \\ \beta = 5 \\ \Omega^n, n = 1 \end{cases}$$



Краевые условия на границах:

$$\begin{cases} R_{03} = III_0 \\ R_{32} = I_0 \\ R_{21} = II_1 \\ R_{10} = II_0 \end{cases} -> \begin{cases} I_0 = 50 + 10y \\ II_0 = -40 \\ II_1 = 40 \\ III_0 = 10x + 2 \end{cases}$$

Содержимое файлов:

xy.txt

$$\begin{bmatrix} (1,1) \\ (1,9) \\ (5,9) \\ (5,1) \end{bmatrix}$$

finitel.txt

$$\begin{bmatrix} (0,1,2) \\ (0,2,3) \end{bmatrix}$$

param.txt

$$\begin{bmatrix} 4\,2\,1\,4 \end{bmatrix}$$

XY_area.txt

$$\begin{bmatrix} \{0...0\}_4 \end{bmatrix}$$

area.txt

$$\begin{bmatrix} 2\,5 \\ \{0...0\}_2 \end{bmatrix}$$

boards.txt

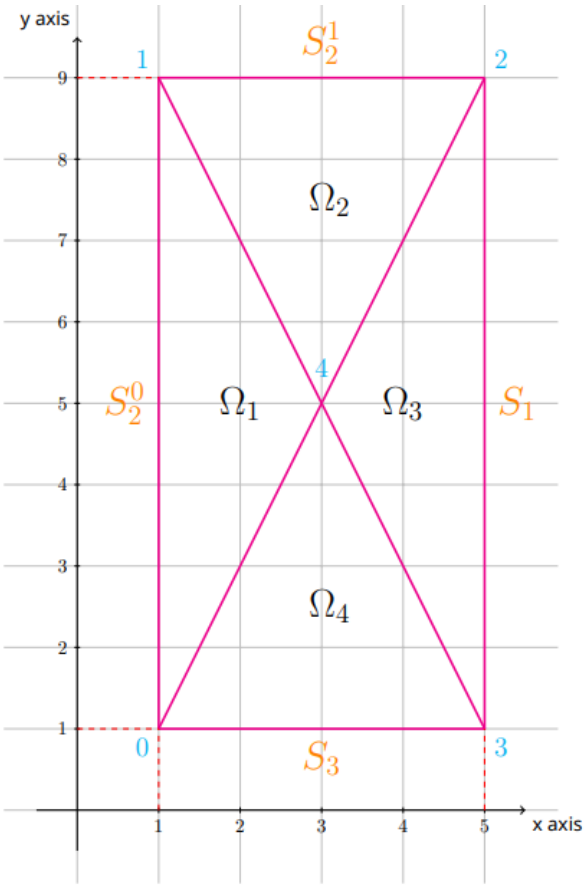
$$\begin{bmatrix} 0\,0\,3\,3\,0 \\ 0\,3\,2\,1\,0 \\ 0\,2\,1\,2\,1 \\ 0\,1\,0\,2\,0 \end{bmatrix}$$

Табличка с решением:

X`	X	|X` - X|	||X` - X||
20	19,999999999999986	1,421E-014	
100	100,00000000000006	5,684E-014	
140	139,99999999999997	2,842E-014	6,512E-014
60	60	0,000E+000	

Данные задачи:

$$\begin{cases} u(x,y) = 10x + 10y \\ f(x,y) = 20x + 20y \\ \lambda = 4 \\ \gamma = 2 \\ \beta = 5 \\ \Omega^n, n = 1 \end{cases}$$



Краевые условия на границах:

$$\begin{cases} R_{03} = III_0 \\ R_{32} = I_0 \\ R_{21} = II_1 \\ R_{10} = II_0 \end{cases} -> \begin{cases} I_0 = 50 + 10y \\ II_0 = -40 \\ II_1 = 40 \\ III_0 = 10x + 2 \end{cases}$$

Содержимое файлов:

xy.txt

$$\begin{matrix} (1,1) \\ (1,9) \\ (5,9) \\ (5,1) \\ (3,5) \end{matrix}$$

finitel.txt

$$\begin{matrix} (0,1,4) \\ (1,2,4) \\ (2,3,4) \\ (0,3,4) \end{matrix}$$

param.txt

$$5\,4\,1\,4$$

XY_area.txt

$$\{0 \ldots 0\}_5$$

area.txt

$$25$$
$$\{0 \ldots 0\}_4$$

boards.txt

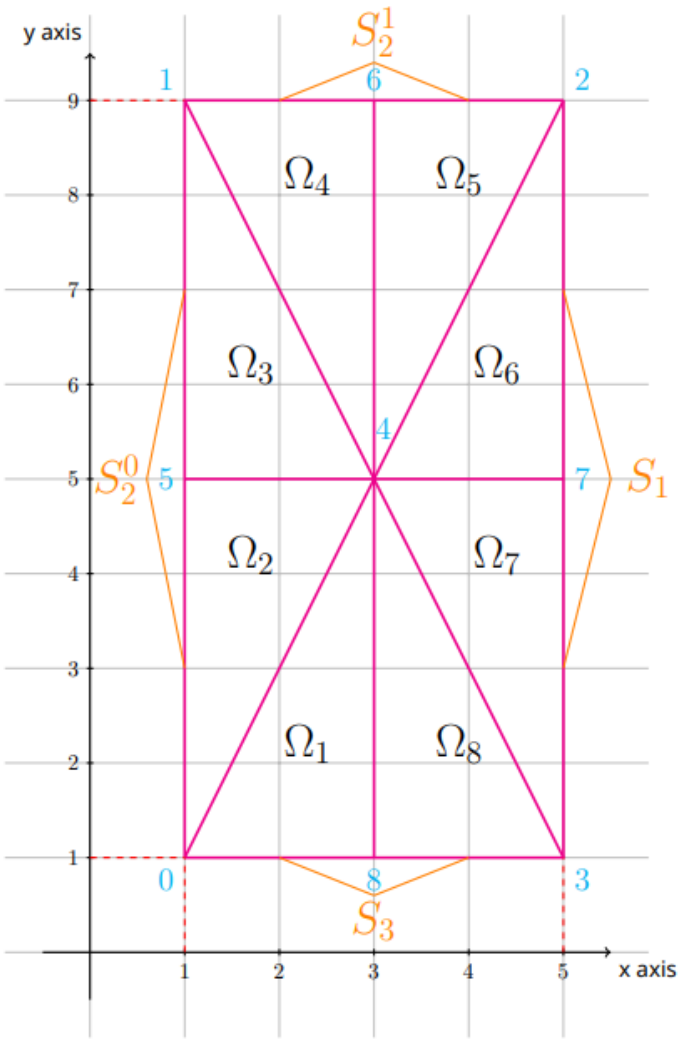
$$\begin{matrix} 00330 \\ 03210 \\ 02121 \\ 01020 \end{matrix}$$

Табличка с решением:

X`	X	|X` - X|	||X` - X||
20	19,999999999999996	3,553E-015	
100	100,00000000000001	1,421E-014	
140	140	0,000E+000	2,041E-014
60	60	0,000E+000	
80	79,99999999999999	1,421E-014	

Данные задачи:

$$\begin{cases} u(x,y) = 10x + 10y \\ f(x,y) = 20x + 20y \\ \lambda = 4 \\ \gamma = 2 \\ \beta = 5 \\ \Omega^n, n = 1 \end{cases}$$



Краевые условия на границах:

$$\begin{cases} R_{08} = III_0 \\ R_{83} = III_0 \\ R_{37} = I_0 \\ R_{72} = I_0 \\ R_{26} = II_1 \\ R_{61} = II_1 \\ R_{15} = II_0 \\ R_{50} = II_0 \end{cases} \quad -> \begin{cases} I_0 = 50 + 10y \\ II_0 = -40 \\ II_1 = 40 \\ III_0 = 10x + 2 \end{cases}$$

Содержимое файлов:

xy.txt

$$(1,1),(1,5)$$
$$(1,9),(3,9)$$
$$(5,9),(5,5)$$
$$(5,1),(3,1)$$
$$(3,5)$$

finitel.txt

$$(0,4,8),(2,4,6)$$
$$(0,4,5),(2,4,7)$$
$$(1,4,5),(3,4,7)$$
$$(1,4,6),(3,4,8)$$

param.txt

9 8 1 8

XY_area.txt

$\{0...0\}_9$

area.txt

25

$\{0...0\}_8$

boards.txt

00830;02621
08330;06121
03710;01520
07210;05020

Табличка с решением:

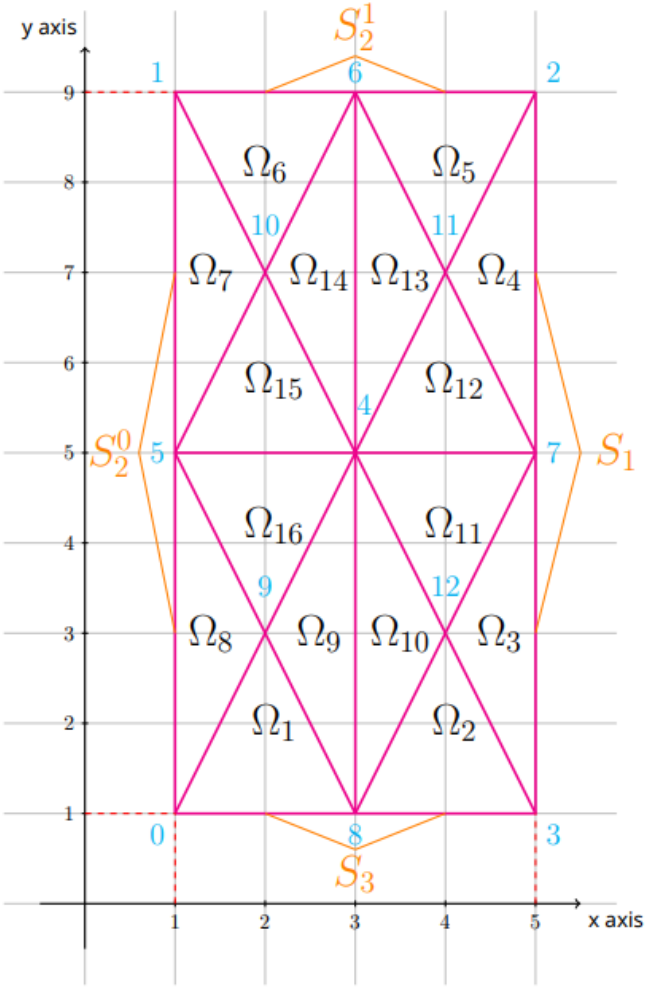
X`	X	|X` - X|	||X` - X||
20	20	0,000E+000	
100	100	0,000E+000	
140	140,00000000000003	2,842E-014	
60	60	0,000E+000	
80	80,00000000000003	2,842E-014	4,263E-014
60	60,000000000000014	1,421E-014	
120	120	0,000E+000	
100	100	0,000E+000	
40	40	0,000E+000	

Данные задачи:

$$\begin{cases} u(x,y) = 10x + 10y \\ f(x,y) = 20x + 20y \\ \lambda = 4 \\ \gamma = 2 \\ \beta = 5 \\ \Omega^n, n = 1 \end{cases}$$

Краевые условия на границах:

$$\begin{cases} R_{08} = III_0 \\ R_{83} = III_0 \\ R_{37} = I_0 \\ R_{72} = I_0 \\ R_{26} = II_1 \\ R_{61} = II_1 \\ R_{15} = II_0 \\ R_{50} = II_0 \end{cases} \quad -> \begin{cases} I_0 = 50 + 10y \\ II_0 = -40 \\ II_1 = 40 \\ III_0 = 10x + 2 \end{cases}$$



Содержимое файлов:

xy.txt

$(1,1),(5,5)$
$(1,9),(3,1)$
$(5,9),(2,3)$
$(5,1),(2,7)$
$(3,5),(4,7)$
$(1,5),(4,3)$
$(3,9)$

finitel.txt

$(0,8,9),\ (4,8,9)$
$(3,8,12),(4,8,12)$
$(3,7,12),(4,7,12)$
$(2,7,11),(4,7,11)$
$(2,6,11),(4,6,11)$
$(1,6,10),(4,6,10)$
$(1,5,10),(4,5,10)$
$(0,5,9),\ (4,5,9)$

param.txt

13 16 1 8

XY _ area.txt

$\{0...0\}_{13}$

area.txt

25
$\{0...0\}_{16}$

boards.txt

00830;02621
08330;06121
03710;01520
07210;05020

Табличка с решением:

x`	x	|x` - x|	||x` - x||
20	19,999999999999996	3,553E-015	
100	99,99999999999997	2,842E-014	
140	140	0,000E+000	
60	60,000000000000014	1,421E-014	
80	79,99999999999999	1,421E-014	
60	59,99999999999998	2,132E-014	
120	120,00000000000001	1,421E-014	7,079E-014
100	100	0,000E+000	
40	39,99999999999999	7,105E-015	
50	49,99999999999998	2,132E-014	
90	89,99999999999996	4,263E-014	
110	110	0,000E+000	
70	69,99999999999997	2,842E-014	

Данные задачи:

$$u(x,y)=\begin{cases} y^2, & (x,y)\in\Omega^0 \\ 20y-19, & (x,y)\in\Omega^1 \end{cases}$$

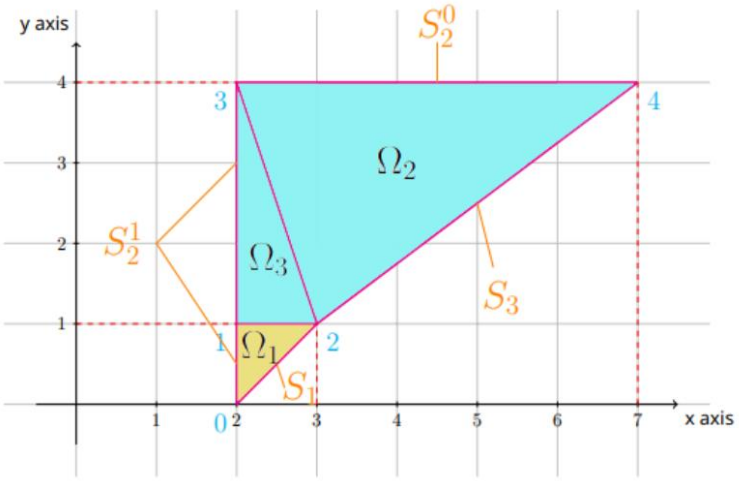
$$f(x,y)=\begin{cases} -20, & (x,y)\in\Omega^0 \\ 0, & (x,y)\in\Omega^1 \end{cases}$$

$$\lambda=\begin{cases} 10, & (x,y)\in\Omega^0 \\ 1, & (x,y)\in\Omega^1 \end{cases}$$

$$\gamma=0$$

$$\beta=2$$

$$\Omega^n, n=2$$



Краевые условия на границах:

$$\begin{bmatrix} R_{01}=II_1 \\ R_{13}=II_1 \\ R_{34}=II_0 \\ R_{42}=III_0 \\ R_{20}=I_0 \end{bmatrix} \rightarrow \begin{cases} I_0=y^2 \\ II_0=20 \\ II_1=0 \\ III_0=20y-27 \end{cases}$$

Содержимое файлов:

xy.txt

$$\begin{bmatrix} (2,0) \\ (2,1) \\ (3,1) \\ (2,4) \\ (7,4) \end{bmatrix}$$

finitel.txt

$$\begin{bmatrix} (0,1,2) \\ (1,2,3) \\ (2,3,4) \end{bmatrix}$$

param.txt

$$\begin{bmatrix} 5 & 3 & 2 & 5 \end{bmatrix}$$

XY_area.txt

$$\begin{bmatrix} \{0..0\}_3, \{1..1\}_2 \end{bmatrix}$$

area.txt

$$\begin{bmatrix} 0 & 2 \\ 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}$$

boards.txt

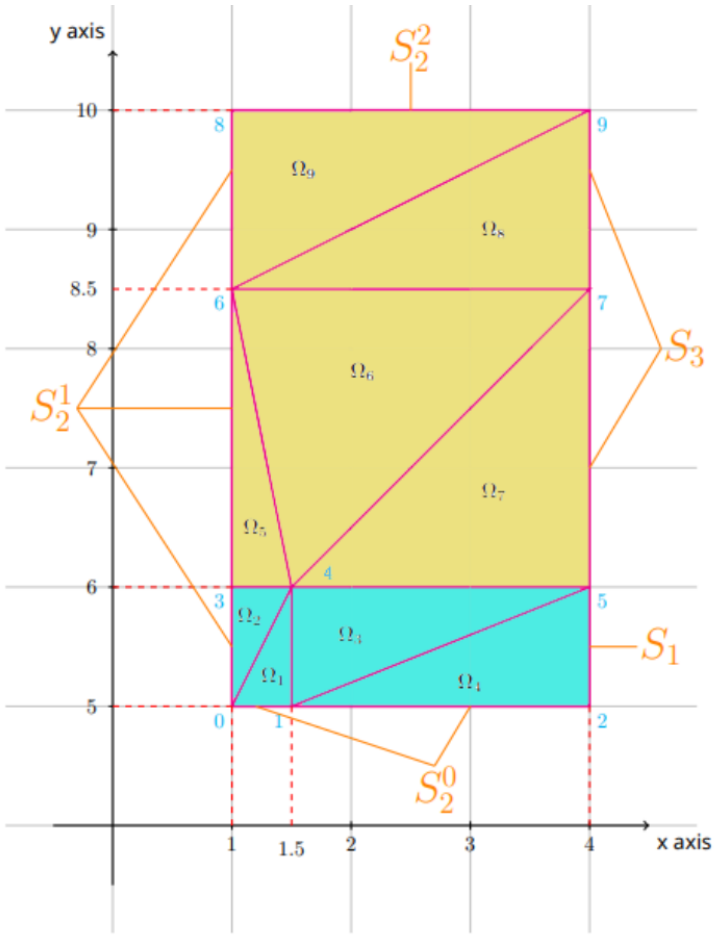
$$\begin{bmatrix} 0 & 0 & 1 & 2 & 1 \\ 1 & 1 & 3 & 2 & 1 \\ 1 & 3 & 4 & 2 & 0 \\ 1 & 4 & 2 & 3 & 0 \\ 0 & 0 & 2 & 1 & 0 \end{bmatrix}$$

Табличка с решением:

X`	X	|X` - X|	||X` - X||
0	0	0,000E+000	
1	1,1335811106252733	1,336E-001	
1	0,9999999999999999	1,110E-016	6,636E-001
61	60,350677743769126	6,493E-001	
61	60,9704853519895	2,951E-002	

Данные задачи:

$$\begin{cases} u(x,y) = x + 6y - 2 \\ f(x,y) = \begin{cases} 5x + 30y - 10, & (x,y) \in \Omega^1 \\ 0, & (x,y) \in \Omega^2 \end{cases} \\ \lambda = 1 \\ \gamma = \begin{cases} 5, & (x,y) \in \Omega^1 \\ 0, & (x,y) \in \Omega^2 \end{cases} \\ \beta = 10 \\ \Omega^n,\ n = 2 \end{cases}$$



Краевые условия на границах:

$$\begin{cases} R_{01} = II_0 \\ R_{12} = II_0 \\ R_{25} = I_0 \\ R_{57} = III_0 \\ R_{79} = III_0 \\ R_{98} = II_2 \\ R_{86} = II_1 \\ R_{63} = II_1 \\ R_{30} = II_1 \end{cases} \quad -> \begin{cases} I_0 = 6y + 2 \\ II_0 = -6 \\ II_1 = -1 \\ II_2 = 6 \\ III_0 = 6y + 2.1 \end{cases}$$

Содержимое файлов:

xy.txt

$$\begin{matrix} (1,5), & (4,6) \\ (1.5,5), & (1,8.5) \\ (4,5), & (4,8.5) \\ (1,6), & (1,10) \\ (1.5,6), & (4,10) \end{matrix}$$

finitel.txt

$$\begin{matrix} (0,1,4), & (4,6,7) \\ (0,3,4), & (4,5,7) \\ (1,4,5), & (6,7,9) \\ (1,2,5), & (6,8,9) \\ (3,4,6) & \end{matrix}$$

param.txt

10 9 2 9

XY_area.txt

$$\left[\{0..0\}_3, \{1..1\}_7 \right]$$

area.txt

5 10
0 10
$\{0..0\}_4, \{1..1\}_5$

boards.txt

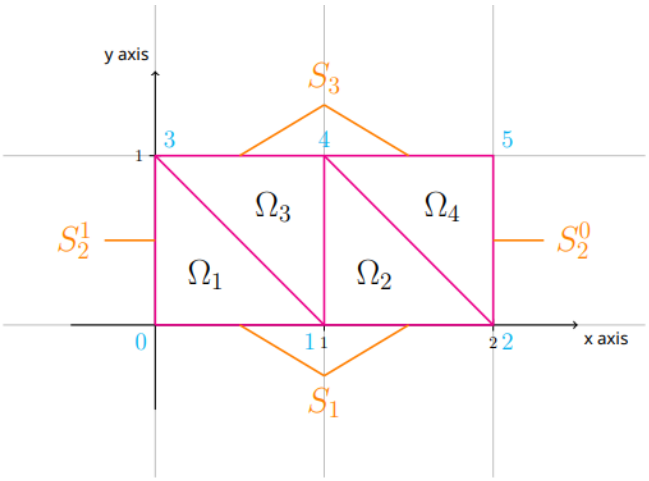
02510,16321
15730,03021
17930,00120
19822,01220
18621,

Табличка с решением:

X`	X	|X` - X|	||X` - X||
29	29,000000000000004	3,553E-015	
29,5	29,499999999999993	7,105E-015	
32	32,000000000000014	1,421E-014	
35	34,99999999999999	7,105E-015	
35,5	35,50000000000001	7,105E-015	
38	38	0,000E+000	2,161E-014
50	50	0,000E+000	
53	53	0,000E+000	
59	58,99999999999999	7,105E-015	
62	61,99999999999999	7,105E-015	

Данные задачи:

$$\begin{bmatrix} u(x,y) = x \\ f(x,y) = 2x \\ \lambda = 1 \\ \gamma = 2 \\ \beta = 2 \\ \Omega^n, n = 1 \end{bmatrix}$$



Краевые условия на границах:

$$\begin{bmatrix} R_{01} = I_0 \\ R_{12} = I_0 \\ R_{25} = II_0 \\ R_{54} = III_0 \\ R_{43} = III_0 \\ R_{30} = II_1 \end{bmatrix} \rightarrow \begin{cases} I_0 = x \\ II_0 = 1 \\ II_1 = -1 \\ III_0 = x \end{cases}$$

Содержимое файлов:

xy.txt

$$\begin{bmatrix} (0,0) \\ (1,0) \\ (2,0) \\ (0,1) \\ (1,1) \\ (2,1) \end{bmatrix}$$

finitel.txt

$$\begin{bmatrix} (0,1,3) \\ (1,3,4) \\ (1,2,4) \\ (2,4,5) \end{bmatrix}$$

param.txt

$$\begin{bmatrix} 6 & 4 & 1 & 6 \end{bmatrix}$$

XY_area.txt

$$\begin{bmatrix} \{0..0\}_6 \end{bmatrix}$$

area.txt

$$\begin{bmatrix} 2 & 2 \\ \{0..0\}_4 \end{bmatrix}$$

boards.txt

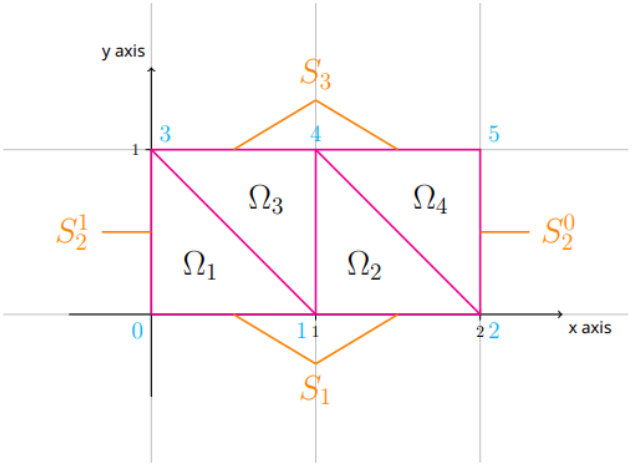
$$\begin{bmatrix} 02520 \\ 04530 \\ 03430 \\ 01210 \\ 00321 \\ 00122 \end{bmatrix}$$

Табличка с решением:

X`	X	|X` - X|	||X` - X||
0	1,8312852319637973E-18	1,831E-018	
1	1,0000000000000002	2,220E-016	
2	2,0000000000000004	4,441E-016	
0	-4,324103195733203E-18	4,324E-018	6,662E-016
1	1	0,000E+000	
2	1,9999999999999996	4,441E-016	

Данные задачи:

$$\begin{cases} u(x,y) = x \\ f(x,y) = 2x - 1 \\ \lambda = x \\ \gamma = 2 \\ \beta = 2 \\ \Omega^n, n = 1 \end{cases}$$



Краевые условия на границах:

$$\begin{cases} R_{01} = I_0 \\ R_{12} = I_0 \\ R_{25} = II_0 \\ R_{54} = III_0 \\ R_{43} = III_0 \\ R_{30} = II_1 \end{cases} \rightarrow \begin{cases} I_0 = x \\ II_0 = x \\ II_1 = -x \\ III_0 = x \end{cases}$$

Содержимое файлов:

xy.txt

$$\begin{bmatrix} (0,0) \\ (1,0) \\ (2,0) \\ (0,1) \\ (1,1) \\ (2,1) \end{bmatrix}$$

finitel.txt

$$\begin{bmatrix} (0,1,3) \\ (1,3,4) \\ (1,2,4) \\ (2,4,5) \end{bmatrix}$$

param.txt

$$\begin{bmatrix} 6 & 4 & 1 & 6 \end{bmatrix}$$

XY_area.txt

$$\begin{bmatrix} \{0..0\}_6 \end{bmatrix}$$

area.txt

$$\begin{bmatrix} 2 & 2 \\ \{0..0\}_4 \end{bmatrix}$$

boards.txt

$$\begin{bmatrix} 02520 \\ 04530 \\ 03430 \\ 01210 \\ 00321 \\ 00122 \end{bmatrix}$$

Табличка с решением:

X`	X	|X` - X|	||X` - X||
0	0	0,000E+000	
1	1,0000000000000002	2,220E-016	
2	2,0000000000000004	4,441E-016	
0	3,469446951953614E-17	3,469E-017	5,885E-016
1	1,0000000000000002	2,220E-016	
2	1,9999999999999998	2,220E-016	