|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
|  | | | |
|  | | | |
| Курсовая работа по курсу | | | |
| **«Уравнения математической физики»** | | | |
|  | | | |
|  | Группа | ПМ-05 |
|  |  |
| Студент(ка) | Лепский е.ю |
|  |  |
|
|  |
| Новосибирск | | | |

2023

1. **Постановка задачи**

Вариант: МКЭ для двумерной краевой задачи для параболического уравнения в декартовой системе координат. Базисные функции квадратичные на треугольниках. Краевые условия всех типов. Коэффициент диффузии разложить по линейным базисным функциям. Матрицу СЛАУ генерировать в разреженном строчном формате. Для решения СЛАУ использовать МСГ или ЛОС с неполной факторизацией. Схема по времени двуслойная неявная.

Уравнение

Заданно в области с границей и краевыми условиями

и начальным условием

* 1. Вариационная постановка

Будем называть пространством H - множество функций , которые вместе со своими производными до m-го порядка включительно суммируемы с квадратом на . Потребуем, чтобы невязка дифференциального уравнения была ортогональна (в смысле скалярного произведения пространства ) некоторому пространству функций , которое называется пространством пробных функций, т.е.

Применив формулу Грина к 1 слагаемому в интеграле, получим

Воспользовавшись краевыми условиями, получим

Выберем в качестве пространство пробных функций равных 0 на границе , а чтобы учесть краевые условия первого рода будем искать решение среди функций, удовлетворяющих этому условию.

* 1. Дискретизация по времени

Будем искать решение не на всем временном отрезке, а на определенных временных слоях. Значение на первом временном слое известно из начального условия.

Будем полагать, что ось t разбита на временные слои значениями . Тогда значения искомой функции

Чтобы найти решение на втором и следующих слоях применим двуслойную неявную схему:

Где

Применив схему аппроксимации по времени к вариационной постановке, получим:

Для каждого j=1,…,J получим

Где – вклады в глобальную матрицу от краевых условий 3 рода

* 1. Конечномерная дискретизация

При построении конечноэлементной аппроксимации мы будем искать проекцию искомой функции u на конечномерное пространство натянутое на кусочно-полиномиальные финитные базисные функции . В качестве пространства пробных функций возьмем то же самое подпространство (только функции на границе будут равны 0). Так как принадлежит его можно представить в виде линейной комбинации базисных функций этого пространства

Таким образом, взяв в качестве , получим

Получим СЛАУ относительно

1. **Базисные функции и - координаты**

Мы разбиваем расчетную область на конечные элементы треугольного вида и на каждом элементе вводим 6 линейных базисных функций.

Базисные функции будем выражать через - координаты

Введем на треугольнике с вершинами три линейные функции каждая из которых равна 1 в своей вершине и 0 в двух других

Так как линейная функция для нее справедливо следующее представление

Тогда справедливо следующее соотношение

Из этого соотношения очевидно, что чтобы найти коэффициенты нужно обратить матрицу

Получим базисные функции выраженные через L – координаты

1. **Локальные матрицы и вектора**

Так как область разбита по элементам, то объемные интегралы в вариационной постановке можно считать, как сумму от интегралов по элементам. Базисные функции финитные поэтому на каждом элементе интегралы с произведением базисных функций будут ненулевыми только для тех функций, которые ненулевые на данном элементе (которых 6). Таким образом глобальная матрица и вектор правой части может быть сгенерирован из локальных матриц и векторов каждого конечного элемента.

Для вычисления интегралов по треугольнику от произведения в некоторой степени использовалось следующее соотношение

Для вычисления матрицы масс мы усредняем значение на элементе и считаем интеграл по соотношению с факториалами.

Локальная матрица масс считается следующим образом:

Учитывая, что и и заменив линейным интерполянтом получим

где определяются из соотношения

1. **Учет краевых условий**

Краевые условия 2 рода учитываются следующим образом:

На ребре базисные функции будут совпадать с квадратичными одномерными лагранжевыми базисными функциями, тогда разложив по квадратичным базисным функциям получим

Аналогично с локальным вектором и матрицей третьего краевого условия

Первые краевые условия учитываются следующим образом:

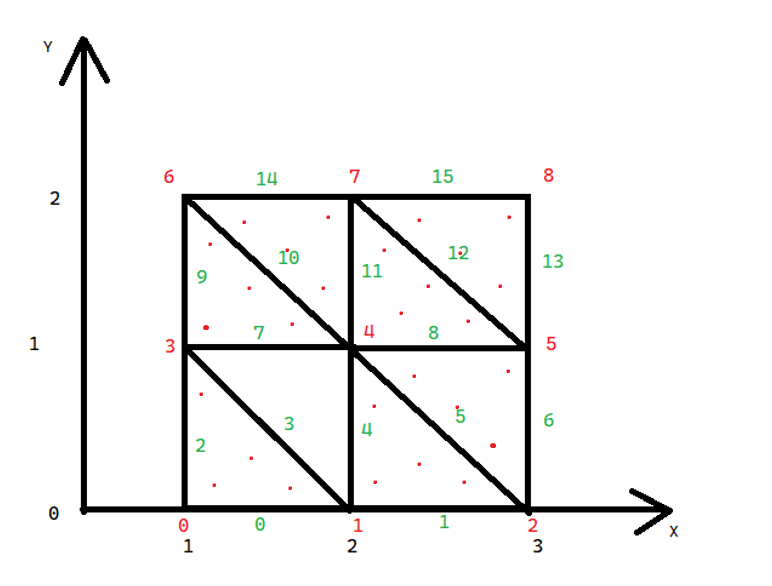
Строка, соответствующая узлу, к которому нужно применить первое краевое условие заполняется нулями, на диагонали ставится единица, а в правой части значение, которое нужно задать в этом узле. Чтобы не нарушалась симметричность матрицы так же зануляется и соответствующий столбец, но, чтобы СЛАУ не изменилась от каждой компоненты правой части отнимается зануленный элемент, умноженный на необходимое значение в узле. Таким образом при решении СЛАУ веса у функций, связанных с узлами, находящимися на границе будут удовлетворять первому краевому условию.

1. **Решение СЛАУ**

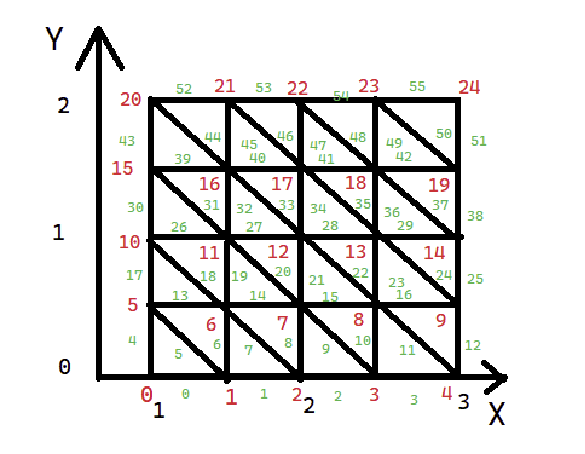
СЛАУ решается с помощью МСГ с неполным разложение Холецкого.

1. **Тестирование**

Тесты будут проводиться на этой сетке:



Вложенная сетка:



Сетка по времени

|  |
| --- |
| k |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |

1. **U=(x+y)t**

Заданы первые краевые условия слева и справа и вторые краевые условия снизу и сверху

= 1

= 1

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 1 | 3,44606E-16 |
| 2 | 1,03054E-15 |
| 3 | 4,8871E-15 |
| 4 | 0 |
| 5 | 2,06304E-15 |

1. **Тесты на порядок апроксимации по времени**

**U=()t**

Заданы первые краевые условия слева и справа и вторые краевые условия снизу и сверху

= 1

= 1

|  |  |
| --- | --- |
| t | Погрешность на слое |
| 0 | 0 |
| 1 | 3.04486E-16 |
| 2 | 2.01651E-16 |
| 3 | 1.82973E-16 |
| 4 | 2.01651E-16 |
| 5 | 0 |

**U=()**

Заданы первые краевые условия слева и справа и вторые краевые условия снизу и сверху

= 1

= 1

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 1 | 0,154088116 |
| 2 | 0,050095825 |
| 3 | 0,023742215 |
| 4 | 0,013593072 |
| 5 | 0,008743243 |

При использовании полинома второй степени получаем решение с большой погрешностью, что ожидаемо для двухслойной неявной схемы

1. **Тест на порядок сходимости по пространству**

Заданы первые краевые условия слева и справа и вторые краевые условия снизу и сверху

u =

= 1

= 1

Результат на исходной сетке:

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 1 | 0,002867661 |
| 2 | 0,002870819 |
| 3 | 0,002870819 |
| 4 | 0,002872523 |
| 5 | 0,002873569 |
| **Среднее** | **0.00239256516** |

Результат на вложенной сетке:

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 1 | 0,000443793 |
| 2 | 0,000443658 |
| 3 | 0,000443608 |
| 4 | 0,000443583 |
| 5 | 0,000443569 |
| **Среднее** | **0.00036970183** |

**Отношение погрешностей: 6.47160756548 , совпадает с ожидаемым результатом**

1. **Тест на порядок сходимости по времени**

**U=()**

Заданы первые краевые условия слева и справа и вторые краевые условия снизу и сверху

= 1

= 1

**Исходная сетка:**

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 1 | 0,154643378 |
| 2 | 0,050118888 |
| 3 | 0,0237458 |
| 4 | 0,013595182 |
| 5 | 0,008744808 |
| **Среднее** | **0.04180800933** |

**Дробление в 2 раза:**

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 0.5 | 0,238217698 |
| 1 | 0,087123996 |
| 1,5 | 0,04422823 |
| 2 | 0,026263232 |
| 2,5 | 0,017204627 |
| 3 | 0,01207067 |
| 3,5 | 0,00890867 |
| 4 | 0,006834546 |
| 4,5 | 0,00540503 |
| 5 | 0,004379849 |
| **Среднее** | **0.0227787155** |

**Дробление в 4 раза:**

|  |  |
| --- | --- |
| t | относительная погрешность |
| 0 | 0 |
| 0,25 | 0,1334864 |
| 0,5 | 0,0738537 |
| 0,75 | 0,0465972 |
| 1 | 0,0318201 |
| 1,25 | 0,0229542 |
| 1,5 | 0,0172533 |
| 1,75 | 0,0133936 |
| 2 | 0,010672462 |
| 2,25 | 8,690E-03 |
| 2,5 | 7,205E-03 |
| 2,75 | 0,006065771 |
| 3 | 0,00517476 |
| 3,25 | 0,004465265 |
| 3,5 | 0,003891549 |
| 3,75 | 0,003421289 |
| 4 | 0,00303116 |
| 4,25 | 0,002704017 |
| 4,5 | 0,002427038 |
| 4,75 | 0,017204627 |
| 5 | 0,002190494 |
| **Среднее** | **0.00881482933** |

Отношение погрешностей между первой и второй сеткой: 1.83

Отношение погрешностей между второй и третьей сеткой: 2.5

Результат совпадает с ожидаемым

**Текст программы:**

**Файл Matrix.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace course

{

public class Matrix

{

public int n;

public List<int> ia = new();

public List<int> ja = new();

public List<double> di = new();

public List<double> al = new();

public List<double> b = new();

public List<double> diLU = new();

public List<double> auLU = new();

public List<double> alLU = new();

public Matrix(int n)

{

this.n = n;

}

public void Clear()

{

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

{

di[i] = 0;

b[i] = 0;

}

for (int i = 0; i < diLU.Count; i++)

{

diLU[i] = 0;

}

int m = al.Count;

for (int i = 0; i < al.Count; i++)

{

al[i] = 0;

}

for (int i = 0; i < alLU.Count; i++)

{

alLU[i] = 0;

}

for (int i = 0; i < auLU.Count; i++)

{

auLU[i] = 0;

}

}

public void LLT()

{

if (diLU.Count == 0)

{

foreach (var item in di)

{

diLU.Add(item);

}

foreach (var item in al)

{

alLU.Add(item);

}

}

else

{

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

{

diLU[i] = di[i];

}

for (int i = 0; i < al.Count; i++)

{

alLU[i] = al[i];

}

}

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

{

double sumdi = 0;

for (int k = ia[i]; k < ia[i + 1]; k++)

{

int j = ja[k];

int j0 = ia[j];

int j1 = ia[j + 1];

int ik = ia[i];

int kj = j0;

double suml = 0.0;

while (ik < k)

{

if (ja[ik] == ja[kj])

{

suml += alLU[ik] \* alLU[kj];

ik++;

kj++;

}

else

{

if (ja[ik] > ja[kj])

{

kj++;

}

else

{

ik++;

}

}

}

alLU[k] = (alLU[k] - suml) / diLU[j];

sumdi += alLU[k] \* alLU[k];

}

if (diLU[i] - sumdi < 0)

throw new Exception();

diLU[i] = Math.Sqrt(diLU[i] - sumdi);

}

}

public List<double> GaussLLT(IReadOnlyList<double> b)

{

List<double> res = new();

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

{

res.Add(b[i]);

}

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

{

double sum = 0.0;

for (int j = ia[i]; j < ia[i + 1]; j++)

sum += alLU[j] \* res[ja[j]];

res[i] -= sum;

res[i] /= diLU[i];

}

for (int i = n - 1; i >= 0; i--)

{

double summ = 0;

for (int j = n - 1; j > i; j--)

{

int index = ja.BinarySearch(ia[j], ia[j + 1] - ia[j], i, default);

if (index > 0)

summ += alLU[index] \* res[j];

}

res[i] = (res[i] - summ) / diLU[i];

}

return res;

}

public List<double> MSGPreCond(IReadOnlyList<double> x0, double eps, int maxiter)

{

double bnorm = Math.Sqrt(DotProduct(b, b));

List<double> r = MatrixMult(x0);

List<double> z;

List<double> res = new List<double>();

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

{

res.Add(x0[i]);

r[i] = b[i] - r[i];

}

z = GaussLLT(r);

double relresidual = 1;

int k = 0;

var Mr = new List<double>();

Mr = GaussLLT(r);

while (relresidual > eps && k < maxiter)

{

var Az = MatrixMult(z);

double rsuqared = DotProduct(Mr, r);

double alpha = rsuqared / DotProduct(Az, z);

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

{

res[i] += alpha \* z[i];

r[i] -= alpha \* Az[i];

}

Mr = GaussLLT(r);

double betta = DotProduct(Mr, r) / rsuqared;

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

{

z[i] = Mr[i] + betta \* z[i];

}

relresidual = Math.Sqrt(DotProduct(r, r)) / bnorm;

k++;

}

return res;

}

double DotProduct(List<double> vec1, List<double> vec2)

{

if (vec1.Count != vec2.Count)

throw new Exception();

double res = 0;

int m = vec1.Count;

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

{

res += vec1[i] \* vec2[i];

}

return res;

}

List<double> MatrixMult(IReadOnlyList<double> x)

{

if (x.Count != n)

throw new Exception();

List<double> res = new List<double>(x.Count);

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

{

res.Add(0);

}

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

{

res[i] = x[i] \* di[i];

for (int k = ia[i]; k < ia[i + 1]; k++)

{

int j = ja[k];

res[i] += al[k] \* x[j];

res[j] += al[k] \* x[i];

}

}

return res;

}

public List<double> MSG(IReadOnlyList<double> x0, double eps, int maxiter)

{

LLT();

return MSGPreCond(x0, eps, maxiter);

}

}

}

**Файл Elem.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace course

{

public class Elem

{

public Elem(int v1, int v2, int v3, int e1, int e2, int e3, int countOfVerts)

{

this.vertex1 = v1;

this.vertex2 = v2;

this.vertex3 = v3;

this.edge1 = e1;

this.edge2 = e2;

this.edge3 = e3;

LocalToGlobal = new Dictionary<int, int>() { { 0, v1 }, { 1, v2 }, { 2, v3 },

{ 3, e1 + countOfVerts },

{ 4, e2 + countOfVerts },

{ 5, e3 + countOfVerts } };

}

public IReadOnlyDictionary<int, int> LocalToGlobal { get; init; }

public int vertex1 { get; init; }

public int vertex2 { get; init; }

public int vertex3 { get; init; }

public int edge1 { get; init; }

public int edge2 { get; init; }

public int edge3 { get; init; }

}

}

**Файл Edge.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace course

{

public class Edge

{

public Edge(int vertex1, int vertex2)

{

this.vertex1 = vertex1;

this.vertex2 = vertex2;

}

public int vertex1 { get; init; }

public int vertex2 { get; init; }

}

}

**Файл Boundary.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace course

{

public class Boundary

{

public Boundary(int edge)

{

this.e = edge;

}

public int e { get; init; }

}

public class Boundary1 : Boundary

{

public Func<double, double, double, double> ug;

public Boundary1(int edge, Func<double, double, double, double> ug) : base(edge)

{

this.ug = ug;

}

}

public class Boundary2 : Boundary

{

public Func<double, double, double, double> theta;

public Boundary2(int edge, Func<double, double, double, double> theta) : base(edge)

{

this.theta = theta;

}

}

public class Boundary3 : Boundary

{

public double betta;

public Func<double, double, double, double> ubetta;

public Boundary3(int edge, double betta, Func<double, double, double, double> ubetta) : base(edge)

{

this.betta = betta;

this.ubetta = ubetta;

}

}

}

**Файл FEM.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace course

{

public class FEM

{

Func<double, double, double, double> func;

Func<double, double, double> u0;

List<Boundary> BoundaryConditions;

Func<double, double, double> lambda;

Func<double, double, double> sigma;

List<Elem> elems;

public List<Edge> edges;

public List<double> xverts;

public List<double> yverts;

public List<List<double>> q;

Matrix mat;

public List<double> TimeMesh;

public FEM(Func<double, double, double> u0, Func<double, double, double, double> f, List<Boundary> boundaryConditions, Func<double, double, double> lambda, Func<double, double, double> sigma)

{

this.func = f;

this.u0 = u0;

BoundaryConditions = boundaryConditions;

this.lambda = lambda;

this.sigma = sigma;

elems = new();

edges = new();

xverts = new();

yverts = new();

TimeMesh = new();

q = new();

ReadMesh();

}

private void ReadMesh()

{

var str = File.ReadAllLines("txt/Verts.txt");

int vertscount = str.Count();

foreach (var item in str)

{

var cur = item.Split(' ');

xverts.Add(double.Parse(cur[0]));

yverts.Add(double.Parse(cur[1]));

}

str = File.ReadAllLines("txt/Edges.txt");

foreach (var item in str)

{

var cur = item.Split(' ');

edges.Add(new Edge(int.Parse(cur[0]), int.Parse(cur[1])));

}

str = File.ReadAllLines("txt/elems.txt");

foreach (var item in str)

{

var cur = item.Split(' ');

elems.Add(new Elem(int.Parse(cur[0]), int.Parse(cur[1]), int.Parse(cur[2]), int.Parse(cur[3]), int.Parse(cur[4]), int.Parse(cur[5]), vertscount));

}

str = File.ReadAllLines("txt/TimeMesh.txt");

foreach (var item in str)

{

TimeMesh.Add(double.Parse(item));

}

}

// Generate profile

private void GenerateProfile()

{

mat = new(xverts.Count + edges.Count());

mat.di = new List<double>(mat.n);

mat.ia = new List<int>(mat.n + 1);

mat.ja = new List<int>();

mat.al = new List<double>();

mat.b = new List<double>(mat.n);

q.Add(new());

List<SortedSet<int>> list = new List<SortedSet<int>>(mat.n);

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

{

list.Add(new SortedSet<int>());

}

/\*

\* Если между i и j узлом есть связь то list[i] будет содержать j

\*/

foreach (var item in elems)

{

// получаем глобальную нумерацию узлов и ребер

var global = item.LocalToGlobal;

// каждому элементу в глобальной нумерации, который является ключом списка ставим в соответствие остальные элементы

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

list[global[i]].Add(global[j]);

}

}

}

mat.ia.Add(0);

mat.ia.Add(0);

mat.di.Add(0);

mat.b.Add(0);

q[0].Add(0);

for (int i = 1; i < mat.n; i++)

{

q[0].Add(0);

mat.di.Add(0);

mat.b.Add(0);

int count = 0;

foreach (var item in list[i])

{

if (item < i)

{

mat.ja.Add(item);

mat.al.Add(0);

count++;

}

}

// определяем индекс в ia, с которого начнется элемент i строки

mat.ia.Add(mat.ia[i] + count);

}

foreach (var item in elems)

{

double[] u0s = new double[]

{

u0(xverts[item.vertex1],yverts[item.vertex1]),

u0(xverts[item.vertex2],yverts[item.vertex2]),

u0(xverts[item.vertex3],yverts[item.vertex3]),

u0((xverts[item.vertex3]+xverts[item.vertex2])/2,(yverts[item.vertex3]+yverts[item.vertex2])/2),

u0((xverts[item.vertex3]+xverts[item.vertex1])/2,(yverts[item.vertex3]+yverts[item.vertex1])/2),

u0((xverts[item.vertex1]+xverts[item.vertex2])/2,(yverts[item.vertex1]+yverts[item.vertex2])/2),

};

var global = item.LocalToGlobal;

for (int i = 0; i < 6; i++)

{

q[0][global[i]] = u0s[i];

}

}

}

// Create local matrix and add it to global matrix

private void AddLocal(Elem elem, int layer)

{

var d = CalcDet(elem);

double[] ax = new double[6]

{

(yverts[elem.vertex2] - yverts[elem.vertex3]) / d \* (yverts[elem.vertex2] - yverts[elem.vertex3]) / d, // L1L1

(yverts[elem.vertex3] - yverts[elem.vertex1]) / d \* (yverts[elem.vertex2] - yverts[elem.vertex3]) / d, // L1L2

(yverts[elem.vertex1] - yverts[elem.vertex2]) / d \* (yverts[elem.vertex2] - yverts[elem.vertex3]) / d, // L1L3

(yverts[elem.vertex3] - yverts[elem.vertex1]) / d \* (yverts[elem.vertex3] - yverts[elem.vertex1]) / d, // L2L2

(yverts[elem.vertex1] - yverts[elem.vertex2]) / d \* (yverts[elem.vertex3] - yverts[elem.vertex1]) / d, // L2L3

(yverts[elem.vertex1] - yverts[elem.vertex2]) / d \* (yverts[elem.vertex1] - yverts[elem.vertex2]) / d // L3L3

};

double[] ay = new double[6]

{

(xverts[elem.vertex3] - xverts[elem.vertex2]) / d \* (xverts[elem.vertex3] - xverts[elem.vertex2]) / d,

(xverts[elem.vertex1] - xverts[elem.vertex3]) / d \* (xverts[elem.vertex3] - xverts[elem.vertex2]) / d,

(xverts[elem.vertex2] - xverts[elem.vertex1]) / d \* (xverts[elem.vertex3] - xverts[elem.vertex2]) / d,

(xverts[elem.vertex1] - xverts[elem.vertex3]) / d \* (xverts[elem.vertex1] - xverts[elem.vertex3]) / d,

(xverts[elem.vertex2] - xverts[elem.vertex1]) / d \* (xverts[elem.vertex1] - xverts[elem.vertex3]) / d,

(xverts[elem.vertex2] - xverts[elem.vertex1]) / d \* (xverts[elem.vertex2] - xverts[elem.vertex1]) / d

};

d = Math.Abs(d);

double sigmaloc = sigma((xverts[elem.vertex1] + xverts[elem.vertex2] + xverts[elem.vertex3]) / 3, (yverts[elem.vertex1] + yverts[elem.vertex2] + yverts[elem.vertex3]) / 3);

double[] lambdas = new double[3]

{

lambda(xverts[elem.vertex1], yverts[elem.vertex1]),

lambda(xverts[elem.vertex2], yverts[elem.vertex2]),

lambda(xverts[elem.vertex3], yverts[elem.vertex3])

};

// значения базисных функций

double[] fs = new double[6]

{

func(xverts[elem.vertex1], yverts[elem.vertex1],TimeMesh[layer]),

func(xverts[elem.vertex2], yverts[elem.vertex2],TimeMesh[layer]),

func(xverts[elem.vertex3], yverts[elem.vertex3],TimeMesh[layer]),

func((xverts[elem.vertex2] + xverts[elem.vertex3]) / 2, (yverts[elem.vertex2] + yverts[elem.vertex3]) / 2,TimeMesh[layer]),

func((xverts[elem.vertex1] + xverts[elem.vertex3]) / 2, (yverts[elem.vertex1] + yverts[elem.vertex3]) / 2,TimeMesh[layer]),

func((xverts[elem.vertex2] + xverts[elem.vertex1]) / 2, (yverts[elem.vertex2] + yverts[elem.vertex1]) / 2,TimeMesh[layer])

};

double dt = TimeMesh[layer] - TimeMesh[layer - 1];

var global = elem.LocalToGlobal;

for (int i = 0; i < 6; i++)

{

// add to di and b global elem

for (int m = 0; m < 6; m++)

{

// add lambdas

for (int k = 0; k < 3; k++)

{

mat.di[global[i]] += (ax[m] + ay[m]) \* Matrices.GMatr[i][i][m][k] \* d \* lambdas[k];

}

mat.b[global[i]] += fs[m] \* Matrices.MMatr[i][m] \* d;

//(7.11)

mat.b[global[i]] += q[layer - 1][global[m]] \* Matrices.MMatr[i][m] \* d / dt;

}

mat.di[global[i]] += sigmaloc \* Matrices.MMatr[i][i] \* d / dt;

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

{

// string number should be large than row number

int max = global[i] > global[j] ? global[i] : global[j];

int min = global[i] > global[j] ? global[j] : global[i];

int index = mat.ja.BinarySearch(mat.ia[max], mat.ia[max + 1] - mat.ia[max], min, default);

for (int m = 0; m < 6; m++)

{

for (int k = 0; k < 3; k++)

{

mat.al[index] += lambdas[k] \* (ax[m] + ay[m]) \* Matrices.GMatr[i][j][m][k] \* d;

}

}

mat.al[index] += sigmaloc \* Matrices.MMatr[i][j] \* d / dt;

}

}

}

private void AddBoundary1(int layer)

{

foreach (var item in BoundaryConditions.OfType<Boundary1>())

{

var global = new Dictionary<int, int> { { 0, edges[item.e].vertex1 }, { 1, item.e + xverts.Count }, { 2, edges[item.e].vertex2 } };

var ug = new double[3] { item.ug(xverts[global[0]], yverts[global[0]],TimeMesh[layer]),

item.ug((xverts[global[2]] + xverts[global[0]]) / 2, (yverts[global[2]] + yverts[global[0]]) / 2, TimeMesh[layer]),

item.ug(xverts[global[2]], yverts[global[2]],TimeMesh[layer]) };

for (int m = 0; m < 3; m++)

{

// when apply 1 boundary, it need to set to 1 diagonal elements with boundary pointer

mat.di[global[m]] = 1;

mat.b[global[m]] = ug[m];

/\*

In order not to break the symmetry of the matrix, the corresponding column is also set to zero, but,

so that the SLAE does not change, the zeroed element is subtracted from each component of the right side,

multiplied by the required value in the node

\*/

for (int k = mat.ia[global[m]]; k < mat.ia[global[m] + 1]; k++)

{

mat.b[mat.ja[k]] -= ug[m] \* mat.al[k];

mat.al[k] = 0;

}

for (int i = global[m] + 1; i < mat.n; i++)

{

int index = mat.ja.BinarySearch(mat.ia[i], mat.ia[i + 1] - mat.ia[i], global[m], default);

if (index >= 0)

{

mat.b[i] -= mat.al[index] \* ug[m];

mat.al[index] = 0;

}

}

}

}

}

private void AddBoundary2(int layer)

{

foreach (var item in BoundaryConditions.OfType<Boundary2>())

{

// 0 - index of the first vertex for the boundary condition, 2 - index of the second vertex for the boundary condition, 1 - number of the node corresponding to the edge

var global = new Dictionary<int, int> { { 0, edges[item.e].vertex1 }, { 1, item.e + xverts.Count }, { 2, edges[item.e].vertex2 } };

// write values for theta, getting a specific value of x and y by index

var thetas = new double[3] {

item.theta(xverts[global[0]], yverts[global[0]],TimeMesh[layer]),

item.theta((xverts[global[2]] + xverts[global[0]]) / 2, (yverts[global[2]] + yverts[global[0]]) / 2,TimeMesh[layer]),

item.theta(xverts[global[2]], yverts[global[2]],TimeMesh[layer])

};

// длина ребра

double len = Math.Sqrt((xverts[global[2]] - xverts[global[0]]) \* (xverts[global[2]] - xverts[global[0]]) +

(yverts[global[2]] - yverts[global[0]]) \* (yverts[global[2]] - yverts[global[0]]));

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

mat.b[global[i]] += thetas[j] \* Matrices.BCMatr[i][j] \* len;

}

}

}

}

private void AddBoundary3(int layer)

{

foreach (var item in BoundaryConditions.OfType<Boundary3>())

{

var global = new Dictionary<int, int> { { 0, edges[item.e].vertex1 }, { 1, item.e + xverts.Count }, { 2, edges[item.e].vertex2 } };

var ubetta = new double[3] {

item.ubetta(xverts[global[0]], yverts[global[0]],TimeMesh[layer]),

item.ubetta((xverts[global[2]] + xverts[global[0]]) / 2, (yverts[global[2]] + yverts[global[0]]) / 2,TimeMesh[layer]),

item.ubetta(xverts[global[2]], yverts[global[2]],TimeMesh[layer])

};

// длина ребра

double len = Math.Sqrt((xverts[global[2]] - xverts[global[0]]) \* (xverts[global[2]] - xverts[global[0]]) +

(yverts[global[2]] - yverts[global[0]]) \* (yverts[global[2]] - yverts[global[0]]));

for (int i = 0; i < 3; i++)

{

mat.di[global[i]] += Matrices.BCMatr[i][i] \* item.betta \* len;

for (int j = 0; j < 3; j++)

{

mat.b[global[i]] += ubetta[j] \* Matrices.BCMatr[i][j] \* len \* item.betta;

}

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

{

int max = global[i] > global[j] ? global[i] : global[j];

int min = global[i] > global[j] ? global[j] : global[i];

int index = mat.ja.BinarySearch(mat.ia[max], mat.ia[max + 1] - mat.ia[max], min, default);

mat.al[index] += Matrices.BCMatr[i][j] \* item.betta \* len;

}

}

}

}

// Create profile, Add all local items to global, add all boundary conditions and get q

public void Solve(double eps, int maxiter)

{

GenerateProfile();

for (int i = 1; i < TimeMesh.Count; i++)

{

foreach (var item in elems)

{

AddLocal(item, i);

}

AddBoundary2(i);

AddBoundary3(i);

AddBoundary1(i);

q.Add(mat.MSG(q[i - 1], eps, maxiter));

mat.Clear();

}

}

// Get sollution in specific point

public double Sollution(double x, double y, int layer)

{

double res = 0;

int indexOfElem = searchElem(x, y);

var global = elems[indexOfElem].LocalToGlobal;

var L1 = getL1(x, y, elems[indexOfElem]);

var L2 = getL2(x, y, elems[indexOfElem]);

var L3 = getL3(x, y, elems[indexOfElem]);

res += L1 \* (2 \* L1 - 1) \* q[layer][global[0]];

res += L2 \* (2 \* L2 - 1) \* q[layer][global[1]];

res += L3 \* (2 \* L3 - 1) \* q[layer][global[2]];

res += 4 \* L2 \* L3 \* q[layer][global[3]];

res += 4 \* L1 \* L3 \* q[layer][global[4]];

res += 4 \* L1 \* L2 \* q[layer][global[5]];

return res;

}

// calculate finit element determinate

public double CalcDet(Elem elem)

{

return (xverts[elem.vertex2] - xverts[elem.vertex1]) \* (yverts[elem.vertex3] - yverts[elem.vertex1]) - (xverts[elem.vertex3] - xverts[elem.vertex1]) \* (yverts[elem.vertex2] - yverts[elem.vertex1]);

}

private int searchElem(double x, double y)

{

bool flag = true;

int i = 0;

for (i = 0; i < elems.Count && flag; i++)

{

double S23 = Math.Abs((xverts[elems[i].vertex3] - xverts[elems[i].vertex2]) \* (y - yverts[elems[i].vertex2]) - (x - xverts[elems[i].vertex2]) \* (yverts[elems[i].vertex3] - yverts[elems[i].vertex2]));

double S31 = Math.Abs((xverts[elems[i].vertex1] - xverts[elems[i].vertex3]) \* (y - yverts[elems[i].vertex3]) - (x - xverts[elems[i].vertex3]) \* (yverts[elems[i].vertex1] - yverts[elems[i].vertex3]));

double S12 = Math.Abs((xverts[elems[i].vertex2] - xverts[elems[i].vertex1]) \* (y - yverts[elems[i].vertex1]) - (x - xverts[elems[i].vertex1]) \* (yverts[elems[i].vertex2] - yverts[elems[i].vertex1]));

// if point outside mesh

if (Math.Abs(Math.Abs(CalcDet(elems[i])) - (S23 + S31 + S12)) <= 1e-7)

flag = false;

}

i--;

if (flag)

throw new Exception();

return i;

}

public double getL1(double x, double y, Elem elem)

{

return (xverts[elem.vertex2] \* yverts[elem.vertex3] - xverts[elem.vertex3] \* yverts[elem.vertex2] + x \* (yverts[elem.vertex2] - yverts[elem.vertex3]) + y \* (xverts[elem.vertex3] - xverts[elem.vertex2])) / CalcDet(elem);

}

public double getL2(double x, double y, Elem elem)

{

return ((xverts[elem.vertex3] \* yverts[elem.vertex1] - xverts[elem.vertex1] \* yverts[elem.vertex3] + x \* (yverts[elem.vertex3] - yverts[elem.vertex1]) + y \* (xverts[elem.vertex1] - xverts[elem.vertex3]))) / CalcDet(elem);

}

public double getL3(double x, double y, Elem elem)

{

return ((xverts[elem.vertex1] \* yverts[elem.vertex2] - xverts[elem.vertex2] \* yverts[elem.vertex1] + x \* (yverts[elem.vertex1] - yverts[elem.vertex2]) + y \* (xverts[elem.vertex2] - xverts[elem.vertex1]))) / CalcDet(elem);

}

static class Matrices

{

public static double[][] MMatr = new double[6][]

{

new double[6] {0.01666666666666665, -0.002777777777777775, -0.002777777777777775, -0.01111111111111111, 0, 0, },

new double[6] {-0.002777777777777775, 0.01666666666666665, -0.002777777777777775, 0, -0.01111111111111111, 0, },

new double[6]{ -0.002777777777777775, -0.002777777777777775, 0.01666666666666665, 0, 0, -0.01111111111111111, },

new double[6]{ -0.01111111111111111, 0, 0, 0.08888888888888889, 0.044444444444444446, 0.044444444444444446, },

new double[6]{ 0, -0.01111111111111111, 0, 0.044444444444444446, 0.08888888888888889, 0.044444444444444446, },

new double[6]{ 0, 0, -0.01111111111111111, 0.044444444444444446, 0.044444444444444446, 0.08888888888888889, },

};

// 1 - i elem, 2 - j elem, 3 - which L coordinates use, 4 - decompo-sition of the diffusion coefficient

public static double[][][][] GMatr = new double[6][][][]

{

**new double[6][][]**

**{**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.30000000000000004, 0.1, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.06666666666666665, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.033333333333333326, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, -0.033333333333333326, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0.1, -0.06666666666666665, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.1, -0.033333333333333326, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.46666666666666673, 0.1, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.1, -0.06666666666666665, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0.46666666666666673, 0.1, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**},**

**new double[6][][]**

**{**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.06666666666666665, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.30000000000000004, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, -0.06666666666666665, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, 0.1, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0.1, 0.46666666666666673, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, 0.1, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, 0.1, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.46666666666666673, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, 0.1, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**},**

**new double[6][][]**

**{**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.033333333333333326, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, -0.06666666666666665, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.1, 0.30000000000000004,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.1, 0.46666666666666673,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, -0.06666666666666665, 0.1,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.1, 0.46666666666666673,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.033333333333333326, 0.1,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, -0.06666666666666665, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.033333333333333326, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**},**

**new double[6][][]**

**{**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, -0.033333333333333326, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0.1, -0.06666666666666665, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, 0.1, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0.1, 0.46666666666666673, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.1, 0.46666666666666673,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, -0.06666666666666665, 0.1,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.8,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.5333333333333333, 0.5333333333333333,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.8, 0.26666666666666666,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.8,**

**},**

**new double[3]**

**{**

**0.13333333333333333, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.13333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.13333333333333333,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.13333333333333333, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.8, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.13333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.13333333333333333,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**},**

**new double[6][][]**

**{**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.1, -0.033333333333333326, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.46666666666666673, 0.1, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, 0.1, -0.06666666666666665,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, 0.1, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.1, 0.46666666666666673,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.033333333333333326, 0.1,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.8,**

**},**

**new double[3]**

**{**

**0.13333333333333333, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.13333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.13333333333333333,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.8,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.5333333333333333, 0.26666666666666666, 0.5333333333333333,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.8, 0.26666666666666666, 0.26666666666666666,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.13333333333333333, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.13333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.13333333333333333,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.8, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**},**

**new double[6][][]**

**{**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.1, -0.06666666666666665, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0.46666666666666673, 0.1, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.1, 0.46666666666666673, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, 0.1, -0.033333333333333326,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.033333333333333326, -0.06666666666666665, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**-0.06666666666666665, -0.033333333333333326, 0.1,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.13333333333333333, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.8, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.13333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.13333333333333333,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.13333333333333333, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.13333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.26666666666666666, 0.26666666666666666, 0.13333333333333333,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.8, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**new double[6][]**

**{**

**new double[3]**

**{**

**0.26666666666666666, 0.8, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0.5333333333333333, 0.5333333333333333, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0.8, 0.26666666666666666, 0.26666666666666666,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**new double[3]**

**{**

**0, 0, 0,**

**},**

**},**

**},**

**};**

**public static double[][] BCMatr = new double[3][]**

**{**

**new double[3]**

**{**

**4.0/30,2.0/30,-1.0/30**

**},**

**new double[3]**

**{**

**2.0/30,16.0/30,2.0/30**

**},**

**new double[3]**

**{**

**-1.0/30,2.0/30,4.0/30**

**}**

**};**

**}**

**}**

**}**

**Файл Program.cs:**

using System.IO;

using System.Threading.Channels;

namespace course;

class Program

{

static void Main(string[] args)

{

Func<double, double, double, double> u = (x, y, t) => (x+y)\*t;

Func<double, double, double, double> f = (x, y, t) => x+ y;

Func<double, double, double> lambda = (x, y) => 1;

Func<double, double, double> sigma = (x, y) => 1;

Func<double, double, double> u0 = (x, y) => 0;

List<Boundary> boundaryCouditions = new();

/\*boundaryCouditions.Add(new Boundary1(2, u));

boundaryCouditions.Add(new Boundary1(9, u));

boundaryCouditions.Add(new Boundary1(6, u));

boundaryCouditions.Add(new Boundary1(13, u));

boundaryCouditions.Add(new Boundary2(0, (x, y, t) => -2\*y\*t\*t));

boundaryCouditions.Add(new Boundary2(1, (x, y, t) => -2\*y\*t\*t));

boundaryCouditions.Add(new Boundary2(14, (x, y, t) => 2\*y\*t\*t));

boundaryCouditions.Add(new Boundary2(15, (x, y, t) => 2\*y\*t\*t));\*/

/\*Func<double, double, double, double> u = (x, y, t) => x \* t + y \* t;

Func<double, double, double, double> f = (x, y, t) => x + y;

Func<double, double, double> lambda = (x, y) => 1;

Func<double, double, double> sigma = (x, y) => 1;

Func<double, double, double> u0 = (x, y) => 0;

List<Boundary> boundaryCouditions = new();

boundaryCouditions.Add(new Boundary1(2, u));

boundaryCouditions.Add(new Boundary1(9, u));

boundaryCouditions.Add(new Boundary1(6, u));

boundaryCouditions.Add(new Boundary1(13, u));

boundaryCouditions.Add(new Boundary2(0, (x, y, t) => -1));

boundaryCouditions.Add(new Boundary2(1, (x, y, t) => -1));

boundaryCouditions.Add(new Boundary2(14, (x, y, t) => 1));

boundaryCouditions.Add(new Boundary2(15, (x, y, t) => 1));\*/

boundaryCouditions.Add(new Boundary1(2, u));

boundaryCouditions.Add(new Boundary1(9, u));

boundaryCouditions.Add(new Boundary1(13, u));

boundaryCouditions.Add(new Boundary1(6, u));

/\*boundaryCouditions.Add(new Boundary1(12, u));

boundaryCouditions.Add(new Boundary1(25, u));

boundaryCouditions.Add(new Boundary1(38, u));

boundaryCouditions.Add(new Boundary1(51, u));\*/

boundaryCouditions.Add(new Boundary2(0, (x, y, t) => -t));

boundaryCouditions.Add(new Boundary2(1, (x, y, t) => -t));

boundaryCouditions.Add(new Boundary2(14, (x, y, t) => t));

boundaryCouditions.Add(new Boundary2(15, (x, y, t) => t));

/\*boundaryCouditions.Add(new Boundary2(52, (x, y, t) => 0 ));

boundaryCouditions.Add(new Boundary2(53, (x, y, t) => 0));

boundaryCouditions.Add(new Boundary2(54, (x, y, t) => 0));

boundaryCouditions.Add(new Boundary2(55, (x, y, t) => 0));\*/

FEM fem = new FEM(u0, f, boundaryCouditions, lambda, sigma);

fem.Solve(1e-15, 10000);

//Console.WriteLine($"{fem.Getsollution(1, 1)} {fem.Getsollution(2, 1)} {fem.Getsollution(3, 1)}");

double[] err = new double[fem.q.Count];

using (StreamWriter writer = new StreamWriter("./data.txt"))

// добавление в файл

{

for (int l = 0; l < fem.q.Count; l++)

{

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

writer.WriteLineAsync($"{fem.Sollution(j+1, i, l)}\t{u(j+1, i, fem.TimeMesh[l])}\t{Math.Abs(u(j + 1, i, fem.TimeMesh[l]) - fem.Sollution(j + 1, i, l))}");

Console.WriteLine($"{fem.Sollution(j + 1, i, l)}\t{u(j + 1, i, fem.TimeMesh[l])}\t{Math.Abs(u(j + 1, i, fem.TimeMesh[l]) - fem.Sollution(j + 1, i, l))}");

}

}

writer.WriteLineAsync($"\n");

Console.WriteLine("-----------------------");

}

}

}

}