|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Курсовая работа | | |
| по дисциплине «Численные методы» | | |
| **Метод конечных элементов** | | |
|  | | |
|  | Группа |  |
| Студент |  |
| Преподаватель |  |
| Дата | 28.12.2021 |
|  |  |
|  |  |
|  |  |
|  |  |
| Новосибирск, 2023 | | |

1. Постановка задачи
   1. Уравнение

Заданное в области с границей и краевыми условиями

Вариант: МКЭ для двумерной краевой задачи для эллиптического 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.

* 1. Вариационная постановка

Будем называть пространством Hm- 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

Чтобы избавиться от интеграла 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.

Таким образом получим

* 1. Конечномерная дискретизация

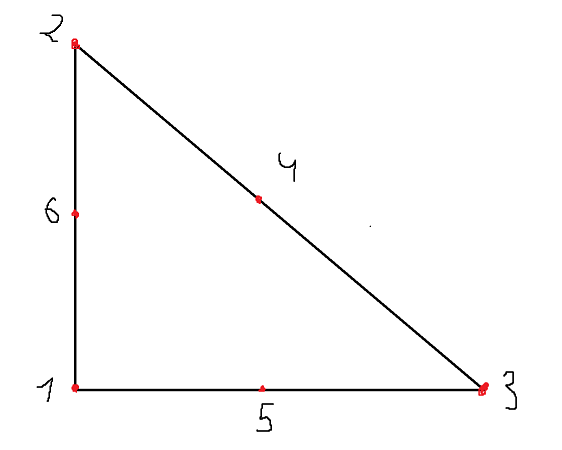
При построении конечноэлементной аппроксимации мы будем искать проекцию искомой функции u на конечномерное пространство натянутое на кусочно-полиномиальные финитные базисные функции . В качестве пространства пробных функций возьмем то же самое подпространство (только функции на границе будут равны 0). 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 3 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 r z для нее справедливо следующее представление

Тогда справедливо следующее соотношение

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

Теперь выпишем 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:

И заменив линейным интерполянтом получим

где определяются из соотношения

Локальный вектор правой части вычисляется следующим образом: f заменяется своим квадратичным интерполянтом и интегрируется получается выражение похожее на умножение локальной матрицы масс на вектор значений f в точках

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:

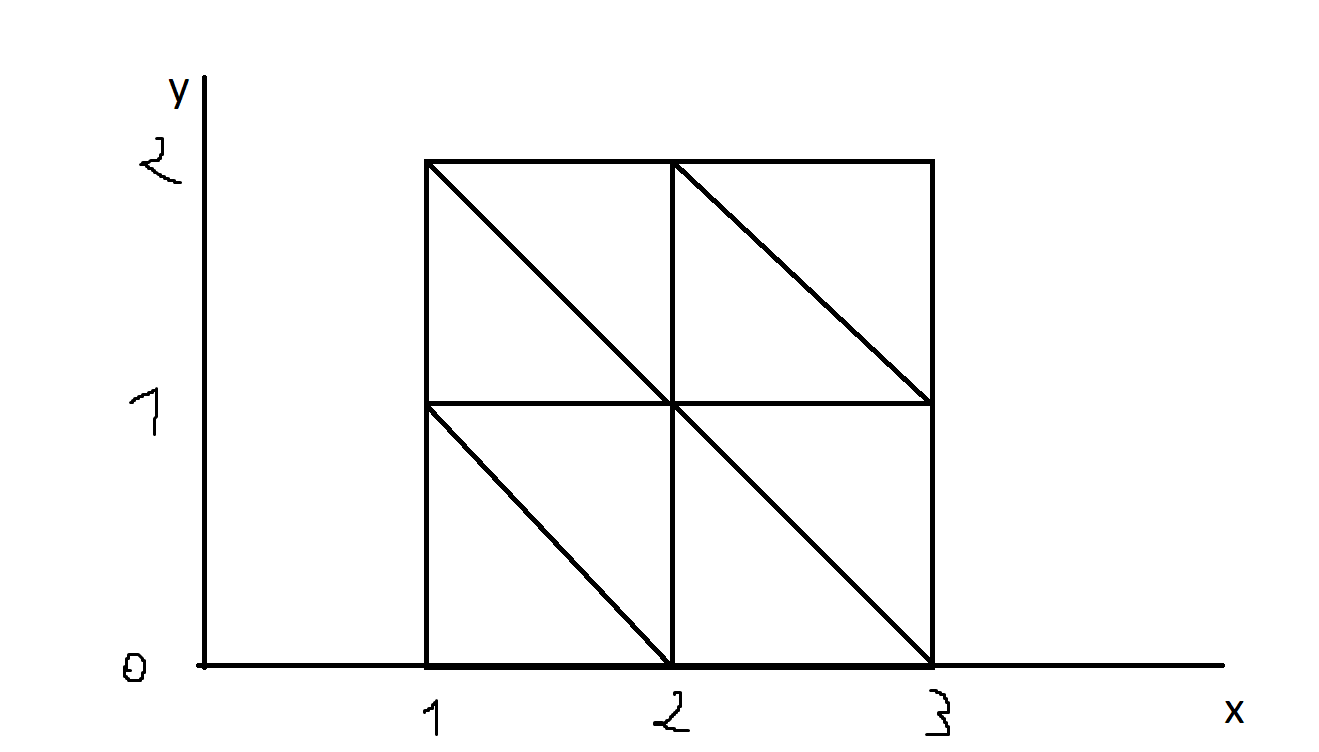
Строка, соответствующая узлу, к которому нужно применить первое краевое условие заполняется нулями, на диагонали ставится единица, а в правой части значение, которое нужно задать в этом узле. Чтобы не нарушалась симметричность матрицы так же зануляется и соответствующий столбец, но, чтобы СЛАУ не изменилась 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 ЛОСа с LU предобуславливанием .

1. Тестирование

Тесты будут проводиться на этой сетке:



Чтобы оценивать точность будем смотреть относительную погрешность вектора полученного вектора весов q относительно истинного значения.

* 1. Тест 1

u = x

= 1

= 0

Заданы первые краевые на боковых ребрах и вторые на верхнем и нижнем

Относительная погрешность = 0

* 1. Тест 2

u = x

= 1

= 0

Заданы вторые краевые на левом ребре и третье краевое на правом и вторые на верхнем и нижнем

Относительная погрешность = 9.4372E-15

* 1. Тест 3

u =

= 1

= 1

Заданы вторые краевые на левом ребре и третье краевое на правом и вторые на верхнем и нижнем

Относительная погрешность = 6.88556E-15

* 1. Тест 4

u =

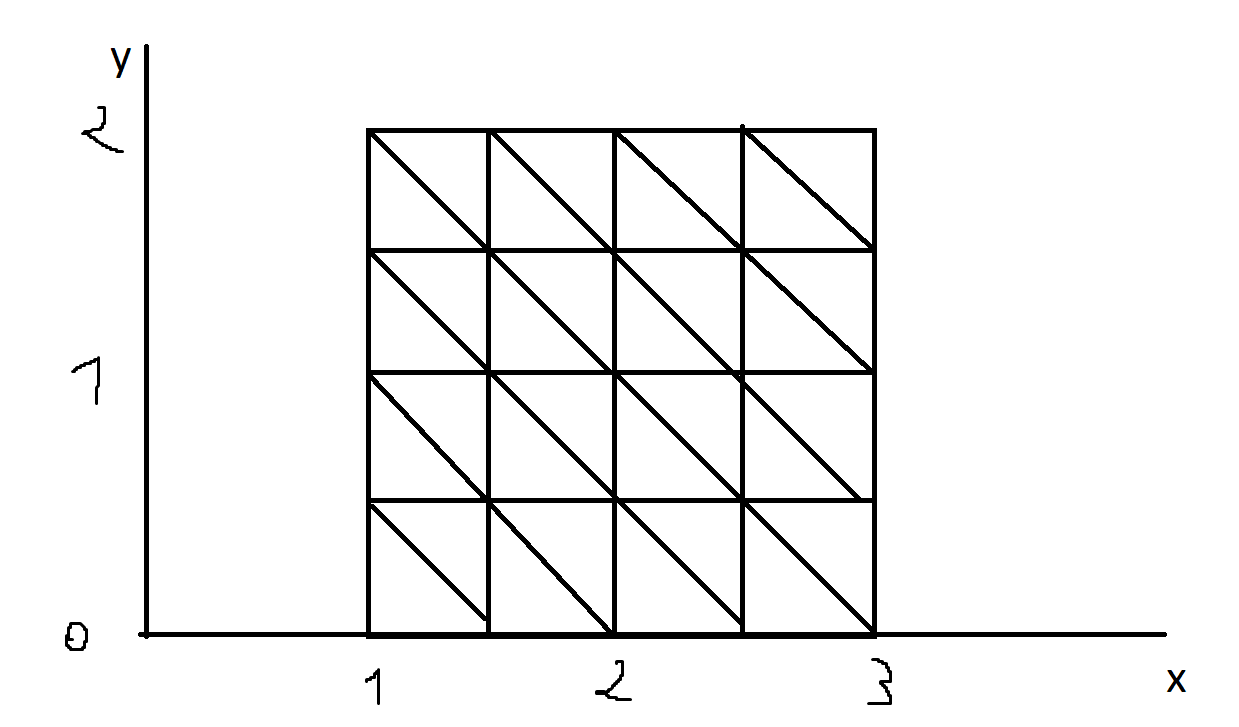
= 1

Заданы первые краевые на боковых ребрах и вторые на верхнем и нижнем

Относительная погрешность = 7.16825E-15

* 1. Тест на порядок сходимости

Для теста на сходимость понадобится вложенная сетка



Будем смотреть на значения полученной функции в серединах элементов вложенной сетки и относительную погрешность.

u =

= 0

Заданы первые краевые на боковых ребрах и вторые на верхнем и нижнем

Относительная погрешность на перовой сетке = 0.002625475

Относительная погрешность на вложенной сетке = 0.000389992

Отношение этих погрешностей =6.73

Порядок сходимости совпадает с теоретическим третьим

1. Текст программы

Файл Matrix.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KursEgor

{

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> di\_LU = new();

public List<double> au\_LU = new();

public List<double> al\_LU = new();

public Matrix(int n)

{

this.n = n;

}

public void Clear()

{

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

{

di[i] = 0;

di\_LU[i] = 0;

b[i] = 0;

}

int m = al.Count;

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

{

al[i] = 0;

al\_LU[i] = 0;

au\_LU[i] = 0;

}

}

public void LU()

{

foreach (var item in di)

{

di\_LU.Add(item);

}

foreach (var item in al)

{

au\_LU.Add(item);

al\_LU.Add(item);

}

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

{

di\_LU[i] = di[i];

}

var q = al.Count;

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

{

al\_LU[i] = al[i];

au\_LU[i] = al[i];

}

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

{

double sumdi = 0.0;

int i0 = ia[i];

int i1 = ia[i + 1];

for (int k = i0; k < i1; k++)

{

int j = ja[k];

int j0 = ia[j];

int j1 = ia[j + 1];

int ik = i0;

int kj = j0;

double suml = 0.0;

double sumu = 0.0;

while (ik < k)

{

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

{

suml += al\_LU[ik] \* au\_LU[kj];

sumu += au\_LU[ik] \* al\_LU[kj];

ik++;

kj++;

}

else

{

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

{

kj++;

}

else

{

ik++;

}

}

}

al\_LU[k] = al\_LU[k] - suml;

au\_LU[k] = (au\_LU[k] - sumu) / di\_LU[j];

sumdi += al\_LU[k] \* au\_LU[k];

}

di\_LU[i] = di\_LU[i] - sumdi;

}

}

List<double> LUDirect(List<double> rpart)

{

List<double> res = new();

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

{

res.Add(rpart[i]);

}

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

{

double sum = 0.0;

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

sum += al\_LU[j] \* res[ja[j]];

res[i] -= sum;

res[i] /= di\_LU[i];

}

return res;

}

List<double> LUReverse(List<double> rpart)

{

List<double> res = new();

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

{

res.Add(rpart[i]);

}

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

{

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

res[ja[j]] -= au\_LU[j] \* res[i];

}

return res;

}

public List<double> LoS\_precond(List<double> x0, double eps, int maxiter)

{

int k = 1;

List<double> buf = MatrixMult(x0);

double bnorm = 0;

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

{

buf[i] = b[i] - buf[i];

}

double rnorm = Math.Sqrt(DotProduct(buf, buf));

List<double> r = LUDirect(buf);

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

List<double> z = LUReverse(r);

buf = MatrixMult(z);

List<double> p = LUDirect(buf);

double resid = 1;

while (resid > eps && k < maxiter)

{

double pp = DotProduct(p, p);

double pr = DotProduct(p, r);

double alpha = pr / pp;

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

{

x0[i] += alpha \* z[i];

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

}

rnorm = Math.Sqrt(DotProduct(r, r));

List<double> Ur = LUReverse(r);

buf = MatrixMult(Ur);

buf = LUDirect(buf);

double betta = -(DotProduct(p, buf) / pp);

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

{

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

p[i] = buf[i] + betta \* p[i];

}

double test1 = 0;

double test2 = 0;

var asd = MatrixMult(x0);

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

{

test1 += (asd[i] - b[i]) \* (asd[i] - b[i]);

test2 += b[i] \* b[i];

}

resid = Math.Sqrt(test1 / test2);

k++;

}

//Console.WriteLine($"{k} {rnorm / bnorm} {resid}");

return x0;

}

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

{

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

List<double> Ar = new();

List<double> r = MatrixMult(x0);

List<double> z = new();

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

{

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

z.Add(r[i]);

}

List<double> p = MatrixMult(z);

int k = 0;

double alpha, betta, rnorm = Math.Sqrt(DotProduct(r, r));

while (k < maxiter && rnorm / bnorm > eps)

{

alpha = DotProduct(p, r) / DotProduct(p, p);

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

{

x0[i] += alpha \* z[i];

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

}

Ar = MatrixMult(r);

betta = -DotProduct(p, Ar) / DotProduct(p, p);

rnorm = Math.Sqrt(DotProduct(r, r));

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

{

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

p[i] = Ar[i] + betta \* p[i];

}

k++;

}

Console.WriteLine($"{rnorm / bnorm} {k}");

return x0;

}

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(List<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> SolveLosLUPrecond(double eps,int maxiter)

{

List<double> x0 = new();

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

{

x0.Add(0);

}

LU();

return LoS\_precond(x0, eps, maxiter);

}

}

}

Файл Elem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KursEgor

{

public class Elem

{

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

{

this.v1 = v1;

this.v2 = v2;

this.v3 = v3;

this.e1 = e1;

this.e2 = e2;

this.e3 = 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 v1 { get; init; }

public int v2 { get; init; }

public int v3 { get; init; }

public int e1 { get; init; }

public int e2 { get; init; }

public int e3 { get; init; }

}

}

Файл BoundaryCondition.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KursEgor

{

public class BoundaryCondition

{

public BoundaryCondition(int edge)

{

this.e = edge;

}

public int e { get; init; }

}

public class BoundaryCondition1 : BoundaryCondition

{

public Func<double, double, double> ug;

public BoundaryCondition1(int edge, Func<double, double, double> ug) : base(edge)

{

this.ug = ug;

}

}

public class BoundaryCondition2 : BoundaryCondition

{

public Func<double, double, double> theta;

public BoundaryCondition2(int edge, Func<double, double, double> theta) : base(edge)

{

this.theta = theta;

}

}

public class BoundaryCondition3 : BoundaryCondition

{

public double betta;

public Func<double, double, double> ubetta;

public BoundaryCondition3(int edge, double betta, Func<double, double, double> ubetta) : base(edge)

{

this.betta = betta;

this.ubetta = ubetta;

}

}

}

Файл Edge.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KursEgor

{

public class Edge

{

public Edge(int v1, int v2)

{

this.v1 = v1;

this.v2 = v2;

}

public int v1 { get; init; }

public int v2 { get; init; }

}

}

Файл FEM.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KursEgor

{

public class FEM

{

Func<double, double, double> f;

List<BoundaryCondition> BoundaryConditions;

Func<double, double, double> lambda;

Func<double, double, double> gamma;

List<Elem> elems;

List<Edge> edges;

public List<double> xverts;

public List<double> yverts;

public List<double> q;

Matrix mat;

public FEM(Func<double, double, double> f, List<BoundaryCondition> boundaryConditions, Func<double, double, double> lambda, Func<double, double, double> gamma)

{

this.f = f;

BoundaryConditions = boundaryConditions;

this.lambda = lambda;

this.gamma = gamma;

elems = new();

edges = new();

xverts = new();

yverts = 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));

}

}

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);

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

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

{

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

}

foreach (var item in elems)

{

var dic = item.LocalToGlobal;

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

{

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

{

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

}

}

}

mat.ia.Add(0);

mat.ia.Add(0);

mat.di.Add(0);

mat.b.Add(0);

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

{

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++;

}

}

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

}

}

private void AddLocal(Elem elem)

{

var d = CalcDet(elem);

double[] ax = new double[6]//11 12 13 22 32 33

{

(yverts[elem.v2] - yverts[elem.v3]) / d \* (yverts[elem.v2] - yverts[elem.v3]) / d,

(yverts[elem.v3] - yverts[elem.v1]) / d \* (yverts[elem.v2] - yverts[elem.v3]) / d,

(yverts[elem.v1] - yverts[elem.v2]) / d \* (yverts[elem.v2] - yverts[elem.v3]) / d,

(yverts[elem.v3] - yverts[elem.v1]) / d \* (yverts[elem.v3] - yverts[elem.v1]) / d,

(yverts[elem.v1] - yverts[elem.v2]) / d \* (yverts[elem.v3] - yverts[elem.v1]) / d,

(yverts[elem.v1] - yverts[elem.v2]) / d \* (yverts[elem.v1] - yverts[elem.v2]) / d

};

double[] ay = new double[6]

{

(xverts[elem.v3] - xverts[elem.v2]) / d \* (xverts[elem.v3] - xverts[elem.v2]) / d,

(xverts[elem.v1] - xverts[elem.v3]) / d \* (xverts[elem.v3] - xverts[elem.v2]) / d,

(xverts[elem.v2] - xverts[elem.v1]) / d \* (xverts[elem.v3] - xverts[elem.v2]) / d,

(xverts[elem.v1] - xverts[elem.v3]) / d \* (xverts[elem.v1] - xverts[elem.v3]) / d,

(xverts[elem.v2] - xverts[elem.v1]) / d \* (xverts[elem.v1] - xverts[elem.v3]) / d,

(xverts[elem.v2] - xverts[elem.v1]) / d \* (xverts[elem.v2] - xverts[elem.v1]) / d

};

d = Math.Abs(d);

double gammaloc = gamma((xverts[elem.v1] + xverts[elem.v2] + xverts[elem.v3]) / 3, (yverts[elem.v1] + yverts[elem.v2] + yverts[elem.v3]) / 3);

double[] lambdas = new double[3]

{

lambda(xverts[elem.v1], yverts[elem.v1]),

lambda(xverts[elem.v2], yverts[elem.v2]),

lambda(xverts[elem.v3], yverts[elem.v3])

};

double[] fs = new double[6]

{

f(xverts[elem.v1], yverts[elem.v1]),

f(xverts[elem.v2], yverts[elem.v2]),

f(xverts[elem.v3], yverts[elem.v3]),

f((xverts[elem.v2] + xverts[elem.v3]) / 2, (yverts[elem.v2] + yverts[elem.v3]) / 2),

f((xverts[elem.v1] + xverts[elem.v3]) / 2, (yverts[elem.v1] + yverts[elem.v3]) / 2),

f((xverts[elem.v2] + xverts[elem.v1]) / 2, (yverts[elem.v2] + yverts[elem.v1]) / 2)

};

var dic = elem.LocalToGlobal;

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

{

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

{

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

{

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

}

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

}

mat.di[dic[i]] += Matrices.MMatr[i][i] \* d \* gammaloc;

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

{

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

int min = dic[i] > dic[j] ? dic[j] : dic[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] += gammaloc \* Matrices.MMatr[i][j] \* d;

}

}

}

private void AddBoundary1()

{

foreach (var item in BoundaryConditions.OfType<BoundaryCondition1>())

{

var dic = new Dictionary<int, int> { { 0, edges[item.e].v1 }, { 1, item.e + xverts.Count }, { 2, edges[item.e].v2 } };

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

item.ug((xverts[dic[2]] + xverts[dic[0]]) / 2, (yverts[dic[2]] + yverts[dic[0]]) / 2),

item.ug(xverts[dic[2]], yverts[dic[2]]) };

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

{

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

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

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

{

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

mat.al[k] = 0;

}

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

{

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

if (index >= 0)

{

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

mat.al[index] = 0;

}

}

}

}

}

private void AddBoundary2()

{

foreach (var item in BoundaryConditions.OfType<BoundaryCondition2>())

{

var dic = new Dictionary<int, int> { { 0, edges[item.e].v1 }, { 1, item.e + xverts.Count }, { 2, edges[item.e].v2 } };

var thetas = new double[3] { item.theta(xverts[dic[0]], yverts[dic[0]]),

item.theta((xverts[dic[2]] + xverts[dic[0]]) / 2, (yverts[dic[2]] + yverts[dic[0]]) / 2),

item.theta(xverts[dic[2]], yverts[dic[2]]) };

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

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

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

{

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

{

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

}

}

}

}

private void AddBoundary3()

{

foreach (var item in BoundaryConditions.OfType<BoundaryCondition3>())

{

var dic = new Dictionary<int, int> { { 0, edges[item.e].v1 }, { 1, item.e + xverts.Count }, { 2, edges[item.e].v2 } };

var ubetta = new double[3] { item.ubetta(xverts[dic[0]], yverts[dic[0]]),

item.ubetta((xverts[dic[2]] + xverts[dic[0]]) / 2, (yverts[dic[2]] + yverts[dic[0]]) / 2),

item.ubetta(xverts[dic[2]], yverts[dic[2]]) };

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

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

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

{

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

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

{

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

}

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

{

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

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

}

}

}

}

public void Solve(double eps, int maxiter)

{

GenerateProfile();

foreach (var item in elems)

{

AddLocal(item);

}

AddBoundary2();

AddBoundary3();

AddBoundary1();

q = mat.SolveLosLUPrecond(eps, maxiter);

}

public double Getsollution(double x, double y)

{

double res = 0;

bool flag = true;

int i = 0;

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

{

double S23 = Math.Abs((xverts[elems[i].v3] - xverts[elems[i].v2]) \* (y - yverts[elems[i].v2]) - (x - xverts[elems[i].v2]) \* (yverts[elems[i].v3] - yverts[elems[i].v2]));

double S31 = Math.Abs((xverts[elems[i].v1] - xverts[elems[i].v3]) \* (y - yverts[elems[i].v3]) - (x - xverts[elems[i].v3]) \* (yverts[elems[i].v1] - yverts[elems[i].v3]));

double S12 = Math.Abs((xverts[elems[i].v2] - xverts[elems[i].v1]) \* (y - yverts[elems[i].v1]) - (x - xverts[elems[i].v1]) \* (yverts[elems[i].v2] - yverts[elems[i].v1]));

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

flag = false;

}

i--;

if (flag)

throw new Exception();

var dic = elems[i].LocalToGlobal;

var L = XtoLTransform(x, y, elems[i]);

res += L.L1 \* (2 \* L.L1 - 1) \* q[dic[0]];

res += L.L2 \* (2 \* L.L2 - 1) \* q[dic[1]];

res += L.L3 \* (2 \* L.L3 - 1) \* q[dic[2]];

res += 4 \* L.L2 \* L.L3 \* q[dic[3]];

res += 4 \* L.L1 \* L.L3 \* q[dic[4]];

res += 4 \* L.L1 \* L.L2 \* q[dic[5]];

return res;

}

public double CalcDet(Elem el)

{

return (xverts[el.v2] - xverts[el.v1]) \* (yverts[el.v3] - yverts[el.v1]) - (xverts[el.v3] - xverts[el.v1]) \* (yverts[el.v2] - yverts[el.v1]);

}

public (double L1, double L2, double L3) XtoLTransform(double x, double z, Elem el)

{

return ((xverts[el.v2] \* yverts[el.v3] - xverts[el.v3] \* yverts[el.v2] + x \* (yverts[el.v2] - yverts[el.v3]) + z \* (xverts[el.v3] - xverts[el.v2])) / CalcDet(el),

((xverts[el.v3] \* yverts[el.v1] - xverts[el.v1] \* yverts[el.v3] + x \* (yverts[el.v3] - yverts[el.v1]) + z \* (xverts[el.v1] - xverts[el.v3]))) / CalcDet(el),

((xverts[el.v1] \* yverts[el.v2] - xverts[el.v2] \* yverts[el.v1] + x \* (yverts[el.v1] - yverts[el.v2]) + z \* (xverts[el.v2] - xverts[el.v1]))) / CalcDet(el));

}

public (double r, double z) LtoXTransform(double L1, double L2, double L3, Elem el)

{

return (L1 \* xverts[el.v1] + L2 \* xverts[el.v2] + L3 \* xverts[el.v3], L1 \* yverts[el.v1] + L2 \* yverts[el.v2] + L3 \* yverts[el.v3]);

}

}

}

Файл Program.cs

namespace KursEgor

{

internal class Program

{

static void Main(string[] args)

{

List<BoundaryCondition> BCs = new();

BCs.Add(new BoundaryCondition1(1, (x, y) => 1));

BCs.Add(new BoundaryCondition3(3, 1, (x, y) => 21));

BCs.Add(new BoundaryCondition2(0, (x, y) => 0));

BCs.Add(new BoundaryCondition2(4, (x, y) => 0));

FEM fem = new FEM((x, y) => -4 + 4 \* x \* x, BCs, (x, y) => 2, (x, y) => 4);

fem.Solve(3e-15, 10000);

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

/\*for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

Console.WriteLine($"{1 + 1.0 / 8 + i / 2.0} {1.0 / 8 + j / 2.0} {fem.Getsollution(1 + 1.0 / 8 + i / 2.0, 1.0 / 8 + j / 2.0)}");

Console.WriteLine($"{1 + 3.0 / 8 + i / 2.0} {3.0 / 8 + j / 2.0} {fem.Getsollution(1 + 3.0 / 8 + i / 2.0, 3.0 / 8 + j / 2.0)}");

}

}\*/

Console.WriteLine("Hello World!");

}

}

}