СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Под векторными задачами мы будем понимать задачи, в которых решением является некоторая вектор-функция. Будем рассматривать такие векторные задачи, решениями которых являются вектор-функции с компонентами, каждая из которых будет удовлетворять дифференциальному уравнению второго порядка и как минимум непрерывна. Таким образом, каждая из компонент искомой вектор-функции может быть найдена в виде линейной комбинации непрерывных базисных функции, которые использовались при решении скалярных задач. Такие базисные функции обычно называют узловыми (к ним относятся не только лагранжевы и эрмитовы базисные функции, но и иерархические). Соответственно и МКЭ, использующий при нахождении численного решения, такие базисные функции называют узловым.

Технологию построения конечноэлементных аппроксимации векторных задач на основе узлового МКЭ мы рассмотри на примере задачи (), описывающие нестационарное электромагнитное поле в однородной по магнитной проницаемости среде (и без учета токов смещения).

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. УСЛОВИЕ ЗАДАЧИ

Пусть имеется некоторый круглый индукционный источник, с радиусом $R_0 << 1000$. На рисунке 1.1 имеем однородные краевые условие на правой и нижней границах, и естественные на левой и верхней границах.

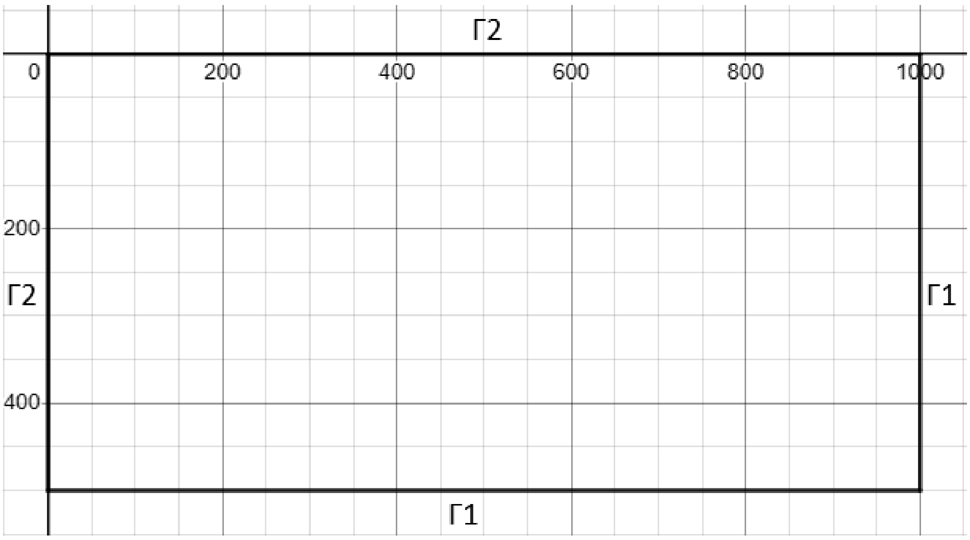

Рисунок 1.1 – Образец сетки

1.2. МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА

Будем считать, что электромагнитное поле возбуждается круговым током, а вмещающая среда имеет круговую симметрию. Тогда при условии однородности среды по магнитной проницаемости электромагнитное поле полностью описывается одной компонентой $A_\varphi = A_\varphi(r, z, t)$ вектор-потенциала $\overline{A}$ (в цилиндрической системе координат), и эта функция $A_\varphi(r, z, t)$ может быть найдена из решения двумерного уравнения:

$$-\frac{1}{\mu_0}\Delta A_\varphi + \frac{A_\varphi}{\mu_0 r^2} + \sigma\frac{\partial A_\varphi}{\partial t} = J_\varphi, \tag{1.1}$$

где: J_φ - дельта-функция равная 1 в одной из подобластей, описывающей кольцо, и равная 0 в остальных.

Переведем это дифференциальное уравнение в частных производных в слабую форму.

$$\int\limits_{\Omega} \left(-\frac{1}{\mu_0} \boldsymbol{\nabla} \left(\boldsymbol{\nabla} A_\varphi \right) + \frac{A_\varphi}{\mu_0 r^2} + \sigma \frac{\partial A_\varphi}{\partial t} \right) v d\Omega = \int\limits_{\Omega} J_\varphi v d\Omega. \tag{1.2}$$

$$\int\limits_{\Omega} \boldsymbol{\nabla} \left(-\frac{1}{\mu_0} \boldsymbol{\nabla} A_\varphi \right) v d\Omega + \int\limits_{\Omega} \frac{A_\varphi}{\mu_0 r^2} v d\Omega + \int\limits_{\Omega} \sigma \frac{\partial A_\varphi}{\partial t} v d\Omega = \int\limits_{\Omega} J_\varphi v d\Omega. \tag{1.3}$$

Применив формулу Гаусса-Остроградского, и принимая во внимание, что по условию задачи в некоторых местах поток через границу равен нулю, получим:

$$\int\limits_{\Omega} \frac{1}{\mu_0} \boldsymbol{\nabla} A_\varphi \boldsymbol{\nabla} v d\Omega + \int\limits_{\Omega} \frac{A_\varphi}{\mu_0 r^2} v d\Omega + \int\limits_{\Omega} \sigma \frac{\partial A_\varphi}{\partial t} v d\Omega - \int\limits_{\Omega} J_\varphi v d\Omega = 0. \tag{1.4}$$

1.3. ПРИНЦИПЫ ПОСТРОЕНИЯ ЛОКАЛЬНЫХ ВЕКТОРОВ, МАТРИЦ ЖЕСТКОСТИ И МАСС

Поскольку решаемое уравнение в (r, z) координатах и имеется особый нелинейный коэффициент $\gamma = \frac{1}{r^2}$, локальные матрицы жесткости и масс для одномерной задачи выглядят следующим образом:

$$\hat{G}^r = \hat{\lambda} \frac{r_k + h_k/2}{h_k} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

$$\hat{M}^r = \ln\left(1+\frac{1}{d}\right)\begin{pmatrix} (1+d)^2 & -d(1+d) \\ -d(1+d) & d^2 \end{pmatrix} - d\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} -3 & 1 \\ 1 & 1 \end{pmatrix}$$

где $d = \frac{r_k}{h_k}$.

$$\hat{G}^z = \frac{\hat{\lambda}}{h_k}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

$$\hat{M}^z = \frac{\hat{\gamma} h_k}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

Тогда элементы верхнего треугольника матрицы жесткости для двумерных задач, можем представить в виде:

$$\begin{aligned}
\hat{G}_{11} &= \hat{\lambda}\left(G^r_{11}M^z_{11} + M^r_{11}G^z_{11}\right), & \hat{G}_{12} &= \hat{\lambda}\left(G^r_{12}M^z_{11} + M^r_{12}G^z_{11}\right), \\
\hat{G}_{13} &= \hat{\lambda}\left(G^r_{11}M^z_{12} + M^r_{11}G^z_{12}\right), & \hat{G}_{14} &= \hat{\lambda}\left(G^r_{12}M^z_{12} + M^r_{12}G^z_{12}\right), \\
\hat{G}_{22} &= \hat{\lambda}\left(G^r_{22}M^z_{11} + M^r_{22}G^z_{11}\right), & \hat{G}_{23} &= \hat{\lambda}\left(G^r_{21}M^z_{12} + M^r_{21}G^z_{12}\right), \\
\hat{G}_{24} &= \hat{\lambda}\left(G^r_{22}M^z_{12} + M^r_{22}G^z_{12}\right), & \hat{G}_{33} &= \hat{\lambda}\left(G^r_{11}M^z_{22} + M^r_{11}G^z_{22}\right), \\
\hat{G}_{34} &= \hat{\lambda}\left(G^r_{12}M^z_{22} + M^r_{12}G^z_{22}\right), & \hat{G}_{44} &= \hat{\lambda}\left(G^r_{22}M^z_{22} + M^r_{22}G^z_{22}\right).
\end{aligned}$$

Верхний треугольник элементов матрицы масс может быть представлены в виде:

$$\begin{aligned}
\hat{M}_{11} &= \hat{\gamma} M^r_{11}M^z_{11}, & \hat{M}_{12} &= \hat{\gamma} M^r_{12}M^z_{11}, \\
\hat{M}_{13} &= \hat{\gamma} M^r_{11}M^z_{12}, & \hat{M}_{14} &= \hat{\gamma} M^r_{12}M^z_{12}, \\
\hat{M}_{22} &= \hat{\gamma} M^r_{22}M^z_{11}, & \hat{M}_{23} &= \hat{\gamma} M^r_{21}M^z_{12}, \\
\hat{M}_{24} &= \hat{\gamma} M^r_{22}M^z_{12}, & \hat{M}_{33} &= \hat{\gamma} M^r_{11}M^z_{22}, \\
\hat{M}_{34} &= \hat{\gamma} M^r_{12}M^z_{22}, & \hat{M}_{44} &= \hat{\gamma} M^r_{22}M^z_{22}.
\end{aligned}$$

Выразим матрицу $\hat{M}$ следующим образом:

$$\hat{M} = \hat{\gamma}\hat{C}.$$

Для генерации вектора правой части, воспользуемся следующим соотношением:

$$\hat{b} = \hat{C}\hat{f}.$$

1.4. АППРОКСИМАЦИЯ КРАЕВОЙ ЗАДАЧИ ПО ВРЕМЕНИ

Представим искомое решение u на интервале (t_{j-2}, t_j) в следующем виде:

$$u(r,z,t) \approx u^{j-2}(r,z)\eta_2^j(t) + u^{j-1}(r,z)\eta_1^j(t) + u^j(r,z)\eta_0^j(t). \tag{1.5}$$

где функции $\eta_2^j(t)$, $\eta_1^j(t)$, $\eta_0^j(t)$ - базисные квадратичные полиномы Лагранжа (с двумя корнями из набора значений времен t_{j-2}, t_{j-1}, t_j), которые могут быть записаны в виде:

$$\eta_2^j(t) = \frac{1}{\Delta t_1 \Delta t}\left(t - t_{j-1}\right)\left(t - t_j\right),$$

$$\eta_1^j(t) = -\frac{1}{\Delta t_1 \Delta t_0}\left(t - t_{j-2}\right)\left(t - t_j\right),$$

$$\eta_0^j(t) = \frac{1}{\Delta t \Delta t_0}\left(t - t_{j-2}\right)\left(t - t_{j-1}\right),$$

где:

$$\Delta t = t_j - t_{j-2}, \;\; \Delta t_1 = t_{j-1} - t_{j-2}, \;\; \Delta t_0 = t_j - t_{j-1}.$$

Применим представление 1.5 для аппроксимации производной по времени параболического уравнения 1.1 на временном слое $t = t_j$:

$$\sigma \frac{\partial}{\partial t} \left(u^{j-2}(r,z)\eta_2^j(t) + u^{j-1}(r,z)\eta_1^j(t) + u^j(r,z)\eta_0^j(t) \right) - \frac{1}{\mu_0}\Delta A_\varphi + \frac{A_\varphi}{\mu_0 r^2} = J_\varphi \tag{1.6}$$

Выполняя конечноэлементную аппроксимацию краевой задачи для уравнения 1.6, получим СЛАУ следующего вида:

$$\left(\frac{\Delta t + \Delta t_0}{\Delta t \Delta t_0} M + G + M \right) q^j \;\; = \;\; b^j \; - \; \frac{\Delta t_0}{\Delta t \Delta t_1} M q^{j-2} \; + \; \frac{\Delta t}{\Delta t_1 \Delta t_0} M q^{j-1}. \tag{1.7}$$

2. ИССЛЕДОВАНИЯ

2.1. ПРЕДВАРИТЕЛЬНОЕ ОПИСАНИЕ

Проведем тестирование программы на работоспособность сначала для уравнения $-\frac{1}{\mu_0}\Delta A_\varphi + \frac{1}{\mu_0}A_\varphi = f$, затем на уравнения $-\frac{1}{\mu_0}\Delta A_\varphi + \frac{1}{\mu_0 r^2}A_\varphi = f$, где f - любая функция, достаточно очевидная для проверки результата. После этого рассмотрим интересующее нас уравнение $-\frac{1}{\mu_0}\Delta A_\varphi + \frac{1}{\mu_0 r^2}A_\varphi = J_\varphi$. Кольцо, создающее поле, расположено в точке $(10; 0)$ и отмечено на рисунке синей точкой. Базисные функции билинейные. Образец расчетной области изображен на рисунке 2.1:

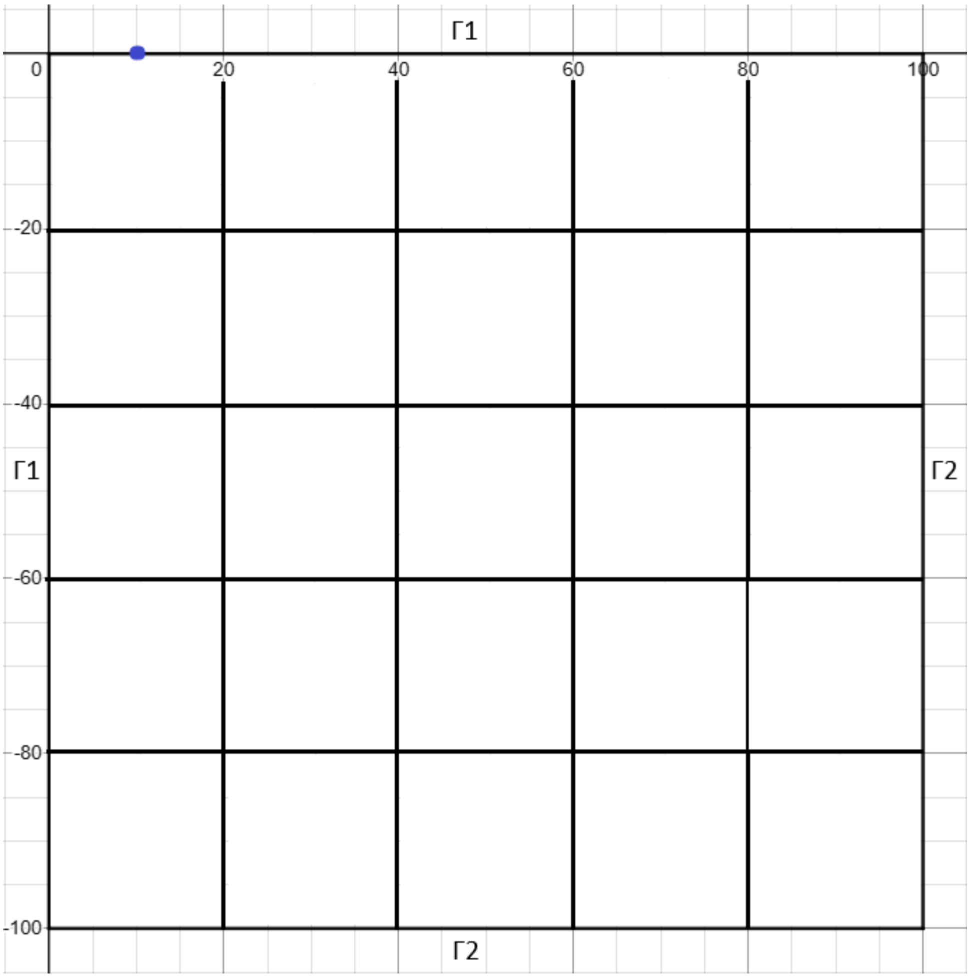

Рисунок 2.1 – Расчетная область

2.2. ТЕСТИРОВАНИЕ НА РАБОТОСПОСОБНОСТЬ

Пусть искомая функция будет $u = r * z$, а $\mu_0 = 1$, однородные условия первого рода расположены на верхней и левой границах расчетной области, на правой и нижней границе условия однородные второго рода. Получим следующее.

Таблица 2.1 – Тестирование при $-\frac{1}{\mu_0}\Delta A_\varphi + \frac{1}{\mu_0}A_\varphi = f$, $u = rz$, $f = rz$, $\mu_0 = 1$

Узел	Значение	Абсолютная погрешность	Относительная погрешность
(20,0008; -20)	-3.99991668E+002	2.43321134E-002	6.08278503E-005
(40,0006; -20)	-8.00041536E+002	2.95364092E-002	3.69199577E-005
(60,0004; -20)	-1.19973146E+003	2.76538286E-001	2.30447035E-004
(80,0002; -20)	-1.60083585E+003	8.31847171E-001	5.19903182E-004
(100; -20)	-1.99651310E+003	3.48690361E+000	1.74345181E-003
(20,0008; -40)	-8.00073346E+002	4.13461873E-002	5.16806669E-005
(40,0006; -40)	-1.60026312E+003	2.39123420E-001	1.49449896E-004
(60,0004; -40)	-2.39973286E+003	2.83136319E-001	1.17972680E-004
(80,0002; -40)	-3.20203212E+003	2.02411646E+000	6.32534811E-004
(100; -40)	-3.99347480E+003	6.52519957E+000	1.63129989E-003
(20,0008; -60)	-1.19978676E+003	2.61241225E-001	2.17692313E-004
(40,0006; -60)	-2.39974806E+003	2.87944345E-001	1.19975011E-004
(60,0004; -60)	-3.59862984E+003	1.39415571E+000	3.87262893E-004
(80,0002; -60)	-4.80175375E+003	1.74174814E+000	3.62863288E-004
(100; -60)	-5.98860115E+003	1.13988464E+001	1.89980774E-003
(20,0008; -80)	-1.60091687E+003	8.52871444E-001	5.33023331E-004
(40,0006; -80)	-3.20206687E+003	2.01886922E+000	6.30887168E-004
(60,0004; -80)	-4.80177546E+003	1.74346432E+000	3.63219313E-004
(80,0002; -80)	-6.40714832E+003	7.13231677E+000	1.11442171E-003
(100; -80)	-7.99078772E+003	9.21228366E+000	1.15153546E-003
(20,0008; -100)	-1.99661901E+003	3.46099134E+000	1.73042645E-003
(40,0006; -100)	-3.99352786E+003	6.53213746E+000	1.63300987E-003
(60,0004; -100)	-5.98864281E+003	1.13971859E+001	1.89951832E-003
(80,0002; -100)	-9.96592479E+003	9.21283395E+000	1.15160136E-003
(100; -100)	-9.96592479E+003	3.40752140E+001	3.40752140E-003

Исходя из приведенных результатов в таблице 2.1, делаем вывод, что программа работает верно.

Теперь протестируем программу для уравнения $-\frac{1}{\mu_0}\Delta A_\varphi + \frac{1}{\mu_0 r^2}A_\varphi = f$. Пусть искомая функция все также будет $u = rz$, тогда $f = \frac{z}{r}$, а $\mu_0 = 1$, однородные условия первого рода расположены на верхней и левой границах расчетной области, на правой и нижней границе условия однородные второго рода. Получим следующее.

Таблица 2.2 – Тестирование при $-\frac{1}{\mu_0}\Delta A_\varphi + \frac{1}{\mu_0 r^2}A_\varphi = f$, $u = rz$, $f = \frac{z}{r}$, $\mu_0 = 1$

Узел	Значение	Абсолютная погрешность	Относительная погрешность
(20,0008; -20)	-5.30195216E+003	4.90193616E+003	1.22543502E+001
(40,0006; -20)	1.38386203E+003	2.18387403E+003	2.72980159E+000
(60,0004; -20)	-3.63322442E+002	8.36685558E+002	6.97233316E-001
(80,0002; -20)	1.00509320E+002	1.70051332E+003	1.06281817E+000
(100; -20)	-4.94549306E+001	1.95054507E+003	9.75272535E-001
(20,0008; -40)	-1.06050859E+004	9.80505388E+003	-1.22558271E+001
(40,0006; -40)	2.76802513E+003	4.36804913E+003	2.72998975E+000
(60,0004; -40)	-7.26721979E+002	1.67329402E+003	6.97201194E-001
(80,0002; -40)	2.01039370E+002	3.40104737E+003	1.06282465E+000
(100; -40)	-9.89198385E+001	3.90108016E+003	-9.75270040E-001
(20,0008; -60)	-1.59033866E+004	1.47033386E+004	1.22522921E+001
(40,0006; -60)	4.15095747E+003	6.55099347E+003	2.72953967E+000
(60,0004; -60)	-1.08980656E+003	2.51021744E+003	6.97277974E-001
(80,0002; -60)	3.01484793E+002	5.10149679E+003	1.06280917E+000
(100; -60)	-1.48344038E+002	5.85165596E+003	9.75275994E-001
(20,0008; -80)	-2.12202745E+004	1.96202105E+004	1.22621411E+001
(40,0006; -80)	5.53861996E+003	8.73866796E+003	2.73079278E+000
(60,0004; -80)	-1.45410075E+003	3.34593125E+003	6.97064363E-001
(80,0002; -80)	4.02254968E+002	6.80227097E+003	1.06285218E+000
(100; -80)	-1.97924358E+002	7.80207564E+003	9.75259455E-001
(20,0008; -100)	-2.64659667E+004	2.44658867E+004	1.22324541E+001
(40,0006; -100)	6.90817680E+003	1.09082368E+004	2.72701830E+000
(60,0004; -100)	-1.81376823E+003	4.18627177E+003	6.97707310E-001
(80,0002; -100)	5.01783996E+002	8.50180400E+003	1.06272284E+000
(100; -100)	-2.46908390E+002	9.75309161E+003	9.75309161E-001

Теперь протестируем программу для уравнения 1.1 для нулевого слоя по времени. J_φ - δ-функция, равная 1 при $(r, z) = (10, 0)$ и 0 во всех остальных точках. Сетка изображена на рисунке 2.1. Однородные краевые условия первого рода находятся на правой и нижней границе, однородные второго рода на верхней и левой границах. Таким образом получим следующее.

Таблица 2.3 – Тестирование исходной функции

Узел	Значение
(0,001; 0)	9.89596002E+000
(20,0008; 0)	4.59762041E+000
(40,0006; 0)	2.05043455E+000
(60,0004; 0)	1.04570134E+000
(80,0002; 0)	4.41814204E-001
(0,001; -20)	6.58639311E-002
(20,0008; -20)	3.00036881E-002
(40,0006; -20)	1.67783663E-002
(60,0004; -20)	9.20512908E-003
(80,0002; -20)	3.99440695E-003
(0,001; -40)	-1.72040980E-002
(20,0008; -40)	-7.83509188E-003
(40,0006; -40)	-4.37808675E-003
(60,0004; -40)	-2.40083452E-003
(80,0002; -40)	-1.04159964E-003
(0,001; -60)	4.47563199E-003
(20,0008; -60)	2.03775872E-003
(40,0006; -60)	1.13777548E-003
(60,0004; -60)	6.23635462E-004
(80,0002; -60)	2.70511050E-004
(0,001; -80)	-1.09467887E-003
(20,0008; -80)	-4.98291674E-004
(40,0006; -80)	-2.78024921E-004
(60,0004; -80)	-1.52325956E-004
(80,0002; -80)	-6.60620166E-005

Исходя из результатов в таблице 2.3, интуитивно можно предположить, что результат программы необходимо аппроксимирует решение задачи.

СПИСОК ЛИТЕРАТУРЫ

1. Ю.Г. Соловейчик, М.Э. Рояк, М.Г. Персова Метод конечных элементов для скалярных и векторных задач Учеб. пособие. — Новосибирск: Изд-во НГТУ, 2007 — 896 с.

2. А.Н. Тихонов, А.А. Самарский Уравнения математической физики: Учеб.пособие. / А.Н. Тихонов, А.А. Самарский — 6-е изд., — М: Изд-во МГУ, 1999 — 799 с.

ПРИЛОЖЕНИЕ. ТЕКСТ ПРОГРАММЫ.

Program.cs

```
using Project;
using System.Globalization;
using MathObjects;
using DataStructs;
using Solver;
CultureInfo.CurrentCulture = CultureInfo.InvariantCulture;

const string CalculationArea = @"D:\CodeRepos\CS\Diplom\Data\Input\Info.dat";
const string BordersInfo = @"D:\CodeRepos\CS\Diplom\Data\Input\Borders.dat";
const string AnswerPath = @"D:\CodeRepos\CS\Diplom\Data\Output\Answer.dat";


FEM myFEM = new();
myFEM.ReadData(CalculationArea, BordersInfo);
myFEM.ConstructMesh();
myFEM.BuildMatrixAndVector();
myFEM.SetSolver(new MCG());
myFEM.Solve();
myFEM.WriteData(AnswerPath);
```

LocalMatrix.cs

```
using DataStructs;

namespace MathObjects;

public class LocalMatrix : Matrix
{
    private readonly double _lambda;

    private readonly double _rk;

    private readonly double _hr;

```

```
    private readonly double _gamma;

    private readonly double _hz;

    private readonly double _d;


    public double this[int i, int j]
    {
        get
        {
            if (i > 3 || j > 3) throw new IndexOutOfRangeException("Local matrix error.");
                return _lambda * ((_rk / _hr + 0.5) * _G[i % 2, j % 2] * _hz * _Mz[i / 2, j / 2] +
                        (Math.Log(1 + 1 / _d) * _Mr1[i % 2, j % 2] - _d * _G[i % 2, j % 2] + _Mr2[i % 2, j % 2]) * _G[i / 2, j / 2] / _hz) +
                        _gamma * (_hz * _Mz[i / 2, j / 2] * (Math.Log(1 + 1 / _d) * _Mr1[i % 2, j % 2] - _d * _G[i % 2, j % 2] + _Mr2[i % 2, j % 2]));
        }
    }


/*
    public double this[int i, int j]
    {
        get
        {
            if (i > 3 || j > 3) throw new IndexOutOfRangeException("Local matrix error.");
                return _lambda * (_G[i % 2, j % 2] / _hr * _hz * _Mz[i / 2, j / 2] +
                                   _hr * _M[i % 2, j % 2] * _G[i / 2, j / 2] / _hz) +
                        _gamma * (_hz * _Mz[i / 2, j / 2] * _hr * _M[i % 2, j % 2]);
        }
    }*/

    private readonly double[,] _G =  {{ 1.0, -1.0},
                                       {-1.0,  1.0}};

    private readonly double[,] _M = {{0.3333333333333333, 0.16666666666666666},
                                      {0.16666666666666666, 0.3333333333333333}};

```

```
    private readonly double[,] _Mz = {{0.3333333333333333, 0.1666666666666666},
                                      {0.1666666666666666, 0.3333333333333333}};

    private readonly double[,] _Mr1;

    private readonly double[,] _Mr2 = {{-1.5, 0.5},
                                       {0.5, 0.5}};



    public LocalMatrix(double lambda, double gamma, double rk, double hz, double hr)
    {
        _lambda = lambda;
        _gamma = gamma;
        _rk = rk;
        _hr = hr;
        _hz = hz;
    }

    public LocalMatrix(List<int> elem, ArrayOfPoints arrPt, double lambda = 1, double gamma = 1)
    {
        _rk = arrPt[elem[0]].R;
        _hr = arrPt[elem[1]].R - arrPt[elem[0]].R;
        _hz = arrPt[elem[2]].Z - arrPt[elem[0]].Z;
        _d = _rk / _hr;
        _lambda = lambda;
        _gamma = gamma;
        _Mr1 = new double[2,2] {{(1 + _d) * (1 + _d), -_d * (1 + _d)},
                                {     -_d * (1 + _d),        _d * _d}};
    }
}
```

LocalVector.cs

```
using DataStructs;

namespace MathObjects;

public class LocalVector : Vector
```

```
{
    private readonly bool _isRingBoundaryInside;


    private double F(double r, double z)
    {
        if (_isRingBoundaryInside && z == 0.0)
        {
            double h = Math.Abs(10.0D - r);
            return 1.0D - h / Math.Abs(_hr);
        }
        return 0.0D;
    }

    //private static double F(double r, double z) => z / r;

    private readonly double _r0;

    private readonly double _r1;

    private readonly double _z0;

    private readonly double _z1;

    private readonly double _hr;

    private readonly double _hz;

    private readonly double _d;

    private readonly double[,] _M2R = {{0.08333333333333333, 0.08333333333333333},
                                       {0.08333333333333333, 0.25}};

    private readonly double[,] _Mz = {{0.3333333333333333, 0.1666666666666666},
                                     {0.1666666666666666, 0.3333333333333333}};

    private readonly double[,] _M = {{0.3333333333333333, 0.1666666666666666},
                                    {0.1666666666666666, 0.3333333333333333}};
```

```
    private readonly double[,] _Mr1;

    private readonly double[,] _G = {{1, -1},
                                     {-1, 1}};

    private readonly double[,] _Mr2 = {{-1.5, 0.5},
                                       { 0.5, 0.5}};



    public override double this[int i] => i switch
    {
        0 => _hz * _hr * (_Mz[0, 0] * (Math.Log(1 + 1 / _d) * _Mr1[0, 0] - _d * _G[0, 0] + _Mr2[0, 0]) * F(_r0, _z0) +
                    _Mz[0, 0] * (Math.Log(1 + 1 / _d) * _Mr1[0, 1] - _d * _G[0, 1] + _Mr2[0, 1]) * F(_r1, _z0) +
                    _Mz[0, 1] * (Math.Log(1 + 1 / _d) * _Mr1[0, 0] - _d * _G[0, 0] + _Mr2[0, 0]) * F(_r0, _z1) +
                    _Mz[0, 1] * (Math.Log(1 + 1 / _d) * _Mr1[0, 1] - _d * _G[0, 1] + _Mr2[0, 1]) * F(_r1, _z1)),

        1 => _hz * _hr * (_Mz[0, 0] * (Math.Log(1 + 1 / _d) * _Mr1[1, 0] - _d * _G[1, 0] + _Mr2[1, 0]) * F(_r0, _z0) +
                    _Mz[0, 0] * (Math.Log(1 + 1 / _d) * _Mr1[1, 1] - _d * _G[1, 1] + _Mr2[1, 1]) * F(_r1, _z0) +
                    _Mz[0, 1] * (Math.Log(1 + 1 / _d) * _Mr1[1, 0] - _d * _G[1, 0] + _Mr2[1, 0]) * F(_r0, _z1) +
                    _Mz[0, 1] * (Math.Log(1 + 1 / _d) * _Mr1[1, 1] - _d * _G[1, 1] + _Mr2[1, 1]) * F(_r1, _z1)),

        2 => _hz * _hr * (_Mz[1, 0] * (Math.Log(1 + 1 / _d) * _Mr1[0, 0] - _d * _G[0, 0] + _Mr2[0, 0]) * F(_r0, _z0) +
                    _Mz[1, 0] * (Math.Log(1 + 1 / _d) * _Mr1[0, 1] - _d * _G[0, 1] + _Mr2[0, 1]) * F(_r1, _z0) +
                    _Mz[1, 1] * (Math.Log(1 + 1 / _d) * _Mr1[0, 0] - _d * _G[0, 0] + _Mr2[0, 0]) * F(_r0, _z1) +
                    _Mz[1, 1] * (Math.Log(1 + 1 / _d) * _Mr1[0, 1] - _d * _G[0, 1] + _Mr2[0, 1]) * F(_r1, _z1)),

```

```
        3 => _hz * _hr * (_Mz[1, 0] * (Math.Log(1 + 1 / _d) * _Mr1[1, 0] - _d * _G[1, 0] + _Mr2[1, 0]) * F(_r0, _z0) +
                    _Mz[1, 0] * (Math.Log(1 + 1 / _d) * _Mr1[1, 1] - _d * _G[1, 1] + _Mr2[1, 1]) * F(_r1, _z0) +
                    _Mz[1, 1] * (Math.Log(1 + 1 / _d) * _Mr1[1, 0] - _d * _G[1, 0] + _Mr2[1, 0]) * F(_r0, _z1) +
                    _Mz[1, 1] * (Math.Log(1 + 1 / _d) * _Mr1[1, 1] - _d * _G[1, 1] + _Mr2[1, 1]) * F(_r1, _z1)),
        _ => throw new IndexOutOfRangeException("Vector out of index"),
    };
/*
    public override double this[int i] => i switch
    {
        0 => _hz * _hr * (_Mz[0, 0] * _M[0, 0] * F(_r0, _z0) +
                    _Mz[0, 0] * _M[0, 1] * F(_r1, _z0) +
                    _Mz[0, 1] * _M[0, 0] * F(_r0, _z1) +
                    _Mz[0, 1] * _M[0, 1] * F(_r1, _z1)),

        1 => _hz * _hr * (_Mz[0, 0] * _M[1, 0] * F(_r0, _z0) +
                    _Mz[0, 0] * _M[1, 1] * F(_r1, _z0) +
                    _Mz[0, 1] * _M[1, 0] * F(_r0, _z1) +
                    _Mz[0, 1] * _M[1, 1] * F(_r1, _z1)),

        2 => _hz * _hr * (_Mz[1, 0] * _M[0, 0] * F(_r0, _z0) +
                    _Mz[1, 0] * _M[0, 1] * F(_r1, _z0) +
                    _Mz[1, 1] * _M[0, 0] * F(_r0, _z1) +
                    _Mz[1, 1] * _M[0, 1] * F(_r1, _z1)),

        3 => _hz * _hr * (_Mz[1, 0] * _M[1, 0] * F(_r0, _z0) +
                    _Mz[1, 0] * _M[1, 1] * F(_r1, _z0) +
                    _Mz[1, 1] * _M[1, 0] * F(_r0, _z1) +
                    _Mz[1, 1] * _M[1, 1] * F(_r1, _z1)),
        _ => throw new IndexOutOfRangeException("Vector out of index"),
    };
*/

    public LocalVector(List<int>? elem, ArrayOfPoints arrPt)
    {
        _r0 = arrPt[elem[0]].R;
        _r1 = arrPt[elem[1]].R;
```

```

        _isRingBoundaryInside = _r0 <= 10.0D && 10.0D <= _r1;

        _z0 = arrPt[elem[0]].Z;
        _z1 = arrPt[elem[2]].Z;
        _hr = _r1 - _r0;
        _hz = _z1 - _z0;
        _d = _r0 / _hr;
        _Mr1 = new double[2, 2] {{(1 + _d) * (1 + _d), -_d * (1 + _d)},
                                 {     -_d * (1 + _d),        _d * _d}};
    }

    public LocalVector(ArrayOfPoints arrPt, List<int>? arrBr)
    {
        Console.WriteLine("II bc commited :)");
    }

    public LocalVector(double r0, double r1, double z0, double z1)
    {
        _r0 = r0;
        _r1 = r1;
        _z0 = z0;
        _z1 = z1;
        _hr = _r1 - _r0;
        _hz = _z1 - _z0;
        _d = _r0 / _hr;
    }
}
```

MCG.cs

```
using MathObjects;

namespace Solver;

public class MCG : ISolver
{
    private const int _maxIter = 10000;

    private const double _eps = 1E-15;
```

```
    public GlobalVector Solve(GlobalMatrix A, GlobalVector b)
    {
        GlobalVector x = new(b.Size);
        GlobalVector x_ = new(b.Size);

        GlobalVector r = new(b.Size);
        GlobalVector r_ = new(b.Size);

        GlobalVector z = new(b.Size);
        GlobalVector z_ = new(b.Size);

        double alph = 0.0D;
        double beta = 0.0D;

        r_ = b - A * x_;
        z_ = r_;

        int iter = 0;
        do
        {
            alph = (r_ * r_) / ((A * z_) * z_);

            x = x_ + alph * z_;
            r = r_ - alph * (A * z_);

            beta = (r * r) / (r_ * r_);
            z = r + beta * z_;

            iter++;

            x_ = x;
            z_ = z;
            r_ = r;
            Console.WriteLine($"{r.Norma() / b.Norma():E15}");
        } while (iter < _maxIter && r.Norma() / b.Norma() >= _eps);

        Console.WriteLine(
        $@"Computing finished!
Total iterations: {iter}
```

```
Relative residuality: {r.Norma() / b.Norma():E15}");
        return x;
    }
}
```