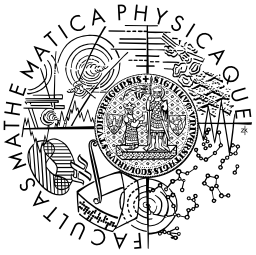


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Vojtěch Příhoda

Numerická simulace problémů elektrohydrodynamiky

Katedra numerické matematiky

Vedoucí diplomové práce:	prof. RNDr. Vít Dolejší, Ph.D., DSc.
Studijní program:	Fyzika
Studijní obor:	Matematické a počítačové modelování ve fyzice a technice

Praha 2017

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Rád bych poděkoval rodině, přátelům a své přítelkyni, kteří mě trpělivě podporovali během psaní této práce. Také bych rád poděkoval svému vedoucímu, učitelům a spolužákům na MFF UK a všem ostatním, kteří přispěli k mému studiu.

Název práce: Numerická simulace problémů elektrohydrodynamiky

Autor: Vojtěch Příhoda

Katedra: Katedra numerické matematiky

Vedoucí diplomové práce: prof. RNDr. Vít Dolejší, Ph.D., DSc., katedra

Abstrakt: Tato práce se zabývá rozšířením softwaru *adgfem* o výpočet nelineárního statického magnetického pole v komplexní geometrii. Software *adgfem* implementuje nespojitou Galerkinovu metodu a dosud byl využívám především pro rovnice typu konvekce-difuse a postrádal ucelený přístup k vytváření sítě. Tato práce přináší podrobný návod, jak tvořit komplexní geometrie s využitím softwaru *SALOME*. Ty jsou pak pomocí nově napsaného konvertoru *datToAdgfem* převedeny do formátu čitelného pro *adgfem*. Na takto vytvořené síti je pak proveden výpočet nelineárního statického magnetického pole.

Klíčová slova: adgfem DGM SALOME datToAdgfem

Title: Numerical simulation of problems of elektrohydrodynamics

Author: Vojtěch Příhoda

Department: Department of Numerical Mathematics

Supervisor: prof. RNDr. Vít Dolejší, Ph.D., DSc., department

Abstract: In this thesis, software *adgfem* is extended to be capable of calculation of non-linear magnetic field in complex geometry. Software *adgfem* implements discontinuous Galerkin method and so far has been used mainly to solve convection-diffusion problems and lacked streamlined approach to computational mesh generation. This thesis contains step-by-step guideline to creation of complex geometry using software *SALOME*. This mesh is then converted to format suitable for *adgfem* using newly written convertor *datToAdgfem*. Mesh created in this way is then used for calculation of non-linear static magnetic field.

Keywords: adgfem DGM SALOME datToAdgfem

Obsah

Úvod

Původní myšlenkou práce bylo rozšířit software *adgfem* tak, aby v něm bylo možné počítat úlohy z magnetohydrodynamiky. Program *adgfem* implementuje nespojitou Galerkinovu metodu, je vyvíjen na Katedře numerické matematiky MFF UK pod vedením prof. Víta Dolejšího. Je napsán v jazyce FORTRAN. Více o nespojité Galerkinově metodě např. v (Vít Dolejší, 2015).

Už dnes je *adgfem* schopný počítat jak rovnice proudění, tak i Poissonovu rovnici, na kterou lze stacionární magnetické pole převést. Nicméně, místo snahy rovnou implementovat rovnice magnetohydronamiky jsem si vybral jako první krok úlohu stacionárního nelineárního magnetického pole. Tuto úlohu jsem chtěl řešit na nějaké realistické geometrii. Ukázalo se však, že *adgfem* není vybaven žádným robustním síťovacím nástrojem, ve kterém by bylo možné požadovanou geometrii osíťovat. Navíc, z takového nástroje by mohli profitovat i další uživatelé *adgfemu*. Tím se těžiště práce přesunulo z magneticého pole na síťování.

Ukázalo se, že nejlepší cestou je použít již exitstující nástroje, konkrétně *WebPlotDigitizer* a *SALOME*. Kromě toho bylo třeba tyto nástroje nějak spojit. Výstup z *WebPlotDigitizeru* se podařilo dostat do *SALOME* pomocí jednoduchého skriptu v Pythonu. Pro převedení výstupu ze *SALOME* do *adgfem* bylo třeba napsat konvertor *datToAdgfem*, protože nešlo o pouhé přeskládání čísel v souboru, nýbrž bylo zajistit některé vlastnosti, které *adgfem* od sítě očekává, jako např. správně orientované trojúhelníky a hrany na okrajích sítě. Kromě toho se *datToAdgfem*umí vypořádat i s vnitřními hranicemi, tj. hranami, které oddělují např. jednotlivé materiály a musí být zachovány i po tom, co *adgfem* síť adaptuje.

Tím ovšem původní záměr práce ustoupil do pozadí. Z úlohy na výpočet magnetického pole se stal spíš prostředek, jak ověřit, že *datToAdgfem*je skutečně funkční. Na implementaci magnetohydrodnamiky se z časových důvodů už bohužel nedostalo.

1. Magnetické pole ve 2D

1.1 Zadání

Pro tuto práci jsem převzal úlohou č. 4 v (Křížek M., 2001). Buď $\Omega \subset \mathbb{R}^2$ oblast, zabýváme se tedy řezem nějakým magnetickým polem, typicky nějakého elektromagnetického stroje. Systém Maxwellových rovnic popisující stacionární magnetické pole je

$$\operatorname{rot} H = f \qquad \text{v } \Omega \tag{1.1}$$

$$\operatorname{div} B = 0 \qquad \text{v } \Omega, \tag{1.2}$$

kde H je magnetická intenzita, B je magnetická indukce a f je proudová hustota (resp. její složka kolmá na rovinu, kterou uvažujeme). Vzhledem k tomu, že pracujeme v $\mathbb{R}^2$, budeme chápat, že

$$\operatorname{rot} H = \frac{\partial H_2}{\partial x_1} - \frac{\partial H_1}{\partial x_2}.$$

Vztah mezi H a B budeme uvažovat ve tvaru

$$H(x) = \nu(x, |B(x)|^2)B(x) \qquad \forall x \in \Omega, \tag{1.3}$$

kde ν je magnetická reluktivita splňující

$$\nu(x,\eta) = \begin{cases} \dfrac{1}{\mu_0}, & \forall x \in \Omega_1, \\ \nu_2(\eta), & \forall x \in \Omega_2. \end{cases}$$

Konstanta μ_0 je permeabilita vakua. Množiny Ω_1 a Ω_2 jsou disjunktní části Ω, $\Omega = \overline{\Omega_1} \cup \overline{\Omega_2}$, představující různé materiály. V našem případě půjde o vzduch pro Ω_1 a ocel statoru a rotoru pro Ω_2. Reluktivita této oceli je dána vztahem

$$\nu_2(\eta) = \frac{1}{\mu_0}\left(\alpha + (1-\alpha)\frac{\eta^4}{\eta^4 + \beta}\right)$$

pro konstanty $\alpha = 0.0003$ a $\beta = 16000$, viz (R. Glowinski, 1974). Okrajovou podmínku pro náš problém budeme předpokládat ve tvaru

$$\vec{n} \cdot B = 0 \qquad \text{na } \partial\Omega, \tag{1.4}$$

kde $\vec{n}$ je jednotková vnější normála k Ω.

1.2 Magnetický potenciál

Díky (1.2) můžeme předpokládat existenci magnetického potenciálu u (daného až na konstantu) splňujícího

$$B = \operatorname{curl} u,$$

ke chápeme $\operatorname{curl} u = (\partial_2 u, -\partial_1 u)$. Detailnější zdůvodnění viz (Vivette Girault, 1986). Vidíme, že

$$|B(x)| = |\operatorname{curl} u(x)| = |\operatorname{grad} u(x)|,$$

což nám spolu s (1.3) dává

$$\nu(x, |B(x)|^2) = \nu(x, |\operatorname{grad} u(x)|^2) \qquad \forall x \in \Omega.$$

To spolu s (1.1) vede k

$$f = \operatorname{rot} H = \operatorname{rot}\left(\nu(\cdot, |B|^2)B\right) = \operatorname{rot}\left(\nu(\cdot, |\operatorname{curl} u|^2)\operatorname{curl} u\right)$$
$$= -\operatorname{div}\left(\nu(\cdot, |\operatorname{curl} u|^2)\operatorname{grad} u\right).$$

Pro vektor $\vec{t} = (-n_2, n_1)$ kolmý na jednotkovou vnější normálu $\vec{n}$ dostaneme

$$0 = \vec{n} \cdot B = \vec{n} \cdot \operatorname{curl} u = \vec{t} \cdot \operatorname{grad} u = \frac{\partial u}{\partial t},$$

neboli u je podél hranice $\partial\Omega$ konstantní. Bez újmy na obecnosti můžeme zvolit $u = 0$ na $\partial\Omega$. V řeči magnetického potenciálu tedy můžeme původní úlohu přeformulovat do

$$-\operatorname{div}\left(\nu(\cdot, |\operatorname{curl} u|^2)\operatorname{grad} u\right) = f \qquad \text{v } \Omega,$$
$$u = 0 \qquad \text{na } \partial\Omega.$$

Tímto jsme získali nelineární Dirichletův problém. Díky příznivým vlastnostem funkce $\nu(x, \eta)$ je možné dokázat, že operátor působící na pravé straně na u je lipschitzovsky spojitý a silně monotóní. Tyto vlastnosti nám s pomocí teorie monotóních operátorů zajistí řešitelnost této úlohy. Detailní provedení důkazu viz (Křížek M., 2001).

2. Tvorba sítě

Sítě pro *adgfem* byly doposud vytvářeny buď zcela ručně (pro ty úplně nejjednodušší případy), nebo pomocí ad hoc programů nebo skriptů na jedno použití. Tento přístup doposud stačil, protože *adgfem* má schopnost sítě adaptivně upravovat, takže stačilo vygenerovat relativně hrubou síť s malým počtem elementů. Pro tuto práci jsem si ovšem vybral geometrii, která byla podstatně komplexnější, než ty, které se s pomocí *adgfem* počítaly doposud. Bylo by sice možné napsat skript, který síť pro tuto geometrii vygeneroval, bylo by to ovšem velmi pracné. Rozhodl jsem se proto místo toho vytvořit obecnější postup, který bude znovu použitelný i pro úplně jiné geometrie.

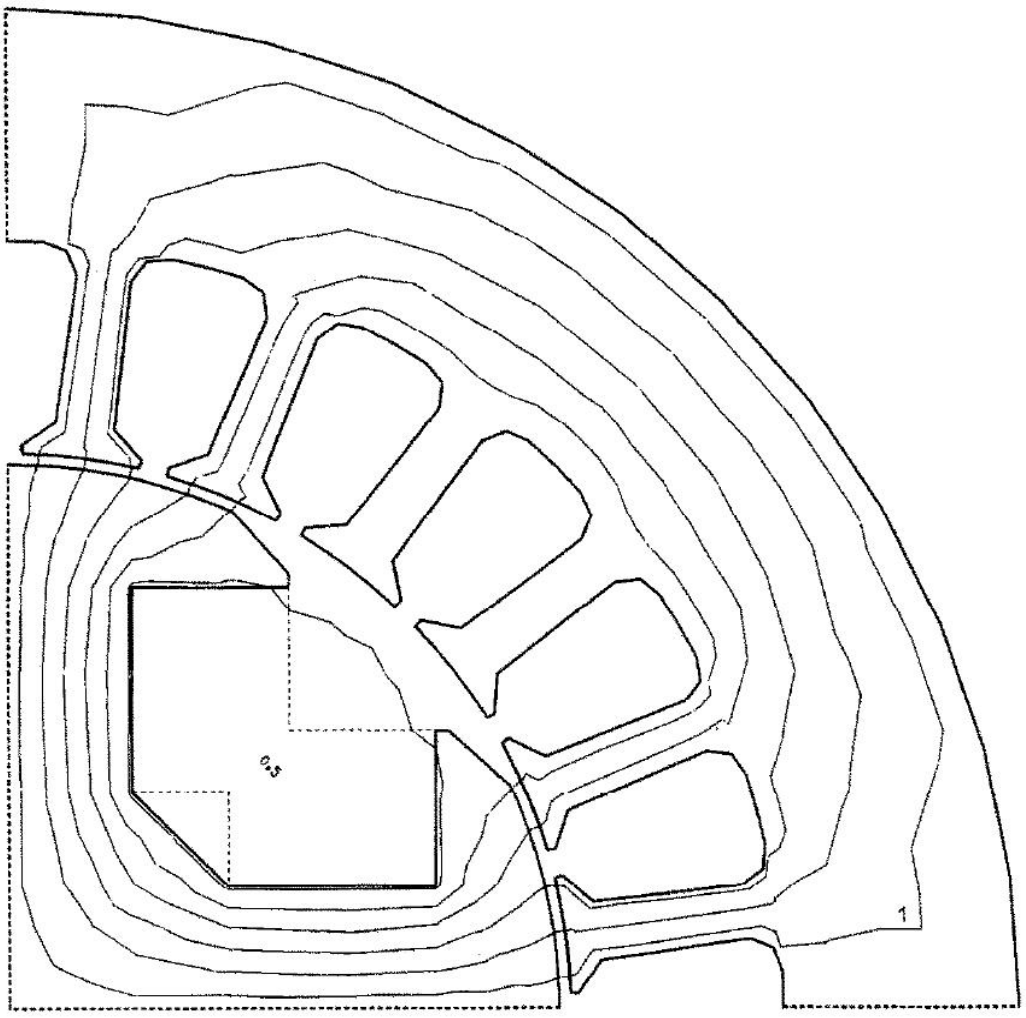
2.1 Definice geometrie

Geometrie pro tuto práci je převzata z R. Glowinski (1974). Geometrii použitou pro výpočet můžete vidět na obrázku 2.1. Ta představuje čtvrtinu z řezu alternátorem (viz. 2.2). Podobně jako autor původního článku předpokládáme pro jednoduchost symetrii okolo vodorovné a svislé osy, takže si můžeme dovolit počítat jen jednu čtvrtinu. Původní článek neobsahuje kromě obrázku žádnou přesnější definici geometrie, bylo proto nutné se spokojit s tím. Abych z obrázku pokud extrahoval pokud možno co nejpřesnější geometrii, použil jsem nástroj *WebPlotDigitizer* (viz Rohatgi). *WebPlotDigitizer* je svobodná opensource aplikace (licence GPLv3), kterou lze spustit buď přímo ve webovém prohlížeči, nebo stáhnout a spustit na vlastním počítači. Zdrojový kód je k dispozici na GitHubu. *WebPlotDigitizer* je přímo určen pro odečítání hodnot z obrázků. V tomto případě se jednalo o obrázek 2.1. Stačí pak zadat dva referenční body pro každou osu, čímž je určen souřadný systém, a pak lze odečítat souřednice kteréhokoli bodu na obrázku. Tím jsem získal soubor se souřadnicemi význačných bodů, který lze nalézt v příloze 3.3. Soubor jsem si pojmenovat `points.xy`. Původní článek neobsahuje informaci o rozměru alternátoru, předpokládal jsem tedy, že jeho průměr je 1 m. Protože je na první pohled vidět, že stator i rotor má několikačetnou symetrii, ušetřil jsem si práci tím, že jsem vynesl jen ty body, které nebylo možné získat pomocí nějakého zrcadlení nebo otočení. Tím se počet bodů podařilo udržet daleko snesitelnější pro ruční zacházení.

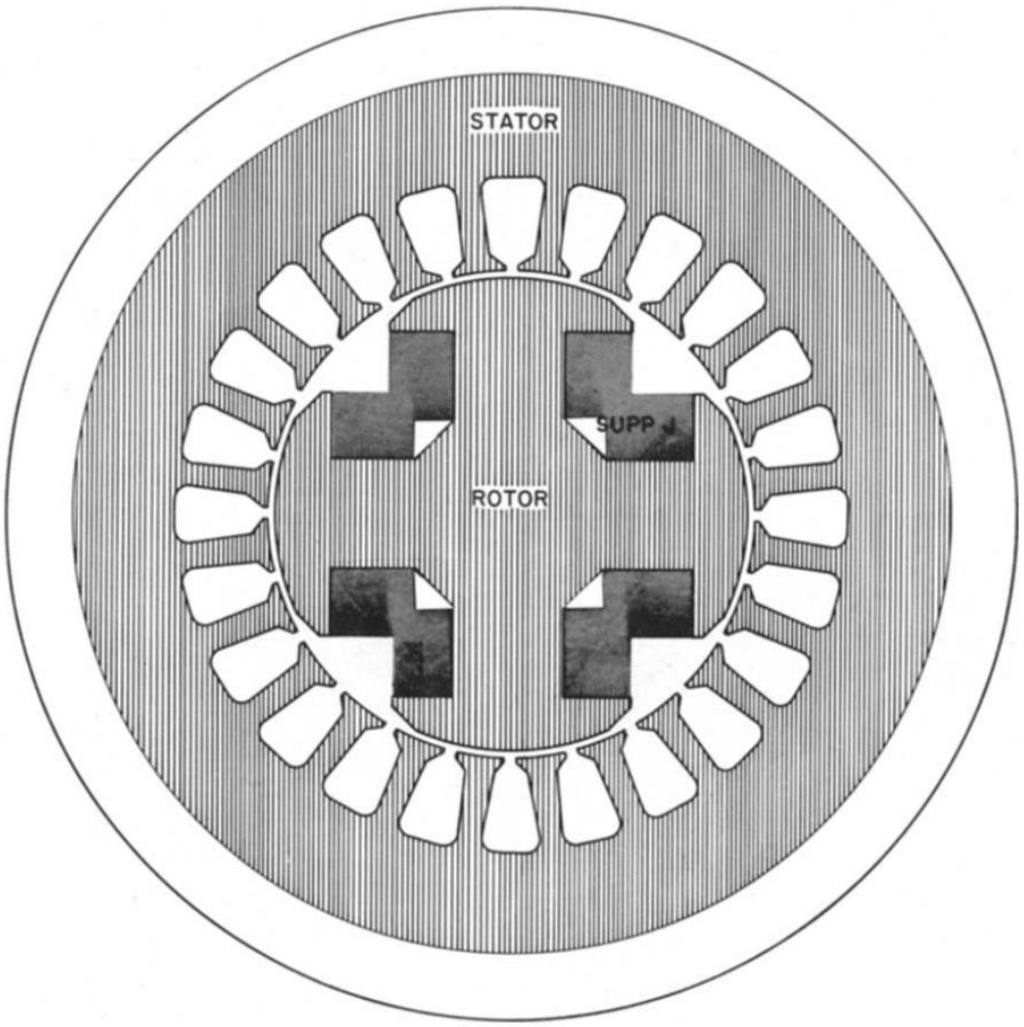
2.2 SALOME

Dalším nástrojem v řetězci je program *SALOME*. *SALOME* je multiplatformní open-source program určený pro preprocessing a postprocessing numerických výpočtů, viz SAL. Krom jiného obsahuje nástroje na práci s geometrií a síťovače, které použijeme.

Jako první krok, který je potřeba učinit, musíme načíst souřadnice bodů, které jsme dostali z *WebPlotDigitizeru*. Bylo by samozřejmě možné načíst body ručně kopírováním ze souboru `points.xy` číslo po čísle, ale bylo by to pracné. Místo toho jsem hledal cestu, jak proces automatizovat. *SALOME* pro tento účel obsahuje funkci *Dump Study*. Ta zapíše právě otevřenou studii do Pythonového skriptu,



Obrázek 2.1: Geometrie alternátoru z R. Glowinski (1974) použitá jako vzor pro vytváření sítě. Zdroj bohužel nezmiňuje rozměry, pro jednoduchost jsme tedy předpokládali, že průměr alternátoru je 1 m.



Obrázek 2.2: Celý alternátor z R. Glowinski (1974). Na obrázku je popsán stator a rotor (svisle šrafované části). Šedá část popsaná jako SUPP J je vinutí cívek rotoru, kde bude proudit proud budící magnetické pole. Vinutí cívek statoru je zanedbáno. Bílé části jsou vzduch.

který je pak možné ze *SALOME* znovu otevřít a skript vytvoří všechny objekty, které původní studie obsahovala. Mezi tím je ovšem možné skript modifikovat tak, aby kromě původní studie načetl i další objekty. Základy Pythonu není těžké si osvojit a tudíž se otvírá možnost velkou část repetitivní práce automatizovat.

Právě tímto způsobem jsem vytvořil Pythonový skript (viz příloha 3.4), který načte ze souboru všechny body předtím odečtené ve *WebPlotDigitizeru*. Většinu skriptu vytvořilo samo *SALOME*, část, kterou bylo třeba měnit jsem vyznačil a stručně okomentoval. Následuje postup tvorby geometrie krok po kroku:

1. Skript spustíme pomocí *File > Load Script*. Tím načteme body ze souboru `points.xy`. Měli bysme se tím dostat do situace nakreslené na obrázku 2.3 vlevo nahoře.

2. Dalším krokem je rozkopírování bodů pomocí funkcí *Operations > Transformation > Mirror Image* a *Operations > Transformation > Rotation*. Jednotlivé operace jistě není třeba rozepisovat dopodrobna. Tím se dostáváme do situace nakreslené na obrázku 2.3 vpravo nahoře.

3. Body pospojujeme pomocí objektu *curve*, kde zaklikneme možnost *Build a closed curve* a manuálně vyklikáme, které body chceme pospojovat. Tím se na obrázku 2.3 posouváme do situace zachycené vlevo dole.

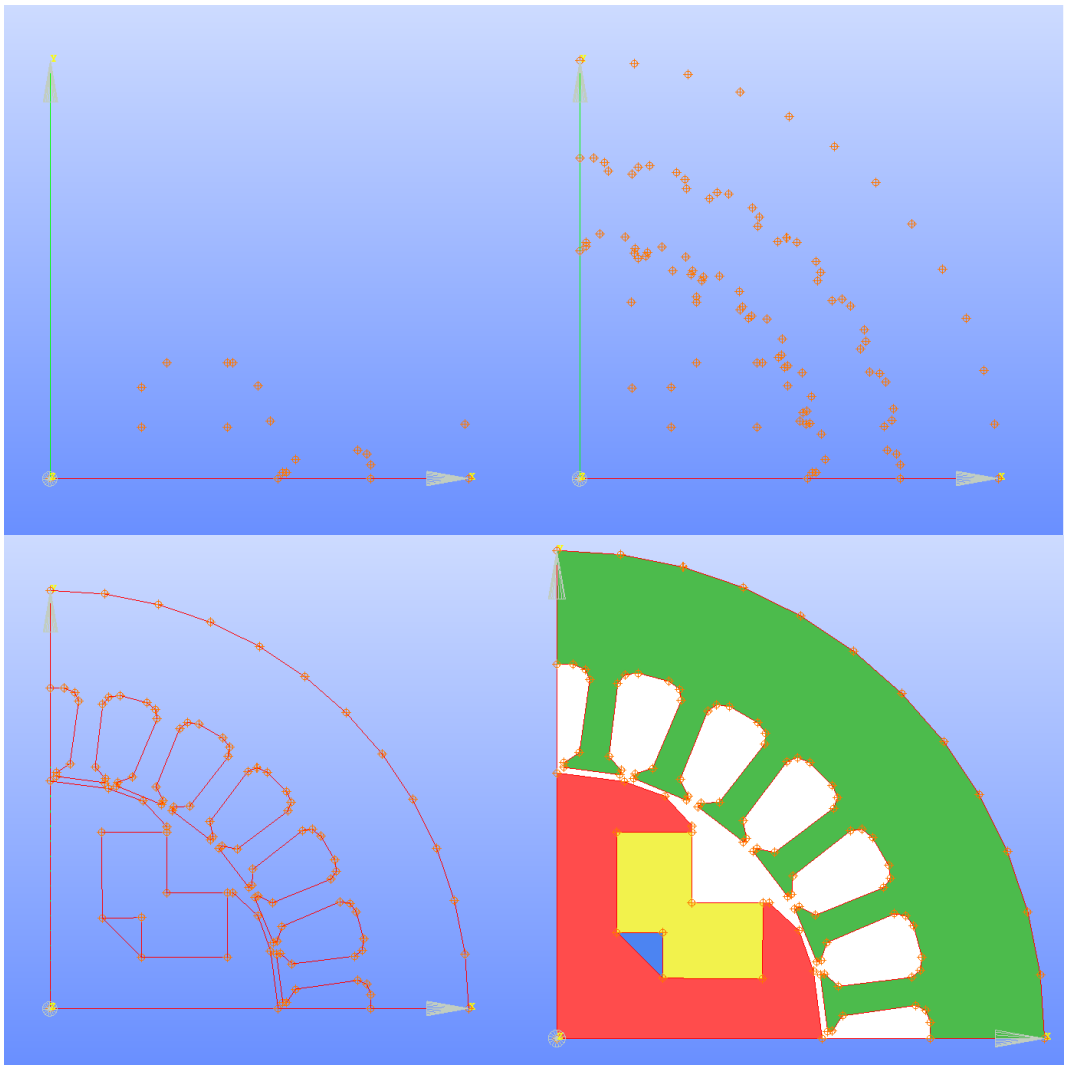
4. Nyní vytvoříme každou oblast zvlášť pomocí objektu *Face*. Tím si zajistíme, že síť bude pro tyto oblasti generována zvlášť a vnitřní hrany oddělující jednotlivé oblasti budou respektovány. Dostáváme se do situace na obrázku 2.3 vpravo dole. Nakonec ještě vytvoříme jeden objekt typu *shell*, do kterého zahrneme všechny objekty typu *face*. Právě na tento objekt se budeme odkazovat během tvorby sítě.

Nyní máme připravenou geometrii a můžeme se pustit do samotného síťování. V *SALOME* přepneme z modulu *Geometry* do modulu *Mesh*. Zvolíme možnost *Create Mesh* a v dialogovém okně vyplníme parametry síťování. V tabulce 2.1 jsem vypsal možnosti, které jsem použil já a které vedou na použitelné sítě, nicméně laskavý čtenář může experimentovat. Jak je vidět u parametru Length, který řídí, jak jemné je dělení hran, použil jsem tři různé hodnoty a vygeneroval tři různé sítě, viz obrázek 2.4. Sítě mají 28328, 7294 resp. 2154 trojúhelníků.

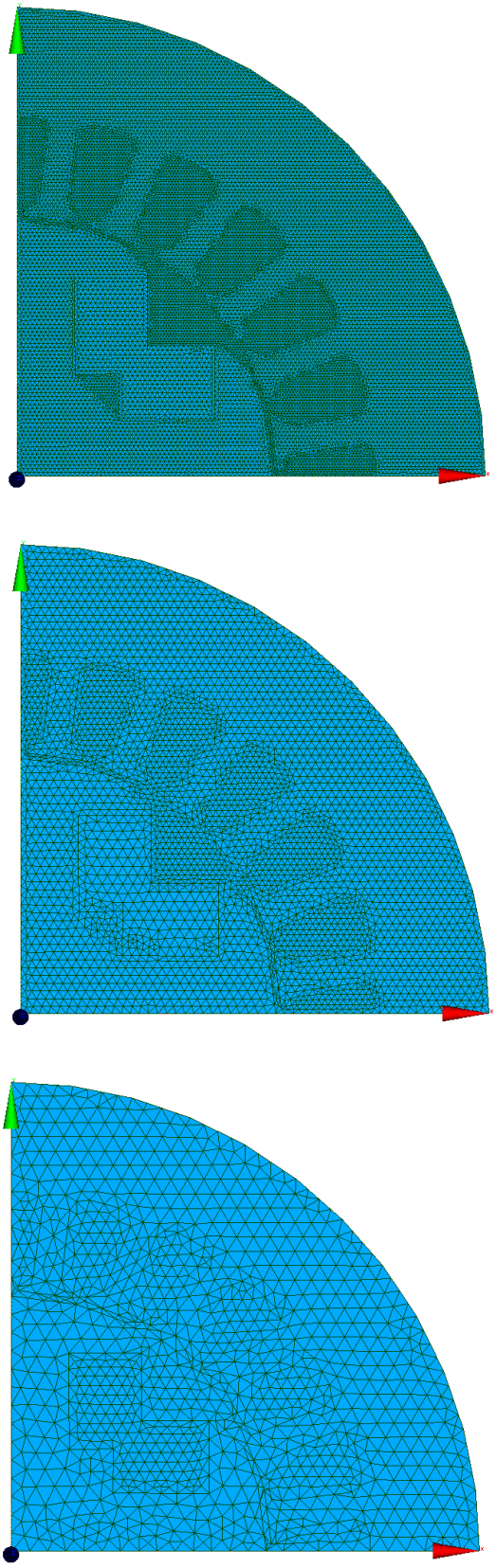
Posledním krokem v *SALOME* je export sítě. Ten uděláme např. pomocí kontextové nabídky vyvolané pravým kliknutím na síť a pak zadáme *Export > DAT file*. *SALOME* umí exportovat síť do mnoha formátů, nicméně .dat je ze všech zdaleka nejjednodušší pro další zacházení. Máme tedy připravené tři sítě na konverzi z formátu DAT do formátu vhodného pro *adgfem*.

2.3 datToAdgfem

Pro účely konverze sítě z formátu .dat jsem napsal program *datToAdgfem*. Tím byl vytvořen důležitý most mezi programem *SALOME*, který je mocným nástrojem na přípravu sítí a programem *adgfem*, který doposud ucelený síťovací nástroj postrádal. Program *datToAdgfem*je napsán v jazyce C++. Dokumentace formátu, který *adgfem* očekává, je v (Dolejší, 2005). Před použitím je potřeba jej zkompilovat. Program je tvořen následujícími soubory:



Obrázek 2.3: Vlevo nahoře: Načtené body ze souboru. Vpravo nahoře: Body rozkopírované pomocí symetrií. Vlevo dole: Body pospojované křivkami. Vpravo dole: Vytvořené oblasti (typ *face*), pro přehlednost ručně obarveno.



Obrázek 2.4: Tři různě jemné sítě, od horní po spodní obsahují 28328, 7294 resp. 2154 trojúhelníků.

classes.cpp je soubor kde jsou implementovány všechny třídy použité v *datToAdgfem*, tedy kde se odehrává většina skutečné práce, kterou *datToAdgfem*vykonává.

classes.h hlavičkový soubor, kde jsou definovány všechny použité třídy. Všechny použité funkce jsou okomentovány, což by mělo usnadnit případné budoucí úpravy.

main.cpp je soubor, který ovládá třídy vytvořené v `classes.cpp` a v podstatě tvoří uživatelské rozhraní, pokud by uživatel nebyl spokojen s tím, jak program funguje a chtěl by si jej upravit. Soubor je opět okomentovaný.

makefile je standardní soubor obsahující nastavení pro kompilaci na Unix-like systémech.

Kompilaci na Unix-like systému za nás obstará utilita *make*. Stačí výše zmíněné soubory zkopírovat do cílové složky a spustit příkaz *make*:

```
>make
g++ -Wall -c main.cpp
g++ -Wall -c classes.cpp
g++ -Wall -o datToAdgfem main.o classes.o
```

Výsledkem pak je spustitelný soubor `datToAdgfem`. Abychom mohli program spustit, očekává v adresáři, ve kterém je spuštěn, dva soubory:

mesh.dat je přímo ten soubor, který exportujeme ze *SALOME*.

components je soubor, který slouží k očíslování komponent, na které chceme síť rozdělit. V našem případě má síť pět částí: stator, rotor, vinutí cívek a dvě části, kde je vzduch. Příklad, jak takový soubor `components` může vypadat:

Parametr	Hodnota
OD	
Algorithm	Segments around Vertex
Hypothesis	Length Near Vertex_1
Length	1
1D	
Algorithm	Wire Discretization
Hypothesis	Local Length_1
Length	0.01; 0.02; 0.04
Precision	1e-7
2D	
Algorithm	Triangle: Mephisto
Hypothesis	Length From Edges_1

Tabulka 2.1: Hodnoty parametrů použité pro síťování v *SALOME*.

```
1 0.01 0.01
2 0.6 0.6
3 0.4 0.2
4 0.2 0.2
4 0.4 0.4
```

V prvním sloupci je číslo, které chceme komponentě přiřadit, *datToAdgfem* očekává kladné celé číslo. Je vidět, že komponenta číslo 4 se opakuje, tudíž se skládá ze dvou částí. Druhý a třetí sloupec je x-ová a y-ová souřednice bodu, který leží uvnitř dané komponenty. Komponenty musí být disjunktní, jinak program skončí chybou. Pokud nějaká komponenta není vybrána, dostane automaticky přiřazeno číslo -1. Tuto funkci není nutné využít, stačí soubor ponechat prázdný.

Máme-li k dispozici výše zmíněné soubory, můžeme program spustit. Výstup z programu by měl vypadat takto:

```
>./datToAdgfem
reading file mesh.dat
ordering the triangles
0 triangles reordered
sorting edges
reordering boundary edges
14351 points, 372 boundary edges, 645 internal edges and 28328\
    triangles read
checking mesh integrity
minimum triangles for one point is 2, average is 5.92182
edges oriented properly
reading file components
creating components
looking for neighbours
creating component 1
creating component 2
creating component 3
creating component 4
creating component 4
writing the mesh into file triang.grid
writing internal boundary into file internalBoundary
printing boundary edges into file boundaryEdges
printing internal edges into file internalEdges
printing internal mesh into file internalMesh
printing component 1 into file internalMesh_1
printing component 2 into file internalMesh_2
printing component 3 into file internalMesh_3
printing component 4 into file internalMesh_4
printing internal mesh into file internalMeshVectors
```

Rozeberme si nyní podrobněji, co se při jednotlivých krocích děje:

1.
```
reading file mesh.dat
ordering the triangles
0 triangles reordered
```

Jsou čtena data ze souboru `mesh.dat`. Trojúhelníky jsou očíslovány proti směru hodinových ručiček, jak *adgfem* očekává. V tomto případě jsou všechny očíslovány správně, ale testováním bysme došli k závěru, že se na to nelze spolehnout.

2.
```
sorting edges
reordering boundary edges
```

Nyní se třídí hrany. Formát .dat nerozlišuje mezi vnitřními hranami a hranami na hranici oblasti, takže je *datToAdgfem* roztřídí. Potom je správně zorientuje, aby byl vnitřek oblasti z pohledu hrany vlevo. Nakonec jsou hrany seřazeny tak, aby na sebe navazovaly jedna za druhou. Obě dvě tyto vlastnosti *adgfem* očekává.

3.
```
14351 points, 372 boundary edges, 645 internal edges and 28328\
    triangles read
checking mesh integrity
minimum triangles for one point is 2, average is 5.92182
edges oriented properly
```

Statistiky a ověřování sítě. Ověřuje se, že každý bod náleží alespoň do jednoho trojúhelníku, že každá hrana náleží nejvýše do dvou trojúhelníků, že existují alespoň tři hrany na hranici oblasti. Kontroluje se, zda jsou hrany skutečně správně srovnány jedna za druhou.

4.
```
reading file components
creating components
looking for neighbours
creating component 1
creating component 2
creating component 3
creating component 4
creating component 4
```

Čte se soubor `components`. Pokud obsahuje alespoň jednu komponentu, *datToAdgfem* vybuduje topologii sítě, tj. najde všem trojúhelníkům všechny sousedy. Tato funkce je implementována velmi jednoduchým, ale pravděpodobně neefektivním způsobem, takže má kvadratickou náročnost v počtu trojúhelníků. Nicméně, pro velikosti sítí, které *adgfem* typicky řeší, je to řešení dostatečné. Po vybudování topologie jsou trojúhelníkům přiřazena čísla komponent.

5.
```
writing the mesh into file triang.grid
writing internal boundary into file internalBoundary
```

Principiální část výstupu. Je zapsán soubor `triang.grid`, což je samotná síť ve formátu již vhodném pro *adgfem*. Kromě standartních informací obsahuje u každého elementu ještě jeden sploupec navíc, kde je zaznamenáno číslo komponenty. Pak se zapíše soubor `internalBoundary`, který obsahuje vnitřní hrany, také ve formátu připraveném pro *adgfem*.

6.
```
printing boundary edges into file boundaryEdges
printing internal edges into file internalEdges
printing internal mesh into file internalMesh
printing component 1 into file internalMesh_1
printing component 2 into file internalMesh_2
printing component 3 into file internalMesh_3
printing component 4 into file internalMesh_4
printing internal mesh into file internalMeshVectors
```

Vedlejší výstup. Zapisují se soubory `boundaryEdges`, `internalEdges`, `internalMesh`, `internalMeshVectors` a pro každou komponentu soubor `internalMesh_n`, kde n je číslo komponenty. Tyto soubory jsou určeny pro snadnější vizualizaci a kontrolu výsledné sítě.

2.4 Vizualizace výsledné sítě

Kromě souborů `classes.cpp`, `classes.h`, `main.cpp` a `makefile`, které jsou přímo součástí *datToAdgfem*, jsem napsal ještě jednoduchý skript `mesh.gp` pro *gnuplot*, který využívá soubory `boundaryEdges`, `internalEdges` a `internalMesh`. Díky němu je možné snadno vizuálně zkontrolovat výslednou síť a orientaci hran na hranici. Výsledek můžete vidět na obrázku 2.5.

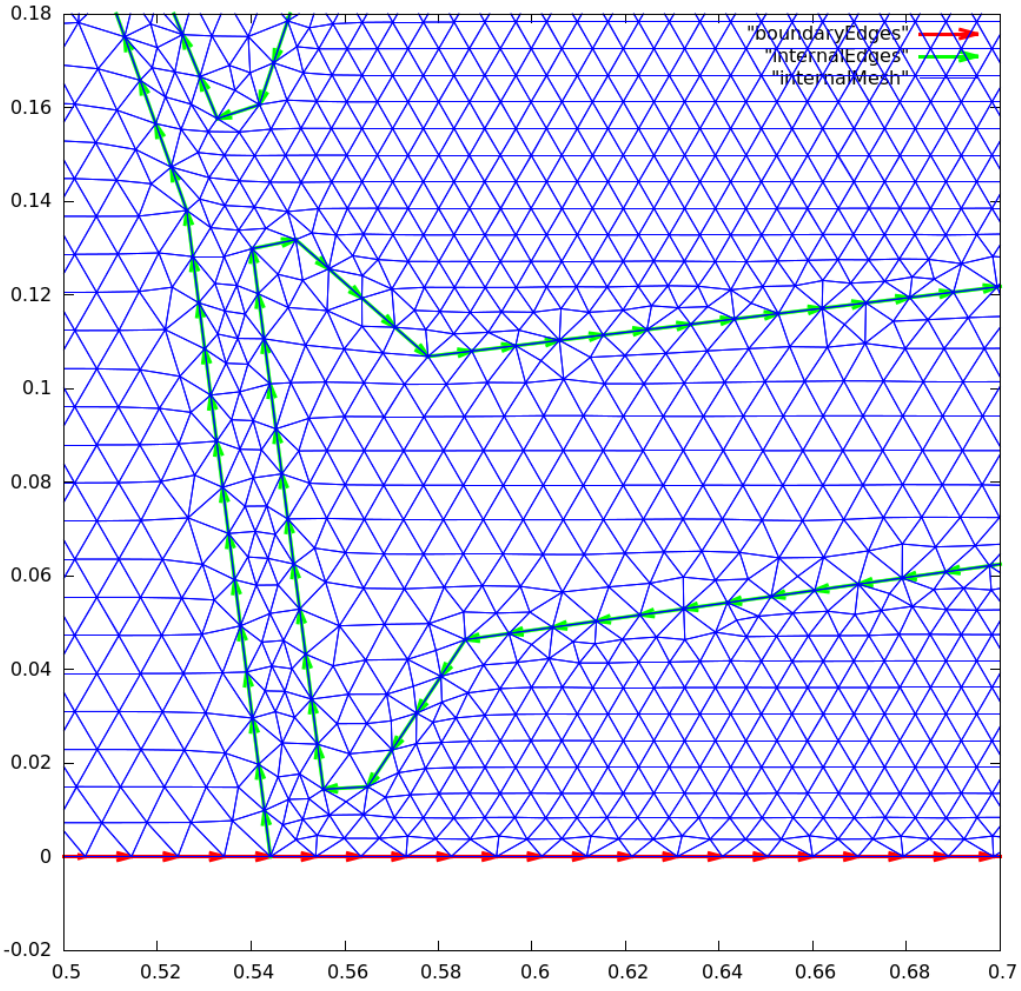
Podobný smysl mají i soubory `internalMesh_n`, které obsahují jen trojúhelníky z určité komponenty a lze je vykreslit (viz 2.6) jediným příkazem v gnuplotu:

```
gnuplot> plot "internalMesh_4" with lines ti "komponenta 4"
```

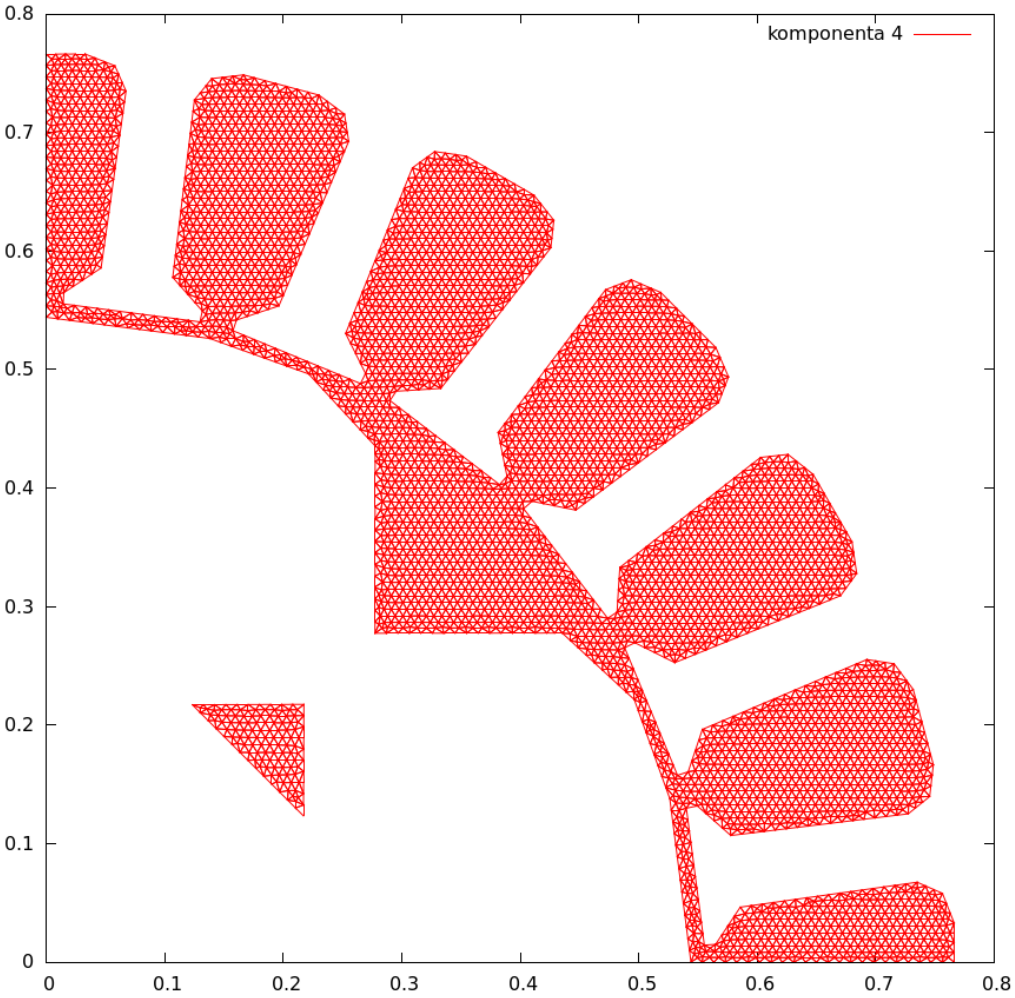
2.5 Omezení

Program *datToAdgfem* má zatím nějaká omezení, na která by uživatel mohl narazit:

- Program *datToAdgfem* zvládá pouze trojúhelníky.
- Hranice sítě musí být jednoduše souvislá, jinak ji *datToAdgfem* nedokáže správně uspořádat.
- Nezvládá křivočaré hranice. Nicméně uživatel se s nimi může vypořádat ručně.
- Výroba komponent může být pomalá pro velké sítě (cca nad 100 000).



Obrázek 2.5: Obrázek sítě, výřez. Hrany na hranici sítě (červené šipky) jsou správně orientovaná vůči vnitřku sítě.



Obrázek 2.6: Vykreslená čtvrtá komponenta.

3. Výpočty v adgfem

Veškerou práci s *adgfem*, o které budu v této kapitole mluvit, jsem prováděl s revizí 1404. Bohužel, aktuální revizi se mi nepodařilo úspěšně zkompilovat.

3.1 Ověření funkčnosti

Jako první jsem provedl výpočet na velmi hrubé síti se zapnutou adaptivitou, abych ověřil, že jak soubor `triang.grid`, tak soubor `internalBoundary` jsou v pořádku a *adgfem* je dokáže načíst. Nastavení inicializačního souboru si můžete prohlédnout zde:

```
scalar   1.0000E+00   1   0.0000E+00
1E+20
1.00E-08  1.00E+05  1.00E+06  1.00E+05
2   'triang.grid'
1   'arc_sector.prof'   'internalBoundary'
 1       G0.rsol
test
IIPG    4.000E+02    1
BDF  1
adapt   1.000E-12   tRES
RES    1.0000E-12  1.000E-01  -1.00   0
AMAhp  0003     n
5
Newton  aRES   1.0000E-03    60    20
none     GMRES_ILU   1E-3
0.0000E-00   0
1
1    0  1.0000E+00
2.00   0.00   0.50   0.00
0.0    0.0   0.00    0
```

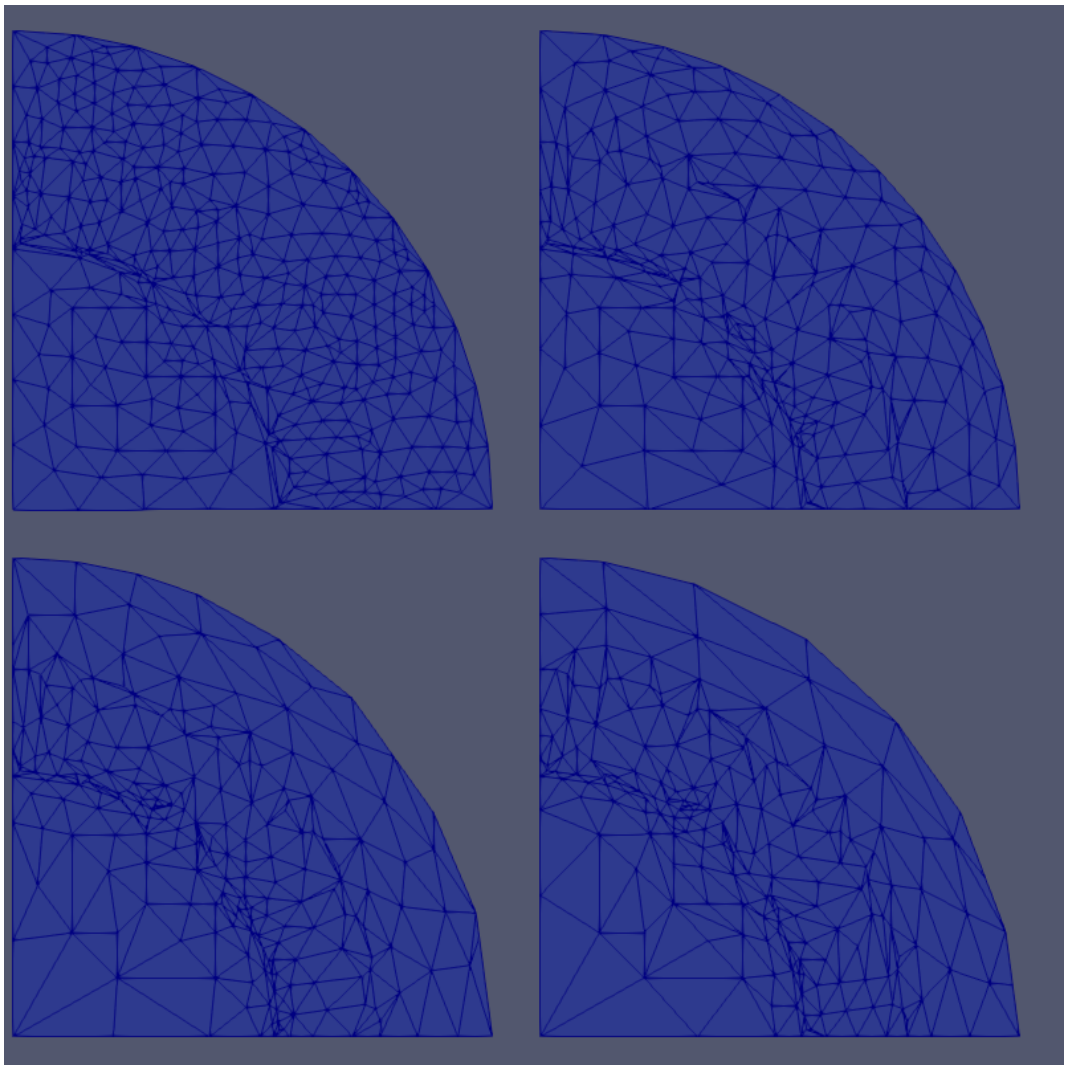
Výpočet je nefyzikální, jedná se o Laplaceovu rovnici s okrajovou podmínkou nastavenou tak, aby přesné řešení bylo $u(x) = x$. Nicméně i tento jednoduchý výpočet posloužil k ověření funkčnosti *datToAdgfem*. Postupnou adaptaci sítě můžete vidět na obrázku 3.1. Všechny soubory použité k tomuto výpočtu jsou k nalezení v elektronické příloze.

3.2 Magnetické pole

Abych se mohl pustit do výpočtu mangetického pole, bylo potřeba učinit tři zásadní kroky:

1. Bylo nejprve potřeba *adgfem* upravit tak, aby uměl načíst informaci o tom, který trojúhelník patří do které komponenty.

2. Bylo třeba v souboru `model.f90` a `scalar.f90` vytvořit nový model, který by odpovídal zadání a odkazoval na správnou pravou pravou stranu a správný difusní koeficient.



Obrázek 3.1: Ověření funkčnosti konvertoru, zleva doprave, zhora dolů, postupně se adaptující síť.

3. A konečně, bylo třeba vytvořit funkce pro pravou stranu a pro difusní koeficient, které mají přístup k informaci o své komponentě a podle toho nastaví správnou hodnotu.

Bohužel, třetího kroku se mi nepodařilo dosáhnout - přes všechno úsilí se mi nepodařilo spustit výpočet, jehož koeficienty by respektovaly rozložení komponent. Návrh prof. Dolejšího, abych koeficienty při přechodu z oblasti do oblasti shladil se sice zdál nadějný, ale bohužel jsem ho už nestihl implementovat. Musle jsem tedy ze svého cíle slevit. Implementoval jsem tedy aspoň nelineární výpočet tak, jako kdyby celou oblast pokrývala ocel, tedy místo vztahu (1.1) máme

$$\nu(x,\eta) = \frac{1}{\mu_0}\left(\alpha + (1-\alpha)\frac{\eta^4}{\eta^4+\beta}\right) \qquad \forall x \in \Omega. \tag{3.1}$$

Všechny potřebné soubory k replikování výpočtu jsou zahrnuty v elektronické příloze.

Závěr

Přestože se nakonec nepodařilo zprovoznit výpočty v magnetohydrodynamice a i výsledky s nelineárním magnetickým polem jsou pouze částečné, tato práce přinesla svoje nečekané ovoce ve formě nástroje na přípravu sítí pro současné i budoucí uživatele *adgfem*. Navíc, pokud by se ukázalo, že je o takový nástroj zájem, *datToAdgfem* určitě skýtá možnosti pro další rozšíření, vyšší míru automatizace a robustnosti.

Seznam použité literatury

Salome. http://www.salome-platform.org/. dne: 1. 2. 2017.

Dolejší, V. (2005). Angener user's guide. http://www.karlin.mff.cuni.cz/~dolejsi/angen/angen3.1.htm. Version 3.1.

Křížek M., S. K. (2001). *Numerické modelování problémů elektrotechniky*. Karolinum, Praha. ISBN 80-246-0287-3.

R. Glowinski, A. M. (1974). Analyse numerique du champ magnetique d'un alternateur par elements finis et sur-relaxation ponctuelle non lineaire. *Computer Methods in Applied Mechanics and Engineering*, **3**, 55–85.

Rohatgi, A. Webplotdigitizer. http://arohatgi.info/WebPlotDigitizer/. dne: 1. 2. 2017.

Vivette Girault, P.-A. R. (1986). *Finite Element Methods for Navier-Stokes Equations*. Springer-Verlag. ISBN 978-3-642-64888-5.

Vít Dolejší, M. F. (2015). *Discontinuous Galerkin Method*. Springer International Publishing. ISBN 978-3-319-19266-6.

Seznam obrázků

Přílohy

3.3 Souřadnice vrcholů geometrie

Toto je výpis souboru se souřednicemi získaný z *WebPlotDigitizer*. V prvním sloupci je x-ová souřadnice, ve druhém y-ová.

```
-0.0008725605416075102 -0.00043056036216104765
0.5438473935999029 -0.00036772300538401304
0.54338591605173 0.020503328024061185
0.5415701677902928 0.056044137017311035
0.5377003920105186 0.08256753396358907
0.5307390182773003 0.12038255257332375
0.5228758027996142 0.15030419377411885
0.5115952405109503 0.18418609654844786
0.4985954481408592 0.21807403170902728
0.437013833101381 0.27638609340066833
0.42211383906234373 0.27643837408150695
0.4218574626466923 0.12132863181696296
0.21612594576234231 0.12261453548605394
0.12321211653713157 0.21600693402048077
0.12144462736569922 0.42132322165124614
0.2773184718886555 0.42021225718342314
0.2779548886380961 0.43205483679110335
0.22156565006384102 0.49683512463997404
0.17781576878120248 0.5139097939223605
0.12804104173075478 0.5295955062705413
0.07451266233585732 0.5399360217018165
0.0006068821863576199 0.5441436111116256
```

3.4 Skript pro načtení souřadnic do SALOME

Toto je skript v Pythonu použitý k načtení souboru se souřadnicemi do *SALOME*.

```
# -*- coding: utf-8 -*-

###
### This file is generated automatically by SALOME v8.2.0
### with dump python functionality
###

import sys
import salome

salome.salome_init()
theStudy = salome.myStudy

import salome_notebook
```

```
notebook = salome_notebook.NoteBook(theStudy)
sys.path.insert( 0, r'/home/vojta/Meshing/alternator')

###
### GEOM component
###

import GEOM
from salome.geom import geomBuilder
import math
import SALOMEDS


geompy = geomBuilder.New(theStudy)

O = geompy.MakeVertex(0, 0, 0)
OX = geompy.MakeVectorDXDYDZ(1, 0, 0)
OY = geompy.MakeVectorDXDYDZ(0, 1, 0)
OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)
geompy.addToStudy( O, 'O' )
geompy.addToStudy( OX, 'OX' )
geompy.addToStudy( OY, 'OY' )
geompy.addToStudy( OZ, 'OZ' )

#
# Begining of user edited part
#

# open file with coordinates
f = open("points.xy")

# counter of points
n = 0

# for each line in file
for l in f:

    # parse line into variables x and y
    x, y = [ float(v) for v in l.split() ]

    # create new vertex,
    # x, y are coordinates from previous step, z coordinate is 0
    pt = geompy.MakeVertex(x, y, 0)

    # add new vertex to the study
    geompy.addToStudy(pt, "Pt_%s"%(n))
    n += 1
    pass

# update GUI, so user can see the points
import salome
```

```
salome.sg.updateObjBrowser(0)

#
# End of user edited part
#

if salome.sg.hasDesktop():
  salome.sg.updateObjBrowser(True)
```

3.5 Přiložené soubory

Elekronická příloha obsahuje celkem čtyři složky. První z nich je `datToAdgfem` spolu se vzorovou sítí ve formátu .dat a gnuplotovým skriptem `mesh.gp`. Další dvě jsou `adaptivni_case` a `nelinearni_case`, kde jsou všechny soubory nutné pro spuštění výpočtů diskutovaných v kapitole 3. Poslední složka obsahuje všechny soubory, které jsem při své práci v *adgfem* změnil oproti revizi 1404.