Podklady k semináru

Open-Source nástroje pre FEM

Peter Fabo

15. februára 2019

Obsah

Kapitola 1

Open-Source nástroje pre FEM

Seminár je venovaný prehľadu open-source nástrojov použiteľných pre FEM simulácie v akademickom prostredí.

1.1 Obsah semináru

- Prehľad dostupných programov a nástrojov
- Výmena dát medzi programami, používané formáty
- Tvorba modelu a generovanie mriežky
- Nastavenie simulácie
- Spracovanie a vizualizácia výsledkov simulácie

1.2 Podklady k semináru

Zdrojové texty vo formáte Jupyter-Notebook su dostupné na

`https://github.com/pfabo/notebook-fem-opensource`

Textová pdf verzia je vygenerovaný z Jupyter-Notebook-ov pomocou skriptu pdf2ltx

`https://github.com/pfabo/jupyter-to-latex`

1.3 Použitý software

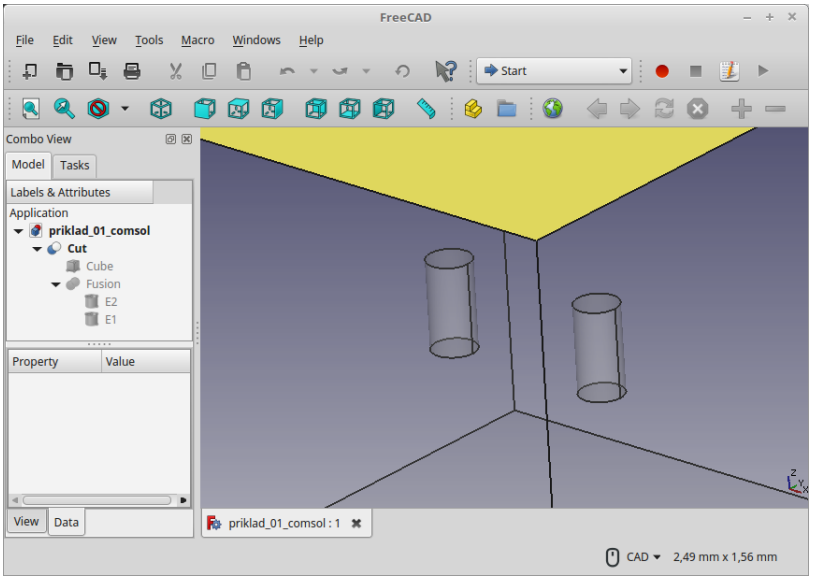
V rámci semináru sú použité nasledujúce programy a nástroje.

- Jupyter Notebook http://jupyter.org/index.html
- Gmsh http://gmsh.info/
- Elmer-CSC https://www.csc.fi/web/elmer
- ParaView https://www.paraview.org/
- FreeCAD https://www.freecadweb.org/

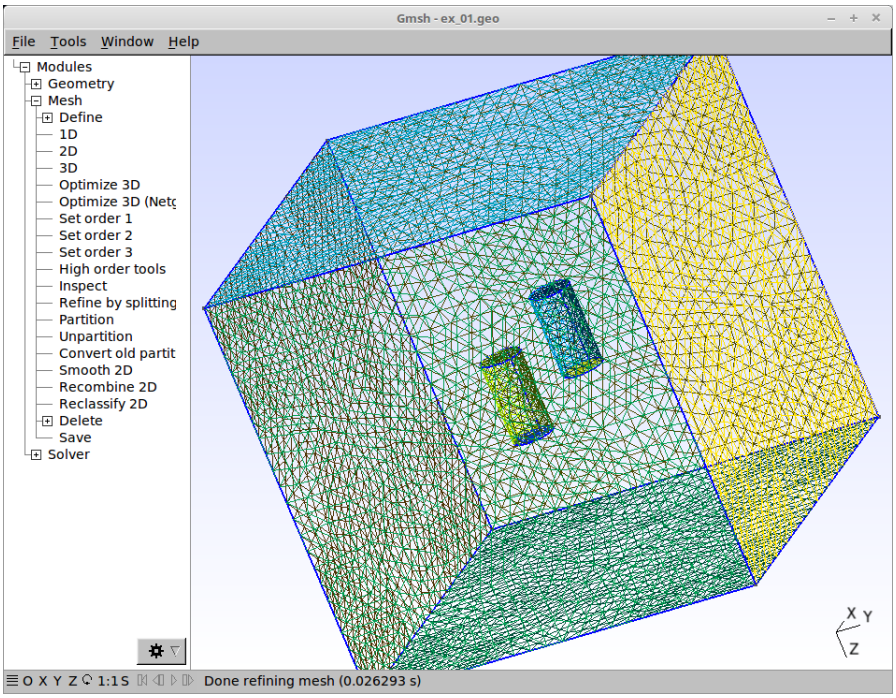
Programy sú prioritne vytvorené a kompilované pre OS Linux a štandardne sa nachádzajú v repozitároch distribúcií. Pretože do repozitárov distribúcií sa balíky zaraďujú s nejakým časovým odstupom, pre aktuálne verzie je lepšie použiť návod na inštaláciu resp. stiahnutie skopilovaných verzií priamo na stránkach projektov (inštalácia cez ppa, pip a pod.).

1.4 Demo

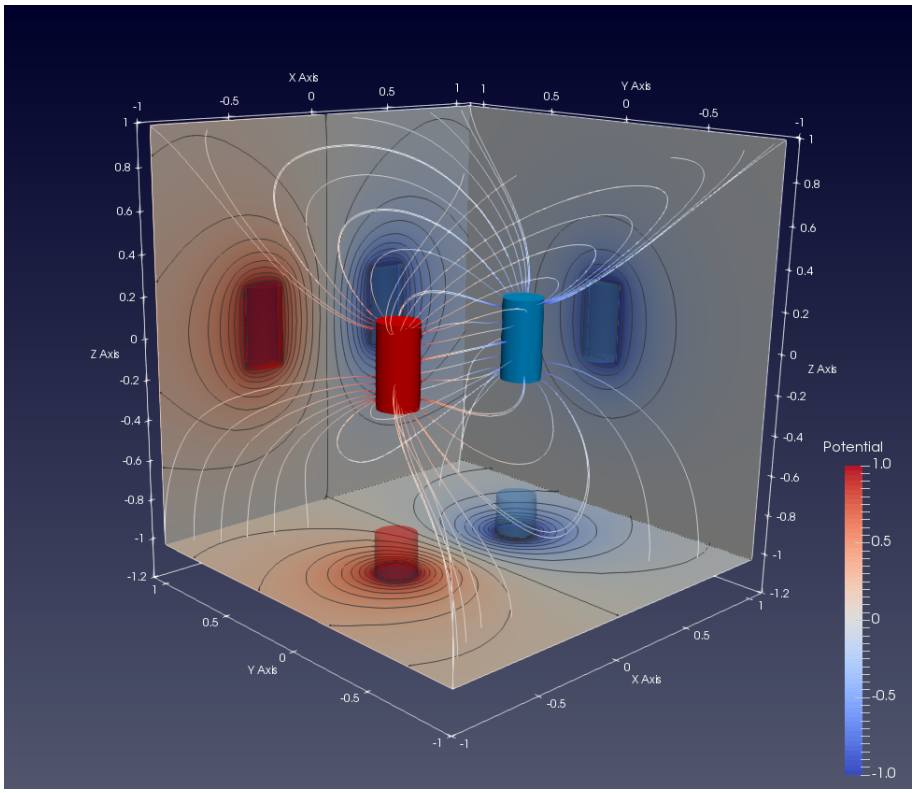
Na úvod krátka ukážka využitia open-source nástrojov. 3D model problému vytvorený v programe *FreeCAD* - dva vodivé valce s opačným potenciálom. Generátor siete *gsmh* vygeneroval sieť povrchov objektov, simulácia elektrostatických polí bola spočítaná v programe Elmer-CSC. Výsledok simulácie bol vizualizovaný pomocou programu *ParaView*.



Obr. 1.1: 3D model problému



Obr. 1.2: Generátor siete



Obr. 1.3: Výsledok simulácie

Kapitola 2

Pre-Processing

Úlohou predprocesora je vytvorenie simulačného modelu, prostredia a vygenerovanie vstupných dát pre simulačné prostredie. Vytvorený model sa často používa po simulácii v post-procesore pri zobrazení výsledkov, pretože výsledkom simulácie bývajú zvyčajne dáta o simulovanom jave (napr. intenzita elektrického poľa) a vlastný model nemusí byť súčasťou týchto dát.

2.1 Vytvorenie modelu pomocou programu *gmsh*

Gmsh je 2D/3D generátor siete modelu, je možno ho použiť aj na zobrazenie výsledkov simulácií. Je súčasťou komplexného prostredia OneLab, v ktorom je integrovaný univerzálny solver pre riešenie problémov. Aktuálne je podľa štatistík (Elmer) najpoužívanejším multifyzikálnym simulátorom v akademickom prostredí (44%).

Vytvorenie modelu pozostáva z definovania elementárnych entít - bodov, spojovacích línií, vytvorení plôch a objemov. Na základe elementárnych entít sú definované fyzické skupiny, z ktorých sú potom generované siete pre konečné prvky. Fyzické skupiny môžu pozostávať z viacerých elementárnych entít.

Pre vytvorenie modelu môžeme použiť niekoľko postupov, prípadne ich vzájomne kombinovať.

2.2 Vytvorenie modelu manuálne v GUI

Manuálnym zadaním je vhodné vytvárať len veľmi jednoduché modely pre overenie funkčnosti prostredia (aj keď tvorcovia *gmsh* sú iného názoru), konfigurácií a v pedagogickom procese pri demonštrácii princípov pri vytváraní modelu.

```
import os
_ = os.system("rm *.geo")
_ = os.system("gmsh")
```

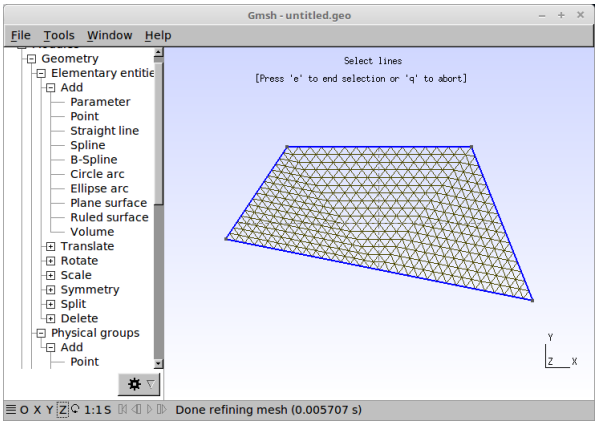
2.2.1 Postup

- Vytvorenie sústavy bodov
  - *Modules -> Geometry -> Elementary entities -> Add -> Point*
- Vytvorenie prepojovacích línií - sú orientované (začiatok - koniec)
  - *Modules -> Geometry -> Elementary entities -> Add -> Line*
- Vytvorenie plôch - 2D plocha
  - *Modules -> Geometry -> Elementary entities -> Add -> Plane surface*
  - Referenčným bodom plochy je jej stred

- Konverzia elementárnej plochy na fyzický komponent (môže pozostávať napr. z viacerých logických plôch a pod.)
  - *Modules -> Geometry -> Physical groups -> Add -> Surface*
- Vytvorenie siete
  - *Mesh -> Define - 2D*
- Zjemnenie siete
  - *Mesh -> Define - Refine by splitting*

Výsledok vytvárania modelu a siete sa automaticky zapisuje do súboru **.geo*, ktorý je možné editovať a použiť ako skript (nižšie).

2.2.2 Príklad vygenerovanej siete



Obr. 2.1: Manuálne vygenerovaná sieť

2.3 Vytvorenie modelu skriptom

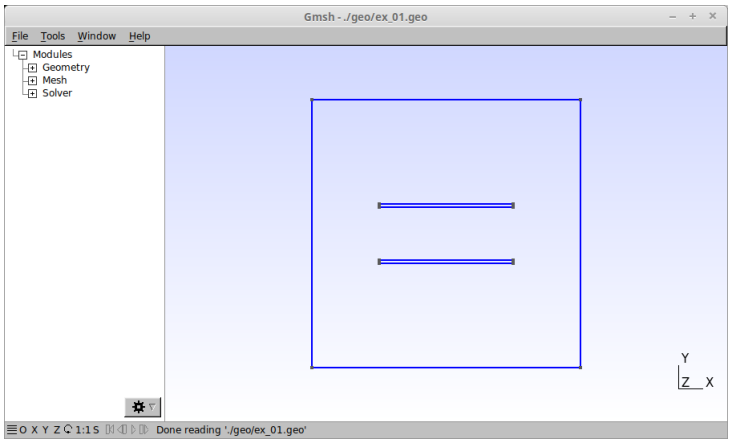
Gmsh má vlastný skriptovací jazyk pre definovanie modelov, jednotlivé príkazy sa zadávajú do súboru, ktorý sa načíta pri spustení programu. Pri zadaní bodov zárovň definujeme aj hustotu generovanej siete v okolí bodu.

2.3.1 Príklad fragmentu skriptu zo súboru *ex_01.geo*

```
// elementarne entity
Point(1) = {-10,  10, 0, 1.0};          // bod x,y,z,hustota siete v okoli bodu
Point(4) = { 10,  10, 0, 1.0};
Line(1) = {1, 4};                       // usecka z dvoch bodov
                                        // vytvorenie obrysu - uzatvorenej krivky
Line Loop(5) = {3, 4, 1, 2, 13, 14, 11, 12, 23, 24, 21, 22};
Plane Surface(6) = {5};                 // vytvorenie plochy z obrysu...

// fyzicke entity
Physical Line("e1") = {11, 14, 13, 12}; // lomena ciara zlozena z useciek
..
Physical Surface("air") = {6};          // fyzicka plocha - moze byt zlozena z viacerych
```

2.3.2 Model vygenerovaný skriptom *ex_01.geo*

```
_ = os.system("gmsh ./data/ex_01.geo")
```



Obr. 2.2: Model vytvorený skriptom

2.4 Vytvorenie parametrického modelu skriptom

Pre problémy spojené so zmenou parametrov modelu, napríklad pre potreby optimalizácie, je potrebné mať model alebo jeho časti zadané pomocou parametra, ktorý môžeme meniť. Pre generovanie parametrického modelu existujú podporné programy a knižnice, jedným z nich je knižnica pygmsh.

V *pygmsh* je možné model vytvoriť pomocou API, ktoré pozostáva z dvoch častí

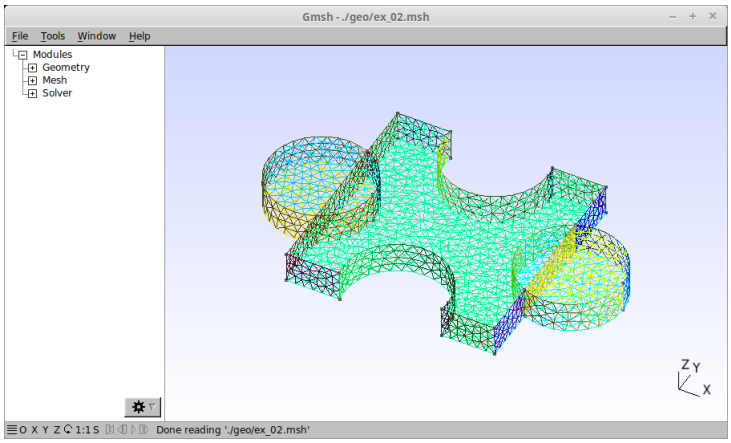
- emulujúcich príkazy skriptu *gmsh* (add_point, add_line ...)
- funkcií kompatibilných s openCASCADE (add_ball, add_box ...), tieto sú primárne určené pre modelovanie 3D objektov

```
# Priklad vytvorenie modelu pomocou openCASCADE
# Vytvorenie 2D struktury a extrudovanie do 3D modelu

import pygmsh

geom = pygmsh.opencascade.Geometry(
  characteristic_length_min=0.1,
  characteristic_length_max=0.1,
  )

# Vytvorenie 2D objektov na ploche
rectangle = geom.add_rectangle([-1.0, -1.0, 0.0], 2.0, 2.0)
disk1 = geom.add_disk([-1.2, 0.0, 0.0], 0.5)
disk2 = geom.add_disk([+1.2, 0.0, 0.0], 0.5)
union = geom.boolean_union([rectangle, disk1, disk2])

disk3 = geom.add_disk([0.0, -0.9, 0.0], 0.5)
disk4 = geom.add_disk([0.0, +0.9, 0.0], 0.5, char_length=0.1)
flat = geom.boolean_difference([union], [disk3, disk4])

# Vytvorenie 3D objektu
geom.extrude(flat, [0, 0, 0.8])

# zapis vygenerovaneho objektu v gmsh formate
fd = open('./data/ex_02.geo', 'w')
fd.write(geom.get_code())
fd.close()

# vygenerovanie siete a zapis do suboru *.msh
points, cells, point_data, cell_data, field_data = pygmsh.generate_mesh(geom,
    verbose=False)
```

```
# zobrazenie vygenerovaneho *.geo suboru
_ = os.system("gmsh ./data/ex_02.geo")
```

```
# zobrazenie vygenerovanej siete
# !!! mriezka neobsahuje fyzicke entity
_ = os.system("gmsh ./data/ex_02.msh")
```

2.4.1 Model vygenerovaný skriptom

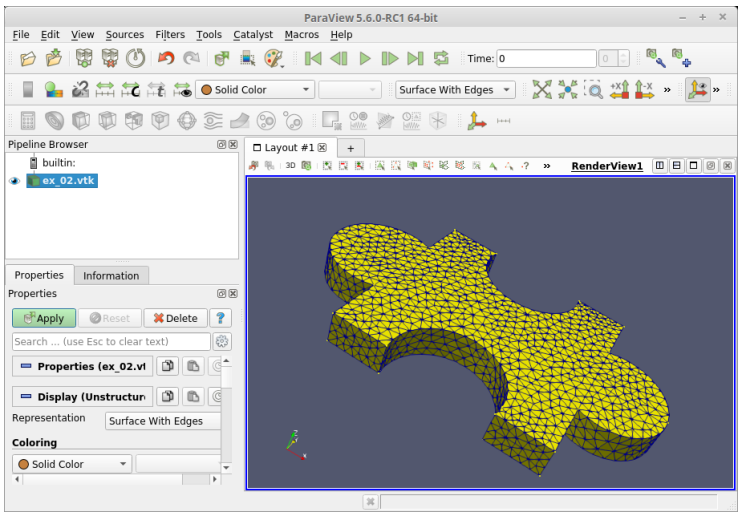


Obr. 2.3: Programovo vygenerovaný model

V *gmsh* môžeme vytvoriť sieť a objekt exportovať do formátu *VTK* (*Visualisation Toolkit*), ktorý vie načítať program pre pos-processing *ParaView*.

```
_ = os.system("paraview ./data/ex_02.vtk")
```

2.4.2 Zobrazenie modelu v ParaView

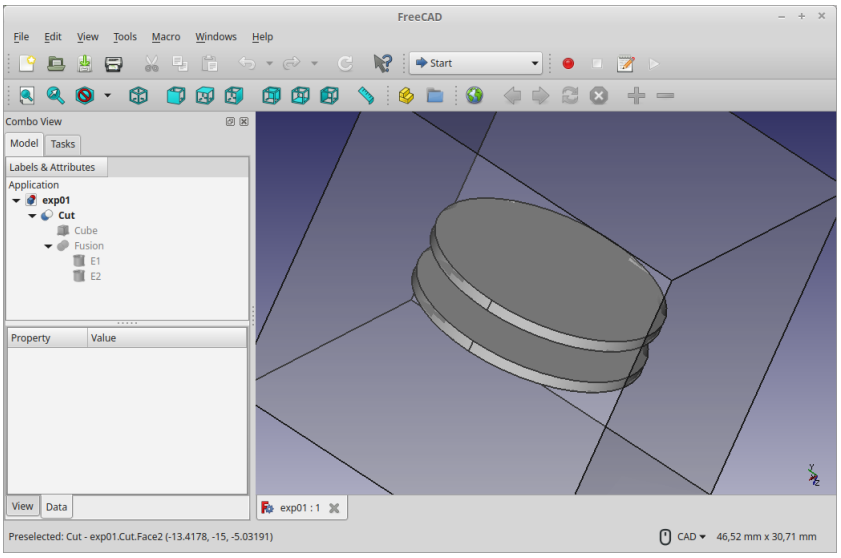


Obr. 2.4: Vizualizácia modelu

2.5 Tvorba modelu pomocou CAD programov

Komplexnejšie modely je možné tvoriť pomocou open-source CAD programov, FreeCAD poskytuje aj možnosť skriptovania modelu v Pythone. Typ modelu závisí od predpokladaného typu simulácie, napr. pre simuláciu elektrostatického poľa - kde nás zaujíma tvar elektrického poľa v prostredí - vytvárame model v tvare dutín v prostredí, kde jednotlivé plochy dutín reprezentujú elekródy so stanoveným potenciálom alebo okrajové podmienky.

2.5.1 Vytvorenie modelu vo FreeCAD



Obr. 2.5: FreeCad model

```
_ = os.system("freecad ./data/ex_03.fcstd")
```

Vo FreeCad-e exportujeme model vo formáte BREP, ktorý vie čítať generátor mriežky *gmsh*.

```
_ = os.system("gmsh ./data/ex_03.brep")
```

Súbor **.brep* obsahuje model rozložený na elementárne entity, neobsahuje ale fyzické entity. Mená objektov z FreeCAD-u nie sú súčasťou formátu **.brep* (a zrejme by to prinášalo problémy), preto musíme jednotlivé fyzické entity "poskladať"ručne - doplnením do súboru **.geo*.

Zásadným rozdielom medzi elementárnou a fyzickou entitou je tem, že jedna fyzická entita (napr. elektróda, zemná plocha atď.) môže byž poskladaná z viacerých fyzických entít.

Naopak - jeden objekt v CAD-e (napr. valec) môže (ako sústava 3 entít - dno, dekel, plášť) reprezentovať samostatné fyzické entity (2 elektródy spojené izolátorom).

2.5.2 Konfigurácia komverzie *ex_03.geo*

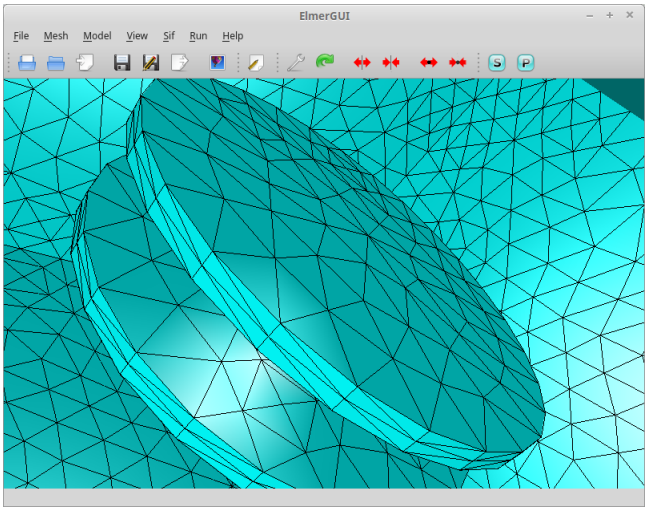
Do súboru importujeme vygenerovaný súbor *ex_03.brep* a doplníme definície fyzických entít. Nepríjemné je hlavne určenie jednotlivých plôch objektov, naštastie sú generované v jednotlivých objektoch v poradí a pomôcť si môžeme aj zobrazením označenia entít v *gmsh*.

```
Merge "exp_03.brep";
Physical Volume("air") = {1};

Physical Surface("s2")  = {7,8,9};
Physical Surface("q1")  = {10,11,12};
Physical Surface("ext") = {1,2,3,4,5,6};
```

```
_ = os.system("gmsh ./data/ex_03.geo")
```

2.6 Generovanie siete

V program *gmsh* môžeme vytvárať mriežku podľa potreby jej zjemnením, vyhladením a pod. Program (ver. 4.0 a vyššia) automaticky pri ukladaní zvolí formát mriežky vo verzii 4, ktorú simulačný engine ElmerCSC (zatiaľ) nepozná, preto je potrebné pri ukladaní zvoliť export mriežky vo formáte 2.0 (ASCII 2).
Vygenerovanú sieť môžeme skontrolovať v grafickom rozhraní ElmerGUI (s obmedzeniami ...).



Obr. 2.6: Zobrazenie siete v ElmerGUI

2.7 Generovanie siete skriptom

Pri častejších zmenách modelu je ručné opakované generovanie mriežky nepríjemné, môžeme si pomôcť vygenerovaním siete priamo v súbore **.geo*

2.7.1 Konfigurácia automatického generovania siete *ex_04.geo*

```
Merge "ex_03.brep";
Physical Volume("air") = {1};

Physical Surface("s2")  = {7,8,9};
Physical Surface("q1")  = {10,11,12};
Physical Surface("ext") = {1,2,3,4,5,6};

Mesh 3;
RefineMesh;
RefineMesh;
```

Sieť vygenerujeme a zapíšeme v správnom formáte v povelovom riadku príkazom

```
gmsh ex_04.geo -3 -format msh2 -save_all -o ex_04.msh
```

Kapitola 3

Simulačné nástroje

3.1 Kompaktné prostredia

- http://www.petr-lorenz.com/emgine/ mikropáskové vedenia
- https://www.freecadweb.org/ parametrický CAD s možnosťou FEM simulácie - mechanika
- http://www.agros2d.org/ EM, štrukturálna mechanika, prúdenie kvapalín
- http://onelab.info/ gmsh + GetDP solver
- ...

3.2 FEM knižnice a solvery

- https://goma.github.io/ Sandia Research Lab., mechanika, prúdenie kvapalín, kapilárne javy, odparovanie ... (solver)
- https://www.dealii.org/ univerzálny FEM simulátor pre PDE (knižnica)
- https://github.com/mfem/mfem univerzálny FEM simulátor pre PDE (knižnica)
- http://oofem.org/ štrukturálna mechanika (solver)
- https://code-aster.org/ termomechanika (solver)
- http://getfem.org/ univerzálny FEM simulátor, Python interface (solver)
- http://www.hpfem.org/hermes/ univerzálny FEM simulátor pre PDE (C++ knižnica)
- http://www.dhondt.de/ Calculix univerzálny FEM simulátor (solver)
- ...

3.3 Simulátory

- https://projectchrono.org/ multifyzikálny simulátor, FEM, dynamika ... (knižnica, C++, Python)
- ...

Kapitola 4

Elmer-CSC

Elmer je open-source multifyzikálny simulačný solver vyvíjaný CSC-IT Center for Science (1995) v spolupráci fínskymi univerzitami a ďalšími organizáciam, od roku 2005 je open-source. Obsahuje fyzikálne modely pre dynamiku kvapalín, štrukturálnu mechaniku, EM, transport tepla a akustiku.

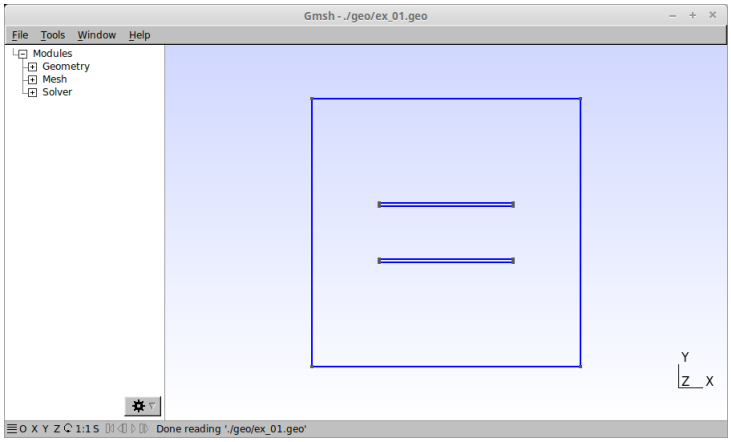
4.1 Inštalácia

V Linuxe je súčasťou distribúcií v balíku *elmer-csc-eg* (Linux Mint 19.0), tento balík obsahuje jednoduché prostredie pre generovanie povelových súborov.

4.2 Jednoduchá elektrostatická simulácia

Vstupnými dátami pre *Elmer* je vygenerovaná sieť s definovanými a pomenovanými fyzickými entitami. Použijeme sieť vygenerovanú z ppríkladu *ex_01.geo*. V príklade budeme simulovať 2D elektrostatické pole dvoch elektród na rôznych potenciáloch. Sieť pre simulátor *Elmer* **musí** byť vygenerovaná vo formáte mesh 2.0.

Elektródy sú vytvorené ako dutiny, v ktorých sieť negenerujeme, pretože v uzatvorenom kovovom priestore má intenzita statického elektrické poľa nulovú hodnotu.



Obr. 4.1: Model v gmsh

```
import os
_ = os.system("gmsh ./sim/ex_01.msh")  # zobrazenie vygenerovanej siete,
    neobsahuje logicke entity
```

4.3 Postup simulácie

Proces simulácie je riadený textovým súborom s príponou **.sif*, v ktorom je definovaný typ simulácie, počiatočné podmienky, meteriálové parametre a požadované výstupy. V Elmeri je možné kombinovať viacej typov simulácie, napr. prúdenie kvapaliny v prostredí pri zmenách teploty a pod.

- Import mriežky a vvygenerovanie interných mriežok pre solver pomocou programu ElmerGrid
- Vytvorenie povelového súboru **.sif* - v textovom editore alebo pomocou programu ElmerGUI
- Spustenie simulácie ElmerSolver
- Spracovanie výsledkov simulácie

4.3.1 Import siete

Interné siete pre solver vygenerujeme pomocou programu ElmerGrid

```
ElmerGrid 14 2 ex_01.msh
```

Prorgarm vytvorí pomocý adresár s názvom *ex_01*, do ktorého uloží vygenerované súbory *mesh.**.

```
_ = os.system("ElmerGrid 14 2 ./sim/ex_01.msh")
```

4.3.2 Vytvorenie povelového súboru

Štruktúra **.sif* súboru je detailne popísaná v dokumentácii a na množstve príkladov, na začiatku je možné využiť program *ElmerGUI*, pomocou ktorého môžeme vygenerovať (”vyklikať”) základnú formu súboru.
Poznámka: Program ElmerGUI je už značne zastaralý, pochádza ešte z dôb, kedy mal byť súčasťou solveru aj pre- a post-processing, vzhľadom k dostupnosti špecializovaných programov (ParaView ...) tvorcovia už tento program nevyvíjajú a ich aktivity sa sústreďujú len na samotný solver.

4.3.3 Vytvorenie povelového súboru v ElmerGUI

Pre definovaním povelového súboru simulácie je potrebné v ElmerGUI importovať súbor s vygenerovanou sieťou.

```
# spustenie grafick0ho rozhrania
_ = os.system("ElmerGUI")
```

4.3.4 Definícia typu simulácie

V type simulácie určujeme typ súradnicovej sústavy geometrie usporiadania problému (rotačne-symetrická, karteziánska ...), fyzikálne konštanty, vstupné a výstupné súbory a pod.

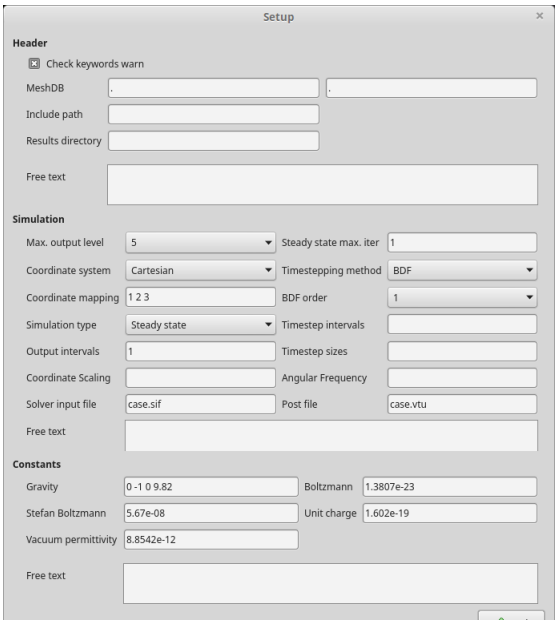
4.3.5 Výber typu problému, materiálových parametrov, počiatočných a okrajových podmienok

Vmenu *Model* postupne definujeme všetky parametre simulácie. Mená fyzických entít sú nahradené ich poradím, toto poradie je uvedené v záhlaví súboru s vygenerovanou sieťou **.msh* Ako okrajové podmienky definujeme potenciály elektród a typ okolia, materiálové podmienky definjú permitivitu prostredia.
Po definovaní všetkých podmienok v menu *Sif* vygenerujeme povelový súbor a v pod-menu *Edit* ho uložíme (pod menom napr. *case.sif*) do adresáru s vygenerovanými internými sieťami. Do tohoto adresáru uložíme aj štartovací súbor simulátor *ELMERSOLVER_STARTINFO*, ktorý obsahuje meno povelového súboru pre simulátor *ElmerSolver*, ktorý chceme spustiť (pre jednu kofiguráciu/mriežku môžeme mať vytvorených niekoľko simulácií, s rôznymi počiatočnými podmienkami a pod.)

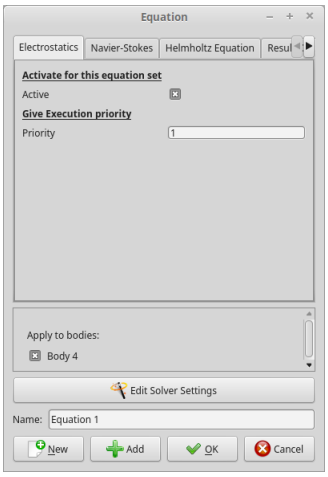
```
case.sif
```

alebo simuláciu spustíme z konzoly priamo aj s vygenerovaním štartovacieho súboru

```
echo case.sif > ELMERSOLVER_STARTINFO & ElmerSolver
```



Obr. 4.2: Základná konfigurácia



Obr. 4.3: Konfigurácia parametrov

4.3.6 Vytvorenie povelového súboru v textovom editore

Vytváranie povelového súboru z menu môže byť zdĺhavé a neflexibilné, ak potrebujeme meniť čiastkové parametre napr. okrajové podminky pri opakovaných simuláciách. Povelový súbor môžeme vytvoriť aj jednoducho v textovom editore alebo ho môžeme generovať jednoduchým skriptom.
Detailne popísané povelové súbory k príkladom z rôznych typov simulácií sú popísané v dokumentácii ElmerTutorials_nonGUI.
Povelový súbor je rozdelený na niekoľko sekcií:
Všeobecné parametre simulácie

```
Simulation
  Max Output Level = 5
  Coordinate System = Cartesian        ! vyber typu suradnicoveho systemu
  Coordinate Mapping(3) = 1 2 3        ! moznost vymeny suradnicovyxh osi
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
```

```
  Solver Input File = case.sif          ! ignorovane ...
  Post File = case.vtu
End
```

Konštanty

```
Constants                               ! staci uvadzat len tie konstanty
  Gravity(4) = 0 -1 0 9.82              ! ktore su relevantne pre dany typ simulacie
  Stefan Boltzmann = 5.67e-08
  Permittivity of Vacuum = 8.8542e-12
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End
```

Definícia problému
Pre komplexnejšie problémy môžeme mať definovaných niekoľko solverov. V procese simulácie sa v každom kroku na riešenie úlohy použijú všetky solvery, ich poradie a prioritu je možné voliť.

```
Solver 1
  Equation = Electrostatics
  Calculate Electric Energy = True
  Procedure = "StatElecSolve" "StatElecSolver"
  Variable = Potential
  Calculate Electric Flux = True
  Calculate Electric Field = True
  Exec Solver = Always
  ... kopa dalsich nastaveni
  ... pre linearny a nelinearny solver
End
```

Výber skupiny rovníc pre riešenie problému
V definícii problému môžeme mať definovaných niekoľko typov solverov, je možné z nich kombinovať rôzne typy rovníc, ktoré môžu byť použité na rôzne časti problému. Napr. pri simulácii prúdenia kvapaliny potrubím, ktorého rozmery sa menia s teplotou, nebudeme pre simuláciu zmeny potrubia od teploty používať solver pre riešenie prúdenia kvapaliny.

```
Equation 1
  Name = "Equation 1"
  Active Solvers(1) = 1
End
```

Definícia materiálov a ich vlastností

```
Material 1
  Name = "Material 1"
  Relative Permittivity = 1
End
```

Výber rovníc pre jednotlivé fyzické entity
Teleso 1 je z materiálu 1 a pre riešenie sa použije skupina rovníc 1.

```
Body 1
  Target Bodies(1) = 4
  Name = "Body 1"
  Equation = 1
  Material = 1
End
```

Okrajové podmienky
Definícia okrajových podmienok pre hranice problému. Položky závisia od typu problému, jednotky sú v SI.

```
Boundary Condition 1
  Target Boundaries(1) = 1
  Name = "BoundaryCondition 1"
  Potential = 10                    ! potencial elektrody +10 [V]
End

Boundary Condition 2
  Target Boundaries(1) = 2
  Name = "BoundaryCondition 2"
  Potential = -10
End

Boundary Condition 3
  Target Boundaries(1) = 3
  Name = "BoundaryCondition 3"
  Potential = 0
End
```

Kapitola 5

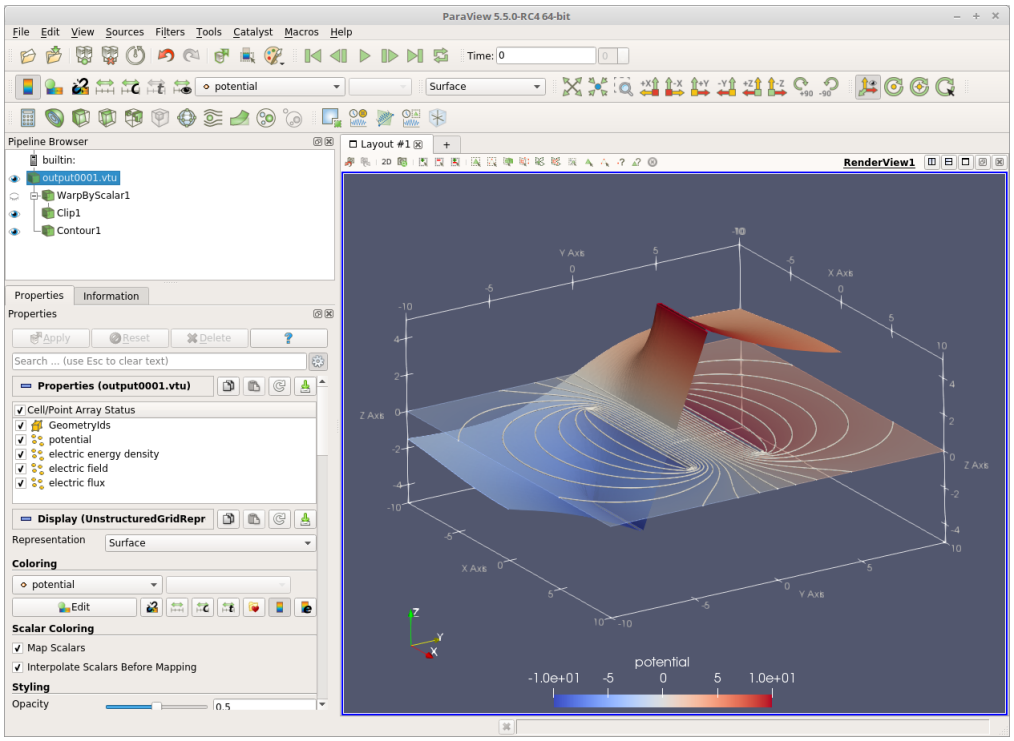
Post-Processing

Výsledok simulácie vo formáte vtu (ParaView VTK Unstructured Data) možné ďalej spracovávať a vizualizovať. Pre formát VTU existuje množstvo programov, zoznam je uvedený na https://www.vtk.org/Wiki/VTK_Tools

5.1 Vizualizácia VTU dát pomocou ParaView

ParaView je komplexný vizualizačný nástroj určený pre spracovanie rozsiahlych dát, skriptovateľný v Pythone s možnosťou využitia paralelnej architektúry CPU.

```
import os
_ = os.system("paraview ./sim/ex_01/output0001.vtu")
```



Obr. 5.1: Vizualizacia v Paraview

5.1.1 Import dát a ich úpravy

V ParaView je spracovanie dát organizované vo forme grafu, kde na výsledok predchádzajúcej aktivity uplatňujeme rôzne filtre. Vstupné dáta (program rozpoznáva veľké množstvo formátov) je možné spracovať rôznymi spôsobmi a výsledky kombinovať vo finálnom zobrazení. Filtrov je veľa typov, najčastejšie používané sú

- WarpByScalar - 3D reprezentácia skalárnych dát
- Clip - orezanie dát
- Slice - rez dátami
- Contour - vrstevnice
- Delaunay - vyhladzovanie 2D/3D dát
- ...

Je možné vytvárať si aj vlastné špecifické filtre. Vysledok každého kroku spracovania je možné samostatne zobraziť. Je možné zo série simulácií vytvárať animácie, program poskytuje množstvo ďaľších možností. Program je skriptovateľný v Pythone.

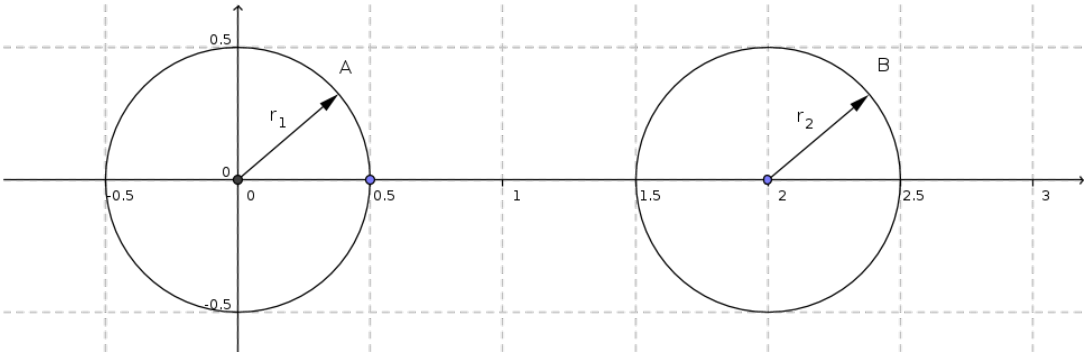
Kapitola 6

Príklad: Kapacita dvoch vodivých gulí

6.1 Zadanie

Úlohou je vypočítať vlastnú a vzájomnú kapacitu dvoch vodivých gulí analytickým postupom a porovnať výsledky s FEM simuláciou.

Predpokladajme, že máme dve vodivé gule A, B s rovnakým priemerom $r_1 = r_2$, stredy ktorých sú od seba vzdialené d, jedna z gulí sa nachádza v počiatku súradnicovej sústavy. Vzťah pre kapacitu odvodíme metódou zrkadlenia náboja.



Obr. 6.1: Usporiadanie gulí

6.2 Teoretické odvodenie

Pre kapacitu ľubovolného vodivého objektu platí všeobecný vzťah

$$C = \frac{q}{\phi}$$

ktorý deklaruje kapacitu ako schopnosť telesa zhromažďovať náboj. Ak sa na teleso dostane nejaký náboj q, tak jeho potenciál bude mať hodnotu ϕ voči referenčnému nulovému potenciálu v nekonečne, ale na nulovom potenciáli môže byť aj nejaké iné teleso. Potenciál je ekvivalentom práce, ktorú musíme vynaložiť na to, aby sme na teleso prepravili jednotkový náboj z bodu s nulovým potenciálom.

V našom príklade budeme preto hľadať *vzájomnú kapacitu* gule A voči uzemnenej guli B a jej *vlastnú kapacitu* voči nulovému potenciálu v nekonečne.

6.2.1 Potenciál na povrchu nabitej gule

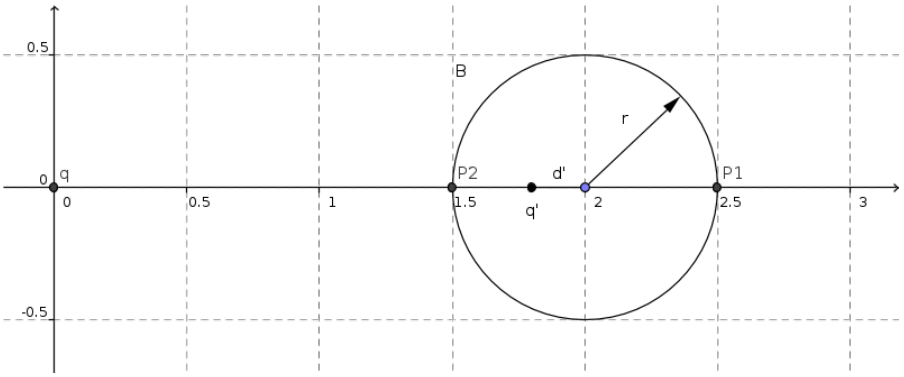
Potenciál na povrchu nabitej gule je rovnaký, ako potenciál bodového náboja umiestneného v jej strede vo vzdialenosti rovnej polomeru gule. Ak preto do začiatku súradnicovej sústavy umiesnime bodový náboj q, vo vzdialenosti r_1 bude potenciál daný vzťahom

$$\phi(r) = \frac{1}{4\pi\epsilon_0}\frac{q}{r_1}$$

6.2.2 Zrkadlenie náboja

Do priestoru bez gulí do miesta kde by sa nachádzal stred gule A, t.j. v počiatku súradnicovej sústavy umiestnime náboj q. Tento okolo seba vytvorí elektrostatické pole, ktorého intenzita vo vzdialenosti r_1 bude rovnaká ako na povrchu vodivej guli s rovnakým polomerom nabitej rovnakým nábojom.
Niekde v blízkosti miesta, kde by sa nachádzal stred gule B umiestnime teraz náboj q' tak, aby tento vytvoril okolo seba guloplochu s nulovým potenciálom priemerom pôvodnej gule B, ktorá by bola ekvivalentom uzemnenej gule B. Intuitívne je zrejmé, že náboj q' bude musieť mať opačnú polaritu ako náboj q. Hladanými veličinami sú veľkosť náboja q' a jeho poloha.

6.2.3 Guloplocha s nulovým potenciálom



Obr. 6.2: Guloplocha s nulovým potenciálom

Pre potenciál množiny nábojov platí princíp superpozície, takže pre hodnoty potenciálu v bodoch P1 a P2, ktoré sa nachádzajú na hladanej guloploche s nulovým potenciálom, platí

$$P1: \quad \frac{q}{d+r} + \frac{q'}{r+d'} = 0 \tag{6.1}$$

$$P2: \quad \frac{q}{d-r} + \frac{q'}{r-d'} = 0 \tag{6.2}$$

$$\tag{6.3}$$

Triviálnou úpravou rovníc dostaneme sústavu

$$2qr + 2q'd = 0$$
$$2qd' + 2q'r = 0$$

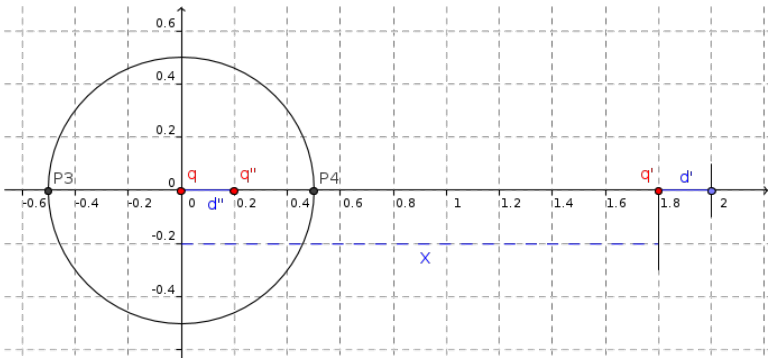
ktorej riešením je

$$q' = -\frac{r}{d}q \qquad d' = -q'\frac{r}{q} = \frac{r^2}{d} \tag{6.4}$$

Je zrejmé, že pole záporného náboja q' zrejme zdeformovalo ekvipotenciálne hladiny v okolí kladného náboja q a tieto už asi nebudú mať gulová tvar. Túto deformáciu môžeme kompenzovať novým (kladným) nábojom q'', umiestneným niekde v blízkosti stredu gule A, počiatku súradnicovej sústavy. Dôležité je to, že náboj q'' slúži pre korekciu tvaru plochy, **potenciál v mieste korigovanej guloplochy zostáva rovnaký** ako na začiatku, pred umiestnením náboja q'.

Ak teraz odstránime pôvodný náboj q, dostaneme pododobnú situáciu ako v predchádzajúcom prípade a budeme požadovať, aby sa v mieste gule A nachádzala guloplocha s nulovým potenciálom.

6.2.4 Korekcia náboja



Obr. 6.3: Korekcia náboja

Pre potenciál v bodoch P3 a P4 potom obdobne platí

$$P3: \quad \frac{q''}{r+d''} + \frac{q'}{x+r} = 0$$
$$P4: \quad \frac{q''}{r-d''} + \frac{q'}{x-r} = 0$$

kde

$$x = d - d' = d - \frac{r^2}{d} = \frac{1}{d}(d^2 - r^2)$$

Riešením sústavy dostaneme

$$q'' = -\frac{r}{x}q' = \frac{r^2}{d^2 - r^2}q \qquad d'' = -\frac{q''r}{q'} = \frac{r^2}{d - \frac{r^2}{d}} \tag{6.5}$$

Doplnením náboja q'' do sústavu sa ale zase zmenilo pole v okolí náboja q', čo môžeme zase kompenzovať ďalším nábojom q''' atď.
Polohy a veľkosti nábojov dostaneme opakovaním predchádajúceho postupu, je zrejmé, že hodnoty nábojov a ich vzdialeností tvoria nekonečnú postupnosť. Niekoľko členov postupnosti pre veľkosti a polohy nábojov pre pravú a lavú gulu je uvedených v nasledujúcich tabuľkách:

Lavá gula (A)

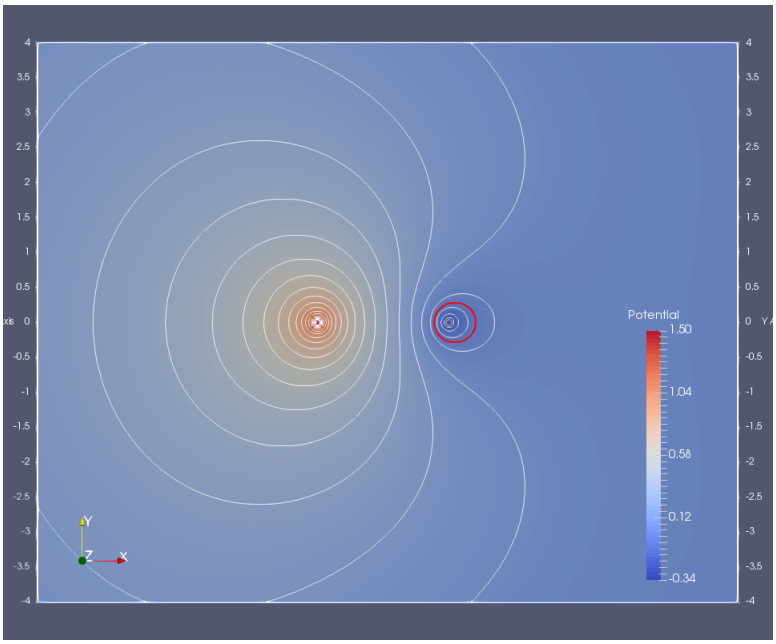
Náboj	Vzdialenosť
q	0
$q'' = \frac{r^2}{d^2 - r^2}\, q$	$d'' = \frac{r^2}{d - \frac{r^2}{d}}$
$q'''' = \frac{r^4}{d^4 - 3d^2r^2 + r^4}\, q$	$d'''' = \frac{r^2}{d - \frac{r^2}{d - \frac{r^2}{d}}}$

Pravá gula (B)

Náboj	Vzdialenosť
$q' = -\frac{r}{d}\,q$	$d' = \frac{r^2}{d}$
$q''' = -\frac{r^2}{d^2 - r^2}\,\frac{r}{d}\,q$	$d''' = \frac{r^2}{d - \frac{r^2}{d - \frac{r^2}{d}}}$
$q''''' = -\frac{r^4}{d^4 - 4d^2r^2 + 3r^4}\,\frac{r}{d}q$	$d''''' = \frac{r^2}{d - \frac{r^2}{d - \frac{r^2}{d - \frac{r^2}{d}}}}$

6.2.5 Simulácia ekvipotenciálnych hladín v 2D

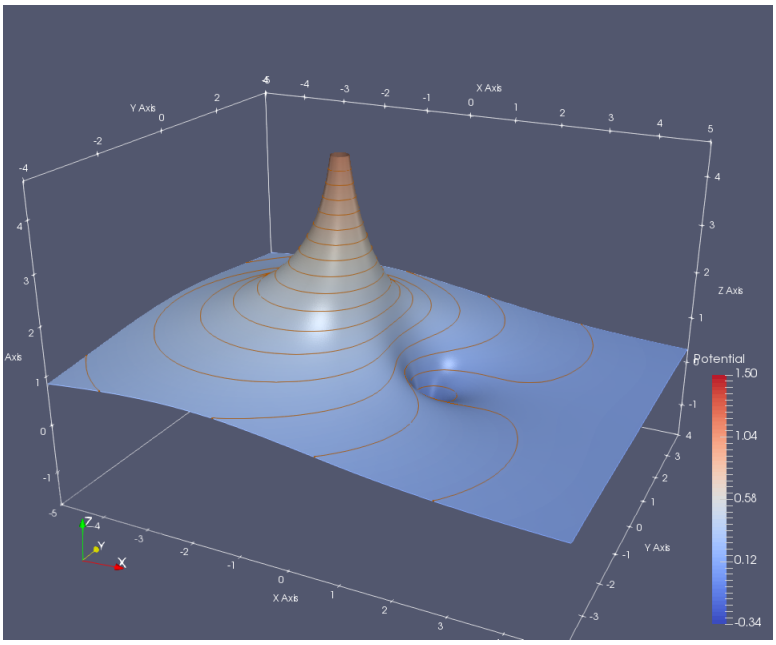
Zjednodušená simulácia potenciálu v okolí bodových nábojov q a q' bola použitá najmä na odhad potrebného rozmeru simulovaného priestoru s ohľadom na okrajové pomienky. Červeno je vyznačená nulová potenciálová hladina.



Obr. 6.4: Ekvipotenciálne hladiny v 2D

6.3 Numerický výpočet potenciálu

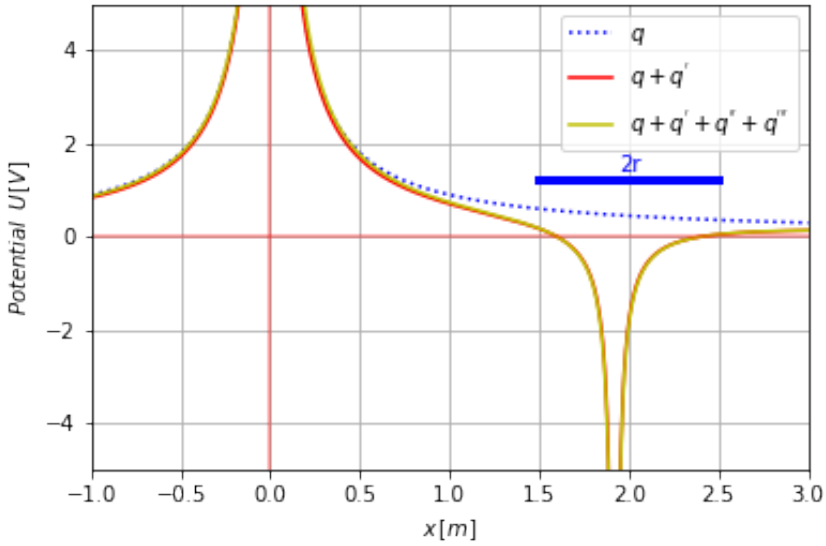
```
from scipy import *
import pylab as plt

q=1e-10    # [C] charge
r=0.4      # [m] sphere diameter
d=2.0      # [m] distance between center of spheres
```



Obr. 6.5: Ekvipotenciálne hladiny v 3D

```
k = r/d; w=r**2/d
q0 = q                      # q
d0 = 0.0

q1 = -k*q                   # q'
d1 = d-w

q2 = w/(d-w)*q              # q''
d2 = r**2/(d-w)

q3 = -w/(d-2*w)*k*q         # q'''
d3 = d-r**2/(d-r**2/(d-w))

def phi(q, d, x):
    q=q/(4*pi*8.854e-12)
    if d==x:
        if x<d1+d2/2.:
            return 1e6
        else:
            return -1e6
    if d>x:
        return q/(d-x)
    else:
        return -q/(d-x)
x=[]; p0=[]; p1=[]; p2=[]; p3=[]
step=0.001
for k in arange(d0-1, d3+1.5, step):
    x.append(k)
    s1=phi(q0, d0, k)
    s2=phi(q1, d1, k)
    s3=phi(q2, d2, k)
    s4=phi(q3, d3, k)
    s=s1;   p0.append(s)
    s=s+s2; p1.append(s)
    s=s+s3; p2.append(s)
    s=s+s4; p3.append(s)

plt.plot(x,p0,'b:', label="$q$")
```

```
plt.plot(x,p1,'r',  label=r"$q+q^{'}$")
plt.plot(x,p3,'y', label=r"$q+q^{'}+q^{''}+q^{'''}$")
plt.plot([-1,3], [0,0], 'r', alpha=0.4)
plt.plot([0,0], [-5,5], 'r', alpha=0.4)
plt.plot([1.5,2.5], [1.2,1.2], 'b', linewidth=4)
plt.text(1.95, 1.4,  '2r', color='blue')
plt.xlabel('$x\, [m]$')
plt.ylabel('$Potential \,\,\, U\,[V]$')
plt.ylim(-5.0, 5.0)
plt.xlim(-1, 3)
plt.legend()
plt.grid()
plt.show()
```



6.4 Výpočet kapacity

Výsledný celkový náboj lavej gule A je

$$q_{sum} = q + q'' + q'''' \cdots = q\left(1 + \frac{r^2}{d^2 - r^2} + \frac{r^4}{d^4 - 3d^2r^2 + r^4} + \dots\right)$$

a jej celková kapacita je potom

$$C_{sum} = \frac{q}{\phi} = \frac{q}{\frac{q}{4\pi\epsilon_0 r}} = 4\pi\epsilon_0\, r\left(1 + \frac{r^2}{d^2 - r^2} + \frac{r^4}{d^4 - 3d^2r^2 + r^4} + \dots\right)$$

Celková kapacita pozostáva z dvoch častí, z kapacity voči nulovému potenciálu v nekonečne a kapacite voči druhej guli. Vzájomná kapacita je úmerná indukovanému náboju gule B na guli A, pretože veľkosť náboja gule B poznáme, môžeme náboj na guli A rozdeliť

$$\begin{aligned} C_{sum} = C_{gnd} + C_{mut} &= \frac{q - q' + q'' - q''' + q'''' + \cdots + q' + q''' - \dots}{\phi} = \\ &= 4\pi\epsilon_0\, r\left(1 - \frac{r}{d} + \frac{r^2}{d^2 - r^2} - \frac{r^2}{d^2 - r^2}\frac{r}{d} + \frac{r^4}{d^4 - 3d^2r^2 + r^4} - \frac{r^4}{d^4 - 4d^2r^2 + 3r^4}\frac{r}{d} + \dots\right) + \\ &+ 4\pi\epsilon_0\, r\left(\frac{r}{d} + \frac{r^2}{d^2 - r^2}\frac{r}{d} + \frac{r^4}{d^4 - 4d^2r^2 + 3r^4}\frac{r}{d} + \dots\right) \end{aligned}$$

Na základe vyššie uvedených vzťahov môžeme spočítať hodnotu kapacity (pre prvé tri členy radov) s hodnotami

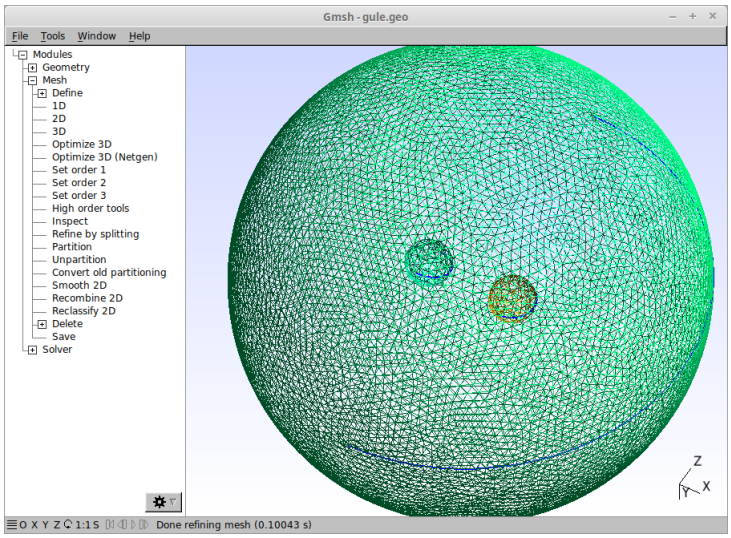
- $r = 0.5\,[m]$ - polomer gulí
- $d = 2.0\,[m]$ - vzdialenosť stredov gulí

```
eps = 8.8542e-12 # F/m

def C12(a,d):
    # vzajomna kapacita
    c = 4*pi*eps*(a**2)/d*(1 + a**2/(d**2 - 2*(a**2)) + \
        (a**4)/(d**4 - 4*(d**2)*(a**2) + 3*(a**4)) )
    return c
    def C10(a,d):
    # vlastna kapacita
    c = 4*pi*eps*a*(1 - a/d + a**2/(d**2 - (a**2)) + \
    (a**3)/(d**3 - 2*d*(a**2)))
    return c

print('C12 =', C12(.5, 2.) * 1e12, '[pF]')
print('C10 =', C10(.5, 2.) * 1e12, '[pF]')
```

```
C12 = 14.972907573290463 [pF]
C10 = 46.42671205015176 [pF]
```

6.5 Simulácia

Pri simulácii príkladu pomocou FEM postupujeme štandardne, vytvoríme model (FreeCAD, gmsh ...) a vygenerujeme mriežku.



Obr. 6.6: Simulačné prostredie

V povelovom súbore simulácie definujeme výpočet kapacitnej matice a v okrajových podmienkach identicikáciu telies, ktorých kapacitu budeme počitať. Pretože na výpočet kapacity je použitý iný algoritmus ako na výpočet polí, tokov a pod. je vhodné mať pre každý typ simulácie vlastný povelový súbor.

```
Solver 1
  Equation = Electrostatics
  Calculate Capacitance Matrix = True              ! vypocet kapacitnej matice
  Calculate Electric Flux = True
  Variable = Potential
```

```
  Capacitance Matrix Filename = cap_matrix.dat      ! ulozenie vysledkov
  Procedure = "StatElecSolve" "StatElecSolver"
  Calculate Electric Field = True
  Calculate Electric Energy = True
  ...
End

Boundary Condition 1
  Target Boundaries(1) = 4
  Name = "1v"
  Capacitance Body = 2                               ! doplnenie ID telies pre vypocet kapacity
  Potential = 1
End

Boundary Condition 2
  Target Boundaries(1) = 3
  Name = "-1v"
  Capacitance Body = 3
  Potential = -1
End
```

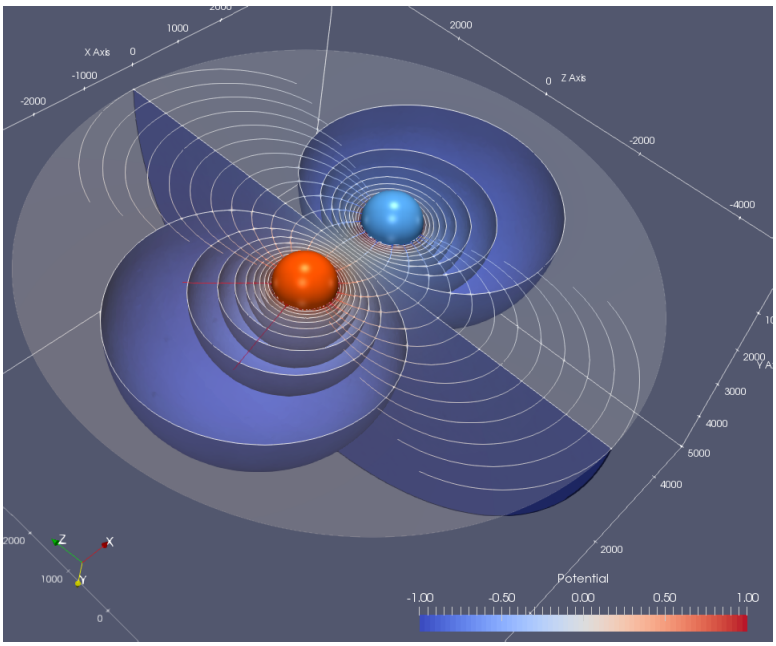
Pre výpočet bola použitá kapacitná matica z solveru pre elektrostatické úlohy, simulačná oblasť bola gula s priemerom 5m. Výpočítané hodnôt z 3D simulácie sú

$$C_{gnd} = 44.70\ pF \qquad C_{mut} = 14.90\ pF$$

Pre porovnanie - analytické hodnoty

$$C_{gnd} = 46.42\ pF \qquad C_{mut} = 14.97\ pF$$

Rozdiel v hodnotách je zrejme spôsobený v nedostatočnej hustote siete a okrajovými efektami pri malej simulačnej oblasti.mSimulačný model môžeme samozrejme využiť aj na štandardnú simuláciu a zobrazenie vlastností elektrostatického poľa.



Obr. 6.7: Vizualizácia výsledkov

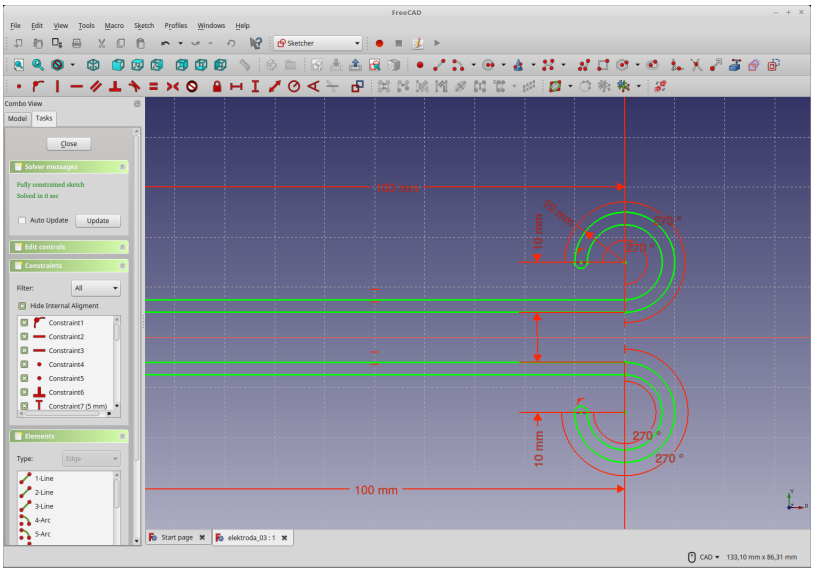
Kapitola 7

Príklad: VN kondenzátor

Pri konštrukcii vysoko-napäťových zariadení je potrebné zabrániť vzniku korónyna miestach, kde je vysoká intenzita elektrického poľa. Jedno z možných riešení ukazuje nasledujúci príklad - ostrú hranu ”zabalíme”do miesta s nízkou intenzitou elektrického poľa.

7.1 Vytvorenie náčrtu v FreeCad Sketcher

Sketcher zjednodušuje kreslenie, princíp je založený na vzájomnom kotvení komponentov pomocou rôznych druhov obmedzení tak, aby vo výsledku bol vytvorený náčrt ”pevný”.



Obr. 7.1: Náčrt

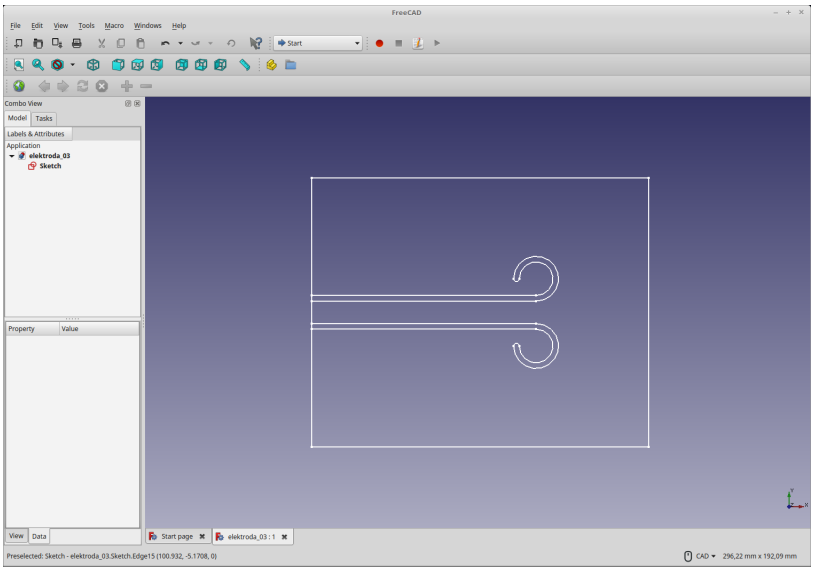
7.2 FreeCAD PartDesigner

Náčrt môžeme ďalej upravovať, v našom prípade ho vyexportujeme vo formáte **.brep* pre ďalšie spracovanie v *gmsh*. Simulácia kondenzátora bude v 2D a rotačnej symetrii.

7.3 Vytvorenie siete

Súbor **.brep* obsahuje len logické entity, pre korektne generovanie siete musíme definovať fyzické entity. Skript pre **gmsh** **.geo* má potom tvar

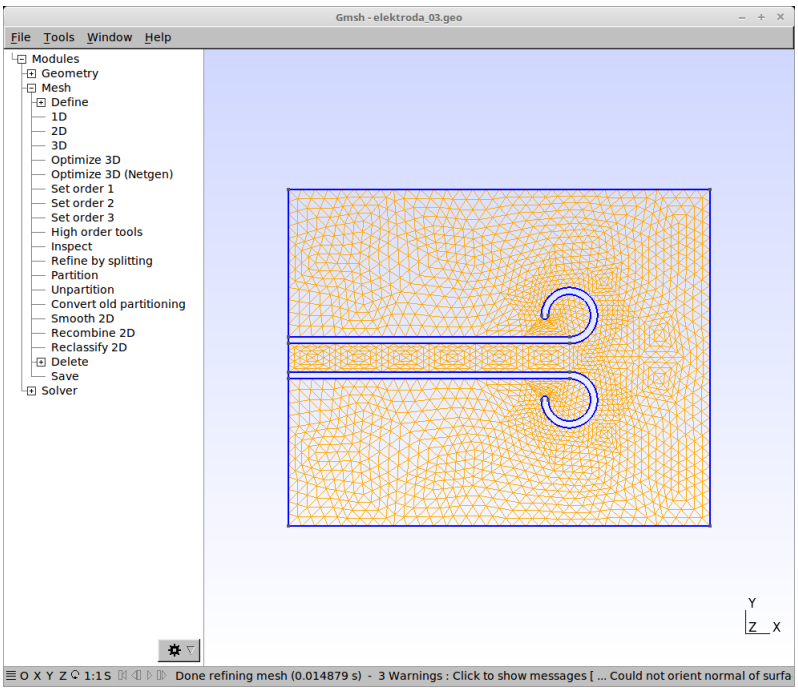
```
Merge "elektroda_03.brep";
```



Obr. 7.2: Export náčrtu

```
Physical Line("e1") = {2,3,4,5,6};          ! cisla segmentov ciar v subore brep
Physical Line("e2") = {12,13,14,15,16};
Physical Line("ex") = {7,8,9,10,11,18};

Line Loop(1) = {2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,18};
Plane Surface(2) = {1};
Physical Surface("air") = {2};
```

Vytvorená sieť je potom len na fyzickej ploche ohranicenej elektródami.



Obr. 7.3: Vytvorenie siete

7.4 Simulácia v Elmer

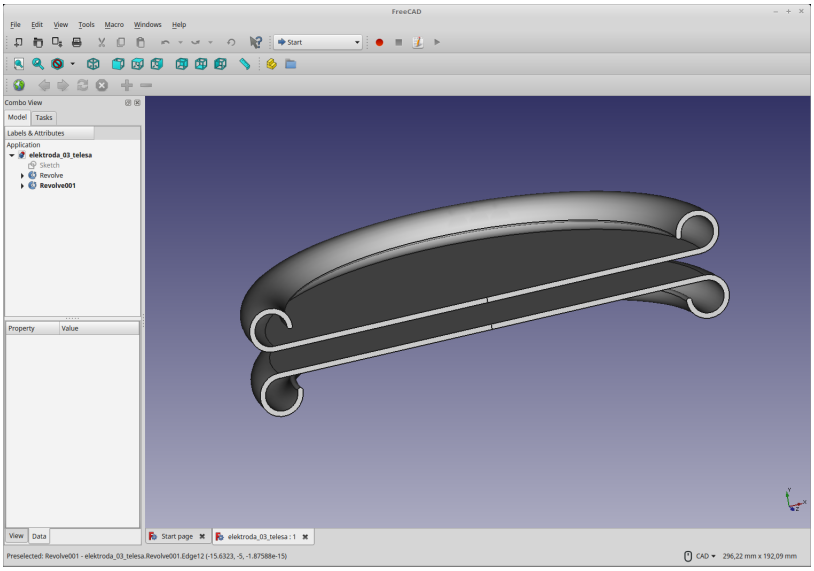
V povelovom súbre simulátora **.sif* musíme deklarovať, že riešenie problému bude v rotačnej symetrii

```
Simulation
  Coordinate System = Axi Symmetric
  ...
End
```

Ostatné časti súboru sú rovnaké ako v prípade rovinného kondenzátora.

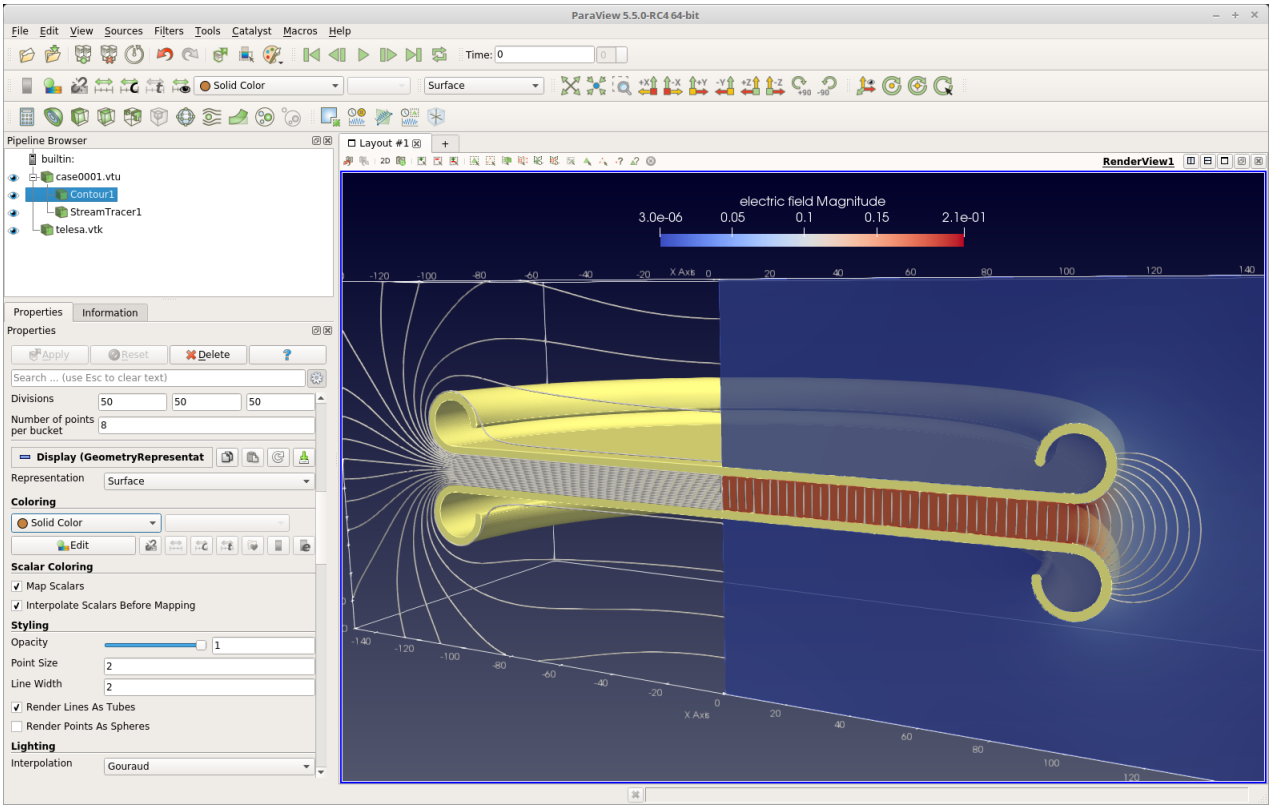
7.5 Post-procesing

Ak chceme znázorniť polia spolu s fyzickým objektom, vygenerujeme si vo FreeCAD-e 3D objekt, exportujeme ho do *gmsh*, kde si vytvoríme sieť na povrchu objektu a exportujeme ho do formátu **.vtu*.



Obr. 7.4: 3D model

V programe *ParaView* importujeme výsledok simulácie ako aj 3D vygenerovaný objekt. Na znázornenie potenciálových hladín ako aj elektrických siločiar nám stačia len základné filtre programu. Zo simulácie je zrejmé, že na krajoch elektród nie je zvýšená intenzita elektrického poľa a ostrá hrana sa nachádza v mieste s nizkou intenzitou.



Obr. 7.5: Vizualizácia simulácie