FEM Modeling of PMSMs Using Elmer

Research · May 2015

DOI: 10.13140/RG.2.1.1721.4886

Pavel Ponomarev
ABB

Open your mind. LUT.
Lappeenranta University of Technology

FEM Modeling of PMSMs Using *Elmer*

Example workflow.

Elmer 7 (rev. 6790).

Version 1.1

18.07.2014

Authors:

Pavel Ponomarev

Contents

Introduction

Open source software for engineering computations has several benefits and drawbacks of usage. Among the main benefits are – free licenses (free usage and modification), good extensibility and customizability, deep involvement of the academy in the process. The drawbacks are usually – low support, low quality of documentation, non-optimized user interfaces (usually lack of GUI). These drawbacks usually imply versatile IT skills from the end users. However, continuous increase of the community of open-source projects users should bring more contributors to the project, which, in turn, should improve quality of the product and end-user experience.

The main aim of this report is to demonstrate an example basic workflow for electrical machine electromagnetic design and optimization process with open-source FEM software *Elmer*. This should help the end-users (electrical machine designers) in adaptation of open source software in their workflow.

Elmer is open-source software package for multi-physics FEM modeling (see http://www.csc.fi/english/pages/elmer/). *Elmer* has modular structure. Each module (or 'solver' in terms of *Elmer* documentation) can make computation and data manipulation on a given mesh. *Elmer* includes modules for solving various physical problems. For electrical machine designers especially important modules are modules for electro-magnetic, thermal, fluid-flow, and structural mechanical analysis problems. With *Elmer* modular structure it is possible to perform conjugated simulations of fluid flow and thermal phenomenon to determine the cooling requirements for simulated equipment. On the same mesh an electro-magnetic and mechanical analysis can be performed as well. *Elmer* supports parallel computations with MPI, which allows solving of a large 3D problem by mesh partitioning and parallelization. *Elmer* can be compiled for Linux, Windows and MacOS, and can be run on a supercomputer for parallel computations.

Standard FEM simulation procedure for electro-magnetic phenomenon consists of several stages

- Problem definition
- Geometry definition
- Meshing
- Pre-processing (boundary conditions, faces properties)
- Solving
- Post-processing

In this report an example workflow is demonstrated for each stage using an open-source/free software. Several alternative open software packages are also listed for the main stages of a FEM simulation. It is advisable to read Elmer documentation before continuing this tutorial [4].

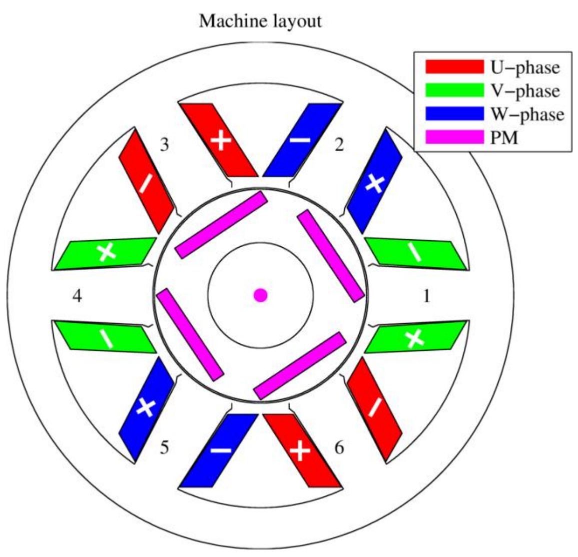
This work is partially financially supported by the FiDiPro project "LUT GSEEE FiDiPro Fellowship in Permanent magnet machines", and by TES scholarship for "Investigation of Self-Sensing Capabilities of Different Topologies of Tooth-Coil Permanent Magnet Synchronous Machines for Sensorless Drives".

1. Problem Definition

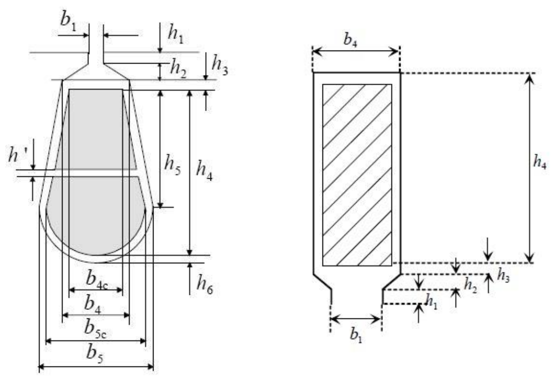
Below is initial data for a 3 kW 3000 rpm 6/4 TC-IPMSM obtained analytically using procedures described in [1,2]. It is a permanent magnet synchronous motor for fan applications. It utilizes IPM rotor topology with embedded NdFeB magnets and tooth-coil winding construction, which results in high-torque density of this machine.

```
Main data:
---------------------------------------------
 Number of slots              | Qs               |    6
 Number of poles              | p*2              |    4
 Number of slots/pole/phase   | q                |  0.5 = 1 / 2
 Input power                  | P_in             | 3.2e+03 W
 Shaft power                  | P                | 3e+03 W
 Efficiency                   | eff              | 0.938
 Rated torque                 | Tr               | 9.55 Nm
 Rated speed                  | n                | 3e+03 rpm
 cos(phi)                     | pf               | 0.884
 Line-to-line voltage         | U                |  230 V
 Stator phase voltage         | Uph              |  133 V
 PM induced voltage           | E_PM             |  117 V
 Frequency                    | f                |  100 Hz
 Stator current               | Is               | 9.08 A
 Stator d-current             | Id               | -0.896 A
 Stator q-current             | Iq               | 9.04 A
 Phase resistance             | Rph              | 0.345 Ohm
 Magnetizing inductance       | Lmd               | 3.48 mH
 Airgap leakage ind.          | L_gap            |  1.6 mH
 Slot leakage ind.            | L_u              | 2.58 mH
 Tooth tip leakage ind.       | L_tt             | -0.407 mH
 End winding leakage ind.     | L_ew             | 0.754 mH
 Stator leakage inductance    | L_ssigma         | 4.53 mH
 Synchronous inductance       | Ld               | 8.01 mH
 Synchronous inductance       | Lq               | 11.5 mH
 Number of phase turns        | N                |  174
 Wire Current density         | J_s              | 3.46e+06 A/m^2
 Windings` wire diameter      | D_Cu             | 0.00183 m
 Number of slot conductors    | z_Qs             |  174
 Wound areae of the slot/kCu  | S_Cus            | 0.000725 m^2
 Total areae of the slot      | Sslot            | 0.00155 m^2
 Carter factor                | kC               | 1.03
 Winding factor               | kw_1             | 0.866
 Target Tangential stress     | sigma_Ftan       | 2e+04 Pa
 Obtained Tangential stress   | sigma_Ftan_real  | 1.9e+04 Pa
 Linear current density       | Arms             | 3.95e+04 kA/m
 Airgap flux density peak     | B1_peak          | 0.85 T
 Max. of rect. flux density   | Bmax             | 0.725 T
 Stator tooth flux density    | Bdapp            | 1.55 T
 Stator yoke flux density     | Bys              |  1.2 T
 Rotor yoke flux density      | Byr              |  1.4 T
 Stator resistive losses      | P_Cu             | 85.7 W
 Iron losses                  | P_Fe             | 64.8 W
 Additional losses            | P_ex             |   15 W
 Mechanical losses            | P_mech           | 11.2 W
 PM losses                    | P_PMEC           |   20 W
 Total losses                 | P_loss           |  197 W
 Load_angle                   | delta_load       | 29.6 degrees
 Overload                     | overload         | 3.04
```

```
Dimensions:
---------------------------------------------
 Length                       | l              |   50 mm
 Equivalent length            | l_e            | 50.3 mm
 Airgap                       | gap            |  0.6 mm
 Thickness of supporting band | h_sb           |    0 mm
 Physical airgap              | gap_ph         |  0.6 mm
 Outer stator diameter        | Ds_outer       | 204.9 mm
 Stator diameter              | D_s            | 81.2 mm
 Rotor diameter               | D_r            |   80 mm
 Inner Rotor Diameter         | Dr_inner       | 39.06 mm
 Height of stator yoke        | h_ys           | 22.59 mm
 Height of rotor yoke         | h_yr           | 14.63 mm
 Height of PM                 | h_PM           | 5.239 mm
 Magnet width                 | w_PM           | 38.96 mm
 Relative rect. gap flux dens | alpha_gap      | 0.75
 Pole pitch                   | tau_p          | 63.77 mm
 Width of stator tooth        | b_d            | 20.85 mm
 Slot pitch                   | tau_u          | 42.52 mm
 PM mass                      | m_PM           | 0.306 kg
 Cu mass                      | m_Cu           | 3.045 kg
 Rotor mass                   | m_rot          | 1.254 kg
 Total mass                   | m_tot          | 11.05 kg

---------------------------------------------
```



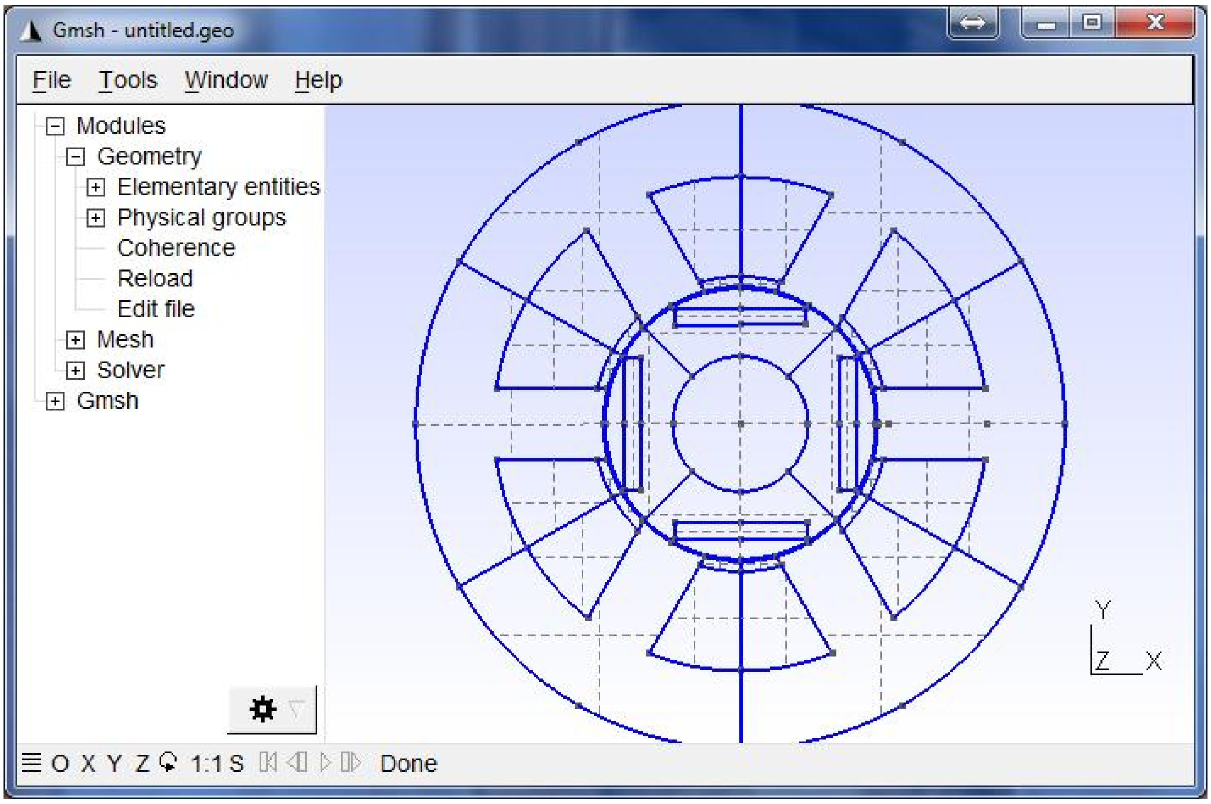
```
Slot Dimensions:
-----------------------------------------
 b1                           | b1             | 21.7 mm
 b4                           | b4             | 25.3 mm
 b4c                          | b4c            | 23.1 mm
 b5                           | b5             | 70.5 mm
 b5c                          | b5c            | 69.1 mm
 h1                           | h1             |    2 mm
 h2                           | h2             |  1.5 mm
 h3                           | h3             |  0.7 mm
 h5                           | h5             | 34.4 mm
 h6                           | h6             |  0.7 mm
 bc                           | bc             |  1.5 mm
 Height of stator slots       | h4             | 34.4 mm
 Total height of stator slots | hss_tot        | 39.3 mm

---------------------------------------------
```



The purpose of FEM simulation is validation of given analytical design. After the validation usually some optimization modifications in the geometry and additional FEM simulations are performed before machine is being manufactured.

2. Geometry Definition

Based on the analytically derived dimensions, the geometry of the simulated machine is built. There are several open-source/free software packages to define the geometry for the model – e.g. Salome, FreeCAD, GMSH. In this report the geometry is defined using GMSH.



The geometry for the example case is not optimized and modeled with simplifications. Instead of tooth tips, the semi-magnetic wedges are utilized. There are no flux barriers at the sides of the permanent magnets. The rotor surface is circular and smooth. These simplifications result in worse machine performance. However, these simplifications agree well with the main purposes of this report – demonstration of a basic example workflow for electrical machine design.

It is possible to utilize periodicities and symmetries in the machine construction to simplify the mesh and decrease the number of elements (and DOFs) in the model. This measure will significantly decrease the numerical complexity of the model and computational time will also decrease. Refer to the Elmer Manuals for instructions on how to define periodic boundary conditions. An example case with periodicities in a rotating electrical machine can be found in Elmer source code *Elmer/fem/tests/mgdyn2D_em*. For simplicity, in this paper the machine is modeled without utilization of periodicities.

GMSH

Gmsh is a 3D finite element grid generator with a build-in CAD engine and post-processor. It can be run on Windows, Linux and MacOS. Its design goal is to provide a fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities. Gmsh is built around four modules: geometry, mesh, solver and post-processing [http://geuz.org/gmsh/]. In this section only geometry module is described.

The main features of GMSH include [http://geuz.org/gmsh/doc/texinfo/gmsh.html]:

- quickly describe simple and/or “repetitive” geometries, thanks to user-defined functions, loops, conditionals and includes;
- parameterize these geometries. Gmsh's scripting language enables all commands and command arguments to depend on previous calculations;
- generate 1D, 2D and 3D simplicial (i.e., using line segments, triangles and tetrahedra) finite element meshes for CAD models in their native format (without translations) when linked with the appropriate CAD kernel;
- specify target element sizes accurately. Gmsh provides several mechanisms to control the size of the elements in the final mesh: through interpolation from sizes specified at geometry points or using flexible mesh size fields;
- create simple extruded geometries and meshes;

Several weaknesses can also be mentioned:

- user interface is only exposing a limited number of the available features (time is required to get acquainted with all script commands);
- Gmsh's internal CAD engine is fairly limited: it only handles simple primitives and does not perform any complex geometrical operations;
- Gmsh produces only conforming meshes;
- there is no global “undo” capability. You will often need to edit a text file to correct mistakes.

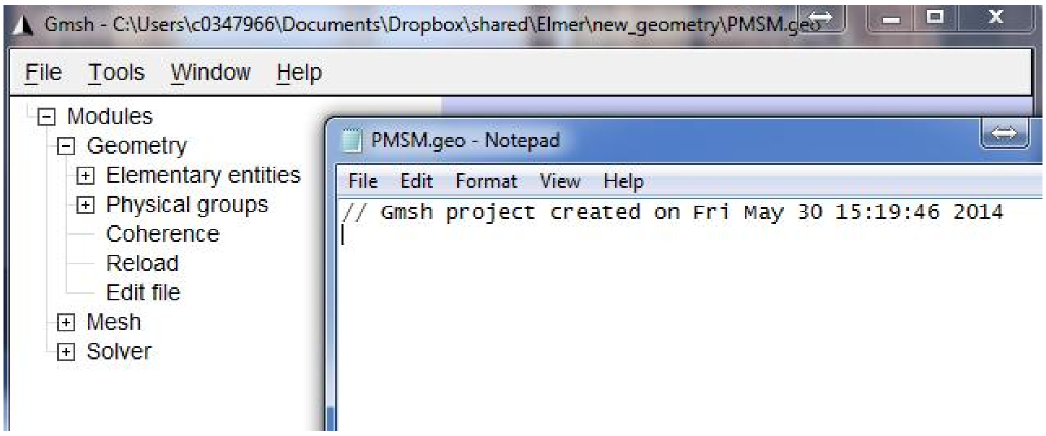
The geometry can be defined in separate files for rotor and for stator which produces two meshes for stationary and for rotating domains. It is however not yet possible to load two mesh files to ElmerGUI with current Elmer revision (6790). Elmer allows definition of stationary and rotating domains within 1 conforming mesh file using special boundary condition (`Mortar BC`) which enables motion of one part of the mesh.

First, the geometry variables will be defined; then, stator geometry, then rotor geometry, and then faces for the physical model preparation will be also defined in this section. The listing of the complete GMSH script file is given in the Appendix A.

Stator Geometry

Open GMSH and in the menu File select New and save your geometry file with the name *PMSM.geo*.

Then in the GMSH project tree select Modules > Geometry > Edit file. A text editor will pop up with opened file *PMSM.geo*.



```
// Gmsh project created on Fri May 30 15:19:46 2014
```

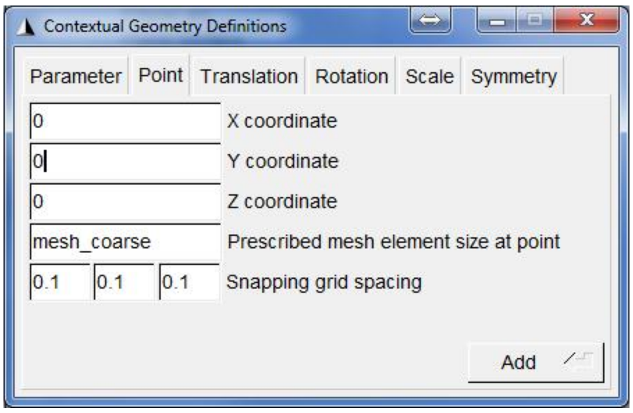
In GMSH the geometry is defined using its own scripting language. The variables and different control structures can be used for defining the geometry and mesh. So, first, the geometry variables are defined based on the analytically computed data. Type the following strings in the text editor window and save it.

```
// Dimensions of the machine
DefineConstant[ R_rot_in = { 20}];
DefineConstant[ R_rot_out = { 40}];
DefineConstant[ R_stat_out = { 96}];
DefineConstant[ Gap = { 0.6}];
DefineConstant[ b_d = { 21}];
DefineConstant[ hss_tot = { 32.4}];
DefineConstant[ bridge = { 0.6}];
DefineConstant[ w_PM = { 39}];
DefineConstant[ h_PM = { 4.9}];

// Mesh density points
DefineConstant[ mesh_gap = { Gap/2}];
DefineConstant[ mesh_fine = { 0.5}];
DefineConstant[ mesh_normal = { 3}];
DefineConstant[ mesh_coarse = { 10}];
```

The commented strings start with a double slash '//'. Each string with an operator should end with a semicolon. The geometry can be defined using scripting language or interactively. In this tutorial we use hybrid method, some parts of the geometry are defined using editing the source file *PMSM.geo* and other geometry features are defined interactively in the GMSH window.

The mesh is partly defined during the geometry definition. The prescribed mesh density is controlled by constants which names start with *mesh_*.

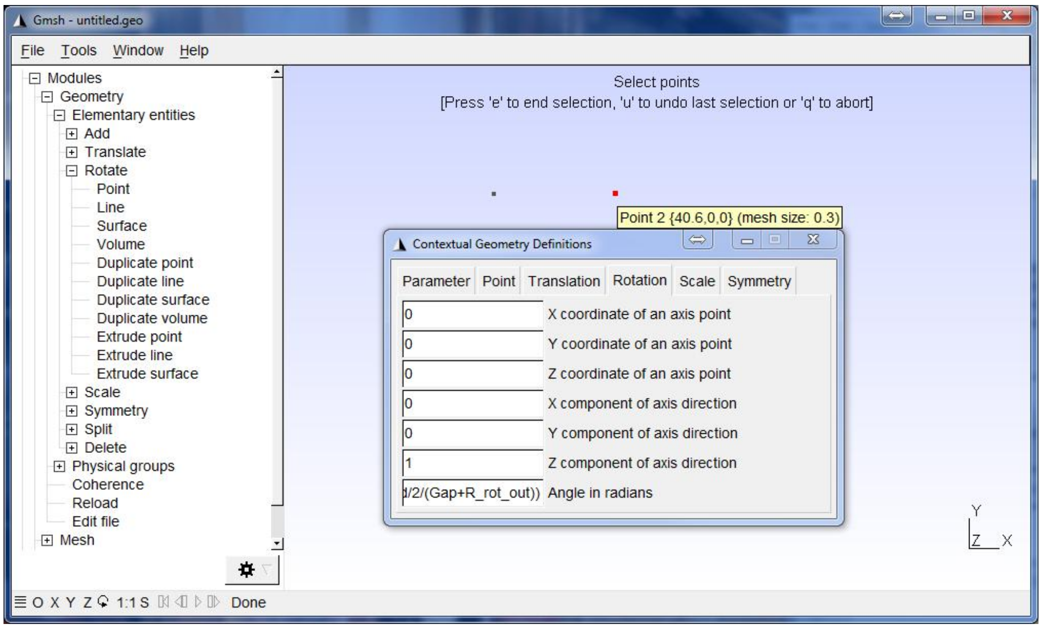
Create first point of the center of the rotor in the origin of the coordinate system by Geometry > Elementary entities > Add > Point. In the popup window verify that x,y,z coordinates are 0 and mesh element size is set to *mesh_coarse*, and press Add. Be sure that you are not moving mouse pointer over the drawing canvas as X,Y,Z coordinates will change according to the pointer position over canvas.



0	X coordinate
0	Y coordinate
0	Z coordinate
mesh_coarse	Prescribed mesh element size at point
0.1 0.1 0.1	Snapping grid spacing

Next, add second point in the center of the tooth in the air gap. Ensure that the X coordinate is Gap+R_rot_out and mesh size is mesh_gap.

Then, duplicate the last point two times by rotation to define the tooth tips in the air gap by Geometry > Elementary entities > Rotate > Duplicate point. Ensure that axis point has zero coordinates and axis direction has Z component 1. Angle in radians is defined as ±Asin(b_d/2/(Gap+R_rot_out)). Then by mouse select the second point and press ‘e’. Notice that Gmsh provides instructions for completion of active operation at the top of the drawing canvas.



Select points
[Press 'e' to end selection, 'u' to undo last selection or 'q' to abort]

Point 2 {40.6,0,0} (mesh size: 0.3)

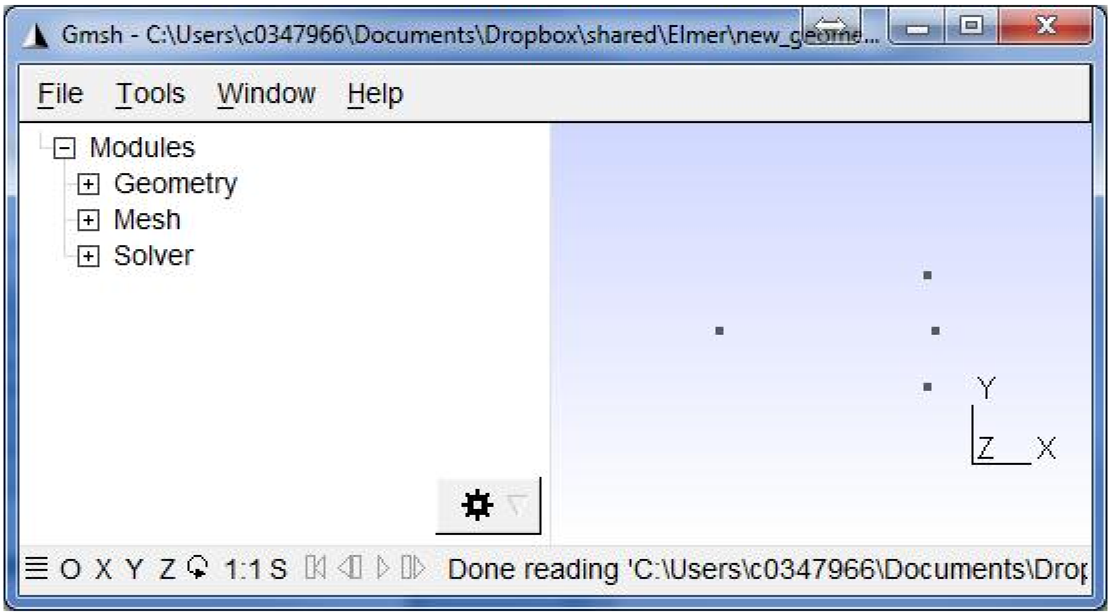
0	X coordinate of an axis point
0	Y coordinate of an axis point
0	Z coordinate of an axis point
0	X component of axis direction
0	Y component of axis direction
1	Z component of axis direction
/2/(Gap+R_rot_out))	Angle in radians

Done

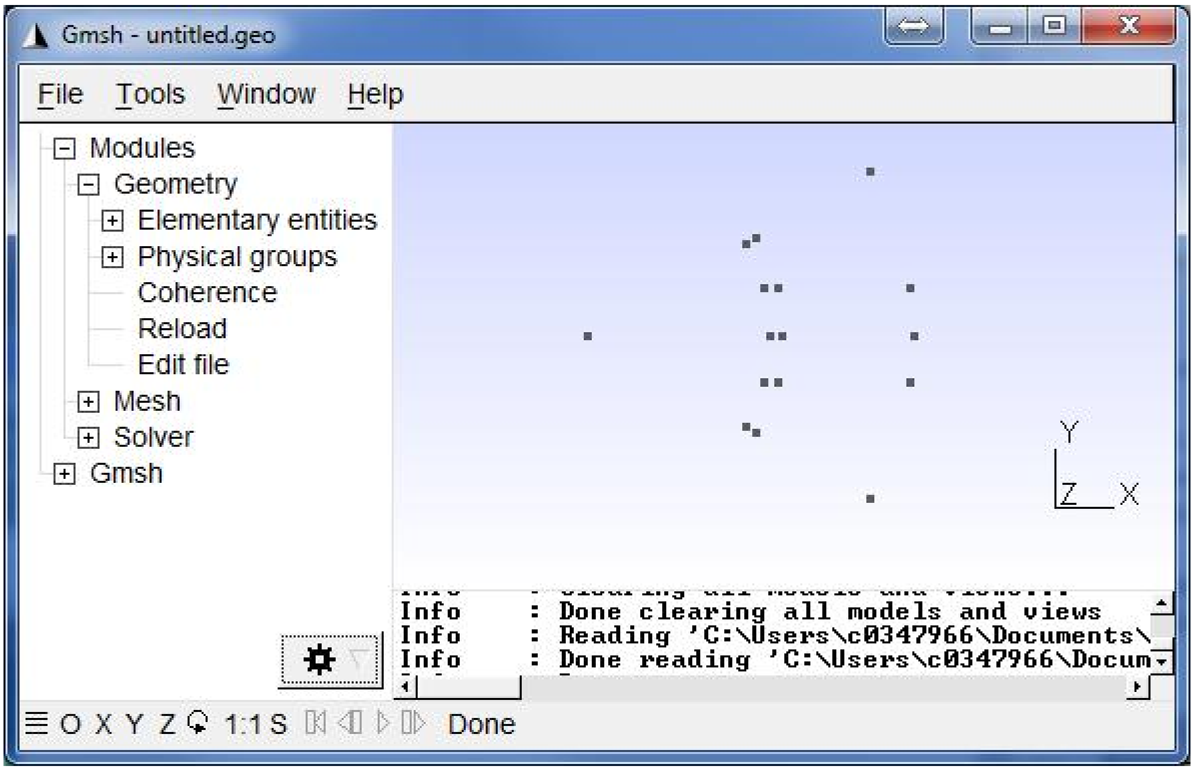
It is advisable that each time after inputting some parameter or entity in interactive mode, the result is verified by checking the geometry source file contents by Geometry > Edit file. There is no global “undo” capability. You will often need to edit a text file to correct mistakes. Simultaneously some comments could be added to the source file in order to simplify possible modification in the future. By this point the contents of the PMSM.geo file should be as follows.

```
// Dimensions of the machine
DefineConstant[ R_rot_in = { 20}];
DefineConstant[ R_rot_out = { 40}];
DefineConstant[ R_stat_out = { 96}];
DefineConstant[ Gap = { 0.6}];
DefineConstant[ b_d = { 21}];
DefineConstant[ hss_tot = { 32.4}];
DefineConstant[ bridge = { 0.6}];
DefineConstant[ w_PM = { 39}];
DefineConstant[ h_PM = { 4.9}];
DefineConstant[ h_wedge = { 3}];

// Mesh density points
DefineConstant[ mesh_gap = { Gap/2}];
DefineConstant[ mesh_fine = { 0.5}];
DefineConstant[ mesh_normal = { 3}];
DefineConstant[ mesh_coarse = { 7}];

//=================================================================
// Geometry definition

Point(1) = {0, 0, 0, mesh_coarse};

// ==========================================================
// =============== Stator Geometry ==============================
// ==========================================================

//Tooth bottom
Point(2) = {Gap+R_rot_out, 0, 0, mesh_gap};
Rotate {{0, 0, 1}, {0, 0, 0}, Asin(b_d/2/(Gap+R_rot_out))} {
  Duplicata { Point{2}; }
}
Rotate {{0, 0, 1}, {0, 0, 0}, -Asin(b_d/2/(Gap+R_rot_out))} {
  Duplicata { Point{2}; }
}
```

The GMSH window shows 4 points – one for the machine axis and 3 which are on the head of the first tooth.

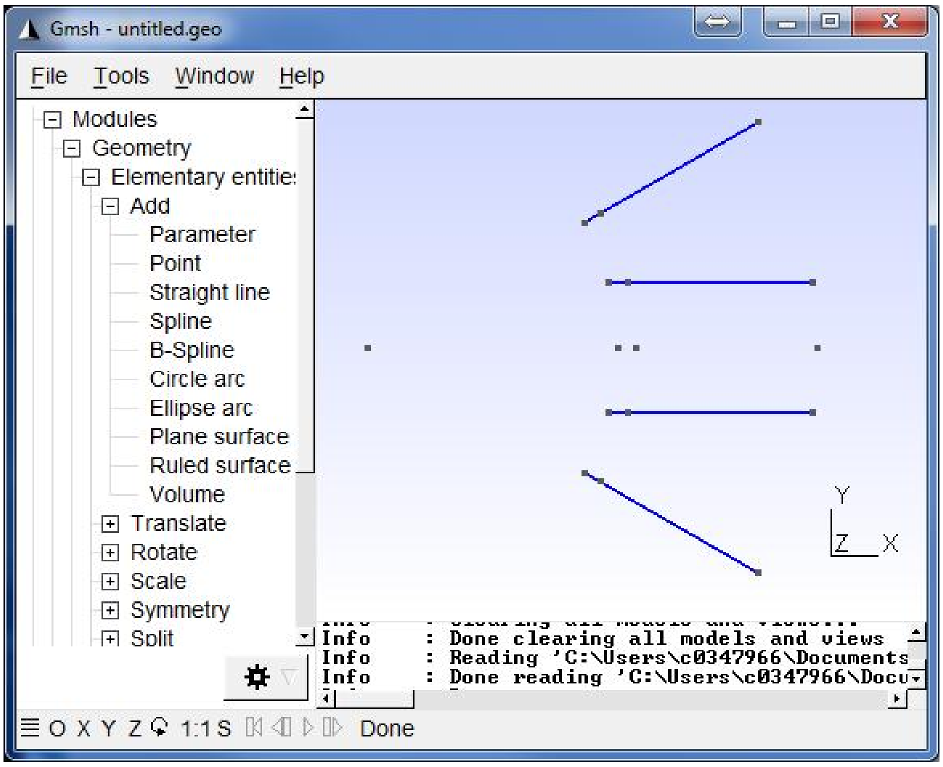

Gmsh - C:\Users\c0347966\Documents\Dropbox\shared\Elmer\new_geome...
File
Tools
Window
Help
Modules
Geometry
Mesh
Solver
Y
Z
X
O X Y Z 1:1 S
Done reading 'C:\Users\c0347966\Documents\Drop

Next, we define 2 points in the tooth – one represents the height of semi-magnetic wedges (X coordinate Gap+R_rot_out+h_wedge, mesh density is mesh_fine), the other represents the height of the tooth (X coordinate Gap+R_rot_out+hss_tot, mesh density is mesh_normal). And then, duplicate these points by rotation – rotation angles are ±Asin(b_d/2/(Gap+R_rot_out+h_wedge)) and ±Asin(b_d/2/(Gap+R_rot_out+hss_tot)). Then duplicate tooth points one more time to define tooth coil edges. The rotation angle is ±Pi/6. The result is 16 points of one coil sector.



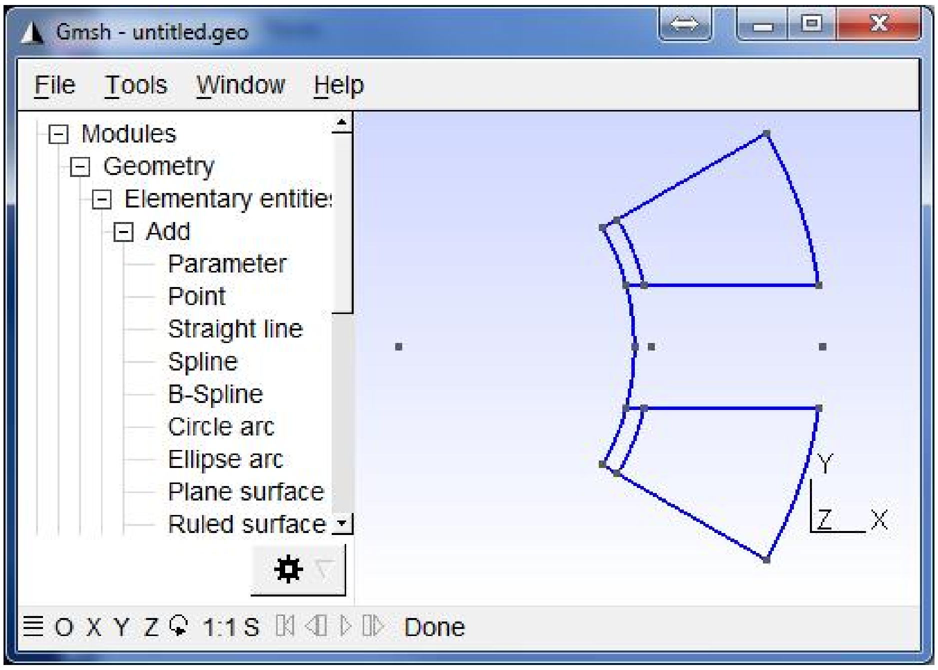
```
Info     : Done clearing all models and views
Info     : Reading 'C:\Users\c0347966\Documents\
Info     : Done reading 'C:\Users\c0347966\Docum
```

Then, the lines are defined. These lines represent borders of the coils and wedges. Select Geometry > Elementary entities > Add > Straight line and consequently select points in the window of GMSH to obtain the following view, and then press ‘q’.

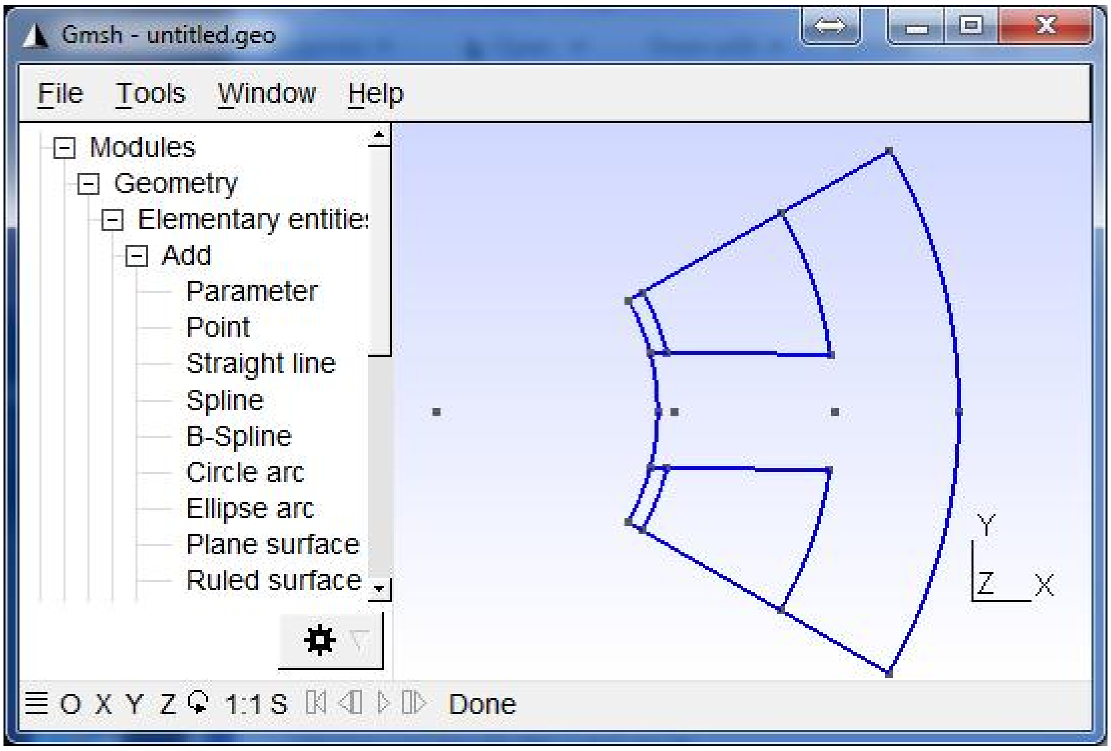


```
Info     : Done clearing all models and views
Info     : Reading 'C:\Users\c0347966\Documents
Info     : Done reading 'C:\Users\c0347966\Docu
```

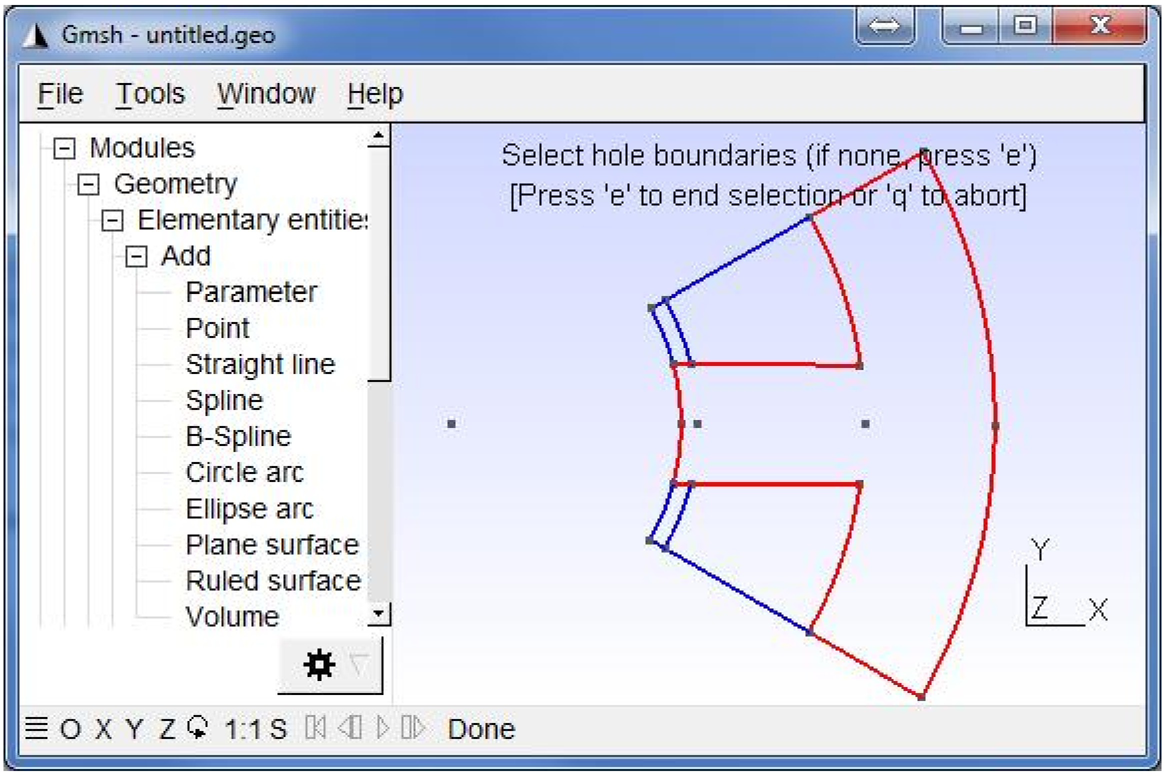
Then, select Geometry > Elementary entities > Add > Circle arc and consequently select points to complete the geometry of the coil and sector wedges.



Add outer edge of the stator yoke and boundaries of the tooth coil segment by adding point (with X coordinate R_stat_out and mesh density mesh_coarse), then duplicating it and adding arcs and lines.

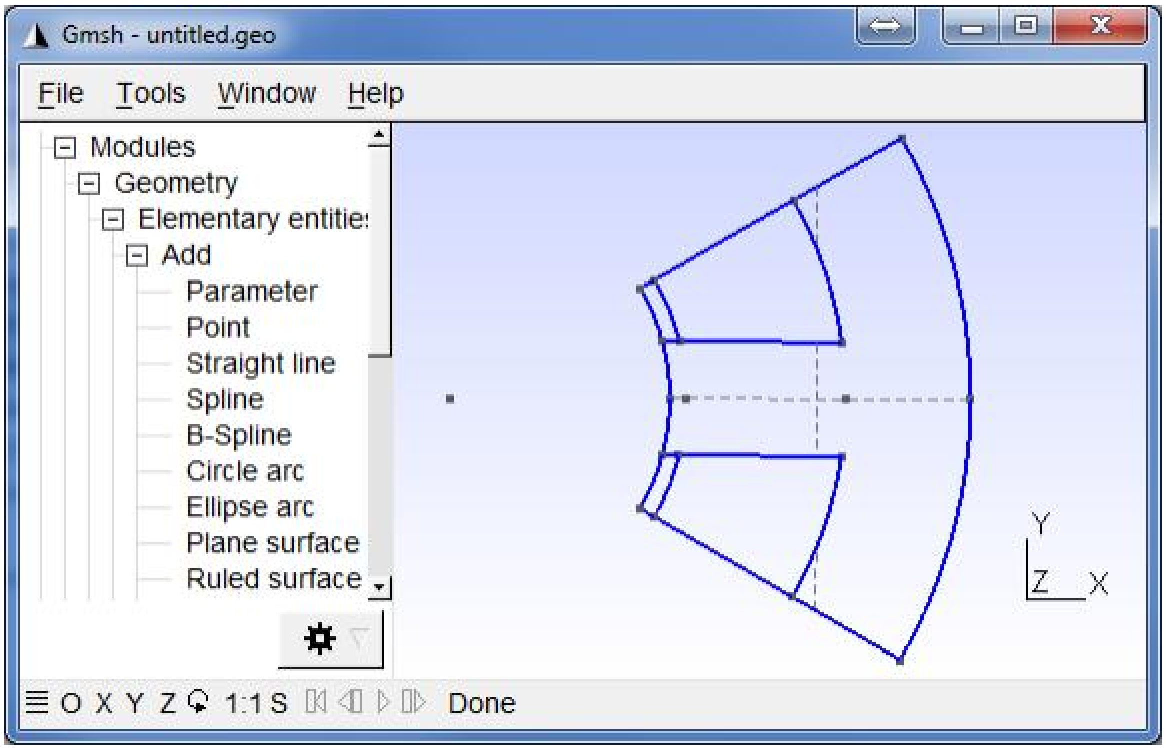


No the plane surfaces are defined. The plane surface for the steel segment is defined first by selecting Geometry > Elementary entities > Add > Plane surface, and choosing boundaries of the steel segment and pressing ‘e’.

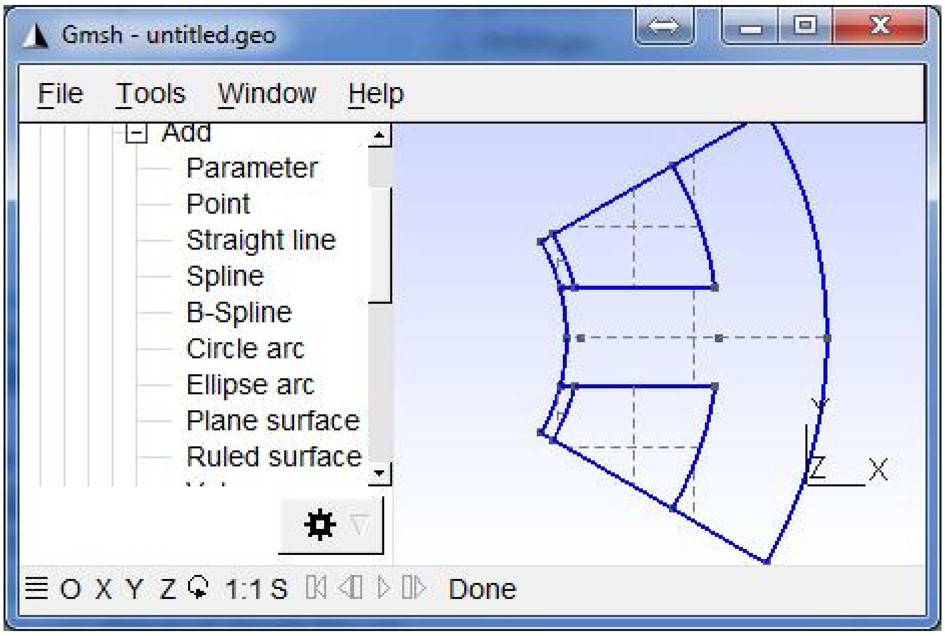


Select hole boundaries (if none, press 'e')
[Press 'e' to end selection or 'q' to abort]

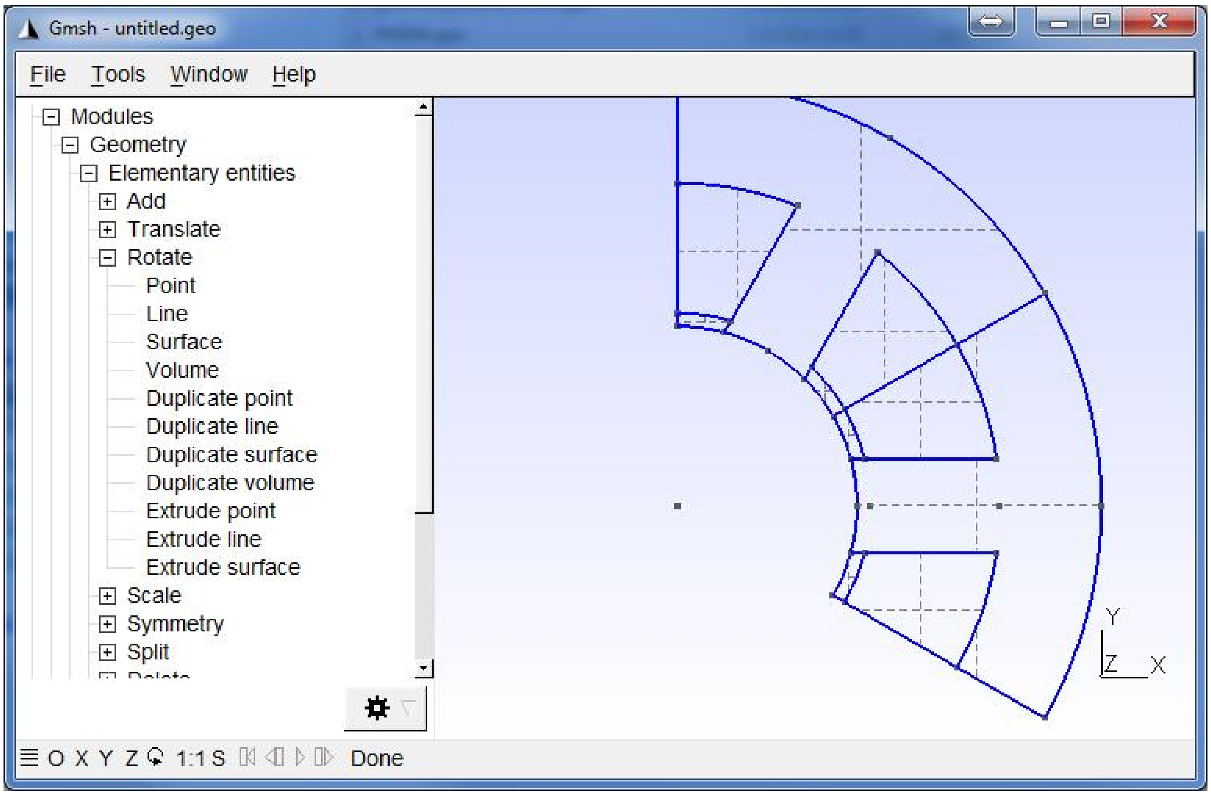
The created plane surface is marked with a gray dashed cross.



Similarly, define 4 more plane surfaces for positive and negative coil sides and for two wedges.



Next, the geometry of the stator is finalized by duplicating the plane surfaces of the created stator segment. Select Geometry > Elementary entities > Rotate > Duplicate surface. The angle of rotation around Z axis is pi/3, as the machine has 6 coils. Select all 5 surfaces and press ‘e’.



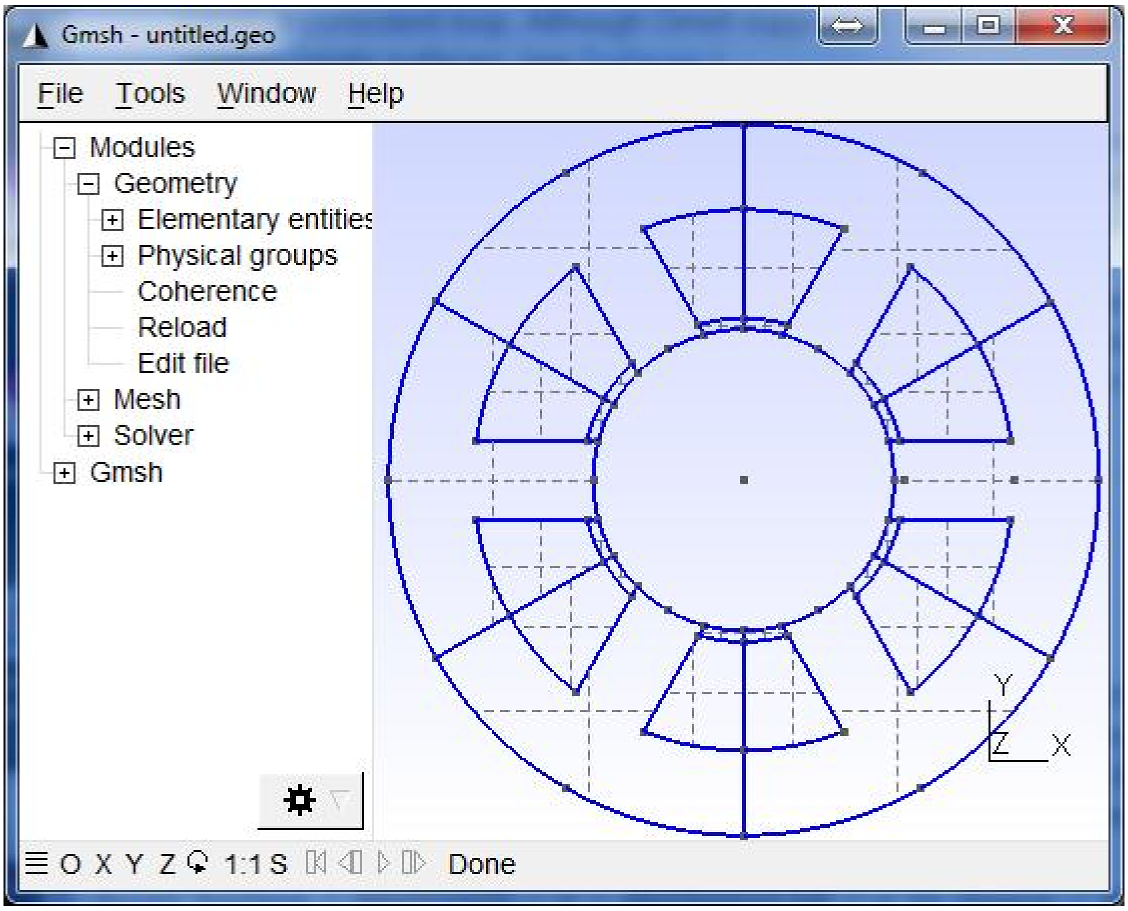
To complete the geometry, one should repeat previous procedure 5 times. Here, the scripting possibilities of GMSH are utilized instead. Open the source file PMSM.geo by Geometry > Edit file, and modify the last command

```
Rotate {{0, 0, 1}, {0, 0, 0}, Pi/3} {
  Duplicata { Surface{22, 24, 28, 30, 26}; }
}
```

by inserting it into a for-loop as follows

```
//Duplicate stator sectors
For it In {1:5}
            Rotate {{0, 0, 1}, {0, 0, 0}, it*Pi/3} {
             Duplicata { Surface{22, 24, 28, 30, 26}; }
            }
EndFor
```

In this code piece a new variable it is introduced to work as an iterator to count all the remaining stator sectors. Close the text editor and select Geometry > Reload. The stator geometry is complete.



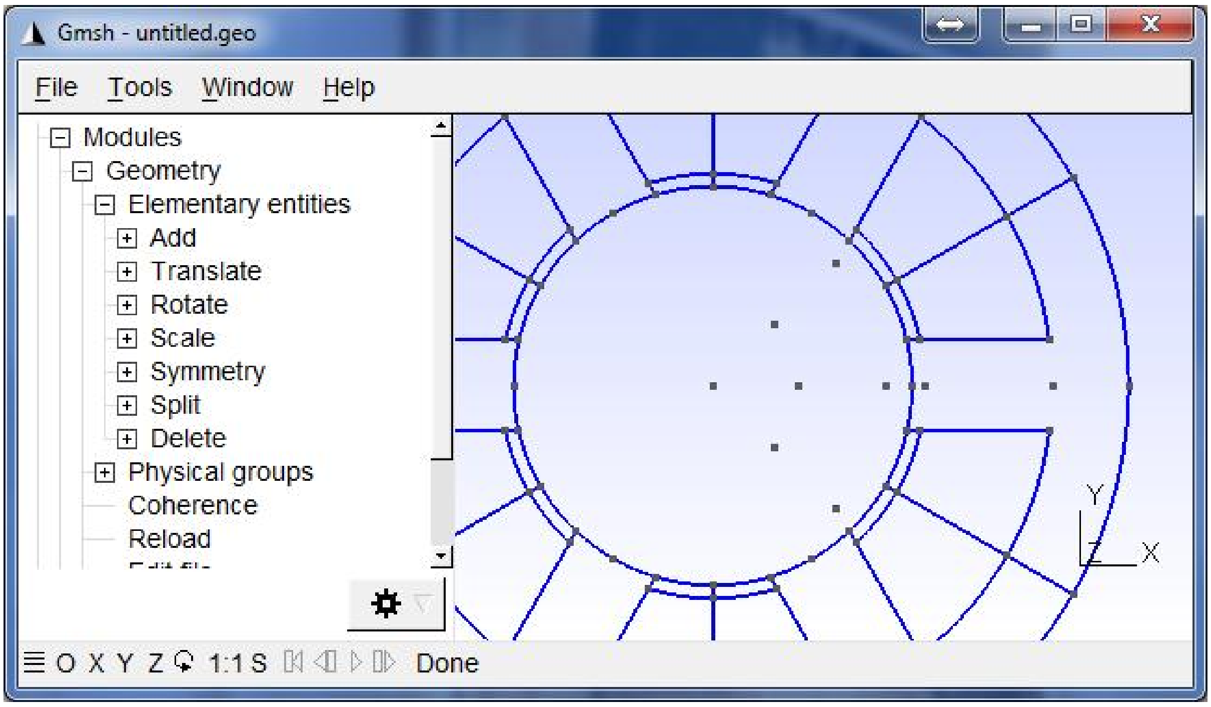
GMSH is very effective when it is required to create geometry with multiple repetitive segments.

Rotor Geometry

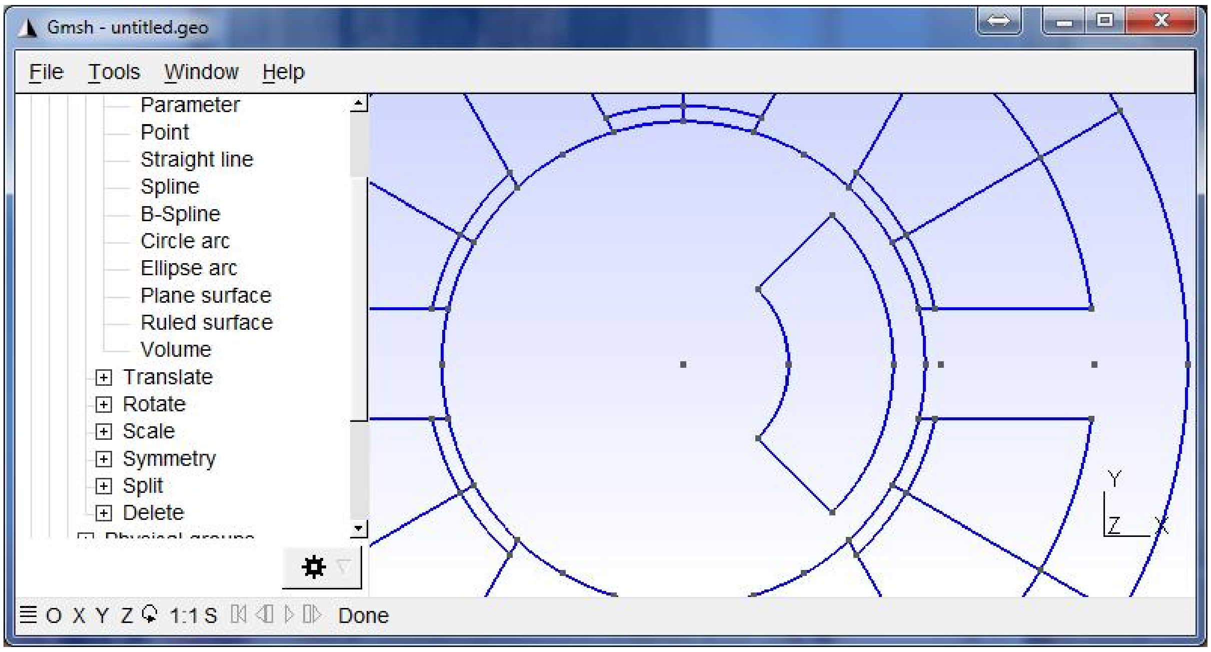
The rotor of the modeled machine consists of four pole segments. Here, the first segment is created. First, two points on inner and outer rotor surfaces are created (Geometry > Elementary entities > Add > Point, X coordinates are R_rot_in and R_rot_out, and mesh densities are mesh_coarse and mesh_gap correspondingly), and then they are duplicated to denote the borders of the sector (rotation angle along Z axis is ±pi/4). The resultant code in the source file is

```
// ==========================================================
// =============== Rotor geometry ===========================
// ==========================================================

Point(379) = {R_rot_in, 0, 0, mesh_coarse};
Point(380) = {R_rot_out, 0, 0, mesh_gap};

//Rotor sector borders
Rotate {{0, 0, 1}, {0, 0, 0}, Pi/4} {
  Duplicata { Point{379, 380}; }
}
//Rotor geometry
Rotate {{0, 0, 1}, {0, 0, 0}, -Pi/4} {
  Duplicata { Point{379, 380}; }
}
```

In order to verify visually that points are built properly, it is advisable to temporarily increase the air gap height in the script file. Select Geometry > Edit file, change the Gap variable to 6, close editor, and select Geometry > Reload. Now the air gap is clearly seen.



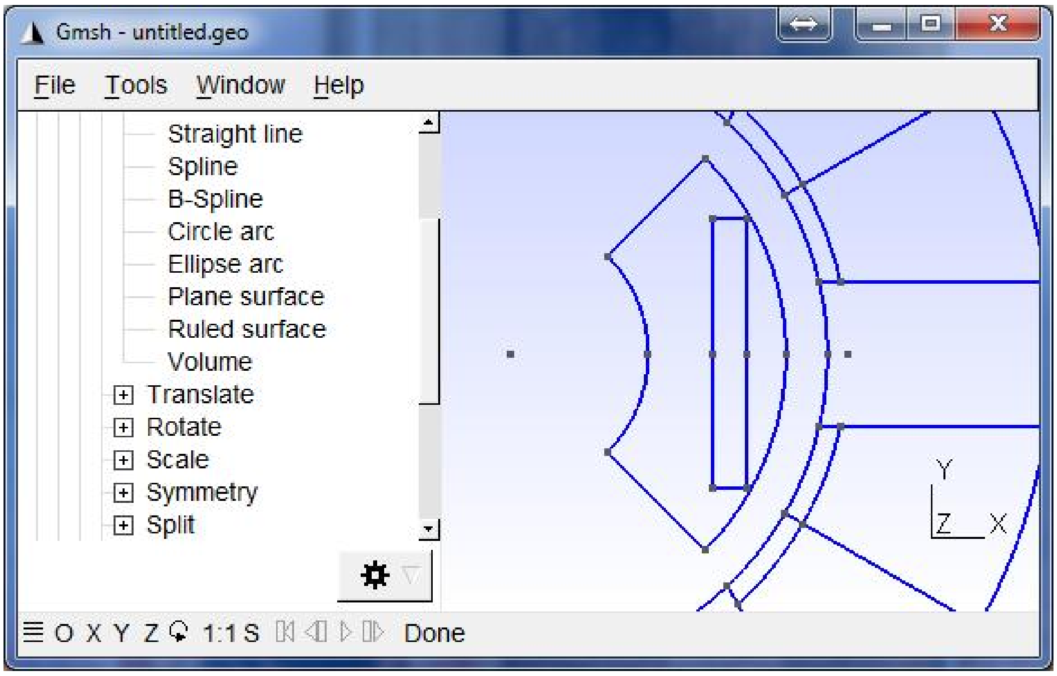
Then the borders of the pole sector are drawn – two straight lines and four arcs.



Then the edges of the rectangular permanent magnet face are defined. The X,Y,Z coordinates and correspondent mesh densities are given in the next table. The variable bridge indicates the distance of the magnet corners from the rotor surface. In other words this is height of the iron bridge. This height should be considered carefully taking into account centrifugal forces and lamination material mechanical strength. It should be as small as possible in order to decrease the magnetic flux leakage, but not too thin in order to maintain the mechanical integrity of the rotor at high speeds. Usually near the bridge the magnetic flux density changes rapidly, and therefore the nonlinear electrical steel properties can also change rapidly. Therefore, at these points the mesh density should be high enough in order to capture correctly the magnetic flux leakage in the rotor.

Point	X coordinate	Y coordinate	mesh density
1	(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))	0	mesh_normal
2	(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))	w_PM/2	mesh_gap
3	(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))	-w_PM/2	mesh_gap
4	(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))-h_PM	0	mesh_normal
5	(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))-h_PM	w_PM/2	mesh_normal
6	(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))-h_PM	-w_PM/2	mesh_normal

After points are built they are connected with 6 straight lines to create the edges of the PM face.



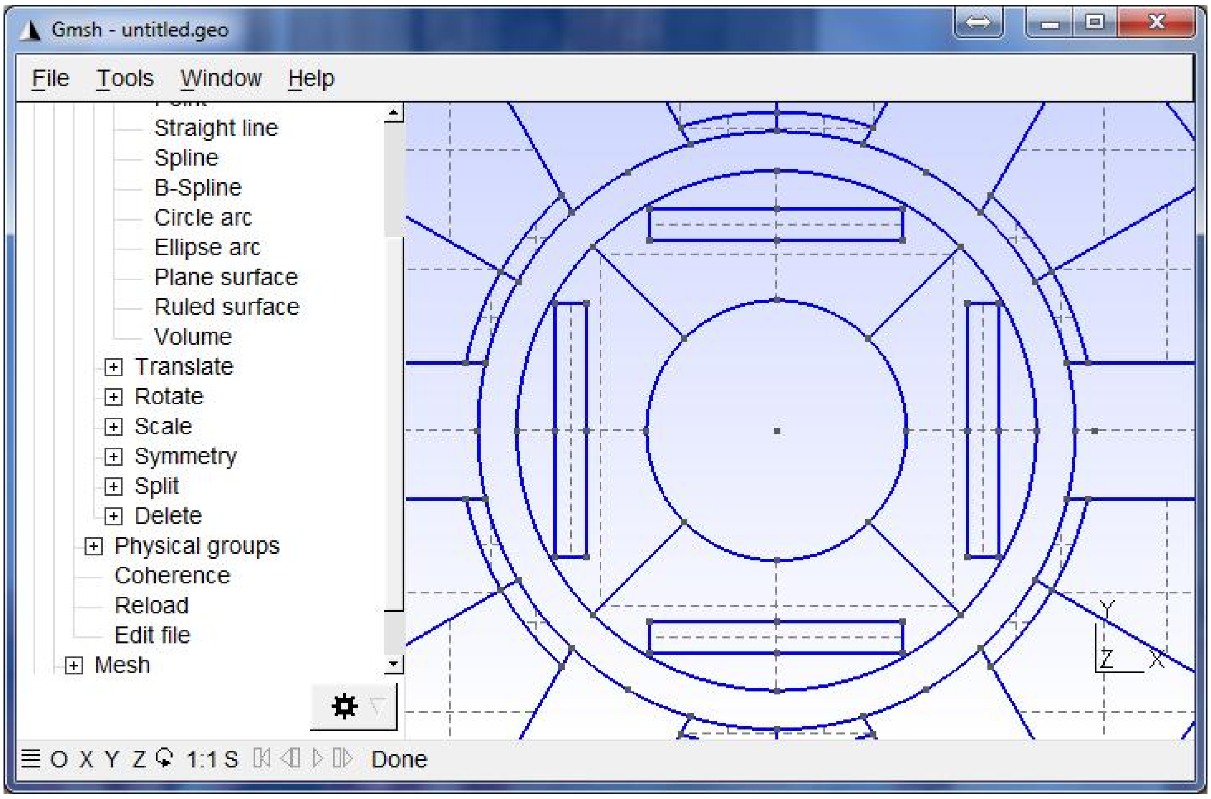
Then, the magnet plane surface is created. By Geometry > Elementary entities > Add > Plane surface, and selecting the 6 lines of the PM edges. And after that the plane surface of the iron rotor sector is created by first selecting the edges of the sector, and then selecting the edges of the magnet to create a hollow plane surface.

When creating a hollow surface the GMSH can produce erroneous script entry. Check the source file, and verify that the plane surface for the pole sector iron (198) is created in the way that first goes a line loop with the edges of the iron sector (197), and after that goes the line loop of the PM edges (195), as it is shown in the following listing.

```
//PM surface
Line Loop(188) = {184, 185, 186, 187, 182, 183};
Plane Surface(189) = {188};

//Pole sector iron
Line Loop(190) = {181, 176, -179, -178, 177, 180};
Plane Surface(191) = {190, 188};
```

When rotor pole plane surfaces are created, then they can be duplicated in order to complete the rotor geometry.

```
//Duplicate rotor poles
For it In {1:3}
          Rotate {{0, 0, 1}, {0, 0, 0}, it*Pi/2} {
            Duplicata { Surface{191, 189}; }
          }
EndFor
```



The last part in the rotor geometry is the shaft plane surface.

```
//shaft plane surface
Line Loop(234) = {196, -179, -178, 223, 224, 209, 210, 195};
Plane Surface(235) = {234};
```

Air Gap

Next, the sliding surface in the air gap is created. Sliding surface divides rotating and stationary parts of the geometry. Here, the sliding surface is placed in the center of the air gap.

Create 4 points in the center of the air gap at every pi/2 with the mesh density mesh_gap, and then create 4 arc lines. Use mouse scroll to zoom in and out to select precisely the arc points.

```
Point(529) = {Gap/2+R_rot_out, 0, 0, mesh_gap};
Point(530) = {0, Gap/2+R_rot_out, 0, mesh_gap};
Point(531) = {0, -Gap/2-R_rot_out, 0, mesh_gap};
Point(532) = {-Gap/2-R_rot_out, 0, 0, mesh_gap};

Circle(236) = {529, 1, 530};
Circle(237) = {530, 1, 532};
Circle(238) = {532, 1, 531};
Circle(239) = {531, 1, 529};
```

Now, the plane surface of the stator part of the air gap is created by selecting Geometry > Elementary entities > Add > Plane surface. Select first boundary on the stator surface, and then sliding surface. If plane surface is defined properly the four dashed lines can be seen in the stator air gap denoting the plane surface with a hole.

```
Line Loop(240) = {13, 14, -57, -39, -38, -52, -86, -68, -67, -81, -115, -97, -
96, -110, -144, -126, -125, -139, -173, -155, -154, -168, 11, 12};
Line Loop(241) = {239, 236, 237, 238};
Plane Surface(242) = {240, 241};
```

The rotor geometry is finalized by creation of the rotor air gap surface. Verify that when the plane surface of the rotor part of the air gap is created, the first index belongs to the sliding surface and the second belongs to the Line Loop of the rotor surface.

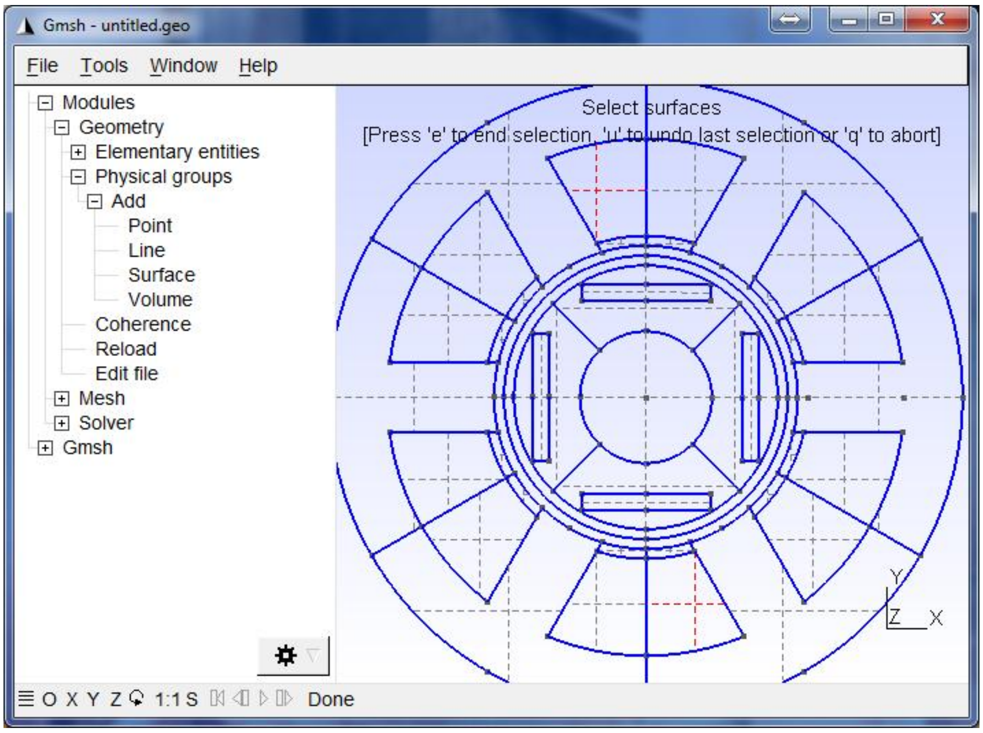
```
//create air gap part of the rotor
Line Loop(243) = {181, 198, 193, 212, 207, 226, 221, 180};
Plane Surface(244) = {241, 243};
```

Check that on the geometry view all the surfaces are denoted by gray dashed crosses. Next, the physical bodies are created from the plane surfaces.

Physical Bodies

The physical bodies combine plane surfaces with the same physical properties. The creation of physical bodies simplifies the assignment of physical properties and boundary conditions to the model.

First, the physical surfaces of the stator windings are defined according to the analytical design. The indexes are from 1 through 6. Select Geometry > Physical groups > Add > Surface, and then select all phase U positive coil sides plane surfaces so that they are highlighted. After that press 'e' and 'q' to finalize the selection.



Select surfaces
[Press 'e' to end selection, 'u' to undo last selection or 'q' to abort]

Create 6 Physical surfaces for all phases, positive and negative. Change indexes to 1—6. The book-keeping of indexes of physical surfaces and boundaries is required to simplify the manual creation of SIF. When SIF is created with ElmerGUI, the boundary conditions and body properties are assigned automatically in interactive mode to the selected entities.

```
// ============================================================
// =============== Bodies ===============================
// ============================================================

// u+-
Physical Surface(1) = {88, 175};
Physical Surface(2) = {73, 160};

// v+-
Physical Surface(3) = {26, 117};
Physical Surface(4) = {24, 102};

// w+-
Physical Surface(5) = {59, 146};
Physical Surface(6) = {44, 131};
```

Next, the stator iron plane surfaces are united into one stator iron physical group.

Check that in the source file that 6 indexes of the plane surfaces are united into one Physical Surface. Change the index of the physical surface to 7, so later we could refer to 7 as the stator iron physical body. Do not forget to save changes to the source file and reload it after each change.

```
//Stator iron
Physical Surface(7) = {31, 22, 147, 118, 89, 60};
```

Next, 4 iron rotor pole sectors are united into a physical surface with index 8. (see the source file in the Appendix A)

Next, 12 regions of the wedges are united into a physical surface with index 9. The shaft Physical Surface has index 10.

```
//Wedges
Physical Surface(9) = {28, 54, 49, 83, 78, 112, 107, 141, 136, 170, 165, 30};

//shaft
Physical Surface(10) = {235};
```

Then, the permanent magnet physical faces are defined. Indexes are 11 – 14.

```
//PMs
Physical Surface(11) = {189};
Physical Surface(12) = {205};
Physical Surface(13) = {219};
Physical Surface(14) = {233};
```

Next the air gap region is defined as two surfaces – one belonging to stationary stator, and the second belongs to the moving rotor. These surfaces are selected by clicking on the gray dashed lines of the temporarily increased air gap regions. Change the indexes of physical surfaces to 9 and 10.

```
//stator air gap
Physical Surface(15) = {242};

//rotor air gap
Physical Surface(16) = {244};
```

And finally the two boundary lines are defined by Geometry > Physical groups > Add > Line. The first is the outer boundary of the computation domain (outer stator boundary). And the second line is sliding line between rotating and stationary regions.

```
// ==========================================================
// =============== Boundaries ===============================
// ==========================================================

//outer boundary
Physical Line(1) = {32, 33, 61, 62, 90, 91, 119, 120, 148, 149, 19, 20};
//sliding surface
Physical Line(2) = {239, 236, 237, 238};
```

And finally the air gap width is changed back to 0.6.

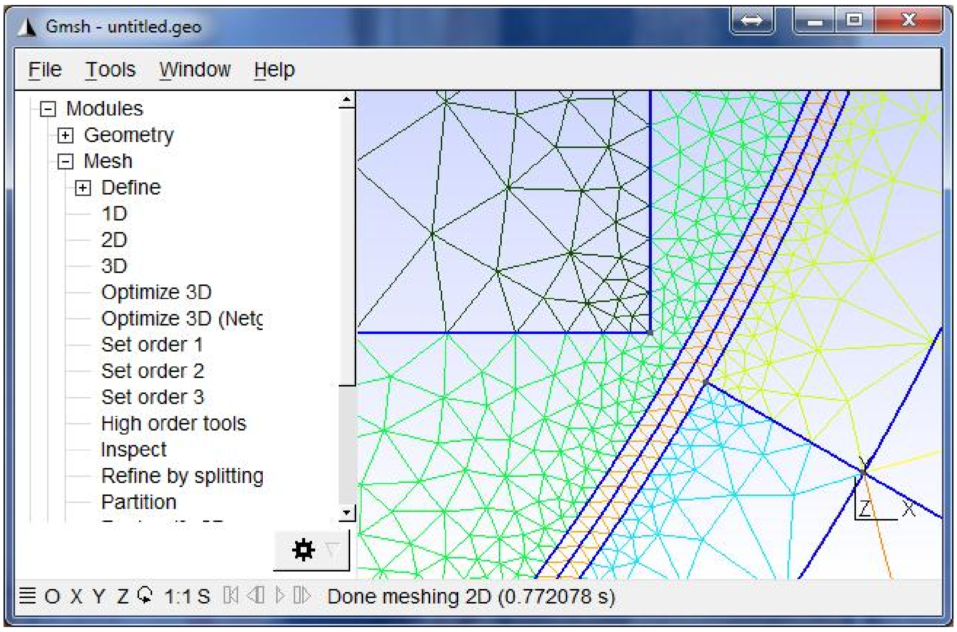
3. Meshing

The GMSH allows creation of very sophisticated meshes in 2D and 3D. For further information refer to the GMSH documentation. In this modeling case the mesh density is controlled simply by mesh points.

```
// Mesh density points
DefineConstant[ mesh_gap = { Gap/2}];
DefineConstant[ mesh_fine = { 0.5}];
DefineConstant[ mesh_normal = { 3}];
DefineConstant[ mesh_coarse = { 10}];
```

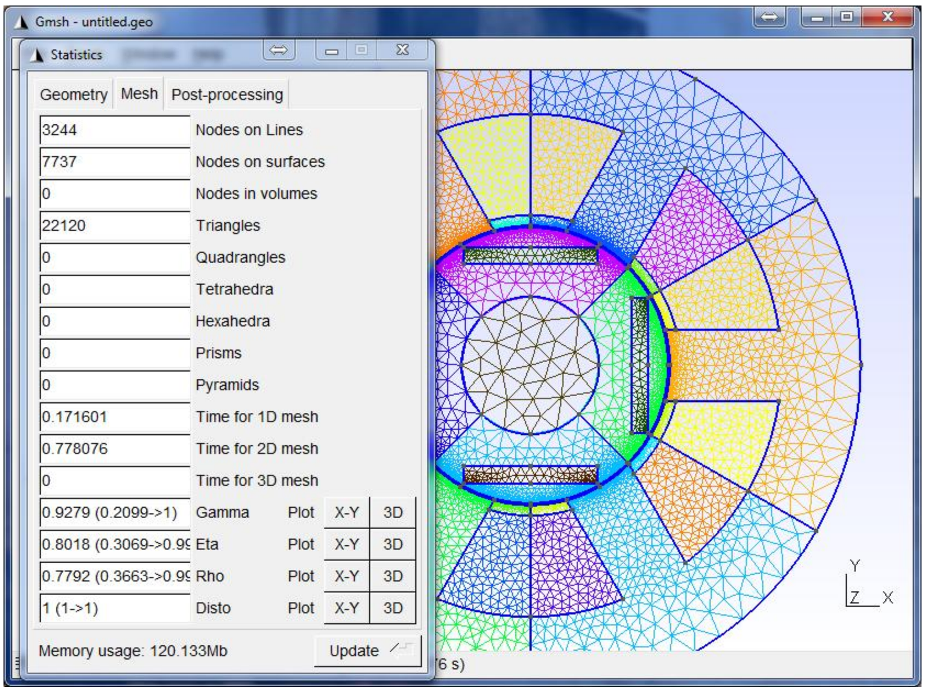
To obtain accurate results the mesh density should be highest to approach as close as possible to the continuous field solution. However, a high number of elements significantly increases the computational time. Therefore, a vise selection of the mesh density over the computational domain should be performed. The most important region in electrical machines is the air gap – here the most of the magnetic energy is stored. The air gap mesh density should be the highest. The mesh density near the domain boundaries can be low as the magnetic field near the boundaries changes not significantly. Different mesh optimization and refinement techniques can be used with the GMSH, however for this example the simplest mesh point approach is utilized for simplification purposes.

Select Mesh > Define > 2D in order to generate 2D mesh.



Done meshing 2D (0.772078 s)

In the air gap there are two layers of elements. The accuracy of torque estimation strongly depends on the mesh density in the air gap. The denser mesh gives better results in the expense of computational time.



3244	Nodes on Lines			
7737	Nodes on surfaces			
0	Nodes in volumes			
22120	Triangles			
0	Quadrangles			
0	Tetrahedra			
0	Hexahedra			
0	Prisms			
0	Pyramids			
0.171601	Time for 1D mesh			
0.778076	Time for 2D mesh			
0	Time for 3D mesh			
0.9279 (0.2099->1)	Gamma	Plot	X-Y	3D
0.8018 (0.3069->0.99	Eta	Plot	X-Y	3D
0.7792 (0.3663->0.99	Rho	Plot	X-Y	3D
1 (1->1)	Disto	Plot	X-Y	3D

From the figure it is seen that the highest mesh density is in the air gap, and the lowest is near the outer boundary and in the shaft. 22120 triangular elements were created.

Set the order of elements to 2 by Mesh > Define > Set order 2. This increases the number of nodes in the mesh. *ElmerSolver* includes some internal possibilities to refine the mesh by equally splitting of existent mesh elements or increasing the mesh order (keywords `Mesh Levels = 2` and `Element = p:2`). These possibilities can be utilized in the SIF's `Simulation section`.

Save the generated mesh by Mesh > Save. Verify that PMSM.msh file is created in the working directory. The size is about 3 Mb.

Next, the file format is converted from .msh to ElmerSolver format using console utility ElmerGrid. The console command is

```
➢ ElmerGrid 14 2 PMSM.msh
```

It has 3 input parameters – input file format, output file format, and the name of the input file. For more information on ElmerGrid capabilities and available line arguments refer to ElmerGrid Manual [3]. After executing the ElmerGrid the working directory will contain a folder named PMSM with 4 files starting with mesh..

Sometimes, the internal mesh refinement options work unreliably. The element order can be increased directly in the mesh file when converting mesh from a first order external mesh format to the Elmer mesh format by adding parameter `–increase` to the ElmerGrid parameters.

If book-keeping of indexes is not performed, the ElmerGrid command should additionally contain parameter –*autoclean*. The –autoclean option builds new consecutive index of physical surfaces starting from 1. The indexes are ordered according to the creation order in the *geo* file.

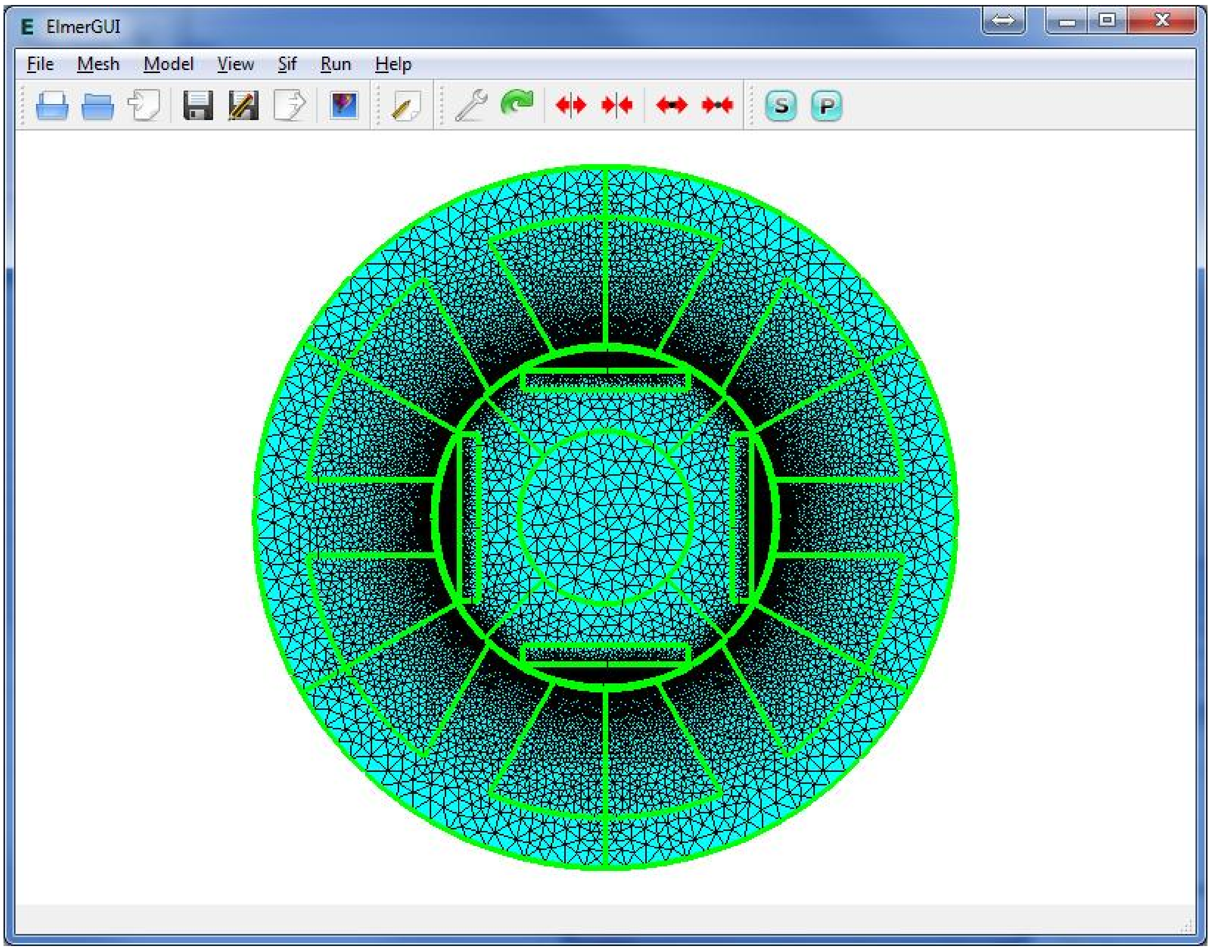
ElmerGrid allows for conversion between several mesh formats. Also this utility can partition mesh for solving the problem in parallel. For complete list of ElmerGrid capabilities and parameters refer to [3].

The complete listing of the GMSH input file is given in Appendix A.

4. Model Preparation

The solving process starts from the preparation of a solver input file (SIF), which contains basic simulation settings, description of the physical regions and boundary conditions, definition of the materials, solver specific settings and settings for the numerical solvers (direct or iterative). Before reading this section it is advisable to read [5].

It is possible, and, perhaps, easy for beginners, to prepare the SIF file using ElmerGUI in interactive mode. However, in this section the SIF is created manually. An example case SIF with periodicities in a rotating electrical machine can be found also in Elmer source code *Elmer/fem/tests/mgdyn2D_em*.



In this example case the torque ripple is modeled at first, and, then, the back EMF waveform, and loading with nominal current will be studied.

Simulation Initialization

In the working directory create a text file and rename it to case.sif. The SIF (solver input file) is read by the ElmerSolver at the start of the simulation. The complete listing of the SIF is given in the Appendix B.

First, the simulation parameters are introduced. The SIF begins with the following simulation parameters.

```
! PMSM 2D Magneto-Static Simulation
! Parameters
$ w_m = 3000/60*2*pi                    ! [rad/s mech.] mechanical frequency
$ pp = 2                                ! number of polepairs
$ w_el = w_m*pp                         ! [rad/s el.] electrical frequency
$ B_PM = 1.17                           ! [T] remanent flux density
$ mu_PM = 1.06                          ! relative permeability of PMs
$ H_PM = B_PM/(mu_PM*pi*4d-7)           ! [A/m] magnetization of PMs
$ Id = 0                                ! [A] d-axis current
$ Iq = 0                                ! [A] q-axis current
$ Nph = 186                             ! Number of phase turns
$ Scs = 0.00154555                      ! [m^2] area of the coil side
```

The comments start with the '!'. Symbol '$' denotes a variable, value of which is evaluated at the start of the simulation.

The SIF contains several sections. The first one is the Header section where paths to the mesh files and path for the results are specified (the paths should exist). For detailed description of the keywords and sections refer to [5].

```
Header
  CHECK KEYWORDS Warn
  Mesh DB "PMSM" "."
  Include Path ""
  Results Directory "results"
End
```

Next section is the constants section where simulation constants are defined. For the 2D electromagnetic problems this section can be left empty as the constants are defined within the solver.

```
Constants
   Permittivity of Vacuum = 8.8542e-12
End
```

Next section is simulation section dedicated to describe general simulation parameters which are not specific to a particular involved physical model.

```
Simulation
  Max Output Level = 3
  Coordinate System = Cartesian
  Coordinate Scaling = 0.001
  Simulation Type = Transient
  Timestepping Method = BDF
  BDF Order = 2
  Timestep Sizes = $ 1/(w_el/2/pi)/6/30  !30 samples per el. period/6
  Timestep Intervals = 31
  Output Intervals = 1
  Solver Input File = case.sif
!!!!!!!! mesh interpolation tolerances
  Interpolation Numeric Epsilon = Real 5.0e-8
End
```

Max Output Level describes the verbosity of the ElmerSolver (to Elmer developers: it would be a great improvement if the level of verbosity can be set for each solver separately).

Coordinate Scaling is required in order to scale mesh units (mm) to SI units utilized by Elmer (m).

Next keywords (*Simulation Type, Timestepping Method, BDF Order, Timestep Sizes, Timestep Intervals*) describe the time-stepping procedure in order to implement the rotor motion with time.

The settings given in the listing allow for computations during one cogging torque period for this machine, which is equal to 1/6 of electrical period. And 31 points will be computed within the cogging torque period.

Interpolation Numeric Epsilon is required here to allow some tolerances when determining the coordinates along sliding surface.

Material Properties

In order to describe the material properties of the regions, a separate Material section is created for each material. To perform simple simulations, 6 materials are required – air, electrical steel, and 4 materials for PM's with different directions of magnetization.

The materials are indexed. These indexes will be used in the bodies sections to assign materials to bodies. First, air is described.

```
Material 1
  Name = "Air"
  Relative Permeability = 1
  Electric Conductivity = 0
End
```

The keywords describe the electromagnetic properties of the materials.
Next, the electromagnetic steel is described. The electromagnetic steel is described by the BH curve. The BH curve is interpolated from a table of curve points written in a text file in the next format.

```
0           0
0.1         29.38
0.2         37.39
0.3         45.24
0.4         50.18
0.5         55.45
0.6         62.07
0.7         70.32
0.8         81.92
0.9         96.58
1.0         118.68
1.1         155.51
1.2         226.67
1.3         416.07
1.4         1059.27
1.5         2756.65
1.59        5441.9
1.64        7069.65
1.67        8213.34
1.723       10000
1.859       20000
1.922       30000
1.991       50000
2.089       100000
2.233       200000
2.365       300000
4           10000000
```

Each point is written on a new line in table format in txt-format and saved in file 'BH'. The first column contains B-values in Tesla. The second column contains H-values in A/m.
The BH table is included in the SIF using the following structure. The BH curve is modeled using monotonic cubic interpolation.

```
Material 2
  Name = "Iron"
  Electric Conductivity = 0
! keyword is H-B curve, but values are in B-H format
  H-B Curve = Variable coupled iter
    Real Monotone Cubic
      Include BH
    End
End
```

The PMs have to be described by 4 different materials because they have different directions of magnetization. Moreover, the direction of magnetization changes in the absolute stationary Cartesian coordinate system of Elmer with the rotor motion. Therefore the direction of magnetization has to be position-dependent (or time-dependent if a speed is imposed).

The direction of magnetization depends on the rotor position. The rotor position, in turn, depends on the time, if a constant speed is imposed. The magnetization has two components – X and Y – defined in terms of Elmer as Magnetization 1 and Magnetization 2 correspondingly.

To calculate the X and Y components of the magnetization at each time step, a MATC expression can be used. For detailed information and example of usage see [5, 7]. The material for the first PM with initial direction of magnetization to the right side has the following properties.

```
Material 3
  Name = "PM_right"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "H_PM*cos(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "H_PM*sin(w_m*(tx(0)-tx(1)))"
  Electric Conductivity = 0
End
```

In the same way the X and Y components of the magnetization are defined for 3 remaining magnets. The last material is a semi-magnetic material of the wedges with relative permeability of 3. See SIF listing in the Appendix B.

Boundary Conditions and Body Forces

The outer boundary (the outer surface of the stator) has zero flux through which is specified by Dirichlet boundary condition. It has constant magnetic potential Az. The value of which is naturally selected as Az = 0. The magnetic flux can be only tangential near this boundary. From the .geo file the index of the target outer boundary is 1. This boundary condition is defined in Elmer as follows.

```
Boundary Condition 1
  Target Boundaries(1) = 1
  Name = "Boundary_Outer"
  Az = 0
End
```

The second boundary condition defines the sliding surface. On this surface with rotor motion the mesh becomes non-conforming. From the .geo file the target boundary index is 2. Such a condition is defined in Elmer as follows.

```
Boundary Condition 2
   Target Boundaries(1) = 2
   Name = "Sliding"
   Discontinuous Boundary = Logical True
   Save Line = True
   Mortar BC = 3
End
```

The rotor motion is obtained in Elmer using remapping of the mesh, which requires imposing of the rotational force on the moving bodies of the rotor. The rotational force is a position function of time in mechanical degrees. It is defined using MATC function as follows.

```
!rotation of the rotor
Body Force 1
  Name = "BodyForce_Rotation"
  Mesh Rotate 3 = Variable time, timestep size
    Real MATC "180/pi*w_m*(tx(0)-tx(1))"   ! in degrees
End
```

The body forces later will be used to impose currents in the windings to simulate the electrical machine under the load.

Solvers

Solver in Elmer is a computational module which performs some manipulations or computations on the mesh or field data. Usually the solver performs just one function – computation of potentials, or saving of previously computed data, or mesh manipulation. On the same domain different physical models can be simulated utilizing different solvers.

For the studied case at each time step several solvers are executed consequently. The first solver is mesh deformation solver which performs the moving domain rotation defined by the Body Force 1.

```
!mesh rotation
Solver 1
  Exec Solver = Before Timestep
  Equation = MeshDeform
  Procedure = "RigidMeshMapper" "RigidMeshMapper"
End
```

After mesh has been fixed for a new rotor position the main solver which calculates the magnetic vector potential field is executed using the following parameters.

```
! solver for magnetic vector potential A [Vs/m]
Solver 2
  Equation = MgDyn2D
  Procedure = "MagnetoDynamics2D" "MagnetoDynamics2D"
  Exec Solver = Always

  Nonlinear System Convergence Tolerance = 1.0e-5
  Nonlinear System Max Iterations = 30
  Nonlinear System Relaxation Factor = 1

  Linear System Solver = Direct
  Linear System Direct Method = UMFPACK
End
```

Exec Solver keyword instructs ElmerSolver when to execute this particular solver. Instruction *Always* allows execution of this solver at each time step.

Then follow parameters for a nonlinear iterative solution procedure. For detailed descriptions of possible keywords and settings see [5]. For a 2D MagnetoStatic problem the nonlinear system settings given above are found producing meaningful results within reasonable time.

There are two possibilities for algebraic linear system solver – iterative and direct. The iterative methods are preferable for large or 3D models. For small relatively simple 2D models the direct solvers produce reasonable results within shortest time. Therefore, for the given problem a direct linear solver UMFPACK is chosen in order to minimize the computational time.

Then follow solvers which simplify the post-processing of the results. First executes the B-field solver, which calculates the magnetic flux density in the domain from the solved magnetic vector potential A-field.

```
!solver for magnetic flux density B [T]
!does not require settings for nonlinear solver
Solver 3
  Equation = MgDyn2DPost
  Procedure = "MagnetoDynamics2D" "BSolver"
  Exec Solver = After Timestep

  Linear System Solver = Direct
  Linear System Direct Method = UMFPACK

  Discontinuous Galerkin = True
  Average Within Materials = True
End
```

The *Discontinuous Galerkin* keyword allows for the B-field be discontinuous at the borders between materials with low and high magnetic permeabilities.

Next solvers are required to save the solved fields into VTU format for post-processing with ParaView and to save the scalar torque computed at each timestep. The available scalars are different for various solvers – see list of available scalars for a specific solver in the Models Manual [6] or in the FORTRAN sourcecode of the solver.

```
!save to VTU format for ParaView with faces' IDs
Solver 4
  Exec Solver = After Timestep
  Procedure = "ResultOutputSolve" "ResultOutputSolver"
  Output File Name = "step"
  Vtu Format = True
  Binary Output = True
  Single Precision = True
  Save Geometry Ids = True
End

!save scalar variables - torque T [Nm]
Solver 5
  Exec Solver = After Timestep
  Filename = "scalars.dat"
  Procedure = "SaveData" "SaveScalars"
  Show Norm Index = 1
End
```

The order of the execution is determined in the Equation section using keyword *Active Solvers* and solvers' indexes. Several Equation sections can be utilized e.g. for multi-physical problems within the same mesh.

```
Equation 1
  Name = "ModelDomain"
  Active Solvers(5) = 1 2 3 4 5
End
```

Bodies

Next bodies are defined using indexes of Physical Surfaces defined in the .geo file.

```
!U+
Body 1
  Target Bodies(1) = 1
  Name = "U+"
  Equation = 1
  Material = 1
End

!U-
Body 2
  Target Bodies(1) = 2
  Name = "U-"
  Equation = 1
  Material = 1
End
```

Target Bodies keyword accepts the indexes of Physical Surfaces from the mesh file. *Equation* keyword accepts indexes of the equation sections where active solvers are defined. *Material* keyword accepts indexes of the Material sections. In the above given listing two bodies are created from the Physical Surfaces of the U-phase winding assigning the air as the material and enabling all the solvers of equation section 1 on these bodies. See the definitions of the remaining bodies in the Appendix B.

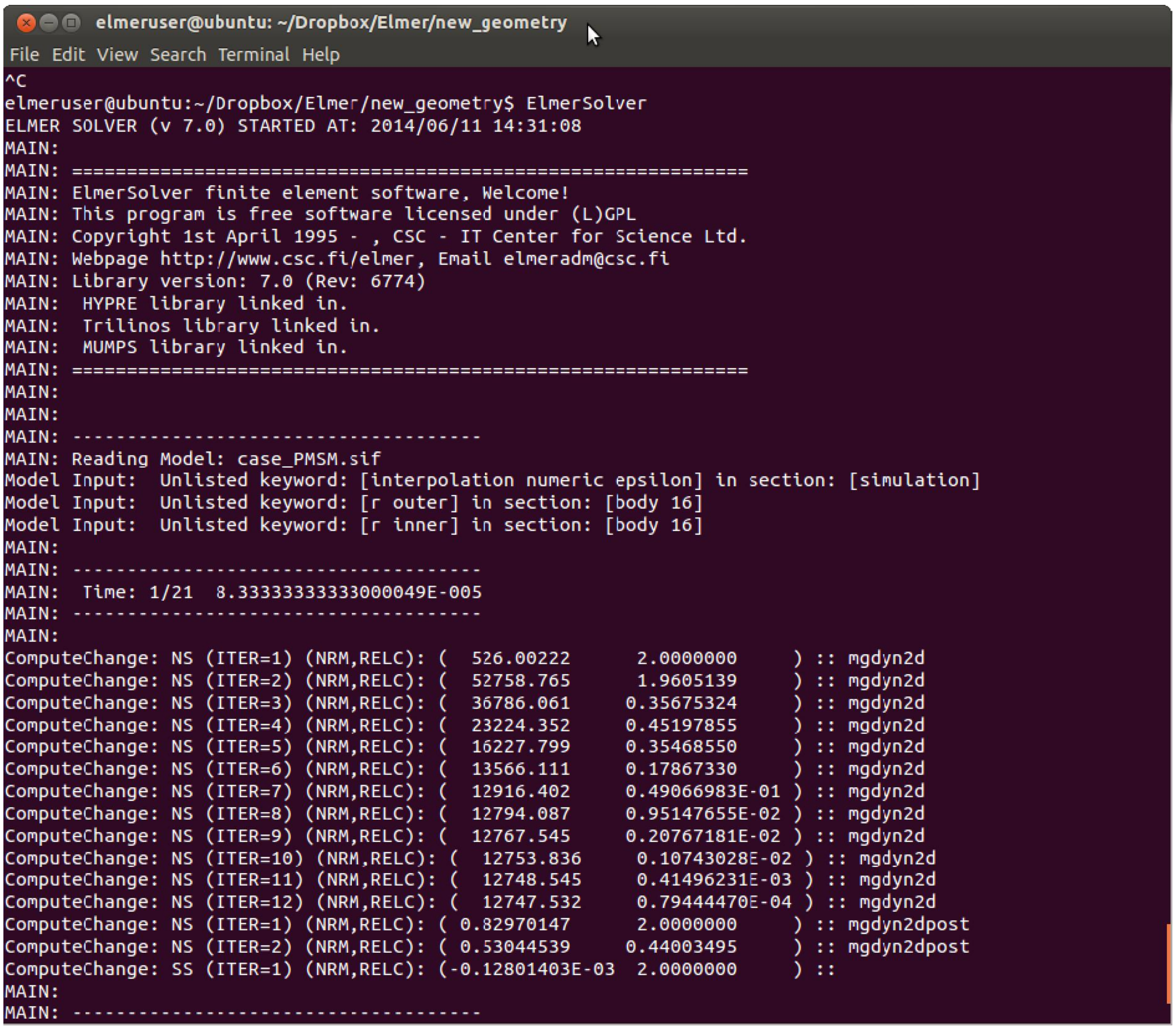
It should be noted that for all the bodies belonging to the rotor, the *Body Force* keyword is assigned with the index 1. One of the body sections should contain keywords '*r inner*' and '*r outer*' with inner and outer radii of the air gap. There should be no iron parts within the region limited by these radii. These keywords are utilized in torque calculation routines utilizing the Arkkio's equation for the torque [8].

```
Body 16
  Target Bodies(1) = 16
  Name = "rotor airgap"
  Equation = 1
  Material = 1
  Body Force = 1
  r outer = Real 0.0406
  r inner = Real 0.040
End
```

5. Solving

The solving process starts with execution of ElmerSolver in console. The first argument should be name of the SIF.

- ElmerSolver case_PMSM.sif


elmeruser@ubuntu: ~/Dropbox/Elmer/new_geometry
File Edit View Search Terminal Help

```
^C
elmeruser@ubuntu:~/Dropbox/Elmer/new_geometry$ ElmerSolver
ELMER SOLVER (v 7.0) STARTED AT: 2014/06/11 14:31:08
MAIN:
MAIN: =============================================================
MAIN: ElmerSolver finite element software, Welcome!
MAIN: This program is free software licensed under (L)GPL
MAIN: Copyright 1st April 1995 - , CSC - IT Center for Science Ltd.
MAIN: Webpage http://www.csc.fi/elmer, Email elmeradm@csc.fi
MAIN: Library version: 7.0 (Rev: 6774)
MAIN:  HYPRE library linked in.
MAIN:  Trilinos library linked in.
MAIN:  MUMPS library linked in.
MAIN: =============================================================
MAIN:
MAIN:
MAIN: -------------------------------------
MAIN: Reading Model: case_PMSM.sif
Model Input:  Unlisted keyword: [interpolation numeric epsilon] in section: [simulation]
Model Input:  Unlisted keyword: [r outer] in section: [body 16]
Model Input:  Unlisted keyword: [r inner] in section: [body 16]
MAIN:
MAIN: -------------------------------------
MAIN:  Time: 1/21  8.3333333333000049E-005
MAIN: -------------------------------------
MAIN:
ComputeChange: NS (ITER=1) (NRM,RELC): (  526.00222      2.0000000     ) :: mgdyn2d
ComputeChange: NS (ITER=2) (NRM,RELC): (  52758.765      1.9605139     ) :: mgdyn2d
ComputeChange: NS (ITER=3) (NRM,RELC): (  36786.061     0.35675324     ) :: mgdyn2d
ComputeChange: NS (ITER=4) (NRM,RELC): (  23224.352     0.45197855     ) :: mgdyn2d
ComputeChange: NS (ITER=5) (NRM,RELC): (  16227.799     0.35468550     ) :: mgdyn2d
ComputeChange: NS (ITER=6) (NRM,RELC): (  13566.111     0.17867330     ) :: mgdyn2d
ComputeChange: NS (ITER=7) (NRM,RELC): (  12916.402     0.49066983E-01 ) :: mgdyn2d
ComputeChange: NS (ITER=8) (NRM,RELC): (  12794.087     0.95147655E-02 ) :: mgdyn2d
ComputeChange: NS (ITER=9) (NRM,RELC): (  12767.545     0.20767181E-02 ) :: mgdyn2d
ComputeChange: NS (ITER=10) (NRM,RELC): (  12753.836     0.10743028E-02 ) :: mgdyn2d
ComputeChange: NS (ITER=11) (NRM,RELC): (  12748.545     0.41496231E-03 ) :: mgdyn2d
ComputeChange: NS (ITER=12) (NRM,RELC): (  12747.532     0.79444470E-04 ) :: mgdyn2d
ComputeChange: NS (ITER=1) (NRM,RELC): ( 0.82970147      2.0000000     ) :: mgdyn2dpost
ComputeChange: NS (ITER=2) (NRM,RELC): ( 0.53044539     0.44003495     ) :: mgdyn2dpost
ComputeChange: SS (ITER=1) (NRM,RELC): (-0.12801403E-03  2.0000000     ) ::
MAIN:
MAIN: -------------------------------------
```

During the solution process, messages from different solvers are printed to the console. The amount of the messages depends on the level of verbosity set in the SIF by *Max Output Level* keyword. Here one can follow the convergence of nonlinear computations. After the assessment of convergence time and accuracy of results the linear system convergence tolerances can be modified in order to speed-up the computational process or increase accuracy.

6. Post-processing

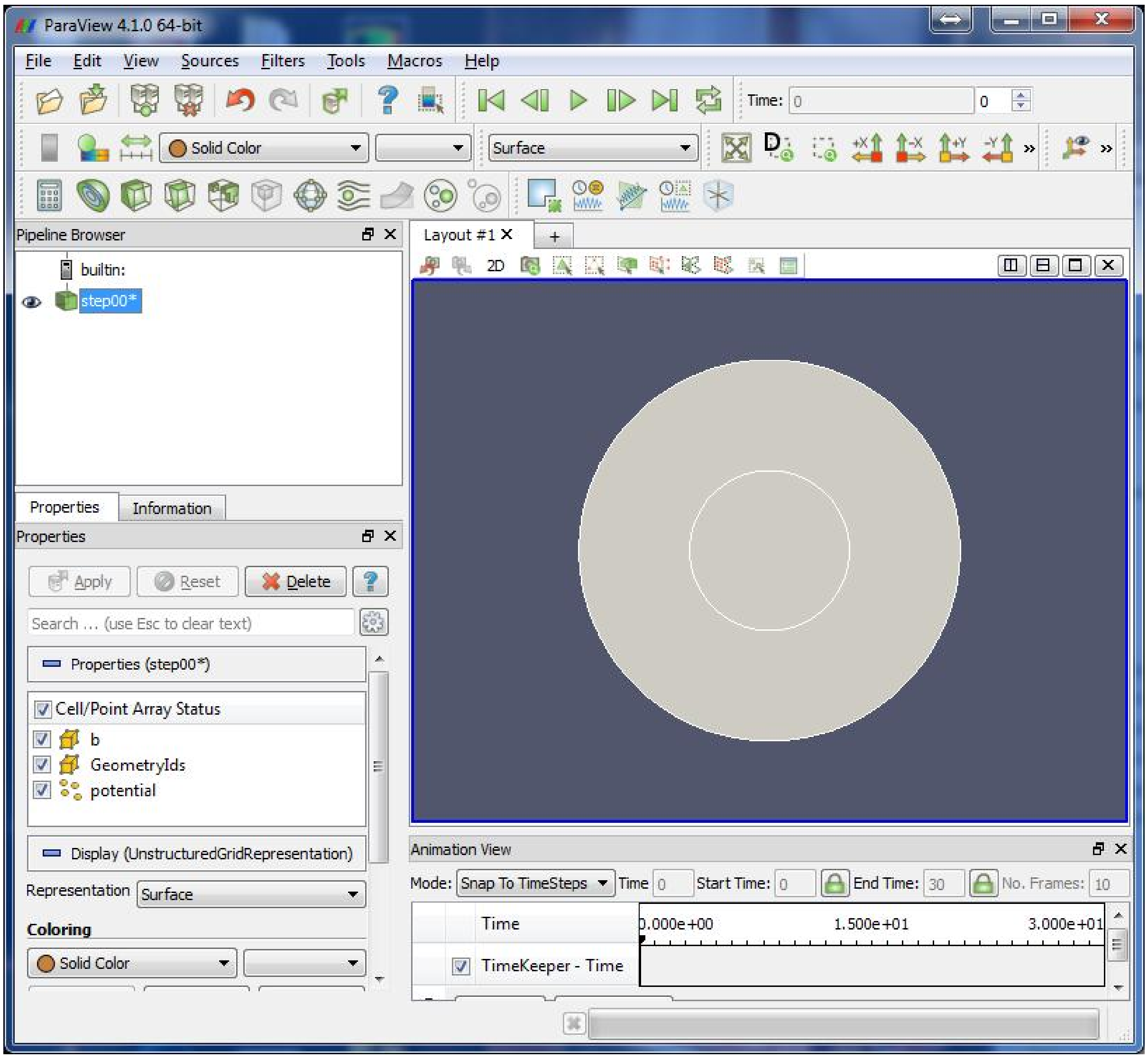
Elmer has its own legacy postprocessor *ElmerPost* which accepts result files of format *.ep*. It has somewhat limited capabilities and no longer developed further. It is recommended to use *ParaView* as a postprocessor with *Elmer*.

ParaView is an open-source cross platform application for scientific visualization. It can be run in parallel environment and process parallel data. It works on Linux, MacOS and Windows. The current version of application (v 3.14) can be very unstable and can crush unexpectedly. However, after some experience with the application the work effectiveness can be high.

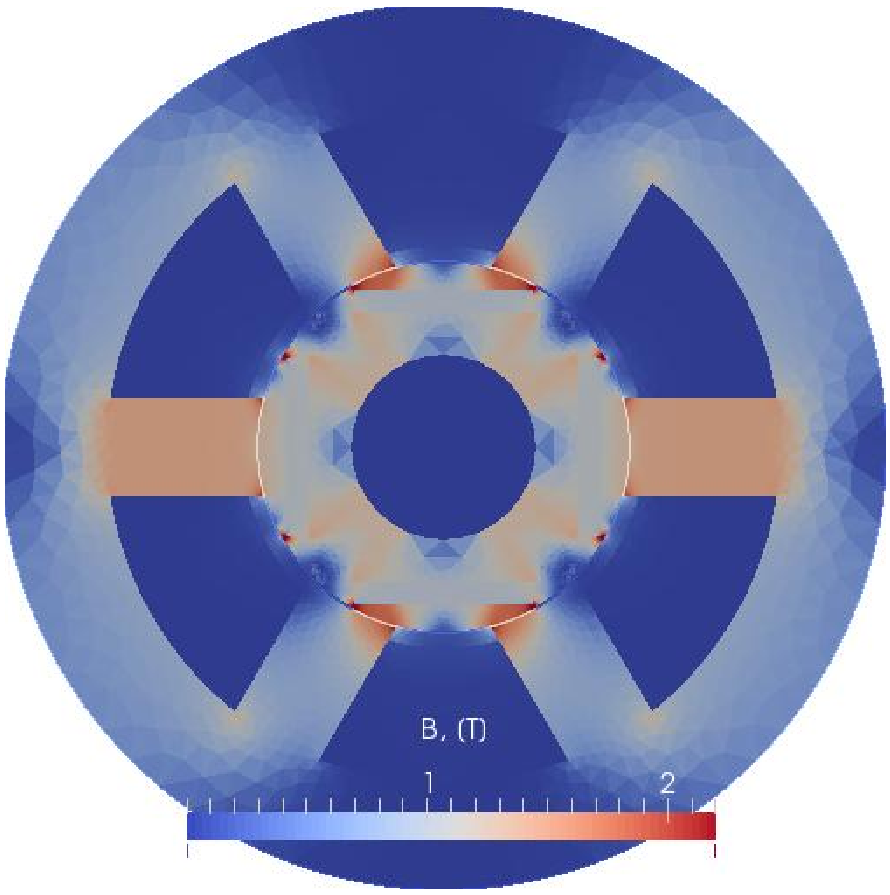
The first thing to assess the quality of electromagnetic simulations is check that equipotential flux lines and flux densities are physically possible and reasonable in the computational domain.

ParaView

Run ParaView and select File > Open > 'results' > 'step..vtu' > OK > Apply. The computational domain with highlighted boundaries will appear in the Layout window.

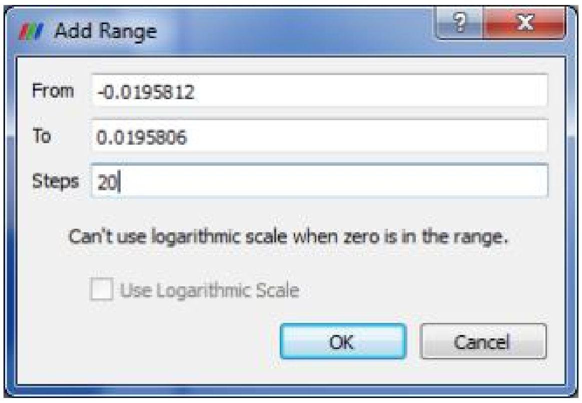


Then select in the *Properties* scroll in the left bottom side of the window Coloring >b (instead of Solid Color). This will change color of the computational domain according to the flux density.



The results of visualization can be highly customized. Refer to ParaView tutorials and user guides (e.g. at http://www.paraview.org/).

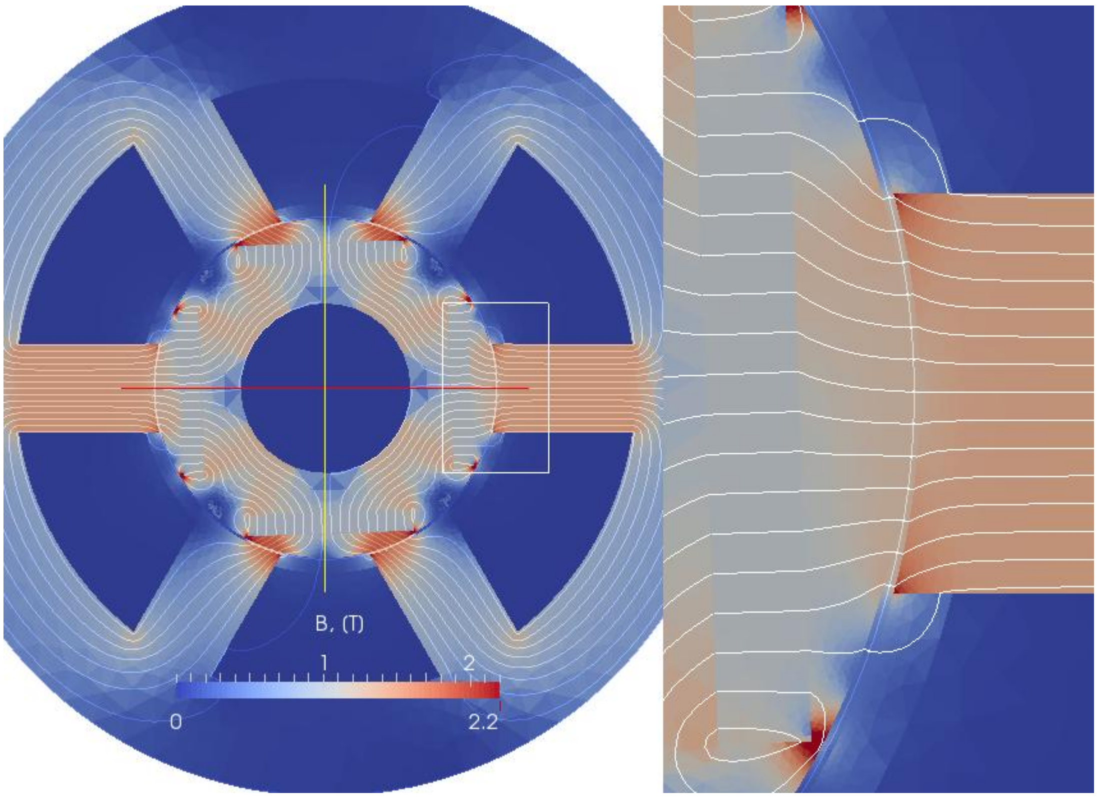
To show the equipotential flux lines select Filters > Alphabetical > Contour. In the *Properties > Isosurfaces* scroll highlight Value Range single entry and press '-' to delete it. Then press small button with a ruler to add a range of values, and in the pop-up window write number of steps 20. Press OK > Apply.



From -0.0195812
To 0.0195806
Steps 20

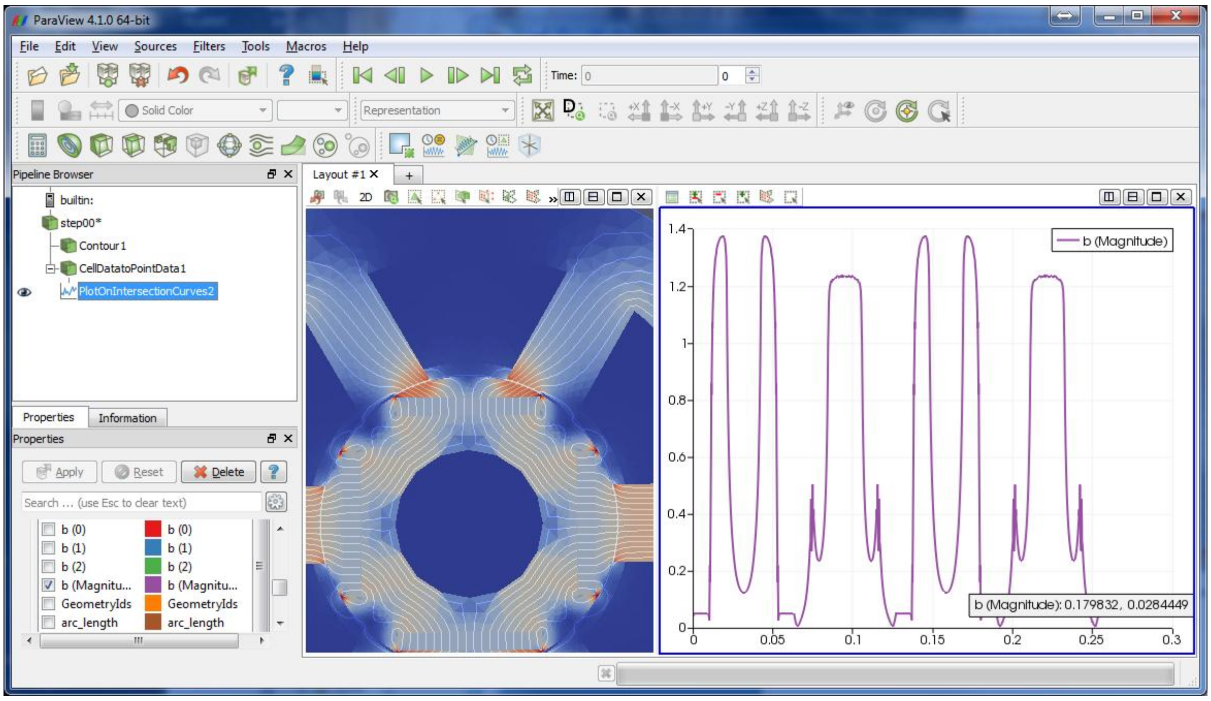
Can't use logarithmic scale when zero is in the range.

In the ParaView window the flux lines should be visible. Check that flux lines are closed. Press little eye symbol near the first filter in the *Pipeline Browser* scroll in the top left area of the window, to enable simultaneous display of flux lines and flux density in the graphical view.



Press play button in the top part of the window to verify that results at other rotor positions are reasonable. Pay attention to the magnetization direction of PMs.

The air gap flux density can be easily obtained in *ParaView*. For this purpose first apply Cell Data To Point Data filter in the pipeline, which will convert vector data to scalar data on the points. And then, apply Plot On Intersection Curves filter. Where first select Slice Type as Sphere with radius of the middle of the air gap. And then, tick in the Line Series just b (Magnitude). Same curve can be obtained by post-processing results of the "SaveData" "SaveLine" solver.



Python

ParaView is dedicated for visualization of scientific data on the mesh. But for post-processing very often visualization of scalar data is required. E.g. *ElmerSolver* saves torque values in text files in scalar form. These files are still very difficult to process with *ParaView*. Therefore some other software should be utilized for processing scalar data.

Commercial software packages such as *Matlab*, or *MS Excel* are very convenient for processing scalar data. They can be efficiently used to process and visualize scalar data. But purpose of this report is to utilize open-source and free software in the post-processing. Among the free/open-source software for post-processing several packages can be listed – Octave, Python, R.

Here, a powerful open-source free python plotting library *matplotlib* is utilized. *Matplotlib* provides similar to *Matlab* plotting environment, and can produce high-quality highly-customized 2D and 3D plots. There are additional libraries – *SciPy* and *NumPy* – that provide mathematical functions for scientific computing. To successfully perform post-processing tasks, the *python* + *matplotlib* should be installed in the working environment. These tools are available for MacOS, Linux and Windows, and they are free and open source.

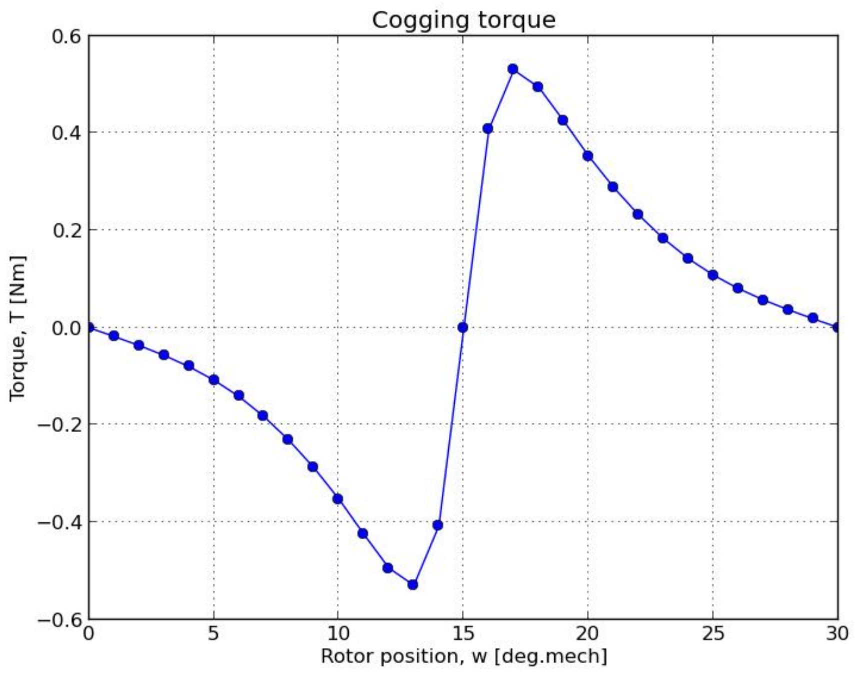
ElmerSolver saves the scalars in the text format. The file can be found in the directory specified in the *Header* section of SIF with keyword *Results Directory*. The name of the output file is specified in the section of the 'SaveData' solver with the keyword *Filename*. In the example case the output scalars are saved in the './results/scalars.dat' file. The description of saved variables is given in the file './results/scalars.dat.names'. The torque value is saved for each time step in the first column of the 'scalars.dat'.

Next a python script listing is shown. Each command has comments after '#' sign. The result of running this script in the working directory is a plot of one period of the cogging torque.

```
# file 'scalars.dat' contains scalar data calculated by solver
MagnetoDynamics2D
# and saved by SaveData.SaveScalars procedure
# file 'scalars.dat.names' contains physical names of the saved variables


# importing functions of 'pylab' and 'matplotlib' for plotting data
import pylab as pl

# importing functions of numpy
#import numpy as np
#import math
######################################################################
######################################################################

# open file for reading
infile = open('./results/scalars.dat', 'r') # path to solver scalar results
file

# initializing array for T
torque = []

# length of the stack in m
```

```
stack_len = 0.05

# processing lines of file
for line in infile:
            line = line.split() # splitting each word
            torque.append(float(line[0])*stack_len) # first word is T

# close file
infile.close()

# plotting torque curve
pl.figure()
pl.plot(torque,'o-')
pl.grid(True)
pl.title('Cogging torque')
pl.xlabel('Rotor position, w [deg.mech]')
pl.ylabel('Torque, T [Nm]')
pl.show()
```

The routine for building a plot is very similar to Matlab.



The peak-to-peak cogging torque is quite high – it is 10% from the rated torque. It depends on the application whether it is acceptable or too high. Various methods of decreasing cogging and pulsating torque can be found in the literature.

7. Back-EMF Simulation

In order to capture the back-EMF waveform the Simulation section of SIF is modified as follows.

```
Timestep Sizes = $ 1/(w_el/2/pi)/180! 180 samples per el. period
Timestep Intervals = 181
```

This gives 180 samples per electrical period, or one sample every 2 el.deg.

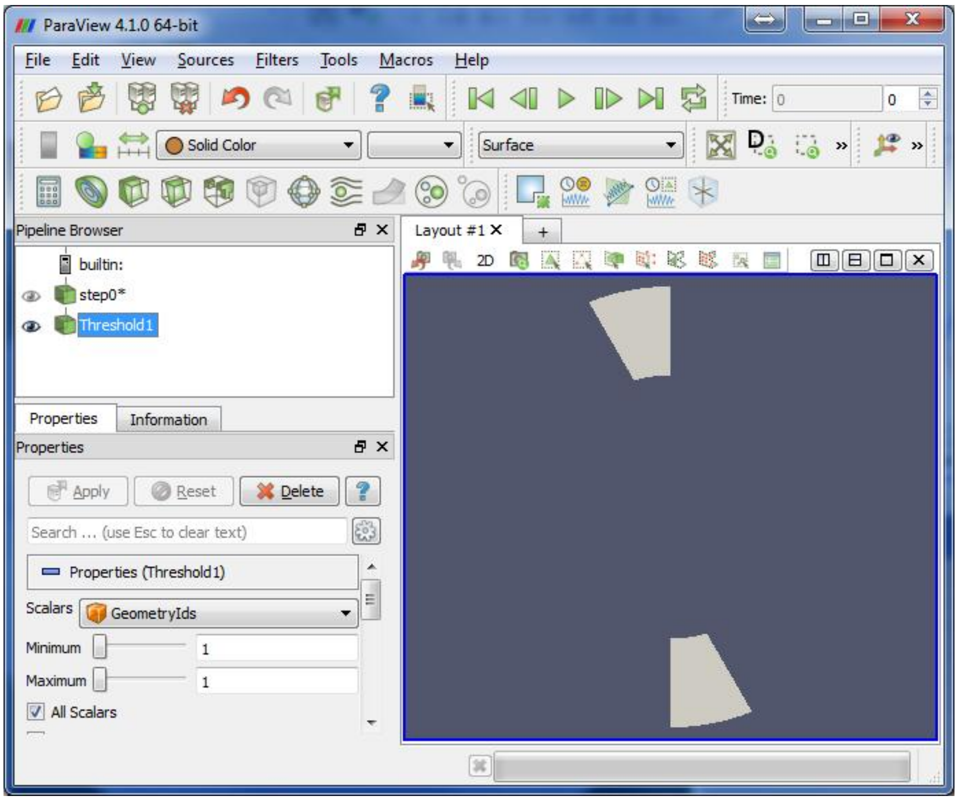
The back-EMF is calculated by differentiation of the flux linkage using finite difference method

$$U = -\frac{d\Psi}{dt} \approx -\frac{\Delta\Psi}{\Delta t}$$

The flux linkage is estimated from the magnetic vector potentials *A* as follows.

$$\Psi = \frac{\left(\oint_{S_{phase+}} A\,dS - \oint_{S_{phase-}} A\,dS\right)}{S_{phase}} \cdot N_{ph} \cdot l_{stack}$$

The estimation of the flux linkages of all phases is performed in ParaView.
Select in ParaView the output of the ElmerSolver File > Open > '../results/step..vtu' > OK > Apply. In order to integrate magnetic potentials over phase region, the phase surface must be selected. Select Filters > Alphabetical > Threshold. In the *Properties* window in the *Scalars* dropdown list select *GeometryIds*, and for *Minimum* and *Maximum* write 1 and 1. This will isolate a physical surface with index 1. Index was assigned in the .geo file. Index 1 corresponds to Uplus phase. Index 2 corresponds to Uminus phase.



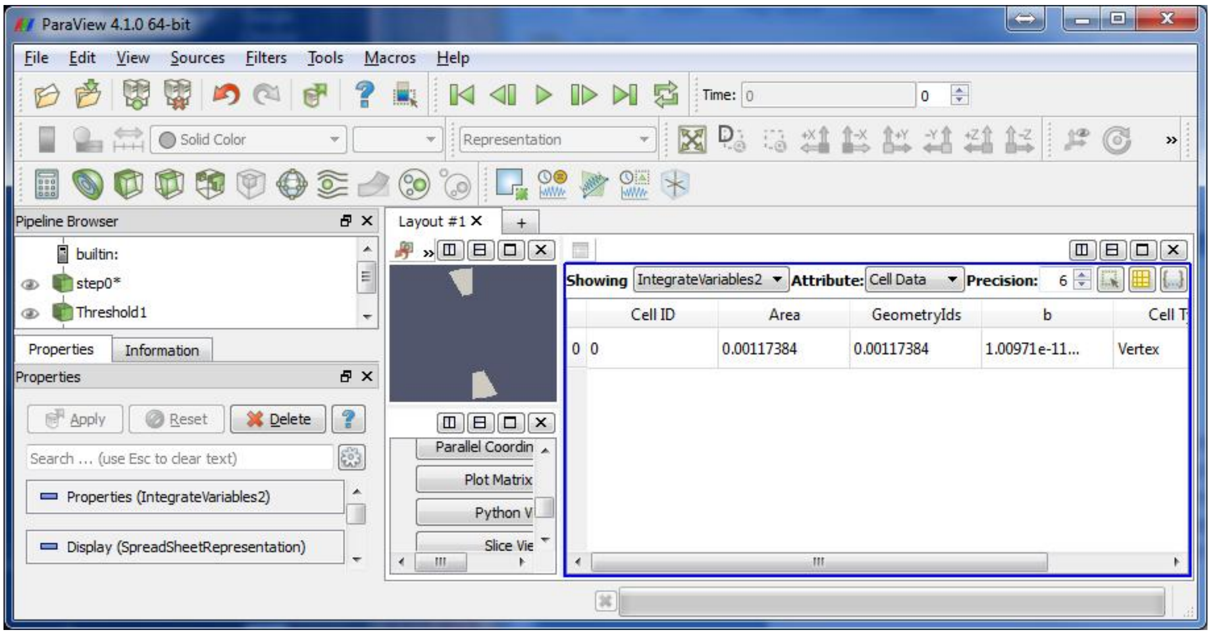
builtin:
step0*
Threshold1

Properties (Threshold1)

Scalars: GeometryIds
Minimum: 1
Maximum: 1
☑ All Scalars

The surface of U phase with positive direction of current is isolated, and now the magnetic vector potential can be integrated over this surface. Select Filters > Alphabetical > Integrated Variables > Apply. A new spreadsheet view window will open where in the Point Data attributes under variable *potential* will be the integral of the magnetic vector potential over the isolated surface (potential = -

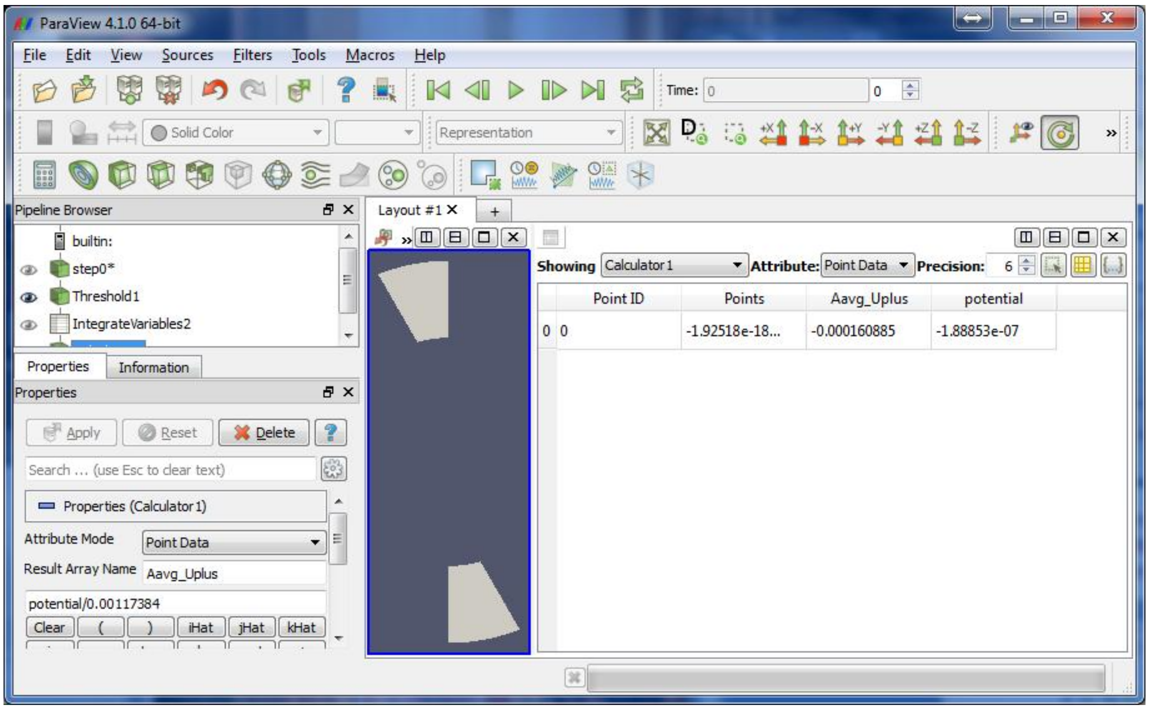
1.889e-7). Under the Cell Data attribute the area of the surface is computed as well (Area = 0.001174).



Showing IntegrateVariables2 Attribute: Cell Data Precision: 6

	Cell ID	Area	GeometryIds	b	Cell T
0	0	0.00117384	0.00117384	1.00971e-11...	Vertex

Now the average magnetic vector potential over the surface of the phase is calculated by selecting Filters > Alphabetic > Calculator and dividing the potential variable by the area of the surface. The resultant variable has name *Aavg_Uplus*.



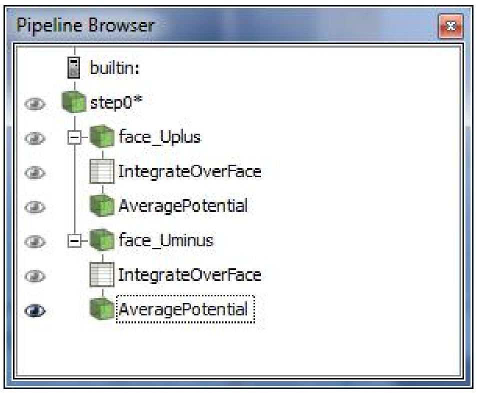
Attribute Mode: Point Data
Result Array Name: Aavg_Uplus
potential/0.00117384

Showing Calculator1 Attribute: Point Data Precision: 6

	Point ID	Points	Aavg_Uplus	potential
0	0	-1.92518e-18...	-0.000160885	-1.88853e-07

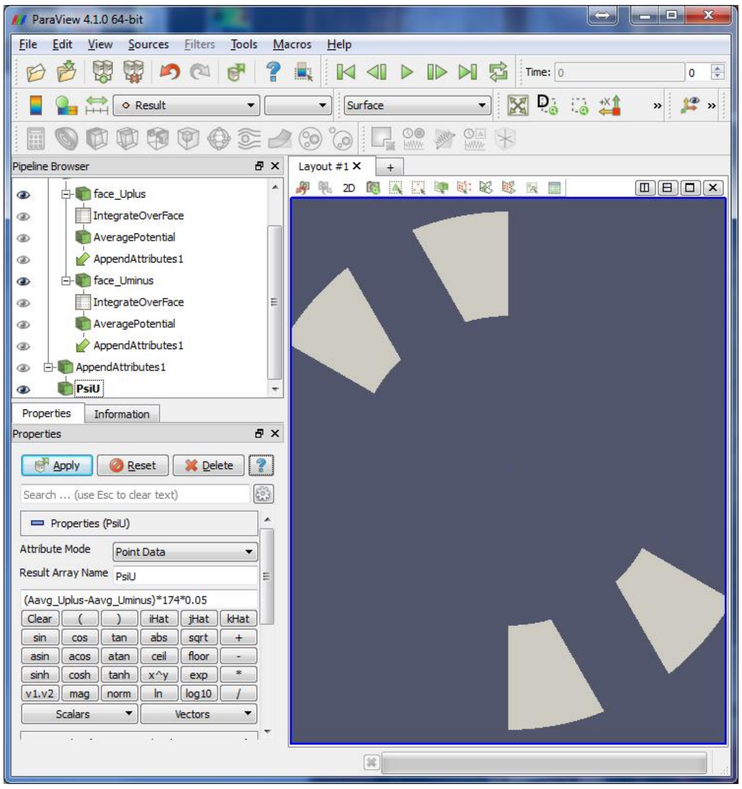
The filters in the pipeline browser can be renamed in order to have more reasonable naming.

Highlight the source filter and calculate the average magnetic vector potential over the Uminus surface with geometric id 2.



These calculations can be repeated for other phases.

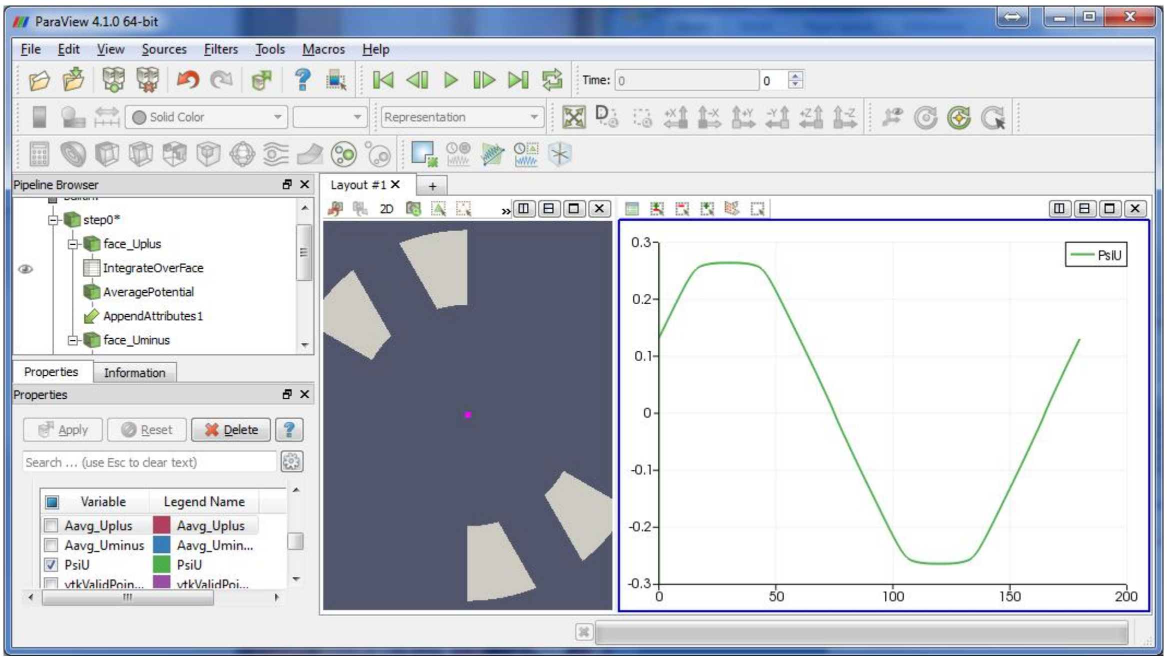
Now the pipeline is divided into two brunches. In order to use variables from these two brunches simultaneously, highlight the AveragePotential filters in both brunches and select Filters > Alphabetical > Append Attributes. In the end of the pipeline will appear a new filter. Highlight that filter and select Filters > Alphabetic > Calculator in order to calculate the flux linkage of the U phase.



Properties (PsiU)
Attribute Mode: Point Data
Result Array Name: PsiU
(Aavg_Uplus-Aavg_Uminus)*174*0.05

The Ψ_U is now calculated by (Aavg_Uplus-Aavg_Uminus)*174*0.05. On some installations of ParaView the comma is used instead of decimal point.

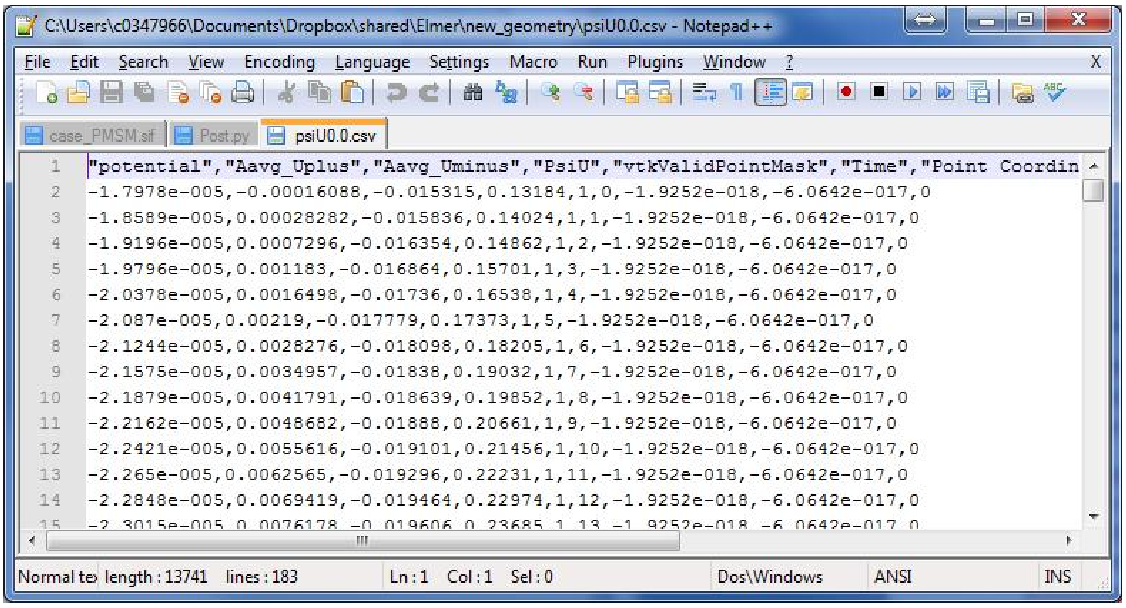
Highlight the last Calculator filter and select Filters > Alphabetic > Plot Selection Over Time. A new window with Line Chart View will appear. In the properties window in the Line Series section remove marks from all lines except the line with PsiU variable. Now the window shows the flux linkage waveform of phase U over time.



The pipeline structure with the filter settings can be saved to utilize it later in other simulations of various electrical machines. Select File > Save State in order to save the structure of the pipeline for later usage with other input data.

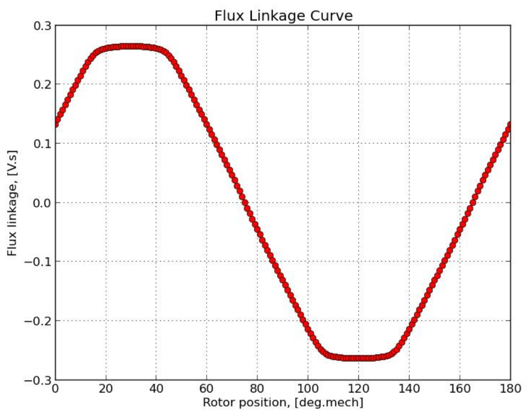
The flux linkage curve was obtained using readymade filters. All post processing including torque estimation and back-EMF estimation can be performed solely within the ParaView environment. However this would require creation of custom filters. In this study the creation of custom filters is not considered. The back-EMF waveform is obtained by post-processing outside the ParaView environment.

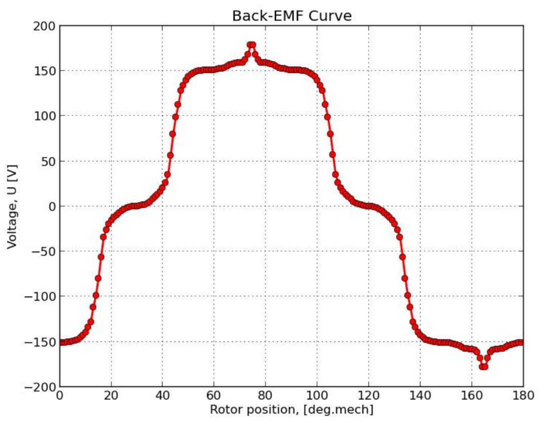
Highlight the Line View window and select File > Save Data > 'psiU.csv' > OK > Write All Time Steps > OK. All the active variables will be saved in CSV text format in file psiU0.0.csv. Open this file in a text editor and find in the first string position of the variable *PsiU*.

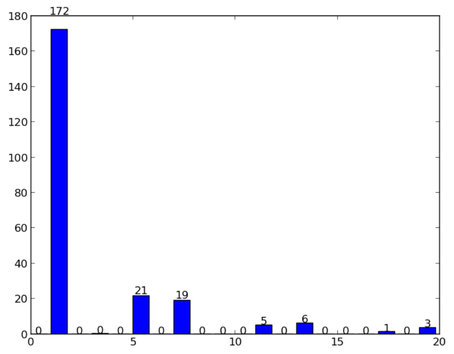

C:\Users\c0347966\Documents\Dropbox\shared\Elmer\new_geometry\psiU0.0.csv - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
case_PMSM.sif
Post.py
psiU0.0.csv
Normal te length : 13741 lines : 183 Ln : 1 Col : 1 Sel : 0 Dos\Windows ANSI INS

```
"potential","Aavg_Uplus","Aavg_Uminus","PsiU","vtkValidPointMask","Time","Point Coordin
-1.7978e-005,-0.00016088,-0.015315,0.13184,1,0,-1.9252e-018,-6.0642e-017,0
-1.8589e-005,0.00028282,-0.015836,0.14024,1,1,-1.9252e-018,-6.0642e-017,0
-1.9196e-005,0.0007296,-0.016354,0.14862,1,2,-1.9252e-018,-6.0642e-017,0
-1.9796e-005,0.001183,-0.016864,0.15701,1,3,-1.9252e-018,-6.0642e-017,0
-2.0378e-005,0.0016498,-0.01736,0.16538,1,4,-1.9252e-018,-6.0642e-017,0
-2.087e-005,0.00219,-0.017779,0.17373,1,5,-1.9252e-018,-6.0642e-017,0
-2.1244e-005,0.0028276,-0.018098,0.18205,1,6,-1.9252e-018,-6.0642e-017,0
-2.1575e-005,0.0034957,-0.01838,0.19032,1,7,-1.9252e-018,-6.0642e-017,0
-2.1879e-005,0.0041791,-0.018639,0.19852,1,8,-1.9252e-018,-6.0642e-017,0
-2.2162e-005,0.0048682,-0.01888,0.20661,1,9,-1.9252e-018,-6.0642e-017,0
-2.2421e-005,0.0055616,-0.019101,0.21456,1,10,-1.9252e-018,-6.0642e-017,0
-2.265e-005,0.0062565,-0.019296,0.22231,1,11,-1.9252e-018,-6.0642e-017,0
-2.2848e-005,0.0069419,-0.019464,0.22974,1,12,-1.9252e-018,-6.0642e-017,0
-2.3015e-005,0.0076178,-0.019606,0.23685,1,13,-1.9252e-018,-6.0642e-017,0
```

In Appendix C a listing of a python script is given. Execution of this script produces the following figures.







The back-EMF curve is far from sinusoidal. The back-EMF waveform has high harmonic content. In order to reduce the harmonic content the shape of the rotor surface should be changed. The fundamental harmonic has RMS value of 121.6 V which is a bit higher than desired analytical value of 117 V.

From these plots the phase shift of the U-phase can be determined. The maximum flux linkage in the phase is obtained when back-EMF curve crosses zero – 30 degrees mechanical, or 60 degrees electrical. This phase shift will be utilized for Clark transformation later.

8. Loaded Machine

Current density can be imposed over a physical surface by introducing additional body forces and applying them on the correspondent bodies. Using inverse Park transformation the values of the phase currents can be expressed utilizing constant steady state currents in the synchronous dq-reference frame. According to design values the RMS Id = 9.1 A and Iq = -0.9 A.

In order to simplify equations for current densities, here the initial rotor position is moved by 30 mechanical degrees to compensate for the phase shift by modification of the Body Force 1 as follows.

```
!rotation of the rotor
Body Force 1
  Name = "BodyForce_Rotation"
  Mesh Rotate 3 = Variable time, timestep size
    Real MATC "180/pi*w_m*(tx(0)-tx(1))+30"                       ! in degrees
End
```

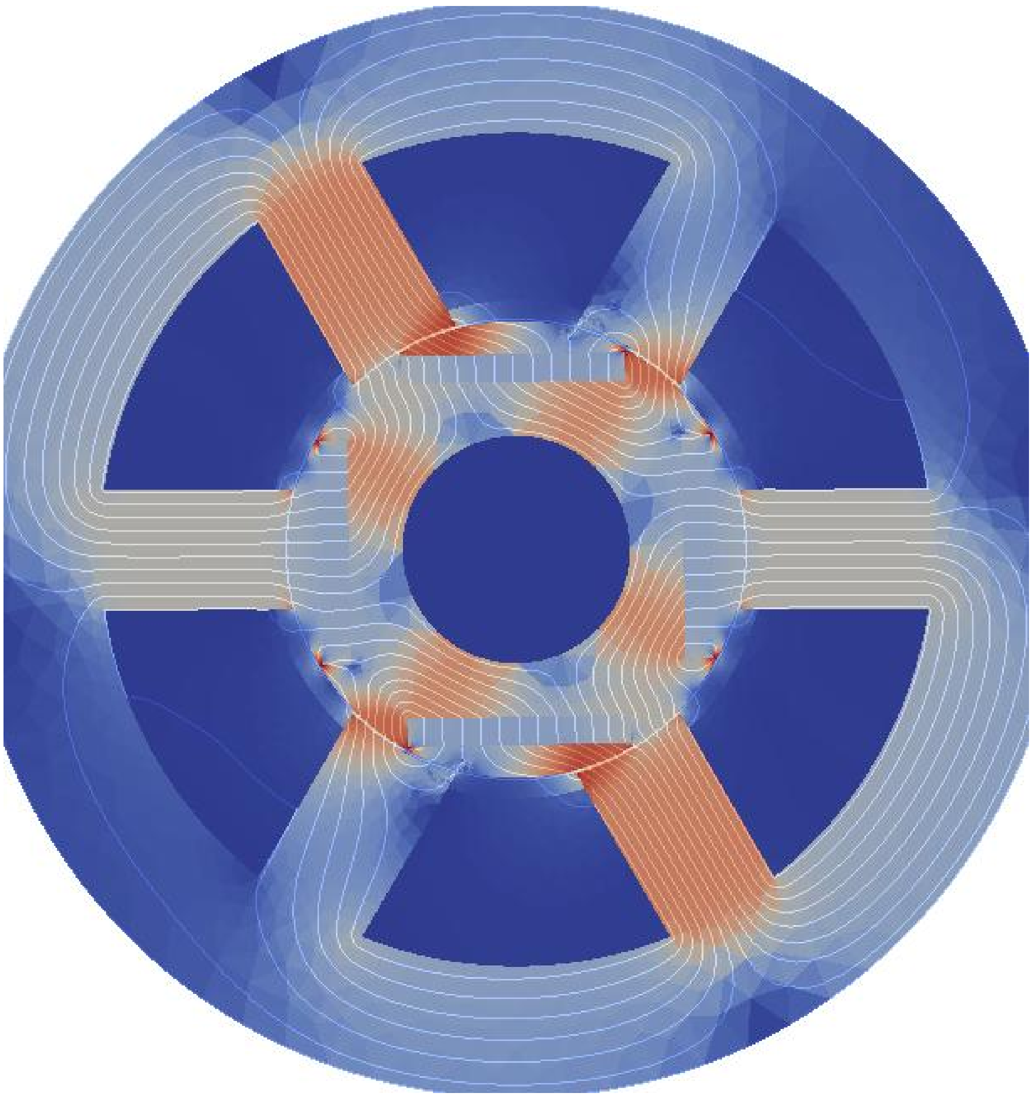
In order to get time dependency, the MATC functionality is utilized. Therefore, the U-phase current can be expressed as follows utilizing the dq- transformations.

```
!current in the phase U+
Body Force 2
  Name = "Current_U_plus"
  Current Density = Variable time, timestep size
    Real MATC "sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1)))-Iq*sin(w_el*(tx(0)-tx(1))))*Nph/Scs"
End

!current in the phase U-
Body Force 3
  Name = "Current_U_minus"
  Current Density = Variable time, timestep size
    Real MATC "-sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1)))-Iq*sin(w_el*(tx(0)-tx(1))))*Nph/Scs"
End

!U+
Body 3
  Target Bodies(1) = 3
  Name = "Body 3"
  Equation = 1
  Material = 7
  Body Force = 2
End

!U-
Body 4
  Target Bodies(1) = 4
  Name = "Body 4"
  Equation = 1
  Material = 7
  Body Force = 3
End
```

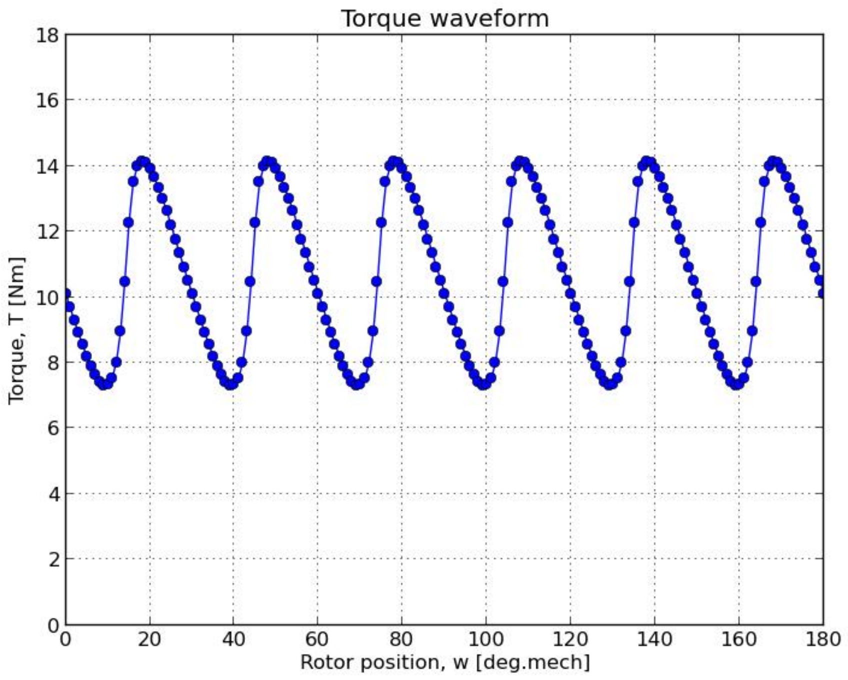
The currents in the remaining phases is defined with the phase shift of 120 electrical degrees assuming symmetrical 3-phase system.

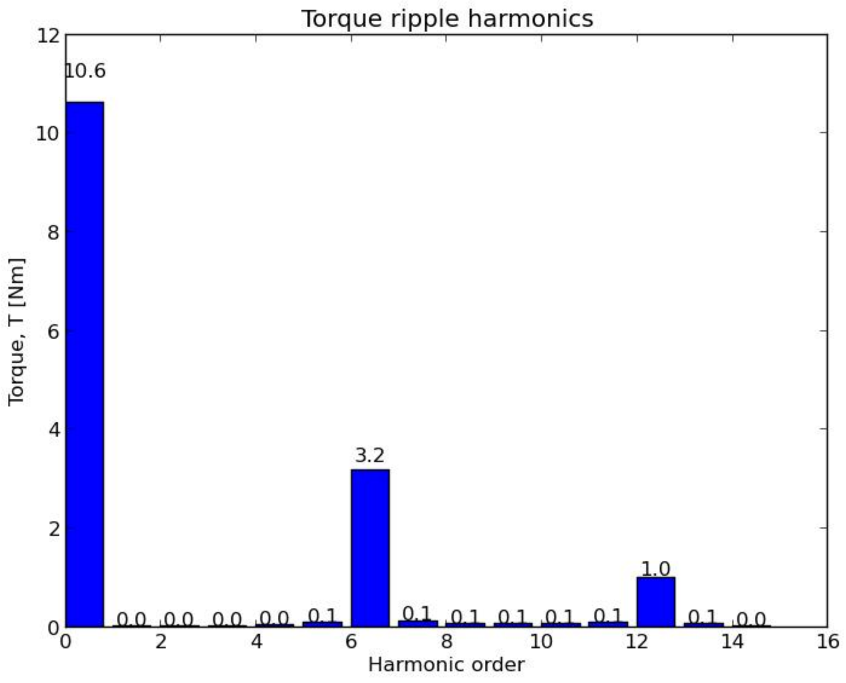
```
I_V = sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))-120/180*pi)-Iq*sin(w_el*(tx(0)-
tx(1))-120/180*pi))*Nph/Scs

I_W = sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))+120/180*pi)-Iq*sin(w_el*(tx(0)-
tx(1))+120/180*pi))*Nph/Scs
```

The following figure demonstrates the flux lines under nominal loading conditions. Compare it with a no-load view in the Section 6.



The resultant torque waveform is shown in the following figure. The waveform exhibits significant torque ripple. The torque ripple is resulted from non-sinusoidal flux density in the air gap, which resulted in non-sinusoidal back-EMF.





The average torque is a bit higher than expected (10.6 Nm instead of 9.6 Nm). However it will decrease if corrections for torque ripple reduction (such as skewing, shaping of the rotor poles, increase of the permeability of wedges) will be applied.

The SIF for current driven simulation can be found in the Appendix D.

Voltage Driven Simulation

The voltage driven simulation requires writing of separate solver to take into account the changing inductances of the machine depending on the magnetic state. Elmer manuals contain documentation and examples on how to write own solvers for Elmer. It is not performed in this tutorial.

9. Losses Calculation

The losses in electrical machines consist mainly of Joule losses (AC and DC) in the windings (copper losses), eddy-current, hysteresis and additional losses in the lamination material (iron losses) and mechanical losses.

The Elmer is an open-source software package. Therefore it is an ideal platform for developing and testing custom models for power losses in the lamination material. In current version of Elmer (v. 6790) there are no publicly available modules for estimation losses in lamination material.

In the current version of Elmer (v. 6790) the electric potential field computation is not yet implemented in 2D transient magneto dynamic model (it is available only for harmonic fields). However it is implemented in the 3D formulation. Therefore, the modeling of eddy currents in solid parts (solid conductors, bulk permanent magnets, solid iron parts) is only available in 3D. Eddy currents in PMs, armature-slot-leakage-induced eddy currents in armature slot solid conductors, and skin and proximity effects can be estimated currently only in 3D.

10. Parallel Computing

The most important feature of Elmer is its capability for massive parallelization. Elmer utilizes MPI standard for multi-process communications. This allows to run Elmer on multi-cores of a single-processor environment as well as on multi-processor environment of a computational cluster. Utilization of parallel computation allows significantly decrease computational time required to solve large 3D models. Refer to Elmer Solver Manual section on parallel runs to find more information.

The parallel runs can be achieved on platforms which support MPI. In this paper the parallelization is demonstrated using Linux with MPI on a 4-core CPU. Elmer parallel runs can be utilized in a powerful massive parallel cluster environment as well. For Finnish academic users CSC provides its computing resources for scientific computations. They have installed Elmer with MPI support on some of their servers. Refer to CSC website and ”CSC Computing environment user guide” for more information.

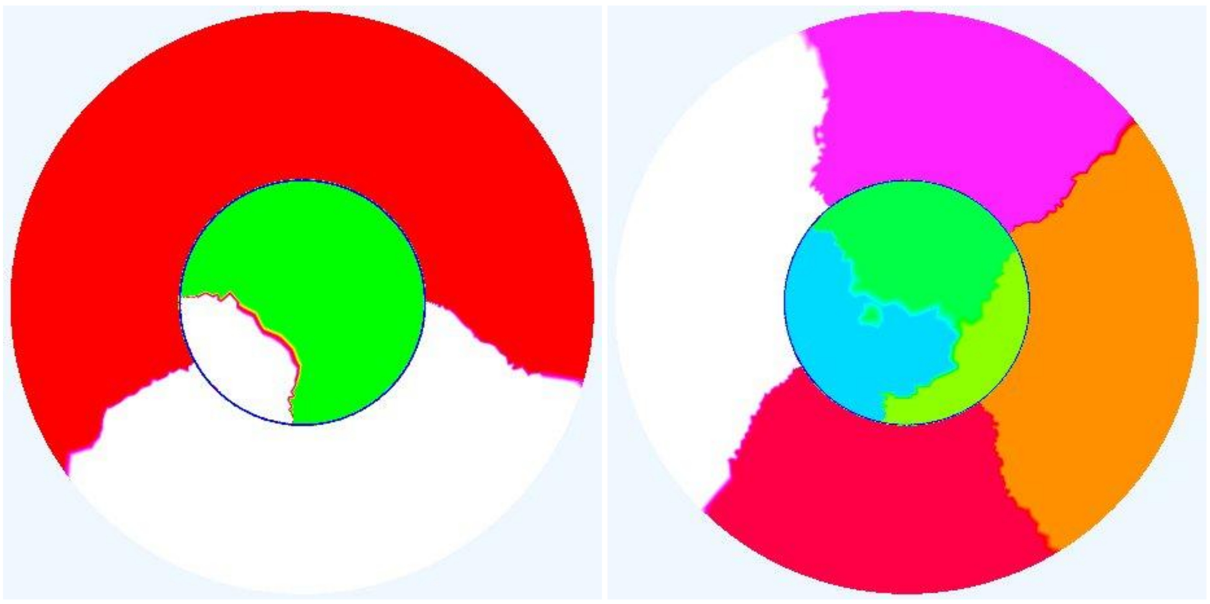
Preprocessing for Parallel Computations

Parallel computation requires mesh partitioning to distribute the computational domain among the parallel processes. The partitions from existent mesh can be split using *ElmerGrid*. ElmerGrid has lots of parameters to tune the mesh conversion and partitioning [3]. In this section only parameters utilized in this specific case are described.

The mesh partitions are obtained by execution of the following command:

```
ElmerGrid 14 2 PMSM.msh -metis 4 3 -connect 2 -partdual
```

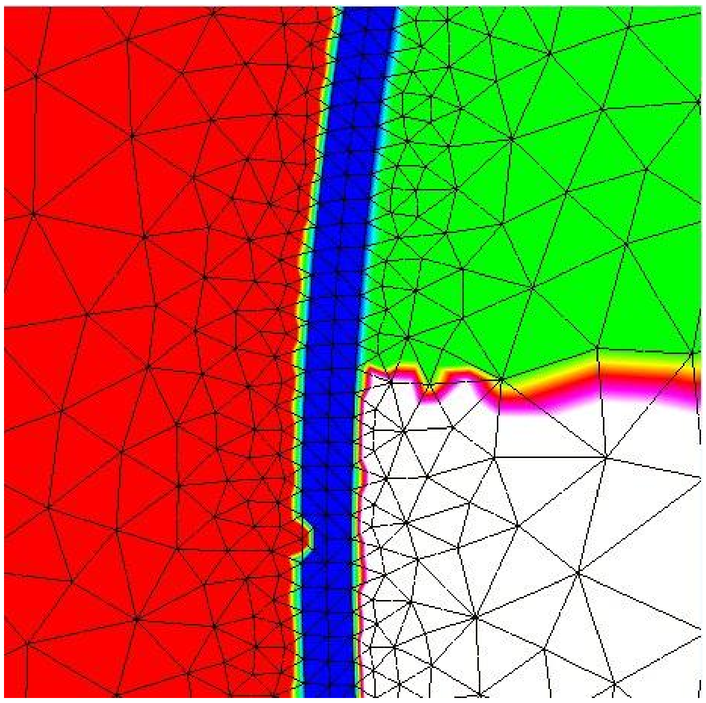
First parameter indicates the input format, where *14* corresponds to GMSH. Second parameter sets the output format, where *2* corresponds to ElmerSolver mesh file format. Then goes the input file name. After that option *–metis* commands ElmerGrid to utilize Metis library for partitioning. First argument for Metis is number of partitions – *4*. The following figure demonstrates result of partitioning on 4 and 8 partitions. The coloring is according to the partition number.



For complex geometry of typical electrical machine it is perhaps not worth to pursue very high number of partitions as the inter-process communication during solving process will consume a lot of computational resources and time. Perhaps for 2D electrical machine model the highest feasible number of partitions is 16.

Second Metis parameter relates to the partitioning algorithm of Metis – *3*.

-partdual and *–connect 2* parameters are required for sliding surface. This enforces the sliding boundary with index 2 to belong to only one partition. The following figure demonstrates that sliding boundary in the center of the air gap belongs to only one blue partition.



After execution of this command in the directory correspondent to input filename a new directory with name *`partitioning.4`* is created with 4 partition meshes in ElmerSolver format.

Solving Process

The parallel run of ElmerSolver requires some modifications to SIF. The solution methods should support the parallelization. Most of iterative methods have natural support for parallel computations. The convergence time of iterative methods is usually somewhat lower in parallel than in series, however this is compensated by time savings due to the parallelization.

For low complexity problems (read 2D, number of nodes is <500000, number of partitions < 16) capabilities of MUMPS direct solver can be effectively utilized in parallel. MUMPS solver is not a part of Elmer, therefore special arrangements during installation (compilation) should be performed in order to link MUMPS and ElmerSolver. Refer to Elmer Manuals and www.elmerfer.org/forum for more information. In the SIF change direct solvers from UMFPACK to MUMPS, or to iterative solvers.

In order to run ElmerSolver in parallel, the platform should support MPI and Elmer itself should be compiled with MPI support. In this paper the parallel runs are performed using Ubuntu virtual machine with Elmer MPI installation. Elmer is tested and supported to run in parallel in Linux environment. However, it is possible to compile Elmer with MPI support for Windows and MacOS as well. For more information read Elmer Manuals and www.elmerfer.org/forum. Currently, most parallel runs of Elmer are done on Linux.

Execution command is

```
mpirun –np 4 ElmerSolver_mpi
```

The console output is the same as for serial run except that in the beginning the information on number of parallel processes can be found.

```
elmeruser@ubuntu:~/model$ mpirun -np 4 ElmerSolver_mpi
ELMER SOLVER (v 7.0) STARTED AT: 2014/07/17 17:13:30
ELMER SOLVER (v 7.0) STARTED AT: 2014/07/17 17:13:30
ELMER SOLVER (v 7.0) STARTED AT: 2014/07/17 17:13:30
ELMER SOLVER (v 7.0) STARTED AT: 2014/07/17 17:13:30
ParCommInit:  Initialize #PEs:            4
MAIN:
MAIN: =============================================================
MAIN: ElmerSolver finite element software, Welcome!
MAIN: This program is free software licensed under (L)GPL
MAIN: Copyright 1st April 1995 - , CSC - IT Center for Science Ltd.
MAIN: Webpage http://www.csc.fi/elmer, Email elmeradm@csc.fi
MAIN: Library version: 7.0 (Rev: 6814)
MAIN:  Running in parallel using 4 tasks.
MAIN:  HYPRE library linked in.
MAIN:  Trilinos library linked in.
MAIN:  MUMPS library linked in.
MAIN: =============================================================
```

The parallel run of the current driven simulation test case on Ubuntu Virtual machine (which was running on a Windows host machine using VMWare Player) finished within 1219 s. The results in serial and in parallel were identical. Serial run in the same environment required 1551 s. In native installation of Linux the execution time difference should be perhaps more considerable.

Conclusion

In this tutorial an example workflow for electrical machine designers is demonstrated. 2D FEM model was built and analyzed using only open source and free software. Main parameters of designed PMSM (back EMF waveform, torque waveform) were estimated using Elmer FEM software package and visualized using ParaView and Python post-processing routines. Some advanced features such as parallelization of FEM computations were also demonstrated.

Even though some advanced features (eddy current estimation in solids in 2D, skin effect, iron losses models, readymade routines for voltage driven simulations) were not yet implemented from the box, the Elmer software is ready to be effectively utilized for the purpose of electrical machine design. Moreover, its open source code can be a powerful and effective workbench for testing own models of iron losses in the steel and lamination materials and homogenization models for the lamination stacks and copper windings – especially as all required examples of writing own code for Elmer are available in the Elmer documentation.

The parallelization capabilities of Elmer allow for effective utilization of processing capabilities of modern PCs and computational clusters in electrical machine design and optimization (especially for large 3D models).

Multi-physics capabilities of Elmer can be utilized for conjugated modeling of electro-magnetic, mechanical and thermal phenomena in electrical machines. These capabilities can be utilized also in development of complex multi-physics models of materials.

Appendix A

Listing of the `PMSM.geo` input script file for GMSH.

```
// Dimensions of the machine
DefineConstant[ R_rot_in = { 20}];
DefineConstant[ R_rot_out = { 40}];
DefineConstant[ R_stat_out = { 96}];
DefineConstant[ Gap = { 0.6}];
DefineConstant[ b_d = { 21}];
DefineConstant[ hss_tot = { 32.4}];
DefineConstant[ bridge = { 0.6}];
DefineConstant[ w_PM = { 39}];
DefineConstant[ h_PM = { 4.9}];
DefineConstant[ h_wedge = { 3}];

// Mesh density points
DefineConstant[ mesh_gap = { Gap/2}];
DefineConstant[ mesh_fine = { 0.5}];
DefineConstant[ mesh_normal = { 3}];
DefineConstant[ mesh_coarse = { 10}];

//===================================================================
// Geometry definition

Point(1) = {0, 0, 0, mesh_coarse};

// ==========================================================
// =============== Stator Geometry ===============================
// ==========================================================

//Tooth bottom
Point(2) = {Gap+R_rot_out, 0, 0, mesh_gap};
Rotate {{0, 0, 1}, {0, 0, 0}, Asin(b_d/2/(Gap+R_rot_out))} {
  Duplicata { Point{2}; }
}
Rotate {{0, 0, 1}, {0, 0, 0}, -Asin(b_d/2/(Gap+R_rot_out))} {
  Duplicata { Point{2}; }
}

//Tooth top
Point(5) = {Gap+R_rot_out+hss_tot, 0, 0, mesh_normal};
Rotate {{0, 0, 1}, {0, 0, 0}, Asin(b_d/2/(Gap+R_rot_out+hss_tot))} {
  Duplicata { Point{5}; }
}
Rotate {{0, 0, 1}, {0, 0, 0}, -Asin(b_d/2/(Gap+R_rot_out+hss_tot))} {
  Duplicata { Point{5}; }
}

Point(8) = {Gap+R_rot_out+h_wedge, 0, 0, mesh_normal};
Rotate {{0, 0, 1}, {0, 0, 0}, Asin(b_d/2/(Gap+R_rot_out+h_wedge))} {
  Duplicata { Point{8}; }
}
Rotate {{0, 0, 1}, {0, 0, 0}, -Asin(b_d/2/(Gap+R_rot_out+h_wedge))} {
  Duplicata { Point{8}; }
}

//Sector edges
Rotate {{0, 0, 1}, {0, 0, 0}, Pi/6} {
  Duplicata { Point{2, 5, 8}; }
}
Rotate {{0, 0, 1}, {0, 0, 0}, -Pi/6} {
  Duplicata { Point{2, 5, 8}; }
}
Line(1) = {12, 13};
Line(2) = {13, 11};
Line(3) = {6, 9};
Line(4) = {9, 3};
Line(5) = {7, 10};
Line(6) = {10, 4};
Line(7) = {15, 16};
Line(8) = {16, 14};
Circle(9) = {15, 1, 7};
Circle(10) = {16, 1, 10};
Circle(11) = {14, 1, 4};
Circle(12) = {4, 1, 2};
Circle(13) = {2, 1, 3};
Circle(14) = {3, 1, 11};
Circle(15) = {9, 1, 13};
Circle(16) = {6, 1, 12};

//Finalize stator yoke segment
Point(17) = {R_stat_out, 0, 0, mesh_coarse};

Rotate {{0, 0, 1}, {0, 0, 0}, -Pi/6} {
  Duplicata { Point{17}; }
}
Rotate {{0, 0, 1}, {0, 0, 0}, Pi/6} {
  Duplicata { Point{17}; }
}

Line(17) = {19, 12};
Line(18) = {18, 15};
Circle(19) = {18, 1, 17};
Circle(20) = {17, 1, 19};

//Define plane surface for the  steel segment

Line Loop(21) = {19, 20, 17, -16, 3, 4, -13, -12, -6, -5, -9, -18};
Plane Surface(22) = {21};

//Define plane surfaces of the coil (plus and minus of the phase coil) and wedges
Line Loop(23) = {16, 1, -15, -3};
Plane Surface(24) = {23};
Line Loop(25) = {5, -10, -7, 9};
Plane Surface(26) = {25};
Line Loop(27) = {15, 2, -14, -4};
Plane Surface(28) = {27};
Line Loop(29) = {10, 6, -11, -8};
Plane Surface(30) = {29};

//Duplicate stator sectors
For it In {1:5}
               Rotate {{0, 0, 1}, {0, 0, 0}, it*Pi/3} {
                Duplicata { Surface{22, 24, 28, 30, 26}; }
               }
EndFor

// ==========================================================
// =============== Rotor geometry ===============================
// ==========================================================

Point(379) = {R_rot_in, 0, 0, mesh_coarse};
Point(380) = {R_rot_out, 0, 0, mesh_gap};

//Rotor sector borders
Rotate {{0, 0, 1}, {0, 0, 0}, Pi/4} {
  Duplicata { Point{379, 380}; }
```

```
}
Rotate {{0, 0, 1}, {0, 0, 0}, -Pi/4} {
  Duplicata { Point{379, 380}; }
}

Line(176) = {382, 381};
Line(177) = {383, 384};
Circle(178) = {383, 1, 379};
Circle(179) = {379, 1, 381};
Circle(180) = {384, 1, 380};
Circle(181) = {380, 1, 382};

//Magnet points
Point(385) = {(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge))), 0, 0, mesh_normal};
Point(386) = {(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge))), w_PM/2, 0, mesh_gap};
Point(387) = {(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge))), -w_PM/2, 0, mesh_gap};
Point(388) = {(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))-h_PM, 0, 0, mesh_normal};
Point(389) = {(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))-h_PM, w_PM/2, 0,

mesh_normal};
Point(390) = {(R_rot_out-bridge)*Cos(Asin(w_PM/2/(R_rot_out-bridge)))-h_PM, -w_PM/2, 0,

mesh_normal};

//PM edges
Line(182) = {389, 386};
Line(183) = {386, 385};
Line(184) = {385, 387};
Line(185) = {387, 390};
Line(186) = {390, 388};
Line(187) = {388, 389};

//PM surface
Line Loop(188) = {184, 185, 186, 187, 182, 183};
Plane Surface(189) = {188};

//Pole sector iron
Line Loop(190) = {181, 176, -179, -178, 177, 180};
Plane Surface(191) = {190, 188};

//Duplicate rotor poles
For it In {1:3}
               Rotate {{0, 0, 1}, {0, 0, 0}, it*Pi/2} {
                 Duplicata { Surface{191, 189}; }
               }
EndFor

//shaft plane surface
Line Loop(234) = {196, -179, -178, 223, 224, 209, 210, 195};
Plane Surface(235) = {234};

// ==========================================================
// =============== Air gap ===============================
// ==========================================================

//sliding surface
Point(529) = {Gap/2+R_rot_out, 0, 0, mesh_gap};
Point(530) = {0, Gap/2+R_rot_out, 0, mesh_gap};
Point(531) = {0, -Gap/2-R_rot_out, 0, mesh_gap};
Point(532) = {-Gap/2-R_rot_out, 0, 0, mesh_gap};

Circle(236) = {529, 1, 530};
Circle(237) = {530, 1, 532};
Circle(238) = {532, 1, 531};
Circle(239) = {531, 1, 529};
```

```
//create air gap part of the stator
Line Loop(240) = {13, 14, -57, -39, -38, -52, -86, -68, -67, -81, -115, -97, -96, -110, -144,

-126, -125, -139, -173, -155, -154, -168, 11, 12};
Line Loop(241) = {239, 236, 237, 238};
Plane Surface(242) = {240, 241};


//create air gap part of the rotor
Line Loop(243) = {181, 198, 193, 212, 207, 226, 221, 180};
Plane Surface(244) = {241, 243};

// =========================================================
// =============== Bodies ===============================
// =========================================================

// u+-
Physical Surface(1) = {88, 175};
Physical Surface(2) = {73, 160};

// v+-
Physical Surface(3) = {26, 117};
Physical Surface(4) = {24, 102};

// w+-
Physical Surface(5) = {59, 146};
Physical Surface(6) = {44, 131};

//Stator iron
Physical Surface(7) = {60, 31, 22, 147, 118, 89};

//Rotor iron
Physical Surface(8) = {206, 192, 191, 220};

//Wedges
Physical Surface(9) = {28, 54, 49, 83, 78, 112, 107, 141, 136, 170, 165, 30};

//shaft
Physical Surface(10) = {235};

//PMs
Physical Surface(11) = {189};
Physical Surface(12) = {205};
Physical Surface(13) = {219};
Physical Surface(14) = {233};

//stator air gap
Physical Surface(15) = {242};

//rotor air gap
Physical Surface(16) = {244};


// =========================================================
// =============== Boundaries ===============================
// =========================================================

//outer boundary
Physical Line(1) = {32, 33, 61, 62, 90, 91, 119, 120, 148, 149, 19, 20};

//sliding boundary
Physical Line(2) = {239, 236, 237, 238};
```

Appendix B

SIF listing for cogging torque simulation:

```
! PMSM 2D Magneto-Static Simulation
! Parameters
$ w_m = 3000/60*2*pi                         ! [rad/s mech.] mechanical frequency
$ pp = 2                                     ! number of polepairs
$ w_el = w_m*pp                              ! [rad/s el.] electrical frequency
$ B_PM = 1.17                                ! [T] remanent flux density
$ mu_PM = 1.06                ! relative permeability of PMs
$ H_PM = B_PM/(mu_PM*pi*4d-7)                ! [A/m] magnetization of PMs
$ Id = 0                                     ! [A] d-axis current
$ Iq = 0                                     ! [A] q-axis current
$ Nph = 186                                  ! Number of phase turns
$ Scs = 0.00154555                           ! [m^2] area of the coil side



Header
  CHECK KEYWORDS Warn
  Mesh DB "PMSM" "."
  Include Path ""
  Results Directory "results"
End

Constants
   Permittivity of Vacuum = 8.8542e-12
End

Simulation
  Max Output Level = 3
  Coordinate System = Cartesian
  Coordinate Scaling = 0.001
  Simulation Type = Transient
  Timestepping Method = BDF
  BDF Order = 2
  Timestep Sizes = $ 1/(w_el/2/pi)/6/30! 180 samples per el. period
  Timestep Intervals = 31
  Output Intervals = 1
!!!!!!!! mesh interpolation tolerances
  Interpolation Numeric Epsilon = Real 5.0e-9
End



!!!!!!!!!!!!!!!!!!!!!!!!!!  Materials !!!!!!!!!!!!!!!!!!!!!!!!!!!

Material 1
  Name = "Air"
  Relative Permeability = 1
End

Material 2
  Name = "Iron"
! actually B-H values are in the file BH (name is H-B, but values are in B-H format)
  H-B Curve = Variable coupled iter
    Real Monotone Cubic
      Include BH
    End
End
```

```
Material 3
  Name = "PM_right"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "H_PM*cos(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "H_PM*sin(w_m*(tx(0)-tx(1)))"
End

Material 4
  Name = "PM_up"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "-H_PM*sin(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "H_PM*cos(w_m*(tx(0)-tx(1)))"
End

Material 5
  Name = "PM_left"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "-H_PM*cos(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "-H_PM*sin(w_m*(tx(0)-tx(1)))"
End

Material 6
  Name = "PM_down"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "H_PM*sin(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "-H_PM*cos(w_m*(tx(0)-tx(1)))"
End

Material 7
  Name = "Wedge"
  Relative Permeability = 1
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Boundary conditions  !!!!!!!!!!!!!!!!!!!!!!!!!!!

!outer boundary Dirichlet
Boundary Condition 1
  Target Boundaries(1) = 1
  Name = "Boundary_Outer"
  Potential Condition = 0
  Potential = 0
End

Boundary Condition 2
   Target Boundaries(1) = 2
   Name = "Sliding"
   Discontinuous Boundary = Logical True
   Save Line = True
   Mortar BC = 3
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Body Forces  !!!!!!!!!!!!!!!!!!!!!!!!!!!

!rotation of the rotor
Body Force 1
  Name = "BodyForce_Rotation"
  Mesh Rotate 3 = Variable time, timestep size
```

```
    Real MATC "180/pi*w_m*(tx(0)-tx(1))"     ! in degrees
End


!!!!!!!!!!!!!!!!!!!!!!!!!  Solver parameters !!!!!!!!!!!!!!!!!!!!!!!!!!

!mesh rotation
Solver 1
  Exec Solver = Before Timestep
  Equation = MeshDeform
  Procedure = "RigidMeshMapper" "RigidMeshMapper"
End


! solver for magnetic vector potential A [Vs/m]
Solver 2
  Equation = MgDyn2D
  Procedure = "MagnetoDynamics2D" "MagnetoDynamics2D"
  Exec Solver = Always

  Nonlinear System Convergence Tolerance = 1.0e-6
  Nonlinear System Max Iterations = 20
  Nonlinear System Relaxation Factor = 1

  Linear System Solver = Direct
  Linear System Direct Method = UMFPACK
End


!solver for magnetic flux density B [T]
!does not require settings for nonlinear solver
Solver 3
  Equation = MgDyn2DPost
  Procedure = "MagnetoDynamics2D" "BSolver"
  Exec Solver = After Timestep

  Linear System Solver = Direct
  Linear System Direct Method = UMFPACK

  Discontinuous Galerkin = True
  Average Within Materials = False
End


!save to VTU format for ParaView with faces' IDs
Solver 4
  Exec Solver = After Timestep
  Procedure = "ResultOutputSolve" "ResultOutputSolver"
  Output File Name = "step"
  Vtu Format = True
  Binary Output = True
  Single Precision = True
  Save Geometry Ids = True
End


!save scalar variables - torque T [Nm]
Solver 5
  Exec Solver = After Timestep
  Filename = "scalars.dat"
  Procedure = "SaveData" "SaveScalars"
  Show Norm Index = 1
End

Equation 1
  Name = "ModelDomain"
  Active Solvers(5) = 1 2 3 4 5
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Bodies are here !!!!!!!!!!!!!!!!!!!!!!!!!!!

Body 1
  Target Bodies(1) = 1
  Name = "U+"
  Equation = 1
  Material = 1
End

Body 2
  Target Bodies(1) = 2
  Name = "U-"
  Equation = 1
  Material = 1
End

Body 3
  Target Bodies(1) = 3
  Name = "V+"
  Equation = 1
  Material = 1
End

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

Body 5
  Target Bodies(1) = 5
  Name = "W+"
  Equation = 1
  Material = 1
End

Body 6
  Target Bodies(1) = 6
  Name = "W-"
  Equation = 1
  Material = 1
End

Body 7
  Target Bodies(1) = 7
  Name = "stator lamination"
  Equation = 1
  Material = 2
End

Body 8
  Target Bodies(1) = 8
  Name = "rotor lamination"
  Equation = 1
  Material = 2
  Body Force = 1
End

Body 9
  Target Bodies(1) = 9
  Name = "wedges"
  Equation = 1
  Material = 7
End

Body 10
  Target Bodies(1) = 10
  Name = "shaft"
  Equation = 1
  Material = 1
  Body Force = 1
End

Body 11
  Target Bodies(1) = 11
  Name = "PM1"
  Equation = 1
  Material = 3
  Body Force = 1
End

Body 12
  Target Bodies(1) = 12
  Name = "PM2"
  Equation = 1
  Material = 6
  Body Force = 1
End

Body 13
  Target Bodies(1) = 13
  Name = "PM3"
  Equation = 1
  Material = 5
  Body Force = 1
End

Body 14
  Target Bodies(1) = 14
  Name = "PM4"
  Equation = 1
  Material = 4
  Body Force = 1
End

Body 15
  Target Bodies(1) = 15
  Name = "stator airgap"
  Equation = 1
  Material = 1
End

Body 16
  Target Bodies(1) = 16
  Name = "rotor airgap"
  Equation = 1
  Material = 1
  Body Force = 1
  r outer = Real 0.0406
  r inner = Real 0.040
End
```

Appendix C

Python script listing for back-EMF curve:

```
# importing functions of 'pylab' and 'matplotlib' for plotting data
import pylab as pl

# importing functions of numpy
import numpy as np

# reading flux linkage of a phase
infile = open('./psiU0.0.csv', 'r')

# initializing array for flux linkage
psiU = []

for line in infile:
               line = line.split(',')
               psiU.append(line[3]) # fourth word is Psi

# close file
infile.close()

# converting string array to float array
psiU = np.array(psiU[1:], float)

# drawing flux linkage curve
pl.figure()
pl.plot(psiU,'o-r')
pl.grid(True)
pl.title('Flux Linkage Curve')
pl.xlabel('Rotor position, [deg.mech]')
pl.ylabel('Flux linkage, [V.s]')
#pl.show()

# calculating voltage using finite difference method
w_el = 3000./60.*2.*np.pi*2.
U = -(psiU[1:]-psiU[:-1])/(1./(w_el/2./np.pi)/180.)

# plotting the voltage waveform
pl.figure()
pl.plot(U,'o-r', linewidth = 2.0)
pl.grid(True)
pl.title('Back-EMF Curve')
pl.xlabel('Rotor position, [deg.mech]')
pl.ylabel('Voltage, U [V]')

# calculating the harmonics content
harm_U = np.abs( np.fft.fft(U) / len(U) )*2

# plotting harmonics
pl.figure()
spec_U = pl.bar(range(len(harm_U[:20])),harm_U[:20])

# adding labels to the bar chart
def autolabel(rects):
    for rect in rects:
        h = rect.get_height()
        pl.text(rect.get_x()+rect.get_width()/2., 1.05*h, '%d'%int(h),
                ha='center', va='bottom')

autolabel(spec_U)

pl.show()
```

Appendix D

SIF listing for current driven simulation:

```
! PMSM 2D Magneto-Static Simulation
! Parameters
$ w_m = 3000/60*2*pi                         ! [rad/s mech.] mechanical frequency
$ pp = 2                                     ! number of polepairs
$ w_el = w_m*pp                              ! [rad/s el.] electrical frequency
$ B_PM = 1.17                                ! [T] remanent flux density
$ mu_PM = 1.06               ! relative permeability of PMs
$ H_PM = B_PM/(mu_PM*pi*4d-7)                ! [A/m] magnetization of PMs
$ Id = -0.87                                 ! [A] d-axis current
$ Iq = 9.04                                  ! [A] q-axis current
$ Nph = 186                                  ! Number of phase turns
$ Scs = 0.001173                             ! [m^2] area of the coil side
$ sigma_PM = 1/(1.4d6)                       ! [1/(Ohm*m)] electric conductivity of PMs

Header
  CHECK KEYWORDS Warn
  Mesh DB "PMSM" "."
  Include Path ""
  Results Directory "results"
End

Simulation
  Max Output Level = 3
  Coordinate System = Cartesian
  Coordinate Scaling = 0.001
  Simulation Type = Transient
  Timestepping Method = BDF
  BDF Order = 2
  Timestep Sizes = $ 1/(w_el/2/pi)/180! 180 samples per el. period
  Timestep Intervals = 181
  Output Intervals = 1
!!!!!!!! mesh interpolation tolerances
  Interpolation Numeric Epsilon = Real 5.0e-7
!  Mesh Levels = 1
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Materials !!!!!!!!!!!!!!!!!!!!!!!!!!

Material 1
  Name = "Air"
  Relative Permeability = 1
End

Material 2
  Name = "Iron"
! actually B-H values are in the file BH (name is H-B, but values are in B-H format)
  H-B Curve = Variable coupled iter
    Real Monotone Cubic
      Include BH
    End
End

Material 3
  Name = "PM_right"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "H_PM*cos(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "H_PM*sin(w_m*(tx(0)-tx(1)))"
  Electric Conductivity = $ sigma_PM
End

Material 4
  Name = "PM_up"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "-H_PM*sin(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "H_PM*cos(w_m*(tx(0)-tx(1)))"
  Electric Conductivity = $ sigma_PM
End

Material 5
  Name = "PM_left"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "-H_PM*cos(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "-H_PM*sin(w_m*(tx(0)-tx(1)))"
  Electric Conductivity = $ sigma_PM
End

Material 6
  Name = "PM_down"
  Relative Permeability = $ mu_PM
  Magnetization 1 = Variable time, timestep size
    Real MATC "H_PM*sin(w_m*(tx(0)-tx(1)))"
  Magnetization 2 = Variable time, timestep size
    Real MATC "-H_PM*cos(w_m*(tx(0)-tx(1)))"
  Electric Conductivity = $ sigma_PM
End

Material 7
  Name = "Wedge"
  Relative Permeability = 3
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Boundary conditions  !!!!!!!!!!!!!!!!!!!!!!!!!!!

!outer boundary Dirichlet
Boundary Condition 1
  Target Boundaries(1) = 1
  Name = "Boundary_Outer"
  Potential Condition = 0
  Potential = 0
End

Boundary Condition 2
   Target Boundaries(1) = 2
   Name = "Sliding"
   Discontinuous Boundary = Logical True
   Save Line = True
   Mortar BC = 3
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Body Forces  !!!!!!!!!!!!!!!!!!!!!!!!!!!

!rotation of the rotor
Body Force 1
  Name = "BodyForce_Rotation"
  Mesh Rotate 3 = Variable time, timestep size
    Real MATC "180/pi*w_m*(tx(0)-tx(1))"    ! in degrees
End
```

```
!current in the phase U+
Body Force 2
  Name = "Current_U_plus"
  Current Density = Variable time, timestep size
    Real MATC "sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))-60/180*pi)-Iq*sin(w_el*(tx(0)-tx(1))-
60/180*pi))*Nph/Scs"
End

!current in the phase U-
Body Force 3
  Name = "Current_U_minus"
  Current Density = Variable time, timestep size
    Real MATC "-sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))-60/180*pi)-Iq*sin(w_el*(tx(0)-tx(1))-
60/180*pi))*Nph/Scs"
End

!current in the phase V+
Body Force 4
  Name = "Current_V_plus"
  Current Density = Variable time, timestep size
    Real MATC "sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))-120/180*pi-60/180*pi)-Iq*sin(w_el*(tx(0)-tx(1))-
120/180*pi-60/180*pi))*Nph/Scs"
End

!current in the phase V-
Body Force 5
  Name = "Current_V_minus"
  Current Density = Variable time, timestep size
    Real MATC "-sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))-120/180*pi-60/180*pi)-Iq*sin(w_el*(tx(0)-tx(1))-
120/180*pi-60/180*pi))*Nph/Scs"
End

!current in the phase W+
Body Force 6
  Name = "Current_W_plus"
  Current Density = Variable time, timestep size
    Real MATC "sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))+120/180*pi-60/180*pi)-Iq*sin(w_el*(tx(0)-
tx(1))+120/180*pi-60/180*pi))*Nph/Scs"
End

!current in the phase W-
Body Force 7
  Name = "Current_W_minus"
  Current Density = Variable time, timestep size
    Real MATC "-sqrt(2)*(Id*cos(w_el*(tx(0)-tx(1))+120/180*pi-60/180*pi)-Iq*sin(w_el*(tx(0)-
tx(1))+120/180*pi-60/180*pi))*Nph/Scs"
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Solver parameters !!!!!!!!!!!!!!!!!!!!!!!!!!

!mesh rotation
Solver 1
  Exec Solver = Before Timestep
  Equation = MeshDeform
  Procedure = "RigidMeshMapper" "RigidMeshMapper"
End


! solver for magnetic vector potential A [Vs/m]
Solver 2
!  Element = p:2
  Equation = MgDyn2D
  Procedure = "MagnetoDynamics2D" "MagnetoDynamics2D"
  Exec Solver = Always
  Nonlinear System Convergence Tolerance = 1.0e-4
  Nonlinear System Max Iterations = 30
  Nonlinear System Relaxation Factor = 1

  Linear System Solver = Direct
  Linear System Direct Method = UMFPACK!!!! for serial
!  Linear System Direct Method = Mumps !!!! for parallel
End

!solver for magnetic flux density B [T]
!does not require settings for nonlinear solver
Solver 3
!  Element = p:2
  Equation = MgDyn2DPost
  Procedure = "MagnetoDynamics2D" "BSolver"
  Exec Solver = After Timestep

  Linear System Solver = Direct
  Linear System Direct Method = UMFPACK!!!! for serial
!  Linear System Direct Method = Mumps !!!! for parallel

  Discontinuous Galerkin = True
  Average Within Materials = True
End

!save to VTU format for ParaView with faces' IDs
Solver 4
  Exec Solver = After Timestep
  Procedure = "ResultOutputSolve" "ResultOutputSolver"
  Output File Name = "step"
  Vtu Format = True
  Binary Output = True
  Single Precision = True
  Save Geometry Ids = True
End

!save scalar variables - torque T [Nm]
Solver 5
  Exec Solver = After Timestep
  Filename = "scalars.dat"
  Procedure = "SaveData" "SaveScalars"
  Show Norm Index = 1
End

Solver 6
  Exec Solver = After All
  Equation = SaveLine
  Filename = "line.dat"
  Procedure = "SaveData" "SaveLine"
End

Equation 1
  Name = "ModelDomain"
  Active Solvers(6) = 1 2 3 4 5 6
End

!!!!!!!!!!!!!!!!!!!!!!!!!!  Bodies are here !!!!!!!!!!!!!!!!!!!!!!!!!!

Body 1
  Target Bodies(1) = 1
  Name = "U+"
  Equation = 1
  Material = 1
  Body Force = 2
End
```

```
Body 2
  Target Bodies(1) = 2
  Name = "U-"
  Equation = 1
  Material = 1
  Body Force = 3
End

Body 3
  Target Bodies(1) = 3
  Name = "V+"
  Equation = 1
  Material = 1
  Body Force = 4
End

Body 4
  Target Bodies(1) = 4
  Name = "V-"
  Equation = 1
  Material = 1
  Body Force = 5
End

Body 5
  Target Bodies(1) = 5
  Name = "W+"
  Equation = 1
  Material = 1
  Body Force = 6
End

Body 6
  Target Bodies(1) = 6
  Name = "W-"
  Equation = 1
  Material = 1
  Body Force = 7
End

Body 7
  Target Bodies(1) = 7
  Name = "stator lamination"
  Equation = 1
  Material = 2
End

Body 8
  Target Bodies(1) = 8
  Name = "rotor lamination"
  Equation = 1
  Material = 2
  Body Force = 1
End

Body 9
  Target Bodies(1) = 9
  Name = "wedges"
  Equation = 1
  Material = 7
End

Body 10
  Target Bodies(1) = 10
  Name = "shaft"
  Equation = 1
  Material = 1
  Body Force = 1
End

Body 11
  Target Bodies(1) = 11
  Name = "PM1"
  Equation = 1
  Material = 3
  Body Force = 1
End

Body 12
  Target Bodies(1) = 12
  Name = "PM2"
  Equation = 1
  Material = 6
  Body Force = 1
End

Body 13
  Target Bodies(1) = 13
  Name = "PM3"
  Equation = 1
  Material = 5
  Body Force = 1
End

Body 14
  Target Bodies(1) = 14
  Name = "PM4"
  Equation = 1
  Material = 4
  Body Force = 1
End

Body 15
  Target Bodies(1) = 15
  Name = "stator airgap"
  Equation = 1
  Material = 1
End

Body 16
  Target Bodies(1) = 16
  Name = "rotor airgap"
  Equation = 1
  Material = 1
  Body Force = 1
  r outer = Real 0.0406
  r inner = Real 0.040
End
```

References

1. J. Pyrhönen, T. Jokinen, V. Hrabovcova. ”Design of Rotating Electrical Machines”
2. P. Ponomarev. “Tooth-Coil Permanent Magnet Synchronous Machine Design for Special Applications”
3. ElmerGrid Manual. http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerGridManual.pdf
4. Overview of Elmer. http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerOverview.pdf
5. ElmerSolver Manual. http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerSolverManual.pdf
6. Elmer Models Manual. http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerModelsManual.pdf
7. MATC Manual. ftp://ftp.funet.fi/index/elmer/doc/MATCManual.pdf
8. A. Arkkio, ”Analysis of Induction Motors Based on the Numerical Solution of the Magnetic Field and Circuit Equations”