Step-by-step guide for the modeling of a simple geometry and solving for its electric field with CAELinux

1 Introduction

What is CAELinux?

CAELinux is a Linux distribution intended to provide a fully functional and preconfigured CAE workstation based on Open Source CAD/FEA/CFD softwares. It exists now in two versions: a liveDVD version that can also be installed on hard disk and a Virtual Machine version (VMWare) that is intended to be used from a guest operation system (for example: Windows). CAELinux and most of the included packages are distributed under GPL licence and thus are free to use, even for commercial applications.

You can get it from http://www.caelinux.com

The fastest way getting into it without problems is booting from the LiveDVD. So we assume that you burned the ISO to a DVD and booted it, so that you now see the login screen. Type in “caelinux” as user name and password. (You can install it later from the DVD, which is recommended)

2 Creating a model with Salomé

- Start Salome_Meca by going to the menu “CAE software” which can be accessed through the “PC”-symbol in the lower left corner of the desktop.
- Hint: Salome has separate modules for modeling, meshing, post processing, etc.
- Activate the geometry module by choosing it from the drop down menu or by clicking on the corresponding symbol.
- Then click on New Entity → Primitives → Torus
- Now you can choose if you want to create the torus at a previously entered point with a given direction by a previously entered vector. But because we have entered neither one of that, just choose to create the torus at the center of the coordinate system and put in some sensible values for radius 1 and 2 or just click ok. (in the programs SI units are used, so what you are entering here is always in meters)
- Now you can see it as an object in the object browser on the left. Click on it and then right click on the wire frame model, choose Display mode → Shading.
Now we want to create a box around the torus. For that we need two points to define it. So choose New Entity → Basics → Point to create a point and choose some sensible values for the first corner of the box. Do the same for the second one.

Then click on New Entity → Primitives → Box, click on the arrow after “Point 1” in order to select the starting point for the box. Select your first point from the object browser, do the same for the second point and click ok.

Now choose Operations → Boolean → Cut. Then select the box as main object from the object browser and the torus as tool object. Ok.

Now that we have our simple geometry, we need the faces for definition of the boundary conditions. Right click on the cut object and select Shape Type → Faces, then click Select All. Every face has a number. When you click on a number a face is highlighted. Select the number which highlights the torus, remove it, name the group “walls” and click ok. Now do the same for the torus, remove all the wall faces and name it “torus”.

Now smile! You have done the geometry!

3 Meshing the model

Save the model before proceeding! Save it best to your standard working directory /home/caelinux.

Activate the meshing module.

Click on Mesh → Create mesh

Choose your geometry (the cut) in the object browser.

Use Netgen 1D-2D-3D as algorithm and set up the parameters by clicking on the symbol next to the Hypothesis drop down box: The most important is the “Max. Size” option. The value depends on the dimensions of your geometry. For this case we don't need a huge mesh, so put in a value for which the calculation of the mesh takes only some seconds. A good guess would be the smaller radius of the torus.

Compute the mesh by right clicking on the mesh object and choosing Compute.

If everything worked well you see the mesh now in the viewer. If not, you can also have a look on one of the tutorials available by clicking on CAElinux Docs → GettingStarted.html on the desktop. I recommend the OpenFOAM tutorial in this case.

Tip: By right clicking on the mesh in the viewer and choosing Clipping you can have a look inside the mesh to check if really everything worked correctly.

Make a break and smile!

Now the groups defined in the geometry module have to be applied to the
mesh: Right click on the mesh in the object browser and select *Create group.*
- Select *Face* from the *Elements Type*
- Select *Group on geometry* from the *Group type* menu.
- Click on the arrow, select *Direct geometry selection* and select the “walls”
group from your cut geometry object.
- Then click ok and do the same for the torus.
- Now we have to export the mesh for the solver: Right click on the mesh in the
object browser and select *Export to UNV file*. Save it also to /home/caelinux.
- Close Salome and save the file.

4 *Solving the model with OpenFOAM*

- Ok, now it becomes a bit tricky, because you have to add the calculation of the
electric field strength to the electrostatic solver module of OpenFOAM on your
own. The funny thing with OpenFOAM is that it mainly is intended for
simulating dynamic processes. So even for static problems you have to
simulate some time steps in order to get a stable solution and we only want to
calculate a static electric field in this guide.
- Open Konqueror by clicking on “Home” and go to
/home/caelinux/OpenFOAM/OpenFOAM-1.4.1/applications/solvers/electromagn
etics/electrostaticFoam (put it directly into the “Location” address line).
- Add the following code to createFields.H:

```c++
Info<< "Calculating field E\n" << endl;
volVectorField E
(
     IOobject
     (
         "E",
         runTime.timeName(),
         mesh,
         IOobject::NO_READ,
         IOobject::AUTO_WRITE
     ),
     -fvc::grad(phi)
);
```
Info<< "Calculating field magE\n" << endl;
volScalarField magE
(
    IOobject
    (
        "magE",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mag(fvc::grad(phi))
);

- Hint: At the end of the file has to be an empty line!
- Then add the following to electrostaticFoam.C:
    after the line
    rhoFlux = -k*mesh.magSf()*fvc::snGrad(phi);
    add

        E = -fvc::grad(phi);
        magE = mag(fvc::grad(phi));

    dimensionedScalar energy = 0.5*epsilon0*sum(mesh.V()*magE*magE);

    Info << "\n Energy of field: " << energy.value() << "\n" << endl;

- Some explanations to the added code: With E you have the electric field
  strength as a vector and with magE the magnitude of this vector. Besides this
  the total energy is calculated and outputted to every time step in the log file.
  More to that later.
- Now start the “CAE Console” similar to starting Salome_Meca but just clicking a
  bit above it.
- Type “cd