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 seperate 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 C*ompute*.
- 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/electromagnetics/electrostaticFoam* (put it directly into the “Location” address line).
- Add the following code to createFields.H:

```
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*

/home/caelinux/OpenFOAM/OpenFOAM-1.4.1/applications/solvers/electromagnetics/electrostaticFoam” in. (“cd” is the command for changing directory)

- Then type “wmake”.
- Wait until everything is compiled and check for errors. If everything is correct compiled, we can now start with the actual solving.
- Type “*cd $FOAM_RUN*” in the console. With that you change to the directory where youre OpenFOAM cases are stored, which is: */home/caelinux/OpenFOAM/caelinux-1.4.1/run*
- Then start OpenFOAM by typing *“FoamX&”.*
- Double Click on *Hosts* → *localhost* → *.*
- The point is the directory, where your cases are stored. Right click on the point and create a new case. As class choose electrostaticFoam and name it after the .unv-file you have exported with Salome.
- Then type the following into the console: “*ideasUnvToFoam . xyz /home/caelinux/xyz.unv*”. The command converts your Salome mesh into a Foam mesh. The point is the OpenFOAM case directory. xyz is your case and the last is the path to your unv-file.
- Now that you have the mesh in your Foam case, you can set up the boundary conditions. To do that double click on *Mesh* → *Patches* → *walls.*
- Then choose *fixedCharge* as boundary type and do the same for the torus.
- Then go to *Fields* → *phi* and enter a uniform potential value for the torus (remember: because SI Units are used you enter the potential in volts).
- Now you have to edit the dictionaries: Double click on *physicalProperties* and enter 8.85e-12 as value for epsilon0.
- Then open fvSchemes → *divSchemes* → *div(rhoFlux,rho)* → *interpolationScheme* and choose *limitedLinear* as scheme and close the windows.
- Finally open the *controlDict* to enter the timing settings.
- Choose an arbitrary *endTime* e.g. 5 and enough time steps so that a stable solution is generated. In this case 1 as value for *deltaT* should be sufficient.
- Now you can run the calculation by clicking on the play button (the one which says “Start Calculation” when you hover with you cursor over it).
- Click *Execute* and wait a little until the solution is generated.
- If it's done, you can view the log and check if everything worked correctly. When the energy doesn't change much from step to step you have a stable solution. That's the point to smile again!

5 Post processing

- Right click on you xyz case in the case browser and select *Foam Utilites* → *postProcessing* → *graphics* → *paraFoam* and execute it.
- Your case is being loaded into ParaView. Now it depends on what you want to view. If it is the potential field just click on the green “Accept” button. But if you want to view the electric field you have to disable the other fields first. To do that, click on “All Off” in the *Vol Field* section. Then select an other time step and click on “Accept”.
- Nevertheless you will most probably only see a blue box so far. To change that choose the clipping filter from the *Filter* menu, set the clipping plane in such a manner that you will see something after clipping and click on accept.
- Then you can change the view by holding the left mouse button pressed or zoom it by holding the right button pressed.
- Now think, whether the field distribution makes sense! If it does so, be happy - you just calculated the electric field of a torus in a box in three dimensions!

6 Advanced post processing

- Now perhaps you want to know the field strength along a specific line or the capacitance of the whole thing. First to the capacitance calculation: That's simple, because you already have the total energy of the field (view the log file) and the capacitance is in this case C = 2*energy/(applied voltage)^2.
- Hint: The numerical calculated capacitance differs from the analytical somewhat depending on the geometry. The reason is not clear. The differences I encountered so far where up to 20%. So be careful!
- To extract data you have to use the “sample utility”.
- Copy the file “sampleDict” from */home/caelinux/OpenFOAM/OpenFOAM-1.4.1/tutorials/solidDisplacementFoam/plateHole/system*

  to the system folder of your case which you can find in */home/caelinux/OpenFOAM/caelinux-1.4.1/run/xyz*
- Open it and change the *writeFormat* to “gnuplot” (or use the raw output and plot the data by the tool of your choice).
- Then you can enter the name of your plot, choose the coordinates for start and end point of your path and set the axis according to your path, e.g. if you set

  start (1 0 0);

  end (2 0 0);

  as path, it would be interesting to see the field over the x-axis.

- In “fields” you write “magE” instead of “sigmaxx”, if you want to see the magnitude of the electric field strength.
- Then close and save the file and go to the case browser in FoamX again.
- Right click on your case and select *Foam Utilities* → *postProcessing* → *miscellaneous* → *sample* to execute the script. The results are written in a new samples folder in your case directory for every time step.
- Right click on a time step folder and choose Actions Open Terminal Here
- Type “gnuplot” in the console.
- Then type “load 'xyz_magE.gplt' (replacing xyz with your name) and gnuplot will write the plot to the time step folder.
- That's all! I hope everything worked well and if there are any problems or recommendations feel free to send me a nice mail:

  andre.kunze@mailbox.tu-dresden.de